# Fully Automated Radiation Hardened by Design

# Circuit Construction

by

Nathan Hindman

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2012 by the
Graduate Supervisory Committee:

Lawrence Clark, Chair
Keith Holbert
Hugh Barnaby
David Allee

ARIZONA STATE UNIVERSITY

December 2012

## ABSTRACT

A fully automated logic design methodology for radiation hardened by design (RHBD) high speed logic using fine grained triple modular redundancy (TMR) is presented. The hardening techniques used in the cell library are described and evaluated, with a focus on both layout techniques that mitigate total ionizing dose (TID) and latchup issues and flip-flop designs that mitigate single event transient (SET) and single event upset (SEU) issues. The base TMR self-correcting master-slave flip-flop is described and compared to more traditional hardening techniques. Additional refinements are presented, including testability features that disable the self-correction to allow detection of manufacturing defects. The circuit approach is validated for hardness using both heavy ion and proton broad beam testing. For synthesis and auto place and route, the methodology and circuits leverage commercial logic design automation tools. These tools are glued together with custom CAD tools designed to enable easy conversion of standard single redundant hardware description language (HDL) files into hardened TMR circuitry. The flow allows hardening of any synthesizable logic at clock frequencies comparable to unhardened designs and supports standard low-power techniques, e.g. clock gating and supply voltage scaling.

ACKNOWLEDGMENTS

I’d like to thank my committee chair, Lawrence T. Clark, for his support and guidance throughout this research effort.

I’d also like to thank my co-workers, Dan Patterson, Satendra Maurya, Srivatsan Chellappa, Chandarasekaran Ramamurthy, Sandeep Shambhulingaiah and David Pettit for their work on this and related projects.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

# CHAPTER 1

# INTRODUCTION

Protecting high performance integrated circuits (ICs) from ionizing radiation-induced upset is a key issue in the design of microcircuits for spacecraft, which must function in a high radiation environment [1]. A radiation-induced error occurs in a semiconductor device when a high-energy particle travels through the chip, producing an ionized track. The resulting collected charge may cause a transient voltage glitch, i.e., a single event transient (SET), or flip a bistable storage cell to the opposite state, i.e., a single event upset (SEU). Radiation Hardening By Design (RHBD) promises to allow circuits hardened against these errors to run at commercial circuit speeds by using state-of-the-art foundries. However, many traditional RHBD techniques significantly affect the circuit speed. For instance, hardened microprocessor frequencies have been below 200 MHz, while commercial embedded designs, albeit on more advanced fabrication processes, reach over 1 GHz [2-3].

In this dissertation, RHBD sequential circuits and an automated computer-aided design (CAD) flow are presented that implement self-correcting soft-error hardened circuits using triple mode redundant (TMR) logic. The TMR circuits are based on a flip-flop that has been experimentally proven to be hard in both proton and heavy ion testing on the standard version of the IBM 90 nm bulk CMOS technology, and with ion testing on the low standby power version of the process. This flip-flop uses voting circuits to correct only the slave feedback path, which results in a significant advantage in speed over traditional designs. Since this design is easily compatible with clock gating and power scaling techniques, it can also be used in circuits where low power consumption is a critical design constraint.

## 1.1 FLIP-FLOP DESIGN CONSIDERATIONS

Digital circuits are generally pipelined finite state machines comprised of combinational and sequential circuits. The combinational logic operates on the input and circuit state signals to generate the next state and output signals, while the sequential circuits provide synchronization from one state to the next under control of the system clock. The clock rate is determined by the sum of the delays through the combinational and sequential circuits. All logic functions must complete their operations between controlling clock edges. A typical pipeline stage operation consists of a clock rising edge, followed by the master-slave flip-flop (MSFF) outputs (Q) transitioning after delay $t_{CLK2Q}$, whereupon the combinational logic operates and generates outputs that are sampled by the next pipeline stage flip-flop. The maximum path delay through the combinational logic added to $t_{CLK2Q}$ and the subsequent flip-flop setup time $t_{SETUP}$ determines the shortest clock cycle.

Microprocessor designers typically estimate logic speed using a metric of how many NAND2 gates driving a fanout of 4 load can fit in a clock cycle, referred to as FO4 per cycle. The embedded XScale microprocessor in a 180 nm technology averaged 27 FO4 per cycle, and the 90 nm version less than 24 [3]. The 1 GHz DEC Alpha was estimated to have 14-16 FO4 per clock cycle [4]. A primary difficulty in scaling clock frequencies, besides wire delays, is that timing overhead of master slave flip-flops (MSFFs) and latches, as well as clock skew, use increasing portions of this delay. As an extreme example, the Pentium-4 low voltage swing differential logic blocks were designed with integrated sensing latches, since otherwise only two gate delays were left for logic gates in each 7 GHz clock cycle [5]. Therefore, reducing the $t_{CLK2Q}$ and $T_{SETUP}$ increases the overall frequency at which the circuit can operate.

Traditional temporal hardening techniques add double the $t_{SET}$ to the $t_{SETUP}$ time, and can dramatically slow the circuit. At higher energies, SET durations have been experimentally measured to exceed 1 ns [6-7], which dramatically degrades the performance of circuits hardened for these energies. In contrast, the TMR flip-flop used here was designed specifically to keep these timings short while maintaining hardness, and can therefore offer timings that are almost identical to unhardened master slave flip flops.

Modern ICs are also increasingly power-constrained, particularly for aerospace applications. When using clock gating, the most prevalent technique for limiting logic power consumption [8], activity factors can be as low as 10% to 20% in microprocessors [3,9]. However, many traditional RHBD flip-flop designs are not hardened to clock SETs, which means that the clock nodes need to be immune [10]. The only way to ensure this immunity is to increase the size of the clock nodes, so that the capacitance is large enough that an SET cannot move the voltage past the transition point. These large clock nodes cannot be effectively clock gated. In contrast, the TMR flip flop design used for this project relies on three separate flip flops, each having its own separate clock. Thus, it can correct clock SETs by outvoting the affected flip flop with the two flip flops that have unaffected clocks. This allows it to take full advantage of the power benefits of clock gating.

Since IC power scales with the square of the supply voltage $V_{DD}$, supply voltage scaling is also commonly used to reduce power in commercial ICs. However, this also increases the delay in all circuits. Traditional temporal hardening techniques suffer more than most from this increased delay, because it relies on delay elements and relatively short SET durations to achieve reasonable speeds. When power is reduced, the delay of both the delay elements and the SETs themselves increases rapidly, making the overall circuit frequency degrade. Since TMR flip flops rely on spatial separation instead of delays, power reductions only affect it as much as they would a standard flip flop, and power scaling becomes much more effective.

## 1.2 TMR WORKFLOW DESIGN

TMR hardening with commercial synthesis and APR tools is a difficult task due to the random nature of how logic cells are placed. When an ion strikes the chip, the charge collection can affect an area with multiple circuit nodes. If two or more redundant nodes are in this area, they can both be affected and the redundancy is ineffective. Thus, the critical nodes of redundant circuits must be sufficiently separated so that one ionizing track cannot affect multiple redundant circuit nodes. Since TMR hardening relies on spatial separation to ensure that the triplicated logic blocks are isolated from each other, each of the copies needs to be placed in its own separate section of the chip. However, these different copies of the logic need to interact with each other in order to vote out incorrect data. Thus, if they are too far apart, the wires needed to perform this voting quickly fill the available metal tracks and make the circuit unroutable. In the past, these conflicting design constraints have limited most TMR hardening to hand placement and/or relatively small blocks of logic.

However, the workflow introduced in this paper has no such limitation [11]. It uses the high speed flip-flop discussed above and uses commercial CAD tools to perform most of the tasks, but it shapes the logic into interleaved stripes that adjacent yet spatially separate. Voting is handled by flip-flops that are a fixed, relatively close distance to each other, while still maintaining a gap between separate copies to ensure hardness. Since the TMR flip-flop discussed above is used, it takes advantage of its performance characteristics to produce high speed, low power TMR hardened circuits.

This workflow relies on commercial CAD tools and glues them together with custom CAD software to handle triplicating the logic and to ensure that the tools work together properly. Though the current implementation uses a specific set of commercial tools, the glue logic only modifies the tool's save files, and thus does not directly interact with them. This approach makes it easy to adapt for use with other sets of commercial CAD tools. A simplified block diagram of the workflow is shown in Fig. 1.1 at the end of this chapter. This is divided into two major parts: the "Library Design and Characterization" block that must be done to integrate the library with the workflow and the "Block Design" flow used to design each block once the library itself is characterized.

## 1.3 DISSERTATION ORGANIZATION

The dissertation is organized as follows: The introduction and discussion of the motivation behind the paper was discussed in chapter 1. Chapter 2 discusses the mechanisms and effects of radiation strikes, along with tests done to estimate the hardness of the spatial separation used in this workflow. This background information is used as the basis for the hardening process, and informs the library design. Chapter 3 shows how these radiation effects are mitigated with appropriate cell library design and includes a discussion of the flip-flop design used in this workflow. This chapter then concludes with the abstraction and simulation processes that complete the “Library Design and Characterization” section in Fig. 1.1. Chapter 4 details the workflow, with a focus on the custom tools that glue each of the commercial tools together. It discusses in depth the features and details of the “Block Design” section in Fig. 1.1. Chapter 5 looks at how TMR blocks created with this workflow interact with the outside world, using a processor that was designed with this workflow as an example. Although these details are not directly part of the workflow, they must be considered in order to ensure proper integration with surrounding circuits. Chapter 6 concludes this work and discusses the lessons learned in creating the example chip. It also discusses some tool and workflow limitations that might be worthy of future refinement.

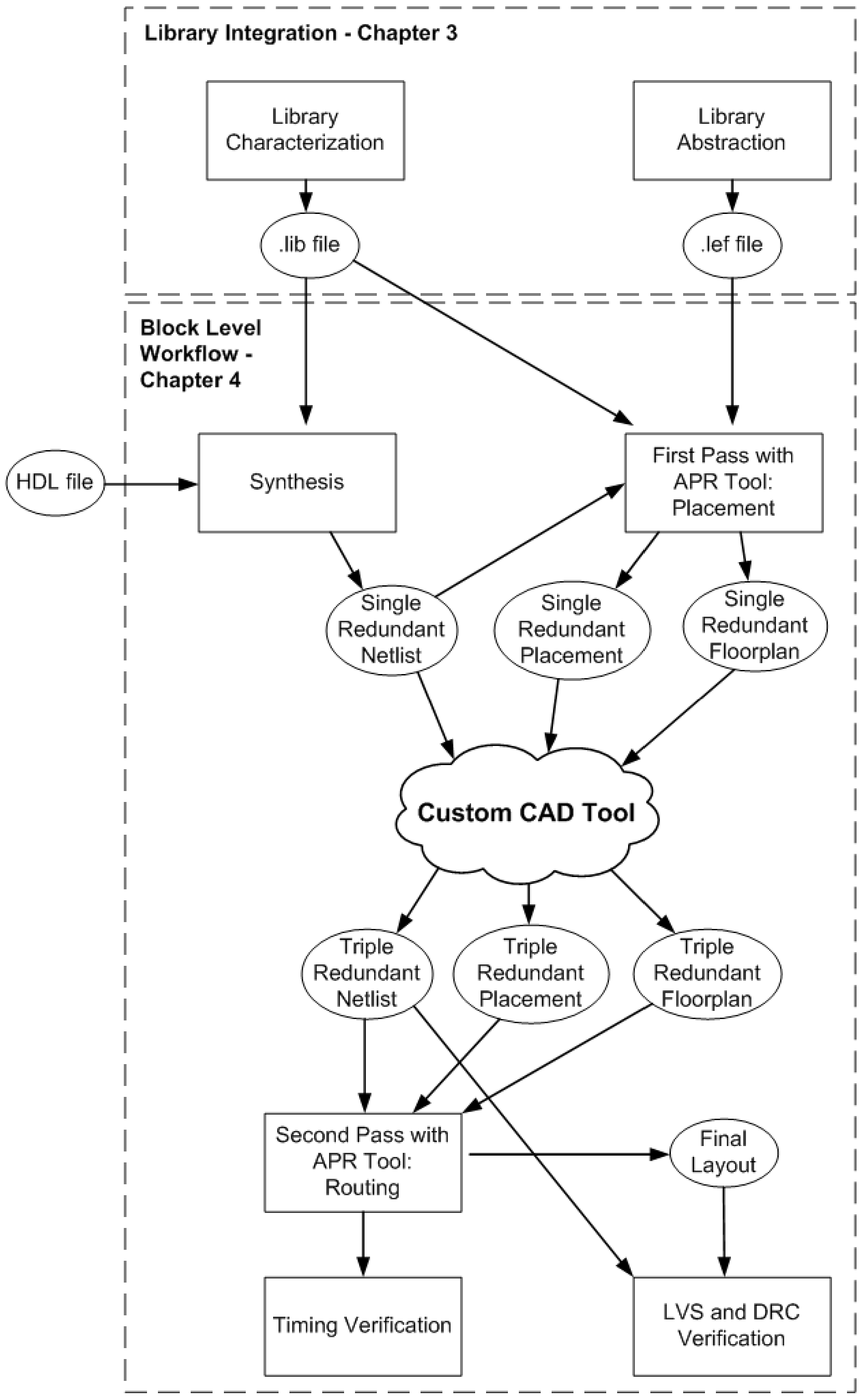

Library Integration - Chapter 3
Library Characterization
Library Abstraction
.lib file
.lef file
Block Level Workflow - Chapter 4
HDL file
Synthesis
First Pass with APR Tool: Placement
Single Redundant Netlist
Single Redundant Placement
Single Redundant Floorplan
Custom CAD Tool
Triple Redundant Netlist
Triple Redundant Placement
Triple Redundant Floorplan
Second Pass with APR Tool: Routing
Final Layout
Timing Verification
LVS and DRC Verification

Fig. 1.1. The basic block diagram for the workflow described in this paper. The top section deals with integrating the library into the workflow, and will be discussed in Chapter 3. The bottom section deals with the workflow steps that convert a block from RTL into a completed layout and is discussed in Chapter 4.

# CHAPTER 2

# RADIATION EFFECTS

Although radiation can affect circuits through several different mechanisms, the major effects can be categorized into Total Ionizing Dose (TID) effects and Single Event Effects (SEEs). TID is the result of cumulative damage or charge collection as radiation bombards the chip [1,12-23], while SEEs are the result of the passage of a single particle through the layers of silicon on the chip [6-7,24-32]. The magnitude and type of these effects is highly dependent on fabrication processes and the circuit design, but is also dependent on the radiation environment. Since there is a minimum threshold energy for SEE effects, these only occur with higher energy particles, which are quite rare in most environments. TID effects, on the other hand, often depend on the total cumulative energy of all particles that strike an area, so the quantity of lower energy particles can make up for their individually lower deposited charge.

Particles that deposit extremely high amounts of energy can cause physical damage to the chip and thus prevent a component on the chip from ever functioning correctly. These “hard errors” are typically only preventable with process modification and a radiation hardened process. Thus, radiation hardened by design (RHBD) methods typically focus on medium to high energy particles that only cause temporary malfunctions, which are referred to as “soft errors” [24,30]. The triple redundant methods explored later in this paper can provide some hardness against hard errors, but a hard error will weaken the circuit to subsequent soft errors in the same section of the chip.

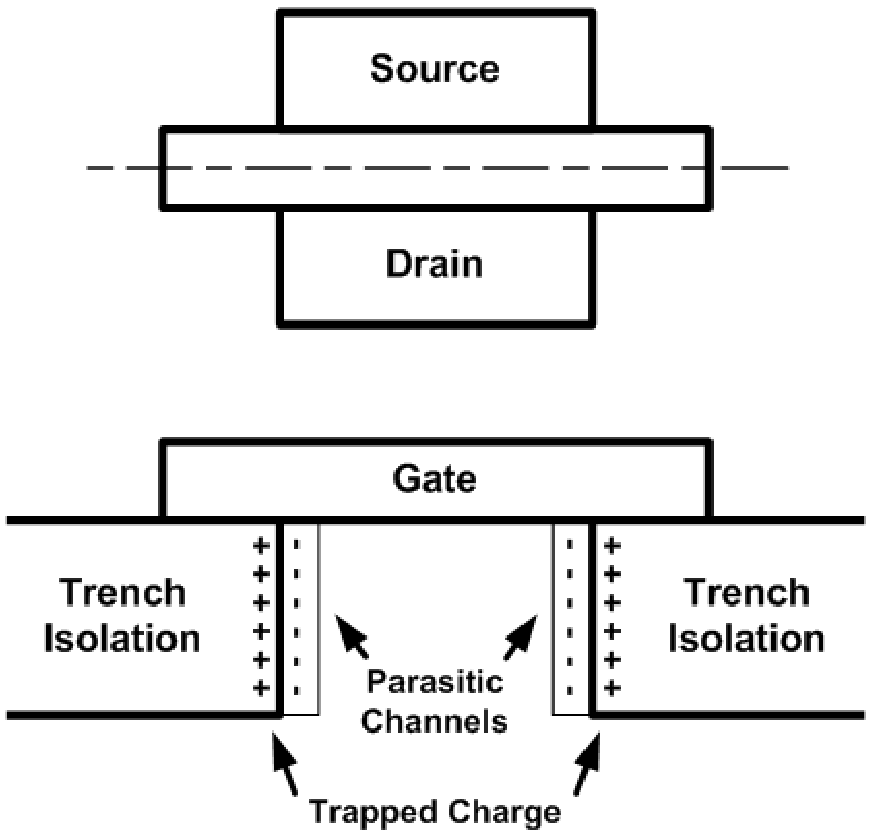


Fig. 2.1. A cutaway view showing the parasitic transistor created when trapped charge in the trench isolation creates a channel. Fig. (a) shows the top view, with the dashed line showing the cut line for Fig. (b). Fig. (b) shows the charge trapped in the trench isolation, which inverts the parasitic channels underneath the gate. These channels connect the source and drain (above and below the plane of this diagram), allowing charge to flow between them.

## 2.1 TID EFFECTS

Although TID effects can be the result of several different mechanisms, the primary TID mechanism for the circuits and process in this paper is trapped charge in the isolation oxide [13,21,31]. As particles travel through the chip, electron hole pairs are created in every layer, but in most silicon and metal layers, these charge pairs recombine after a period of time (<1 second) without causing major changes to chip operation. However, in the oxides, the charges can be

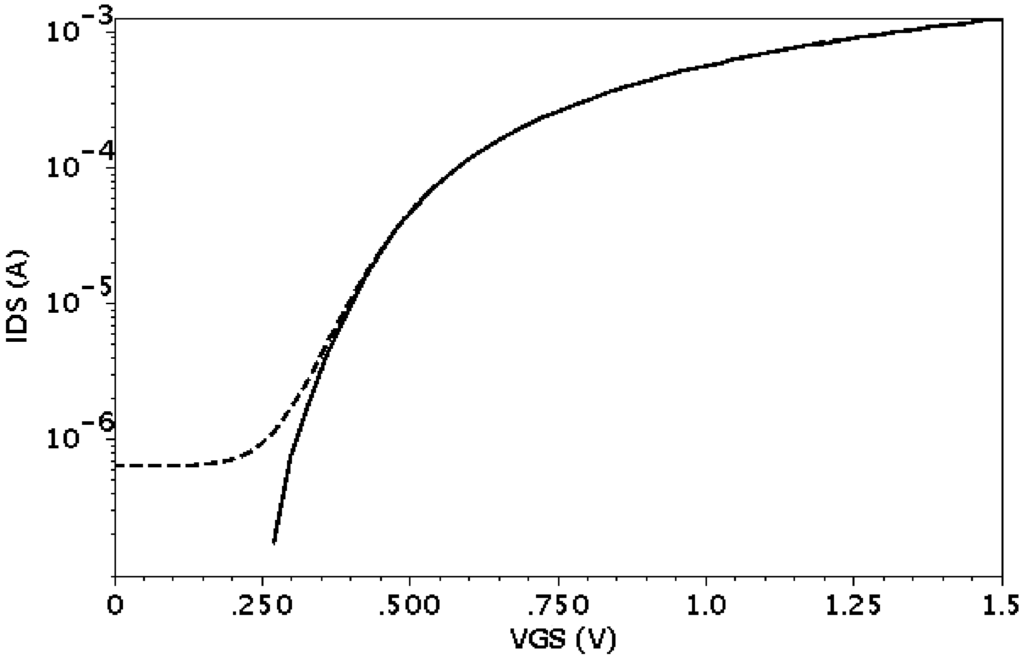
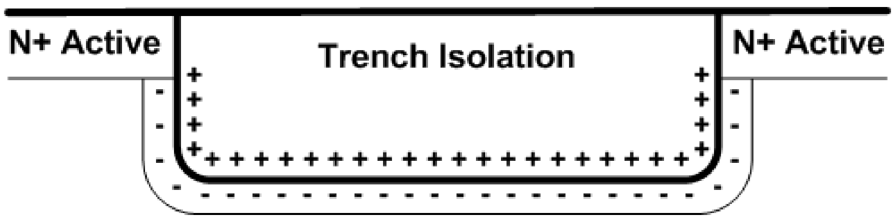


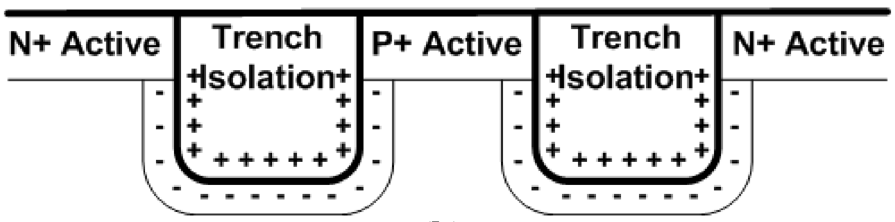
Fig 2.2. The simulated current characteristics of a normal transistor (solid) compared to a transistor with an active parasitic transistor (dashed). The parasitic transistor makes very little difference when $V_{GS}$ is well above the threshold voltage, but it creates a current floor that raises the leakage current at low $V_{GS}$ voltages.

trapped and prevented from recombining either permanently or for very long periods of time. The gate oxide is thin enough to prevent significant charge buildup due to its small volume and mechanisms that make it easier for charges near the surface of the oxide to escape, but the shallow trench isolation oxide between active areas can build up a significant number of holes and electrons. Since the electrons are more mobile, they are much more likely to exit the oxide, leaving the holes behind to gradually build up a positive charge.

The effects of this positive charge depend on the location of the oxide. When the isolation oxide is over the N-well, the N-doped silicon surrounding the oxide shows no major effect from this positive charge. However, when the oxide is over the substrate (i.e. next to an N transistor), the P-doped substrate can be



(a)



(b)

Fig 2.3. A diagram of how inter-device leakage paths are created and how guard rings prevent them. Fig. (a) shows how the positive charge trapped in the trench isolation creates an inversion layer underneath it, connecting two active areas. Fig. (b) shows how the introduction of a P+ active area breaks this path by creating a channel in the trench isolation.

inverted if the positive charge is sufficient. One possible result of this inversion is the creation parasitic transistors where the gate crosses from an active area to an isolation oxide [22]. See Fig. 2.1. Although this parasitic transistor is always turned on, it has poor performance properties and the induced electric field from the thick oxide layer is relatively low. Thus, the current from this transistor is negligible when the transistor is on, but contributes to leakage when the transistor is off. Fig. 2.2 gives an example of the performance characteristics for a transistor when its parasitic transistor is charged.

Another possible result of the charge built up in isolation oxides is the creation of an inversion layer in the substrate between separate active areas [33]. Fig. 2.3 shows how this leakage path forms. When this path is between two active areas

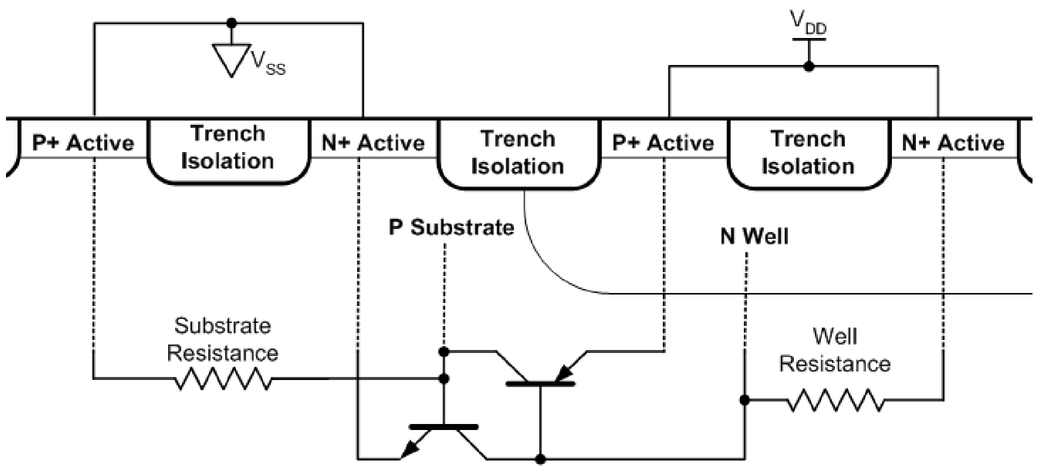


Fig 2.4. The parasitic PNPN device formed by the active areas, N well, and substrate that causes latchup. When charge is injected and the device enters the forward conducting mode, it conducts a large current from the P+ Active tied to VDD to the N+ active tied to VSS.

that are both tied to $V_{SS}$ no current flows. However, when this leakage path occurs between an active area that is not tied to $V_{SS}$ and one that is, it contributes additional leakage to the circuit.

