10/11-1-89 JS (1)

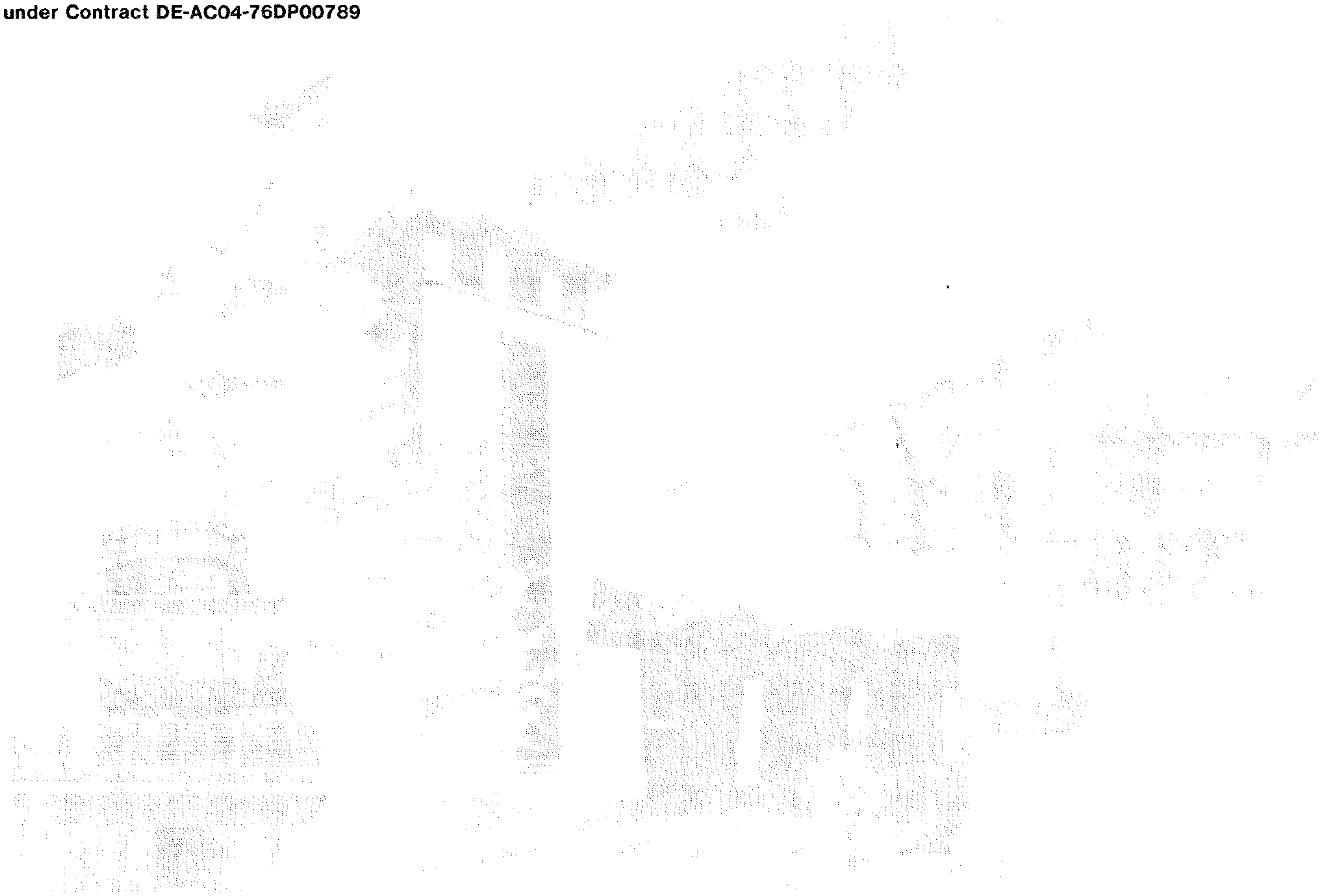
SANDIA REPORT

SAND89—2158 • UC—705
Unlimited Release
Printed July 1989

CMOS ASIC Design Guide

Linda Reuter Nuckolls

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789



SF2900Q(8-81)

SAND--89-2158

DE90 001866

ACKNOWLEDGEMENTS

I would like to thank Wilson Barnard, Joe Hass, Karen Long, Ed Nuckolls and B. D. Shafer for their time, interest, and suggestions.

TABLE OF CONTENTS

TABLE OF CONTENTS [continued]

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

1. SEMICUSTOM INTEGRATED CIRCUITS

In a full custom chip, each transistor is individually sized and layed out to provide optimum speed, power and size. For this reason, the cost of a full custom chip in time and dollars may be impractical, especially if a design is not pushing the capability of the technology. In most cases, an application specific integrated circuit (ASIC) can be implemented adequately and more efficiently as a standard cell or gate array design.

1.1 Standard Cell

A standard cell design is composed from a library of predefined cells. Department 2110 supports several different standard cell families, of various technologies. The types of library cells provided to a designer include logic gates, latches, flipflops, and various MSI functions such as counters, decoders, multiplexers, shift registers, and adders. In most standard cell families, the transistor sizes vary from cell to cell in order to maximize performance, although within the cells the sizes are fixed. A standard cell design does not have a fixed die size; it can be small or large (up to the capability of the technology).

1.2 Gate Array

Gate arrays are a collection of predefined transistors in fixed positions on a die, surrounded by general purpose input/output (i/o). The gate array wafer is processed up through the first layer of metallization (the base array), and later personalized for a specific customer's design in the final steps of processing by definition of its metal interconnect and contact layers.

In general, a gate array is a more cost effective approach than a standard cell design, and provides faster turnaround time. However, the restriction on the number of transistors, the transistor sizes, and interconnect will produce a design that is larger, slower, and uses more power than its standard cell counterpart.

2. RADIATION EFFECTS AND HARDNESS

CMOS devices operating in a space or weapon environment will be exposed to various types of ionizing radiation which can degrade their performance. Department 2110 uses design and processing techniques known as radiation hardening, to minimize the effect of radiation on an integrated circuit.

2.1 Total Dose

As ionizing radiation interacts with a CMOS transistor, electron–hole pairs are generated in the gate oxide (SiO2). The number of such pairs depends on the quantity of energy (total dose) absorbed and the thickness of the oxide, but their effect on transistor behavior is a function of temperature during and after irradiation, the bias on the irradiated devices, and the time after irradiation.

Simply stated, total dose radiation in both n and p channel devices causes a change in threshold voltage due to trapped positive charge in the oxide, and a degradation of mobility in both devices due to scattering from created surface states. Initially, radiation causes n channel device thresholds to decrease, so they are easier to turn on, whereas p channel device thresholds are changed in such a way that they become more difficult to turn on. These threshold shifts along with reduced mobility, are the mechanisms which result in sluggish circuit performance.

The actual *magnitude* of the threshold voltage change is a function of several parameters. In the oxide, most of the free electrons will move quickly toward the gate and be swept out by the gate bias, whereas the less mobile holes move slowly toward the Si/SiO2 interface and become trapped. The threshold change is a function of the position of these trapped holes, which is itself a function of the gate voltage, and temperature. It can be concluded, then, that the worst case bias condition during irradiation is n channels on and p channels off, because this will cause the greatest number of electrons to escape recombination.

To further complicate matters the threshold change is a function of time as well. After bombardment the trapped positive charge in the oxide of n channel transistors may be annealed by electrons which tunnel into the oxide at a rate which is dependent on the bias and temperature. Eventually, electrons will annihilate the trapped charge leaving behind negatively charged surface states such that the device becomes more difficult to turn on. In this case, which is referred to as 'rebound', the threshold voltage of an n transistor can permanently increase to a value that is higher than its pre–radiation value. The p channel effect is not as significant.

Hardening devices against total dose involves gate oxide processing issues such as cleanliness and rate of growth of the oxide. The goal is to eliminate troublesome surface states. A thinner oxide would experience less threshold shift, but would result in increased gate capacitance.

2.2 Gamma Dot

A transient ionizing pulse of radiation, also referred to as 'gamma dot' radiation, creates electron hole pairs throughout an irradiated device. The e–h pairs which are created in the reverse biased p–n junction depletion regions are separated by the field, creating a component of photocurrent which alters the behavior of the device. The current which is created in a reverse biased drain of a logic gate will be in the direction which tends to lower a logic high or raise a logic low. As long as the change in the output voltage during the transient is less than the noise margin, there will not be a logic upset.

Another more significant result of photocurrent is caused by the e–h pairs generated in the largest reversed biased p–n junction -- that of the p well to n substrate. This photocurrent can be of sufficient magnitude to turn on a silicon controlled rectifier configuration existing within the CMOS device, composed of the n channel source/p–well/n–substrate npn device, and p channel source/n–well/p–substrate pnp device. This behavior is referred to as 'latch–up'; it may cause a permanent change in the output voltage and supply current, and may ultimately burn up the device. CMOS devices at the CRM employ an epitaxial substrate, along with frequent p well ties to ground (and some substrate ties to power), which serve to reduce the resistance at the base–emitter junctions of the parasitic bipolar devices, thus decreasing their gain and therefore preventing latch–up.

The p well to n substrate photocurrent mentioned above can create another transient current problem in large chips with long Vdd and Vss busses. Because this component of photocurrent is a shorted path from power to ground, a great deal of additional current flows through the supply busses. These busses have finite resistance, so the excess current causes a reduction in the magnitude of the power supply voltage applied to the cells, which is dependent on the distance of the cell from the supply connection. With weaker supply voltage there is not only a variable reduction in the operating frequency of the cells, but also an increased chance that the cells with the weak sources will flip states in response to the drain component of transient photocurrent. This supply degradation known as "rail span collapse" can be largely corrected by employing wide busses and increasing the number of supply bus 'source connections', i.e., pins, on the chip. The designer should keep this in mind when determining pinout.

2.3 Single Event Upset

If an individual heavy charged particle bombards a device, localized electron–hole pairs are created along the path of the particle through the semiconductor. Consider a CMOS device which is 'hit' on the drain of an off p–channel transistor (this is the worst case). In this event, the e–h pairs in the depletion region will cause a prompt photocurrent which may be sufficient to drive the fanout load to the opposite logic value. In memory elements such as latches, flipflops, and

RAM cells, this type of logic upset can change the stored value. In order to protect against this single event upset (SEU), it is desirable that the n-channel(s) of the 'p hit' device be able to pull off the collected charge faster than the fanout can be charged to its switching threshold. This can be accomplished in a variety of ways, including increasing the operating voltage, increasing the size of the pull-down devices and/or using resistors in the feedback path of memory elements. An 'SEU resistor' serves to slow down regeneration of a stable state, so that the excess charge from an SEU hit will bleed off before the cell can flip.

