Box 37
Folder 19

1 of 3

Aquarius / VAX 9000 and Medusa / B7 development, 1989-1992

Digital Equipment Corporation records
Engineers' papers: Tim Litt collection: Product development
X2675.2004, *Box 37; 38* 102737471

MEDUSA

VAX Based, Gate Level Simulation Beyond Equivalent DECSIM Behavioral Model and ZYCAD Throughput

Mark Firstenberg
Aquarius Logic Design
HPS

Presented 1/26/89

→ Medusa Building

① Run Build_INOS.com
② Run Ncomp_cpu.com (Cover.right)
③ Run Link_cpu.com
④ Run AQUARIUS-SAV.com
⑤ Run Regression
Build_QRT.com
Build_ITI4.com
Build_Vconfluct.com
⑥ Run Run_Reg_TEST.com

Checking

① SEARCH ncomp_cpu.log "-W-"
just CDC not found

→ UIT VLTD Meeting 4-2-91

-Cache model

4 KBYTE DUAL PORTED PRIMARY
128 KBYTE SECONDARY

12 MSEC

Aquarius Design Rules

Digital Equipment Corporation

RESTRICTED DISTRIBUTION

DO NOT DUPLICATE

Restricted Distribution
This information shall not be disclosed to non-Digital Personnel or generally distributed within Digital. Distribution is restricted to persons authorized and designated by the responsible Engineer/Manager. This document shall not be left unattended and when not in use shall be stored in a locked storage container.
These Restrictions are to be enforced until:
December 31, 1989
Robert B. Pratt — January 15, 1987
Responsible Engineer/Manager — date

Release 2.2

January 15, 1987

Aquarius Technology/Design Process Integrity Group

Copy No. —

Contents

1 Introduction

The intent of the Aquarius Design Rules is to ensure the use of a consistent set of assumptions and rules of use concerning the technologies, design style, tools, and other design constraints to be considered by the Aquarius Design Team. Companion documents to the Aquarius Design Rules will include the Aquarius Technology Rules Notebook and the Aquarius CAD Notebook. The intent of the Aquarius Design Notebook is to include all of the information necessary for the designers to design by. The companion documents will have more explicit information which must serve the larger group of technologists and CAD implementors. Much of the content of the Aquarius Design Rules will be drawn from the specifics in the companion documents. However, rules in the Aquarius Design Notebook may be more conservative or more restrictive than those in the companion documents. Rules in the Aquarius Design Notebook should be considered to have superiority over rules found elsewhere.

2 Resources

2.1 MCA III

The MCA III (macro cell array) is a high performance ECL array which provides a raw functionality of between 8000 and 10000 equivalent 2 input gates. Since the technology implements current mode logic with three level series gated structures, gate equivalency is not a very meaningful measure of its capabilities. In more specific terms the functionality is roughly eight times that available in the MCA I which was used to implement the VAX 8600.

The physical resources are the following:

414 Major Cells (Mcells)

200 Output Cell (Ocells)

224 Input Cells (Icells) (also known as interface cellls)

256 Pad Cells (Pcells) (also known as I/O pins)

Given our design approach and other constraints however, not all of these resources are directly available to the logic designer. Certain resources will be reserved for implementing the clock distribution system, scan support, AC test, and die temperature monitoring.

Basically these requirements will require approximately:

38 Mcells (Clock - 34, SCAN - 2, AC test - 2)

16 Icells (Clock - 6, SCAN - 5, AC test - 5)

3 Ocells (SCAN - 1, AC test - 2)

15 Pcells or I/O pins (Clock - 4, SCAN - 6, temp monitor - 2, AC test - 3)

Although there are 200 output *cells* available that does not mean that a design may implement 200 output signals. Due to signal integrity constraints the actual number of output signals will be

constrained to 128 unique signals. 140 output signals may be implemented *if* the additional 12 signals are complementary signals of 12 of the original 128 signals. However, more than 128 output cells may be used to implement those 128 output signals as some some output macros require more than one cell. Output cells used for SCAN or for the AC test circuitry will not be included in the 128 output cell limit.

Originally, it was thought that the structural design would progress through 2 or 3 design releases. Since that time, we have structured the design process differently, and there should be only two major structural design releases. The first release where the design is functionally correct but is not quite making the 16 nsec. cycle. The second release is then, the final release where the design must be correct and meet the timing requirements.

Resources currently physically realizable but not necessarily available to the logic designer:

414 Mcells

224 Icells

200 Ocells

256 Pcells

128 output signals

Resources available after required overhead is considered:

376 Mcells

208 Icells

197 Ocells

241 Pcells

128 output signals

For the first release of structural designs which must meet an 18 nsec. cycle time but not necessarily the 16 nsec. cycle time, about 5 percent of the MCA's resources should be held back. This

will enable some changes to be made either in functionality or implementation for performance, without a major impact on partitioning. The following limits on resource usage should therefore be observed:

357 Mcells

187 Ocells

198 Icells

229 Pcells

125 outputs

If the overhead resources (clock, scan, etc.) are also included as the SID output report will include them, then the following limits should be observed:

395 Mcells

190 Ocells

214 Icells

244 Pcells

125 outputs

Other restrictions such as power or routability restrictions may take precedence over the above restrictions for particular designs.

2.2 STRAMs

Self Timed Random Access Memories (STRAMs) will be the major RAM component used in the AQUARIUS design. No standard type RAM components will be available for use in the Hi-Tech portion of the machine due to the requirement for STECL outputs. These STRAM components include a normal RAM structure including the storage array, decode logic, write logic, sense amps, etc. to which have been added input latches for data and address, output data latches, a write clock circuit, and series terminated emitter coupled logic (STECL) outputs.

Two sizes of the STRAMs are to be available. A 1K x 4 at 6 nsec and a 4K x 4 at 10.5 nsec. access time from address.

2.3 Custom Chips

In addition to the Clock Distribution Chip there is currently an opportunity to do three additional custom chips. The following is a general statement of the resource restrictions to be kept in mind when specifying a custom part.

The maximum silicon area available for a custom chip is the same as that for the MCA III which is about 380 mils per side. This restriction is due to the packaging constraints around putting a chip on an MCU and using the same TAB design. In other words, if the chip is to go onto an MCU then the silicon will have to be the same size as the MCA III in order to support the required number of pins. Even if the active *used* area of the chip requires less silicon, the chip must be 380 mils square. *IF* the chip is not to be packaged onto an MCU then there is an opportunity to implement the design with a smaller die.

The power dissipation for a part of the size of an MCA III should be restricted to about 25 watts. This will prevent the necessity of redesigning the part for use in Aridus. This level of power dissipation is only available to devices packaged on MCU's. This is probably a difficult constraint to meet as there is a major opportunity in custom design to substantially increase the device density and to therefore increase the power density.

For custom chips going onto an MCU there will also be the contraint that it use the same TAB design as the MCA III. This means that there is a maximum of 256 signal I/O available (104 for power and ground) and that the power and signal pins need to be in the same location as the same pads on an MCA III. A custom part may use fewer pins however, the pads will have to be there for all 360 pads. A custom part which uses A and/or B phase clocks shall have the same clock and scan pinning as an MCA III and will be required to implement the clock generation and distribution with the same timing characteristics as the MCA III.

For any custom parts not designed to go onto an MCU, then there is a major question as to a package for that part and what the

power and pinning constraints will be. Some reasonable packages for consideration may be the MCA I and MCA II packages. The MCA I package will support a 4-5 watt device about 250 mils square with 68 pins (8 power, 60 signal I/O). The MCA II package will support about an 8 watt device 270 mils square with 149 pins (29 power, 120 signal I/O). Other packages will probably represent an incremental resource to design and develop such a part and its risk to the projects schedule must be taken into consideration.

2.4 Clock

The clock system interfaces with the the rest of the machine at the Clock Distribution Chip (CDC) which supplies the clocks to which the designer has some limited access. There is a CDC on each and every MCU which supplies the clocking signals for all chips on that particular MCU. This CDC supplies a 500 Mhz (typ.) clock signal and a gated reference signal to each of the MCA III or equivalent custom chips on the MCU. The MCA III or equivalent then generates the **A** and **B** phase clocks and distributes these clocks throughout the chip. The CDC provides nine copies of the clock signals. Eight are used for MCA III chips or equivalents and one for test.

The CDC also supplies clocks for the STRAM chips. Six copies each of four groups are available for STRAM clocking and for the custom general register file part. Each of the four groups of STRAM clocks may be independently positioned in one of eight possible phase positions relative to the **A** phase clock signal. Each group then has six copies and each copy may drive the clock inputs of two adjacent STRAM parts or one custom part.

In general the CDC connections are predetermined for the MCA III and equivalent parts. The connections for the STRAMs will be mostly predetermined with only the phase selection being a designer option.

2.5 MCU

A multiple chip unit (MCU) provides a high density interconnect system, a power distribution sytem, and a heat removal system for a number of closely associated chips. The MCU is designed to provide adequate resources in the above systems to support the anticipated configurations.

Each MCU will have one clock distribution chip located in the center of the MCU. This chip will provide clock signals to all of the other chips on the MCU and also provide some common support circuitry for SCAN testing. The maximum number of additional chips, (excepting STRAMs) either custom or MCA III's is eight. It is expected that MCA III's and custom parts will have the same form factor and use the same TAB layout. It is also expected that the location of the chips will be to one of the predetermined locations on the MCU. This may be modified for those MCU's having a number of STRAM chips on them.

Since STRAM chips are significantly smaller than MCA III's or custom parts, several STRAM parts may replace an MCA III. A cluster of nine STRAMs may replace an MCA. The exception to this is when a cluster of STRAMs replaces an MCA which is not in a corner position *and* there is a cluster of STRAMs in an adjacent corner. If the STRAMs are 4K x 4 parts then only 6 STRAMs are allowed in the non corner cluster. The following are the maximum allowable configurations.

number of MCA III's	number of 1k x 4's	number of 4K x 4's
8	0	0
7	9	9
6	18	18 (15)
5	27	27 (24)
4	36	36 (30)
3	45	42
2	48**	48
1	48**	48**

(*) indicate the number of STRAMs if they are adjacent clusters.

** Although physical limitations would allow more STRAMs, the number of STRAM clocks available from the CDC will allow only 48 STRAMs per MCU on the basis of 2 STRAM loads per clock signal. Additional loading will probably not be allowed as this directly affects clock skew.

The MCU is designed to provide a cooling capacity of about 300 watts total or about 33 watts per MCA III or equivalent. Since one MCA is replaced at most by nine STRAMs there is enough cooling capacity to handle STRAMs of more than 3 watts, well above the maximum specified power dissipation for STRAMs.

The power distribution values are not yet available, but are to be expected to be in line with the possible requirements and with the available cooling capacity.

There are currently available 804 I/O pins on the MCU for signals to enter or leave the MCU. Some small but not yet determined number will be reserved for testing and for SCAN distribution (19). Clock distribution may yet require a few (7) of those pins also.

MCA III and custom chips will be mounted on every MCU with a fixed and predetermined rotation. The designer will not

be allowed to assume that a chip may be rotated to afford a more favorable routing. Accomodations for mounting STRAMs in more than one orientation may be made although not more than two orientations per MCU will be allowed. Currently, however, we are limiting the STRAM chips to one orientation per MCU.

2.6 Modules (Hi-Tech)

A Hi-Tech module will be a multi layer controlled impedance module which will hold the interfaces between MCU's and between MCU's and the rest of the system. A Hi-tech module will consist only of water cooled MCU's and passive components. No active components may be directly mounted to the module and the heat dissipation of the passive components is severely restricted.

There are currently two Hi-tech modules envisioned for the Aquarius system. The first and largest is the CPU module which will carry up to 16 MCU's and connectors to the console, clock, and SCU module. The second is the SCU module which will hold six MCU's and connectors to I/O, memory, four CPU modules, console, and the clock.

The CPU module will be organised as a four by four array of MCU's with the connectors along one edge of the module. The module will be approximately 24 inches by 24 inches in size and will contain up to five orthogonal signal layer pairs. The SCU module will be 18 inches by 24 inches in size.

The module is not expected to participate directly in the power distribution on the module but only to provide signal interconnectivity. A separate power distribution assembly will provide D.C. power to the MCUs.

The MCU's are mounted on the module with a fixed and predetermined rotation. If a MCU needs to be rotated, then it must be layed out as a unique design.

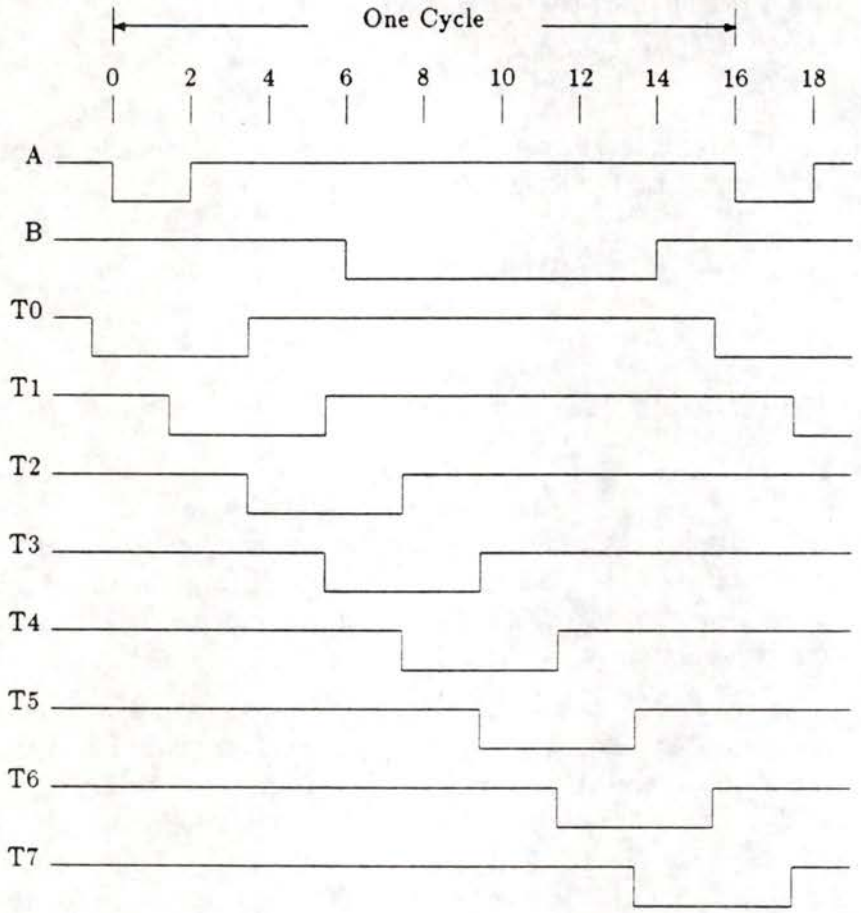
3 Clock Rules

3.1 Clock description

The Aquarius clock system is comprised of a master clock source, a passive distribution network, CDC chips which buffer and derive certain clock signals, and clock derivation logic and a distribution network on the MCA III chips. The clock network is considered and designed from the master source down to the latch or flip flop input as an integrated entity. The end result of this network is a clock system which provides two major non-overlapping clock phases for primary use in the logic design and eight additional clock phases for use with STRAMs.

The two major clock phases consists of a narrow clock phase (1/8 of a cycle) and a wide clock phase (4/8 of a cycle). The clock phases are called A phase and B phase respectively. Nominally, latches which are clocked on the A phase are considered to be state elements in the design whereas, the latches which are clocked by the B phase are inserted in order to prevent race conditions or minimum delay requirements. The STRAM clocks are nominally 2/8 of a cycle in width and occur on 1/8 cycle phases. The STRAM clocks are phase shifted slightly such that the edges occur nominally 0.5 nsec. prior to A or B phase edges or equivalent positions. This is a compromise due to the relatively long setup and hold requiremnts of the STRAM devices.

The following diagram shows the clock relationships. The STRAM clocks (T0 - T7) are offset from the system clocks by 0.5 nsecs.



The low level of the signal corresponds to the assertion of the clock, whereas the high level of the signal corresponds to its unassertion.

3.2 Clock skews

The clock system has been designed to provide low skew relationships between the clock in the system, allowing a relatively simple set of design rules be applied to the logic design.

The worst case skew between any two latches clocked by either A phase or B phase is 1.5 nsec. This is true for latches between modules such as the CPU module and the SCU module.

The worst case skew between an A or B latch and the STRAM clock on the same MCU is 1.0 nsec. or less.

The worst case skew between any STRAM clocked in the system and any A or B phase latch clock is 1.5 nsec.

3.3 Latch resources

T. B. D.

3.4 Design usage rules

The use of an A clock or B clock in the design is not explicicty determined. Rather the design system implicictly determines the clock phase based on the choice of RTL macro. This automatically prevents incorrect assignment of clocks obviating the need for complex rules to check for correct asssignment.

A cycle of clocks starts with an A phase and a T0 clock and sequences through T7 and the B phase clock. Nominally, the starting edge of the A phase clock defines the beginning of the cycle and the next starting edge of the A phase clock will define the end of the cycle. In actuality, due to the minus 0.5 nsec offset of the STRAM clocks, the assertion of T0 will occur prior to the assertion of the A phase clock. Also, the assertion of T7 will overlap into the time of the next cycle. However, a cycles worth of clocks includes the assertion and subsequent unassertion of A phase, B phase and T0 through T7 clocks. Stopping the clock will conclude with the unassertion of all clocks.

The Aquarius design center for the cycle time is 16 nsec.

In designing sequential logic using the A and B latches there are three timing constraints which must be met.

1. The logic delay (gates plus interconnect) between an A phase latch and a B phase latch must be less than or equal to 7/8 of the

cycle time minus clock skew minus setup time for the B latch minus the clock to data and output delay of the A latch.

```
cycle = 16 nsec.
skew = 1.5 nsec.
tsetup(M946) = 400 psec.
tpd(M945) = 400 psec.
delay for 3 loads and 160 mil of metal equals
          240 psec. (hi power output)

7/8 * 16 - 1.5 - .4 - .4 -.24 = 11.46 nsec
           available for logic.
```

2. The logic delay (gates plus interconnect) between an B phase latch and a A phase latch must be less than or equal to 6/8 of the cycle time minus clock skew minus setup time for the A latch minus the clock to data and output delay of the B latch.

```
cycle = 16 nsec.
skew = 1.5 nsec.
tsetup(M945) = 400 psec.  (A latch = M945)
tpd(M946) = 400 psec.     (B latch = M946)
delay for 3 load and 160 mil of metal equals
          240 psec. (hi power output)

6/8 * 16 - 1.5 - .4 - .4 -.24 = 9.46 nsec
          available for logic.
```

3. In addition there is also a combined requirement that the logic delay between A and B added to the logic delay between B and A must be less than or equal to 9/8 times the cycle time minus clock skew minus the clock to data and output delay of the A latch minus the data to data and output delay of the B latch minus the setup time of the final A latch.

```
cycle = 16 nsec.
skew = 1.5 nsec.
tpd(M945) = 400 psec.
load delay = 240 psec.
tpd(M946) = 400 psec.
load delay = 240 psec.
tsetup(M945) = 400 psec.

9/8 * 16 - 1.5 - .4 - .24 - .4 - .24 - .4 = 14.82
```

Given the above typical calculations, one can see that there is some latitude in the logical positioning of the B latch. For example in a path where the full time from the A latch to the next A latch is taken and given the above values as an example,the delay window for positioning the B latch is about 6 nsec.

```
Logic delay #1 = A to B delay
Logic delay #2 = B to A delay


Max (logic delay #1 + logic delay #2) = 14.82
```

If Logic delay #1 is at max (11.46 nsec) then logic delay #2 must be less then 3.36 nsec. (14.82 - 11.46).

If Logic delay #2 is at max (9.46 nsec.) then logic delay #1 must be less then 5.36 nsec. (14.82 - 9.46).

The positioning window for the B latch is 6.10 nsec wide (9.46 -3.36) in this example. Actual values will be slightly different as more accurate numbers are obtained for the latches, latch load conditions vary, and for the particular latches used.

Given the approximation that a loaded macro delay is 750 psec., the implication is that there may be up to 15 macro levels between the A latch and the B latch, or up to 12 macro levels between the

B latch and the A latch as long as there are no more than 20 macro levels between the A latch and the next A latch. The B latch has a positioning window of about 7 macro levels.

4 MCA III

4.1 Macro library

This is the current list of DEC supported macros. This does not mean that all of these macros are supported by SID nor are all necessarily available to the designer as Clegos. The designations are not firm and are subject to change without prior notification. Size estimates have not been confirmed by layout design but are expected to be relatively accurate. Most macros will be available in both a high power and a low power version. The designations for a high power or low power macro will have an H or L prefix respectively. In this list an H,Lxxx indicates that the macro is available in both power levels.

MCA III

Macro	Size	Description
H,L200	.25	5 input OR/NOR
H,L201	.25	4 input OR/NOR
H,L202	.25	2 input OR/NOR
H,L203	.5	8 input OR/NOR
H,L207	.25	6 input OR/NOR
H,L211	.25	2-2 OR/AND
H,L212	.5	3-2-2-2 OR/AND
H,L214	1.0	2-2-2-2-1-1-1-1 OR/AND
H,L215	1.0	2-2-3-3-3 OR/AND
H,L219	.25	3-3 OR/AND
H,L221	.25	2-2 OR/EXOR
H,L223	.5	4 input EXNOR
H,L224	.5	4 input EXOR
H,L225	.5	2-1-1-2 OR/AND/EXOR

H,L226	.5	2-1-1-2 OR/AND/EXNOR
H,L228	.25	2-1 AND/EXOR
H,L252	1.0	Quad 2 to 1 MUX
H,L253	.25	2 to 1 MUX w/ENABLE(low)
H,L254	.25	2 to 1 MUX w/ gated inputs
H,L255	.5	Dual 2 to 1 MUX
H,L256	.25	2 to 1 MUX
H,L263	1.0	1 of 4 DECODE (hi)
H,L281	1.0	FULL ADDER
H,L282	.5	FULL ADDER w/ gated inputs
H,L283	1.0	2 Bit LOOKAHEAD CARRY
H,L284	.25	HALF ADDER w/ gated inputs
H,L286	.5	3 Bit ADDER CARRY
H,L313	.25	2-2 OR/AND
H,L315	.5	2-2-1-1 OR/AND
H,L331	.5	3-2-2 AND/OR
H,L332	.5	Gated OR
H,L333	.5	Gated OR
H,L370	.25	Differential BUFFER
H,L371	.25	2 to 1 MUX w/ Differential Inputs
H,L372	.5	D FLIP-FLOP w/ Diff' clock and data
H,L373	.25	2 to 1 MUX w/ Diff' inputs and control
H,L374	.25	Differential BUFFER
H,L404	.5	12 input OR/NOR
H,L413	.5	4-3-3-3 OR/AND
H,L416	.75	4-2-3-2-3 OR/AND
H,L417	.5	5-4-3-2 OR/AND
H,L418	.75	5-4-3-2-1 OR/AND

H,L422	.5	Dual 2-2 OR/AND/EXOR
H,L427	.5	2-1 EXOR/AND/NAND
H,L451	.5	4 to 1 MUX w/ENABLE(low)
H,L453	.25	2 to 1 MUX w/ENABLE(low)
H,L458	.5	4 to 1 MUX w/ENABLE(hi)
H,L461	.5	1 of 4 DECODER w/ENABLE(low)
H,L462	.5	1 of 4 DECODER w/ENABLE(hi)
H,L464	1	8 to 3 PRIORITY ENCODER
H,L485	.5	3 Bit ADDER
H,L510	.5	4-4-4-4 OR/AND
H,L511	.5	3-3-3-3 AND/OR
H,L512	.5	3-3-3 AND/OR
H,L518	.25	3-3 AND/OR
H,L519	.5	3-3-2-1 AND/OR
H,L520	.5	2-3-4-4 AND/OR w/ENABLE(hi)
H,L523	1.0	3-2-2-2-2-3 AND/OR
H,L618	.5	5-4-3-2-1 OR/AND
H,L658	.5	4 to 1 MUX
H,L685	.5	MFADDER
H,L802	.25	3 input EXOR/EXNOR
H,L803	.25	3 input EXNOR/EXOR
L804	.25	Dual 2 input AND
L805	.25	Dual 2 input NAND
L806	.25	2 input AND, 2 input NAND
L807	.25	2 input NAND, 2 input AND
H,L809	1.0	8 to 1 MUX w/ENABLE(hi)
L811	.25	Dual 4 input OR
L812	.25	4 input OR, 4 input NOR
L813	.25	4 input NOR, 4 input OR

L814	.25	Dual 4 input NOR
L815	.25	Dual 2 input OR
L816	.25	2 input OR, 2 input NOR
L817	.25	2 input NOR, 2 input OR
L818	.25	Dual 2 input NOR
H,L819	.25	3-2-1 OR/AND
H,L941	.5	AND gated FLIP-FLOP
H,L942	.5	FLIP-FLOP
H,L943	.5	OR gated FLIP-FLOP
H944	.25	CLOCK BUFFER
H,L945	.5	SCAN LATCH
H,L946	.25	LATCH
H,L947	.5	Dual LATCH w/ common clock
H,L948	.5	4 input OR - LATCH
H,L950	.5	2 to 1 MUX - LATCH
H,L951	.5	2 input EXOR - LATCH
H,L952	.5	2-2 OR/AND - LATCH
H,L954	.5	Dual LATCH w/ common clock

Input Cells

H,LI00	1	2 input OR/NOR
H,LI01	1	2 input OR
H,LI02	1	2 input NOR
H,LI03	1	Differential Buffer
H,LI04	1	Differential Buffer
H,LI05	1	Differential Buffer
H,LI06	1	Buffer/Inverter
H,LI07	1	Buffer
H,LI08	1	Inverter

Output Cells

H,LX01	1	2 input OR/NOR
H,LX02	1	4 input OR/NOR
H,LX03	1	2-2 gated OR
H,LX04	1	2-2 OR
H,LX11	1	2-2 OR/AND
H,LX21	1	2-2 OR/EXOR
H,LX45	2	SCAN LATCH
H,LX46	1	LATCH
H,LX47	2	Dual LATCH
H,LX51	1	2 to 1 MUX
H,LX52	2	Dual 2 to 1 MUX
H,LX53	1	2 to 1 MUX w/ENABLE(low)
H,LX58	2	4 to 1 MUX
H,LX71	2	4 to 1 Diff. MUX

4.2 Latch constraints

Latches are in general divided into two groups, defined by the particular clock to which they are attached. Since the clocking scheme has been defined as a two phase non-overlapping clock, with a relatively narrow clock enable signal and a relatively wide enable signal. The narrow clock has been called the **A** phase and the wide clock has been called the **B** phase. Latches which connect to the **A** phase clock are known as **A** latches or sometimes **TA** latches. **B** latches or **TB** latches are those which are connected to the **B** phase clock.

A latches are in general to be thought of as the **logical** storage elements in the design and **B** latches are added where appropriate to prevent race conditions. **A** latches are to be scannable latches.

B latches come in two basic flavors. The first type is used in conjunction with each **A** latch to generate a scannable latch. This **B** latch can also be used to provide the necessary feedback path, if an **A** latch with a **hold** function is desired. The second type of **B** latch which has also been termed a **mission B** latch, is used to prevent race conditions between **A** latches through the mission logic. Finding appropriate places to locate **mission B** latches involves balancing the requirements for performance and the desire to implement the fewest necessary **B** latches.

A **scannable** latch is an **A** latch and a **B** latch paired together and interconnected such that one output of the **A** latch is connected to the input of the **B** latch. If the **A** latch is also to have a hold function then an output of the **B** latch is connected back to one of the inputs of the **A** latch. The logical front end of the **A** latch implements a 2 to 1 multiplexor. The hold function is identical with the multiplexor select input. The **B** latch of the scannable latch pair presents a load to the **A** latch output and will have to be taken into consideration in timing analysis. It is believed however that the **B** latch should be placed very close to the **A** latch and therefore the additional delay caused by the load will be minimal. An output of the **B** latch of the scannable pair will be considered to be the scan data output and will eventually be connected to the SCAN IN input of some **A** latch other than the one paired with the source. Since this connection will be made late in the physical design process to minimize its effect on routing, the length of the interconnection will be somewhat unpredictable, having an unpredictable effect on the interconnect delay. Therefore the strategy will be to dedicate a **B** latch output for **SCAN DATA OUT** and for the hold feedback connection if needed.

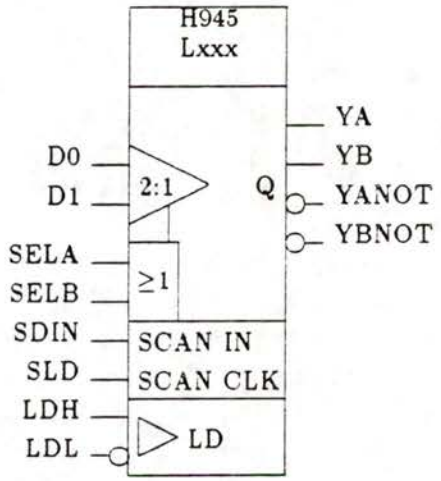
B latches are available as a dual **B** latch macro in a half cell (H,L947).One non inverting output from each latch output will be dedicated for scan data out and hold feedback, which will leave one non inverting output and two inverting outputs per latch available for driving logic.

Mission **B** latches will be available as half cell functions with two high power outputs available, and with a variety of logical

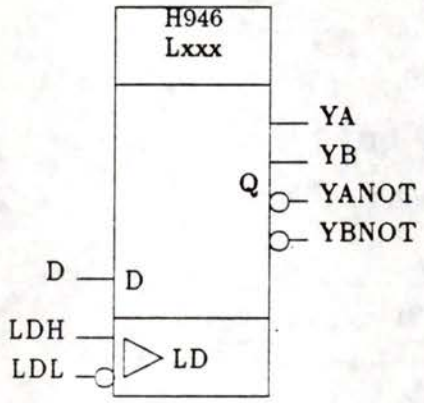
frontends. It will not be necessary for the logic designer to select which **B** latch option is appropriate, as SID is expected to make the appropriate choice.

A latch output cells will be available. However, for the connection to an internal **B** latch, an interface cell is required.

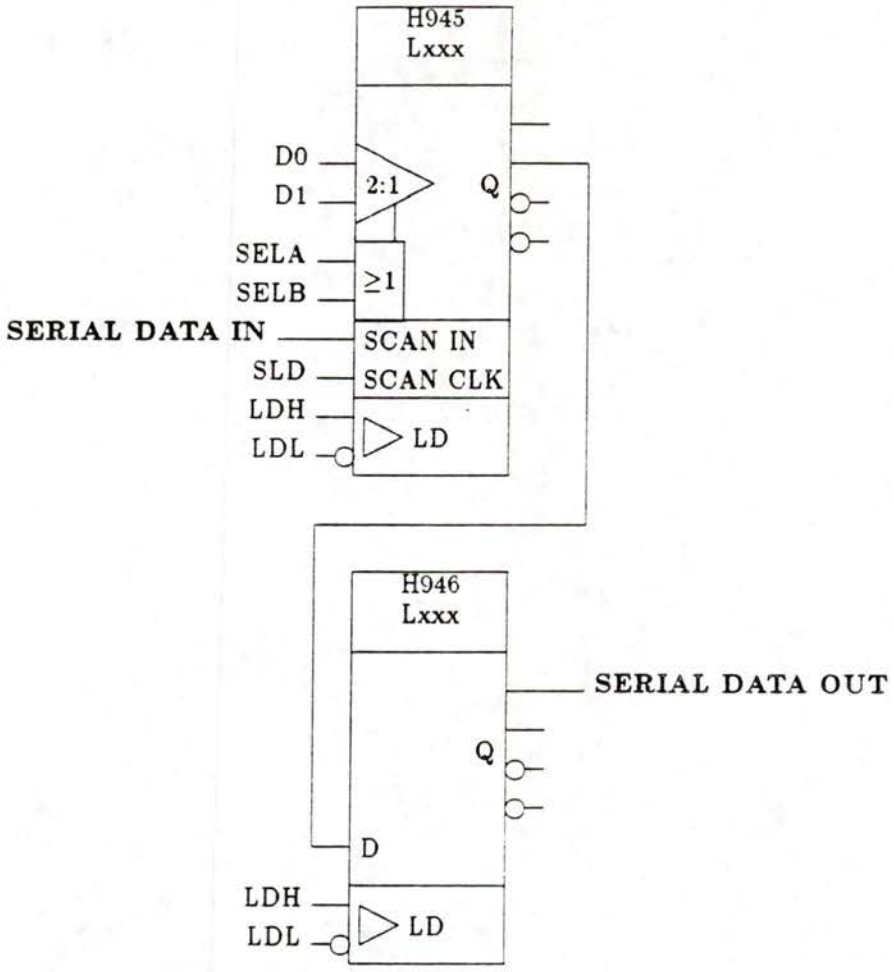
B latch output cells will also be available. A scan latch may be created using both an output cell A latch and and an output cell B latch. However, an interface cell will be required between the A latch and the B latch and an additional interface cell between the B latch and the A latch for scan requirements. If the scan latch is also a holding latch, then the interface cell on the B latch can also be used to feed back to the A latch.



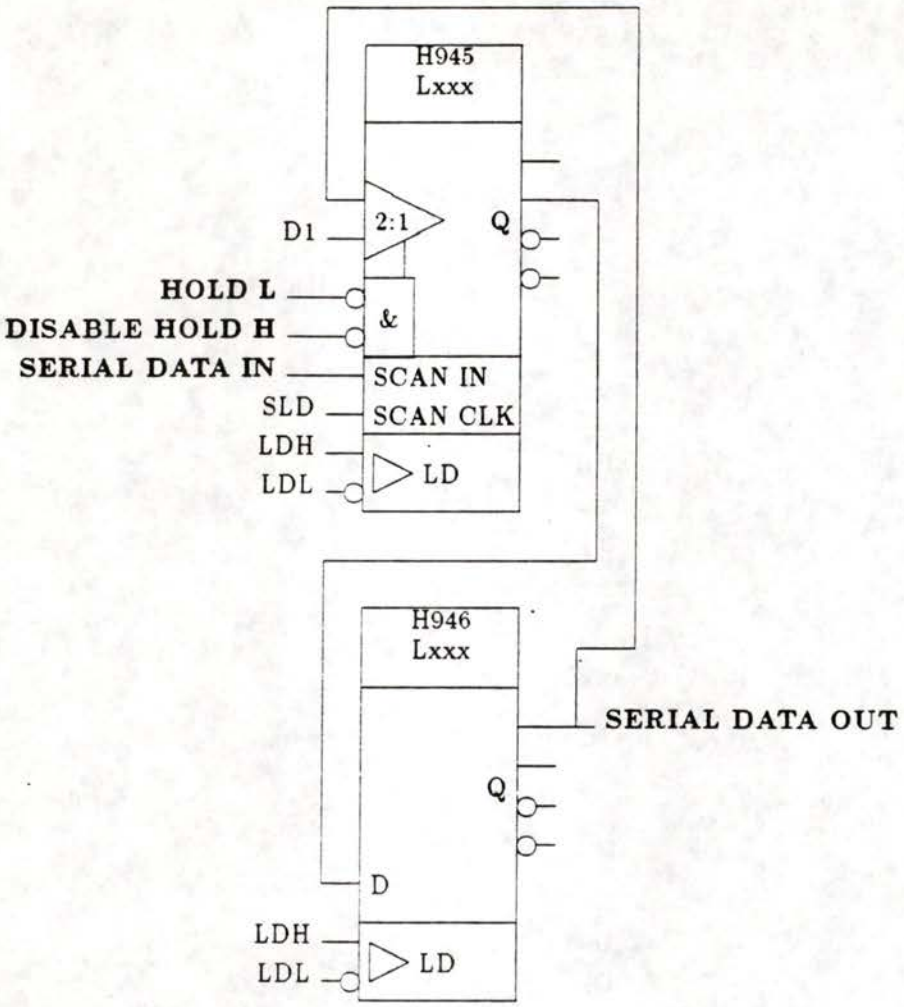
An **A** latch macro.



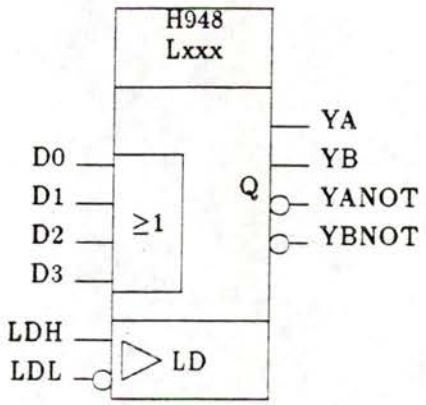
A **B** latch macro.



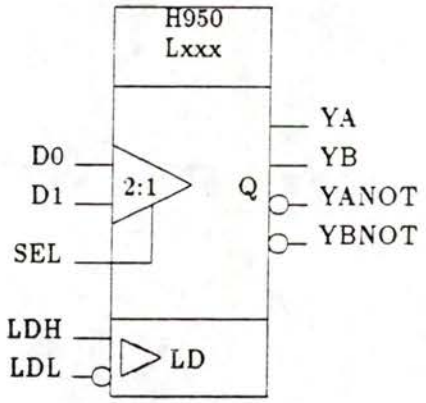
A latch and **B** latch connected as a scan latch.



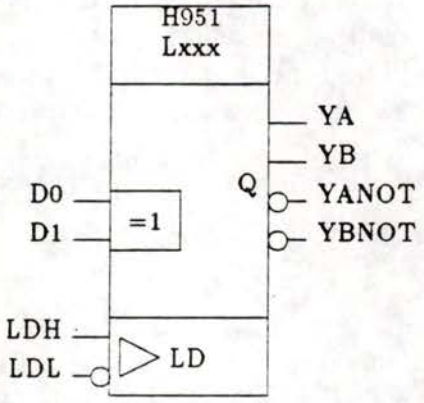
A latch and **B** latch connected as a holding scan latch.



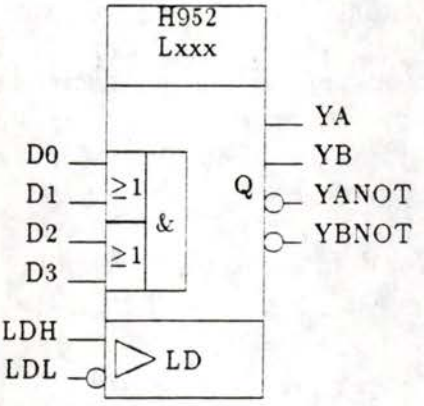
B latch with **OR** input.



B latch with 2 to 1 Multiplexor input.



B latch with XOR input.



B latch with OR/AND input.

4.3 Interface / Input cells

Interface or input cells (either name may be used interchangeably) will be required on all external signals which are brought into the array. The functionality of an interface cell is somewhat limited, since only a non series gated structure is supported.

The basic functionality of an interface cell is that of a 2 input OR/NOR.

Input signals to an interface cell MUST be external logic /it levels. Therefore an interface cell may be driven either from an external source of the proper characteristics or from the compatible levels of an internal OUTPUT cell. An interface cell input MAY NOT be driven from either another interface cell nor from an internal Mcell.

Input pads which are connected to an interface cell will be constrained to be connected with less than 50 mils of metal to its associated interface cell(s). This will be necessary in order to limit the input capacitance to some defined value.

Up to four input cells may be connected to one input pin. There will be a slight performance penalty over one cell connected to one pin but it should be less than the possible advantage obtained in saving a level of buffering in a heavily loaded tree.

Input pads for differential signals should be located next to each other.

4.4 Output cell and constraints

Output cells have two types of outputs. The first is a CML level which is connected to an output pad driver. The second type is an ECL level which may be used to drive an interface cell and then back into the internal array. The ECL output may not be used externally, to drive an internal cell, nor to drive an output cell. The CML output must only be used to drive a pad driver.

Most output cells have differential outputs and either or both outputs may be used. Also both the CML outputs and the ECL outputs may be used simultaneously.

Output cells have two switch current levels. The higher power level will allow the CML outputs to drive two pad drivers on each output if additional copies of an output are desired. In general however, it is more advantageous if the the complement output is utilized first before utilizing two pad drivers per output as it reduces the noise felt by the chip with simultaneously switched outputs. Output delays will include the STECL pad driver.

The CML output pins on the output macros are indicated by a "X" prefix and the ECL outputs are indicated by a "Z" prefix.

4.5 Pad cells and constraints

Pad cells may be configured as either inputs or as outputs. Some small number (to be determined) of pad cells may be constrained to being input cells only.

As an input the pad cell can be configured with a high resistance pulldown (75K ohms) to VEE2. This provides a logic low on an unused input and some level of electric static discharge (ESD) protection.

As an output, the pad cell can be configured in one of five different outputs. In general however, we will be using only the 11 ma. STECL output.

1. Series terminated 11 ma. (STECL) output. Series resistance will be optimised for a 50 ohm impedance line and with an 11 ma. active current source pulldown.

2. Series terminated with 6 ma. output.

3. Standard 22 ma 50 ohm emitter coupled output.

4. Standard 44 ma. 25 ohm emitter coupled output. Requires two pad drivers and two output cells.

5.Standard 22 ma. 60 ohm ECL output.

For signal integrity reasons the number of STECL signal outputs will be restricted to 128 unique signals. These are fully available to the designer as overhead outputs are already taken into consideration. The number of output signals may be extended to 140 *if* the additional signals are complementary copies of outputs already implemented in the original 128 output signals. Use of complementary signals allows partial cancellation of noise due to switching currents and crosstalk. The complementary output pins should be on adjacent pads which must be connected to the same Vcco pin, and must be driven from the same output macro. There will also be a requirement that output pins be interspersed or alternated with input pins or unused pins in some yet to be determined pattern. No more than 32 non-complementary outputs per side will be allowed and no more than 35 outputs per side including complementary outputs.

4.6 Interconnect constraints

Although it is not a **rule** it is recommended that the loading of nets be restrained to 4 or 5 for performance reasons. It is worthwhile to note however that the performance degradation is more the result of the metal interconnecting the loads rather than the load itself. This implies that loads which are closely associated with each other (such as being on the same macro) on the same net may be considered to be connected with a minimal amount of metal (8 mils).

The rules for wire "OR"ing have not yet been worked out. They may turn out to be too restrictive to yield useful results.

All external input signals must interface to the MCA III chip through an interface cell. External signals may not be inputs to internal cells (Mcells) or output cells. External signals must be 100K ECL compatible.

Interface cells may drive internal cells or output cells. They may not drive other interface cells or drive a pad driver or drive directly off chip.

Internal cells may receive inputs from either interface cells or other internal cells. They can not receive inputs from an external source or from an output cell.

Internal cells may drive other internal cells or output cells. They can not drive interface cells, pad drivers, or off chip.

Output cells may receive their inputs from either interface cells or from internal cells. They can not be driven from external sources or from other output cells.

Output cells have two sets of outputs. One set can only drive a pad driver which in turn drives off chip. The other set of outputs can only be used to drive an interface cell. An output cell can not directly be used to drive an internal Mcell nor another output cell.

External cells can only be driven from a pad driver which in turn can only be driven by an output cell. Outputs will be 100K ECL compatible with a nominal 50 ohm series termination.

	Loads			
Sources	External	Mcell	Icell	Ocell
External	-	No	Yes	No
Icell	No	Yes	No	Yes
Mcell	No	Yes	No	Yes
Ocell (Z)	No	No	Yes	No
Ocell (X)	Yes	No	No	No

Table of allowed connections.

4.7 Delay calculations

The following are the preliminary timing rules to use in considering system timing constraints.

4.7.1 Macro delays

A simple rule will be to use 750 ps. per macro level inclusive of the switch delay and fanout and metal delay. For critical paths one may assume 650 ps. per macro level. For more detail see below.

A more detailed estimate may be made by using the delay tables for the macros in the appendices. These are the macro switch times only. The metal delay must be calculated separately (see below) and added to the macro delay.

Output cells should be considered to be 1.5 nsec inclusive of metal delays. For critical paths where more simple output cells may be used, 1.0 nsec may be considered for the delay. Again, for a more detailed estimate of delays, use the dealy values in the appendices. The output pad delays are included in the values.

Input interface cell should be considered to have a 500 ps. delay inclusive of metal.

4.7.2 Metal delays

Metal delay would be 120 ps for the first load and 60 ps. per additional load for a high power gate, and 240 ps. for the first load and 120 ps per additional load for the low power gate. This is based on an assumption of an average of 80 mils of metal for the first load and about 40 mils per additional load. Actual lengths will vary due to actual placement and routing. If you have reason to believe that your interconnect is substantially longer then you should add a delay of about 3.2 ps per mil for the low power case and 1.6 ps. per mil for the high power case. For instance, if you knew that your signal came onto the chip on one side, thru an input cell, across the chip and thru an output cell on the opposite side, then the metal length would be on the order of 450 mils minimum if the pins were opposite each other. If they were diagonally spaced then the length could be as bad as 900 mils (assume manhattan distance plus 20% for meander).

4.8 Clock useage

A clock generation and distribution circuit will be predesigned for the MCA III and will be the source for A and B clocks for latches in the Aquarius design. No other source of A and B clocks will be allowed on MCA III designs for Aquarius.

The A and B clock distribution includes distribution of either an A or a B clock to each and every Mcell or Ocell location on the chip. In the internal array the A or B clock will be distributed on alternating rows of half cells.. Placing a latch macro on an appropriate row will determine whether it becomes an A latch or a B latch. The placement tool is expected to provide the capability of selecting the appropriate row to assign a latch to, depending upon its useage. The designer is not expected to have direct access to either A or B clocks, but will determine which of the two clocks he would like a latch connected to based upon which RTL body he selects. In other words there is a B latch body and an A latch body which may functionally be equivalent but the difference is in which clock they are connected to.

4.9 Testability constraints

T.B.D.

4.10 Power

Two voltages besides VCC (ground) are required for the MCA III.

VEE1 is -5.2V with a tolerance of +/- 8% at the chip.

VEE2 is -3.4 V with a tolerance of +/- 8% at the chip.

In general the VEE1 supply is primarily used for the series gated current switches which allows three levels of series gating. The VEE1 supply is also used for levelshifting signals for the lower levels of series gating. VEE2 is used to provide the pulldown current for both internal and external ECL signals. This reduces the power dissipation in the chip considerably.

The total power dissipation for an MCA III must be less than a maximum of 30 watts worst case. At the current time there is no minimum power dissipation requirement.

Detailed rules for calculating the power dissipation may be found in the MCA III design manual in the appendix. The DECALC program called NUPOWER is in the AQUA directory INDIAN:[DECALC] and may also be used to calculate the power dissipation. Also, SID does generate accurate power dissipation numbers as part of its output report.

4.11 Reserved Resources

The following MCA III pins have dedicated uses and are not available to the designer for use.

Pin	Signal Name
1	Thermal Diode - Anode
2	Thermal Diode - Cathode
221	SCAN_A_CLK_H
222	SCAN_LOAD_H
229	SCAN_SELECT_H
230	SCAN_B_CLK_H
250	RING_OSC_ENABLE_H
256	TEST_DATA_IN_H
258	SCAN_DATA_OUT_H
268	AC_TEST_OUT_L
269	AC_TEST_OUT_H
309	CLOCK_H
310	CLOCK_L
321	GATED_REF_CLOCK_H
322	GATED_REF_CLOCK_L

The following pins are the preassigned power and ground pins.

VCC	Top	Right Side	Bottom	Left Side
	9,10	95, 96	189,190	275,276
	25,26	103,104	205,206	283,284
	39,40	115,116	219,220	295,296
	51,52	121,122	231,232	301,302
	65,66	131,132	245,246	311,312
	81,82	139,140	261,262	319,320
		149,150		329,330
		155,156		335,336
		167,168		347,348
		175,176		355,356

VEE1	Top	Right Side	Bottom	Left Side
	47,48	109,110	227,228	289,290
		127,128		307,308
		143,144		323,324
		161,162		341,342

VEE2	Top	Right Side	Bottom	Left Side
	43,44	107,108	223,224	287,288
		124,125		304,305
		146,147		326,327
		163,164		343,344

5 Testability

5.1 SCAN Design Guidelines

Raise to Dick Beaven or Mike Evans any problems whose solution seems to require an exception to any of the following rules. Variations have potential impact on console software development or on testability in Manufacturing and the Field.

5.1.1 Scan Design Rules

The following set of rules define the restrictions applied to scan latch usage.

1. All **A** phase latches are scan latches with a corresponding **B** phase slave. The **B** phase slave is not necessarily unique to the scan latch pair, i.e. it may be used for system data.

2. Scan latch elements can not be driven from STRAM clocks since they are not active during scan operations. If a STRAM clock is used to load latches, those latches can not be restored after a STRAM access. Examples are the latches in STRAM parts themselves and latches in the STREG custom part.

3. There exists a one to one correspondence between scan latch pairs and scan address, i.e. for each address generated there is one and only one latch accessed.

4. **B** phase slave devices copy **A** phase data independent of any state variables. For example, reading ring 0 does not require any other ring to contain specific data.

5. Inversions can exist at arbitrary points in the path. This may be avoided by adding a second version of the **A** phase latch that inverts the data internally. The cost for correcting these inversions is a 2Kbyte RAM on the scan controller to hold the inversion patterns per processor.

6. The clock distribution chip revision field appears as the first n bits of ring 14 and are EXACTLY the same for each MCU. This is required because the revision and type information, which is used to determine the inversion patterns and ring lengths must be

read from the clock distribution chip. While it is possible to map MCU selects to MCU types it is not desirable, especially if variable configurations exist.

5.1.2 Stram Design Rules

STRAM Usage

STRAMs will be used on the CPU module and on the SCU module where RAM storage is required. STRAM clocks will be generated on the clock distribution chip chip and distributed to the STRAMS directly. These clocks are controllable via the scan logic in the clock distribution chip chip.

STRAM Access

There are some memory locations in an Aquarius processor which hold registers which the service processor must access transparently. Transparent access is defined as access which allows the state of the processor to be restored to EXACTLY that state at which the clocks were stopped. There are a number of memories for which this access is highly desirable for error recovery, test case execution, diagnosis and processor/operating system debug. For software consistency, the access to all RAM structures will allow transparent read/write.

STRAM Access Requirements

To implement this strategy requires adherence to the following set of rules:

1. One to one correspondence between STRAM inputs and scanable state devices and one to one correspondence between STRAM outputs and scan mux devices with no combinational logic between the STRAM and scan devices. (See exception below).

Exception: There may be ONE vector (maximum width 32 bits) which when driven into the scan rings will establish this correspondence. (i.e. setting the select line on a mux to steer the data in). This vector is associated with a logical STRAM structure, i.e. EBOX.GPR.

2. There can be no side affects to driving either the setup vector or STRAM inputs. A side affect is defined as any bit pattern which when applied to the scan rings, will cause an input to another STRAM structure, using the same STRAM clock, to change.

This rule can be waived if there are unique controllable STRAM clocks to the structure.

3. For STRAMs that are clocked early in the cycle, i.e. the data held in the STRAM is a function of the previous state, backup latches must be added to the inputs to restore the internal state correctly. These latches may be the same as the scan mux latches used to read output data as long as they satisfy rule 2 above. (Note: A phase STRAMs are still under evaluation).

If a one to one correspondence does not exist and can not be established as per rule 1 above, logic must be added such that the STRAM inputs can be indirectly set. A multiplexer placed between the STRAM and normal system outputs driving the STRAM will allow a set of latches to be added which will satisfy the access requirements. There will be no timing constraints on which phase latches are used as the STRAM clocks are stopped well after the scan clocks have stopped. Note that existing latches may be used as the indirect control registers thereby reducing the overhead.

Any STRAM that satisfies the above is testable and can be accessed transparently. If STRAMs clocked early in the cycle are not backed up, they are still testable, however transparent access can not be guaranteed. Such structures, if they exist, will be marked by the service processor as write only and transparent access to them will not be allowed. (This may be overridden if the operator so decides. If the access results in destruction of state, a reset operation will place the processor in a known state and may be required).

6 STRAMS

6.1 Description

The Self-Timed Random Access Memory or STRAM is a device which has integrated an ECL RAM with address and data latches as well as write pulse generation circuitry. The desired goal was to produce a RAM structure which would allow higher performance in a system environment. By integrating the write pulse generation onto the same chip as the RAM structure, the skew between data and the write pulse is correlated and a significant portion of it may be discounted. The integration of the address and data latches provides savings in wire and output buffer delays.

Since we intend to package the STRAM stuctures on MCUs along with MCA III arrays and other parts, the STRAMs will require series termination or STECL outputs. It is intended, however, that the same part may be sold as a commodity item by the vendor, and therefore the STRAM must also support normal 50 ohm ECL outputs.

The STRAMs have input address and data latch registers, and a data out register. Both the input address register and the input data register are opened with the assertion of the clock. The data output data register is closed with the assertion of the clock and is opened with the unassertion or negation of the clock.

A read access is initiated by the assertion of the clock signal. If the address lines were stable prior to the assertion of the clock and remain so during the assertion of the clock, then the read access time will be governed by the assertion of the clock. If the address lines are not stable prior to the assertion of the clock but only stabilize during its assertion, then the read access time will be governed by the address lines. Address lines must be stable 400 ps. before the negation of the clock to ensure predictable operation. Given that the STRAMs in the Aquarius system will be operated with STRAM CLOCKS which are to be one quarter of a cycle in duration, then the output data should be strictly determined by the read access time and not by the negation of the clock.

Should the STRAMs be operated with a clock width greater than its access time, then data will not be valid until after the negation of the clock.

Write cycles are initiated by the assertion of the write enable signal and with the unassertion of clock.

6.2 Timing rules

T. B. D.

6.3 Testability rules

T. B. D.

6.4 Interconnect rules

6.4.1 Clock

STRAMs used in the AQUARIUS design will be clocked with a differential clock signal generated by a Clock Distribution Chip (CDC). This clock will be one quarter of a machine cycle in duration. Only two orthogonally *adjacent* STRAM chips may be connected to one clock signal pair from the CDC which must be on the same MCU in order to reduce clock skew.. The pair of differential clock signals will be routed adjacently, have a yet to be determined fixed length, and have a controlled number of crossovers.

6.4.2 Address, Chip Select

Address lines should be sourced from an MCA III or equivalent chip on the same MCU as the STRAM chip for performance reasons. However, this requirement is not an inviolable rule, and if required the address lines (and any other inputs except clocks) may be sourced from off MCU. For performance reasons it is probably desirable to limit the loading of address lines to four loads, although up to eight loads can be supported but with an additional timing penalty but only if the address inputs are sourced on the same MCU. If the address lines are sourced off the MCU, then the number of STRAM loads is limited to four, regardless of the

performance criteria. This is due to the extra HDSC etch on the sourcing MCU and concern about the IR drop causing loss of DC noise margin. Again the address lines must be treed to *adjacent* STRAM chips to reduce the timing penalty for additional loads. It is believed that each additional load will add about 300 ps of delay to the worst case (closest to source) load.

6.4.3 Data In, Write Enable

The same rules as apply o the address lines. However, make sure that you can meet the STRAM othogonally adjacency requirement for all inputs (address, data in, chip select, and write enable) simultaneously.

6.4.4 DATA OUT

The DATA OUT lines may drive off of the MCU and/or module as long as there is only one source on the line. Two orthogonally *adjacent* STRAMs may have their outputs wire "ORed" if the designer guarantees that only one of the STRAMs has active outputs. In other words only, one of the STRAM chips tied together may be selected by the assertion of CHIP SELECT. Outputs which are wire "ORed" are *are not allowed to go off the MCU* and will be more restricted in length (3 inches for example). Due to restrictions on the length of etch between the two wire "ORed" STRAMS, the STRAMS must be adjacent. The HDSC etch length between the two wired OR STRAM outputs must be less than 0.75 in.

7 Signal Naming Conventions

The following rules define the signal naming conventions to be used for the Aquarius design.

7.1 Character length

The legal character length for a signal may be 32 characters, not including the bit subscript.

7.2 Legal characters

The legal characters for a signal name (not including bit subscript) are the UPPERCASE Alpha characters (A-Z), numeric characters (0-9), and the underscore character. No imbedded spaces are allowed.

7.3 Assertion levels

Assertion levels must appear on all signals. The assertion levels are represented by an L for asserted low and an H for asserted high. The assertion level is the last character in a scaler signal name and is preceded by an underscore. The assertion level is the last character before the bit subscript in a vectored signal and again the assertion level is preceded by an underscore. The assertion level should correspond to the polarity of the source component.

7.4 Vectored signals

The width of a vectored signal is denoted with a bit subscript. The format of the bit subscript is [MSB:LSB]. MSB and LSB may be numeric characters only and MSB must be greater than LSB. No multiple subscripts between brackets are allowed.

7.5 Examples

Scaler: `LOAD_OVERFLOW_REG_H`

Vector: `MBOX_IB_DATA_H[63:0]`

7.6 Logically equivalent signals

A signal which is logically equivalent to another signal should have the same signal name with only the following variance. When two or more signals are logically equivalent (and have the same assertion level) then each signal name should have an single letter bounded on both sides by an underscore and should be located just preceeding the assertion level. If a signal has a single letter preceding the assertion level, then there must be at least one other logically equivalent signal.

Logical equivalents:

```
TEST_A_L
TEST_B_L
```

Not logical equivalents:

```
MY_A_CLK_L
MY_B_CLK_L
```

Nor:

```
ZAP_L
ZAP_H
```

7.7 Clock assertion levels

First, ASSERTION of a clock at a latch is defined as that level which causes the latch to open. Therefore a clock may be defined as ASSERTED or UNASSERTED. Although latches may be clocked by the inverted phases (in a scannable flip/flop for example), we will use ASSERTED and UNASSERTED to apply to those levels of the clocks which are predominately used by the latches in the machine. With a differential clock, the two clock lines must be in opposing states. The clock is undefined if both of the clock signals of a differential pair are at the same logic level.

Logical clock lines may be assumed to be asserted low. A clock signal with no explicit assertion level such as CLK BB will be assumed to have a low assertion level. NOT CLK BB or CLK BB will be assumed to have an assertion level of high to ASSERT the clock.

The following are equivalent signals;

```
CLK AA
CLK_AA_L
```

and are asserted when low.

Whereas the following are equivalent;

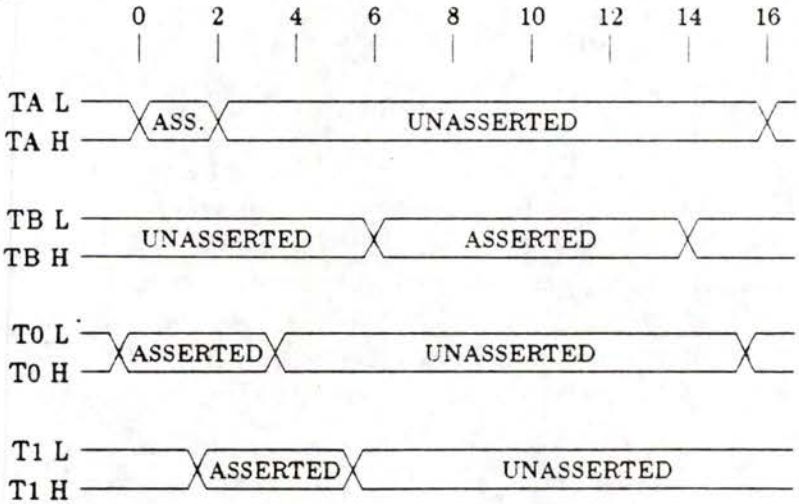
```
CLK_AA_H
NOT CLK AA
~CLK AA
```

and are asserted when high.

The preferred designation will be `CLK_AA_L` and `CLK_AA_H` to represent the two physical signals which comprise the differential pair.

In text descriptions, clocks should be defined as either ASSERTED or UNASSERTED or as LOADING or HOLDING in preference to defining them as LOW or HIGH. This should remove some of the ambiguity of which clock signal to which you are applying the description.

Examples:



8 MCU/MCA/STRAM Names

The following conventions will be followed in assigning names to MCU's, STRAMs, and to MCA's on the Aquarius project. Each logical entity will have a unique name within logical entities and each physical entity will have a unique name within physical entities. The name for a physical entity as used here is not the same name as a physical reference designator. Each physical entity will eventually have a physical reference designator but theere are more physical reference designations than there are physical entities. A physical entity is used to represent the fact that some logical designs are used multiple times in the AQUARIUS design. When there is a one to one mapping between the logical entity and the physical entity then the same name is used to represent both. When a logical entity represents multiple physical entities, then both the physical and logical names will be unique but are related in a predetermined manner.

8.1 MCU's

An MCU name shall be comprised of three uppercase alpha characters or of 2 alpha characters and a numeric character. An X shall be used as the last character in a logical name which represents more than one physical MCU. The numeric character may only be used if there are two or more physical entities represented by the same logical entity. The numerics should start with 1 and increase monotonically.

For example:

MCX could be the logical name for an MCU which will have four physical copies (MC1, MC2, MC3, and MC4)

As the last character of a physical name the numeric will uniquely identify the device.

ADD could be name of an MCU that will have only one physical copy.

8.2 MCA's

An MCA name will consist of 4 alpha characters and again the last character may be a numeric indicating multiple physical copies of that particular array. Again, X is reserved as the last character for those logical names representing more than one physical MCA. All uniquely designed MCA's must have unique logical names and each physically implemented MCA must have a unique physical name. If an MCA is used more than once in an MCU that is likewise used more than once, then each MCA used must have a unique name.

MC1 (MCU #1) Contains MCA1 and MCA2 and

MC2 (MCU #2 - copy of MC1) Contains MCA3 and MCA4 where MCA1, MCA2, MCA3, and MCA4 are copies of the same MCA.

Four character names which are the same except for ending in A, B, C, etc. may be used to indicate chips of similiar but not identical functionality.

MCA naming conventions shall also apply to the custom chips with the exception of the CDC chip which will be handled differently.

8.3 STRAMs

STRAM names will consist of five uppercase alpha characters where the last two characters may be numeric. STRAM names may be assigned to a group or array of STRAMs of not more than 99 strams. The logical name for the cluster must end in two X's. An intermediate grouping of up to 9 STRAMs is named by the three unique uppercase alpha characters as in the logical cluster name plus either another alpha character if the the grouping is unique to itself or numeric if it is part of a larger logical grouping of STRAMs. The last character should again be an X in this case. To uniquely indentify a STRAM within that group each STRAM would be assigned a name where the X is replaced with a numeric digit between 1 and 9.

For example STRXX could be the logical name for an array of STRAMs containing 12 stram devices. Two logical groups of 6 STRAMs each are created in order to trade off physical locations with MCAs. These two groups would be named STR1X and STR2X respectively. The individual STRAMs of the first group would have the names: STR11, STR12, STR13, etc. and the STRAMs of the second group would be named STR21, STR22, STR23, etc.

Each MCA, MCU, STRAM, or custom chip in the CPU and the JBOX shall have a unique designation.

9 Custom Chips

9.1 Macro library

T. B. D.

9.2 Technology constraints

T. B. D.

10 MCU

10.1 Physical layout constraints

The MCU provides a high density signal interconnect medium to interconnect several TAB'ed chips together and to interconnect them externally via connectors. The MCU provides an orthogonal pair of signal layers with reference planes above and below the pair. The signal interconnect density is 3 mil or about 330 tracks per inch. The MCU will contain a Clock Distribution Chip plus up to eight MCA III or custom chips. Each MCA III or Custom Chip could be replaced by approximately nine STRAM chips, depending upon STRAM size and orientation. Connectors on each of the four sides of the MCU provide 804 pins for I/O. A few of the pins are reserved for dedicated signal functions, but none are required for ground or reference which has been provided for separately.

Power distribution layers are also provided on the MCU for ground, VEE1 (-5.2 volts), and for VEE2 (-3.4 volts).

10.2 Power constraints

T. B. D.

10.3 Pin constraints

There are four signal connectors provided on the periphery of the HDSC of the MCU. Viewd from the component side of the HDSC the connectors are labled FP1, FP2, FP3, and FP4 starting with the right hand connector and proceeding counter clockwise. Each connector consists of 269 pins of which 68 are reference signals and will be connected to ground. Pin numbers 1 + 4*i where i = 0, 1, ...,67 are grounds. The rest of the pins may be input or output signals and are available to the designer with a few exceptions. The exceptions are the following reserved pins:

FP3.91	MCLK_L
FP3.92	MCLK_H
FP3.178	RCLK_L

FP3.179	RCLK_H
FP3.138	CD_CLK_CNTL_H
FP3.148	CLOCK_CHECK_IN_H
FP3.120	CLOCK_CHECK_OUT_H
FP2.123	SBUS_SELECT_IN_H[0]
FP2.124	SBUS_SELECT_IN_H[1]
FP2.131	SBUS_SELECT_IN_H[2]
FP2.130	SBUS_SELECT_IN_H[3]
FP2.120	SBUS_SELECT_OUT_H[0]
FP1.152	SBUS_SELECT_OUT_H[1]
FP1.151	SBUS_SELECT_OUT_H[2]
FP1.150	SBUS_SELECT_OUT_H[3]
FP1.75	SBUS_A_CLK_IN_H
FP1.78	SBUS_A_CLK_OUT_H
FP1.76	SBUS_B_CLK_IN_H
FP1.79	SBUS_B_CLK_OUT_H
FP1.146	SBUS_FCT_IN_H[0]
FP1.142	SBUS_FCT_IN_H[1]
FP2.119	SBUS_FCT_OUT_H[0]
FP2.118	SBUS_FCT_OUT_H[1]
FP1.74	SBUS_DATA_IN_H
FP1.80	SBUS_DATA_OUT_H
FP1.147	SBUS_HDSC_SEL_H

10.4 Clock constraints

The master clock signal pair and the reference clock signal pair will be prerouted from the connector pins to the Clock Distribution Chip. Also, the buffered master clock and gated reference clocks from the Clock Distribution Chip to the MCA III chips will also be prerouted. This is to ensure that the lengths of these runs will be identical from one MCU to the next. The STRAM clocks will require that the router route them with a fixed length (yet to be determined). Unfortunately it is not possible to preroute the STRAM clocks as there are too many options. These clock pairs should also be flanked on both side by an unused track.

10.5 ECO's

ECO's are constrained to being chip replacements only. There is no planned capability to ECO the signal interconnect on the HDSC.

10.6 Testability constraints

The current testability constraints are to implement those SCAN and CLOCK pins which are already reserved.

11 Module

11.1 Physical layout constraints

The CPU module will be constructed on a 24" x 24" module wtih 24 layers, comprised of 5 signal layer pairs, a clock layer, 11 reference planes, and two pad layers. This module will provide the signal interconnectivity for up to 16 MCU packages, and a row of connectors on one edge of the module. This module will provide for signal and reference planes only and specifically will not be used for power distribution.

The System Control Unit (SCU) module will be constructed on a 24 inch by 18 inch module, with the same 24 layers as the CPU module. The SCU is designed to support 6 MCU packages and numerous connectors. There are connectors for clock distribution, connection to up to four CPU modules, connections to the memory backplane, and connections to the I/O subsystem.

At this time all layer pairs are orthogonal pairs and run parallel to the edges of the module.

11.2 Power constraints

T. B. D.

11.3 Pin constraints

T. B. D.

11.4 Clock constraints

Each MCU receives the master clock signal and reference clock signal from signal power dividers mounted on the back side of the module. Each MCU in turn must have a receiving Clock Distribution Chip (and only one) which receives these clocks and generates appropriate clocking signals for other chips on the MCU. If an MCU is not present then the clock must be terminated with ?????.

11.5 Testability constraints

There will be a 300 pin connector on the CPU module dedicated for bringing out a number of signals to assist in testing/diagnosing the module. Selection of the signals to be brought out to the connector will be determined by the designers.

12 Interconnect Rules

All Hi-Tech connections will be made using Series Teminated Emitter Coupled Logic (STECL) with 100K ECL temperature compensated compatible levels. The series termination is a result of the relatively high resistance of the interconnection medium on the MCUs and the need to maintain good noise margins for reasonable lengths of interconnect. This does mean that wire "ORing" is difficult at best and has to be explicitly supported with special STECL outputs requiring a minimum of two pins per output or three pins for a generalized output. Also, because of the high resistivity of the interconnect, the lengths of interconnect possible with wired "OR" outputs are severely shortened. Therefore, it has been deemed that no wire "ORing" be allowed on the Hi-Tech portion of the machine between chips other than in a very restricted circumstance around STRAMs.

12.1 MCA to MCA within MCU

There should be no more than four loads on a net. For critical nets, loads should probably be restricted to one. Remember that the worst case delay is the trip from the source to the farthest load and then back to the load which is closest to the source. It is not planned that at the present time that the treeing order can be specified.

Effectively, daisy chain routing will be used for connections on the MCU, minimizing stub lengths.

12.2 MCA to STRAM within MCU

Nets which are sourced by an MCA and loaded by a STRAM shall be restricted to absolutely no more than eight loads which must be clustered together as adjacent STRAMs. A good design goal is to restrict the loading to four loads if interconnect delays are at all important.

12.3 STRAM to MCA within MCU

There should be no more than four loads on a net. For critical nets, loads should probably be restricted to one. Remember that the worst case delay is the trip from the source to the farthest load and then back to the load which is closest to the source.

12.4 MCA to MCA within module

Nets which are sourced on one MCU and include at least one load on another MCU shall be restricted to only two MCU's. In otherwords, a source on one MCU may not be loaded by two other different MCU's. A net should not be loaded on the same MCU as its source if it also is loaded by another MCU. Otherwise, the round trip delay times become very long.

There shall not be more than four loads on the net. Critical nets should be restricted to one or two loads.

12.5 MCA to STRAM within module

Nets which originate on another MCU on the same module may drive up to 4 STRAM loads. No other loads are allowed on the net. The STRAMs must be orthogonally adjacent. No more than the source MCU and one load MCU may be involved in the net.

12.6 STRAM to MCA within module

Nets which are sourced on one MCU and include at least one load on another MCU shall be restricted to only two MCU's. In otherwords, a source on one MCU may not be loaded by two other different MCU's. A net should not be loaded on the same MCU as its source if it also is loaded by another MCU. Otherwise, the round trip delay times become very long. Nets which have two STRAMs wire "ORed" are not allowed off MCU.

There shall not be more than four loads on the net. One or two loads should be considered for critical nets.

12.7 MCA to MCA between modules

Nets which are sourced on one module and loaded on another module must be differential pairs. The differential pairs will have only one load and will be received by a differential receiver macro. The differential pair will be routed as an adjacent pair at all levels of interconnect which include cable, module, and HDSC.

12.8 MCA to STRAM between modules

Not allowed. Since signals between modules must be differential pairs and there are no differential receivers on STRAMs (other than clock), then no MCA to STRAM connections between modules are allowed.

12.9 STRAM to MCA between modules

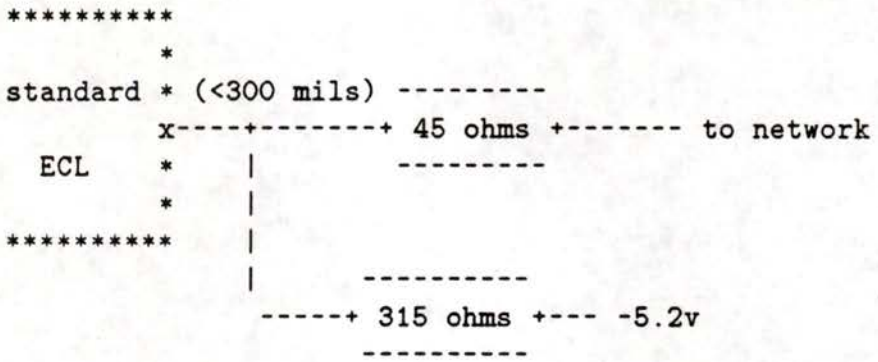
Not allowed. Since there are no differential drivers on the STRAMs then no STRAM to MCA connections between modules are allowed.

12.10 MCA to non-MCA

No more than 2 loads on a net. The SCU module and the memory backplane module are tightly coupled together electrically and for our purposes may be considered to be a single module. Signals which are otherwise, between modules, must be differential and have a single load. These inlcude CPU and SCU to console connections and SCU to XJA connections.

12.11 Non-MCA to MCA

Non-MCA ECL parts will require a source termination consisting of a pull down resistor of value 315 ohms connected between the source output pin and -5.2V. Additionally, a series resistor of value 45 ohms must be connected from the source output pin and the network which it will drive. Less than 300 mils of etch should connect the output pin and the resistor pins.



If the signal is sourced on the memory module and loaded on the SCU, then two loads in the same MCU are allowed. Other connections between modules must be made with a differential pair. These include Master Clock Module (MCM) to CPU and SCU connections, console to CPU and SCU connections, memory module to SCU connections, and XDA to SCU connections. Differential signals may have only one load.

12.12 Lo-Tech Connections

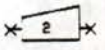
T. B. D.

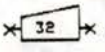
APPENDICES

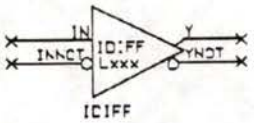
A RTL Bodies

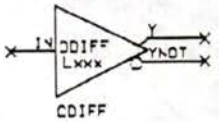
Legal RTL bodies as of January 19, 1986.

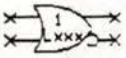
RTL Bodies as of 1/16/87

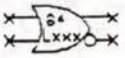


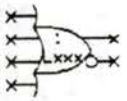


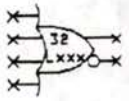


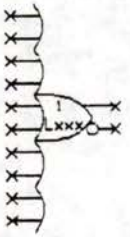


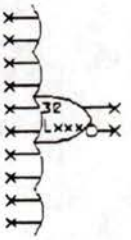


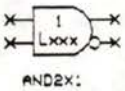


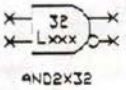


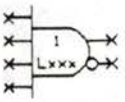


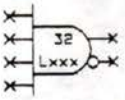


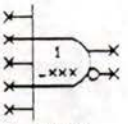


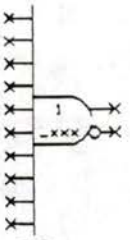


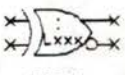


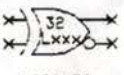


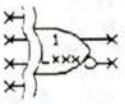


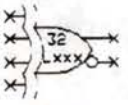


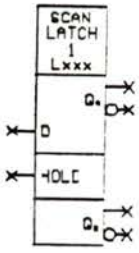




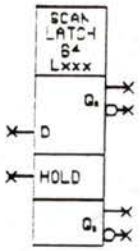




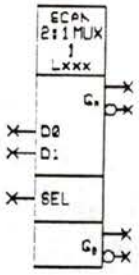




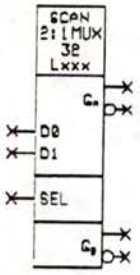
SCANLATCH1



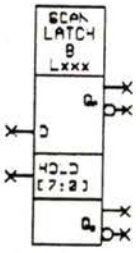
SCANLATCH64



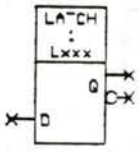
SCANMUX2X1



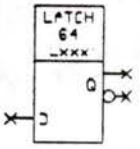
SCANMUX2X32



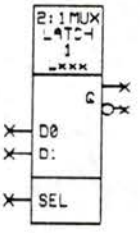
SCANLATCH8



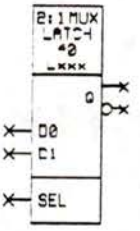
LATCH1



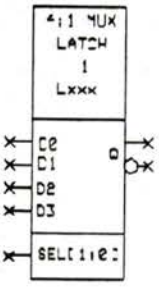
LATCH64



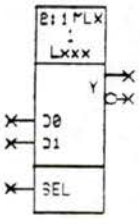
MUX2LATCH1



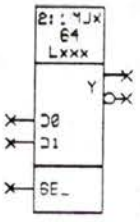
MUX2LATCH40



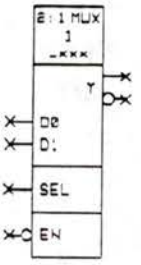
MUX4LATCHX1



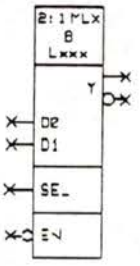
MUX2X1



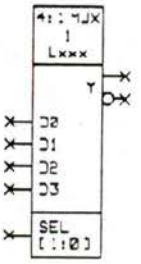
MUX2X64



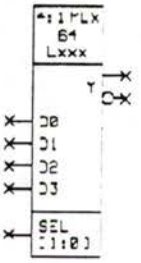
MUXEN2X1



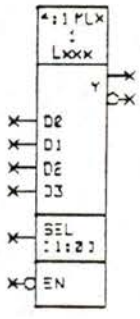
MUXEN2X8



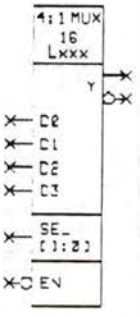
MUX4X1



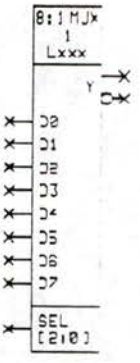
MUX4X64



MUXEN4X1

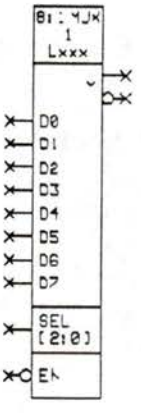


MUXEN4X16

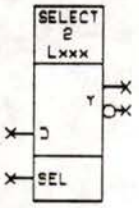


MUX8X1

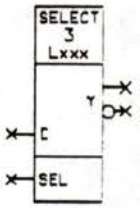
MUX8X32



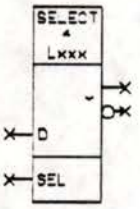
MUXEN8X1



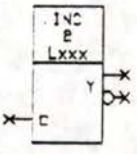
2SELECT



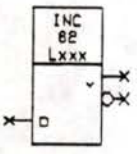
3SELECT



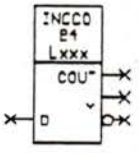
4SELECT



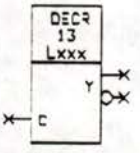
INC2



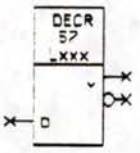
INC32



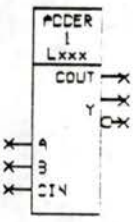
INCCO24



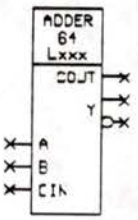
DECR13



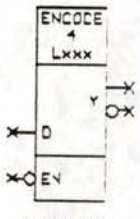
DECR57



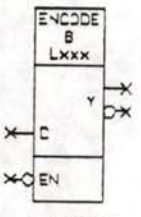
ADDER1



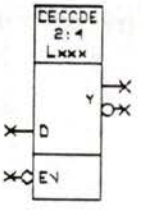
ADDER64



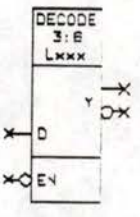
U4ENCODE



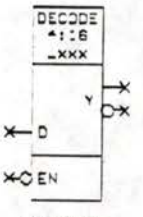
U8ENCODE



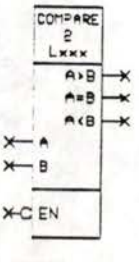
U2DECODE



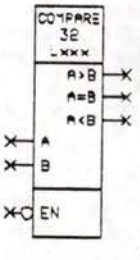
U3DECODE



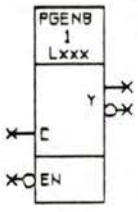
U4DECODE



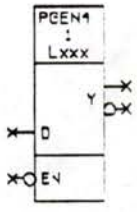
COMPARE2



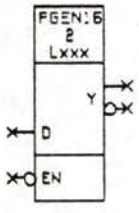
COMPARE32



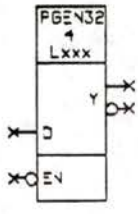
PGEN8X1



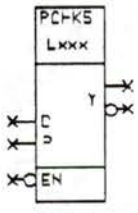
PGEN4X1



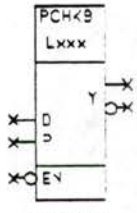
PGEN16X2



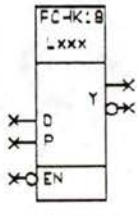
PGEN32X4



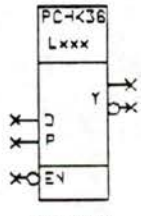
PCHK5X1



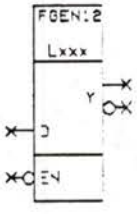
PCHK9X1



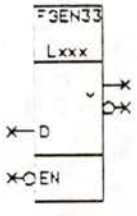
PCHK18X2



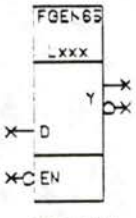
PCHK36X4



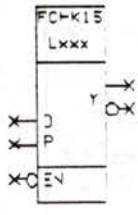
PGEN12X1



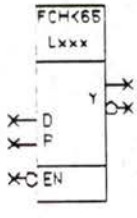
PGEN33X1



PGEN65X1

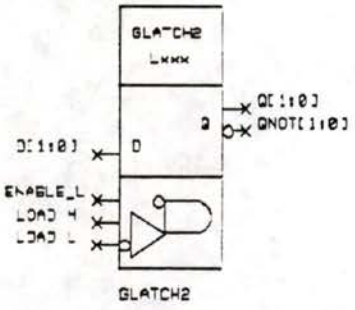


PCHK15X1

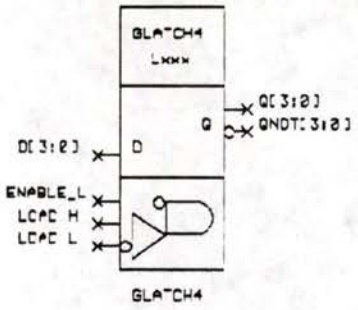


PCHK65X1

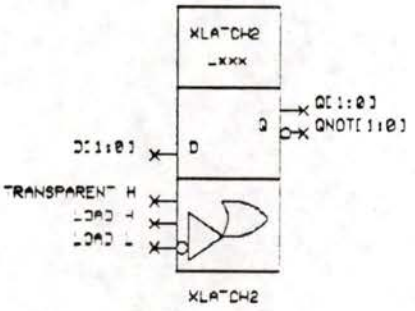
These bodies are not for general use. They are to be used in the I/O interface and SCAN control paths only.



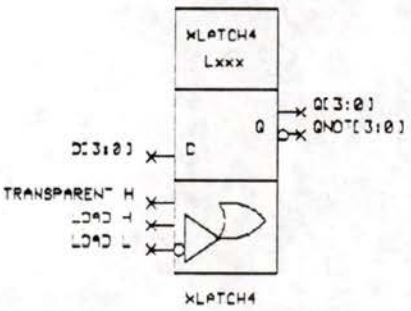
GLATCH2



GLATCH4

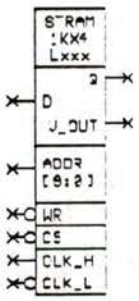


XLATCH2

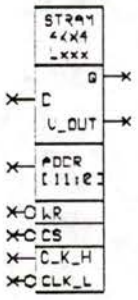


XLATCH4

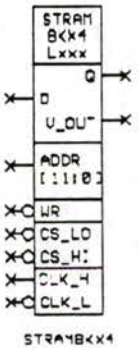
These STRAM bodies are found in the STRAM library and not in the RTL library.



STRAM1KX4



STRAM4KX4

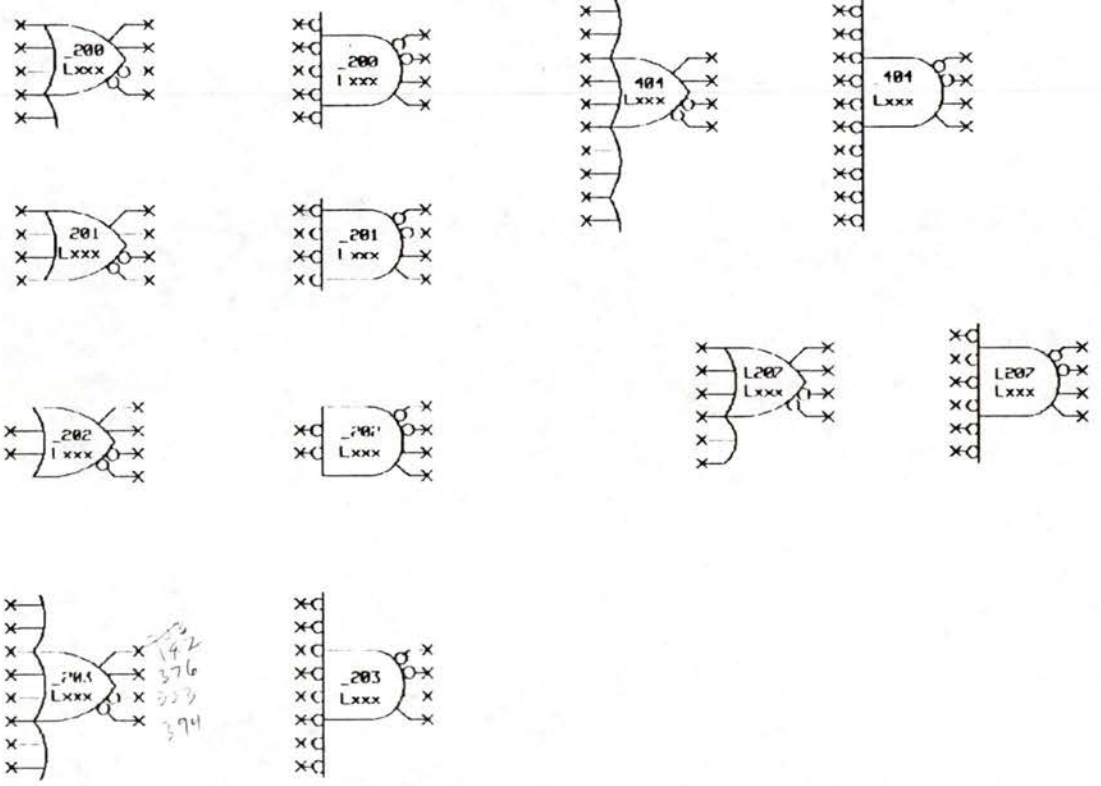


STRAM8KX4

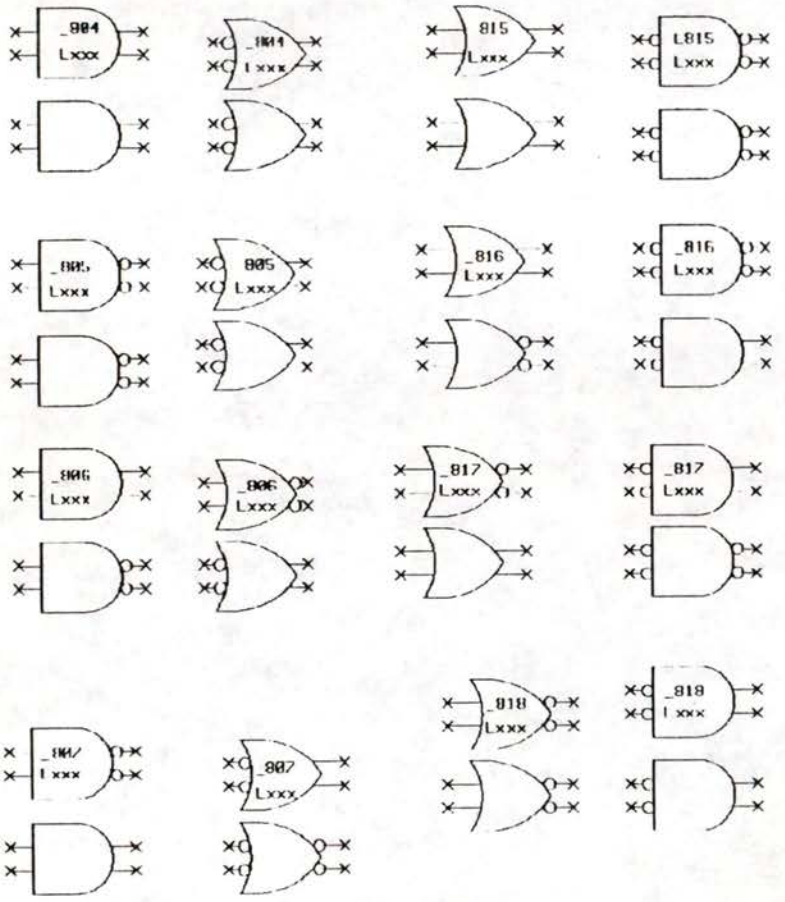
B Macro Drawings

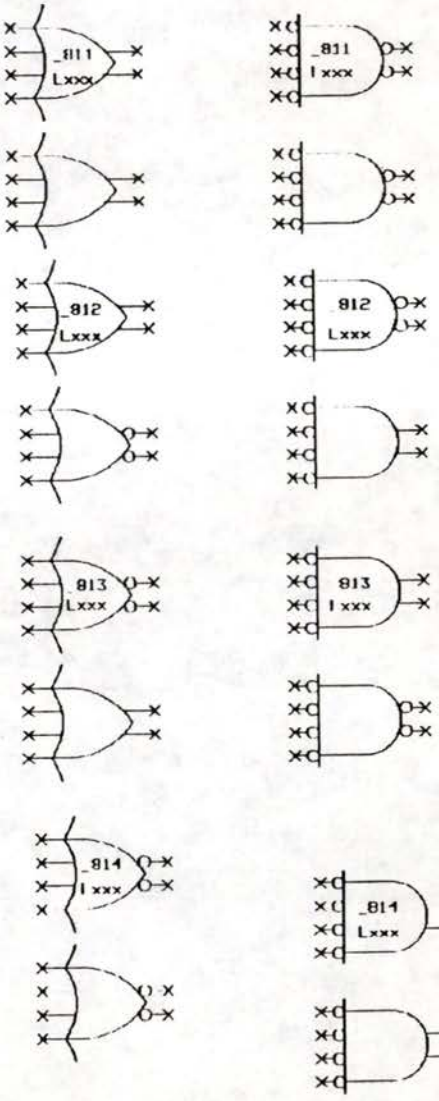
Legal CLEGO bodies as of January 19, 1986.

