INTEL/311847-001

Touchstone Project DELTA Numeric Node Special Technical Report

Milestone Event Q4

Paul Close
Anthony Zilka

Intel Corporation

December 28, 1989

Defense Advanced Research Projects Agency
Information Science and Technology Office

~~Limited Rights Legend~~

Contract No. MDA972-89-C-0034

Contractor: Intel Corporation

Case #90-2830

Limited rights are not subject to an expiration date.

The restrictions governing the use and disclosure of technical data marked with this legend are set forth in the definition of "limited rights" in paragraph (a)(15) of the clause at 252.277-7013 of the contract listed above.

This legend, together with the indications of the portions of this data which are subject to limited rights, shall be included on any reproduction hereof which includes any part of the portions subject to such limitations. This technical data will remain subject to limited rights only so long as it remains "unpublished" as defined in paragraph (a) above.

INTEL/311847-001

DELTA Numeric Node Special Technical Report

Milestone Event Q4

Prepared By:

Paul Close
Anthony Zilka

STATEMENT "A" per Karen Schroder
DARPA Library, 1400 Wilson Blvd.
Arlington, VA 22209-2308
TELECON 7/25/90 VG

Intel Corporation
Intel Scientific Computers
15201 NW Greenbrier Parkway
Beaverton, OR 97006

December 28, 1989

Sponsored by

Defense Advanced Research Projects Agency
Information Science and Technology Office
Research in Concurrent Computer Systems
ARPA Order No. 6402, 6402-1; Program Code No. 8E20 & 9E20
Issued by DARPA/CMO under Contract #MDA972-89-C-0034

DISCLAIMER

Abstract

The aggregate performance of any parallel computer system is directly related to the performance of the system's individual processors. The DELTA numeric node prototype addresses this issue by utilizing the latest in high performance 'off the shelf' microprocessor technology.

The DELTA numeric node, named RX-1, is based on the Intel 80860 microprocessor. This single component provides an unprecedented amount of hardware integration which in turns allows a very high performance processing node to be packaged on a single printed circuit board.

The RX-1 node design had two simple goals:

1. Provide an i860 node for the DELTA prototype. The DELTA prototype is a 2D mesh-organized multicomputer also being developed under the DARPA sponsored Touchstone program; AND
2. Through the process of designing both hardware and software for the i860 node, gain valuable experience which could then be applied to a second generation effort.

The RX-1 node has been designed, built and tested. The major features are: 40MHz system clock; On board 8 M-byte DRAM array expandable to 32 M-bytes with 4 M-bit DRAM technology; A general purpose expansion connector for attaching additional DRAM, second level cache, frame buffer or any other imagined appendage; A FIFO based interface to the internal system network; A modular approach for attaching the internal system message passing circuitry; A low speed high reliability serial channel for system boot and diagnostics; I/O support for performance analysis instrumentation; and a 56-bit counter with 100ns granularity for system timing.

This report describes the implementation strategies used to develop the RX-1, the functions it provides, and finally what has been learned in terms of suggested improvements for a second generation node.

Keywords: PAL (Programmable Array Logic), (RISC)

Contents

Figures

1. Summary

New microprocessors like the Intel i860 make it possible for individual processing nodes of scalable parallel computers to attain unprecedented performance levels. The DELTA numeric node, also named the RX-1, is such a node.

The RX-1 was developed as a vehicle for gaining experience with the i860 from both the hardware and software perspectives, and for providing such a node as part of the DELTA prototype.

The physical description of the RX-1 node is: one circuit board containing the i860, a boot PROM, fourteen PAL (Programmable Array Logic) devices, a number of discrete TTL and CMOS components, a number of CMOS DRAMs for data and parity, and the mounting hardware for the routing module.

The implementation strategy was to make a reasonably powerful i860 based node bounded by the schedule constraints of the Touchstone program. It was packaged in the same form factor as an Intel iPSC/2 node and in fact can be used in the iPSC/2. This was done to allow the RX-1 node to be tested and made available to users in a system configuration well before the DELTA prototype is scheduled.

Performance of the RX-1 node with a system clock of 40MHz has been measured to be 8.76 double precision MFLOPs on a 100 X 100 Linpack and a harmonic mean of 4.45 double precision MFLOPs on the Livermoore Loops.

A number of potential improvements have become apparent as a result of developing and using the DELTA numeric node. The most significant is the architecture for an ASIC for supporting the i860 data bus with higher integration, higher performance, and greatly improved reliability through ECC (Error Checking and Correction) circuitry.

2. Introduction

Highly parallel systems have demonstrated enormous computational power as they scale to a large number of processing nodes. The performance of these systems is roughly equal to multiplying the performance of a single node by the number of those elements in a system. Because of this, the individual node performance is still an interesting variable.

The new generation of super microprocessors led by the Intel i860 provide significant performance in a very small physical space. This lends itself well to parallel systems where processing nodes must be simple and reliable if a large number of them are to be successfully packaged together.

The intent of this node research activity was to incorporate the i860 into a message-passing multicomputer, namely the DELTA prototype system. The DELTA prototype is being developed by Intel under the DARPA Touchstone program. The result of this effort is the RX-1, a single 40MHz i860-based processing node. Refer to Figure 1. With some limitations the node can be thought of as a stand-alone computer with specialized I/O and a large DRAM-based memory.

This paper describes the implementation strategies used to develop the RX-1, the functions provided by it, and what has been learned by its design in terms of suggested improvements for a second generation i860 node.

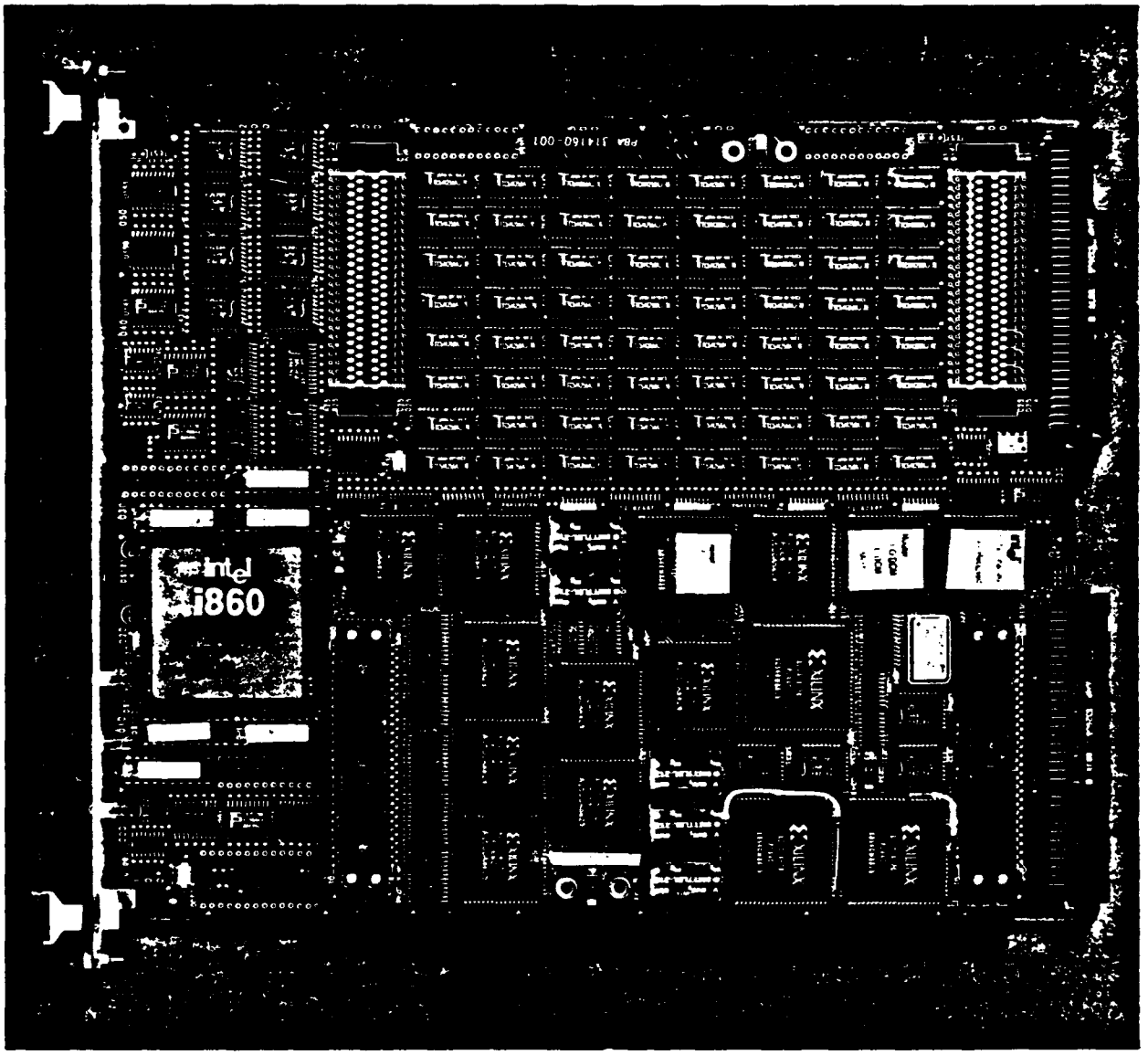


Figure 1
Photograph of DELTA Numeric Node

3. Implementation Strategy

This chapter discusses the strategies adopted for the definition and design of the RX-1 node.

3.1 Constraints

We had several constraints on the RX-1 design. One was the form factor of the node. This was dictated by the DELTA prototype in which we planned to have the same form factor as the iPSC/2. Another constraint was the schedule. Because the time frame of the DELTA prototype was later than the RX-1, an implementation strategy was chosen that would allow the RX-1 node to be used in the existing Intel iPSC/2 hypercube system. This allowed software to be developed for the node in parallel with the DELTA hardware development. Practically, this meant that the interface to the system's internal message-passing network would be accomplished by attaching an add-on daughter card with same form factor and interface as that used for the iPSC/2.

Two other constraints on this effort also influenced the implementation. The first was the base 40MHz clock rate of the i860. This meant that 'low technology' prototyping techniques like wire wrapping were not an option. The second constraint which is not appreciated outside the relatively new parallel computing industry is that a prototype parallel system involves the production of a relatively large number of the individual nodes. This meant that the RX-1 implementation had to accommodate large scale manufacturing concerns. Both these issues led us to treat the prototype just as carefully and completely as any product design headed for manufacturing.

3.2 Design Conceptualization

The design was conceptualized by engineers at both Intel Scientific Computers and the i860 group. A feature set was established by iSC taking into account tradeoffs of power budget, available physical board space, and development time. An architectural specification was written describing the node in detail so the hardware, diagnostics, and software engineers all had the same reference point.

3.3 Circuit Design

The circuit design of the RX-1 was entered into schematic form using the FutureNet schematic editor from the Data IO corporation. The PALs were designed using the ABLE programmable array logic tools also from Data IO.

3.4 Simulation

During functional simulation only the PALs were simulated. This was accomplished via the ABEL software from the Data IO corporation. This was the case because at the time no board simulation tools were available at iSC. This was acceptable because the logic of the node is fairly simple; most of the complexity is internal to the i860.

We were very concerned about the physical design issues. The base clock frequency of the i860 is a respectable 40MHz. An example of how this compares to previous technology can be made with the address and data busses. The address and data lines of the i860 can toggle at half the clock rate, so with a 40MHz clock, some ninety six signals toggle at a frequency greater than the system clock of the 16MHz 80386 iPSC/2 node. Needless to say, most signals in the RX-1 design were considered critical. Some were considered ultra-critical, specifically the clock signals and several control signals from the i860.

All ultra-critical signals were electrically simulated with PC SPICE. The results of this simulation was used to direct the physical layout of these traces during the printed circuit board design stage. With SPICE, we looked for effects like signal ringing and propagation delays. The 40MHz system clock on the node had too many destinations to be serviced by only one signal. Therefore the clock is buffered into eight separate clock traces, each running separate but interrelated parts of the design. Our concerns were that these clocks be absolutely clean, low in propagation delay, and maintain a minimum of skew between each other. Our design goal was that the clocks have no more than 2ns of skew. The SPICE simulations helped us orient the components, determine required trace routings, and select the proper termination circuitry in such a way as to optimize the design. As it turned out this was a very successful effort, in fact we have measured no more than 0.75ns of skew between clock lines and all signals are near 'picture perfect'.

The extensive circuit simulation resulted in a 120mil thick twelve-layer circuit board with components oriented for minimum trace lengths. All critical signals (most signals on the board) are dual stripline to maintain uniform impedance. The board is built to achieve impedance above 55ohms which was found to be a threshold above which the CMOS and TTL technologies used displayed the best transmission line characteristics.

3.5 Printed Circuit Board Design

The printed circuit board design was accomplished on a DAISY board layout station. This process was iterative with initial component placement and layout of the ultra-critical traces done first. The geometries described by this initial layout were then used to describe the transmission line models which in turn were used in SPICE electrical simulation. The results of the simulation were used to modify the layout, which in turn fed the simulator. This process continued until the board's electrical characteristic's were acceptable.

3.6 Board Fabrication and Manufacture

The bare boards were fabricated at Tektronix, in Forest Grove, Oregon. The actual board build (soldering the components) was done at the Intel board manufacturing facility in Hillsboro, Oregon.

4. High-Level Description

The purpose of this section is to describe the RX-1 node from a high level view.

4.1 Nomenclature

X means don't care

stands for active low signals

Signals are listed in the appendix as they appear in the schematics.

4.2 Features

The following list describes the basic features of the RX-1 node. Later sections detail these features more completely.

1. 40MHz i860 processor.
2. 8M-bytes of onboard 70ns DRAM, expandable to 32M-byte with 4M-bit DRAMs.
3. iPSC/2 node form factor, including support for both the DCM hypercube routing module and the mesh router interface module.
4. Bi-directional memory mapped FIFO interface to the routing module.
5. 64K-byte EPROM for booting.
6. 52-bit i860 readable timer with 100 ns granularity.
7. Control and status ports, memory mapped.
8. Interrupt controller maskable by bits in the control register.
9. Expansion connector that makes both address and data buses, as well as i860 control signals, available to a daughter board. By having all control signals, any potential resident of the i860 bus can be designed, including memory subsystems and network interfaces.

4.2.1 Performance

The RX-1 Node board runs at 40MHz with one wait state on every write cycle. Parity generation is the worst case path which requires the wait state.

Compute Performance

The i860 is a high performance numeric processor. It also is a good RISC machine. At 40MHz, absolute peak floating point numbers are 80SP/40DP, and a MIPs rating of 32 is fair.

The following are measured benchmarks. They were performed with a minimal amount of optimization and very young compilers so the results should be considered conservative.

Livermore loops (24 loop version. All numbers are in MFLOPs):

Scalar (Full Compiler Optimization)

Mean Vector Length:	471	90	19
Range:	1.47 to 8.76	1.47 to 9.96	1.47 to 9.86
Arith:	4.46	4.85	5.06
Median:	4.39	4.27	5.01
Harmon:	3.56	3.89	3.90

(24 loop version vectorized with VAST. All numbers are in MFLOPs):

Vector (Full Compiler Optimization)

Mean Vector Length:	471	90	19
Range:	1.45 to 16.93	1.24 to 16.56	0.64 to 9.80
Arith:	7.15	6.34	4.38
Median:	5.22	4.63	4.26
Harmon:	4.45	4.22	2.66

Linpack 100 x 100 Double precision:

Class	
Compiled BLAS	4.18 Mflops
Vasted	N.A.
Coded BLAS	8.5 Mflops

Whetstones:

Single Precision	10682 whetstones
Double Precision	14205 whetstones

See *Access Times* for cycle information.

Message Passing Performance

The compute speed of the i860 coupled with the FIFO-based interface to the message network result in significantly lower message latency when compared with the 80386 based node. The hardware latency to start a message is very low, essentially the access time of the FIFO and the fall through time. This is roughly 10 i860 clocks or 250ns.

The bandwidth has two components:

- i860 to FIFOs:. 30Mb/S is a good estimate.
- FIFO to Network interface: 80MB/s however this depends greatly on the routing module.

Refer to Message Passing Interface for more detailed bandwidth performance numbers.

4.2.2 Reliability

The RX-1 node will operate reliably over:

Temp. 0 - 60° C

Volt. 4.75-5.25 VDC

The following table lists estimates of MTBF (Mean Time Between Failure) for the RX-1 node with 8M-bytes DRAM.

RX-1 Node MTBF at 55° C

SOFT ERRORS	HARD ERRORS	TOTAL ERRORS
69.4K hours	54.7Khours	30.6Khours

4.2.3 Diagnosability

The RX-1 node has several features to enhance diagnostic capability. These are important features which make the task of testing and repairing the node much easier.

- Refresh defeat bit (REFDEF) in control register. This bit turns off refresh, and can be used to determine if refresh logic is working if, when defeated, DRAM data decays.
- Parity defeat bit (FRCPAR) in control register. When active this bit disables accesses to the parity RAMs. In this mode what was written into the parity RAMs on previous memory writes will remain even if other writes to the same memory occurs. This allows data and associated parity to be written (in normal mode), then new data can be written in force parity mode. Later reads should then cause parity errors.
- FIFO wrap mode. This mode, selected by a bit in the control register, causes outgoing FIFO data to fall through the transmit FIFO and written into the receive FIFO, where it can be read by the i860 to verify.

4.2.4 Manufacturability

The RX-1 board meets all Intel design for manufacturing rules except for one: some components are oriented in opposing directions. This is done to shorten clock and other critical signal traces for optimum performance.

4.2.5 Physical & Power

The form factor is the same as existing iPSC/2 node boards, which is 11.024" by 9.18". The node current has been measured at 5.6A at +5V (28 watts) without a routing module. The maximum power allowance for the daughter card is approximately 10 watts. The node board is predominately surface mount, with active components on the primary side only. The processor, EPROM, PALS, and delay lines are through hole components.

5. Architectural Discussion

As can be seen in Figure 2, the RX-1 node is made up of just a few major functional blocks. The DRAM block contains the DRAM controller which is a state machine built with high speed programmable devices, the address latches, the data buffers, the parity generation and check circuits, and the data and parity DRAM. The FIFO block contains the FIFOs and the corresponding control logic. The FIFOs supply the buffering function between the node and the communication circuitry which is attached via the communication module. The local control block contains a variety of items which support system operation. These items are a 56-bit counter for system timing, a status port for reading the various hardware status signals, the control port whose outputs configure and control the node, the serial port which is used as a diagnostic channel as well as a timing unit, and last but not least, the boot PROM which contains the lowest level program that executes upon node power up or reset.

The following sections describe each function in greater detail.

5.1 Memory Map

The i860 memory space is divided into sixteen 256M-byte blocks as shown in Figure 3. This relative coarse granularity simplified address decode circuitry. The memory map decode logic will not allow any hardware contention which could cause damage.

The following subsections discuss reasons for this particular memory map, as well as the functions.

5.1.1 PROM Space

The i860 comes out of reset in CS8 mode, which allows it to boot from an 8-bit device. Reset clears all control register bits, including BOOT#, which maps the on board EPROM into the top half block of memory. The 64Kbytes of EPROM may be accessed in any 64K section within this space (F800000-FFFFFFFFhex).

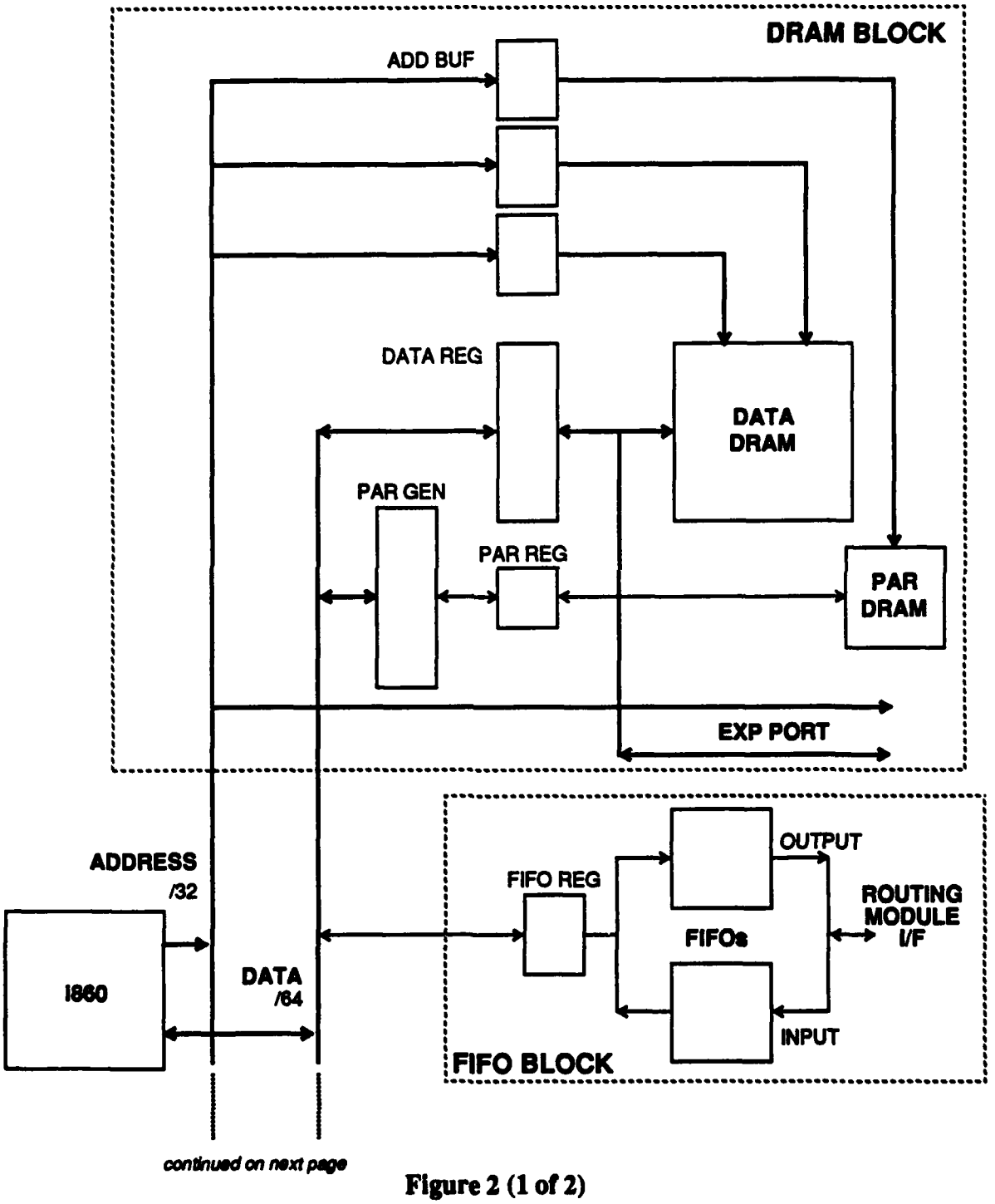
Only instructions (not data) can be fetched from the EPROM, so data must come from the Serial channel or be implicit in the boot code (i.e.: no data tables in the EPROM). This is an i860 implementation limitation.

