The dissertation committee for
Kartik Mohanram certifies this
is the approved version of:

Reliability and Test of

High-Performance Integrated Circuits

Committee:

____

Nur Touba, Supervisor

____

Margarida Jacome

____

Tony Ambler

____

Gustavo De Veciana

____

Dinos Moundanos

Reliability and Test of High-Performance Integrated Circuits

by

Kartik Mohanram, B. Tech., M. S. E.

Dissertation
Presented to the Faculty of the Graduate School of
the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of
Doctor of Philosophy
The University of Texas at Austin
August 2003

Acknowledgements

The single biggest influence on this work is Nur Touba, who got me started in the field of design automation for VLSI. Nur's unflagging support and encouragement have sustained and driven me throughout the last five years. To the extent that the work presented herein is worthy of credit, Nur deserves a large portion of that credit. His standards in professionalism, integrity, and honesty have raised the bar of excellence for me to follow.

Another important influence is Gustavo De Veciana, whose professional life is grounded in a deep understanding of and love for electrical engineering and the teaching and learning process. Words cannot do justice to express all that Gustavo's mentoring has come to mean to me. Thanks also to Margarida Jacome, not just for her support on my dissertation committee, but also for her encouragement and particularly forceful endorsement that has helped me embark on a career in academics.

I must also thank Dinos Moundanos and Jawahar Jain for their part in making my stay at and collaborations with the Fujitsu Laboratories of America highly productive. It was there that I took my first tentative steps into the world of research and there that I found the courage and confidence to pursue my own ideas. I would also like to acknowledge Adnan Aziz at the University of Texas at Austin, H. Narayanan and Dinesh Kumar Sharma at the Indian Institute of Technology, Bombay, and Michael Gössel at the University of Potsdam.

Thanks to Tony Ambler and the other members of my dissertation committee who gave it a careful and intelligent reading. Of course, I am also indebted to the department of Electrical and Computer Engineering at the University of Texas at Austin and all the people therein that make it tick. I consider myself privileged to have been a part of it, and to have graduated from the University of Texas at Austin.

No thanks can do justice to the support and help extended to me by all my friends. I am particularly indebted to Niraj, Sriraman, Bala, Shreemoy, Gayathri and Randy, and Divya and Harish whose kindness and patience saw me through my prolonged convalescence. And to all the CATs (past and present) who have provided a highly collegial atmosphere, whether in the claustrophobic confines of ENS or the airy opulence of ACES.

And lastly, I acknowledge the support of my parents, Pushpa and Mohanram, who have patiently endured my every quirk and supported me steadfastly in all my endeavors. To conclude, I dedicate my doctorate to my grandparents – Jayapal, Subramanian, Madhavi, and Vatsala – whose standards and integrity have helped shape mine.

Reliability and Test of High-Performance Integrated Circuits

Publication No. _______________

Kartik Mohanram, Ph.D.
The University of Texas at Austin, 2003

Supervisor: Nur Touba

As high-density, low-cost, high-performance computing devices become more ubiquitous, there is an increased necessity to address the reliable operation of such systems. Both on-line test (concurrent error detection (CED)) and off-line (manufacturing test) techniques contribute to ensuring high levels of product reliability. The first part of this thesis focuses on techniques for CED in integrated circuits. The goal is to develop techniques for the insertion of CED circuitry at higher levels of design abstraction, as well as techniques that make it easier to absorb the associated overhead costs of CED. Approaches for automated design of logic circuits that meet failure rate requirements while minimizing the impact to area, performance, and power are described. The primary emphasis in this thesis is on reducing the soft error failure rate in integrated circuits (which dominates). The latter part of this thesis focuses on off-line test techniques for high-frequency I/O ports in integrated circuits. A low-cost trigger-based solution to eliminate the problem of non-determinism that may arise due to limitations in tester edge placement accuracy during at-speed functional test of high-speed source synchronous I/O ports is described. An analysis of when the problem of non-determinism becomes significant enough to warrant the implementation of the proposed solution is also provided.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

As process technology scales below 100 nanometers, the ability to integrate over a billion transistors on a single die presents exciting opportunities and challenges in all segments of VLSI design. As high-density, low-cost, high-performance computing devices become more ubiquitous, there is an increased necessity to address the reliable operation of such systems. Both on-line test (concurrent error detection (CED)) and off-line (manufacturing test) techniques contribute to ensuring high levels of product reliability. The first part of this thesis focuses on techniques for CED in integrated circuits. The goal is to develop techniques for the insertion of CED circuitry at higher levels of design abstraction, as well as techniques that make it easier to absorb the associated overhead costs of CED. The latter part of this thesis focuses on off-line test techniques for high-frequency I/O ports in integrated circuits.

