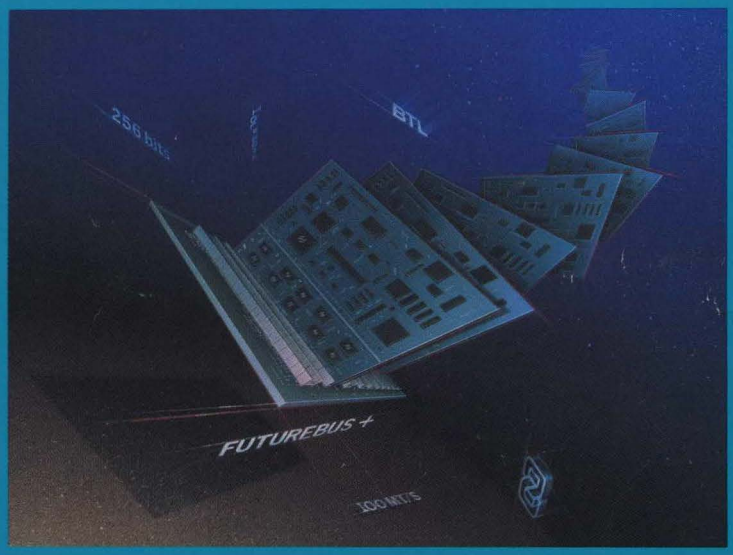
National Semiconductor

550095

Futurebus+, PI-Bus, BTL



High-Performance Bus Interface Designer's Guide

HIGH PERFORMANCE BUS INTERFACE DESIGNER'S GUIDE

1991 Edition

TRADEMARKS

Following is the most current list of National Semiconductor Corporation's trademarks and registered trademarks.

ABiC™
Abuseable™
Anadig™
ANS-R-TRAN™
APPS™
ASPECT™
Auto-Chem Deflasher™
BCP™
BI-FET™
BI-FET II™
BI-LINE™
BIPLAN™
BLC™
BLX™
BMAC™
Brite-Lite™
BSI™
CDD™
CheckTrack™
CIM™
CIMBUS™
CLASIC™
Clock✓Chek™
COMBO®
COMBO I™
COMBO II™
COPS™ microcontrollers
CRD™
DA4™
Datachecker®
DENSPAK™
DIB™
Digitalker®
DISCERN™
DISTILL™
DNR®
DPVM™
E²CMOS™
ELSTAR™
Embedded System Processor™
E-Z-LINK™
FACT™
FACT Quiet Series™
FAIRCAD™
Fairtech™
FAST®
FASTr™
5-Star Service™
Flash™
GENIX™
GNX™
GTO™
HAMR™
HandiScan™
HEX 3000™
HPC™
I³L®
ICM™
INFOCHEX™
Integral ISE™
Intelisplay™
ISE™
ISE/06™
ISE/08™
ISE/16™
ISE32™
ISOPLANAR™
ISOPLANAR-Z™
KeyScan™
LMCMOS™
M²CMOS™
Macrobus™
Macrocomponent™
MAPL™
MAXI-ROM®
Meat✓Chek™
MenuMaster™
Microbus™ data bus
MICRO-DAC™
μtalker™
Microtalker™
MICROWIRE™
MICROWIRE/PLUS™
MOLE™
MPA™
MST™
Naked-8™
National®
National Semiconductor®
National Semiconductor Corp.®
NAX 800™
Nitride Plus™
Nitride Plus Oxide™
NML™
NOBUS™
NSC800™
NSCISE™
NSX-16™
NS-XC-16™
NTERCOM™
NURAM™
OPAL™
OXISS™
P²CMOS™
PC Master™
Perfect Watch™
Pharma✓Chek™
PLAN™
PLANAR™
PLAYER™
Plus-2™
Polycraft™
POSilink™
POSitalker™
Power + Control™
POWERplanar™
QUAD3000™
QUIKLOOK™
RAT™
RTX16™
SABR™
Script✓Chek™
SCX™
SERIES/800™
Series 900™
Series 3000™
Series 32000®
Shelf✓Chek™
Simple Switcher™
SofChek™
SONIC™
SPIRE™
Staggered Refresh™
STAR™
Starlink™
STARPLEX™
Super-Block™
SuperChip™
SuperScript™
SYS32™
TapePak®
TDS™
TeleGate™
The National Anthem®
Time✓Chek™
TINA™
TLC™
Trapezoidal™
TRI-CODE™
TRI-POLY™
TRI-SAFE™
TRI-STATE®
TURBOTRANSCEIVER™
VIP™
VR32™
WATCHDOG™
XMOS™
XPU™
Z STAR™
883B/RETS™
883S/RETS™

GAL® is a registered trademark of Lattice Semiconductor.

PAL® is a registered trademark of and used under license from Advanced Micro Devices, Inc.

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation 2900 Semiconductor Drive, P.O. Box 58090, Santa Clara, California 95052-8090 1-800-272-9959 TWX (910) 339-9240

National Semiconductor

High-Performance Bus Interface Introduction

Dan Mansur

Backplane Transceiver Logic (BTL) is widely recognized as a future technology for the next generation computer architectures. Invented by National in 1985 to enhance the performance of backplane buses, BTL makes such buses as Futurebus+ and PI-Bus possible by solving the transmission-line problems that limit the speed of buses based on the earlier TTL (transistor-transistor logic) technology.

BTL devices provide today's densely populated backplanes with the increased speed and drive capability they require while minimizing any capacitive loading on the bus. Drivers use a Schottky diode in series with an open-collector driver output to isolate the drive transistor capacitance, minimizing bus loading.

National's BTL transceivers have several other advantages over their TTL counterparts that further reduce power consumption and improve system reliability. Most of the power savings come from a reduced voltage swing of 1V on the bus, which also reduces the crosstalk on the backplane. The BTL receivers also have precise reference thresholds (1.55V ±5%) that provide maximum noise immunity. In addition, BTL's incident-wave switching eliminates the settling-time delays that slow the performance of the TTL-based buses.

BTL is a proven, accepted interface technology for high-speed transmission. The IEEE P1114.1 standard is based on National's BTL technology. This BTL technology has fuel several new bus standards for which National has developed products:

Futurebus+ Chip Set

Our new Futurebus+ Chip Set includes five devices: two 9-bit Data Transceivers (latched and non-latched), a Handshake Transceiver, an Arbitration Transceiver. and an Arbitration Controller. See *Figure 1*.

The Futurebus+ standard is a scalable, multi-segmented bus architecture that permits the highest data-transfer rate possible with the technology available at the time of the design. Futurebus+ is capable of data widths from 32 bits to 256 bits, addressing widths of 32 and 64 bits, and a data transfer rate that scales from 25 megatransfers/second (MT/s) today to a projected 100 MT/s. See *Figure 2* for Futurebus+ Features.

National advanced BTL devices support all distributed and central arbitration implementations of the Futurebus+ architecture, as well as both Compelled and Non-Compelled (Packet) Mode of data transfer.

In either data-transfer mode the **DS3883** (unlatched) and/or **DS3886** (latched) transceivers receive and transmit Address/Data, Data, Parity Command and Status signals. In a 32-bit system, only six such devices are required, with each sending and receiving a byte of information and its associated parity bit. A 64-bit system requires just an additional four data transceivers per board. The 9-bit BTL Data Transceiver (unlatched or latched) will support the fault-tolerant requirements of Futurebus+ systems. Its architecture is divided between eight data bits and one parity bit, and the parity bit is used by the system for error detecting/checking and/or correction.

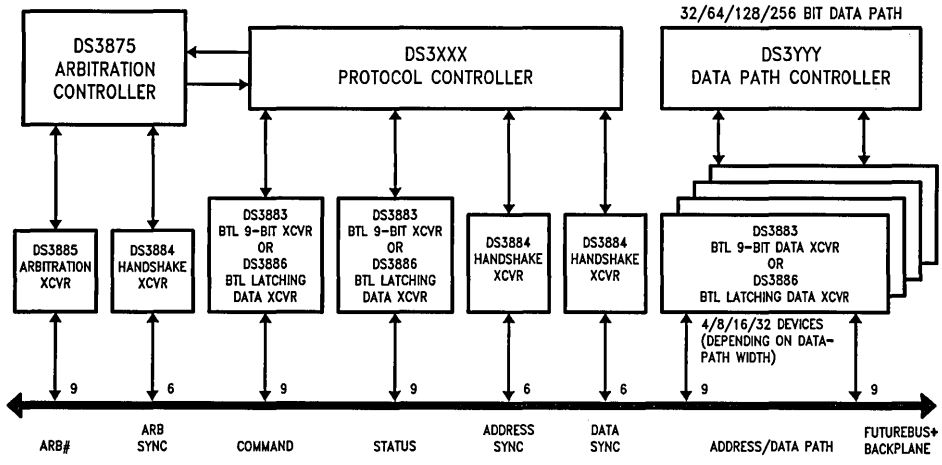


FIGURE 1. National's Futurebus+ Chip Set Diagram

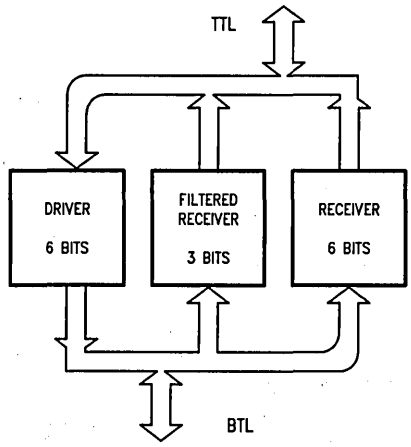
Feature	Futurebus+ Option	Technology Independence	Increased Bus Performance	Upgradeability	Increased System Reliability	System-Level Interoperability	IEEE Documentation
Bus-Driving Technology	BTL	X	X		X		IEEE P1194.1, P896.2
Data-Path Width	32 to 256 Bits (Data) 32 to 64 Bits (Address)		X	X			P8961.1, P896.2
Data-Transfer Protocols	Compelled Packet	X	X	X			P896.1,P896.2
Arbitration	Central and Distributed	X	X	X			P896.1
Fault Tolerance	Parity Protection Live Insertion Dual-Bus Support				X		P896.1, P896.2, P896.3
Message-Passing Protocol			X			X	P896.1
Full Cache-Coherence Support			X			X	P896.1
Profiles for Target Applications	(See List Below)		X		X	X	P896.2

Futurebus+ Profiles

Profile A: Cache-Coherent, System-Bus Architectures
Profile B: I/O Architectures/Applications
Profile D: Desktop Computer Applications
Profile F: High-Performance, Packet-Mode Computer Architectures
Profiles M1, M2 & M3: Military Applications
Profile T: Telecommunications Applications

FIGURE 2. Futurebus+ Features

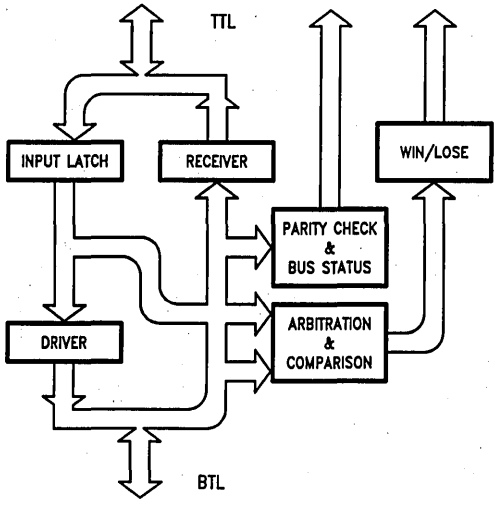
The **DS3884** Handshake Transceiver used Futurebus+ programmable glitch filters to handle signals that require filtering out glitches that result from wire-ORing of the bus signal lines. We offer both filtered and non-filtered receiver outputs to permit optimal performance for broadcast as well as single-slave transactions. Three DS3884s are used to handle Arbitration Synchronization, Address Synchronization and Data Synchronization signals.



TL/F/11162–2

FIGURE 3. DS3884, Handshake Transceiver

The **DS3885** Arbitration Transceiver handles all arbitration bus lines and implements Futurebus+ contention logic. It's designed to be used with our DS3875 Arbitration Controller. By partitioning the arbitration logic between two devices and integrating the competition logic into a BTL transceiver, we've created the fastest and most cost-effective hardware solution. The contention and parity logic are included within the DS3885, offering a 30% to 50% improvement in arbitration time over other methods.



TL/F/11162–3

FIGURE 4. DS3885, Arbitration Transceiver

The **DS3875** Arbitration Controller is capable of implementing all of the mandatory and optional requirements to the IEEE P896.1 Futurebus+ standard for distributed arbitration, as well as the central arbitration message-passing protocol. It interfaces with the bus through the handshake transceiver and the arbitration transceiver. In a single chip, the DS3875 implements the Futurebus+ arbitration function-set specified by the Acquisition, Allocation, and Alignment protocol. A wide range of other functions are also managed by the arbitration controller, including user-programmable arbitration timing delays, protocol timeouts and parity generation, among others.

Together, these devices in the Futurebus+ Chip Set offer several built-in advantages designed to satisfy the unique demands of high-speed data transmission through densely populated buses. All four transceivers are offered in 44-pin PLCC and space-saving 44-pin PQFP packages (40-pin TapePak® will also be available), and the Arbitration Controller is offered in a 68-pin PLCC package.

PI-Bus

The DS1776 PI-Bus Transceiver was designed to meet the low power requirements of the military by combining the companies leading BTL experience with an advanced BiCMOS process.

National's patented design and BiCMOS process enable the DS1776 to operate at approximately one-fourth the power of competing devices. The reduced power consumption is reflected in the worst-case current (I_{CC}) requirement of only 37 mA for the DS1776, compared with 145 mA for competing devices. This low power savings can offer up to 1A per board savings, reducing the concerns of avionics manufacturers regarding excessive power consumption in the limited space available.

Design Support and Next Generation Solutions

A variety of development tools for BTL and Futurebus+ are now available. To start with, this designer's guide provides several new BTL application notes and Futurebus+ system application notes. Spice Models and instruction manual of BTL Trapezoidal, Turbo, PI-Bus and Futurebus+ transceivers are available upon request along with Verilog® behavioral models.

In anticipation of higher frequencies and even more densely populated backplanes, National continues to enhance our BTL product line to provide smarter, faster, more highly integrated interface solutions the computer world demands. As a major contributor on the IEEE committee, National's identified several new products, such as standard protocol controllers for the most popular profiles and processors. In addition, several new products will be military-qualified for use in the Navy's Next Generation Computer Resources project, as well as several other programs.

National Semiconductor

Futurebus + Overview

David Hawley

A new standard computer bus with the muscle to match the speed of the next generation of 32-bit systems is about to bow. Now being balloted by the IEEE, the proposed P896 Futurebus+ standard promises a maximum data-transfer rate of better than 50 million transfers/s, a 500 percent improvement over current 32-bit buses. What's more, Futurebus+ will be extendable to 256 bits.

The new bus will offer a lot to system designers. Its extremely high data-transfer rate makes it attractive for high-performance I/O operations, such as FDDI or high-resolution graphics. The fine task scheduling provided by the arbitration protocol is a requirement for real-time systems.

Also, its cache coherence, message passing, and split-transaction support allow the design of efficient multiprocessing systems. The standard has generated significant technological advances throughout its long development, starting 10 years ago as the original Futurebus. These include the creation of Backplane Transceiver Logic to boost bus performance, the development of high-performance asynchronous and source-synchronous data-transfer protocols, and the formulation of a unified theory of cache coherence. It is currently being examined with great interest by the user community as a step beyond the current generation of 32-bit TTL standards, such as the VMEbus and Multibus II. As it has the possibility of becoming a universal standard bus, it deserves close consideration by anyone designing a backplane-based system.

The performance of Futurebus+ can be expected to vary from system to system, depending largely on the data-transfer mode supported. The asynchronous, full-handshake mode (similar to that of the old Futurebus) uses burst transfers and can be expected to peak between 20 million and 25 million transfers/s. A new source-synchronous mode should operate at over 50 million transfers/s with the next generation of silicon support. Because Futurebus+ supports data-path widths of 32, 64, 128, and 256 bits, a raw data-transfer rate of 1.6 Gbytes/s is conceivable. Even at 32 bits, the 200-Mbytes/s source-transfer rate is five times the peak of VMEbus or Multibus II.

The original Futurebus standard was designed by a small group of dedicated visionaries without major corporate backing. The P896 committee, formed by the IEEE in 1979, wanted to create a single industry-standard 32-bit bus for multiprocessing systems. By the time the standard was approved by the IEEE in 1987, though, the industry-designed VME and Multibus II buses had already established a firm hold in the marketplace.

At the same time, the performance of these buses was being stretched to the limit by the new generation of cache-based reduced-instruction-set processors. So, rather than risk obsolescence, the manufacturers of existing 32-bit systems looked for a new platform upon which to develop applications. Futurebus was the only high-performance standard that could be revised and extended to meet the latest system requirements. That's because there was no large base of products already designed to the Futurebus standard, and its specs were still pliable.

The designers of the original Futurebus had already anticipated many of the extensions, such as faster data transfer and the caching protocol, and so the process of revising the standard went fairly quickly. A number of the changes, however, were incompatible with the 1987 version, and so the new P896 standard was renamed Futurebus+. Currently in ballot are documents covering the mechanical, electrical, arbitration, data-transfer, and bus-management layers of the specification, as well as the caching and message-passing protocols.

Expected to follow in short order are documents on the use of Futurebus+ in real-time and high-availability systems and those that describe special requirements for industrial and military operating environments. Standard bridges are also being specified to VMEbus and Multibus II.

Futurebus+ has been endorsed by vendors of existing 32-bit buses, including the VME International Trade Association and the Multibus Manufacturer's Group. It also has been selected by the U.S. Navy as one of the standards for future computer contracts.

The high speed of Futurebus+ is due to backplane transceiver logic, which was first produced by National Semiconductor in 1984. BTL was designed specifically to drive backplane transmission lines and provides the fastest possible bus interface in a CMOS or TTL environment. Its characteristics are the foundation upon which the Futurebus+ protocol rests.

BTL devices use open-collector drivers with an output capacitance of only 5 pF (possible because the drive transistor is isolated from the bus by a series Schottky diode). This allows the combined connector, trace, and package capacitance to be limited to 10 pF for each slot, increasing the impedance of the backplane. BTL also operates with a reduced signal swing of only 1V and a precisely controlled switching threshold of 1.55V. The result is that the backplane can be properly terminated at its fully loaded impedance, while allowing the drivers to cleanly switch the bus signals with only 50 mA of drive current.

Reprinted by permission of High Performance System magazine.

With this interface technology, a bus designer can guarantee that a signal will cross the input thresholds of every receiver on the backplane on the incident edge of the propagating wavefront. A BTL bus never has to wait for reflections to settle before signals can be sampled. This allows Futurebus to implement much more efficient and high-performance data-transfer protocols than any TTL-based competitor.

Arbitration

The Futurebus+ specification carefully works out an arbitration procedure designed to optimize the scheduling of requests from multiple modules and to prevent more than one module from trying to transfer data on the bus at the same time. Futurebus+ provides a large number of priority levels for accurate real-time task scheduling, as well as a fairness protocol that allows an even allocation of bus bandwidth to multiple modules. The arbitration takes place on its own independent set of lines in parallel with transfers on the data bus. The Futurebus arbitration mechanism provides a number of other facilities, including error detection and recovery, parking, bus-master identification, emergency messages, and a live insertion-and-withdrawal mechanism for board replacement in high-availability systems.

A real-time system requires that task priorities be assigned accurately. This guarantees the deadlines of periodic system tasks, decreases the response latency of the system to asynchronous events, and ensures that critical tasks will be completed even under heavy system loads. In order to achieve a high degree of task scheduling, Futurebus+ provides up to 8 bits (256 levels) of priority, which can be assigned dynamically to all system tasks.

In priority arbitration, the competing module with the highest priority always wins, and there is no limitation on the frequency of bus requests. Those modules that are subject to real-time constraints can be assigned priorities based on the maximum latencies they can tolerate. The only drawback is that modules with low priority may be completely shut out during periods of heavy bus usage. However, the dynamic allocation capability makes it possible to increase the priority of a long-waiting task.

In most time-sharing multiprocessing systems, though, processors need approximately equal access to the bus. If a task has been divided among a number of processors, the optimum performance results when all the subtasks are completed at roughly the same time. Futurebus+ provides a round-robin fairness protocol that can operate within each priority level. Requests for the system bus at each level are serviced in the order of the competing module's unique ID field, typically based on slot position. The round-robin bit is set according to the ID of the most recent arbitration winner and serves to keep each requesting module's place in the circular queue. This scheme guarantees every module a fair slice of the overall bus bandwidth.

Futurebus+ allows two alternate implementations of this arbitration protocol. The first, a central arbitration protocol, uses two request lines and one grant line per module. These are connected in a star pattern to a central arbiter. The priority of each request can be modified to implement any of the protocols described above. Arbitration latency will be roughly 60 ns in this type of system.

The second, a distributed protocol is based on an asynchronous three-wire handshake. This bus handshake controls the state machines within each module as they request the bus, perform the actual arbitration, check for errors and wait for the current bus master to complete its tenure, and transfer ownership of the bus. Arbitration performance depends on the modules participating in the protocol, but can typically be expected to range from 150 ns to 350 ns with existing technology.

Most events in a Futurebus+ system are signaled with a virtual interrupt mechanism, requiring direct accesses to specific memory locations. There are no physical interrupt signals in a Futurebus+ backplane, so the arbitration protocol is used to fill this gap. Arbitration messages—special arbitration numbers that can be recognized by any module in the system—can be used to broadcast interrupts quickly, without first obtaining bus mastership or disturbing transfers in progress.

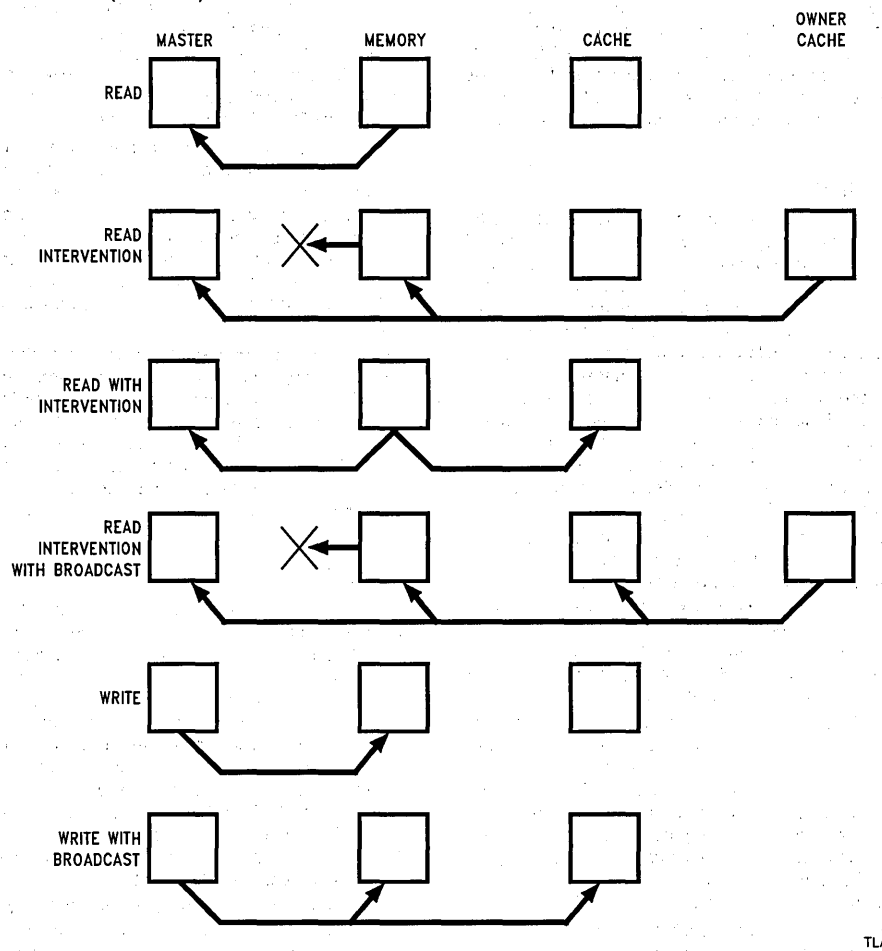
Data Transfer

Each transaction on Futurebus+ consists of a broadcast connection or address transfer, followed by one of a variety of data transfer types, and finally a broadcast disconnection. The connection phase is used to transmit addresses and commands from the master to the slaves, to return status to the master from the slaves, and for all participating modules to establish their data-transfer capabilities. Those modules that have been selected can participate in the data-transfer handshake, as can any caching modules that have chosen to "snarf" (induce data broadcast) or intervene. The disconnection phase is used to transfer information only during split transactions, when it provides the identity of the requester and the status of the response.

There are a number of transfer options that interact dynamically, providing a transaction set that supports applications ranging from the most basic to a multilevel-caching bus hierarchy. Transfers typically involve only the master and a single slave. However, because Futurebus+ was designed to allow multiple modules to maintain data coherence in shared memory environments, the standard also provides support for broadcast and intervention *(Figure 1)*.

Transactions also may be connected or split. The more typical is the connected transaction in which all data and status information associated with that transaction are returned before the address handshake is complete. A split transaction, in contrast, typically consists of two transfers separated in time.

Data Transfer (Continued)



TL/F/11131–1

FIGURE 1. Futurebus+ provides transactions beyond the basic read and write to memory. Efficient multiprocessor cache-coherence protocols require that the bus support cache-induced intervention and broadcast.

The first transfer, a request from the master to the slave, may include write data. The second transfer, generated by the slave, may include read data. Because both transfers are required, the split-transaction protocol is most useful during transfers across bus repeaters, where the data-access time can be much greater than the arbitration and address-transfer overhead.

The address/data path on Futurebus+ consists of 32 or 64 address/data lines on a single connector, with optional additional lines to support 128- or 256-bit data paths, and 8 user-definable tag bits. Each byte of the address/data highway is protected by a single odd parity bit. There is also a 8-bit command field (Table I), protected by parity, plus eight status lines and three capability lines (Table II).

Futurebus+ provides a special set of commands to support the higher level cache coherence and split-transaction protocols, as shown in Table I. A system that maintains the coherence of shared data among multiple modules requires that the master let other snooping caches know if it intends to keep a copy (share) of the addressed data or if it will write (modify) it. Likewise, the other caches must perform certain actions if they already have shared or modified copies of that data.

The Futurebus+ data-transfer protocol uses six synchronization lines. Three of these, the address-handshake lines, are used to establish and break a connection between a master and one or more slaves. The other three, the data-handshake lines, are used to transfer data or packets between the master and those slaves that have established the connection. In Futurebus+, information is usually transferred with every transition of these handshake lines.

Data Transfer (Continued)

Single-slave transactions involve only two modules and therefore have the most efficient data handshake *(Figure 2)*. However, if another slave has an active request for the data being transferred—such as a cache with a pending request for that data—it can snarf the transaction and turn it into a broadcast. In either case, the directly accessed slave may not have the most recent copy of the data in a cache system. The cache that has modified the data internally must then intervene in the transaction, providing the updated data to the master and the selected (and any snarfing) slave.

Futurebus+ also has two distinct data-transfer modes: a fully handshaken, asynchronous compelled transfer and a high-performance, source-synchronous packet transfer.

The compelled data-transfer protocol uses an asynchronous handshake. Information is transferred from the master to the slave(s) between the transition of the data strobe and the release of one of the data acknowledge lines. Information passes from the slave(s) to the master between that release and the next data-strobe transition. The transfer speed is controlled by all the participating parties, and it's limited by the round-trip handshake time—40 ns to 50 ns.

In the packet mode, the data handshake surrounds the transfer of an entire packet of data, and multiple packets can be transferred in a single transaction *(Figure 3)*. Each packet is transferred at a selected rate, synchronized to the source of the data. The transmit clock is embedded within the data on a bit-by-bit basis. Every packet consists of a sync bit, 8, 16, 32, or 64 NRZ data bits, and a parity bit that returns the data line to its original state. Because every bit is transmitted independently, there is no clock-skew timing penalty. The maximum transfer rate is probably limited by backplane physics at around 100 Mbit/s.

TABLE I. Command Lines

COMMAND LINES

32- OR 64-BIT ADDRESSES
32-, 64-, 128-, OR 256-BIT DATA
READ AND WRITE

WORD AND PARTIAL-WORD TRANSFERS
UNLOCKED AND LOCKED TRANSFERS

CACHE COMMAND SET

CACHE-SHARING TRANSFER
CACHE-MODIFYING TRANSFER
CACHE-COPYBACK TRANSFER
CACHE INVALIDATE

CACHE-SHARING RESPONSE
CACHE-MODIFYING RESPONSE

SPLIT-TRANSACTION COMMANDS

SPLIT-TRANSACTION RESPONSE
REMOTE TRANSFER WITHOUT RESPONSE

PACKET-SIZE SELECTION
ATOMIC PRIMITIVE OPERATIONS

TABLE II. Status and Capability Lines

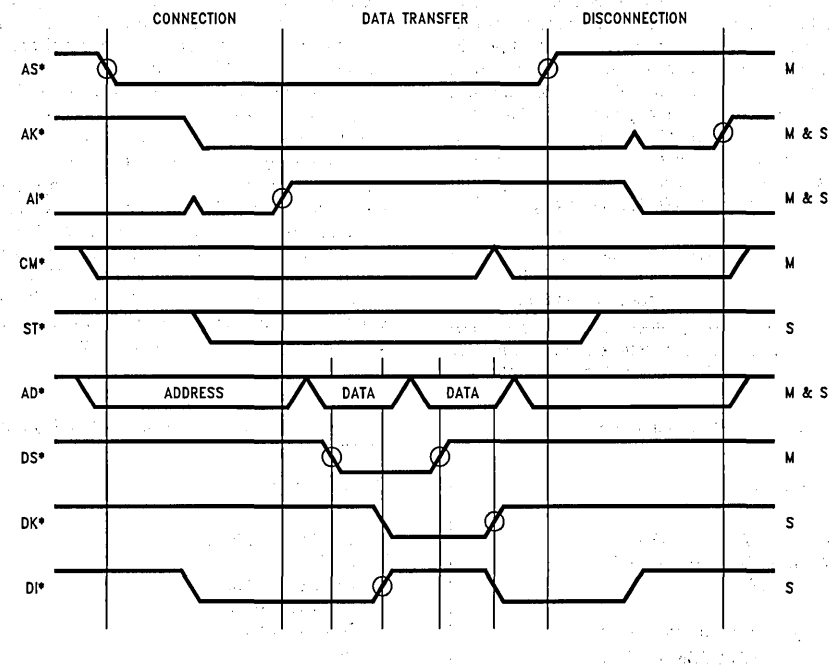
STATUS LINES

WAIT/END OF DATA
ERROR
INTERVENTION
CACHE SHARING
BROADCAST
SELECTED
BUSY
SYSTEM ERROR

CAPABILITY

SPLIT-RESPONSE TRANSFER
COMPELLED OR PACKET TRANSFER
PACKET TRANSMIT SPEED

Data Transfer (Continued)

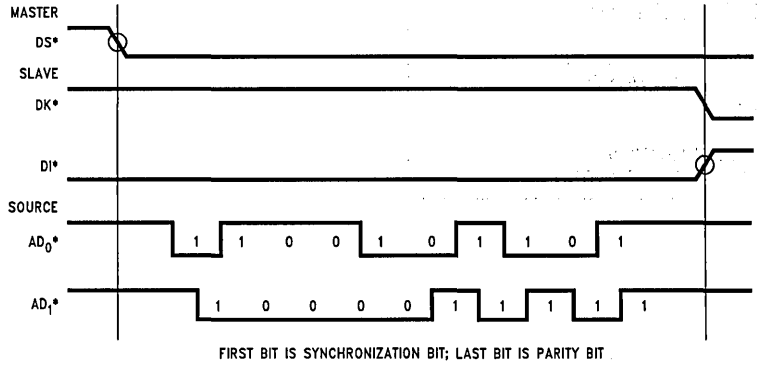


TL/F/11131–2

AS* = ADDRESS STROBE
AK* = ADDRESS ACKNOWLEDGE
AI* = ADDRESS ACKNOWLEDGE INVERSE
CM* = COMMAND
ST* = STATUS
AD* = ADDRESS/DATA
DS* = DATA STROBE
DK* = DATA ACKNOWLEDGE
DI* = DATA ACKNOWLEDGE INVERSE

M = MASTER
S = SLAVE

FIGURE 2. Following the connection, or address phase, of this Futurebus+ transaction, the master transfers two words to the selected slave, then releases the bus, possibly broadcasting additional information during disconnection.



TL/F/11131–3

FIGURE 3. Packet-mode transfers allow a block of data to be transmitted at a predetermined source-synchronous clock rate. Since the clock is embedded in each data bit, most traditional sources of skew are eliminated.

Cache Coherence

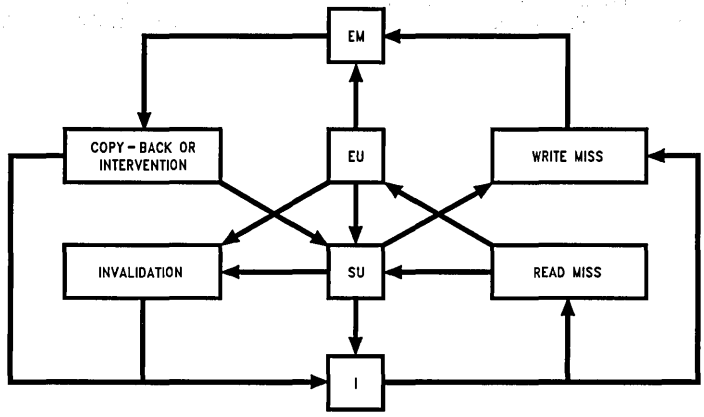
The Futurebus+ cache protocols allow this specialized memory to perform its three main functions automatically and completely transparent to the software. The first function is to convert a microprocessor's semirandom reads and writes into efficient burst transfers on the bus. The second is to provide the microprocessor with a fast local window into the system memory space. The third is to provide the basis for a multiprocessing architecture.

The original Futurebus cache task group developed the five-state MOESI cache-coherence model, the acronym coming from the five states—Modified, Owned, Exclusive, Shared and Invalid. MOESI was a superset of all previously known cache-coherence solutions, thereby allowing any combination of coherence protocols to coexist in the same backplane.

However, as the understanding of cache protocols improved, it became apparent that the complexity required to support the five-state MOESI model was not justified by the return in performance. So for Futurebus+, the group selected a four-state MESI copy-back protocol that can be generalized for caching over a hierarchy of buses using split transactions. Memory- and cache-agent pairs act as repeaters between processors on multiple buses accessing a single memory source.

In this cache-coherence protocol *(Figure 4)*, every processor-cache line has associated with it one of four states: invalid (I); shared unmodified (SU); exclusive unmodified (EU); and exclusive modified (EM). In order for a pocessor to read data out of its cache, the data must first be valid—in the SU, EU, or EM state. If the data is invalid—in the I state—the cache must read the correct data from the bus. For a processor to write data, the cache must first ensure that no other cache has a copy of it; in other words, the cache must obtain an exclusive copy of the data—the EU or EM states. Once the processor has modified data in the cache—so that it is in the EM state—the cache must intervene or copy back to provide the system with the correct data.

A cache must modify its state information, or tags, in response to internal-processor and external-bus accesses, according to a set of rules described in the P896 standard. (Bus repeaters have a slightly different set of responsibilities, also described). An action by one cache affects every other cache in such a way that a consistent view of shared data is maintained. Futurebus+ provides the transaction set necessary to implement this shared-memory system efficiently.



TL/F/11131–4

EM = EXCLUSIVE MODIFIED
EU = EXCLUSIVE UNMODIFIED
SU = SHARED UNMODIFIED
I = INVALID

FIGURE 4. In the four-state cache-coherence model, a processor has private read permission if data is in the EM, EU, or SU states; it has private write permission in the EM and EU states; and it has responsibility to intervene in the EM state.

National Semiconductor

Product Status Definitions

Definition of Terms

Data Sheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This data sheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This data sheet contains preliminary data, and supplementary data will be published at a later date. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
No Identification Noted	Full Production	This data sheet contains final specifications. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

National Semiconductor Corporation reserves the right to make changes without further notice to any products herein to improve reliability, function or design. National does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

Table of Contents

Alpha-Numeric Index

Section 1

Futurebus + /BTL Devices

Section 1 Contents

National Semiconductor

PRELIMINARY

DS3875
Futurebus+ Arbitration Controller

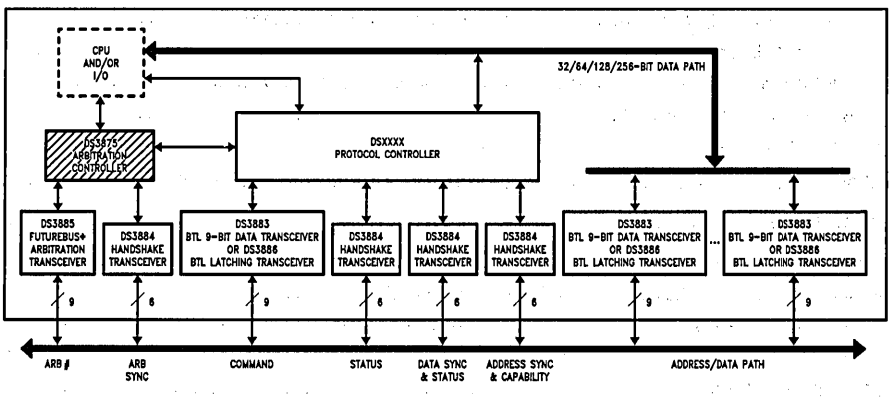
General Description

The Futurebus+ Arbitration Controller is a member of the chipset that National Semiconductor has designed for the IEEE P896.1 Futurebus+ standard. The DS3875 implements in a single chip the Arbitration function set specified by Acquisition, Allocation & Alignment protocol. This Controller interfaces with Futurebus+ through the NSC designed Futurebus+ Arbitration Transceiver and Handshake Transceiver. NSC's Futurebus+ Arbitration Transceiver implements the arbitration circuit. The Handshake Transceiver can be used to transceive the Arbitration Sequencing and Arbitration Condition signal lines. Also in the chipset to support the parallel protocol, two other Handshake Transceivers transceive the Address Synchronization and Data Synchronization lines. The BTL 9-bit Data Transceivers or the Latched Data Transceivers transceive the Address/Data, Data, Bus Parity, Command, Command Parity and Status signal lines.

Features

- The controller implements the complete requirements of the IEEE P896.1 specification
- Supports Arbitration message sending and receiving
- Supports the two modes of operation (RESTRICTED/UNRESTRICTED)
- Software configurable double/single pass operation, slow/fast, IBA/Parking and restricted/unrestricted modes of arbitration
- Built-in 1 μs timer for use in the arbitration cycle
- User programmable 16 arbitration delays (8 slow and 8 fast)
- Built-in PLL for accurate delays. The PLL accepts clocks from 2 MHz to 40 MHz in steps of 1 MHz
- Signal to unlock slave modules on transfer of tenure. Auto unlock through a dummy cycle if the current master locked resources
- Programmable delay for releasing ar* after issuing COMPETE/IBA_CMPT. This is to ensure the assertion of the arbitration number during competition, before the release of ar*. Also this delay ensures there is sufficient time to assert the AD/DATA lines during Idle_Bus Arbitration before the release of ar*
- Read/Write facility with data acknowledge for the host to load arbitration numbers, an arbitration message, and control registers
- On chip parity generator unloads the host of the additional parity generation function
- Separate interrupts to indicate error occurrence and arbitration message received. Interrupts cleared on a register write. Error status is available in a separate status register
- A special output pin to indicate that a POWERFAIL message was received
- Hardwired register to hold the first word of the arbitration message
- FIFO strobe provided to store more than one arbitration message externally to prevent overrun
- Idle Bus Arbitration (IBA) supported
- Parking implemented
- Bus initialization, system reset and Live-insertion supported. (The logic to detect these conditions must be implemented externally.)
- Testability in the form of reading from key registers which include the STATE, MCW, 1 μs timer and programmable input clock divider

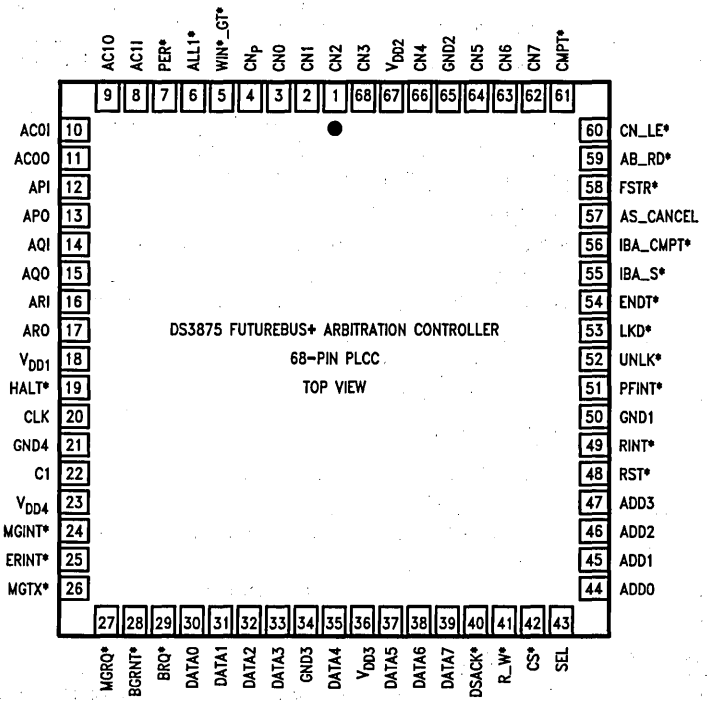
National Semiconductor Futurebus+ Transceivers and Arbitration Controller



TL/H/10747-1

Table of Contents

Table of Contents (Continued)



TL/H/10747-2

Pin Definition

Pin	# of Pins	Type	Description
SIGNAL TO/FROM THE HANDSHAKE TRANSCEIVER			
APO	1	O	Arbitration handshake signal from the controller.
AQO	1	O	Arbitration handshake signal from the controller.
ARO	1	O	Arbitration handshake signal from the controller.
AC0O	1	O	Arbitration condition signal from the controller.
AC1O	1	O	Arbitration condition signal from the controller.
APl	1	I	Arbitration handshake signal from Futurebus+. This signal is the filtered and inverted version of the Futurebus+ backplane signal AP*.
AQl	1	I	Arbitration handshake signal from Futurebus+. This signal is the filtered and inverted version of the Futurebus+ backplane signal AQ*.
ARl	1	I	Arbitration handshake signal from Futurebus+. This signal is the filtered and inverted version of the Futurebus+ backplane signal AR*.
AC0l	1	I	Arbitration condition signal from Futurebus+.
AC1l	1	I	Arbitration condition signal from Futurebus+.
AS_CANCEL	1	I	Indicates the start of the disconnection phase of the current master or cancel the current arbitration cycle.
SIGNAL TO/FROM THE ARBITRATION TRANSCEIVER (Note: These pins are mapped to/from the DS3885 Futurebus+ Arbitration Transceiver.)			
CN(7:0)	8	I/O	The bus to carry competition number to/from the arbitration transceiver.
CNp	1	O	Parity bit of the competition number.
CMPT*	1	O	Enables the Arbitration number onto Futurebus+.
AB_RD*	1	O	Direction control for the competition number bus to/from the transceiver.
CN_LE*	1	O	Latch enable for latching the Arbitration number from the controller into the transceiver.
PER*	1	I	**PARITY ERROR:** Indicates that a parity error was detected on the winner's arbitration number.
WIN*_GT*	1	I	Win signal when competing/greater than signal when not competing (used to preempt).
ALL1*	1	I	Indicates that all the arbitration number lines on the bus are asserted (used for messages).
SIGNALS TO/FROM THE PARALLEL PROTOCOL CONTROLLER			
UNLK*	1	O	**UNLOCK:** Transfer of tenure indication to the parallel protocol controller for unlocking its resources. Generated only if the LKD* signal is asserted.
ENDT*	1	I	**END OF TENURE:** Indicates the true end of bus tenure of the current master. This line may be asserted only after all the parallel protocol lines are released.
LKD*	1	I	**LOCKED:** Signal to indicate that resources have been locked in the current tenure and hence generate either a dummy cycle if current master or UNLK* otherwise.

Pin Definition (Continued)

Pin	# of Pins	Type	Description
SIGNALS TO/FROM THE PARALLEL PROTOCOL CONTROLLER (Continued)			
IBA_CMPT*	1	O	Signal to indicate that the Parallel Protocol controller may assert its bit on the ADDRESS/DATA bus if it is participating in an Idle Bus Arbitration.
IBA_S*	1	I	This signal indicates that IBA was successful. If this module was a competitor in the IBA competition (IBRQ*), then this module is the winner and now the bus master. If this module was the master, but did not compete in the IBA competition and IBA was successful, then the M bit (Status register) is negated.
SIGNALS TO/FROM THE HOST			
DATA(7:0)	8	I/O	Data bus for the host to access the register bank of the controller.
ADD(3:0)	4	I	Address bits for the register bank of the controller.
CS*	1	I	**CHIP SELECT:** The host can read or write to/from the controller.
R_W*	1	I	Read/write signal from the host.
DSACK*	1	O	Data acknowledge pin for host read/write.
SEL	1	I	**SELECT:** Determines how the controller latches in data. A "1" on the pin uses the rising edge of CS*. A "0" on the pin uses the falling edge of DSACK*.
BRQ*	1	I	**BUS REQUEST:** Indicates to the controller to acquire the bus for the module's use.
BGRNT*	1	O	**BUS GRANT:** Signal asserted by the controller after the detection of a bus request. The module can start using the bus.
MGRQ*	1	I	**MESSAGE REQUEST:** Indicates to the controller to send an arbitration message.
MGTX*	1	O	**MESSAGE TRANSMIT:** Indicates the successful transmission of an arbitration message.
ERINT*	1	O	**ERROR INTERRUPT:** Indicates that an error occurred during the arbitration cycle.
MGINT*	1	O	**MESSAGE INTERRUPT:** Indicates the reception of an arbitration message.
PFINT*	1	O	**POWER FAIL INTERRUPT:** Indicates that a powerfail message was received.
FSTR*	1	O	**FIFO STROBE:** Signal generated to load an external FIFO for received arbitration messages.
HALT*	1	I	Will halt the arbitration controller in phase 0. This signal is for use during live insertion.
RINT*	1	I	Will put the arbitration controller in phase 0 and release all the bus lines except AR*. A selective reset is performed. The rising edge will release controller from phase 0. This reset is to be used for bus initialization.
RST*	1	I	Reset signal from the host. An internal reset is performed. All bus signals are released. The rising edge will put the controller in phase 0 (same as power-up reset).
CLK	1	I	Clock input to the internal PLL.
C1	1	I	External capacitor input for PLL—0.1 μF.

1.0 Introduction to Futurebus+

Futurebus+ is a high-performance asynchronous multiplexed address/data backplane bus designed by the IEEE 896.1 committee for use in a wide variety of multiprocessor architectures. The Futurebus+ standard is a next generation backplane bus standard developed to a set of requirements including openness, performance, and system facilities and flexibilities so as not to hinder systems using this bus for many generations of computer systems. Futurebus+ is a single cache coherent backplane architecture featuring scalability, technology independent protocols, and explicit provisions to extend to future applications. Requirements set for the Futurebus+ architecture by the IEEE 896.1 Futurebus+ Working Group include:

1. Architecture, processor, and technology independent
2. A basic asynchronous (compelled) transfer protocol
3. An optional extended source-synchronized protocol
4. No technology-based upper limit to bus performance
5. Fully distributed parallel and arbitration protocols
6. Support for fault-tolerant and high-availability systems
7. Support for cache-based shared memory
8. Compatible message transport definition.

Compatibility of varying speed and technology boards connected to the same Futurebus+ backplane is dependent upon broadcast capability, or a snooping cache coherence scheme. The performance of a Futurebus+ backplane is scalable over time to allow a low-end 32-bit system operating at 100 Mbytes/sec to be expanded to a 256-bit system operating at 3.2 Gbytes/sec in the future. This performance is attainable by the use of source-synchronized protocols which eliminate spatial skews and receivers which are triggered by the incident wave from the driver. The synchronization protocol of the Futurebus+ backplane allows the synchronization domain of the sender to extend along the backplane, presenting only one synchronization interface between the bus and the receiver.

The Futurebus+ backplane uses "Backplane Transceiver Logic" (BTL). BTL is a signaling standard that has been developed to enhance the performance of backplane buses. This standard eliminates the settling time delays, that severely limit the TTL bus performance, to provide significantly higher bus transfer rates. BTL compatible transceivers feature low output capacitance drivers to minimize bus loading, a 1V nominal signal swing for reduced power consumption, and receivers with precision thresholds for maximum noise immunity. For example, all Futurebus+ signals are open collector with termination resistors (selected to match the bus impedance) connected to 2V at both ends. The low voltage is typically 1V. All Futurebus+ signals are active low, indicated by an * after the signal name. (Refer to Table I.) Further, signals can be driven simultaneously by several modules. This requires that glitch filtering will be needed for those times to filter out the transmission line effect called the wire-or glitch. Refer to the Futurebus+ specifications for details.

TABLE I. Signal Definitions

Signal Type, Example Signal	Terminology	Logic Level	TTL	BTL
Active High Signal_Name	Asserted	Logic 1	5V	—
	Negated	Logic 0	0V	—
Active Low Signal_Name*	Asserted	Logic 1	0V	1V
	Negated	Logic 0	5V	—
	Released (Open Collector)	Logic 0	—	2V

2.0 Introduction to Futurebus+ Arbitration

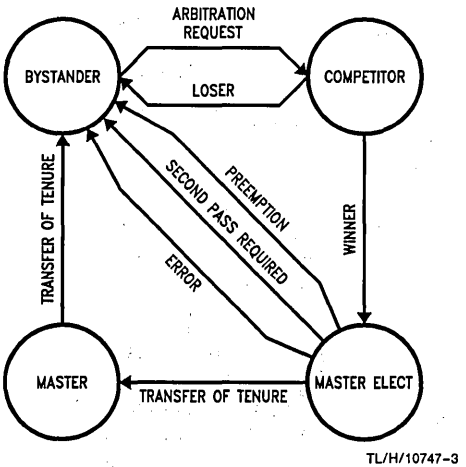
Futurebus+ uses an evolved version of the Parallel Contention Arbiter (see *Figure 4*). Through the application of a unique arbitration vector to this logic, only one contender will be uniquely selected as the winner. This implementation requires no central logic on the backplane and gains the following advantages:

1. The current master can see any requests and their priority to determine whether it should give up the bus.
2. Multiple priority levels can be implemented to allow systems to allocate bus bandwidth to modules running the most critical tasks.
3. A Round Robin (fairness) protocol is implemented within each priority level to ensure fair and equitable allocation of bus tenure to all modules.
4. A master elect can be preempted by a higher-priority contender. This allows the higher priority contender to access the bus with minimum latency (with some sacrifice to the system performance).
5. Arbitration Messages or events can be broadcast on the arbitration bus without disturbing the current transaction on the parallel bus. Important system control functions and interrupts can be sent using this mechanism without the need of dedicated bus lines.
6. A Module may support either Idle Bus Arbitration or Parking. Idle Bus Arbitration can be enabled by the current master to decrease the arbitration latency when ony one competitor is requesting the bus. If Idle Bus Arbitration is not selected then Parking may be enabled by the master. Parking allows the current master to regain bus tenure (during Phase 0) to perform new bus transactions.

2.0 Introduction to Futurebus+ Arbitration (Continued)

2.1 THE ARBITRATION STATES

Figure 1 is an Arbitration State Diagram showing four states that a module may enter as well as the events that cause the module to change states during the arbitration process. Transitions from one state to another occur only when certain conditions are met, based on the arbitration phase and the arbitration bus lines. Once a control acquisition cycle has started, all modules must participate in the arbitration to remain synchronized within the system.



TL/H/10747–3

FIGURE 1

3.0 Introduction to DS3875 Arbitration Controller

The DS3875 Arbitration Controller implements the complete requirements fo the IEEE 896.1 specifications. For example, both arbitration modes of operation (Unrestricted and Restricted) are supported. Also, either Idle Bus Arbitration or Parking may be selected for cases when only one request or no request for bus competition exist. The controller is software configurable to operate as a slow/fast module. This selects the minimum time the arbitration controller must wait during the arbitration competition before it can read the resulting competition status. This delay allows arbitration competition lines (AB[7 . . . 0]*, ABP*) to settle due to several arbitration numbers being applied to them (Wire-ored bus lines). Further, a built-in PLL, which accepts a clock signal from 2 MHz to 40 MHz, in steps of 1 MHz, controls several programmable delay lines used for releasing ar* after issuing CMPT*/IBA_CMPT* (PS(1:0)). This delay compensates for the chip to chip skew to ensure sufficient time to drive the arbitration competition number onto the arbitration competition lines during normal arbitration; or to assert the AD/DATA lines during Idle Bus Arbitration before releasing ar*. Internally, the PLL also generates the 1 μs timer used during arbitration phase 2 and phase 4.

The Arbitration Controller supports Arbitration message sending. A FIFO strobe, FSTR*, is provided to store more than one arbitration message externally to prevent overrun. Also a hardwired register contains the first word of the arbitration message (h'1ff). Additional registers which hold arbitration numbers, status information, controller configuration information, and an arbitration message can be accessed by the host through Read/Write operations. For outgoing arbitration numbers, the on chip parity generator unloads the host of the parity generation function. For incoming arbitration numbers, the DS3885 Arbitration Transceiver performs the parity check function and drives the PER* input signal to the Controller. Separate interrupt pins for error occurrence, reception of an arbitration message, and reception of Powerfail message are available. These interrupts are cleared by performing a dummy write cycle to these registers.

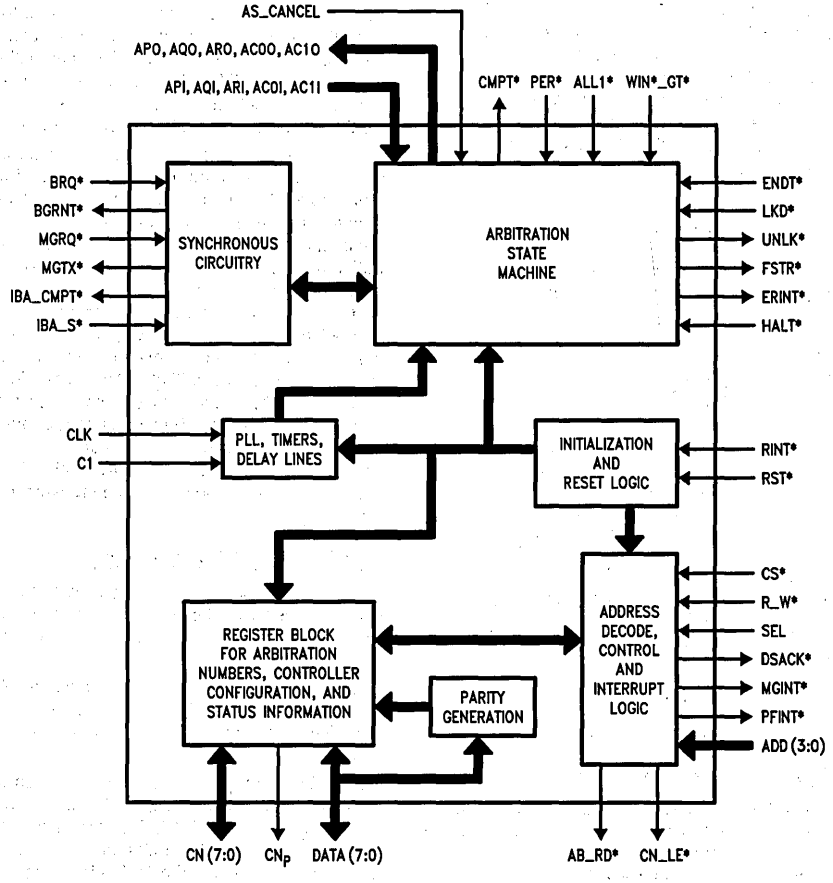
4.0 DS3875 Interfaces

The Arbitration Controller interfaces with the DS3884 Handshake Transceiver, DS3885 Arbitration Transceiver, host with other support chips, Protocol Controller, and Reset and Initialization logic.

Figure 2 depicts an internal block diagram of the DS3875 Arbitration Controller. *Figure 3* shows the interface between the DS3875 Arbitration Controller and the other logic on the module.

The DS3884 Handshake Transceiver drives the arbitration handshake and condition signals to the arbitration controller (API, AQI, ARI, AC0I, AC1I). The Arbitration Controller continuously monitors the Futurebus+ backplane through these signals. Whether the Arbitration Controller is competing in the current control acquisition cycle or not, it drives the arbitration handshake and condition signals (APO, AQO, ARO, AC0O, AC1O) which the handshake transceiver drives onto the Futurebus+ backplane.

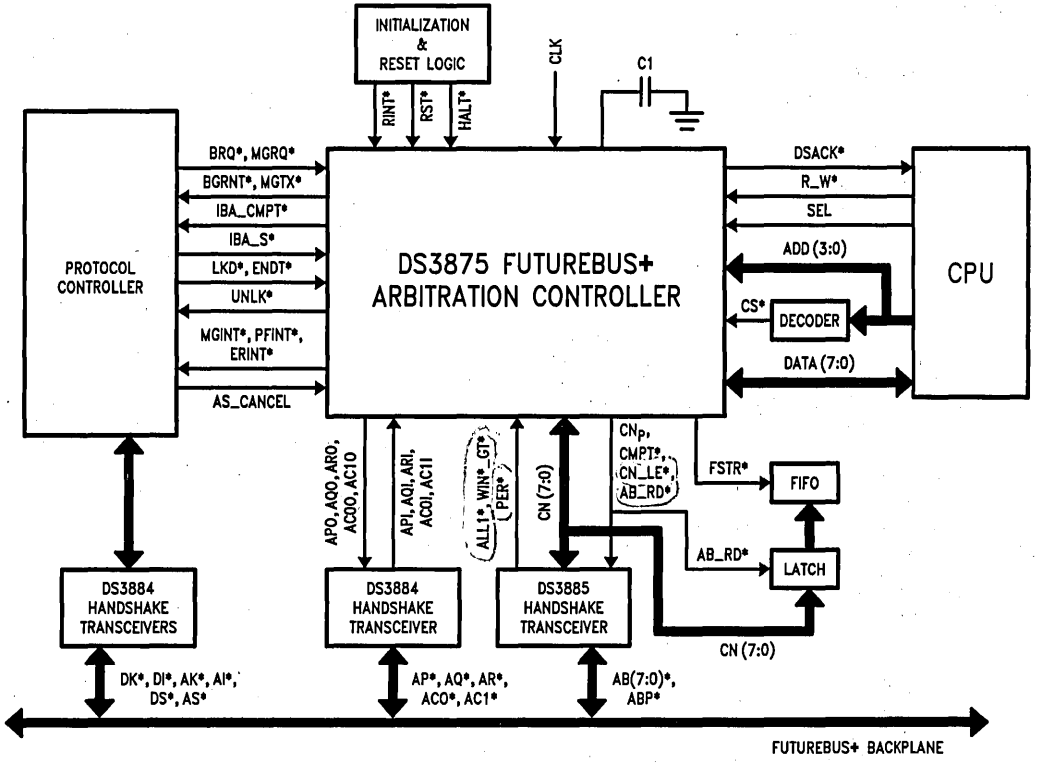
4.0 DS3875 Interfaces (Continued)



TL/H/10747–4

FIGURE 2

4.0 DS3875 Interfaces (Continued)



TL/H/10747–5

FIGURE 3

When competing the DS3885 Arbitration Transceiver is enabled to place the competition numbers CN(7 . . . 0) and its associated parity bit, CNp, onto the Futurebus+ backplane. During every cycle, whether or not competing, the winning module's arbitration number is read, the value of WIN*_GT* signal and PER* signal is determined and updated in the appropriate internal registers.

The host can read or write (R/W) into the Arbitration Controller registers to change the controller configuration, check status, or R/W a new arbitration number/message. The host can select to latch data either on the falling edge of DSACK* or on the rising edge of CS* by releasing or asserting the SEL pin.

The Module may become bus master during Phase 5 if the normal arbitration cycle was successful, Phase 0 if Parking was successful, or Phase 2 if IBA was successful. Upon becoming bus master, the Arbitration Controller will issue BGRNT*. This signal indicates to the Protocol Controller that it can now perform the desired transactions on the Parallel address/data bus.

The Protocol Controller will let the Arbitration Controller know if it has locked resources by asserting the LKD* signal. If resources were locked, then at transfer of tenure, the Arbitration Controller will issue UNLK* to the Protocol Controller to unlock resources.

A dummy cycle will be initiated by the Arbitration Controller to perform the unlocking function if lock (LKD*) is still asserted after end of tenure (ENDT*) is asserted and no other modules are competing. Unlock will be asserted at the transfer of bus tenure (even if this module wins the competition). If another module initiates arbitration competition this module will participate in the competition and will issue the unlock (UNLK*) signal upon transfer of bus tenure. When the bus transaction is complete and no resources are locked, the Protocol Controller has the option of enabling either Idle Bus Arbitration or parking (IBA_PK* in CTRL3).

4.0 DS3875 Interfaces (Continued)

The input signal ALL1* (AB[7 . . . 0]* and ABp* all asserted) is used to determine if any module is sending an arbitration message during pass 1. For convenience, the Arbitration Controller outputs FSTR* (to an external FIFO) during message reception so more than one arbitration message may be stored by the module.

The Arbitration Controller has dedicated interrupt pins (ERINT*, MGINT*, PFINT*) that interface with the Protocol Controller so that important messages and error indications can be quickly detected.

The Protocol Controller will issue the message request or bus request signals to the DS3875. When a message has been transmitted (Second Pass of arbitration, Phase 5), the Arbitration Controller will assert MGTX*.

On board reset, initialization, power-up, and live insertion logic will inform the Arbitration Controller which type of reset operation to perform: Power-Up Reset (RST*), Initialization (RINT*), or live insertion (HALT*). See Sections 10.0 and 11.0 for more information.

5.0 Arbitrating for Futurebus+

The arbitration process allows a module to seek and gain tenure of Futurebus+ to transfer data to or from another module. The arbitration process is independent of the data transfer process and may take place concurrently with data transactions on Futurebus+. If a module (or several modules) want to use the bus, an arbitration competition takes place. The module with the highest arbitration number gets tenure of the bus.

The National Semiconductor solution to Futurebus+ arbitration includes the DS3875 Arbitration Controller, the DS3885 Arbitration Transceiver and the DS3884 Handshake Transceiver (see front page system block diagram). More information on Arbitration as it applies to the Futurebus+ IEEE standard is available in Section 5 of the **"Futurebus+ P896.1 Logical Layer Specification"**.

5.1 UNRESTRICTED/RESTRICTED MODES OF OPERATION

The Arbitration Controller supports either the Unrestricted or the Restricted mode of arbitration. In the system environment, all modules must be configured to operate in the same mode at any given time.

During initialization, the Unrestricted mode is set since it must be supported by all modules. The Unrestricted mode allows a single pass of an 8-bit arbitration number or a two pass of a 16-bit arbitration number to be used.

Futurebus+ allows arbitration numbers requiring a single pass control acquisition cycle to be mixed with those requiring a two pass control acquisition cycle. During the first pass of a two pass cycle, more than one module may have the same number; however, during the second pass only one winner results where the competition numbers must be unique. A logic zero in the RU__ bit of the CTRL2 register selects the unrestricted mode.

The Restricted mode of operation is optional and is selected by setting the RU__ bit to a logic one in the CTRL2 register. This mode limits arbitration numbers to 8 bits. Thus, only a single control acquisition cycle occurs where all numbers are unique. The arbitration numbers are assigned by the module and can be dynamically changed.

5.2 THE ARBITRATION NUMBER AND ARBITRATION CIRCUIT

Each module has a unique arbitration number. When two or more modules compete for the bus, the module with the highest arbitration number will win the competition.

The DS3885 Arbitration Transceiver contains the arbitration circuit. See *Figure 4* for a functional model of the arbitration circuit. A Parallel Contention Arbitration Protocol controls how modules assert and release the AB[7 . . . 0]* and ABp*. After a period of time (ta) the protocol ensures that only the winners arbitration number will remain on the AB[7 . . . 0]* and ABp*.

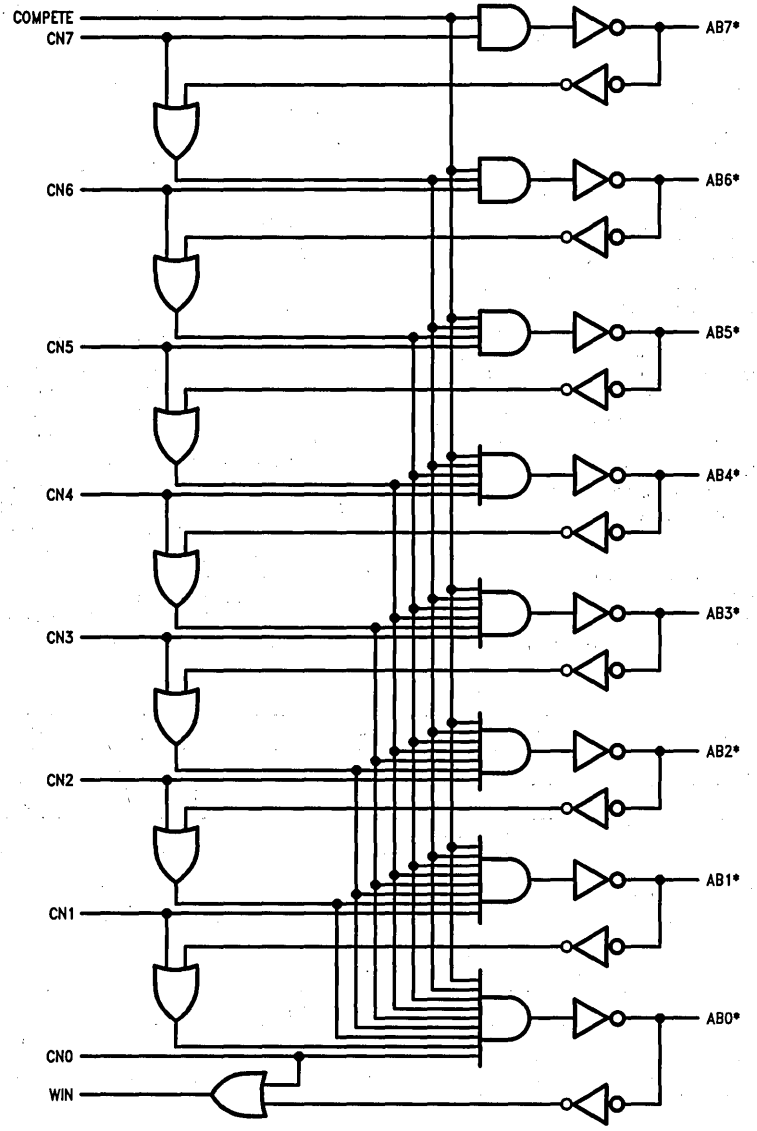
The arbitration number consists of one or two competition numbers, CN(7 . . . 0). The Arbitration Controller Transmits/Receives the competition number to/from the Arbitration Transceiver. The CN__LE* signal latches the competition number into the transceiver while the AB__RD* signal allows the controller to read in the winner's competition number. Along with the competition number, the Arbitration Controller transmits the CNp bit, the generated parity bit of the competition number, to the Arbitration Transceiver. Odd parity, as specified in the Futurebus+ specifications, is implemented. Thus, CNp is set when even number of ones are present in the competition number. When CN(7 . . . 0) is received from the Arbitration Transceiver, the PER* bit is checked for error detection.

Referring to Table II, in the Unrestricted mode, the CN7 bit indicates if the module requires another arbitration pass. When a one pass and a two pass arbitration number occur in the same control acquisition cycle, the one pass arbitration number will win.

The DS3875 allows the module to dynamically change the arbitration number by writing into the TXCN0 and/or the TXCN1 registers (see Section 7). If the arbitration number is changed during an arbitration cycle it will be used in the next arbitration competition.

The arbitration number is composed of three fields: Priority Field, Round Robin Field, and a Unique Field.

5.0 Arbitrating for Futurebus+ (Continued)



TL/H/10747–6

FIGURE 4. Futurebus+ Functional Model of the Arbitration Circuit

5.0 Arbitrating for Futurebus+ (Continued)

5.2.1 Priority Field (PR)

The priority field represents a module's priority class which is determined by the system designer. In the unrestricted mode, the length of this field is a maximum of eight bits while in the restricted mode it is two bits.

5.2.2 Round Robin Field (RR)

This field is represented by a single bit, CN5, in both modes of operation. The round robin protocol ensures that all modules in the same priority class have fair and equal access of the bus. A module is allowed to get tenure of the bus in a sequence defined by its value in the unique field.

The round robin bit is adjusted by the Arbitration Controller each time a transfer of tenure occurs in the module's same priority class whether or not the module is competing in the control acquisition cycle. In the module's same priority class, the round robin bit is set when tenure of the bus is transferred to a module with a larger unique field value or the round robin bit is cleared when tenure of the bus is transferred to a module with a lesser unique field value.

In the event, where the module's arbitration number is changed, after the next control acquisition cycle, the round robin bit is adjusted accordingly.

5.2.3 Unique Field (U)

The five bit unique field, CN(4 . . . 0), guarantees the uniqueness of the module's arbitration number. The unique number may correspond to the module's geographical address or may be allocated by the system designer as he chooses while also in such a way that minimizes the arbitration settling time. The value "11111" is not allowed in systems using arbitration messages.

5.3 ARBITRATION CATEGORIES

A module can be in one of six categories when a control acquisition cycle is in progress;

- Competitor for the Parallel bus
- Competitor to send a message
- By-Stander
- By-Stander who decides to invoke Pre-emption
- Master
- Master Elect

5.3.1 Competitor for the Parallel Bus

A Module may become a competitor for the parallel bus if it issues a Bus Request (I BRQ*) to the arbitration controller before the arbitration cycle Phase 1 starts, see *Figure 5a.*

If the module issues a bus request and the arbitration cycle is in phase 0, the controller will assert APO to start an arbitration competition. If an arbitration competition has already started, and the module's bus request comes prior to API being asserted on the arbitration bus, the module may enter this arbitration cycle. If an arbitration competition has already started, and the module's bus request comes after AP* being asserted on the arbitration bus, the module will have to wait until the next arbitration cycle to compete (or pre-empt the current arbitration cycle in Phase 3).

TABLE II. Arbitration Number

Bit	CN7	CN6	CN5	CN4	CN3	CN2	CN1	CN0
Unrestricted Mode—Single Pass								
Pass 1	1	PR0	RR	U4	U3	U2	U1	U0
Unrestricted Mode—Two Pass								
Pass 1	0	PR7	PR6	PR5	PR4	PR3	PR2	PR1
Pass 2	1	PR0	RR	U4	U3	U2	U1	U0
Restricted Mode								
Pass 1	PR1	PR0	RR	U4	U3	U2	U1	U0
Arbitration Message								
Pass 1	1	1	1	1	1	1	1	1
Pass 2	a7	a6	a5	a4	a3	a2	a1	a0

5.0 Arbitrating for Futurebus+ (Continued)

5.3.2 Competitor to Send a Message

The Arbitration Controller supports message sending. Message sending is implemented in the unrestricted mode in a two pass arbitration cycle. The first pass of all ones (1FF H) in the competition number identifies the transaction as a message. The ALL1* input signal is asserted by the Arbitration Transceiver when it detects 1FF H on the AB[7 . . . 0]* and ABp* lines. The arbitration controller has a hardwired register that holds the first pass word. Further, a dedicated pin MGRQ* (message request) places the Arbitration Controller in the message sending mode.

The message that is to be transmitted is loaded into the TXMSG register. The message of 1FF H is reserved as the powerfail message. Other messages are to be coded by the system designer with the greater priority messages having a higher arbitration number. Obviously, if more than one module simultaneously desires to send a message, the message with the highest arbitration number will be transmitted.

When a message is sent, no transfer of tenure takes place. Thus, the master (M) or the round robin (RR) bits are not updated. Upon successfully transmitting the message, MGTX** signal is asserted.

When a message is received by the Arbitration Controller (Phase 5), message interrupt (MGINT*) is asserted and FIFO Strobe (FSTR*) is negated. In the case of a Powerfail message being received; the PFINT* signal is asserted. When the PFINT* signal is asserted, the MGINT* and FSTR* signals are not generated. See *Figure 5d*.

When sending a message, once the MGRQ* signal is asserted, until the message has been transmitted, all other requests are blocked. Upon the MGTX* signal being released, the message request signal will be reevaluated to see if another message needs to be sent.

If this module is the module sending the arbitration message, then the message interrupt (MGINT*) or the Powerfail interrupt (PFINT*) will not be generated on this module. These interrupts are generated only upon the reception of a message from another module. Refer to *Figure 5e*.

5.3.2.1 Using an External FIFO to Store Messages

The Arbitration Controller provides a FIFO strobe (FSTR*) signal to store more than one arbitration message in an external FIFO. A rising edge on FSTR* is generated upon the reception of an arbitration message.

See *Figure 3* and Timing Diagrams: T6, Phase 2 and Phase 5.

1. FSTR* is always asserted (I FSTR*) during phase 2.
2. FSTR* is negated (FSTR*) during phase 5 given the following conditions.
   1. The message is not being sent by this module.
   2. This is the second pass of an arbitration message.
   3. No errors occurred during this arbitration cycle.
   4. This is not a powerfail interrupt message (I PFINT*).

The external latch shown in *Figure 3* is enabled by AB__RD* to temporarily hold the message. While AB__RD* is low (see timing diagrams phase 2 and 5), the latch is fall through. Then, during phase 5 on the rising edge of FSTR*, the message held in the latch will be strobed into the FIFO.

5.3.3 By-Stander

A Module is considered a by-stander in the arbitration competition if it does not issue a Bus Request (I BRQ*) to the arbitration controller before arbitration cycle Phase 1 starts.

5.3.4 By-Stander Who Decides to Invoke Pre-emption

A module with a higher arbitration number than the master elect may initiate a new arbitration cycle to establish a new master. This process, referred to as pre-emption, allows a high priority to acquire tenure of the bus with minimum latency.

Pre-emption is allowed in phase 3 when all of the following conditions are met:

1. the arbitration number is greater than the arbitration number on the bus
2. a bus request signal was recently received
3. did not participate in the arbitration competition phase 2.

The master elect may be preempted by asserting AC1O.

Pre-emption is not allowed during:

- the first pass of a two pass cycle
- message sending

These two events are given higher precedence.

If a module decides to preempt and changes its competition number during phase 2 or phase 3 in the Arbitration Controller, the Arbitration Transceiver will still use the current latched competition number to make the greater than comparison. When the next arbitration cycle occurs, the new competition number will be used.

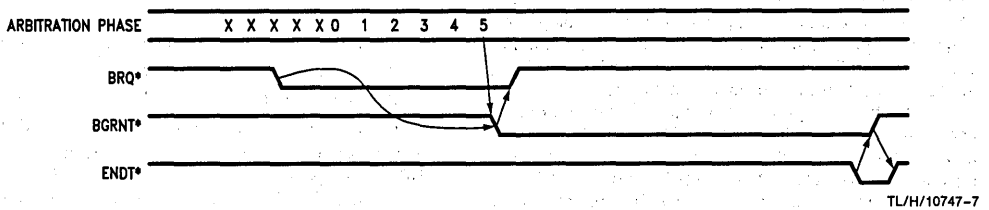
5.3.5 Master

The Master is the module that currently has tenure of the parallel bus.

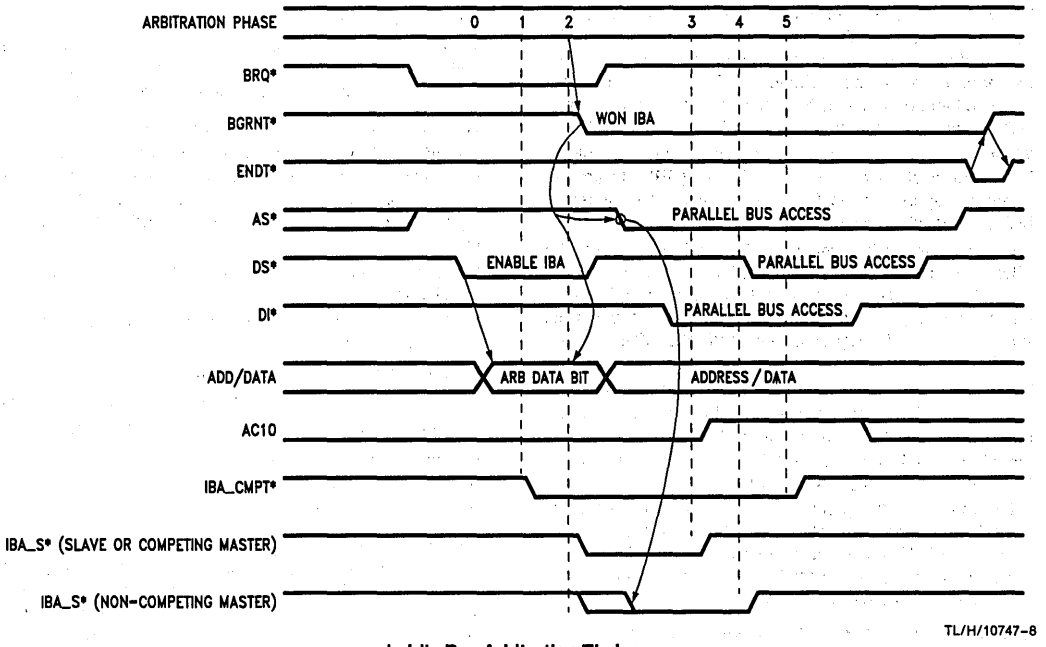
5.3.6 Master Elect

The Master Elect is the module that won the current arbitration cycle. The Master Elect will become master upon transfer of tenure.

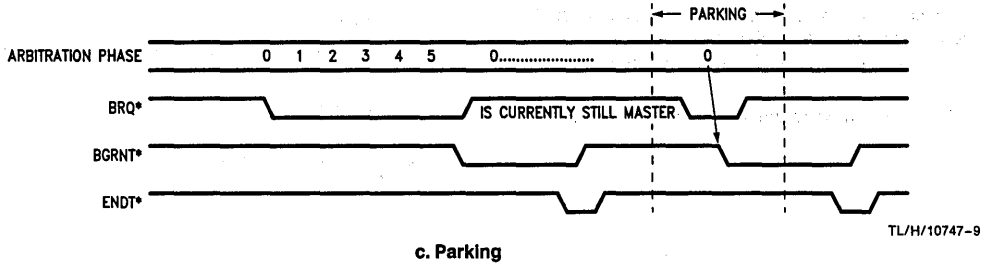
5.0 Arbitrating for Futurebus+ (Continued)



a. Normal Bus Request Timing



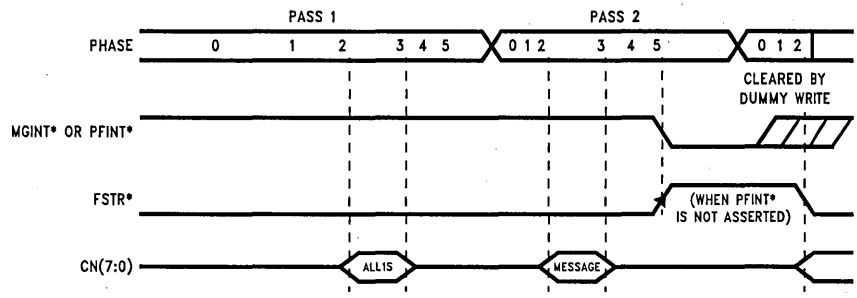
b. Idle Bus Arbitration Timing



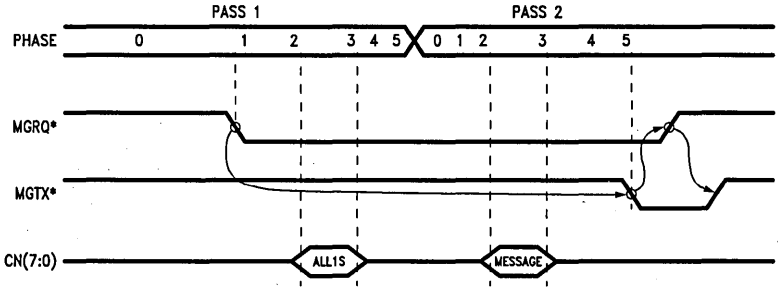
c. Parking

FIGURE 5. Bus Request Timing (Normal Arbitration, Idle Bus Arbitration and Parking)

5.0 Arbitrating for Futurebus+ (Continued)



d. Message Receiving



e. Message Sending

FIGURE 5. Bus Request Timing (Normal Arbitration, Idle Bus Arbitration and Parking) (Continued)

5.4 FUTUREBUS+ OPTIONAL MEANS OF ARBITRATION

Besides the normal Futurebus+ Parallel Contention Arbitration, the Futurebus+ Specification allows two other optional arbitration methods to acquire the parallel bus quickly:

- Idle Bus Arbitration
- Parking

Both of these arbitration methods are supported by the National Semiconductor DS3875 Arbitration Controller.

5.4.1 Idle Bus Arbitration (IBA)

Idle Bus Arbitration (IBA) is selected by setting the IBAPK* bit in CTRL3 register. The aim of IBA is to give a module quick access to Futurebus+ when the current master has completed its transfers.

IBA is invoked the same as a normal arbitration bus request (I BRQ*) but uses the parallel bus (AS* negated, DS* asserted, and the parallel address/data bus) to determine the winner of the arbitration cycle, see *Figure 5a, b.*

During normal arbitration the transfer of Futurebus+ tenure occurs during phase 5. During IBA the transfer of Futurebus+ tenure occurs during phase 2. During IBA the AC1O arbitration condition signal will be asserted by the new master. This will cancel the transfer of tenure during the normal arbitration cycle. Note that Futurebus+ tenure was transferred but only the two modules involved in the IBA (the master and the master elect) know that the transfer of tenure has taken place. All the other modules involved in the normal arbitration cycle see that the transfer of tenure during the normal arbitration cycle was canceled.

If two or more modules have a request, then normal arbitration will determine which module will gain access. IBA takes place on the parallel highway, AD/DATA (31 . . . 0). Each module is assigned a bit which is to be asserted during IBA. If only one bit is asserted during the IBA competition, then IBA issues the bus grant signal to that module. Concurrently with IBA, normal arbitration takes place. If a module does not want IBA to issue a bus grant, IBA may be inhibited by asserting DI*. During phase 5, normal arbitration will determine appropriate actions.

5.4.1.1 Masters Support Circuitry to Enable IBA

If IBA is allowed, it is invoked during phase 0. In order to support IBA external logic is needed in addition to the arbitration controller. The Masters external logic must monitor the arbitration control acquisition synchronization signals (AP*, AQ*, AR*) and the parallel bus address handshake signals (AS*, AK*, AI*). If the master supports IBA it should wait until it is ready to end its bus tenure (ENDT* asserted). Then when it releases AS* it should assert ds* to enable modules to participate in IBA competition.

5.4.1.2 Modules Support Circuitry to Participate in IBA

If a module wants to use IBA to gain tenure of Futurebus+ it must use external logic to monitor the arbitration control

5.0 Arbitrating for Futurebus + (Continued)

acquisition synchronization signals (AP*, AQ*, AR*), the parallel bus address handshake signals (AS*, AK*, AI*), the parallel bus data sync signal (DS*) and data acknowledge inverse (DI*), and the IBA_CPT* output from the arbitration controller.

5.4.2 Parking

During Power Up, the Arbitration Controller is programmed to perform either Idle Bus Arbitration or Parking by setting or clearing the IBAPK* bit in the CTLR3 register. Parking is selected by clearing the IBAPK* bit. The aim of Parking is to give the Futurebus+ bus Master quick access to the bus to perform other transfers when no one else desires to use it. Thus, it is not necessary for the master to go through the entire arbitration cycle to get the BGNT* signal. Parking issues BGNT* in Phase 0, see *Figure 5c*. If another module gets a message request or bus request, the arbitration competition cycle begins like it normally does to handle the request.

All of the following conditions should be met to take advantage of Parking:

1. Parking in selected
2. module must be the current Master
3. ENDT* signal is high
4. BGRNT* signal is high
5. BRQ* signal is asserted low while in phase 0
6. LKD* signal is high (Resources are not locked)

When these conditions are true, the Arbitration Controller issues the BGRNT* signal (asserted low) in phase 0. Upon end of tenure, BGRNT* signal will be released. The Master may continue to use Parking to perform transactions until after the arbitration competition cycle selects a new master (see *Figures 5c* and *7*).

Parking is allowed only when the Master has all resources unlocked (indicated by LKD* signal being high). During phase 0, if the master still has resources locked (ENDT* and BGRNT* signals are high), the Arbitration Controller will immediately initiate a dummy cycle to unlock resources. The UNLK* signal is generated during phase 5 upon the successful transfer of tenure. The transfer of tenure may be with itself or another module.

5.5 THE ARBITRATION PHASES

The arbitration process consists of transitioning through six phases (Phase 0 thru Phase 5) of the three arbitration synchronization signals (AP*, AQ*, AR*). To transition between control acquisition phases only one of the three arbitration synchronization signals will transition.

The Arbitration process is asynchronous and occurs as fast as all the modules that contain arbitration logic can transition through the arbitration phases. If a two-pass arbitration competition has been selected the entire control acquisition cycle is repeated twice.

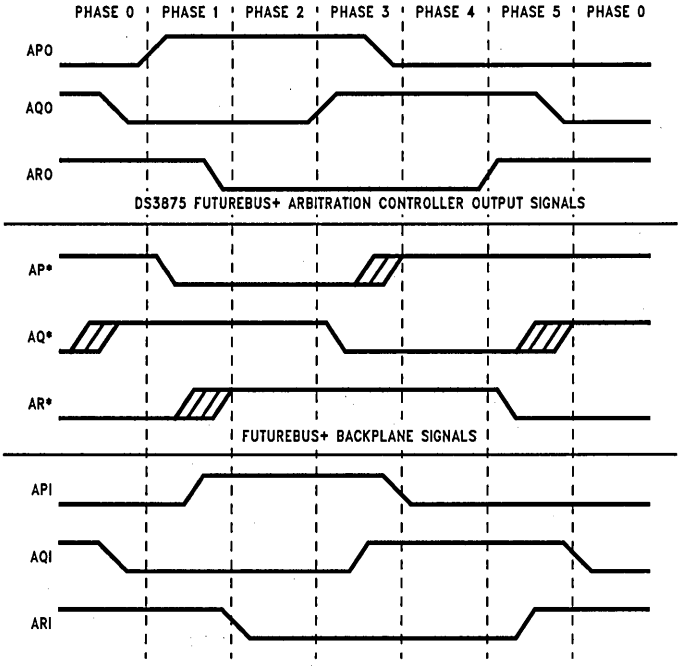
Each arbitration synchronization signal (AP*, AQ*, AR*) represents the wire-OR of each of the individual synchronization signals from each of the modules. For any of the synchronization signals on the bus to be released, all the modules must release their individual synchronization signals (ap*, aq*, ar*). Therefore, the release of a synchronization signal forms a global synchronization point for all the modules. Likewise, during the assertion of a synchronization signal all modules must remain synchronized by asserting their own signals in response.

Each module must participate in the control acquisition cycle, whether or not the module is competing, to remain synchronized with the other modules. *Figure 6* is a timing diagram of the Control Acquisition Sequence. *Figures 7a–f* are state transition diagrams of the DS3875 Arbitration Controller. Tables III and IV represent the internal register bits that are affected by the arbitration states and their transitions.

The DS3875 Arbitration Controller transitions to the next arbitration phase upon (see *Figure 6*):

1. APO output signal being asserted to phase 1
2. ARI input signal being negated to phase 2. (This input is filtered and indicates all modules have released the AR* synchronization signal on the Futurebus+ backplane.)
3. AQO output signal being asserted to phase 3
4. API input signal being negated to phase 4. (This input is filtered and indicates all modules have released the AP* synchronization signal on the Futurebus+ backplane.)
5. ARO output signal being asserted to phase 5
6. AQI input signal being negated to phase 0. (This input is filtered and indicates all modules have released the AQ* synchronization signal on the Futurebus+ backplane.)

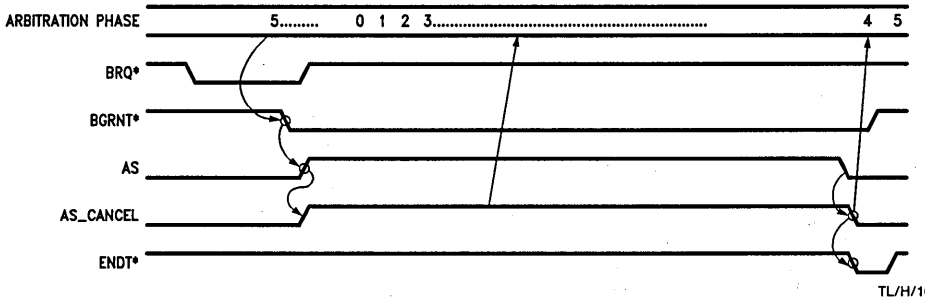
5.0 Arbitrating for Futurebus+ (Continued)



TL/H/10747–12

DS3875 FUTUREBUS+ Arbitration Controller Input Signals
FIGURE 6a. Arbitration Phases

B. Single Access, AS_CANCEL Timing Diagram (AS is tied directly to the AS_CANCEL Input of the Arbitration Controller)

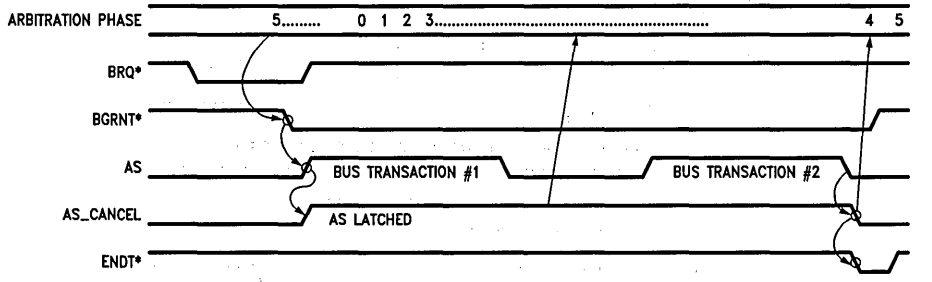


TL/H/10747–13

FIGURE 6b. Timing of the AS_CANCEL Signal

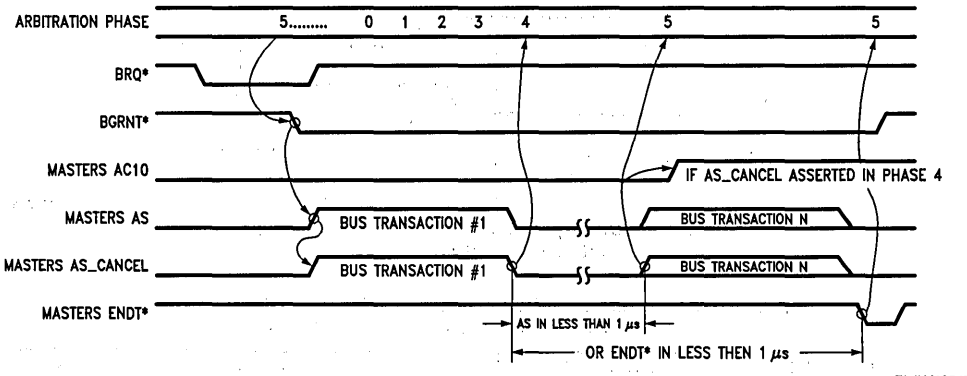
5.0 Arbitrating for Futurebus+ (Continued)

C. Multiple Accesses, AS_CANCEL Timing Diagram (AS is latched before being input to the AS_CANCEL input of the Arbitration Controller. At the end of the last transfer the latch is reset)



TL/H/10747–14

D. Multiple Accesses, AS_CANCEL Timig Diagram (AS is tied directly to the AS_CANCEL input of the Arbitration Controller. The Master must guarantee that the second transfer, or ENDT*, will come within 1 μs of the falling edge of the initial transfer AS signal).



TL/H/10747–15

FIGURES 6c, 6d. Timing of the AS_CANCEL Signal

5.5.1 Phase 0, Idle Phase

This Phase is characterized by AP* released and AR* asserted on Futurebus+ and AQf negated on the module board. See *Figure 7a* for the DS3875 Arbitration Controller Phase 0 state diagram.

The arbitration controller will be negating APO, asserting ARO and be receiving AQI negated. This is the state of the arbitration synchronization lines when no control acquisition cycle is in progress or between the two control acquisition cycles of a two-pass arbitration sequence.

While in Phase 0 the following actions are performed:

1. If ! HALT* is asserted the arbitration controller will not transition to Phase 1 until HALT* is negated.
2. If any of the "New Bits" are set (NCN, NGO, NDS) the corresponding internal arbitration controller registers will be updated and then the "New Bits" will be reset.
3. The arbitration condition lines (AC1*, AC0*) will be negated.

One of the following three types of arbitration may occur during Phase 0:

1. Normal arbitration
2. Idle Bus Arbitration (IBA)
3. Parking

5.5.1.1 Phase 0, Normal Arbitration Events That Cause a Transition to Phase 1

There are five conditions during normal arbitration where APO will be asserted causing a transition to **Phase 1:**

1. If Two-Pass Arbitration is selected, a bus request has been received (! BRQ* + ! MGRQ* + Dummy Cycle) and this module was a winner during the first pass of arbitration, then APO will be asserted.
2. Though this module may not be requesting Futurebus+ another module may want to arbitrate for Futurebus+ and assert AP*. This action will cause the AP input (API) on this module to be asserted. API asserted will cause this modules arbitration controller to assert APO.
3. If this module is currently the master and ! LK* is asserted, BGRNT* is negated, ! ENDT* has been received, no other requests have been received and API is negated, the module will initiate a dummy cycle to unlock its resources by asserting APO (if it is doing Single Pass or the First Pass of Two-Pass Arbitration).
4. A message request (! MGRQ*) will cause APO to be asserted (if it is doing Single Pass or the First Pass of Two-Pass Arbitration). Note that if both a bus request (! BRQ*) and ! MGRQ* are received during phase 0,

5.0 Arbitrating for Futurebus+ (Continued)

! MGRQ* will be given a higher priority and will be acted upon during this arbitration cycle. ! BRQ* will wait to arbitrate during the next arbitration cycle.

5. A Bus Request (! BRQ*) will cause APO to be asserted (if it is doing Single Pass or the First Pass of Two-Pass Arbitration). This will cause the Futurebus+ AP* to be asserted and will indicate to the other modules that an arbitration competition is beginning.

Note that the user will violate the Futurebus+ specification if he leaves the local modules resources locked (! LKD*) and has IBA enabled in the arbitration controller. This incident is dangerous because it could lead to another module winning IBA with the local modules resources being locked. If the other module tried to access the local modules resources it would result in deadlock.

5.5.1.2 Phase 0, Idle Bus Arbitration Events That Cause a Transition to Phase 1

If IBA is desired and the arbitration cycle is in Phase 0 (! APO, ! AQI, ARO), the parallel bus is idle (AS*, AK*, ! AI*), and DS* has been released for a minimum period of time (Futurebus+ spec.) the Masters external logic may assert ! DS*. This will alert any module capable of doing IBA that IBA has been enabled.

5.5.1.3 Phase 0, Parking

The aim of Parking is to give the Futurebus+ bus master quick access to the bus to perform other transfers when no one else desires to use it. If Parking is enabled (! IBA_PK*) and is successful the arbitration controller will issue BGNT* in Phase 0. If another module gets a message request or bus request, the arbitration competition cycle begins like it normally does to handle the request (see Section 5.5.4 for more information).

5.5.2 Phase 1, Decision Phase

This Phase is characterized by AP* and AR* asserted and AQ* released on Futurebus+. See *Figure 7b* for the DS3875 Arbitration Controller Phase 1 state diagram.

The arbitration controller will be asserting APO and ARO and negating AQO. This is the state of the arbitration synchronization lines when the decision phase (1) is in progress.

During Phase 1 the individual modules must make the decision whether they want to compete. This decision will be based upon the state of the modules bus request, message request or locked status (! BRQ* or ! MSGRQ* or ! LKD*) at the time APO is asserted. Since this condition is subject to mestastability a metastable hardened latch is used internal to the DS3875 to resolve this potential condition.

If the module is going to compete the arbitration competition number (CN(7:0)) and its parity bit (CNp) will be asserted to the arbitration transceiver, Latch enable of the arbitration transceiver will be asserted and negated (! CN_LE* asserted for 20 ns) to latch in the arbitration number, and compete will be asserted (! CMPT*) to enable the arbitration competition number onto Futurebus+.

If the module is a slow module (! FS*, see Section 7.6) the arbitration handshake signal AC0O will be asserted.

Once the decision to compete has been made, the arbitration handshake signal (! AC1O) that cancels the arbitration cycle will be negated. The Programmable Skew (PS(1:0)) gives time for the arbitration number to become valid on Futurebus+ before ARO is negated. Once the Programmable Skew has timed out ARO will be negated. Once all modules have negated AR* the arbitration cycle will transition to **Phase 2.**

Once all modules have negated AR* a 1 μs timer is started. This timer is used to guarantee that phase 2 is completed within one to 2 μs.

5.5.2.1 Phase 1, Idle Bus Arbitration Events That Cause a Transition to Phase 2

During phase 1 the arbitration controller will output ! IBA_CPT*. When the external logic sees ! IBA_CTP* and the parallel bus is inactive (AS*, AK*, ! AI*) it should drive one of the data bits of the parallel address/data bus.

5.5.3 Phase 2, Competition Phase

This Phase is characterized by AP* asserted and AQ* and AR* released on Futurebus+. See *Figure 7c* for the DS3875 Arbitration Controller Phase 2 state diagram.

The arbitration controller will be asserting APO, negating AQO and receiving ARI negated. This is the state of the arbitration synchronization lines when the competition phase (2) is in progress.

While in Phase 2 the following actions are performed:

1. AB_RD* is asserted after 30 ns. This allows the arbitration controller to monitor the winning competition number.
2. The FIFO STRobe is now asserted (! FSTR*).

There are two conditions that can cause AQO to be asserted causing a transition to **Phase 3:**

1. The AQI input being asserted.
2. Competing, arbitration competition settling time (t_s) expired, one to 2 μs Phase 2 arbitration error timer not expired, and the WIN*_GT* input being asserted.

5.5.3.1 Phase 2, Idle Bus Arbitration Events That Cause a Transition to Phase 3

The external logic should drive the IBA Success signal (! IBA_S*) during phase 2 if DI* is released, only this modules data bit is driven on the data bus, and the arbitration settling Time (t_s) has not expired.

When the arbitration controller sees ! IBA_S* asserted it will issue bus grant (! BGRNT*) to give access of the bus to the module that won the IBA. If two or more modules have a request, then normal arbitration will determine which module will gain access.

5.5.4 Phase 3, Error Check Phase

This Phase is characterized by AP* and AQ* asserted AR* released on Futurebus+. See *Figure 7d* for the DS3875 Arbitration Controller Phase 3 state diagram. Also see *Figures 6b, c, d* for how AS_CANCEL relates to Phase 3.

The arbitration controller will be asserting APO and AQO, and negating ARO. This is the state of the arbitration synchronization lines when the error check phase (3) is in progress.

While in Phase 3 the following actions are performed:

1. If the module is a competitor and winner of the arbitration, the W(winner) bit in the STATUS register is set.
2. Upon entering phase 3, after 20 ns, AB_RD* is negated.

5.0 Arbitrating for Futurebus + (Continued)

There are seven conditions that will cause the arbitration controller to negate APO:

1. This module has had a Bus ReQuest (! BRQ*), won IBA, and asserted AC1O and AS.
2. The current master releases AS (! AS_CANCEL). The master has completed its transactions on the parallel bus, see *Figures 6b, c, d.*
3. This module detected the 1 μs timeout or a parity error occurred during the arbitration cycle. This error condition will cause the assertion of AC0O and AC1O and ERINT* signals which will inhibit the transfer of tenure during this arbitration cycle.
4. Another module has detected that transfer of tenure is to be inhibited (AC1I).
5. This module was competing and won the competition where this is the first pass of a two pass arbitration cycle.
6. This module did not compete (CMPT*) but now has a bus request (! BRQ*) and its competition number is greater than (! WIN*_GT*) the modules competition number that won the current arbitration competition. This module preempts the current arbitration competition by asserting AC1O. This inhibits the transfer of tenure during this arbitration cycle and allows a new arbitration cycle to be initiated in which this module can compete.
7. This module was competing (! CMPT*) and won the competition (W bit of Status register set) but the bus request or message request has been negated (BRQ* or MGRQ*). This error condition will cause the assertion of AC0O, AC1O and ERINT* signals which will inhibit the transfer of tenure during this arbitration cycle.

After all the modules have negated AP*, upon receiving IAPI, the arbitration cycle will transition to **Phase 4.**

5.5.4.1 Phase 3, Idle Bus Arbitration Events That Cause a Transition to Phase 4

During phase 3 of the arbitration cycle, when the current master has not yet released the bus (AS), the new master (a module that was competing (IIBA_CMPT*) and won (IIBA_S*)) will drive AC1O to inhibit the transfer of tenure during the normal bus arbitration cycle (tenure was transferred during the IBA cycle).

Note: The new master will now continue the normal arbitration process to completion but may begin conducting transactions on the bus.

5.5.5 Phase 4, Master Release Phase

This Phase is characterized by AP* released, AQ* asserted, and AR* released on Futurebus+. See *Figure 7e* for the DS3875 Arbitration Controller Phase 4 state diagram and *Figures 6b, c, d* for how AS_CANCEL relates to phase 4.

The arbitration controller will be receiving API negated, asserting AQO and negating ARO. This is the state of the arbitration synchronization lines when the master release phase (4) is in progress.

While in Phase 4 the following actions are performed:

1. All modules, other then the bus master, start a 1 μs timer. This is referred to as the Master release timer. The master has 1 μs to complete its transactions or to inhibit the transfer of tenure and proceed to phase 5. This timer is specified in the Futurebus+ specification so in the event where the current master does not respond, the master elect may assume mastership during phase 5.
2. The CMPT* signal is negated (CMPT*).

There are six conditions that can cause ARO to be asserted causing a transition to **Phase 5:**

1. Transfer of tenure is inhibited indicated by the assertion of AC1I.
2. Was not Master and the ARI input was asserted.
3. Was Master and did not cancel (! AS_CANCEL) the arbitration competition, see *Figures 6b, c, d.*
4. There is no current master (immediately following power up) so the master elect may assert ARO.
5. Was Master and did cancel (! AS_CANCEL) the arbitration competition, see *Figure 6d.* In this case the current master will assert AC1O to inhibit the transfer of tenure during this arbitration cycle.
6. All modules other then the current master detect the Master Release Timeout.

5.5.6 Phase 5, Tenure Transfer Phase

This Phase is characterized by AP* released, AQ* asserted, and AR* asserted on Futurebus+. See *Figure 7f* for the DS3875 Arbitration Controller Phase 5 state diagram.

The arbitration controller will be negating API and asserting AQO and ARO. This is the state of the arbitration synchronization lines when the tenure transfer phase (5) is in progress.

While in Phase 5 the following actions are performed:

1. If IBA_CMPT* signal was asserted, it is now negated (IBA_CMPT*).
2. The unlock signal (! UNLK*) will be asserted if locked (! LKD*) is asserted and the arbitration condition signals (AC0I, AC1I) are negated.
3. This phase will update the following status bits:
   1. The Master status bit (bit 5) in the status register.
   2. The Competitor status bit (bit 4) in the status register.
   3. The WIN status bit (bit 3) in the status register.
   4. The Uniqueness, round robin and priority fields of the RXCN1 register.
   5. The Priority field of the RXCN0 register.

There are four conditions that can cause AQO to be negated:

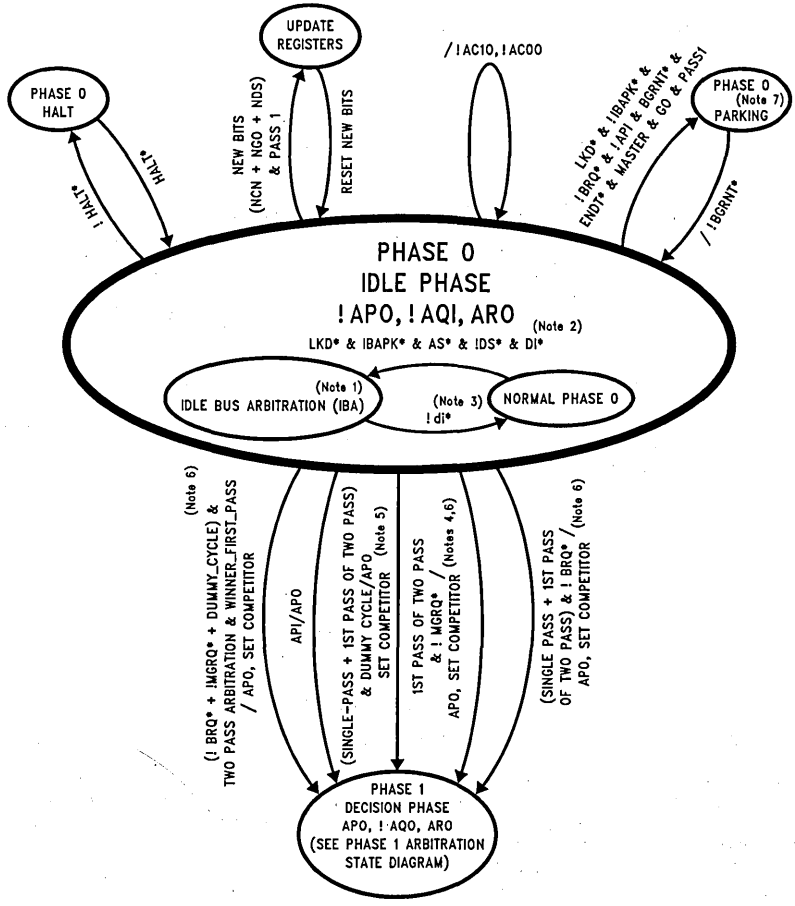
1. Had a message request (! MGRQ*) and won the competition (! WIN*_GT*). In this case the message transmitted output will also be asserted (! MSGTX*).
2. Competed for the bus (! CMPT*), there were no errors (! AC0I, ! AC1I), and won the competition (! WIN*_GT*). In this case the bus grant output will be asserted (! BGRNT*) and the internal master bit will be set (M, bit 5 in the status register).
3. There were no errors (! AC0I, ! AC1I) and the module is not the winner. Resets the M bit in the status register.
4. A Message was received. In this case either message interrupt will be asserted (! MGINT*) or the powerfail interrupt (! PFINT*) will be asserted.

5.0 Arbitrating for Futurebus+ (Continued)

When a message is received the FIFO strobe (FSTR*) will be negated, strobing a new message into the external FIFO, given the following conditions:

1. The message is not being sent by this module.
2. This is the second pass of an arbitration message.
3. No errors occurred during this arbitration cycle.
4. This is not a powerfail interrupt message (! PFINT*).

Once all the modules have negated AQ*, upon receiving !AQI, the arbitration cycle will transition to **Phase 0.**



TL/H/10747–16

FIGURE 7a. DS3875 Phase 0 Arbitration State Diagram

Note: **HALT* is assumed to be negated in this state diagram. Also, the arbitration Controller GO bit in CTRL3 register must be asserted for the controller to accept requests. Otherwise this module may not compete but will be a by-stander.

Note 1: Idle Bus Arbitration during Phase 0 is not an internal state of the Arbitration Controller. It is shown in this state diagram only to give a better understanding of Phase 0 arbitration.

Note 2: The Arbitration Controller does not check for LKD*, but it should not be asserted during IBA to be in compliance with the Futurebus+ specification. If lock was asserted during IBA another module could win IBA and try to access this modules locked resources, this would result in deadlock. Also, the Arbitration controller does not check AS* & ! DS* & DI*, this is left up to external logic and this must be the first pass if two pass arbitration has been selected.

Note 3: The output ! di* from this module is driven from external control logic, it is only shown in this diagram to allow a clearer understanding of how IBA relates to normal arbitration.

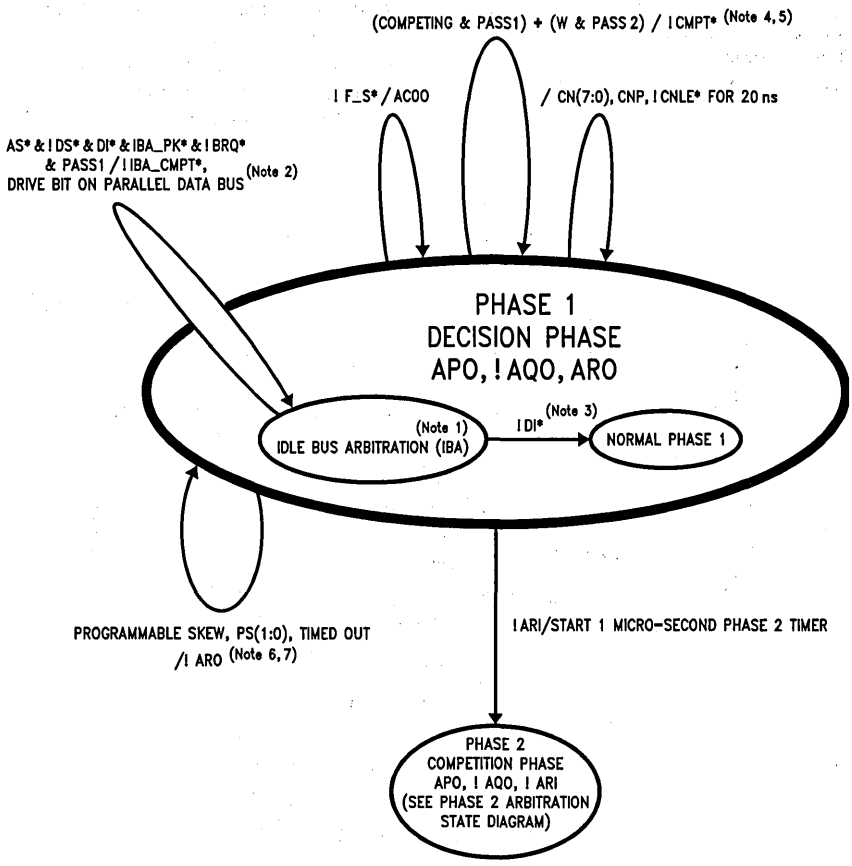
Note 4: Note that message request (MGRQ*) has a higher priority inside the arbitration controller then does bus request (BRQ*).

Note 5: This is a dummy arbitration cycle initiated to unlock (UNLK*) the modules resources. The dummy cycle is initiated, by the current master, if the modules resources are still locked (! LKD*) after end of tenure (ENDT*) has been issued (*BGRNT* & Master & ! LKD* & ENDT*) and the arbitration bus is idle.

Note 6: If this is the second pass of a two-pass arbitration cycle and a new request (! BRQ* or ! MGRQ*) is generated, that request will not be allowed to compete until the next arbitration cycle. Note that pre-emption is allowed.

Note 7: If Two-Pass arbitration has been selected, Parking is only possible during Phase 0 of the first pass.

5.0 Arbitrating for Futurebus+ (Continued)



TL/H/10747–17

FIGURE 7b. Phase 1 Arbitration State Diagram

Note 1: Idle Bus Arbitration during Phase 1 is not an internal state of the Arbitration Controller. It is shown in this state diagram only to give a better understanding of Phase 1 arbitration.

Note 2: One bit on the parallel data bus is driven by external logic (the arbitration controller only drives the ! IBA_CMPT output) only if IBA is supported by the module. Note that the arbitration controller does not require AS* & ! DS* & DI*, external logic will require these conditions (from the Futurebus+ specification).

Note 3: If a module does not want IBA to select a new master external logic around the arbitration controller will drive ! di*. This transition is shown in this diagram to allow a clearer understanding of how IBA relates to normal arbitration.