3. CIRCUIT DESIGN CONSIDERATIONS

3.1 Synchronous Design

A purely synchronous design changes state exclusively upon the assertion of a system clock. It is imperative that a radiation hardened digital design be as synchronous as possible so that transient upsets do not cause an inadvertent change of state. Actually, keeping any design synchronous is good practice regardless of radiation requirements, because it will be easier to test and debug, and will have fewer timing complications than an asynchronous design. Another element of safe design is providing complete circuit initialization; that is, using only latches and flipflops which can be forced to a known state by means of a set or reset line upon system power-up. In addition, any stable state of the device should not be dependent on specific component or wiring delays in the network, i.e. the design should be immune to lot-to-lot process variations. In light of this philosophy, the memory elements in the Sandia library of standard cells and gate array cells are exclusively clocked devices; output changes in response to an input change occur synchronously with a clock. There are, however, some latches and flipflops which have asynchronous set and/or reset lines, which can easily be misused by the designer.

The intent of the asynchronous set and reset feature on memory elements is to provide an immediate state initialization upon power-up, or any other major device reset. Using these lines to force a particular device high or low during normal clocked operation renders a design asynchronous, however, and can lead to unexpected results.

3.1.1 Hazards

Any time two or more inputs to a logic gate are expected to change simultaneously, such that the output of the gate remains constant, you have a candidate for a static hazard. Consider for example, the nand gate of Figure 3-1. In this case input signal A is a function of flipflop 1 and input signal B is a function of flipflop 2, and each of the two flipflops are clocked by a common clock origin. Without taking into account the geography of the chip layout, it would appear that the output of the nand gate would remain fixed at a logic 1.

Suppose, though, that flipflop 2 is placed physically closer to the clock signal and/or the nand gate than flipflop 1. If the difference in the interconnect length of the two nand gate inputs is significant, the additional capacitance in line A could allow signal B to change to a logic 1 before A changes to 0. A static hazard will occur, then, in which the output of the nand gate will temporarily glitch to logic zero, when it should have remained at logic one.

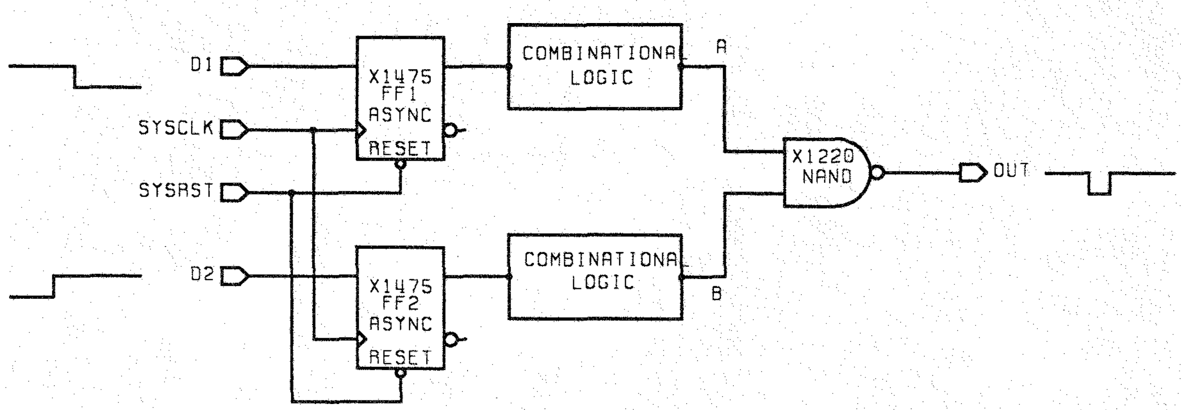


Figure 3–1 Introduction of a Glitch

Obviously, if the combinational logic of A requires logic resulting in more gate delay the combinational logic of B, the arrival time of signal A will lag behind that of signal B due to gate propagation delay as well interconnect delay. Gate delay can be simulated fairly accurately and easily, however, interconnect delay cannot be incorporated in a timing simulation until after the design is laid out; it is preferable not to carry hazard prevention into this stage of the design.

In a latch or flipflop, the clock line(s) and any asynchronous set or reset are considered device 'enables'; that is, a change on these lines can result in a direct change of state. It is possible for a glitch on an enable line due to a static hazard to cause an inadvertent, asynchronous change of state. Alternatively, the data line D is an 'input'; a change on this line should be restricted to occur when the latch is locked. In this manner, the new data can only be reflected at the device output in conjunction with a clock enable, so a glitch on the data line will not destroy the proper state of the circuit.

3.2 Setup, Hold, Race and Lag

Setup, hold, race, and lag times are all terms which serve to identify the timing sequence of input signals which must be obeyed for correct flipflop or latch operation. The definition of these terms can be better understood by referring to figure 3–2, which is a D latch with asynchronous NR (active low reset).

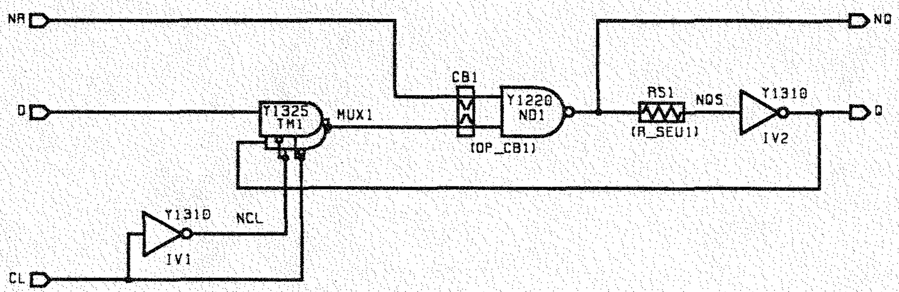


Figure 3–2 D Latch

When the latch is transparent, a change on the data line D must have time to propagate around to the input of the feedback loop transmission gate before the falling clock edge. The setup time is defined as the time for new data to traverse this loop. A latch with SEU protection will have an increased setup time dependent on the RC time constant introduced by the SEU resistor. In the graphs of Figure 3–3, it can be seen that a latch without SEU protection experiences a setup violation on Q1, if the data IN1, changes on the order of 1ns before a falling CLK1 clock edge, whereas the SEU latch setup time is approximately 10 ns.

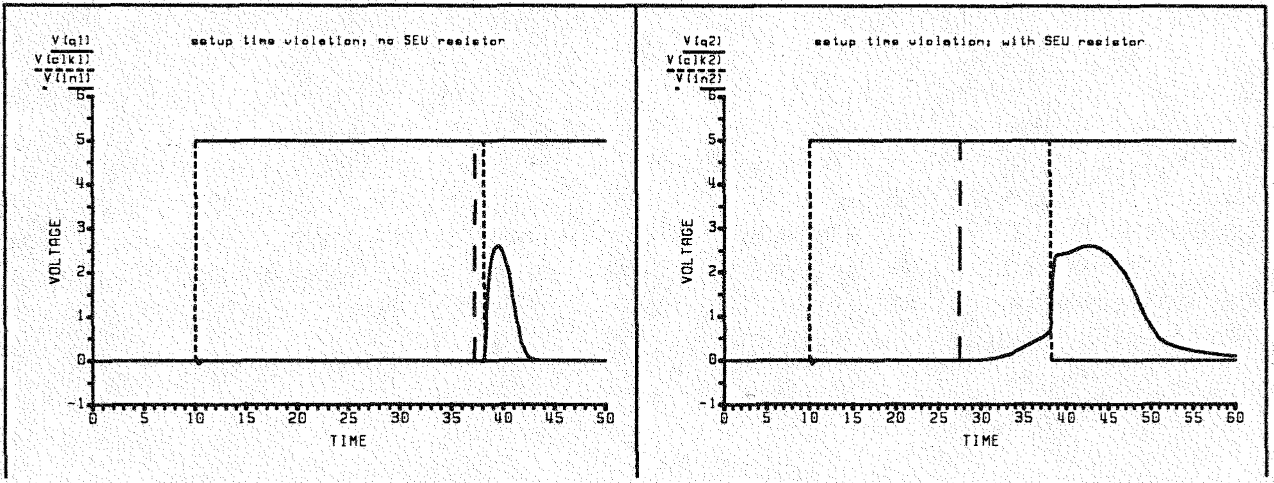


Figure 3–3 Setup Time Violations

The maximum frequency of operation of an IC is influenced by the setup time of the memory elements used; in the case of SEU immune devices it will most likely be the dominant force. Hold time is defined as the time in which data must remain constant after the falling clock edge, and it is effectively zero in the Sandia Library memory devices.

Race time refers to the time that must pass after the reset line is disabled, before a falling clock edge occurs. Violation of the race time requirement can result in a glitch in the output of the latch as shown in Figure 3–4. This particular example involves a latch with an SEU resistor.

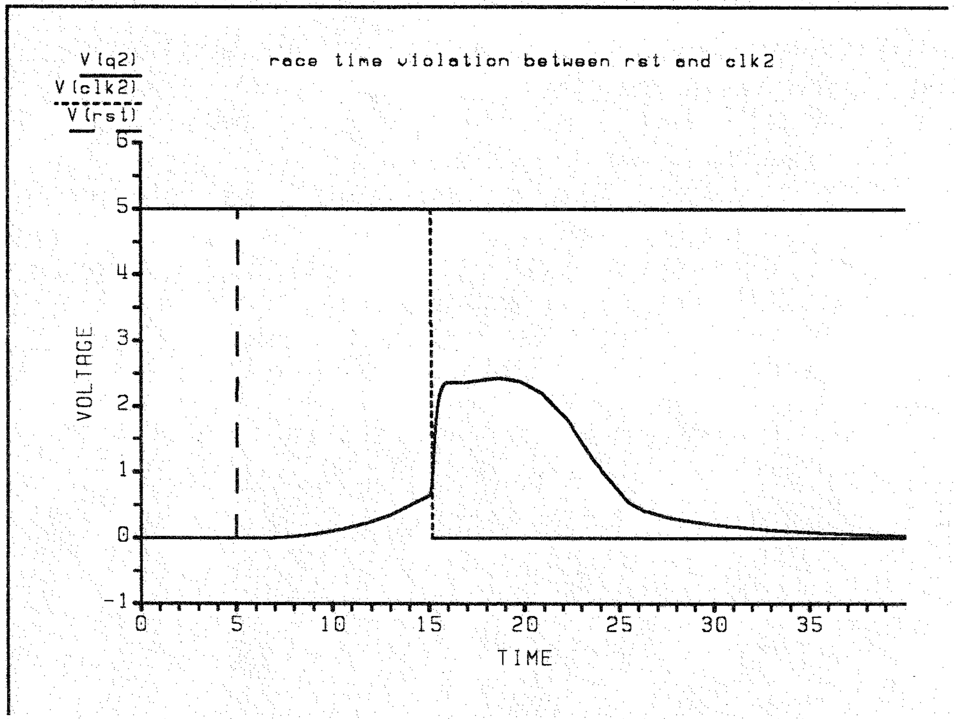


Figure 3–4 Race Time Violation

The data line Q2 is a logic one during the device reset; when the reset is disabled (NR = 1), the data stored in the latch should change from a logic zero to a logic one. The logic one, however, must have time to propagate around to the input of the feedback loop transmission gate before the latch is locked by a falling clock edge. It is seen by comparing Figures 3–3 and 3–4 then, that the race time for the conventional CRM memory elements is approximately equivalent to the defined setup time. If a circuit is to be properly initialized and free of races, the designer must insure that the system reset is synchronized with the system clock. Finally, the lag time for a latch can be described as the maximum time allowed for a clock line to be an unknown logic value. Lag time is an issue if latches or flipflops with external not–clock inputs are used, because they are susceptible to clock skew. In semicustom design, therefore, it is not recommended that these types of memory elements be used. The preferred latches and flipflops have an internal inverter to generate not-clock from clock, with negligible skew.

