1990

A multiple-bus, active backplane architecture for multiprocessor systems

Scott Alan Irwin
Iowa State University

Order Number 9110511

A multiple-bus, active backplane architecture for multiprocessor systems

Irwin, Scott Alan, Ph.D.

Iowa State University, 1990

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

A multiple-bus, active backplane architecture for multiprocessor systems

by

Scott Alan Irwin

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Electrical Engineering and Computer Engineering
Major: Computer Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Members of the Committee:

Signature was redacted for privacy.

Iowa State University
Ames, Iowa
1990

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

System designers have traditionally relied on rapid advances in semiconductor and packaging technologies to deliver higher-performance processors. As these technologies mature and begin to approach their physical limitations, designers are looking to multiple-processor architectures to achieve their desired performance. The appeal of such architectures is further enhanced by the availability of inexpensive microprocessors.

To achieve their performance potential, multiprocessor systems require an efficient method of interconnecting the processors. The most common interconnection method is the single, time-shared bus. While being simple and expandable, it has also become a major bottleneck in multiprocessor systems. Fully interconnected systems, however, are very expensive to implement. The various other interconnects that have been proposed, such as hypercubes, are best suited to a limited number of specific applications. A review of current literature dealing with multiprocessor architectures and interconnection methods is presented in Chapter 2.

This research identifies several problems associated with current multiprocessor interconnection networks, focusing primarily on general-purpose, shared-memory configurations. A description of the problems addressed in this dissertation is presented in Chapter 3.

The primary motivation behind this project is Control Data Corporation's desire to upgrade their AN/AYK-14 airborne computer system, which is used in various military projects. In addition to higher system throughput, their needs include high reliability, real-time support, a prioritized event structure, and cache memory support. Although driven by CDC's needs, these requirements are important to commercial systems, as well.

In addition, it has become increasingly important for military systems to exploit commercial silicon, standards, and expertise in order to remain cost-effective. Therefore, this project examines a broader range of general-purpose systems, rather than focusing strictly on CDC's military requirements.

A multiple-bus, active backplane architecture is proposed as an alternative to current interconnects. It represents a unique combination of new and existing technologies, applied to several levels of the backplane. The architecture is scalable and expandable, with support for synchronization and cache coherence. An overview of the proposed backplane architecture is presented in Chapter 4.

Each bus uses a source-synchronous, word-serial transfer protocol. This allows data to be streamed at a very high rate, while minimizing the number of signals required for each bus. A new control acquisition scheme was developed which combines collision detection and priority arbitration. This minimizes bus access time without requiring additional signal lines. A complete specification of the bus protocol is given in Chapter 5.

To reduce the capacitive loading of the bus, an active backplane is employed. The transceiver and bus interface unit are mounted below the connector, directly on the backplane. This increases the characteristic impedance of the bus, while reducing

the end-to-end propagation delay. In addition, the variability of the backplane load is eliminated, allowing the bus to be properly terminated.

With reduced drive current requirements, a CMOS transceiver, suitable for VLSI implementation, is used. It incorporates the collision detection capability required for the control acquisition scheme. Initial transceiver prototypes were designed and fabricated in 2-μm CMOS. These have been successfully tested at transfer rates in excess of 50 MHz. The results of backplane and transceiver prototype tests are presented in Chapter 6.

Finally, Chapter 7 includes a summary of the project along with future directions for this research.

CHAPTER 2. LITERATURE REVIEW

Multiple-processor architectures have been studied extensively. This chapter presents a review of current literature dealing with shared-memory multiprocessors. Included are discussions of interconnection methods and backplane technology.

Multiprocessors

The availability of inexpensive, highly-integrated processors has made multiple-processor architectures a popular choice for increasing the performance of computer systems. These systems can be categorized as *multiprocessors* or *multicomputers*, depending on the level at which the processors communicate [7].

As illustrated in Figure 2.1a, multiprocessors are characterized as being tightly coupled, with the processors communicating via shared memory. The shared memory may be located on global memory modules, distributed among the processors as local memory, or a combination of both. In any case, the global address space is used for processor synchronization and data sharing.

Multicomputers, on the other hand, are more loosely coupled, as illustrated in Figure 2.1b. Each processor has its own local memory, and cannot directly access another processor's memory. These systems communicate at a higher level, typically through a message-passing mechanism.

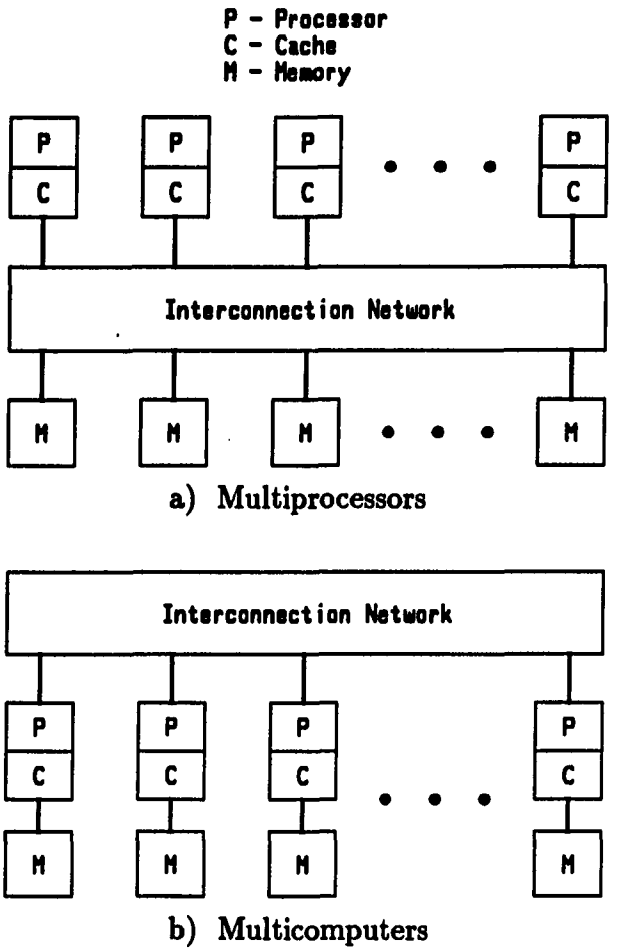


Figure 2.1: Multiple-processor architectures

This research will focus on shared-memory multiprocessors and their interconnection methods.

There are several multiprocessor design considerations which are not present with uniprocessor systems. These include coherency of data in private cache memories, synchronization between processors, and the implementation of an efficient message passing mechanism.

Cache Coherency

Cache memories are used extensively in uniprocessor systems to reduce the access delay to main memory. They are even more important in multiprocessor systems [17]. In addition to reducing data access times, private caches can also reduce contention among processors for the interconnection network, since most requests can be satisfied locally, without an access to main memory.

A cache is typically organized into blocks, which are some multiple number of processor words. Although a processor can access a single word in the cache, all transfers between the cache and memory are done at the block level. This leads to a third advantage of using cache memories. Because of the fixed overhead in accessing main memory, a single word access is less efficient than a burst access. Therefore, the cache transforms the inefficient word accesses of the processor into more efficient block transfers to memory.

Shared-memory multiprocessors offer an efficient method of sharing data and instructions. However, when private caches are used, several copies of a shared block may be stored in different caches. When a shared block is writable, a data consistency problem exists. This is known as the *cache coherence problem.*

Many solutions to the cache coherence problem have been proposed [37]. These range from purely software methods, which attempt to eliminate shared, writable data during compilation, to complicated hardware solutions. Although hardware-based enforcement of coherency can add considerable complexity to the cache controller, it has the advantage of being more efficient and transparent to the programmer. As hardware costs continue to fall, these methods will become even more cost-effective than software solutions.

There are two primary policies for maintaining cache consistency. These are generally referred to as *copy-back* and *write-through*. In either case, processor reads are satisfied by the cache if a local copy exists. Where the policies differ is with respect to write operations.

With the copy-back policy (also known as write-invalidate), when a processor modifies a cache block, all other copies are invalidated. Subsequent modifications to this block by the same processor can be satisfied locally, since no other copies exist. The block is written back to main memory when the cache block is flushed. If another processor requests a modified block, the cache must supply the data, since memory is out-of-date.

The write-through policy (also known as write-update) enforces consistency by updating, rather than invalidating, all other copies of a cache block. Main memory may be updated at the same time as the other caches. In general, write-through caches are not as effective in reducing interconnection contention. This is due to the fact that a significant percentage of all processor requests involve writes, which a write-through cache must pass through to the interconnection network.

A large number of hardware-based cache coherency models have been proposed and compared [2], including *Write-Once* [18], *Berkeley* [30], *Illinois* [34], *Dragon* [2], and *Firefly* [2], to name a few. It has been shown that many of these (some with slight modifications) fall into a single class of compatible protocols [38]. Within this class, different protocols can interoperate, while consistency is maintained throughout the system.

This class of protocols is supported by a single coherency model, known as MOESI. The MOESI model was originally developed by the IEEE 896 Futurebus

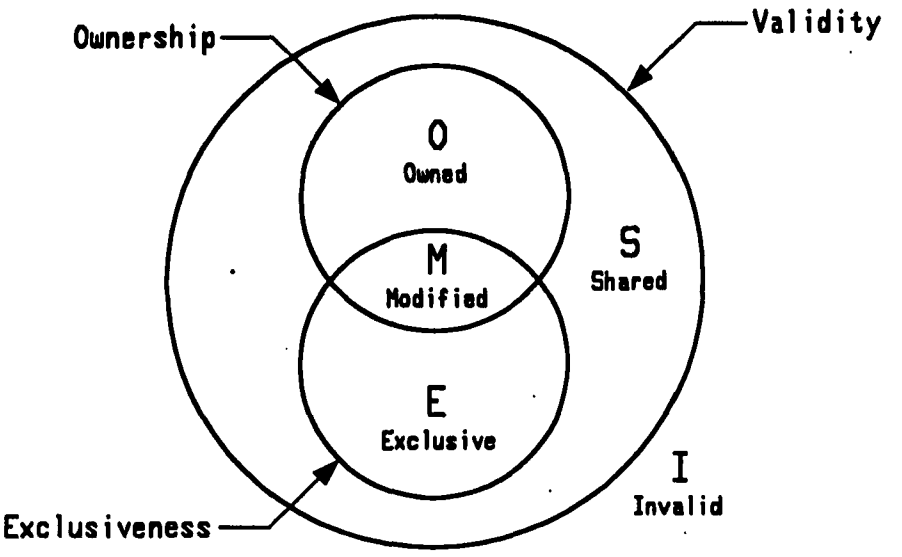


Figure 2.2: The MOESI Cache Model [39]

committee [26]. As illustrated in Figure 2.2, five possible states can be associated with each cache block. Each state represents the *validity, ownership,* and *exclusiveness* of a particular cache block relative to other caches. The model has since been modified for incorporation into the IEEE Futurebus+ standard [30], where it is known as the MESI model, due to the deletion of the *owned* state.

Synchronization

Multiprocessors offer an effective method of increasing the performance of computer systems, whether it be to increase the throughput of multiple processes, or decrease the execution time of a single task [16]. In either case, efficient processor synchronization methods are needed to reduce the number of operations required

over the interconnect and minimize the blocking of shared resources [19]. The finer the granularity of parallelism, the more important these methods become.

Synchronization primitives serve two primary purposes [16]. The first is to guarantee mutually exclusive access to shared resources, such as queues, databases, and process tables. A second purpose is to support the proper ordering of execution tasks. For instance, a consumer process must not use an item until it has been completed by a producer process.

As with cache coherency, synchronization solutions range from software methods, built on simple hardware primitives, to extremely complex hardware solutions. These include *test-and-set* [1, 19], *compare-and-swap* [16], *test-and-test-and-set* [1], *fetch-and-op* [14], *queuing locks* [19, 20], and *barrier* [16, 19] synchronization schemes.

Simple synchronization primitives, such as test-and-set and compare-and-swap, are relatively easy to implement in hardware. These instructions read and then write a memory location in a single atomic operation. Higher level constructs, such as semaphores and barriers, are then implemented in software using the hardware primitives. These methods can be inefficient, since a blocked processor may spin on a memory location, generating interconnection traffic. This traffic can be reduced in some cases [1].

More complicated hardware methods, such as fetch-and-op and queuing locks, require very complex memory controllers, but can significantly reduce the contention for the interconnection network. This contention can be reduced further by using *positive wake-up* rather than *busy-waiting* [14]. These complex hardware synchronization schemes are necessary for systems with a large number of processors or very fine-grained parallelism

Message Passing

Another form of interprocessor communication, known as message passing, can be used to exchange data or synchronize processor events. It is typically implemented at a higher level than the other forms of synchronization just discussed.

Message passing is more prominent in multicomputer architectures, where a processor cannot directly access another processor's memory. However, it can easily be implemented in the shared-memory of a multiprocessor. Its primary importance to multiprocessor architectures lies in using a standard message format to allow different architectures to communicate [29].

Interconnection Networks

To realize their performance advantage, multiple-processor systems require an efficient method of interconnecting the processors and memory. There have been many proposals, spanning the entire spectrum of cost and performance. These range from fully-connected systems to simple, time-shared bus configurations.

Complete Interconnects

The *fully* or *completely connected* network [24] offers the maximum possible bandwidth of any interconnection method. Since a direct link exists between each processor, the need for a complex routing mechanism is avoided. However, the number of interconnections increases with the square of the number of processors. This can become unwieldy, especially with a large number of processors.

Another complete interconnect is the *crossbar* network [24], which uses an array of switches to connect the processors to memory. Although each processor requires

only a single connection, crossbar networks can be very expensive to implement for a large number of processors [7]. Also, crossbar and fully connected networks can be under-utilized [31], since they typically exceed the bandwidth required.

Multistage Interconnection Networks

Multistage interconnection networks have been studied extensively in the last decade [7], including the *Banyan, Delta, Cube,* and *Omega* networks, to name a few. These share many of the same characteristics. They are typically implemented as a matrix of small switches, with some type of shuffle between consecutive stages.

Multistage interconnects strike a balance between tradeoffs in performance and cost [7]. While being less expensive than the crossbar, they offer higher bandwidth than single-bus systems. However, the lack of an efficient cache coherency mechanism makes them less suitable for shared-memory systems [18].

Bus-Based Systems

The most popular method of interconnection is the single, time-shared bus. This is typical of both commercial [26, 27, 28, 29] and military [11] applications. It has the advantage of being simple and expandable. While being the least expensive method, it offers only limited performance and reliability. The single system bus has become a major bottleneck in the throughput of multiprocessor systems [33].

Although technological gains have achieved limited performance improvements, most recent increases have come from expanded data and control paths. However, this can lead to high pin-count backplanes. For example, the 67 signal lines in the original 32-bit Futurebus standard are increased to 338 signals for a full 256-bit Futurebus+

implementation [9, 29]. When combined with the need for high-current bus drivers, this can lead to serious implementation problems from excessive power consumption.

An alternate way to increase the bandwidth of bus-based systems is by using multiple buses [31, 32]. It has been shown that for a given number of signal lines, multiple buses can yield a higher total bandwidth than a single wide bus [23].

Multiple-bus interconnects have the added benefit of increased reliability [33]. The number of signals can become unwieldy, however, if a high pin-count bus is replicated many times. This problem can be reduced by using high-speed, serial buses, which stream the transmission of control, address and data over a minimal width bus [3].

Multicomputer Interconnects

Various other interconnects have been proposed, such as the *hypercube* [7], *ring, star,* etc. These are typically used in multicomputer architectures and communication networks. Since the processors are not directly connected in most of these networks, some type of store-and-forward message routing must be used. The hypercube, while matching the communication patterns of several algorithms [35], is not well suited to general-purpose systems, due to its lack of scalability and expandability.

Backplane Technology

In terms of backplane technology, most modern buses, such as VMEbus and Multibus II, bear a striking resemblance to previous generation systems. These systems utilize bipolar transceivers, driving a passive bus, with all bus interface logic located on the plug-in modules. If error detection is provided, it is usually limited

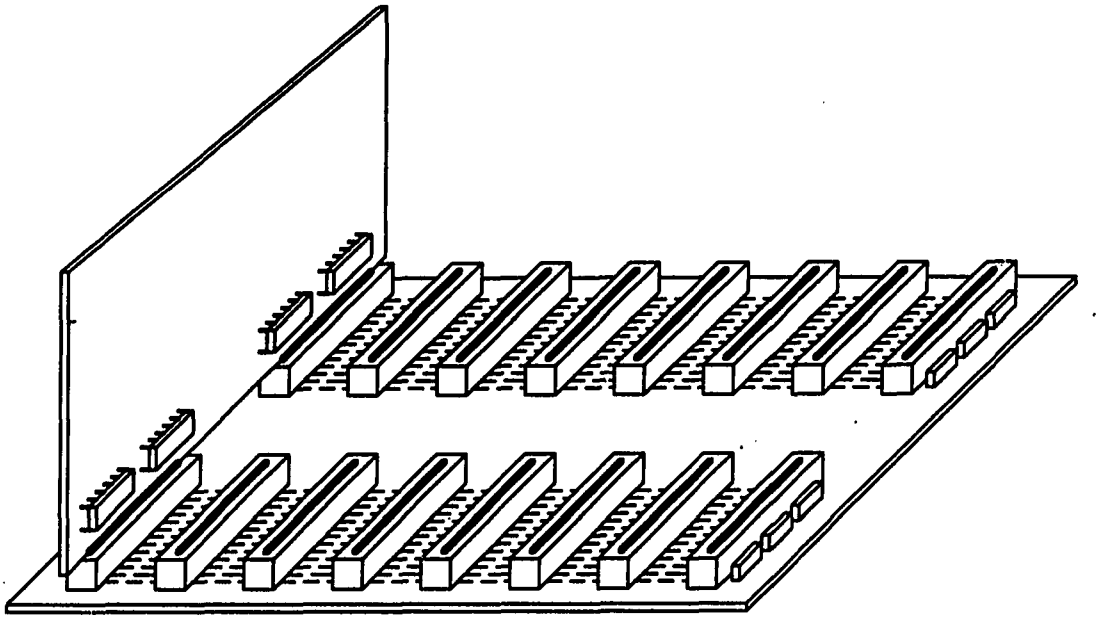


Figure 2.3: Typical passive backplane

to byte parity. Most performance increases can be attributed to advances in other areas, such as high-current drivers, reduced logic delays, and new control acquisition and synchronization algorithms [5, 8, 12, 40, 41, 42, 43, 44].

Typical Backplane

A typical passive backplane structure is illustrated in Figure 2.3. The backplane consists of a single or multi-layer printed circuit board. Stripline construction is used for bus signals, with termination resistor networks located at each end of the bus. Power is supplied to the modules from the backplane, either from striplines or power and ground planes. Modules are attached via finger or pin-and-socket connectors, with the bus transceivers mounted on the plug-in modules.

There are several problems which arise from this type of backplane structure. These include:

- Random loading at each slot, depending on whether a module is inserted
- Low characteristic bus impedance
- High drive current requirements
- Under-terminated bus lines, due to low bus impedance
- Bipolar drivers not suited to VLSI implementation

Each of these problems is directly related to the capacitive loading of the bus. By reducing the capacitive loading, each of these problems can be minimized.

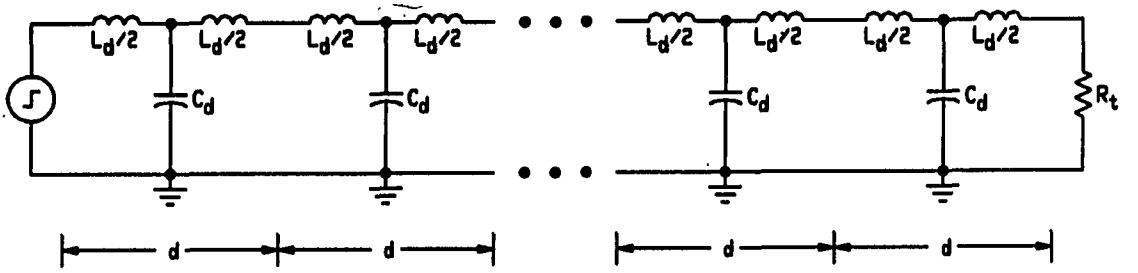
A primitive active backplane, with the transceiver input and output transistors placed below the connector, has been proposed [6]. Although it removes the connector capacitance from the bus, it utilizes a bipolar transceiver, which has higher output capacitance than CMOS, and is less suitable for VLSI implementation.

Transmission Line Characteristics

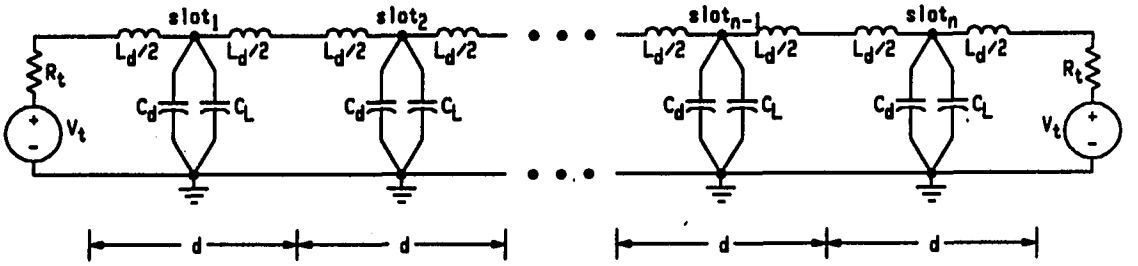
A backplane signal can be modeled as a *lumped-constant transmission line* [3], as shown in Figure 2.4a. The characteristic impedance and signal propagation velocity for this model can be derived as:

$$Z_0 = \sqrt{\frac{L_d}{C_d}} \tag{2.1}$$

$$\nu_p = \frac{d}{\sqrt{L_d C_d}} \tag{2.2}$$



a) Unloaded



b) Loaded

Figure 2.4: Transmission line model

where C_d and L_d are are the distributed capacitance and inductance of the signal line per unit distance d. Typical values for stripline construction are $Z_0 = 100\,\Omega$ and $\nu_p = 20\,\text{cm/ns}$ (2/3 the speed of light). Reversing these equations and solving for a slot separation of $d = 2\,\text{cm}$ gives $L_d = 10\,\text{nH/slot}$ and $C_d = 1\,\text{pF/slot}$. These numbers match experimental measurements taken from prototype backplanes.

The transmission line model must be modified to take into account the load introduced at each slot by the connector, traces, vias, and transceiver. The resulting model for a loaded backplane bus line is shown in Figure 2.4b. The characteristic

impedance and signal propagation velocity are:

$$Z_0 = \sqrt{\frac{L_d}{(C_d + C_L)}} \tag{2.3}$$

$$\nu_p = \frac{d}{\sqrt{L_d(C_d + C_L)}} \tag{2.4}$$

where C_L is the load introduced at each slot.

The end-to-end signal propagation delay down the backplane can be calculated by dividing the length of the backplane by the propagation velocity. This is given as:

$$T_{pd} = \frac{(n-1)d}{\nu_p} = (n-1)\sqrt{L_d(C_d + C_L)} \tag{2.5}$$

where n is the number of backplane slots. Several values of Z_0 and T_{pd} versus load capacitance for a 24-slot backplane are listed in Table 2.1. Note the drastic reduction in Z_0, even with small capacitive loads. This highlights the importance of keeping the bus loading to a minimum.

Table 2.1: Z_0 and T_{pd} versus C_L for a 24-slot backplane

C_L (pF)	Z_0 (Ω)	T_{pd} (ns)
0	100	2.3
1	71	3.3
2	58	4.0
3	50	4.6
4	45	5.1
5	41	5.6
6	38	6.1
7	35	6.5
8	33	6.9
9	31	7.3
10	30	7.6

Transmission Line Termination

A bus requires proper termination at each end to avoid signal reflections. Ideally, this is done with a resistance which matches the bus's characteristic impedance. Due to their low (and variable) characteristic impedance, this is unfeasible with most passive backplanes.

To reduce the DC drive current required, passive buses are usually terminated with a resistance higher than their characteristic impedance. However, this means the incident signal transition produced on the bus does not reach its final value until after several round-trip propagation delays. This effect is known as the *bus driving problem* [5, 25].

CHAPTER 3. PROBLEM DEFINITION

There are several deficiencies in current multiprocessor interconnection networks. This chapter identifies the problems addressed in this dissertation.

Bandwidth

The demand for more and more processors in tightly-coupled multiprocessor systems has placed a strain on current interconnection networks. These networks do not provide the bandwidth necessary to keep a large number of high-performance microprocessors fed with instructions and data, even with the use of private caches. Only multicomputer architectures currently support a large number of processors, due to their lower interconnect bandwidth requirements.