In short, the major contribution from TID effects on the process used for this paper is an increase in the leakage current for a circuit. This leakage is generated either through an increase in transistor $I_{off}$ current or directly through an inverted channel under the isolation oxide. Hardening against this additional leakage will be discussed in section 3.1b.

## 2.2 SEE EFFECTS

Single event effects (SEEs) occur when passage of an ionizing radiation particle in the semiconductor deposits a charge track that is then collected. As stated earlier, SEEs are further split into destructive events (hard errors) and non-destructive events (soft errors). Destructive events caused by extremely high energy particles are hard to address with RHBD techniques, so are largely ignored in this dissertation. The simplifying assumption is that any ion striking the chip is not energetic enough by itself to cause permanent damage to a component. However, the charge deposited by a single heavy ion can still have a significant negative effect on the operation of a circuit.

One possible negative result of this charge deposition is Single Event Latchup (SEL), which can produce a high current between the power rails when a parasitic SCR PNPN device becomes active [25-27]. The P-transistor source, N-Well, substrate and N-transistor source form this parasitic PNPN device (see Fig. 2.4). A PNPN device is essentially two interlocked bipolar transistors that form their own feedback loop when activated. Once activated, the parasitic PNPN device then allows a large amount of current to flow, in this case from P-transistor source at $V_{DD}$ to the N-transistor source at $V_{SS}$.

Since PNPN devices need charge to be injected into the intermediate nodes in order to activate, this charge injection is often done by adding a voltage to one of those nodes or using a pulse of light to generate carriers when these devices are used intentionally in circuits. However, a heavy ion strike also generates charge carriers that can trigger activation. The active current of the parasitic PNPN device can be strong enough to drop the power voltage below functional values and to cause large portions of the chip to be unusable. Even when SELs are non-destructive, they require power to the chip to be cycled off to break the feedback loop in order to turn the SCR off and allow proper function to resume. Toggling the power can be difficult in space applications since human intervention is limited, so preventing SELs from happening in the first place is essential.

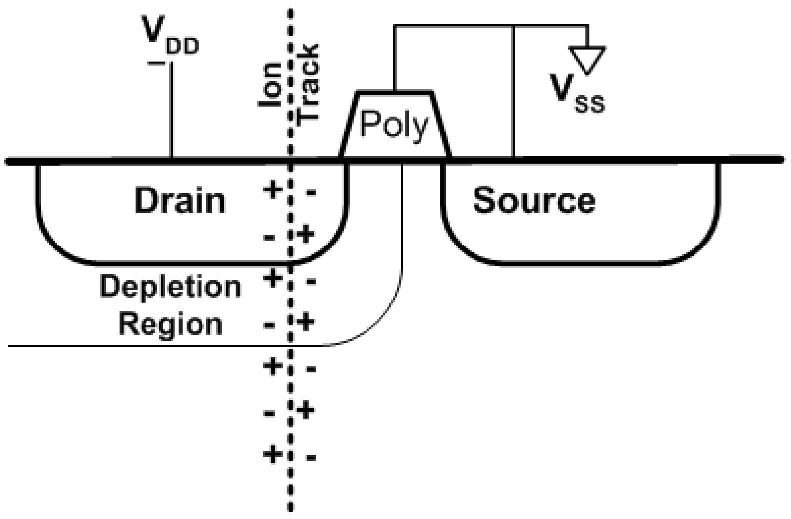


Fig 2.5. Ion strike on a N transistor in its off state. The electrons and holes deposited in the drain's depletion region are separated by the electric field and flow in opposite directions. This creates a current that can pull down the attached node.

To prevent SEL events, the intermediate nodes (the substrate and N-well) must be tied tightly to their respective power rails. However, standard chip design practices use substrate and N-well taps only for low-current biasing, and thus place them sparsely. This sparse placement means that there is a significant resistance through the substrate and N-well (see Fig. 2.4). A large enough LET heavy ion can provide more current than this path can easily remove, and the voltage drop across the resistance is thus large enough that the device can still activate. Increasing the density of the well and substrate taps solves this problem by reducing the resistance which makes activating the parasitic PNPN device difficult or impossible.

Other types of soft errors can also occur when a heavy ion's charge track changes the logic state of a circuit node. This happens when deposited charge from an ion strike is collected by the reverse biased drain-to-substrate diode of a transistor that is currently turned off (see Fig. 2.5). This charge can be either directly deposited in the depletion region, or diffuse into the depletion region from an ion track that passes nearby. Holes and electrons entering the depletion region of this diode are forced by the electric field to separate and flow in opposite directions, which creates a current. If enough charge is deposited, this current can easily override the current from the opposing transistor and force the output node of the logic to the incorrect value. This voltage change then collapses the depletion region, and charge collection slows [7,30].

However, since recombination times on modern processes are relatively long (>10 ns) [6-7,24], the charge remains in the substrate or N-well below the transistor, only dissipating through diffusion to nearby transistors or deeper into the substrate. As the opposing transistor pulls charge out of the affected transistor, the depletion region gradually increases, which pulls in more of this charge and keeps the node in the incorrect state. The output only returns to the correct state after all of the charge has been removed by the opposing transistor. Such an SEE can affect combinational logic, producing a single event transient (SET) that may upset the circuit’s architectural state when sampled by a latch. Similarly, an impinging ionizing radiation particle may upset a stored logic value in a memory cell, creating a single event upset (SEU) [34]. SEUs are then further classified as either a single bit upset (SBU) or a multiple bit upset (MBU) depending upon how many bits are affected by a single heavy ion strike.

The only direct ways to prevent SETs and SEUs from affecting the output of a circuit or storage node are to make the load capacitance large enough that the charge deposited from the ion is lower than the transition voltage ($V_{tr}$) or to change the process characteristics to make charge collection more difficult. However, this is impractical for most logic since increasing the capacitance dramatically increases the size of each transistor and because using hardened processes is expensive. The indirect methods used in this paper for hardening against SEEs make the circuit itself resistant to incorrect values on nodes. These methods will be further discussed in Section 3.3, but it is important to note that each of the techniques used for circuit hardening require that certain vulnerable nodes are never upset at the same time. In other words, if both a node and its redundant copy are upset by the same ion, the hardening is defeated.

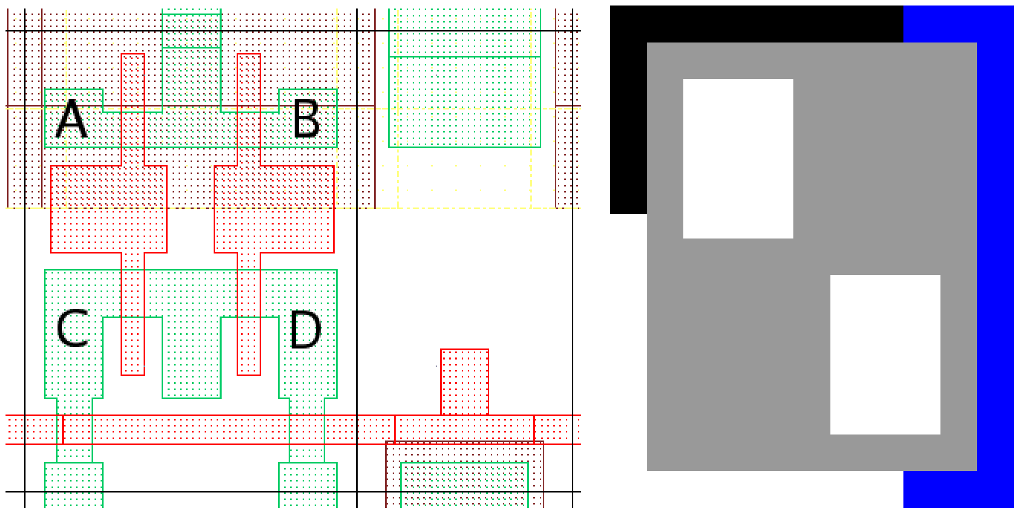
To prevent multiple vulnerable nodes from being upset at the same time, these vulnerable nodes are separated from each other so that an ion of a specific strength cannot deposit enough charge to affect both of them at the same time [35-36]. However, the space required for this separation varies greatly, depending on the energy of the ion, the angle at which it strikes the chip and process characteristics [37-38]. Thus, measurement of these effects on the specific process used is essential. Since the same physical mechanisms underlie both SEUs and SETs, the study of SEUs can then be applied to SETs. Specifically, measuring MBUs can give insight into how far charge from an ion strike can travel. This measurement can then be applied to estimate spacing rules for SET mitigation.

## 2.3 SEE SPACING ANALYSIS

The easiest way to study spacing requirements is to measure SEUs in a storage array and to use the pattern of flipped bits to estimate the charge spread when a heavy ion strikes a chip. In order to gather data on heavy ion strikes, an array of storage cells is placed in a beam chamber, initialized with a specific pattern, and bombarded with a stream of heavy ions [39-41]. The storage cells are continually read and compared to the initial pattern. Any change is recorded before being re-initialized. These upset bits are analyzed later to determine the effect of each detected ion. The ions used in these tests were fired both directly into the chip and at an angle.

The geometry of the storage cells and the nodes inside of them play an important role in how each cell can be affected, and must be accounted for in order to analyze the data properly. The distance between sensitive nodes can vary not just on the layout, but also on the pattern used to initialize the storage cells.

Although the ions in a typical test environment have a very uniform energy, the actual energy deposited into the substrate can be affected by the location in which it strikes the chip. The most significant variation in charge deposition is caused by the shallow trench isolation oxide, which can suppress charge deposition near the surface if it is struck. Also, cells don't always flip when they are hit, adding additional uncertainty to each individual measurement [42-43]. Together, these two effects make statistical modeling essential in measuring the true extent of charge generated by an ion strike, since each individual measurement may or may not reflect the true extent of the charge spread.



**(a)** **(b)**

Fig 2.6. The layout of the SRAM cell used in the SEU testing and its stylized representation. Fig. (a) shows the cell layout with the four vulnerable nodes labeled, as well as the abutting well and substrate tap cell to the right. Fig. (b) shows the representation it would have if nodes B and C were initialized in a vulnerable state. It also indicates the position of the N-well with a black horizontal border along the top, and the location of the tap cell with a vertical border to the right.

*A. SRAM Cell*

The SRAM cell layout used for the tests is shown in Fig. 2.6, alongside its stylized icon. This icon is used in the display of strike patterns in a custom visualization tool. The layout does not use special SRAM DRC rules, but is otherwise minimum size. This gives the maximum possible resolution for the measurements while still allowing full control over the layout dimensions. Since the storage for this arrangement is simply a pair of symmetric back-to-back inverters, there are four active areas that can be hit to change its state. These active areas are labeled A, B, C and D on the layout. Depending on the value stored in the cell, nodes B and C or nodes A and D are turned off and vulnerable to an ion strike. The vulnerable nodes of this SRAM cell are always opposing corners.

The cell is 1.2 microns by 1.6 microns, and the spacing between the closest nodes on different cells is 0.15 microns horizontally, 0.2 microns vertically and 0.25 microns diagonally. All values are rounded to the nearest 0.05 micron.

Because of the relatively close spacing of nodes A and B or C and D, the SRAM cell is very vulnerable to being left in a metastable state from an ion strike. In many recorded strikes at high linear energy transfer (LET), there are gaps in the pattern of disturbed cells where cells should have been affected but did not change state.

## *B. SEE Features*

Several different patterns are noticeable in the data from looking at the patterns in individual strikes and the cumulative size of patterns when the data is aggregated.

### ***Metastable Strikes***

A storage cell with nodes that are both opposing and near to each other can be driven into a metastable state if the amount of charge deposited is high enough. As the charge drives one node to its rail, this tends to flip the circuit and turn the opposing transistor off. This flip begins to create a depletion region in the opposing transistor that sucks charge into the opposing node. With enough deposited charge and weak enough restoring transistors, the limiting factor in the duration of this effect is the diffusion of charges further into the substrate. This

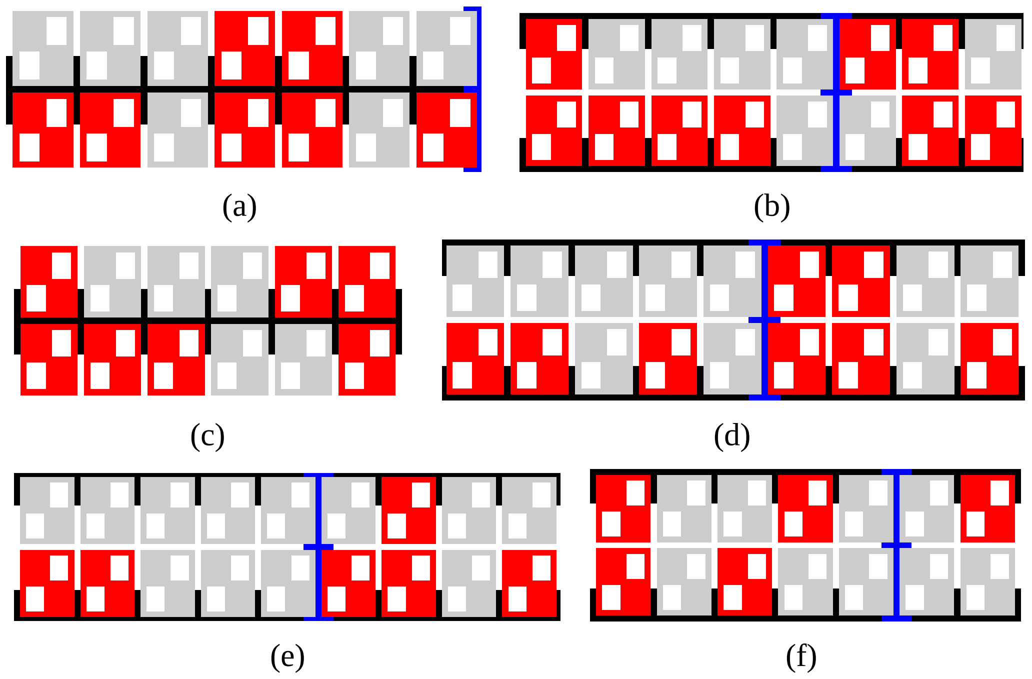


Fig. 2.7. Strike patterns showing meta-stable cells using Xe at 65º.

mechanism tends to result in roughly equivalent voltages at both nodes as the charge also diffuses from whichever node has the most to whichever has the least. The resulting difference in voltage once the charge is depleted is small enough that manufacturing variations combined with the initial state's residual effects have a larger effect on the final state of the storage cell than the small difference in charge deposited at each node by the heavy ion strike.

Examples of this effect are shown in Fig. 2.7. These are displays of the visualization program written for the SRAM. Each cell is represented by an outlined rectangle and two shaded rectangles showing the vulnerable nodes. The examples were selected from the highest LET run with the SRAM, Xe at an angle of 65 degrees. The resulting patterns from this run show many wide hits, but almost all of them have gaps where cells could have flipped but did not. Which

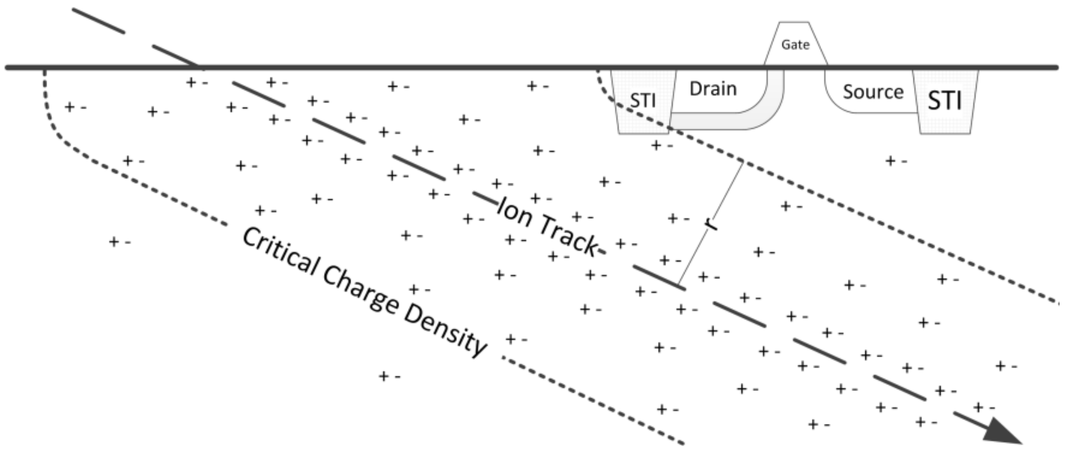


Fig 2.8. Charge diffusion for a high energy angled heavy ion strike. Positive and negative charge carriers are generated and diffuse through the substrate equally until they reach the depletion region underneath a transistor drain. The critical charge density indicates where enough charges remain to force the drain voltage past $V_{tr}$ for the node.

cells flip is not obviously systematic and seems to be due to manufacturing variation on individual cells.

## *LET vs. LET$_{eff}$*

Since the ion beams used in testing are often fixed in strength based on the ion used, the angle of the ion is modified to make quick adjustments between tests. If an assumption is made that the charge deposited is relatively small and confined to a single transistor drain, the angled ion travels a longer distance through the collection area and transfers more energy near the surface, thus depositing more total charge in the depletion region. Therefore, the effective LET for some cases can be adjusted by $1/\cos\theta$ where $\theta$ is the angle of the ion [44-45]. For this adjustment, the label $LET_{eff}$ is used to differentiate the measurement from the true LET of the ion itself.

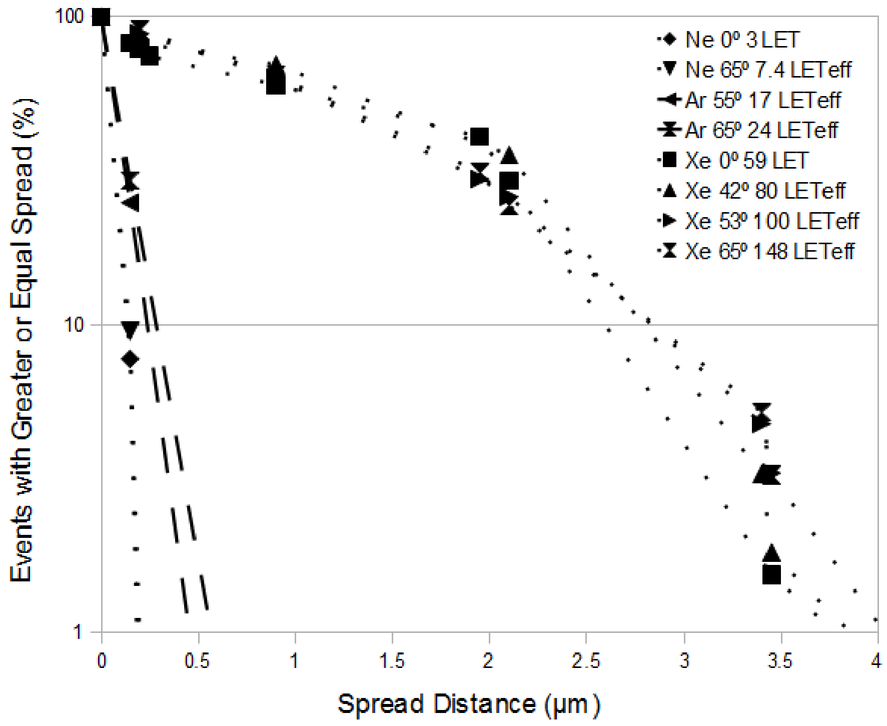


Fig 2.9. The spread due to LET for various ions, perpendicular to the incident angle. The correlation between the ion species regardless of angle shows that the charge spread around the impact point depends primarily on the actual LET of the ion, not the $LET_{eff}$ that attempts to factor in the incident angle.

However, this does not apply to higher energy strikes. Instead, one useful simplifying assumption is that the charge deposited by heavy ion strikes diffuses out from the track of the ion in a cylindrical shape, where the charge density decreases proportionally to $1/r^2$ where r is the radius. Since it takes a specific amount of charge collected in order to drive a node past the transition voltage of the inverter and flip the cell, a strike on a storage cell can be modeled as a cylinder whose radius is such that the charge at the edge is just enough to flip a cell [37-38]. Fig. 2.8 shows a cross sectional illustration of this model for an angled heavy ion strike. If the critical charge density reaches the depletion region underneath the transistor drain, the transistor will be affected. Since the positive and negative charges are generated together and there are no significant obstructions in the substrate, both carriers diffuse approximately equally.

Fig. 2.9 shows a distance comparison for the spread of each LET. The x axis is the minimum possible distance between vulnerable active areas. The minimum value is used because it is impossible to tell which vulnerable node is hit, so this value may end up underestimating the actual node distance. The y axis is the percentage of cells showing at least this spread distance. This plotting method is used to allow comparison between different initialization vectors and cell geometries. Thus, all graphs start with 100% at 0 microns and decrease towards 0%. The distance used for angled strikes is perpendicular to the direction of the ion, thus it measures the spread of the charge away from the ion track.

At the lower end of the plot, multi-bit hits were detectable at energies as low as 3 MeV-cm$^2$/mg. However, at this LET, strikes did not result in MBUs past the minimum detectable 0.15 microns and MBUs were completely absent when the initialization pattern moved the vulnerable nodes further apart. At 7.4 MeV-cm$^2$/mg LET$_{eff}$ (also using Ne, but at 65º), there are a few hits on nodes 0.2 microns apart.

Further, when the charge deposited increases and the charge is delivered to the transistor drains mostly by diffusing through the substrate, the 1/cos θ effect disappears. The Xe ion at 59 MeV-cm$^2$/mg LET generates roughly the same diffusion to the sides of the strike no matter what angle is used. An angular strike will deposit more charge, but it is spread out in an ellipse instead of constrained under a single collection area. The axis perpendicular to the direction of the strike

TABLE 2.1
COMPARISON OF MEASURED VS. CALCULATED SPREAD DISTANCE

| Effective LET | Min. Spread | Max. Spread | Calculated Min. | Calculated Max. |
|---|---|---|---|---|
| 59 (0º) | 3.5 µm | 4.5 µm | | |
| 80 (42º) | 4.5 µm | 5 µm | 4.7 µm | 6.1 µm |
| 100 (53º) | 5.7 µm | 6.9 µm | 5.8 µm | 7.5 µm |
| 148 (65º) | 9.8 µm | 11.7 µm | 8.3 µm | 10.6 µm |

TABLE 2.2
AVERAGED GEOMETRY OF XE 65º STRIKE ALIGNED WITH N-WELL
(% OF CELLS FLIPPED PER COLUMN/ROW)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 4 | 3 | 1 | 1 | 2 | 2 | 0 | 0 |
| 2 | 2 | 12 | 24 | 29 | 33 | 32 | 37 | 27 | 10 | 1 |
| 3 | 2 | 14 | 34 | 38 | 42 | 37 | 42 | 27 | 9 | 1 |
| 4 | 0 | 3 | 6 | 6 | 4 | 1 | 2 | 2 | 0 | 0 |

is expected to be the same as the previous diameter, while the length is extended by 1/cos θ. This indicates that the high energy cylindrical model is correct for the Xe tests.

Fig. 2.10 shows the same Xe tests, but the measured distance is in the direction the ion is traveling instead of perpendicular. This measures the longer axis of the ellipse. Table 2.1 gives the measured and calculated values for each of the different angles, based on the minimum and maximum spread in the 0º test. In order to eliminate outliers, the minimum and maximum used are from the data points before and after the 0º test drops below 1%. The distance is calculated simply by dividing the spread by cos θ.