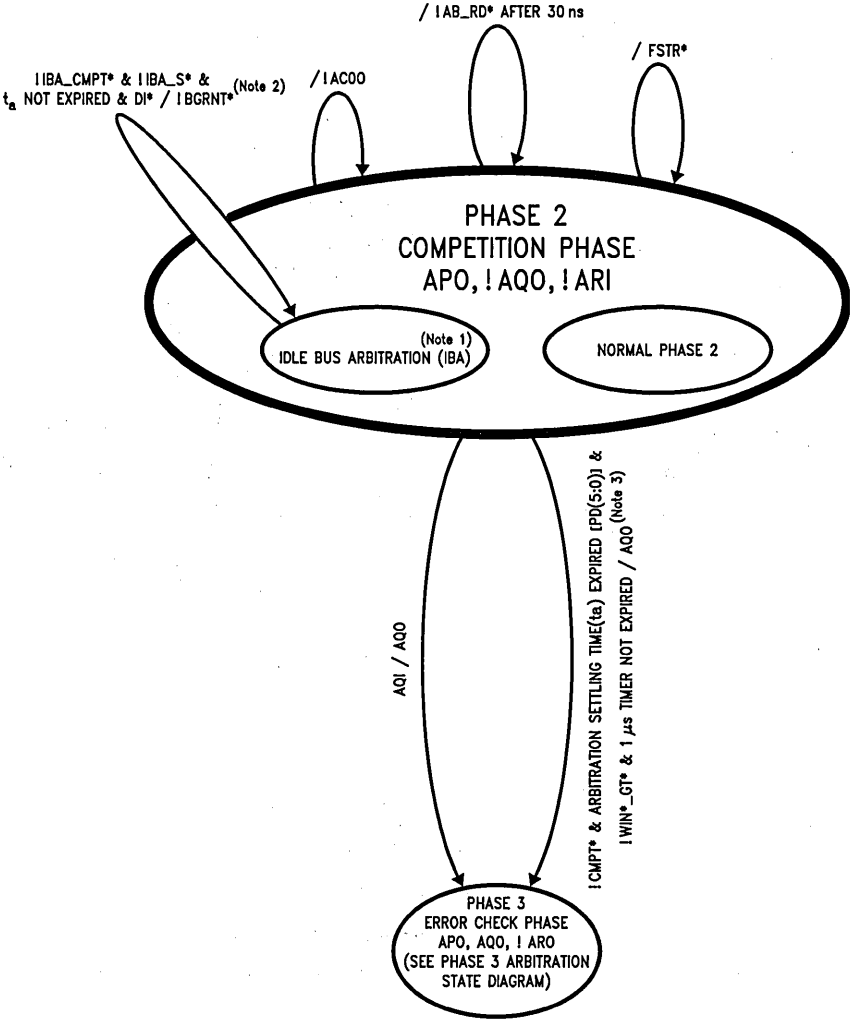
Note 4: Competing = "! BRQ* + ! MGRQ* + Dummy Cycle", W = W bit of STATUS register.

Note 5: These Requests must have occurred before APO was asserted or they will not compete until the next arbitration cycle.

Note 6: The possible metastable condition of "(! BRQ* + ! MGRQ* + Dummy Cycle) occurring at the same time as API" must be resolved before driving ! ARO.

Note 7: All conditions shown above the Phase 1 state must be completed before driving ! ARO.

5.0 Arbitrating for Futurebus+ (Continued)



TL/H/10747–18

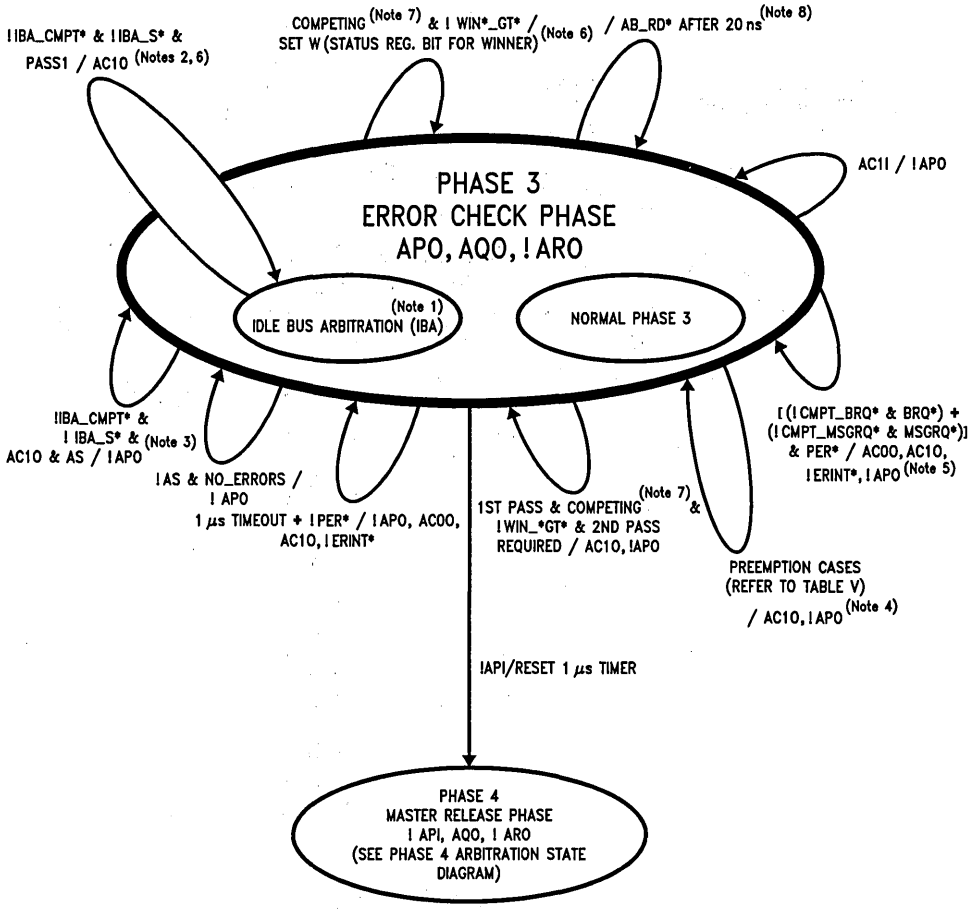
FIGURE 7c. Phase 2 Arbitration State Diagram

Note 1: Idle Bus Arbitration during Phase 2 is not an internal state of the Arbitration Controller. It is shown in this state diagram only to give a better understanding of Phase 2 arbitration.

Note 2: Bus Grant (! BGRNT*) is driven by the arbitration controller during Phase 2 only if IBA was successful for this module and the Arbitration settling time (ta) has not expired. Also, DI* is shown here to indicate that IBA is allowed to select a new Master. DI* is not an Arbitration Controller signal. DI* is represented here for completeness shake only to give a better understanding. Also, external logic will release the bit on the Parallel data bus.

Note 3: This transition to Phase 3 occurs when this module was competing, its arbitration settling time expired, and the one micro-second Phase 2 error timer has not expired.

5.0 Arbitrating for Futurebus+ (Continued)



TL/H/10747–19

FIGURE 7d. Phase 3 Arbitration State Diagram

Note 1: Idle Bus Arbitration during Phase 3 is not an internal state of the Arbitration Controller. It is shown in this state diagram only to give a better understanding of Phase 3 arbitration.

Note 2: AC1O is asserted to cancel the transfer of bus tenure if IBA was successful.

Note 3: AS is asserted if IBA was successful for this module.

Note 4: A Module will pre-empt another module if it did not compete in the present competition yet its arbitration number is higher than the current winner. In this instance the pre-emption module will assert AC1O to cancel the transfer of tenure and allow a new competition to take place. Preemption is not allowed in the 1st pass when the arbitration cycle consists of two pass.

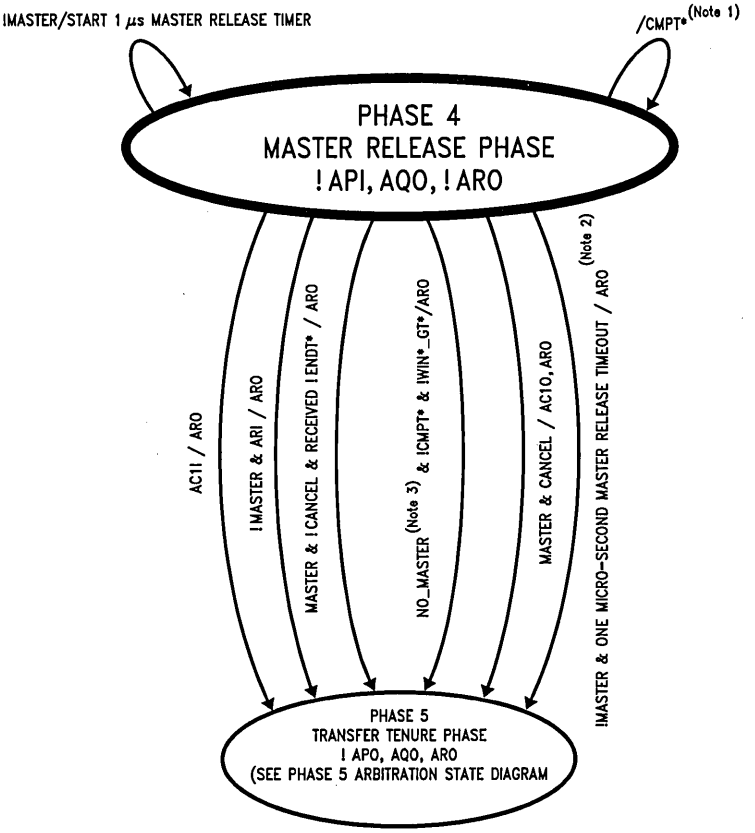
Note 5: If a module that competed and won no longer is issuing a bus or message request, AC1O will be driven to cancel the transfer of tenure and ! ERINT* will be driven to indicate an error has occurred.

Note 6: AC1O (if IBA was successful) and the W STATUS bit (if competing and won) must be asserted before driving ! APO.

Note 7: Competing = ! BRQ* + ! MGRQ* + Dummy Cycle.

Note 8: All actions in this phase are performed after the arbitration number is latched.

5.0 Arbitrating for Futurebus+ (Continued)



TL/H/10747–20

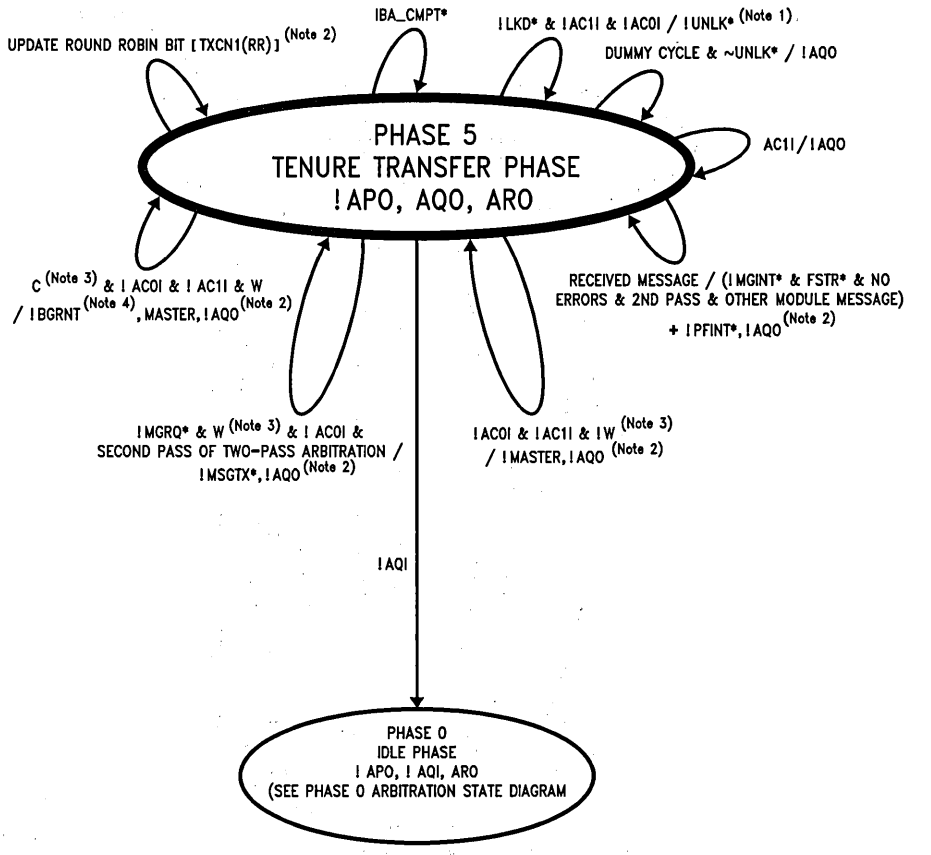
FIGURE 7e. Phase 4 Arbitration State Diagram

Note 1: The Arbitration controller releases its arbitration number on Futurebus+ by negating CMPT* to the arbitration transceiver.

Note 2: If no master or master error occurred, on timeout, assert ARO so the master elect may take over the bus.

Note 3: No_Master = No Current Master of Futurebus+ during Powerup or Initialization.

5.0 Arbitrating for Futurebus+ (Continued)



TL/H/10747–21

FIGURE 7f. Phase 5 Arbitration State Diagram

Note 1: Unlock this modules resources if its resources were locked and this is restricted mode, one pass arbitration, or the second pass of a two-pass arbitration cycle.

Note 2: Only update Master status bit, TXCN1 (RR), !MGINT*, FSTR*, !PFINT*, !BGRNT*, or !MSGTX* given that this is restricted mode, one pass arbitration, or the second pass of a two-pass arbitration cycle.

Note 3: W = W bit in the STATUS register, C = C bit in the STATUS register.

Note 4: !BRGNT* is not issued if this is a dummy arbitration cycle, initiated by the arbitration controller to unlock resources.

Arbitration Controller Internal Register Bit Changes

TABLE III. Restricted Mode Arbitration or Unrestricted Mode Arbitration (One Pass, or Second Pass of Two Pass)

Modules Category during Arbitration	TXCN1 RR	STATE All Bits (Note 1)	STATUS			RXCN0 All Bits	RXCN1 (Note 2)				RXMSG A(0:7)	CLRERI Reset, Set	CLRMGI CLRPFI Reset, Set
			W	C	M		U(0:4)	RR	PO	P1			
Competitor	5	X	1, 2	0, 1	5		3	3	3	3	3	X, 4	X, 5
By-Stander	5	X	1	0	5		3	3	3	3	3	X, 4	X, 5
Sending Message		X	1, 2	0, 1							3	X, 4	X, 5
Receiving Message		X	1	0, 1							3	X, 4	X, 5
Pre-Emption		X					3	3	3	3			
IBA	5 (Note 4)	X			5 (Note 5)		3	3	3	3	3	X, 4	X, 5
Parking (Note 3)													

Note: The number shown within the box defines the arbitration phase where the particular bit may change state.

Note 1: Note that the STATE Register is updated in each Phase of Arbitration.

Note 2: This is the new Masters Competition Number.

Note 3: Parking is only a Phase 0 event.

Note 4: The RR bit is only updated on the New Master.

Note 5: The M bit is updated if IBA succeeded during Restricted Mode, or Unrestricted Mode (One Pass or the First Pass of Two Pass Arbitration).

"X" means that any Phase can update this bit (or bits).

"," is used in the table to depict the condition where a particular Status bit is Reset and where it is Set (Ex. 0, 1 means that this Status bit is Reset in Phase 0 and Set in Phase 1).

"–" is used in the table to depict the condition where a particular Status bit may change within several sequential phases (Ex. 2–4 means that this Status bit can change in Phases 2, 3, or 4).

TABLE IV. Unrestricted Mode, First Pass of Two Pass Arbitration

Modules Category during Arbitration	TXCN1 RR	State All Bits (Note 1)	Status			RXCN0 All Bits (Note 2)	RXCN1				RXMSG A(0:7)	CLRERI Reset, Set	CLRMGI CLRPFI Reset, Set
			W	C	M		U(0:4)	RR	PO	P1			
Competitor		X	1, 2	0, 1		3						X, 4	X, 5
By-Stander		X	1	0		3						X, 4	X, 5
Sending Message		X	1, 2	0, 1								X, 4	X, 5
Receiving Message		X	1	0, 1								X, 4	X, 5
Pre-emption		X				3							
IBA		X			5	3						X, 4	X, 5
Parking (Note 3)													

Note: The Number shown within the box defines the arbitration phase where the particular bit may change state.

Note 1: Note that the STATE Register is updated in each Phase of Arbitration.

Note 2: This is the new Masters Competition Number.

Note 3: Parking is only a Phase 0 event.

Note: "X" means that any Phase can update this bit (or bits).

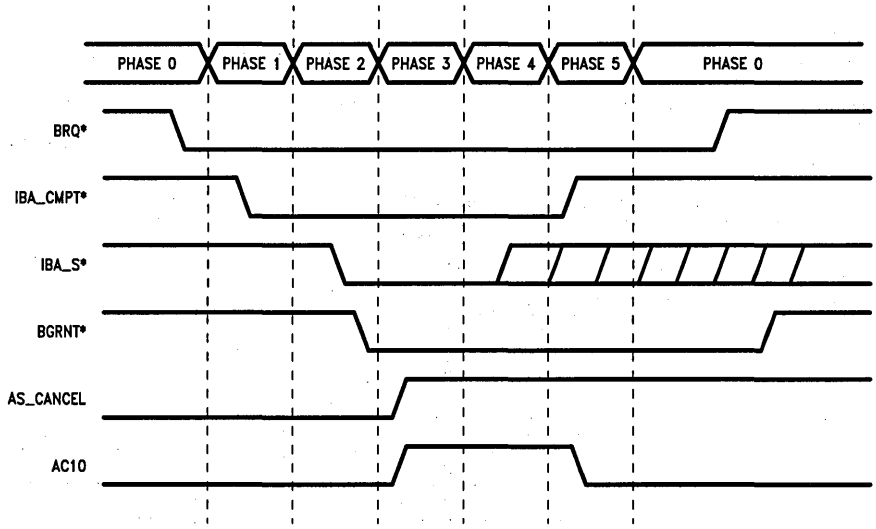
Note: "," is used in the table to depict the condition where a particular Status bit is Reset and where it is Set (Ex. 0, 1 means that this Status bit is Reset in Phase 0 and Set in Phase 1).

Note: "-" is used in the table to depict the condition where a particular Status bit may change within several sequential phases (Ex. 2-4 means that this Status bit can change in Phases 2, 3, or 4).

TABLE V. Pre-emption

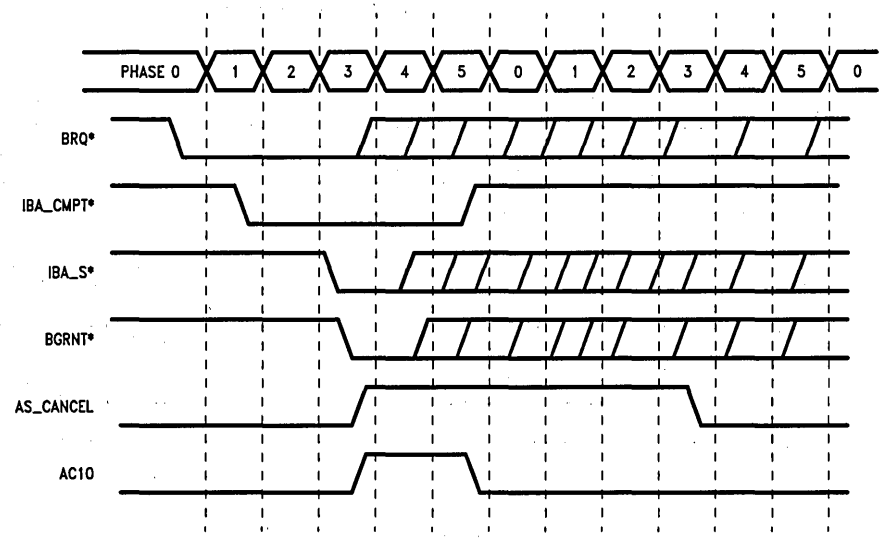
Category	Pre-emption Case	Result
Messages	message vs message	no pre-emption
	message vs anything else	pre-empt: no number comparison
	anything vs message	no pre-emption
Restricted Mode	restricted vs restricted	pre-empt if: TXCN1 > RXCN1
	restricted vs message	no pre-emption
Unrestricted Mode	1 pass vs 2 pass	pre-empt: no number comparison
	1 pass vs 1 pass	pre-empt if: TXCN1 > RXCN1
	2 pass vs 2 pass	pre-empt if: TXCN0 > RXCN0 and TXCN1 > RXCN1

Note: All pre-emptions take place in the second pass of arbitration or the only pass of arbitration.



TL/H/10747–29

FIGURE 12a. IBA—Restricted or Unrestricted Single Pass



TL/H/10747–30

FIGURE 12b. IBA—Unrestricted Two Pass

6.0 The DS3875 Arbitration Controller Support of Locking and Unlocking

The DS3875 Arbitration Controller supports locking and unlocking of the modules resources whether it is a master or a slave (see *Figure 8*).

If the Module becomes master and wishes to lock the slave's resources, it should assert LKD* (! LKD*) after receiving bus grant (! BGRNT* from the arbitration controller), and drive the appropriate lock command during the parallel bus connection phase. This lock command (during the connection phase of the parallel bus) will cause the selected slave module(s) to assert lock (! LKD*) to their arbitration controller. Once the module has finished its parallel bus transactions it will assert end of tenure (! ENDT*).

If the Master is in, or enters, Phase 0:

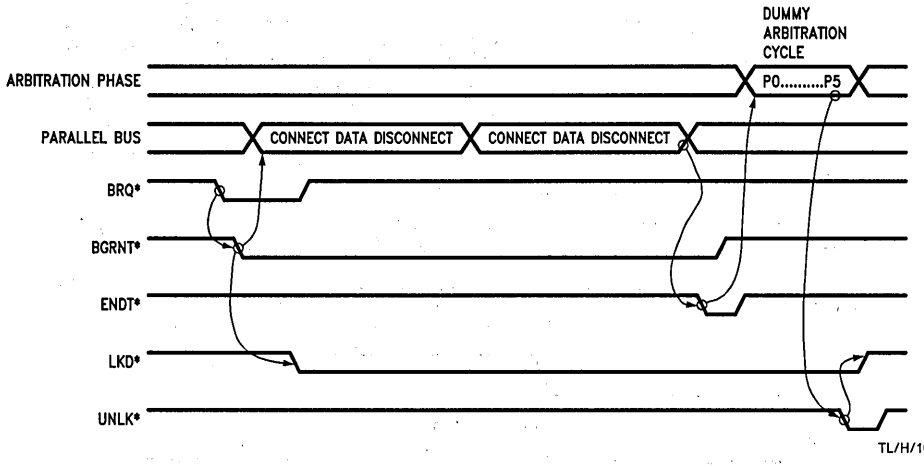
1. with end of tenure issued,
2. no other module has started an arbitration competition,
3. LKD* is still asserted,
4. a new bus request (! BRQ*) has not been issued

it will start a dummy arbitration competition cycle to unlock all resources on the bus that were locked. The dummy cycle will proceed through the normal arbitration competition except that no bus grant will be asserted if the module wins the arbitration competition.

If another module has initiated an arbitration competition the master will participate in the competition and no dummy cycle will be initiated.

During Phase 5, upon successful completion of the arbitration cycle unlock will be asserted (! UNLK*) until the lock input is negated.

A. Master Module Lock and Unlock Timing Diagram



B. Slave Module Lock and Unlock Timing Diagram

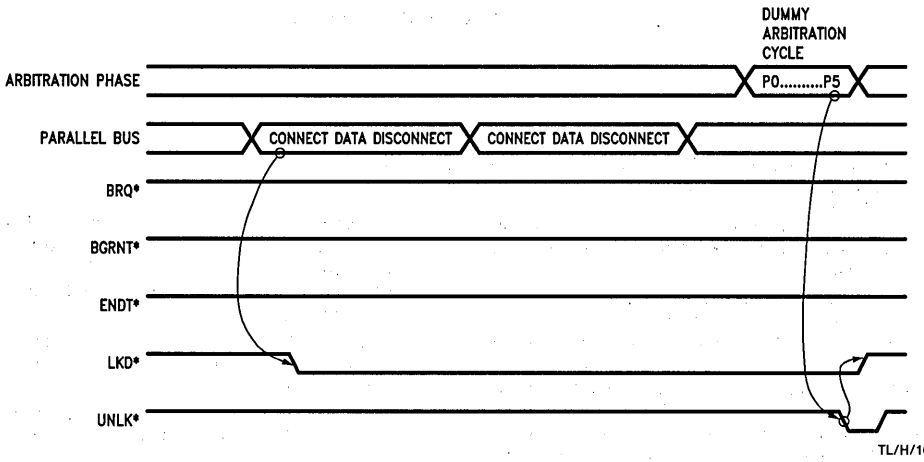


FIGURE 8. Master and Slave Lock and Unlock Timing Diagrams

7.0 Register Description

TOTAL MEMORY MAP

ADDRESSABLE REGISTERS

0000	TXCN0 7	(R/W) 0
0001	TXCN1 7	(R/W) 0
0010	TXMSG 7	(R/W) 0
0011	CTRL1 7	(R/W) 0
0100	CTRL2 7	(R/W) 0
0101	CTRL3 7	(R/W) 0
0110	STATE 7	(R) 0
0111	STATUS 7	(R) 0
1000	RXCN0 7	(R/W) 0
1001	RXCN1 7	(R/W) 0
1010	RXMSG 7	(R/W) 0
1011	CLRERI 7	(R/W) 0
1100	CLRMGI 7	(R/W) 0
1101	CLRPFI 7	(R/W) 0
1110	Reserved	
1111	REV NO. 7	(R) 0

HARDWIRED REGISTER

ALL 1s 8	0

Legend:

	Register Type	
Register Address ADD(3:0)	Register_Name MSB	(Read/Write) LSB

7.0 Register Description (Continued)

This section describes the addressable registers and the ALL1s hardwired register.

7.1 ALL1S

This register carries the first word of an arbitration message (h'1ff).

7.2 TXCN0 (ADD(3:0) = 0000)

For the Unrestricted Mode, this register stores the pass 1 arbitration number of a two pass arbitration. (CN7 = 0). Defaults to h'00.

7	6	5	4	3	2	1	0
0	P7	P6	P5	P4	P3	P2	P1

Bit	Symbol	Description
0–6	P(1:7)	Priority field of the arbitration number.
7	0	This bit is fixed at zero as specified in the Futurebus+ specification for the pass 1 number of a two pass arbitration.

Note: The Parity bit CNp for the arbitration number is internally generated during the time the host writes the numbers into the controller. The odd parity as specified in the Futurebus+ specifications is generated. CNp is set to one when even number of ones are present in the arbitration number.

7.3 TXCN1 (ADD(3:0) = 0001)

For the Unrestricted Mode, this register stores the pass 2 arbitration number of a two pass arbitration, or stores the single pass arbitration number (CN7 = 1). This register also stores the arbitration number used in the Restricted Mode. Defaults to h'10.

7	6	5	4	3	2	1	0
1/P1	P0	RR	U4	U3	U2	U1	U0

Bit	Symbol	Description
0–4	U(0:4)	Uniqueness field of the arbitration number.
5	RR	Round Robin bit of the arbitration number. **Note:** This bit is updated during phase 5, upon the successful transfer of tenure. See Table III.
6	P0	Priority field of the arbitration number. If configured as a two pass arbitor, P0 is part of P(0:7). If configured as a single pass arbitor, P0 is the priority field. If configured to arbitrate in restricted mode then P0 is part of P(0:1).
7	1/P1	This bit is fixed at one as specified in the spec for the pass 2 number of a two pass arbitration, and for the single pass arbitration number. This bit is P1 for the restricted mode arbitration number.

Note: The Parity bit CNp for the arbitration number is internally generated during the time the host writes the numbers into the controller. The odd parity as specified in the Futurebus+ specifications is generated. CNp is set to one when even number of ones are present in the arbitration number.

7.0 Register Description (Continued)

7.4 TXMSG (ADD(3:0) = 0010)

This register holds the arbitration message to be transmitted. Defaults to h'ff (Powerfail).

7	6	5	4	3	2	1	0
A7	A6	A5	A4	A3	A2	A1	A0

Bit	Symbol	Description
0–7	A(0:7)	Arbitration message to be sent.

Note: The Parity bit CNp for the arbitration number is internally generated during the time the host writes the numbers into the controller. The odd parity as specified in the Futurebus+ specifications is generated. CNp is set to one when even number of ones are present in the arbitration number.

7.5 CTRL1 (ADD(3:0) = 0011)

Control register 1. PS(1:0) programs the delay needed to ensure the assertion of the most significant 1 in CN(7:0) after compete is issued before the release of ar*, or, during IBA, this delay ensures there is sufficient time to assert the AD/DATA lines before the release of ar*. PC(5:0) programs the internal divider to receive a clock input from 2 MHz to 40 MHz in steps of 1 Mz. During chip reset, (RST*) the divider is set to receive a clock of 20 MHz. This input clock divider should be programmed during initialization for the desired frequency. Defaults to h'14.

7	6	5	4	3	2	1	0
PS1	PS0	PC5	PC4	PC3	PC2	PC1	PC0

Bit	Symbol	Description
0–5	PC0–PC5	Programs the internal input clock divider to receive a clock from 2 MHz to 40 MHz, in steps of 1 MHz. The binary value of the clock frequency is coded with PC5 being the MSB. For example, to program the divider to receive a 25 MHz signal PC5:PC0 bits are: 011001. To program the divider to receive a 10 MHz signal PC5:PC0 bits are: 001010.
6–7	PS0–PS1	Programs the chip to chip skew when releasing AR* in phase 1. PS0 PS1 DELAY 0 0 0 ns 0 1 5 ns 1 0 15 ns 1 1 25 ns This delay ensures that after compete is issued to the transceiver sufficient time is allowed for the number to be put on the bus before releasing AR*. It also provides a means to provide sufficient time for asserting AD/DATA lines during IBA before the release of ar*.

7.0 Register Description (Continued)

7.6 CTRL2 (ADD(3:0) = 0100)

Control register 2. This register stores two parameters FS__(Fast/Slow) and DS__(Restricted/Unrestricted) that configure the controller to operate in the chosen mode. PD(5:3) selects one of eight delays for the fast module and PD(2:0) selects one of eight delays for the slow module. Defaults to h'00.

7	6	5	4	3	2	1	0
F__S*	R__U*	PD5	PD4	PD3	PD2	PD1	PD0

Bit	Symbol	Description
0–5	PD0–PD5	Programs the arbitration delay that is timed in a competition. These bits give a total of 16 selectable delays, 8 fast and 8 slow. PD0–PD2 select the slow module's delay and PD3–PD5 select the fast module's delay.

(PD5 PD4 PD3)	Fast	(PD2 PD1 PD0)	Slow
000	60	000	80
001	80	001	120
010	100	010	160
011	120	011	200
100	140	100	240
101	160	101	280
110	180	110	320
111	200	111	360

6	R__U*	R__U*	Restricted/Unrestricted:
		0	Configures the controller to arbitrate in unrestricted mode.
		1	Configures the controller to arbitrate in restricted mode.
7	F__S*	F__S*	Fast/Slow:
		0	Configures the controller as a slow module.
		1	Configures the controller as a fast module.

7.7 CTRL3 (ADD(3:0) = 0101)

Defaults to h'00.

7	6	5	4	3	2	1	0
	TCSEL	IBA__PK*	NCN	ND__S*	NGO	D__S*	G0

Bit	Symbol	Description
0	GO	GO Bit: 1 indicates that all the required initialization is complete and the controller may accept requests. 0 the controller will follow the normal flow of arbitration along with other modules (bystander).
1	D__S*	D__S* Double/Single 0 Configures the controller to arbitrate in a single pass arbitration mode. 1 Configures the controller to arbitrate in a two pass arbitration mode.
2	NGO	A "one" indicates that a new GO bit has been loaded. This bit is reset automatically when the new GO bit takes effect. To be set by the user when the GO bit is changed.
3	ND__S*	A "one" indicates that a new D__S* bit has been loaded. This bit is reset automatically when the new D__S* bit takes effect. To be set by the user when the D__S* bit is changed.
4	NCN	A "one" indicates that a new CN has been loaded. This bit is reset automatically when the new CN takes effect. This bit is set internally when TXCN1 is accessed while the GO bit is set.
5	IBA__PK*	Idle Bus Arbitration/Parking: 1 Configures the controller to do Idle Bus Arbitration. 0 Configures the controller to do Parking.
6	TCSEL	A "one" indicates that a test clock is being fed through the CLK pin to test the timers and counters in the controller.

7.0 Register Description (Continued)

7.8 STATE (ADD(3:0) = 0110)

This register contains the present state of the arbitration controller. This information may be used for testing and debugging purposes. Defaults to h'01.

7	6	5	4	3	2	1	0
		ST5	ST4	ST3	ST2	ST1	ST0

Bit	Symbol	Description
0–5	ST(0:5)	These bits indicate the current phase of the arbitration controller.

7.9 STATUS (ADD(3:0) = 0111)

This register contains status information and internal counter/divider test bits. The M, C, W bits default to zero. The counter test bits may be evaluated when feeding in a test clock.

7	6	5	4	3	2	1	0
M	C	W	CT2	CT1	CT0		

Bit	Symbol	Description
2–4	CT(0:2)	Counter test bits: CT2—set to one when 1 μs timeout occurs. CT1—set to one when divide by 40 divider has divided 40 times. CT0—set to one when divide by n divider has divided n times.
5	W	Win attribute of the module. A "one" indicates that the module is the winner of the current arbitration cycle/pass.
6	C	Competitor attribute of the module. A "one" indicates that the module is competing for the bus.
7	M	Master attribute of the module. A "one" indicates that the module is the current master of the bus.

7.10 RXCN0 (ADD(3:0) = 1000)

This register stores the received pass 1 arbitration number of a two pass number (CN = 0). Defaults to h'00.

7	6	5	4	3	2	1	0
0	P7	P6	P5	P4	P3	P2	P1

Bit	Symbol	Description
0–6	P(1:7)	Priority field of the current master/master elect arbitration number. This register is used only if the controller is configured to arbitrate in Unrestricted two pass mode.
7	0	This field is fixed at zero as specified in the Futurebus+ spec for the pass 1 number of a two pass arbitration.

7.0 Register Description (Continued)

7.11 RXCN1 (ADD(3:0) = 1001)

This register stores the received pass 2 arbitration number of a two pass arbitration number, or the single pass arbitration number, (CN7 = 1) in the Unrestricted mode. Also this register stores the Restricted Mode arbitration number. Defaults to h'10.

7	6	5	4	3	2	1	0
1/P1	P0	RR	U4	U3	U2	U1	U0

Bit	Symbol	Description
0–4	U(0:4)	Uniqueness field of the current master/master elect.
5	RR	Round Robin bit of the current master/master elect.
6	P0	Priority field of the current master/master elect. If configured as a two pass arbitor, P0 is part of P(0:7). If configured as a single pass arbitor, P0 is the priority field. If configured to arbitrate in restricted mode then P0 is part of P(0:1).
7	1/P1	This field is fixed at one as specified in the spec for the pass 2 number of a two pass arbitration, and for the single pass arbitration number. This bit is P1 for the restricted mode arbitration number of the current master/master elect.

7.12 RXMSG (ADD(3:0) = 1010)

This register stores the received non-powerfail arbitration message. Defaults to h'00.

7	6	5	4	3	2	1	0
A7	A6	A5	A4	A3	A2	A1	A0

Bit	Symbol	Description
0–7	A(0:7)	Non-powerfail arbitration message received from another module.

7.13 CLRERI (ADD(3:0) = 1011)

Error bits to indicate Parity, no BRQ/MGRQ or a timeout error. A dummy write into this register clears the error interrupt and all the error bits. Defaults to h'00.

7	6	5	4	3	2	1	0
ER3	ER2	ER1					

Bit	Symbol	Description
7	ER3	Error bit 3: indicates to the winner of the arbitration cycle that BRQ/MGRQ signal is no longer present.
6	ER2	Error bit 2: indicates a timeout error.
5	ER1	Error bit 1: indicates a Parity error.

7.14 CLRMGI (ADD(3:0) = 1100)

7.15 CLRPFI (ADD(3:0) = 1101)

A dummy write into these registers clears the respective interrupt.

7	6	5	4	3	2	1	0

Bit	Symbol	Description
0–7		

7.0 Register Description (Continued)

7.16 REV NO. (ADD(3:0) = 1111)

Revision number of the controller.

7	6	5	4	3	2	1	0
R7	R6	R5	R4	R3	R2	R1	R0

Bit	Symbol	Description
0–7	R(0:7)	REV NO.

8.0 Programming Registers

During power up, the registers should be initially programmed. The host may read/write to/from the register block at any time. The host may select to write data into the register either on the falling edge of DSACK* or on the rising edge of CS*. A zero on the SEL input pin configures the controller to latch in data on the falling edge of DSACK* while a one sets the controller to latch in data on the rising edge of CS*. Refer to the timing diagrams (see *Figures T2a, b, c*).

8.1 HOST WRITE CYCLE USING FALLING EDGE OF DSACK* *(Figure T2A)*

1. R_W* signal is negated.

   ADD(3:0) contains the address of the register to be accessed. (Note: The setup time with respect to CS* must be satisfied.)
2. CS* is asserted (! CS*).
3. When the proper setup time of CS* to the rising edge of clock is met and CS* is low for at least 3 clock cycles, then DATA(7:0) will be latched into the register, by the falling edge of DSACK*, on the 3rd rising clock edge after the assertion of CS*. If the CS* setup time is not met and CS* is low for at least 3 clock cycles, then DSACK* is asserted and the data (DATA(7:0)) is latched into the arbitration register on the following 3rd or 4th rising clock edge after the assertion of CS*. If CS* is negated (CS*) before 3 clock cycles, then DSACK* is not generated and as a default, DATA (7:0) is latched into the arbitration controller register on the rising edge of CS*.
4. Host negates CS*.
5. Arbitration Controller negates DSACK* (DSACK*) if it was asserted.

8.2 HOST WRITE CYCLE USING RISING EDGE OF CS* *(Figure T2b)*

1. R_W* signal is negated.

   ADD(3:0) contains the address of the register to be accessed. (Note: The setup time with respect to CS* must be satisfied.)
2. CS* is asserted (! CS*).
3. DSACK* is asserted (! DSACK*) by the Arbitration Controller when CS* is asserted for at least three clock cycles. If CS* is negated (CS*) before three clock cycles, DSACK* is not asserted. The DSACK* signal may be used or ignored by the system designer, in this case.
4. Host negates CS*. DATA (7:0) which satisfied the setup time to the rising edge of CS* is latched into the arbitration controller register.
5. Arbitration Controller negates DSACK* (DSACK*) if it was asserted.

8.3 HOST READ CYCLE *(Figure T2c)*

1. R_W* signal is set high.

   ADD (3:0) contains the address of the register to be accessed. (Note: The setup time with respect to CS* must be satisfied.)
2. CS* is asserted (! CS*).
3. The data will be available on the DATA (7:0) bus within the access time specified in the AC timing section.
4. The Data Strobe ACKnowledge (DSACK*) signal is generated as an acknowledge to the host signifying the validity of the accessed data. DSACK* may be used to insert WAIT states to the host during a host read cycle. When the proper setup time of CS* to the rising edge of clock is met and CS* is low for at least three clock cycles, DSACK* is asserted (! DSACK*) on the 3rd rising clock edge after the assertion of CS*. If the CS* setup time was not met and CS* is low for at least three clock cycles, then DSACK* is asserted on the following 3rd or 4th rising clock edge after the assertion of CS*. If CS* is not low for at least 3 clock cycles, DSACK* is not generated.
5. Host reads data and negates CS* (CS*).
6. Arbitration Controller negates DSACK* (DSACK*) if it was asserted.

9.0 Clock/Timer/Delay Lines

(See *Figure 9.*) The input clock signal (CLK) to the arbitration controller is assumed to be a clock from 2 MHz to 40 MHz, in steps of 1 MHz.

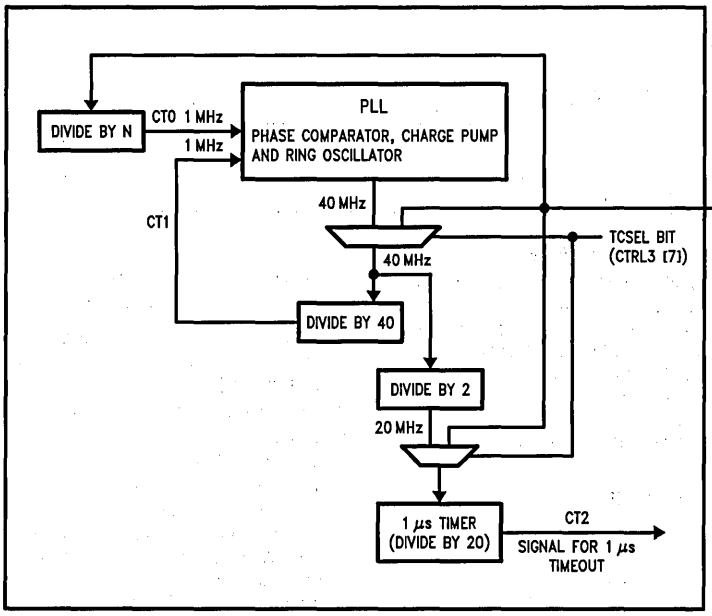
The clock signal is also used for synchronization purposes during read or write transfers. This is accomplished by synchronizing the DSACK* (Data Strobe Acknowledge) output from the Arbitration Controller to the clock during arbitration controller register reads or writes.

The binary value of the input clock (CLK) frequency is loaded into the CTRL1 [5:0] register. This value is used to program the divide by n counter to scale the input clock down to a 1 MHz clock internally. The PLL ring oscillator also has a clock divider (divide by 40) to scale it down to a 1 MHz clock. These two clocks are compared and the difference between the two clocks is fed back to the ring oscillator to cause it to lock onto the appropriate frequency.

The PLL is used to generate several programmable delay lines and the 1 μs Timer used during phase 2 and phase 4.

The 1 μs timer, divide by 40 divider and divide by n divider can be tested to determine proper functionality. Refer to Testing the Arbitration Controller and Register Description sections for details.

9.0 Clock/Timer/Delay Lines (Continued)



TL/H/10747–24

FIGURE 9. PLL, Timer and Test Circuitry

10.0 Reset/Initialization/Power Up

Two distinct input reset pins are available on the Arbitration Controller, RINT* (Bus Reset and Initialization) and RST* (Power Up Reset). These signals are activated by the module upon detecting either RE* (in an appropriate condition) on Futurebus+ or reset from the host or the Protocol Controller. Both these signals are asynchronous inputs so that the reset function is performed immediately after the signal is asserted.

RINT* may be asserted by external logic from RE* being asserted for at least 2 ms or some other appropriate condition. When RINT* is asserted (! RINT*):

1. the arbitration controller is placed in phase 0
2. all outputs of the arbitration controller are negated, except ARO
3. The following registers are cleared: RXCN0, RXCN1, RXMSG, CLRERI, CLRMGI, CLRPFI, and M C W bits of the STATUS register
4. ST0 bit is set in the STATE register
5. Contents of other registers remain, except for the round robin (RR) bit of TXCN1 which is reset.

The rising edge of RINT* will release the arbiter from phase 0. This reset signal should be used during Futurebus+ initialization.

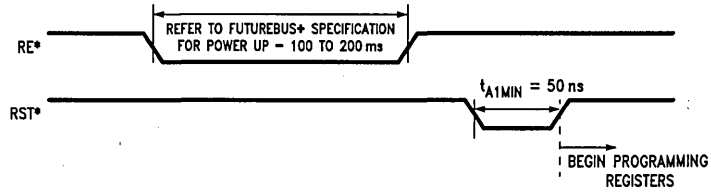
When RST* is asserted (! RST*):

1. all outputs of the arbitration controller are negated
2. the registers are set to default values as given in the register description, Section 7.

The rising edge of RST* will cause ARO to be asserted causing the arbiter to be in phase 0. This reset signal should be used during power-up and live withdrawal.

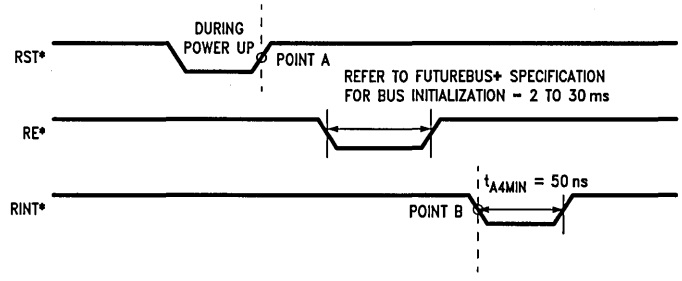
Figure 10 illustrates how the reset and initialization signals may be used during power up. While powering up, when RINT* is asserted for 100 ms–200 ms, the registers may be programmed. BRQ* should not be issued until after 10 ms after the register CTRL1 is programmed giving the PLL a sufficient time to lock onto the CLK signal.

10.0 Reset/Initialization/Power Up (Continued)



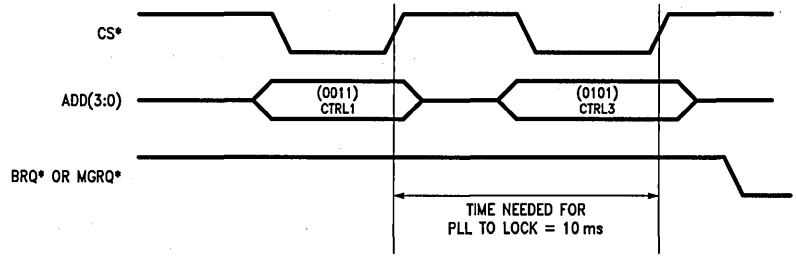
TL/H/10747–25

FIGURE 10a. Programming Registers: After Power-Up



TL/H/10747–26

FIGURE 10b. Programming Registers: During Bus Initialization



TL/H/10747–27

FIGURE 10c. PLL Lock Period

11.0 Live Insertion

The arbitration Controller supports live insertion (see *Figure 11*). When a module is being live inserted into an active Futurebus+ system it will be powered up and reset. The live inserted module will reset it's arbitration controller with the RST* input and will drive re* on the Futurebus+ backplane.

RE* asserted will cause all active Futurebus+ modules to release their bus lines to prepare for live insertion to the Futurebus+ backplane.

When all the active Futurebus+ modules detect RE* asserted, they should assert HALT* to their arbitration controller and limit the current parallel bus transaction to 128 μs. The active modules will then remain in Phase 0, upon completion of the current control acquisition cycle, until HALT* is negated.

When the live inserted module detects both:

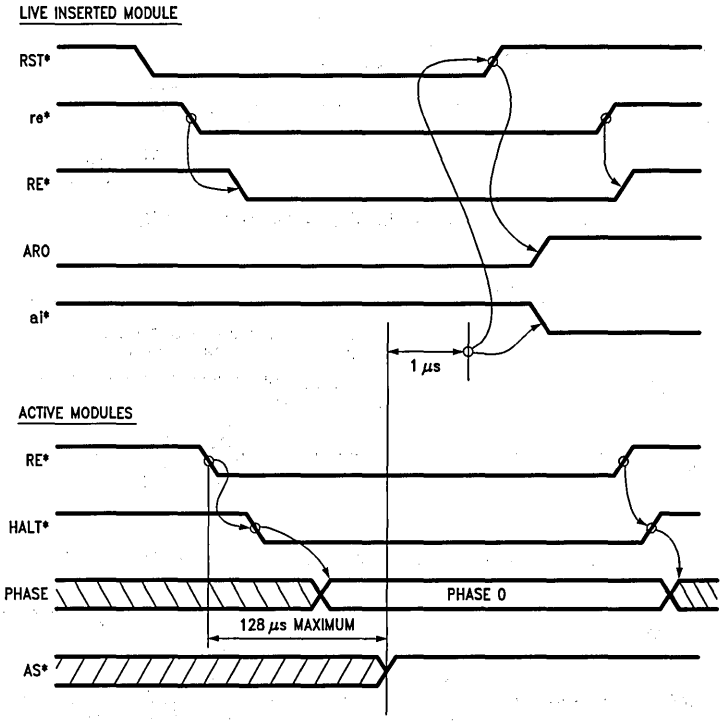
1. the Parallel bus in the Idle state (AS* and AKf negated),
2. the arbitration bus lines in Phase 0

for 1 μs then it should assert ai* (protocol controller) and negate the RST* input. The arbitration controller will assert ARO upon detecting the negation of RST*. These actions will complete the Futurebus+ alignment. The live inserted module will then be in arbitration Phase 0; thus, aligned with the other live modules. Then the live inserted module will negate re*.

When the active modules detect the negation of RE* they negate the HALT* input to their arbitration controllers, thus allowing arbitration competition to proceed.

At this point, the live inserted module can only participate as a by-stander during the arbitration competition cycle until it is programmed and 10 ms has elapsed (PLL lock on time). Once the 10 ms has elapsed the module can enable arbitration competition by setting the GO bit in CTRL3. At this time bus requests (BRQ*, MGRQ*) can be accepted to join competition for the parallel bus.

11.0 Live Insertion (Continued)



TL/H/10747–28

FIGURE 11. Live Insertion

12.0 Live Withdrawal

When a module is notified to be live withdrawn it will:

1. complete any tasks currently in progress
2. inhibit itself from participating in any more bus transactions (becomes a by-stander)

The module can withdraw from the arbitration protocol in one of three phases:

1. During phase 0 by negating ARO
2. During phase 2 by negating APO
3. During phase 4 by negating AQO

This may be achieved in the DS3875 by issuing the HALT* signal, then when it is observed that the arbitration controller is in Phase 0, asserting the RST* signal to cause all outputs of the arbitration bus to be released. The module may then be withdrawn from the Futurebus+ system.

13.0 Testing the DS3875

The Arbitration Controller has features designed in which ease the testing and monitoring tasks for the user. Registers may be read to determine proper functionality of the chip. For instance, while the Arbitration Controller is operating, the arbitration phase can be monitored by reading the state register. Also to check proper operation of the 1 μs timer, divide by n divider, and divide by 40 divider, the TCSEL bit in CTRL3 register can be set so that a test clock may be used. This will allow the clock signal to be applied to the circuitry, thus disabling the internally generated signals that are used during normal operation. See *Figure 10.* To determine the status of the timer and dividers, the status register CT(0:2) bits can be read.

If the user wants to check the proper operation of the CN port:

1. An arbitration number can be loaded into the CN port by performing a write cycle to any of the receive registers (RXCN0, RXCN1, or RXMSG). The Arbitration Controller will load the number from the CN port instead of the DATA port into the register upon detecting a write to any of the receive registers. The timing is the same as those given for the register write using the DATA port. Refer to Timing Section T2.
2. Next, by performing a read cycle to the register just written to (RXCN0, RXCN1, or RXMSG), the arbitration number stored will be placed onto the DATA port. Thus, the proper operation of the CN port is tested.

14.0 Electrical Characteristics

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6.5V
Control Input Voltage	5.5V
Power Dissipation at 70°C	0.6W
Storage Temperature Range	−65°C to +150°C
Lead Temperature	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage, V_{DD}	4.5	5.5	V
Operating Free Air Temperature	0	70	C

Electrical Characteristics (Notes 2 and 3) T_A = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Minimum Input High Voltage		2.0			V
V_{IL}	Minimum Input Low Voltage				0.8	V
V_{OH}	Voltage Output High	I_{OH}, I_{OL} for Several Drivers i.e., I_{OL} 8 mA and 4 mA	2.4	3.2		V
V_{OL}	Voltage Output Low	I_{OH}, I_{OL} for Several Drivers i.e., I_{OL} 16 mA and 4 mA		0.35	0.5	V
I_I	Input Leakage Current	Input at V_{DD} or V_{SS}	−1.0		1.0	µA
I_{IH}	Input High Current	Input at V_{IH}			1.0	µA
I_{IL}	Input Low Current	Input at V_{IL}	−1.0			µA
I_{DD}	Supply Current	Dynamic Supply Current			100	mA
I_{CC}	Static Supply Current	Input at Standby			30	mA

Note 1: "Absolute maximum ratings" are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provide conditions for actual device operation.

Note 2: All currents into device pins are positive; all currents out of device pin are negative. All voltages are referenced to device ground unless otherwise specified.

Note 3: All typicals are given for V_{DD} = 5V, and T_A = 25°C.

15.0 AC Parameters

Legend to AC Parameter Number Assignments

Parameter Number	Description
t000–t099	Phase 0
t100–t199	Phase 1
t200–t299	Phase 2
t300–t399	Phase 3
t400–t499	Phase 4
t500–t599	Phase 5
tAXX	Reset, Initialization
tBXX	Register Access Data Port
tCXX	Register Access CN Port - Input
tDXX	Clearing Interrupts
tEXX	FIFO Strobe
tFXX	WIN*__GT* Valid
tGXX	Message Signals
tHXX	Busrequest, Busgrant, End of Tenure
tJXX	Locked, Unlock Handshake Signals

15.0 AC Parameters (Continued)

AC Timing Parameters Unless otherwise stated: T_A = 0°C to +70°C, V_{CC} = 5V ±10%

All transitions are specified after the input signals are stable and valid for evaluation. This table will describe the parameters as given in the following pages. All values are given in nanoseconds (ns), unless otherwise stated.

Number	Symbol	Parameter Description	Min	Typ	Max
t_1	t_{BQBG}	BRQ* Asserted to BGRNT* Asserted		19	33
t_2	t_{AQAC}	AQI Negated to AC0O, AC1O Negated		22	30
t_3	t_{APIO}	API Asserted to APO Asserted		16	26
t_4	t_{RQAP}	MGRQ* or BRQ* Asserted to APO Asserted		19	32
t_5	t_{EDAP}	(Dummy Cycle) ENDT* Asserted to APO Asserted		28	40
t_6	t_{BQAP}	(Consecutive Bus Requests) BRQ* Asserted to APO Asserted		17	28
t_7	t_{HTAP}	HALT* Negated to APO Asserted		16	26
t_{100}	t_{CNLEAR}	CN__LE* Negated to ARO Negated. Determined by Programmable Value		2+(0−25)	4+(0−25)
t_{101}	t_{CNLE}	CN__LE* Width	18	20	24
t_{102}	t_{CNS}	CN Port Setup Time	23	28	
t_{103}	t_{CNZ}	ARI Negated to TRI-STATE CN Port	16	20	
t_{104}	t_{IBCAR}	(IBA Mode) IBA-CMPT* Asserted to ARO Negated. Determined by Programmable Value	20+(0−25)	30+(0−25)	
t_{105}	t_{CPTAR}	CMPT* Asserted to ARO Negated. Determined by Programmable Value	20+(0−25)	30+(0−25)	
t_{106}	t_{AC0AR}	(Slow Mode) AC0O Asserted to ARO Negated. Determined by Programmable Value	20+(0−25)	30+(0−25)	
t_{107}	t_{APCNLE}	APO Asserted to CN-LE* Asserted. CTRL3[0], "G0" Bit is Set		22	32
t_{108}	t_{APCPTA}	APO Asserted to CMPT* Asserted		14	22
t_{109}	t_{APIBC}	APO Asserted to IBA__CMPT* Asserted		13	18
t_{110}	t_{APAC}	APO Asserted to AC0O Asserted (Slow Mode)		18	28
t_{200}	t_{ARABRD}	ARI Negated to AB__RD* Asserted		52	62
t_{201}	t_{ARIBS}	(IBA Mode) ARI Negated to IBA__S*. T_A = Arbitration Timer 60-300		T_A	T_A
t_{202}	t_{IBSBG}	(IBA Mode) IBA__S* Asserted to BGRNT* Asserted		19	32
t_{203}	t_{WINAQ}	WIN*__GT* Asserted to AQO Asserted. After T_A Expired.		18	28
t_{204}	t_{AQIO}	AQI Asserted to AQO Asserted		13	19
t_{205}	t_{ARAQ}	ARI Negated to AQO Asserted		18+T_A	28+T_A
t_{207}	t_{IBSAQ}	(IBA Mode) IBA-S* Asserted to AQO Asserted		20	33
t_{300}	t_{AC10AP}	AC10 Asserted to APO Negated		15	24
t_{301}	t_{ASNAP}	AS__Cancel* Negated to APO Negated		15	24
t_{310}	t_{APABRD}	APO Negated to AB__RD* Negated		5	8
t_{320}	t_{AQAC1}	AQO Asserted to AC1O Asserted		6	10
t_{321}	$t_{AC1IAPN}$	AC1I Asserted to APO Negated	7	12	18
t_{322}	t_{ASAAP}	AS__Cancel* Asserted to APO Negated	7	12	18
t_{330}	t_{AC1IAP}	AC1I Asserted to APO Negated		13	22
t_{340}	t_{RQAC}	MGRQ* or BRQ* Negated to AC0O, AC1O Asserted		21	30
t_{341}	t_{AQAC0}	AQO Asserted to AC0O Negated		0	0
t_{342}	t_{AQER}	AQO Asserted to ERIT* Asserted		14	22
t_{400}	t_{APCPTN}	API Negated to CMPT* Negated		18	30

15.0 AC Parameters (Continued)

AC Timing Parameters Unless otherwise stated: T_A = 0°C to +70°C, V_{CC} = 5V ±10% (Continued)

All transitions are specified after the input signals are stable and valid for evaluation. This table will describe the parameters as given in the following pages. All values are given in nanoseconds (ns), unless otherwise stated.

Number	Symbol	Parameter Description	Min	Typ	Max
t_{401}	t_{AC1IAR}	AC1I Asserted to ARO Asserted		12	18
t_{403}	t_{CANAC1}	AS_Cancel* Asserted to AC1O Asserted	12		18
t_{405}	t_{ARIO}	ARI Asserted to ARO Asserted		14	19
t_{407}	t_{EDAR}	ENDT* Asserted to ARO Asserted		15	24
t_{500}	t_{ARBG}	ARO Asserted to BGRNT* Asserted		7	12
t_{501}	t_{OAQ}	BGRNT*, MGTX*, UNLK* or any Interrupt Asserted to AQO Negated	5	7	
t_{502}	t_{ARFTR}	ARO Asserted to FSTR* Negated		5	8
t_{503}	t_{ARIBC}	ARO Asserted to IBA_CMPT* Negated		5	8
t_{504}	t_{ARABSA}	ARO Asserted to MGTX*, UNLK* or any Interrupt Asserted		5	9
t_{A1}	t_{RSTPW}	RST* Pulse Width	50	50	
t_{A2}	t_{RSTRE}	Output Reset Time		30	45
t_{A3}	t_{RSTAR}	RST* Negated to ARO Asserted		15	25
t_{A4}	t_{RINTPW}	RINT* Pulse Width	50	50	
t_{A5}	t_{RINTRE}	Output Initialization Reset Time		30	45
t_{B1}	t_{CSPW}	CS* Pulse Width	35		
t_{B2}	t_{CSPWN}	CS* Recovery Time	15		
t_{B3}	t_{CSDKA}	CS* Asserted to DSACK* Asserted		12	18
t_{B4}	t_{ADDS}	ADD (3:0) Setup Time	5	2	
t_{B5}	t_{ADDH}	ADD (3:0) Hold Time	6	5	
t_{B6}	t_{CSDKN}	CS* Negated to DSACK* Negated	7	12	
t_{B7}	t_{RWS}	R_W* Setup Time	0		
t_{B8}	t_{SELS}	SEL Setup Time	0		
t_{B10}	$t_{DATASDK}$	Data (7:0) Setup Time with Respect to DSACK*	20	15	
t_{B11}	$t_{DATAHDK}$	Data (7:0) Hold Time with Respect to DSACK*	5	5	
t_{B12}	t_{SELH}	SEL Hold Time	0		
t_{B24}	$t_{DATASCS}$	Data (7:0) Setup Time with Respect to CS* Negated	15	6	
t_{B25}	$t_{DATAHCS}$	Data (7:0) Hold Time with Respect to CS* Negated	5	5	
t_{B27}	t_{DATALZ}	Data (7:0) Low Z Time		14	22
t_{B28}	t_{DATAV}	Data (7:0) Valid Time before DSACK*			
t_{B29}	t_{DATAA}	Data (7:0) Access Time with Respect to CS* Asserted		22	30
t_{B30}	$t_{DATAHRC}$	Data (7:0) Hold Time	6	10	
t_{B31}	t_{DATAHZ}	Data (7:0) Hi-Z Time		12	25
t_{C2}	$t_{ABRDCNZ}$	AB_RD* Negated to CN (7:0) TRI-STATE	0	0	
t_{D1}	$t_{CSXINTN}$	CS* Asserted to any Interrupt Negated		22	36
t_{E1}	t_{FSTR}	FSTR* Recovery Time	80		

15.0 AC Parameters (Continued)

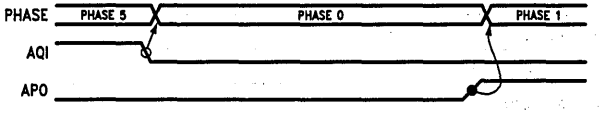
AC Timing Parameters Unless otherwise stated: T_A = 0°C to +70°C, V_{CC} = 5V ±10% (Continued)

All transitions are specified after the input signals are stable and valid for evaluation. This table will describe the parameters as given in the following pages. All values are given in nanoseconds (ns), unless otherwise stated.

Number	Symbol	Parameter Description	Min	Typ	Max
t_{E2}	$t_{FSTABRD}$	AB__RD* High Time with Respect to FSTR*	60		
t_{F1}	$t_{WINALL1}$	ALL1* Asserted with Respect to WIN*__GT*			5
t_{F2}	t_{WINPER}	PER* Asserted with Respect to WIN*__GT*			5
t_{G1}	t_{MGXN}	MGRQ* Negated to MGTX* Negated		14	22
t_{H2}	t_{EDBGN}	ENDT* Asserted to BGRNT* Negated		17	25
t_{J1}	t_{LKULKN}	LKD* Negated to UNLK* Negated		12	18

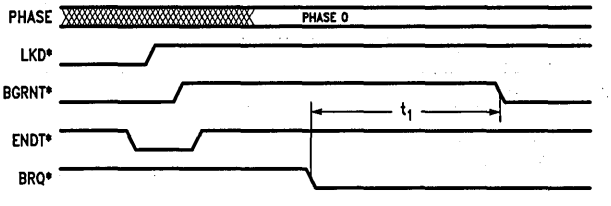
Phase 0 Timing

A. Transitioning Into and Out of Phase 0



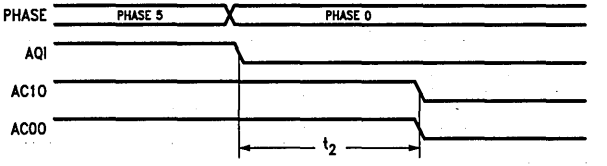
TL/H/10747–31

B. Parking (Assume I IBA__PK*)



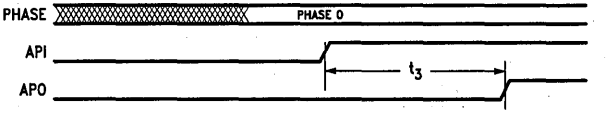
TL/H/10747–32

C. Negating AC10 and AC00 When Entering Phase 0



TL/H/10747–33

D. Another Module Initiating Competition

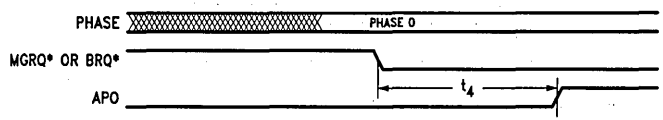


TL/H/10747–34

15.0 AC Parameters (Continued)

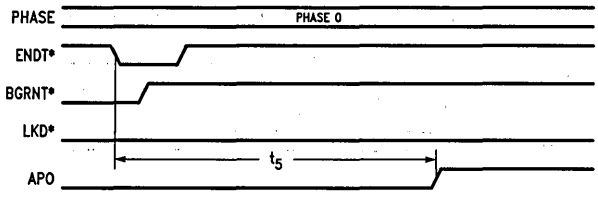
Phase 0 Timing (Continued)

E. Message Request or Bus Request Initiating Competition



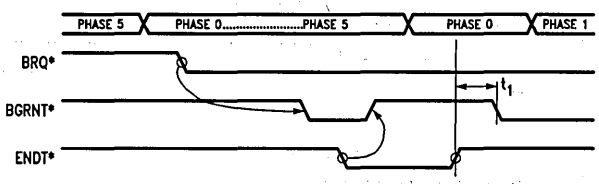
TL/H/10747–35

F. Dummy Cycle to Generate Unlock (I UNLK*) During Phase 5



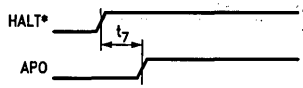
TL/H/10747–36

G. Consecutive Bus Requests: Parking



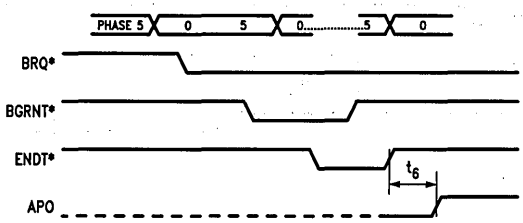
TL/H/10747–37

H. HALT* Release



TL/H/10747–75

I. Consecutive Bus Requests: Non-Parking

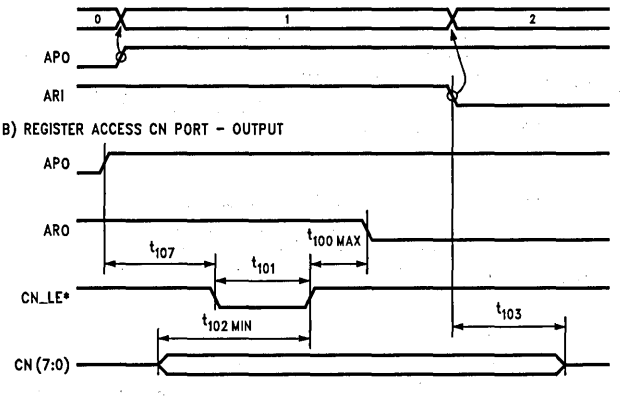


TL/H/10747–78

15.0 AC Parameters (Continued)

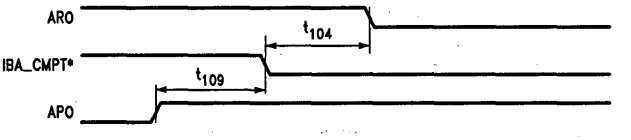
Phase 1 Timing

A) TRANSITIONING INTO AND OUT OF PHASE 1



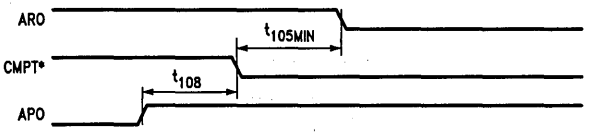
TL/H/10747–38

C. IBA Mode Selected



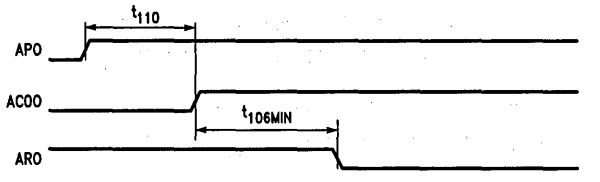
TL/H/10747–39

D. Competing



TL/H/10747–40

E. Asserting ACOO - Slow Mode (F__S*) Is Selected

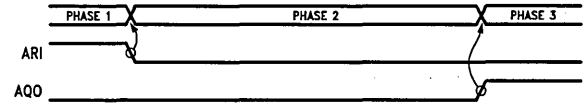


TL/H/10747–41

15.0 AC Parameters (Continued)

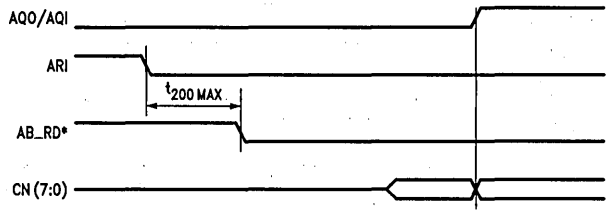
Phase 2 Timing

A. Transitioning Into and Out of Phase 2



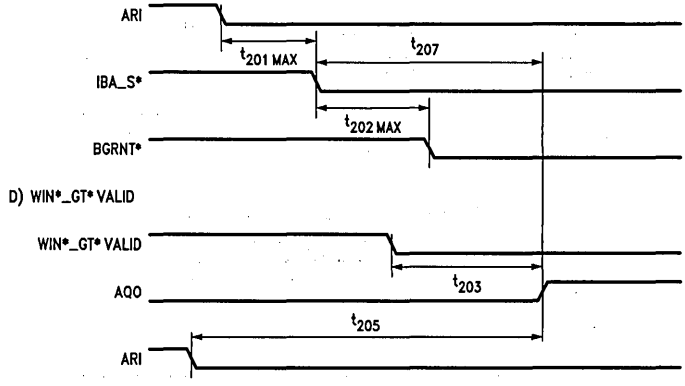
TL/H/10747–42

B. Register Access CN Port–Input



TL/H/10747–43

C. IBA Mode Selected–Winner of Competition



TL/H/10747–44

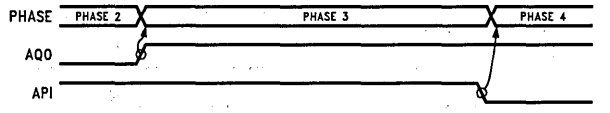
E. Bystander

TL/H/10747–45

15.0 AC Parameters (Continued)

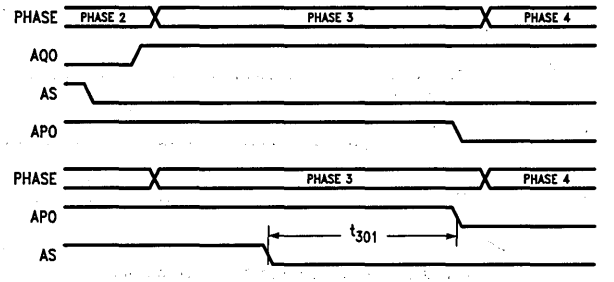
Phase 3 Timing

A. Transitioning Into and Out of Phase 3



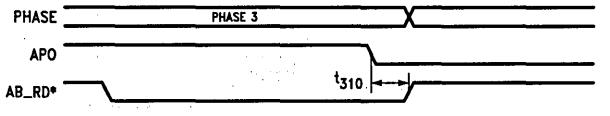
TL/H/10747–46

B. AS Low and No Errors



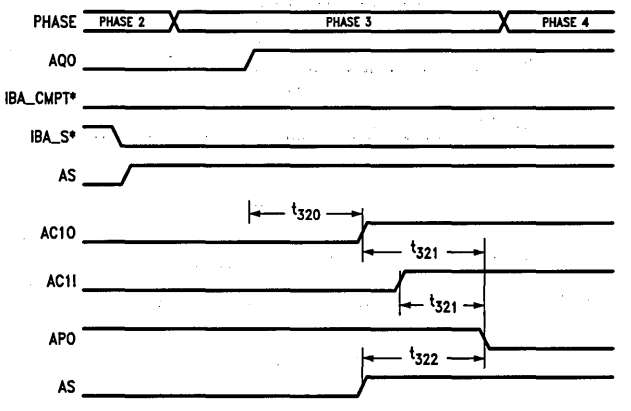
TL/H/10747–47

C. Reading the Winning Competition Number (I AB_RD*)



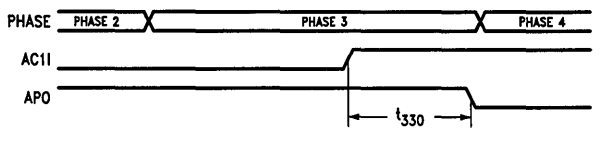
TL/H/10747–48

D. Successful IBA Competition Timing



TL/H/10747–49

E. Transfer of Tenure Inhibited

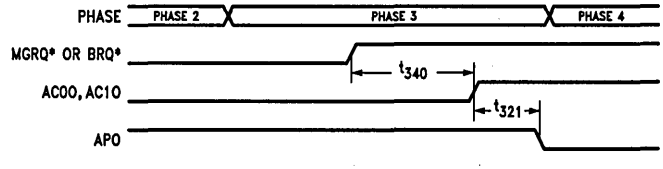


TL/H/10747–50

15.0 AC Parameters (Continued)

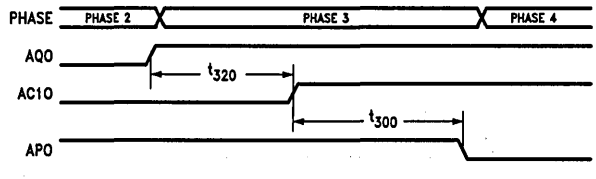
Phase 3 Timing (Continued)

F. Message Request or Bus Request Being Negated During Phase 3 (If This Module was the Winner)



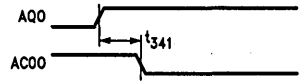
TL/H/10747–51

G. Second Pass Required



TL/H/10747–52

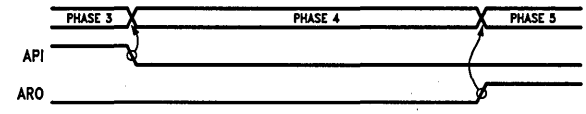
H. Slow Case ACO Recovery



TL/H/10747–76

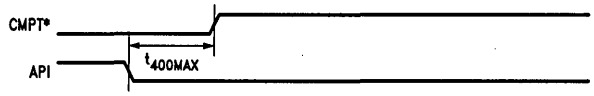
Phase 4 Timing

A. Transitioning Into and Out of Phase 4



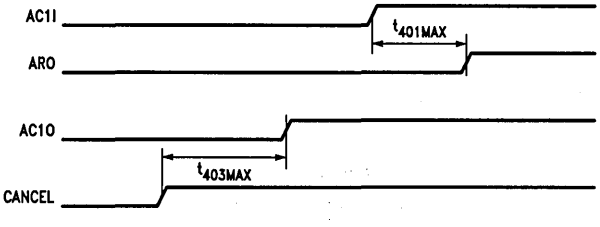
TL/H/10747–53

B. Complete Signal Is Negated



TL/H/10747–54

C. Inhibit Transfer of Tenure

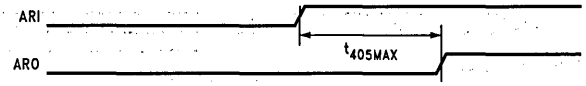


TL/H/10747–55

15.0 AC Parameters (Continued)

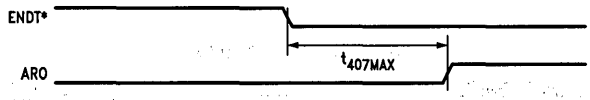
Phase 4 Timing (Continued)

D. Module (Except Master Module) Receives ARI



TL/H/10747–56

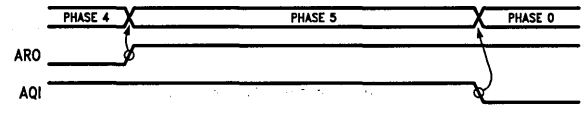
E. End of Tenure



TL/H/10747–57

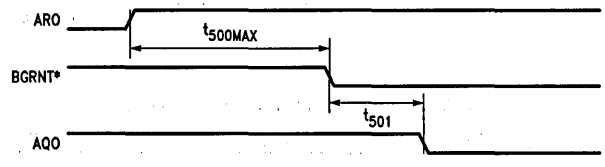
Phase 5 Timing

A. Transitioning Into and Out of Phase 5



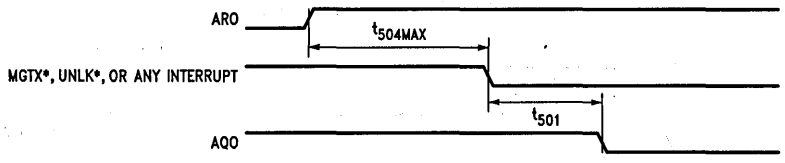
TL/H/10747–58

B. Master Elect Gets Bus Grant When No Errors Occurred



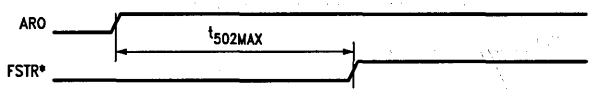
TL/H/10747–59

C. Message was Transmitted (During 2nd Pass–No Errors Occurred), Generation of Interrupt Signal, PFINT*, or MGINT*, Generation of Unlock Signal When Locked (! LKD*)



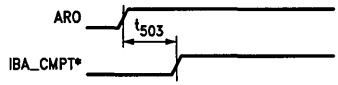
TL/H/10747–60

D. FIFO Strobe



TL/H/10747–61

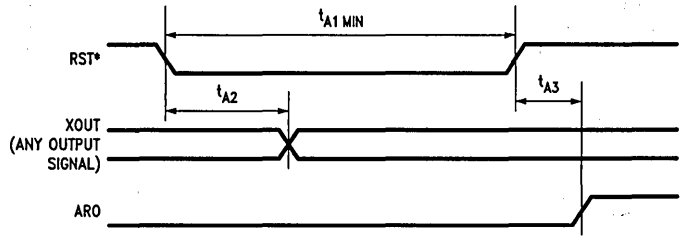
E. Release of IBA_CMPT*



TL/H/10747–77

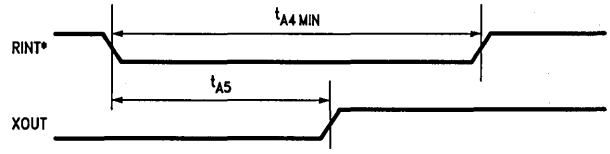
15.0 AC Parameters (Continued)

T1a. RST* Signal



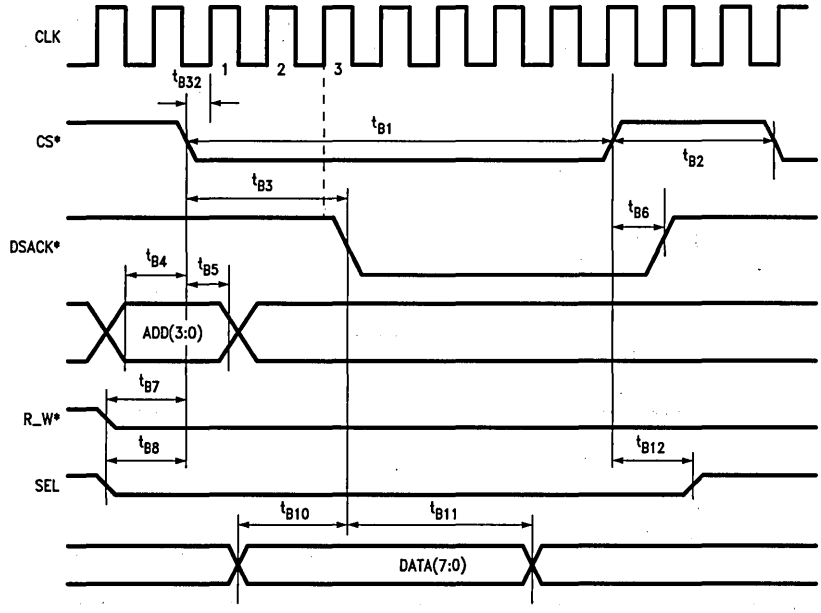
TL/H/10747–62

T1b. RINT* Signal



TL/H/10747–63

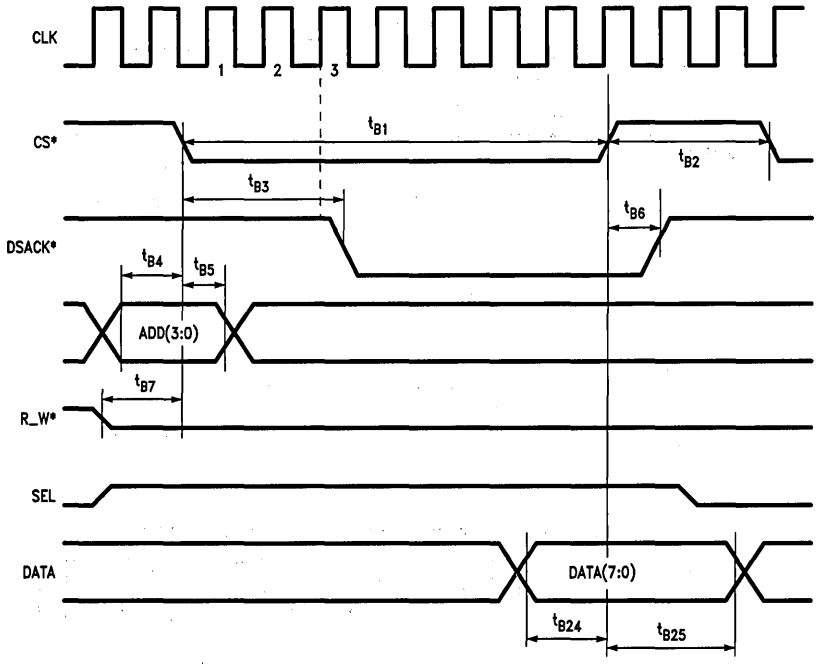
T2a. Register Access DATA Port– WRITE CYCLE USING DSACK*



TL/H/10747–64

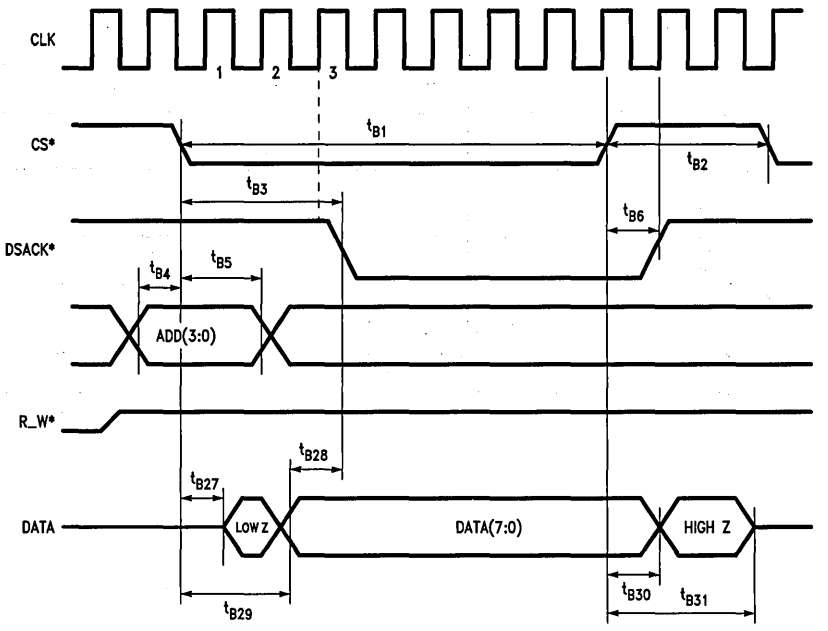
15.0 AC Parameters (Continued)

T2b. Register Access DATA Port–Write CYCLE Using CS*



TL/H/10747–65

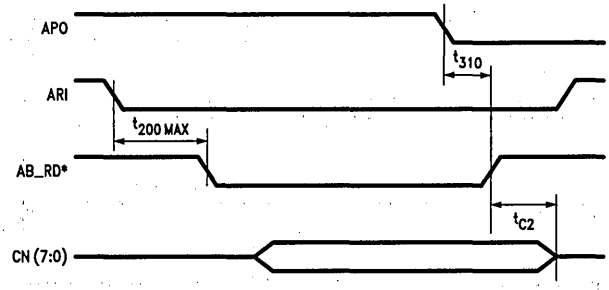
T2c. Register Access DATA Port–Read CYCLE



TL/H/10747–66

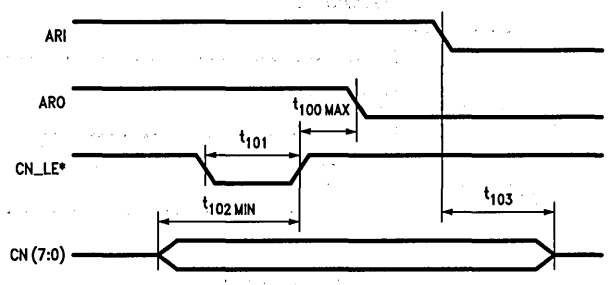
15.0 AC Parameters (Continued)

T3. Register Access CN Port–Input



TL/H/10747–67

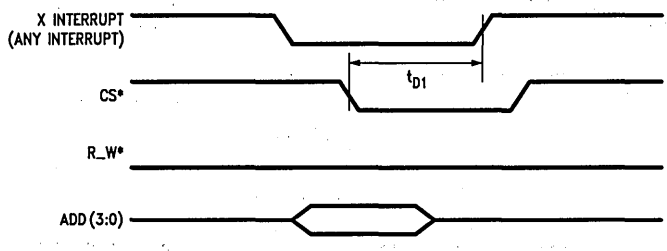
T4. Register Access CN Port–Output



TL/H/10747–68

These sections are shown to give a better understanding of the CN port timing. Most of the parameters are given under the phase timing diagrams. Also a few parameters are added here to these diagrams.

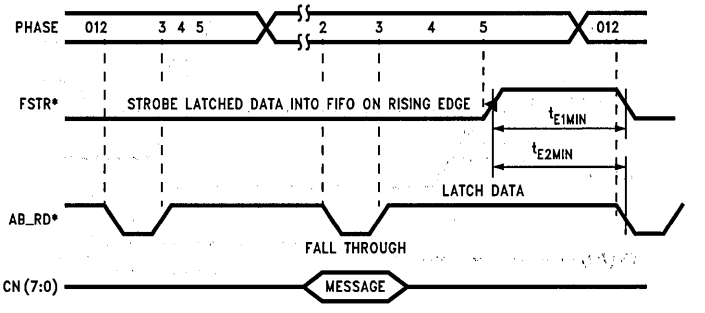
T5. Clearing Interrupts



TL/H/10747–69

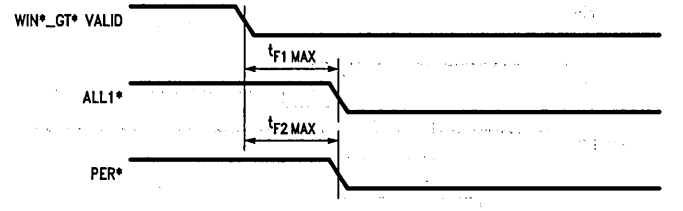
15.0 AC Parameters (Continued)

T6. FIFO Strobe



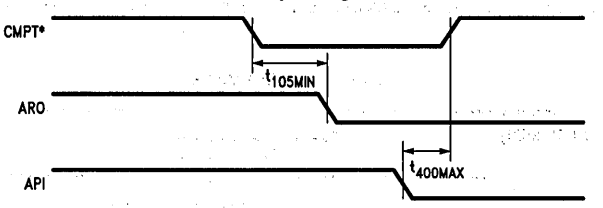
TL/H/10747–70

T7. WIN*_GT* Valid

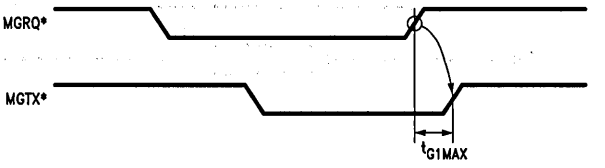


TL/H/10747–71

T8. Compete Signal



T9. MESSAGE SIGNALS

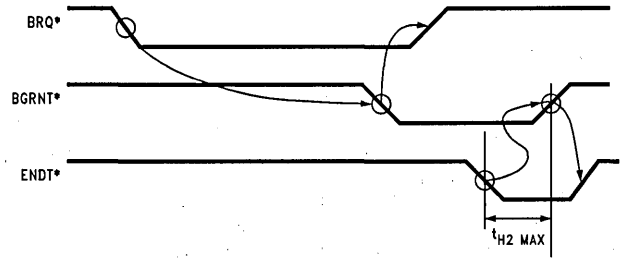


TL/H/10747–72

These sections give some additional parameters, these are not the complete set of parameters specified for these signals. Also refer to the phase timing diagrams.

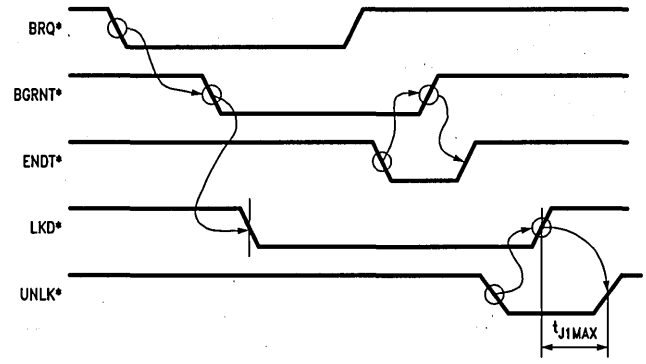
15.0 AC Parameters (Continued)

T10. BUSREQUEST, BUSGRANT and END of TENURE HANDSHAKE



TL/H/10747–73

T11. LKD* and UNLK* Handshake Signals



TL/H/10747–74

National Semiconductor

PRELIMINARY

DS3883 BTL 9-Bit Data Transceiver

General Description

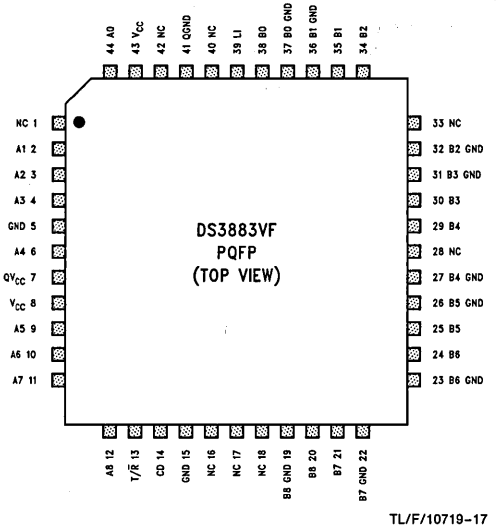
The DS3883 BTL 9-Bit Data Transceiver is designed to conform to IEEE P1194.1 (BTL) as specified in IEEE P896.2 (Futurebus+). The DS3883 simplifies the implementation of all byte wide address/data with parity signals. It can also be used for the status, tag and command lines.

The DS3883 is one of five current devices in the Futurebus+ Chip Set. The DS3886 BTL 9-Bit Latching Transceiver gives the user an option of having latches within the transceiver. The DS3884 Handshake Transceiver has a pulse width selectable wired-OR glitch filter. The DS3885 Arbitration Transceiver has the competition logic for the AB<8:0> signal lines. The DS3875 Arbitration Controller supports all the required and optional modes for the Futurebus+ arbitration protocol. It is designed to be used in conjunction with the DS3884 and DS3885.

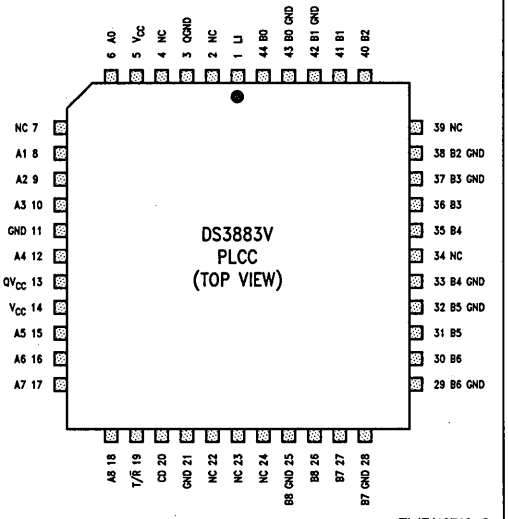
Features

- Meets P1194.1 standard for Backplane transceiver logic
- Pinout is designed specifically for implementing a Futurebus+ system
- Supports live insertion
- Low skew
- Controlled rise/fall time
- Glitch free power-up/down protection
- Built-in bandgap reference provides very accurate thresholds
- 44-pin PLCC
- 44-pin PQFP reduces board real-estate

Connection Diagrams



TL/F/10719-17



TL/F/10719-2

Order Number DS3883V or DS3883VF
See NS Package Number V44A or VF44B

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6.5V
Control Input Voltage	6.5V
Driver Input and Receiver Output	6.5V

Note: At no time including power-up shall the control input, driver input or receiver output exceed V_{CC} + 1V. (See Note 2.)

Receiver Input and Driver Output	3V
Power Dissipation at 70°C	1.22W
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 4 Seconds)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage, V_{CC}	4.5	5.5	V
Bus Termination Voltage (V_T)	2.06	2.14	V
Operating Free Air Temperature	0	70	°C

DC Electrical Characteristics (Notes 2, 3 and 4) T_A = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DRIVER AND CONTROL INPUT: (CD, T/$\overline{R}$, An)						
V_{IH}	Minimum Input High Voltage		2.0			V
V_{IL}	Maximum Input Low Voltage				0.8	V
I_{IH}	Input High Current	V_{IN} = 2.4V			40	µA
I_{IL}	Input Low Current	V_{IN} = 0.5V			−100	µA
V_{CL}	Input Diode Clamp Voltage	I_{Clamp} = −12 mA			−1.2	V
DRIVER OUTPUT/RECEIVER INPUT: (Bn)						
V_{OLB}	Output Low Bus Voltage (Note 5)	An = T/$\overline{R}$ = 2.4V, I_{OL} = 80 mA	0.75	1.0	1.1	V
I_{OLBZ}	Output Low Bus Current	An = 0.5V, T/$\overline{R}$ = 2.4V, Bn = 0.75V			−250	µA
I_{OHBZ}	Output High Bus Current	An = 0.5V, T/$\overline{R}$ = 2.4V, Bn = 2.1V			100	µA
		An = 0.5V, T/$\overline{R}$ = 2.4V, Bn = 2.1V, V_{CC} = 0V			100	µA
V_{TH}	Receiver Input Threshold	T/$\overline{R}$ = CD = 0.5V	1.47	1.55	1.62	V
V_{CLP}	Positive Clamp Voltage	V_{CC} = Max or 0V, Bn = 1 mA	2.4	3.4	4.5	V
		V_{CC} = Max or 0V, Bn = 10 mA	2.9	3.9	5.0	V
V_{CLN}	Negative Clamp Voltage	I_{Clamp} = −12 mA			−1.2	V
RECEIVER OUTPUT: (An)						
V_{OH}	Voltage Output High	Bn = 1.1V, T/$\overline{R}$ = CD = 0.5V, I_{OH} = −2 mA	2.4	3.2		V
V_{OL}	Voltage Output Low	T/$\overline{R}$ = CD = 0.5V, Bn = 2.1V, I_{OL} = 12 mA		0.35	0.5	V
		T/$\overline{R}$ = CD = 0.5V, Bn = 2.1V, I_{OL} = 8 mA		0.35	0.4	V
I_{OS}	Output Short Circuit Current	Bn = 1.1V, T/$\overline{R}$ = CD = 0.5V	−40	−70	−100	mA
SUPPLY CURRENT						
I_{CC}	Supply Current: Includes	T/$\overline{R}$ = All An Inputs = 2.4V, CD = 0.5V		75	100	mA
	V_{CC}, QV_{CC} and LI	T/$\overline{R}$ = 0.5V		60	90	mA
I_{LI}	Live Insertion Current	T/$\overline{R}$ = 0.5V		1.5	3	mA
		T/$\overline{R}$ = All An = 2.4V		3	5	mA

Note 1: "Absolute Maximum Ratings" are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All input and/or output pins shall not exceed V_{CC} + 1 and shall not exceed the absolute maximum rating at anytime, including power-up and power down. This prevents the ESD structure from being damaged due to excessive currents flowing from the input and/or output pins to QV_{CC} and V_{CC}. There is a diode between each input and/or output to V_{CC} which is forward biased when incorrect sequencing is applied. Alternatively, a current limiting resistor can be used when pulling-up the inputs to prevent damage. The current into any input/output pin shall be no greater than 50 mA. Exception, LI and Bn pins do not have power sequencing requirements with respect to V_{CC} and QV_{CC}. Furthermore, the difference between V_{CC} and QV_{CC} should never be greater than 1V at any time including power-up.

Note 3: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 4: All typical values are specified under these conditions: V_{CC} = 5V and T_A = 25°C, unless otherwise stated.

Note 5: Referenced to appropriate signal ground. Do not exceed maximum power dissipation of package.

AC Electrical Characteristics T_A = 0°C to +70°C, V_{CC} = 5V ±10% (Note 6)

Symbol	Parameter		Conditions	Min	Typ	Max	Units
DRIVER							
t_{PHL}	An to Bn	Prop. Delay	CD = 0V, T/$\overline{R}$ = 3V (*Figures 1* and *2*)	2.5	5	7.5	ns
t_{PLH}				2.5	5	7.5	ns
t_{PHL}	CD to Bn	Enable Time	T/$\overline{R}$ = An = 3V (*Figures 1* and *3*)	3	7	12	ns
t_{PLH}		Disable Time		3	8	12	ns
t_{PHL}	T/$\overline{R}$ to Bn	Enable Time	CD = 0V, An = 3V (*Figures 8* and *9*)		12		ns
t_{PLH}		Disable Time	CD = 0V, An = 3V (*Figures 1* and *2*)	4	8.5	12	ns
t_r	Bn Rise Time 20% to 80%		CD = 0V, T/$\overline{R}$ = 3V		3		ns
t_f	Bn Fall Time 20% to 80%				3		ns
SR	Slew Rate is Calculated from 1.3V to 1.8V		(*Figures 1* and *4*)			0.5	V/ns
t_{skew}	An to Bn Skew in the Same Package		(Note 7)		1		ns
RECEIVER							
t_{PHL}	Bn to An		CD = T/$\overline{R}$ = 0V (*Figures 4* and *5*)	2.5	5	8.5	ns
t_{PLH}				2.5	5	8.5	ns
t_{skew}	Bn to An Skew in the Same Package		(Note 7)		1		ns
t_{PLZ}	CD to An	Disable Time	Bn = 2.1V, T/$\overline{R}$ = 0V (*Figures 6* and *7*)	4	8.5	11	ns
t_{PZL}		Enable Time		4	9	14	ns
t_{PHZ}		Disable Time	Bn = 2.1V, T/$\overline{R}$ = 0V (*Figures 6* and *7*)	4	8.5	11	ns
t_{PZH}		Enable Time		4	9	12	ns
t_{PLZ}	T/$\overline{R}$ to An	Disable Time	CD = 0V (*Figures 8* and *9*)	4	8.5	12	ns
t_{PZL}		Enable Time			15		ns
t_{PHZ}		Disable Time	Bn = 1.1V, CD = 0V (*Figures 6* and *7*)	4	8.5	12	ns
t_{PZH}		Enable Time		4	9	12	ns
PARAMETERS NOT TESTED (Guaranteed by Design)							
Cap	BTL Output Capacitance		(Note 8)			5	ns
t_{NR}	Noise Rejection		(Note 9)		1		ns

Note 6: Input waveforms shall have a rise/fall time of 3 ns.

Note 7: t_{SKEW} is an absolute value, defined as differences seen in propagation delays between drivers or receivers in the same package with identical load conditions.

Note 8: This parameter is tested during device characterization and is guaranteed by design. The parameter is tested using TDR techniques described in P1194 BTL backplane Design Guide.

Note 9: This parameter is tested during device characterization and is guaranteed by design. The measurement revealed that the part will reject 1 ns pulse widths.

Pin Description

Pin Name	Number of Pins	Input/ Output	Description
A0–A8	9	I/O	TTL TRI-STATE® Receiver Output and Driver Input
B0–B8	9	I/O	BTL Receiver Input and Driver Output
B0GND–B8GND	9	NA	Driver Output Ground Reduces Ground Bounce Due to High Current Switching of Driver Outputs. (Note 10)
CD	1	I	Chip Disable
GND	2	NA	Ground Reference for Switching Circuits. (Note 10)
LI	1	NA	V_{CC} Supply for Live Insertion. Boards that Require Live Insertion Should Connect LI to the Live Insertion Pin on the Connector. (Note 11)
NC	8	NA	No Connect
QGND	1	NA	Ground Referenece for Receiver Input Bandgap Reference and Non-Switching Circuits. (Note 10)
QV_{CC}	1	NA	V_{CC} Supply for Bandgap Reference and Non-Switching Circuits. (Note 11)
T/$\overline{R}$	1	I	Transmit/$\overline{\text{Receive}}$—Transmit (An to Bn), Receive (Bn to An)
V_{CC}	2	NA	V_{CC} Supply for Switching Circuits. (Note 11)

Note 10: The multiplicity of grounds reduces the effective inductance of bonding wires and leads, which then reduces the noise caused by transients on the ground path. The various ground pins can be tied together provided that the external ground has low inductance. (i.e., Ground plane with power pins and many signal pins connected to the backplane ground.) If the external ground floats considerably during transients, precautionary steps should be taken to prevent QGND from moving with reference to the backplane ground. The receiver threshold should have the same ground reference as the signal coming from the backplane. A voltage offset between their grounds will degrade the noise margin.

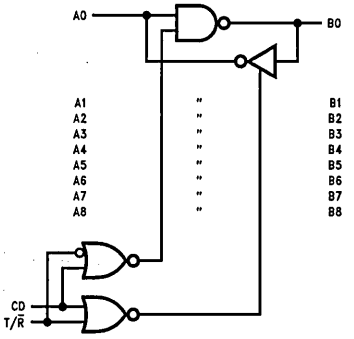
Note 11: The same considerations for ground was used for V_{CC} in reducing lead inductance (see Note 10). QV_{CC} and V_{CC} should be tied together externally. If live insertion is not supported, the LI pin can be tied together with QV_{CC} and V_{CC}.

Truth Table

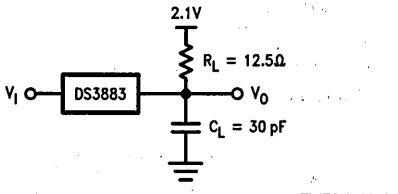
CD	T/$\overline{R}$	An	Bn
H	X	Z	H
L	L	L	H
L	L	H	L
L	H	H	L
L	H	L	H

X = High or low logic state
Z = High impedance state
L = Low state
H = High state

Logic Diagram

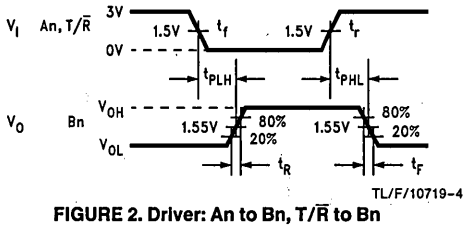


TL/F/10719–1



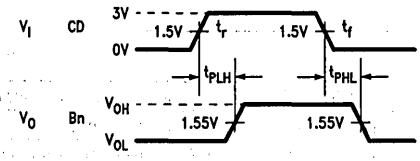
TL/F/10719–3

FIGURE 1. Driver Propagation Delay Set-Up



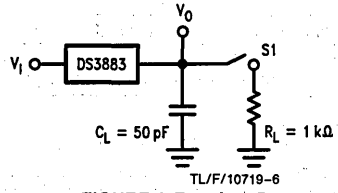
TL/F/10719–4

FIGURE 2. Driver: An to Bn, T/$\overline{R}$ to Bn



TL/F/10719–5

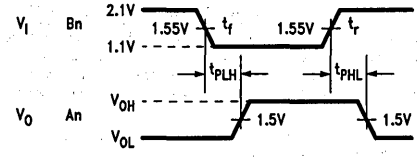
FIGURE 3. Driver: CD to Bn



Switch Position		
	t_{PLH}	t_{PHL}
S1	Open	Close

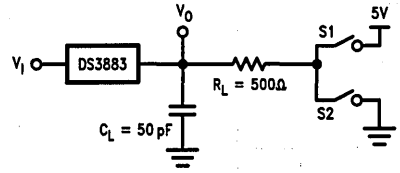
TL/F/10719–6

FIGURE 4. Receiver Propagation Delay Set-Up



TL/F/10719–7

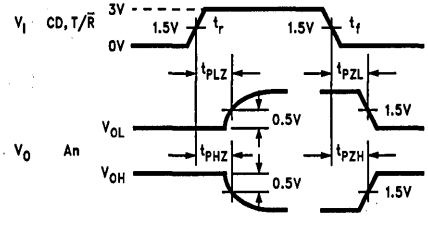
FIGURE 5. Receiver: Bn to An



Switch Position		
	t_{PZL} t_{PLZ}	t_{PZH} t_{PHZ}
S1	Close	Open
S2	Open	Close

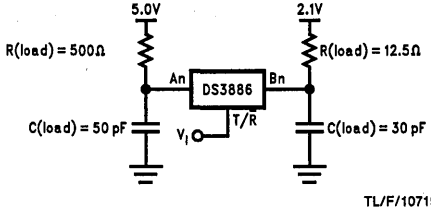
TL/F/10719–8

FIGURE 6. Receiver Enable/Disable Set-Up



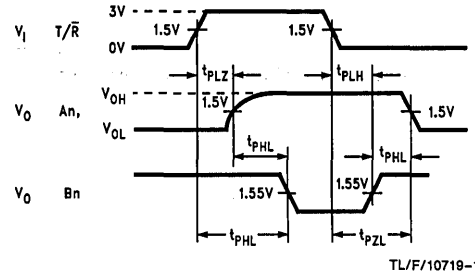
TL/F/10719–9

FIGURE 7. Receiver: CD to An, T/$\overline{R}$ to An



TL/F/10719–18

FIGURE 8. T/$\overline{R}$ to An, T/$\overline{R}$ to Bn



TL/F/10719–19

FIGURE 9. t_{PHL}(T/$\overline{R}$ to Bn), t_{PZL}(T/$\overline{R}$ to An)

National Semiconductor

PRELIMINARY

DS3884 BTL Handshake Transceiver

General Description

The DS3884 Handshake Transceiver is designed to conform to IEEE P1194.1 (BTL) as specified in IEEE P896.2 (Futurebus+). The DS3884 simplifies the implementation of all handshake signals which require wired-OR glitch filtering. The DS3884 is a six-bit wide device. Three of the six receivers have an additional parallel wired-OR glitch filter receiver as shown in the logic diagram, giving a total of 9 receiver outputs.

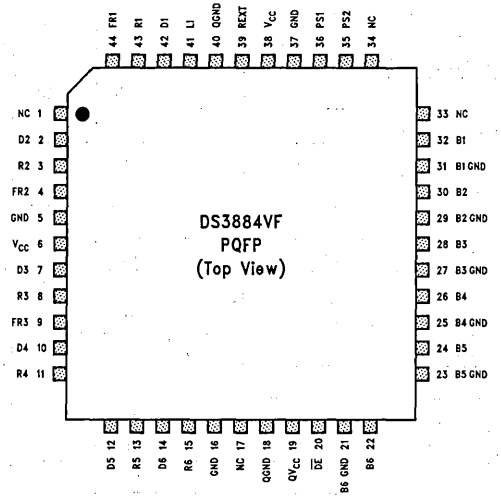
Two pulse selection pins on the DS3884 give four different settings for the glitch filters. Since the wired-OR glitch is a function of backplane length and loading, the wired-OR glitch filter can be optimized for a given backplane.

The DS3884 is one of five current devices in the Futurebus+ Chip Set. The DS3883 BTL 9-Bit Data Transceiver and DS3886 BTL Latching Transceiver are used for the address/data, command, tag, and status lines. The DS3885 Arbitration Transceiver has the competition logic for the AB<8:0> signal lines. The DS3875 Arbitration Controller supports all the required and optional modes for the Futurebus+ arbitration protocol. It is designed to be used in conjunction with the DS3884 and DS3885.

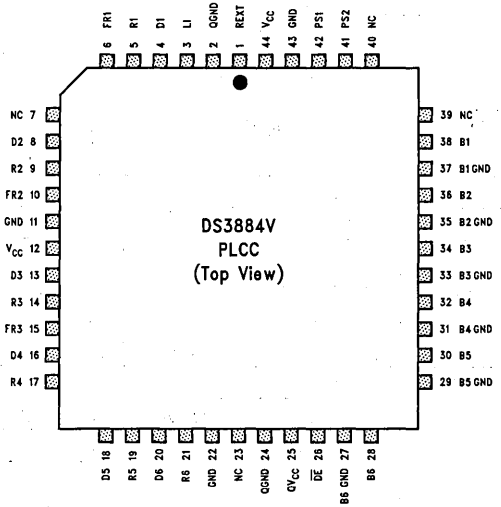
Features

- Meets 1194.1 standard for backplane transceiver logic
- Pinout is designed specifically for implementing a Futurebus+ system
- Supports live insertion
- Low skew
- Controlled rise/fall time
- Glitch free power-up/down protection
- Built-in bandgap reference provides very accurate thresholds
- 44-pin PLCC
- 44-pin PQFP saves board real-estate

Connection Diagrams



TL/F/10720–10



TL/F/10720–2

Order Number DS3884V or DS3884VF
See NS Package V44A or VF44B

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6.5V
Control Input Voltage	6.5V
Driver Input and Receiver Output	6.5V

Note: At no time including power-up shall the control input, driver input or receiver output exceed V_{CC} plus 1V. (See Note 2).

Receiver Input and Driver Output	3V
Power Dissipation at 70°C	1.22W
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 4 sec.)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
Bus Termination Voltage (V_T)	2.06	2.14	V
Operating Free Air Temperature	0	+70	°C

DC Electrical Characteristics (Notes 2, 3 and 4) T_A = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DRIVER AND CONTROL INPUT (Dn, $\overline{DE}$, PS1 and PS2)						
V_{IH}	Minimum Input High Voltage		2.0			V
V_{IL}	Maximum Input Low Voltage				0.8	V
I_{IH}	Input High Current	V_{IN} = 2.4V			40	µA
I_{IL}	Input Low Current	V_{IN} = 0.5V			−100	µA
V_{CL}	Input Diode Clamp Voltage	I_{CLAMP} = −12 mA			−1.2	V
DRIVER OUTPUT/RECEIVER INPUT (Bn)						
V_{OLB}	Output Low Bus Voltage (Note 5)	Dn = 2.4V, $\overline{DE}$ = 0V, I_{OL} = 80 mA	0.75	1.0	1.1	V
I_{OLBZ}	Output Low Bus Current	Dn = 0.5V, $\overline{DE}$ = 0V, Bn = 0.75V			−250	µA
I_{OHBZ}	Output High Bus Current	Dn = 0.5V, $\overline{DE}$ = 0V, Bn = 2.1V			100	µA
		Dn = 0.5V, $\overline{DE}$ = 0V, Bn = 2.1V, V_{CC} = 0V			100	µA
V_{TH}	Receiver Input Threshold	$\overline{DE}$ = 2.4V	1.47	1.55	1.62	V
V_{CLP}	Positive Clamp Voltage	V_{CC} = Max or 0V, I_{Bn} = 1 mA	2.4	3.4	4.3	V
		V_{CC} = Max or 0V, I_{Bn} = 10 mA	2.9	3.9	4.7	V
V_{CLN}	Negative Clamp Voltage	I_{CLAMP} = −12 mA			−1.2	V
RECEIVER OUTPUT (FRn and Rn)						
V_{OH}	Voltage Output High	Bn = 1.1V, $\overline{DE}$ = 2.4V, I_{OH} = −2 mA	2.4	3.2		V
V_{OL}	Voltage Output Low	Bn = 2.1V, $\overline{DE}$ = 2.4V, I_{OL} = 12 mA		0.35	0.5	V
		Bn = 2.1V, $\overline{DE}$ = 2.4V, I_{OL} = 8 mA		0.35	0.4	V
I_{OS}	Output Short Circuit Current	Bn = 1.1V, $\overline{DE}$ = 2.4V	−40	−70	−100	mA
SUPPLY CURRENT						
I_{CC}	Supply Current: Includes V_{CC},	$\overline{DE}$ = 0.5V, All Dn = 2.4V		80	100	mA
	QV_{CC} and LI	$\overline{DE}$ = 2.4V, All Bn = 2.1V		60	85	mA
I_{LI}	Live Insertion Current	$\overline{DE}$ = 2.4V		1.5	3	mA
		$\overline{DE}$ = 0.5V, All An = 2.4V		3	5	mA

Note 1: Absolute Maximum Ratings are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All input and/or output pins shall not exceed V_{CC} plus 1V and shall not exceed the absolute maximum rating at anytime, including power-up and power down. This prevents the ESD structure from being damaged due to excessive currents flowing from the input and/or output pins to QV_{CC} and V_{CC}. There is a diode between each input and/or output to V_{CC} which is forward biased when incorrect sequencing is applied. Alternatively, a current limiting resistor can be used when pulling-up the inputs to prevent damage. The current into any input/output pin shall be no greater than 50 mA. Exception, LI and Bn pins do not have power sequencing requirements with respect to V_{CC} and QV_{CC}. Furthermore, the difference between V_{CC} and QV_{CC} should never be greater than 1V at any time including power-up.

Note 3: All current into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 4: All typical values are specified under these conditions: V_{CC} = 5V and T_A = 25°C.

Note 5: Referenced to appropriate signal ground. Do not exceed maximum power dissipation of package.