Scalability and Expandability

It is important for general-purpose multiprocessor systems to be both scalable and expandable. The throughput of the interconnection network must be capable of scaling with performance needs in order to protect hardware investments. A system must also be expandable to allow for changing needs in performance and functionality. For instance, more processors, I/O devices, disk interfaces, or graphics subsystems may need to be added to a system.

With the exception of bus-based systems, most interconnects lack the required expandability. Much of this is due to difficulties in mapping the logical interconnect structure into physical packaging. While single-bus systems are easily expanded, they typically are not scalable.

Fault-Tolerance

System reliability, while being important to all applications, is critical in several areas. These include aircraft, aerospace, and transaction processing applications, to name a few. Most current interconnection standards lack inherent fault-tolerance. When it is required, custom interconnects are developed for each application. This can be less cost-effective, however, since the economies-of-scale for general-purpose interconnects cannot be exploited.

Capacitive Loading

Several problems with current bus-based interconnects can be directly attributed to high capacitive loading. As discussed in Chapter 2, these include large end-to-end propagation delays, low characteristic impedances, high drive current requirements, and under-terminated bus lines. A primary goal of this research is the reduction of capacitive bus loading and its effects.

Number of Signal Lines

The most common method of increasing the bandwidth of current interconnects is by expanding the width of the data path. This can lead to a high pin-count

backplane. When combined with the need for high-current drivers, this can result in excessive power requirements. In addition, the large number of connector pins can reduce the system reliability.

Suitability to VLSI Implementation

Most current backplane transceivers and bus drivers are implemented in a bipolar technology. While offering higher drive current capabilities than CMOS transceivers, they typically require separate packaging from the bus interface logic. This takes up additional board space on the modules. Although BiCMOS processes can incorporate bipolar transceivers on-chip, they remain more expensive than CMOS processes.

Multiprocessor Support

To efficiently support multiple-processor systems, the interconnection network must provide the mechanisms for maintaining cache consistency and interprocessor communication. While some current interconnection networks provide this support, it is mentioned again for emphasis.

CHAPTER 4. THE PROPOSED ARCHITECTURE

The remainder of this dissertation presents a multiprocessor interconnection network which minimizes the problems discussed in Chapter 3. An overview of the architecture is presented in this chapter, followed by specifications of the protocol and backplane.

Overview

A multiple-bus, active backplane architecture is proposed as an alternative to current multiprocessor interconnects. An example of how this backplane can be physically implemented is illustrated in Figure 4.1.

By moving the bus transceivers from the plug-in modules to the backplane, the connector and module load capacitances are isolated from the bus. There are several benefits to this scheme. First, proper bus termination can be provided because the bus impedance is increased and the variability in loading is eliminated. Also, the support of live-insertion and withdrawal becomes easier.

Finally, the increase in characteristic bus impedance and resulting reduction in drive current requirements make it feasible to implement the bus transceivers in CMOS. Much of the bus interface logic can now be integrated into the backplane transceiver chip, thus freeing board space on the plug-in modules.

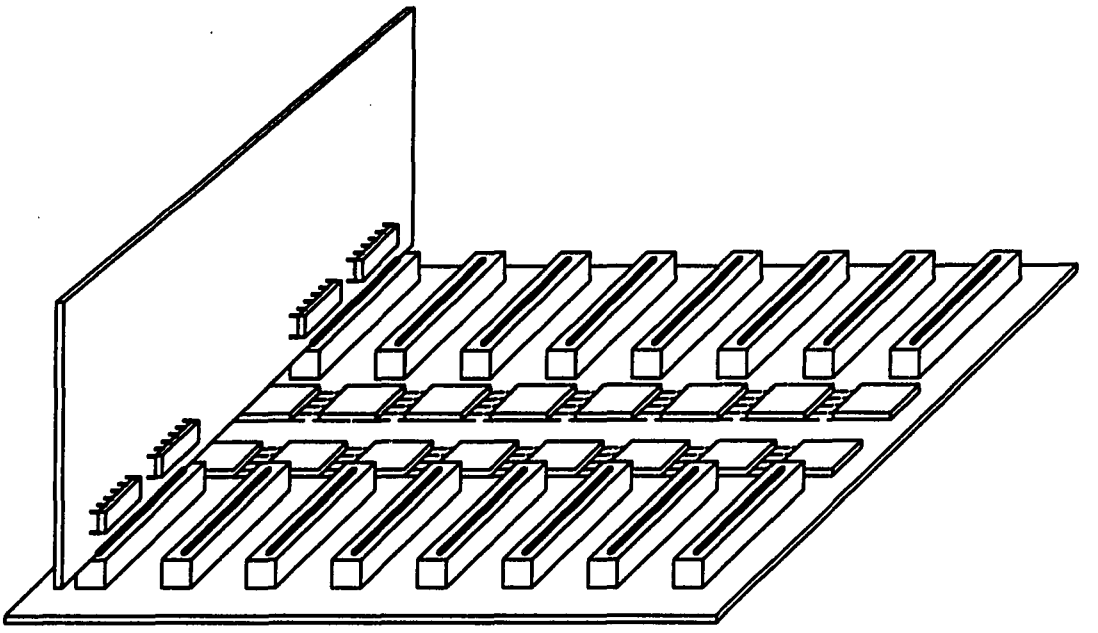


Figure 4.1: Active backplane structure

A block diagram of the architecture is shown in Figure 4.2. Each module connects to a minimum of two buses. The bandwidth of the interconnection network can be scaled by increasing the number of buses. Also, the expandability of bus-based systems is retained.

By using multiple buses, the fault-tolerance of the interconnect is increased. As long as any of the buses are operational, the system can continue to function, albeit with a loss in available bandwidth as buses fail.

To further improve the fault-tolerance of the architecture, the buses support *Single-bit Error Correction - Double-bit Error Detection (SEC-DED)*. By providing check-bits rather than the usual byte-parity, the ability of the system to tolerate and detect errors is greatly improved.

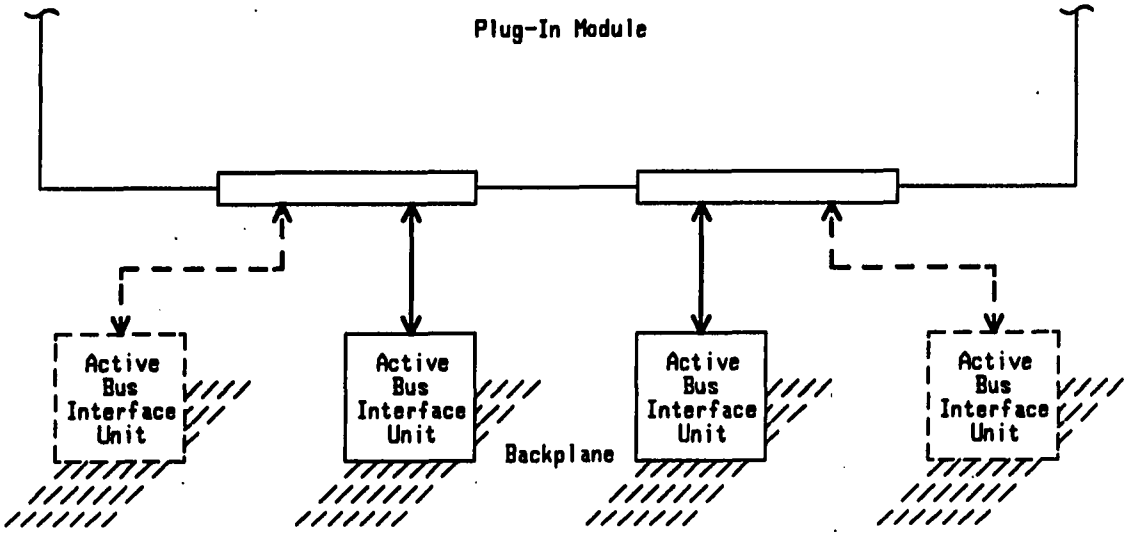


Figure 4.2: Block diagram of active backplane architecture

Each bus transfers control, address, and data in a word-serial manner over a multiplexed, 32-bit data path. Arbitration is also accomplished over these same signal lines. This keeps the total number of backplane signals manageable when replicating the bus. A source-synchronous transfer protocol allows data to be streamed on the bus at a very high rate. When combined with the use of minimal bus lines, this leads to improved pin-efficiency.

A new control acquisition scheme was developed to minimize the overhead of arbitrating on the data lines. The scheme combines the fast access of collision detection in lightly loaded systems with the deterministic outcome of priority arbitration under heavy loads. It also incorporates a fairness mode to prevent starvation. The bus protocol is specified in detail in Chapter 5.

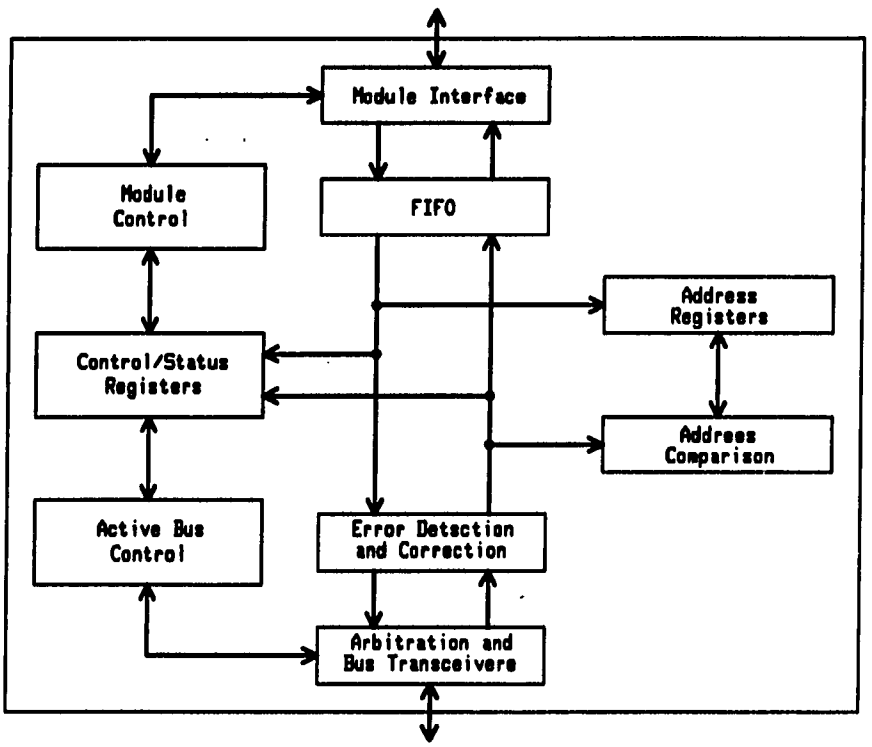


Figure 4.3: Active Bus Interface Unit block diagram

Active Bus Interface Unit

A block diagram of the *Active Bus Interface Unit (ABIU)* is shown in Figure 4.3. This represents the portion of the interconnection logic which is mounted on the backplane. It is important to incorporate as much bus interface functionality as possible into the ABIU.

Included in the ABIU are address recognition, priority arbitration, and transfer protocol circuitry. Also included are FIFOs and the transmit and receive state machines. The module interface could be customized for different microprocessor architectures. This dissertation will focus on the lower layers of the ABIU.

Multiprocessor Support

To efficiently implement multiprocessor systems, the proposed interconnection network provides cache coherence and synchronization primitives. Because of its generality, the MESI cache model [30] was chosen for maintaining cache consistency. All cache operations performed on the backplane are specified relative to this model. The protocol supports both write-through and copy-back caches.

To support processor synchronization, several hardware primitives are provided within the bus protocol. Read and write operations can be locked, both temporarily and permanently. Also included are *swap, compare-and-swap, fetch-and-add,* and *compare-and-add* commands. The backplane signals and commands for supporting cache coherency and synchronization are discussed in Chapter 5.

Bridging to Other Buses

An important aspect of the architecture is its ability to support existing bus standards. This can be accomplished with a *bridge*, as illustrated in Figure 4.4.

The bridge and connector can be mounted directly on the backplane, with each slot configured for a particular board type. However, as needs change, this method is not very flexible

As an alternative, the bridge could be mounted on an adaptor card between the backplane and plug-in module. This has the advantage of allowing any slot to be configured for any type of board.

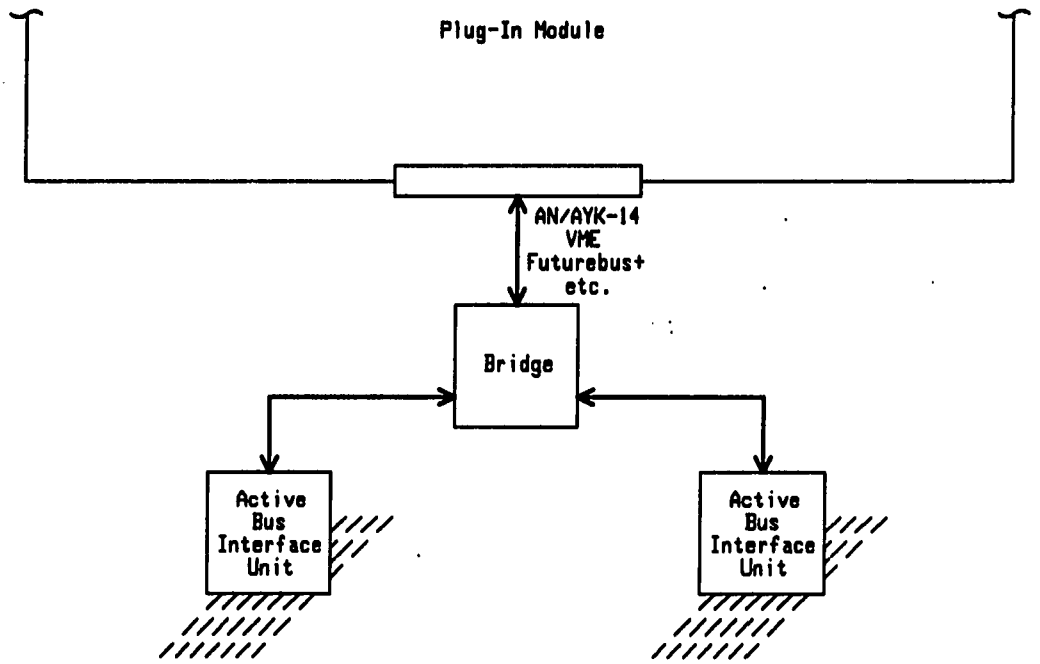


Figure 4.4: Block diagram of a bus bridge

CHAPTER 5. PROTOCOL DESCRIPTION

This chapter provides a detailed specification of the active bus protocol used between ABIUs on the backplane. The bus signals are described, along with the control acquisition scheme and various bus transactions. While several aspects of the ABIU-to-module interface are discussed, the complete specification of that interface is left for future work.

To achieve fault-tolerance, each bus operates synchronously, but independently. Therefore, the protocol is presented from a single-bus perspective, with multiple-bus features discussed where appropriate. A multiple-bus system can be implemented by simply combining two of more of the individual buses.

Active Bus Lines

This section describes the signaling environment of the backplane. The active bus signal lines are defined and the system clocking mechanism presented.

Signaling Environment

As previously discussed, the bus transceivers are mounted directly on the backplane. This reduces the load capacitance and increases the characteristic impedance of the signal lines. Initial transceiver designs have reduced the capacitive load to

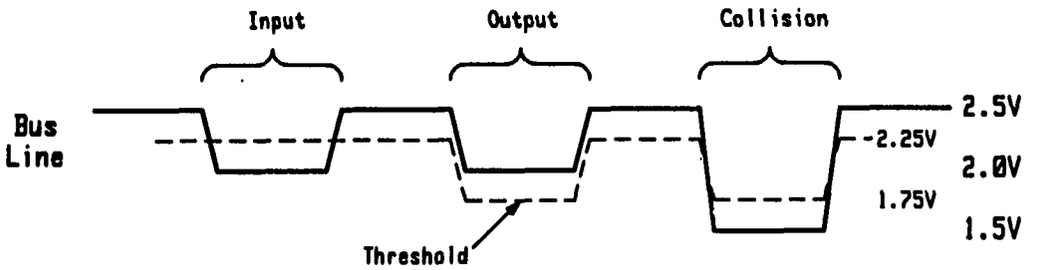


Figure 5.1: Active bus signal levels

3 pF/slot, resulting in a characteristic bus impedance of 50 Ω. The bus transceiver and backplane prototype will be discussed in Chapter 6.

All bus lines are wired-or and active-low, with open-drain bus drivers. The signal levels are illustrated in Figure 5.1. When all bus drivers have released a signal line, it rises to the bus termination voltage of 2.5 V, which represents a logic zero.

To further decrease the drive current requirements, a reduced signal voltage swing of 0.5 V is used. The lower current and voltage swings lead to reduced signal crosstalk. However, because of the reduced noise margins, the use of a ground plane plus alternating signal and ground lines is recommended.

When asserted, each ABIU driver can sink 20 mA constant current from the bus. Each output must drive two 50 Ω lines in parallel (one in each direction), or 25 Ω. Therefore, the 20 mA current lowers the bus voltage by 0.5 V. Since the currents are additive, each driver activated on a line reduces the voltage by an additional 0.5 V (limited by driver saturation as the line voltage approaches 0 V).

Table 5.1: Active bus signal lines

Mnemonic	Description	Number
AD[31:0]	Address/Data	32
CB[6:0]	Check-Bits (SEC-DED)	7
ST	STrobe	1
BB	Bus-Busy	1
AK	AcKnowledge	1
AI	Acknowledge Inverse	1
IV	InterVention	1
FL	FLag	1
CL	CLock	1
RE	REset	1
Total Active Signals		**47**

The input stage of the bus transceiver utilizes a differential amplifier with a threshold of 2.25 V. A line voltage greater than this denotes a logic zero, while a value less than this denotes a logic one. By reducing the threshold to 1.75 V, a collision can be detected on a bus line. The bus transceiver presented in Chapter 6 provides this capability.

Bus Signal Definitions

Each bus consists of 47 active signal lines, as defined in Table 5.1. With the exception of the clock signal, all bus lines use the wired-or bus transceiver. In addition, the bus-busy and flag signals utilize the collision detection capability.

The bus is centered around a 32-bit data path (**AD[31:0]**), with an additional seven check-bits (**CB[6:0]**) provided for SEC-DED. Data transfers are signaled by a source-synchronous strobe (**ST**). All transactions begin with the assertion of the master bus-busy signal (**BB**), which is used to detect a bus collision. Handshaking is provided by the slave acknowledge signals (**AK,AI**).

The intervention (**IV**) line is used during cache transactions. By asserting **IV**, a cache can intervene for main memory to supply a requested block. This is used when the cache block is in the *modified* state and main memory is out-of-data.

The flag (**FL**) line is used for various transaction types. For basic read and write transactions, a slave asserts **FL** to indicate the end of data. During synchronization transactions, a slave asserts **FL** according to the outcome of any comparison, such as from a compare-and-swap.

The collision detection capability of the **FL** line is used in cache transactions. Any module keeping a copy of the current block being transferred asserts **FL**. If a collision occurs, the block is placed in the *shared* state. Otherwise, it is place in the *exclusive* state.

All bus timing is provided by a global system clock (**CL**). This is a special input-only signal to the ABIUs. The reset (**RE**) signal is used to initialize or reset a bus.

Clocking

The backplane is based on a synchronous, deskewed clocking scheme. To avoid the delays associated with metastability, the synchronization domain is extended from the backplane, through the module interface, and onto the plug-in modules. This allows synchronization to be done at the packet level, rather than on individual bus transitions. All clock signals are deskewed, such that clock edges are received simultaneously at each ABIU.

Two clock signals are superimposed on the **CL** line. This is accomplished as shown in Figure 5.2. The higher-rate clock (**M_CLK**) is used for the source-

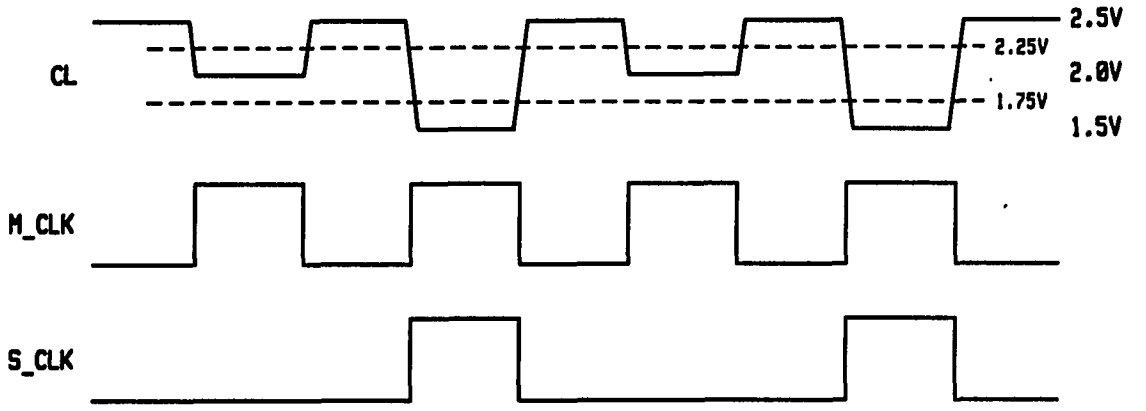


Figure 5.2: Clock timing

synchronous streaming of data on the bus. The frequency of **M_CLK** is determined by the maximum possible data transmission rate on the bus, limited only by signal skew.

The second clock (**S_CLK**) is used for all bus transitions which depend on the round-trip propagation delay of the backplane. This includes the start of a bus cycle, the sampling of collision detection signals, and the assertion of acknowledge signals.

The frequency of **S_CLK** can be any fraction of **M_CLK**, including the same frequency, one-half, one-third, etc. The example of Figure 5.2 illustrates an **S_CLK** frequency one-half that of **M_CLK**. The frequency of **S_CLK** will vary, depending on the length of the backplane being implemented.

The clock transceiver can be implemented in the same fashion as the standard transceiver. Since this is an input-only signal, the output driver can be replaced with a second differential amplifier input stage. The two CMOS core logic signals, **M_CLK** and **S_CLK**, can then be generated with 2.25 V and 1.75 V thresholds, respectively.

Geographical Address

Each slot on the backplane will be assigned a unique, eight-bit, geographical address (**GA[7:0]**). This allows up to 256 slots on a single backplane. The address can be assigned by tying the appropriate lines to 5 V and ground. The numbering on all backplanes should start at zero. The **GA[]** lines will be available to the module and the ABIUs at each slot.

Control Acquisition

The process by which a requesting master gains the exclusive right to use a bus is known as *control acquisition.* A *collision* occurs when more than one master requests the use of a bus at the same time. An *arbitration* mechanism can be used to decide which master will gain access to the bus.

Collision detection access schemes, such as *CSMA/CD,* provide fast and efficient access when a system is lightly loaded. However, under heavy loads, the effective bandwidth of this type of system can drop dramatically. Priority arbitration schemes [40, 43] provide a deterministic means of choosing a winner, even under heavy loads. However, these require at least three round-trip bus propagation delays before the winner is elected [44].

A control acquisition scheme, which combines collision detection and priority arbitration, is presented in this section. The scheme minimizes bus access latency while guaranteeing the deterministic election of a master. It also incorporates a fairness mechanism to prevent starvation.

Selection of a Bus

Each requesting module must decide on which bus to attempt a transaction. The algorithm for doing this is not specified in this dissertation. This allows the module designer freedom in making cost versus performance tradeoff decisions.

The primary goal of the selection algorithm should be the equal distribution of requests to all buses. This maximizes the throughput of the interconnect. One method of accomplishing this involves a modulo-B primary bus assignment, where B is the number of buses. This method is described in [3].

Control Acquisition Scheme

Once the bus selection has been made, the control acquisition process is initiated. A state diagram of the proposed algorithm is illustrated in Figure 5.3, followed by bus timing examples in Figure 5.4.

The ABIU waits for the release of **BB**, which signifies the end of the current transaction. Once the bus is idle, the ABIU waits either one or two **S_CLK** cycles before beginning a transaction.

The fairness mode is implemented by allowing any ABIU which lost the previous arbitration to begin transmitting after one **S_CLK** cycle (Figures 5.4a and b). This group is restricted to those requests with the same priority as the winner. If there are none, then the others begin transmitting after two cycles (Figure 5.4c). A higher priority request can override the fairness mode by waiting only one **S_CLK** cycle.

An ABIU begins a transaction by applying a control word to **AD[]** and asserting **BB**. After one **S_CLK** cycle, the **BB** collision detection input is sampled. If no collision occurs, then the ABIU continues with the transaction (Figure 5.4a).