1.1 Concurrent error detection

As feature sizes continue to shrink, high-performance circuits, characterized by high operating frequencies, low voltage levels, and small noise margins, are increasingly susceptible to temporary faults, which have emerged as an important failure mode. Temporary faults include those caused by crosstalk, substrate and power supply noise, charge sharing, etc. and pose a significant challenge to ensuring signal integrity even in present-day (deep sub-micron) process technologies. In addition, current studies indicate that circuits will become increasingly sensitive to temporary faults caused by terrestrial cosmic rays and alpha particles (that originate from impurities in the packaging materials), and that this will result in unacceptable soft error rates even in mainstream commercial electronics [Ziegler 96]. Figure 1.1 presents a graph from a recent study in [Cohen 99] that predicts a 100X increase in the soft error failure rate in core combinational logic by 2012. In a recent study, it has been projected that by 2011, the soft error rate in logic circuits will be comparable to that of present-day unprotected memory elements [Shivakumar 02].

Circuits with CED have the capability to detect both temporary and permanent faults and are widely used in systems where dependability[1] and data integrity are of importance [Gössel 93], [Nicolaidis 98], [Siewiorek 98]. High dependability comes at a cost however, since CED schemes impose extra overhead (area, timing, and power) on the design. The use of CED depends on the failure rate requirements of the application. Consider two classes of applications: mission critical applications (e.g., traffic control, banking, medical, etc.) and mainstream applications. Previous research in CED has focused on mission critical systems where very high levels of dependability are required. In these applications, the main goal is to provide high levels of coverage, while the overhead cost of CED is of less importance. In mainstream applications, cost and performance are the primary objectives. As CED becomes increasingly important in more high-volume mainstream commercial electronics, there is a strong need not only for computer-aided design (CAD) tools to automate the process of inserting CED circuitry, but also for methodologies that allow easy tradeoffs between overhead and coverage.

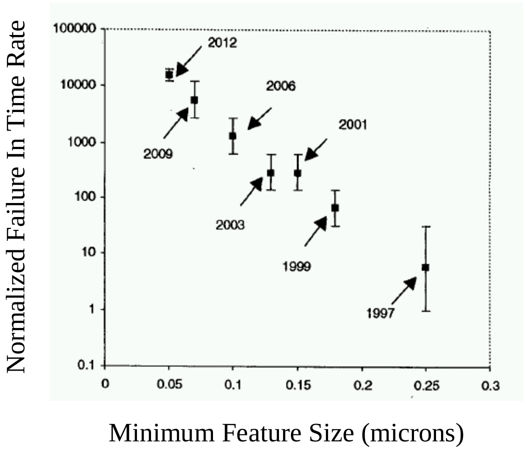


Figure 1.1 Projected soft error failure in time (FIT) rate[2] [Cohen 99]

---

[1] Dependability is defined as the trustworthiness of a computer system such that reliance can be justifiably placed on the service that it delivers [Laprie 92]. Reliability is the attribute of dependability with respect to the *continuity of service*.

[2] FIT rates have been normalized to a product with a 0.25 micron minimum feature size operating at 2.5V. The error bars account for the range of supply voltage provided by the [ITRS 99] roadmap.

1.1.1 Objective

The objective of the first part of this thesis will be to develop methodologies for the cost-effective incorporation of CED circuitry in integrated circuits. The two main sources of cost in incorporating CED circuitry in a design are:

1. Design time – The time required to properly insert the CED circuitry in a design and to verify that it works correctly and meets coverage requirements.
2. Overhead – The CED circuitry requires additional silicon area and can impact the performance, power consumption, and other design criteria.

The techniques (presented in Chapters 2, 3, 4, and 5) focus on the reduction of one (or more) of the above costs associated with the incorporation of CED circuitry without compromising on coverage.

1.1.2 Automated insertion of CED circuitry

In Chapter 2, we present a methodology for automated insertion of CED circuitry in synthesizable Verilog RTL that allows easy tradeoff between overhead and coverage. Experimental results are shown which demonstrate the broad range of design points that the proposed methodology offers the designer to choose from to satisfy cost/coverage requirements. This work was completed as part of a research project sponsored by Hewlett-Packard.

1.1.3 Estimation and reduction of soft error failure rate in logic circuits

In Chapter 3, we present a new paradigm for designing logic circuits with concurrent error detection (CED). The key idea is to exploit the asymmetric soft error susceptibility of nodes in a logic circuit. Rather than target all modeled faults, CED is targeted towards the nodes that have the highest soft error susceptibility to achieve cost-effective tradeoffs between overhead and reduction in the soft error failure rate. Under this new paradigm, we present one particular approach that is based on partial duplication and show that it is capable of reducing the soft error failure rate significantly with a fraction of the overhead required for full duplication. A procedure for characterizing the soft error susceptibility of nodes in a logic circuit, and a heuristic procedure for selecting the set of nodes for partial duplication are described. A full set of experimental results demonstrate the cost-effective tradeoffs that can be achieved.