Table 2.2 shows how the two measurements of spread compare for the largest angle, Xe at 65º. Each event has a center point calculated for it, and then all the strikes are added together to form a picture of the diffusion pattern. Since the

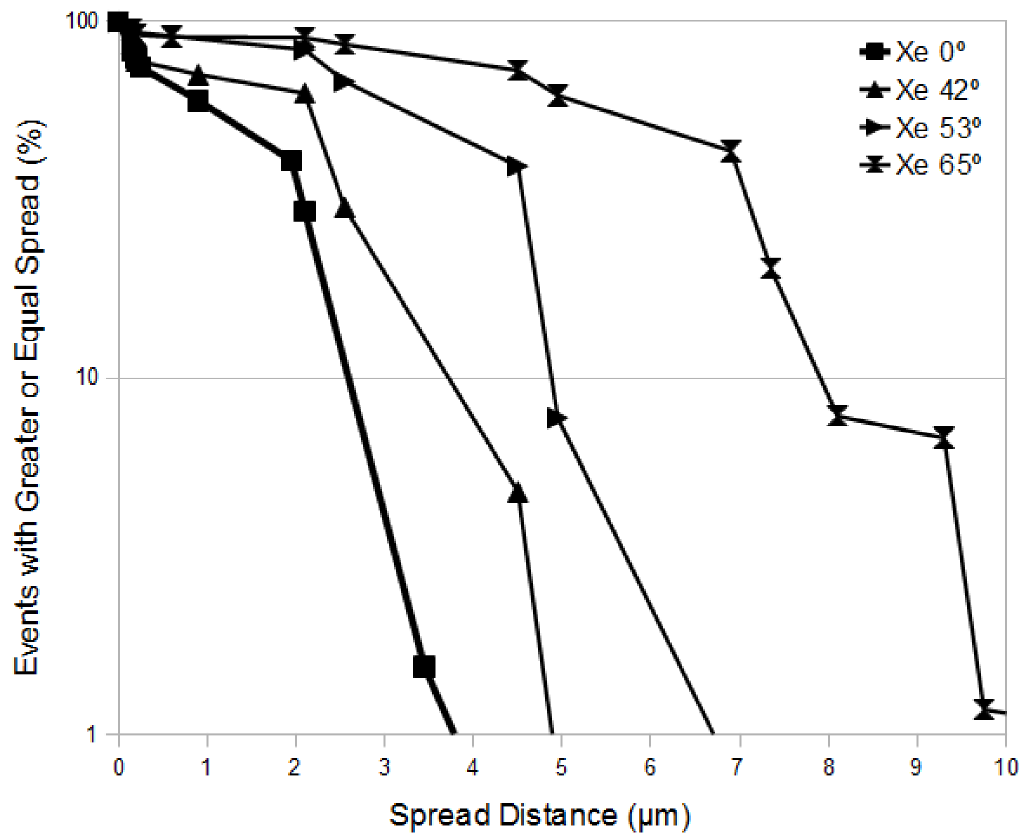


Fig 2.10. The spread due to LET for various ions, parallel to the incident angle. The increased spread due to higher angles gives a measurement of the long axis of the elliptical strike pattern.

center is calculated as an integer, there is a small bias to the bottom left side of the plot. This bias is <=1 row or column. For instance, the highest value on the table (42%) occurs where the SEUs are placed on the grid, since SEUs are 9% of the total events. The center 2 rows and 8 columns have an average upset chance of only 28%, even though all of these cells should have collected more than the upset threshold charge.

Rows 1 and 4 of this table show a distinct skew to the left side of the chip. This skew is due to the ion angle striking the chip moving from right to left. If it

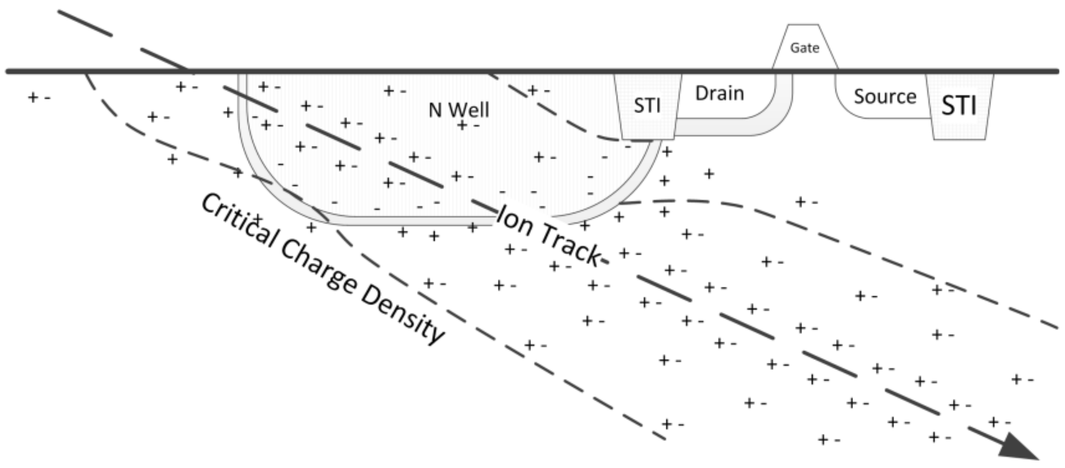


Fig 2.11. An angled high energy heavy ion strike through an N-well. The critical charge density is shown for negative carriers. The N-well reduces the carriers that reach the N transistor drain.

strikes in the N-well, the charge is primarily constrained within the well for the first portion of its length [32]. Further along the track of the ion, it can exit below the N-well and diffuse charge up to nearby transistors in the row of cells above or below.

***N-Well Orientation***

Another factor that effects how much charge appears at the transistor for angular strikes is the orientation of the strike compared to the N-well [46]. The calculations above assume that all the charge diffuses outward evenly from the path of the strike, and the surface of the chip is essentially flat and featureless. However, this assumption does not account for obstructions disrupting the uniform diffusion of charge. N-Wells are deep enough and form large enough depletion regions that they form significant barriers to this flow.

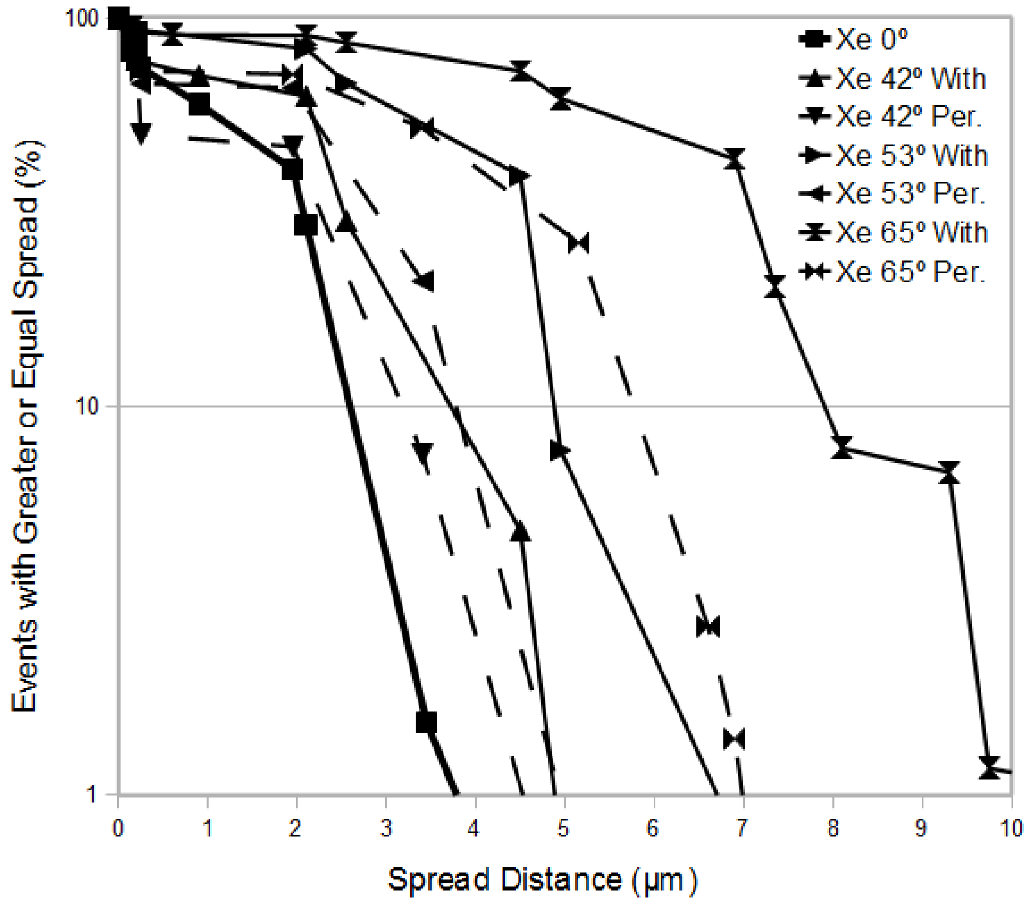


Fig 2.12. The spread due to angular strikes for Xe, aligned both with and against the N-Well. Ions aligned with the N-Well are solid lines, while those aligned against it are dashed.

If the ion strike is in the same direction as the N-wells, the strike and its charge tend to travel between or through the N-well, resulting in good agreement with the calculated spread. However, if the ion travels perpendicular to the N-well, the intrusion of the N-well structure and its depletion region can reduce the effect of ion strikes. Fig. 2.11 illust rates this effect, focusing on the negative carriers that can affect the N transistor drain shown on the right. Negative carriers near the N-well depletion region will tend to diffuse into the N-well, but cannot diffuse the opposite direction. This lowers the negative carrier density on the outside of the

N-well, and reduces the number of electrons that reach the transistor drain. The opposite effect occurs for positive carriers, which tend to diffuse out of the N-well and away from the P transistors that can be affected by them.

Fig. 2.12 shows the Xe tests at angles both with and against the N-wells. In every case, the test angled against the direction of the N-well resulted in a significantly shorter W. The exact effect is hard to predict due to its dependence on layout variations, but it can reduce the angular contribution to an MBU's extent by over 50%.

## *C. Spacing Analysis*

Data from heavy ion irradiations of 90nm bulk silicon show the effects of several predictable variables. Rotation of the chip, angle of incidence of the ion, and the base LET of the ion all have significant effect. In a normal incident hit, a heavy Xe ion with a base energy of 59 MeV-cm$^2$/mg can cause disruptions in nodes 3.5 microns apart. With higher angles, this can exceed 10 microns on strikes aligned with the N-Well. With strikes aligned against the N-Well, however, this number drops to somewhere over 5 microns. The cells discussed in the next chapter have a cell height of 4.48 microns and use one row of filler cells as a gap. This spacing puts the nearest possible vulnerable nodes at 5.2 microns apart. This provides reasonable protection against 59 MeV-cm$^2$/mg ions at 65º, but there are still some strikes that will likely cross that boundary. Further calculation of the odds that a strike will cause an error depends on many additional factors, including the flux of the radiation environment in which the chip is expected to operate, as well as the probability that two nodes in the same logic cone will be placed across from each other by the APR tool.

## 2.4 CHAPTER SUMMARY

The TID effects on circuits primarily increase leakage currents. This increased leakage both burns more power and complicates the design process when a node needs to be carefully balanced around $I_{off}$ currents (i.e. large numbers of transistors are connected to a node with a keeper). In addition, SEE effects complicate circuit design, since it must be assumed that almost every node in a circuit can be driven to an incorrect value at random times. Experimental data is used to demonstrate that the spread of these incorrect values is localized to a small portion of the chip per event. This localization gives separation guidelines for hardening methods. The next chapter will discuss hardening methods to reduce or eliminate TID and SEE effects.

# CHAPTER 3

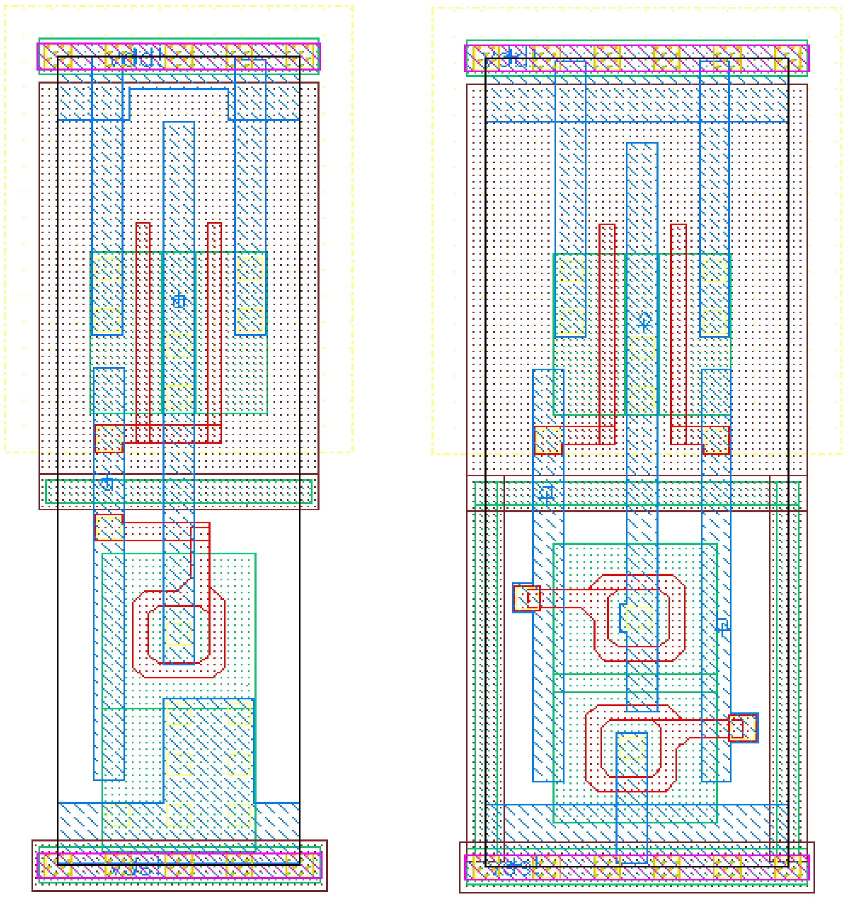
# CELL LIBRARY DESIGN

The basis for any synthesis process is the cell library, which has to include a multitude of different cells and drive strengths to achieve good performance [47-50]. Additionally, the library needs to be designed to handle many TID and SEE effects, as well as include various versions of the triple redundant flip-flops that will provide protection from SETs and SEUs. Many of these hardening features increase the size over unhardened gates, but are required to achieve the needed performance.

Originally, the cell library was going to be used primarily for hand placement of circuits. Thus, many features were added that allowed easier placement and routing by hand. Although these are rarely used in the current implementation, they are retained to ease understanding of the layout.

## 3.1 ORIGINAL COMBINATIONAL CELL LAYOUT

The first iteration of the example inverter and nand cells is shown in Fig. 3.1. The library has a cell height of 4.48 microns and a horizontal cell pitch of 0.32 microns. Basic hardening against SEL and TID effects are added to each cell in order to maintain consistent protection against effects. These cells use guard rings and a strip of substrate/N-well contact underneath the power rails to reduce SEL effects, as well as annular transistors to reduce TID leakage effects. The layout density and complexity impact of these added structures is discussed below.



**(a)** **(b)**

Fig. 3.1. Example cells from the original version of the library.

## *A. Single Event Latchup hardening*

Single Event Latchup (SEL) resistance is provided in two ways, with a guard ring between the N-well and the N transistors, and with a line of substrate and well contacts under the power rails. As discussed in Section 2.2, SEL is a result of a single ion striking the chip, and depositing charge in the substrate. This charge then turns on a parasitic PNPN device consisting of a P-transistor source tied to $V_{DD}$, the N-Well, the substrate, and an N transistor source tied to $V_{SS}$ (See Fig. 2.5).

The easiest way to reduce the risk of SEL is to tie the substrate and/or N-Well to their respective power rails with low impedance connections. Non-hard, commercial substrate and well contacts are inserted only rarely between gates because they only need a tiny current to maintain the proper biasing conditions. By increasing the density of these contacts, charge in the substrate can be removed more quickly, and the risk of a self-sustaining latchup condition is reduced. In this cell design, a strip of P-doped active area is created beneath the power rail and tied with a row of contacts to create a large substrate contact as part of every cell. Additionally, a guard ring is inserted between the N transistors and the N-well, consisting of a strip of P-doped active that is occasionally tied down to the substrate contact beneath the power rail. Although there are no direct contacts to this guard ring, which increases its impedance to $V_{SS}$, it is placed directly in the area where the parasitic latchup device forms and has very low impedance to the parasitic device. Thus, the resistance shown in Fig. 2.5 is reduced and the parasitic PNPN device is much more difficult to activate.

Since the guard ring is an active area between the N and P transistors, poly structures cannot cross it without breaking it. This complicates the cell design by requiring separate headers for P and N gates, and increases the wire density in the center of the cell. However, the cell design complications are a necessary price to pay in order to maintain reliable operation of the device in high-radiation environments.

*B. TID Leakage Hardening*

As discussed in section 2.1, leakage from TID effects usually results from charge buildup in oxides. Most importantly, this occurs at the end of transistors where the poly meets the edge of the active area. As charges accumulate in the isolation oxide, it can turn on the edge of the active area, forming a parasitic transistor that is always slightly on. The increased leakage current can dramatically increase passive power dissipation. Since the insulator builds up a positive charge, P-transistors are largely immune, but normal N-transistors have 2 parasitic transistors per drawn transistor. Annular transistors, however, have only 1 parasitic transistor [51-52]. This transistor has no current flow or leakage because both the source and drain of the parasitic transistor are connected to the same node. In the example inverter in Fig. 3.1(a), both sides of the parasitic transistor for the inverter are connected to $V_{SS}$, while the example nand gate in Fig. 3.1(b) has the parasitic transistors connected to the intermediate node in the middle of the stack. These transistors do have a small “neck” of useless poly to connect the ring to the poly head, which adds extra capacitance. This “neck” requires a small extra transistor in the schematic that is simply tied to ground to act as additional capacitance.

Since 1.2 microns is the smallest width possible for an annular transistor on this process, this placed stringent limits on how small cells can be in the original version of the library. This size limitation also increases the overall power consumption of the circuit. The end result is circuits that have high active power consumption, but their power use does not increase due to TID effects.

The same oxide charge buildup that creates parasitic transistors can also create leakage paths between N-transistor active areas [33]. As shown in Fig. 2.3(a), oxide charge buildup at the bottom interface of isolation oxides can invert the substrate beneath it, effectively creating an N channel region at the substrate/oxide interface. This charge buildup cannot affect active areas that are completely enclosed with poly, which makes the inner node of annular gates immune to the effect. Thus, any annular transistor that has its outer active area connected to $V_{SS}$ is immune to this leakage path.

However, for cells like the nand gate shown in Fig. 3.1(b), additional guard structures in the form of a fully enclosed ring of substrate contact are required to create an effective block to this leakage path. As shown in Fig. 2.3(b), the inclusion of a P-doped contact creates a pair of back-to-back diodes in the leakage path, which blocks current flow. In the layout, this substrate contact is essentially extending the guard rings discussed in the previous section with vertical connections along the side such that the vulnerable node is fully isolated. Since only some cells require this, the original cell library leaves these structures out of many cells in order to increase density. However, this omission requires either hand layout or a placement program that provides extra space when necessary to avoid design rule violations when cells cannot be placed next to each other.

It is worth noting that the cell names in this library are based on the size of the N transistor. The resulting convention was a description of the cell type, an “x” character, then the size of the N transistor which was normalized such that the smallest one created a gate of size “010”. Thus, the smallest inverter cell is named “invx010” and the smallest nand cell is named “nand2x010”. Larger cells were created in values that approximated the square root of 2, such that going up two sizes would double the drive strength of the cell. Thus, cells types with 3 sizes were typically “x010”, “x014”, and “x020”.

### *C. Overall Hardening impact*

The hardening structures in the previous section can have a significant negative impact on transistor density. Both the guard ring and well taps underneath the power rails require additional space to avoid the power lines, as well as forcing the use of separate headers for N and P transistor gates. Since the guard ring is an active area placed between the N and P transistors, it is impossible to use poly routing between N and P transistors, as is common practice in non-hardened libraries. With all these considerations, the chosen cell height of 4.48 microns allows a maximum N transistor width of 1.2 microns, while the maximum width of a P transistor is 1.6 microns. Without these features, the same cell height could fit transistors that are 1.75 microns and 1.85 microns, respectively.

In addition to the changes mentioned above, the power rail is reinforced with M2 and a row of vias to reduce its resistance and increase the current drive available at the local level. This helps reduce recovery time from the SETs that will be discussed in the next section. However, it does force M2 to be routed in a horizontal direction, instead of the preferred vertical direction. Although the improvement to power rail stability is a great advantage in SET recovery, this change in orientation does have a negative effect on routing density.

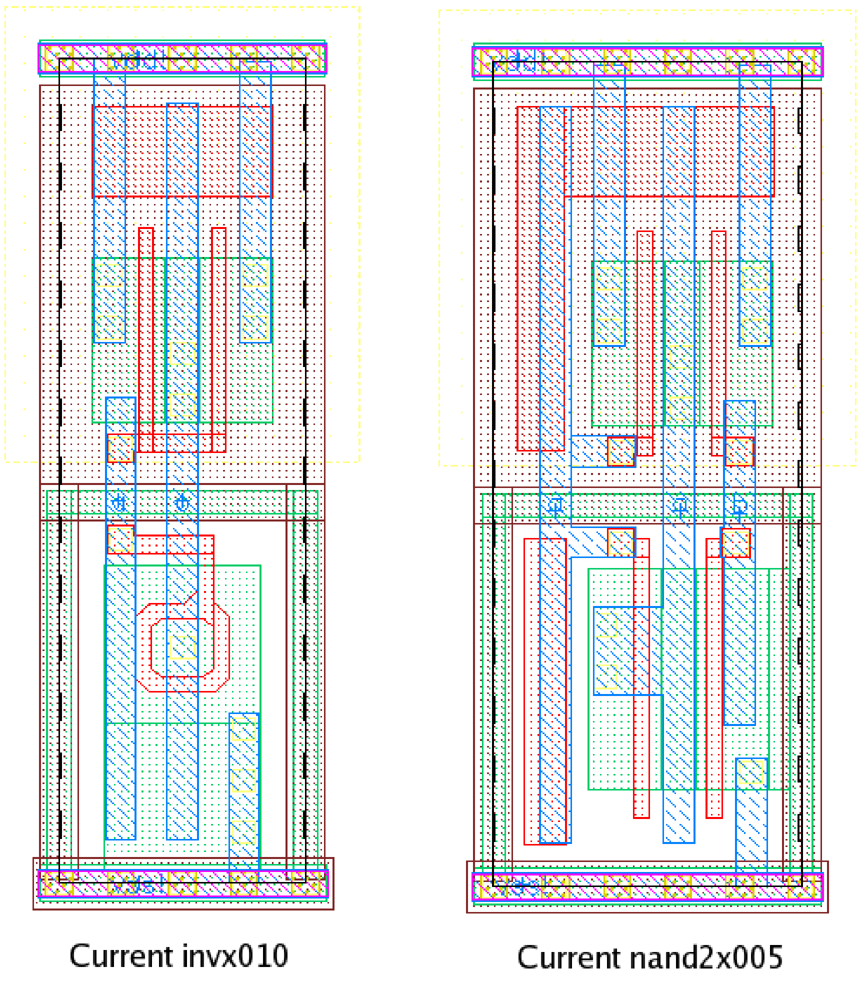


Fig. 3.2.  Example cells from the new version of the library.

## 3.2 IMPROVED COMBINATIONAL CELL LAYOUT

The original cell library was used in the design of several chips that proved to work well in testing.  However, a few issues with ease of use became apparent with both hand-built and APR layouts. These issues lead to an adjustment of several elements of the layout for the final version of the cell library.  A set of example cells that were the result of these changes is shown in Fig. 3.2.

## *A. Transistor Size Limitations*

In order to allow for a wider variety of cell sizes and reduce active power consumption, the first major change was the inclusion of standard 2-edge N transistors in some cases. The minimum size of annular transistors increases the total transistor widths by a significant amount, which in turn increases the total power consumption of a circuit, as discussed above. While this can be a necessary penalty to reduce leakage power, in the tests run on circuits for our applications it was found that the increased leakage power of 2-edge transistors was offset by the smaller active power of a smaller transistor. This benefit is especially true when transistors are stacked, as in the NAND example. Since the TID induced parasitic transistors are only turned on by trapped charge, their effective gate voltage is low enough that they never saturate, and thus operate in the triode region. In this region, the current flow is exponentially dependent on the drain to source voltage. This voltage is typically cut by half or more when transistors are stacked, resulting in dramatically lower leakage current.

