AP-733

APPLICATION NOTE

Switched Ethernet Reference Design Description

Rod Mullendore
SPG 80960 Applications Engineer

Intel Corporation
Semiconductor Products Group
Mail Stop CH6-412
5000 W. Chandler Blvd.
Chandler, Arizona 85226

July 23, 1996

Order Number: 272907-001

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect IL 60056-764
or call 1-800-548-4725

Switched Ethernet Reference Design Description

FIGURES

TABLES

1.0 Introduction

1.1 Purpose

This document describes a reference design for a 24-port 10BASE-T switched ethernet hub. The reference design is based on the Galileo Technology GT-48001 Switched Ethernet Controller (SEC) and an Intel i960® microprocessor. The GT-48001 provides the ethernet switching functions while the i960 processor provides network management and control functions.

1.2 Reference Information

1.2.1 Intel Documentation

Intel documentation is available from your Intel Sales Representative or Intel Literature Sales.

Intel Corporation
Literature Sales P.O. Box 7641
Mt. Prospect, IL 60056-764
1-800-879-46831

The following resources are referenced in this document.

Document Title	Order/Contact
i960® Jx Microprocessor User's Manual	Intel Order # 272483
i960® Cx Microprocessor User's Manual	Intel Order # 270710
i960® Hx Microprocessor User's Manual	Intel Order # 272484
PCI Bus Interface and Clock Distribution Chips Product Catalog	PLX Technology, Inc.
Am8530H/Am85C30 Serial Communications Controller Technical Manual	Advanced Micro Devices
PCI Local Bus Specification Revision 2.1	PCI Special Interest Group 1-800-433-5177
GT-48001 Product Specification Revision 1.0	Galileo Technology, Inc.
PCI System Architecture by Tom Shanley and Don Anderson of Mindshare, Inc.	Publisher: Addison-Wesley ISBN: 0-201-40993-3

1.2.2 Appendices

The appendices to this document are available on Intel's World Wide Web location at http://www.intel.com/embedded/products/index.html. The appendices can be viewed and printed using the Adobe Acrobat Reader, which is available at http://www.adobe.com.

1.2.3 Software

Low level hardware device drivers for the reference design have been developed by Logic Solutions, Inc.

Logic Solutions has also ported an SNMP/RMON software stack from Routerware to the reference design. The device drivers are free and can be obtained from the following bulletin board service (BBS):

Intel Corporation
America's Application Support BBS
(916) 356-3600

For information on customizing the SNMP/RMON software stack, contact Logic Solutions at:

Logic Solutions, Inc.
371 Moddy St.
Suite 109
Waltham, MA 02154
(617) 647-4885
E-mail: lsi@world.com

For information on licensing of the SNMP/RMON software stack contact Routerware at:

Routerware
3961 MacArthur Blvd.
Suite 212
Newport Beach, CA 92660
(714) 442-0770

1.2.4 Availability of Reference Hubs

Intel has produced a small quantity (10) of the hub reference design, which are available to local Intel sales offices. The reference hubs may be loaned to selected customers who wish to evaluate the design's performance. Interested customers should contact their local Intel sales office for information on borrowing a reference hub.

1.3 Important Information Regarding the Reference Design

1.3.1 Modifications to the Physical Interface

The 10BASE-T physical interface described in this document was modified slightly. The 22 KΩ resistors that set the reference levels for the differential receivers were removed. This was necessary because the input circuitry of the 26LS32 devices already provides a bias voltage at the associated reference point. The approximate thevenin equivalent is a +2.5 V source through a 4800 Ω resistor. As a result, the reference voltage was approximately 500 mV when the 22 KΩ resistors were installed and 360 mV when the 22 KΩ resistors were not installed. The voltage at the "+" terminal of the RxD receiver is ≈ 560 mV when the receiver output is a logic "1" and ≈ 230 mV when the receiver output is a "0". Increasing the value of the resistors that set the reference voltage (R650-R673) to 910 Ω would shift the reference level (≈ 398 mV) closer to the middle of the hysteresis thresholds.

Biasing the center tap of the transformer to +2.5 V (as done in Galileo Technology's reference physical interface) eliminates most of the effects of the 26LS32 input circuitry on the hysteresis. The number of components is reduced when the center tap is grounded, but the hysteresis becomes dependent on the 26LS32 input circuitry.

1.3.2 Performance Testing

Intel has not tested the reference design to determine packet switching performance. Interested companies may be able to borrow a reference hub to perform their own testing or can contact Galileo Technology for information on the GT-48001.

1.3.3 Compliance Testing

The reference design described in this document has not been subjected to any testing related to EMI or safety compliance.

1.3.4 Before Starting a Design

Before starting a design, contact Galileo Technology for the latest information on the GT-48001.

1.3.5 Miscellaneous Notes and Questions

- The PCI expansion connector was not tested.
- The RSTQUEUE# pins on each GT-48001 device were disconnected from the output port on the reference hubs. As a result, these pins are always pulled to a logic "1" (i.e., deasserted) through resistors.

The following questions have arisen regarding the reference design.

Q: *Why is there a separate 80 MHz oscillator for each GT-48001?*

A: A separate 80 MHz oscillator was used to ensure the clock integrity of the 80 MHz signal to each GT-48001 device. The oscillators were placed close to each GT-48001, allowing a very short trace from the oscillator output to the GT-48001. A single 80 MHz oscillator could be used if the PC board trace that routes the signal to each GT-48001 is properly terminated to ensure good signal integrity at each GT-48001 device. This can be accomplished by terminating the signal trace in the characteristic impedance of the trace.

Q: *Why are there two RS-232 ports on the reference design?*

A: One RS-232 port is intended to be dedicated for use by the monitor/debugger (i.e., MON960) and the other RS-232 port is intended to be connected to a terminal for displaying/modifying management information.

1.4 Notational Conventions and Abbreviations

The following notation conventions are consistent with other i960 processor documentation and generic "industry standards."

#	Pound symbol (#) appended to a signal name indicates signal is active low.
`typewriter font`	This proportional font is used for code examples. All characters are equal width; this is useful for maintaining accurate character spacing (e.g., the letter "i" is the same width as the letter "m").
UPPERCASE	In text, signal names are shown in uppercase. When several signals share a common name, each signal is represented by the signal name followed by a number; the group is represented by the signal name followed by a variable (*n*). e.g., interrupt request signals are named IRQ3, IRQ4, ... and collectively called IRQ*n*. In code examples, signal names are shown in the case required by the software development tool in use.
Number designations for hex, decimal, binary	In text — instead of using subscripted "base" designators (e.g., FF_{16}) or leading "0x" (e.g., 0xFF) — hexadecimal numbers are represented by a string of hex digits followed by the letter *H*. A zero prefix is added to numbers that begin with *A* through *F*. (e.g., *FF* is shown as *0FFH*.) In examples of actual code, "0x" is used. Decimal and binary numbers are represented by their customary notations. (e.g., 255 is a decimal number and 1111 1111 is a binary number. In some cases, the letter *B* is added for clarity.)

Units of Measure

NOTE: Ones listed are frequently used; other units and symbols are used as necessary.

A	amps, amperes
Kbit, Kbyte	kilobits, kilobytes
KW	kilo-ohms
mA	milliamps, milliamperes
Mbit, Mbyte	megabits, megabytes
KHz, MHz	kilohertz, megahertz
ms	milliseconds
ns	nanoseconds
µs	microseconds
µF	microfarads
W	watts
V	volts

The following abbreviations and acronyms are used in this document.

AUI	Attachment Unit Interface
CAS	Column Address Strobe
CBR	CAS Before RAS
CSMA/CD	Carrier Sense Multiple Access/Collision Detect
DIP	Dual In-Line Package
DRAM	Dynamic Random Access Memory
DUART	Dual UART
EDO	Extended Data Out
EPROM	Electrically Programmable Read Only Memory
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FPM	Fast Page Mode
IBR	Initialization Boot Record
IDC	Insulation Displacement Connector
IEEE	Institute of Electrical and Electronic Engineers
LED	Light Emitting Diode
MAC	Media Access Control
MAU	Media Attachment Unit
MIB	Management Information Base
MPR	Multi-Port Repeater
NIC	Network Interface Card
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PLCC	Plastic Leaded Chip Carrier
PQFP	Plastic Quad Flat Pack
RAS	Row Address Strobe
SEC	Switched Ethernet Controller
SIG	Special Interest Group
SIMM	Single In-line Memory Module
TP	Twisted Pair
TSOP	Thin Small Outline Package
UART	Universal Asynchronous Receiver/Transmitter
UTP	Unshielded Twisted Pair

2.0 Switched Ethernet Background

Ethernet networks were originally based on a single shared coaxial cable. This design permits only one of the devices attached to the network to transmit at any given time. The method for determining which device can transmit is known as CSMA/CD (Carrier Sense Multiple Access/Collision Detect). With a single shared cable, a device must monitor the cable and wait until there are no transmissions occurring before attempting to transmit. A problem can result if two or more devices attempt to transmit at approximately the same time, causing a "collision." Thus, it is necessary for devices to incorporate collision detection. When a device transmits, it also checks the cable to see if the received and transmitted data match. A failure to produce matching data indicates that multiple devices are driving the cable, resulting in corrupted data. When a collision is detected, a device must terminate its transmission (i.e., backoff), then wait a specific period before attempting to transmit again. The wait period is defined in the IEEE standard that governs ethernet networks (IEEE 802.3). The period is based on a random number generator such that the devices are unlikely to attempt to transmit again at the same time.

The coaxial cable versions of ethernet are known as 10BASE-5 and 10BASE-2. The *10* refers to the data rate (10 MHz), *BASE* refers to baseband transmission, and *-5* and *-2* refer to the type of cable. 10BASE-5 ("thick ethernet") is the original ethernet and uses a thick coaxial cable. 10BASE-5 can have a maximum segment length of 500 meters. 10BASE-2 ("thin ethernet") uses a thinner coaxial cable and can have a maximum segment length of 185 meters.

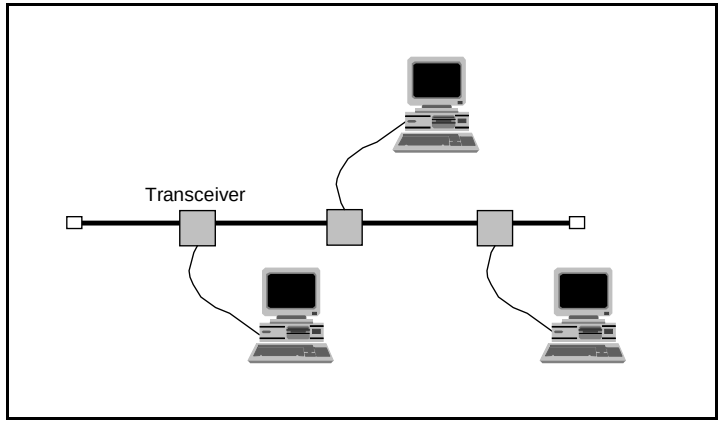


Figure 1. 10BASE-5 Implementation

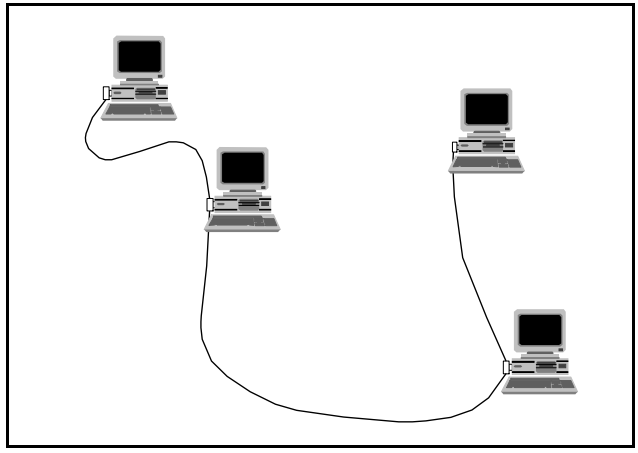


Figure 2. 10BASE-2 Implementation

2.1 10BASE-T Ethernet

For the past several years, most ethernet networks have been based on 10BASE-T cabling, which uses unshielded twisted pair (UTP) cable. The main reasons for the popularity of 10BASE-T are cost and ease of maintenance and installation. The use of UTP cable allows the use of existing phone cable to implement computer networks. The implementation of ethernet over UTP is different than for 10BASE-5 or 10BASE-2. 10BASE-T uses two pairs of wires (four wires total), with one pair for transmitting data and one pair for receiving data. Having separate receiving and transmitting wire pairs prevents directly connecting more than two computers together. For 10BASE-T, a device is needed to route traffic from the transmitting wire pairs onto the receiving wire pairs. Multi-port repeaters (MPR) were developed for this purpose. An MPR (also known as a hub or concentrator) makes the UTP cabling appear as a single cable in the following manner:

1. When a device transmits data, the data is “repeated” onto all of the receive wire pairs.
2. When two devices attempt to transmit at the same time, the MPR simulates a collision by transmitting an incorrect pattern on all receive wire pairs.

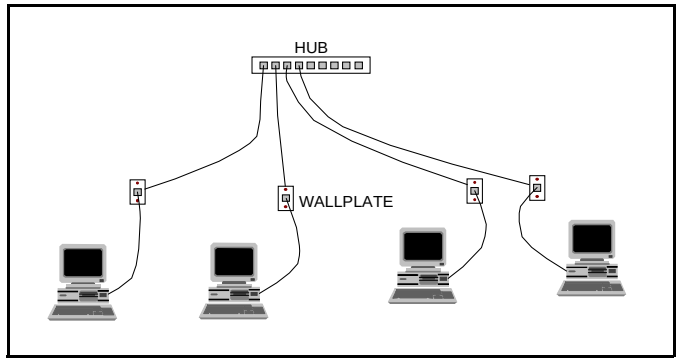


Figure 3. 10BASE-T Implementation

Thus, the MPR makes 10BASE-T cabling act the same as 10BASE-5 and 10BASE-2 cabling to a network interface card (NIC). In fact, this was necessary for legacy reasons, since the existing NICs were designed for 10BASE-5 and 10BASE-2. Most NICs have an Attachment Unit Interface (AUI), which is a generic interface not specific to a type of cable. Another device, the media attachment unit (MAU) provides the interface from the AUI port to the actual cable used. MAUs exist for connecting to all of the various ethernet cabling used. When 10BASE-T first became popular, the NICs were connected via 10BASE-T MAUs. Newer NIC cards almost always provide an AUI port and a 10BASE-T port. The AUI port is usually provided to allow compatibility, via an MAU, to ethernet cabling other than 10BASE-T.

Two new technologies have been advanced recently to improve the performance of 10BASE-T networks. These technologies, switched ethernet and full-duplex ethernet, take advantage of 10BASE-T's two-way path for data communications (i.e., a wire pair for receiving data and a wire pair for transmitting data).

2.2 Switching Hubs

Switched ethernet makes the hub intelligent by reducing the number of ports to which data is repeated. Consider the following examples:

A repeating hub connects four devices: A, B, C, and D. A wants to transmit a packet to B and C wants to transmit a packet to D at the same time. The sequence of events may proceed as follows:

1. A and C detect that the cabling is idle (no transmissions) and both begin to transmit at essentially the same time, creating a collision. The hub detects data transmitted from two or more ports and simulates a collision to all ports. A and C detect the collision, terminate their transmissions, and "backoff" for a random time.

2. C's random backoff timer expires and the device starts to transmit its data packet. The hub repeats C's transmission to all ports (A, B, C, and D). The devices connected to ports A and B ignore the data packet received.

3. A detects no activity on the network and transmits its data packet to B. The hub repeats A's transmission to all ports, with C and D ignoring the received packet.

A “switching” hub treats this scenario in a more intelligent manner. The sequence of events would proceed as follows:

1. A and C detect that the cabling is idle (no transmissions) and both begin to transmit at essentially the time. The hub repeats the received transmission from A back to A and the received transmission from C back to C. The interfaces to B and D are idle at this time.

2. The hub buffers the data packets received from A and C then analyzes the packets to determine their destination. The hub then retransmits the packet received from A to B and the packet received from C to D.

With a switching hub, the two packets can now be transmitted in parallel, although there is a delay due to the buffering of the data packets within the hub. Consider the same example, but in this case A and C are both transmitting to B. In this case, the switching hub again receives both packets, but after analyzing the destination addresses, it sends the packets serially to B.

From these examples, it is apparent that the switching hub must maintain two buffers or queues for each of its ports: an input buffer and an output buffer. When the hub receives data packets from a port, the data packets are entered into the port’s input buffer. The packets in a port’s input buffer are then analyzed to determine the destination. The packets are then copied to the destination port’s output buffer. For multi-cast and broadcast addresses, the packets are copied to the output buffer of all ports (except the port from which it was sent).

A switching hub must know the addresses of device(s) connected to each of its ports. In general, a switching hub learns the device addresses for a port by monitoring the source addresses in packets received from that port. When a packet is received, the destination address is compared against the known device addresses. If the address is found, the packet is entered in the destination port’s output buffer. If the address is not found, the packet is entered into the output buffer of all ports except for the port on which it was received.

Switching hubs can become overloaded and lose data if several ports send data packets to the same destination. The switching hub is able to buffer the packets for a limited period of time, determined by the size of its buffer memory. When the switching hub’s buffer memory is exhausted, the hub can discard packets it receives, resulting in the need for those packets to be retransmitted at a later time. Another approach taken by switching hubs is to simulate collisions when its internal buffers are full. This method is known as “back pressure” and works only for half-duplex ethernet. The advantage to this approach is that the device sending data to the hub knows immediately that the packet has not been transmitted successfully. When the hub discards a packet, the upper-layer network protocols must detect a missing packet and request that it be retransmitted. Therefore, it is important that a switching hub either have a large buffer memory or a method for delaying data transmission, such as “back pressure.”

2.3 Full-Duplex Ethernet

With full-duplex ethernet, an NIC does not implement collision detection, since data can be received and transmitted at the same time, effectively doubling the bandwidth of the interface. However, it must appear to a NIC that there is only one device at the other end of the cable (i.e., there can be no collisions). A switching hub eliminates collisions by buffering received data until the data can be transmitted.

2.4 Managed and Unmanaged Hubs

The need for network management has increased dramatically as networks have become an integral part of a company's day-to-day operations. With this reliance on the data network, the ability to determine how a network is operating is critical to improving performance and detecting problems early. Many network devices today have a built-in network management function that collects information on the device and/or allows the device to be configured. This management function may be accessed using several methods. For example, the device could be controlled/monitored locally via a terminal or remotely via the network or modem.

A switching hub can be either unmanaged or managed. An unmanaged hub has the basic switching circuits while a managed hub adds a processor that collects and proceses information from the hub. An unmanaged hub is cheaper, because there is less circuitry, and costs less to develop (particularly since there is no software development required). However, building a network around an unmanaged hub may provide far less performance than can be realized with managed hubs. This results from the fact that determining the best configuration can be very difficult. Data networks are often partitioned to even the load across the network components (hubs, bridges, routers, etc.). With a managed hub, it is relatively easy to determine the usage pattern for the various ports. This information can lead to a more optimal network configuration.

Although unmanaged hubs may be cheaper in the short term, managed hubs are often less expensive in the long term due to increased productivity and network utilization. This is especially true with organizations whose networks and users are constantly changing. An optimum configuration one day may be a poor solution the next.

3.0 Design Information

This document describes a reference design for a managed switched ethernet hub that provides support for full-duplex ethernet. This design is intended to form the basis from which an organization can design a switched ethernet product.

3.1 Features

The design presented in this document has the following features.

- 24 switched 10BASE-T ethernet ports provided by three GT-48001 devices from Galileo Technology.
  - — RMON Data acquisition via 1 Kx32 FIFO
  - — Four status LEDs per port
- i960 processor for network management and control.
  - — Allows 80960Jx, Cx, or Hx processor to be installed for evaluation.
- PCI Bus
  - — Interconnection of GT-48001 devices and i960 processor.
  - — One expansion PCI connector.
- Boot Block Flash
  - — 1 Mbyte to 4 Mbyte
- DRAM
  - — 1 Mbyte to 16 Mbyte banks
  - — Two SIMM sockets
  - — Fast Page Mode or EDO
- 128 Kx8 EPROM
- 64x16 serial EEPROM
- Dual serial ports

3.2 Reference Design Overview

Figure 4 shows a block diagram of the reference design. Figure 5 shows the "8 Port Switch" block in more detail.

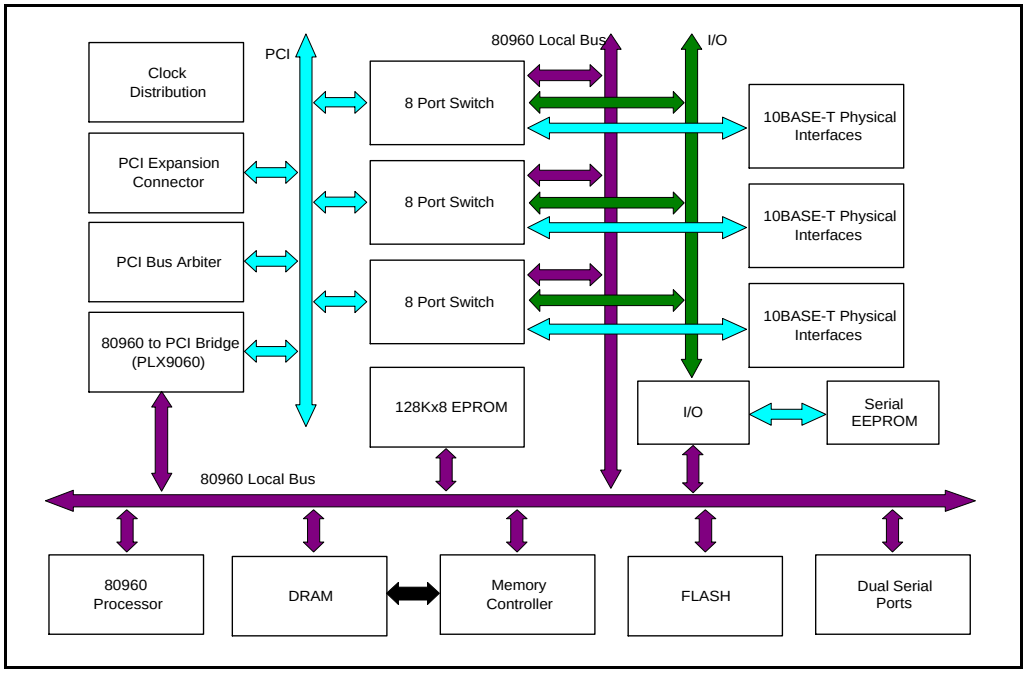


Figure 4. Reference Design Block Diagram

The reference design implements 24 ethernet ports using three GT-48001 Switched Ethernet Controller (SEC) ICs from Galileo Technology, Inc. The physical interface for all 24 of the ethernet ports is 10BASE-T. The “8 Port Switch” block contains an SEC, DRAM for the SEC, LED interface cicuitry, a FIFO, and an 80 MHz clock oscillator. A PCI bus provides the interface between the SECs and an i960 processor. A FIFO that collects RMON information provides a second interface between each SEC and the processor’s local bus. The input side of the FIFO connects to the SEC and the output side connects to the processor’s local bus. I/O signals are also routed to the eight Port Switch blocks to allow the processor to initialize the SEC chips and control the LED interface (for testing).

The PCI bus arbiter block, implemented with an FPGA, provides bus arbitration and bus parking functions. A PCI expansion connector has been provided to allow PCI-based boards to be interfaced when the reference design is used for evaluation purposes. For example, a fast ethernet, ATM, or FDDI board could be installed to allow development of a 10 Mbit switched ethernet hub that has one high-speed port.

The clock distribution block provides low skew clocks for the PCI interface devices and the processor-related devices. The design provides for separate PCI and processor clocks. For example, the PCI bus could operate at 33 MHz while the processor bus could operate at 25 MHz.

The design allows an i960 processor from the 80960Jx, 80960Cx, or 80960Hx families to be installed. The remaining blocks provide memory (DRAM, FLASH, EPROM) and two serial ports.

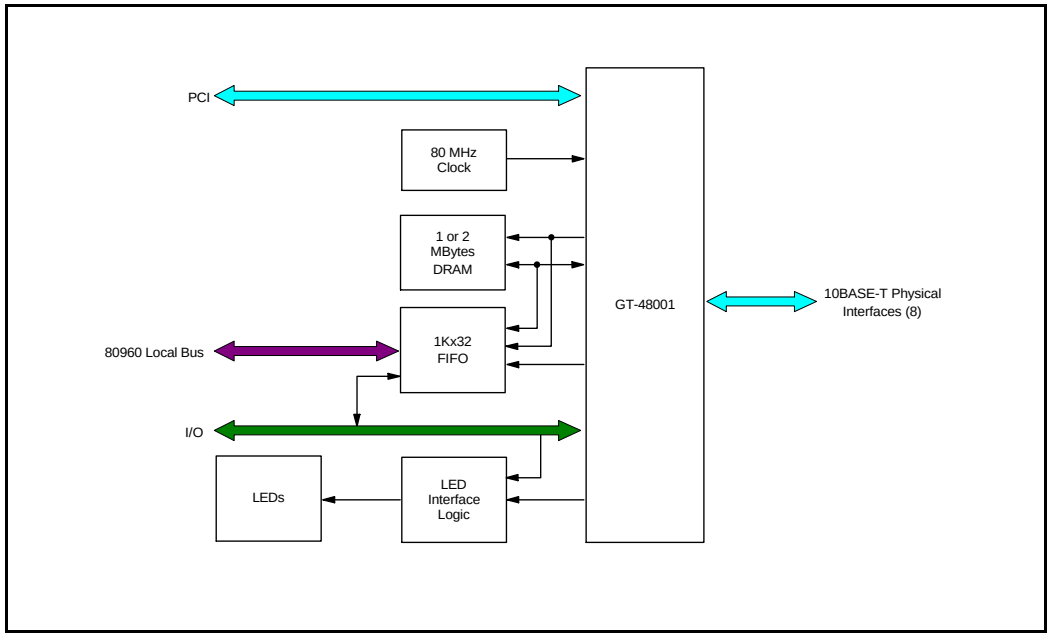


Figure 5. "8 Port Switch" Block Diagram

3.3 PCI Bus and Arbiter

A PCI bus interface is used for exchanging data between the SEC chips and the i960 processor. Connecting devices that utilize a PCI interface is relatively simple. The majority of the signals are connected as a bus with some signals run point-to-point. The signals that must be run point-to-point are the request, grant, and clock signals. The PCI clock distribution is discussed in Section 3.4, Clock Synthesis and Distribution. Pull-up resistors are also required for a number of signals.

Figure 6 shows a block diagram of the devices on the PCI bus and how they are connected.

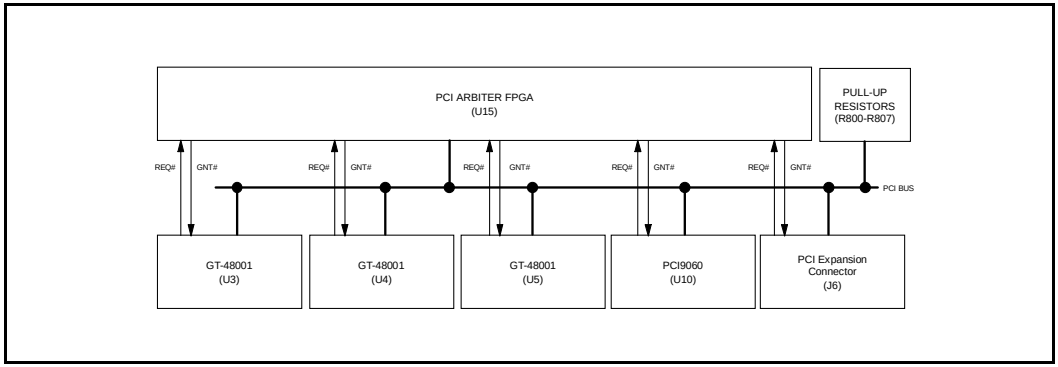


Figure 6. PCI Bus Interconnect

Pull-up resistors are required on the following PCI bus signals:

FRAME#	STOP#
TRDY#	SERR#
IRDY#	PERR#
DEVSEL#	LOCK#

The i960 processor local bus is interfaced to the PCI bus through a bridge chip, the PCI9060 from PLX technology. The PCI bridge chip allows devices on the PCI bus to access the processor bus and for the processor to access the PCI bus. The two buses operate independently except when accesses are made between them.

PCI Arbiter

An arbiter is required to determine which PCI device will have access to the bus. An FPGA device, (a Quicklogic QL8X12B) implements the arbitration function. When a PCI device needs to perform accesses on the bus, the device asserts its REQ# signal. Before performing the access, the device must wait until it is granted the bus by the arbiter and the bus becomes idle. The arbiter grants the bus to a device by asserting the device's GNT# signal. PCI bus arbitration is "hidden," which means that the arbitration can take place while a bus transaction is in progress. This is why a device must wait for the bus to become idle before performing its transaction. A device's grant can be asserted, then deasserted, if another device with a higher priority requests the bus. Thus, a bus grant is only valid when the bus has become idle.

The bus arbiter for this reference design implements the following priority scheme:

- The PCI9060 (i.e., processor) always has the highest priority. Its request will always be granted ahead of any other device requests.
- The other four devices have equal priority, with the bus granted in a round-robin fashion.

In addition to bus arbitration, the FPGA performs the "bus parking" function for the PCI bus. Bus parking means that the PCI bus signals that are normally not driven when the bus is idle must be driven by the bus parker to prevent a potential high current condition (due to the CMOS inputs floating to a level that turns on both the N and P channel transistors of the input buffer). The following signals must be driven by the bus parker:

AD[31:0]
C/BE[3:0]#
PAR

NOTE: PAR should be parked one clock later than AD[31:0] and C/BE[3:0]#.

The bus should be parked during reset and when the bus has become idle (FRAME# and IRDY# both deasserted). The *PCI Local Bus Specification* Revision 2.1 (page 9) defines the bus parking requirements during reset as follows:

> Anytime RST# is asserted, all PCI output signals must be driven to their benign state. In general, this means they must be asynchronously tri-stated ... REQ# and GNT# must both be tri-stated (they cannot be driven low or high during reset). To prevent AD, C/BE#, and PAR signals from floating during reset, the central resource may drive these lines during reset (bus parking) but only to a logic low level—they may not be driven high.

The design of the bus arbiter parks the bus during reset (drives the AD, C/BE#, and PAR signals low). However, the bus arbiter does not float the GNT# signals because this would result in a floating input to each PCI device unless pull-up resistors were used on the GNT# signals. The REQ# signals are pulled up with resistors. This is especially important for the expansion connector; if this slot is empty the REQ# line would always float.

The PCI arbiter design can be implemented, when the PCI bus is not reset, to park the bus in one of two ways:

- By performing the bus parking function itself.
- By letting the bus master that was last granted the bus perform the bus parking function. This is done by maintaining GNT# asserted to the last bus master. The PCI specification requires that a device whose GNT# is asserted when the bus is idle assume the bus parking function.

The option implemented in the reference design is for the arbiter to perform the bus parking function. The primary reason for choosing this option is that it is easier to implement.

Appendix C contains the schematics for the PCI Bus Arbiter FPGA. Figure 7 below shows the state diagram for performing bus parking. The bus is parked during State 1. State 2 serves as an intermediate state when transitioning to and from State 1 to allow the device currently driving the bus to turn off its output drivers. Bus grants will only be active during State 3. Note that the state machine remains in State 3 until the bus becomes idle and no REQ# or GNT# line is asserted. This is necessary to ensure that the bus parker and a device do not simultaneously drive the bus. For example, a device could remove its REQ# after its GNT# is asserted (during its bus transaction); however, the bus is not idle.

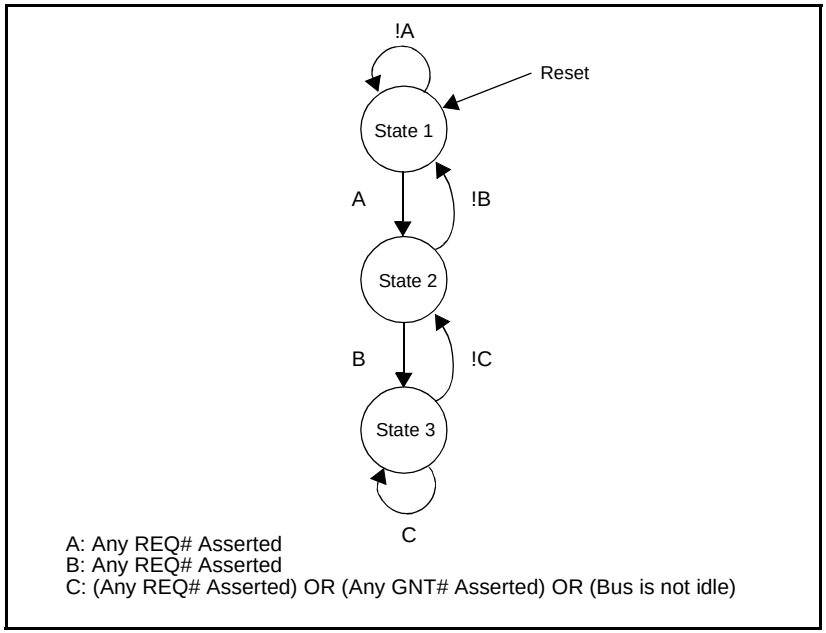


Figure 7. PCI Arbiter State Diagram 1

Sheet 1 of the PCI Bus Arbiter Schematics shows the state machine that implements the above state diagram. The actual implementation is somewhat different than the state diagram. The difference is as follows:

A: (Any REQ# Asserted) OR (Any GNT# Asserted) OR (Bus is not idle)

B: (Any REQ# Asserted) OR (Any GNT# Asserted) OR (Bus is not idle)

C: (Any REQ# Asserted) OR (Any GNT# Asserted) OR (Bus is not idle)

By making all of the cases identical, the design is simplified. Note that in State 1 or State 2, a GNT# should never be asserted and the bus should always be idle, so including them in the state equations does not change the functionality of the state machine.

Sheet 5 of the PCI Bus Arbiter Schematics shows the bus parking drivers. Note that the drivers, as implemented, are actually open drain outputs turned on during State 1, with the PAR driver delayed one clock cycle.

The remainder of the logic performs the arbitration for asserting GNT# for the device with the highest pending priority REQ#. For the processor GNT# (PCI_PLX_GNT#), the logic is relatively simple. The processor's request is always granted (it is the highest priority device). The logic is shown on Sheet 4 of the PCI Bus Arbiter Schematic. However, the GNT# must be asserted in an orderly manner; i.e., the GNT# cannot be asserted immediately if the bus is parked. This is accomplished by OR'ing State 1 into the GNT# logic. In addition, the bus arbiter cannot assert a GNT# to one device and deassert a GNT# to another device in the same clock cycle. Therefore, the GNTS_ACTIVE signal is AND'd with the GNT0# signal to prevent GNT0# from becoming active in the next clock cycle if GNT# is asserted to a different device in the current clock cycle. The requirement that all GNT#s be deasserted for one clock cycle is the result of a process called "stepping." The reader is referred to the PCI specification or a book on PCI for information on stepping. The transition state is required only when the bus is idle, but for simplicity, it is always performed by the PCI arbiter. The last signal that is part of the GNT0# logic is PRI_REQ, which is treated the same as REQ0#. PRI_REQ provides a

method for preventing "deadlock," which is discussed later. PRI_REQ is derived from the PLX_PRI_REQ# input, which is run through a three stage synchronizer before being used. The signal must be synchronized internally because it is a programmable output from the PCI9060 bridge chip and thus its relationship to the PCI clock is unknown (it is referenced to the processor clock).

Sheet 4 (top center) of the PCI Bus Arbiter Schematic shows the logic that indicates which of the devices with equal priority (also known as Round Robin devices) was last a bus master. The RR_Device_Took_Bus signal (see Sheet 1 of the Schematic) is active when the bus transitions from idle to not idle and a Round Robin device GNT# was active in the previous clock cycle. This indicates that the device whose GNT# was active in the previous clock cycle is now the bus master. The Round Robin device that was most recently bus master is indicated by the PM[4:1] signals, one of which will be active. This information is used to determine a device's request priority in the logic for the Round Robin devices (see Sheets 2 and 3 of the Schematic).

The priority (lower value = higher priority) is shown in the following table:

Table 1. Device Request Priority

PM[4:1]	REQ0# Priority	REQ1# Priority	REQ2# Priority	REQ3# Priority	REQ4# Priority
0001	0	4	3	2	1
0010	0	1	4	3	2
0100	0	2	1	4	3
1000	0	3	2	1	4

The combinatorial logic before each GNT# flip-flop allows the GNT# to be asserted for the device with the highest priority request pending (lowest value in table). The Round Robin GNT# logic is basically the same as the GNT0# logic but with the priority resolution included.

Sheet 1 of the Reference Design Schematic in Appendix A shows the PCI Arbiter FPGA.

Deadlock

The PLX_PRI_REQ# input and PLX_PRI_GNT# output are included to allow a potential solution to the "deadlock" problem as described in PLX Technology's *PCI Bus Interface and Clock Distribution Chips Product Catalog/1995* (page 19). Deadlock can occur when the processor attempts to access the PCI bus and, at approximately the same time, a device on the PCI bus attempts to access the processor bus. Neither access can be completed until one of the devices terminates its access. Eventually, the PCI device will timeout and then attempt to re-arbitrate for the bus, allowing the PCI9060 (processor) to get the PCI bus before the bus is granted to the device. However, deadlock can be either "partial" (the processor is accessing a different device from the one that is trying to access the processor bus) or "full" (the processor is accessing the same device that is accessing the processor bus). The PCI9060 datasheet implies that in this case, even if the device times out and relinquishes the PCI bus, the same device will not respond to the processor's request, creating "full deadlock."

The *PCI Local Bus Specification* Revision 2.1 does not describe this condition, but assuming that it does occur, one of the remedies suggested by PLX Technology should be performed. One method is to force the processor to "backoff" (i.e., terminate its access) and allow the PCI bus access to take place. This approach works for the 80960Cx and 80960Hx families of processors, which have a BOFF# pin that forces a "Bus Backoff." The BOFF# approach is implemented in the MISC_LOGIC FPGA. However, the 80960Jx family does not support Bus Backoff; therefore another approach is required. This approach makes use of PLX_PRI_REQ# (an output signal from the PCI9060) and PLX_PRI_GNT# (an input to the PCI9060). In this approach, described in the PCI9060 datasheet, the processor asserts PLX_PRI_REQ#, then waits until PLX_PRI_GNT# is asserted before performing the PCI access. When the access is completed, PLX_PRI_REQ# is deasserted. The logic in the PCI Arbiter FPGA treats PLX_PRI_REQ# as if it were a request by the PCI9060 (highest priority). PLX_PRI_GNT# is asserted when

GNT0# is asserted and the bus is idle (indicating the PCI bus is free).

PCI Arbiter Timing

The critical timing parameters (from the PCI Specification) for the PCI Arbiter FPGA are as follows:

Symbol	Min	Max	Units
CLK to Signal Valid Delay - point-to-point	2	12	ns
Input Setup Time to CLK - point-to-point (REQ#)	12		ns

The first parameter indicates the time from the rising edge of the CLK signal until the GNT# signals are valid. The second parameter indicates the guaranteed setup time for the REQ# signals relative to the rising edge of CLK. The PCI Arbiter FPGA is implemented in a QuickLogic QL8x12B-0PL68C FPGA (Field Programmable Gate Array). The "-0" speed grade is the second slowest of the speed grades available (-X, -0, -1, -2). The -0 part can be used by doing the following:

1. Duplicating the Flip-Flops for the GNT# outputs as shown on Sheets 2, 3, and 4 of the PCI Arbiter Schematics. One Flip-Flop drives only the output buffer while the other Flip-Flop drives the combinatorial logic that uses the GNT# signals internally. The Flip-Flop driving the output pad is also placed in a logic module close to the signal's assigned pin using the "PLACE" attribute. Thus the Flip-Flop outputs driving the GNT# outputs have the lowest possible internal loading.

2. Registering the REQ# inputs before using them internally (i.e., the logic is "pipelined"). This is necessary because the internal delay of the REQ# inputs cannot satisfy the setup time to the Flip-Flops that generate the GNT# signals. The setup times cannot be satisfied using even the fastest Quicklogic device (-2 speed grade).

The "Path Analyzer" tool, which is part of the QuickLogic software, was used to analyze the delays. The following are the delay values for a -0 speed grade:

Symbol	Min	Max	Units
CLK to Signal Valid Delay - point-to-point	3	11.7	ns
Input Setup Time to CLK - point-to-point (REQ#)	6.3		ns

Expansion Connector

A PCI expansion connector is provided to allow other PCI-based devices to be installed when the reference design is being used as an evaluation board. For example, a Fast Ethernet, ATM, or FDDI PCI board could be installed in this slot and the processor could be used to transfer data between the SEC chips and the installed PCI card.

Sheet 33 of the Reference Design Schematic in Appendix A shows the expansion connector (J6).

This connector is for a 5 V, 32-bit PCI expansion card. The expansion slot requires four power supplies: +5 V, +3.3 V, +12 V and -12 V. The +5 V power is provided by the +5 V supply used to power the reference design (and is input to the board on P25). The other three PCI expansion connector voltages, if required, must be provided using P26. Sheet 33 of the Reference Design Schematic in Appendix A shows the pinout for P25 and P26. Note that no other devices in the reference design require +12 V or -12 V. The 80960Hx processor requires +3.3 V, which is supplied by a 5 V to +3.3 V converter located on the board. However, this converter cannot supply enough current for both the 80960Hx processor and the PCI expansion slot. Therefore, +3.3 V for the PCI expansion slot is provided from an external supply via P26.

Interrupts

The PCI specification defines four interrupt signals: INTA#, INTB#, INTC#, and INTD#. These interrupt signals are shared between devices using Open-Drain drivers. The SEC chip has one INT# pin, the expansion connector has INTA#,

INTB#, INTC#, and INTD# pins, and the PCI9060 has an INTA# pin. The PCI9060 also has other pins that can be used to generate an interrupt: LSERR# and LINTO#. Normally, PCI device interrupt pins are connected by connecting the INTA# signals together, the INTB# signals together, the INTC# signals together, and the INTD# signals together, then running the resulting four interrupt signals to the interrupt controller (in this case the 80960 CPU). Note that the SEC INT# pin is equivalent to INTA#. The disadvantage to this approach is that when an interrupt occurs on INTA#, the software must poll each device to determine the source of the interrupt. Therefore, in the reference design, the interrupts for the SEC chips and PCI9060 are run directly to the 80960 CPU. The expansion connector interrupt pins cannot be run directly to the 80960 CPU because there are only eight interrupt pins available (XINT[7:0]#). Therefore, the four interrupt pins from the expansion connector are tied together and a single connection is made to the 80960 CPU.

Refer to the section on “Interrupt Sources” for more information on the interrupt connections to the 80960 CPU.

Additional PCI Information

A detailed discussion of the PCI bus is beyond the scope of this document. For more information on the PCI bus, refer to the *PCI Local Bus Specification* Revision 2.1 or to another resource concerning the PCI bus such as *PCI System Architecture* (see page 1).

3.4 Clock Synthesis and Distribution

The reference design requires clock signals for the following purposes:

1. PCI Bus
2. Processor Bus
3. DUART (AM85C30) PCLK
4. DUART Baud Rate Generator

The clock for the PCI bus must be distributed with a maximum clock skew of 2 ns between any two components according to the PCI specification. To satisfy this requirement, a low-skew clock driver, the CDC340 from Texas Instruments, is used. The CDC340 provides eight low-skew outputs with a maximum output skew of 0.6 ns. In addition, the trace lengths of the PCI clock signals are restricted to be within ±0.1 inches of each other. The trace length to the expansion connector was shortened to account for the clock trace length on the expansion card (2.5 ± 0.1 inches). All outputs from the clock driver are series terminated with 22 Ω resistors at the source to reduce ringing of the signal lines.

Figure 8 shows a block diagram of the PCI clock distribution implementation (refer to Sheet 3 of the Reference Design Schematic in Appendix A).

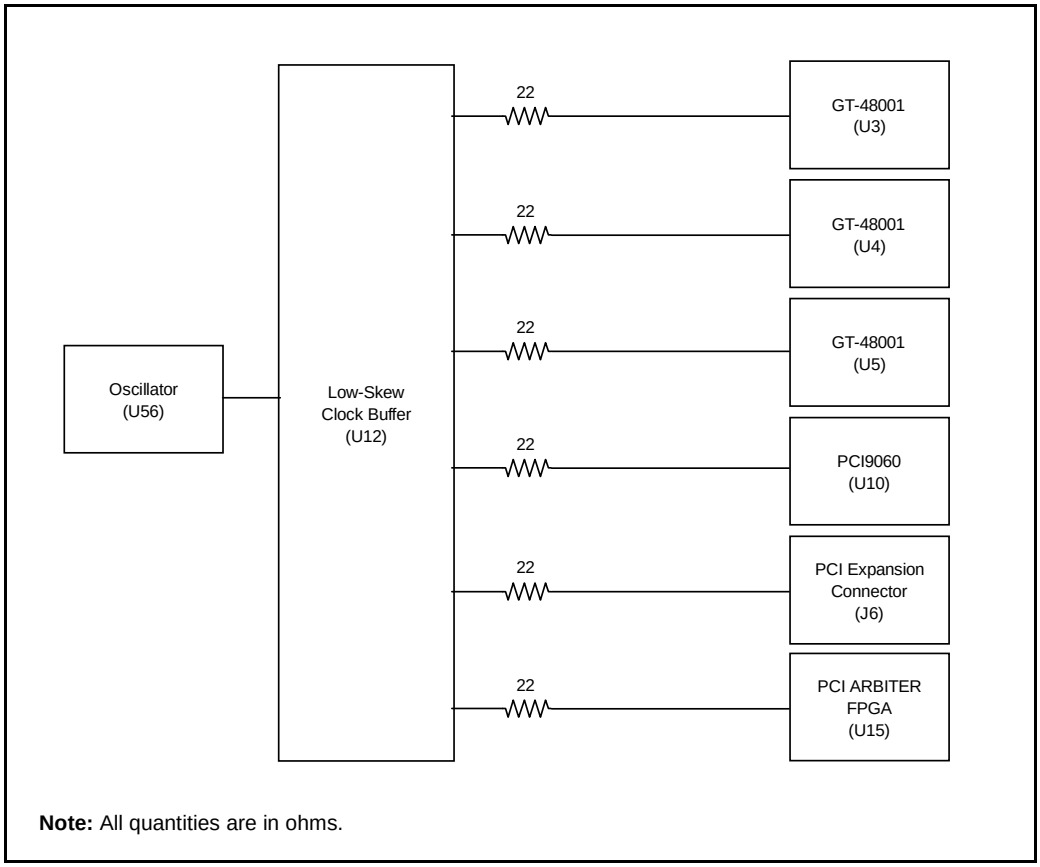
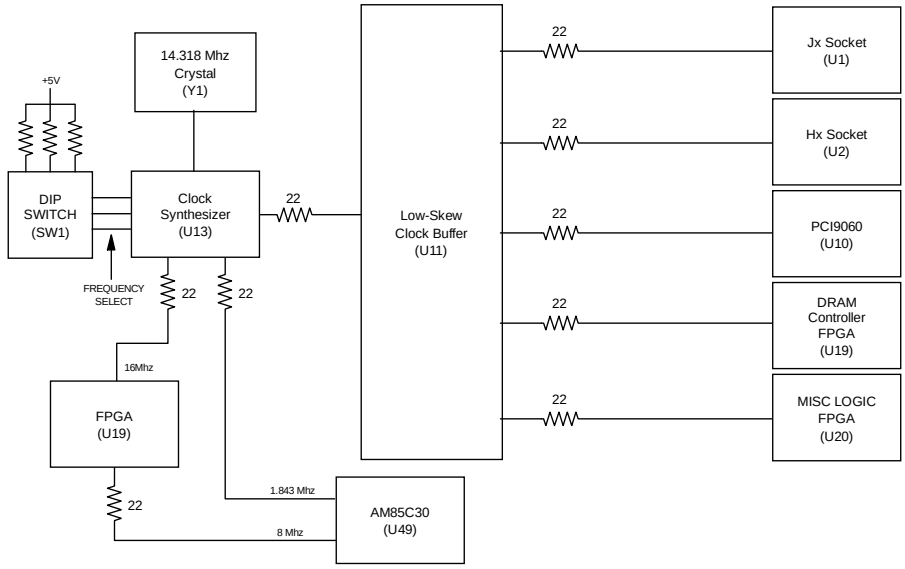


Figure 8. PCI Clock Distribution

Figure 9 shows the implementation for the remaining clocks. A clock synthesizer chip from Integrated Circuit Systems is used to derive the processor and DUART clocks. The clock synthesizer, AV9155A-23, can be programmed to generate eight different output frequencies that are derived from a 14.318 MHz crystal. The frequency that is generated is determined by a 3-bit code programmed using three poles of an eight-pole DIP switch. Table 2 shows the possible frequencies. The ability to program the processor frequency is useful when using the reference design as an evaluation platform. In addition to the programmed clock, the clock synthesizer provides 16 MHz and 1.843 MHz clock signals that are used by the DUART (AM85C30). The 1.843 MHz clock is used by an internal baud rate generator to generate the serial clocks (e.g., 19200 baud). The 16 MHz clock is divided by 2 in an FPGA to generate an 8 MHz clock for the DUART's PCLK input. The DUART interface is explained in more detail Section 3.12.6, Dual UART (DUART) Control Signals. Note that the 16 MHz and 1.843 MHz clock signals are not affected by the programmed frequency.