AC Electrical Characteristics T_A = 0°C to +70°C, V_{CC} = 5V ±10% (Note 6)

Symbol	Parameter		Conditions	Min	Typ	Max	Units
DRIVER							
t_{PHL}	Dn to Bn Propagation Delay		$\overline{DE}$ = 0V (*Figures 1* and *2*)	2.5	5	7.5	ns
t_{PLH}				2.5	5	7.5	ns
t_{PHL}	$\overline{DE}$ to Bn	Enable Time	Dn = 3V (*Figures 1* and *3*)	4	6	12	ns
t_{PLH}		Disable Time		4	8	14	ns
t_T	Transition Time-Rise/Fall 20% to 80%		*(Figures 1 & 2)*		3		ns
SR	Slew Rate is Calculated from 1.3V to 1.8V					0.5	V/ns
t_{SKEW}	Skew between Drivers in the Same Package		(Note 7)		1		ns
RECEIVER							
t_{PHL}	Bn to Rn Propagation Delay		$\overline{DE}$ = 3V (*Figures 4* and *5*)	2.5	5	8	ns
t_{PLH}				2.5	5	8	ns
t_{SKEW}	Skew between Receivers in Same Package		(Note 7)		1		ns
FILTERED RECEIVER							
t_{PHL}	Bn to FRn Propagation Delay		PS1 = 0V, PS2 = 0V, $\overline{DE}$ = 3V (*Figures 4* and *5*)	8	12	15	ns
			PS1 = 0V, PS2 = 3V, $\overline{DE}$ = 3V (*Figures 4* and *5*)	13	16	22.5	ns
			PS1 = 3V, PS2 = 0V, $\overline{DE}$ = 3V (*Figures 4* and *5*)	18	22	30	ns
			PS1 = 3V, PS2 = 3V, $\overline{DE}$ = 3V (*Figures 4* and *5*)	28	35	45	ns
t_{PLH}	Bn to FRn Propagation Delay		$\overline{DE}$ = 3V (*Figures 4* and *5*) (Note 8)	2.5	5	9	ns
t_{GR}	Glitch Rejection		PS1 = 0V, PS2 = 0V, $\overline{DE}$ = 3V *(Figures 4 and 6)*	5			ns
			PS1 = 0V, PS2 = 3V, $\overline{DE}$ = 3V (*Figures 4* and *6*)	10			ns
			PS1 = 3V, PS2 = 0V, $\overline{DE}$ = 3V (*Figures 4* and *6*)	15			ns
			PS1 = 3V, PS2 = 0V, $\overline{DE}$ = 3V (*Figures 4* and *6*)	25			ns
FILTERED RECEIVER TIMING REQUIREMENTS							
t_S	PSn to Bn Set-Up Time		*(Figure 7)*	100			ns
PARAMETERS NOT TESTED (Guaranteed by Design)							
C_{output}	Capacitance at Bn		(Note 9)			5	pF
t_{NR}	Noise Rejection		(Note 10)		1		ns

Note 6: Input waveforms shall have a rise/fall time of 3 ns.

Note 7: t_{SKEW} is an absolute value, defined as differences seen in propagation delays between drivers or receivers in the same package with identical load conditions.

Note 8: t_{FRLH} is independent of filter setting.

Note 9: This parameter is tested during device characterization and is guaranteed by design. The parameter is tested using TDR techniques described in P1194 BTL Backplane Design Guide.

Note 10: This parameter is tested during device characterization and is guaranteed by design. The measurement revealed that the part will typically reject 1 ns pulse width.

Pin Description

Pin Name	Number of Pins	Input/ Output	Description
B1–B6	6	I/O	BTL receiver Input and Driver Output.
B1GND–B6GND	6	NA	Driver output ground reduces ground bounce due to high current switching of driver outputs. (Note 11)
$\overline{DE}$	1	I	Driver Enable Low
D1–D6	6	I	TTL driver input
FR1–FR3	3	O	TTL TRI-STATE® filtered receiver output
GND	3	NA	Ground reference for switching circuits. (Note 11)
LI	1	NA	V_{CC} supply for live insertion. Boards that require live insertion should connect LI to the live insertion pin on the connector. (Note 12)
NC	4	NA	No connect
PS1, PS2	2	I	Pulse Width Selection pin determines glitch filter setting. (Note 13)
R1–R6	6	O	TTL TRI-STATE receiver output
REXT	1	NA	External Resistor pin. External resistor is used for internal biasing of filter circuitry. The 6.0 kΩ resistor shall be connected between (REXT) = REXT and (GND) = GND. The resistor shall have a tolerance of 1% and a temperature coefficient of 100 ppm/°C or better.
QGND	2	NA	Ground reference for receiver input bandgap reference and non-switching circuits. (Note 11)
QV_{CC}	1	NA	V_{CC} supply for bandgap reference and non-switching circuits. (Note 12)
V_{CC}	2	NA	V_{CC} supply for switching circuits. (Note 13)

Note 11: The multiplicity of grounds reduces the effective inductance of bonding wires and leads, which then reduces the noise caused by transients on the ground path. The various ground pins can be tied together provided that the external ground has low inductance (i.e., ground plane with power pins and many signal pins connect to the backplane ground). If the external ground floats considerably during transients, precautionary steps should be taken to prevent QGND from moving with reference to the backplane ground. The receiver threshold should have the same ground reference as the signal coming from the backplane. A voltage offset between their grounds will degrade the noise margin.

Note 12: The same considerations for ground was used for V_{CC} in reducing lead inductance (see Note 11:). QV_{CC} and V_{CC} should be tied together externally. If live insertion is not supported, the LI pin can be tied together with QV_{CC} and V_{CC}.

Note 13: See AC Characteristics for filter setting.

Truth Table

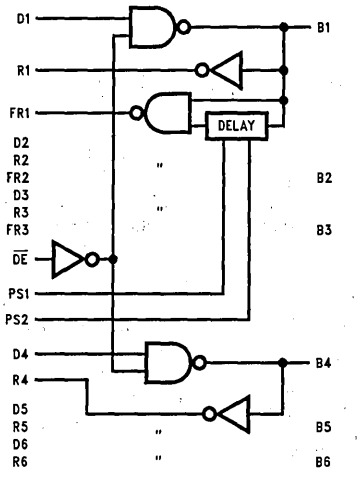
$\overline{DE}$	Dn	FRn	Rn	Bn
H	X	H	H	L
H	X	L	L	H
L	H	H	H	L
L	L	L	L	H

X: High or Low Logic State
Z: High Impedance State
L: Low State
H: High State
L–H: Low to High Transition

Glitch Filter Table

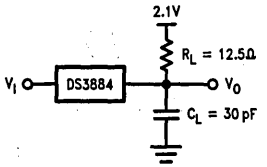
PS1	PS2	Filter Setting
L	L	≤5 ns
L	H	≤10 ns
H	L	≤15 ns
H	H	≤25 ns

Logic Diagram



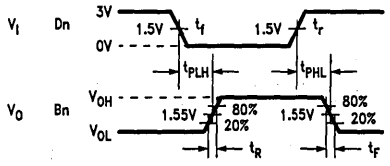
TL/F/10720–1

Timing Waveforms



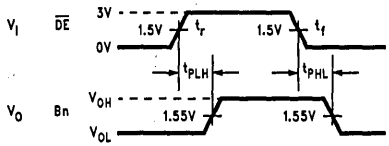
TL/F/10720–3

FIGURE 1. Driver Propagation Delay Set-Up



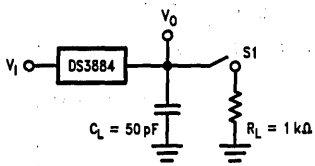
TL/F/10720–4

FIGURE 2. Driver: Dn to Bn



TL/F/10720–5

FIGURE 3. Driver: $\overline{DE}$ to Bn

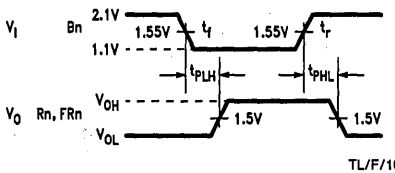


Switch Position

	t_{PLH}	t_{PHL}
S1	Open	Close

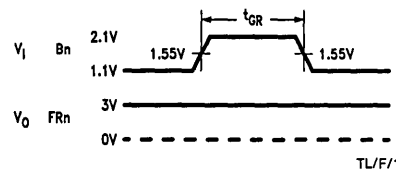
TL/F/10720–6

FIGURE 4. Receiver Propagation Delay Set-Up



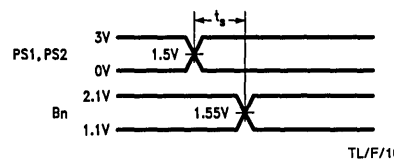
TL/F/10720–7

FIGURE 5. Receiver: Bn to FRn, Bn to Rn



TL/F/10720–8

FIGURE 6. Receiver: T_{GR}



TL/F/10720–9

FIGURE 7. Receiver: PSn to Bn

National Semiconductor

PRELIMINARY

DS3885 BTL Arbitration Transceiver

General Description

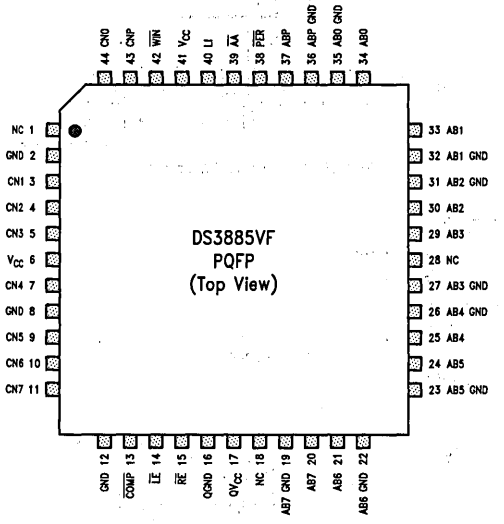
The DS3885 Arbitration Transceiver conforms to IEEE P896 Futurebus+ specifications. The Arbitration transceiver incorporates the competition logic internally which simplifies on board logic and reduces competition delays.

The DS3885 is one of five current devices in the Futurebus+ Chip Set. The DS3883 BTL 9-Bit Data Transceiver and DS3886 BTL Latching Transceiver are used for the address/data, command, tag, and status lines. The DS3884 Handshake Transceiver has a pulse width selectable wired-OR glitch filter and is used on the handshake lines. The DS3875 Arbitration Controller supports all the required and optional modes in the Futurebus+ arbitration protocol. It is designed to be used in conjunction with the DS3884 and DS3885.

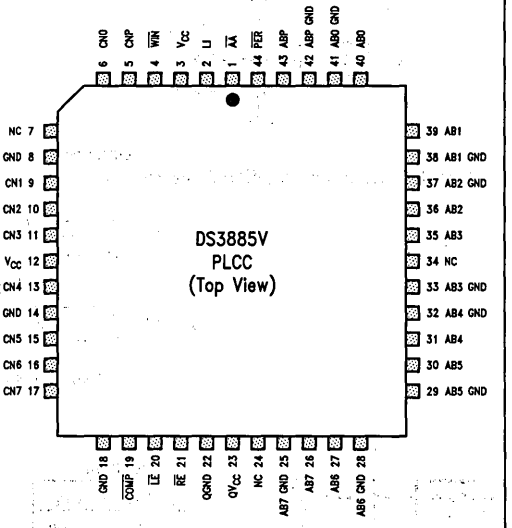
Features

- Meets P1194.1 standard for Backplane transceiver logic
- Pinout is designed specifically for implementing a Futurebus+ system
- Supports Live Insertion
- Low Skew
- Controlled rise/fall time
- Glitch free power up/down protection
- Built-in bandgap reference provides very accurate thresholds
- 44-pin PLCC
- 44-pin PQFP saves board real-estate

Connection Diagrams



TL/F/10721–2



TL/F/10721–13

Order Number DS3885V or DS3885VF
See NS Package Number V44A or VF44B

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6.5V
Control Input Voltage	6.5V
Driver Input and Receiver Output	6.5V

Note: At no time including power-up shall the control input, driver input or receiver output exceed V_{CC} + 1V. (See Note 2)

Receiver Input and Driver Output	3V
Power Dissipation at 70°C	1.22W
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 4 sec.)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage, V_{CC}	4.5	5.5	V
Bus Termination Voltage (V_T)	2.06	2.14	V
Operating Free Air Temperature	0	70	C

DC Electrical Characteristics (Notes 2, 3 and 4) T_A = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DRIVER AND CONTROL INPUT: (CNn CNP, $\overline{LE}$, $\overline{COMP}$, and $\overline{RE}$)						
V_{IH}	Minimum Input High Voltage		2.0			V
V_{IL}	Maximum Input Low Voltage				0.8	V
I_{IH}	Input High Current	V_{IN} = 2.4V			40	µA
I_{IL}	Input Low Current	V_{IN} = 0.5V			−100	µA
V_{CL}	Input Diode Clamp Voltage	I_{CLAMP} = −12 mA			−1.2	V
DRIVER OUTPUT/RECEIVER INPUT: (ABn and ABP)						
V_{OLB}	Output Low Bus Voltage (Note 5)	CNn = $\overline{RE}$ = 2.4V, $\overline{LE}$ = $\overline{COMP}$ = 0.5V I_{OL} = 80 mA	0.75	1.0	1.1	V
I_{OLBZ}	Output Low Bus Current	$\overline{COMP}$ = $\overline{RE}$ = 2.4V, ABn = 0.75V			−250	µA
I_{OHBZ}	Output High Bus Current	$\overline{COMP}$ = $\overline{RE}$ = 2.4V, ABn = 2.1V			100	µA
		$\overline{COMP}$ = $\overline{RE}$ = 2.4V, ABn = 2.1V, V_{CC} = 0V			100	µA
V_{TH}	Receiver Input Threshold	$\overline{COMP}$ = $\overline{LE}$ = 2.4V	1.47	1.55	1.62	V
V_{CLP}	Positive Clamp Voltage	V_{CC} = Max or 0V, ABn = 1 mA	2.4	3.4	4.5	V
		V_{CC} = Max or 0V, ABn = 10 mA	2.9	3.9	5.0	V
V_{CLN}	Negative Clamp Voltage	I_{CLAMP} = −12 mA			−1.2	V
RECEIVER OUTPUT: (CNn, CNP, $\overline{AA}$, $\overline{PER}$, and $\overline{WIN}$)						
V_{OH}	Voltage Output High	ABn = 1.1V, $\overline{RE}$ = 0.5V, $\overline{COMP}$ = $\overline{LE}$ = 2.4V, I_{OH} = −2 mA	2.4	3.2		V
V_{OL}	Voltage Output Low	ABn = 2.1V, $\overline{RE}$ = 0.5V, $\overline{COMP}$ = $\overline{LE}$ = 2.4V, I_{OL} = 12 mA		0.35	0.5	V
		ABn = 2.1V, $\overline{RE}$ = 0.5V, $\overline{COMP}$ = $\overline{LE}$ = 2.4V, I_{OL} = 8 mA		0.35	0.4	V
I_{OS}	Output Short Circuit Current	ABn = 1.1V, $\overline{RE}$ = 0.5V $\overline{COMP}$ = $\overline{LE}$ = 2.4V	−40	−70	−100	mA
SUPPLY CURRENT						
I_{CC}	Supply Current: includes V_{CC}, QVCC and LI	$\overline{COMP}$ = $\overline{LE}$ = 0.5V, All CNn = $\overline{RE}$ = 2.4V		75	100	mA
		$\overline{COMP}$ = $\overline{LE}$ = $\overline{RE}$ = 2.4V		26	40	mA
I_{LI}	Live Insertion Current	$\overline{COMP}$ = $\overline{RE}$ = 2.4V, $\overline{LE}$ = 0.5V		1.5	3	mA
		$\overline{COMP}$ = $\overline{LE}$ = 0.5V, All CNn = $\overline{RE}$ = 2.4		3	5	mA

Note 1: Absolute Maximum Ratings are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provide conditions for actual device operation.

Note 2: All input and/or output pins shall not exceed V_{CC} + 1V and shall not exceed the absolute maximum rating at anytime, including power-up and power down. This prevents the ESD structure from being damaged due to excessive currents flowing from the input and/or output pins to QV_{CC} and V_{CC}. There is a diode between each input and/or output to V_{CC} which is forward biased when incorrect sequencing is applied. Alternatively, a current limiting resistor can be used when pulling-up the inputs to prevent damage. The current into any input/output pin shall be no greater than 50 mA. Exception, LI and Bn pins do not have power sequencing requirements with respect to V_{CC} and QV_{CC}. Futhermore, the difference between V_{CC} and QV_{CC} should never be greater than 1V at any time including power-up.

Note 3: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 4: All typicals are specified under these conditions: V_{CC} = 5V and T_A = 25°C.

Note 5: Referenced to appropriate signal ground. Do not exceed maximum power dissipation of package.

AC Electrical Characteristics T_A = 0°C to +70°C, V_{CC} = 5V ±10% (Note 6)

Symbol	Parameter		Conditions	Min	Typ	Max	Units
DRIVER (*Figures 1* and *2*)							
t_{PHL}	$\overline{LE}$ to AB7	Propagation Delay	$\overline{COMP}$ = 0V, $\overline{RE}$ = 3V	8	12	16	ns
t_{PLH}				7	12	16	ns
t_R	Rise Time	20% to 80%	$\overline{RE}$ = 3V, $\overline{COMP}$ = $\overline{LE}$ = 0		2		ns
t_F	Fall Time	20% to 80%	$\overline{RE}$ = 3V, $\overline{COMP}$ = $\overline{LE}$ = 0		2		ns
DRIVER TIMING REQUIREMENTS (*Figures 1* and *2*)							
t_s	CNn to $\overline{LE}$	Set-up Time	$\overline{RE}$ = 3V, $\overline{COMP}$ = 0V	9			ns
t_h	$\overline{LE}$ to CNn	Hold Time	$\overline{RE}$ = 3V, $\overline{COMP}$ = 0V	0			ns
t_{pw}	$\overline{LE}$ Pulse Width		$\overline{RE}$ = 3V, $\overline{COMP}$ = 0V	15			ns
RECEIVER							
t_{PHL}	ABn to CNn	Prop. Delay	$\overline{RE}$ = 0V, $\overline{COMP}$ = $\overline{LE}$ = 3V (*Figures 4* and *5*)	6	11	18	ns
t_{PLH}				6	11	18	ns
t_{PLZ}	$\overline{RE}$ to CNn	Disable Time	$\overline{COMP}$ = $\overline{LE}$ = 3V, ABn = 2.1V (*Figures 6* and *7*)	4	7	10	ns
t_{PZL}		Enable Time		5	8	11	ns
t_{PHZ}		Disable Time	$\overline{COMP}$ = $\overline{LE}$ = 3V, ABn = 1.1V (*Figures 6* and *7*)	4	7	11	ns
t_{PZH}		Enable Time		3	6	10	ns
OTHERS							
t_{PHL}	AB0 to $\overline{AA}$ All Asserted Condition	Prop. Delay	AB <7:1> = 1.1V (*Figures 4* and *8*)	7	13	23	ns
t_{PLH}				7	13	23	ns
t_{PHL}	AB0 to $\overline{WIN}$ Win Condition	Prop. Delay	$\overline{COMP}$ = $\overline{LE}$ = 0V, $\overline{RE}$ = 3V, CN <7:0> = 0V (*Figures 4* and *9*) AB<7:0> = 2.1V	6	11	23	ns
t_{PLH}				6	11	23	ns
t_{PHL}	AB0 to $\overline{WIN}$ Greater Than Condition	Prop. Delay	$\overline{COMP}$ = $\overline{LE}$ = 3V, $\overline{RE}$ = 0V, CN <7:1> = 0V, CN0 = 3V (*Figures 4* and *9*) AB<7:0> = 2.1V	6	14	23	ns
t_{PLH}				6	14	23	ns
t_{PHL}	ABP to $\overline{PER}$ Parity Error Condition	Prop. Delay	$\overline{COMP}$ = $\overline{LE}$ = $\overline{RE}$ = 3V, AB <7:1> = 1.1V, AB0 = 2.1V (*Figures 4* and *8*)	6	12	23	ns
t_{PLH}				6	12	23	ns
t_{PHL}	ABn to AB <n−1>	Prop. Delay	$\overline{COMP}$ = $\overline{LE}$ = 0V, $\overline{RE}$ = 3V, CNn = 0V, CN <n−1> = 3V, CN <7:n+1> = 0V (*Figures 1* and *10*)	5	11	20	ns
t_{PLH}				5	13	20	ns
t_{PHL}	$\overline{COMP}$ to AB7	Prop. Delay	$\overline{LE}$ = 0V, $\overline{RE}$ = CN7 = 3V (*Figures 1* and *3*)	5	8	12	ns
t_{PLH}				6	9	15	ns
t_{PHL}	AB7 to ABP	Prop. Delay	$\overline{COMP}$ = $\overline{LE}$ = 0V, $\overline{RE}$ = CNP = 3V, CN <7:0> = 0V (*Figures 1* and *10*)		30	49	ns
t_{PLH}					30	49	ns
PARAMETERS NOT TESTED (Guaranteed by Design)							
C_{output}	Capacitance at Bn		(Note 7)			5	pF
t_{NR}	Noise Rejection		(Note 8)		1		ns

Note 6: All input rise/fall times should be 3 ns.

Note 7: This parameter is tested during device characterization and is guaranteed by design. The parameter is tested using TDR techniques described in P1194.0 - BTL Backplane Design Guide.

Note 8: This parameter is tested during device characterization and is guaranteed by design. The measurement revealed that the part will typically reject 1 ns pulse width.

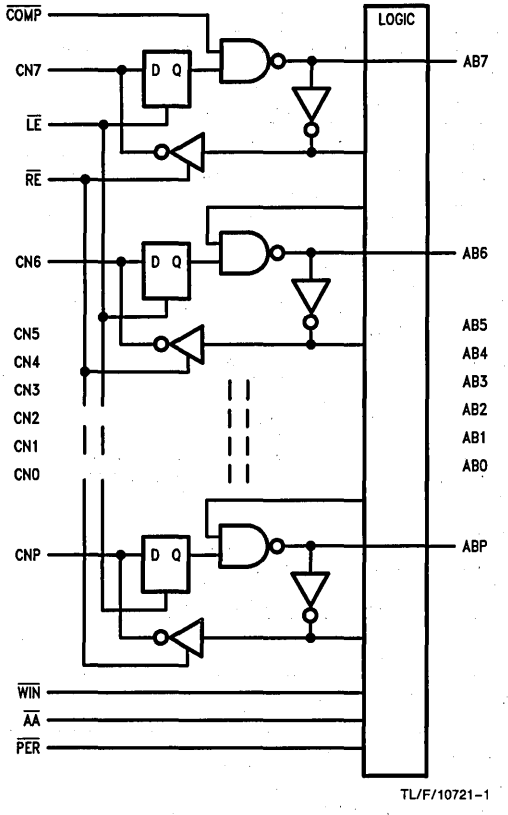
Pin Description

Pin Name	Number of Pins	Input/ Output	Description
$\overline{AA}$	1	O	TTL—All asserted (A logic "0" indicates that all the competition bits are asserted.)
AB<7:0>	8	I/O	BTL—Futurebus+ wired-OR competition bits
ABP	1	I/O	BTL—Futurebus+ wired-OR competition parity bit
AB<7:0> and ABP GND	9	NA	Driver output ground reduces ground bounce due to high current switching of driver outputs (Note 9)
CN<7:0>	8	I/O	TTL—Module competition bits
CNP	1	I/O	TTL—Module competition parity bit
$\overline{COMP}$	1	I	TTL—Competition bit (A logic "0" indicates that the module will compete in the arbitration.)
GND	3	NA	Ground reference for switching circuits. (Note 9)
$\overline{LE}$	1	I	TTL—CNn latch enable (A logic "0" indicates that the CNn logic states are latched with corresponding parity bit.
LI	1	NA	V_{CC} supply for live insertion. Boards that require live insertion should connect LI to the live insertion pin on the connector. (Note 10)
NC	3	NA	No Connect
$\overline{PER}$	1	O	TTL—ABn odd parity (A logic "0" indicates parity error)
$\overline{RE}$	1	I	TTL—Receiver Enable (A logic "0" enables receivers)
QGND	1	NA	Ground reference for receiver input bandgap reference and non-switching circuits. (Note 9)
QV_{CC}	1	NA	V_{CC} supply for bandgap reference and non-switching circuits. (Note 10)
V_{CC}	2	NA	V_{CC} supply for switching circuits. (Note 10)
$\overline{WIN}$	1	O	TTL—Win signal (active low). During competition, $\overline{WIN}$ indicates that the module has won the competition. For a module not participating in the competition, $\overline{WIN}$ indicates that the module has a number which is greater than winner's number.

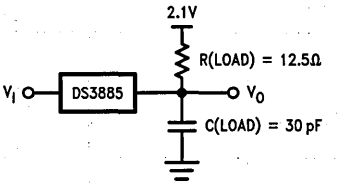
Note 9: The multiplicity of grounds reduces the effective inductance of bonding wires and leads, which then reduces the noise caused by transients on the ground path. The various ground pins can be tied together provided that the external ground has low inductance. (i.e., Ground plane with power pins and many signal pins connected to the backplane ground.) If the external ground floats considerably during transients, precautionary steps should be taken to prevent QGND from moving with reference to the backplane ground. The receiver threshold should have the same ground reference as the signal coming from the backplane. A voltage offset between their grounds will degrade the noise margin.

Note 10: The same considerations for ground was used for V_{CC} in reducing lead inductance. (See Note 9) QV_{CC} and V_{CC} should be tied together externally. If live insertion is not supported, the LI pin can be tied together with QV_{CC} and V_{CC}.

Logic Diagrams

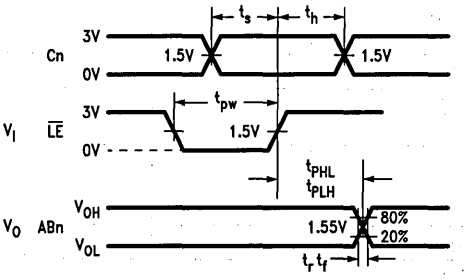


TL/F/10721–1



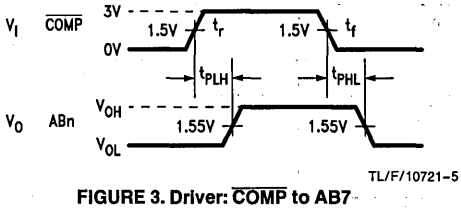
TL/F/10721–3

FIGURE 1. Driver Propagation Delay Set-up



TL/F/10721–4

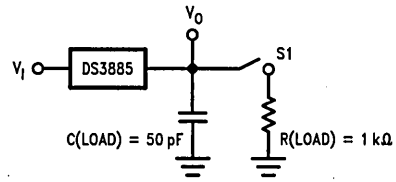
FIGURE 2. Driver: $\overline{LE}$ to AB7, t_S, t_h, t_{pw}



TL/F/10721–5

FIGURE 3. Driver: $\overline{COMP}$ to AB7

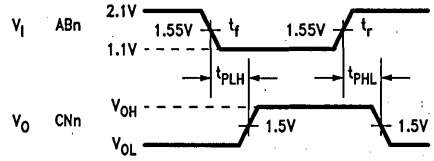
Logic Diagrams (Continued)



Switch Position	t_{PLH}	t_{PHL}
S1	open	close

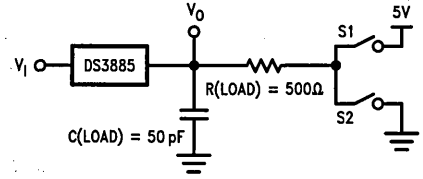
TL/F/10721–6

FIGURE 4. Receiver Propagation Delay Set-up



TL/F/10721–7

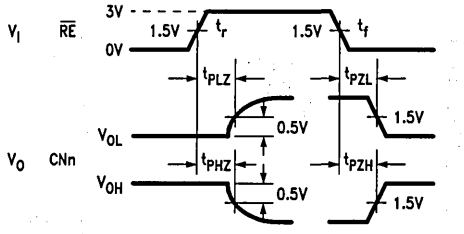
FIGURE 5. Receiver: ABn to CNn



Switch Position	t_{PZL} t_{PLZ}	t_{PZH} t_{PHZ}
S1	close	open
S2	open	close

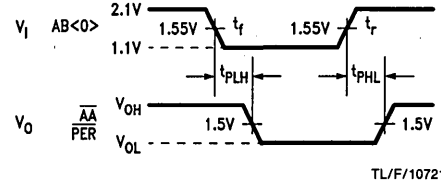
TL/F/10721–8

FIGURE 6. Receiver Enable/Disable Set-up



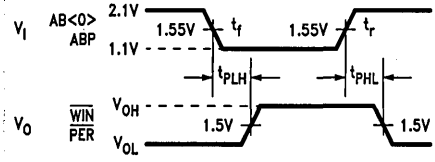
TL/F/10721–9

FIGURE 7. Receiver: $\overline{RE}$ to CNn



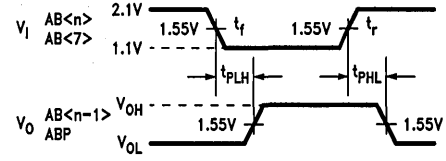
TL/F/10721–10

FIGURE 8. AB0 to $\overline{AA}$



TL/F/10721–11

FIGURE 9. AB0 to $\overline{WIN}$, ABP TO $\overline{PER}$



TL/F/10721–12

FIGURE 10. ABn to AB<n-1>, AB7 to ABP

National Semiconductor

PRELIMINARY

DS3886 BTL 9-Bit Latching Data Transceiver

General Description

The DS3886 BTL 9-Bit Latching Transceiver is designed to conform to IEEE P1194.1 (BTL) as specified in IEEE P896.2 (Futurebus+). The DS3886 simplifies the implementation of all byte wide address/data with parity signals. It can also be used for the status, tag and command lines.

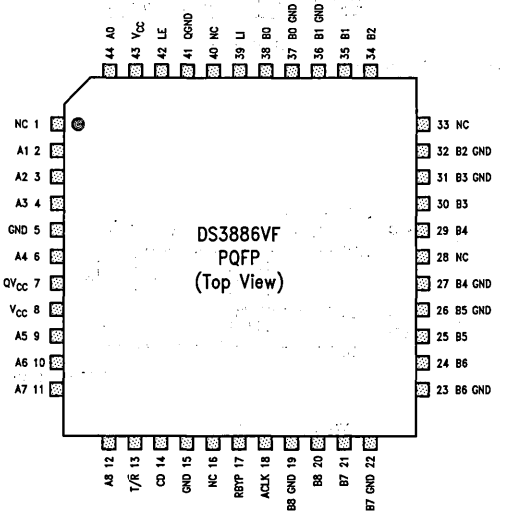
The Driver has an edge-triggered latch which can be bypassed during fall-through mode. The receiver has a transparent latch.

The DS3886 is one of five current devices in the Futurebus+ Chip Set. The DS3883 BTL 9-Bit Data Transceiver is an option to the DS3886 without the latches. The DS3884 Handshake Transceiver has pulse width selectable wired-OR glitch filters. The DS3885 Arbitration Transceiver has the competition logic for the AB<8:0> signal lines. The DS3875 Arbitration Controller supports all the required and optional modes for the Futurebus+ arbitration protocol. It is designed to be used in conjunction with the DS3884 and DS3885.

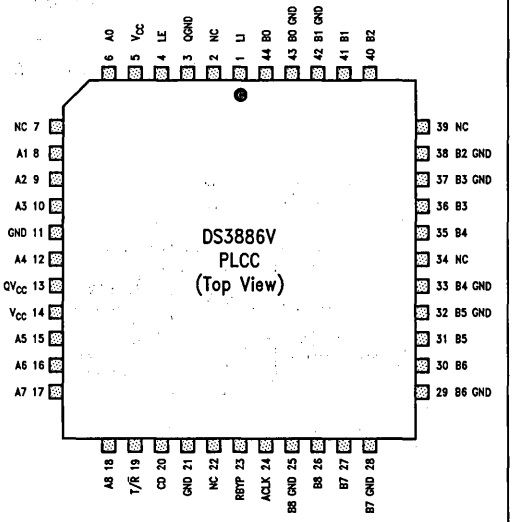
Features

- Meets P1194.1 standard for Backplane transceiver logic
- Pinout is designed specifically for implementing a Futurebus+ system
- Supports live insertion
- Low skew
- Controlled rise/fall time
- Glitch free power-up/down protection
- Built-in bandgap reference provides very accurate thresholds
- 44-pin PLCC
- 44-pin PQFP reduces board real-estate
- On chip latches

Connection Diagrams



TL/F/10722–12



TL/F/10722–13

Order Number DS3886V or DS3886VF
See NS Package V44A or VF44B

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6.5V
Control Input Voltage	6.5V
Driver Input and Receiver Output	6.5V

Note: At no time including power-up shall the control input, driver input or receiver output exceed V_{CC} + 1V. (See note 2.)

Receiver Input and Driver Output	3V
Power Dissipation at 70°C	1.22W
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 4 seconds)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage, V_{CC}	4.5	5.5	V
Bus Termination Voltage (V_T)	2.06	2.14	V
Operating Free Air Temperature	0	70	°C

DC Electrical Characteristics (Notes 2, 3 and 4) T_A = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DRIVER AND CONTROL INPUT (CD, T/$\overline{R}$, An, ACLK, LE, and RBYP)						
V_{IH}	Minimum Input High Voltage		2.0			V
V_{IL}	Maximum Input Low Voltage				0.8	V
I_{IH}	Input High Current	V_{IN} = 2.4V			40	µA
I_{IL}	An, CD, T/$\overline{R}$ Inputs	V_{IN} = 0.5V			−100	µA
V_{CL}	Input Diode Clamp Voltage	I_{clamp} = −12 mA			−1.2	V
DRIVER OUTPUT/$\overline{R}$ECEIVER INPUT (Bn)						
V_{OLB}	Output Low Bus Voltage (Note 5)	An = T/$\overline{R}$ = 2.4V, I_{OL} = 80 mA	0.75	1.0	1.1	V
I_{OLBZ}	Output Low Bus Current	An = 0.5V, T/$\overline{R}$ = 2.4V, Bn = 0.75V		−5	−250	µA
I_{OHBZ}	Output High Bus Current	An = 0.5V, T/$\overline{R}$ = 2.4V, Bn = 2.1V		5	100	µA
		An = 0.5V, T/$\overline{R}$ = 2.4V, Bn = 2.1V, V_{CC} = 0V		5	100	µA
V_{TH}	Receiver Input Threshold	T/$\overline{R}$ = CD = 0.5V	1.47	1.55	1.62	V
V_{CLP}	Positive Clamp Voltage	V_{CC} = Max or 0V, Bn = 1 mA	2.4	3.4	4.5	V
		V_{CC} = Max or 0V, Bn = 10 mA	2.9	3.9	5.0	V
V_{CLN}	Negative Clamp Voltage	I_{CLAMP} = −12 mA			−1.2	V
RECEIVER OUTPUT (An)						
V_{OH}	Voltage Output High	Bn = 1.1V, I_{OH} = −2 mA, T/$\overline{R}$ = CD = 0.5V	2.4	3.2		V
V_{OL}	Voltage Output Low	T/$\overline{R}$ = CD = 0.5V, Bn = 2.1V, I_{OL} = 12 mA		0.35	0.5	V
		T/$\overline{R}$ = CD = 0.5V, Bn = 2.1V, I_{OL} = 8 mA		0.35	0.4	V
I_{OS}	Output Short Circuit Current	Bn = 1.1V, T/$\overline{R}$ = CD = 0.5V	−40	−70	−100	mA
SUPPLY CURRENT						
I_{CC}	Supply Current: Includes V_{CC}, QV_{CC} and LI	T/$\overline{R}$ = All An = 2.4V, CD = 0.5V		100	120	mA
		T/$\overline{R}$ = 0.5V		65	90	mA
I_{LI}	Live Insertion Current	T/$\overline{R}$ = 0.5V		1.5	3	mA
		T/$\overline{R}$ = All An = 2.4V		3	5	mA

Note 1: Absolute Maximum Ratings are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All input and/or output pins shall not exceed V_{CC} + 1 and shall not exceed the absolute maximum rating at anytime, including power-up and power down. This prevents the ESD structure from being damaged due to excessive currents flowing from the input and/or output pins to QV_{CC} and V_{CC}. There is a diode between each input and/or output to V_{CC} which is forward biased when incorrect sequencing is applied. Alternatively, a current limiting resistor can be used when pulling-up the inputs to prevent damage. The current into any input/output pin shall be no greater than 50 mA. Exception, LI and Bn pins do not have power sequencing requirements with respect to V_{CC} and QV_{CC}. Furthermore, the difference between V_{CC} and QV_{CC} should never be greater than 1V at any time including power-up.

Note 3: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 4: All typical values are specified under these conditions: V_{CC} = 5V and T_A = 25°C, unless otherwise stated.

Note 5: Referenced to appropriate signal ground. Do not exceed maximum power dissipation of package.

AC Electrical Characteristics T_A = 0°C to +70°C, V_{CC} = 5V ±10% (Note 6)

Symbol	Parameter		Conditions	Min	Typ	Max	Units
DRIVER							
t_{PHL}	An to Bn Propagation Delay Fall-Through Mode		CD = 0V, T/$\overline{R}$ = RBYP = 3V (*Figures 1* and *2*)	3	6	11	ns
t_{PLH}				3	6	11	ns
t_{PHL}	ACLK to Bn Propagation Delay Latch Mode		CD = RBYP = 0V, T/$\overline{R}$ = 3V (*Figures 1* and *4*)	3	6.5	9	ns
t_{PLH}				3	6.5	9	ns
t_{PHL}	CD to Bn	Enable Time	T/$\overline{R}$ = 3V, An = 3V	3.5	5	10	ns
t_{PLH}		Disable Time	(*Figures 1* and *3*)	3	5	10	ns
t_{PHL}	T/$\overline{R}$ to Bn	Enable Time	CD = 0V, An = 3V (*Figures 10* and *11*)		15		ns
t_{PLH}		Disable Time	(*Figures 1* and *2*)	4	6	12	ns
t_r	Bn Rise Time (20% to 80%)		CD = RBYP = 0V, T/$\overline{R}$ = 3V (*Figures 1* and *4*)		3		ns
t_f	Bn Fall Time (20% to 80%)				3		ns
SR	Bn Slew Rate is Calculated from 1.3V to 1.8V		CD = RBYP = 0V, T/$\overline{R}$ = 3V (*Figures 1* and *4*)			0.5	V/ns
t_{skew}	ACLK to Bn Same Package		(Note 7)		1		ns
	An to Bn Same Package		(Note 7)		1		ns
DRIVER TIMING REQUIREMENTS *(Figure 4)*							
t_S	An to ACLK Set-Up Time		CD = RBYP = 0V, T/$\overline{R}$ = 3V	5			ns
t_H	ACLK to An Hold Time		CD = RBYP = 0V, T/$\overline{R}$ = 3V	0			ns
t_{pw}	ACLK Pulse Width		CD = RBYP = 0V, T/$\overline{R}$ = 3V	5			ns
RECEIVER							
t_{PHL}	Bn to An Propagation Delay Bypass Mode		CD = T/$\overline{R}$ = 0V, LE = 3V (*Figures 5* and *6*)	3	6	9	ns
t_{PLH}				3	6	9	ns
t_{PHL}	LE to An Propagation Delay Latch Mode		CD = T/$\overline{R}$ = 0V (*Figures 5* and *7*)	3	6	9	ns
t_{PLH}				3	6	9	ns
t_{PLZ}	CD to An	Disable Time	Bn = 2.1V, T/$\overline{R}$ = 0V (*Figures 8* and *9*)	4	8.5	10	ns
t_{PZL}		Enable Time		4	9	12	ns
t_{PHZ}		Disable Time	Bn = 1.1V, T/$\overline{R}$ = 0V (*Figures 8* and *9*)	4	8.5	10	ns
t_{PZH}		Enable Time		4	9	12	ns
t_{PLZ}	T/$\overline{R}$ to An	Disable Time	CD = 0V (*Figures 10* and *11*)	4	8.5	10	ns
t_{PZL}		Enable Time		4	9	12	ns
t_{PHZ}		Disable Time	Bn = 1.1V, CD = 0V (*Figures 8* and *9*)	4	8.5	10	ns
t_{PZH}		Enable Time		4	9	12	ns
t_{skew}	LE to An Same Package		(Note 7)		1		ns
	Bn to An Same Package		(Note 7)		1		ns
RECEIVER TIMING REQUIREMENTS *(Figure 7)*							
t_S	Bn to LE Set-up Time		CD = T/$\overline{R}$ = 0V	2			ns
t_H	LE to Bn Hold Time		CD = T/$\overline{R}$ = 0V	3			ns
t_{pw}	LE Pulse Width		CD = T/$\overline{R}$ = 0V	4			ns
PARAMETERS NOT TESTED (GUARANTEED BY DESIGN)							
C_{output}	Capacitance at Bn		(Note 8)			5	pF
t_{NR}	Noise Rejection		(Note 9)		1		ns

Note 6: Input waveforms shall have a rise/fall time of 3 ns.

Note 7: t_{skew} is an absolute value, defined as differences seen in propagation delays between drivers or receivers in the same package with identical load conditions.

Note 8: This parameter is tested during device characterization and is guaranteed by design. The parameter is tested using TDR techniques described in P1194.0 BTL Backplane Design Guide.

Note 9: This parameter is tested during device characterization and is guaranteed by design. The measurement revealed that the part will typically reject 1 ns pulse width.

Pin Description

Pin Name	Number of Pins	Input/ Output	Description
A0–A8	9	I/O	TTL TRI-STATE® receiver output and driver input
ACLK	1	I	Clock input for latch
B0–B8	9	I/O	BTL receiver input and driver output
B0GND–B8GND	9	NA	Driver output ground reduces ground bounce due to high current switching of driver outputs. (Note 10)
CD	1	I	Chip Disable
GND	2	NA	Ground reference for switching circuits. (Note 10)
LE	1	I	Latch Enable
LI	1	NA	V_{CC} supply for live insertion. Boards that require live insertion should connect LI to the live insertion pin on the connector. (Note 11)
NC	5	NA	No Connect
QGND	1	NA	Ground reference for receiver input bandgap reference and non-switching circuits. (Note 10)
QV_{CC}	1	NA	V_{CC} supply for bandgap reference and non-switching circuits. (Note 11)
RBYP	1	I	Register bypass enable
T/$\overline{R}$	1	I	Transmit/$\overline{Receive}$—Transmit (An to Bn, A_{CLK} to Bn) Receive (Bn to An, LE to An)
V_{CC}	2	NA	V_{CC} supply for switching circuits. (Note 11)

Note 10. The Multiplicity of grounds reduces the effective inductance of bonding wires and leads, which then reduces the noise caused by transients on the ground path. The various ground pins can be tied together provided that the external ground has low inductance (i.e., ground plane with power pins and many signal pins connected to the backplane ground). If the external ground floats considerably during transients, precautionary steps should be taken to prevent QGND from moving with reference to the backplane ground. The receiver threshold should have the same ground reference as the signal coming from the backplane. A voltage offset between their grounds will degrade the noise margin.

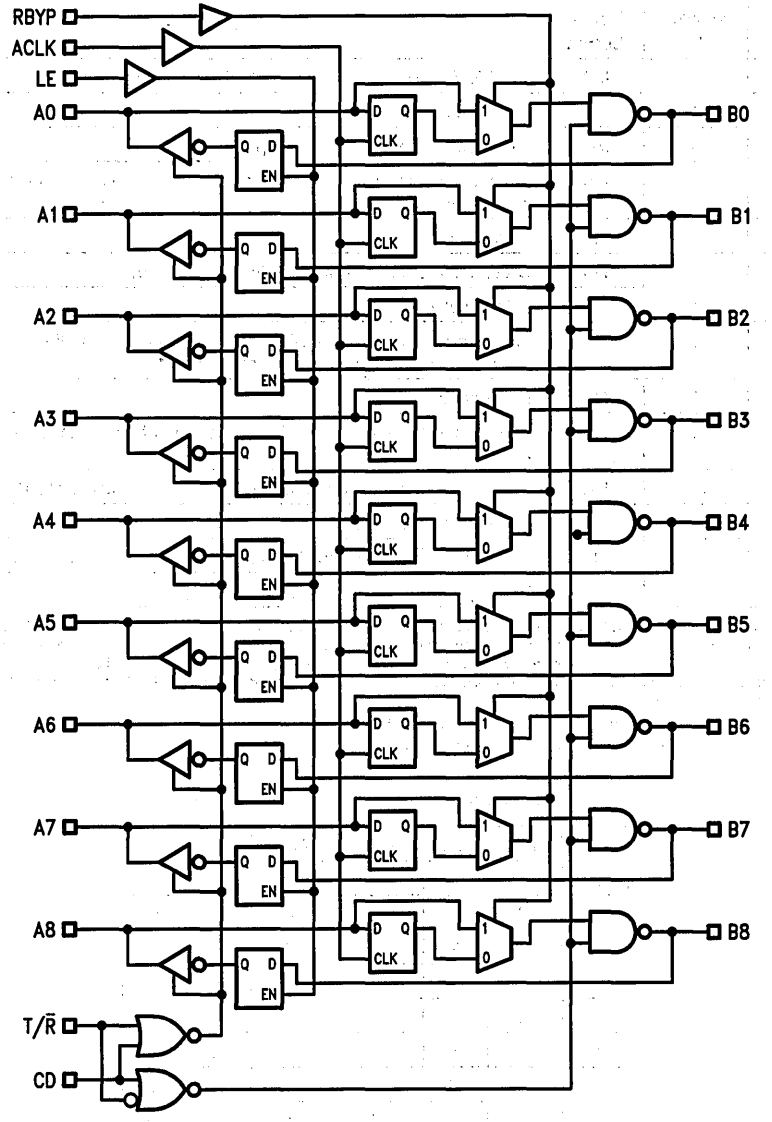
Note 11: The same consideration for ground was used for V_{CC} in reducing lead inductance (see Note: 10). QV_{CC} and V_{CC} should be tied together externally. If live insertion is not supported, the LI pin can be tied together with QV_{CC} and V_{CC}.

Truth Table

CD	T/$\overline{R}$	LE	RBYP	ACK	An	Bn
H	X	X	X	X	Z	H
L	H	X	H	X	L	H
L	H	X	H	X	H	L
L	H	X	L	L–H	H	L
L	H	X	L	L–H	L	H
L	H	X	L	X	X	An
L	L	H	X	X	H	L
L	L	H	X	X	L	H
L	L	L	X	X	Bn	X

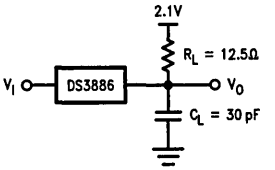
X = High or low logic state
Z = High impedance state
L = Low state
H = High state
L–H = Low to high transition

Logic Diagram



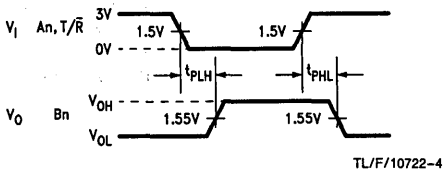
TL/F/10722–1

Test Circuit and Timing Waveforms



TL/F/10722–3

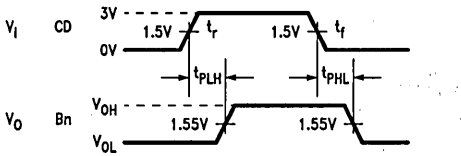
FIGURE 1. Driver Propagation Delay Set-up



TL/F/10722–4

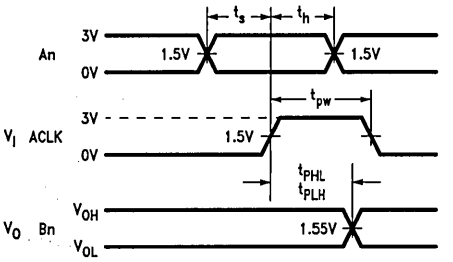
FIGURE 2. Driver: An to Bn, T/R̄ to Bn

Test Circuit and Timing Waveforms (Continued)



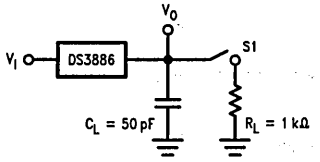
TL/F/10722-5

FIGURE 3. Driver: CD to Bn



TL/F/10722-6

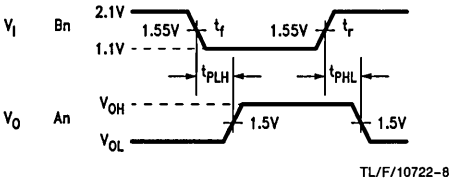
FIGURE 4. Driver: ACLK to Bn, t_S, t_h, t_{pw}



TL/F/10722-7

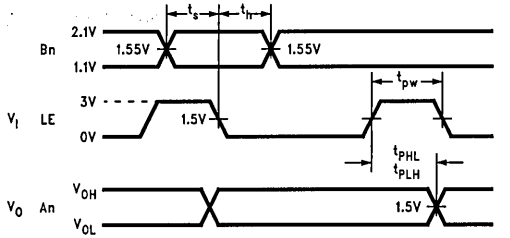
Switch Position		
	t_{PLH}	t_{PHL}
S1	open	close

FIGURE 5. Receiver Propagation Delay Set-up



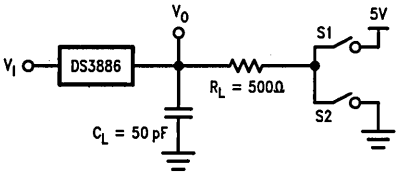
TL/F/10722-8

FIGURE 6. Receiver: Bn to An



TL/F/10722-9

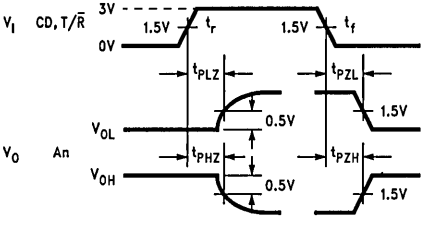
FIGURE 7. Receiver: LE to An, t_S, t_H, t_{pw}



TL/F/10722-10

FIGURE 8. Receiver Enable/Disable Set-up

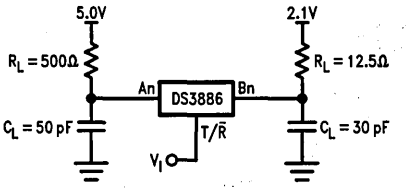
Switch Position		
	t_{PZL} t_{PLZ}	t_{PZH} t_{PHZ}
S1	close	open
S2	open	close



TL/F/10722-11

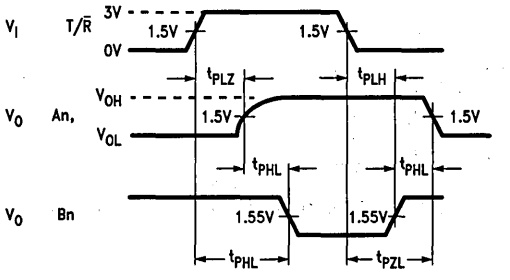
FIGURE 9. Receiver: CD to An, T/$\overline{R}$ to An

Test Circuit and Timing Waveforms (Continued)



TL/F/10722–14

FIGURE 10. T/R̄ to An, T/R̄ to Bn



TL/F/10722–15

FIGURE 11. t_{PHL} (T/R̄ to Bn), t_{PZL} (T/R̄ to An)

National Semiconductor

DS3862 Octal High Speed Trapezoidal Bus Transceiver

General Description

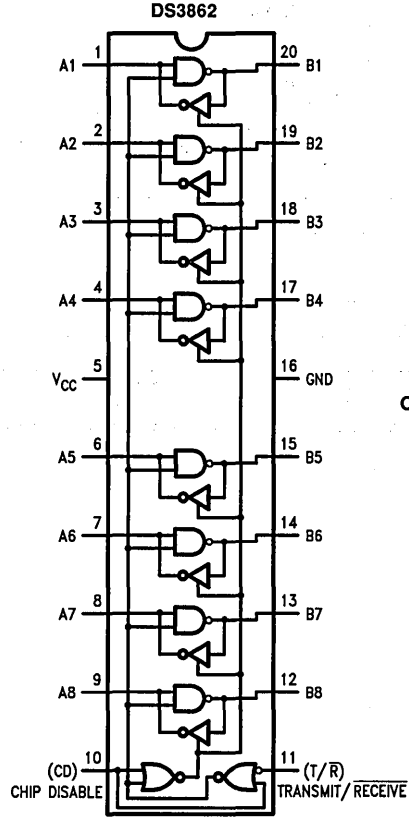
The DS3862 is an octal high speed schottky bus transceiver intended for use with terminated 120Ω impedance lines. It is specifically designed to reduce noise in unbalanced transmission systems. The open collector drivers generate precise trapezoidal waveforms with rise and fall times of 9 ns (typical), which are relatively independent of capacitive loading conditions on the outputs. This reduces noise coupling to the adjacent lines without any appreciable impact on the maximum data rate obtainable with high speed bus transceivers. In addition, the receivers use a low pass filter in conjunction with a high speed comparator, to further enhance the noise immunity. Tightly controlled threshold levels on the receiver provide equal rejection to both negative and positive going noise pulses on the bus.

The external termination is intended to be a 180Ω resistor from the bus to 5V logic supply, together with a 390Ω resistor from the bus to ground. The bus can be terminated at one or both ends.

Features

- Guaranteed A.C. specifications on noise immunity and propagation delay over the specified temperature and supply voltage range
- Temperature insensitive receiver thresholds track bus logic level and respond symmetrically to positive and negative going pulses
- Trapezoidal bus waveforms reduce noise coupling to adjacent lines
- Open collector driver output allows wire-or connection
- Advanced low power schottky technology
- Glitch free power up/down protection on driver and receiver outputs
- TTL compatible driver and control inputs, and receiver outputs
- Control logic is the same as the DS3896

Logic and Connection Diagram



Order Number DS3862J or DS3862N
See NS Package Number J20A or N20A

TL/F/8539–1

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Control Input Voltage	5.5V
Driver Input and Receiver Output	5.5V
Receiver Input and Driver Output	5.5V
Power Dissipation	1400 mW
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 4 seconds)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage, V_{CC}	4.75	5.25	V
Operating Free Air Temperature	0	70	°C

Electrical Characteristics $0°C \le T_A \le 70°C$, $4.75V \le V_{CC} \le 5.25V$ unless otherwise specified (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Driver and Control Inputs:						
V_{IH}	Logical "1" Input Voltage		2.0			V
V_{IL}	Logical "0" Input Voltage				0.8	V
I_I	Logical "1" Input Current	An = V_{CC}			1	mA
I_{IH}	Logical "1" Input Current	An = 2.4V			40	μA
I_{IL}	Logical "0" Input Current	An = 0.4V		−1	−1.6	mA
I_{IL}	CD & T/$\overline{R}$ Logical "0" Input Current	CD = T/$\overline{R}$ = 0.4V		−180	−400	μA
V_{CL}	Input Diode Clamp Voltage	Iclamp = −12 mA		−0.9	−1.5	V
Driver Output/Receiver Input						
V_{OLB}	Low Level Bus Voltage	An = T/$\overline{R}$ = 2V, Ibus = 100 mA		0.6	0.9	V
I_{IHB}	Logical "1" Bus Current	An = 0.8V, Bn = 4V, V_{CC} = 5.25V and 0V		10	100	μA
I_{ILB}	Logical "0" Bus Current	An = 0.8V, Bn = 0V, V_{CC} = 5.25V and 0V			100	μA
V_{TH}	Input Threshold	V_{CC} = 5V	1.5	1.7	1.9	V
Receiver Output						
V_{OH}	Logical "1" Output Voltage	Bn = 0.9V, I_{oh} = −400μA	2.4	3.2		V
V_{OL}	Logical "0" Output Voltage	Bn = 4V, I_{ol} = 16 mA		0.35	0.5	V
I_{OS}	Output Short Circuit Current	Bn = 0.9V	−20	−70	−100	mA
I_{CC}	Supply Current	V_{CC} = 5.25V		90	135	mA

Note 1: "Absolute Maximum Ratings" are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that device should be operated at these limits. The table of "Electrical Characteristics" provide conditions for actual device operation.

Note 2: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

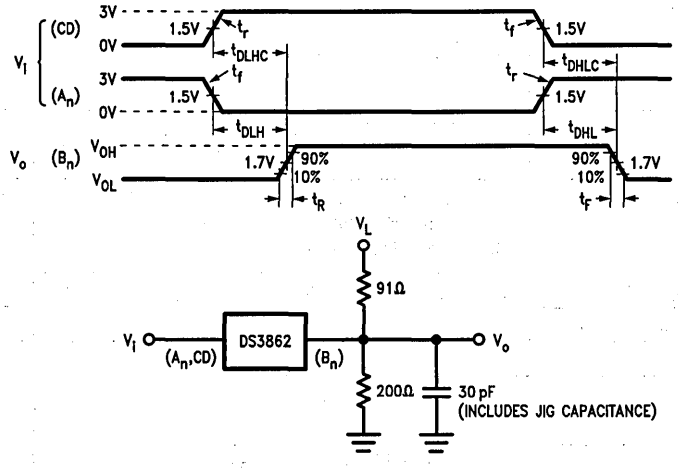
Note 3: All typicals are given for V_{CC} = 5V and T_A = 25°C.

Switching Characteristics $0°C \le T_A \le 70°C$, $4.75V \le V_{CC} \le 5.25V$ unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Driver:						
t_{DLH}	An to Bn	CD = 0.8V, T/$\overline{R}$ = 2.0V, VL = 5V (*Figure 1*)		12	20	ns
t_{DHL}				12	20	ns
t_{DLHC}	CD to Bn	An = T/$\overline{R}$ = 2.0V, VL = 5V, (*Figure 1*)		12	20	ns
t_{DHLC}				15	25	ns
t_{DLHT}	T/$\overline{R}$ to Bn	VCI = An, VC = 5V, (*Figure 2*) CD = 0.8V, RC = 390Ω, CL = 30 pF RL1 = 91Ω, RL2 = 200Ω, VL = 5V		20	30	ns
t_{DHLT}				25	40	ns
t_R	Driver Output Rise Time	CD = 0.8V, T/$\overline{R}$ = 2V, VL = 5V (*Figure 1*)	4	9	20	ns
t_F	Driver Output Fall Time		4	9	20	ns
Receiver:						
t_{RLH}	Bn to An	CD = 0.8V, T/$\overline{R}$ = 0.8V (*Figure 3*)		15	25	ns
t_{RHL}				15	25	ns
t_{RLZC}	CD to An	Bn = 2.0V, T/$\overline{R}$ = 0.8V, CL = 5 pF RL1 = 390Ω, RL2 = NC, VL = 5V (*Figure 4*)		15	25	ns
t_{RZLC}		Bn = 2.0V, T/$\overline{R}$ = 0.8V, CL = 30 pF RL1 = 390Ω, RL2 = 1.6K, VL = 5V (*Figure 4*)		10	20	ns
t_{RHZC}		Bn = 0.8V, T/$\overline{R}$ = 0.8V, VL = 0V, RL1 = 390Ω, RL2 = NC, CL = 5 pF (*Figure 4*)		5	10	ns
t_{RZHC}		Bn = 0.8V, T/$\overline{R}$ = 0.8V, VL = 0V, RL1 = NC, RL2 = 1.6K, CL = 30 pF (*Figure 4*)		8	15	ns
t_{RLZT}	T/$\overline{R}$ to An	VCI = Bn, VC = 3.4V, RC = 39Ω CD = 0.8V, VL = 5V, RL1 = 390Ω, RL2 = NC, CL = 5 pF (*Figure 2*)		20	30	ns
t_{RZLT}		VCI = Bn, VC = 3.4V, RC = 39Ω, CD = 0.8V, VL = 5V, RL1 = 390Ω, RL2 = 1.6K, CL = 30 pF (*Figure 2*)		30	45	ns
t_{RHZT}		VCI = Bn, VC = 0V, RC = 39Ω CD = 0.8V, VL = 0V, RL1 = 390Ω, RL2 = NC, CL = 5 pF (*Figure 2*)		5	10	ns
t_{RZHT}		VCI = Bn, VC = 0V, RC = 39Ω, CD = 0.8V, VL = 0V, RL1 = NC RL2 = 1.6K, CL = 30 pF (*Figure 2*)		10	20	ns
t_{NR}	Receiver Noise Rejection Pulse Width	(*Figure 5*)	9	12		ns

Note: NC means open

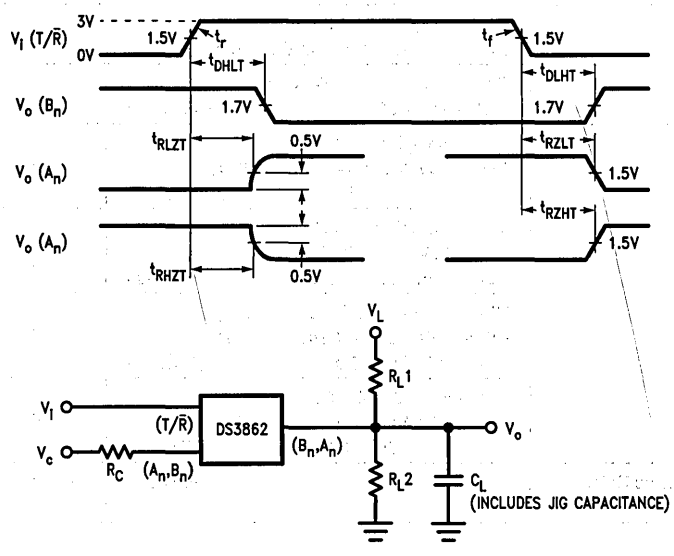
Switching Waveforms



TL/F/8539–2

Note: $t_r = t_f \leq 5$ ns from 10% to 90%

FIGURE 1. Driver Propagation Delays

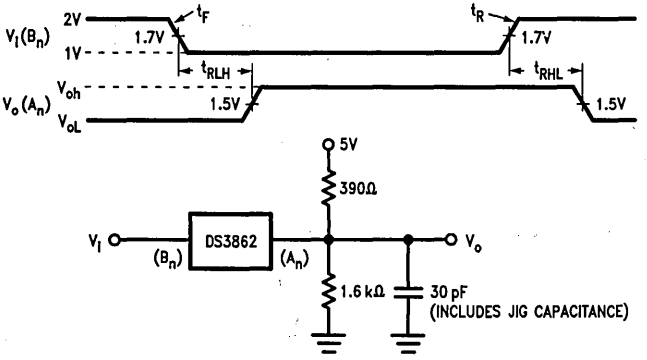


TL/F/8539–3

Note: $t_r = t_f \leq 5$ ns from 10% to 90%

FIGURE 2. Propagation Delay From T/$\overline{R}$ Pin to An or Bn.

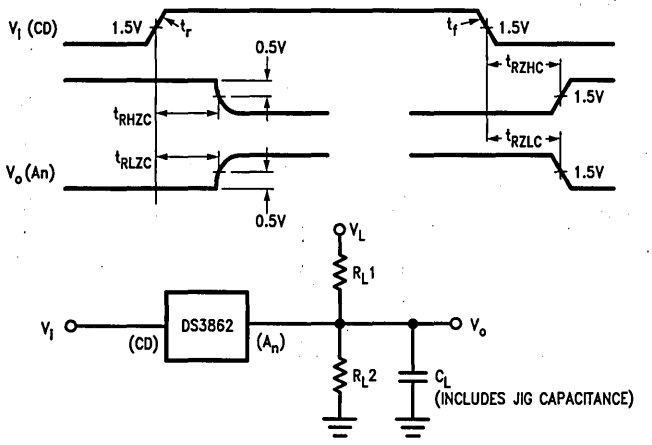
Switching Waveforms (Continued)



TL/F/8539–4

Note: $t_R = t_F \leq 10$ ns from 10% to 90%

FIGURE 3. Receiver Propagation Delays

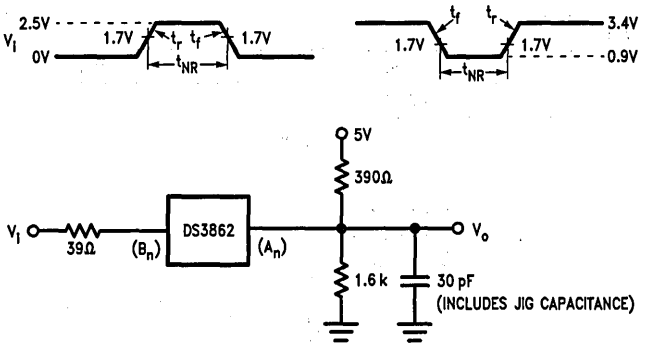


TL/F/8539–5

Note: $t_r = t_f \leq 5$ ns from 10% to 90%

FIGURE 4. Propagation Delay From CD Pin to An

Switching Waveforms (Continued)

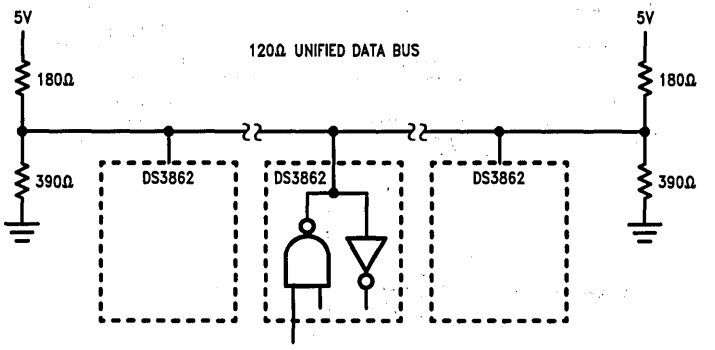


TL/F/8539–6

Note: $t_r = t_f = 2$ ns from 10% to 90%

FIGURE 5. Receiver Noise Immunity: No Response at Output Input Waveform.

Typical Application



TL/F/8539–7

National Semiconductor

DS3890 BTL Octal Trapezoidal Driver
DS3892 BTL Octal TRI-STATE® Receiver
DS3898 BTL Octal Trapezoidal Repeater

General Description

The DS3890, DS3892 and DS3898 are designed specifically to overcome problems associated with driving densely populated backplanes. These products provide significant improvement in both speed and data integrity in comparison to conventional bus drivers and receivers. Their low output capacitance, low voltage swing and noise immunity features make them ideal for driving low impedance busses with minimum power dissipation.

The DS3890 and DS3898 feature open collector outputs that generate precise trapezoidal waveforms with typical rise and fall times of 6 ns which are relatively independent of capacitive loading conditions. These controlled output characteristics significantly reduce noise coupling to adjacent lines.

To minimize bus loading, the DS3890 and DS3898 also feature a schottky diode in series with the open collector outputs that isolates the driver output capacitance in the disabled state. With this type of configuration the output low voltage is typically "1V". The output high level is intended to be 2 volts. This is achieved by terminating the bus with a pull up resistor. Both devices can drive an equivalent DC load of 18.5Ω (or greater) in the defined configuration.

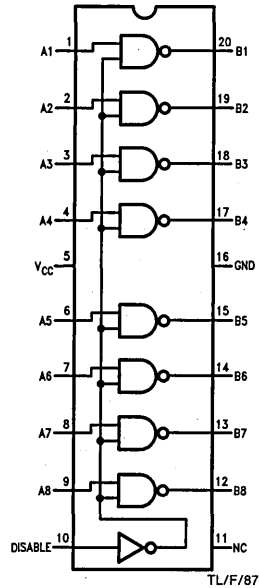
(General Description to be continued)

Features

- Driver output capacitance less than 5 pF
- 1 volt bus signal reduces power consumption
- Trapezoidal driver waveforms (t_r, t_f, typically 6 ns) reduces noise coupling to adjacent lines
- Precise receiver threshold track the bus logic high level to maximize noise immunity in both logic high ahd low states
- Open collector driver output allows wire-or connection
- Advanced low power schottky technology
- Glitch free power up/down protection
- TTL compatible driver and control inputs and receiver output
- BTL compatible

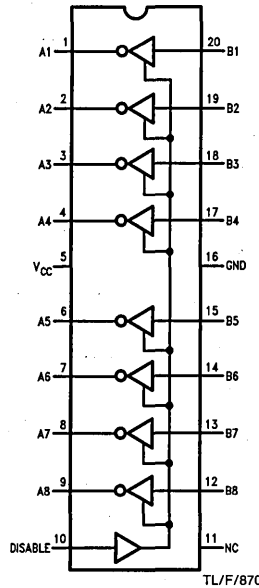
Logic and Connection Diagrams

DS3890 Octal Trapezoidal Driver



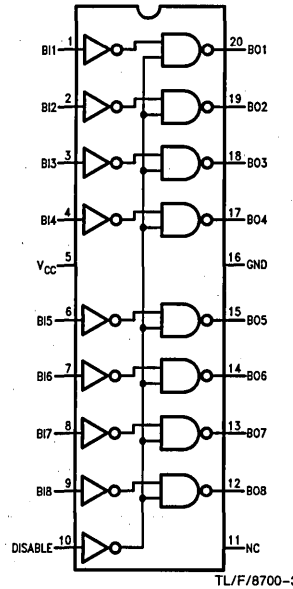
TL/F/8700–1

DS3892 Octal TRI-STATE Receiver



TL/F/8700–2

DS3898 Octal Trapezoidal Repeater



TL/F/8700–3

Order Numbers DS3890J, N, DS3892J, N or DS3898J, N
See NS Package Number J20A or N20A

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Control Input Voltage	5.5V
Driver Input and Receiver Output	5.5V
Receiver Input and Driver Output	2.5V
Storage Temperature Range	−65°C to +165°C
Lead Temperature (Soldering, 4 sec.)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage	4.75	5.25	V
Temperature (T_A)	0	70	°C

DS3890 Electrical Characteristics (Notes 2 and 3)

DRIVER AND CONTROL INPUTS

Symbol	Conditions	Min	Typ	Max	Units
V_{IH}		2.0			V
V_{IL}				0.8	V
I_{IL} An	V_{CC}=Max V_{IN}=0.4V		−1	−1.6	mA
I_{IL} Dis	V_{CC}=Max V_{IN}=0.4V		−180	−400	μA
I_{IH}	V_{CC}=Max V_{IN}=2.4V			40	μA
I_I	V_{CC}=Max V_{IN}=5.25V			1	mA
V_{CL}	V_{CC}=Min I_{IN}=−12 mA		−0.9	−1.5	V

DRIVER OUTPUT

V_{OL}	V_{CC}=Min R_L=18.5Ω	0.75	1.0	1.2	V
I_{OH}	V_{CC}=Max V_{OUT}=2V	−20	10	100	μA
I_O	V_{CC}=0V V_{OUT}=2V			100	μA
I_{IL}	V_{CC}=Max V_{OUT}=0.75V		−100	−250	μA
I_{CC} Low	V_{CC}=Max		50	80	mA
I_{CC} High				100	mA

DS3892 Electrical Characteristics (Notes 2 and 3)

CONTROL INPUTS

Symbol	Conditions	Min	Typ	Max	Units
V_{IH}		2.0			V
V_{IL}				0.8	V
I_{IL}	V_{CC}=Max V_{IN}=0.4V		−180	−400	μA
I_{IH}	V_{CC}=Max V_{IN}=2.4V			40	μA
I_I	V_{CC}=Max V_{IN}=5.25V			1	mA
V_{CL}	V_{CC}=Min I_{IN}=−12 mA		−0.9	−1.5	V

RECEIVER

V_{OL}	V_{CC}=Min I_{OL}=16 mA		0.35	0.5	V
V_{OH}	V_{CC}=Min I_{OH}=−400 μA	2.4	3.2		V
I_{OS}	V_{CC}=Max V_{OUT}=0V	−20	−70	−100	mA
V_{TH} Rec	V_{CC}=5V	1.47	1.55	1.62	V
I_{IH} Rec	V_{CC}=Max V_{IN}=2V		10	100	μA
I_I Rec	V_{CC}=0V V_{IN}=2V			100	μA
I_{IL} Rec	V_{CC}=Max V_{IN}=0.75V			100	μA
I_{CC} Low	V_{CC}=Max			80	mA
I_{CC} High				60	mA

DS3898 Electrical Characteristics (Notes 2 and 3)

CONTROL INPUTS

Symbol	Conditions	Min	Typ	Max	Units
V_{IH}		2.0			V
V_{IL}				0.8	V
I_{IL}	V_{CC}=Max V_{IN}=0.4V		−180	−400	μA
I_{IH}	V_{CC}=Max V_{IN}=2.4V			40	μA
I_I	V_{CC}=Max V_{IN}=5.25V			1	mA
V_{CL}	V_{CC}=Min I_{IN}=−12 mA		−0.9	−1.5	V
RECEIVER INPUT					
V_{TH} Rec	V_{CC}=5V	1.47	1.55	1.62	V
I_{IH} Rec	V_{CC}=Max V_{IN}=2V		10	100	μA
I_I Rec	V_{CC}=0V V_{IN}=2V			100	μA
I_{IL} Rec	V_{CC}=Max V_{IN}=0.75V			100	μA
DRIVER OUTPUT					
V_{OL}	V_{CC}=Min R_L=18.5Ω	0.75	1.0	1.2	V
I_{OH}	V_{CC}=Max V_{OUT}=2V	−20	10	100	μA
I_O	V_{CC}=0V V_{OUT}=2V			100	μA
I_{IL}	V_{CC}=Max V_{OUT}=0.75V		−100	−250	mA
I_{CC} Low	V_{CC}=Max		90	135	mA
I_{CC} High			95	140	mA

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The Table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All currents into device pins are shown as positive values; all currents out of the device are shown as negative; all voltages are referenced to ground unless otherwise specified. All values shown as max or min are classified on absolute value basis and apply to the full operating temperature and V_{CC} range.

Note 3: All typical values are V_{CC}=5V, T_A= 25°C.

DS3890 Switching Characteristics *(Figure 1)*

(0°C≤T_A≤70°C, 4.75V≤V_{CC}≤5.25V unless otherwise specified)

Symbol	Conditions	Min	Typ	Max	Units
T_{dLH}	An to Bn		9	15	ns
T_{dHL}			9	15	ns
T_{dLH}	Dis to Bn		10	18	ns
T_{dHL}			12	20	ns
T_r & T_f	Bn rise and fall time	3	6	10	ns

DS3892 Switching Characteristics (*Figures 2, 3* and *4*)

Symbol	Conditions	Min	Typ	Max	Units
T_{dLH}	Bn to An		12	18	ns
T_{pHL}			10	18	ns
T_{dLZ}	Dis to An		10	18	ns
T_{dZL}			8	15	ns
T_{dHZ}			4	8	ns
T_{dZH}			7	12	ns
TNR	Receiver noise rejection	3	6		ns

DS3898 Switching Characteristics (*Figures 4* and *5*)

Symbol	Conditions	Min	Typ	Max	Units
T_{dLH}	Bi to BOn		20	30	ns
T_{dHL}			20	30	ns
T_{dLH}	Dis to BOn		10	18	ns
T_{dHL}			12	20	ns
T_r & T_f	Bn rise and fall time	3	6	10	ns
TNR	Receiver noise rejection	3	6		ns

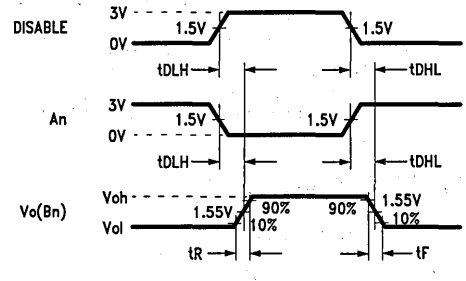
General Descriptions (Continued)

The DS3892 and DS3898 receiver inputs incorporate a low pass filter in conjunction with high speed comparator to further enhance the noise immunity. Both devices provide equal rejection to both positive and negative noise pulses (typically 6 ns) on the bus.

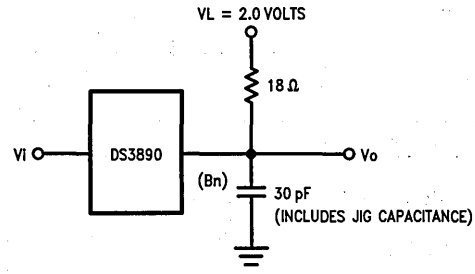
The DS3890 features TTL compatible inputs while both the DS3892 and DS3898 inputs are BTL compatible. The control inputs on all devices are TTL compatible.

BTL "Backplane Transceiver Logic" is a new logic signaling method developed by IEEE P896 Future Bus Standards Committee. This standard was adopted to enhance the performance of Backplane Busses. BTL compatible bus interface circuits feature low capacitance drivers to minimize bus loading, a 1V nominal signal swing for reduced power consumption and receivers with precision thresholds for maximum noise immunity. This new standard overcomes some of the fundamental limitations of TTL bus transceivers in heavily loaded backplane bus applications. Devices designed to this standard provide significant improvements in switching speed and data integrity.

AC Switching Waveforms



TL/F/8700–4



Note: $t_R = t_F < 10$ ns from 10% to 90%

TL/F/8700–5

FIGURE 1
Driver Propagation Delays

AC Switching Waveforms (Continued)

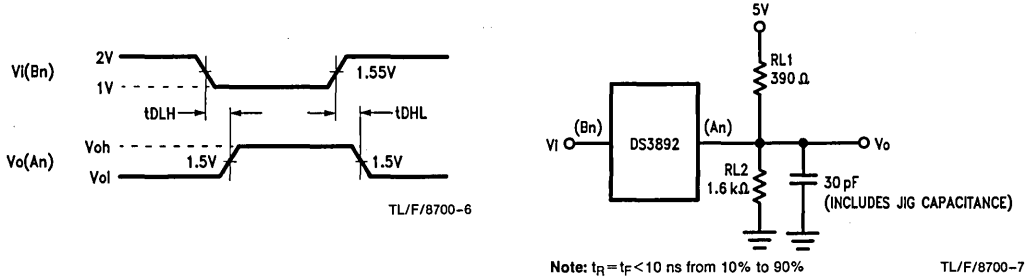
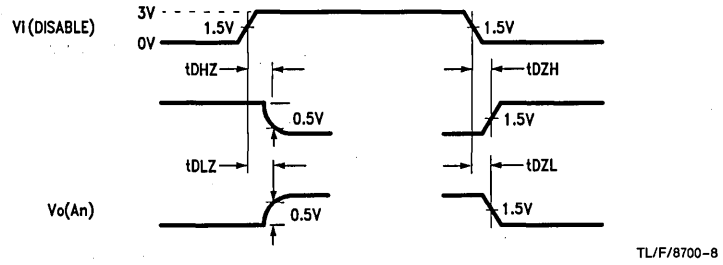
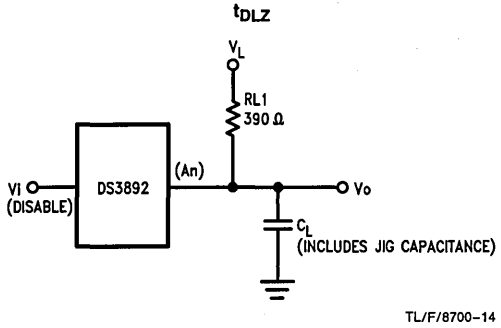
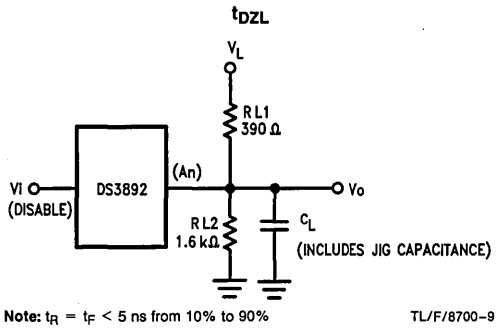
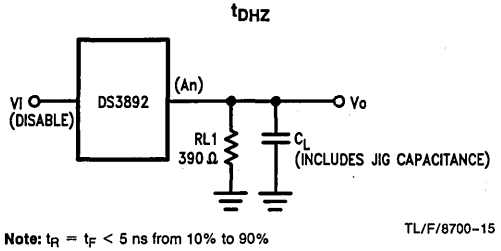


FIGURE 2. Receiver Propagation Delays









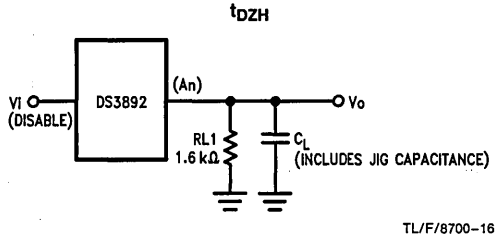
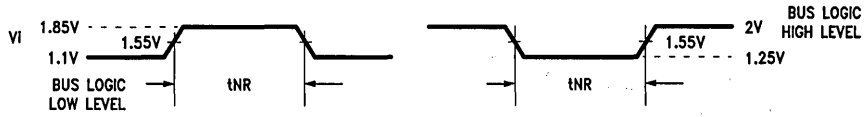


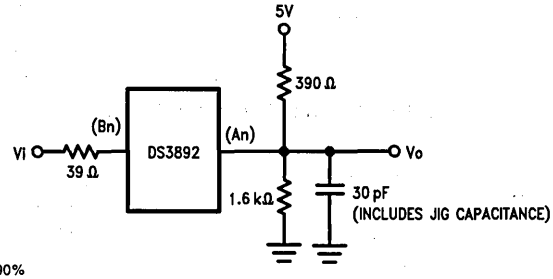
FIGURE 3. Propagation Delay from Disable Pin to An

AC Switching Waveforms (Continued)



TL/F/8700–10

High **Low**

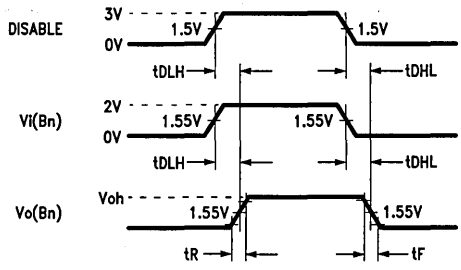


Note: $t_R = t_F < 2$ ns from 10% to 90%

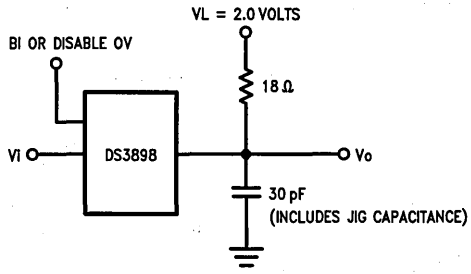
TL/F/8700–11

FIGURE 4

Receiver Noise Immunity:
"No Response at Output" Input Waveforms



TL/F/8700–12



Note: $t_R = t_F < 10$ ns from 10% to 90%

TL/F/8700–13

FIGURE 5

Repeater Propagation Delays

National Semiconductor

DS3893A BTL TURBOTRANSCEIVER™

General Description

The TURBOTRANSCEIVER is designed for use in very high speed bus systems. The bus terminal characteristics of the TURBOTRANSCEIVER are referred to as "Backplane Transceiver Logic" (BTL). BTL is a new logic signaling standard that has been developed to enhance the performance of backplane buses. BTL compatible transceivers feature low output capacitance drivers to minimize bus loading, a 1V nominal signal swing for reduced power consumption and receivers with precision thresholds for maximum noise immunity. This new standard eliminates the settling time delays, that severely limit the TTL bus performance, to provide significantly higher bus transfer rates.

The TURBOTRANSCEIVER is compatible with the requirements of the proposed IEEE 896 Futurebus draft standard. It is similar to the DS3896/97 BTL TRAPEZOIDAL Transceivers but the trapezoidal feature has been removed to improve the propagation delay. A stripline backplane is therefore required to reduce the crosstalk induced by the faster rise and fall times. This device can drive a 10Ω load with a typical propagation delay of 3.5 ns for the driver and 5 ns for the receiver.

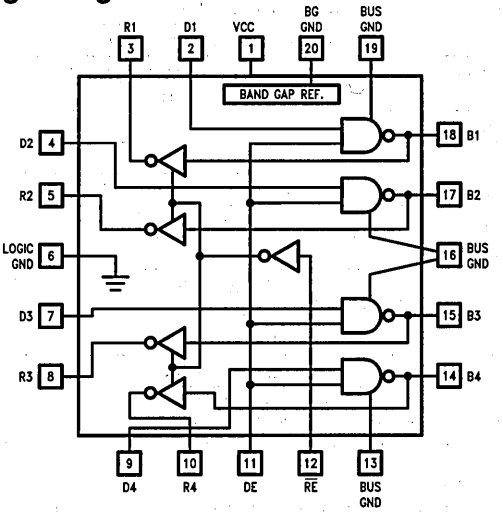
When multiple devices are used to drive a parallel bus, the driver enables can be tied together and used as a common control line to get on and off the bus. The driver enable delay is designed to be the same as the driver propagation delay in order to provide maximum speed in this configuration. The low input current on the enable pin eases the drive required for the common control line.

The bus driver is an open collector NPN with a Schottky diode in series to isolate the transistor output capacitance from the bus when the driver is in the inactive state. The active output low voltage is typically 1V. The bus is intended to be operated with termination resistors (selected to match the bus impedance) to 2.1V at both ends. Each of the resistors can be as low as 20Ω.

Features

- Fast single ended transceiver (typical driver enable and receiver propagation delays are 3.5 ns and 5 ns)
- Backplane Transceiver Logic (BTL) levels (1V logic swing)
- Less than 5 pF bus-port capacitance
- Drives densely loaded backplanes with equivalent load impedances down to 10Ω
- 4 transceivers in 20 pin PCC package
- Specially designed for stripline backplanes
- Separate bus ground returns for each driver to minimize ground noise
- High impedance, MOS and TTL compatible inputs
- TRI-STATE® control for receiver outputs
- Built-in bandgap reference provides accurate receiver threshold
- Glitch free power up/down protection on all outputs
- Oxide isolated bipolar technology

Connection and Logic Diagram



TL/F/8698–1

Order Number DS3893AV
See NS Package Number V20A

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6.5V
Control Input Voltage	5.5V
Driver Input and Receiver Output	5.5V
Driver Output Receiver Input Clamp Current	±15 mA
Power Dissipation at 70°C	900 mW
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 3 sec.)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage, V_{CC}	4.5	5.5	V
Bus Termination Voltage (V_T)	2.0	2.2	V
Operating Free Air Temperature	0	70	°C

Electrical Characteristics (Notes 2, 3 and 4) T_A = 0 to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DRIVER AND CONTROL INPUT: (DE, $\overline{RE}$, Dn)						
V_{IH}	Input High Voltage		2.0			V
V_{IL}	Input Low Voltage				0.8	V
I_I	Input Leakage Current	DE = $\overline{RE}$ = Dn = V_{CC}			100	µA
I_{IH}	Input High Current	DE = $\overline{RE}$ = Dn = 2.5V			20	µA
I_{IL}	Dn Input Low Current	Dn = 0.5V, DE = V_{CC} = Max			−200	µA
	DE Input Low Current	DE = 0.5V, Dn = V_{CC} = Max			−500	µA
	$\overline{RE}$ Input Low Current	$\overline{RE}$ = 0.5V, V_{CC} = Max			−100	µA
V_{CL}	Input Diode Clamp Voltage	I_{clamp} = −12 mA			−1.2	V
DRIVER OUTPUT/RECEIVER INPUT: (Bn)						
V_{OLB}	Output Low Bus Voltage	Dn = DE = V_{IH} *(Figure 2)* R_T = 10Ω, V_T = 2.2V	0.75	1.0	1.2	V
		Dn = DE = V_{IH} *(Figure 2)* R_T = 18.5Ω, V_T = 2.14	0.75	1.0	1.1	V
I_{ILB}	Output Bus Current (Power On)	Dn = DE = 0.8V, V_{CC} = Max Bn = 0.75V	−250		100	µA
I_{IHB}	Output Bus Current (Power Off)	Dn = DE = 0.8V, V_{CC} = 0V Bn = 1.2V			100	µA
V_{OCB}	Driver Output Positive Clamp	V_{CC} = Max or 0V, Bn = 1 mA			2.9	V
		V_{CC} = Max or 0V, Bn = 10 mA			3.2	V
V_{OHB}	Output High Bus Voltage	V_{CC} = Max, Dn = 0.8V *(Figure 2)* V_T = 2.0V, R_T = 10Ω	1.90			V
V_{TH}	Receiver Input Threshold		1.47	1.55	1.62	V
RECEIVER OUTPUT: (Rn)						
V_{OH}	Voltage Output High	Bn = 1.2V, I_{oh} = −3 mA, $\overline{RE}$ = 0.8V	2.5V			V
V_{OL}	Voltage Output Low	Bn = 2V, I_{ol} = 6 mA, $\overline{RE}$ = 0.8V		0.35	0.5	V
I_{OZ}	TRI-STATE Leakage	V_o = 2.5V, $\overline{RE}$ = 2V			20	µA
		V_o = 0.5V, $\overline{RE}$ = 2V			−20	µA
I_{OS}	Output Short Circuit Current (Note 5)	Bn = 1.2V, V_o = 0V $\overline{RE}$ = 0.8V, V_{CC} = Max	−80	−120	−200	mA
I_{CC}	Supply Current	Dn = DE = $\overline{RE}$ = V_{IH}, V_{CC} = Max		70	95	mA

Note 1: "Absolute maximum ratings" are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provide conditions for actual device operation.

Note 2: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 3: All typicals are given for V_{CC} = 5V and T_A = 25°C.

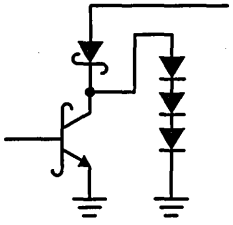
Note 4: Unused inputs should not be left floating. Tie unused inputs to either V_{CC} or GND thru a resistor.

Note 5: Only one output at a time should be shorted.

Switching Characteristics $T_A = 0$ to $+70°C$, $V_{CC} = 5V \pm 10\%$

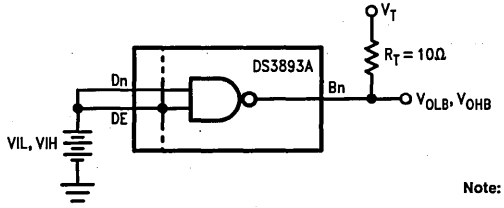
Symbol	Parameter	Conditions	Min	Typ	Max	Units
DRIVER: (*Figures 3* and *6*)						
t_{PHL}	Driver Input to Output	$V_T = 2V$ $R_T = 10\Omega$, $C_L = 30$ pF, DE = 3V	1	3.5	7	ns
t_{PLH}	Driver Input to Output	$V_T = 2V$, $R_T = 10\Omega$, $C_L = 30$ pF, DE = 3V	1	3.5	7	ns
t_r	Output Rise time	$V_T = 2V$, $R_T = 10\Omega$, $C_L = 30$ pF, DE = 3V	1	2	5	ns
t_f	Output Fall Time	$V_T = 2V$, $R_T = 10\Omega$, $C_L = 30$ pF, DE = 3V	1	2	5	ns
t_{skew}	Skew Between Drivers in Same Package	(Note 1)		1		ns
DRIVER ENABLE: (*Figures 3* and *6*)						
t_{PHL}	Enable Delay	$V_T = 2V$, $R_T = 10\Omega$, $C_L = 30$ pF, Dn = 3V	1	3.5	7	ns
t_{PLH}	Disable Delay	$V_T = 2V$, $R_T = 10\Omega$, $C_L = 30$ pF, Dn = 3V	1	3.5	7	ns
RECEIVER: (*Figures 4* and *7*)						
t_{PHL}	Receiver Input to Output	$C_L = 50$ pF, $\overline{RE}$ = DE = 0.3V, S3 Closed	2	5	8	ns
t_{PLH}	Receiver Input to Output	$C_L = 50$ pF, $\overline{RE}$ = DE = 0.3V, S3 Open	2	5	8	ns
t_{skew}	Skew Between Receivers in Same Package	(Note 1)		1		ns
RECEIVER ENABLE: (*Figures 5* and *8*)						
t_{ZL}	Receiver Enable to Output Low	$C_L = 50$ pF, $R_L = 500$, DE = 0.3V S2 Open Bn = 2V	2	6	12	ns
t_{ZH}	Receiver Enable to Output High	$C_L = 50$ pF, $R_L = 500$, DE = 0.3V S1 Open Bn = 1V	2	5	12	ns
t_{LZ}	Receiver Disable From Output Low	$C_L = 50$ pF, $R_L = 500$, DE = 0.3V S2 Open Bn = 2V	1	5	8	ns
t_{HZ}	Receiver Disable From Output High	$C_L = 50$ pF, $R_L = 500$, DE = 0.3V S1 Open Bn = 1V	1	4	8	ns

Note 1: t_D and t_R skew is an absolute value, defined as differences seen in propagation delays between each of the drivers or receivers in the same package of the same delay, V_{CC}, temperature and load conditions.



TL/F/8698–12

FIGURE 1. Equivalent Bus Output

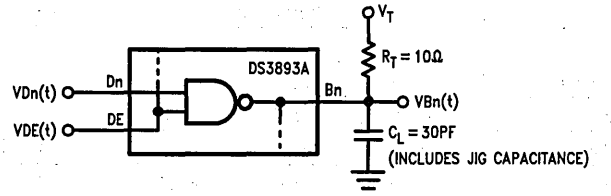


Note: n = 1, 2, 3, 4

TL/F/8698–2

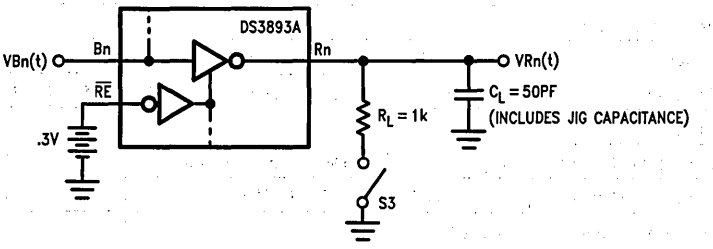
FIGURE 2. Driver Output Voltage

AC Test Circuits



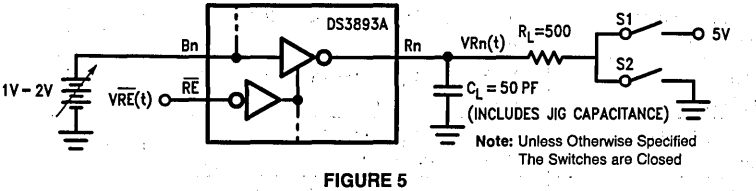
TL/F/8698–3

FIGURE 3



TL/F/8698–4

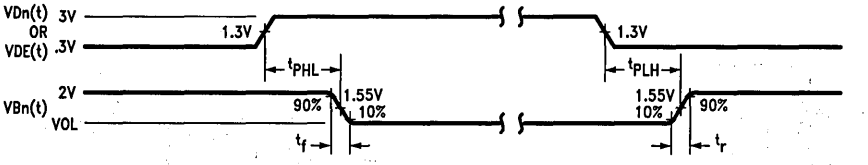
FIGURE 4



TL/F/8698–5

FIGURE 5

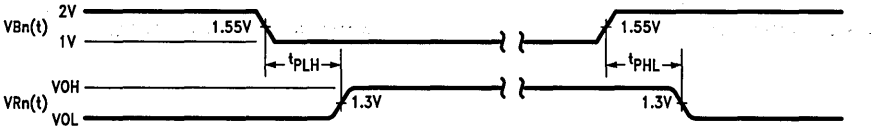
Switching Time Waveforms



$t_R = t_F \leq 4ns$ FROM 10% TO 90%

TL/F/8698–6

FIGURE 6. Driver Propagation Delay

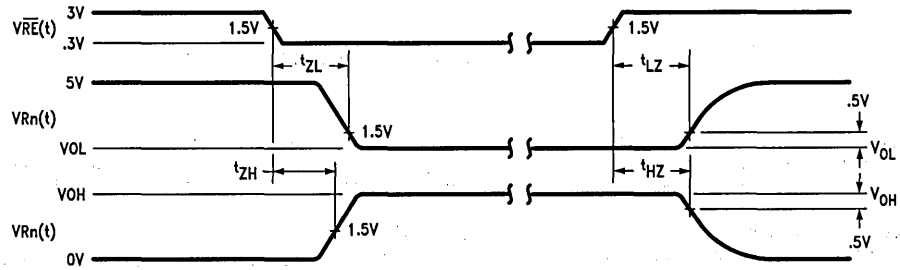


$t_R = t_F \leq 4ns$ FROM 10% TO 90%

TL/F/8698–7

FIGURE 7. Receiver Propagation Delay

Switching Time Waveforms (Continued)



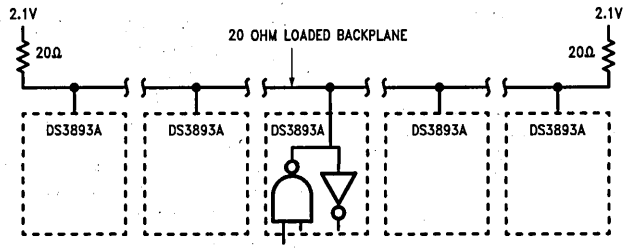
Note: $t_R = t_F \leq 4$ ns From 10% to 90%

Note: n = 1, 2, 3, 4

TL/F/8698–8

FIGURE 8. Receiver Enable and Disable Times

Typical Application



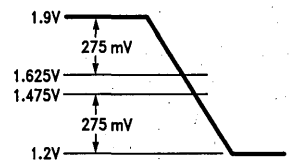
TL/F/8698–9

Application Information

Due to the high current and very high speed capability of the TURBOTRANSCEIVER's driver output stage, circuit board layout and bus grounding are critical factors that affect the system performance.

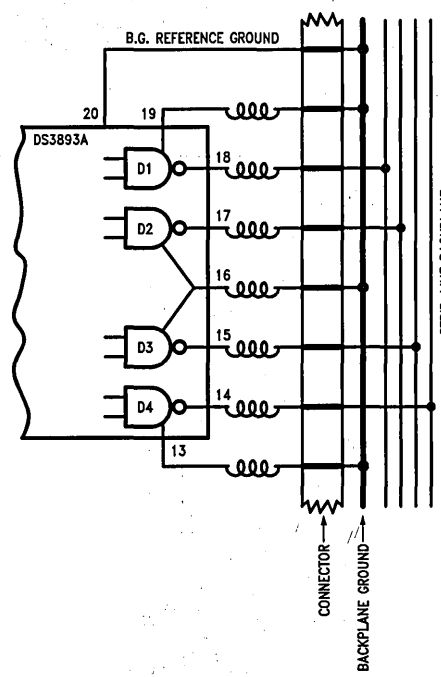
Each of the TURBOTRANSCEIVER's bus ground pins should be connected to the nearest backplane ground pin with the shortest possible path. The ground pins on the connector should be distributed evenly through its length.

Although the bandgap reference receiver threshold provides sufficient DC noise margin *(Figure 9)*, ground noise and ringing on the data paths could easily exceed this margin if the series inductance of the traces and connectors are not kept to a minimum. The bandgap ground pin should be returned to the connector through a separate trace that does not carry transient switching currents. The transceivers should be mounted as close as possible to the connector. It should be noted that even one inch of trace can add a significant amount of ringing to the bus signal.



TL/F/8698–10

FIGURE 9. Noise Margin



TL/F/8698–11

FIGURE 10

National Semiconductor

DS3896/DS3897 BTL Trapezoidal™ Transceivers

General Description

These advanced transceivers are specifically designed to overcome problems associated with driving a densely populated backplane, and thus provide significant improvement in both speed and data integrity. Their low output capacitance, low output signal swing and noise immunity features make them ideal for driving low impedance buses with minimum power consumption.

The DS3896 is an octal high speed schottky bus transceiver with common control signals, whereas the DS3897 is a quad device with independent driver input and receiver output pins. The DS3897 has a separate driver disable for each driver and is, therefore, suitable for arbitration lines. On the other hand, the DS3896 provides high package density for data/address lines.

The open collector drivers generate precise trapezoidal waveforms, which are relatively independent of capacitive loading conditions on the outputs. This significantly reduces noise coupling to adjacent lines. In addition, the receivers use a low pass filter in conjunction with a high speed comparator, to further enhance the noise immunity and provide equal rejection to both negative and positive going noise pulses on the bus.

To minimize bus loading, these devices also feature a schottky diode in series with the open collector output that isolates the driver output capacitance in the disabled state. The output low voltage is typically "1V" and the output high level is intended to be 2V. This is achieved by terminating the bus with a pull up resistor to 2V at both ends. The device can drive an equivalent DC load of 18.5Ω (or greater) in the above configuration.

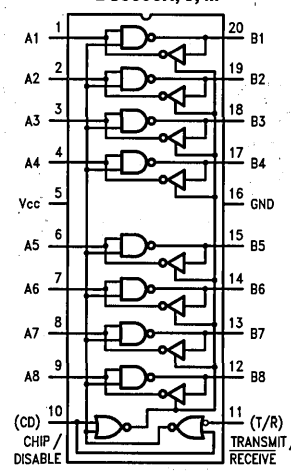
These signalling requirements, including a 1 volt signal swing, low output capacitance and precise receiver thresholds are referred to as Bus Transceiver Logic (BTL™).

Features

- 8 bit DS3896 transceiver provides high package density
- 4 bit DS3897 transceiver provides separate driver input and receiver output pins
- BTL compatible
- Less than 5 pF output capacitance for minimal bus loading
- 1 Volt bus signal swing reduces power consumption
- Trapezoidal driver waveforms (t_r, $t_f \cong 6$ ns typical) reduce noise coupling to adjacent lines
- Temperature insensitive receiver thresholds track the bus logic high level to maximize noise immunity in both high and low states
- Guaranteed A.C. specifications on noise immunity and propagation delay over the specified temperature and supply voltage range
- Open collector driver output allows wire-or connection
- Advanced low power schottky technology
- Glitch free power up/down protection on driver and receiver outputs
- TTL compatible driver and control inputs and receiver outputs

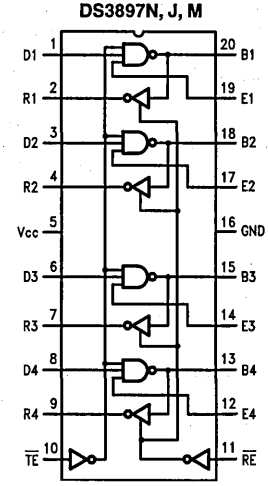
Logic Diagrams

DS3896N, J, M



TL/F/8510–1

DS3897N, J, M



TL/F/8510–2

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Control Input Voltage	5.5V
Driver Input and Receiver Output	5.5V
Receiver Input and Driver Output	2.5V
Power Dissipation at 70°C N Package	1480 mW
J Package	1250 mW
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 4 sec.)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage, V_{CC}	4.75	5.25	V
Bus Termination Voltage	1.90	2.10	V
Operating Free Air Temperature	0	70	°C

Electrical Characteristics: (Note 2 and 3) ($0°C \le T_A \le 70°C$, $4.75V \le V_{CC} \le 5.25V$ unless otherwise specified)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Driver and Control Inputs: (An, Dn, En, CD, T/$\overline{R}$, $\overline{RE}$, $\overline{TE}$)						
V_{IH}	Logical "1" Input Voltage		2.0			V
V_{IL}	Logical "0" Input Voltage				0.8	V
I_I	Logical "1" Input Current	An = Dn = En = V_{CC}			1	mA
I_{IH}	Logical "1" Input Current	An = Dn = En = 2.4V			40	μA
I_{IL}	Logical "0" Input Current	An = Dn = En = 0.4V		−1	−1.6	mA
I_{ILC}	Logical "0" Input Current	CD = T/$\overline{R}$ = $\overline{RE}$ = $\overline{TE}$ = 0.4V		−180	−400	μA
V_{CL}	Input Diode Clamp Voltage	Iclamp = −12 mA		−0.9	−1.5	V
Driver Output/Receiver Input: (Bn)						
V_{OLB}	Low Level Bus Voltage	An = Dn = En = T/$\overline{R}$ = 2V, VL = 2V RL = 18.5Ω, CD = $\overline{TE}$ = 0.8V (*Figure 1*)	0.75	1.0	1.2	V
I_{IHB}	Maximum Bus Current (Power On)	An = Dn = En = 0.8V, V_{CC} = 5.25V Bn = 2V		10	100	μA
I_{ILB}	Maximum Bus Current (Power Off)	An = Dn = En = 0.8V, V_{CC} = 0V Bn = 2V			100	μA
V_{TH}	Receiver Input Threshold	V_{CC} = 5V	1.47	1.55	1.62	V
Receiver Output: (An, Rn)						
V_{OH}	Logical "1" Output Voltage	Bn = 1.2V, I_{OH} = −400 μA CD = T/$\overline{R}$ = $\overline{RE}$ = 0.8V	2.4	3.2		V
V_{OL}	Logical "0" Output Voltage	Bn = 2V, I_{OL} = 16 mA CD = T/$\overline{R}$ = $\overline{RE}$ = 0.8V		0.35	0.5	V
I_{OS}	Output Short Circuit Current	Bn = 1.2V CD = T/$\overline{R}$ = $\overline{RE}$ = 0.8V	−20	−70	−100	mA
I_{CC}	Supply Current (DS3896)	V_{CC} = 5.25V		90	135	mA
I_{CC}	Supply Current (DS3897)	V_{CC} = 5.25V		50	80	mA

Note 1. "Absolute maximum ratings" are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristic" provide conditions for actual device operation.

Note 2. All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 3. All typicals are given for V_{CC} = 5V and T_a = 25°C.

DS3896 Switching Characteristics

(0°C ≤ T_A ≤ 70°C, 4.75V ≤ V_{CC} ≤ 5.25V unless otherwise specified)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Driver:						
t_{DLH}	An to Bn	CD = 0.8V, T/$\overline{R}$ = 2.0V, VL = 2V	5	9	15	ns
t_{DHL}		*(Figure 2)*	5	9	15	ns
t_{DLHC}	CD to Bn	An = T/$\overline{R}$ = 2.0V, VL = 2V	5	10	18	ns
t_{DHLC}		*(Figure 2)*	5	12	20	ns
t_{DLHT}	T/$\overline{R}$ to Bn	VCI = An, VC = 5V, *(Figure 5)*	5	15	25	ns
t_{DHLT}		CD = 0.8V, RC = 390Ω, CL = 30 pF RL1 = 18Ω, RL2 = NC, VL = 2V	5	22	35	ns
t_R	Driver Output Rise Time	CD = 0.8V, T/$\overline{R}$ = 2V, VL = 2V	3	6	10	ns
t_F	Driver Output Fall Time	*(Figure 2)*	3	6	10	ns
Receiver:						
t_{RLH}	Bn to An	CD = 0.8V, T/$\overline{R}$ = 0.8V	5	12	18	ns
t_{RHL}		*(Figure 3)*	5	10	18	ns
t_{RLZC}	CD to An	Bn = 2.0V, T/$\overline{R}$ = 0.8V, CL = 5 pF RL1 = 390Ω, RL2 = NC, VL = 5V *(Figure 4)*	5	10	18	ns
t_{RZLC}		Bn = 2.0V, T/$\overline{R}$ = 0.8V, CL = 30 pF RL1 = 390Ω, RL2 = 1.6k, VL = 5V *(Figure 4)*	5	8	15	ns
t_{RHZC}		Bn = 0.8V, T/$\overline{R}$ = 0.8V, VL = 0V, RL1 = 390Ω, RL2 = NC, CL = 5 pF *(Figure 4)*	2	4	8	ns
t_{RZHC}		Bn = 0.8V, T/$\overline{R}$ = 0.8V, VL = 0V, RL1 = NC, RL2 = 1.6k, CL = 30 pF *(Figure 4)*	3	7	12	ns
t_{RLZT}	T/$\overline{R}$ to An	VCI = Bn, VC = 2V, RC = 18Ω, CD = 0.8V, VL = 5V, RL1 = 390Ω, RL2 = NC, CL = 5 pF *(Figure 5)*	5	10	18	ns
t_{RZLT}		VCI = Bn, VC = 2V, RC = 18Ω, CD = 0.8V, VL = 5V, RL1 = 390Ω, RL2 = 1.6k, CL = 30 pF *(Figure 5)*	14	24	40	ns
t_{RHZT}		VCI = Bn, VC = 0V, RC = 18Ω, CD = 0.8V, VL = 0V, RL1 = 390Ω, RL2 = NC, CL = 5 pF *(Figure 5)*	2	4	8	ns
t_{RZHT}		VCI = Bn, VC = 0V, RC = 18Ω, CD = 0.8V, VL = 0V, RL1 = NC RL2 = 1.6k, CL = 30 pF *(Figure 5)*	2	8	15	ns
t_{NR}	Receiver Noise Rejection Pulse Width	*(Figure 6)*	3	6		ns

Note: NC means open

DS3897 Switching Characteristics

(0°C ≤ T_A ≤ 70°C, 4.75V ≤ V_{CC} ≤ 5.25V unless otherwise specified)

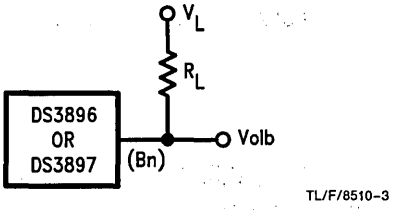
Symbol	Parameter	Conditions	Min	Typ	Max	Units
Driver:						
t_{DLH}	Dn, En to Bn	$\overline{TE}$ = 0.8V, $\overline{RE}$ = 2.0V, VL = 2V	5	9	15	ns
t_{DHL}		*(Figure 2)*	5	9	15	ns
t_{DLHT}	$\overline{TE}$ to Bn	An = $\overline{RE}$ = 2.0V, VL = 2V, *(Figure 2)*	5	10	18	ns
t_{DHLT}		RL1 = 18Ω, RL2 = NC, VL = 2V *(Figure 5)*	5	12	20	ns
t_R	Driver Output Rise Time	CD = 0.8V, T/$\overline{R}$ = 2V, VL = 2V	3	6	10	ns
tF	Driver Output Fall Time	*(Figure 2)*	3	6	10	ns

DS3897 Switching Characteristics (Continued)

(0°C ≤ T_A ≤ 70°C, 4.75V ≤ V_{CC} ≤ 5.25V unless otherwise specified)

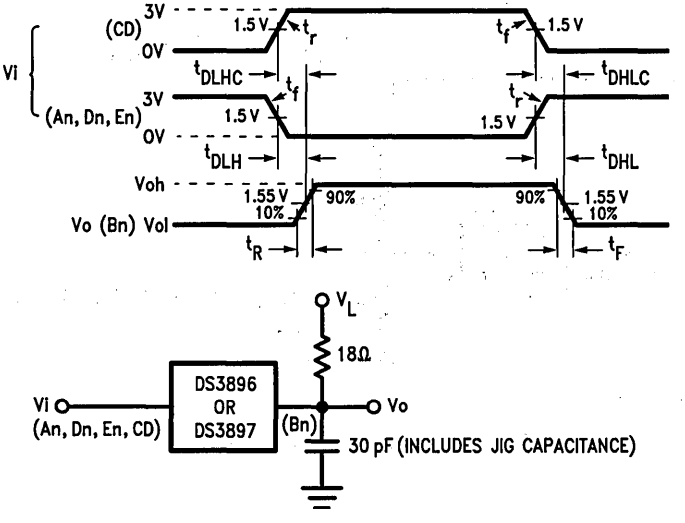
Symbol	Parameter	Conditions	Min	Typ	Max	Units
Receiver:						
t_{RLH}	Bn to Rn	$\overline{TE}$ = 2.0V, $\overline{RE}$ = 0.8V (*Figure 3*)	5	10	18	ns
t_{RHL}			5	12	18	ns
t_{RLZR}	$\overline{RE}$ to Rn	Bn = $\overline{TE}$ = 2V, VL = 5V, CL = 5 pF RL1 = 390Ω, RL2 = NC (*Figure 4*)	5	10	18	ns
t_{RZLR}		Bn = $\overline{TE}$ = 2V, VL = 5V, CL = 30 pF RL1 = 390Ω, RL2 = 1.6k (*Figure 4*)	5	8	15	ns
t_{RHZR}		Bn = 0.8V, $\overline{TE}$ = 2V, VL = 0V, RL1 = 390Ω, RL2 = NC, CL = 5 pF(*Figure 4*)	2	4	8	ns
t_{RZHR}		Bn = 0.8V, $\overline{TE}$ = 2V, VL = 0V, RL1 = NC, RL2 = 1.6k, CL = 30 pF(*Figure 4*)	3	7	12	ns
t_{NR}	Receiver Noise Rejection Pulse Width	(*Figure 6*)	3	6		ns
Driver plus Receiver:						
t_{DRLH}	Dn to Rn	$\overline{TE}$ = $\overline{RE}$ = 0.8V (*Figure 7*)	10	20	30	ns
t_{DRHL}			10	20	30	ns

Note: NC means open



TL/F/8510–3

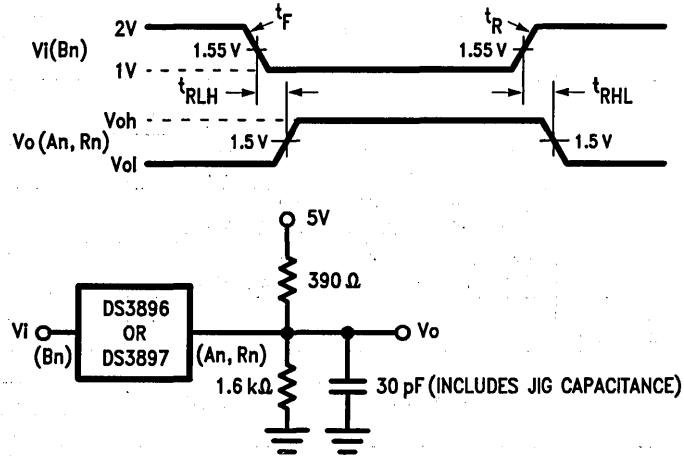
FIGURE 1. Driver Output Low Voltage Test



TL/F/8510–4

Note: $t_r = t_f \leq 5$ ns from 10% to 90%

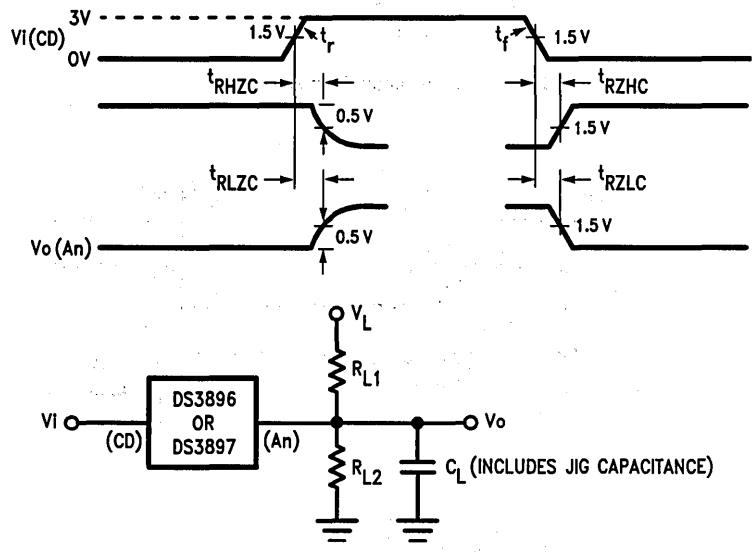
FIGURE 2. Driver Propagation Delays



TL/F/8510–5

Note: $t_R = t_F \leq 10$ ns from 10% to 90%

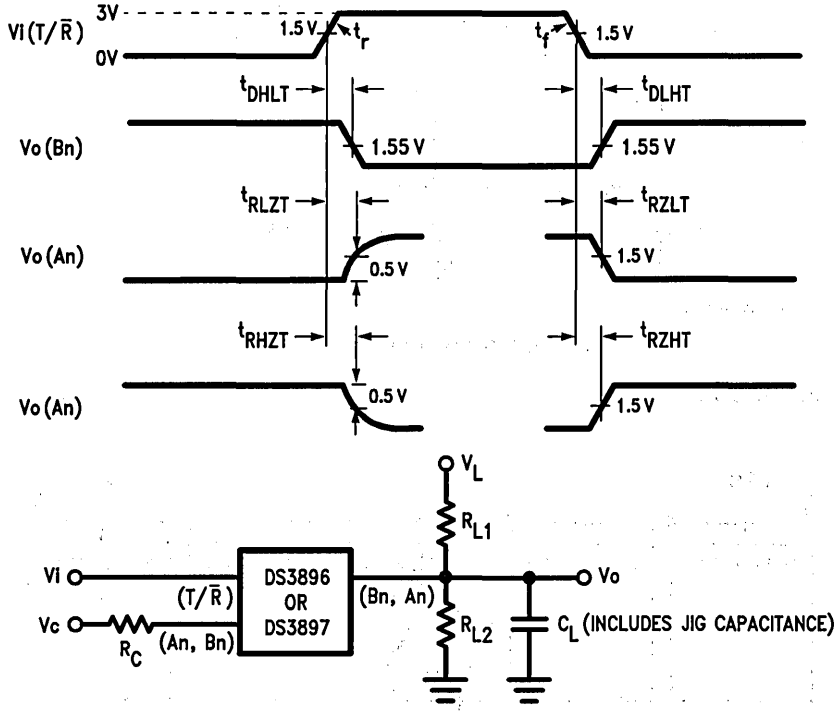
FIGURE 3. Receiver Propagation Delays



TL/F/8510–6

Note: $t_r = t_f \leq 5$ ns from 10% to 90%

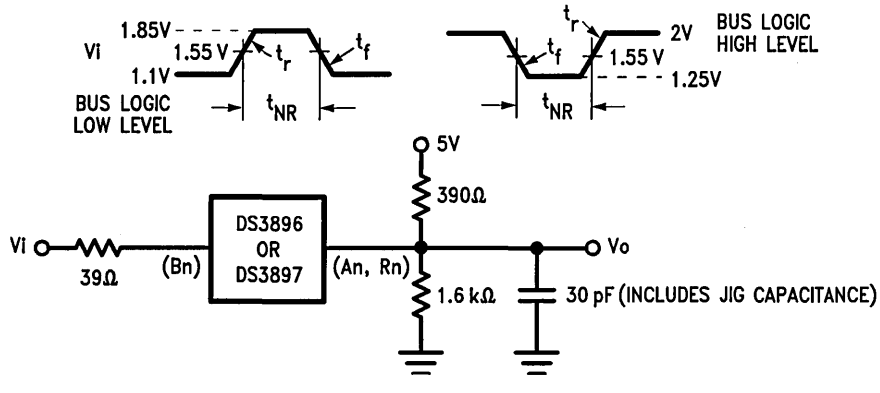
FIGURE 4. Propagation Delay from CD pin to An



TL/F/8510–7

Note: $t_r = t_f \leq 5$ ns from 10% to 90%

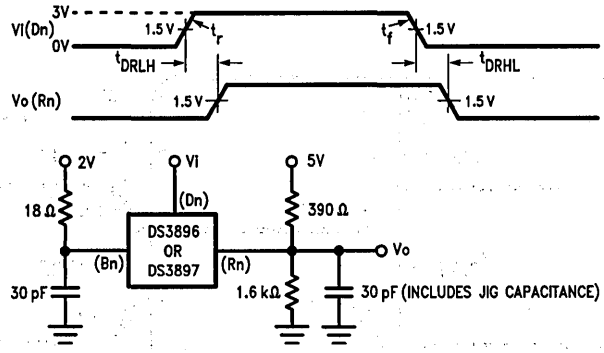
FIGURE 5. Propagation Delay from T/R̄ pin to An or Bn



TL/F/8510–8

Note: $t_r = t_f = 2$ ns from 10% to 90%

FIGURE 6. Receiver Noise Immunity: "No Response at Output" Input Waveforms

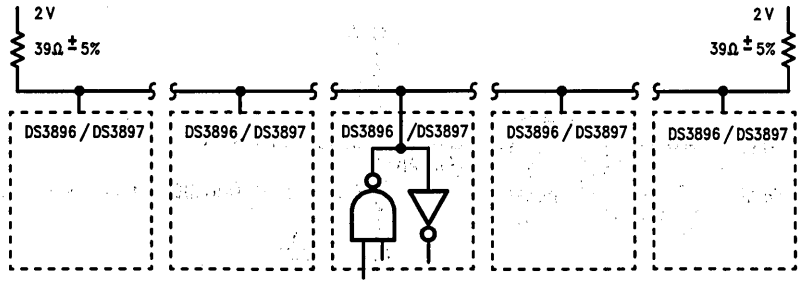


TL/F/8510–9

Note: $t_r = t_f \leq 5\ \mu s$ from 10% to 90%

FIGURE 7. Driver Plus Receiver Delays

Typical Application



TL/F/8510–10

Section 2

BTL Transceiver Application Notes

Section 2 Contents

National Semiconductor

BTL Introduction

Chris Koehle

Since 1985, BTL has grown from a new bus driving technology to the industry standard for driving high performance backplane buses. The evolution of BTL resulted from the need to replace TTL devices which are not suited to drive heavily loaded backplanes at very fast and reliable data rates. BTL transceivers are now required for the implementation of Futurebus+ (IEEE P896) designs. In order to see why BTL is the technology of choice, one must first look at the problems BTL was designed to solve.