Setup, hold, race, and lag times as described above apply to master/slave flipflops and two phase flipflops in a similar manner. However, each of the two latches internal to the flipflops will have different effective clocks, so the timing checks must be made in reference to the appropriate clock.

3.3 Buffering

If the driver of a large capacitive load (either a bus, a large gate fanout, or off chip circuitry) is too small, the overall performance of a system may be degraded as a result of reduced speed and increased power consumption. If too large a driver is used for a particular load, power and real-estate can be wasted. In general, a driver should be able to charge its load capacitance as fast as possible while keeping the output rise and fall time of the driver approximately the same as its input rise and fall time. This balance can be maintained throughout a standard cell or gate array design in the 2 micron technology by limiting the fanout of a gate to no more than four p/n transistor pairs. Following this rule of thumb may require that large fanouts or loads be driven by a buffer tree, as illustrated in Figure 3-5.

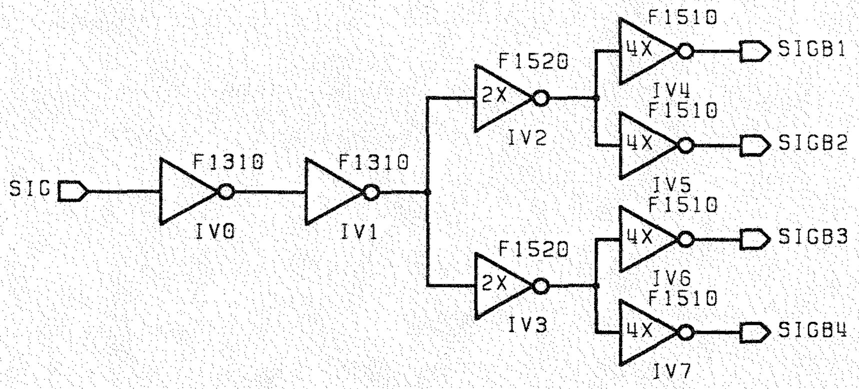


Figure 3-5 Buffer Tree

In a large hierarchical design it is important to keep track of fanout in each block, starting from the bottom up, such that the signals requiring buffering within the block are taken care of locally. As signals cross block boundaries, their loads must be examined globally, to ensure that proper drive strength is attained.

System clock buffering requires a great deal of attention, since this global signal controls the timing in every block of the hierarchy. Remember that each level of buffering a clock origin passes through will add a specific delay to the buffered clock. Therefore, the effective 'local' system clock for each block of hierarchy may be slightly offset. This 'clock skew', which is of most concern when using a master/slave clocking scheme, can also be found within a block of hierarchy due to unequal buffering. Further discussion on clock skew is found in Section 3.4.1, on choosing a clocking scheme.

3.4 Clocking

Ideally, the best possible clocking protocol for both reliability and testability is to have every memory element in a design governed by an externally controllable central 'system' clock. Although practically this may not be possible for every portion of the circuit, it is a primary objective in safe design. In terms of circuit design this translates to the following rules: (1) avoid gating the clock, and (2) avoid gating the asynchronous set/reset lines. Often times it requires more logic and/or creativity to obtain desired behavior by manipulating the data input rather than the clock line or reset line of a particular memory device, but it is a more predictable and testable alternative.

Figure 3–6 gives a simple example of the same logic function obtained by gating the clock, gating the reset, and by gating the data; in Figure 3–7 it is shown how a static hazard on signal A could effect each of the three designs. Only the OUT_C signal is undisturbed.

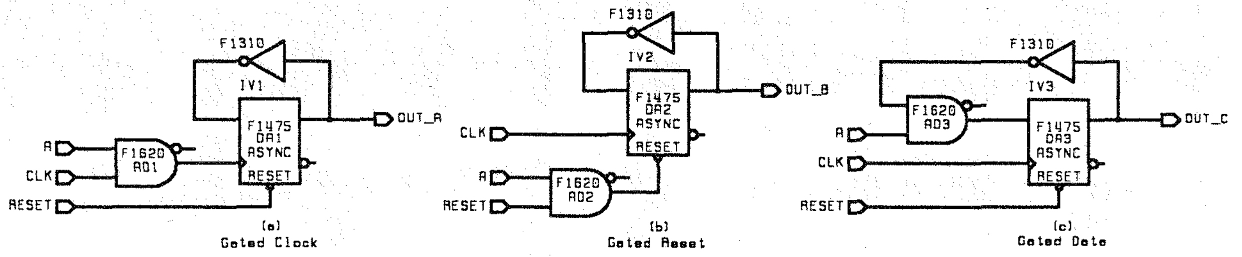


Figure 3–6 Three Implementations of the Same Function

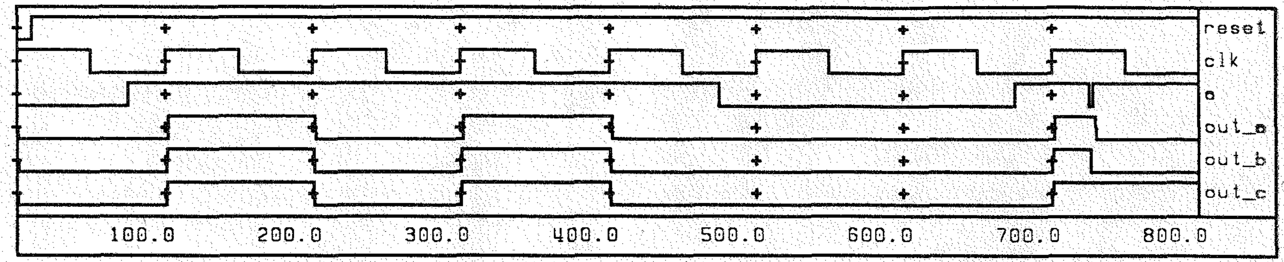


Figure 3–7 Waveforms for Figure 3–6 Circuits

If the designer finds it necessary to gate the clock (or reset) on a memory device, he must insure that these lines are resistant to hazards. This includes hazards originating at the gated enable, and/or glitches which occur on the actual inputs to the logic that is gating the enable. If a hazard is likely, redundant logic should be used to protect the enable; such techniques for hazard prevention can be found in most logic design texts.

3.4.1 Choosing a System Clocking Scheme

The Sandia library of standard cells and gate array cells provides many different memory cell alternatives to the ASIC designer in order to accommodate clocking preference. The flipflop family consists of a set of two phase devices, and a set of master/slave devices for single phase clocking; each family member is a variation on set and reset capability. There is also a family of latches with these set and reset options. The latches and flipflops in the library which have external NCL (not–clock) inputs should be avoided, as mentioned in the discussion on lag time.

The one phase, master/slave clocking scheme is popular because it is conceptually simple. The master latch clock is just the logical inversion of the slave clock, so the master is transparent while the slave is locked, and vice versa. Figure 3–8 is a master/slave flipflop schematic, in which the master clock is CL and the slave clock is NCL.

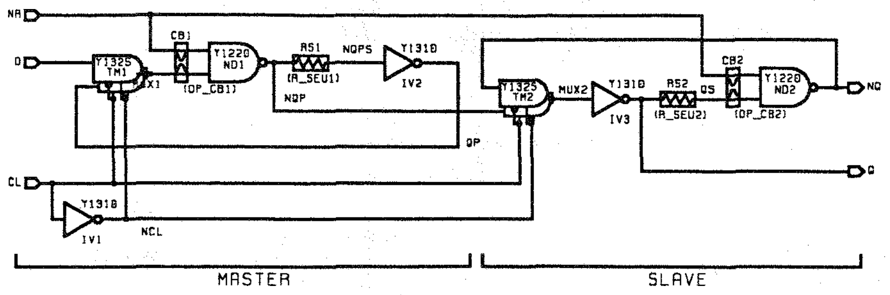


Figure 3–8 Master/Slave Flipflop

Within the confines of a flipflop cell, the timing of data transition from the master into the slave can be considered to be safe and predictable. However, having the 'on/off' clock edges coincident requires the designer to be more aware of differences in clock line propagation delays due to buffering and interconnect (clock skew) for flipflop–to–latch or flipflop–to–flipflop connections. For a demonstration of clock skew, and how it can cause incorrect circuit operation refer to Figure 3–9. In this example, assume that the DA2 flipflop is in a different block of hierarchy than the DA1 flipflop, such that the effective clock to DA2, ECLK2, lags behind that of ECLK1. If this time difference is large enough, DATAIN can flow straight through to DATAOUT in one clock pulse. In Figure 3–10 below, if ECLK2 lags by more than about 2.0ns, it will cause incorrect circuit operation.