Note: All quantities are in ohms unless otherwise indicated.

Figure 9. Processor Clock Distribution

A low-skew clock driver is also used for distributing the processor clock to the processor, PCI bridge, and FPGA devices. This is done because the i960 bus is a synchronous bus referenced to the processor clock input. Therefore, to achieve optimum bus performance, devices interfaced to the bus (or their controllers) need to receive the clock at the same time the processor does. All of the clock signals are series terminated with 22 Ω resistors at the source to reduce ringing of the signal lines. Refer to Sheet 3 of the Reference Design Schematic in Appendix A for the processor clock distribution circuit (the DIP switch is located on Sheet 25).

Table 2. Processor Frequency Programming

Switch Settings			Code	Frequency (MHz)	Actual Frequency (MHz)
3	2	1			
On	On	On	0	37.5	37.585
On	On	Off	1	16	15.970
On	Off	On	2	30	30.068

Table 2. Processor Frequency Programming

On	Off	Off	3	20	20.045
Off	On	On	4	25	25.057
Off	On	Off	5	33.33	33.238
Off	Off	On	6	40	40.091
Off	Off	Off	7	26	25.952

3.5 LED Interface

Most 10BASE-T hubs use LED indicators to display the status of the connected LAN ports. The number of LED indicators varies, but a typical implementation provides two LEDs per port: one indicates port status and another indicates activity. The GT-48001 maintains LED-related information internally and outputs this data using a serial interface. The LED information is output continuously following device reset in a 128-bit frame using the following three signals:

LEDClk:	This signal is a 1 MHz clock with a nominal duty cycle of 50%.
LEDStb:	This signal is active high during the first bit (bit 1) of each LED data frame. LEDStb makes transitions on positive edges of LEDClk and is active for one clock cycle (1 µs) out of every 128 clock cycles (128 µs).
LEDData#:	This signal provides one bit of LED information every clock cycle. LEDData# makes transitions on positive edges of LEDClk. The first bit in an LED frame is numbered 1 with the number of each subsequent data bit number incremented until the number reaches 128, which is the last bit in the frame. Note that this signal is active low.

Figure 10 shows the signal relationships for the three signals at the start of an LED frame.

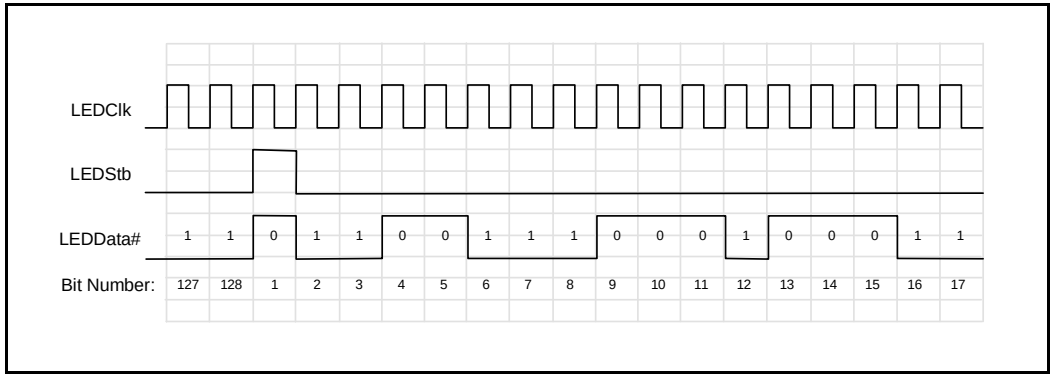


Figure 10. LED Interface Signal Relationships

There are seven LED status bits associated with each LAN port, as shown in Table 3. The port_status bit indicates the status conditions as shown in Table 4. The definition of each bit in the 128-bit LED frame is shown in Table 5. Some of the LED status signals may change at a relatively high rate (e.g., receive_data and transmit_data) and may not provide a visual indication on an LED when they occur if the pulse widths are short. However, the GT-48001 internally stretches pulses associated with the receive_data, transmit_data, and collision status bits such that they are approximately 0.5 seconds wide at the LED interface. Therefore, it is necessary only for external circuitry to capture an LED data frame and use the captured data bits to drive the desired LEDs. Figure 11 shows a block diagram of the basic concept for displaying LED information.

The external circuit design is based mainly on two factors: the LED information that will be displayed and the complexity of the control logic. It is assumed that the LED interface is implemented in some form of Field Programmable Gate Array (FPGA). The circuit design will also be a function of the FPGA family that is used to implement the circuit (i.e., if the FPGA architecture is rich in registers or combinatorial logic).

Table 3. LAN Port LED Status Bits

Name	Description
transmit_data	Indicates if data is currently being transmitted on the port.
receive_data	Indicates if data is currently being received on the port.
collision	Indicates if a collision has occurred on the port.
unknown_enable	Indicates if the forwarding of unknown packets is enabled.
port_sniffer	Indicates if the port has been configured as the sniffer port.
full_duplex	Indicates if the port is configured for full-duplex operation.
port_status	Indicates the status of the port.

Table 4. Port Status LED Modes

LED Condition	Status Indicated	Notes
On (Constantly)	OK	
Off (Constantly)	Port is disabled.	
Blinks Once	Link Integrity test failed.	The LED is active for a 420 ms period every 5 seconds.
Blinks Twice	Partition	The LED is active for two 420 ms periods every 5 seconds. The two 420 ms periods are separated by 1.26 seconds.

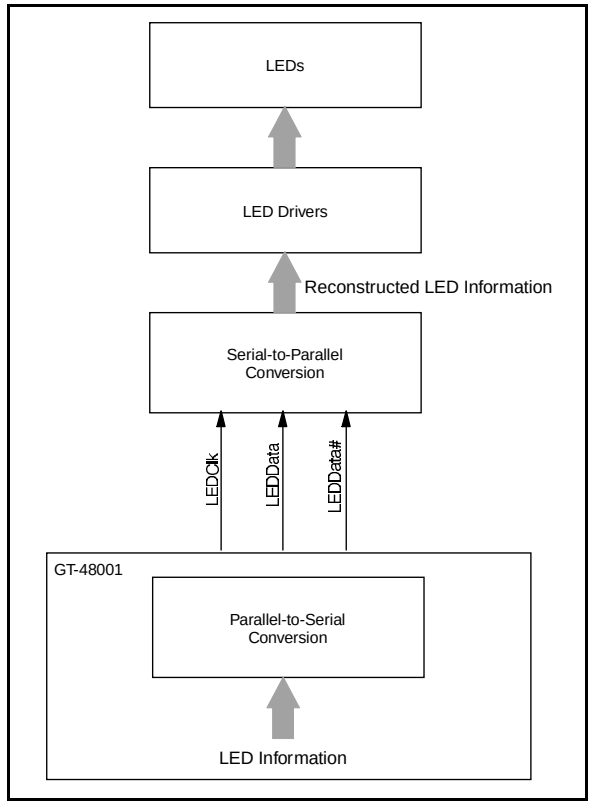


Figure 11. Generic LED Interface Block Diagram

Table 5. LED Data Frame Bit Definition

Bit Number	Usage
1	[0] port_status
2	[1] port_status
3	[2] port_status
4	[3] port_status
5	[4] port_status
6	[5] port_status
7	[6] port_status
8	[7] port_status
9	[0] transmit_data
10	[0] receive_data
11	[0] collision
12	Private
13	[0] unknown_enable
14	[0] port_sniffer
15	[0] full_duplex
16	Private
17	[1] transmit_data
18	[1] receive_data
19	[1] collision
20	Private
21	[1] unknown_enable
22	[1] port_sniffer
23	[1] full_duplex
24	Private
25	[2] transmit_data
26	[2] receive_data
27	[2] collision
28	Private
29	[2] unknown_enable
30	[2] port_sniffer
31	[2] full_duplex

Table 5. LED Data Frame Bit Definition

Bit Number	Usage
32	Private
33	[3] transmit_data
34	[3] receive_data
35	[3] collision
36	Private
37	[3] unknown_enable
38	[3] port_sniffer
39	[3] full_duplex
40	Private
41	[4] transmit_data
42	[4] receive_data
43	[4] collision
44	Private
45	[4] unknown_enable
46	[4] port_sniffer
47	[4] full_duplex
48	Private
49	[5] transmit_data
50	[5] receive_data
51	[5] collision
52	Private
53	[5] unknown_enable
54	[5] port_sniffer
55	[5] full_duplex
56	Private
57	[6] transmit_data
58	[6] receive_data
59	[6] collision
60	Private
61	[6] unknown_enable
62	[6] port_sniffer

Table 5. LED Data Frame Bit Definition

Bit Number	Usage
63	[6] full_duplex
64	Private
65	[7] transmit_data
66	[7] receive_data
67	[7] collision
68	Private
69	[7] unknown_enable
70	[7] port_sniffer
71	[7] full_duplex
72	Private
73-128	Private

NOTES:

1. [] contains the port number for the associated status bit. For example, "[6] receive_data" indicates the associated bit is the receive_data status bit for port 6.
2. "Private" bit locations are used for testing or reserved for future use and should not be used.

Figure 12 shows a block diagram for a conceptually simple circuit for capturing the LED status information. In this circuit, the serial data (LEDData#) is shifted into a 128-bit shift register on the negative edge of the LEDClk. Each time that LEDStb goes high, the data currently in the shift register is captured in a 128-bit register. The register output can then be used to drive the LED circuits. Although this circuit is conceptually simple, it requires a significant number of Flip-Flops (256). The number of Flip-Flops that are required can be reduced because the output register only needs Flip-Flops for status bits used to control LEDs. For example, 16 Flip-Flops would be required to display a port's status and activity (an OR gate would be used to combine the receive_data and transmit_data bits before the output register). However, this circuit always requires that the input shift register be 128 bits long.

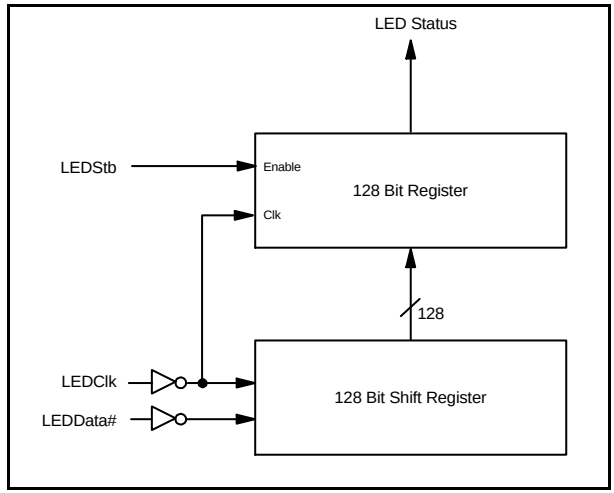


Figure 12. LED Interface Example 1 Block Diagram

Figure 13 shows the block diagram of another circuit for capturing the LED status information. This circuit uses a 7-bit synchronous counter to indicate which bit in the LED frame is currently being received. The counter is synchronously reset by LEDStb. The counter outputs are decoded to enable a specific Flip-Flop that captures the desired LED status bit. If one LED is used to indicate activity, an OR gate would be used to combine the outputs of two Flip-Flops (corresponding to the receive_data and transmit_data status bits). This circuit requires significantly fewer Flip-Flops but more combinatorial logic.

Figure 14 shows the block diagram of another circuit that can be used for capturing the LED status information. This circuit is a combination of the two LED Interface circuits already discussed. In this circuit, the LEDStb signal is shifted through an n-bit shift register. The outputs from this shift register enable Flip-Flops that sample the current value of LEDData. At most, one shift register output is high at any time. The maximum shift register length is 71 bits because there are no LED status bits located after bit 72 (LEDStb serves as the enable for the Flip-Flop that captures the [0] port_status value).

The circuit implemented for the reference design is a combination of the circuits shown in Figures 13 and 14. The port_status bits are captured with the circuit in Figure 14 and the other LED status bits are captured with the circuit in Figure 13. The reference design circuit was implemented in a QuickLogic QL8x12B FPGA in a 68-pin PLCC package (one FPGA for each GT-48001). The lowest available speed grade is used, -X. Four LEDs are available for each LAN port. A dual right angle LED, visible on the front panel, is used to display status (lower LED) and activity (upper LED).

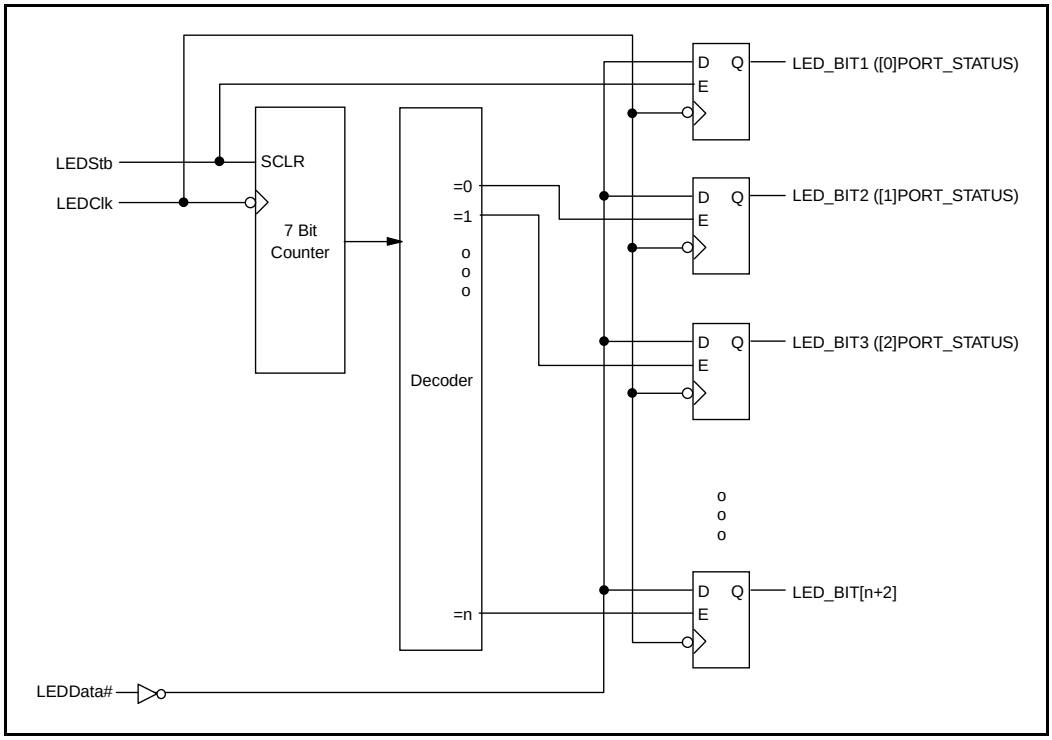


Figure 13. LED Interface Example 2 Block Diagram

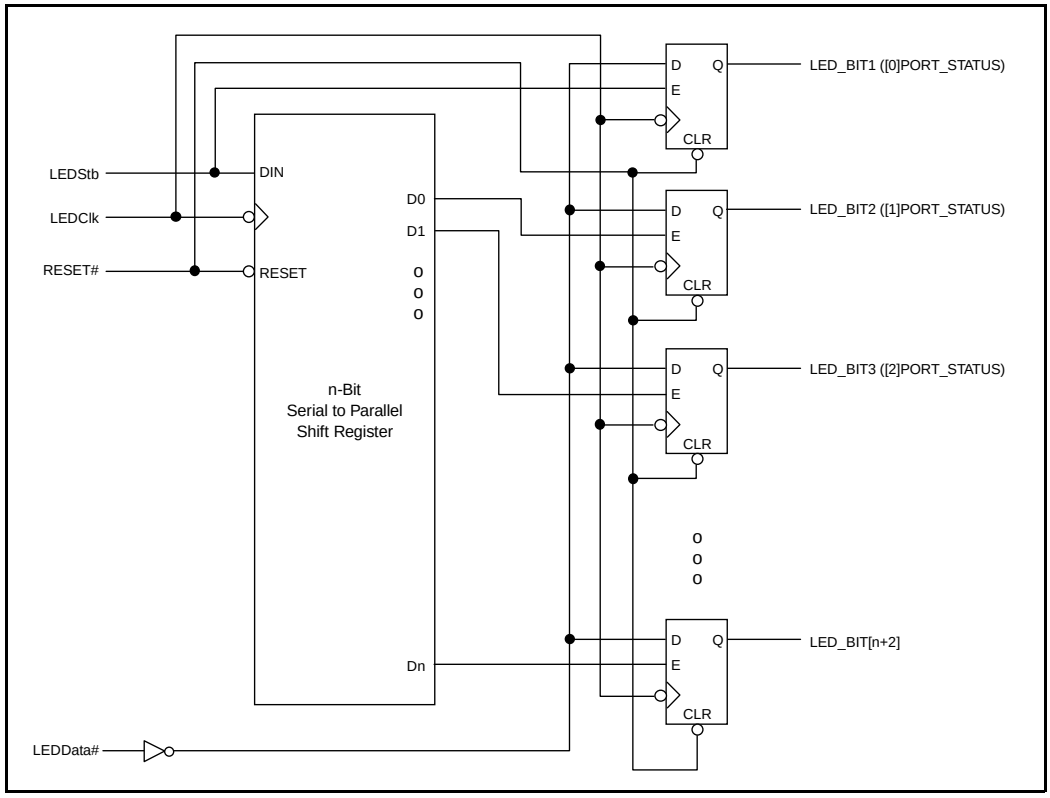


Figure 14. LED Interface Example 3 Block Diagram

Two single top view LEDs, located behind the dual right angle LED, are used to display the receive_data and collision status. The top view LEDs are used only for debugging and are not visible when the reference design is enclosed in a case. The "activity" LED would normally be active when there is data either being received or transmitted (i.e., logical OR of the receive_data and transmit_data status bits). To facilitate debugging, two inputs (ACT_LED_SEL[1:0]) select what status is displayed on the "activity" LED, as shown in Table 6.

Table 6. "Activity" LED Status Displayed

ACT_LED_SEL[1:0]	Status Displayed
00	receive_data OR transmit_data
01	transmit_data
10	receive_data
11	collision

For test/debug purposes, four signals are brought to the FPGA that allow the processor to simulate the LED interface. These signals, from an output port controlled by the processor, are described in Table 7.

Table 7. LED Interface Test Signals

Signal	Description
TEST_ENABLE	This signal selects which set of Clock, Data, and Strobe signals will drive the internal logic of the LED Interface FPGA. A logic 0 selects the GT-48001 signals and a logic 1 selects the test signals (TEST_CLK, TEST_STB, and TEST_DATA#).
TEST_CLK	This signal becomes the LED Interface clock signal to the internal logic of the LED Interface FPGA when TEST_ENABLE is asserted.
TEST_STB	This signal becomes the LED Interface strobe signal to the internal logic of the LED Interface FPGA when TEST_ENABLE is asserted.
TEST_DATA#	This signal becomes the LED Interface data signal to the internal logic of the LED Interface FPGA when TEST_ENABLE is asserted.

Figure 15 shows a block diagram of the test signal multiplexing. This figure also shows the method that was used for routing the clock and reset signals onto low-skew clock networks in the QuickLogic FPGA. The QL8x12B has two low-skew clock networks that can be used to route clock or reset signals to logic modules in the device. The QuickLogic databook indicates that to connect a signal to these low-skew networks, it is necessary to bring the desired signal onto the FPGA device using the CLK input pins. However, the reset signal needs to be inverted and the clock signal needs to be multiplexed, then inverted. Therefore, the multiplexed/inverted clock signal and the inversion of the reset signal are routed off the chip so they could be routed to the FPGA CLK input pins via the Printed Circuit Board (PCB). This implementation is shown in Figure 15. The CLK, CLR, STB, and DATA signals are used by the internal logic that captures a port's LED status information. The CLR signal, when asserted, forces all LED outputs off.

Appendix B contains the schematics for the LED Interface FPGA. Sheet 1 contains the multiplexing logic. Sheet 2 contains an 8-bit shift register whose outputs provide the enable signals (X[7:0]) for the Flip-Flops that capture the port_status LED information. Sheets 3 to 6 contain the logic for capturing the receive_data, transmit_data, and collision data for each port. These sheets also contain the circuitry that selects the output for the "activity" LED. Sheet 7 contains a 7-bit counter that indicates which bit is being received. This counter is synchronously reset to 1 by the STB signal. This counter will halt when it reaches 127 and will remain at a value of 127 until the STB signal is high. Sheet 8 contains the logic that decodes the counter value to determine which LED status bit is being received (status type and associated port). The decoded signals are used on Sheets 3-6. Note that the outputs are active low (inverted by the output buffer) to allow the FPGA to drive an LED directly using the circuit shown in Figure 16. This circuit does not require the FPGA to source current to turn the LED on. In general, it is preferable for a CMOS device to sink current because less voltage drop occurs in the output driver when sinking current. The LEDs used are low current devices that require only 2 mA. Low current LEDs were used to lower the 5 V current required. Note that there are 96 LEDs displaying LAN port status, which would require 960 mA if all were on and 10 mA LEDs are used. The current drops to a maximum of 288 mA using the low current LEDs (the current may be as high as 3 mA with the 1.0 KΩ current limiting resistor).

Refer to Sheets 7, 8, and 9 of the Reference Design Schematic in Appendix A for the LED Interface circuitry implemented on the PCB.

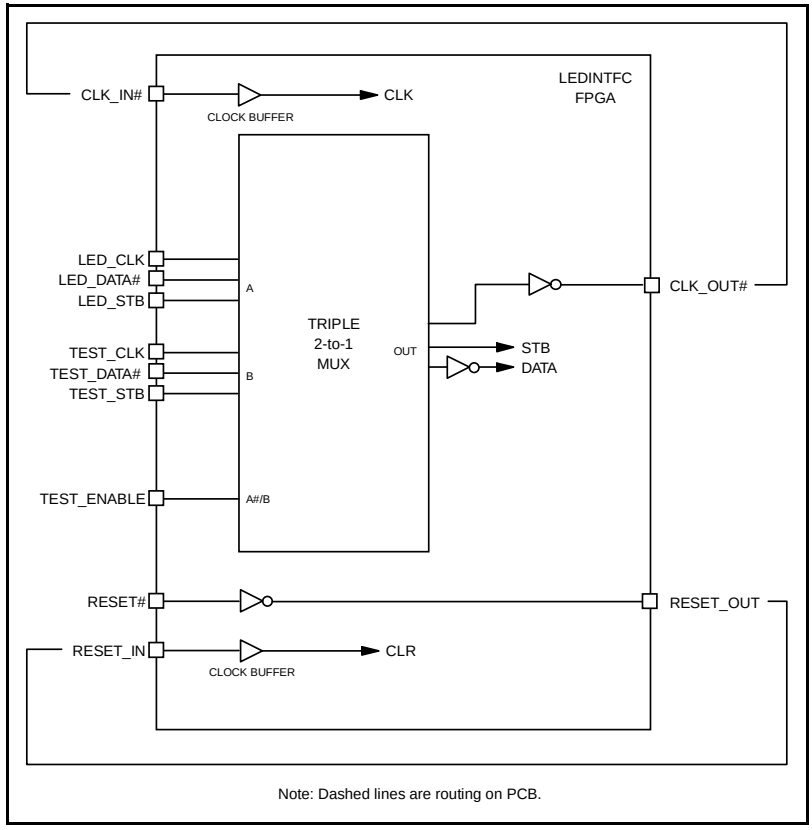


Figure 15. LED Interface Clock/Reset Routing

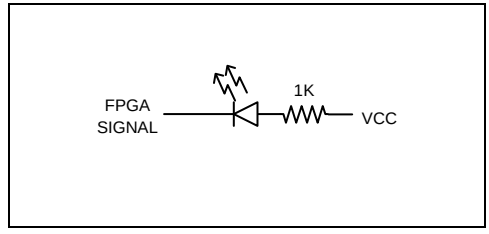


Figure 16. LED Circuit

3.6 Interrupt Sources

Table 8 lists the 80960 interrupt inputs and the source that drives each interrupt.

Table 8. Interrupt Sources

Interrupt Input	Source
NMI#	PCI9060 LSERR#
XINT7#	PCI9060 LSERR#
XINT6#	Bus Error
XINT5#	85C30 Dual Serial Controller
XINT4#	PCI9060 INTA#
XINT3#	SEC #3
XINT2#	SEC #2
XINT1#	SEC #1
XINT0#	PCI Expansion Connector
Note: A jumper selects whether the PLX9060 LSERR# signal goes to NMI#, XINT7#, or neither.	

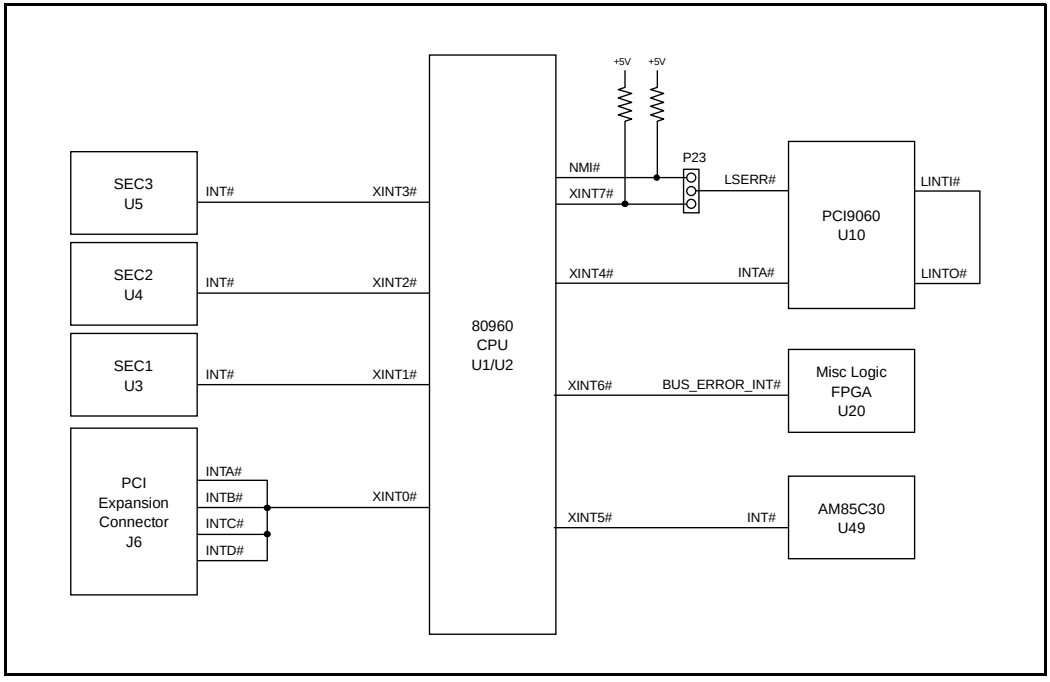


Figure 17. Interrupt Sources

Figure 17 shows the interrupt sources graphically. The connection of LINTO# to LINTI# on the PCI9060 allows INTA# to be asserted if LINTO# is asserted. Thus, any interrupt source for LINTO# becomes an interrupt source for INTA#. Note that the PCI9060 LSERR# output can be connected to either XINT7# or NMI# of the CPU via jumper P23.

3.7 Memory Map

Table 9 shows the memory map for the reference design. Note that complete decoding of the 32-bit address bus is not performed and therefore "aliasing" of addresses occurs. For example, only six address bits, ADDR[31:26], are used in the chip select logic for the boot EPROM, a 128 Kx8 device. Thus, ADDR[25:17], are "don't cares" in the EPROM chip select decoding.

Table 9. Memory Map

Address Range	Bus Width	Device/Function
0000 0000 - 0000 03FF	N/A	Internal i960 Static RAM
0000 0400 - 1FFF FFFF	32	Aliased DRAM
2000 0000 - 3FFF FFFF	32	DRAM
4000 0000 - 4000 00FF	8	85C30-8 Dual Serial Controller
4000 0100 - 4000 01FF	8	Misc Logic FPGA
4000 0200 - 4000 02FF	8	DRAM Controller FPGA
4000 0300 - 5FFF FFFF		Unused
6000 0000 - 6000 00FF	32	Input Register
6000 0100 - 6000 01FF	32	Output Register
6000 0200 - 6000 07FF		Unused
6000 0800 - 6000 09FF	32	SEC #1 RMON FIFO
6000 0A00 - 6000 0BFF	32	SEC #2 RMON FIFO
6000 0C00 - 6000 0DFF	32	SEC #3 RMON FIFO
6000 0E00 - 7FFF FFFF		Unused
8000 0000 - 9FFF FFFF	32	i960 Local Bus to PCI Bus
A000 0000 - BFFF FFFF	32	PLX PCI9060 Internal Registers
C000 0000 - D7FF FFFF		Unused
D800 0000 - DBFF FFFF	32	Flash Bank 2
DC00 0000 - DFFF FFFF	32	Flash Bank 1
E000 0000 - FBFF FFFF		Unused
FC00 0000 - FFFF FFFF	8	Boot EPROM (27C010)

Memory areas that have their device/function specified as "Unused" do not have hardware logic implemented that generates a READY# acknowledgment. Thus, an access to these memory areas causes the processor to halt until the Bus Monitor circuit times out and generates a READY# acknowledgment. Refer to Section 3.12.10, Bus Monitor, for more information.

ROM Swapping

The chip select decoding logic allows the memory areas used by the Flash memory and the Boot EPROM to be swapped, resulting in the following memory map for the C000 0000 to FFFF FFFF memory region.

Table 10. Memory Map When ROM Swapping Enabled

Address Range	Bus Width	Device/Function
C000 0000 - DBFF FFFF		Unused
DC00 0000 - DFFF FFFF	8	Boot EPROM (27C010)
E000 0000 - F7FF FFFF		Unused
F800 0000 - FBFF FFFF	32	Flash Bank 2
FC00 0000 - FFFF FFFF	32	Flash Bank 1

The “ROM Swapping” is controlled by Switch 4 of the DIP Switch (SW1), as shown on Sheet 25 of the Reference Design Schematic in Appendix A. When Switch 4 is open (SWAP_ROM# = 1), the decoding in Table 9 is performed. When Switch 4 is closed (SWAP_ROM# = 0), the decoding is modified as shown in Table 10. “ROM Swapping” is implemented to allow software development to take place using an EPROM until the software has progressed to the point where the FLASH memory can be used as the boot device (software can be downloaded into the Flash memory). When the FLASH memory is used as the boot device, the “ROM Swapping” can be left enabled and the EPROM no longer used.

Design Considerations

The assignment of addresses was driven by a consideration of how the PMCON (Physical Memory Configuration) registers would be programmed to access the memory regions for an 80960Jx processor. The PMCON registers—there are eight in a 80960Jx processor—define the bus width of a memory region. The bus width can be 8, 16, or 32 bits and each memory region is 512 Mbytes. Thus, it is important to group devices with similar bus widths in the same memory region. For example, I/O registers and RMON FIFOs are assigned addresses that place them in the same memory region because they are 32-bit wide devices. When a 80960Cx or 80960Hx processor is used, the memory configuration programming is less restrictive because the memory region granularity is 256 Mbytes (16 memory regions).

3.8 Processor Bus Configuration

Figure 18 shows a block of the various devices connected to the 80960 processor address and data buses. Note that the top three boxes represent the “processor.” Only one of the two processors is enabled. When the 80960Cx/Hx processor is enabled, the address latch (U38-U41) is disabled as well as the 80960Jx processor. The address latch captures the address from the multiplexed address/data bus of the 80960Jx processor, allowing the 80960Jx processor to look like a 80960Cx/Hx processor (separate address and data buses).

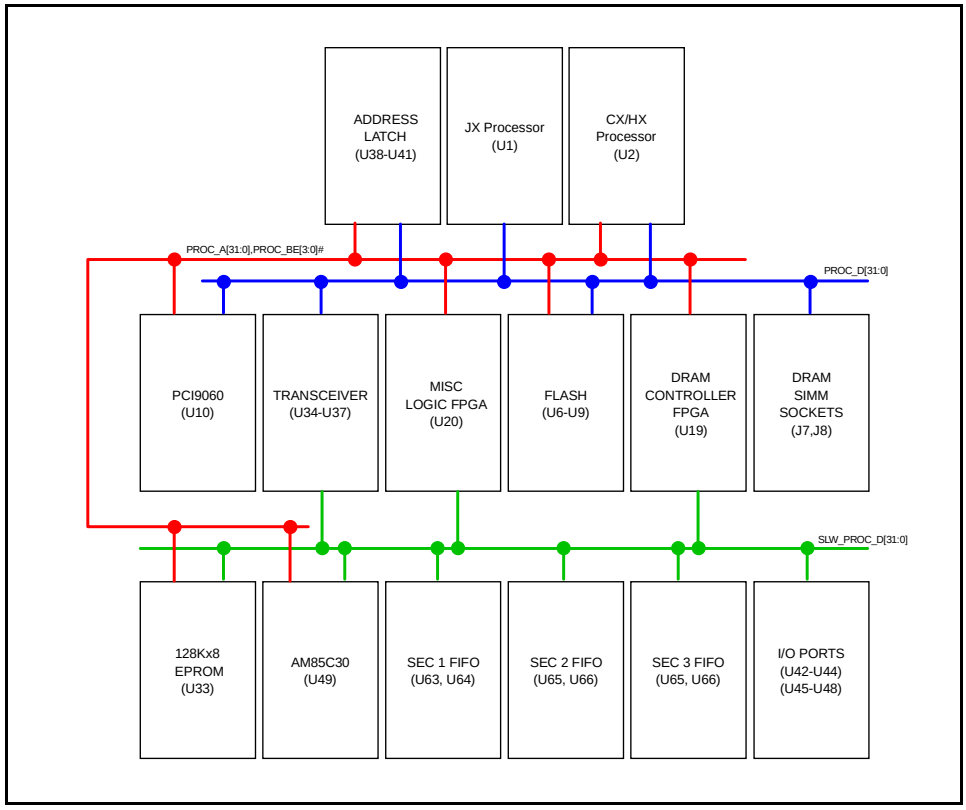


Figure 18. Processor Address and Data Bus Connections

Note from the figure that there is one address bus but two data buses. The address bus is not heavily loaded, as it connects to eight loads, but there are 20 devices (maximum) that have data bus connections (DRAM counts as 4, Flash counts as 2, and the I/O ports count as 2). As a result, the data bus is separated into two data buses (one fast and one slow). Devices that are accessed often such as DRAM and Flash are placed on the "fast" data bus (Proc_D[31:0]), whereas devices that are accessed infrequently (I/O ports) or that are inherently slow (AM85C30) are placed on the "slow" data bus (SLW_PROC_D[31:0]). A second consideration in the assigning of devices to the "slow" data bus is their physical location. The devices on the "fast" data bus can be physically close together, whereas the devices on the "slow" data bus tend to be widely dispersed on the printed circuit board. Therefore, a transceiver (U34-U37) is used to create a secondary data bus. The logic that controls the transceiver is located in the MISC LOGIC FPGA. Sheet 26 of the Reference Design Schematic shows the transceiver.

3.9 I/O

The reference design contains two I/O registers. One register is an input register (24 bits) and the other register is an output register (32 bits). The input register is constructed from three latches (74ACT573), where the latch enable (LE) signal is tied high (asserted) and the output enable (OE#) is asserted when the input register contents are being read. The input register OE# (IO_OE#) is generated by the Misc Logic FPGA. The output register is constructed from four octal registers (74ACT574), where the OE# is tied to POS_PROC_RESET and the clock input is from the Misc Logic FPGA. The output register is tri-stated when the processor is reset, to allow the signals driven by the output register to be set by pull-up or pull-down resistors. This is useful only if the processor is continuously reset using jumper P1 while the remainder of the circuitry is allowed to operate (unmanaged hub mode).

Note that the four octal registers share the same clock signal (IO_CLK) and therefore must be written as a 32-bit variable. If software attempts to write a single byte (or 16-bit word) of the output register, the contents of the other 24 bits (16 bits) will be unknown after the write. Also, the contents of the output register cannot be read back. Therefore, the software should maintain the value of the register in a variable for use when manipulating only a portion of the register. The input register could be treated as individual bytes by the software, if desired, because there are no adverse affects from reading only part of the register. Note that if the software reads only one byte, the entire input register is enabled onto the data bus and the processor's bus controller feeds the desired byte to the processor core.

Sheet 25 of the Reference Design Schematic shows the input and output register circuits. Refer to Section 3.12, Misc Logic FPGA, for information on generating IO_CLK and IO_OE. The individual bits for the input register are as follows:

Bit 0:	SEC1_FIFO_EF#	(Empty Flag, active low, for SEC 1 RMON FIFO)
Bit 1:	SEC1_FIFO_HF#	(Half Empty Flag, active low, for SEC 1 RMON FIFO)
Bit 2:	SEC1_FIFO_FF#	(Full Flag, active low, for SEC 1 RMON FIFO)
Bit 3:	0	(Unused)
Bit 4:	SEC2_FIFO_EF#	(Empty Flag, active low, for SEC 2 RMON FIFO)
Bit 5:	SEC2_FIFO_HF#	(Half Empty Flag, active low, for SEC 2 RMON FIFO)
Bit 6:	SEC2_FIFO_FF#	(Full Flag, active low, for SEC 2 RMON FIFO)
Bit 7:	0	(Unused)
Bit 8:	SEC3_FIFO_EF#	(Empty Flag, active low, for SEC 3 RMON FIFO)
Bit 9:	SEC3_FIFO_HF#	(Half Empty Flag, active low, for SEC 3 RMON FIFO)
Bit 10:	SEC3_FIFO_FF#	(Full Flag, active low, for SEC 3 RMON FIFO)
Bit 11:	0	(Unused)
Bit 12:	PCI9060 DMAPF#	(Direct Master FIFO Almost Full, active low)
Bit 13:	0	(Unused)
Bit 14:	0	(Unused)
Bit 15:	Data Out from the Serial EEPROM (U52)	

Bits 23:16 are the value of the DIP Switch (SW1).

Bits 18:16 select the processor clock frequency (see Section 3.4, Clock Synthesis and Distribution).

Bits 19 indicates the value of the SWAP_ROM# input to the Misc Logic FPGA.

0: Flash Memory is the boot device.

1: EPROM is the boot device.

Bits 23:20 are user-definable.

The individual bits for the output register are as follows:

Bit 0:	SEC1_RSTQUEUE#	(RSTQUEUE# signal for SEC 1)
Bit 1:	SEC1_ENDEV#	(ENDEV# signal for SEC 1)
Bit 2:	SEC1_FIFO_RESET#	(Reset, active low, for SEC 1 RMON FIFO)
Bit 3:		(Unused)

Bit 4:	SEC1_TSTLEDDATA#	(Test Data signal for SEC 1 LED Interface)
Bit 5:	SEC1_TSTLEDSTB	(Test Strobe signal for SEC 1 LED Interface)
Bit 6:	SEC1_TSTLEDCLK	(Test clock signal for SEC 1 LED Interface)
Bit 7:	SEC1_TEST_ENABLE	(Enables test signals to control SEC 1 LED Interface)
Bit 8:	SEC2_RSTQUEUE#	(RSTQUEUE# signal for SEC 2)
Bit 9:	SEC2_ENDEV#	(ENDEV# signal for SEC 2)
Bit 10:	SEC2_FIFO_RESET#	(Reset, active low, for SEC 2 RMON FIFO)
Bit 11:		(Unused)
Bit 12:	SEC2_TSTLEDDATA#	(Test Data signal for SEC 2 LED Interface)
Bit 13:	SEC2_TSTLEDSTB	(Test Strobe signal for SEC 2 LED Interface)
Bit 14:	SEC2_TSTLEDCLK	(Test clock signal for SEC 2 LED Interface)
Bit 15:	SEC2_TEST_ENABLE	(Enables test signals to control SEC 2 LED Interface)
Bit 16:	SEC3_RSTQUEUE#	(RSTQUEUE# signal for SEC 3)
Bit 17	SEC3_ENDEV#	(ENDEV# signal for SEC 3)
Bit 18:	SEC3_FIFO_RESET#	(Reset, active low, for SEC 3 RMON FIFO)
Bit 19:		(Unused)
Bit 20:	SEC3_TSTLEDDATA#	(Test Data signal for SEC 3 LED Interface)
Bit 21:	SEC3_TSTLEDSTB	(Test Strobe signal for SEC 3 LED Interface)
Bit 22:	SEC3_TSTLEDDATA#	(Test clock signal for SEC 3 LED Interface)
Bit 23:	SEC3_TEST_ENABLE	(Enables test signals to control SEC 3 LED Interface)
Bit 24:	Serial EEPROM Chip Select	
Bit 25:	Serial EEPROM Clock	
Bit 26:	Serial EEPROM Data Input	

Bits 31:27 control five user-defined LEDs that are intended for use as an aid in debugging.

NOTES:

1. To turn on an LED, the associated bit should be written with a 0.
2. Refer to Section 3.5, LED Interface, for a description on the use of the LED test signals.

3.10 Reset

Sheet 2 of the Reference Design Schematic shows the power up reset circuit for the reference design. A "µP Supervisory Circuit" chip, the MAX707 from Maxim Integrated Products, generates the power reset. This chip contains a power up reset circuit and a power fail indicator circuit. The power fail indicator circuit is not used. The MAX707 monitors the +5 V supply and asserts RST# when the voltage is below 4.65 V. The MAX707 uses a simple connection for a push-button switch to manually generate reset. The push-button switch (SW2) is a small surface mount switch that would not be accessible if the printed circuit board is installed in a case. It is intended for use as a debugging aid to allow resetting the circuit without removing power.

The RST# output from the MAX707 is then buffered by a quad XOR gate (74AC86). At this point, two reset signals are generated: PCI_RST# and PU_RESET#. The PCI_RST# is simply a buffered version of the MAX707 RST# and is routed to devices connected to the PCI Bus. The PU_RESET# signal is determined by jumper P1, which selects either the MAX707 RST# signal or GND. In the normal configuration, P1 would be set to select MAX707 RST# and the PU_RESET# and PCI_RST# signals would be the same. If P1 is set to select GND, PU_RESET# will be continuously asserted. This capability was provided to allow the processor circuitry to be disabled, converting the reference design into an unmanaged hub. This capability would be used only for debugging or evaluation purposes.

Note that the PCI9060 PCI reset (RST#) is connected to PU_RESET# instead of PCI_RST#. This effectively removes the PCI9060 from the PCI bus since the PCI9060 would serve no purpose if the processor is reset.

LEDs provide a visual indication when PU_RESET# (CR2 LED) and PCI_RST# (CR1 LED) are asserted. These LEDs are for diagnostic/debugging purposes and cannot be seen if the circuit board is in a case.

A watchdog timer circuit is located in the DRAM Controller FPGA. This circuit provides the reset for the processor and related circuits (DUART, Flash, etc.). The watchdog timer is reset by PU_RESET#, but then generates the PROC_RESET# and POS_PROC_RESET signals used elsewhere. The watchdog timer circuit is described in the Section 3.11, DRAM Controller FPGA.

3.11 DRAM Controller FPGA

3.11.1 Overview

The reference design contains two QuickLogic FPGA devices that provide logic for the processor section of the design. One FPGA was originally intended to contain only a DRAM Controller while the other FPGA would contain the remaining required logic (i.e., Misc Logic). However, it was necessary to incorporate part of the remaining logic in the DRAM Controller FPGA because of pin and logic constraints in the Misc Logic FPGA. As a result, the DRAM Controller FPGA contains the following logic:

- DRAM Controller
- Watchdog Timer
- Flash VPP Enable
- DUART Clock Divider

Appendix D contains the DRAM Controller FPGA Schematic.

Figure 19 shows the input and output signals from the DRAM Controller FPGA.

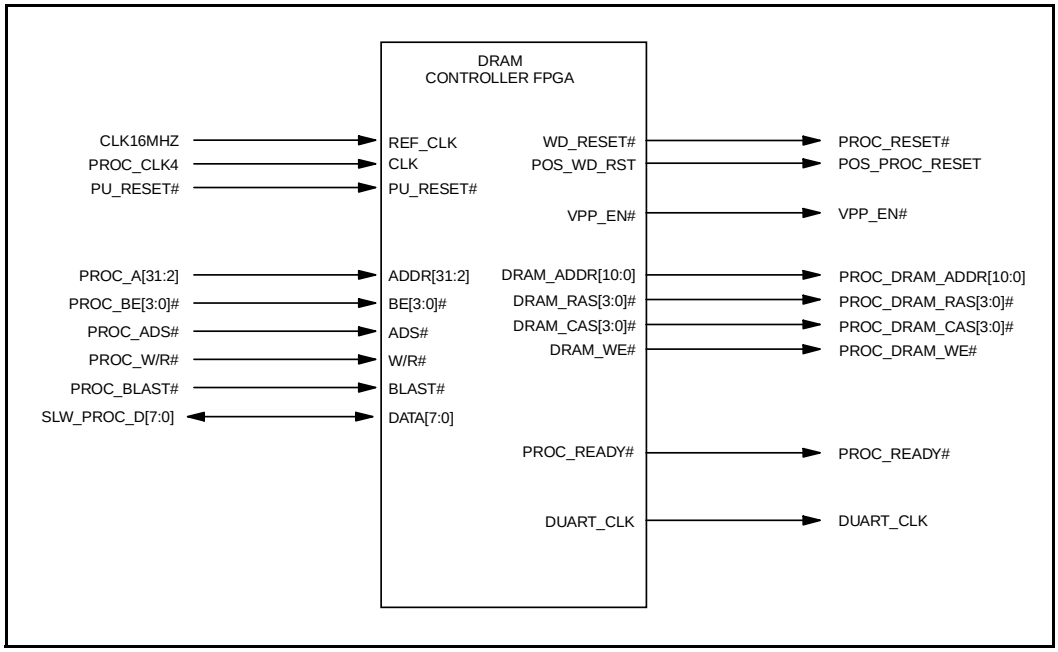


Figure 19. DRAM Controller FPGA Signals

Figure 20 shows a block diagram for the DRAM Controller FPGA. The Device Control block controls writing data to and reading data from the FPGA itself. This block also contains some timing logic. The DUART_CLK is an 8 MHz signal generated by dividing the REF_CLK (a 16 MHz signal) by 2. The RFSH_REQ signal is an input to the DRAM Controller that indicates that a refresh cycle should be performed. The TRFSH1 signal from the DRAM Controller is asserted during a DRAM refresh cycle and is used to clear the RFSH_REQ signal. The EN_67KHZ# signal is used by the Watchdog Timer circuit for timing purposes. This signal is derived from the REF_CLK (divided by 240) and then synchronized to CLK (EN_67KHZ# is low for one CLK cycle every 15 µs). The FPGA is connected as an 8-bit device to the processor's slow data bus (SLW_PROC_D[7:0]). Using only eight data bits reduces the number of pins required by the FPGA and simplifies board layout. The Device Control block generates four "write enable" signals for writing data to the FPGA (there are four byte addresses for the FPGA). When data is being read from the FPGA, the Device Control block asserts DATA_BUS_EN, which enables the output drivers for the Three-statable Output Mux (i.e., allows the FPGA to drive the DATA[7:0] bus).