The TTL Bus Driving Problem

BTL was invented by National Semiconductor Corporation and first introduced in 1985 to solve the bus driving problem created by deficiencies in TTL drivers. TTL devices are not suited to drive characteristically low impedance buses which result from a capacitive load in each backplane card slot in a running computer system. Typical TTL bus drivers have an output capacitance of 12 pF–20 pF per transceiver. This, when coupled with the capacitive loading resulting from the connectors, holes, vias and printed circuit traces results in a lumped slot capacitance of 20 pF–25 pF.

The high output capacitance of each board results in a low characteristic impedance for the backplane signal lines. In order to drive a signal through the threshold region, a high current drive is required. Most TTL devices specify a sink current of 64 mA–200 mA. This large current drive that is needed by these TTL devices results in the high transceiver output capacitance. In order to have incident wave switching of a signal with these parts, (i.e., do not have to wait for reflections to force signal transition through threshold), even more current drive than that which is specified in devices today is required to transition a line through TTL's 3V signal swing. The TTL bus driving problem arises when attempting incident wave switching. A TTL device must have higher drive current than presently available. Increasing this drive current also brings about an increased output capacitance for the device. This scenario increases the overall bus loading and once again lowers the characteristic impedance of the backplane which requires even more current drive *(Figure 1)*.

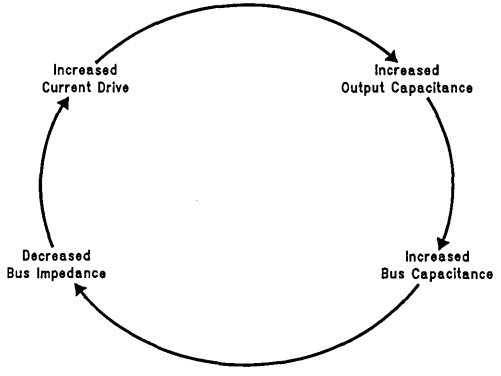
The result of this cyclic problem was a compromise that had to be made in performance. A bus using TTL driving logic must rely upon reflections at the bus terminations in order to send the signal completely through from one logical signal level to another. The performance hit is evident by looking at the specifications for VMEbus in relation to those for Futurebus+. VMEbus requires a 35 ns settling delay for reflections before a board can review the data or control lines that transitioned on the bus. The only delay incorporated in a Futurebus+ system is based upon the actual performance of the BTL drivers and receivers and time for a signal to propagate down a backplane.

The BTL Solution to the Bus Driving Problem

The endless cycle described above is the problem faced by system and transceiver designers in their quest to obtain the highest level of system performance that is physically possible.

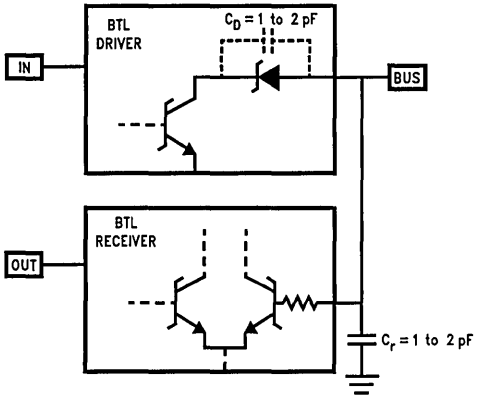
The unique BTL solution addresses the bus driving problem and was originally designed by National specifically for the Futurebus+ IEEE specification. Since 1985, BTL has been adopted by many proprietary backplane bus designers and has become the backplane driver of choice for high performance systems.

BTL incorporates a unique transceiver design which limits the amount of output capacitance for the transceiver *(Figure 2)*.



TL/F/11152–1

FIGURE 1. TTL Bus Driving Problem



TL/F/11152–2

FIGURE 2. The BTL Transceiver

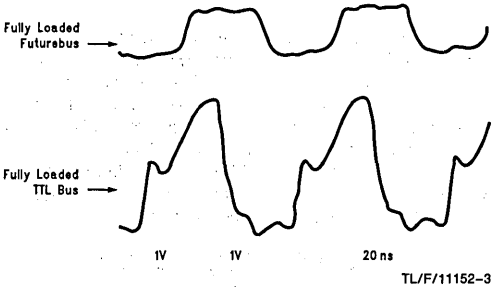
The BTL transceiver solves the output capacitance problem by inserting a Schottky diode in series with the driving transistor. This isolates the driver capacitance to that of the reverse biased Schottky diode, and the result becomes a driver capacitance of 1 pF–2 pF. Another 1 pF–2 pF for a BTL receiver puts the total BTL device capacitance at <5 pF as required by the IEEE P1194 specification. The 5 pF capacitance specification is very crucial for transceiver manufacturers to meet. Any excess capacitance taken up by a transceiver would result in system designers having to layout their boards with BTL transceiver to connector stub lengths that physically are not capable with any sized package.

This reduced transceiver output capacitance is only part of the BTL solution. BTL also specifies a 1V signal swing with tightly controlled receiver thresholds. The 1V swing with the BTL required 80 mA drive current, enables an incident wave switching to occur and does not require reflections to pass a signal through the threshold region. The 1V signal swing also enables reduced power consumption. The tight receiver thresholds are required in order to guarantee accurate noise margin which are critical since a signal only switches between 1V and 2.1V. As a result, any BTL transceiver must have a separate ground and V_{CC} pins which are only connected to separate receiver threshold control circuitry, otherwise noise margins can not be reliably controlled.

A BTL backplane bus is also terminated at both ends to 2.1V, with resistors selected to match the bus impedance. A TTL bus with a 3V signal swing, high current drivers and such a large output capacitance can not be equally matched which also results in a less reliable signal.

The end result of the BTL solution to the TTL bus driving problem can be seen in the actual backplane waveforms below *(Figure 3)*.

Slot Spacing 1.0" and Number of Slots = 18



TL/F/11152–3

FIGURE 3. Actual BTL vs TTL Waveforms

The clean, reliable waveforms with incident wave switching has caught the eye of many proprietary system designers, and now with Futurebus+ about to become a standard, BTL is undoubtedly the bus transceiver of choice for high performance systems.

IEEE 896 Futurebus+—A Solution to the Bus Driving Problem

National Semiconductor
Application Note 671
R. V. Balakrishnan/Joel Martinez/
Stephen Kempainen

The IEEE 896 Futurebus+ is a general-purpose bus standard for high-performance microcomputer systems. With a strong emphasis on speed and reliability, IEEE 896 offers a number of innovative features that are not found in other backplane buses.

A major contribution to its performance comes from its electrical specifications. Futurebus+ solves, for the first time, the fundamental problems associated with driving a densely populated backplane—as a result, it provides significant improvements in both speed and data integrity. Two years of effort by the IEEE 896 committee have culminated in a deeper understanding of the physics of the backplane bus, leading to an ingenious solution to the bus problem.

Speed is probably the most important feature of any bus standard. This is especially true for Futurebus+, since its totally asynchronous protocol permits continuous speed enhancements through advances in technology. In fact, the maximum data transfer rate between any two plug-in cards is determined simply by the sum of the response times of the two cards and the bus delay. Ultimately, as logic devices get faster, bus delay will be the dominating factor limiting bus speed.

There are two components to the bus delay in a typical system, namely, the settling time and the propagation delay. The settling time is the time needed for reflections and crosstalk to subside before data are sampled; it is usually several times longer than the backplane propagation delay. As will be shown later, the settling time is the price the user pays for not driving the bus properly.

By using a new technology, BTL = Backplane Transceiver Logic, Futurebus+ not only eliminates the settling time delay but also reduces the propagation delay of the loaded backplane to provide maximum possible bus throughput.

THE PHYSICS OF THE BACKPLANE BUS

For high-speed signals the bus acts like a transmission line with an associated characteristic impedance and propagation delay whose unloaded values, Z_o and t_{po}, are given by

$$Z_o = \sqrt{\frac{L_0}{C_0}}$$

$$t_{po} = \sqrt{L_0 C_0}$$

L = distributed inductance per unit length, and C = distributed intrinsic capacitance per unit length.(1)

These values can be calculated for a typical microstrip backplane *(Figure 1)* by means of the following equations:

$$Z_o = (87/\sqrt{\epsilon_r + 1.41}) \cdot \ln [5.98h/(0.8w + t)]\Omega$$

$$t_{po} = 1.017\sqrt{0.475\,\epsilon_r + 0.67}\ \text{ns/ft}$$

where ϵ_r = relative dielectric constant of the board material (typically $\epsilon_r = 4.7$ for fiberglass and w,h,t = the dimensions indicated in *Figure 1*. For a typical backplane, t = 1.4 mils, w = 25 mils, h = 1/16 inch, and $\epsilon_r = 4.7$. By substituting these values we get $Z_o = 100\Omega$ and $t_{po} = 1.7$ ns/ft.

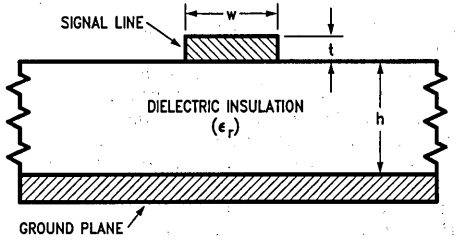
These values correspond to an *unloaded* backplane. When the backplane is uniformly loaded with the capacitance of plug-in cards and connectors at frequent intervals, the loaded values of the impedance, Z_L, and the propagation delay, t_{pL}, are given by

$$Z_L = Z_o/\sqrt{1 + (C_L/C_0)}$$

$$t_{pL} = t_{po}\sqrt{1 + (C_L/C_0)}$$

where C_L = the distributed load capacitance per unit length.(1)

The distributed capacitance, C_0 of the unloaded backplane can be measured in the lab. For our microstrip, it is 20 pF/ft. This does not include, however, the capacitance of the connectors mounted on the backplane and the associated plated-through holes, which can amount to 5 pF per card slot.



TL/F/10782–1

FIGURE 1. Cross Section of a Microstrip Bus Line

The loading capacitance of the plug-in card, however, is dominated by the loading capacitance of the transceiver, which can be 12–20 pF for TTL devices. Allowing another 3–5 pF for printed-circuit traces and the connector, the total loading per card slot can add up to 30 pF. For a system such as IEEE 896, which has 10 slots per foot, C_L = 300 pF/ft. Therefore,

$$Z_L = 100/\sqrt{1 + (300/20)} = 25\Omega$$

$$t_{pL} = 1.7\sqrt{1 + (300/20)} = 6.8\ \text{ns/ft}$$

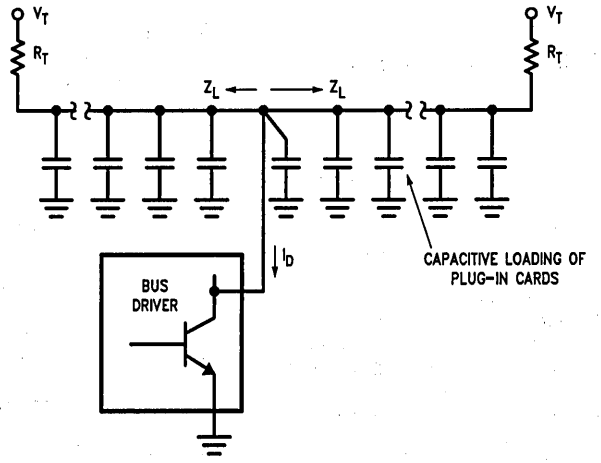
As can be seen above, the capacitive loading drastically alters both the impedance and the propagation delay of the bus. This reduces the bus throughput in two ways. One obvious impact is the increased propagation delay. But the not so obvious and even more serious problem is the reduced bus impedance, which is much harder to drive.

For example, to drive the loaded bus properly with a TTL driver which has a 3V nominal swing, the required drive current, I_D, must be

$$I_D = 3V/(Z_L/2)$$

The impedance seen by the driver is half of Z_L, since from a given board two transmission lines are being driven, one towards each terminator *(Figure 2)*. Therefore,

$$I_D = 3/(25/2) = 240\ \text{mA}$$



TL/F/10782–2

FIGURE 2. The Loaded Bus—Each Driver Sees Two Loaded Line Impedances In Parallel ($Z_L \| Z_L = Z_{L/2}$).

This is much higher than the standard TTL's drive capability of 50 to 100 mA. *Figure 3* shows the effect of using a 50 mA driver, in this situation, on the bus waveform. The voltage swing on the bus has its first transition at 0.5V, the product of the drive current and $Z_L/2$. This value falls well below the upper threshold limit of the TTL receiver. Therefore, *several round-trip delays to the nearest termination are required for the waveform to cross the receiver threshold region.* In our example, one round-trip delay is $2t_{pL}$ = 13.6 ns/ft. Therefore the settling times can exceed 100 ns even for relatively short buses. This long settling time drastically affects bus throughput at high speeds. Even worse, the voltage steps in the threshold region can cause multiple triggering in the cases of the clock and strobe signals.

One way to solve these problems is to use 100 mA drivers with precision receivers that have a narrow threshold region such that the first transition crosses well over the threshold. This technique is widely used for clock lines to avoid multiple triggering. Its use on data/address lines is limited because of the significantly higher power requirement arising from the large number of lines involved (32 address/data lines).

Even if power is not a limitation, switching to higher current drivers provides only a marginal improvement. The reason for this is quite simple. A higher current driver unfortunately has a higher output capacitance, which reduces the bus impedance further. This in turn requires an even higher current drive for proper operation.

The Futurebus+ Transceiver

A more elegant solution—one that is now a part of IEEE 896—directly attacks the root of the problem, namely, the large output capacitance of the transceiver. By simply adding a Schottky diode in series with an open-collector driver output, the capacitance of the drive transistor is isolated by the small reverse-biased capacitance of the diode in the non-transmitting state *(Figure 4)*. The Schottky diode capacitance is typically less than 2 pF and is relatively independent of the drive current. Allowing for a receiver input capacitance of another 2 pF, the total loading of the Futurebus+ transceiver can be kept under 5 pF. IEEE 896 specifies the maximum transceiver capacitance at 5 pF.

In addition to reducing the loading on the bus, the Futurebus+ transceiver features several other enhancements over a conventional TTL transceiver that drastically reduce power consumption and improve system reliability.

A major portion of the power savings comes from a reduced voltage swing—1V—on the bus. Contrary to popular belief, the lower swing does not reduce crosstalk immunity (provided the receiver threshold is tightly controlled).[2] The induced crosstalk from other lines on the bus scales down with the amplitude of the signal transistion causing it. Consequently, if a line receiver has a precision threshold, the noise margin, expressed as a percentage of signal amplitude, remains the same, as does the crosstalk immunity. However, the absolute noise margin, with reference to a noise source external to the bus, does shrink linearly with amplitude. Fortunately, the low impedance and the relatively short length of the bus make this externally generated noise component insignificant in high-speed backplanes. Nevertheless, it is recommended that the backplane be shielded from strong noise sources external to the bus.

Noise Immunity and EMI

The Futurebus+ transceiver has a precision receiver threshold centered between the low and high bus levels of 1V and 2.1V, respectively *(Figure 5)*. Confined to a narrow region of 1.55V ±75 mV (1.47V to 1.62V), the threshold voltage is independent of V_{CC} and temperature. This tight threshold control is achieved by using an internal bandgap reference at the receiver input *(Figure 4)*. And with the smaller 1V swing, EMI is also reduced threefold compared with TTL.

DRIVE CURRENT

The backplane impedance in IEEE 896 is specified as 52Ω minimum and 62Ω maximum with the connectors mounted. In our microstrip example, due to the connector and the plated-through holes, a 52Ω minimum impedance translates into a maximum allowable capacitance of 5 pF per slot. This can be easily attained with some care in printed-circuit board design. A fully loaded Futurebus+ backplane therefore, has an impedance whose worst-case value is given by

$$Z_{min} = \frac{100}{\sqrt{1 + \frac{150}{20}}}\ \Omega$$

$$= 34\Omega$$

The drive current required for a 1V swing is

$$I_D = 1/(34/2) = 58\ mA$$

However, with a precision receiver threshold it is possible for the driver to swing past the threshold with a comfortable margin even if the first step climbs to only 75 percent of the final amplitude under worst-case loading (see again *Figure 5*). Therefore, the drive current can be reduced by 25 percent to save power, without affecting performance:

$$I_D = \frac{1}{34/2}\,0.75 = 45\ mA$$

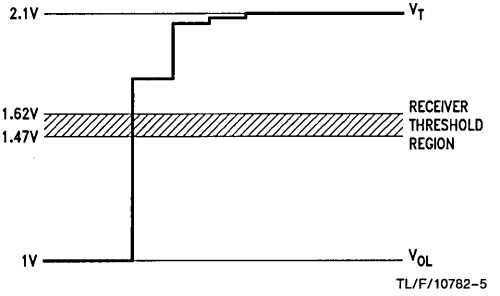


FIGURE 5. IEEE 896 Signaling Levels and the Worst-Case Bus Waveform

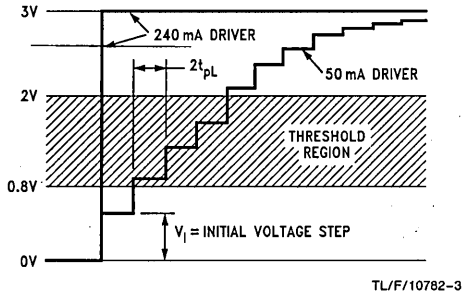


FIGURE 3. TTL Bus Waveforms—50 mA Driver vs 300 mA Driver

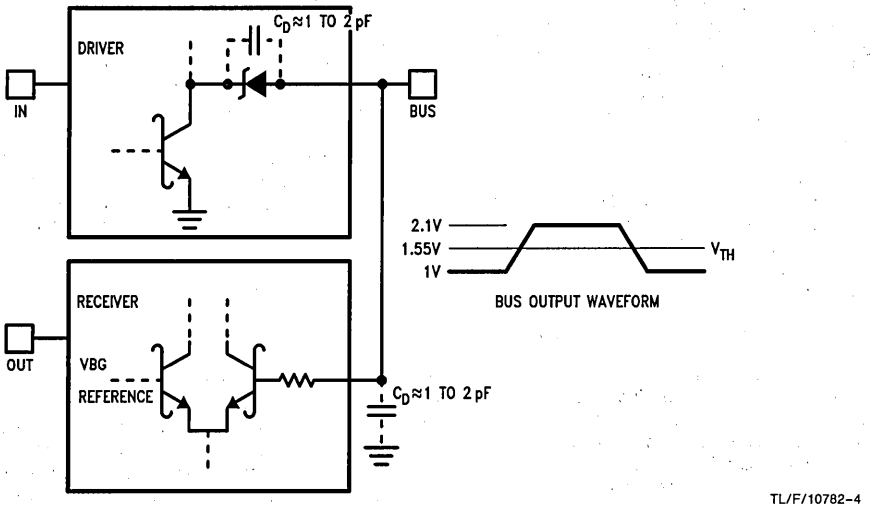
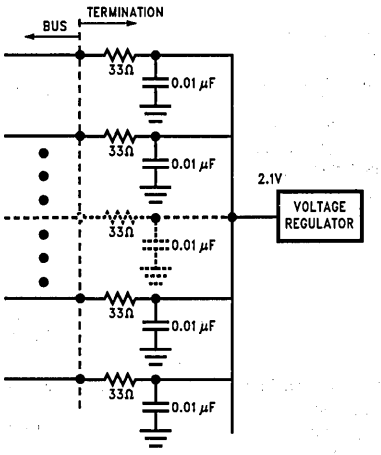


FIGURE 4. The Futurebus+ Transceiver



TL/F/10782–6

FIGURE 6. Equivalent Futurebus+ Termination Circuit

OTHER HIGHLIGHTS

Bus Propagation Delay

There is an additional benefit resulting from reducing the capacitive loading on the bus. This benefit arises from the reduced propagation delay, which further improves the bus speed.

Recalculating the loaded propagation delay for the Futurebus+ transceiver yields

$$t_{PL} = t_{po}\sqrt{1 + (C_L/C_0)}$$
$$= 1.7\sqrt{1 + \frac{150}{20}}$$
$$= 4.9 \text{ ns/ft}$$

This is a 30-percent improvement over the TTL example. It should be noted that this is the worst-case delay per foot and that the asynchronous nature of the Futurebus+ protocol will take full advantage of lower propagation delays in a typical system, either due to lower loading levels or due to the closer spacing of two plug-in boards that are in communication.

Termination

The drive current and the signal swing determine the termination resistors. If the drive current is derived properly, the termination will match the bus impedance under the given loading. For IEEE 896, the value of each of the two termination resistors, R_T, is

$$R_T = \left(\frac{1V}{58 \text{ mA}}\right) 2 = 34\Omega \approx 33\Omega$$

This value is less than the loaded impedance of the Futurebus+, because simulations by the Futurebus+ Electrical Task Group show it is the best value. In a practical bus, the impedance varies with the loading conditions and the above termination is a compromise. Simulations show that the best noise margins can be maintained by using the 33Ω terminations.

The P896 draft requires that the bus be terminated at both ends, with a single resistor of 33Ω connected to an active voltage source of 2.1V *(Figure 6)*. This arrangement has a significantly lower power dissipation than a "Thévenin-equivalent" two-resistor termination connected to ground and the 5V rail. *Figure 6* shows an equivalent circuit that can be used for terminating a few of the bus lines. Since each bus line has the potential of sinking 80 mA, the termination voltage must be able to supply adequate current for the worst case situation of all lines asserted simultaneously. The inherent inductance and decoupling capacitors of the termination voltage supply are crucial to the performance of the system. The source can be shared among bus lines as long as it is properly bypassed for alternating current close to each resistor.

Wire-OR Glitch

One of the advantages of an open-collector bus is a wire-OR capability. This feature is fully exploited in the IEEE 896 bus, particularly in its sophisticated arbitration protocol and broadcast mechanism. Unfortunately, due to the fundamental nature of transmission lines, wire-ORing on the bus can cause erroneous glitches having pulse widths of up to the round-trip delay of the bus. The analysis of the wire-OR glitch is covered well by Theus and Gustavson.[5]

To overcome the wire-OR glitch, the broadcast acknowledge lines (AI* and DI*) and the three arbitration control lines are required to have integrators at the output of the receiver capable of rejecting pulses having widths of up to the maximum round-trip delay of the bus.

And More

Geographic addressing and live insertion and withdrawal capability are some of the other highlights of Futurebus+.

The electrical specification of Futurebus+ is based on a thorough knowledge of backplane operation. A combination of theoretical analysis and bench measurements has been used to create an electrically clean bus environment. Significant improvements have been made in favor of higher performance—at the expense of only a slight increase in today's cost and complexity—to assure a long design lifetime for the standard. The result is a proposed standard that has the performance, in terms of both speed and reliability, to justify the name, "Futurebus+".

ACKNOWLEDGEMENT

I would like to thank Paul Borrill for his help and encouragement in finding this solution to the bus driving problem.

REFERENCES

1. *Motorola MECL System Design Handbook*, Motorola, Inc., Phoenix, AZ.
2. R. V. Balakrishnan, "Reducing Noise on Microcomputer Buses", *IEOCN 82—Professional Workshop Record on Microcomputer Realities,* Nov. 1982.
3. R. V. Balakrishnan, "Cut Bus Reflections, Crosstalk with a Trapezoidal Transceiver", *EDN*, Aug. 4, 1983, pp. 151–156.
4. R. V. Balakrishnan, "Eliminating Crosstalk Over Long-Distance Busing". *Computer Design*, Mar. 1982, pp. 155–162.
5. John Theus and David B. Gustavson, "Wire-OR Logic on Transmission Lines", *IEEE Micro*, Vol. 3, No. 3, June 1983, pp. 51–55.
6. Interim Report of the Electrical Task Group Expert Team, "Futurebus+ Modeling and Noise Margin Results", Raytheon and DEC, September, 14, 1990.

Reducing Noise on Microcomputer Buses

National Semiconductor
Application Note 337
R. V. Balakrishnan

Abstract: *This paper focuses on the noise components that have a significant impact on the performance of a high speed microcomputer bus. An overview of their nature is followed by ways to minimize their contribution by suitable design of the PC board backplane, the termination network and the bus transceiver. The DS3662 trapezoidal bus transceiver, which is specifically designed to minimize such noise on high speed buses, is presented along with its performance data. And to conclude, some possible new transceiver designs for further improvement of the bus performance are explored.*

INTRODUCTION

As the microcomputer bus bandwidth is extended to handle ever increasing clock rates, the noise susceptibility of a single-ended bus poses a serious threat to the overall system integrity. Thus, it is mandatory that the various noise contributions be taken into account in the design of the bus transceiver, the PC board backplane and the bus terminations to avoid intermittent or total failure of the system.

Although noise such as crosstalk and reflections are inevitable in any practical bus configuration, their impact on the system can be determined and minimized by careful design of all three components mentioned above. The combined contribution of the noise under worst-case conditions should be within the noise margin for reliable bus operation.

The design of the transceiver plays a significant role in minimizing crosstalk and reflection. The bus can be optimized for minimum noise at a given bandwidth by using a trapezoidal driver having suitable rise and fall times along with a matched low pass filtered receiver which provides a symmetrical noise margin. The DS3662 is one such transceiver, the first member in the family of trapezoidal bus transceviers available from National Semiconductor Corporation. This device represents a significant improvement in high speed bus circuit design and provides a solution to commonly encountered bus noise problems.

THE MICROCOMPUTER BUS

A typical microcomputer bus usually consists of a printed circuit board backplane with signal and ground traces on one side and a ground plane on the other. The length ranges from a few inches to several feet with as many as 32 closely spaced (0.6" typical) card edge connectors. Each signal line interacts with the ground plane to form a transmission line with characteristic impedance 'Z' in the range of 90Ω–120Ω typical. It is desirable to have as large a 'Z' as possible in order to reduce the drive requirement of the bus driver and to reduce the power dissipated at the terminations. But much larger values of 'Z" translate to significantly larger physical dimensions and therefore are not very practical.

The bus appears like a transmission line to any signal having a transition time 't_r' less than the round trip delay '$2T_L$' of the bus. The bus delay 'T_L' is given by:

$$T_L = L\sqrt{L1\ C1} \quad (1)$$

where L = length of the bus
$L1$ = distributed inductance per unit length
$C1$ = distributed capacitance per unit length

For a typical unloaded 100Ω microstrip line, $C1 \cong 20$ pF/ft and $L1 \cong 0.2\ \mu$H/ft. Therefore, $T_L = 2.0$ ns/ft. This corresponds to approximately half the speed of light. However, the capacitive loading at each connector on the backplane increases the delay time significantly. The loaded delay time 'T_{LL}' is given by:

$$T_{LL} = T_L\sqrt{1 + (C_L/C_1)} \quad (2)$$

where C_L = distributed load capacitance/unit length

Given a 10 pF loading at each connector (connector + transceiver capacitance) and a 0.6" spacing between connectors, $C_L = 200$ pF/ft and $T_{LL} = 6.6$ ns/ft. So even a 6" long bus has a $2T_{LL} = 6.6$ ns, which is higher than the transition time (t_r) of many high speed bus drivers. When in doubt, it is always better to use the transmission line approach than the lumped circuit approach as the latter is an approximation of the former. Also, the transmission line analysis gives more pessimistic (worst-case) values of crosstalk and reflection and is, hence, safer.

CROSSTALK REDUCTION

The crosstalk is due to the distributed capacitive coupling C_C and the distributed inductive coupling L_C between two lines. When crosstalk is measured on an undriven sense line next to a driven line (both terminated at their characteristic impedances), the near end crosstalk and the far end crosstalk have quite distinct features, as shown in *Figure 1.* Their respective peak amplitudes are:

$$V_{NE} = K_{NE}(2T_L)(V_I/t_r) \quad \text{for } t_r > 2T_L \quad (3)$$

$$V_{NE} = K_{NE}(V_I) \quad \text{for } t_r < 2T_L \quad (4)$$

$$V_{FE} = K_{FE}(L)(V_I/t_r) \quad (5)$$

where V_I = signal swing on the drive line.

The coupling constants are given by the expressions:

$$K_{NE} = \frac{L(C_C Z + L_C/Z)}{4T_L} \quad (6)$$

$$K_{FE} = \frac{C_C Z - L_C/Z}{2} \text{ ns/ft} \quad (7)$$

The near end component reduces to zero at the far end and vice versa. At any point in between, the crosstalk is a fractional sum of near and far end crosstalk waveforms shown.

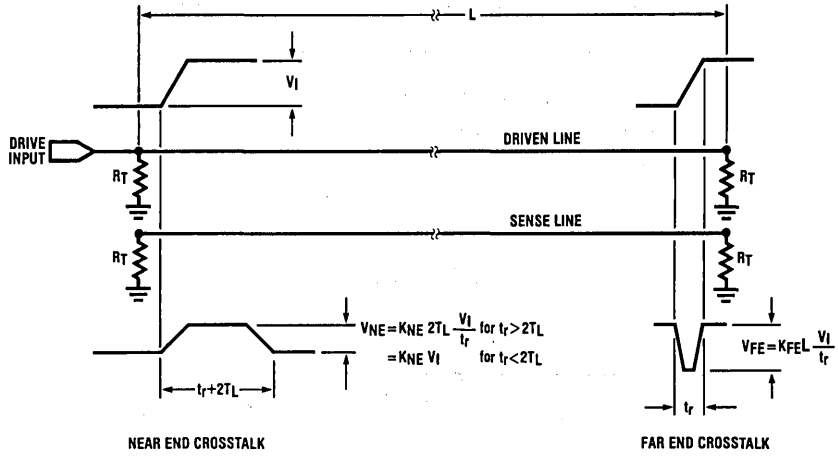
It should be noted from expressions 6 and 7 that the far end crosstalk can have either polarity whereas the near end crosstalk always has the same polarity as the signal causing it. In microstrip backplanes the far end crosstalk pulse is usually the opposite polarity of the original signal.

Although the real world bus is far from the ideal situation depicted in *Figure 1*, several useful observations that apply to a general case can be made:

1. The crosstalk always scales with the signal amplitude.
2. Absolute crosstalk amplitude is proportional to slew rate V_I/t_r, not just $1/t_r$.
3. Far end crosstalk width is always t_r.
4. For $t_r < 2T_L$, the near end crosstalk amplitude V_{NE} expressed as a fraction of signal amplitude V_I is a function of physical layout only.
5. The higher the value of 't_r' the lower the percentage of crosstalk (relative to signal amplitude).

The corresponding design implications are:

1. The noise margin expressed as a percentage of the signal swing is what's important, not the absolute noise margin. Therefore, to improve noise immunity, the percentage noise margin has to be maximized. This is achieved by reducing the receiver threshold uncertainty region and by centering the threshold between the high and low levels.
2. Smaller signal amplitude with the same transition time reduces bus drive requirements without reducing noise immunity.
3. Far end crosstalk is eliminated if the receiver is designed to reject pulses having pulse widths less than or equal to t_r.
4. When $t_r < 2T_L$, the near end crosstalk immunity for a given percentage noise margin has to be built into the backplane PC layout. Since $(V_{NE}/V_I) = K_{NE}$ for this case, K_{NE} should be kept lower than the available worst-case noise margin. K_{NE} may be reduced by either increasing the spacing between lines or by introducing a ground line in between. The ground line, in addition to increasing the spacing between the signal lines, forces the electric field lines to converge on it, significantly reducing crosstalk.
5. For minimum crosstalk the rise and fall times of the signal waveform should be as large as possible consistent with the minimum pulse width requirements of the bus. A driver that automatically limits the slew rate of the transition can go a long way in reducing crosstalk.



TL/F/5281–1

FIGURE 1. Crosstalk under Ideal Conditions

2

CROSSTALK MEASUREMENT

When multiple lines on either side of the sense lines switch simultaneously the crosstalk is considerably larger, typically 3.5 times the single line switching case for microstrip backplanes. Also, the location of the drivers on the driven lines and the receiver on the sense line for worst-case crosstalk differs for the near end and far end cases as shown in *Figure 2* and *3* for a uniformly loaded bus. But if the far end crosstalk is not of the opposite polarity, then the combined effect of far end and near end crosstalk could have a larger amplitude and pulse width at a point near the middle of the sense line in *Figure 2*. So in a general case, or in the case of a non-uniformly loaded bus, it is advisable to check the sense line at several locations along the length of the bus to determine the worst-case crosstalk. The measurement should be made for both the positive and the negative transition of the drive signal.

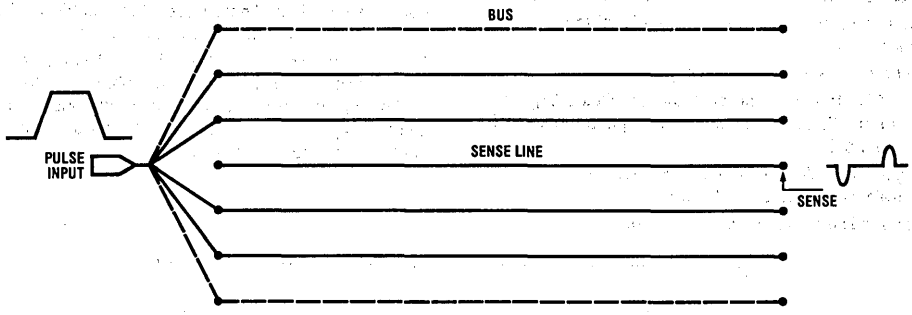
THE TERMINATION

A properly terminated transmission line has no reflections. But a practical microcomputer bus is neither a perfect transmission line nor is it properly terminated under all conditions. The capacitive loading at discrete locations, such as a used card slot, act as sources of reflection. However, in the limiting case when the bus is uniformly populated with a large number of modules, the bus behaves like a lower impedance transmission line. The loaded impedance 'Z_L' of the bus is given by the expression:

$$Z_L = \frac{Z}{\sqrt{1 + C_L/C_1}} \quad (8)$$

where Z = unloaded line impedance

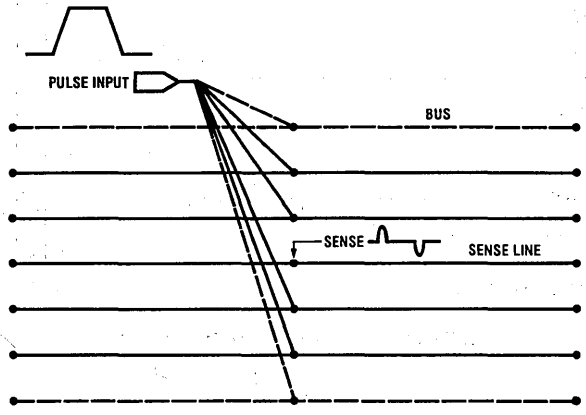
Unfortunately, uniform loading of the bus is not guaranteed at all times and even if it were (by dummy loading of



TL/F/5281–2

Note: All lines terminated at both ends (not shown)

FIGURE 2. Worst-Case Far End Crosstalk Measurement



TL/F/5281–3

Note: All lines terminated at both ends (not shown)

FIGURE 3. Worst-Case Near End Crosstalk Measurement

the unused slots) Z_L is usually too low for proper termination of the bus. For example, a 10 pF per module loading of the 100Ω microstrip bus at 0.6" spacing results in a $Z_L = 30\Omega$. One such termination at each end will require a 200 mA drive capacity on the bus driver for a nominal 3V swing. Such large drive currents and low value terminations increase the power dissipation of the system significantly in addition to causing other problems such as increased ground drop, inductive drops in traces due to large current being switched, etc. As a compromise the bus is usually terminated at an impedance higher than Z_L but less than or equal to Z. Consequently, there is always some amount of reflection present. For a perfect transmission line the reflection coefficient 'Γ' is given by the well known expression:

$$\Gamma = \frac{Z - R_t}{Z + R_t} \tag{9}$$

where Z = impedance of the bus

R_t = termination resistance

The net effect, in the general case of a nonuniformly loaded bus, is that it may take several round trip bus delays after a bus driver output transition, before the quiescent voltage level is established. However, this delay is avoided by using a bus driver that has sufficient drive to generate a large enough voltage step during the first transition to cross well beyond the receiver threshold region under the worst-case load conditions.

Figure 4 illustrates the driver output waveform under such a condition. Here the fully loaded bus (with $Z_L = 30\Omega$), of the previous example, is driven by the DS3662 bus transceiver at the mid point. The driver is actually driving two transmission lines of $Z_L = 30\Omega$ in either direction from the middle and hence the initial step is given by:

$$V1 = \left(\frac{Z_L}{2}\right) 2I_S \tag{10}$$

where I_S = Standing current on the bus due to each termination

For the DS3662, the termination can be designed for $2I_S$ = 100 mA and therefore:

V1 = (30/2)100 = 1.5V

This value of the initial swing is large enough to cross the narrow threshold region of the receiver as shown and therefore no waiting period is required for the reflections to build up the output high level. On the negative transition the problem is less critical due to the much higher sink capability of the DS3662 during pull down.

Reflections can also be caused by resistive loading of the bus by the DC input current of the receiver. The resulting reflectoin coefficient (Γ) is given by the expression:

$$\Gamma = -\tfrac{1}{2}\left(\frac{I_R}{I_S}\right) \tag{11}$$

where I_R = receiver input current

Having a receiver with a high input impedance not only makes this component of reflection insignificant but also reduces the DC load on the driver, allowing the use of lower value termination resistors. This is particularly true when a large number of modules are connected to the bus.

The design implications of the above discussion may be summarized as follows:

1. If the driver has adequate drive to produce the necessary voltage swing under the worst-case loading ($Z_L/2$), reflections do not restrict the bus performance. This translates to a 100 mA minimum drive requirement for a typical microstrip bus.

2. If the drive is insufficient, time should be allowed for the reflections to build up the voltage level before the data is sampled.

3. For signals such as clock, strobe, etc., wherein the edge is used for triggering events, it is mandatory that the driver meet the above drive requirements if delayed or multiple triggering is to be avoided.

4. An ideal TTL bus transceiver should have at least a 100 mA drive, a high input impedance receiver with a narrow threshold uncertainty region.

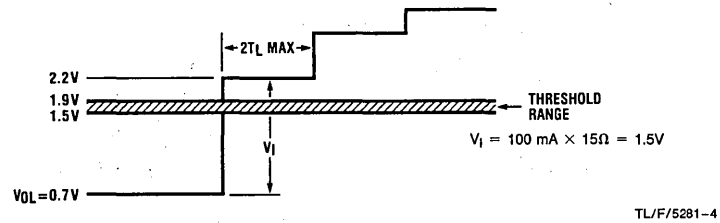


FIGURE 4. Worst-Case DS3662 Output Transition for $Z_L = 15\Omega$ and $R_T = 50\Omega$

THE DS3662 TRANSCEIVER

The DS3662 quad trapezoidal bus transceiver has been designed specifically to minimize the noise problems discussed previously. The driver generates precise trapezoidal waveforms that reduce crosstalk and the receiver uses a low pass filter to reject noise pulses having pulse widths up to the maximum driver output transition times. Precision output circuitry optimizes noise immunity without sacrificing the high data rate capability of Schottky transceivers.

Figure 5 shows the recommended configuration for microcomputer buses. The use of a 3.4V source with a single termination resistor at each end reduces the average power dissipation of the bus. However, a two resistor termination connected between the line and the power rails, having the same Thevenin's equivalent, can be substituted for lower cost.

Using a Miller integrator circuit, the driver generates a linearly rising and falling waveform with a constant slew rate of 0.2 V/ns *(Figure 6)*. This corresponds to a nominal transition time of 15 ns. *Figure 7* compares the output waveform of a typical high speed driver to that of DS3662 under different load conditions. It should be noted that even under heavy loading, the regular drivers have peak slew rates that are much higher than the average. On the other hand, the trapezoidal waveform has a much lower slew rate with only a slight increase in the transition time. Such an increase in the transition time has little or no effect on the data rates. In fact, the high fidelity of the DS3662 driver output waveform allows pulse widths as low as 20 ns to be transmitted on the bus.

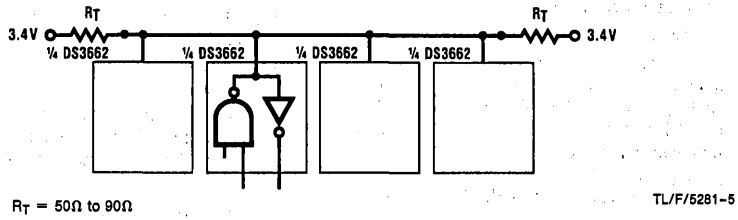


FIGURE 5. Recommended Bus Termination for Heavily Loaded Microstrip Backplanes

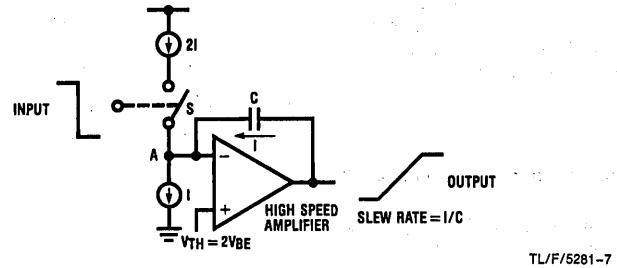
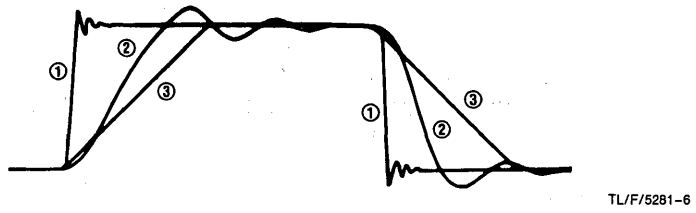


FIGURE 6. DS3662 Driver



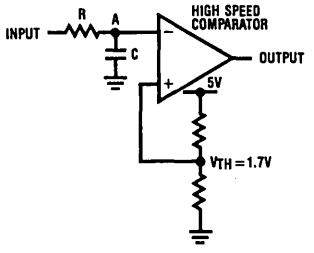
Note 1: Typical high speed driver output unloaded; $t_r = t_f \approx 3$ ns

Note 2: Typical high speed driver output loaded; $t_r = t_f \approx 10$ ns

Note 3: Typical outupt of controlled slew rate driver which is load independent; $t_r = t_f \approx 15$ ns

FIGURE 7. Waveform Comparison

The receiver consists of a low pass filter followed by a high speed comparator, with a typical threshold of 1.7V *(Figure 8)*. The noise immunity of the receiver is specified in terms of the width of a 2.5V pulse that is guaranteed to be rejected by the receiver. The receiver typically rejects a 20 ns pulse going positive from the ground level or going negative from the 3.4V logic 1 level. The receiver threshold lies within a specified 400 mV region over the supply and temperature range and is centered between the low and high levels of the bus for a symmetrical noise margin.



TL/F/5281–8

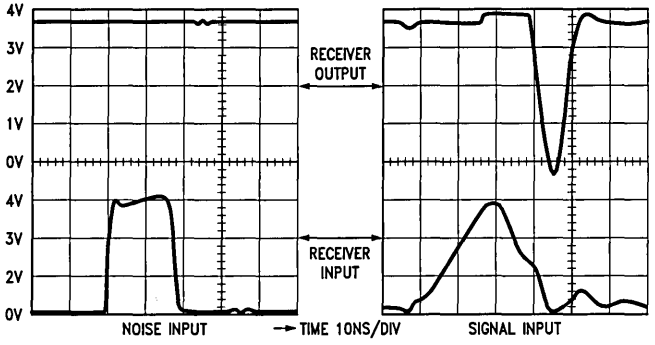
FIGURE 8. DS3662 Receiver

Other features of the device include a 100 μA maximum DC bus loading specification under power ON or OFF condition and a glitch-free power up/down protection on the bus output.

Waveforms in *Figure 9* demonstrate the ability of the receiver to distinguish the trapezoidal signal from noise. Here the receiver rejects a noise pulse of 19 ns width, while accepting a narrower signal pulse (16 ns) of the same peak amplitude (the signal is triangular because of the pulse width which is smaller than the transition time).

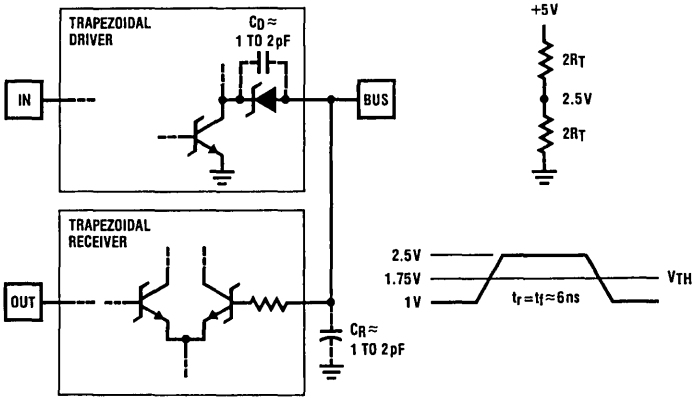
The real-world performance of the DS3662 transceiver shows an order of magnitude improvement in noise immunity over conventional transceivers under actual operating conditions (Reference #3). The controlled rise and fall times on the driver output significantly reduces both near end and the far end crosstalk. As expected, the pulse discrimination at the receiver input virtually eliminates the far end crosstalk, even on extremely long buses (over 100 feet). The near end crosstalk, which is particularly severe on the state of the art backplanes due to the tight spacing between the signal lines, is easily accommodated by the large percentage noise margin (>75%) provided by the receiver.

Field reports indicate that the DS3662 not only solves those mysterious intermittent failure problems in mini and microcomputer systems, but also helps them meet the new FCC emission requirements due to the reduced RF radiation from the bus.



TL/F/5281–12

FIGURE 9. DS3662 Receiver Response



TL/F/5281–21

FIGURE 10. High Speed Bus Transceiver with Low Output Loading for MicroComputer Backplanes

WHAT NEXT?

Since crosstalk scales with the signal amplitude, reducing the signal swing has not effect on the noise immunity as long as the percentage noise margin remains the same. On the other hand, there are several advantages in having lower signal swing. It reduces the drive current requirement of the driver thus reducing its output capacitance. Lower capacitive loading on the bus decreases its impedance reducing the drvie requirement even further. Having a lower current drive not only reduces the power dissipated at the terminations but also allows better matching of the termination due to the increased line impedance. In the ideal limiting case the driver has negligible loading effect on the bus and thus allows perfect termination under all load conditions.

In practice however, there are some obvious limitations. The receiver thresholds have to be maintained within tighter limits at lower signal swings to maintain the same percentage noise margin. Also, the capacitive loading is difficult to reduce beyond a certain point, due to the diminishing return in the way of lower current rating, as the loaded bus impedance approaches the unloaded impedance. However, the capacitance of an open collector driver output can be reduced significantly by using a Schottky diode as shown in *Figure 10.* The diode isolates the driver capacitance when the output is disabled. Using reduced signal swings and precise receiver thresholds, such a transceiver can provide significant improvements in microcomputer bus performance. The transceiver design presented in *Figure 10* is being considered for incorporation into the Futurebus standard by the IEEE.

CONCLUSION

A well designed bus transceiver goes a long way in improving the noise immunity of a single-ended TTL bus. Further improvements in bus performance may come from the use of reduced voltage swings and better transceiver designs for lower bus loading and tighter receiver threshold limits. Although such approaches may not be TTL compatible, the improvement in performance gained may indeed justify a new standard for bus transceivers.

REFERENCES

1) Bill Fowler, *"Transmission Line Characteristics,"* National Semiconductor—Application Note 108, May 1974.

2) A. Feller, H. P. Kaupp, and J. J. Digiacomo, *"Crosstalk And Reflections In High Speed Digital Systems,"* proceedings—Fall Joint Computer Conference, pp. 511–525, 1965

3) R. V. Balakrishnan, *"Bus Optimizer,"* National Semiconductor—Application Note 259, April 1981

4) David Montgomery, *"Borrowing RF Techniques For Digital Design,"* Computer Design, pp. 207–217, May 1982

5) R. V. Balakrishnan, *"Eliminating Crosstalk Over Long Distance Busing,"* Computer Design, pp. 155–162, March 1982

Timing Analysis of Synchronous and Asynchronous Buses

National Semiconductor
Application Note 514
David Hawley and R.V. Balakrishnan

ABSTRACT

This paper presents detailed examples of bus timing calculations for both synchronous and asynchronous busses, showing that bus throughput can be maximized by taking into account the characteristics and limitations of the transceiver technology being used. Based on these examples, a performance analysis of the currently available high speed bus interface technologies is made in terms of their maximum attainable transfer rate on both types of backplane busses. The results show that the use of a faster transceiver, as judged by its data sheet, doesn't necessarily result in a faster bus.

INTRODUCTION

In order to derive the highest possible throughput from a backplane bus, a careful analysis and optimization of timing parameters is essential. The maximum speed attainable at the physical level of the bus is a function of the transceiver technology, the electrical length of the bus, and the type of protocol, synchronous or asynchronous, being used. A clear understanding of the bus timing constraints lets the designer take best advantage of a given technology, such as TTL, ECL, or BTL (Backplane Transceiver Logic). Contrary to intuitive thinking, a faster transceiver will not always result in a faster bus. It can be shown through examples that greater bus transfer rates can be obtained by using specially designed bus transceivers, such as the BTL Trapezoidal, that at first glance may appear to be slower than the equivalent AS or FAST devices. These devices, in addition to improving bus bandwidth, also reduce crosstalk, ground noise, and system power requirements.

BUS PROPAGATION DELAY AND SETTLING TIME

Traditionally, system designers have used standard TTL devices to drive the backplane bus. Unfortunately, although TTL appears to provide fast rise and fall times, it cannot cleanly drive the capacitance of a loaded backplane or the resistance required for proper termination. BTL technology is a result of work that was done within the IEEE 896.1 Futurebus committee specifically to solve the problems of driving a backplane with transmission-line characteristics. By using a smaller voltage swing, lower capacitance drivers, and receivers with precision thresholds, BTL transceivers overcome the "bus driving problem."

Simply stated, the problem is one of driving a low impedance transmission line *(Figures 1* and *2).* The capacitive loading of a bus due to TTL transceivers reduces its impedance from an unloaded value of 60–100Ω to a fully loaded impedance of less than 20Ω. A properly matched termination resistance would therefore require a current of over 300 mA in order to cleanly drive a 3V nominal TTL swing! Since most TTL drivers cannot supply this current, they must depend on reflections to build up the bus voltage to a DC level. This results in a settling-time penalty of one or more bus round-trip propagation delays on every signal transition, or 35 ns on a typical 20" TTL bus.

The low output capacitance of BTL transceivers allows the total capacitive loading of a card in a backplane to be kept under 10 pF. This doubles the impedance of a loaded bus to almost 30Ω. BTL also specifies a reduced signal swing of 1V, which allows a properly terminated bus to be driven cleanly at under 75 mA. Consequently, there are no reflections, and the settling time is zero. A BTL driver can be guaranteed to cross the threshold of every receiver on the backplane with the incident edge of a signal wavefront.

The propagation delay of a bus is also a strong function of the capacitive loading. In the TTL case, the capacitive loading increases the signal propagation delay by a factor of 3 to 5 over an unloaded bus. In a 20" bus, BTL can reduce this delay from a value of 13 ns in the TTL case to less than 9 ns, increasing the potential bus bandwidth significantly.

SYNCHRONOUS BUS TIMING

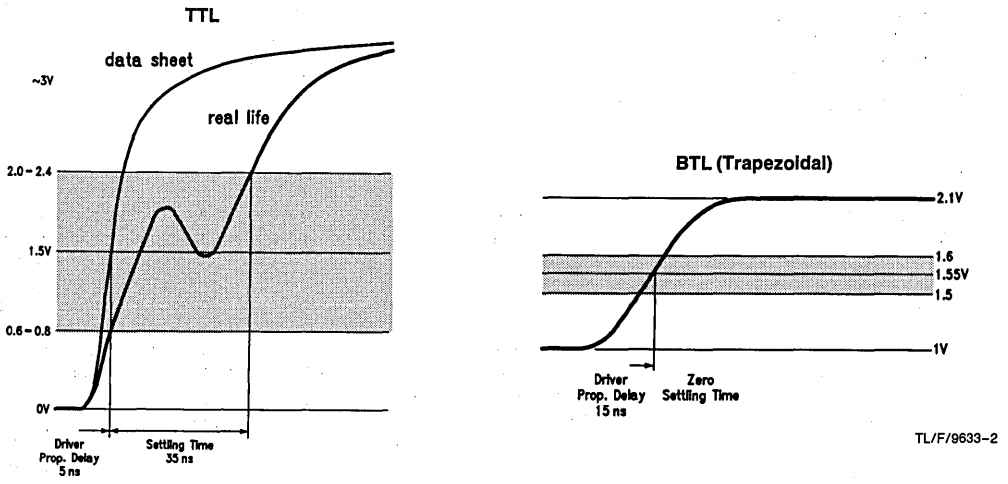
For our first example, let's consider burst data transfers on a synchronous bus. In many backplane systems, burst transfers provide the highest performance, because the overhead associated with the address cycle can be spread out over a number of data cycles. Although other types of transactions may be more complex and require more time (clock cycles), it is likely that many systems will be optimized for burst transfers.

In this example, we are making some simplifying assumptions which ignore some of the penalties associated with a general-purpose synchronous bus. One of these is that the entire interface is synchronized to the bus clock. In general, each card in a backplane will be running off of its own internal high-speed clock. This results in resynchronization metastability problems at both the master and slave interfaces, as well as a clock latency penalty of typically 50% of the clock period. We are also ignoring the return of status from the slave on each data transfer, by assuming all status can be generated before the data is clocked. This would not be true, for example, if parity had to be verified before the next data transfer could take place.

Clock Skew

In this example, the system clock is being distributed to each board through a clock line on the backplane. Since the clock line is being driven from a single point, the loaded capacitance on it is considerably less than on most other lines, and the settling time is typically zero, even in a TTL-based backplane. Due to the finite propagation delay across the bus, however, the clock edge still arrives at each board at different times, creating a relative edge inaccuracy commonly referred to as clock skew.

The worst-case skew can be cut in half by locating the clock source centrally on the backplane, rather than at one end. Additional clock skew will be introduced by the propagation delay differences in the receiver and logic gates that process the clock signal between boards. For a typical 20"



TL/F/9633-1

FIGURE 1. Settling Times

bus, with the clock driver located at the midpoint, total skew can easily exceed 10 ns; in our case, 5 ns for the bus, plus 7 ns for the receiver and a transparent latch used to implement bus wait states.

Synchronous Data Transfer Timing

In this example *(Figure 3),* the worst case data propagation delay from the master to the slave is simply the sum of the delays of the individual components of the data path. This path travels through the master's edge-triggered flip-flop and bus driver, across the length of the bus, and then through the slave's bus receiver and flip-flop, where the incoming data is latched. However, because this is a synchronous system, the data can be "pipelined" to some extent within the intervening logic. This means that the minimum clock cycle possible under this configuration is the sum of the logic skews, plus the maximum bus propagation delay, the set-up and hold times of the receiver, and the clock skew *(Figure 4).*

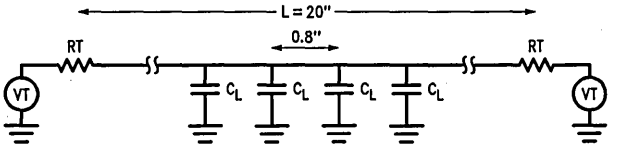
The advantage of a synchronous system is that the absolute timing requirements are set by the clock; the entire system can be optimized with this constraint in mind. This can become a disadvantage as technology advances beyond the point at which the synchronous bus was designed. A synchronous system must be continually redesigned for higher clock rates in order to take advantage of improvements in technology. Synchronous busses are therefore more suited to specific applications than to general-purpose, extended lifespan products.

Synchronous Timing Calculations

The first set of calculations assumes a TTL bus with AS transceivers and logic. As can be seen, the bus settling time overwhelms all the other skews and delays in the system. The upper limit of a discrete TTL synchronous bus implementation is roughly 15 MT (megatransfers/second). No particular advantage is gained by using FAST devices because, while the maximum propagation delays specified for that family are shorter than for AS, the maximum skews are generally greater. The effect of skew specifications is another subtlety of system performance analysis.

Two types of BTL transceivers are currently available, the BTL Trapezoidal and the BTL Turbo. The Trapezoidal transceivers have controlled rise and fall times on their drivers of 6 ns (nominal) to reduce crosstalk interference and switching noise within the backplane. In addition, the receivers incorporate crosstalk filters that practically eliminate far-end crosstalk problems on the bus. The Turbo transceivers eliminate these Trapezoidal features, but are much faster as a result. Switching noise problems are overcome by the use of individual ground return lines for each driver. Stripline backplane construction and careful layout techniques are required to minimize crosstalk.

Although the BTL Trapezoidal transceiver delays are much greater than those of the TTL devices, the absence of settling time results in a smaller overall clock cycle time. A maximum transfer rate of 18 MT becomes possible. When the Turbo devices are used, system throughput increases to 24 MT in this discrete implementation.



TL/F/9633–3

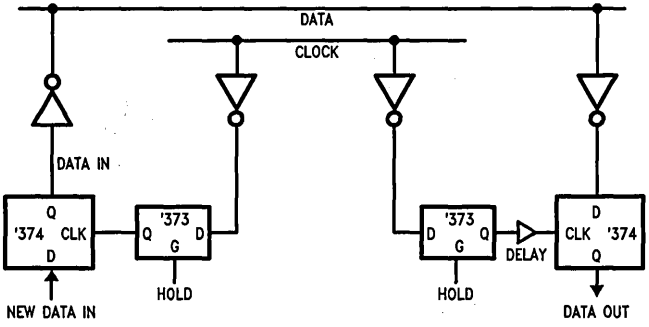
C_L (TTL) ≈ 25 pF/0.8" = 375 pF/ft.
C_L (BTL) ≈ 10 pF/0.8" = 150 pF/ft.

$Z_O \approx 75\Omega$ Unloaded Bus Impedance
$C_O \approx 20$ pF/ft. Distributed Capacitance of Unloaded Bus
$T_O \approx 1.8$ ns/ft. Unloaded Bus Propagation Delay

$Z_L = Z_O/\sqrt{1 + (C_L/C_O)}$ Loaded Bus Impedance
$T_L = L \times T_O \times \sqrt{1 + (C_L/C_O)}$ Loaded Propagation Delay

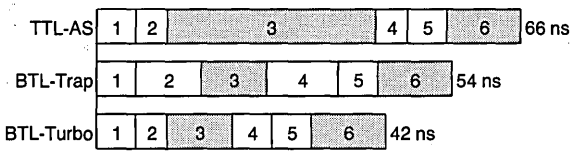
T_L (TTL) ≈ 13.3 ns T_L (BTL) ≈ 8.75 ns

FIGURE 2. Effects of Capacitive Loading



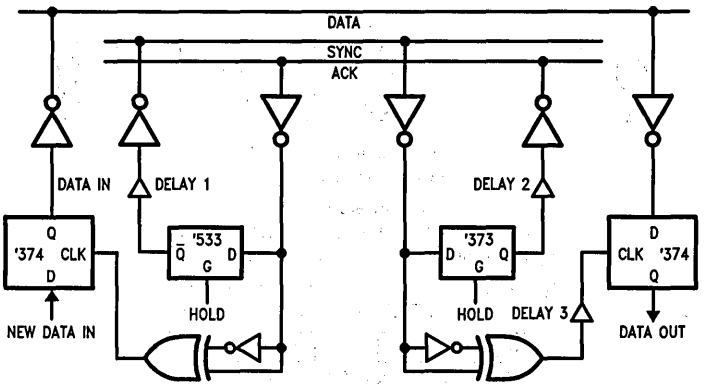
TL/F/9633–4

FIGURE 3. Synchronous Bus Logic for Burst Data Transfers



	TTL AS	BTL Trap	BTL Turbo
1) Max '374 Skew	5.0	5.0	5.0
2) Max Bus Driver Skew	4.5	10.0	5.0
3) Max Bus Delay	35.0	9.0	9.0
4) Max Bus Receiver Skew	4.5	13.0	6.0
5) Max '374 Setup and Hold	5.0	5.0	5.0
6) Max Clock Skew	12.0	12.0	12.0
TOTAL (ns)	66.0	54.0	42.0
MTransfers/second	18.5	18.5	23.8

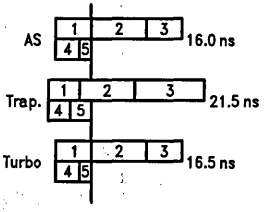
FIGURE 4. Synchronous Burst Data Transfer Timing



TL/F/9633–5

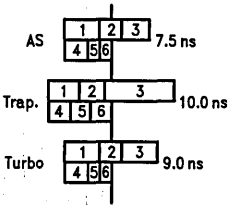
FIGURE 5. Asynchronous Bus Logic for Burst Data Transfers

DELAY 1	TTL AS	BTL Trap	BTL Turbo
1) Max XOR Delay	6.5	6.5	6.5
2) Max '374 Delay	9.0	9.0	9.0
3) Max Data Driver Delay	6.5	15.0	7.0
4) <Min '533 Delay>	−4.0	−4.0	−4.0
5) <Min Sync Driver Delay>	−2.0	−5.0	−2.0
TOTAL (ns)	16.0	21.5	16.5



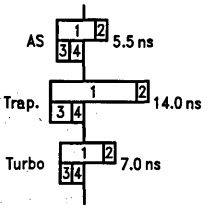
TL/F/9633–6

DELAY 2	TTL AS	BTL Trap	BTL Turbo
1) Max XOR Delay	6.5	6.5	6.5
2) Max '374 Hold Time	3.0	3.0	3.0
3) Delay 3	5.5	14.0	7.0
4) <Min '373 Delay>	−3.5	−3.5	−3.5
5) <Min Ack Driver Delay>	−2.0	−5.0	−2.0
5) <Min Data Receiver Delay>	−2.0	−5.0	−2.0
TOTAL (ns)	7.5	10.0	9.0



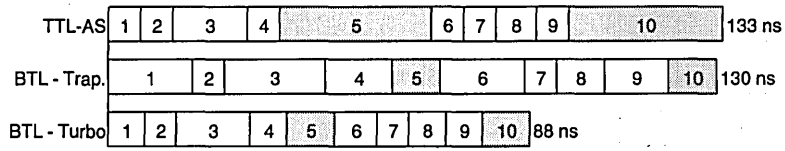
TL/F/9633–7

DELAY 3	TTL AS	BTL Trap	BTL Turbo
1) Max Data Receiver Delay	6.5	18.0	8.0
2) Max '374 Setup Time	2.0	2.0	2.0
4) <Min Sync Receiver Delay>	−2.0	−5.0	−2.0
5) <Min XOR Delay>	−1.0	−1.0	−1.0
TOTAL (ns)	5.5	14.0	7.0



TL/F/9633–8

FIGURE 6. Asynchronous Bus Logic Delay Calculations



	TTL AS	BTL Trap	BTL Turbo
1) Max Ack Receiver Delay	6.5	18.0	8.0
2) Max '533 Delay	7.5	7.5	7.5
3) Delay 1	16.0	21.5	16.5
4) Max Sync Driver Delay	6.5	15.0	7.0
5) Max Bus Delay + Skew	35.0	10.0	10.0
6) Max Sync Receiver Delay	6.5	18.0	8.0
7) Max '373 Delay	6.0	6.0	6.0
8) Delay 2	7.5	10.0	9.0
9) Max Ack Driver Delay	6.5	15.0	7.0
10) Max Bus Delay	35.0	9.0	9.0
TOTAL (ns)	133.0	130.0	88.0
MTransfers/second	7.5	7.7	11.4

FIGURE 7. Asynchronous Burst Data Transfer Timing (Worst Case)

The largest cycle time delay in the final BTL Turbo example is clock skew. Bus skews can be reduced by distributing the clock to each board independently, using a dedicated trace on the backplane such that all lines are of equal length. This makes the clock propagation delay from the driver to each board the same, and thus practically eliminates the bus skew. In addition, better tolerances on driver, receiver, and logic propagation delays (smaller skews) will improve both the clock skew and the effect of transceiver delays on the cycle time.

ASYNCHRONOUS BUS TIMING

Our second example is also of a burst transfer, but this time using asynchronous bus timing. In this system, the master issues a strobe along with the data, and waits for an acknowledgement from the slave before removing the current data from the bus lines. All timing is controlled by the two participants in the data transfer. (Once again, we are assuming that new status does not have to be generated on each data transfer.)

The greatest advantage of an asynchronous bus protocol is its ability to adapt the speed of the bus to the speed of any two communicating boards. The most flexibility is achieved when no technology dependencies are introduced into the protocol. Unlike a synchronous system, where every board is designed with the same timing constraints in mind, a technology-independent module is designed with no assumptions about the timing of the rest of the system. Instead, each transmitting board simply guarantees that its data is valid on the bus at least zero nanoseconds before it issues its synchronization signal, and each receiving board is responsible for ensuring that its data has been successfully latched before issuing an acknowledge. The protocol itself imposes no artificial set-up or hold time limitations.

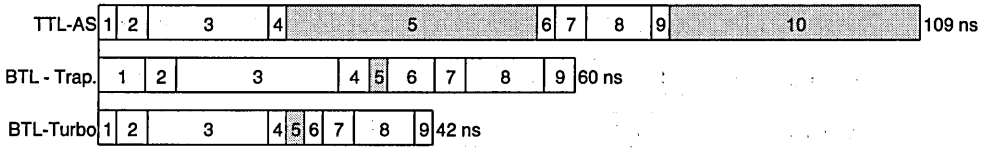
The result of this lack of timing constraints is that a board built today, using today's technology, is guaranteed to work in a system designed perhaps twenty years from now. That system will be forced to slow down whenever necessary to accommodate the greater internal delays and skews of the older module. However, if two future modules are communicating, they will transfer data at the maximum rate allowed by the future technology. The new IEEE Futurebus standard implements this type of protocol.

ASYNCHRONOUS DATA TRANSFER TIMING

The requirement that boards generate their own data synchronization and acknowledge signals, and the likelihood of zero set-up and hold times on the bus, make the timing of the asynchronous system more complicated than the previous example *(Figure 5)*. Also, we are maximizing the performance of the sync/ack handshake by transferring data on each signal transition. This is known as a two-edge handshake.

On the master side, the board must guarantee that its data is valid on the bus before issuing the synchronization signal. This means that a delay must be inserted in the sync signal path (Delay 1) which includes the maximum propagation delays through the XOR clock generation circuit, edge-triggered flip-flop, and data bus driver. This is excessive, however, because the minimum delays through the sync latch and bus driver can be subtracted *(Figure 6)*.

On the slave side, delays are required to guarantee that both the set-up and hold time specifications of the data latch are met. The set-up time delay (Delay 3) ensures that the sync signal, which may have minimum propagation delays through the sync bus receiver and XOR clock generator, arrives at the edge-triggered data flip-flop a set-up time after the data, which may have a maximum delay through



	TTL AS	BTL Trap	BTL Turbo
1) Min Ack Receiver Delay	2.0	5.0	2.0
2) Min '533 Delay	4.0	4.0	4.0
3) Delay 1	16.0	21.5	16.5
4) Min Sync Driver Delay	2.0	5.0	2.0
5) Min Bus Delay + Skew	35.0	1.0	1.0
6) Min Sync Receiver Delay	2.0	5.0	2.0
7) Min '373 Delay	3.5	3.5	3.5
8) Delay 2	7.5	10.0	9.0
9) Min Ack Driver Delay	2.0	5.0	2.0
10) Min Bus Delay	35.0	0.0	0.0
TOTAL (ns)	109.0	60.0	42.0
MTransfers/second	9.2	16.7	23.8

FIGURE 8. Asynchronous Burst Data Transfer Timing (Best Case)

the data bus receiver. The hold time delay (Delay 2) ensures that the data remains at the data flip-flop a hold time after the sync signal, which this time may have a maximum propagation delay through the XOR and the set-up time delay element just introduced. Since the removal of data is controlled by the ack signal, the hold time delay can be reduced by the minimum delays through the ack latch and bus driver, and the minimum propagation delay of the data bus receiver.

This is all very confusing at first, but these delay elements now in place in our circuit guarantee the receiver set-up and hold time requirements while maintaining the technology independence of the bus protocol. Now we can calculate the burst data transfer rate on this asynchronous bus.

The critical path is now the sync/ack handshake. The circuit delays are in place to make sure that data is transferred successfully. To calculate the transfer rate, simply add up all the propagation delays through the sync/ack loop *(Figures 7* and *8):* on the master, the ack receiver, the sync latch, Delay 1, and the sync driver; a bus propagation delay; on the slave, the sync receiver, the ack latch, Delay 2, and the ack driver; and another bus propagation delay.

Should you use worst-case values throughout your evaluation? The beauty of a technology-independent asynchronous protocol is that is will adapt to the speed of the individual logic elements in the sync/ack handshake path. If all the devices happen to have worst-case characteristics, then yes. If they are all fast parts, however, then data transfer will take place under best-case conditions. Both calculations are included, providing the expected operating range of the circuit.

ASYNCHRONOUS TIMING CALCULATIONS

Once again, the TTL design is overwhelmed by the settling time of the bus. Since the sync/ack signal pair are acting as clocks in this system, glitches that may occur during the signal settling time are intolerable. This means that the 35 ns bus settling time must be hard-wired into the receiver logic, and cannot be reduced under best-case conditions. The performance of an asynchronous TTL backplane, from 7.5 to 9.2 MT, cannot approach that of a similar synchronous backplane.

The BTL Trapezoidal system has very similar performance to a TTL backplane under worst-case conditions. However, because there is no settling time penalty associated with BTL signals, the effect of improvements in device operation have a far more pronounced effect. In the best case, the performance is close to that of the equivalent synchronous system. Also, since the bus signal propagation delay is a function only of the distance between the two boards, modules placed in adjacent slots will experience almost no backplane delays.

A BTL Turbo board benefits from the same clean electrical environment that a Trapezoidal one does, except with a 40–50% overall improvement in performance. In the best case, the performance is the same as that of the equivalent synchronous system. Of course, as device parameters improve, with lower propagation delays and skews, the performance of the asynchronous system will continue to improve. The largest reductions in the transfer cycle time will come as interfaces for asynchronous busses such as Futurebus are integrated onto a single piece of silicon, where skews and delays can be more tightly controlled.

CONCLUSION

The use of transceivers designed specifically for the transmission-line environment typical in today's high-speed backplanes provides advantages in both the performance and electrical integrity of a system. The advantages of BTL only become obvious after a careful analysis of data transfer timing considerations. The Trapezoidal and Turbo options provide a designer with the opportunity to make the appropriate application-dependent cost/performance tradeoffs. A sometimes controversial issue is the appropriateness of a synchronous versus an asynchronous design. The former will usually provide an immediate performance advantage in a fully synchronized environment, but a carefully-designed general-purpose asynchronous protocol will often have a longer useful product life.

REFERENCES

1. R.V. Balakrishnan, "The Proposed IEEE 896 Futurebus—A Solution to the Bus Driving Problem," *IEEE Micro,* August 1984.
2. R.V. Balakrishnan, "Physical Layer Timing Design Considerations for High Speed Backplane Busses," Buscon/87 West, January 1987.
3. D.B. Gustavson and J. Theus, "Wire-OR Logic on Transmission Lines," IEEE Micro, June 1983, pp. 51–55.
4. *Interface Databook,* National Semiconductor Corp., Santa Clara, CA, 1986.
5. *ALS/AS Logic Databook,* National Semiconductor Corp., Santa Clara, CA, 1987.
6. *IEEE P896 D7.5a,* "Futurebus: A Bus Standard for Multiprocessing Architectures", IEEE, June 1987.

TABLE I. Device Parameters

Device	Parameter (Transition)	Minimum Prop. Delay	Maximum Prop. Delay	Maximum Skew	Setup/Hold
DM74AS374 Edge-Triggered Flip-Flop	LH HL	3.0 4.0	8.0 **9.0**	5.0 **5.0**	**2.0/3.0**
DM74AS373 Transparent Latch	LH HL	**3.5** **3.5**	**6.0** **6.0**	**2.5** **2.5**	2.0/3.0
DM74AS533 Inverting Transparent Latch	LH HL	**4.0** 4.0	**7.5** 7.0	**3.5** 3.0	2.0/3.0
DM74AS86 2-Input XOR	Other Input L Other Input H	2.0 **1.0**	**6.5** 6.0	4.5 5.0	
DM74AS240 Bus Driver/Receiver	LH HL	2.0 2.0	6.5 5.7	**4.5** 3.7	
DM74AS242 Bus Transceiver	LH HL	**2.0** 2.0	**6.5** 5.7	**4.5** 3.7	
DS3896 BTL Trapezoidal Transceiver	Rx Tx	**5.0** **5.0**	**18.0** **15.0**	**13.0** **10.0**	
DS3893 BTL Turbo Transceiver	Rx Tx	**2.0** **2.0**	**8.0** **7.0**	**6.0** **5.0**	

Note: Values in boldface are those used in the preceding calculations.

Signals in the Futurebus+ Backplane

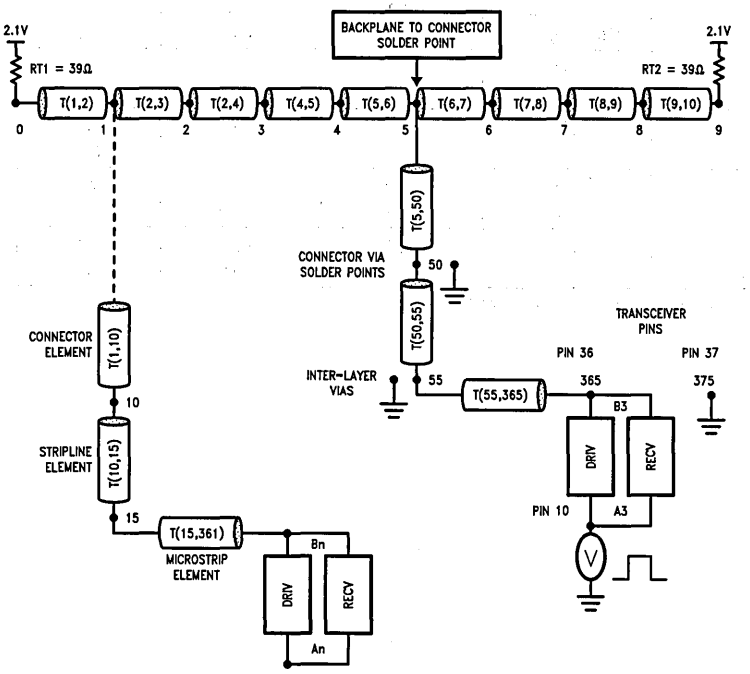
National Semiconductor
Application Note 738
Stephen Kempainen

The Futurebus+ backplane is a complex electrical environment that consists of many circuit elements. The modeling of such an environment can become time consuming and expensive. The Futurebus+ Electrical Task Group Expert Team has detailed the circuit elements in their SPICE simulation. An average Futurebus+ simulation contains over 10,880 individual elements and gobbles 8 CPU hours on a single user VAX 8650. This note is an attempt to simplify the circuit model and gain an intuitive understanding of interactive signal path elements. The elements are investigated by probing at individual impedance breaks that are considered significant. Waveforms of the signal will be correlated with the TDR signals from the same signal paths. An investigation of ground signal variations and crosstalk measurements is included because of relevance to signal measuring in this environment. The relation between the crosstalk, ground bounce and the signal path impedance will be pursued to see their combined effect on the noise margin.

The interconnect effects on the electrical signal become critical in high speed multilayer board design such as Futurebus+. PCB traces must be treated as transmission lines due to the rapid transition times of the signal. Analyzing PCB traces uncovers the impedance mismatches caused by seemingly harmless corner geometries, parasitic and crossover effects, and inter-layer vias. The impedance mismatches also affect crosstalk coupling and signal reflections which are a major concern due to the large chunk of noise margin they may consume. To demonstrate how these circuit elements affect the signal, this article will follow a signal from the transceiver as it propagates into the backplane.

FUTUREBUS+ BACKPLANE AND BOARD MODEL

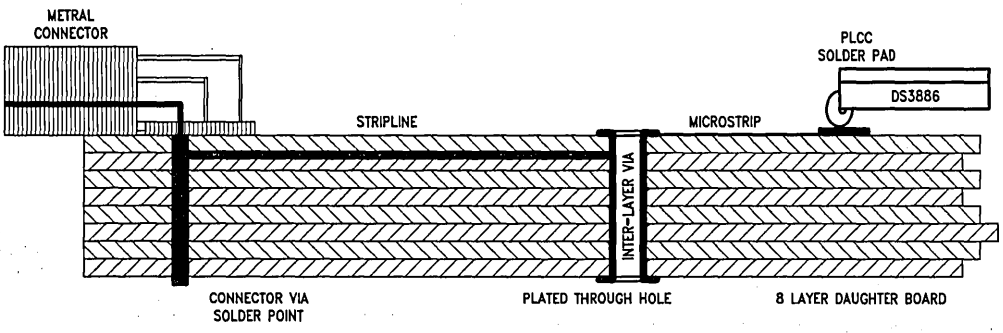
Figure 1 models the signal path from a transceiver in one board, through the backplane, to a transceiver in another board. Both of the boards are mounted in a ten slot backplane. The dashed line to one module indicates it can be removed or moved around to different locations for this analysis. It is the receiver and it is seen as a load by the driver module. To first emphasize the driver module stub effects alone, the receiver is not inserted into the backplane. This focuses the driver response to only the transmission line elements. This model can be generalized to any backplane. However, it is derived from specific equipment used to obtain these waveforms. The backplane is provided by Bicc-Vero Electronics, No. 819-304105E. It uses 39Ω,



TL/F/11107–1

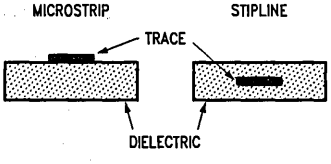
Model of the Futurebus+ backplane with two daughter boards; one in slot 1, and one in slot 5 with an input signal at pin 10.

FIGURE 1



TL/F/11107–2

FIGURE 2. Eight Layer Daughter Board, Side View



TL/F/11107–3

FIGURE 2a. PCB Track Cross Section

surface mount, termination resistors and has a 1 inch spacing between the slots (soft metric). The board is provided by Hybricon. It is a Futurebus+ Wire-Wrappable Board 6U x 280mm. The part number is 031-126-10. The board is laid out (eight layers) for the National Semiconductor Futurebus+ Chip Set transceivers and can accommodate 64 data bits.

Figure 2 is a cross section of the Hybricon board showing the signal path of the DS3886 Latched Data Transceiver, pin-36. The illustration emphasizes the physical impedance differences of the transmission path elements. Examination of the Time Domain Reflection response of the signal path is a good way to "electrically see" these differences.

TIME DOMAIN REFLECTION

TDR uses a step generator to apply a positive going impulse to the signal path being investigated. The step has a very fast rise time of 35 ps and a 200 mV amplitude. The step travels down the path at the propagation velocity of the line. If there are impedance mismatches in the signal path, part of the incident wave will be reflected. The reflected wave is then algebraically added to the incident wave at the point where the mismatch occurs. The total voltage wave appears on the oscilloscope as a road map to the impedance breaks encountered by the propagating step.

A quick review of reflection coefficient (ρ) fundamentals will be helpful to intuitively understand the effects of signal path impedance breaks. Then, investigating three load impedance conditions will suffice. For all cases, the TDR generates the step from a 50Ω source and it is carried by a cable with characteristic impedance, $Z_O = 50\Omega$, to the device under test, DUT. First case is if the DUT were a 50Ω load, then ρ would be 0 and the wave on the scope would appear as a straight line after the step, no reflected wave added to the incident.

$$\rho = (Z_L - Z_O)/(Z_L + Z_O) = 0 \text{ for } Z_L = Z_O$$

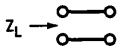
Z_O = cable characteristic impedance

Z_L = load impedance

The equation for adding the reflected wave, Er, to the incident wave, Ei, is as follows.

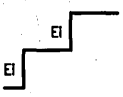
$$E = Ei + Er, \text{ where } Er = Ei(\rho)$$

The second case is infinite load impedance as in *Figure 3a.* ρ is equal to +1 in this case and the reflected wave equals the incident wave. The total wave is then double the incident, *Figure 3b.* Now consider when the incident wave hits an inductive impedance. The current can't change instantaneously so the load momentarily appears as an open due to the increased impedance. The $\rho = +1$ at $t = 0$ in *Figure 3c.* Reflected voltage is ideally the same as the incident voltage for that moment. As the inductor current builds exponentially, the impedance drops toward zero, *Figure 3c.* The voltage a long time after $t = 0$ is determined by resistance in series with the inductor. As t goes to infinity, the reflection coefficient is $\rho = (R - Z_O)/(R + Z_O)$, where R = series resistance of the inductive load.



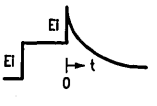
TL/F/11107–4

FIGURE 3a



TL/F/11107–5

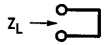
FIGURE 3b



TL/F/11107–6

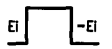
FIGURE 3c

The third case is zero load impedance as in *Figure 4a.* $\rho = -1$ and the reflected wave is subtracted from the incident wave leaving no voltage. Expanding on this idea, when the incident wave hits a capacitive impedance, the capacitor won't accept a sudden voltage change. No change in voltage appears as a short circuit instantaneously and $\rho = -1$ at $t = 0$. The capacitor voltage builds exponentially and the impedance rises to a level determined by the shunt resistive component of the load, *Figure 4c.* The final value of ρ is again $(R - Z_O)/(R + Z_O)$, only R = shunt resistance of the capacitive load.



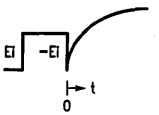
TL/F/11107–7

FIGURE 4a



TL/F/11107–8

FIGURE 4b



TL/F/11107–9

FIGURE 4c

TDR AND IMPULSE ENERGY

Another way to look at TDR results is by considering the energy contained in the step impulse. This energy is transmitted over a non ideal medium so there are losses, but for short distances it is not unrealistic to consider the energy of the pulse to be constant. Now consider the capacitive impedance break; increasing capacitance reduces the characteristic impedance.

$$V/I = Z_O = \sqrt{(L_O/C_O)} \text{ where:}$$

L_O = inductance per unit length

C_O = capacitance per unit length

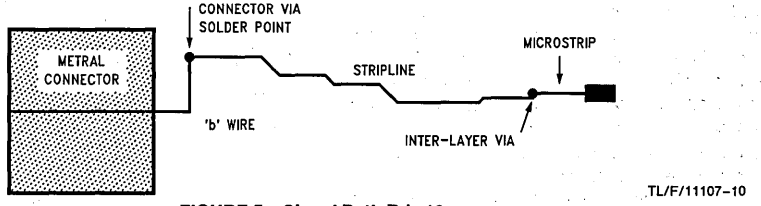
Since the energy of the pulse remains constant and the voltage and the impedance drop, the current must increase proportionally to the lowered voltage. This increased current charges the capacitor at a time constant that is determined by Z_O in parallel with the shunt R to the capacitor. As the capacitor stores the energy, the current drops off and the capacitor appears as an open circuit after a long steady state condition.

TDR SYSTEM ERRORS

The extremely fast rise time of the step impulse is important to TDR analysis. Since the leading edge of the incident step is made up almost entirely of high frequency components, it accentuates the small reactive impedance mismatches of a signal path. As the step travels down a non-ideal transmission line, the higher frequencies are attenuated by skin effect losses and dielectric losses. This distorts the step, and is called cable loss. The degraded rise time limits the accuracy of reflection measurements through a multiple discontinuity signal path. TDR measures each succeeding discontinuity with less accuracy, because the transmitted step degrades and multiple reflections occur. The stub of a daughter board qualifies as a multiple discontinuity path, so the resulting waveforms must be analyzed as such.

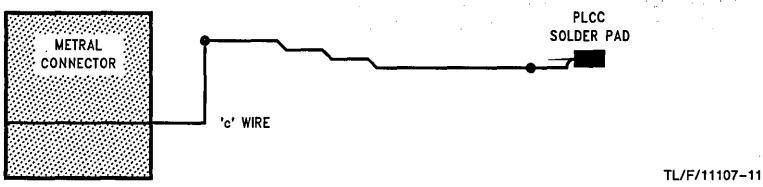
TDR AND FUTUREBUS+ SIGNAL STUBS

With this in mind, an analysis of the TDR waveforms of the signal path can be performed. *Figures 5a* and *5b* show the artwork for two signal path traces. The actual path length is 64mm. *Figure 5a* is the path used for all the waveforms gathered in this analysis and *Figure 5b* is a path for comparison of the TDR responses. The apparent similarity of the artwork does not show the electrical differences of the paths while the TDR waveforms do. *Figures 6* and *7* include the waveforms from these two signal paths that are similar but different enough to demonstrate some characteristics. The signal path from pin 36 of the DS3886 goes through connector B-b-16 as designated by the Futurebus+ standard, *Figure 5c.* This path is included in both figures. Also in both figures is the reflection from just the SMA connector that is used to launch the step impulse into the signal path. It is terminated with a 50Ω load. The inductance of the SMA leads can be seen by the sharp increase in impedance. The return to the 50Ω level is then illustrating the first case in the TDR review for the $\rho = 0$ situation.



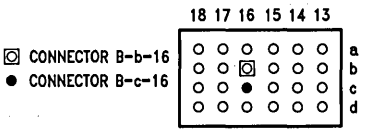
TL/F/11107–10

FIGURE 5a. Signal Path B-b-16



TL/F/11107–11

FIGURE 5b. Signal Path B-c-16

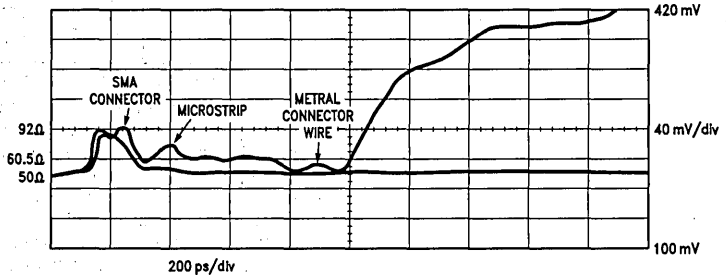


TL/F/11107–14

FIGURE 5c. Signal Connector Block B, Rows 13 to 18

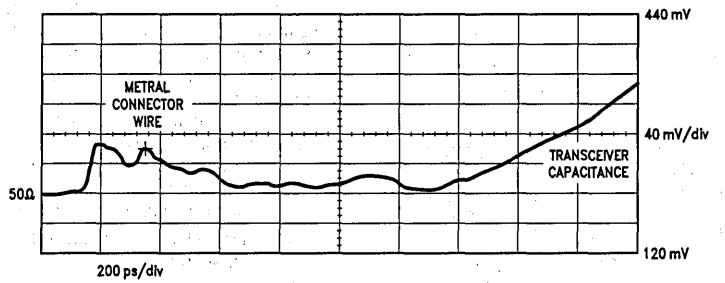
Figure 6 shows the effect of launching the TDR step into the solder pad. The first inductive bump is the error introduced by the SMA launching mechanism. The inductance of the microstrip follows the dip in impedance (capacitive solder pad). The path impedance is raised to 75Ω by both signal paths, *Figure 7.* Notice how the longer microstrip in *Figure 5a* adds distance to the inductive bump compared to the waveform from the shorter microstrip in *Figure 5b.* Then notice that the inter-layer via causes a capacitive drop in impedance. The stripline impedance settles in at about 60Ω for both of the paths. The next dip is the capacitance of the connector via solder point followed by the inductive increase of the connector wires. Notice that the longer "c" wire increases the impedance more than the "b" wire. Finally, the step hits the open end of the connector and the signal voltage doubles which indicates the $\rho = 1$ situation.

Figure 6a is included to show how the degradation of the incident wave through the multiple impedance mismatches of the signal path affects impedance level measurement. The TDR impulse is launched into the connector end of the path rather than the solder pad end. The difference is best seen in how the impedance of the connector is much greater when the high frequency components of the incident wave have not been attenuated by the previous impedance mismatches. The connector launch displays an impedance of 90Ω rather than the small increase that is shown in the end of the launch from the solder pad. *Figure 6a* also shows the capacitance of the transceiver mounted on the solder pad that is charging to the open state at the time constant rate.



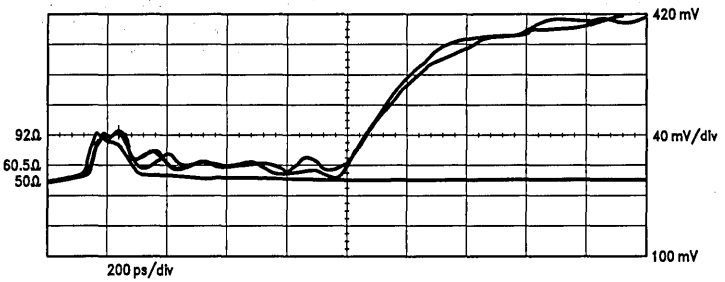
TL/F/11107–12

FIGURE 6. Two TDR Waveforms, 50Ω Termination and Path B-b-16



TL/F/11107–15

FIGURE 6a. TDR Launch Into Connector B-b-16 with DS3886 Mounted On Board

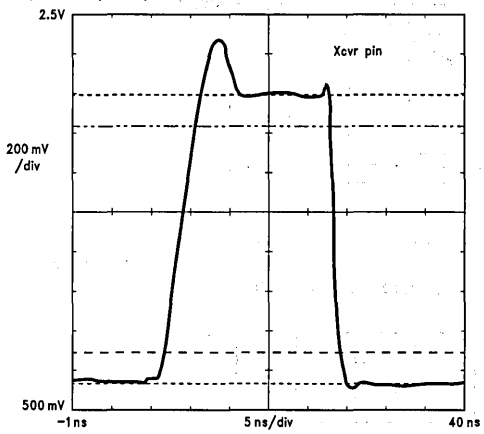


TL/F/11107–13

FIGURE 7. Three TDR Waveforms

THE TRANSCEIVER IN THE UNLOADED BACKPLANE

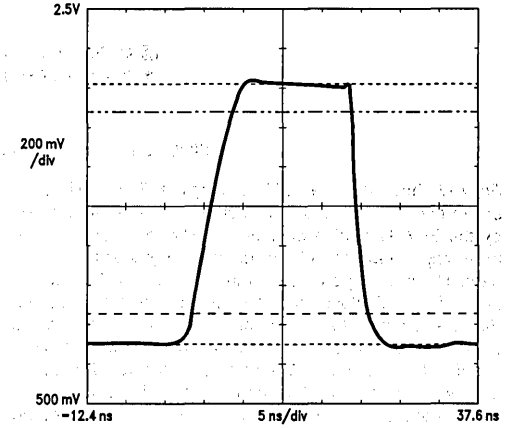
The transmitting transceiver is mounted in slot 5, *Figure 1.* Pin 36 is named B3 in both the DS3883, 9 Bit Data Transceiver, and the DS3886, the Latched Data Transceiver which is used in this application. Pin 37 is the B3 GND pin, the BTL ground for B3 (see section on BTL, QGND, and logic GND). The signal waveform in *Figure 8* is obtained by probing the circuit with a low inductance ground tip at these two pins. The signal displays a rapid fall time, large overshoot and minimal undershoot. This signal is different than that obtained from bench testing with minimal jig inductance and capacitance, *Figure 8a.* That waveform shows slower fall time and no overshoot. There are a couple of reasons for these backplane circuit responses. The first being the microstrip circuit element, T(55,365). Physically, this element is a very narrow microstrip between the solder pad and the inter-layer via. The relatively high inductance of this microstrip, shown in *Figure 6a,* inhibits the sudden change in current presented by rapid transition times. This initially appears as a large impedance mismatch which makes the reflection coefficient (ρ) approach +1 instantaneously. The effect of ρ is to speed the slew rate on the fall time and extend the overshoot on the rising edge. The different response of the two edges is due to the active pull down and the passive pull up.



TL/F/11107–16

Rise 3.881 ns	Fall 1.346 ns	Max 2.37800 V	Measure-ments	Horz Mag 1 x Horz Pos Gr Opts	
Min 616.000 mV	Over Shoot 19.4787 %	Under Shoot 1.37174 %	Statistics Comp & Def Sample # 100	Remove Wfm 1 ST06	Pan/ Zoom on

FIGURE 8



TL/F/11107–20

Rise 5.410 ns	Fall 2.348 ns	Min 780.000 mV	Measure-ment	Distal 90% Proximal 10%
Max 2.13600 V	Over Shoot 1.52207 %	Under Shoot 1.67428 %	Statistics Comp & Def Continuous	Remove Wfm 1 ST05

FIGURE 8a. Bench Test Jig Waveform

The second reason is the path inductance and line delay of the backplane, daughter board, and the termination scheme. The transmitter pull down transistor is turned on with a very large base current needed to supply 80 mA collector current. This collector current is supplied by the termination resistor which is located at some electrical distance (backplane and daughter board paths) from the transistor. As the transistor experiences a hard turn on, it initially sees what appears to be an open circuit. The momentary open circuit causes the overshoot and fast falling edge. This is due to the inductance and delay of the signal path preventing the current from changing immediately at the collector of the transistor. This inductance limits the driver slew rate control at the transceiver pin. Bench testing of the same part shows a fall time 1 ns longer than the daughter board fall time. The difference in fall times under test conditions are due to load proximity and availability of almost instantaneous current on the test jig. The rise time is much slower because it is a passive pull up and is controlled by the exponentially decaying current due to the inductance.

Another look at the waveform will show that there is an increase in voltage just prior to the high to low transition. This is also due to the large, fast voltage change at the base to the output transistor. The voltage step is coupled through from the base to the collector by the Miller capacitance. This small spike only shows up in the backplane environment for the same reasons as already explained.

IMPEDANCE MISMATCHES

The next probe is at the inter-layer via points of the signal line and ground line. Labeled point 55 in *Figure 1.* These vias present a relatively large capacitive impedance to the signal. The capacitance of the plated through hole (PTH) via has been estimated as high as 1.1 pF by Hybricon down to 0.75 pF by the Futurebus+ Expert Team. This is also the point at which the microstrip trace changes to a stripline trace. The capacitance of a PCB track varies directly with track length and width, but inversely with the dielectric thickness. Typically, this corresponds to about a 50% increase in capacitance from outer to inner layer for an eight layer board. As seen by the TDR investigation, the path impedance drops from 75Ω to 60Ω at this point, *Figure 6.* Thus, the stripline circuit element, T(50,55), can be characterized by an increase in the capacitance per unit length.

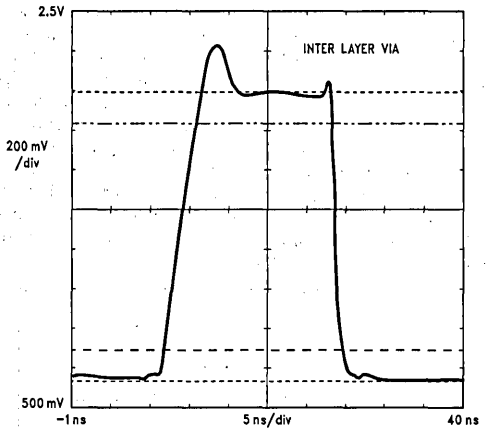
Figure 9 illustrates the damping of the overshoot and undershoot that is caused by the capacitive reactance. The fall time is increased by the capacitance. This can be intuitively understood by considering the capacitive impedance break of *Figure 4c.* This has a counter balance affect on the path inductance so the needed current is available. Initially, the load appears as a short circuit because the capacitance will not accept an immediate change in the voltage. The ρ at the impedance mismatch then initially approaches −1. The quick charging of the capacitance pulls the slew rate out of a nose dive and limits the undershoot. The rise time shows a slight increase, but the resolution of the scope comes into play for times less than 150 pico seconds.

The third major impedance mismatch of the signal path occurs at the PTH to Metral connector solder point. Point 50 in *Figure 1.* The ground reference for the low inductance tip is the solder point for an adjacent connector ground pin. The circuit element for the connector is modeled by T(5,50). A SPICE model is provided by DuPont, the connector manufacturer. The TDR waveform clearly shows the capacitance of the solder filled PTH lowering the impedance to 50Ω, *Figure 6.* The same figure then shows the connector wire presents an inductive impedance increase for the signal. *Figure 10* shows considerable overshoot damping. It also shows an increase in rise time and fall time. The multiple discontinuities to this point have degraded the initial rise time of the signal so that the overall effect is that of line loss. The passive pull up accentuates only the resistive portion of the impedance rather then the reactive.

Notice the reflection that is well defined at the bottom of the falling edge. A closer examination of *Figure 8* shows that this same reflection is present in an attenuated form. The peak of this reflection is about 2 ns from the point where the signal crosses into an undershoot state. The delay per unit length, t_{pd}, of the unloaded backplane depends on the relative magnetic permeability, μ_r, the relative dielectric permittivity, ϵ_r, and the speed of light, c.

$$t_{pd} = \frac{\sqrt{\mu_r \epsilon_r}}{c}$$

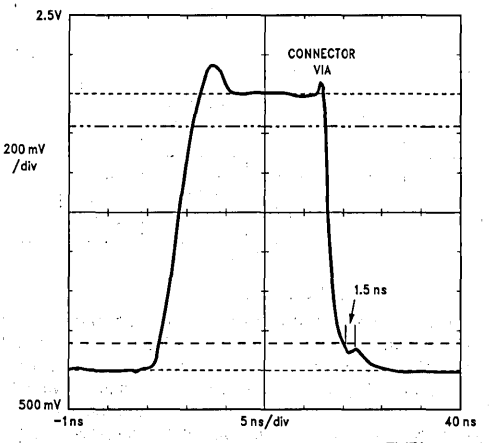
With $\epsilon_r = 4.7$ and $\mu_r = 0.99$, then $t_{pd} = 0.18$ ns/in. So a round trip of 10 in., slot 5 to slot 0 and back, will be a delay of about 1.8 ns. This is almost exactly the delay of the reflected pulse at the connector solder point in *Figure 10.* The delay is measured from the falling edge tangent line. The period of this pulse is about 1.5 ns which is a frequency of 667 MHz.



TL/F/11107–17

Rise 4.000 ns	Fall 1.689 ns	Max 2.34200 V	Measure-ments	Horz Mag 1 x Horz Pos Gr Opts	
Min 652.000 mV	Over Shoot 16.8056 %	Under Shoot 555.556 m%	Statistics Comp & Def Sample # 100	Remove Wfm 1 ST05	Pan/ Zoom on

FIGURE 9



TL/F/11107–18

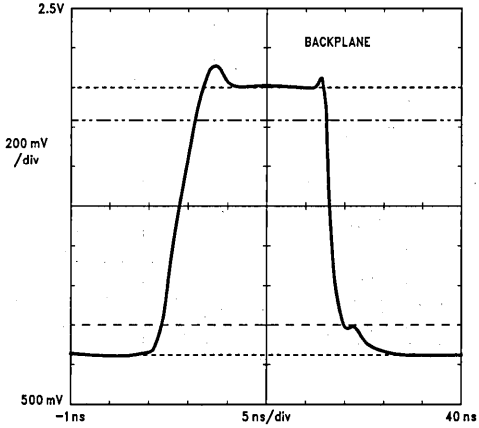
Rise 4.457 ns	Fall 2.410 ns	Max 2.26200 V	Measure-ments	Horz Mag 1 x Horz Pos Gr Opts	
Min 696.000 mV	Over Shoot 11.1270 %	Under Shoot 570.613 m%	Statistics Comp & Def Sample # 100	Remove Wfm 1 ST07	Pan/ Zoom on

FIGURE 10

ENTERING THE BACKPLANE

After the connector, the signal reaches the backplane environment. The signal has been transformed by the daughter board path. Besides the impedance factors, the skin effect losses have rounded top and bottom portions of the edges.

The probe points are at the solder points of the Metral connectors to the backplane PTH. The waveform in *Figure 11* was obtained at the slot where the board is inserted, just on the backplane side of the connector from the previous figure. The connector increases the fall time by 500 ps and damps the overshoot. The increase in fall time here appears to be a result of the reflected pulse increasing in amplitude. As the incident wave propagates further down the backplane the same damping of overshoot and increase in transition times occurs. The backplane characteristics are dependent on the loading that is present in the form of inserted boards with transceivers.



TL/F/11107–19

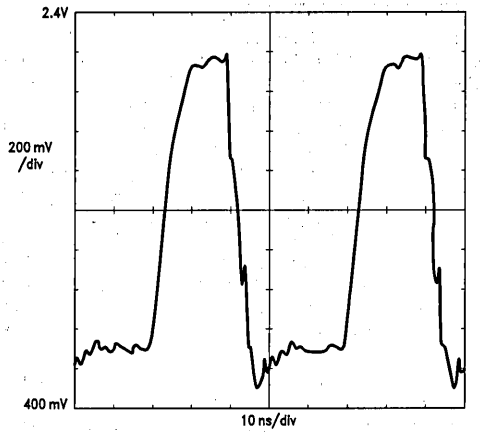
Rise 4.688 ns	Fall 2.898 ns	Max 2.21600 V	Measure-ments	Distal 90% Proximal 10%	
Min 742.000 mV	Over Shoot 8.43195 %	Under Shoot 591.716 m%	Statistics Comp & Def Sample # 100	Remove Wfm 1 ST010	

FIGURE 11

THE LOADED BACKPLANE

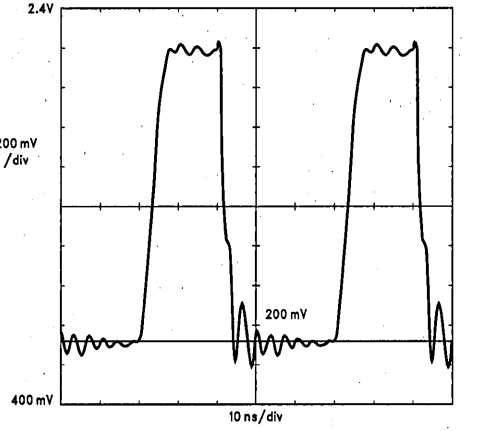
The distributed capacitive loading of the backplane has significant effects on the signal. The position of the loads with respect to the driving board will determine how the reflections add to degrade the signal. The worst case is when the reflections cut into the noise margin; i.e., the reflections that are positive going on the low output and negative going on the high output. The investigative results show that reflections in the high state never go below the 2.1V level by more than 50 mV. The problem on the high end occurs when the bus is fully loaded. At 20 MHz and fully loaded, the rising edge becomes rounded, *Figure 12.*

The worst bite into the noise margin was found in the case of 2 loads, 12 pF each, in specific slots on the backplane. The driver in slot 5 and the loads in slot 6 and 0 caused sustained ringing with a peak amplitude of 200 mV into the noise margin low. The case is shown in *Figure 13.* It should be noted that the period of the ringing in *Figure 13* is about 2 ns. This corresponds to a frequency of 500 MHz. This presents a demand on test equipment to pick up these high frequency signals.



TL/F/11107–21

FIGURE 12. Backplane Probed at Driver = Slot 5, 10 pF to 12 pF Loads in Every Slot



TL/F/11107–22

Rise 4.938 ns	Fall 3.603 ns	Max 2.21200 V	Measure-ments	Horz Mag 1 x Horz Pos Gr Opts	
Min 598.000 mV	Over Shoot 1.64384 %	Under Shoot 8.90411 %	Statistics Comp & Def Continuous	Remove Wfm 3 ST013	Pan/ Zoom on

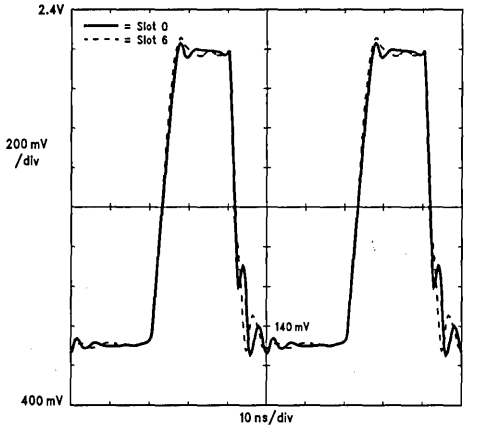
FIGURE 13. Backplane Probed at Driver = Slot 5, 12 pF Loads in Slots 0 and 6

GIGA HERTZ BANDWIDTH

A 400 MHz probe and scope would not pick up all of the frequency components of this ringing. Because of the high frequency components that comprise this signal, all of the measurements done by National Semiconductor on the Futurebus+ chip set are obtained by using the Tektronix

P6204 FET probe and 11A72 amplifier mounted in the 11403 digitizing oscilloscope. This combination has a bandwidth of 1 GHz.

Figure 14 is included to show how a single load of 12 pF will cause different reflections depending on where it is inserted with relation to the driving transceiver. The line delay is evident when the reflection from the adjacent load appears before the reflection from the far end load. The different loading positions will determine the waveform shape and how it will encroach on the noise margin.



TL/F/11107–23

FIGURE 14. Backplane Probed at Driver = Slot 5

WHY ALL THE DIFFERENT GROUNDS?

There are three different types of gound pins on the National Semiconductor Futurebus+ Chip Set. They are the logic ground (GND), the BTL grounds (B0GND–B8GND) and the bandgap reference ground for receiver threshold (QGND). All of these ground reference pins are isolated inside of the chip to limit the interference from high current switching transients. Outside the chip, the bandgap reference ground should be connected to the backplane ground through a quiet channel. The isolation purpose is so the receiver input threshold will follow the same reference as the signals coming off the backplane. The other grounds should be tied to the board ground plane to prevent ground loop currents inside the chip.