With both of these data points taken into consideration, the design guidelines were changed to use annular N transistors only in cases where they weren't part of a stack and where the size of the cell would call for at least a 1.2 micron transistor anyway. Smaller cells were then added to reduce the minimum transistor width down to 0.3 microns.

## *B. Hand Layout Improvements*

Since these cells were originally used for hand layout without APR tools, several tweaks were added to improve ease of use. The first major change was to normalize the size number in the name of the cell to represent the drive strength instead of the N transistor size. I.e., the “nand2x010” cell was renamed to “nand2x005” due to the stacked N transistors reducing its drive strength by approximately half. This convention made calculating the proper size for the fanout of transistors easier convert the load into its equivalent-sized inverter, divide that number by 4, and immediately know what output drive of any cell should be used to achieve a fanout of 4 drive ratio for that load.

However, in hindsight, this could have been improved further to completely remove the dependence on the “invx010” cell as a reference point by directly using the transistor width. Since the invx010 cell has a total of 3 microns of transistor width, it should have been “invx0300” to represent 03.00 microns of drive strength. This would simplify fanout calculations even further, requiring only that you add up the total width of transistors for the load, and dividing that number by 4 to find the ideal cell strength to drive the load.

The other changes to hand designed layout functionality assist with aligning wires and locating pins. Since the vertical wire pitch matches the pitch of the vias in the power supply, it became easy to ensure that vertical wires were aligned properly by ensuring they landed directly on top of a via in the power line. Since the same type of structures do not exist to align horizontal wires, small rectangles were added to the text layer outline at the edge of each cell, indicating the proper spot for horizontal wires to cross cell borders. These indicators sped up drawing interconnect between cells dramatically, since you could drop a path of metal in

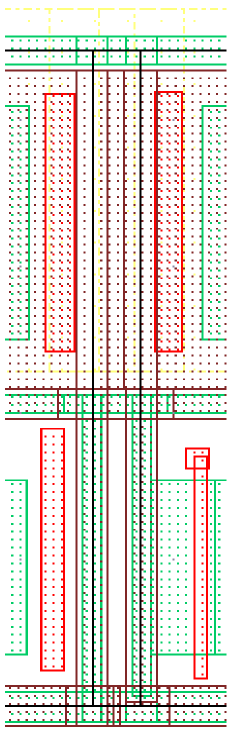


Fig 3.3. Guard ring collision between two standard cells placed 1 cell pitch apart. The spacing between the two rings causes a DRC error unless the filler inserted in between these cells joins the two rings.

the approximate spot, then move it to the markers without any need to measure distances to ensure design rules weren't violated.

Finally, in a small but important change, all of the labels that indicate pin locations were moved to be on top of the guard ring, if possible. This not only created a consistent spot to look for these pins; it was easier to see the blue labels over a green background instead of the normal brown or black background. Although hard to quantify, the increased visibility does increase the speed of hand layout with this cell library.

## *C. Auto-Place and Route Improvements*

Although the original design for cells without guard rings on either side does increase density, it was found that this created several special cases that were not handled properly by the APR tool, most likely due to limitations of the technology file. The APR tool was unaware of the active areas of each cell, which means it was unable to properly account for the extra space needed between some cells without side guard rings, and other cells that did have them. Additionally, if two cells with side guard bands were placed one cell pitch apart, the guard bands were close enough that a special spacer needed to be used to fill in the gap instead of leaving a hole in the active area that was too small to pass design rule checks (see Fig. 3.3). To fix this issue, side guard rings were added to every cell, whether they needed it or not, forcing some cells to increase in size. The single cell pitch spacer was then modified to merge the guard rings of adjacent cells, while a 2-cell pitch spacer was added without the guard ring merge. Note that larger spacers are unnecessary, since a 4 cell pitch decoupling capacitor is used for larger fill areas. The added capacitance on the power rails supplied by this cell assists with recovering from the current spikes that can occur with some SEEs.

One feature of the APR tool that was not understood properly in the initial design was how it places pin vias that connect to pins in the cell. Since all of the metals used in routing are constrained to a grid, pins can only be inserted at the intersection of lines on that grid. The grid is dense enough that it is always able to connect to some part of the pin. However, since the via cell used to connect pins has M1 in it, the via ends up creating an extra stub of metal if the pin isn’t directly

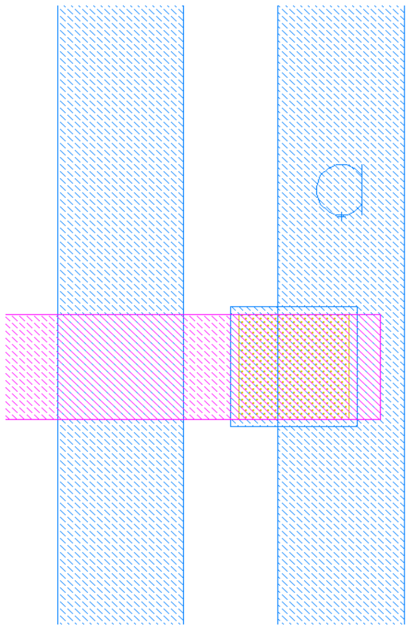


Fig 3.4. DRC error caused by off-grid pin placement. When the router places a via on-grid, it causes a bulge in the metal 1 pin, which reduces the space to the adjacent metal. Since the metal was already placed at the minimum spacing, this causes a DRC error.

underneath this grid (see Fig. 3.4). If there are other wires near or at the minimum distance to the pin, this will then cause design rule errors. Manually fixing these issues is not that difficult, but requires human intervention and delays the routing process as the tool tries to fix DRC errors that are unfixable with the rules it is using. To smooth routing and save manual adjustment time, the revised version of the library requires that all pins are aligned to the routing grid. Thus, a via connecting to a pin will never change the M1 outline of a cell and no new DRC rules will be violated.

## 3.3 FLIP-FLOP HARDENING

The hardening methods used on the combinational cells will handle several TID and SEE issues, but they do not handle SETs and SEUs. To handle these properly, flip-flops must use additional hardening techniques to ensure the state of

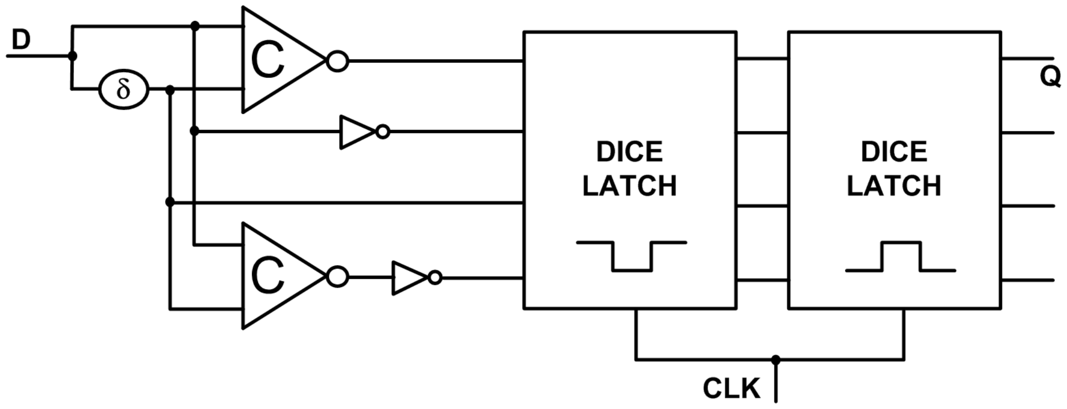


Fig 3.5. Example temporal flip-flop design. The delay δ is tailored to the expected $t_{SET}$ and fed into two Mueller C gates to suppress SETs, while the DICE latches suppress SEUs.

the stored data is correct. The two commonly-used approaches to hardening flip-flops to SETs are temporal redundancy and logic redundancy. Temporal redundancy involves sampling the input at multiple points in time and setting the input to the flip-flop based on the dominant input state [35,53-56]. Logic redundancy (often called Triple-Mode Redundancy or TMR) uses three copies of the input logic and voting circuits to correct for an error in one of the copies [56-57]. TMR also works to prevent SEUs, while designs with temporal hardening often use SEU-hardened latches as part of their structure [58].

### *A. Traditional Temporal Hardening*

An example schematic for a temporal hardened flip-flop is shown in Fig. 3.5. The input and a delayed version of the input are used to drive 2 Mueller-C gates, which combine to drive a dual-interlocked storage cell (DICE) latch [10]. As long as the delay element provides more delay than the expected SET duration ($T_{SET}$), only one of the inputs to the DICE latch is incorrect, and the latch can correct the error. Similarly, an SET on the C gates or on the delay element will only cause one incorrect input.

There are some layout issues that must be considered to make this viable, however. If an SET affects 2 C elements or a C element and a delay node, the protection fails. Similarly, the DICE cell must be designed such that it doesn't receive hits on two nodes at the same time. Protecting these nodes from being hit at the same time requires that they be spatially separated on the chip. To prevent this spatial separation from wasting space, temporal flip-flops may be designed in interleaved banks, so that the space needed for this separation is automatically filled with another flip-flop in the bank [35]. An SET is thus likely to hit two different flip-flops in a bank instead of a single flip-flop twice.

The advantage of the temporal approach is that only a single version of the combinational logic between flip-flops is required, which reduces space and power consumption from those sources. It also tends to be easier to use in APR tools, as long as you can ensure that the multibit cell comprised of 4 flip-flops is handled properly. However, there are also several disadvantages with temporal flip-flops. Chief among them is the speed of the flip-flop, since the flip-flop setup time has to wait for the input to propagate through the delay element and be stable for at least $T_{SET}$, thus reducing your maximum operating frequency. This delay is compounded by the fact that an SET at the right time can delay this by a further $T_{SET}$ before the correct data is stored. Although this delay has traditionally been a relatively small penalty, process scaling effects have increased the duration of $T_{SET}$, resulting in a large speed penalty. The delay element also dissipates a lot of

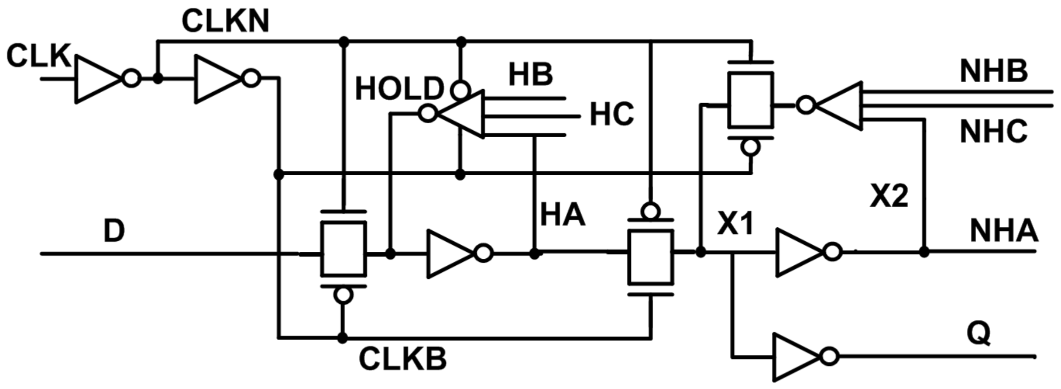


Fig 3.6. The traditional TMR hardened flip-flop design allowing self-correcting during both clock phases. Outputs **ha** and **nha** are sent to the other two copies, while **hb**, **hc**, **nhb** and **nhc** are inputs from the other two copies.

power and takes up a significant amount of space. It is possible to lower the power of a delay element by under driving each stage, but this makes the delay element itself vulnerable to SETs, since an underdriven node can take much longer to recover [6]. Finally, the basic temporal design is not hard to SETs on the clock signal. It is possible to design the clock such that it has enough capacitance to avoid SET upsets, but this means you can't gate it effectively. Design variations that are hardened to clock SETs can be used, but they also have multiple delay elements, which increase the power and space spent on them.

### *B. Traditional TMR Hardening*

An example of traditional TMR flip-flop design is shown in Fig. 3.6. It consists of a master-slave setup, where each latch has been modified to include a majority voter in the feedback loop [59]. These latches are then connected in sets of 3, and the inputs are provided by 3 separate copies of the combinational logic. As long as only one copy of the logic is affected by an SEE, the other two copies will vote to correct it quickly. Even if the output of the flip-flop is hit, forcing it to output an incorrect value, the next flip-flop in the pipeline will vote the data correct as soon as it appears.

However, in order to ensure that only one copy is affected by an ion, each copy needs to be spatially separated from the others. The easiest way to ensure this is to employ them in banks of flip-flops, similar to the temporal case. For purposes of comparison, we used banks of 8 flip-flops. Although there is no direct loss in speed with this design, there are significant issues with parasitic effects. If the flip-flops are separated by 8 cell heights, the latches will need to drive 16 cell heights worth of wire load, in addition to the load of 2 majority gates. Thus, the master latch has to be increased in size in order to drive the wires at a reasonable speed. Adding this transmission delay and loading results in a moderate increase to the setup time of the flip-flop, as a result of the master latch voting.

One advantage of the traditional TMR approach is that it is easily clock gated. No extra structures or logic need to be used, unlike the temporal approach. The impact of this will be discussed further in the performance analysis section below.

*C. High Speed TMR hardening*

Our initial solution for a MSFF for high-speed TMR is shown in Fig. 3.7. The slave latch feedback path uses a majority gate driven by the other redundant copies. When the clock rises, the slave latch contains the state data and the master latch is transparent. In this clock phase, i.e., when **clk** = 1, the state of the slave latch is voted to be the same as the majority of the triple redundant copies, since the feedback gain element is a majority gate. This provides the self-correcting feature, which allows clock gating, in the triple redundant self-correcting MSFF

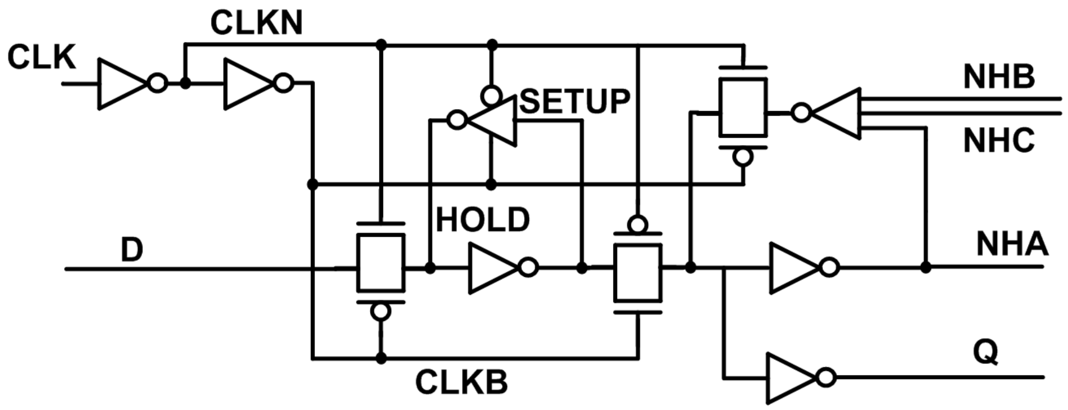
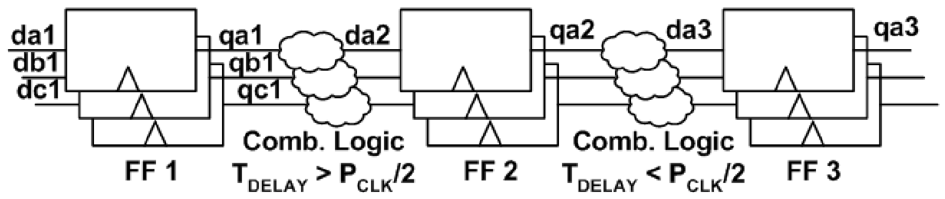


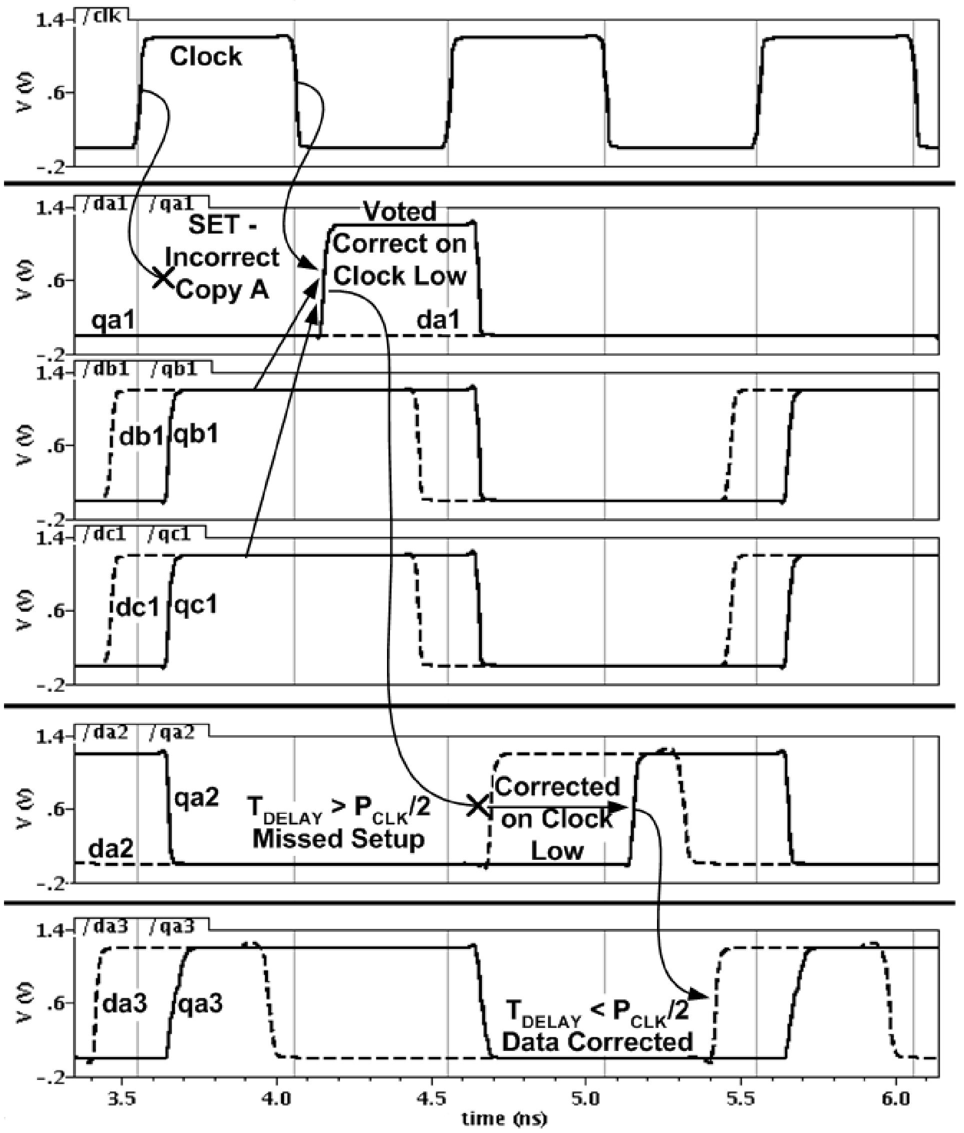
Fig 3.7. Initial MSFF design allowing self-correcting during clock low phase. Output **nha** is sent to the other two copies, while **nhb** and **nhc** are inputs from the other two copies.

(TRSCMSFF). Node **nha** is distributed to the other copies and nodes **nhb** and **nhc** provide the other copies' slave latch states to this copy's latch. Since a MSFF slave latch has the entire clock high phase to propagate the slave latch feedback signals, the added capacitive loading on the **nha** node does not affect the circuit timing. Consequently circuits using the TRSCMSFF have full (commercial, unhardened) speed performance, except for slightly longer local routing.

Operation of the TRSCMSFF group, comprised of A, B, and C copies, is shown in Fig. 3.8. Fig. 3.8(a) describes a series of three FFs and the associated combinational logic between them. The first block of combinational logic has a delay ($T_{DELAY}$) of more than half the clock period ($P_{CLK}/2$), while the second has a $T_{DELAY}$ that is less than half the clock period. Fig. 3.8(b) describes the response of this circuit to a SET. Here, one input (i.e., copy A input **da1**) is driven to the opposite logic level of the other two inputs to demonstrate correction of an SET induced incorrect D input. The effect is to produce an incorrect value on the TRSCMSFF Q output **qa1** at the next clock rising edge.



(a)



(b)

Fig 3.8. Operation of the self-correcting MSFF at full speed with one input driven incorrectly to simulate a state error. The q output of copy A (node **qa1**) is corrected by the majority gate feedback when the flip-flop slave latch is non-transparent, i.e., in the clock low phase as shown. This error still propagates through a combinational block with high delay (**qa2**) but not through one with low delay (**qa3**)

When the flip-flop slave latch goes from the transparent to the latch closed or feedback mode, the slave latch feedback voter corrects the data based on the majority of the latch states. Then the **qa1** output transitions to match copies A and C in the low phase of the clock signal **clk** (the A and C copies are both 1 in the first clock cycle and 0 in the second clock cycle).

Not correcting in the feed-forward path, i.e., the master latch of the TRSCMSFF, ensures no timing impact and saves routing, but does have an impact on how corrections are performed. For logic paths shorter than $P_{CLK}/2$, the corrected copy is sampled at the next stage, i.e., node **da3** in Fig. 3.8(b) is correct at the next stage D input. However, timing critical signals may not be corrected until further stages in pipelined logic. For signals where the logic delay between pipeline stage FFs exceeds $P_{CLK}/2$, where $P_{CLK}$ is the clock period, uncorrected data, node **da2**, propagates through the next stage FF (see Fig. 3.8(b)) and is voted correct on clock low, as in the first case with node **qa2**. This correction does not arrive at the next FF before the clock edge, so the error then propagates through to the next stage. Thus, if there is another SEU or SET in the other redundant copies, in the same logic cone, within a few clocks, an uncorrectable upset may occur. This error cross section is very small as evidenced by this type of error not occurring at all in broad beam testing (see section 3.3e). Note that this error requires two separate radiation particle strikes on specific targeted areas within less than 10 to 20 ns of each other, which is highly unlikely. Additionally, low actual signal activity factors provide adequate time for correction in most clock cycles. When the clock is continuously low for more than one phase, i.e., gated

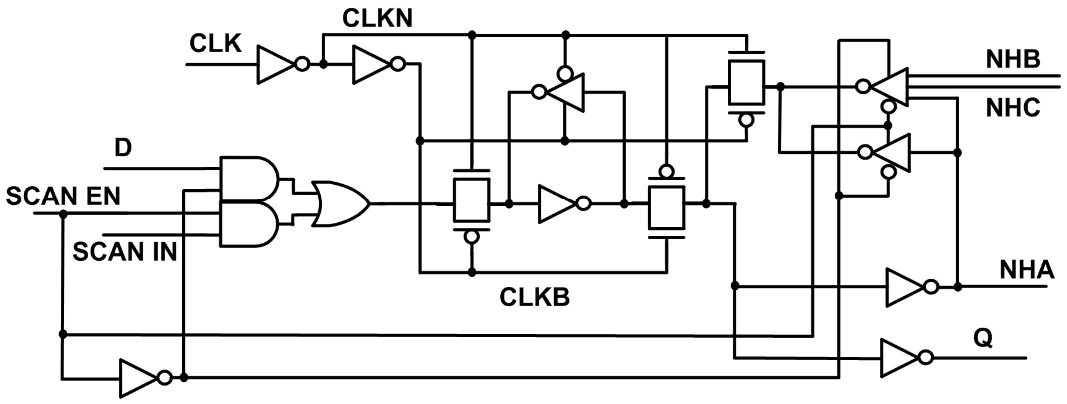


Fig 3.9. Mux-D scan TRSCMSFF. Scan mode disables the slave latch triple redundant feedback, to allow full redundant circuit and voter testing. Timing impact is the same as mux-D scan on a commercial IC.

off, the TRSCMSFF continuously self-corrects SEUs. SETs, of course, do not propagate through pipeline stages when the clock is low.

Scan is the most prevalent design for test technique to detect logic manufacturing defects [60]. However, since the TRSCMSFF in Fig. 3.7 corrects all errors, using scan chains to detect manufacturing errors is ineffective as the defective value is voted out as soon as the clock is driven low. Additionally, a defect acts as a constant error, so when combined with a redundant node corrupted by an actual SEE it produces an uncorrectable error.