Simple Gates

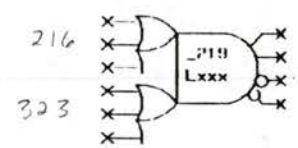


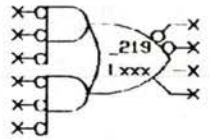
Simple Gates

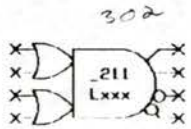


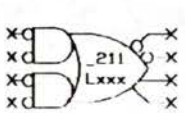


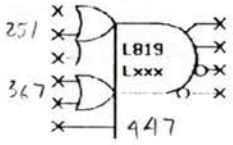
AND/ORs

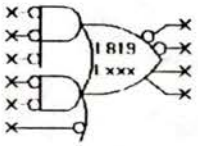

216
323
_219
Lxxx

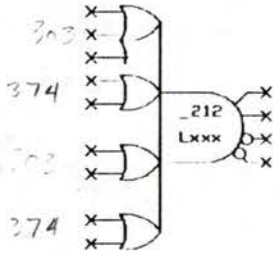

_219
Lxxx

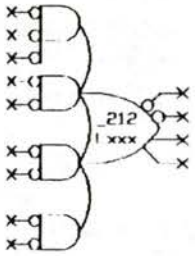

302
_211
Lxxx

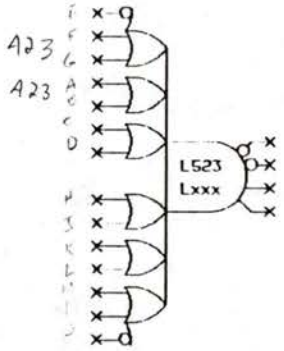

_211
Lxxx

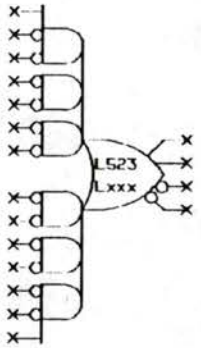

251
367
L819
Lxxx
447

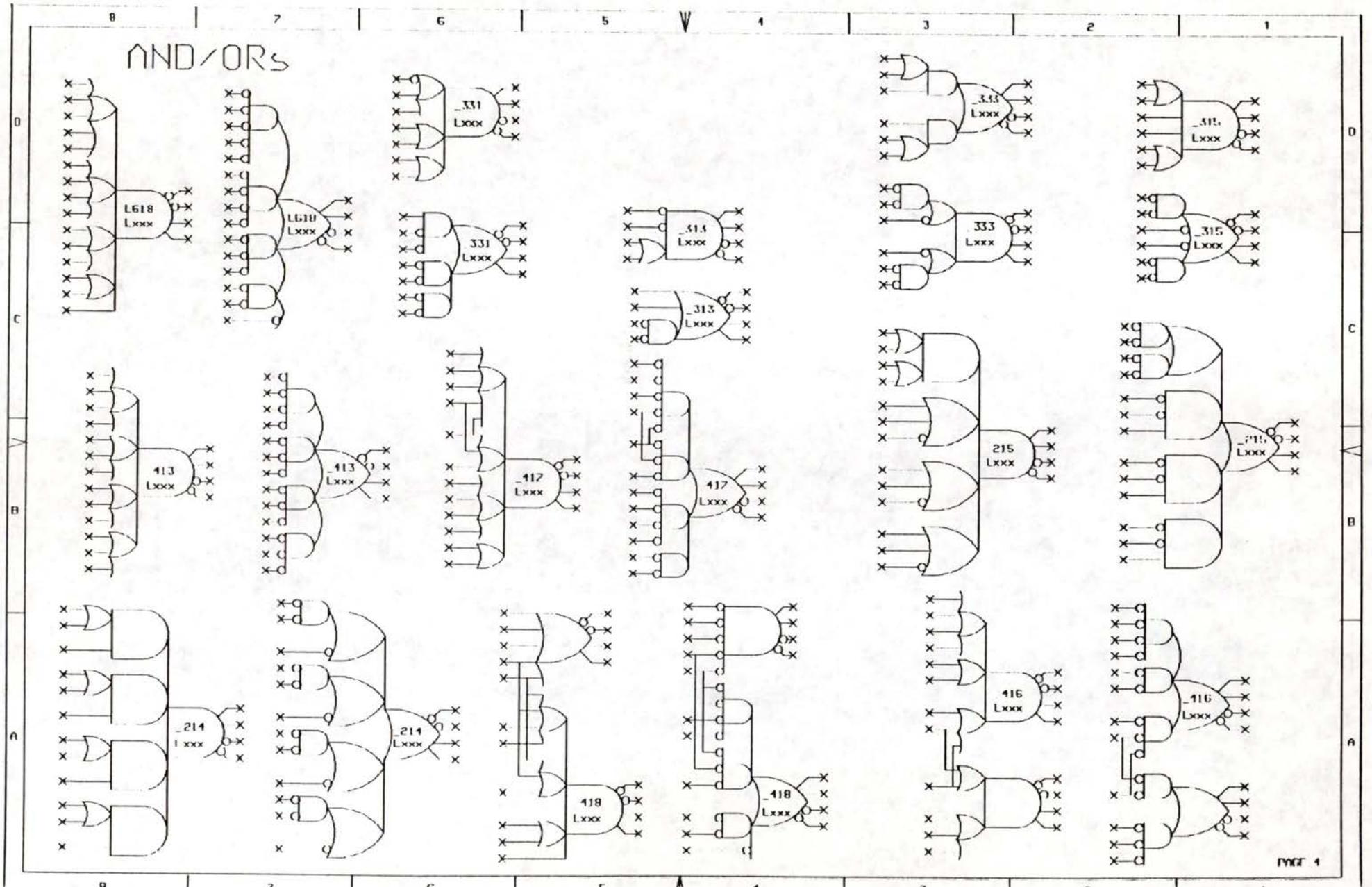

L819
Lxxx

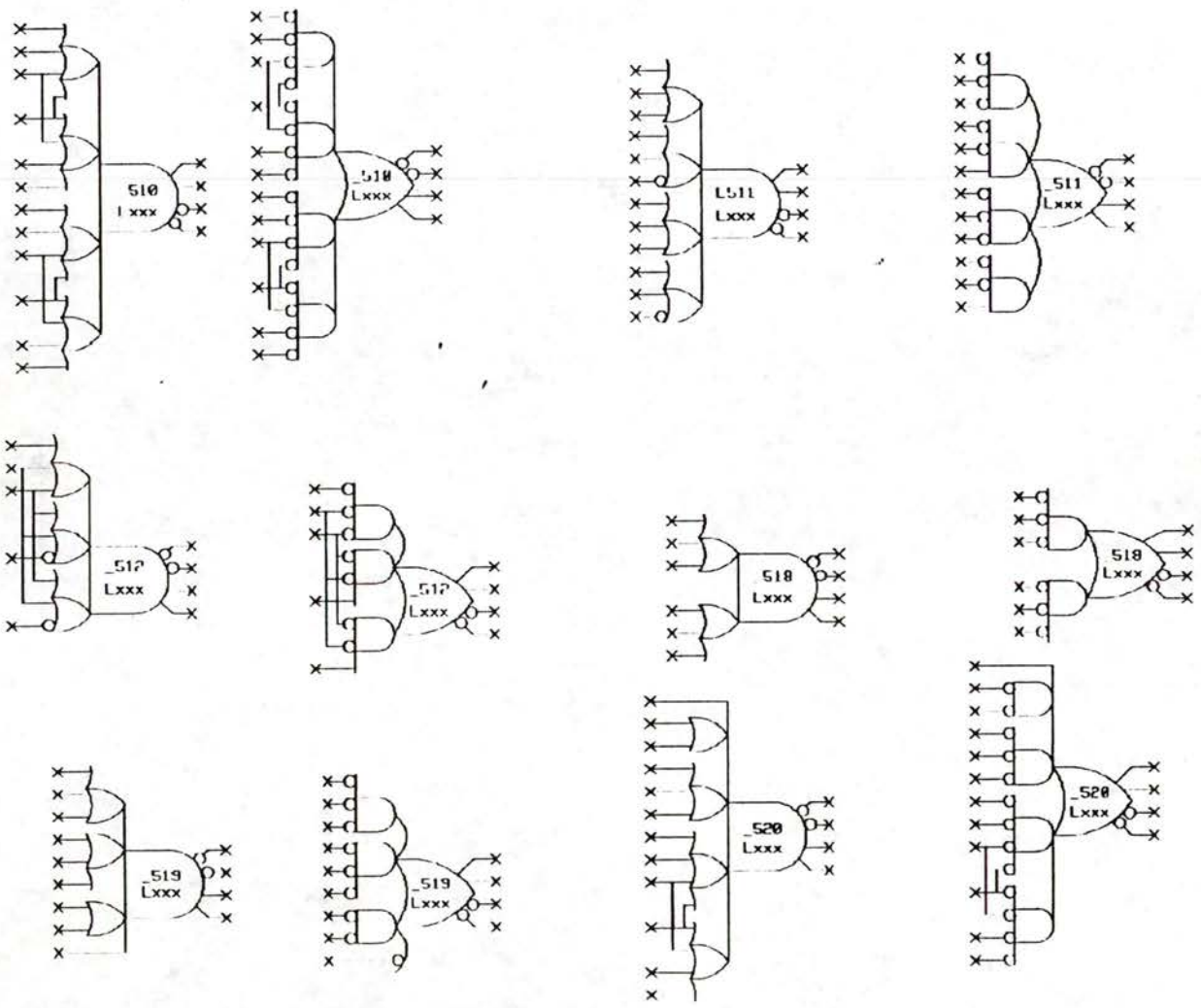

303
374
374
_212
Lxxx


_212
Lxxx

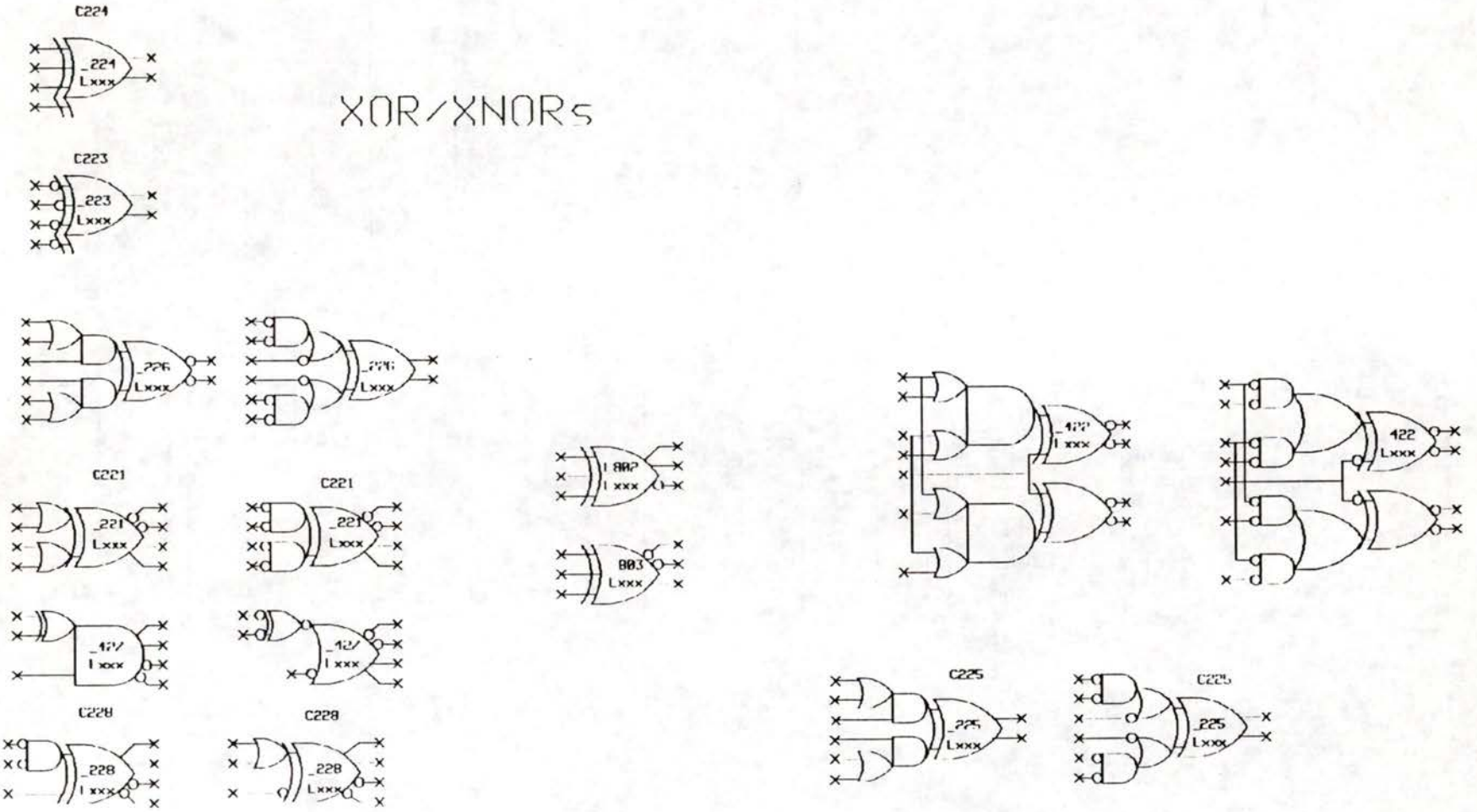

423
423
L523
Lxxx

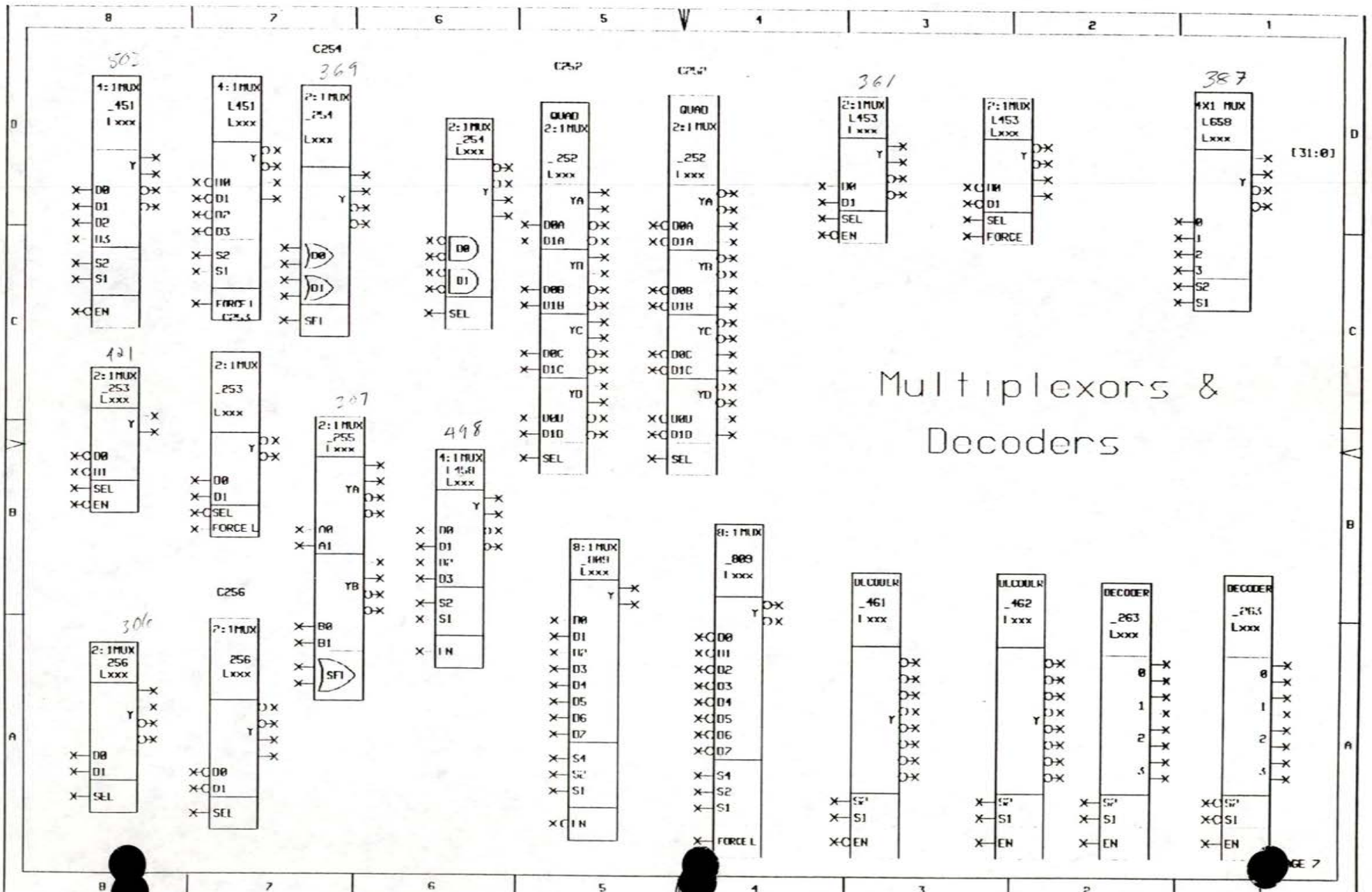

L523
Lxxx

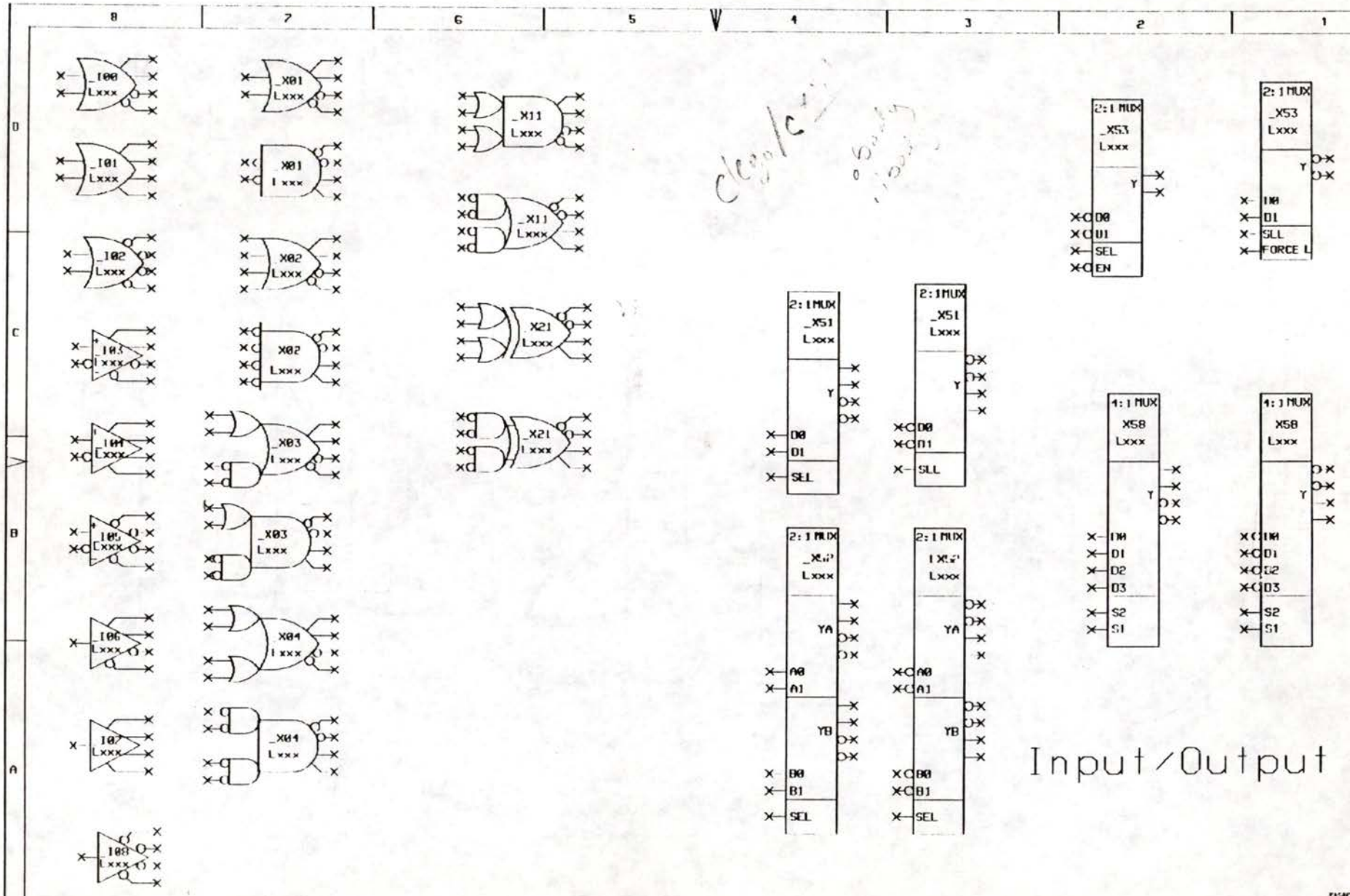

AND/ORs
L618
Lxxx
L610
Lxxx
_331
Lxxx
331
Lxxx
_314
Lxxx
_313
Lxxx
_333
Lxxx
333
Lxxx
_315
Lxxx
413
Lxxx
_413
Lxxx
_417
Lxxx
417
Lxxx
215
Lxxx
_214
Lxxx
_214
Lxxx
418
Lxxx
_418
Lxxx
416
Lxxx
_416
Lxxx
PAGE 1

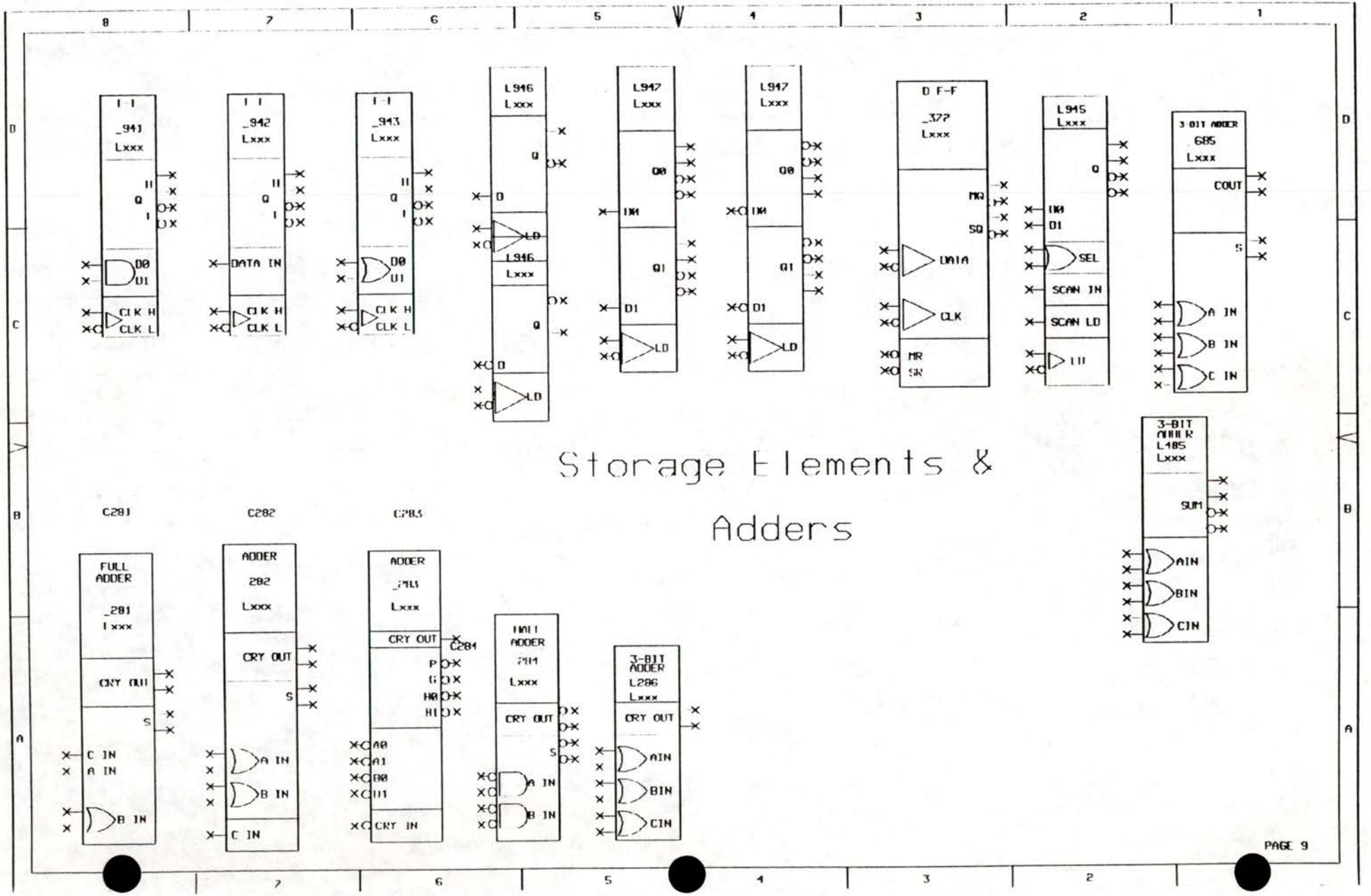

AND/ORs
_510
Lxxx
_510
Lxxx
_511
Lxxx
_511
Lxxx
_512
Lxxx
_512
Lxxx
_518
Lxxx
_518
Lxxx
_519
Lxxx
_519
Lxxx
_520
Lxxx
_520
Lxxx

XOR/XNORs


C224
_224
Lxxx
C223
_223
Lxxx
_226
Lxxx
_226
Lxxx
C221
_221
Lxxx
C221
_221
Lxxx
_227
Lxxx
_227
Lxxx
C228
_228
Lxxx
C228
_228
Lxxx
882
Lxxx
883
Lxxx
_422
Lxxx
422
Lxxx
C225
_225
Lxxx
C225
225
Lxxx


Multiplexors &
Decoders
4:1MUX _451 Lxxx
4:1MUX L451 Lxxx
2:1MUX _254 Lxxx
C254
C252
C253
QUAD 2:1MUX _252 Lxxx
2:1MUX L453 Lxxx
4X1 MUX L658 Lxxx
[31:0]
2:1MUX _253 Lxxx
2:1MUX _255 Lxxx
4:1MUX L458 Lxxx
C256
2:1MUX 256 Lxxx
8:1MUX _849 Lxxx
8:1MUX _800 Lxxx
DLCODER _461 Lxxx
DLCODER _462 Lxxx
DECODER _263 Lxxx
DECODER _263 Lxxx


_I00
_I01
_I02
_I03
_I04
_I05
_I06
_I07
_I08
_X01
X01
X02
X02
_X03
_X03
X04
_X04
_X11
_X11
X21
_X21
2:1 MUX
_X53
D0
D1
SEL
EN
Y
2:1 MUX
_X53
IN
D1
SLL
FORCE L
Y
2:1 MUX
_X51
D0
D1
SEL
Y
2:1 MUX
_X51
D0
D1
SLL
Y
4:1 MUX
X58
IN
D1
D2
D3
S2
S1
Y
4:1 MUX
X58
IN
D1
D2
D3
S2
S1
Y
2:1 MUX
A0
A1
B0
B1
SEL
YA
YB
2:1 MUX
A0
A1
B0
B1
SEL
YA
YB
Input/Output
page 8


_941
Lxxx
D0
D1
CLK H
CLK L
_942
Lxxx
DATA IN
_943
Lxxx
L946
Lxxx
LD
L947
Lxxx
D0
D1
D F-F
_377
Lxxx
DATA
CLK
MR
SR
L945
Lxxx
SEL
SCAN IN
SCAN LD
3 BIT ADDER
685
Lxxx
COUT
A IN
B IN
C IN
Storage Elements &
Adders
C281
C282
C283
FULL
ADDER
_281
Lxxx
CRY OUT
C IN
A IN
B IN
ADDER
282
Lxxx
ADDER
_283
Lxxx
C284
CRY IN
ADDER
284
Lxxx
3-BIT
ADDER
L286
Lxxx
3-BIT
ADDER
L485
Lxxx
SUM
AIN
BIN
CIN

C Macro Delays

T. B. D.

D MCA III Specification

MCA3 DESIGN MANUAL

Revision 1.2
November 20, 1986

Marc Lamere
John Hackenberg
Amar Manohar

Digital Equipment Corporation

Table of Contents

1.0 THE MACROCELL ARRAY CONCEPT

The Macrocell Array is a VLSI gate-array chip which uses cells called macros as logic building blocks to design custom logic. Each cell in the gate-array contains only resistors and transistors. This cell is a generitic element since the devices that it contains are not connected together to form a logic function until the designer defines the cell and uses metal to wire it. A library of macros has been created which interconnects the resistors and transistors to form these logic functions. The Macros form such logic functions as AND, OR, Latches, D flip-flops, and many other logic functions. All Macros are comprised of series-gated ECL structures to optimize performance.

To generate a chip design, the designer need only be concerned with developing circuits by selecting Macros from the library of Macros available in the MCA10000ECL gate-array.

Compared with the conventional approach to custom LSI circuits, the Macrocell approach offers a reduction in delivery time. Fully diffused wafers are stockpiled and only the interconnect needs to be added to a wafer to obtain a customized MCA3.

2.0 FEATURES OF THE MCA10000ECL GATE ARRAY

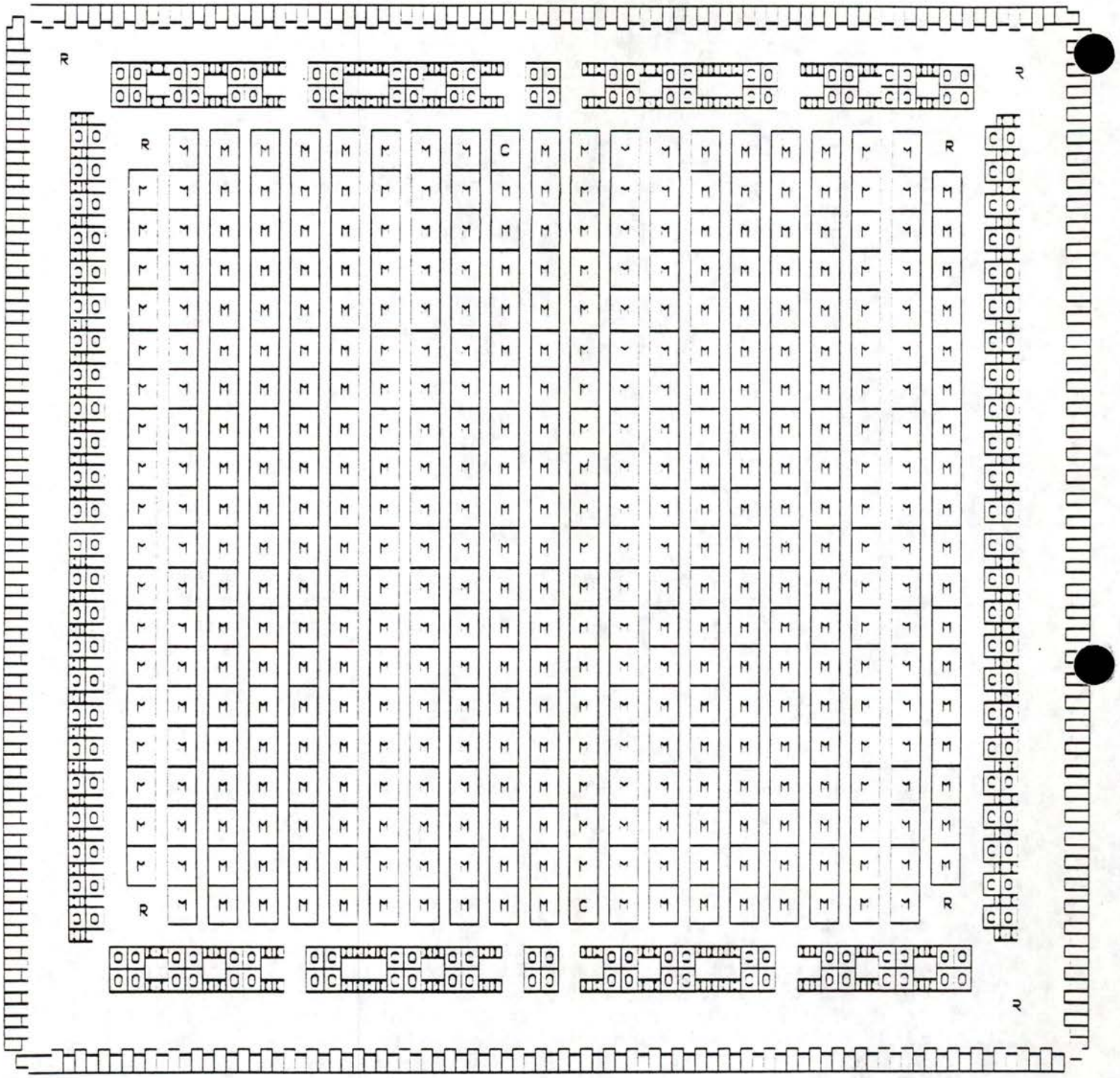
The MCA10000ECL, a gate-array having 10,000 equivalent gates and 256 I-O pins, has been developed using an advanced silicon bipolar process. This process technology utilizes a transistor structure which employs polysilicon base, emitter, and collector electrodes and a resistor structure which also uses polysilicon. This process has been used to demonstrate gate delays of 150 pS.

This very high performance ECL macrocell array, MCA10000ECL, has an approximate gate density of 10,000 equivalent gates. Very low thermal resistance packaging and special low inductance low capacitance packaging and interconnect will be necessary to exploit its highest potential performance.

The gate array floor plan is shown in Figure 1. It is comprised of a central core area of 414 major cells (MCELLS) which are divisible into quarter cell functions and arranged in an array of 20 rows and 21 columns (minus 6 sites for the master bias generators and special clock generator circuits). A ring of 200 Output cells (OCELLS) interspersed by 224 Input cells (ICELLS) surround the Major cells for interfacing to the pad drivers. The 256 pad cells (including I-O pads, called PCELLS) plus the 104 power pads lie at the periphery of the chip. The metalization system will include 3 layers of interconnect. The customized routing metal will reside on the first two metal layers (Metal 1 and Metal 2) with interconnecting vias between. The top metal layer (Metal 3) and parts of Metal 1 and 2 provides power and ground

Figure 1

MCPIII GATE ARRAY FLOORPLAN



M - MCELL

I - ICELL

O - OCELL

C - CLOCK CIRCUIT

R - REGULATOR

distribution.

Previous gate array designs have provided only two levels of series gating thereby limiting the complexity of functions which can be designed within one current switch. Within this gate array, three levels of series gating is possible at both major and output macro cells. This provides additional "and" (product) gate functions at very high speed within one switch delay and at a lower power level. Figure 2 shows the advantage of three level series gating verses two level series gating for a typical logic function.

All current switches within the array are powered from the main supply voltage VEE1. If three level series gated functions are used then VEE1 is set to -5.2V. If only two level series gated functions are used throughout the array, then VEE1 may be set to -4.5V. Also, ICELLS are powered from a second, lower supply voltage VEE2 = -3.4V to save power.

The output emitter followers of M, I, and O CELLS, as well as STECL output followers employ constant current source pulldowns to VEE2 to save power. The constant current source pulldowns minimize the sensitivity of ac performance to power supply variations.

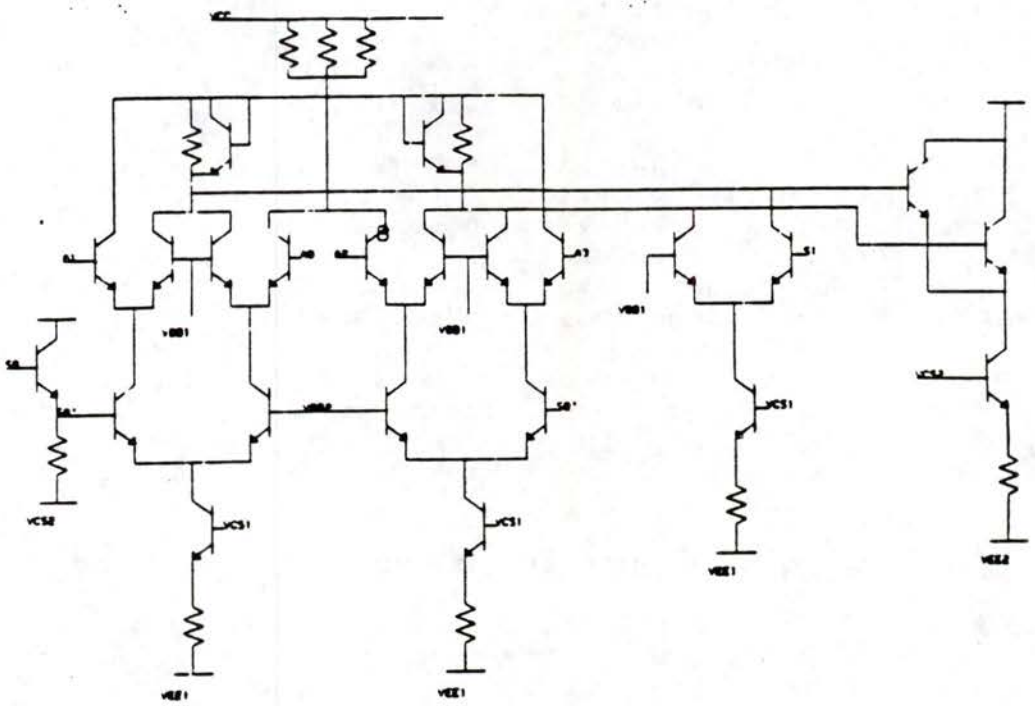
The overall power dissipation is limited to 30 Watts maximum.

Either 10KH or 100K ECL I/O signal levels are compatable with the MCA10000 but both are not supported simultaneously. Therefore both outputs and inputs of the array must either be 10KH ECL or 100K ECL signals. The output pad drivers may be configured as standard 50 or 25 ohm ECL outputs, or they may be configured as series terminated ECL (STECL) outputs which include a constant current source pulldown and a series terminating resistor. This feature allows the elimination of off-chip termination resistors.

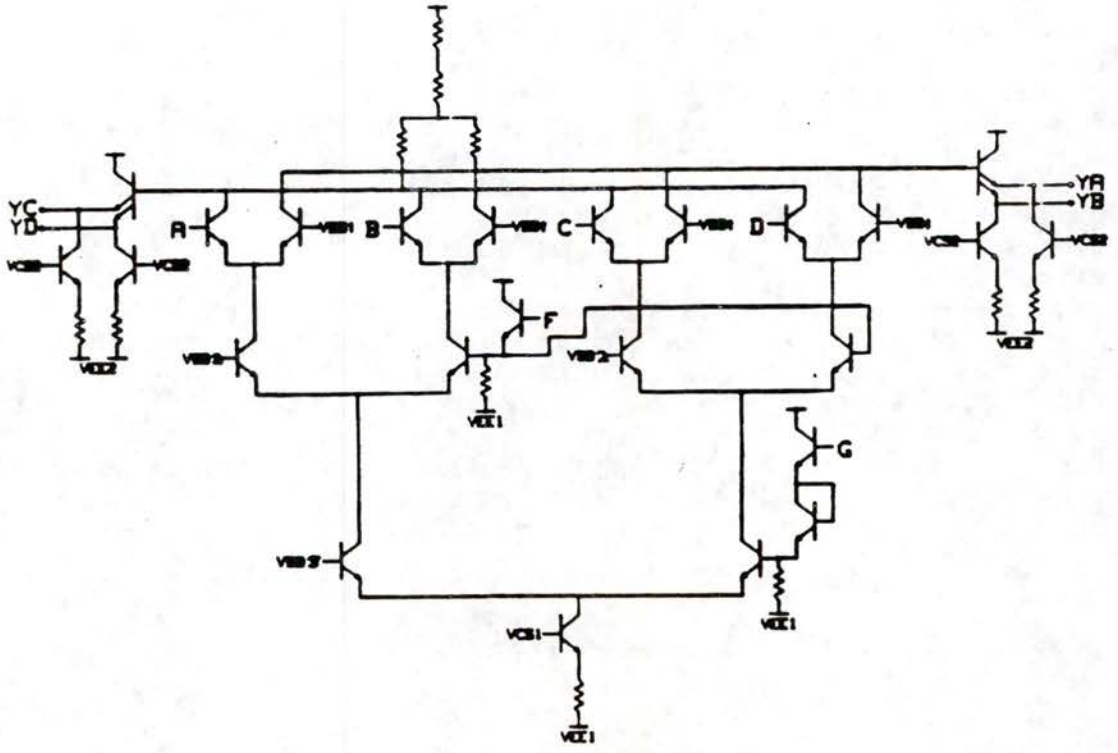
The power, ground, and bias supply lines are not shown in Figure 1. These interconnects are automaticlly accomplished by the CAD system. The illustration shows only the number of free Metal 1 tracks between MCELL columns that can be used by the designer to interconnect the cells in the array. The tracks in the vertical direction are accomplished on Metal 1 while the tracks in the horizontal direction reside on Metal 2. The second layer of metal may be routed over the cell without interfering with the macro in that cell since all macros are intraconnected on the first layer of metal. This is true for all cells and slave bias drivers but not for the bias regulators. Metal 2 routing is not allowed over the bias regulators even though they are also intraconnected on Metal 1 since signal cross-talk can affect their performance. The first layer of metal is separated from the second layer of metal by polyimide dielectric. Connections between Metal 1 and Metal 2 is accomplished with VIA's.

FIGURE 2

MCA2 4 TO 1 MUX
2 LEVEL SERIES GATING



MCA3 4 TO 1 MUX
3 LEVEL SERIES GATING



3.0 MACROCELL ARRAY DESCRIPTION

3.1 Input Cells

The Input or Interface Cells (ICELLS) are located around the periphery of the gate array. In addition to input buffering, the Input Cells provide a small logic capability, ie. a two input OR/NOR gates. ICELLS are required so that proper tracking rates can be provide to the inputs of Major Cells. All signals coming into the gate array are considered EXTERNAL signals and must pass thru an ICELL before the signal can be routed to a MCELL.

NO signal mixing of bonding pad and Major Cell signals is allowed into an Input cell. Only input signals from bonding pads or "Z" outputs of Output Cells can be connected (separately or mixed) to an input of an Input Cell.

Input Cell outputs (as well as Major Cell) can only be connected to inputs of Major Cell and Output Cell macros and are considered INTERNAL signals. These outputs do NOT have enough drive capability to drive signals off the gate array.

Each Input Cell contains 10 transistors and 12 resistors as shown in Figure 3. These components are connected together on first layer of metal to form logic functions with only one level of series-gated structures. All Input macros must fit within one Input Cell.

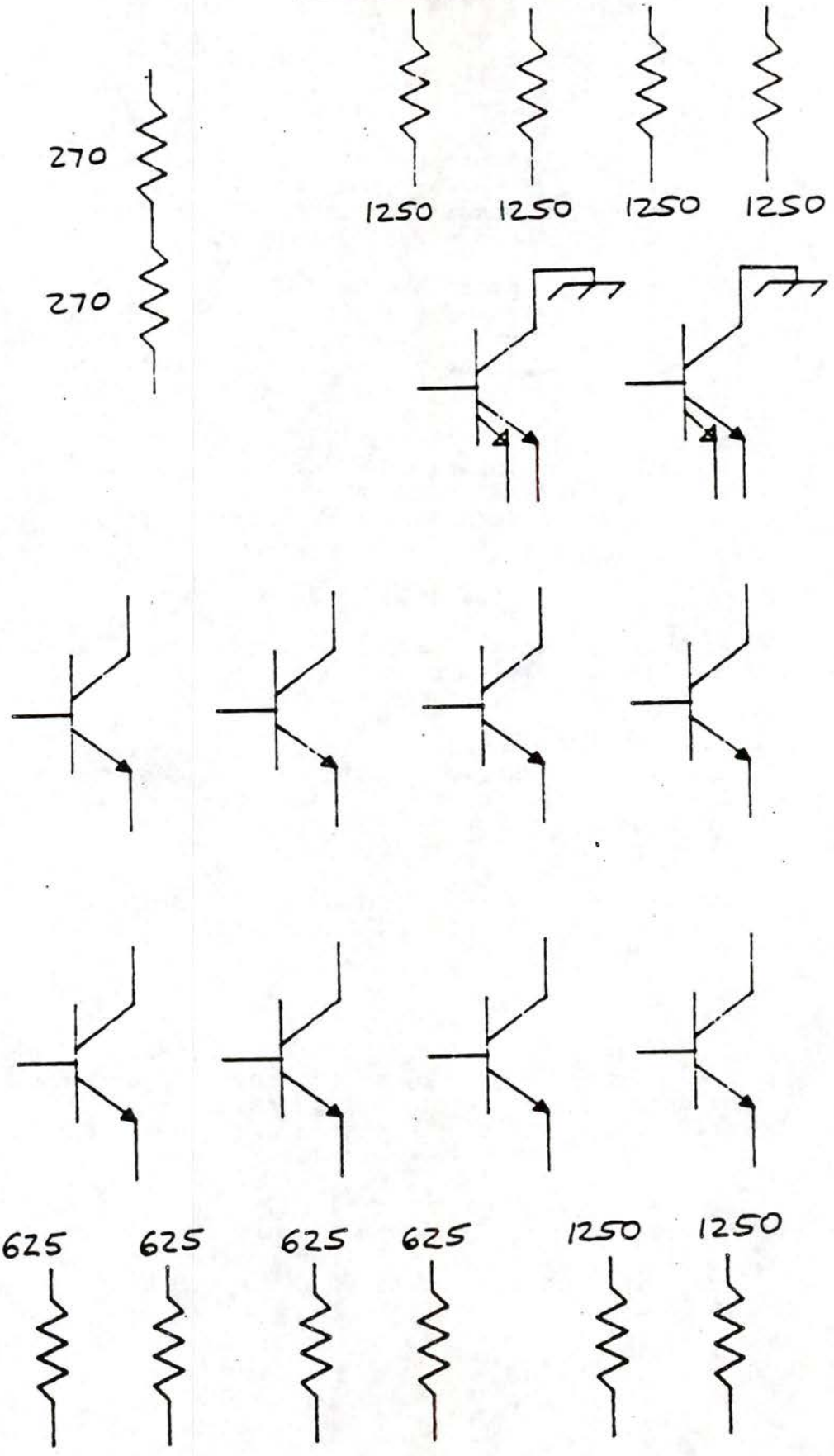
The Input Cell current switch is connected to VEE2 (-3.4 volts) to conserve power on the gate array. Two gate currents, 0.4mA for a low and 0.8mA for a high, are available. Both have 0.8mA output emitter followers (OEF) which are connected to VEE2.

The power dissipation (PD) of an I,M,or OCELL macro does NOT include the output follower current since 0 or 0.8 mA can be specified for each output.

3.2 Major Cells

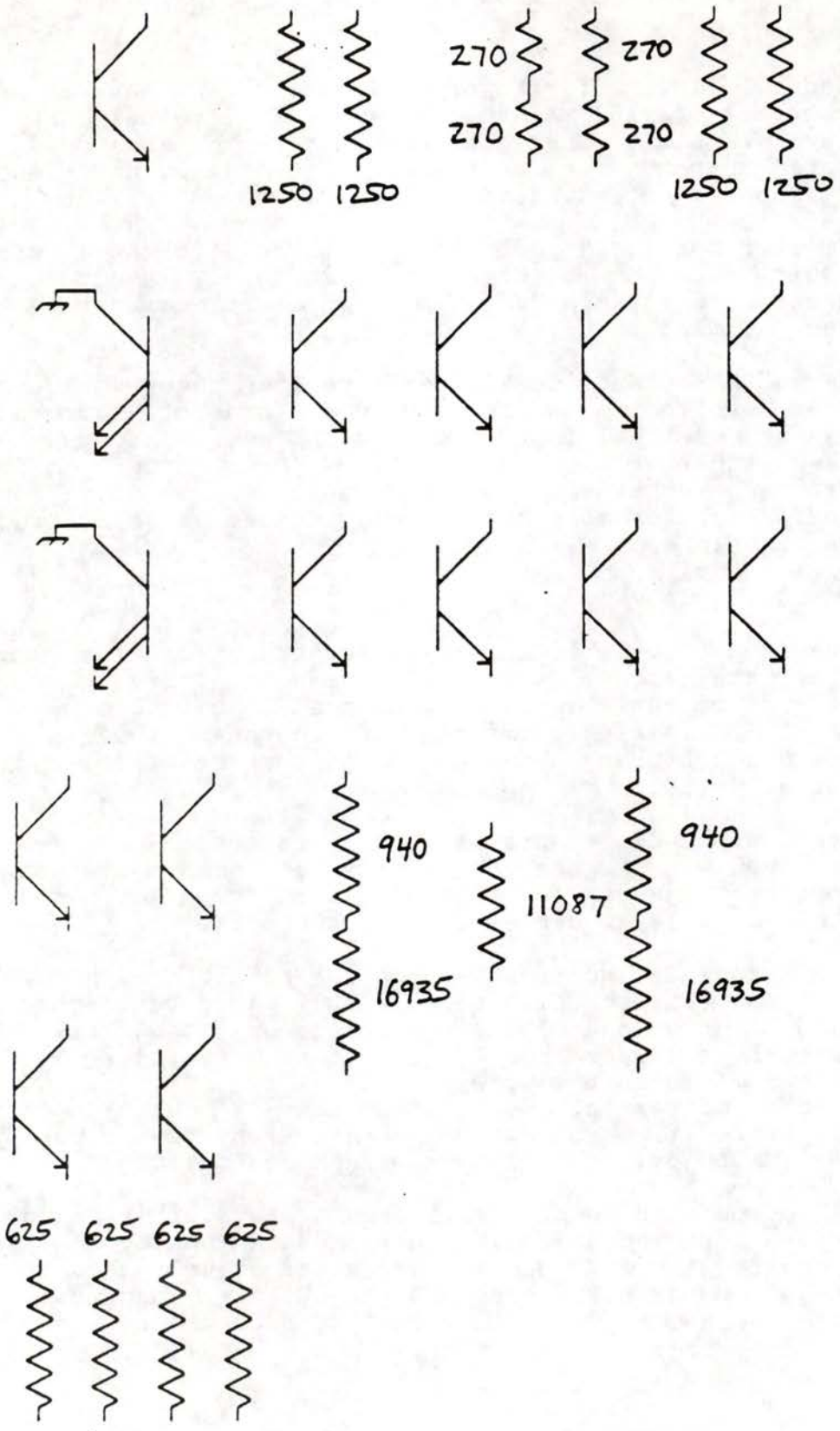
The Major Cells (MCELLS) in the array comprise the internal area on the chip and are used for the majority of the logic. Each Major Cell contains 76 transistors and 82 resistors. These components are connected using Metal 1 to form 1, 2, or 3 level series gated structures. Each macro in the library specifies how much of a cell is required to implement the macro, ie either 1/4, 1/2, 3/4 or 1 Major cell. Figure 4 shows the components in a 1/4 MCELL. One MCELL is the largest discrete unit in the MCA3 gate array, macros larger than one MCELL are not presently available.

FIGURE 3



MCAIII: INPUT CELL SCHEMATIC

FIGURE 4


1250
1250
270
270
270
270
1250
1250
940
16935
11087
940
16935
625
625
625
625
1250
1250

Outputs of a Major Cell macro are considered INTERNAL signals and can only drive inputs of other Major cells and inputs to Output Cells. The current source for each macro is either 0.4mA or 0.8mA for a Low and High power macro respectively.

Input followers for Major Cell macros each have an emitter current of .167 mA or .208 mA (nominal values for 2 and 3 level inputs). These currents are summed and added to the gate current for each macro to determine the power dissipation of the gate.

Emitter dotting of output transistors forms a wired-OR logic function internal to a macro. This occurence for each macro is recorded as a pin property parameter within the CAD data bases.

YA and YB outputs in Figure 2 are low power outputs and are referred to as "twin outputs" to denote macro outputs having identical logic functions. These outputs can be tied together (called a high power output) to improve drive characteristics or used separately to drive separate loads (one or both outputs may be used in generating wire OR functions).

3.3 Output Cells

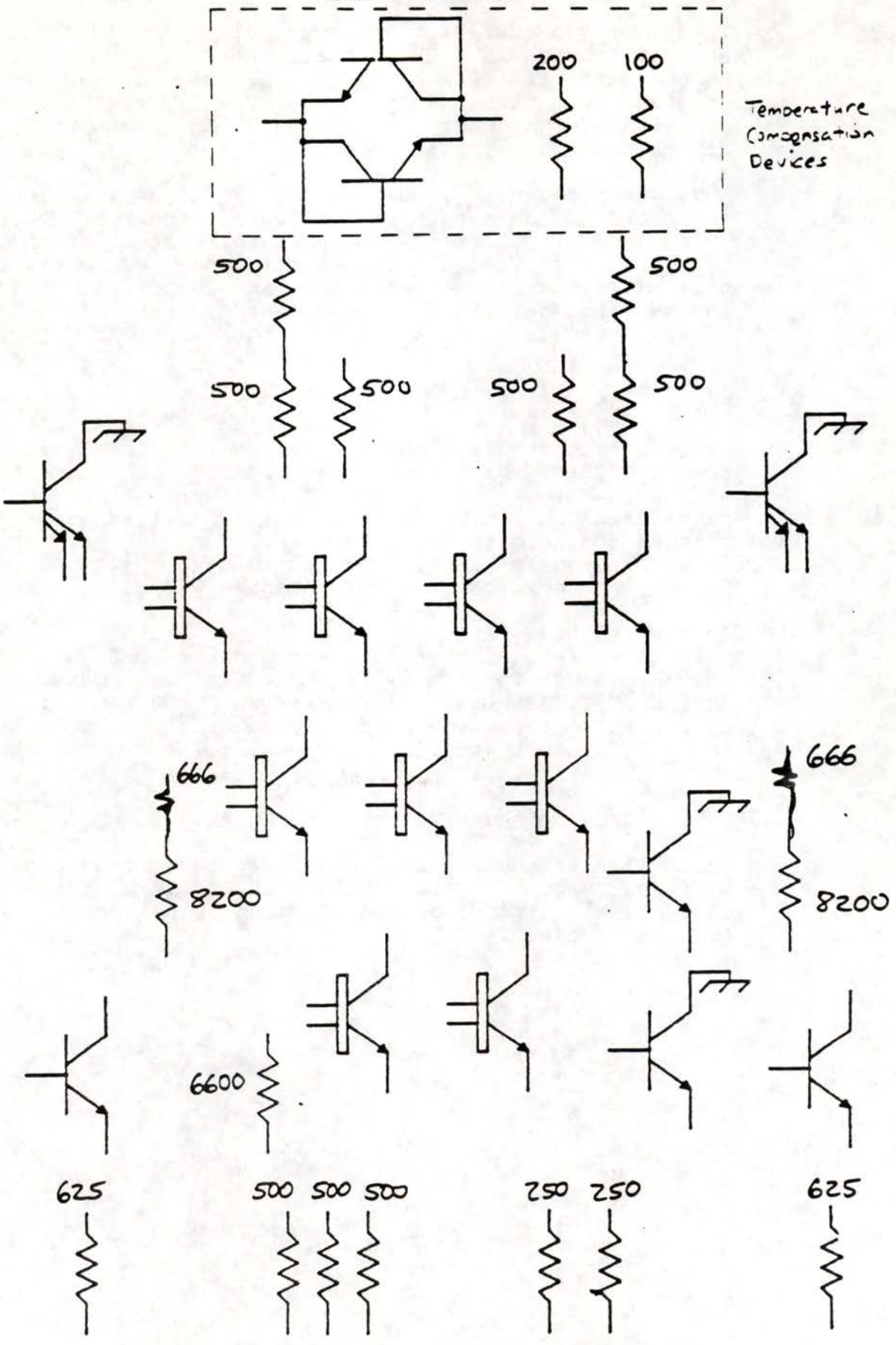
The Output Cells (OCELLS) are located on all four sides of the gate array. They are used to interface to logic outside of the chip by providing a 25 ohm, 50 ohm, 60 ohm or STECL (series terminated ECL) output drive capability. The OCELL Library provides macros that can fit into 1 or 2 Output Cells.

Each Output Cell contains 17 transistors and 20 resistors as show in Figure 5. These components are connected together on Metal 1 to form logic functions consisting of 1, 2, or 3 level series gated structures.

The Output Cell current source can be either 2 mA, 3 mA, or 6 mA. Output Cell macros with 2 mA current sources are used to drive one STECL Pad Cell or a 60 ohm output emitter follower per "X" output pin. Output Cell macros with 3 mA current sources can drive two STECL pads or one 50 ohm emitter follower off the same "X" output pin in the Output Cell. OCELLs with a 6 mA current source can drive one 25 ohm emitter follower per "X" output pin.

"Z" outputs on Output Cells pins are available for driving inputs of Input Cells only and are considered EXTERNAL signals. The "Z" outputs have a larger voltage swing than an ICELL or MCELL output since a larger load resistor is used in the Output Cell. The "Z" output current source can be either 0.0 or 0.8 mA.

FIGURE 5



MCAIII: OUTPUT CELL SCHEMATIC

The primary output of the Output Cell macro is labeled "X" and is considered a CML signal. When it is used, it must be connected to the base of an output emitter follower located near the bonding pads. The "X" output cannot be connected to any macro input. Both "X" and "Z" outputs can be utilized simultaneously.

A network (shown in a box with dotted lines in Figure 5) is added to the output cell macros if the ECL 100K option is selected to alter the temperature tracking of the OCELL output. When the network is not connected in the Output cell, it's output becomes a 10KH output. Note: ALL inputs and outputs on the gate array must be EITHER 100K or 10KH signals. Combination of the two signal types is not allowed.

3.4 Pad Cells

Cells are located on the periphery of the gate array called PCELLS which contain the components needed to form 25 ohm, 50 ohm, 60 ohm, and STECL drivers. The PCELL also contains a 75K ohm resistor which is required on all input signals coming onto the gate array. The 75K ohm resistor is connected to VEE2 (-3.4) and is used to form a logical LOW on inputs which are left unconnected.

Output signal levels are generated at the PCELL. Therefore the minimun path through the MCA3 is to access the chip at an input pad (EXTERNAL signal), pass through an ICELL (INTERNAL signal), enter an OCELL and exit at one of its "X" outputs (CML signal), connect to a PCELL to form an output (STECL, 50 ohm, ...), and attach to an output bonding pad.

3.5 Power, Ground, and Reserved Bonding Pads

Table 1 shows the pre-assigned power and ground pins that will be used on every MCA10000ECL gate array. It also includes a list of reserved bonding pins used for the Aquarius project.

TABLE 1

MCA Power Pin List

	Top	Right Side	Bottom	Left Side
VCC	9,10	95, 96	189,190	275,276
	25,26	103,104	205,206	283,284
	39,40	115,116	219,220	295,296
	51,52	121,122	231,232	301,302
	65,66	131,132	245,246	311,312
	81,82	139,140	261,262	319,320
		149,150		329,330
		155,156		335,336
		167,168		347,348
		175,176		355,356
VEE1	47,48	109,110	227,228	289,290
		127,128		307,308
		143,144		323,324
		161,162		341,342
VEE2	43,44	107,108	223,224	287,288
		124,125		304,305
		146,147		326,327
		163,164		343,344

4.0 LOGIC DESIGN CONSIDERATIONS

4.1 Bonding Pads Connected to Macro Inputs

Any non-power bonding pad may be used as an input pad. This input may connect to any Input Cell input pin but may not be routed to any Major or Output Cell input pin directly.

4.2 Input Bonding Pad Signal Integrity Rules

The following are Signal Integrity input pad rules:

A) Maximum of four Input Cell loads per input bonding pad.

B) Maximum total length of on chip metal for routing to ICELLS = 50 mils (first metal + second metal). 50 mils of worst case metal = .4pf.

C) Differential inputs may lie on adjacent input pins or on input pins separated by one output pin. The metal length connecting each input pin of the pair to the Input Cell load must be within 20% of each other.

4.3 Internal Connections

Outputs of Input and Major Cells macros may only be connected to inputs of Major Cells or Output Cells. For Output Cell macros, the "Z" output may only be connected to inputs of Input cells and the "X" output may only be connected to an output emitter follower in a Pad Cell.

4.4 Output Cell "X" Outputs Connected to Bonding Pads

The only cell output that is allowed to connect to a bonding pad is an Output Cell "X" output. Input and Major Cell macro outputs are not allowed to connect to bonding pads. "Z" outputs of an Output Cell macro are also not allowed since these may only be used to drive Input Cell inputs.

4.5 Output Bonding Pad Signal Integrity Rules

The following are Signal Integrity output pad rules:

A) Only one pad can be utilized as a source for off chip nets when a STECL output is used (NO STECL off chip wire ORs allowed).

B) All STECL outputs must have a series resistor connected to the output pad. A series resistor is provided in the PCELL for this purpose.

C) Only one OCELL "X" output is available per output

bonding pad (NO wire ORing of OCELL "X" outputs).

D) Up to 50% of the I/O pads (128 pads) may be single ended outputs spread evenly between Vcc pins on the chip. Single ended outputs may not lie on adjacent pins.

E) Complementary (differential) output pairs may exceed the 128 output limit up to a maximum of 140 outputs (six additional outputs can be added to the total number as long as they are added as differential pairs). These output pairs may lie on adjacent pins or be separated by one single ended pin. This means that outputs can only be on adjacent pins if the second signal is a simultaneous inverted copy.

F) Differential output pairs must be fully utilized off chip, one output is not allowed to remain floating while the other output is used.

G) Complementary outputs must originate from the same Output Cell. The metal length connecting each output of the pair to its pad must be within 20% of each other.

4.6 Unused Inputs or Outputs Internal to the Gate Array

Any unused outputs in the array are allowed to be left floating unconnected. Unused inputs need to be connected to a low voltage, logic "0" (the input base is shorted to the emitter). This provision should be supplied by a strapping process resident on the user's CAD system. There is no special provisions for providing a high voltage (logical "1") on an unused input, it must be generated. This can be accomplished by using a spare inverter from a major cell.

Outputs need to be strapped if any loads are connected to the output of a macro except for "X" outputs in Output Cells which do NOT require straps. A CAD tool will place straps on outputs which are used. The only exception to this rule is when wire OR functions are preformed by tying outputs of different gates together. In this case, not all outputs need to be strapped, only the ones which furnish the output current source.

4.7 Maximum Fan-Out of Macro Outputs

The maximum DC fan-out for a low power output should be limited to a load factor of ??? on all internal outputs. For high power outputs (twin outputs having the identical logic function), the maximum DC load factor is ???. Values TBD.

4.8 AC Fan-out of Macro Outputs

- TBD -

4.9 Wire ORing

Wire ORing is a circuit technique used to provide an OR function of a number of signals without needing to utilize an OR gate thereby saving a gate delay and the gate's power. This section outlines the interconnect and loading rules associated with wire ORing. Section 5.3 contains delay information associated with wire ORing and section 6.1 includes placement and routing rules.

When a wire OR function is performed, the output follower should be selected such that for each emitter output the maximum current is 0.8 mA per output (when only 1 output is high) and the minimum current is 0.2 mA (when all the outputs of the wire OR are high). Therefore the total number of wire OR's allowed per wired OR function is 4, which must include the macro internal wire OR's. The number of internal wire OR's for each macro output is given in the Macrocell library for the gate array. The true and complement outputs of a macro can be wired ORed separately but can not exceed the 4 wire OR limit stated above. The number of internal wire OR's for the true and complement outputs of a macro are not necessarily the same value.

"Twin outputs" (outputs from the same macro having identical logic functions) can be tied together for improved drive capability but do NOT count in the wire OR total even though a logic wire OR function results. A 1.6mA output follower current (high power output) can be used for a wire OR function ONLY IF these "twin outputs" are both used in ALL of the source outputs. In other words, if a 1.6mA output follower current is desired for drive capability, the twin outputs of each wire ORed source must be connected. The reason for requiring them both to be connected is to adhere to the 0.8mA maximum output current density rule. The maximum number of high power wire ORed sources is still limited to four.

Low and high power output emitter followers can NOT be wire OR'ed together.

Outputs of Major Cells and Input Cells can be tied together to form a wire OR function. Wire ORing of "Z" Output Cell outputs is NOT ALLOWED. Wire ORing of "X" Output Cell outputs is also NOT ALLOWED.

Wire ORing of STECL, 60 ohm, 50 ohm, and 25 ohm outputs is NOT ALLOWED on or off the gate array.

4.10 Designing Latches with Gates

When designing logic requiring latches, the logic designer

is NOT ALLOWED to make his own latches from non-latch macros.

4.11 Calculation of Maximum Power Dissipation

The maximum power dissipation of a gate array option can be calculated following the guidelines of this section. Table 2 contains a breakdown of all the possible power sources on the gate array and includes a description of the variables used in this section. The total power from these sources may be calculated and totaled under the appropriate supply voltage category, either Pveel or Pvee2. A maximum power can then be determined. Note: neglect input pulldown power.

TABLE 2

MCA3 Power Dissipation Sources

Structure	Power	
	VEE1 (-5.2 V)	VEE2 (-3.4 V)
ICELL		cell(Pci), OEF pwr(Poi)
MCELL	cell power (Pcm)	OEF power (Pom)
OCELL	cell power (Pco)	"Z" OEF pwr (Poz)
STECL Output		"X" OEF pwr + SL5 (Pox)
Regulators	VREF1,VREF3 (Pr1)	VREF2,VREF5 (Pr2)
Slaves	SL11,SL12,SL3 (Ps1)	SL21,SL22 (Ps2)

- o all cell power is on a per gate used basis
- o all output emitter follower (OEF) power is on a per output used basis
- o regulator and slave power is fixed on all MCA3 options
- o STECL output power is the sum of the "X" OEF and SL5 slave power. A SL5 slave is only needed when its corresponding STECL output is utilized.

Pci - ICELL gate power per individual ICELL
Pcm - MCELL gate power per individual MCELL
Pco - OCELL gate power per individual OCELL
Poi - ICELL OEF power per output
Pom - MCELL OEF power per output
Poz - OCELL OEF power per "Z" output
Pox - STECL OEF power per "X" output including SL5 (slave reference generator) power
Pr1 - total VREF1 and VREF3 regulator power
Pr2 - total VREF2 and VREF5 regulator power
Ps1 - total SL11,SL12,SL3 slave reference generator power
Ps2 - total SL21,SL22 slave reference generator power

A) The constants needed to determine the gate array power are given below. They are all nominal power values.

Pci, Pcm, Pco : vary with each macro (refer to the cell library for macro gate power)
Poi = Pom = Poz = 2.72mW nominal (per 0.8mA OEF)
Pox = 36.72mW (per STECL output incl. SL5 pwr)
Pr1 = 96.30mW (total regulator VEE1 power)
Pr2 = 164.29mW (total regulator VEE2 power)
Ps1 = 1396.72mW (total slave VEE1 power)
Ps2 = 522.24mW (total slave VEE2 power)

B) Calculate Pc1, the total cell power from the VEE1 supply and Pc2, the total cell power from the VEE2 supply.

Pc1 = ΣPcm + ΣPco :total MCELL + OCELL gate pwr
Pc2 = ΣPci :total ICELL gate power

C) Calculate Po2, the total output emitter follower (OEF) power from the VEE2 power supply.

Po2 = ΣPoi + ΣPom + ΣPoz :total ICELL,MCELL, and OCELL "Z" OEF power

D) Calculate Pox2 which is comprised of one of the following: the total power dissipation due to STECL outputs (added to the VEE2 power numbers, -3.4 volts) which includes the SL5 power; 60 ohm (-2.0 volts) outputs; 50 ohm (-2.0 Volts); or 25 ohm (-2.0 volts).

Pox2 = ΣPox :total OCELL "Z" OEF power

E) Add the power dissipation overhead for the voltage regulators and the slave reference generators.

Pv1 = Pr1 + Ps1 :total Reg + Slave VEE1 power
Pv2 = Pr2 + Ps2 :total Reg + Slave VEE2 power

F) Add the nominal power values together for VEE1 and VEE2 and multiply this total by 1.25 to obtain the maximum total power for the option. The total maximum power dissipation for each gate array option shall not exceed 30 watts.

Pvee1 = Pc1 + Pv1
Pvee2 = Pc2 + Po2 + Pv2 + Pox2

Maximum total power = 1.25 * (Pvee1 + Pvee2)

4.12 Power Sequencing

When powering the MCA3 up or down it is very important that the following power sequencing is followed:

On Power Up : apply the VEE1 supply then apply the VEE2 supply in that order
On Power Down : turn off the VEE2 supply then turn off the VEE1 supply in that order

Note: the VEE2 supply should NEVER be more negative than the VEE1 supply.

5.0 PERFORMANCE

5.1 Input Current

The maximum input current (at Vih max) at the input pin leading to a high power Input cell can be calculated by multipling: 14.3uA times the total fan-in plus 70uA for the pulldown resistor between the input pin and VEE2. The maximum input current (at Vih max) at the input pin leading to a low power ICELL can be calculated in the same way by multipling: 7.1uA times the total fan-in plus 70uA for the pulldown resistor between the input pin and VEE2.

5.2 Propagation Delay Degradation of an Input Due to Rise Time Skew at the I/O pin

When external inputs drive an input pin of the gate array, the rise and fall times should be 0.5ns (20% to 80%) in order for the progation delay of the Input Cells specified in the Macrocell Library to be accurate. Although the driving source may provide a signal of 0.5 ns rise and fall times, the capacitance at the end of the tranmission line will degrade the signal. For each nanosecond increase above .5 ns in the rise and fall time at the input pin of the gate array, a value of delay degradation should be added to the Input Cell macro. This delay can be found in Table XXX for both rising and falling signals. This assumes that the 50% point of the input signal is not skewed by more than +/- 20 mV of nominal Vbb = -1.31 volts.

TABLE XXX

- TBD -

5.3 Wire ORing

This section contains delay information associated with wire ORing. See section 4.9 for wire OR interconnect and loading rules and section 6.1 for placement and routing rules.

Wire ORing of macro outputs will lead to some delay degradation which should be added to the propagation delay

specified in the Macrocell library for the gate array. The maximum propagation delay degradation for the output edge falling (delta Tpd-) is:

Low OEF: delta Tpd- = 68+(# Wire OR's - 2)*9 (pS)

High OEF: delta Tpd- = 24+(# Wire OR's - 2)*6 (pS)

These degradations are fairly small since the delay is due mainly to a very small capacitance associated with the additional output device on the line. The propagation delay for the output rising edge, delta Tpd+, is much larger as shown in Table 3. This can be attributed to a phenomenon called "current hogging." Current hogging occurs when two macro outputs are wire OR'ed and both are in the "low" state, one output could be suppling more current to the load resistors than the other. When the output that is supplying the smallest amount of current switches to the "high" state, it must now supply all of the current resulting in the additional delay shown in Table 3.

The high OEF should only be used if the "twin outputs" of EACH logic function are connected in the wire OR. As mentioned earlier, twin outputs can be tied together for improved drive capability.

TABLE 3

Maximum Delay Degradation for Output Rising, Delta Tpd+, vs. OEF and # of Wire OR's for Low and High Power Gate/Load Combinations

OEF Type	delta Tpd+ (pS) Total # of Wire OR's		
	2	3	4
Low *	125	141	160
High **	113	148	150

* The total number of outputs in a net is limited to 4 including the number of emitters wire OR'ed for that output.

** If 1.6mA is used for a net then every output in the net must have the twin emitters wired together.

5.4 Metal and Fan-out Degradation

The metal interconnect between macros in the array represents a line capacitance to the driving gate. This line capacitance coupled with the input capacitance of the loads primarily affects the fall time but also can affect the risetime. Worst case metal and fan-out curves are shown in Figures 6-9. Figures 6 and 7 include the risetime and falltime delays for a low power output emitter follower (0.8mA) respectively. Figures 8 and 9 include the same for a high power OEF (1.6mA).

An IR voltage drop value of 15mV is being allowed on signal runs originating from any source gate to any load on the MCA3. To calculate this, each net should be dissected into metal segments pertaining to the amount of load current passing through that section. Having this, an incremental IR value may be calculated which can then be used to determine the total IR voltage drop to any load on that net.

6.0 CAD LAYOUT CONSIDERATIONS

6.1 Wire ORing

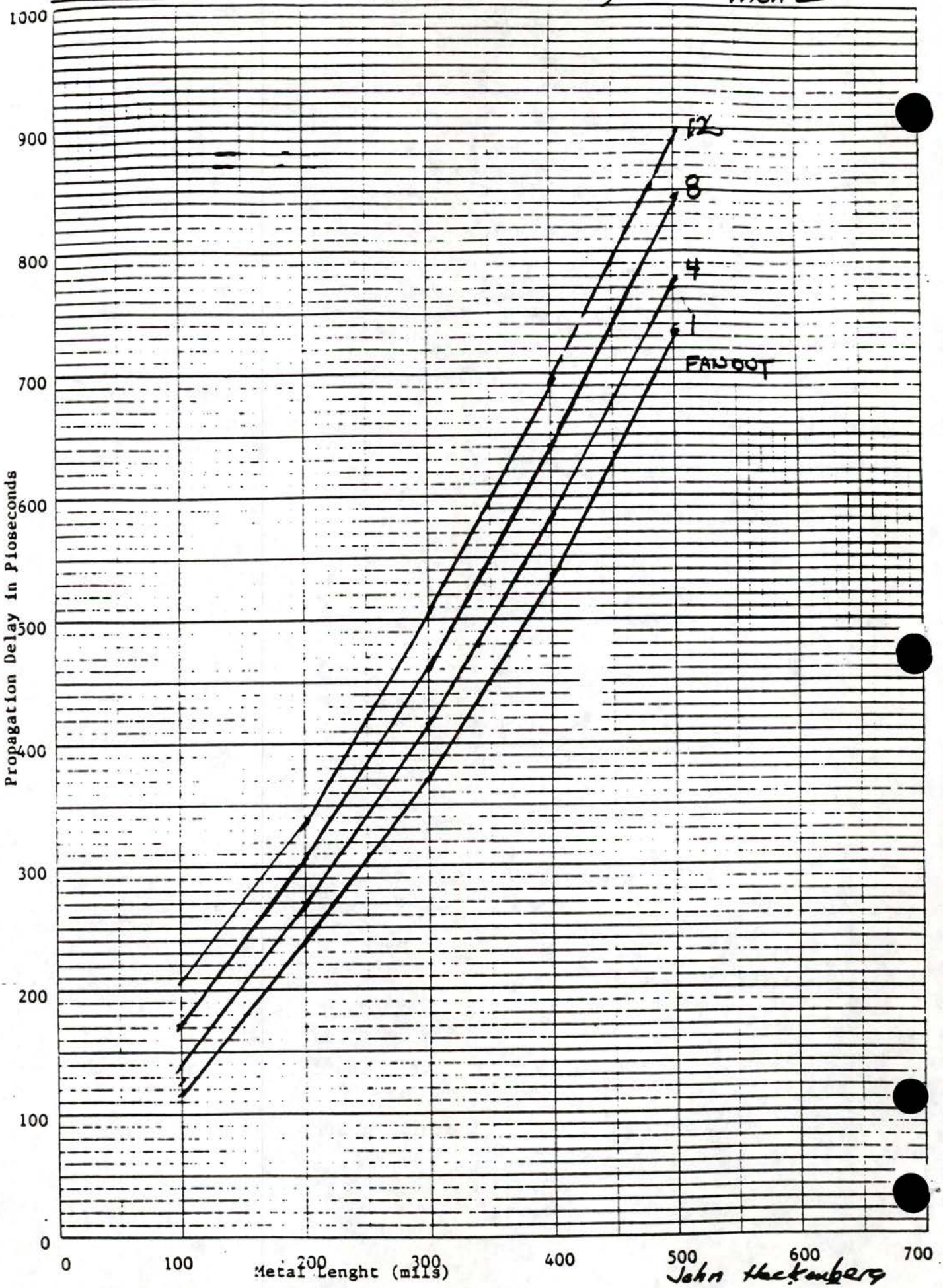
This section contains placement and routing information associated with wire ORing. See section 4.9 for wire OR interconnect and loading rules and section 5.3 for delay information.

When outputs are tied together, current will flow from the output that is "high" to the OCS (output current source) located at one of the other outputs. This current causes a small IR drop, which decreases the signal's noise margin. The maximum IR drop allowed due to wire ORing a net from any source to any load is 20 mV. The resistance of Metal 1 is .46 ohms/mil while the resistance of Metal 2 is .19 ohms/mil. For instance, assume there is a metal run of 100 mils total between two sources wire ORed together where 50 mils is on first layer and 50 mils is on second layer. A 0.8 mA output follower current would mean that 0.8 mA of current would flow from the output that is "high" to the other output when in the "low" state. This could cause a voltage drop of 18.4 mV drop on first metal and 7.6mV drop on second metal for a total drop of 26 millivolts. Note: 0.8 mA is for low power OCS outputs, high power OCS is 1.6 mA. The above example exceeded the 20 mV allowed and illustrates that shorter metal is required to meet the 20mV restriction.

The following wire OR rules should be followed in order to adhere to this 20mV restriction:

A) Loads must originate from a common point "load point"

FIGURE 6 ~~of~~ = .8mA (Rise Time) MCA III

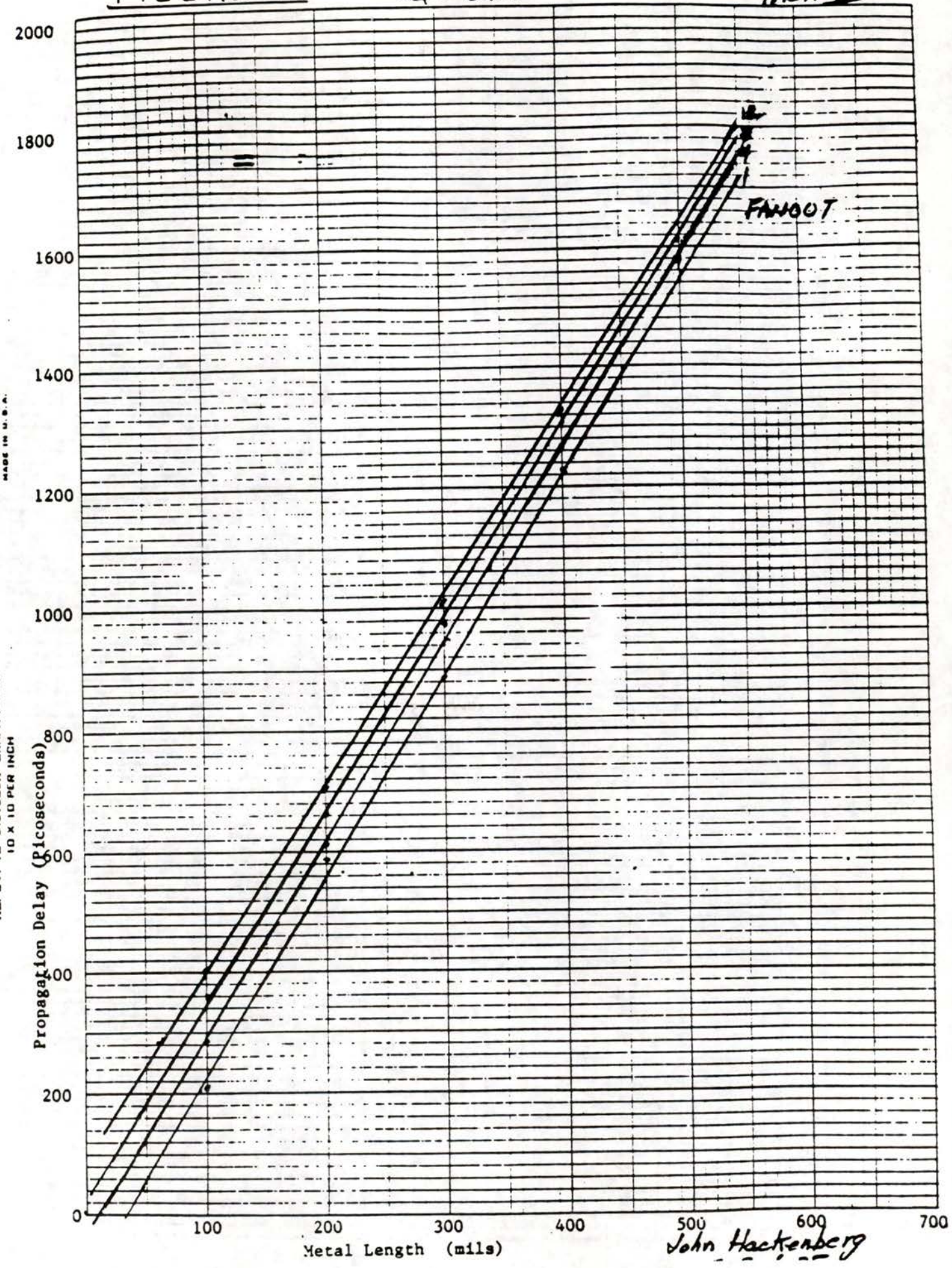


John Hackenberg

FIGURE 7

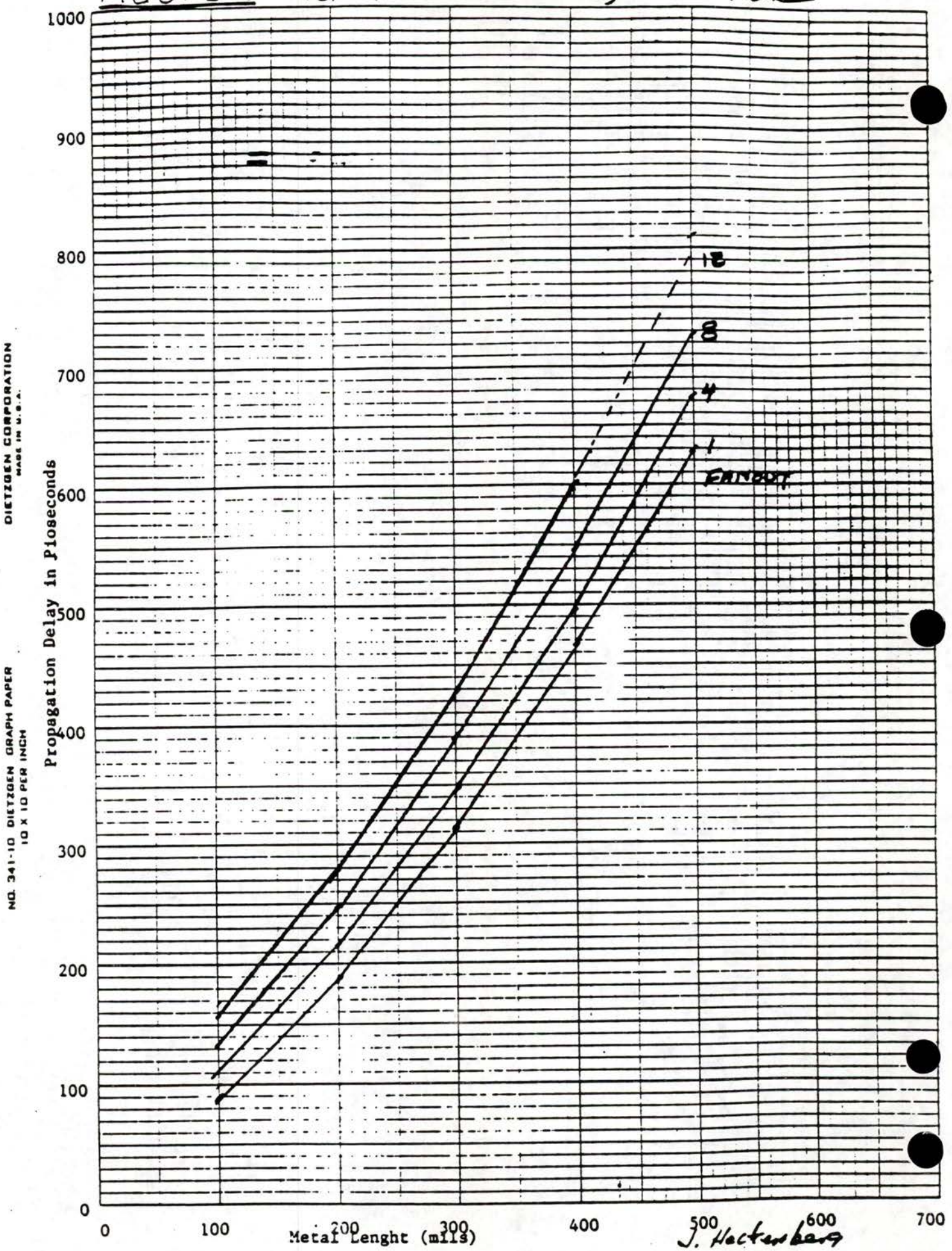
I_{CL} = .8 mA (Falltime)

MCA III



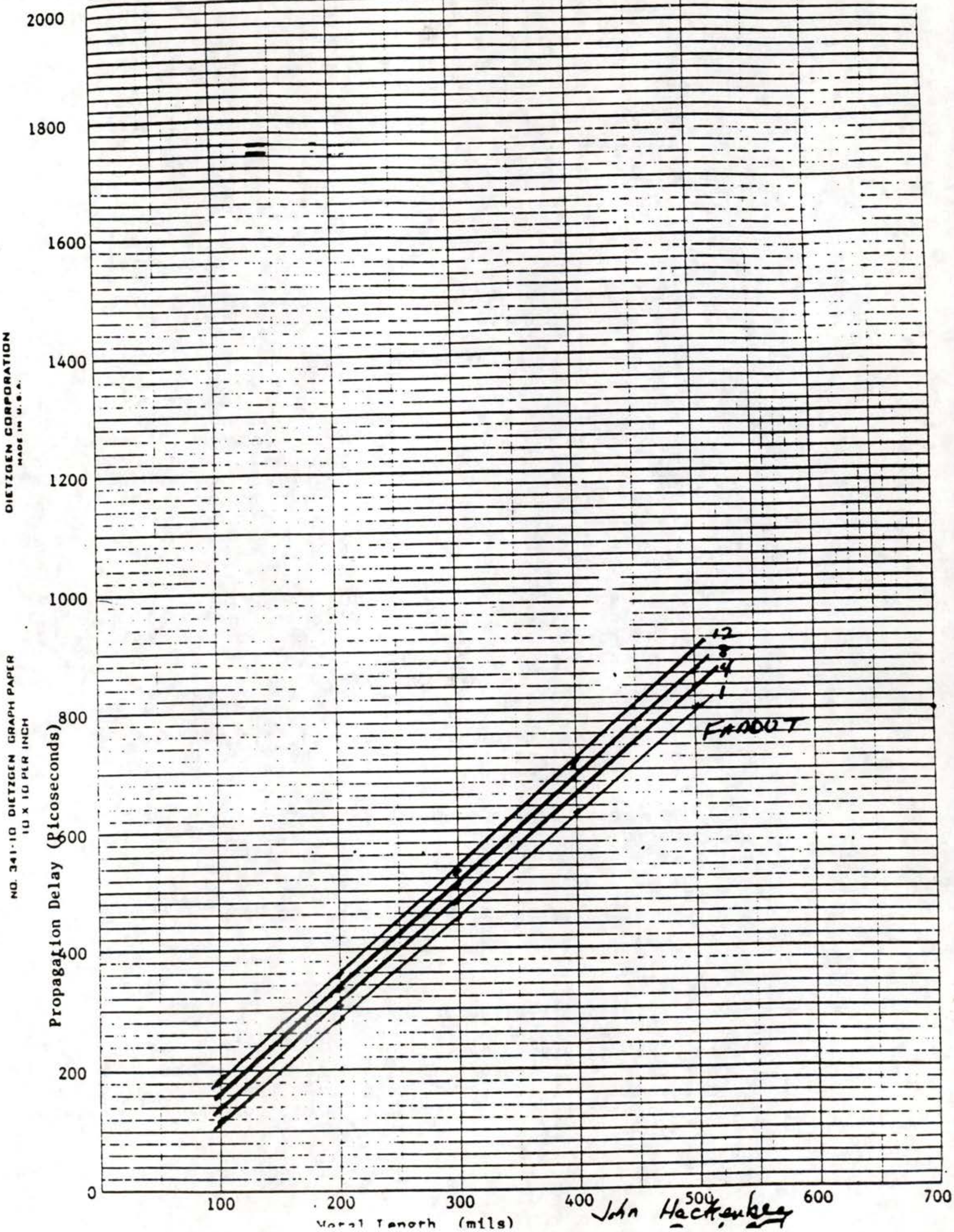
John Hackenberg

FIGURE 8 Iof = 1.6 ma (Risetime) MCA III



J. Hectenberg

FIGURE 9 $I_{of} = 1.6$ mA (Falltime) MCA III



John Hackenberg

in the wired-or net connecting the sources. See Figure XXX for an example of what is ment by a "load point" or a wire ORed net.