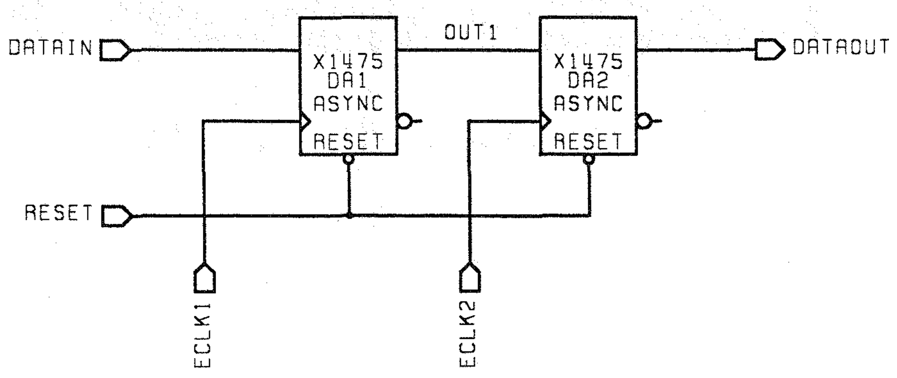


Figure 3–9 Logic to Illustrate Clock Skew

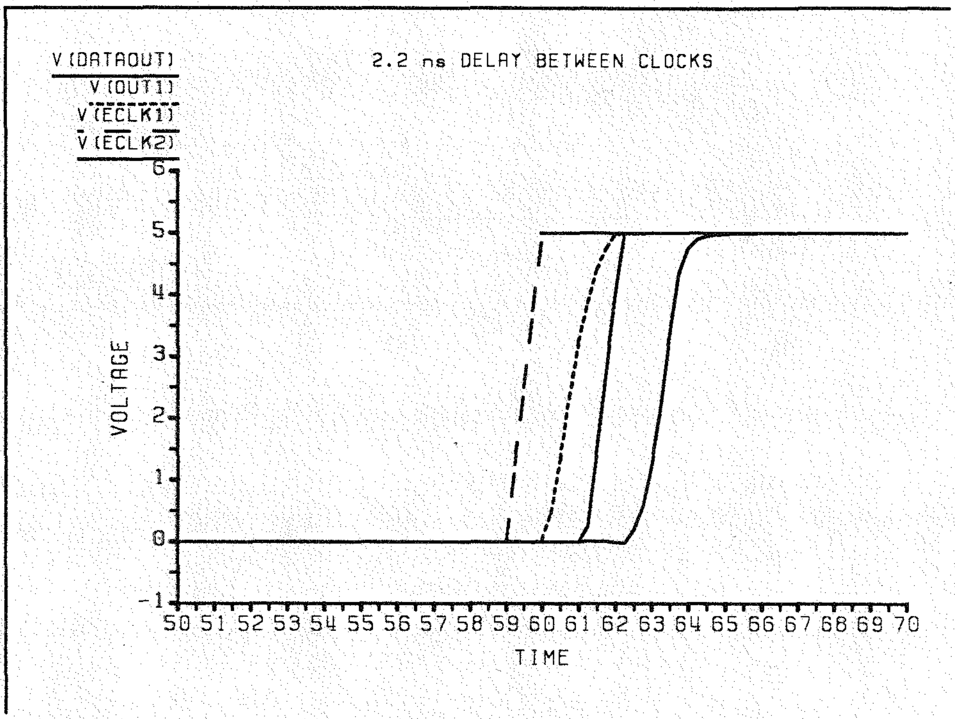


Figure 3–10 Incorrect Circuit Operation Due To Clock Skew

Incidentally, if ECLK1 lags behind ECLK2, there is no problem. This brings attention to a fundamental rule of clock buffering for Master/Slave systems: *the clock should arrive in the direction opposite data flow*. Clock skew can be kept to a minimum if each clock signal experiences balanced loading, and the same number of gate delays. A simple example of proper clock buffering to four blocks of hierarchy is shown in Figure 3–11. The number below the block shows the number of loads the clock must drive within the block.

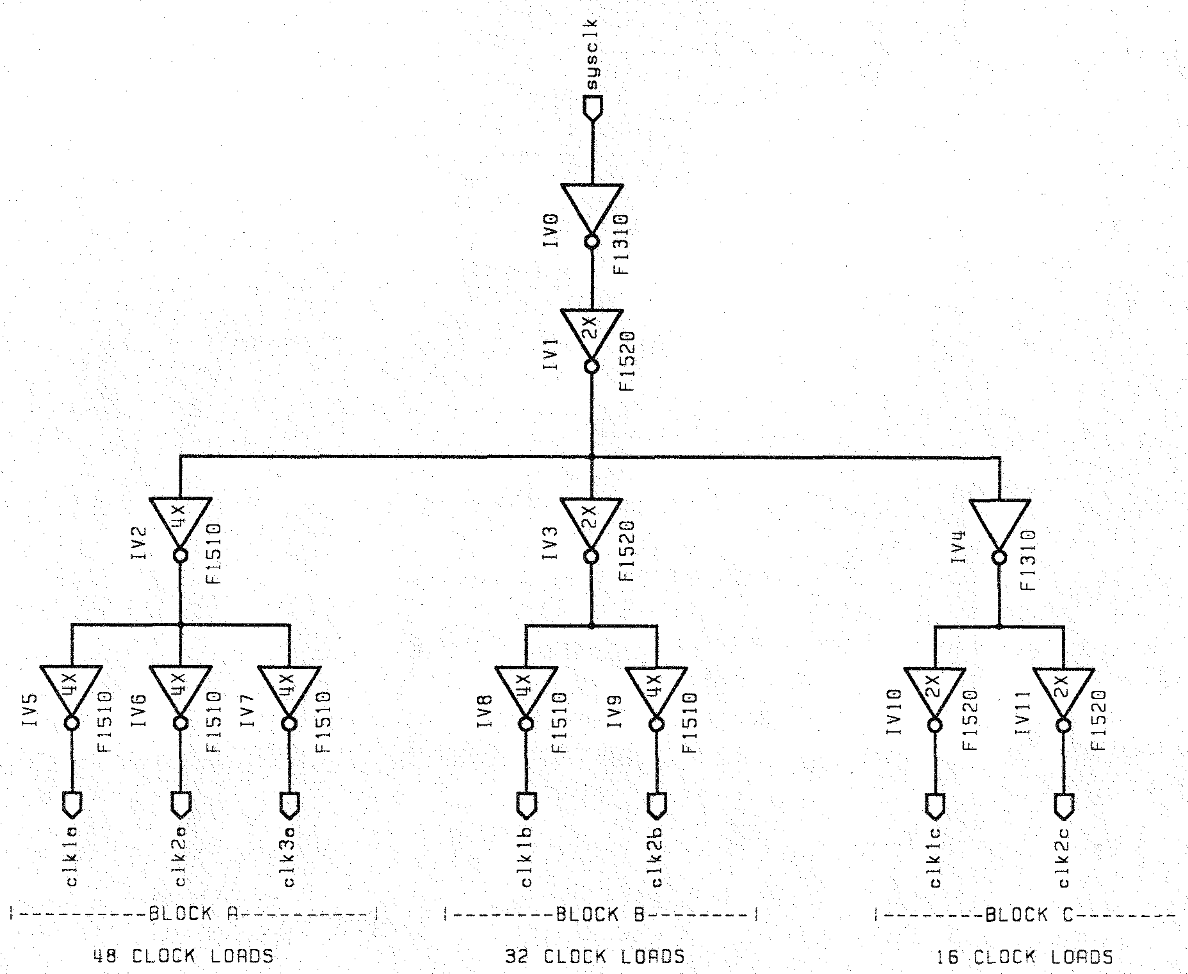


Figure 3–11 Clock Buffering Example

A two phase flipflop allows the use of two non–overlapping central clocks. In this case the phase1 clock edge which opens the first latch in the flipflop can occur after the phase2 edge which closes the latch that it is feeding. Race conditions mentioned above for the master/slave method of clocking are not an issue in this configuration as long as the maximum clock skew present is less than the time between phases. Figure 3–12 shows the relationship of the first latch and second latch clocks for both the master/slave (CL and NCL) and the two phase (PH1 and PH2) flipflops.

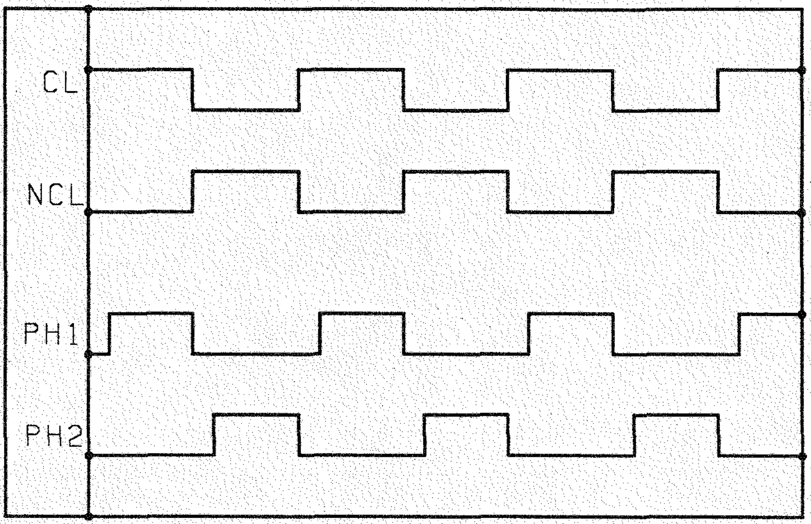


Figure 3–12 Master/Slave and Two Phase Flipflop clocks

If a two phase clock is not an input to the IC, one can be generated on chip with the X1570 library component. By adding inverters in the feedback paths, such as in Figure 3–13, the time between phases can be modified as desired. The designer must verify that sufficient non-overlap exists under all operating conditions.

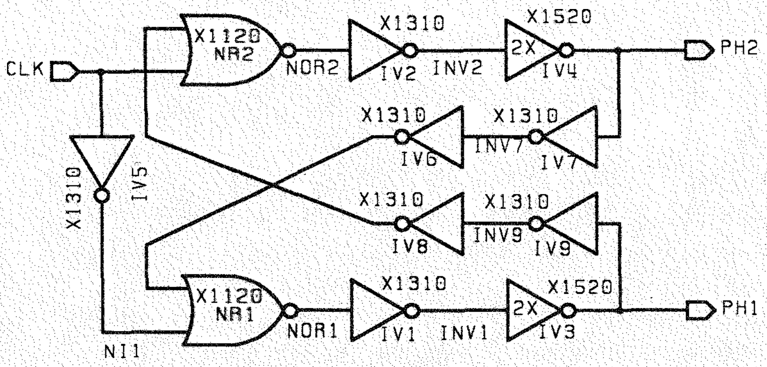


Figure 3–13 Two Phase Clock Generator

3.5 Radiation Design Considerations

Sandia's standard cells and gate array cells are radiation hard due to the nature of certain design and processing characteristics as mentioned in the chapter on radiation effects and hardening. However, in order for an IC to be tolerant to radiation requires more than the use hardened cells; the designer must build a certain amount of immunity into the chip himself.

3.5.1 Total Dose

Because total dose radiation can degrade the threshold voltage and the mobility of both n and p channel devices, it is necessary to allow some margin in the maximum speed of operation of a device. The need for fanout limitation mentioned previously is especially important in a radiation environment in order to maintain a reasonable operating frequency. Simulation with radiation transistor models which imitate total dose degradation will allow prediction of circuit behavior after exposure to the specified radiation dose. Such models are available from Department 2210, as taken from data measured on real devices.