A TRSCMSFF incorporating effective scan-based design for test is shown in Fig. 3.9. Here, the slave latch incorporates two feedback paths, selected by the scan mode input signal **SCAN_EN**. When scan mode is selected, a conventional feedback path replaces the majority gate feedback path, decoupling the A, B, and C slave latches. Consequently, errors in the logic or in a MSFF propagate as in a conventional scannable logic circuit. The A, B, and C copy scan chains must be separated, just like the clocks.

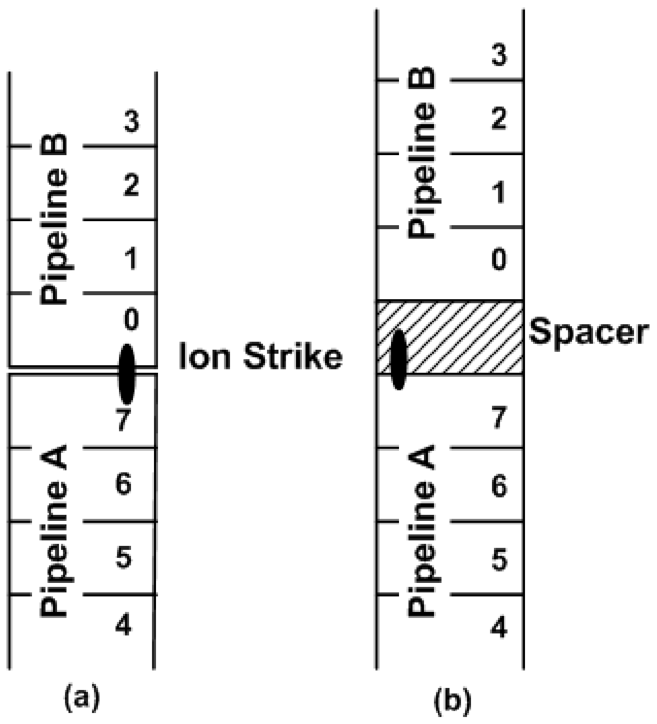


Fig 3.10. Ion strikes on TMR logic. Without a gap, the ion strike in (a) can hit bits in both the A and B pipelines. With a spacer cell inserted as in (b), only one pipeline can be hit as long as the strike is not larger than the spacer height.

The TRSCMSFF design was originally implemented in a macro block containing 8 TRSCMSFFs, which are interleaved to provide the critical spacing. Thus critical MSFF nodes are separated by at least seven standard cell heights (29.4 □m). The MSFF constituent circuits are tightly packed in the same row. Only the voting signals, i.e., **nha**, **nhb**, and **nhc** must be routed vertically. This makes it easy to use a single non-voting version of the flip-flop for synthesis—it is converted into the TMR version later in the process, as described in Section 4.4.

It is possible with adjacent logic cones, e.g., pipelines A and B in Fig. 3.10, for a single ionizing radiation particle to affect both. In Fig. 3.10(a), adjacent gates A7 and B0 collect charge generating SETs that propagate in both logic modules, which could upset two TMR copies. Consequently, in our automated flow, we insert an additional row of spacer cells to ensure that there is one cell height space between adjacent redundant logic copies (see Fig. 3.10(b)) to provide one cell height in separation (at least as much as interleaved temporally hardened MSFFs have).

To avoid the synthesis and APR complexity of using multi-bit cells, the macro block is split into eight single MSFFs with slightly different layouts, each having different vertical routing tracks for the **nha**, **nhb**, and **nhc** signals. They can thus reside in any horizontal multiple of the vertical routing pitch, without interfering with each other. Thus, if less than eight flip-flops are required; logic gates can reside in the space. The original macro block and the separated layouts are shown in Fig. 3.11, along with the wire routing plan. The macro block on the left shows how the copies are arranged with spacers between them, and where the individual layouts on the left can be extracted. The routing plan at the top of Fig. 3.11 illustrates how power wires and gaps for pass through wires are interspaced with voting wires for the different copies.

The decision is postponed as to which layout version of the flip-flop to use until its physical placement by the APR tool is known. In theory, the TRSCMSFF can be placed with the same placement resolution as any other standard cell, but this complicates the CAD flow unnecessarily. For simplicity, and with negligible area impact, the TRSCMSFFs are restricted to be placed only every 30 wire pitches. This allows the custom CAD tool to determine which cell version to use based only on the row number of the cell, knowing that there is a reserved set of wire routes for any particular row and valid location.

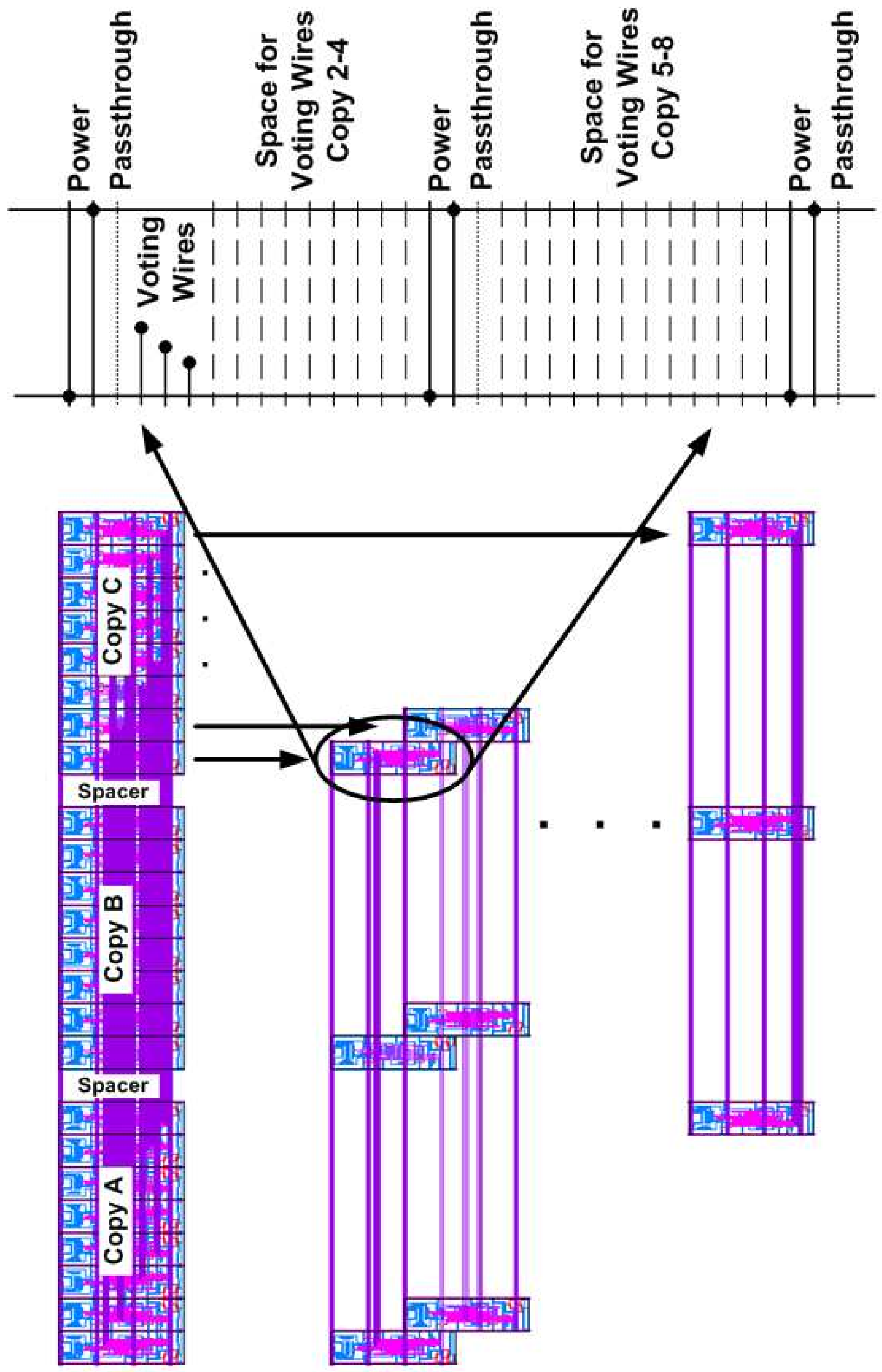


Fig 3.11. TRSCMSFF cell layout. The original macro block is shown on the left, with the individual cells split off and staggered to the right. The wire plan is detailed above, showing the pattern of power wires, voting wires, and a space reserved for wires that may need to pass through metal 3 during routing.

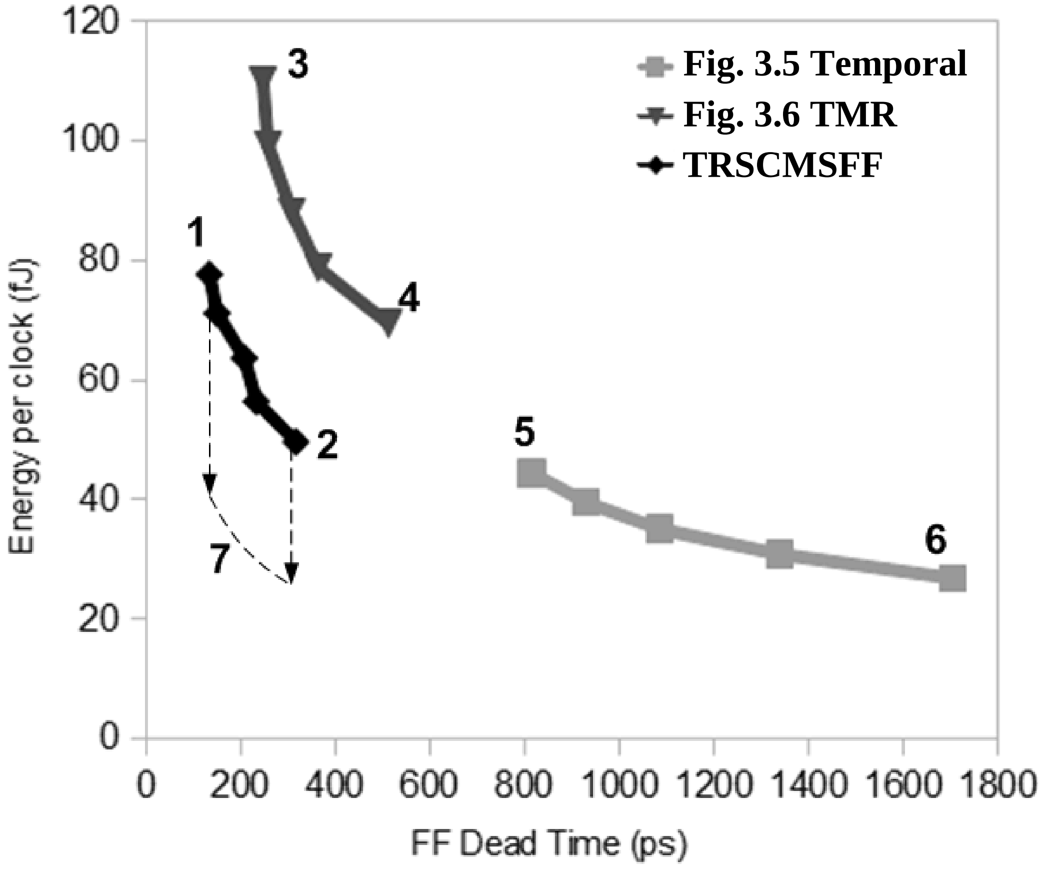


Fig 3.12. Energy per clock vs. FF Dead Time simulation for RHBD FF designs at varying $V_{DD}$. Delay increases as $V_{DD}$ scales down from 1.2 V to 0.8 V in increments of 0.1 V. The TRSCMSFF design maintains a significant delay lead over previous RHBD designs and its power can be scaled to match or improve on the temporal design.

Since the scannable TRSCMSFF version is 58 wire pitches wide this plan has only a minor effect on circuit density, even with circuits that consist of almost entirely FFs. In general, FFs of this design can be densely placed if their width is either an exact multiple of the number of reserved wire tracks, or slightly less.

*D. Power and Delay Comparisons*

For comparison to the TRSCMSFF, simulation models of a published FPGA TMR MSFF with master latch voted feedback [59] and a temporally SET hardened DICE MSFF [10,34] design were created on the same foundry 90 nm process. All simulations were performed with 40% input activity factors. Fig. 3.12 shows energy per clock vs. delay for $V_{DD}$ varying from 1.2 V to 0.8 V. MSFF dead time is defined as $t_{SETUP}$ + $t_{CLK2Q}$, i.e., that wasted by the flip-flop internal delay, for hardened operation, at $V_{DD}$ = 1.2 V. The TRSCMSFF energy per clock is 78 fJ and dead time is 132 ps, as compared to the TMR FF with majority voted feedback in the master and slave latches having 110 fJ and 245 ps, energy per clock and dead time, respectively. The delay and power penalty of using voting in the master latch is thus evident. For comparison, the temporal/DICE hardened MSFF dissipates 44 fJ per clock but has a dead time of 814 ps, including the $t_{SET}$ of 300 ps added to the $t_{SETUP}$ as is required for hardened operation. Since the majority of the power used in a temporal FF is consumed by the delay elements [53], this is a relatively low-power design, using only a single delay element. However, it cannot be clock gated effectively since it is not hardened to clock SETs.

As $V_{DD}$ is decreased, the temporal design slows dramatically due to the increased time lost in the delay circuitry and $t_{SET}$ increase, moving from 814 ps at point 5, to 1704 ps at point 6 in Fig. 3.12. This makes significantly reducing the $V_{DD}$ on temporally hardened designs impractical. SET duration is relatively unimportant in TMR designs however, since they can mitigate $t_{SET} > P_{CLK}$. As shown in Fig. 3.12, the $V_{DD}$ for the TRSCMSFF can be scaled to lower its power dissipation to that of the temporal design at considerably less delay. At $V_{DD}$ = 0.8 V (point 2 in Fig. 3.12), the TRSCMSFF delay dead time is 315 ps, and it dissipates 49 fJ per cycle, comparing favorably to 814 ps and 44 fJ delay and energy for the temporal design at 1.2 V (point 5 in Fig. 3.12).

The TRSCMSFF can be clock gated with no adverse hardening impact (hardening is actually increased since this allows greater local correction time in feed-forward paths, as mentioned). With even a conservative 50% clock activity factor, the TRSCMSFF drops to 39 fJ at 1.2 V $V_{DD}$ with 132 ps dead time. This is less power and less dead time than the temporally hardened MSFF design, which cannot be clock gated. However, the combinational logic power is approximately tripled using the TRSCMSFF. While clock gating and supply voltage scaling affect this power as well, actual circuit power consumption is dependent on the logic function, ratio of sequential elements to combinational logic, and how often the clock is gated. Nonetheless, high-speed logic, e.g., clocked at greater than 200 MHz, will always favor the TMR approach. $V_{DD}$ scaling and clock gating will allow high performance TMR logic to approach the power of temporally hardened circuits when run at similar reduced performance.

## *E. Radiation Testing*

ICs using the TRSCMSFF have been tested with heavy ion broad beams at Texas A & M University (TAMU) and Lawrence Berkeley Laboratories (LBL), as well as with protons at LBL. The TMR logic implements a pipelined built-in self-test engine (BIST) that controls either a Dual Redundant data path logic or an

RHBD cache, implemented on the 9SF and 9LP trusted foundry 90 nm bulk CMOS fabrication processes.

The tested BIST design used hand-built schematics instead of synthesized, and the layout was semi-automated. The design used the initial non-scan design and did not contain a spacer between redundant copies of the logic. Due to speed paths in the hand design, the BIST engine can only operate at speeds up to 250 MHz. Several tests were run that verified the functionality of the BIST engine while in the beam, but the more significant result is that the BIST engine did not fail in recording and reporting the data from the tested circuitry in cumulative days worth of beam time.

There were, however, a few spurious parity check errors reported, where the TMR test circuitry reported an error, but the cache data was clean. This is attributed to the lack of spacers in the test circuitry, which allowed redundant copies to be affected by the same ion strike (as discussed in section 3.3c and Fig. 3.10). The parity check circuitry is the most vulnerable to this type of error because it consists of a large XOR tree that fills the 8 cell height stripe in each copy. If a single bit is struck anywhere in the XOR tree, it changes the single bit parity result. Thus, striking individual bits on opposite sides of the same logic in different pipelines creates an error in the same output bit of data for both copies, which then vote out the third copy.

Due to hand placement, most other circuitry on the tested chip does not have logic trees with vulnerable nodes on both the top and bottom of a layout stripe within a single combinational path, and the low maximum frequency means that almost all combinational paths can complete within the clock low phase, which ensures that these errors are always corrected if they do not immediately combine into the same stored bit (as discussed in section 3.3c and shown in Fig. 3.8). However, future synthesized designs running at faster clock frequencies are more likely to have both of these features combined in the same path, and are thus more likely to require spacers that prevent ions from crossing between copies.

Heavy ion tests at TAMU (on the 9LP design) used 15 MeV N, Ne, Ar, Cu, Kr, and Au ions at angles ranging from 0° (normal incidence) to 72° and effective linear energy transfer ($LET_{eff}$) of 1.4 to 221 MeV-cm$^2$/mg. Ions were targeted in parallel and normal pipeline directions in the angled incidence tests. The unhardened PLL was shielded during testing. All tests were performed at a 100 MHz clock rate with $V_{DD}$ = 1.45 V. This voltage was required for reliable cache operation and is worst-case for charge collection and single event latchup. Only one uncorrected soft failure was recorded, using Ne at LET = 2.9 MeV-cm$^2$/mg. It occurred in a TMR register file, not in the TRSCMSFF protected circuitry. We believe this error was due to a hit on a TMR register cell that contained a manufacturing stuck-at fault in one redundant copy. However, since the tested design was the initial design without a non-voting feedback path enabled in scan mode, there is no test that can confirm this.  Over one million soft errors were recorded in the circuitry (cache) controlled by the TMR test engine during the tests, providing confidence that the tests and TMR logic was fully exercised.

Heavy ion tests at LBL (using 9SF test chips) used B, O, Ne, Ar, and Cu ions with normal incidence LETs of 0.89, 2.19, 3.49, 9.74, and 21.17 MeV-cm$^2$/mg at angles ranging from 0° to 70°. Testing was performed at 100 MHz and 200 MHz at $V_{DD}$ = 1, 1.2, and 1.4 V using the unhardened PLL (foundry IP) which was shielded from the ion beam. No errors in the TMR test engine were recorded.

Proton testing used beam energies of 49.3 and 13.5 MeV. Testing was primarily performed at 60 MHz using the PLL bypass mode, since the PLL is difficult to shield from proton upsets and errors here at the clock root could propagate to all TMR clocks. The total fluence was $5(10^{11})$ protons/cm$^2$ for most tests. No upsets were measured in the TMR circuits in any proton tests, which were performed at $V_{DD}$ = 1.0, 1.2 and 1.4 V.

## 3.4 LIBRARY ABSTRACTION AND CHARACTERIZATION

Although the cell library used here is designed in Cadence, the synthesis and APR tools cannot directly read the cadence database. Instead, the cells need to be converted into a format that the tools can read. First, the layout needs to be abstracted and a .lef file generated. This file tells the APR tool the dimensions and location of areas in the cell where there are blockages that prevent routing, as well as the location of pins. It also handles things like cell pitch and specifies where cells can be placed.

Second, the cell schematics (updated with proper parasitics) are used to create a .lib file. This file describes the functional logic of each cell, and the delay as a function of input slope and output load for each output pin. The .lib file is used in both the synthesis tool and the APR tool to calculate delays and perform optimization.

### *A. Abstract Generation*

The abstract tool comes as part of Cadence, and can read the libraries directly. However, for the 90nm IBM process used here, there were some issues with the technology file in our first attempts to create an abstract. To fix this, a hand-edited tech file is used. Since this file is not entirely compatible with other tools, the library is copied into a completely separate working directory and the abstract program is only allowed to manipulate the copy, not the original. Each time the cell library is copied over, the modified tech library has to be reattached before cells can be processed.

The abstraction process then proceeds in a fairly standard manner, with the exception that TMR cells are separated into their own section and "site" in the output file. The output file is modified at the end to give this site a different cell step of 9.6 microns, to match the power and voting wire plan discussed in section 3.3c and shown in Fig. 3.11. This is necessary to ensure that the APR tool lines up the power and voting wires when placing the cells.

### *B. Cell Extraction*

Although the characterization process discussed below can be run on any netlist, more accurate results are obtained if the netlist is updated with extracted parasitic values from the layout. Calibre PEX was used to extract the annotated netlist from the layout. However, the extraction tech file that properly recognizes annular gates does not work with recent versions of Calibre, so all of the cells need to be extracted with an older version of Calibre. Although this seems straightforward, it is difficult to script within Cadence. Instead, a custom CAD tool was created to run batches of cells through PEX. This tool grabs the proper files from the temp directory, created from running more recent versions of Calibre LVS or DRC on the cells, then sets up PEX and runs it on each of the cells. Finally, it combines all of the netlists into a single file, to make for easier import into the characterization tool used in the next section.

## *C. Library Characterization*

In order for the synthesis and optimization tools to work with a library, they need to know the timing of signals passing through each cell in the library. The characterization tool runs simulations on each input/output path in order to determine the speed at which signals propagate, then condenses this data into a .lib file for other tools to use. For this paper, Encounter Library Characterizer (ELC) was used for this process.

Although this tool is easy to set up and use, it relies heavily on gate recognition algorithms, which cannot handle a TMR flip-flop. It does not understand the voting construct in the slave feedback, and since simulations are typically run by changing one signal at a time, it is very likely to have an input signal voted out as it tests permutations. The solution to this problem is to characterize only a single-redundant version of the flip-flop, with the slave latch voting circuitry removed and the non-voting scan feedback as the only path. Since the slave latch has very little impact on the overall flip-flop timing, this change results in accurate numbers without a complex setup. This same cell is then used in the synthesis and APR tools as a stand-in cell until it is converted into the full TMR cell later in the process.

## 3.5 CHAPTER SUMMARY

The design of a RHBD library can involve the use of many different hardening techniques. TID and SEL hardening is required in every single cell to prevent excessive leakage and latchup issues. SEE hardening in the flip-flops requires careful consideration of the power and speed penalties of varying techniques. Once these steps are complete, it must be analyzed and characterized before it can be used in the synthesis and APR tools discussed in the next chapter. The end result can be improved with careful consideration of how the synthesis and APR tools operate, as well as the strategies used in hand placed layout.

# CHAPTER 4

# AUTOMATED TRIPLE REDUNDANT WORKFLOW

Once the cell library is abstracted and characterized into .lef and .lib files, the library can be used in synthesis and Auto-Place and Route (APR) tools. However, there are no commercially available tools that can properly handle the conversion of single redundant VHDL code into TMR blocks using this library design. For this, a set of custom CAD tools and a specific workflow was designed to automate the process as much as possible. These custom tools work together with commercial synthesis, APR and timing analysis tools to create a hardened block from VHDL code that doesn't need any awareness of the TMR checking.