B) Place current source(s) at farthest source from the load point.

C) Maximum metal length between the load point and the farthest source from the OEF is:

Low pwr OEF: <= 25 ohms (Metal 1 plus Metal 2)
High pwr OEF: <= 12.5 ohms (Metal 1 plus Metal 2)

The metal length resistance in ohms is given by the following equation.

R = (.46 ohms/mil) * (Metal 1 length in mils) +
(.19 ohms/mil) * (Metal 2 length in mils)

D) Wire ORing takes the second hightest priority of to-be-routed nets (just after routing clock nets). Minimum metal length should be used to connect the sources. To accomodate this, placement of wire ORed nets takes the second highest priority of to-be-placed macros.

6.2 VIA Placement

In the routing channels, the only restriction on VIA's (connection between first and second layer metal) is that adjacent horizontal VIA's which are on two different signal metals are NOT allowed.

6.3 Output Metal Length Restrictions

A) The maximum metal length allowed from the "X" output of an Output Cell macro to the Pad Cell is 100 mils for STECL outputs.

B) The metal length from the "Z" output of an Output Cell macro to an input of an Input Cell must be restricted to a maximum of 50 mils total metal length.

6.4 Placement Rules Due to Power Balancing Requirements

- TBD -

6.5 Cell Reference Designator Scheme

The MCELL reference designators are assigned using a 5 character representation. The first character is "E" followed by two characters describing the X location and two characters describing the Y location (Exxyy) of each

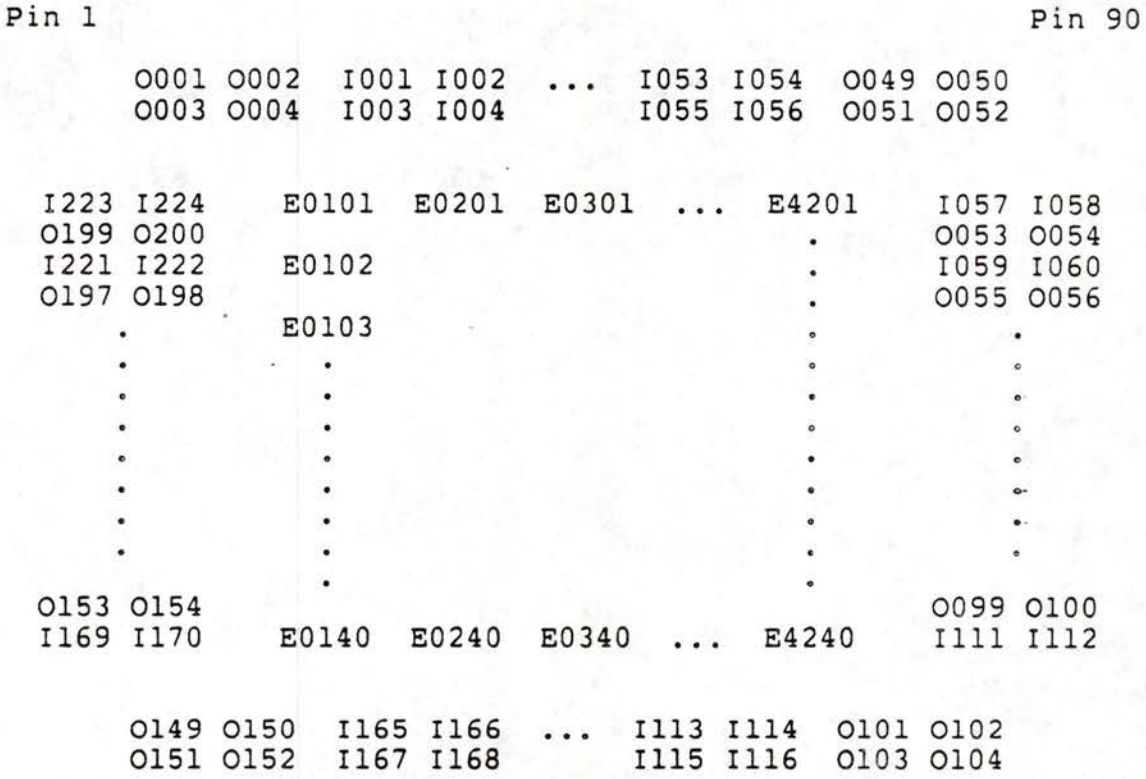
Quarter-MCELL. Starting with pin one of the MCA3 in the upper left hand corner, the Quarter-MCELL in that corner is labled E0101. Moving to the right across the array the X values increase; E0201, E0301, E0401, ... E4201 (there are 42 columns of Quarter-McellS in the MCA3). Moving down the rows the Y values increase; E0102, E0103, E0104, ... E0140 (there are 40 rows of Quarter-MCELLS in the array). See Figure 10.

The ICELLS reference designators follow a different approach. Their designators begin with an "I" and are followed by a 3 character digit representing the number of ICELLS on the array (Iaaa). There are 224 ICELLS on the MCA3 so the designators begin by pin one with I001 and end with I224. Because describing their ordering method would be difficult, Figure 10 is used to convey this method.

The OCELLS reference designators follow the same approach as the ICELLS. Their designators begin with an "O" and are followed by a 3 character digit representing the number of OCELLS on the array (Obbb). There are 200 OCELLS on the MCA3 so the designators begin by pin one with O001 and end with O200. Because describing their ordering method would also be difficult, Figure 10 is again used to convey this method.

Figure 10

MCA3 Reference Designator Locations



E STRAM Specification

T. B. D.

F Clock Specification

Aquarius Clock System Specification

Revision 1.1

26 August 1986

Author: Paul Guglielmi

CONTENTS

1 OVERVIEW

An overview drawing of the Aquarius Clock System is shown in figure 1. The diagram will be the same for both the dual and quad processor Aquarius systems. The only difference between the two configurations is the need for more copies of the Master Clock modules outputs in the quad processor system.

Clocks will originate from an oscillator that generates a sine wave clock. The output of the oscillator is sent to a power amplifier and then to a power divider and Clock fanout network from which copies of the Clock are distributed to the Clock Distribution Chips on the MCUs and modules of the system.

The Clock Reference signal which is a square wave signal that is derived from the Clock, will originate from the Clock Oscillator and be fanned out and distributed to the Clock Distribution chips in the same manner as the Clock. This signal runs at 1/8 the frequency of the Clock Oscillator and is used to define the beginning of a machine cycle.

The Clock Control signal is derived from the Clock Reference signal and is distributed to the Clock Distribution chips. This is a logic signal that is controlled by the console. The Clock and Reference signals that free run as long as the system has power. The Clock Distribution Chip and gate arrays will generate one machine cycles worth of clocks for each Clock Control pulse that the Clock Distribution chip receives.

AQUARIUS CLOCK SYSTEM

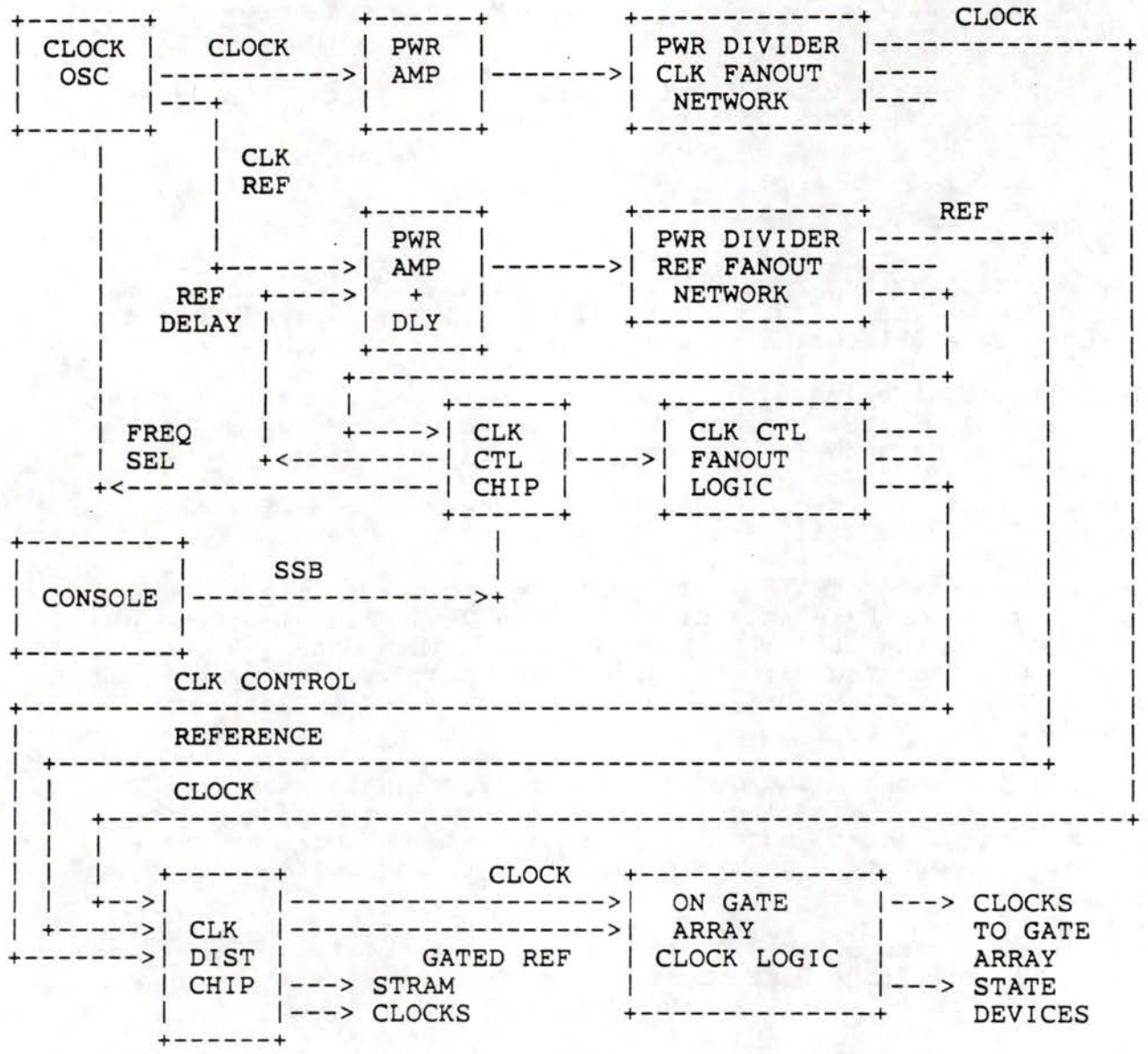


Figure 1

The Clock Control chip has logic on it for gating the Clock Control signals on and off, generating a burst of Clock Control pulses that is a prescribed number of machine cycles in duration, controlling the spacing between Clock Control pulses in machine cycles, specifying the position of the Clock Reference relative to the Clock, specifying the frequency at which the Clock Oscillator is to run, and a frequency counter for measuring the Clock Oscillators output frequency. Commands to this chip will come from a Aquarius Scan System which interfaces to the Console.

The Clock Distribution Chip performs several functions. It fans out the Clock and Gated Reference signals to the Gate Arrays, shapes and fans out the Clocks for the STRAMS, and acts as a central distribution point for the Aquarius Scan System on an MCU.

The on Gate Array Clock logic shapes and fans out the clocks that are used by the state devices on the Gate Array.

1.1 Clock Waveforms

The Aquarius system requires a Clock with 8 Clock edges per machine cycle. The eight Clock edges will be derived from the rising edges of eight cycles of the Clock Oscillator.

To define the beginning of a machine cycle the clock oscillator will provide a signal which runs at 1/8 the frequency of the oscillator. This signal is known as the Clock Reference.

The Clock Control signal determines whether clocks for the current machine cycle will be generated.

The Gated Reference signal is generated from the Clock, Clock Control, and Clock Reference signal in the Clock Distribution Chip and fanned out to the Gate Arrays on an MCU. The Gated Reference will only be asserted if the Clock Control signal was true indicating that the gate array should generate a cycles worth of TA and TB clocks.

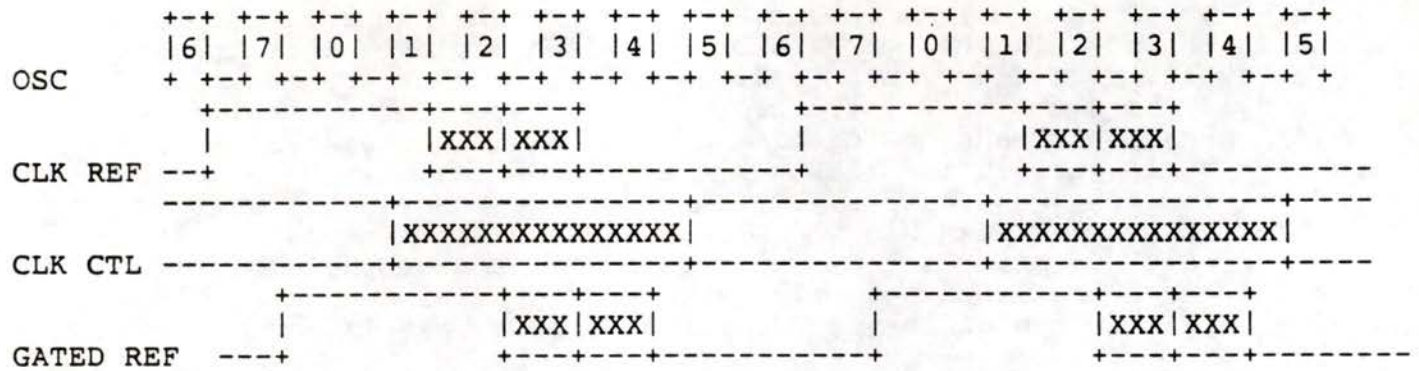
The Clock Reference and Gated Reference signals may not be perfect square waves. The logic that uses these signals should only depend on the leading edge of the signals being accurately positioned. The pulses High state may be from 3/8 to 5/8 of a cycle wide.

A cycle worth of STRAM T0 through T7 clocks will only be generated by the Clock Distribution chip for the STRAMs on the MCU if the Clock Control signal is true.

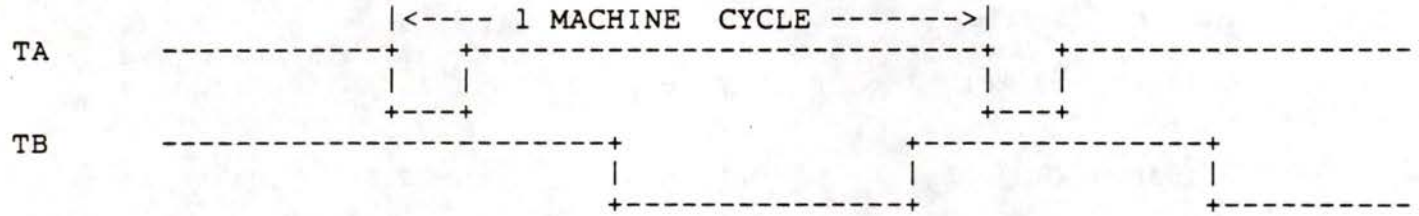
The waveforms of these signals are shown in figure 2. Fixed delays in the Clock System were assumed to be zero for the purpose of drawing the waveforms.

CLOCK WAVEFORMS AQUARIUS

INPUTS TO CLOCK SHAPING LOGIC



ON GATE ARRAY CLOCKS



STRAM CLOCKS

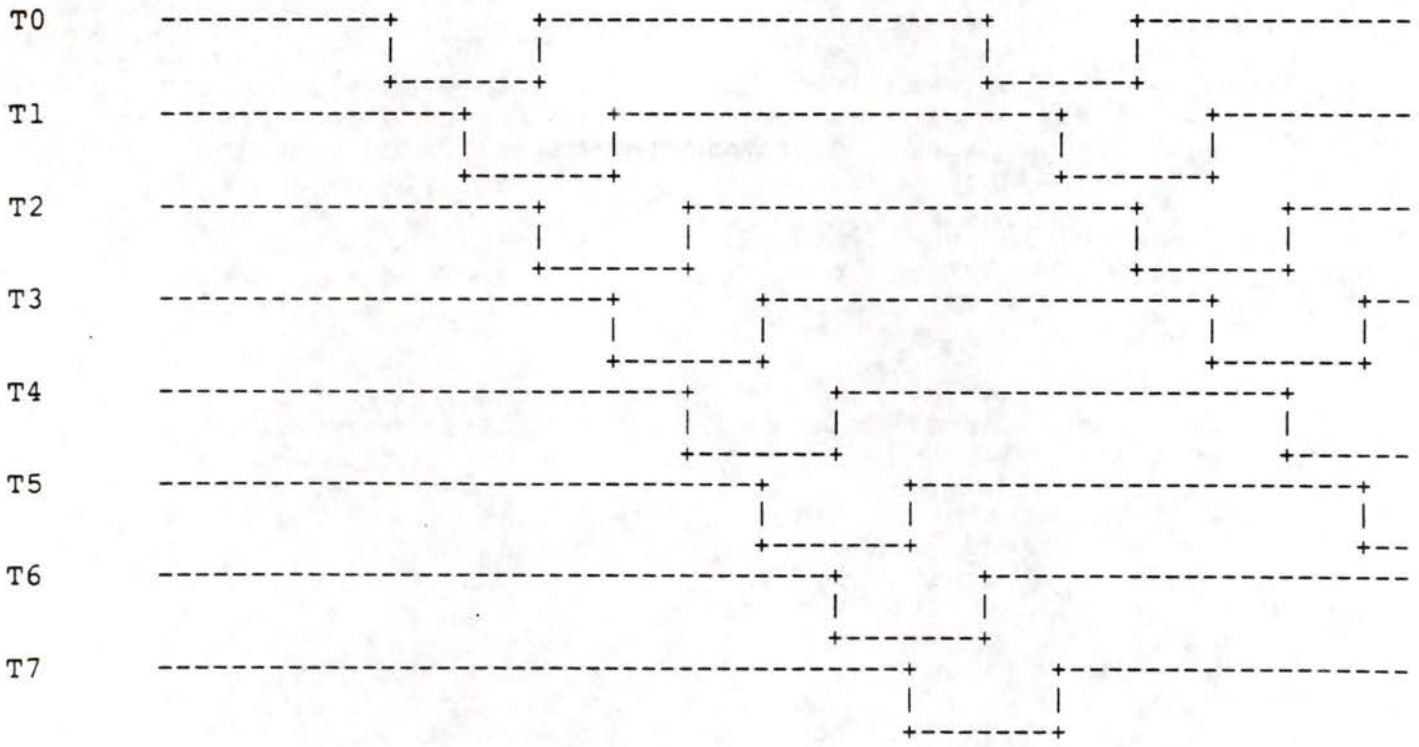


FIGURE 2

The Clocks inside the Gate Array will be shaped by the Clock distribution logic on the Gate Array. Two signals will be formed. One opens latches for 1/8 of a cycle and closes them for 7/8 of a cycle. This is the "A" phase Clock. The other is the "B" phase Clock which opens latches for 4/8 of a cycle and closes them for 4/8 of a cycle. The Clocks are positioned so that they are guaranteed to have 1/8 of a cycle of latch closure time between the closing of a "B" phase latch and the opening of the next "A" phase latch, and 2/8 of a cycle between the closing of an "A" phase latch and the opening of a "B" phase latch . This non-overlap guarantees that min delay races are not possible when transferring data from "A" to "B" or "B" to "A" phase latches, providing the sum of the Clock skew plus the receiving latch hold time minus the minimum delay from Clock to data out of the source latch is less than 1/8 of machine cycle.

The other devices that receive Clocks are the STRAMs. Their Clocks are shaped and distributed from a Clock distribution chip that resides on each MCU. They receive Clocks that are High for 3/4 of a cycle and Low for 1/4 of a cycle. Clocks of this shape are available starting every 1/8 of a cycle throughout an entire machine cycle.

The Aquarius Register File will use two of the STRAM Clocks to allow it to do a read and a write per machine cycle.

1.2 Clock Skew Requirements - Water Cooled Machines

The clock skew between two Clocks to state devices either inside a Gate Array or as received by a STRAM anywhere in subsystems on different printed circuit boards that have their logic inside of MCUs must be less than 1.5 NS.

The clock skew between two Clocks to state devices either inside of a Gate Array or as received by a STRAM anywhere in a subsystem on a single printed circuit board that has its logic inside of MCUs must be less than 1.5 NS.

The clock skew between any two Clocks to state devices either inside of a Gate Array or as received by a STRAM anywhere on the same MCU must be less than 1.0 NS.

The Aquarius machine will use register file parts which will require two different STRAM Clocks to drive the same part. The relative clock skew between two STRAM Clocks to the same register file chip on an MCU must be less than 0.5 NS.

The clock skew is an unsigned quantity that represents that the maximum time difference that can exist between the positions of the Clocks at any two clocked devices in the system. It represents an uncertainty about the position of the Clock due to manufacturing and design tolerances.

Known fixed offsets between Clocks are not considered part of the clock skew. Thus if an "A" phase and "B" phase Clock are known to have a 1/8 cycle separation this 1/8 cycle offset would not be considered as part of the clock skew.

The above clock skews are only guaranteed at the shortest system operating cycle time. At longer system cycle times the clock skew will increase. However, the rate of increase will be sufficiently slow that logic designed to operate at the shortest will not fail as the clock frequency is lowered as long as the timing design rules have been followed.

1.3 Clock Skew Requirements - Air Cooled Machines

The clock skew between two Clocks to state devices either inside a Gate Array or as received by a STRAM anywhere in subsystems on different printed circuit boards that have their logic inside of MCUs must be less than 1.9 NS.

The clock skew between two Clocks to state devices either inside of a Gate Array or as received by a STRAM anywhere in a subsystem on a single printed circuit board that has its logic inside of MCUs must be less than 1.9 NS.

The clock skew between any two Clocks to state devices either inside of a Gate Array or as received by a STRAM anywhere on the same MCU must be less than 1.4 NS.

The Aquarius machine will use register file parts which will require two different STRAM Clocks to drive the same part. The relative clock skew between two STRAM Clocks to the same register file chip on an MCU must be less than 0.5 NS.

The clock skew is an unsigned quantity that represents that the maximum time difference that can exist between the positions of the Clocks at any two clocked devices in the system. It represents an uncertainty about the position of the Clock due to manufacturing and design tolerances.

Known fixed offsets between Clocks are not considered part of the clock skew. Thus if an "A" phase and "B" phase Clock are known to have a 1/8 cycle separation this 1/8 cycle offset would not be considered as part of the clock skew.

The above clock skews are only guaranteed at the shortest system operating cycle time. At longer system cycle times the clock skew will increase. However, the rate of increase will be sufficiently slow that logic designed to operate at the shortest will not fail as the clock frequency is lowered as long as the timing design rules have been followed.

1.4 Memory Subsystem Clocks

The part of the Memory subsystem that resides in the JBOX MCUs will use the same Clocks as the JBOX uses. The JBOX will supply the clocks for the remainder of the memory subsystem by shipping STRAM Clocks, and a B phase clock to the that logic.

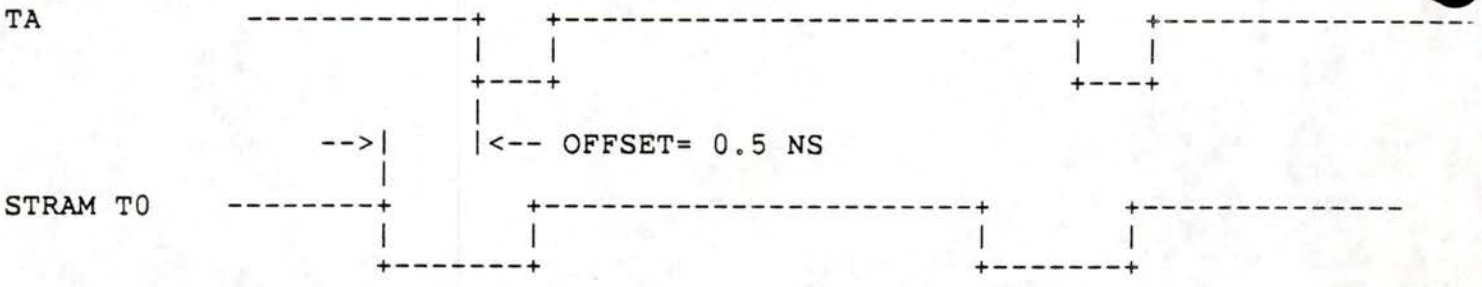
When the frequency of the clocks is changed the console will have to change the phase, "TX", of the STRAM Clock the Jbox is sending to the memory. The exact clock phase/clock frequency relationship will be specified in the Aquarius Memory specification. The procedure for changing clock frequency is TBD.

1.5 I/O Subsystem Clocks

The part of the I/O subsystem that is on the Jbox MCUs will use the same clocks the Jbox uses. The Jbox will supply clock signals to the rest of the logic in the I/O subsystem by shipping a logic signal and a copy of the B phase clock to the I/O subsystem.

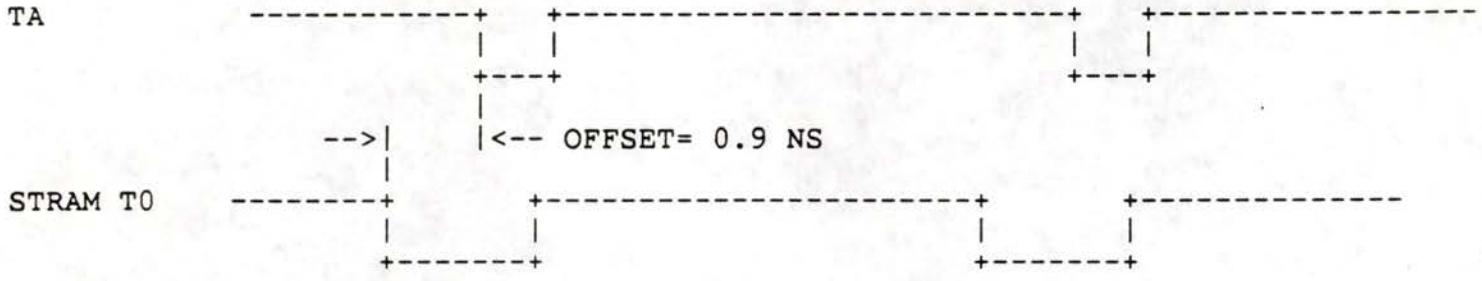
1.6 STRAM/Gate Array Clock Positioning Requirements - Water Cooled Machines

The nominal leading edge of the STRAM T0 clock at a STRAM will occur 0.5 NS before the nominal leading edge of the TA clock at the state devices within the gate array.



1.7 STRAM/Gate Array Clock Positioning Requirements - Air Cooled Machines

The nominal leading edge of the STRAM T0 clock at a STRAM will occur 0.9 NS before the nominal leading edge of the TA clock at the state devices within the gate array.



1.8 Clock Starting And Stopping

The Clock Control distribution chip on the Clock Module will contain a number of registers which will control the operation of the Clock System.

It will be possible to start and stop Clocks to each CPU of a system, and the JBOX independently.

It will be possible to separately enable the generation of Clocks for each of the above subsystems from a common burst counter which issues a programmable number of Clock Control pulses to each of the enabled subsystems. One cycles worth of Clocks will be generated in the subsystem for each Clock Control pulse that is received.

The spacing between the Clock Control pulses is controlled by loading the Interval register which holds a count representing the number of machine cycles between Clock Control pulses.

It will only be possible to step the clocks in the machine in full machine cycle increments. Fractional machine cycle steps will not be possible due to the design of the clock shaping and distribution logic in the machine.

1.9 Clock System Initialization

The Clock System will be initialized and controlled from the Aquarius Scan System that is operated by the Console. Synchronizers will be installed at appropriate points in the logic to bring the Aquarius Scan System control signals into the time frame of the Clock System.

1.10 Hot Disconnect

In a quad processor it shall be possible to power down two of the processors to perform maintenance while the other two processors are running. When this is done it must be possible to remove clock connections to the powered down processors without causing any malfunction in the running processors. This capability is needed to allow component replacement in the powered down processors.

This capability may be required in the dual processor configurations.

2 MASTER CLOCK MODULE

The Master Clock Module will contain the Clock Oscillator, some of the Clock and Clock Reference fannout circuitry, and Clock Control logic.

2.1 Clock Oscillator - Water Cooled Machines

The Clock Oscillator will be a phase locked voltage controlled oscillator that is capable of changing frequency over a range from 352 to 600 MHZ in 4 MHZ steps. The frequency control signals will come from a register in the Clock Control chip. The machines are expected to ship at an operating clock frequency of 500 MHZ.

The steady state clock frequency must have a long term accuracy of better than +/- 0.0025%, and once loop lock is achieved the frequency must not deviate from nominal by more than +/-0.2% due to loop hunting.

When changing frequency the oscillators output must not overshoot or undershoot the desired frequency in the process of acquiring lock by more than 1.0%.

The oscillator will provide a loop unlocked indicator which can be read by the Console.

To change the frequency from 352 MHZ to 600 MHZ, or from 600 MHZ to 352 MHZ or any smaller range should not require more than 1.0 second. This wide a frequency change may have to be made in several steps. The sum of the times required by the individual steps should not exceed 1.0 seconds.

The clock circuitry must provide an output at 1/8th the oscillators frequency whose rising edge 50% point is offset from the 50% point on the oscillators by TBD +/- TBD ps over the 352 to 600 MHZ operating range of the oscillator independent of loop lock. An amplified and appropriately delayed version of this signal will be distributed as the Clock Reference Signal.

The oscillator must always achieve lock following power up after its frequency register has been loaded. If required an initialization signal can be provided from the Console.

The output of the Clock Oscillator will be a differential sine wave.

The 1/8 Oscillator frequency output shall be a square wave signal.

The clock circuitry will provide a 1 MHZ output which has the same long term accuracy as the master oscillator.

Further details of the Clock Oscillator specifications can be found in the Aquarius Clock System Implementation specification from the HPS Technology group.

2.2 Clock Oscillator - Air Cooled Machines

The Clock Oscillator will be a phase locked voltage controlled oscillator that is capable of changing frequency over a range from 240 to 424 MHZ in 4 MHZ steps. The frequency control signals will come from a register in the Clock Control chip. The machines are expected to ship at an operating clock frequency of 352 MHZ.

The steady state clock frequency must have a long term accuracy of better than +/- 0.0025%, and once loop lock is achieved the frequency must not deviate from nominal by more than +/-0.2% due to loop hunting.

When changing frequency the oscillators output must not overshoot or undershoot the desired frequency in the process of acquiring lock by more than 1.0%.

The oscillator will provide a loop unlocked indicator which can be read by the Console.

To change the frequency from 240 MHZ to 424 MHZ, or from 424 MHZ to 240 MHZ or any smaller range should not require more than 1.0 second. This wide a frequency change may have to be made in several steps. The sum of the times required by the individual steps should not exceed 1.0 seconds.

The clock circuitry must provide an output at 1/8th the oscillators frequency whose rising edge 50% point is offset from the 50% point on the oscillators by TBD +/- TBD ps over the 240 to 424 MHZ operating range of the oscillator independent of loop lock. An amplified and appropriately delayed version of this signal will be distributed as the Clock Reference Signal.

The oscillator must always achieve lock following power up after its frequency register has been loaded. If required an initialization signal can be provided from the Console.

The output of the Clock Oscillator will be a differential sine wave.

The 1/8 Oscillator frequency output shall be a square wave signal.

The clock circuitry will provide a 1 MHZ output which has the same long term accuracy as the master oscillator.

Further details of the Clock Oscillator specifications can be found in the Aquarius Clock System Implementation specification from the HPS Technology group.

2.3 Clock Fanout To MCUs

The output of the Clock Oscillator will be sent to a power amplifier which will increase the amplitude of the Clock to the value required to drive the power dividers that will drive the individual cables that will supply the Clocks to modules and MCUs of the system. The actual implementation may use multiple levels of power dividers to achieve the required fanout.

This is shown in figure 1.

Details of the Clock fanout implementation will be found in the Aquarius Clock System Implementation specification from the HPS Technology Group.

2.4 Clock Reference Fanout To MCUs

The 1/8th frequency output of the Clock Oscillator will be sent to a programmable delay circuit to position it, and to a power amplifier which will increase its amplitude to the value required to drive the power dividers that will drive the individual cables that will supply the Clock Reference to modules and MCUs of the system. The actual implementation may use multiple levels of power dividers to achieve the required fanout.

This is shown in figure 1.

Details of the Clock Reference fanout implementation will be found in the Aquarius Clock System Implementation specification from the HPS Technology Group.

2.5 Clock Control Signal Fanout To MCUs

The output of the Clock Control chip will be sent to a gate array in each of the processors and major subsystems of Aquarius where enough copies of the Clock Control signal will be generated to distribute to each Clock Distribution chip in that subsystem.

This is shown in figure 1.

Details of the Clock Control fanout implementation will be found in the Aquarius Clock System Implementation specification from the HPS Technology Group.

2.6 Clock, Clock Reference, And Clock Control Signal Loading

The Clock, Clock Reference, and Clock Control signals must be sent to the modules and MCUs in the system that require Clocks. The total number of loads that the Clock, Clock Reference, and Clock Control signals must drive in each major hardware unit is listed below. Note that electrical considerations will require that the Clock Control signals be sent as a differential signal from the Master Clock Module to the other processors and the Jbox.

Clock Fanout Requirements - Quad Processor

				Within Subsystem Copies Of Signal Required		
Subsystem	Modules	MCUs	Spares	Clock	Ref	Control
CPU 0	1	16	0	16	16	16
CPU 1	1	16	0	16	16	16
CPU 2	1	16	0	16	16	16
CPU 3	1	16	0	16	16	16
JBOX	1	6	0	6	6	6

Table 1

Clock Fanout Requirements - Dual Processor

				Within Subsystem Copies Of Signal Required		
Subsystem	Modules	MCUs	Spares	Clock	Ref	Control
CPU 0	1	16	0	16	16	16
CPU 1	1	16	0	16	16	16
JBOX	1	4	0	4	4	4

Table 2

2.7 Clock Control Chip

The Clock Control chip has to perform a number of functions. These include: forming the Clock Control signal, aligning the Clock Reference signal with the edges of the Clock, gating the Clock Control signal on and off under Console control, supplying Clock frequency commands to the Clock Oscillator, fanning out the Clock Control signal to the modules and MCUs that make the computer system, and measuring the frequency of the clock oscillator.

A block diagram of the Clock Control Chip is shown in figure 5. The exact number of copies of each group of outputs will depend on the final implementation of the Clock Control fanout.

The Clock Control Chip provides Clock Control pulses to the major functional units of the Aquarius system. The Clock Control signal to each of these major units can be gated on and off by control logic that resides on the Chip. The operation of this control logic is determined by the contents of the several control registers that reside on the chip. The functions that are controlled by each of the registers is described in the section of the specification that describes the registers.

The chip contains logic to position the Clock Reference relative to the Clock. For this function there is a register which controls a programmable delay line. The Console reads a Clock Reference Position Comparison Flip Flop which is used to compare the position of a delayed copy of the Clock Reference with the position of the Clock. The rising edge of the Clock Reference must be aligned with the Clocks rising edge. The value of the delay line is adjusted by loading a register with the delay value sent from the Aquarius Scan System. By loading the register with progressively larger values a point is reached where the Clock and Clock Reference edges cross and flip flop will change state indicating correct alignment. The exact location of the Clock Reference Comparison Flip flop is an implementation detail. (Note: An alternative implementation of the reference positioning logic may use a closed loop servo system to position the Clock relative to the Clock Reference. If this is done then the need for the register that controls the programmable delay may go away.)

CLOCK CONTROL CHIP

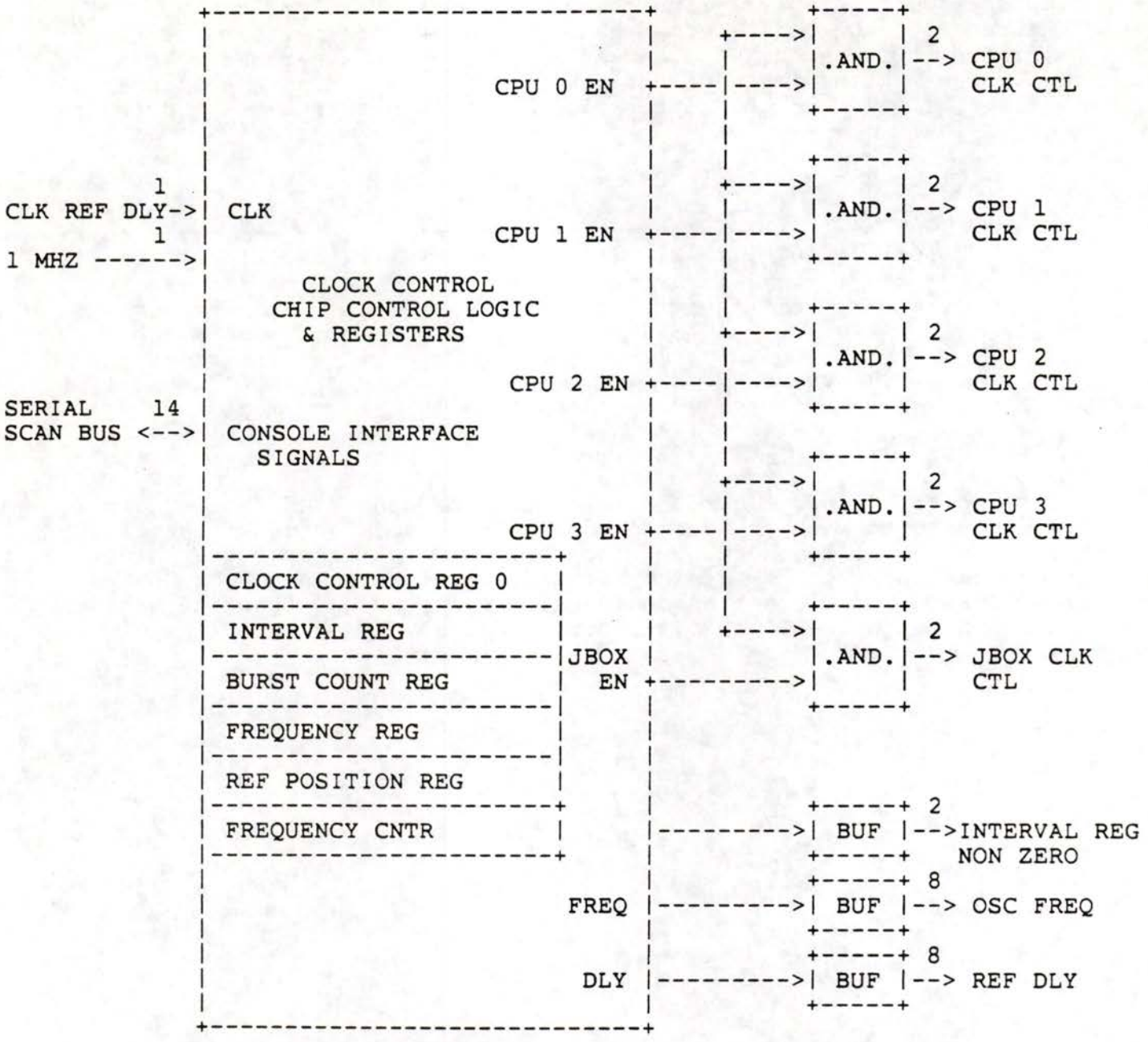
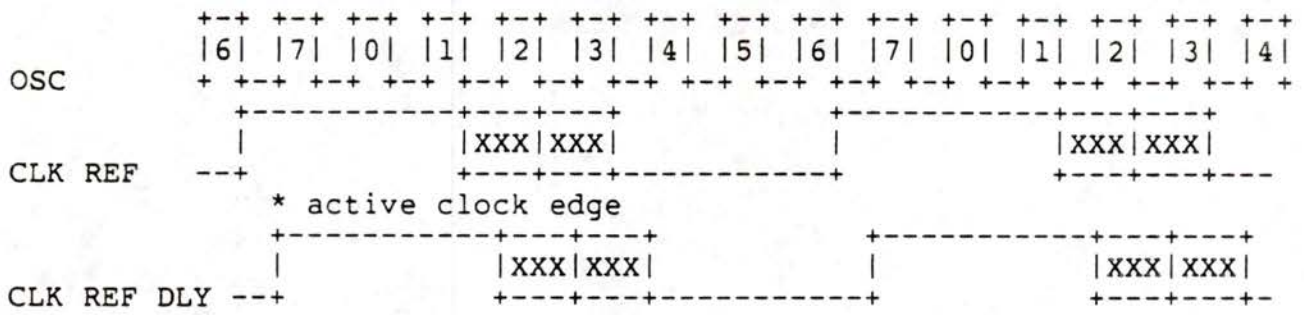


FIGURE 5

The Clock Control logic shall be clocked by the rising edge of a delayed copy of the Clock Reference signal. The delay will be 1/2 a Clock period of the Clock. This delay must be accomplished in high speed off gate array logic logic because the clock control chip is expected to be built from a gate array whose performance lies somewhere between an MCA I and an MCA II. Such a gate array cannot handle the maximum frequency of the Clock. The timing relationship of this clock to the other clocks on the Master Clock module is shown below:

Clock Control Logic Clock Timing



It is important that Clock Control signal be generated from a clock that is aligned with the rising edge of the Clock, because the logic that receives it on the Clock Distribution Chip clocks the signal into the chip on the rising edge of the Clock. The Clock Control signal must takes less than one machine cycle to get from the Master Clock module to the Clock Distribution Chips.

2.7.1 Registers -

The Clock Control Distribution Chip contains several registers which can be read or written over the Serial Scan Bus. The read write status of the register will be indicated in the description of the register. They are:

2.7.1.1 Clock Control Register 0 -

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

BIT NUMBER

0 - CPU 0 CLK RUN
1 - CPU 0 BURST EN
2 - CPU 1 CLK RUN
3 - CPU 1 BURST EN
4 - CPU 2 CLK RUN
5 - CPU 2 BURST EN
6 - CPU 3 CLK RUN
7 - CPU 3 BURST EN
8 - JBOX CLK RUN
9 - JBOX BURST EN
A - BURST GO
B - BURST HALT
C - LOOP UNLOCKED
D - REF POSITION STATUS
E - CLK INTERRUPT ENABLE
F - CLK INTERRUPT CLEAR

CLK RUN and BURST EN form a two bit field which controls whether the designated unit gets Clocks or not. The encoding of the field is as follows:

CLK RUN	BURST EN	FUNCTION
0	0	Stop all clocks to the unit
0	1	Enable running the clocks to the unit from the Burst Counter
1	0	Supply clocks to the unit as long as this encoding is true
1	1	Not allowed

BURST GO - The assertion of this bit causes the contents of the Burst Count register to be transferred to the Burst Counter. This causes a burst of Clock Control pulses equal to the in length to the transferred count to be generated. Each Clock Control pulse causes one machine cycles worth of clocks to be generated in the enabled subsystem. After Burst Go has been asserted the Burst Count register can be loaded with a new value without affecting the burst that is in progress. Write only bit.

BURST HALT _ When true the Clock stopped because the Burst Count has expired. The assertion of this bit will cause a console interrupt if clock interrupts are enabled. Read only bit.

LOOP UNLOCKED - When true the VCO has not acquired lock. The assertion of this bit will cause a console interrupt if clock interrupts are enabled. Read only bit.

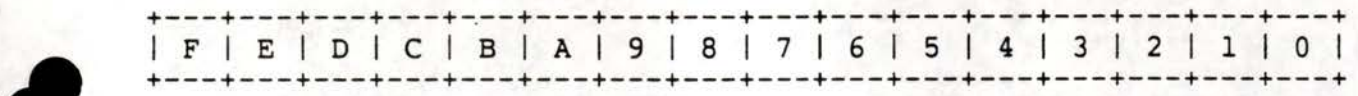
REF POSITION STATUS - This a read only bit which indicates the state of the flip flop which tests the time position of the Clock Reference signal relative to the Clock. It is a Zero if the rising edge of the Clock Reference occurs before the falling edge of the Clock, and a One if it occurs after the falling edge of the Clock.

CLK INTERRUPT ENABLE - When true and either BURST HALT or LOOP UNLOCKED are asserted an interrupt will be sent to the console. This is a read/write bit.

CLK CLEAR INTERRUPT - When true the BURST HALT and LOOP UNLOCKED flags will be cleared. This is a write only bit.

The contents of this register can be read and written from the Aquarius Scan System

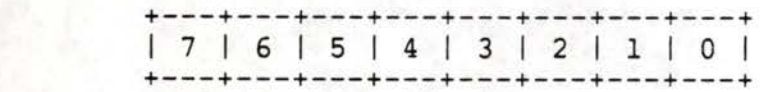
2.7.1.2 BURST COUNT REG -



This register hold a 16 bit binary number which specifies the number of machine cycles worth of Clocks that are to be generated when BURST GO is asserted.

The contents of this register can be read and written from the Aquarius Scan System.

2.7.1.3 Interval Register -



BIT NUMBER

7-0 INTERVAL COUNT

This register hold a 8 bit binary number which specifies the interval in machine cycle time periods between the generation of Clock Control pulses. A Clock Control pulse causes one cycles worth of clocks to be generated.

When a count of zero is loaded into this register Clock Control pulses will be generated on every cycle.

When a count of one is loaded into this register Clock Control pulses will be generated on every other cycle. This will cause the whole multiprocessor system to run at half speed.

When a count of "N" is loaded into this register Clock Control pulses will be generated on every "N+1" cycles. This will cause the whole multiprocessor system to run at one over "N+1" of its full speed.

Procedure for changing the interval count - TBD.

The contents of this register can be read and written from the Aquarius Scan System.

One of the outputs of the Clock Control logic is a signal which is true whenever the Interval register contains a non zero count. This signal may be required by the logic in the memory subsystem so that it can determine when the machine is running with slowed clocks.

2.7.1.4 FREQUENCY REG -

9	8	7	6	5	4	3	2	1	0

BIT

9-0 FREQ REG - This field contains a binary number which specifies the frequency in megahertz at which the Clock Oscillator is to run. The clock oscillator ignores the two least significant bits of this register. As a result the clock frequency is a multiple of 4 MHZ.

Procedure for changing clock frequency - TBD

The contents of this register can be read and written from the Serial Scan Bus.

2.7.1.5 FREQUENCY CNTR -

9	8	7	6	5	4	3	2	1	0

BIT

7-0 FREQ CNTR - This counter contains a binary number which which is equal to the clock frequency in MHZ. It is accurate to +/- 1 MHZ.

This register is read only.

2.7.1.6 REF POSITION REG -

7	6	5	4	3	2	1	0

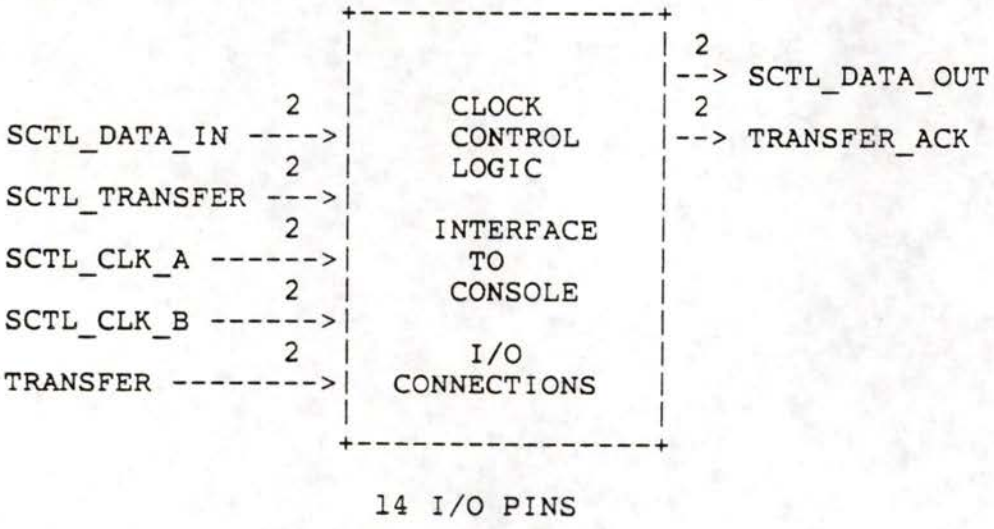
BIT NUMBER

7-0 REF DELAY This field controls the amount of time the Clock Reference is delayed relative to the Clock. A larger number causes that Clock Reference to move later in time relative to the Clock. The intention being to keep incrementing the count in this register until the REF POSITION STATUS bit changes from a Zero to a One thus indicating alignment of the Clock Reference and the Clock.

This field can both be read and written.

2.7.2 Master Clock Module To Console Interface -

For a description of how the Aquarius Scan System interfaces to the Master Clock Module interface see Mike Evans "Master Clock Module to Scan Controller Interface" memorandum. The contents of this memorandum will eventually be added to the Aquarius Scan System specification. A block diagram showing the connections between the console and Master Clock module is shown below. All signals are differential ECL signals.



2.7.3 Clock Reference Generation -

The Clock Reference will be generated by dividing the Clock by eight. The circuit that does this should be self initializing at power up time.

2.7.4 Clock Reference Positioning. -

As stated earlier the rising edge of the Clock Reference needs to be aligned with the rising edge of the Clock. This must be done in such a way that changes to the Clock Oscillators frequency do not cause the alignment to be lost. This can be accomplished if relative to a common origination point both Clock and Clock Reference are delayed by the same amount as they are distributed throughout the machine. It should be noted, that because the Clock runs continuously, that the Clock Reference can be delayed an arbitrary number of Clock cycles by passing it through flip flops clocked by the Clock without impacting the relationship between it and the Clock. This fact may prove useful when designing the control logic in the Clock Reference Distribution Chip.

2.7.5 Clock Reference Fanout -

The fanout requirements of the Clock Reference have been described in a previous sections of this specification. The exact implementation of that fanout will be described in the Aquarius Clock System Implementation specification that will be issued by the HPS Technology Group.

3 ON MCU CLOCK DISTRIBUTION

Each MCU will contain an Clock Distribution Chip which will receive the Clock, Clock Reference, and Clock Control signals from Master Clock Module. The Clock Distribution Chip will be responsible for providing differential copies of the Clock and Gated Clock Reference to the Gate Arrays on the MCU, shaping and supplying the differential Clocks required by the STRAMS on the MCU, receiving and supplying Aquarius Scan System data to the Gate Arrays on the MCU, communicating that scan data to the Console, controlling the clocking of the STRAMS and Gate Arrays on the MCU from the Serial Scan Bus, providing a port for reading the revision number of the Clock Distribution Chip, and the serial, type, and revision numbers of the MCU over the Aquarius Scan System.

3.1 Clock Distribution Chip

A block diagram of the Clock Distribution Chip is shown below.

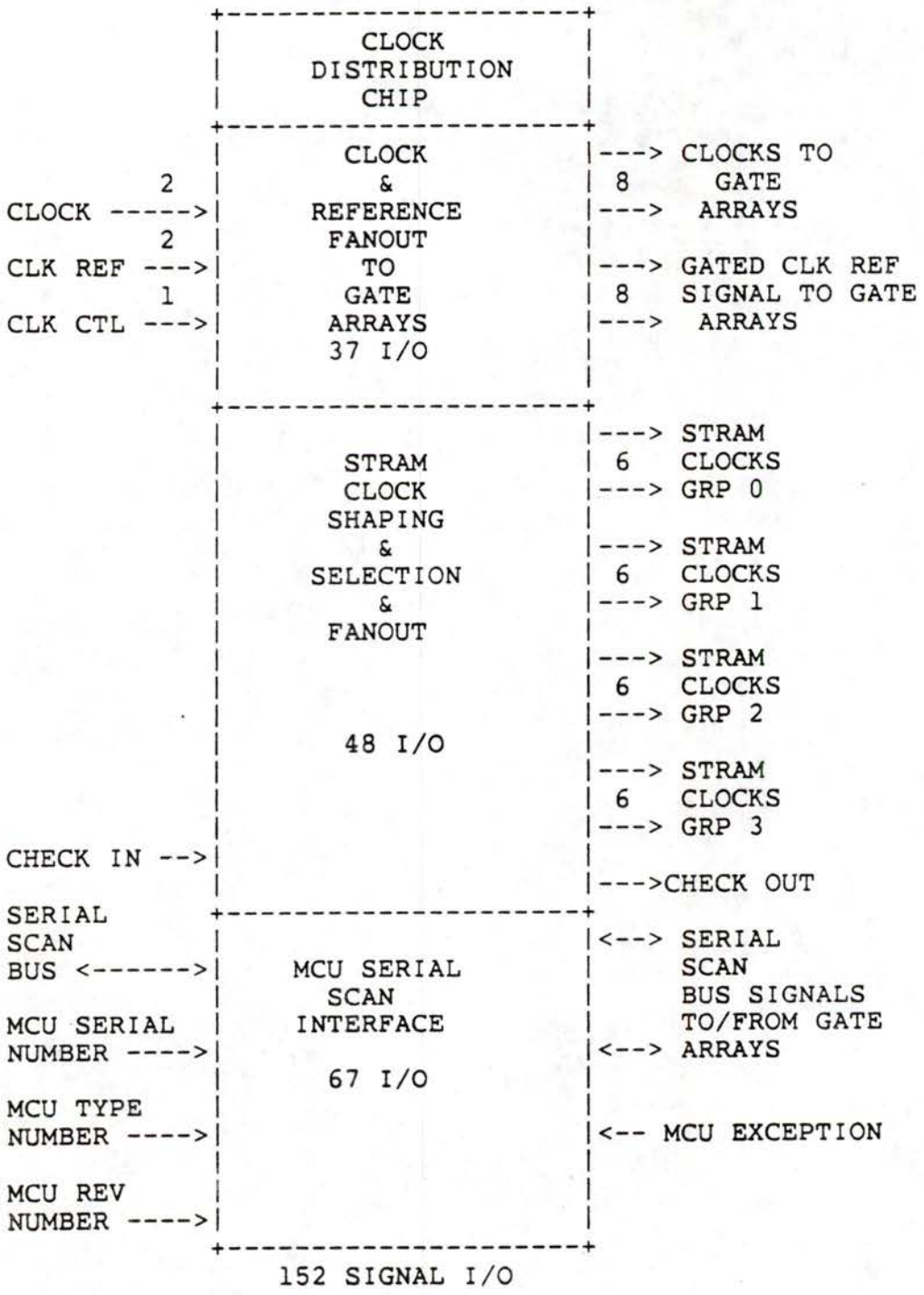


FIGURE 8

As can be see the chip is divided into 3 major functional sections.

The topmost group receives the Clock, Clock Reference, Clock Control signals from the fanout logic and creates one differential copy of the Clock and Gated Clock Reference signals for each of the 8 possible gate arrays on an MCU.

The middle group of signals uses the Clock, Clock Reference, Clock Control signals to form the STRAM Clocks. Four groups of STRAM Clocks with 6 differential copies of each of the clocks are formed. A different STRAM Clock phase can be selected for each group of outputs. Each group can select any of the 8 possible STRAM Clock phases. The selection is done by loading scan rings within the Clock Distribution Chip. In addition this section has an input/output pair of signals called CHECK IN and CHECK OUT. These signals operate in conjunction with a module 2 counter on the CD chip that increments after each T7 STRAM clock. The contents of this counter is compared with the CHECK IN input, which comes from an identical counter on a different CD chip via that chips CHECK OUT output, and if the CHECK IN input does not match the this counter an error flag will be set and a console interrupt generated because two CD chips have generated different numbers of clocks. This checker is intended to detect improperly seated MCUs. The counters on all CD chips are initialized to the same state via the scan system at system power up time.

The bottom signals are the chips interface to the Aquarius Scan System. This interface allows the Console to manipulate the contents of scan rings on the Clock Distribution Chip, and access the scan rings in the Gate Arrays on the MCU. In this latter function the Clock Distribution chip acts a central hub for distributing and collecting scan data from the Gate Arrays on the MCU. The Clock Distribution chip also acts as a collection point for sending interrupt signals to the console. To provide this function the MCU has an Exception input pin. This pin has a pulldown resistor to pull it Low when no input is connected. When driven high an interrupt will be sent to the console. In addition to this the Clock Distribution chip has an on chip temperature sensor which detects excessive MCU temperature an interrupts the console so an orderly system shutdown can take place. This sensor is intended to detect poor heat sink attachment and loss of coolant situations.

3.1.1 Registers -

The Clock Distribution Chip has internal scan rings which have a number of control/status bits which can be read or written over the Serial Scan Bus. Those bits are listed in this specification for completeness. Please refer to the Aquarius Custom Clock Distribution Chip Specification to determine the exact location of the bits within the scan rings.

CD REV NUM <3:0> - This field contains the revision number of the Clock Distribution chip.

MCU TYPE <7:0> - The contents of this field is the type number of MCU. This is a read only field.

MCU REV NUM <4:0> - The contents of this register is the revision number of the MCU. This register is read only.

MCU SERIAL NUM <14:00> - The contents of this register is the serial number of the MCU. This register is read only.

STRAM GRP 0 PHS SEL <3:0> - Contains the phase selection for the STRAM Clocks for STRAM Group 0. The 3 bit binary number is the number of the STRAM Clock phase. This field is read write.

STRAM GRP 1 PHS SEL <3:0> - Contains the phase selection for the STRAM Clocks for STRAM Group 1. The 3 bit binary number is the number of the STRAM Clock phase. This field is read write.

STRAM GRP 2 PHS SEL <3:0> - Contains the phase selection for the STRAM Clocks for STRAM Group 2. The 3 bit binary number is the number of the STRAM Clock phase. This field is read write.

STRAM GRP 3 PHS SEL <3:0> - Contains the phase selection for the STRAM Clocks for STRAM Group 3. The 3 bit binary number is the number of the STRAM Clock phase. This field is read write.

MCU OVER TEMP - Status bit which indicates that temperature of the MCU as measured by a sensor on the Clock Distribution Chip has exceeded 100 Degrees Centigrade +/- 5 Degrees. This indicates that the MCU is not being cooled and, therefor, the system should be shutdown.

Cleared by reducing the temperature of the MCU to 85 Degrees or lower.

MCU OVER TEMP ENABLE - Control bit that enables the overtemperature condition to interrupt the console when true.

MCU EXCEPTION COND ENABLE - Control bit that enables MCU exception conditions to interrupt the console when true.

MCU EXCEPTION - True if MCU EXCEPTION input signal to the Clock Distribution chip is High.

MCU LATCHED EXCEPTION - True if the MCU EXCEPTION line transitioned from Low to High.

Cleared by a Scan Ring 14 load function.

MCU CHECK ENABLE - Control bit that enables Comparison checking that this MCU has generated the same number of cycles worth of clocks as the MCU that is adjacent to it. This checking is done by comparing a divide by two cycle counter in this CD chip with on in an adjacent MCU. If this bit is true and a no compare situation is detected the console will be interrupted.

MCU CHECK ERROR - True if the check comparator is currently indicating an error.

MCU LATCHED CHECK ERROR - True if MCU CHECK ERROR came true since this bit was last cleared.

Cleared by a Scan Ring 14 load function.

3.1.2 MCU Serial Scan Interface -

A block diagram that shows the inputs and outputs of the Serial Scan Logic on the Clock Distribution Chip is shown below. An up to date description of the function of this logic can be found the Aquarius Scan System specification written by Mike Evans. This figure is only for reference.

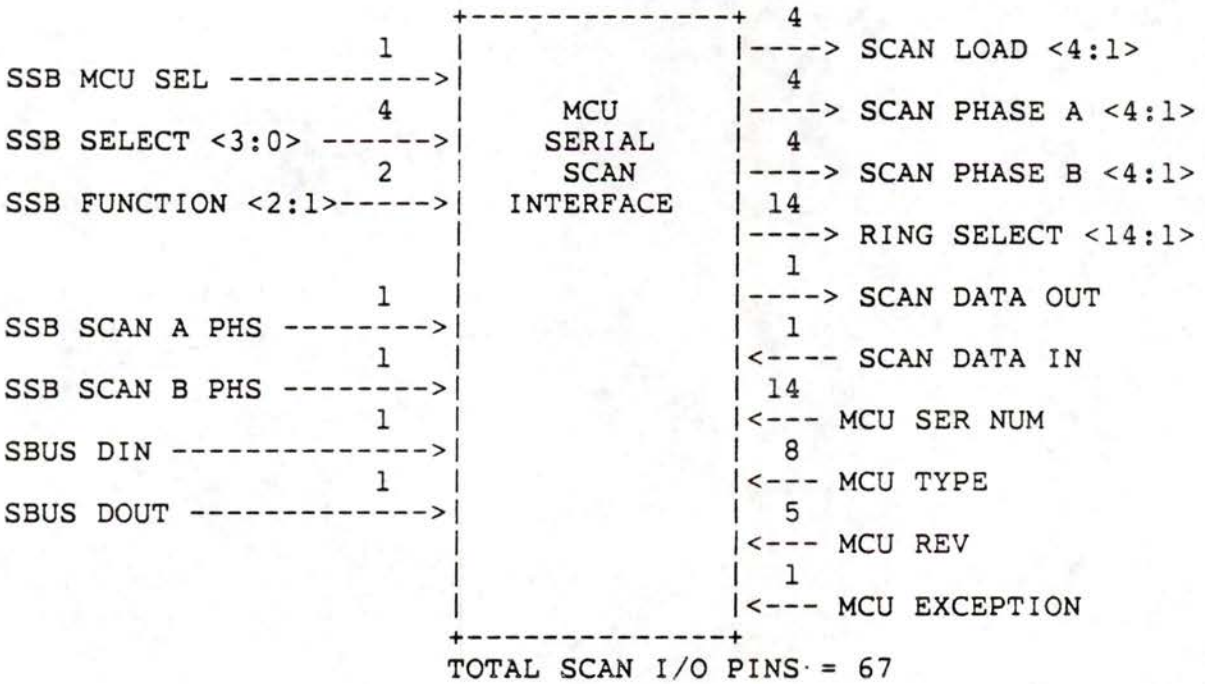


FIGURE 14

4 ON GATE ARRAY CLOCK DISTRIBUTION

On the Gate Array there is logic to shape and fanout clocks to the state devices on the Gate Array. The only clocks that are shaped on the Gate Array are the system TA and TB Clocks. These clocks are formed by shift registers and logic gates.

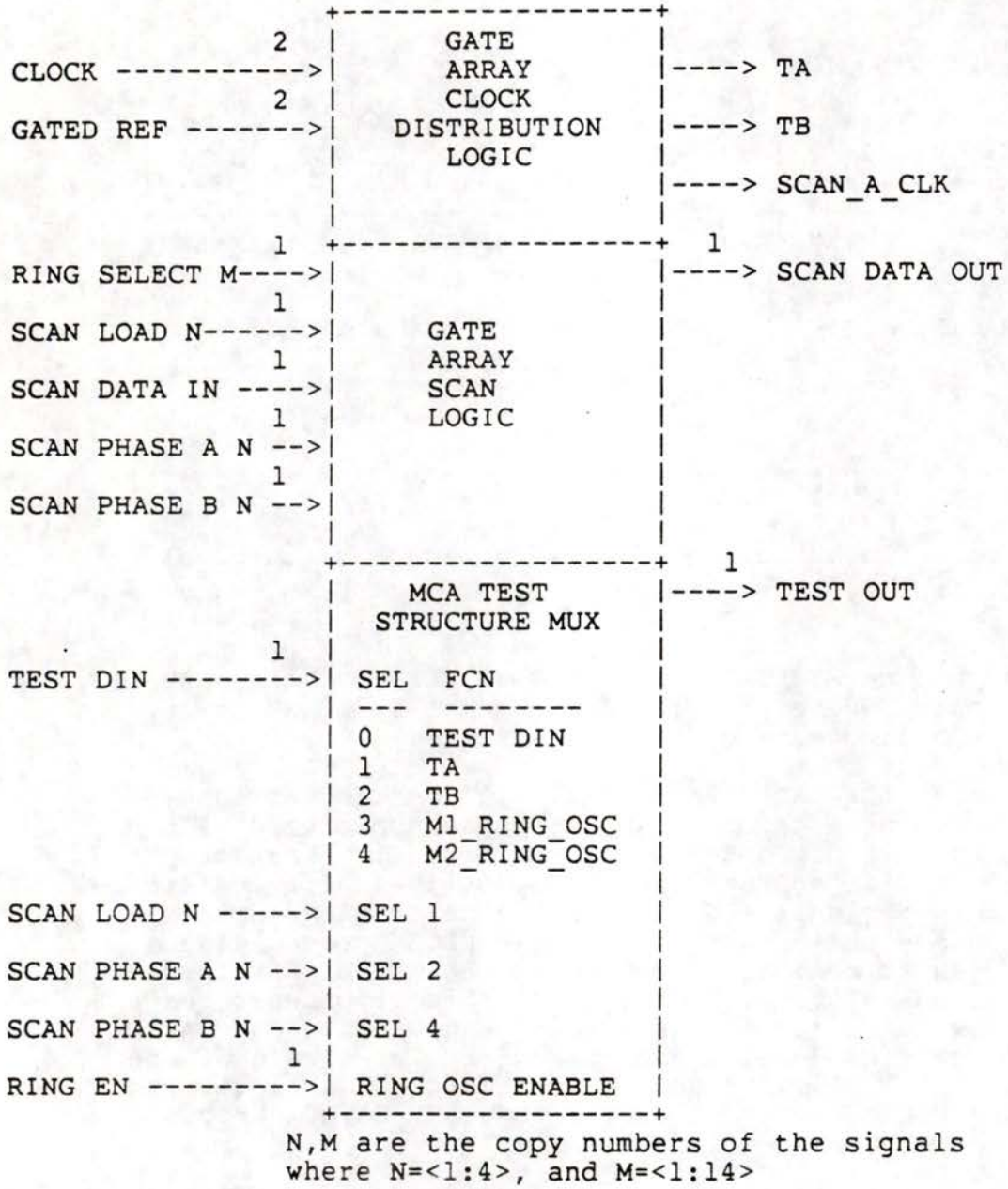
This logic will have a scanable module two counter that is incremented at the end of each machine cycles worth of clocks. The counters on all MCAs will be initialized via the scan system to the same state at system power up time. When a system error occurs the state of the counters on all MCAs can be compared to determine if the error was caused by the clocks on an MCA getting out of sequence with those in the other MCAs in the system.

The gate array also has scan logic which fans out the SCAN_A_CLK to the A phase scan latches in the system. The scan logic can force system TA and TB clocks.

In addition to this the gate array has a test structure multiplexer which brings certain signals out to an I/O pin so that the AC performance of the MCA can be evaluated. These are TA and TB phase clocks for on MCA clock distribution logic delay measurement, an external Test Input to measure the delay through the test MUX, metal 1 and metal 2 ring oscillators for measuring the overall performance of the gate array. The ring oscillators are enabled by a separate Ring Enable input. The MUX selects are shared pins from inputs to the Scan logic to reduce the number of input pins required by the test structure.

A block diagram showing the inputs and outputs of the on gate array clock and scan logic is shown below.

ON GATE ARRAY CLOCK DISTRIBUTION



N,M are the copy numbers of the signals where N=<1:4>, and M=<1:14>

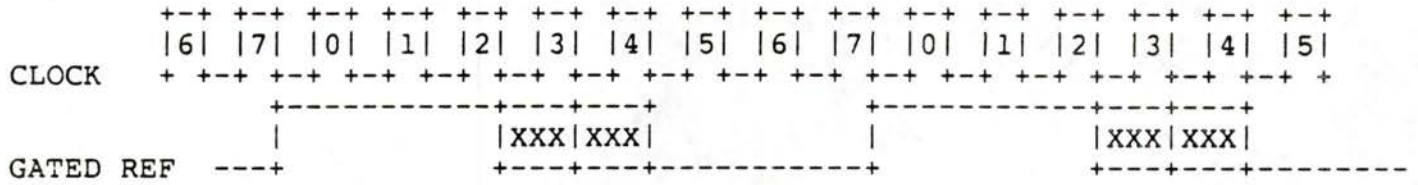
13 I/O PINS

The block diagram shows the signals that are used to form TA, TB, and SCAN_A_CLK, and the scan interface signals to the gate array.

The timing diagram that is shown below shows the waveforms of the input and output signals of the logic that is used to shape the system SYS_TA_CLK and SYS_TB_CLK signals which which become the TA and TB clocks during normal system operation.

ON GATE ARRAY CLOCK SHAPING LOGIC WAVEFORMS

INPUTS TO CLOCK SHAPING LOGIC



OUTPUT OF CLOCK SHAPING LOGIC

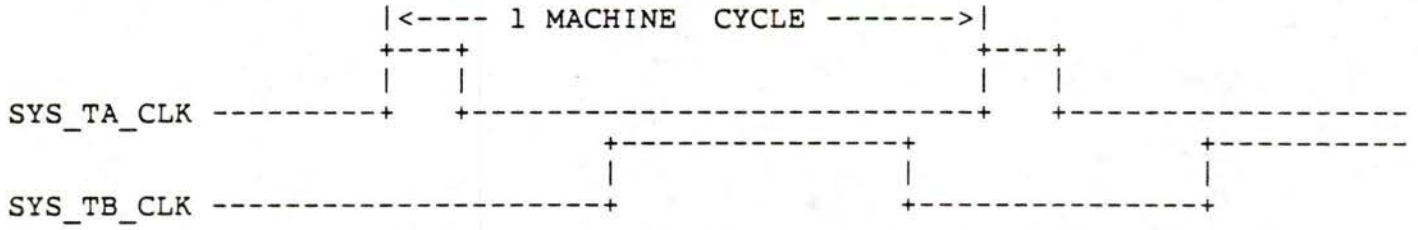


FIGURE 2

This logic shapes the SYS_TA_CLK and SYS_TB_CLK from the GATED REF and CLOCK signals. Note that the rising edge GATED REF signal is aligned with the falling edge of the CLOCK signal.

It is assumed that the GATED REF signal will be clocked into a flip flop by the rising edge of the CLOCK pulse numbered 0. This flip flop is the first bit of an 8 bit shift register. The output of this flip flop and an appropriate number of the shift register outputs will be combined with the GATED REF signal at the input to the flip flop to cause a pulse that is a single CLOCK period wide to be the input to the shift register even though the GATED REF pulse is 3 to 5 CLOCK periods wide. Thus the first flip flop generates a single clock period wide pulse off the leading edge of each GATED REF pulse that is shifted down the shift register at a rate of one bit per CLOCK period. This will cause the clock shaping logic to generate one machine cycles worth of clocks for each GATED REF pulse that is received.

The outputs of the flip flops in the shift register will be combined in logic gates and fed to flip flops which will develop the SYS_TA_CLK and SYS_TB_CLK clock waveforms. Enough copies of the SYS_TA_CLK and SYS_TB_CLK flip flop outputs will be created to drive the all the clock buffer gates on the gate array.

In the clock buffer gates the SYS_TA_CLK and SYS_TB_CLK clocks will be ORed with scan clocks before being sent to the clocked devices on the gate array. The outputs of the clock buffers will be the TA, and TB clocks.

Thus TA=SYS_TA_CLK+(RING_SELECT_M)*(SCAN_PHASE_A_N)*(SCAN_LOAD_N). This allows the scan logic to force the TA clock. This function must not be used to test the paths between the MBOX and the Jbox because these paths depend on a fixed timing relationship between the clocks involved. The delay of this path is greater than one machine cycle.

Thus TB=SYS_TB_CLK+SCAN_PHASE_B_N. Thus the B phase scan clock uses the same clock distribution metal as the system TB clock.

The console will control the generation of the SCAN_PHASE_A and SCAN_PHASE_B clocks. It will have the ability to control the order that these clocks are issued in, and the number of each clock pulses that are generated.

A logic diagram showing how the system TA and TB clocks are combined with the scan A and B clocks is shown the Aquarius Scan System specification that was written by Mike Evans.

4.1 On Gate Array Clock Fanout

The details of the clock fanout on the gate array will be found in an Gate Array Clock Distribution specification that is being written by the HPSC/TR&E group. The intent is that a fixed metal pattern will be used on all gate arrays to distribute the TA, TB, SYS_TA_CLK, and SYS_TB_CLK.

4.2 On Gate Array Scan Clock Distribution

The SCAN_A_CLK distribution will be done by the CAD system. It is a non critical signal from a timing standpoint.

APPENDIX A

MINIMUM OSCILLATOR PERIOD - WATER COOLED MACHINES

The minimum clock period that the Clock Oscillator can be run at is dependent on a number of different constraints, any one of which can set the limit.

These constraints often require state device timing parameters for their computation. Because MCA III is not designed yet the state device timing parameters will be derived by dividing those given for MCA II cells by a factor of two. The values obtained are summarized below:

	LATCH		FLIP FLOP	
PARAMETER	INT CELL	O-CELL	INT CELL	O-CELL
Tpd clk-dout Max	400	750	400	750 PS
Tpd clk-dout Min	100	200	100	200
Tsetup	400	750	400	750
Thold	0	0	0	0
Clk PW Min	625	750	625	750

In addition the timing parameters for the STRAMS must be known. They are given below:

STRAM TIMING PARAMETERS

Tpd clk-dout Min	300 PS
Tsetup	400
Thold	800
Clk PW Low Min	3000
STRAM Clock Offset	500

The minimum delay of an input cell is 0.05 ns and the minimum delay

of etch between two adjacent chips on the same MCU is assumed to be 0.05 ns. The minimum delay of the etch between chips on two different MCUs is assumed to be 0.45 ns. Clock skew for data transfers on the same MCU is 1.0 ns and the clock skew for data transfers between two different MCUs is 1.5 ns. The nominal leading edge of a STRAM T0 clock occurs 0.5 NS before the nominal leading edge of a TA clock inside of a gate array.

A.0.0.1 Minimum Clock Separation Required To Prevent Min Delay Races -

In order to have a design that is free of min delay races a minimum period of non overlap must be guaranteed between clocks of state devices that are participating in a transfer. For the above Clock Systems this time must be greater than the sum of the clock skew plus the destination state device Hold Time minus the minimum propagation delay from clock to data out of the source state device minus any min delay between the source and load minus(STRAM to gate array) or plus(Gate array to STRAM) any STRAM clock offset. This time is also equal to the period of the Clock Oscillator because this is the smallest Clock separation that can be achieved.

In equation form this is:

CLK PW >= (CLKSKEW + THOLDdest - TPD clk to dout source
- TPD min source to load - Toffset)

which an MCA III internal cell latch to latch transfer is:

On MCU: CLK PW >= (1.0 + 0.0 - 0.3- 0.1 - 0.0) = 0.8 NS

Between MCU: CLK PW >= (1.5 + 0.0 - 0.3- 0.5 - 0.0) = 0.7 NS

MIN OSC PERIOD >= (CLK PW) = 0.8 NS

and for an output cell to internal cell latch to latch transfer is:

On MCU: CLK PW >= (1.0 + 0.0 - 0.2 - 0.1 - 0.0) = 0.7 NS

Between MCU: CLK PW >= (1.5 + 0.0 - 0.2 - 0.5 - 0.0) = 0.8 NS

MIN OSC PERIOD >= (CLK PW) = 0.8 NS

and for an output cell to STRAM transfer is:

On MCU: CLK PW >= (1.0 + 0.8 - 0.2 - 0.1 - 0.5) = 1.0 NS

Between MCU: CLK PW >= (1.5 + 0.8 - 0.2 - 0.5 - 0.5) = 1.1 NS

MIN OSC PERIOD >= (CLK PW) = 1.1 NS

and for a STRAM to internal cell latch to latch transfer is:

On MCU: CLK PW >= (1.0 + 0.0 - 0.3 - 0.1 + 0.5) = 1.1 NS

Between MCU: CLK PW >= (1.5 + 0.0 - 0.3 - 0.5 + 0.5) = 1.2 NS

MIN OSC PERIOD >= (CLK PW) = 1.2 NS

A.0.0.2 Optimal Clock Pulse Width -

In Venus there was a problem trying to figure out how much logic would fit between two latches because the latches were clocked with wide Clock pulses. The wide Clock pulses caused the output valid time of a latch to be dependent on the timing of the logic that preceded that latch because the latch was transparent for a large part of a machine cycle. This meant that a designer might have to trace a signal back through a number of cycles worth logic in order to figure out just when the latches output would be valid. This caused problems in logic design and timing verification.

With the above Clock Systems this uncertainty can be made zero if on a machine cycle boundary at least one of the clock pulses are kept sufficiently narrow. This will cause there to be a precise timing Reference point per machine cycle. The "A" phase Clock has been made narrow for precisely this reason.

The optimal Clock pulse width is one which has a width equal to the sum of the Clock Skew plus the latch setup time. A pulse narrower than this width will cause clock skew to be added to the signal propagation delay through the latch and a pulse wider than this width will cause the valid time of the data out of the latch to be dependent on the valid time of the data at the latches input. In this sense this clock pulse width can be termed optimal.

CLK PW OPTIMAL= CLK SKEW + TSETUPdest

which for MCA III internal cell latches is estimated to be:

CLK PW OPTIMAL = 1.5 + 0.4 = 1.90 NS.

OSC PERIOD OPTIMAL = (CLK PW OPTIMAL) = 1.90 NS.

A.0.0.3 Clock Pulse Width Required To Operate A Latch Or Flip Flop

The MCA III latches and flip flops need Clock pulses which are wider than some minimum value for reliable operation. This width determines the extent to which the Clock pulses from the oscillator can be allowed to shrink by the time they arrive at the state devices inside the Gate Arrays.

The minimum Clock pulse width required for MCA III internal and I/O cell operation will be estimated by taking 1/2 the value given in the MCA II manual. This is the best that can be done until better values can be obtained from MCA III circuit simulations.

MINIMUM CLOCK PW REQUIRED TO OPERATE A STATE DEVICE

	MCA II	MCA III
INT CELL	1250	625 PS
O-CELL	1500	750 PS

A.0.0.4 Minimum Clock Pulse Width To A STRAM -

STRAMS need Clock pulses in order to operate. Current plans call for one set of Clock outputs to drive 2 STRAMS. In order to not lose noise margin on the clock lines the Clock pulse must reach full amplitude at the STRAMS. The Clock outputs are driven from source terminated ECL outputs to the loads through lossy interconnect. The STRAMS should appear to be a lumped capacitive load. As a first approximation the source, lossy line, and STRAMS can be modeled as a voltage source driving a capacitive load through a resistor. Where the resistor is the sum of the source impedance and the lossy line resistance. By requiring that the clock pulse width at the STRAM be at least 4 RC time constants wide one can be reasonably sure that the voltage at the load will reach nearly full amplitude.

This calculation proceeds as follows:

MINIMUM CLOCK PULSE WIDTH AT STRAM

Cin STRAM = 3.0 PF MAX
Cin Tab = 0.113 PF MAX
Cetch pin-pin = (0.85 inches)*(182 PS/inch)/(47 ohms) = 3.87 PF

Cload = 2*(Cin STRAM + Cin Tab) + 1*(Cetch pin-pin) = 10.1 PF

Rs = 67 ohms Max
Rline = 3.0 ohms/inch * 7 inches = 21 ohms

Rtotal = Rs + Rline = 21 + 67 = 88 ohms

Min STRAM Clock PW >= 4*Rtotal*Cload = 4*(88.0 Ohms)*(10.1 PF)=
>= 3.55 NS

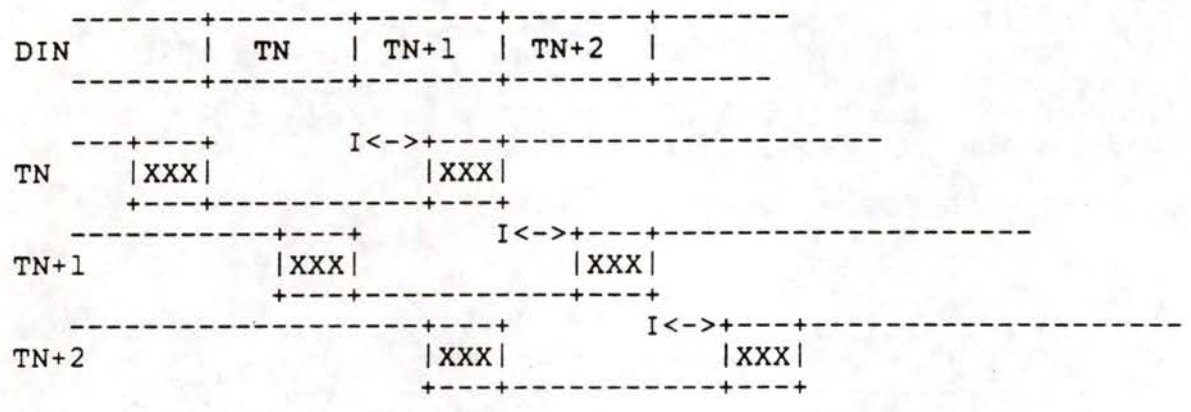
Min Oscillator Period >= 3.55/2= 1.78 NS

Note that the above pulse width is greater than the 3.0 NS minimum pulse with that the STRAM requires for correct operation.

A.0.0.5 Minimum Separation And Low State Width Of STRAM Clocks -

In order to always be able to pick a STRAM clock no matter at what time in the cycle the inputs to a STRAM become valid the separation and width of the STRAM clocks must satisfy certain constraints.

The constraints that must be satisfied are: the separation between different STRAM Clock phases must be greater than the sum of the clock skew plus the setup time of the STRAM, STRAM Clock pulses must be at least twice this width in order to guarantee that the latch is able to accept data for a period of time equal to the STRAM Clock pulse separation while operating the latch with an optimal width clock pulse so that the delay through the latch will be minimized. This is illustrated in the below.



I<-> = Tsetup STRAM

|XXX| = Clock Skew

STRAM Pulse Separation >= Clock Skew + Tsetup STRAM =
= 1.5 + 0.4 = 1.9 ns
STRAM Pulse Width >=2*(Clock Skew + Tsetup STRAM)
=2*(1.9) = 3.8 ns

A further constraint on the STRAM Clock pulses is that they must be formed off of the outputs of shift registers. This means that the pulse width and separation must be in multiples of one "N'th" of a machine cycle if a machine cycle is divided into "N" parts. Therefore, the minimum clock separation is Tcyc/N seconds, and the minimum STRAM Clock pulse width is 2*(Tcyc/N) seconds. This means the minimum clock oscillator period is 1.9 ns.

A.0.0.6 Minimum Clock Period And Pulse Width Summary -

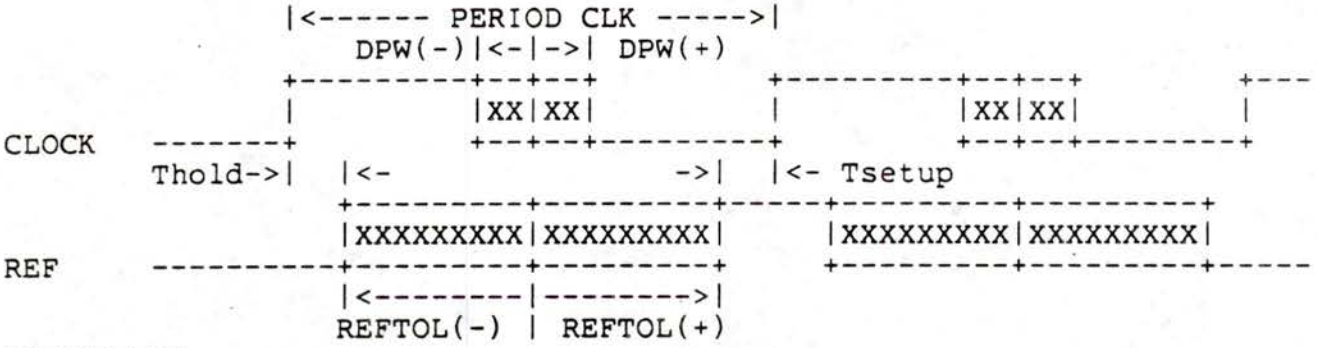
From the above analysis it can be seen that the minimum oscillator period must be greater than 1.90 NS. This corresponds to an oscillator maximum frequency of 526 MHZ. The STRAM Clock Low state must be greater than 3.8 ns and the STRAM clocks should be separated by 1/2 the width of the STRAM Clock Low state.

A.0.1 Clock Reference Positioning Accuracy

The Clock Reference signals rising and falling edges are aligned with the falling edges of the Clock at the Clock Module. In the clock shaping logic on the Clock distribution Chip and on the Gate Arrays the edges of the Clock Reference and the Clock must maintain a certain tolerance in order to guarantee correct operation of the Clock shaping logic.

In the Clock shaping logic the Clock is connected to the Clock input of a flip flop and the Clock Reference to the data input of the flip flop. For correct operation the position of the Clock Reference relative to the Clock must not violate the setup and hold time specifications of this flip flop. For this to be true the following two equations must be satisfied.

CLOCK REFERENCE POSITION TIMING DIAGRAM



AT 526 MHZ

REFTOL(-) = (PERIOD CLK) - Thold = 1.90-0.0 = 1.90 NS

REFTOL(+) = (PERIOD CLK) - TSETUP = 1.90-0.4 = 1.50 NS

NOTE: REFTOL INCLUDES ANY TOLERANCES IN THE POSITION OF THE FALLING EDGE OF THE CLOCK.

FIGURE 4

The relative position of the Clock and Clock Reference as seen by the Clock Distribution Chip and Clock logic on the Gate Arrays should not change when the Clock Oscillators frequency is changed. This can only happen if the Clock and Clock Reference pass through equal delay paths from the common point at which both originate.

Note, however, that the Clock Reference can be delayed any integer number of Clock cycles without affecting this delay because the Clock Reference is generated off of the edges of the Clock which is always running.

APPENDIX B

MINIMUM OSCILLATOR PERIOD - AIR COOLED MACHINES

The minimum clock period that the Clock Oscillator can be run at is dependent on a number of different constraints, any one of which can set the limit.

These constraints often require state device timing parameters for their computation. Because MCA III is not designed yet the state device timing parameters will be derived by dividing those given for MCA II cells by a factor of two. The values obtained are summarized below:

	LATCH		FLIP FLOP	
PARAMETER	INT CELL	O-CELL	INT CELL	O-CELL
Tpd clk-dout Max	800	750	800	750 PS
Tpd clk-dout Min	200	200	200	200
Tsetup	800	750	800	750
Thold	0	0	0	0
Clk PW Min	1250	750	1250	750

In addition the timing parameters for the STRAMS must be known. They are given below:

STRAM TIMING PARAMETERS	
Tpd clk-dout Min	300 PS
Tsetup	400
Thold	800
Clk PW Low Min	3000
STRAM Clock Offset	500

The minimum delay of an input cell is 0.05 ns and the minimum delay

of etch between two adjacent chips on the same MCU is assumed to be 0.05 ns. The minimum delay of the etch between chips on two different MCUs is assumed to be 0.45 ns. Clock skew for data transfers on the same MCU is 1.4 ns and the clock skew for data transfers between two different MCUs is 1.9 ns. The nominal leading edge of a STRAM T0 clock occurs 0.9 NS before the nominal leading edge of a TA clock inside of a gate array.