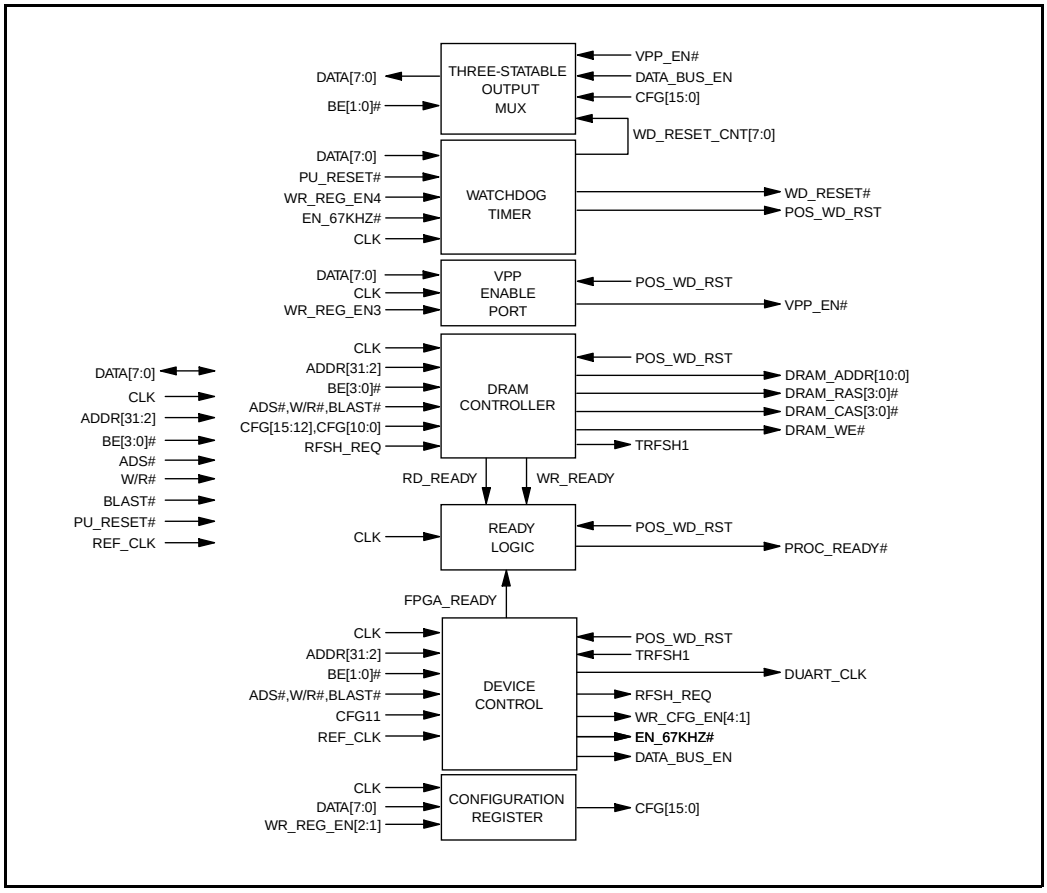


Figure 20. DRAM Controller FPGA Block Diagram

The FPGA responds to reads and writes in a 256 byte memory region located from 0400 0200 to 0400 02FF. This address is fixed inside of the Device Control block's logic. Although the FPGA responds to accesses in a 256 byte memory region, there are only four registers that can be read or written. This results from the fact that not all address bits are used in the chip select decoding logic. ADDR[7:2] are not used and therefore are "don't cares" in the chip select decoding logic. This results in address "aliasing," in which a register can be accessed by more than one address.

3.11.2 Registers

Tables 11 and 12 define the DRAM Controller FPGA registers.

Table 11. DRAM Controller FPGA Write Registers

Primary Address	Description
4000 0200	**Configuration Register (Low Byte)**
4000 0201	**Configuration Register (High Byte)**
4000 0202	**VPP Enable Register** - Writing to this register enables/disables the VPP supply to the Flash memory. VPP is enabled to the Flash memory (allowing the Flash memory to be written) only if the VPP_EN# is asserted. Writing a value of 0xa3 to this register will assert VPP_EN#. Writing any value other than 0xa3 will cause VPP_EN# to be deasserted. VPP_EN# is deasserted as the result of a power up or Watchdog Timer reset.
4000 0203	**Watchdog Timer Register** - Writing a value of 0xf3 resets the Watchdog Timer. Writing any value other than 0xf3 to this register has no effect. The Watchdog Timer is disabled following a power up reset and remains disabled until this register is written with a value of 0xf3. When enabled, the Watchdog Timer will generate a 980 ms reset pulse if not updated with a value of 0xf3 for a period of 14.74 seconds.

Table 12. DRAM Controller FPGA Read Registers

Primary Address	Description
4000 0200	**Configuration Register (Low Byte)**
4000 0201	**Configuration Register (High Byte)**
4000 0202	**VPP Enable/WD Status** - Reading from this location returns the status of the VPP_EN# signal in bit 0 and the Watchdog Timer in bit 7. Bits 1 to 6 should be considered undefined. If bit 0 is a 1, VPP_EN# is asserted (i.e., VPP_EN# is a logic 0). If bit 0 is a 0, VPP_EN# is deasserted. Thus, the value returned in bit 0 is inverted from the VPP_EN# signal. If bit 7 is a 1, the Watchdog Timer is disabled. If bit 7 is a 0, the Watchdog Timer is enabled. The CPU can use the value of bit 7 to determine the source of a CPU reset. Following reset, if this bit is a 1 the reset was caused by a power up reset, whereas a 0 indicates that the source was a Watchdog Timer reset.
4000 0203	**Watchdog Timer Reset Count Register** - Reading from this location returns the number of Watchdog Timer resets that have been generated mod 256 (i.e., the counter will roll over from 255 to 0).

The Configuration register is 2 bytes, with each byte written individually. However, software can treat the Configuration register as a 16-bit word and the processor will perform a burst access where the bytes are read or written consecutively. As shown in Tables 11 and 12, the value of the Configuration register can be read back. The Configuration register is used primarily for configuring the DRAM Controller. Of the 16 bits in the Configuration register, 15 define operation of the DRAM Controller and one is unused.

Table 13. DRAM Controller FPGA Configuration Register Bit Definitions

Configuration Register Bit(s)	Description
15	**DRAM Type -** Specifies the type of DRAM installed. The type of DRAM affects the timing of the DRAM control signals. Note that EDO DRAM will work with either FPM or EDO compatible signals but FPM DRAM will only work with FPM compatible signals. 1 - selects Fast Page Mode (FPM) compatible signals. 0 - selects Extended Data Out (EDO) compatible signals.
14	**SIMM Banks** - Selects the number of DRAM banks to assume for each DRAM SIMM (attached to the processor) in the reference design. There are two DRAM SIMM sockets attached to the processor in the reference design. DRAM SIMMs are available in either single or dual bank configurations. 1 - selects Dual Bank mode (total of four DRAM banks possible). 0 - selects Single Bank mode (total of two DRAM banks possible).
13:12	**DRAM Depth** - Specifies the depth of the DRAM banks, which is related to the number of address bits required. For example, selecting 256 K means that the DRAM banks are 256Kx32 and require nine address signals. 00 - 256 K 01 - 1 M 1x - 4 M
11	**Unused** - This configuration bit is unused.
10	**ADS Delay** - Selects whether the ADS# signal is delayed by one clock signal before being used by the DRAM Controller. 0 - ADS# not delayed 1 - ADS# delayed one clock cycle
9	**Refresh RAS Pulse Width** - Specifies the number of clock cycles the RAS pulses are asserted during DRAM refresh cycles. 0 - two clock cycles 1 - three clock cycles
8	**Minimum RAS Precharge Time -** Determines the minimum number of clock cycles that RAS will be deasserted before another DRAM access is performed. Note that this number can be higher if a DRAM access is not performed immediately following the current DRAM access. 0 - selects one clock cycle. 1 - selects two clock cycles.

Table 13. DRAM Controller FPGA Configuration Register Bit Definitions

7:4	**Wait State Profile for DRAM Writes** - CFG[7:6] is NWDD, CFG[5:4] is NWAD. 0x0x: 1,1,1,1 1x0x: 1,2,2,2 0x10: 2,1,1,1 1x10: 2,2,2,2 0x11: 3,1,1,1 1x11: 3,2,2,2
3:0	**Wait State Profile for DRAM Reads** - CFG[3:2] is NRDD, CFG[1:0] is NRAD. 000x: 1,0,0,0 (Not valid for FPM) 0010: 2,0,0,0 (Not valid for FPM) 0011: 3,0,0,0 (Not valid for FPM) 010x: 1,1,1,1 (Not valid for FPM) 0110: 2,1,1,1 0111: 3,1,1,1 1x0x: 1,2,2,2 (Not valid for FPM) 1x10: 2,2,2,2 1x11: 3,2,2,2

NOTES:

1. "x" represents a "don't care" value.
2. **Following a reset, the Configuration register contents will be 0xFFFF.** This provides the most conservative timing configuration possible, allowing the processor to access devices until they have been configured for a more optimal performance.

The use of the Configuration register bits is explained in more detail in the following sections, which describe individual blocks of the DRAM Controller FPGA.

Notes on Wait State Profile nomenclature:

The wait state profile is defined as four numbers that indicate the number of wait states that are inserted for a burst access in which four reads or four writes are performed. The first number represents the wait states inserted between the address cycle of a bus access and the clock cycle when data is actually read or written. This value is also referred to as NWAD for write accesses and NRAD for read accesses. The second, third, and fourth values represent the wait states inserted between data cycles of a burst access. For example, the last value is the number of wait states inserted between the third data cycle and the fourth data cycle. In general, the wait states inserted between data cycles is the same regardless of the data cycle. The wait states inserted between data cycles is also referred to as NWDD for write accesses and NRDD for read accesses.

The total number of clock cycles required to perform an access can be determined from the wait state profile. One clock cycle is always required to output the address and start the access (the address cycle TA) and one clock cycle is required for each data access performed (TD). Recovery cycles (TR) may also be performed by the processor to allow a device driving the Address/Data bus to turn off its output drivers. A recovery cycle is always performed by an 80960Jx processor. Additional recovery cycles can be inserted in an 80960Jx bus access through the use of the RDYRCV# signal. Recovery cycles can be programmed for the 80960Cx and 80960Hx processors, although in general, recovery cycles are not required because these processors have separate address and data buses.

Example:

Assume the wait state profile is 2,1,1,1 (NRAD=2, NRDD=1) for read accesses and 3,2,2,2 (NWAD=3, NWDD=2) for write accesses and the processor is an 80960Jx that inserts one recovery cycle at the end of the bus access. Tables 14 and 15 show the number of clock cycles required for burst lengths of 1, 2, 3, and 4. Table 14 shows the actual numbers and Table 15 shows the equations used to derive the values in Table 14.

Table 14. Example of Clock Cycles Required for Read/Write Accesses

Number of Words Read or Written	Clock Cycles for Read Access	Clock Cycles for Write Access
1	5	6
2	7	9
3	9	12
4	11	15

Table 15. Equations for Table 14

Number of Words Read or Written	Clock Cycles for Read Access	Clock Cycles for Write Access
1	TA+NRAD+TD+TR	TA+NWAD+TD+TR
2	TA+NRAD+1*(NRDD+TD)+TR	TA+NWAD+1*(NWDD+TD)+TR
3	TA+NRAD+2*(NRDD+TD)+TR	TA+NWAD+2*(NWDD+TD)+TR
4	TA+NRAD+3*(NRDD+TD)+TR	TA+NWAD+3*(NWDD+TD)+TR

3.11.3 DRAM Controller Design

The design of the DRAM Controller is based on the state diagram shown in Figure 21.

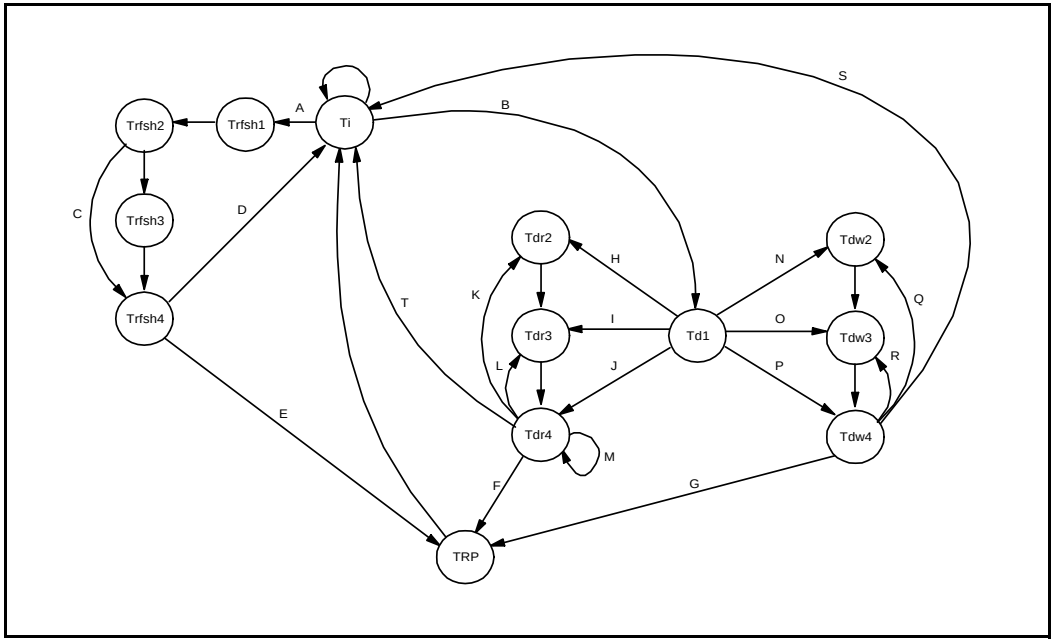


Figure 21. DRAM Controller State Diagram

DRAM Controller State Change Conditions

A: RFSH_REQ

B: !RFSH_REQ & DRAM_RW_REQUEST

C: CFG9 = 0

D: CFG8 = 0

E: CFG8 = 1

F: (BLAST# = 0) & (CFG8 = 1)

G: (BLAST# = 0) & (CFG8 = 1)

H: (W/R# = 0) & (CFG[1:0] = 11)

I: (W/R# = 0) & (CFG[1:0] = 10)

J: (W/R# = 0) & (CFG[1:0] = 0x)

K: (BLAST# = 1) & (CFG[3:2] = 1x)

L: (BLAST# = 1) & (CFG[3:2] = 01)

M: (BLAST# = 1) & (CFG[3:2] = 00)

N: (W/R# = 1) & (CFG[5:4] = 11)

O: (W/R# = 1) & (CFG[5:4] = 10)

P: (W/R# = 1) & (CFG[5:4] = 0x)

Q: (BLAST# = 1) & (CFG[7:6] = 1x)

R: (BLAST# = 1) & (CFG[7:6] = 0x)

S: (BLAST# = 0) & (CFG8 = 0)

T: (BLAST# = 0) & (CFG8 = 0)

As shown in the state diagram and Table 13, some timing parameters are determined by the value of the configuration. This flexibility was designed into the DRAM Controller because the processor clock frequency and DRAM characteristics (type and speed) are not fixed. The ability to modify the DRAM Controller characteristics

allows the reference design to be used as an evaluation platform while selecting the optimum wait state profile for the processor bus frequency, DRAM type, and DRAM speed.

Figure 22 shows the block diagram of the DRAM Controller. The State Machine block implements the state diagram shown in Figure 21 with some additions that aid in generating the DRAM signals. The following outputs indicate when the state machine is in each of the states shown in the state diagram:

TRFSH[4:1], TDR[4:2], TDW[4:2], TD1, TI

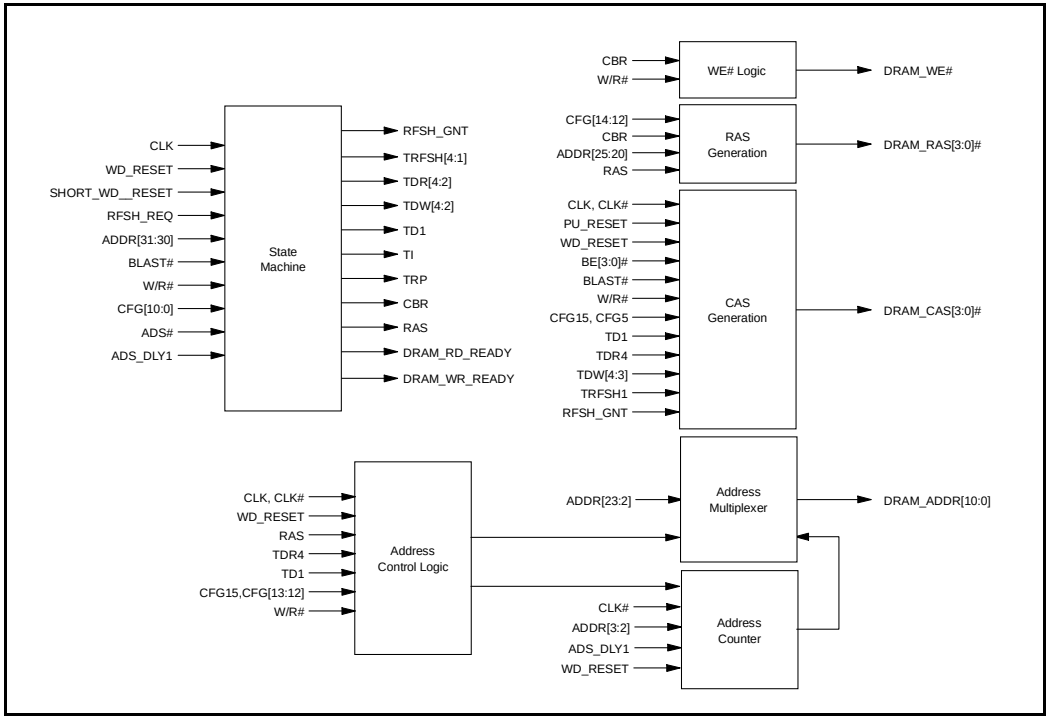


Figure 22. DRAM Controller Block Diagram

The other outputs from the State Machine block are described below:

RAS - Indicates when RAS should be asserted to the DRAM memory banks. The RAS Generation block determines which memory banks will have their RAS signals asserted when RAS is asserted. RAS signal is asserted by the State Machine block when the state machine is in one of the following states:

TD1, TDR2, TDR3, TDR4, TDW2, TDW3, TDW4, TRFSH2, TRFSH3, TRFSH4

CBR - Indicates when a CAS Before RAS refresh cycle is being performed. CBR is asserted by the State Machine block when the state machine is in one of the TRFSHx states.

RSH_GNT - Indicates that the next state in the state machine is TRFSH1 (i.e., a refresh cycle request has been granted).

DRAM_RD_READY - Indicates that the next state in the state machine is TDR4. See Ready Generation description below.

DRAM_WR_READY - Indicates that the next state in the state machine is TDW4. See Ready Generation description below.

The State Machine block has inputs that are not referenced in the state diagram. The purpose of these inputs is described below:

WD_RESET, SHORT_WD_RESET - See Reset description below.

ADS#, ADDR[31:30] - These signals are used to determine when a DRAM request is being requested by the processor. When ADS# is asserted and ADDR[31:30] = 00, the processor is starting an access to the DRAM.

ADS_DLY1 - This signal is ADS# delayed by one clock cycle (and inverted).

CFG10 - This signal determines whether the ADS# signal used by the state machine (in determining when a DRAM access is requested) should be delayed by one clock cycle.

Sheets 7, 8, and 9 of the DRAM Controller FPGA Schematic show the logic for the State Machine block.

Ready Generation

The state machine asserts one of two ready indicators that are used by the Ready Logic Block. The DRAM_RD_READY is asserted one clock prior to entering state TDR4, and DRAM_WR_READY is asserted one clock prior to entering state TDW4. The Ready Logic block asserts the PROC_READY# signal one clock cycle after a "ready indicator" is asserted (i.e., clocked through a Flip-Flop). Thus, PROC_READY# is asserted whenever the state machine is in state TDR4 or TDW4.

Sheet 9 of the DRAM Controller FPGA Schematic shows the source of DRAM_RD_READY and DRAM_WR_READY. Sheet 1 of the schematic shows the PROC_READY# logic. Note that the PROC_READY# output is a "driven tri-state" (DTS). A DTS output acts like an open-drain output, allowing multiple sources for a signal (the signal is pulled high by a resistor - "Wired AND" configuration). However, a DTS output is actively driven high, eliminating the delay that occurs while the signal capacitance is charged through the pull-up resistor. The DTS output is implemented by enabling the output driver whenever the signal is asserted (low) and for one clock cycle after the signal is deasserted. This implementation can be seen in the PROC_READY# logic.

Note that DTS output can only be used only where there is a maximum of one source active at any given time for a signal and where there is a delay of one clock before another source drives the signal. This is true of PROC_READY# in the reference design, because PROC_READY# is driven high during the bus recovery cycle for an 80960Jx processor and during the address cycle for an 80960Cx/Hx processor. There is no source driving PROC_READY# during the cycle immediately following the address cycle because there are no devices that have zero wait states from the address cycle to the first data cycle.

Reset

Note that some Flip-Flops in the state machine are reset by WD_RESET (Watchdog Reset) and others are reset by SHORT_WD_RESET (Short Watchdog Reset). The reset signal used is based on the Flip-Flop's use during a refresh cycle. If the Flip-Flop must be functional for a refresh cycle to be performed (e.g., TRFSH[4:1]), the Flip-Flop is reset by SHORT_WD_RESET; otherwise, the Flip-Flop is reset by WD_RESET. This allows the DRAM to be refreshed while a Watchdog Timer reset is taking place because the SHORT_WD_RESET signal is active for only one clock cycle at the beginning of a Watchdog Timer reset.

DRAM Refresh

The state machine generates "CAS Before RAS" refresh cycles. This type of refresh cycle does not require the DRAM address lines to be controlled and is relatively simple to implement. A refresh cycle is requested by the RFSH_REQ signal. Every 15 µs, RFSH_REQ is asserted, indicating a refresh cycle should be performed. The state machine will then begin a refresh cycle if currently in state TI. The RFSH_REQ signal is deasserted when the state machine enters TRFSH2. If a DRAM access is in progress when RFSH_REQ is asserted, the DRAM access is completed, then the refresh cycle is performed. Note that a refresh cycle has higher priority than a processor DRAM access request. For example, if a refresh request and a

DRAM access request are asserted on the same clock cycle, a refresh cycle is performed first. The DRAM access request will be performed when the refresh cycle has completed. The delay of a DRAM access request results in more wait states being inserted between the Address cycle and the first data cycle. Note that the refresh cycle will not interfere or delay processor accesses to other devices (e.g., Flash). Figure 23 shows the DRAM refresh timing.

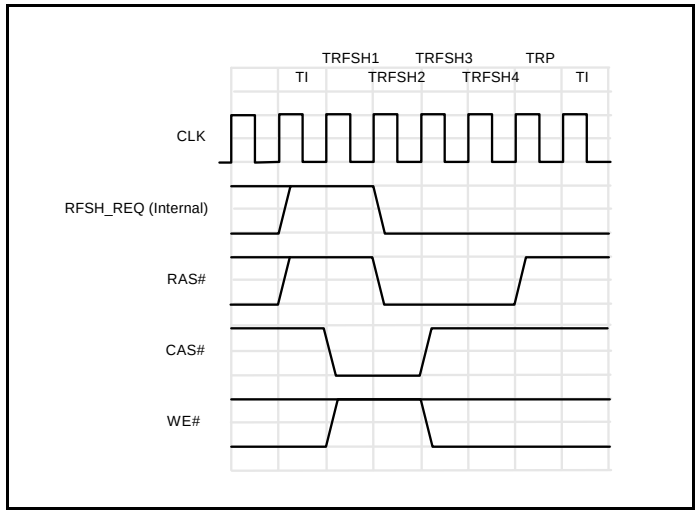


Figure 23. DRAM Refresh Timing

State TRFSH3 can be removed by setting bit 9 of the Configuration register (CFG9) to 0. This reduces the number of clock cycles RAS is asserted from three to two. The need for TRFSH3 is determined from the DRAM's T_{RAS} parameter (RAS# pulse width) and the processor bus clock frequency.

Table 16. T_{RAS} Values for Refresh Cycles

Processor Bus Frequency (MHz)	Clock Cycle Time (ns)	T_{RAS} (ns, CFG9=0)	T_{RAS} (ns, CFG9=1)
16	62	124	186
20	50	100	150
25	40	80	120
30	33	66	99
33	30	60	90

A DRAM is normally referred to by its access time (i.e., 70 ns DRAM) from RAS↓ (T_{RAC}), which is also the same value as the T_{RAS} parameter. Using the data in Table 16, the recommended programming of CFG9 for common speeds of DRAM and various processor bus clock frequencies is summarized in Table 17.

Table 17. Recommended CFG9 Programming for Common DRAM Speeds

Processor Bus Frequency (MHz)	60 ns DRAM	70 ns DRAM	80 ns DRAM
16	0	0	0
20	0	0	0
25	0	0	0
30	0	1	1
33	0	1	1

NOTE:

1. The refresh cycle is the same regardless of the type of DRAM (FPM or EDO).

RAS Precharge Time

The state machine contains a state, TRP, which determines the minimum RAS precharge time (T_{RP}) for the DRAM. The RAS precharge time specifies the length of time that RAS must be deasserted between DRAM operations (read, write, refresh). RAS is deasserted during states TI and TRP and asserted during all other states. Therefore, T_{RP} will always be at least one clock cycle (state TI) and can be two clock cycles if the DRAM Controller is programmed to transition to state TRP before entering state TI. Bit 8 of the Configuration register (CFG8) determines whether state TRP is entered. When CFG8 is a 1, state TRP is entered and the minimum RAS precharge time is two clock cycles. When CFG8 is a 0, state TRP is skipped and the minimum RAS precharge time is one clock cycle. Table 18 shows the possible minimum values of T_{RP} at various processor clock frequencies and values of CFG8.

Table 18. Possible T_{RP} Values

Processor Bus Frequency (MHz)	Clock Cycle Time (ns)	T_{RP} (ns, CFG8=0)	T_{RP} (ns, CFG8=1)
16	62	62	124
20	50	50	100
25	40	40	80
30	33	33	66
33	30	30	60

The minimum value of T_{RP} for common DRAM device speeds is shown in Table 19.

Table 19. Minimum RAS Precharge Time Values for Common DRAM Speeds

DRAM Speed (ns)	Minimum T_{RP} (ns)
60	40
70	50
80	60

Based on Tables 18 and 19, the recommended programming of CFG8 for common speeds of DRAM and various processor bus clock frequencies are summarized in Table 20.

Table 20. Recommended CFG8 Programming for Common DRAM Speeds

Processor Bus Frequency (MHz)	60 ns DRAM	70 ns DRAM	80 ns DRAM
16	0	0	0
20	0	0	1
25	0	1	1
30	1	1	1
33	1	1	1

WE# Logic

The logic for generating the DRAM WE# signal is relatively simple. The processor W/R# is inverted and routed to the DRAM unless a DRAM refresh cycle is being performed. WE# is deasserted (forced high) during a DRAM refresh cycle. This is done for compatibility with older DRAM chips, which required WE# to be deasserted on the falling edge of RAS# during “CAS Before RAS” refresh cycles. For newer DRAM chips, the value of WE# is a “don’t care” during a “CAS Before RAS” refresh.

The WE# logic is located on Sheet 1 of the DRAM Controller FPGA Schematic and consists only of an OR gate that is part of the output pad.

DRAM Timing Diagrams

This section contains a number of timing diagrams showing signal relationships for the DRAM Controller. These figures are used by the sections which follow that discuss the Address, RAS, and CAS generation sections of the DRAM Controller. The figures show both the DRAM Controller signals and the 80960 processor signals. The figures assume an 80960Jx processor because a bus recovery cycle is shown. Note that the type of DRAM affects read signal timing but not write signal timing.

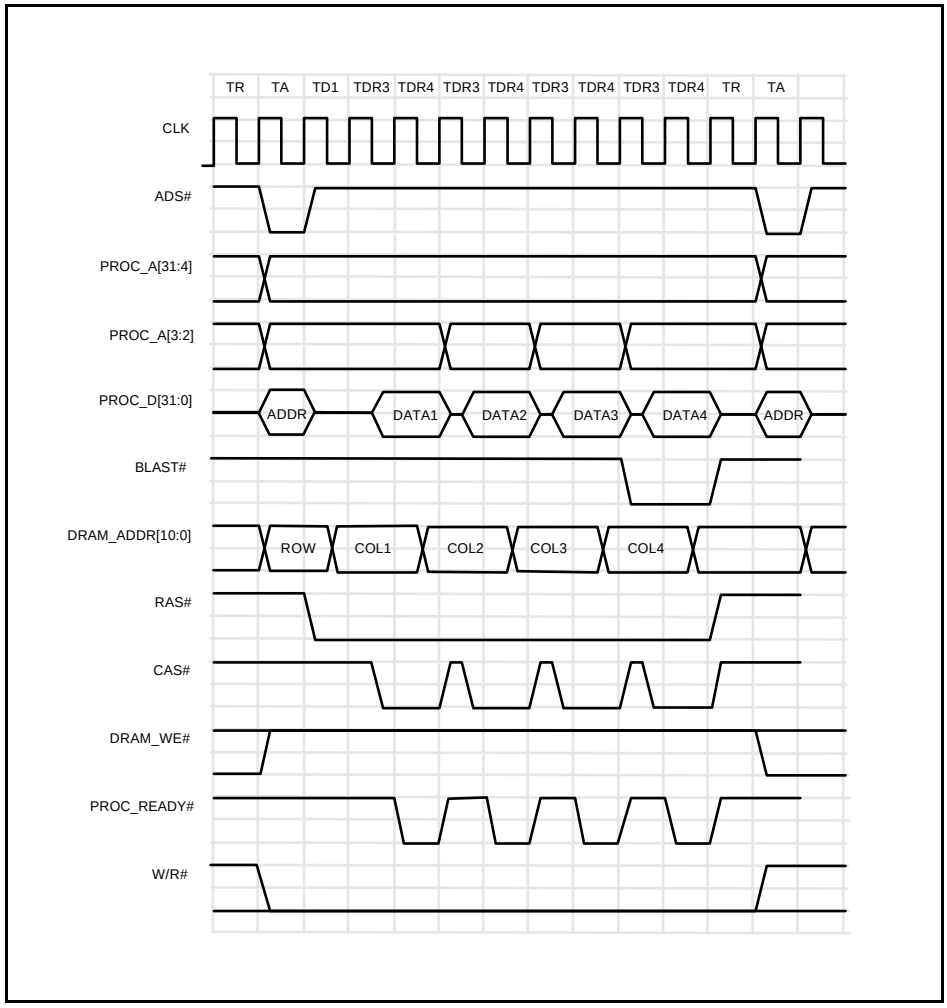


Figure 24. Fast Page Mode Read With 2,1,1,1 Wait State Profile

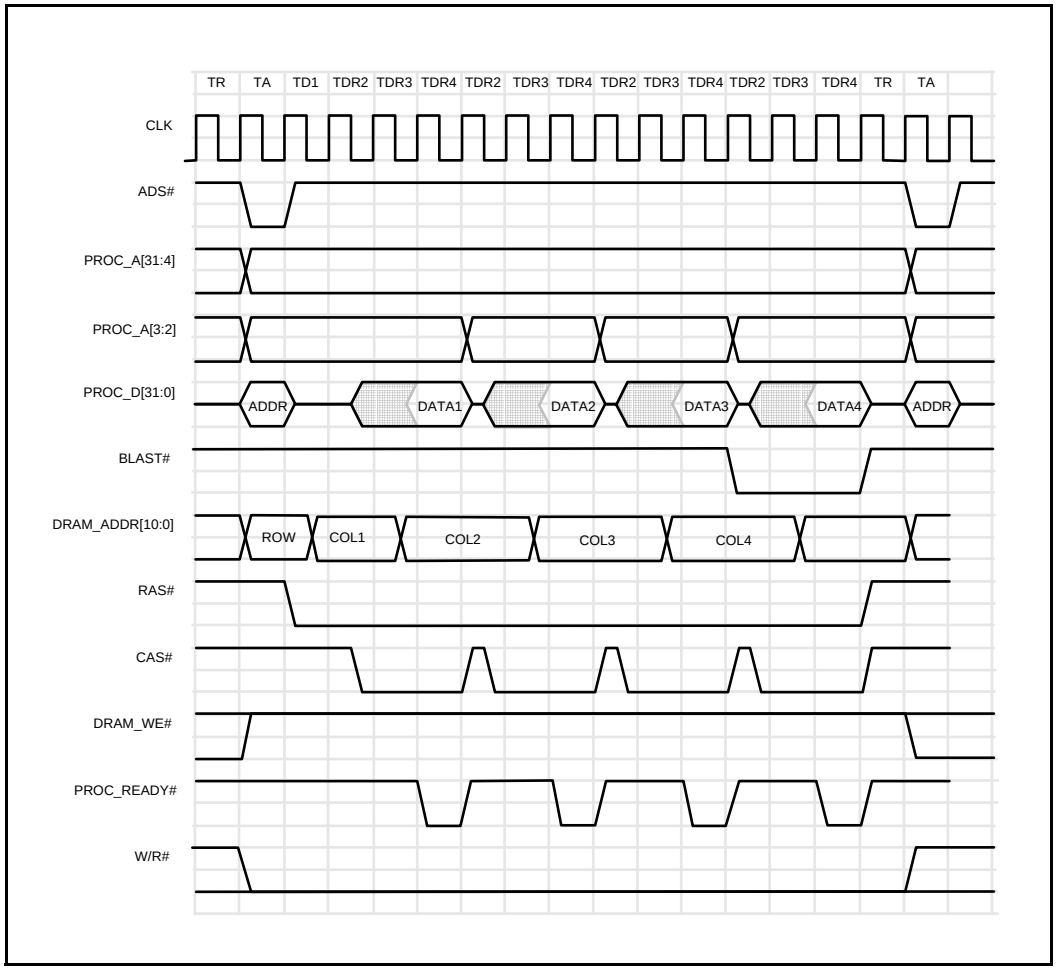


Figure 25. Fast Page Mode Read With 3,2,2,2 Wait State Profile

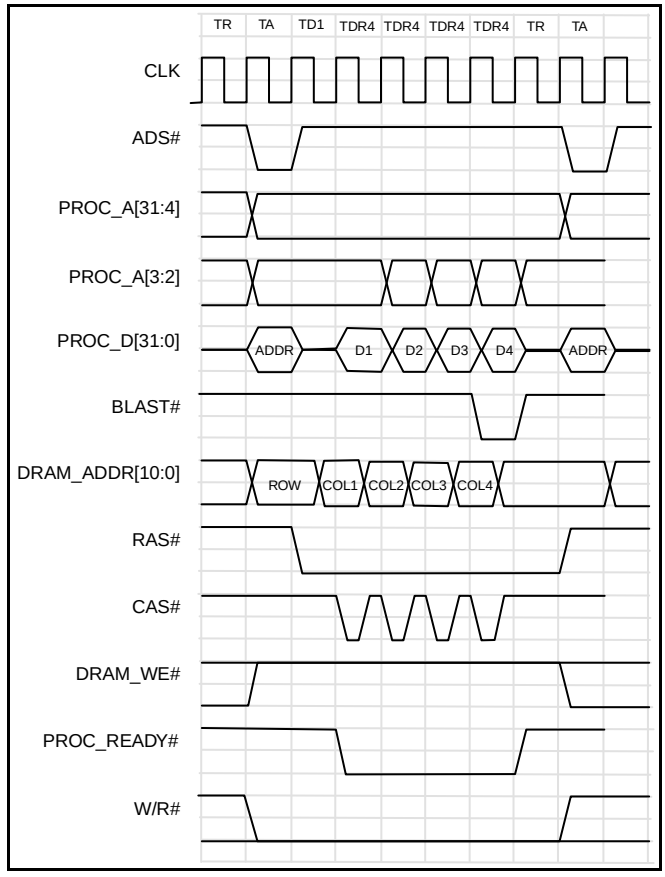


Figure 26. EDO Read With 1,0,0,0 Wait State Profile

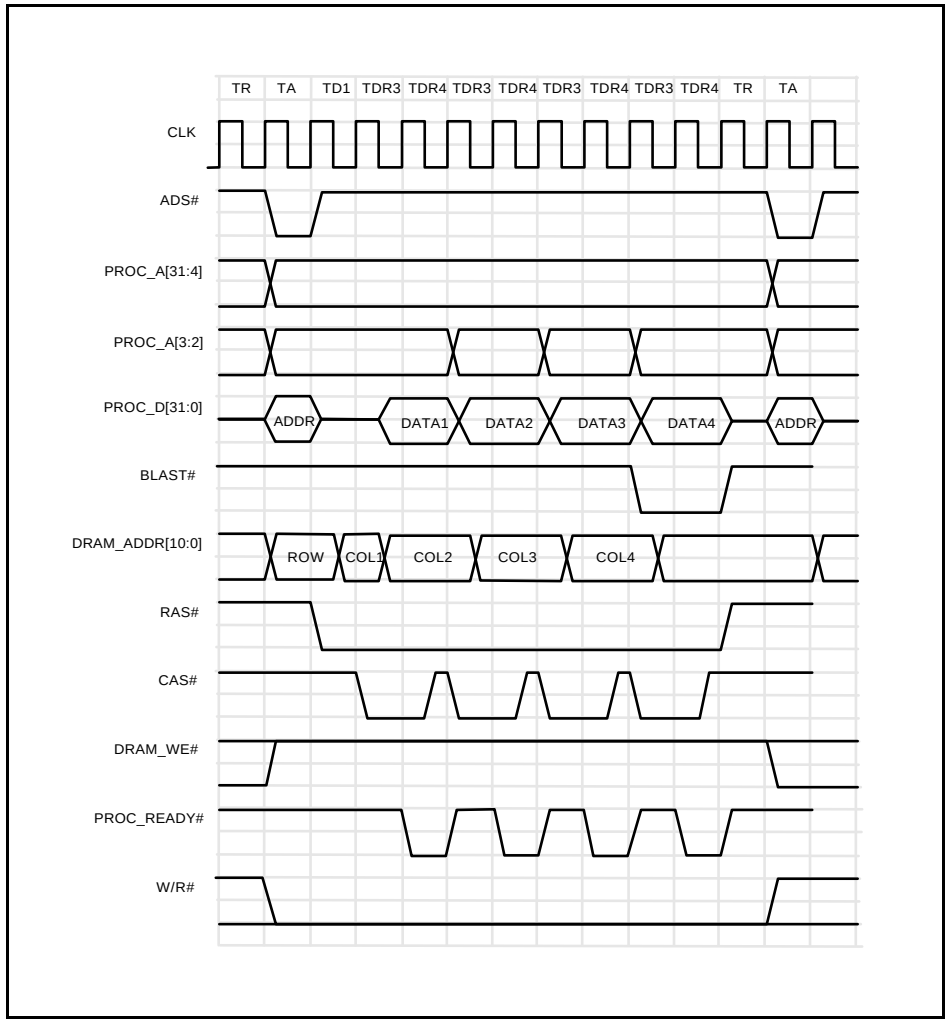


Figure 27. EDO Read With 2,1,1,1 Wait State Profile

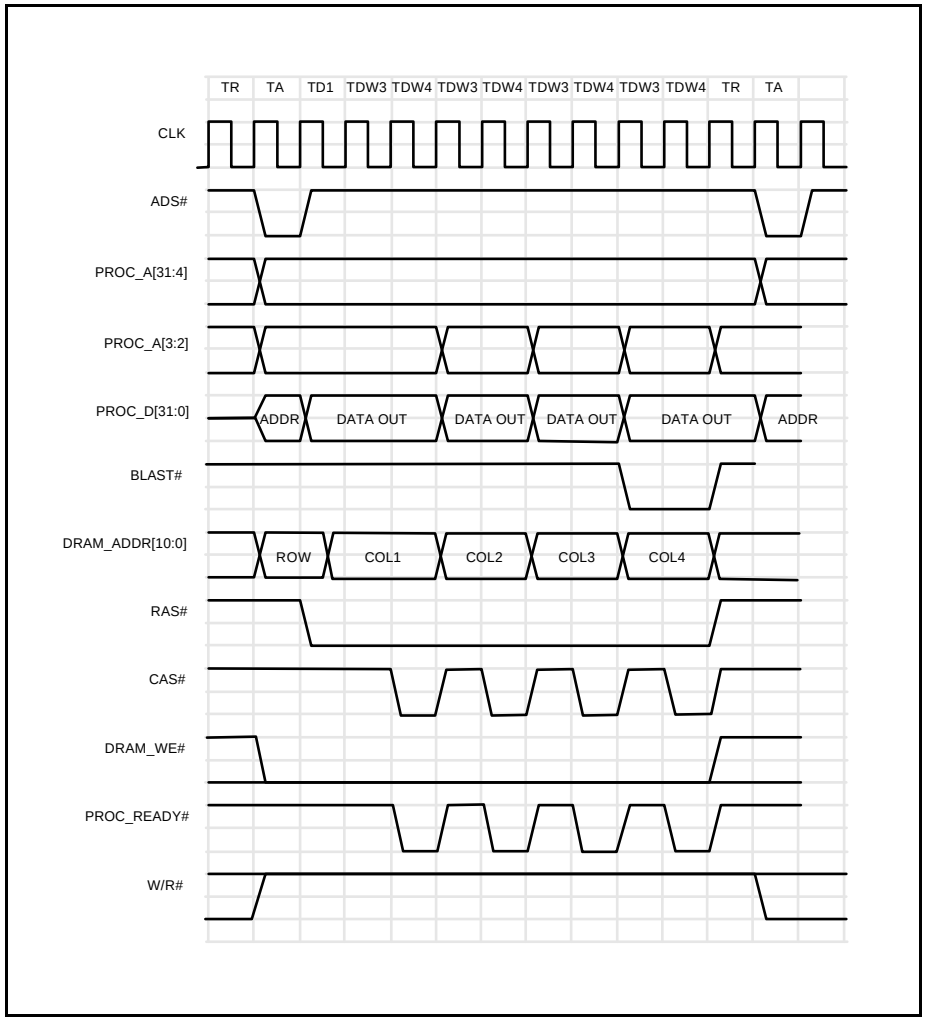


Figure 28. DRAM Write With 2,1,1,1 Wait State Profile

DRAM Addresses

DRAM devices have their addresses multiplexed to reduce the number of pins required on the device. The address information consists of two parts: row address and column address. The row address is output to the DRAM, then RAS (Row Address Strobe) is asserted, the column address is output, and CAS (Column Address Strobe) is asserted. The DRAM will then output (or write) the addressed location. Additional locations can be accessed in the same row by applying a different column address and asserting CAS again. Figures 24 to 28 show the signal relationships related to the address multiplexing for quad DRAM read and write accesses.

There are 11 DRAM address lines (DRAM_ADDR[10:0]) output by the DRAM Controller FPGA. This number of address lines support DRAMs with depths up to 4 M (2^{22}), since the address is 22 bits due to the multiplexing of row and column addresses. Smaller depths are also supported by ignoring the upper one (1 M depth) or two (256 K depth) address lines.

Sheet 11 of the DRAM Controller FPGA Schematic shows the logic for generating the DRAM address signals.

The DRAM address logic consists of a 10-bit multiplexer that selects between the row and column address, a 2-bit counter, and logic for controlling the multiplexer and counter. Table 21 shows the signals that are output for the row and column addresses. ADDR[23:2] are the processor address signals, CNTR[1:0] are the outputs from the 2-bit counter, and GND is a logic 0.

Table 21. DRAM Address Sources

Signal	Source for Row Address	Source for Column Address	Notes
DRAM_ADDR0	ADDR11	ADDR2/CNTR0	1
DRAM_ADDR1	ADDR12	ADDR3/CNTR1	1
DRAM_ADDR2	ADDR13	ADDR4	
DRAM_ADDR3	ADDR14	ADDR5	
DRAM_ADDR4	ADDR15	ADDR6	
DRAM_ADDR5	ADDR16	ADDR7	
DRAM_ADDR6	ADDR17	ADDR8	
DRAM_ADDR7	ADDR18	ADDR9	
DRAM_ADDR8	ADDR19	ADDR10	
DRAM_ADDR9	ADDR21/GND	ADDR20/GND	2
DRAM_ADDR10	ADDR23/GND	ADDR22/GND	2

NOTES:

1. During DRAM write accesses, the lower two bits of the column address are ADDR[3:2]. During DRAM read accesses, the lower two bits of the column address are CNTR[1:0].
2. The upper two bits of the row and column address are dependent on the DRAM device depth that has been programmed into the FPGA Configuration register (bits 13 and 12). When programmed for a device depth of 4 M, the upper two row addresses are ADDR23 and ADDR21 and the upper two column addresses are ADDR22 and ADDR20. When programmed for a device depth of 2 M, the upper two row addresses are GND and ADDR21 and the upper two column addresses are GND and ADDR20. When programmed for a device depth of 256 K, the upper two row addresses are GND and the upper two column addresses are GND.

The multiplexer selects between row and column addresses based on the RAS signal from the DRAM Controller State Machine block. The RAS signal is delayed a half clock period before being used as the control signal for the multiplexer. The row address is output when RAS (delayed by half a clock cycle) is deasserted, and the column address is output when RAS (delayed by half a clock cycle) is asserted.

The WRITE signal (processor's W/R# signal) is used to select the source for the lower two bits of the column address. During DRAM write accesses, the processor's address signals (ADDR[3:2]) are output as the lower two bits of the column address. However, this cannot be done when reading from the DRAM. This results from the fact that, during a burst access, the lower two bits of the column address need to be output before the processor increments its lower two bits. Therefore, a 2-bit counter is used to

generate the lower two bits of the column address during DRAM read accesses. Only a 2-bit counter is required because the 80960 processor ends a burst access if ADDR[3:2] is equal to 11. The counter is initially loaded with the value of ADDR[3:2], which becomes the column address for the first DRAM read access. For EDO DRAM, the counter is then incremented on the negative edge of the clock when the DRAM Controller State Machine is in the state that follows TD1 or TDR4. This state is dependent on the programmed wait state profile. For FPM DRAM, the counter is incremented one clock later than it would be for EDO DRAM (i.e., the second state following TD1 or TDR4).

Refer to Figures 24 to 27 for examples of the column address timing during DRAM read accesses.

Notes:

1. The FPGA Schematic has several optimizations to improve timing characteristics. For example, the RAS signal from the state machine is clocked through two Flip-Flops, generating two identical control signals for the address multiplexer. The use of two Flip-Flops allows the multiplexer control signal to be “buffered” without the resulting delay of two buffers following a single Flip-Flop.

2. The 2-bit counter logic is placed on the schematic instead of the CNTR2 component. This allows the logic module locations of the counter Flip-Flops to be fixed using the “Place” attribute.

RAS Generation

Sheets 1 and 3 of the DRAM Controller FPGA Schematic in Appendix D show the logic for generating the DRAM RAS signals. Sheet 3 contains most of the logic but Sheet 1 has the output pads, which include an OR gate.

The RAS Generation logic determines which of the four RAS outputs (DRAM_RAS[3:0]#) is asserted when RAS is asserted by the DRAM Controller State Machine. To reduce the number of logic delays, the OR gate that is part of the QuickLogic FPGA output pads is used to provide the gating as shown on Sheet 1 of the DRAM Controller FPGA Schematic.

The RAS signal(s) to be enabled is dependent on the type of DRAM access being performed (refresh or read/write), the DRAM depth, the number of DRAM banks on each SIMM, and the DRAM address being accessed.