In gate array and most standard cell multi-input logic gates, there exists an imbalance in rise and fall time which increases as the number of inputs to the cell increases. Figure 3-14 demonstrates this property for 2, 3, and 4 input nand and nor gates, compared to a single inverter delay.

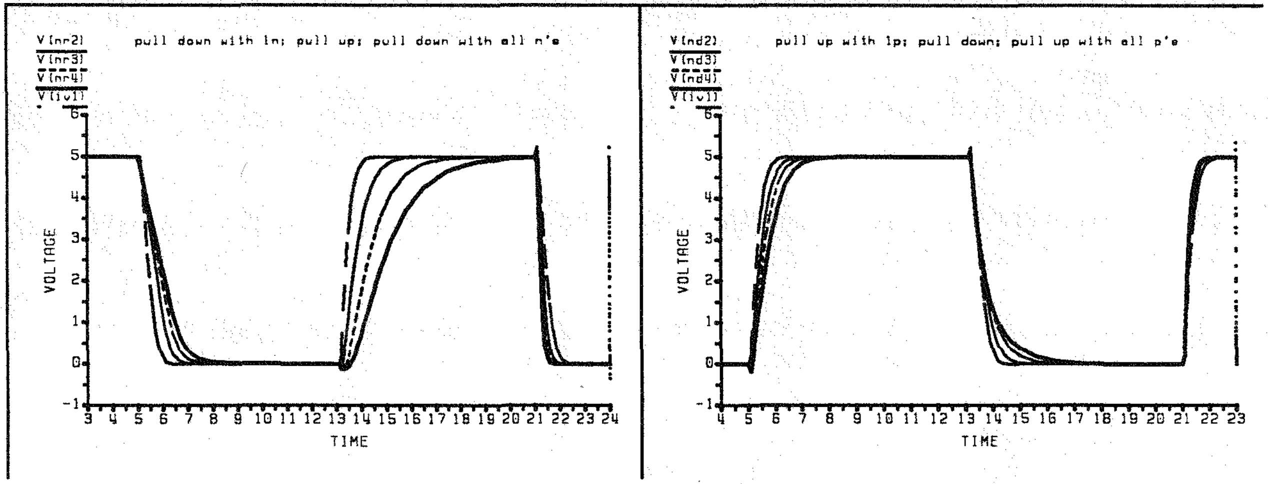


Figure 3-14 Delays for Multi Input Nands and Nors

The cell becomes slow in one direction as a result of the reduced drive of the series connected transistors (actually, this phenomena could be prevented by individually scaling the length to width ratios of each n and p transistor within a multiple input gate, but this sizing detail is not common practice in standard cell families, and not possible in gate array cell design). The degradation of speed is further exaggerated by radiation, and for this reason it is not recommended to use logic gates with more than three inputs.

3.5.2 Single Event Upset

The goal in immunity to single event upset is to prevent an incorrect change of state in the circuit. This can be accomplished in the following manner: by making the memory elements radiation hard, and by avoiding gating of the clock or set/reset. If there is an occurrence of a gated enable, then the 'gating logic' must be SEU immune as well. A hit in any part of the remaining hardware should result only in a slight change in output voltage of the victim cell as long as the voltage transient is less than the noise margin voltage.

'Gating logic' protection from SEU is accomplished in part by using NOR rather than NAND logic, with no more than three inputs per gate. NOR logic is preferred because a NOR gate has its n channels in parallel, which provides a strong pull down capability for the excess charge caused by a hit on a p channel drain. In other words, a NOR provides faster restoration than a NAND on a worst case hit, so the capacitance of the enable line being 'gated' does not get inadvertently charged to the switching threshold.

As discussed in Section 2.3, resistors are used in the feedback paths of latches and flipflops to slow down the regeneration time. Although SEU immunity is achieved in this manner, the resistors greatly increase the setup time of the latch, and therefore decrease the maximum frequency of operation of a design. There are further implications of the resistor which can be understood by referring back to Figure 3–2. Because of the location of the resistor in the latch, the NQ output is significantly faster than the Q output, shown below in Figure 3–15.

It is for this reason, however, that the NQ output is more susceptible to an SEU glitch than the Q output. Even though the state of the latch will resist an upset, the outputs can temporarily waver. The designer should consider the following. Using the Q output in this case would slow down the signal tremendously. However, if the output of this latch is going to some gated enable logic or perhaps a primary output, it would be wise to use NQ instead. Actually, it is very unlikely that a glitch on NQ would be able to clock or reset another SEU protected memory device, since its setup time would be so large, but it is safe design practice to use the less susceptible output. It should be noted that in a master/slave flipflop it is the NQ output that is the slower of the two.

Latches and flipflops can also be hardened to SEU by using large transistors in the latch circuit of the memory elements. These robust transistors will dissipate the excess charge from an SEU hit before the circuit has time to latch the erroneous logic value. The larger memory elements can be used when speed is critical, but there is an area penalty.

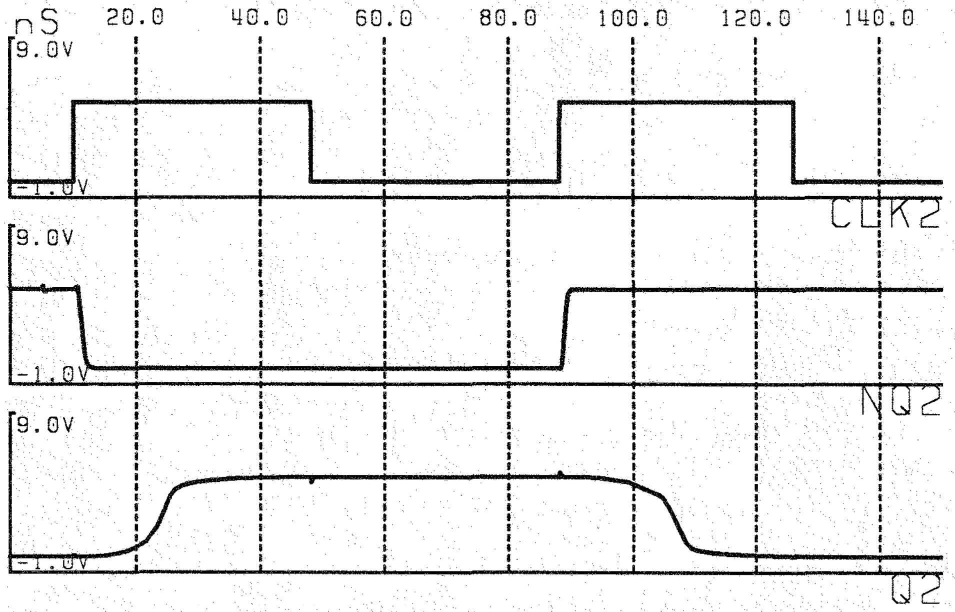


Figure 3–15 Resistor Induced Speed Difference in Q and NQ

The difference in speed of Q and NQ should be considered throughout the design. Obviously both inverted and uninverted outputs of a particular memory element may be needed, so it may be necessary to obtain one output from the other by means of an inverter rather than using both outputs of the cell. Imagine the mess that could be created by having different speed inverted and non-inverted inputs into a state machine decoder. The glitches that would be created at the output of this decoder could create temporary false states which could propagate errors throughout the design.

4. THE DESIGN CYCLE

Presently, when building an ASIC at Sandia, designers must use either a Daisy or Mentor workstation for development of the design. These computers provide the tools to create schematics, to hierarchically simulate (both analog and digital) the design, and to create a layout database for masking. Department 2110 provides cell libraries to Sandia designers, which fully utilize the capabilities of these machines, so as to provide accurate circuit modeling and efficient design layout. A detailed description of the composition of a library is beyond the scope of this document, and is actually transparent to an ASIC designer. The designer must simply choose the appropriate cell library, a decision which will be based on a combination of speed, size, cost, and turnaround requirements. Each technology supported by Department 2110 will have a library database on both the Daisy and

Mentor workstations. Current library choices include the 4/3 Sandia technology, and a 2 micron Harris compatible technology. Two 1.25 micron AT&T compatible libraries will soon be available, one which is optimized for performance, and the other for area. The libraries are differentiated by their cell name prefix, for example, the E1310 is a 4/3 technology inverter, and the H1310 is an inverter from the Harris library of cells. Components with an X prefix are not actually cells, but rather generic unsized representations of a logic device.

Once the ASIC designer has completed schematic capture and simulation to his satisfaction, the design becomes the responsibility of Division 2110, and preparation for fabrication begins. The ASIC is placed and routed with automatic layout tools, as discussed in chapter 6. Then, rule checking is performed to verify that design rules were obeyed, and that the layout matches the schematics which were simulated. A clean design database (netlist) in the required format is then sent by Department 2110 to a mask house, and successful masking is followed by chip fabrication at the appropriate vendor.

Department 2110 uses Mentor tools for internal design, so there is presently a more robust user environment on these machines. Each design application has been enhanced by 2110 personnel with software macros for adaptation into our design cycle. A designer will find the customized features for each tool are found on line in the *Sandia Menu*.

4.1 Circuit Hierarchy and Drawing

In order to make a very large integrated circuit more manageable, it is necessary to break up the design into functional blocks. The object is to form a 'boundary' around a group of logic such that a recognizable function is defined. The top level blocks can be divided into smaller blocks, each of which can be divided further, and so on, until the desired level of simplicity is obtained. The resulting design hierarchy makes a complex design easier to understand, simulate, and layout.

Schematic capture on supported work stations accommodates hierarchical design. A functional block of logic is represented in a drawing by both a symbol and a sheet. The sheet is a schematic consisting of the actual logic connectivity; it will consist of logic gates from a chosen library, and/or user-created symbols. The symbol is a geometric shape created by the designer, to represent its corresponding sheet on the drawing of a higher level block. For example, Figure 4-1 shows the top level schematic of a bit-slice microprocessor which contains three top level blocks represented by appropriate symbols. The sheet for the Memory block, one level of hierarchy below, is shown in Figure 4-2. It contains logic gates, and user-defined symbols that represent additional blocks of hierarchy.