FUTUREBUS+ TRANSCEIVER GROUND BOUNCE

A single transceiver can have up to 9 BTL channels switching at the same time. If each channel sinks 80 mA, there is substantial current switching taking place. The combination of the ground lead inductance and finite resistance of the current return paths cause voltage drops and rises to occur along this path that are proportional to the changing current.

$$V = L\,(di/dt)$$

where V = amplitude of the ground bounce

L = inherent inductance of signal and ground trace

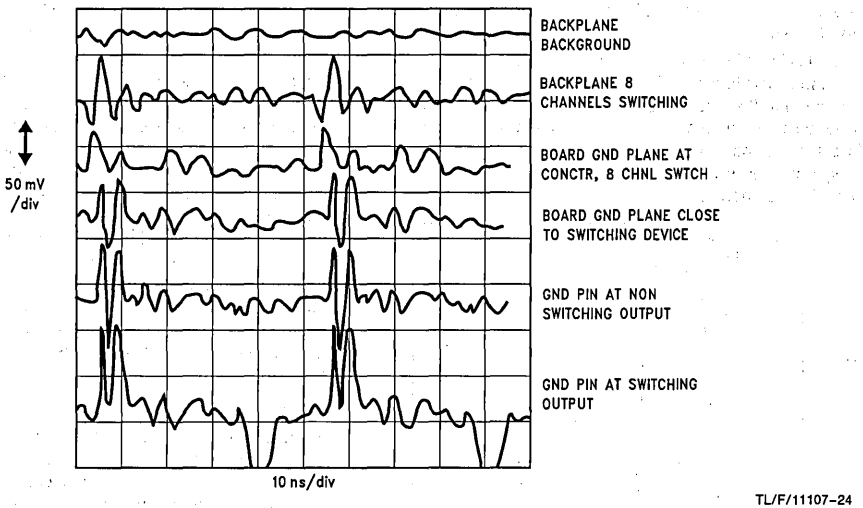
The DS3886 Latched Data Transceiver mounted in the Hybricon proto board was used to investigate the amount of ground shift that is experienced in the Futurebus+ environment.

Eight channels are connected to the same input so that they are switching simultaneously. The ninth channel, B3 (located between the other eight), is driven to the asserted state and used as a reference. Six other data transceivers were also on the board and allowed to switch at random (open driver inputs). The Futurebus+ connector pin layout uses 1 of every 3 pins as a ground pin. The Hybricon board links all these pins to the board ground plane as they enter through the Metral connector. The transceiver BTL ground pins are mounted to the solder pad and then traverse a microstrip track to a PTH via to make ground plane contact. The microstrip adds inductance to the ground path but is necessary for even heating to solder the chip package to the thermally isolated pad.

PROBING THE GROUND

The backplane ground plane is used as a reference to investigate all of the ground differences in the circuit. It is accessed through a ground tab connector on the Metral power connector module. *Figure 15* shows the idle backplane noise at the top of the picture. The GHz probe with a short, low inductance alligator clip ground was used to probe two of the empty slot ground tabs. There was no transceiver activity for this situation. The second from the top waveform is the same probe position only eight transceiver channels are now switching. Large disturbances in the signal occur at the time that eight channels are all going from high to low, the time of substantial active current change. Notice how the low to high transition does not create the same sort of voltage spike on the ground signal. This is because the collapsing current doesn't have a large *di/dt.*

The same backplane ground reference was used for all of the measurements in *Figure 15.* The third pattern was obtained probing the daughter board ground plane close to the Metral connector between the switching transceiver and the backplane. The disturbances are muted in this case by the bypass capacitors of the board. The board is decoupled by 4–180 μF and 14–0.1 μF capacitors. The next waveform is from the same ground plane but it is probed at the via that connects the microstrip from the transceiver BTL ground pin to the board ground plane. This waveform is a slightly attenuated version of the waveform seen on the non switching ground pin. This is because the waveform seen at the B3GND pin is coming from the board ground planel The inductance of the microstrip and the lead frame inside the package increase the overshoot and the undershoot of the ground bounce. The worst ground disturbance is measured at one of the switching channel BTL ground pins. This is as expected from interior transceiver noise caused by the hard turn on of the output transistor creating very rapid build and collapse of current.



TL/F/11107–24

FIGURE 15

CROSSTALK IN THE FUTUREBUS+ ENVIRONMENT

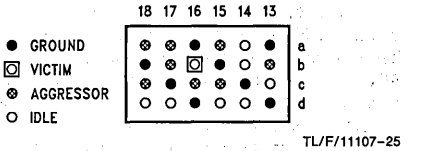
The crosstalk problem has received a lot of attention. There is the potential for significant forward and backward crosstalk due to the high speed signals, multiple transmission path media, and the density of the signal lines. These are the simplified equations relating the contributing factors.

$V_{bkwd} = (V_a/t_r)\,(\ell/2t_\ell)\,(C_CZ + L_C/Z)$
= backward coupled voltage to victim line

$V_{frwd} = (V_a/t_r)\,(\ell/2)\,(C_CZ - L_C/Z)$
= forward coupled voltage to victim line

V_a = aggressor signal amplitude
t_r = aggressor signal transition time
ℓ = line length
t_ℓ = line delay
Z = line impedance
C_C = capacitive coupling due to electric field
L_C = inductive coupling due to magnetic field

Both types of crosstalk are directly proportional to the amplitude, and inversely proportional to the transition times of the aggressor signal. The capacitive and inductive coupling affect both types of crosstalk. In backward crosstalk they add together and are multiplied by the aggressor amplitude to give a same polarity pulse to the victim. In forward crosstalk, the quantity $(C_CZ - L_C/Z)$ is multiplied by the aggressor amplitude to give a pulse of either polarity depending on the relative size of the coupled reactances. The connector does present a special problem due to the open wire configuration. The inherent inductance of the open wires and the proximity in the Metral connector are favorable situations for crosstalk. Not modeled in the above equations but still a factor is the signal wave velocity differences. Forward crosstalk also results from velocity differences of an aggressor signal due to the conductive medium contacting substances of different dielectric constants. The microstrip line is such a medium that contacts both air and epoxy glass. This creates an energy pulse that will couple electrostatically to the victim. For these reasons, crosstalk was investigated in two different ways.



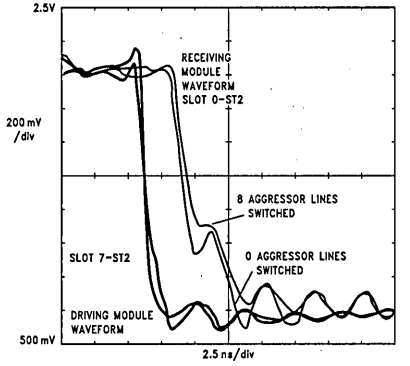
TL/F/11107–25

FIGURE 16. Module B, Section 3, Used In Crosstalk Measurements. The Victim Line Is Labeled B-b-16

Futurebus+ standards committees set up the pin designations on the Metral connector so that there is one ground pin for every 2 signal carrying pins. The worst case then is the situation where 1 signal pin can be surrounded by 5 signal pins and 3 ground pins as in *Figure 16.* The Expert Team tested crosstalk using a switching line as victim and measured the difference between 0 and 5 aggressors at a receiving module. *Figure 17* shows these same tests for DS3886.

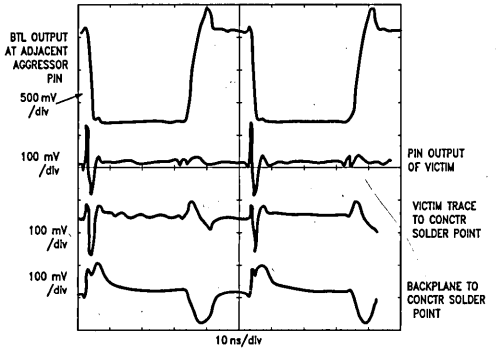
The driver module is located in slot 7 and two receiver modules are in slots 0 and 9. As mentioned in the section on reflections, this is the worst configuration for cutting into

the noise margin. It is also a long length for the parallel backplane tracks to cross couple. The signal line ST2 was used as the victim and ST0–ST7 as the aggressors. *Figure 17* shows the falling edges at the driver and receiver pins for the two conditions, only the victim switching and then all aggressors and the victim switching. The largest amount of induced voltage onto the victim line is 50 mV. This is the same result as the Expert Team.



TL/F/11107–26

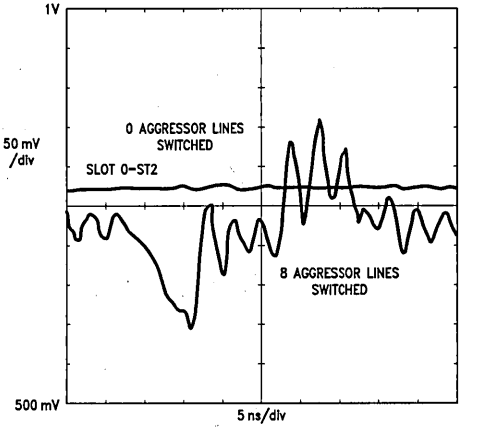
FIGURE 17



TL/F/11107–27

FIGURE 18. Victim Is Only Asserted, 8 Aggressor Channels Switching

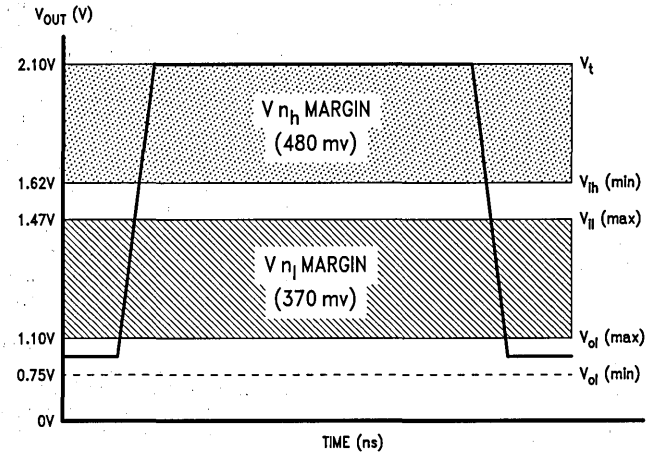
Figure 18 shows results of a different way of investigating the crosstalk than the Futurebus+ Expert Team method. The victim line was held in the asserted state while 8 aggressor lines were switching. The top waveform is the aggressor signal. The three lower waveforms are the victim signal probed at daughter board impedance points. The figure shows that the victim experiences a 100 mV pulse into the noise margin at the high to low transition of all the aggressors. This case is measured on the backplane at the same slot as the driver. A demonstration of the high inductance of the Metral connector is the inversion of the induced signal through the connector. The inductance of the connector is large enough to give the forward crosstalk an inverted pulse at this point. In an actual data transmission, the crosstalk concern would be at the input to the receiving transceiver. The slowing of the edge rates by the time they reach a receiver on another board will further reduce the magnitude of the crosstalk. *Figure 19* shows the signal at the same receiver input pin with no aggressors and with 8. The coupled voltage intrusion to the noise margin is 85 mV.



TL/F/11107–28

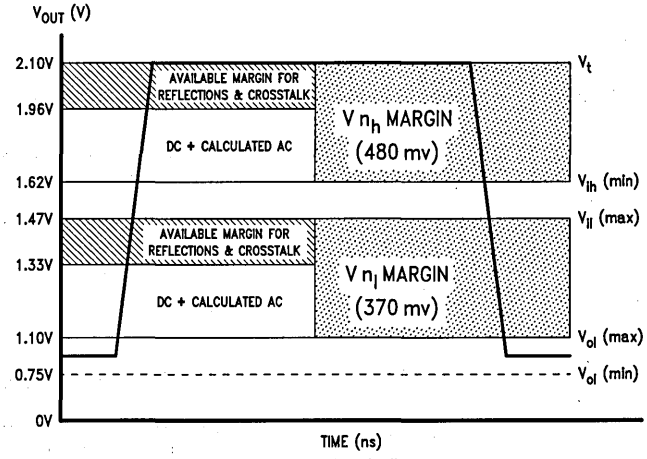
Rise 466.7 ps	Fall 435.0 ps	Min 590.000 mV	Measure-ments	Horz Mag 1 x Horz Pos Gr Opts	
Max 862.000 mV	Frequency 274.3 MHz		Statistics Comp & Def Continuous	Remove Wfm 2 ST014	Pan/ Zoom On

FIGURE 19. Crosstalk Voltage at Receiver with 8 Aggressor Lines Compared to 0 Aggressors



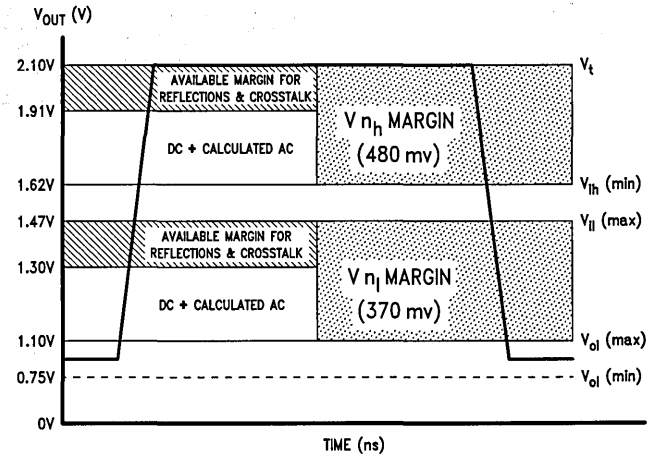
TL/F/11107–29

FIGURE 20a.



TL/F/11107–30

(all slots loaded)
FIGURE 20b.



TL/F/11107–31

slots 0 to 6 loaded
FIGURE 20c.

CONCLUSION

The Futurebus+ environment presents impedance mismatches to the high speed data signals. These circumstances make the measurement of the signals dependent on where they are measured. The fast edge rates of the signals have high frequency components that compose a large part of the waveform and can not be ignored. For these reasons, National Semiconductor uses 1000 MHz bandwidth test equipment, and specially designed low impedance test jigs for all of the data sheet specifications.

The placement of modules in a partially loaded backplane is crucial to the magnitude of the ringing. The equal distribution of the modules in the backplane appears to be the best condition for the lowest magnitude ringing.

Figure 20a shows the noise margins for BTL. *Figure 20b* shows the allocation of the noise margin for a fully loaded backplane and partially loaded in *Figure 20c* according to the Futurebus+ Expert Team. They have done extensive simulations that are reported in the Interim Report presented on September 14, 1990. This investigation of the reflections and crosstalk will be compared to their findings.

Figure 17 shows the combination of worst case crosstalk and reflections that were found in this investigation. The crosstalk added to the reflections to cut into the noise margin low by 100 mV. The 100 mV intrusion is deduced from the fact that the incident edge has a distinct edge at the 1.2V level. This is within the allowed 170 mV range in the Expert Team analysis of the partially loaded backplane, *Figure 20c*. This investigation also showed that the fully loaded backplane produced lower magnitude reflections than the partially loaded backplane. The measurements collected here support the Expert Teams allowance of only 140 mV for reflections and crosstalk in the fully loaded backplane, *Figure 20b*.

The National Semiconductor DS3886 Latched Data Transceiver maintained signal integrity in the Futurebus+ Backplane environment under severe operating conditions. Worst case situations of crosstalk, stub length and ground bounce combined with a transmission speed of 40 MBaud were used to test the DS3886 in real backplane operating conditions. The incident edge of the BTL signal consistently crossed the receiver threshold without a problem.

REFERENCES

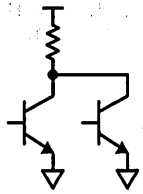
1. *The Multilayer Printed Circuit Board Handbook,* J.A. Scarlett, Electromechanical Publications, Ayer, Scotland, 1985.
2. *TDR Fundamentals,* Application Note 62, Hewlett Packard, April, 1988.
3. *FAST Applications Handbook,* National Semiconductor Corporation, 1987.
4. *Handbook of Printed Circuit Design, Manufacture, Components and Assembly,* Giovanni Leonida, Electromechanical Publications, Ayer, Scotland, 1981.
5. Interim Report of the Electrical Task Group Expert Team ... Futurebus+ Modeling and Noise Margin Results, Raytheon and DEC, Sept. 14, 1990.

Futurebus+ Wired-OR Glitch Effects and Filter

National Semiconductor
Application Note 744
Joel Martinez/Stephen Kempainen

INTRODUCTION

Futurebus+ addresses the needs of the high-end user who requires more bus performance than what has previously existed. In order to optimize bus performance, the backplane bandwidth has been increased to where the backplane line delays are in the same order of magnitude as the transfer periods. At this level, the backplane can no longer be treated as lumped loads but must be modeled as distributed loads which is in the realm of transmission lines. Designers must now deal with transmission line effects and be aware of glitches that occur when performing wired-OR functions. As the name implies, the wired-OR glitch occurs on lines that perform wired-OR logic. Wired-OR logic is implemented by connecting open-collector drivers in parallel and tying their collectors to a resistor pull-up (*Figure 1*). National Semiconductor's Futurebus+ Handshake Transceiver addresses this concern by incorporating a programmable low pass filter into the receiver. It provides optimum noise rejection while maintaining high bus through-put.



TL/F/11133–1

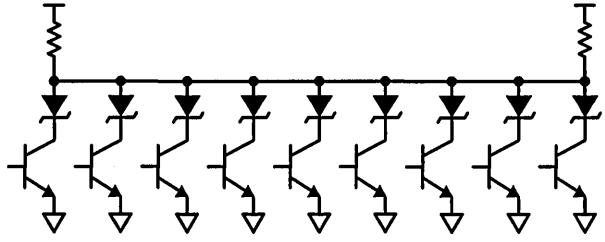
FIGURE 1

BTL (Backplane Transceiver Logic) is the driving technology behind the Futurebus+ physical layer. Driver outputs are open-collector with a series Schottky diode. The additional diode on BTL drivers isolates the normally large collector capacitance associated with the output transistor from the bus thus reducing bus loading. BTL is used on all the bus lines in a Futurebus+ backplane. Termination of the bus is done at both ends as shown in *Figure 2*. All Futurebus+ lines are connected in a wired-OR fashion, however, only a subset of these lines actually perform the wired-OR logic. These lines are the handshake, status, capability and arbitration lines. The critical timing lines used in handshaking must have wired-OR glitch filters to maintain signal integrity. The lines requiring glitch filtering are AK*, AI*, DK*, DI*, AP*, AQ*, AR*, and RE*.

In the wired-OR configuration, the glitch observed on the backplane is caused by the release of one or more drivers on the bus while others remain asserted. The resulting positive voltage pulse is the glitch characteristic that could cross the receiver threshold and degrade signal integrity. The transmission line effect, enhanced by the fast transition times and transmission line propagation delay, dictates how the wave reflections will affect the data signal. If the rise and fall times were longer than the line delays, the reflections would be included (shadowed) in these transition portions of the signal. Then the bus would not exhibit transmission line effects. However, the Futurebus+ transition times on the backplane can be one fourth to one fifth of the line delay. A transition at one end of the backplane takes some time before it reaches the other end and voltage levels will vary significantly, due to reflections, before equilibrium.

Low to High Transition of Open Collector Bus Driver

First the low to high transition of a single driver releasing the bus will be studied. The same characteristics are involved for the wired-OR glitch as will be seen later. Referring to *Figure 3*, what happens when Q1 releases?



TL/F/11133–2

FIGURE 2

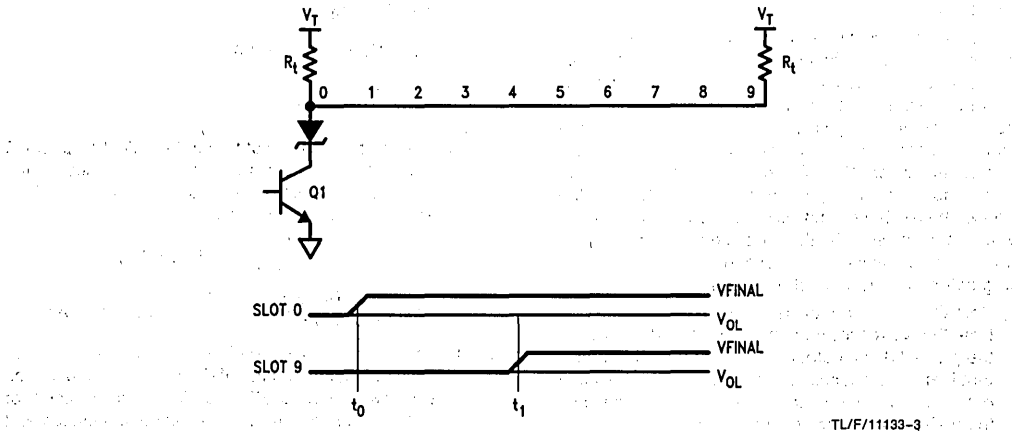


FIGURE 3

$t < t_0$; Q1 is on (asserted) and maintains a V_{OL} level on the bus.

$$V_{OL} = V_T * \frac{Zdriver}{Zdriver + (R_t/2)}$$

where Zdriver = driver output impedance

$t = t_0$; Q1 releases and a low to high wave front propagates from slot 0 towards slot 1. The current which was previously sunk by Q1 now gets injected back into the line. The driver sees an impedance of the termination resistance in parallel with the line impedance. The amplitude of the signal propagating down the line is described by the following equation:

$$VFINAL = V_{OL} + [I_{OL} * (R_t // Z_0')]$$

$$= V_{OL} + \frac{V_T - V_{OL}}{R_t/2} * (R_t // Z_0') \qquad (1.a)$$

For $R_t = Z_0'$, where Z_0' is the unloaded line impedance.

$$VFINAL = V_{OL} + V_T - V_{OL}$$
$$= V_T$$

If the driver was located in slots 1 to 8 then the equation would be;

$$VFINAL = V_{OL} + \frac{V_T - V_{OL}}{R_t/2} * (Z_0' // Z_0') \qquad (1.b)$$

$t = t_1$; The signal reaches slot 9. For $R_t = Z_0'$, all the energy is absorbed at slot 9 by the termination and the bus will be at equilibrium. For $R_t \neq Z_0'$ then secondary reflections will occur until the reflections settle.

Figure 4 shows the actual waveform from a 10 slot backplane with 1 in. pitch and 39Ω termination resistors. The 3 waveforms were obtained by probing the backplane at the solder points of the board connectors. In this case there were two boards inserted into the backplane, slots 0 and 9. The driver in slot 0 is switching while the other in slot 9 is off. The propagation delay of the line, tpd, is then defined as $t_1 - t_0$. This is plainly illustrated by the delay in the waveform probed at slot 0 and at slot 9. The tpd = 3 ns in this arrangement.

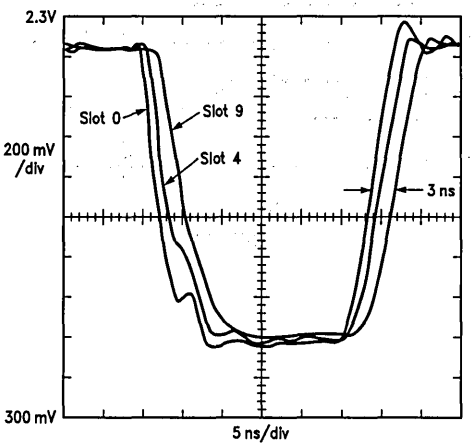


FIGURE 4

Wired-OR Glitch

As shown in *Figure 5*, things become more complicated by adding another driver at the opposite end of the bus. Initially, both drivers are asserted. Assuming they share the current from the termination equally, the V_O will be less than V_{OL} for a single driver. V_O is also influenced by the fact that the driver impedance increases with reduced current through the transistor. When Q1 releases, a glitch occurs at the drivers bus slot with a pulse-width equal to 2 times the line delay. The glitch amplitude is dependent on the amount of current that was sunk by the driver prior to releasing the line. So the more drivers that release the line simultaneously, the greater the amplitude of the glitch.

$t < t_0$ Both Q1 and Q9 are on keeping the bus voltage low. Note that V0 results from the resistor divider between the parallel driver output impedance and the parallel termination resistance. V2 is slightly higher because only one driver is on and the bus voltage is then just the resistor divider between a single driver output impedance and the two termination resistors in parallel.

$t = t_0$ Q1 turns off and transitions into a high impedance state, a low to high wave front propagates down the bus towards slot 9.

$$V1 = V0 + \frac{1}{2} * \frac{(V_T - V0)}{R_t/2} * R_t//Z_0' \qquad (2.1)$$

For $R_t = Z_0'$

$$V1 = V0 + \frac{1}{2}(V_T - V0)$$

$$= \frac{1}{2}(V_T + V0) \qquad (2.2)$$

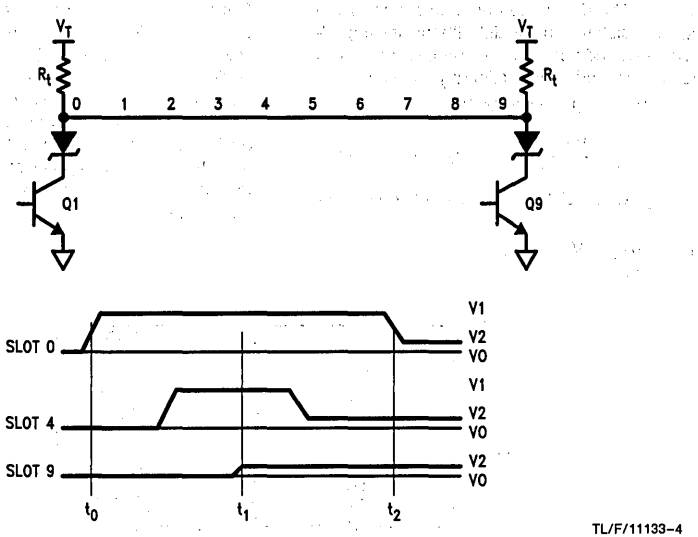
$t = t_1$ The glitch reaches slot 9 and the reflection at slot 9 is negative because the output impedance of the driver is small compared to the termination resistor and backplane impedance.

$$V2 = V1 + \frac{1}{2} * \frac{(V_T - V0)}{R_t/2} * R_t//Z_0' * \rho$$

where $\rho = \dfrac{R_t//Zdriver - Z_0'}{R_t//Zdriver + Z_0}$ (negative number)

$t = t_2$ The reflected wave reaches slot 0. It is important to point out that the output of Q1 is in a high impedance state when turned off. Therefore, the mismatch at slot 0 is only due to the backplane impedance and termination. For $R_t = Z_0'$, all the energy is absorbed at slot 0 by the termination and the bus will be at equilibrium. When $R_t \neq Z_0'$, then secondary reflections will occur and will continue until the reflections settle to the termination voltage.

The pulse width at the end of the backplane where the driver is still asserted is as wide as the propagation delay of the bus. Since the reflection must return to the released driver end, the pulse width there will be 2 tpd, where tpd = $(t_1 - t_0) = (t_2 - t_1)$.



TL/F/11133–4

FIGURE 5

Wired-OR Glitch Calculation for Futurebus+

To calculate the wired-OR glitch for a Futurebus+ backplane the following assumptions are used;

Z_0'	60Ω	Unloaded Backplane with Connector and Vias
Z_0''	31Ω	Fully Loaded Backplane
R_t	33Ω	Termination Resistor
$R_t//Z_0'$	21Ω	
$R_t//Z_0''$	16.0Ω	
$R_t//Zdriver$	4.3Ω	
Zdriver	5Ω	Driver Series Resistance
Vdriver	0.7V	Driver on Voltage in Series with the Resistance
V_T	2.1V	Termination Voltage
V_{TH}	1.47V to 1.62V	Receiver Input Threshold
tpd	1.8 ns	Unloaded, 10 Slot by 1 in. Pitch Backplane
	4.4 ns	Same Backplane Fully Loaded

For drivers at each end *(Figure 5)* and assuming an Unloaded condition we have;

$$V0 = Vdriver - (V_T - Vdriver) * \frac{\frac{1}{2}Zdriver}{\frac{1}{2}Zdriver + \frac{1}{2}R_t}$$

$$V0 = 0.52V$$

From equation (2.1)

$$V1 = V0 + \frac{1}{2} * \frac{(V_T - V0)}{R_t/2} * R_t//Z_0$$

$$V1 = 1.3V$$

When the reflected wave hits the mismatch at slot 9, a negative reflection adds to V1.

$$V2 = V1 + \rho^1 * V_x{}^1 \qquad \rho^1 = \frac{R_t//Zdriver - Z_0'}{R_t//Zdriver + Z_0'}$$

$$= 0.62V \qquad Vx^1 = \frac{1}{2} * \frac{(V_T - V0)}{R_t/2} * R_t//Z_0'$$

When this wave front reaches slot 0, another negative reflection adds to V2.

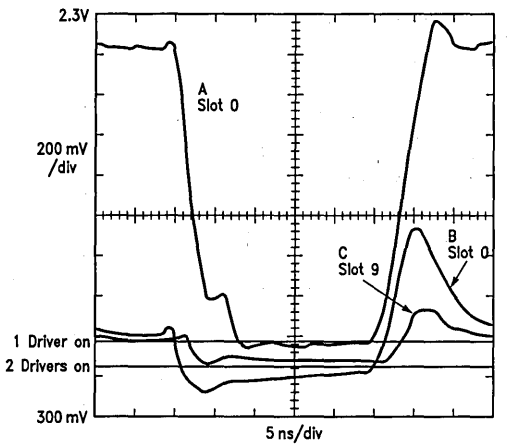
$$V3 = V2 + \rho^2 * Vx^2 \qquad \rho^2 = \frac{R_t - Z_0'}{R_t + Z_0'}$$

$$= 0.78V \qquad Vx^2 = Vx^1 * \rho^2$$

Subsequent reflections will be smaller and smaller until equilibrium is reached on the line.

The glitch has a maximum amplitude of 1.3V for two drivers sharing the bus current equally, which is below the receiver threshold of 1.47V. This isn't enough amplitude to false trigger the receiver and thus data corruption will not occur. The glitch also has a maximum pulse width equal to 2 tpd or 3.6 ns. If Q1 was sinking most of the current, then the resulting glitch will have a greater amplitude than that calculated, but the pulse width will remain the same. The greater amplitude may cross the receiver threshold and cause false triggering. The purpose of the glitch filter on the DS3884 is to filter any glitch that crosses the receiver threshold and thereby prevent false triggering. Since the glitch width is independent of amplitude, a filter can be set to reject certain duration glitches.

Figure 6 shows actual waveforms from the 10 slot backplane. The drivers are mounted in the end slots and R_t = 39Ω. The Vdriver (1 driver on) level is 685 mV and VO (2 drivers on) is 545 mV. The switching driver is shown without the other driver asserted, waveform A in *Figure 6*. Then the wired-OR glitch is shown at the end slots with one driver switching and the other holding the asserted level, waveforms B and C. As predicted by the model, the amplitude and pulse width of the glitch is greatest at the driver slot that is releasing the line. The maximum amplitude of the glitch is 1.25V. The propagation delay of the backplane with 2 boards inserted was shown in *Figure 4* to be 3 ns. The model predicted the pulse width to be twice this delay, but it is shown to be 2.33 times the delay at 50% of the amplitude. This is because the model does not take into account the



TL/F/11133–6

FIGURE 6

wave dispersion effects of the backplane which filter the high frequency components and round the signal corners. Since the glitch amplitude will never equal V_T as indicated by equation (2.1), the glitch pulse width at the receiver threshold will always be less than the 50% amplitude pulse width. Two times the measured backplane propagation delay is the worst case width that will be seen at the receiver threshold.

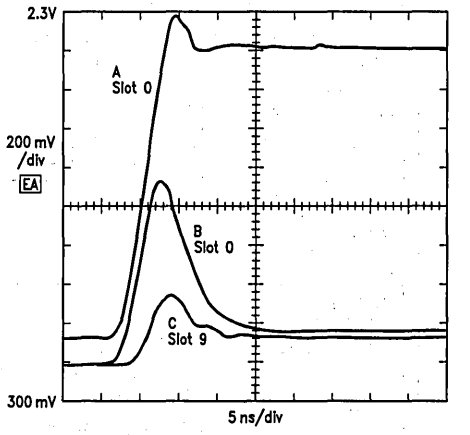
Theory states that glitch width is only dependent on the bus propagation delay. *Figure 7* and *8* are included to demonstrate this theory in the backplane application. *Figure 7* shows the glitch for a 1 MHz signal. The pulse width is the same as that in *Figure 6* which has a 20 MHz signal. *Figure 8* is the same situation as *Figure 6* except that the backplane slots between 0 and 9 are loaded with 10 pF to 12 pF each. This increases the propagation delay of the backplane. The width of the pulse is 10.7 ns. Using the fully loaded backplane tpd of 4.4 ns, this pulse width is 2.43 times the propagation delay. This figure correlates well with the previous partially loaded glitch width. The amplitude is noticeably less when the backplane is fully loaded due to the decreased impedance seen by the driver.

The simultaneous release of the bus line by more than one driver at a time will increase the amplitude of the glitch but the width still depends on the propagation delay of the bus. The current of more than one driver injected into the line directly affects the amplitude of the pulse. *Figure 9* shows the glitch when 2 drivers release the line simultaneously and 1 holds the line asserted. The releasing drivers are in slots 0 and 3, and the hold driver is in slot 9. The glitch amplitude of 1.81V will cause a false trigger of receivers on the line. The pulse width of the glitch at the receiver threshold low is 6 ns, about 2 tpd. The 50% of amplitude pulse width is slightly larger than the case of 2 drivers on the bus, but this is explained by the additional load capacitance of the third driver which increases the line delay.

WIRED-OR FILTER

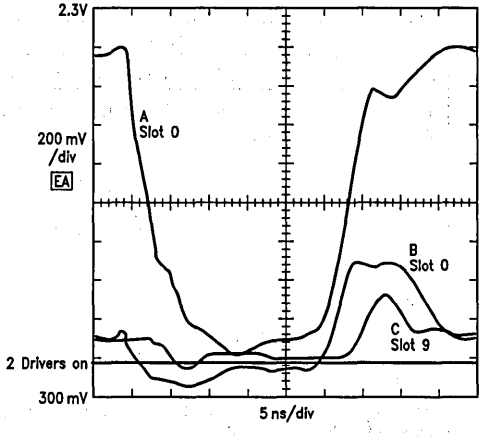
The worst case glitch that can occur on the bus will have a pulse width equal to the round trip delay of the backplane. The filter must reject glitch pulses which are equal to and less than the worst case round trip bus delay, 2 tpd. Since the line delay is dependent on the length and the loading of the backplane, the simplest way to determine the worst case delay is by measuring the line delay of a fully populated backplane. Once the worst case delay is determined, the receivers should provide a filter that will reject the glitch and allow signals to pass through with minimum delay.

The DS3884 Handshake Transceiver which is part of National's Futurebus+ chip set has a built in glitch filter. The filter can be tailored to a specific backplane delay to minimize timing delays. Four different settings—(5 ns, 10 ns, 15 ns and 25 ns) are available via the pulse select pins—PS1 and PS2. These settings can be fine tuned by varying the resistor to ground at pin REXT. For example, if the round trip delay of the backplane is 8 ns, then the filter setting should be rounded-off to the higher setting which is 10 ns.



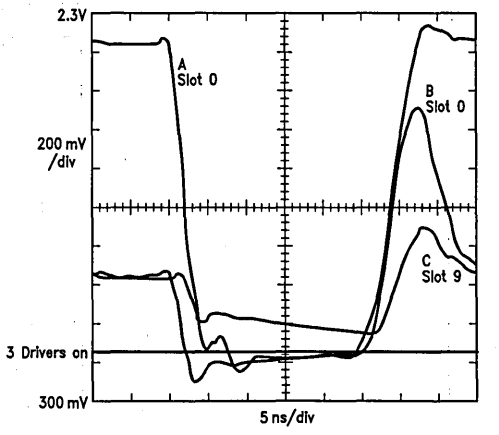
TL/F/11133–7

FIGURE 7



TL/F/11133–8

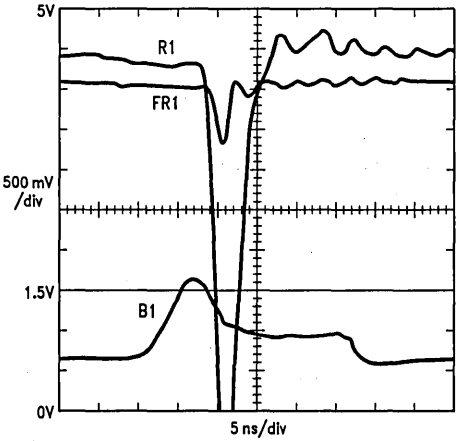
FIGURE 8



TL/F/11133–9

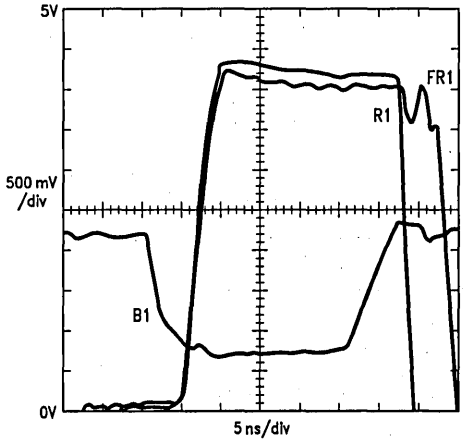
FIGURE 9

This setting will reject glitches with pulse widths of 10 ns and less. The maximum propagation delay from the transceiver bus input, Bn, to the filtered receiver output, FRn, for the receiver high to low transition will be 22.5 ns at this setting. *Figure 10* shows three waveforms from the DS3884 operating in slot 0 of the same backplane configuration as *Figure 9*. Waveform B1 results from the same 2 drivers releasing the line simultaneously as Waveform B slot 0 in *Figure 9*. The lower signal amplitude is due to probing directly at the receiver input pin rather than the backplane solder point as in *Figure 9*. R1 and FR1 are the waveforms from DS3884, channel 1, for the receiver and filtered receiver outputs respectively. PS1 and PS2 are both set to 0V to obtain a glitch rejection of 5 ns and less. Waveform FR1 clearly rejects the false triggering of the glitch, which is about 3 ns wide at the 1.5V receiver threshold. *Figure 11* shows the propagation delays for B1 to R1 and FR1 for both the high to low and low to high transitions at the 5 ns glitch rejection setting. The t_{PHL} is extended from 5 ns to 10 ns for the filtered output. The ringing on the receiver output signals results from wire wrap board frequency response limits.



TL/F/11133–10

FIGURE 10



TL/F/11133–11

FIGURE 11

The wired-OR glitch is inherent in wired-OR busses and will exist in a high speed transmission line environment. However, it is comforting to know that it is a well understood phenomena and there are ways of maintaining signal integrity through the use of filters.

DS3885 Arbitration Transceiver

National Semiconductor
Application Note 742
Chai Vaidya, IPG Applications

INTRODUCTION

Today's digital systems with their higher clock speeds and data throughput, must have higher transfer rates to keep the processors running at top speed. With these types of circuits, it is essential to have high performance bus transceivers which tie together everything on the bus. Without specialized bus interface transceivers, board designers must dedicate a substantial amount of pc-board real estate to the bus interface. Typically, these devices implemented with discrete ICs or PALs can take up to 20% to 30% of the available real estate. Also, interrupt structure, arbitration scheme, and burst-data transfer features result in added interface complexity.

The trend toward distributed intelligence increases the need to place more functions on a single board. Therefore, interface ICs play an important role, especially in the design of boards for high performance buses such as Futurebus+. The purpose of this Application Note is to highlight the important features of the DS3885 Arbitration Transceiver.

DS3885 ARBITRATION TRANSCEIVER

The DS3885 Arbitration Transceiver is one of the 5 devices in the chipset for Futurebus+. Other devices include; the DS3883 BTL 9-bit Data Transceiver, DS3884 Handshake Transceiver, DS3886 BTL Latching Transceiver, and the DS3875 Arbitration Controller, which supports all modes of the Futurebus+ distributed arbitration protocol *(Figure 1)*.

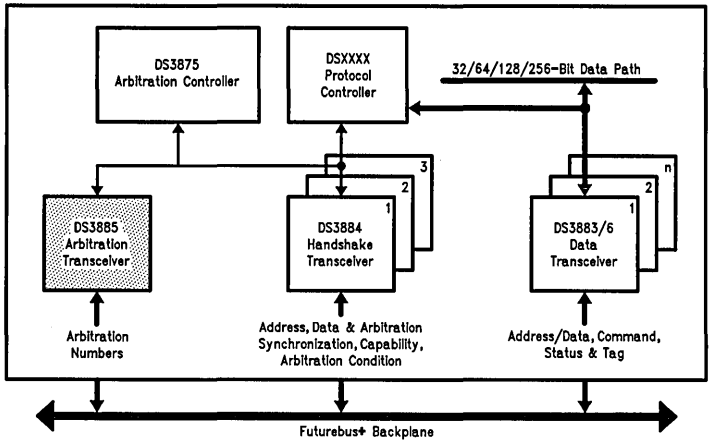
The DS3885 Arbitration transceiver conforms to the IEEE P896 Futurebus+ specifications for the distributed arbitration. However, it can also be used to support messaging function in the central arbitration scheme. It allows several modules to compete using the same distributed arbitration bus. The DS3885 integrates this function within a bus transceiver to reduce part count and competition delay.

The DS3885 supports live insertion as required by the Futurebus+ modules. Futurebus+ specifications provide support for those applications requiring dynamic reconfiguration and fault tolerance. Thus, the Futurebus+ protocol supports the insertion or replacement of any module on the bus without the need for power-down on the backplane. However, it is expected that the system software must support diagnostics to determine if any module is faulty. If a faulty module is detected, the software should disable this module and pass this information to the maintenance logic. The pin LI (Live Insertion) of this device should be connected to the live insertion power connector. If the live insertion is not supported, LI (pin 2) should be tied to V_{CC}.

As system bus such as Futurebus+ gets faster and wider, the interfacing transceivers must supply clean signals with controlled rise and fall times, and a very low skew. The minimum and maximum rise and fall times of the on-chip drivers on the DS3885 are 2 ns and 4 ns respectively. These controlled rise and fall times help reduce crosstalk on the bus. In order to reduce the loading on the bus, a Schottky diode is added in series with an open-collector driver output. Due to this, the capacitance of the driver transistor is isolated by the small reverse-biased capacitance of the Schottky diode in the non-transmitting state. With the Schottky diode capacitance of 2 pF, the total loading capacitance of this device is under 5 pF.

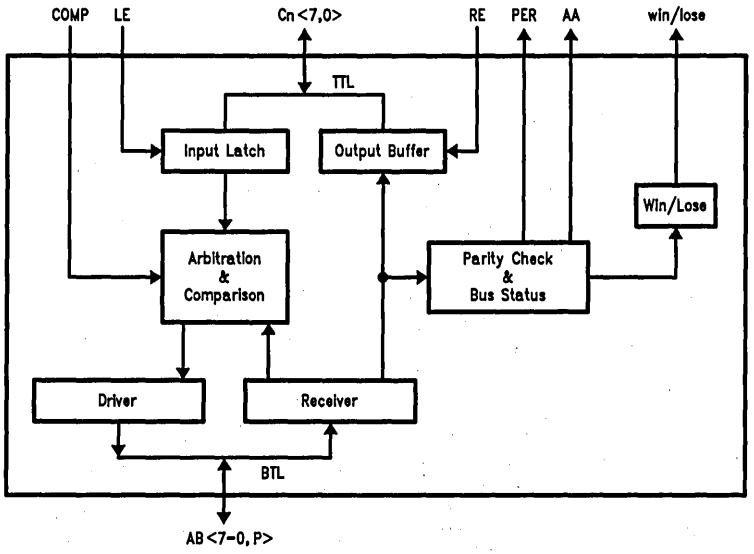
Figure 2a and *2b* are block and logic diagrams of the DS3885 device. It has an input latch for latching the arbitration number and a driver to place this number on the bus backplane. The DS3885 generates win/lose signal depending on the outcome of the competition. The worst case delay from COMPETE to WIN is 75 ns maximum. The WIN/LOSE signal can also be used by any module not participating in the competition to determine whether its competition number is greater than the number of the winning module. When all signals AB0 to AB7 are asserted the device will inform the Arbitration Controller of this condition by asserting the signal AA.

In addition to receiving and transmitting signals on Futurebus, the device contains arbitration comparison and parity checking logic. By combining the BTL and the arbitration logic, the DS3885 device reduces the logic propagation delay for arbitration competition.



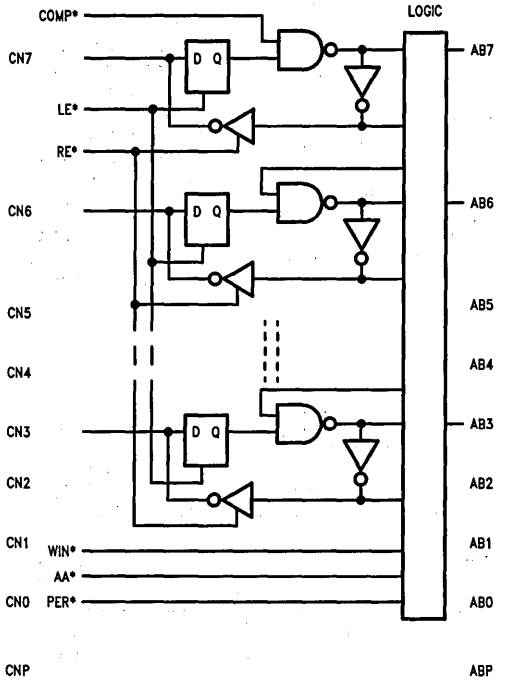
TL/F/11130–9

FIGURE 1. IEEE 896 Compatible Futurebus+ Chipset



TL/F/11130–10

FIGURE 2a. Block Diagram of DS3885



TL/F/11130–1

FIGURE 2b. Logic Diagram of DS3885

The following section describes the mechanism involved in the process of arbitration on the Futurebus+. It may be useful for those designing systems using National's Futurebus+ chipset including the Arbitration Controller.

FUTUREBUS+ ARBITRATION

Futurebus+ provides a distributed arbitration scheme to select mastership of the bus for one module at a time. The equivalent parallel contention logic (shown in *Figure 3*) is used to determine which module has the highest arbitration number. This arbitration competition for control of the bus takes place in parallel with the data transmission process. Each module is assigned a unique arbitration number, which is used to resolve the arbitration competition.

The Futurebus+ arbitration protocol uses 14 lines on the backplane. Due to the limitations of the number of bus lines used for arbitration, some competition numbers require two passes of the control acquisition cycle. Each pass of the control acquisition cycle drives half of the given arbitration number. In addition to the 9-bit arbitration bus, there are three protocol handshake lines and two arbitration condition lines which define the protocol of the operation of the arbitration protocol. The three handshake lines are; AP*, AQ* and AR*. These lines control the arbitration process, in which each module competes, checks for errors and in case of the winner, wait until the current master has finished its tenure. They allow the modules to step through the various phases of arbitration, and adapt the speed of the protocol to the slowest participating module. These three handshake signals make use of wired-OR characteristic of BTL to provide six phases that can be used to describe the arbitration sequence. There are two additional condition signals; AC0* and AC1*. AC0* is used to signal arbitration cycle errors. This helps to determine the arbitration settling time more accurately. AC1* can be used by any module to prevent the exchange of bus mastership on the current arbitration cycle.

The DP3885 receives the arbitration number from the module and compares it to the number on the bus to determine if the module has won bus tenure. After a certain settling time, the module with the highest competition number will find that its number matches the number remaining on the bus; therefore, this module is a winner.

The actual arbitration competition takes place on a 9-bit arbitration bus. Each competing Futurebus+ module latches its competition number into its DS3885 Transceiver. All modules apply this number through the parallel contention arbitration logic onto the open collector arbitration lines, AB(7 . . . 0,P), where P is the odd parity bit (LSB). The arbitration logic on the module senses the resulting wired-OR levels on the arbitration bus, and modifies the number it is applying to the bus, according to the following rule:

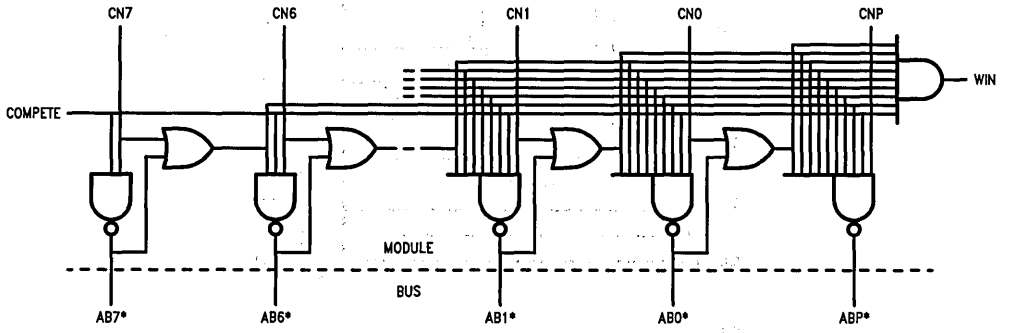
If, for any bit of a module's competition number, the corresponding arbitration line shows a greater value, all lower order bits of the competition number are withdrawn from the bus.

Once the DP3885 has received the arbitration competition number (cn) from the Arbitration Controller, the controller latches these numbers into the transceiver along with the corresonding parity bit. A low voltage on the COMP (Competition) signal indicates that the module is competing in the arbitration. The value of each bit is compared one bit at a time. As shown in the *Figure 4*, at first, the first two bits are compared and, in this example, they are found equal. The next two bits are compared. The value of the bit 2 for module A is "1". The value of bit 2 of module B is "0". Therefore, module B will pull back its bit, and subsequently all lower bits; hence, it has lost the competition. Module A can now place its bit on the bus. This process continues until all bits are checked. Notice that the value of the bit 4 (Cn4) for module B is "1", but this bit will not be placed as "0" on its output (AB4), since module A has won the bus.

The Futurebus+ standard requires that all modules place their worst-case internal arbitration logic delay from any bus line ABx* to the adjacent line, AB(x−1)* or AP* into a module status register. This, combined with the same numbers from other modules, and the propagation delay of the bus, can be used to calculate a worst-case competition settling time. The equation used is:

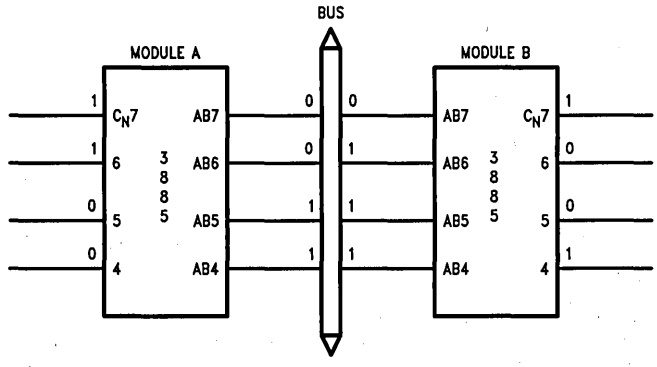
$$10 \times t_{pd} + 5 \times t_{ext} + 4 \times t_{int}$$

where t_{pd} = bus propagation delay
t_{int} = Module's delay (AB<n> to AB<n−1>)
t_{ext} = max. external module delay (if any external parts used).



TL/F/11130–2

FIGURE 3. Equivalent Parallel Contention Logic



TL/F/11130–3

FIGURE 4. Compete Logic

ARBITRATION COMPETITION SETTLING TIME

Although the worst-case settling time can be calculated with a single equation, this gives a very long delay which is unnecessary for most cases. Instead, a more detailed equation can be generated which gives a more accurate settling time.

The settling time is the period that starts when the indication to begin arbitration is issued. It ends when the winner's competition number is valid on the arbitration bus as well as within the winner's arbitration logic. This time is a function of the following three factors:

1. Length of the backplane
2. Module's competition number
3. The worst-case transceiver and logic delays of both the internal arbitration logic of the module and that of its competitors.

The interactions among the various competitor's arbitration logic can be extremely complex. In most cases the winning arbitration number will be selected quickly. However, the worst-case resolution time must be calculated in order to guarantee that a valid win signal is always sampled by the module. In the event where the system has prior knowledge of the set of competition numbers in use by the system at any one time, it may be possible to calculate a lower delay value.

For any competition number, the individual elements of the worst-case delay can be isolated into the following five basic parts; *Figures 5* and *6* show these delay components. The terminology is explained below in Table I:

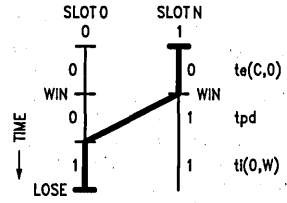
ti(x,y) is the worst-case delay from signal x to signal y of the internal logic of the module involved in the calculations. te(x,y) is the same worst-case delay for all possible external competitors. Within the brackets, "C" and "W" represent the "compete" signal and the "win" signal respectively. The letter "P" represents the signal ABP* (parity). The number "n" represents the signal "ABn". To begin the competition, all modules are required to assert "compete" before releasing the handshake signal AR*. The minimum skew between the assertion of "compete" and the release of AR* can be subtracted from all of the delays, and it is given by (C,AR*). Similarly, a module signals that it has won the competition by asserting AQ*. The minimum delay from the recognition of the "win" condition to the assertion of AQ* can be subtracted from some of delays and is indicated by (W,AQ*). Finally tpd represents the propagation delay of signals on the bus from one competing module to another.

TABLE 1. Delay Component Terminology

Delay Component	Description
ti(C,4)	Internal Delay from Compete Asserted to Change on ab4*
ti(C,ar*)	Internal Delay from Compete Asserted to Release of ar*
te(6,2)	External Delay from Change on AB6* to Change on ab2*
te(5,[43]) + te([43],2)	Worst-case of te(5,4) + te(4,2) and te(5,3) + (3,2)
te(2,P)	External Delay from Change on AB2* to Change on abp*
ti(4,W)	Internal Delay from Change on AB4* to Change on win
ti(W,aq*)	Internal Delay from Valid win to Assertion of aq*

COMPETE DELAY

Assume that the two modules are competing with the numbers "0" and "1" at opposite ends of the backplane as shown in *Figure 5a*. Before the first module finds out that it has lost the competition, its competitor must assert its competition number on the bus; this is te(C,0). Then the signal must propagate across the backplane, which is tpd, and the result must reach the internal "win" signal; this is ti(0,W). However, since the module will not begin timing the competition until it has actually received the release of the signal AR* by its competitor. Therefore, it can subtract its competitor's minimum competition skew, te(C,AR*), and the propagation delay of AR* across the bus, which is tpd. Thus, the total delay for the first module is given by te(C,0) + tpd + ti(0,W) − te(C,AR*) − tpd. This external compete logic delay minus the minimum compete-to-AR* delay of the external module, is known as the competition delay. It occurs whenever the most significant bit of the competition number is "0". It adds to the worst case delay only if there are no other delays. (Note that it must always be less than or equal to zero if there is only one leading zero in the competition number.)



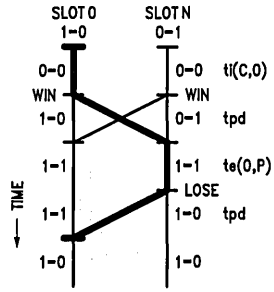
TL/F/11130–4

te(C,0) + tpd + ti(0,W) − te(C,**AR***) − tpd

FIGURE 5a. Compete Delay

PARITY DELAY

Assume that two modules are competing with the numbers "1" and "0" at the opposite ends of the backplane as shown in *Figure 5b*. The first module has a parity bit of "0" and the second module has a parity bit of "1". The first module finds out that it has won the competition, immediately; i.e., te(C,0). However, parity will not be immediately correct on the bus until its number propagates across the bus, this delay is tpd. Its competitor can respond by withdrawing its parity bit te(0,P), and the release propagates back across the bus, which is tpd. The parity must be valid at the first module before it can assert AQ*. The total delay for the first module is given by ti(C0) + tpd + te(0,P) + tpd − ti(W,AQ*). An external parity logic and round-trip propagation delay, minus the minimum win-to-AQ* delay of the module, is known as the parity delay. It occurs whenever the least significant bit of the competition number is "1" and the parity bit is "0". The worst-case delay cannot include both a defeat or compete delay and a parity delay.



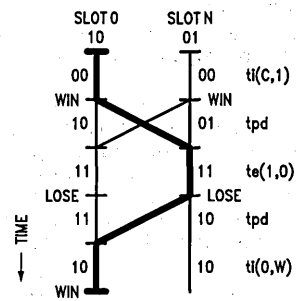
TL/F/11130–5

ti(C,0) + tpd + te(0,P) + tpd − ti(W,**AQ***)

FIGURE 5b. Parity Delay

ITERATE DELAY

Let us assume that two modules are competing with the numbers "10" and "01" at opposite ends of the backplane *(Figure 5c)*. In this case, the first module asserts its competition number on the bus, ti(C,0). The most significant bit must propagate through across the bus, tpd. Its competitor must release its least significant bit, te(1,0). The release must then propagate back across the bus, tpd and the module must recognize that it has won, ti(0,W). The total delay for the first module is given by ti(C,1) + tpd + te(1,0) + tpd + ti(0,W). An internal logic delay is known as an iterate delay. Iterate delay occurs wherever the pattern "10" appears in a module's competition number.



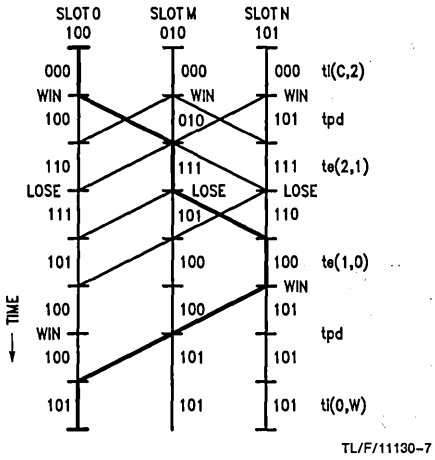
TL/F/11130–6

ti(C,1) + tpd + te(1,0) + tpd + ti(0,W)

FIGURE 5c. Iterate Delay

DEFEAT DELAY

As shown in *Figure 6a*, assume that three modules are competing with the numbers "100", "010" and "101", with the first one located at the opposite end of the backplane from the other two. Here, the first module asserts the most significant bit of its competition number on the bus. This causes the second competitor to release its middle bit. This in turn allows the third competitor to assert its least significant bit, which causes the first module to lose the competition. *Figure 6a* shows the total delay for the first module given by ti(C,2) + tpd + te(1,0) + tpd + ti(0,W). An extra defeat-causing external logic delay is known as a defeat delay. This delay occurs wherever the pattern "100" appears in module's competition number. It adds to the worst-case only if it is part of the last iterate delay of the competition number.



TL/F/11130–7

ti(C,2) + tpd + te(2,1) + te(1,0) + tpd + ti(0,W)

FIGURE 6a. Defeat Delay

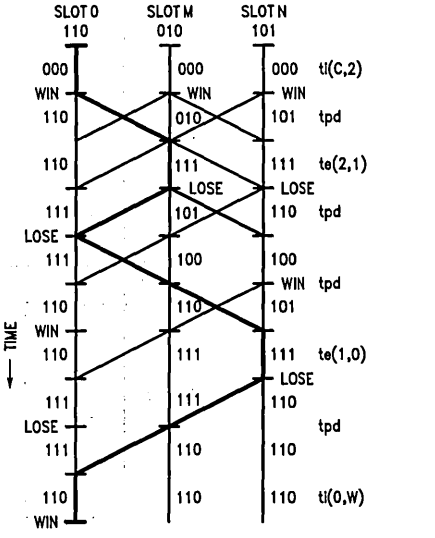
GLITCH DELAY

Let us now assume that three modules are competing with the numbers "110", "010" and "101", with the first one at the opposite end of the backplane from the other two (see *Figure 6b*). The first module asserts the most significant two bits of its competition number on the bus. This causes the second competitor to release its middle bit. Now in a transmission line environment, the current that was flowing through this bit crosses the receiver threshold of other modules. This glitch will propagate to the first module, where the driver will adjust the amount of sink current to bring back the signal to its correct voltage level. In the mean time, for the third module, it appears as if it has won the competition. This module will not release its least significant bit until the correct voltage has propagated across the bus. The third module will then allow the first module to win the competition. The total delay for the first module is given by ti(C,2) + tpd + te(2,1) + 2tpd + te(1,0) + tpd + ti(0,W). An extra wireOR-caused external logic and round-trip propagation delay is known as a glitch delay. It occurs wherever the pattern "110" appears in a module's competition number.

It is important to note that the above listed delays can be added to each other. However, the compete delay cannot combine with any other delay. The parity delay cannot coincide with a defeat delay. The defeat delay will add to the worst-case settling time only if it is part of the last iterate delay of the competition number.

For example, a competition number with the shortest settling time is "11111111-1". From the assertion of "complete" to the detection of "win", the total time is ti(C,W), or the maximum propagation delay of the parallel contention logic. The minimum internal skew, ti(C,AR*), can then be subtracted from this. A competition number with the longest settling time is "11010110-0". This number has three iterate and two glitch delays. The worst-case settling time is:

ti(C,7) − ti(C,AR*) + te(7,6) + te(6,5) + ti(5,4) + te(4,3) + ti(3,2) + te(2,1) + te(1,0) + ti(0,W) + 10tpd.



TL/F/11130–8

ti(C,2) + tpd + te(2,1) + 2tpd + te(1,0) + tpd − ti(0,W)

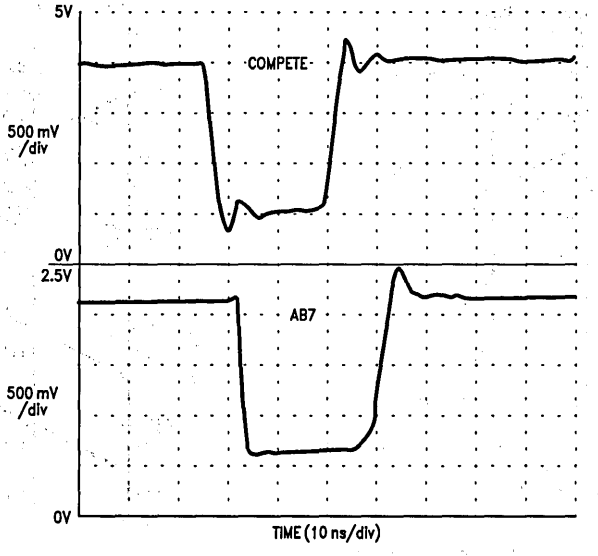
FIGURE 6b. Glitch Delay

The minimum and maximum propagation delays from the AB0 to "WIN" signal asserted are 6 ns and 28 ns respectively for the "win" or "greater than" conditions. The minimum and maximum propagation delays from the ABP to PER are 6 ns and 28 ns respectively.

Figures 7 through *9* show output waveforms of the DS3885 under various test conditions. The DS3885 device was mounted on a wire-wrapped board which was placed in the slot 1 of the Futurebus+ backplane. The supply voltage was 5V. As shown in *Figure 7*, the propagation delay from the time COMPETE goes low (active) to AB7 is 6.135 ns. The rise and fall times of the output signal AB7 are 4.775 ns and 1.384 ns respectively. *Figure 8* shows the WIN condition which is simulated by connecting the six competition bits inputs Cn<7,1> to ground and providing a pulse to the competition bit input Cn<0>. The load on the output comprised of a 33 pF capacitor in parallel with 1 kΩ resistor. However, the total load capacitance including the probe, wiring etc., is approximately 50 pF. The propagation delay from AB0 to WIN, in this case is 14.65 ns. As can be seen, even with faster rise and fall times, the DS3885 generates noise-free output signal with BTL voltage levels. *Figure 9* shows propagation delay from AB7 to ABP (parity) which is 36.91 ns.

SUMMARY

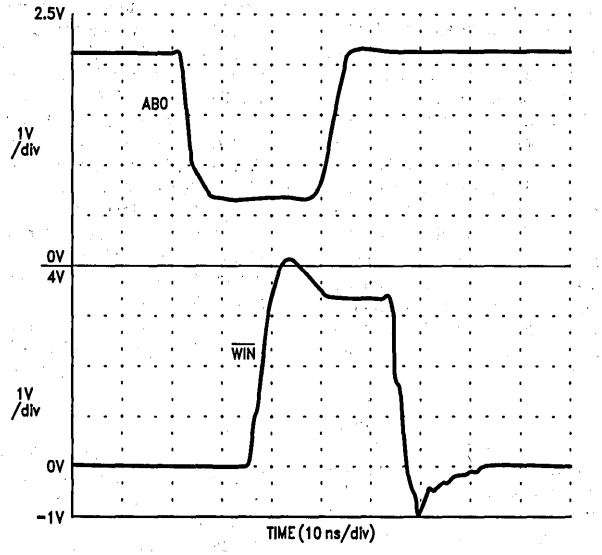
Like all other transceivers in the Futurebus+ chipset, the DS3885 also contains glitch-free power up-down feature, on-chip bandgap reference for the receiver and has controlled rise and fall times. It contributes as a valuable device for resolving the arbitration competition process on the Futurebus+. Although it has been designed to support the Futurebus+ arbitration process, it can also be used to support the messaging function in the central arbitration scheme.



TL/F/11130–11

Prop Delay	Rise Time	Fall Time
6.135 ns	4.775 ns	1.384 ns

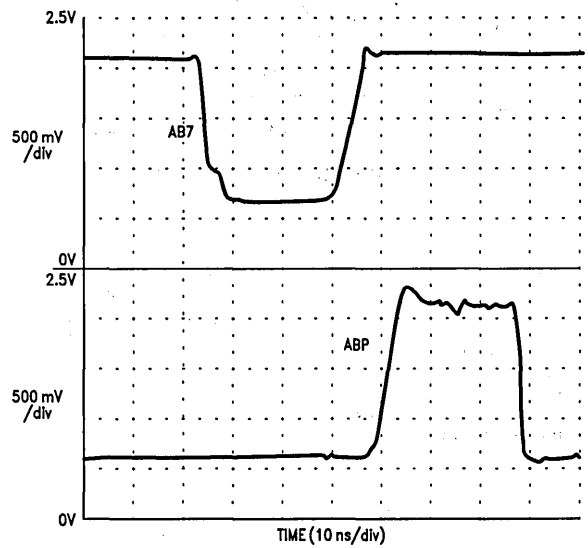
FIGURE 7



TL/F/11130–12

Prop Delay	Rise Time	Fall Time
14.65 ns	3.718 ns	2.683 ns

FIGURE 8

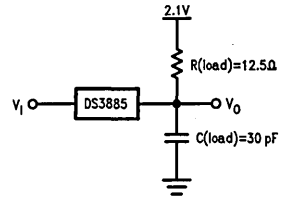


TL/F/11130–23

Prop Delay	Rise Time	Fall Time
36.91 ns	4.162 ns	1.459 ns

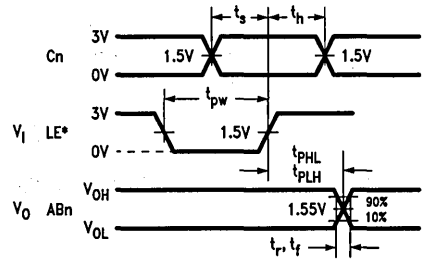
FIGURE 9

Figures 10 through *19* show test circuits and timing diagrams used to measure various parameters of this device.



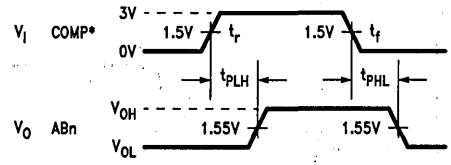
TL/F/11130–13

FIGURE 10. Driver Propagation Delay Set-up



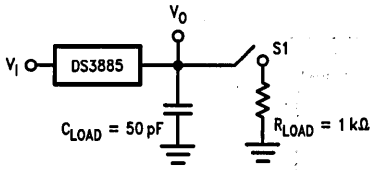
TL/F/11130–14

FIGURE 11. Driver: LE* to AB7, t_s, t_h, t_{pw}



TL/F/11130–15

FIGURE 12. Driver: COMP* to AB7

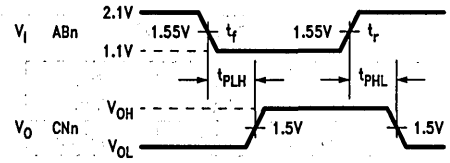


Switch Position

	t_{PLH}	t_{PHL}
S1	open	close

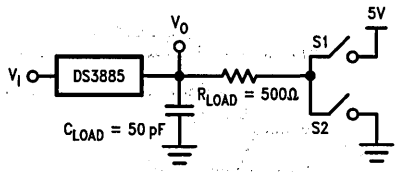
TL/F/11130–16

FIGURE 13. Receiver Propagation Delay Set-Up



TL/F/11130–17

FIGURE 14. Receiver: ABn to CNn

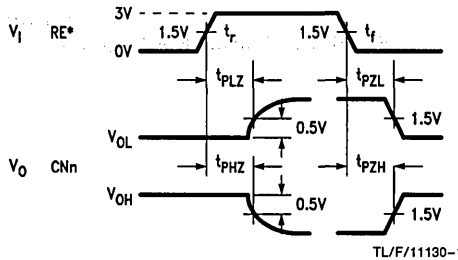


Switch Position

	t_{PZL} t_{PLZ}	t_{PZH} t_{PHZ}
S1	close	open
S2	open	close

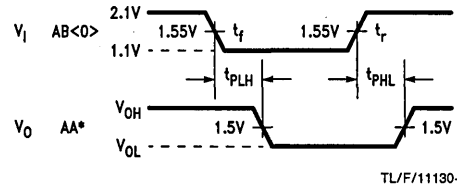
TL/F/11130–18

FIGURE 15. Receiver Enable/Disable Set-Up



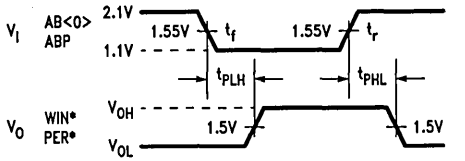
TL/F/11130–19

FIGURE 16. Receiver: RE* to CNn



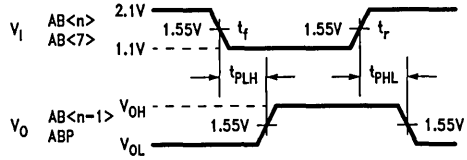
TL/F/11130–20

FIGURE 17. AB0 to AA*



TL/F/11130–21

FIGURE 18. AB0 to WIN*, ABP to PER*



TL/F/11130–22

FIGURE 19. ABn to AB<n−1>, AB7 to ABP

Section 3

Futurebus+ Application Notes

Section 3 Contents

National Semiconductor

What is Futurebus+?

Paul Borrill

ABSTRACT

Futurebus+ is a specification for a scalable (32/64/128 or 256-bit wide) bus architecture. Arbitration is provided by a fully distributed, one or two pass, parallel contention arbiter with allocation rules to suit the needs of both real-time (priority based), and fairness (equal opportunity access based) configurations. Two transmission methods are provided: (1) a technology-independent, compelled protocol, supporting broadcast, broadcall and transfer intervention (the minimum requirement for all Futurebus+ Systems), and (2) a configurable transfer-rate source-synchronized protocol supporting only block transfers and broadcast for the maximum performance systems. Futurebus+ takes its name from its goal of being capable of the highest possible transfer rate consistent with the technology available at the time modules are designed, while ensuring compatibility with all other modules designed to this standard before *and after*. The plus sign (+) refers to the extensible nature of the specification, and the hooks provided to allow further evolution to meet unanticipated needs of specific application architectures. This paper describes the history, structure and applications of the Futurebus+ architecture.

HISTORICAL PERSPECTIVE

Futurebus+ is a revised and substantially extended version of the original IEEE 896.1—1987 Futurebus standard, where the basic protocols and facilities of the new Futurebus+ were developed.

From 1983 to 1987 the IEEE P896 Futurebus Working Group, provided a forum for leading experts to develop innovative technology and protocols for a scalable performance multiprocessor system bus. The most recent Futurebus+ efforts, from 1988 onwards, represented a commercial consolidation of all the basic Futurebus philosophies into a realizable and practical standard as an industry consensus developed between the major organizations who became interested in developing products based on Futurebus+.

During early 1988, the VME International Trade Association (VITA) saw the need to develop a strategy which would lead to the definition of a Next Generation Architecture Bus Standard, to follow the widely successful IEEE 1014: VMEBus standard. They developed a set of requirements which included openness, performance goals, and system facilities and flexibility which would not hinder systems using this bus for many generations of computer systems. In December 1988, VITA formally announced its intention to base its "VME Futurebus+ Extended Architecture" (VFEA), on a revised and extended IEEE 896.1—1987 Standard, in conjunction with the IEEE Futurebus+ Working Group.

Around the same time, the US Navy's Next Generation Computer Resources Program (NGCR) chartered a Backplane Working Group to develop a set of requirements for use in future Mission Critical Computers. A principal requirement was that the chosen standard be likely to become a major commercial success also, in order that the US Navy could capitalize upon the systems expertise developed from R & D Funding provided by the commercial and industrial sectors. All known existing and proposed bus architectures were evaluated against a set of criteria.

Futurebus scored substantially higher than the other contenders in almost all categories. After significant internal discussion at the highest levels of the DOD, the Pentagon announced in December 1988, their selection of the IEEE 896.1—1987 Futurebus as the basis for all future US Navy mission critical computers.

A third major influence on the specification came from the IEEE P1496 Rugged Bus Working Group, who had, in 1986, decided to develop a ruggedized specification for a backplane bus. Rather than develop an entirely new bus from scratch, the Rugged Bus Working Group chose to use the IEEE P896.1—1987 Standard as a base from which to develop their application needs. In November 1988, at a Joint Rugged Bus/Futurebus Working Group meeting, the two groups agreed to merge their efforts for a single bus standard, to be called Futurebus+. The Rugged Bus Working Group brought significant additional skills to the joint activity: real-time, fault-tolerance, maintainability and environmental conditions, in addition to the dedication and hard work of their members who became an integrated part of the team which defined and reviewed the latest Futurebus+ specification.

An additional strong influence on the specification came from the Multibus Manufacturer's group who, in February 1989, announced their intention to pair IEEE 1296 (Multibus II) with the developing Futurebus+ Specifications, into a common "Futurebus+ Extended Architecture" (FBX). From that point onwards, member companies of the Multibus Manufacturers Group provided technical support for the Working Group activities, and contributed greatly to the cooperative spirit among what were previously competing factions in the bus industry, to develop a *single* cache coherent bus architecture for the industry, which would now compete as a truly open standard against the myriad of proprietary and semi-proprietary architectures which had plagued the industry in previous generations of computers.

MAJOR FEATURES

Futurebus+ represents a major paradigm shift for the bus industry. It is the first comprehensive bus architecture designed *a-priori* to be an *OPEN standard (An interface specification for which there are* ***no*** *restrictions for who may use it),* and which was explicitly designed to support multiple generations of computer technology.

Futurebus+ will benefit end users immeasurably, by allowing vendors to build systems which not only have a much greater degree of compatibility, but which are "gracefully upgradable", preserving investments already made in the boards, enclosures, power systems, peripherals and other hardware (and software) as new improvements in processor architecture, levels of integration, and clock speed, are implemented.

Futurebus+ derives its name from its lack of built-in obsolescence parameters, and its upwardly compatible architecture and protocol extensions. Futurebus+ represents a significant departure from the philosophy of other standard or proprietary buses. The most important objectives of the Futurebus+ project were: (1) to create a bus standard that would provide a *significant* step forward in the facilities and performance available to the designers of future multiprocessor systems, and (2) to provide a stable platform upon which manufacturers can base several generations of computer systems.

In order to meet this objective, the following requirements were set:

- Architecture, processor and technology independent (attributes of a truely *OPEN* Standard).
- A basic asynchronous (compelled) transfer protocol for simple, higher reliability operation with a handshake flow control over each word of data transferred.
- An optional source-synchronized (packet) protocol for the highest possible performance, with flow control over each block of data transferred.
- No technology-based upper limit to the performance of the bus in both modes (i.e., limited only by the laws of physics—not by the technology).
- Fully distributed parallel and arbitration protocols to provide the minimum possible number of single point failure mechanisms.
- Parity protection on all lines, and feedback checking where possible (e.g., modules may write to themselves to facilitate self-testing).
- Multi-level mechanisms for locking of modules, *and* the avoidance of deadlock or livelock.
- Circuit switched and split transaction protocols. Plus support for memory controller commands to implement remote lock and other SIMD-like operations.
- Support of real-time mission critical computations. (Multiple priority levels with which to arbitrate, and the consistent treatment of priorities throughout the arbitration, message and DMA protocols. Plus support for the distributed clock synchronization protocol defined in IEEE P896.3.)
- Support for fault-tolerant and high availability systems (Dual bus operation, fault detection and isolation mechanisms and live insertion and withdrawal of modules).
- Direct support for snoopy-cache based shared-memory systems, with recursive protocols to support single or unlimited size systems consisting of arbitrarily connected buses.
- Recognition and support of strong and weak sequential consistency (time order assumptions).
- Compatible message transport definition supporting a number of message passing protocols.

TECHNOLOGY INDEPENDENCE

A unique attribute of the design of the Futurebus+ protocols is their technology independence: achieved through basing the protocols on fundamental protocol and physics principles and optimizing them for maximum communication efficiency (and hence throughput) rather than for a particular generation or type of processor. Timing and handshake protocols are thereby governed by "law of nature" types of constraints rather than limitations of current and projected technology such as device propagation delays, and capture windows.

The benefits of technology independence are reflected in the principle of no technology-based upper limit to the performance of the bus. The configuration or transaction capability modes guarantee interoperability when two boards of a different speed, or different generation, communicate on the same bus segment; thus providing the Futurebus+ with an unprecidented ability to support multiple generations of computers, well into the 21st century.

This principle of design longevity also provides a self-adapting performance span which allows graceful upgrades in computer equipment. This leads to less customer annoyance, greater confidence in their supplier and the equipment they sell. Manufacturers are also likely to find significant advantages in their manufacturing capability, the learning curve of their design engineers, and a strong strategic advantage in credibility to their customers.

Fundamental to the compatibility of slow, fast, old or new modules attached to the same Futurebus+ segment, is the notion of *broadcast*. Each connection phase (address cycle) is broadcast in a way which guarantees that each module on the same bus segment is able to decide whether or not to participate in the forthcoming data transfer phase. This turns out to be a basic requirement of a snooping cache coherence scheme, which allows multiple caches to search their directories to see whether or not they ought to be interested in the transaction. Even then, the protocols provide for the performance enhancements possible when a master knows that other modules will not be interested, by indicating whether other caches should even bother to search their directories or not (for example, when a cache block is replaced and ownership is transferred back to the main memory, or when the transactions are using the message passing mode).

The P896.1 specification may be implemented using *any* logic family (e.g., TTL, BTL, CMOS, ECL, GaAs) providing the physical implementation is constrained such that it meets the Futurebus+ signaling requirements (incident wave switching, and relative skews between information and synchronization lines).

Futurebus+ protocols may be used at *any* level in the system hierarchy. Device to device, board to board, or system to system. These protocols are particularly powerful when used at two or more levels in a hierarchical bus system.

ARCHITECTURE INDEPENDENCE

Futurebus+ is architecture independent, providing a flexible and elegant general purpose solution to cache consistency, within which other cache protocols will operate compatibly, while at the same time providing an elegant unification with a message passing transport protocol.

There are many exceptionally powerful features implicit within the Futurebus+ specifications which may not be obvious on a first reading of the specification. The Working Group chose to provide a flexible set of tools within which an implementation can be crafted to suit a wide spectrum of applications. Many of the protocol features may be used not only for the immediately obvious purpose, but are capable of being utilized for other purposes, many of which were recognized by the Working Group, but which were omitted from the descriptions in order not to over-complicate them.

Implicit in the philosophy of the Futurebus+ specification, is the concept that large systems may be built from smaller subsystems, and those small subsystems may in turn be built from yet smaller subsystems. Examples of which include the choice of split or interlocked transactions and a fully recursive, truly hierarchical cacheing paradigm. This is in anticipation of future generations of systems, which will inevitably utilize many levels of caches and internal/external buses. The Futurebus+ protocols will work well at *any* or *all* levels in this hierarchy; since for each generation of computer systems, each bus is absorbed into the packaging technology one level lower than the previous generation: traffic flowing on backplane buses today will flow on the on-board buses of tomorrow; and on chip-level buses soon after that!

Futurebus+ has been designed with large scale integration of the bus interface in mind. A severe restriction on the number of signals has allowed the bus to steer clear of being intrinsically expensive. Hence, while initial implementations of the Futurebus+ may require multiple interface devices, the learning curve, and volume, will allow these to be further integrated and for the market forces to rapidly bring down the cost of the interface to become consistent with even the lowest cost workstations, industrial and personal computers.

PARALLEL PROTOCOL

The principal objective behind the design of the parallel protocol, was to achieve the highest possible transfer rate consistent with the implementation technology and manufacturing constraints at the time the modules were designed, allowing both backwards and forwards compatibility. This "technology-independence" aspect of the Futurebus+ protocols provides the following advantages:

- A long design longevity for the bus, consistent with the attributes of a widely implemented standard for commercial, industrial and military requirements. Futurebus+ has been designed in anticipation of its continued use well into the 21st century.
- Graceful upgrade of system components, rather than the more usual "throw it all away and start again" scenario that occurs with traditional computers when higher clock-rate processors become available.
- Flexible tradeoff between cost and performance at any one point in time. Boards may be designed with exotic technologies to achieve the highest possible performance, or with more cost-effective technologies to achieve excellent cost/performance tradeoffs, and both can operate within the same system (providing they conform to the same profile).

The Futurebus+ parallel protocols support *both* connected and split transactions. Simplistically speaking, connected transactions are cheaper, easier to implement, but are slowed by the access time of the memory. Split transactions are more complex, expensive, but provide higher multi-stream system bandwidth since the bus need not be idle during the memory access time. Connected transfers provide lower latency for access to memory (hence single-stream performance is optimized), split transactions provide more concurrency in the protocols, hence they optimize multi-stream performance. Both are needed in a system architecture. Connected transfers provide a cost-effective performance for I/O operations, while split transactions provide the maximum performance required by MPU/Cache-memory operation.

Current RISC or CISC Microprocessors do not easily support a command structure for memory controllers to perform remote lock operations. Futurebus+ provides a simple workaround for this: use the split transactions for everything with the exception of lockvariables, and automatically revert to using connected transactions to perform RMW type operations. This allows the performance enhancements of split transactions to be obtained without having to redesign the processor, or utilize ugly software workarounds which require existing code to be modified and adversely affect the performance of the system.

SYNCHRONIZATION DOMAINS

The most frequently misunderstood aspect of bus design and the one which invokes the most religious fervor among protagonists, is the synchronization protocol.

One of the basic arguments for the asynchronous (source synchronized) protocols in Futurebus+ is that it allows the synchronization domain of the sender to extend along the bus segment, presenting only one synchronization interface between the bus and the receiver. This contrasts with centrally synchronized buses which require three separate synchronization domains (sender, bus, receiver). Synchronization interfaces mean performance is lost due to resynchronization delays and MTBF is reduced due to metastability. Since Futurebus+ has only one synchronization interface, it will naturally yield a higher performance than any centrally synchronized bus.

SCALABILITY AND PERFORMANCE

Scalability of cost and performance were early requirements in the design of the Futurebus+ protocols. Although Futurebus+ is basically a 64-bit address architecture, employing a standard 64-bit multiplexed address/data highway, a 32-bit Address/Data *subset*, and 128-bit or 256-bit data *superset* is also provided. Perhaps more importantly, the performance scales over time with a fixed width highway, allowing for example, systems to be built with 20 ns transfers in 1991 and 10 ns transfers by 1995.

Note: While 10 ns per transfer is often regarded as an upper limit due to the physics of the bussed transmission line environment, this is a conservative estimate based on measurements with a full-length "backplane" implementation of Futurebus+ protocols and an electrical environment, supporting upwards of 20 boards. Systems with fewer modules, shorter stubs, and shorter backplanes or no backplanes at all (i.e., on-board implementations of the Futurebus+ protocols), will be able to significantly exceed this limit.

Multiplying these numbers by the data highway width options (32, 64, 128, and 256 bits), provides a protocol throughput (with an appropriate electrical environment), of approximately 100 Mbytes/second at the low-end for systems using the 32-bit subset in 1990, while systems using the 256-bit superset in 1995 will peak at 3.2 GBytes/second—a dramatic demonstration of the benefits of a scalable design.

There are two principal reasons why Futurebus+ is able to attain this level of performance, while other buses cannot: (1) All Futurebus+ protocols are *source synchronized,* meaning that the synchronization signal is always emitted by the same module which emits the information signals, thereby eliminating spatial skews, and (2) substantial effort has been put into the understanding the signaling environment to guarantee that receivers are triggered by the *incident* wave from the driver as it passes by along the bus lines.

Note: Spatial Skews result when there is a space (and hence time) difference between the transmitting and receiving modules, and the passage of data between them is synchronized by a signal transmitted by a 3rd party. For example, a bus with a central clock source must account for an additional uncertainty in the arrival of data at a receiver equal to the space (time) difference between the sender and receiver of the data, independently of the mechanism used to distribute the central clock.

Another dimension of scalability for the Futurebus, which lends itself also to fault-tolerant applications, is the use of dual or multiple parallel Futurebus's. The locking mechanisms, cache coherence mechanisms, and the general architectural facilities are designed such that they can be readily applied to such applications. Moreover, it may well be possible that implementations of dual 64-bit or 128-bit buses (as opposed to single 128-bit or 256-bit buses) are likely to lead to better multi-stream system throughput (with a fixed number of bytes per block), due to the smaller fraction of the overall transaction time used as overhead in the narrower width implementations of the bus.

A NOTE ON PERFORMANCE

Since Futurebus+ is a technology-independent specification, there is very little in the protocol specification which would indicate what their actual performance would be in a specific implementation, without building and testing such a system.

Although there is, in principle, no upper limit to the performance of the Futurebus, there are some basic issues regarding the achievable performance which are not widely understood. The actual performance of any bus in any particular system will be a function of the following three critical elements:

- *The theoretical cleanliness of the protocols.* e.g., do the protocols carry any unnecessary overhead; do they carry any unnecessary constraints of synchronism to a central clock, or too many round-trip delays in the handshakes; do they incorporate fixed technology constraints such as arbitrary set-up and hold times?
- *The architecture of the system in which the bus is used.* The most theoretically perfect bus protocol will grind to a snail's pace in a poorly architected system. Very often, it is the memory system bandwidth which is being measured rather than the intrinsic protocol bandwidth of the bus. Systems which perform random transfers can never sustain as high a transfer rate over the main highway as systems specifically designed to take advantage of the intrinsic efficiencies of the memories and bus protocols, such as block transfers.
- *The implementation.* Only if full advantage is taken of available semiconductor technologies, will the protocols perform at their full potential. Since the Futurebus+ is a truly *open* standard, its benefits are available to any and all who choose to adopt the bus. Competitiveness, and added-value, will now focus on the quality and thoroughness of the implementations. Since Futurebus+ protocols provide significant room for learning-curve effects in the industry, one can expect a rapid evolution in the performance of systems which use Futurebus+, through a steady but definite improvement in the implementation.

Data from various simulation results and reports on anticipated semiconductor performance have been used to predict the expected peak performance figures for the Futurebus+ over time as shown in *Figure 1.* The packet protocols are shown from a 1991 baseline because of the anticipated wide availability of devices which can implement this mode during that time-frame. Conservative assumptions were used in the development of this model; it is quite possible for some commercial organizations to be well ahead of these performance curves when they introduce their products.

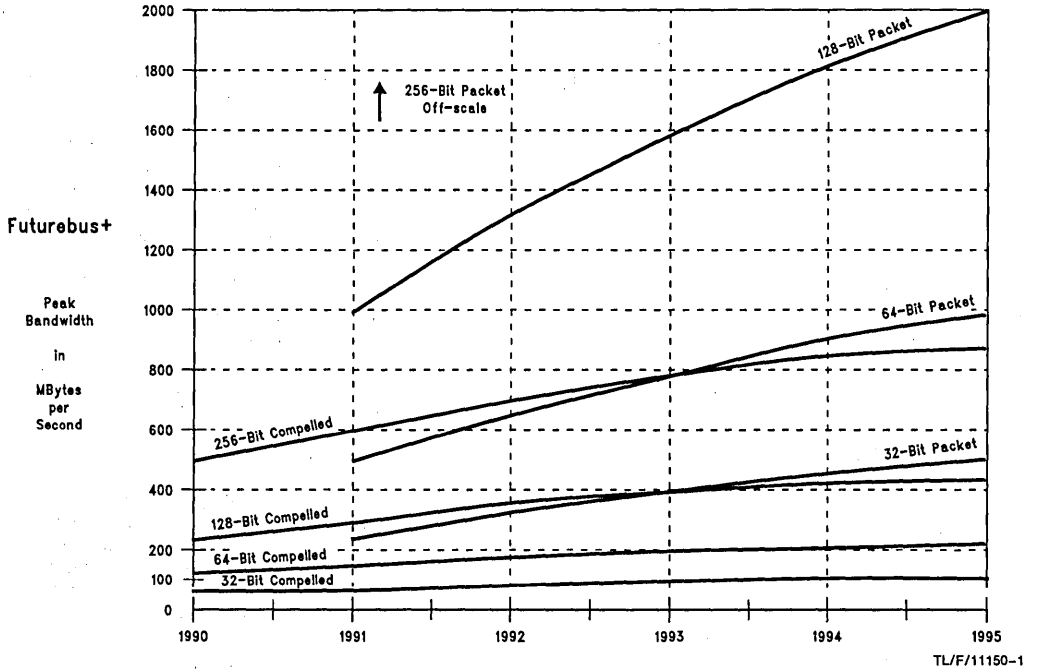


FIGURE 1. Futurebus+ Protocol Bandwidth (Peak)

ARBITRATION

Futurebus+ uses a highly evolved version of the Parallel Contention Arbiter. This mechanism arbitrates among active contenders through a combinatorial logic network distributed among all possible contenders using a set of simple bussed lines connecting them. By applying a unique arbitration vector to this logic, one contender only (with the highest arbitration vector) will be uniquely selected as the winner. The Futurebus+ implementation additionally distributes the responsibility for synchronization of this mechanism among all the contenders, such that no central logic whatsoever is required on the bus segment.

Although not as familiar as a central request/grant scheme, the parallel contention arbiter used in Futurebus+ has a number of significant advantages:

- The current master can observe any requests (and their priority). This allows the current master to make a reasonable decision as to whether it should give up the bus immediately (high priority request), at the next "convenient" breakpoint (low priority request), or to continue until the end of a potentially long block transfer or series of transactions (no requests pending).
- Multiple priority levels. Allows real-time systems to deterministically allocate bus bandwidth to those processors running the most critical tasks in situations of high system load, i.e., it allows one to determine if a process is schedulable (guaranteed to meet its deadlines even under conditions of high system load), and improves the probability that a process is schedulable within a system with fixed available bandwidth.
- A master-elect can be pre-empted by a higher priority contender which arrives after the arbitration has taken place, but before the current master has relinquished the bus. This allows the higher priority contender a significantly shorter average latency to access the bus.
- Arbitration Messages (or events) can be broadcast on the arbitration bus without the need to disturb the traffic currently underway on the parallel bus. This mechanism can be used to implement, for example, interrupts, targeted events and parallel processing rendezvous at synchronization points before proceeding; all without the need for dedicated lines on the bus for those miscellaneous system control functions.

Implemented in a purely text-book fashion, and with all the parameters of the arbitration network set to their maximums, this arbitration mechanism will perform approximately similar to a daisy chained mechanism, but without the inherent disadvantages of such a scheme. Significantly faster implementations may be designed, which still fully conform, by noting that the arbitration parameters (arbitration vector, arbitration contest settling time, Wired-OR Glitch filters, etc.) can be traded off against the number of active masters in the system, the total number of modules attached to any one bus segment, and the length of that bus segment: all parameters which can be ascertained during the system initialization phase. Also, in order to take advantage of the performance of higher speed modules (i.e., modules with faster logic), two arbitration "speeds" are dynamically selectable. This ensures that the compatibility is maintained among old and new boards attached to any bus segment.

A unique aspect about the Futurebus+ arbitration scheme, is the optional "idle-bus" arbitration method, used when there is only one requesting master. This method, using the Address/Data lines to signal a request when the bus is idle, provides the fastest possible latency for a module to access a memory subsystem. In the event that two or more masters are detected during this request phase, the scheme automatically reverts to using the parallel contention arbiter method, already started concurrently with the fast idle bus method. The result is a significantly faster average latency for a module to access the memory subsystem. Used in combination with the connected bus protocols. This can result in a significant reduction in the miss penalty of modern RISC processors, as compared to all other bus designs.

SYSTEM ARCHITECTURE SUPPORT

The Futurebus+ specification has been written to provide support for a wide variety of system architectures. Within a single bus segment, both functionally distributed (message passing) and shared memory (cache coherent) systems may coexist. For multi-crate systems, Futurebus+ has a number of companion standards offering a bridging service to other bus standards. This wide range of options will allow a smooth transition from existing buses to Futurebus+ and beyond to other advanced interconnection architectures such as the Scalable Coherent Interface (SCI): a ring-based interconnection for large scale parallel processing system implementations. It also allows the needs of secure data and tightly-coupled parallel processing regimes to be balanced.

A message passing architecture is, in its purest form, a write-only system. In such a system, a module requiring access to data sends a request message to the module owning the data; the responding module will reply at some time later by writing the data to the requestor. It is easy to see that access restrictions may be readily implemented in such a system without cutting down on the available bus bandwidth; the responder simply checks the authorization of the requestor before replying. What is not immediately obvious is that such a system can make very efficient use of the bus. First, only write transactions are being used, and write transactions are inherently more efficient: requiring only the presentation of the data and a response, rather than a request, an access time for the data and a response, as is required for read transactions. Second, messages use fixed size blocks, allowing burst transactions to amortize the cost of transaction setup over the block. Message passing systems also have the advantage of being simple to design and analyze since nothing is shared and all transactions are inherently split.

Futurebus+ describes an optional message passing architecture in Chapter 9. The P896.1 model uses a default message frame of 64 bytes (note the compatibility with the buffer sizes needed by the cache coherence protocols), a broadcast mailbox with a message filter and point to point request and response mailboxes. Messages are transmitted without the use of any special bus level protocols. Only two of the Futurebus+ transaction types are used: Write-unlocked and write-no-acknowledge. This makes Futurebus+ message passing easy to implement and easy to add to an existing interface design.

Shared memory architectures represent a different approach to system data flow. In this model, all data needed for use by more than one resource within the system is held in a global, shared memory subsystem, accessible to all bus masters within the system. This method of sharing data would be costly in terms of bus bandwidth and access latency if all modules had to compete for access to the same physical memory module(s), so caching and split transactions are provided to enable much higher performances to be obtained by eliminating unnecessary contention. A shared memory model is extremely useful when multiple processors are operating on the same data and is the preferred architecture for use in tightly-coupled multiprocessing. The use of constructs such as caches and split transactions requires that the hardware implement a well-thought out series of protocols to ensure that all processors always operate on valid data only.

The shared memory and message passing models are not mutually exclusive within the Futurebus+ environment: they can readily co-exist, allowing the system integrator to meet seemingly orthogonal requirements.

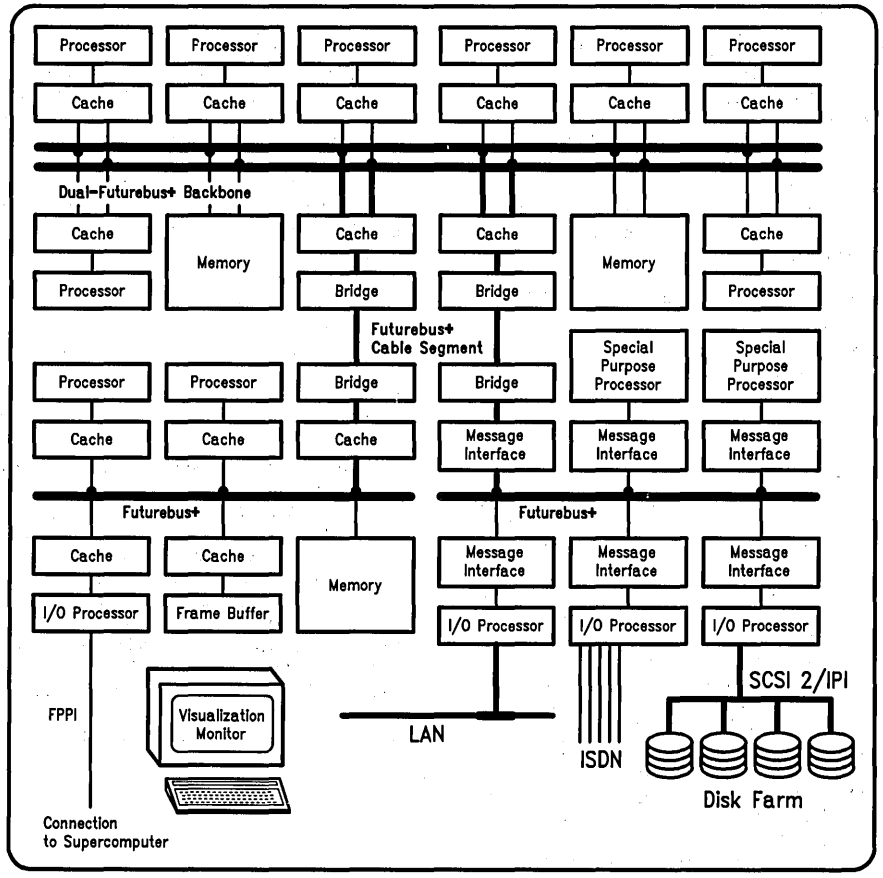
CACHE COHERENCY

One specific application which the Futurebus+ is extraordinary well suited to, is the shared memory multiprocessor. The Futurebus+ protocols include all the transaction types necessary to drive the states of various cache modules to conform with almost any arbitrary cache coherence protocol. This turns out to be easier than it sounds. The Futurebus Working Group discovered, early in 1986, that all the existing cache coherence schemes are really variations on the same theme, but with an arbitrary nomenclature for the cache states. The result was the development of a unified cache model called the MOESI model. Futurebus+, uses a slight restriction of this model (MESI), in which tradeoffs were made to increase system performance and to allow operation of the protocols across bridges. It is recognized that different schemes may be needed to yield the best price/performance tradeoff for different applications, but that all the chosen schemes within any one system are required to interoperate.

More interestingly, perhaps, is the notion that the Futurebus+ cache consistency scheme works in an environment with multiple logical Futurebus's connected in a hierarchical fashion. Although the parallel protocols are notionally comprised of complete "connected" transactions, they may be recursively "split", by bus repeaters which insist that a copy of the transaction must be repeated on adjacent buses to which are known to have copies of the same cache block. Inherent in this assumption is that each bus repeater contains a cache (at least the cache tag), which can record the passage of a cache block to the neighbor bus, and thereby maintain the knowledge that they must pass on future references within that block to the other bus.

Perhaps most interestingly, and this is one of the many hidden benefits of the Futurebus+ architecture, these protocols are exactly the same whether they are being recursively applied to a system with multiple Futurebus+ backplane segments in different crates, or to a system within one crate (a single backplane), and multiple local Futurebus+ segments on each of the boards, each with multiple processors on them.

Futurebus+ based multiprocessor systems scale well with the number of processors because clusters of processors can be interconnected on different buses, with each bus connected to another by a cacheing bus repeater. Thus, traffic on one bus will disturb a remote bus only if the bus repeater cache misses on a block which exists on the remote bus. Just as a cache acts as a filter to keep traffic generated by a processor from reaching the first level bus, the cacheing bus repeater also acts as a filter to keep traffic generated on the first level bus from reaching the second level bus. Even moderately well architected Futurebus+ based systems will scale up to many 10's of buses, and several hundred processors without showing signs of bandwidth saturation.



TL/F/11150–2

FIGURE 2. Typical Futurebus + Application

SEQUENTIAL CONSISTENCY

Computer Architecture has been evolving for many years, and many problems which now seem familiar, were far from obvious when they were initially recognized. Several years ago, the Working Group solved some fundamental physical and conceptual problems concerning the physics of the backplane environment, and the design of the bus protocols to provide general mechanisms to support cache coherency. Recognizing that there are likely to be many more problems which have not yet been either widely recognized, or well articulated, the Futurebus+ Working Group has incorporated many (mostly hidden) hooks into the specifications to allow its evolution, in a compatible way, to meet both anticipated and unforeseen future requirements.

For example, the Futurebus+ protocols have been carefully designed to be fully consistent with the physical model of the universe implied by special relativity. The model dispels the myth of the concept of simultaneity, because all laws of physics, including electrodynamics, optics, mechanics, *and data location and movement* are invariant with respect to all coordinate systems, and that spatial distribution is equivalent to time (in this case, latency). It is the lack of recognition of this principle in computer science which gave rise to the conceptual difficulties called sequential consistency.

Sequential consistency was first recognized in early 1988 by the Futurebus+ Working Group when trying to reconcile the view of the universe from the different perspectives of the programmer and hardware engineer. It is the assumption, often made without explict reference by programmers, that the order of events observed by any one device is the same as any other device. Unfortunately, when the transmission time of an event or protocol packet (due to the finite speed of light) is significant relative to the period between events occurring in the system, this assumption is no longer valid, since it violates the principle of relatively (that there is no absolute frame of reference, and the order of events depends on both the space and time coordinates of the observer relative to the rest of the system).

Futurebus+ provides support for dynamically choosing between strong and weak sequential consistency. One way this is supported is through the split transaction protocol, by either posting the writes and continuing without waiting for a response (weak consistency), or by requiring the processor/cache to wait for all responses from other blocks before continuing (strong consistency).

The effect of strong sequential consistency is to sequentialize several variable updates through one place in space-time (the current owner). Since in the MOESI model there is by definition only *one* current owner (although there may be several surrogate owners in the bridges), the effect is to create a single point in space through which the synchronization can take place.

It was important to provide support for strong consistency since many experts believe that sequential ordering of events must be guaranteed in order to ensure the correct operation of certain programs (this is currently an outstanding research question in the academic world). However, strong sequential consistency significantly reduces the amount of concurrency allowed in the system, and can therefore have a highly detrimental effect on multiprocessor system performance. Weak sequential consistency is provided for those *new* programs wishing to take maximum advantage of the concurrency available in a system, but which are cognizant of the programming constraints needed to ensure correct operation when strong sequential consistency is not guaranteed.

Mechanisms to support synchronization barriers for the rendezvous of multiple threads in a task executed in parallel are also provided by the broadcast and broadcall facilities in the parallel protocol, and by the arbitration messages provided by the bus allocation scheme.

BRIDGES

The Futurebus+ Working Group has been firmly committed to the concept of bridges for many years, and believe that bridges are a requirement for any new bus architecture to succeed. There is presently an enormous investment made in existing bus products. This investment is two dimensional, representing a great depth in quantity of systems sold and breadth in terms of available functionality within a given bus architecture. Since a new bus will not be able to achieve this penetration for some time, and furthermore does not wish to require previous investment to be written-off, bridges provide a pragmatic solution to this problem.

A bridge represents a concept more than a physical implementation. The idea is to allow transactions begun on one bus to be completed on another with minimal impact on the software and hardware of the standard modules on either bus. This means more than simply converting protocols. A bridge must provide a nearly seamless means of communication between what are basically incompatible architectures. This includes methodologies for defining how interrupts are handled on a bus which does not recognize the single-processor concept of interrupts, for example.

For Futurebus+, this has been addressed for a number of existing and future bus standards. Futurebus+ provides its own specifications within the basic protocols to bridge to itself. Bridges have also been defined between Futurebus+ and VMEBus (IEEE P1014.2), Multibus II (IEEE P1296.2), and the Scalable Coherent Interface (IEEE P1596). Those will allow Futurebus+ to be used with existing or future hardware and software from any of those bus architectures. A smooth upgrade path from lower performance buses to higher performance buses is assured, as is the usefulness of the newer bus standard even without a broad product line in its early days.

TESTABILITY AND MAINTAINABILITY

Futurebus+ has been designed from the outset to make it straightforward to build testable implementations. Facilities to support error detection go hand in hand with hooks to stimulate those mechanisms to deliberately cause an error, so that the mechanism may be tested.

Much discussion took place on the need for a boundary scan standard (such as IEEE P1149) to be included within the Futurebus+ signal set. Since the bandwidth requirements for such a bus are so low, the relative cost of bus signal lines is not negligible, it was decided to support these facilities through the use of a boundary scan controller on the modules themselves (logic is cheap inside LSI devices), and to stimulate and control this controller through commands and telemetry transmitted through the IEEE P1394 High-Serial Bus mechanism provided on two of the pins of the Futurebus+ signal set.

In addition, IEEE P896.3 may specify a test access port as a requirement on the access panel of modules built to a profile which includes requirements for Maintainability and Testability. See IEEE P896.3 for further details.

REAL-TIME APPLICATIONS

Futurebus+ is also designed to support real-time applications where the correctness of computation depends upon not only the results of the computation but also the time at which outputs are generated. Examples of real-time applications include air-traffic control, avionics, process control and mission critical computations. The measures of merit in a real-time system include:

- Fast response to urgent events and accurate timing information.
- High degree of schedulability. Schedulability is the degree of resource utilization at or below which the deadlines of tasks can be ensured. It can be thought of as "real-time throughput", i.e., the number of timely transactions per second.
- Stability under transient overload. When the system is overloaded by events and it is impossible to meet all the deadlines, we must still guarantee the deadlines of selected critical tasks.

The distributed clock synchronization protocol defined in P896.3 provides users with accurate and reliable timing information. The prioritized arbitration message mechanism provides the basis of fast response to urgent events. The multiple priority levels in the arbitration protocol and the consistent handling of priority among arbitration, message and DMA protocols provides the user with a consistent and powerful scheduling tool. As a result, users can employ analytical scheduling algorithms, for example, the rate monotonic algorithm, to ensure a high degree of schedulability and stability. In short, Futurebus+ architecture facilitates the development of real-time systems, whose timing behavior can be analyzed and predicated *a-priori*.

FAULT-TOLERANT APPLICATIONS

Futurebus+ is also designed to be suitable for use in high integrity systems: the address/data, command and arbitration lines are all protected by a parity bit per byte, there are *no* central elements to adversely affect the survivability of the system, and modules can be inserted or removed while the system is operating to facilitate high availability applications.

Included in the arbitration protocols and initialization facilities are the mechanisms to support the live insertion and removal of modules. Although Futurebus+ includes the necessary hooks. Full specification of how they are used in an application system can be found in P896.3.

PROFILES AND INTEROPERABILITY

Older bus standards often suffered from interoperability problems. Boards from different manufacturers would often not communicate with one another. Since this defeats the purpose of open standard architectures, the Futurebus+ specifications have been laid out to maximize interoperability. This has been done in three ways: First, the specifications have been written in a form which is unambiguous (using formal equations where possible rather than purely English statements). Second, an adequate amount of background description has been provided, to encourage the implementor to understand and share in the instinctive elegance of the protocols with the original designers of the specifications. Third, by providing a hierarchical system of constraints to allow a tradeoff between application specific (e.g., low cost versus high performance), and application generality without compromising interoperability. This latter point has been implemented through the concept of profiles.

The Futurebus+ family contains a number of specifications which do not individually constitute a bus specification, but provide a rich set of tools with which to build optimal implementations for a wide spectrum of applications. When called up within profiles, however, these specifications, and the options within them, are much more tightly constrained. This is to provide the implementor with clear guidelines, which if followed, will maximize interoperability with any other implementation which follows the same guidelines.

As an example, the specifications which may be treated as a family for use within a typical Futurebus+ profile are: P896.1 Futurebus+ Logical Layer Specifications, P896.2 Futurebus+ Physical Layer and Profiles, P1194.1 Backplane Transceiver Logic, P1212 CSR Architecture, P1394 (High speed serial bus), and P896.3: Futurebus+ Systems Configuration Guide.

A profile consists of a mapping of which sections of which specifications and which options within those sections are appropriate for use together within an implementation. Profiles sanctioned by the Working Group will be contained within the P896.2 document, and higher level profiles suitable for specialized system application environments will be included within P896.3. This explicit discussion of what is required and what is not within a given profile addresses the interoperability issues brought about by arbitrary assignment of optionality by manufacturers. Profiles also allow the buyer of Futurebus+ based products to know exactly what features come with each product. If a manufacturer follows the requirements laid out within a profile, that product is much more likely to be interoperable with the products of any other manufacturer meeting that profile.

ACKNOWLEDGEMENTS

Futurebus+ would never have been possible without the visionary outlook of several key individuals: Lyman Hevle, Executive Director of VITA, Shlomo PriTal of Motorola, LCDR Harrison Beasley, of the US Navy, Paul Cook, Chair of the P1496 Rugged Bus Working Group, and Vice Chair of the Futurebus+ Working Group. John Hyde, Chair of the P1296.2 Committee and representative of the Multibus Manufacturers Group, and Wayne Fischer of Force, and Chair of the P1014.2 Working Group.

REFERENCES

1. IEEE P896.1: Futurebus+ Draft 8.2, Published by the IEEE Computer Society, 1730 Massachusetts Avenue, N.W., Washington, D.C. 20036-1903.

Futurebus+ Asynchronous Slave Memory Design

National Semiconductor
Application Note 668
Webster (Rusty) Meier, Jr.
David Hawley, Fred Leung,
and James Jones III

INTRODUCTION

This application note describes a Futurebus+ asynchronous slave memory board design. Part 1 of this application note contains an introduction to Futurebus+ and how the asynchronous slave memory board interfaces to it. Part 1 then goes into a more detailed description of the Futurebus+ slave protocol controller and the control lines that interface it to Futurebus+ and the Memory Module. Part 2 of this application note describes the Memory Module and how the Memory Module Control interfaces between the two National Semiconductor DP8422A-25 DRAM controllers, the two banks of DRAM, and the Futurebus+ slave protocol controller.

This application note implements a Futurebus+ asynchronous slave memory design (see *Figures 1* and *2*) using National Semiconductor GAL's (or PAL's) to achieve a peak transfer rate of 20 mega-transfers/second speed (approximately 80 Mbytes/sec using 32 data bits per memory bank and 160 Mbytes/sec using 64 data bits per memory bank) during compelled burst transfers. The Futurebus+ protocol supports the compelled mode of operation. The word compelled refers to the fact that after the master gives a request for a transfer the slave is compelled to provide a response before the master can proceed to the next transfer. The request/acknowledge protocol is asynchronous and edge sensitive where both rising and falling edges are used to transfer information. This application note assumes that the reader is familiar with the DP8422A-25 DRAM controller, GAL/PAL design, and Futurebus+.

INDEX TO APPLICATION NOTE

PART 1: FUTUREBUS+, THE PROTOCOL CONTROLLER, AND THE INTERFACE TO THE MEMORY MODULE

1.0 INTRODUCTION TO FUTUREBUS+

Futurebus+ is a high-performance asynchronous 64-bit multiplexed address/data backplane bus designed by the IEEE 896.1 committee for use in a wide variety of multiprocessor architectures. The Futurebus+ standard is a next generation backplane bus standard developed to a set of requirements including openness, performance, and system facilities and flexibilities so as to not hinder systems using this bus for many generations of computer systems. Futurebus+ is a single cache coherent backplane architecture featuring scalability, technology independent protocols, and explicit provisions to extend to future applications. Requirements set for the Futurebus+ architecture by the IEEE 896.1 Futurebus+ Working Group include:

1. Architecture, processor, and technology independent
2. A basic asynchronous (compelled) transfer protocol
3. An optional extended source-synchronized protocol
4. No technology-based upper limit to bus performance
5. Fully distributed parallel and arbitration protocols
6. Support for fault-tolerant and high-availability systems
7. Support for cache-based shared memory
8. Compatible message transport definition

Compatibility of varying speed and technology boards connected to the same Futurebus+ backplane is dependent upon broadcast capability, or a snooping cache coherence scheme. The performance of a Futurebus+ backplane is scalable over time to allow a low-end 32-bit system operating at 100 Mbytes/sec to be expanded to a 256-bit system operating at 3.2 Gbytes/sec in the future. This performance is attainable by the use of source-synchronized protocols which eliminate spatial skews and receivers which are triggered by the incident wave from the driver. The synchronization protocol of the Futurebus+ backplane allows the synchronization domain of the sender to extend along the backplane, presenting only one synchronization interface between the bus and the receiver.