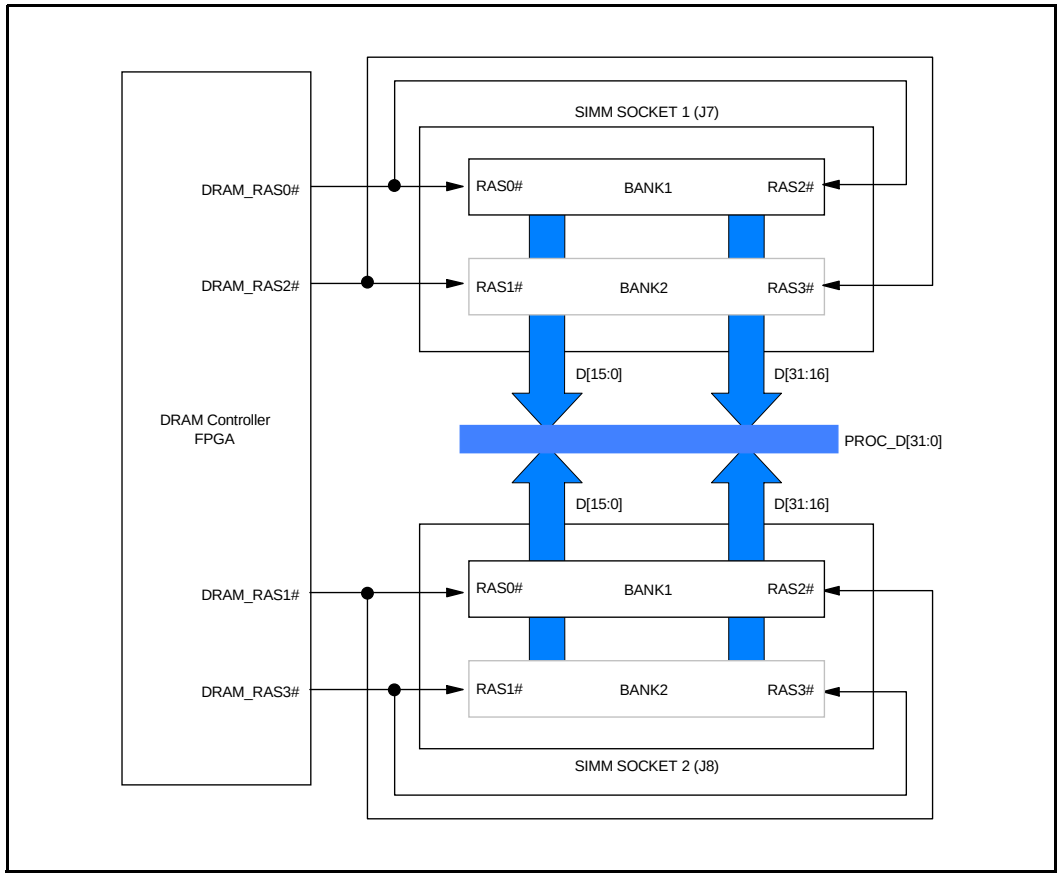


Figure 29. RAS Signal Connections to Processor DRAM SIMM Sockets

DRAM is installed in the reference design using 72-pin SIMMs (Single In-line Memory Module), which are installed in sockets. There are two sockets available on the reference design, J7 and J8. The use of SIMM sockets allows various configurations of memory (size and type) to be installed for evaluation purposes. A 72-pin SIMM can be configured as 16- or 32-bit wide memory. This is possible because the memory is divided into two separate 16-bit data buses with a separate RAS signal controlling each bus. If the SIMM is used as 16-bit memory, the two data buses are tied together and separate RAS signals are generated. If the SIMM is used as 32-bit memory, the two data buses form a 32-bit bus and the RAS signals are tied together.

Figure 29 shows a block diagram of the SIMM socket implementation in the reference design, in which the SIMMs are configured as 32-bit wide memory. When configured as 32-bit memory, there can be one or two banks of DRAM on the SIMM. For example, a 256Kx32 SIMM has one bank and a 512Kx32 SIMM has two banks. The number of RAS signals that need to be generated is thus determined by the number of DRAM banks installed, which is a function of the number of SIMMs installed and the number of DRAM banks on each SIMM. There can be one, two, three, or four DRAM banks installed, with the possible configurations shown in Table 22. The use of a single bank SIMM in socket 1 and a dual bank SIMM in socket 2 is possible but is not recommended, because the DRAM memory will not appear as contiguous memory in the address space.

Table 22. Possible DRAM SIMM Configurations

DRAM Banks	Socket 1 SIMM Type	Socket 2 SIMM Type
1	Single Bank	None
2	Single Bank	Single Bank
2	Dual Bank	None
3	Dual Bank	Single Bank
4	Dual Bank	Dual Bank

The type of SIMM(s) that are installed is specified as a configuration option. Bit 14 of the DRAM Controller FPGA Configuration register (CFG14) indicates whether the RAS generation logic should assume single bank SIMMs (CFG14 = 0) or dual bank SIMMs (CFG14 = 1).

As shown in Figure 29, there are four RAS outputs from the DRAM Controller FPGA (DRAM_RAS[3:0]#). When the DRAM Controller is configured for single bank SIMMs, DRAM_RAS2# and DRAM_RAS3# will never be asserted by the RAS generation logic. The decoding of the address logic will then locate the bank selected by DRAM_RAS0# and the bank selected by DRAM_RAS1# in contiguous memory addresses (explained in more detail later). When the DRAM Controller is configured for dual bank SIMMs, all four RAS signals are generated. The RAS generation logic will order the DRAM banks in the address space as follows: Socket 1/Bank 1 (RAS0#), Socket 1/Bank 2 (RAS2#), Socket 2/Bank1 (RAS1#) and Socket 2/Bank 2 (RAS3#). Note that RAS2# is generated at a lower address than RAS1# in this case. This was done so that when only one dual bank SIMM is installed, the DRAM memory is contiguous in the address space.

For refresh cycles, all RAS signals are asserted, refreshing all DRAM banks simultaneously. Note that when programmed for single bank SIMMs, DRAM_RAS2# and DRAM_RAS3# are never asserted (including refresh cycles).

For DRAM read or write accesses, only one RAS signal is asserted to indicate the addressed DRAM bank. The processor address bits that are decoded to identify the addressed DRAM bank are dependent on the DRAM depth. The DRAM depth is specified by bits 12 and 13 in the DRAM Controller FPGA Configuration register. As shown on Sheet 3 of the DRAM Controller FPGA Schematic, CFG[13:12] are used as control signals for a multiplexer to select the address bits to be decoded. Table 23 shows the address bits selected for the programmed DRAM depth.

Table 23. Address Bits Used in RAS Generation

CFG[13:12]	DRAM Depth	Address Bits Selected
00	256Kx32	ADDR[21:20]
01	1Mx32	ADDR[23:22]
1x	4Mx32	ADDR[25:24]

Note that there are several processor address bits that are not used in the address decoding for the DRAM. ADDR[31:30] are used by the DRAM Controller State Machine to determine if a DRAM access is being made. ADDR[29:26] are never used in the DRAM address decoding. ADDR[25:24] is used only when the DRAM depth is 4 Mx32 and ADDR[23:22] is used when the DRAM depth is 1 Mx32 or 4 Mx32. This should be noted because it results in “aliasing” of the DRAM memory. In other words, a location in DRAM will have many processor addresses that can be used to access it.

Aliased DRAM can be useful if it is desirable to treat sections of the DRAM differently. For example, the data cacheability of a memory location for an 80960 processor is determined by its address. Blocks of memory can be set as cacheable while other blocks can be set as non-cacheable. The data cacheability of memory is programmable in the processor.

Notes:

The DRAM Controller will respond to DRAM accesses as if the entire address space from 0000 0000 to 3FFF FFFF is occupied by DRAM. In other words, the DRAM Controller will generate a ready acknowledgment for any access in this address space even when the RAS signal is not generated for an address.

The following examples are provided to clarify DRAM addressing.

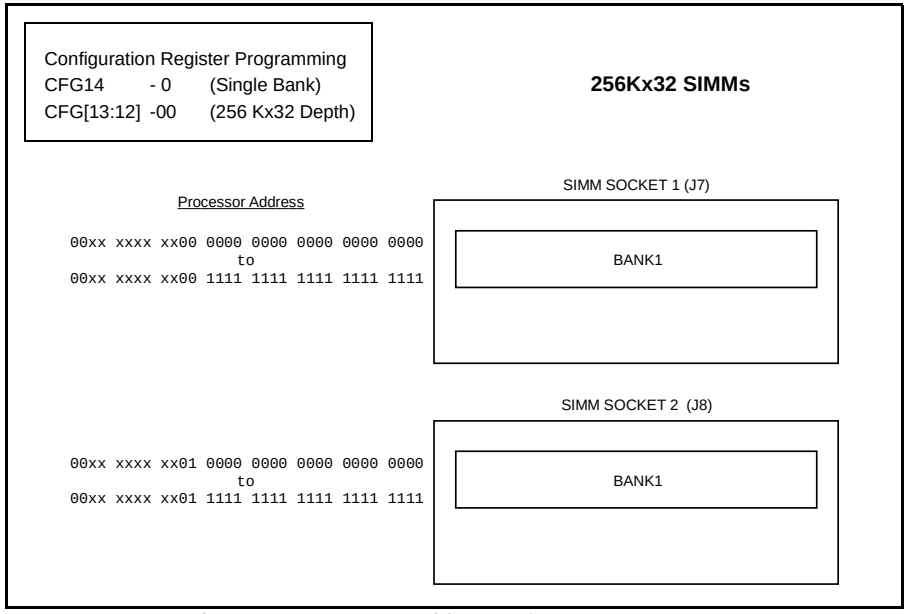


Figure 30. Processor Addresses for 256Kx32 SIMMs

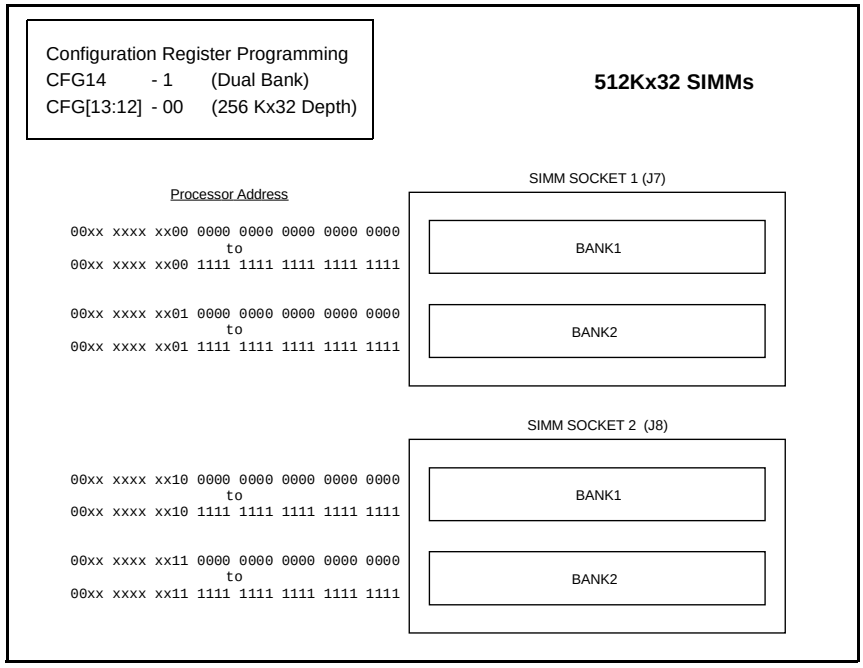


Figure 31. Processor Addresses for 512Kx32 SIMMs

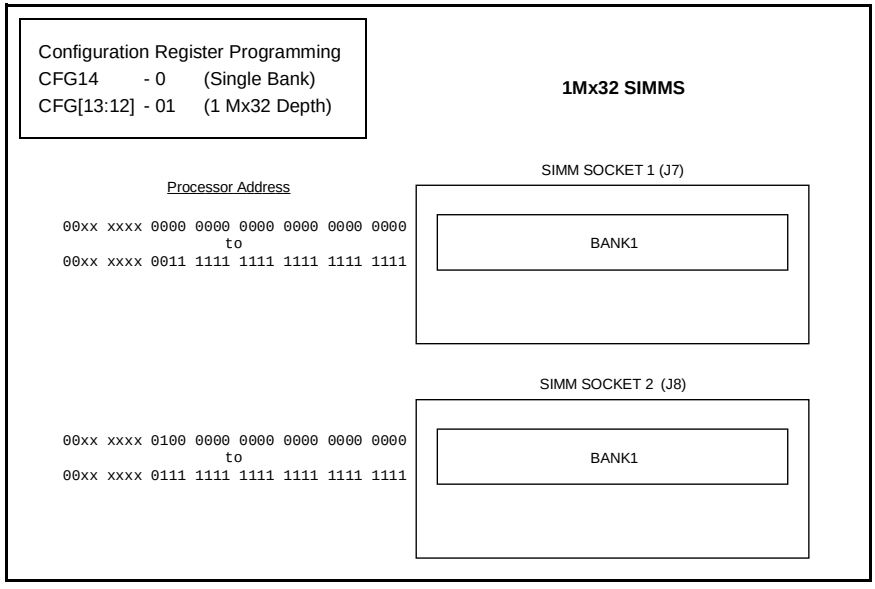


Figure 32. Processor Addresses for 1Mx32 SIMMs

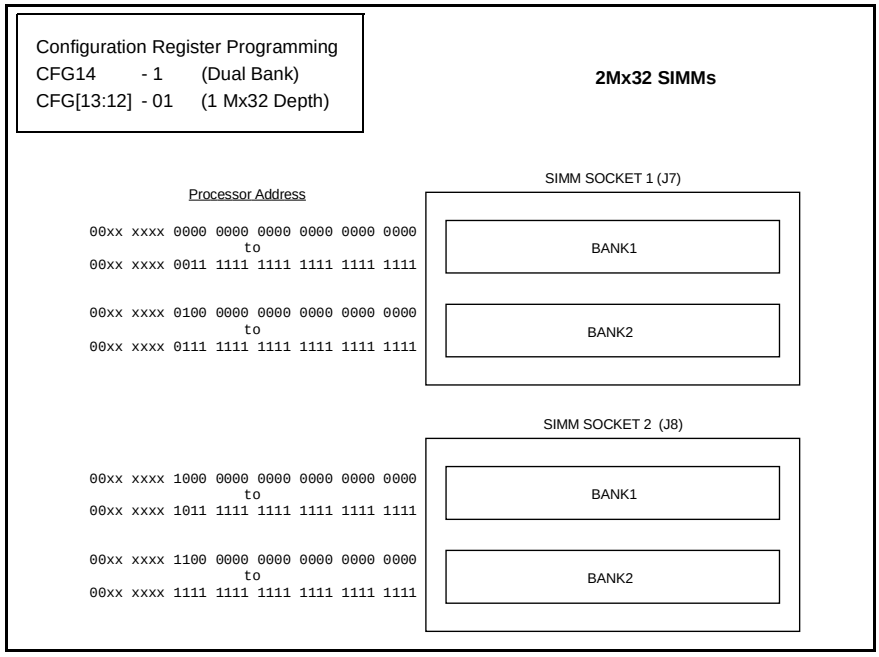


Figure 33. Processor Addresses for 2Mx32 SIMMs

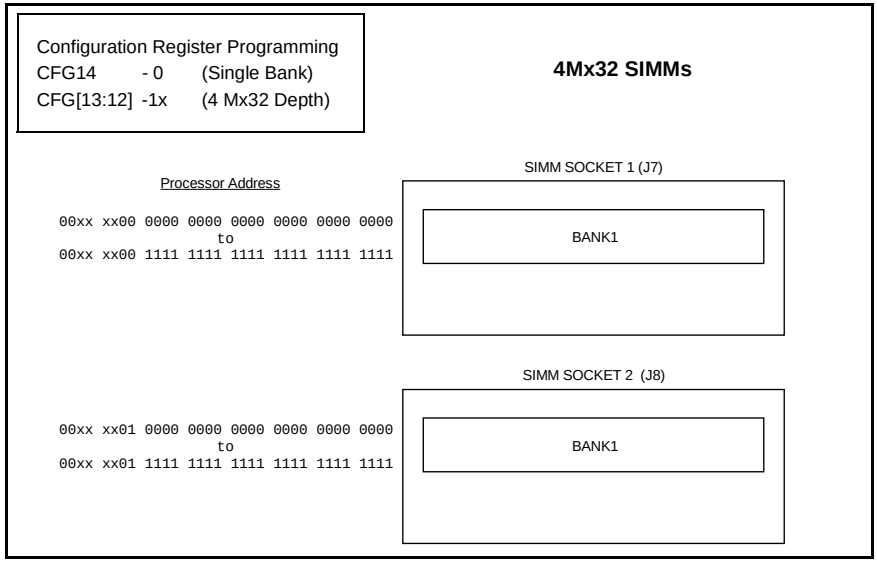


Figure 34. Processor Addresses for 4Mx32 SIMMs

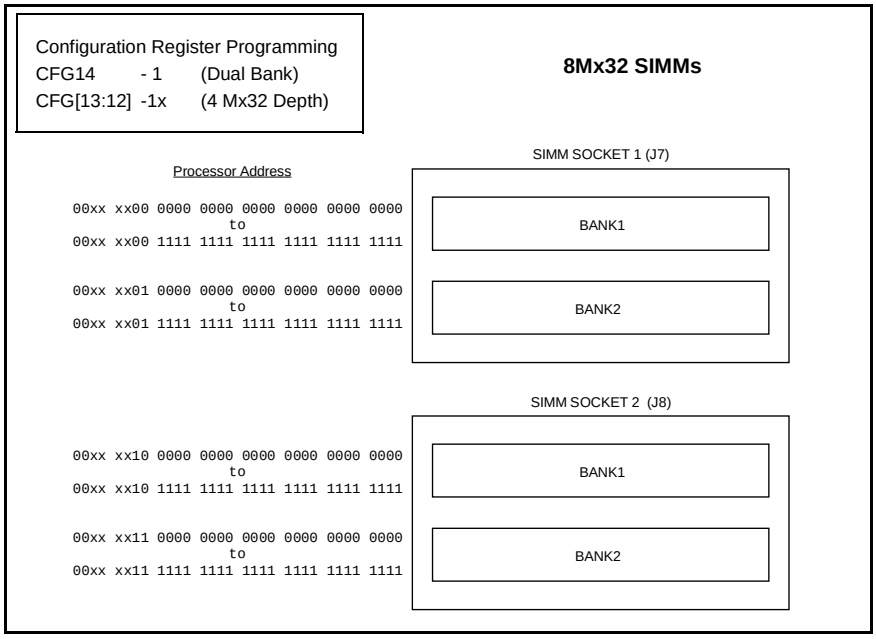


Figure 35. Processor Addresses for 8Mx32 SIMMs

CAS Generation

Sheets 1 and 11 of the DRAM Controller FPGA Schematic show the logic for generating the DRAM CAS signals. Sheet 11 contains most of the logic but Sheet 1 has the output pads which include an OR gate.

The generation of the CAS signals is dependent on the type of DRAM (FPM or EDO) installed. EDO is an enhanced version of FPM DRAM. For FPM DRAM read accesses, the data outputs are tri-stated when CAS is deasserted. This requires that CAS remain asserted until the data has been loaded into the processor. EDO DRAM does not tri-state its data outputs during read accesses until both CAS and RAS are deasserted. This allows the CAS signal to be deasserted earlier in the access, which allows CAS to be reasserted earlier for a subsequent access. Note that if only one location is read from DRAM, the performance of EDO and FPM DRAM will be the same. The advantage of EDO DRAM is for burst accesses. The difference in FPM and EDO signal timing can be seen in Figures 24 to 27.

DRAM Timing

A number of different timing parameters related to the DRAM interface need to be determined to select the appropriate DRAM Controller configuration.

The analysis will first examine the timing associated with generating the RAS signals, which is independent of the type of DRAM. The first area of concern is the ability to assert the appropriate RAS# signal in the clock cycle that immediately follows the address cycle (T_A). This requires that the FPGA recognize that a bus access is occurring (ADS# asserted) and the access is to the DRAM memory space (PROC_A[31:30] = 00). A timing analysis of the DRAM Controller FPGA indicates that the following input setup times must be met (referenced to the rising edge of CLK following the TA cycle):

- Address input setup time of 7 ns minimum.
- ADS# input setup time of 11 ns minimum.

Tables 24 and 25 show the time required for the address and ADS# signals to satisfy the FPGA timing. Note that the address has a margin of -2 ns at 33 MHz (30 ns clock cycle)

and only 1 ns at 30 MHz (33 ns clock cycle). The ADS# propagation delay has a margin of 2 ns at 33 MHz and 5 ns at 30 MHz. Because of these small or negative margins, the FPGA can be configured to delay the ADS# signal by one clock cycle before the DRAM Controller uses it. This results in one extra clock cycle for the address and ADS# signals to propagate. Note that ADS# is still sampled (to create the delayed version), but that the setup time is reduced to 6 ns because there is less delay in this logic path within the FPGA. These timings are based on an 80960Jx processor. When an 80960Cx/Hx processor is used, the margin for the address propagation delay will be positive because the delay caused by the address latch is removed.

Table 24. Address Propagation Delay Analysis

Parameter	Value	Units
Clock Skew	2	ns max
Processor Address Output Delay	15	ns max
Address Latch Delay	8	ns max
Input Setup Time	7	ns min
Total	32	ns

Table 25. ADS# Propagation Delay Analysis

Parameter	Value	Units
Clock Skew	2	ns max
Processor Output Delay	15	ns max
Input Setup Time	11	ns min
Total	28	ns

Based on the above results, the recommended programming of the FPGA configuration is to delay ADS# for an 80960Jx processor operated at 30 MHz or 33 MHz and for an 80960Cx/Hx Processor operated at 33 MHz. The delaying of ADS# to the DRAM Controller logic is controlled by bit 10 of the Configuration register.

The next set of timing parameters that will be examined are those related to the row address setup and hold times relative to the falling edge of RAS. The row address setup time (T_{ASR}) will be the address setup time into the FPGA plus the delay of the RAS signal from CLK minus the propagation delay of the address inputs to the DRAM_ADDR outputs. Timing analysis of the FPGA shows that the delay of RAS from the clock is slightly more than the propagation delay of the processor address to DRAM_ADDR. At the worst case operating conditions (high temperature, low voltage, slowest processor) the RAS delay is 13.9 ns, compared with 13.0 ns for the address. At the best operating conditions, the RAS delay is 5.0 ns, compared with 4.8 ns for the address. Therefore, to make the analysis simpler, the value of T_{ASR} will be considered equal to the address input setup time to the FPGA. T_{ASR} is normally specified as 0 ns minimum for 60, 70, and 80 ns versions; therefore, T_{ASR} is always satisfied because the address setup time must be 7 ns minimum for proper operation of the FPGA (discussed earlier in this section).

The row address is held for ½ clock cycle after RAS is asserted. Thus, the row address hold time (T_{RAH}) will be the width of the clock plus the propagation delay of the column address from CLK↓ minus the propagation delay of RAS from CLK↑. The propagation delay of the column address from CLK↓ is 15.8 ns to 20.3 ns minimum for the

worst case conditions and 5.7 ns to 7.5 ns for the best case conditions. The range is because there are 11 DRAM address lines and they have different propagation delays. From the T_{ASR} discussion, the RAS delays are 5.0 ns (best case) and 13.9 ns (worst case). Table 26 shows the calculated value of T_{RAH} for various processor clock frequencies, assuming a 40% duty cycle for the clock. T_{RAH} is normally specified as 10 ns minimum for 60, 70, and 80 ns DRAM. This value is satisfied at all processor clock speeds, as shown in Table 26.

Table 26. T_{RAH} Values versus Processor Clock Frequency

Clock Frequency (MHz)	Best Case Value	Worst Case Value	Units
16	24	25	ns min
20	20	21	ns min
25	16	17	ns min
30	13	14	ns min
33	12	13	ns min

The next timing parameter that will be examined is the column address setup time (T_{ASC}). The column address changes ½ of a clock cycle prior to CAS being asserted in certain modes. This will occur only when the DRAM Controller is configured for EDO DRAM. For FPM DRAM, there is always a full clock cycle for the column address to be setup. The column address setup time will be the clock width plus the CAS output delay from CLK↑ minus the column address delay from CLK↓. The CAS output delay from CLK↑ is 3.4 to 4.3 ns (best case conditions) and 9.2 to 11.7 ns (worst case conditions). The column address delay from CLK↓ is 5.7 to 8.0 ns (best case conditions) and 15.8 to 20.3 ns (worst case conditions). Table 27 shows the calculated value of T_{ASC} for various processor frequencies, assuming a 40% duty cycle for the clock. T_{ASC} is normally specified as 0 ns minimum for 60, 70, and 80 ns DRAM. This value is satisfied at all processor clock speeds, as shown in Table 27, although there is little margin at the higher clock frequencies.

Table 27. T_{ASC} Values versus Processor Clock Frequency

Clock Frequency (MHz)	Best Case Value	Worst Case Value	Units
16	19	13	ns min
20	15	9	ns min
25	11	5	ns min
30	8	2	ns min
33	7	1	ns min

The next timing parameter that will be examined is the column address hold time (T_{CAH}). The column address changes ½ of a clock cycle after CAS is asserted when the DRAM Controller is configured for EDO DRAM. For FPM DRAM, there is always a full clock cycle for the column address to be setup. The column address hold time is the clock width minus the CAS output delay from CLK↑ plus the column address delay from CLK↓. The CAS output delay from CLK↑ is 3.4 to 4.3 ns (best case conditions) and 9.2 to 11.7 ns (worst case conditions). The column address delay from CLK↓ is 5.7 to 8.0 ns (best case conditions) and 15.8 to 20.3 ns (worst case conditions). Table 28 shows the calculated value of T_{CAH} for various processor frequencies, assuming a 40% duty cycle for the clock. T_{CAH} is normally specified as 10 ns min for 60, 12 ns min for 70 ns DRAM, and 15 ns min for 80 ns DRAM. **Table 28 shows that T_{CAH} cannot be met for 80 ns EDO DRAM at processor clock frequencies above 25 MHz.**

Table 28. T_{CAH} Values versus Processor Clock Frequency

Clock Frequency (MHz)	Best Case Value	Worst Case Value	Units
16	25	28	ns min
20	21	24	ns min
25	17	20	ns min
30	14	17	ns min
33	13	16	ns min

The next set of timing parameters that will be examined determine the wait state profile required for read accesses. There are three parameters to consider: access time from RAS (T_{RAC}), access time from CAS (T_{CAC}) and access time from column address (T_{AA}). Table 29 shows the values of these parameters for the common DRAM speeds of 60, 70, and 80 ns. The parameter values are the same for both EDO and FPM DRAM.

Table 29. DRAM Access Times

Parameter	60 ns DRAM	70 ns DRAM	80 ns DRAM	Units
T_{RAC}	60	70	80	ns min
T_{CAC}	15	20	20	ns min
T_{AA}	30	35	40	ns min

In calculating the number of wait states required, only the maximum delays of RAS (14 ns), CAS (12 ns), and the DRAM address (21 ns) need to be considered. In addition, the timing calculation needs to account for a clock skew of 2 ns and a data setup time into the CPU of 6 ns. Using these values, the wait state profiles required for FPM and EDO DRAM read access are shown in Tables 30 and 31 for the common DRAM speeds and processor clock frequencies.

Table 30. FPM DRAM Wait State Profiles for Read Accesses

Clock Frequency (MHz)	60 ns DRAM	70 ns DRAM	80 ns DRAM
16	2,1,1,1	2,1,1,1	2,1,1,1
20	2,1,1,1	2,1,1,1	2,1,1,1
25	2,1,1,1	2,1,1,1	2,1,1,1
30	2,1,1,1	2,1,1,1	3,1,1,1
33	2,1,1,1	3,1,1,1	3,1,1,1

Table 31. EDO DRAM Wait State Profiles for Read Accesses

Clock Frequency (MHz)	60 ns DRAM	70 ns DRAM	80 ns DRAM
16	2,0,0,0	2,0,0,0	2,0,0,0
20	2,0,0,0	2,0,0,0	2,0,0,0
25	2,1,1,1	2,1,1,1	2,1,1,1
30	2,1,1,1	2,1,1,1	3,1,1,1 †
33	2,1,1,1	3,1,1,1	3,1,1,1 †

NOTE:

† Not possible because T_{CAH} is violated.

Note that the "effective" wait state profiles at 30 and 33 MHz may have an additional wait state for T_{RAD} if ADS# is delayed by one clock cycle (discussed earlier in this section).

The DRAM write timing parameters are easily met with the timing shown in Figure 28 except for T_{RCD} (RAS# to CAS# Delay). This parameter is specified as 45, 50, and 60 ns minimum for 60, 70, and 80 ns DRAM, respectively. There are two clock cycles available in the timing (for a 2,1,1,1 wait state profile) to satisfy T_{RCD}. However, 80 ns DRAM at 30 and 33 MHz requires a 3,1,1,1 wait state profile. Table 32 summarizes the recommended wait state programming for the various DRAM speeds and processor frequencies.

Table 32. DRAM Wait State Profiles for Write Accesses

Clock Frequency (MHz)	60 ns DRAM	70 ns DRAM	80 ns DRAM
16	1,1,1,1	1,1,1,1	2,1,1,1
20	2,1,1,1	2,1,1,1	2,1,1,1
25	2,1,1,1	2,1,1,1	2,1,1,1
30	2,1,1,1	2,1,1,1	3,1,1,1†
33	2,1,1,1	2,1,1,1	3,1,1,1†

NOTE:

† Not possible because T_{CAH} is violated.

Note: In order to improve some of the timing parameters, the DRAM_CTL.CHP file was edited after the FPGA had been placed but before the FPGA was routed. The logic cells that contain the multiplexers for DRAM_ADDR [4:0] were then moved closer to their output pads by changing the logic cell names as follows:

DRAM_ADDR4 and DRAM_ADDR2:	A12
DRAM_ADDR3:	A13
DRAM_ADDR1:	A14
DRAM_ADDR0:	A15

Logic cells A12 to A15 were unused and DRAM_ADDR4 and DRAM_ADDR2 were located in the same logic cell.

Table 33 lists the recommended programming of the Configuration register based on the processor clock frequency and the DRAM type and speed.

Table 33. DRAM Timing Configuration Summary

Clock Frequency	DRAM Type	Configuration Register
16 MHz	60 ns FPM	1xxx x000 0000 0110
16 MHz	70 ns FPM	1xxx x000 0000 0110
16 MHz	80 ns FPM	1xxx x000 0010 0110
16 MHz	60 ns EDO	0xxx x000 0000 0010
16 MHz	70 ns EDO	0xxx x000 0000 0010
16 MHz	80 ns EDO	0xxx x000 0010 0010
20 MHz	60 ns FPM	1xxx x000 0010 0110
20 MHz	70 ns FPM	1xxx x000 0010 0110
20 MHz	80 ns FPM	1xxx x001 0010 0110
20 MHz	60 ns EDO	0xxx x000 0010 0010
20 MHz	70 ns EDO	0xxx x000 0010 0010
20 MHz	80 ns EDO	0xxx x001 0010 0010
25 MHz	60 ns FPM	1xxx x000 0010 0110
25 MHz	70 ns FPM	1xxx x001 0010 0110
25 MHz	80 ns FPM	1xxx x001 0010 0110
25 MHz	60 ns EDO	0xxx x000 0010 0110
25 MHz	70 ns EDO	0xxx x001 0010 0110
25 MHz	80 ns EDO	0xxx x001 0010 0110
30 MHz	60 ns FPM	1xxx xa01 0010 0110
30 MHz	70 ns FPM	1xxx xa11 0010 0110
30 MHz	80 ns FPM	1xxx xa11 0011 0111
30 MHz	60 ns EDO	0xxx xa01 0010 0110
30 MHz	70 ns EDO	0xxx xa11 0010 0110
30 MHz	80 ns EDO	1xxx xa11 0011 0111
33 MHz	60 ns FPM	1xxx x101 0010 0110
33 MHz	70 ns FPM	1xxx x111 0010 0111
33 MHz	80 ns FPM	1xxx x111 0011 0111
33 MHz	60 ns EDO	0xxx x101 0010 0110
33 MHz	70 ns EDO	0xxx x111 0010 0111
33 MHz	80 ns EDO	1xxx x111 0011 0111

NOTES:

1. “x” represents a bit in the Configuration register that is not related to the DRAM timing.
2. “a” is dependent on processor type: 0 for 80960Cx/Hx Processor, 1 for 80960Jx Processor.
3. 80 ns EDO DRAM at 30 and 33 MHz is configured as FPM DRAM to satisfy T_{CAH}.

3.11.4 FPGA Interface Design

The DRAM Controller FPGA contains logic that allows internal registers to be read and written. This logic, the FPGA Access Controller, is based on the state diagram shown in Figure 36. The FPGA is an 8-bit device on the "slow" data bus. An 8-bit data interface was chosen to reduce the number of pins used on the FPGA. Since the FPGA is rarely accessed, the overhead introduced by performing 8-bit bus accesses is insignificant. The circuit allows burst reads and writes to the FPGA. The burst capability is provided mainly to allow the configuration word (a 16-bit word) to be read and written with a single instruction. The FPGA wait state profiles are 3,2,2,2 for read accesses and 2,1,1,1 for write accesses.

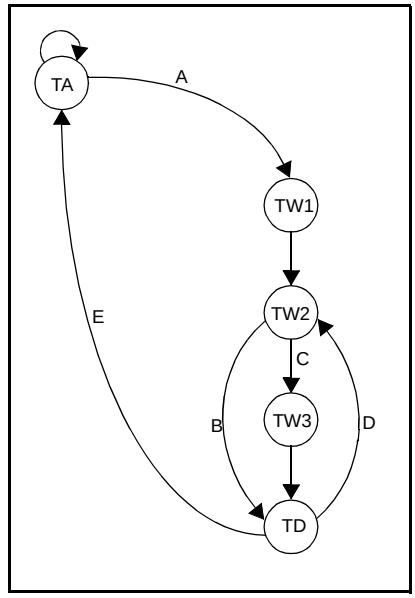


Figure 36. State Diagram for FPGA Access Controller

Table 34. FPGA Access Controller State Changes

A	(ADS# = 0) & (FPGA Selected)
B	W/R# = 1
C	W/R# = 0
D	BLAST# = 1
E	BLAST# = 0

Sheet 6 of the DRAM Controller FPGA Schematic shows the state machine that implements the state diagram in Figure 36. The state machine serves three purposes:

1. **Assert READY# during state TD**. The FPGA_READY signal is asserted to indicate that the next state is TD. FPGA_READY is routed to the READY logic on Sheet 1 of the DRAM Controller FPGA Schematic.

2. **Generate write enable pulses during state TD**. A two-input decoder, located in the lower-right quadrant on Sheet 6 of the Schematic, is used to generate an enable signal during state TD when a write is being performed to the FPGA. The decoder is enabled in state TD (by FPGA_TD) and decodes the BE1# and BE0# signals (lower two address bits for an 8-bit access). The decoder outputs, WR_REG_EN[4:1], enable one of the four FPGA registers to be loaded on the next rising clock edge.

3. **Assert the DATA_BUS_EN signal during FPGA read accesses**. FPGA reads are performed using a multiplexer to select the data to be output (Sheet 5) then enabling the bidirectional buffer (Sheet 2) to output the data on the slow data bus. The DATA_BUS_EN signal is asserted when the FPGA can drive the slow data bus. Its timing is determined by the logic on Sheet 6 of the schematic. Note that DATA_BUS_EN is asserted only during an FPGA read.

Figures 37 to 40 show the timing associated with FPGA read and write accesses. Note that the SLW_BUS_DIR and SLW_BUS_EN# signals, which control the slow data bus transceiver, are generated by the MISC Logic FPGA. The generation of these signals is described in the Section 3.12, Misc Logic FPGA. The timing of these signals mainly affects when the FPGA can enable its output drivers to prevent bus contention between the transceiver and the FPGA.

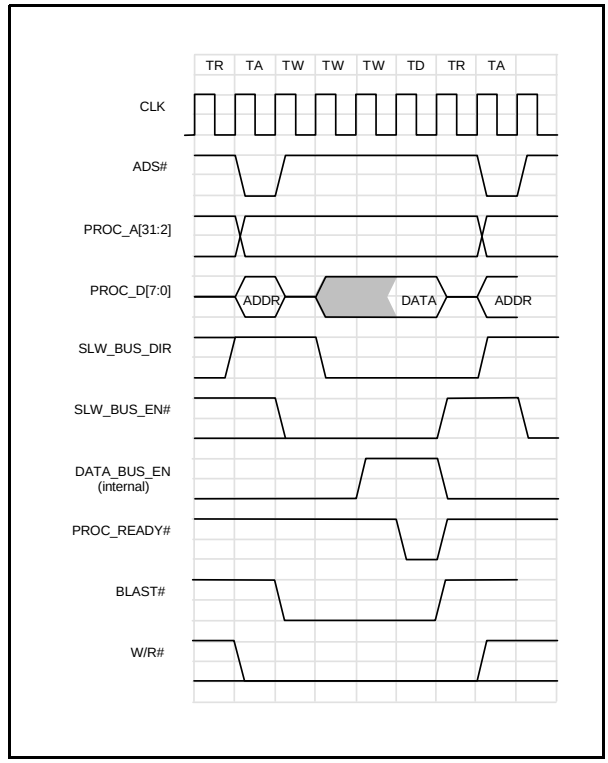


Figure 37. FPGA Single Byte Read

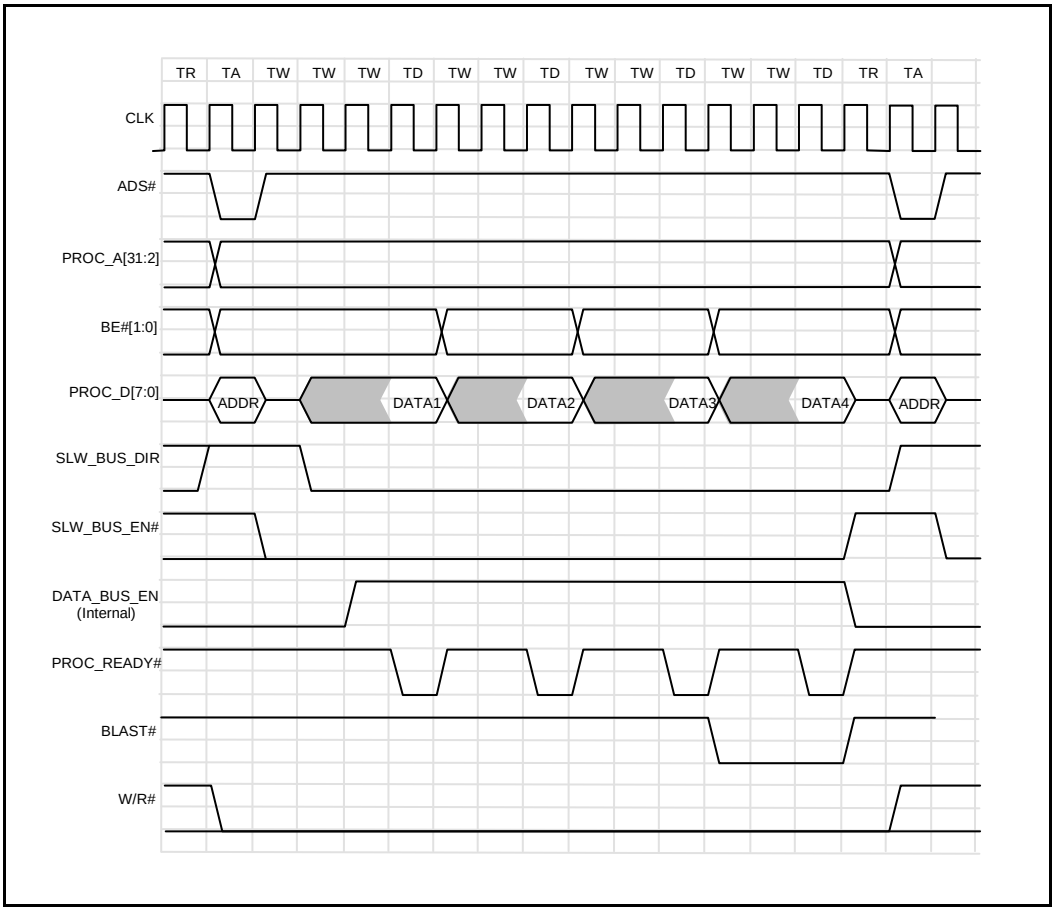


Figure 38. FPGA 4 Byte Burst Read

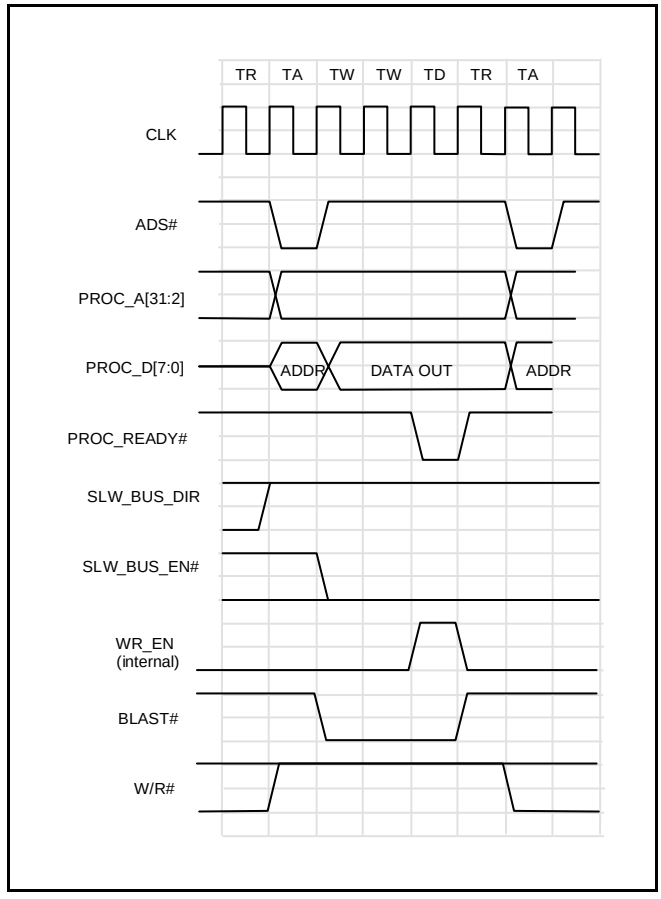


Figure 39. FPGA Single Byte Write

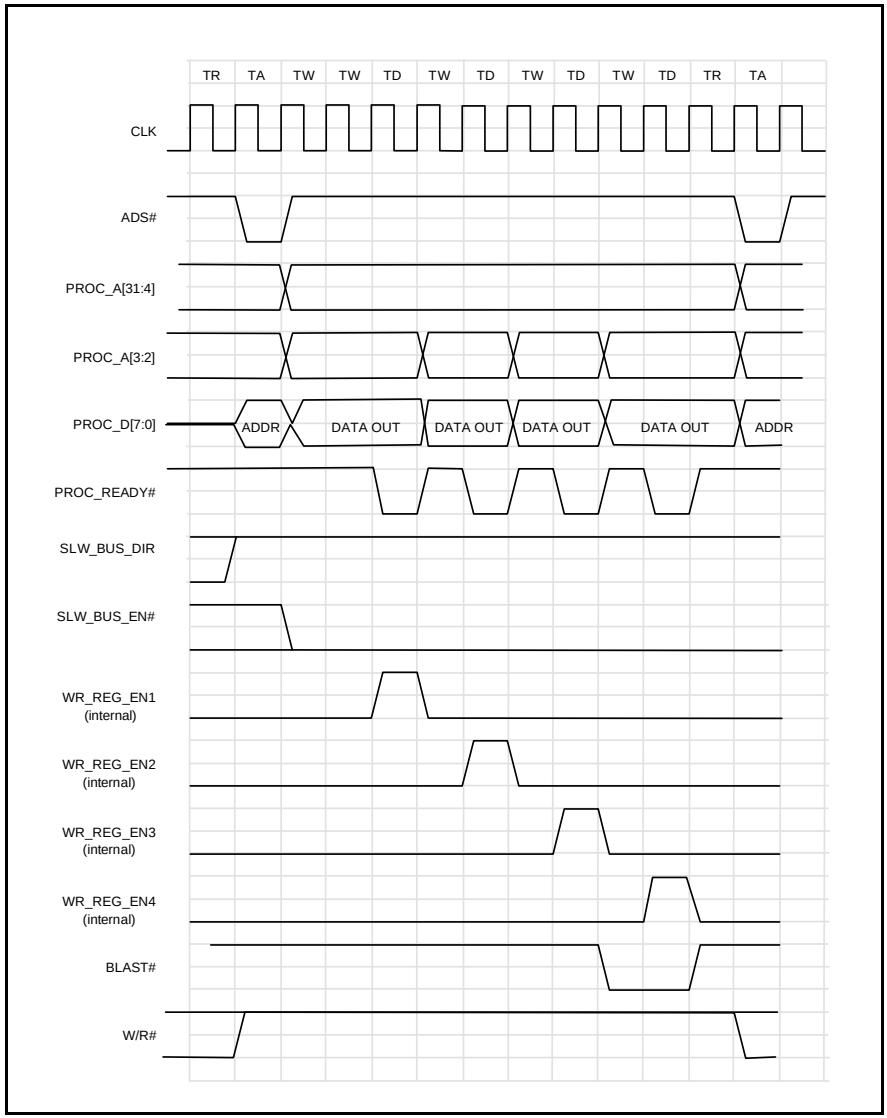


Figure 40. FPGA 4 Byte Burst Write

Notes:

a. SLW_BUS_DIR = 1 when the processor is driving the slow data bus.
b. SLW_BUS_DIR = 0 when a device on the slow data bus is driving the processor local bus.
c. SLW_BUS_EN# controls the output enable on the transceiver (0 = active, 1 = tri-stated).

Timing Analysis

The timing analysis for FPGA reads is determined in two parts. The first timing parameter that must be determined is the number of wait states required from address to first data. This parameter is dependent on the output enable time of the FPGA and is not dependent on the propagation delay of the lower two address bits (BE1# and BE0#), which will have had sufficient time to propagate to the output mux in the FPGA and select the desired data. The second parameter is the number of wait states required between data reads in a burst access. This parameter does not depend on the FPGA Output Enable Delay since the FPGA output is already enabled. Instead, it is dependent on the propagation delay of the address signals (BE1# and BE0#) to the output mux and the delay from the output mux to the FPGA outputs. Table 35 summarizes the delay for the first read. Having three wait states provides 26 ns of margin, which is more than adequate.

Table 35. DRAM Controller FPGA First Read Timing Analysis

Parameter	Value	Units
Optimal Time (two clock cycles)	60	ns min
Clock Skew	2	ns max
FPGA Output Enable Delay	18	ns max
Transceiver Propagation Delay	8	ns max
Processor Data Input Setup Time	6	ns min
Margin	26	ns

Table 36 summarizes the delay for the second to fourth data reads in a burst. Two wait states provides 39 ns of margin, which is more than adequate. Note that the timing analysis did not account for extra bus loading or signal propagation delays on the printed wiring board.

Table 36. DRAM Controller FPGA Burst Read Timing Analysis

Parameter	Value	Units
Optimal Time (three clock cycles)	90	ns min
Clock Skew	2	ns max
Processor Address Output Delay	15	ns min
FPGA Address Select Delay	20	ns max
Transceiver Propagation Delay	8	ns max
Processor Data Input Setup Time	6	ns min
Margin	39	ns

The timing analysis for FPGA writes is essentially the same regardless of which byte in a write access (first to fourth) is being written, with one exception. The exception occurs when an FPGA write follows a read access to the slow data bus. In this case, the enabling of the transceiver outputs after the transceiver direction has been turned around is slower than the propagation of the data and address signals from the processor through the transceiver. For a processor frequency of 33 MHz, Table 37 shows the timing analysis for the first write access and Table 38 shows the analysis for the second to fourth write accesses in a burst. The analysis indicates that a wait state profile of 1,1,1,1 could be used at 33 MHz if desired. Note that the delay of the address latch for an 80960Jx processor, bus loading, and signal delay on the motherboard were not considered in the analysis.

Table 37. DRAM Controller FPGA First Write Timing Analysis

Parameter	Value	Units
Optimal Time (three clock cycles)	90	ns min
Clock Skew	2	ns max
IO_OE# Delay (from Misc Logic FPGA)	13	ns min
Transceiver Output Disable Delay	14	ns max
FPGA Data Input Setup Time	18	ns min
Margin	43	ns

Table 38. DRAM Controller FPGA Burst Write Timing Analysis

Parameter	Value	Units
Optimal Time (two clock cycles)	60	ns min
Clock Skew	2	ns max
Processor Data/Address Output Delay	15	ns max
Transceiver Propagation Delay	8	ns max
FPGA Data Input Setup Time	18	ns min
Total	17	ns

3.11.5 Watchdog Timer Design

Figure 41 shows a block diagram of the Watchdog Timer.