Once enough boot code has been loaded into RAM, the BOOT# bit is set to 1, which maps the EPROM out of memory, until the next reset or power up.

For details on CS8 mode, see the i860 data sheet.

5.1.2 On Board DRAM Space

On board DRAM is placed at the top of memory to support i860 TRAPs. It can be selected in the top three blocks of memory in different ways:



continued on next page

Figure 2 (1 of 2)
RX-1 Node Block Diagram

continued from previous page

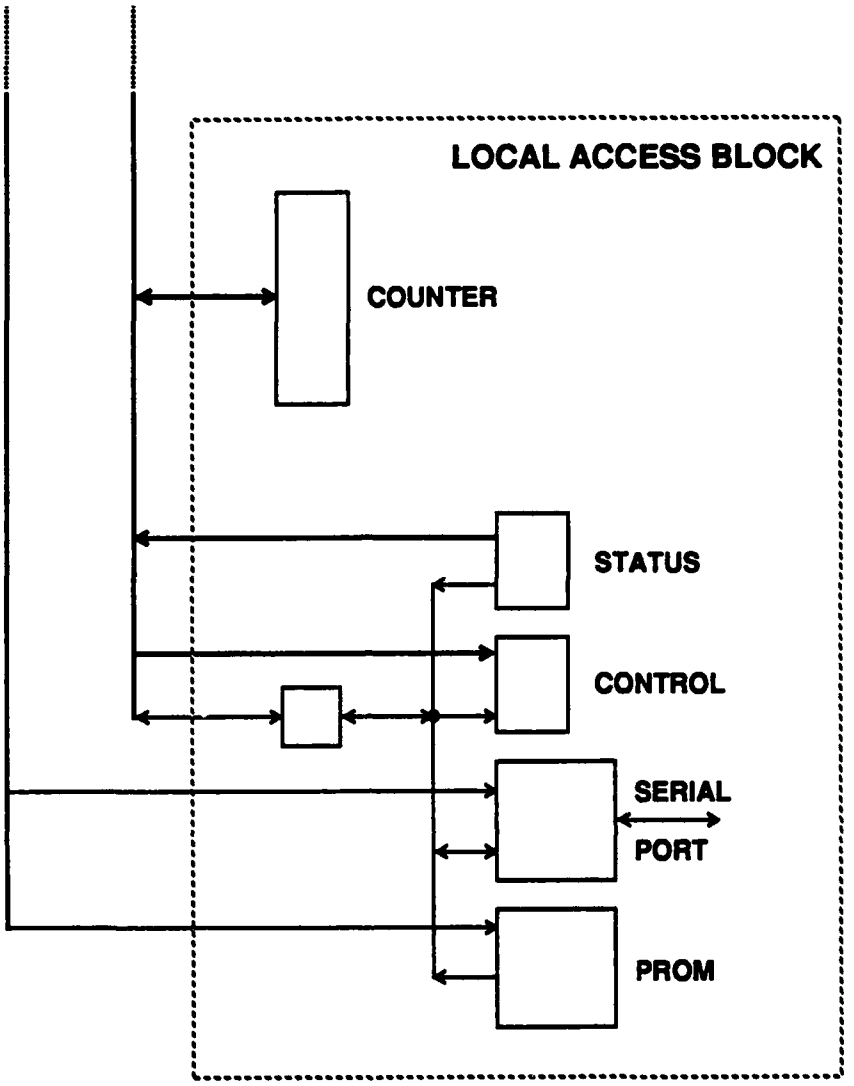


Figure 2 (2 of 2)
RX-1 Node Block Diagram

	ON BOARD DRAM (BOOT/ =0) PROM (BOOT/ =0)
FXXXXXXX	DRAM
EXXXXXXX	ON BOARD DRAM
DXXXXXXX	ON BOARD DRAM NO CACHE.
CXXXXXXX	NOT USED
BXXXXXXX	NOT USED
AXXXXXXX	NOT USED
9XXXXXXX	NOT USED
8XXXXXXX	NOT USED
7XXXXXXX	NOT USED
6XXXXXXX	SERIAL CHANNEL
5XXXXXXX	COUNTER (READ) PA PORT (WRITE)
4XXXXXXX	STATUS (READ) CONTROL (WRITE)
3XXXXXXX	FIFO (EOD) FIFO (NOT EOD)
2XXXXXXX	NOT USED
1XXXXXXX	NOT USED
0XXXXXXX	NOT USED

Figure 3
i860 Node Memory Map

1. The top half block (F8000000hex and above) is EPROM only when the BOOT# bit is active and DRAM when it is not. This allows accesses to EPROM when in CS8 mode. The lower half block (F0XXXXXX to F7XXXXXX) is always DRAM.
2. The next block (EXXXXXXXhex) is always on-board DRAM.
3. The third block (DXXXXXXXhex) is also always DRAM, and is provided to disable caching for debug with logic analyzer type instruments. Accesses within this space disables KEN# (cache enable) to the i860, forcing accesses not to be cached. The KEN# signal is NOT piped with bus transactions. Bus transactions that cross this memory space boundary must be done very carefully to ensure proper cache operation.

DRAM exists modulo 8M-bytes (or 32M-bytes with 4M-bit DRAMs) in each block (Example: EX000000-EXFFFFFFhex).

5.1.3 Daughter Board DRAM

Off-board memory can take any unused memory space. See the Expansion Port chapter for more details.

5.1.4 Ports

The Serial port, and the A and B status/control ports are called local accesses. Each port uses an entire 256M-byte block. See sections on these functions for more details.

5.1.5 Network FIFO

The network FIFO also uses a 256M-byte block. For FIFO writes, the block is split into two equal sized sections. Writes to the lower half are for data transfers within the message. Writes to the upper half are for the last 32 bits of a message, or the EOD (End Of Data) transfer. This is used to delimit, or frame, the message. Reads are not differentiated within the 256M-byte block.

See the FIFO section for more details.

5.2 Memory System

The i860 has on-chip cache for both instruction and data. No external cache is supplied. The caching function of the i860 can be disabled by utilizing the non-cached address space. This feature of the RX-1 node should be used with caution since the design simply decodes the cache disable function from the address bits. Since the cache disable function in not piped with the data, care must be taken when crossing boundaries between cached and non-cached memory.

The DRAM is not interleaved because of the increased hardware, additional power, and added design time.

5.2.1 On Board DRAM

The RAM is configured logically as four 2M-byte banks (8M-byte banks with 4M-bit DRAM). Any combination of banks may be installed, although the i860 requires DRAM in certain sections of the address space, especially for TRAPs. The DCTLA PAL must be programmed to support the combination of installed banks.

Accesses spanning banks pay the not 'nearest neighbor' penalty (See Access Times). The main reason for the bank approach versus 'stacking' the columns is that other approaches (such as RAS to all banks, but CAS only to the bank under access), is power. RAS active will bring DRAM out of stand by mode. Each bank of active RAM uses (16 X 65mA) 1040mA (worst case), versus (16 X 2mA)= 32mA when inactive.

Access times have two components:

1. Starting up pipelined accesses: read after write, write after read, or access outside of the last column.
2. Pipelined accesses. Pipelining in this case means two or more like (reads or writes) accesses to addresses within a DRAM column. Start up depends on the previous access. If starting up a write, the read pipe must first be cleared. As many as 11 clocks may be necessary for the first access (see Access Times). Subsequent accesses, as long as they fall into the column, will be 2 clocks long (0 wait state) for all reads and 3 clocks (one wait state) for writes.

5.2.2 Parity

Even parity is generated and checked on all DRAM accesses. A parity bit exists for each byte of the path (8 bits).

Parity RAMs are 8, 1M X 1 bit DRAMs, which are adequate for 8M-bytes of on board RAM and 8, 4M X 1 DRAMs for 32M bytes.

5.2.3 DRAM Daughter Card

Connectors are provided for a daughter card. The electrical restrictions of the daughter card is described in the chapter on the Expansion Port.

5.2.4 Upgrade To 4M-bit DRAMs

The board is designed with the option of using 4M-bit DRAMs for both data and parity. This change is accomplished simply by installing the larger DRAMS, changing the DCTLA PAL to reflect the change in address space of the banks and reconfiguring jumper resistors for the DRAM address buffers. This would expand on board DRAM to a maximum of 32M bytes. Only one bank could be installed if desired, which would provide 8M bytes, but with increased MTBF.

5.3 Message Passing Interface

The message passing interface connects the node to the routing logic through which messages are passed between nodes. On the RX-1, the i860 transfers data via FIFOs to the message passing logic. This interface is designed such that the routing logic is attached via a daughter board. The daughter board for the iPSC/2 hypercube (GAMMA prototype) is the Direct Connect Module (DCM). The board for the DELTA prototype will be the Mesh Routing Interface (MRI).

The reasons behind this approach are:

- This approach automatically handles cache coherency with the i860, removing yet another degradation to message latency.
- A significant additional message latency incurred for setting up a DMA channel. This is eliminated with FIFOs.

Drawbacks with the FIFO approach are:

- FIFOs are expensive and take up board space.
- The FIFOs may be a limiting factor in network bandwidth of future machines.
- More processor cycles are used to transfer data than with a DMA approach because the i860 must read the data into an intermediate register, then write out to the destination, where a DMA approach would (should) perform single cycle transfers.

Because the i860 has Hold/Hold Ack, this design could easily be modified for DMA type transfers, but does not support this feature directly.

5.3.1 The FIFO Interface

The interface is bi-directional FIFOs, mapped into the i860 memory space. The FIFOs are seen as physical memory within the memory space. Data must be read from memory, DRAM or cache, then written to the FIFO. The bandwidth depends on the block size read out of DRAM into the i860 and whether the data was in cache.

The RX-1 uses uni-directional FIFOs. Because of the hardware required, only 32-bit transfers from the i860 are supported. This provides 4K bytes of FIFO each direction.

Refer to Figure 4 for a block diagram of this approach.

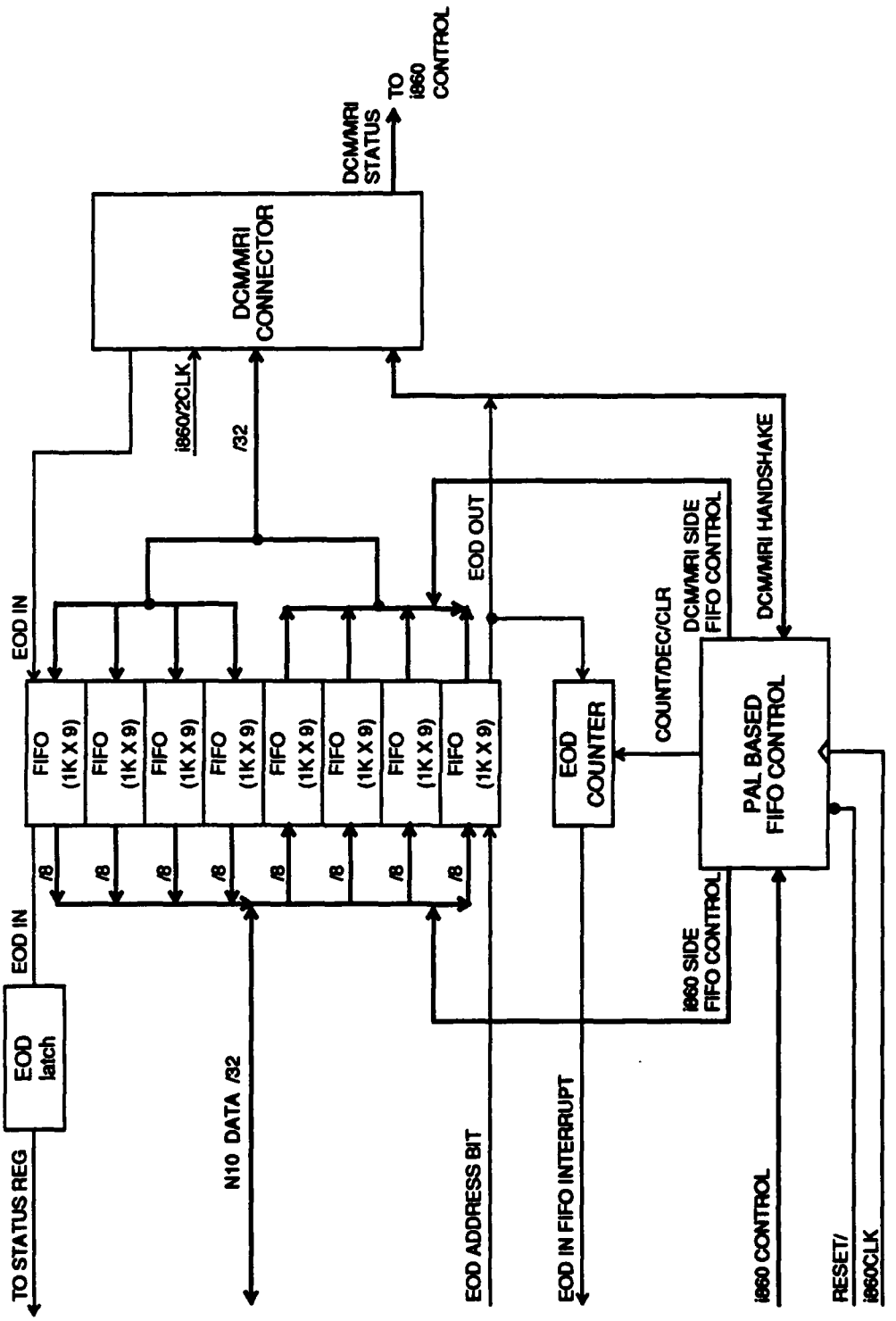


Figure 4
FIFO Block Diagram

The data buffers shown in Figure 4 are for latching data to and from the FIFO, as those cycles are pipelined.

Message framing, or EOD (End of Data), is handled with a special address space called the 'EOD space' (See Memory Map). FIFO writes within this address space access the FIFOs, but have the side-effect of setting outgoing EOD bits.

An EOD counter is provided to inform the i860 that the end of a message is in the FIFO. The counter is incremented whenever an EOD comes in from the interface. It is decremented when the i860 reads EOD data from the FIFO.

A non-zero count indicates one or more EODs are in the FIFO. An EOD in FIFO bit (EODINF) indicating the EOD counter is non-zero is presented to the i860 both as a maskable interrupt, and as a bit in the status port. If EODINF is logic high, one or more EODs are in the receive FIFO.

When the last 32-bit word of a message is read out of the receive FIFO, and an EOD has followed it, as it would during correct operation, the EOD will be latched and supplied to the status port (EODLAST#). This way the software can make sure it reads an EOD word. It will remain set until another, non EOD access occurs.

5.3.2 DCM Interface

The following is a list of the handshake signals between the RX-1 node and DCM:

i860	Function
REOD#	EOD from receiver
XEOD#	EOD to transmitter
NETRRDY#	Receiver has 32 bit word
NETXRDY#	Transmitter can accept word
RACK#	Enable word from receiver
XACK#	Enable word to transmitter

The receiver and transmitter are the devices by those names (or are synonymous) on the DCM. Note that the programming for these devices is different than that of a DCM on the 386 node board.

The clock supplied to the DCM is synchronous with both the i860 and the FIFO control state machine, so this simplified and faster handshake is possible.

This design has the RX-1 FIFO state machine controlling the interface. This state machine is designed such that by modifying one PAL, the DCM, or other module, can drive the interface, with the FIFO state machine acting as a slave. This would allow network/FIFO access asynchronous to the i860, which could enhance performance.

If both receive and transmit operations are possible (FIFOs and DCM have space/data available), accesses will ping pong between the two operations.

5.3.3 DELTA Mesh Routing Interface

The Mesh Routing Interface (MRI) is a module which replaces the DCM to form a portion of the circuitry between the node and the mesh routing network. The MRI will be designed and constructed as part of the DELTA prototype effort and be fully documented in the associated DARPA technical report.

Because the DELTA machine will have greater than an order of magnitude higher message passing bandwidth than the iPSC/2, the demand on this interface will be greater. Flexibility is architected into the control of the node to MRI interface as that control is provided by one programmable logic device. To modify the protocol of the interface, only this socketed device needs to be replaced.

5.3.4 Message Passing Performance

Hardware Latency

Because the FIFOs used in interface have zero fall through time, the hardware latency from node to routing network is effectively the bandwidth of the interface.

i860 to FIFO Bandwidth

Bandwidth is affected by the block transfer size, because starting up the DRAM pipeline costs cycles, and the larger the block size, the more those cycles can be amortized over transfers. For this analysis, an 8-word (64-byte) block size is assumed, and that the instruction execution is transparent.

i860 bandwidth for reading and writing the FIFOs is different. The FIFO access time (25ns) combined with the pre-charge time for the FIFOs costs a clock on the read cycles. Data pipelining is used on the write cycles resulting in 1 wait state to start out, then 0 wait states after that.

With the above in mind, the i860/FIFO bandwidth is:

i860 Writes to FIFO	
4 clocks to start read pipe	4
2 clocks per word after that	18
2 clocks for DRAM access to clear	2
5 clocks first 32 bits into FIFO	5
2 clocks per 32 bits after that	30
1 clocks for a conditional branch	1
Total Clocks/64 bytes	60
	~ 42MB/sec

i860 Reads from FIFO	
5 clocks for first 32 bits from FIFO	5
3 clocks each 32 bits from FIFO	45
6 clocks first word to DRAM	6
3 clocks per word after that	21
1 clocks for a conditional branch	1
Total Clocks/64 bytes	78
	~ 33MB/sec

5.4 High Resolution Counter

A high resolution 52-bit, counter is provided. It is synchronous to the i860 and clocked at 10Mhz. This ensures valid data for i860 accesses.

The counter is cleared by reset or power up, and may be written for diagnostics purposes. Writing is limited to Boot mode only (BOOT# bit active). Attempts to write in normal mode will result in a bus timeout. This is to ensure the counter can never be modified by user code. Counter addresses are:

Read Address	=	5XXXXXXX	BOOT#	=	X
Write Address	=	FXXXXXXX	BOOT#	=	0
D0 - D52	=	Counter bits	0-52	respectively	

Because of the size of the counter, it is assumed to never roll over (actual roll over time is 14.2 years). It may be read by the i860 as a double precision floating point number as the upper 12 bits will always be 0 (actually the upper eight bits are always zero, bits 53-56 are counter bits, but would take years to set, and therefore assumed to be 0). Access is accomplished with a 64 bit read of the memory space selecting the counter.

5.5 Boot PROM

A 64K-byte boot PROM is provided for boot function only. The i860 boots up in 8-bit mode. A control bit (BOOT#) is cleared at reset, and selects the PROM in the DRAM space (see memory map). Once this bit is set by the processor, PROM can no longer be accessed, until another reset.

The PROM is memory-mapped (see Memory Map) to address F8000000 - FFFFFFFF. Because of the slow access of the device, and hardware optimization, the access time can vary greatly from cycle to cycle. No software timing loops should be executed out of this device. Note that boot can be active even after CS8 mode is not.

5.6 Serial Port

The serial port is a memory-mapped Intel 82510. Because of the slow access of the device, and hardware optimization, access time can vary greatly from cycle to cycle.

The serial port is an 8-bit device, so only the lower eight bits of the data bus are used during accesses.

The 82510 has considerable resources which can be used by the processor, including a timer to count intervals. Refer to the Intel Micro Communications manual for details.

The 82510 is also used to capture some interrupts, namely the Serial interrupt, the network interrupt (on the DCM this is defined as checksum error), and the USM interrupt. They are mapped into the 82510 as follows:

82510 pin	function
CTS#	USM interrupt in
DCD#/ICLK	Network interrupt in
OUT2/X2	USM interrupt out

The 82510 (SERCLK) is clocked by the processor clock divided by 8 (40MHz: 5MHz).

Refer to the Intel Micro Communications manual for details of the 82510 and what the internal ports are. The 82510 has three address lines (A0-A2) which select the internal resources. This pins are connected to i860 address lines A3-A5 respectively. This causes accesses to the 82510 to be aligned on 64 bit word boundaries, not byte. It can be accessed anywhere in the 256Mbyte space of 6XXXXXXXH (60000000H-6FFFFFFFH). The 3 low order address bits, which in RX-1 space select byte enables, are don't care. The addressing algorithm is:

Upper 3 bytes of address (hex) 6XXXXXH

Lower byte of address (binary) X X A5 A4 A3 X X X

Where A2, A1, and A0 are the 82510 inputs.

Assuming don't cares (X) are 0, the address map is then:

i860 Address	82510 Address (A2 A1 A0)
60000000H	000
60000008H	001
60000010H	010
60000018H	011
60000020H	100
60000028H	101
60000030H	110
60000038H	111

5.7 Interrupt Logic

The i860 has only one interrupt input. On the RX-1 design, it is the logic OR of all possible interrupt signals. Software must read the status port to determine which event(s) occurred, and take the appropriate action.

The interrupt control consists of an interrupt mask (in the control register), the interrupts, which can be read in the status register, and an ORing function for combining the various interrupts. The interrupts are generally defined as:

1. Hardware Interrupt (parity error or bus timeout)
2. Serial interrupt (combination or serial port, network interrupt, and USM interrupt)
3. Receive FIFO interrupt (EOD in FIFO or receive FIFO half full)
4. Transmit FIFO interrupt (transmit FIFO half empty)
5. Expansion port interrupt

Each of these interrupts is maskable. The serial interrupt can be masked either via 82510 configuration register, or by a control register mask bit. All other interrupts are maskable via control register.

The mask bits are active low, meaning when low, the interrupt is masked. They come out of reset low.

5.8 Performance Port

The i860 node features a 5-bit performance port. Four bits are for data, the fifth is the strobe. This mates with iPSC/2 backplanes.

5.8.1 Performance Port Bit definition

D0: Performance bit 0

D1: Performance bit 1

D2: Performance bit 2

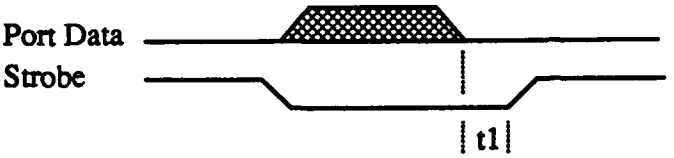
D3: Performance bit 3

D4: Performance strobe

D5 - D63: Not used

5.8.2 Performance Port Strobe Timing

The strobe bit is automatically generated when an access is made to the performance port. The strobe is two i860 clocks long, and has the following timing to the port outputs:



t1 = data valid to trailing strobe = 2 clocks - 23ns = 27ns

5.9 Control and Status registers

Both the control and status registers can be accessed anywhere within the 256M-byte space 4XXXXXXXXH. The control register is write only, and the status port is read only. Unfortunately board real estate dictates this limitation. All reads are non destructive.

All control register bits are cleared (reset to 0) upon reset or power up, and must be initialized.