## 4.1 CUSTOM CAD TOOLSET

Since the workflow used in this process uses several different tools from different vendors, a set of custom CAD tools was created during this project in order to ensure these tools work together properly. This entire workflow is then organized into a set of directories that contain setup files and working directories for all of the tools. The file structure for the toolset is shown in Fig. 4.1. Note that each CAD tool has a "reference" directory underneath it, as well as directories for each VHDL block that is processed. The workflow initialization script (found in the base directory) creates these directories automatically, and copies over each of the scripts and setup files used by the tools. Since the scripts often require the name of the VHDL module be included in various commands, the module name is defined at the top of all scripts and this variable is used

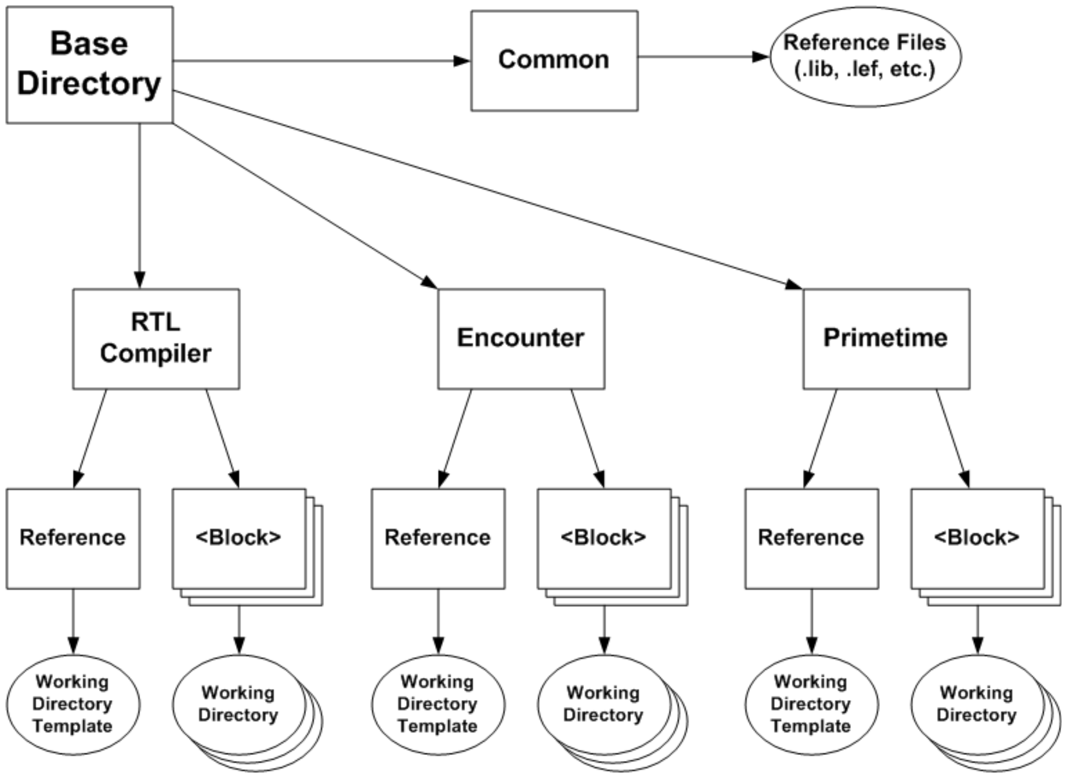


Fig. 4.1. File structure used during the block level synthesis workflow. Each tool and block has their own working directory to maintain separation between runs and allow individual tailoring of blocks if necessary, while reference files for the library are stored in a common location.

throughout. Then, during initialization, the initialization script modifies this line to the new block name as it copies over the files. Since each block has its own run directory for every tool, files in these directories can be modified for block-specific commands if necessary, without interfering with other blocks.

Once all the files and directories are created, a series of "walkthrough" scripts are run in order to guide the user through the steps necessary to run each tool. Each walkthrough tool has a Graphical User Interface (GUI) that takes the form of a list of steps and commands, where each step that can be automated is a button and each step that must be run inside the tool is either an entry that can be copy and pasted or directions to a menu command that can be selected inside the tool.

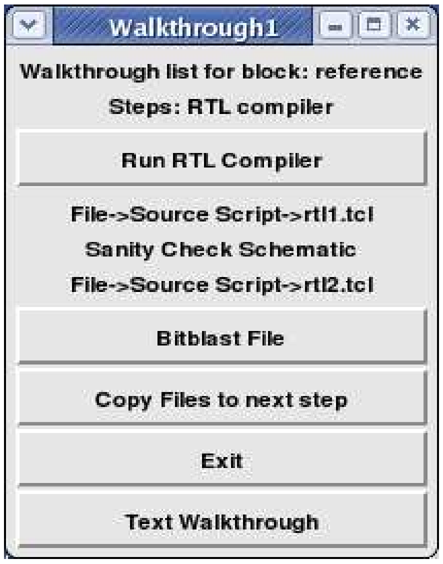
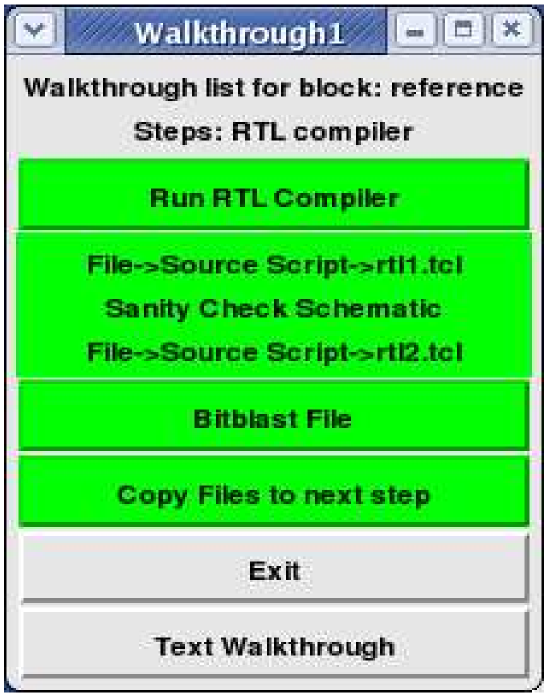


Fig 4.2. Walkthrough GUI for the synthesis step. Actions and scripts that can be run by the GUI itself are presented as buttons, while actions that must be taken in the tool are presented as a list.

Fig. 4.2 shows the walkthrough GUI for the synthesis tool, which is one of the simplest sections of the workflow. Note that adding and modifying sections from the walkthrough is easy, since the GUI is run by a simple perl script that spawns additional tools from the command line as needed. Once the user has designed a new tool, several lines of code are all that is necessary to add the new tool into the walkthrough GUI.

As the workflow progresses, the walkthrough GUI will also keep track of the actions taken. As buttons are pressed, they change their background color to green as a reminder of what step the user is currently on, and this also changes the color of all the labels above the button (See Fig. 4.2). This color change helps keep track of which steps have already been done, to ensure that nothing is skipped by accident.



Walkthrough list for block: reference

Steps: RTL compiler

File->Source Script->rtl1.tcl

Sanity Check Schematic

File->Source Script->rtl2.tcl

Fig. 4.3. GUI status indicator for the synthesis step. Since each button before the "Exit" button has been pressed, the steps have had their background changed to green.

## 4.2 SYNTHESIS

The first step in producing a TMR block is synthesis, which takes the VHDL code and converts it into a netlist that uses only the cells present in the cell library. For this process, we used Cadence RTL Compiler (RC). Both the original VHDL code and the synthesis output are non-redundant circuits, and thus need no knowledge of the TMR scheme in order to work properly. Consequently, standard soft intellectual property (IP), such as soft-cores, can be used and the synthesis methods are almost entirely the same as for non-TMR circuitry. Since the synthesis is non-redundant, the TRSCMSFF version without the voting slave latch feedback path is used, as mentioned in section 3.3c.

Each block has very specific timings that must be met in order to reach the desired operating frequency, and providing this information to RC is one of the most finicky tasks in this section of the workflow. RC can read timings from .sdc timing files or the timing paths can be set directly in the .tcl scripts, but figuring out what the actual timing should be can be difficult. If the timing is too lax, RC will stop optimizing the path once it meets speed, and the block will run slower than it should. If the timing is too tight, every path will be optimized towards speed and not power, and the entire block will be filled with low-$V_t$ transistors, causing a significant increase in power usage.

Since for the test design we care more about speed than power, the first attempt at setting up timings was to give all paths the same overly-tight timings in the script. Later attempts with .sdc files also worked, but since the .sdc file was generated at the top level, there were several paths that ended up not being correctly constrained. Since these paths went through several blocks, the slack they had at the top ended up being given to each individual block, resulting in the path being significantly under-tuned. While this can be fixed intelligently with manual adjustment on these paths, the easier fix for the test design was to tighten up all of the timings and live with the increase in power usage.

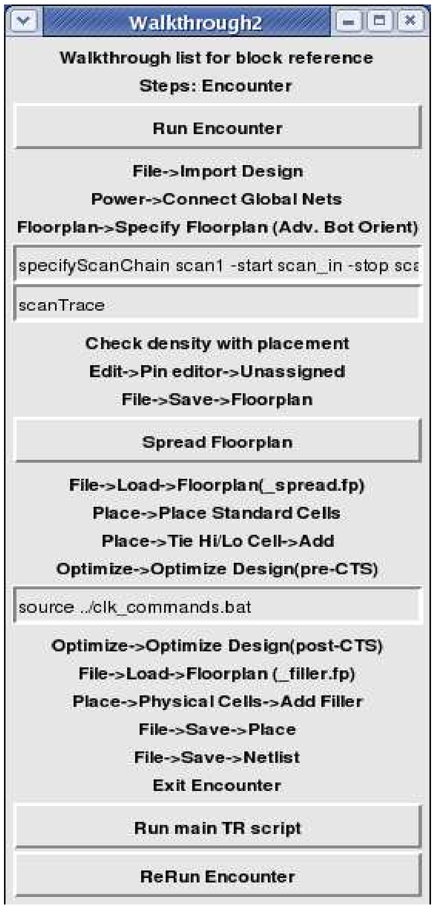
Once the netlist is fully synthesized, the final steps in this workflow section adjust it for use with the next set of tools. The final lines of the tcl script used by RC flatten the netlist. This ensures a unique, one-to-one relationship between an instance in the netlist and an instance placed using the APR tool. This correlation ensures that when the TMR stand-in flip-flops need to be replaced with the fully

TMR flip-flops, they can each be changed individually to their proper cell version with the correct wiring (as discussed in Section 3.3c). The “uniqueify” command could also be used for this, but a completely flat netlist is easier to parse and simplifies the custom triplication tool (discussed in Section 4.4).

Additionally, an issue arose with instance names in the netlist. Encounter’s placement file adds extra characters (some of which are non-printable) to its placement file when it finds that an instance is part of an array (i.e. its name is followed by brackets and a number). Although the triplication script could be modified to parse these extra characters, it is easier to remove instance arrays during synthesis. This process of converting arrays to single instances is called “bitblasting.” Using the “bitblast constants” command in RC ensures that each instance in the netlist is defined separately, even when part of an array. Then a separate CAD tool processes the netlist to bitblast the instance names.

## 4.3 PLACEMENT

The next section of this workflow is placement and optimization to create a single-redundant placement file. For this, the workflow uses the Encounter tool. The placement section of the Encounter Walkthrough GUI is shown in Fig. 4.4. This GUI can be further separated into the floorplan section and into the placement and optimization section. All of these commands are processed with the single redundant version of the netlist.


Walkthrough2
Walkthrough list for block reference
Steps: Encounter
Run Encounter
File->Import Design
Power->Connect Global Nets
Floorplan->Specify Floorplan (Adv. Bot Orient)

```
specifyScanChain scan1 -start scan_in -stop sca
scanTrace
```

Check density with placement
Edit->Pin editor->Unassigned
File->Save->Floorplan
Spread Floorplan
File->Load->Floorplan(_spread.fp)
Place->Place Standard Cells
Place->Tie Hi/Lo Cell->Add
Optimize->Optimize Design(pre-CTS)

```
source ../clk_commands.bat
```

Optimize->Optimize Design(post-CTS)
File->Load->Floorplan (_filler.fp)
Place->Physical Cells->Add Filler
File->Save->Place
File->Save->Netlist
Exit Encounter
Run main TR script
ReRun Encounter

Fig 4.4. Walkthrough GUI for the placement section of the workflow. The more complicated commands that need to be run inside the tool's command line are stored in entry boxes, allowing easy copy and pasting.

Generating the initial floorplan uses a standard set of commands for this tool. The major difference is that the height of the floorplan needs to be a multiple of 9 cell heights in order for the custom CAD tools to convert it properly. Once the dimensions are chosen, the scan chain is defined, power settings are applied, and the floorplan is saved.

The final floorplan step is to run a custom tool that generates "spread" and "filler" versions of this floorplan. These files are based on the initial floorplan, but the height is expanded by a factor of three, and the valid placement rows are grouped in sections that represent where they will be in the final triplicated

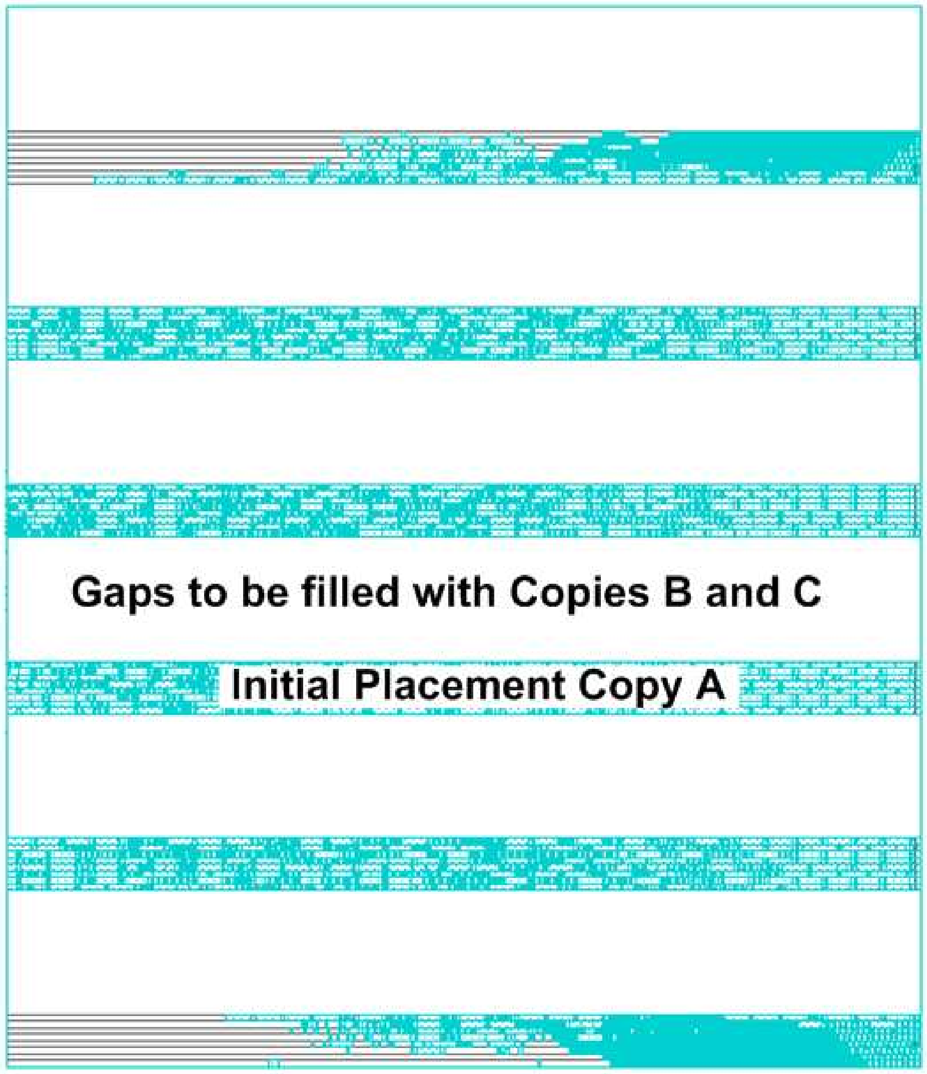


Fig 4.5. Initial cell placement. Gaps in between cells provide space for the two other redundant copies. The extra spacing ensures that the tool comprehends the correct routing distances.

design. The “spread” version has 8 rows per section and is used for placement and optimization of standard cells. The “filler” version has 9 rows per section and is loaded after the design is fully placed and optimized to allow the placement of filler cells in the 9th row of each bundle. This row is thus guaranteed to be filled with only filler cells, providing the necessary spacing between different copies of the triplicated logic.

The circuit is then placed as normal in the allowable placement stripes, and the circuit is optimized using the standard Encounter commands. The resulting placement is shown in Fig. 4.5. Since the gaps in between the placement stripes are the correct size to be filled with the two additional copies, the routing lengths between cells are correct and can be used during optimization to estimate the wire load properly. This added wire load is critical in order to get reasonable optimization out of the tool. Note that clock tree synthesis and post-clock optimization (including hold time fixing) are part of this process, but there is no easy way to do post-route optimization because routing only occurs with the full triplicated design.

Once the design is fully placed and optimized, the filler floorplan is loaded and filler cells are added to create the final single redundant placement. This placement file is then saved, along with the matching optimized netlist. These files are then fed into the custom triplication CAD tool discussed below. The encounter program is then closed, since its state is based on the single redundant version of the netlist, which cannot be converted to triple redundant without restarting the tool.

## 4.4 TRIPLICATION

In this transition step, the floor plan file, the cell placement file, and the netlist are all modified to be TMR using a custom CAD tool. Because this occurs only on the save files, no integration with Encounter is required.

*A. Floorplan Parsing*

The floorplan file modifications are the most straightforward of the three files. Most of it is irrelevant to the triplication process, with only three sections that need adjustment: the headers that determine block size, the valid placement row definitions, and the wire routing definitions.

The first modifications take place in the headers and consist of adjustments to the floorplan size. Since the "spread" floorplan is used as the basis for this file, the size is already expanded 3x. However, because of how the power wires are set up, power vias need to exist at the edge of the chip. Encounter's default settings do not place power vias outside the block boundaries, so it will shift power vias slightly on the edge. Moving these vias shifts their alignment to the vias already present in the standard cells, which causes DRC errors. Although this can be changed in the tool, it is easier to have the triplication script extend the borders of the block by 0.2 microns to allow plenty of space for power vias inside the block dimensions.

Additionally, since the block is in a multiple of 9 rows high, there is a chance that the top row will end on an N-well boundary with only filler cells connected to it. Since ending on substrate is preferred for our design, the CAD tool detects this condition and decreases the size by 1 row. It then sets this reduced height as a maximum for the placement file parsing discussed below, so that any cell higher than this maximum is removed. This modification ensures a consistent top border for the block.

To convert the allowed placement rows to the triple redundant version of the floorplan, most of the previous section of the file is discarded and the section is regenerated. Since the dimensions of the block are known and the rows need to completely fill it, it is easier to recreate this section than modify it. Rows are generated with both the “core” site and the “coreTR” site with the proper orientation and length such that the entire block is filled. These rows fill in the gaps present in the “spread” version of the floorplan with valid locations for the B and C copies of the design.

The final floorplan section that needs to be modified is the routing section. This section defines the allowable wires and metal pitch inside the block for each metal layer. If this section is left as is, only the bottom third of the block would route correctly. To convert this section to its TMR version, simply lengthen the definition for each vertical wire and add three times the number of wires for the horizontal metal layers.

### *B. Netlist Parsing*

The current implementation of the netlist triplication is surprisingly simple. Since the netlist is flattened during its exit from the synthesis tool, parsing only needs to consider each wire and instance on its own. This means that large files can be parsed without additional overhead, since only a few lines are processed at a time. As the CAD tool traverses the file, it looks at each input, output, wire, and instance statement. All wires and nodes are automatically triplicated, but instances are first checked against a list of TMR cells. Standard cells are triplicated, but TMR flip-flop cells are instead converted into their respective TMR version.

However, this conversion is a two-pass process. Due to legacy issues, the netlist is parsed before the placement file, so there is no placement information available when the netlist is parsed. Thus, there is no knowledge of which cell name to use in order to properly align the voting wires. Instead, an intermediate netlist is generated with placeholder cell names. A final pass after the placement file has been parsed looks for these placeholder names and references their location in the placement file in order to find the final instance name in the finished netlist.

## *C. Placement Parsing*

The placement file used by Encounter consists of several different parts to define the relationship each cell has in the netlist hierarchy. Parsing this file line-by-line is impractical, so each section is first read and stored in an array, then these arrays are modified as needed. Similarly to the netlist parsing, each placement entry is checked against a list of TMR flip-flops. Combinational cells are triplicated with their copies moved up 9 and 18 cell heights, while TMR cells are left in the same location but converted to the proper TMR cell name. At this point, the location of each TMR cell is known, so its name can be converted to align the voting wires based on its row in the design.

In the current design, TMR cells also have their orientation adjusted to match the way they fit together when they were first designed. Since the orientation of the original design flipped each sub-cell instead of flipping the entire cell, it needs

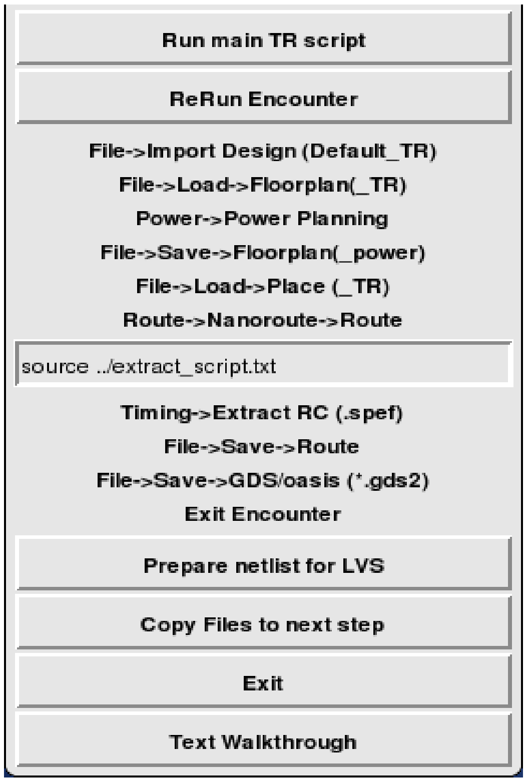


Fig 4.6. Walkthrough GUI from the routing section of the workflow.

the cell to have the default “R0” orientation so that it matches properly. This can cause some confusion and extraneous warnings when the placement file is used in Encounter, but since Encounter doesn’t need to touch the placement after this step is done, these warnings can be ignored.

## 4.5 ROUTING

After the triplication tool modifications are complete, the block’s logic is fully TMR and TRSCMSFFs are used throughout the design. (Note that both self-correcting and non-self-correcting transparent latches can also be used.) Since the only parts of a circuit that need to maintain critical node spacing against charge collection are the transistor drains, the wires have no radiation hardening restrictions and standard routing methods may be used. Encounter can thus be

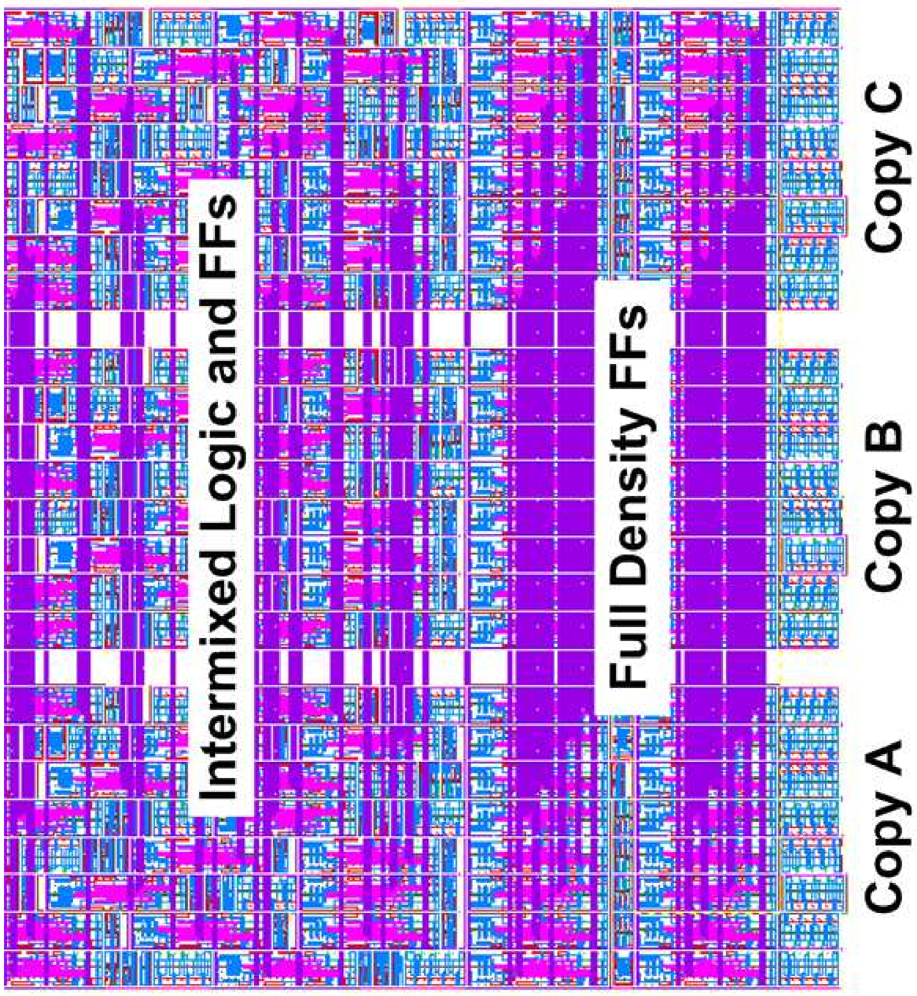


Fig 4.7. A section of the final layout. The right side is fully filled with TRSCMSFFs to re-create the macro block shown in Fig. 3.13, while the left is intermixed with combinational logic while retaining density.

restarted with the TMR placements and netlist to route power wires and interconnects. The second half of the Encounter walkthrough GUI is shown in Fig. 4.6.