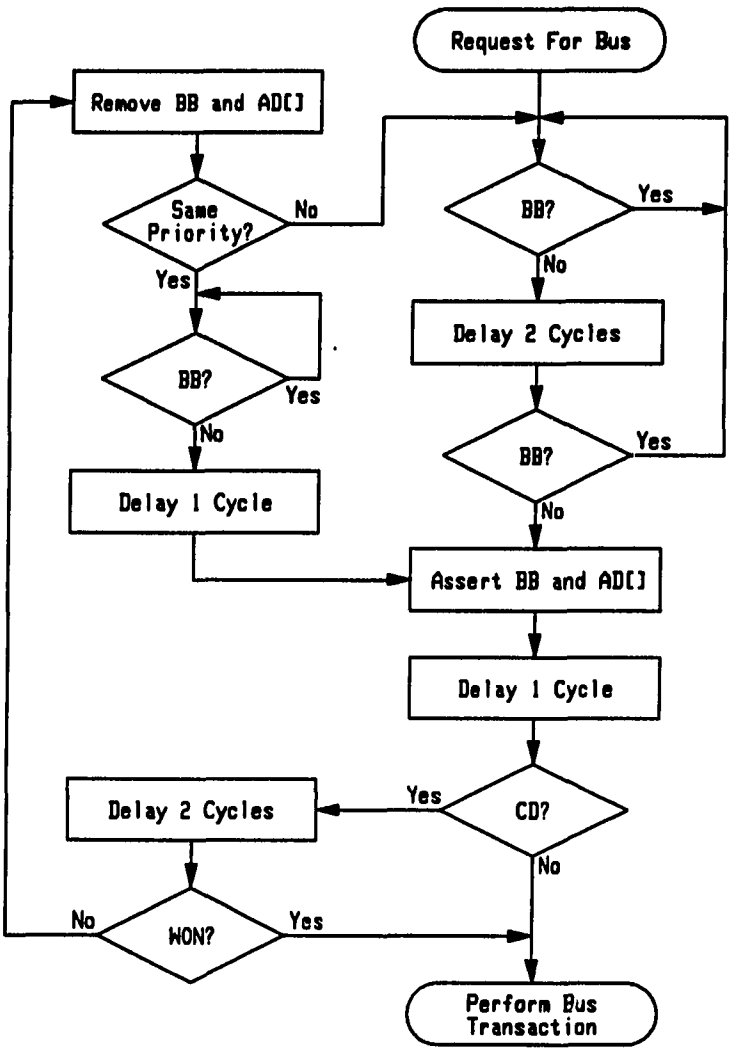
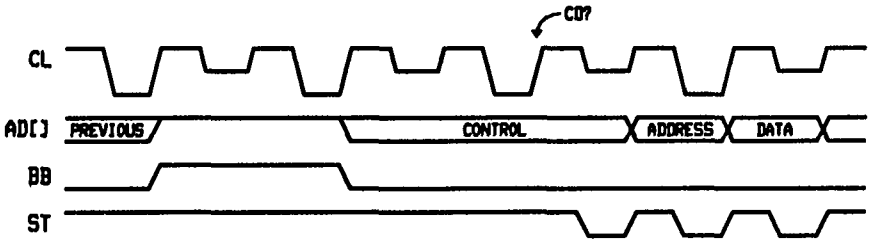
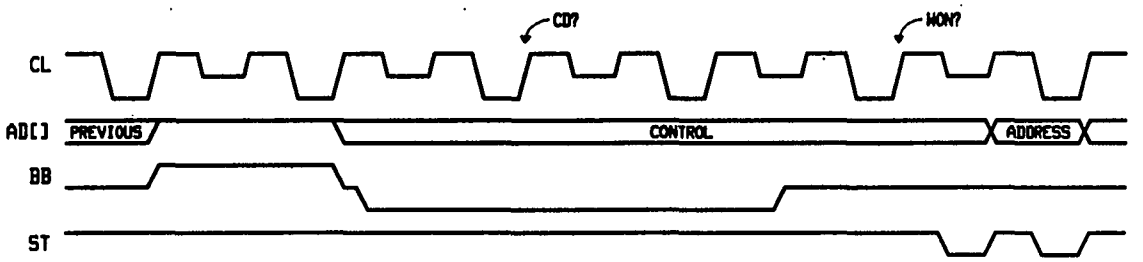


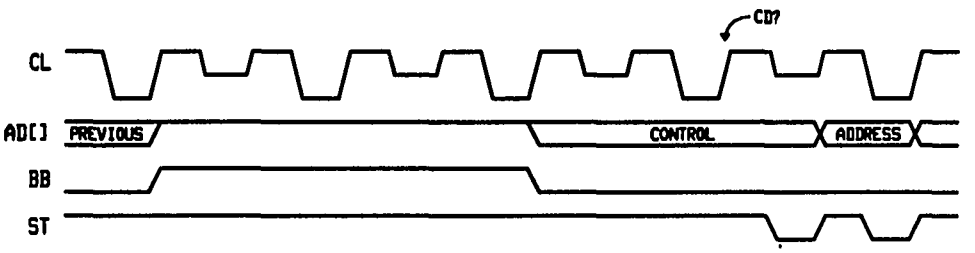
Figure 5.3: State diagram of the control acquisition scheme



a) without collision



b) with collision



c) with fairness delay

Figure 5.4: Control acquisition timing examples

If a collision is detected, then the priority arbitration algorithm is completed to determine the next master. This is done via the high-order sixteen bits of the data path (**AD[31:16]**), which incorporate distributed priority arbitration logic similar to that used in the Futurebus+ standard [29]. The prototype priority arbitration circuit is presented in Chapter 6.

The high-order sixteen bits of the control word represent the priority (**PR[7:0]**) and source-address (**SA[7:0]**) of the current bus request. Higher **PR[]** values have higher priority, while the **SA[]** field is used to resolve requests at the same priority level. The **SA[]** field must be unique for each slot on the backplane. This can be done by using the geographical address (**GA[]**).

The priority arbitration circuit requires three round-trip propagation delays to determine a winner. Since the priority was placed on the line when **BB** was asserted, the first cycle overlaps with the cycle required for collision detection. Therefore, only two more **S_CLK** cycles are required (Figure 5.4b) before the **WON** signal is sampled. The winning ABIU continues its transaction, while the losers remove their control word and release **BB**.

When no collision occurs, this protocol provides an efficient method of accessing the bus. When a collision does occur, the priority arbitration circuit guarantees the deterministic election of a master. By partially overlapping this with collision detection, the arbitration overhead is reduced.

The fairness mode can be utilized to prevent the starvation of bus requests. Also, since arbitration is performed on the data path, additional signal lines are not required. This keeps the total number of signal lines low.

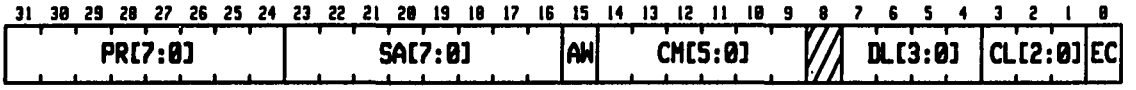


Figure 5.5: Control word format

Data Transfer

This section describes the data transfer protocol. The control word fields are presented and the various transaction types are discussed.

Control Word

For all transactions, the first word transferred is the control word. The format of the control word is shown in Figure 5.5. As previously discussed, the priority (**PR[7:0]**) and source address (**SA[7:0]**) fields are used in the control acquisition protocol.

The address width (**AW**) field specifies whether 32-bit or 64-bit addressing is to be used. The type of transaction to be performed is specified by the command (**CM[5:0]**) field. The data length (**DL[3:0]**) field specifies the length of the data for the current bus transaction. For cache transactions, **DL** specifies the line size for the current operation.

The class (**CL[2:0]**) field allows modules to be grouped into classes, such that transactions in an unsupported class are ignored. The extended control (**EC**) field specifies whether a second control word will be transferred. This is reserved for future extensions to the protocol. Currently, all modules should ignore any transactions with this bit set.

Table 5.2: Address width encoding

AW	Address Width
0	32-bit
1	64-bit

Addressing

The protocol is based on a byte-addressable, memory-mapped model. Each transaction is specified with an address, which immediately follows the control word. As specified by **AW**, two address widths are supported. The encoding of this field is given in Table 5.2. For 64-bit addresses, the high-order bits (**A[63:32]**) are transferred first, followed by the low-order bits (**A[31:0]**).

All modules must support 32-bit addressing, with 64-bit addressing support being optional. Any module not supporting 64-bit addressing will ignore such transactions. The 32-bit address space is a subset of the 64-bit address space. All accesses to this space should be made using 32-bit addressing, even if 64-bit addressing is supported.

Transaction Types

A robust set of transaction types are supported within the bus protocol. These are specified by **CM[5:0]** of the control word. The encoding of this field is given in Table 5.3. Several entries in the command field are reserved for future enhancements to the protocol.

Basic Transactions The first sixteen entries in the table are read and write transactions. Various modifications to the read and write operations are supported.

Table 5.3: Encoding of commands

CM[5:0]	Command	CM[5:0]	Command
0 0 0 0 0 0	Read	1 0 0 0 0 0	Read Invalid
0 0 0 0 0 1	Write	1 0 0 0 0 1	Write Invalid
0 0 0 0 1 0	Read Response	1 0 0 0 1 0	Read Shared
0 0 0 0 1 1	Write Response	1 0 0 0 1 1	Copyback
0 0 0 1 0 0	Read/Lock/Temp	1 0 0 1 0 0	⟨Reserved⟩
0 0 0 1 0 1	Write/Lock/Temp	1 0 0 1 0 1	⟨Reserved⟩
0 0 0 1 1 0	Read/Lock/Perm	1 0 0 1 1 0	Read Modified
0 0 0 1 1 1	Write/Lock/Perm	1 0 0 1 1 1	Invalidate
0 0 1 0 0 0	Read/Non-Split	1 0 1 0 0 0	⟨Reserved⟩
0 0 1 0 0 1	Write/Non-Split	1 0 1 0 0 1	Shared Resp.
0 0 1 0 1 0	⟨Reserved⟩	1 0 1 0 1 0	⟨Reserved⟩
0 0 1 0 1 1	⟨Reserved⟩	1 0 1 0 1 1	⟨Reserved⟩
0 0 1 1 0 0	Read/N-S/Lk/Tmp	1 0 1 1 0 0	⟨Reserved⟩
0 0 1 1 0 1	Write/N-S/Lk/Tmp	1 0 1 1 0 1	⟨Reserved⟩
0 0 1 1 1 0	Read/N-S/Lk/Prm	1 0 1 1 1 0	⟨Reserved⟩
0 0 1 1 1 1	Write/N-S/Lk/Prm	1 0 1 1 1 1	Modified Resp.
0 1 0 0 0 0	Swap	1 1 0 0 0 0	Event
0 1 0 0 0 1	Compare-and-swap	1 1 0 0 0 1	⟨Reserved⟩
0 1 0 0 1 0	Fetch-and-add	1 1 0 0 1 0	⟨Reserved⟩
0 1 0 0 1 1	Compare-and-add	1 1 0 0 1 1	Write No Ack.
0 1 0 1 0 0	⟨Reserved⟩	1 1 0 1 0 0	⟨Reserved⟩
0 1 0 1 0 1	⟨Reserved⟩	1 1 0 1 0 1	⟨Reserved⟩
0 1 0 1 1 0	⟨Reserved⟩	1 1 0 1 1 0	⟨Reserved⟩
0 1 0 1 1 1	⟨Reserved⟩	1 1 0 1 1 1	⟨Reserved⟩
0 1 1 0 0 0	⟨Reserved⟩	1 1 1 0 0 0	⟨User Defined⟩
0 1 1 0 0 1	⟨Reserved⟩	1 1 1 0 0 1	⟨User Defined⟩
0 1 1 0 1 0	⟨Reserved⟩	1 1 1 0 1 0	⟨User Defined⟩
0 1 1 0 1 1	⟨Reserved⟩	1 1 1 0 1 1	⟨User Defined⟩
0 1 1 1 0 0	⟨Reserved⟩	1 1 1 1 0 0	⟨User Defined⟩
0 1 1 1 0 1	⟨Reserved⟩	1 1 1 1 0 1	⟨User Defined⟩
0 1 1 1 1 0	⟨Reserved⟩	1 1 1 1 1 0	⟨User Defined⟩
0 1 1 1 1 1	⟨Reserved⟩	1 1 1 1 1 1	⟨User Defined⟩

First, the operations can be *split.* In a split transaction, the master sends the control and address (and data in the case of a write) to the slave. If the slave cannot respond immediately, it requests a split transaction and terminates the current cycle. When it is ready to respond, the slave arbitrates for the bus and sends back a read or write response (plus data in the case of a read response). In a multiple-bus system, the slave does not have to respond on the same bus as the original request.

Operations are also available to *lock* the slave resource, either *temporarily* or *permanently.* In the case of a temporary lock, when the transaction completes, the lock is released. The permanent lock is held past the end of the transaction. An unlocked or temporarily locked transaction from the same master is required to unlock the resource. The slave responds with a busy status when a transaction to a locked resource is requested by another master.

When a master wishes to hold the bus until a transaction is completed, it uses a *non-splittable* read or write. These can be used where the master cannot wait for the slave to arbitrate for a bus on which to respond.

Synchronization Transactions The second group of sixteen entries are synchronization commands. These include *Swap, Compare-and-swap, Fetch-and-add,* and *Compare-and-add* commands. Other synchronization operations are under consideration for incorporation into the bus protocol.

Except for the compare-and-add command, these transactions operate the same as the Futurebus+ transactions of the same name [29]. The compare-and-add, however, requires three operands to be transferred. First the compare operand and add operand are transferred to the slave. If the compare operand is not equal to the data

stored in memory, then the add operand is added to the memory data, and the result transferred to the master and stored in memory. If the compare operand equals the data stored in memory, then the memory value is transferred to the master, without performing the add.

For synchronization operations, the **FL** line is driven according to the result of any comparison done in the transaction. The master can sample **FL** at the end of the transaction to determine the outcome. For example, if the comparison in the compare-and-swap command succeeds, the slave returns the original memory value and asserts **FL**. For unconditional synchronization operations, such as the swap command, the **FL** line is always asserted.

Cache Transactions The third group of sixteen commands in Table 5.3 are cache commands. All operations to cache coherent memory should be done using these commands. To ease the bridging to Futurebus+ systems, the commands are very similar to the cache commands of the Futurebus+ specification [25].

The *Read Invalid* and *Write Invalid* commands are used by non-caching modules to access cache coherent memory. The master does not keep a copy of the block after the completion of these commands.

The *Read Shared* command is used by a cache to read a block which is not intended to be modified. The module asserts the **FL** line, and places the block in the *exclusive* or *shared* state, depending on whether a collision occurs on **FL**.

The *Copyback* command is used to flush a *modified* block by writing it back to main memory. Any caches wishing to snarf a copy of the block assert **FL**, and place the block in the appropriate state.

The *Read Modified* command is similar to read shared, except that the block is intended for modification. The module asserts the **FL** line, and if a collision occurs, another module is still using the block. The cache must then place the block in the *shared* state until an *Invalidation* operation can be performed. The invalidation command transfers no data, but simply invalidates all other copies of a shared cache block.

The *Read Response* and *Modified Response* commands are used for responding to read shared and read modified transactions, respectively, which may have previously been split.

Miscellaneous Transactions The final group of sixteen commands are for miscellaneous transactions. These include the *Event* and *Write No Acknowledge* commands, plus several user definable commands.

The event command is for sending processor events and interrupts. This can be used as an address-only event, event with data, vectored interrupt, etc, depending on the requirements of the modules.

Data Length

The length of the data for the current operation is specified by **DL**. This field is interpreted differently during various transaction types.

For basic read and write transactions, this field specifies the length of the data to be transferred. The encoding of the field for basic transactions is summarized in the *Basic Length* column of Table 5.4. This can range from an address-only cycle to a 2048 word (8192 byte) packet.

Table 5.4: Data length encoding

DL3	DL2	DL1	DL0	Data Length	Sync Length	Line Size
0	0	0	0	0	—	—
0	0	0	1	1 Byte	1 Byte	—
0	0	1	0	2 Byte	2 Byte	—
0	0	1	1	1 Word	1 Word	1 Word
0	1	0	0	2 Word	2 Word	2 Word
0	1	0	1	4 Word	4 Word	4 Word
0	1	1	0	8 Word	8 Word	8 Word
0	1	1	1	16 Word	—	16 Word
1	0	0	0	32 Word	—	32 Word
1	0	0	1	64 Word	—	64 Word
1	0	1	0	128 Word	—	128 Word
1	0	1	1	256 Word	—	256 Word
1	1	0	0	512 Word	—	—
1	1	0	1	1024 Word	—	—
1	1	1	0	2048 Word	—	—
1	1	1	1	Signaled by **FL**	—	—

For data lengths which are unknown ahead of time or not a power of two, the **FL** signal can be used to specify end-of-data during a transaction. This is also useful when a request crosses the boundary of a slave's address spaces.

For synchronization transactions, **DL** specifies the size of the synchronization variable. The encoding for these transactions is shown in the *Sync Length* column of Table 5.4.

During cache transactions, the line size for the current cache operation is specified by **DL**. Legal values for cache operations are given in the *Line Size* column of Table 5.4. This value does not necessarily specify the amount of data to be transferred. For instance, the *Invalidate* transaction is an address only transaction operates on a cache line of size **DL**.

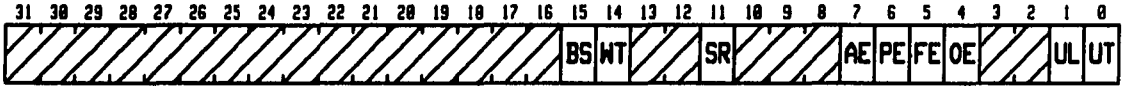


Figure 5.6: Status word format

A caching module only responds to transactions of the same line size. Rather than specifying a fixed block size for all caches in the system, this allows multiple block sizes to coexist in the same card cage. Cache coherency will not, however, be maintained between caches of differing line sizes.

Status Word Format

For transactions which terminate normally, no status is sent back to the master (a "no news is good news" philosophy). Any slave which detects a protocol error or wishes to terminate the current transaction can do so by pulling the **AI** line before releasing the **AK** line. If this occurs, the slave places a status word on the bus to be read by the master.

The format of the status word is shown in Figure 5.6. The busy (**BS**) and wait (**WT**) fields indicate resources which are busy for the short and long term, respectively. A retry can be generated immediately after a **BS** response, while a master should delay in the case of a **WT**. If a slave wishes to split the current transaction, it terminates the transaction with a status response and asserts the split response (**SR**) flag.

Error conditions include arbitration error (**AE**), uncorrectable SEC-DED error (**PE**), framing error (**FE**), and overrun error (**OE**). Other errors related to unim-

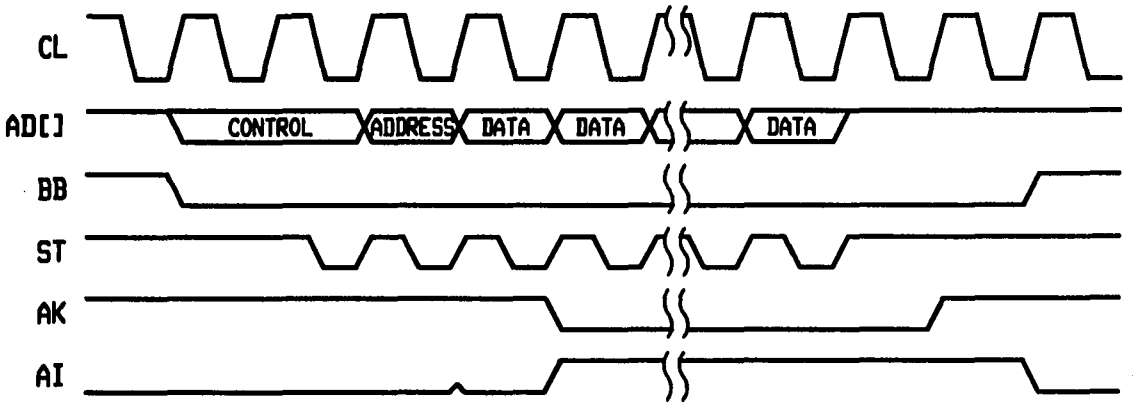


Figure 5.8: Write transaction timing

The master releases the **AD[]** and **ST** lines and waits for the slave to respond with the requested data. If a slave requests a split response, the current transaction ends with a status termination, and the slave would respond later. Otherwise, the slave transfers the data back to the master via **AD[]** and **ST**.

The transaction terminates with the addressed slave(s) releasing **AK**, followed by all slaves asserting **AI**. Once the master detects the release of **AK**, it releases **BB**, signifying the end of its bus tenure. If any slave wishes to respond with a status condition, it asserts **AI** before releasing **AK**. An example of a status terminated transaction is presented later.

A write transaction is illustrated in Figure 5.8. The transaction begins in the same manner as the read. However, rather than stopping after sending the control and address, the master continues transmitting the data. The transaction ends with the release of **AK**, the assertion of **AI**, and then the release of **BB**. The write transaction can terminate with a status response in the same fashion as the read.

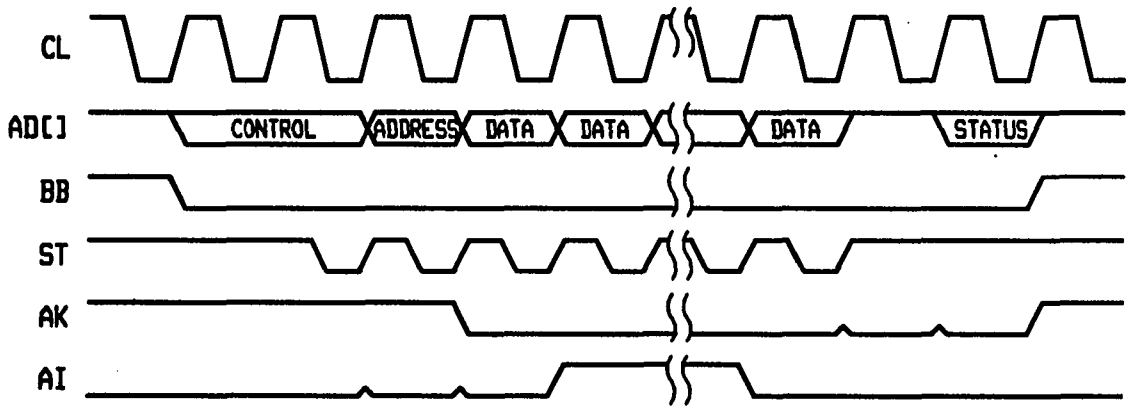


Figure 5.9: Status terminated transaction

Finally, Figure 5.9 illustrates a write transaction which terminates with a status condition. This is denoted by the assertion of **AI** before the release of **AK**. When the slave(s) has placed its status word on **AD[]**, it releases **AD**, and the transaction terminates.

CHAPTER 6. RESULTS

Prototype bus transceiver and priority arbitration circuits have been designed and fabricated in 2-μm CMOS. The results of transceiver, backplane, and arbitration simulations and prototype tests are presented in this chapter. Source listings for all SPICE simulations are given in the Appendix.

Active Bus Transceiver

A bus transceiver was designed to implement the active bus signaling protocol of Chapter 5. The design, layout, and simulation of the transceiver are presented here. Prototype test results are presented later in this chapter.

Design

By using surface mount packaging, such as the *quad flat-pack* or *leaded chip carrier*, preliminary studies show that the load capacitance can be kept to 3 pF/slot. This includes approximately 1 pF for the driver and bonding pad, plus another 2 pF for the package. Thus, from Table 2.1, the transceiver was designed for driving a 50 Ω signal line.