The pinout has been selected based on data bus load balancing, and for ease of use by software. The following is a bit description of the ports:

Status Port

D00:	LXEMP#	- Transmit FIFO is empty (active low)
D01:	LXHF	- Transmit FIFO is half empty (active high)
D02:	XFULL	- Transmit FIFO full (active high)
D03:	REMP#	- Receive FIFO empty (active low)
D04:	LRHF	- Receive FIFO is half full (active high)
D05:	LRFULL	- Receive FIFO is full (active high)
D06:	EODINF	- An EOD is in the receive FIFO (active high)
D07:	EODLAST	- Last read from receive FIFO = EOD (active high)
D08 - D15:		Network status bits 0-7
D16 - D23:		Slot ID bits 0-7
D24:	BOOT#	- In reset boot mode (active low)
D25:	40/33#	- 1 = 40MHz, 0 = 33MHz
D26:	PARINT	-Parity interrupt (active high)
D27:	BUSTOUT	- Bus time out interrupt (active high)
D28:	RINT	- Receive FIFO interrupt (active high)
D29:	XINT	- Transmit FIFO interrupt (active high)
D30:	LEXPINT	- Expansion module interrupt (active high)
D31:	LSERINT	- Serial interrupt (active high)

D32:	XPR0#	- Expansion bit 0
D33:	XPR1#	- Expansion bit 1
D34:	XPR2#	- Expansion bit 2
D35:	XPR3#	- Expansion bit 3
D36 - D63:		- unused

Control Register

D00 - D39:		Unused
D40 - D47:		Network (DCM) control bits 0-7
D48:	GREENON	- green LED
D49:	REDON	- red LED
D50:	YELON	- yellow LED
D51:	REFDEF	- Defeat refresh logic, diagnostic bit
D52:	CLRBUS#	- Clear the bus timeout (active low)
D53:	CLRPAR#	- Clear the parity error (active low)
D54:	FRES#	- Reset FIFOs (active low)
D55:	FIFOWRAP#	- Diagnostic FIFO wrap mode (active low)
D56:	BOOT#	- Boot mode bit (active low)
D57:		- unused
D58:	HWMSK#	- Hardware error interrupt mask (active low)
D59:	EODMSK#	- EOD in receive FIFO interrupt mask (active low)
D60:	RMSK#	- Network receive interrupt mask (active low)
D61:	XMSK#	- Network transmit interrupt mask (active low)
D62:	EXPMSK#	- Expansion interrupt mask (active low)
D63:	SERMSK#	- Serial chip interrupt mask (active low)

5.10 Access Times

The following is a summary of access times for the various entities accessible by the i860. Zero wait state would be 2 clock cycles.

DRAM

Cycle	Last Cycle	Total Clock Cycles (2 = 0 wait state)
Far Read	idle	7
Far Read	Read	11
Far Read	Write	11
Near Read	Read	2

	Cycle	Last Cycle	Total Clock Cycles
	Near Read	Write	7
	Far Write	idle	5
	Far Write	Read	9
	Far Write	Write	9
	Near Write	Write	4
	Near Write	Read	5
PROM			
			37/99 note 1
Serial port			
	Read		37/99 note 1
	Write		37/99 note 1
Status port			
		-	5
Control port			
		-	5
Timer			
		-	5
Network FIFO			
	FIFO read	other	5
	FIFO read	FIFO read	3
	FIFO write	other	5
	FIFO write	FIFO write	2

Note 1: Access time is variable.

6. Theory Of Operations

This chapter describes the RX-1 node control hardware in greater detail, and makes reference to the PAL equations included in the Appendices.

6.1 Support Hardware

The iSC RX-1 node board consists of an i860 processor and functional support hardware. The support hardware is mapped into 256M-byte blocks of the i860 address space. These mapped functions are as follows:

1. DRAM
2. FIFO Interface
3. Serial Interface
4. PROM
5. Port A
6. Port B

Each of these functions is controlled by a state machine implemented with a PAL which controls the read and write accesses. Port A, B, PROM and Serial accesses are under the control of one state machine, and are referenced as 'local' accesses.

After RESET, each state machine is in an IDLE state waiting to be accessed. In the IDLE state the RDY and NA signals to the i860 are tri-stated by all state machines. When a specific memory space is accessed the four most significant address bits are decoded to select the mapped hardware. The combination of the decoded select and an active Latched ADS (LADS) signal initiates read or write sequences by the specified state machine. When the read or write operation completes, the state machine returns RDY to the i860 for subsequent operations.

An expansion board capability is also provided. The function of the expansion board is undefined.

The DRAM and FIFO hardware utilize the pipelined read capability of the 860. Instead of waiting until a READ operation completes and returning RDY to initiate a subsequent operation, NA is returned before the READ is completed and another operation is initiated early. RDY is returned with the READ data. This impacts the node design in several ways. First, a BSY signal must be generated by every state machine which can operate in a piped mode. All other state machines cannot start until piped state machines are not busy. This is to prevent the possibility of two READs to two different pieces of hardware returning at the same time. Second, if a state machine implements a deep READ pipe, it may not be at a

point to be recognized a subsequent operation after returning NA. In this case a cycle pending bit (CPEN) must be used. Fortunately, only the DRAM state machine has this problem. All i860 support hardware implement only one level of READ pipelining. Writes are not pipelined.

6.2 DRAM Control

The DRAM is controlled by the five PALs listed below:

PAL Name	Part Number	Description
DSTAT	314233	Five bit state machine
DCTLA	314234	RAS decoder
DCTLB	314235	Advanced clocked control signals
DCTLC	314236	Control signals
DCTLD	314237	Write Enable decoder

6.2.1 DRAM State Machine PAL (DSTAT)

The DSTAT state machine PAL implements the state diagram illustrated in Figure 5.

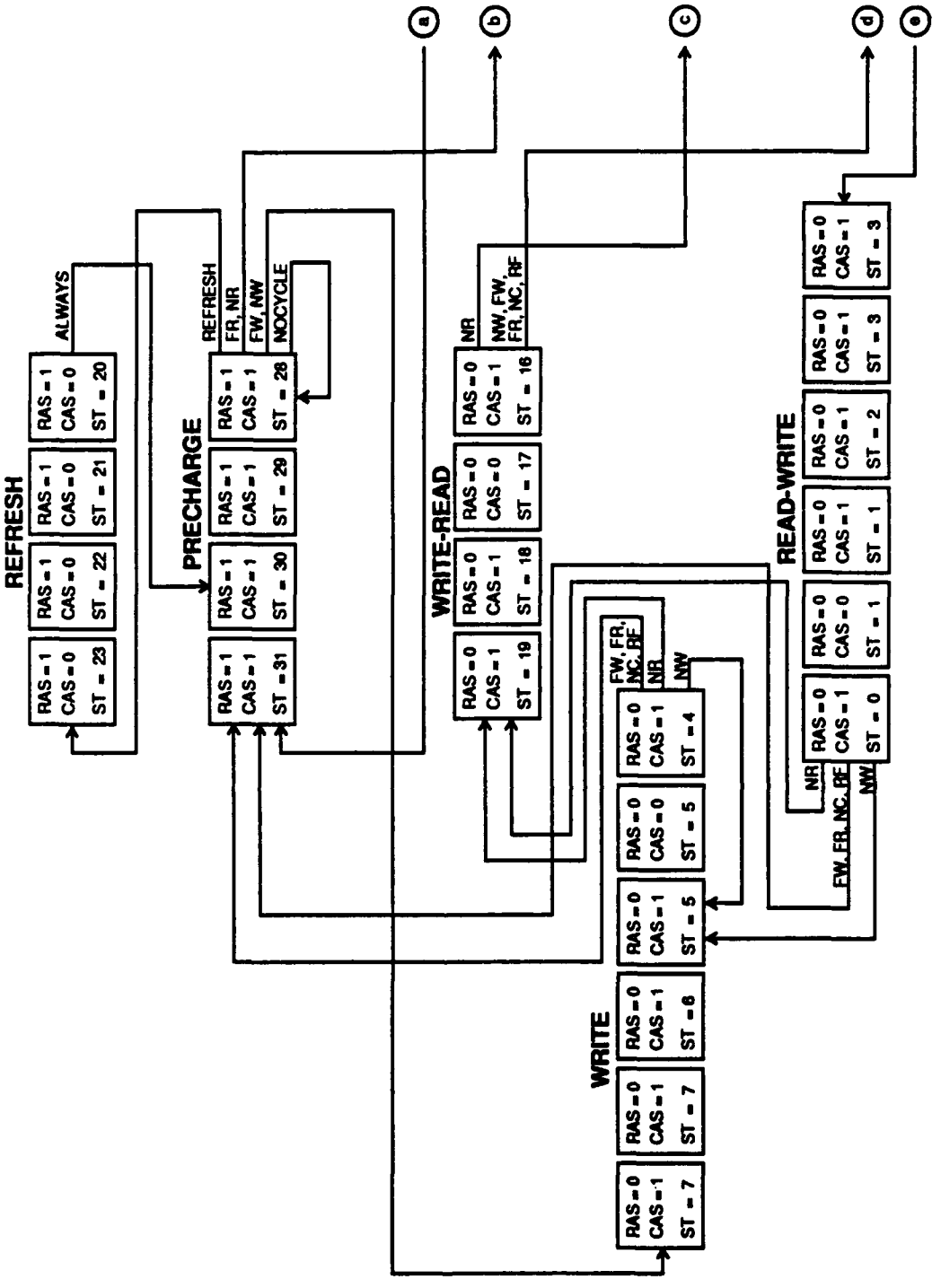


Figure 5 (1 of 2)
DRAM State Machine

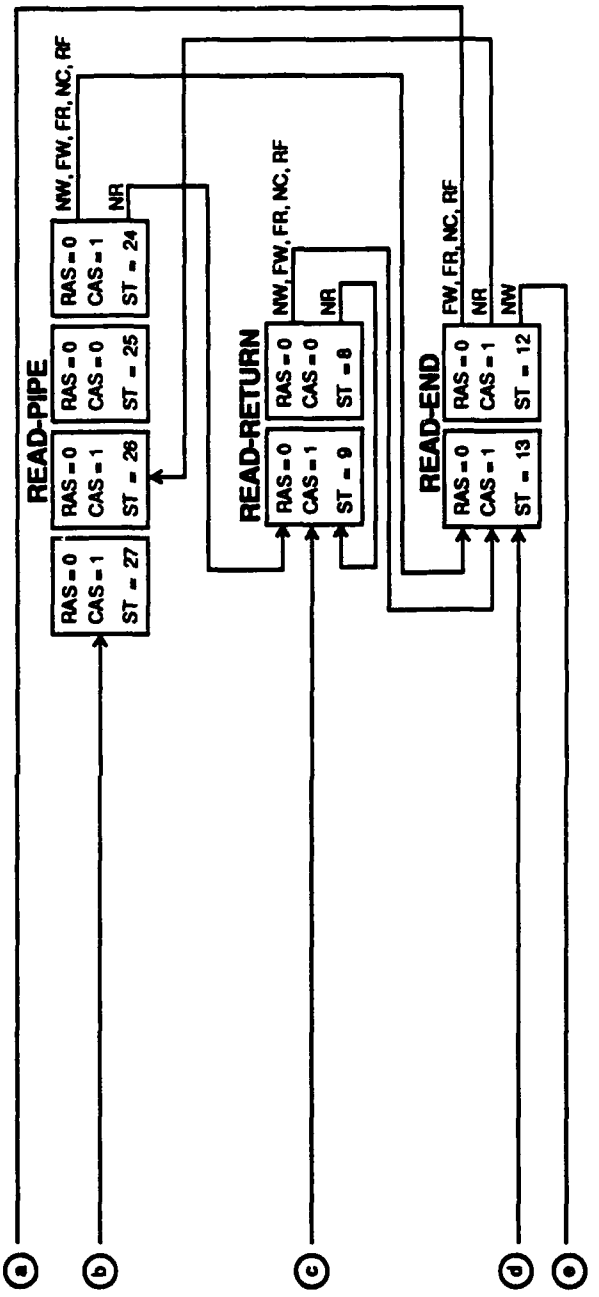


Figure 5 (2 of 2)
DRAM State Machine

Near reads or writes are defined as reads or writes to the same row as the previous operation, far reads or writes are reads or writes to different rows. A complete state machine description is in the DRAM State Machine Truth Tables given in Appendix A. A brief description of each sequence is given in the following sections. Note that some of the sequences could have been optimized to less than the existing number of states. Some states are redundant to optimize encoding of the state machine.

Precharge (PRx-1)

The Precharge sequence (PRx-1) implements a DRAM precharge cycle. In this sequence both CAS and RAS are disabled to satisfy the DRAM precharge time. This sequence is executed when transitioning from one DRAM access to another access on a different row or when going to a Refresh sequence. The IDLE state is the last state of this sequence. The Precharge sequence consists of States 31-28.

Refresh (RFx)

The Refresh sequence (RFx) implements a CAS before RAS refresh. The Clock Logic PAL, GPCLK, notifies the DRAM state machine when a refresh is required (REFREQ). DSTAT transitions to the Refresh sequence to perform the refresh and sends back a clear refresh signal to GPCLK (CLRFRQ). The Refresh sequence is always entered via a Precharge sequence and exits to a Precharge sequence to ensure proper RAS and CAS levels. The Refresh sequence consists of States 23-20.

Write (Wx)

The Write sequence (Wx) performs initial DRAM writes. For initial write operations the state machine leaves the IDLE state and sequences through the Write sequence. Subsequent far writes cycle back through the Precharge and Write sequences. Subsequent near writes cycle back to a state within the Write sequence itself (W3). The Write sequence consists of States 7-4.

Write sequences are three cycle bus transactions requiring one wait state. The wait state is implemented by disabling state transitions in States 7, 5, 3, 1. The state machine waits in these states for one additional cycle until the wait disable signal (WAITDIS) from DCTLC allows the state transitions to proceed. Wait states are required in order to eliminate the worst case switching overlap possibility of DRAM control signals.

Read Pipe (RPx)

DRAM reads are implemented with three Read sequences. The first read sequence is the Read Pipe sequence (RPx) and is executed only after a Precharge cycle. This sequence sets up the read pipe, returns NA to the i860 and performs the initial read. Subsequent near reads result in the transition from this sequence to the Read Return sequence. All other operations cause the transition to the Read End sequence. The Read Pipe sequence consists of States 27-24.

Read Return (RRx-1)

The Read Return sequence (RRx-1) returns read data once the read pipe is set up. Once the Read Return sequence is entered, subsequent near reads are performed by repeatedly cycling on the Read Return sequence. Subsequent far reads cause the transition to the Read End sequence. The Read Return sequence consists of States 9-8.

Read End (REx)

The Read End sequence terminates read operation by returning previously piped read data and not signaling NA back to the i860. This sequence is used to separate subsequent non-near reads. It consists of States 13-12.

Write-Read (WRX-1)

The Write-Read sequence (WRX-1) implements a near read after a write operation. Subsequent near reads transition to the Read Return sequence. All others transition to the Read End sequence. The Write-Read sequence consists of States 19-16.

Read-Write (RWx)

The Read-Write sequence (RWx) implements a near write after a read operation. It is used to control the data bus direction changes of the DRAMs and i860. The Read-Write sequence consists of States 3-0.

6.2.2 DRAM Control A PAL (DCTLA)

This PAL generates the RAS signals to the four, two megabyte banks of DRAM and the parity DRAMs. It uses the state and address bits to control which bank is activated and when. The no cache signal (DRAMNC) is decoded from the Address Decode PAL as an address space which is not cached in the i860. This is used for instruction tracing during debug. When DRAMNC is active, the cache enable signal (KEN) to the i860 is not active. This PAL also drives the HOLD signal to the i860 depending on the hold request (HREQ) input from the expansion connector.

6.2.3 DRAM Control B PAL (DCTLB)

This PAL generates DRAM control signals which are clocked by the delayed clock (DCLK). This results in these signals changing in advance of other signals clocked by the non-delayed system clock. These signals are the CAS signal (CSX), the Write Enable Latch (WEL) which clocks the write enable signals in DCTLD, the Column Address Latch signal (CAL) which latches the DRAM column address lines, the Column Address Enable signal (CAE) which enables the DRAM column address lines, the Clear Cycle signal (CLRCYC) which clears the cycle pending bit, the Read Data Latch (RDL) which latches the read data from the DRAMs, and the Write Clock Enable signal (WCE) which enables the clocking of the DRAM write data transceiver registers. The Column Address Latch signal can be advanced from the system clock because of the small address delay from the i860. Advancing the address provides earlier DRAM static column read access. Unfortunately,

the DRAMs are not quite quick enough to latch the Read data from the same delayed clock. Therefore the Read Data Latch signal must be delayed an additional 5ns from the Column Address Latch. This provides enough time to get the Read data from the DRAMs and still allows an early latching of data to the i860. Write Clock Enable (WCE) must be delayed to adequately frame the system clock to the DRAM data transceivers. It could be argued that this signal could come from DCTLC, but expansion board control of data clocking is implemented in this PAL, not DCTLC.

6.2.4 DRAM Control C PAL (DCTLC)

This PAL generates DRAM control signals clocked from the system clock. These signals are the Read and Write Data Output Enables (RDDOE, WRDOE) for the DRAM data transceivers, the Row Address Enable (RAE), the Output Enable for the DRAMs, the Read Parity Latch (RPL) which clocks the byte parity error signals, and the Clear Refresh signal (CLRFRQ) which acknowledges a refresh cycle execution. This PAL also generates the wait disable signal (WAITDIS). The state machine PAL is programmed to wait in States 1, 3, 5, and 7 until the wait disable signal is deasserted.

6.2.5 DRAM Control D PAL (DCTLD)

This PAL generates the DRAM write enable signals for individual bytes of data. It is clocked by the Write Enable Latch signal (WEL) from the DRAM Control C PAL. The byte enable signals (BEx) from the i860 specify which byte is written.

6.3 FIFO Control

Access to the FIFOs that interface the i860 with the routing module is provided by the i860 Access PALs. These PALs are:

PAL Name	Part Number	Description
i860AC1	314238	FIFO state machine
i860AC2	314239	Control signal decoder

The first PAL (i860AC1) implements a four bit state machine which is used by the second PAL (i860AC2) to generate the FIFO read and write strobes (RFRD and XFWR). The bidirectional transceivers between the FIFOs and the i860 are also controlled by a clock enable and data enable signals from this PAL. The state diagram for i860AC1 is shown in Figure 6.

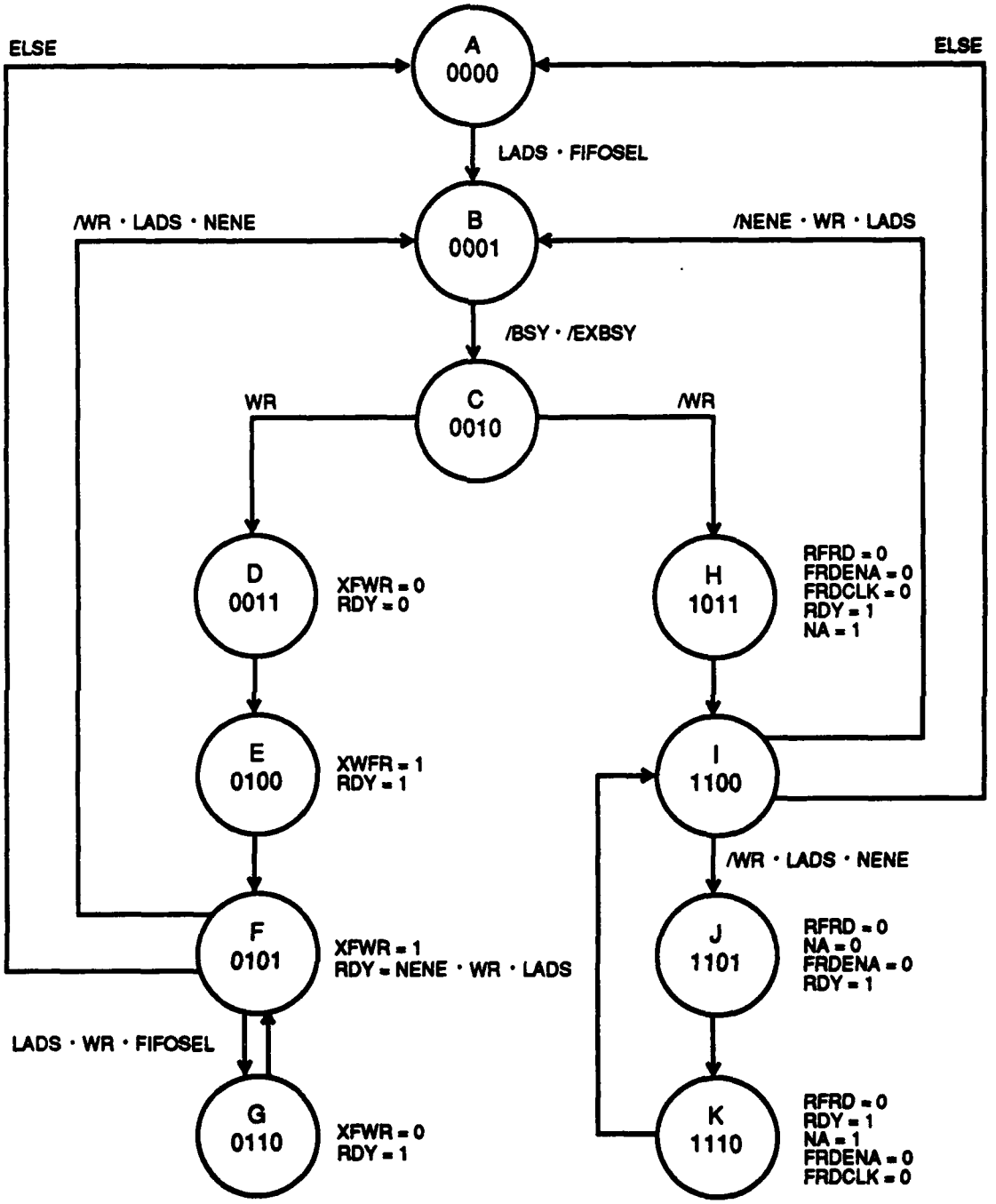


Figure 6
In FIFO Access State Diagram

State A is the IDLE state and is exited upon receiving a select and LADS. The state machine stays in State B until all other piped state machines are not busy. FIFO writes sequence through State SC to SF to complete the first write. In State SF subsequent writes signified by NENE cycle between SF and SG. FIFO reads detected in State SF return to State SB. The absence of FIFOSEL and LADS in State SF indicating no subsequent FIFO accesses return the state machine to IDLE. FIFO reads sequence from SB through SC, SH and SI returning NA to the i860 to initiate other accesses while completing the first read. For subsequent reads the state machine cycles from SI to SJ and SK and back to SI returning NA for another access and providing the read data. Writes are detected in State SI whereupon the state machine branches to State SB and again sequences through SI or SF. An absence of FIFOSEL and LADS in State SI returns the state machine to idle.