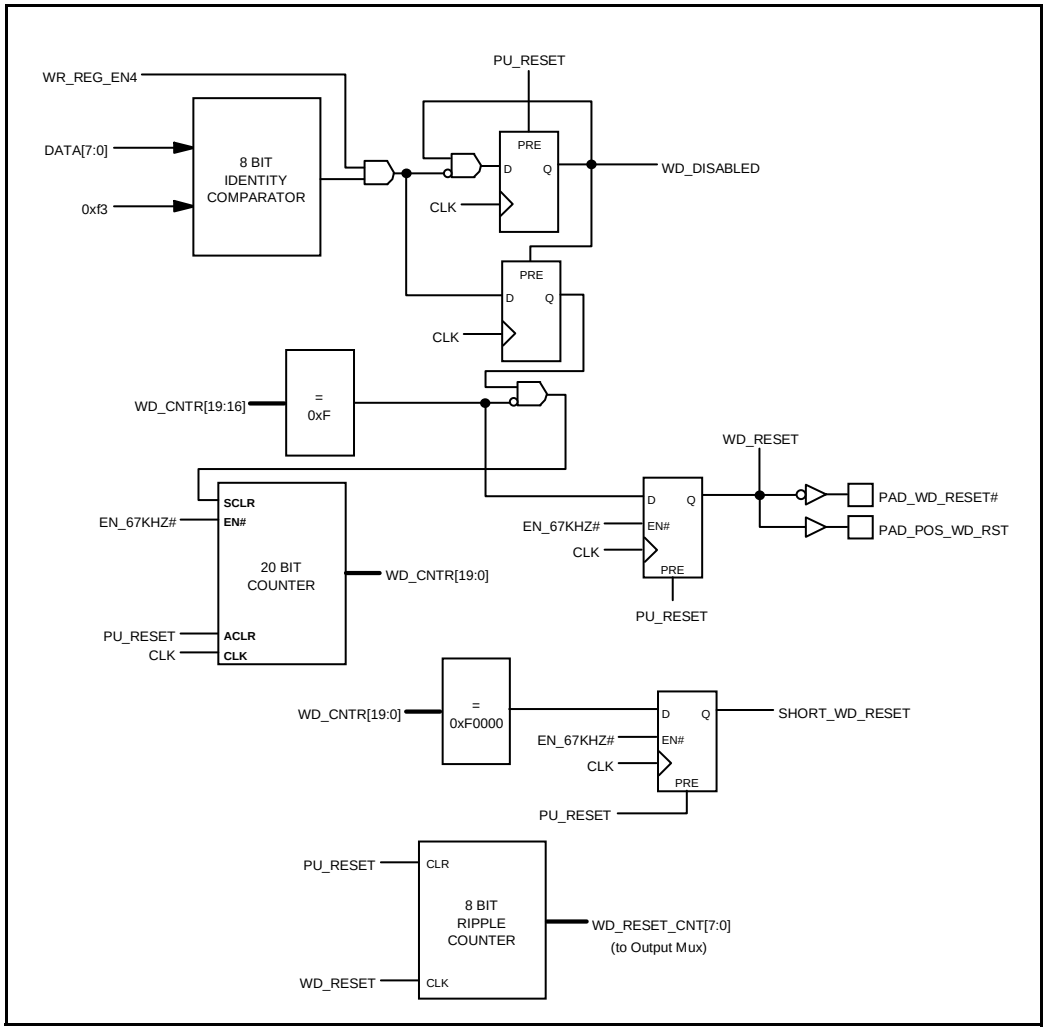


Figure 41. Watchdog Timer Circuit Block Diagram

A Watchdog Timer circuit is useful in embedded systems to ensure that the processor does not “hang,” disabling the system until the system is reset (usually by cycling power). The processor can hang for a number of reasons. For example, a software bug could result in an infinite loop being executed, an error in a DRAM location could result in the wrong code being executed, a stack overflow could occur, etc. When these types of errors occur in a personal computer environment, the user acts as the “watchdog timer” by resetting the PC when the system has not responded within a reasonable period of time.

A watchdog timer can be implemented in several ways. The following describes the watchdog timer implementation in the reference design.

The Watchdog Timer contains a 20-bit counter that is incremented at a rate of 67 KHz (66666.6 Hz). If this counter reaches a value of 0xf0000, two reset signals are generated: WD_RESET and SHORT_WD_RESET. The difference in these two signals is the time the signals are asserted. WD_RESET is asserted for 65536 clock cycles (of the 67 KHz signal) for a pulse width of 980 ms. SHORT_WD_RESET is asserted for only one clock cycle for a pulse width of 15 µs. The counter is reset whenever a value of 0xf3 is written to the Watchdog Timer register (an 8-bit register). Following a “power up” reset, the Watchdog Timer is disabled until the Watchdog Timer register is updated. Disabling the Watchdog Timer following power up reset is useful during code development prior to the Watchdog Timer code being implemented or in tracing down certain bugs (e.g., the Watchdog Timer is being updated too slowly). The Watchdog Timer is enabled when the circuit is first updated by writing 0xf3 to the Watchdog Timer register (clearing the WD_DISABLED flip-flop). Once the circuit is enabled, it can be disabled only by a power up reset.

If the Watchdog Timer is enabled, Watchdog Timer updates must be performed every 14.7 seconds or less. In other words, if two updates are separated by more than 14.7 seconds, a Watchdog Timer reset is generated. The system software would normally have a single task that updates the Watchdog Timer. This task would be responsible for monitoring other tasks to ensure that they are being executed and the hardware is properly configured. For example, if the task that updates the Watchdog Timer is the only task executing, the Watchdog Timer would never time-out even though the system is not functioning properly. Therefore, the task that updates the Watchdog Timer must also act as a “software” watchdog timer.

The Watchdog Timer circuit contains an 8-bit ripple counter (the Watchdog Timer Reset Counter) that indicates the number of Watchdog resets that have occurred since the last power up reset. This counter is cleared by power up reset and is incremented (i.e., clocked) by WD_RESET. The processor can read the value of this counter for use as diagnostic information. The processor can determine the value of the WD_DISABLED signal by reading from the “VPP ENABLE/WD STATUS” register. Bit 7 of this register returns the value of WD_DISABLED. Following a reset, the processor can read this register to determine the cause of the reset. If bit 7 (WD_DISABLED) is a 1, a power up reset was the cause of the reset. If bit 7 is a 0, a Watchdog Timer reset caused the processor to be reset.

The Watchdog Reset Counter will wrap to 0 if it reaches a value of 255. Thus, a watchdog reset counter value of 0 cannot be used to determine the cause of a processor reset. A non-zero value indicates that a processor reset was caused by a Watchdog Timer reset.

Sheets 14 and 15 of the DRAM Controller FPGA Schematic show the Watchdog Timer circuit implementation. Sheet 1 of the Schematic shows the output pads. Note that the PAD_WD_RESET# and PAD_POS_WD_RST signals become PROC_RESET# and POS_PROC_RESET, respectively, on the printed wiring board.

3.11.6 VPP Enable Design

The DRAM Controller FPGA contains a circuit for generating a signal that turns on and off the VPP power supply to the Flash memory. The VPP power supply is the +5 V supply routed through a P-Channel FET switch. When the FET’s gate signal (VPP_EN#) is low, the FET is turned on and +5 V is applied to the VPP pins of the Flash, permitting the Flash memory to be written. When VPP_EN# is high, the FET is off and VPP is ground. Sheet 3 of the Reference Design Schematic in Appendix A shows the VPP FET Switch circuit.

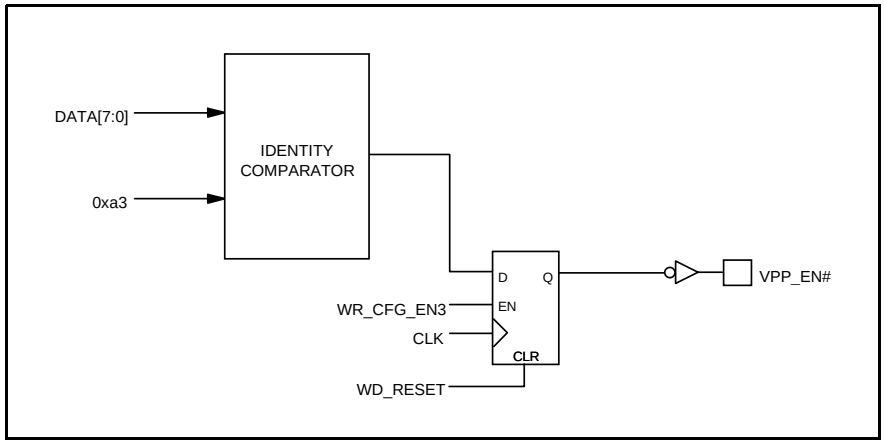


Figure 42. VPP Enable Circuit Block Diagram

The VPP_EN# signal is generated by the circuit whose block diagram is shown in Figure 42. When a value of 0xa3 is written to the VPP Enable register, VPP_EN# is set low. Writing any value other than 0xa3 to the VPP Enable register causes VPP_EN# to be set high. VPP_EN# is forced high by a Watchdog Timer reset (note that WD_RESET is also asserted during a power up reset).

Sheet 13 of the DRAM Controller FPGA Schematic shows the logic for the VPP Enable circuit. Sheet 1 of the schematic shows the output pad that inverts the output.

VPP_EN# is generated by a more complex circuit so that accidental enabling of VPP to the Flash memory is less likely. Thus, the Flash memory contents are less likely to be corrupted.

NOTE: The state of the Flip-Flop (VPP_EN) can be determined by reading from the VPP Enable Status register.

3.12 Misc Logic FPGA

3.12.1 Overview

The Misc Logic FPGA is one of two QuickLogic FPGA devices that provide logic for the processor section of the reference design. The Misc Logic FPGA device was originally intended to contain all of the logic except the DRAM Controller. However, due to pin limitations on the Misc Logic FPGA, some of the “miscellaneous” logic was incorporated into the DRAM Controller FPGA (see Section 3.11, DRAM Controller FPGA). The Misc Logic FPGA contains the following logic:

- I/O Control Signal logic
- Flash Memory Control Signal logic
- EPROM Control Signal logic
- DUART Control Signal logic
- RMON FIFO Read Control Signal logic
- 80960Cx/Hx BOFF# logic
- Bus Monitor
- ADLATCH_OE# logic
- JXPROC_ONCE# logic

Appendix E shows the Misc Logic FPGA Schematic.

Figure 43 shows the input and output signals from the Misc Logic FPGA.

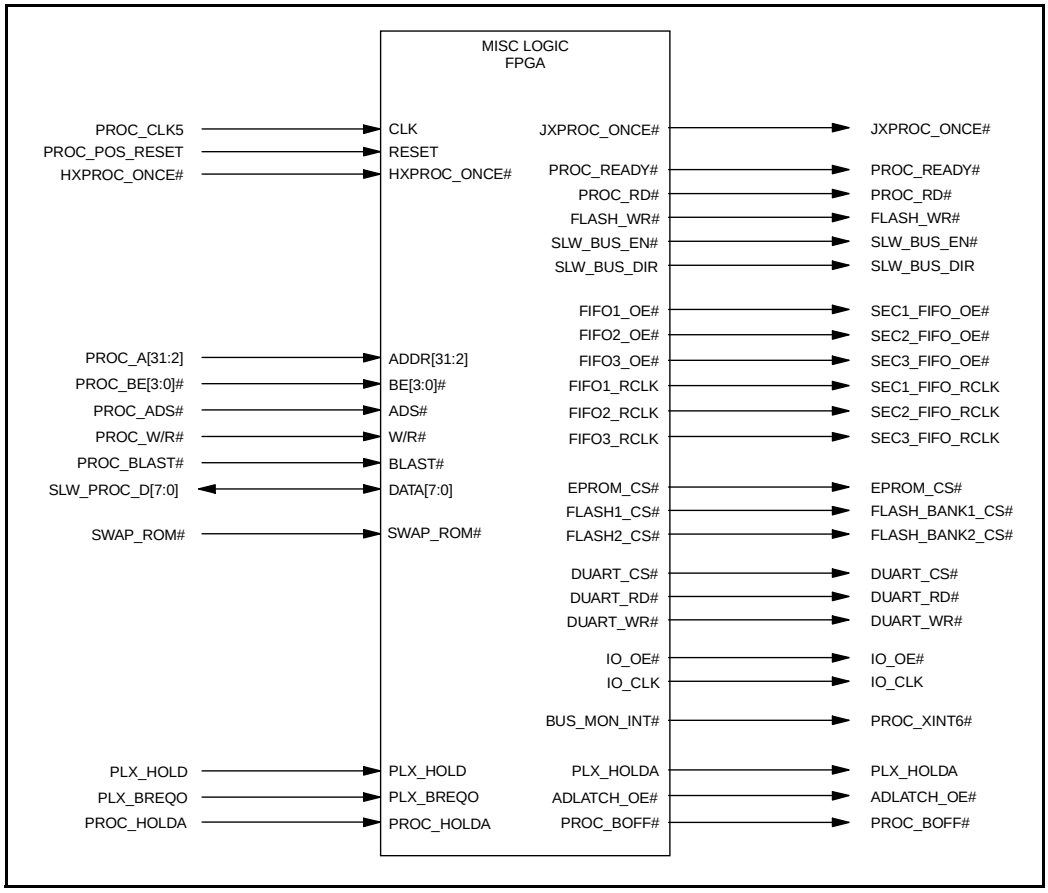


Figure 43. Misc Logic FPGA Signals

Figure 44 shows a block diagram of the Misc Logic FPGA. The Device Control block generates the control signals for the RMON FIFOs, DUART, EPROM, Flash memory and slow bus data transceiver. This block also controls the reading of data from and writing of data to the FPGA itself. The FPGA is connected as an 8-bit device to the processor's slow data bus (SLW_PROC_D[7:0]). Using only 8 data bits reduces the number of pins required by the FPGA and simplifies the board layout.

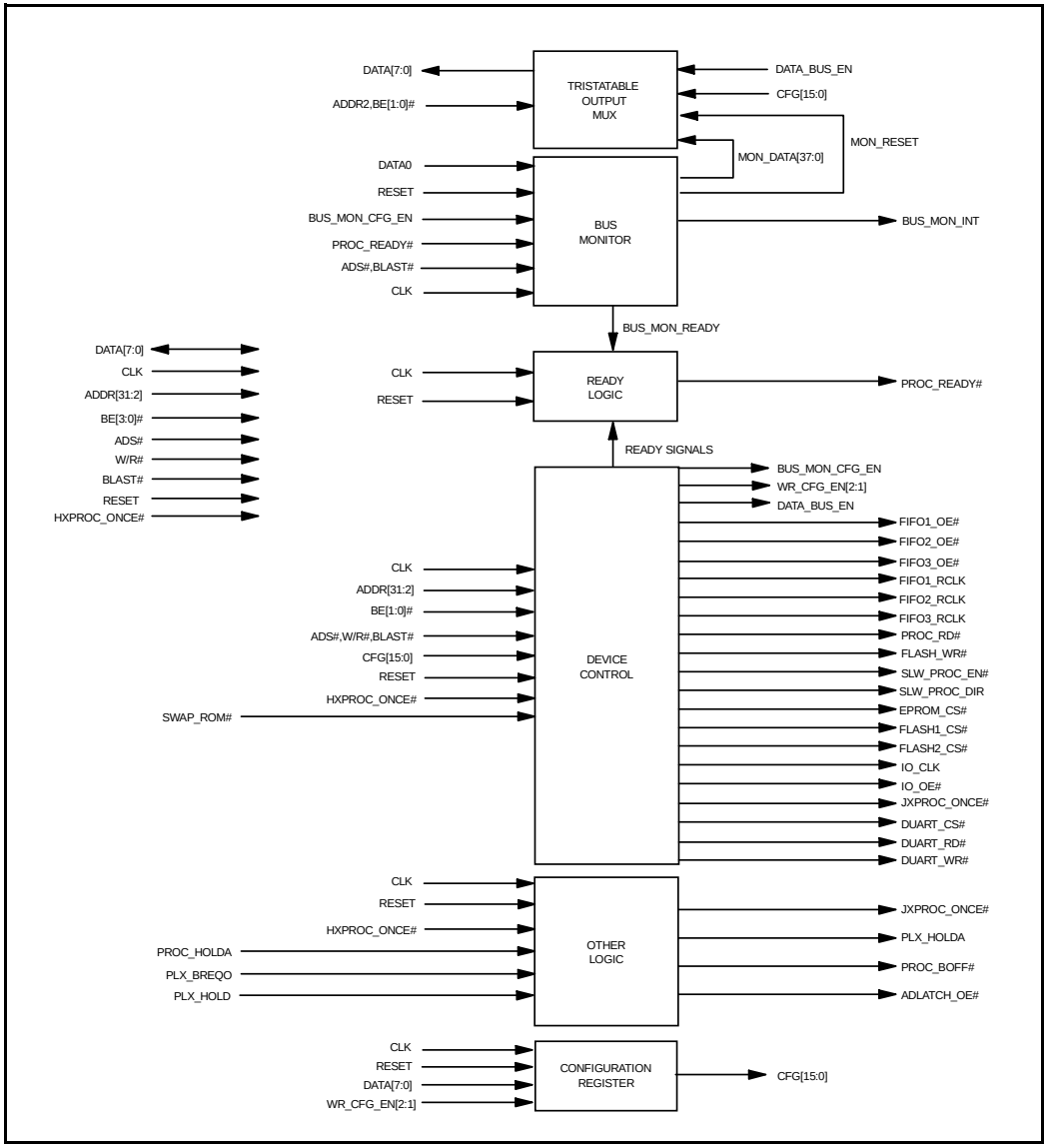


Figure 44. Misc Logic FPGA Block Diagram

3.12.2 Registers

Tables 39 and 40 show the Misc Logic FPGA register definitions.

Table 39. Misc Logic FPGA Write Registers

Primary Address	Description
4000 0100	**Configuration Register (Low Byte)**
4000 0101	**Configuration Register (High Byte)**
4000 0102	**Bus Monitor Reset Register** - Bit 0 of this register controls the Bus Monitor Circuit reset. Writing xxxxxxx1 to this register puts the Bus Monitor Circuit into reset. Writing xxxxxxx0 to this register removes the Bus Monitor Circuit from reset. Following a Bus Monitor interrupt, the Bus Monitor Circuit should be reset after the Bus Monitor data has been read to allow the Bus Monitor to collect data on the next non-responding access.

Table 40. Misc Logic FPGA Read Registers

Primary Address	Description
4000 0100	**Configuration Register (Low Byte)**
4000 0101	**Configuration Register (High Byte)**
4000 0102	**Bus Monitor Reset Status** - Reading from this location returns the value of the Bus Monitor Reset signal in bit 0 (1 = reset, 0 = enabled), bits 1 to 7 will be 0.
4000 0103	**Bus Monitor Data** - Reading from this location returns bits 7:0 of the Bus Monitor Data.
4000 0104	**Bus Monitor Data** - Reading from this location returns bits 15:8 of the Bus Monitor Data.
4000 0105	**Bus Monitor Data** - Reading from this location returns bits 23:16 of the Bus Monitor Data.
4000 0106	**Bus Monitor Data** - Reading from this location returns bits 31:24 of the Bus Monitor Data.
4000 0107	**Bus Monitor Data** - Reading from this location returns bits 37:32 of the Bus Monitor Data (note: the upper two bits read are 0).

Bus Monitor Data is collected by the Bus Monitor when a bus access is in progress but no device has responded with a READY# acknowledgment within 1000 clock cycles. When this timeout occurs, the Bus Monitor captures data related to the event (e.g., address) then generates the READY# acknowledgment, allowing the processor to continue. The Bus Monitor also generates an interrupt when it captures data. The Bus Monitor will not collect additional data until it has been reset, but will continue to generate READY# pulses for unacknowledged bus data. The Bus Monitor data consists of 37 bits:

- Bus Monitor Data [31:0]: These bits contain the value of the Address Bus when the Bus Monitor asserts READY# for an unacknowledged bus access. Note: bits 1:0 will always be 0.
- Bus Monitor Data [35:32]: These bits contain the value of the BE[3:0]# signals when the Bus Monitor asserts READY# for an unacknowledged bus access.
- Bus Monitor Data [36]: These bits contains the value of the W/R# signal when the Bus Monitor asserts READY# for an unacknowledged bus access.

- Bus Monitor Data [37]: These bits contains the value of the BLAST# signal when the Bus Monitor asserts READY# for an unacknowledged bus access.

The Configuration register is 2 bytes, with each byte written individually. However, software can treat the Configuration register as a 16-bit word and the processor will perform a burst access where the bytes are read or written consecutively. As shown in Tables 39 and 40, the value of the Configuration register can be read back. The Configuration register is used primarily for configuring the wait state profiles for devices in the system.

Notes:

1. **Following a reset, the Configuration register contents are 0xFFFF.** This provides the most conservative timing configuration possible, allowing the processor to access devices until they have been configured for a more optimum performance.

2. **Following a reset, the Bus Monitor circuit will be reset (i.e., disabled).**

Table 41. Misc Logic FPGA Configuration Register Bit Definitions

Configuration Register Bit(s)	Description
15:11	Unused
10:8	**DUART (AM85C30) Wait State Profile** - This parameter selects the number of wait states inserted for a DUART read or write access. Note that burst accesses to the DUART are not allowed. 000 - 5 100 - 9 001 - 6 101 - 10 010 - 7 110 - 11 011 - 8 111 - 12
7:6	**Wait State Profile for EPROM Reads** - This parameter specifies the wait state profile for EPROM read accesses. Note that EPROM writes are not possible. 00: 3,3,3,3 10: 5,5,5,5 01: 4,4,4,4 11: 6,6,6,6
5	**Wait State Profile for Flash Writes** - This parameter specifies the wait state profile for Flash memory write accesses. 0: 2,2,2,2 1: 3,3,3,3
4	**Insert Recovery State for Flash Reads (80960Jx Processor Only)** - This parameter specifies whether the READY# signal should be asserted for an extra clock following a Flash read access to allow additional time for the 80960Jx Address/Data bus to be "turned around." This may be necessary at high processor bus frequencies (25-33 MHz) due to the length of the Flash memory output disable time and the delay within the Misc Logic FPGA in outputting the Flash RD# control signal. 0: Recovery State **NOT** Inserted 1: Recovery State Inserted
3:0	**Wait State Profile for Flash Reads** - CFG[3:2] is NRDD, CFG[1:0] is NRAD. 0000: 1,1,1,1 1000: 1,3,3,3 0001: 2,1,1,1 1001: 2,3,3,3 0010: 3,1,1,1 1010: 3,3,3,3 0011: 4,1,1,1 1011: 4,3,3,3 0100: 1,2,2,2 1100: 1,4,4,4 0101: 2,2,2,2 1101: 2,4,4,4 0110: 3,2,2,2 1110: 3,4,4,4 0111: 4,2,2,2 1111: 4,4,4,4

The use of the configuration register bits is explained in more detail in the following sections, which describe individual blocks of the Misc Logic FPGA.

3.12.3 Slow Data Bus Transceiver Control Signals

The reference design contains a secondary data bus, the "Slow Processor Data Bus," which is isolated from the processor's data bus. The components on the slow data bus have slow access times or are accessed infrequently. Isolating these components reduces the capacitive loading on the processor data bus. The components are isolated using four 74ACT245 transceiver devices (see Sheet 26 of the Reference Design Schematic). These devices require two control signals: output enable (EN#) and direction (DIR). The EN# signal determines whether the transceiver outputs are enabled (EN# = 0) or tri-stated (EN# = 1). The DIR signal determines which bus signals are the inputs and which are the outputs. When DIR is a 1, the transceiver drives from the A side (pins 2-9) to the B side (pins 11-18). When DIR is a 0, the transceiver drives from the B side to the A side. The A side is the (fast) processor data bus and the B side is the slow processor data bus.

The transceiver control signals are generated by the Misc Logic FPGA and are called SLW_BUS_EN# and SLW_BUS_DIR. Normally, the transceiver is enabled and drives from the processor data bus to the slow data bus (A to B). This mode allows the slow data bus to be driven when accesses are made to devices on the processor data bus and when a device on the slow data bus is being written. Therefore, the only time the transceiver control signals need to be changed is when a read access is made to a device on the slow data bus.

Sheet 11 of the Misc Logic FPGA Schematic shows the transceiver control logic. When the logic determines that a read is occurring to a device on the slow data bus, the SLW_BUS_DIR signal is set low two clocks after the address cycle, as shown in Figure 45. This permits an 80960Jx processor to stop driving the address bus before the transceiver starts driving the bus.

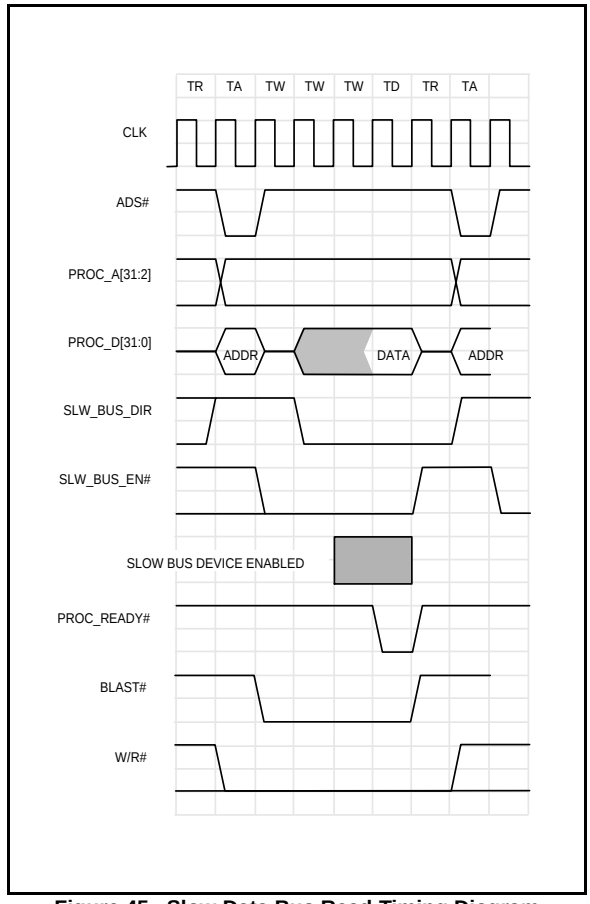


Figure 45. Slow Data Bus Read Timing Diagram

The control signals that enable the output drivers onto the slow data bus must wait until the transceiver has stopped driving the slow data bus (i.e., it has been turned around) before being asserted. These signals are controlled by the Misc Logic FPGA or DRAM Controller FPGA and enforce this requirement by waiting for three clock cycles after the address cycle before enabling a device onto the slow data bus.

When the slow data bus read access ends (BLAST# = 0 and READY# = 0), the SLW_BUS_EN# signal is deasserted for two clock cycles, as shown in Figure 45. The SLW_BUS_DIR is then set to 1 one clock period after SLW_BUS_EN# is deasserted. Thus, when the processor starts another bus access following a slow data bus read, the transceiver is tri-stated during the address cycle. When the address cycle is completed, the transceiver has been returned to its normal configuration (driving from the processor data bus to the slow data bus). Note that the only time the transceiver is disabled is during the two clock cycles after a slow data bus read access ends.

Timing Analysis

The critical timing parameter for the transceiver is the time it takes for the transceiver outputs to be disabled following the IO_OE# being deasserted. The transceiver output should be disabled before the processor starts driving the bus during the address cycle.

Table 42. Slow Data Bus Transceiver Timing Analysis

Parameter	Value	Units
Optimal Time (one clock cycle)	30	ns min
Clock Skew	2	ns max
Transceiver Output Disable Delay	14	ns max
IO_OE# Delay	13	ns min
Margin	1	ns

The margin is only 1 ns; however, the analysis did not take into account the output enable delay for the processor. In addition, the IO_OE# delay was calculated for a -0 speed grade of FPGA and not a -1 or -2 speed grade, which would add a few ns of margin. Therefore, the circuit should operate correctly at 33 MHz.

Notes:

1. The 80960Cx/Hx processors should be programmed to insert a recovery cycle for read accesses to the slow data bus.

2. The transceiver will turn around at the end of every slow data bus read access even when the processor performs back-to-back slow data bus reads. The transceiver is not turned around between reads within a burst access.

3.12.4 FPGA Interface Design

The FPGA interface design for the Misc Logic FPGA is basically the same as that previously described for the DRAM Controller FPGA. The main differences are that the Misc Logic FPGA has eight registers that can be read and three registers that can be written, as opposed to four read and write registers for the DRAM Controller FPGA.

The FPGA Interface logic is located on Sheets 2 (data bus transceiver), 9 (output data mux), and 11 (state machine).

Timing Analysis

The timing analysis for the Misc Logic FPGA is slightly different from the DRAM Controller FPGA because a slower speed grade device is used for the Misc Logic FPGA. The timing is also affected by the internal use of signals within the FPGA. Table 43 shows the FPGA Interface timing parameters for the Misc Logic FPGA as a function of the FPGA speed grade. The delays are almost identical for the Misc Logic FPGA and the DRAM Controller (-2 speed grade), with the exception of the Data input setup time, which is much shorter for the Misc Logic FPGA.

Table 43. Misc Logic FPGA Timing Parameters vs. FPGA Speed Grade

Parameter	-0 Speed Grade	-1 Speed Grade	-2 Speed Grade	Units
FPGA Output Enable Delay	23	20	19	ns max
FPGA Data Input Setup Time	7	6	6	ns min
FPGA Address Select Delay	25	22	20	ns max

Table 44 indicates the margin for the first FPGA read (24 ns) for a 3,2,2,2 wait state profile at 33 MHz using a -1 speed grade FPGA. Table 45 provides the same information for the second to fourth reads in a burst access (37 ns). These margins are slightly less than for the DRAM Controller FPGA. Note that the timing analysis does not account for extra bus loading or signal propagation delays on the printed wiring board.

Table 44. Misc Logic FPGA First Read Timing Analysis

Parameter	Value	Units
Optimal Time (two clock cycles)	60	ns min
Clock Skew	2	ns max
FPGA Output Enable Delay	20	ns max
Transceiver Propagation Delay	8	ns max
Processor Data Input Setup Time	6	ns min
Margin	24	ns

Table 45. Misc Logic FPGA Burst Read Timing Analysis

Parameter	Value	Units
Optimal Time (three clock cycles)	90	ns min
Clock Skew	2	ns max
Processor Address Output Delay	15	ns min
FPGA Address Select Delay	22	ns max
Transceiver Propagation Delay	8	ns max
Processor Data Input Setup Time	6	ns min
Margin	37	ns

Table 46 indicates the margin for the first FPGA write (55 ns) for a 2,1,1,1 wait state profile at 33 MHz using a -1 speed grade FPGA. Table 47 provides the same information for the second to fourth writes in a burst access (29 ns). These margins are higher than for the DRAM Controller FPGA due to the lower data setup time required for the Misc Logic FPGA. Note that the timing analysis does not account for extra bus loading or signal propagation delays on the printed wiring board.

Table 46. Misc Logic FPGA First Write Timing Analysis

Parameter	Value	Units
Optimal Time (three clock cycles)	90	ns min

Table 46. Misc Logic FPGA First Write Timing Analysis

Clock Skew	2	ns max
IO_OE# Delay (from Misc Logic FPGA)	13	ns min
Transceiver Output Disable Delay	14	ns max
FPGA Data Input Setup Time	6	ns min
Margin	55	ns

Table 47. Misc Logic FPGA Burst Write Timing Analysis

Parameter	Value	Units
Optimal Time (two clock cycles)	60	ns min
Clock Skew	2	ns max
Processor Data/Address Output Delay	15	ns max
Transceiver Propagation Delay	8	ns max
FPGA Data Input Setup Time	6	ns min
Total	29	ns

3.12.5 I/O Port Control Signals

The reference design contains a 32-bit output port and a 24-bit input port (which looks like a 32-bit port with eight undefined bits). The Misc Logic FPGA generates the control signals for these I/O ports.

The output port requires a clock, IO_CLK, which loads the current value of the data bus into the registers used to implement the port (U45-U48 on Sheet 25 of the Reference Design Schematic). Figure 46 shows the timing for the IO_CLK signal when the output port is written.

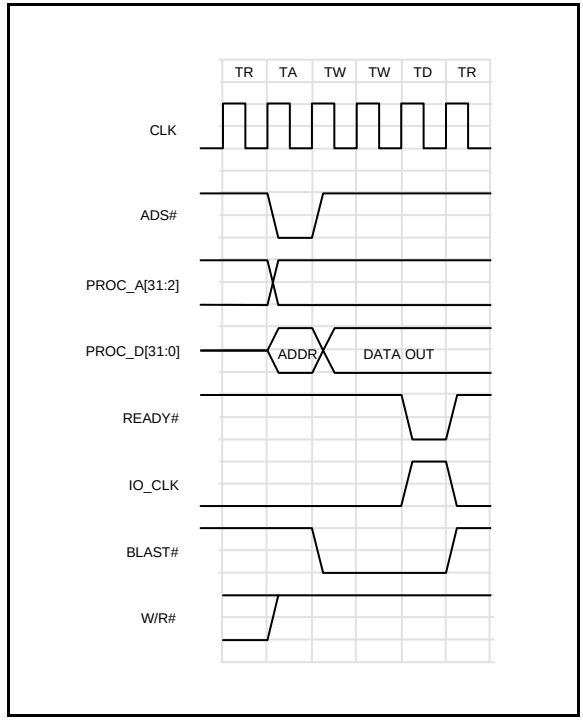


Figure 46. Output Port Write Timing Diagram

Sheet 3 of the Misc Logic FPGA Schematic shows the logic that generates IO_CLK. This logic is very simple; when a write to the output port is being performed, the IO_CLK and READY# signals are asserted three clock cycles after ADS# is asserted.

Notes:

1. The output port is always written as a 32-bit word. In other words, it is not possible to write individual bytes of the output port.
2. The Misc Logic FPGA does not support burst accesses to the output port.
3. The output port timing (i.e., wait states) is fixed at two inserted wait states.

The timing for the output port is worst case at a processor clock frequency of 33 MHz. Timing analysis is performed at this frequency. Table 48 shows the timing analysis. The resulting margin, which is in excess of one clock cycle, indicates that one of the wait states could be removed.

Note that the IO_CLK delay *increases* the margin.

Table 48. IO_CLK Timing Analysis

Parameter	Value	Units
Optimal Time (two clock cycles)	60	ns min
Clock Skew	2	ns max
Processor Data Out Delay	15	ns max
Transceiver Propagation Delay	8	ns max
74ACT574 Setup Time	2	ns min
IO_CLK Delay	3	ns min
Margin	36	ns

The input port requires an output enable signal, IO_OE#, which enables the input port signals onto the slow data bus. The value then propagates through the slow data bus transceiver to the processor data bus. Figure 47 shows the timing for the IO_OE# signal when the input port is read.

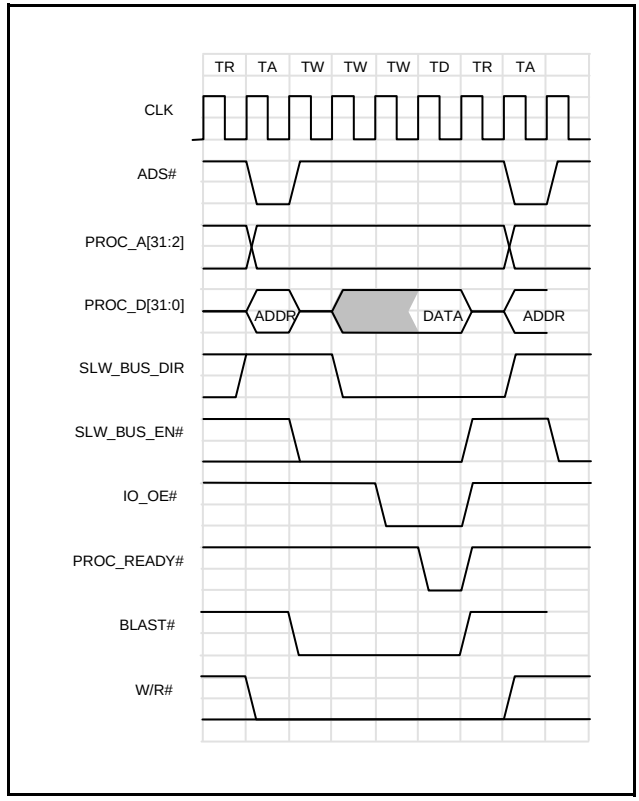


Figure 47. Input Port Read Timing Diagram

Sheet 3 of the Misc Logic FPGA Schematic shows the logic that generates IO_OE#. This logic is very simple: when a write to the output port is being performed, the IO_CLK and READY# signals are asserted four clock cycles after ADS# is asserted and the IO_OE# signal is asserted for two clock cycles three clock cycles after ADS# is asserted.

Notes:

1. The Misc Logic FPGA does not support burst accesses to the input port.

2. The input port timing (i.e., wait states) is fixed at three inserted wait states.

3. The SLW_BUS_DIR and SLW_BUS_EN# signals, which control the slow bus transceiver, are discussed in another section of the Misc Logic FPGA description.

The timing for the input port is worst case at a processor clock frequency of 33 MHz. Timing Analysis is performed at this frequency. The timing analysis indicates that three wait states is appropriate for operation at 33 MHz.

Table 49. IO_OE# Timing Analysis

Parameter	Value	Units
Optimal Time (two clock cycles)	60	ns min
Clock Skew	2	ns max
IO_OE# Delay	13	ns max
74ACT573 Output Enable Delay	11	ns max
Transceiver Propagation Delay	8	ns max
Processor Input Data Setup Time	6	ns max
margin	20	ns

3.12.6 Dual UART (DUART) Control Signals

The Misc Logic FPGA provides the control logic to the DUART (AM85C30), which is an 8-bit device on the slow data bus. The DUART requires three control signals provided by the Misc Logic FPGA:

DUART_CS#
DUART_RD#
DUART_WR#

The DUART has a bus interface timing requirement (T_{RC} - Valid Access Timing Recovery) that requires device accesses to be separated by a minimum time, which is specified as 3.5 PCLK cycles. Thus, it is desirable to use as fast a clock as possible for the PCLK input of the 85C30. The maximum PCLK frequency is 8.192 MHz for an 85C30-8 (speed grades up to 20 MHz are available). A PCLK frequency of 8 MHz is derived from the 16 MHz clock output from the AV9155A-23 clock synthesizer. The DRAM Controller FPGA contains a divide-by-two circuit for generating PCLK.

An 8 MHz clock for PCLK results in T_{RC} having a minimum value of 437.5 µs. The processor could attempt to perform accesses to the DUART that violate this timing requirement. Note that the timing requirement is between accesses to the DUART. The processor can access another device on the bus immediately after accessing the DUART. Software could prevent this timing requirement from being violated by ensuring that there is always a sufficient number of instructions between DUART accesses. However, care must be taken since the timing between accesses is influenced by a number of factors, such as execution from cache, queuing of bus requests, processor clock rate, etc. Therefore, software enforcement of the T_{RC} timing requirement would be difficult to implement and verify under all conditions. For these reasons, this timing requirement is enforced in hardware.

NOTE: T_{RC} is measured between the falling edges of the DUART RD# and WR# control signals.

The method used to ensure T_{RC} is satisfied is to make every access to the DUART take T_{RC} minimum; that is, additional wait states are inserted such that if the next bus access were

to the DUART, the bus accesses would be separated by T_{RC} minimum.

The DUART control logic allows the number of wait states for read and write accesses to be specified as a configuration option. Bits 8, 9, and 10 of the Configuration Register select the number of wait states, which can be five to twelve. Figures 48 and 49 show the DUART signal timing for read and write accesses. The only difference in the timing for the different wait state profiles is the width of the DUART_RD# and DUART_WR# pulses. The pulse widths of these signals is the number of wait states minus 2. Note that the timing for the DUART_RD# and DUART_WR# pulses is identical. This simplifies the task of determining the minimum value of T_{RC}, which is (in clock cycles) the number of wait states plus 3 (which are the TA, TR, and TD states). Note that the skew of the RD# and WR# signals (the difference in delay from the clock signal) needs to be considered in calculating T_{RC}.

NOTE: The logic does not support burst accesses to the DUART.

Sheet 12 of the Misc Logic FPGA Schematic shows the logic that generates the DUART control signals. Sheet 1 of the Schematic shows the output pads.

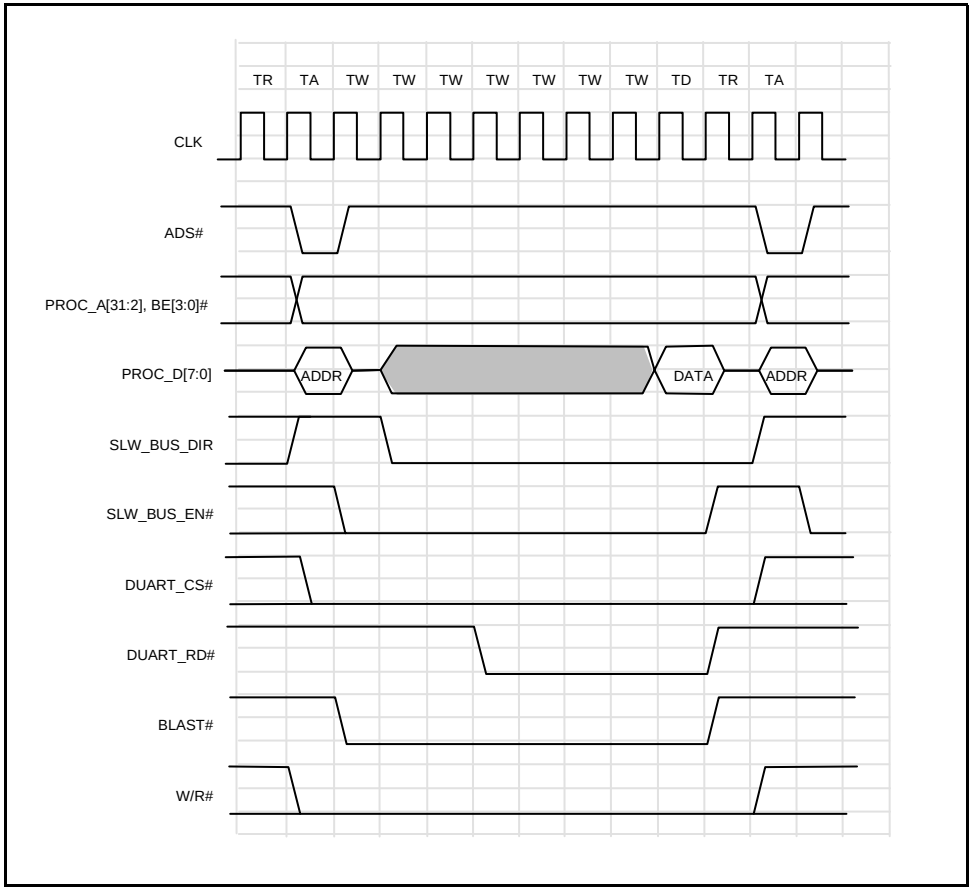


Figure 48. DUART Read Timing Example (Seven Wait States)

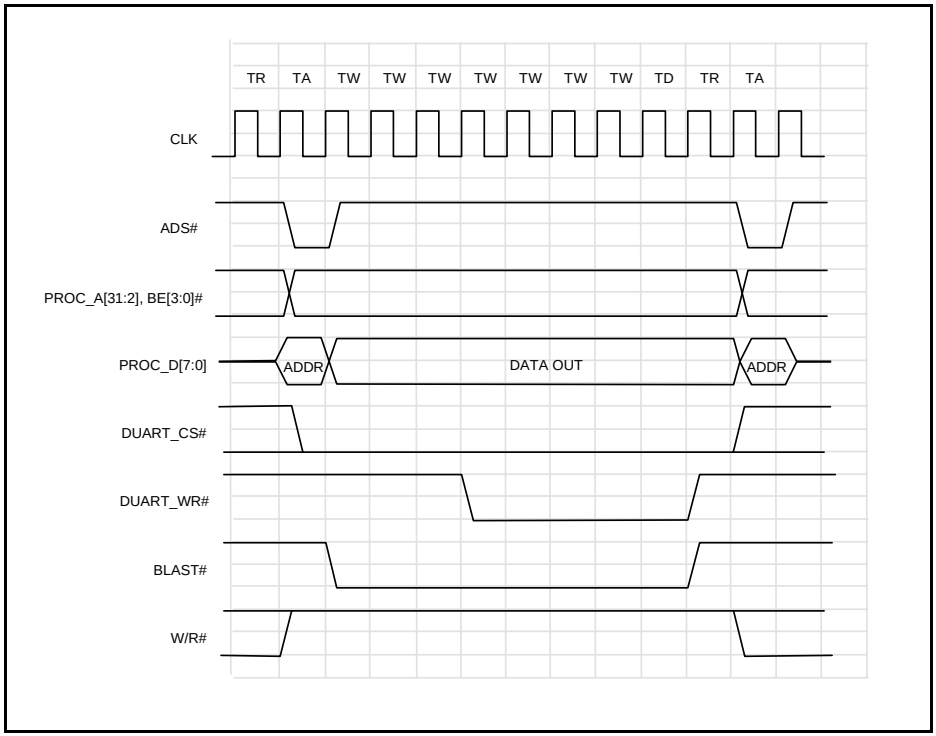


Figure 49. DUART Write Timing Example (Seven Wait States)

Timing Analysis

Table 50 shows the DUART timing parameters that must be considered in analyzing the timing for the Misc Logic FPGA.

Table 50. Important DUART Timing Parameters

Parameter	Value	Units
Address to RD#↓ or WR#↓ Setup Time (T_{AS})	70	ns min
Address to RD#↓ or WR#↓ Hold Time (T_{AH})	0	ns min
WR# Low Width (T_{WP})	150	ns min
RD# Low Width (T_{RP})	150	ns min
RD#↓ to Valid Data Delay (T_{ACC})	140	ns max

Table 51 shows the timing delays of the DUART control signals generated by the Misc Logic FPGA. As shown in this table, the delay for the DUART RD# and WR# signals is essentially the same; therefore the skew between the two signals is relatively small.

Table 51. DUART Parameters vs. FPGA Speed Grade

Parameter	-0 Speed Grade	-1 Speed Grade	-2 Speed Grade	Units
DUART_CS# Delay	29	25	23	ns max
DUART_RD# Delay	13	11	10	ns min
DUART_WR# Delay	13	11	10	ns max

The number of DUART timing parameters that need to be considered can be reduced since T_{AH} is always satisfied by the basic timing. In addition, T_{RP} can be ignored since the data access time (T_{ACC}) will always be longer. The minimum for data to be valid is 140 ns, but the RD# pulse needs to be longer to allow the data to propagate through the slow bus transceiver. The resulting RD# pulse will thus be longer than T_{RP}, allowing T_{RP} to be ignored.

The timing analysis for T_{AS} needs to be performed only for a clock frequency of 33 MHz, since the number of clock cycles is fixed between the address cycle (ADS# asserted) and the clock cycle in which DUART RD# or WR# is asserted. Note that the DUART address consists of PROC_BE[1:0]# and therefore does not propagate through an address latch for the 80960Jx processor. Table 52 shows the timing analysis for T_{AS} and indicates that there is plenty of margin. Note that the timing analysis did not account for the minimum delay of WR# and RD# from the Misc Logic FPGA (which would add to the margin). Timing analysis also did not account for signal propagation on the printed wiring board or capacitive loading higher than the manufacturer's datasheet specification.

Table 52. DUART T_{AS} Timing Analysis

Parameter	Value	Units
Optimal Timing (Four Clocks)	120	ns
Clock Skew	2	ns max
Processor Address Output Delay	15	ns max
Address to RD#↓ or WR#↓ Setup Time (T_{AS})	70	ns min
Margin	33	ns

Table 53 shows the minimum low pulse width of the DUART RD# as determined from the timing analysis. The result of 166 ns removes the need to analyze the timing for the minimum WR# low pulse width, since 166 ns is more than the specified value of 150 ns, and the minimum low pulse widths of WR# and RD# are the same for a programmed wait state profile.

Table 53. DUART Minimum RD# Pulse Width Timing Analysis

Parameter	Value	Units
Clock Skew	2	ns max
RD#↓ to Valid Data Delay (T_{ACC})	140	ns max
Transceiver Propagation Delay	8	ns max
Processor Data Input Setup	6	ns min
Total	166	ns

Table 54 shows the minimum number of wait states recommended for the possible processor frequencies and the resulting margins for T_{RC} and T_{ACC}. Note that at high frequencies, the number of wait states is driven by the T_{RC} parameter, while T_{ACC} determines the number of wait states at lower processor frequencies.

Table 54. DUART Wait State Profiles and Margins vs. Processor Frequency

Frequency (MHz)	Clock Cycle (ns)	Wait States	T_{RC} Margin (ns)	T_{ACC} Margin (ns)
16	66	5	90	32
20	50	6	12	34
25	40	9	42	114
30	33	11	24	131
33	30	12	12	134

3.12.7 EPROM Control Signals

The reference design contains a 128Kx8 EPROM that can be used to hold the boot code during development. This EPROM may also contain a monitor to aid in debugging. The wiring on the printed wiring board allows either a 256Kx8 or 512Kx8 EPROM to be installed in the same socket if desired. (PROC_A[18:17] are connected to the EPROM DIP, although they are not used by the 128Kx8 EPROM.) Sheet 27 of the Reference Design Schematic shows the EPROM. The EPROM is located on the slow data bus because it is a relatively slow device and would normally only be used during development; therefore the number of clock cycles needed to access the EPROM is not critical.