A 20 mA output driver and differential input stage are used for generating and detecting a 0.5 V logic swing on the bus lines. Also included is a private voltage

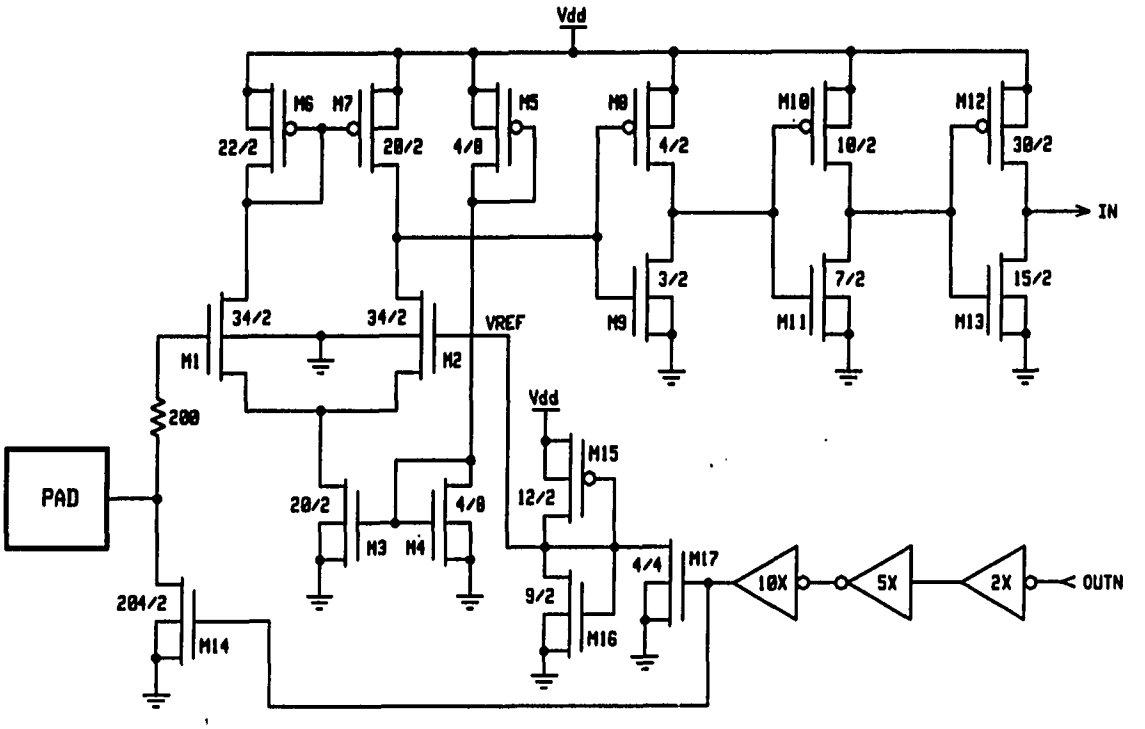


Figure 6.1: Bus transceiver schematic

reference circuit used to adjust the threshold of the differential amplifier for collision detection. The transceiver schematic is shown in Figure 6.1.

The transceiver is designed as a 2-μm CMOS pad for use in a standard padframe. The core logic signals (**IN, OUTN**) are shown to the right, while the bonding pad connection (**PAD**) is shown to the left. The top portion of the schematic represents the input stage, while the output stage and private voltage reference circuit are shown toward the bottom.

The **OUTN** signal is an active-low output enable. Three inverters are used to step the signal level up before reaching the driver. A $204\mu/2\mu$ NMOS driver is used to sink 20 mA from the bus.

The voltage divider generates a private voltage reference (**VREF**) of 2.25 V. This is used as the threshold for the differential amplifier. To adjust the threshold for collision detection, **OUTN** is used to lower **VREF** by 0.5 V when the output is enabled (see Figure 5.1).

The input stage consists of a CMOS differential amplifier, followed by three inverters to square and step-up the signal. The two inputs to the amplifier come from the bus and the private voltage reference. When the bus voltage is higher than the reference voltage, **IN** will be low (logic 0). When the bus voltage drops below the reference, **IN** will be high (logic 1).

Since the private reference is lowered when the output is asserted, an ABIU will never see its own signal on **IN**. In addition to its use for collision detection in the active bus protocol, this feature has the potential to be used for bidirectional data transfer on a single set of lines.

Input ESD protection is provided by a 200 Ω n-diffused resistor between the pad and input stage. Also, a diode clamp to ground is provided by the drain diffusion of the $204\mu/2\mu$ NMOS driver.

Layout

The active bus transceiver pad was implemented in VLSI Technology's 2-μm CMN20 design rules. The layout cell is shown in Figure 6.2. For fabrication within the MOSIS TinyChip program, the layout cell conforms to the standard pad abutment box of $200\mu \times 210\mu$.

The core input and output connections are located at the top right and left, respectively. The output stage is located to the left, while the differential input stage

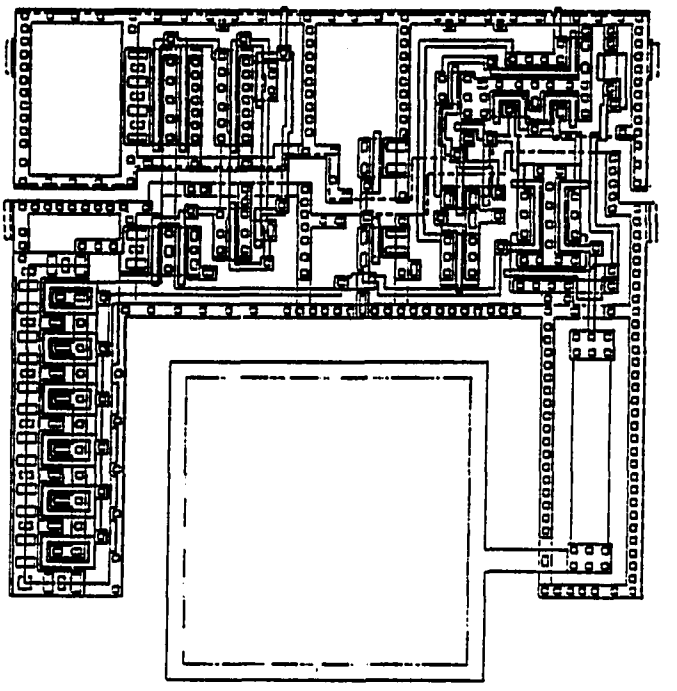


Figure 6.2: CMOS transceiver pad layout

is shown to the right. The private voltage reference is located in the top-center. The bonding pad, shown at the bottom, is $100\mu \times 100\mu$.

Simulation

The transceiver was simulated with SPICE before fabrication. All simulations were performed using worst-case CMN20 transistor models. The SPICE subcircuit for the transceiver (**XCVR**) was created using VLSI Technology's netlist extractor. All stray capacitances were included.

The output simulation is shown in Figure 6.3. The **PAD** is tied to 2.5 V through a 25 Ω resistor to simulate the driving of a bus line.

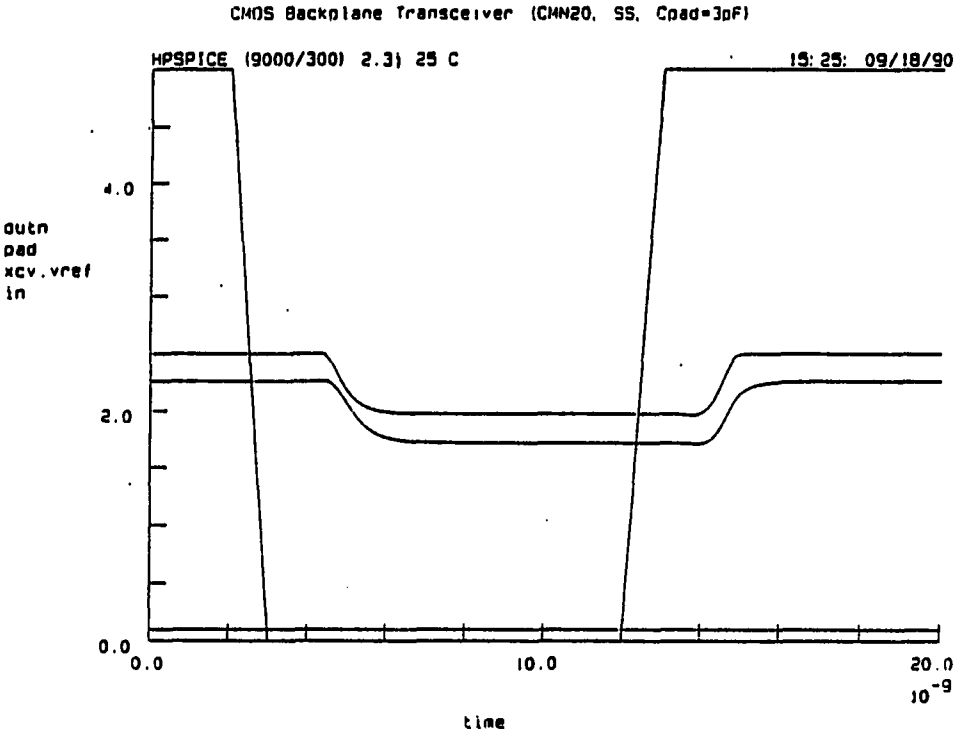


Figure 6.3: Bus transceiver output simulation

A 20 ns pulse is applied to **OUTN**. This, in turn, lowers the **PAD** voltage by 0.5 V. The transceiver's private voltage reference (**XCV.VREF**) is also lowered by 0.5 V. As previously discussed, the bus signal is not seen on **IN** due to the reduced threshold. The rise and fall delay from **OUTN** to **PAD** are 2.03 ns and 2.36 ns, respectively.

The transceiver input simulation is illustrated in Figure 6.4. The **PAD** is tied to a pulsed voltage source which generates a 20 ns pulse of -0.5 V on the bus line. The **OUTN** signal is disabled (5 V) so that the output is unasserted. The delay from the falling edge of **PAD** to the rising edge of **IN** is 4.93 ns. The delay from the rising edge of **PAD** to the falling edge of **IN** is 4.42 ns.

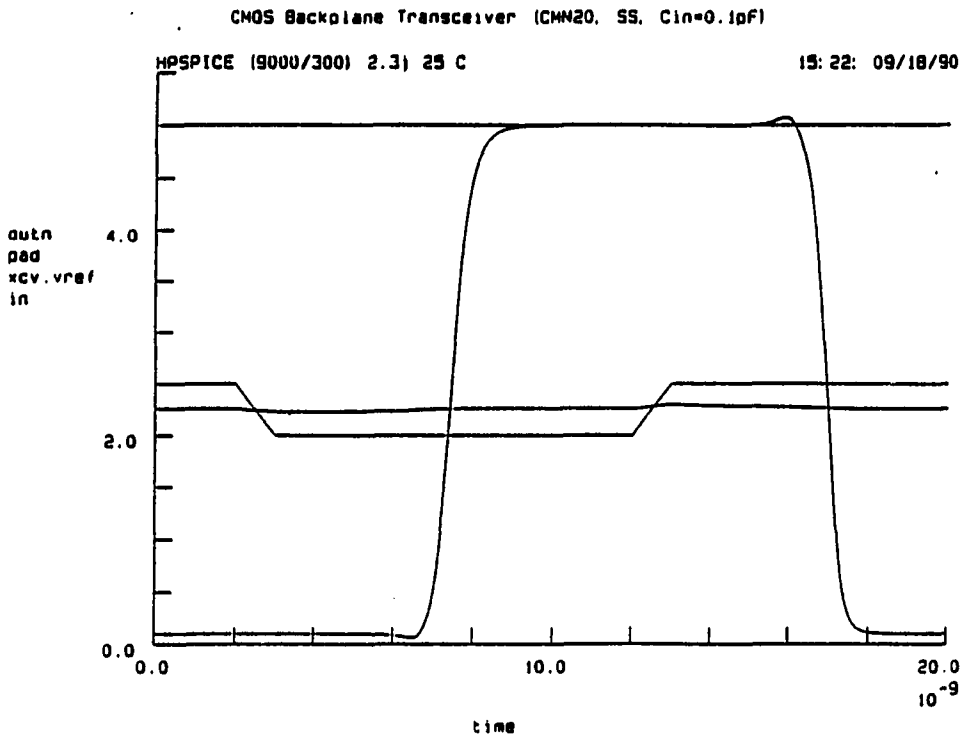


Figure 6.4: Bus transceiver input simulation

Active Backplane

This section describes the design and simulation of a 24-slot active backplane utilizing the bus transceivers just described.

Design

As previously discussed, the active bus transceiver can reduce the bus load capacitance to 3 pF/slot. This results in a characteristic bus impedance of 50 Ω. To eliminate reflections, a termination resistance of 50 Ω is required at each end of the signal line.

The simulation model for the 24-slot active backplane is shown in Figure 6.5. This model includes a bus transceiver at each end. The load capacitance at each slot is modeled by a 3 pF capacitor (C_L). The distributed capacitance (C_d) and inductance (L_d) of the signal line are 1 pF and 10 nH, respectively. The line is terminated with a 50 Ω resistor (R_t) to 2.5 V (V_t).

Simulation

Figure 6.6 illustrates the end-to-end signal delay for the 24-slot active backplane. A 20 ns output enable pulse on **OUTN1** generates a 0.5 V signal swing at **SLOT1**. The signal then propagates to the other end of the bus line (**SLOT24**), where it is converted to the core CMOS logic signal **IN24**.

The end-to-end propagation delay for the simulated backplane was measured as 4.7 ns. This is only slightly more pessimistic than the theoretical value of 4.6 ns from Table 2.1. The core-to-core signal delay from **OUTN1** to **IN24** was measured to be approximately 11 ns.

A simulation of the collision detection capability of the transceiver is illustrated in Figure 6.7. A 20 ns output enable pulse is applied simultaneously to both **OUTN1** and **OUTN24**. The outputs at **SLOT1** and **SLOT24** become asserted after one output logic delay.

The signal arrives at **SLOT1** from **SLOT24** after one end-to-end propagation delay. This reduces the bus voltage by an additional 0.5 V. Since the bus voltage drops below the differential amplifier threshold (**XCV1.VREF**), the **IN1** signal becomes asserted.

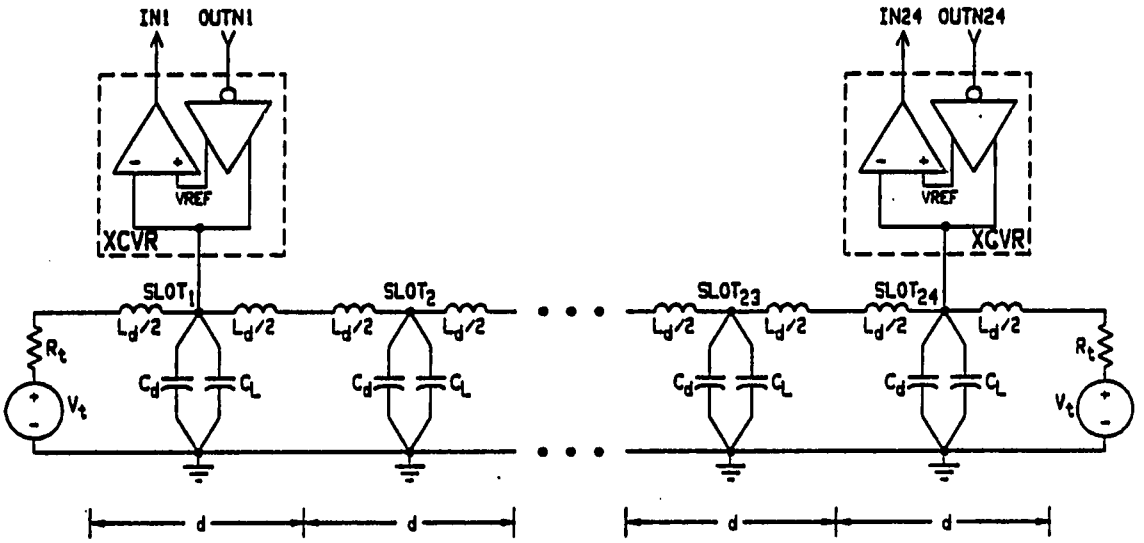


Figure 6.5: 24-Slot backplane simulation model

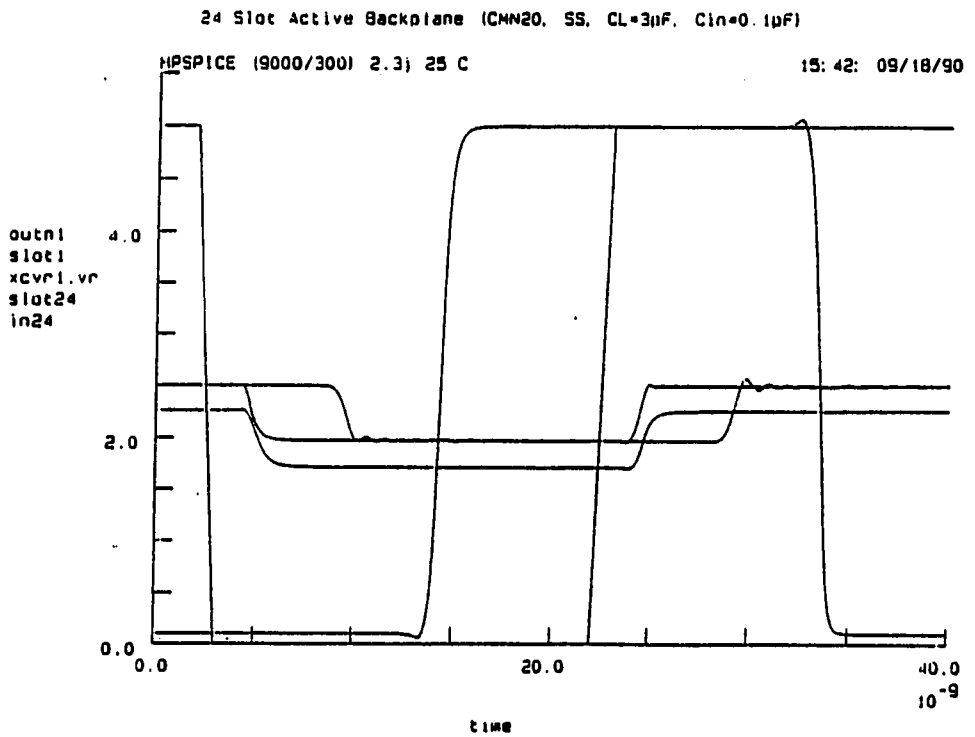


Figure 6.6: 24-slot backplane simulation

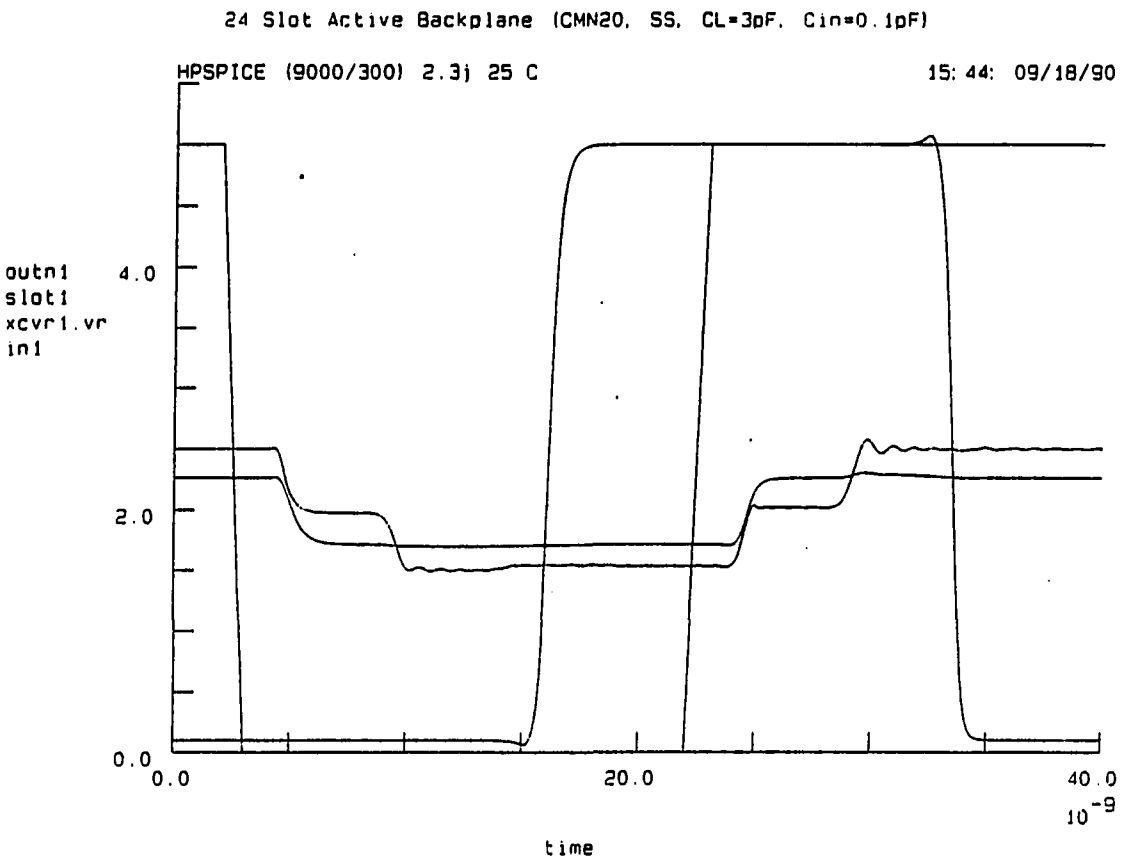


Figure 6.7: 24-slot backplane collision simulation

Priority Arbitration

The distributed priority arbitration circuit used for the bus control acquisition scheme was implemented. The design and simulation results are presented in this section.

Design

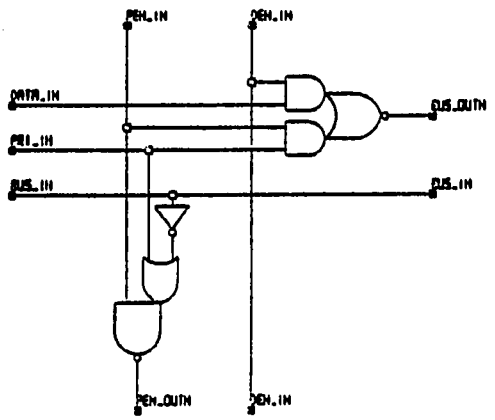
An 8-bit priority arbitration circuit, similar to that used in Futurebus+ [30, 43], was designed and fabricated. Since the signal for electing a winner must ripple through every stage, this circuit represents a critical path in the arbitration delay. Therefore, the logic delay through each stage must be kept to an absolute minimum.

To accomplish this, a single gate delay is used between each bit-slice. With inverting logic, this requires the use of alternating *odd* and *even* cells. The schematics for these are shown in Figure 6.8a and b. By using OR-AND-INVERT and AND-OR-INVERT logic, the delay through each bit-slice is kept to a single gate.

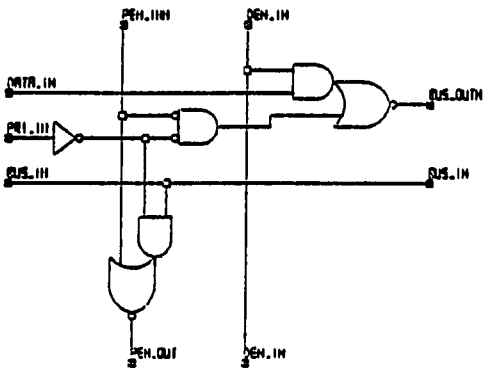
The bus interface signals, **BUS_OUTN** and **BUS_IN**, can be tied directly to the corresponding transceiver signals. Since arbitration is done over the data path, a multiplexing function between data (**DATA_IN**) and priority (**PRI_IN**) is provided.

Figure 6.8c illustrates a data enable cell used for driving up to nine of the bit-slice cells. The prototype incorporates a byte-parity cell, as shown in FIgure 6.8b. This cell passes **PEN_OUT** directly through. It can be used for those bus lines which do not participate in arbitration (**AD[15:0]**, **BD[6:0]**).