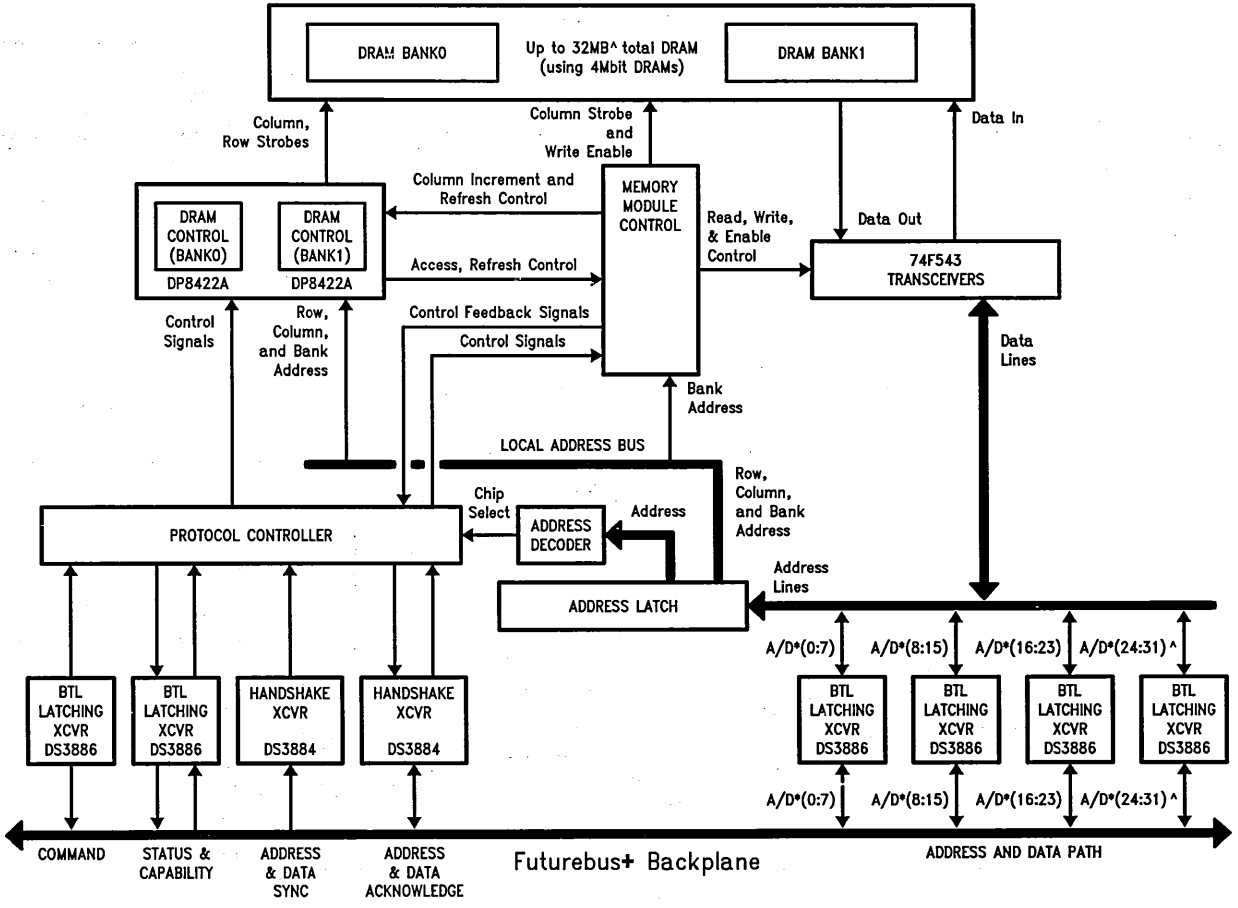
Futurebus+ uses an evolved version of the Parallel Contention Arbiter. Through the application of a unique arbitration vector to this logic, one contender only will be uniquely selected as the winner. This implementation requires no central logic on the backplane and gains the following advantages:

1. The current master can see any requests and their priority and determine whether it should give up the bus
2. Multiple priority levels, allowing systems to allocate bus bandwidth to processors running the most critical tasks
3. A master can be pre-empted by a higher-priority contender allowing a shorter average latency to access the bus

In addition, Futurebus+ has a number of companion standards offering bridging to other backplane and distributed buses such as VMEBus (IEEE P1014.2), Multibus II (IEEE P1296.2), and the Scalable Coherent Interconnect (IEEE P1596).

2.0 INTRODUCTION TO SLAVE MEMORY BOARD DESIGN

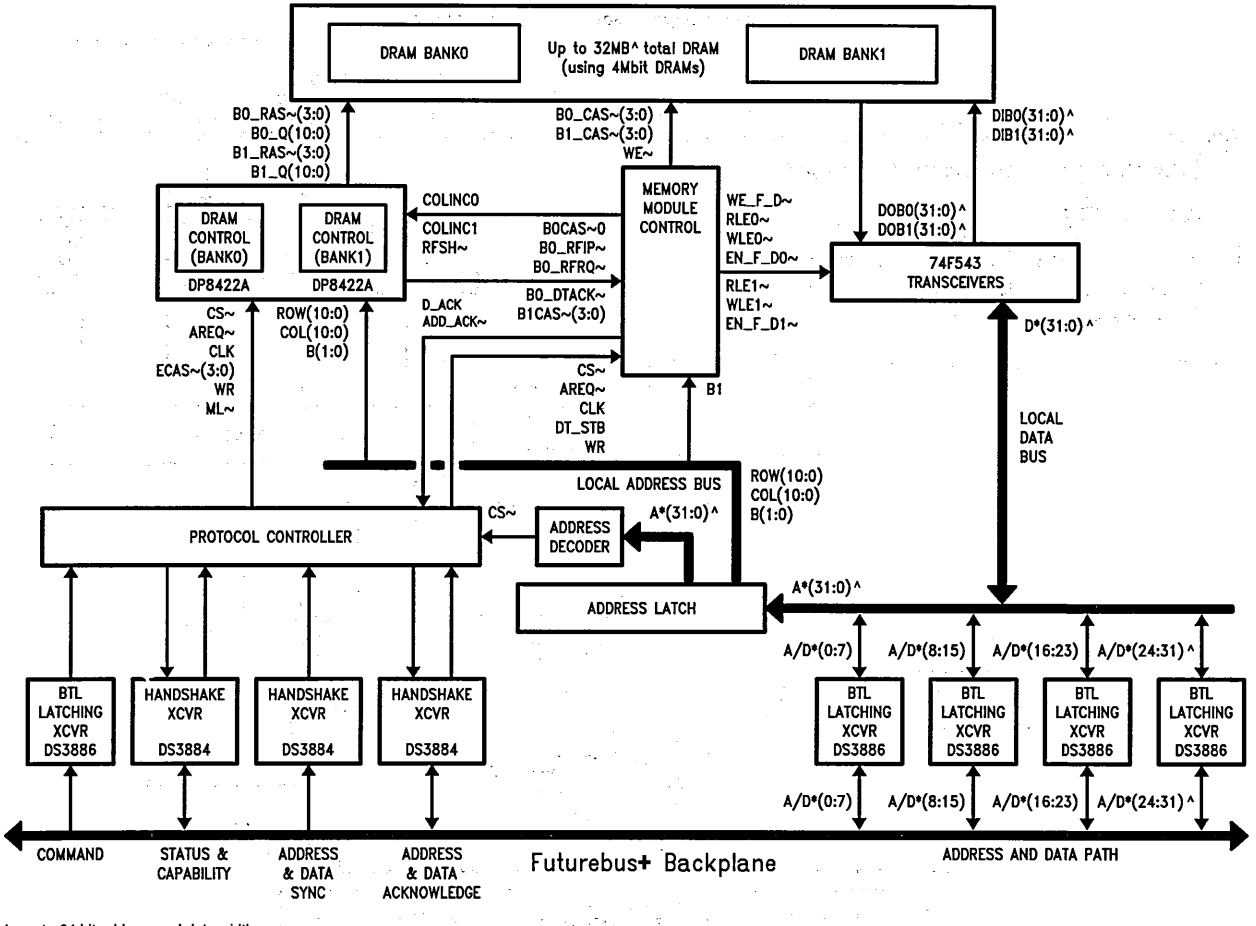
This application note describes a Futurebus+ slave memory board (*Figures 1* and *2*). This memory board *(Figure 3)* incorporates a GAL based Futurebus+ Protocol Controller consisting of two GALs and 6 delay lines *(Figure 4)*, a GAL based Memory Module Control unit consisting of five GALs and two delay lines *(Figure 7)*, two DP8422A-25 DRAM controllers *(Figure 6)*, two DRAM memory banks (32 bits each, but expandable to 64 bits each along with parity bits, *Figures 3* and *5*), two 74F543 latched transceivers *(Figure 3)*, an address latch *(Figure 3)*, an address decoder, two DS3884 Handshake Transceivers *(Figure 3)*, and six DS3886 BTL Latching Data Transceivers *(Figure 3)*.



TL/L/10772–18

^Easily expandable up to 64-bit address and data width

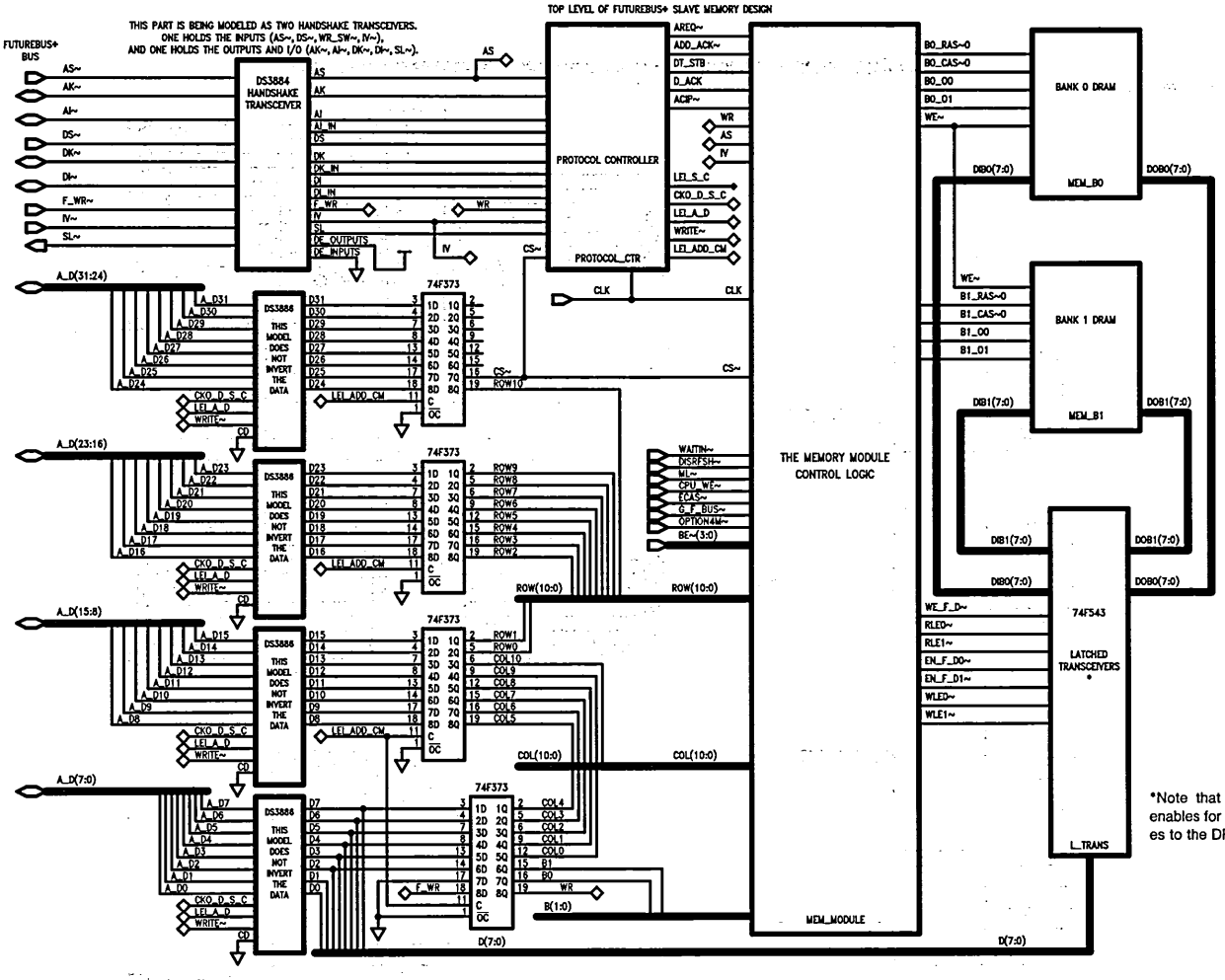
FIGURE 1. Slave Memory Module Block Diagram 1



TL/L/10772–19

^ Easily expandable up to 64-bit address and data width

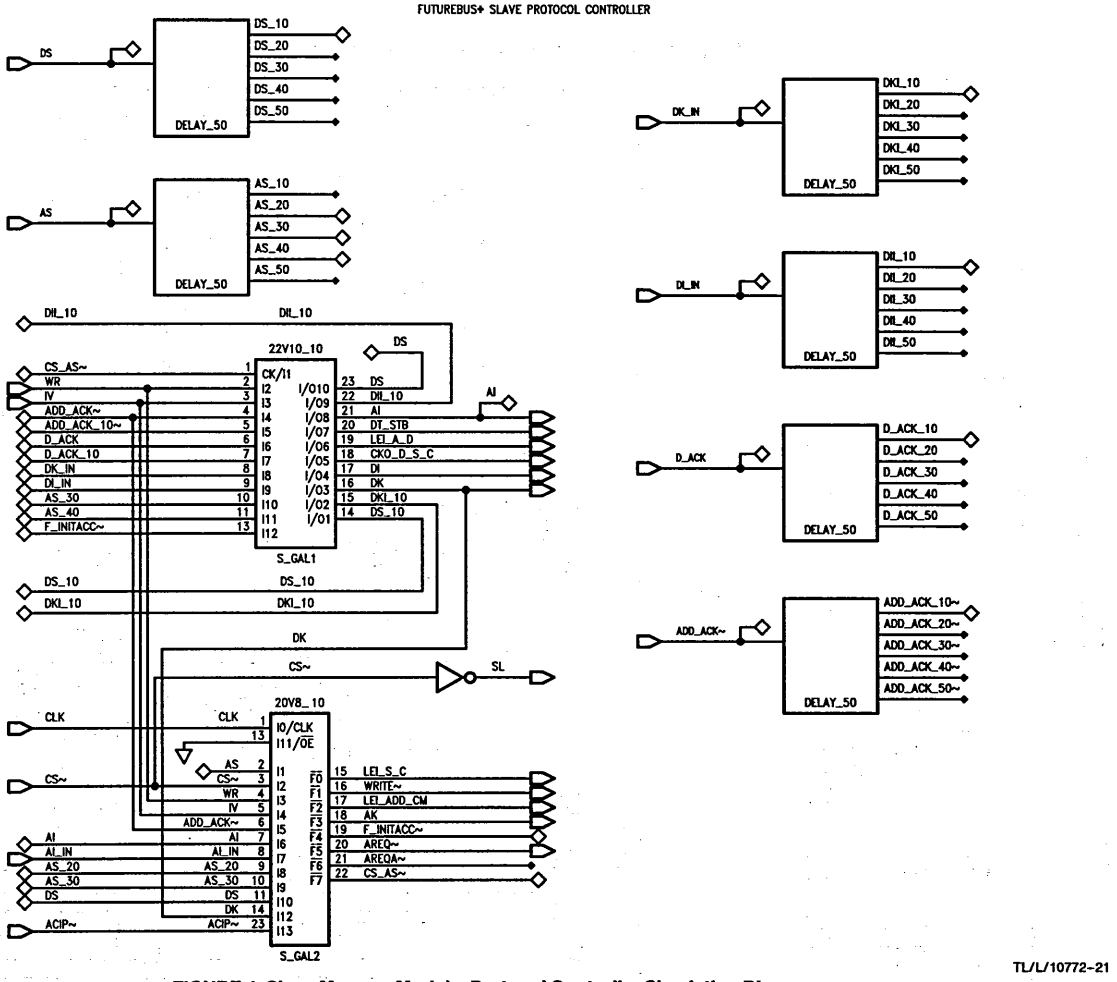
FIGURE 2. Slave Memory Module Block Diagram 2



*Note that the 74F543 chip enables for data write accesses to the DRAM are enabled.

TL/L/10772–20

FIGURE 3. Slave Memory Module, Top Level Simulation Diagram



TL/L/10772-21

FIGURE 4. Slave Memory Module, Protocol Controller Simulation Diagram

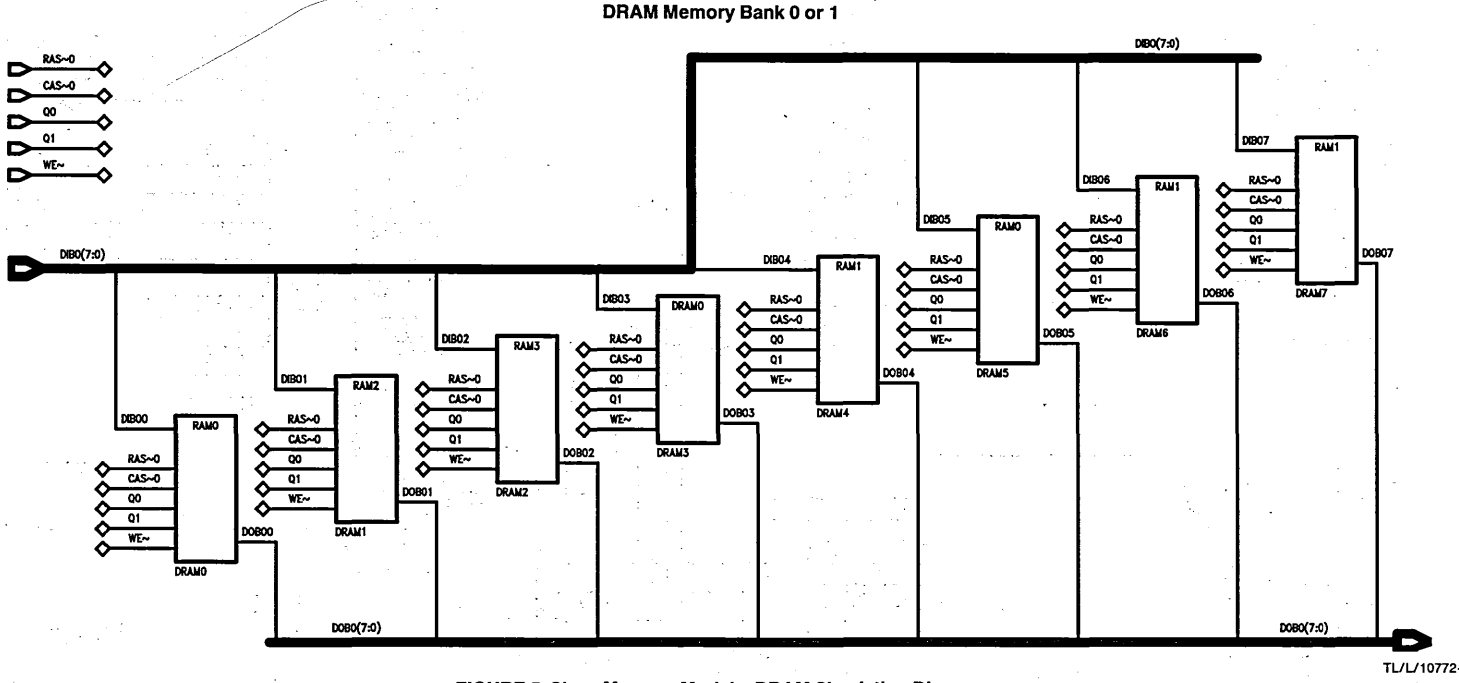
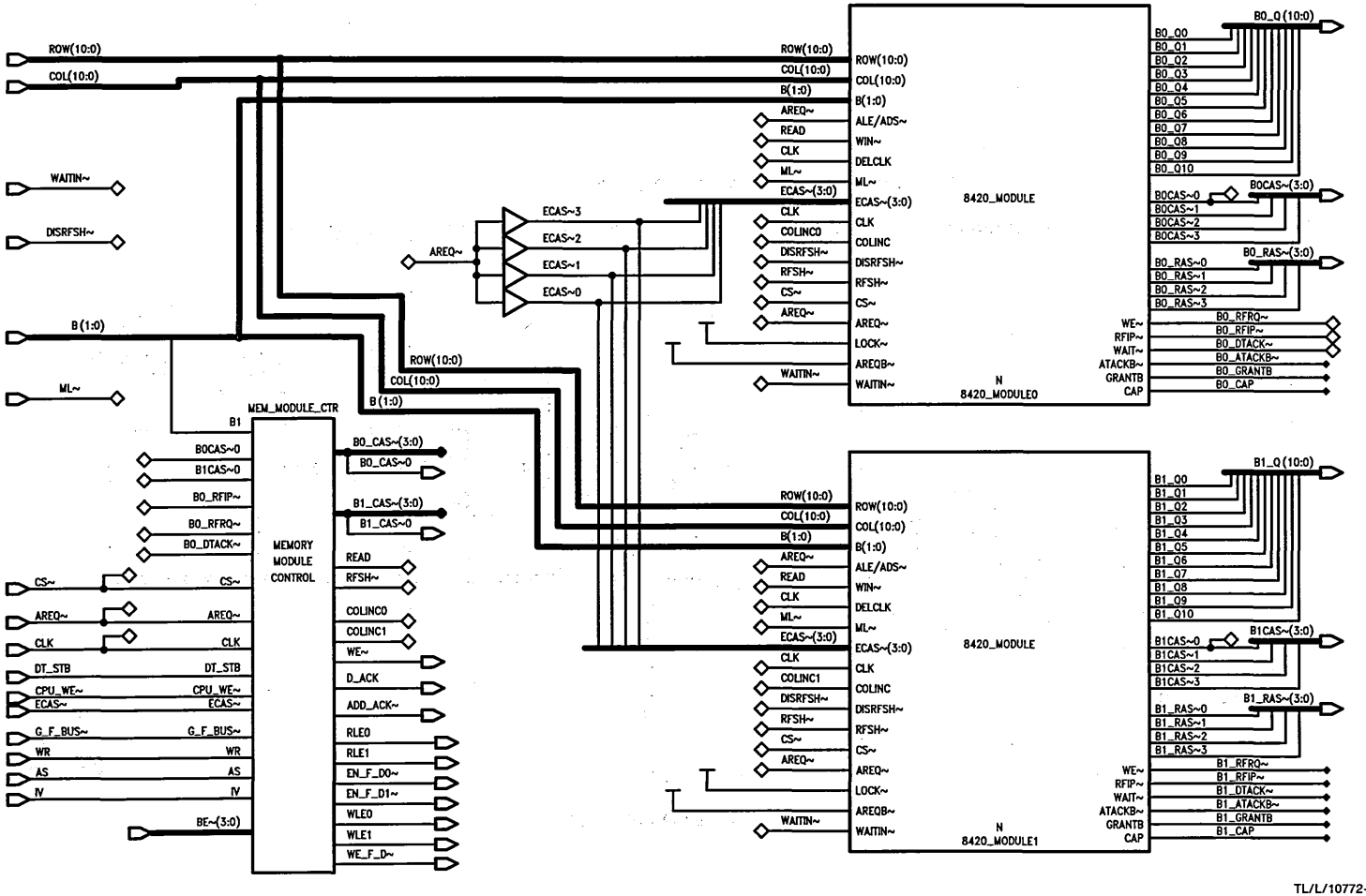


FIGURE 5. Slave Memory Module, DRAM Simulation Diagram



TL/L/10772–9

FIGURE 6. Memory Module Block Diagram for Simulation

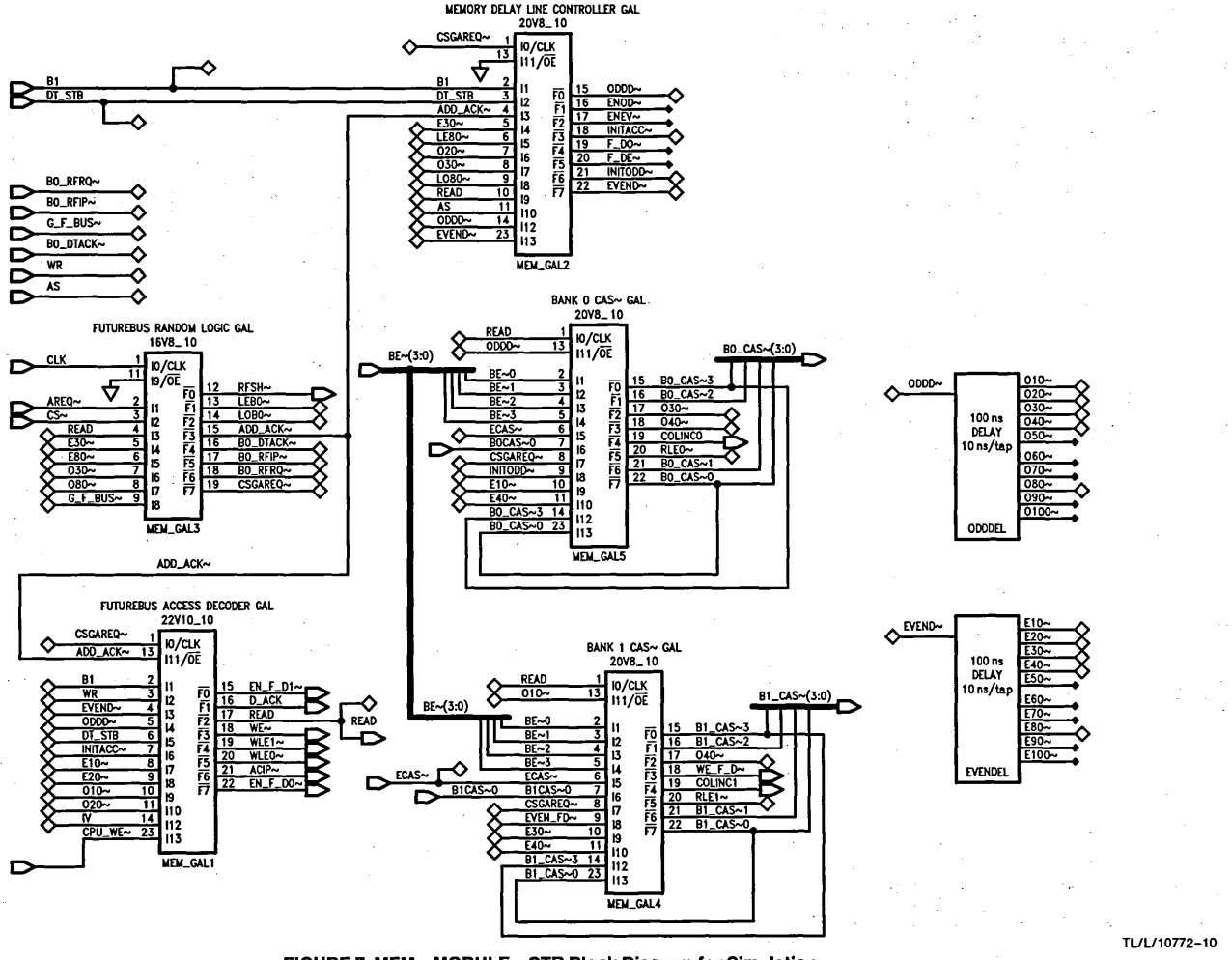


FIGURE 7. MEM__MODULE__CTR Block Diagram for Simulation

This Futurebus+ slave memory board can achieve a 20 mega-transfer/second rate during burst accesses. This performance is approximately 80 Mbytes/sec using 32 bits per memory bank and 160 Mbytes/sec using 64 bits per memory bank. This performance is calculated in the timing section of this application note (Section 5.0 Futurebus+ System Timing).

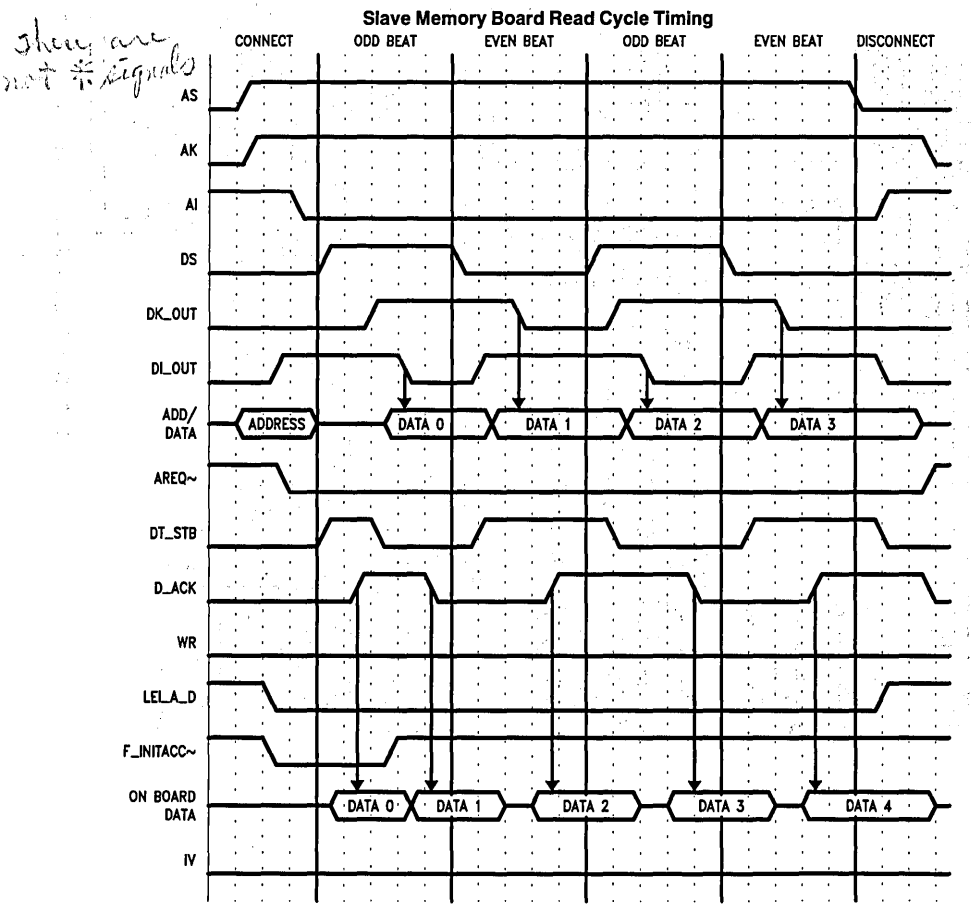
A Futurebus+ access to the DRAM of the slave memory board can be divided into three operations: The Connection Phase, Data Phase, and Disconnection Phase. The Data Phase can be further divided into odd or even beat read operations, odd or even beat write operations, and odd or even beat intervention accesses. The odd or even beat operations of the Data Phase are due to the state of the request-acknowledge handshake operation of the Futurebus+ protocol and are not to be related to the odd or even address banks of DRAM. In other words, a Futurebus+ odd beat read access may be an access to the even or odd address bank of DRAM, dependent upon the state of the least significant double word address bit of the access.

3.0 CONNECTION PHASE

Any access to the slave memory board DRAM begins with the Connection Phase. The Futurebus+ bus master asserts address and command lines to access a particular slave board or boards and asserts the Address Sync line (AS*) to tell the slave boards that the current address and command are valid (see *Figures 8* or *11*).

Upon seeing AS* asserted, the slave board Protocol Controller asserts Address Acknowledge (AK*). Each slave board latches and decodes the address and generates an internal Chip Select (CS~) signal. If the board is not selected, it does not take part during the rest of the current bus tenure except by asserting AI* and releasing AK* during the Disconnection Phase (see *Figure 16*).

The selected slave board decodes the Futurebus+ command, and asserts address status and capability lines on the Futurebus+ for the bus master. The slave board will assert the selected status bit (SL*) if it is selected for the current transaction, and will assert the broadcast status bit (BC*) if the master is accessing an area of memory held in common with other slave boards, such as the CSR space. The beat error status bit (BE*) will be asserted if the slave board is not capable of executing a command or a address or command parity error is detected. The slave memory board will also drive the capability bits. The split response capability bit (SR*) will be released and the compelled capability bit (CO*) will be asserted.



TL/L/10772–23

FIGURE 8. Slave Memory Read Access Timing

The Protocol Controller also asserts the Data Acknowledge Inverse signal (DI*) on the Futurebus+ to prepare for the Data Phase. After a delay to allow for transceiver skew between driving the status and capability bits and AI*, the Protocol Controller releases Address Acknowledge Inverse (AI*). This indicates to the bus master that the status and capability information is valid and that the slave (or slaves) has completed the connection phase.

Internal to the slave board, the Protocol Controller generates control signals to the Memory Module to cause the requested read or write access to take place. The Memory Module Control asserts the internal Address Acknowledge line (ADD_ACK~) to the Protocol Controller to indicate that the DRAM is ready to be accessed.

The slave memory board then enters the odd beat Data Phase. Note that the master may have already entered the odd beat Data Phase by asserting DS* but the slave memory board will not enter the data phase until ADD_ACK~ is received.

4.0 DATA PHASE

The Data Phase consists of odd beat read or write *(Figure 17),* even beat read or write *(Figure 18),* and odd or even beat intervention access operations (*Figures 19* and *20*). The Protocol Controller passes WR* to the Memory Module to indicate whether the master is requesting a read or write. The initial access following the Connection Phase is considered odd beat, as indicated by Data Sync (DS*) asserted (see *Figures 8* or *11*).

The DRAM bank being accessed first (odd or even) is indicated by the lower address bank select input (B1) from the address in the address latch. If AS* becomes released or invalid for any reason before entering the odd or even beat operations, the board would immediately pass into the Disconnection Phase.

If IV* is asserted, the board passes into the Intervention access phase. With IV* released and AS* asserted, the board passes into either an odd beat read (WR* released) or an odd beat write (WR* asserted) operation *(Figure 17).*

The Data Sync (DS*) signal from the bus master tells the slave that either the write command and data are valid to start the write operation or that the read command is valid and the master is ready to receive read data.

The status and capability bits that were driven in the connection phase are also driven in the data phase before it is terminated by the release of DK* or DI*, depending upon whether it is the end of an odd or even data beat. The end of data status bit (ED*) could also be set, during the data phase, if the memory module reads or writes the last double word in a page of the DRAM. This bit signals the master that no further accesses can take place on the slave board.

4.1 Odd Beat Read Operation

With AS*, DI*, and DS* asserted and WR*, IV*, and DK* released, the slave board starts the odd beat read operation (see *Figures 8, 11, 17*).

If it is the first read access after the Connection Phase, the Memory Module performs two operations for the initial odd beat read only. The latching data transceivers are turned around to transmit onto the Futurebus+ and the Memory Module Control is requested to do two read accesses via the DT_STB signal. On each subsequent read access (even or odd beat) only one DRAM access will take place. That access will be a pipelined read access in preparation for the next read request from the master.

The initial read access request (DT_STB asserted) reads the data needed for the current read request from the master. Once this data has been accessed (D_ACK asserted) the Protocol controller will request another read access (DT_STB released) to the next sequential address in preparation for the next read request from the master (even beat read access).

During an initial read access, the first DT_STB edge will always be a rising edge. The DT_STB edge causes the Memory Module Control to send control signals to the DRAM Controller and DRAM so that the addressed data is output through the 74F543 transceivers. The internal Data Output Acknowledge signal (D_ACK) is then asserted to the Protocol Controller from the Memory Module Control. When D_ACK transitions, the data is valid at the slave board 74F543 transceivers ready to be clocked out (CKO_D_S_C) to Futurebus+ via the DS3886 latching transceivers.

The following actions occur for every odd beat read operation: With D_ACK asserted, the requested data and status are clocked (CKO_D_S_C) from the 74F543 transceivers into the DS3886 latching transceivers and onto the Futurebus+. To prepare for the even beat read, DK* is asserted. So as to pipeline data for the next read, the Protocol Controller deasserts DT_STB to request the next sequential piece of data from the DRAM (burst access). This next piece of data is output into the 74F543 transceivers. If this is the initial read operation, F_INITACC~ is released. After a delay to allow for transceiver skew (between the read data, status on Futurebus+ and DI* released), DI* is released to indicate that the data on the Futurebus+ is valid.

When the data for the next (even beat) read is valid at the 74F543 transceivers, D_ACK is deasserted. The board now waits for the bus master to release DS* to indicate that it is ready for the next piece of data. If AS* is released by the master before DS*, the board enters the Disconnection Phase. If DS* is released, the board enters the even beat read operation.

4.2 Even Beat Read Operation

With AS* and DK* asserted and DS*, DI*, WR*, and IV* released, the slave board starts the even beat operation (see *Figures 8, 11, 18*).

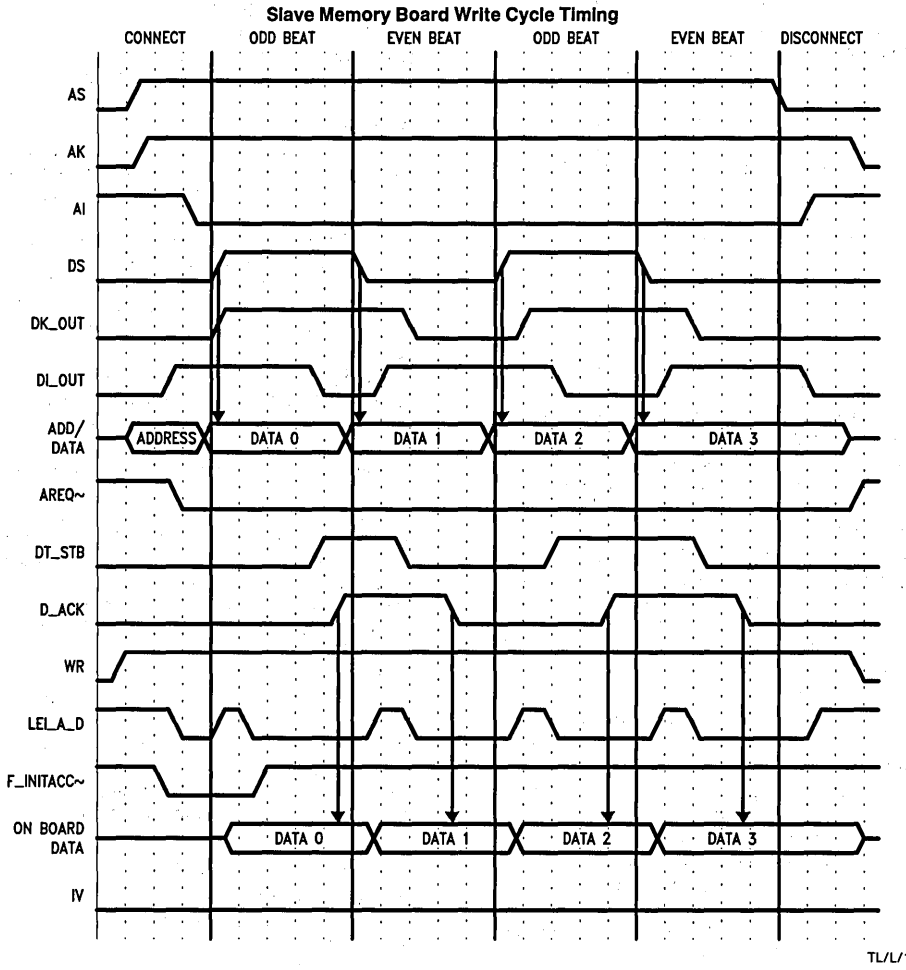
For the even beat read, the current requested data is valid at the 74F543 transceivers and DS* has been released to indicate that the bus master is ready to read data. The DS3886 transceivers are clocked (CKO_D_S_C) and the address and status information is latched in and onto the Futurebus+. DI* is asserted to prepare for the odd beat operation and DT_STB is asserted to the Memory Module Control to read the next piece of sequential data into the 74F543 transceivers.

Following a delay to allow for transceiver skew (between data, status and DK* released) DK* is released to indicate that the next piece of data is valid on the Futurebus+. When the data for next (odd beat) read is valid at the 74F543 transceivers (initiated by the previous DT_STB transition), D_ACK is asserted. The board now waits for the

bus master to assert DS* to indicate its request for the next piece of data. If AS* is released by the master before DS* is asserted, the board enters the Disconnection Phase. If DS* is asserted, the board enters the odd beat read operation.

4.3 Odd Beat Write Operation

With AS*, DI*, DS*, and WR* asserted and IV* and DK* released, the slave board starts the odd beat write operation (see *Figures 9, 12, 17*).



TL/L/10772–24

FIGURE 9. Slave Memory Write Access Timing

With DS* asserted, indicating that the data on the Futurebus+ is valid, the DS3886 latching transceivers latch (LEI_A_D) the data to be written into the DRAM. DK* is asserted to prepare for the even beat write operation.

Status information from the Protocol Controller is clocked (CKO_D_S_C) into the latched transceivers (DS3886) and onto the Futurebus+. If this is the initial write operation F_INITACC~ is deasserted.

DI* is then released, after a delay to allow for transceiver skew, to acknowledge that the status information is valid on Futurebus+ and the write data has been received.

DT_STB is then asserted to the Memory Module Control to write the data latched in the DP3886 latches into the DRAM. When the requested write data has been written into the DRAM the Memory Module Control asserts D_ACK.

The slave board now waits for the bus master to release DS* to indicate that the next piece of data is valid on the Futurebus+. If AS* is released by the master before DS*, the board enters the Disconnection Phase. If DS* is released, the board enters the even beat write operation.

4.4 Even Beat Write Operation

With AS*, DK*, and WR* asserted and DS*, DI*, and IV* released, the slave board starts the even beat write operation (see *Figures 9, 12, 18*).

With DS* released indicating that the data on the Futurebus+ is valid, the DS3886 latching transceivers latch the current data to be written to the DRAM (LEI_A_D). DI* is asserted to prepare for the odd beat write. Status information from the Protocol Controller is clocked (CKO_D_S_C) into the status transceiver and onto the Futurebus+.

DK* is released, after a delay to allow for transceiver skew, to acknowledge that the data has been received and status information is valid on Futurebus+.

DT_STB is deasserted to the Memory Module Control to write the data latched in the DP3886 latches into the DRAM. When the requested write data has been written into the DRAM the Memory Module Control deasserts D_ACK.

The board now waits for the bus master to assert DS* to indicate that the next piece of data is valid on the Futurebus+. If AS* is released before DS* is asserted, the board enters the Disconnection Phase. If DS* is asserted, the board enters the odd beat write operation.

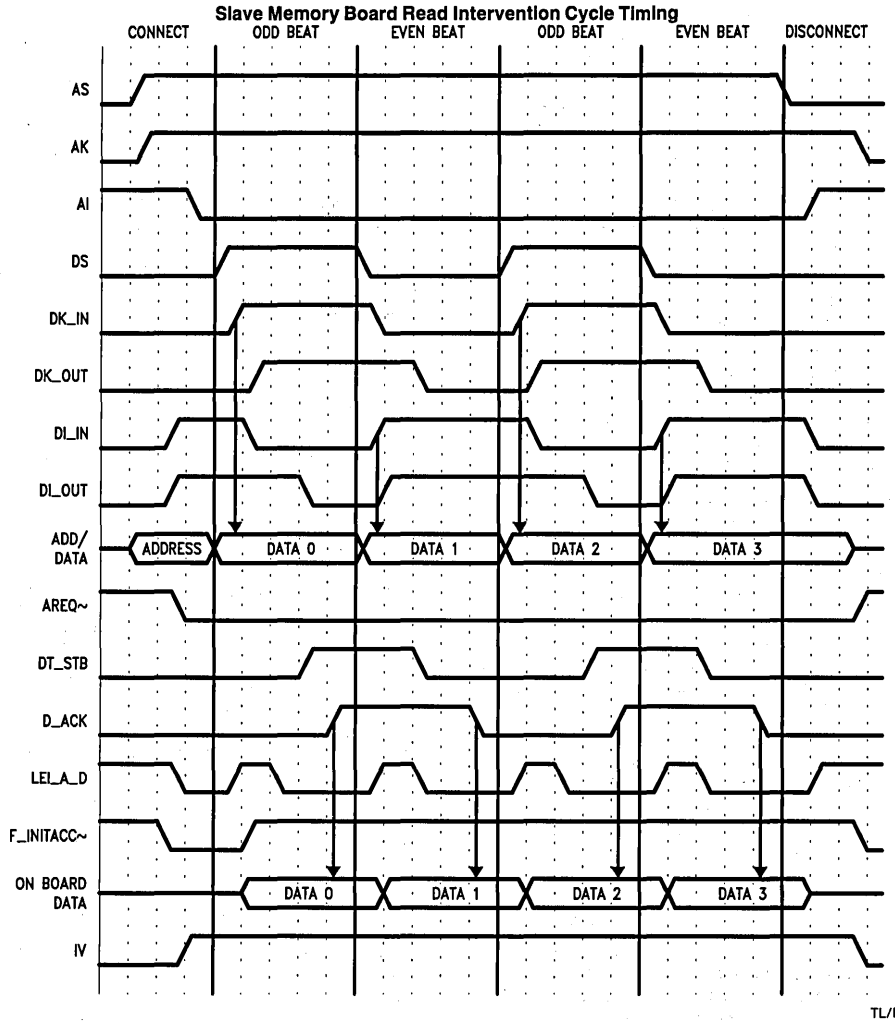
4.5 Read Intervention

When a master performs a read request on Futurebus+ the slave memory board that contains this data may not always be the board that sources this data back to the master. For example, another Futurebus+ cache board may have had a copy of the data and changed it (written to it) but has not yet written this updated data to the slave memory board.

Therefore, when the master selects a memory module to read a block of data, and a cache on another module sees that it has a more up-to-date copy, the cache module will intervene in the transaction by diverting the memory module and providing the data to both the master and the slave memory module. In this case the slave memory module that was going to perform a read access of its memory now instead does a write access to write the more up-to-date data (from the intervening module) into its memory. In other words, the memory module read access has been changed into a wire access due to intervention (see *Figures 10, 13, 19, 20*).

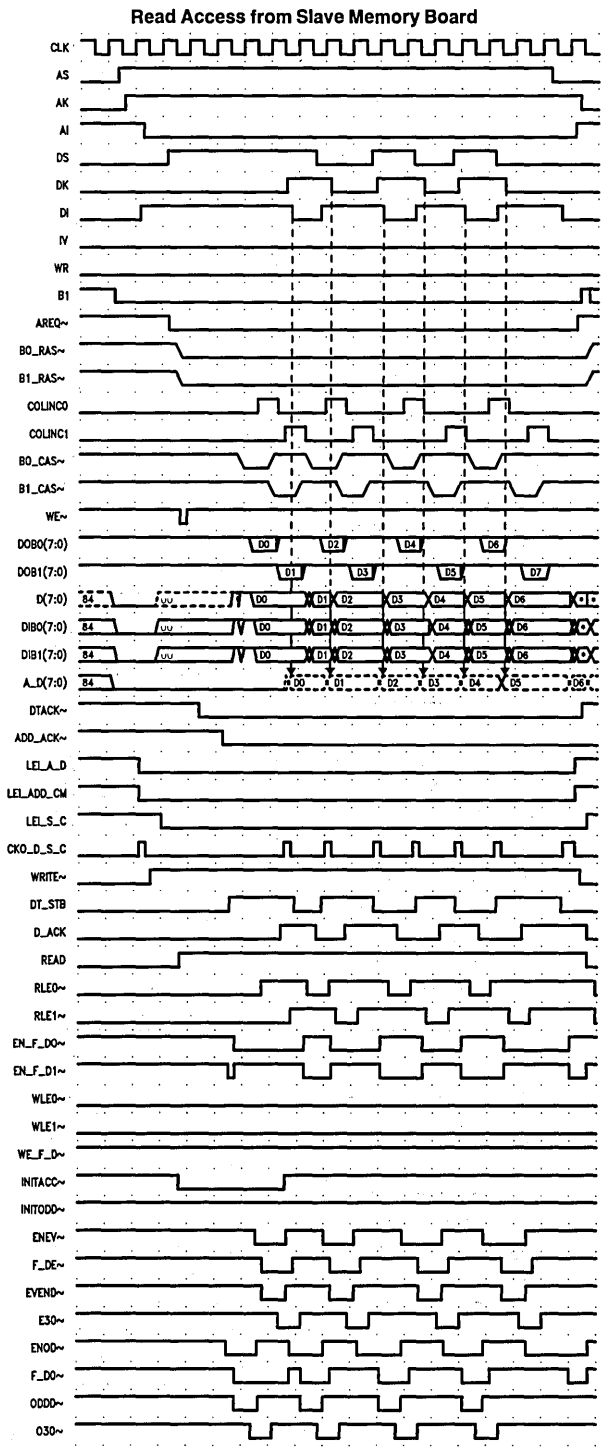
During intervention, the changed data is passed to the slave board and bus master from the owner and the system DRAM is updated with the correct value. To account for this, the initial odd beat read operation must not commence until it has been determined that the current transaction does not involve intervention. For an initial read access, it is assumed that the Intervention line (IV*) will be asserted prior to the transition of AI* filtered during the Connection Phase. The slave memory board of this application note requires that the IV* status bit is asserted before the initial assertion of DS* at the beginning of the odd data beat.

This slave memory board supports intervention through the intervention status bit (IV*) and does not look at the slave write command bit (SW*). This should cause no problems using the compelled mode of operation. The reasons that the IV* status bit was used instead of the SW* command bit are as follows. First, the IV* status bit will be valid on the Futurebus+ earlier then the SW* bit. IV* will be valid earlier on the slave memory board also since the status latch is in fall-thru mode until AI* filtered is released. The slave memory board does not use AI* filtered released to validate IV*. This allows the slave board to guarantee that IV* will be valid before DS* is asserted. This leads into the second reason which is if SW* was used it would have to be validated by DS*. To do this would require an extra delay in the first access of every transaction (to account for transceiver skew) which would effect system performance.



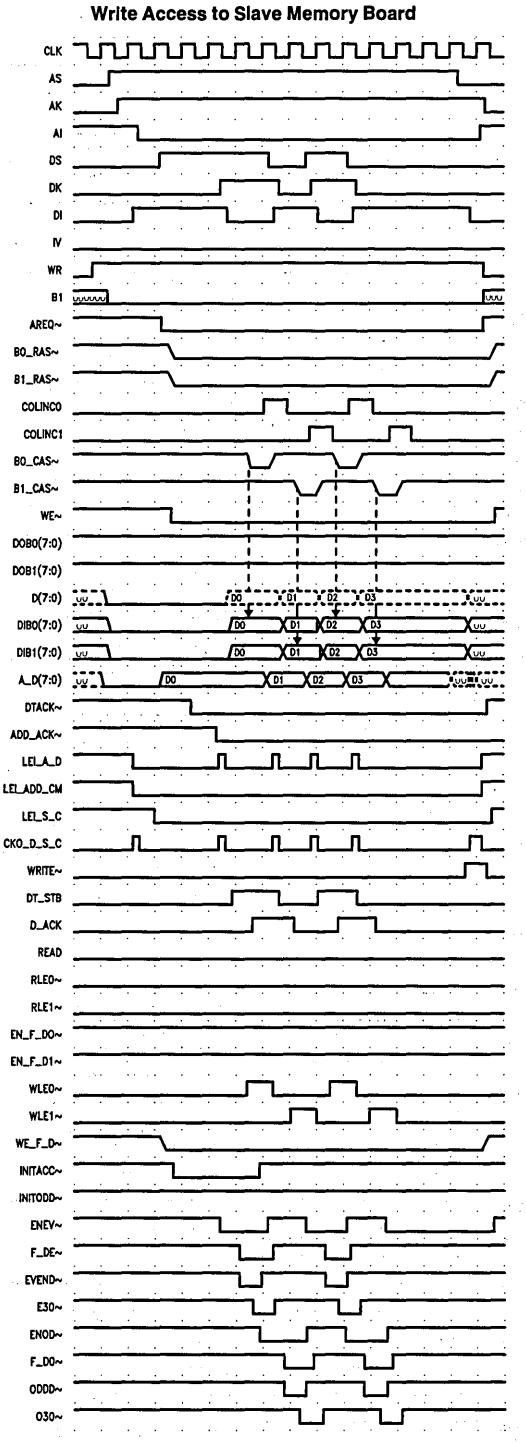
TL/L/10772-25

FIGURE 10. Slave Memory Read with Intervention Timing



TL/L/10772–26

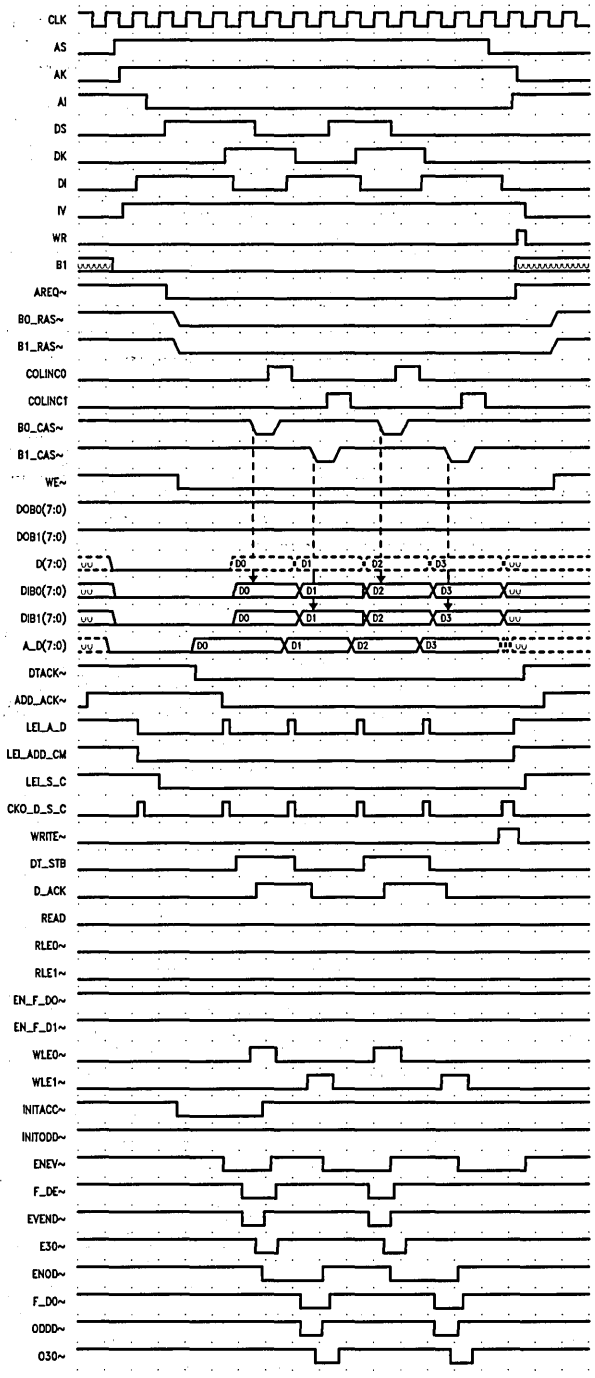
FIGURE 11. Slave Memory Read Access Timing



TL/L/10772–27

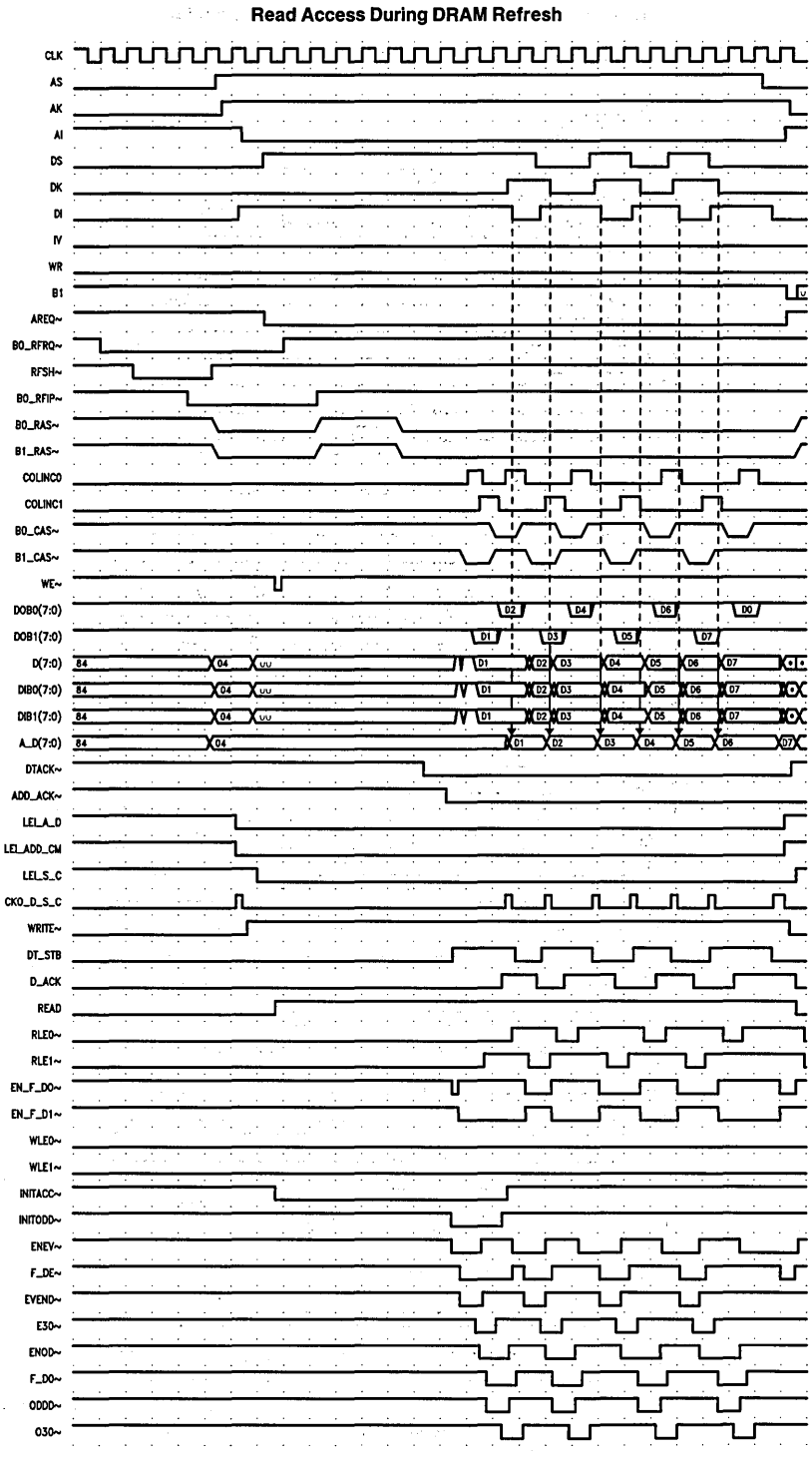
FIGURE 12. Slave Memory Write Access Timing

Read Access To Slave Memory Board with Intervention



TL/L/10772–28

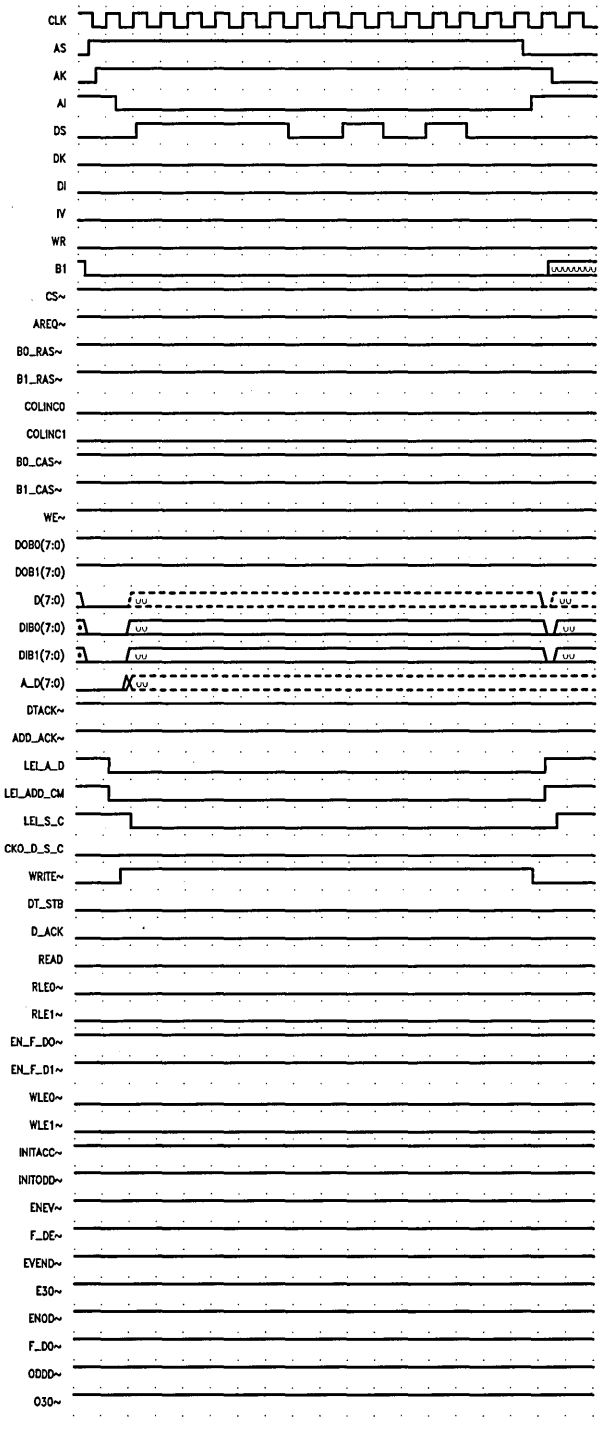
FIGURE 13. Slave Memory Read Access with Intervention Timing



TL/L/10772-29

FIGURE 14. Slave Memory Read During Refresh Timing

Non-CS~ Read Access



TL/L/10772–30

FIGURE 15. Slave Memory Non-CS~ Read Access Timing

The intervention access odd and even beat write is controlled by the data "owner" module with DK* and DI*. During odd beats of the intervention access, only the module sourcing the data (the data "owner") initially asserts DK*. All other modules wait until they detect DK* asserted before asserting their own DK*. During even beats, the data source module initially asserts DI* and the other modules wait until they detect DI* asserted before asserting their own DI*.

DK* and DI* input to the slave board from Futurebus+ will be denoted DK_IN and DI_IN. DK* and DI* from the slave memory board onto the Futurebus+ will be denoted DK and DI. The bus master will monitor the open-collector driven DK* and DI* lines on the Futurebus+ and will not issue the next DS* until DK* and DI* have terminated the present data beat and the master has received the requested data.

4.5.1 Odd Beat Read Intervention

With AS*, IV*, DI*, and DS* asserted, and DK* released, the slave board enters the odd beat read intervention operation (see *Figures 10, 13, 19*). If AS* is released by the master before DS*, the board enters the Disconnection Phase.

DS* asserted indicates that the bus master is ready to read the data from the Futurebus+.

Upon seeing DS* asserted, the intervening module issues the first double-word of data onto the Futurebus+. When this data is valid on Futurebus+, the intervening module asserts DK*. The slave memory board see this as DK_IN asserted and responds by asserting DK*. With DK_IN asserted the DS3886 latching transceivers latch (LEI_A_D) the data to be written into the DRAM.

Status information from the Protocol Controller is clocked (CKO_D_S_C) into the latched transceivers (DS3886) and onto the Futurebus+. If this is the initial access operation F_INITACC~ is deasserted.

DI* is then released, after a delay to allow for transceiver skew, to acknowledge to the master and intervening module that the status information is valid on Futurebus+ and the write data has been received.

DT_STB is then asserted to the Memory Module Control to write the data latched in the DP3886 latches into the DRAM. When the requested write data has been written into the DRAM the Memory Module Control asserts D_ACK.

The board now waits for the bus master to release DS* to indicate that the bus master is ready for the next piece of data. If AS* is released by the bus master before DS*, the slave board and intervening module enter the Disconnection Phase. If DS* is released, the board enters the even beat read intervention phase.

4.5.2 Even Beat Read Intervention

With AS*, IV*, and DK* asserted, and DS* and DI* released, the slave board enters the even beat read intervention operation (see *Figures 10, 13, 20*). If AS* is released by the master before DS* is asserted, the board enters the Disconnection Phase.

With DS* released, the intervening module asserts the next double-word of data onto the Futurebus+. When the data is valid on Futurebus+, the intervening module asserts DI*.

The slave board sees this as DI_IN asserted and responds by asserting DI*. With DI_IN asserted the DS3886 latching transceivers latch (LEI_A_D) the data to be written into the DRAM.

Status information from the Protocol Controller is clocked (CKO_D_S_C) into the latched transceivers (DS3886) and onto the Futurebus+.

DK* is then released, after a delay to allow for transceiver skew, to acknowledge to the master and intervening module that the status information is valid on Futurebus+ and the write data has been received.

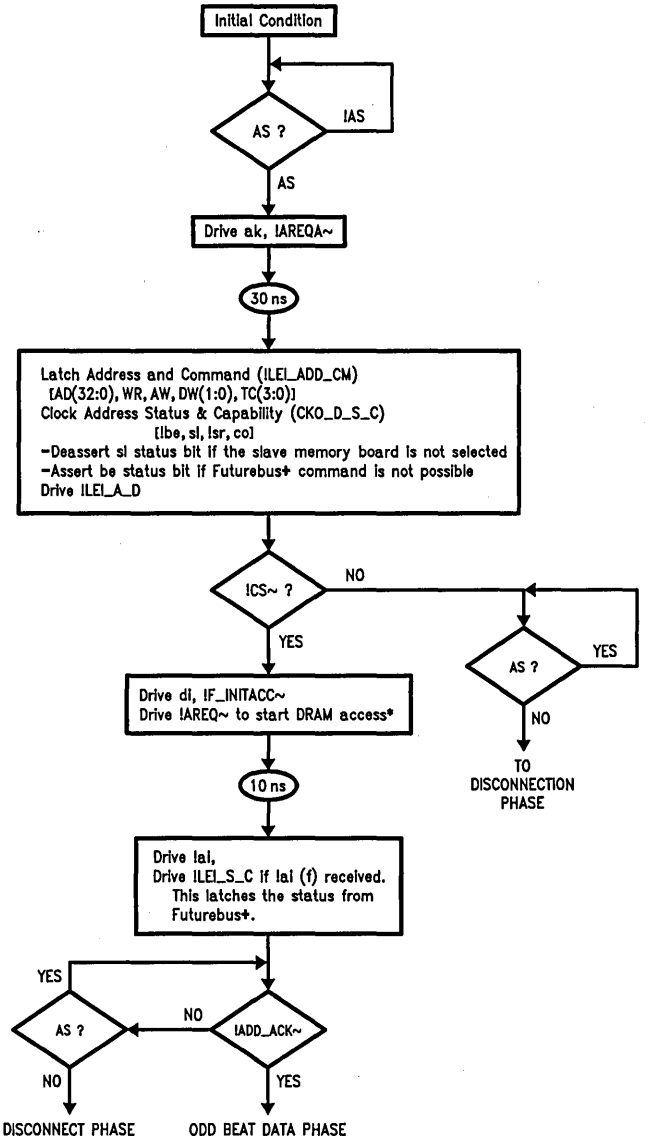
DT_STB is then deasserted to the Memory Module Control to write the data latched in the DP3886 latches into the DRAM. When the requested write data has been written into the DRAM the Memory Module Control deasserts D_ACK.

The board now waits for the bus master to assert DS* to indicate that the bus master is ready for the next piece of data. If AS* is released by the bus master before DS* is asserted, the slave board and intervening module enter the Disconnection Phase. If DS* is asserted, the board enters the odd beat read intervention phase.

4.6 Disconnection Phase

The bus master releases AS* to indicate to the participating slaves that the transaction is being terminated (see *Figures 8, 11, 21*).

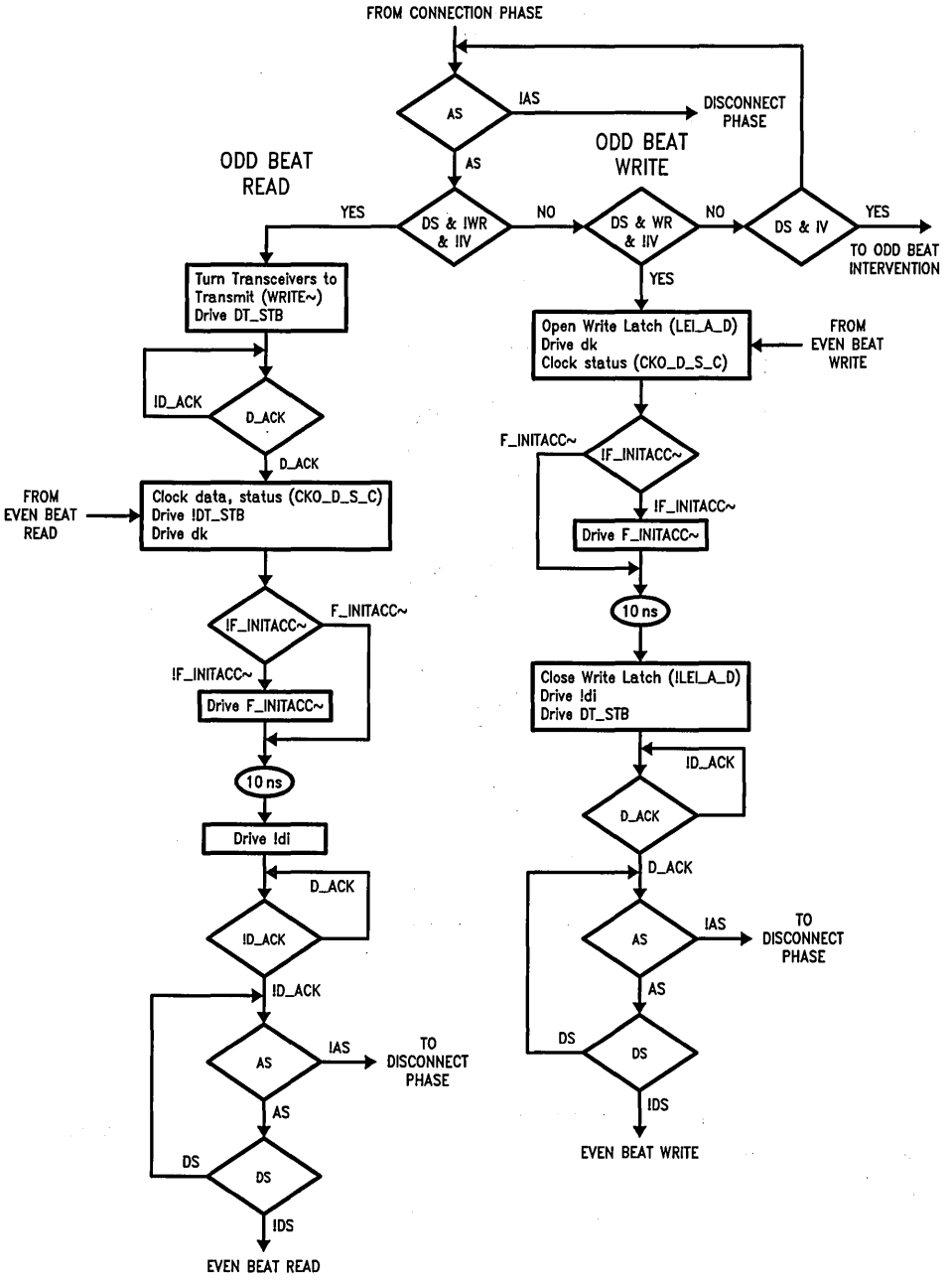
Upon detecting AS* released, the slave board, evaluates the command lines and asserts AI* and deasserts DK* and DI* on the Futurebus+. Internal to the slave board, DT_STB is deasserted. The DS3886 latching transceivers are opened and the address latch is set to fall-through, preparing the slave board to sense and analyze the next address asserted on the Futurebus+. The status and capability from the Protocol Controller are clocked onto the Futurebus+ for the bus master to analyze. Finally, as soon as the memory module has finished reading or writing data from the previous data beat the slave board releases AK* and the transaction ends. The slave board is now ready to enter the Connection Phase for the next transaction.



TL/L/10772-31

*Note that the predecessor to AREQ~ (AREQA~) is initiated from AS at the beginning of the connection phase.

FIGURE 16. Connection Phase State Diagram



TL/L/10772–32

FIGURE 17. Odd Beat Data Phase State Diagram

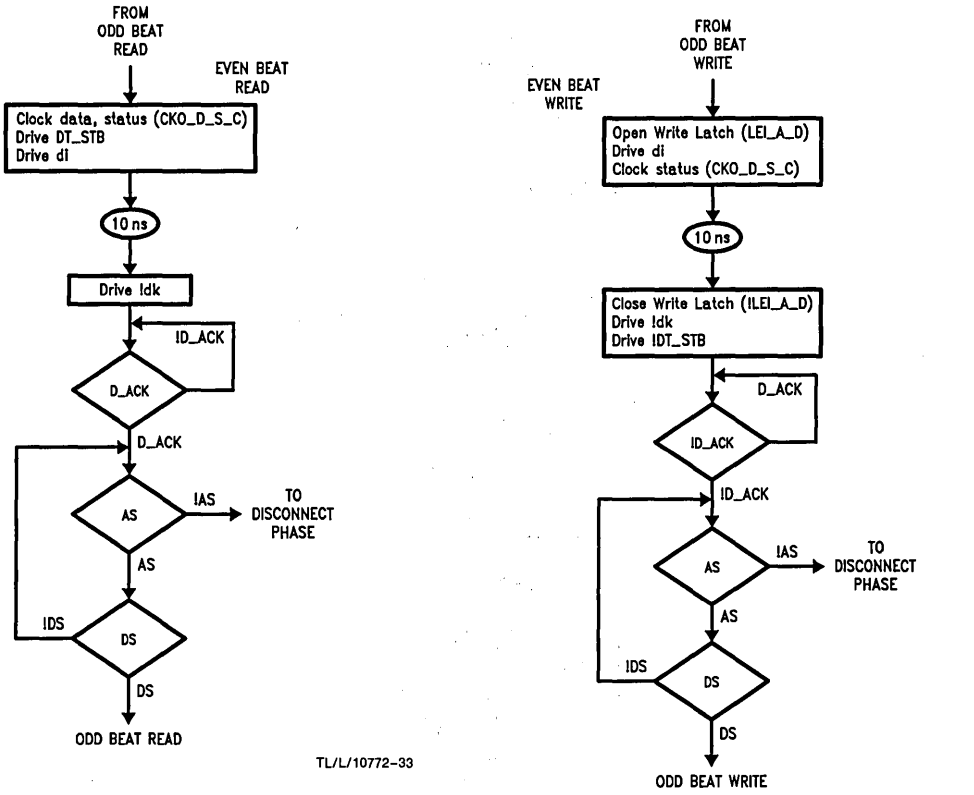
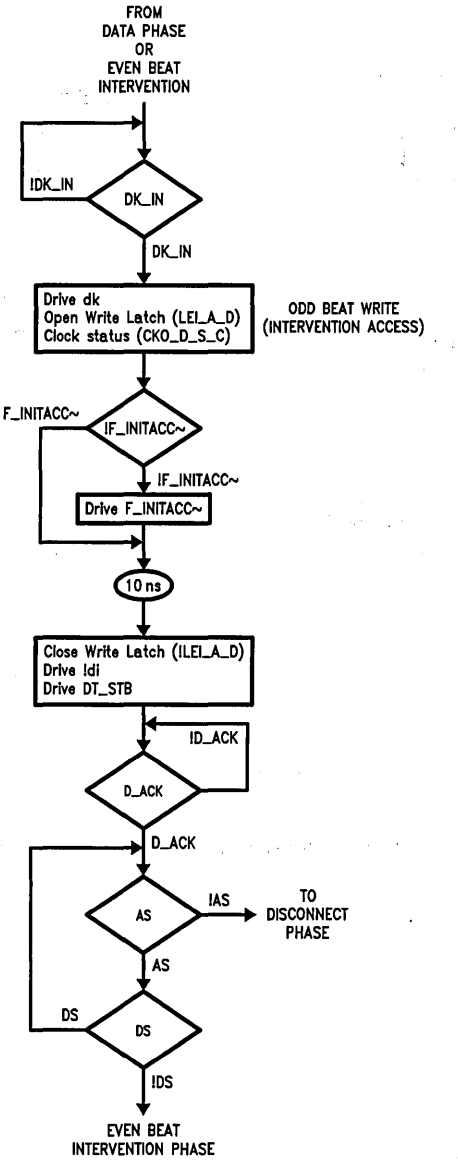
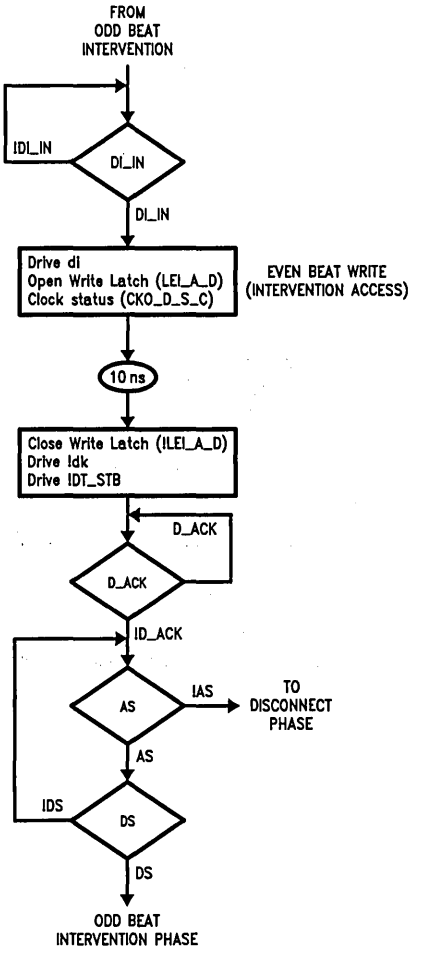


FIGURE 18. Even Beat Data Phase State Diagram



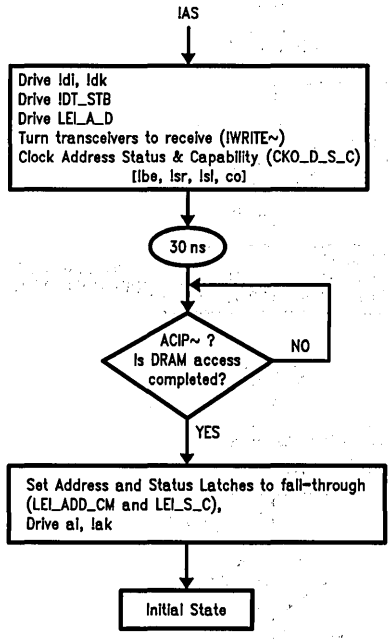
TL/L/10772–35

FIGURE 19. Odd Beat Intervention State Diagram



TL/L/10772-36

FIGURE 20. Even Beat Intervention State Diagram



TL/L/10772–37

FIGURE 21. Disconnect Phase State Diagram

5.0 FUTUREBUS+ SYSTEM TIMING CALCULATIONS

5.1 Maximum time from the master generating AS* to the master receiving AI* is 116 ns.

Parameter (Maximum Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Produces AS* (Assume 10 ns PAL)	10	10
Master's Handshake Transceiver (DS3884)	7	17
Futurebus+ Delay	7.5	24.5
Slaves Handshake Transceiver (DS3884)	7	31.5
Delay Line (± 2 ns), 40 ns Tap ***	42	73.5
Slaves PAL Produces AI* (Assume 10 ns PAL)	10	83.5
Slaves Handshake Transceiver (DS3884)	7	90.5
Futurebus+ Delay	7.5	98
Masters Handshake Transceiver (DS3884 Filtering AI* with Glitch Reject Filter of 10 ns, Cn = 10)	18	116

Typical time from the master generating AS* to the master receiving AI* is 89 ns.

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Produces AS* (Assume 110 ns PAL)	7	7
Master's Handshake Transceiver (DS3884)	5	12
Futurebus+ Delay	2	14
Slaves Handshake Transceiver (DS3884)	5	19
Delay Line (± 2 ns), 40 ns Tap ***	40	59
Slaves PAL Produces AI* (Assume 10 ns PAL)	7	66
Slaves Handshake Transceiver (DS3884)	5	71
Futurebus+ Delay	2	73
Masters Handshake Transceiver (DS3884 Filtering AI* with Glitch Reject Filter of 10 ns, Cn = 10)	16	89

***The 40 ns Delay is composed of a 30 ns delay that is used to determine whether the slave memory board is selected for the current access. At this time the status and capability bits are also driven out to the Futurebus+. The next 10 ns delay is used to delay AI* from the slave memory board to guarantee that the status and capability are valid before asserting AI*.

5.2 Maximum time during a read access from the master generating AS* to the master receiving DI* is 359 ns.

Parameter (Maximum Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Produces AS* (Assume 10 ns PAL)	10	10
Master's Handshake Transceiver (DS3884)	7	17
Futurebus+ Delay	7.5	24.5
Slaves Handshake Transceiver (DS3884)	7	31.5
Two Clocks Maximum to Produce AREQ~	80	111.5
Slaves PAL Produces AREQ~ (Assume 10 ns PAL)	10	121.5
Two Clocks at 25 MHz (used in Generating ADD_ACK~)	80	201.5
Memory Module Control Producing ADD_ACK~ from PAL	10	211.5

← The odd beat DS* from the master must have been received by this time or it will become the determining factor in the maximum time to DI* (see 5.5 below).

Parameter (Maximum Delays, continued)	(In nano-seconds) Single Delay	Total Delay
Slave Protocol Controller Drives DT_STB to Memory Module (Assume 10 ns PAL)	10	221.5
Initial Read Access DT_STB to D_ACK, (see Part 2 of This Application Note, Memory Module Design Section 4.2).	94	315.5
Delay Line (± 2 ns), 10 ns Tap	12	327.5
Slaves PAL Drives DI*	10	337.5
Slaves Handshake Transceiver (DS3884)	7	344.5
Futurebus+ Delay	7.5	352
Master's Handshake Transceiver (DS3884)	7	359

Typical time during a read access from the master generating AS* to the master receiving DI* is 287 ns.

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Produces AS* (Assume 10 ns PAL)	7	7
Master's Handshake Transceiver (DS3884)	5	12
Futurebus+ Delay	2	14
Slaves Handshake Transceiver (DS3884)	5	19
One and One Half Clocks Typical to Produce AREQ~	60	79
Slaves PAL Produces AREQ~ (Assume 10 ns PAL)	7	86
Two Clocks at 25 MHz (used in Generating ADD_ACK~)	80	166
Memory Module Control Producing ADD_ACK~ from PAL	7	173

← The odd beat DS* from the master must have been received by this time or it will become the determining factor in the typical time to DI* (see 5.5 below).

Parameter (Typical Delays, continued)	(In nano-seconds) Single Delay	Total Delay
Slave Protocol Controller Drives DT_STB to Memory Module (Assume 10 ns PAL)	7	180
Initial Read Access DT_STB to D_ACK, (see Part 2 of This Application Note, Memory Module Design Section 4.2)	78	258
Delay Line (± 2 ns) 10 ns Tap	10	268
Slaves PAL Drives DI*	7	275
Slaves Handshake Transceiver (DS3884)	5	280
Futurebus+ Delay	2	282
Master's Handshake Transceiver (DS3884)	5	287

5.4 Maximum time during a write access from the master generating AS* to the master receiving DI* is 255 ns.

Parameter (Maximum Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Produces AS* (Assume 10 ns PAL)	10	10
Master's Handshake Transceiver (DS3884)	7	17
Futurebus+ Delay	7.5	24.5
Slaves Handshake Transceiver (DS3884)	7	31.5
Two Clocks Maximum to Produce AREQ~	80	111.5
Slaves PAL Produces AREQ~ (Assume 10 ns PAL)	10	121.5
Two Clocks at 25 MHz (used in Generating ADD_ACK~)	80	201.5
Memory Module Control Producing ADD_ACK~ from PAL	10	211.5

← The odd beat DS* from the master must have been received by this time or it will become the determining factor in the maximum time to DI* (see 5.5 below)

Parameter (Maximum Delays, continued)	(In nano-seconds) Single Delay	Total Delay
Delay Line (± 2 ns), 10 ns Tap	12	223.5
Slaves PAL Drives DI*	10	233.5
Slaves Handshake Transceiver (DS3884)	7	240.5
Futurebus+ Delay	7.5	248
Master's Handshake Transceiver (DS3884)	7	255

Typical time during a write access from the master generating AS* to the master receiving DI* is 202 ns.

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Produces AS* (Assume 10 ns PAL)	7	7
Master's Handshake Transceiver (DS3884)	5	12
Futurebus+ Delay	2	14
Slaves Handshake Transceiver (DS3884)	5	19
One and One Half Clocks Typical to Produce AREQ~	60	79
Slaves PAL Produces AREQ~ (Assume 10 ns PAL)	7	86
Two Clocks at 25 MHz (used in Generating ADD_ACK~)	80	166
Memory Module Control Producing ADD_ACK~ from PAL	7	173

← The odd beat DS* from the master must have been received by this time or it will become the determining factor in the typical time to DI* (see 5.5 below)

Parameter (Typical Delays, continued)	(In nano-seconds) Single Delay	Total Delay
Delay Line (± 2 ns), 10 ns Tap	10	183
Slaves PAL Drives DI*	7	190
Slaves Handshake Transceiver (DS3884)	5	195
Futurebus+ Delay	2	197
Master's Handshake Transceiver (DS3884)	5	202

5.4 Maximum time during a read access with intervention from the master generating AS* to the master receiving DI* is 359 ns.

Parameter (Maximum Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Produces AS* (Assume 10 ns PAL)	10	10
Master's Handshake Transceiver (DS3884)	7	17
Futurebus+ Delay	7.5	24.5
Slaves Handshake Transceiver (DS3884)	7	31.5
Two Clocks Maximum to Produce AREQ~	80	111.5
Slaves PAL Produces AREQ~ (Assume 10 ns PAL)	10	121.5
Two Clocks at 25 MHz (used in Generating ADD_ACK~)	80	201.5
Memory Module Control Producing ADD_ACKL~ from PAL	10	211.5

← The odd beat DK* from the intervening module must have been received by this time or it will become the determining factor in the maximum time to DI*.

Parameter (Maximum Delays, continued)	(In nano-seconds) Single Delay	Total Delay
Slave Protocol Controller Drives DT_STB to Memory Module (Assume 10 ns PAL)	10	221.5
Initial Read Access DT_STB to D_ACK, (see Part 2 of This Application Note, Memory Module Design Section 4.2)	94	315.5
Delay Line (± 2 ns), 10 ns Tap	12	327.5
Slaves PAL Drives DI*	10	337.5
Slaves Handshake Transceiver (DS3884)	7	344.5
Futurebus+ Delay	7.5	352
Master's Handshake Transceiver (DS3884)	7	359

Typical time during a read access with intervention from the master generating AS* to the master receiving DI* is 287 ns.

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Produces AS* (Assume 10 ns PAL)	7	7
Master's Handshake Transceiver (DS3884)	5	12
Futurebus+ Delay	2	14
Slaves Handshake Transceiver (DS3884)	5	19
One and One Half Clocks Typical to Produce AREQ~	60	79
Slaves PAL Produces AREQ~ (Assume 10 ns PAL)	7	86
Two Clocks at 25 MHz (used in Generating ADD_ACK~)	80	166
Memory Module Control Producing ADD_ACK~ from PAL	7	173

← The odd beat DK* from the intervening module must have been received by this time or it will become the determining factor in the typical time to DI*.

Parameter (Typical Delays, continued)	(In nano-seconds) Single Delay	Total Delay
Slave Protocol Controller Drives DT_STB to Memory Module (Assume 10 ns PAL)	7	180
Initial Read Access DT_STB to D_ACK, (see Part 2 of This Application Note, Memory Module Design Section 4.2)	78	258
Delay Line (± 2 ns), 10 ns Tap	10	268
Slaves PAL Drives DI*	7	275
Slaves Handshake Transceiver (DS3884)	5	280
Futurebus+ Delay	2	282
Master's Handshake Transceiver (DS3884)	5	287

5.5 Maximum time allowed (worst case conditions) for the master to generate odd beat DS* from the reception of AI* to guarantee that the maximum time from AS* to master generating the even beat DS* (calculated above in 5.2 and 5.3) is accurate.

Point in time where odd beat DS* must be valid (see Sections 5.2 and 5.3 above)	211.5
Maximum time from AS* to masters reception of AI * (see Section 5.1 above)	−116
Masters Handshake Transceiver (DS8334)	−7
Futurebus+ Delay	−7.5
Slaves Handshake Transceiver (DS3884)	−7
Maximum time for master to generate odd beat DS* from the masters reception of AI*	74 ns

Maximum time allowed (typical conditions) for master to generate odd beat DS* from the reception of AI* to guarantee that the typical time from AS* to master generating the even beat DS* (calculated above in 5.2 and 5.3) is accurate.

Point in time where odd beat DS* must be valid (see Sections 5.2 and 5.3 above)	173
Typical time from AS* to masters reception of AI* (see Section 5.1 above)	−89
Masters Handshake Transceiver (DS3884)	−5
Futurebus+ Delay	−2
Slaves Handshake Transceiver (DS3884)	−5
Typical time for master to generate odd beat DS* from the masters reception of AI*	72 ns

5.6 Maximum time during a slave read access (master write access) from the master generating DS to the master generating the next edge of DS is 89 ns. This is based on the assumption that the DT_STB request/D_ACK acknowledge between the protocol controller and the memory module (of the I/O board) will be less than or equal to the speed of the Futurebus+ request/acknowledge handshake signals (DS, DK, and DI).

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Master's GAL Generates DS (Assume 10 ns GAL)	10	10
Master's Handshake Transceiver (DS3884)	7.5	17.5
Futurebus+ Delay	7.5	25
Slaves Handshake Transceiver (DS3884)	7.5	32.5
Clock Out to Futurebus+ Next Piece of Data to Write		
Delay Line (±2 ns), 10 ns tap***	12	44.5
Slaves GAL Produces DK or DI (Assume 10 ns GAL)	10	54.5
Slaves Handshake Transceiver, DK/DI (DS3884)	7.5	62
Futurebus+ Delay	7.5	69.5
Master's Handshake Transceiver, DK/DI (DS3884)	7.5	77
Delay Line (±2 ns), 10 ns tap***	12	89
Master's GAL Generates DS (Assume 10 ns GAL)	—	—

Typical time during a slave read access (master write access) from the master generating DS to the master generating the next edge of DS is 58 ns.

***The 10 ns delay line used to compensate for possible skew between the Latched Transceiver (DS3886) and the Handshake Transceiver (DS3884).

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Master's GAL Generates DS* (Assume 10 ns GAL)	7	7
Master's Handshake Transceiver, DS (DS3884)	5	12
Futurebus+ Delay	2	14
Slaves Handshake Transceiver, DS (DS3884)	5	19
Clock Out to Futurebus+ Next Piece of Data to Write to the Master		
Delay Line (±2 ns), 10 ns Tap***	10	29
Slaves GAL Produces DK or DI (Assume 10 ns GAL)	7	36
Slaves Handshake Transceiver, DK/DI (DS3884)	5	41
Futurebus+ Delay	2	43
Master's Handshake Transceiver, DK/DI (DS3884)	5	48
Delay Line (±2 ns), 10 ns Tap	10	58
Master's GAL Generates DS (Assume 10 ns GAL)	—	—

This gives a typical transfer rate of greater than 17 Mega-Transfers/second during the Data Phase of the Futurebus+ Transfer.

5.7 Maximum time during a slave write access (master read access) from the master generating DS to the master generating the next edge of DS is 89 ns. This is based upon the assumption that the DT_STB request/D_ACK acknowledge between the protocol controller and the memory module (of the I/O board) will be less than or equal to the speed of the Futurebus+ request/acknowledge handshake signals (DS, DK, and DI).

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Master's GAL Generates DS (Assume 10 ns GAL)	10	10
Master's Handshake Transceiver, DS (DS3884)	7.5	17.5
Futurebus+ Delay	7.5	25
Slaves Handshake Transceiver, DS (DS3884)	7.5	32.5
Delay Line (±2 ns), 10 ns Tap***	12	44.5
Slaves GAL Produces DK or DI (Assume 10 ns GAL)	10	54.5
Slaves Handshake Transceiver, DK/DI (DS3884)	7.5	62
Futurebus+ Delay	7.5	69.5
Master's Handshake Transceiver, DK/DI (DS3884)	7.5	77
Clock Out to Futurebus+ Next Piece of Data to Write to the Slave		
Delay Line (±2 ns), 10 ns Tap***	12	89
Master's GAL Generates DS (Assume 10 ns GAL)	—	—

***The 10 ns delay line used to compensate for possible skew between the Latched Transceiver (DS3886) and the Handshake Transceiver (DS3884).

Typical time during a slave write access (master read access) from the master generating DS to the master generating the next edge of DS is 58 ns.

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Master's GAL Generates DS (Assume 10 ns GAL)	7	7
Master's Handshake Transceiver, DS (DS3884)	5	12
Futurebus+ Delay	2	14
Slaves Handshake Transceiver, DS (DS3884)	5	19
Delay Line (±2 ns), 10 ns Tap***	10	29
Slaves GAL Produces DK or DI (Assume 10 ns GAL)	7	36
Slaves Handshake Transceiver, DK/DI (DS3884)	5	41
Futurebus+ Delay	2	43
Master's Handshake Transceiver, DK/DI (DS3884)	5	48
Clock Out to Futurebus+ Next Piece of Data to Write to the Slave		
Delay Line (±2 ns), 10 ns Tap***	10	58
Master's GAL Generates DS (Assume 10 ns GAL)	—	—

This gives a typical transfer rate of greater than 17 Mega-Transfers/second during the Data Phase of the Futurebus+ Transfer.

***The 10 ns delay line used to compensate for possible skew between the Latched Transceiver (DS3886) and the Handshake Transceiver (DS3884).

5.8 Maximum time during read accesses with intervention (master must use filtered DK* and DI*) from the master generating DS* to the master receiving DK* or DI* is 129.5 ns

Parameter (Maximum Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Generates DS* (Assume 10 ns PAL)	10	10
Master's Handshake Transceiver (DS3884)	7	17
Futurebus+ Delay	7.5	24.5
Intervening Module Handshake Transceiver	7	31.5
Delay Line (± 2 ns), 10 ns tap ***	12	43.5
Intervening Module PAL Produces DK* or DI* (Assume 10 ns PAL)	10	53.5
Intervening Module Handshake Transceiver	7	60.5
Futurebus+ Delay	7.5	68
Slaves Handshake Transceiver (DS3884)	7	75
Delay Line (± 2 ns), 10 ns tap ***	12	87
Slaves PAL Produces DK* or DI* (Assume 10 ns PAL)	10	97
Slaves Handshake Transceiver (DS3884)	7	104
Futurebus+ Delay	7.5	111.5
Masters Handshake Transceiver (DS3884 Filtering DI* or DK* with Glitch Reject Filter of 10 ns, (Cn = 10)	18	129.5

Typical time during read accesses with intervention from the master generating DS* to the master receiving DK* or DI* is 88 ns.

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Master's PAL Generates DS* (Assume 10 ns PAL)	7	7
Master's Handshake Transceiver (DS3884)	5	12
Futurebus+ Delay	2	14
Intervening Module Handshake Transceiver	5	19
Delay Line (± 2 ns), 10 ns tap ***	10	29
Intervening Module PAL Produces DK* or DI* (Assume 10 ns PAL)	7	36
Intervening Module Handshake Transceiver	5	41
Futurebus Delay+	2	43
Slaves Handshake Transceiver (DS3884)	5	48
Delay Line (± 2 ns), 10 ns tap ***	10	58
Slaves PAL Produces DK* or DI* (Assume 10 ns PAL)	7	65
Slaves Handshake Transceiver (DS3884)	5	70
Futurebus+ Delay	2	72
Masters Handshake Transceiver (DS3884 Filtering DI* or DK* with Glitch Reject Filter of 10 ns, Cn = 10)	16	88

***The 10 ns delay line used in the slave and intervening module board is needed to provide for the skew between the latched transceivers (DS3886) and the handshake transceiver (DS3884). This allows the slave and intervening module board to guarantee that the read data or status is valid on the Futurebus+ by the time DK* or DI* is valid on Futurebus+.

5.9 Computation of the Delay Necessary to Allow

a. Master to guarantee the command setup to AS* or;

b. Master to guarantee the write data and command setup to DS* or;

c. Slave to guarantee status and capability setup to AI* or;

d. Slave to guarantee read data and status setup to DK* or DI*.

Maximum clock to data valid on Futurebus+ through the DS3886 latched transceiver	12 ns
Minimum handshake signal (AS*, AK*, AI*, DS*, DK*, DI*) into output valid on Futurebus+ through the DS3884 handshake transceiver	−3.5 ns
Assumed Maximum skew between outputs on the PAL22V10-10 (10 ns 22V10, "CKO__D__S__C" clock for data, status and capability out to Futurebus+ versus "DK*, DI* and AI*")	+ 1.5 ns
Delay line necessary to guarantee data, status, capability and command setup to AS*, AK*, AI*, DS*, DK* and DI*	10 ns

***Note that this application note uses a 10 ns delay line. Delay lines generally have a 2 ns, ±5%, whichever is greater skew specification. Therefore, a 10 ns delay line can only guarantee an 8 ns delay. In the worst case timing this would not be acceptable. In this case a 12 ns delay line should be used for the AS, DS, DK__IN, DI__IN, and F__ACK delay lines (see *Figure 4*).

5.10 Disconnection Phase, Maximum time from the master receiving DK* or DI* (from the last data beat) to the master receiving AK* released is 116 ns.

Parameter (Maximum Delays)	(In nano-seconds) Single Delay	Total Delay
Masters delay line (± 2 ns) 10 ns tap. Note that AS* is being produced from the reception of the slave DK* or DI*	12	12
Master's PAL Produces AS* (Assume 10 ns PAL)	10	20
Master's Handshake Transceiver (DS3884)	7	27
Futurebus+ Delay	7.5	34.5
Slaves Handshake Transceiver (DS3884)	7	41.5
Delay Line (± 2 ns) 30 ns Tap ***	32	73.5
Slaves PAL Produces AK* (Assume 10 ns PAL)	10	83.5
Slaves Handshake Transceiver (DS3884)	7	90.5
Futurebus+ Delay	7.5	98
Masters Handshake Transceiver (DS3884 Filtering AK* with Glitch Reject Filter of 10 ns, Cn = 10)	18	116

Disconnection Phase, Typical time from the master receiving DK* or DI* (from the last data beat) to the master receiving AK* released is 89 ns.

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Masters delay line (± 2 ns) 10 ns tap. Note that AS* is being produced from the reception of the slave DK* or DI*	10	10
Master's PAL Produces AS* (Assume 10 ns PAL)	7	17
Master's Handshake Transceiver (DS3884)	5	22
Futurebus+ Delay	2	24
Slaves Handshake Transceiver (DS3884)	5	29
Delay Line (± 2 ns) 20 ns tap ***	30	59
Slaves PAL Produces DK* or DI* (Assume 10 ns PAL)	7	66
Slaves Handshake Transceiver (DS3884)	5	71
Futurebus+ Delay	2	73
Masters Handshake Transceiver (DS3884 Filtering AK* with Glitch Reject Filter of 10 ns, Cn = 10)	16	89

***The 30 ns delay was used to guarantee that the slaves status is valid before AK* is released on Futurebus+. Notice that AK* is in a different GAL (S__GAL2) then the signal that is clocking the status out to the Futurebus+ (CKO__D__S__C in S__GAL1). A 20 ns delay is needed to offset the skews between the two different GALs and the data transceiver (DS3886) verses handshake transceiver (DS3884). Since no more inputs could fit on S__GAL1 a 30 ns delay had to be used.

5.11 Example calculation of maximum and typical time of a 8 transfer burst transaction between a master and the slave memory board.

Maximum time calculation for 8 transfer burst read transfer across Futurebus+ is 1000 ns.

Parameter (Maximum Delays)	(In nano-seconds) Single Delay	Total Delay
Maximum Time for Read Access AS* to DI* (Section 5.2, this Includes the First Transfer)	359	359
Maximum Time of Next 7 Read Access Transfers (Section 5.6, DS* to DK* or DI* received, 7 x 75 ns)	525	884
Maximum Time of Disconnected Phase, DK* or DI* Received to AK* Released (Section 5.10)	116	1000

13,3 NTS

Typical time calculation for 8 transfer burst read transfer across Futurebus+ is 712 ns.

Parameter (Typical Delays)	(In nano-seconds) Single Delay	Total Delay
Typical Time of Read Access AS* to DI* (Section 5.2, this Includes the First Transfer)	287	287
Typical Time of Next 7 Read Access Transfers (Section 5.6, DS* to DK* or DI* Received, 7 x 48 ns)	336	623
Typical of Disconnect Phase, DK* or DI* Received to AK* Released (Section 5.10)	89	712

6.0 DESIGN CONSIDERATIONS OF THIS APPLICATION NOTE

This Futurebus+ slave memory board was designed and simulated but it has not been built and tested in a real Futurebus+ system. Though this board was designed to go into a Futurebus+ system there are several modes of operation that have not been supported.

6.1 Partial Transactions

The compelled read (or write) partial transaction has not been supported in this application. This could be designed into this application but would further complicate the PAL equations. Also a partial read could perform a double-word read without any problems.

To do a partial write a double-word read could be performed first. The byte (or bytes) to be written could then be written into the double-word in the masters board. A double-word write could then be performed.

6.2 End of Data Status Indication (Out-of-Page)

This board does not have a page comparitor built into it. This would be needed if the master has the possibility of continuing to read or write beyond the end of a page (incrementing beyond a DRAM column address of all ones).

This could be designed by using a loadable counter that could be incremented by one for each DRAM access. If the master read or wrote the last double-word in a page of DRAM the end of data status bit (ED*) would be set to end that transaction.

6.3 Parity Detection

Though this application does not show any parity detection circuitry this could easily be added. This circuitry would probably appear as parity check circuitry only on data being written to the slave memory board. In order to keep the board performance as high as possible the parity check information would lag the data being written to memory by one bus beat. In other words, the parity check information would influence the status bits during the next consecutive bus beat.

6.4 Filtering of Address/Data Synchronization Signals

This board does not use filtered versions of the address or data synchronization signals (AS*, AK*, DS*, DK*, DI*). AI* filtered is usod to latch the input status and capability from Futurebus+.

The slave memory board does filter DK* and DI* in the PALs for glitches up to 8 ns using a 10 ns delay line (± 2 ns). This is necessary during read accesses with intervention or during broadcast or broadcall transactions. If the slave board needs glitch filtering greater then 8 ns (worst case) the filtered versions of DK* and DI* must be used from the handshake transceivers (DS3884). Note that this would slow down all accesses on Futurebus+ and hurt the performance of this board. For example, it would add an additional 11 ns (18-7) worst case for 10 ns glitch filtering, Cn = 10.

6.5 Initial Accesses

The initial accesses may seem slow. There are several reasons behind this issue. Once the slave memory board knows that it is chip-selected it starts a memory access. But it must synchronize the first memory access to the on-board clock of 25 MHz (two flip-flops, 80 ns maximum). Also during read accesses the slave memory board cannot start the access until it is known whether intervention may occur or not. This is not known until DS* is asserted. If not for this requirement the memory board could start the initial read access earlier thereby perhaps shortening the initial read access.

The synchronization of the initial access could be accomplished faster using a synchronizing D-Type Flip-Flop with metastable immune characteristics, such as the 74F5074. With only one Flip-Flop synchronizing stage, utilizing the 74F5074, the initial access would gain approximately one clock period (see *Figure 34*).

6.6 CSR Space

This was not dealt with in this application note.

6.7 Module Registers

Even though all the Module Registers were not simulated it is assumed that they existed in this application note. This includes the Module Capability Registers, Module Control Registers, and the Module Status Registers.

6.8 Status and Capability Bits

In *Figure 3* the selected bit is shown being output to the Futurebus+ from the DS3884 Handshake transceiver. Actually this application note assumes that all status and capability bits are clocked out to the Futurebus+ through a DS3886 Latched Data transceiver by the signal CKO_D_S_C. Though this transceiver was not shown in the simulation block diagrams of *Figures 3-7*, the assumed connection diagram can be seen in *Figure 35*.

7.0 PROTOCOL CONTROLLER PAL/GAL INPUT AND OUTPUT DEFINITIONS

In this section inputs and outputs of the PALs/GALs are designated as **active low by a trailing "~"** (ex. ADD_ ACK~). An active low signal is valid when it is low, or logic 0. An **active high** signal is designated by not ending with a trailing "~" (ex. WR). An active high signal is valid when it is high, or logic 1.

7.1 S_GAL1 Inputs

CS_AS~	Indicates that a Chip-Selected Address Synchronization access request has been received from Futurebus+ when asserted.

WR	A latched and inverted version of the WRite command (SR*) from Futurebus+. This command indicates that a write access is in progress when asserted and is held latched until AS* is released.
IV	A inverted version of the InterVention (IV*) status bit from Futurebus+. When this bit is asserted during a read access it indicates that another module is intervening in the current access and providing the data. The slave memory module will perform a write access, in order to update the memory with the correct data, instead of performing the requested read access. See Section 4 for more information.
ADD_ACK~	ADDress Acknowledge from the memory module. This input (when asserted) indicates that the DRAM is ready to be read or written.
ADD_ADD_10~	A delayed version of ADD_ACK~ by 10 ns.
D_ACK	The Futurebus+ protocol controller ACKnowledge. This is a signal from the memory module indicating that the current read or write access request (DT_STB) from the protocol controller has been accomplished. This is an edge sensitive signal. In other words, both the rising and falling edges indicate a response from the memory module. DT_STB and D_ACK form a request/acknowledge type of handshake.
D_ACK_10	A 10 ns delayed version of D_ACK.
DK_IN	A received inverted version of the Futurebus+ Data acKnowledge (DK*). This is needed during accesses with intervention to tell the protocol controller when it is allowed to drive its DK* signal. See Section 4 for more information.
DKI_10	The Futurebus+ DK* input inverted and delayed by 10 ns.
DI_IN	A received inverted version of the Futurebus+ Data acknowledge Inverse (DI*). This is needed during accesses with intervention to tell the protocol controller when it is allowed to drive its DI* signal. See Section 4 for more information.
DII_10	The Futurebus+ DI* input inverted and delayed by 10 ns.
AS_30	The Futurebus+ AS* input inverted and delayed by 30 ns.
AS_40	The Futurebus+ AS* input inverted and delayed by 40 ns.
F_INITACC~	A signal indicating the initial Futurebus+ access when asserted. This input is deasserted during the first access.
DS	The Futurebus+ DS* input inverted.
DS_10	The Futurebus+ DS* input inverted and delayed by 10 ns.

7.2 S_GAL1 Outputs

AI	Address acknowledge Inverse (inverted) output to the Futurebus+ AI* signal.
DT_STB	The access request (read or write) signal from the protocol controller to the memory module. This is an edge sensitive signal. In other words, both the rising and falling edges indicate a DRAM request to the memory module. DT_STB and D_ACK form a request/acknowledge type of handshake.
LEI_A_D	The Latch Enable for Input Address and Data from Futurebus+. This is an input to the DS3886 latched data transceivers and allows the address or data from the Futurebus+ to be enabled (when asserted) or latched (deasserted).
CKO_D_S_C	The ClocK for Data, Status, and Capability bits. This is an input signal to the DS3886 latched data transceivers and clocks data, status and capability bits out to Futurebus+ on the rising edge.
DI	Data acknowledge Inverse (inverted) output to the Futurebus+ DI* signal.
DK	Data acKnowledge (inverted) output to the Futurebus+ DK* signal.

7.3 S_GAL2 Inputs

CLK	The 25 MHz on-board system clock for the DP8422A-25 DRAM controller.
AS	The inverted AS* from Futurebus+.
CS~	Indicates that a Chip-Selected Futurebus+ address is asserted.
WR	A latched and inverted version of the WRite command (WR*) from Futurebus+. This command indicates that a write access is in progress when asserted and in held latched until AS* is released.
IV	An inverted version of the InterVention (IV*) status bit from Futurebus+. When this bit is asserted during a read access it indicates that another module is intervening in the current access and providing the data. The slave memory module will perform a write access, in order to update the memory with the correct data, instead of performing the requested read access. See Section 4 for more information.
ADD_ACK~	ADDress Acknowledge from the memory module. This input (when asserted) indicates that the DRAM is ready to be read or written.

AI	Address acknowledge Inverse (inverted) output to the Futurebus+ AI* signal.
AI_IN	A received inverted version of the Futurebus+ Address acknowledge Inverse (AI*) filtered. This is needed during accesses with intervention, broadcast or broadcall to tell the protocol controller when it is allowed to latch the status and capability bits input from Futurebus+.
AS_20	The Futurebus+ AS* input inverted and delayed by 20 ns.
AS_30	The Futurebus+ AS* input inverted and delayed by 30 ns.
DS	The Futurebus+ DS* input inverted.
DK	Data acKnowledge (inverted) output to the Futurebus+ DK* signal from the S_GAL1.
ACIP~	A feedback signal from the Memory Module Controller indicating that an ACcess is still in Progress S_GAL2 will not end the current memory access (AREQ~) or allow AK* to be released until ACIP~ is deasserted.

7.4 S_GAL2 Outputs

CS_AS~	Indicates that a Chip-Selected Address Synchronization access request has been received from Futurebus+ when asserted.
AREQA~	Access REQuest input used to produce AREQ~. This output when asserted implies that a chip-selected request from Futurebus+ has been received and clocked by the local clock.
AREQ~	The Access REQuest input to the memory module. This output when asserted implies that a chip-selected request from Futurebus+ has been received and clocked through two rising edges of the local 25 MHz clock. This synchronizes the request from Futurebus+ to the local clock. This output is used as an input to the memory module to request a DRAM access.
F_INITACC~	A signal indicating the initial Futurebus+ access when asserted. This input is deasserted during the first access.
AK	Address acKnowledge (inverted) output to the Futurebus+ AK* signal.
LEI_ADD_CM	The Latch Enable for Input Address and Command bits from Futurebus+. This is an input to the 74F373 latches and allows the address or command from the Futurebus+ to be enabled (when asserted) or latched (deasserted). Note that this latch follows the Futurebus+ DS3886 latches and holds the address and command latched until the complete transaction is terminated (AS* released).
WRITE~	The direction input to the DS3886 Futurebus+ latched data transceivers. When asserted address or data is received into the slave memory board. When deasserted data is transmitted to the Futurebus+ backplane.
LEI_S_C	The Latch Enable for Input Status (and Capability bits if desired) from Futurebus+. This is an input to the 74F373 latches and allows the status and capability bits from the Futurebus+ to be enabled (when asserted) or latched (deasserted). Note that this latch follows the Futurebus+ DS3886 latches and holds the status and capability latched until the complete transaction is terminated (AS* released). This output is very similar to LEI_ADD_CM but does not latch until the filtered version of AI* is received.

PART 2. DESCRIPTION OF MEMORY MODULE DESIGN

8.0 GENERAL DESCRIPTION

This application note describes a Futurebus+ like Memory Module that can support burst transfer rates of 20 megatransfers/second. This Memory Module utilizes two DP8422A-25 DRAM controllers (8420_MODULE0 and 1), a GAL interface module consisting of 5 PALs, (MEM_MODULE_CTR), and latched transceivers (i.e., 74F543's).

This design could form the basis of a Futurebus+ slave-only DRAM memory board *(Figure 1, 2, 22),* or it could be used in an I/O board design. An I/O board can act as both a master and a slave with respect to the Futurebus+. *Figure 36* shows one possible I/O board design containing DRAM memory, a CPU, and an I/O controller. In *Figure 36* the I/O controller is the National Semiconductor DP83932 Systems-Oriented Network Interface Controller (SONIC).

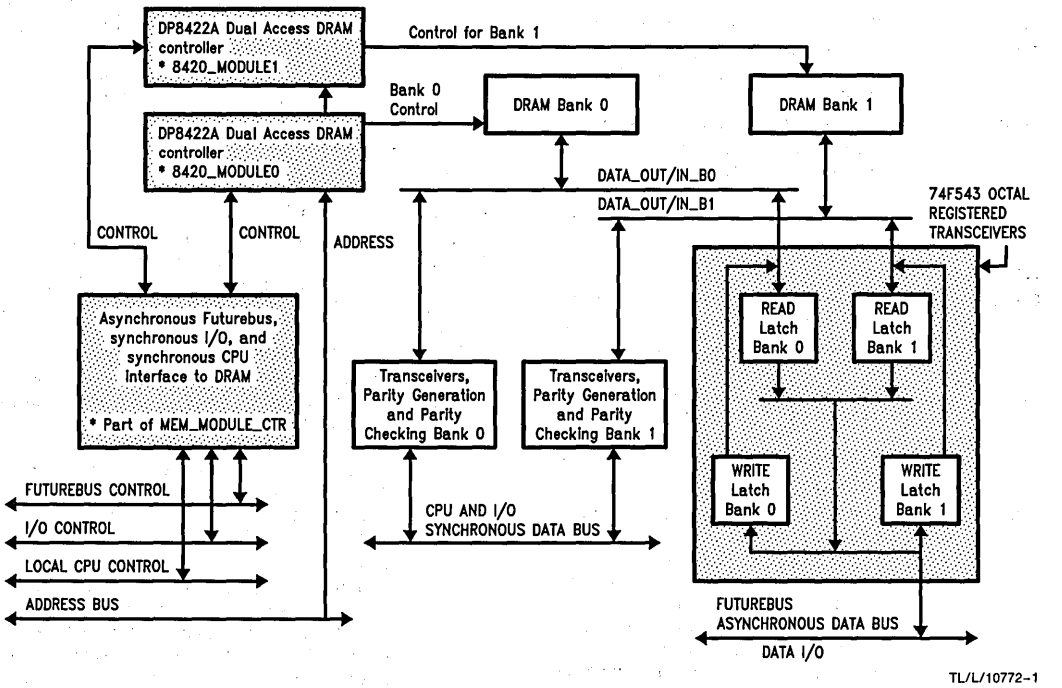
All DRAM access requests are assumed to be handled by an on-board arbiter in the case where the board contains processors along with the Futurebus+ interface and DRAM memory. The dual access control logic of the DP8422A-25 is not used in this design. This is because an external arbiter is needed to control the local busses, and this same logic can control accesses to the DRAM as well. Many processors have built-in support for local bus arbitration, such as the 68030, which includes three signals to perform this task (i.e., Bus Request, Bus Grant, and Bus Grant ACKnowledge). The protocol controller could have a similar type of arbitration structure. When one of the devices is granted control of the local bus, the other devices TRI-STATE® their address, data, and control busses. The control signals necessary to interface this memory design to the Futurebus+ protocol controller and on-board CPU are described in Section 14.