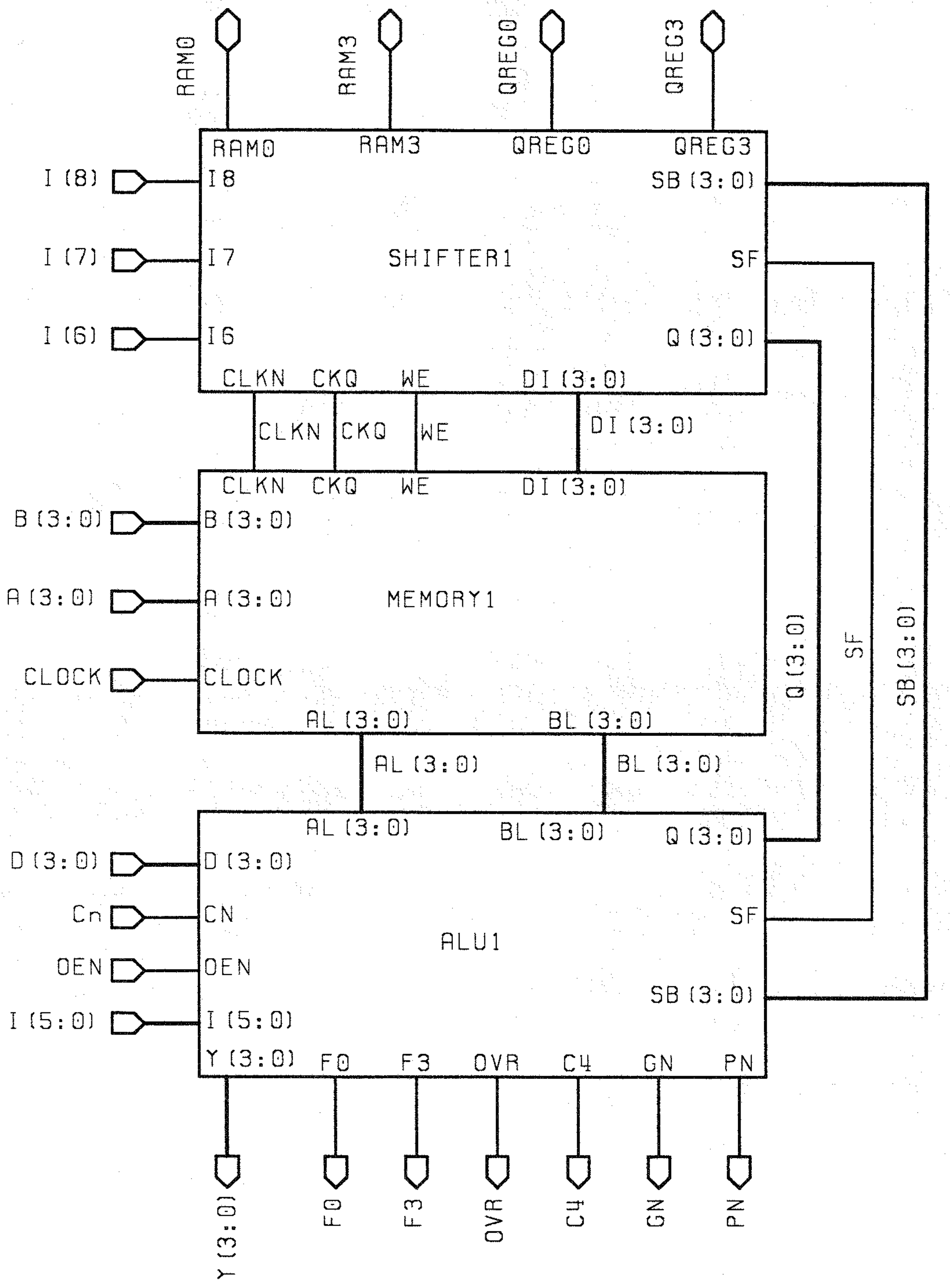


Figure 4-1 Bit Slice Top Level Schematic

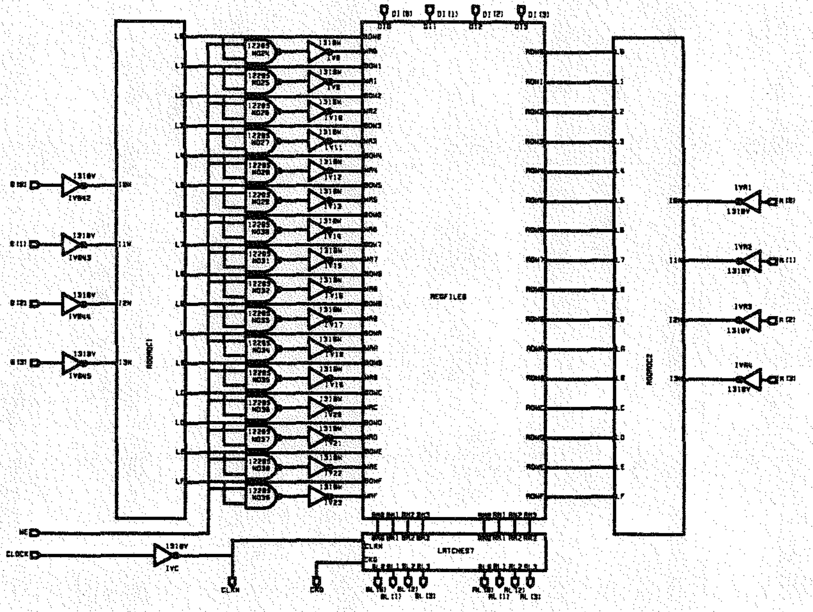


Figure 4–2 Bit Slice Memory Schematic

Although the number of levels of hierarchy in a design is an individual preference, it is handy to keep each schematic small enough as to fit on one sheet. Another point to keep in mind, which greatly simplifies placement and routing: when placing input and output pads in the design, it is preferred that they are either placed alone, in their own logic block, or that they exist around the periphery of the top level schematic.

4.1.1 Schematic Capture on the Mentor

The Mentor schematic capture tool is called *Neted*. Sheet creation in *Neted* begins with the definition of a Sandia sheet boundary, which contains a skeletal format for providing the drawing information necessary for proper archiving. Macros are available for adding specific data to the drawing boundary. Under the 'crm_lib' heading in the *Sandia Menu*, the user will find libraries of standard cells and gate array logic cells in various existing technologies, with which to create his schematic. Successful completion of a schematic produces a directory of the same name, under the current working directory, in which the drawing sheet files are located. It is recommended that each schematic directory be contained under a dedicated design directory.

A user defined symbol is created in *Symed*, which can be conveniently accessed from within *Neted*. Instructions for making a symbol with the properties necessary for various types of simulations are given in the *CRM Library User's Guide, SAND87 - 0491.*

4.1.2 Schematic Capture on the Daisy

Daisy schematics, called drawings, are created in either *ACE* or *DED2*. An appropriate drawing size boundary is chosen, and components are selected from various technology libraries provided by department 2110.

Symbols which represent drawing sheets are categorized as either *blocks* or *components*. A *block* is to be used when a level in hierarchy is being defined. Creation of a block on a schematic produces a directory in which the drawing page files for the block are located. This directory is located one level in the tree structure below the directory of the drawing on which the block is located. Blocks are meant to be used once in the design; if a symbol is to represent logic that will be used many times throughout the design schematics, a *component (*or a 'nested' or 'shared' block*)* should be created. Components and nested/shared blocks do not have an implied hierarchy level as do blocks; their corresponding schematic directories can be located arbitrarily, perhaps in someone else's user area. Properties must be attached to a developed component or nested block which reference its drawing location in the directory tree structure.

4.2 Circuit Compilation

Once a schematic is complete, it is ready for 'compilation' into a design file (in some cases called a netlist). The design file is a database consisting of information about the circuit's connectivity and characteristics, and in the case of a Daisy design file, input stimulus. In most cases a separate design file is needed for each type of simulation to be performed on a particular sheet; every tool uses a different collection of information extracted from the schematic, and the data will be influenced by the environmental conditions specified. In order to simplify design verification and debugging, it is recommended that each block in the hierarchy have its own set of design files so that the blocks may be simulated individually. As the design progresses, compilation time of higher level blocks can be reduced by inserting the appropriate design files of lower level schematics. This approach also minimizes the effect of a design change by keeping it as local as possible. The remainder of section 4.2 will briefly describe design files in both workstation environments.

4.2.1 Mentor

Expand, the Mentor tool for design compilation, reduces the hierarchy of the design into a flattened collection of primitive elements. A cell can function as a 'primitive' if a software model (called a BLM) exists which describes its logical behavior. Each *crm_lib* component symbol is represented by a BLM **and** logic schematic(s), so any cell can be expanded to and simulated at various levels of hierarchy, all the way down to transistor level primitives. The simulation tool for which you are *expanding* dictates which primitive models are appropriate. For example, in simple digital logic simulation, an inverter is considered a primitive instance; a Boolean description of the inverter would suffice. A design file is referred to as 'gate level' if it contains an inverter in this form. If the same inverter was to be used in an analog simulation, *expanding* down to individual n and p transistor models would be necessary. In this case the design file would be at the 'transistor level'. The *Sandia Menu* in *Expand* provides a means to easily choose the appropriate expansion for a given simulation.

Because the compiled schematic necessary for each type of simulation is unique, each design file should be given a different name, indicating the simulation tool, and the radiation and temperature conditions to be simulated. When *Expand* is invoked, the user should choose from the Sandia Menu an item called *Crm_lib Standard Setups*. The entries of interest are setups for *SIM, MSPICE, STAR* or the *Mach1000,* the variety of simulation tools available to the Mentor user. Once the appropriate setup is chosen, the *Device Size Parameters* option under the *Crm_lib Parameters* menu item will incorporate the appropriate cell transistor size data into the design file for the chosen technology library (this step is not needed for a zero delay simulation design file). *Run -List* will execute the compilation.

4.2.2 Daisy

Daisy design compilation occurs in several steps. *DANCE* and *DRINK* are the connectivity compilers; *DANCE* compiles individual pages, and *DRINK* links the compiled pages. Functionality and timing information are added to the compiled design by *SIFT* via the *SPARC* model file input. And, finally, *SOM* inserts input stimulus into the design file.

A Sandia library cell which is placed on a Daisy schematic is only a symbol; presently there are no associated drawings representing descending levels of hierarchy for the cell. The logic function of each cell is represented by a Boolean description entry in a *SPARC* file (provided by Department 2110). Propagation delay values at various loading conditions for each cell are included in the *SPARC* file as well. Because delay is also dependent on operating conditions, there is a family of *SPARC* files for each Sandia cell library.

4.3 Available Simulation Tools

A large selection of simulation tools are available to the Sandia ASIC designer. Digital logic simulation is available on the Daisy, the Mentor, and, by way of 2110's Apollo (Mentor) ring, the Mach1000 hardware accelerator. Each of these machines has its own set(s) of simulation models which are referenced in appropriate design files. *SPICE* code for analog simulation is available locally both on the Daisy and Mentor workstations. Additional analog simulation capability includes *SANCA* (SANdia Circuit Analysis) and *PACSIM*, either of which can be run on department 2110's Alliant via the ethernet (Mentor ring).