B.0.0.1 Minimum Clock Separation Required To Prevent Min Delay Races -

In order to have a design that is free of min delay races a minimum period of non overlap must be guaranteed between clocks of state devices that are participating in a transfer. For the above Clock Systems this time must be greater than the sum of the clock skew plus the destination state device Hold Time minus the minimum propagation delay from clock to data out of the source state device minus any min delay between the source and load minus(STRAM to gate array) or plus(Gate array to STRAM) any STRAM clock offset. This time is also equal to the period of the Clock Oscillator because this is the smallest Clock separation that can be achieved.

In equation form this is:

CLK PW >= (CLKSKEW + THOLDdest - TPD clk to dout source
- TPD min source to load - Toffset)

which an MCA III internal cell latch to latch transfer is:

On MCU: CLK PW >= (1.4 + 0.0 - 0.4- 0.1 - 0.0) = 0.9 NS

Between MCU: CLK PW >= (1.9 + 0.0 - 0.4- 0.5 - 0.0) = 1.0 NS

MIN OSC PERIOD >= (CLK PW) = 1.0 NS

and for an output cell to internal cell latch to latch transfer is:

On MCU: CLK PW >= (1.4 + 0.0 - 0.2 - 0.1 - 0.0) = 1.1 NS

Between MCU: CLK PW >= (1.9 + 0.0 - 0.2 - 0.5 - 0.0) = 1.2 NS

MIN OSC PERIOD >= (CLK PW) = 1.2 NS

and for an output cell to STRAM transfer is:

On MCU: CLK PW >= (1.4 + 0.8 - 0.2 - 0.1 - 0.9) = 1.0 NS

Between MCU: CLK PW >= (1.9 + 0.8 - 0.2 - 0.5 - 0.9) = 1.1 NS

MIN OSC PERIOD >= (CLK PW) = 1.1 NS

and for a STRAM to internal cell latch to latch transfer is:

On MCU: CLK PW >= (1.4 + 0.0 - 0.3 - 0.1 + 0.9) = 1.9 NS

Between MCU: CLK PW >= (1.9 + 0.0 - 0.3 - 0.5 + 0.9) = 2.0 NS

MIN OSC PERIOD >= (CLK PW) = 2.0 NS

B.0.0.2 Optimal Clock Pulse Width -

In Venus there was a problem trying to figure out how much logic would fit between two latches because the latches were clocked with wide Clock pulses. The wide Clock pulses caused the output valid time of a latch to be dependent on the timing of the logic that preceded that latch because the latch was transparent for a large part of a machine cycle. This meant that a designer might have to trace a signal back through a number of cycles worth logic in order to figure out just when the latches output would be valid. This caused problems in logic design and timing verification.

With the above Clock Systems this uncertainty can be made zero if on a machine cycle boundary at least one of the clock pulses are kept sufficiently narrow. This will cause there to be a precise timing Reference point per machine cycle. The "A" phase Clock has been made narrow for precisely this reason.

The optimal Clock pulse width is one which has a width equal to the sum of the Clock Skew plus the latch setup time. A pulse narrower than this width will cause clock skew to be added to the signal propagation delay through the latch and a pulse wider than this width will cause the valid time of the data out of the latch to be dependent on the valid time of the data at the latches input. In this sense this clock pulse width can be termed optimal.

CLK PW OPTIMAL= CLK SKEW + TSETUPdest

which for MCA III internal cell latches is estimated to be:

CLK PW OPTIMAL = 1.9 + 0.8 = 2.7 NS.

OSC PERIOD OPTIMAL = (CLK PW OPTIMAL) = 2.7 NS.

B.0.0.3 Clock Pulse Width Required To Operate A Latch Or Flip Flop

The MCA III latches and flip flops need Clock pulses which are wider than some minimum value for reliable operation. This width determines the extent to which the Clock pulses from the oscillator can be allowed to shrink by the time they arrive at the state devices inside the Gate Arrays.

The minimum Clock pulse width required for MCA III internal and I/O cell operation will be estimated by taking 1/2 the value given in the MCA II manual. This is the best that can be done until better values can be obtained from MCA III circuit simulations.

MINIMUM CLOCK PW REQUIRED TO OPERATE A STATE DEVICE

	MCA II	MCA III
INT CELL	1250	1250 PS
O-CELL	1500	750 PS

B.0.0.4 Minimum Clock Pulse Width To A STRAM -

STRAMS need Clock pulses in order to operate. Current plans call for one set of Clock outputs to drive 2 STRAMS. In order to not lose noise margin on the clock lines the Clock pulse must reach full amplitude at the STRAMS. The Clock outputs are driven from source terminated ECL outputs to the loads through lossy interconnect. The STRAMS should appear to be a lumped capacitive load. As a first approximation the source, lossy line, and STRAMS can be modeled as a voltage source driving a capacitive load through a resistor. Where the resistor is the sum of the source impedance and the lossy line resistance. By requiring that the clock pulse width at the STRAM be at least 4 RC time constants wide one can be reasonably sure that the voltage at the load will reach nearly full amplitude.

This calculation proceeds as follows:

MINIMUM CLOCK PULSE WIDTH AT STRAM

Cin STRAM = 3.0 PF MAX
Cin Tab = 0.113 PF MAX
Cetch pin-pin = (0.85 inches)*(182 PS/inch)/(47 ohms) = 3.87 PF

Cload = 2*(Cin STRAM + Cin Tab) + 1*(Cetch pin-pin) = 10.1 PF

Rs = 67 ohms Max
Rline = 3.0 ohms/inch * 7 inches = 21 ohms

Rtotal = Rs + Rline = 21 + 67 = 88 ohms

Min STRAM Clock PW >= 4*Rtotal*Cload = 4*(88.0 Ohms)*(10.1 PF)=
>= 3.55 NS

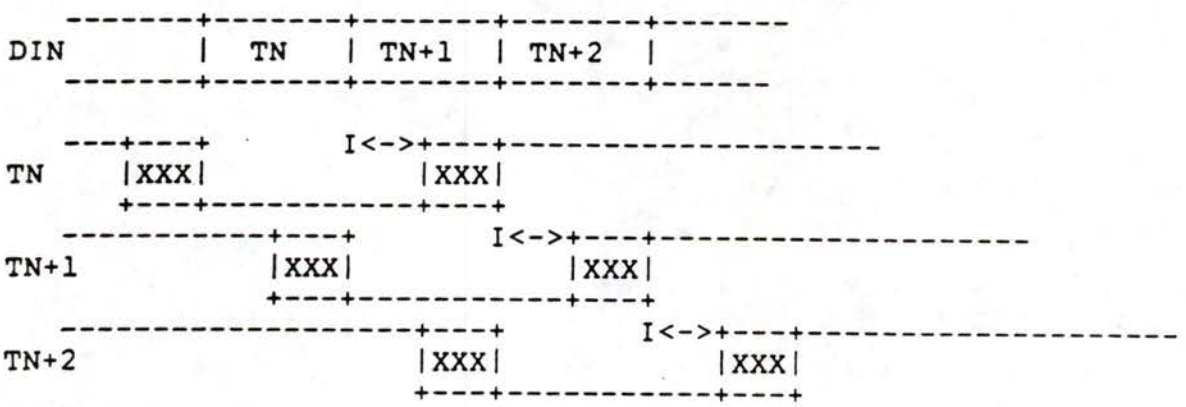
Min Oscillator Period >= 3.55/2= 1.78 NS

Note that the above pulse width is greater than the 3.0 NS minimum pulse with that the STRAM requires for correct operation.

B.0.0.5 Minimum Separation And Low State Width Of STRAM Clocks -

In order to always be able to pick a STRAM clock no matter at what time in the cycle the inputs to a STRAM become valid the separation and width of the STRAM clocks must satisfy certain constraints.

The constraints that must be satisfied are: the separation between different STRAM Clock phases must be greater than the sum of the clock skew plus the setup time of the STRAM, STRAM Clock pulses must be at least twice this width in order to guarantee that the latch is able to accept data for a period of time equal to the STRAM Clock pulse separation while operating the latch with an optimal width clock pulse so that the delay through the latch will be minimized. This is illustrated in the below.



I<-> = Tsetup STRAM

|XXX| = Clock Skew

STRAM Pulse Separation >= Clock Skew + Tsetup STRAM =
= 1.9 + 0.8 = 2.7 ns
STRAM Pulse Width >=2*(Clock Skew + Tsetup STRAM)
=2*(2.7) = 5.4 ns

A further constraint on the STRAM Clock pulses is that they must be formed off of the outputs of shift registers. This means that the pulse width and separation must be in multiples of one "N'th" of a machine cycle if a machine cycle is divided into "N" parts. Therefore, the minimum clock separation is Tcyc/N seconds, and the minimum STRAM Clock pulse width is 2*(Tcyc/N) seconds. This means the minimum clock oscillator period is 2.7 ns.

B.0.0.6 Minimum Clock Period And Pulse Width Summary -

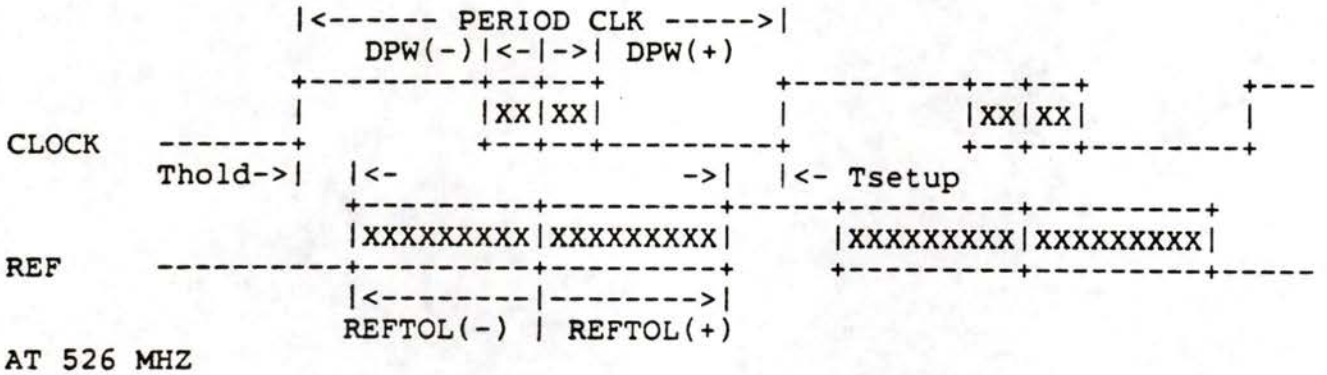
From the above analysis it can be seen that the minimum oscillator period must be greater than 2.7 NS. This corresponds to an oscillator maximum frequency of 370 MHZ. The STRAM Clock Low state must be greater than 5.4 ns and the STRAM clocks should be separated by 1/2 the width of the STRAM Clock Low state.

B.0.1 Clock Reference Positioning Accuracy

The Clock Reference signals rising and falling edges are aligned with the falling edges of the Clock at the Clock Module. In the clock shaping logic on the Clock distribution Chip and on the Gate Arrays the edges of the Clock Reference and the Clock must maintain a certain tolerance in order to guarantee correct operation of the Clock shaping logic.

In the Clock shaping logic the Clock is connected to the Clock input of a flip flop and the Clock Reference to the data input of the flip flop. For correct operation the position of the Clock Reference relative to the Clock must not violate the setup and hold time specifications of this flip flop. For this to be true the following two equations must be satisfied.

CLOCK REFERENCE POSITION TIMING DIAGRAM



AT 526 MHZ

REFTOL(-) = (PERIOD CLK) - Thold = 2.7-0.0 = 2.7 NS

REFTOL(+) = (PERIOD CLK) - TSETUP = 2.7-0.8 = 1.9 NS

NOTE: REFTOL INCLUDES ANY TOLERANCES IN THE POSITION OF THE FALLING EDGE OF THE CLOCK.

FIGURE 4

The relative position of the Clock and Clock Reference as seen by the Clock Distribution Chip and Clock logic on the Gate Arrays should not change when the Clock Oscillators frequency is changed. This can only happen if the Clock and Clock Reference pass through equal delay paths from the common point at which both originate.

Note, however, that the Clock Reference can be delayed any integer number of Clock cycles without affecting this delay because the Clock Reference is generated off of the edges of the Clock which is always running.

G Register File Specification

T. B. D.

E STRAM Specification

NOTE THIS VERSION OF THE SPEC INCLUDES CHANGES RESULTING FROM DEC'S VISIT TO JAPAN IN FEB 1986, AND ANY SUBSEQUENT MEETINGS AND LETTERS. IT IS BEING REVISED IN JUNE 1986.

SPEC CHANGES

PAGE	DESCRIPTION OF CHANGE
COVER	Revision number and date
1	Eliminate section 3.7.
2	Sections 1.2 and 1.4
5	Delete A-PS-2100002-GS, DELETE X-XX-XXXXXXX, add A-PS-2100013-GS. Section 3.1.1, all paragraphs.
6	Section 3.1.4.
7	Section 3.4.2, section 3.6, and section 3.7.
8	Section 3.8, added reference to marking location.
9	Section 3.8.2, changed referenced specification
10	Add low end to temperature range. Delete reference to storage. Add note 5.
11	Change temperature range, and supply voltage range.
13	Delete former note 2.
16	Add resistor RK.
20	Change VBB
21	Change tracking rate and IIH for clk. ADD RK.
22	Change tracking rates and temperature range
23	Changes in AC parameters
24	Changes in AC parameters
25	Change note 4. Added notes on Rin, Vee tolerence, and temperature range.
27	Change temperatures.
?	Change copper and polyimide thickness.
-	Add figure V, shows input waveform.
33	Revised connection diagram, added Rk and proposed change in VCC and VEE locations
34	Revised connection diagram, added Rk and proposed change in VCC and VEE locations

SOME MCU PROCESS TEMPERATURES ARE UNKNOWN, THUS THE TERM TBD APPEARS ON PAGES 5 AND 27.

DIGITAL EQUIPMENT CORPORATION

Marlborough, Massachusetts

PRELIMINARY SPECIFICATION
---------- -------------

ENGINEERING SPECIFICATION

for

Self-Timed RAM (STRAM)

AUTHORS: Paul M. Guglielmi
John J. Kelly

Revision 7

18 JUNE, 1986

TABLE OF CONTENTS

1.0 GENERAL DESCRIPTION

1.1 OVERVIEW

This specification describes a family of synchronous, self-timed, static, random access memory devices, STRAMS. The STRAM is similar to the traditional RAM in that it has chip select, input address and data, and output data. However, the design of the STRAM includes several non-traditional input signals, such as, write, a differential clock, and a reference voltage.

The core structure of the STRAM includes decoders, memory cell array, and read/write circuitry. The core is surrounded by latches which store address, data-in, control signals, and data out. During a write operation, an internal pulse generator, which is driven by the external differential clock signals, causes information stored in the data-in latches to be written to the array, and the output latches. During a read operation, the selected word is stored in the output latches. See Figures I and II.

The output circuit configuration provides the option of employing a series, or shunt terminating scheme. In addition, the outputs may be wire-or connected.

The STRAM shall be provided as TABBED, passivated die, mounted on a slide carrier.

1.2 ARCHITECTURE

There are two STRAM architectures defined in this specification, the smaller is 1K X 4 bits, and the larger is 4K X 4 bits. The two designs are the same except for number of address pins and some propagation delays.

1.3 CLOCK

The two clock input pins allow the clock signal to be supplied from a differential source. For those applications where the clock must be distributed in a single ended manner, an internal reference voltage, VBB, is available to replace the complimentary clock input signal.

1.4 OUTPUT CONFIGURATION

Figure III (A) shows a typical output, with its associated current source, and series termination resistor. The current source provides a low level drive source when series termination is being used. The termination resistor absorbs reflections from the load at the far end of the net.

Figure III (B) shows how these components would be wired when source series termination is being used.

1.4 OUTPUT CONFIGURATION (Continued)

Figure III (C) shows how these components would be wired when far end shunt (parallel) termination is used.

Figure III (D) shows how these components would be wired when a Wire-Or of several outputs is needed and source series termination is being used.

1.5 OPERATION

Data and address flows to the memory array through input latches. The WRITE signal determines whether the STRAM does a read or write operation, while the clock initiates the cycles.

Besides pinning, the principal difference between the STRAM and its traditional counterpart is the assertion of the inputs and valid time of the outputs with respect to Clock, and the pulse width requirements of the clock.

Basically, the STRAM is fully self-timed. Write operations are initiated by the rising edge of CLK H. The CLK H high state must be of sufficient duration to allow the new word to be written to the memory array. Read operations are initiated on the falling edge of the clock, however, the output data does not change until the CLK H rising edge.

Internally, Address (ADR), Data-In (DIN), WRITE, and Chip-Select (CS) are passed through the STRAM input latches when CLK H is low, and are held by the input latches while CLK H is High. The Data-Out (DO) latches are held when CLK H is Low, and become transparent when CLK H is High.

In order for the STRAM to work properly, control pulses for writing to the array must be generated and be precisely delayed from the Clock rising edge. For example, assertion of write pulse should occur just after Address (ADR) and Data-In (DIN) are valid at a memory cell. The circuitry for forming this write pulse is shown in both Figures I and II. The row delay line in Figure I can be implemented by a chain of gates. Its purpose is to provide a delay that is slightly longer than the delay incurred in sending the address through the address decoder and driving a row of RAM cells.

The circuit in Figure II shows how a write pulse of the required width might be formed. This circuit works where relatively wide pulses are required. If a narrow pulse is required then other, more simple, schemes can be used.

Operation of the STRAM is show in the timing diagrams, Figures IV, X, XI, and XII. A read operation proceeds as follows. When CLK H goes low, the Data-In (DIN), Address (ADR), Chip-Select(CS), and WRITE latches open to allow propagation of these signals to the storage array, without waiting for the CLK H rising edge. At at same time, the Data-Out (DO) latches close, to hold the read data from the previous cycle. When CLK H goes high, all inputs are latched, and the Data-Out(DO) register latches are opened. This allows new read data to propagate to the outputs. The previous read data that was being held in the Data-Out (DO) latches is overwritten.

1.5 OPERATION (Continued)

The timing of the RAM Data-Out (DO)'s depends on the width of the clock, CLK PW L. Two cases are shown in Figure XII, and described below.

If CLK H has a low state whose width is less than the memory array access time, then the Data-Out (DO)'s become valid a storage array access ti TACCmax, after the falling edge of the clock.

If CLK H has a wide low state, that is the width of the CLK H low state is greater than the access time of the memory storage array, then the Data-Out (DO) register will be loaded on the rising edge of the clock. The Data-Out (DO)'s will become valid a latch and an output cell delay, TDRmax, after the CLK H rising edge. The data will remain valid until the rising edge of CLK H in the next cycle.

If Chip-Select (CS) is false on the CLK H rising edge, and a read operation is specified then a low will be loaded into the Data-Out (DO) register.

If Chip-Select (CS) is true on the CLK H rising edge, and a read operation is specified, then the read data from the memory array will be loaded into the Data-Out (DO) latches.

The Write operation occurs during the period of CLK H High, only if WRITE has been asserted prior to the CLK H transition from low to high. The High state of CLK H must be of sufficient duration to allow for the worst case storage cell writing time, including the tolerance on the write strobe generation.

During a write cycle, the data being written into the STRAM appears at its outputs when CLK H goes high, provided that Chip-Select (CS) is asserted prior to the CLK H transition from low to high. If Chip-Select (CS) is fals when CLK H transitions from low to high, and a write operation is specified, then the write operation will not be performed and the Data-Out (DO) latches will be loaded with a logic low.

1.6 FUNCTIONAL COMPATIBILITY

The DC characteristics of the input and output signals of the STRAM are specified to be compatible with the industry standard 100K ECL family of devices. The power supply, however, is specified at negative 5.2 volts, similar to 10K ECL.

2.0 APPLICABLE DOCUMENTS (PER LATEST REVISION ON DATE OF ORDER)

Digital Equipment Corporation:

A-PS-21-00013-GS	General Specification for Custom Integrated Circuits (Note: This document includes the Qualification Requirements for Mechanical Testing of TABBED Semiconductor Packages)
Y-YY-YYYYYYYY	Specification for Slide Carrier *
Z-ZZ-ZZZZZZZZ	Specification for Tab Lead Frame *

* Indicates document to be written.

Military Standards:

MIL-STD-883	Test Methods and Procedures for Microelectronics

3.0 REQUIREMENTS

The item described herein shall meet the applicable requirements of Digital Specifications A-PS-21-00013-GS, and parameters specified herein. In the event of a conflict, the requirements of this document shall take precedence.

3.1 MECHANICAL

3.1.1 General Packaging Requirements:

STRAMS shall be provided as TABBED, passivated, and encapsulated die, mounted on a slide carrier.

Die bonding pads shall be provided with bumps of gold, or alloy to accommodate connection of the TAB leads to the die. In order to inhibit corrosion and electro-migration in a non-hermetic environment, the chip surface, including the TAB lead to bonding pad interface, shall be coated with an encapsulation material which is to be specified by the vendor, and approved by DEC. Any other die coating, such as polyimide for alpha particle shielding, shall also be specified by the vendor, and approved by DEC.

The die and TAB assembly must withstand TBD deg C for 10 minutes without degradation.

3.1 MECHANICAL (Continued)

3.1.2 Configuration and Dimensions:

TAB lead frame shall provide connection for the bonding pads shown in Connection Diagrams, see Figures VI and VII. Exact placement of pad locations and other physical dimensions are to be determined by DEC after negotiations with vendors.

3.1.3 Die Mechanical:

Die thickness shall be 0.500 mm, +/- 0.025 mm.

Planar dimensions of die are to be determined by DEC after negotiations with vendors.

3.1.4 Die Attach Method:

The die will be epoxied to a substrate for cooling. The back of the die will be polished to a 1.5 micron finish, and the flatness will be less than 10.0 micron. The back of the die will be electrically isolated from the substrate.

3.2 ELECTRICAL

3.2.1 Absolute Maximum Ratings: Per Table I On Page 10.

3.2.2 Operating Conditions: Per Table II On Page 11.

3.2.3 Pin Signal Definitions: Per Table III On Pages 12 and 13.

3.2.4 Parameter Definitions: Per Table IV On Pages 14 - 18.

3.2.5 DC Characteristics: Per Table V On Pages 19 - 22.

3.2.6 AC Characteristics: Per Tables VI and VII On Pages 23 - 25.

3.2.7 Truth Table: Per Table VIII On Page 26.

3.2.8 Logic Symbol: Per Figures VIII-A, AND VIII-B On Pages 35 - 36.

3.2.9 Functional Block Diagram: Per Figure IX on Page 37.

3.3 ENVIRONMENTAL

The die and TAB assembly shall withstand the temperature and humidity conditions described in Table IX.

3.4 RELIABILITY

3.4.1 Hard failure rate - Less than 40 FITS at a junction temperature of 85 deg C., and dew point of 35 deg C.

3.4.2 Soft failure rate - Less than 300 FITS over the range of temperatures and voltages specified in Table II.

3.5 MANUFACTURING PROCESS COMPATIBILITY

Manufacturing process compatibility requirements are to be determined.

3.6 MARKING

3.6.1 DIE - Each die shall be marked with the vendor name or symbol, and part number with revision level.

3.6.2 TAB Lead Frame - Each tab lead frame shall be marked with the vendor name or symbol, date code, and DEC part number. This marking shall be located on the polyimide support ring, and shall consist of a 10 OCR-A alpha-numeric character field. Character size shall be 0.81 mm x 0.81 mm. A pin 1 identifier shall be an an integral part of the tab assembly.

3.6.3 Carrier - Each carrier shall also be marked with a pin 1 identifier. The die lead frame assembly shall be mounted in the carrier, and oriented so that the pin 1 identifier on the carrier agrees with the corresponding marker on the TAB.

3.7 PACKAGING AND SHIPPING

Each STRAM, along with it's TAB frame lead assembly, shall be packaged in an individual carrier which provides protection from mechanical damage, and electrostatic discharge. Carriers will conform to the requirements outlined in document Y-YY-YYYYYYY. (Note, document to be written.)

3.8 TAB LEAD FRAME

The chip shall be delivered on a TAB lead frame in a slide carrier. See Figures XV-A, XV-B, AND XVI. The TAB will be a one metal-layer lead frame with a polyimide insulating layer suitably adjoined. The insulat layer will be mounted on the top of the copper frame (furthest from the active surface of the device when the TAB is assembled). The TAB/device assembly shall include the following features:

Inner lead bond (ILB) - Copper lead will extend toward die, beyond the polyimide support ring, to be bonded to the die bonding pads. ILB metallurgy dependent on the bond process.

Lead fanout - Leads may fanout from fine pitched ILB to more coarse pitched OLB. A polyimide support ring is required in this area. Marking is located on the support ring in this area.

Outer lead bond (OLB) - The OLB is the part of the TAB which mates with the substrate. The OLB must match with specified pad locations, to be determined.

Excise area - The Tabbed STRAM is removed from the tape by a punching process called excising. The excise area is in the window immediately outside the OLB.

Test probe pads with polyimide support - The STRAM must be tested on the tape, before excising. An array of pads will be located outside the OLB for testing.

Slide carrier - The tape must be mounted on a standard slide carrier. DEC to provide critical dimensions.

3.8 TAB LEAD FRAME (Continued)

3.8.1 TAB Frame Mechanical Design :

The mechanical requirements of the TAB lead frame are shown in Table X, and Figures XV-A, XV-B, and XVI.

3.8.2 Qualification Requirements and Methods:

The mechanical integrity of the TAB-mounted STRAM will undergo qualification testing as outlined in document A-PS-21-00013-GS.

3.8.3 Tape Format:

The TAB tape format will comply with the features and dimensions indicated in Figures XV-A, XV-B, and XVI. In addition, it must be compatible with the slide carrier described in Section 3.7.

4.0 QUALITY ASSURANCE PROVISIONS

Per Digital Specifications A-PS-21-00013-GS.

TABLE I

ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
DESCRIPTION	SYMBOL		
Supply Voltage	VEE	+0.5 to -7.0	Vdc
Input Voltage	VIN	+0.5 to VEE	Vdc
Output Current, Series Terminated			
Continuous	IO(S)	30	mA
Surge (Pulsed)	IO(S)	100	mA
Output Current, Parallel Terminated			
Continuous	IO(P)	30	mA
Surge (Pulsed)	IO(P)	100	mA
Temperature Range Operating	Tj	0 - 115	deg C

Notes for Table I:

1. Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only. Functional operation of the device at these or any other conditions beyond those indicated in Tables II and V is not implied.

2. All voltages are specified with respect to Network Ground, VCC.

3. The device must survive any combination of voltages and temperature within the ratings indicated above. As indicated in the table, the input voltage must not be allowed to go more negative than VEE.

4. Also see Table IX, Environmental Requirements.

5. For surge conditions, pulse width is 10uSec, duty cycle is 1%.

TABLE II

REQUIRED OPERATING CONDITIONS

PARAMETER		RATING	UNIT
DESCRIPTION	SYMBOL		
Supply Voltage	VEE	-5.62 to -4.78	Vdc
Temperature	Tj	20 to 90	deg C

Notes for Table II:

1. Supply voltage is specified with respect to network ground, VCC.

2. No backside electrical contact will be guaranteed, thus, the substrate connection to VEE must be provided by a topside electrical contact.

3. Prolonged operation of the device outside the limits specified in Table II may affect device reliability.

4. Also see Table IX, Environmental Requirements.

TABLE III

PIN SIGNAL DEFINITIONS

SIGNAL PINS:

Signal Name	Assert Level	Definition
ADR <I>	H	One of M address input lines
DIN <I>	H	One of 4 data input lines
CS	L	Chip select
CLK H	H	Clock signal input, initiates read and write operations
CLK L	L	Compliment of clock signal input, used for differential clock operation
VBB	-	Internally generated reference voltage, used in place of CLK L in single-ended clock systems
WRITE	L	When asserted, allows writing new word into array and output registers. When un-asserted, inhibits writing new word to array
DO(ST)	<I>	H One of four outputs, used in series termination configuration
DO(PT)	<I>	H One of four outputs, used in shunt (parallel) termination configuration
DO(CS)	<I>	- Connection to one of four constant current sources

TABLE III

PIN SIGNAL DEFINITIONS (Continued)

WER/GROUND PINS:

NAME	VALUE	Description
VCC	GND	Network ground for array and logic.
VCC(O)	GND	Network ground for output circuits.
VEE	-5.2V	Power Supply

Notes for Table III

1. Assertion Level.

   High (H) assertion signals are true, or asserted, for high level voltages as specified in Table V, and false, or negated, for low level voltages as specified in Table V. Low (L) assertion signals are true, or asserted, for low level voltages and false ,or negated, for high level voltages.

2. Unused Inputs.

   Unused signal inputs will be permanently connected to a power supply, set at -3.4V, in order to provide proper logical operation of the device. Such connections must not effect the long term operation or reliability of the device.

3. Unused DO(CS) Connections.

   Any unused DO(CS) connections must be permanently connected to VCC, network ground.

TABLE IV

PARAMETER DEFINITIONS

SYMBOL	NAME/DEFINITION
VIH	HIGH LEVEL INPUT VOLTAGE An input voltage level within the more positive (less negative) of the two ranges of values used to represent the binary variables.
VIL	LOW LEVEL INPUT VOLTAGE An input voltage level within the less positive (more negative) of the two ranges of values used to represent the binary variables.
VOH	HIGH LEVEL OUTPUT VOLTAGE The voltage at an output terminal with input conditions applied that, according to the specification, will establish a high level at the output.
VOL	LOW LEVEL OUTPUT VOLTAGE The voltage at the output terminal with input conditions applied that, according to the specification, will establish a low level at the output.
VI(H)Max	The most positive gate input level which will occur under normal operating conditions.
VI(H)Min	Positive edge of the transition region. Input voltages equal to or more positive than VI(H)Min will produce an output level which is equal to or more positive than VO(H)C, for a non-inverting gate, or more negative than VO(L)C for an inverting gate. If a gate input is taken more negative than VI(H)Min, the output level shall begin to move towards VO(L) for a non-inverting gate, or VO(H) for an inverting gate.
VI(L)Max	Negative edge of the transition region. Input voltages equal to or more negative than VI(L)Max will produce an output level which is equal to or more negative than VO(L)C, for a non-inverting gate, or more positive than VO(H)C for an inverting gate. If a gate input is taken more positive than VI(L)Max, the output level shall begin to move towards VO(H) for a non-inverting gate, or VO(L) for an inverting gate.

TABLE IV:

PARAMETER DEFINITIONS (Continued)

SYMBOL	NAME/DEFINITION
VI(L)Min	The most negative gate input level which will occur under normal operation.
VO(H)Max	The most positive output level which will occur with an input voltage equal to VI(H)Max, for a non-inverting gate, or VI(L)Min, for an inverting gate.
VO(H)Min	The least positive output level which will occur with an input voltage equal to VI(H)Max, for a non-inverting gate, or VI(L)Min, for an inverting gate.
VO(H)C	The most negative high logic level which will occur at a gate output with an input voltage equal to VI(H)Min, for a non-inverting gate, or VI(L)Max, for an inverting gate.
VO(L)C	The most positive low logic level which will occur at a gate output with an input voltage equal to VI(L)Max, for a non-inverting gate, or VI(H)Min, for an inverting gate.
VO(L)Max	The most positive output level which will occur with a voltage input equal to VI(L)Min, for a non-inverting gate, or VI(H)Max, for an inverting gate.
VO(L)Min	The most negative output level which will occur with an input voltage equal to VI(L)Min, for a non-inverting gate, or VI(H)Max, for an inverting gate.
IIH	HIGH LEVEL INPUT CURRENT The current into an input when a high level voltage is applied to that input.
IIL	LOW LEVEL INPUT CURRENT The current into an input when a low level voltage is applied to that input.

TABLE IV:

PARAMETER DEFINITIONS (Continued)

SYMBOL	NAME/DEFINITION
IEE	POWER SUPPLY CURRENT The current into the VEE supply terminal of the device.
IO(S)H	HIGH LEVEL OUTPUT CURRENT Output current in series terminated configuration.
IO(S)L	LOW LEVEL OUTPUT CURRENT Output current in series terminated configuration.
IO(P)H	HIGH LEVEL OUTPUT CURRENT Output current in parallel terminated configuration.
IO(P)L	LOW LEVEL OUTPUT CURRENT Output current in parallel terminated configuration.
ICS	PULL DOWN SOURCE CURRENT A constant current, flowing into the current source.
IBB	VBB REFERENCE CURRENT Current into the VBB reference terminal.
PWD	POWER DENSITY Power dissipated in the die, per unit area.
RS	SOURCE TERMINATION RESISTOR Value of resistor as measured between DO(PT) and DO(ST).
tR	RISE-TIME The time between a specified low-level voltage and a specified high-level voltage on a waveform that is changing from the low level to the high level.
tF	FALL-TIME The time between a specified high-level voltage and a specified low-level voltage on a waveform that is changing from the high level to the low level.
RK	CLOCK TERMINATION RESISTOR Value of resistor as measured between RK(1) and RK(2).

TABLE IV:

PARAMETER DEFINITIONS (Continued)

SYMBOL	NAME/DEFINITION
CIN	INPUT CAPACITANCE The capacitance measured at the specified Package pins with power applied to the device.
RIN	REAL INPUT IMPEDANCE The real portion of the input impedance measured at the specified package pins with power applied to the device.
CLK PW H	CLOCK PULSE WIDTH HIGH The portion of time during which the input clock signal (CLK H) is within the VIH range.
CLK PW L	CLOCK PULSE WIDTH LOW The portion of time during which the input clock signal (CLK H) is within the VIL range.
TSA	ADDRESS SET UP TIME Period of time, during which address (ADR) must be true, prior to low to high transition of CLK H.
TSDI	DATA SET UP TIME Period of time, during which data (DIN) in must be true, prior to low to high transition of CLK H.
TSW	WRITE SET UP TIME Period of time, during which WRITE must be true, prior to low to high transition of CLK H.
TSCS	CHIP SELECT SET UP TIME Period of time, during which chip select (CS) must be true, prior to low to high transition of CLK H.
THA	ADDRESS HOLD TIME Period of time, during which ADR must be held true, after low to high transition of CLK H.
THDI	DATA HOLD TIME Period of time, during which DIN must be held true, after low to high transition of CLK H.

TABLE IV:

PARAMETER DEFINITIONS (Continued)

SYMBOL	NAME/DEFINITION
TSW	WRITE HOLD TIME Period of time, during which WRITE must be held true, after low to high transition of CLK H.
THCS	CHIP SELECT HOLD TIME Period of time, during which CS must be held true, after low to high transition of CLK H.
TCYC	CYCLE TIME Minimum cycle time which results in reliable operation of the STRAM. TCYC is the sum of CLK PW H and CLK PW L.
TDR	OUTPUT DELAY Delay form rising edge of CLK H to output. (This parameter defines the delay through the clock buffer and clock to output path of the output latch.)
TACC ADR	ACCESS TIME FROM ADDRESS Propagation delay from ADR transition to output, with TSA minimum. (This parameter defines the delay through input latch, the memory array, and output latch.)
TACC W	ACCESS TIME FROM WRITE Propagation delay from WRITE transition to output with TSW minimum. (This parameter defines the delay through input latch, output mux, and output latch.)
TACC CS	ACCESS TIME FROM CHIP SELECT Propagation delay from CS transition to output with TSCS minimum. (This parameter defines the delay through input latch, output mux, and output latch.)
TACC CLK	ACCESS TIME FROM CLOCK Propagation delay from falling edge of CLK H to output, with CLK PW L at minimum value. (This parameter defines the delay through the clock buffer, clock to output path of input latch, memory array, output mux and output latch.)
TACC DIN	ACCESS TIME FROM DATA IN Propagation delay from DIN to output with TSDIN minimum. (This parameter defines the delay through input latch, output mux, and output latch.)

TABLE V

DC CHARACTERISTICS

PARAMETER		TEST CONDITION	REQUIREMENTS		
NAME	SYMBOL	REFER TO NOTES 1 THROUGH 6	MIN	MAX	UNIT
HIGH LEVEL OUTPUT VOLTAGE All outputs including wire-or configuration with one output in Logic High State	VO(H)	VIN = VI(H)Max or VI(L)Min IO(S)H = -10uA (See Fig. XIII)	-1025	-880	mV
		With DO(PT) terminated to -2V thru 50 ohms (See Fig. XIV)	-1025	-880	mV
LOW LEVEL OUTPUT VOLTAGE All outputs including wire-or configuration with all outputs in Logic LOW State	VO(L)	VIN = VI(L)Min or VI(H)Max IO(S)L =-10uA (See Fig. XIII)	-1810	-1620	mV
		With DO(PT) terminated to -2V thru 50 ohms (See Fig. XIV)	-1810	-1620	
HIGH LEVEL OUTPUT THRESHOLD VOLTAGE All outputs including wire-or configuration with one output in Logic HIGH State	VO(H)C	VIN = VI(H)MIN or VI(L)MAX IO(S)H = -10uA (See Fig. XIII)	-1035		mV
		With DO(PT) terminated to -2V thru 50 ohms (See Fig. XIV)	-1035		mV

TABLE V

DC CHARACTERISTICS (Continued)

PARAMETER		TEST CONDITION	REQUIREMENTS		
NAME	SYMBOL	REFER TO NOTES 1 THROUGH 6	MIN	MAX	UNIT
LOW LEVEL OUTPUT THRESHOLD VOLTAGE All outputs including wire-or configuration with all outputs in Logic LOW State	VO(L)C	VIN = VI(L)MAX or VI(H)MIN IO(S)L = -10uA (See Fig. XIII)		-1610	mV
		With DO(PT) terminated to -2V thru 50 ohms (See Fig. XIV)		-1610	mV
HIGH LEVEL INPUT VOLTAGE	VI(H)		-1165	-880	mV
LOW LEVEL INPUT VOLTAGE	VI(L)		-1810	-1475	mV
REFERENCE VOLTAGE	VBB	VBB Terminal connected to -5.2V, through 78K ohms.	-1370	-1270	mV
LOW LEVEL INPUT CURRENT	II(L)	VIN = VIL(MIN)	-0.5	50	uA
HIGH LEVEL INPUT CURRENT	II(H)	VIN = VIH(MAX)	0	50	uA
HIGH LEVEL INPUT CURRENT ($\overline{CLK}$ AND $\overline{CLK}$ ONLY)	II(H)	VIN = VIH(MAX)	0	220	uA

TABLE V

DC CHARACTERISTICS (Continued)

PARAMETER		TEST CONDITION	REQUIREMENTS		
NAME	SYMBOL	REFER TO NOTES 1 THROUGH 6	MIN	MAX	UNIT
OUTPUT EMITTER FOLLOWER LEAKAGE CURRENT	IOEF	VCC = Open VEE = Open VCC(O) = 2V Output = 0V Inputs = Open Note: Only Output Under Test To 0V, All Others Open	-0.5	+10.0	uA
POWER SUPPLY CURRENT	IEE	VIN = VI(H)MAX, All Inputs		-465	mA
PULL DOWN SOURCE CURRENT	ICS	DO(CS) = -1.30V	8.5	11.5	mA
POWER DENSITY	PWD	VIN = VI(H)MAX all inputs,		15.0	W/cm*2
SOURCE TERMINATION RESISTANCE	RS	Measured between DO(PT) - DO(ST) with I(RS)=20uA	25.5	34.5	ohms
CLOCK TERMINATION RESISTANCE	RK	Measured between RK(1) - RK(2) with I(RK)=10mA	96.0	144.0	ohms
REFERENCE VOLTAGE TRACKING RATE /dVEE	dVBB	VBB Terminal connected to -5.2V, through 78K ohms.	0	+25	mV/V

TABLE V

DC CHARACTERISTICS (Continued)

PARAMETER		TEST CONDITION	REQUIREMENTS		
NAME	SYMBOL	REFER TO NOTES 1 THROUGH 6	MIN	MAX	UNIT
OUTPUT VOLTAGE TRACKING RATE	dVO(H) /dVEE	With DO(PT) connected to DO(CS) (See Fig. XIII)	0	+25	mV/V
	dVO(L) /dVEE	With DO(PT) connected to DO(CS) (See Fig. XIII)	0	+55	mV/V
	dVO(H) /dVEE	With DO(PT) terminated to -2V thru 50 ohms (See Fig. XIV)	0	+25	mV/V
	dVO(L) /dVEE	With DO(PT) terminated to -2V thru 50 ohms (See Fig. XIV)	0	+55	mV/V

Notes for Table V

1. Parameters specified for Tj = 20 degrees Celsius to 90 degrees Celsius, and VEE = -5.2V +/-10mV, unless otherwise specified. Parameter values over the VEE range are defined by the voltage tracking rates.

2. All voltages referenced to the VCC terminal.

3. Unless otherwise specified, VEE is connected to -5.2V.

4. Unless otherwise specified, DO(CS) is connected to VCC level (ground).

5. Positive current flows into any terminal, negative current flows out of any terminal.

6. The Min/Max limits are algebraic quantities.

TABLE VI

AC CHARACTERISTICS - 1K X 4 STRAM

PARAMETER	NOTES:	REQUIREMENTS MIN	MAX	UNITS
CLK PW H	SEE FIGURE XI	7.0	---	ns
CLK PW L	SEE FIGURE XI	3.0	---	ns
TSA	SEE FIGURE X	0.4	---	ns
TSDI	SEE FIGURE X	0.4	---	ns
TSW	SEE FIGURE X	0.4	---	ns
TSCS	SEE FIGURE X	0.4	---	ns
THA	SEE FIGURE X	0.8	---	ns
THDI	SEE FIGURE X	0.8	---	ns
THW	SEE FIGURE X	0.8	---	ns
THCS	SEE FIGURE X	0.8	---	ns
TDR	SEE FIGURE XII	0.3	1.5	ns
TACC ADR	SEE FIGURE XII	---	6.0	ns
TACC W	SEE FIGURE XII	0.3	3.0	ns
TACC CS	SEE FIGURE XII	0.3	3.0	ns
TACC DIN	SEE FIGURE XII	0.3	3.0	ns
TACC CLK	SEE FIGURE XII	---	6.5	ns
TCYC	SEE FIGURE XI	10.0	---	ns
tR(out)		0.3	1.5	ns
tF(out)		0.3	1.5	ns
CIN		---	1.3	pf
CIN	CLK AND $\overline{\text{CLK}}$ ONLY	---	2.3	pf
RIN		Positive	---	ohm

See notes on page 25

TABLE VII

AC CHARACTERISTICS - 4K X 4 STRAM

PARAMETER	NOTES:	REQUIREMENTS MIN	REQUIREMENTS MAX	UNITS
CLK PW H	SEE FIGURE XI	10.5	---	ns
CLK PW L	SEE FIGURE XI	3.0	---	ns
TSA	SEE FIGURE X	0.4	---	ns
TSDI	SEE FIGURE X	0.4	---	ns
TSW	SEE FIGURE X	0.4	---	ns
TSCS	SEE FIGURE X	0.4	---	ns
THA	SEE FIGURE X	0.8	---	ns
THDI	SEE FIGURE X	0.8	---	ns
THW	SEE FIGURE X	0.8	---	ns
THCS	SEE FIGURE X	0.8	---	ns
TDR	SEE FIGURE XII	0.3	1.5	ns
TACC ADR	SEE FIGURE XII	---	10.5	ns
TACC W	SEE FIGURE XII	0.3	3.0	ns
TACC CS	SEE FIGURE XII	0.3	3.0	ns
TACC DIN	SEE FIGURE XII	0.3	3.0	ns
TACC CLK	SEE FIGURE XII	---	11.0	ns
TCYC	SEE FIGURE XI	13.5	---	ns
tR(out)		0.3	1.5	ns
tF(out)		0.3	1.5	ns
CIN		---	1.3	pf
CIN	CLK AND $\overline{\text{CLK}}$ ONLY	---	2.3	pf
RIN		Positive	---	ohm

See notes on page 25

Notes For AC Characteristics:

1. AC characteristics are measured from the mid point of the input waveform to the midpoint of the output waveform, except for tR and tF.

2. Rise and fall times are measured between the 20% point and the 80% point of the waveform.

3. Output waveforms are required to be monotonic.

4. AC parameters are defined with an input rise, or fall time of 700 ps, see Figure V.

5. CIN is measured at the die bonding pad. Tab lead capacitance is not included in the value specified in Tables VI and VII. Measurement technique to be defined.

6. RIN, the real portion of the input impedance, measured at each input pin, with power applied to the device, must be positive for frequencies up to 2.0 GHz, over the input voltage range. Measurement technique to be defined.

7. For AC parameters, assume output is configured in the series termination configuration. See Figure XIII.

8. AC parameters are defined with VEE = -5.2V +/- 8%, over the required operating temperature range.

TABLE VIII

STRAM TRUTH TABLE

INPUTS			
CHIP SELECT	WRITE	RAM ARRAY	OUTPUT REGISTER
H	H	READ	LOW
H	L	READ	LOW
L	H	READ	RAM
L	L	WRITE	DIN

KEY:

RAM = LOAD D-OUT REGISTER FROM ADDRESSED RAM LOCATION

DIN = LOAD D-OUT REGISTER FROM OUTPUT OF D-IN REGISTER

LOW = LOAD D-OUT REGISTER WITH A LOGIC LOW LEVEL

READ = READ FROM THE ADDRESSED LOCATION

WRITE = WRITE THE ADDRESSED LOCATION

NOTE:

WRITES ARE INITIATED BY THE RISING EDGE OF THE CLOCK IF THE ENABLING CONDITIONS ARE TRUE.

TABLE IX

ENVIRONMENTAL REQUIREMENTS

CONDITION	TEMPERATURE deg C.	HUMIDITY
Operating See Note 1.	$20 < T_j < 35$ $35 < T_j < 90$	95% R.H. Dew point = 35 deg C.
Storage See Note 2.	$-65 < T_a < 35$ $35 < T_a < TBD$	95% R.H Dew point = 35 deg C.
Assembly process See Note 3.	$T_s < TBD$	Dew point = 35 deg C.

Notes for Table IX:

1. The device shall continue to meet all DC and AC characteristics specified herein, when operated over this range of temperature/humidity conditions. Tj is junction temperature.

2. After prolonged storage at the conditions specified, device shall meet all DC and AC characteristics specified herein. Failure rate shall not be degraded by storage. Ta is ambient temperature.

3. During assembly process, the die and lead frame will survive contact with surfaces at the temperature specified for ten minutes maximum. Ts is surface temperature.

TABLE X

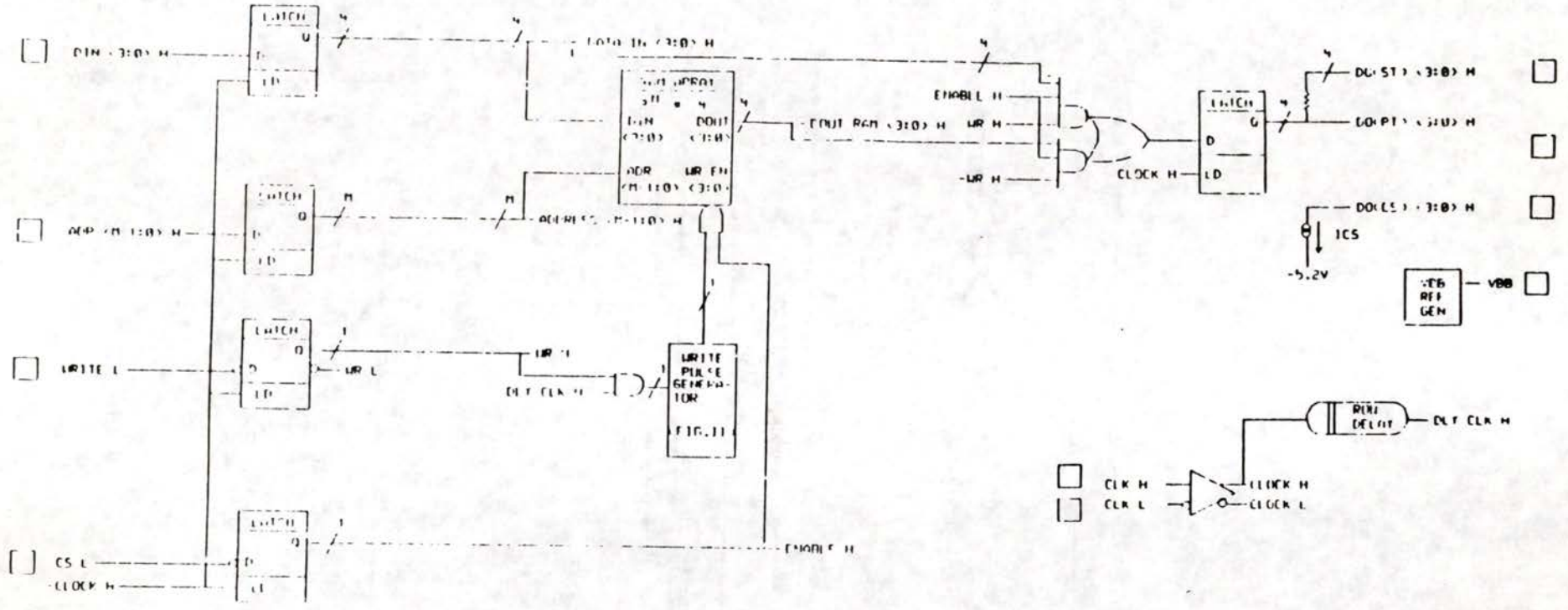
TAB LEAD FRAME MECHANICAL DIMENSIONS

PARAMETER	DIMENSIONS		
	Min	Max	Units
Copper Lead Thickness	See Note 1.		micron
Polyimide Thickness	See Note 2.		micron
Plating: Sn over Ni			
Sn	1.0	-	micron
Ni	1.0	-	micron
Inner Lead:			
Pitch	100	-	micron
Width	50	-	micron
Outer Lead:			
Pitch	450	-	micron
Width	90	110	micron
Length	750	-	micron
Excise/OLB window:			
Width	1000	-	micron
Test pad:			
Length	400	-	micron
Width	400	-	micron
Pitch	TBD	TBD	micron

1. Copper lead thickness is 35 micron nominal.

2. Vendors may supply tab lead frame with either 75 micron, or 125 micron (nominal) polyimide thicknesses.

3. Plating requirements may vary due to bonding and testing process variations.



SCHEMATIC DIAGRAM

FIGURE 1

NOTES: 1. CIRCLE ON AN INPUT OR OUTPUT MEANS THE SIGNAL IS TRUE WHEN AT A LOGIC LOW VOLTAGE LEVEL.

2. NO CIRCLE ON AN INPUT OR OUTPUT MEANS THE SIGNAL IS TRUE AT A LOGIC HIGH VOLTAGE LEVEL.

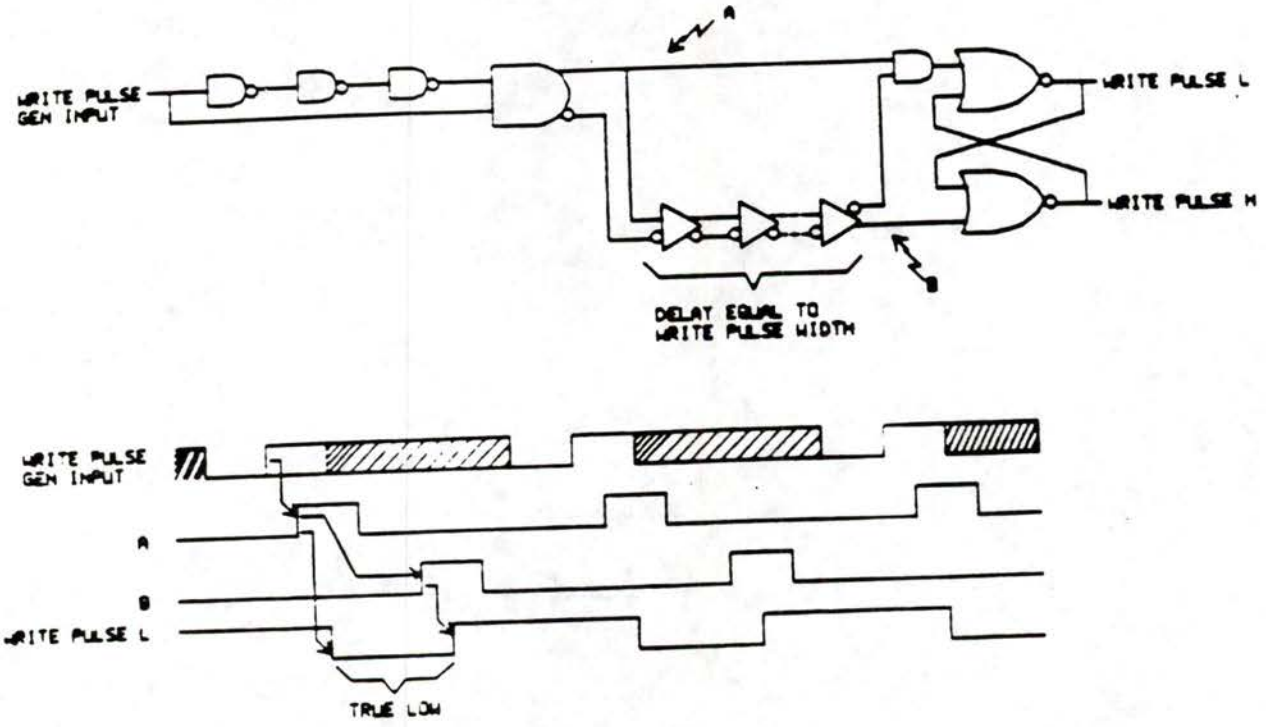
3. "-SIGNAL NAME H" IS THE SAME SIGNAL AS "SIGNAL NAME L".

4. "-SIGNAL NAME L" IS THE SAME SIGNAL AS "SIGNAL NAME H".

5. LATCHES ARE TRANSPARENT WHEN THE "LD" INPUT IS "HIGH".

6. ADDRESS, DIN, WRITE, AND CS LATCHES ARE TRANSPARENT WHEN "CLK H" IS "LOW".

7. "DO" LATCHES ARE TRANSPARENT WHEN "CLK H" IS "HIGH".

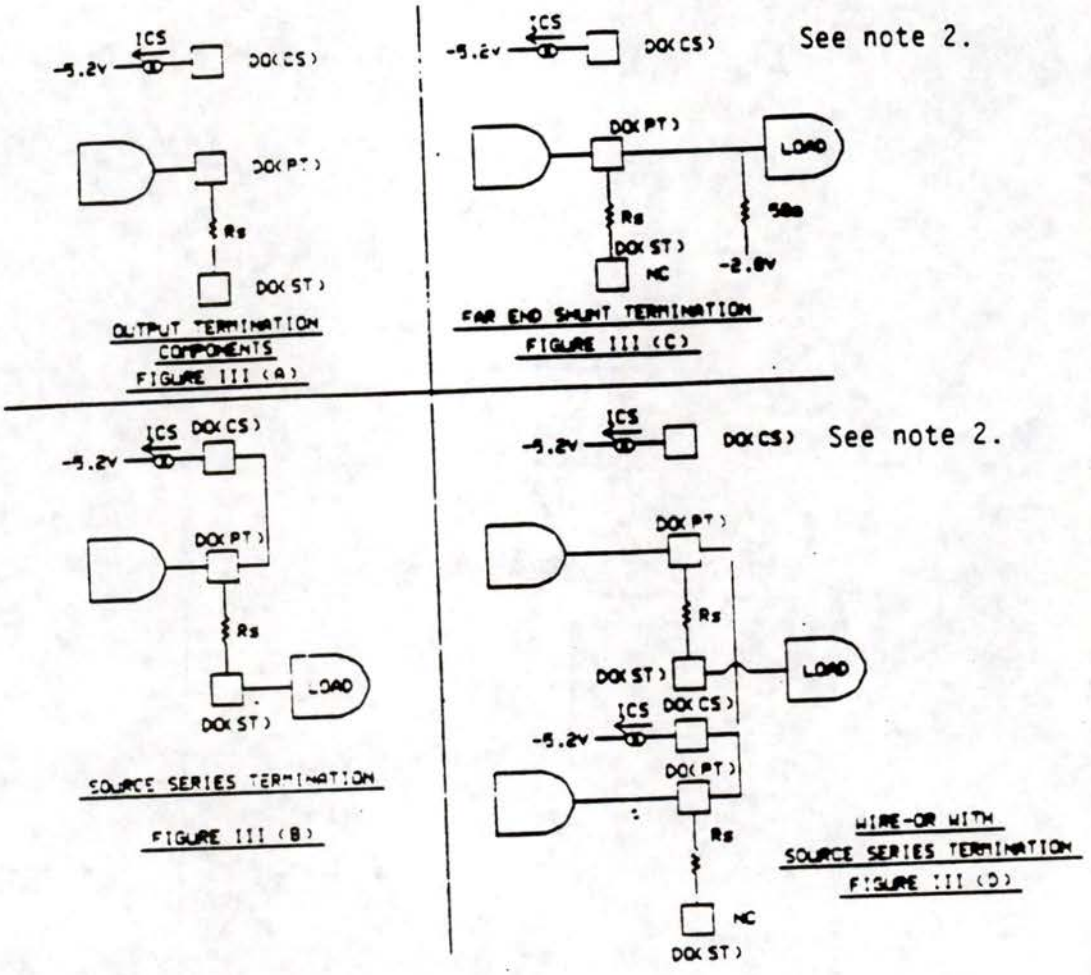


WRITE PULSE GENERATOR

FIGURE II

NOTE:

This schematic is provided as an example of one way to configure The Write Pulse generation.

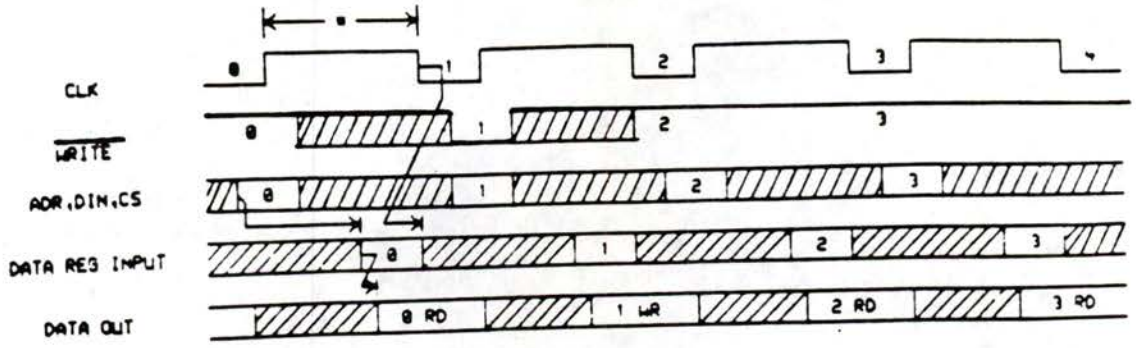


OUTPUT CIRCUIT DIAGRAM

FIGURE III

NOTES:

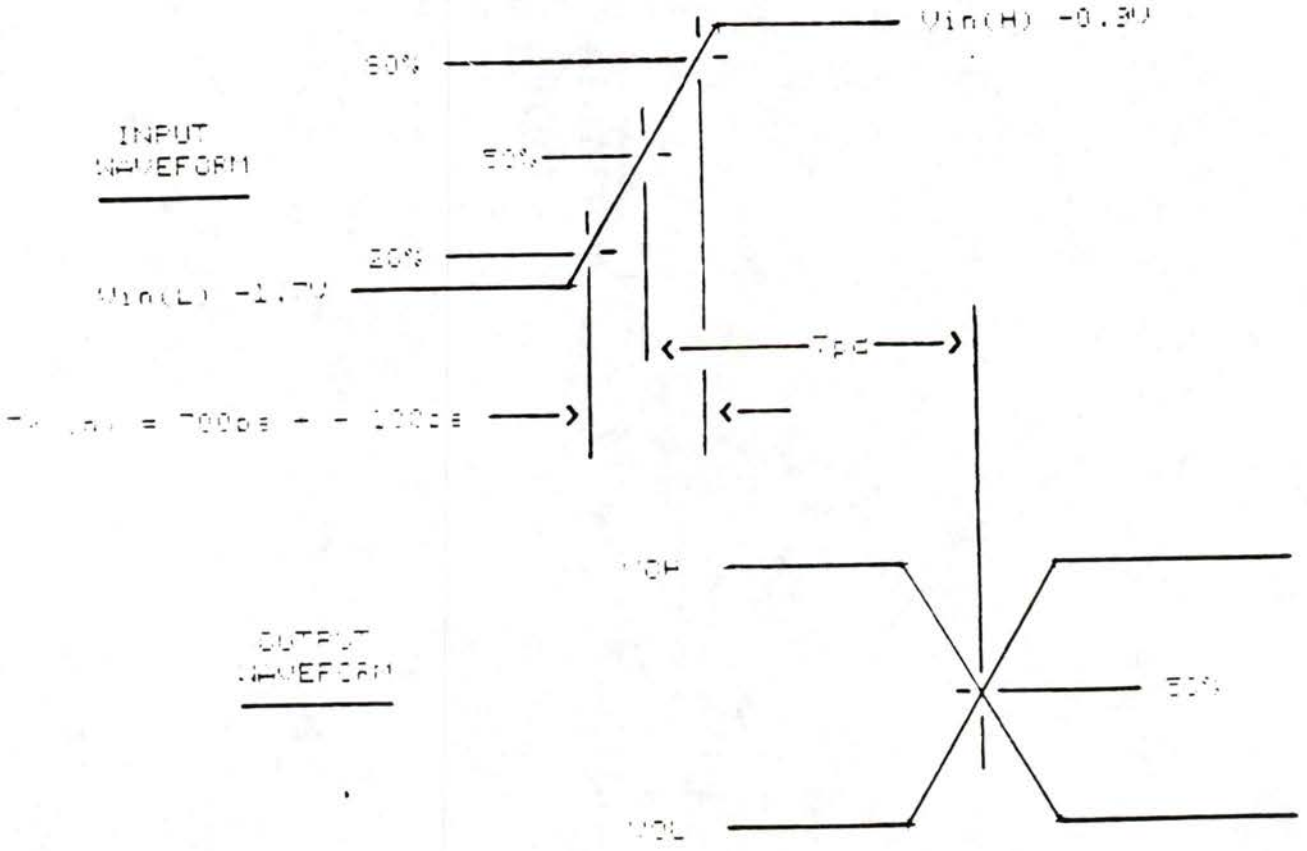
1. NC SHOWS NO CONNECTION.
2. UNUSED CURRENT SOURCE CONNECTIONS MUST BE CONNECTED TO GROUND.



* CLOCK HIGH STATE MUST LAST LONG ENOUGH TO COMPLETE A WRITE CYCLE

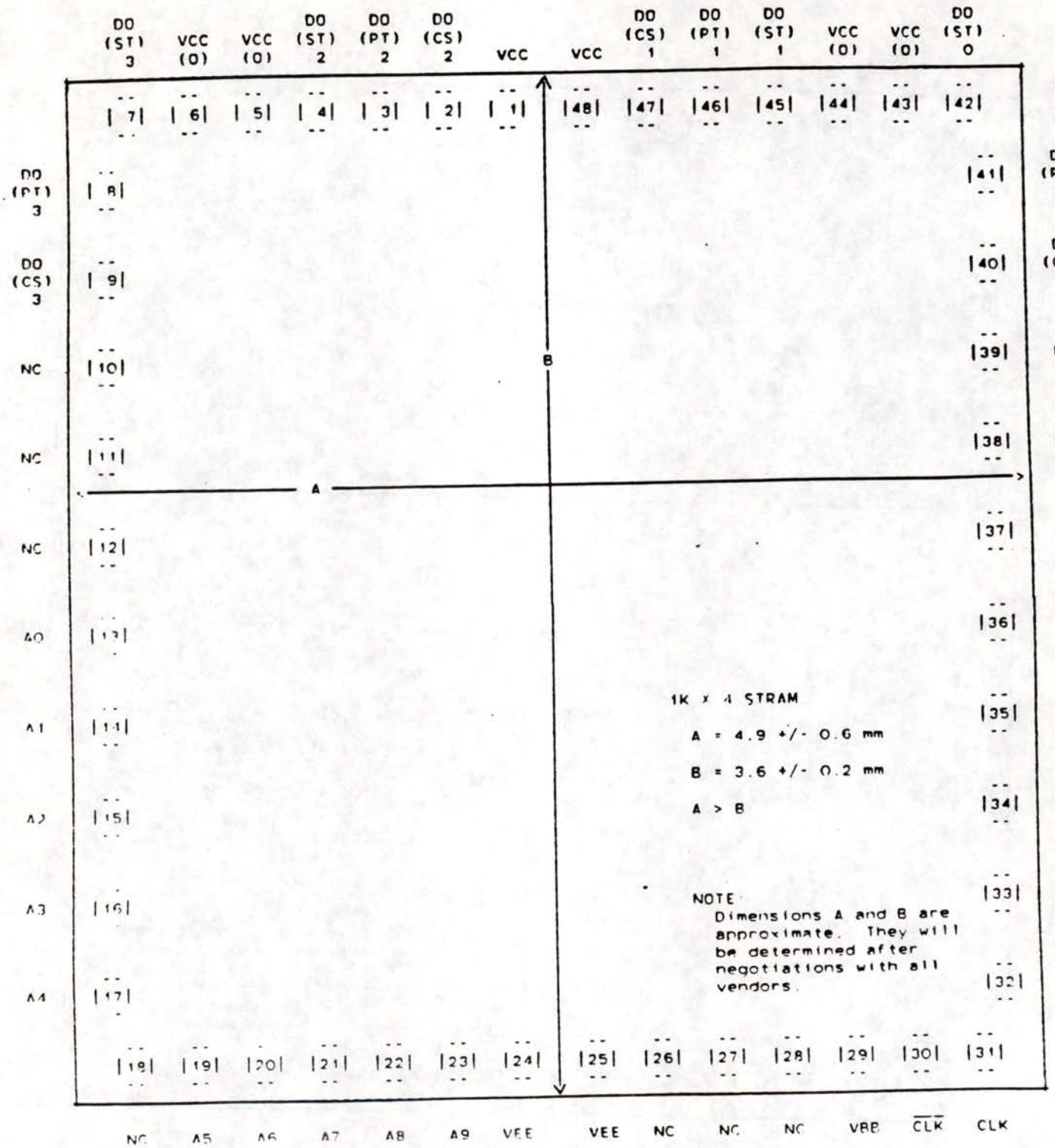
TIMING DIAGRAM

FIGURE IV



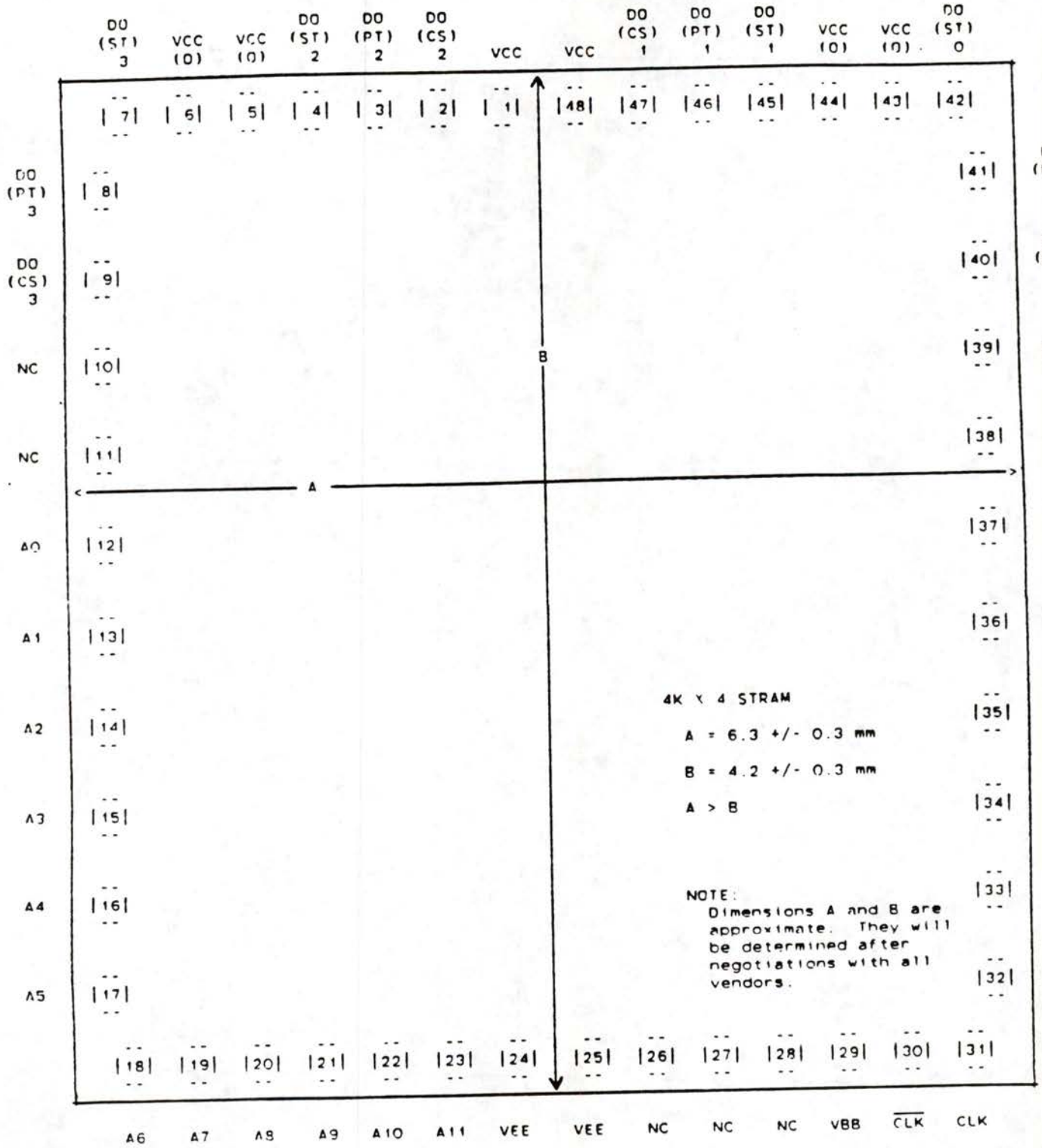
INPUT AND OUTPUT WAVEFORMS

FIGURE V



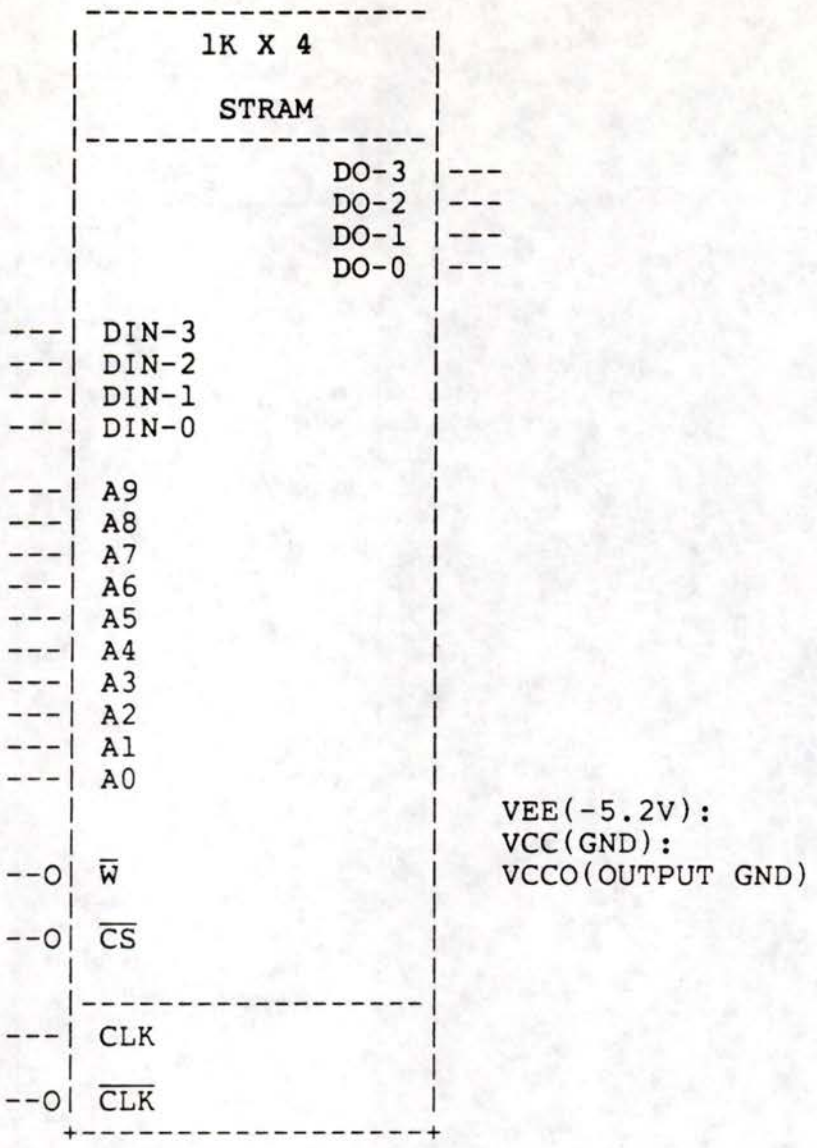
CONNECTION DIAGRAM

FIGURE VI



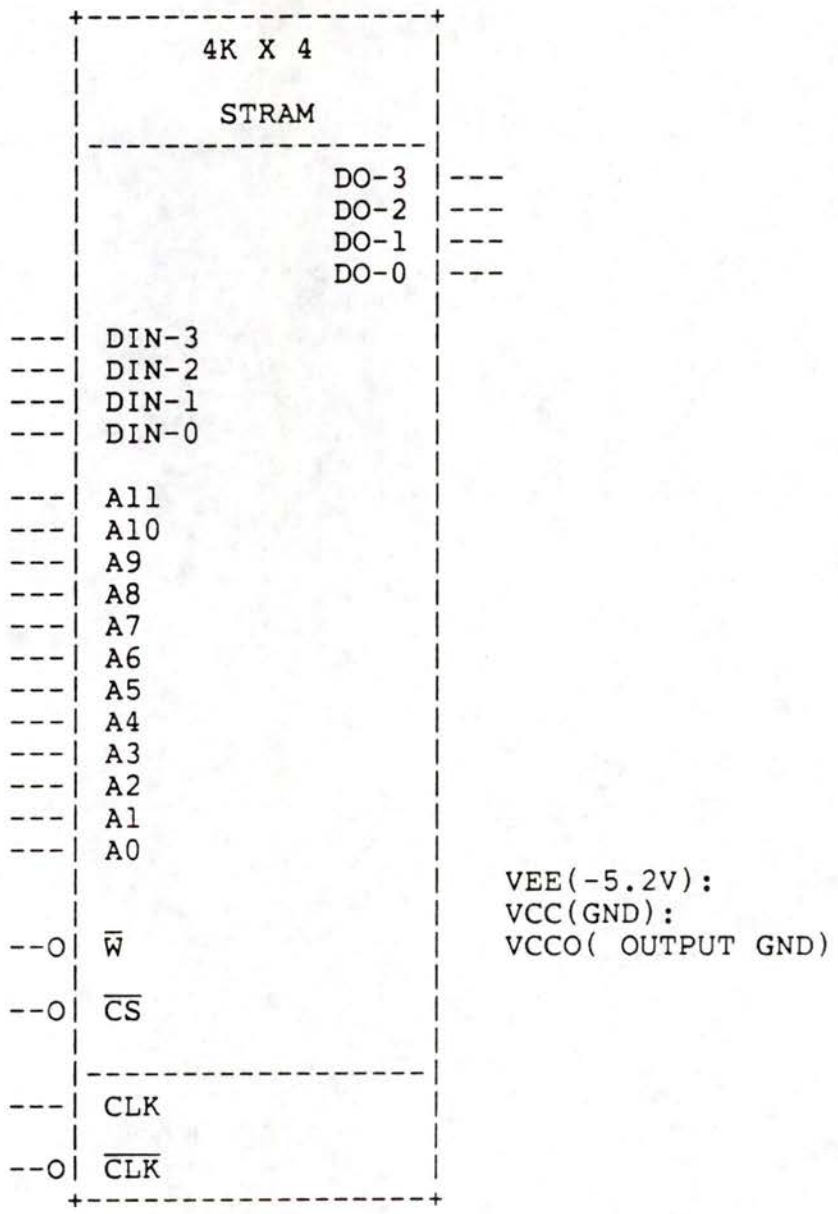
CONNECTION DIAGRAM

FIGURE VI



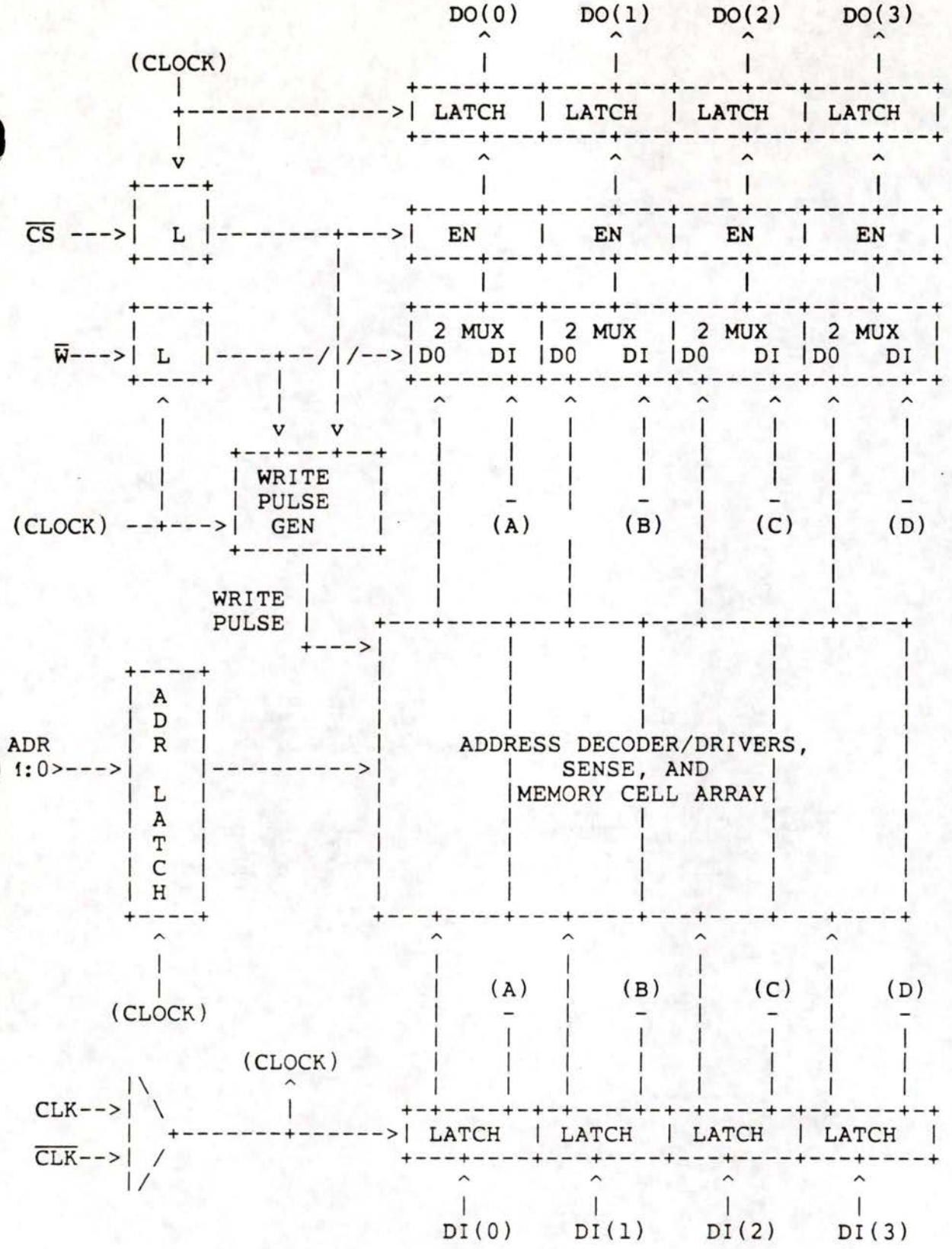
LOGIC SYMBOL

FIGURE VIII-A



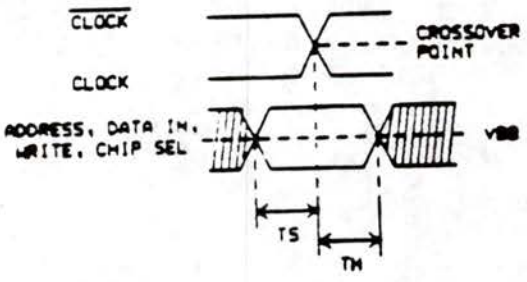
LOGIC SYMBOL

FIGURE VIII-B



FUNCTIONAL BLOCK DIAGRAM

FIGURE IX

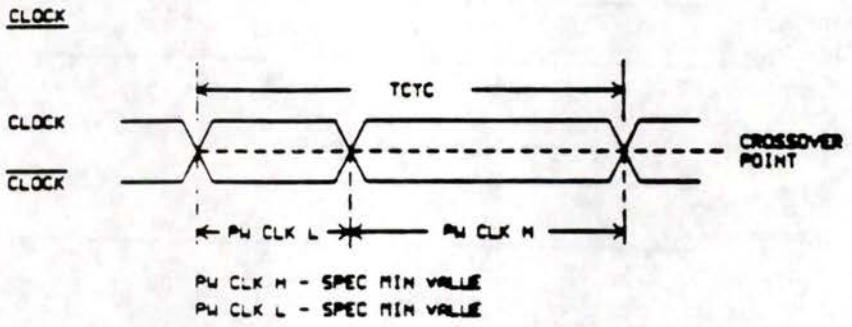


TSA - ADDRESS SETUP TIME
TSDI - DATA IN SETUP TIME
TSW - WRITE SETUP TIME
TSCS - CHIP SELECT SETUP TIME

THA - ADDRESS HOLD TIME
THDI - DATA IN HOLD TIME
THW - WRITE HOLD TIME
THCS - CHIP SELECT HOLD TIME

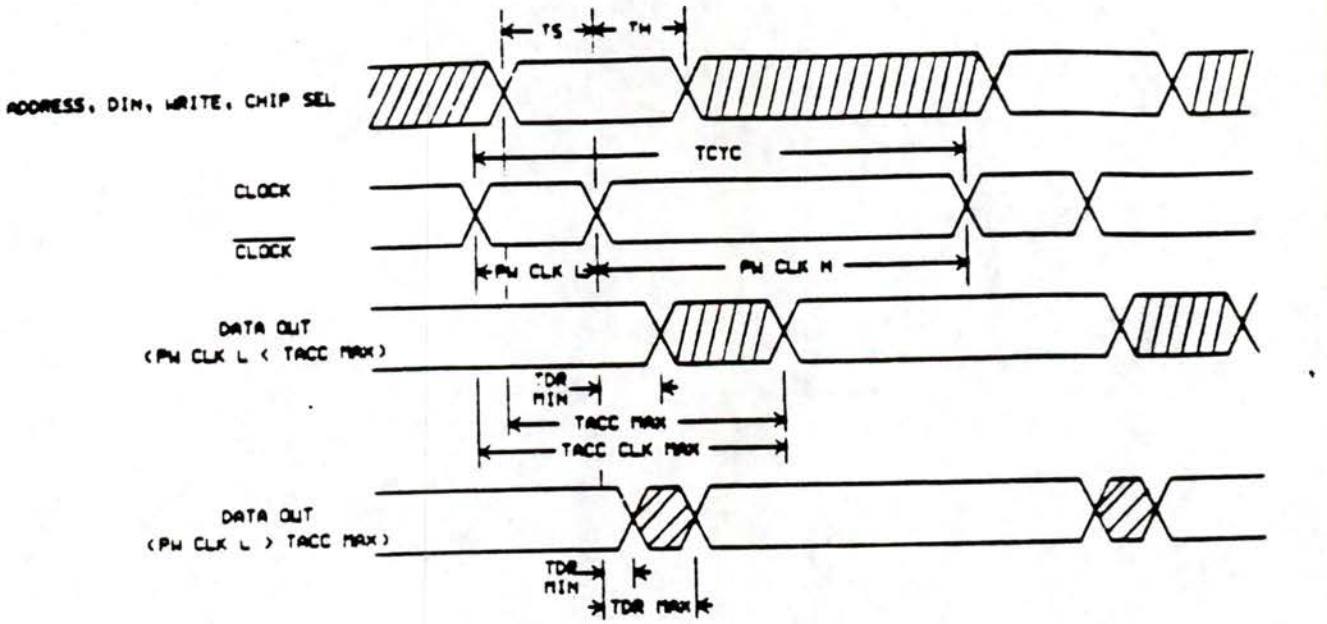
SETUP AND HOLD TIMES

FIGURE X



CLOCK CYCLE

FIGURE XI



TS, TH - SEE FIGURE X

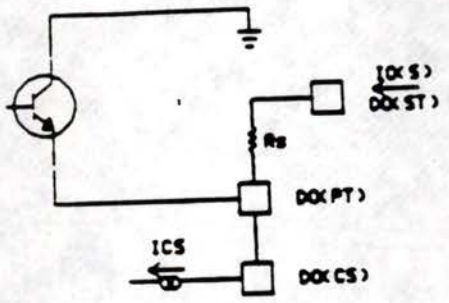
TACC ADR - ACCESS TIME FROM ADDRESS
TACC W - ACCESS TIME FROM WRITE

TACC CS - ACCESS TIME FROM CHIP SELECT
TACC CLK - ACCESS TIME FROM CLK-FALLING EDGE
TACC DIN - ACCESS TIME FROM DATA IN

TDR - OUTPUT DELAY FROM CLOCK RISING EDGE
PW CLK L - CLOCK PW LOW STATE
PW CLK H - CLOCK PW HIGH STATE

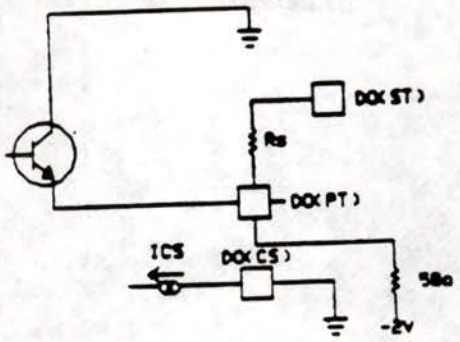
ACCESS AND DELAY TIMING

FIGURE XII



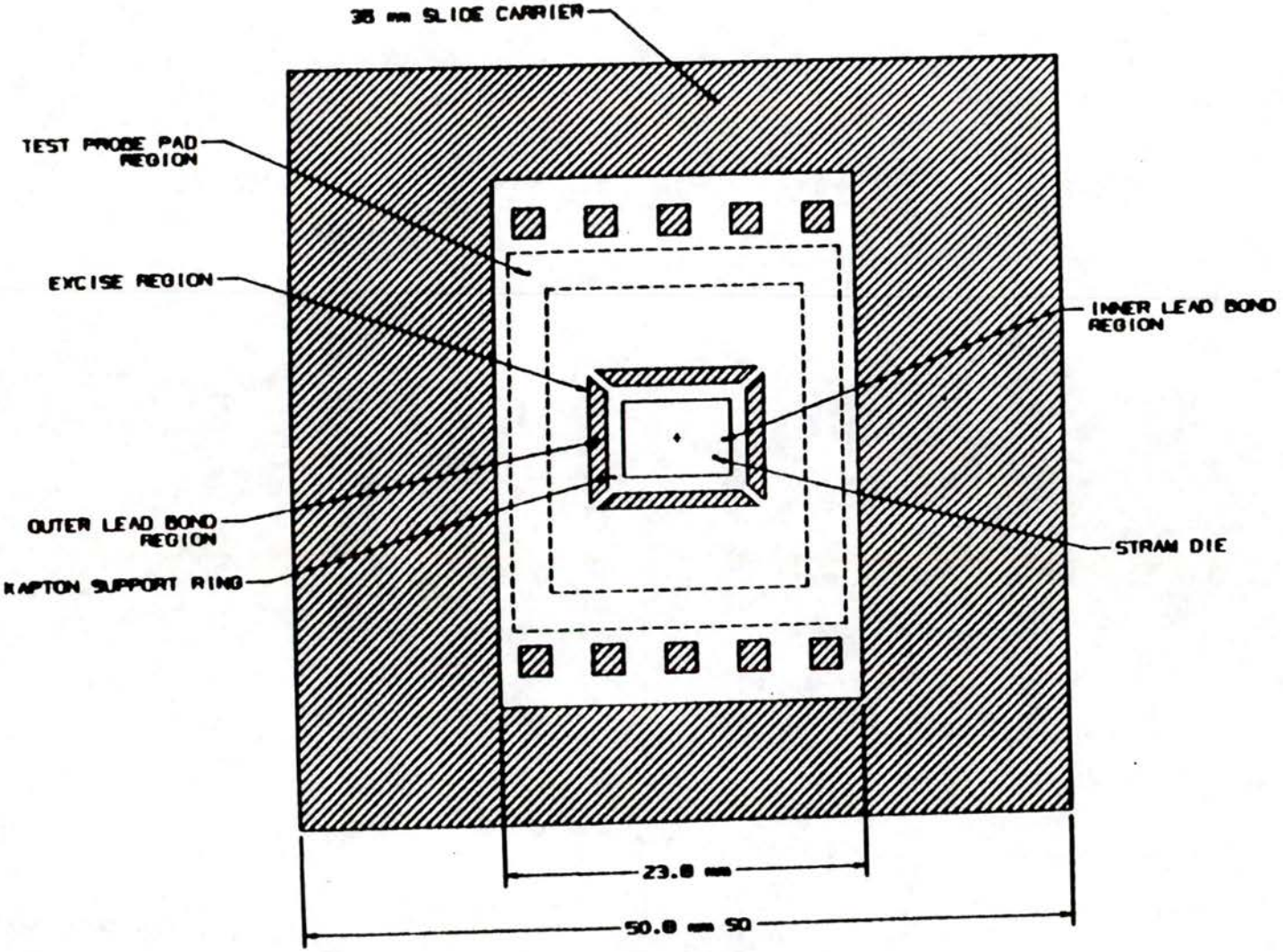
OUTPUT TEST CIRCUIT SERIES TERMINATION

FIGURE XIII

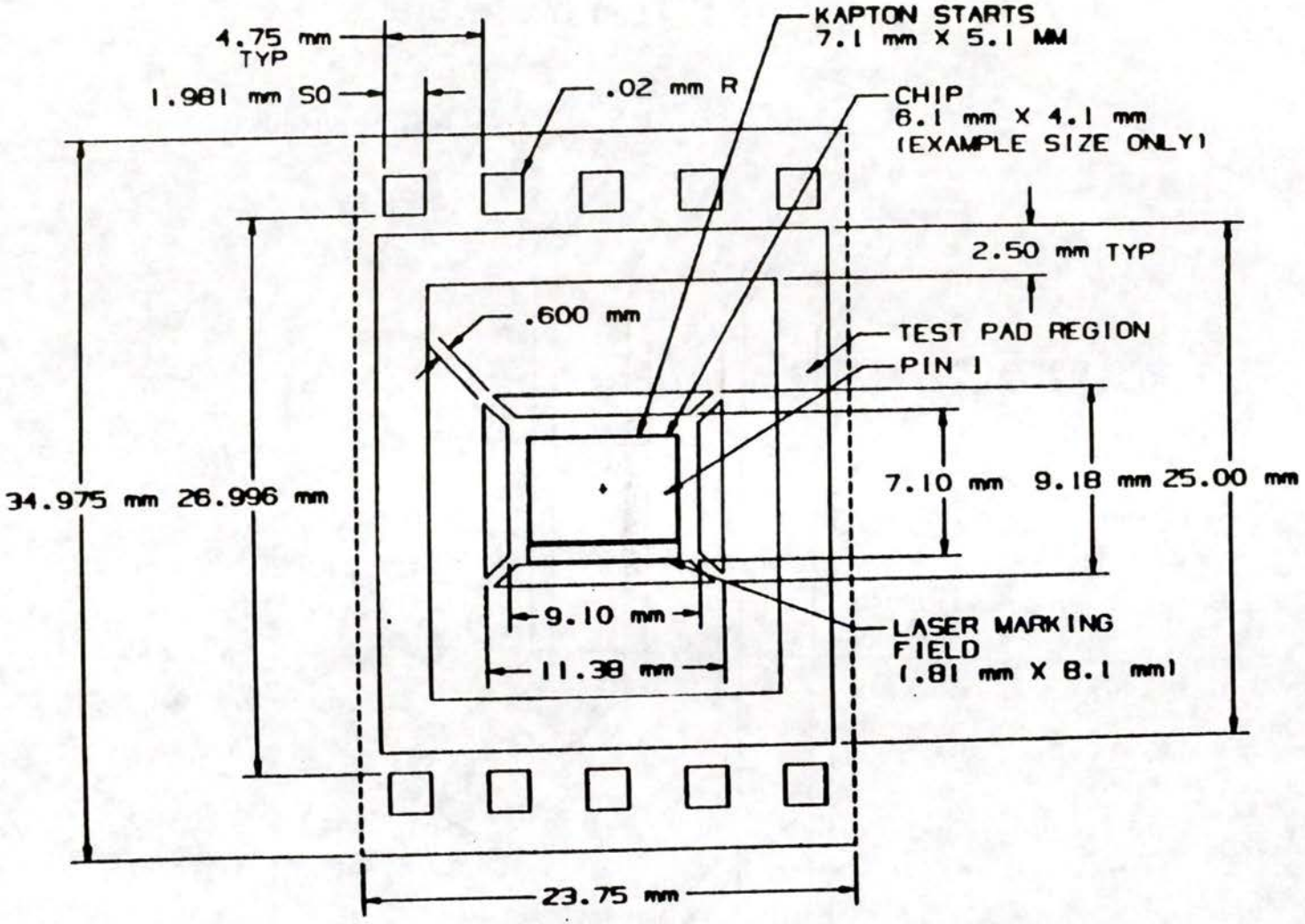


OUTPUT TEST CIRCUIT PARALLEL TERMINATION

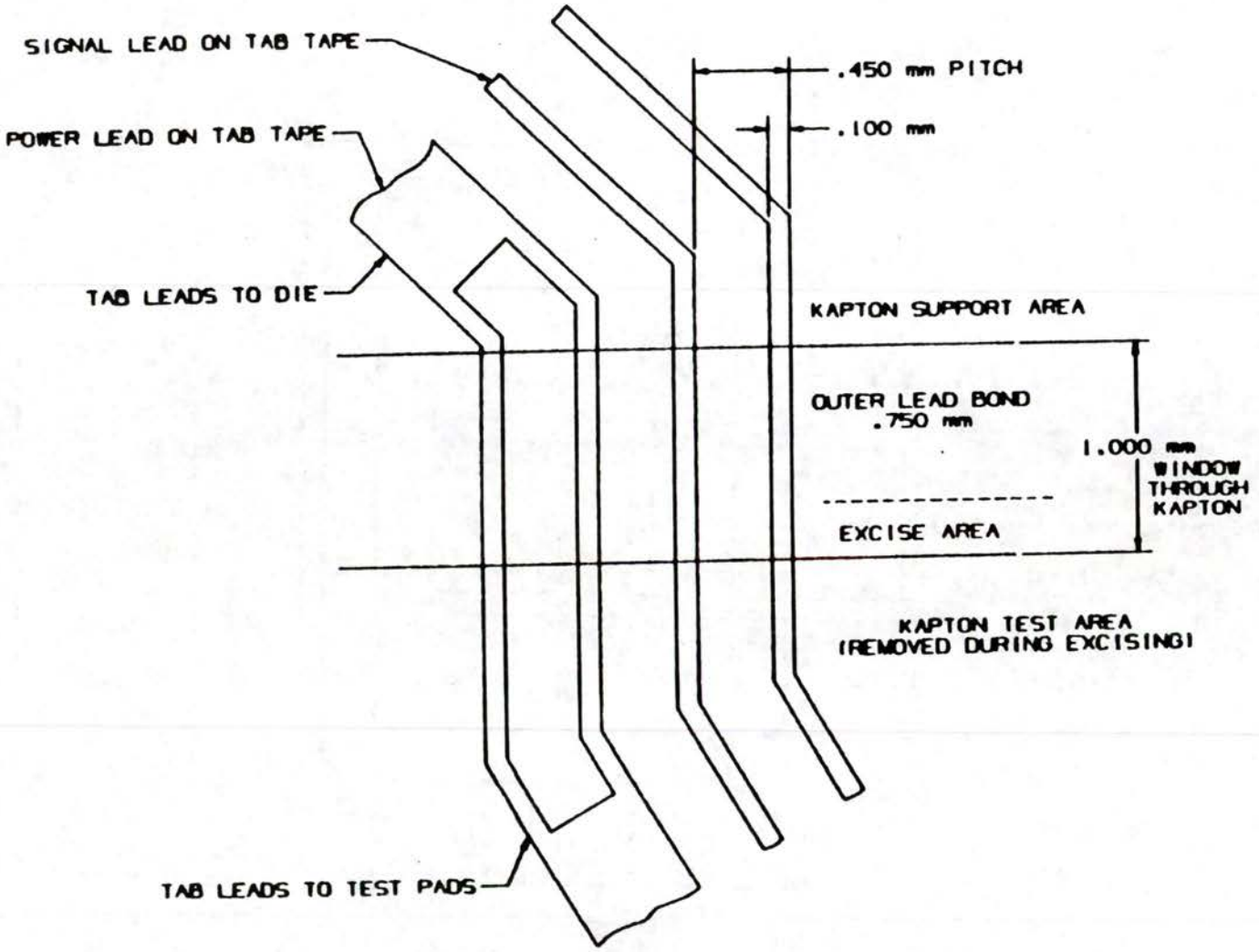
FIGURE XIV



EXAMPLE STRAM ON TAB TAPE
FIGURE XV-A



EXAMPLE STRAM ON TAPE
FIGURE XV-B



STRAM TAB DETAIL IN OLB REGION
FIGURE XVI

----- I N D E X -----

APRIL 10, 1987

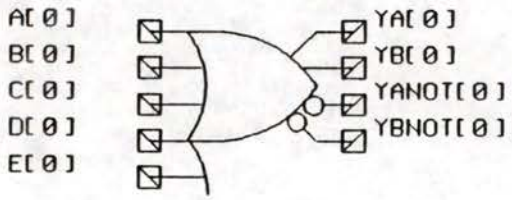
RESTRICTED
DISTRIBUTION

MCA3 SHOW

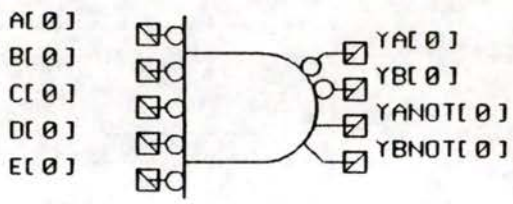
POWER INFO

CELL SIZE
.25

H200



H200I



BOOLEAN EXPRESSION

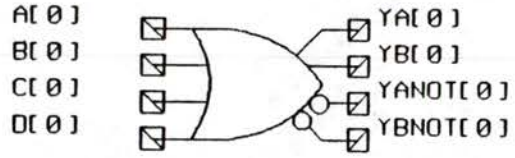
YA=YB=A+B+C+D+E
YANOT=YBNOT=NOT(A+B+C+D+E)