1.1.4 Partial error masking in logic circuits

In Chapter 4, we present a new approach based on partial error masking to reduce the soft error failure rate in logic circuits. The proposed algorithm is composed of two reduction techniques, cluster sharing reduction and dominant value reduction, and achieves a very high reduction in the estimated soft error failure rate within the specified overhead constraints. This allows cost-effective tradeoffs between overhead and soft error failure rate reduction. The proposed technique scales very well for large circuits and is highly compatible with standard synthesis flows.

1.1.5 Reduction of power consumption in checkers used for CED

Power consumption and dissipation, for long concerns confined to the realm of portable systems, are today first order factors influencing integrated circuit design at all levels of the design flow. Power consumption and dissipation issues can also contribute to reliability concerns through electro-migration and hot-electron degradation effects. In Chapter 3, we address the problem of reducing power consumption in checkers used for CED. The basic idea is to exploit the functional symmetry of concurrent checkers with respect to their inputs to order the inputs such that switching activity (and hence power consumption) in the checker is minimized. There are no performance penalties, and no modifications are required either to the checker or the design. Experimental results demonstrate that a reduction in power consumption of 16% on the average for several types of checkers can be obtained using the proposed technique.

1.2 Test of high-frequency I/O ports in integrated circuits

Another important factor that influences the reliability of integrated circuits is the standard of the manufacturing test process. Manufacturing test techniques use automated test equipment (ATE), that through various structural and functional tests push the device under test (DUT) to its operating limits to ensure that it conforms to the specifications.

Those devices that make it through should, if operated within rated limits, have low and predictable field failure rates. In addition, ATE are used to weed out design and manufacturing deficiencies during new process and new product stabilization by subjecting prototypes to rigorous testing. To sum up, ATE play a significant role in ensuring high levels of integrated circuit reliability. Shortcomings in ATE capability can detract significantly from the efficiency of the manufacturing process and the reliability of the final product.

1.2.1 Functional fest of high-frequency I/O ports

Despite high levels of structured testability provided by scan and built-in self test, deep sub-micron designs, with multiple clock domains and internal asynchronous boundaries generally require a full complement of “at-speed” functional tests to be exercised in test mode. This is important not only for speed binning but also for detecting complex defects that are only uncovered when the full capabilities of the DUT are exercised [Maxwell 00].

Most DUTs generate high-frequency local clocks on-chip from the reference clock provided by the ATE using phase locked loop (PLL) oscillators. With the integration of high-speed I/O ports into the DUT, ATE will have to have (at least) a small number of pins capable of testing these ports. These ports include high-speed, source synchronous, low power-low voltage differential signaling (LP-LVDS) serial communication I/O buffers for systems such as SONET, Firewire, etc. with data rates exceeding 1 Gbps. Further, the increasing demand for throughput in mainstream as well as high-end systems has led to the emergence of high-speed, packet switched architectures that use source synchronous LP-LVDS bus technologies. These are rapidly replacing older hierarchical bus protocols as the choice for intra-system and inter-systems interconnect and are projected to attain performance levels scaling to 10 Gbps and beyond.

The at-speed functional testing of devices equipped with such high-speed I/O ports and buses, and the asynchronous nature of the I/O transactions poses significant challenges on the test methodology as well as ATE fronts. This is because the frequency of operation of these high-speed differential buses has increased at a rate much faster than the rate of improvement in ATE overall timing accuracy (OTA), eroding large margins of comfort that were previously available. In [ITRS 01], it is estimated that off-chip speeds have improved at a rate of 30% per year, while ATE OTA has improved at a rate of 12% per year. ATE OTA is a measure of the capability of the ATE and the rule of thumb is to strive for an OTA of 5% of the minimum cycle period of the required clock speed. For example, a DUT with I/O ports operating at a frequency of 10GHz has a cycle period of 100ps. State-of-the-art ATE feature OTA of ± 50ps across all I/O channels, and this is an order of magnitude higher than the desired OTA value of 5ps. The [ITRS 01] roadmap projects yield losses as high as 50% by 2014 due to shortcomings in ATE OTA.