Design verification consists of functional, general timing, and critical path analysis, each of which should be done block-wise and hierarchically. As changes are made to the design, the verification phases become iterative. Demonstration of correct functionality is accomplished quickly and easily with zero delay digital simulation, in which propagation through a logic gate is instantaneous. Races or hazards present in the logic will be uncovered in this phase of the design cycle. In order for the simulation to execute, however, it may be necessary to manually insert delays into any feedback paths. For highest accuracy timing analysis one would like to run a transistor level analog simulation of the entire design, but this is not practical or possible. *SPICE* simulation on a schematic in excess of 400 transistors is not recommended; it should be reserved for critical path and diagnostic timing analysis. Methods do exist, however, to incorporate propagation delay values into a digital logic simulation with accuracy adequate for general timing delay analysis. Timing delays for a circuit are dependent not only on transistor sizes and loading, but are also highly dependent on operating temperature and radiation environment.

4.3.1 Digital Logic Simulation

Circuits entered on the Daisy are simulated in the *DLS2* or *MDLS2* application; a designer using a Mentor can run *SIM* locally, or *MACHSIM* on the *Mach1000*. Each of these digital logic simulators requires a properly compiled schematic(s) and a file providing input stimulus (on the Daisy the input stimulus is compiled into the design file).

On the Daisy, simulation with zero delay is specified simply by including a *$TIMING_INFO* entry for zero delay in a schematics' *SOM* file. This will over ride the SPARC file timing delay information selected in *SIFT*. Zero delay simulation on the Mentor can be run on a design file which was *'setup for SIM'* in *Expand*. It should be noted that all of the *crm_lib* components will be represented at the BLM level (as primitives) in this type of expansion.

4.3.1.1 Digital Logic Timing Analysis

After a design has been demonstrated to be functionally correct with zero delay simulation, timing analysis can begin. In a digital simulation, propagation time is obtained from the summation of appropriate rise and fall delays through each logic component in all circuit paths. Digital timing analysis should provide a means for determining the maximum frequency of operation of the part, and locating the critical timing paths in the circuit. Obvious speed problems encountered in timing analysis may require logic changes in the schematic; subtle timing difficulties should be further investigated using analog simulation techniques before proceeding with redesign.

4.3.1.1.1 Mentor

In order to determine proper rise and fall delays for simulation, a new design file must be created in *Expand*, and a program called *STAR* (Sandia Timing Analysis Routine) must be run on this file. Upon completion of *Expand*, the newly created design file (the *STAR* design file) is composed of *STAR* primitives, all of which have zero rise and fall delay values. Running *STAR* inserts into the design file best, nominal, and worst case rise and fall delays for each primitive instance. Delays are calculated based on information in the design file about transistor sizes, gate fanout, and operating voltage, as well as temperature, radiation, and processing parameters provided by the user via the SPICE model specified. If desired, the switching threshold used in best, worst and nominal delay calculations may be changed. It should be noted that interconnect capacitance delay is not presently included by *STAR* for all cell libraries.

Once *STAR* has been run on the *STAR* design file, a digital simulation with propagation delay can be performed using *SIM*. The user may specify in *SIM* whether he wants to simulate with the best, nominal or worst case timing delays. Now, if the user wants to simulate his schematic with timing delays on the *Mach1000* accelerator additional steps must be taken. First, a *Mach1000* design file must be created in *Expand*. Next, a program called *FASTSTAR* must be run, which uses the *Mach1000* and *STAR* design files as input. *FASTSTAR* essentially transfers delay information from the *STAR* design file to the *Mach1000* design file. At the present, *FASTSTAR* is not publicly released, but is available from a 2110 designer upon request.

One other option should be mentioned regarding timing analysis. In the discussion of zero delay simulation above it was mentioned that all of the *crm_lib* components can be represented as primitives; this includes cells such as flipflops and latches. In this case the design file is classified as *multigate*, a name implying more advanced primitives. The utility in this is that timing check flags for each flip flop in the circuit can locate violations such as setup and race conditions. In the *STAR* design file the flipflop is not a primitive, so in order to simulate it as such in timing delay simulation, *STAR* must be used to transfer delays to a multigate level design file. The *STAR* design file, however, will provide the most accurate overall propagation delay information, and should be used for most of the analysis.

4.3.1.1.2 Daisy

Until recently, the calculation of propagation delay through a gate based on transistor sizing and fanout for Sandia library cells was not available to Daisy designers at Sandia. Instead, determination of gate delay was done in two places; first by *SPARC* model selection in *SIFT,* and second by the simulation *mode* chosen in *MDLS/DLS2*. Each *SPARC* file has rise and fall delay value entries for each component, (which were calculated and/or measured at Sandia) for three different loading conditions, at the temperature and radiation level indicated in the file name. In the simulator the user may specify delay values representing either best, worst, or nominal loading conditions.

A Daisy product called *TCAL* is presently being integrated into Sandia's Daisy design kit. *TCAL* makes delay calculations based on gate output capacitance and fanout as specified by custom CRM data. This will provide an increase in the accuracy of timing delay simulations over the present *SPARC* file method. Before *TCAL* can be invoked, the design must be *SIFT*ed with the appropriate *TCAL* library, followed by *SOM*. *TCAL* can then be run to insert delays into the SOM file. Because *TCAL* is currently under cosmetic construction, the details of invocation will not be included here; consult a member of Department 2110 for the latest methodology. As *TCAL* becomes more developed, the user will be able to specify in a configuration file, parameters such as the technology, radiation conditions, temperature, and device equations used in the program to calculate delay, although currently these choices are very limited in scope.

4.3.2 Analog Simulation

Timing problems which are not reconciled in digital timing analysis should be investigated using analog simulation. Because of the database size limitation in analog simulators it is often necessary to copy the troublesome segments from design schematics to a separate, reasonably small drawing, to be used for diagnostics. Keep in mind that any delay path to be examined in the pseudo drawing should be terminated with proper fanout gate loading (effective capacitance and resistance). This is particularly important in determining whether or not a particular output buffer is adequate in driving a formidable off chip load.

There may be situations in which a 'glitch' will appear in the output of a digital logic timing simulation which will not be evident in the analog waveform. In this situation it is likely that the logic simulator is unable to properly differentiate signal strengths. It is tempting to dismiss the perturbation by virtue of a clean analog waveform. The designer should, however, determine the conditions that caused the problem in the digital simulation; it may be that some aspect of circuit operation (such as initialization) has been overlooked.

4.3.2.1 Mentor

Once again, a new design file needs to be created, this time for *MSPICE*, which is the Mentor 'front end' for either *SPICE*, *SANCA*, or *PACSIM*. The primitives which will be simulated in this case are transistor level models. In the *MSPICE* application the user declares the operating temperature and total dose radiation environment by choosing appropriately from a set of transistor models (see section 4.4).

4.3.2.2 Daisy

Analog simulation on a Daisy schematic must be preceded by *DANCE, DRINK*, and *SING*. *SING* is a program which utilizes a directory containing *SPICE* netlists for every cell in each CRM cell library to create a *SPICE* netlist for the compiled design. This *SPICE* netlist is input to *DSPICE*, the Daisy analog simulator. The Master Control File needed to run *SING* in this capacity is another example of Sandia userware for the Daisy which is still in the development stages.

4.4 Choosing Transistor Models for Simulation

4.4.1 Mentor

Environmental operating conditions are incorporated in *STAR* and *PACSIM* or *SANCA* calculations with the choice of proper transistor model files. These files consist of *SPICE* parameters which have been adjusted according to temperature and/or radiation conditions as measured from representative devices. The directory */user/spice_mod* contains a list of transistor model file names with the following format:

gate length.nom.radiation condition.temperature.level

The gate length depends on which cell family is being used, and nom refers to the SPICE data taken from nominal devices. It is preferred that nominal models be used, and 'best' and 'worst' case simulation be derived from *STAR* design file simulations by virtue of changing the switching thresholds.

The radiation condition mentioned above includes both the dosage and the transistor bias conditions during irradiation. The effect of radiation on a transistor depends on the type of device (n or p channel), and the voltage on its gate during bombardment. It would be completely impractical, however, to simulate post-rad circuit behavior in response to radiation during each of the possible circuit states. Instead, post-rad simulations can be executed with models that reflect irradiation with all similar type devices biased identically. Also available are rebound models, which reflect annealed radiation damage as seen in non-weapon space applications. A design should be simulated with appropriate pre and post rad models from both ends of given operating temperature extremes, as well as room temperature. Although high temperature and high radiation dosage will degrade the performance of the transistors in a circuit, the value of any SEU resistor in the circuit is actually inversely proportional to temperature, which is beneficial to circuit speed.

Level refers to the specific algorithms used by *SPICE* to solve for current and voltage. Level 4 models are Sandia written, and intended to be used only with SANCA, whereas level 11 models are preferred for use with *PACSIM*.

update_information file in the */user/spice_mod* directory contains a summary of current simulation model choices.

4.4.2 Daisy

The choice of environmental operating conditions for a Daisy schematic simulation is made by indication of the appropriate *SPARC* model file when running *SIFT*. A library of *SPARC* models provided by Department 2110 should be listed under the *$SIFT_CONFIGURATION_REF$* section in the */PROJECT/PROFILE* file. The first SPARC file listed will be the default; other files may be specified with the *–mod* switch in *SIFT*. The names of each entry in the *SPARC* file library indicate the radiation level, temperature (hot, nominal, or cold), and the supply voltage, which determined the propagation delays for each component listed in the file.

5. FAULT ANALYSIS AND DESIGN FOR TEST

5.1 Fault Simulation

There are numerous types of defects that can occur in an integrated circuit which modify predicted behavior. The purpose of fault simulation is to exercise a circuit in such a manner that the presence of any defect in a circuit can be identified and located. Presently, the only tools available for fault simulation utilize the stuck–at fault model. In this case any defect in the circuit is assumed to manifest itself in such a way that the input or output of a particular logic gate is 'stuck' at one logic value and cannot be changed. A test vector to catch any one fault has to exercise the fault, and sensitize a path to a primary output. In some cases this requires a sequence of test vectors. Simply stated, when a fault simulation is run, a fault is inserted at a node in the circuit and the circuit is simulated with the input test vectors provided. The 'good' circuit outputs are then compared to the 'faulty' circuit outputs; if they differ, then the inserted fault was detected. This type of simulation/comparison is continued for every node in the circuit. Obviously there are more 'nodes' in a transistor level design file than there are in a gate level design file.