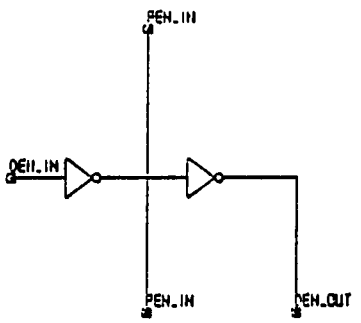
The schematic for the 8-bit arbitration circuit is illustrated in Figure 6.9. It includes a single data enable cell, eight alternating odd and even arbitration cells, and a single parity cell. The 16-bit active bus arbitration circuit can be implemented



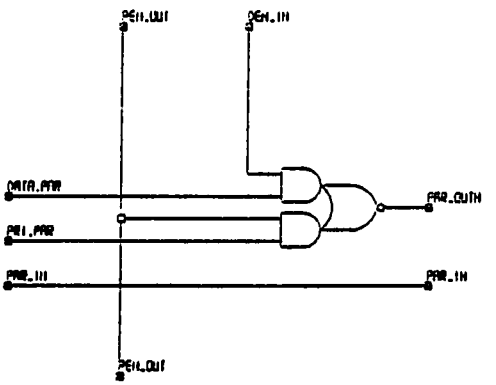
a) arb_odd



b) arb_even



c) arb_den



d) arb_par

Figure 6.8: Priority arbitration schematics

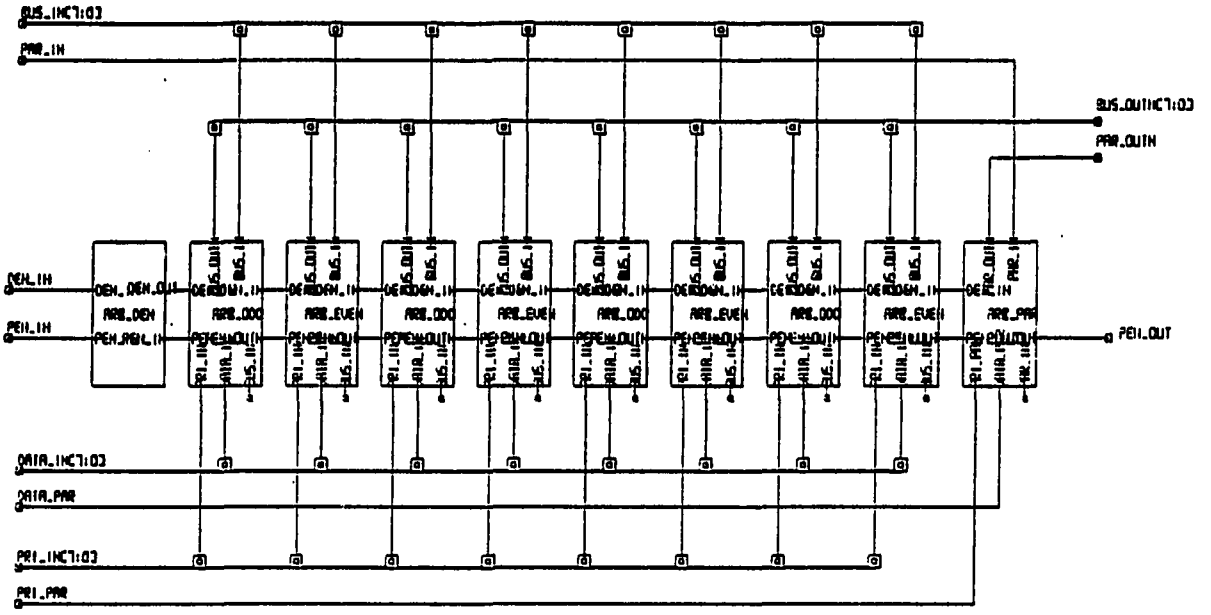
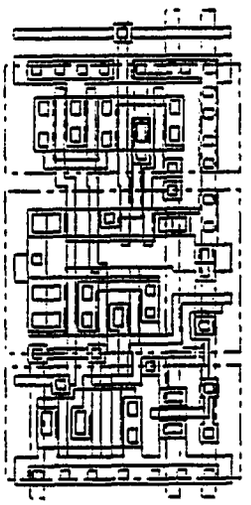


Figure 6.9: 8-Bit priority arbitration schematic

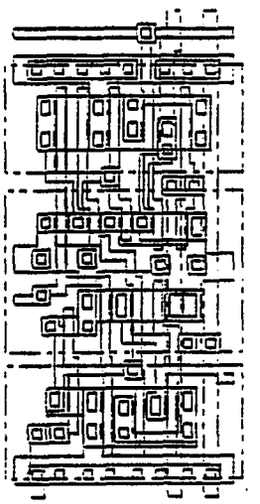
by cascading two of the 8-bit circuits (minus the parity cells). The remaining data lines can be implemented with cells similar to the parity cell.

Layout

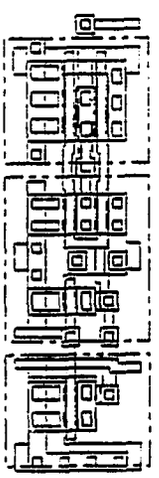
Each circuit from Figure 6.8 was implemented in the 2-μm CMN20 design rules. The corresponding layout cells are shown in Figure 6.10. The cells are designed for direct abutment. This is shown in the 8-bit arbitration circuit layout of Figure 6.11.



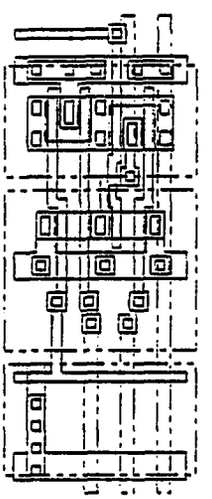
a) arb_odd



b) arb_even



c) arb_den



d) arb_par

Figure 6.10: Priority arbitration layouts

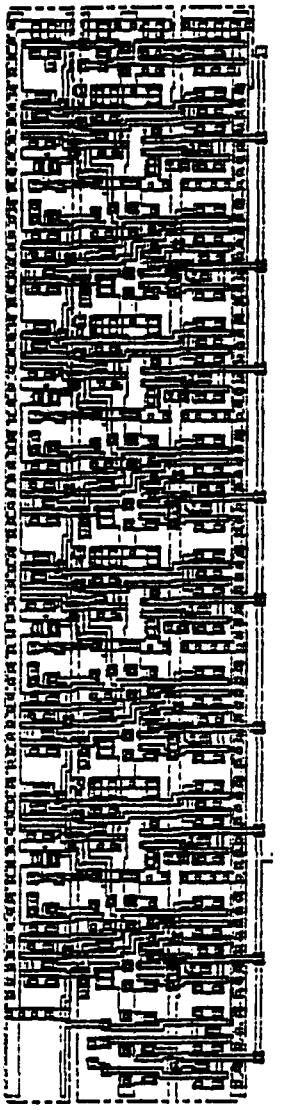


Figure 6.11: 8-Bit priority arbitration layout

Simulation

The priority arbitration circuit was simulated in SPICE, using the worst-case CMN20 transistor models. As with the bus transceiver, the layout cell was extracted to obtain the SPICE input circuit, which includes all stray capacitances. Although each cell has been simulated extensively, only the results of the 8-bit arbitration circuit simulations are presented here.

The primary measurement of interest is the ripple delay through all eight bits. The start of an arbitration cycle is signified by the assertion of **PEN_IN**. The election of a winner is then designated when **PEN_OUT** stabilizes.

The logic delay component is illustrated in Figure 6.12. For this simulation, the eight **PRI_IN** signals were tied to 5 V (logic 1). This allows the logic delay from **PEN_IN** to **PEN_OUT** to be measured. This was shown to be approximately 8.8 ns for the 8-bit circuit, or 17.6 ns when extended to a 16-bit arbitration. Simulations of a 1.2-μm CMOS implementation show a delay of 1/2 that for 2-μm implementation.

Prototype Tests

The bus transceiver and priority arbitration circuit (plus several test structures) were fabricated through the MOSIS 2-μm TinyChip program. The four packaged parts were returned 100% functional. The top-level layout for this chip is shown in Figure 6.13.

Because of MOSIS TinyChip packaging constraints, the parts were bonded in 40-pin DIP packages. Since these have a higher lead capacitance than the surface-mount devices mentioned earlier, the measurements will be somewhat pessimistic. This is further exacerbated by the oscilloscope's 10 pF probe capacitance.

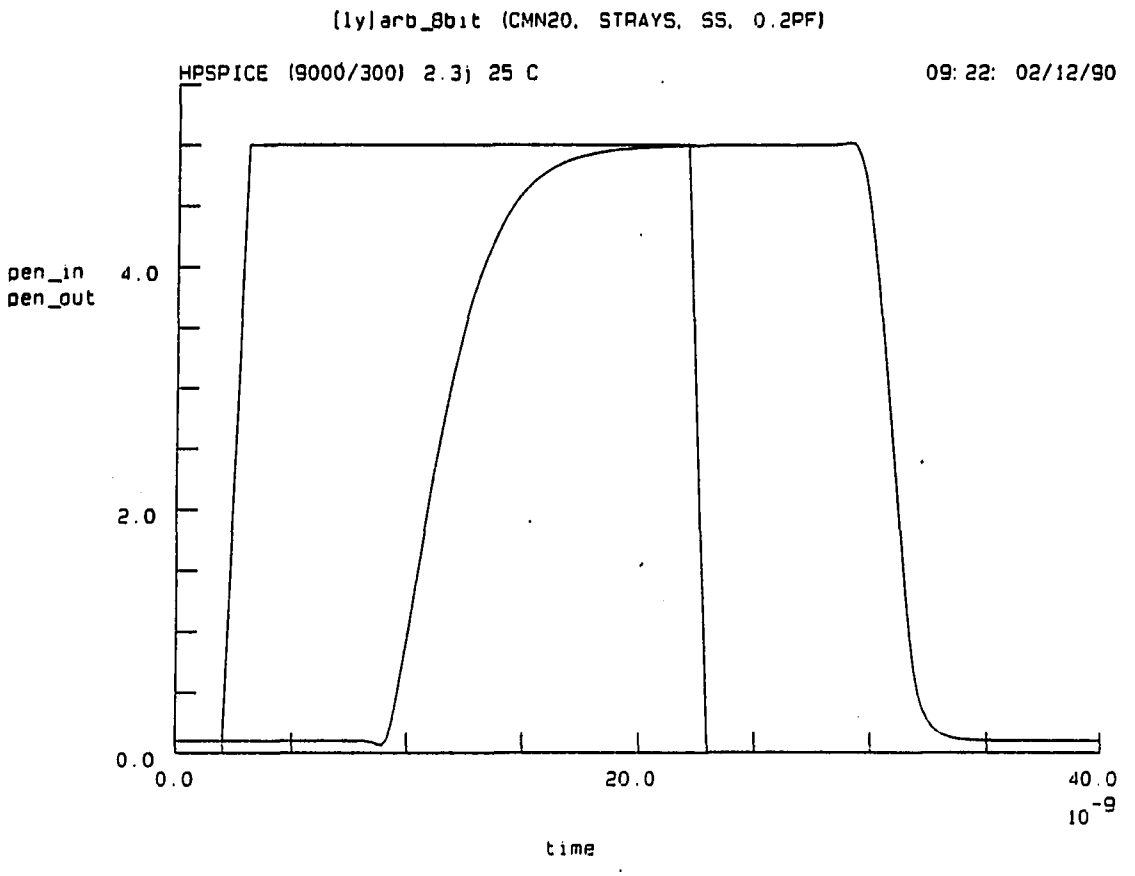


Figure 6.12: 8-Bit priority arbitration simulation

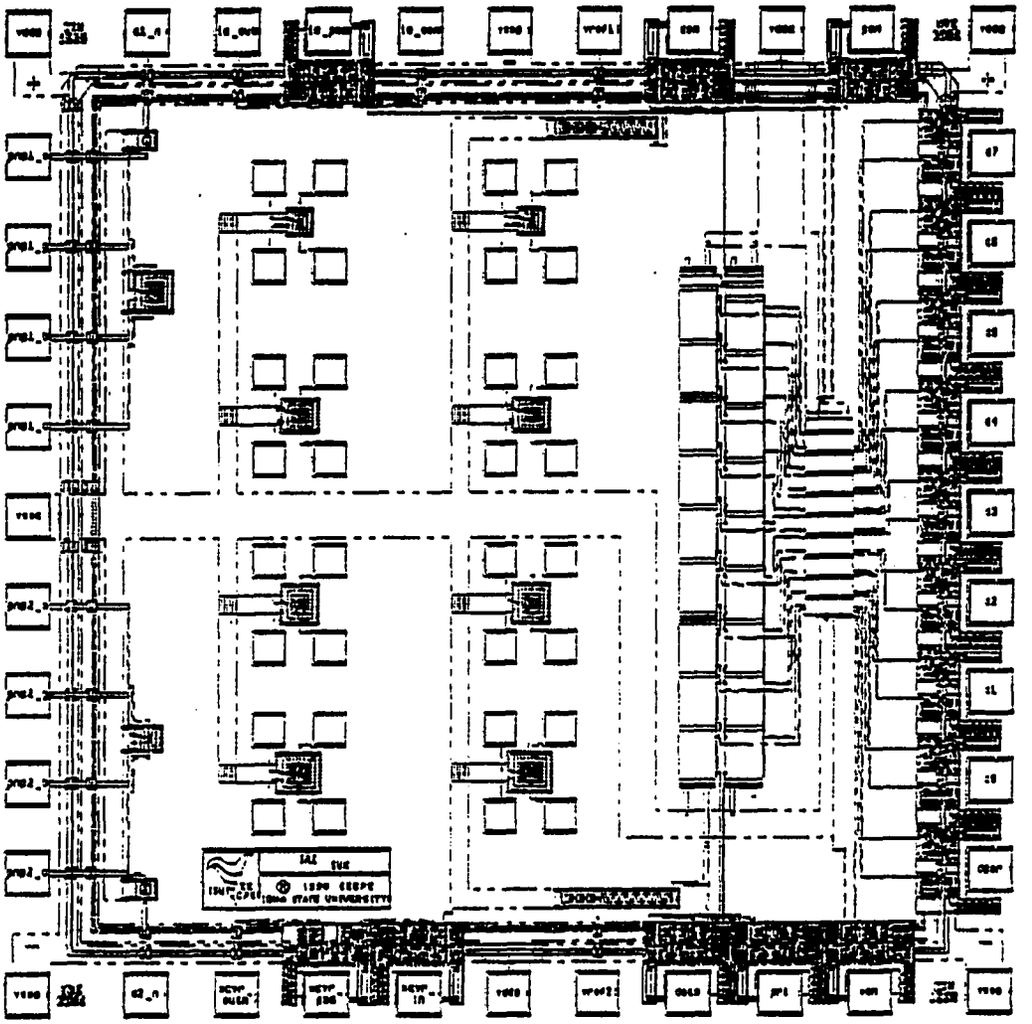


Figure 6.13: Prototype chip layout

Several transceiver pads are included in the layout. A standalone test pad for delay measurements is shown toward the lower-left. A direct external connection to the transceiver's **OUTN** signal permits output delay measurements. The **IN** signal, however, must be buffered through a digital I/O pad for external measurement. Thus the I/O pad delay must be subtracted from the input delay measurements.

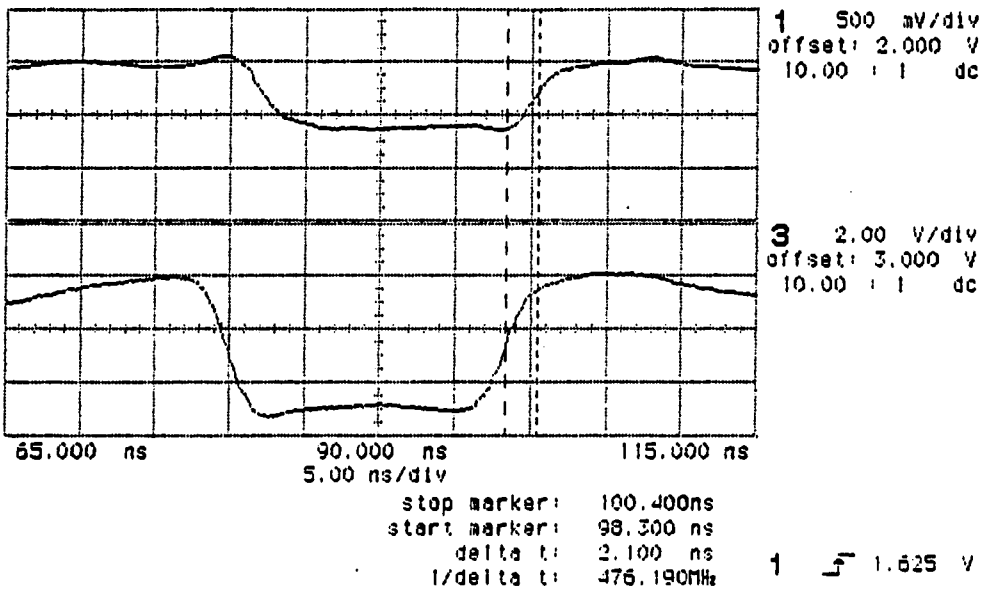
The priority arbitration circuit can be seen at the right-center of the chip. The 8-bit arbitration circuit is connected to the nine transceiver pads to the right. The start and won signals are brought in and out through I/O pads at the top and bottom, respectively. Thus, the I/O pad delays must be subtracted from measurements involving these signals.

To the left of the arbitration circuit are shift registers for loading data and priority values for testing. The data and priority inputs are located at the bottom, while the shift clock is brought in from the top, in conjunction with the I/O test pad.

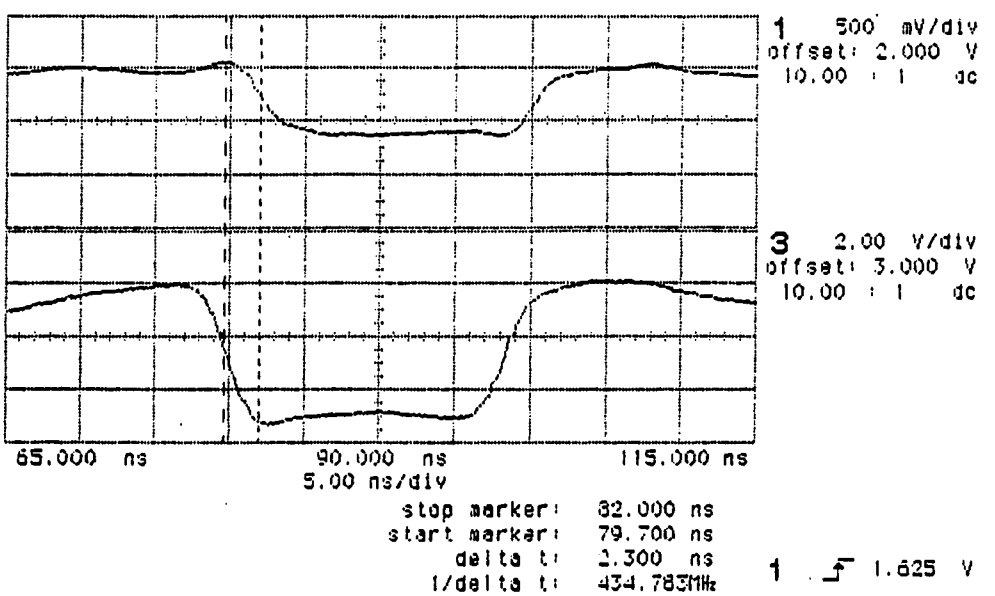
Bus Transceiver

Figure 6.14 shows the output delay measurements for the transceiver test pad. The output was tied to 2.5 V through a 25 Ω resistor. Channel 1 represents the bus voltage (**PAD**), while the output enable (**OUTN**) is shown on channel 3. Even with the increased load capacitance, the rise and fall delays are well within those predicted by simulation.

The input delay measurements for the transceiver test pad are illustrated in Figure 6.15. These were implemented by using a second test chip to generate the bus voltage swing. As before, channel 1 represents the bus voltage (**PAD**), while the input signal (**IN**) is shown on channel 3.

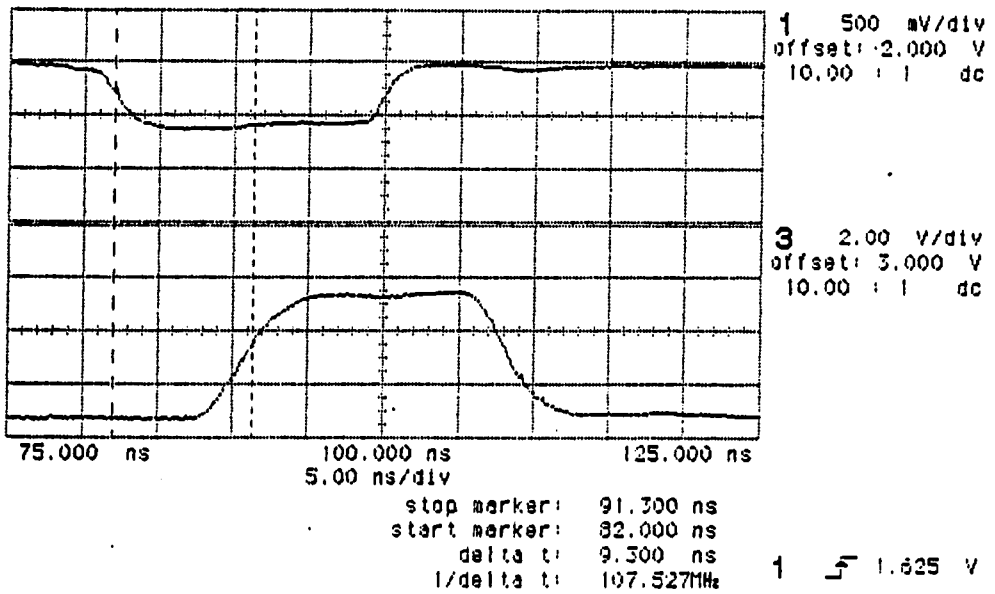


a) Rise delay

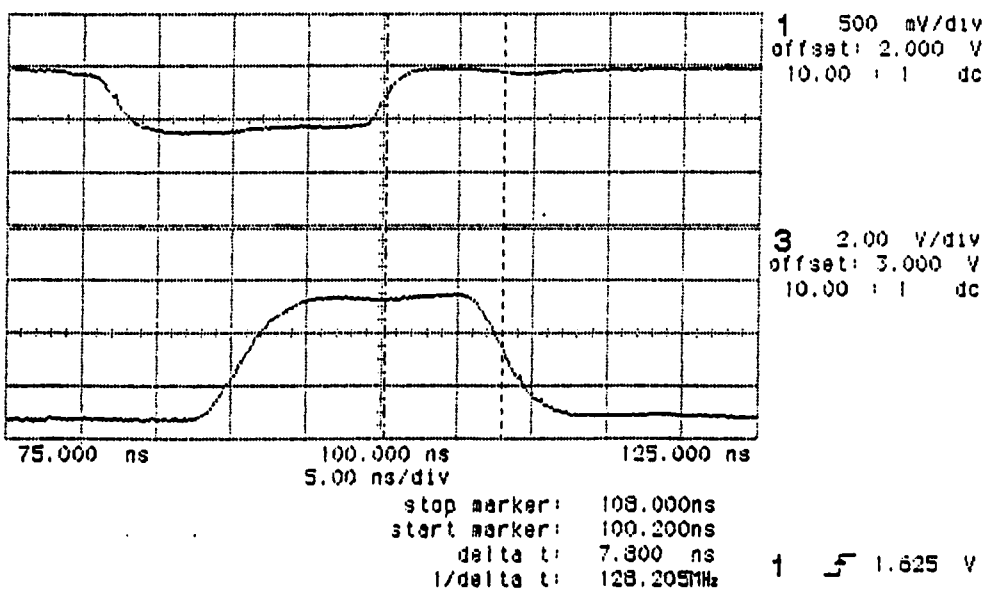


b) Fall delay

Figure 6.14: Transceiver output test waveforms



a) Rise delay



b) Fall delay

Figure 6.15: Transceiver input test waveforms

Since the core **IN** signal is buffered through a digital I/O pad, the I/O pad delay must be subtracted from the observed delays. The test I/O pad on this chip has a measured rise delay of 5.8 ns and fall delay of 4.7 ns. This results in a rise delay of 3.5 ns and a fall delay of 3.1ns for the core **IN** signal. Again, these are within the delays predicted by simulation.

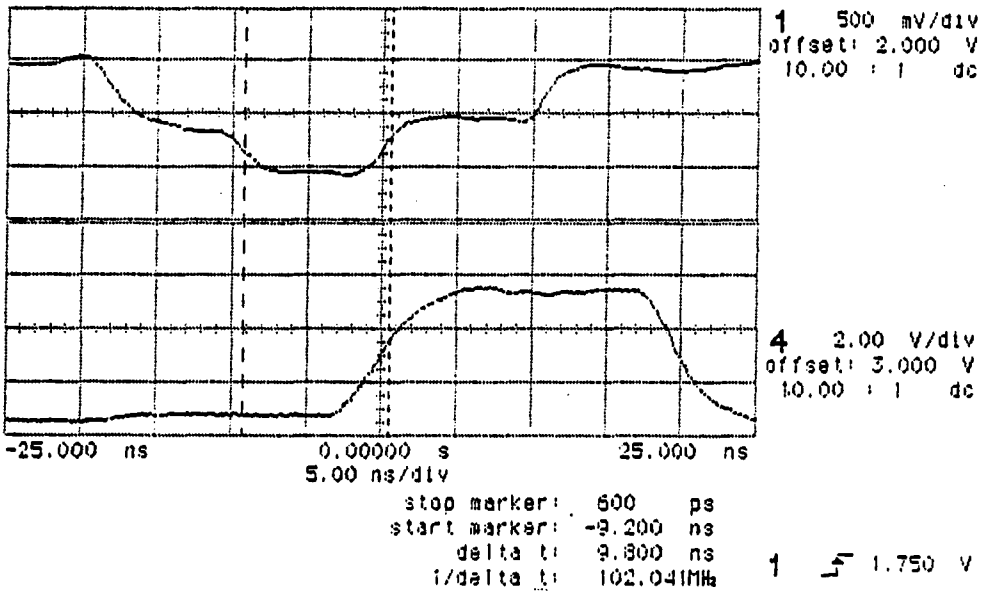
Backplane

A prototype 24-slot backplane was implemented with test transceivers at each end of a 50 Ω bus line. The line was properly terminated with 50 Ω resistors to 2.5 V. A bus collision waveform is shown in Figure 6.16.