After the placement file is loaded, the location of each cell is fixed while routing is performed. This also means that post-route optimization is impossible with the current implementation. The final routed design then has its parasitics extracted

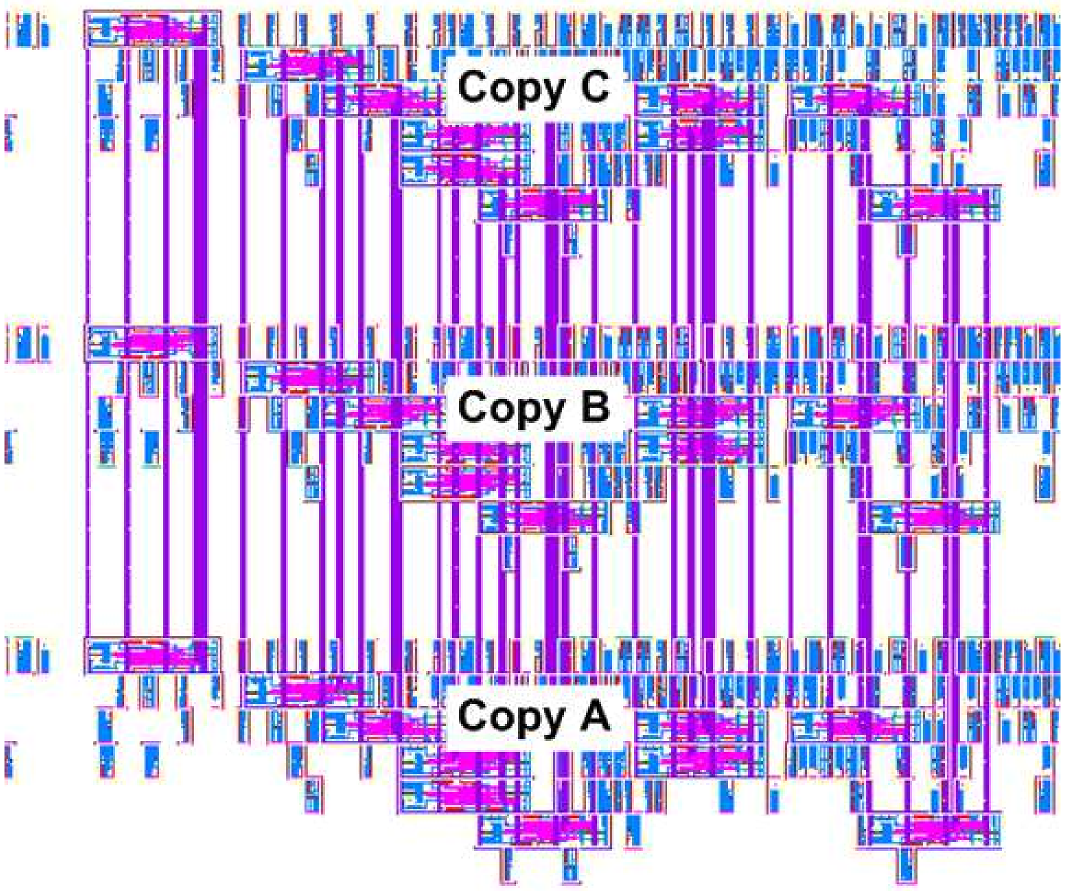


Fig. 4.8. A section of the final layout. Three different copies of the same logic with TRSCMSFFs connecting them together. This section is sparsely populated and omits spacer cells to highlight the pattern of triplicated cells.

for use with the timing tool (discussed in the next section) and the design is saved as a .gds2 file for import into Cadence.

A final series of custom CAD tools are then run to handle interfacing this output with the next steps of the workflow. First, the Verilog netlist is converted to Spice (using the "v2lvs" command) to make it easier for the LVS checking tool to read it. However, we use decap cells as filler for this design, and these need to be added to the netlist. Thus, a second tool is used to count the number of decap cells in the .gds2 file and append the proper number of cells to the Spice netlist.

Finally, a third custom tool is run on the Verilog netlist to convert its bus style into one that is readable by primetime. In the Cadence series of tools (including Encounter and RC) it is valid to define a wire bus one bit at a time, but this causes errors when importing the netlist into primetime. Thus, these single bit definitions need to be merged together into one multi-bit definition.

The final result of this walkthrough is a set of files that can be imported into Cadence for layout checking and final output, as well as a netlist and parasitics file that can be used in Primetime for timing analysis. Example sections of the Cadence output are shown in Figs. 4.7 and 4.8. The right side of Fig. 4.7 consists of fully packed flip-flops, re-creating the original macro block shown in Fig. 3.13, while the left side consists of intermixed flip-flops and combinational logic with no loss in density. Fig. 4.8 is sparser and spacer cells have been omitted to highlight how cells patterns are triplicated.

## 4.6 TIMING ANALYSIS AND VERIFICATION

Although the output from encounter is functionally correct, the difficulty in directly adding the triple redundant version of the TRSCMSFF to the .lib file makes generating the final timing analysis problematic without using a separate tool. Additionally, a separate timing tool is often used to provide more accuracy and standardization across a project. Thus, a final custom CAD tool was added to the workflow to handle timing analysis.

Post route timing analysis uses Primetime, and its custom tool GUI can be seen in Fig. 4.9. The standard project timing flow can be used almost unaltered on the triple redundant netlist annotated with parasitic routing data. However, the netlist

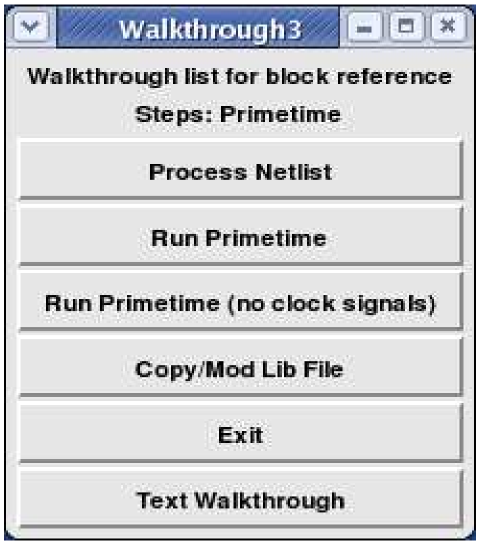


Fig 4.9. Walkthrough GUI from the timing verification section of the workflow.

references the triple redundant versions of the TRSCMSFF which have no .lib file entry to give its timings. There are two ways to fix this issue. One involves the more complicated library characterization setup dicussed previously, in order to add these cells to the .lib file. However, it is simpler to append a supplemental netlist to the end of the routed one, which links each TRSCMSFF to three copies of the stand-in characterization cell. This adjustment is performed by the "setup netlist" button in the GUI. The "copy/mod lib file" button adjusts the lib file for further synthesis in a case where the VHDL is aware of the triple redundancy. This will be discussed further in section 5.2.

As a final step, the final netlist is put through a formal verification tool and simulated with logic vectors to ensure that it functions the same as the HDL input file. Since this verification step is performed on the TMR version of the circuit, it will catch errors introduced by the custom CAD tool logic and can also provide confirmation of the static timing analysis results.

## 4.7 CHAPTER SUMMARY

An automated triple redundant workflow is difficult or impossible with standard off-the-shelf CAD tools. However, using a custom CAD toolset, these tools can be integrated into an effective design process. Fig. 4.10 shows the detailed block diagram for this process, including the tools used for this specific implementation of it. This workflow utilizes single-redundant, unhardened versions of the logic for synthesis and placement, then converts the logic into TMR for routing and timing analysis. Although the tools used in this example were RC, Encounter and Primetime, the workflow can be modified to work with most commercial tools. However, though this block itself is TMR, eventually it will need to communicate with either the outside world or sections of the chip that are not hardened in the same manner. These connections will be discussed in the next chapter.

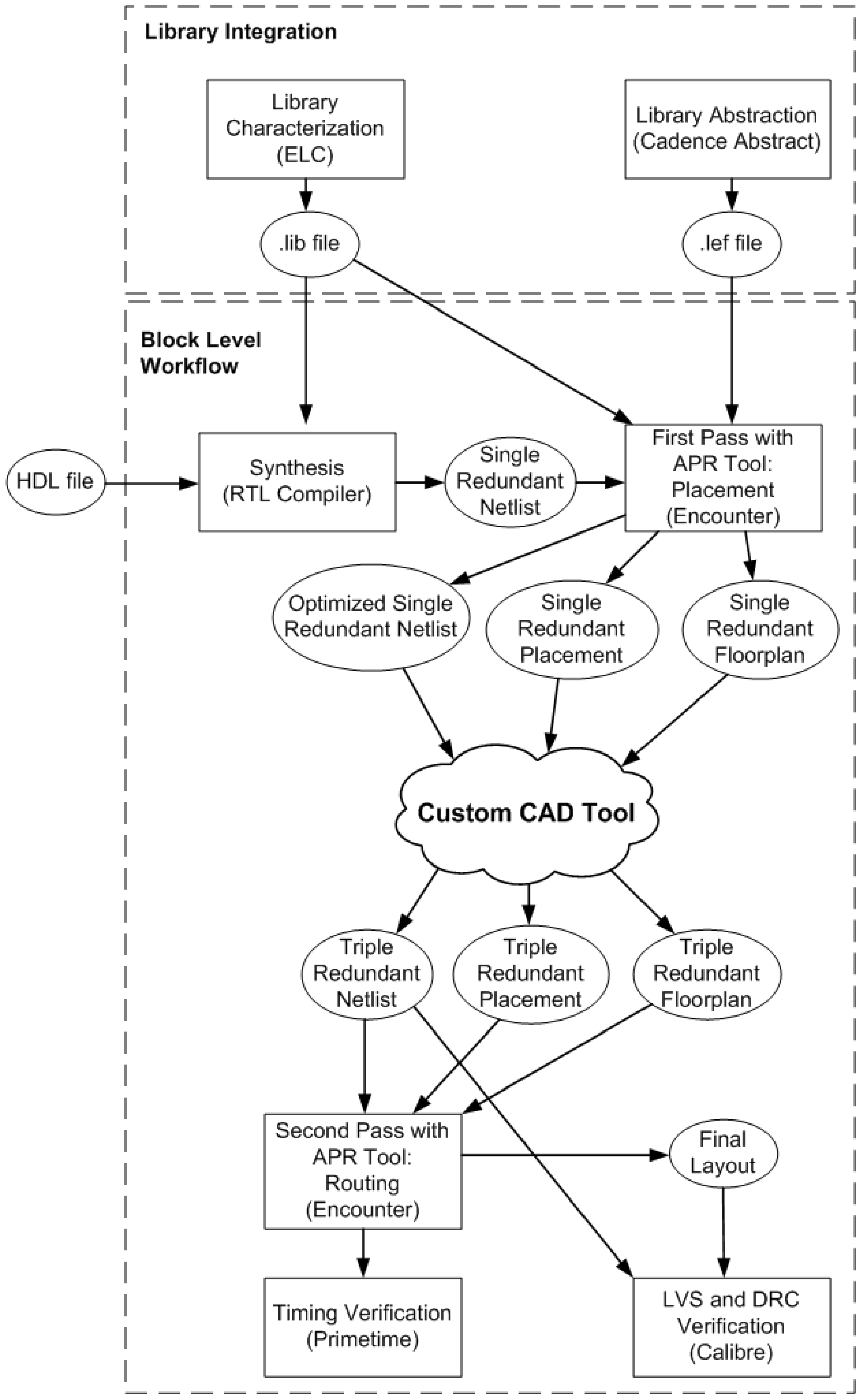

Library Integration
Library Characterization (ELC)
Library Abstraction (Cadence Abstract)
.lib file
.lef file
Block Level Workflow
HDL file
Synthesis (RTL Compiler)
Single Redundant Netlist
First Pass with APR Tool: Placement (Encounter)
Optimized Single Redundant Netlist
Single Redundant Placement
Single Redundant Floorplan
Custom CAD Tool
Triple Redundant Netlist
Triple Redundant Placement
Triple Redundant Floorplan
Second Pass with APR Tool: Routing (Encounter)
Final Layout
Timing Verification (Primetime)
LVS and DRC Verification (Calibre)

Fig 4.10. Fully detailed block diagram listing the tools used for each step and the important files that are passed between programs.

# CHAPTER 5

# INTEGRATION WITH NON-TMR CIRCUITS

Although the TMR hardened workflow operates well on its own, it will eventually need to communicate with either the outside world or with sections of the circuit which are not hardened in the same manner. Interfacing with external circuits without the loss of hardening requires careful consideration of how errors can be introduced and which methods are effective at preventing them. The exact structure of this interface depends on the type of hardening present in the other domain and whether that domain is within or outside of the chip.

## 5.1 INTERFACING WITH OFF-CHIP LOGIC

If the entire chip is to be hardened using this TMR method, the workflow can often be used with little alteration. The commercial tools have the ability to use input and output pins in their designs, so the only major change is to add pad cells to the library file to allow for the creation of a pad ring. However, inputs and outputs from the chip are rarely triplicated, so effort must be made to ensure that errors do not appear at the input or output pins. This hardening can be done by modifying only the pad cells themselves.

For output pins, this hardening is done using 3 balanced drivers for the output pad, each of them large enough to drive the pad on its own and each spatially separated from the others. Thus, if one copy of the signal entering the output pin is affected by an SET, the other two will overpower its drive strength and drive the pin to the proper value. This driver contention can waste power and shorten transistor lifetimes, but errors are expected to be rare enough that this effect is negligible. Additionally, the output drivers need to be large enough that their output capacitance makes them SET-immune. This immunity is often the case with standard output pad drivers as well, so it usually requires very little additional design work.

For input pins, the major concern is that the pad is split off into three internal signals as soon as possible. Care needs to be taken to ensure that there is no unhardened node that can be hit to cause errors on 2 or more copies of the signal and that each copy is spatially separated from the others. The input pin itself has a large capacitive value which makes it resistant to SETs, but as soon as this signal is propagated to on-chip transistors it becomes vulnerable.

To integrate hardened pads into the workflow, they need to be treated similarly to the TMR hardened flip-flops. Thus, they should be synthesized with a stand-in, single redundant version of the pad, then converted to the TMR version of each pad during the triplication phase. Small changes need to be made to the custom CAD tools to account for pad ring locations in the floorplan file and to prevent triplication of the output nodes. However, the structure of the workflow remains the same.

Alternatively, the pad ring can be built by hand and the TMR logic created by this workflow used as a sub-block. This requires additional VHDL code to wrap the single redundant logic in a TMR wrapper as well as a standard APR step to wire the pad ring and logic together. This method is also used to connect to non-TMR logic on the same chip, which is discussed further in the next section.

## 5.2 INTERFACING WITH ON-CHIP LOGIC

If this TMR workflow will be used only for part of the logic on the chip, the VHDL needs to be designed to match the output of the workflow. Thus, a wrapper is added around the single redundant VHDL that corresponds to the changes that the TMR workflow makes during the triplication process. In order to maintain the block hierarchy, this wrapper has to represent a precise match of the triplicated signal names and logic. Thus, the wrapper must contain three instances of the original VHDL code and the wires that connect their inputs and outputs to the wrapper interface. The naming of these pins can be defined in the custom triplication CAD tool, but the default is simply appending "A", "B" and "C" to each pin name. Since this block matches the output of the TMR workflow, this wrapper cell can then be inserted into arbitrary VHDL code and used as needed.

Setting up a consistent naming scheme for these blocks is essential to avoid confusion. In the example chip design used in this paper, the single redundant block always ends in the letters "ST", while the TMR version of the same block ends in the letter "C". In theory, the workflow should automatically change this name when it performs the triplication step, but this complicates some of the file parsing. Instead, a combination of hand edits and scripts is used to change the block name on all the output files.

Additionally, care should be taken with the logic to ensure that data enters and leaves the TMR hardened areas safely without risking SEE data corruption. Even if the data is traveling from one hardened area of the chip to another, the interface itself can be vulnerable to SEEs.  An example transition from dual-redundant logic to TMR logic is discussed in the next section.

## 5.3 EXAMPLE CHIP INTERFACES

As part of the testing process for this workflow, it was used in several sub-blocks in the Highly Efficient Radiation-hard Microprocessor Enabling Spacecraft (HERMES) processor design.  This processor is based on the MIPS-32 architecture and designed from the ground up to be aware of hardening constraints.  Since it uses custom code, it can utilize several different hardening schemes for differing areas of the processor.  These schemes fall into three general categories: 1) synthesized TMR utilizing the described workflow 2) synthesized dual-redundant logic and 3) “semi-custom” cache blocks.  The TMR sections of the chip are used primarily to store values and logic that has to be correct at all times, i.e. the architectural state of the machine.  The dual-redundant blocks are “speculative” processor states that can easily be flushed and restarted from a known good state.  The cache is built from hand-placed logic utilizing error correction coding, but it also uses automated routing to connect different sections together.  In practice, the cache and TMR blocks are generated and abstracted at the block level, creating hard macros that are placed on the chip. The dual redundant sections of the chip and the connections between the blocks are then synthesized and placed as a sea of gates surrounding these macro blocks. Partitioning is used to ensure that the A and B dual redundant copies are separated enough that no single ion can hit both of them.

The advantage of this approach is that each area is only as hard as it needs to be and uses dual redundancy for the majority of the datapath pipeline to save space and power. The tradeoff for this is that the VHDL code needs to be completely custom and this code must understand the interactions between each of the hardening domains.

### *A. Dual-Redundant Interface*

Since the dual redundant sections have detectable errors but not correctable ones, it is designed such that all the dual redundant sections can have their data flushed and reloaded without permanent data loss. This mechanism mimics how normal processors handle branch prediction failures. The data in transit through the dual-redundant datapath is speculative, so it is only committed at the end. If it is discovered that an error has occurred, the datapath is flushed and restarted from the last known good state.

When interfacing TMR logic with this scheme, converting from the TMR domain to the dual-redundant domain is straightforward. Two of the TMR redundant paths are sent to the dual redundant logic, and the third path is left unconnected. However, when this data flows the opposite direction, from the dual redundant region to the TMR region, it needs to add an extra layer of redundancy. If one of the inputs were to be fed into two copies of the TMR flip-flop, this duplication of a possible error would allow that input the power to outvote the other one. Instead, each input is converted into its own set of TMR flip-flops. This means that a single logical value ends up being stored in 6 separate storage nodes. Once the value is established in the TMR domain,

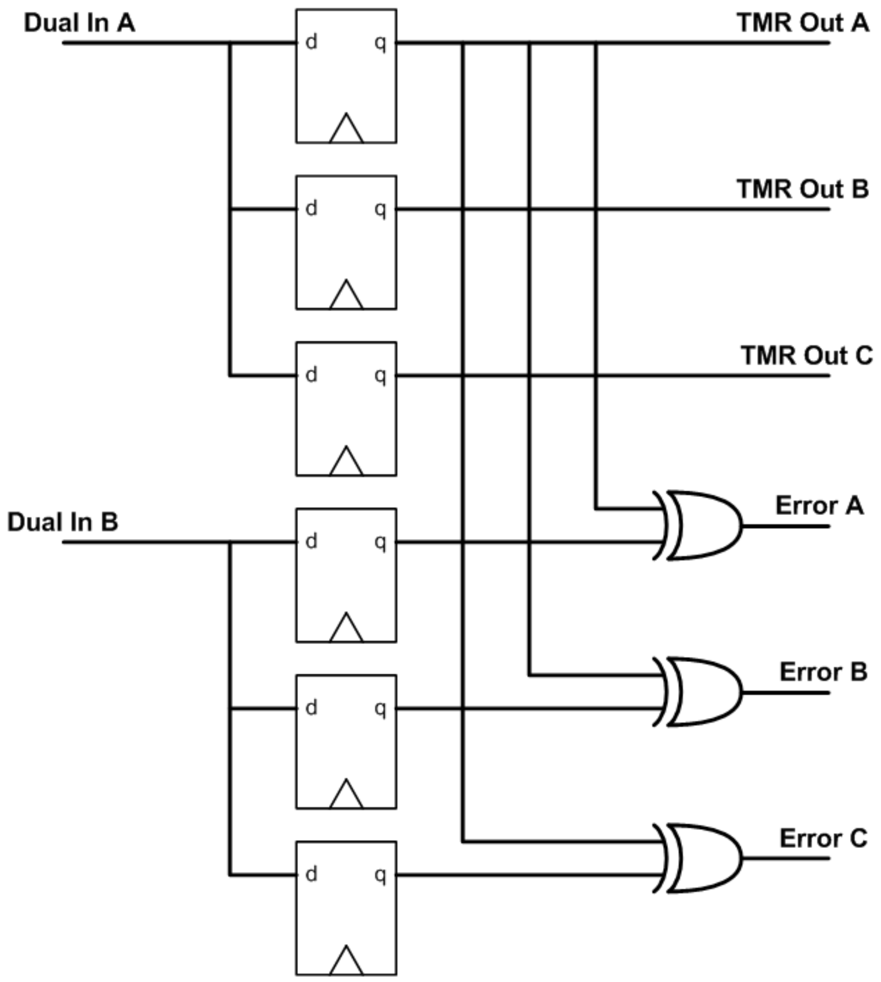


Fig. 5.1. Translation from dual redundant to TMR domains. The dual redundant signals are triplicated and stored in the TMR flip flops, then checked for dual redundancy errors after the inputs are safely stored in the TMR domain.

checking is performed to ensure that there was no dual redundant mismatch before one of these copies is discarded and the other kept. If a mismatch has occurred, the new data is not stored in the next stage and the du al redundant pipeline is restarted. Fig. 5.1 shows a diagram of how this checking is performed.

While storing a value in 9 separate storage nodes during the conversion may seem excessive, running the checks in the TMR domain is required in order to avoid errors in the dual redundant domain. Since there is no easy way to control where the APR tool places the checking logic in the dual redundant domain, the checking logic could easily be placed such that it is too close to one copy of the dual redundant logic, thus making it vulnerable to an ion strike. Additionally, although the VHDL logic might work properly, buffers can be inserted by the APR tool that do not show up in the code itself. If these buffers are inserted after the checking logic but before the TMR region, they can be hit by an ion strike, which corrupts the data. Once the data enters the TMR domain, there is more control over where these vulnerabilities can occur, and these types of errors can be suppressed. Even if the data and the checking circuit are hit in the TMR domain, only one copy will be affected, and it will be voted out in the next stage.

## *B. Cache Interface*

Since the cache has its own hardening scheme that relies on error correction logic, the interface between TMR logic and the cache is almost entirely based on the custom cache hardening. In the example processor, no data is directly sent from the cache into TMR domains. Instead, the interface runs through dual redundant logic and uses the same hardening scheme as the dual redundant interface discussed above. When data is sent to the cache from the TMR logic, it only uses 2 of the 3 outputs. One is used to write the cache, while the other is used to check the data. The data storage in the cache is interleaved to separate vulnerable bits and to ensure that error correction codes can correct any SEUs on the RAM cells that are used to store the data. Hand placement is used to ensure that the write logic is spatially separated to prevent SETs from writing incorrect data into more than one bit.  Single bit SETs on the write logic can be corrected using the same mechanisms that correct SEUs.

## 5.4 CHAPTER SUMMARY

Communicating and interfacing between hardening domains is critical for a chip to work in conjunction with larger structures and other chips.  Off-chip interfacing can be handled primarily with modification to the pad ring cells, and the TMR workflow can be used with very little alteration.  On-chip interfaces can use the workflow as-is, but require more awareness of how each domain must be handled. The conversion between domains can add risk if handled improperly and often requires additional space and timing considerations.  For the example chip discussed, the overhead added for transitioning to the TMR region is significant, and the impact of this overhead will be discussed further in the concluding chapter.

# CHAPTER 6

# CONCLUSION

The TMR synthesis workflow described in this paper provides a low power, high speed method for hardening circuits. Although triplicating the logic and storage nodes also triples the area and power, the fact that this hardening method does not require delay elements and is easily clock gated makes power consumption extremely competitive with temporal hardening methods. Since there are no delay elements in the paths, there are only parasitic speed losses when compared with a standard master-slave flip-flop. This reduced delay gives a huge speed advantage over temporal designs and removes scalability issues that can limit temporal operating frequencies on newer processes.