6.4 Local Control

The PROM and Control Ports A and B are controlled by the Local Access (LCLACC) PAL, part number 314232. Local access is divided into 'slow' and 'fast' accesses decoded from the i860 address and specified by LOCSEL1. The 'slow' accesses are to the Serial hardware (82510) and the PROM. 'Fast' access is to the registered Port A and B. The encoding is shown in the following table:

LOCSEL	LOCSEL1	LOCSEL0	Read	Write
0	0	0	Serial	Serial
0	0	1	PROM	Counter
0	1	0	Port A (Ctr)	Port A (Perf)
0	1	1	Port B (Stat)	Port B (Ctl)

This PAL implements the state diagram shown in Figure 7.

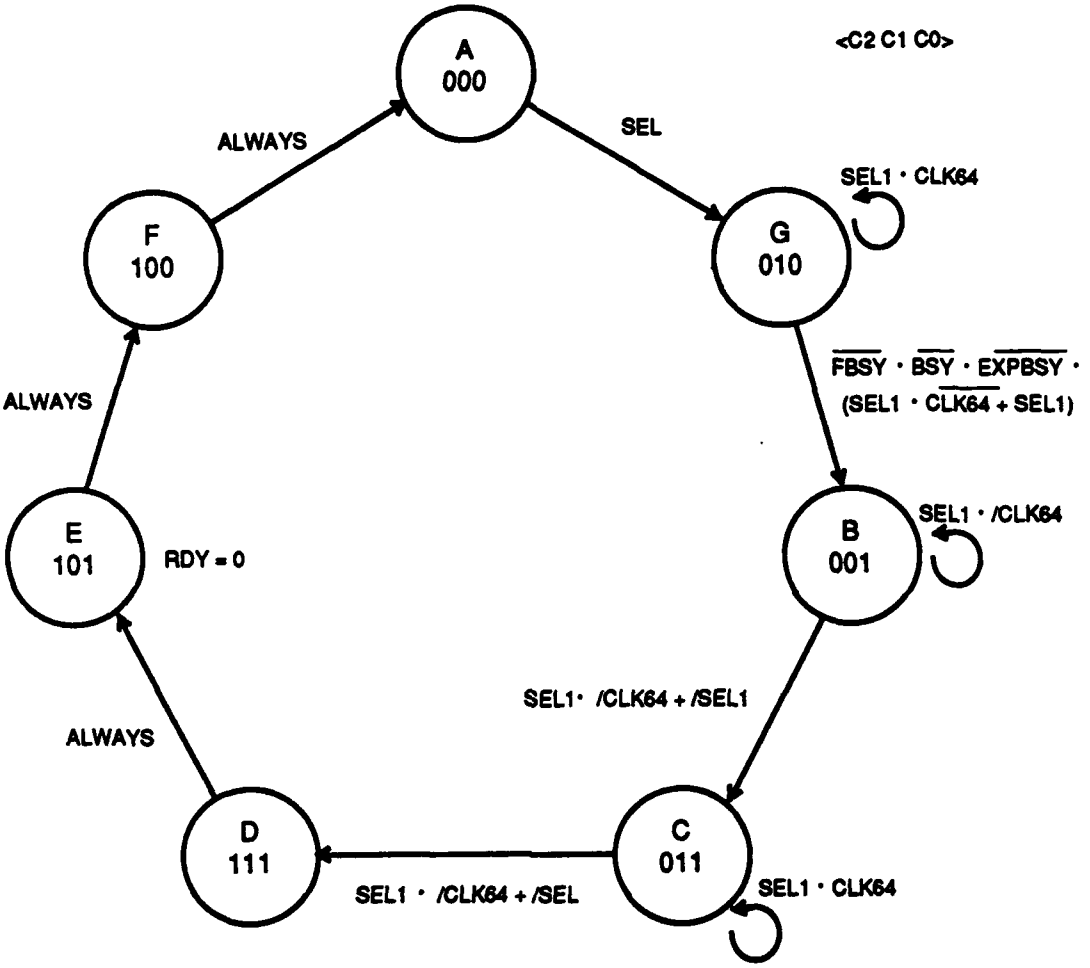


Figure 7
Local Control State Diagram

State A is the IDLE state and is entered after reset and when the state machine is not busy. Upon receiving LOCSEL and a LADS signal, the state machine enters State G. This is a wait state. Here the state machine waits until all other piped state machines are not busy, or if the access is 'slow', until CLK64 is low. CLK64 is the system clock divided by 64. For 'slow' accesses the state machine must wait until CLK64 has been low and has gone high to ensure the actual access has an entire phase of CLK64 to complete. The access is performed by insuring CLK64 is low in State B and generating a strobe signal starting in State C and waiting in State C until CLK64 goes low again. The strobe (PRTSTRB) generated by this PAL is decoded to the appropriate hardware by a 74F138. After State C, the state machine cycles through States D, E, F setting the appropriate data buffer direction with i860WT and returning RDY back to the i860. Note that writing to the counter is implemented as a 'slow' write access to PROM space. Note also that the Port A Performance data bus and the lower byte of the Port B status bus are de-coupled from the 860 data bus by a bidirectional buffer.

6.5 Other PALs

6.5.1 DCM Access PAL (DCMACC)

The DCM access pal (DCMACC), part number 314240, implements a state machine that controls the interface between the i860 FIFOs and the DCM. The state diagram for this PAL is shown in Figure 8. This PAL will be modified for use with the MRI to increase the performance of this interface and make use of the higher bandwidth of the MRI.

(DCMACC1)

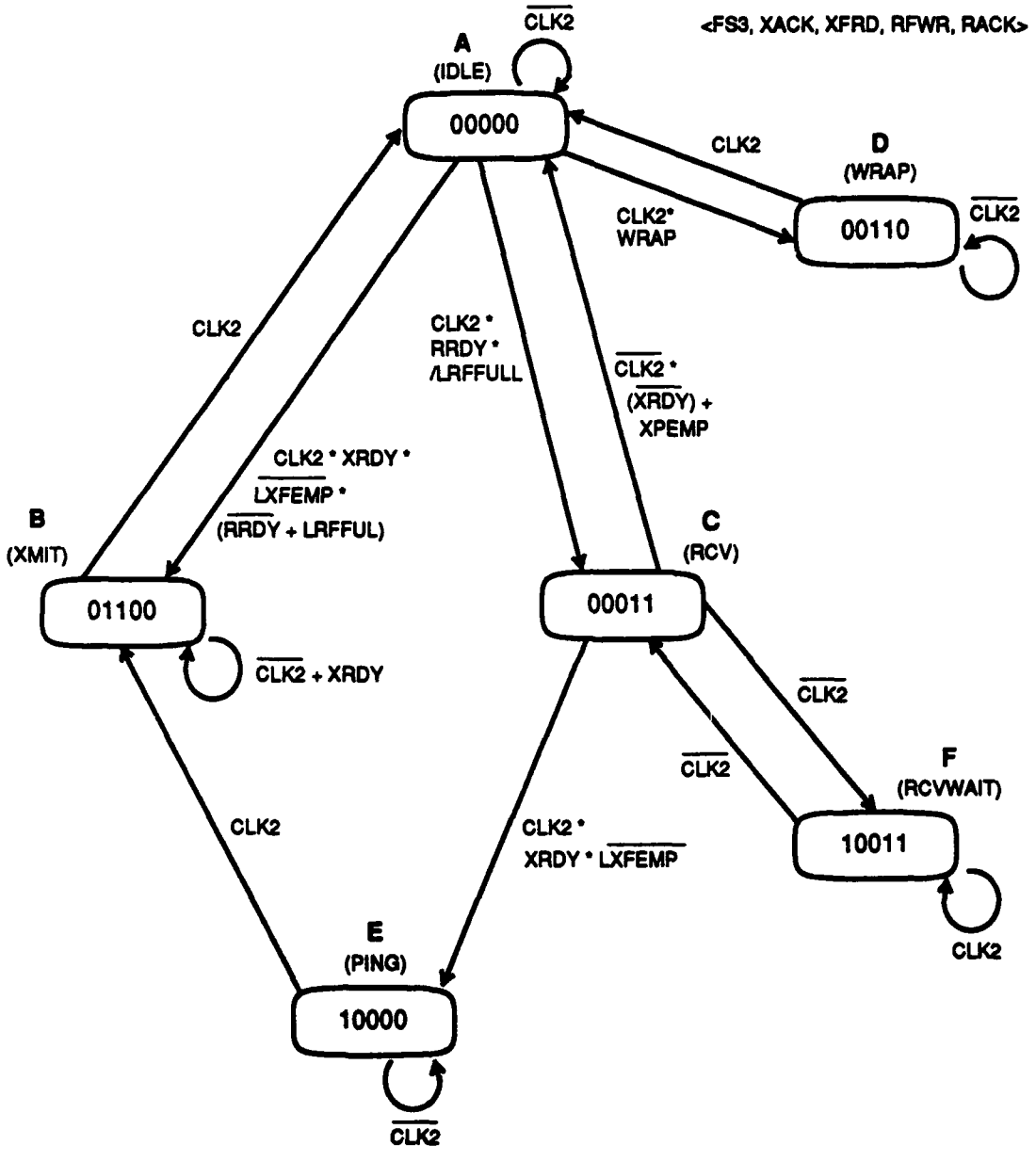


Figure 8
DCM Access State Diagram

Note that in this PAL each state bit is a control signal. State A is the IDLE state where all control signals are deasserted. The DCM asserts Receive Ready (RRDY) or Transmit Ready (XRDY) when it wants to send or receive data to/from the RX-1 node's FIFOs. The state machine waits until CLK2 is high and transitions to the Receive State C if RRDY is active and the Receive FIFO is not full. If XRDY is active and the transmit FIFO is not empty the state machine transitions from the IDLE state to the Transmit State B. The read and write signals to the FIFOs are enabled in the Receive and Transmit states along with the Receive and Transmit Acknowledge signals to the DCM (RACK, XACK). The state machine allows three system clock cycles to perform the receive and two clock cycles to perform the write. The additional receive cycle is achieved by entering the Receive Wait state. If a Receive and Transmit Ready occur simultaneously the Receive is performed first then instead of the state machine returning to IDLE it transitions to a intermediate Ping State E before going to the Transmit State. In the Ping State neither FIFO read or write line is active. The Transmit state always exits to IDLE. Setting the WRAP bit cycles the state machine between IDLE and Wrap State D where both Receive FIFO Read and Transmit FIFO Write signals are enabled without regard to the DCM request signals.

6.5.2 General Purpose Clock PAL (GPCLK)

The General Purpose Clock (GPCLK) PAL, part number 314241, contains two state machines to control the Bus Timeout Interrupt (BUSTOUT) and the Refresh Request (REFREQ). The state diagrams are shown in Figures 9 and 10.

@40 MHz, refresh rate = 12.8 µs

40 MHz uses Ftime7

33 MHz uses Ftime6

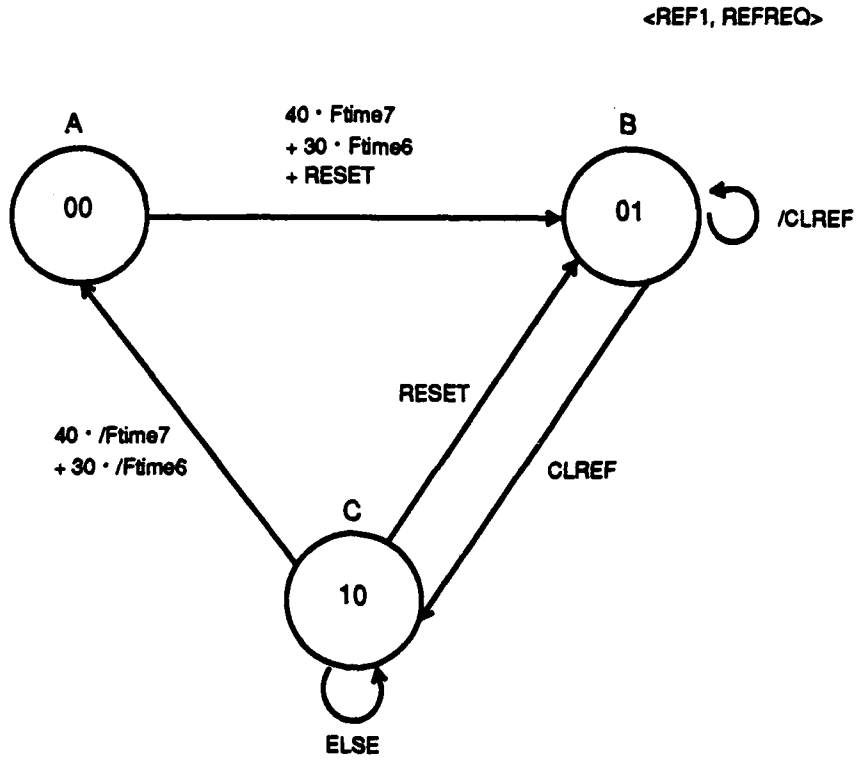


Figure 9
Bus Timeout Interrupt State Diagram

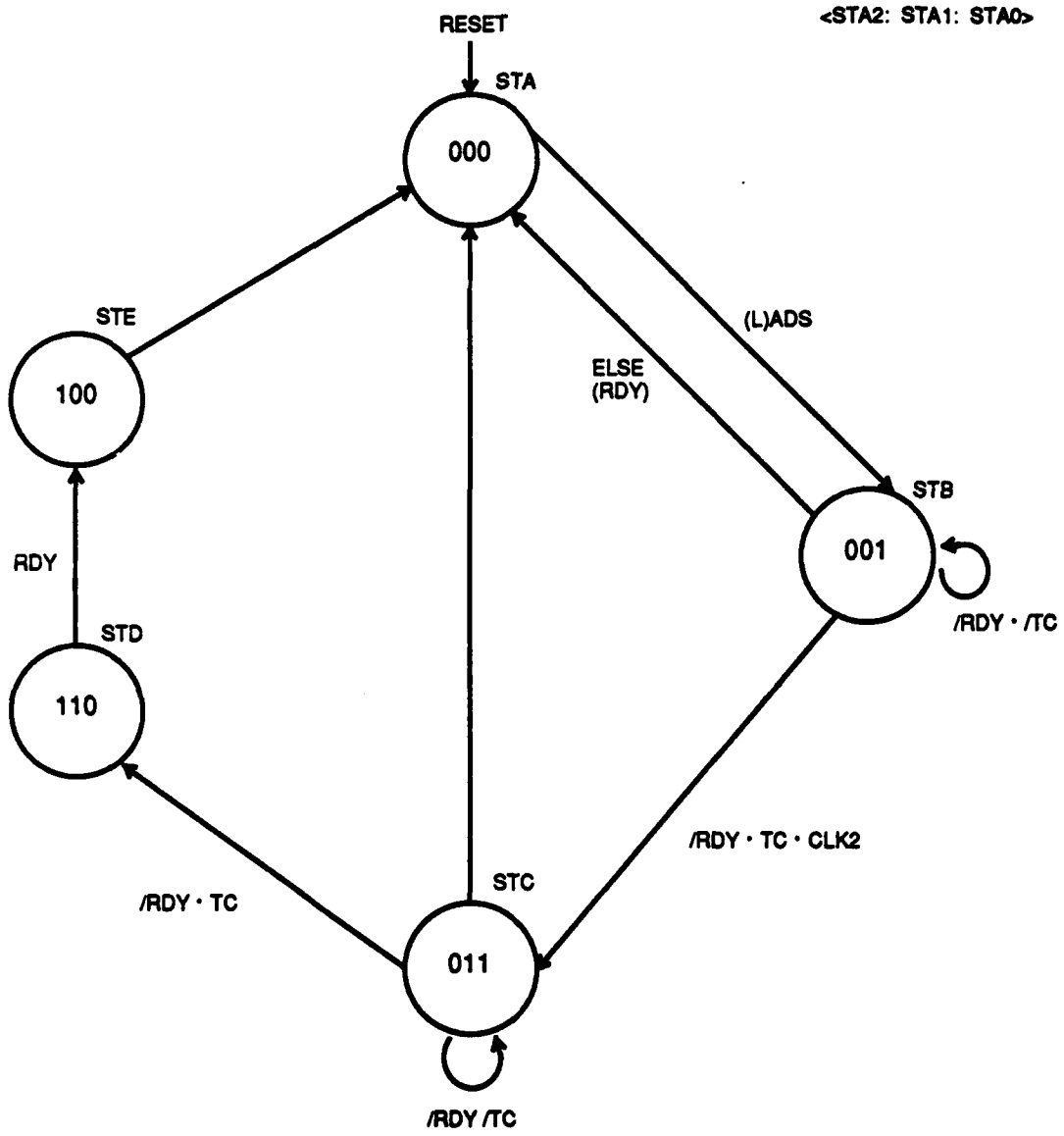


Figure 10
Refresh Request State Diagram

After receiving an ADS from the i860, the Bus Timeout state machine transitions from State STA to STB where it keeps the Bus Timeout Interrupt disabled waiting for a Ready (RDY), which causes a transition back to the initial state. If RDY is not received before the Terminal Count bit (FTIMETC) is active the state machine transitions to State STC where it again waits for RDY or the FTIMETC signal. This ensures that at least one Terminal Count and possibly two will occur before a bus timeout error is signaled. If a RDY is not detected the state machine transitions to State STD where the Bus Timeout bit (BUSTOUT) is set until a Bus Timeout Clear signal (CLRBUS) is received, sending the state machine back to the initial state. RDY is generated for one cycle after transitioning from STC when the timeout interval expires.

The Refresh state machine initiates a Refresh Request to the DRAMs after waiting an appropriate refresh time period or during a reset. During reset the state machine cycles between the Refresh Request and acknowledge states.

6.5.3 Interrupt Control PAL (INTCNT)

The Interrupt Control (INTCNT) PAL, part number 314242, controls the RX-1 node interrupts. The Receive Interrupt (RINT) consists of the Receive FIFO Half Full (LRHF) and existing EOD flag (EODINF). These are masked by the Receive Mask (RMSK) and EOD Mask (EODMSK) bits of the Control Port B. The RINT signal is read as a Port B status bit. The Transmit Interrupt (XINT) consists of the synchronized Transmit Half Full signal (LXHF) which is masked by the Port B Transmit Mask bit (XMSK). The XINT signal is also read as a Port B status bit. The i860 Interrupt (i860INT) is the combination of the Expansion Interrupt (LEXPINT), Parity Interrupt (PARINT), Bus Timeout Interrupt (BUSTOUT), Serial Interrupt (LSERINT), each of which has a separate mask bit from the Control Port B, and the Receive and Transmit Interrupts (RINT, XINT).

7. Expansion Port

The RX-1 node design consists of an i860 processor acting as a master, initiating operations of several support functional blocks which are considered slaves. These slaves respond to read and write bus transactions on the i860 interface bus. The RX-1 node expansion port allows adding functional blocks to the node by attaching a daughter card to the expansion connectors, which connects the CPU address bus, DRAM data bus and bus control signals. Note that the data connection is not directly to the i860 data bus, but on the bidirectional data bus between the i860 data transceivers and the DRAM. The expansion port can be an additional slave, responding only to read and write operations of the processor, or it can be a master, placing the i860 in hold and initiating bus transactions to other function blocks on the node.

7.1 Expansion Port Theory of Operation

The RX-1 node board design does not support any particular functional capability of the expansion port. Any use of the expansion port requires reprogramming of node control PALs. The following is a high level description of the possible functions of the expansion port and specification of which node control PALs must be reprogrammed. Detailed design considerations of the impact on PAL operation and required reprogramming can be determined by understanding the RX-1 node theory of operations chapter.

7.1.1 Expansion Port Slave Operation

All slave functional blocks are specified by memory mapping the i860 address space. The expansion port responds to i860 bus transactions by mapping unused memory space of the node design. i860 data is accessed on the DRAM data bus by reprogramming the DRAM control PALs to send or receive data from the i860 through the DRAM data transceivers under control of the expansion port.

Specific DRAM control signals are brought out to the expansion port to facilitate increases in memory size. The control signals are used to reproduce the node memory design. The memory increase need not conform to any particular size, but requires use of 70ns Trac and 20ns Tcac fast page DRAMs. A configuration of four banks of two megabytes using one megabit DRAMs or four banks of eight megabytes using four megabit DRAMs is recommended. This is a replication of memory available on the node. Node memory wraps after the node memory size is exceeded. Addition of expansion memory requires the disabling of node memory wrapping while expansion memory is accessed.

7.1.2 Expansion Port Master Operation

In order to access other functional blocks on the node, the expansion port acts as a node master by placing the i860 in hold and initiating bus transactions in its place. The DRAM control PALs must be reprogrammed to control the i860/DRAM data bus transceivers to provide i860 data bus access from the DRAM data bus available to the expansion port.

7.1.3 Expansion Port DMA Operation

The expansion port can initiate direct data transfer between itself and the node DRAM. In this case, the expansion port places the i860 in hold and initiates a bus transaction to the node memory. The i860 data bus transceivers are disabled allowing write data to be driven by the expansion port instead of the transceivers. Read data is latched by both the transceivers and the expansion port. The DMA bandwidth and latency is essentially the same as the node memory. This data can be subsequently transferred to other node functional blocks such as the FIFO.

DMA between DRAM and other functional blocks without going through the expansion port is possible but probably not practical. The expansion port would have to alternate between DRAM bus cycles and other functional block bus cycles. Each cycle would have a significant startup latency severely impacting bandwidth. Using the expansion port to store and forward data permits pipelining of bus transactions.

Of course, a special memory space could be programmed to allow concurrent operation of the DRAM and other functional blocks. This would require entire reprogramming of the state machines that control bus interfacing in order for two functional block operations to dovetail. This is a difficult, if not challenging exercise left to the reader.

7.1.4 Expansion Port Node Impact

The following table specifies the RX-1 node PALs that must be reprogrammed for the specific operation of the expansion port.

Operation	Control PAL	Comments
General Slave	ADDEC1	Provide memory map decoding to select expansion port. (See EXPSEL# signal description.)
	DCTLB/DCTLC	Permit control of DRAM data transceivers by ERDL#, EWDL#, ERDE#, and EWDE# in order for expansion port to access i860 data bus when DRAM not active.
Memory Expansion	ADDEC1	Provide memory map decoding to select expansion port.

	DCTLA	Disable node memory wrapping.
Bus Master	DCTLB/DCTLC	Permit control of DRAM data transceivers by ERDL#, EWDL#, ERDE#, and EWDE# in order for expansion port to access i860 data bus when DRAM not active.
DMA	DCTLB /DCTLC	Disable DRAM data transceiver writes in order for expansion port to drive node DRAM.

7.2 Expansion Port Signal Descriptions

7.2.1 I860 Bus Signals

The following signals directly access the i860 processor. Additional descriptions of these signals can be found in the i860 specifications.

A<31:03>	i860 address bus. Expansion port input/output.
BE<7:0>#	i860 byte write enables. Expansion port input/output.
ADS#	i860 Address Strobe. Initiates bus cycle from i860. Expansion port input/output.
W/R#	Write/Read#. Specifies bus cycle type as write or read from i860. Expansion port input/output.
NENE#	Next Near. Specifies that bus cycle is within a DRAM page from the previous cycle. Expansion port input/output.
i860RDY#	i860 Ready. Completes bus cycle to i860. Expansion port input/output.
NA#	Next Address. Requests new bus cycle from i860 before completing previous outstanding cycle. Initiates bus cycle pipelining. Expansion port input/output.
BREQ	Bus request. Indication of pending bus cycle from i860. Expansion port input.
HLDA	Bus hold acknowledge from i860. Expansion port input.
LOCK#	Bus cycle lock signal from i860. Expansion port input/output.