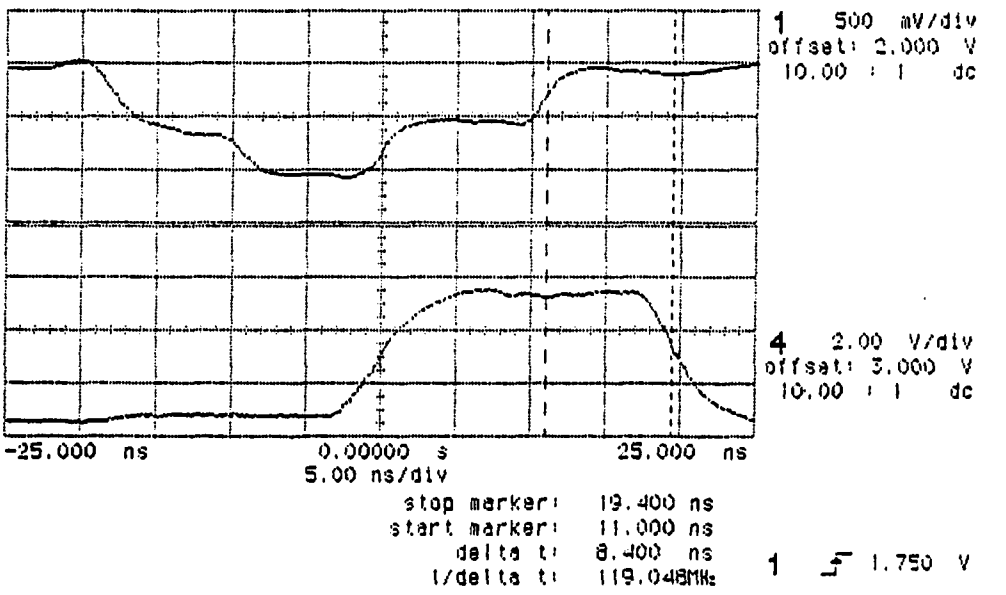
This signal represents the partial overlap of two 20 ns pulses. The first 2/3 of the signal represents the local drivers contribution to the bus signal. The latter 2/3 represents another transceivers contribution. The local transceiver's input detects only that portion of the signal generated by the other transceiver.

As shown in Figure 6.17, the 2-μm transceivers can easily accommodate a data transfer rate of 50 Mbit/s. The 20 ns pulses of logic 0, logic 1, and collisions are easily decipherable. The bus voltage is shown on channel 1, while the input signals of the two transceivers are shown on channels 3 and 4. The output enable of one of the transceivers is shown on channel 2, while the other one (not shown) is generating an output signal at 1/2 that rate. Each transceiver detects only the other's signal.

Due to limits in the pulse generating apparatus, the circuit was only tested up to about 65 Mbit/s. However, the transceiver was still functioning at that frequency. In any case, the design goal of 50 Mbit/s was met. A 100 Mbit/s transfer rate is the ultimate goal for a complete prototype implementation in 1.2-μm CMOS.



a) Rise delay



b) Fall delay

Figure 6.16: Backplane collision test waveforms

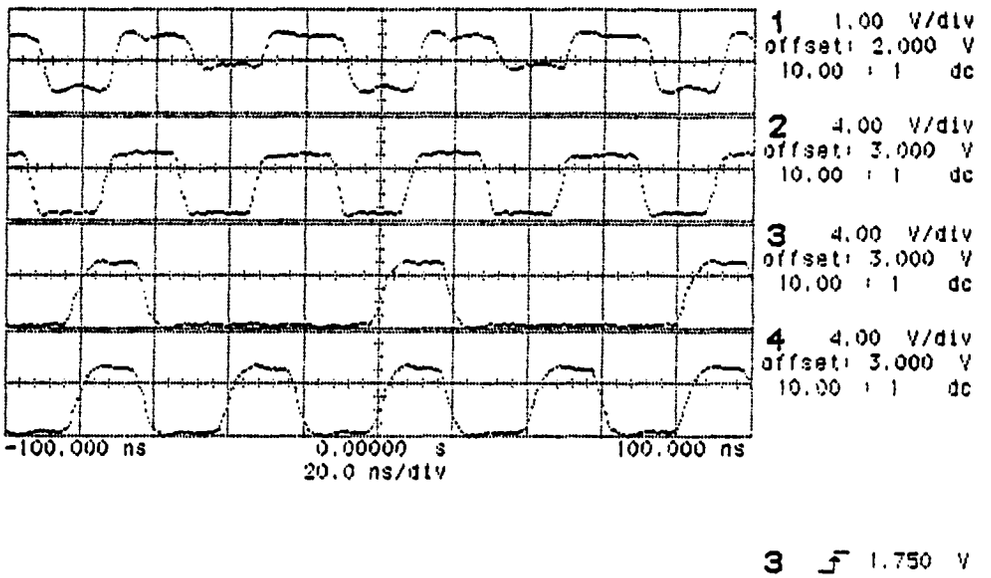


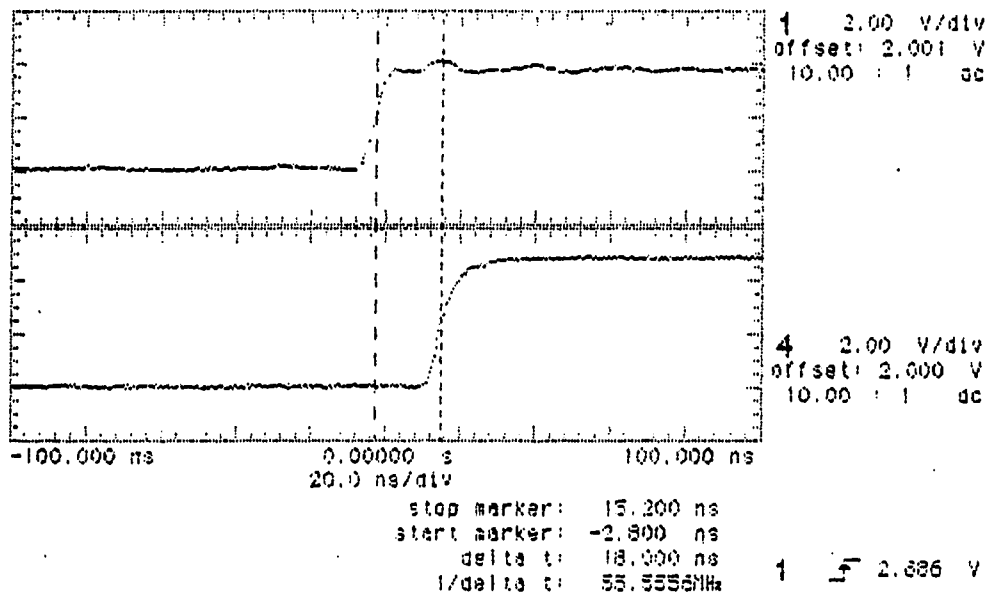
Figure 6.17: Backplane bandwidth test waveform

Priority Arbitration

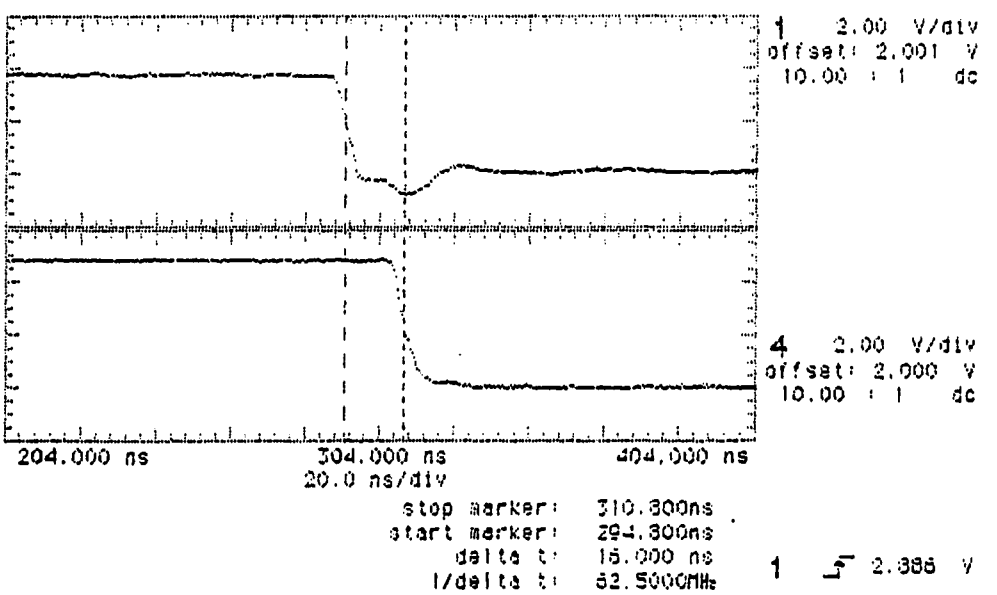
The 8-bit priority arbitration circuit was tested and the ripple delay through the circuit was measured. The resulting waveforms are shown in Figure 6.18. Before asserting **PEN_IN**, the **PRI_IN** values were loaded into the shift register. These were all set to a logic 1.

Since the **PEN_IN** and **PEN_OUT** signals must be brought in and out through digital I/O pads, the I/O pad delays must be subtracted from the observed values at the pins. From the delay for each signal edge, one input and one output delay must be subtracted.

The rising delay for an I/O pad input is 2.5 ns, while the falling delay is 2.4 ns. When combined with the I/O pad output delays previously mentioned, this results in



a) Rise delay



b) Fall delay

Figure 6.18: 8-Bit arbitration test waveforms

a rise delay of 9.7 ns and a fall delay of 8.9 ns from **PEN_IN** to **PEN_OUT** through the 8-bit arbitration circuit.

Further testing of the priority arbitration circuit should include an analysis of the different priority number combinations and their affect on the arbitration settling time.

CHAPTER 7. CONCLUSIONS

Multiple-processor architectures offer a cost-effective method of increasing the performance of computer systems. This is true, whether it is used to increase the throughput of multiple jobs, or decrease the execution time of a single task. However, as discussed in Chapter 2, there are several limitations to current interconnection networks for general-purpose, shared-memory multiprocessors.

The problems addressed in this dissertation were presented in Chapter 3. These included bandwidth limitations, scalability and expandability, fault-tolerance, capacitive bus loading, number of signal lines, VLSI implementation suitability, and multiprocessor support.

A multiple-bus, active backplane architecture was proposed as an alternative to current interconnection methods. An overview of the bus architecture was presented in Chapter 4, followed by a detailed specification in Chapter 5 of the protocol used between Active Bus Interface Units on the backplane. Included in the protocol are an efficient control acquisition scheme and hardware support for cache coherency and processor synchronization.

Chapter 6 presented the results of the physical layer implementation of the active bus protocol. This included the design and simulation of an active bus transceiver, along with the distributed priority arbitration circuit used in the control acquisi-

tion scheme. Once simulations were completed, prototypes were fabricated in 2-μm CMOS. The operation of a 24-slot backplane prototype was verified at transfer rates in excess of 50 Mbit/s.

Contribution

The primary contribution of this research is increased backplane performance. In addition to raw throughput, this also includes increased fault-tolerance, scalability, and expandability. This is achieved with a unique combination of new and existing technology at several levels of the backplane architecture:

The use of multiple buses not only increases raw throughput, but improves the fault tolerance of the system, as well. The fault-tolerance is further improved by the incorporation of check-bits on the data path for SEC-DED. The throughput can be scaled with the number of buses, while the expandability of bus-based interconnects is maintained.

The source-synchronous, word-serial transfer protocol allows data to be streamed at a high rate, while keeping the signal count under control. Also, an improved control acquisition scheme provides efficient, deterministic, and fair bus access without the need for extra signal lines. Together, these factors provide a significant improvement in pin-efficiency over current bus standards.

The new CMOS transceiver design, which is suitable for VLSI implementation, will increase the level of integration, while providing a low-capacitance bus connection. The overall reduction in load capacitance reduces both the signal propagation delay and the drive current requirements. VLSI implementation and reduced current requirements can also increase the system reliability.

This research demonstrated the feasibility of implementing the physical layer of the active bus protocol. The 2-μm transceiver prototype was shown to operate in excess of 50 Mbit/s, which was the design goal. This results in a 200 Mbyte/s transfer rate over a 32-bit datapath. The 1.2-μm transceiver design goal of 100 Mbit/s would increase that figure to 400 Mbyte/s.

Future Directions

The next step for this research is the implementation of a complete prototype. This requires the specification of the module interface and connectors. This will be done in conjunction with the implementation of bridges to other standards, such as CDC's AN/AYK-14 bus. To achieve the desired speed and level of integration, complete prototypes will require implementation in a higher-density process, such as 1.2-μm CMOS.

Other work remaining includes the high-level simulation of the complete protocol. This includes the cache coherency and synchronization primitives, along with the bus bridges. The timing and delay parameters from the prototype tests presented in this dissertation can be used to develop more accurate simulation models of the active bus protocol.

BIBLIOGRAPHY

[1] Anderson, T. E. "The Performance of Spin Lock Alternatives for Shared-Memory Multiprocessors." *IEEE Trans. on Parallel and Distributed Systems* 1 (January 1990): 6-16.

[2] Archibald, J.; and Baer, J.-L. "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model." *ACM Trans. on Computer Systems* 4 (November 1986): 273-298.

[3] Armstrong, W. J. "A High Speed, Byte-Serial, Multibus Interconnection Network." M.S. Thesis, Iowa State University, 1988.

[4] Bakoglu, H. B. *Circuits, Interconnections, and Packaging for VLSI.* Reading, MA: Addison-Wesley, 1990.

[5] Balakrishnan, R. V. "The Proposed IEEE 896 Futurebus - A Solution to the Bus Driving Problem." *IEEE Micro* 4 (August 1984): 23-27.

[6] Balakrishnan, R. V. "Active Bus Backplane." U.S. Patent 4,697,858, October, 1987.

[7] Bhuyan, L. N. "Interconnection Networks for Parallel and Distributed Processing." *Computer* 20 (June 1987): 9-12.

[8] Borrill, P.; and Theus, J. "An Advanced Communication Protocol for the Proposed IEEE 896 Futurebus." *IEEE Micro* 4 (August 1984): 42-56.

[9] Borrill, P. "Objective Comparison of 32-bit Buses." *Microprocessors and Microprogramming* 20 (March 1986): 94-100.

[10] Conte, G.; and Del Corso, D. *Multi-Microprocessor Systems for Real-Time Applications.* Dordrecht: D. Reidel Publishing Company, 1985.

[11] Control Data Corporation. *Naval Air Systems Command Specification Computer Set, Standard Airborne AN/AYK-14(V).* Control Data Corporation, Minneapolis, 1979.

[12] Davis, R. W. "Novel Logic-Signal Standard Multiplies Backplane Speeds." *Electronic Design* 36 (August 20, 1987): 81-84.

[13] Del Corso, K.; and Verrua, L. "Contention Delay in Distributed Priority Networks." *Microprocessing and Microprogramming* 13 (January 1984): 21-29.

[14] Dinning, A. "A Survey of Synchronization Methods for Parallel Computers." *Computer* 22 (July 1989): 66-77.

[15] Dubois, M.; Scheurich, C.; and Briggs, F. "Memory Access Buffering in Multiprocessors." *Proc. 13th Annual Int'l Symp. on Computer Architecture* (June 1986): 434-442.

[16] Dubois, M.; Scheurich, C.; and Briggs, F. "Synchronization, Coherence, and Event Ordering in Multiprocessors." *Computer* 21 (February 1988): 9-21.

[17] Dubois, M.; and Thakkar, S. "Cache Architectures in Tightly Coupled Multiprocessors." *Computer* 23 (June 1990): 9-11.

[18] Goodman, J. R. "Using Cache Memory to Reduce Processor-Memory Traffic." *Proc. 10th Annual Int'l Symp. on Computer Architecture* (June 1983): 124-131.

[19] Goodman, J. R.; Vernon, M. K.; and Woest, P. J. "Efficient Synchronization Primitives for Large-Scale Cache-Coherent Multiprocessors." *Proc. 3rd Int'l Conf. on Architectural Support for Programming Languages and Operating Systems* (April 1989): 64-73.

[20] Goodman, J. R.; and Woest, P. J. "The Wisconsin Multicube: A New Large-Scale Cache-Coherent Multiprocessor." *Proc. 15th Annual Int'l Symp. on Computer Architecture* (June 1988): 422-431.

[21] Graunke, G.; and Thakkar, S. "Synchronization Algorithms for Shared-Memory Multiprocessors." *Computer* 23 (June 1990): 60-69.

[22] Gray, P. R.; and Meyer, R. G. *Analysis and Design of Analog Integrated Circuits.* New York: John Wiley and Sons, Inc., 1984.

[23] Gustavson, D. B.; and Theus, J. "Wire-OR Logic on Transmission Lines." *IEEE Micro* 3 (June 1983): 51-55.

[24] Hopper, A.; Jones, A.; and Lioupis, D. "Multiple vs Wide Shared Bus Multiprocessors." *Proc. 16th Annual Int'l Symp. on Computer Architecture* (June 1989): 300-306.

[25] Hwang, K.; and Briggs, F. *Computer Architecture and Parallel Processing.* New York: McGraw-Hill, Inc., 1984.

[26] IEEE. *The IEEE 896 Futurebus Tutorial.* IEEE 896 Futurebus Tutorial Working Group, 1987.

[27] IEEE. *IEEE Standard Backplane Bus Specification for Multiprocessor Architectures: Futurebus.* The Institute of Electrical and Electronic Engineers, Inc., New York, 1988.

[28] IEEE. *IEEE Standard for a High Performance Synchronous 32-bit Bus: Multibus II.* The Institute of Electrical and Electronic Engineers, Inc., New York, 1988.

[29] IEEE. *IEEE Standard for a Versatile Backplane Bus: VMEbus.* The Institute of Electrical and Electronic Engineers, Inc., New York, 1988.

[30] IEEE. *IEEE Draft Standard P896.1: Futurebus+ Logical Layer Specification.* IEEE Computer Society Press, Los Alamitos, CA, 1989.

[31] Katz, R.; Eggers, S.; Wood, D.; Perkins, C.; and Sheldon, R. "Implementing a Cache Consistency protocol." *Proc. 12rd Annual Int'l Symp. on Computer Architecture* (June 1985): 276-283.

[32] Lang, M.; Valero, M.; and Alegre, I. "Bandwidth of Crossbar and Multiple-Bus Connections for Microprocessors." *IEEE Transactions on Computers* C-31 (December 1982): 1227-1233.

[33] Mudge, T. M.; Hayes, J. P.; Buzzard, G. D.; and Windsor, D. C. "Analysis of Multiple-Bus Interconnection Networks." *Journal of Parallel and Distributed Computing* 3 (September 1986): 328-343.

[34] Mudge, T. M.; Hayes, J. P.; and Windsor, D. C. "Multiple Bus Architectures." *Computer* 20 (June 1987): 42-48.

[35] Papamarkos, M.; and Patel, J "A Low-Overhead Coherence Solution for Multiprocessors with Private Cache Memories." *Proc. 11rd Annual Int'l Symp. on Computer Architecture* (June 1984): 348-354.

[36] Reed, D. A.; Grunwald, D. C. "The Performance of Multicomputer Interconnection Networks." *Computer* 20 (June 1987): 63-73.

[37] Scheurich, C.; and Dubois, M. "Correct Memory Operation of Cache-Based Multiprocessors." *Proc. 14th Annual Int'l Symp. on Computer Architecture* (June 1987): 234-243.

[38] Stenstrom, P. "A Survey of Cache Coherence Schemes for Multiprocessors." *Computer* 23 (June 1990): 12-24.

[39] Sweazey, P.; and Smith, A. J. "A Class of Compatible Cache Consistency Protocols and their Support by the IEEE Futurebus." *Proc. 13th Annual Int'l Symp. on Computer Architecture* (June 1986): 414-423.

[40] Taub, D. M. "Contention Resolving Circuits for Computer Interrupt Systems." *Proceedings of the IEE* 123 (September 1976): 845-850.

[41] Taub, D. M. "Overcoming Effects of Spurious Pulses on Wired-OR Lines in Computer Bus Systems." *Electronics Letters* 19 (April 28, 1983): 340-341.

[42] Taub, D. M. "Hardware Method of Synchronizing Processes Without Using a Clock." *Electronics Letters* 19 (September 15, 1983): 772-773.

[43] Taub, D. M. "Arbitration and Control Acquisition in the IEEE 896 Futurebus." *IEEE Micro* 4 (August 1984):28-41.

[44] Taub, D. M. "Improved Control Acquisition in the IEEE 896 Futurebus." *IEEE Micro* 7 (June 1987): 52-56.

[45] West, H.; and Eshraghian, K. *Principles of CMOS VLSI Design.* Reading, MA: Addison-Wesley, 1985.

[46] Wilson, A. "Hierarchical Cache/Bus Architecture for Shared Memory Multiprocessors." *Proc. 14th Annual Int'l Symp. on Computer Architecture* (June 1987): 244-252.

[47] Wilson, D. "Building Bridges with Buses." *ESD* 19 (October 1989): 55-59.

ACKNOWLEDGMENTS

I would like to thank Dr. Arthur Pohm and Dr. Charles Wright for their guidance throughout my graduate studies. Also, I wish to thank Sagar for those many "discussions", without which this work would not have been possible. And finally, I would like to thank my wife, Kelly, for her support and encouragement.