MCA3 SHOW

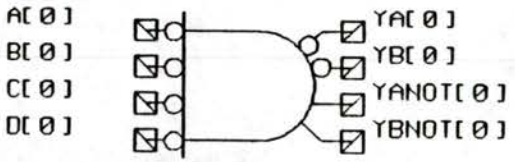
POWER INFO

CELL SIZE
.25

H201



H201I



BOOLEAN EXPRESSION

YA=YB=A+B+C+D

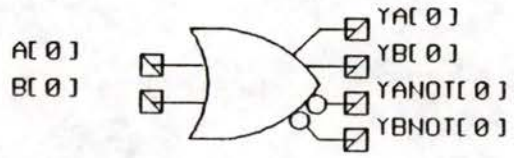
YANOT=YBNOT=NOT(A+B+C+D)

MCA3 SHOW

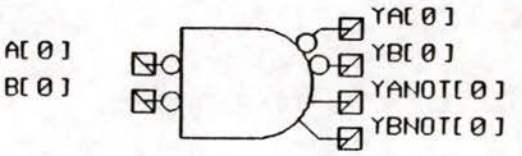
CELL SIZE
.25

POWER INFO

H202



H2021



BOOLEAN EXPRESSION

YA=YB=A+B

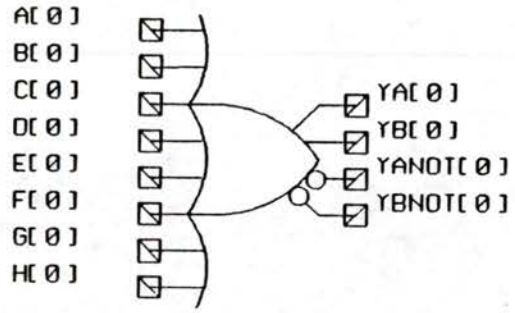
YANOT=YBNOT=NOT(A+B)

MCA3 SHOW

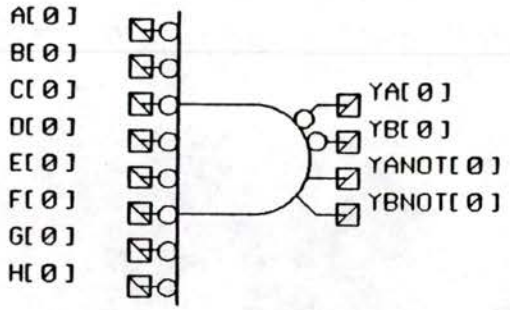
POWER INFO

CELL_SIZE
.5

H203



H203I



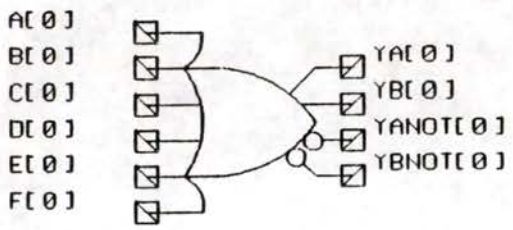
BOOLEAN EXPRESSION
YA = YB = A+B+C+D+E+F+G+H
YANOT = YBNOT = NOT(A+B+C+D+E+F+G+H)

MCA3 SHOW

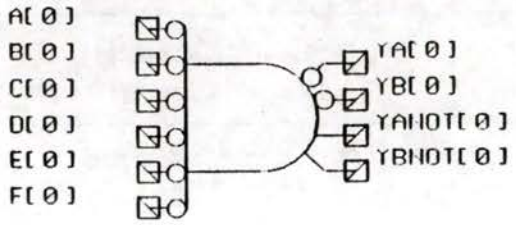
POWER INFO

CELL_SIZE
.25

H207



H2071



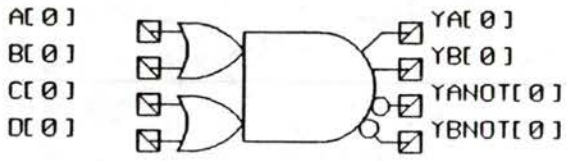
BOOLEAN EXPRESSION
YA = YB = A+B+C+D+E+F
YANOT = YBNOT = NOT(A+B+C+D+E+F)

MCA3 SHOW

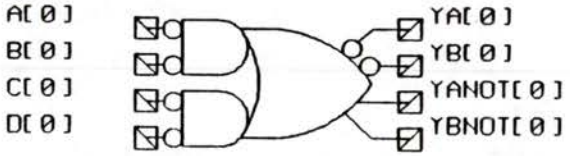
CELL SIZE
.25

POWER INFO

H211



H2111



BOOLEAN EXPRESSION

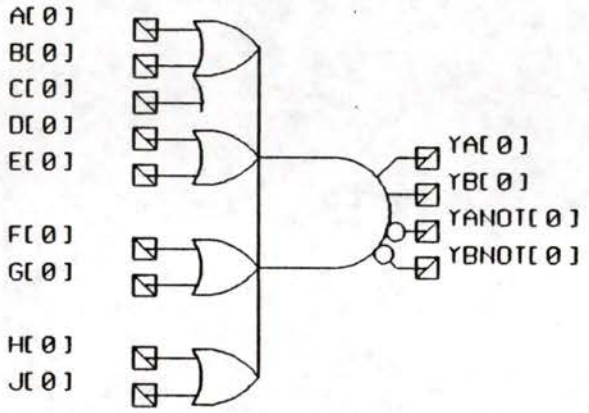
YA=YB=(A+B)*(C+D)
YANOT=YBNOT=NOT((A+B)*(C+D))

MCA3 SHOW

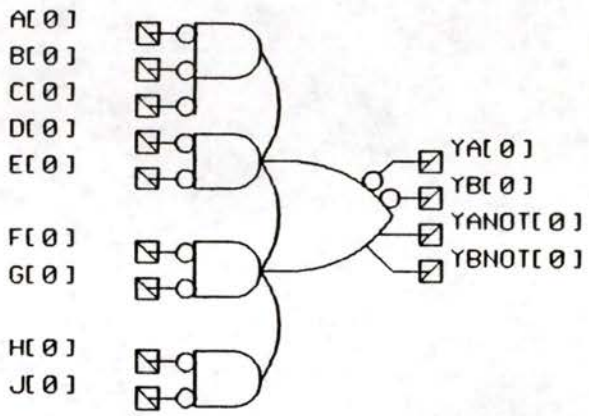
POWER INFO

CELL_SIZE .5

H212



H2121



BOOLEAN EXPRESSION

YA = YB = (A+B+C)*(D+E)*(F+G)*(H+J)

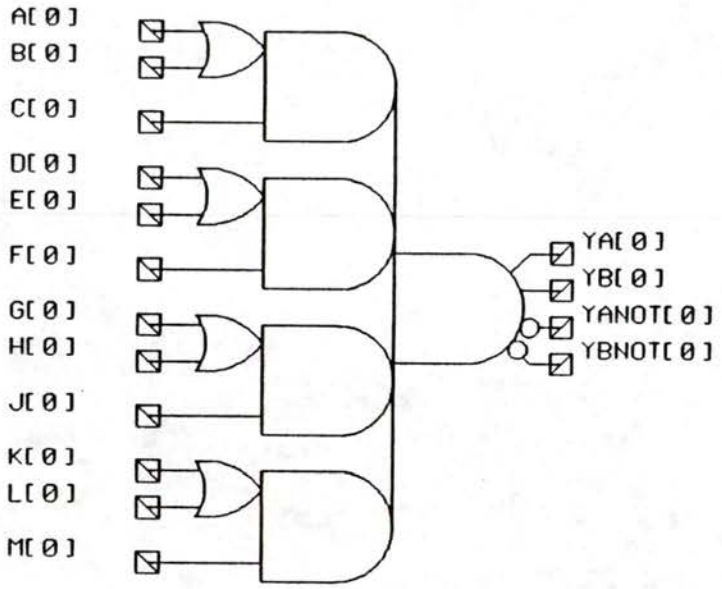
YANOT = YBNOT = NOT((A+B+C)*(D+E)*(F+G)*(H+J)

MCA3 SHOW

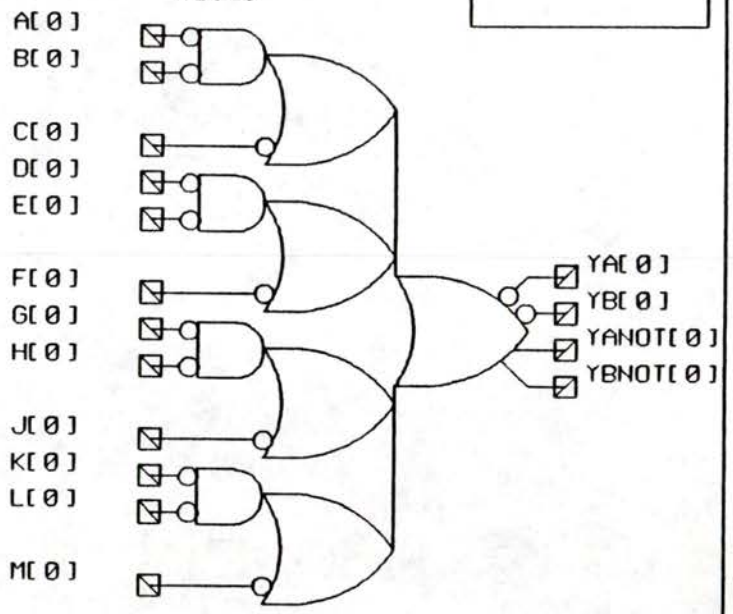
POWER INFO

CELL SIZE
1

H214



H2141



BOOLEAN EXPRESSION

YA = YB = [(A+B)*C]*[(D+E)*F]*[(G+H)*J]*[(K+L)*M]

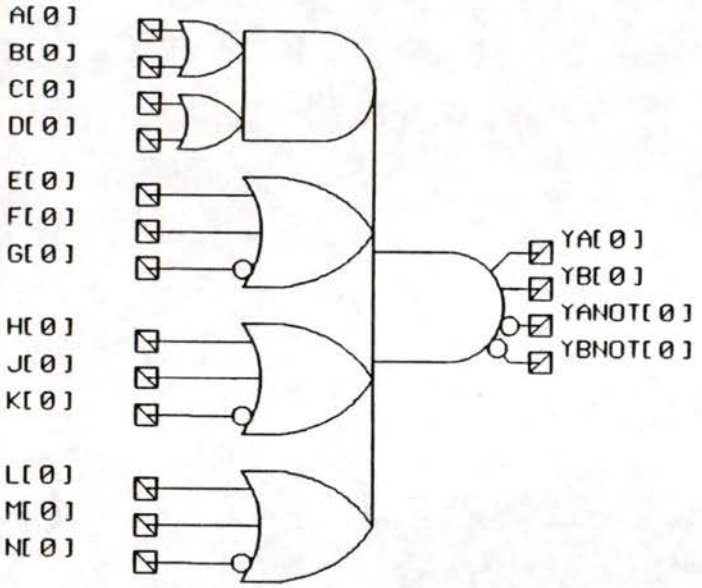
YANOT = YBNOT = NOT([(A+B)*C]*[(D+E)*F]*[(G+H)*J]*[(K+L)*M])

MCA3 SHOW

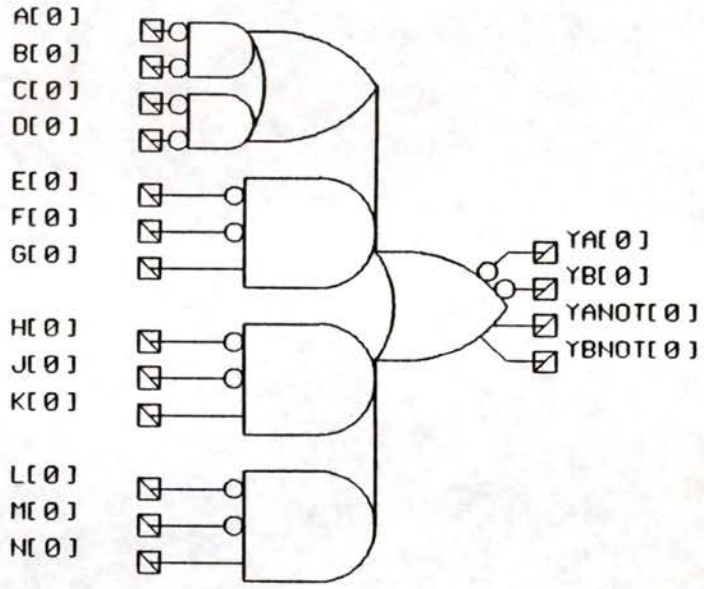
CELL SIZE
1

POWER INFO

H215



H2151



BOOLEAN EXPRESSION

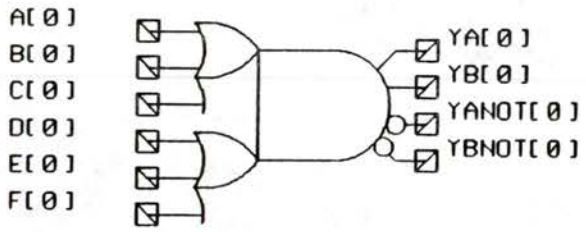
YA=YB=[(A+B)*(C+D)]*
(E+F+NOT(G))*
(H+J+NOT(K))*
(L+M+NOT(N))]
YANOT=YBNOT=NOT[(A+B)*(C+D)]*
(E+F+NOT(G))*
(H+J+NOT(K))*
(L+M+NOT(N))]

MCA3 SHOW

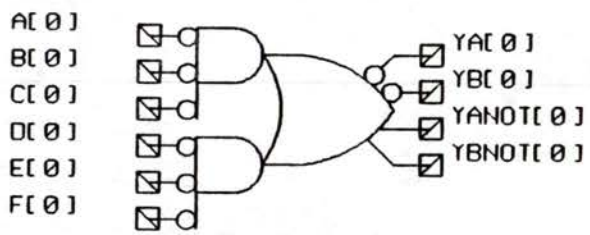
POWER INFO

CELL_SIZE
.25

H219



H2191



BOOLEAN EXPRESSION

YA = YB = (A+B+C)*(D+E+F)

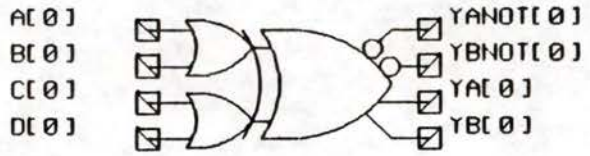
YANOT = YBNOT = NOT((A+B+C)*(D+E+F))

MCA3 SHOW

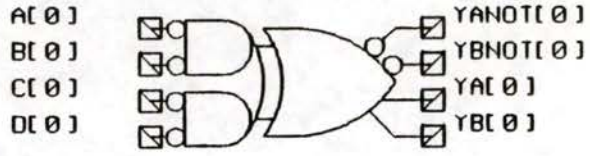
POWER INFO

CELL_SIZE
.25

H221



H2211



BOOLEAN EXPRESSION

YA = YB = (A+B)XOR(C+D)

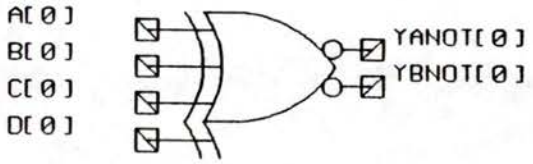
YANOT = YBNOT = NOT((A+B)XOR(C+D))

MCA3 SHOW

POWER INFO

CELL_SIZE
.5

H223



BOOLEAN EXPRESSION

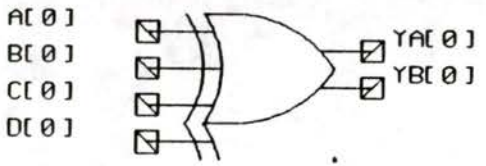
YANOT = YBNOT = NOT(A XOR B XOR C XOR D)

MCA3 SHOW

POWER INFO

CELL_SIZE
.5

H224



BOOLEAN EXPRESSION

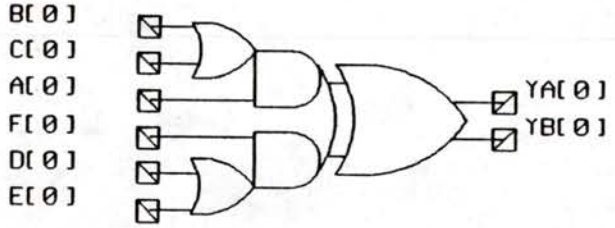
YANOT = YBNOT = A XOR B XOR C XOR D

MCA3 SHOW

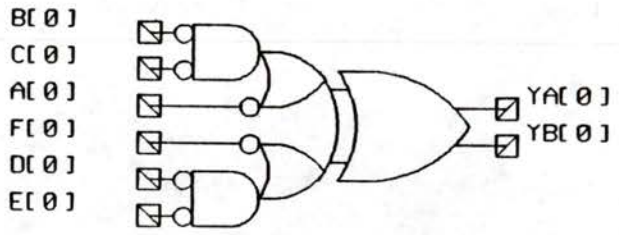
POWER INFO

CELL_SIZE
.5

H225



H2251



BOOLEAN EXPRESSION

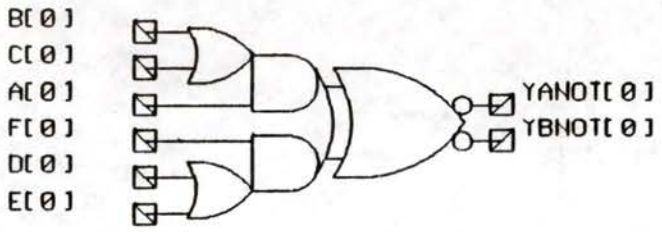
YA = YB = ((B+C)*A)) XOR ((D+E)*F))

MCA3 SHOW

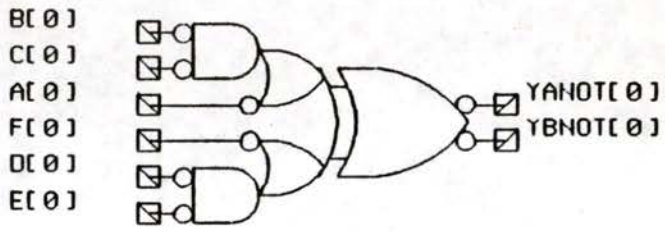
POWER INFO .

CELL_SIZE
.5

H226



H2261



BOOLEAN EXPRESSION

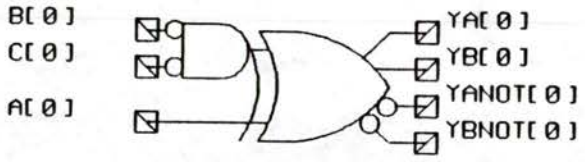
YA = YB = NOT(((B+C)*A)) XOR ((D+E)*F)))

MCA3 SHOW

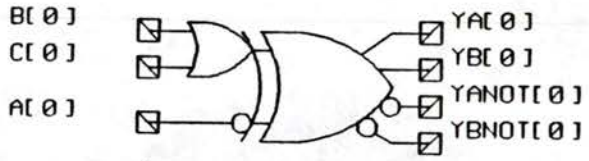
POWER INFO

CELL_SIZE
.25

H228



H2281



BOOLEAN EXPRESSION

YA = YB = (B+C) XOR NOT(A)

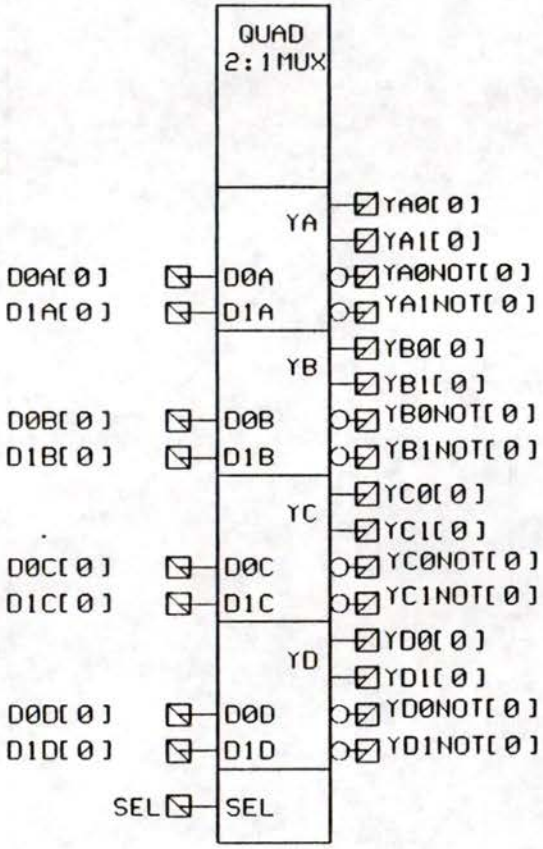
YANOT = YBNOT = NOT((B+C) XOR NOT(A))

MCA3 SHOW

POWER INFO

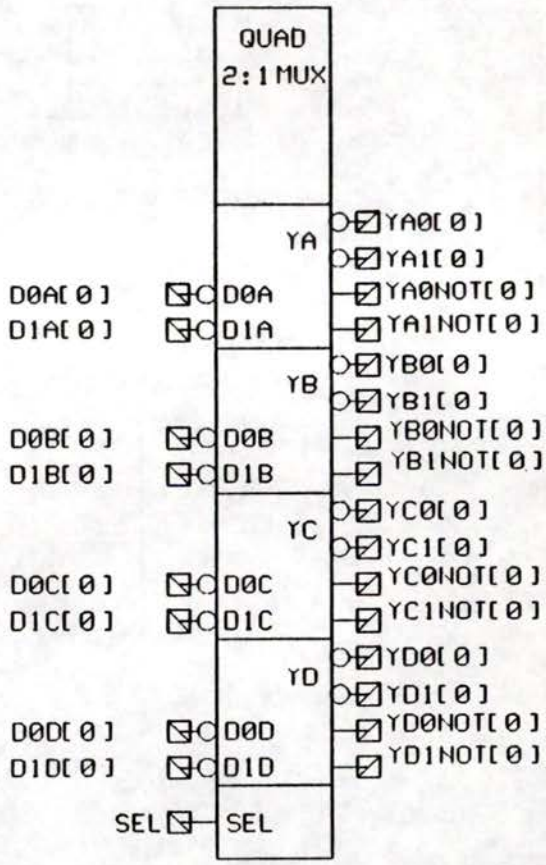
CELL SIZE
1

H252



BOOLEAN EXPRESSION				
SEL	YA0 YA1	YB0 YB1	YC0 YC1	YD0 YD1
L	D0A	D0B	D0C	D0D
H	D1A	D1B	D1C	D1D

H2521

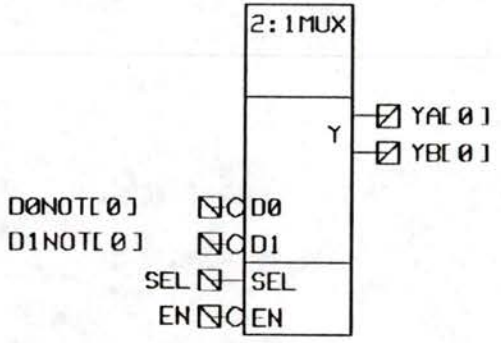


MCA3 SHOW

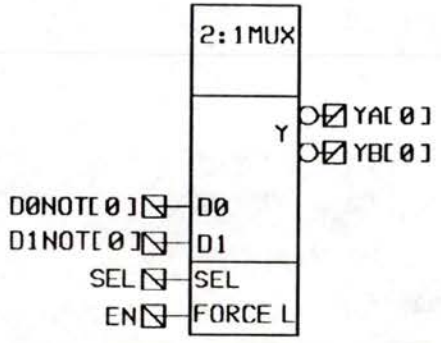
CELL SIZE
.25

POWER INFO

H253



H253I



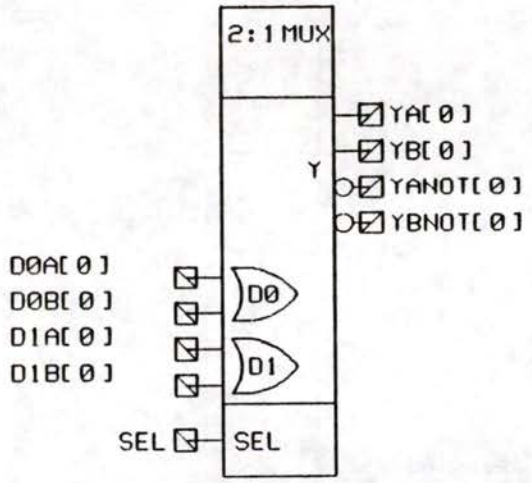
SEL	EN	YA/YB
X	H	L
L	L	NOT(D0NOT)
H	L	NOT(D1NOT)

MCA3 SHOW

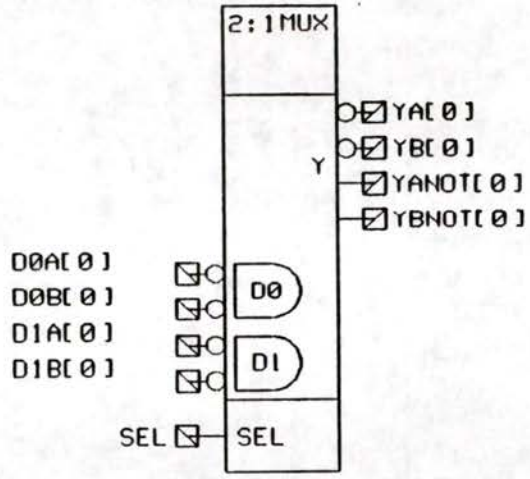
POWER INFO

CELL SIZE
.25

H254



H254I



SEL	YA, YB	YANOT, YBNOT
L	D0A+D0B	NOT(D0A+D0B)
H	D1A+D1B	NOT(D1A+D1B)

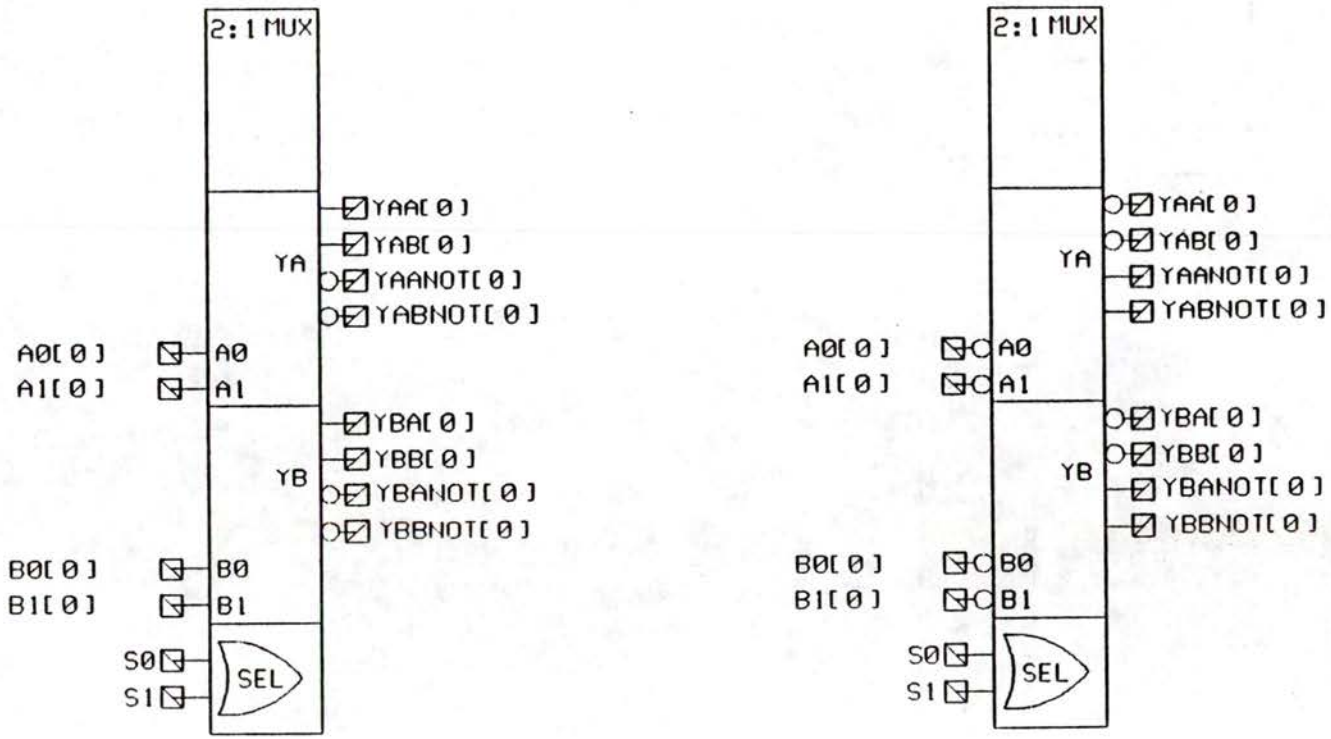
MCA3 SHOW

CELL SIZE
.5

H255

POWER INFO

H2551



S0+S1	YAA YAB	YAANOT YABNOT	YBA YBB	YBANOT YBBNOT
L	A0	NOT(A0)	B0	NOT(B0)
H	A1	NOT(A1)	B1	NOT(B1)

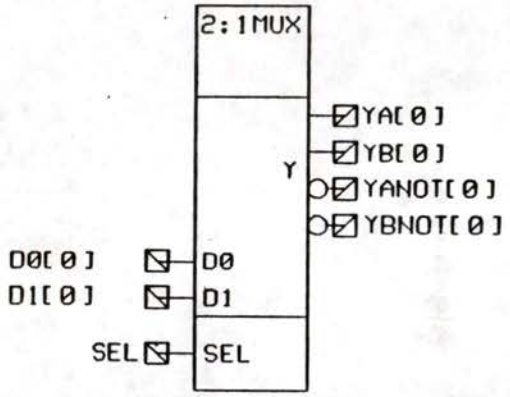
MCA3 SHOW

CELL SIZE
.25

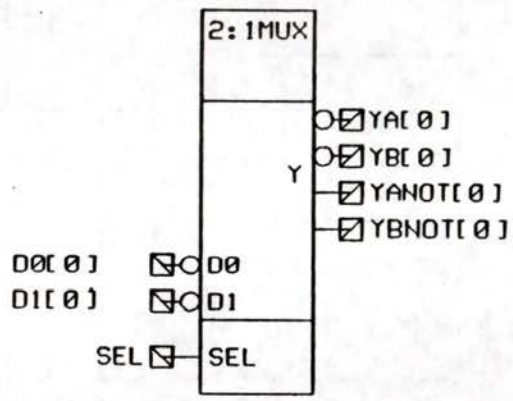
POWER INFO

306

H256



H256I



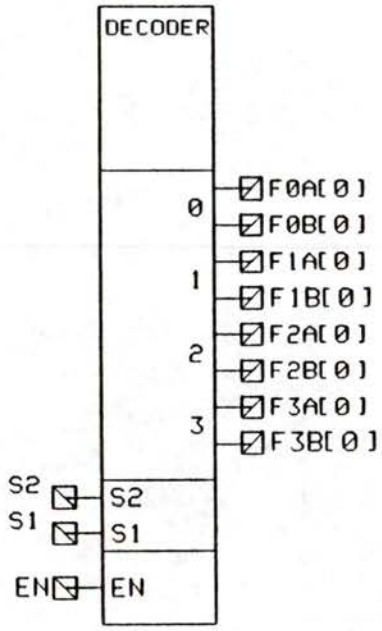
SEL	YA YB	YANOT YBNOT
L	D0	NOT(D0)
H	D1	NOT(D1)

MCA3 SHOW

POWER INFO

CELL SIZE
1

H263



NO DeMORGAN'S EQUIVALENT

EN	S2	S1	F0A F0B	F1A F1B	F2A F2B	F3A F3B
L	X	X	L	L	L	L
H	L	L	H	L	L	L
H	L	H	L	H	L	L
H	H	L	L	L	H	L
H	H	H	L	L	L	H

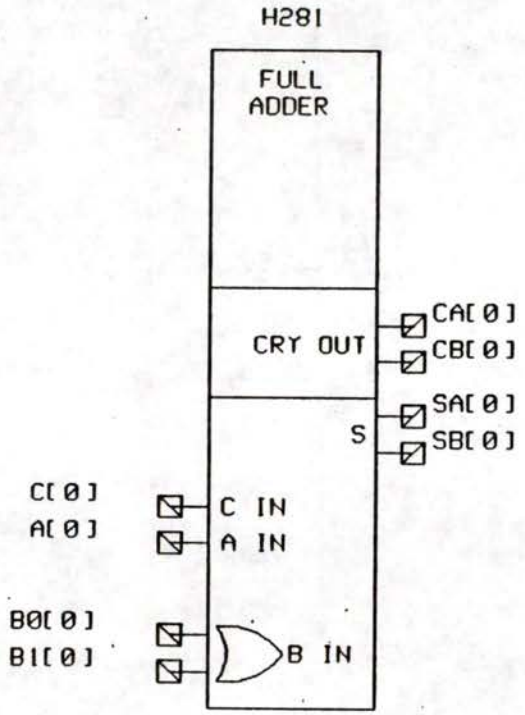
X= DON'T CARE

MCA3 SHOW

POWER INFO

CELL SIZE
1

NO DeMORGAN'S EQUIVALENT



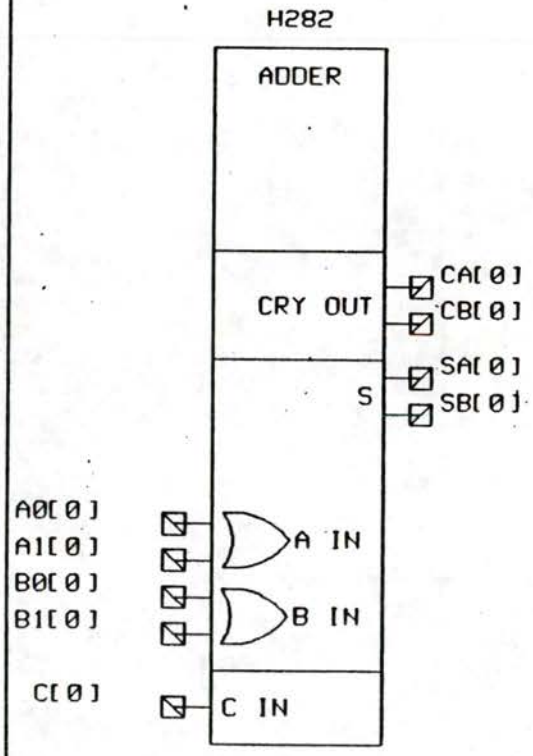
A	B0	B1	C	SA,SB	CA,CB
L	L	L	L	L	L
L	L	L	H	H	L
L	H	X	L	H	L
L	X	H	L	H	L
L	H	X	H	L	H
L	X	H	H	L	H
H	L	L	L	H	L
H	L	L	H	L	H
H	H	X	L	L	H
H	X	H	L	L	H
H	H	X	H	H	H
H	X	H	H	H	H

MCA3 SHOW

POWER INFO

CELL SIZE
.5

NO DeMORGAN'S EQUIVALENT



A0	A1	B1	B0	C	SA,SB	CA,CB
L	L	L	L	L	L	L
L	L	H	X	L	H	L
L	L	X	H	L	H	L
L	L	H	X	H	L	H
L	L	X	H	H	L	H
H	X	L	L	L	H	L
X	H	L	L	L	H	L
H	X	L	L	H	L	H
X	H	L	L	H	L	H
H	X	H	X	L	L	H
X	H	X	H	L	L	H
H	X	H	X	H	H	H
X	H	X	H	H	H	H
L	L	L	L	H	H	L

CELL SIZE

1

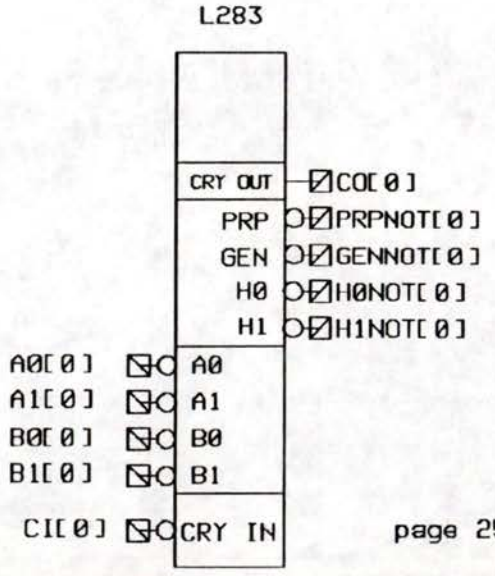
MCA3 SHOW

POWER INFO

NO DeMORGAN'S EQUIVALENT

B1	A1	B0	A0	GENNOT	PRPNOT	H0NOT	H1NOT
L	L	L	L	L	H	H	H
L	L	L	H	L	H	L	H
L	L	H	L	L	H	L	H
L	L	H	H	L	H	H	H
L	H	L	L	L	H	H	L
L	H	L	H	H	L	L	L
L	H	H	L	H	L	L	L
L	H	H	H	H	H	H	L
H	L	L	L	L	H	H	L
H	L	L	H	H	L	L	L
H	L	H	L	H	L	L	L
H	L	H	H	H	H	H	L
H	H	L	L	H	H	H	H
H	H	L	H	H	H	L	H
H	H	H	L	H	H	L	H
H	H	H	H	H	H	H	H

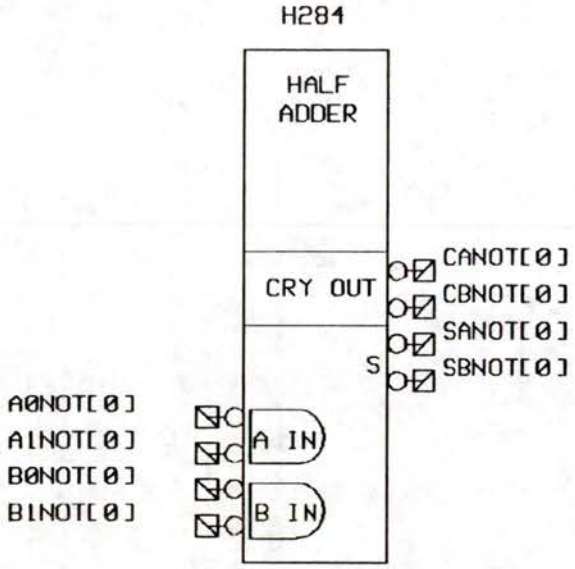
A0	B0	CRYIN	CRYOUT
L	L	L	H
L	L	H	H
L	H	L	H
L	H	H	L
H	L	L	H
H	L	H	L
H	H	L	L
H	H	H	L



MCA3 SHOW

POWER INFO

CELL SIZE
.25



NO DeMORGAN'S EQUIVALENT

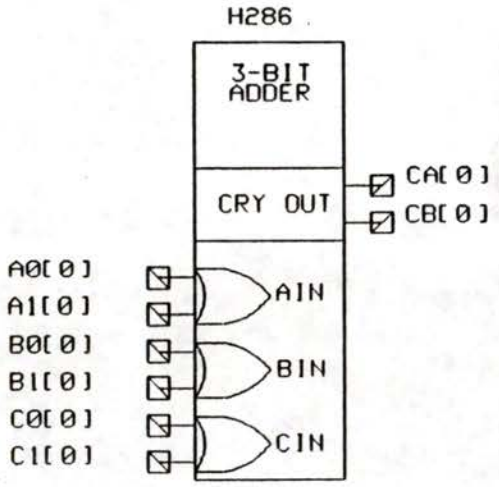
NOT(A0NOT)*NOT(A1NOT)	NOT(B0NOT)*NOT(B1NOT)	CANOT, CBNOT	SANOT, SBNOT
L	L	H	H
L	H	H	L
H	L	H	L
H	H	L	H

MCA3 SHOW

CELL SIZE
.5

POWER INFO

NO DeMORGAN'S EQUIVALENT



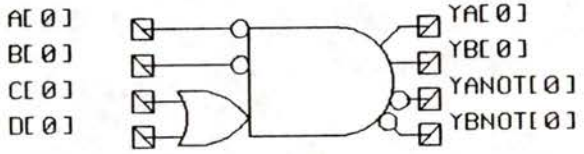
A0+A1	B0+B1	C0+C1	CA,CB
L	L	L	L
L	L	H	L
L	H	L	L
L	H	H	H
H	L	L	L
H	L	H	H
H	H	L	H
H	H	H	H

MCA3 SHOW

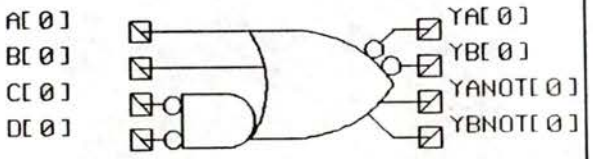
CELL SIZE
.25

POWER INFO

H313



H313I



BOOLEAN EXPRESSION

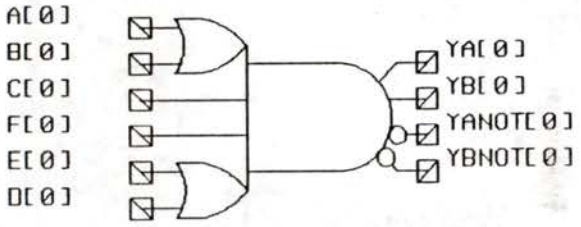
YA=YB=NOT(A)*NOT(B)*(C+D)
YANOT=YBNOT=(A+B+(NOT(C)*NOT(D)))

MCA3 SHOW

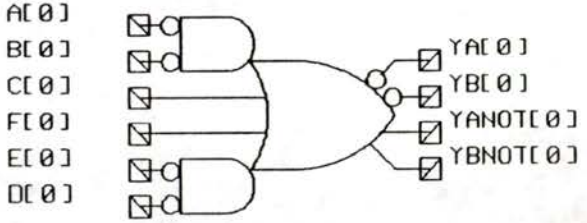
CELL SIZE
.5

POWER INFO

H315



H315I



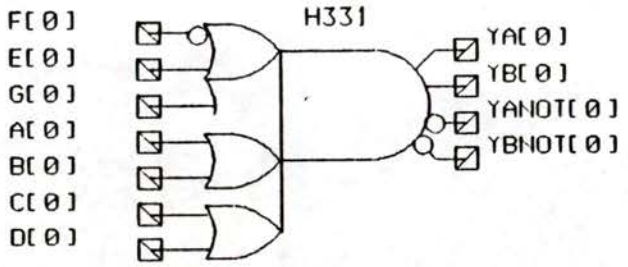
BOOLEAN EXPRESSION

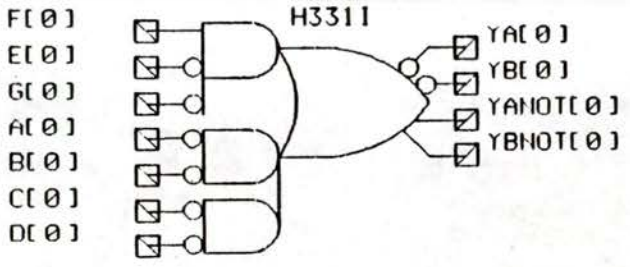
YA=YB=(A+B)*C*F*(D+E)
YANOT=YBNOT=NOT((A+B)*C*F*(D+E))

MCA3 SHOW

CELL SIZE
.5

POWER INFO





BOOLEAN EXPRESSION

YA=YB=((NOT(F)+E+G)*(A+B)*(C+D))

YANOT=YBNOT=NOT((NOT(F)+E+G)*(A+B)*(C+D))

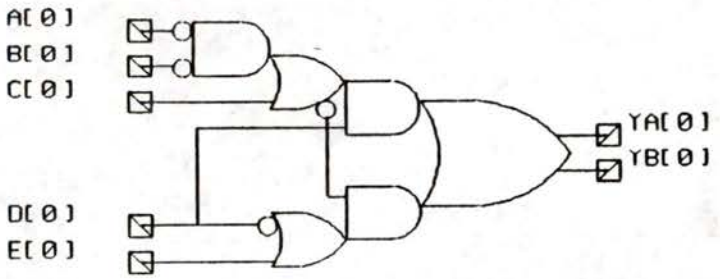
MCA3 SHOW

CELL SIZE

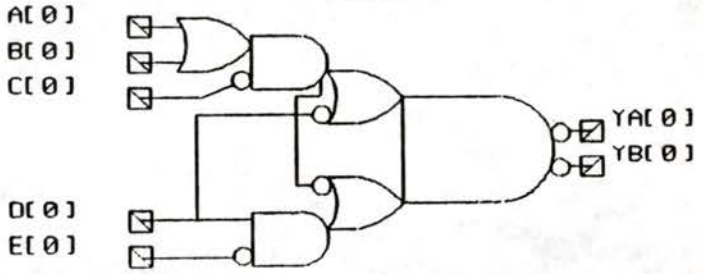
.5

POWER INFO

H332



H3321



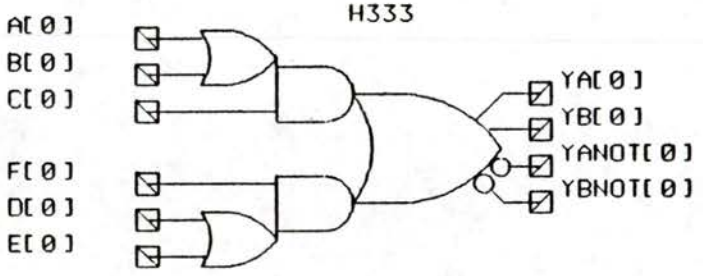
YA=YB=(((NOTA)*(NOTB))+C)*D)+
(((A+B)+NOTC)*((NOTD)+E))

MCA3 SHOW

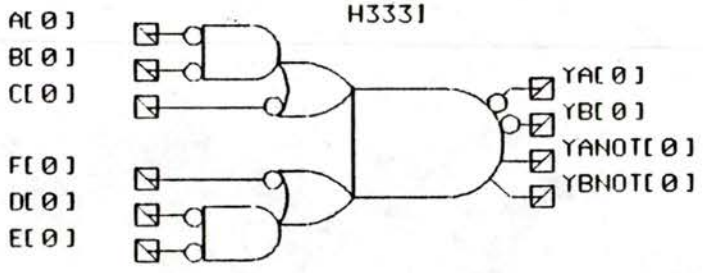
CELL SIZE
.5

POWER INFO

H333



H3331



BOOLEAN EXPRESSION

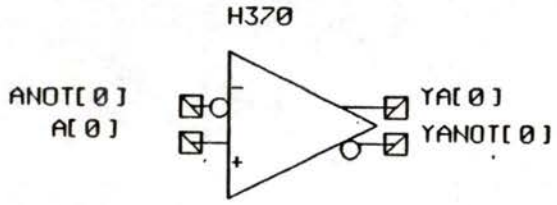
YA=YB=((A+B)*C)+((D+E)*F)

YANOT=YBNOT=NOT((A+B)*C)+((D+E)*F))

MCA3 SHOW

CELL SIZE
.25

POWER INFO



NO DeMORGAN'S EQUIVALENT

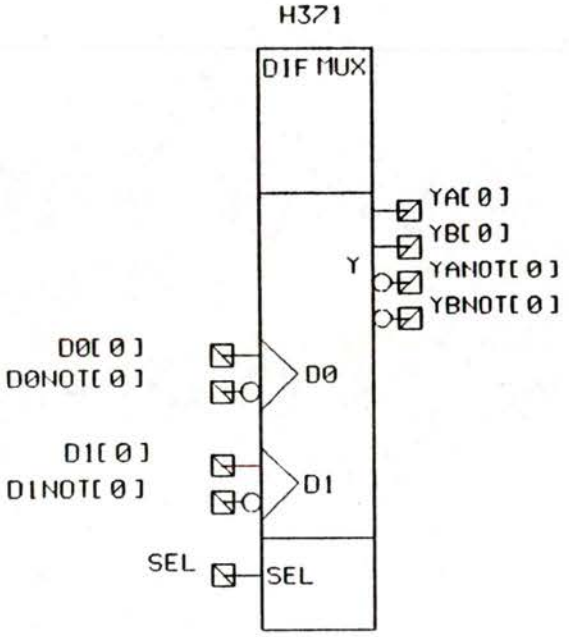
ANOT	A	YA	YANOT
L	L	N.D.	N.D.
L	H	H	L
H	L	L	H
H	H	N.D.	N.D.

MCA3 SHOW

POWER INFO

CELL SIZE
.25

NO DeMORGAN'S EQUIVALENT

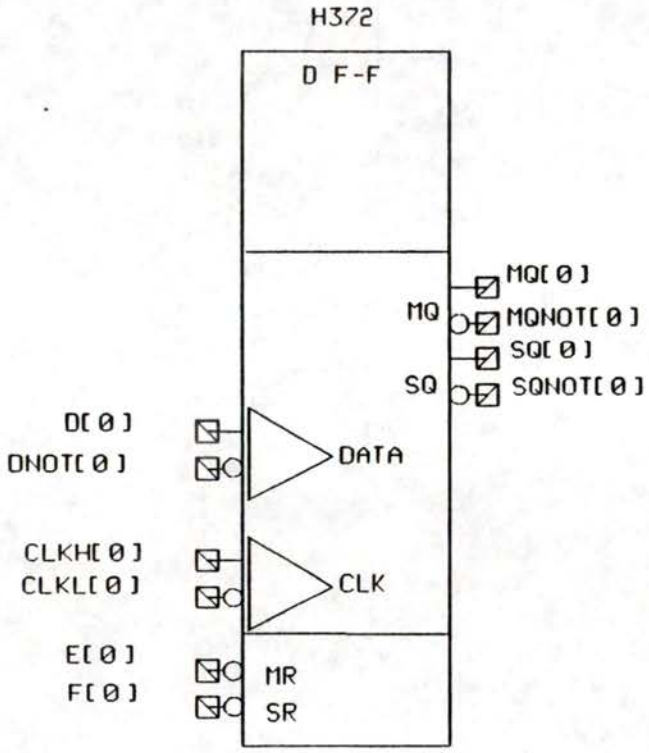


D0	D0NOT	D1	D1NOT	SEL	YA, YB	YANOT, YBNOT
L	L	X	X	L	N.D.	N.D.
L	H	X	X	L	L	H
H	L	X	X	L	H	L
H	H	X	X	L	N.D.	N.D.
X	X	L	L	H	N.D.	N.D.
X	X	L	H	H	L	H
X	X	H	L	H	H	L
X	X	H	H	H	N.D.	N.D.

MCA3 SHOW

POWER INFO

CELL SIZE
.5



NO DeMORGAN'S EQUIVALENT

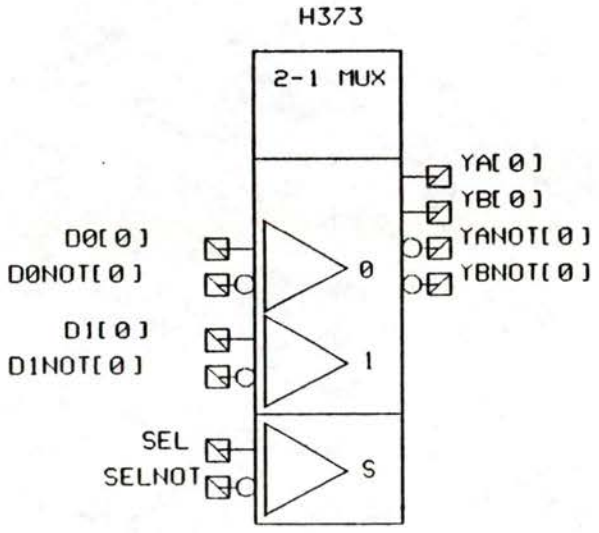
MASTER RESET	SLAVE RESET	DATA		CLOCK		MASTER Q	SLAVE Q
E	F	D	DNOT	CKH	CKL	MQ	SQ
L	L	X	X	H	L	—	—
L	L	L/H	H/L	L	H	L/H	—
L	L	L	H	L → H	H → L	L	L
L	L	H	L	L → H	H → L	H	H
H	X	X	X	H	L	L	L
H	L	L/H	H/L	L	H	L/H	—/—
X	H	L/H	H/L	L	H	L/H	L/L
L	H	H	L	L → H	H → L	⌐	⌐⌐
X	X	L	H	L → H	H → L	L	L
H	L	H	L	L → H	H → L	⌐	⌐⌐
L	H	H	L	L → H	H → L	H	⌐

MCA3 SHOW

CELL SIZE
.25

POWER INFO

NO DeMORGAN'S EQUIVALENT



SEL	YA	YANOT
L	D0	NOT D0
H	D1	NOT D1

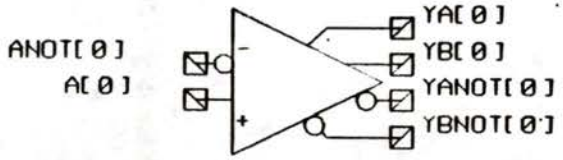
MCA3 SHOW

CELL SIZE
.25

POWER INFO

NO DeMORGAN'S EQUIVALENT

H374



YA=YB=A
YANOT=YBNOT=NOT(A)

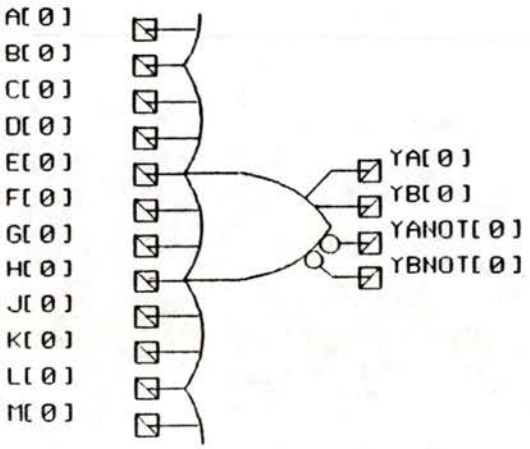
CELL SIZE

.5

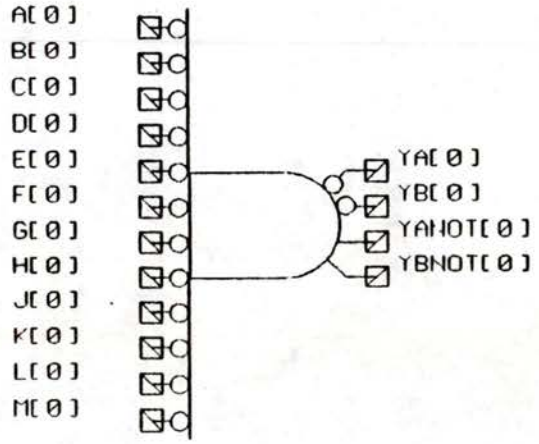
MCA3 SHOW

POWER INFO

H404



H4041



BOOLEAN EXPRESSION

YA=YB=A+B+C+D+E+F+G+H+J+K+L+M

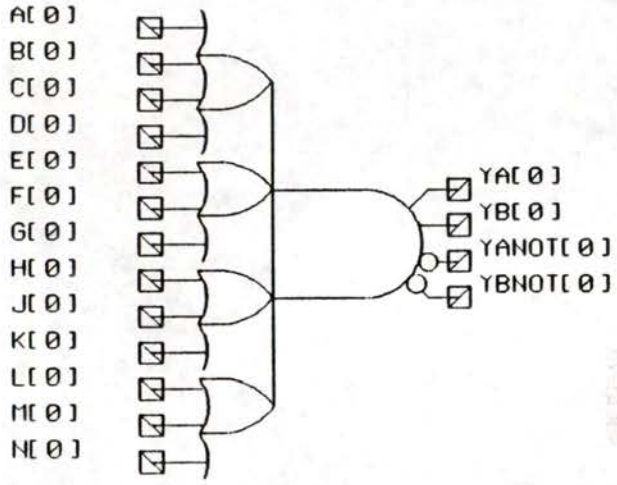
YANOT=YBNOT=NOT(A+B+C+D+E+F+G+H+J+K+L+M)

MCA3 SHOW

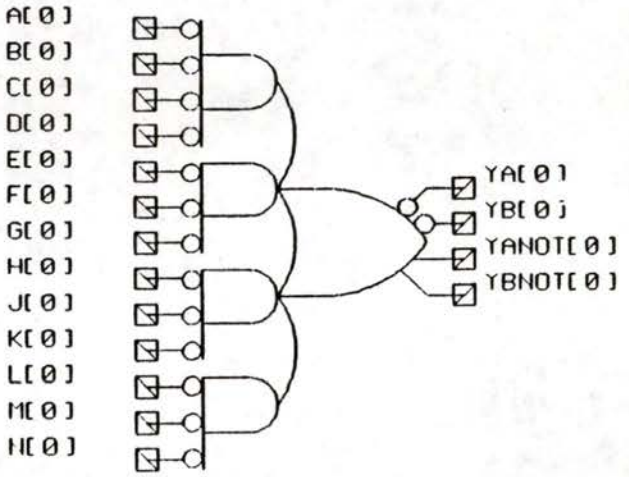
CELL SIZE
.5

POWER INFO

H413



H4131



BOOLEAN EXPRESSION

YA = YB = (A+B+C+D)*(E+F+G)*(H+J+K)*(L+M+N)
YANOT=YBNOT=NOT([A+B+C+D]*[E+F+G]*
[H+J+K]*[L+M+N])

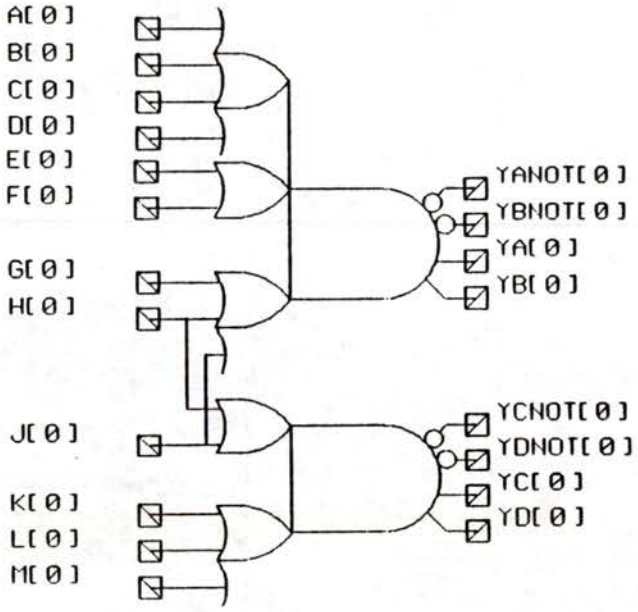
MCA3 SHOW

POWER INFO

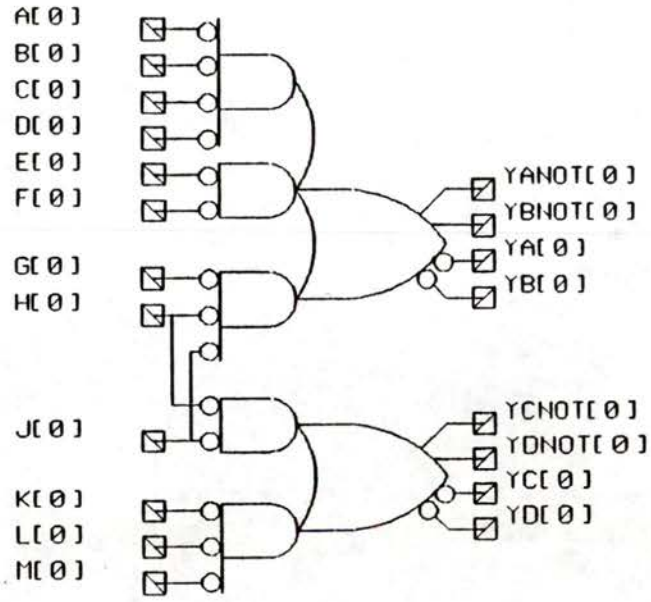
CELL SIZE

1.0

H416



H416I



BOOLEAN EXPRESSION

YA=YB=[A+B+C+D]*[E+F]*[G+H+J]
YANOT=YBNOT=NOT([A+B+C+D]*[E+F]*
[G+H+J]
YC=YD=[H+J]*[K+L+M]
YCNOT=YDNOT=NOT([H+J]*[K+L+M])

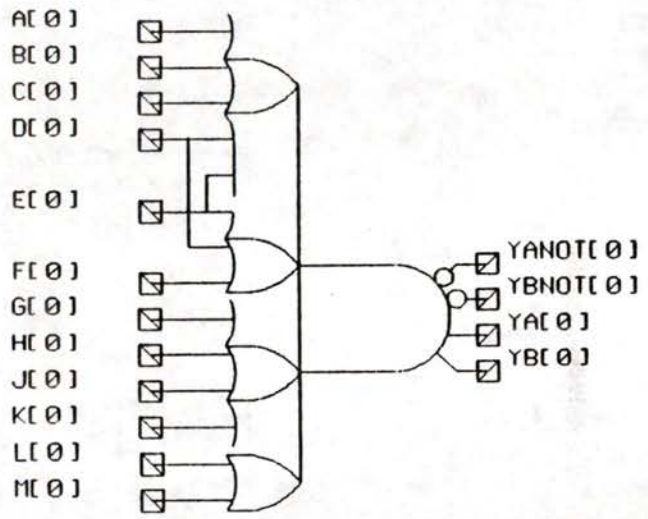
MCA3 SHOW

CELL SIZE

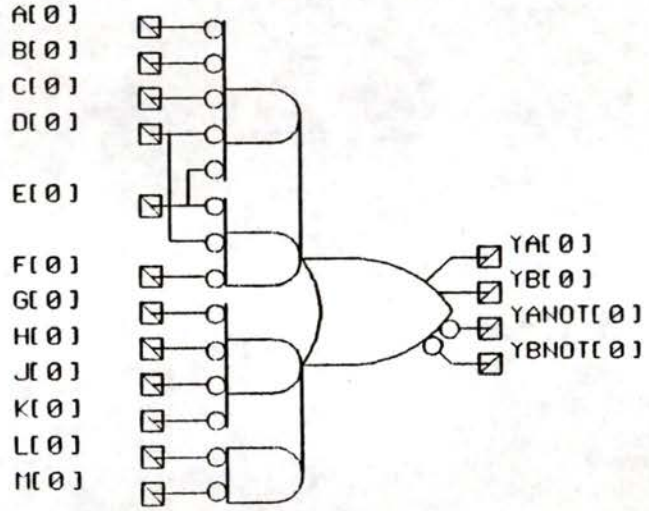
.5

POWER INFO

H417



H4171



BOOLEAN EXPRESSION

YA=YB=(A+B+C+D+E)*(D+E+F)*
(G+H+J+K)*(L+M)

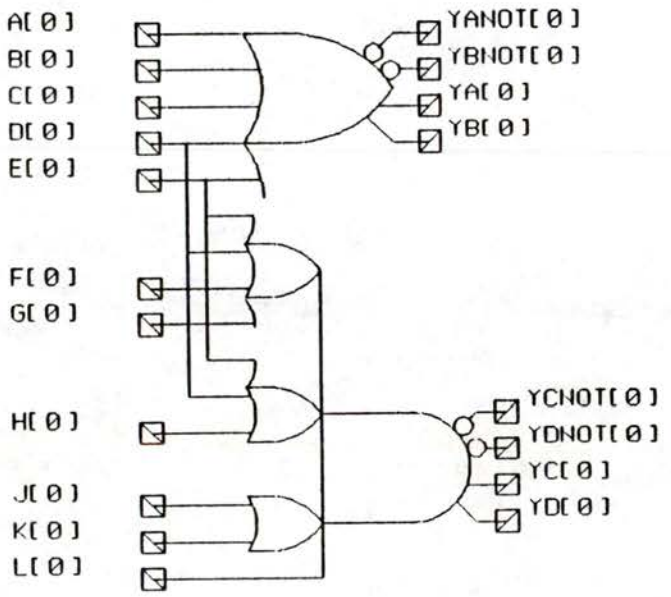
YANOT=YBNOT=NOT((A+B+C+D+E)*
(D+E+F)*(G+H+J+K)*(L+M))

MCA3 SHOW

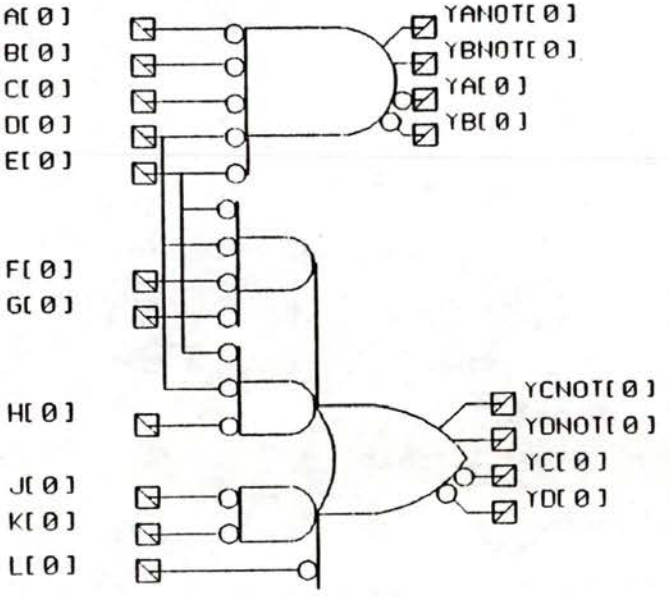
CELL SIZE
.75

POWER INFO

H418



H4181



BOOLEAN EXPRESSION

YA=YB=A+B+C+D+E

YANOT=YBNOT=NOT(A+B+C+D+E)

YC=YD=[D+E+F+G]*[D+E+H]*[J+K]*L

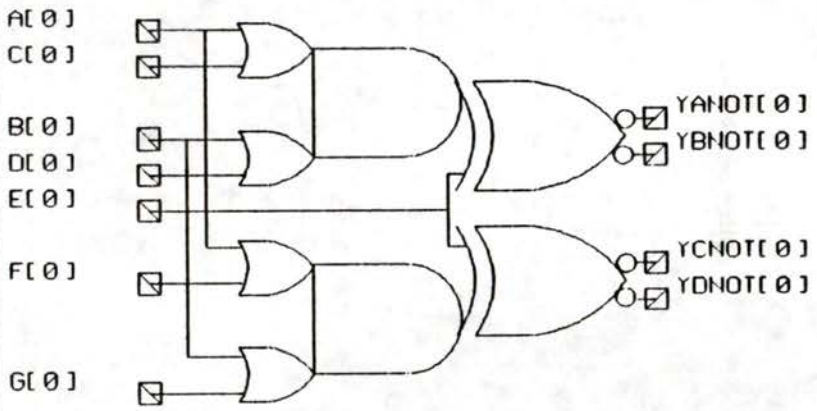
YCNOT=YDNOT=NOT([D+E+F+G]*[D+E+H]*[J+K]*L)

MCA3 SHOW

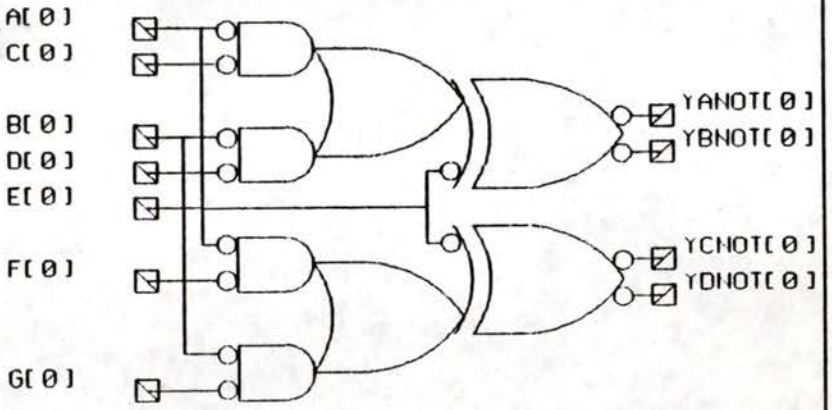
POWER INFO

CELL SIZE
.5

H422



H4221



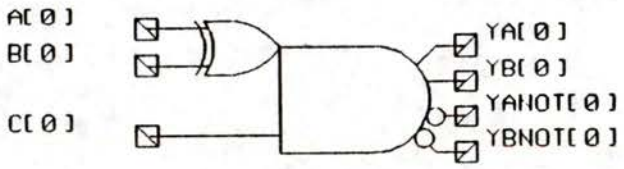
BOOLEAN EXPRESSION
YANOT=YBNOT=(A+B)AND(C+D)XNOR(E)
YCNOT=YDNOT=(A+F)AND(C+G)XOR(E)

MCA3 SHOW

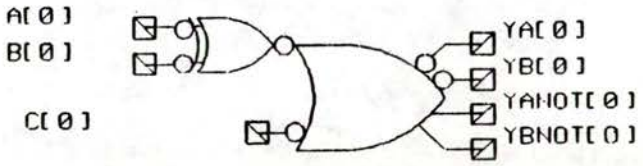
POWER INFO

CELL SIZE
.5

H427



H4271



BOOLEAN EXPRESSION

YA=YB=C AND (A XOR B)

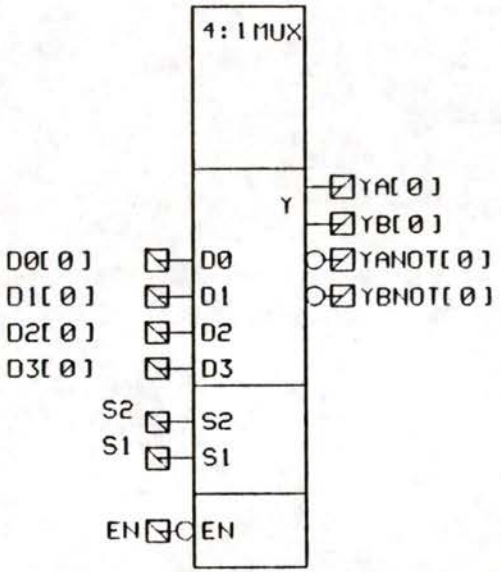
YANOT=YBNOT=NOT(C AND (A XOR B))

MCA3 SHOW

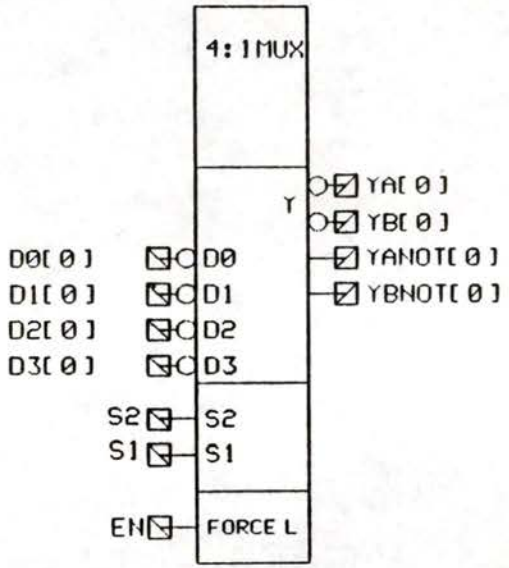
CELL SIZE
.5

POWER INFO

H451



H451I



EN	S2	S1	YA, YB
H	X	X	L
L	L	L	D0
L	L	H	D1
L	H	L	D2
L	H	H	D3

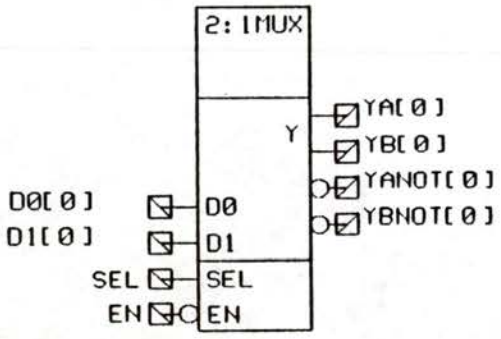
X= DON'T CARE

MCA3 SHOW

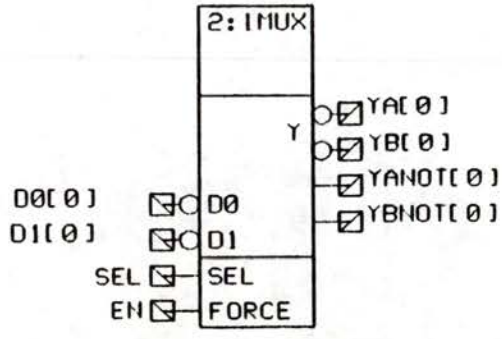
POWER INFO

CELL SIZE
.25

H453



H4531



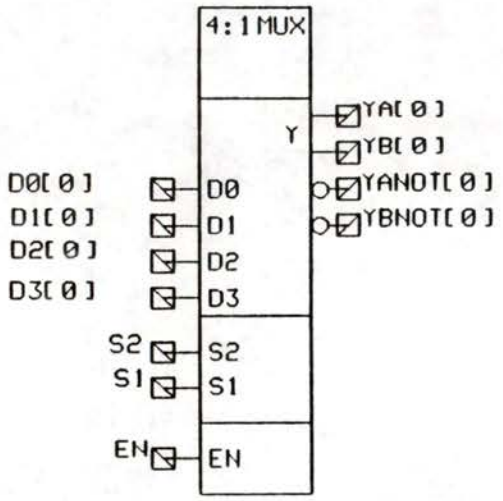
BOOLEAN EXPRESSION		
SEL	EN	YA/YB
X	H	L
L	L	D0
H	L	D1

MCA3 SHOW

POWER INFO

CELL SIZE
.5

H458



NO DeMORGAN'S EQUIVALENT

BOOLEAN EXPRESSION

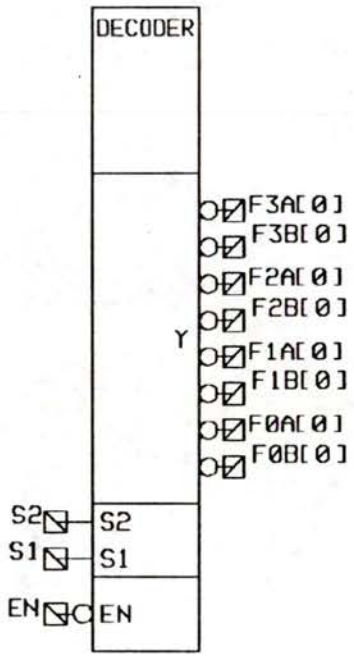
S2	S1	EN	YA/YB
X	X	L	L
L	L	H	D0
L	H	H	D1
H	L	H	D2
H	H	H	D3

MCA3 SHOW

POWER INFO

CELL SIZE
.5

H461



NO DeMORGAN'S EQUIVALENT

BOOLEAN EXPRESSION

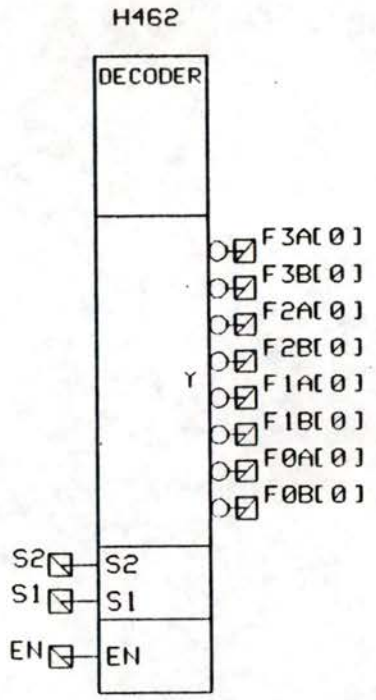
S2	S1	EN	F3	F2	F1	F0
X	X	H	H	H	H	H
L	L	L	H	H	H	L
L	H	L	H	H	L	H
H	L	L	H	L	H	H
H	H	L	L	H	H	H