7.2.2 Data Bus Signals

The following signals describe the expansion port data bus. This data bus is an extension of the bus between the DRAMs and i860 data transceivers and will be active during node memory accesses. Control signals to the DRAM control PALs can be used to enable the data transceivers to send or receive data from the expansion port when node memory is not being accessed.

D<63:00> Bidirectional DRAM Data Bus between DRAM banks and bidirectional data transceivers. Expansion port input/output.

WPD<7:0> Write Parity Data from parity latch to parity DRAMs. Even parity is generated for each byte of data. Expansion port output.

RPD<7:0> Read Parity Data from parity DRAMs to parity latch. Expansion port input.

7.2.3 Expansion Port Control Signals

The following signals are used by the expansion port to integrate with RX node operation.

HREQ# Bus hold request, inverted and sent to i860. Expansion port output.

LADS3# Latched Address Strobe 3. Expansion port latched address strobe indicating initiation of a bus cycle. Expansion port input.

EXPSEL# Expansion Select. Decoded address bits specifying memory map location of expansion port. Expansion port input.

EXPBSY# Expansion Busy. This signal is driven to the PALs that control i860 bus access on the node in order to prevent these accesses from starting before the completion of the access by the expansion port. Expansion port output.

S<4:0> DRAM control state machine bits. Used to specify RAS to DRAM banks. Expansion port input.

BSY# (DRAM) Busy. Signal driven to the expansion port indicating that the DRAM control state machine is active. This indicates that the DRAM data bus may have active DRAM data driven on it. Expansion port input.

EWDL# Expansion Write Data Latch. This signal goes to the DRAM control PAL which drives the DRAM data transceiver's write clock enable. Expansion port output.

ERDL# Expansion Read Data Latch. This signal goes to the DRAM control PAL which drives the clock to the DRAM data transceiver's read registers. Expansion port output.

EWDE# Expansion Write Data Enable. This signal goes to the DRAM control PAL which drives the data transceiver's write data enable. Expansion port output.

ERDE# Expansion Read Data Enable. This signal goes to the DRAM control PAL which drives the data transceiver's read data enable. Expansion port output.

EXPINT# Expansion Interrupt. This signal is combined with all other interrupt sources to interrupt the i860. It can be disabled with a separate mask bit. Expansion port output.

7.2.4 Memory Expansion Signals

The following signals are driven by the node DRAM control PALs in order to use the expansion port as additional memory.

WE<7:0># Write Enable. DRAM byte write enables. Expansion port input.

RAE# Row Address Enable. Enables expansion address buffers to drive row address to DRAMs. Expansion port input.

CAL# Column Address Latch. Clocks column address into column address registers. Expansion port input.

CAE# Column Address Enable. Enables expansion address registers to drive columns address to DRAMs. Expansion port input.

CSX# CAS. CAS signal driven from a node DRAM control pal. This signal should be buffered and driven to all expansion DRAM banks. Expansion port input.

OEX# (DRAM) Output Enable. This signal is driven from a node DRAM control PAL. It should be buffered and driven to all expansion DRAM banks. Expansion port input.

7.2.5 RX-1 Node System Signals

CLKC 40MHz System Clock. Expansion port input.

i860RESET System Reset. Active for 1ms after stabilization of Vcc or termination of reset signal from Direct Connect Module. Expansion port input.

7.3 Signal Loading

7.3.1 DC loading

The drive capability of all sources on the node board far exceed the node's DC loading. The source current available to the expansion port is specified without consideration of the node's DC loading. This is because AC loading limits expansion port loading much sooner than DC loading. Unless the expansion port loads require an excessive drive capability, the AC impact of the expansion port should overshadow any DC loading effect. As a rule of thumb, the expansion port can utilize up to 75 percent of the DC drive capability of the signals provided.

7.3.2 AC loading

All AC timing parameters are specified with a test circuit of 50pf. If the cumulative capacitive loading of the expansion port and the node board exceed 50pf, appropriate derating must be applied. The following table is an estimate of the maximum capacitive load available for the expansion port devices. The table uses a rule of thumb of 10pf for each load including routing. The worst case timing of node board is very tight. Exceeding the capacitive loading recommended here may impact node board operation. AC loading is the load limiting characteristic of the expansion port.

Signal	Load
A<31:03>	10pf
BE<7:0>#	40pf
ADS#	10pf
W/R#	10pf
NENE#	10pf
i860RDY#	10pf
NA#	10pf
BREQ	50pf
HLDA	50pf
LOCK#	50pf
D<63:00>	40pf
WPD<7:0>	80pf
RPD<7:0>	80pf
LADS3#	40pf
EXPSEL#	40pf
S<4:0>	10pf
BSY#	40pf
WE<7:0>	40pf
RAE#	20pf
CAL#	20pf
CAE#	20pf
CSX#	10pf
OEX#	10pf

7.4 Expansion Port Signals

The electrical characteristics of the expansion module signals are determined by the source/ termination and loading of the node board.

7.4.1 i860 Signals

A<31:3>, BE<7:0>#, ADS#, W/R#, NENE#, i860RDY#, NA#, BREQ, HLDA, LOCK#

DC Characteristics

Symbol	Parameter/Signal	Min	Max	Units
Vil	Input Low Voltage	-0.3	+0.8	Volts
Vih	Input High Voltage	2.0	Vcc+0.3	Volts
Vol	Output Low Voltage		0.45	Volts[1]
Voh	Output High Voltage	2.4		Volts[2]
Iil	Input Leakage Current			
	A<11:03>		60	µA
			-1.5	mA
	A<12>		120	µA
			-2.1	mA
	A<20:13>		60	µA
			-0.6	mA
	A<21>		85	µA
			-0.6	mA
	A<24:23>		50	µA
			-500	µA
	A<27>,BE<7:3>#,ADS#		25	µA
			-250	µA
	A<31:28>		25	µA
			-250	µA
	BE<2:0>#		35	µA
			-240	µA
	W/R#		175	µA
			1.75	mA
	NENE#		75	µA
			750	µA
	All others		±15	µA

[1] Measured at 4.0mA for address and byte enables, 5.0mA for definition and control.

[2] Measured at 1.0mA for address and byte enables, 0.9mA for definition and control.

AC Characteristics

Symbol	Parameter/Signal	Min	Max	Units
Tco	Clock to output valid			
	A<31:3>,W/R#,NENE#	3.5	19.0	ns
	BE<7:0>#	3.5	21.0	ns
	ADS#,BREQ,LOCK#,HLDA	3.5	15.0	ns
Tsu	Set Up Time			
	A<31:28>	17.0		ns
	All Others	8.0		ns
Thld	Hold Time			
	BE<7:0>#	6.5		ns
	All Others	3.0		ns

7.4.2 PAL Signals

All the following signals are from or to control PALs on the node board.

HREQ#, LADS3#, EXPSEL#, EXPBYS#, S<4:0>, BSY#, EWDL#, ERDL#, EWDE#, ERDE#, EXPINT, WE<7:0>, RAE#, CAL#, CAE#, CSX#, OEX#

DC Characteristics

Symbol	Parameter/Signal	Min	Max	Units
Vil	Input Low Voltage		+0.8	Volts
Vih	Input High Voltage	2.0		Volts
Vol	Output Low Voltage		0.5	Volts[1]
Voh	Output High Voltage	2.4		Volts[2]
Iih	Input High Leakage Current		25	µA
Iil	Input Low Leakage Current		-250	µA

[1]Iol = 24.0 mA.

[2]Ioh = - 3.2 mA.

AC Characteristics

Symbol	Parameter/Signal	Min	Max	Units
Tco	Clock to output valid			
	BSY#, LADS3#, S<4:0>, RAE#, OEX#	2.0	6.5	ns
	CAL#, CAE#, CSX#	16.0	22.5	ns[1]
	EXPSEL#	5.5	29.0	ns
	WE<7:0>#	4.0	13.0	ns
Tsu	Set Up Time			
	HREQ#	10.5		ns
	EXPBSY#, EWDE#, ERDE#	7.0		ns
	EWDL#, ERDL#	18.0		ns[1]
	EXPINT#			Asynchronous[2]
Thld	Hold Time			
	HREQ#	0.0		ns
	EXPBSY#, EWDE#, ERDE#	0.0		ns
	EWDL#, ERDL#	-11.0		ns[1]
	EXPINT#			Asynchronous[2]

[1]These signals are clocked by CLKA which is advanced 10 ±1 ns from the system clock.

[2]Double synched into RX-1 interrupt logic.

7.4.3 Data Signals

D<63:00>, WPD<7:0>, RPD<7:0>.

Bidirectional buffers (IDT29FCT52B)

DC Characteristics

Symbol	Parameter/Signal	Min	Max	Units
Vil	Input Low Voltage		+0.8	Volts
Vih	Input High Voltage	2.0		Volts
Vol	Output Low Voltage		0.55	Volts[1]
Voh	Output High Voltage	2.4		Volts[2]
Iih	Input High Leakage Current		15	µA
Iil	Input Low Leakage Current		-15	µA

[1]Iol = 64.0 mA.

[2]Ioh = -24.0 mA.

AC Characteristics

Symbol	Parameter/Signal	Min	Max	Units
Tco	Clock to output valid	2.0	6.5	ns
Tsu	Set Up Time	2.0		ns
Thld	Hold Time	2.0		ns

DRAM

DC Characteristics

Symbol	Parameter/Signal	Min	Max	Units
Vil	Input Low Voltage	-1.0	+0.8	Volts
Vih	Input High Voltage	2.4	6.5	Volts
Vol	Output Low Voltage		0.4	Volts[1]
Voh	Output High Voltage	2.4		Volts[2]
Iih	Input High Leakage Current		40	µA
Iil	Input Low Leakage Current		-40	µA

[1]Iol = 4.2 mA.

[2]Ioh = - 5.0 mA.

AC Characteristics

Symbol	Parameter/Signal	Min	Max	Units
Tco	Clock to output valid	17.0	23.7	ns
Tsu	Set Up Time	7.0		ns
Thld	Hold Time	18.7		ns

Note: These parameters are normally referenced to the DRAM CAS signal. Due to the complexity of the DRAM control logic, the timing windows have been referenced to the appropriate system clock edge.

7.4.4 System Signals

CLKC, i860RESET

DC Characteristics

Symbol	Parameter/Signal	Min	Max	Units
Vol	Output Low Voltage			
	i860RESET		0.5	Volts[1]
	CLKC		1.65	Volts[2]
Voh	Output High Voltage			
	i860RESET	2.7		Volts[3]
	CLKC	3.85		Volts[4]

[1]Iol = 20.0 mA.

[2]Iol = 75.0 mA.

[3]Ioh = -1.0 mA.

[4]Ioh = -75.0 mA.

AC Characteristics

i860RESET		Asynchronous
CLKC	Frequency Tolerance	± .01%
	Jitter	Std Dev < 100 ps
	Symmetry	65/35
	Skew (to other node clocks)	±1 ns

7.5 Expansion Port Power

The expansion port power is provided in two ways: Pins on the connectors for +5V, +12V, -12V, and ground; Two mounting studs which provide +5V and ground. The amount of +5V power is limited by the +5V and ground pins between the RX-1 node and the backplane This limit depends how much power the communication module is using. Ten +5V pins connect the node to the backplane and at 1 amp per pin 10 total amps are available. The RX-1 and DCM pair is rated at 7.2 amps, which leave 2.8 amps for the expansion port. The MRI will have similar power requirements as the DCM. +12V and -12V currents are limited to 1 amp each by the one pin connecting the node to the backplane.

8. Built in RX-1 Hardware Test Programs

Along with low level 'go-no go' diagnostic routines, some H/W debug routines are also resident in the boot PROM on board. The PROM has Intel part number 314247. The PROM boots the node or runs debugging code depending on the state of the TESTFIXT# signal on pin P1-C19. A flow diagram of the PROM code is given in Figure 11.

The RX-1 node PROM code is further described in the Diagnostics Reference Manual, 314305.

8.1 Boot Code

In the system environment, TESTFIXT# is left floating and is pulled up by the node. The system PROM sees an inactive state of TESTFIXT# and runs the boot code. The boot code runs several node confidence tests (NCTs) before booting the node. The NCTs are a subset of the debugging test suite discussed later. Any NCT failure results in flashing the red LED with a failure message and any possible failure characteristics. This message can actually be read by a light detector attached to a CRT terminals serial input port. The node will not boot if the RAM or UART NCTs fail. The node will boot even if the FIFO, counter, and parity NCTs fail. For these failures, the failure message is repeated until a character is received from the serial port.

8.2 Debugging Code

An active TESTFIXT# signal specifies running the debugging code instead of the boot code. The debugging code is further differentiated between a standard suite of tests and individual tests. The tests are specified by the XPR<3:0> signals. The XPR signals are pulled high on the node. If none of them are pulled low, the standard debug test suite is continually run. The individual debug tests are specified by the XPR bit patterns shown below:

XPR<3:0>	Test
F	Standard Test Suite
B	UART Loopback Test
A	Parity Test
9	Counter Test
8	FIFO Test
7	Byte Wide RAM Test
6	32 Bit Wide RAM Test
5	Write/Read Counter Test
4	Write/Read RAM
3	Write/Read FIFO
2	Write/Read UART
1	Write/Read PA Port/Counter
0	Write/Read Control/Status Ports]

8.2.1 Standard Suite of Debugging Tests

The standard debugging test suite continually runs the 32-bit RAM test, byte wide RAM test, FIFO test, counter test, parity test and UART test in the order given. If an error is discovered, a failure message and any possible failure characteristics are flashed on the red LED and the test number flashed on the yellow LED. The tests stop on a failure while the message continues to flash. This test suite consists of the most thorough hardware tests available and passage of these tests is required for node functional acceptance. These tests are somewhat complex. To use these tests for debugging requires intimate familiarity with the structure of the test and extensive use of the diagnostic capabilities of the test itself.

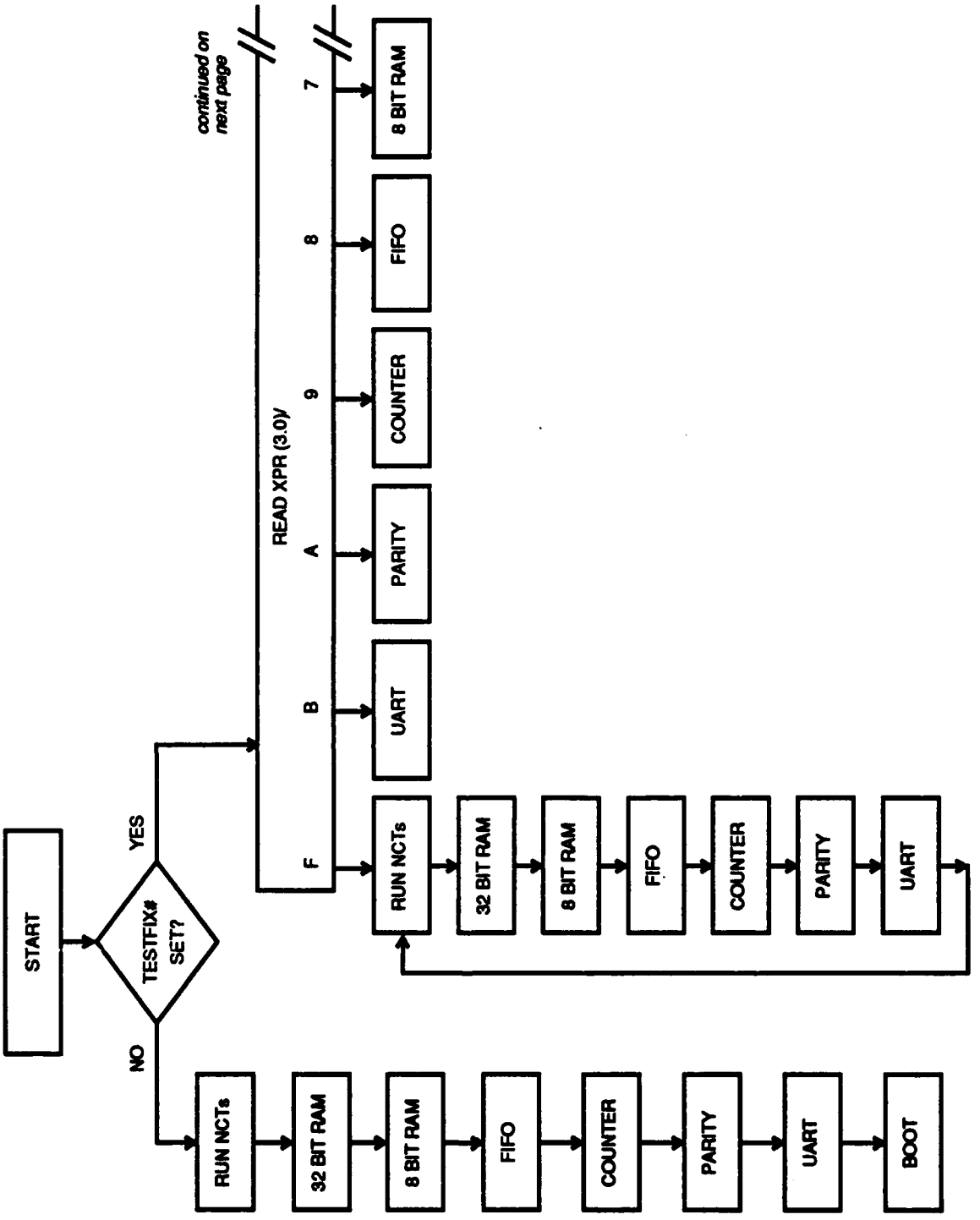


Figure 11 (1 of 2)
i860 Node Firmware Flowchart

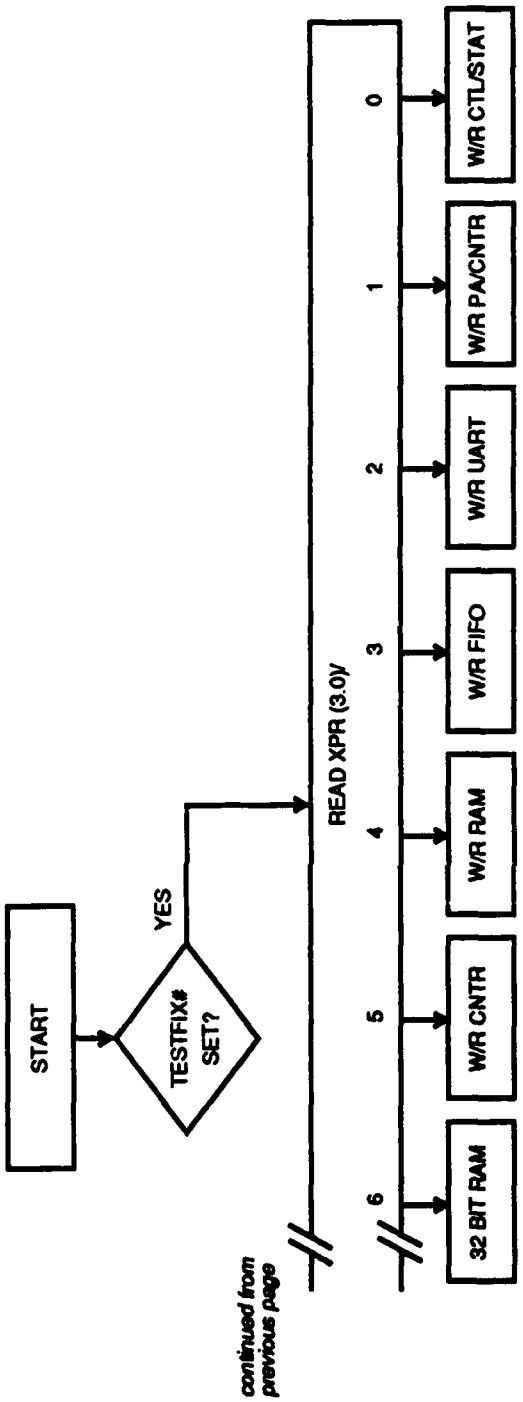


Figure 11 (2 of 2)
i860 Node Firmware Flowchart

8.2.2 Individual Debugging Tests

The individual debugging tests consist of the tests that make up the standard test suite and simple loops that write and read specific functional blocks of the node. The individual tests loop continually with the test suite tests indicating a pass or fail.

32-Bit RAM Test

This test loads all of memory with a looping pattern of three 32-bit words. After the entire memory is written, each 32-bit word, starting at the beginning of memory, is read, verified and written with the next 32-bit word of the pattern. This process is repeated for four passes through the node memory.

Byte-Wide RAM Test

This test is the same as the 32-bit RAM test only on a byte wide basis.

Counter Test

This test preloads the counter with a very large value. The counter is then read to check that it has been preloaded. After waiting a certain amount of time, the counter is read to check if the preloaded value has incremented and the appropriate upper bits have rolled over.

FIFO Test

The FIFO test writes values into the transmit FIFO, including EOD and checks the correct values of the flags. The WRAP function is enabled, wrapping the transmit data into the receive FIFO. The receive FIFO is read, checking the correct operation of the flags, and the data verified. Two functions are checked in this test: the wrapped FIFO data and the FIFO flags. The test differentiates the type of error by the error location.

Parity Test

The parity test loads the entire memory with a recurring byte pattern of different parity. The memory is then read and checked for the absence of a parity error. The byte pattern is rotated by one byte, and the test is repeated twice more. After the absence of a parity error is checked, the parity DRAMs are disabled and byte patterns inconsistent with the parity in memory is loaded into the data DRAMs. The memory is then read and checked that a parity error is present. This is done twice.

UART Test

The UART test sets the UART into a loopback mode and echo checks data written to the UART.

Write/Read DRAM Test

This test writes and reads two individual patterns into a single location of memory. The patterns alternate in byte parity in order to check the parity generate/check circuit.

Write/Read FIFO Test

This test writes and reads two individual patterns into the transmit and receive FIFOs in the wrap mode. One pattern is used as an EOD message terminator to check EOD operation.

Write/Read UART

This test writes and reads the UART without any consideration of what will be read. It simply checks the bus transaction operation to the UART.

Write/Read Counter

This test writes and reads a value to/from the counter. The value read from the counter should be similar to the value written only incremented slightly.

Write/Read PA Port/Counter

Due to the memory mapping of the node, the memory space of the PA port when read, gives counter data. This test writes two values to the PA port and reads the counter.

Write/Read Control/Status

This test repeatedly writes control data and reads status data.