The EPROM speed is 150 ns, which means that the access time from a stable address and chip enable is 150 ns. The access time from the output enable being asserted to valid data is 65 ns.

The EPROM requires two control signals: Output Enable (OE#) and Chip Enable (CE#). The EPROM's CE# pin is connected to PROC_RD# from the Misc Logic FPGA. The EPROM's CE# pin is connected to EPROM_CS# from the Misc Logic FPGA (chip enable and chip select are equivalent terms). The Misc Logic FPGA provides four possible wait state profiles, which are selected by bits 6 and 7 of the Configuration Register. The possible wait state profiles are 3,3,3,3 to 6,6,6,6. Figure 50 shows the EPROM signal timing for a single byte read with a 6,6,6,6 wait state profile.

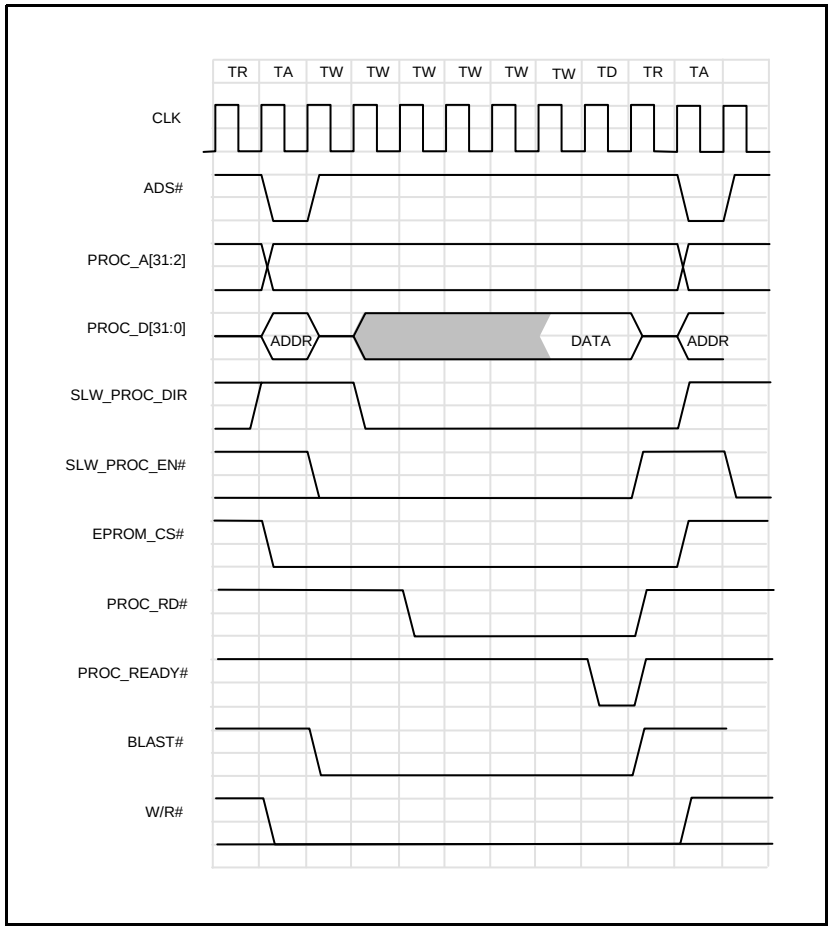


Figure 50. EPROM Signal Timing for Single Read (6,6,6,6 Wait State Profile)

Figure 51 shows the signal timing for a 4 byte burst read from the EPROM with a 3,3,3,3 wait state profile. The EPROM_CS# is decoded from the processor address and is generated whenever the processor address is in the EPROM address range. Sheet 4 of the Misc Logic FPGA Schematic shows the EPROM chip select decoding logic. Sheet 2 of the Schematic shows the output pad for the EPROM_CS#.

The chip select decoding logic uses the upper six bits of the processor address and is also dependent on the value of the SWAP_ROM# input. When SWAP_ROM# is a 1 (deasserted), EPROM_CS# is asserted when PROC_A[31:26] is 111111. When SWAP_ROM# is a 0 (asserted), EPROM_CS# is asserted when PROC_A[31:26] is 110111. The purpose of the SWAP_ROM# signal is to allow the Flash memory and EPROM address spaces to be swapped, allowing the Flash devices to contain the boot up code (allowing the EPROM to be removed if desired).

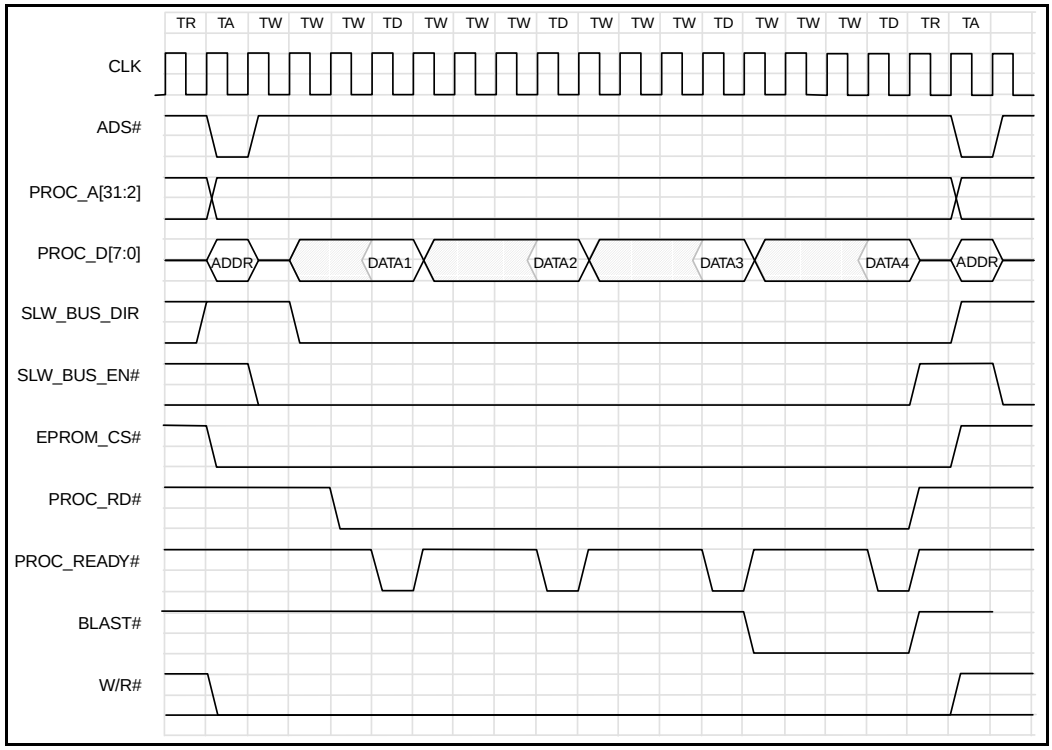


Figure 51. EPROM Signal Timing for Burst Read (3,3,3,3 Wait State Profile)

The PROC_RD# signal is asserted whenever an EPROM or Flash memory read access occurs. PROC_RD# is used as the OE# signal for both the EPROM and the Flash devices in the reference design. The PROC_RD# timing for EPROM read accesses is shown in Figures 50 and 51, where PROC_RD# is asserted one clock cycle after SLW_BUS_DIR goes low and is deasserted in the clock cycle after BLAST# and READY# are both asserted. The PROC_RD# timing is slightly different for Flash read accesses (refer to Section 3.12.8, Flash Control Signals).

Sheet 7 of the Misc Logic FPGA Schematic shows the logic for determining the number of wait states in an EPROM read access. Sheet 6 of the Schematic shows the logic for generating PROC_RD#.

Timing Analysis

The timing analysis needs to look at three scenarios regarding the EPROM. For the first byte read from the EPROM during a read access, the number of wait states required can depend on either the OE# to valid data delay or the address access time delay (related to EPROM_CS#). The second to fourth EPROM bytes read during a burst read access are not dependent on the EPROM control signals, because these signals remain asserted from the first byte read. Therefore, the delay for these accesses is dependent only on the address access timing (the delay from processor address changing to valid EPROM data at the processor).

The first consideration is the delay of the EPROM control signal generation within in the Misc Logic FPGA, which is summarized in Table 55 for various FPGA speed grades.

Table 55. EPROM Control Signal Delays vs. FPGA Speed Grade

Parameter	-0 Speed Grade	-1 Speed Grade	-2 Speed Grade	Units
PROC_RD# Delay	13	12	11	ns max
EPROM_CS# Delay	27	23	22	ns max

Table 56 shows the timing analysis for the first EPROM byte read for the delay controlled by the timing of the EPROM's output enable signal (PROC_RD#). The number of clock cycles available for data to be valid is the number of wait states minus 1. For example, when the wait state profile is 6,6,6,6, there are five clock cycles (150 ns at 33 MHz) in the EPROM timing to compensate for this delay.

Table 56. First EPROM Read Access Delay (Output Enable Controlled)

Parameter	Value	Units
Clock Skew	2	ns max
PROC_RD# Delay	12	ns max
EPROM Output Enable Delay	65	ns max
Transceiver Propagation Delay	8	ns max
Processor Input Data Setup Time	6	ns min
Total	93	ns

Table 57 shows the timing analysis for the first EPROM byte read for the delay controlled by the timing of the EPROM's chip select signal (EPROM_CS#). The number of clock cycles available for data to be valid is the number of wait states plus 2. For example, when the wait state profile is 6,6,6,6, there are eight clock cycles (240 ns at 33 MHz) in the EPROM timing to compensate for this delay.

Table 57. First EPROM Read Access Delay (Chip Select Controlled)

Parameter	Value	Units
Clock Skew	2	ns max
Processor Address Output Delay	15	ns max
Address Latch Delay	8	ns max
EPROM_CS# Delay	23	ns max
EPROM Address Access Delay	150	ns max
Transceiver Propagation Delay	8	ns max
Processor Input Data Setup Time	6	ns min
Total	212	ns

Table 58 shows the timing for the second, third, and fourth EPROM byte reads of a burst access for the delay controlled by the EPROM's address access timing. The number of clock cycles available for data to be valid is the number of wait states plus 1. For example, when the wait state profile is 6,6,6,6, there are seven clock cycles (210 ns at 33 MHz) in the EPROM timing to compensate for this delay.

Note that the address latch delay does not have to be considered in this case because the PROC_A[3:2] are routed directly from the processor.

Table 58. Second-Fourth EPROM Read Access Delay

Parameter	Value	Units
Clock Skew	2	ns max
Processor Address Output Delay	15	ns max
EPROM Address Access Delay	150	ns max
Transceiver Propagation Delay	8	ns max
Processor Input Data Setup Time	6	ns min
Total	181	ns

Table 59 shows the margins for the relevant delay parameters as a function of processor frequency. The margins are calculated for the wait state profile that provides the smallest positive margin.

Table 59. EPROM Wait State Profiles and Margins vs. Processor Frequency

Frequency (MHz)	Clock Cycle (ns)	Wait State Profile	1st Byte-OE Margin (ns)	1st Byte-CS Margin (ns)	2nd-4th Byte Margin (ns)
16	66	3,3,3,3	39	118	83
20	50	3,3,3,3	7	38	19
25	40	4,4,4,4	27	28	19
30	33	5,5,5,5	39	19	17
33	30	6,6,6,6	57	28	29

NOTES:

1. The delay calculations were performed using the -1 speed grade timings.
2. The timing analysis did not include delays that could result from signal propagation on the printed wiring board or capacitive loading of address and data buses above the value specified in the manufacturer's datasheets.

3.12.8 Flash Control Signals

Sheet 28 of the Reference Design Schematic shows the locations for four Flash memory devices (U6 to U9). The intent is that locations U6 and U7 would be populated with U8 and U9 available for expansion. The devices U6 and U7 comprise “Flash Bank 1,” and the devices U8 and U9 comprise “Flash Bank 2.” The Flash devices, 28F400CVT80, are from Intel’s SmartVoltage boot block Flash memory family and are in a 48-lead TSOP (Thin Small Outline Package). This package was chosen because the 128Kx16 and 512Kx16 devices in the SmartVoltage boot block family are also available in this package. Thus, the amount of Flash memory installed in the system can be scaled up or down using other devices in the SmartVoltage boot block family.

Boot Block devices were chosen because they allow a portion of the device (the “boot block”) to be write protected while the rest of the device is modified. The boot block can contain the boot up code that is normally located within an EPROM, resulting in reduced system cost by eliminating the EPROM device. The boot block architecture includes “parameter” blocks to facilitate storage of frequently updated parameters that would normally require an EEPROM. The boot block devices are available in two types: Top Boot and Bottom Boot. The type refers to the location of the boot block (at the lowest addresses or at the highest addresses). Top Boot devices are used for the reference device because the 80960Jx/Cx/Hx processor families boot from the top of memory where the initialization boot record (IBR) is located.

Figure 52 shows the memory map for the Flash memory blocks (when 256Kx16 devices are used).

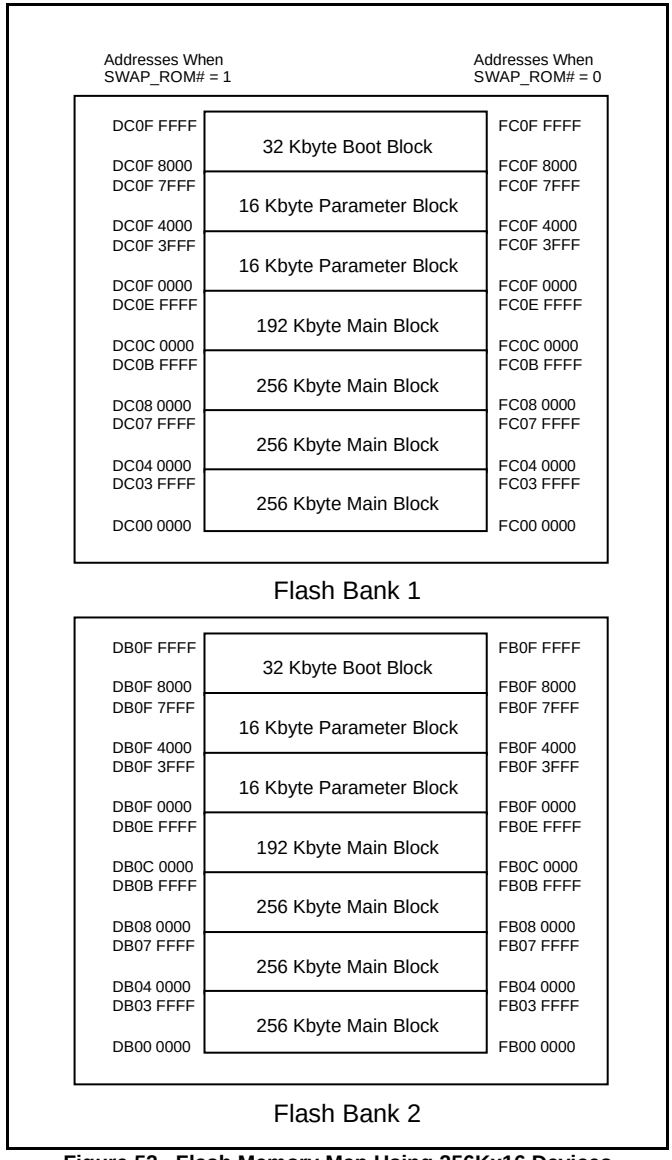


Figure 52. Flash Memory Map Using 256Kx16 Devices

The memory map for the Flash memory is dependent on the value of the SWAP_ROM# signal, which is determined by switch 4 of the DIP switch (see Section 3.7, Memory Map). This permits the EPROM and Flash address spaces to be swapped, allowing the boot device (EPROM or Flash) to be selected.

From the memory map, it would appear that the Flash memory could not contain the IBR for the processor when the Flash memory is used as the boot device (SWAP_ROM# = 0). The IBR is located at FEFF FF30 for the 80960Jx and 80960Hx processors and at FFFF FF00 for an 80960Cx processor. The IBR can be located in the Flash memory because the Flash memory address chip select decoding uses only the upper six address bits. Therefore, when the Flash memory map is defined as in Figure 52, the IBR addresses shown in Table 60 should be used.

The difference in IBR addresses is because the memory map in Figure 52 places the two Flash memory banks contiguously in memory. Flash Bank 1 is placed at the bottom of its memory space and bank 2 is placed at the top of its memory space. If it is known that only Flash Bank 1 will be used (or it is acceptable to have non-contiguous Flash memory), the memory maps in Figure 53 could be used, placing Flash Bank 1 at the top of its memory address space.

Table 60. IBR Addresses Due to Flash Address Aliasing

Processor	IBR Address
80960Cx	FC0F FF00
80960Jx	FC0F FF30
80960Hx	FC0F FF30

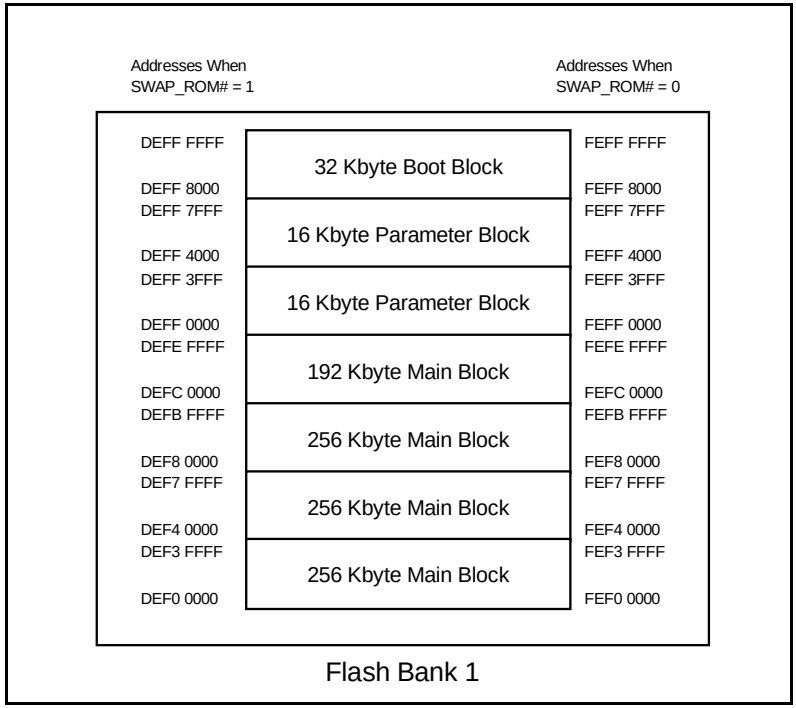


Figure 53. Alternate Flash Memory Map For One Flash Bank of 256Kx32

Notes:

1. In the normal configuration, U6 and U7 are installed with 256Kx16 devices, resulting in 256Kx32 (1 Mbyte) of Flash memory being available.

2. A parameter block can also be used for code storage if desired.

3. The devices in Flash Bank 2 have their WP# pins connected to VPP, allowing the boot block to be programmed in the same manner as the main and parameter blocks. This differs from Flash Bank 1, which connects WP# to VPP or ground through a 3-pin header that allows the boot block to be write protected by tying WP# to ground.

The Flash devices are located on the "fast" processor data bus to provide the fastest read access times possible. The Misc Logic FPGA provides four control signals for the Flash memory devices (signal names on Reference Design Schematic):

FLASH_BANK1_CS#
FLASH_BANK2_CS#
PROC_RD#
FLASH_WR#

The CS# signals are decoded from the processor address using combinatorial logic as shown on Sheet 4 of the Misc Logic FPGA Schematic. Sheet 2 of the Schematic shows the output pads for the CS# signals. The PROC_RD# signal is also connected to the OE# of the EPROM and is asserted for both EPROM and Flash memory read accesses. The FLASH_WR# signal is asserted for Flash memory write accesses.

The Misc Logic FPGA allows 16 possible wait state profiles to be selected for Flash memory reads and two wait state profiles for Flash memory writes (wait profiles are given in Table 41). The possible read access wait state profiles range from 1,1,1,1 to 4,4,4,4 and the write access wait state profiles are 2,2,2,2 and 3,3,3,3.

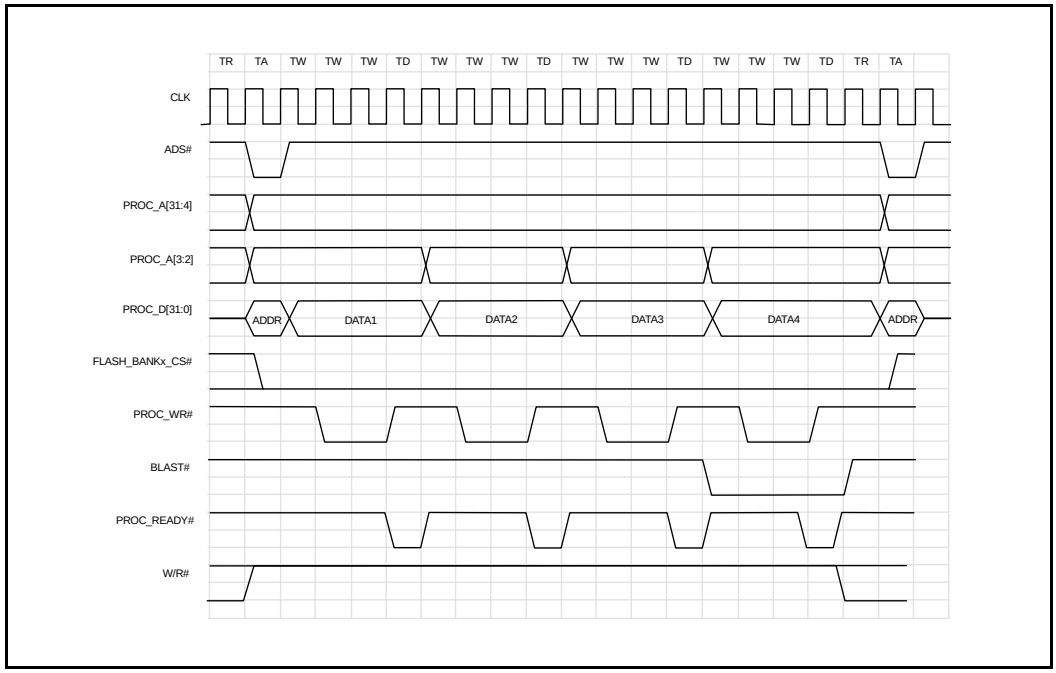


Figure 54. Flash Burst Read Timing for 3,3,3,3 Wait State Profile

Figure 54 shows the signal timing for a four-word burst read from the Flash memory with a 3,3,3,3 wait state profile. The PROC_RD# signal is asserted one clock cycle after the address cycle and deasserted the clock cycle after the burst access ends (BLAST#=0 and READY#=0).

A potential bus contention problem exists at higher processor frequencies. This bus contention occurs if the Flash memory output is not disabled before the processor drives the address for the next cycle. While this scenario is for an 80960Jx processor, the same problem could exist for an 80960Cx/Hx processor that performs a write access after reading from Flash memory. In both cases, a single clock cycle exists for the data bus to be tri-stated after a Flash memory read. The delay of the PROC_RD# signal plus the Flash output disable time determines if there is sufficient time to tri-state the data bus. The output disable time for the -80 speed Flash device is 30 ns max, which is one clock cycle at 33 MHz. The 80960Cx/Hx processors allow an extra clock cycle to be inserted before an address cycle that follows a bus access. The extra bus cycle can be inserted by programming the 80960Cx MCON or the 80960Hx PMCON register associated with the Flash memory region. The 80960Jx processor does not support this feature internally but does allow extra “recovery” bus cycles to be inserted using the READY# signal. A recovery cycle is inserted by an 80960Jx processor following each clock cycle in which READY# is asserted at end of bus access. Normally, this is one cycle, since READY# is asserted for only one clock cycle. However, the Misc Logic FPGA allows READY# to be asserted for an extra clock cycle when a Flash memory read access is performed and an 80960Jx processor is being used (HXPROC_ONCE# = 0). This option is selected by bit 4 of the Configuration Register (1 = add recovery state, 0 = normal timing).

Figure 55 shows the burst read timing example when the extra recovery state is inserted. The timing analysis description provides information on when the extra recovery state is required.

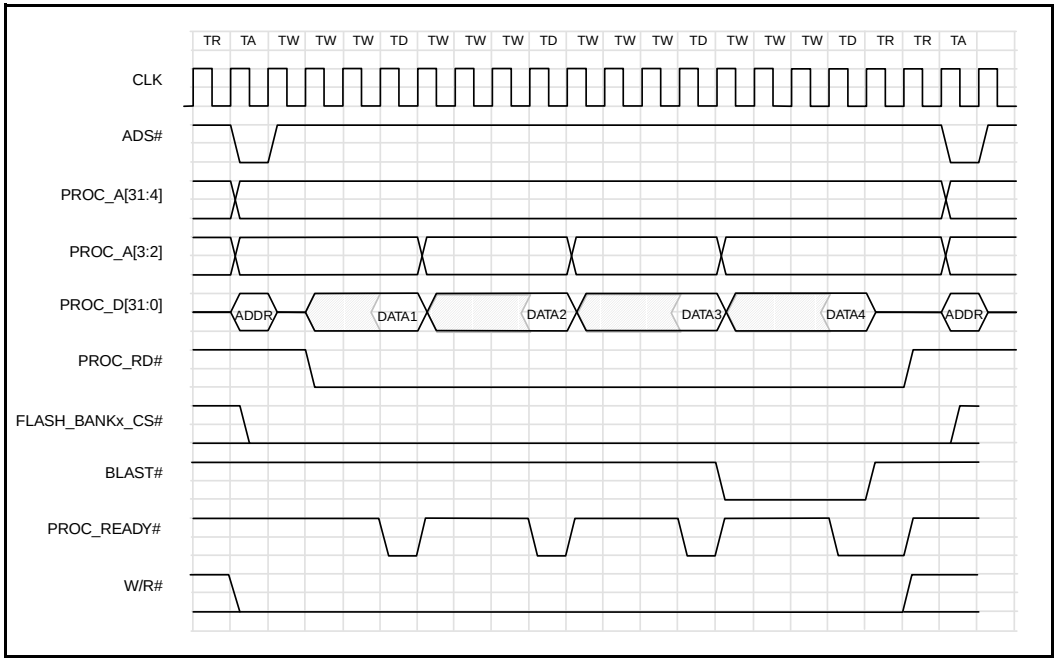


Figure 55. Flash Burst Read Timing Example With Extra Recovery Cycle

Figures 56 and 57 show the timing for burst writes to Flash memory. Only two wait state profiles are possible for Flash memory writes: 2,2,2,2 and 3,3,3,3. A limited number of wait state profiles is provided because Flash memory is written infrequently and therefore the timing does not affect performance as much as Flash read performance could. The key difference between the two wait state profiles (besides the additional wait state) is the width of the FLASH_WR# pulses. The pulse width for a 2,2,2,2 profile is one clock cycle, whereas the pulse width for a 3,3,3,3 profile is two clock cycles.

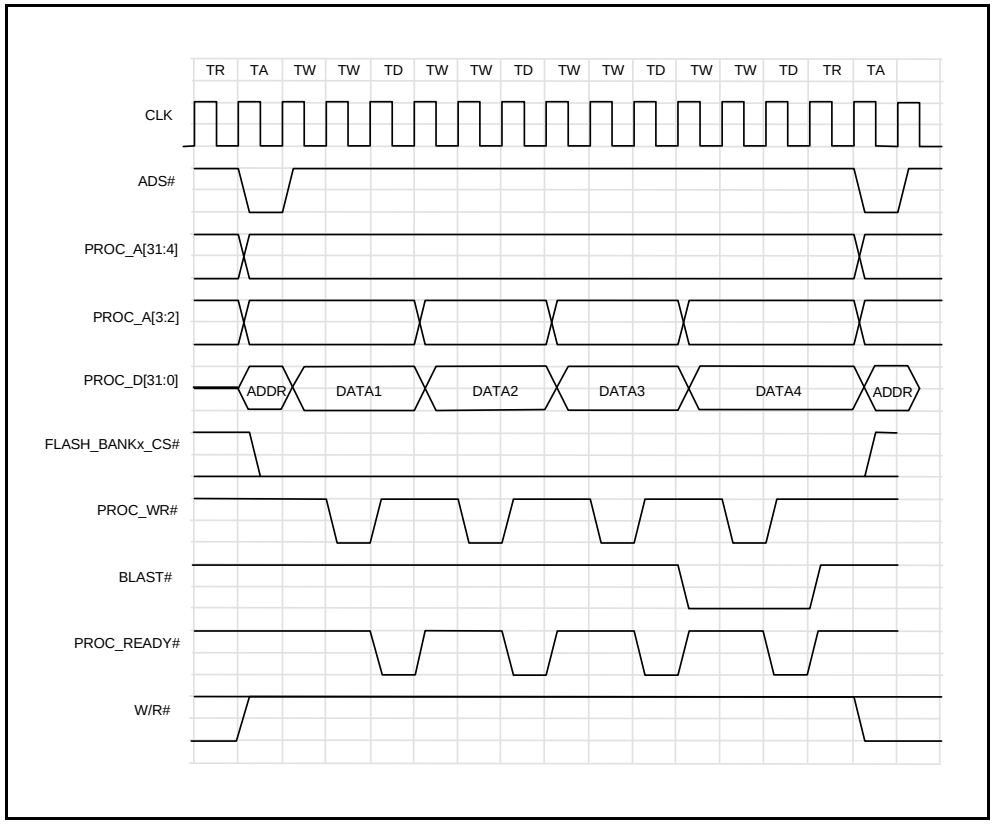


Figure 56. Flash Burst Write Timing for 2,2,2,2 Wait State Profile

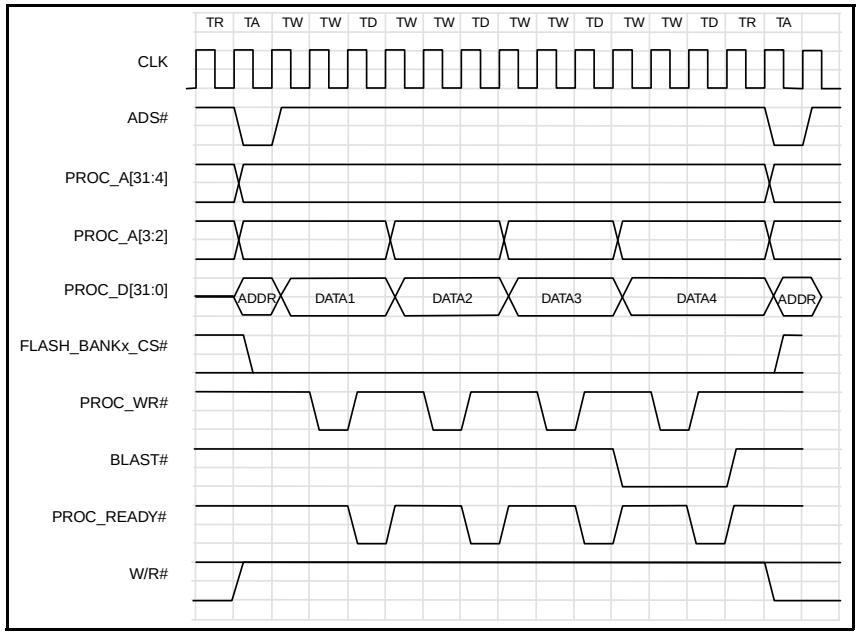


Figure 57. Flash Burst Write Timing for 3,3,3,3 Wait State Profile

Sheet 6 of the Misc Logic FPGA Schematic shows the logic that generates the Flash memory read timing. Sheet 7 of the Schematic shows the logic that generates the Flash memory write timing and Sheet 1 shows the output pads and READY# logic.

Timing Analysis

Table 61 shows the delays of the Flash memory control signals generated by the Misc Logic FPGA as a function of the FPGA speed grade.

Table 61. FPGA Flash Timing Parameters vs. FPGA Speed Grade

Parameter	-0 Speed Grade	-1 Speed Grade	-2 Speed Grade	Units
FLASH_CS# Delay	19	17	16	ns max
FLASH_WR# Delay	13	12	11	ns max
PROC_RD# Delay	13	12	11	ns max

The first Flash read timing may be dependent on the address access time or the output enable delay. Tables 62 and 63 show the delays that occur for the first Flash word read. There are NRAD+2 cycles to satisfy the delay controlled by the chip select (NRAD is the number of wait states between the address and first data cycles of a read access). There are NRAD cycles to satisfy the delay controlled by the output enable.

Table 62. First Flash Read Access Delay (Chip Select Controlled)

Parameter	Value	Units
Clock Skew	2	ns max
Processor Address Output Delay	15	ns max
Address Latch Delay	8	ns max
FLASH_CS# Delay	17	ns max
Flash Address Access Delay	80	ns max
Transceiver Propagation Delay	8	ns max
Processor Input Data Setup Time	6	ns min
Total	136	ns

Table 63. First Flash Read Access Delay (Output Enable Controlled)

Parameter	Value	Units
Clock Skew	2	ns max
PROC_RD# Delay	12	ns max
Flash Output Enable Delay	40	ns max
Transceiver Propagation Delay	8	ns max
Processor Input Data Setup Time	6	ns min
Total	68	ns

Table 64 shows the timing analysis for the second, third, and fourth Flash reads of a burst access when the delay is controlled by the Flash device's address access timing. The number of clock cycles available for data to be valid is the number of wait states between data accesses (NRDD) plus 1. For example, when the wait state profile is 4,3,3,3 (NRAD = 4, NRDD = 3) there are four (NRDD+1) clock cycles (120 ns at 33 MHz) in the Flash timing to compensate for this delay.

Note that the address latch delay does not have to be considered in this case because the PROC_A[3:2] are routed directly from the processor.

Table 64. Second-Fourth Flash Read Access Delay

Parameter	Value	Units
Clock Skew	2	ns max
Processor Address Output Delay	15	ns max
Flash Address Access Delay	80	ns max
Processor Input Data Setup Time	6	ns min
Total	103	ns

Table 65 shows the margins for the relevant delay parameters as a function of processor frequency. The margins are calculated for the wait state profile that provides the smallest positive margin.

Table 65. Flash Read Wait State Profiles and Margins vs. Processor Frequency

Frequency (MHz)	Clock Cycle (ns)	Wait State Profile	1st Byte-OE Margin (ns)	1st Byte-CS Margin (ns)	2nd-4th Byte Margin (ns)
16	66	2,1,1,1	64	128	29
20	50	2,2,2,2	32	64	47
25	40	2,2,2,2	12	24	17
30	33	3,3,3,3	31	29	29
33	30	3,3,3,3	22	14	17

The last timing parameter to consider for Flash reads is the output disable delay, to determine if an additional recovery state needs to be inserted. Table 66 shows the worst case delay for disabling the Flash outputs when a Flash read access ends. Note that the processor output enable delay reduces the delay. An extra recovery cycle must be inserted when the clock period is less than the output disable delay since there is one clock period (the recovery cycle) available in the timing to satisfy this requirement. **Therefore, an extra recovery cycle is needed at 30 and 33 MHz and is not needed at 16 and 20 MHz. The margin at 25 MHz is -1 ns, but since this is a worst case analysis, an extra recovery cycle should not be necessary.**

Table 66. Flash Output Disable Delay

Parameter	Value	Units
Clock Skew	2	ns max
PROC_RD# Delay	12	ns max
Flash Output Disable Delay	30	ns max
Processor Output Enable Delay	(3)	ns min
Total	41	ns

A detailed analysis of the Flash write timing is not provided. The relevant parameters are listed in Table 67. The "Data Setup Time before WR#↑" parameter cannot be met with a wait state profile of 2,2,2,2 at 30 and 33 MHz. All of the timing parameters for a wait state profile of 2,2,2,2 are easily satisfied at 25 MHz and below. Therefore, for Flash memory writes, use a wait state profile of 2,2,2,2 for 16, 20, and 25 MHz and a wait state profile of 3,3,3,3 at 30 and 33 MHz.

Table 67. Relevant Flash Write Timing Parameters

Parameter	Device	Value
Processor Address Output Delay	80960	15 ns max
Address Latch Delay	80960	15 ns max
Data Output Delay	80960	15 ns max
FLASH_CS# Delay	Misc Logic FPGA	17 ns max
CLK↑ to WR# Asserted	Misc Logic FPGA	12 ns max
CLK↑ to WR# Deasserted	Misc Logic FPGA	12 ns max
Write Cycle Time	28F400	80 ns max
Address Setup Time before WR#↑	28F400	80 ns max
Data Setup Time before WR#↑	28F400	50 ns min
Data Hold Time from WR#↑	28F400	0 ns min
CS# Setup Time to WR#↓	28F400	0 ns min
WR# Pulse Width	28F400	30 ns min

NOTES:

1. The timing analysis was performed using values for the -80 Flash speed grade and -1 FPGA speed grade.

3.12.9 RMON FIFO Control Signals

A 1Kx32 FIFO exists between each GT-48001 device and the slow data bus. Data is written into the FIFO that indicates the size of a packet and the source and destination addresses for a packet. The FIFO contents can be read by the processor. Three status signals for each FIFO are connected to the input port and indicate when the FIFO is empty (EF#), half-full (HF#), or full (FF#).

Sheets 10, 11, and 12 of the Reference Design Schematic show the FIFOs. Each FIFOs requires three control signals: OE#, REN#, and RCLK. The control signals for the FIFOs are generated by the Misc Logic FPGA. Because of pin limitations on the FPGA, the OE# and REN# signals on each FIFO are connected together, reducing to six the number of control signals required (two for each of the three FIFOs). These signals are as follows:

SEC1_FIFO_OE#
SEC1_FIFO_RCLK
SEC2_FIFO_OE#
SEC2_FIFO_RCLK
SEC3_FIFO_OE#
SEC3_FIFO_RCLK

A word is read from the FIFO on the rising edge of RCLK when REN# is low. The word is then held in a register whose contents can be enabled onto the slow data bus by asserting OE#. By tying OE# and REN# together, the FIFO RCLK must be asserted while FIFO output is enabled. Figure 58 shows the timing for a single word read from the FIFO. As shown, the FIFO OE# is asserted, and RCLK is asserted one clock later. There are then two clock periods for the FIFO to output the data and for the data to propagate through the slow data bus transceiver to the processor.

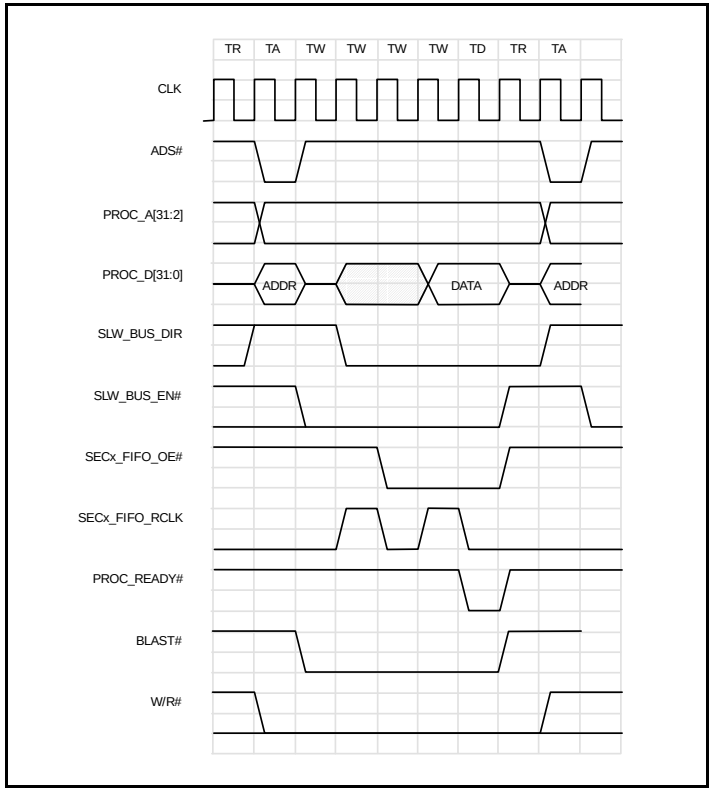


Figure 58. Signal Timing for Single FIFO Read

As shown in Figure 58, the FIFO read access has four wait states inserted. Note that the FIFO OE# cannot be asserted until the slow bus transceiver has been turned around (SLW_BUS_DIR = 0). The FIFO Read Control logic implementation allows burst reads from the FIFO using the timing shown in Figure 59. This wait state profile for burst reads is 4,1,1,1. Note that the FIFO address space is 1024 bytes; thus a burst access will work since consecutive addresses in this space continue to access the FIFO.

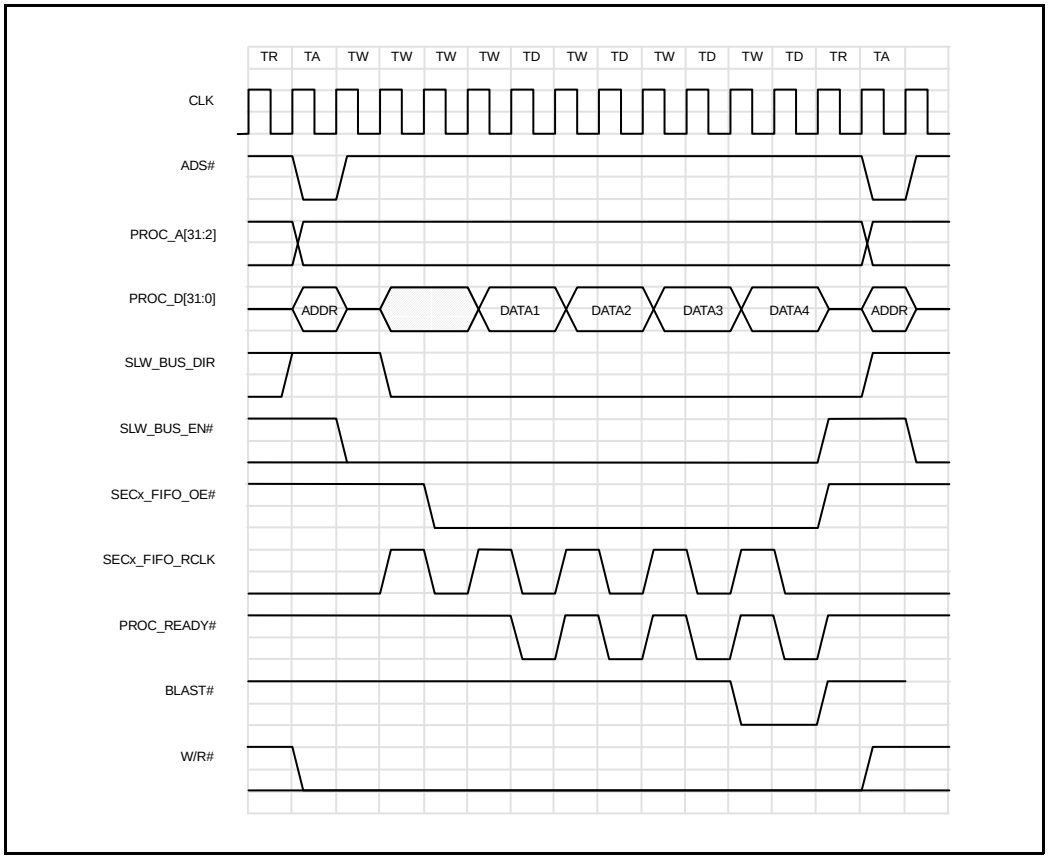


Figure 59. Signal Timing for Burst FIFO Read

Normally, the RMON FIFOs cannot be read using a burst access because it is not known how many words are in the FIFO. The processor would first have to check the FIFO's empty flag (EF#) and, if the FIFO is not empty, read a word, then repeat the process. The only scenario where a burst read could be used would be when the FIFO's half full flag (HF#) is asserted. The processor would then know that the FIFO has at least 512 words and could use burst accesses to reduce the delay associated with accessing the FIFO.

Sheet 5 of the Misc Logic FPGA Schematic shows the logic that generates the FIFO OE# and RCLK signals. Sheet 4 of the schematic shows the address decoding (source of FIFO1_SEL, FIFO2_SEL and FIFO3_SEL) and Sheet 1 shows the output pads.

Notes:

1. A FIFO RCLK pulse is generated for each of the RMON FIFOs two clock cycles after ADS# is asserted. This occurs whether the bus access is to an RMON FIFO or to another bus device. This RCLK pulse will have no affect on the FIFO contents since the FIFO's REN# input is not asserted on the positive edge of this RCLK pulse. The reason for generating this extra RCLK pulse is to update the FIFO's EF#, HF# and FF# status flags. These flags are updated only on the rising edge of RCLK. Without the extra RCLK pulse, the software would think the FIFO was always empty. This is because the software reads the FIFO only when the EF# flag is deasserted—but the EF# would not be deasserted until after the FIFO is read.
2. A single FIFO RCLK could be generated and routed to all RMON FIFOs. This was not done because of concerns over the potential signal integrity (i.e., transmission line reflections) of a clock signal routed to several sources. Separate signals allowed point-to-point connections that were series terminated with 22 Ω resistors (see Sheet 29 of the Reference Design Schematic).

Timing Analysis

The timing analysis for RMON FIFO read accesses is relatively simple because OE# is asserted early in the access and held for the entire access. Therefore, the critical timing path is the propagation delay for the FIFO data from the rising edge of the FIFO RCLK until the data is read by the processor. Table 68 shows the delays for the RMON FIFO control signals versus the FPGA speed grades.

Table 68. RMON FIFO Control Signal Delays vs. FPGA Speed Grade

Parameter	-0 Speed Grade	-1 Speed Grade	-2 Speed Grade	Units
SECx_FIFO_OE# Delay	18	15	14	ns max
SECx_FIFO_RCLK Delay	18	15	14	ns max

Table 69 shows the timing analysis for the RMON FIFO read accesses at 33 MHz. The delay calculations used the -1 FPGA speed grade timings and a 72225LB25 FIFO device. The analysis did not account for delays due to propagation of signals on the printed wiring board or higher bus capacitance than specified by manufacturers' datasheets. A slower speed device (72225LB35) could be used to reduce cost, but with a reduced margin (9 ns) at 33 MHz. An even slower device (72225LB50) could be used at processor clock frequencies of 30 MHz or less to further reduce costs. At 30 MHz, the margin would be 14 ns for the 72225LB50. Note that this analysis only considered the read side of the RMON FIFO interface. The write interface must also be considered when selecting the FIFO speed.

Table 69. FIFO Read Access Timing Analysis

Parameter	Value	Units
Optimal Time (two Clock Cycles)	60	ns
Clock Skew	2	ns max
SECx_FIFO_RCLK Delay	15	ns max
FIFO Data Out From RCLK↑ Delay	15	ns max
Transceiver Propagation Delay	8	ns max
Processor Input Data Setup Time	6	ns min
Margin	14	ns

3.12.10 Bus Monitor

The Misc Logic FPGA contains a circuit, the Bus Monitor, which monitors the processor bus for bus accesses that no device acknowledges. A bus access is acknowledged when a device asserts READY# for each read or write performed. The Bus Monitor starts a counter at the beginning of a processor data bus access (read or write). The counter is incremented on each clock cycle in which the access is not acknowledged. The counter is cleared when an access is acknowledged (READY# asserted). The counter is held in reset when the bus is not being used. In other words, the counter is enabled when ADS# is asserted and is disabled when READY# and BLAST# are asserted in the same clock cycle (bus access ended).

The processor will wait until READY# is asserted before continuing. Therefore, if the processor performs a bus access to an area of memory that is unused or an area of memory with a defective device, the processor will "hang," waiting for a READY# acknowledgment that never occurs.

If the counter reaches a value of 1000, the Bus Monitor "times out" and asserts READY#, allowing the processor to continue to the next bus access or read/write access within a burst access. For example, if the processor were performing a four-word burst access to an unused memory region, the Bus Monitor would assert READY# four times (once for each word accessed). There would also be a delay of 1000 cycles between each assertion of READY#. When the Bus Monitor asserts READY#, an interrupt is also asserted and the current value of the address (PROC_A[31:2]), BE[3:0]#, W/R#, and BLAST# are saved in a register. When data is loaded into the Bus Monitor register, additional READY# assertions by the Bus Monitor will not cause the Bus Monitor register to be loaded until the Bus Monitor has been reset. Thus, if multiple bus accesses are not acknowledged before the Bus Monitor is reset, the Bus Monitor register will contain the data from the first unacknowledged bus access.