MCA3 SHOW

CELL SIZE
.5

POWER INFO

NO DeMORGAN'S EQUIVALENT



EN	S2	S1	F3	F2	F1	F0
L	X	X	H	H	H	H
H	L	L	H	H	H	L
H	L	H	H	H	L	H
H	H	L	H	L	H	H
H	H	H	L	H	H	H

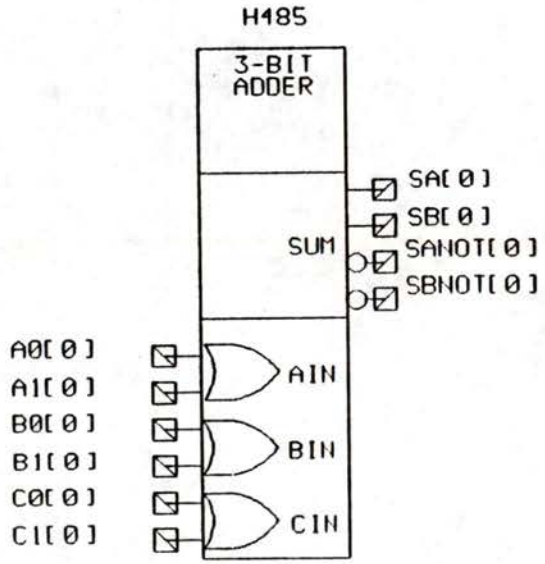
X= DON'T CARE

MCA3 SHOW

CELL SIZE
.5

POWER INFO

NO DeMORGAN'S EQUIVALENT



A0+A1	B0+B1	C0+C1	SA.SB
L	L	L	L
L	L	H	H
L	H	L	H
L	H	H	L
H	L	L	H
H	L	H	L
H	H	L	L
H	H	H	H

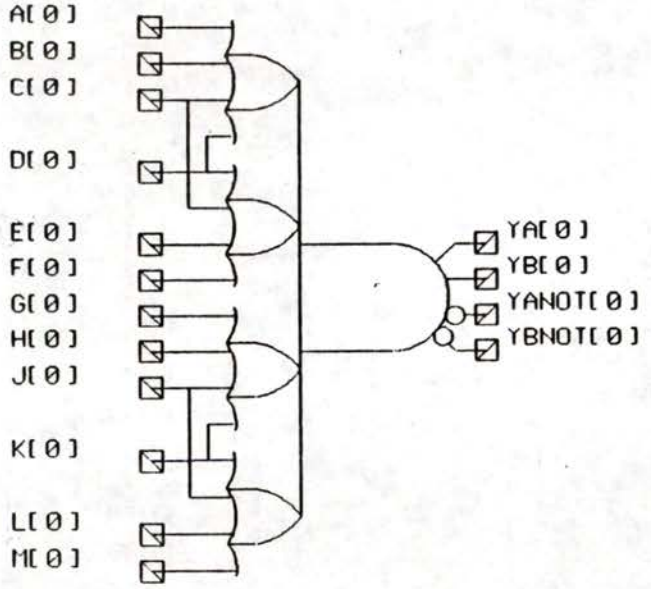
MCA3 SHOW

CELL SIZE

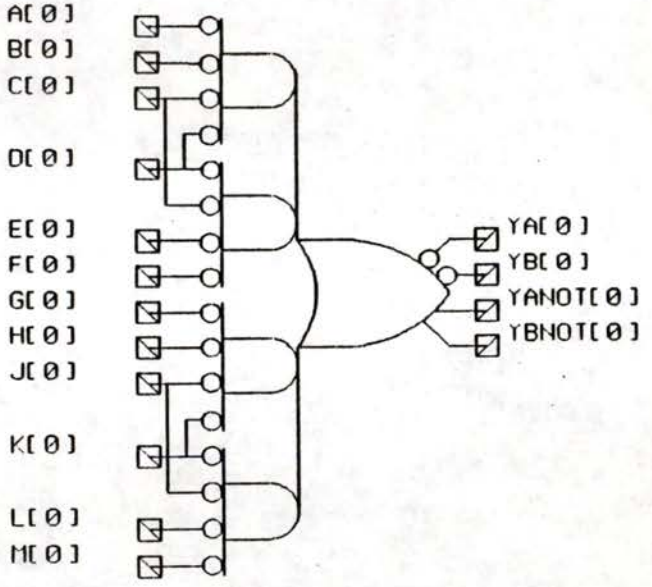
.5

POWER INFO

H510



H5101



BOOLEAN EXPRESSION

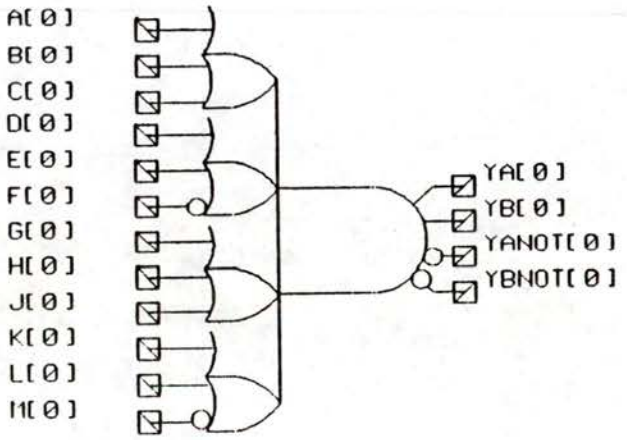
YA=YB=(A+B+C+D)*(C+D+E+F)*
(G+H+J+K)*(J+K+L+M)
YANOT=YBNOT=NOT(A+B+C+D)*(C+D+E+F)*
(G+H+J+K)*(J+K+L+M))

MCA3 SHOW

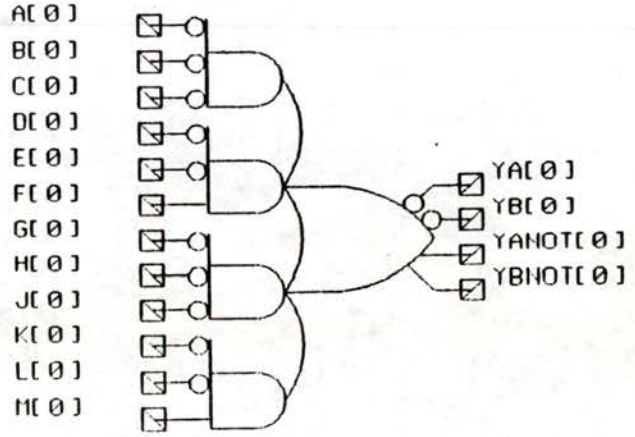
CELL SIZE
.5

POWER INFO

H511



H511



BOOLEAN EXPRESSION

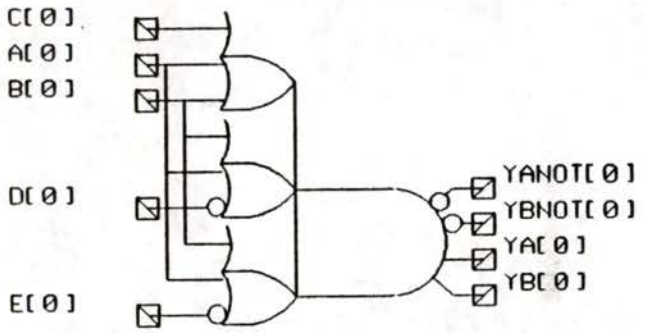
YA=YB=(A+B+C)*(D+E+NOT(F))*(G+H+J)*
(K+L+NOT(M))
YANOT=YBNOT=NOT((A+B+C)*(D+E+NOT(F))*
(G+H+J)*(K+L+NOT(M)))

MCA3 SHOW

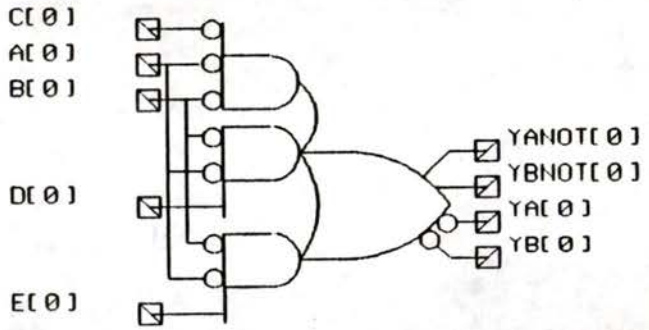
CELL SIZE
.5

POWER INFO

H512



H5121



BOOLEAN EXPRESSION

YA=YB=(A+B+C)*(B+C+NOT(D))*(B+C+NOT(E))
YANOT=YBNOT=NOT(A+B+C)*(B+C+NOT(D))
*(B+C+NOT(E))

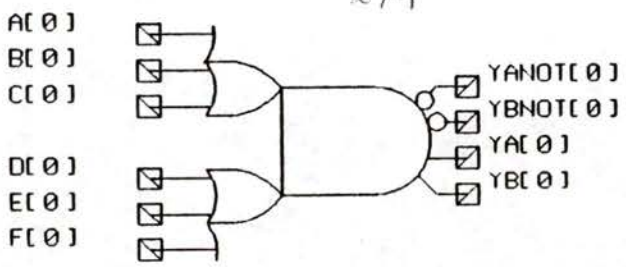
MCA3 SHOW

CELL SIZE
.25

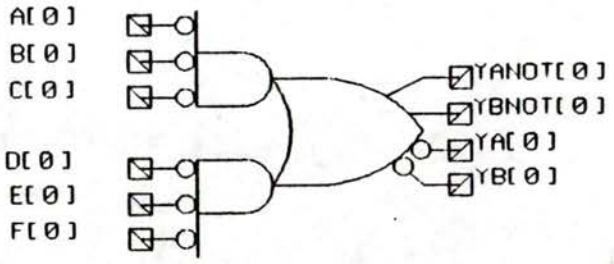
POWER INFO

H518

279



H5181



BOOLEAN EXPRESSION

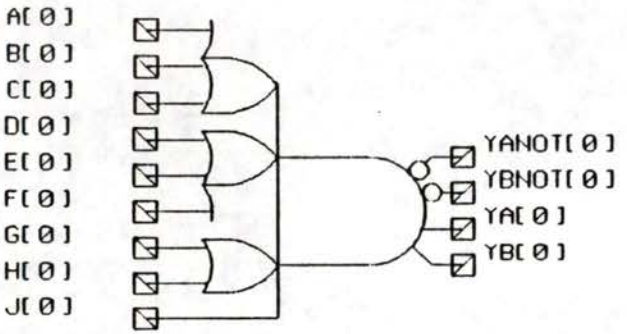
YA=YB=(A+B+C)*(D+E+F)
YANOT=YBNOT=NOT((A+B+C)*(D+E+F))

MCA3 SHOW

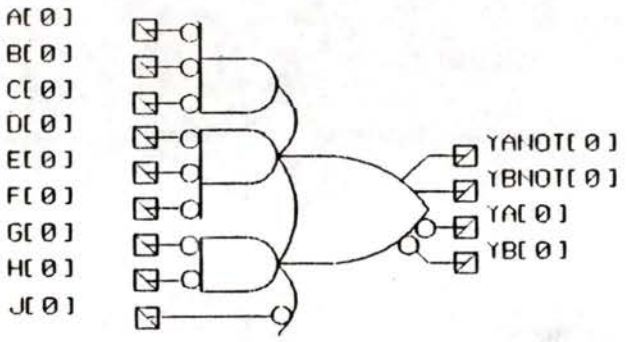
POWER INFO

CELL SIZE
.5

H519



H519I



BOOLEAN EXPRESSION

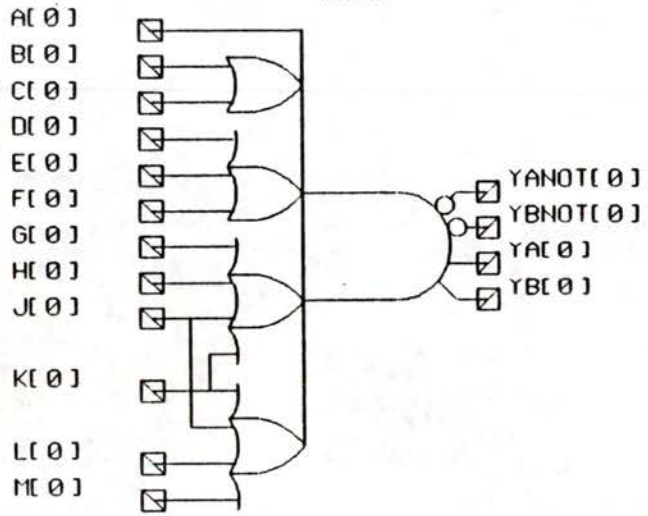
YA=YB=(A+B+C)*(D+E+F)*(G+H)*(J)
YANOT=YBNOT=NOT((A+B+C)*(D+E+F)*(G+H)*(J))

MCA3 SHOW

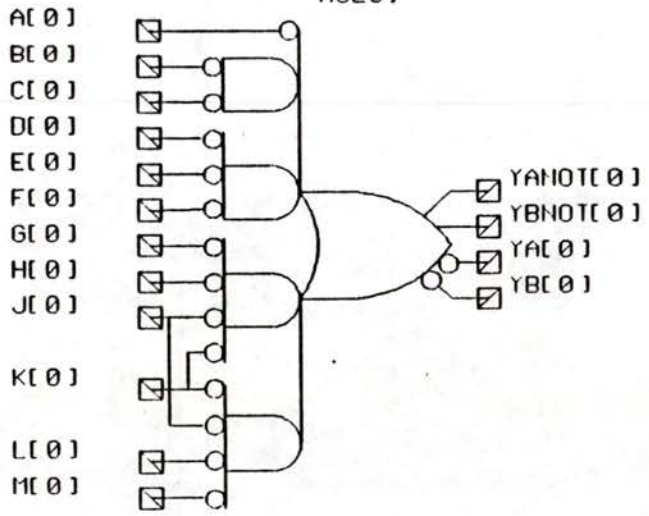
CELL SIZE
.5

POWER INFO

H520



H5201



BOOLEAN EXPRESSION

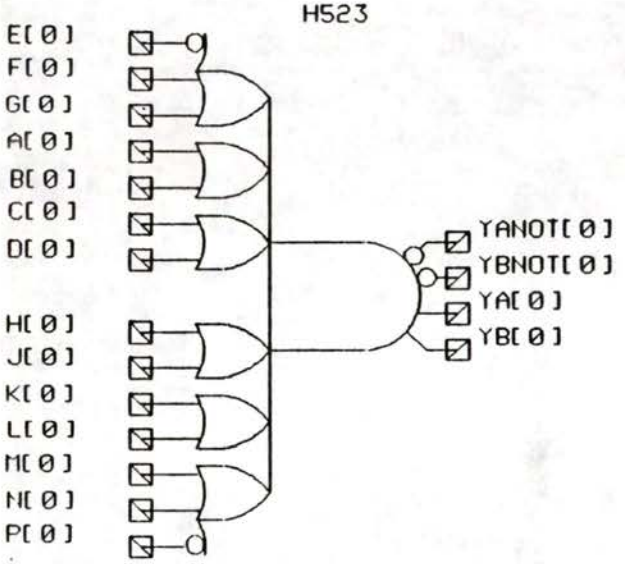
YA=YB=(A)*(B+C)*(D+E+F)*(G+H+J+K)*
(J+K+L+M)
YANOT=YBNOT=NOT((A)*(B+C)*(D+E+F)*
(G+H+J+K)*(J+K+L+M))

MCA3 SHOW

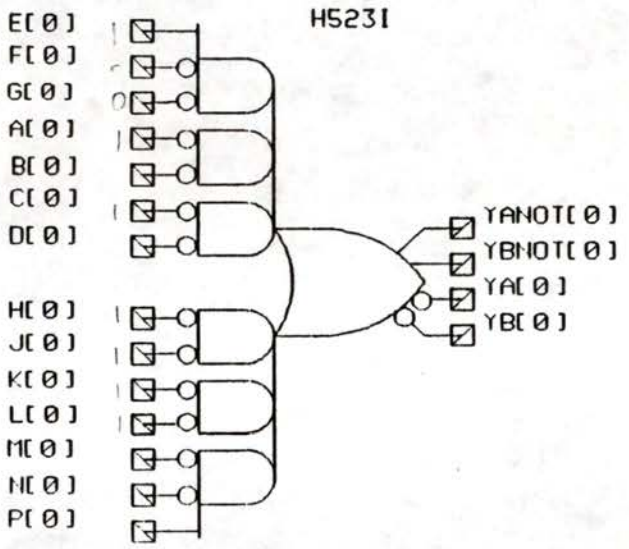
POWER INFO

CELL SIZE
1

H523



H5231



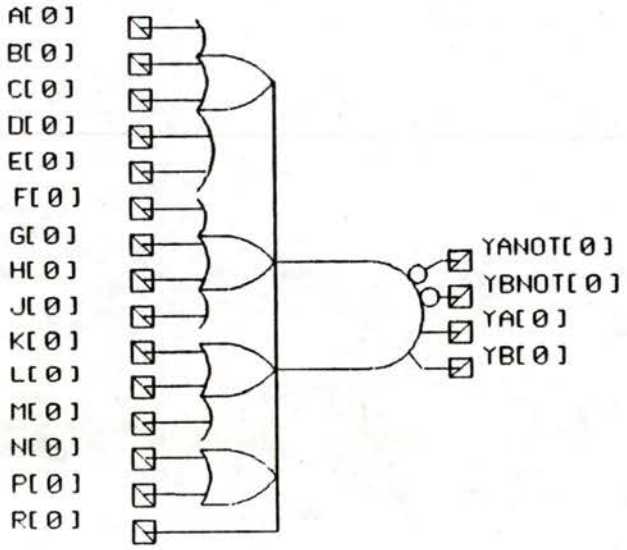
YA=YB=(NOT(E)+F+G)*(A+B)*(C+D)*
(H+J)*(K+L)*(M+N+NOT(P))
YANOT=YBNOT=NOT((NOT(E)+F+G+A+B+C+D+
H+J+K+L+M+N+P)))

MCA3 SHOW

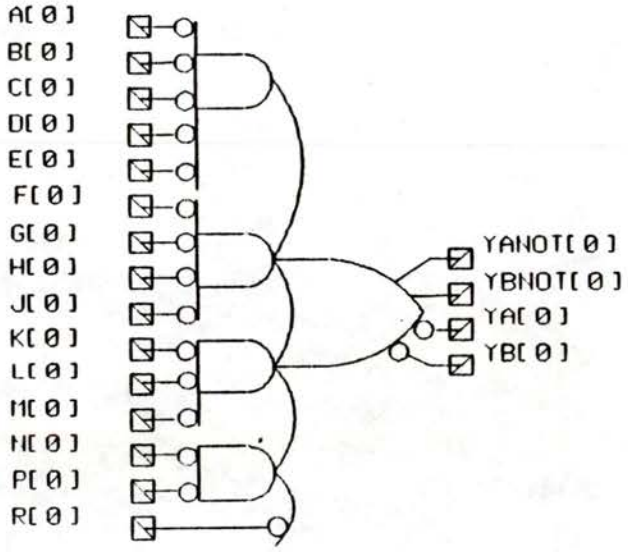
CELL SIZE
.5

POWER INFO

H618



H6181



BOLLEAN EXPRESSION

YA=YB=((A+B+C+D+E)*(F+G+H+J)*(K+L+M)*(N+P)*R)

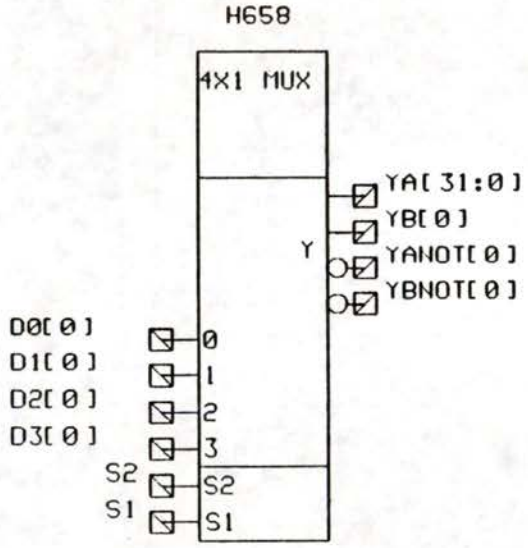
YANOT=YBNOT=NOT((A+B+C+D+E)*(F+G+H+J)*(K+L+M)*(N+P)*R)

MCA3 SHOW

POWER INFO

CELL SIZE
.5

NO DeMORGAN'S EQUIVALENT



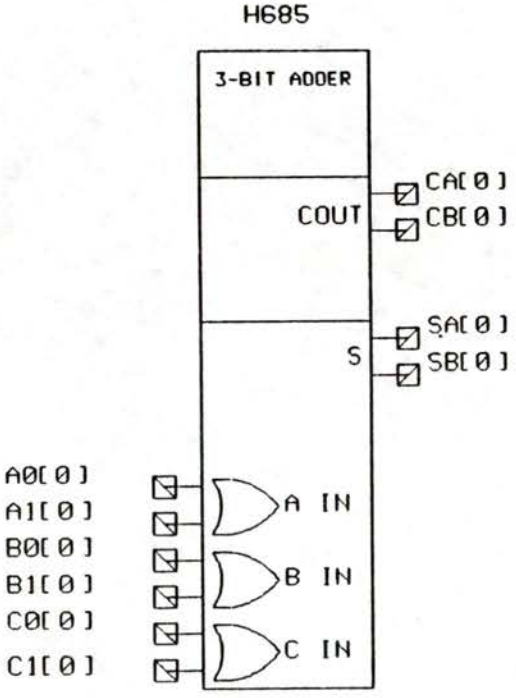
S2	S1	YA, YB
L	L	D0
L	H	D1
H	L	D2
H	H	D3

MCA3 SHOW

POWER INFO

CELL SIZE
.5

NO DeMORGAN'S EQUIVALENT



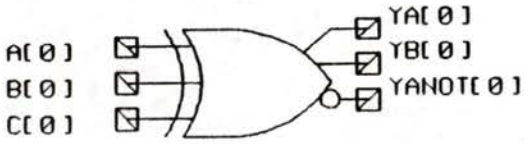
A0+A1	B0+B1	C0+C1	SA.SB	CA.CB
L	L	L	L	L
L	L	H	H	L
L	H	L	H	L
L	H	H	L	H
H	L	L	H	L
H	L	H	L	H
H	H	L	L	H
H	H	H	H	H

MCA3 SHOW

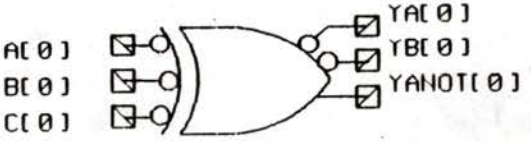
CELL SIZE
.25

POWER INFO

H802



H8021

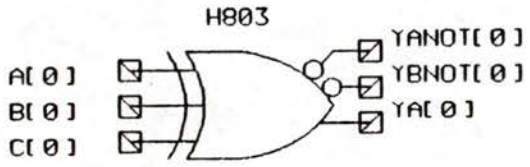


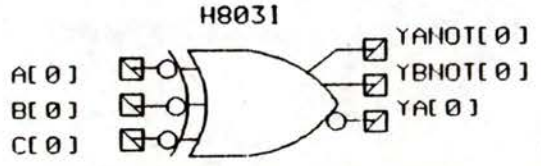
BOOLEAN EXPRESSION
YA=YB=A XOR B XOR C
YANOT=NOT(A XOR B XOR C)

MCA3 SHOW

CELL SIZE
.25

POWER INFO





BOOLEAN EXPRESSION

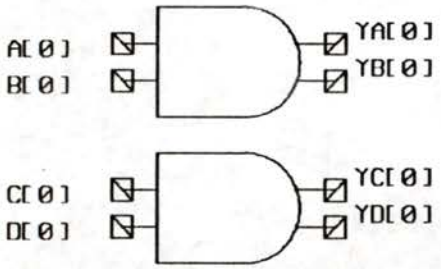
YANOT=YBNOT=NOT(A XOR B XOR C)
YA=A XOR B XOR C

MCA3 SHOW

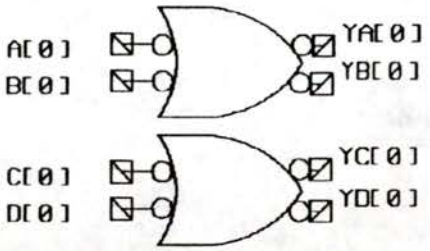
CELL SIZE
.25

POWER INFO

L804



L804I



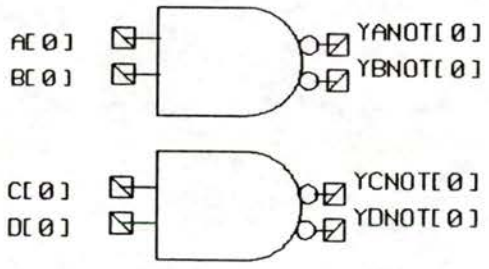
BOOLEAN EXPRESSION
YA=YB=A*B
YC=YD=C*D

MCA3 SHOW

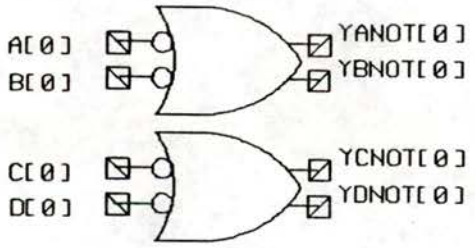
CELL SIZE
.25

POWER INFO

L805



L805I



BOOLEAN EXPRESSION

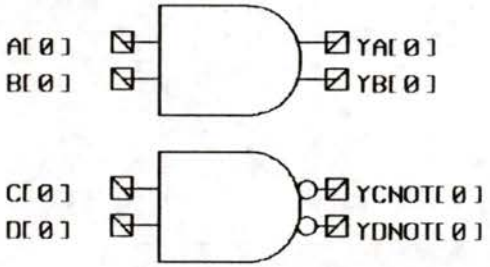
YANOT=YBNOT=NOT(A*B)
YCNOT=YDNOT=NOT(C*D)

MCA3 SHOW

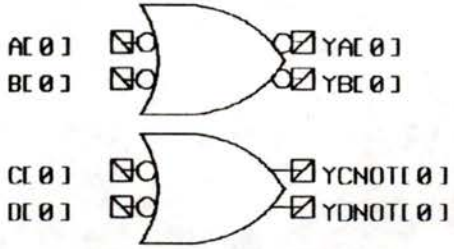
CELL SIZE
.25

POWER INFO

L806



L806I



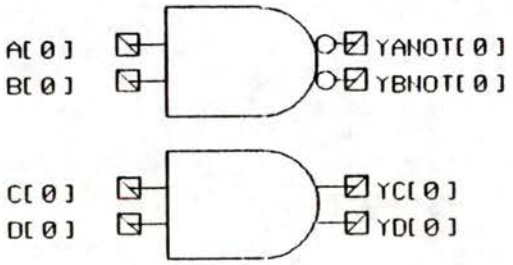
BOOLEAN EXPRESSION
YA=YB=A*B
YCNOT=YDNOT=NOT(C*D)

MCA3 SHOW

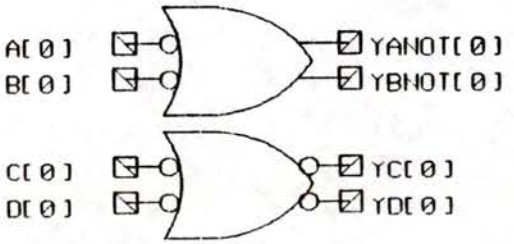
CELL SIZE
.25

POWER INFO

L807



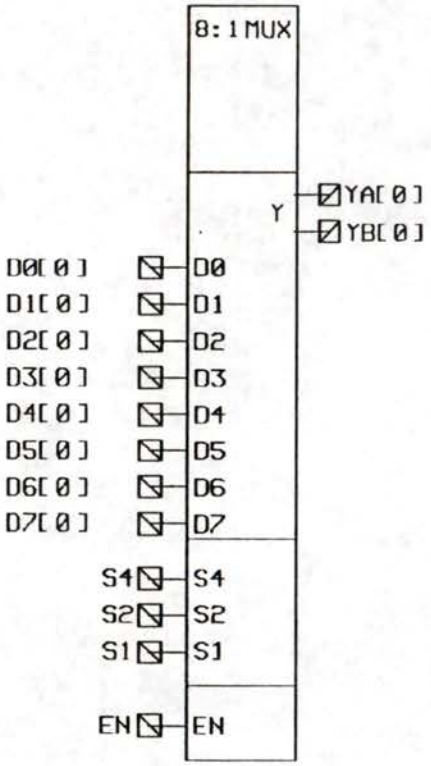
L8071



BOOLEAN EXPRESSION
YANOT=YBNOT=NOT(A*B)
YC=YD=(C*D)

MCA3 SHOW

L809



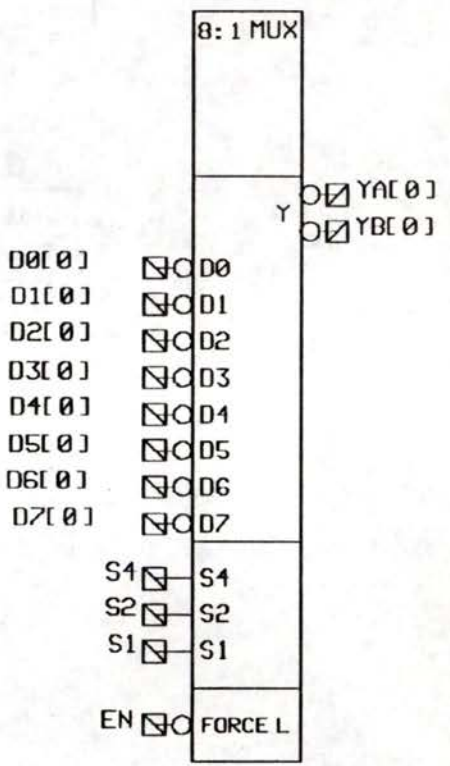
POWER INFO

CELL SIZE
1

BOOLEAN EXPRESSION

S 1	S 2	S 4	E N	YA/YB
X	X	X	L	L
L	L	L	H	D0
L	L	H	H	D1
L	H	L	H	D2
L	H	H	H	D3
H	L	L	H	D4
H	L	H	H	D5
H	H	L	H	D6
H	H	H	H	D7

L809I

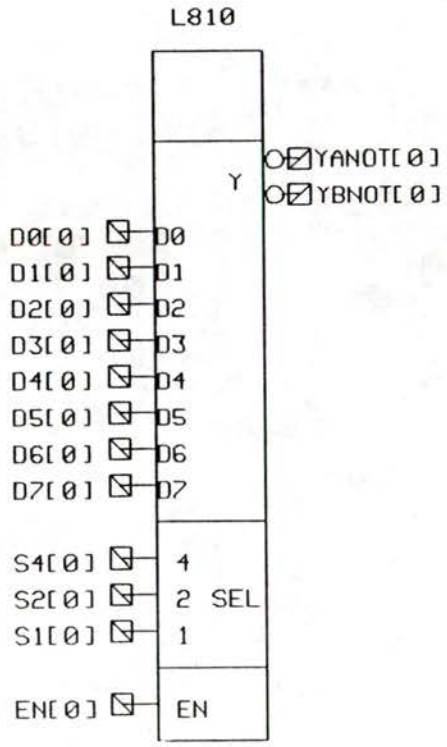


MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT



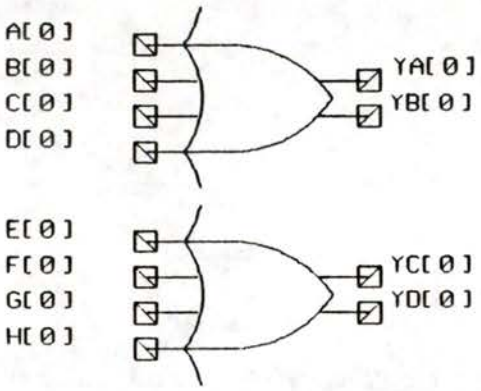
EN	S1	S2	S4	YANOT YBNOT
H	H	H	H	NOT D7
H	H	H	L	NOT D6
H	H	L	H	NOT D5
H	H	L	L	NOT D4
H	L	H	H	NOT D3
H	L	H	L	NOT D2
H	L	L	H	NOT D1
H	L	L	L	NOT D0
L	X	X	X	H

MCA3 SHOW

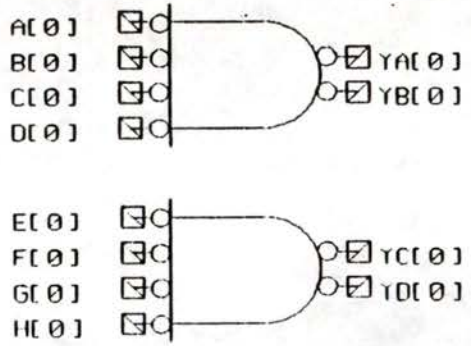
CELL SIZE
.25

POWER INFO

L811



L811



BOOLEAN EXPRESSION

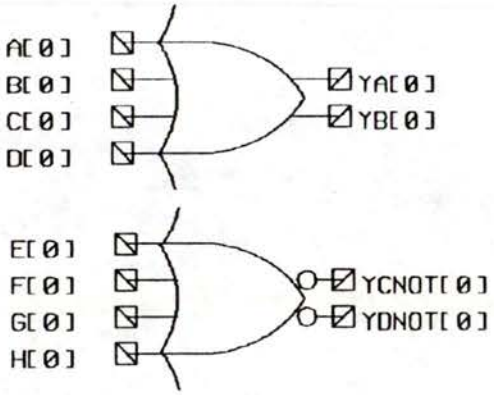
YA=YB=A+B+C+D
YC=YD=E+F+G+H

MCA3 SHOW

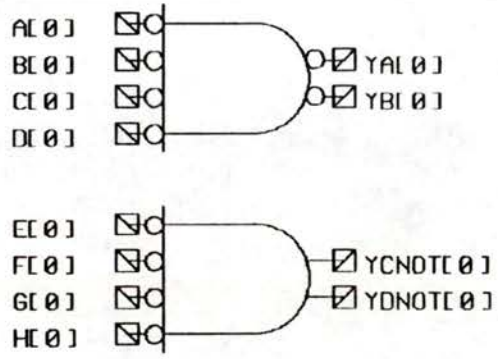
CELL SIZE
.25

POWER INFO

L812



L812I



BOOLEAN EXPRESSION

YA=YB=(A+B+C+D)

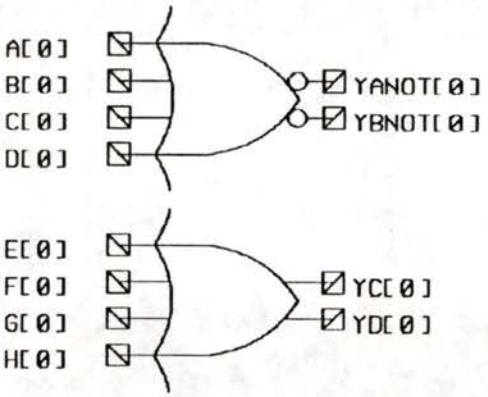
YCNOT=YDNOT=NOT(E+F+G+H)

MCA3 SHOW

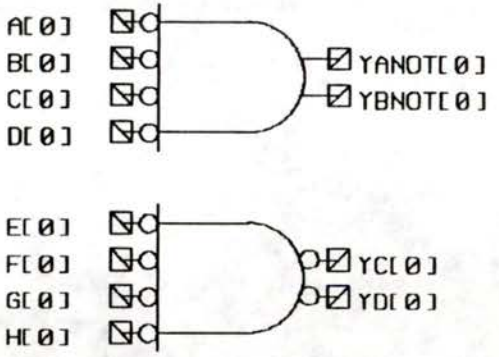
CELL SIZE
.25

POWER INFO

L813



L8131



BOOLEAN EXPRESSION

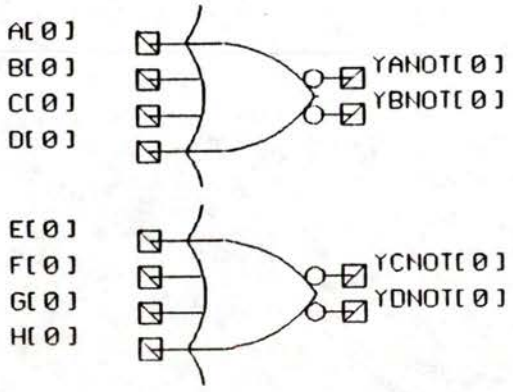
YANOT=$\overline{YBNOT}$=NOT(A+B+C+D)
YC=YD=(E+F+G+H)

MCA3 SHOW

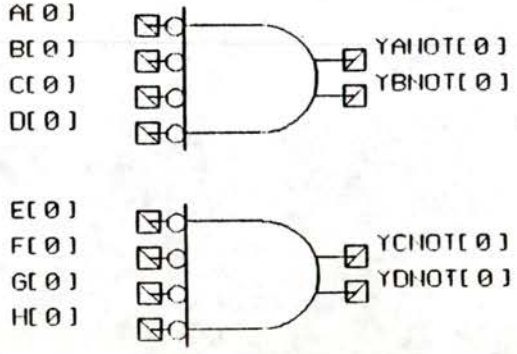
CELL SIZE
.25

POWER INFO

L814



L814I



BOOLEAN EXPRESSION

YANOT=YBNOT=NOT(A+B+C+D)
YCNOT=YDNOT=NOT(E+F+G+H)

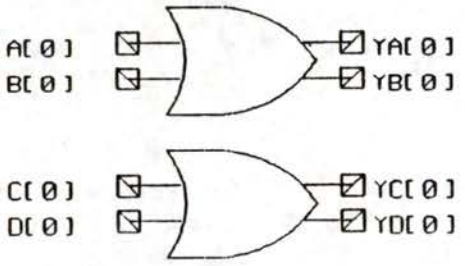
MCA3 SHOW

POWER INFO

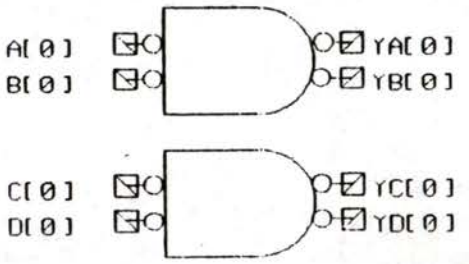
CELL SIZE

.25

L815



L8151



BOOLEAN EXPPESSION

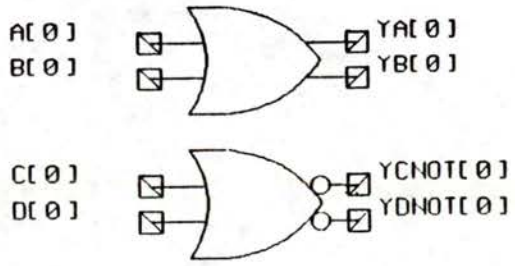
YA=YB=A+B
YC=YD=C+D

MCA3 SHOW

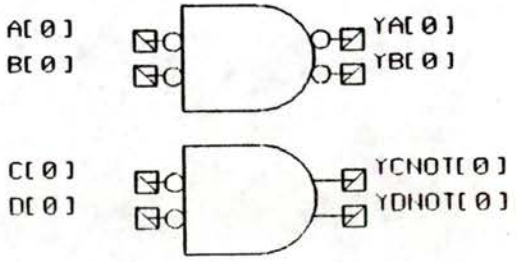
POWER INFO

CELL SIZE
.25

L816



L816I



BOOLEAN EXPRESSION

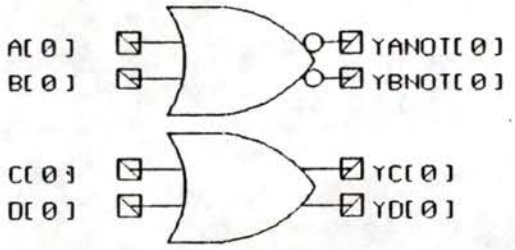
YA=YB=A+B
YCNOT=YDNOT=NOT(C+D)

MCA3 SHOW

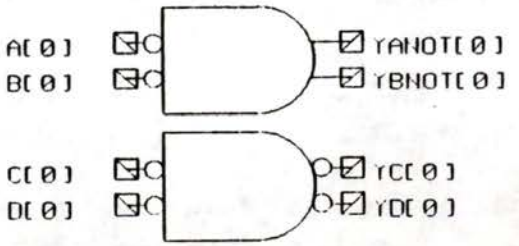
CELL SIZE
.25

POWER INFO

L817



L8171



BOOLEAN EXPRESSION

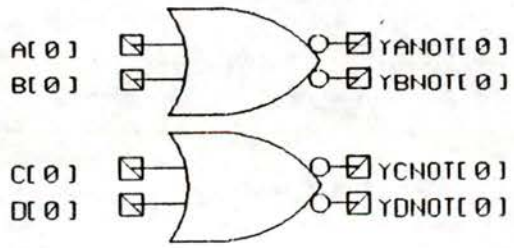
YANOT=YBNOT=NOT(A+B) YC=YD=C+D

MCA3 SHOW

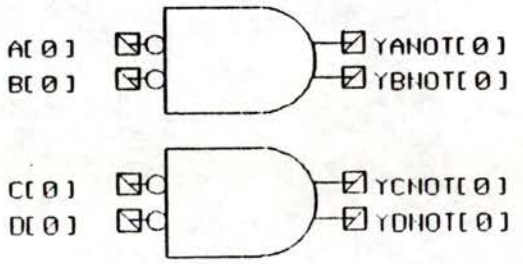
CELL SIZE
.25

POWER INFO

L818



L8181



BOOLEAN EXPRESSION

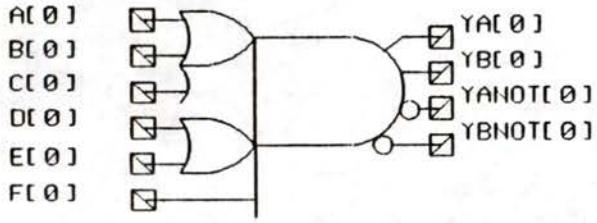
YANOT=YBNOT=NOT(A+B)
YCNOT=YDNOT=NOT(C+D)

MCA3 SHOW

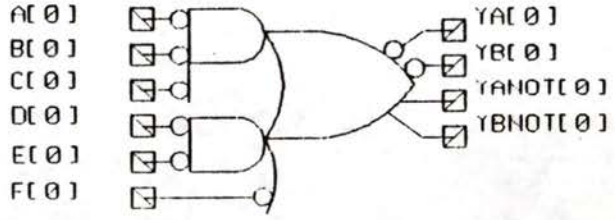
CELL SIZE
.25

POWER INFO

H819



H8191



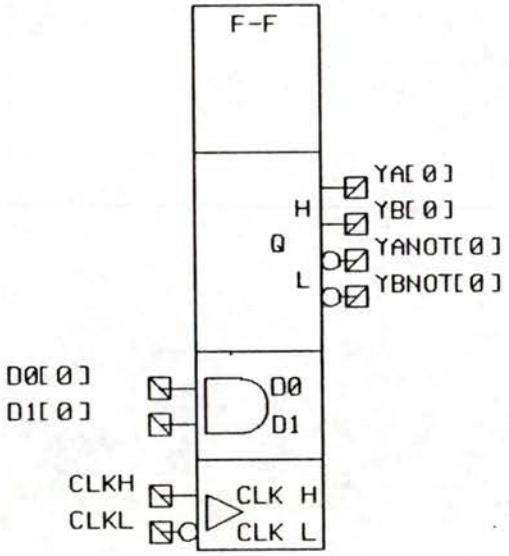
YA=YB=((A OR B OR C)AND(D OR E)AND F)
YANOT=YBNOT=NOT((A OR B OR C)AND(D OR E)AND F)

MCA3 SHOW

POWER INFO

CELL SIZE
.75

H941



NO DeMORGAN'S EQUIVALENT

BOOLEAN EXPRESSION

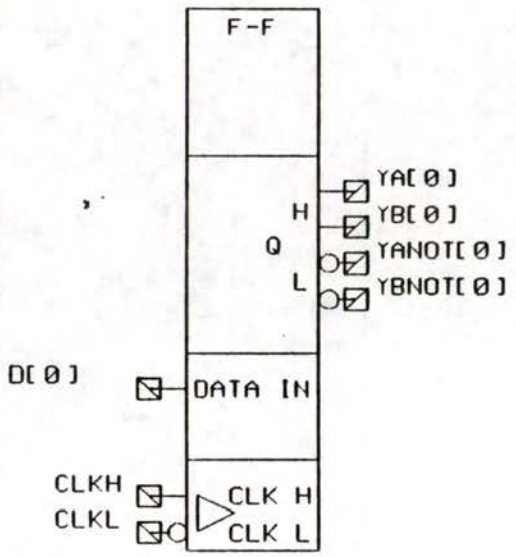
CLKH	D0	D1	YA/YB	YANOT/YBNOT
L→ H	L	L	L	H
L→ H	L	H	L	H
L→ H	H	L	L	H
L→ H	H	H	H	L

MCA3 SHOW

CELL SIZE
.5

H942

POWER INFO



NO DeMORGAN'S EQUIVALENT

CLKH	D	YA/YB	YANOT/YBNOT
L ⟶ H	L	L	H
L ⟶ H	H	H	L

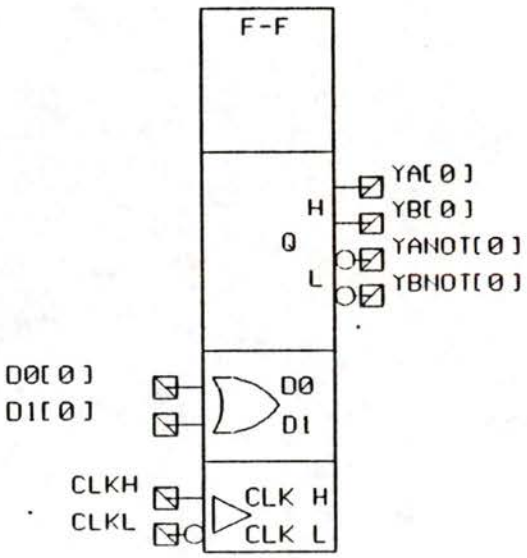
MCA3 SHOW

POWER INFO

CELL SIZE
.5

NO DeMORGAN'S EQUIVALENT

H943



BOOLEAN EXPRESSION

CLKH	D0	D1	YA/YB	YANOT/YBNOT
L→ H	L	L	L	H
L→ H	L	H	H	L
L→ H	H	L	H	L
L→ H	H	H	H	L

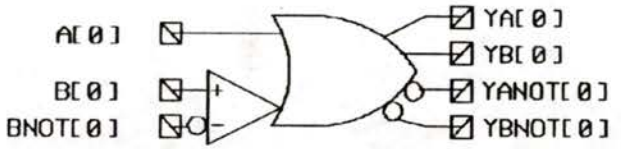
MCA3 SHOW

CELL SIZE
.25

POWER INFO

NO DeMORGAN'S EQUIVALENT

H944



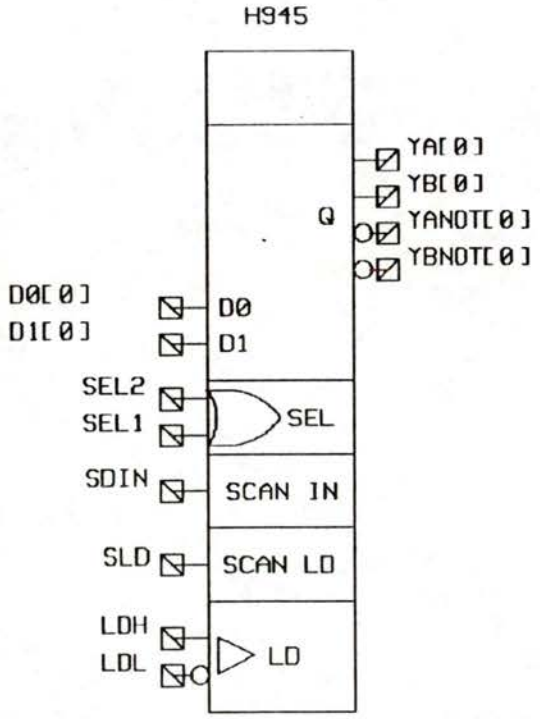
BOOLEAN EXPRESSION
YA=YB=A+(B*NOT(BNOT))
YANOT=YBNOT=NOT(A+(B*NOT(BNOT)))

MCA3 SHOW

CELL SIZE
.5

POWER INFO

NO DeMORGAN'S EQUIVALENT



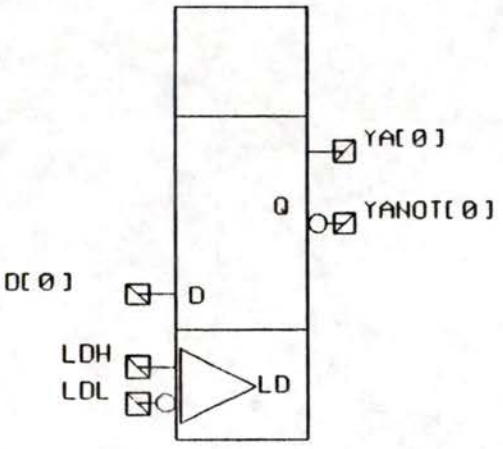
BOOLEAN EXPRESSION					
SEL1 OR SEL2	SLD	LDH	LDL	YA, YB	YANOT, YBNOT
X	L	L	H	-	-
X	H	L	H	SDIN	NOT SDIN
L	X	H	L	D0	NOT D0
H	X	H	L	D1	NOT D1

MCA3 SHOW

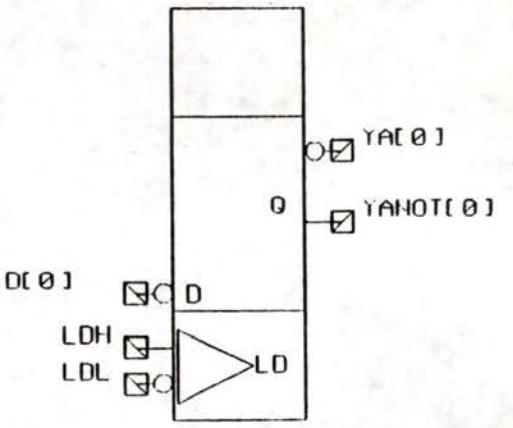
POWER INFO

CELL SIZE
.25

H946

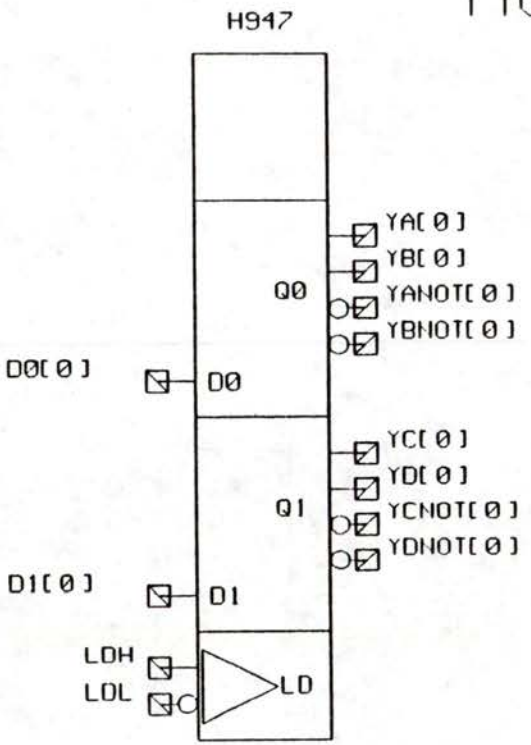


H946I



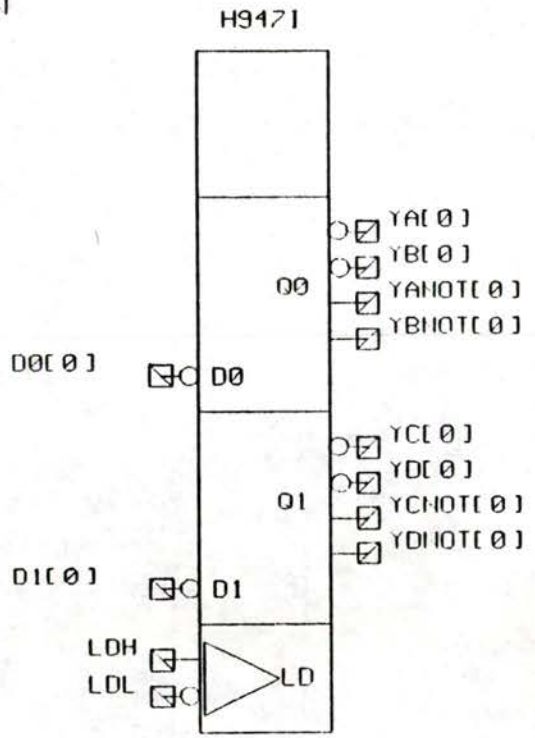
BOOLEAN EXPRESSION			
LDH	LDL	YA	YANOT
L	H	--	---
H	L	D	NOT(D)

MCA3 SHOW



POWER INFO

CELL SIZE
.5



BOOLEAN EXPRESSION

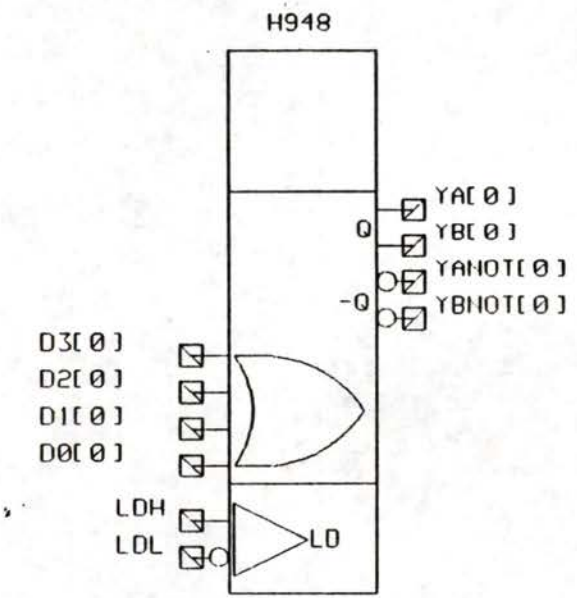
LDH	LDL	D0,D1	YA,YB,YC,YD	YANOT,YBNOT YCNOT,YDNOT
L	H	X	-	-
H	L	L	L	H
H	L	H	H	L

MCA3 SHOW

CELL SIZE
.5

POWER INFO

NO DeMORGAN'S EQUIVALENT



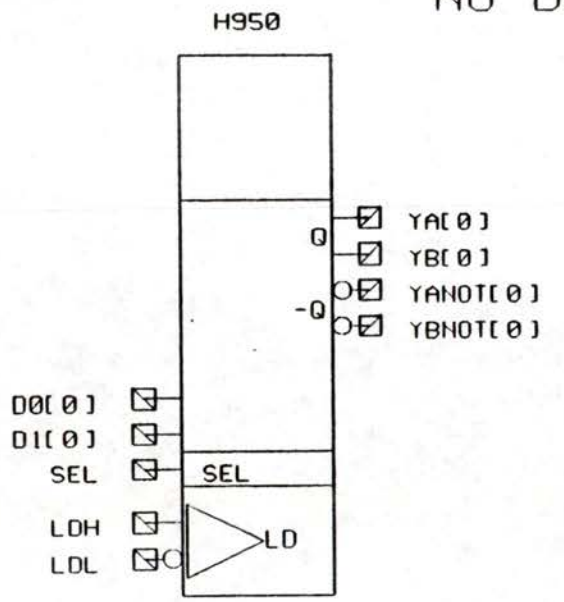
LDH	LDL	YA, YB
L	H	----------
H	L	D0+D1+D2+D3

MCA3 SHOW

CELL SIZE
.5

POWER INFO

NO DeMORGAN'S EQUIVALENT



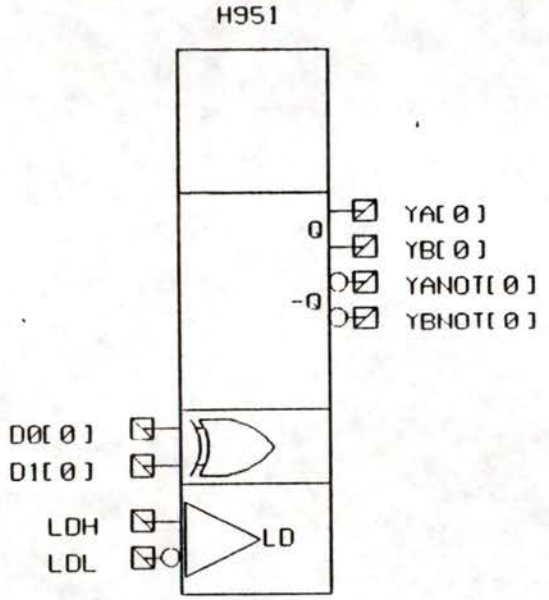
LDH	LDL	SEL	YA, YB
L	H	X	---
H	L	L	D0
H	L	H	D1

MCA3 SHOW

CELL SIZE
.5

POWER INFO

NO DeMORGAN'S EQUIVALENT



LDH	LDL	YA, YB
L	H	---
H	L	D0 XOR D1

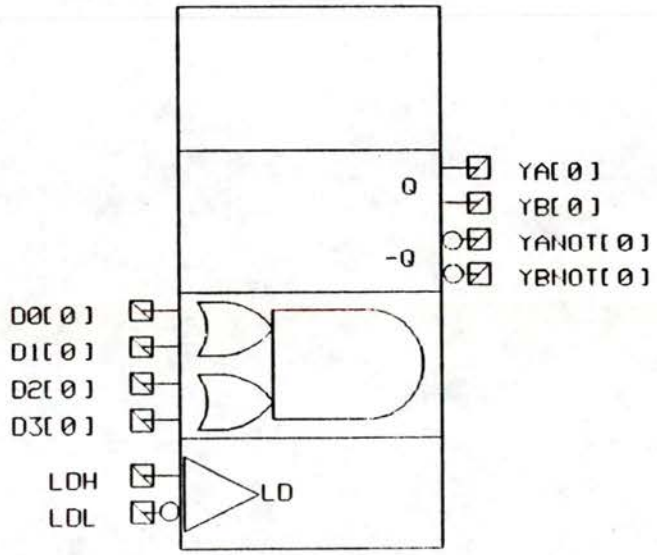
MCA3 SHOW

CELL SIZE
.5

POWER INFO

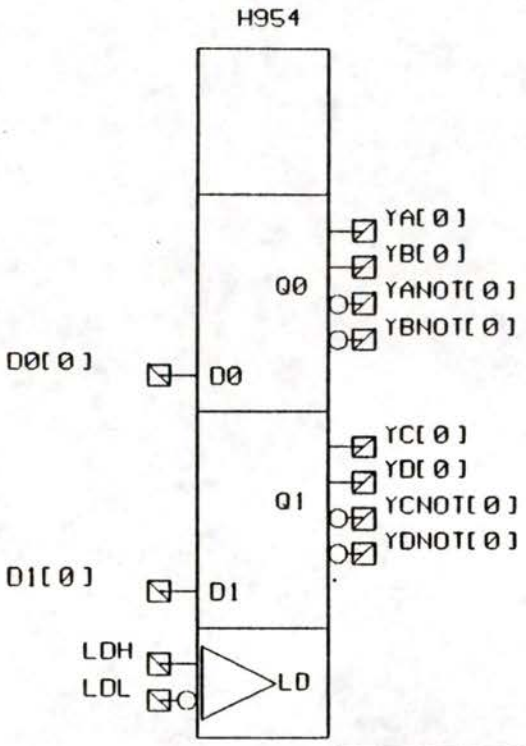
NO DeMORGAN'S EQUIVALENT

H952



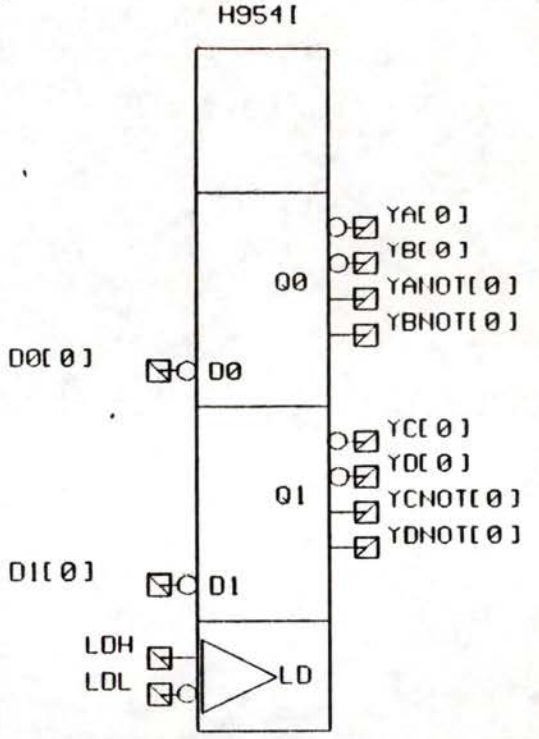
LDH	LDL	YA, YB
L	H	---
H	L	(D0+D1)*(D2+D3)

MCA3 SHOW



POWER INFO

CELL SIZE
.5



BOOLEAN EXPRESSION

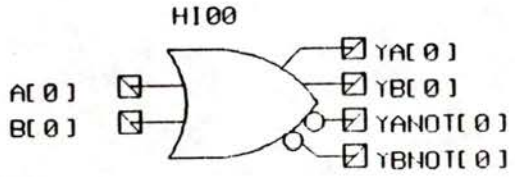
LDH	LDL	D0,D1	YA,YB,YC,YD	YANOT,YBNOT YCNOT,YDNOT
L	H	X	-	-
H	L	L	L	H
H	L	H	H	L

MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT



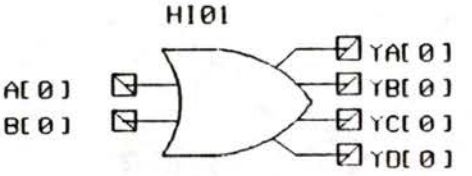
BOOLEAN EXPRESSION
YA=YB=A+B
YANOT=YBNOT=NOT(A+B)

MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT



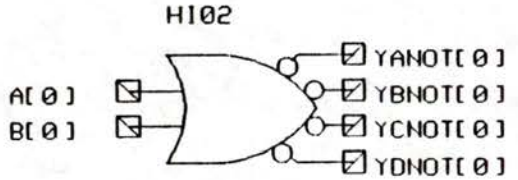
BOOLEAN EXPRESSION
YA=YB=YC=YD=A+B

MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT



BOOLEAN EXPRESSION

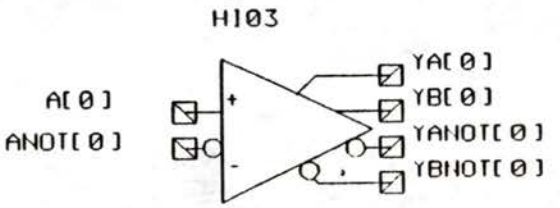
YANOT=YBNOT=YCNOT=YDNOT=NOT(A+B)

MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT



BOOLEAN EXPRESSION
YA=YB=A
YANOT=YBNOT=NOT(A)

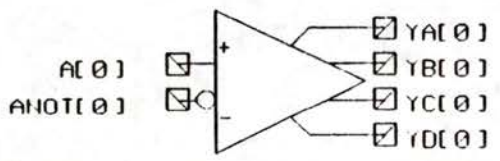
MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT

HI04



BOOLEANS EXPRESSION
YA=YB=YC=YD=A

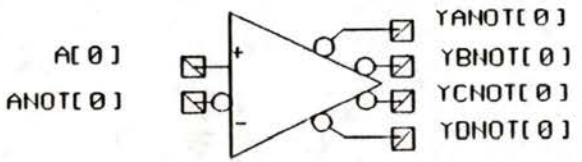
MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT

HI05



BOOLEAN EXPRESSION
YANOT=YBNOT=YCNOT=YDNOT=NOT(A)

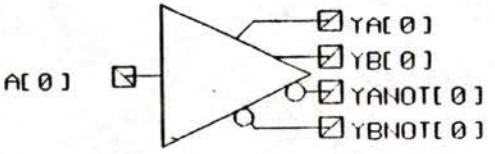
MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT

H106



BOOLEAN EXPRESSION
YA=YB=A
YANOT=YBNOT=NOT(A)

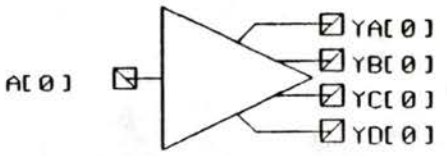
MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT

HI07



BOOLEAN EXPRESSION
YA=YB=YC=YD=A

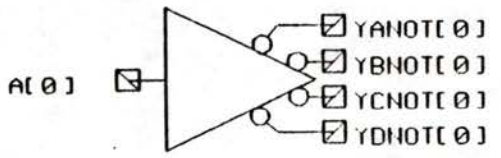
MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT

HI08



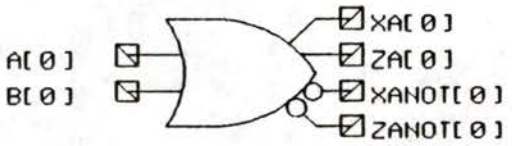
BOOLEAN EXPRESSION
YANOT=YBNOT=YCNOT=YDNOT=NOT(A)

MCA3 SHOW

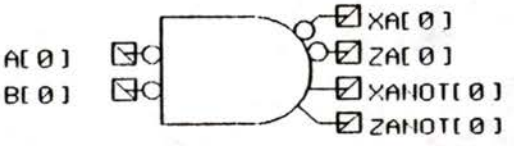
CELL SIZE
1

POWER INFO

HX01



HX011



BOOLEAN EXPRESSION

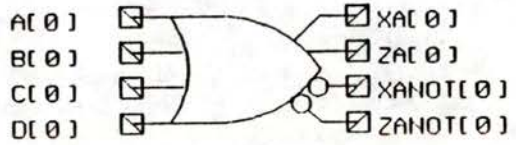
XA=ZA=(A+B)
XANOT=ZANOT=NOT(A+B)

MCA3 SHOW

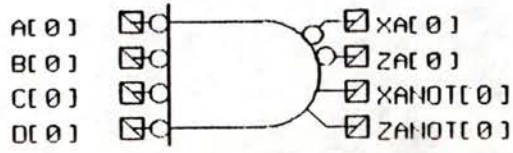
CELL SIZE
1

POWER INFO

HX02



HX021



BOOLEAN EXPRESSION

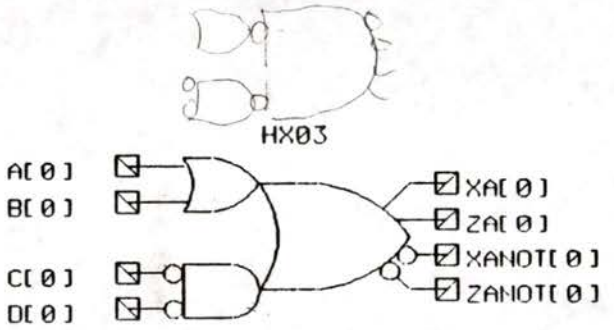
XA=ZA=(A+B+C+D)
XANOT=ZANOT=NOT(A+B+C+D)

MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT



BOOLEAN EXPPESSION

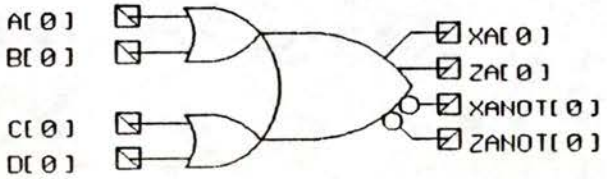
```
XA=ZA=A+B+(NOT(C)*NOT(D))
XANOT=ZANOT=NOT(A+B+(NOT(C)*NOT(D)))
```

MCA3 SHOW

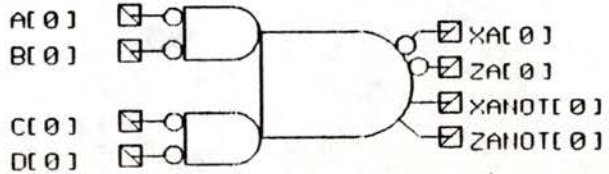
CELL SIZE
1

POWER INFO

HX04



HX041



BOOLEAN EXPRESSION

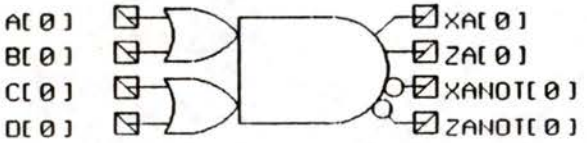
XA=ZA=(A+B+C+D)
XANOT=ZANOT=NOT(A+B+C+D)

MCA3 SHOW

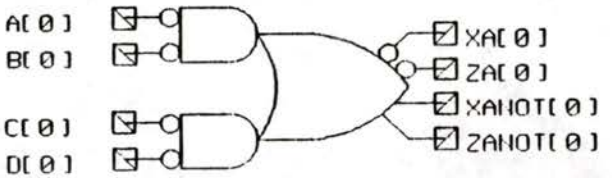
CELL SIZE
1

POWER INFO

HX11



HX111



BOOLEAN EXPRESSION

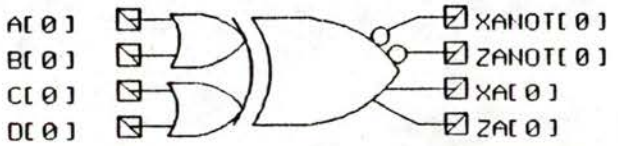
XA=ZA=(A+B)*(C+D)
XANOT=ZANOT=NOT((A+B)*(C+D))

MCA3 SHOW

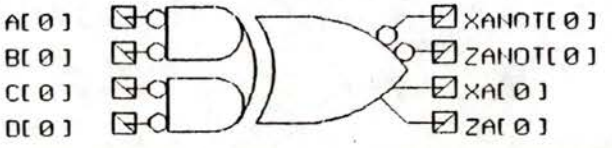
CELL SIZE
1

POWER INFO

HX21



HX211



BOOLEAN EXPRESSION

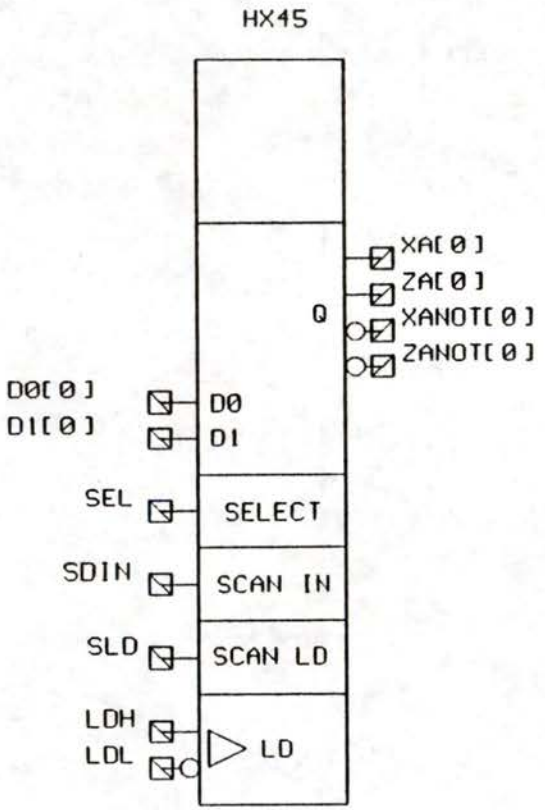
XA=ZA=(A+B)XOR(C+D)
XANOT=ZANOT=NOT((A+B)XOR(C+D))

MCA3 SHOW

POWER INFO

CELL SIZE
2

NO DeMORGAN'S EQUIVALENT



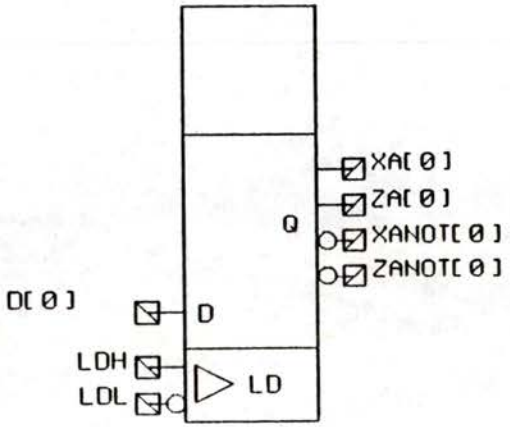
BOOLEAN EXPRESSION					
SEL	SLD	LDH	LDL	XA, ZA	XANOT, YANOT
X	L	L	H	-	-
X	H	L	H	SIN	NOT SIN
L	X	H	L	D0	NOT D0
H	X	H	L	D1	NOT D1

MCA3 SHOW

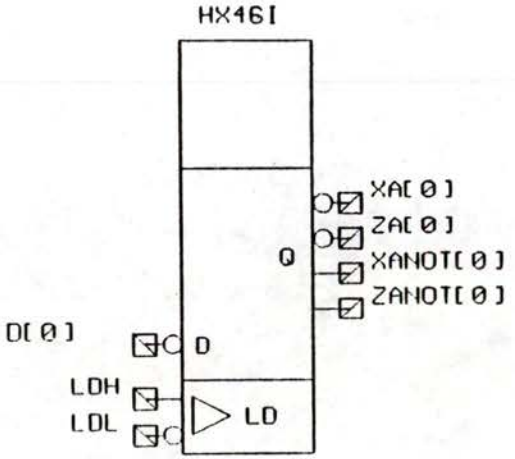
CELL SIZE
1

POWER INFO

HX46

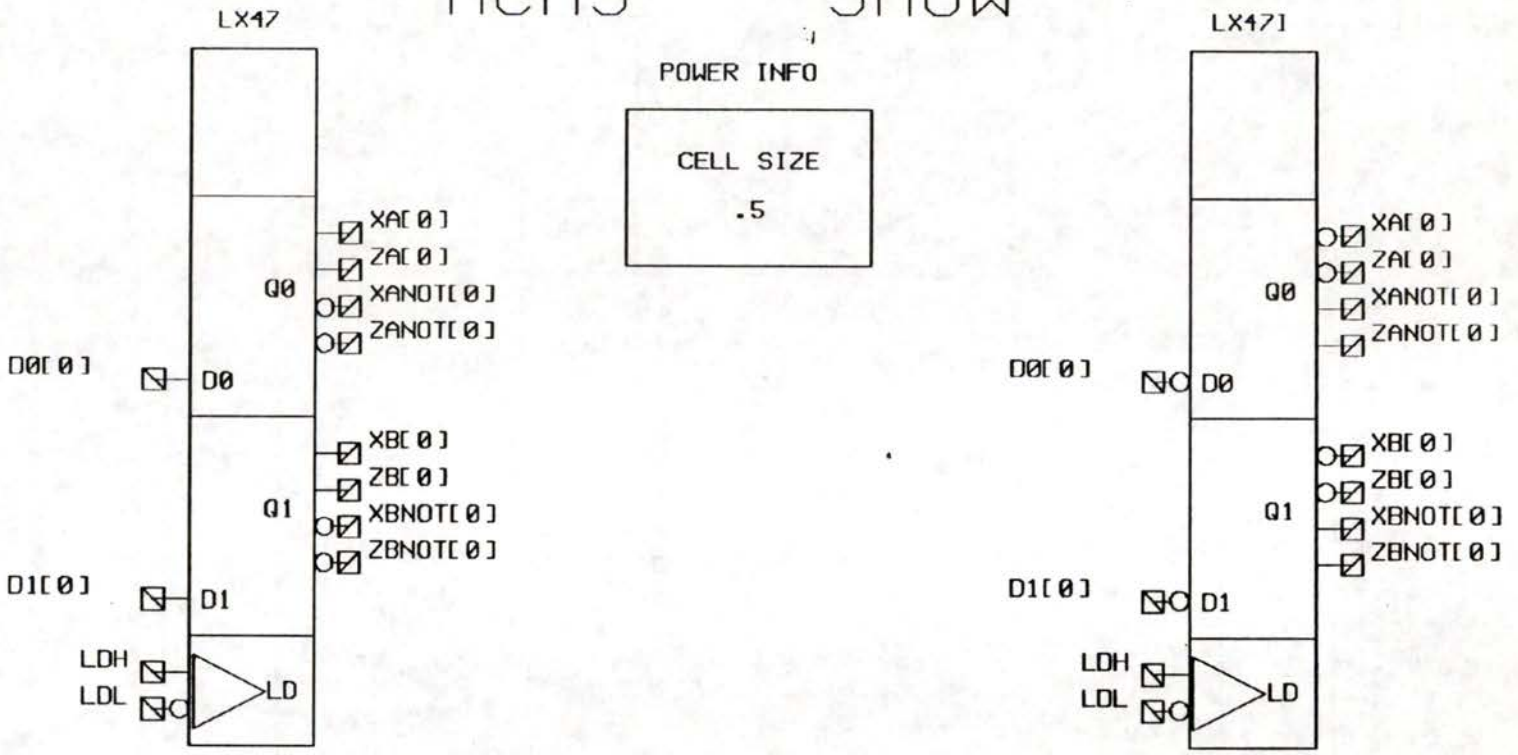


HX46I



BOOLEAN EXPRESSION			
LDH	LDL	XA, ZA	XANOT, ZANOT
L	H	-	-
H	L	D	NOT D

MCA3 SHOW



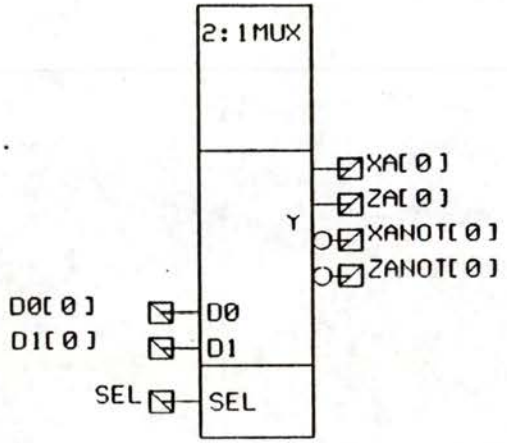
BOOLEAN EXPRESSION				
LDH	LDL	D0, D1	XA, ZA, XB, ZB	XANOT, ZANOT XBNOT, ZBNOT
L	H	X	-	-
H	L	L	L	H
H	L	H	H	L

MCA3 SHOW

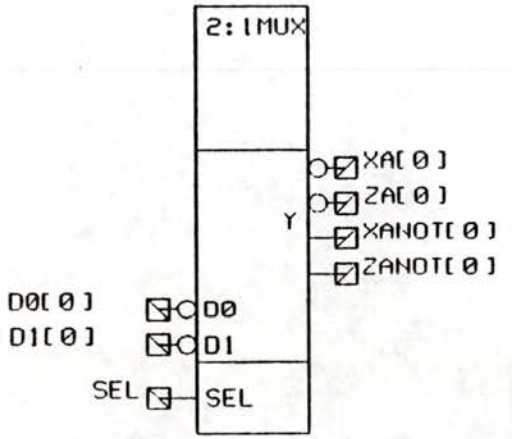
CELL SIZE
1

POWER INFO

HX51



HX511



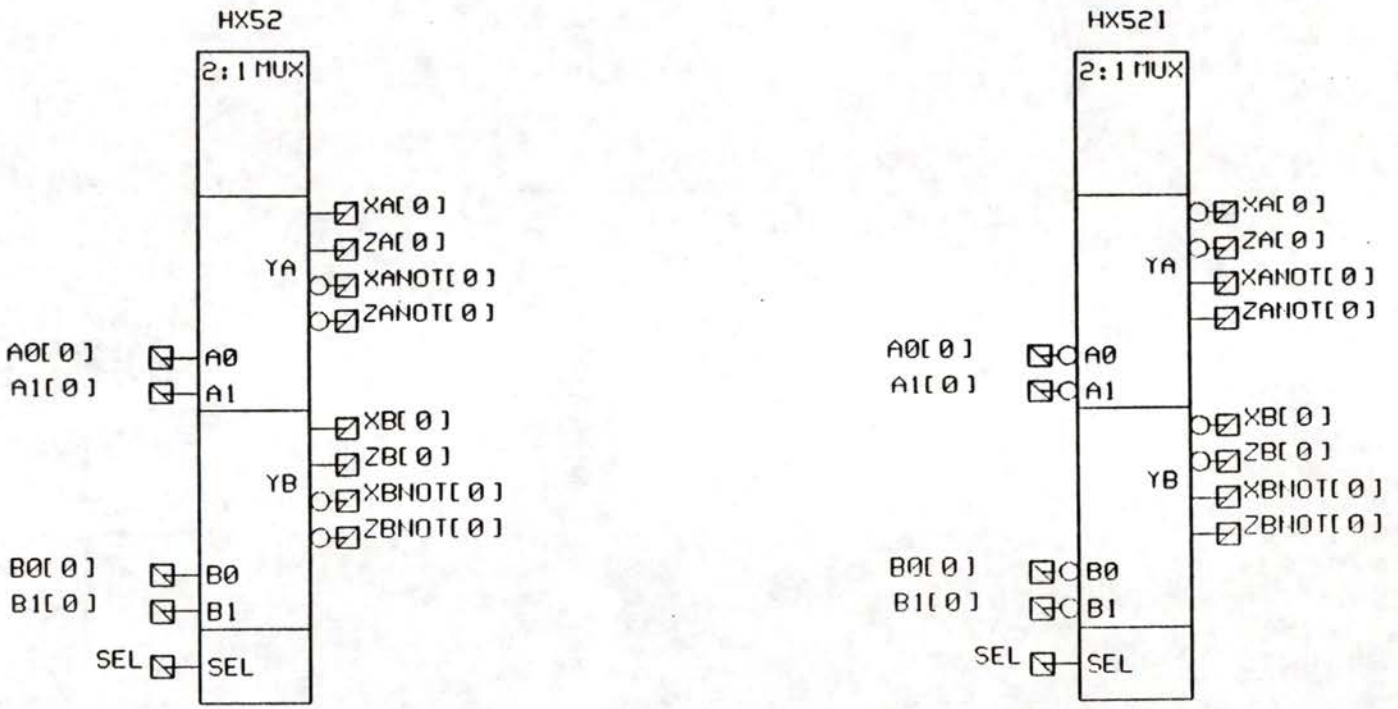
BOOLEAN EXPRESSION

SEL	XA/ZA	XANOT/ZANOT
L	D0	NOT(D0)
H	D1	NOT(D1)

MCA3 SHOW

CELL SIZE
2

POWER INFO



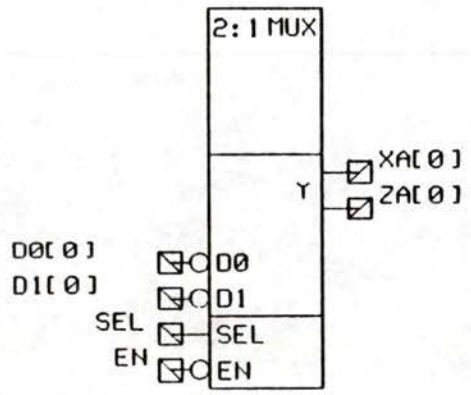
SEL	XA/ZA	XB/ZB	XANOT/ZANOT	XBNOT/ZBNOT
L	A0	B0	NOT(A0)	NOT(B0)
H	A1	B1	NOT(A1)	NOT(B1)

MCA3 SHOW

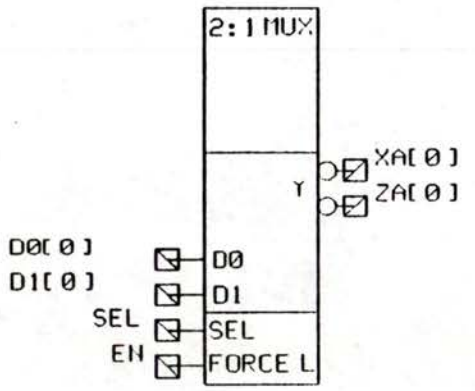
CELL SIZE
1

POWER INFO

HX53



HX531



BOOLEAN EXPRESSION

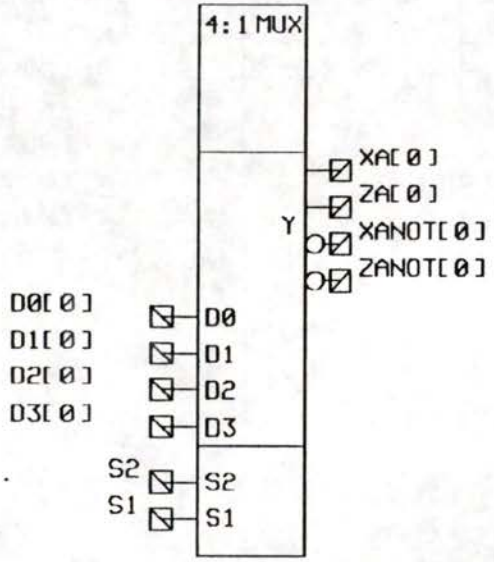
SEL	EN	XA/ZA
X	H	L
L	L	D0
H	L	D1

MCA3 SHOW

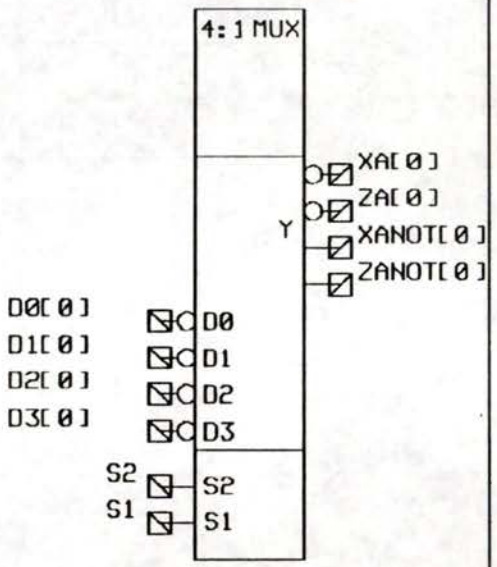
CELL SIZE
2

POWER INFO

HX58



HX58I



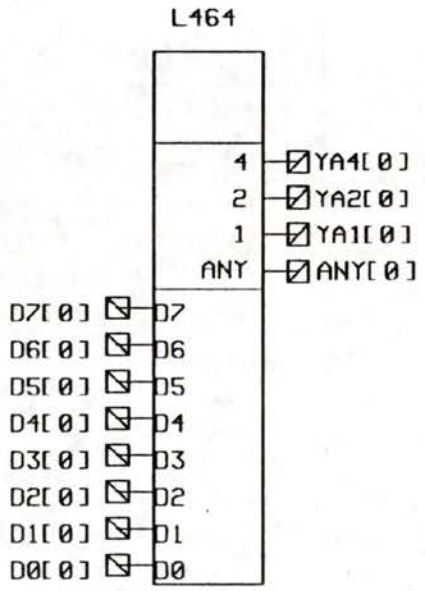
BOOLEAN EXPRESSION			
S2	S1	XA, ZA	XANOT, ZANOT
L	L	D0	NOT(D0)
L	H	D1	NOT(D1)
H	L	D2	NOT(D2)
H	H	D3	NOT(D3)

MCA3 SHOW

CELL SIZE
1

POWER INFO

NO DeMORGAN'S EQUIVALENT



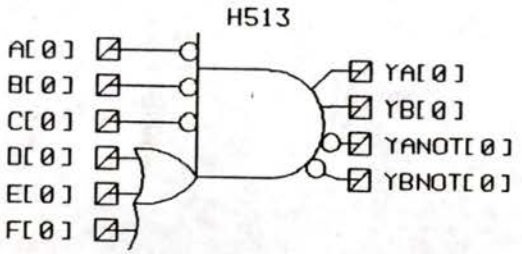
D7	D6	D5	D4	D3	D2	D1	D0	YA4	YA2	YA1	ANY
H	X	X	X	X	X	X	X	H	H	H	H
L	H	X	X	X	X	X	X	H	H	L	H
L	L	H	X	X	X	X	X	H	L	H	H
L	L	L	H	X	X	X	X	H	L	L	H
L	L	L	L	H	X	X	X	L	H	H	H
L	L	L	L	L	H	X	X	L	H	L	H
L	L	L	L	L	L	H	X	L	L	H	H
L	L	L	L	L	L	L	H	L	L	L	H
L	L	L	L	L	L	L	L	L	L	L	L

MCA3 SHOW

CELL SIZE
.25

POWER INFO

NO DeMORGAN'S EQUIVALENT



BOOLEAN EXPRESSION

YA=YB=(NOT(A)*NOT(B)*NOT(C)*(D+E+F))

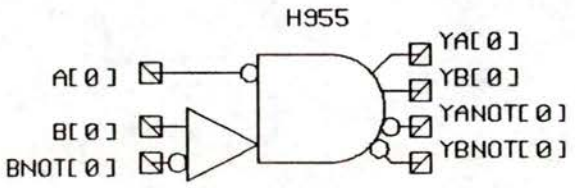
YANOT=YBNOT=NOT(NOT(A)*NOT(B)*NOT(C)*(D+E+F))

MCA3 SHOW

CELL SIZE
.25

POWER INFO

NO DeMORGAN'S EQUIVALENT



BOOLEAN EXPRESSION

YA=YB=NOT(A)*(B*NOT(BNOT))

YANOT=YBNOT=NOT(NOT(A)*(B*NOT(BNOT)))

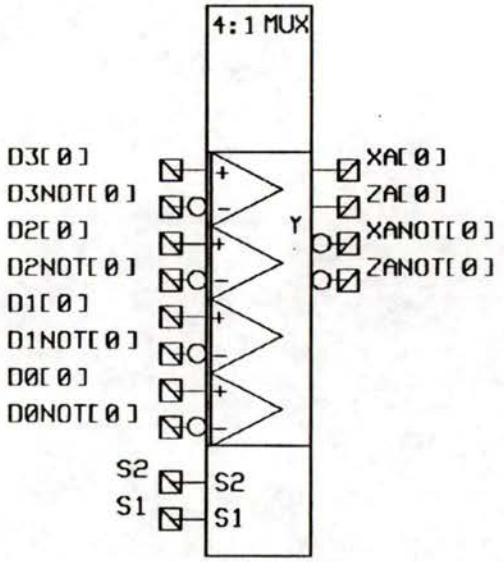
MCA3 SHOW

CELL SIZE
2

POWER INFO

NO DeMORGAN'S EQUIVALENT

HX48



S2	S1	XA ZA	XANOT ZANOT
H	H	D3	NOT D3
H	L	D2	NOT D2
L	H	D1	NOT D1
L	L	D0	NOT D0

From: AQUA::HPSCAD::KECK "Dale MRO1-3/T2 297-4106" 5-MAR-1991 13:58:41.30
To: MXZLPX::Fontaine, Aqua::Firstenberg
CC: VCSESU::Scott
Subj: New versions of STRDABRA and STRINI for Medusa builds

([redacted]lie, Mark,

I have modified the STRDABRA and STRINI as Charlie requested. The new versions are in AQUA::CARIBBEAN:[DV.TOOLS]. (Program STRMEM was also modified, but I don't know of any users.)

The new versions are fully backward compatible; all existing procedures that run these programs will continue to work exactly as before.