Figure 36 shows a block diagram of a portion of a Futurebus+ I/O card that includes DRAM, the SONIC, and a local CPU. The SONIC, CPU, and the Futurebus+ protocol controller share the local data bus, address bus, and control

bus. The local data bus is buffered from the DRAM and separately buffered from the asynchronous Futurebus+ data bus. This was done to simplify the control logic, though it should be possible to use only one set of buffers for both the Futurebus+ and the local data bus.

This memory design supports intervention. Intervention allows the system memory to operate in a multiprocessing cache-based environment. In intervention a read access from DRAM may be changed to a write access if another Futurebus+ cache boards owns (has modified) that particular piece of data but not yet updated the system memory. During intervention the true data is passed to the system from the owner, and the system DRAM is updated with the correct value. In other words, a DRAM read transaction may become a write transaction after the address has been issued. To allow this feature the DRAM design has the restriction that the initial DT_STB' must not be generated until it is known whether the current transaction involves intervention. In other words, the 'READ' input to the DRAM interface must be valid before the intial 'DT_STB' edge is given to the DRAM interface.

Another restriction in the DRAM interface is that the access request (AREQ~) input should be synchronized to the input clock 'CLK'. Also the initial 'DT_STB' should not be generated until the DRAM interface outputs address acknowledge, 'ADD_ACK'.



**Shaded areas indicate the logic that was simulated.

FIGURE 22. Memory Module Block Diagram

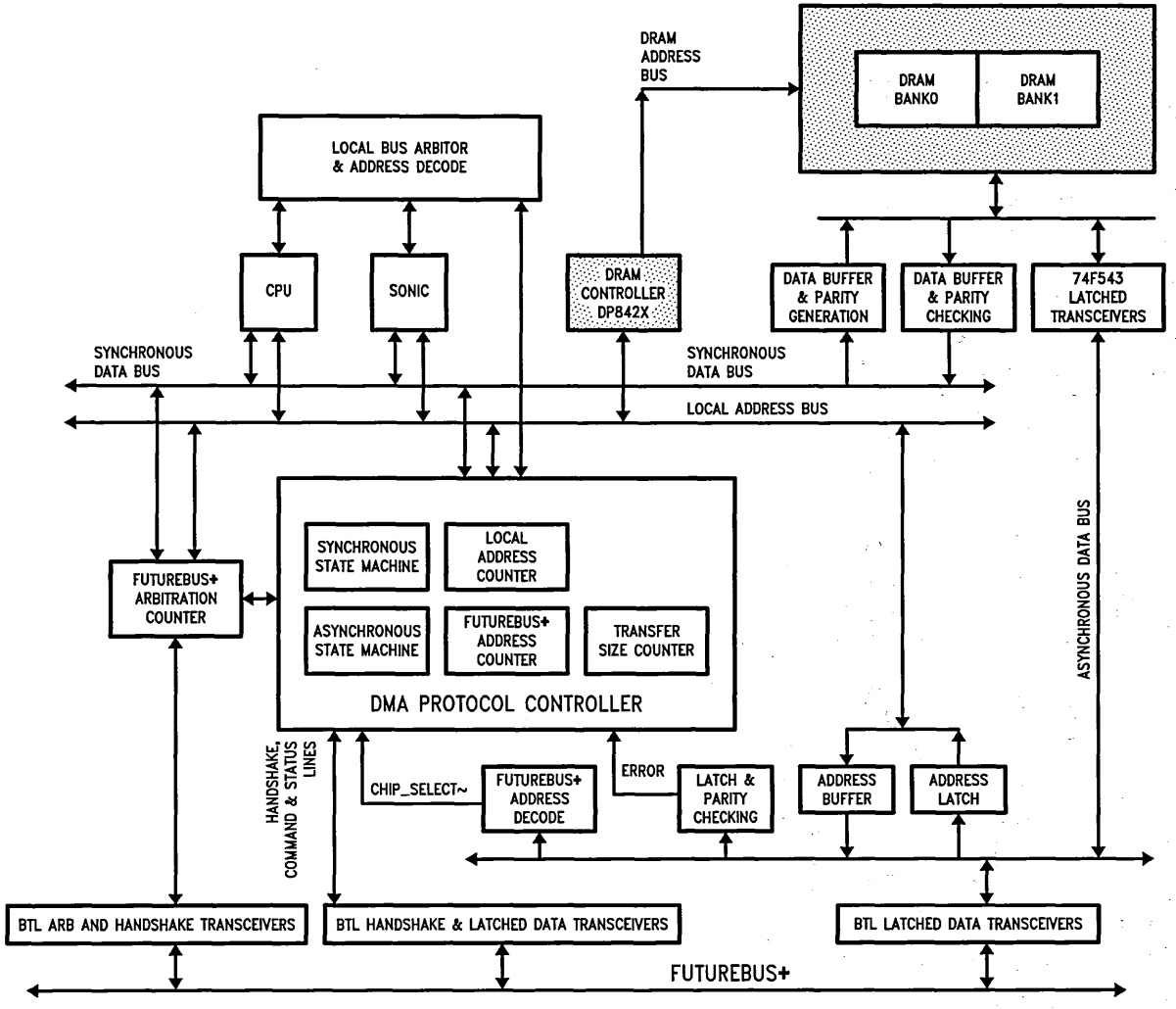


FIGURE 36. Futurebus+ I/O Board Block Diagram

9.0 DRAM CONTROLLER DESCRIPTION (8420_MODULE0, 1)

Two DP8422A-25 DRAM controllers were utilized in this design. This allows one DRAM controller per bank of DRAM. The Memory Module was designed this way for the following reasons:

1. The control logic could be simplified since the two memory banks can be controlled completely independently;
2. Greater performance because of interleaved memory banks;
3. Greater performance because of less capacitance on the output drivers; and,
4. Increased drive capability (up to two banks of 64 bits per bank) without a loss in performance.

This memory design takes care of:

1. All DRAM timing;
2. Arbitration between refresh and access cycles;
3. The automatic insertion of wait states to the accessing device when needed (i.e., RAS~ precharge, refreshing);
4. Byte reads and writes to 32-bit (or 64-bit) memories;
5. Normal and burst operations, and;
6. Incrementing the column address of each bank during burst accessing.

Only port A of the DRAM controllers is used in this design, port B is disabled by tying 'AREQB~' high. In other words, the dual access arbitration circuitry of the DP8422A-25's was not used in this design. All arbitration between different devices trying to access the DRAM, as well as DRAM refresh, is controlled external to the DP8422A-25's.

10.0 GAL INTERFACE MODULE DESCRIPTION (MEM_MODULE_CTR)

There are 5 PALs (GALs were used in the computer simulation) used in this design to interface between the DRAM controllers, the 74F543 transceivers, and the Futurebus+ protocol controller. Section 14 describes the inputs and outputs of the GAL interface, while Section 15 shows the GAL equations that were programmed into the GALs.

Figures 22, 23, 24, 25, 26 are shown to give a graphical representation of the GAL design logic of the Memory Module through block diagrams and state transition and decoder diagrams.

Figures 27 through *33* show actual simulation timing diagrams of the Memory Module design through various read, write and DRAM refresh operations.

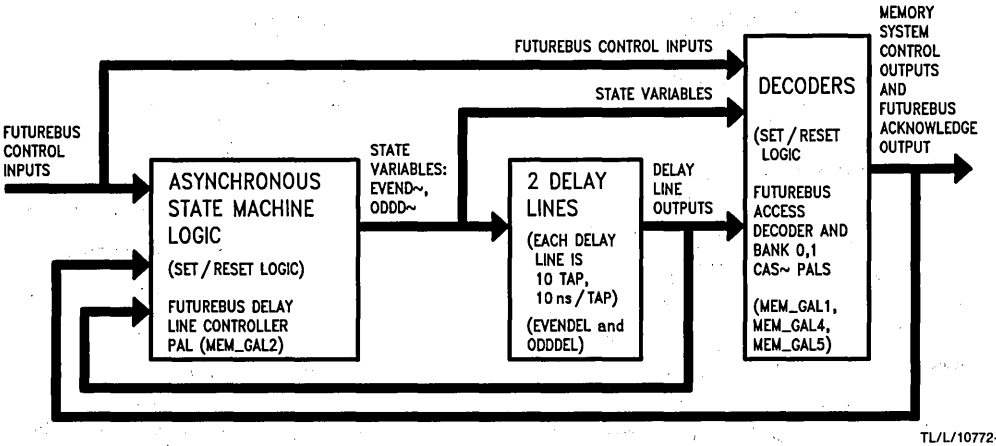


FIGURE 23. Futurebus+ Memory System Asynchronous Controller Block Diagram (MEM_MODULE_CTR)

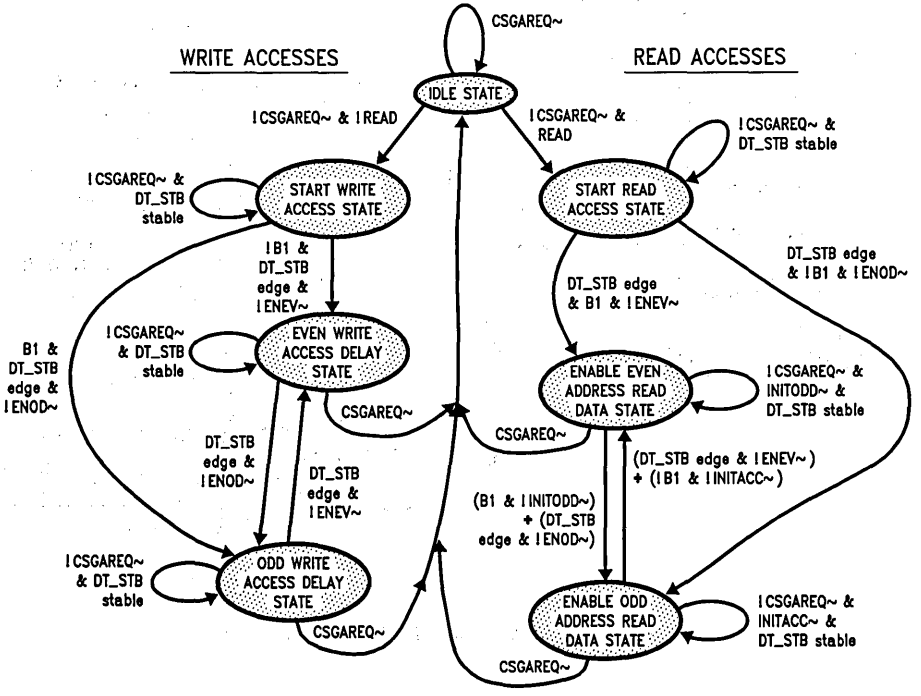
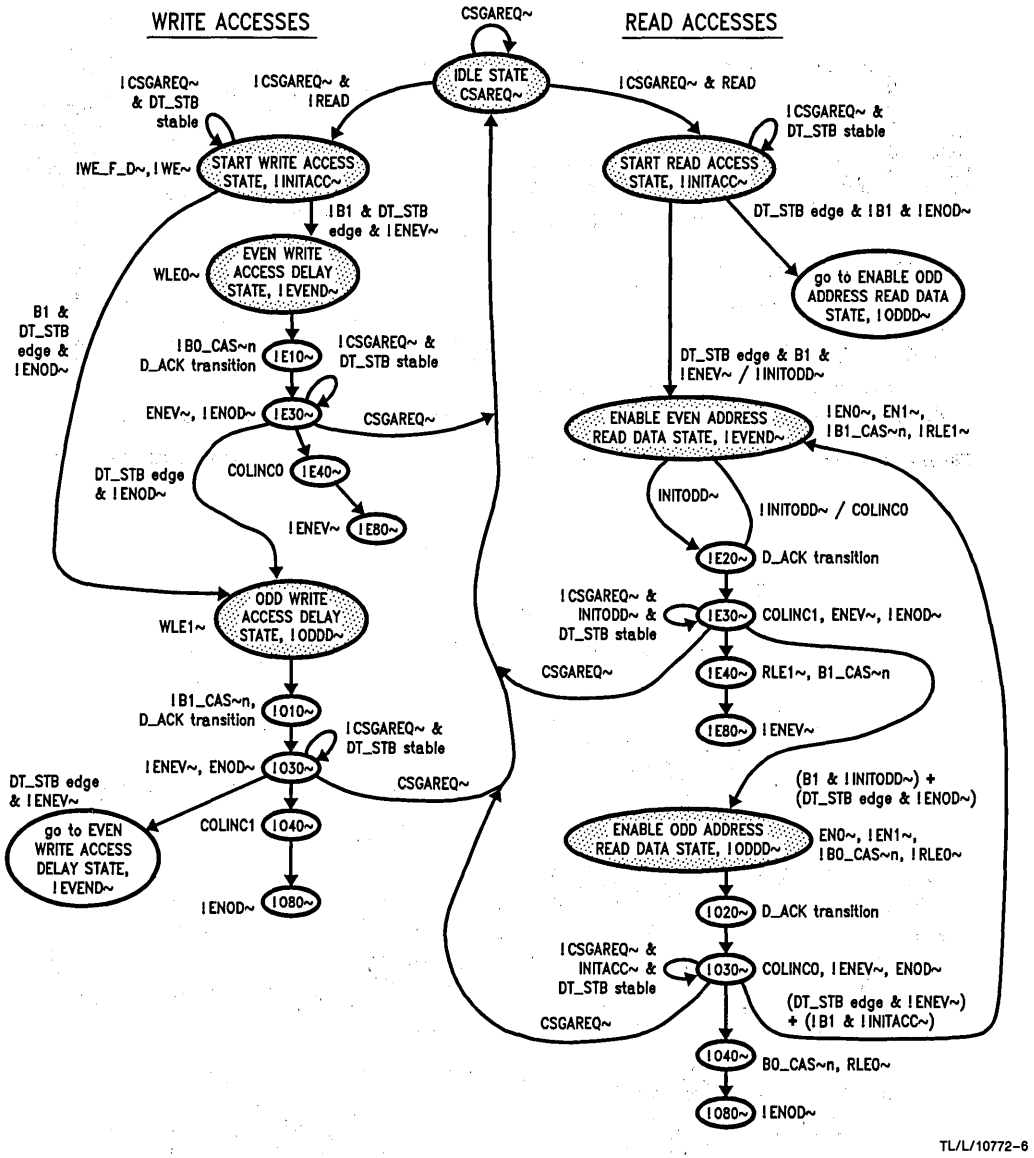
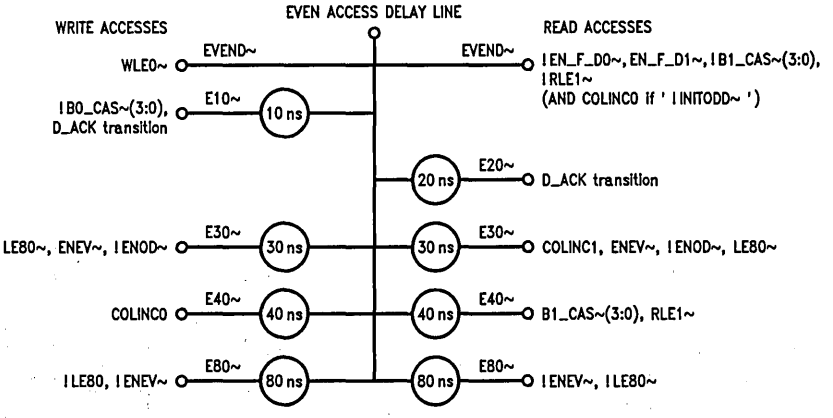


FIGURE 24. Futurebus+ Memory System State Transition Diagram



TL/L/10772-6

FIGURE 25. Delay Line/Decoder Transition Diagram



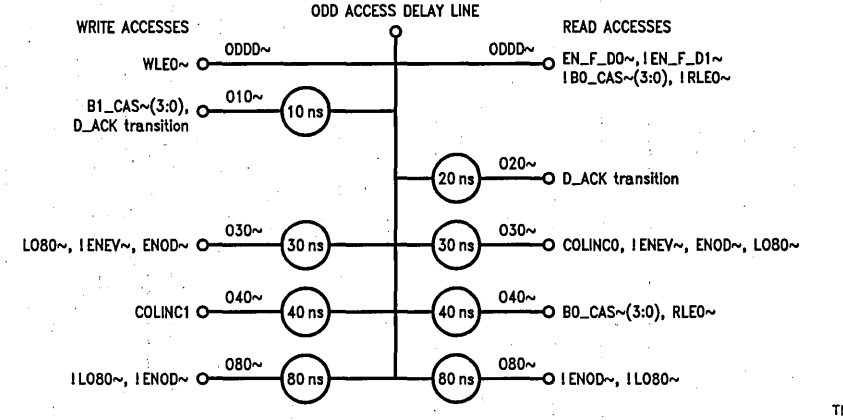
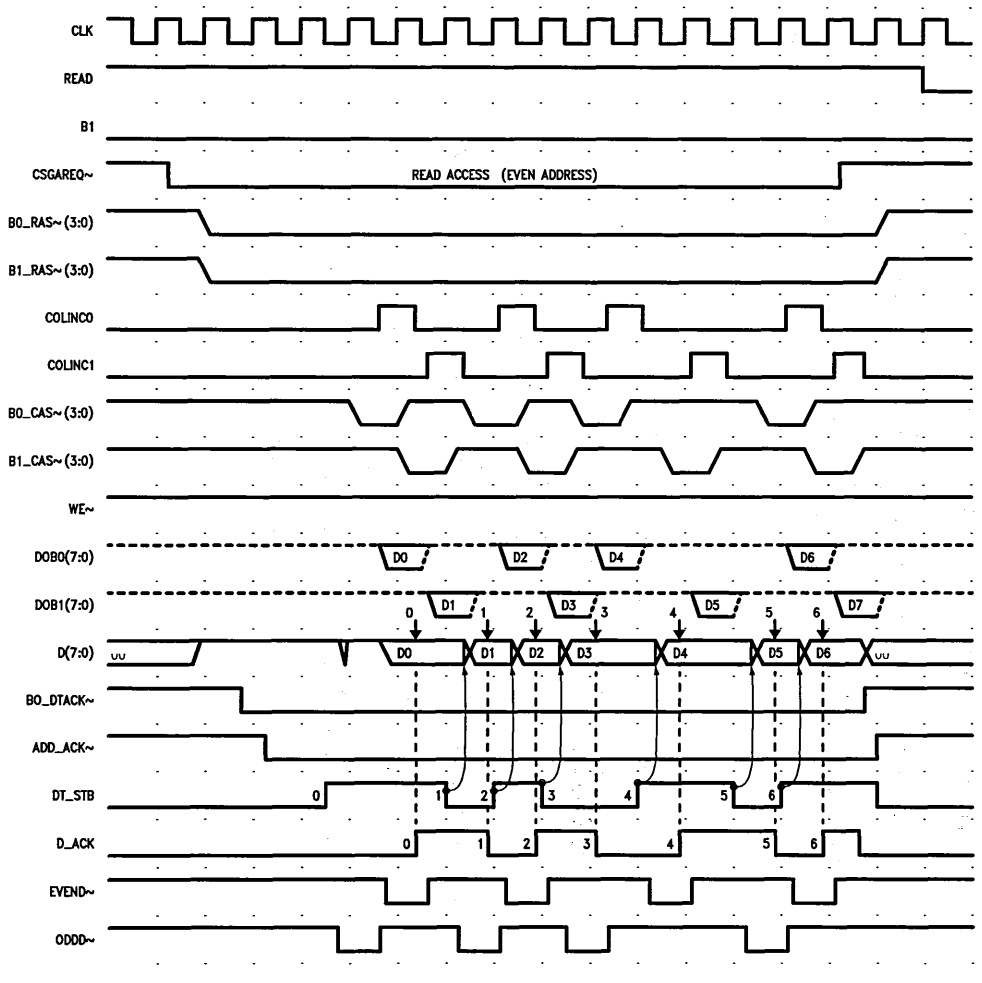
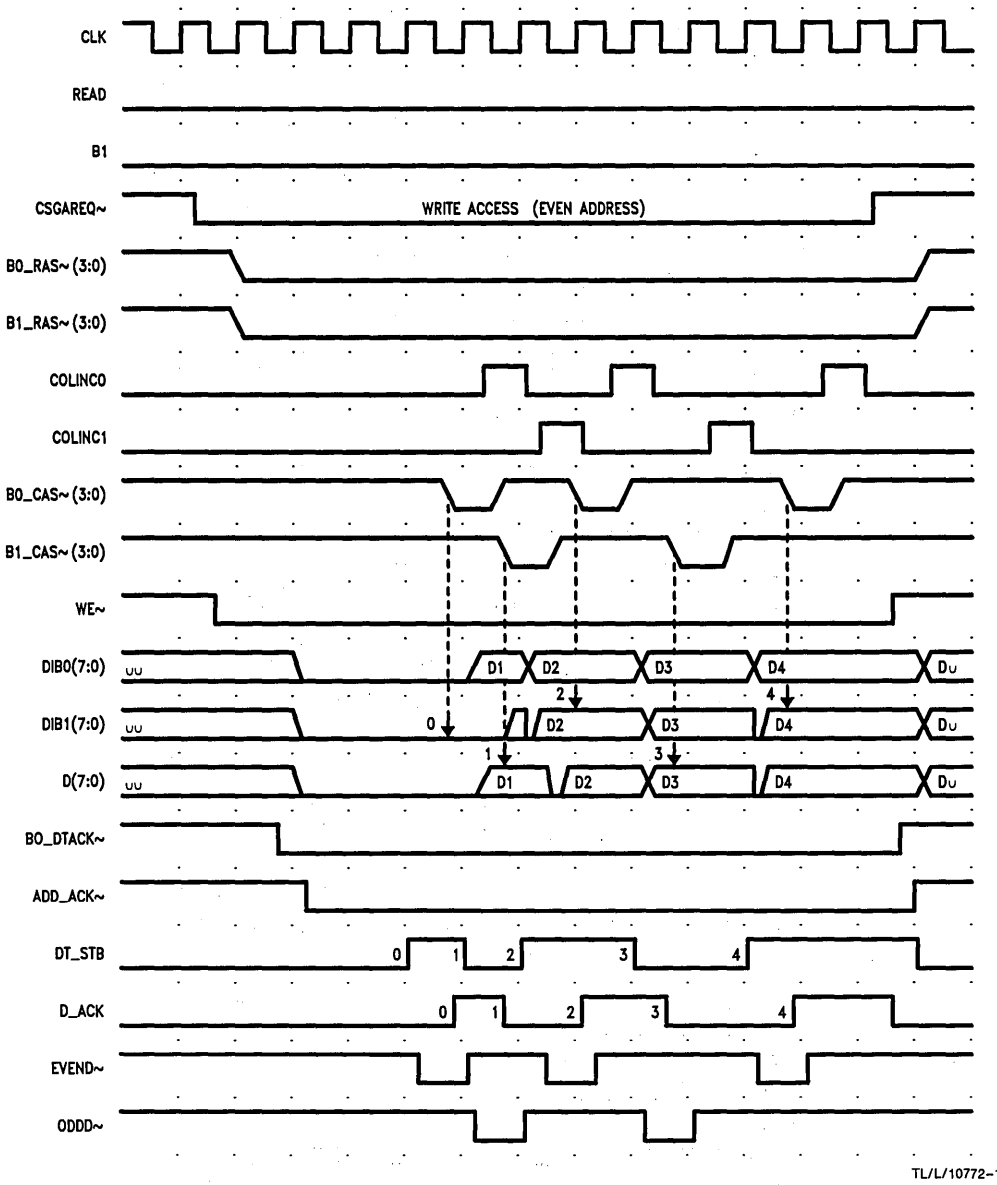


FIGURE 26. Delay Line/Decoder Diagram



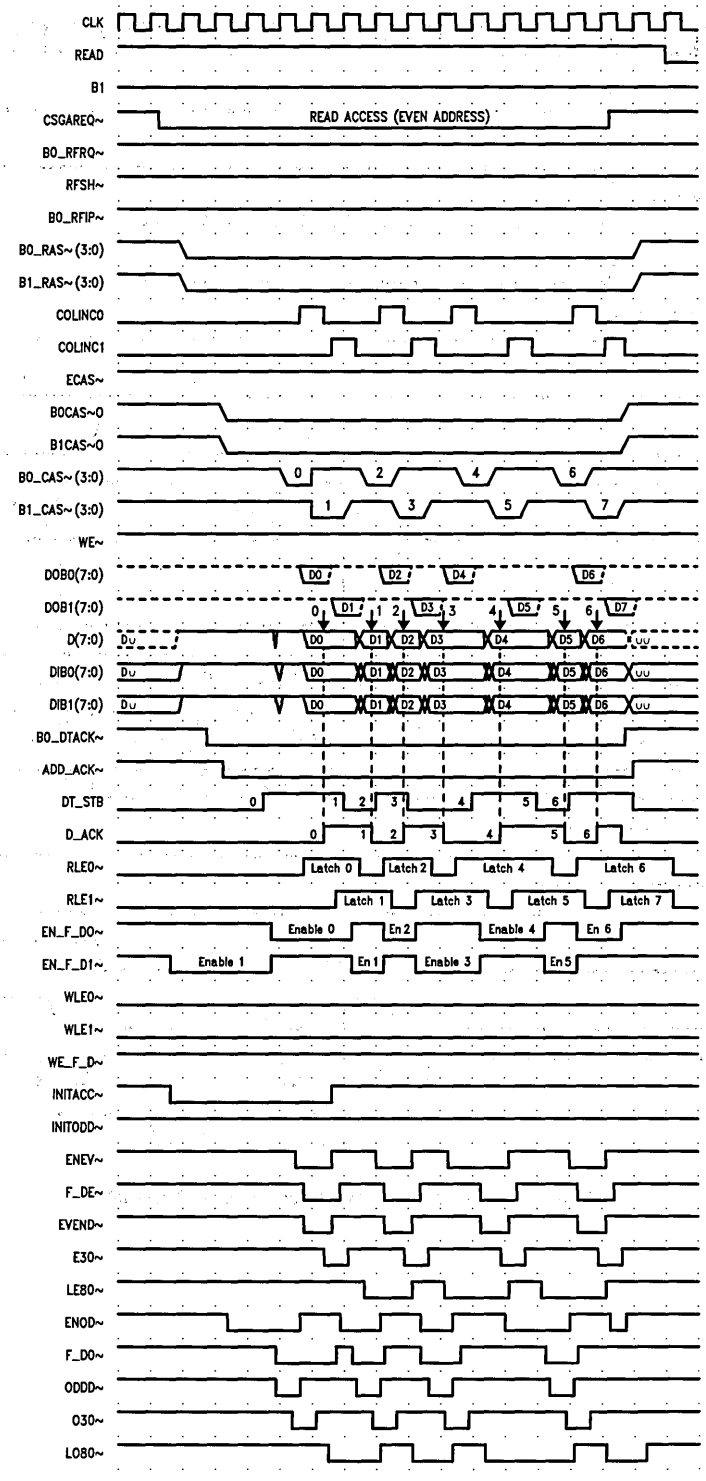
TL/L/10772–11

FIGURE 27. Memory Module Read Access (Even Address)



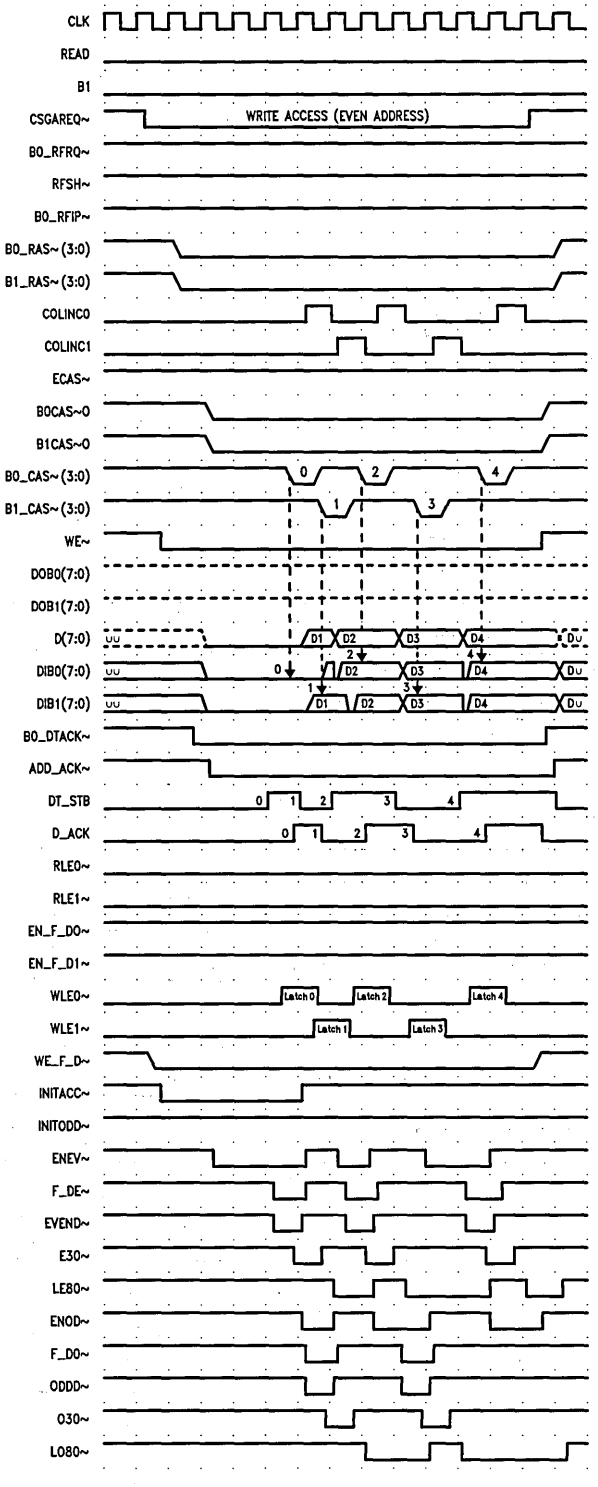
TL/L/10772–12

FIGURE 28. Memory Module Write Access (Even Address)



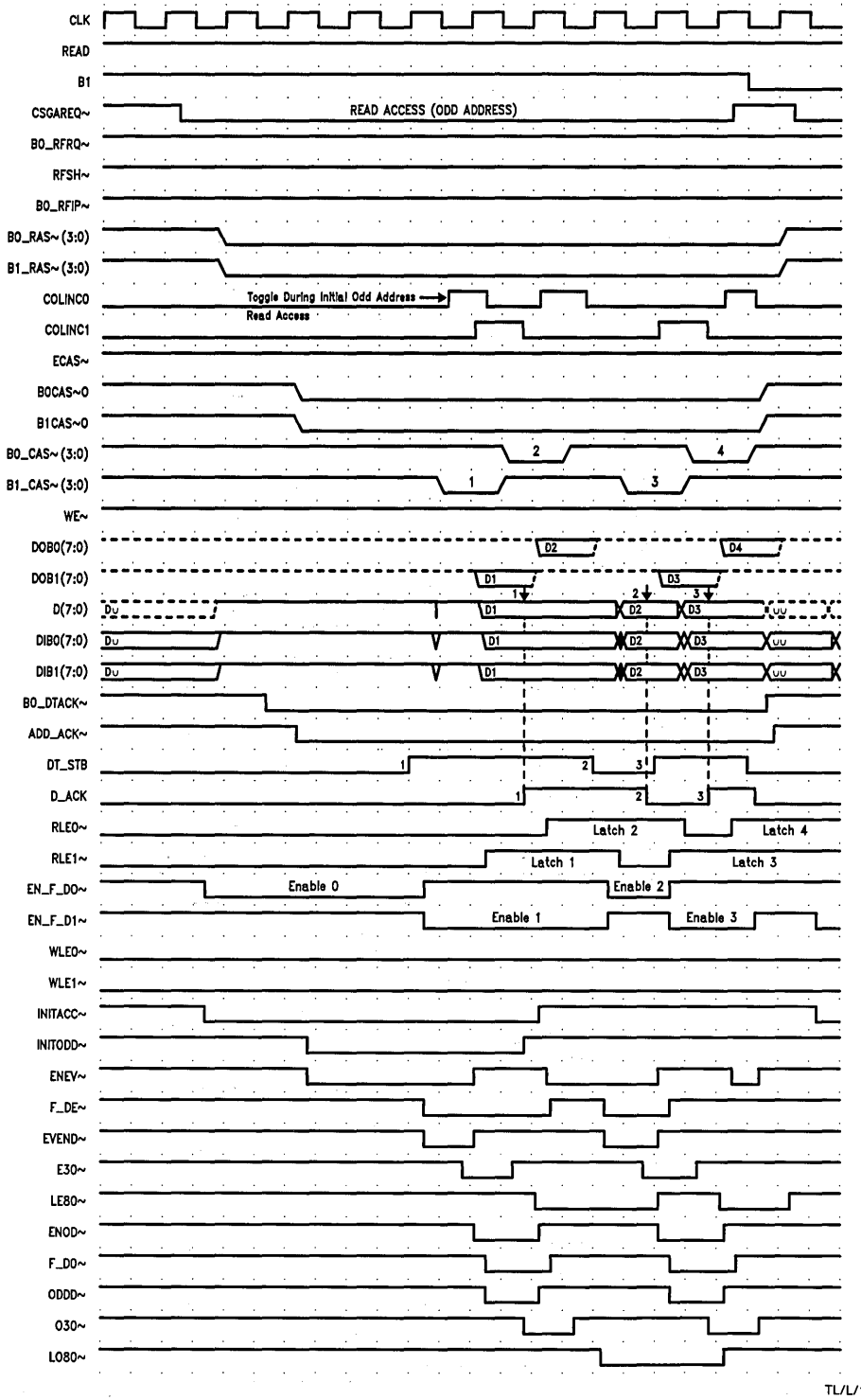
TL/L/10772–13

FIGURE 29. Memory Module Read Access (Even Address)



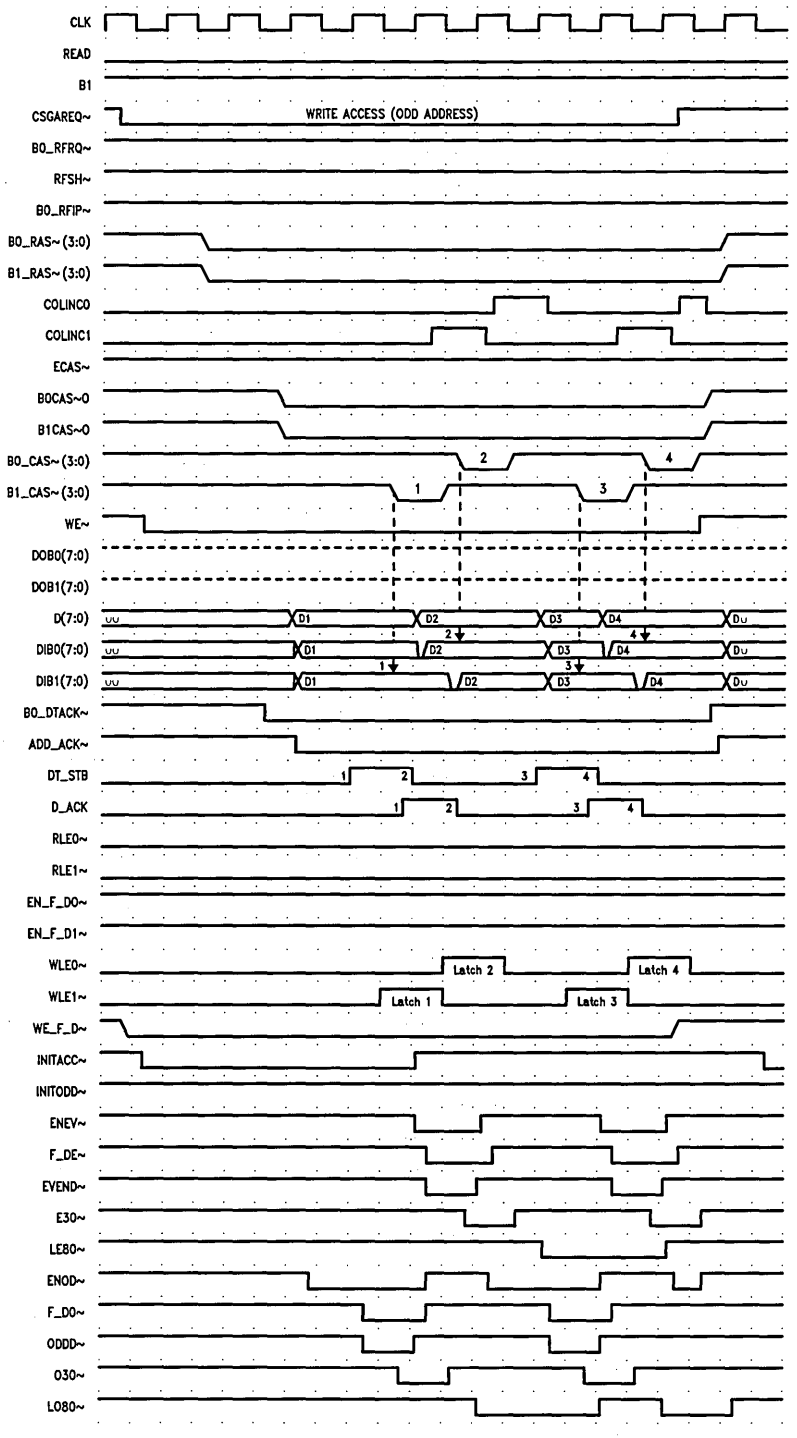
TL/L/10772–14

FIGURE 30. Memory Module Write Access (Even Address)



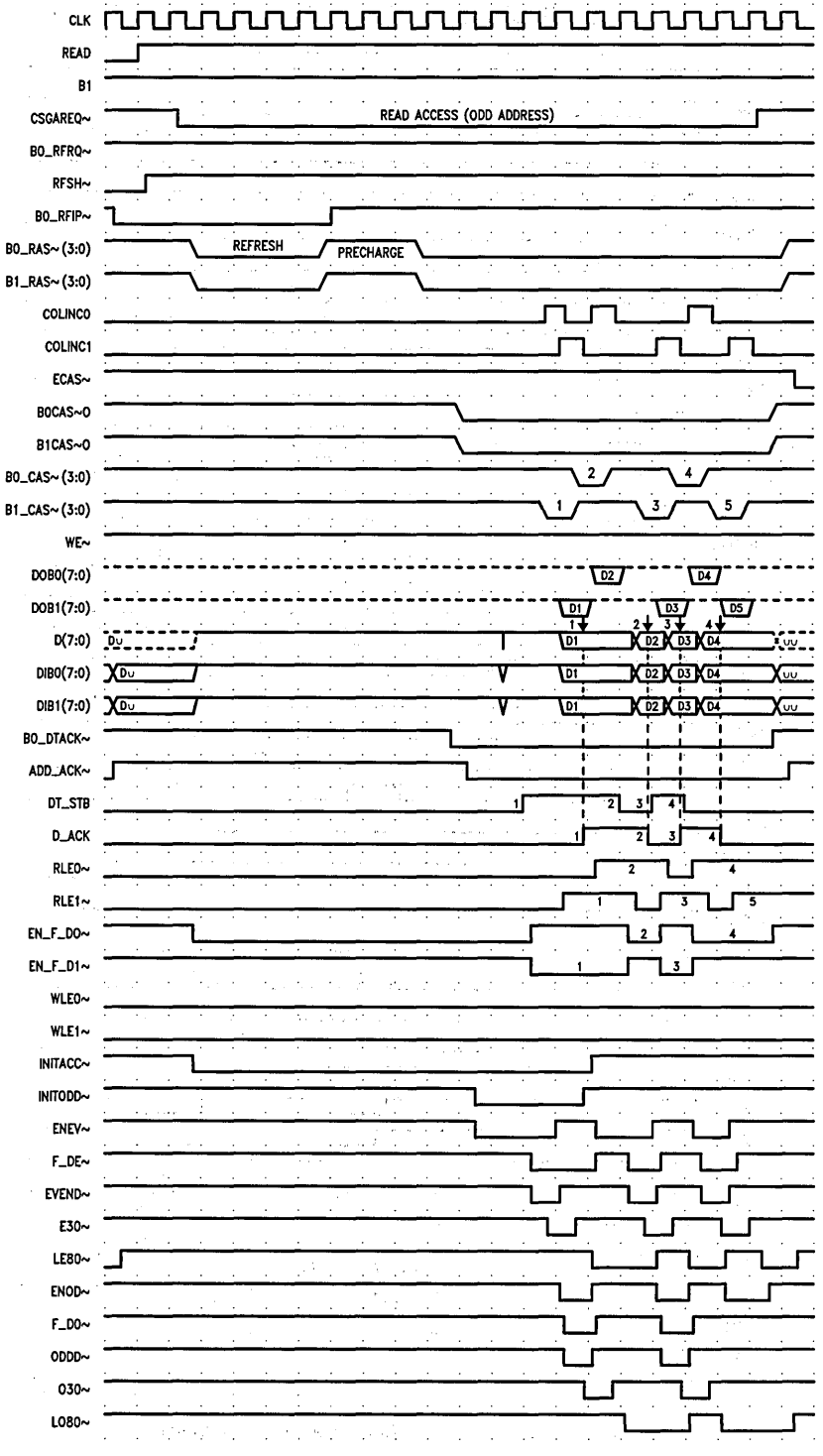
TL/L/10772–15

FIGURE 31. Memory Module Read Access (Odd Address)



TL/L/10772–16

FIGURE 32. Memory Module Write Access (Odd Address)



TL/L/10772–17

FIGURE 33. Memory Module Refresh then Read Access (Odd Address)

A potential DRAM access is initiated by CS~ and AREQ~ (Chip Select and Access REQuest) being driven low. Once the RAS~s are low and the DRAM column address is valid at the DRAM inputs the GAL Interface Module drives the ADDress phase ACKnowledge (ADD_ACK~) low to the Memory Module controller. Seeing ADD_ACK~ the Memory Module Controller can now request a DRAM read or write access through the Futurebus+ DaTa STroBe (DT_STB edge). During the initial access of a read or write operation the first DT_STB edge will always be a rising edge.

This Memory Module design assumes two banks of memory. These two banks of Memory (DRAM) are an even address bank (bank 0) and an odd address bank (bank 1) controlled by the least significant double word address bit (B1). The Memory Module will determine whether the access is an even address (bank 0) access or an odd address (bank 1) access by looking at the lower address bank select input (B1).

Each DRAM access is initiated through an DT_STB edge from the Memory Module Controller. When the GAL Interface Module sees this edge an access to the even address bank 0 or the odd address bank 1 takes place. The MEM_GAL2 produces the state variables (EVEND~ and ODDD~) from the DT_STB edge that cause each DRAM access to take place. These state variables are inputs to delay lines (EVENDEL, ODDDEL, see *Figure 7*) that produce delayed versions of the initial state variable input. The delayed versions of these inputs (E10-80~ and O10-80~) are used as inputs to the other four GALs in this Interface to produce the outputs that control the entire operation of the Memory Module Interface (see *Figures 3, 6,* and *7*).

The first access request (DT_STB edge) of any read operation causes two accesses (see *Figure 27*). The first access retrieves and latches the data needed for the present access. The second access outputs the needed data onto the Futurebus+ asynchronous data bus, drives the data acknowledge output (D_ACK) for the current access, and performs a DRAM access to get the next sequential piece of data for the next access request (DT_STB edge). In other words, read operations are pipelined. The data needed for the current access (DT_STB edge) was accessed and latched into the output transceivers during the previous access.

Each subsequent read access, during a burst read operation, performs only one memory access operation. As an example see *Figure 29* and refer to the DT_STB edge for the data 'D4'. This rising edge of DT_STB has a '4' to the left of the rising edge. In this timing diagram the current access request (DT_STB edge) is a read access from an even address (bank 0). The currently needed even address data was accessed and latched in the preceeding odd address read access (see the 'B0CAS~' signal labeled '4' and the 'RLE0~' signal labeled 'Latch 2'). An enable signal is generated during the current access to drive the data, D4, onto the Futurebus+ asynchronous data bus (see the 'EN_F_D0~' signal labeled 'Enable 4' on the timing diagram). The data for the next sequential transfer request (from bank 1) is accessed and latched during the current access (see 'B1CAS' signal labeled '5' and the 'RLE1' signal labeled 'Latch 5'). Therefore, the present read access outputs even address bank 0 data and performs a DRAM access and latching of the odd address bank 1 data in preparation for a subsequent odd address read access.

During a Memory Module read access, each piece of data accessed from the DRAM is guaranteed a setup time on the Futurebus+ asynchronous data I/O bus (see *Figure 22*) with respect to the Futurebus+ data ACKnowledge (D_ACK), and is held valid until the next access is requested (DT_STB edge), or the access is terminated (CSGAREQ~ high).

During write accesses, the data to be written to the Memory Module is latched into the write transceiver latches, and then an D_ACK signal is output to allow the next piece of write data to be enabled onto the Futurebus+ asynchronous data I/O bus.

In controlling the DP8422A-25's, the GAL Interface Module must increment the column address of the DRAMs after each access via the COLumn INCrement (COLINC0 and 1) input, one for each DRAM bank.

This interface guarantees a column address setup and hold time for the present access (see Section 4.7 and 4.9) and then increments the column address to prepare for the next access through the DP8422A-25 COLINC inputs.

During an initial odd address read access, COLINC0 is pulsed before the first even address bank 0 access is performed to guarantee that the even access is to the next sequential double word of data (see *Figures 25* and *31*).

It should be noted that this Memory Module does not include logic to sense when the DRAM accesses cross a DRAM page boundary. It is assumed that the master that is reading or writing the Memory Module contains the required logic to stop accessing the DRAM before crossing a DRAM page boundary.

The MEM_GAL1 controls the enable signals to the read access transceivers (EN_F_D0~, EN_F_D1~), the latch enable signals during write transactions (WLE0~, 1~), the write enable signal to the DRAMs (WE~), and the Futurebus+ data ACKnowledge (D_ACK) output to the Memory Module controller.

The MEM_GAL3 controls the random logic needed in the Memory Module interface, including; ADDress ACKnowledge (ADD_ACK~) to the Memory Module controller, the ReFreSH input to the DRAM controllers (RFSH~), the Chip Selected and Granted Access REQuest (CSGAREQ~) signal, and the latched versions of the E80~ and O80~ outputs of the delay lines (LE80~ and LO80~).

MEM_GAL4 and MEM_GAL5 control the byte CAS~s for each bank (CAS~0-3), the Read access Latch Enable for the individual bank (RLE0~ or RLE1~), and the COLumn INCrement signal for the individual bank (COLINC0 or COLINC1).

11.0 LATCHED TRANSCEIVER DESCRIPTION (MEM_MODULE_CTR)

The 74F543 latched transceivers are needed to gain the necessary performance by performing pipelining of the data. For example, in an even bank 0 read access, the even piece of data (from the bank 0 transceiver latch) is enabled onto the Futurebus+ asynchronous data bus while the next sequential piece of data from the odd bank 1 is accessed and latched into the bank 1 transceiver latch. During a write access, an acknowledge (D_ACK) can be generated after the data is latched into the write transceiver latch. The actual writing of the data into the DRAM occurs simultaneously with D_ACK.

12.0 FUTUREBUS+ MEMORY MODULE DP8422A PROGRAMMING BITS

Programming Bits	Description
ECAS0~ = 1	Extend CAS~ and Get Refresh Request (RFRQ~) Output Instead of WE~
B1 = 1	Access Mode 1
B0 = 0	ADS~ Low Latches the Address
C9 = 1	Delay CAS~ during Write Accesses by One Clock
C8 = 1	Row Address Hold Time of 15 ns Minimum
C7 = 1	Column Address Setup Time of 0 ns Minimum
C6 = 0	RAS~ 0-3 and CAS~ 0-3 are all Selected during an Access
C5 = 1	B0 and 1 are Not Used during an Access
C4 = 1	
C3 = 0	15 μs Refresh Period
C2 = 0	Assuming a 20 MHz DELCLK Input, a Delay
C1 = 0	Line Clock Divisor of 10 was Selected
C0 = 0	"
R9 = 0	RAS~0-3 All Low Simultaneously during Refreshing
R8 = 1	Non-Address Pipelined Mode
R7 = 1	Data Transfer ACKnowledge (DTACK~) Output Selected
R6 = 0	WAITIN~ Low Adds One Clock Delay to DTACK~
R5 = 0	DTACK~ Will Remain Low during Burst
R4 = 0	Accesses
R3 = 0	DTACK~ Will be Asserted on the First Rising Clock Edge
R2 = 1	After the Access RAS~ Transitions Low
R1 = 1	RAS~ Low during Refresh for 4 Clock Periods and 3 Clock Periods of RAS~ Precharge between Any Two Access to the Same Bank
R1 = 1	

13.0 TIMING CALCULATIONS

1. The Minimum time from Access REQuest (AREQ~) to the initial DaTa STroBe (DT_STB) is:

   100 ns (ADD_ACK~ gets generated on the second clock after AREQ~ transitions low)

   −10 ns (assume 10 ns maximum delay of Clock to AREQ~ generated)

   +10 ns (GAL16V8-10 max delay of ADD_ACK~ of MEM_GAL3)

   = 100 ns, Therefore from AREQ~ to the initial DT_STB is 100 ns minimum since DT_STB must not transition until ADD_ACK~ is generated.

2. Initial Read access even (or odd) address DT_STB to D_ACK:

   10 ns (GAL20V8-10 max delay of DT_STB to ODDD~ of MEM_GAL2)

   +32 ns (max delay time of 30 ns tap of delay line)

   +10 ns (GAL20V8-10 max delay of ENEV~ of MEM_GAL2)

   +10 ns (GAL20V8-10 max delay of EVEND~ of MEM_GAL2)

   +22 ns (max delay time of 20 ns tap of delay line)

   +10 ns (GAL22V10-10 max delay of D_ACK of MEM_GAL1)

   = 94 ns maximum

   Substituting into the above equation typical values (7 ns + 30 ns + 7 ns + 7 ns + 20 ns + 7 ns) gives a typical time of 78 ns.

3. All Write accesses and none-initial Read accesses DT_STB to D_ACK:

   10 ns (GAL20V8-10 max delay of ENEV~ or ENOD~ of MEM_GAL2 from previous access)

   +10 ns (GAL20V8-10 max delay of DT_STB to EVEND~ or ODDD~ of MEM_GAL2)

   +22 ns (max delay time of 20 ns tap of delay line)

   +10 ns (GAL22V10-10 max delay D_ACK of MEM_GAL1)

   = 52 ns

   Substituting into the above equation typical values (7 ns + 7 ns + 20 ns + 7 ns) gives a typical access time of 41 ns.

4. Worst case and typical minimum time between D_ACK to the next D_ACK is:

   52 ns (Worst Case, see #3 above) + Xns,

   41 ns (Typical Case, see #3 above) + Xns. Where 'X' is the time from D_ACK to the next DT_STB.

5. Worst case and typical minimum time between D_ACK to the next D_ACK to the same memory bank (Even address access to the next Even address access, or Odd to Odd address access):

   10 ns (GAL20V8-10 max delay of DT_STB to EVEND~ or ODDD~ of MEM_GAL2)

   +84 ns (max delay time of 80 ns tap of delay line)

   +10 ns (GAL20V8-10 max delay of LE80~ of MEM_GAL3)

   +10 ns (GAL20V8-10 max delay of ENEV~ of MEM_GAL2)

   = 114 ns Worst Case Minimum

   The typical minimum time would be (7 ns + 80 ns + 7 ns + 7 ns) 101 ns. This is where the 20 Mega-transfers per second transfer rate comes from (two transfers in 100 ns is 20 Mega-transfers per second).

   Here the minimum time for enabling EVEND~ after the previous EVEND~ was enabled was calculated to be the same as Even address D_ACK from the previous Even address D_ACK.

6. Worst case Data Setup time to D_ACK during Read access cycles (assume that 74F543 Octal Registered Transceivers are used):

3 ns (GAL20V8-10 min delay of DT_STB to EVEND~ or ODDD~ of MEM_GAL2)

+18 ns (minimum delay of 20 ns tap of delay line, starting from EVEND~ or ODDD~ of MEM_GAL2)

+8 ns (D_ACK delay of MEM_ACCESS)

−10 ns (worst case delay of DT_STB to EN_F_D0~ or EN_F_D1~ of MEM_GAL1; 2 ns worst case difference between EN_F_D0~, 1~ and D_ACK delays, since they are in the same GAL and their delays will tend to track each other)

−12 ns (74F543 max delay of OEBA~ or OEAB~ to output valid)

= 7 ns

7. COLINC setup time to the DRAM CAS~ low (the DP8422A-25 dictates that 39 ns minimum is necessary to guarantee a column address setup time of 0 ns, see parameter #27 in DP8422A-25 data sheet. We will calculate this value starting from the 30 ns tap of the delay line that asserts COLINC):

   −10 ns (GAL20V8-10 max delay of COLINC of MEM_GAL4, 5)

   +47 ns (minimum delay from 30 ns tap to 80 ns tap of delay line)

   +3 ns (GAL16V8-10 minimum delay of LE80~ of MEM~GAL3)

   +3 ns (GAL20V8-10 minimum delay of ENEV~ of MEM_GAL2)

   +3 ns (GAL20V8-10 minimum delay of EVEND~ of MEM_GAL2)

   +8 ns (GAL20V8-10 delay of CAS~ or MEM_GAL4, 5; this gives a 2 ns worst case difference between COLINC and CAS~ delays, since they are in the same GAL) = 54 ns.

   This value translates into 54 ns−39 ns (COLINC to column address valid on DP8422A-25 outputs) = 15 ns. Therefore the worst case column address setup time to CAS~ low is 15 ns.

8. Worst case data setup time to D_ACK during Read Access Cycles, calculated from CAS~ low to data valid, assuming 80 ns DRAMs. The DRAM access time can be calculated as follows; #7 above calculated 15 ns column address setup time to CAS~ low. Since the column address access time of an 80 ns DRAM is 40 ns, this gives 40 − 15 = 25 ns real worst case DRAM access time. Since the CAS~ access time of an 80 ns DRAM is 20 ns, the 25 ns value calculated above is used as the real CAS~ low to data valid access time of the DRAM.:

   −10 ns (GAL20V8-10 max delay of EVEND~ or ODDD~ to CAS~ low of MEM_GAL4, 5)

   −25 ns (CAS~ low to data valid of the DRAMs)

   −12 ns (74F543 max delay of enable to data valid)

   +28 ns (min delay of 30 ns tap of delay line)

   +3 ns (GAL20V8-10 min delay of ENOD~ or ENEV~ of MEM_GAL2)

   +3 ns (GAL20V8-10 min delay of ODDD~ or EVEND~ of MEM_GAL2)

   +18 ns (min delay of 20 ns tap of delay line)

   +3 ns (GAL20V8-10 min delay of D_ACK of MEM_GAL1)

   = 8 ns worst case data setup.

9. Worst case Column Address Hold time from CAS~ low (80 ns DRAMs require 20 ns), calculated from EVEND~ or ODDD~:

   28 ns (min delay of 30 ns tap of delay line)

   +8 ns (GAL20V8-10 delay of COLINC of MEM_GAL4, 5)

   +3 ns (assumed min delay of DP8422A-25 COLINC to 'Q' address outputs starting to change)

   −10 ns (GAL20V8-10 max delay of CAS~ low of MEM_GAL4, 5, this gives a 2 ns worst case difference between COLINC and CAS~ since they are in the same GAL)

   = 29 ns

14.0 GAL INPUT AND OUTPUT DEFINITION

14.1 Definition of Memory Module State Variables

EVEND~ — This output is driven low during an access to the even address bank as indicated by either a rising or falling edge of 'DT_STB' (depending upon whether the initial access was an even or odd bank access as indicated by 'B1'). This output drives the Even Delay Line (EVENDEL) whose outputs drive the other PALs that control the operation of the Memory Module.

ODDD~ — This output is driven low during an access to the odd address bank as indicated by either a rising or falling edge of 'DT_STB (depending upon whether the initial access was an even or odd bank access as indicated by 'B1'). This output drives the Odd Delay Line (ODDDEL) whose outputs drive the other PALs that control the operation of the Memory Module.

14.2 Definition of Memory Module Control Inputs

CLK — The Memory Module system CLocK (20 MHz was used in this design, though any frequency could be used up to 25 MHz). This clock drives the DP8422A-25 and synchronizes the RFSH~ and ADD_ACK~ outputs. It also synchronizes the AREQ~ input to the DP8422A-25.

AREQ~ — This input, Access REQuest, starts a Memory Module access along with Chip Select (CS~).

CS~ — The Chip Select input selects an access to the Memory Module when low.

READ — An active high signal indicating a READ access from the DRAM when asserted.

B1 — The least significant double-word (32 bits) or quad-word (64 bits) address bit, depending upon whether the Memory Module is 32 bits or 64 bits wide. During an access where the initial access is an even address from the even address bank (Bank 0), B1 will equal 0. During an access where the initial access is an odd address from the odd address bank (Bank 1), B1 will equal 1.

WR — A latched and inverted version of the WRite command (WR*) from Futurebus+. This command indicates that a write access is in progress when asserted and is held latched until AS* is released.

IV — A inverted version of the InterVention (IV*) status bit from Futurebus+. When this bit is asserted during a read access it indicates that another module is intervening in the current access and providing the data. The slave memory module will perform a write access, in order to update the memory with the correct data, instead of performing the requested read access. See Section 4 for more information.

DT_STB — This is the DaTa STroBe and must be generated from the module that interfaces to this Memory Module. This signal is edge sensitive. In other words, each rising or falling edge occurring after AREQ~ and ADD_ACK~ are low and before AREQ~ transitions high means that the Control Module requires that data be either read or written from/to the DRAM array. The initial transition on this input will always be a rising edge, and the B1 input defines whether this edge is an even or odd address access.

B0_RFRQ~ — This is the ReFresh ReQuest output of the Bank 0 DP8422A-25 DRAM controller. This output signals that a refresh of the DRAM should be performed as soon as is convenient, but within the next 15 μs.

B0_RFIP~ — This is the ReFresh In Progress output of the Bank 0 DRAM controller. This output signals that a DRAM refresh is currently in progress.

B0_DTACK~ — This is the DaTa ACKnowledge output of the Bank 0 DRAM controller. This output signals that the requested DRAM access is currently in progress. This output will not go low until RAS~ goes low for the current access guaranteeing that the previous access and all RAS~ precharge requirements from the present access have been met. See the DP8422A-25 data sheet for more details.

G_F_BUS~ — This is the Grant FutureBUS signal from an external arbiter. This signal would be needed in a board design where several different types of devices would be accessing the Memory Module. When this signal is low, it signifies that the high speed memory design protocol explained in this application note currently has control of the DRAM. When high, a normal CPU access can take place over a local bus with its own separate transceivers and control logic.

CPU_WE~ — A Write Enable signal from a local on-board CPU, if one exists.

BE~(3:0) — Byte enable signals for a 32-bit wide Memory Module that is assumed by this design. These inputs are needed to control the byte CAS~ inputs to the DRAM array. For a 64-bit wide Memory Module the interface would have to be redesigned slightly to accommodate 8 Byte Enable signals.

ECAS~ — A CAS~ enable signal for a local on-board CPU to control along with the BE~(3:0) signals. This input allows an on-board CPU to control the byte CAS~s to the DRAMs.

B0CAS0~ — The Bank 0 DRAM controller B0CAS~0 output.

B1CAS0~ — The Bank 1 DRAM controller B1CAS~0 output.

14.3 Definition of Memory Module Control Outputs

B0_CAS~(3:0) — These are the Bank 0 Byte CAS~ outputs that drive the Bank 0 DRAM array.

B1_CAS~(3:0) — These are the Bank 1 Byte CAS~ outputs that drive the Bank 1 DRAM array.

CSGAREQ~ — This signal indicates a Chip Selected and arbitration Granted Access REQuest to the DRAM array.

RFSH~ — This output to the DRAM controller causes a DRAM ReFreSH cycle to occur as soon as possible.

COLINC0 — This is the COLumn INCrement (COLINC) input to the Bank 0 DRAM controller.

COLINIC1 — This is the COLumn INCrement (COLINC) input to the Bank 1 DRAM contoller.

WE~ — This is the Write Enable input to the DRAM array.

D_ACK — This is the Data ACKnowledge output response to the DT_STB access input request. A rising or falling edge of this signal signifies that the desired read or write access that was requested earlier by DT_STB has been completed and a new access request, via the DT_STB input, may be initiated.

ACIP~ — A signal from the Memory Module Controller indicating that an ACcess is still in Progress. The FB_ADD GAL will not end the current memory access (AREQ~) or allow AK* to be released until ACIP~ is deasserted.

14.4 Definition of Other Signals in the Design

ROW(10:0) — ROW Address to the DRAM Controller.

COL(10:0) — COLumn Address to the DRAM Controller.

B(1:0) — Bank Address to the DRAM controller. In this design the bank inputs do not cause any change in the function of the DRAM controller, since the DP8422A-25 has been programmed in the all RAS~ low during an access mode. However, the "B1" input is used to signal whether the initial access is to the even address bank 0 or the odd address bank 1.

WAITIN~ Allows wait states to be added to an access (see DRAM controller data sheet).

DISRFSH~ Controls whether internal automatic refreshes or externally controlled refreshes are enabled (see the DRAM controller data sheet).

ML~ Used to program the DRAM controller (see the DRAM controller data sheet).

DIB0(7:0) These are the data inputs to the bank 0 DRAMs. This simulation only looked at 8 bits, but in a normal design, each bank of DRAM would contain 32 or 64 data bits.

DIB1(7:0) These are the data inputs to the bank 1 DRAMs. This simulation only looked at 8 bits, but in a normal design, each bank of DRAM would contain 32 or 64 data bits.

DOB0(7:0) These are the data outputs of the bank 0 DRAMs. This simulation only looked at 8 bits, but in a normal design, each bank of DRAM would contain 32 or 64 data bits.

DOB1(7:0) These are the data outputs of the bank 1 DRAMs. This simulation only looked at 8 bits, but in a normal design, each bank of DRAM would contain 32 or 64 data bits.

D(7:0) This is the I/O data bus on the other side of the Memory Module transceivers. This simulation only looked at 8 bits, whereas a normal design might contain 32 or 64 bits.

14.5 Other Signals from MEM_GAL2

ENOD~ ENable an ODd address access to occur.

ENEV~ ENable an EVen address access to occur.

INITACC~ The INITial ACCess (first access) is in progress.

F_DO~ A delayed version of DT_STB during an odd address access.

F_DE~ A delayed version of DT_STB during an even address access.

INITODD~ Indicates that the initial access is an odd address access. This is useful during read accesses, because during the initial odd bank access, COLINC0 must be asserted so that the next even address access is to the next sequential word.

14.6 Other Signals from MEM_GAL3

LE80~ A latched version of E80~ low. This output stores the E80~ low transition, and is reset by E30~ transitioning low.

LO80~ A latched version of O80~ low. This output stores the O80~ low transition, and is reset by O30~ transitioning low.

ADD_ACK~ This output signals that the address phase of a DRAM access is complete, ADDress phase ACKnowledge. This output is synchronized to the system clock (CLK), and indicates that data can now be read or written to the DRAM array (DT_STB can now transition).

14.7 Other Signals from MEM_GAL1

READ An active high signal indicating a READ access from the DRAM when asserted. This output is derived from the latched WR* command bit from Futurebus+ and held latched in this GAL until the DRAM access is completed as indicated by CSGAREQ~ being deasserted.

EN_F_D0~ This output enables the transceiver to output data from the Bank 0 DRAM array, DOB0(7:0), to the data I/O bus, D(7:0).

EN_F_D1~ This output enables the transceiver to output data from the Bank 1 DRAM array, DOB1(7:0), to the data I/O bus, D(7:0).

WLE0~ This output latches the data from 'D(7:0)' into the transceiver to write to the Bank 0 DRAM, DIBO(7:0) when high.

WLE1~ This output latches the data from 'D(7:0)' into the transceiver to write to the Bank 1 DRAM, DIB1(7:0) when high.

14.8 Other Signals from CAS_GAL1 and CAS_GAL0

RLE0~ This output latches the data out of Bank 0 of the DRAM, DOB0(7:0), into the transceiver to be output to the data I/O bus, D(7:0) when high.

RLE1~ This output latches the data out of Bank 1 of the DRAM, DOB1(7:0), into the transceiver to be output to the data I/O bus, D(7:0) when high.

WE_F_D~ This output enables the daa to be written into the DRAM (D(7:0)) through the 74F543 transceivers when low.

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN

```
MODULE    S_GAL1
TITLE     'S_GAL1, THIS IS THE FUTUREBUS+ DATA PHASE CONTROLLER GAL
          S_GAL1.ABL
          RUSTY MEIER 02.22.90'
S_GAL1 device  'P22V10';
CS_AS~      pin 1;      VCC          pin 24;
WR          pin 2;      DS           pin 23;
IV          pin 3;      DII_10       pin 22;
ADD_ACK~    pin 4;      AI           pin 21;
ADD_ACK_10~ pin 5;      DT_STB       pin 20;
D_ACK       pin 6;      LEI_A_D      pin 19;
D_ACK_10    pin 7;      CKO_D_S_C    pin 18;
DK_IN       pin 8;      DI           pin 17;
DI_IN       pin 9;      DK           pin 16;
AS_30       pin 10;     DKI_10       pin 15;
AS_40       pin 11;     DS_10        pin 14;
GND         pin 12;     F_INITACC~   pin 13;
EQUATIONS
!AI       = (AS_30 & AS_40);                                  " BRING AI VALID ONCE THE ADDRESS STATUS IS VALID
DT_STB    = (!CS_AS~ & !WR & !IV & DS & !F_INITACC~ &         " REQUEST -READ- MEMORY ACCESS DURING THE
                                        !ADD_ACK~)            "   INITIAL F_BUS REQUEST ONCE THE DRAM
                                                              "   ADDRESS ACKNOWLEDGE IS RECEIVED
          # (!CS_AS~ & !WR & !IV & !ADD_ACK~ & !DS &          " REQUEST READ MEMORY ACCESS DURING
                                    !D_ACK & F_INITACC~)      "   SUBSEQUENT F_BUS DS READ REQUESTS
          # (DT_STB & !CS_AS~ & !WR & !IV & !DS &             " HOLD DT_STB HIGH WHILE DS IS LOW
                                              !ADD_ACK~)
          # (DT_STB & !CS_AS~ & !WR & !IV & !D_ACK &          " HOLD DT_STB HIGH UNTIL D_ACK CHANGES
                                              !ADD_ACK~)
          # (!CS_AS~ & WR & !IV & !ADD_ACK~ & F_INITACC~      " REQUEST -WRITE- MEMORY ACCESS DURING A
                          & DS & DS_10 & !D_ACK_10)           "   F_BUS DS WRITE REQUEST
          # (DT_STB & !CS_AS~ & WR & !IV & !ADD_ACK~ &        " HOLD DT_STB HIGH WHILE D_ACK IS LOW
                            F_INITACC~ & !D_ACK_10)
          # (DT_STB & !CS_AS~ & WR & !IV & !ADD_ACK~ &        " HOLD DT_STB HIGH UNTIL DS HAS BEEN
                              F_INITACC~ & DS_10)             "   LOW AND D_ACK HIGH FOR 10NS
          # (!CS_AS~ & IV & !ADD_ACK~ & F_INITACC~ & DS       " REQUEST -INTERVENTION- MEMORY ACCESS
                        & DK_IN & DKI_10 & !D_ACK_10)
          # (DT_STB & !CS_AS~ & IV & !ADD_ACK~ &              " HOLD DT_STB HIGH DURING INTERVENTION
                            F_INITACC~ & !DI_IN)
          # (DT_STB & !CS_AS~ & IV & !ADD_ACK~ &              " HOLD DT_STB HIGH DURING INTERVENTION
                           F_INITACC~ & !DII_10)
          # (DT_STB & !CS_AS~ & IV & !ADD_ACK~ &              " HOLD DT_STB HIGH DURING INTERVENTION
                            F_INITACC~ & !D_ACK_10)
          # (DT_STB & !CS_AS~ & IV & !ADD_ACK~ &              " HOLD DT_STB HIGH DURING INTERVENTION
                            F_INITACC~ & DS & DK);
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

```
LEI_A_D   =  (!AS_30)                                    " LET ADDRESS FALL-THRU
          # (!CS_AS~ & !IV & !ADD_ACK~ & WR & DS &       " FALL-THRU LATCHES DURING ODD BEAT WRITES
                                !D_ACK & !DS_10)
          # (!CS_AS~ & !IV & !ADD_ACK~ & WR & DS &       " FALL-THRU DURING ODD BEAT WRITE ACCESSES
                              !D_ACK & D_ACK_10)
          # (!CS_AS~ & !IV & !ADD_ACK~ & WR & DS &       " FALL-THRU DURING ODD BEAT WRITE ACCESSES
                            !D_ACK & ADD_ACK_10~)
          # (!CS_AS~ & !ADD_ACK~ & WR & !IV & !DS &      " FALL-THRU DURING EVEN BEAT WRITES
                                DS_10 & D_ACK)
          # (!CS_AS~ & !ADD_ACK~ & WR & !IV & !DS &      " FALL-THRU DURING EVEN BEAT WRITES
                              D_ACK & !D_ACK_10)
          # (!CS_AS~ & !ADD_ACK~ & IV & DK_IN & DS &     " FALL-THRU DURING ODD BEAT INTERVENTION
                              !DKI_10 & !D_ACK)
          # (!CS_AS~ & !ADD_ACK~ & IV & DK_IN & DS &     " FALL-THRU DURING ODD BEAT INTERVENTION
                            !D_ACK & D_ACK_10)
          # (!CS_AS~ & !ADD_ACK~ & IV & DK_IN & DS &     " FALL-THRU DURING ODD BEAT INTERVENTION
                            !D_ACK & ADD_ACK_10~)
          # (!CS_AS~ & !ADD_ACK~ & IV & DI_IN & !DS &    " FALL-THRU DURING EVEN BEAT INTERVENTION
                              !DII_10 & D_ACK)
          # (!CS_AS~ & !ADD_ACK~ & IV & DI_IN & !DS &    " FALL-THRU DURING EVEN BEAT INTERVENTION
                    D_ACK & !D_ACK_10);
CKO_D_S_C = (AS_30 & !AS_40 & AI)                        " CLOCK STATUS AND CAPABILITIES
                                                           OUT TO THE FUTUREBUS+
          # (!CS_AS~ & !IV & !ADD_ACK~ & DS & !WR &      " CLOCK DATA, STATUS AND CAPABILITY OUT
                              D_ACK & !DS_10)            "   DURING ODD BEAT READ ACCESSES
          # (!CS_AS~ & !IV & !ADD_ACK~ & DS & !WR &      " CLOCK DURING ODD BEAT READ ACCESSES
                            D_ACK & !D_ACK_10)
          # (!CS_AS~ & !IV & !ADD_ACK~ & WR & DS &       " CLOCK STATUS AND CAPABILITY OUT DURING
                              !D_ACK & !DS_10)             ODD BEAT WRITE ACCESSES
          # (!CS_AS~ & !IV & !ADD_ACK~ & WR & DS &       " CLOCK DURING ODD BEAT WRITE ACCESSES
                            !D_ACK & D_ACK_10)
          # (!CS_AS~ & !IV & !ADD_ACK~ & WR & DS &       " CLOCK DURING ODD BEAT WRITE ACCESSES
                          !D_ACK & ADD_ACK_10~)
          # (!CS_AS~ & !IV & !ADD_ACK~ & !DS & !WR &     " CLOCK DURING EVEN BEAT READ ACCESSES
                    F_INITACC~ & !D_ACK & DS_10)
          # (!CS_AS~ & !IV & !ADD_ACK~ & !DS & !WR &     " CLOCK DURING EVEN BEAT READ ACCESSES
                  F_INITACC~ & !D_ACK & D_ACK_10)
          # (!CS_AS~ & !IV & !ADD_ACK~ & WR & !DS &      " CLOCK DURING EVEN BEAT WRITE ACCESSES
                    F_INITACC~ & D_ACK & DS_10)
          # (!CS_AS~ & !IV & !ADD_ACK~ & WR & !DS &      " CLOCK DURING EVEN BEAT WRITE ACCESSES
                  F_INITACC~ & D_ACK & !D_ACK_10)
          # (!CS_AS~ & IV & !ADD_ACK~ & DS & DK_IN &     " CLOCK DURING ODD BEAT INTERVENTION ACCESS
                              !D_ACK & !DKI_10)
          # (!CS_AS~ & IV & !ADD_ACK~ & DS & DK_IN &     " CLOCK DURING ODD BEAT INTERVENTION ACCESS
                              !D_ACK & D_ACK_10)
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

```
        # (!CS_AS~ & IV & !ADD_ACK~ & DS & DK_IN &          " CLOCK DURING ODD BEAT INTERVENTION ACCESS
                        !D_ACK & ADD_ACK_10~)
        # (!CS_AS~ & IV & !ADD_ACK~ & F_INITACC~ &         " CLOCK DURING EVEN BEAT INTERVENTION ACCESSES
                D_ACK & !DS & DI_IN & !DII_10)
        # (!CS_AS~ & IV & !ADD_ACK~ & F_INITACC~ &         " CLOCK DURING EVEN BEAT INTERVENTION ACCESSES
              D_ACK & !DS & DI_IN & !D_ACK_10)
        # (CS_AS~ & AS_30);                                " CLOCK DURING DISCONNECT
DI      = (!CS_AS~ & !IV & AS_30 & !DS)                    " MAKE DI HIGH FROM DS BEING LOW
        # (!CS_AS~ & !IV & DI & !DS_10)                    " HOLD DI HIGH (DI* ON F_BUS IS LOW) UNTIL 10NS AFTER THE
                                                           "   ODD DATA BEAT STARTS FOR READS AND WRITES
        # (!CS_AS~ & !IV & DI & WR & ADD_ACK_10~)          " HOLD DI FOR THE INITIAL WRITE ACCESS
        # (!CS_AS~ & !IV & DI & !WR &                      " HOLD DI HIGH UNTIL 10NS AFTER D_ACK DURING READS
                              !D_ACK_10)
        # (!CS_AS~ & !IV & WR & DI & !ADD_ACK~             " HOLD DI HIGH UNTIL 10NS AFTER D_ACK DURING WRITES
                            & D_ACK_10)
        # (!CS_AS~ & IV & AS_30 & !F_INITACC~)             " START DI DURING THE INITIAL ACCESS OF INTERVENTION
        # (!CS_AS~ & IV & DI & ADD_ACK_10~)                " HOLD DI FOR THE INITIAL INTERVENTION ACCESS
        # (!CS_AS~ & IV & DI_IN & !DS &                    " DURING INTERVENTION WAIT UNTIL DI_IN IS VALID
                              !ADD_ACK~)                   "   BEFORE BRINGING DI HIGH
        # (!CS_AS~ & IV & DI & !ADD_ACK~ & !DK)            " HOLD DI HIGH UNTIL 10NS AFTER THE ODD DATA BEAT
        # (!CS_AS~ & IV & DI & !ADD_ACK~ &                 " STARTS DURING INTERVENTION
                                !DKI_10)
        # (!CS_AS~ & IV & DI & !ADD_ACK~ &                 " HOLD DI HIGH UNTIL 10NS AFTER D_ACK DURING INTERVENTION
                              D_ACK_10);
DK      = (!CS_AS~ & !IV & DS & !WR &                      " MAKE DK HIGH FROM DS BEING HIGH (DS* ON F_BUS IS LOW)
                    !ADD_ACK~ & D_ACK)                     "   AND D_ACK IS HIGH DURING READ ACCESSES
        # (!CS_AS~ & !IV & DS & WR & !ADD_ACK~)            " MAKE DK HIGH FROM DS BEING HIGH DURING WRITES
        # (!CS_AS~ & !IV & DK & !ADD_ACK~ &                " HOLD DK HIGH (DK* ON F_BUS IS LOW) UNTIL 10NS AFTER
                                DS_10)                     "   THE EVEN DATA BEAT STARTS FOR READS OR WRITES
        # (!CS_AS~ & !IV & DK & !ADD_ACK~ &                " HOLD DK HIGH UNTIL 10NS AFTER D_ACK
                            !WR & D_ACK_10)
        # (!CS_AS~ & !IV & DK & !ADD_ACK~ &                " HOLD DK HIGH UNTIL 10NS AFTER D_ACK DURING WRITES
                            WR & !D_ACK_10)
        # (!CS_AS~ & IV & DK_IN & DS & !ADD_ACK~)          " DURING INTERVENTION WAIT UNTIL DK_IN IS VALID
        # (!CS_AS~ & IV & DK & !ADD_ACK~ & !DI)            " HOLD DK HIGH UNTIL 10NS AFTER THE EVEN DATA BEAT
        # (!CS_AS~ & IV & DK & !ADD_ACK~ & !DII_10)        " STARTS DURING INTERVENTION
        # (!CS_AS~ & IV & DK & !ADD_ACK~ & !D_ACK_10);
END;
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

```
MODULE    S_GAL2
TITLE     'S_GAL2, THIS IS THE FUTUREBUS+ ADDRESS PHASE CONTROLLER GAL
S_GAL2.ABL
RUSTY MEIER 02.22.90'
S_GAL2 device  'P20V8R';
CLK          pin 1;      VCC          pin 24;
AS           pin 2;      ACIP~        pin 23;
CS~          pin 3;      CS_AS~       pin 22;
WR           pin 4;      AREQA~       pin 21;
IV           pin 5;      AREQ~        pin 20;
ADD_ACK~     pin 6;      F_INITACC~   pin 19;
AI           pin 7;      AK           pin 18;
AI_IN        pin 8;      LEI_ADD_CM   pin 17;
AS_20        pin 9;      WRITE~       pin 16;
AS_30        pin 10;     LEI_S_C      pin 15;
DS           pin 11;     DK           pin 14;
GND          pin 12;     OE~          pin 13;
EQUATIONS
!CS_AS~   = (!CS~ & AS);              " IF CHIP SELECTED AND FUTUREBUS+ 'AS' IS VALID BRING LOW
!AREQA~   := (AS);                    " SYNCHRONIZE FUTUREBUS+ AS* TO ON-BOARD CLOCK IN PREPARATION TO
                                      "   PRODUCE AREQ~.
!AREQ~    := (!CS~ & AS & AS_30 &     " SECOND SYNCHRONIZATION STAGE, INCLUDING CS~, AS, AND DELAYED AS
                       !AREQA~)       "   TO PRODUCE DRAM CONTROLLER ACCESS REQUEST
          # (!AREQA~ & !ACIP~);       " HOLD UNTIL DRAM ACCESS IS FINISHED
!F_INITACC~ = (!CS~ & AS & ADD_ACK~)              " START THE INITIAL ACCESS
          # (!CS~ & AS & !F_INITACC~ & DS & !DK); " CONTINUE INITACC~ UNTIL THE FUTUREBUS+ ODD
                                                  "   BEAT IS IN PROGRESS
AK        = (AS)                          " IF CHIP SELECTED BUS ACCESS MAKE AK VALID
          # (AK & AS_30)                  " HOLD UNTIL END OF ACCESS AND STATUS IS VALID
          # (AK & !ACIP~);                "   OR UNTIL DRAM ACCESS IS COMPLETED
!LEI_ADD_CM  = (AS_20);                   " LATCH THE INPUT ADDRESS AND COMMAND
!WRITE    = (AI)                          " ENABLE ADDRESS ONTO BOARD WHILE AI IS HIGH
          # (AS & !AS_30)                 " ENABLE ADDRESS ONTO BOARD FROM AS UNTIL 30NS AFTER AS
          # (!CS~ & AS & AS_30 & WR)      " ENABLE DATA ONTO BOARD DURING WRITE ACCESS
          # (!CS~ & AS & AS_30 & IV)      " ENABLE DATA ONTO BOARD DURING INTERVENTION
          # (CS~);                        " ENABLE DATA ONTO BOARD IF NOT CHIP SELECTED
!LEI_S_C  = (AS & AS_30 & !AI_IN)         " LATCH THE INPUT STATUS AND CAPABILITY
          # (!LEI_S_C & AK);
END;
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

```
MODULE    MEM_GAL2
TITLE     'MEM_GAL2, THIS IS THE FUTUREBUS+ ACCESS DELAY LINE CONTROLLER GAL
MEM_GAL2.ABL
RUSTY MEIER 1101.89'
MEM_GAL2 device  'P20V8C';
CSGAREQ~    pin 1;      VCC         pin 24;
B1          pin 2;      ED~         pin 23;
DT_STB      pin 3;      EVEND~      pin 22;
ADD_ACK~    pin 4;      INITODD~    pin 21;
E30~        pin 5;      F_DE~       pin 20;
LE80~       pin 6;      F_DO~       pin 19;
020~        pin 7;      INITACC~    pin 18;
030~        pin 8;      ENEV~       pin 17;
L080~       pin 9;      ENOD~       pin 16;
READ        pin 10;     ODDD~       pin 15;
AS          pin 11;     OD~         pin 14;
GND         pin 12;     NC          pin 13;
EQUATIONS
!EVEND~   = (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & !READ &        " INITIAL EVEN WRITE
                DT_STB & F_DE~ & !ENEV~ & !B1)
          #(!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & !ENEV~ &       " INITIAL EVEN READ
                !030~ & READ & !B1)
          # (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & DT_STB &       " INITIAL ODD READ
                F_DE~ & READ & B1)
          # (AS & !CSGAREQ~ & INITACC~ & !ADD_ACK~ & !ENEV~ &  " START 'EVEND~' FROM RISING OR FALLING
                DT_STB & F_DE~ & !B1)                          " 'DT_STB' DEPENDING UPON B1
          # (AS & !CSGAREQ~ & INITACC~ & !ADD_ACK~ & !ENEV~ &
                !DT_STB & F_DE~ & B1)
          # (!ED~ & !ENEV~ & E30~);                            " 'ENEV~' ENDS THIS TERM

!INITODD~  = (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ &             " START DURING AN INITIAL ACCESS TO THE ODD
                020~ & 030~ & READ & B1);                      " MEMORY BANK. THIS OUTPUT IS USED TO PROVIDE
                                                               " AN INITIAL 'COLINC' SIGNAL TO BANK 0 SO
                                                               " THAT THE NEXT SEQUENTIAL ACCESS TO THE EVEN
                                                               " BANK 0 RETRIEVES THE CORRECT DATA.
!F_DE~    = (!CSGAREQ~ & !ENEV~ & DT_STB & !B1)                      " DELAYED VERSION OF EVEN ACCESS
          # (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & DT_STB & 030 &      " 'DT_STB'
                                    READ & B1)
          # (!CSGAREQ~ & INITACC~ & !ENEV~ & !DT_STB & B1);
!F_DO~    = (!CSGAREQ~ & !ENOD~ & DT_STB & B1)                       " DELAYED VERSION OF ODD ACCESS
          # (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & DT_STB & E30 &      " 'DT_STB'
                                    READ & !B1)
          # (!CSGAREQ~ & INITACC~ & !ENOD~ & !DT_STB & !B1);
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

```
!INITACC~   = (!CSGAREQ~ & ADD_ACK~ & AS)                                  " END 'INITACC~' WITH '030~' OR 'E30~'
            # (!INITACC~ & E30~ & !B1 & AS)                                " DEPENDING UPON THE STATE OF B1
            # (!INITACC~ & 030~ & B1 & AS);
!ENEV~      = (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & !READ & !B1)            " INITIAL EVEN WRITE
            # (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & !030~ & !READ & B1)     " INITIAL ODD WRITE
            # (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & !030~ & READ & !B1)     " INITIAL EVEN READ
            # (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & E30~ & OD~ & 030~       " INITIAL ODD READ
                                                    & READ & B1)
            # (!CSGAREQ~ & !ADD_ACK~ & !LE80~ & !030~ & INITACC~)          " SUBSEQUENT ACCESSES
            # (!ENEV~ & !CSGAREQ~ & !ADD_ACK~ & E30~);                     " HOLD VALID UNTIL '!E30~'
!ENOD~      = (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & !E30~ & !READ & !B1)    " INITIAL EVEN WRITE
            # (!CSGAREQ~ & !INITACC~ & !ADD_ACK & B1 & !READ)              " INITIAL ODD WRITE
            # (!CSGAREQ~ & !INITACC~ & !ADD_ACK & 030~ & ED~ & E30~ &      " INITIAL EVEN READ
                                                    READ & !B1)
            # (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & !E30~ & READ & B1)      " INITIAL ODD READ
            # (!CSGAREQ~ & INITACC~ & !ADD_ACK~ & !L080~ & !E30~)          " SUBSEQUENT ACCESSES
            # (!ENOD~ & !CSGAREQ~ & !ADD_ACK~ & 030~);                     " HOLD VALID UNTIL !030~'
!ODDD~      = (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ &  !ENOD~ & DT_STB &    " INITIAL ODD WRITE
                                                  F_DO~ & !READ & B1)
            # (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & READ & B1 &            " INITIAL ODD READ
                                                 !ENOD~ & !E30~)
            # (!CSGAREQ~ & !INITACC~ & !ADD_ACK~ & DT_STB & F_DO~ &       " INITIAL EVEN READ
                                                    READ & !B1)
            # (AS & !CSGAREQ~ & INITACC~ & !ADD_ACK~ & !ENOD~ &           " START ODD DELAY FROM EITHER
                                          !DT_STB & F_DO~ & !B1)         " RISING OR FALLING 'DT_STB'
            # (AS & !CSGAREQ~ & INITACC~ & !ADD_ACK~ & !ENOD~ &           " DEPENDING UPON B1.
                                         DT_STB & F_DO~ & B1)
            # (!OD~ & !ENOD~ & 030~);                                      " 'ENOD~' ENDS THIS TERM
END;
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

```
MODULE    MEM_GAL1
TITLE     'MEM_GAL1, THIS GAL CONTAINS FUTUREBUS+ READ AND WRITE CONTROL SIGNALS.
          RUSTY MEIER 1016.89'
MEM_GAL1 device  'P22V10'
CSGAREQ~    pin 1;      VCC         pin 24;
B1          pin 2;      CPU_WE~     pin 23;
WR          pin 3;      EN_F_D0~    pin 22;
EVEND~      pin 4;      ACIP~       pin 21;
ODDD~       pin 5;      WLE0~       pin 20;
DT_STB      pin 6;      WLE1~       pin 19;
AS          pin 7;      WE~         pin 18;
E10~        pin 8;      READ        pin 17;
E20~        pin 9;      D_ACK       pin 16;
010~        pin 10;     EN_F_D1~    pin 15;
020~        pin 11;     IV          pin 14;
GND         pin 12;     ADD_ACK~    pin 13;
EQUATIONS
!EN_F_D0~= (!ADD_ACK~ & AS & DT_STB & !CSGAREQ~ & READ & !B1)      " ENABLE READ LATCH OUTPUTS FOR BANK 0
         # (!ADD_ACK~ & AS & !DT_STB & !CSGAREQ~ & READ & B1);
!ACIP~   = (!EVEND~)                                              " INDICATES THAT A DRAM ACCESS IS IN PROGRESS
         # (!E10~)
         # (!ODDD~)
         # (!010~)
         # (WLE0~)
         # (WLE1~);
WLE0~    = (!READ & !EVEND~)                                      " THIS SIGNAL RELIES ON THE PULSE WIDTH OF 'EVEND'
         #    (!READ & !E10~);                                    "  TO GUARANTEE APPROXIMATELY 40-60 ns PULSE WIDTH
                                                                  "  FOR THE WRITE LATCH FOR BANK 0 'WLE0~'.
WLE1~    = (!READ & !ODDD~)                                       " THIS SIGNAL RELIES ON THE PULSE WIDTH OF 'EVEND'
         #    (!READ & !010~);                                    "  TO GUARANTEE APPROXIMATELY 40-60 ns PULSE WIDTH
                                                                  "  FOR THE WRITE LATCH FOR BANK 1 'WLE1~'.
!WE~     = (!CSGAREQ~ & !READ)                                    " WRITE ENABLE TO THE DRAM ARRAY
         # (CSGAREQ~ & !CPU_WE~);
READ     = (!WR & !CSGAREQ~ & !IV)                                " GENERATE READ SIGNAL AND HOLD UNTIL AREQ~ IS HIGH
         # (READ & !CSGAREQ~ & !IV);                              " UNLESS INTERVENTION
D_ACK    = !((CSGAREQ~)                                           " FUTUREBUS+ ACCESS ACKNOWLEDGE HANDSHAKE.
         #   (!D_ACK & !CSGAREQ~ & READ & E20~ & !B1 & AS)        " HOLD DURING READ
         #   (!D_ACK & !CSGAREQ~ & READ & 020~ & B1 & AS)         " HOLD DURING WRITE
         # (!CSGAREQ~ & READ & !020~ & !B1 & AS)                  " TWO TERMS WERE USED FOR BOTH
         # (!CSGAREQ~ & READ & !E20~ & B1 & AS)                   " READS AND WRITES.
         # (!D_ACK & !CSGAREQ~ & !READ & E10~ & !B1 & AS)         " HOLD DURING WRITE
         # (!D_ACK & !CSGAREQ~ & !READ & 010~ & B1 & AS)          " HOLD DURING WRITE
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

```
          # ( !CSGAREQ~ & !READ & !010~ & !B1 & AS)
          # ( !CSGAREQ~ & !READ & !E10~ & B1 & AS)  );
!EN_F_D1~= ( !ADD_ACK~ & AS & !DT_STB & !CSGAREQ~ & READ & !B1)     " ENABLE READ LATCH OUTPUTS FOR BANK 1
          # ( !ADD_ACK~ & AS & DT_STB & !CSGAREQ~ & READ & B1);
END;
```

```
MODULE    MEM_GAL3
TITLE     'MEM_GAL3, THIS GAL IMPLEMENTS THE RANDOM LOGIC THAT WAS USED IN
          THIS DESIGN.
          RUSTY MEIER 1025.89'
MEM_GAL3  device  'P16V8R';
CLK         pin 1;      VCC         pin 20;
AREQ~       pin 2       CSGAREQ~    pin 19;
CS~         pin 3       RRFRQ~      pin 18;
READ        pin 4       RFIP~       pin 17;
E30~        pin 5       DTACK~      pin 16;
E80~        pin 6       ADD_ACK~    pin 15;
030~        pin 7       L080~       pin 14;
080~        pin 8       LE80~       pin 13;
G_F_BUS~    pin 9       RFSH~       pin 12;
GND         pin 10      OE~         pin 11;
EQUATIONS
!ADD_ACK~   := ( !G_F_BUS~ & !DTACK~ & !CS~ & !AREQ~);          " DRAM ADDRESS PHASE COMPLETE,
                                                                " READY TO START DATA TRANSFER PHASE.
!RFSH~      :=  ( !RFRQ~ & RFIP~);                              " DO EXTERNALLY CONTROLLED REFRESH
!CSGAREQ~   =  ( !CS~ & !AREQ~ & !G_F_BUS~);                    " DRAM ACCESS IN PROGRESS
!L080~      #  ( !080~ & !ADD_ACK~)                            " LATCH THAT 080~ HAS TRANSITIONED LOW
            #  ( !L080~ & 030~ & !ADD_ACK~);
!LE80~      =  ( !E80~ & !ADD_ACK~)                             " LATCH THAT E80~ HAS TRANSITIONED LOW
            #  ( !LE80~ & E30~  !ADD_ACK~);
END;
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

```
MODULE    MEM_GAL4
TITLE     'MEM_GAL4, THIS DEVICE IS USED TO CONTROL THE BANK 0 CAS INPUTS
          OF THE DRAMS, ALONG WITH THE READ LATCH FOR BANK 0 (RLE0), AND THE
          COLUMN INCREMENT INPUT (COLINC0) TO THE DRAM CONTROLLER FOR BANK 0.'
          " THE ECAS~ INPUT ALLOWS A LOCAL CPU (AND, OR I/O CONTROLLER IF USED) TO
          " CONTROL THE CAS INPUTS TO THE BANK 0 DRAMS. WHEN CSGAREQ~ IS LOW THE
          " FUTUREBUS ACCESS CONTROLS THE CAS INPUTS TO THE BANK 0 DRAMS.
          " RUSTY MEIER 1016.89
MEM_GAL4  device  'P20V8C';
READ        pin 1;      VCC        pin 24;
BE~0        pin 2;      CS~0       pin 23;
BE~1        pin 3;      CAS~0      pin 22;
BE~2        pin 4;      CAS~1      pin 21;
BE~3        pin 5;      RLE0~      pin 20;
ECAS~       pin 6;      COLINC0    pin 19;
ICAS~0      pin 7;      040~       pin 18;
CSGAREQ~    pin 8;      030~       pin 17;
INITODD~    pin 9;      CAS~2      pin 16;
E10~        pin 10;     CAS~3      pin 15;
E40~        pin 11;     CS~3       pin 14;
GND         pin 12;     ODDD~      pin 13;
EQUATIONS
!CAS~0    = (!ECAS~ & !ICAS~0 & !BE~0 & CSGAREQ~)                  " CAS~ DURING NON-F_BUS ACCESS
          # (!CSGAREQ~ & READ & !ODDD~ & !ICAS~0 & !BE~0)          " START F_BUS READ CAS~
          # (!CS~0 & !CSGAREQ~ & READ & 040~ & !ICAS~0 & !BE~0)    " HOLD F_BUS READ CAS~
          # (!CS~0 & !CSGAREQ~ & READ & !RLE0~ & !ICAS~0 & !BE~0)  " HOLD F_BUS READ CAS~ UNTIL DATA IS LATCHED
          # (!CSGAREQ~ & !READ & !E10~ & !ICAS~0 & !BE~0);         " F_BUS WRITE CAS~
!CAS~1    = (!ECAS~ & !ICAS~0 & !BE~1 & CSGAREQ~)                  " CAS~ DURING NON-F_BUS ACCESS
          # (!CSGAREQ~ & READ & !ODDD~ & !ICAS~0 & !BE~1)          " START F_BUS READ CAS~
          # (!CAS~1 & !CSGAREQ~ & READ & 040~ & !ICAS~0 & !BE~1)   " HOLD F_BUS READ CAS~
          # (!CAS~1 & !CSGAREQ~ & READ & !RLE0~ & !ICAS~0 & !BE~1)" HOLD F_BUS READ CAS~ UNTIL DATA IS LATCHED
          # (!CSGAREQ~ & !READ & !E10~ & !ICAS~0 & !BE~1);         " F_BUS WRITE CAS~
RLE0~     =  (READ & !040~)                                        " RLE0~ HIGH TO LATCH BANK 0 READ DATA
          #     (RLE0~ & READ & ODDD~);                            " HOLD RLE0~ HIGH
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

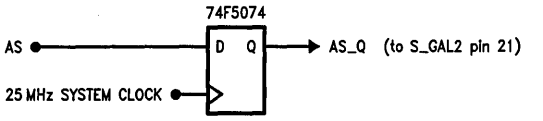
```
COLINCO   =  !( (CSGAREQ~)                                          " COLINCO LOW IF NOT ACCESSING
          #     (!COLINCO & INITODD~ & 030~ & READ)                 " HOLD LOW DURING READ UNTIL !030~ UNLESS !INITODD~
          #     (!COLINCO & E10~ & 030~ & READ)                     " HOLD LOW DURING READ
          #     (!CSGAREQ~ & !E40~ & 030~ & READ)                   " COLINCO LOW DURING READ ACCESSES
          #     (!CSGAREQ~ & INITODD~ & 030~ & READ)                " COLINCO LOW DURING READS UNLESS !INITODD~
          #     (!COLINCO & E40~ & !READ)                           " HOLD LOW DURING WRITE UNTIL !E40~
          #     (!CSGAREQ~ & E40~ & !READ) );                       " COLINCO LOW DURING WRITE ACCESSES
!CAS~2    =  (!ECAS~ & !ICAS~0 & !BE~2 & CSGAREQ~)                  " CAS~ DURING NON-F_BUS ACCESS
          #  (!CSGAREQ~ & READ & !ODDD~ & !ICAS~0 & !BE~2)          " START F_BUS READ CAS~
          #  (!CAS~2 & !CSGAREQ~ & READ & 040~ & !ICAS~0 & !BE~2)   " HOLD F_BUS READ CAS~
          #  (!CAS~2 & !CSGAREQ~ & READ & !RLE0~ & !ICAS~0 & !BE~2) " HOLD F_BUS READ CAS~ UNTIL DATA IS LATCHED
          #  (!CSGAREQ~ & !READ & !E10~ & !ICAS~0 & !BE~2);         " F_BUS WRITE CAS~
!CAS~3    =  (!ECAS~ & !ICAS~0 & !BE~3 & CSGAREQ~)                  " CAS~ DURING NO-F_BUS ACCESS
          #  (!CSGAREQ~ & READ & !ODDD~ & !ICAS~0 & !BE~3)          " START F_BUS READ CAS~
          #  (!CS~3 & !CSGAREQ~ & READ & 040~ & !ICAS~0 & !BE~3)    " HOLD F_BUS READ CAS~
          #  (!CS~3 & !CSGAREQ~ & READ & !RLE0~ & !ICAS~0 & !BE~3)  " HOLD F_BUS READ CAS~ UNTIL DATA IS LATCHED
          #  (!CSGAREQ~ & !READ & !E10~ & !ICAS~0 & !BE~3);         " F_BUS WRITE CAS~
END;
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

```
MODULE    MEM_GAL5
TITLE     'MEM_GAL5, THIS DEVICE IS USED TO CONTROL THE BANK 1 CAS INPUTS
          OF THE DRAMS, ALONG WITH THE READ LATCH FOR BANK 1 (RLE1), AND THE
          COLUMN INCREMENT INPUT (COLINC1) TO THE DRAM CONTROLLER FOR BANK 1.'
          " THE ECAS~ INPUT ALLOWS A LOCAL CPU (AND, OR I/O CONTROLLER IF USED) TO CONTROL
          " THE CAS INPUTS TO THE BANK 1 DRAMS. WHEN CSGAREQ~ IS LOW THE FUTUREBUS
          " ACCESS CONTROLS THE CAS INPUTS TO THE BANK 1 DRAMS.
          " RUSTY MEIER 1016.89
MEM_GAL5  device  'P20V8C';
READ         pin 1;      VCC        pin 24;
BE~0         pin 2;      CS~0       pin 23;
BE~1         pin 3;      CAS~0      pin 22;
BE~2         pin 4;      CAS~1      pin 21;
BE~3         pin 5;      RLE1~      pin 20;
ECAS~        pin 6;      COLINC1    pin 19;
ICAS~0       pin 7;      WE_F_D~    pin 18;
CSGAREQ~     pin 8;      040~       pin 17;
EVEND~       pin 9;      CAS~2      pin 16;
E30~         pin 10;     CAS~3      pin 15;
E40~         pin 11;     CS~3       pin 14;
GND          pin 12;     010~       pin 13;
EQUATIONS
!CAS~0   =  (!ECAS~ & !ICAS~0 & !BE~0 & CSGAREQ~)                    " CAS~ DURING NON-F_BUS ACCESS
         #  (!CSGAREQ~ & READ & !EVEND~ & !ICAS~0 & !BE~0)           " START F_BUS READ CAS~
         #  (!CS~0 & !CSGAREQ~ & READ & E40~ & !ICAS~0 & !BE~0)      " HOLD F_BUS READ CAS~
         #  (!CS~0 & !CSGAREQ~ & READ & !RLE1~ & !ICAS~0 & !BE~0)    " HOLD F_BUS READ CAS~ UNTIL DATA IS LATCHED
         #  (!CSGAREQ~ & !READ & !010~ & !ICAS~0 & !BE~0);           " F_BUS WRITE CAS~
!CAS~1   =  (!ECAS~ & !ICAS~0 & !BE~1 & CSGAREQ~)                    " CAS~ DURING NON-F_BUS ACCESS
         #  (!CSGAREQ~ & READ & !EVEND~ & !ICAS~0 & !BE~1)           " START F_BUS READ CAS~
         #  (!CAS~1 & !CSGAREQ~ & READ & E40~ & !ICAS~0 & !BE~1)     " HOLD F_BUS READ CAS~
         #  (!CAS~1 & !CSGAREQ~ & READ & !RLE1~ & !ICAS~0 & !BE~1)   " HOLD F_BUS READ CAS~ UNTIL DATA IS LATCHED
         #  (!CSGAREQ~ & !READ & !010~ & !ICAS~0 & !BE~1);           " F_BUS WRITE CAS~
RLE1~    =  (READ & !E40~)                                           " RLE1~ HIGH TO LATCH BANK 1 READ DATA
         #     (RLE1~ & READ & EVEND~);                             " HOLD RLE1~ HIGH
!WE_F_D~ =  (!READ & !CSGAREQ~);                                    " ENABLE WRITE DATA THRU
                                                                     " 74F543 TRANSCEIVERS
```

15.0 GAL EQUATIONS WRITTEN IN ABEL FOR THE MEMORY MODULE DESIGN (Continued)

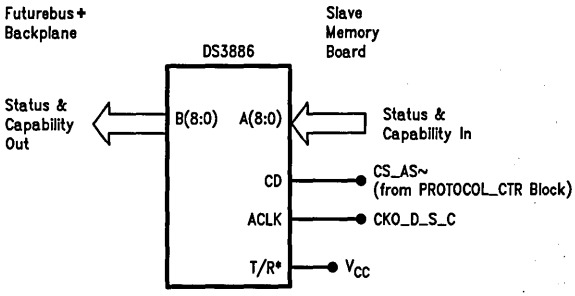
```
COLINC1  = !( (CSGAREQ~)                                          " COLINC1 LOW IF NOT ACCESSING
         #    (!COLINC1 & E30~ & READ)                            " HOLD LOW DURING READ UNTIL !E30~ UNLESS !INITODD~
         #    (!CSGAREQ~ & E30~ & READ)                           " LOW DURING READ
         #    (!COLINC1~ & 040~ & !READ)                          " HOLD LOW DURING WRITE ACCESS UNTIL !E40~
         #    (!CSGAREQ~ & 040~ & !READ)  );                      " COLINC1 LOW DURING WRITE ACCESS
!CAS~2   =  (!ECAS~ & !ICAS~0 & !BE~2 & CSGAREQ~)                  " CAS~ DURING NON-F_BUS ACCESS
         #  (!CSGAREQ~ & READ & !EVEND~ & !ICAS~0 & !BE~2)         " START F_BUS READ CAS~
         #  (!CAS~2 & !CSGAREQ~ & READ & E40~ & !ICAS~0 & !BE~2)   " HOLD F_BUS READ CAS~
         #  (!CAS~2 & !CSGAREQ~ & READ & !RLE1~ & !ICAS~0 & !BE~2) " HOLD F_BUS READ CAS~ UNTIL DATA IS LATCHED
         #  (!CSGAREQ~ & !READ & !010~ & !ICAS~0 & !BE~2);         " F_BUS WRITE CAS~
!CAS~3   =  (!ECAS~ & !ICAS~0 & !BE~3 & CSGAREQ~)                  " CAS~ DURING NO-F_BUS ACCESS
         #  (!CSGAREQ~ & READ & !EVEND~ & !ICAS~0 & !BE~3)         " START F_BUS READ CAS~
         #  (!CS~3 & !CSGAREQ~ & READ & E40~ & !ICAS~0 & !BE~3)    " HOLD F_BUS READ CAS~
         #  (!CS~3 & !CSGAREQ~ & READ & !RLE1~ & !ICAS~0 & !BE~3)  " HOLD F_BUS READ CAS~ UNTIL DATA IS LATCHED
         #  (!CSGAREQ~ & !READ & !010~ & !ICAS~0 & !BE~3);         " F_BUS WRITE CAS~
END;
```



TL/L/10772–38

* Revised S_GAL2 AREQ~ equation
!AREQ~ = AS_Q & AS & !CS~ & AS_30
!AREQ~ & !ACIP~

FIGURE 34. Improving Performance during the Initial Memory Module Access



TL/L/10772–39

FIGURE 35. Status and Capability Driver Connection Diagram

Futurebus+ I/O Board Design

National Semiconductor
Application Note 751
Webster (Rusty) Meier Jr.,
David Hawley, and
Shilpa Parikh

1.0 INTRODUCTION

This application note describes a Futurebus+ I/O board design. The first part of this application note contains an introduction to Futurebus+ and how the I/O board interfaces to it. Next, a more detailed description of the Futurebus+ protocol controller and the control lines that interface it to Futurebus+ and the Memory Module is presented.

This application note implements a Futurebus+ I/O board design (see *Figure 1*) using National Semiconductor GAL®s (or PAL®s) to achieve a peak transfer rate of approximately 20 M-transfers/s (approximately 80 Mb/s using 32 data bits per memory bank, or 160 Mb/s using 64 data bits per memory bank) during compelled burst transfers. The Futurebus+ protocol controller supports the compelled mode of operation. The word compelled refers to the fact that after the master gives a request for a transfer the master is compelled to wait for a response from the slave before it can proceed to the next transfer. This request/acknowledge protocol is asynchronous and edge sensitive where both rising and falling edges are used to transfer information. This application note assumes that the reader is familiar with the DP8422A-25 DRAM controller, GAL/PAL design, and Futurebus+. This application note also builds on Application Note 668, *Futurebus+ Asynchronous Slave Memory Design,* by adding Futurebus+ mastership capabilities to that design.

1.1 Introduction to Futurebus+

Futurebus+ is a high-performance asynchronous 64-bit multiplexed address/data backplane bus designed by the IEEE 896.1 committee for use in a wide variety of multiprocessor architectures. The Futurebus+ standard is a next generation backplane bus standard developed to a set of requirements including openness, performance, and system facilities and flexibilities so as to not hinder systems using this bus for many generations of computer systems. Futurebus+ is a single cache coherent backplane architecture featuring scalability, technology independent protocols, and explicit provisions to extend to future applications.

2.0 INTRODUCTION TO THE FUTUREBUS+ I/O BOARD DESIGN

This Futurebus+ I/O board design consists of the following blocks:

1. Futurebus+ interface block,
2. Futurebus+ Reset and Initialization block,
3. CPU block,
4. I/O block,
5. Local Bus Arbiter block,
6. DRAM Interface block,
7. DRAM block,
8. DRAM, local and Futurebus+ interface transceiver block,
9. Futurebus+ Protocol and DMA controller block.

Figure 2 is a flowchart for the I/O board becoming the Futurebus+ Master. The slave flowchart (and all details of the slave portion of this board) will be found in Application Note 668, *Futurebus+ Asynchronous Slave Memory Design.*

2.1 Futurebus+ Interface Block

This block consists of the transceivers and arbitration controller that interface to Futurebus+. It contains the DS3886 9-bit BTL latched data transceivers, the DS3884 BTL Handshake transceivers, the DS3885 Arbitration transceivers, and the DS3875 Futurebus+ Arbitration Controller.

2.2 Futurebus+ Reset and Initialization Block

This block detects and initiates Futurebus+ initialization and system reset functions. These events are decoded and encoded within the time allocation period as given in the P896.1 Futurebus+ Logical Layer Specifications. Live Insertion and Live Withdrawal are not implemented in this design since these functions were not needed. However, if an application requires these functions, this design can be easily expanded to add the needed signals.

This block monitors the RE* signal on the Futurebus+ backplane. When RE* is asserted, a counter begins counting until RE* is released. While RE* is asserted, the bus initialize, reset system and Faulty RE* signals are evaluated. If system reset is detected, then RE* is driven until the board is reset (10 μs). When the module initiates a bus intialize or system reset, RE* is driven for the specified time, and until the on board bus initialization and reset functions are complete.

2.3 CPU Block

This block provides the local intelligence for the I/O board. The CPU responsibilities include:

— programming the DRAM controllers,
— programming the Futurebus+ Protocol Controller,
— programming the I/O device,
— programming the DS3875 Futurebus+ Arbitration Controller, and,
— all housekeeping chores associated with the I/O board.

A National Semiconductor NS32GX320 was chosen as the CPU. This is a 32-bit CPU that includes an on-board instruction and data cache. Read, Write, and 4 word Burst Read (with wrap around) accesses are supported to the DRAM.

2.4 I/O Block

The I/O block could be any I/O device, i.e., Ethernet interface, FDDI interface, Hard Disk interface . . . etc.

2.5 Local Bus Arbiter and Address Decode Block

This block provides arbitration for the local bus on the I/O board. Note that to access the local DRAM, one must gain access to the local bus. The local bus arbiter provides arbitration between the local bus and;

— the CPU,
— the I/O device, and,
— the Futurebus+ Protocol Controller.

The local bus is a separate synchronous address and data bus. The local bus arbiter accepts bus request signals from the above mentioned devices and issues bus grant signals. Only one bus grant signal is asserted at any given time.

Each requestor is expected to limit its own transactions across the bus to less than 15 μs. This allows the DRAM to be refreshed as required.

When no one is requesting the bus, the arbiter gives bus grant to the CPU. The BG_CPU* signal is asserted following power up, and when bus request signals from the I/O device and Futurebus+ are not asserted (Idle State). Any bus request signal from another device asserts HOLD* to the CPU. When HOLD* is asserted, all the CPU signals are TRI-STATE®.

The I/O device has been given a higher priority than the Futurebus+ bus request. Thus, when the two bus request signals (BR_IO* and BR_FB*) are simultaneously asserted, the arbiter will issue bus grant to the I/O first. Then, when BR_IO* is released, the arbiter will issue bus grant to Futurebus+.

The Address Decode section decodes the devices being addressed by generating the Chip Select (CS*) signals. The address space is allocated as follows:

00000000 to FEFFFFFF:	DRAM Memory
FF000000 to FF7FFFFF:	Arbitration Controller Registers
	DMA Protocol Controller Registers
	I/O Device Registers
FF8FFFFF to FFFFFFFF:	Used by CPU or unused

2.6 DRAM Interface Block

The DRAM Interface block provides an interface to the local DRAM memory for the following devices:

— CPU,
— I/O device, and,
— Futurebus+ Protocol and DMA Controller.

2.7 DRAM Block

For the purposes of this design a two bank DRAM system was designed. Each bank is controlled by a DP8421A DRAM Controller and is assumed to be 256k x 36 bits wide (32 bits data plus 4 parity bits). Therefore the total memory on the I/O board is 512k x 36 bits, or 2 MB. This configuration can be expanded up to 2M x 72 bits (16 MB) using the DP8421A DRAM Controller, or 8M x 72 bits (64 MB) using the DP8422A DRAM Controller.

This Futurebus+ memory module can achieve a 20 M-transfer/s rate during burst accesses. This performance is approximately 80 MB/s using 32 bits per memory bank and 160 MB/s using 64 bits per memory bank. This performance is calculated in Application Note #668, *Futurebus+ Asynchronous Slave Memory Design.*

2.8 DRAM, Local and Futurebus+ Interface Transceiver Block

There are two DRAM banks. Each DRAM bank has a separate interface to the local I/O bus and the Futurebus+ thru data transceivers.

The DRAM interface to the local I/O bus is thru 74F657 transceivers. These transceivers include parity generation and parity checking. Any data that is being written to the DRAM via the local bus has parity generated and stored along with the data. When data is read from the DRAM to the local bus the parity is checked for errors.

The DRAM interface to Futurebus+ is thru 74F543 transceivers. These transceivers include read and write latches. The latches in the transceivers allow the data being read or written to the DRAM to be pipelined.

During a DRAM read operation the two transceivers are used as a two word FIFO. The data access time to the transceiver based FIFO is 12 ns. Therefore the Futurebus+ sees a 12 ns memory when reading from the DRAM.

Likewise a DRAM write access is acknowledged once the data is latched into the transceiver write latch. The actual write access to the DRAM occurs later.

2.9 Futurebus+ Protocol and DMA Controller Block

The Futurebus+ Protocol Controller contains the logic necessary to implement a high performance GAL based Futurebus+ I/O board:

— Master and Slave capability,
— Compelled read and write transfers, partial, single and block (any length),
— address only transactions, and,
— Split transactions

The Futurebus+ Protocol Controller contains the following blocks:

— Asynchronous State Machine,
— Synchronous State Machine,
— DMA Local Address Register,
— DMA Futurebus+ Address Register and Parity Generator,
— DMA Transfer Size Counter,
— Slave Address and Data Command Latches and Parity Checkers, and
— Master Address and Data Command Latches and Parity Generator.

2.9.1 Asynchronous State Machine

The asynchronous state machine provides:

— high performance compelled Futurebus+ interface,
— interface to the DMA controller,
— interface to the DRAM controller.

The Futurebus+ interface includes:

— Futurebus+ arbitration controller (DS3875) interface,
— The output clock, input latch, and direction control of the Futurebus+ latched transceivers (DS3886), and,
— Futurebus+ Command, Capability, and Status interface.

The DMA controller interface includes enables for the Futurebus+ Address Register (FAR) and the Local Address Register (LAR). The "LAST_XFER" output of the Transfer Size Counter (TSC) is used to end the masters Futurebus+ transaction.

The DRAM controller interface controls the handshake signals between the DRAM Controller Module and the Futurebus+ Protocol and DMA Controller Module (DT_STB, D_ACK).