APPENDIX SPICE SOURCE LISTINGS

Transceiver Subcircuit

```
*******************************************************************************
*******************************************************************************
*****                                                                     *****
*****     File: /users/staff/irwin/spice/include/xcvr_cmn20               *****
*****                                                                     *****
*****     By:   S. A. Irwin                                               *****
*****     Date: 01/22/90                                                  *****
*****                                                                     *****
*****     SPICE subcircuit include file for CMOS backplane                *****
*****     transceiver pad with collision detection.  This                 *****
*****     circuit was extracted from the 2u CMN20 layout                  *****
*****     cell [ly]padxcvr, and includes stray capacitance.               *****
*****     This MUST be used with 2u MOS transistor models.                *****
*****                                                                     *****
*******************************************************************************
*******************************************************************************

*******************************************************************************
*****                  XCVR Subcircuit Declaration                    *****
*****       %outn - core output enable signal, active low.            *****
*****       %in   - core input signal, active high.                   *****
*****       %pad  - bonding pad, does not include the bond            *****
*****                  and package capacitance.                       *****
*****       %vdd  - +5V power rail connection.                        *****
*****       %vss  - GND rail connection.                              *****
*******************************************************************************
.SUBCKT XCVR %outn %pad %in %vdd %vss
```

```
**********************************************************************
*****                    Parasitic Capacitors                    *****
**********************************************************************
C1  %outn 2 36.9350F
*   total=.075 PF  gate=.038 linear=.037
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C2  %in   2 52.3560F
*   total=.130 PF  gate=.000 linear=.052
*   difsub=.017 difside=.014 pdifsub=.034 pdifside=.012
C3  5     2 39.0380F
*   total=.186 PF  gate=.108 linear=.039
*   difsub=.008 difside=.009 pdifsub=.014 pdifside=.008
C4  6     2 42.8790F
*   total=.187 PF  gate=.103 linear=.043
*   difsub=.006 difside=.009 pdifsub=.016 pdifside=.009
C5  7     2 54.9230F
*   total=.346 PF  gate=.202 linear=.055
*   difsub=.017 difside=.014 pdifsub=.041 pdifside=.016
C6  8     2 133.1100F
*   total=.702 PF  gate=.489 linear=.133
*   difsub=.019 difside=.006 pdifsub=.049 pdifside=.005
C7  9     2 50.9740F
*   total=.171 PF  gate=.097 linear=.051
*   difsub=.004 difside=.008 pdifsub=.005 pdifside=.006
C8  10    2 25.6860F
*   total=.085 PF  gate=.036 linear=.026
*   difsub=.004 difside=.008 pdifsub=.005 pdifside=.006
C9  11    2 30.0570F
*   total=.167 PF  gate=.093 linear=.030
*   difsub=.021 difside=.011 pdifsub=.010 pdifside=.002
C10 12    2 30.5930F
*   total=.085 PF  gate=.013 linear=.031
*   difsub=.021 difside=.011 pdifsub=.008 pdifside=.002
C11 %vref 2 51.9210F
*   total=.229 PF  gate=.124 linear=.052
*   difsub=.014 difside=.014 pdifsub=.016 pdifside=.009
C12 14    2 22.5460F
*   total=.127 PF  gate=.000 linear=.023
*   difsub=.080 difside=.025 pdifsub=.000 pdifside=.000
C13 %pad  2 248.5420F
```

```
*    total=.743 PF  gate=.079 linear=.249
*    difsub=.297 difside=.119 pdifsub=.000 pdifside=.000

*********************************************************************
*****                      Transistors                          *****
*********************************************************************
M1  %vdd  5     %in   %vdd P L=2U W=32U AS=125P  PS=30U
M2  7     6     %vdd  %vdd P L=2U W=30U AD=150P  PD=40U
M3  8     7     %vdd  %vdd P L=2U W=30U AD=90P   PD=6U
M4  %vdd  7     8     %vdd P L=2U W=30U AS=90P   PS=6U
M5  9     9     %vdd  %vdd P L=8U W=4U  AD=20P   PD=14U
M6  %vdd  %outn 6     %vdd P L=2U W=12U AS=60P   PS=22U
M7  5     10    %vdd  %vdd P L=2U W=10U AD=50P   PD=20U
M8  11    11    %vdd  %vdd P L=2U W=22U AD=36P   PD=6U
M9  12    11    %vdd  %vdd P L=2U W=20U AD=30P   PD=6U
M10 10    12    %vdd  %vdd P L=2U W=4U  AD=20P   PD=14U
M11 %vref %vref %vdd  %vdd P L=2U W=12U AD=60P   PD=22U
M12 14    %pad  11    %vss N L=2U W=34U AS=96P   PS=20U
+                                       AD=124P  PD=15.33U
M13 12    %vref 14    %vss N L=2U W=34U AS=124P  PS=15.33U
+                                       AD=96P   PD=20U
M14 %vss  12    10    %vss N L=2U W=3U  AS=19P   PS=15U
M15 5     10    %vss  %vss N L=2U W=7U  AD=35P   PD=17U
M16 7     6     %vss  %vss N L=2U W=16U AD=80P   PD=26U
M17 %vss  %outn 6     %vss N L=2U W=6U  AS=30P   PS=16U
M18 %vref %vref %vss  %vss N L=2U W=9U  AD=32.5P PD=12.5U
M19 8     7     %vss  %vss N L=2U W=15U AD=45P   PD=6U
M20 %vss  7     8     %vss N L=2U W=15U AS=45P   PS=6U
M21 %in   5     %vss  %vss N L=2U W=16U AD=80P   PD=26U
M22 %vss  9     14    %vss N L=2U W=20U AS=124P  PS=15.33U
M23 %vss  8     %vref %vss N L=4U W=4U  AS=32.5P PS=12.5U
M24 %vss  9     9     %vss N L=8U W=4U  AS=20P   PS=14U
M25 %vss  8     %pad  %vss N L=2U W=34U AS=230P  PS=36.67U
M26 %vss  8     %pad  %vss N L=2U W=34U AS=230P  PS=36.67U
M27 %vss  8     %pad  %vss N L=2U W=34U AS=230P  PS=36.67U
M28 %vss  8     %pad  %vss N L=2U W=34U AS=230P  PS=36.67U
M29 %vss  8     %pad  %vss N L=2U W=34U AS=230P  PS=36.67U
M30 %vss  8     %pad  %vss N L=2U W=34U AS=230P  PS=36.67U

.ENDS
```

Transceiver Output Simulation

```
CMOS Backplane Transceiver (CMN20, SS, Cpad=3pF)
*******************************************************************************
*******************************************************************************
*****                                                                     *****
*****      File: /users/staff/irwin/spice/xcvr/xcvr_out.cki               *****
*****                                                                     *****
*****      By:   S. A. Irwin                                              *****
*****      Date: 03/07/90                                                 *****
*****                                                                     *****
*****      SPICE circuit file for simulating the output                   *****
*****      characteristics of the CMOS backplane transceiver,             *****
*****      using the worst-case CMN20 process parameters.                 *****
*****                                                                     *****
*******************************************************************************
*******************************************************************************

.OPTION UCB LIMPTS=1001
.TEMP 25
.TRANS 0.02n 20n

*******************************************************************************
*****                N-Channel / Enhancement / Slow                       *****
*******************************************************************************
.MODEL N NMOS LEVEL=2
+ VTO=0.88      TOX=430E-10  NSUB=1.0E16   XJ=0.15E-6    LD=0.20E-6
+ UO=620        UCRIT=0.62E5 UEXP=0.125    VMAX=5.1E4    NEFF=4.0
+ DELTA=1.4     RSH=42       CGSO=2.75E-10 CGDO=2.75E-10 CJ=215E-6
+ CJSW=540E-12 MJ=0.76      MJSW=0.30     PB=0.80

*******************************************************************************
*****                P-Channel / Enhancement / Slow                       *****
*******************************************************************************
.MODEL P PMOS LEVEL=2
+ VTO=-0.95     TOX=430E-10  NSUB=6.6E15   XJ=0.05E-6    LD=0.20E-6
+ UO=240        UCRIT=0.86E5 UEXP=0.29     VMAX=3.0E4    NEFF=2.65
+ DELTA=1.0     RSH=145      CGSO=2.45E-10 CGDO=2.45E-10 CJ=275E-6
+ CJSW=400E-12 MJ=0.535     MJSW=0.34     PB=0.80
```

```
*************************************************************************
*****                      Load Include Files                       *****
*************************************************************************
.INCLUDE /users/staff/irwin/spice/include/xcvr_cmn20

*************************************************************************
*****                      Define the Circuit                       *****
*************************************************************************

XCV    %outn %pad %in  %vdd  %vss XCVR
R1     %vt   %pad  25
CL     %pad  %vss  2.257P

*************************************************************************
*****                      Voltage Sources                          *****
*************************************************************************
VDD    %vdd  0 5
VSS    %vss  0 0.1
VT     %vt   0 2.5
VOUTN %outn 0 PULSE(5 0 2n 1n 1n 9n 20n)

.END
```

Transceiver Input Simulation

```
CMOS Backplane Transceiver (CMN20, SS, Cin=0.1pF)
*******************************************************************************
*******************************************************************************
*****                                                                     *****
*****     File: /users/staff/irwin/spice/xcvr/xcvr_in.cki                 *****
*****                                                                     *****
*****     By:   S. A. Irwin                                               *****
*****     Date: 03/07/90                                                  *****
*****                                                                     *****
*****     SPICE circuit file for simulating the input                     *****
*****     characteristics of the CMOS backplane transceiver,              *****
*****     using the worst-case CMN20 process parameters.                  *****
*****                                                                     *****
*******************************************************************************
*******************************************************************************

.OPTION UCB LIMPTS=1001
.TEMP 25
.TRANS 0.02n 20n

*******************************************************************************
*****                N-Channel / Enhancement / Slow                       *****
*******************************************************************************
.MODEL N NMOS LEVEL=2
+ VTO=0.88      TOX=430E-10  NSUB=1.0E16   XJ=0.15E-6    LD=0.20E-6
+ UO=620        UCRIT=0.62E5 UEXP=0.125    VMAX=5.1E4    NEFF=4.0
+ DELTA=1.4     RSH=42       CGSO=2.75E-10 CGDO=2.75E-10 CJ=215E-6
+ CJSW=540E-12 MJ=0.76      MJSW=0.30     PB=0.80

*******************************************************************************
*****                P-Channel / Enhancement / Slow                       *****
*******************************************************************************
.MODEL P PMOS LEVEL=2
+ VTO=-0.95     TOX=430E-10  NSUB=6.6E15   XJ=0.05E-6    LD=0.20E-6
+ UO=240        UCRIT=0.86E5 UEXP=0.29     VMAX=3.0E4    NEFF=2.65
+ DELTA=1.0     RSH=145      CGSO=2.45E-10 CGDO=2.45E-10 CJ=275E-6
+ CJSW=400E-12 MJ=0.535     MJSW=0.34     PB=0.80
```

```
*******************************************************************************
*****                        Load Include Files                           *****
*******************************************************************************
.INCLUDE /users/staff/irwin/spice/include/xcvr_cmn20

*******************************************************************************
*****                        Define the Circuit                           *****
*******************************************************************************

XCV   %outn %pad %in  %vdd  %vss XCVR
CL    %in   %vss  0.1P

*******************************************************************************
*****                          Voltage Sources                            *****
*******************************************************************************
VDD   %vdd  0 5
VSS   %vss  0 0.1
VOUTN %outn 0 5
VPAD  %pad  0 PULSE(2.5 2.0 2n 1n 1n 9n 20n)

.END
```

24-Slot Active Backplane Simulation

```
24 Slot Active Backplane (CMN20, SS, CL=3pF, Cin=0.1pF)
*******************************************************************************
*******************************************************************************
*****                                                                     *****
*****     File: /users/staff/irwin/spice/backplane/bp.cki                 *****
*****                                                                     *****
*****     By:   S. A. Irwin                                               *****
*****     Date: 03/07/90                                                  *****
*****                                                                     *****
*****     SPICE circuit file for simulating a 24-slot active              *****
*****     backplane using transmission line elements and                  *****
*****     CMOS bus transceivers.                                          *****
*****                                                                     *****
*******************************************************************************
*******************************************************************************

.OPTION UCB LIMPTS=2001
.TEMP 25
.TRANS 0.02n 40n

*******************************************************************************
*****                N-Channel / Enhancement / Slow                   *****
*******************************************************************************
.MODEL N NMOS LEVEL=2
+ VTO=0.88      TOX=430E-10  NSUB=1.0E16   XJ=0.15E-6    LD=0.20E-6
+ UO=620        UCRIT=0.62E5 UEXP=0.125    VMAX=5.1E4    NEFF=4.0
+ DELTA=1.4     RSH=42       CGSO=2.75E-10 CGDO=2.75E-10 CJ=215E-6
+ CJSW=540E-12 MJ=0.76      MJSW=0.30     PB=0.80

*******************************************************************************
*****                P-Channel / Enhancement / Slow                   *****
*******************************************************************************
.MODEL P PMOS LEVEL=2
+ VTO=-0.95     TOX=430E-10  NSUB=6.6E15   XJ=0.05E-6    LD=0.20E-6
+ UO=240        UCRIT=0.86E5 UEXP=0.29     VMAX=3.0E4    NEFF=2.65
+ DELTA=1.0     RSH=145      CGSO=2.45E-10 CGDO=2.45E-10 CJ=275E-6
+ CJSW=400E-12 MJ=0.535     MJSW=0.34     PB=0.80
```

```
*************************************************************************
*****                    Load Global Include Files                  *****
*************************************************************************
.INCLUDE /users/staff/irwin/spice/include/xcvr_cmn20

*************************************************************************
*****                    Define Local Subcircuits                   *****
*************************************************************************
.SUBCKT TLINE %left %slot %right %vss
LD1   %left   %slot    5NH
CD    %slot   %vss     1PF
CL    %slot   %vss     3PF
LD2   %slot   %right   5NH
.ENDS

*************************************************************************
*****                      Define the Circuit                       *****
*************************************************************************
RT1      %vt       1         50
XL1      1         %slot1    2        %vss   TLINE
XCVR1    %outn1    %slot1    %in1     %vdd   %vss   XCVR
CIN1     %in1      %vss      0.1PF
XL2      2         %slot2    3        %vss   TLINE
XL3      3         %slot3    4        %vss   TLINE
XL4      4         %slot4    5        %vss   TLINE
XL5      5         %slot5    6        %vss   TLINE
XL6      6         %slot6    7        %vss   TLINE
XL7      7         %slot7    8        %vss   TLINE
XL8      8         %slot8    9        %vss   TLINE
XL9      9         %slot9    10       %vss   TLINE
XL10     10        %slot10   11       %vss   TLINE
XL11     11        %slot11   12       %vss   TLINE
XL12     12        %slot12   13       %vss   TLINE
XL13     13        %slot13   14       %vss   TLINE
XL14     14        %slot14   15       %vss   TLINE
XL15     15        %slot15   16       %vss   TLINE
XL16     16        %slot16   17       %vss   TLINE
XL17     17        %slot17   18       %vss   TLINE
XL18     18        %slot18   19       %vss   TLINE
XL19     19        %slot19   20       %vss   TLINE
```

```
XL20     20        %slot20  21       %vss  TLINE
XL21     21        %slot21  22       %vss  TLINE
XL22     22        %slot22  23       %vss  TLINE
XL23     23        %slot23  24       %vss  TLINE
XL24     24        %slot24  25       %vss  TLINE
XCVR24   %outn24   %slot24  %in24    %vdd  %vss  XCVR
CIN24    %in24     %vss     0.1PF
RT2      %vt       25       50

*******************************************************************************
*****                        Voltage Sources                              *****
*******************************************************************************
VDD      %vdd     0 5
VSS      %vss     0 0.1
VT       %vt      0 2.5
VOUTN1   %outn1   0 PULSE(5 0 2NS 1NS 1NS 19NS 50NS)
VOUTN24  %outn24  0 5

.END
```

24-Slot Backplane With Collision

```
24 Slot Active Backplane (CMN20, SS, CL=3pF, Cin=0.1pF)
*******************************************************************************
*******************************************************************************
*****                                                                     *****
*****     File: /users/staff/irwin/spice/backplane/bp_cd.cki              *****
*****                                                                     *****
*****     By:   S. A. Irwin                                               *****
*****     Date: 04/07/90                                                  *****
*****                                                                     *****
*****     SPICE circuit file for simulating the collision                 *****
*****     detection of the CMOS bus transceivers on a 24-slot             *****
*****     active backplane.                                               *****
*****                                                                     *****
*******************************************************************************
*******************************************************************************

.OPTION UCB LIMPTS=2001
.TEMP 25
.TRANS 0.02n 40n

*******************************************************************************
*****                N-Channel / Enhancement / Slow                   *****
*******************************************************************************
.MODEL N NMOS LEVEL=2
+ VTO=0.88      TOX=430E-10  NSUB=1.0E16   XJ=0.15E-6    LD=0.20E-6
+ UO=620        UCRIT=0.62E5 UEXP=0.125    VMAX=5.1E4    NEFF=4.0
+ DELTA=1.4     RSH=42       CGSO=2.75E-10 CGDO=2.75E-10 CJ=215E-6
+ CJSW=540E-12 MJ=0.76      MJSW=0.30     PB=0.80

*******************************************************************************
*****                P-Channel / Enhancement / Slow                   *****
*******************************************************************************
.MODEL P PMOS LEVEL=2
+ VTO=-0.95     TOX=430E-10  NSUB=6.6E15   XJ=0.05E-6    LD=0.20E-6
+ UO=240        UCRIT=0.86E5 UEXP=0.29     VMAX=3.0E4    NEFF=2.65
+ DELTA=1.0     RSH=145      CGSO=2.45E-10 CGDO=2.45E-10 CJ=275E-6
+ CJSW=400E-12 MJ=0.535     MJSW=0.34     PB=0.80
```

```
*************************************************************************
*****                  Load Global Include Files                  *****
*************************************************************************
.INCLUDE /users/staff/irwin/spice/include/xcvr_cmn20

*************************************************************************
*****                  Define Local Subcircuits                   *****
*************************************************************************
.SUBCKT TLINE %left %slot %right %vss
LD1   %left   %slot    5NH
CD    %slot   %vss     1PF
CL    %slot   %vss     3PF
LD2   %slot   %right   5NH
.ENDS

*************************************************************************
*****                     Define the Circuit                      *****
*************************************************************************
RT1      %vt       1         50
XL1      1         %slot1    2         %vss   TLINE
XCVR1    %outn1    %slot1    %in1      %vdd   %vss   XCVR
CIN1     %in1      %vss      0.1PF
XL2      2         %slot2    3         %vss   TLINE
XL3      3         %slot3    4         %vss   TLINE
XL4      4         %slot4    5         %vss   TLINE
XL5      5         %slot5    6         %vss   TLINE
XL6      6         %slot6    7         %vss   TLINE
XL7      7         %slot7    8         %vss   TLINE
XL8      8         %slot8    9         %vss   TLINE
XL9      9         %slot9    10        %vss   TLINE
XL10     10        %slot10   11        %vss   TLINE
XL11     11        %slot11   12        %vss   TLINE
XL12     12        %slot12   13        %vss   TLINE
XL13     13        %slot13   14        %vss   TLINE
XL14     14        %slot14   15        %vss   TLINE
XL15     15        %slot15   16        %vss   TLINE
XL16     16        %slot16   17        %vss   TLINE
XL17     17        %slot17   18        %vss   TLINE
XL18     18        %slot18   19        %vss   TLINE
XL19     19        %slot19   20        %vss   TLINE
```

```
XL20     20        %slot20  21       %vss  TLINE
XL21     21        %slot21  22       %vss  TLINE
XL22     22        %slot22  23       %vss  TLINE
XL23     23        %slot23  24       %vss  TLINE
XL24     24        %slot24  25       %vss  TLINE
XCVR24   %outn24   %slot24  %in24    %vdd  %vss  XCVR
CIN24    %in24     %vss     0.1PF
RT2      %vt       25       50

*********************************************************************
*****                      Voltage Sources                      *****
*********************************************************************
VDD      %vdd      0 5
VSS      %vss      0 0.1
VT       %vt       0 2.5
VOUTN1   %outn1    0 PULSE(5 0 2NS 1NS 1NS 19NS 50NS)
VOUTN24  %outn24   0 PULSE(5 0 2NS 1NS 1NS 19NS 50NS)

.END
```