The change involves the /SCAN_FILES qualifier. This qualifier is used to specify the location of the .IDXSCAN files for each MCA type that must be made available to the program.

The previous program versions required that all the .IDXSCAN files be placed in a single directory and the files had to have a filename matching the name of the MCA. This did not allow referencing the files where they reside in CCM.

The new method expands these capabilities by replacing any "*" character in the file-spec provided with the /SCAN_FILES qualifier with the name of the MCA. This allows the building of file names. These file names point to specific files, specific directories, or to logical names.

Following is an extraction from HELP on STRDABRA:

[redacted]N_FILES=file-spec

Specifies a default file specification to be applied to all RTL to STR translation data files. These are indexed files containing information about scan-latch and supermacro bodies. These files are created using the SCANDATA procedure.

This file-spec is really used to specify many different input files. A uniquely named file must be read for each MCA type referenced in the .INI file. Specify a wildcard character "*" in the file-spec where the name of the MCA appears. STRDABRA will replace each "*" character with the name of a specific MCA, e.g. ADRX. By default, /SCAN_FILES=*.IDXSCAN is assumed.

An error message is reported if no "*" character is included in file-spec and the file name is specified. If no file name is specified, it is assumed to be "*". If no file type is specified, it is assumed to be .IDXSCAN.

Examples:

/SCAN_FILES=SCANFILES:*.IDXSCAN
specifies that all translation data files are in a directory pointed to by a logical name SCANFILES, and with a name the same as the MCA; i.e. SCANFILES:ADRX.IDXSCAN

/SCAN_FILES=MCA$*:MCA_*.IDXSCAN
[redacted] specifies that each translation data file is in a specific directory pointed to by a logical name that begins with MCA$ and ends with the name of the MCA and has a name beginning with MCA_; i.e. MCA$ADRX:MCA_ADRX.IDXSCAN

/SCAN_FILES=PTR$*IDX
specifies that each translation data file is specified by a name

that begins with PTR$ and ends with the MCA and IDX; i.e. PTR$ADRXIDX. Since there is no punctuation in this name, a logical named PTR$ADRXIDX may provide the full or partial specification. (e.g. PTR$ADRXIDX may be defined as IDX$DISK:MCA_ADRX.IDXSCAN). If no logical name exists, then the file PTR$ADRXIDX.IDXSCAN in the default directory will be opened.

MEDUSA

VAX Based, Gate Level Simulation Beyond Equivalent DECSIM Behavioral Model and ZYCAD Throughput

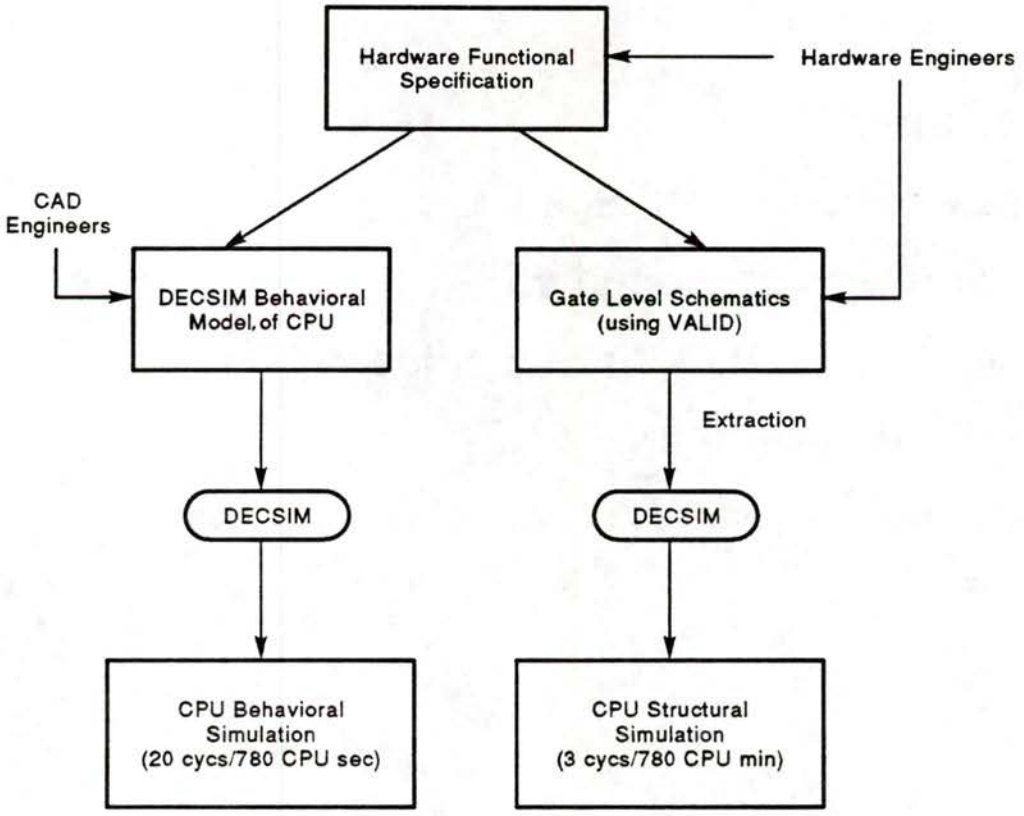
Mark Firstenberg
Aquarius Logic Design
HPS

Presented 1/26/89

Talk Outline

1. Background
2. Network Compiler Specifics
3. User Interface Specifics
4. User Supplied Code Specifics
5. Aquarius Results
6. Conclusions

Nautilus Design/Simulation Strategy



Behavioral Model:

- Fast, but not totally accurate
- Used mainly for micro-code debug
- Over 5 million AXE cases run

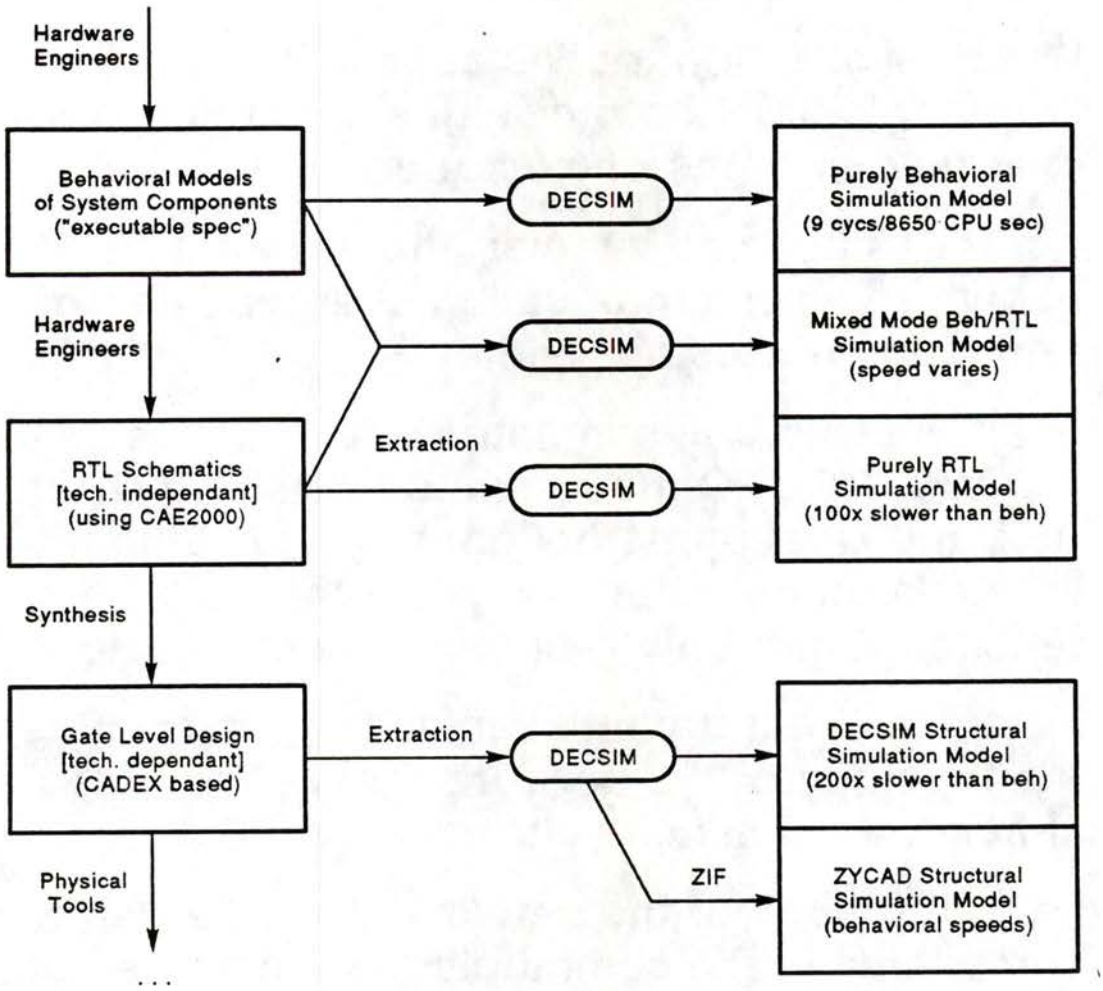
Structural Model:

- Accurate, but painfully slow
- Many approaches used to find bugs
- < 1000 AXE cases run (still best bug finder!)

So Why Is Structural Model So Slow (circa 1984)?

- When few network nodes appear in a model, the node's function outweighs communication overhead in terms of execution time
- As nodes become simpler and more numerous, *event driven* communication overhead begins to dominate execution time
- In a full structural simulation, an estimated 100 VAX MACRO instructions (including CALLx, RET and Queue instructions) execute whenever a node's output changes, while the node's function might only require a single instruction
- In effect, event driven simulation re-determines the topology of the network during each simulated machine cycle
- If the topology of the network could be determined at compile time, the communication overhead could be eliminated and much execution time saved
- Such a *compiled logic* simulator, called PRESTO, was written in Littleton in 1984 and proved to be 60 times faster than the equivalent DECSIM structural model for Nautilus

Aquarius Design/Simulation Strategy



Notes:

- **Run AXE/MAX on behavioral, mixed mode and RTL models (to verify logic design)**
- **Run MACRO diagnostics on ZYCAD (to ensure that synthesis has not introduced functional bugs)**

Concerns Regarding Aquarius Simulation Strategy (1986):

- AXE/MAX could not be used with ZYCAD, so that our best bug finders could not be applied to the most accurate simulation model
- Synthesized design may not be adequately tested by MACRO diagnostics running on ZYCAD
- Unclear that a system model would fit in ZYCAD
- Unknown how fast RTL simulation would run
- ZYCAD is a singular resource, while we have access to a lot of VAXes . . .

MEDUSA Goals:

- Encourage gate level simulation by:
  - Providing an alternative simulator with sufficient performance (throughput, if not speed)
  - Enabling AXE/MAX to be used on our most accurate simulation model
  - Allowing exploitation of available VAX resources
- Complement (as opposed to replace) DECSIM/ZYCAD simulation model use
- Be general enough to be used outside of Aquarius project

MEDUSA Objectives and Non-goals

Objectives (prioritized):

1. Performance (throughput, if not speed)
2. Support Aquarius effort
3. Generality
4. Reasonable model compilation time (overnight at very worst)

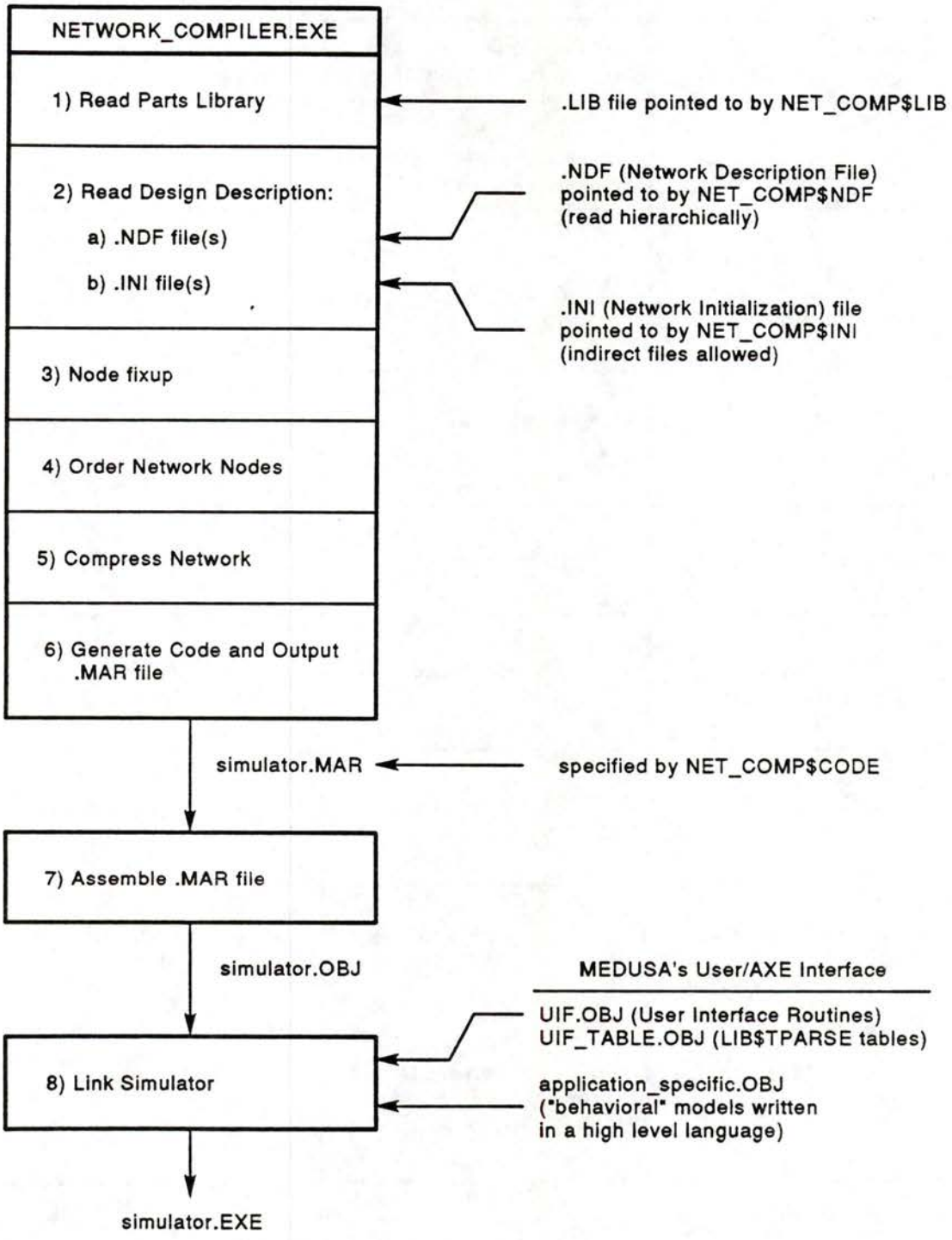
Non-goals:

- DECSIM compatibility
- Timing verification
- "On the fly" patching capability
- Fault simulation capability
- RTL support

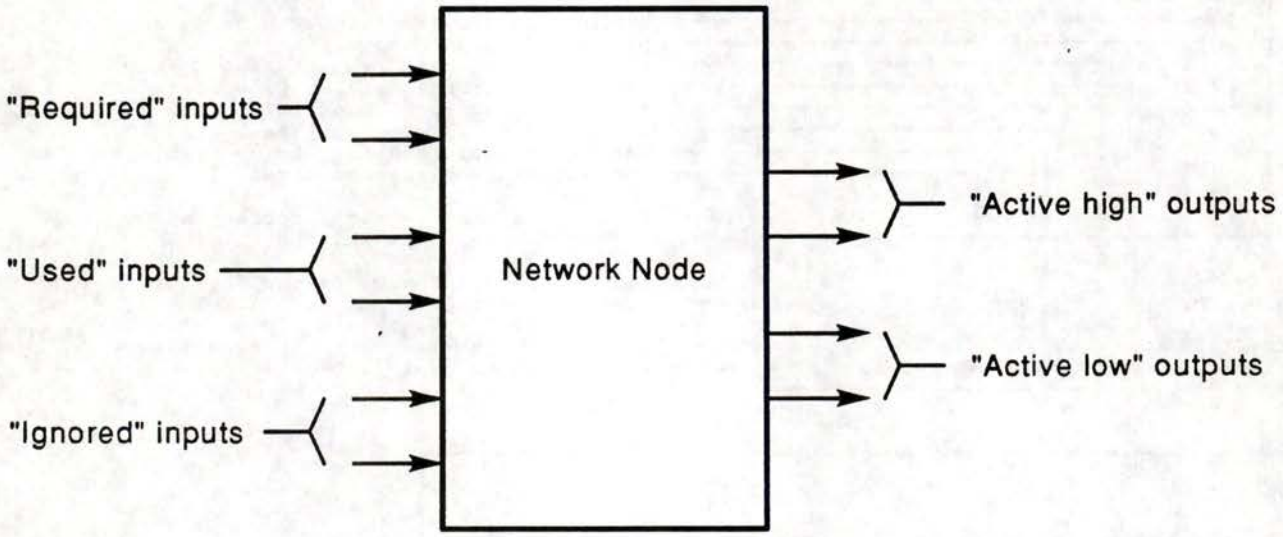
MEDUSA/DECSIM Comparison

Item to Compare	MEDUSA (compiled logic)	DECSIM (event driven)
Simulation States	0 and 1	0, 1, U and Z
Number of times an individual node can be activated per cycle	Once (and only once!)	0 to many
Simulated cycle execution time	Constant	Depends on number of events . . .
Pulses within a Cycle	Not possible	Possible/probable
Simulation of asynchronous logic	Difficult at best	No problem!
Simulation time granularity	Simulated machine cycle (possibly sub–cycle)	As fine grained as you like!
Behavioral modeling possible	Yes (but in MACRO or a high level language)	Yes (in DECSIM's behavioral language)
Mixed mode debugging integration	Split between MEDUSA User Interface and VAX DEBUGGER	Well integrated

MEDUSA Network Compilation Overview



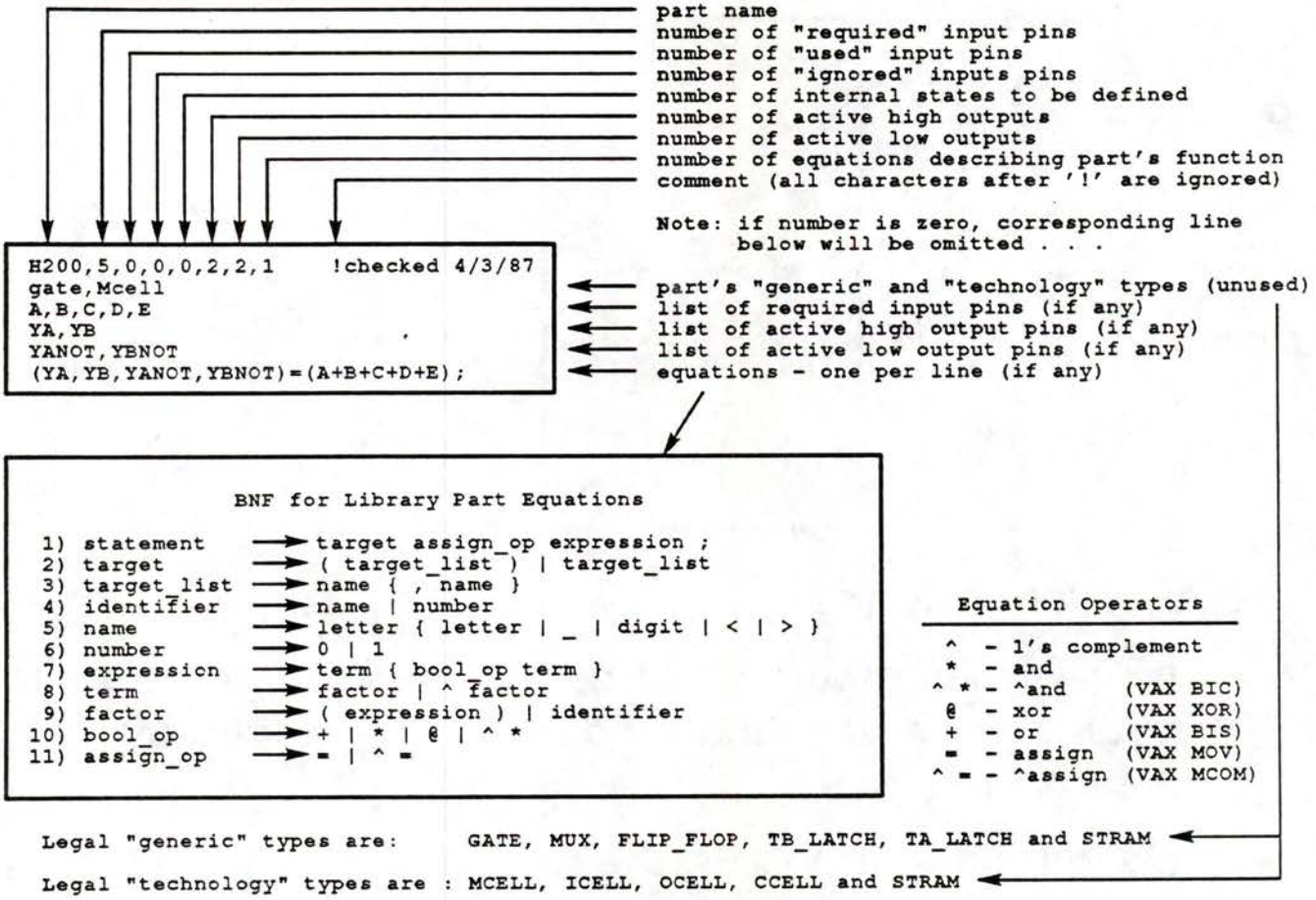
Simple Node Ordering



Node ordering terms:

- All *required* inputs must be available before the node can *fire* (i.e. it's output pin values can be determined)
- *Used* inputs appear in the node's functional equations but are not required for the node to fire
- *Ignored* inputs can be discarded immediately as network is being read . . .
- Once a node fires, all of it's outputs can be marked as *available*, potentially causing other nodes to fire
- Note that the node's function does not need to be known for the node to be ordered . . .

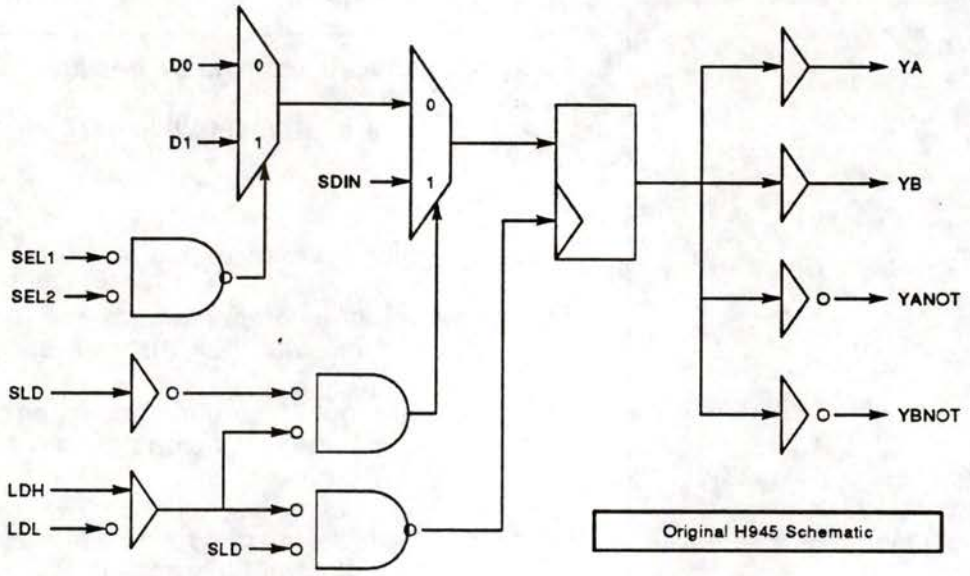
MEDUSA Parts Library (.LIB) File Syntax



Notes:

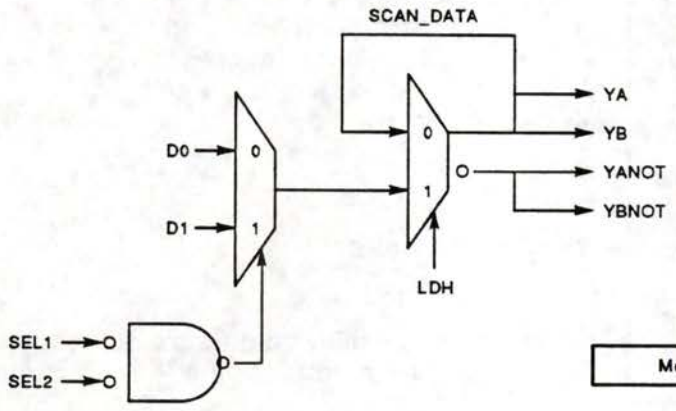
- Pin names must be scalar (vectors not allowed)
- No timing information provided . . .
- No line continuation character supported
- Equations must be on a single line

Example Library Part: H945 (scan latch)



Original H945 Schematic

If SDIN, SLD and LDL are ignored, and latches are represented as 2x1 muxes:



Modified H945 Schematic

.LIB file entry for H945

```
H945,1,4,3,2,2,2,2                    ! changed 12/8/87
Ta_latch,Mcell                        ! Generic and technology types
LDH                                   ! 'Required' inputs
D0,D1,SEL1,SEL2                       ! 'Used' inputs
SDIN,SLD,LDL                          ! 'Ignored' inputs
SCAN_DATA,SCAN_FLAG                   ! Internal states
YA,YB                                 ! Active high outputs
YANOT,YBNOT                           ! Active low outputs
  (SCAN_DATA,YA,YB,YANOT,YBNOT)=
        ((((D0*^(SEL1+SEL2)) +(D1*(SEL1+SEL2)))*
        (LDH*^SCAN_FLAG))+(SCAN_DATA*^(LDH*^SCAN_FLAG)));
  (SCAN_FLAG)=0;
```

SCAN_FLAG added to emulate scan function lost by ignoring SDIN and SLD

Note: in reality, equations must be on a single line!!

MEDUSA Network Description File (.NDF) Syntax

```
/SIGNALS/

$SIG signal_name
        .
        .
        .
$IO signal_name
        .
        .
        .

/NODE-DATA/

$REF logical_reference_designator
$PART library_part_name
$PIN pin_name/signal_name
        .
        .
        .
$REF logical_reference_designator
$PART library_part_name
$PIN pin_name/signal_name
        .
        .
        .
/END-NODE-DATA/

[$SYN signal_name=signal_name]
        .
        .
        .
```

Notes on Network Description File (.NDF) Syntax

All signal_names used must be declared either SIG or IO:
1) SIGnals are internal to this network

2) IO signals are pins to this network

Any unused, non-IO output signals can be removed, along with their PIN references

Any unsourced, non-IO input signals can be removed, with their PIN signal_name set to 0

All 'signal_name's and 'pin_name's must be:
1) scalar (vectors not allowed)
2) enclosed in double quotes

'Logical_reference_designator's and 'library_part_name's should NOT be enclosed in double quotes . . .

SYNonyms are optional . . .

Comments are allowed and are delimited by '!' (all characters after the '!' are ignored)

Hierarchical .NDF Files

- Each hierarchical design element must have a:
  - File name of design_element.NDF
  - Logical name NDF$design_element pointing to it
- Design hierarchical extensions similar to DECSIM:

  Nested hierarchical design element's logical reference designator are concatenated with '.'s to form fully extended signal and STRAM names
- There is no modular compilation (since all network nodes must be available during node ordering), so individual .NDF files may be read many times
- If any design element is not in the parts library and does not have a .NDF file associated with it:

  The design element's pins are treated as I/O pins of the overall model (to ensure that behavioral models will have access to them!)
- Up to 8 hierarchical levels are currently supported

MEDUSA Network Initialization (.INI) File Syntax

```
@file_name ["scope_name"]
        .
        .
        .
        .
/TIMEx-INITS/
"signal_name" [= value]
        .
        .
        .
/TIMEx-INITS/
"signal_name" [= value]
        .
        .
        .
/END-INITS/
```

Notes on Network Initialization (.INI) File Syntax

Indirect .INI files can be specified . . .
Scope_name will precede signal_names in new file:
1) scope_name must be enclosed in double quotes
2) if not specified, current scope used

'X' can range from –1 to 255 (i.e. byte integer)

Signal_names must be:
1) scalar (vectors not allowed)
2) enclosed in double quotes

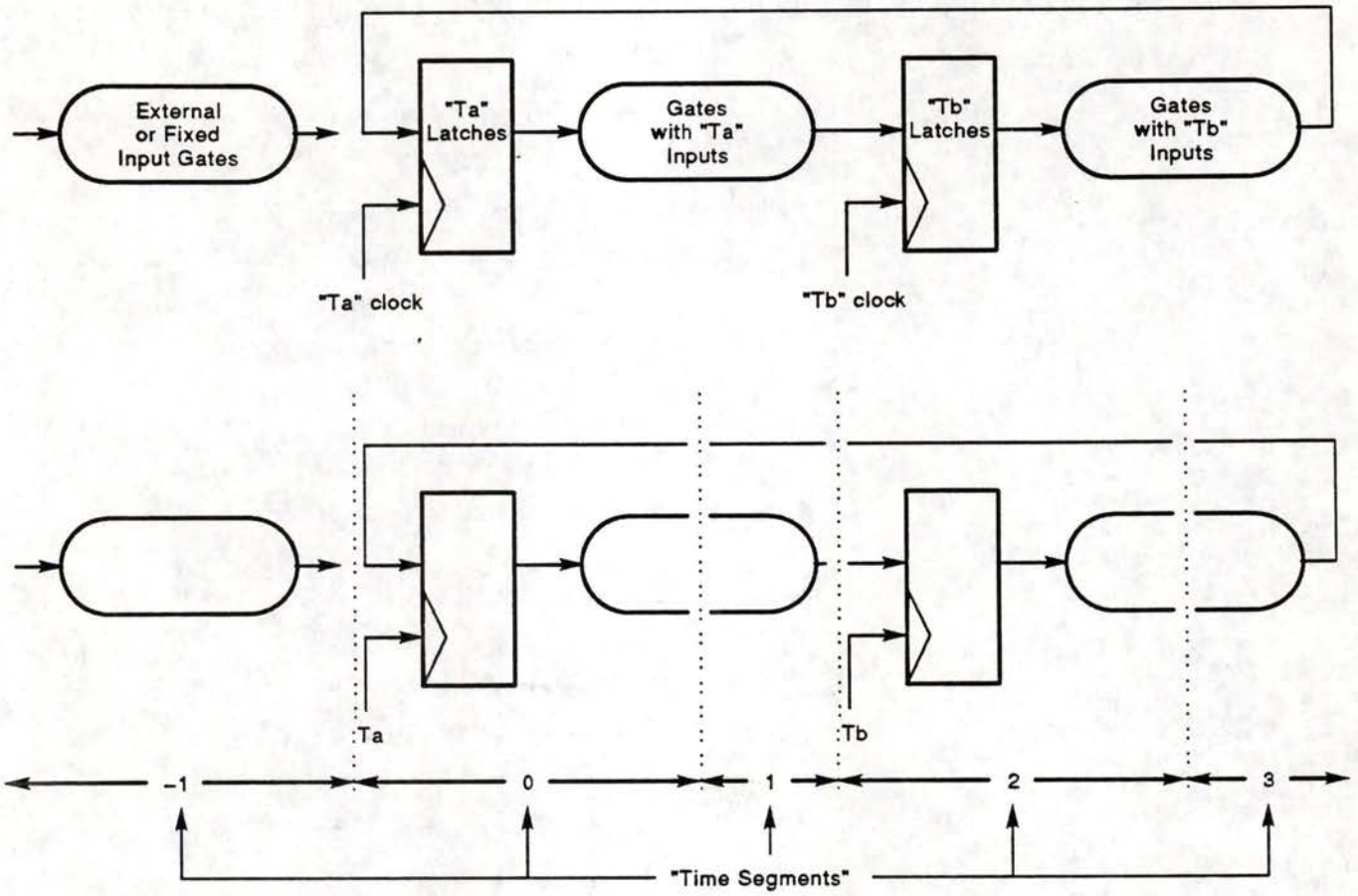
Values are:
1) optional
2) can only be 0 or 1
3) thought of as compile–time constants

Comments are allowed and are delimited by '!'
(all characters after the '!' are ignored)

Notes:

- **.INI files used to supply additional information which may not be apparent from the .NDF files**
- **The *time segment* constraints specified for a signal are actually applied to the network nodes they drive**
- **The largest value for 'x' among all .INI files read will determine the number of node ordering passes required to order entire network**
- **8 levels of .INI file nesting currently supported**

Latch Based Design Example



Time Segments **are used to establish:**

- **Delays between clocks in a multi-clock system**
- **Synchronization points with behavioral models (similar to DECSIM "wait" statements)**

Use of "used" (versus "required") pins within library parts can effectively break loop-back path associated with synchronous designs (see previous H945 example!), so that node ordering has a starting point

Generic Aquarius MCA .INI File

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
!      MCA initialization . . .
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
/TIME-1-INITS/                    !*** Segment -1 initializations ***
"RING_OSC_ENABLE_H" = 0           !
"SCAN_DATA_IN_H" = 0              ! Disable 'ring osc.' circuit
"E0440_YD_H" = 1                  !
                                  ! - - - - - - - - - - - - - - - - - - - - - - -
"T1_H" = 0                        !
"T2_H" = 0                        !
"T4_H" = 0                        ! Disable on-MCA clock generation circuit
"T5_H" = 0                        !
"T6_H" = 0                        !

/TIME0-INITS/                     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
"SYS_TA_CLK_1_H" = 1              !*** Segment 0 initializations ***
"SYS_TA_CLK_1_L" = 0              !
"SYS_TA_CLK_2_H" = 1              ! Supply Ta clocks at 'time 0'
"SYS_TA_CLK_2_L" = 0              !
"T0_H" = 0                        !
"T0A_L" = 1                       ! Supply 't0' Ta clocks at 'time 0'
                                  !
/TIME2-INITS/                     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
"SYS_TB_CLK_1_H" = 1              !*** Segment 2 initializations ***
"SYS_TB_CLK_1_L" = 0              !
"SYS_TB_CLK_2_H" = 1              ! Supply Tb clocks at 'time 2'
"SYS_TB_CLK_2_L" = 0              !
"T3_H" = 0                        !
                                  ! Supply 't3' Tb clock at 'time 2'

/END-INITS/                       ! End of MCA initialization
```

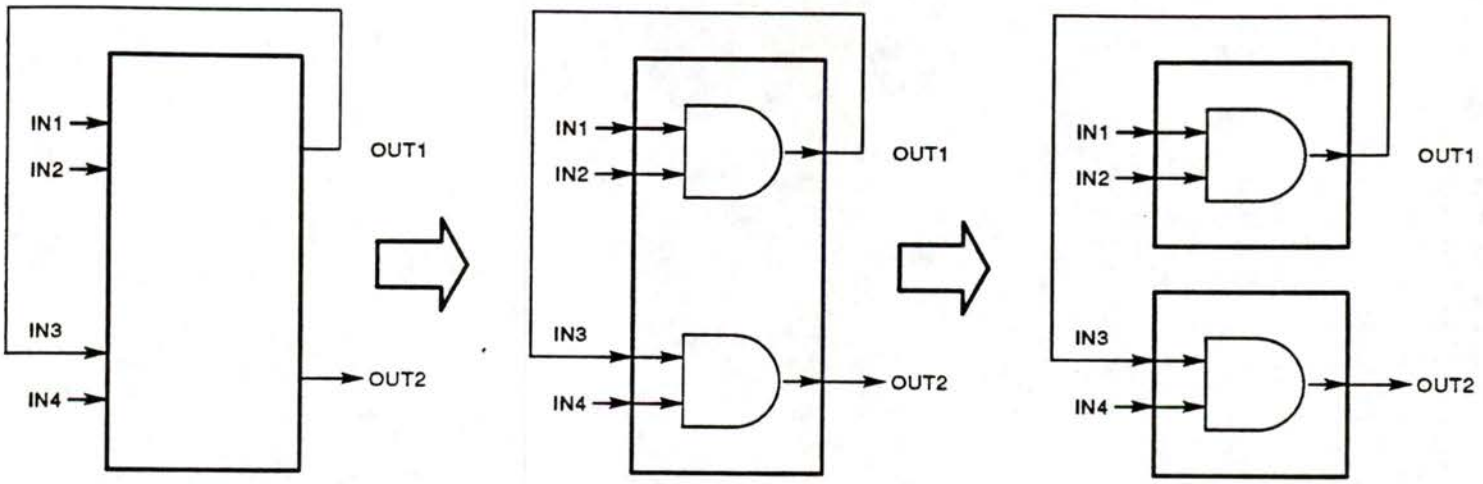
Notes:

- If 'ring oscillator' circuit was not disabled, it would be called out as on illegal loop during network node ordering . . .
- Time segments 1 and 3 can be used to place RAMs relative to TA and TB latches . . .

Example Aquarius MCU .INI File

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
!       DST MCU Initialization
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
@NDF$BUILD:MCA  "DST0"          !
@NDF$BUILD:MCA  "DST1"          !
@NDF$BUILD:MCA  "DST2"          ! Initialize DST MCAs
@NDF$BUILD:MCA  "DST3"          !
@NDF$BUILD:MCA  "SRCS"          !
@NDF$BUILD:STGX "STG0"          ! Initialize DST STGXs
@NDF$BUILD:STGX "STG1"          !
                                !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
/TIME1-INITS/                   !*** Segment 1 initializations ***
"STRAM2_CLK2_H<0>" = 1          !
"STRAM2_CLK2_H<1>" = 1          ! Treat T2 clocks as 'pre-Tb' clocks!?
"STRAM2_CLK2_L<0>" = 0          !
"STRAM2_CLK2_L<1>" = 0          !
                                !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
/TIME2-INITS/                   !*** Segment 2 initializations ***
"STRAM3_CLK1_H<0>" = 1          !
"STRAM3_CLK1_H<1>" = 1          !
"STRAM3_CLK1_H<2>" = 1          !
"STRAM3_CLK1_H<3>" = 1          !
"STRAM3_CLK1_H<4>" = 1          !
"STRAM3_CLK1_L<0>" = 0          !
"STRAM3_CLK1_L<1>" = 0          ! Treat T3 clocks as Tb clocks!?
"STRAM3_CLK1_L<2>" = 0          !
"STRAM3_CLK1_L<3>" = 0          !
"STRAM3_CLK1_L<4>" = 0          !
"STRAM3_CLK3_H<0>" = 1          !
"STRAM3_CLK3_H<1>" = 1          !
"STRAM3_CLK3_L<0>" = 0          !
"STRAM3_CLK3_L<1>" = 0          !

/END-INITS/                     ! End of DST MCU initialization
```

Node Fixup Compilation Phase



If IN1, IN2, IN3 and IN4 are all required, node will never fire . . .

But if node contains independant elements, the apparent loop is legal.

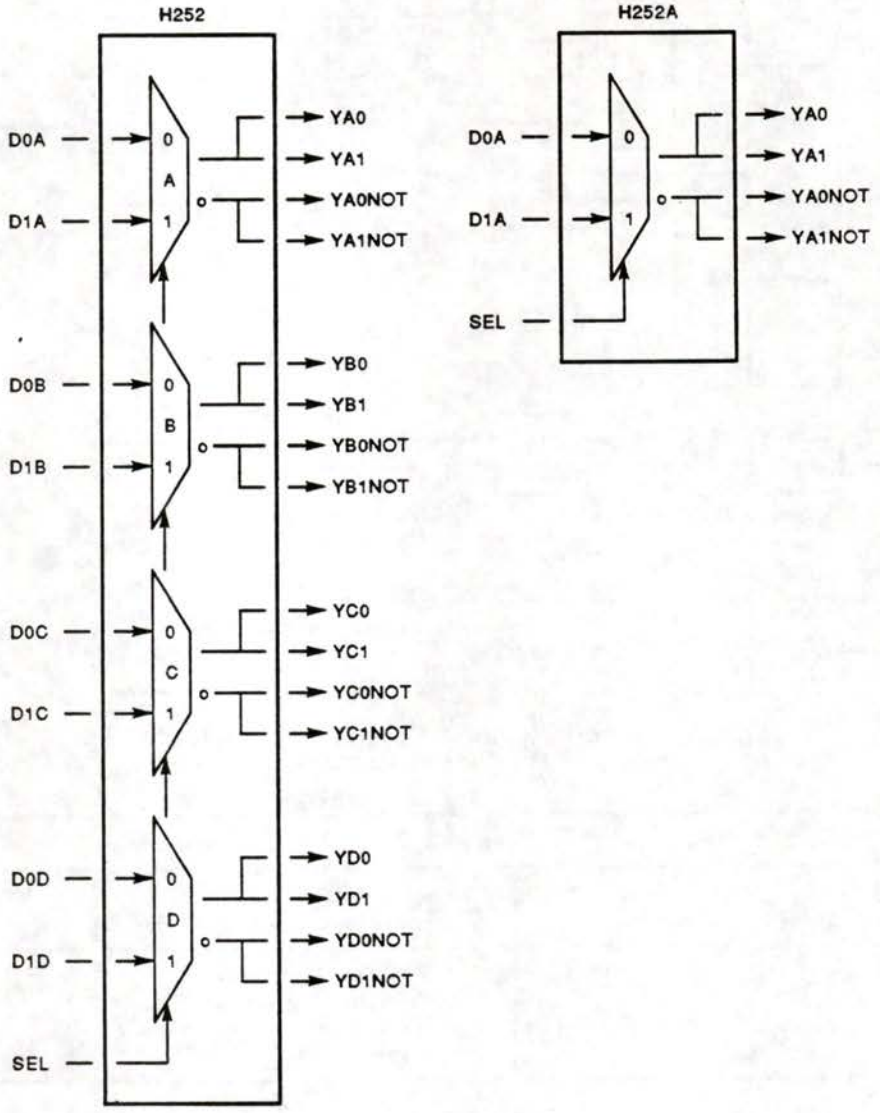
Splitting node in two solves problem.

Node fixup functions performed:

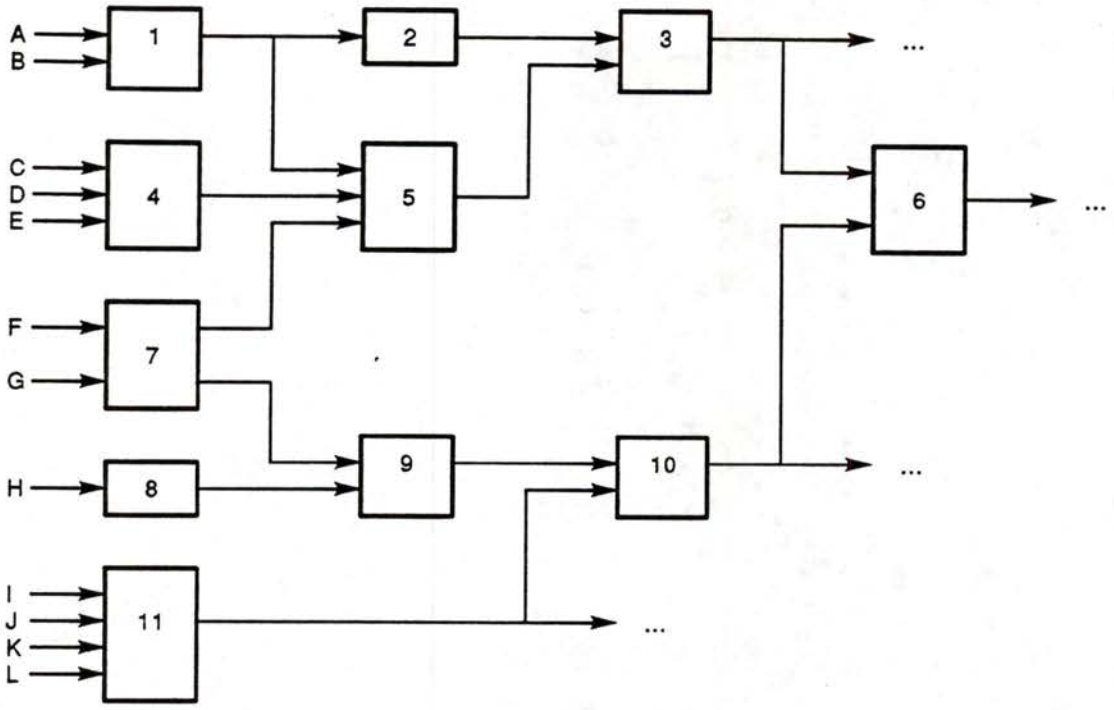
- Split segmented nodes (Aquarius specific):
  - 'L8xx's into 'L8xxA's and 'L8xxB's
  - 'xx47's into 'xx47A's and 'xx47B's
- Split TA/TB latch pair nodes (Aquarius specific):
  - 'C727D's, 'C727D2A's and 'C727M's into 'C727NP_TA's and 'C727P_TB's
  - 'C727MPx's into 'C727P_TA's and 'C727P_TB's
  - 'C727DA's into 'C727NP_TA's and 'C727NP_TB's
  - 'C728DP's into 'C728DP_TA's and 'C727P_TB's
- Insert 2 input OR gate at each wire-tie junction (only supported wire-tie function currently)

Example .NDF File for Segmented Part

```
/SIGNALS/

$IO "D0A"
$IO "D1A"
$IO "D0B"
$IO "D1B"
$IO "D0C"
$IO "D1C"
$IO "D0D"
$IO "D1D"
$IO "SEL"
$IO "YA0"
$IO "YA1"
$IO "YA0NOT"
$IO "YA1NOT"
$IO "YB0"
$IO "YB1"
$IO "YB0NOT"
$IO "YB1NOT"
$IO "YC0"
$IO "YC1"
$IO "YC0NOT"
$IO "YC1NOT"
$IO "YD0"
$IO "YD1"
$IO "YD0NOT"
$IO "YD1NOT"

/NODE-DATA/

$REF MUXA
$PART H252A
$PIN "D0A"/"D0A"
$PIN "D1A"/"D1A"
$PIN "SEL"/"SEL"
$PIN "YA0"/"YA0"
$PIN "YA1"/"YA1"
$PIN "YA0NOT"/"YA0NOT"
$PIN "YA1NOT"/"YA1NOT"

$REF MUXB
$PART H252A
$PIN "D0A"/"D0B"
$PIN "D1A"/"D1B"
$PIN "SEL"/"SEL"
$PIN "YA0"/"YB0"
$PIN "YA1"/"YB1"
$PIN "YA0NOT"/"YB0NOT"
$PIN "YA1NOT"/"YB1NOT"

$REF MUXC
$PART H252A
$PIN "D0A"/"D0C"
$PIN "D1A"/"D1C"
$PIN "SEL"/"SEL"
$PIN "YA1"/"YC0"
$PIN "YA1"/"YC1"
$PIN "YA0NOT"/"YC0NOT"
$PIN "YA1NOT"/"YC1NOT"

$REF MUXD
$PART H252A
$PIN "D0A"/"D0D"
$PIN "D1A"/"D1D"
$PIN "SEL"/"SEL"
$PIN "YA0"/"YD0"
$PIN "YA1"/"YD1"
$PIN "YA0NOT"/"YD0NOT"
$PIN "YA1NOT"/"YD1NOT"

/END-NODE-DATA/
```



Possible Node Ordering Methods



Assuming that A, B, C, D, E, F, G, H, I, J, K and L are all initially available and node numbers indicate appearance in 'unordered' linked list:

Examples of Possible Search Methods		
Strict Linear Search	Linear Search with gate level sensitivity	Recursive Search
Pass 1: 1,2,4,7,8,9,11 Pass 2: 5,10 Pass 3: 3,6	Pass 1: 1,4,7,8,11 Pass 2: 2,5,9 Pass 3: 3,10 Pass 4: 6	Pass 1: 1,2,4,7,5,3, 8,9,11,10,6
Simplest method	Allows for 'on the fly' model patching!	Moves sources and loads closer together (higher performance!)

MEDUSA uses the recursive search method

- **Faster compilation time (fewer passes)**
- **Faster simulation execution time!**

MEDUSA Node Ordering Specifics

First, for each unsourced input pin:

- If non-model I/O pin, assign it a value of zero
- If model I/O pin, assume that a behavioral model will drive it (i.e. do not assign specific value to it)
- In either case, mark pin as available . . .

Node ordering process:

- For time segments -1 to maximum specified:
  - For each node in 'unordered' linked list:
    If node's count of outstanding inputs is zero and timing constraint <= current time segment:
    Fire node
  - Place 'segment end' marker node at the end of 'ordered' linked list
- If 'unordered' linked list not empty, issue warning message and display contents of unfired nodes . . .

Node firing specifics:

- Remove node from 'unordered' linked list
- Place node at end of 'ordered' linked list
- Broadcast fact that each output is available:
  - Decrement count of outstanding inputs for any node sourced by any of the firing node's outputs
  - If count decremented to zero and time segment constraints met, recursively fire that node

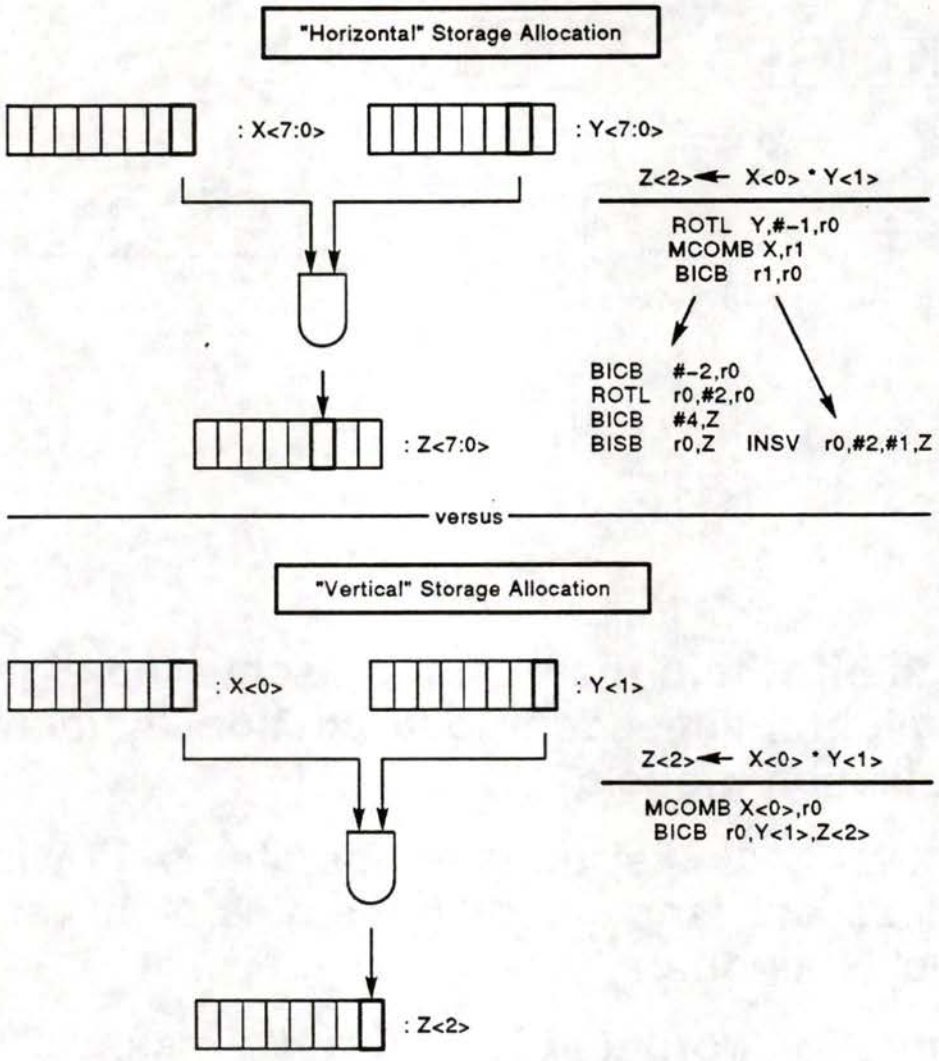
Network Compression Compilation Phase

Searching backwards through 'ordered' linked list, for each node in network:

- **If none of the node's output pins source anything and none of them are model I/O pins:**
  - **Delete node's output pins**
  - **Delete node's input pins (which may cause other nodes 'upstream' to have unused outputs!)**
  - **Delete node from 'ordered' linked list**
- **Otherwise, leave node as is**

'Ordered' linked list now ready for code generation . . .

Storage Allocation Alternatives

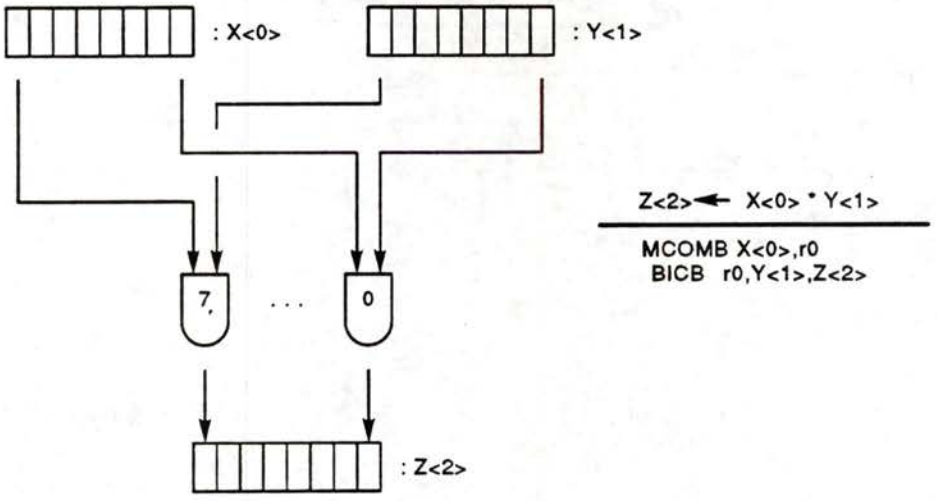


Benefits of "vertical" storage allocation:

- Easier to generate code
- Code produced executes faster

But what about the "unused" bits? . . .

Multiple, Simultaneous Simulations



Notes:

- Each bit within the byte represents the signal's state for eight independent, but functionally identical, simulation models
- VAX MACRO instructions operate as bit-wise parallel processors (same code generates eight results as quickly as one!)
- Works for word and longword data as well
- Nomenclature:

  Byte data: *8 machine* model (0 to 7)
  Word data: *16 machine* model (0 to 15)
  Longword data: *32 machine* model (0 to 31)

- MEDUSA network compiler allows choice of data size (i.e. amount of parallelism possible)

So What's in a Name?

Wanted a name that reflected simulator's multi-faceted nature:

MEDUSA: in Greek mythology, most famous of the three GORGON sisters. Once a beautiful woman, she offended Athena, who changed her hair into snakes and made her face so hideous that all who looked at her were turned to stone. Because of this power, her image frequently appeared on Greek armor.

(CERBERUS and HYDRA were also considered, but were names of previously canceled DEC projects . . .)

MEDUSA speed versus throughput:

- ***Speed*** **refers to performance of a single machine**
- ***Throughput*** **refers to aggregate performance of multiple machines**

Code Generation Notes

VAX GPR usage in generated code:

- R0 - reserved for MCOMx target during code generation for AND operator
- R1 through R8 - temporary values (within one equation)
- R9 through R11 - fixed base registers (covers 192K bytes of signal data with byte/word displacement specifiers)
- R12 (AP) - floating base register (autoincrements through signal data)

Check number of equations for each node in 'ordered' linked list:

- If non-zero, generate code for them one at a time
- If zero, but node is a STRAM, output appropriate STRAM code model (Aquarius specific)
- Otherwise, output warning message and ignore node

Code Generation Notes (cont.)

Optimizations performed:

- Assign functionally equivalent signals to the same storage (i.e. avoid generating MOVx instructions)
- Keep track of 1's complemented versions of signals (i.e. avoid generating multiple MCOMx instructions for the same signal)
- Perform optimizations on a per-equation basis
- Bind signals to signal storage locations as late as possible (results in sequential D-stream writes during simulator execution)
- Generate smallest operand specifiers possible (in terms of I-stream bytes)

Optimizations not performed:

- Storing frequently used signals in registers
- Common sub-expression recognition
- Optimizations between equations
- Optimizations between nodes

Equation Optimizations Performed

- ***NOT*** **Operator Optimizations:**
  1. (^0) —> 1
  2. (^1) —> 0
  3. (^^X) —> X
- ***OR*** **Operator Optimizations:**
  1. (1+X) or (X+1) —> 1
  2. (0+X) or (X+0) —> X
  3. (X+X) —> X [where X not an operator node]
  4. (^X+Y) —> (Y+^X) —> ^(X^*Y)
  5. (^X+^Y) —> ^(^X^*Y)
- ***AND*** **Operator Optimizations:**
  1. (0*X) or (X*0) —> 0
  2. (1*X) or (X*1) —> X
  3. (X*X) —> X [where X not an operator node]
  4. (X*^Y) —> (^Y*X) —> (Y^*X)
  5. (^X*^Y) —> ^(X+Y)

Equation Optimizations Performed (cont.)

- *"NOT AND"* Operator Optimizations:
  1. (1^*X) or (X^*0) —> 0
  2. (X^*X) —> 0 [where X not an operator node]
  3. (0^*X) —> X
  4. (X^*1) —> ^X
  5. (X^*^Y) —> ^(X+Y)
  6. (^X^*^Y) —> ^(^X+Y) [—> (Y^*X)!!]
- *XOR* Operator Optimizations:
  1. (X@X) —> 0 [where X not an operator node]
  2. (0@X) or (X@0) —> X
  3. (1@X) or (X@1) —> ^X
  4. (X@^Y) —> (^Y@X) —> ^(X@Y)
  5. (^X@^Y) —> (X@Y)
- *ASSIGN* Operator Optimizations:
  1. =(^X) —> ^=X

Notes:

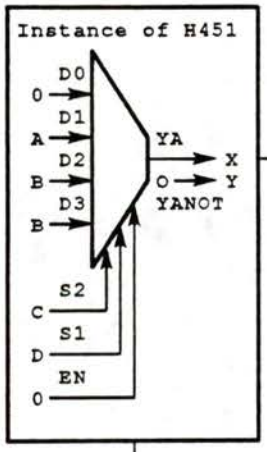
Only single level optimizations performed
Optimizations listed in priority order
Heuristic - push *NOTs* upwards
Remember there is no VAX "AND" instruction!

Code Generation Example

```
H451,7,0,0,0,2,2,1        !checked 4/3/87
mux,Mcell
D0,D1,D2,D3,S2,S1,EN
YA,YB
YANOT,YBNOT
(YA,YB,YANOT,YBNOT)=(((D0*^S2*^S1)+(D1*^S2*S1)+(D2*S2*^S1)+(D3*S2*S1))*^EN);
```

0 — When .LIB file originally read, equation was optimized and stored as follows:

```
(YA,YB,YANOT,YBNOT)=(EN^*(((S2+S1)^*D0)+((D1*(S2^*S1))+((D2*(S1^*S2))+(D3*(S2*S1))))));
```



1 — Build 'abstract syntax tree' of particular part's equation and populate with specifics of part's instantiation

```
(X,Y)=(0^*(((C+D)^*0)+((A*(C^*D))+((B*(D^*C))+(B*(C*D))))));
```

2 — Optimize populated AST

```
(X,Y)=((A*(C^*D))+((B*(D^*C))+(B*(C*D))));
```

3 — Allocate registers for AST's 'operator' nodes

```
(X,Y)=((A*(C^*D))+((B*(D^*C))+(B*(C*D))));
        r1 r1 r3 r2 r2 r3 r3 r3
```

4 — Generate code (then go back to step 1 with part's next equation, if it has any more . . .)

```
BICx3   C,D,r1      ; 'x' is 'B', 'W' or 'L', depending on data size selected
MCOMx   A,r0        ; '*' operator forces 1's complement of left operand
BICx2   r0,r1       ; 'BICx3 r0,r1,r1' compressed
BICx3   D,C,r2      ;
BICx2   ^B,r2       ; Assume 1's complemented B, '^B', previously generated
MCOMx   C,r0        ; '*' operator forces 1's complement of left operand
BICx3   r0,D,r3     ;
BICx2   ^B,r3       ; Assume 1's complemented B, '^B', previously generated
BISx2   r2,r3       ; 'BISx3 r2,r3,r3' compressed
BISx3   r1,r3,X     ; 'BISx2 r1,r3'/'MOVx r3,X' compressed
MCOMx   X,Y         ; Record that X and Y complemented versions of each other
```

Stylized STRAM Model Code

```
;
;       R0:     temp value . . .                        R1:     ram index
;       R2:     machine shift count                     R3:     machine mask
;       R4:     1's complimented machine mask           R5:     current data address
;       R6:     -1                                      R7:     loop count limit
;       R8:     loop count
;
        CLRL    r8                                      ; Initialize loop count
        MOVL    #-1,R6                                  ; Save I-stream immediate bytes!
        SUBL3   #1,MEDUSA$MAX_NUM_MACHINES,r7           ; Compute loop count limit
log_ref$LOOP_TOP:
        ROTL    r8,#1,r3                                ; Compute current machine's mask
        MCOML   r3,r4                                   ; Compute complimented mask
- [if cs used:  ------------------------------------------------------------------
        BICx3   r4,cs,r0                                ; Check current machine's CS
        BEQL    log_ref$ENABLED                         ; If clear, RAM is enabled!!
    [do n=0 to data_bits-1 by 1:                        ;
      if q<n> used:                                     ; If Q<N> output used:
        BICx2   r3,q<n>                                 ;   Clear machine's Q<N> bit
    end]                                                ;
        ACBL    r7,#1,r8,log_ref$LOOP_TOP               ; Disable finished - close loop
        BRW     log_ref                                 ; All machine's processed - exit
log_ref$ENABLED:                                        ;
- end] ------------------------------------------------------------------------
        MNEGL   r8,r2                                   ; Compute machine's shift count
        SUBL3   r8,#31,r0                               ; Shift current machine's
        ROTL    r0,addr<0>,r0                           ;   ADDR<0> into highest R0 bit
- [do n=1 to addr_bits-2 by 1: ------------------------------------------------
        ROTL    r2,addr<n>,r1                           ; Put current machine's ADDR<N>
                                                        ;   into lowest R1 bit
        ASHQ    r6,r0,r0                                ; Accumulate result in R0
- end] ------------------------------------------------------------------------
        ROTL    r2,addr<addr_bits-1>,r1                 ; Put current machine's
                                                        ;   ADDR<N-1> into lowest R1 bit
        BICL2   #-2,r1                                  ; Clear all but lowest R1 bit
        ASHQ    #<addr_bits-1>,r0,r0                    ; Shift ZEXTed RAM index into R1
        MOVAL   MEDUSA$RAM_DATA+                        ; Compute starting address of
                   ram_start*data_size_val,r5           ;   RAM's '0th bit/0th element'
        MULL2   #data_bits,r1                           ; Account for RAM's width!!
        BICx3   r4,wr,r0                                ; Check current machine's WR
        BEQL    log_ref$WRITE                           ; If clear, RAM write requested!
;log_ref$READ:                                          ; Read RAM element indexed:
- [do n=0 to data_bits-1 by 1: ------------------------------------------------
    if q<n> used:                                       ; If Q<N> output used:
        BICx3   r4,(r5)+[r1],r0                         ;   R0 gets isolated read data
        BICx2   r3,q<n>                                 ;   Clear machine's Q<N> bit
        BISx2   r0,q<n>                                 ;   Load isolated data into Q<N>
    else:                                               ; Else:
        ADDL2   #data_size_val,r5                       ;   Skip over RAM element's bit
- end] ------------------------------------------------------------------------
        ACBL    r7,#1,r8,log_ref$LOOP_TOP               ; Read finished - close loop
        BRB     log_ref                                 ; All machine's processed - exit
log_ref$WRITE:                                          ; Write RAM element indexed:
- [do n=0 to data_bits-1 by 1: ------------------------------------------------
        BICx3   r4,d<n>,r0                              ;   R0 gets isolated write data
        BICx2   r3,(r5)[r1]                             ;   Clear machine's RAMs nth bit
        BISx2   r0,(r5)+[r1]                            ;   Load write data into nth bit
    if q<n> used:                                       ;   If Q<N> output used:
        BICx2   r3,q<n>                                 ;     Clear machine's Q<N> bit
        BISx2   r0,q<n>                                 ;     Load write data into Q<N>
- end] ------------------------------------------------------------------------
        ACBL    r7,#1,r8,log_ref$LOOP_TOP               ; Write finished - close loop
log_ref:                                                ; All machine's processed - exit
```

MEDUSA Simulation Loop Code

```
        .PSECT  MEDUSA$CODE, NOWRT, EXE

        .ENTRY  MEDUSA$$SIM,^M<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>

        MOVAL   MEDUSA$SIG_DATA+32768,r9        ; Set up 'fixed base' registers
        ADDL3   #65536,r9,r10                   ;   R9, R10 and R11 . . .
        ADDL3   #65536,r10,r11                  ;

MEDUSA_LOOP_START:                              ; Set up 'moving base' reg. AP
        MOVAL   MEDUSA$SIG_DATA+(data_size_val*2),ap

          Time Segement -1 code

; End of simulation time segment -1
        CALLS   #0,MEDUSA$$USC                  ; Call 'user supplied code'
        INCL    MEDUSA$SEGMENT_NUM              ; Update time segment indicator

          Time Segement 0 code

; End of simulation time segment 0
        CALLS   #0,MEDUSA$$USC                  ; Call 'user supplied code'
        INCL    MEDUSA$SEGMENT_NUM              ; Update time segment indicator
                .
                .
                .
          Time Segement n code

; End of simulation time segment n
        CALLS   #0,MEDUSA$$USC                  ; Call 'user supplied code'
        MOVL    #-1,MEDUSA$SEGMENT_NUM          ; Init. time segment indicator

        MOVAL   MEDUSA$CYCLE_CNTS,r0            ; Bump individual machine
        INCL    (r0)+                           ;   cycle counters . . .
                .                               ;
                .                               ;   (for each possible machine)
                .

        SOBGTR  MEDUSA$BURST_CNT,MEDUSA_LOOP_END
        RET

MEDUSA_LOOP_END:
        BLBC    MEDUSA$RET_TO_UIF,MEDUSA_SKIP_UIF_RET
        RET
MEDUSA_SKIP_UIF_RET:
        JMP     MEDUSA_LOOP_START
```

Example NETWORK_COMPILER.EXE Output

```
$SET NOVERIFY
12-NOV-1988 17:11:57.80                    User: FIRSTENBERG
Accounting information:
 Buffered I/O count:        246  Peak working set size:        2018
 Direct I/O count:           83  Peak virtual size:            2850
 Page faults:             14673  Mounted volumes:                 0
 Images activated:            6
 Elapsed CPU time:      0 00:00:06.72 Connect time:          0 00:00:11.57

*****************************************************************************
*                              Process update                               *
*                              --------------                               *
*   Buffered I/O delta:       251        Peak working set delta:       2018   *
*   Direct I/O delta:          85        Peak virtual size delta:      2850   *
*   Page Fault delta:       14763                                           *
*   CPU seconds delta:       6.81        Connect seconds delta:        6.81   *
b
*****************************************************************************
%NCOMP-I-LIBREAD        Reading parts library
%RLIB-I-NUMPARTS        450 library parts loaded
*****************************************************************************
*                              Process update                               *
*                              --------------                               *
*   Buffered I/O delta:         2        Peak working set delta:          0   *
*   Direct I/O delta:          14        Peak virtual size delta:         0   *
*   Page Fault delta:        1380                                           *
*   CPU seconds delta:      12.53        Connect seconds delta:       13.46   *
*****************************************************************************
%NCOMP-I-NETREAD        Reading network description
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU]CPU.NDF;15 openned
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.EBOX.CTL]CTL.NDF;6 openned
%RNDF-W-NDFNOTFND       NDF$CDC:CDC not found
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.EBOX.CTL.ISSA]ISSA.NDF;4 openned
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.EBOX.CTL.ISSB]ISSB.NDF;5 openned
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.EBOX.CTL.ISSC]ISSC.NDF;3 openned
                        .
                        .   (3997 files)
                        .
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.IBOX.XBR.PCHI]PCHI.NDF;9 openned
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.IBOX.XBR.PCLO]PCLO.NDF;10 openned
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.IBOX.XBR.VICT]VICT.NDF;8 openned
%RNDF-W-NDFNOTFND       NDF$CDC:CDC not found
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.IBOX.XBR.XDTA]XDTA.NDF;9 openned
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.IBOX.XBR.XDTB]XDTB.NDF;4 openned
%RNDF-I-NDFOPENNED      NDF$ROOT:[CPU.IBOX.XBR.XSCA]XSCA.NDF;10 openned
%RNET-I-NETSTATS        Total number of internal signals read:   341341
                        Total number of I/O signals read:          2083
                        Total number of network nodes read:      132780
                        Total number of node pins read:          993209
                        Total number of node pins ignored:       266787
*****************************************************************************
*                              Process update                               *
*                              --------------                               *
*   Buffered I/O delta:     16291        Peak working set delta:      61982   *
*   Direct I/O delta:        9456        Peak virtual size delta:    109191   *
*   Page Fault delta:      154991                                           *
*   CPU seconds delta:    2167.92        Connect seconds delta:     2485.28   *
*****************************************************************************
```

Example NETWORK_COMPILER.EXE Output (cont.)

```
%NFIX-I-FIXUPSTATS      Total number of new signals added:          4665
                        Total number of network nodes added:       34826
                        Total number of new node pins added:        8674
                        Total number of node internal states:      73929
*****************************************************************************
*                                 Process update                            *
*                                 --------------                            *
*   Buffered I/O delta:       233         Peak working set delta:           0 *
*   Direct I/O delta:         116         Peak virtual size delta:      93720 *
*   Page Fault delta:      772010                                             *
*   CPU seconds delta:     726.12         Connect seconds delta:       838.09 *
*****************************************************************************
%NCOMP-I-ORDERNET       Ordering network nodes
%ONET-I-ALLIPINSCHK     Beginning "all input pins" network check
%ONET-I-SEGMENTSTART    Beginning "time segment -1" node ordering . . .
%ONET-I-SEGMENTSTART    Beginning "time segment 0" node ordering . . .
%ONET-I-SEGMENTSTART    Beginning "time segment 1" node ordering . . .
%ONET-I-SEGMENTSTART    Beginning "time segment 2" node ordering . . .
%ONET-I-SEGMENTSTART    Beginning "time segment 3" node ordering . . .
*****************************************************************************
*                                 Process update                            *
*                                 --------------                            *
*   Buffered I/O delta:         3         Peak working set delta:           0 *
*   Direct I/O delta:           9         Peak virtual size delta:          0 *
*   Page Fault delta:      403930                                             *
*   CPU seconds delta:      58.04         Connect seconds delta:       111.90 *
*****************************************************************************
%NCOMP-I-COMPRESSNET    Compressing network . . .
%CNET-I-NOOPINSCHK      Beginning "no output pins" node check
%CNET-I-NETSTATS        Total number of node pins removed:                58844
                        Total number of "unused outputs" nodes removed:    4700
*****************************************************************************
*                                 Process update                            *
*                                 --------------                            *
*   Buffered I/O delta:         1         Peak working set delta:           0 *
*   Direct I/O delta:           5         Peak virtual size delta:          0 *
*   Page Fault delta:       54224                                             *
*   CPU seconds delta:      34.00         Connect seconds delta:        53.85 *
*****************************************************************************
%NCOMP-I-OUTPUTCODE     Outputing code for ordered network nodes
%CGEN-I-CGENSTATS       Number of nodes going into code generation:      162576
                        Number of nodes "in-line" code generated for:    112740
                        Number of RAM nodes:                                330
                        Number of MACRO instructions generated:          633739
                        Estimated I-stream bytes generated:             3781730
                        Number of signal storage bytes allocated:        177788
                        Number of RAM storage bytes allocated:          2732032
*****************************************************************************
*                                 Process update                            *
*                                 --------------                            *
*   Buffered I/O delta:      2870         Peak working set delta:           0 *
*   Direct I/O delta:       11475         Peak virtual size delta:          1 *
*   Page Fault delta:      403051                                             *
*   CPU seconds delta:    1962.27         Connect seconds delta:     2650.45 *
*****************************************************************************
```

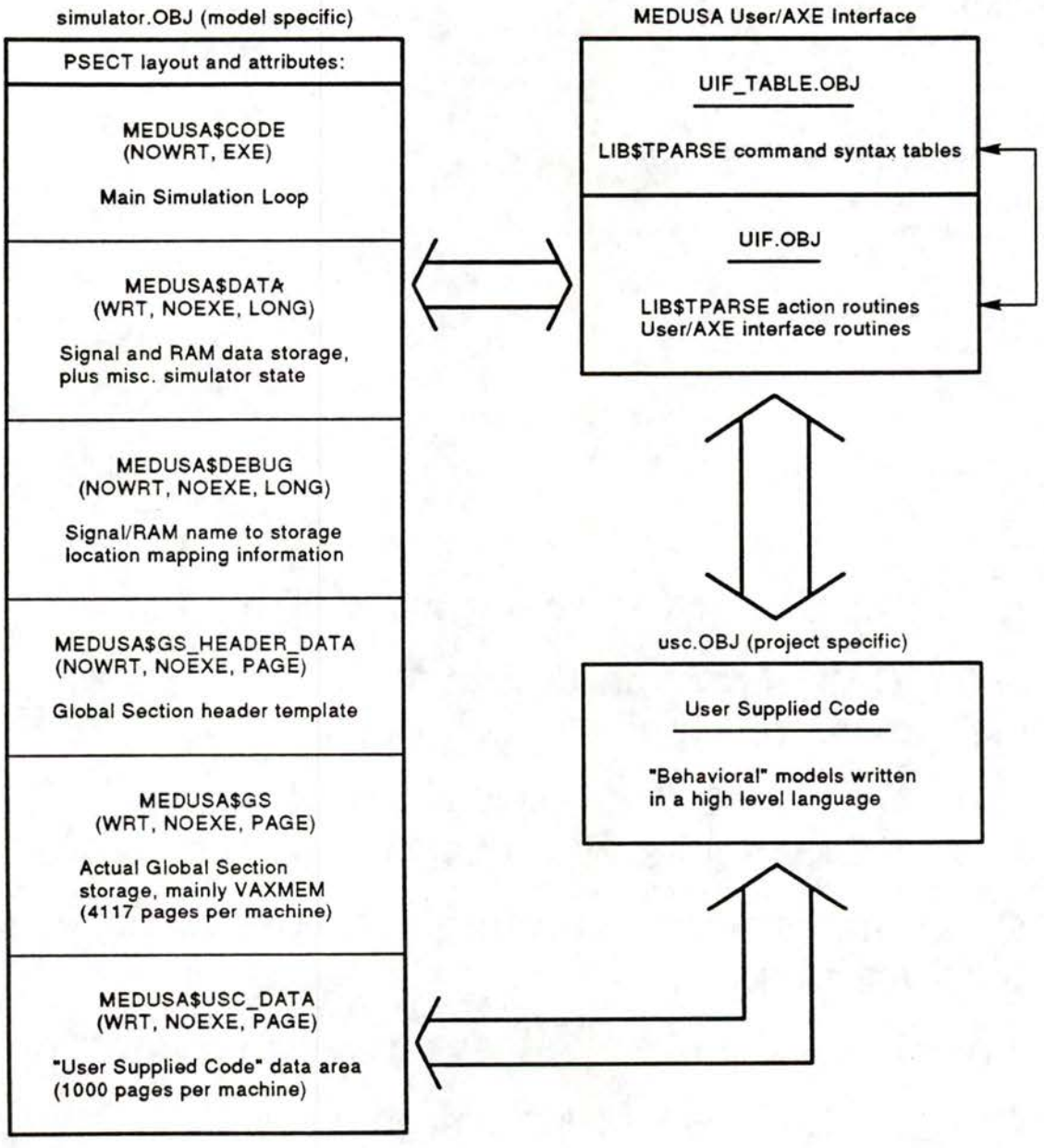
Example NETWORK_COMPILER.EXE Output (cont.)

```
12-NOV-1988 18:54:53.01                          User: FIRSTENBERG
Accounting information:
 Buffered I/O count:      19659  Peak working set size:       64000
 Direct I/O count:        21165  Peak virtual size:          205762
 Page faults:           1804542  Mounted volumes:                 0
 Images activated:            8
 Elapsed CPU time:      0 01:23:07.91
 Connect time:          0 01:43:06.84

Directory WORKSPACE:[WORKSPACE.MEDUSA]

CPU8.MAR;1                94496  (,RWED,R,)

Total of 1 file, 94496 blocks.
%DELETE-I-FILDEL, WORKSPACE:[WORKSPACE.MEDUSA]CPU8.MAR;1 deleted (94497 blocks)
  FIRSTENBERG  job terminated at 12-NOV-1988 19:34:14.87
  Accounting information:
  Buffered I/O count:          21339      Peak working set size: 64000
  Direct I/O count:            31019      Peak page file size:  205762
  Page faults:               2093187      Mounted volumes:           0
  Charged CPU time:      0 01:55:11.01    Elapsed time:     0 02:22:28.65
```

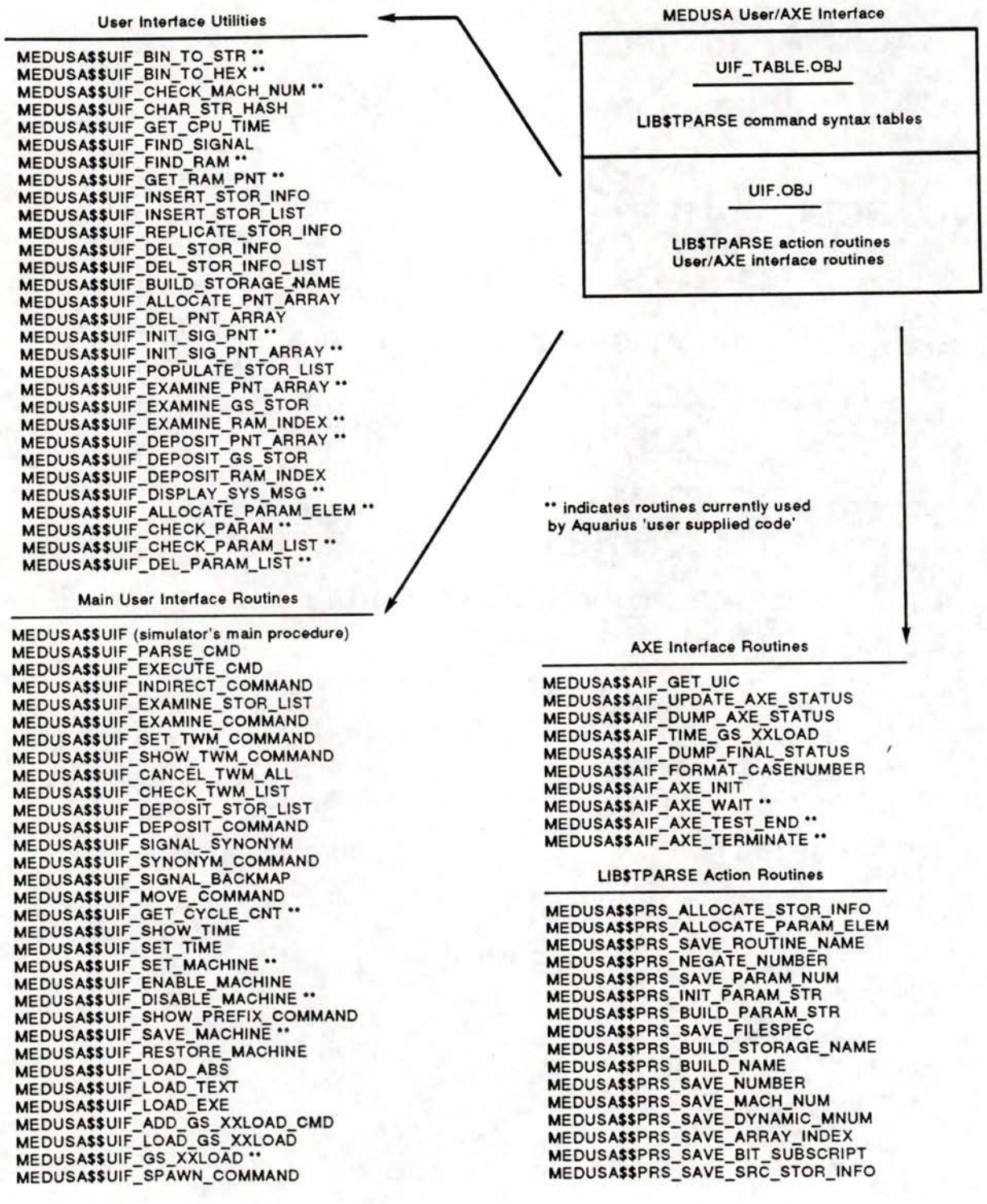
Notes:

- Aquarius vector CPU being compiled
- 450 MCA3 library parts (but many are high and low power versions of each other)
- Over 4000 .NDF files read (mainly because STGX and VRGX are described by hierarchical .NDF files)
- One third of nodes entering code generation produce no code
- Approximately six bytes per generated instruction
- Over 3.6 Mbytes of I-stream generated
- Generated .MAR file almost 100K blocks
- Process's peak virtual size ~2/3s of VMS V4 limit
- 'Process update' statistics will be examined later . . .

Final Simulation Layout



User Interface Layout



MEDUSA User Interface Command Syntax

- @file_spec
- BACKMAP number
- CALL routine_name [param_spec]
- CANCEL { { MATCH / TRACE / WATCH } /ALL | PREFIX [/ALL] }
- DEPOSIT [mach_switch] stor_name = number
- DISABLE { AXE_MACHINE / MACHINE } [mach_num]
- ENABLE { AXE_MACHINE / MACHINE } [mach_num]
- EXAMINE [mach_switch] stor_list
- EXIT
- GO
- GS_LOAD
- GS_UNLOAD
- LOAD { { /ABS / /EXE [/BASE=number] / /TEXT } [mach_switch] | /GS_LOAD | /GS_UNLOAD } file_spec

MEDUSA User Interface Command Syntax (cont.)

- MOVE [mach_switch] [move_switches]
  stor_list => stor_list
- QUIT
- RESTORE [mach_switch] file_spec
- SAVE [mach_switch] [/KEEP=number] file_spec
- SET { MACHINE mach_num | MATCH [mach_switch] stor_name = number | NOVERIFY | PREFIX [name] | TIME [mach_switch] number | TRACE [mach_switch] stor_list | VERIFY | WATCH [mach_switch] stor_list }
- STEP [number]
- SHOW { { MATCH | TIME | TRACE | WATCH } [mach_switch] | PREFIX }
- SPAWN [/NOWAIT] [/OUTPUT=file_spec] [char_str]
- SYNONYM stor_list

MEDUSA User Interface Command Syntax (cont.)

Where:

a. number –> { hexadecimal_num#16 | octal_num#8 | decimal_num[#10] }

b. stor_list –> stor_name { , stor_name }

c. stor_name –>
name [[number[:number]]] [<number[:number]>]

d. name –> { letter | - | % } { letter | digit | _ | % | . | - | $ }

e. mach_switch –> /MACHINE:mach_num

f. mach_num –> number | ALL | DYNAMIC

g. routine_name –> letter { any character except (}

h. param_spec –> ([parameter { , parameter }])

i. parameter –> number | char_str

j. char_str –> " { and character except " } "

k. move_switches –> move_switch { move_switch }

l. move_switch –> { /ODDPARITY | /INVERT | /EVENPARITY | /ACCUMULATE | /CHECK }

Stor_name Classes

Signals:

- Scalar specifications:
  - name
  - name<number> (where 0<=number)
  - name and name<0> are different!!
- Vector specification: name<number1:number2> (where 0<=number2<number1)

RAMs:

- RAM's logical reference designator used as storage name
- Specification: name[number1[:number2]][<number3[:number4]>]
  - '[' and ']' distinguish RAMs from signals
  - 0<=number2<number1<RAM's total elements
  - 0<=number4<number3<RAM's total bit width
  - Element range specification optional
  - Bit range specification optional
  - If no bit subscripting, full width used

Stor_name Classes (cont.)

Global Section Storage:

- Same format as RAMs, but names are reserved:
  - VAXGR[15:0]<31:0> (VAX GPRs)
  - VAXPRVGPR[255:0]<31:0> (VAX IPRs)
  - VAXVECTOR[2047:0]<31:0> (Vector regs.)
  - VAXVECCNTL[3:0]<31:0> (Vector control)
  - VAXMEM[524287:0]<31:0> (Main memory)
  - VAXCASENUMBER[1:0]<31:0> (AXE/MAX case#)
- Same format as signals, but name is reserved: VAXPSL<31:0> (VAX PSL)

General notes:

- Bit ranges greater than 32 are broken up into 32 bit chunks
- Wildcarding not directly available . . .