8.3 Debugging Methodology

The debugging approach recommended for the RX-1 node is as follows:

1. Run standard debugging test suite and check for an indication of failing or non-failing tests.
2. If no indication occurs, i.e. the board is dead, rerun the standard debugging test suite and use a lab instrument like a logic analyzer to debug PROM, DRAM, STATUS, CONTROL and/or UART access. PROM, DRAM, STATUS, CONTROL and UART accesses are the minimum requirements for some sort of hardware diagnostic indication of board malfunction by the system PROM. Note that a large part of the board must be functional in order to get some useful information by using the system PROM. Debugging the operation of these minimum requirements using the complex system PROM may not be practical. In this case other PROMs containing the simple loop programs may be required to get these basic functions operational in order to use the system PROM.

3. Once an indication is given as to the malfunctioning part of the node, run the simple codes that exercise the malfunctioning part and debug.
4. Once the simple code operation is verified, select and debug the individual hardware NCT that fails.

The node board has basically three access modes. These are: local, FIFO and DRAM, listed in increasing complexity. Local accesses are relatively simple and include the majority of the minimum requirements for diagnostic self-help of the FIFO and DRAM modes.

8.3.1 Local Access Debugging

Local accesses include PROM, Counter, Status, Control and UART accesses. They are all controlled by the LCLACC PAL, U211. The basic operation of this PAL is to recognize a local access, provide a read or write strobe of appropriate length, enable and specify the direction of the 8-bit data buffer U202, and provide RDY to complete the access. Debugging is accomplished by verifying the address decoding, state machine operation and control line operation to all local functions. A complication can occur if a FIFO or DRAM access conflicts with the local access. This can be checked by removing the FIFO and/or DRAM control PALs and deasserting their BSY signals.

The correct operation of local accesses should be simple to achieve and is essential in using the RX-1 diagnostic capabilities to debug the FIFO and DRAM operation.

8.3.2 FIFO Debugging

FIFO debugging is very complex and requires an intimate knowledge of the FIFO control and operation. Once the local access is operational, the simple FIFO Write/Read program can be run and debugged with a logic analyzer. Failure of the hardware FIFO NCT is very difficult to diagnose. Failure messages and symptoms must be analyzed to determine the cause of any failure. Familiarization of the FIFO hardware NCT is also essential.

8.3.3 DRAM debugging

DRAM debugging is the most complex debugging task for the RX-1 node. Local access operation is essential before DRAM debugging can be attempted. The simple DRAM Write/Read program can be used with a logic analyzer to verify basic DRAM operation. All DRAM control signals should be verified as active at the DRAM chips. The pipelined access of the DRAMs can be very confusing and requires intimate knowledge of the DRAM control state machine operation. Once the simple Write/Read program is

operational, the hardware NCT diagnostic messages should help with NCT debugging. Characteristics such as which bit(s) fail should lead to conclusions of DRAM chip failure or bank control signal failure. The approach here is to use the diagnostic message to focus the debugging effort. The NCT tests are much too complex to try to trace the error from the beginning of the test.

9. Conclusions

The goals of the DELTA numeric node program were certainly met. The node was designed, constructed, and tested over an environmental range of 0 to 70° C, and voltage margined at ±5%. The NX/2 operating system has been ported as well as a good deal of diagnostic code. The node is now ready for applications in the iPSC/2 hypercube, and will soon find it's real home in the DELTA system.

The main worry that the project team had was that a 40MHz design could be done reliably without some exotic technologies. That worry has been answered with a definite yes. This has given us optimism that a higher frequency can be attained given some subtle changes in the implementation, and of course i860s rated at those higher frequencies.

10. Recommendation

This section discusses lessons learned during this effort, and relates those to suggestions for the next numeric node design.

10.1 DATA PATH ASIC

An ASIC, to replace many of the discrete components in the memory data/address path, would be fairly simple in design, but enhance node in several ways. It would be used twice in an RX-2 design. Functions provided are:

1. Data pipeline. This function replaces eight high-speed bi-directional data latches.
2. Error detection and correction. Hand in hand with the data pipe is the generation and checking of an error detection and correction code or syndrome. This mechanism would replace the parity scheme used on the RX-1 which would greatly increase reliability. One proposed method would provide four-bit detect and one-bit detect on each of the 32-bit words making up the 64-bit data word.
3. PMRAM cache comparators. This stands for Page Mode RAM cache. It extends the use of page mode DRAMs from the RX-1 such that up to four banks could have active row address selects (RAS) active. This would reduce the access time penalty when accesses thrash between up to four address ranges. The ASIC would provide the address latching and comparing for this function.
4. Faster performance port. The RX-1 sped up the PA port by automatically generating the strobe. The ASIC would speed it up further by taking a wider word from the node than four bits, probably thirty two bits. It would then strobe these four bits at a time out the PA lines of the node.
5. Integration of miscellaneous features. The status and control ports and the counter would be embedded in the ASIC. This results in a ten to twelve IC reduction.

10.2 Clock Frequency

From our experience with the RX-1, we are optimistic that the clock frequency of the next node can be increased. Naturally this assumes Intel produces faster i860s. Our goal for the next effort would be 50MHz, a 25% increase.

10.3 Network Interface Improvements

Three functional changes have been identified which would speed up message passing performance. Two are specific, and the third is more open ended:

1. Wider FIFO data path. Widening the data path from 32 to 64 bits would increase node to routing logic bandwidth by 25%. This wasn't done on the RX-1 because of the expense in board real estate using FIFO devices available at the time. A new bi-directional FIFO is now available which addresses these concerns.
2. Single cycle FIFO transfers. The RX-1 one requires the i860 to load and store data moving between FIFO and DRAM. This was done to automatically handle the internal cache of the i860. This limits bandwidth to less than half the DRAM bandwidth since a load and store must take place. There is no good way to handle the cache with the current i860. For messages which can be guaranteed not to be affected by cache-like resultant vectors, a single-cycle technique could help. This would have the i860 still move the data, but instead of loads *and* stores it would just do loads *or* stores. The address selected would trigger single cycle accesses between the DRAMs and FIFOs. With this approach the maximum memory bandwidth could be approached.
3. Message start/stop assist. We would like to pursue techniques through which the hardware could unburden the OS of some or all of the message protocol overhead. We are currently reviewing papers and discussing techniques. We believe that a solution to this problem is the key to reducing the latency associated with message passing machines.

10.4 Flash Electrically Erasable EPROM

This enhancement would simply supply the hardware mechanism to use electrically erasable PROMs which would allow the boot PROM to be modified in the field without physically touching the node.

10.5 PMRAM Cache

This was described in the ASIC section. The ASIC would provide the address comparators. Control logic would also be needed to support this function.

10.6 Integration of Control Logic

The RX-1 has several state machines for the numerous memory mapped entities on it's local bus. In a next effort these state machines would be folded into one master state machine. GaS state machine components now exist that could make this possible. The result is higher reliability and a better chance to operate at higher frequencies.

11. References

1. RX-1 Node Schematic Drawings, Intel Document Number: 314284.
2. Diagnostics Reference Manual, Intel Document Number: 314305.
3. *i860 64-bit Microprocessor Programmer's Reference Manual*, Intel Order Number: 240329-002.
4. *Microcommunications Handbook*, Intel Order Number: 231658.

12. Distribution List

Delivery of this report has been made to the following:

DARPA/STC
Attn: Stephen L. Squires
1400 Wilson Boulevard
Arlington, VA 22209-2308
(one copy)

DARPA/RMO/Retrieval Services
1400 Wilson Boulevard
Arlington, VA 22209-2308
(one copy)

Defense Technical Information Services
Building 5, Cameron Station
Attn: Selections
Alexandria, VA 22304
(two copies)

FURTHER DISSEMINATION ONLY AS DIRECTED BY DARPA/INTO, 31 JANUARY 1989, OR HIGHER DoD AUTHORITY.

Appendix A
DRAM Truth Table

RX-1 DRAM STATE MACHINE DESCRIPTION
REV 0.1 06/23/89

Original EVAT design with bus contention fixes, WAITDIS, synchronous RPL, tcac CAS elimination, CLRCYC fix, and various 40MHz stuff. Toggle CAS during reads for page access. This design attempts to correct the DRAM WE vs CAS timing problem by delaying WEL by one cycle. Unfortunately, this created another problem by creating the same problem this time on the falling edge of CAS when transitioning from State 0 or 4 to State 17. To remedy this race condition States 0 and 4 now transition to State 19 on a Near Read and CAS is still active in State 17. Note that this change also requires some additional encoding of active signals in State 19 and 18.

STA	WD/	RAE	RAS	CAE	CAL	CAS	WCE
0	1	1	0	0	1	1	0
1	0	1	0	0	0	0	1
1	1	1	0	0	0	1	1
2	1	1	0	0	1	1	0
3	0	1	0	0	0	1	1
3	1	1	0	0	0	1	1
4	1	1	0	0	1	1	0
5	0	1	0	0	0	0	1
5	1	1	0	0	0	1	1
6	1	1	0	0	1	1	0
7	0	0	0	1	0	1	1
7	1	0	0	1	0	1	1
8	1	1	0	0	1	1	1
9	1	1	0	0	0	0	1
12	1	1	0	0	1	1	1
13	1	1	0	0	0	1	1
16	1	1	0	0	1	1	1
17	1	1	0	0	0	0	1
18	1	1	0	0	0	1	1
19	1	1	0	0	0	1	1
20	1	1	0	1	1	0	1
21	1	1	0	1	0	0	1
22	1	1	1	1	1	0	1
23	1	0	1	1	0	0	1
24	1	1	0	0	1	1	1
25	1	1	0	0	0	0	1
26	1	1	0	0	1	1	1
27	1	0	0	1	0	1	1
28	1	0	1	1	1	1	1
29	1	0	1	1	0	1	1
30	1	1	1	1	1	1	1
31	1	1	1	1	0	1	1

STA	REN	WEL	OE/	RDL	WEN	RPL	NA/
0	1	0	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
2	1	0	1	1	1	1	1
3	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1
4	1	0	1	1	0	1	1
5	1	1	1	1	0	1	1
5	1	1	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1
8	0	1	0	1	1	1	0**
9	0	1	0	0	1	0	1
12	1	1	1	1	1	1	1
13	0	1	1	1	1	0	1
16	0	1	0	1	1	1	0**
17	0	1	0	0	1	1	1
18	0	1	0	1	1	1	0
19	0	1	0	1	1	1	1
20	1	1	1	1	1	1	1
21	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1
23	1	1	1	1	1	1	1
24	0	1	0	1	1	1	0**
25	0	1	0	0	1	1	1
26	1	1	0	1	1	1	0
27	1	1	0	1	1	1	1
28	1	1	1	1	1	1	1
29	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1
31	1	1	1	1	1	1	1

STA	RDY	CLC	CRF	Ref
0	1	1	0	RW4
1	1	0	0	RW3
1	0	1	0	RW3
2	1	1	0	RW2
3	1	1	0	RW1
3	1	1	0	RW1
4	1	1	0	W4
5	1	0	0	W3
5	0	1	0	W3
6	1	1	0	W2
7	1	1	0	W1
7	1	1	0	W1
8	1	1	0	RR6
9	0	0	0	RR5
12	1	1	0	RE8
13	0	1	0	RE7
16	1	1	0	WR2
17	1	0	0	WR1
18	1	1	0	WR2
19	1	1	0	WR1
20	1	1	0	RF1
21	1	1	0	RF2
22	1	1	1	RF3
23	1	1	0	RF4
24	1	1	0	RP4
25	1	0	0	RP3
26	1	1	0	RP2
27	1	1	0	RP1
28	1	1	0	IDL
29	1	1	0	PR3
30	1	1	0	PR2
31	1	1	0	PR1

* = Block RDY to i860 unless the next cycle is a near write.
** = Block NA/ to i860 unless the next cycle is a near read.

Appendix B
DRAM State Machine Control

```
TITLE       RX-1 DRAM Control State Machine.
PATTERN     DSTAT3.PDS
REVISION    314233-001
AUTHOR      ANTHONY M. ZILKA
COMPANY     INTEL SCIENTIFIC COMPUTERS CO1-01
DATE        08/15/89

; U15
; This PAL contains the state machine for the DRAM controller.
; Original EVAT design with bus contention fixes, WAITDIS, synchronous
; RPL, tcac CAS elimination, CLRCYC fix, and various 40MHz stuff.
; This state machine transitions to state 19 when going from a write
; to a near read (state 0,4 to state 19) This is in order to eliminate
; a race condition from the end of DRAM WE in state 0,4 to the falling
; edge of the CAS read in state 17. In order to keep the clrcyc pulse
; in state 17, NA had to be moved from 0 & 4 on a near read to cycle 18.

CHIP U403 PAL20R6;-7

CLKA /LADS /EWAIT /DSEL /NENE /RESET /CPEN /WAITDIS /EXPBSY WR /
REFREQ GND /OE /FBSY /NA /S4 /S3 /S2 /S1 /S0 /DRAMBSY /RDY /TP15A VCC

EQUATIONS

S0      := S4 * /S3 * /S2 * S1 * S0 * LADS * /WR * DSEL * NENE * /REFREQ
           * /RESET
           + S4 * /S3 * /S2 * S1 * S0 * CPEN * /WR * DSEL * NENE * /REFREQ *
           /RESET
           + /S4 * S3 * /S2 * S1 * S0 * /RESET
           + /S4 * /S3 * /S2 * S1 * S0 * /LADS * /CPEN * /REFREQ * /RESET
           + /S4 * /S3 * /S2 * S1 * S0 * /DSEL * /REFREQ * /RESET
           + /S4 * /S0 * /RESET
           + /S3 * /S0 * /RESET
           + /S0 * WAITDIS * /RESET
           ;Decrement when /(State 1,3,5,7) or Wait disabled

S1      := S4 * S3 * S1 * S0 * LADS * WR * DSEL * NENE * /REFREQ * /
           RESET
           + S4 * /S3 * S2 * S1 * S0 * /RESET
           + /S4 * S2 * S1 * S0 * /RESET
```

```
        + /S4 * /S3 * /S2 * S1 * S0 * /LADS * /CPEN * /REFREQ * /RESET
        + /S4 * /S3 * /S2 * S1 * S0 * /DSEL * /REFREQ * /RESET
        + /S1 * S0 * /RESET
        + S1 * /S0 * /RESET

S2      := S4 * S1 * S0 * LADS * /WR * DSEL * NENE * /REFREQ * /RESET
        + S4 * /S3 * /S2 * S1 * S0 * LADS * DSEL * NENE * /REFREQ * /RESET
        + S4 * /S3 * /S2 * S1 * S0 * CPEN * DSEL * NENE * /REFREQ * /RESET
        + /S4 * S2 * S1 * S0 * LADS * /WR * DSEL * NENE * /REFREQ * /
        RESET
        + /S4 * /S3 * /S2 * S1 * S0 * LADS * /WR * DSEL * /REFREQ * /RESET
        * /EXPBSY * /FBSY * /RESET
        + /S4 * /S3 * /S2 * S1 * S0 * CPEN * /WR * DSEL * /REFREQ * /RESET
        * /EXPBSY * /FBSY * /RESET
        + S2 * /S1 * /RESET
        + S2 * /S0 * /RESET

S3      := S4 * S3 * S1 * S0 * LADS * DSEL * NENE * /REFREQ * /RESET
        + S4 * /S3 * /S2 * S1 * S0 * LADS * WR * DSEL * NENE * /REFREQ *
        /RESET
        + S4 * /S3 * /S2 * S1 * S0 * CPEN * WR * DSEL * NENE * /REFREQ *
        /RESET
        + /S4 * /S3 * /S2 * S1 * S0 * LADS * WR * DSEL * /REFREQ * /RESET
        * /EXPBSY * /FBSY * /RESET
        + /S4 * /S3 * /S2 * S1 * S0 * CPEN * WR * DSEL * /REFREQ * /RESET
        * /EXPBSY * /FBSY * /RESET
        + /S4 * /S3 * /S2 * S1 * S0 * REFREQ * /RESET
        + S3 * /S1 * /RESET
        + S3 * /S0 * /RESET

S4      := S4 * S1 * S0 * LADS * WR * DSEL * NENE * /REFREQ * /RESET
        + S4 * /S3 * S2 * S1 * S0 * /RESET
        + S4 * /S3 * /S2 * S1 * S0 * CPEN * WR * DSEL * NENE * /REFREQ *
        /RESET
        + /S4 * S2 * S1 * S0 * /RESET
        + /S4 * /S3 * /S2 * S1 * S0 * LADS * WR * DSEL * /REFREQ * /RESET
        * /EXPBSY * /FBSY * /RESET
        + /S4 * /S3 * /S2 * S1 * S0 * CPEN * WR * DSEL * /REFREQ * /RESET
        * /EXPBSY * /FBSY * /RESET
        + S4 * /S1 * /RESET
        + S4 * /S0 * /RESET
```

```
DRAMBSY   := S4 * S3 * /RESET
           + S4 * /S3 * /RESET
           + /S4 * S3 * S2 * /RESET
           + /S4 * /S3 * S2 * /RESET
           + CPEN * /RESET

NA        = /S4 * S3 * S2 * /S1 * S0
           + S4 * /S3 * S2 * S1 * S0 * LADS * /WR * NENE * /REFREQ
           + /S4 * S2 * S1 * S0 * LADS * /WR * NENE * /REFREQ
           + /S4 * /S3 * S2 * /S1 * S0

NA.TRST   = /EXPBSY * /FBSY * DRAMBSY * TP15A * /RESET

RDY       = ;S4 * S3 * S1 * S0 * LADS * WR * NENE * /REFREQ S4 * S3 * S1 * /
           S0 * /WAITDIS
           + S4 * /S3 * S1 * /S0

RDY.TRST = /EXPBSY * /FBSY * DRAMBSY * TP15A * /RESET

; Description: This PAL is the state machine for the DRAM interface to the i860.
; There are 32 states some of which are redundant to minimize the number of
; product terms. This design supports a non-interleaved memory design using
; Static-column 256K x4 DRAMs for the main memory, and 1M x1 DRAMs for the
; parity bits.
;
; There are six types of cycles possible: far read, far write, near read,
; near write, refresh, and idle. Near read and near write cycles imply memory
; accesses with the same row address as the preceding cycle. Refresh request
; has the highest priority and will interrupt near memory cycles.
;
; Read cycles will use full pipelining, write cycles are never pipelined.
; Near read cycles will be zero wait-state when following read cycles, and
; near write cycles will be zero wait-state when following write cycles.
;
; DRAMBSY prevents other units from starting a cycle when the DRAM controller
; is busy.
;
; WAITEN prevents the least two significant bits from sequencing in order to
; add wait states for states 1,3,5 and 7.
;
; FEXBSY is the piped OR of EXPBSY and FBSY.
```

Appendix C
DRAM Control A

```
TITLE       iSC i860 DRAM Control A.
PATTERN     IDCTLA3.PDS
REVISION    314234-001
AUTHOR      ANTHONY M. ZILKA
COMPANY     INTEL SCIENTIFIC COMPUTERS CO1-01
DATE        08/07/89

; U11
; Reference Sheet 4 of the Schematics.
; Original EVAT design with bus contention fixes, WAITDIS, synchronous
; RPL, tcac CAS elimination, CLRCYC fix, and various 40MHz stuff.
;

CHIP U421 PAL20L8;-10

/BOOT A23 A22 A21 /S0 /S1 /S2 /S3 /S4 A24 /TP11 GND /RASHLD FRCPAR HOLD /
HREQ /RASP0 /RAS3 /RAS2 /RAS1 /RAS0 /KEN /DRAMNC VCC

STRING ADD0 '/A23*/A22*/A21'

EQUATIONS

/RAS0.TRST  = TP11

/RAS1.TRST  = TP11

/RAS2.TRST  = TP11

/RAS3.TRST  = TP11

/RASP0.TRST = TP11

/KEN.TRST   = TP11

/HOLD.TRST = TP11

RAS0        = /A22 * /A21 * S4 * S3 * /S2 * /S1 * /S0 ;start write
              + /A22 * /A21 * /S4 * /S3 * S2 * /S1 * /S0 ;start read
              + /S4 * S3 * /S2 * S1 * /S0 ;start refresh
              + RAS0 * S4 ;hold
              + RAS0 * S3 ;hold
              + RAS0 * S2 ;hold
              + RAS0 * RASHLD ;hold
```

```
RAS1        = /A22 * A21 * S4 * S3 * /S2 * /S1 * /S0 ;start write
            + /A22 * A21 * /S4 * /S3 * S2 * /S1 * /S0 ;start read
            + /S4 * S3 * /S2 * S1 * /S0 ;start refresh
            + RAS1 * S4 ;hold
            + RAS1 * S3 ;hold
            + RAS1 * S2 ;hold
            + RAS1 * RASHLD ;hold

RAS2        = A22 * /A21 * S4 * S3 * /S2 * /S1 * /S0 ;start write
            + A22 * /A21 * /S4 * /S3 * S2 * /S1 * /S0 ;start read
            + /S4 * S3 * /S2 * S1 * /S0 ;start refresh
            + RAS2 * S4 ;hold
            + RAS2 * S3 ;hold
            + RAS2 * S2 ;hold
            + RAS2 * RASHLD ;hold

RAS3        = A22 * A21 * S4 * S3 * /S2 * /S1 * /S0 ;start write
            + A22 * A21 * /S4 * /S3 * S2 * /S1 * /S0 ;start read
            + /S4 * S3 * /S2 * S1 * /S0 ;start refresh
            + RAS3 * S4 ;hold
            + RAS3 * S3 ;hold
            + RAS3 * S2 ;hold
            + RAS3 * RASHLD ;hold

RASP0       = S4 * S3 * /S2 * /S1 * /S0 * FRCPAR ;start write
            + /S4 * /S3 * S2 * /S1 * /S0 * FRCPAR ;start read
            + /S4 * S3 * /S2 * S1 * /S0 * FRCPAR ;start refresh
            + RASP0 * S4 * FRCPAR ;hold
            + RASP0 * S3 * FRCPAR ;hold
            + RASP0 * S2 ;hold
            + RASP0 * RASHLD ;hold

KEN         = /DRAMNC

/HOLD       = /HREQ
;
; Description:
; This PAL decodes and drives the RAS lines for the DRAMs from
; the main state machine and the i860 addresses. It drives the RAS
; lines for the 8 Mbytes on the EV-AT board.
;
; The addresses (A21-A23) are decoded and activated when the
; DRAM state machine starts a read or a write cycle. They are held
; active until the state machine enters the Precharge sequence
; (State 31). All RAS lines are active during a refresh cycle.
;
;
```