The stuck–at fault model is not necessarily a good one; although there are defects which manifest themselves as stuck–ats, the majority do not. The stuck–at model is widely used, however, because of its computational feasibility. In addition, the input vector set which gives high stuck–at fault coverage fully exercises the circuit, so that parametric measurements (such as quiescent power supply current) which identify other types of defects can be made with each vector. Composition of this robust vector set is the goal of performing fault simulation.

The Mach1000 fault simulator is available for the designer on the Mentor workstations, whereas the Daisy users would run a DFS or MCFS simulation. Before attempting to fault grade a VLSI circuit, the designer should understand what a stuck at fault is, and experiment with generation of a vector(s) to control and observe a fault in a simple circuit. Many texts are available on this subject.

5.2 Interpreting Results

After running a set of input vectors through fault simulation, the results of the simulation should be examined to determine which portions of the chip have not been thoroughly tested. Both the Daisy and Mach1000 fault simulation output will identify the node and sa0/sa1 fault status. However, looking at each individual fault and determining a test for that fault can quickly become a tedious, overwhelming job. It is best to step back, and examine relative fault coverage on a block-by-block basis by traversing the hierarchy of the design from the top down. In this manner, blocks which have particularly low coverage can be identified, and the input test set can be enhanced to more completely exercise the block, rather addressing each individual fault one at a time. *Fault_Analyze* is a Sandia written program which enables block-wise fault analysis for fault simulations done on the Mach1000, by traversal through the hierarchy of the design. Presently, there is not a comparable tool for users of the Daisy fault simulator.

5.3 Increasing Testability

The intent in designing increased testability into a chip is to provide higher fault coverage, as well as a decrease in the effort and time to create and apply test vectors for design verification. Of course, incorporating design for test (DFT) techniques into the design has a price -- increased chip area, power consumption, and package pinout, and possible decreased speed.

There are many structured approaches to designing for testability, including providing on-board self-test capability. However, the method which is most easily implemented and conceptually simple is the scan path technique. The scan path method of DFT requires that each memory element in the design is scannable, and strung together in a shift register configuration to increase controllability and observability. With reference to the devices in Figure 5-1, assertion of the Scan/Normal select pin T will set the register in test mode, and the scan clock SCL will propagate the state of the circuit, one bit at a time, to a primary output to be observed. When T is low, the circuit behaves normally, with its appropriate clock. This configuration allows the designer to force the circuit to any state using test mode, run the circuit in normal mode, and then scan any desired intermediate states to an external pin. Not only can a circuit enigma be uncovered in this manner, but it can also be located. This includes both processing defects and functional design errors.

Presently the Sandia library has one scannable cell, which is a master/slave D flip–flop with a built in data and clock mux. This particular cell, the X1476, is a direct replacement for the X1475; that is, the designer can incorporate this flip–flop into his design without having to deal with multiplexing his clock or data, or forcing his clock to a certain state before or after going into test mode.

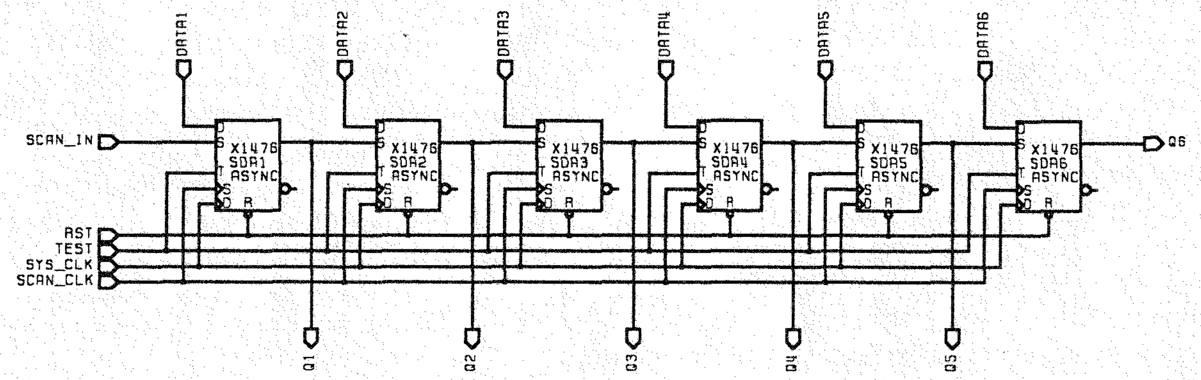


Figure 5–1 Example Scan Chain

Figure 5–1 is an example of a generic six flipflop design hooked up for scan capability. The reset inputs of the scan flipflops *must not* be controlled by the outputs of other latches.
It should be noted that there is a required input sequence to take the scannable elements in and out of test mode, so as to avoid 'false clocks' that will give unpredicted results. A schematic and input sequence rules for the X1476 is given in appendix A.

Because there are always numerous structures on a chip which can be easily scanned because of their connectivity and/or clocking (counters, shift–registers, etc.), a multiplexed data *and* clock line in every flip–flop cell may not be necessary, and would be a waste of chip area. For this reason, an external flipflop mux(s) which muxes just the clock, or just the data, can be used as the front end to an *appropriate* chain of regular flip–flops, and provide the same degree of testability as a chain of X1476's.

6. CIRCUIT LAYOUT

When an engineer is confident that he has a working design, the layout phase begins. A netlist of the appropriate format from the Mentor or Daisy can be used, although it is likely that a Daisy design will be converted to a Mentor compatible format to be layed out by Department 2210. A brief overview of what happens after the design phase is discussed.

A layout of each of the library cells already exits, so what remains to be done is to optimally place these cells in regularly spaced rows and route them together. There are semi–automatic tools which 2110 uses to accomplish these tasks, however, an estimate of the chip area must be made before actual layout begins. Ideally, the entire flat netlist could be input to the automatic place and route tools with no additional effort. Unfortunately, this is only possible with small chips; the majority of designs have to be broken up into smaller sections, each of which are individually placed and routed, and then these sections are routed together. A designer should be aware of the capability of the placement and routing tools, so that he can plan his schematic hierarchy accordingly. It is convenient to have the top level schematic composed of blocks which contain a number of components that can be auto placed and routed in a tolerable amount of time. In this manner, the designer can indicate the desired physical location of his blocks based on critical paths and i/o, and the layout engineer can determine the aspect ratio of each of the blocks based on cell dimensions and routing area approximation, such that the chip takes on a reasonable shape. An example of this floorplanning approach is given in Figure 6–1.

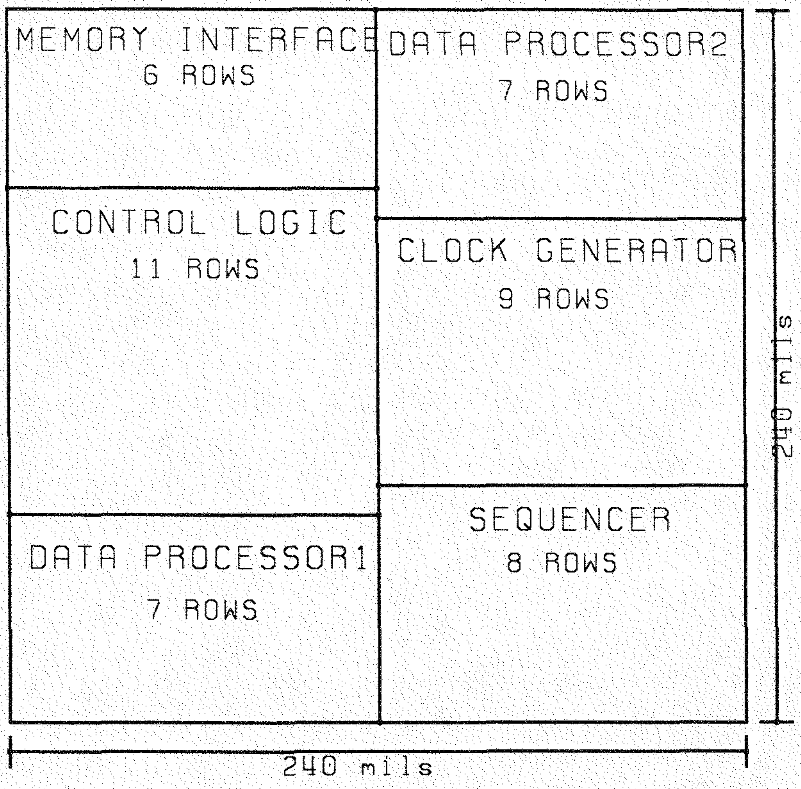
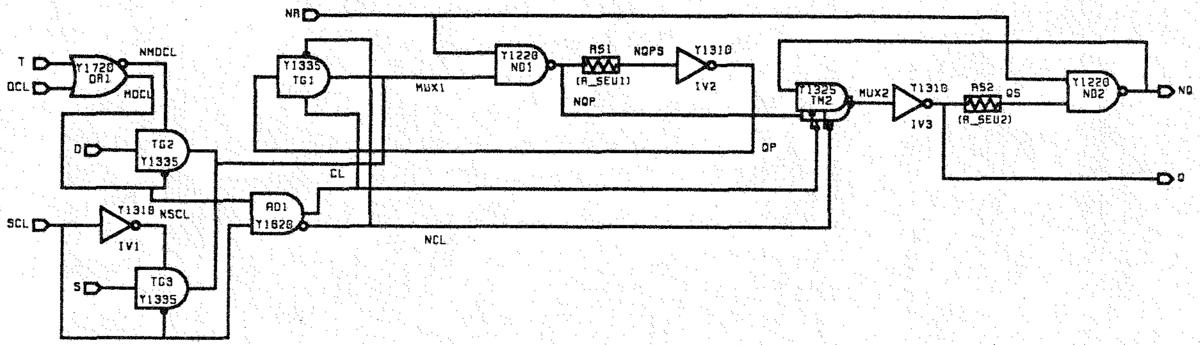


Figure 6–1 Example Floorplan

APPENDIX A



Scannable D Flipflop

X1476 USAGE REQUIREMENTS --

test mode:

T = 1
SCL active

normal mode:

T = 0
SCL = 1

MODE CHANGE SIGNAL ORDERING:

test mode --> normal mode

T --> 0
SCL --> 1 (leave SCL high after last rising edge)

normal mode --> test mode

SCL --> 0
T --> 1

DISTRIBUTION:

2110	R. E. Bair
2113	B. D. Shafer (25)
2113	Staff (11)
2114	M. G. Knoll
2114	Staff (9)
2115	F. W. Hewlett
2115	Staff (6)
2116	W. J. Barnard
2116	Staff (8)
2340	M. W. Callahan
2345	B. C. Walker
2345	L. R. Nuckolls (5)
3141	S. A. Landenberger (5)
3141-1	C. Ward (8) for DOE/OSTI
3151	W. I. Klein (3)
8524	J. A. Wackerly