8-Bit Priority Arbitration Circuit

```
8-Bit CMOS Priority Arbitration (CMN20, SS, C=0.2pF)
*********************************************************************
*********************************************************************
*****                                                           *****
*****      File: /users/staff/irwin/spice/xcvr/xcvr_in.cki      *****
*****                                                           *****
*****      By:   S. A. Irwin                                    *****
*****      Date: 02/21/90                                       *****
*****                                                           *****
*****      SPICE circuit file for simulating the logic delay    *****
*****      through the 8-bit priority arbitration circuit       *****
*****      using the worst-case CMN20 process parameters.       *****
*****                                                           *****
*********************************************************************
*********************************************************************

.OPTION UCB LIMPTS=1001
.TEMP 25
.TRANS 0.2n 40n

*********************************************************************
*****              N-Channel / Enhancement / Slow               *****

*********************************************************************
.MODEL N NMOS LEVEL=2
+ VTO=0.88      TOX=430E-10  NSUB=1.0E16   XJ=0.15E-6    LD=0.20E-6
+ UO=620        UCRIT=0.62E5 UEXP=0.125    VMAX=5.1E4    NEFF=4.0
+ DELTA=1.4     RSH=42       CGSO=2.75E-10 CGDO=2.75E-10 CJ=215E-6
+ CJSW=540E-12 MJ=0.76      MJSW=0.30     PB=0.80

*********************************************************************
*****              P-Channel / Enhancement / Slow               *****
*********************************************************************
.MODEL P PMOS LEVEL=2
+ VTO=-0.95     TOX=430E-10  NSUB=6.6E15   XJ=0.05E-6    LD=0.20E-6
+ UO=240        UCRIT=0.86E5 UEXP=0.29     VMAX=3.0E4    NEFF=2.65
+ DELTA=1.0     RSH=145      CGSO=2.45E-10 CGDO=2.45E-10 CJ=275E-6
+ CJSW=400E-12 MJ=0.535     NJSW=0.34     PB=0.80
```

```
*******************************************************************************
*****                        Define the Circuit                           *****
*******************************************************************************

*******************************************************************************
*****                      Parasitic Capacitors                           *****
*******************************************************************************
C1 %den_in 2 7.2670F
*   total=.046 PF  gate=.038 linear=.007
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C2 %pen_in 2 20.4140F
*   total=.111 PF  gate=.091 linear=.020
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C3 5 2 131.9820F
*   total=1.322 PF  gate=.000 linear=.132
*   difsub=.349 difside=.482 pdifsub=.231 pdifside=.128
C4 6 2 118.3670F
*   total=1.311 PF  gate=.000 linear=.118
*   difsub=.381 difside=.561 pdifsub=.193 pdifside=.058
C5 7 2 20.2410F
*   total=.020 PF  gate=.000 linear=.020
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C6 8 2 9.7780F
*   total=.042 PF  gate=.000 linear=.010
*   difsub=.006 difside=.005 pdifsub=.016 pdifside=.004
C7 9 2 17.9160F
*   total=.056 PF  gate=.038 linear=.018
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C8 10 2 17.1880F
*   total=.056 PF  gate=.038 linear=.017
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C9 %pen_out 2 30.2950F
*   total=.103 PF  gate=.038 linear=.030
*   difsub=.008 difside=.005 pdifsub=.018 pdifside=.004
C10 12 2 11.1250F
*   total=.048 PF  gate=.000 linear=.011
*   difsub=.006 difside=.005 pdifsub=.020 pdifside=.005
C11 13 2 32.4710F
*   total=.052 PF  gate=.020 linear=.032
```

```
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C12 14 2 9.7970F
*   total=.046 PF  gate=.000 linear=.010
*   difsub=.006 difside=.006 pdifsub=.020 pdifside=.005
C13 15 2 30.8750F
*   total=.074 PF  gate=.043 linear=.031
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C14 16 2 22.2560F
*   total=.118 PF  gate=.095 linear=.022
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C15 17 2 23.1110F
*   total=.061 PF  gate=.038 linear=.023
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C16 18 2 10.5920F
*   total=.030 PF  gate=.020 linear=.011
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C17 19 2 21.6300F
*   total=.060 PF  gate=.038 linear=.022
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C18 20 2 22.2550F
*   total=.118 PF  gate=.095 linear=.022
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C19 21 2 23.1110F
*   total=.061 PF  gate=.038 linear=.023
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C20 22 2 32.4710F
*   total=.052 PF  gate=.020 linear=.032
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C21 23 2 10.5920F
*   total=.030 PF  gate=.020 linear=.011
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C22 24 2 21.6300F
*   total=.060 PF  gate=.038 linear=.022
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C23 25 2 30.8750F
*   total=.074 PF  gate=.043 linear=.031
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C24 26 2 22.2550F
*   total=.118 PF  gate=.095 linear=.022
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
```

```
C25 27 2 23.1110F
*   total=.061 PF  gate=.038 linear=.023
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C26 28 2 32.4710F
*   total=.052 PF  gate=.020 linear=.032
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C27 29 2 10.5920F
*   total=.030 PF  gate=.020 linear=.011
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C28 30 2 21.6300F
*   total=.060 PF  gate=.038 linear=.022
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C29 31 2 30.8750F
*   total=.074 PF  gate=.043 linear=.031
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C30 32 2 22.2550F
*   total=.118 PF  gate=.095 linear=.022
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C31 33 2 23.1110F
*   total=.061 PF  gate=.038 linear=.023
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C32 34 2 32.4710F
*   total=.052 PF  gate=.020 linear=.032
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C33 35 2 10.5920F
*   total=.030 PF  gate=.020 linear=.011
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C34 36 2 21.6300F
*   total=.060 PF  gate=.038 linear=.022
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C35 37 2 30.8750F
*   total=.074 PF  gate=.043 linear=.031
*   difsub=.000 difside=.000 pdifsub=.000 pdifside=.000
C36 38 2 9.8770F
*   total=.047 PF  gate=.000 linear=.010
*   difsub=.006 difside=.006 pdifsub=.020 pdifside=.005
C37 39 2 11.1250F
*   total=.048 PF  gate=.000 linear=.011
*   difsub=.006 difside=.005 pdifsub=.020 pdifside=.005
C38 40 2 9.7970F
```

```
*   total=.046 PF  gate=.000 linear=.010
*   difsub=.006 difside=.006 pdifsub=.020 pdifside=.005
C39 41 2 11.1250F
*   total=.048 PF  gate=.000 linear=.011
*   difsub=.006 difside=.005 pdifsub=.020 pdifside=.005
C40 42 2 9.7970F
*   total=.046 PF  gate=.000 linear=.010
*   difsub=.006 difside=.006 pdifsub=.020 pdifside=.005
C41 43 2 11.1250F
*   total=.048 PF  gate=.000 linear=.011
*   difsub=.006 difside=.005 pdifsub=.020 pdifside=.005
C42 44 2 177.3210F
*   total=.566 PF  gate=.345 linear=.177
*   difsub=.010 difside=.005 pdifsub=.025 pdifside=.004
C43 45 2 21.3870F
*   total=.177 PF  gate=.118 linear=.021
*   difsub=.006 difside=.008 pdifsub=.015 pdifside=.008
C44 46 2 10.4430F
*   total=.077 PF  gate=.034 linear=.010
*   difsub=.004 difside=.005 pdifsub=.015 pdifside=.008
C45 47 2 23.2760F
*   total=.126 PF  gate=.077 linear=.023
*   difsub=.004 difside=.008 pdifsub=.008 pdifside=.006
C46 48 2 31.2870F
*   total=.151 PF  gate=.077 linear=.031
*   difsub=.014 difside=.005 pdifsub=.019 pdifside=.005
C47 49 2 24.6860F
*   total=.150 PF  gate=.091 linear=.025
*   difsub=.008 difside=.005 pdifsub=.018 pdifside=.004
C48 50 2 10.4430F
*   total=.077 PF  gate=.034 linear=.010
*   difsub=.004 difside=.005 pdifsub=.015 pdifside=.008
C49 51 2 23.2760F
*   total=.126 PF  gate=.077 linear=.023
*   difsub=.004 difside=.008 pdifsub=.008 pdifside=.006
C50 52 2 31.2870F
*   total=.151 PF  gate=.077 linear=.031
*   difsub=.014 difside=.005 pdifsub=.019 pdifside=.005
C51 53 2 24.6860F
*   total=.150 PF  gate=.091 linear=.025
```

```
*   difsub=.008 difside=.005 pdifsub=.018 pdifside=.004
C52 54 2 10.4430F
*   total=.077 PF  gate=.034 linear=.010
*   difsub=.004 difside=.005 pdifsub=.015 pdifside=.008
C53 55 2 23.2760F
*   total=.126 PF  gate=.077 linear=.023
*   difsub=.004 difside=.008 pdifsub=.008 pdifside=.006
C54 56 2 31.2870F
*   total=.151 PF  gate=.077 linear=.031
*   difsub=.014 difside=.005 pdifsub=.019 pdifside=.005
C55 57 2 24.6860F
*   total=.150 PF  gate=.091 linear=.025
*   difsub=.008 difside=.005 pdifsub=.018 pdifside=.004
C56 58 2 10.4430F
*   total=.077 PF  gate=.034 linear=.010
*   difsub=.004 difside=.005 pdifsub=.015 pdifside=.008
C57 59 2 23.2760F
*   total=.126 PF  gate=.077 linear=.023
*   difsub=.004 difside=.008 pdifsub=.008 pdifside=.006
C58 60 2 31.2870F
*   total=.151 PF  gate=.077 linear=.031
*   difsub=.014 difside=.005 pdifsub=.019 pdifside=.005
C59 61 2 10.9260F
*   total=.078 PF  gate=.000 linear=.011
*   difsub=.000 difside=.000 pdifsub=.046 pdifside=.021
C60 62 2 6.5260F
*   total=.052 PF  gate=.000 linear=.007
*   difsub=.000 difside=.000 pdifsub=.033 pdifside=.013
* C61 63 2 .0000F
*   total=.010 PF  gate=.000 linear=.000
*   difsub=.000 difside=.000 pdifsub=.008 pdifside=.002
C62 64 2 7.7220F
*   total=.075 PF  gate=.000 linear=.008
*   difsub=.000 difside=.000 pdifsub=.046 pdifside=.021
C63 65 2 6.5260F
*   total=.052 PF  gate=.000 linear=.007
*   difsub=.000 difside=.000 pdifsub=.033 pdifside=.013
C65 67 2 7.7220F
*   total=.075 PF  gate=.000 linear=.008
*   difsub=.000 difside=.000 pdifsub=.046 pdifside=.021
```

```
C66 68 2 6.5260F
*   total=.052 PF  gate=.000 linear=.007
*   difsub=.000 difside=.000 pdifsub=.033 pdifside=.013
C68 70 2 7.7220F
*   total=.075 PF  gate=.000 linear=.008
*   difsub=.000 difside=.000 pdifsub=.046 pdifside=.021
C69 71 2 6.5260F
*   total=.052 PF  gate=.000 linear=.007
*   difsub=.000 difside=.000 pdifsub=.033 pdifside=.013
C71 73 2 7.7220F
*   total=.075 PF  gate=.000 linear=.008
*   difsub=.000 difside=.000 pdifsub=.046 pdifside=.021
C72 74 2 .5650F
*   total=.008 PF  gate=.000 linear=.001
*   difsub=.004 difside=.003 pdifsub=.000 pdifside=.000
C74 76 2 .5650F
*   total=.008 PF  gate=.000 linear=.001
*   difsub=.004 difside=.003 pdifsub=.000 pdifside=.000
C76 78 2 .5650F
*   total=.008 PF  gate=.000 linear=.001
*   difsub=.004 difside=.003 pdifsub=.000 pdifside=.000
C78 80 2 .5650F
*   total=.008 PF  gate=.000 linear=.001
*   difsub=.004 difside=.003 pdifsub=.000 pdifside=.000
C82 84 2 .7530F
*   total=.008 PF  gate=.000 linear=.001
*   difsub=.004 difside=.003 pdifsub=.000 pdifside=.000
C83 85 2 .7530F
*   total=.008 PF  gate=.000 linear=.001
*   difsub=.004 difside=.003 pdifsub=.000 pdifside=.000
C84 86 2 .7530F
*   total=.008 PF  gate=.000 linear=.001
*   difsub=.004 difside=.003 pdifsub=.000 pdifside=.000
C85 87 2 .7530F
*   total=.008 PF  gate=.000 linear=.001
*   difsub=.004 difside=.003 pdifsub=.000 pdifside=.000
C86 88 2 17.4810F
*   total=.100 PF  gate=.057 linear=.017
*   difsub=.004 difside=.008 pdifsub=.007 pdifside=.006
C87 89 2 17.4810F
```

```
*   total=.100 PF  gate=.057 linear=.017
*   difsub=.004 difside=.008 pdifsub=.007 pdifside=.006
C88 90 2 17.4810F
*   total=.100 PF  gate=.057 linear=.017
*   difsub=.004 difside=.008 pdifsub=.007 pdifside=.006
C89 91 2 17.4810F
*   total=.100 PF  gate=.057 linear=.017
*   difsub=.004 difside=.008 pdifsub=.007 pdifside=.006
C90 92 2 2.6410F
*   total=.046 PF  gate=.000 linear=.003
*   difsub=.027 difside=.017 pdifsub=.000 pdifside=.000
C91 93 2 2.6410F
*   total=.046 PF  gate=.000 linear=.003
*   difsub=.027 difside=.017 pdifsub=.000 pdifside=.000
C92 94 2 2.6410F
*   total=.046 PF  gate=.000 linear=.003
*   difsub=.027 difside=.017 pdifsub=.000 pdifside=.000
C93 95 2 2.6410F
*   total=.046 PF  gate=.000 linear=.003
*   difsub=.027 difside=.017 pdifsub=.000 pdifside=.000
C98 101 2 5.3280F
*   total=.052 PF  gate=.000 linear=.005
*   difsub=.000 difside=.000 pdifsub=.034 pdifside=.013
C100 103 2 5.4140F
*   total=.052 PF  gate=.000 linear=.005
*   difsub=.000 difside=.000 pdifsub=.034 pdifside=.013
C102 105 2 5.4140F
*   total=.052 PF  gate=.000 linear=.005
*   difsub=.000 difside=.000 pdifsub=.034 pdifside=.013
C104 107 2 5.4130F
*   total=.052 PF  gate=.000 linear=.005
*   difsub=.000 difside=.000 pdifsub=.034 pdifside=.013

**********************************************************************
*****                      Transistors                           *****
**********************************************************************
M1   44       45       6        6 P L=2U W=18U AS=35.10P PS=7.20U
+                                               AD=45P    PD=5U
M2   6        45       44       6 P L=2U W=18U AS=45P    PS=5U
+                                               AD=35.10P PD=7.20U
```

```
M3   8        9          61       6 P L=2U W=12U AS=42P     PS=13U
+                                                AD=30P     PD=5U
M4   61       44         8        6 P L=2U W=12U AS=30P     PS=5U
+                                                AD=42P     PD=13U
M5   6        10         61       6 P L=2U W=12U AS=42P     PS=13U
+                                                AD=35.10P  PD=7.20U
M6   61       %pen_out   6        6 P L=2U W=12U AS=35.10P  PS=7.20U
+                                                AD=42P     PD=13U
M7   38       36         62       6 P L=2U W=12U AS=40P     PS=10.67U
+                                                AD=36P     PD=6U
M8   62       44         38       6 P L=2U W=12U AS=36P     PS=6U
+                                                AD=40P     PD=10.67U
M9   6        46         62       6 P L=2U W=12U AS=40P     PS=10.67U
+                                                AD=35.10P  PD=7.20U
M10  63       47         6        6 P L=2U W=12U AS=35.10P  PS=7.20U
+                                                AD=15P     PD=2.50U
M11  46       48         63       6 P L=2U W=12U AS=15P     PS=2.50U
+                                                AD=54P     PD=21U
M12  39       33         64       6 P L=2U W=12U AS=42P     PS=13U
+                                                AD=36P     PD=6U
M13  64       44         39       6 P L=2U W=12U AS=36P     PS=6U
+                                                AD=42P     PD=13U
M14  6        32         64       6 P L=2U W=12U AS=42P     PS=13U
+                                                AD=35.10P  PD=7.20U
M15  64       49         6        6 P L=2U W=12U AS=35.10P  PS=7.20U
+                                                AD=42P     PD=13U
M16  40       30         65       6 P L=2U W=12U AS=40P     PS=10.67U
+                                                AD=36P     PD=6U
M17  65       44         40       6 P L=2U W=12U AS=36P     PS=6U
+                                                AD=40P     PD=10.67U
M18  6        50         65       6 P L=2U W=12U AS=40P     PS=10.67U
+                                                AD=35.10P  PD=7.20U
M19  66       51         6        6 P L=2U W=12U AS=35.10P  PS=7.20U
+                                                AD=15P     PD=2.50U
M20  50       52         66       6 P L=2U W=12U AS=15P     PS=2.50U
+                                                AD=54P     PD=21U
M21  41       27         67       6 P L=2U W=12U AS=42P     PS=13U
+                                                AD=36P     PD=6U
M22  67       44         41       6 P L=2U W=12U AS=36P     PS=6U
+                                                AD=42P     PD=13U
```

```
M23  6       26       67       6 P L=2U W=12U AS=42P     PS=13U
+                                              AD=35.10P PD=7.20U
M24  67      53       6        6 P L=2U W=12U AS=35.10P PS=7.20U
+                                              AD=42P    PD=13U
M25  42      24       68       6 P L=2U W=12U AS=40P     PS=10.67U
+                                              AD=36P    PD=6U
M26  68      44       42       6 P L=2U W=12U AS=36P     PS=6U
+                                              AD=40P    PD=10.67U
M27  6       54       68       6 P L=2U W=12U AS=40P     PS=10.67U
+                                              AD=35.10P PD=7.20U
M28  69      55       6        6 P L=2U W=12U AS=35.10P PS=7.20U
+                                              AD=15P    PD=2.50U
M29  54      56       69       6 P L=2U W=12U AS=15P     PS=2.50U
+                                              AD=54P    PD=21U
M30  43      21       70       6 P L=2U W=12U AS=42P     PS=13U
+                                              AD=36P    PD=6U
M31  70      44       43       6 P L=2U W=12U AS=36P     PS=6U
+                                              AD=42P    PD=13U
M32  6       20       70       6 P L=2U W=12U AS=42P     PS=13U
+                                              AD=35.10P PD=7.20U
M33  70      57       6        6 P L=2U W=12U AS=35.10P PS=7.20U
+                                              AD=42P    PD=13U
M34  14      19       71       6 P L=2U W=12U AS=40P     PS=10.67U
+                                              AD=36P    PD=6U
M35  71      44       14       6 P L=2U W=12U AS=36P     PS=6U
+                                              AD=40P    PD=10.67U
M36  6       58       71       6 P L=2U W=12U AS=40P     PS=10.67U
+                                              AD=35.10P PD=7.20U
M37  72      59       6        6 P L=2U W=12U AS=35.10P PS=7.20U
+                                              AD=15P    PD=2.50U
M38  58      60       72       6 P L=2U W=12U AS=15P     PS=2.50U
+                                              AD=54P    PD=21U
M39  12      17       73       6 P L=2U W=12U AS=42P     PS=13U
+                                              AD=36P    PD=6U
M40  73      44       12       6 P L=2U W=12U AS=36P     PS=6U
+                                              AD=42P    PD=13U
M41  6       16       73       6 P L=2U W=12U AS=42P     PS=13U
+                                              AD=35.10P PD=7.20U
M42  73      %pen_in  6        6 P L=2U W=12U AS=35.10P PS=7.20U
+                                              AD=42P    PD=13U
```

```
M43  44      45      2       2 N L=2U W=9U   AD=22.50P PD=5U
M44  2       45      44      2 N L=2U W=9U   AS=22.50P PS=5U
M45  74      33      2       2 N L=2U W=6U   AD=9P     PD=3U
M46  39      44      74      2 N L=2U W=6U   AS=9P     PS=3U
+                                            AD=15P    PD=5U
M47  75      32      39      2 N L=2U W=6U   AS=15P    PS=5U
+                                            AD=9P     PD=3U
M48  2       49      75      2 N L=2U W=6U   AS=9P     PS=3U
M49  76      27      2       2 N L=2U W=6U   AD=9P     PD=3U
M50  41      44      76      2 N L=2U W=6U   AS=9P     PS=3U
+                                            AD=15P    PD=5U
M51  77      26      41      2 N L=2U W=6U   AS=15P    PS=5U
+                                            AD=9P     PD=3U
M52  2       53      77      2 N L=2U W=6U   AS=9P     PS=3U
M53  78      21      2       2 N L=2U W=6U   AD=9P     PD=3U
M54  43      44      78      2 N L=2U W=6U   AS=9P     PS=3U
+                                            AD=15P    PD=5U
M55  79      20      43      2 N L=2U W=6U   AS=15P    PS=5U
+                                            AD=9P     PD=3U
M56  2       57      79      2 N L=2U W=6U   AS=9P     PS=3U
M57  80      17      2       2 N L=2U W=6U   AD=9P     PD=3U
M58  12      44      80      2 N L=2U W=6U   AS=9P     PS=3U
+                                            AD=15P    PD=5U
M59  81      16      12      2 N L=2U W=6U   AS=15P    PS=5U
+                                            AD=9P     PD=3U
M60  2       %pen_in 81      2 N L=2U W=6U   AS=9P     PS=3U
M61  82      9       2       2 N L=2U W=6U   AD=9P     PD=3U
M62  8       44      82      2 N L=2U W=6U   AS=9P     PS=3U
+                                            AD=15P    PD=5U
M63  83      10      8       2 N L=2U W=6U   AS=15P    PS=5U
+                                            AD=9P     PD=3U
M64  2       %pen_out 83     2 N L=2U W=6U   AS=9P     PS=3U
M65  84      36      2       2 N L=2U W=6U   AD=9P     PD=3U
M66  38      44      84      2 N L=2U W=6U   AS=9P     PS=3U
+                                            AD=13P    PD=6U
M67  2       46      38      2 N L=2U W=4U   AS=13P    PS=6U
M68  46      47      2       2 N L=2U W=4U   AD=10P    PD=5U
M69  2       48      46      2 N L=2U W=4U   AS=10P    PS=5U
M70  85      30      2       2 N L=2U W=6U   AD=9P     PD=3U
M71  40      44      85      2 N L=2U W=6U   AS=9P     PS=3U
```

```
+                                                    AD=13P     PD=6U
M72   2          50         40         2 N L=2U W=4U  AS=13P     PS=6U
M73   50         51         2          2 N L=2U W=4U  AD=10P     PD=5U
M74   2          52         50         2 N L=2U W=4U  AS=10P     PS=5U
M75   86         24         2          2 N L=2U W=6U  AD=9P      PD=3U
M76   42         44         86         2 N L=2U W=6U  AS=9P      PS=3U
+                                                    AD=13P     PD=6U
M77   2          54         42         2 N L=2U W=4U  AS=13P     PS=6U
M78   54         55         2          2 N L=2U W=4U  AD=10P     PD=5U
M79   2          56         54         2 N L=2U W=4U  AS=10P     PS=5U
M80   87         19         2          2 N L=2U W=6U  AD=9P      PD=3U
M81   14         44         87         2 N L=2U W=6U  AS=9P      PS=3U
+                                                    AD=13P     PD=6U
M82   2          58         14         2 N L=2U W=4U  AS=13P     PS=6U
M83   58         59         2          2 N L=2U W=4U  AD=10P     PD=5U
M84   2          60         58         2 N L=2U W=4U  AS=10P     PS=5U
M85   48         88         92         2 N L=2U W=13U AS=41.17P  PS=10.67U
+                                                    AD=32.50P  PD=5U
M86   92         32         48         2 N L=2U W=13U AS=32.50P  PS=5U
+                                                    AD=41.17P  PD=10.67U
M87   2          49         92         2 N L=2U W=13U AS=41.17P  PS=10.67U
M88   52         89         93         2 N L=2U W=13U AS=41.17P  PS=10.67U
+                                                    AD=32.50P  PD=5U
M89   93         26         52         2 N L=2U W=13U AS=32.50P  PS=5U
+                                                    AD=41.17P  PD=10.67U
M90   2          53         93         2 N L=2U W=13U AS=41.17P  PS=10.67U
M91   56         90         94         2 N L=2U W=13U AS=41.17P  PS=10.67U
+                                                    AD=32.50P  PD=5U
M92   94         20         56         2 N L=2U W=13U AS=32.50P  PS=5U
+                                                    AD=41.17P  PD=10.67U
M93   2          57         94         2 N L=2U W=13U AS=41.17P  PS=10.67U
M94   60         91         95         2 N L=2U W=13U AS=41.17P  PS=10.67U
+                                                    AD=32.50P  PD=5U
M95   95         16         60         2 N L=2U W=13U AS=32.50P  PS=5U
+                                                    AD=41.17P  PD=10.67U
M96   2          %pen_in    95         2 N L=2U W=13U AS=41.17P  PS=10.67U
M97   96         37         2          2 N L=2U W=7U  AD=8.75P   PD=2.50U
M98   %pen_out  47         96         2 N L=2U W=7U  AS=8.75P   PS=2.50U
+                                                    AD=17.50P  PD=5U
M99   2          48         %pen_out   2 N L=2U W=7U  AS=17.50P  PS=5U
```

```
M100 97        31        2         2 N L=2U W=7U   AD=8.75P   PD=2.50U
M101 49        51        97        2 N L=2U W=7U   AS=8.75P   PS=2.50U
+                                                  AD=17.50P  PD=5U
M102 2         52        49        2 N L=2U W=7U   AS=17.50P  PS=5U
M103 98        25        2         2 N L=2U W=7U   AD=8.75P   PD=2.50U
M104 53        55        98        2 N L=2U W=7U   AS=8.75P   PS=2.50U
+                                                  AD=17.50P  PD=5U
M105 2         56        53        2 N L=2U W=7U   AS=17.50P  PS=5U
M106 99        15        2         2 N L=2U W=7U   AD=8.75P   PD=2.50U
M107 57        59        99        2 N L=2U W=7U   AS=8.75P   PS=2.50U
+                                                  AD=17.50P  PD=5U
M108 2         60        57        2 N L=2U W=7U   AS=17.50P  PS=5U
M109 2         %den_in   45        2 N L=2U W=6U   AS=27P     PS=15U
M110 88        34        2         2 N L=2U W=4U   AD=20P     PD=14U
M111 89        28        2         2 N L=2U W=4U   AD=20P     PD=14U
M112 90        22        2         2 N L=2U W=4U   AD=20P     PD=14U
M113 91        13        2         2 N L=2U W=4U   AD=20P     PD=14U
M114 47        35        2         2 N L=2U W=4U   AD=20P     PD=14U
M115 51        29        2         2 N L=2U W=4U   AD=20P     PD=14U
M116 55        23        2         2 N L=2U W=4U   AD=20P     PD=14U
M117 59        18        2         2 N L=2U W=4U   AD=20P     PD=14U
M118 %pen_out  37        101       5 P L=2U W=13U  AS=41.17P  PS=10.67U
+                                                  AD=32.50P  PD=5U
M119 101       47        %pen_out  5 P L=2U W=13U  AS=32.50P  PS=5U
+                                                  AD=41.17P  PD=10.67U
M120 5         48        101       5 P L=2U W=13U  AS=41.17P  PS=10.67U
+                                                  AD=40P     PD=15.29U
M121 47        35        5         5 P L=2U W=6U   AS=40P     PS=15.29U
+                                                  AD=30P     PD=16U
M122 102       88        5         5 P L=2U W=13U  AS=40P     PS=15.29U
+                                                  AD=16.25P  PD=2.50U
M123 48        32        102       5 P L=2U W=13U  AS=16.25P  PS=2.50U
+                                                  AD=34.25P  PD=6.50U
M124 49        31        103       5 P L=2U W=13U  AS=41.17P  PS=10.67U
+                                                  AD=32.50P  PD=5U
M125 103       51        49        5 P L=2U W=13U  AS=32.50P  PS=5U
+                                                  AD=41.17P  PD=10.67U
M126 5         52        103       5 P L=2U W=13U  AS=41.17P  PS=10.67U
+                                                  AD=40P     PD=15.29U
M127 51        29        5         5 P L=2U W=6U   AS=40P     PS=15.29U
```

```
+                                                    AD=30P     PD=16U
M128 104       89        5        5 P L=2U W=13U    AS=40P     PS=15.29U
+                                                    AD=16.25P  PD=2.50U
M129 52        26        104      5 P L=2U W=13U    AS=16.25P  PS=2.50U
+                                                    AD=34.25P  PD=6.50U
M130 53        25        105      5 P L=2U W=13U    AS=41.17P  PS=10.67U
+                                                    AD=32.50P  PD=5U
M131 105       55        53       5 P L=2U W=13U    AS=32.50P  PS=5U
+                                                    AD=41.17P  PD=10.67U
M132 5         56        105      5 P L=2U W=13U    AS=41.17P  PS=10.67U
+                                                    AD=40P     PD=15.29U
M133 55        23        5        5 P L=2U W=6U     AS=40P     PS=15.29U
+                                                    AD=30P     PD=16U
M134 106       90        5        5 P L=2U W=13U    AS=40P     PS=15.29U
+                                                    AD=16.25P  PD=2.50U
M135 56        20        106      5 P L=2U W=13U    AS=16.25P  PS=2.50U
+                                                    AD=34.25P  PD=6.50U
M136 57        15        107      5 P L=2U W=13U    AS=41.17P  PS=10.67U
+                                                    AD=32.50P  PD=5U
M137 107       59        57       5 P L=2U W=13U    AS=32.50P  PS=5U
+                                                    AD=41.17P  PD=10.67U
M138 5         60        107      5 P L=2U W=13U    AS=41.17P  PS=10.67U
+                                                    AD=40P     PD=15.29U
M139 59        18        5        5 P L=2U W=6U     AS=40P     PS=15.29U
+                                                    AD=30P     PD=16U
M140 108       91        5        5 P L=2U W=13U    AS=40P     PS=15.29U
+                                                    AD=16.25P  PD=2.50U
M141 60        16        108      5 P L=2U W=13U    AS=16.25P  PS=2.50U
+                                                    AD=34.25P  PD=6.50U
M142 5         %den_in   45       5 P L=2U W=12U    AS=54P     PS=21U
+                                                    AD=40P     PD=15.29U
M143 5         49        48       5 P L=2U W=11U    AS=34.25P  PS=6.50U
+                                                    AD=40P     PD=15.29U
M144 5         53        52       5 P L=2U W=11U    AS=34.25P  PS=6.50U
+                                                    AD=40P     PD=15.29U
M145 5         57        56       5 P L=2U W=11U    AS=34.25P  PS=6.50U
+                                                    AD=40P     PD=15.29U
M146 5         %pen_in   60       5 P L=2U W=11U    AS=34.25P  PS=6.50U
+                                                    AD=40P     PD=15.29U
M147 5         34        88       5 P L=2U W=6U     AS=27P     PS=15U
```

```
+                                                    AD=40P    PD=15.29U
M148 5        28        89        5 P L=2U W=6U      AS=27P    PS=15U
+                                                    AD=40P    PD=15.29U
M149 5        22        90        5 P L=2U W=6U      AS=27P    PS=15U
+                                                    AD=40P    PD=15.29U
M150 5        13        91        5 P L=2U W=6U      AS=27P    PS=15U
+                                                    AD=40P    PD=15.29U

*******************************************************************************
*****                          Capacitors                                 *****
*******************************************************************************
CPENOUT %pen_out 2 0.2PF

*******************************************************************************
*****                        Voltage Sources                              *****
*******************************************************************************
VDD1    6         0  5
VDD2    5         0  5
VSS     2         0  0.1

Vpin7   16        0  5
Vpin6   18        0  5
Vpin5   20        0  5
Vpin4   23        0  5
Vpin3   26        0  5
Vpin2   29        0  5
Vpin1   32        0  5
Vpin0   35        0  5
Vppar   10        0  5

Vdata7  17        0  0
Vdata6  19        0  0
Vdata5  21        0  0
Vdata4  24        0  0
Vdata3  27        0  0
Vdata2  30        0  0
Vdata1  33        0  0
Vdata0  36        0  0
Vdpar   9         0  0
```

```
Vbin7   13        0   5
Vbin6   15        0   5
Vbin5   22        0   5
Vbin4   25        0   5
Vbin3   28        0   5
Vbin2   31        0   5
Vbin1   34        0   5
Vbin0   37        0   5
Vbpar   7         0   5

VPEN    %pen_in 0   PULSE(0 5 2n 1n 1n 19n 40n)
VDEN    %den_in 0   0

.END
```