Appendix D
DRAM Control B

```
TITLE       iSC i860 DRAM Control B.
PATTERN     IDCTLB3.PDS
REVISION    314235-001
AUTHOR      ANTHONY M. ZILKA
COMPANY     INTEL SCIENTIFIC COMPUTERS CO1-01
DADATE      08/02/89

; U13
; This PAL decodes and drives the advanced DRAM control signals.
; Must be "E" speed.
; This revision is for fast past DRAMs with tcac of 20ns.
; Original EVAT design with bus contention fixes, WAITDIS, synchronous
; RPL, tcac CAS elimination, CLRCYC fix, and various 40MHz stuff.
; CAS is toggled for page access.
; This PAL corrects the race condition between the rising edge of CAS
; and the termination of the DRAM WE by delaying WEL by one cycle.
; This causes a race condition between the termination of DRAM WE and the
; falling edge of CAS when transitioning from writes to a near read (0,4 to 17)
; This forces the addition of states 19 and 18 in the Write-Read sequence.
; Note the encoding of signals active in state 19 and 18 in addition to the
; delay of WEL.

CHIP U411 PAL20R8;-7

DCLK /ERDL P3 /S0 /S1 /S2 /S3 /S4 /RTDS WR /WAITDIS GND /OE P14 /RASHLD /
WCE /CAE /CLRCYC /CSX /RDL /CAL /WEL /EWDL VCC

EQUATIONS

; RTAB= RTDS
; RTBA= RTDS
; EXP FUNCTIONALITY LOST DUE
; TO USE OF 2952

RDL         := S4 * /S3 * S2 * S1 * /S0   ;state 9
               + /S4 * S3 * S2 * S1 * /S0   ;state 17
               + /S4 * /S3 * S2 * S1 * /S0  ;state 25
               + ERDL
```

```
WEL     := S4 * S3 * S2 * S1 * S0 ;state 0
           + S4 * S3 * S2 * /S1 * S0 ;state 2
           + S4 * S3 * /S2 * S1 * S0 ;state 4
           + S4 * S3 * /S2 * /S1 * S0 ;state 6

WCE     := S4 * S3 * S2 * S1 * S0 ;state 0
           + S4 * S3 * S2 * /S1 * S0 ;state 2
           + S4 * S3 * /S2 * S1 * S0 ;state 4
           + S4 * S3 * /S2 * /S1 * S0 ;state 6

CSX     := S4 * S3 * S2 * S1 * /S0 * waitdis;state 1(, 3)
           + S4 * S3 * /S2 * S1 * /S0 * waitdis;state 5
           + S4 * /S3 * S2 * S1 * /S0 ;state 9
           ;+ S4 * /S3 * /S2 * S1 ;state 12,13 (No Need)
           + /S4 * S3 * S2 * S1 * /S0 ;state 17
           + /S4 * S3 * /S2 ;state 22,23,(20,21)
           + /S4 * /S3 * S2 * S1 * /S0 ;state 25
           ;+ /S4 * /S3 * S2 * /S1 * S0;state 26

CAL     := /S0; all odd states
           + /S4 * S3 * S2 * /S1 * S0; or state 18

CAE     := S4 * S3 * S2 ;states 0,1,2,3
           + S4 * S3 * /S2 * S1 ;states 4,5
           + S4 * S3 * /S2 * /S1 * S0;state 6
           + S4 * /S3 * S1 ;states 8,9,12,13
           + /S4 * S3 * S2 ;states 16,17,18,19
           + /S4 * /S3 * S2 * S1 ;states 24,25
           + /S4 * /S3 * S2 * /S1 * S0;state 26

CLRCYC  := S4 * S3 * S1 * /S0 * WAITDIS;state 5,1 (Moved
           + S4 * /S3 * S2 * S1 * /S0;state 9 from
           + /S4 * S3 * S2 * /S1 * /S0;state 17 IDCTLC0)
           + /S4 * /S3 * S2 * S1 * /S0;state 25

RASHLD  := S4 * /S3 * /S2 * S1 * S0 ;state 12
           + /S4 * S3 * S2 * S1 * S0;state 16
           + /S4 * /S3 * S2 * S1 * S0;state 24

;   Description: Refer to the DRAM State Machine document for an explanation
;   of when these signals are to be active.
;
;   CAEColumn Address Enable. Turns on the Column address drivers.
;
;   CALColumn Address Latch. Latches the i860 addresses for the DRAM
;   column address.
;
```

```
;   CSXChip Select for the DRAM array.
;
;   WELWrite Enable Latch. Latches the BEN bits to drive the active
;   DRAM WEs.
;
;   WCEWrite Data Latch CLK enable. For DRAM accesses only.
;   Used with 2952 to reduce errors for i860 data transfers.
;
;   RDL Read Data Latch. For DRAM accesses only.
;
;   CLRCYC Clear Cycle. Clears queued cycles.
```

Appendix E
DRAM Control D

```
TITLE       iSC i860 DRAM Control D.
PATTERN     IDCTLD0.PDS
REVISION    314237-001
AUTHOR      ANTHONY M. ZILKA
COMPANY     INTEL SCIENTIFIC COMPUTERS CO1-01
DATE        03/03/89

;This PAL decodes and drives the DRAM Write Enable signals.
;Must be "E" speed.
;Original EVAT design with bus contention fixes, WAITDIS, synchronous
;RPL, tcac CAS elimination, CLRCYC fix, and various 40MHz stuff.
;

CHIP U420 PAL20R8;-7

/WEL /CPEN /DSEL /BE0 /BE1 /BE2 /BE3 /BE4 /BE5 /BE7 /BE6 GND /OE WR /WE7 /
WE6 /WE5 /WE4 /WE3 /WE2 /WE1 /WE0 /LADS VCC

EQUATIONS

WE7     := LADS * DSEL * WR * BE7
         + CPEN * DSEL * WR * BE7

WE6     := LADS * DSEL * WR * BE6
         + CPEN * DSEL * WR * BE6

WE5     := LADS * DSEL * WR * BE5
         + CPEN * DSEL * WR * BE5

WE4     := LADS * DSEL * WR * BE4
         + CPEN * DSEL * WR * BE4

WE3     := LADS * DSEL * WR * BE3
         + CPEN * DSEL * WR * BE3

WE2     := LADS * DSEL * WR * BE2
         + CPEN * DSEL * WR * BE2

WE1     := LADS * DSEL * WR * BE1
         + CPEN * DSEL * WR * BE1

WE0     := LADS * DSEL * WR * BE0
         + CPEN * DSEL * WR * BE0
```

; Description: This PAL samples the BEN lines during write cycles and ; drives the DRAM
Write Enables directly.

Appendix F
DRAM Control D

```
TITLE       iSC i860 DRAM Control D.
PATTERN     IDCTLD0.PDS
REVISION    314237-001
AUTHOR      ANTHONY M. ZILKA
COMPANY     INTEL SCIENTIFIC COMPUTERS CO1-01
DATE        03/03/89

;This PAL decodes and drives the DRAM Write Enable signals.
;Must be "E" speed.
;Original EVAT design with bus contention fixes, WAITDIS, synchronous
;RPL, tcac CAS elimination, CLRCYC fix, and various 40MHz stuff.
;

CHIP U420 PAL20R8;-7

/WEL /CPEN /DSEL /BE0 /BE1 /BE2 /BE3 /BE4 /BE5 /BE7 /BE6 GND /OE WR /WE7 /
WE6 /WE5 /WE4 /WE3 /WE2 /WE1 /WE0 /LADS VCC

EQUATIONS

WE7     := LADS * DSEL * WR * BE7
           + CPEN * DSEL * WR * BE7

WE6     := LADS * DSEL * WR * BE6
           + CPEN * DSEL * WR * BE6

WE5     := LADS * DSEL * WR * BE5
           + CPEN * DSEL * WR * BE5

WE4     := LADS * DSEL * WR * BE4
           + CPEN * DSEL * WR * BE4

WE3     := LADS * DSEL * WR * BE3
           + CPEN * DSEL * WR * BE3

WE2     := LADS * DSEL * WR * BE2
           + CPEN * DSEL * WR * BE2

WE1     := LADS * DSEL * WR * BE1
           + CPEN * DSEL * WR * BE1

WE0     := LADS * DSEL * WR * BE0
           + CPEN * DSEL * WR * BE0
```

; Description: This PAL samples the BEN lines during write cycles and
; drives the DRAM Write Enables directly.

Appendix G
i860 Access1

Title i860ACCESS1

Revision 374238-002

Pattern n10ac1r6.pds

Author LOC NGUYEN, ANTHONY ZILKA

Company Intel Corporation

Date 10/04/89

```
; Far reads or writes, without NENE, must branch upon FIFOSEL and LADS,
; ELSE go back to IDLE. Recoding of READ states for MSB product term
; overflow.
; REV 1. Update signal NA.
; REV 2. Remove FIFO return to state B. Limit the FIFO page to DRAM page.
; REV 5. Fix i860EOD hold term, eliminate restrictions by decoding off of
; fifosel.
; REV 6. Back to state b on absence of NENE.

chip U230 pal16r6;-7

clk /nene wr /bsy /fifosel /reset /lads /expbsy a27 gnd /oe /n10rdy /n10eod /fbsy /s3 /s2 /s1
/s0 /na vcc

string sa ' /s3 * /s2 * /s1 * /s0 ' ;IDLE
string sb ' /s3 * /s2 * /s1 * s0 ' ;
string sc ' /s3 * /s2 * s1 * /s0 ' ;
string sh ' s3 * /s2 * s1 * s0 '
string si ' s3 * s2 * /s1 * /s0 ' ;NENE
string sj ' s3 * s2 * /s1 * s0 ' ;read
string sk ' s3 * s2 * s1 * /s0 ' ;sequence.
string sd ' /s3 * /s2 * s1 * s0 '
string se ' /s3 * s2 * /s1 * /s0 '
string sf ' /s3 * s2 * /s1 * s0 ' ;NENE
string sg ' /s3 * s2 * s1 * /s0 ' ;write sequence

equations

/na.trst      = fbsy

/n10rdy.trst  = fbsy
```

```
s3      := sc * /wr * /reset ;goto sh
        + sh * /reset ;goto si
        + si * /wr * lads * nene * /reset ;goto sj
        + sj * /reset ;goto sk
        + sk * /reset ;goto si

s2      := sd * /reset ;state sd
        + se * /reset ;state se
        + sf * wr * lads * nene * /reset ;state sf to state sg
        + sg * /reset ;state sg
        + sh * /reset ;state sh
        + si * /wr * lads * nene * /reset ;state si to state sj
        + sj * /reset ;state sj
        + sk * /reset ;state sk

s1      := sb * /bsy * /expbsy * /reset ;goto sc
        + sc * /reset ;goto sd, sh
        + sf * lads * wr * nene * /reset ;goto sg
        + sj * /reset ;state sj

s0      := sa * fifosel * lads * /reset ;kick off
        + sb * bsy * /reset ;stay in sb
        + sb * expbsy * /reset
        + sc * /reset ;goto to sd, sh
        + se * /reset ;se to state sf
        + sg * /reset ;sg to state sf
        + si * lads * fifosel * /reset ;state si to sj, sb
        + sf * lads * fifosel * /nene * /reset ;state sf to sb

n10eod  := wr * a27 * lads * fifosel * /reset
        + n10eod * sa * fifosel * /reset
        + n10eod * sb * fifosel * /reset
        + n10eod * sc * fifosel * /reset

fbsy    := sb * fifosel * /reset
        + fbsy * s3 * /reset
        + fbsy * s2 * /reset
        + fbsy * s1 * /reset
        + fbsy * s0 * /reset

na      = sc * /wr * /reset ;pipe on read cycles.
        + sj * /reset

n10rdy  = si * /reset
        + sd * /reset
        + sf * nene * wr * lads * /reset
```

```
;FAST VERSION
;Description
; This pal implements the state machine control for i860 access of the FIFOs.
; SA is the idle state, which is exited upon a FIFOSEL and LADS. SB is entered
; after SA, and here the state machine waits until the busy signals of other
; piped accesses go away. The state machine then sequences through state SC
; to SF to complete the first write. In state SF subsequent near writes
; signified by NENE cycle between SF and SG. Far reads or writes detected in
; state SF return to state SB. The absence of FIFOSEL and LADS in state SF
; indicating no subsequent FIFO accesses return the state machine to idle.
; FIFO reads sequence from SB through SC, SH and SI returning NA to the i860
; to initiate
; another accesses while completing the first read. For subsequent near
; reads the
; state machine cycles from SI to SJ and SK and back to SI returning NA for
; another i860 access and providing the read data. Far reads or writes are
; detected in state SI in the absences of NENE where upon the state machine
; branches to state SB and again sequences through SI or SF. An absence of
; FIFOSEL and LADS in state SI returns the state machine to idle.
```

Appendix H
i860 Access2

Title i860ACCESS2

Revision 314239-001

Pattern n10ac2r3.pds

Author NGUYEN, ANTHONY ZILKA

Company Intel Corporation

Date 08/04/89

```
; Coding for states sh, si, sj, and sk changed in n10ac1r2.pds
; REV 2. add hold term for eodlast
; REV 3. eodlast is sampled in sk and sh instead of si.

chip U231 pal16r6;-7

clk /nene wr /s0 /s1 /s2 /s3 /lads /eodin gnd /oe p12 eodlast /frdclk /frdena /fwrena /rfrd /
xfwr /fwclkena vcc

string sa ' /s3 * /s2 * /s1 * /s0 '
string sb ' /s3 * /s2 * /s1 * s0 '
string sc ' /s3 * /s2 * s1 * /s0 '
string sh ' s3 * /s2 * s1 * s0 ' ;1011
string sj ' s3 * s2 * /s1 * s0 ' ;1101
string si ' s3 * s2 * /s1 * /s0 ' ;1100
string sk ' s3 * s2 * s1 * /s0 ' ;1110
string sd ' /s3 * /s2 * s1 * s0 '
string se ' /s3 * s2 * /s1 * /s0 '
string sf ' /s3 * s2 * /s1 * s0 '
string sg ' /s3 * s2 * s1 * /s0 '

equations

xfwr    := sc * wr
         + sf * nene * wr * lads

rfrd    := sb * /wr
         + sc * /wr
         + si * nene * /wr * lads
         + sj
```

```
frdena      := sc * /wr
            + sh
            + sj
            + sk

frdclk      := sc * /wr
            + sj

fwrena      := sb * wr
            + sc * wr
            + sd
            + se
            + sf * nene * wr * lads
            + sg

fwclkena    = /xfwr ;is xfwr delayed to meet 29F52 CPB hold time.

/eodlast    := /eodin * /eodlast
            + s3 * s2 * s1 * /s0 * /eodin ;eodlast := sk * eodin
            + s3 * /s2 * s1 * s0 * /eodin ;eodlast := sh * eodin
            + /eodlast * /s3
            + /eodlast * /s1

;
; description
; This version is the fast version of the i860-FIFO State Machine.
```

Appendix I
Local Access

```
Title       Local Access
Revision    314232-001
Pattern     lclacc1.pds
Author      LOC NGUYEN, ANTHONY ZILKA
Company     Intel Corporation
Date        07/21/89

; Addition of state g between a and b which synchs to /clk64
; wait until state machines which can be piped are not busy.
; Additional synch state is required for minimum worst case
; synch possibility. BUFENA was made synchronous in order
; to use an asynchronous output as an input (LADS).
; Start SM with LACCGO, don't enable buffer when /LOCSEL1 and
; LOCSEL0.
;

chip U211 pal16r6;-7

clk /reset /locsel1 /locsel0 /expbsy wr /bsy clk64 /fbsy gnd /oe /laccgo /n10wt /c0 /c1 /c2 /
prtstrb /bufena /n10rdy vcc

string sa '/c2 * /c1 * /c0 ' ;state A = 000 Idle
string sb '/c2 * /c1 * c0 ' ;state B = 001 Second synch to CLK64
string sc '/c2 * c1 * c0 ' ;state C = 011 Wait for CLK64 phase
string sd ' c2 * c1 * c0 ' ;state D = 111
string se ' c2 * /c1 * c0 ' ;state E = 101
string sf ' c2 * /c1 * /c0 ' ;state F = 100
string sg '/c2 * c1 * /c0 ' ;state G = 010 First synch to /CLK64 or /BSY

equations

n10rdy.trst    = c2

c0             := sg * locsel1 * /clk64 * /bsy * /fbsy * /expbsy * /reset ;goto B
                  + sg * /locsel1 * /bsy * /fbsy * /expbsy * /reset ;goto B
                  + sb * /reset ;always
                  + sc * /reset ;always
                  + sd * /reset ;always
```

```
c1          := sa * laccgo * /reset ;goto G
            + sg * locsel1 * clk64 * /reset ;stay at G
            + sb * locsel1 * clk64 * /reset ;goto B
            + sb * /locsel1 * /reset ;goto B
            + sc * /reset ;always
            + sg * expbsy * /reset
            + sg * fbsy * /reset
            + sg * bsy * /reset

c2          := sc * locsel1 * /clk64 * /reset ;goto D
            + sc * /locsel1 * /reset ;goto D
            + sd * /reset ;always
            + se * /reset ;always

n10rdy      = se * /reset ;active in state E, combinatorial

; prtstrb   := sc * /reset ;start once in C
            + sd * /wr * /reset ;active in state E on READ cycles only

bufena      := sb * locsel1 * clk64 * /reset ;enable except for
            + sb * /locsel1 * wr * /reset ;counter reads
            + sb * /locsel1 * /locsel0 * /reset ;
            + sc * bufena * /reset ;
            + sd * bufena * /reset ;

n10wt       := sb * wr * /reset ;change in state B
            + n10wt * c2 * /reset ;hold for the rest
            + n10wt * c1 * /reset
            + n10wt * c0 * /reset

;description
; This pal controls i860 accesses to 'local' space. The local space is
; encoded from the address bus with locsel as an enable, and locsel1
; generally differentiating between devices that require a large amount of
; time to access, such as the EPROMs or UART vs a small amount of time for
; status or control registers.
; The state machine leaves idle upon a locsel and lads. It waits in the next
; state for other piped state machines to complete by watching the busy
; signals. If the access is to a 'slow' device the state machine also waits
; for the start of a CLK64 cycle to ensure enough time is allocated for the
; access.
; After leaving this wait state the state machine completes its sequence
; after allowing a CLK64 cycle for 'slow' devices or ignoring CLK64 for 'fast'
; devices.
```

Appendix J
DCM Access

Title DCMACCESS

Revision 314240-003

Pattern dcmacc5.pds

Author LOC NGUYEN, ANTHONY ZILKA, ROGER TRAYLOR

Company Intel Corporation

Date 09/05/89

```
;REV - New EOD count clock enables (COUNTUP, COUNTDOWN) and direction logic.
;REV - State changes occur on positive phase of clk2. Added wait state for
; dcm receiver chip, spin condition for xmit chip. DCM inputs to
; pal changed to positive polarity. (RT)
;REV - Inverted xrdy and rrdy for XIlinx modification n10bit01.
;REV - Added condition to xfrd and rfrd for diagnostic loopback capability.
;REV - countdown needs clk2, not /clk2, and state bit modifier
;REV 003 - no count when counting down and up, remove CLK2 reset.

chip U131 pal20r8;-7

;pins 1 2 3 4 5 6 7 8 9 10 11 12
clk p2 /fwrap /reod /reset /lxfemp /lrffull /xrdy /rrdy /eodin /rfrd gnd
;pins 13 14 15 16 17 18 19 20 21 22 23 24
/oe p14 /xfrd /xack /fs3 eodu /eodcnt clk2 /rack /rfwr /frdclk vcc

string sa ' /fs3 * /xack * /xfrd * /rfwr * /rack * /reset ' ;IDLE
string xmit ' /fs3 * xack * xfrd * /rfwr * /rack * /reset ' ;XMIT state
string receive ' /fs3 * /xack * /xfrd * rfwr * rack * /reset ' ;RECEIVE state
string wrap ' /fs3 * /xack * xfrd * rfwr * /rack * /reset ' ;WRAP state
string ping ' fs3 * /xack * /xfrd * /rfwr * /rack * /reset ' ;RCV-PING-XMT
string rcvwait ' fs3 * /xack * /xfrd * rfwr * rack * /reset ' ;RCV_WAIT


string countdown ' rfwr * clk2 * reod * /fs3';Enable counter when FIFO reads EOD
string countup ' frdclk * eodin ' ;Enable counter when i860 removes EOD

equations

/clk2 := clk2
```

```
eodcnt      := countdown * /frdclk ;EOD is received
             + countdown * /eodin
             + countup * /rfwr ;i860 takes EOD
             + countup * /clk2
             + countup * /reod
             + countup * fs3

/eodu       := countdown ;count down when EOD is received.

fs3         := receive * clk2 * xrdy * /lxfemp ;receive to ping state xition
             + ping * /clk2 ;spin in ping state
             + receive * /clk2 ;going to wait state
             + rcvwait * clk2 ;spin in wait state

xack        := sa * clk2 * xrdy * /lxfemp * /rrdy * /fwrap ;idle to xmit xition
             + sa * clk2 * xrdy * /lxfemp * lrffull * /fwrap ;idle to xmit xition
             + xmit * /clk2 ;spin in xmit while clk2 is invalid
             + xmit * xrdy ;or xrdy is still asserted
             + ping * clk2 ;ping to xmit xition

xfrd        := sa * clk2 * xrdy * /lxfemp * /rrdy * /fwrap ;idle to xmit xition
             + sa * clk2 * xrdy * /lxfemp * lrffull * /fwrap ;idle to xmit xition
             + xmit * /clk2 ;spin in xmit while clk2 is invalid
             + xmit * xrdy ;or xrdy is still asserted
             + sa * clk2 * fwrap * /lxfemp * /lrffull ;idle to wrap xition
             + wrap * /clk2 * /reset ;spin in wrap state
             + ping * clk2 ;ping to xmit xition

rfwr        := sa * clk2 * rrdy * /lrffull * /fwrap ;idle to rcv xition
             + receive * /clk2 * /fwrap ;rcv to wait xition
             + rcvwait * clk2 ;spin in wait state
             + rcvwait * /clk2 ;wait to rcv xition
             + sa * clk2 * fwrap * /lxfemp * /lrffull ;idle to wrap xition
             + wrap * /clk2 * /reset ;spin in wrap state

rack        := sa * clk2 * rrdy * /lrffull * /fwrap ;idle to rcv xition
             + receive * /clk2 ;rcv to wait xition
             + rcvwait * clk2 ;spin in wait state
             + rcvwait * /clk2 ;wait to rcv xition

;Description
; This pal implements state machine control of the interface between
; the i860 FIFOs and the DCM. The DCM asserts RRDY
; and/or XRDY and data is read or written from or to the FIFOs and
; the appropriate transfer acknowledged. If both receive and transmit
; are requested simultaneously, data is first received then alternates
```

; between transmitting and receiving.
; Setting the WRAP bit wraps data from the receive FIFO to the XMIT FIFO
; without regard to the DCM request signals.