This workflow relies heavily on commercial software tools for its synthesis and APR functions, but does not directly communicate with them. All of the workflow-specific functions are handled through editing save files, which makes the workflow easy to adapt to different sets of tools. However, since the tools were not designed with this workflow in mind, they bring their own sets of limitations with them.

## 6.1 TOOL AND WORKFLOW LIMITATIONS

One of the biggest strengths and weaknesses of the current toolset used is that placement cannot be adjusted once the circuit is triplicated. This fixed placement ensures good separation between vulnerable nodes by placing identical nodes 8 cell heights away from each other, but it also prevents the use of post-route

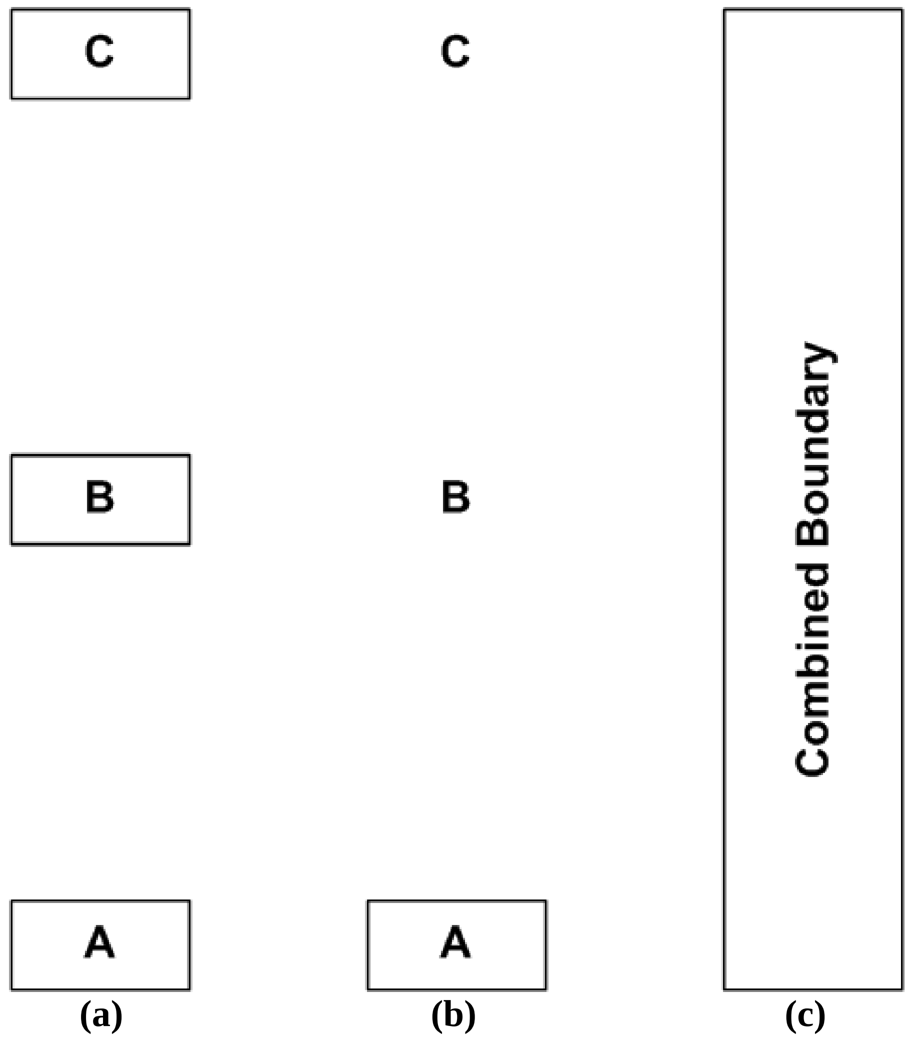


Fig. 6.1. Place and Route Boundary issues. The correct prBoundary layer for the TMR flip flop is shown in (a), however, the tools used require that this layer be a single continuous polygon. Thus, it discards two of the three polygons, leaving just the A boundary, as shown in (b). To prevent having structures without prBoundaries, the prBoundary layer is modified in the standard cell, such that it encompasses all three copies, as shown in (c).

optimization. In theory, such optimization is possible, but the limitations of the tools hinder its use.

## *A. Place-and-Route Limitations*

The most prominent element of the tool limitation shows up in how the cell boundary is defined and used. The cell boundary describes the outline of the cell, and defines the area each cell requires. In the version of the tools used for this paper, this cell outline must be a contiguous polygon, so that the cell is considered one solid object. With the TMR flip flops, however, the cell boundary should consist of 3 separate rectangles that are not directly connected together. Since the tools do not properly recognize this, they create a cell boundary that encompasses these rectangles and the area between them, as shown in Fig. 6.1. When these rectangles are used in the current workflow, they create overlap errors with every cell that is placed in between the three parts of the TMR flip-flop. If placement optimization is attempted once the triplication step is complete, the tool attempts to fix these overlap problems by moving these cells away from the area between TMR flip-flop copies, creating either a dysfunctional design or a badly sub-optimal one. Thus, the placement generated from the single redundant optimization must remain exactly as is, and these overlap warnings must be ignored.

If this cell boundary overlap issue were resolved, optimization after triplication would still not be straightforward, because by default the tool would intermix the separate copies of the circuit and destroy the spatial separation necessary for the hardening. This problem might be fixable through partitioning in the tool to separate the different copies into proper stripes, but this possible solution is made difficult due to the flattening of the netlist. The TMR flip-flops cross between the copies to perform voting, which makes it difficult to assign instances in the netlist to specific copies based on netlist hierarchy.

A more complex but reliable method of allowing post-route placement adjustment is to use a custom CAD tool to triplicate the cell library and .lef file to add differently-named “A”, “B” and “C” copies to all the cells. These cells would then be restricted to different sites in the .lef file. In the triplication process, these sites could then be set up in the floorplan to create the proper interleaved structure. Next, when cells are triplicated in the placement and netlist files, their cell name as well as their instance name would be adjusted so that they match the proper copy of the circuit. The end result would be that cells that are assigned to the “A”, “B” and “C” copies can only move within the rows restricted to that copy, since only those rows have the proper floorplan site. Thus, optimization could move cells as desired while still ensuring that they are separated by at least 1 row of spacers.

### *B. Hierarchical Netlist Limitations*

The current TMR flip flop design uses 8 different cells to determine voting wire locations, based on the row that each cell occupies. Although this works well, it requires a one-to-one correspondence between a cell in the placement file and a cell instance in the netlist. Hierarchical netlists break this relationship by allowing the same sub-block to be instantiated multiple times, potentially causing naming collisions if the same instance definition is used in two different rows, thus requiring two different names to define the voting wires. As a result, the netlist must be either flat or “uniquified”. For larger hierarchical designs, however, this can provide severe penalties to processing speed when designing a circuit and checking it for errors.

The only solution to this problem is to use a cell that does not care about row position, and can be converted blindly without needing to adjust its name. A macro block such as the one used in the original design of the cells fixes this problem. Unfortunately, this adjustment creates some issues of its own. The characterization and synthesis tools have to be set up in such a way that they can understand that there are 8 separate copies of the cell, which was difficult on the tool version used at the start of this project. Additionally, using a macro block wastes space unless the entire block is used. If only a single flip-flop is needed, 7 of them will be wasted, yet still take up space and need to be created and tied to ground properly. Currently, the extra processing time necessary to deal with a flat netlist is small enough that the more flexible single flip-flop design is preferred over the macro-block implementation.

## 6.2 INTERFACING WITH DUAL-REDUNDANT MODULES

In the HERMES processor used as an example for this workflow, we designed the logic such that it only used triple redundancy when needed. Thus, areas that contained only the speculative state of the machine were dual redundant, which means that they can catch errors but not correct them. In theory, these dual redundant sections save a third of the space and power required for a fully TMR design. In practice, however, the conversion between these two regions of the processor absorbed a large enough area that the space advantage was essentially neutralized. In the HERMES processor design, the TMR blocks take up 1.82 mm$^2$, while the dual redundant area occupies approximately 2.5mm$^2$. Of the TMR region, 0.26 mm$^2$ is taken up by the blocks that convert dual redundant data to TMR. These conversion blocks could be completely removed if the entire chip was hardened with the TMR workflow. Since this area can be considered lost because of the dual redundant region, it can be directly applied to its area cost, which reduces the theoretical 33% savings down to only 28%.

In addition, the code required to address this change was labor intensive and introduced the risk that an SEE on an unanticipated node could cause data corruption. The interactions between these regions, plus the analysis of which logic needs to be TMR and which can be dual redundant, absorbed the vast majority of the code design work. Accounting for possible SEE induced errors was the most time-consuming element, since the designer must account for possible errors on every single node. Additionally, there is the risk of missing a node or critical timing for that node, which can leave the chip vulnerable to an SEE.

Considering the minimal gain and large effort needed to add dual redundancy to the chip, it does not seem cost effective in hindsight. Future redesigns of the chip are likely to do all of the core logic in the TMR domain using this workflow and to add a conversion stage into the cache blocks. Converting to the TMR domain in the cache means that it becomes part of the cache's semi-custom layout and allows more care to be used to ease the transition to TMR.

## 6.3 ON-THE-FLY DECOUPLING

The current version of the TMR flip-flop disables its voting circuitry when scan is enabled in order to detect manufacturing defects. However, this disable signal can be separated from the scan enable signal, which splits the logic into three single-redundant copies.  Decoupling the copies from each other removes the hardening from the circuit, but allows it to act as three separate cores, in theory processing up to three times as much data.  This feature might be useful if a chip is only rarely going to be used in a high radiation environment, yet still needs to be hardened periodically.  The TMR voting can be switched on before entering a dangerous environment and then switched off after radiation levels have dropped far enough that an SEE is acceptably rare.  While this feature depends heavily on the application in order to make it worthwhile, the end result would be a circuit that has no significant hardening penalty when running in this decoupled mode.

## 6.4 CONCLUDING SUMMARY

Even though the use of this TMR workflow is limited by the tools used in it, it still provides significant advantages.  Chief among them is that the workflow requires no special logic considerations for hardening and can be run on any commercial IP, even IP that has no knowledge of hardening on its own. However, care must be taken at the interfaces between the TMR region and other regions in order to avoid accidentally opening a vulnerable area or adding too much conversion overhead.  The current design for the HERMES chip has highlighted some of these interfacing limitations, but future development of the chip points towards extending the use of this workflow instead of other hardening methods.  If all of these issues are accounted for, the TMR workflow described here provides a highly scalable, high speed method for automating hardened circuit creation.  The additional possibility of allowing the decoupling of TMR circuits may also provide a huge boost to processing power for some applications, making its speed and power on-par with circuits that are completely unhardened.

# REFERENCES

1) R. Lacoe, *et al*., "Application of hardness-by-design methodology to radiation-tolerant ASIC technologies," *IEEE Trans. Nuc. Sci*., vol. 47, no. 6, Dec. 2000, pp. 2334-2341.

2) D. Rea, *et al*., "PowerPC RAD750-A microprocessor for now and the future," *Proc. IEEE Aerospace Conf.*, 2005, pp. 1-5.

3) F. Ricci, *et al*., "A 1.5 GHz 90-nm embedded microprocessor core," *VLSI Cir. Symp. Tech. Dig*., pp. 12-15, June 2005.

4) R. Ho, K. Mai and M. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, April 2001, pp. 490-504.

5) D. Deleganes, *et al*., "Low-voltage swing logic circuits for a Pentium 4 processor integer core," *IEEE J. Solid-state Circuits*, vol. 40, no. 1, Jan. 2005, pp. 36-43.

6) M. Gadlage, R. Schrimpf, J. Benedetto, P. Eaton, D. Mavis, M. Sibley, K. Avery, T. Turflinger, "Single event transient pulse widths in digital microcircuits," *IEEE Trans. Nuc. Sci.*, vol. 51, no. 6, pp. 3285-3290, Dec. 2004.

7) P. Dodd, M. Shaneyfelt, J. Felix, J. Schwank, "Production and propagation of single-event transients in high-speed digital logic ICs," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 6, pp. 3278–3284, Dec. 2004.

8) L. Beninni, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 8, pp. 299-316, 2000

9) L. Clark, *et al*., "An embedded 32-b microprocessor core for low-power and high-performance applications," *IEEE J. Solid-state Circuits*, vol. 36, no. 11, Nov. 2001, pp. 1599-1608.

10) K. Warren, A. Sternberg, J. Black, R. Weller, R. Reed, M. Mendenhall, R. Schrimpf, L. Massengil, "Heavy ion testing and single event upset rate prediction considerations for a DICE flip-flop," *IEEE Trans. Nuc. Sci*., vol. 56, no. 6, pp. 3130-3137, Dec. 2009.

11) N. Hindman, L. T. Clark, D. Patterson, K. E. Holbert, "Fully Automated, Testable Design of Fine-grained Triple Mode Redundant Logic," *IEEE Trans. On Nuc. Sci*. vol. 58, no. 6, Dec. 2011

12) G. Anelli, *et al*., “Radiation tolerant VLSI circuits in standard deep submicron CMOS technologies for the LHC experiments: practical design aspects,” *IEEE Trans. Nuc. Sci*., vol. 46, no. 6, Dec. 1999, pp. 1690-1696.

13) X. Yao, N. Hindman, L. T. Clark, K. E. Holbert, D. Alexander, and W. Shedd. “The Impact of Total Ionizing Dose on Unhardened SRAM Cell Margins.” IEEE Trans. On Nuc. Sci., Vol. 55, no. 6, Dec. 2008

14) L. Clark, et al., “Optimizing radiation hard by design SRAM cells,” *IEEE Trans. Nuc. Sci*., 54(6), pp. 2028-2036, Dec. 2007.

15) J. Felix, et al., “Radiation response and variability of advanced commercial foundry technologies,” *IEEE Trans. Nuc. Sci*., 53(6), pp. 3187 – 3194, Dec. 2006.

16) E. Seevinck, F. List, and J. Lohstroh, “Static-noise margin analysis of MOS SRAM cells,” *IEEE J. Solid-State Circuits*, SC-22(5), pp. 748-754, Oct. 1987.

17) K. Agarwal and S. Nassif, “Statistical analysis of SRAM cell stability,” *Proc. DAC*, pp. 57-62, July, 2006.

18) N. Gierczynski, et al., “A new combined methodology for write-margin extraction of advanced SRAM,” *IEEE Int. Conf. on Microelectronic Test Structures*, pp. 97-100, March 2007.

19) J. Schwank, et al., “Effects of total dose Irradiation on single-event upset hardness,” *IEEE Trans. Nuc. Sci*., 53(4), pp. 1772 – 1778, Aug. 2006.

20) B. Chappell, A. Duncan, K. Ganesh, M. Gunwani, A. Sharma, M. Swarma, “Library Architecture Challenges for Cell-Based Design,” *Intel Technology Journal*, 8(1), pp. 55 – 61, Feb. 2004.

21) H. Barnaby, “Total-Ionizing Dose Effects in Modern CMOS Technologies,” *IEEE Trans. Nuc. Sci.*, 53(6), pp. 3103-3121, Dec. 2006.

22) F. Faccio, and G. Cervelli, “Radiation-Induced Edge Effects in Deep Submicron CMOS Transistors,” *IEEE Trans. Nuc. Sci.*, 52(6) pp. 2413-2420, Dec. 2005

23) L. T. Clark, D. Pettit, K. Holbert, N. Hindman, “Validation of and Delay Variation in Total Ionizing Dose Hardened Standard Cell Libraries,” Circuits and Systems (ISCAS), Inter. Symp., 15-19, May 2011, pp. 2051-2054

24) C. Weaver, J. Emer, S. Mukherjee, and S. Reinhardt, “Techniques to reduce the soft error rate of a high-performance microprocessor,” *Proc. ISCA*, 2004, pp. 264-275.

25) K. Soliman and D. K. Nichols, “Latchup in CMOS Devices from Heavy Ions,” *IEEE Trans. Nucl. Sci.*, vol. 30, pp. 4514–4519, 1983.

26) R. Ecoffet and S. Duzellier, “Estimation of latch-up sensitive thickness and critical energy using large inclination heavy ion beams,” *IEEE Trans. Nucl. Sci.*, vol. 44, pp. 2378–2385, 1997.

27) C. J. Marshall, P. W. Marshall, R. L. Ladbury, A.Waczynski, R. Arora, R. D. Fotlz, J. D. Cressler, D. M. Kahle, D. Chen, G. S. Delo, N. A. Dodds, J. A. Pellish, E. Kan, N. Boehm, R. A. Reed, and K. A. LaBel, “Mechanisms and temperature dependence of single event latchup observed in a CMOS readout integrated circuit from 16–300 K,” *IEEE Trans. Nucl. Sci.*, vol. 57, no. 6, pp. 3078–3086, 2010.

28) N. Hindman, L. T. Clark, D. Patterson, K. E. Holbert, “Fully Automated, Testable Design of Fine-grained Triple Mode Redundant Logic,” *IEEE Trans. On Nuc. Sci*. vol. 58, no. 6, Dec. 2011

29) L. T. Clark, D. Patterson, K. Holbert, S. Maurya, S. Guertin, “A Dual Mode Redundant Approach for Microprocessor Soft Error Hardness,” *IEEE Trans. Nuc. Sci*., vol. 58, no. 6, Dec. 2011, pp. 3018-3025

30) D. Mavis, P. Eaton, “Soft error rate mitigation techniques for modern microcircuits,” *Proc. IEEE IRPS*, pp. 216-225, 2002.

31) X. Yao, L. T. Clark, S. Chellappa, K. E. Holbert, and N. Hindman. "Design and Experimental Validation of Radiation Hardened by Design SRAM Cells," IEEE Trans. On Nuc. Sci., vol.57, no.1, Feb. 2010

32) X. Zhu, *et al.*, "A Quantitative Assessment of Charge Collection Efficiency of N+ and P+ Diffusion Areas in Terrestrial Neutron Enviroment," *IEEE Trans. Nuc. Sci.*, vol.54, no.6, pp.2156-2161, Dec. 2007

33) I. Esqueda, H. Barnaby, K. Holbert, Y. Boulghassoul, “Modeling Inter-Device Leakage in 90 nm Bulk CMOS Devices,” *IEEE Trans. Nucl. Sci.*, vol. 58, no. 3, pp. 793-799, June 2011

34) D. Hansen, *et al.*, “Clock, flip-flop, and combinatorial logic contributions to the SEU cross section in 90 nm ASIC technology,” *IEEE Trans. Nuc. Sci.*, vol. 56, no. 6, pp. 3542-3550, Dec. 2009.

35) J. Knudsen and L. Clark, “An area and power efficient radiation hardened by design flip-flop,” *IEEE Trans. Nuc. Sci.*, vol. 53, no. 6, pp. 3392-3399, Dec. 2006.

36) H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, “A review of Xilinx FPGA architectural reliability concerns from Virtex to Virtex-5,” *Radiation and Its Effects on Components and Systems (RADECS)*, 1-8, Sept. 2007.

37) P. E. Dodd, *et al.*, “Charge Collection and SEU from Angled Ion Strikes,” *IEEE Trans. Nuc. Sci*., vol.44, no. 6, pp.2256-2265, Dec. 1997

38) V. Correas, *et al.*, "Innovative Simulations of Heavy Ion Cross Sections in 130 nm CMOS SRAM," *IEEE Trans. Nuc. Sci*., vol.54, no.6, pp.2413-2418, Dec. 2007

39) D.G. Mavis, P.H. Eaton, *et al.*, "Multiple Bit Upsets and Error Mitigation in Ultra-Deep Submicron SRAMS," *IEEE Trans. Nuc. Sci*., vol.55, no.6, pp.3288-3294, Dec. 2008

40) D.F. Heidel, *et al.*, "Single-Event Upsets and Multiple-Bit Upsets on a 45 nm SOI SRAM," *IEEE Trans. Nuc. Sci*., vol.56, no.6, pp.3499-3504, Dec. 2009

41) D. Giot, P. Roche, *et al.*, "Multiple-Bit Upset Analysis in 90 nm SRAMs: Heavy Ions Testing and 3D Simulations," *IEEE Trans. Nuc. Sci*., vol.54, no.4, pp.904-911, Aug. 2007

42) A. D. Tipton, *et al.*, “Multiple-Bit Upset in 130nm CMOS Technology,” *IEEE Trans. Nuc. Sci*., vol.53, no.6, pp.3259-3264, Dec. 2006

43) S. Uznanski, *et al.*, "Single Event Upset and Multiple Cell Upset Modeling in Commercial Bulk 65-nm CMOS SRAMs and Flip-Flops," *IEEE Trans. Nuc. Sci*., vol.57, no.4, pp.1876-1883, Aug. 2010

44) O. A. Amusan, *et al.*, “Single Event Upsets in a 130nm Hardened Latch Design Due to Charge Sharing,” *Reliability Physics Symposium, 2007.*, pp.306-311, April 2007

45) F.X. Ruckerbauer, *et al.*, "Soft Error Rates in 65nm SRAMs--Analysis of new Phenomena," *On-Line Testing Symposium, 2007.*, pp.203-204, July 2007

46) A.D. Tipton, *et al.*, “Device-orientation effects on multiple-bit upset in 65 nm SRAMs,” *IEEE Trans. Nuc. Sci.*, vol. 55, no. 6, pp. 2880-2885, Dec. 2008.

47) T. Mozdzen, “Design Methodology For A 1.0 μm Cell-Based Library Efficiently Optimized for Speed and Area,” *IEEE ASIC Seminar and Exhibit*, pp. P12/3.1 – P12/3.5, Sept. 1990.

48) B. Chappell, A. Duncan, K. Ganesh, M. Gunwani, A. Sharma, M. Swarma, “Library Architecture Challenges for Cell-Based Design,” *Intel Technology Journal*, 8(1), pp. 55 – 61, Feb. 2004.

49) K. Kloukinas, F. Faccio, A. Marchioro and P. Moreira,“Development of a radiation tolerant 2.0V standard cell library using a commercial deep submicron CMOS technology for the LHC experiments” *Proc. of the fourth workshop on electronics for LHC experiments*, pp. 574-580, 1998

*50)* N. Duc and T. Sakurai, “Compact yet High-Performance (CyHP) Library for Short Time-to-Market with New Technologies,” *In Proc. ASP-DAC*, pp.475-480, *Jan. 2000*

51) A. Giraldo, A. Paccagnella, and A. Minzoni, “Aspect ratio calculation in n-channel MOSFETs with a gate-enclosed layout,” *Solid-State Electronics*, 44(6), pp. 981-989, Jun. 2000.

52) L. Clark, K. Mohr, and K. Holbert, “Reverse-Body Biasing For Radiation-Hard by Design Logic Gates,” *IRPS Proc.*, pp. 582 – 583, Apr. 2007.

53) B. Matush, T. Mozdzen, L. Clark, J. Knudsen, “Area-efficient temporally hardened by design flip-flop circuits,” *IEEE Trans. Nuc. Sci.*, vol. 57, no. 6, pp. 3588-3595, Dec. 2010.

54) M. Martin, “The design of a self-timed circuit for distributed mutual exclusion,” in Chapel Hill Conference on VLSI, 1985, pp. 245-260.

55) D. Muller and W. Bartky, "A theory of asynchronous circuits," *Proc. Int'l Symp. Theory of Switching*, 1959, pp. 204–243.

56) R. Shuler, B. Bhuva, P. O’Neill, J. Gambles, S. Rezgui, “Comparison of dual-rail and TMR logic cost effectiveness and suitability for FPGAs with reconfigurable SEU tolerance,” *IEEE Trans. Nuc. Sci.*, vol. 56, no. 1, pp. 214-219, Dec. 2009.

57) K. Morgan, D. McMurtrey, B. Pratt, M. Wirthlin, “A comparison of TMR with alternative fault-tolerant design techniques for FPGAs,” *IEEE Trans. Nuc. Sci.*, vol. 54, no. 6, pp. 2065-2072, Dec. 2007.

58) T. Calin, M. Nicolaidis, and R. Velazco, “Upset hardened memory design for submicron CMOS technology,” *IEEE Trans. Nuc. Sci.*, vol. 43, no. 6, pp. 2874-2878, Dec. 1996.

59) Actel Corporation, “RTAX-S/SL Rad Tolerant FPGAs,” May 2009. [online] Avalable: www.actel.com/documents/RTAXS_DS.pdf.

60) N. Weste, and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Boston: Addison Wesley, 2005.