Figure 60 shows a block diagram of the Bus Monitor circuit.

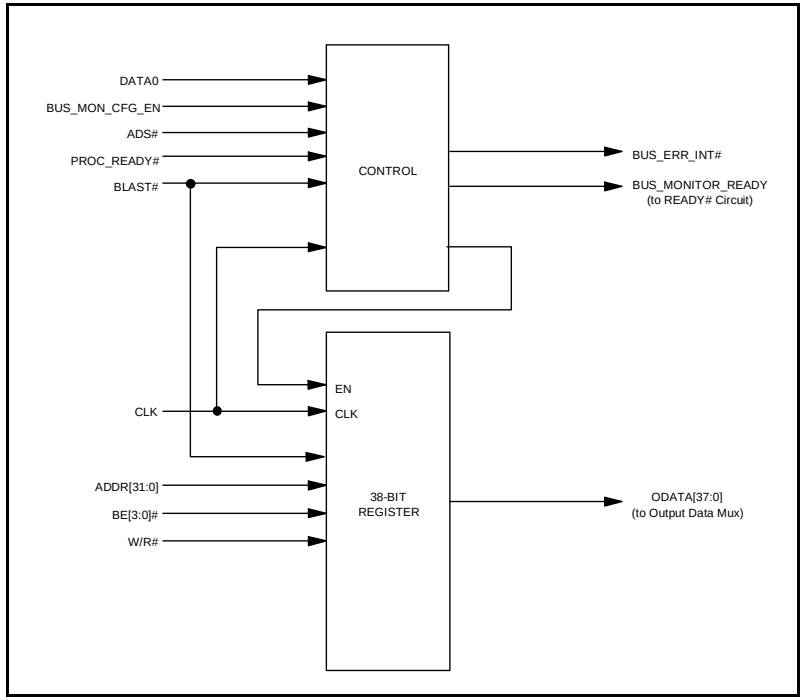


Figure 60. Bus Monitor Block Diagram

The interrupt generated by the Bus Monitor, BUS_ERR_INT#, is routed to XINT6# of the processor on the printed wiring board. If the Bus Monitor is used, the software would presumably contain an Interrupt Service Routine (ISR) that reads the Bus Monitor data and displays this information (part of a debugger or monitor). The ISR could also display the return address for the interrupt (presumably the source in the code where the error occurred). This information could be used in identifying misbehaving code (e.g., a bad pointer) or defective devices. Note that the defective device is the device that asserts READY# for an access. For example, if the Bus Monitor times out for an access to an RMON FIFO, the defective device is the Misc Logic FPGA, since it generates READY# for the RMON FIFOs.

The Bus Monitor contains a 1-bit register, the Bus Monitor Reset register, that resets the Bus Monitor circuit. This register must be written with a 1 to place the Bus Monitor into reset and clear the Bus Monitor interrupt. A 0 would then be written to the register to enable the Bus Monitor. Following reset, the Bus Monitor is disabled because a reset initializes this register to 1. The Bus Monitor will remain disabled until a 0 is written to the Bus Monitor Reset register. If the Bus Monitor is not used, the Bus Monitor should be disabled.

Sheet 10 of the Misc Logic FPGA Schematic shows the Bus Monitor circuit.

NOTE: **The Bus Monitor does not properly terminate an access to the PCI bus that is not acknowledged by the PCI9060.** This occurs because the PCI9060 drives the READY# line high (until the bus access is to be acknowledged) after it detects that an access is being made to the PCI9060 or the PCI bus. The possibility exists that some PCI bus accesses will not be acknowledged by the PCI9060 but will cause the PCI9060 to drive the READY# line high.

An example of this is when the PCI9060 attempts an access on the PCI bus but no device responds, resulting in a master abort. The first such master abort results in the processor bus access being acknowledged by the PCI9060 but with the Master Abort error flag set. If another PCI access is made while the Master Abort error flag is set, the PCI9060 will not acknowledge the processor bus access. The Bus Monitor will not be able to acknowledge the bus access because the PCI9060 is driving the READY# line high; therefore, the processor bus will "hang" unless the watchdog timer has been enabled.

Reference the "Master/Target Abort" section on page 15 of PLX Technology's 1995 product catalog, *PCI Bus Interface and Clock Distribution Chips*.

3.12.11 ADLATCH_OE# Logic

The Misc Logic FPGA contains a circuit for generating the output enable signal for the 80960Jx processor address latch (ADLATCH_OE#). The address latch output needs to be disabled under the following conditions:

- The 80960Jx Processor is disabled (i.e., the 80960Cx/Hx processor is enabled). The 80960Jx processor is disabled if HXPROC_ONCE# is deasserted.
- The processor has granted the bus to another device (PROC_HOLDA asserted).

Sheet 1 of the Misc Logic FPGA Schematic shows the logic for ADLATCH_OE#, which consists simply of an OR gate that is part of the output pad.

3.12.12 JXPROC_ONCE# Logic

The Misc Logic FPGA contains a circuit for generating the ONCE# signal for the 80960Jx processor. The ONCE# signal (On-Circuit Emulation) allows the processor to be disabled when it is in a circuit permitting an emulator to be used. The ONCE# signals are used in the reference design to enable only one of the two processors. The HXPROC_ONCE# signal is generated by a jumper that pulls the signal high or low and is connected to the 80960Cx/Hx processor and the Misc Logic FPGA. The 80960Jx processor ONCE# signal, JXPROC_ONCE#, must not be driven when deasserted because this pin becomes an output (LOCK) following reset. However, this pin can be driven low (asserting ONCE#) since the 80960Jx processor will not drive this pin if it is disabled.

The logic for JXPROC_ONCE# is located on Sheet 1 of the Misc Logic FPGA schematic and consists simply of a tri-statable buffer with the input grounded and HXPROC_ONCE# connected to the enable (i.e., an open-drain inverter).

Note: The use of the ONCE# signals allows both processors to be installed in the board, with one always disabled. It is not necessary to have both processors installed. The unused processor is disabled to prevent the possibility of both processors being enabled at the same time.

3.12.13 80960Cx/Hx BOFF# Logic

The Misc Logic FPGA contains a circuit for generating the BOFF# (Bus Backoff) signal for the 80960Cx and 80960Hx processors (the 80960Jx processors do not support BOFF#). This pin can be used to request that the processor suspend a bus access to allow another bus master to take control of the bus. When the processor regains control of the bus, it resumes the bus access that was suspended.

The Bus Backoff capability is used to resolve a deadlock condition that occurs when the processor attempts to access a device on the PCI bus and the same device attempts to access the processor's local bus. See **Section 3.3, PCI Bus and Arbiter**, for more information on the deadlock condition.

The PCI9060 asserts its BREQO output if (a) the processor is performing an access to the PCI bus, (b) a PCI device is trying to access the processor local bus and has not been granted access, and (c) the PCI9060 BREQO timer has expired. These conditions indicate a potential deadlock condition. Asserting the BREQO signal causes the Misc Logic FPGA to assert PROC_BOFF# (connected to BOFF# on the 80960Cx/Hx processor) if all of the following are true:

- The processor is currently performing a bus access. The BOFF# signal should not be asserted if the processor is not performing a bus access.
- The 80960Cx/Hx processor is enabled (HXPROC_ONCE# signal is deasserted). This prevents asserting BOFF# when an 80960Jx processor is being used. Since the Misc Logic FPGA asserts the PLX_HOLDA signal one clock cycle after asserting BOFF#, it is important not to assert BOFF# when an 80960Jx processor is being used.
- The PCI9060 is requesting the processor's local bus (PLX_HOLD is asserted).

If BOFF# is asserted, the PLX_HOLDA is asserted one clock cycle later to indicate to the PCI9060 that it can now access the processor local bus. This is necessary because the processor does not assert its HOLDA when it grants bus mastership as a result of BOFF# being asserted. Once the processor has asserted BOFF#, BOFF# will continue to be asserted until the PLX_HOLD signal is deasserted. This is necessary because the PCI9060 deasserts BREQO when it is granted the local bus.

Sheet 3 of the Misc Logic FPGA Schematic shows the BOFF# logic. Sheet 1 shows the logic for generating the PLX_HOLDA signal.

3.13 Jumper Definitions

The design contains several jumpers that allow the power up configuration to be defined (e.g., the amount of DRAM connected to the GT-48001) or that define basic circuit operation (e.g., 80960Jx processor or 80960Cx/Hx processor enabled). The following is a list of the jumpers and a description of the settings. These jumpers are created using three terminal headers and must have a jumper installed in one of the two possible positions. The positions are defined as 1-2 (installed across pins 1 and 2) or 2-3 (installed across pins 2 and 3). For consistency, all jumpers have pin 1 oriented towards the front of the PCB and pin 3 oriented towards the back of the PCB. The LEDs are located on the front of the PCB and the RJ45 ports (J1-J3) are located on the back.

Table 70. Reference Design Jumper Definitions

Jumper	Description
P1	Enables/Disables the processor circuitry. 1-2 Position: Allows the processor circuitry to be reset normally. 2-3 Position: Places the processor circuitry (including the PCI9060) into a constant reset. In effect, this position creates an unmanaged hub.
P2	Specifies type of DRAM connected to U5. 1-2 Position: EDO. 2-3 Position: Fast Page Mode.
P3	Specifies amount of DRAM connected to U3, U4, and U5. 1-2 Position: 1 Mbyte. 2-3 Position: 2 Mbyte.
P4-P8	Specifies device number for U3. P8 is the most significant bit and P4 is the least significant bit. The value of each is determined from the jumper position as follows: 1-2 Position: 1 2-3 Position: 0
P9-P13	Specifies device number for U4. P13 is the most significant bit and P9 is the least significant bit. The value of each is determined from the jumper position as follows: 1-2 Position: 1 2-3 Position: 0
P14-P18	Specifies device number for U5. P18 is the most significant bit and P14 is the least significant bit. The value of each is determined from the jumper position as follows: 1-2 Position: 1 2-3 Position: 0
P19-P20	Specifies the LED function for the Activity LEDs associated with U16-U18. The Activity LEDs are the top LEDs on each of the dual right angle LED assemblies located at the front of the PCB. These jumpers set the value of the ACT_SEL[1:0] inputs to U16-U18 (the LED Interface FPGAs). The function of the Activity LED is as follows: P20 in 1-2 Position, P19 in 1-2 Position (00): The LED indicates when data is being transmitted or received on the associated port. P20 in 1-2 Position, P19 in 2-3 Position (01): The LED indicates when data is being transmitted on the associated port. P20 in 2-3 Position, P19 in 1-2 Position (10): The LED indicates when data is being received on the associated port. P20 in 2-3 Position, P19 in 2-3 Position (11): The LED indicates when collisions are occurring on the associated port.

Table 70. Reference Design Jumper Definitions (Continued)

P21	Determines whether boot block programming for the Flash memory devices is allowed. 1-2 Position: Disabled 2-3 Position: Enabled
P22	Determines which processor socket is enabled. 1-2 Position: Enables 80960Jx processor socket. 2-3 Position: Enables 80960Cx/Hx processor socket. NOTE: This jumper should only be changed with power removed from the PCB.
P23	Specifies which processor interrupt the LSERR# output of the PCI9060 will drive: 1-2 Position: NMI# 2-3 Position:XINT7# This jumper can be removed entirely, which results in no interrupt input being driven by the PCI9060 LSERR# signal.
P24	Specifies source of the PCI9060 BREQ input. 1-2 Position: Processor BSTAT (80960Jx) or BSTALL (80960Hx) signal. 2-3 Position: Processor BREQ signal (80960Hx only).

3.14 Serial Ports

3.14.1 Overview

The reference design incorporates an 85C30-8 DUART (Dual UART) to provide two serial ports. The ports can be used for debug/monitor purposes. For example, one UART can be used for a monitor (e.g., Intel's MON960) and the other UART can be connected to a terminal for displaying information on the hub's performance. The original intent was to locate the connectors for both UART channels at the rear of the board next to the Ethernet RJ-45 connectors. However, due to printed wiring board layout considerations (i.e., width of the board), one of the UART connectors was located in the rear of the hub and the other UART connector was located in the front of the hub.

The RS232 drivers and receivers for each UART are provided by a MAX233 device. These devices contain all of the circuitry to internally generate the ±12 V required by the drivers/receivers. A 6-pin modular jack connector is used for each serial port to save board space (compared to a DB-9 connector).

Sheet 27 of the Reference Design Schematic shows the DUART circuit.

3.14.2 DUART Addressing

The DUART has two control inputs that select the channel (A/B#) and the type of data transfer (D/C#). The data transfers can be data (D/C# = 1) or commands (D/C# = 0). These control inputs are connected as address lines on the printed wiring board (see Sheet 27 of the Reference Design Schematic). Because the DUART is an 8-bit device, the lower two address lines are PROC_BE[1:0]#. The A/B# input is connected to PROC_BE1# and the D/C# input is connected to PROC_BE0#.

This configuration makes the four addresses used by the DUART contiguous. The DUART address is fixed (in the Misc Logic FPGA) from 4000 0000 to 4000 00FF (256 byte region). As a result, there are 64 addresses for each DUART register, since address bits 2 to 7 are not included in the address decoding. Table 71 shows the DUART accesses performed for the "primary" DUART addresses. The table contents could be repeated 64 times to account for the address aliasing. For example, a Channel B Command access is performed at addresses 4000 0000 + 4n (where n = 0, 1, 2, ..., 63).

Table 71. DUART Addresses

Address	DUART Transfer Performed
4000 0000	Channel B Command
4000 0001	Channel B Data
4000 0002	Channel A Command
4000 0003	Channel A Data

3.14.3 Cabling

This section describes the cabling requirements for the serial ports on the reference design. Each serial port has a 6-pin modular jack connector on the printed wiring board. Two types of cable assemblies are available that use ribbon cable as shown in Figure 61. The type of cable assembly depends on the adaptor used to convert from the RJ-11 modular jack to a standard DB-9 (or DB-25) serial connector. A DB-9 connector is used with the intent of connecting it to a standard serial port on a PC. The adapter is not prewired and therefore the pinout could be used for either cable assembly. The wiring arrangement shown in Figure 62 is used because the “Standard” cable assemblies could be purchased off-the-shelf, whereas the “Reverse” cable assemblies could not.

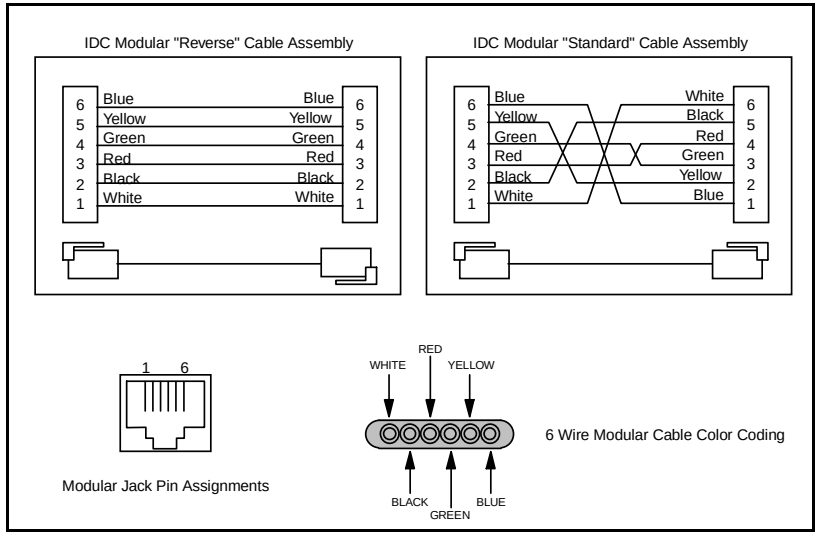


Figure 61. Modular Jack Serial Port Cabling Information

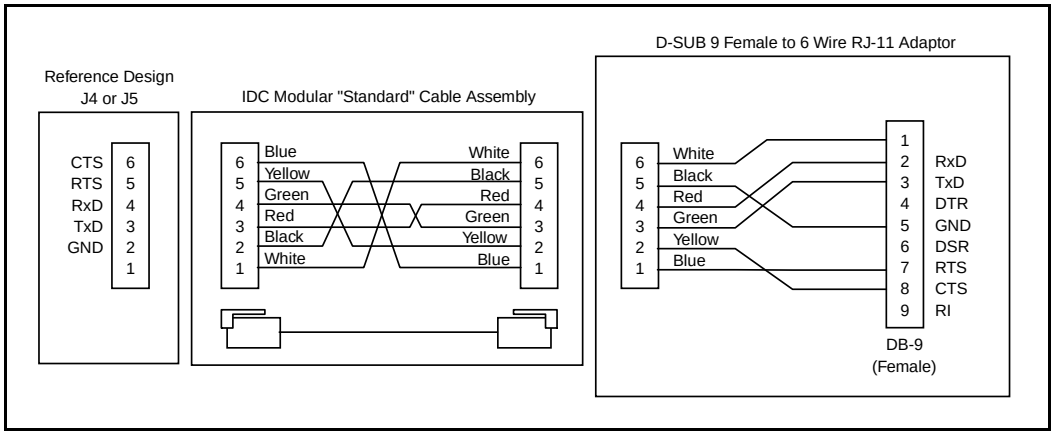


Figure 62. Reference Design Serial Port Cabling

3.15 10Base-T Physical Interface

The 10 BASE-T physical interface implemented on the reference design is based on the design recommended by Galileo Technology, but has been modified slightly. Figure 63 shows the physical interface implemented for each 10BASE-T port on the reference design. The physical interface circuits are located on Sheets 13 to 24 of the Reference Design Schematic.

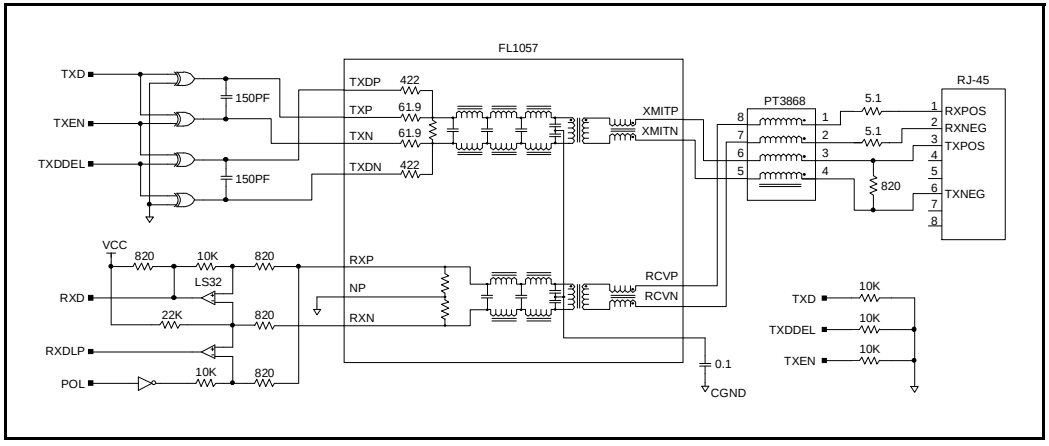


Figure 63. 10BASE-T Physical Interface

The transmit side of the physical interface is the same as the circuit recommended by Galileo Technology. The receive side has been modified as follows:

- The center point of the load resistance (NP) is connected to ground, whereas the Galileo circuit connects this point to VCC/2 using a resistive divider for each port. This requires offsetting the negative side of the RxD differential receiver and the positive side of the RxDLP differential receiver by +200 mV to achieve the same functionality. The POL signal is also inverted and used to set the level of the RxDLP differential receiver negative terminal to GND (POL=0) or +400 mV (POL=1).

  Note that the 26L32 differential receivers function correctly for common mode input voltages of ±7 V and the use of VCC/2 is to allow the receiver threshold to be modified. POL modified the RxDLP receiver threshold and the RXD signal modified the RXD receiver threshold to implement hysteresis.
- The “smart squelch” circuit is not implemented on the reference design.
- The 510 Ω pull up resistor for RxDLP is not implemented since the 26LS32 differential receiver generates TTL compatible outputs. A pull up resistor is implemented on the RxD signal to ensure the amount of hysteresis for the RxD differential receiver.

NOTES:

1. Each port is configured as 10BASE-T by pulling the TXD and TXDDEL signals to ground. The GT-48001 floats TXD and TXDDEL during reset and looks at their value when reset ends to determine the port type.

2. Each port is also configured for half-duplex by pulling TXEN low through a resistor. A jumper could have been used to allow TXEN to be pulled to +5 V (full-duplex) or GND (half-duplex). This was not done because a high current condition can exist during reset when TXEN is pulled high because the output drivers are enabled but the TXD and TXDDEL signals are not changing. The transformer in the FL1057 will thus appear as a short, allowing up to 50 mA of current to flow through the series source resistors in the FL1057. Although this should not harm the FL1057, it does require the ability to supply a higher current during reset (1.2 amps for 24 ports). Therefore, the processor software must configure the GT-48001 devices for those ports that are to be full-duplex.

3. Use of the discrete common mode chokes (U110-U133) is discretionary, since the FL1057 contains a common mode choke. The need for the discrete common mode chokes depends on the amount of EMI generated.

Crossover Wiring

The RJ45 connector pinout shown in Figure 63 is not the same as that of an RJ-45 connector on a network interface card (NIC) plugged into a PC. When connecting two 10BASE-T ports together, the transmit wire pair on each connector must be wired to the receive pins on the other connector. Thus, the wires must be “crossed over,” which can be accomplished either in the cabling or within the hub.

In general, the crossover wiring is performed within the hub to simplify cable management (allowing all cable segments to be pin-to-pin, also known as “straight through.” When the crossover wiring is performed within the hub, the port should be marked with an “X.” Note that if two hubs are connected together, the crossover must be performed in the cable if both hub ports have crossover wiring.

3.16 Power Supplies

The reference design requires only a +5 V power supply unless a PCI card is installed in the PCI expansion slot (J6); then, additional voltages may be required for the PCI card. The reference design circuitry also requires +3.3 V for the 80960Hx processor power supply, and VPP for the Flash memory programming supply, but these voltages are derived from +5 V.

The +3.3 V supply is generated by a 5 V to +3.3 V DC-DC converter located on the PCB. This circuit, shown on Sheet 3 of the Reference Design Schematic, uses a power supply controller chip, the Maxim MAX767. The circuit is based on the recommended circuit from Maxim. For more information on the +3.3 V power supply design, refer to Maxim’s documentation for the MAX767.

The VPP programming voltage for the Flash memory is +5 V when programming is enabled and ground when the programming is disabled. VPP is generated using a P-channel FET switch that is controlled by a signal (VPP_EN#) from the Misc Logic FPGA. When VPP_EN# is asserted, the FET transistor is on and VPP is connected to the +5 V supply. When VPP_EN# is deasserted, the FET

transistor is off and VPP is pulled to ground through a 1 KΩ resistor. See the Section 3.12, Misc Logic FPGA, for information on the VPP_EN# signal. The VPP switch is located on Sheet 3 of the Reference Design Schematic.

The PCI expansion slot may require +3.3 V, -12 V, and/or +12 V in addition to +5 V, depending on the card installed in the slot. The +3.3 V supply does not supply current to the PCI expansion slot because of its limited current capability, which is intended only for an 80960Hx processor. Therefore, if these additional voltages are required for the PCI expansion slot, they must be supplied from an external supply. Power is supplied to the board through P25 and P26. P25 is used only for +5 V and P26 is used for the PCI +3.3 V, -12 V, and +12 V supplies.

Refer to Sheet 33 of the Reference Design Schematic for pinouts of P25 and P26.

3.17 RMON FIFO Interface

The GT-480001 provides an interface that allows packet address and size information to be acquired using a FIFO. A dedicated pin on the GT-48001, CHIPSEL#, indicates when it is asserted that the data words that contain a packet's size, source address, and destination address are being read from the DRAM. These four data words can be captured by connecting a FIFO to the DRAM data lines and enabling the FIFO to be loaded with the DRAM data read when CHIPSEL# is asserted.

Figure 64 shows the signal relationships when fast page mode DRAM is used. Contact Galileo Technology for the location of the packet parameters (size, source address, and destination address) within the four words that are read when CHIPSEL# is asserted. With fast page mode DRAM, the data becomes valid after the CAS# signal is asserted and remains valid until CAS# is deasserted.

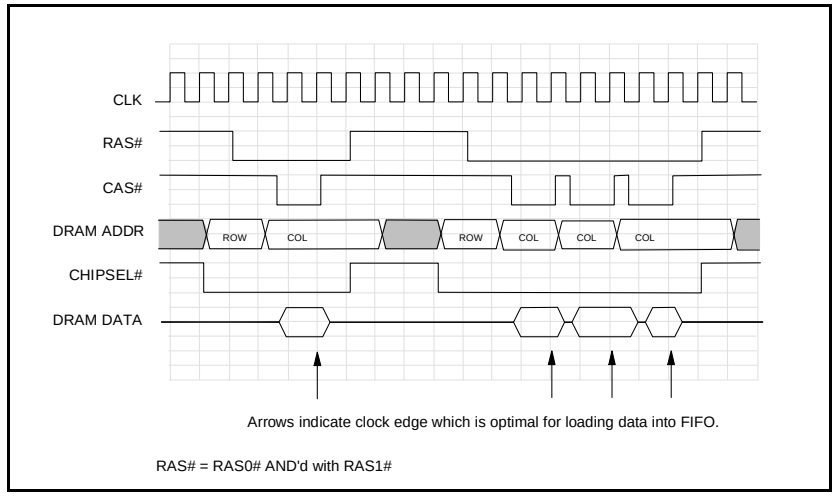


Figure 64. FPM Signal Relationships

Figure 65 shows the signal relationships when Extended Data Out (EDO) DRAM is used. The timing is similar to the timing for Fast Page Mode DRAM. EDO DRAM is essentially an enhanced version of Fast Page Mode DRAM. The difference lies in the fact that the output data is not tri-stated when CAS# is deasserted. This allows a shorter CAS# pulse to be used, as shown in Figure 65. As a result, memory accesses require less time when EDO DRAM is used.

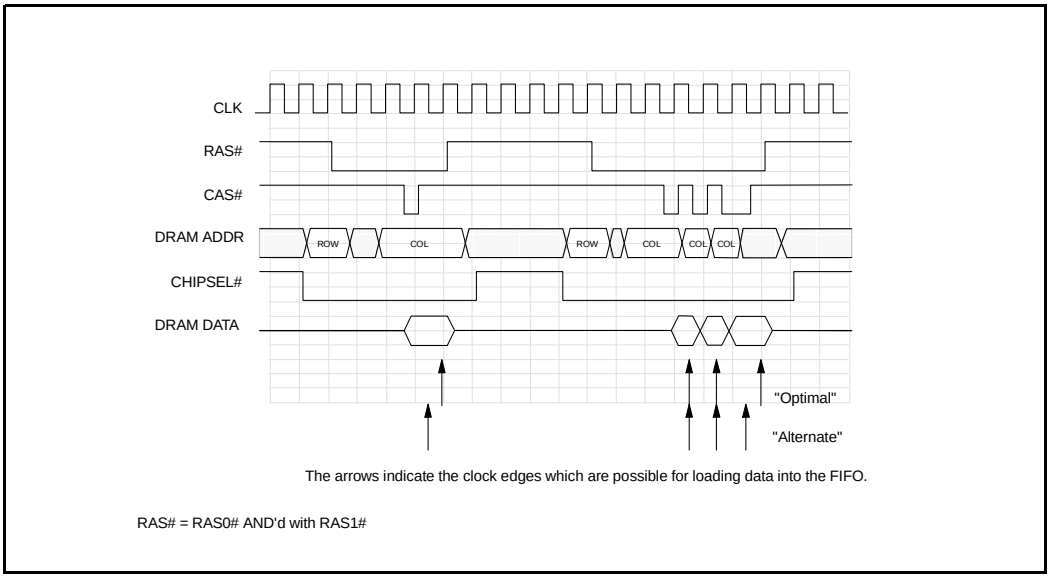


Figure 65. EDO Signal Relationships

Two types of FIFOs are generally available: asynchronous and synchronous. An asynchronous FIFO requires one or more chip enable signals and a WR# signal. The data is loaded into the FIFO when the WR# signal is deasserted. A synchronous FIFO requires a write clock and a write enable. Data is written into the FIFO whenever the write enable signal is asserted and the clock makes a positive transition.

A synchronous FIFO, the IDT72225LB, is used in the reference design because asynchronous FIFOs are generally not available in widths larger than 8 or 9 bits. Synchronous FIFOs, on the other hand, are widely available in 18-bit widths, with some 36-bit wide devices available. The IDT72225LB is a 1Kx18 FIFO that requires that two devices be used to implement a 1Kx32 FIFO. Note that two bits on each device are unused. An advantage of using the IDT72225LB is that it is one of a family of (pin compatible) devices with sizes varying from 256Kx18 to 4Kx18. Therefore, if a 1Kx32 FIFO proves to be too small or too large, another device in the family can be utilized. Also, there are compatible devices from other manufacturers (e.g., the Cypress CY7C4225).

The IDT72225LB requires an active low write enable (WEN#) and a clock (WCLK). Data is loaded into the IDT72225LB on a rising edge of WCLK when WEN# is low.

For FPM DRAM, CAS# can be connected to WCLK and CHIPSEL# can be connected to WEN#. However, the same is not true for EDO DRAM, which cannot use CAS# as the clock signal for the FIFO.

Because no inexpensive method was found for generating the WCLK signal when EDO DRAM was used, the RMON FIFO interface that was implemented only supports FPM DRAM. **Therefore, the GT-48001 needs to be configured for FPM DRAM when packet information is being collected in the FIFO even if EDO DRAM is used for the packet DRAM**.

Another consideration for the RMON FIFO interface is the fact that CHIPSEL# may be asserted during packet DRAM refresh cycles. Therefore, a circuit is required that inhibits CHIPSEL# when the packet DRAM is being refreshed (the refresh cycles are CAS before RAS).

Figure 66 shows a block diagram of the FIFO interface implemented on the reference design.

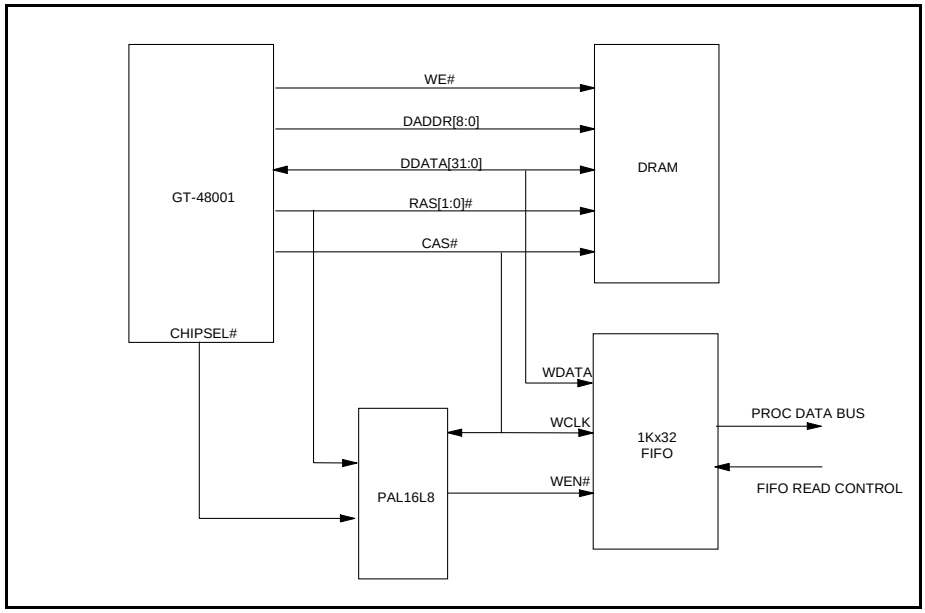


Figure 66. Block Diagram of FIFO Interrupt Implemented

A PLD (PAL16L8) is used to implement the circuit that inhibits FIFO writes during DRAM refresh. The PLD uses the RAS[1:0]# and CAS# inputs to determine if a refresh cycle is being performed. If a refresh cycle is detected, the WEN# signal is forced high (deasserted); otherwise CHIPSEL# is passed through to the FIFO WEN# input.

The logic for the PLD is shown in Figure 67. This logic only allows the FIFO WEN# input (GATED_CS#) to be asserted when one of the RAS# signals (RAS1#, RAS0#) is asserted, CHIPSEL# is asserted, and a DRAM refresh is not being performed. The PLD used is the Cypress PALC16L8-25. Its logic was defined in VHDL, then synthesized using the WARP2 VHDL compiler from Cypress.

Sheets 10, 11, and 12 of the Reference Design Schematic show the RMON FIFO interfaces.

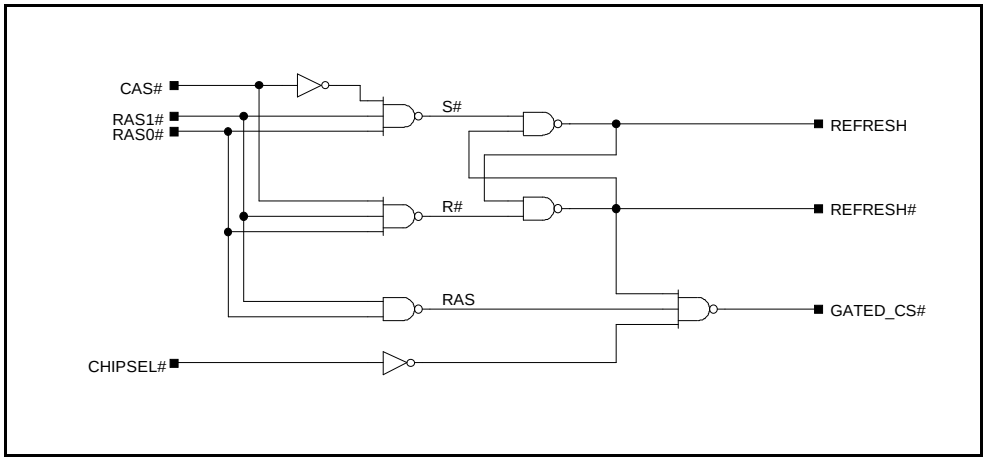


Figure 67. RMON FIFO PLD Logic

The following is the VHDL implementation of the logic for the RMON FIFO PLD:

```
use work.IOPKG.all;
use work.GATESPKG.all;
use work.CYPRESS.all;
use work.RTLPKG.all;

entity GATED_CS is
  port (
        nCAS        : IN bit;
        nRAS0       : IN bit;
        nRAS1       : IN bit;
        nCHIPSEL    : IN bit;
        REFRESH     : INOUT bit;
        nGATED_CS   : OUT bit
       );
  attribute order_code of GATED_CS:entity is "PALC16L8-25PC";
  attribute part_name  of GATED_CS:entity is "C16L8";
end GATED_CS;

architecture archGATED_CS of GATED_CS is
  signal CAS      : bit;
  signal RAS      : bit;
  signal CHIPSEL  : bit;
  signal nS       : bit;
  signal nR       : bit;
```

```
  signal nREFRESH : bit;
begin
  inv_1: INV
    port map (
                A  => nCAS,
                QN => CAS
              );
  inv_2: INV
    port map (
                A  => nCHIPSEL,
                QN => CHIPSEL
              );
  nand3_1: NAND3
    port map (
                A  => CAS,
                B  => nRAS0,
                C  => nRAS1,
                QN => nS
              );
  nand3_2: NAND3
    port map (
                A  => nCAS,
                B  => nRAS0,
                C  => nRAS1,
                QN => nR
              );
  nand2_1: NAND2
    port map (
                A  => nRAS0,
                B  => nRAS1,
                QN => RAS
              );
  nand2_2: NAND2
    port map (
                A  => nS,
                B  => nREFRESH,
                QN => REFRESH
              );
  nand2_3: NAND2
    port map (
                A  => REFRESH,
                B  => nR,
                QN => nREFRESH
```

```
            );
  nand3_3: NAND3
    port map (
              A  => nREFRESH,
              B  => RAS,
              C  => CHIPSEL,
              QN => nGATED_CS
            );
end archGATED_CS;
```

Another approach to reading data from the FIFO is to read the FIFO contents via the PCI bus. Galileo Technology implemented a circuit that allows the FIFO to be mapped into the packet DRAM memory space on their GT-48001 evaluation board. The concept is shown in Figure 68. The main disadvantage of this approach is that the amount of DRAM is limited to 1 Mbyte. For more details on this circuit, contact Galileo Technology.

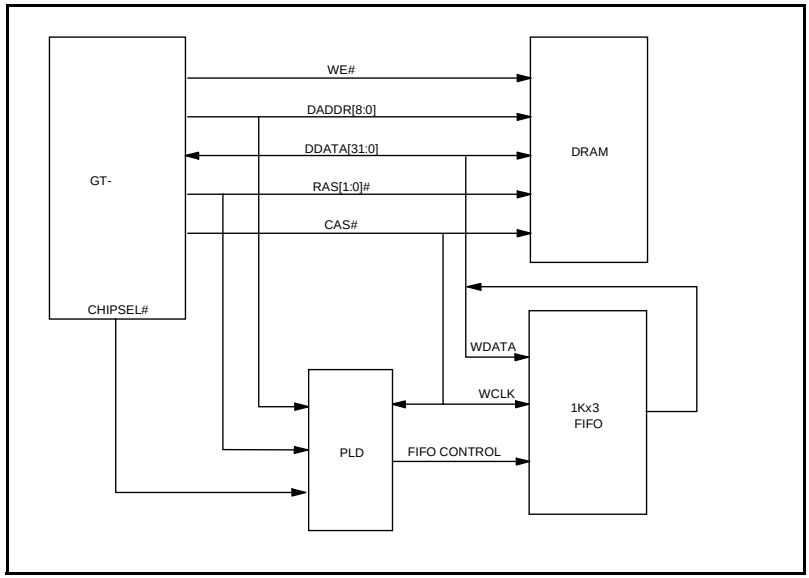


Figure 68. Reading FIFO Contents Using PCI Bus

3.18 PCB Layout Considerations

3.18.1 Board Dimensions and Component Placement

Figure 69 shows the dimensions of the reference design printed circuit board (PCB). All dimensions are in inches and are nominal.

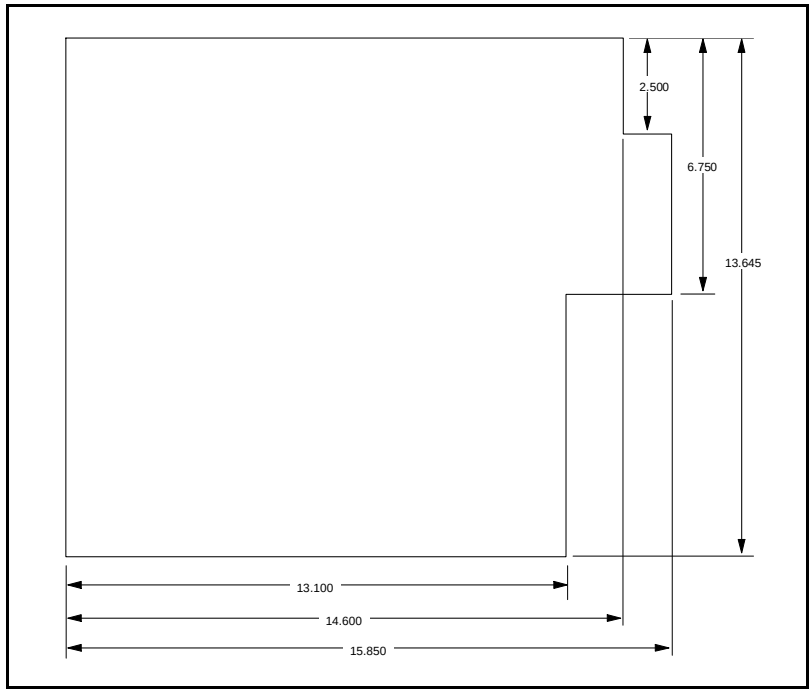


Figure 69. PCB Dimensions

Figure 70 shows the locations of components that are fixed for compatibility with the enclosure. All dimensions are in inches and are nominal. The irregular board shape (i.e., not rectangular) is a result of the desire to put the board into an enclosure for demonstration purposes.

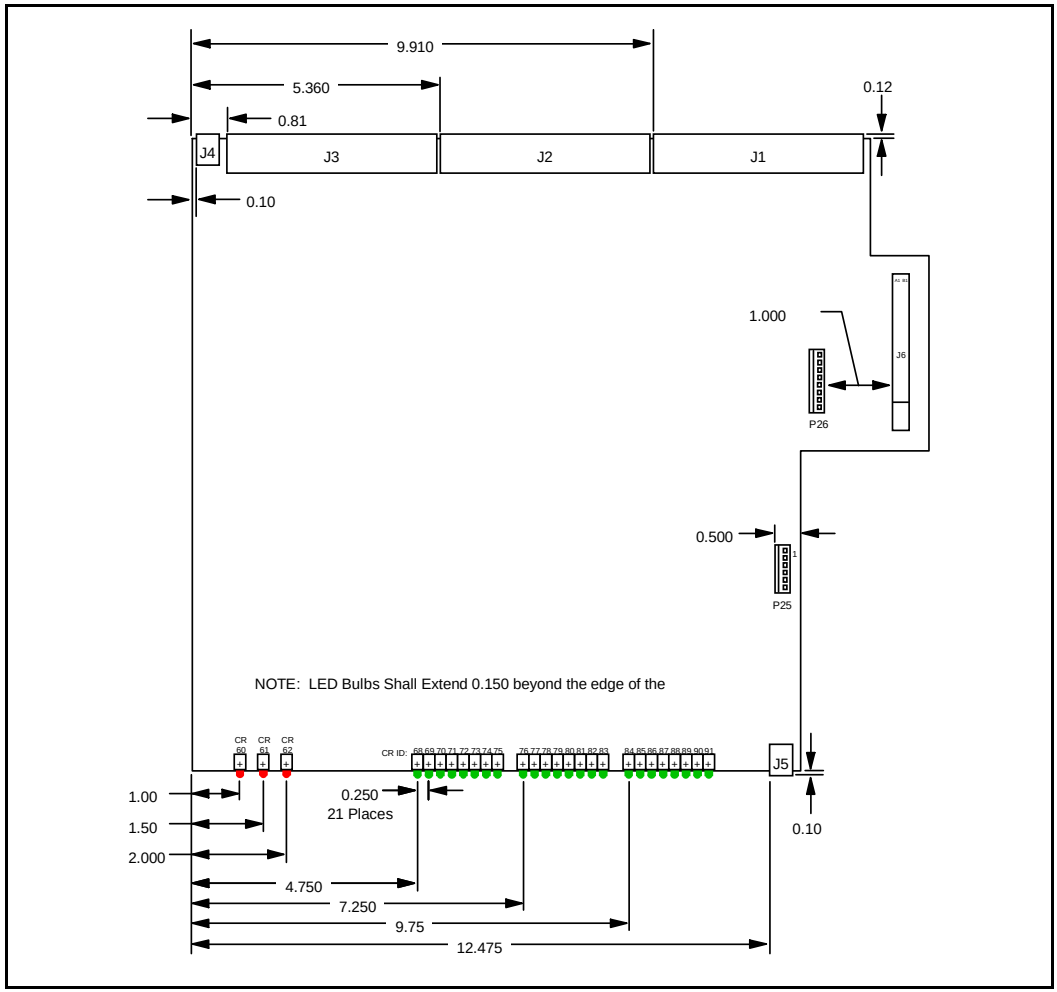


Figure 70. Required Component Locations

Figure 71 shows how the PCB, a +5 V power supply, an AC inlet, and an AC switch are installed in an enclosure. The enclosure used is a 19-inch rack-mountable case from Lansing Instrument Corporation (Ithaca, NY). The enclosure, model P1F14-0B1B, was customized by Lansing to include cutouts in the front and rear panels for the LEDs, AC inlet, AC switch, and J1-J5. Lansing also silkscreened lettering on the front and rear panels for LED and connector identification. The PCB was mounted on 3/8" standoffs and the power supply was mounted on 1/4" standoffs. The power supply, model LPS42 from Astec, was chosen because of its low height, which was necessary because the enclosure height is 1.72" (with 1.50" height inside the enclosure).

Figure 71 also shows a potential component layout based on the expected routing of address and data buses between components. Figures 72 through 76 show the locations of these buses on the potential layout.