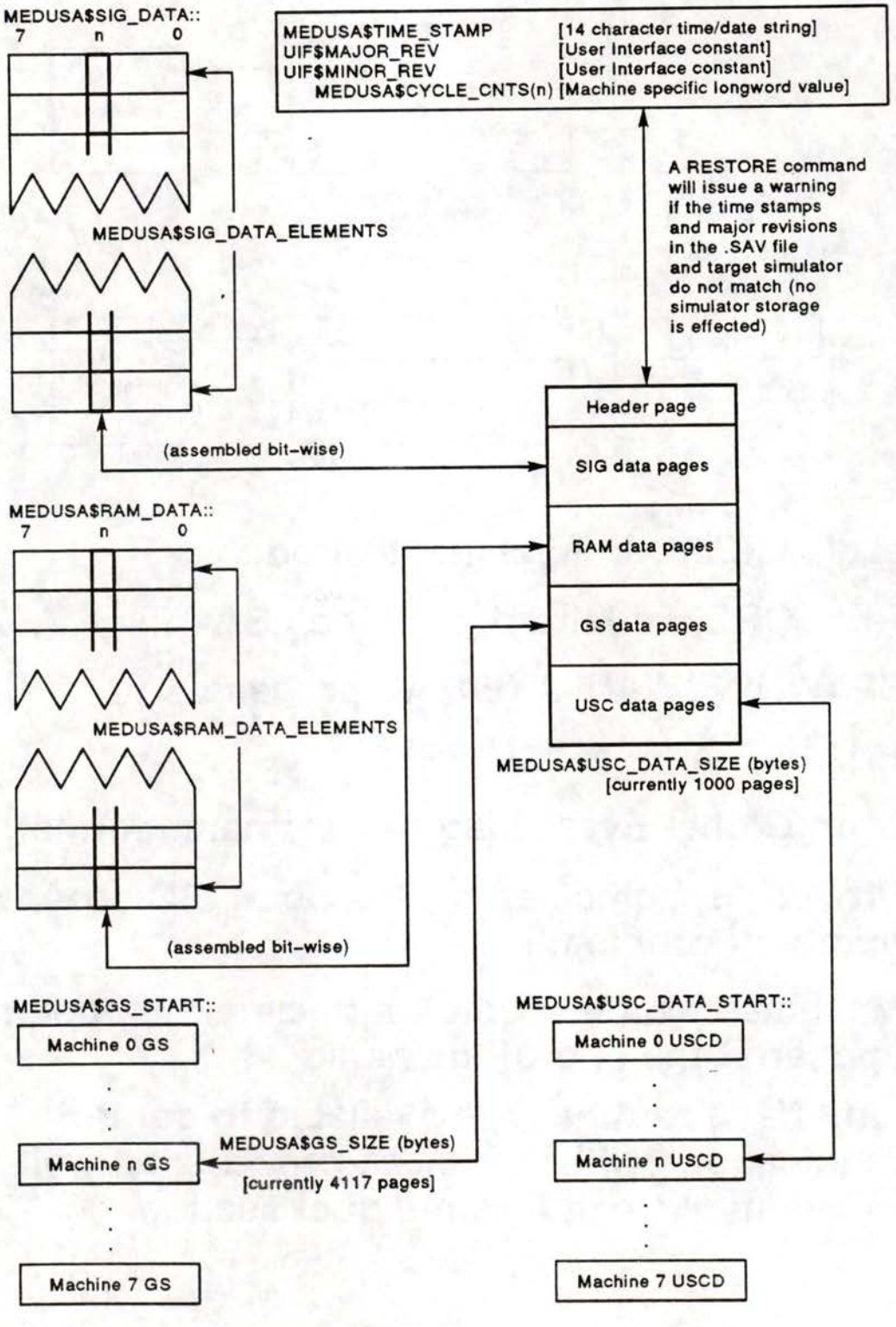
User Interface Concepts

- ENABLE/DISABLE MACHINE: indicates whether or not user is interested in machine's activities (machines are always doing something!)
- ENABLE/DISABLE AXE_MACHINE: establish/terminate communications with associated AXE/MAX process
- SET MACHINE: sets default 'scope' to particular machine (most commands understand 'all machines' scope also)
- PREFIX stack: maintains character strings to precede storage names by default (similar to DECSIM SCOPE, but does not work exactly the same . . .)
- STEP: simulate specified number of machine cycles (default is a single cycle)
- TRACE point: if storage entity has changed during the last simulated machine cycle, display new value
- WATCH point: like TRACE point, but also returns control back to user interface
- MATCH point: like WATCH point, except triggered by storage entity matching a specified value (versus simply changing)

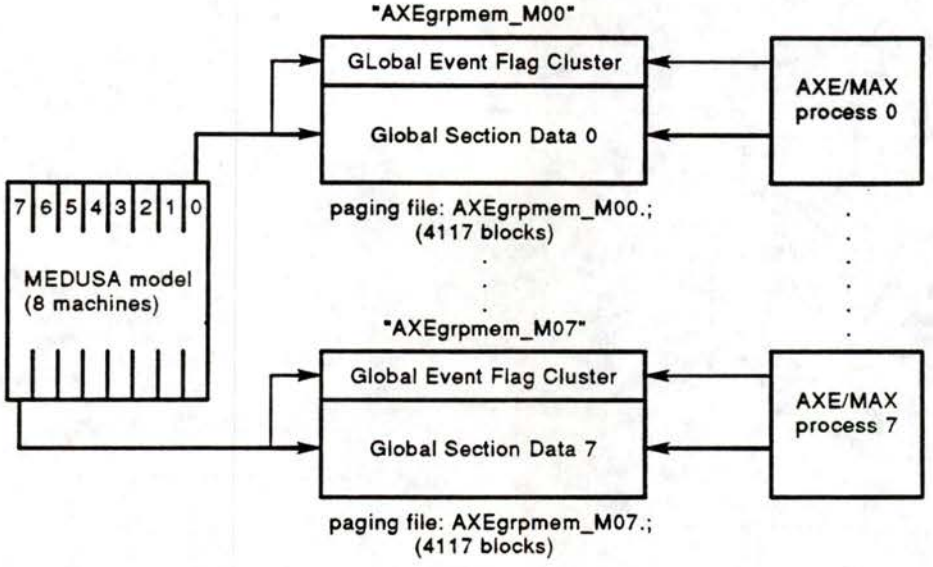
Miscellaneous User Interface Commands

- CALL: method for executing specific user supplied code routines
- LOAD/ABS: method for loading a .ABS file (created by EXETOABS) into particular machine's global section data area
- LOAD/EXE: method for loading a .EXE file (binary memory image) directly into VAXMEM (with an optional, non-zero starting address)
- LOAD/TEXT: method for loading RAM structures (patterned after DECSIM command by same name)
- GS_UNLOAD: 'macro' which moves data from the global section data area into or manipulates simulation signal or RAM storage (i.e. prepare AXE/MAX cases or .ABS file data for simulation)
- LOAD/GS_UNLOAD: loads GS_UNLOAD 'macro'
- GS_LOAD: 'macro' which moves data from simulation signal and RAM storage into the global section data area (i.e. allow AXE/MAX to check case results)
- LOAD/GS_LOAD: loads GS_LOAD 'macro'
- SYNONYM: display all the storage names assigned to the same location as the storage name specified

.SAV File Format



AXE/MAX Interface



For each AXE/MAX machine desired:

- **RESTORE 'initialized machine' .SAV file**
- **SPAWN/NOWAIT AXE/MAX process**
- **ENABLE AXE_MACHINE**

Notes on Global Event Flag Cluster name format:

- **Process specific name uses octal UIC group/member number ('grpmem')**
- **Machine specific, two digit decimal number, must appear at the end of the name . . .**

User interface routines are available to control communications with AXE/MAX process, as well as data movement to and from global section

User Interface Deficiencies:

- Log file support
- PRINT command
- CANCEL TRACE/WATCH/MATCH stor_list
- Generalized 'macro' support
- Wildcarding within storage names
- Storage name abbreviations
- HELP command
- Any others?

None of the above were required (or seriously missed?) for debugging Aquarius MEDUSA models . . .

Notes on User Supplied Code

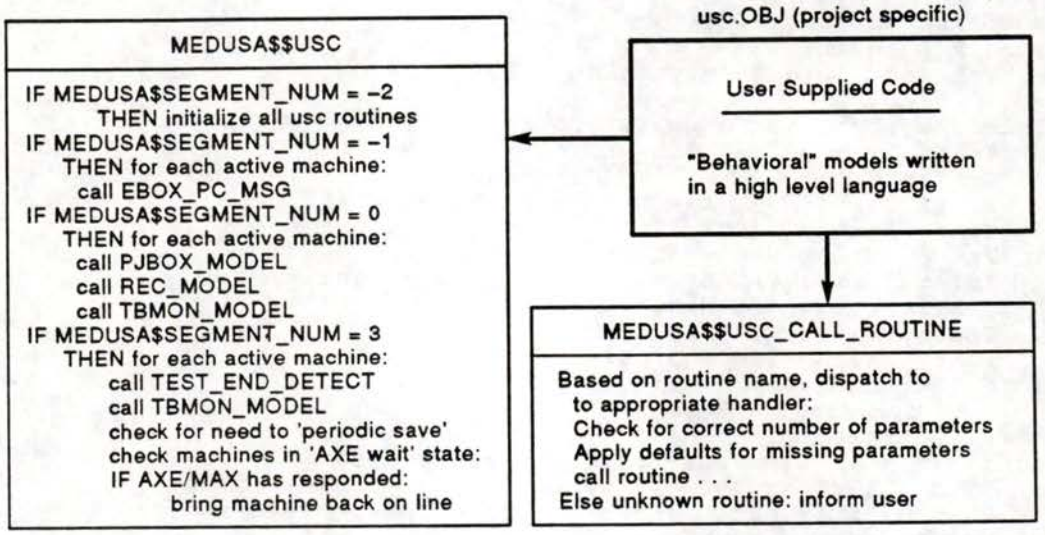
Goals for supporting user supplied code:

- **Provide access to all simulation resources, either directly or indirectly**
- **Yet isolate user from implementation details whenever possible (user interface routines are layered as simulator 'system services')**
- **Put as few restrictions on the user as possible**
- **But must be able to save state of behavioral models, just as user interface does for other simulation state**

'User Supplied Code' Data usage:

- **User must determine behavioral model states which need to be saved**
- **Design data structure representing that saved state**
- **Negotiate with other behavioral model designers USCD offset where virtual structures will reside**
- **User interface can then SAVE and RESTORE entire simulation state without involving behavioral models directly . . .**

Aquarius CPU Model User Supplied Code Layout



Two routines required to link simulator without errors:

- **MEDUSA$$USC**
  - **MEDUSA$SEGMENT_NUM is an implied parameter (none are explicitly passed)**
  - **MEDUSA$SEGMENT_NUM used to determine what actions, if any, should occur during this call to MEDUSA$$USC . . .**
- **MEDUSA$$USC_CALL_ROUTINE**
  - **'Parameter list' passed as parameter**
  - **First parameter list entry is name of user supplied code routine to be called**
  - **User responsible for actual routine calls, though there are user interface routines which help manipulate parameter list . . .**

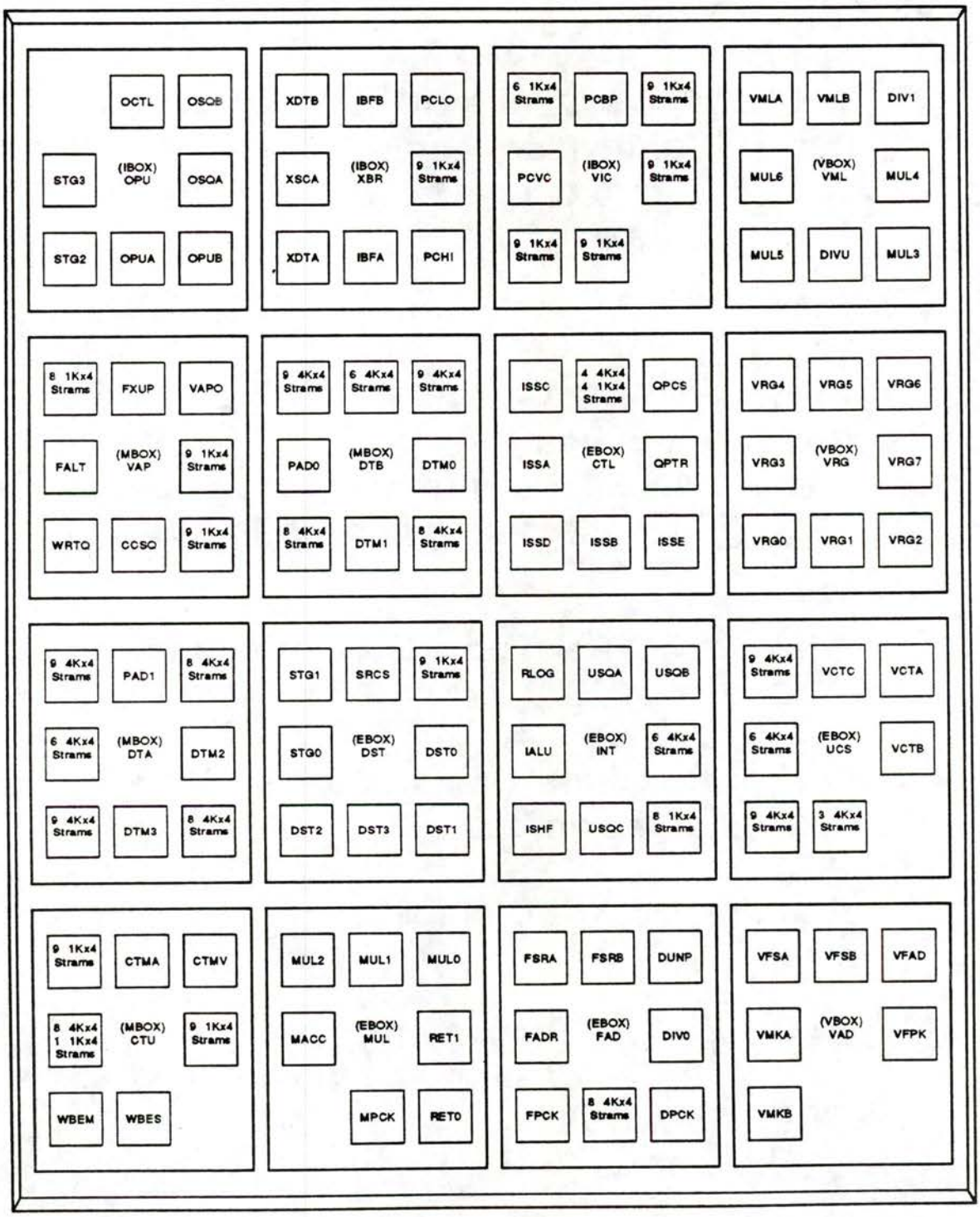
Aquarius CPU Model CALLable Routines

```
$RUN/NODEBUG NDF$REL:AQUARIUS

        MEDUSA User Interface version 1.19

        Initializing "user supplied code" routines:
           Initializing TEST_END_DETECT routine . . .
           Initializing EBOX_PC_MSG routine . . .
           Initializing GPR_xxx_MAP data structures . . .
           Initializing UTEMP_REG_MAP data structure . . .
           Initializing VRGX_MAP data structure . . .
           Initializing CACHE_xxx_INFO data structures . . .
           Initializing PJBOX_MODEL routine . . .
           Initializing REC_MODEL routine . . .
           Initializing TBMON_MODEL routine . . .
        Initializing user interface itself . . .

machine0> call directory
   Current "user supplied code" callable routines:
      DUMP_STREG("EBOX_or_IBOX",streg_bank_num,machine_number)
      DUMP_VAXMEM("file_name",start_va,end_va,pa_to_va_offset,machine_num)
      FAST_SWEEP(machine_number)
      LOAD_UTEMP_REG(register_num,value,machine_number)
      MOVE_GPRS_FROM_GS(machine_number)
      MOVE_GPRS_TO_GS(machine_number)
      MOVE_VRGX_FROM_GS(machine_number)
      MOVE_VRGX_TO_GS(machine_number)
      PEEK(set_number,physical_address,machine_number)
      PERIODIC_SAVE(save_period,"file_name",saves_to_keep,machine_number)
      PJBOX_DEBUG(numeric_parameter,machine_number)
      PJBOX_FLAGS(numeric_parameter,machine_number)
      PJBOX_INIT()
      PJBOX_JDUMP(transaction_count,machine_number)
      PJBOX_LATENCY(memory_latency,machine_number)
      PJBOX_SWEEP(machine_number)
      PJBOX_WIO(register_address,data,machine_number)
      REC_DUMP(machine_number)
      SAVE_AXE_CASE(case_number,restart_number,machine_number)
      STALL_CNT(stall_cnt,sweep_cnt,machine_num)
      TBMON(machine_number)
      WILDCARD("storage_name_str")
machine0> exit
$
```

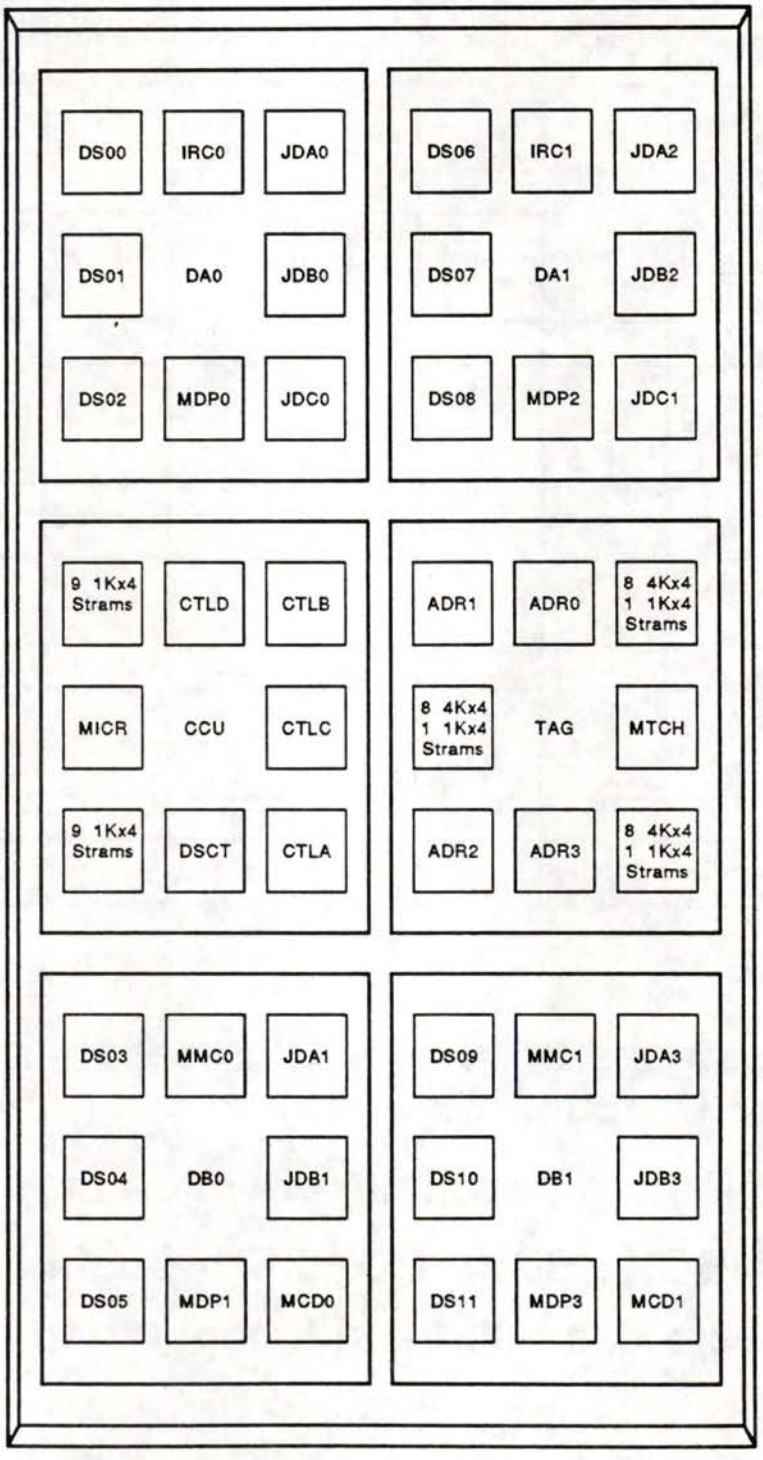
Aquarius "Hi-tech" Components

- MCA III
  - 256 signal I/Os (TAB)
  - 414/200/224 MCELLs/OCELLs/ICELLs
  - 8 to 10K equivalent gates
  - 200 psec gate delay
  - 33 watts maximum
- STRAMs
  - TABBED I/Os
  - Latches on ADDR, DATA IN and DATA OUT
  - Self timed writes
  - 1Kx4 @ 4.5/5.5 nsec (read access from ADDR)
  - 4Kx4 @ 9.5 nsec (read access from ADDR)
- HDSC/MCU
  - 4 in. x 4 in. Polyimide/Copper substrate
  - CDC plus 8 MCA IIIs (or 9 STRAMs/MCA III)
  - 804 I/Os
  - 9 layers (1 signal pair/1 pad/6 power and ground)
  - 300 watts maximum
  - FRU
  - Requires TABBED chips
- Modules
  - 24 in. x 24 in. (maximum)
  - 24 layers (5 signal pairs/1 clock/2 pad/11 ref.)
  - CPU - 16 MCUs
  - SCU/Memory - 6 MCUs

Aquarius CPU Floorplan



Aquarius SCU Floorplan



MCA Equivalence Calculations

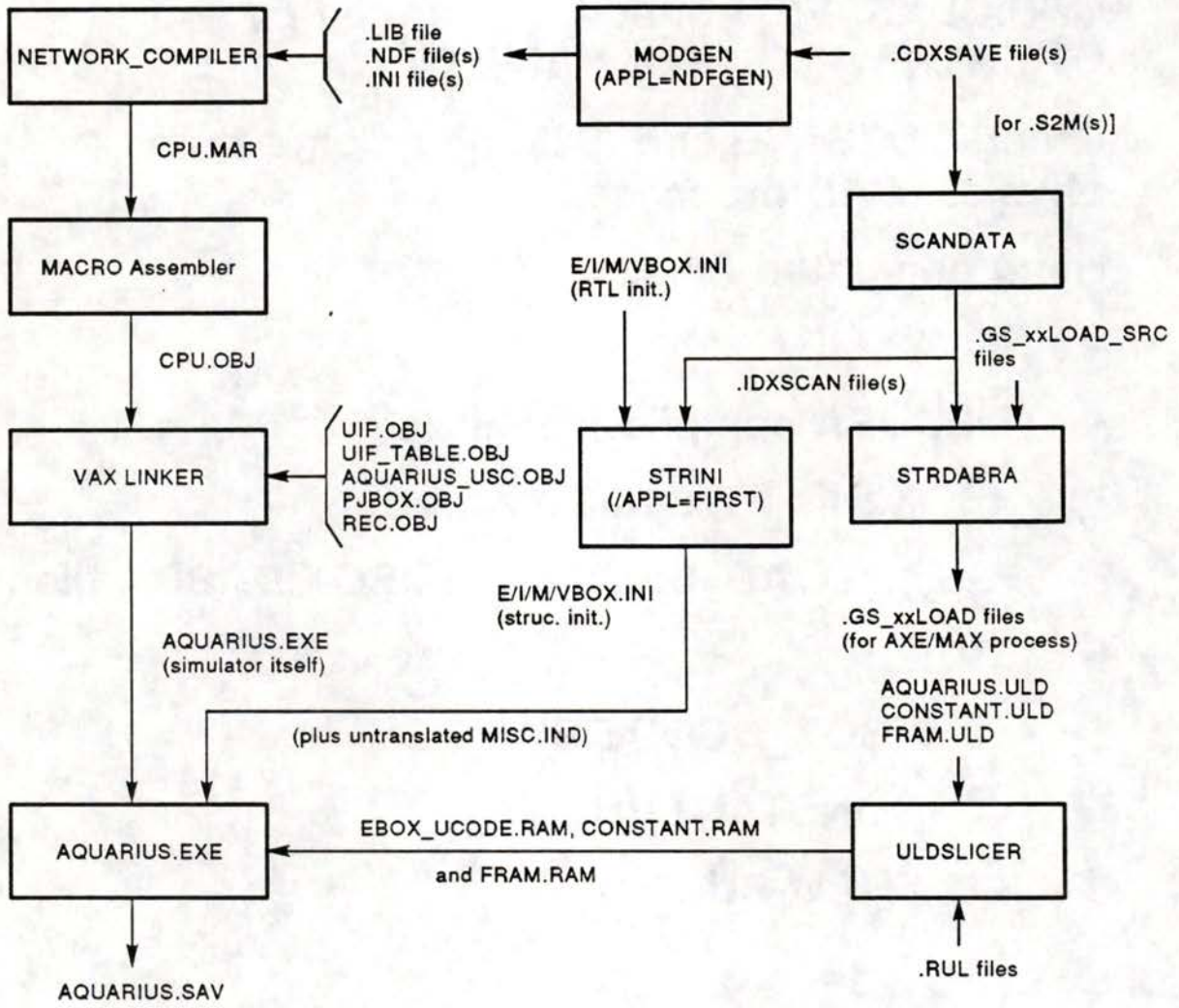
		Design Type (including "MCA equivalence" value)							
	System Component	MCAs (1)	OPU STGXs (1.5)	DST STGXs (4)	MULXs (3)	DIVXs (2)	VRGXs (2)	Stram Clusters (.1)	Total MCA Equivs
**	OPU	5	2	0	0	0	0	0	8.0
	XBR	7	0	0	0	0	0	1	7.1
	VIC	2	0	0	0	0	0	5	2.5
	IBOX	14	2	0	0	0	0	6	17.6
	VAP	5	0	0	0	0	0	3	5.3
	DTB	3	0	0	0	0	0	5	3.5
	DTA	3	0	0	0	0	0	5	3.5
	CTU	4	0	0	0	0	0	3	4.3
	MBOX	15	0	0	0	0	0	16	16.6
	CTL	7	0	0	0	0	0	1	7.1
	DST	5	0	2	0	0	0	1	13.1
	INT	6	0	0	0	0	0	2	6.2
	UCS	3	0	0	0	0	0	4	3.4
	MUL	4	0	0	3	0	0	0	13.0
	FAD	6	0	0	0	1	0	1	8.1
**	EBOX	31	0	2	3	1	0	9	50.9
	VML	3	0	0	4	1	0	0	17.0
	VRG	0	0	0	0	0	8	0	16.0
	VAD	6	0	0	0	0	0	0	6.0
	VBOX	9	0	0	4	1	8	0	39.0
	DAX	8	0	0	0	0	0	0	8.0
	DBX	8	0	0	0	0	0	0	8.0
	CCU	6	0	0	0	0	0	2	6.2
	TAG	5	0	0	0	0	0	3	5.3
	SCU	43	0	0	0	0	0	5	43.5
**	CPU	69	2	2	7	2	8	31	124.1
**	System	112	2	2	7	2	8	36	167.6

Notes:

- OPU uses one third of STGX's capabilities (the rest is optimized away)
- Stram clusters have little effect on model size/speed
- IBOX+MBOX/EBOX/VBOX/SCU roughly equal in size

** models were compiled and timed (along with OPUA)

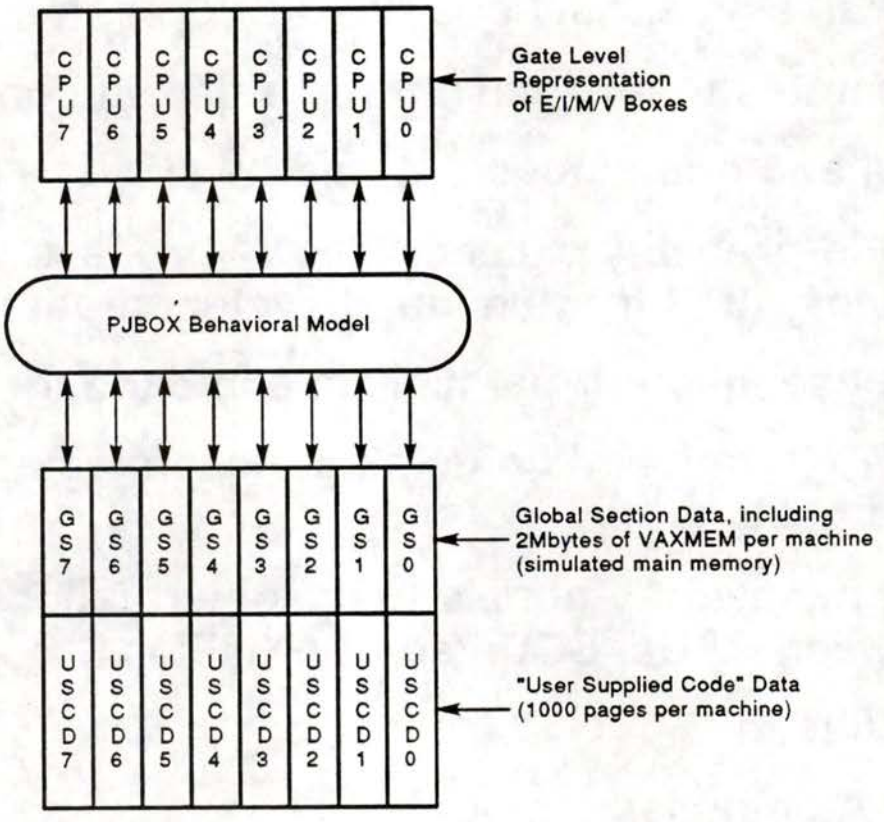
Aquarius Vector CPU Model Build Process



Model Build Process Notes:

- CPU.CDXSAVE indirectly generated by UTILGEN/WRITE_CONNECTIVITY
- Other .CDXSAVE and .S2M files come from physical CAD processes
- Hand generated files:
  - MCA3.LIB
  - MEDUSA compile-time .INIs
  - xBOX.INIs (RTL initialization)
  - .GS_xxLOAD_SRCs (from xBOX.DABRA files)
  - .RULs:
    - EBOX_UCODE.RUL
    - CONSTANT.RUL
    - FRAM.RUL

Aquarius Vector CPU Configuration



Notes:

- Vertical boxes represent code and data for a single machine (supplied by network compiler)
- Ovals represent code and data for behavioral models (incorporated by linker)
- 8 machine model drawn

PJBOX only supports single CPU configurations . . .

Options for Building System Model

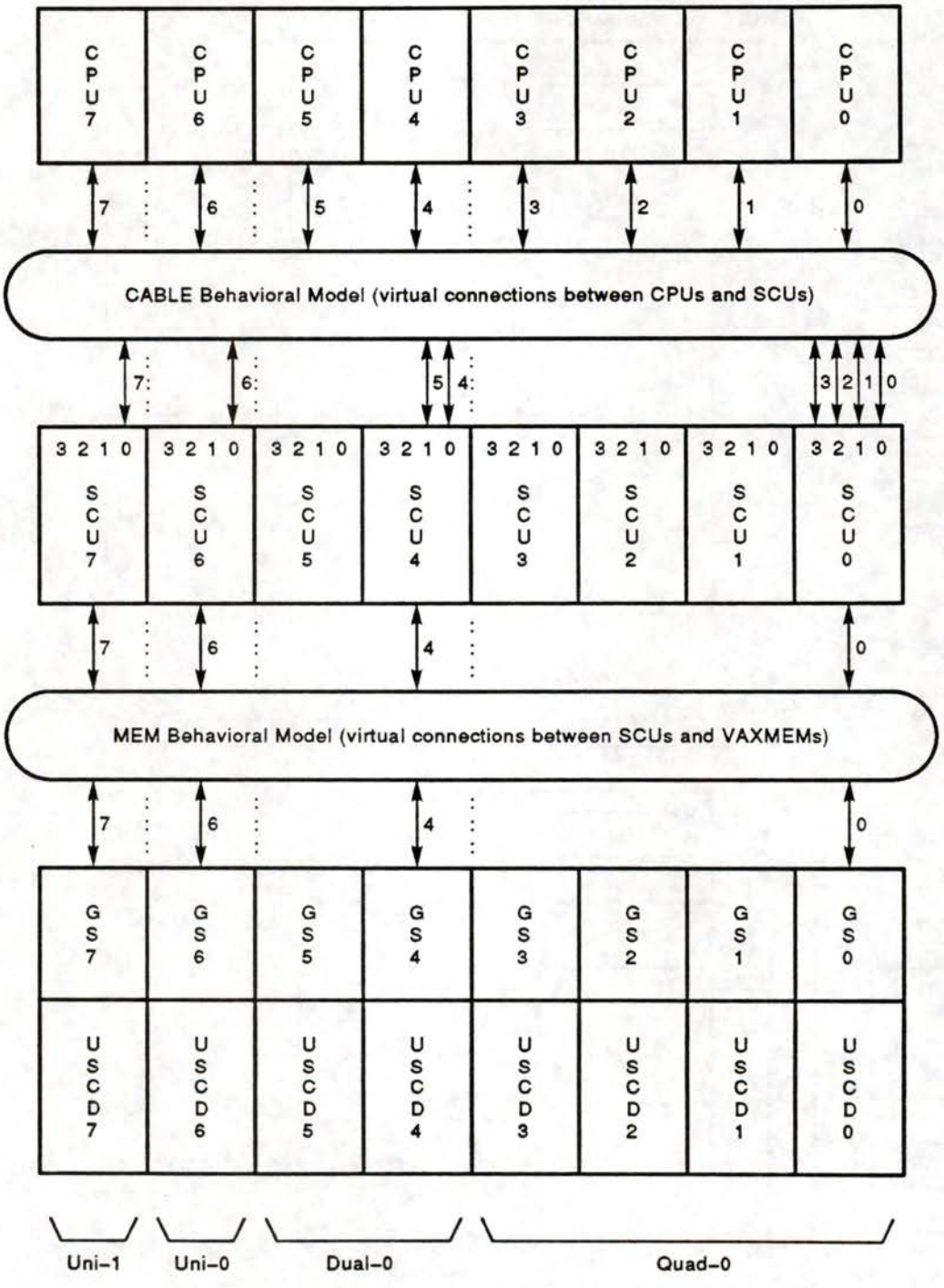
Build 'real' single, dual and quad CPU systems:

- Separate models mean three different compiles
- Dual and quad models would be slow
- Would probably exceed VMS V4 process virtual memory limit for dual/quad system model build

Instead, use inherent parallelism exhibited in CPU model:

- Compile 'system' containing disconnected CPU and SCU designs
- Use behavioral models to make virtual connections between CPUs, SCUs and VAXMEMs
- Results in:
  - A compilable model
  - Only one compile required
  - Single, dual and quad models equally fast!
  - Can configure different system types within a single simulation session
- Drawbacks are that:
  - Fewer tests can be run in parallel
  - Separate SAVEs are required to snap-shot the state of dual and quad models

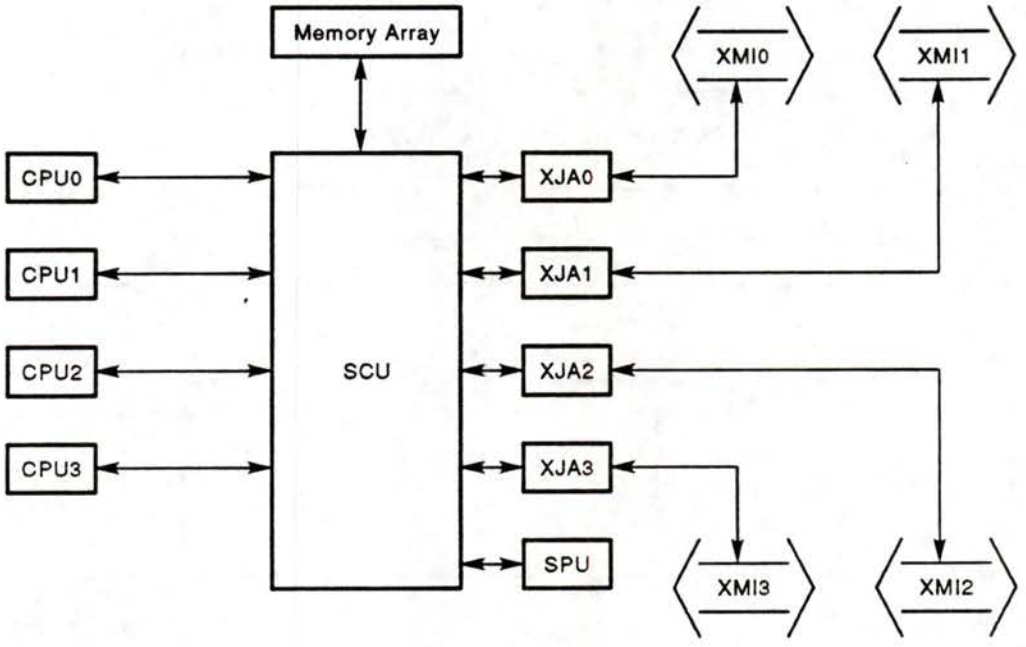
Aquarius System Configuration



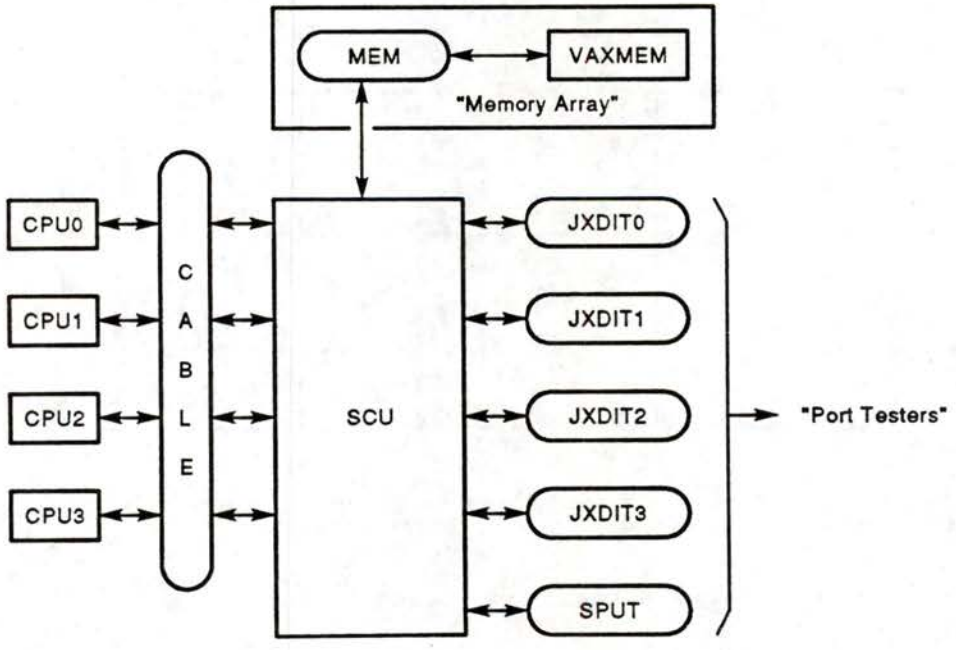
Note: SCU/GS/USCDs 5, 3, 2 and 1 are ignored/unused

Real versus Simulated System Block Diagrams

BLOCK DIAGRAM OF FULLY CONFIGURED SYSTEM



BLOCK DIAGRAM OF COMPARABLE MEDUSA SYSTEM



Why Were Port Testers Used?

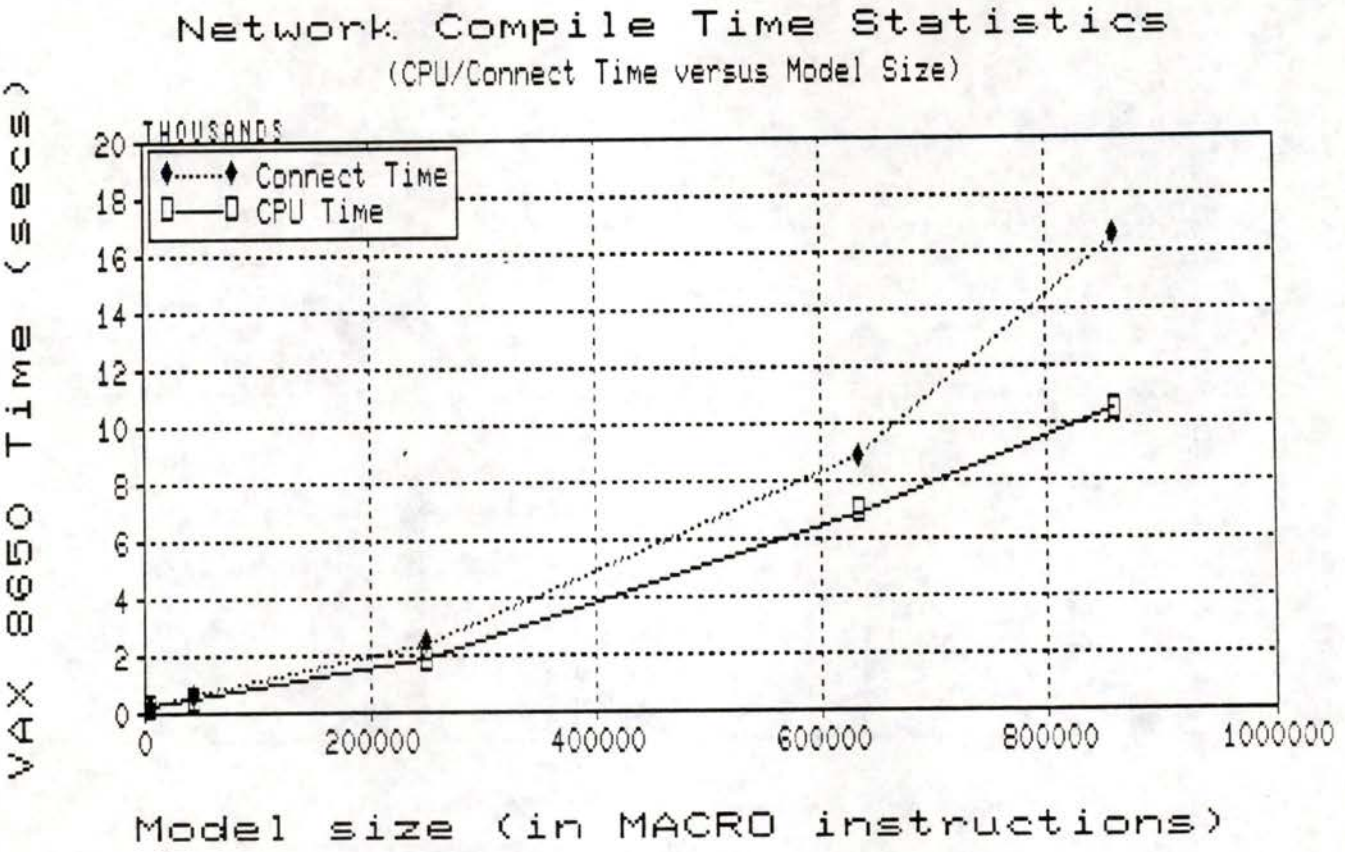
- XJA, SPU interface and Memory arrays do not use 'hi-tech' components:
  - New library parts required
  - Different clocking scheme potentially used
  - Different cycle times potentially used
- System model by itself was at VMS V4 process virtual memory limit (probably could not have added anything else!)
- XJA and SPU interface have been built and are working in the lab (XJA team slogan: 'the scum have won!'), so that it was felt remaining bugs were most likely in the SCU
- Modeling memory arrays, even if possible, would have caused significant AXE/MAX performance degradation (with questionable gain)
- Port testers can:
  - More directly control interface
  - Stress interface more heavily than actual device (while still within interface specs)
  - Act as demons . . .

Network Compile Time Statistics

Network Compiler Phase	OPUA8 Values	OPUA8 %	OPU8 Values	OPU8 %	EBOX8 Values	EBOX8 %	CPU8 Values	CPU8 %	SYS8 Values	SYS8 %
1) Read Parts Library	12.62	4.96	12.61	2.50	12.51	.64	12.53	.18	12.59	.11
	14.75	4.76	13.41	2.12	13.47	.54	13.46	.15	13.49	.08
	1388	6.83	1721	3.13	1408	.60	1380	.06	1538	.03
2) Read .NDF and .INI files	15.11	5.94	90.15	17.92	388.94	20.13	2167.92	31.23	3281.09	31.18
	16.38	5.29	104.96	16.61	437.39	17.58	2485.28	27.92	3711.17	22.35
	847	4.17	8797	16.04	37033	15.94	154991	7.27	515733	13.00
3) Node Fixup	27.48	10.80	57.34	11.40	185.07	9.57	726.12	10.46	1287.03	12.23
	29.27	9.46	62.53	9.89	204.70	8.22	838.09	9.41	1927.11	11.61
	839	4.13	14702	26.80	44684	19.24	772010	36.24	1449470	36.54
4) Order Network Nodes	.13	.05	1.34	.26	9.03	.46	58.04	.83	123.45	1.17
	.16	.05	1.48	.23	31.87	1.28	111.90	1.25	1327.38	7.99
	0	.00	6	.01	39206	16.88	403930	18.96	713168	17.98
5) Compress Network	.61	.23	.54	.10	6.46	.33	34.00	.48	57.56	.54
	.71	.22	.60	.09	7.56	.30	53.85	.60	97.58	.58
	1	.00	1	.00	24	.01	54224	2.54	94941	2.39
6) Output .MAR file	159.69	62.78	145.54	28.94	616.04	31.88	1962.27	28.27	3032.21	28.81
	186.28	60.21	185.66	29.38	828.88	33.32	2650.45	29.78	5737.17	34.56
	82	.40	41	.07	13908	5.98	403051	18.92	731923	18.45
7) Assemble .MAR file	26.03	10.23	179.07	35.60	687.73	35.59	1923.10	27.71	2656.66	25.24
	31.12	10.05	203.70	32.23	813.41	32.69	2361.81	26.54	3261.71	19.65
	2443	12.03	12935	23.58	66669	28.71	288645	13.55	392309	9.89
Link Simulator	12.66	4.97	16.31	3.24	26.30	1.36	55.94	.80	72.14	.68
	30.69	9.92	59.50	9.41	150.34	6.04	383.72	4.31	522.66	3.14
	14694	72.40	16637	30.33	29260	12.60	51888	2.43	67204	1.69
Total (CPU secs)	254.33	100	502.90	100	1932.08	100	6939.92	100	10522.73	100
(Connect secs)	309.36	100	631.84	100	2487.62	100	8898.56	100	16598.27	100
(Page faults)	20294	100	54840	100	232192	100	2130119	100	3966286	100
Subtotal (1+4+5)	13.36	5.25	14.49	2.88	28.00	1.44	104.57	1.50	193.60	1.83
	15.62	5.04	15.49	2.45	52.90	2.12	179.21	2.01	1438.45	8.66
	1389	6.84	1728	3.15	40638	17.50	459534	21.57	809647	20.41

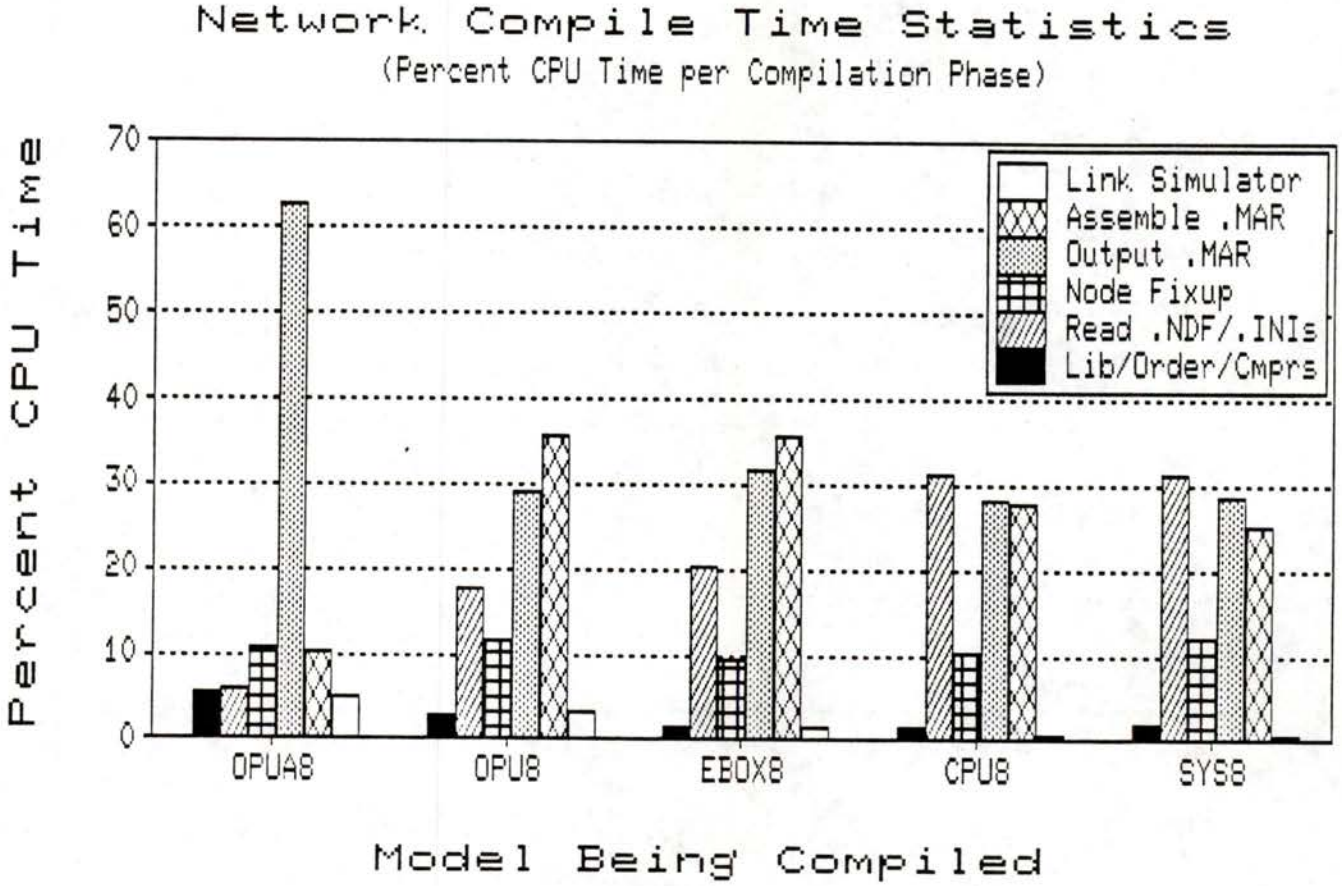
Notes:

- Values being tabulated:
  - CPU time (in seconds)
  - Connect time (in seconds)
  - Page Faults
- Overall percentages for each value above
- MACRO assembly and LINK statistics are included



Notes:

- **Compile time increases (almost) linearly with model size**
- **Excessive page faulting accelerates connect time**
- **MACRO assembly and LINK time included**



Notes:

- **CPU time eventually split among I/O intensive activities:**
  - **Reading .NDF and .INI files**
  - **Code generation and writing .MAR file**
  - **Assembling .MAR file**
- **Network compiler could output .OBJ file directly to reduce overall compile time . . .**

Run Time and Model Size Statistics

MEDUSA model	Cycles Simed	VAX 8800 CPU secs	VAX 8800 Cycles/sec	VAX 8650 CPU secs	VAX 8650 Cycles/sec	VAX 6200 CPU secs	VAX 6200 Cycles/sec	uVAX II CPU secs	uVAX II Cycles/sec
OPUA8	500000	892.63	560.14	985.24	507.49	1674.49	298.60	3443.38	145.21
OPUA16	500000	933.18	535.80	1174.68	425.65	1675.67	298.39		
OPUA32	500000	724.85	689.80	1257.26	397.69	1744.00	286.70		
OPU8	50000	929.72	53.78	1012.48	49.38	1643.32	30.43	3024.27	16.53
OPU16	50000	970.61	51.51	1114.42	44.87	2081.62	24.02		
OPU32	50000	1014.52	49.28	1207.25	41.42	2249.94	22.22		
EBOX8	10000	1130.13	8.85	1200.12	8.33	2817.03	3.55	3772.02	2.65
EBOX16	10000	1308.61	7.64	1359.31	7.36	3073.43	3.25		
EBOX32	10000	1510.12	6.62	1647.30	6.07	3765.76	2.66		
CPU8	5595	1936.21	2.89	1845.00	3.03	4269.91	1.31	6233.66	0.90
CPU16	5595	2568.79	2.18	2464.55	2.27	5312.66	1.05		
CPU32	5595	3185.75	1.76	3099.32	1.81	6733.43	0.83		
SYS8	6287	3113.30	2.02	3051.38	2.06	6913.04	0.91	42243.75	0.15
SYS16	6287	3973.65	1.58	3877.69	1.62	8351.37	0.75		
SYS32	6287	4781.41	1.31	4719.12	1.33	10245.01	0.61		

MEDUSA Model	Nodes into Code gen.	Nodes code gen. for	Signal Storage	RAM Storage	MACRO Instrs.	I-stream Bytes	Bytes per Instr.	File Sizes (in blocks) .MAR	.OBJ	.EXE
OPUA8	1584	1068 (67.42%)	1788	0	5337	30365	5.69	950	432	869
OPUA16			3576	0		31034	5.81	967	447	879
OPUA32			7152	0		31467	5.90	992	476	903
OPU8	16044	10004 (62.35%)	15674	0	42838	262671	6.13	8053	3057	3434
OPU16			31348	0		262745	6.13	8040	3071	3447
OPU32			62696	0		262960	6.14	7995	3098	3469
EBOX8	62356	43469 (69.71%)	71941	823296	249181	1507856	6.05	32958	11826	12044
EBOX16			143882	1646592		1508417	6.05	33259	11841	12056
EBOX32			287764	3293184		1666344	6.69	33367	12182	12388
CPU8	162576	112740 (69.34%)	177788	2732032	633739	3781730	5.97	94496	34366	34613
CPU16			355576	5464064		4458590	7.04	94650	35734	35956
CPU32			711152	10928128		4892800	7.72	94789	36624	36826
SYS8	238294	165836 (69.59%)	256079	3211264	859773	5636977	6.56	129771	47724	48256
SYS16			512158	6422528		6430671	7.48	129928	49319	49821
SYS32			1024316	12845056		6868957	7.99	130076	50219	50702

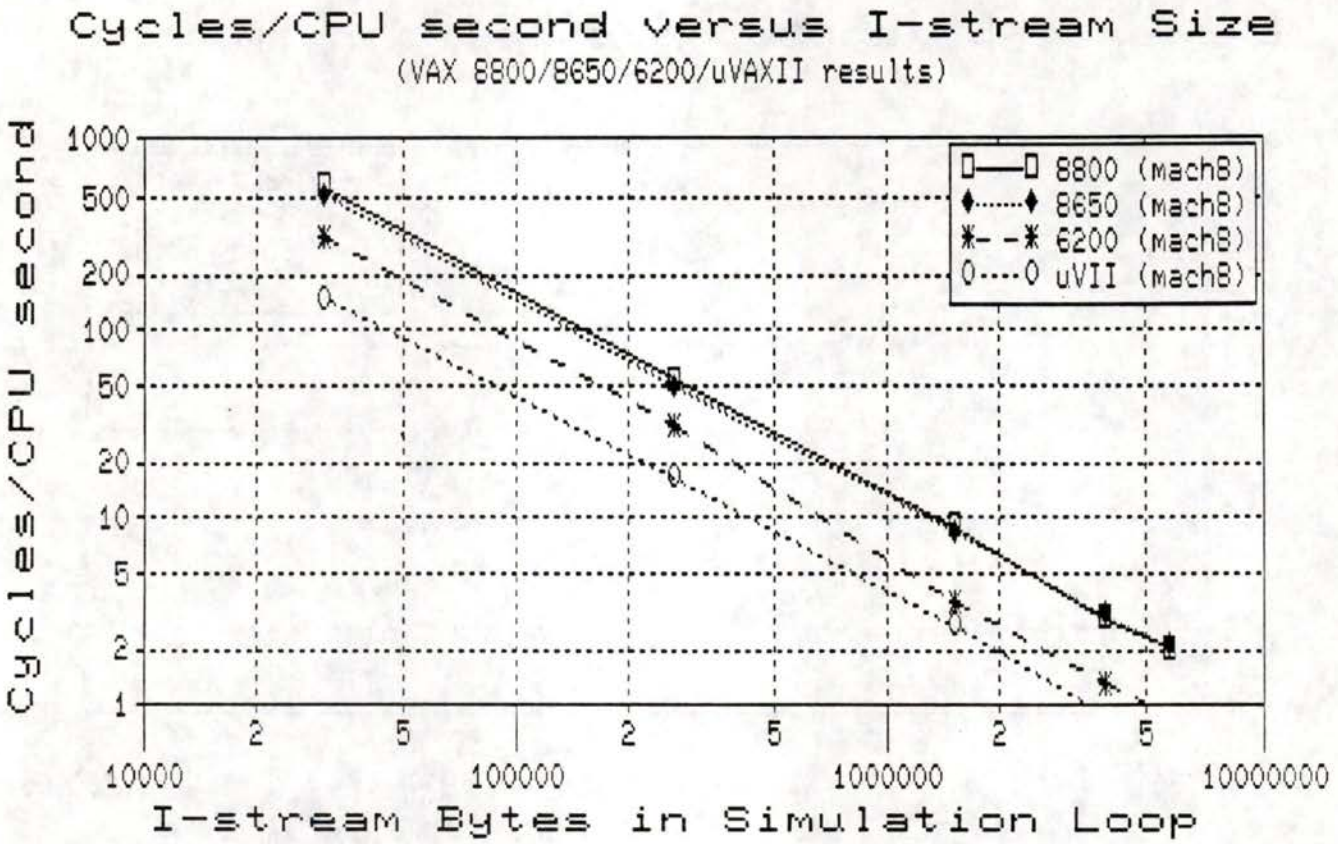
Run Time and Model Size Statistics (cont.)

Run Time Notes:

- OPUAx, OPUx and EBOXx models were simply cycled
- CPUx and SYSx models were real simulations (running QRT MACRO diagnostic)
- Only 8 machine models run on Micro-VAXII (due to excessive virtual size of other models)
- 8, 16 and 32 machine models have decreasing cycles/CPU second values (except OPUA on VAX 8800, which happens to fit in cache)

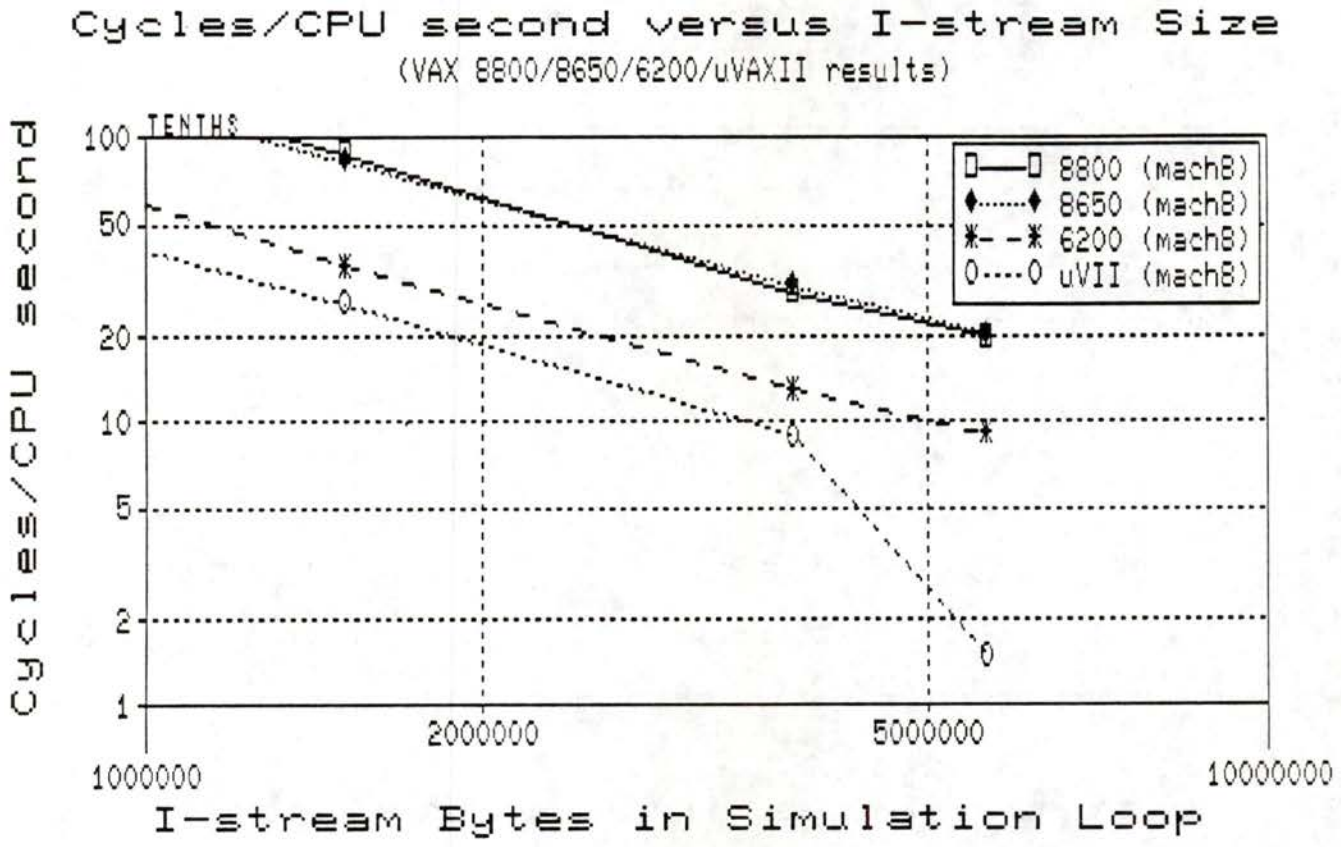
Model Size Notes:

- No code is produced for 30% of nodes going into code generation
- Greater increase in bytes per instruction when signal storage size exceeds 192K bytes (i.e. exceeds word displacement range of three fixed base registers)
- Compiling CPUx and SYSx models required a lot of disk space



Notes:

- **Logarithmic scaling for both axes**
- **Straight line relationship (slope = -1) between model speed and I-stream size**
- **VAX 8800 and 8650 performance essentially the same, except that:**
  - **8800's larger cache better for smaller models**
  - **8800's multi-processor memory contention worse for larger models . . .**

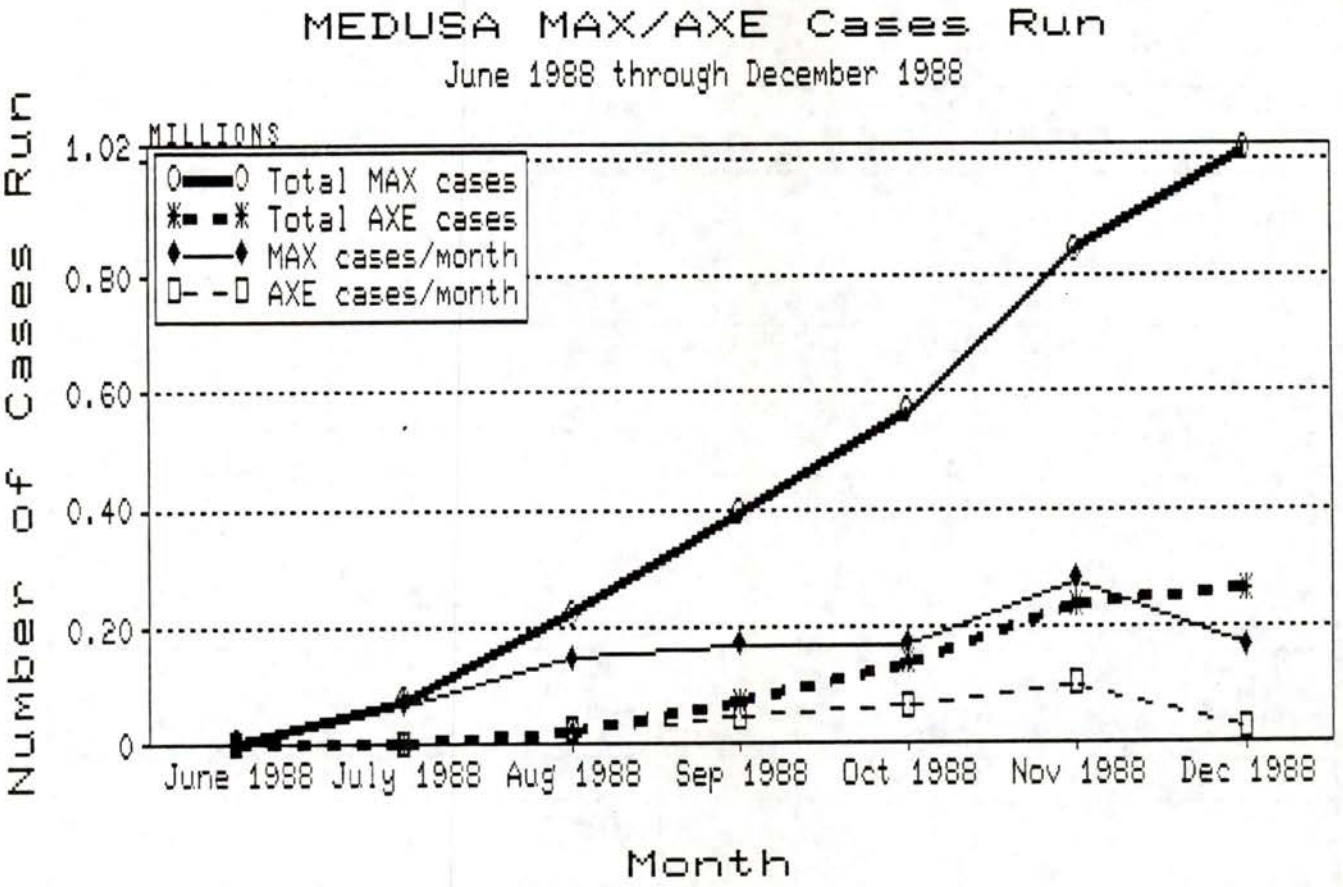


Notes:

- Detail of previous graph's lower quadrant
- Straight line relationship holds for 8800/8650/6200
- System model size exceeds working set and begins page faulting on Micro-VAXII

MEDUSA Usage and Bugs Found Statistics

Week's Date (Friday)	MEDUSA AXE Cases /week	MEDUSA AXE Running total	MEDUSA AXE Bugs /week	MEDUSA MAX Cases /week	MEDUSA MAX Running total	MEDUSA MAX Bugs /week	Misc MEDUSA MACRO CPU diag	Misc MEDUSA MACRO SYS diag	Misc DECSIM/ ZYCAD
06/10/88	0	0	0	71	71	0	2	0	0
06/17/88	0	0	0	471	542	3	3	0	0
06/24/88	0	0	0	89	631	0	5	0	0
07/01/88	0	0	0	5143	5774	5	3	0	0
June total	0	0	0	5774	5774	8	13	0	0
07/08 88	0	0	0	8528	14302	10	1	0	0
07 15/88	0	0	0	16233	30535	4	2	0	0
07/22/88	0	0	0	11854	42389	9	1	0	1
07/29/88	1710	1710	0	32244	74633	8	3	0	0
July total	1710	1710	0	68859	74633	31	7	0	1
08/05 88	1780	3490	2	31927	106560	2	3	0	1
08 12 88	1946	5436	2	28850	135410	2	1	1	0
08 19/88	1377	6813	1	20524	155934	4	4	0	0
08/26/88	9181	15994	4	36860	192794	5	2	1	0
09/02/88	9000	24994	4	31204	223998	0	1	1	0
Aug total	23284	24994	13	149365	223998	13	11	3	1
09/09 88	6467	31461	2	37947	261945	2	3	5	0
09 16/88	10709	42170	7	44960	306905	1	4	2	0
09/23 88	26535	68705	3	40163	347068	2	2	1	0
09 30/88	2000	70705	0	47039	394107	2	0	3	0
Sept total	45711	70705	12	170109	394107	7	9	11	0
10/07 88	0	70705	0	37371	431478	3	2	1	0
10 14 88	10300	81005	0	40705	472183	3	0	4	0
10 21 88	36304	117309	0	45851	518034	2	1	3	0
10 28/88	19780	137089	0	46658	564692	1	0	8	0
Oct total	66384	137089	0	170585	564692	9	3	16	0
11/04 88	42560	179649	0	42685	607377	1	0	5	0
11 11 88	22549	202198	0	83509	690886	1	0	5	0
11 18/88	25000	227198	0	83342	774228	2	1	4	0
12/02/88	9100	236298	0	69941	844169	4	2	8	0
Nov total	99209	236298	0	279477	844169	8	3	22	0
12/09 88	11000	247298	0	51938	896107	0	2	6	0
12 16 88	7500	254798	0	53418	949525	1	0	5	0
12 23 88	0	254798	0	47685	997210	1	1	5	0
12 30/88	7500	262298	0	15258	1012468	0	4	6	0
Dec total	26000	262298	0	168299	1012468	2	7	22	0
Grand Tot.	262298	262298	25	1012468	1012468	78	53	74	2

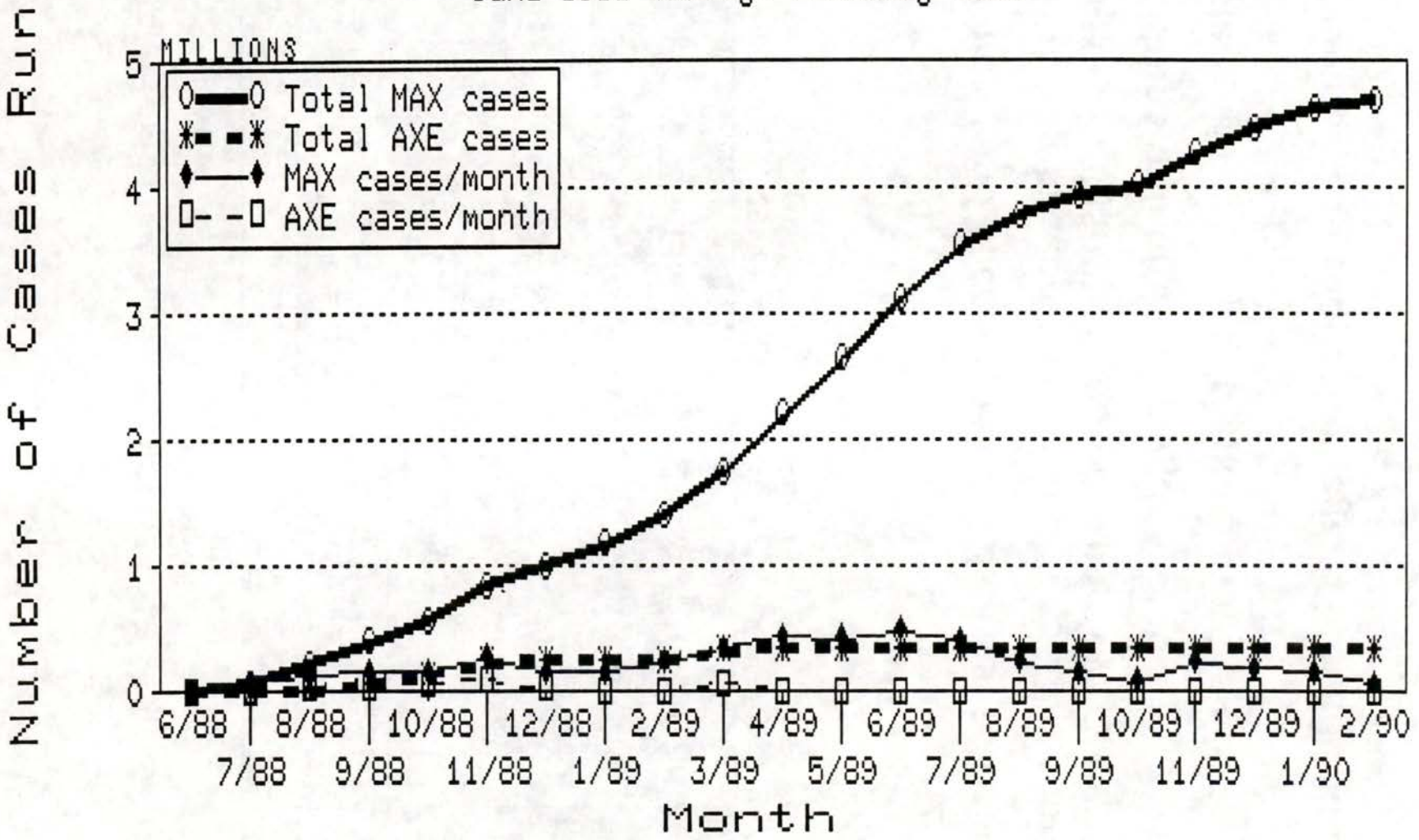


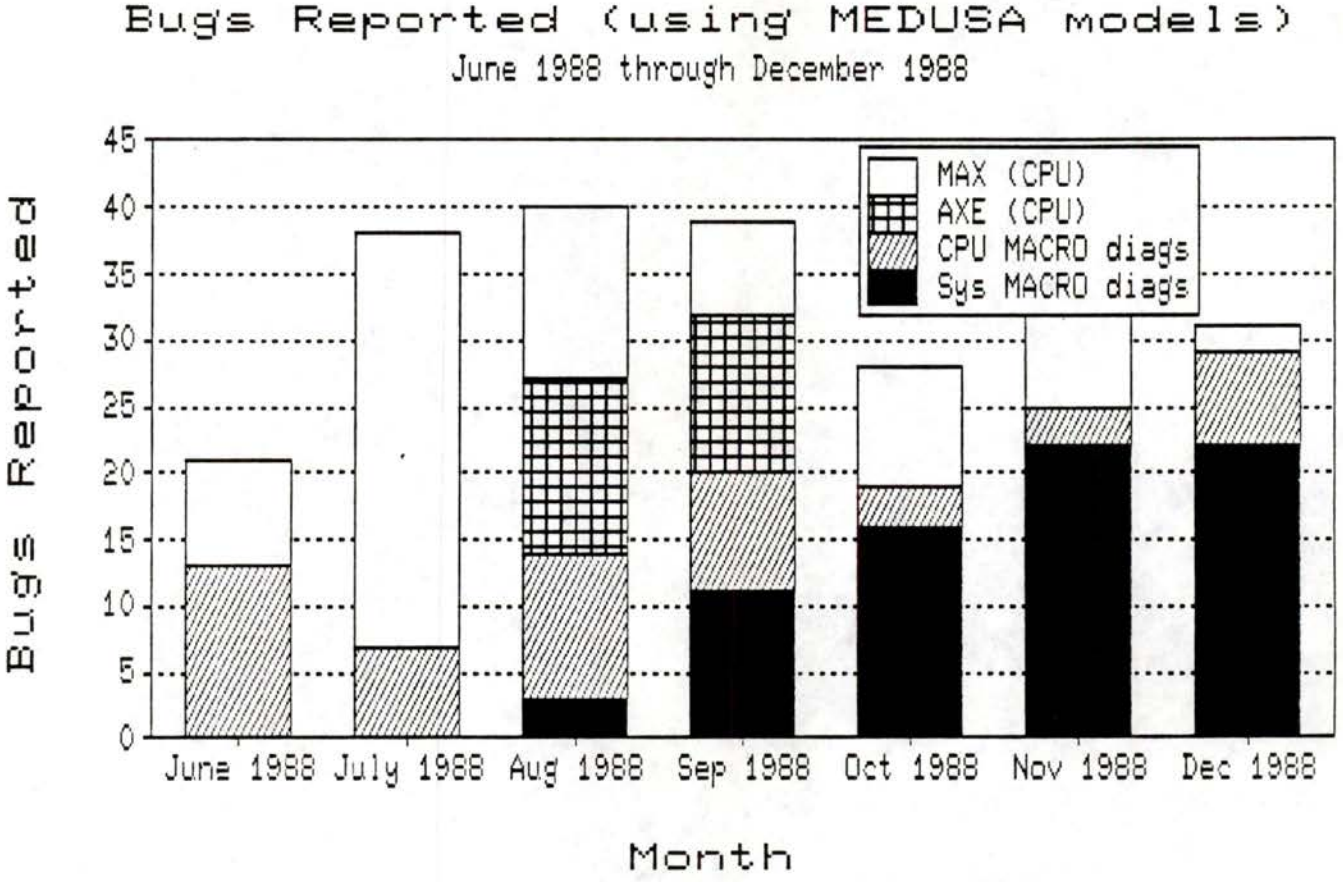
Notes:

- Over one million MAX cases run in seven months
- Steady progress was made running MAX
- MAX cases alone represent 2 minutes of real Aquarius CPU time simulated!
- AXE run sporadically to fill holes in MAX coverage
- Yet over 260000 AXE cases were run . . .

MEDUSA MAX/AXE Cases Run

June 1988 through February 1990



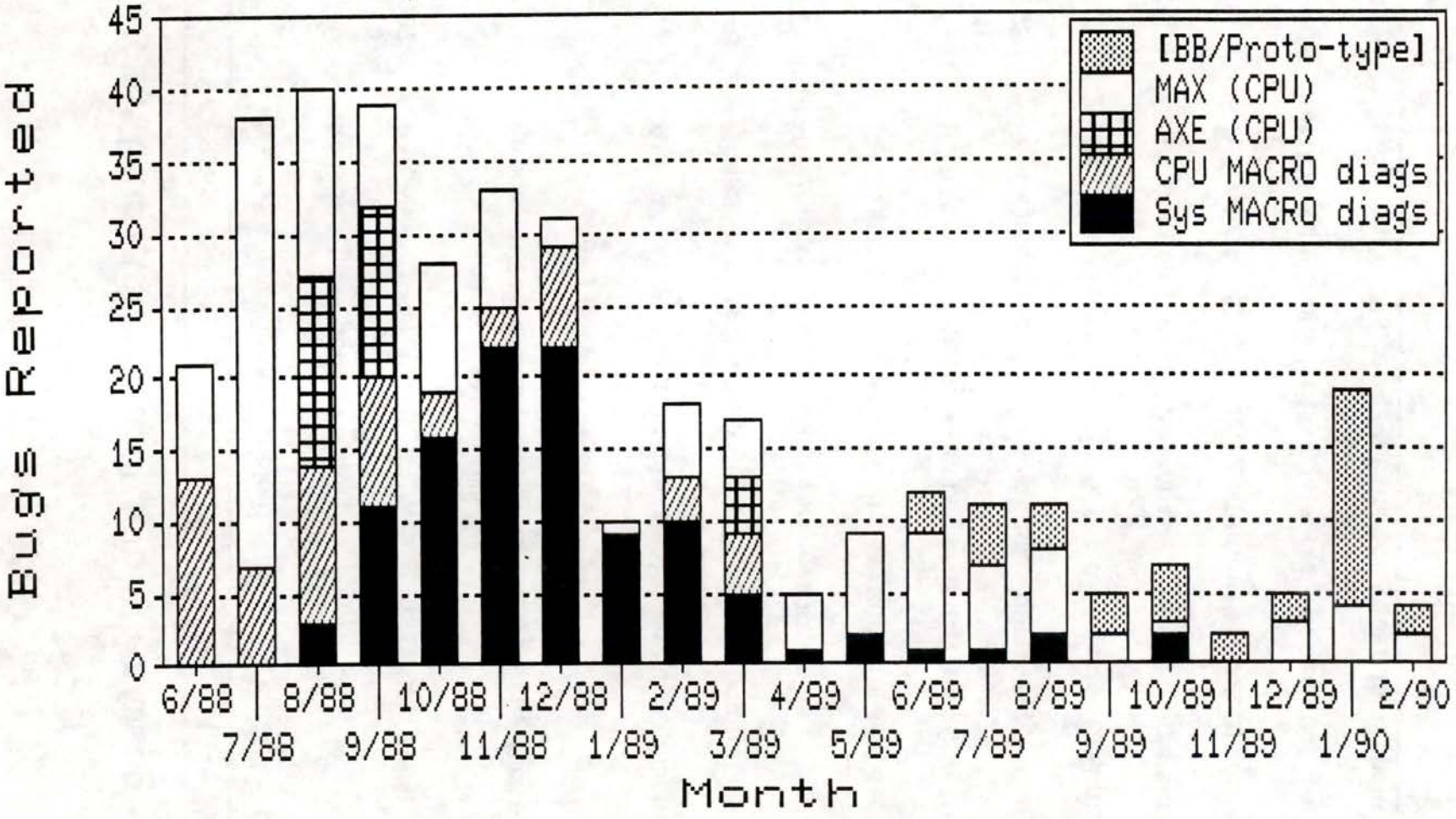


Notes:

- ***Structural*** **and** ***Micro-code*** **QAR data bases used**
- **CPU debug based on MACRO diagnostics prior to June (~20 bugs found in April and May)**
- **CPU bugs trail off beginning in October**
- **System debug ramps up during months of August and September . . .**
- **[November data clipped by legend - total is 33 bugs]**

Bug's Reported (using MEDUSA models)

June 1988 through February 1990



Significant Dates in MEDUSA Evolution/Usage

Date	Event
March, 1986	MEDUSA development begins
July 17, 1987	MEDUSA goes public . . .
December 24, 1987	Portion of QRT runs on CPU model - MEDUSA lives!
April, 1988	PJBOX added to CPU model (programs can now run out of memory, versus cache)
May, 1988	VBOX added to CPU model
June 9, 1988	AXE/MAX interface available (single simulation machine only)
June 24, 1988	Multiple simulation machine AXE/MAXing begins
June 26, 1988	HW cluster (10 VAX 8800s) becomes available
July 22, 1888	GS_xxLOAD process reduced from 30 to 10 CPU seconds
August 6, 1988	PJBOX default memory latency changed from 0 to 10 cycles
September 1, 1988	First system model available (duals/quads supported)
October 1, 1988	Start compiling 16 (versus 8) machine vector CPU models
October 26, 1988	500000th MAX case/42 seconds of Aquarius time simulated
November 3, 1988	GS_xxLOAD process reduced from 10 to 1.3 CPU seconds
November 24, 1988	MAX, with Character String generation, becomes available
January 4, 1989	1 millionth MAX case/2 minutes of Aquarius time simulated

MEDUSA AXE/MAX Run-time Statistics

```
MBOX fast cache sweep completed for machine 8 during cycle number 59816!!
%AIF-I-AXEPROCEXIT      AXE/MAX process exited - shutting down machine 8
%AIF-I-AXESTATDUMP      AXE status dump for machine 8:
                        Total number of AXE/MAX case run: 40                    (a)
                        Total number of AXE/MAX restarts: 503                   (b)
                        Total "useful" simulation cycles: 1133730               (c)
                        Total cycles waiting for AXE/MAX: 190                   (d)
%UIF-I-NOACTMACHS       No more active machines - returning to user interface
EXIT
%AIF-I-FINALSTATDUMP    Final AXE status dump:
                        Total number of AXE/MAX cases run: 1368                 (e)
                        Total number of AXE/MAX restarts:  20006                (f)
                        Total "useful" simulation cycles:  17616570             (g)
                        Total cycles waiting for AXE/MAX:  17603                (h)
                        AXE/MAX restarts per case:              14.624          (i)
                        Simulation cycles per restart:         880.564          (j)
                        "Wait cycles" per restart:               0.880          (k)
                        Total CPU time (in seconds):        626650.870          (l)
                        AXE/MAX cases per CPU hour:              7.859          (m)
                        "GS unload and load" CPU time:           1.138          (n)
                        Effective cycles per CPU second:        29.172          (o)
                        Available cycles per CPU second:        30.057          (p)
$EXIT
  MACRI        job terminated at  3-JAN-1989 17:09:05.38
  Accounting information:
  Buffered I/O count:         13891      Peak working set size: 28853
  Direct I/O count:           54616      Peak page file size:  260972
  Page faults:               282171      Mounted volumes:           0
  Charged CPU time:     7 06:19:10.93    Elapsed time:     7 08:46:31.63
```

Equations Used for Calculating Statistics	
(a), (b), (c) and (d) are tabulated for each machine running AXE/MAX	
(e) = summation of (a)s (f) = summation of (b)s (g) = summation of (c)s (h) = summation of (d)s	(i) = (f) / (e) (j) = (g) / (f) (k) = (h) / (f)
(m) = (e) / [(l) / 3600] (o) = (g) / [(l) - [(n) * (f)]] (p) = [[largest (b) + (c) + (d)] * num_machines] / [(l) - [(n) * (f)]]	

Hindsight being 20/20:

- Synthesis was less of a problem than expected (~15 functional bugs found, mainly in early 1987 using DECSIM structural models)
- ZYCAD size limitations a real problem (we have yet to get a vector CPU to work, let alone a system model . . .)
- RTL simulation much slower than expected (though changes to the clocking scheme made a significant difference in mid-1988)
- RTL boxes became available over an 18 month period, forcing a dependence on mixed mode simulation

MEDUSA's impact on Aquarius simulation effort:

- More readily accepted than anticipated
- More of a replacement for DECSIM/ZYCAD than expected (last behavioral and RTL model QARs entered August 1988)
- Use of AXE/MAX has made the difference in terms of solidifying CPU design
- Fast dual and quad system models allowed bugs to be found which were not possible with DECSIM
- Potential savings in proto-type debug time of an estimated six months to a year
- Final impact unknown until we have a proto-type!

Compiled logic simulators have come of age within DEC:

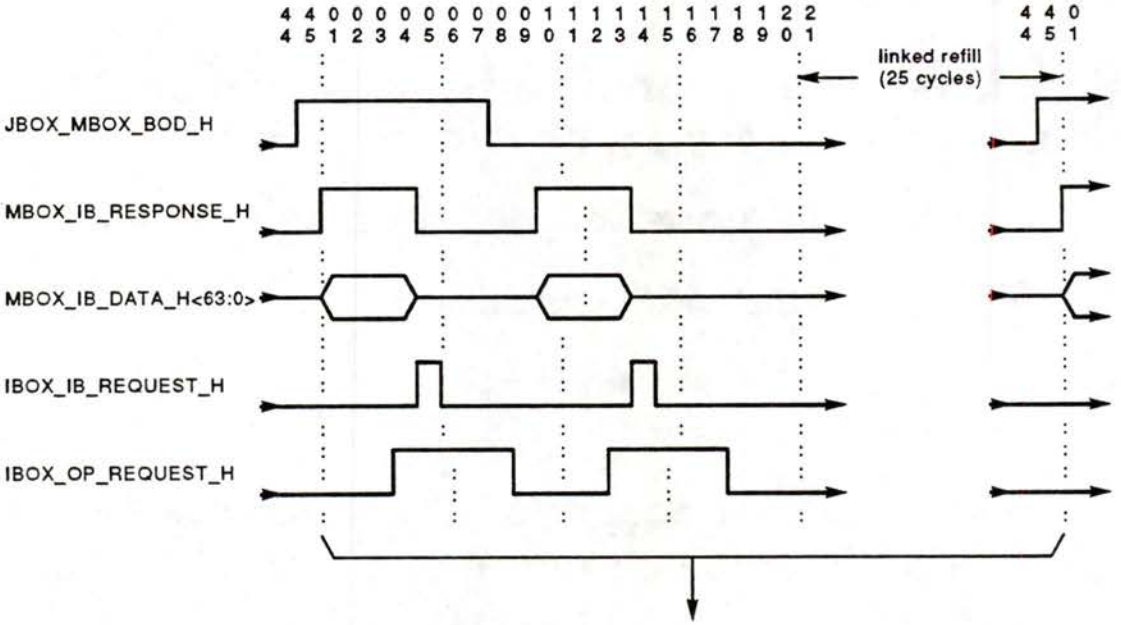
- Littleton has continually enhanced PRESTO and use it as their main method of simulation
- MEDUSA (as an advance development effort) has proven it's usefulness in Marlboro and probably will not be the final version . . .
- Hudson is considering developing their own compiled logic simulator . . .

What is needed for the future?

- Simulators which can be adapted to simulation needs for each project development stage:
  - Behavioral modeling capabilities
  - Fast mixed model simulation
  - Slower, but patchable, model when bugs are frequent
  - Fastest, most accurate model possible when design is stable (even at expense of flexibility)
  - Model speed needed for diagnosing bugs as well as for finding bugs in the first place . . .
- Investigate hardware accelerators for ultimate speed (but recognize fact that they may not cover full range of simulation needs)
- Combine efforts and share resources!!

Hypothetical MEDUSA Execution Model on Aquarius CPU

- Assume I-stream is infinitely long (i.e. 100% I-stream cache/TB miss)
- 6 I-stream bytes per instruction
- Simple specifiers (mainly register and word disp.)
- Therefore, 1 D-stream reference per instruction
- 100% D-stream cache/TB hit (degrade final figure by 20%)



$$\left[\frac{3781730 \text{ bytes}}{\text{CPU8 simulated cycle}} \times \frac{45.625 \text{ cycles}}{64 \text{ bytes}} \times \frac{16 \text{ nsec}}{\text{Aquarius cycle}}\right]^{-1} = 23.2 \frac{\text{simulated cycles}}{\text{Aquarius CPU sec}}$$

$$\frac{23.2 \text{ sim cycs/Aquarius CPU sec X 0.8 fudge factor}}{3.03 \text{ sim cycs/VAX 8650 CPU sec (measured for CPU8)}} = \boxed{\text{6.1 Aquarius to VAX 8650 speed ratio}}$$

Acknowledgments

Person or Group	Contribution
Jim Keller, Kevin Ladd, et. al.	PRESTO
Bob Stewart	Multiple, simultaneous simulation concept
Mike Evans	OUTGEN
Skip Gaede	MODGEN/APPL=NDFGEN
Dave Webb	UTILGEN/WRITE_CONNECTIVITY
Dale Keck	STRINI/APPL=FIRST STRDAB
Mark Palmer	AXE Interface routines LOAD/TEXT command
Jim Sloan	CHROMA-to-CADEX converter
Larry Herman and Matt Adiletta	STGX/MULX/DIVX support
Keith Westgate	ULDSLICER
Mark Baxter	PJBOX model CABLE and MEM models JXDIT model
Bill Rogers	VRGX .NDF files
Don Denning	REC_DUMP code TBMON code
Joe OConnor	SPUT model
Joe Macri, Brad Hollister + DV Group	"super-users"

- MAF 18
  - — 'Node fixup' no longer splits segmented and TA/TB latch pair nodes.
  - — All such library parts have been replaced with hierarchical .NDF files.
- MAF 33 through MAF 35
  - — The number of .NDF files read while compiling a vector CPU model has increased from ~4000 to ~21100, due to the new hierarchical .NDF files.
  - — The overall compile time for large MEDUSA models has been increased by up to 50%.
  - — The characteristics of the generated code has remained unchanged, except that there are extra signal names associated with wiring inside the new hierarchical .NDF files.
- MAF 42
  - — The list of global section storage entities has been extended to include:

    VAXINITTB[31:0]<31:0>
    VAXINVALTB<31:0>
    VAXINITCACHE[63:0]<31:0>
    VAXINVALCACHE<31:0>
    VAXINTERRUPT<31:0>
    VAXASCII[3:0]<31:0>
    VAXEA[15:0]<31:0>
    VAXBRSTATE<31:0>

    Although they are recognized by the user interface, they are not currently used.
- MAF 50
  - — A new user supplied code routine has been added:

    CORRECT_VPSR(machine_number)

    whose function is to clear VPSR<0> and set VPSR<7> if VAER<15:0> is not zero.
- MAF 55
  - — STRINI accepts .INI files as input and outputs .IND files (not .INI files, as depicted).
- MAF 62 through MAF 64
  - — Due to extra hierarchical .NDF files being read, these network compile time statistics are out of date. The general trends indicated, though, are probably still accurate.
- MAF 69 through MAF 70, MAF 72
  - — Over 5 million MAX cases and 7 minutes of real Aquarius time have been simulated.
- MAF 73
  - — The 30 simulated cycles per VAX 8650 CPU second listed is for a 16 machine vector CPU model. 50 cycles/CPU second has been observed for the scalar CPU model.
- MAF 74
  - — In retrospect, an estimate of one to one and a half years of proto-type debug savings would probably have been more accurate.
- MAF 76
  - — A speed-up factor of 4.6 has been measured for the VAX 9000 versus the VAX 8650.