2.9.2 DMA Controller (Synchronous State Machine)

The Futurebus+ Protocol Controller contains a simple DMA controller used when the I/O board becomes the Futurebus+ master. It is used to transfer blocks of data between the local DRAM and the Futurebus+ system. The DMA controller consists of four blocks:

The LAR (local address register) contains the start address of the data block in DRAM. This can be either the source or destination of the transfer, depending on whether a bus write or read has been programmed. It is programmed from the data bus by the local CPU at the start of each transfer. It is implemented in four '573 latches.

The FAR (Futurebus address register) contains the start address of the data block on Futurebus. Once again, this is independent of the direction of data transfer. It is also programmed from the data bus by the local CPU at the start of each transfer. It is implemented in four '899 latched transceivers, which are also used to generate the parity for the Futurebus address. Because the address and data bus are multiplexed on Futurebus+, these transceivers can also be used to check the address and data parity on all slave transactions as well.

The TSC (transfer size counter) determines the number of words in the block DMA transfer. The 8-bit counter allows block sizes of up to 256 words (1 kB on a 32-bit bus) in a single transfer, subject to the limitations of slave. This is also programmed by the local CPU at the start of each transfer. Because it must be clocked asynchronously as the DMA transfer progresses on Futurebus, it could not be implemented in the same device as the rest of the DMA state machine. The CPU's DMA transfer size data is latched into a '573, which is loaded into a '269 counter by the DMA state machine at the start of each transaction. A '541 buffer allows the CPU to read the contents of the counter in the event of an error or unexpected DMA termination condition.

The DMA state machine is implemented as part of a MAPL128 PLD. Although only a few of the available product terms are used in this design, the additional room is available for implementing more sophisticated DMA control algorithms. This DMA controller waits for the CPU to signal, via an external I/O control register bit, that the DMA registers described above have been programmed and the transfer is ready to go. The synchronous DMA state machine requests first Futurebus+, then the local bus. Once both buses are available, it enables the addresses onto both the local and system buses and starts the asynchronous Futurebus+ master state machine. With each data transfer, the transfer counter is decremented. Once it reaches zero, the master state machine signals the DMA state machine. The DMA state machine checks to ensure that the transfer has completed successfully, and interrupts the CPU. The DMA controller is now ready to execute another transfer.

2.9.3 Slave Address and Data Command Latches and Parity Checkers

The slave address and data command latches hold the address and data command of the current Futurebus+ transaction.

2.9.4 Master Address and Data Command Latches and Parity Generator

The master address and data command latches hold the address and data command of the current Futurebus+ transaction when the I/O board is the current master of Futurebus+. These command latches are loaded by the on-board CPU before starting the Futurebus+ transaction.

3.0 FUTUREBUS+ SYSTEM TIMING CALCULATIONS

3.1 Maximum Time during a Master Read Access (Master reading data from slave) from the master generating DS to the master generating the next edge of DS is 89 ns. This is based upon the assumption that the DT_STB request/D_ACK acknowledge between the protocol controller and the memory module (of the I/O board) will be less than or equal to the speed of the Futurebus+ request/acknowledge handshake signals (DS, DK, and DI).

Parameter (Maximum Delays)	(In ns) Single Delay	Total Delay
Master's GAL Generates DS (assume 10 ns GAL)	10	10
Master's Handshake Transceiver, DS (DS3884)	7.5	17.5
Futurebus+ Delay	7.5	25
Slaves Handshake Transceiver, DS (DS3884)	7.5	32.5
Clock Out to Futurebus+ Next Piece of Data to Write to the Master		
Delay line (±2 ns), 10 ns Tap***	12	44.5
Slaves GAL Produces DK or DI (assume 10 ns GAL)	10	54.5
Slaves Handshake Transceiver, DK/DI (DS3884)	7.5	62
Futurebus+ Delay	7.5	69.5
Master's Handshake Transceiver, DK/DI (DS3884)	7.5	77
Delay Line (±2 ns), 10 ns Tap***	12	89
Master's GAL Generates DS (assume 10 ns GAL)	—	—

Typical Time during a Master Read Access (Master reading data from slave) from the master generating DS to the master generating the next edge of DS is 58 ns:

Parameter (Typical Delays)	(In ns) Single Delay	Total Delay
Master's GAL Generates DS (assume 10 ns GAL)	7	7
Master's Handshake Transceiver, DS (DS3884)	5	12
Futurebus+ Delay	2	14
Slaves Handshake Transceiver, DS (DS3884)	5	19
Clock Out to Futurebus+ Next Piece of Data to Write to the Master		
Delay Line (±2 ns), 10 ns Tap	10	29
Slaves GAL produces DK or DI (assume 10 ns GAL)	7	36
Slaves Handshake Transceiver, DK/DI (DS3884)	5	41
Futurebus+ Delay	2	43
Master's Handshake Transceiver, DK/DI (DS3884)	5	48
Delay Line (±2 ns), 10 ns Tap	10	58
Master's GAL generates DS (assume 10 ns GAL)	—	—

This gives a typical transfer rate of greater than 17 M-transfers/s during the Data Phase of the Futurebus+ Transfer.

***The 10 ns delay line used to compensate for possible skew between the Latched Transceiver (DS3886) and the Handshake Transceiver (DS3884).

3.2 Maximum time during a Master Read Access (Master reading data from slave) from the master generating DS to the master generating the next edge of DS is 89 ns. This is based upon the assumption that the DT__ STB request/ D__ACK acknowledge between the protocol controller and the memory module (of the I/O board) will be less than or equal to the speed of the Futurebus+ request/acknowledge handshake signals (DS, DK, and DI).

Parameter (Maximum Delays)	(In ns)	
	Single Delay	Total Delay
Master's GAL Generates DS (assume 10 ns GAL)	10	10
Master's Handshake Transceiver, DS (DS3884)	7.5	17.5
Futurebus+ Delay	7.5	25
Slaves Handshake Transceiver, DS (DS3884)	7.5	32.5
Delay Line (±2 ns), 10 ns Tap***	12	44.5
Slaves GAL Produces DK or DI (assume 10 ns GAL)	10	54.5
Slaves Handshake Transceiver, DK/DI (DS3884)	7.5	62
Futurebus+ Delay	7.5	69.5
Master's Handshake Transceiver, DK/DI (DS3884)	7.5	77
Clock Out to Futurebus+ Next Piece of Data to Write to the Slave		
Delay Line (±2 ns), 10 ns Tap***	12	89
Master's GAL Generates DS (assume 10 ns GAL)	—	—

Typical time during a Master Read Access (Master reading data from slave) from the master generating DS to the master generating the next edge of DS is 58 ns:

Parameter (Typical Delays)	(In ns)	
	Single Delay	Total Delay
Master's GAL Generates DS (assume 10 ns GAL)	7	7
Master's Handshake Transceiver, DS (DS3884)	5	12
Futurebus+ Delay	2	14
Slaves Handshake Transceiver, DS (DS3884)	5	19
Delay Line (±2 ns), 10 ns Tap***	10	29
Slaves GAL Produces DK or DI (assume 10 ns GAL)	7	36
Slaves Handshake Transceiver, DK/DI (DS3884)	5	41
Futurebus+ Delay	2	43
Master's Handshake Transceiver, DK/DI (DS3884)	5	48
Clock Out to Futurebus+ Next Piece of Data to Write to the Slave		
Delay Line (±2 ns), 10 ns Tap***	10	58
Master's GAL Generates DS (assume 10 ns GAL)	—	—

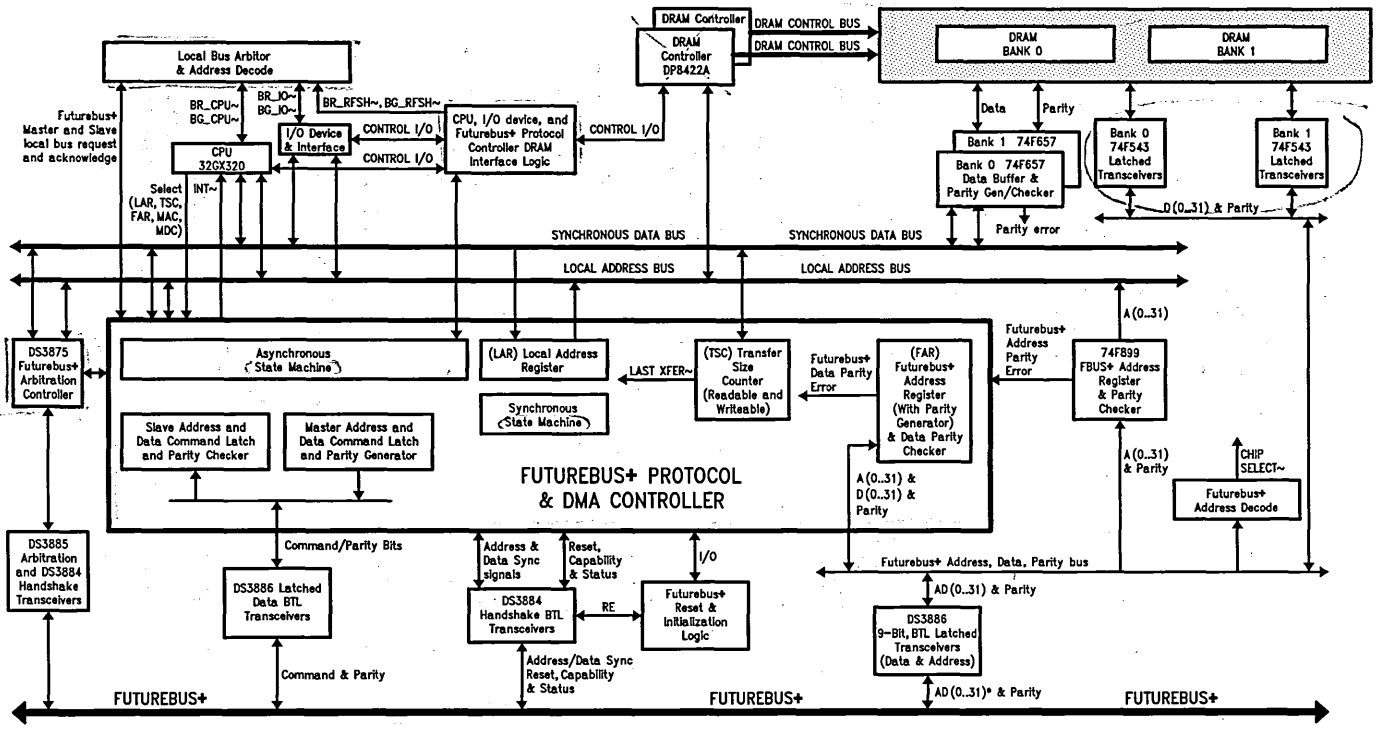
This gives a typical transfer rate of greater than 17 M-Transfers/s during the Data Phase of the Futurebus+ Transfer.

***The 10 ns delay line used to compensate for possible skew between the Latched Transceiver (DS3886) and the Handshake Transceiver (DS3884).

4.0 DESIGN CONSIDERATIONS OF THIS APPLICATION NOTE

4.1 CSR Space

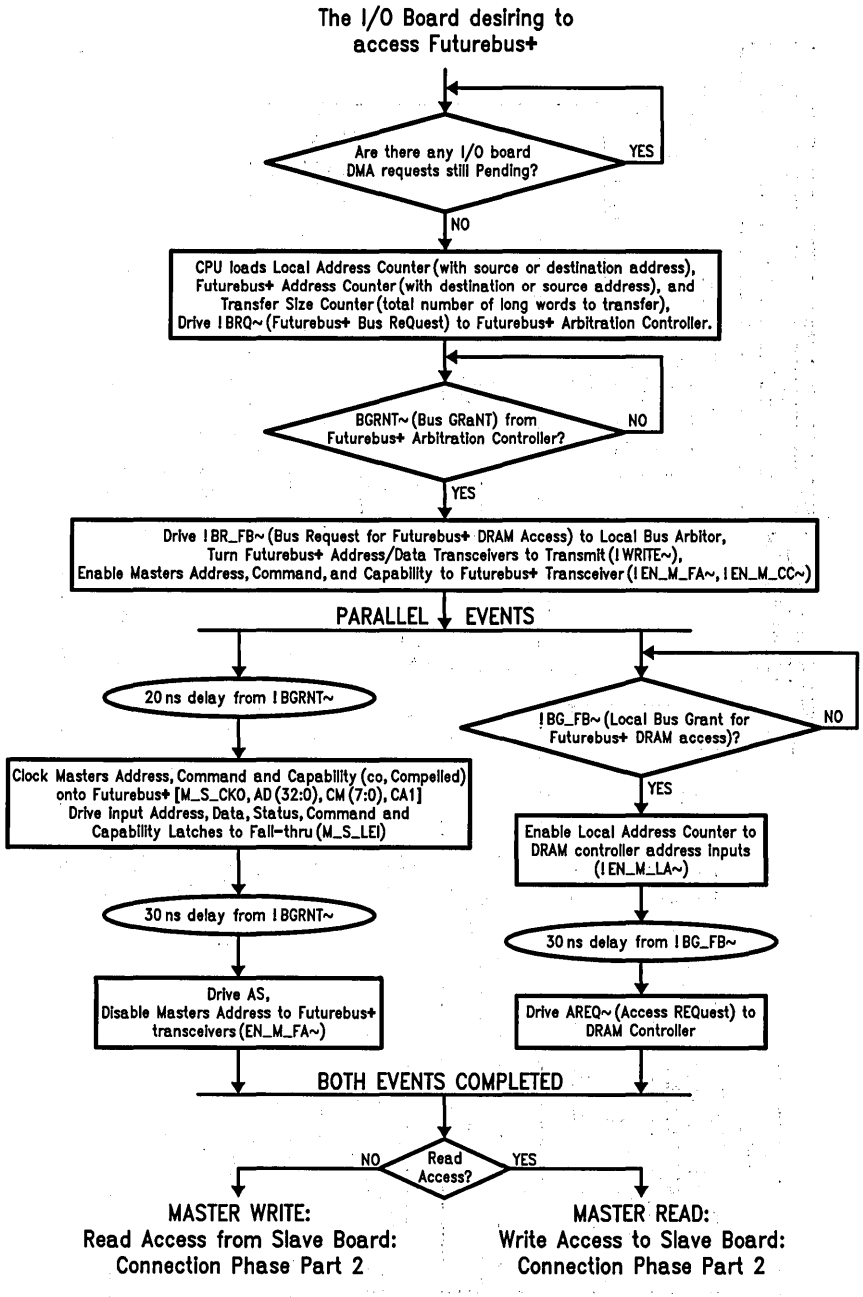
CSR registers are implemented in DRAM, more sophisticated implementations are possible.



TL/F/11155–1

FIGURE 1. Futurebus+ I/O Board

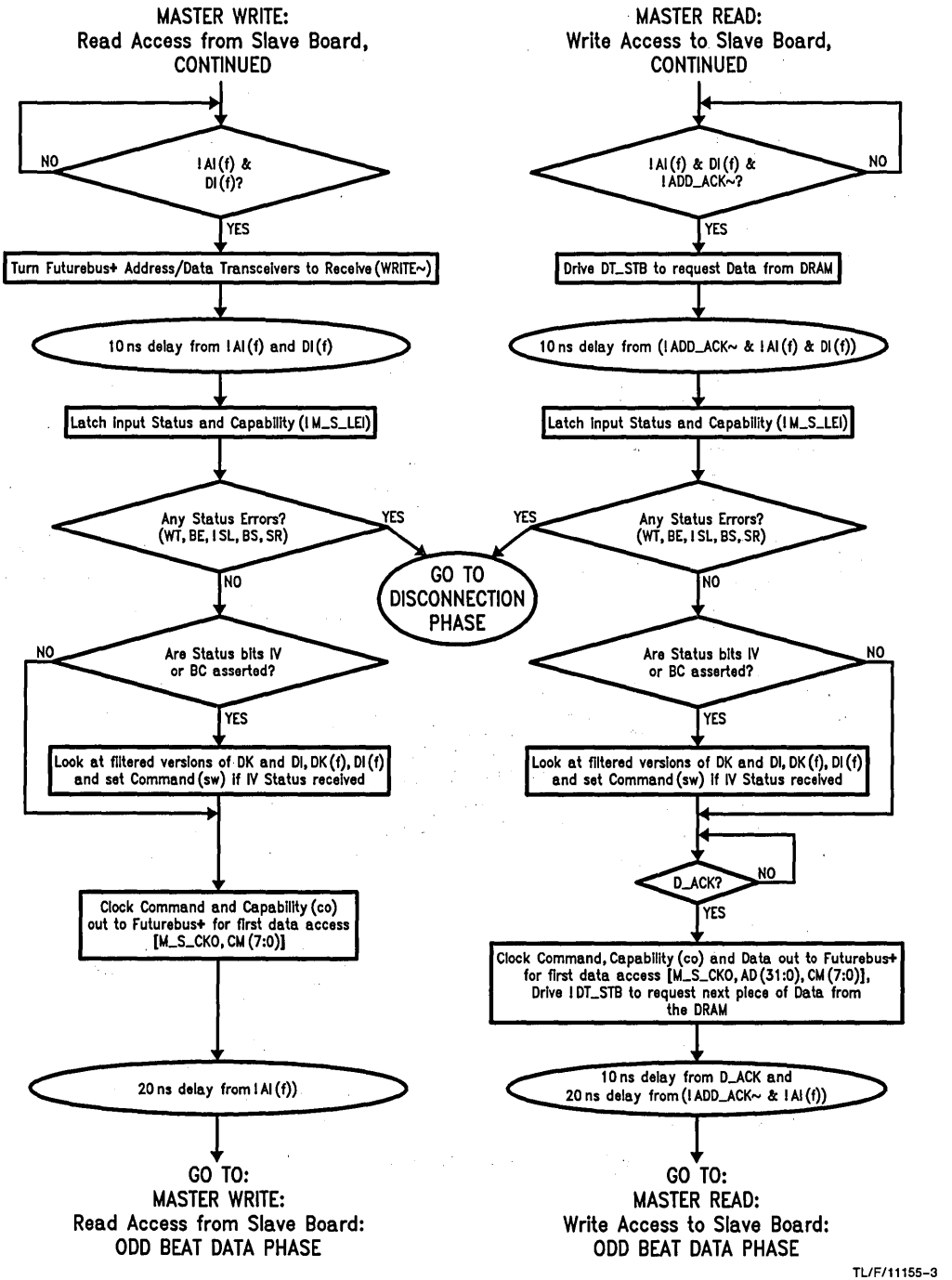
Date . . . November 5, 1990



TL/F/11155–2

FIGURE 2a. Master Connection Phase: Part 1, Flow Chart

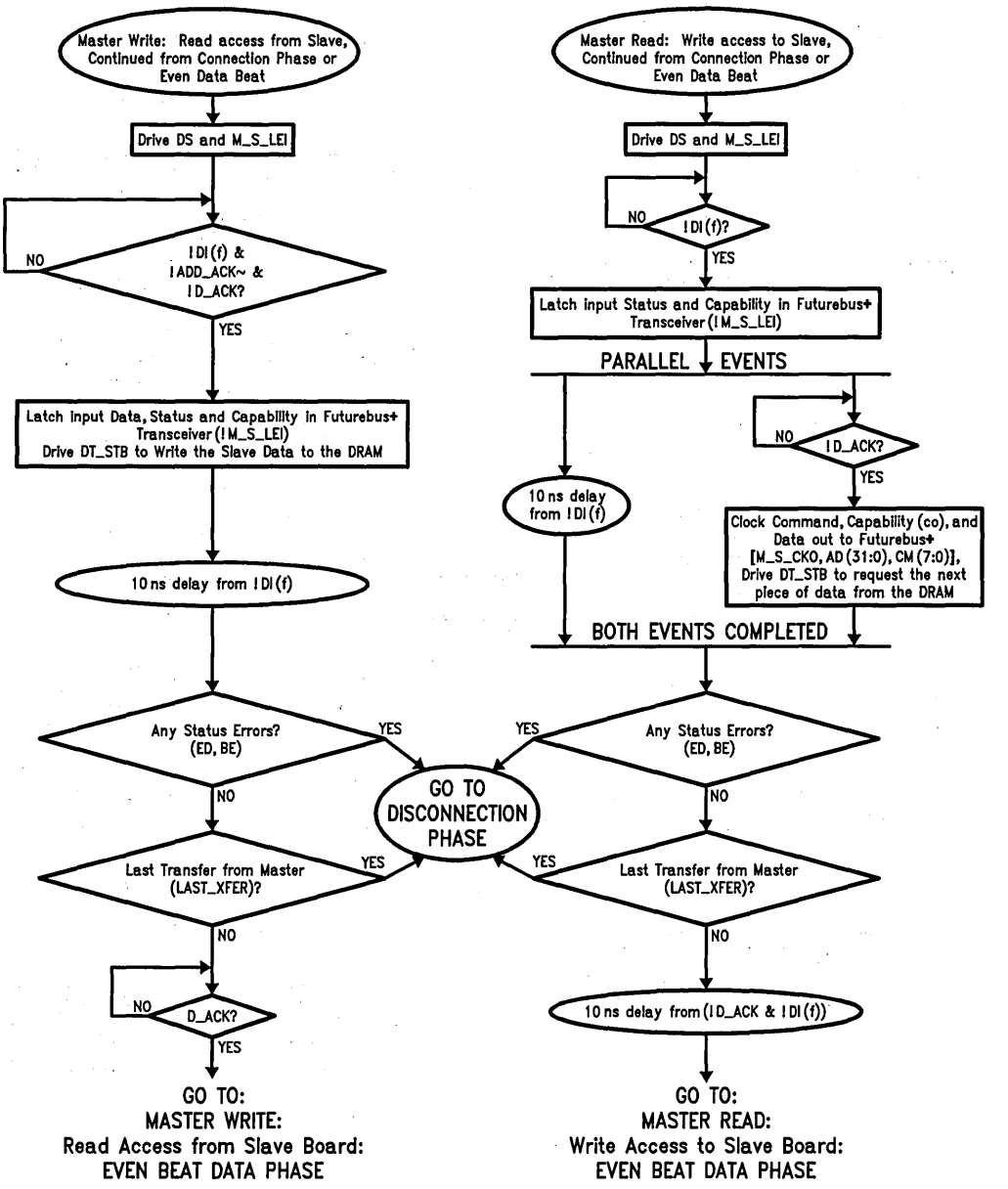
Date . . . September 25, 1990



TL/F/11155–3

FIGURE 2b. Master Connection Phase: Part 2, Flow Chart

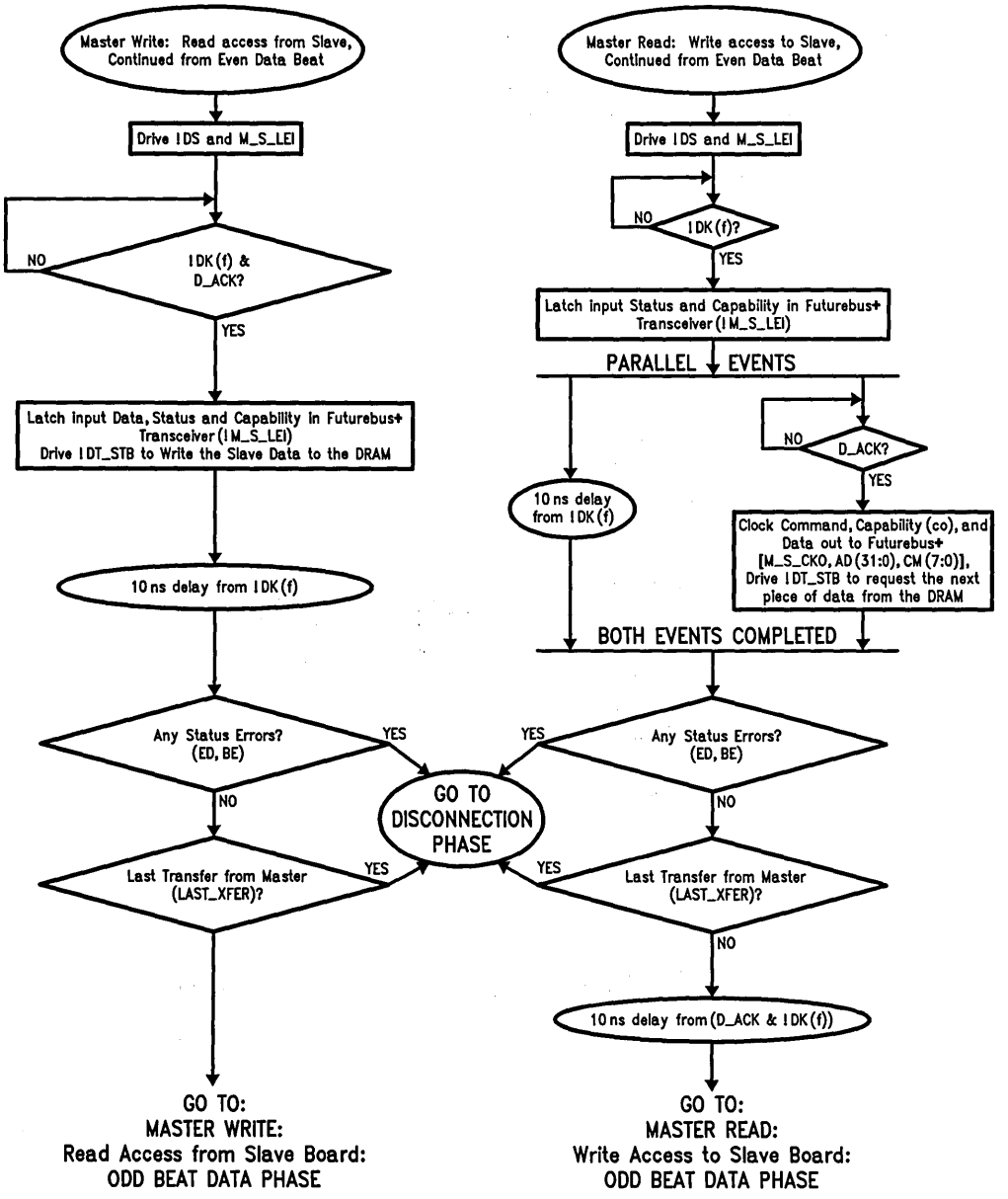
Date . . . December 12, 1990



TL/F/11155–4

FIGURE 2c. Master Odd Data Beat: Flow Chart

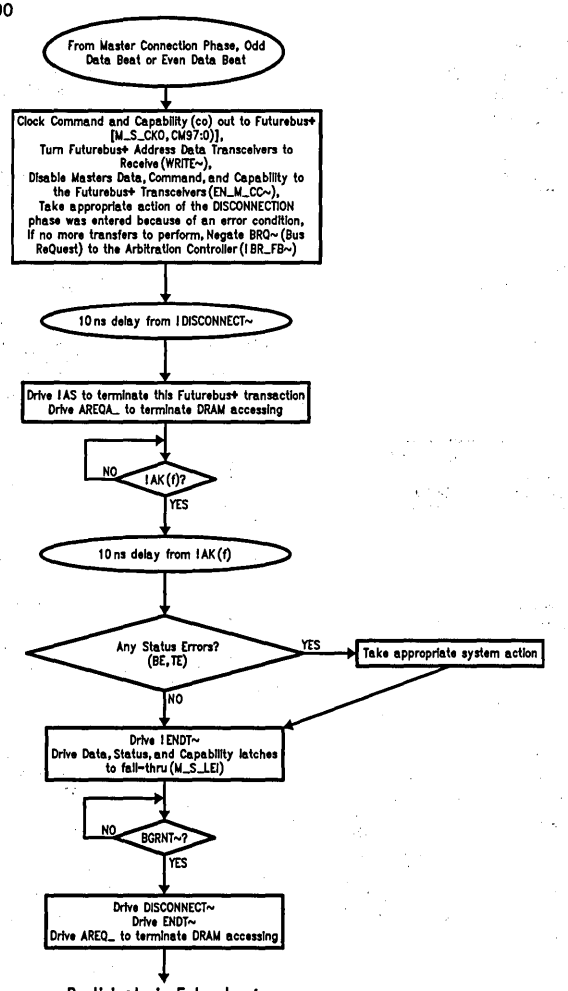
Date . . . December 12, 1990



TL/F/11155–5

FIGURE 2d. Master Even Data Beat: Flow Chart

Date . . . December 12, 1990



TL/F/11155–6

FIGURE 2e. Master Disconnection Phase: Flow Chart

Section 4

PI-Bus

Section 4 Contents

National Semiconductor

PI-Bus Overview

Dan Mansur

PI-Bus, Parallel Interface Bus, has evolved over the late 1980's to become the backplane of choice for avionics applications. National Semiconductor has utilized its BTL experience and supports the PI-Bus standard by offering a BiCMOS Octal PI-Bus transceiver.

PI-Bus Standard

The PI-Bus standard initially was developed under the Very High Speed Integrated Circuits (VHSIC) program, a program that funds development of advanced semiconductor technology for military use. In an effort to reduce redundant development of avionics subsystems, the U.S. Congress mandated that certain new aircraft programs will need to coordinate technology requirements. To this end, the Joint Integrated Avionics Working Group (JIAWG) was created of armed service and industry representatives to coordinate and standardize duplicative efforts.

The principle output of JIAWG regarding hardware was the creation of a common modular avionics architecture that could be configured to any aircraft. Known as the Advanced Avionics Architecture, it prescribes a bus oriented approach using various combinations of modules to achieve the different avionic needs. The current version of this architecture is known as Common Avionics Baseline III (CAB III) and is also incorporated in the Society of Automotive Engineers' (SAE) specifications. Development is also underway within Aeronautical Radio, Inc. (ARINC) to make PI-Bus the standard avionics backplane for commercial aircraft.

This architecture addresses those functions which could be implemented with common hardware and common computer programs to allow adaptation to either air-to-air or air-to-ground missions. The architecture features a building-block design, using standardized modules that can be plugged into a rack and linked by high speed data busses. Instead of line-replaceable units, the new approach features line-replaceable modules or, as they are now called, common modules.

Applications for the PI-Bus systems include flight control, communications, target acquisitions, weapon delivery, battlefield management, navigation, electronic countermeasures, stores management and radar management. Specific mandated programs include the U.S. Navy Advanced Tactical Aircraft (ATA), the U.S. Air Force Advanced Tactical Fighter (ATF), the U.S. Army Light Helicopter Experiment (LHX). Several existing airframes, such as the F-15 or the F-16, will be retrofitted. In addition, PI-Bus systems are expected to rapidly migrate into the commercial avionics as well.

PI-Bus Backplane Features

The PI-Bus backplane is a linear, multidrop synchronous bus with supports digital message communications between up to 32 modules residing in a single backplane. Messages are transferred datum serial and bit parallel using data size of 16 bits (single word) or 32 bits (double word).

PI-Bus standard uses a master-slave communications protocol which allows the current bus master to read data from one slave or write data to any number of slaves in a single message sequence. Messages may be routed to a particular module using either logical or physical addressing. A number of independent messages may be transmitted during a bus master's tenure. This message formats provide a 32-bit virtual addressing range for each module.

The PI-Bus protocol specifies a set of bus state transitions which control the bus to operate in a pipeline manner at the maximum clock rate allowed by the bus signal propagation delay. Master-Slave handshaking is provided with minimum performance penalty by operating the slave modules in synchronization with the master and using the bus state look-ahead.

National's PI-Bus Transceiver

National's octal PI-Bus transceiver was designed to meet the low power requirements of the military by combining the company's leading BTL (Backplane Transceiver Logic) with an advanced BiCMOS process.

BTL transceivers feature a nominal 1V signal swing for low power consumption, with receivers having precise thresholds for maximum noise immunity, and drivers with low power capacitance to minimize bus loading. These features combine to allow higher bus-data-transfer rates and improved overall system reliability. They also eliminate performance-degrating settling-time delays.

National's patented design and BiCMOS process enable the DS1776 to operate at approximately one-fourth the power of competing devices. The reduced power consumption is reflected in the worst-case current (I_{CC}) requirement of only 37 mA for the DS1776, compared with 145 mA for competing devices. This low power solution can offer up to 1 Amp per board savings, reducing the concerns of avionics manufacturers regarding excessive power consumption in the limited space available.

PI Bus

National Semiconductor
Application Note 725
Joe Wert

HISTORY

Throughout the 70's and early 80's the typical backplane was driven by standard TTL logic parts with tristateable outputs such as 54/74XX240 and 245. For design purposes these busses were modeled as lumped capacitances and transmission line effects were ignored because the bus data rates were generally not fast enough to require designers to be concerned with transmission line effects of the backplane. With the lower bus speeds there was sufficient bus settling time for reflections to dampen, and the signals to stabilize close enough to steady state to avoid problems. If a bus was heavily loaded and attempts were made to run the bus faster than 10 MHz, errors in data were frequently seen due to line reflections causing line voltage levels to transition through threshold more than once for a single transition of the buffer input. And as backplane densities increased, loading from the transceivers became so great that this increased the problem of driving the bus quickly and error free.

With today's higher bus data rates it has become imperative that the transmission line characteristics of the backplane be accounted for in backplane analysis. By doing so it becomes apparent that there may be better ways to drive backplanes than with traditional TTL family logic.

To solve some of the inherent problems with running densely loaded busses, the PI Bus structure was developed. PI Bus (parallel interface bus) was developed by IBM, Unisys and TRW for the VHSIC 2.2 Interoperability program. PI Bus has since been adopted by the Joint Integrated Avionics Working Group (JIAWG) section of SAE.

TRANSMISSION LINE PHYSICS

Several problems must be overcome to run a faster, more heavily loaded bus. When the bus propagation delay plus settling time decreases below about 100 ns the backplane must be considered as a transmission line. If the backplane is densely populated the problem of running it at high data rates is exacerbated.

Z_0 is the characteristic impedance of the backplane, and is calculated from the physical characteristics of the backplane. From *Figure 1*, a simple model of a transmission line, the following equation may be written:

$$Z_0 = [(R + jwl) / (G + jwc)]^{1/2}$$

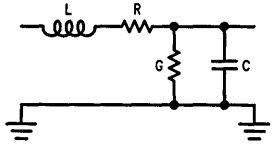
For high frequency, the R and G terms become insignificant due to the increases in the w terms,

$$w = 2\pi f \text{ where f is frequency}$$

and the equation simplifies to

$$Z_0 = (L/C)^{1/2}$$

where L is in units of inductance per unit length and C is in units of capacitance per unit length



TL/F/11071–1

FIGURE 1

EFFECTS OF CAPACITIVE LOADING ON BUS

Values for Z_0 can be calculated from the physics of the backplane construction. For a typical PI Bus epoxy-glass backplane the value for Z_0 is about 65Ω. The basic equation describing the impedance of a loaded bus is

$$Z_1 = Z_0 / (1 + C_1/C)^{1/2}$$

where C_1 is the additional load added to the bus by modules, connectors and connector mounting vias. So as the capacitive load of the modules increases, the impedance of the bus decreases. As the impedance of the bus decreases, the current required to drive the bus between state changes in a given time period increases (I = V/Z). The equation for the propagation delay of a signal on an unloaded transmission line is

$$T_{P0} = \ell (L/C)^{1/2}$$

where ℓ = length of the bus.

For a loaded bus,

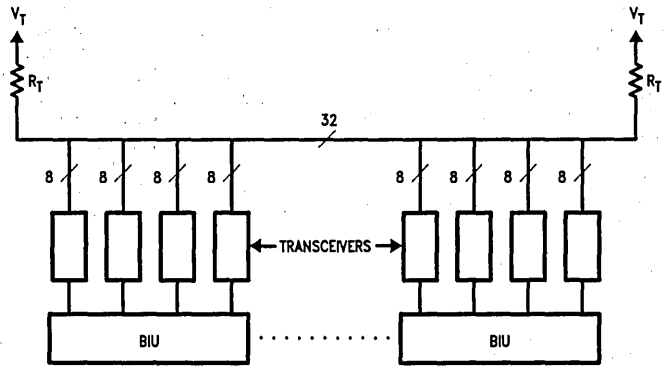
$$T_{P1} = T_{P0}/(1 + C_1/C)^{1/2}$$

describes the propagation delay on a transmission line. From the equation for the loaded bus, it is obvious that as the capacitive loading of the bus increases, the propagation delay for a given length of bus increases. As this time increases bus settling time increases, and hence affects how fast the bus may be operated. So a bus transceiver which had low capacitive loading would improve bus operation in several ways. A low capacitive transceiver would raise the value of Z_1, and hence the drive current requirements for a given performance would be reduced, the propagation delay for a given length of backplane would be reduced, and any bus settling time required would also be reduced.

OUTPUT V_{OH} SWING

The standard TTL swing for an output is from 0.2V to V_{CC} − two diodes (for CMOS 0V to V_{SS}). For a bipolar device, this swing could be as much as from 0.2V to 4.1V. The smaller this swing between high and low, the less charge which must be moved in any given time to effect a change of state. So from the standpoint of power usage, smaller in this case would be better.

PI Bus developed as a bus to address both these conclusions in a manner similar to Future Bus. The bus side outputs are open Schottkys, and the bus is terminated on both ends by a resistor to a voltage source with limits of 1.9V to 2.1V *(Figure 2)*. The resistors used to terminate the bus may be from 30Ω to 40Ω. By terminating the bus in this way, (Z_0 matches 40Ω termination with no loads on the bus, but including loading from raw backplane plus vias and mating connectors, and a 30Ω termination matches a fully loaded PI bus backplane) the bus is somewhat tuned to the Z_1 of the bus so that reflections can be minimized. By terminating the bus to a maximum of 2.1V, the maximum voltage swing on the bus will be from the V_{OL} minimum spec of 0.4V to the maximum 2.1V. Thus the voltage swing for the PI Bus transceiver would be 1.7V versus the 3.9V for a standard bipolar transceiver.



TL/F/11071-2

FIGURE 2

OUTPUT CAPACITANCE OF TRANSCEIVER

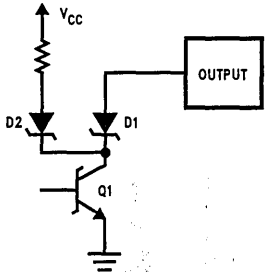
In order to decrease the capacitive loading of each transceiver, Schottky diode outputs are used, and are reverse biased when not active low. The PI Bus driver output is shown in *Figure 3a*. When the output is in a high state both the output Schottky diode and the collector-substrate diode of Q1 are reverse biased. By reverse biasing both the Schottky diode junction and the collector-substrate junction the parasitic capacitive loading of the bus driving output can be greatly decreased. The following equation gives an approximation of the impact reverse biasing has upon junction capacitance:

$$C_j = C_{j0}/(1 + V_d/V_0)^{1/2}$$

where C_{J0} equals the capacitance of the diode junction under unbiased conditions, and V_d equals the reverse biased voltage of the diode junction. V_0 is the built-in zero bias potential across a diode junction, and would be on the order of 0.7V. So, by reverse biasing the cathode of both D1 and Q1, their parasitic capacitive loading on the bus is decreased. If the bus were at 2.0V, and V_{CC} was at 5V, C_{J0} would be decreased by a factor of 2.1. Then of course, C1 and C2 are in series, so their total capacitance would be decreased according to the following equation:

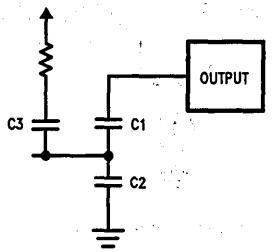
$$C_T = C_P + [(C_1)(C_2 + C_3)] \div (C_1 + C_2 + C_3)$$

The capacitance of C3 (from D3) is basically negligible in comparison to C1 because it is approximately 0.5% the area Q1. The package capacitance (C_P) may vary from 0.3 pF to 1.5 pF, depending upon the package and the pin location on the package. The other capacitive loading on each bus pin would be the base-collector and base-emitter



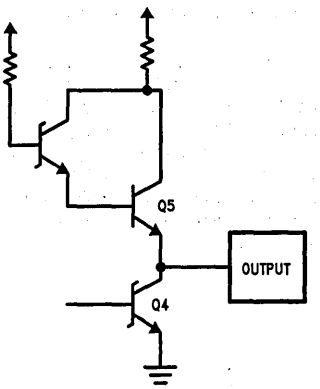
TL/F/11071-3

a. Simplified Circuit



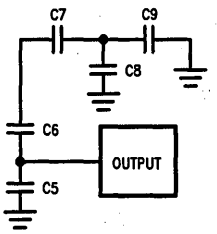
TL/F/11071-4

b. Parasitic Capacitance
FIGURE 3



TL/F/11071-5

a.



TL/F/11071-6

b.
FIGURE 4

capacitance of Q7 *(Figure 5b)*, which connects to the anode of D1. For a typical TTL part *(Figure 4)*, the capacitive loading of the output would be primarily C5 because C6 is in series with C7 and the parallel combination of C8 and C9. For Q4 and Q5 being about the same size devices, the capacitive loading of the TTL circuit would be several times larger than that of the Schottky terminated circuit.

PRECISION THRESHOLD

In order to handle the smaller output swing on the bus, it is necessary to use a more precise threshold circuit on the bus receiver input. *Figure 5a* is a typical TTL input buffer. The threshold set by this type of circuit varies considerably with temperature, and typically ranges from about 1V at high temperatures to about 1.8V at low temperatures. Such movement could not be tolerated with a bus swing which could be as small as from 1.15V to 1.9V. The circuit shown in *Figure 5b* is used on PI Bus transceivers to set a more precise threshold. A voltage reference with a very small V_{CC} and temperature dependence is placed on the chip to establish a precision threshold for the PI Bus to BIU input buffer. By using a differential pair with one of the pair controlled by the reference voltage, the threshold can be maintained within a 150 mV window centered at 1.52V.

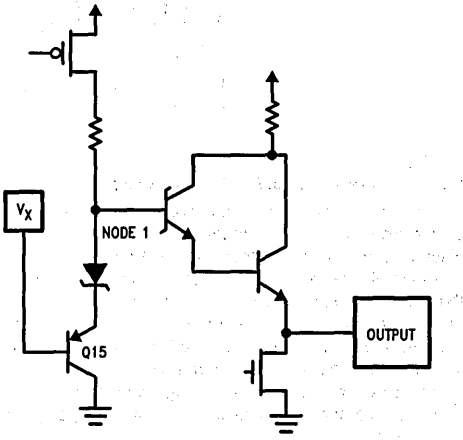
OUTPUT RAMPS AND NOISE IMMUNITY

Ramped outputs have been touted as the solution to problems of bus reflections and crosstalk. The amount of ramp time put into rise and fall times directly related to the propagation delays of a transceiver, so longer ramps require longer delay times. An important question to ask is how much of a ramp buys what degree of decreased bus settling time. Many TTL parts have peak ramps of about 2V/ns. This rate of ramp certainly seems to increase bus settling times. The PI Bus transceiver will have a typical ramp rate of about 0.5V/ns.

Having some amount of noise rejection on the bus receiver input allows the bus buffer input to ignore small excursions above or below threshold without affecting the data being transmitted to the BIU. However the greater the amount of noise immunity, the greater the propagation delay on the path from the bus to the BIU. The PI Bus transceiver offers a compromise between noise immunity and prop delays with typically 4 ns of pulsewidth protection measured at 1.5V for a 1 to 2 volt input.

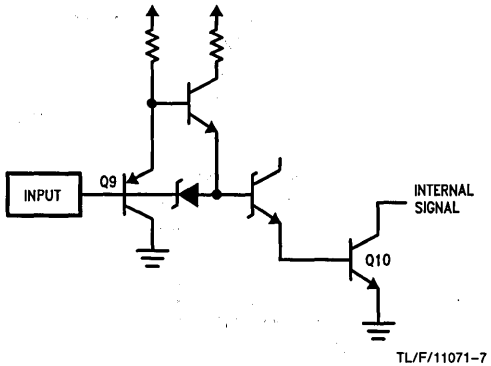
SUBMICRON BIU PROTECTION

To accommodate future submicron BIU processing, the PI Bus transceivers offer a feature for limiting the output V_{OH} excursion to the BIU. The V_X pin on the chip can be used as a clamp voltage to limit the V_{OH} of the BIU side output. The basic propagation delays on the transceiver with the exception of the BIU side low to high ramp rate are unaffected because the remainder of the transceiver is still powered by the normal V_{CC} pin. Only the BIU V_{OH} is controlled by the V_X pin. The *Figure 6* schematic shows how the V_X voltage sets the output V_{OH}.



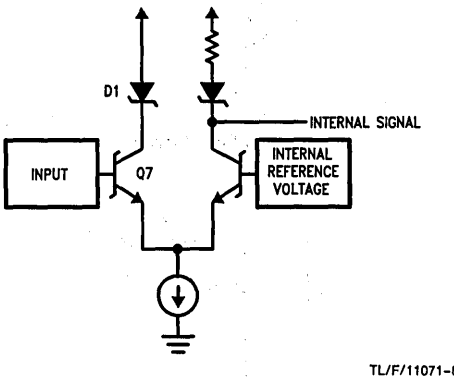
TL/F/11071–9

FIGURE 6



TL/F/11071–7

a. Typical TTL Input



TL/F/11071–8

b. PI Bus Receiver Input

FIGURE 5

LOW POWER

Many applications of PI Bus are designed with a redundant bus, and as such always have a full set of transceivers in the inactive mode. And in addition, on the active bus there may be 15 inactive (for a 16 drop bus) transceivers with one active transceiver. So in order to lower the power requirements for the PI Bus backplane application the National PI Bus transceiver, the DS1776, was designed using BiCMOS in order to take advantage of the power savings possible with the use of CMOS in appropriate portions of the circuit.

A pure bipolar transceiver would have an I_{CC} inactive specification of about 100 mA. By using a BiCMOS process it was possible to design a PI Bus transceiver with an I_{CC} inactive of 35 mA.

BIPOLAR vs CMOS vs BICMOS TECHNOLOGY

A full bipolar transceiver could be built to provide excellent performance in most areas with the exception of power consumption in both the active and inactive modes. A pure CMOS part could be designed with a much lower I_{CC}, but it would have the following drawbacks. It would be slower than a bipolar part, no Schottky diode would be available to construct a low capacitance output with limited swing, and stable CMOS voltage references which have a reasonable silicon area are non-existent, so setting a precise input voltage threshold would be difficult.

By making judicious use of bipolar and CMOS circuits where each is appropriate, it was possible to design a PI Bus transceiver with an I_{CC} inactive of typically 35 mA and yet maintain the full features which make PI Bus a clean and quiet bus to run.

REFERENCES

Chapman, Robert A. *Transmission Lines.* New York: McGraw-Hill, 1968.

Muller, Richard S., Theodore I. Kamins. *Device Electronics for Integrated Circuits.* New York: John Wiley and Sons, 1977.

Neff, Herbert P. *Basic Electromagnetic Fields.* New York: Harper and Row, 1981.

JIAWG PI-Bus Specification—J89-N1A Draft

Balakrishnan, R.V. "Eliminating Crosstalk Over Long-Distance Busing." *Computer Design,* March 1982, pp. 155–162.

Balakrishnan, R.V. "DS3662—The Bus Optimizer." National Semiconductor—Application Note 337 *Interface Databook* 1988.

"P1194.0/d2—A Guide to Backplane Electrical Performance Measurements." June 1989—Prepared by BTL Working Group from I.E.E.E.

National Semiconductor

PRELIMINARY

DS1776 PI-Bus Transceiver

General Description

The DS1776 is an octal PI-bus Transceiver. The A to B path is latched. B outputs are open collector with series Schottky diode, ensuring minimum B output loading. B outputs also have ramped rise and fall times (2.5 ns typical), ensuring minimum PI-bus ringing. B inputs have glitch rejection circuitry, 4 ns typical.

Designed using National's Bi-CMOS process for both low operating and disabled power. AC performance is optimized for the PI-Bus inter-operability requirements.

The DS1776 is an octal latched transceiver and is intended to provide the electrical interface to a high performance wired-or bus. This bus has a loaded characteristic impedance range of 20Ω to 50Ω and is terminated on each end with a 30Ω to 40Ω resistor.

The DS1776 is an octal bidirectional transceiver with open collector B and TRI-STATE® A port output drivers. A latch function is provided for the A port signals. The B port output driver is designed to sink 100 mA from 2V and features a controlled linear ramp to minimize crosstalk and ringing on the bus.

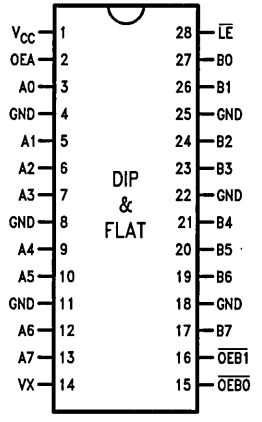
A separate high level control voltage (V_X) is provided to prevent the A side output high level from exceeding future high density processor supply voltage levels. For 5V systems, V_X is tied to V_{CC}.

Features

- Mil-Std-883C qualified
- Low power I_{CCL} = 37 mA max
- B output controlled ramp rate
- B input noise immunity, typically 4 ns
- Available in 28-pin DIP, Flatpak and CLCC
- Pin and function compatible with Signetics 54F776

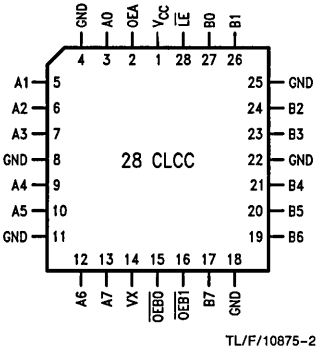
Pin Configurations

Pin Configuration



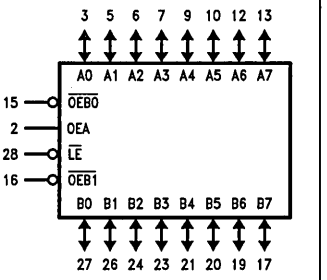
TL/F/10875–1

Pin Configuration CLCC



TL/F/10875–2

Logic Symbol



TL/F/10875–3

Order Number DS1776E/883, DS1776J/883 or DS1776W/883
See NS Package E28A, J28B or W28B

DEVICE SPECIFICATIONS

Absolute Maximum Ratings (Notes 1 and 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +7.0V
V_X, V_{OH} Output Level Control Voltage (A Outputs)	−0.5V to +7.0V
$\overline{OEB}$n, OEA, $\overline{LE}$ Input Voltage (V_I)	−0.5V to +7.0V
A0–A7, B0–B7 Input Voltage (V_I)	−0.5V to +5.5V
Input Current (I_I)	−40 mA to +5 mA
Voltage Applied to Output in High Output State (V_O)	−0.5V + V_{CC}V
A0–A7 Current Applied to Output in High Output State (I_O)	40 mA
B0–B7 Current Applied to Output in High Output State (I_O)	200 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Lead Temperature (Soldering 10 Sec.)	260°C
ESD Tolerance: C_{ZAP} = 120 pF, R_{ZAP} = 1500Ω	2.0 kV

Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
Operating Temp. Range (T_A)	−55	+125	°C
Input Rise or Fall Times (t_r, t_f)		50	ns

DC Electrical Characteristics V_{CC} = 5V ±10% (Unless Otherwise Specified)

Symbol	Parameter	Conditions (Note 3)	Guaranteed Limits −55°C to +125°C			Units
			Min	Typ (Note 4)	Max	
V_{IH}	High Level Input Voltage $\overline{OEB}$0, 1, OEA, $\overline{LE}$, An Bn		 2.0 1.6			 V V
V_{IL}	Low Level Input Voltage Except B0–B7 B0–B7 (Note 5)				 0.8 1.45	 V V
I_{OH}	High Level Output Current A0–A7	V_{IN} = V_{IH} or V_{IL} V_{OH} = V_{CC} − 2.0V			−3	mA
I_{OH}	High Level Output Current B0–B7	V_{CC} = Max, V_{IL} = 0.8V, V_{IH} = 2.0V, V_{OH} = 2.1V			100	μA
I_{OL}	Low Level Output Current A0–A7 B0–B7	V_{IN} = V_{IH} or V_{IL} V_{OL} = 0.5V V_{OL} = 1.15V			 20 100	 mA mA
I_{IK}	Input Clamp Current Except A0–A7 A0–A7				 −18 −40	 mA mA
I_{OZ}	TRI-STATE Output Leakage Current A0–A7 B0–B7				±70	μA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: For conditions shown as Min or Max, use the appropriate value specified under recommended operating conditions for the applicable type and function table for operating mode. Unless otherwise specified, V_X = V_{CC} for all test conditions.

Note 4: All typical values are at V_{CC} = 5V, T_A = 25°C.

Note 5: Due to test equipment limitations, actual test conditions are for V_{IH} = 1.9V and for V_{IL} = 1.2V, however the specified test limits and conditions are guaranteed.

DC Electrical Characteristics V_{CC} = 5V ±10% (Unless Otherwise Specified) (Continued)

Symbol	Parameter		Conditions		Guaranteed Limits −55°C to +125°C			Units
					Min	Typ	Max	
V_{OH}	High Level Output Voltage	A0–A7	V_{CC} = Min, V_{IL} = Max, V_{IH} = Min	I_{OH} = −3 mA, V_X = V_{CC}	2.5	2.9	V_{CC}	V
				I_{OH} = −0.4 mA, V_X = 3.13V & 3.47V	2.5		V_X	V
V_{OL}	Low Level Output Voltage	A0–A7	V_{CC} = Min, V_{IL} = Max, V_{IH} = Min	I_{OL} = 20 mA, V_X = V_{CC}		0.3	0.5	V
		B0–B7	V_{CC} = Min, V_{IL} = Max, V_{IH} = Min	I_{OL} = 100 mA I_{OL} = 4 mA	0.40		1.15	V V
V_{IK}	Input Clamp Voltage	A0–A7	V_{CC} = Min, I_I = −40 mA				−0.5	V
		Except A0–A7	V_{CC} = Min, I_I = −18 mA				−1.2	V
I_{IH2}	Input Current at Max Input Voltage	$\overline{OEB}$n, OEA, $\overline{LE}$	V_{CC} = Max, V_I = 7.0V			1	100	μA
		A0–A7	V_{CC} = Max, V_I = 5.5V			0.01	1	mA
		B0–B7	V_{CC} = Max, V_I = 5.5V			0.01	1	mA
I_{IH1}	High Level Input Current	$\overline{OEB}$n, OEA, $\overline{LE}$	V_{CC} = Max, V_I = 2.7V				20	μA
		B0–B7	V_{CC} = Max, V_I = 2.1V				100	μA
I_L	Low Level Input Current	$\overline{OEB}$n, OEA, $\overline{LE}$	V_{CC} = Max, V_I = 0.5V				−20	μA
		B0–B7	V_{CC} = Max, V_I = 0.3V				−100	μA
I_{OZH} + I_{IH}	TRI-STATE Output Current, High Level Voltage Applied	A0–A7	V_{CC} = Max, V_O = 2.7V				70	μA
I_{OZL} + I_{IL}	TRI-STATE Output Current, Low Level Voltage Applied	A0–A7	V_{CC} = Max, V_O = 0.5V				−70	μA
I_X	High Level Control Current		V_{CC} = Max, V_X = V_{CC}, $\overline{LE}$ = OEA = $\overline{OEB}$n = 2.7V, A0–A7 = 2.7V, B0–B7 = 2.0V		−100		100	μA
			V_{CC} = Max, V_X = 3.13V & 3.47V, $\overline{LE}$ = OEA = 2.7V, $\overline{OEB}$n = A0–A7 = 2.7V, B0–B7 = 2.0V		−10		10	mA
I_{OS}	Short-Circuit Output Current (Note 6)	A0–A7 Only	V_{CC} = Max, Bn = 1.6V, OEA = 2.0V, $\overline{OEB}$n = 2.7V		−60	−75	−150	mA
I_{CC}	Supply Current (Total)	I_{CCH}	V_{CC} = Max (Note 7)				37	mA
		I_{CCL}	V_{CC} = Max, V_{II} = 0.5V				32	mA
		I_{CCZ}	V_{CC} = Max, V_{II} = 0.5V				35	mA
I_{OFF}	Power Off Output Current	B0–B7	Bn = 2.1V, V_{CC} = 0.0V, V_{IL} = Max, V_{IH} = Min				100	μA

Note 6: Not more than one output should be shorted at a time. For testing I_{OS}, the use of high speed test apparatus and/or sample-and-hold techniques are preferable in order to minimize internal heating and more accurately reflect operational values. Otherwise, prolonged shorting of a High output may raise the chip temperature well above the normal and thereby cause invalid readings in other parameter tests. In any squence of parameter test, I_{OS} tests should be performed last.

Note 7: 8 inputs @ V_{CC}.

AC Electrical Characteristics

Path	Parameter	Test Conditions	Min	Typ	Max	Units
B TO A PATH						
t_{PLH}	Propagation Delay B to A	Waveform 1, 2	4.5		16	ns
t_{PHL}			6.0		17	ns
t_{PZH}	Output Enable OEA to A	Waveform 3, 4	4.0		20	ns
t_{PZL}			4.0		21	ns
t_{PHZ}	Output Disable OEA to A	Waveform 3, 4	2.0		12	ns
t_{PLZ}			2.0		13	ns
A TO B PATH						
t_{PLH}	Propagation Delay A to B	Waveform 1, 2	2.0		13	ns
t_{PHL}			2.5		13	ns
t_{PLH}	Propagation Delay LE to B	Waveform 1, 2	2.0		16	ns
t_{PHL}			2.0		16	ns
t_{PLH}	Enable/Disable OEBn to B	Waveform 1, 2	2.0		13	ns
t_{PHL}			3.5		13	ns
t_{TLH}	Transition B Side 1.2V to 1.8V, 1.8V to 1.2V			2		ns
t_{THL}				2		ns
SETUP/HOLD/PULSE WIDTH SPECS						
t_S	A to LE Setup	Waveform 5	5.0			ns
t_H	A to LE Hold	Waveform 5	0.0			ns
t_W	LE Pulse Width Low	Waveform 5	12			ns

Description

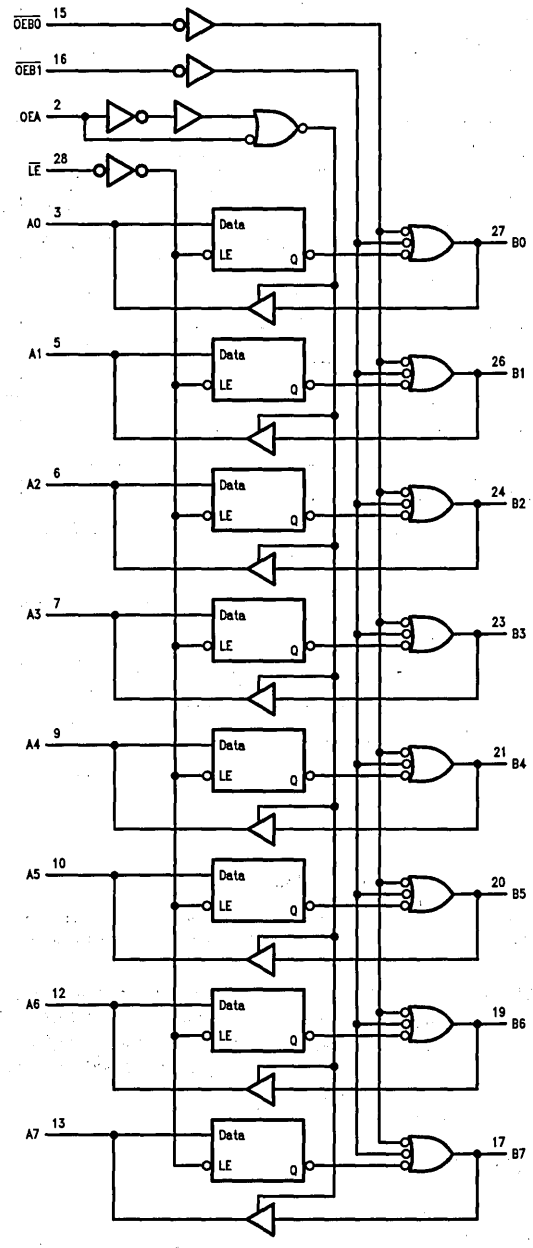
PIN DESCRIPTION

TABLE I. Pin Description

Symbol	Pins	Type	Name and Function
A0	3	I/O	PNP latched input/TRI-STATE output (with V_X control option)
A1	5	I/O	
A2	6	I/O	
A3	7	I/O	
A4	9	I/O	
A5	10	I/O	
A6	12	I/O	
A7	13	I/O	
B0	27	I/O	Data input with special threshold circuitry to reject noise/Open Collector output, High current drive
B1	26	I/O	
B2	24	I/O	
B3	23	I/O	
B4	21	I/O	
B5	20	I/O	
B6	19	I/O	
B7	17	I/O	
$\overline{OEB}0$	15	I	Enables the B outputs when both pins are low
$\overline{OEB}1$	16	I	
OEA	2	I	Enables the A outputs when High
$\overline{LE}$	28	I	Latched when High (a special delay feature is built in for proper enabling times)
V_X	14	I	Clamping voltage keeping V_{OH} from rising above V_X ($V_X = V_{CC}$ for normal use)

Description (Continued)

FUNCTION DESCRIPTION



TL/F/10875–4

V_{CC} = Pin 1
V_X = Pin 14
GND = Pins 4, 8, 11, 18, 22, 25

FIGURE 1. Functional Logic Diagram

4

Description (Continued)

TABLE II. Function Table

Inputs						Latch State	Outputs		Mode
An	Bn (Note 3)	$\overline{LE}$	OEA	$\overline{OEB0}$	$\overline{OEB1}$		An	Bn	
H	X	L	L	L	L	H	Z	H	A TRI-STATE, Data from A to B
L	X	L	L	L	L	L	Z	L	
X	X	H	L	L	L	Qn	Z	Qn	A TRI-STATE, Latched Data to B
—	—	L	H	L	L	(Note 1)	(Note 1)	(Note 1)	Feedback: A to B, B to A
—	H	H	H	L	L	H (Note 2)	H	off (Note 2)	Preconditioned Latch Enabling Data Transfer from B to A
—	L	H	H	L	L	H (Note 2)	L	off (Note 2)	
—	—	H	H	L	L	Qn	Qn	Qn	Latch State to A and B
H	X	L	L	H	X	H	Z	off	B off and A TRI-STATE
L	X	L	L	H	X	L	Z	off	
X	X	H	L	H	X	Qn	Z	off	
—	H	L	H	H	X	H	H	off	B off, Data from B to A
—	L	L	H	H	X	L	L	off	
—	H	H	H	H	X	Qn	H	off	
—	L	H	H	H	X	Qn	L	off	
H	X	L	L	X	H	H	Z	off	B off and A TRI-STATE
L	X	L	L	X	H	L	Z	off	
X	X	H	L	X	H	Qn	Z	off	
—	H	L	H	X	H	H	H	off	B off, Data from B to A
—	L	L	H	X	H	L	L	off	
—	H	H	H	X	H	Qn	H	off	
—	L	H	H	X	H	Qn	L	off	

H = High Voltage Level
L = Low Voltage Level
X = Don't Care
— = Input not externally driven
Z = High Impedance (off) state
Qn = High or Low voltage level one setup time prior to the Low-to-High $\overline{LE}$ transition
Note 1: Condition will cause a feedback loop path; A to B and B to A.
Note 2: The latch must be preconditioned such that B inputs may assume a High or Low level while $\overline{OEB}0$ and $\overline{OEB}1$, are Low and $\overline{LE}$ is high.
Note 3: Precaution should be taken to ensure that the B inputs do not float. If they do, they are equal to a Low state.
off = Applies to "B" (OC) outputs only. Indicates that the outputs are turned off.

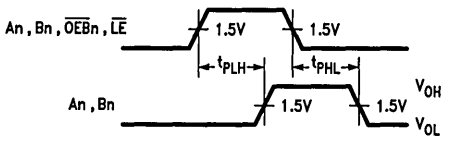
CONTROLLER POWER SEQUENCING OPERATION

The DS1776 has a design feature which controls the output transitions during power up (or down). There are two possible conditions that occur.

1. When $\overline{LE}$ = Low and $\overline{OEB}$n = Low, the B outputs are disabled until the $\overline{LE}$ circuit can take control. This feature ensures that the B outputs will follow the A inputs and allow only one transition during power up (or down).
2. If $\overline{LE}$ = High or $\overline{OEB}$n = High, then the B outputs still remain disabled during power up (or down).

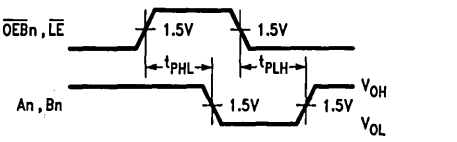
Switching Characteristics

AC WAVEFORMS



TL/F/10875–5

Waveform 1: Propagation Delay for Data to Output

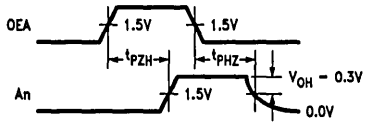


TL/F/10875–6

Waveform 2: Propagation Delay for Data to Output

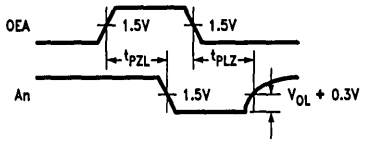
Switching Characteristics (Continued)

AC WAVEFORMS (Continued)



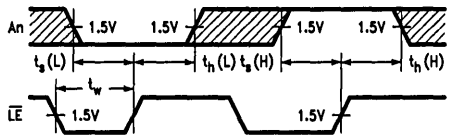
TL/F/10875–7

Waveform 3: TRI-STATE Output Enable Time to High Level and Output Disable Time from High Level



TL/F/10875–8

Waveform 4: TRI-STATE Output Enable Time to Low Level and Output Disable Time from Low Level



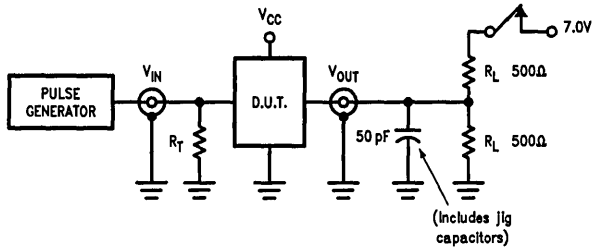
TL/F/10875–9

Waveform 5: Data Setup and Hold Times and LE Pulse Widths

The shaded areas indicate when the input is permitted to change for predictable output performance.

TEST CIRCUIT AND WAVEFORMS

Test Circuit for TRI-STATE Outputs on A Side

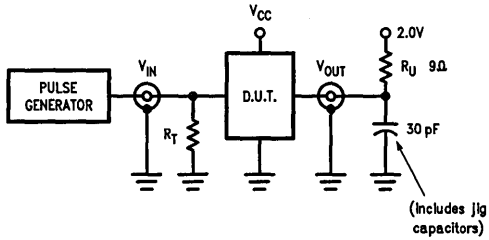


TL/F/10875–10

Switch Position

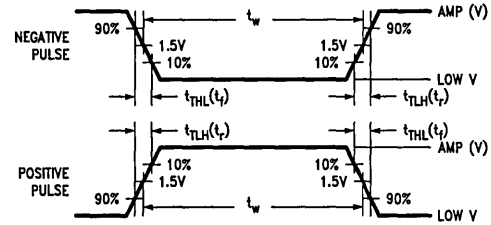
Test	Switch
t_{PLZ}, t_{PZL}	Closed
All Other	Open

Test Circuit for TRI-STATE Outputs on B Side



TL/F/10875–12

Input Pulse Definition

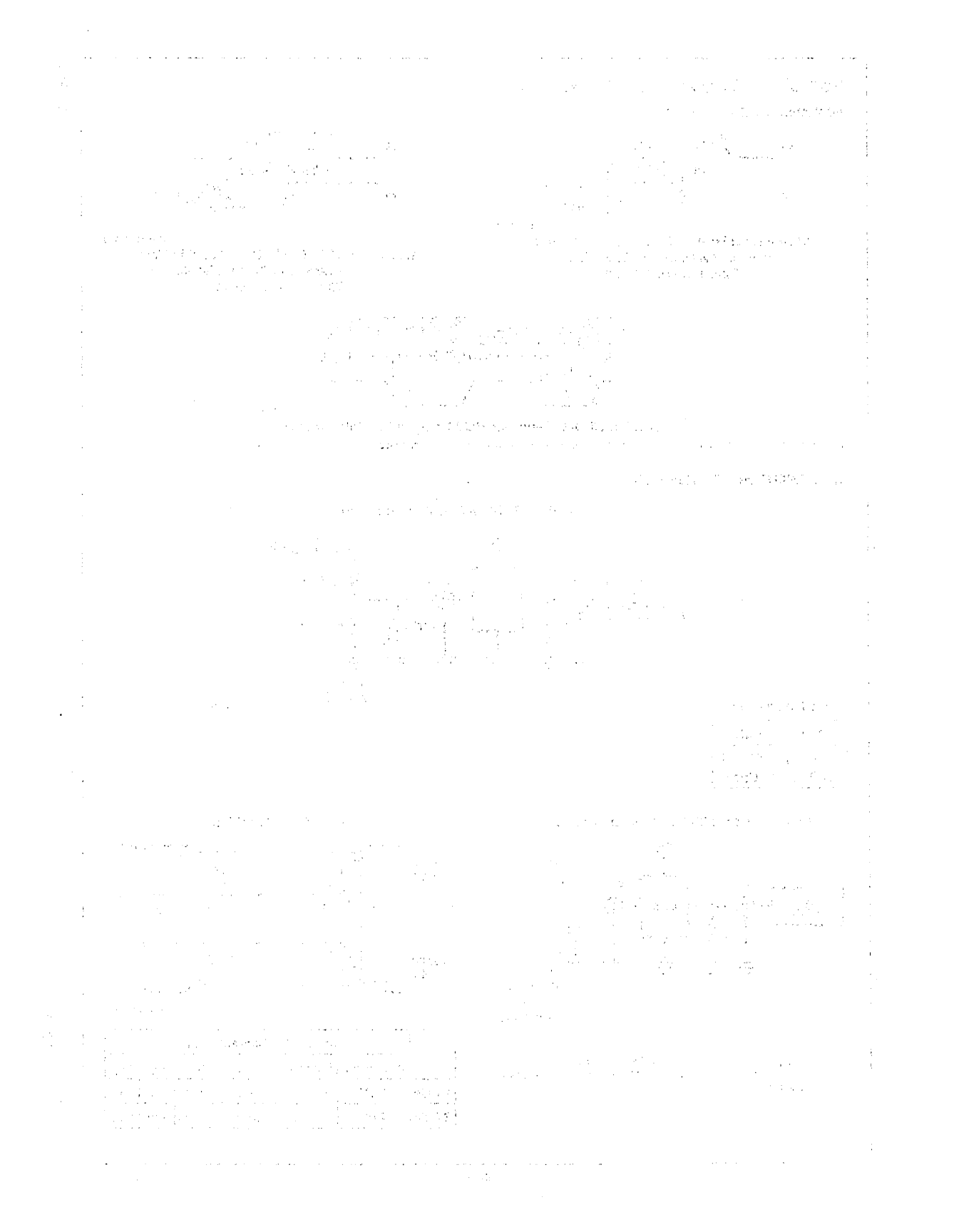


TL/F/10875–11

DEFINITIONS

R_L = Load resistor 500Ω
C_L = Load capacitance includes jig and probe capacitance
R_T = Termination resistance should be equal to Z_{OUT} of pulse generators.
R_U = Pull up resistor

	Input Pulse Characteristics					
	Amplitude	Low V	Rep. Rate	t_W	t_{TLH}	t_{THL}
A Side	3.0V	0.0V	1 MHz	500 ns	2 ns	2 ns
B Side	2.0V	1.0V	1 MHz	500 ns	2 ns	2 ns



Section 5

Futurebus + / BTL Reference

Section 5 Contents

National Semiconductor

Futurebus+/BTL Reference

Futurebus+/BTL Modeling Support

National Semiconductor, in an effort to assist system designers with board level design and simulation, has made available a series of spice and behavioral level models for its Futurebus+/BTL devices.

SPICE MODELING SUPPORT

BTL Spice Modeling Manual
BTL/Futurebus+ Spice Models
BTL Turbo Transceiver Spice Models
BTL Trapezoidal Transceiver Spice Models

Behavioral Model Support: (All are Verilog Models)

DS3883 9-Bit BTL Data Transceiver
DS3884 6-Bit BTL Handshake Transceiver
DS3885 Arbitration Transceiver
DS3886 9-Bit BTL Latched Data Transceiver
DS3875 Arbitration Controller

Wire Wrap Boards with NSC's Futurebus+ Chipset

Presently, wire-wrap boards with NSC's Futurebus+ chipset are being offered by two companies, Mupac and Hybricon.

These boards have National's Futurebus+ transceivers surface mounted for optimum stub length and signal integrity. The wire-wrap option enable designers to implement and test different levels of the Futurebus+ specification in an actual Futurebus+ system.

These products offer a system designer a time-to-market advantage as transceiver/connector/board/backplane electrical evaulation can be completed while actual board design is in progress. A user can complete electrical signal characterization and noise budget analysis, without having to wait for fabricated prototype boards. Full evaluation on NSC's BTL transceivers and technology is now made one step easier for the end user.

Futurebus+ Backplane Manufacturers

Augat
33 Perry Avenue/P.O. Box 779
Attleboro, MA 02703
(508) 222-2202

BICC-VERO Electronics, Inc.
1000 Sherman Avenue
Hamden, CT 06514-1336
(203) 288-8001

Hybricon Corporation
12 Willow Road/P.O. Box 149
Ayer, MA 01432
(508) 772-5422

Mupac Corporation
10 Mupac Drive
Brockton, MA 02401
(508) 588-6110

Schroff, Inc.
170 Commerce Drive
Warwick, RI 02866
(401) 732-3770

Futurebus+ 2mm Connector Manufacturers

DuPont Electronics
Barley Mill Plaza
P.O. Box 80019
Bloomington, DE 19880-0019
(800) 237-2374

ITT ElectroMechanical Components Worldwide
1851 East Deere Avenue
Santa Ana, CA 92705-6500
(714) 261-5300

ATT Microelectronics
555 Union Blvd
Allentown, PA 18103
(800) 372-2447

Futurebus+ and Related Standards

P896.1	Futurebus+ Logical Layer Specifications
P896.2	Futurebus+ Physical Layer Specifications Includes: Profiles A, B, D, F, M and T
P896.3	Futurebus+ Systems Configuration Guide
P896.4	Futurebus+ Conformance Test
P1014.1	VME/Futurebus+ Bridge
P1101.2	2mm Connector Specification
P1101.3	Conduction Cooling Specification
P1156	Environmental and Power Supply Specification
P1194.1	Backplane Transceiver Logic (BTL) Specification
P1212	CSR Architecture
P1296.2	MultibusII/Futurebus+ Bridge
P1301	Metric Equipment Practice
P1301.1	Futurebus+ Hard Metric Boards + 2mm Connector
P1394	High Speed Serial Bus
P1596	SCI (Scalable Coherent Interface)

All specifications/standards for Futurebus+ are available from either the IEEE or VITA. Please see user groups for information on how to contact these associations.

Futurebus+ Users Groups and Standards Associations

VFEA International Trade Association (VITA)
10229 North Scottsdale Road
Suite B
Scottsdale, AZ 85253-1437 USA
(602) 951-8866

Futurebus Manufacturers and Users Group (FMUG)
P.O. Box 130
Fleet, Hampshire GU13 8XU England
0252-629937

Standards Department
IEEE Computer Society
1730 Massachusetts Avenue, N.W.,
Washington, D.C. 20036-1903
(202) 371-0101

National Semiconductor

Glossary of Terms for Futurebus + (Listed Alphabetically)

ADDRESS ONLY TRANSACTION: A bus transaction that does not include a data phase. The only information transferred is contained within the connection phase and, in some cases, the disconnection phase.

APPLICATION ENVIRONMENT PROFILE (AEP): An application environment profile is a document that describes functional requirements and points to existing standards, selecting and binding options within those standards. An implementor who then designs a specific board and/or system SHOULD be reasonably assured that another designer's (manufacturer or supplier) boards will properly function within the same system. This includes all aspects of definition: mechanical, electrical, protocol environmental, and system considerations.

ARBITRATION: The process of selecting the next bus master.

ARBITRATION MESSAGE: An event number broadcast on the arbitration bus to all modules on the bus.

BACKPLANE: An electronic circuit board and connectors used to connect other boards together electrically. The backplane connects selected pins of the connectors, thus providing the medium for the transfer of signals needed for the operation of the bus.

BACKPLANE BUS: A means to connect circuit modules using common signal traces on a backplane and a standard set of rules.

BEAT: An event that begins with the transition on a synchronization line by the master followed by the release of an acknowledge line by one or more slaves. Command and data information may be transferred from the master to one or more slaves in the first half of the beat. During the second half of the beat the slaves may transfer capability, status, and data information back to the master.

BIU: The BIU (Bus Interface Unit) refers to the logic on a module that converts bus signals and the protocols to and from signals which are compatible with the functional logic of the module.

BLOCK READ TRANSACTION: An address beat followed by a block of one or more data read transfers from a set of contiguous addresses beginning with the address in the address beat. This is terminated by the appropriate style of end beat.

BLOCK WRITE TRANSACTION: An address beat followed by a block of one or more data write transfers to a set of contiguous addresses beginning with the address in the address beat. This is terminated by the appropriate style of end beat.

BRIDGE: A bridge is a protocol converter and Control and Status Register Architecture (CSRA) unit which interfaces between two or more buses. The buses may adhere to different bus standards for mechanical, electrical and logical operation (such as a bridge from Futurebus + to VMEbus or to Multibus II).

BUS TENURE: The duration of a master's control of the bus; i.e., the total time that a module has the right to initiate bus transactions.

BUS TRANSACTION: An event initiated with a connection phase and terminated with a disconnection phase. Data may or may not be transferred during a bus transaction.

BUSY: If a slave cannot or should not accept a bus transaction at that time it, or any other module, may issue a Busy status to the master of the transaction. The master must then release the bus. The master must relinquish the bus and may re-acquire and re-try the transaction after a suitable time interval.

BYTE LANE: A data path formed by eight data lines and one parity line used to carry a single byte among the system modules.

CACHE COHERENCE: A system of caches is said to be coherent with respect to a cache line if each cache and main memory in the coherence domain observes all modifications of that same cache line. A modification is said to be observed by a cache when any subsequent read would return the newly written value.

CACHE MEMORY: A buffer memory inserted between one or more processors and the bus, used to hold currently active copies of blocks from main memory. Cache memories exploit spatial locality by what is brought into a cache. Temporal locality is exploited by what is removed from the cache.

COHERENCE DOMAIN: A coherence domain is a region in a multiple-cache system inside which cache consistency measures are enforced. In a system which contains bus bridges, a coherence domain may or may not be extensible beyond the local bus through a bus bridge to remote buses.

COHERENCE LINE: A data block for which cache consistency attributes are maintained.

COMPETITOR: A module actively participating in the current control acquisition cycle of the arbitration process.

CONNECTED SLAVE: A slave that is permitted to participate actively in the data transfer handshake. A selected slave that is not disabled (including a diverted slave), a reflecting slave, and an intervening slave are connected slaves. A disabled slave or, an unselected slave that is not intervening or reflecting, is not a connected slave.

CONNECTED TRANSACTION: A transaction in which both the request and response are performed within the same bus transaction.

CONNECTION PHASE: A beat that begins with the assertion of the address synchronization line followed by the release of an address acknowledge line. It is used to broadcast the address and command information. Modules determine whether they wish to take part in the transaction based on this information.

CONTROL ACQUISITION: The total of all bus activity associated with acquiring exclusive control of the bus.

COPYBACK CACHE: A cache memory scheme with the attribute that data written from the processor is normally written to the cache rather than the main memory. Modified data in the cache is written to the main memory when a cache line flush or replacement forces it to be written to main memory to avoid loss of the data.

CSR: Control and Status Register.

CSRA: Control and Status Register Architecture.

DATA PHASE: A period within a transaction used to transfer data.

DEADLOCK: Deadlock occurs when actors are waiting on actions that can only be performed by those waiting.

DISABLED SLAVE: A selected slave that detects that an intervening slave is participating in the data transfer in its place. A disabled slave treats the transaction as an address-only transaction and does not participate in the data transfer phase. A slave may be disabled only during single-slave mode transactions.

DISCONNECTION PHASE: A period within a transaction used to return the bus signals to their quiescent state. In addition, this phase might be used to transfer additional information required to perform or abort the requested operation.

DIVERTED SLAVE: A selected slave that detects that a reflecting slave is participating in the data transfer in its place. A diverted slave participates in the transfer by reading the data being transferred between the master and the reflecting slave. A slave may be diverted only during single-slave mode transactions.

DMA: A direct memory access architecture is an option capability of an I/O controller. After being started by the processor, the I/O controller with DMA capabilities can access their commands, fetch data, and report status by accessing memory directly.

DOUBLET: A set of two adjacent bytes.

EMERGENCY MESSAGE: An arbitration cycle with a special high arbitration number, which is selected from a set of numbers assigned to emergency messages.

ENTRANT: A live inserted module in the process of aligning itself with the arbitration protocol.

FAIRNESS: A bus request protocol in which a module refrains from acquiring the bus in order to allow other modules of the fairness class to use the bus.

FORWARD PROGRESS: A module which is not blocked from performing the tasks necessary to achieve its goal is said to be making forward progress. Forward progress is guaranteed only in the absence of deadlock or starvation.

FREE BYSTANDER: A free bystander can be a participating slave that is no longer an entrant, or a potential master that has no current need to acquire the bus and is not fairness inhibited.

GEOGRAPHICAL ADDRESS: A unique identifier statically assigned to each logical module slot on the bus and assumed by any module connected to that slot.

INACTIVE MODULES: This category contains all the modules that have no need to participate in the control acquisition process in any way.

INHIBITED BYSTANDER: An inhibited bystander is a potential master that has no current need to acquire the bus and is fairness inhibited.

INTERVENING SLAVE: An unselected slave that disables the selected slave and replaces that selected slave with itself. Intervening slaves can operate only during single-slave mode transactions.

LIVE INSERTION: Insertion of boards into a backplane can be performed when power is OFF or when power is ON. The process of inserting or withdrawing boards into and from a backplane when power is ON is referred to as "Live Insertion".

LIVELOCK: Livelock is a metastable situation in which some modules acquire and release resources in a way that none of them make progress.

LOCKING: A facility whereby a module is requested to guarantee exclusive access to addressed data, blocking other modules from accessing that data. This allows indivisible operations to be performed on addressed resources.

MASTER: A module which has acquired control of the bus through the control acquisition procedure.

MASTER ELECT: A master elect is the module that has won the arbitration process and is waiting for the master to transfer control to it.

MIXED TRANSACTION: An address beat followed by any number or combination of data write and data read transfers to a single location using the single address transfer mode. This is terminated by the appropriate style of end beat.

MODULE: A collection of circuitry designed to perform Futurebus+ operations.

MONARCH PROCESSOR: A monarch processor, or simply the monarch, is the processor selected to manage the configuration and initialization of all modules on that logical bus. A grand monarch is a processor selected to direct the configuration and initialization of an entire system with multiple interconnected logical buses.

NODE: A node is a set of control registers addresses (including identification-ROM and reset command register), which are initially defined in a 4k-byte (minimum) initial node address space. Each node can be reset independently.

OCTLET: A set of eight adjacent bytes.

PACKET DATA TRANSFER PROTOCOL: A very fast but technology dependent non-compelled transfer mechanism which uses a compelled protocol over the entire packet to provide flow control.

PARALLEL CONTENTION ARBITRATION: A process whereby modules assert their unique arbitration number of a parallel bus and release signals according to an algorithm such that after a period of time the winner's number appears on the bus.

PARKING: The process whereby the master retains control of the bus without actively using in. Parking typically occurs when no other module has requested the bus. The idle bus arbitration mechanism may be used when the bus is parked.

PARTICIPATING SLAVE: A participating slave may be in any of these states; 1) entrant, 2) bystander.

PORT: A port is the interconnection of a bus bridge to a bus. From the point of view of a particular node on a bus, the node connects to the local port of a bus bridge, while ports to other buses connected to the same bus bridge are called remote ports.

POTENTIAL MASTER: A potential master is a module that is capable of participating in the control acquisition process and taking full control of the bus. A potential master may be in any of these states; 1) entrant, 2) free bystander, 3) inhibited bystander, 4) competitor, 5) withholder, 6) master elect, 7) master, 8) recompeting master.

POTENTIAL SLAVE: A module that is capable of being addressed by and is able to carry out transactions with the master.

PRIORITY: A bus request protocol in which the module with the highest arbitration number acquires the bus.

PREEMPTION: Preemption occurs when the arbitration process is restarted after a master elect has been chosen in order to allow a high priority module to jump the queue.

QUADLET: A set of four adjacent bytes.

RECOMPETING MASTER: The module that is in control of the bus and has initiated a control acquisition procedure in order to unlock slave interfaces.

REFLECTING SLAVE: An unselected slave that forces the selected slave into a write only mode. In read transfers the reflecting slave substitutes itself for the selected slave in providing data while causing the selected slave to store the data that appears on the bus. In write transfers the reflecting slave copies the data into itself as well as the selected slave. Reflecting slaves can operate only during single-slave mode transactions.

REPOSITORY OF LAST RESORT: In a cache-based environment, the repository of last resort of shared data is a physical storage location that is the ultimate source and destination of the datum that is shared.

REQUEST: A request is a command generated by a requester, to initiate an action on a responder. For a processor-to-memory read transaction, for example, the request transfers the memory address and command from the processor to memory. In the case of a split transaction the request would be a separate bus transaction. In the case of a connected transaction the request would be the connection phase of a bus transaction.

REQUESTER: A transaction initiated by a module sending a request (containing address, command, and sometimes data). This module is referred to as the requester.

RESPONDER: A transaction is completed by a module sending a response (containing the completion status and sometimes data).

RESPONSE: A reply generated by a responder, to complete a transaction initiated by a requester. For a processor-to-memory read transaction, for example, the response returns the data and status from the memory to the processor. In the case of a split transaction the response would be a separate bus transaction. In the case of a connected transaction the response would be the data and disconnection phases of a bus transaction.

ROUND ROBIN: An arbitration mode where, after a module acquires and uses the bus, it must then wait until all other modules currently requesting the bus at the same priority level have had a chance to use the bus.

SELECTED SLAVE: A potential slave that has been selected by the master because it recognized its address on the bus lines during an address transfer, and will become a connected slave, unless an intervening slave prevents it from becoming so.

SHARED MEMORY: The address space in the system accessible to all caching modules.

SKEW: The difference between the propagation delays of two or more signals passing through multiple paths in a device or along a set of signal lines in parallel.

SLAVE: A module that can be addressed and is capable of participating in bus transactions.

SNARF: A module is said to snarf a transaction if it takes a copy of data passing by on the bus even though it did not request it.

SNOOP: A module is said to snoop a transaction if it is not the master that originated the transaction or the repository of last resort for the data, but it still monitors the transaction. Cache memories snoop transactions to maintain coherence.

SPATIAL LOCALITY: The tendency for a program to reference closely related clusters of memory addresses over short time intervals.

SPLIT TRANSACTION: An operation in which the request is transmitted in one bus transaction and the response is transmitted in a separate subsequent bus transaction.

STARVATION: A system condition which occurs when one or more modules perform no useful work for an indefinite period of time due to lack of access to the bus or other system resources.

STRONG SEQUENTIAL CONSISTENCY: A system is strong sequentially consistent if each cache in the system observes all modifications to all cache lines in the same order.

TEMPORAL LOCALITY: The tendency for a program to reference the same memory locations over short time intervals.

TRANSFER LINE SIZE: The size of the block of data transferred to or from main memory in a caching environment.

UNSELECTED SLAVE: A potential slave module that does not recognize its address on the bus during an address transfer.

WEAK SEQUENTIAL CONSISTENCY: A system exhibits weak sequential consistency if references to global synchronizing variables are strong sequential consistency, and if no reference to a synchronizing variable is issued by any processor until all previous modifications to global data have been observed by all caches, and if no reference to global data is issued by any processor until all previous modifications to synchronizing variables have been observed by all caches.

WITHHOLDER: A withholder is a potential master that requires control of the bus module and is fairness inhibited.

Section 6

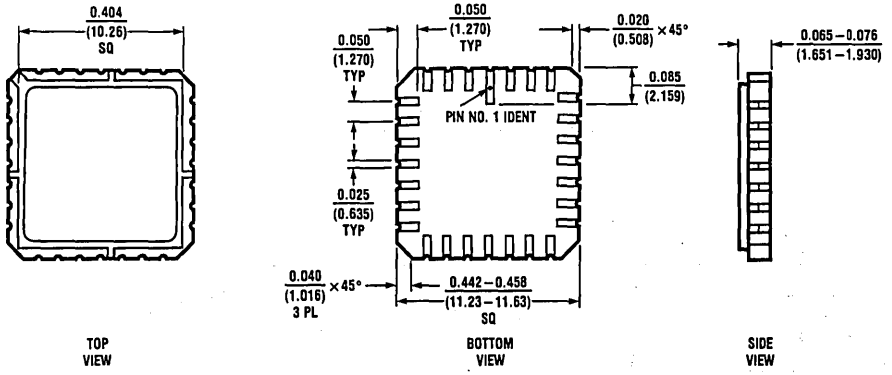
Physical Dimensions

Section 6 Contents

National Semiconductor

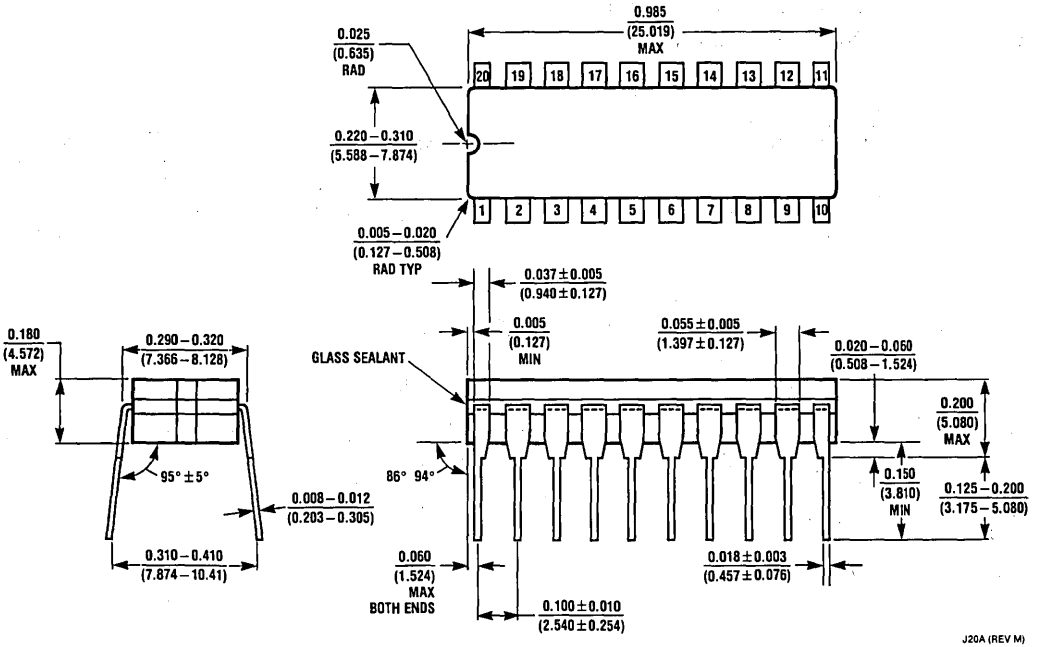
All dimensions are in inches (millimeters)

28 Terminal Ceramic Leadless Chip Carrier (E)
NS Package Number E28A



E28A (REV C)

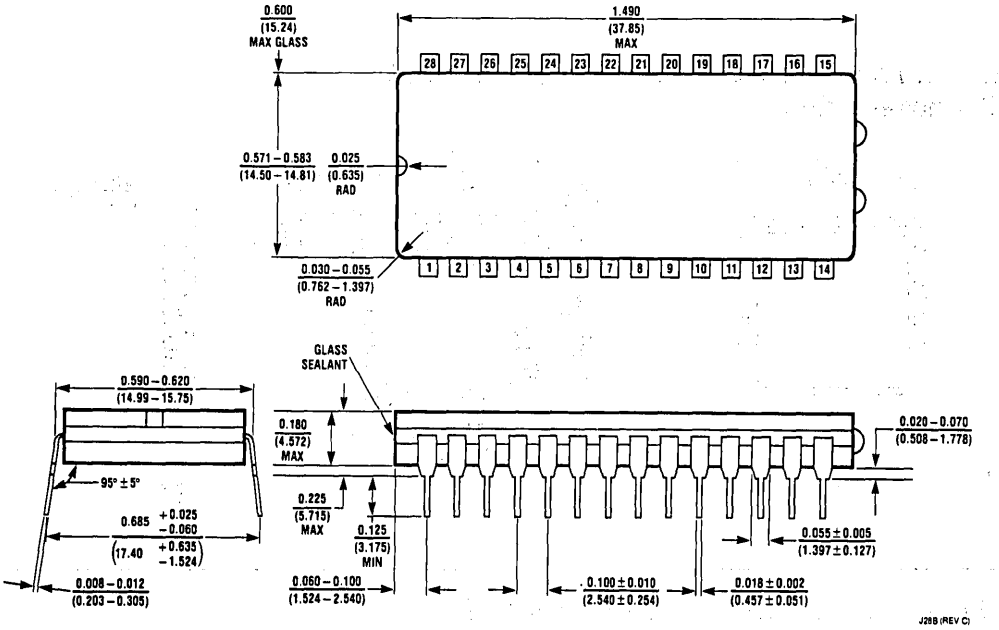
20 Lead Ceramic Dual-In-Line Package (J)
NS Package Number J20A



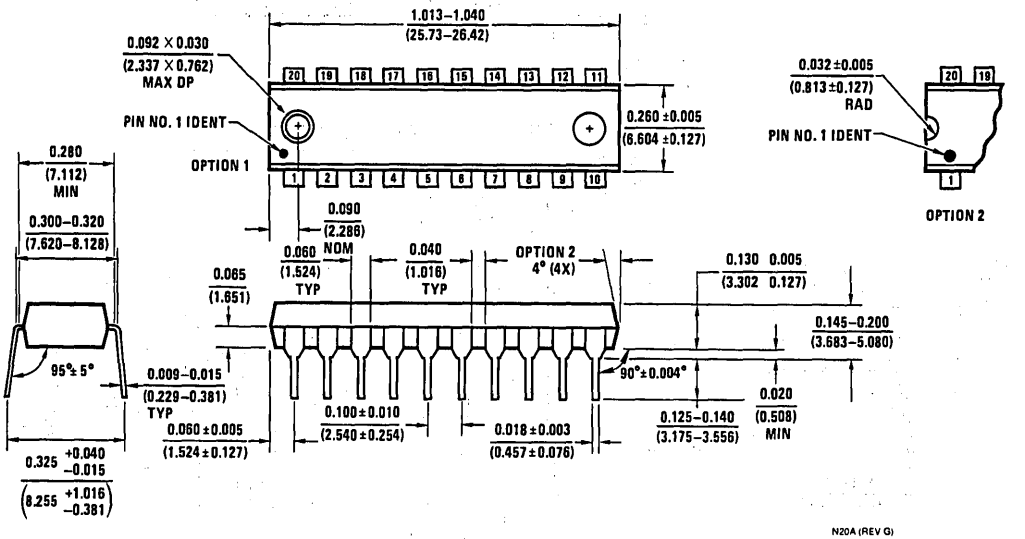
J20A (REV M)

6

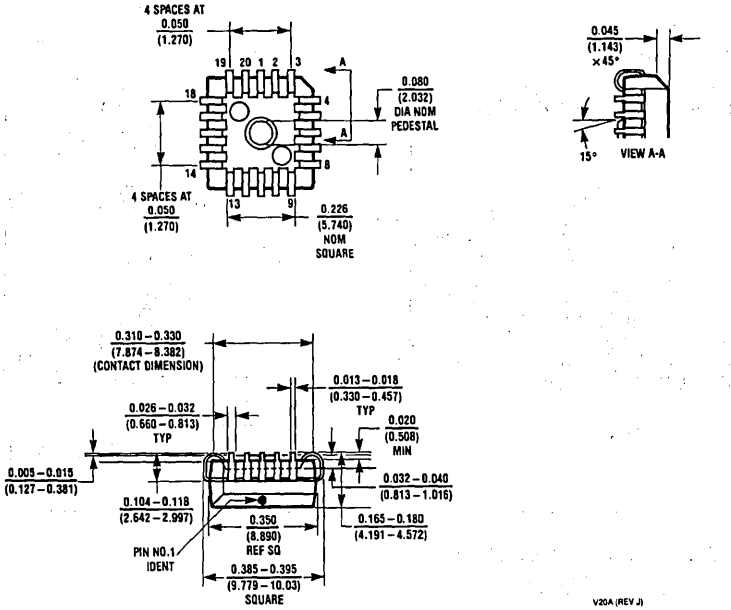
28 Lead Ceramic Dual-In-Line Package (J)
NS Package Number J28B



20 Lead Molded Dual-In-Line Package (N)
NS Package Number N20A

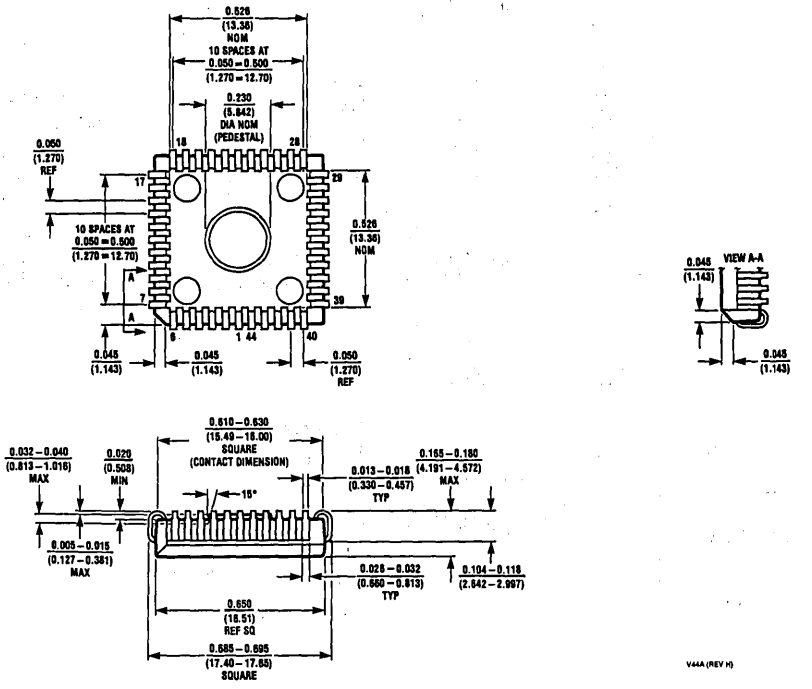


20 Lead Plastic Chip Carrier (V)
NS Package Number V20A



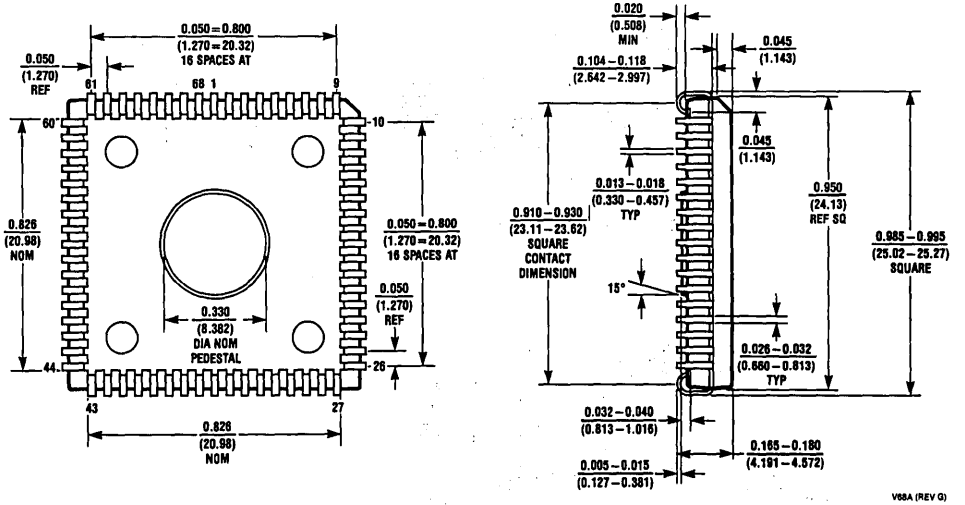
V20A (REV J)

44 Lead Plastic Chip Carrier (V)
NS Package Number V44A

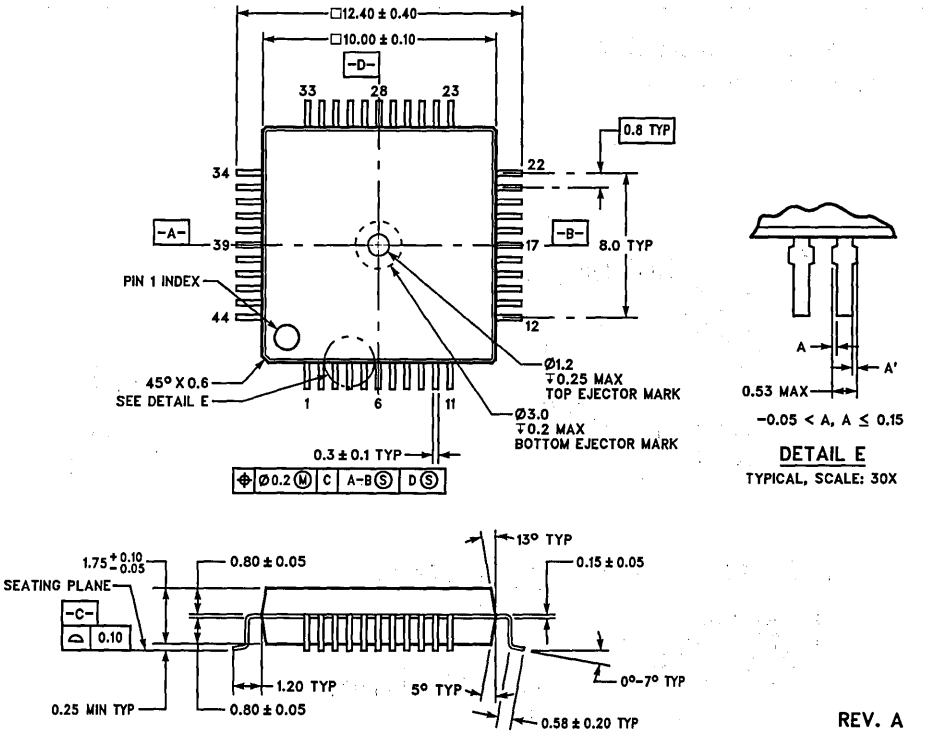


V44A (REV H)

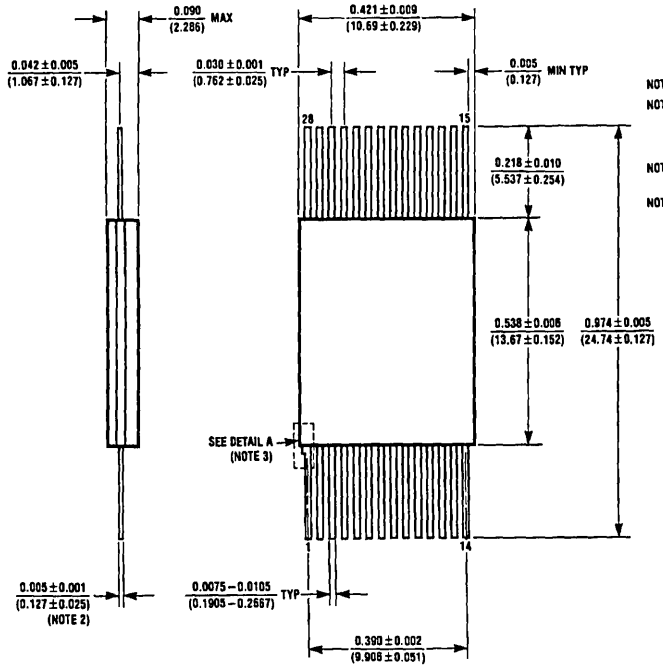
68 Lead Plastic Chip Carrier (V)
NS Package Number V68A



44 Lead Plastic Quad Flatpak (VF)
NS Package Number VF44B



28 Lead Ceramic Flatpak (W)
NS Package Number W28B

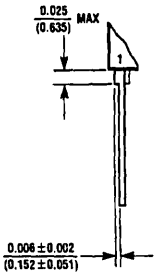


NOTES: UNLESS OTHERWISE SPECIFIED

NOTE 1. LEAD FINISH: SOLDER DIPPED WITH Sn60 OR Sn63 SOLDER CONFORMING TO MIL-M-38510 TO A MINIMUM THICKNESS OF 200 MICROINCHES (5.08 MICROMETER). SOLDER MAY BE APPLIED OVER LEAD BASIS METAL OR Sn PLATE.

NOTE 2. LEAD THICKNESS MAY BE INCREASED BY 0.003 INCHES (0.08mm). MAXIMUM AFTER LEAD FINISH IS APPLIED.

NOTE 3. LEAD 1 IDENTIFICATION SHALL BE:
a) A NOTCH OR OTHER IDENTIFICATION MARK WITHIN THIS AREA, OR
b) A TAB ON LEAD 1, EITHER SIDE.



DETAIL A (NOTE 3)

W28B (REV A)

NOTES

NOTES

National Semiconductor

Bookshelf of Technical Support Information

National Semiconductor Corporation recognizes the need to keep you informed about the availability of current technical literature.

This bookshelf is a compilation of books that are currently available. The listing that follows shows the publication year and section contents for each book.

Please contact your local National sales office for possible complimentary copies. A listing of sales offices follows this bookshelf.

We are interested in your comments on our technical literature and your suggestions for improvement.

Please send them to:

Technical Communications Dept. M/S 16-300
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090

ALS/AS LOGIC DATABOOK—1990

Introduction to Advanced Bipolar Logic • Advanced Low Power Schottky • Advanced Schottky

ASIC DESIGN MANUAL/GATE ARRAYS & STANDARD CELLS—1987

SSI/MSI Functions • Peripheral Functions • LSI/VLSI Functions • Design Guidelines • Packaging

CMOS LOGIC DATABOOK—1988

CMOS AC Switching Test Circuits and Timing Waveforms • CMOS Application Notes • MM54HC/MM74HC MM54HCT/MM74HCT • CD4XXX • MM54CXXX/MM74CXXX • Surface Mount

DATA ACQUISITION LINEAR DEVICES—1989

Active Filters • Analog Switches/Multiplexers • Analog-to-Digital Converters • Digital-to-Analog Converters Sample and Hold • Temperature Sensors • Voltage Regulators • Surface Mount

DATA COMMUNICATION/LAN/UART DATABOOK—1990

LAN IEEE 802.3 • High Speed Serial/IBM Data Communications • ISDN Components • UARTs Modems • Transmission Line Drivers/Receivers

DISCRETE SEMICONDUCTOR PRODUCTS DATABOOK—1989

Selection Guide and Cross Reference Guides • Diodes • Bipolar NPN Transistors Bipolar PNP Transistors • JFET Transistors • Surface Mount Products • Pro-Electron Series Consumer Series • Power Components • Transistor Datasheets • Process Characteristics

DRAM MANAGEMENT HANDBOOK—1989

Dynamic Memory Control • Error Detection and Correction • Microprocessor Applications for the DP8408A/09A/17/18/19/28/29 • Microprocessor Applications for the DP8420A/21A/22A Microprocessor Applications for the NS32CG821

EMBEDDED SYSTEM PROCESSOR DATABOOK—1989

Embedded System Processor Overview • Central Processing Units • Slave Processors • Peripherals Development Systems and Software Tools

FDDI DATABOOK—1991

FDDI Overview • DP83200 FDDI Chip Set • Development Support • Application Notes and System Briefs

F100K ECL LOGIC DATABOOK & DESIGN GUIDE—1990

Family Overview • 300 Series (Low-Power) Datasheets • 100 Series Datasheets • 11C Datasheets
ECL BiCMOS SRAM, ECL PAL, and ECL ASIC Datasheets • Design Guide • Circuit Basics • Logic Design
Transmission Line Concepts • System Considerations • Power Distribution and Thermal Considerations
Testing Techniques • Quality Assurance and Reliability • Application Notes

FACT™ ADVANCED CMOS LOGIC DATABOOK—1990

Description and Family Characteristics • Ratings, Specifications and Waveforms
Design Considerations • 54AC/74ACXXX • 54ACT/74ACTXXX • Quiet Series: 54ACQ/74ACQXXX
Quiet Series: 54ACTQ/74ACTQXXX • 54FCT/74FCTXXX • FCTA: 54FCTXXXA/74FCTXXXA

FAST® ADVANCED SCHOTTKY TTL LOGIC DATABOOK—1990

Circuit Characteristics • Ratings, Specifications and Waveforms • Design Considerations • 54F/74FXXX

FAST® APPLICATIONS HANDBOOK—1990

Reprint of 1987 Fairchild FAST Applications Handbook
Contains application information on the FAST family: Introduction • Multiplexers • Decoders • Encoders
Operators • FIFOs • Counters • TTL Small Scale Integration • Line Driving and System Design
FAST Characteristics and Testing • Packaging Characteristics

GENERAL PURPOSE LINEAR DEVICES DATABOOK—1989

Continuous Voltage Regulators • Switching Voltage Regulators • Operational Amplifiers • Buffers • Voltage Comparators
Instrumentation Amplifiers • Surface Mount

GRAPHICS HANDBOOK—1989

Advanced Graphics Chipset • DP8500 Development Tools • Application Notes

INTERFACE DATABOOK—1990

Transmission Line Drivers/Receivers • Bus Transceivers • Peripheral Power Drivers • Display Drivers
Memory Support • Microprocessor Support • Level Translators and Buffers • Frequency Synthesis • Hi-Rel Interface

LINEAR APPLICATIONS HANDBOOK—1986

The purpose of this handbook is to provide a fully indexed and cross-referenced collection of linear integrated circuit applications using both monolithic and hybrid circuits from National Semiconductor.

Individual application notes are normally written to explain the operation and use of one particular device or to detail various methods of accomplishing a given function. The organization of this handbook takes advantage of this innate coherence by keeping each application note intact, arranging them in numerical order, and providing a detailed Subject Index.

LS/S/TTL DATABOOK—1989

Contains former Fairchild Products
Introduction to Bipolar Logic • Low Power Schottky • Schottky • TTL • TTL—Low Power

MASS STORAGE HANDBOOK—1989

Rigid Disk Pulse Detectors • Rigid Disk Data Separators/Synchronizers and ENDECs
Rigid Disk Data Controller • SCSI Bus Interface Circuits • Floppy Disk Controllers • Disk Drive Interface Circuits
Rigid Disk Preamplifiers and Servo Control Circuits • Rigid Disk Microcontroller Circuits • Disk Interface Design Guide

MEMORY DATABOOK—1990

PROMs, EPROMs, EEPROMs • TTL I/O SRAMs • ECL I/O SRAMs

MICROCONTROLLER DATABOOK—1989

COP400 Family • COP800 Family • COPS Applications • HPC Family • HPC Applications
MICROWIRE and MICROWIRE/PLUS Peripherals • Microcontroller Development Tools

MICROPROCESSOR DATABOOK—1989

Series 32000 Overview • Central Processing Units • Slave Processors • Peripherals
Development Systems and Software Tools • Application Notes • NSC800 Family

PROGRAMMABLE LOGIC DATABOOK & DESIGN MANUAL—1990

Product Line Overview • Datasheets • Designing with PLDs • PLD Design Methodology • PLD Design Development Tools
Fabrication of Programmable Logic • Application Examples

REAL TIME CLOCK HANDBOOK—1989

Real Time Clocks and Timer Clock Peripherals • Application Notes

RELIABILITY HANDBOOK—1986

Reliability and the Die • Internal Construction • Finished Package • MIL-STD-883 • MIL-M-38510
The Specification Development Process • Reliability and the Hybrid Device • VLSI/VHSIC Devices
Radiation Environment • Electrostatic Discharge • Discrete Device • Standardization
Quality Assurance and Reliability Engineering • Reliability and Documentation • Commercial Grade Device
European Reliability Programs • Reliability and the Cost of Semiconductor Ownership
Reliability Testing at National Semiconductor • The Total Military/Aerospace Standardization Program
883B/RETS™ Products • MILS/RETS™ Products • 883/RETS™ Hybrids • MIL-M-38510 Class B Products
Radiation Hardened Technology • Wafer Fabrication • Semiconductor Assembly and Packaging
Semiconductor Packages • Glossary of Terms • Key Government Agencies • AN/ Numbers and Acronyms
Bibliography • MIL-M-38510 and DESC Drawing Cross Listing

SPECIAL PURPOSE LINEAR DEVICES DATABOOK—1989

Audio Circuits • Radio Circuits • Video Circuits • Motion Control Circuits • Special Function Circuits
Surface Mount

TELECOMMUNICATIONS—1990

Line Card Components • Integrated Services Digital Network Components • Analog Telephone Components
Application Notes

National Semiconductor

 CA/3307 RRD/RRD10M41/Printed in U.S.A.