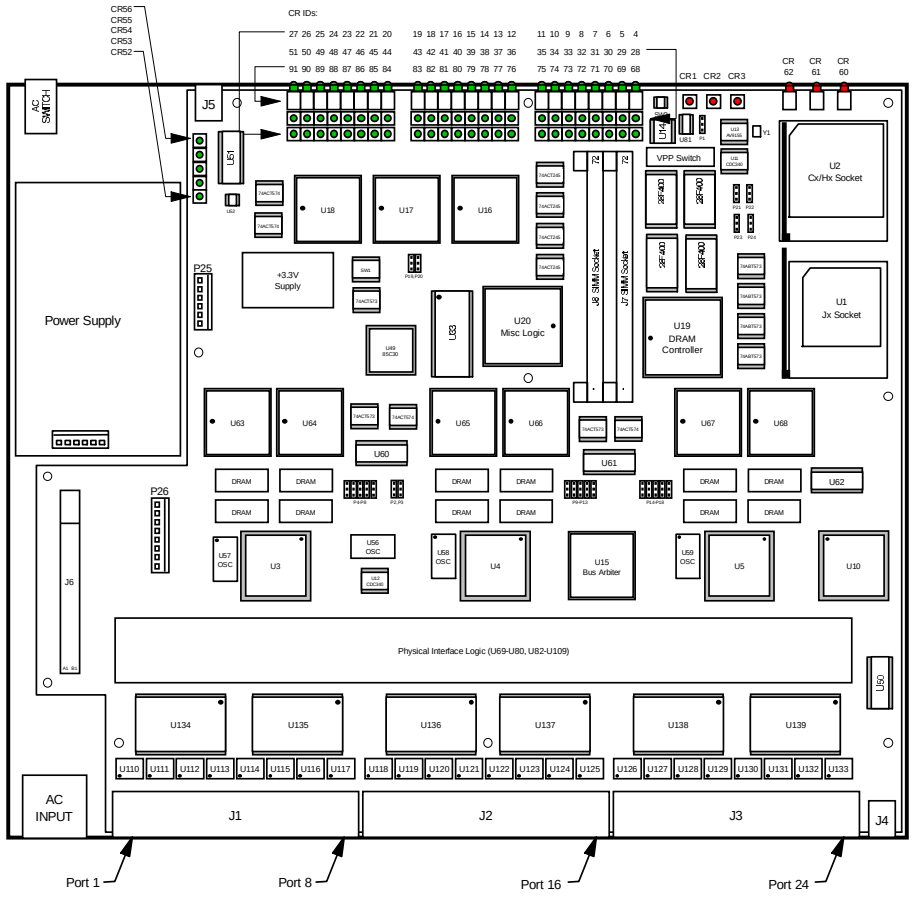


Figure 71. Locations of Components in Case

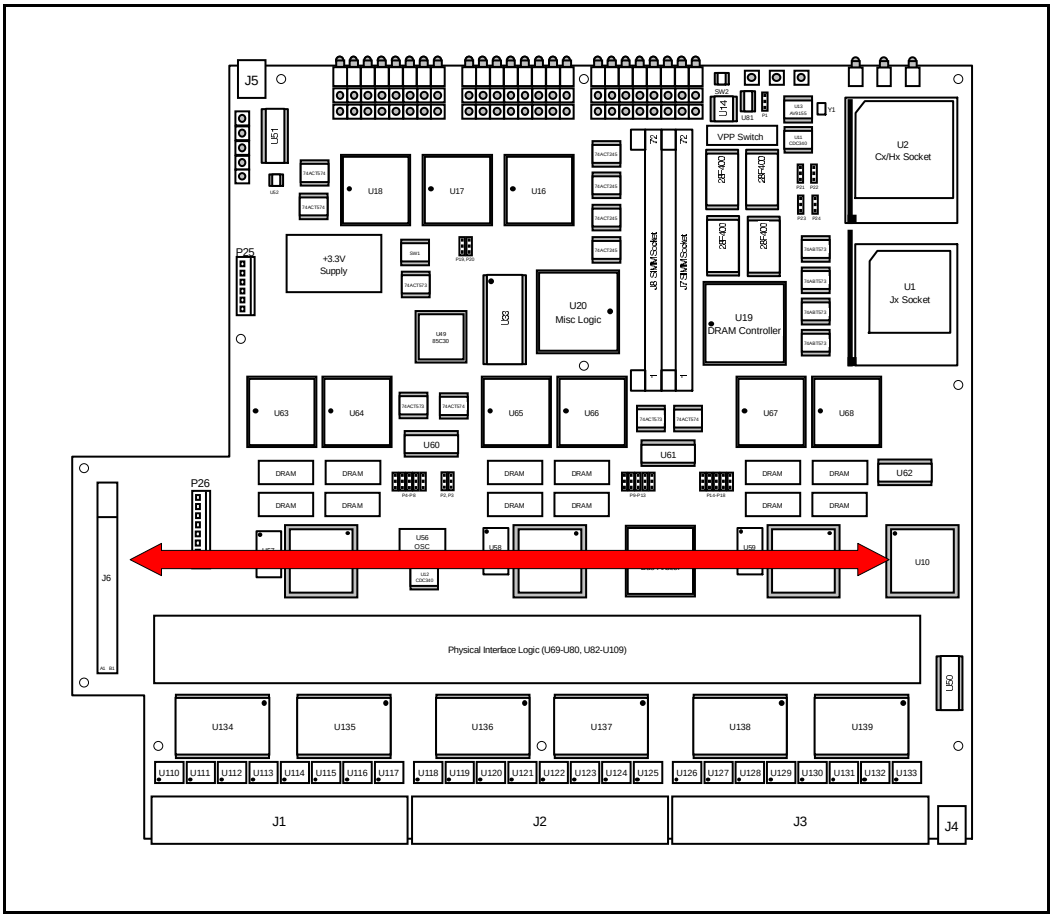


Figure 72. PCI Bus Routing

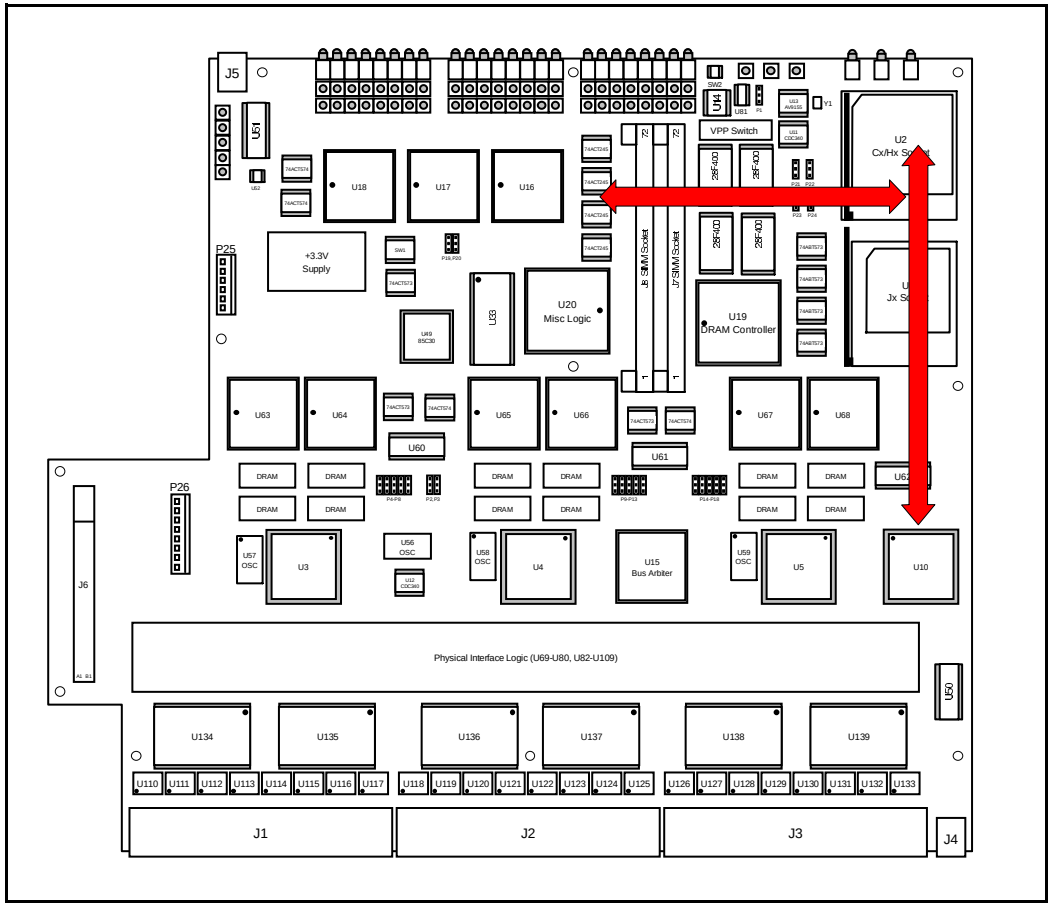


Figure 73. Processor Data Bus Routing

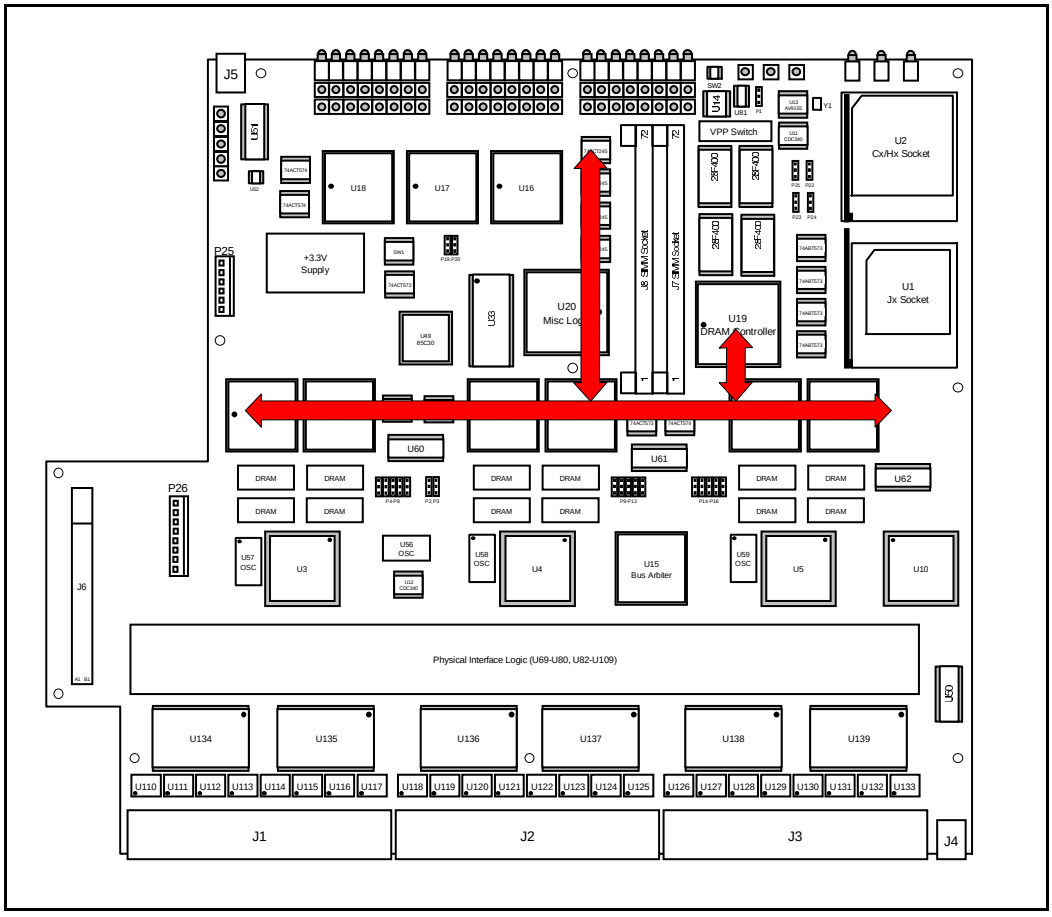


Figure 74. Processor Slow Data Bus Routing

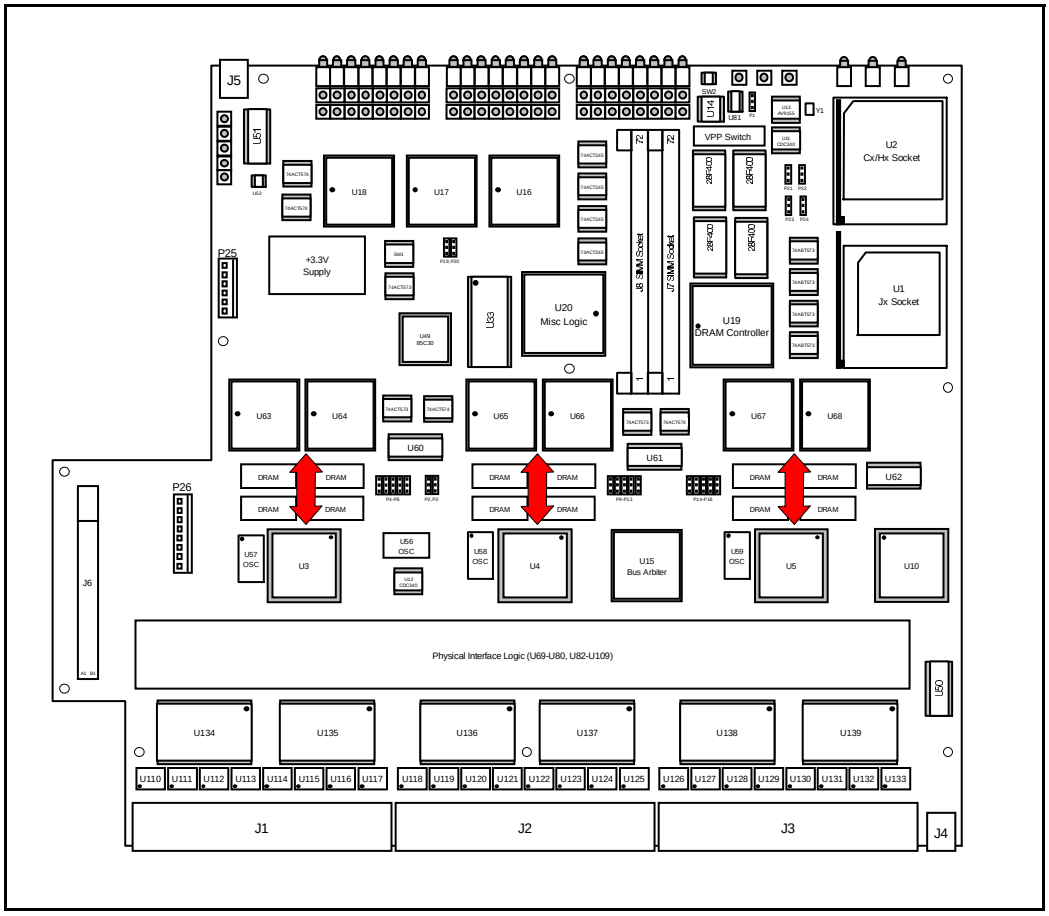


Figure 75. GT-48001 DRAM BUS Routing

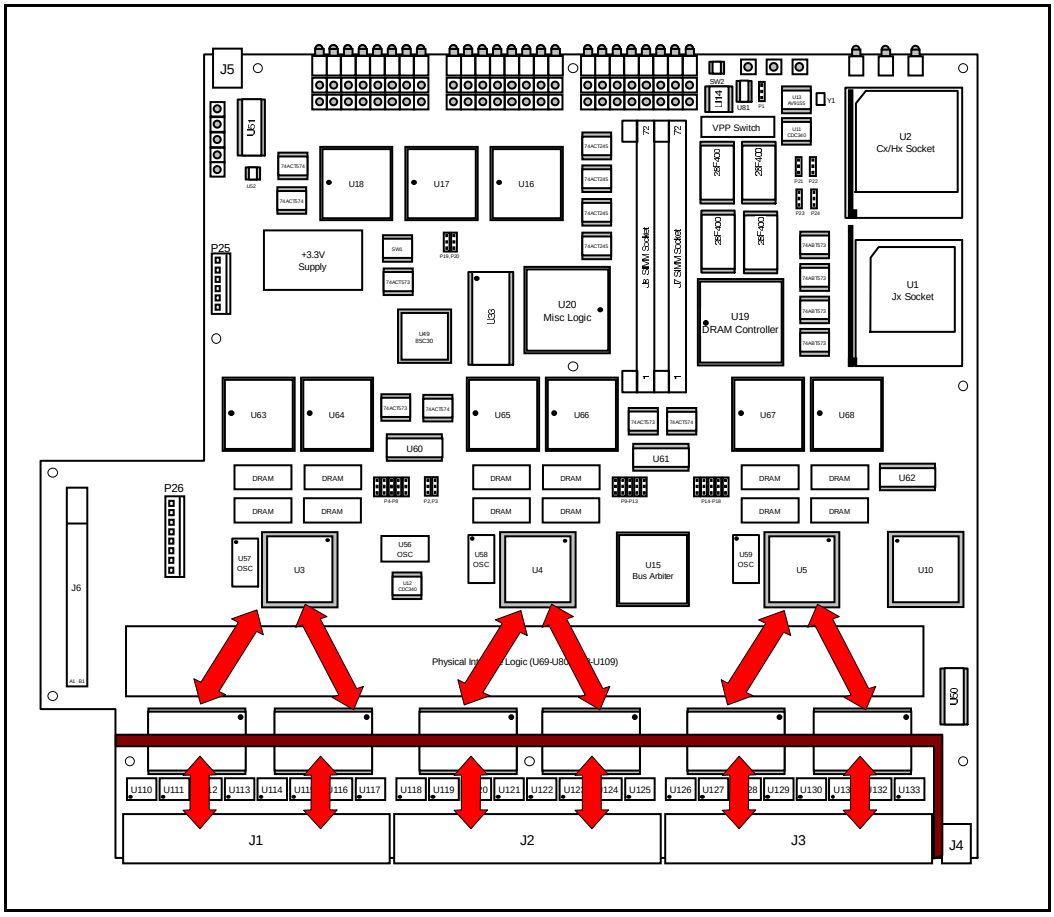


Figure 76. 10BASE-T Physical Interface Routing

3.18.2 PCB Layers

The PCB is a 6-layer board with 2 layers serving as power planes, as shown in Figure 77. The two power planes are partitioned as shown in Figures 78 and 79. The reason for providing CGND on each of these layers is to provide shielding for the signals routed from the quad 10BASE-T filters to the RJ-45 connectors. In general, these signals are routed on internal layers (C, D) with no other signal routing allowed in the region enclosed by the CGND planes. The layer E plane was also partitioned to provide a low resistance path for the 80960Cx/Hx power supply.

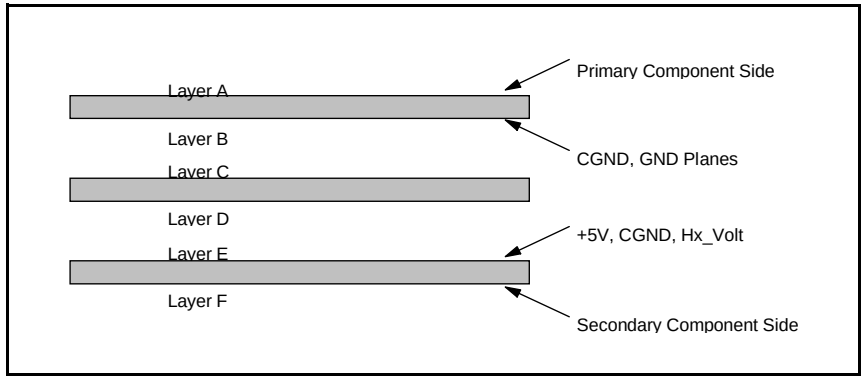


Figure 77. PCB Layers

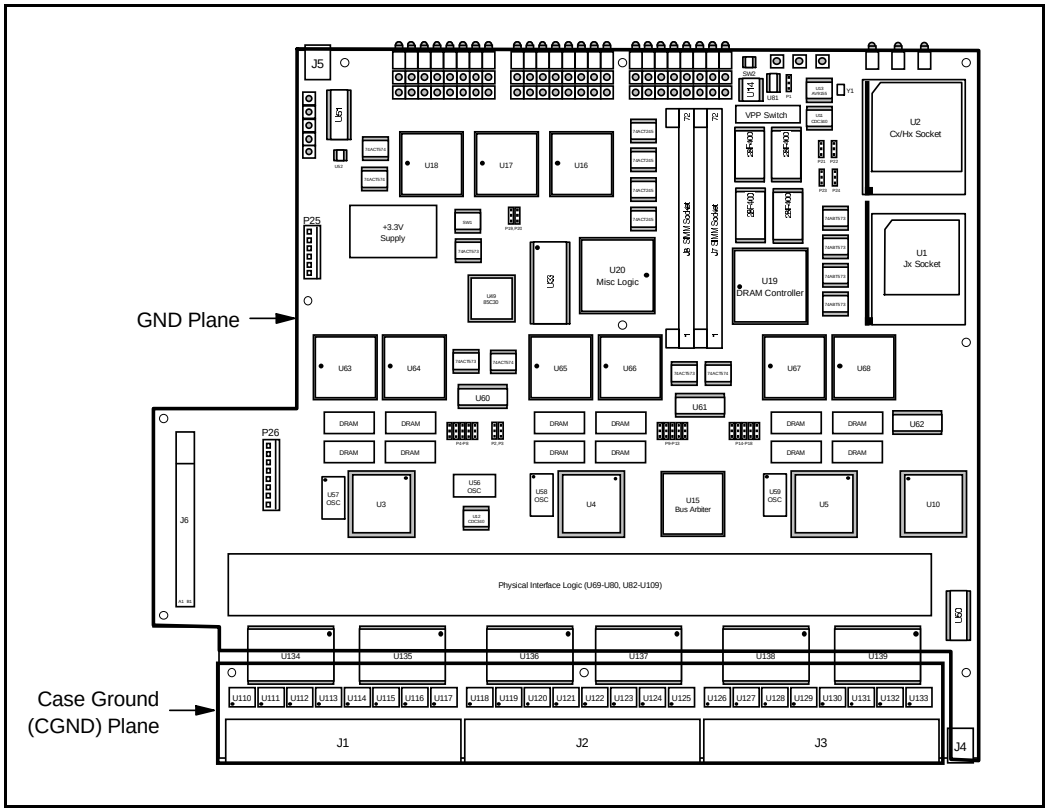


Figure 78. Layer B Partitioning

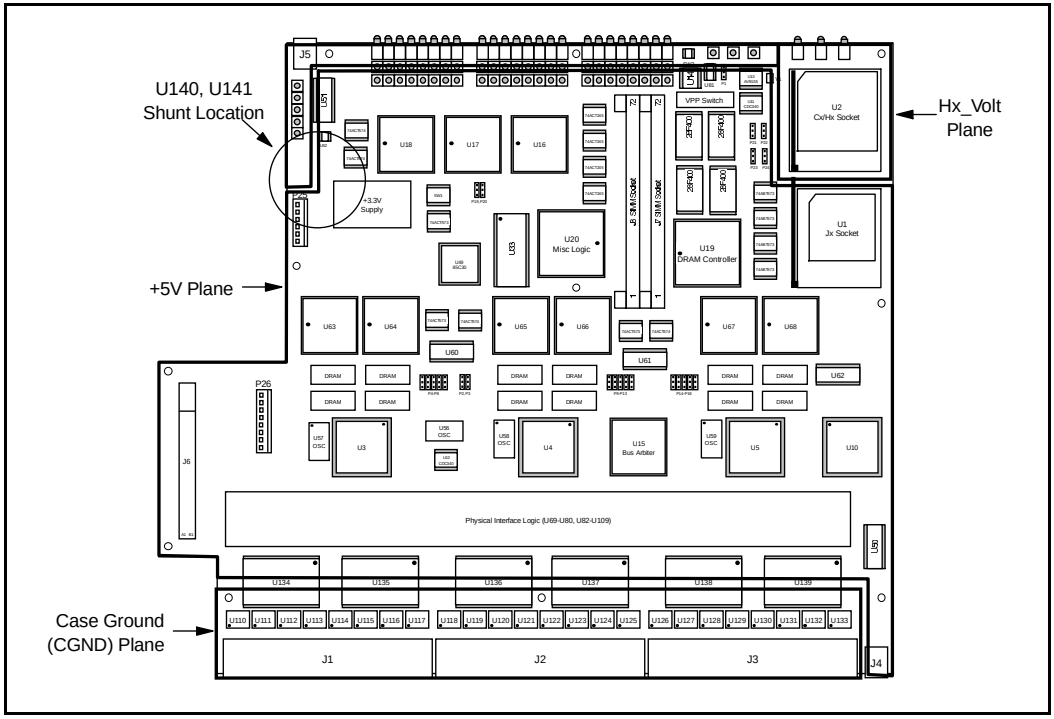


Figure 79. Layer E Partitioning

Additional Layout Notes

1. The mounting hole between U136 and U137 should provide an electrical connection from the CGND plane to the case when a conductive spacer is used.

2. The mounting hole in the center of the board should provide an electrical connection from the GND plane to the case when a conductive spacer is used.

3. J1, J2, and J3 have three case pins, of which at least one (preferably the one in the center) should connect to Case Ground (CGND). Note that there is a 20 mil gap allowed between J1/J2 and between J2/J3 to allow for component tolerances.

4. The following signals (80 MHz clock signals) should be routed with as short a trace as possible (reference Sheets 4, 5, and 6 of the Reference Design Schematic):

   U57-8 to U3-91
   U58-8 to U4-91
   U59-8 to U5-91

5. The following signals should be routed such that they are within ±0.1 inches of the same length.

   PCI_CLK1
   PCI_CLK2
   PCI_CLK3
   PCI_CLK4
   PCI_CLK5

   The trace length for PCI_CLK6 signals should be 2.5 ± 0.1 inches shorter than the nominal length of the above signals to account for the clock trace length on the PCI expansion board. These signals originate in U12 (see Sheet 3 of the Reference Design Schematic) and are routed through 22 Ω series terminating resistors to a single destination. The length of a net includes the distance from U12 to the series terminating resistor.

6. The following signals should be routed such that they are within ±0.1 inches of the same length.

   PROC_CLK1
   PROC_CLK2
   PROC_CLK3
   PROC_CLK4
   PROC_CLK5

   These signals originate in U11 (see Sheet 2 of the Reference Design Schematic) and are routed through 22 Ω series terminating resistors to a single destination. The length of a net includes the distance from U11 to the series terminating resistor.

7. The traces between the components listed in the following table and their associated 22 Ω series terminating resistors should be as short as practical.

Component	Associated Resistors	Schematic Sheet
U13	R300-R302	2
U11	R303-R307	2
U12	R310-R315	3
U3	R316-R328	4
U4	R329-R341	5
U5	R342-R354	6
U20	R355-R357	29
U19	R358-R377	29

8. The following signals are routed on wider traces to reduce the path resistance:

   +VPP (100 mil)
   PCI +3.3 V (200 mil)
   PCI -12 V (50 mil)
   PCI +12 V (50 mil)

9. The traces from U53 to R303 should be routed directly to the pads of R949. Note that these signals serve as sense leads and would provide an inaccurate reading if not connected directly to R949.

10. The traces between U134-U139 and U110-U133 and the traces between U110-U133 and J1-J3 are located on the inside routing layers (C and D) except as necessary to connect to resistors.

11. Figure 80 shows signals in the +3.3 V supply that are high current traces (wide lines) and should be routed with wider traces. Since these traces should be short, they can be routed with 50 mil traces. The trace which connects U141 to R949/C449 should be a minimum of 200 mil wide. Note that the traces can be reduced in width when connecting to a component.

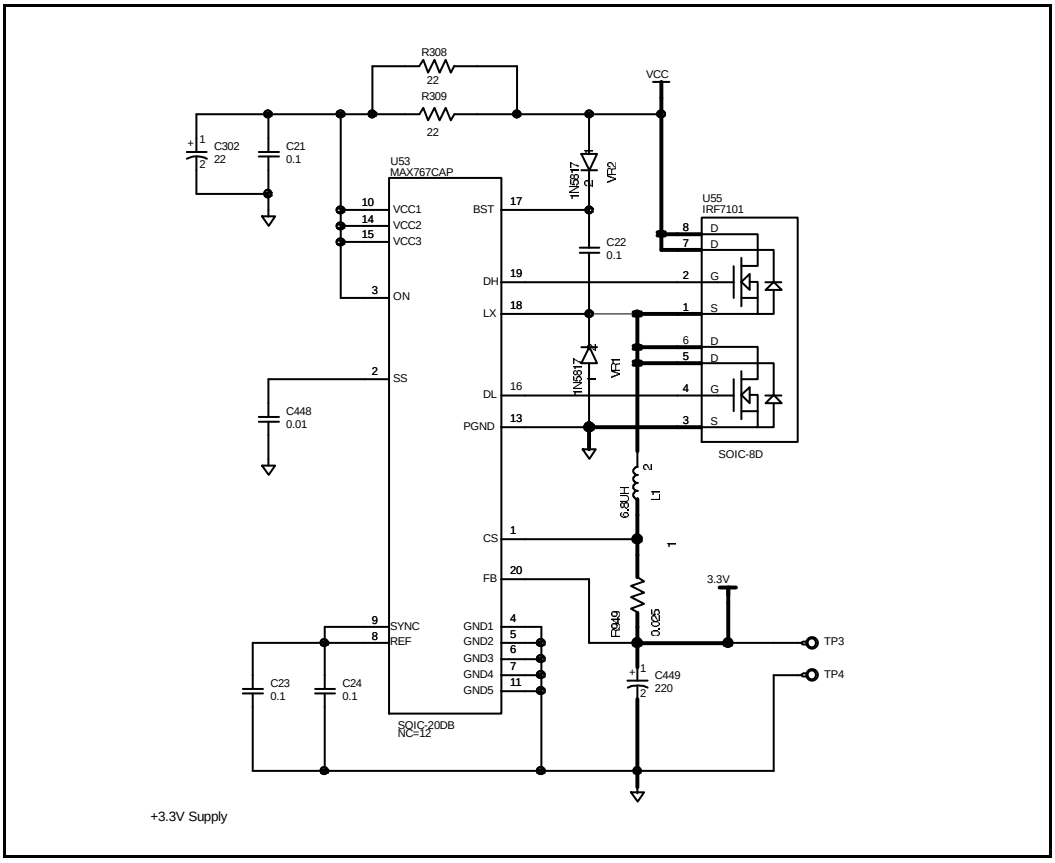


Figure 80. High Current Paths in 3.3 V Supply

3.18.3 Device Pin Numbering

The reference design printed wiring board allows ZIF (Zero Insertion Force) sockets to be installed for the processors, which allows the processors to be easily replaced for evaluation purposes. The choice of PGA packages for the processors was also made with this idea in mind, since all speed grades of the 80960Jx, 80960Cx, and 80960Hx processor families are available in PGA packages.

The 80960Jx processor PGA package is a 132-pin PGA package (14x14 pin matrix with the inside of the matrix missing). It was not possible to purchase a 14x14 ZIF socket off-the-shelf; therefore, a 225-pin PGA ZIF socket was used. The ZIF socket was modified by removing pins to provide a footprint for a 132-pin PGA. Figure 81 shows how the 15x15 ZIF socket was modified to provide a footprint for a 132-pin PGA and the pinout that was used. Note that from a layout standpoint, the circuit board is designed for a 132-pin PGA package. The only effect of the ZIF socket was to require a reserved area for installing the socket.

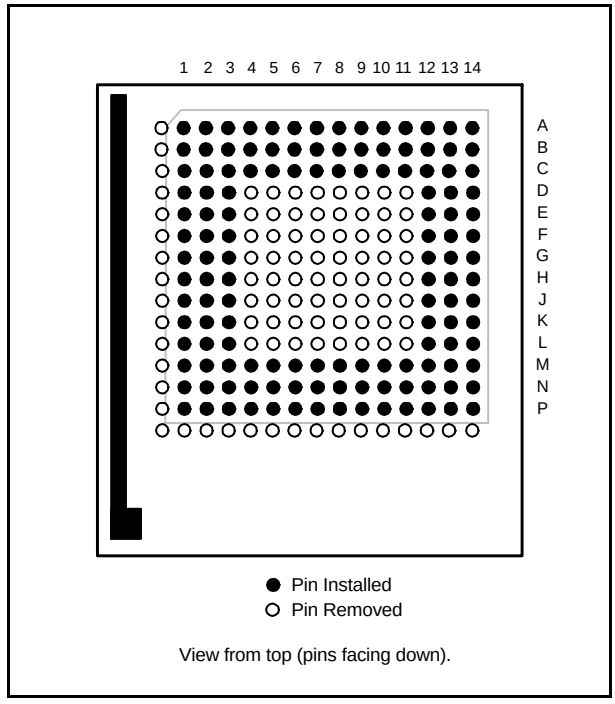


Figure 81. 80960Jx Processor ZIF Socket Pin Numbering

Figure 82 shows the pinout for the 168 pin 17x17 ZIF socket used for the 80960Cx/Hx processor. This socket was a standard product which could be purchased off the shelf. The pinout is shown here for clarity.

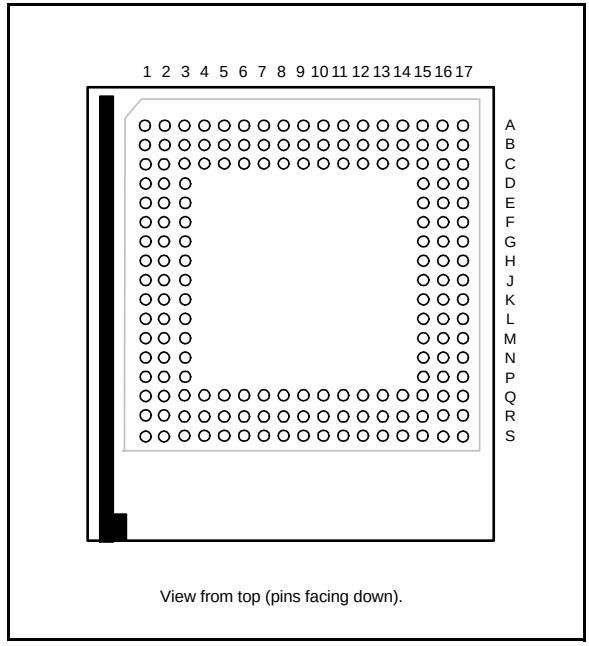


Figure 82. 80960Cx/Hx Processor ZIF Socket Pin Numbering

Figures 83 through 86 show pin numbering for components that are not standard to clarify pin locations in relation to the schematic.

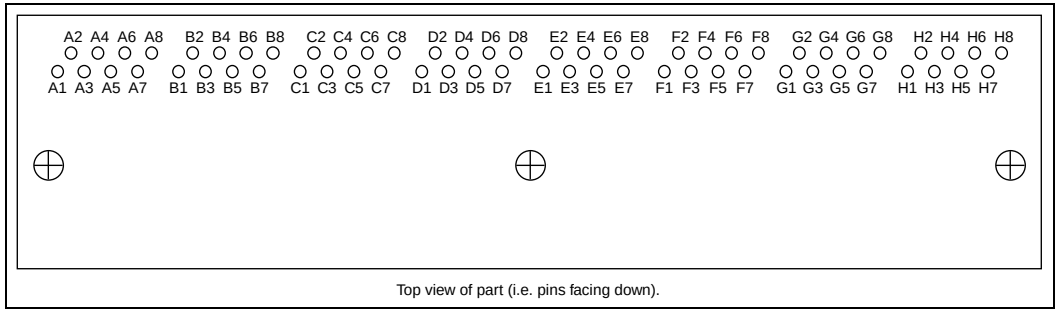


Figure 83. Post RJ45 Pin Numbering (J1, J2, and J3)

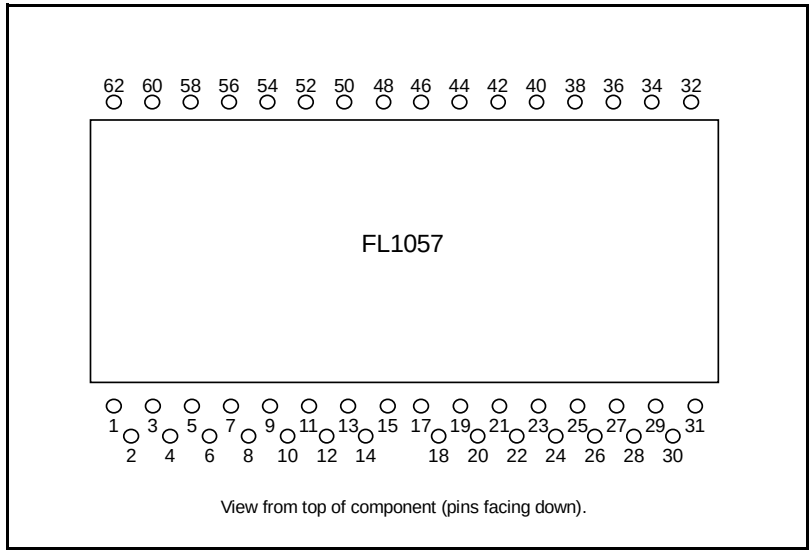


Figure 84. FL1057 Pin Numbering (U134-U139)

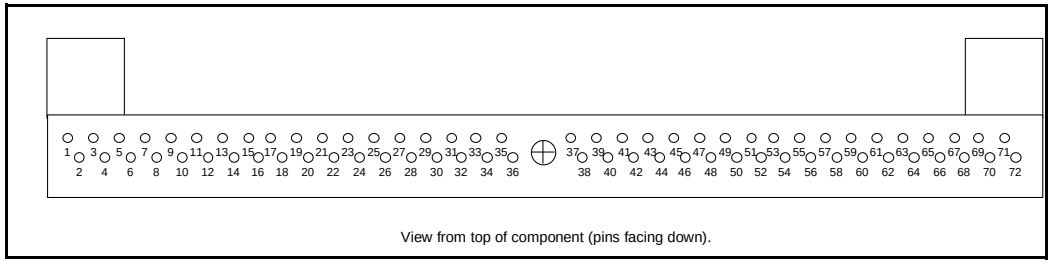


Figure 85. SIMM Socket (J7, J8) Pin Numbering

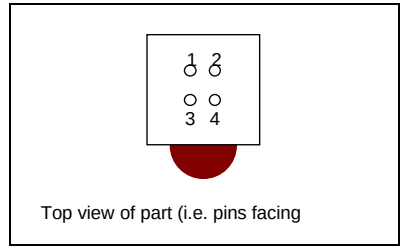


Figure 86. Dual LED (CR68-CR91) Pin Numbering

3.19 Enhancements/Cost Reduction

This section will describe possible changes to the reference design to add features or reduce costs.

3.19.1 High Speed Port, 80960RP

A switching hub often incorporates the ability to install a high speed LAN interface such as ATM or Fast Ethernet. The high speed port could be connected to a server or a corporate backbone to prevent a bottleneck in the network if many of the 10 Mbps ports are communicating with a single device (i.e., a file server). The high speed port can be added to either the PCI bus or the processor's local bus. Many high speed LAN ICs now incorporate a PCI bus because it simplifies the design of a PCI card based on the device. Figure 87 shows a block diagram of the reference design with a 100 Mbps Ethernet port connected to the PCI bus. The number of 10 Mbps ports was reduced to 16 based on the assumption that high speed port would utilize the board area occupied by the circuitry for eight 10 Mbps ports. The data transmitted or received via the high speed port must be processed by the 80960 processor, since the high speed port and the GT-48001 devices cannot communicate with each other.

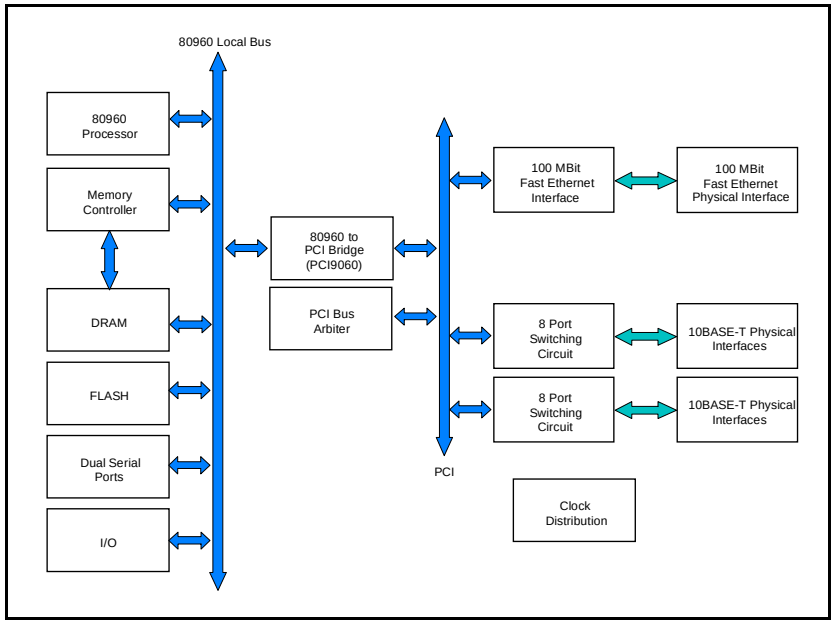


Figure 87. Reference Design with 100 Mbps Ethernet Port

The 80960RP processor can be a very cost effective solution in the reference design since it integrates a significant number of the functions shown in Figure 88. The 80960RP contains an 80960Jx core, a PCI bridge, PCI bus arbiter and a memory controller. Thus, the reference design can be simplified while reducing costs by using an 80960RP. Figure 88 shows the reference design with an 80960RP replacing the functions it incorporates. Note that since the 80960RP has two PCI buses, the high speed port is connected to one bus by itself and will not interfere (i.e., delay) bus transactions on the GT-48001 PCI bus. Even though the 80960RP has a PCI bridge, the bridge is essentially used as two separate PCI interfaces.

Note: The 80960RP was not used in the switched ethernet reference design because it was not available when the design was started.

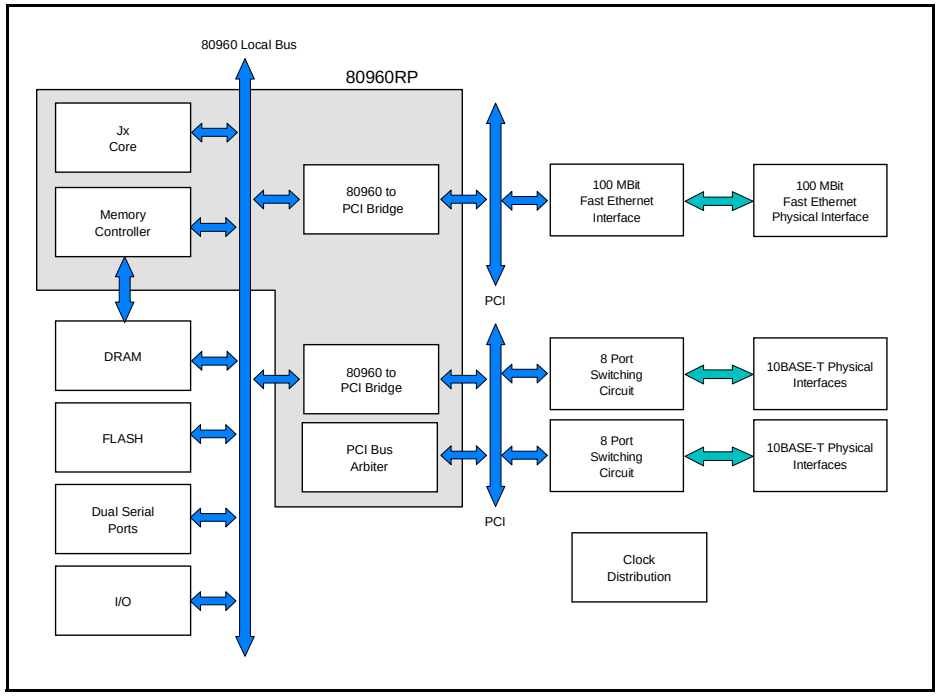


Figure 88. Reference Design With High Speed Port Utilizing an 80960RP

3.19.2 Galileo GT-32090 Jx System Controller Chip

The DRAM Controller FPGA and some functions in the Misc Logic FPGA can be replaced by the GT-32090 system controller from Galileo Technology. The GT-32090 is a highly integrated system controller for embedded control applications. The GT-32090 controls two separate and independent buses, the CPU's 32-bit wide address/data bus, and a 16-bit I/O bus (named simple I/O bus or SIO bus). The two buses can work concurrently and at different frequencies.

The GT-32090 has a direct interface to the 80960Jx family of processors. It has the capability to support various device types through programmable address decoding and timing. It controls device types such as ROM, Flash, and SRAM with different size and timing requirements. The GT-32090 includes a DRAM controller that supports both fast page mode and EDO DRAM types.

The GT-32090 has a three-channel DMA controller that supports chaining via linked lists of descriptors. The DMA controller can move data between devices and DRAM on the CPU bus or between the CPU bus and devices on the SIO bus.

The SIO bus is an 80186-like bus that interfaces to a large variety of support components, such as UARTs, SCSI controllers, and other low-cost devices. The GT-32090 includes a direct interface to two 16-bit PCMCIA slots.

3.19.3 Other Cost Reductions

Other possible cost reductions that could be implemented are listed below:

- Eliminate the EPROM and serial EEROM devices and use the boot block Flash memory devices to store the boot code and parameters.
- Attach the processor DRAM directly to the printed circuit board, eliminating the SIMM sockets.
- Eliminate LEDs and jumpers that have been incorporated to aid in debugging or are primarily evaluation oriented. For example, the PCI connector could be eliminated.
- Operate the PCI bus and the processor at the same clock frequencies. This would eliminate at least one clock oscillator and potentially eliminate the clock synthesizer.

3.19.4 Performance Improvements

The performance of the processor can be improved for certain configurations by tailoring the device wait state profiles for a specific configuration. This allows logic that has been added to allow flexibility to be removed from either the Misc Logic or DRAM Controller FPGA. For example, if processor DRAM is installed directly on the PCB, the number of DRAM banks (and depth) that need to be supported are known, allowing that logic to be eliminated. This should improve timing within the FPGA, possibly allowing the number of required wait states to be reduced.

Table 72. Parts List (Sheet 1 of 3)

Item	Qty	Reference	Description	Mfg. Part Number	Manufacturer	Notes
1	1	U1	32-Bit Microprocessor	80960Jx	Intel	
2	1	U2	32-Bit Microprocessor	80960Cx/Hx	Intel	
3	3	U3-U5	Switched Ethernet Controller	GT-48001	Galileo Technology	
4	4	U6-U9	256Kx16 Flash	E28F400CVT80	Intel	
5	1	U10	80960 to PCI Interface	PCI9060	PLX Technology	
6	2	U11,U12	Low-Skew Clock Buffer	CDC340DW	Texas Instruments	
7	1	U13	Frequency Generator	AV9155A-23	ICS	
8	1	U14	Reset Circuit	MAX707CPA	Maxim	
9	1	U15	FPGA Programmed as PCI Bus Arbiter	QL8x12B-0PL68C	Quicklogic	
10	3	U16-U18	FPGA Programmed as LED Interface	QL8x12B-XPL68C	Quicklogic	
11	1	U19	FPGA Programmed as DRAM Controller	QL12x16B-2PL84C	Quicklogic	
12	1	U20	FPGA Programmed as Misc Logic	QL12x16B-1PL84C	Quicklogic	
13	12	U21-U32	256Kx32 EDO DRAM	MT4C16270DJ-6	Micron	
14	1	U33	128Kx8 EPROM	AM27C010-150	AMD	
15	4	U34-U37	Octal Bus Transceiver	MC74ACT245DW	Motorola	
16	4	U38-U41	Octal Latch	SN74ABT573DW	Texas Instruments	
17	3	U42-U44	Octal Latch	MC74ACT573DW	Motorola	
18	4	U45-U48	Octal Register	MC74ACT574DW	Motorola	
19	1	U49	Dual UART	AM85C30-8J	AMD	
20	2	U50,U51	Dual RS-232 Trans-mitter/Receiver	MAX233CPP	Maxim	
21	1	U52	64x16 Serial EEPROM	93C46/SN	Microchip	
22	1	U53	5 V to 3.3 V Power Supply Controller	MAX767CAP	Maxim	
23	1	U54	Logic Level P-Channel MOSFET	IRF7202	International Rectifier	
24	1	U55	Dual N-Channel Power MOSFET	IRF7101	International Rectifier	
25	1	U56	Oscillator, 33 MHz	F6233-33.000	Fox	
26	3	U57-U59	Oscillator, 80 MHz	F6233-80.000	Fox	
27	3	U60-U62	PAL	PALC16L8-25PC	Cypress	

Table 72. Parts List (Sheet 2 of 3)

Item	Qty	Reference	Description	Mfg. Part Number	Manufacturer	Notes
28	6	U63-U68	1Kx18 FIFO	IDT72225LB25J	IDT	
29	12	U69-U80	RS-422 Differential Receiver	AM26LS32SC	AMD	
30	25	U81-U105	Quad XOR Gate	MC74AC86D	Motorola	
31	4	U106-U109	Hex Inverter	MC74AC04D	Motorola	
32	24	U110-U133	Common Mode Choke	PT3868	Valor	
33	6	U134-U139	10BASE-T Filter Module	FL1057-002	Valor	
34	2	U140,U141	4 Position Shunt	TS04	Augat	1
35	208	C1-C208	Capacitor, 0.1 µF	C1206C104K5RAC	Kemet	
36	42	C300-C341	Capacitor, 22 µF, 20 V	T491D226M020AS	Kemet	
37	1	C448	Capacitor, 0.01 µF	C1206C103K5RAC	Kemet	
38	1	C449	Capacitor, 220 µF, 6.3 V	595D227X06R3D2T	Sprague	
39	48	C450-C497	Capacitor, 150 pF	C1206C151K5RAC	Kemet	
40	3	CR1-CR3	LED, Red, Top View	550-1104	Dialight	
41	56	CR4-CR56	LED, Green, Top View	550-1304	Dialight	
42	3	CR60-CR62	LED, Red, Right Angle	550-1106	Dialight	
43	24	CR68-CR91	LED, Dual Green, Right Angle	552-0922	Dialight	
44	200	R1-R200	Resistor, 10 KΩ, 5%	CRCW1206-103J	Dale	
45	1	R299	Resistor, 100 Ω, 5%	CRCW1206-101J	Dale	
46	78	R300-R377	Resistor, 22 Ω, 5%	CRCW1206-220J	Dale	
47	13	R400-R508	Resistor, 1.0 KΩ, 5%	CRCW1206-102J	Dale	
48	121	R600-R720	Resistor, 820 Ω, 5%	CRCW1206-821J	Dale	
49	24	R750-R773	Resistor, 22 KΩ, 5%	CRCW1206-223J	Dale	
50	25	R800-R824	Resistor, 2.7 KΩ, 5%	CRCW1206-272J	Dale	
51	3	R946-R948	Resistor, 0 Ω, 5%	CRCW1206-000J	Dale	
52	1	R949	Resistor, 25 mΩ	LR2010-01-R025	IRC	
53	48	R950-R997	Resistor, 5.1 Ω	CRCW1206-050J	Dale	2
54	1	FB1	Ferrite Bead	2743021446	Fair Rite	
55	3	J1-J3	8 Port RJ45 Module	557573-1	AMP	
56	2	J4, J5	Modular Jack	555165-1	AMP	
57	1	J6	PCI Connector	646255-1	AMP	
58	2	J7, J8	Socket, 72 Pin SIMM	822134-3	AMP	
59	1	L1	6.8 µH	DT3316P-682	Coilcraft	
60	24	P1-P24	1x3 Header	103327-3	AMP	

Table 72. Parts List (Sheet 3 of 3)

Item	Qty	Reference	Description	Mfg. Part Number	Manufacturer	Notes
61	1	P25	Input Power Connector	26-48-1065	Molex	
62	1	P26	PCI Input Power Connector	26-48-1085	Molex	
63	1	SW1	8 Position DIP Switch	GDH08S	Augat	
64	1	SW2	Pushbutton Switch	FSM4JSMA	Augat	
65	12	TP1-TP6	Test Point	5002	Keystone	
66	2	VR1-VR2	Schottky Diode	PRLL5817	Phillips	
67	1	Y1	Crystal, 14.318 MHz	FOXS/143-20	Fox Electronics	

NOTES:

1. Only one of the shunts can be installed. When a 80960Cx processor is installed in U112, U130 should be installed. When an 80960Hx processor is installed, U131 should be installed. If no processor is installed in U112, either shunt may be installed.
2. These resistors are not installed.