Appendix K
General Purpose Clock

Title GENERAL-PURPOSE-CLOCK

Revision 314241-002

Pattern gpclk2.pds

Author LOC NGUYEN, ANTHONY ZILKA

Company Intel Corporation

Date 10/06/89

```
; Addition of state in Bus Timeout state machine to synch up
; to FTIMETC, uses a phase of CLK2 so as not to fall through without
; waiting for TC again. Synchronize RDY so that it is only active for
; one cycle, this forces elimination of RESET. Add another states so
; RDY can go high before it is tristated. Delete Refresh requests
; during RESET.
; Change polarity of CLK2.
;07/13/89, add hold term to bustout so that it will remain asserted until
;clear while the bus timeout state machine continues to run.
;REV 002 - Grey code state C for bustimeout signal, delete state F.

chip U210 pal20r6;-7

clk lpdet /clrpar /clrbus refdef /ftc clk2 f6 f7 reset clrfrq gnd /oe /ads bustout /refreq /ref1 /
sta0 /sta1 /sta2 parint /n10rdy s43 vcc

string sra ' /ref1 * /refreq ' ;00 IDLE
string srb ' /ref1 * refreq ' ;01 Refresh Request, wait for CLR
string src ' ref1 * /refreq ' ;10 Wait for beginning of cycle
string sta ' /sta2 * /sta1 * /sta0 ' ;000 IDLE
string stb ' /sta2 * /sta1 * sta0 ' ;001 synch to first TC
string stc ' /sta2 * sta1 * sta0 ' ;011 synch to second TC
string std ' sta2 * sta1 * /sta0 ' ;110 sta1 is RDY
string ste ' sta2 * /sta1 * /sta0 ' ;100 sta2 is RDY Enable

equations

refreq      := sra * s43 * f7 * /refdef * /reset ;40 MHz, sra to srb
            + sra * /s43 * f6 * /refdef * /reset ;30 MHz, sra to srb
            + srb * /clrfrq * /reset ;stay in srb
```

```
ref1        := srb * clrfrq * /reset ;srb to src
            + src * s43 * f7 * /reset ;stay in src
            + src * /s43 * f6 * /reset ;stay in src

sta0        := sta * ads * /reset ;start
            + stb * /n10rdy * /reset ;synch to tc
            + stc * /ftc * /n10rdy * /reset

sta1        := stb * /n10rdy * ftc * /clk2 * /reset ;synch to tc
            + stc * /n10rdy * /reset ;wait for tc phase

sta2        := stc * /n10rdy * ftc * /reset ;enable RDY when active
            + std * /reset ;enable RDY when inactive

/bustout    = sta0 * /bustout
            ;/(sta2 * /sta1 * /sta0 + bustout * /clrbus)
            + sta1 * /bustout
            + /sta2 * /bustout
            + sta0 * clrbus
            + sta1 * clrbus
            + /sta2 * clrbus

n10rdy      = sta1 ;Bidirectional I/O

n10rdy.trst = sta2 * /reset ;Enable RDY in std

/parint     := /lpdet * /parint * /reset ;/( lpdet * /clrpar * /reset
            + clrpar * /reset ; parint * /clrpar * /reset)
            + reset
```

;Description
; BUSTOUT is the Bus Timeout signal. After synching with FTC and waiting
; an entire FTC cycle after initiating an ADS, if RDY is not returned
; the Bus Timeout state machine return RDY and sets the Bus Timeout signal.
; The Bus Timeout interrupts the 860 and can be read as a status bit.
; REFREQ is the Refresh Request signal to the DRAM controller. After reset
; refresh cycles are requested after an appropriate time,
; in this pal: refresh rate at 40 MHz = 12.8 us
; 30 MHz = 8.4 us , OVER KILL!
; The REFDEF control bit disables the requesting of refresh cycles.
; This PAL also latches the parity error signal which causes and interrupt to
; the processor and can be read as a status bit. The parity error is cleared
; by the CLRPAR control bit.
; Finally, this PAL synchronizes the reset signal.

Appendix L
Interrupt Controller

Title Interrupt Controller

Revision 314242-002

Pattern intcntr1.pds

Author LOC NGUYEN, ANTHONY ZILKA

Company Intel Corporation

Date 08/24/89

```
; Separate masking of EODINF with /EODMSK and /LRHF with /RMSK.
; Added reset into interrupt (n10rst)
; /LXHF should be LXHE

chip U130 pal16l8;-10

/eodmsk /xmsk /rmsk eodinf /lrhf lxhe parint /lexpint lserint gnd bustout rn10int xint n10rst
/rn10int1 /expmsk /hwmsk /sermsk rint vcc

equations

/xint       = /lxhe * /xmsk ;/[lxhe * /xmsk)
              + xmsk

rn10int1    = lexpint * /expmsk
              + parint * /hwmsk
              + bustout * /hwmsk
              + lserint * /sermsk
              + lrhf * /rmsk ;rint
              + eodinf * /eodmsk
              + xint

/rint       = /lrhf * /eodinf ;/[(lrhf * /rmsk) + (eodinf * /eodmsk)]
              + rmsk * /eodinf
              + /lrhf * eodmsk
              + rmsk * eodmsk

/rn10int    = /rn10int1 * /n10rst

;Description
; XINT is the transmit interrupt bit read by the status port. It is masked by
; XMSK.
; RINT is the receive interrupt bit read by the status port. It is true when
; the receive FIFO is half full or when there are EODs in the receive FIFO.
```

```
; These interrupt bits can be masked individually with RMSK and EODMSK.
; Ri860INT1 is the signal that is synchronized and sent to the i860.
; It is comprised of the expansion interrupt (LEXPTINT), a parity error
; interrupt (PARINT), a bus time out interrupt (BUSTOUT), an interrupt from
; the UART (LSERINT), and receive and transmit FIFO interrupts (RINT, XINT).
; Each of these interrupts can be masked.
```

Appendix M
Address Decode 1

Title Address Decode 1

Revision 374230-001

Pattern addec1r0.pds

Author LOC NGUYEN, ANTHONY ZILKA

Company Intel Corporation

Date 06/06/89

```
; Elimination of selects latched by LADS. Each state machine
; uses LADS with select to come out of Idle. The DRAM machine is not
; shallow enough
; and utilizes a pending bit. Each machine only starts after
; busys from piped machines are disabled.
; LOCSEL/ on output pin 16. LOCSEL0/ on output pin 14.

chip U111 pal16l8;-10

p1 p2 a27 a28 a29 a30 a31 p8 /pd gnd /boot p12 /fifosel /locsel0 /locsel1 /locsel /expsel /
dramnc /dramsel vcc

equations

/dramsel.trst   = pd

/dramnc.trst    = pd

/expsel.trst    = pd

/locsel.trst    = pd

/locsel1.trst   = pd

/locsel0.trst   = pd

/fifosel.trst   = pd

dramsel         = a31 * a30 * a29 * a28 * a27 * /boot ;F8XX XXXX DRAM over EPROM
                + a31 * a30 * a29 * a28 * /a27 ;F0XX XXXX DRAM
                + a31 * a30 * a29 * /a28 ;EXXX XXXX DRAM
                + a31 * a30 * /a29 * a28 ;DXXX XXXX DRAM no cache

dramnc          = a31 * a30 * /a29 * a28 ;DXXX XXXX DRAM no cache

fifosel         = /a31 * /a30 * a29 * a28 ;3XXX XXXX FIFO
```

```
locsel1     = /a31 * a30 * a29 * /a28 ;6XXX XXXX UART (serial chnl)
            + a31 * a30 * a29 * a28 * a27 * boot ;F8XX XXXX EPROM

locsel0     = /a31 * a30 * a29 * /a28 ;6XXX XXXX UART (serial chnl)
            + /a31 * a30 * /a29 * a28 ;5XXX XXXX PORT A

locsel      = /a31 * a30 * a29 * /a28 ;6XXXX XXXX UART (serial chnl)
            + a31 * a30 * a29 * a28 * a27 * boot ;FXXXX XXXX EPROM
            + /a31 * a30 * /a29 * a28 ;5XXXX XXXX PORT A
            + /a31 * a30 * /a29 * /a28 ;4XXXX XXXX PORT B

expsel      = /pd

; description
; FIFO, dram, and local cycles only sample select lines during LADS. They
; will enter their pending state accordingly waiting until any other
; piped state machine becomes not busy. These select lines
; are OK to be held by LADS. In this case, during i860 Idle conditions, these
; select lines may be active but the corresponding state machines (Dram,
; Local, and FIFO) should not start a cycle based upon only selects without
; LADS.
```

Appendix N
Address Decode 2

```
Title       Address Decode 2
Revision    314231-001
Pattern     Addec2r1.pds
Author      LOC NGUYEN, ANTHONY ZILKA
Company     Intel Corporation
Date        08/15/89
; Addition of CLK2
; LOCSEL/ on input pin 2. LOCSEL0/ on input pin 4.
; Local select added.
;
chip U110 pal16r8;-7
clk /locsel /locsel1 /locsel0 /reset p6 p7 /pd /ads gnd /oe /uartcs clk2 /llsel1 /llsel0 /lads3 /
lads2 /lads1 /lselgo vcc
equations
lads3       := ads * /reset
lads2       := ads * /reset
lads1       := ads * /reset
llsel0      := locsel * locsel0 * /reset
llsel1      := locsel * locsel1 * /reset
uartcs      := locsel * locsel1 * locsel0 * /reset
/clk2       := clk2 * /reset
lselgo      := locsel * lads1 * /reset ;start lclacc state machine
;description
;locsel locsel1 locsel0 Read Write
; 0 0 0 uart uart
; 0 0 1 prom counter
; 0 1 0 counter port A
; 0 1 1 status control
```

Appendix O
Signal Definition

CLKA	40Mhz system clock driving DRAM control logic.
CLKB	40Mhz system clock driving local access control and general purpose clock logic.
CLKC	40Mhz system clock provided to expansion port.
CLKD	40Mhz system clock driving i860 CPU.
CLKE	40Mhz system clock driving FIFO logic.
CLKF	40Mhz system clock driving DRAM data transceivers.
CLKG	40Mhz system clock driving counters.
CLKH	40Mhz system clock driving FIFO data transceivers.
HOLD	i860 input that requests ownership of bus.
KEN/	i860 input that enables/disables internal caching.
i860INT	i860 interrupt input, logical or of all RX node interrupts which is double synched.
i860NA/	i860 next address input.
i860RDY/	i860 ready input.
i860RESET	Active high node reset signal.
SHI	Unused i860 input.
Bi860RST/	Active low node reset.
BOOT/	Status port bit that specifies boot or i860 CS8 mode. When active this bit maps portion of address space as PROM instead of RAM and permits writing to the counter.
BSY/	DRAM busy signal. Active when DRAM control is in operation. This signal prevents other state machines from starting until the DRAM operation has completed.
BUFENA/	Output enable signal to eight bit bidirection buffer between lowest byte of i860 data bus and the BD data bus which connects to UART, PROM, Control and Status ports.
CLK/64	625kHz clock to local access PAL to provide timing for 'slow' access devices (UART and PROM).

CONTWR/	Control port write strobe. Latches data from i860 bus into node control registers.
COUNTRD/	Counter read strobe. Enables counter data onto the i860 data bus.
COUNTWR/	Counter write strobe. Latches data from i860 bus into node counter.
DRAMNC/	DRAM no cache signal. Signals address decoding of no cache memory space. This signal is buffered and sent to the i860 as KEN/.
DRAMSEL/	DRAM select signal. Address decoding of DRAM space.
EXPBSY/	Expansion busy. Signals operation of the expansion port.
EXPSEL/	Expansion select. Address decoding of expansion space.
FBSY/	FIFO busy. Signals operation of FIFO state machine control.
FIFOSEL/	FIFO select. Address decoding of FIFO space, including EOD.
LACCGO/	Local access go. Initiates local access state machine operation. This signal is generated to minimize inputs to LCLACC PAL. It is a combination of LADS and an address decode to local space.
LADS<3:1>/	Multiple copies of latched ADS.
LCLK/2	20Mhz clock generated
LOCSEL<1:0>	Encoded bits specifying memory mapping of 'local' memory space functions.
LOCSEL/	Encoded bit specifying memory mapping of 'local' memory space functions.
LLSEL<1:0>/	Latched LOCSEL<1:0>/ signals.
i860WT/	Encoded local control signal specifying local control read or write.
PAWR/	Port A (Performance Port) write strobe.
PROMOE/	PROM output enable strobe.
PRTSTRB/	Local control strobe, decoded as read or write strobe to appropriate local device.
STATRD/	Status port read strobe.
UARTCS/	UART chip select strobe.

UARTRD/	UART read strobe.
UARTWR/	UART write strobe.
40/33#	Node frequency specifier, set high for 40MHZ operation. Specifies wait states in DRAM control, and refresh count.
CLRCYC/	Clear cycle. Used in DRAM control to clear pending DRAM bus transactions when they were initiated before the DRAM controller was able to act upon them.
CPEN/	Cycle pending. Signal to DRAM controller that a bus transaction is pending. This bit is set upon an initiation of a DRAM bus transaction.
DCLK	Delayed clock. System clock delayed by 15ns (or advanced by 10ns). Used to place DRAM control signals at times other than system clock edges.
EWAIT/	Expansion port signal to DRAM control state machine.
FRCPAR	Force parity. This signal must be active in order to allow the operation of the RAS signal to the parity DRAMs. After reset this signal is inactive, disabling the parity DRAM bank.
HREQ/	Hold request. Signal driven from the expansion port which is inverted and driven to the i860.
RASHLD/	RAS hold. The RAS control signal has asynchronous feedback and is susceptible to race conditions on certain state transitions. RAS hold holds the RAS signals active during non-Grey code transitions of the DRAM state machine where RAS must not change.
RDL/	Read data latch. The DRAM read access is not quite fast enough to latch data on the system clock edge. RDL is generated and delayed to occur just after the system clock edge to permit a little bit more DRAM read access time at the expense of transfer time to the i860.
REFREQ/	Refresh request. Signal indicating that a refresh sequence be performed by the DRAM state machine.
WAITDIS/	Wait disable. The DRAM state machine will not advance beyond states 7,5,3 and 1 until receiving the wait disable signal. In 40mhz operation the wait disable signal is active for one cycle after entering states 7,5,3 and 1.
WCE/	Write clock enable. DRAM control signal which enables the write clock in the DRAM data transceivers.

WEL/	Write enable latch. Clock signal to DCTLD PAL which clocks the write control lines to the DRAMs.
CSP/	CAS parity. CAS control signal to parity DRAMs.
INTUSM	Node interrupt driven to USM.
PA10	Parity DRAM address bit 10. Used only for 4Mbit DRAMS.
SINTUSM	Intermediate USM interrupt signal before driven by tri-state driver to USM.
SINTUSM/	Inverted polarity of SINTUSM. Inverted to create SINTUSM.
TP103	10-1
PAREDET	Parity error detect. Signal that is driven when any byte parity error is detected on a DRAM read.
EODIN/	End of message in. End of message specifier of receive FIFOs.
FRDCLK	FIFO read clock. Signal that clocks receive FIFO data back to i860.
FRDENA/	FIFO read enable. Enables FIFO data transceivers to drive i860 data bus.
FWCLKENA/	FIFO write clock enable. Enables FIFO data transceivers to clock data from the i860 data bus into transceivers.
FWRENA/	FIFO write enable. Enables FIFO data transceivers to drive FIFO data bus.
i860EOD/	i860 end of data. End of data bit specifying end of message in transmit FIFOs of RX node.
RXEOD/	Receive-Transmit end of data bit. Bi-directional end of message bit between RX node and DCM.
CLK/2	20Mhz clock send from RX node to DCM to synchronize data transfer.
EODCNT/	EOD count. Signal that enables EOD counter to count up or down depending on whether an EOD bit is read or written into the RX node receive or transmit FIFOs.
EODLAST	Signal indicating that the last word read from the receive FIFOs has an EOD bit associated with it.
EODU/D#	EOD up/down. Signal specifying the direction of the EOD counter when enabled by EODCNT#. The counter counts down when an EOD is removed from the receive FIFO, and up when an EOD is written into the receive FIFO.

FIFOWRAP/	Control signal specifying the wrap operation of the DCM interface. The wrap mode removes words from the transmit FIFO and places them in the receive FIFO without using the DCM.
FS<3:0>	FIFO state machine bits.
GOTEOD	Got EOD. Output of EOD counter that indicates a non-zero count of EOD bits (messages).
NETRRDY/	Network receive ready. Signal from DCM indicating it is able to receive words from the transmit FIFO.
NETXRDY/	Network transmit ready. Signal from DCM indication it is able to transmit words to the receive FIFO.
RACK/	Receive acknowledge. Signal to DCM acknowledging a completion of transmitting a word from RX node.
XACK/	Transmit acknowledge. Signal to DCM acknowledging transmission of word from DCM to RX node.
BRXD	Buffered serial data to UART.
BSERCLK	Buffered serial data clock.
NETINT	Network interrupt. Interrupt from DCM.
SCLK	5Mhz system clock used for synchronizing UART.
SERINT	Serial interrupt. UART driven interrupt. Combination of NETINT and USMINT#.
TESTFIXT~	Test fixture. Signal distinguishing between operation in a test fixture or in a system.
TXD	Serial data output of UART from a tri-state buffer.
USMINT/	USM interrupt. Interrupt from USM.
BUSTOUT	Bus timeout. Signal from a watchdog timer that indicates a bus transaction has not completed in the allotted time.
CLRBUS/	Clear bus. Control signal that clears the BUSTOUT signal.
CLRPAR/	Clear parity. Signal, when asserted, clears the parity error status bit.
EXPINT	Expansion interrupt. Interrupt signal from expansion port.
LRFL	Latched receive FIFO full. Active when receive FIFO is full.
LRHF	Latched receive FIFO half full. Active when receive FIFO is half full or more.

LXHF	Latched transmit FIFO half full. Active when transmit FIFO is half full or more.
PARINT	Parity error interrupt. Signal set upon a parity error, used as a maskable interrupt and a status bit. Must be cleared with CLRPAR#.
REFDEF	Refresh defeat. Control bit when set disables refresh request to DRAM control.
RINT	Receive interrupt. Interrupt generated from message receiving. Consists of logical OR of receive FIFO half full and EOD in FIFO flags.
XFL	Transmit FIFO full. Asynchronous FIFO full flag from transmit FIFOs.
XINT	Transmit interrupt. Interrupt generated from message transmitting. Active when transmit FIFO is less than half full.
FTIME6	312.5khz clock to establish refresh requests to DRAM controller at 40Mhz operation.
FTIME7	156.25khz clock to establish refresh requests to DRAM controller at 30Mhz operation.
FTIMETC/	Carry output of counter to establish time interval for bus timeout.
LPDET	Latched parity detect signal from parity detection circuit. Note that this signal can only be reset by performing a DRAM read without a parity error.
RESET	Reset signal. Active for approximately 1ms after power up or termination of the USMTRESET/ signal.
Ri860INT	Logical OR of all interrupt which is eventually sent to i860 after double synchronization.
COUNT/	Carry output of clock divide counter to establish time base for RX node 52 bit counter.
RXD	Unbuffered serial receive data from USM.
SERCLK	Unbuffered serial receive clock from USM.
USMRESET/	Node reset input from USM.
A<31:3>	i860 address bus.
BE<7:0>#	i860 byte enable signals.
LADS#	i860 address strobe.

BREQ	i860 bus request signal.
HLDA	i860 hold acknowledge signal.
LOCK/	i860 bus lock signal.
NENE#	i860 next near signal.
W/R#	i860 write/read signal.
D<63:0>	i860 data bus.
CS<3:0>/	DRAM CAS signal for each memory bank.
OE<3:0>/	DRAM output enable signal for each memory bank.
RAS<3:0>/	DRAM RAS signal for each memory bank.
RASP0/	DRAM RAS signal for parity.
WE<7:0>/	DRAM write enable signal for each byte of data.
CAE/	Column address enable. Enables DRAM column address buffers to drive i860 address bits on DRAM address bus.
CAL/	Column address latch. Latches column address in column address buffers for subsequent page access of DRAMs.
CSX/	DRAM CAS signal which is buffered to each memory bank.
OEX/	DRAM output enable signal which is buffered to each memory bank.
RAE/	Row address enable. Enable DRAM row address buffers to drive i860 address bits on DRAM address lines.
S<4:0>	DRAM control state bits.
ERDE/	Expansion read data enable. DRAM control input from expansion port to enable DRAM data transceivers to drive the i860 data bus.
ERDL/	Expansion read data latch. DRAM control input from expansion port to control latching of data from the DRAM data bus.
EWDE/	Expansion write data enable. DRAM control input from expansion port to enable DRAM data transceivers to drive the DRAM data bus.
EWDL/	Expansion write data latch. DRAM control input from expansion port to control latching of data from the i860 data bus.
RTDS/	Extra expansion port input.

CLRFRQ	Refresh request acknowledge from DRAM control.
DRDL<2:1>/	Delayed read data latch, multiple copies.
DWCE/	Delayed write clock enable. Clock enable for data latch in DRAM transceivers. Enables clocking of write data from i860 data bus.
RDDOE/	Read data output enable. Enables DRAM data transceivers to drive i860 data bus.
RPL/	Read parity latch. Latches parity errors from i860 data bus.
WRDOE/	Write data output enable. Enables DRAM data transceivers to drive DRAM data bus.
DADDAA<9:0>	DRAM address bus for memory banks one and two.
DADDBA<9:0>	DRAM address bus for memory banks three and four.
DRAMDATAD<63:0>	DRAM data bus.
PADDA<0:9>	DRAM address buf for parity memory.
RPD<7:0>	Read parity data. Parity DRAM data outputs.
WPD<7:0>	Write parity data. Parity DRAM data inputs.
EPD<7:0>	Even parity data. Byte parity check bits, low when parity error exists.
DP<7:0>	Byte even parity bits latched from parity DRAMs. Used to check parity with read data on i860 bus.
OPD<7:0>	Byte parity bits generated from write data on the i860 data bus. (Mnemonic indicates odd parity, but parity bit is actually even.)
FRES/	FIFO reset.
RFRD/	Receive FIFO read.
RFWR/	Receive FIFO write.
XFRD/	Transmit FIFO read.
XFWR/	Transmit FIFO write.
REMP/	Receive FIFO empty flag.
RFULL/	Receive FIFO full flag.
RHF/	Receive FIFO half full flag.
XEMP/	Transmit FIFO empty flag.
XFL/	Transmit FIFO full flag.

XHF/	Transmit FIFO half full flag.
NETD<31:0>	RX-DCM data. Output of transmit FIFO, input of receive FIFO.
LRFL/	Latched receive FIFO full flag.
LRHF/	Latched receive FIFO half full flag.
LXEMP/	Latched transmit FIFO empty flag.
LXHF/	Latched transmit FIFO half full flag.
EODINF	EOD in FIFO status bit. Set whenever an EOD bit is in the receive FIFO.
BD<7:0>	Bi-directional data between buffer and PROM, UART, Performance port and lower byte of status port.
XPR<3:0>/	Expansion port status bits.
EODMSK/	EOD mask. Control bit which masks the EOD interrupt.
EXPMSK/	Expansion mask. Control bit which masks the expansion interrupt.
HWMSK/	Hardware mask. Control bit which masks the parity and bus timeout interrupts.
RMSK/	Receive mask. Control bit which masks the receive FIFO half full and EOD in FIFO interrupts.
SERMSK/	Serial mask. Control bit which masks the serial (UART) interrupt.
XMSK/	Transmit mask. Control bit which masks the transmit FIFO half empty interrupt.
GREENON	Control bit which drives the green LED.
REDON	Control bit which drives the red LED.
YELON	Control bit which drives the yellow LED.
NETST<7:0>	DCM status bits.
NETCTL<7:0>	DCM control bits.
PORT<3:0>	Performance port bits.
PASTB/	Performance port strobe.
SLOTID<7:0>	Slot ID bits.