1.2.2 Objective

We address the problem of non-determinism in the output response of the DUT that may arise due to jitter and limited ATE edge placement accuracy (EPA) in the source synchronous clock of the stimulus stream to the high-speed I/O port from the ATE. It is important that the tester not classify a defect-free device as a failure just because the received response did not match the expected one due to non-determinism. ATE OTA is usually characterized by ATE EPA. The EPA is a measure of the accuracy with which the ATE can place an input drive edge at exactly the same point in time relative to a common reference. The OTA of the ATE is usually given by

$$\left(ATE\ OTA = 2 \cdot ATE\ EPA\right)$$

The objective of second part of this thesis will be to use an analytical model for the EPA of the ATE to study the problem of non-determinism in the output response of the DUT, and to develop effective solutions for the elimination of the non-determinism. Solutions not only from the DUT perspective, but also from the ATE perspective will be explored. Solutions from the DUT perspective would include design-for-test (DFT) methods on the DUT to eliminate the effects of non-determinism. These methods allow for the application of functional test patterns without any degradation in performance. Solutions from the ATE perspective include modifications in the ATE to avoid the occurrence of non-determinism, as well as innovative test methodologies that allow for the application of at-speed functional patterns to the DUT. All the solutions need to be evaluated for test time, hardware overhead, and possible loss of coverage to ensure the effectiveness of the test process.

1.2.3 Use of a trigger signal to eliminate non-determinism

In Chapter 6, we present a simple yet effective solution that uses a trigger signal to initiate a deterministic transfer of test inputs into the core clock domain of the DUT from the high-speed I/O port is presented. The solution allows the application of at-speed functional patterns to the DUT, while incurring a very small hardware overhead and trivial increase in test application time. An analysis of the probability of non-determinism as a function of clock speed and EPA is presented. It shows that as the frequency of operation of high-speed I/Os continues to rise, non-determinism will become a significant problem that can result in an unacceptable yield loss.

1.3 Organization of the thesis

This thesis is organized as follows. Chapters 2, 3, 4, and 5 are devoted to work on CED for integrated circuits using error detecting codes. Chapter 2 presents work on automated insertion of CED circuitry at the Verilog RTL level. Chapter 3 presents work on estimation of the soft error failure rate in logic circuits. The results of such an estimation are exploited to achieve cost-effective soft error failure rate reductions using partial duplication. Chapter 4 presents an extension of the ideas from Chapter 3 to partial error masking to reduce the soft error failure rate in logic circuits. Chapter 5 presents work on power reduction in checkers used for CED. Chapter 6 presents work on the test of integrated circuits equipped with high-speed packet-switched I/O ports. Chapter 7 presents conclusions and directions for future research.

Chapter 2. Automated Insertion of CED Circuitry in Synthesizable Verilog RTL

In this chapter, a methodology for automated insertion of CED circuitry in synthesizable Verilog RTL which allows easy tradeoff between overhead and coverage is presented [Mohanram 02a]. The approach of inserting the CED circuitry at the Verilog RTL level can be easily and seamlessly incorporated into the standard design flow used in industry. Experimental results are shown which demonstrate the broad range of design points that the proposed methodology offers the designer to choose from to satisfy cost/coverage requirements.

A common design flow used in industry is for the designer to describe a design in synthesizable Verilog RTL. A synthesis tool is then used to generate an implementation from the Verilog RTL description. An advantage of having a Verilog RTL description of the design is that it can be used for fast RTL simulation. It also provides a high-level, easy to understand documentation of the design's function. Insertion of the CED circuitry at the RTL level rather than at the gate level, i.e., at the front-end of the synthesis process, is highly advantageous. It allows the CED circuitry to be optimized along with the functional circuitry. The synthesis tool can take the CED circuitry into account when satisfying timing constraints (as well as other constraints on power, testability, etc.). This avoids problems with having to reiterate the synthesis process because back-end insertion of the CED circuitry caused timing or other constraints to be violated. This contributes to a reduction in overall design turnaround time. CAD tools to support this methodology were developed and evaluated on existing Verilog designs. We describe the methodology and discuss the experimental results that have been obtained in this chapter.

The rest of this chapter is organized as follows. Section 2.1 is an analogy between high-level synthesis techniques and the present work. Section 2.2 is an overview of the methodology. Section 2.3 describes the CED insertion tool and how it modifies a synthesizable Verilog design. Section 2.4 describes the coverage evaluation tool and the process used to measure the coverage provided by the CED circuitry. Section 2.5 presents the experimental results that were obtained. Section 2.6 is a conclusion.

2.1 High-level test synthesis – an analogy

High-level synthesis (HLS) techniques accept as input an abstract behavioral specification of a digital design, that through a series of transformations is converted into an equivalent register-transfer-level (RTL) architecture. Most HLS techniques explore the design space using area, performance, and power metrics, and provide the designer with a set of architectures that meet these constraints. However, the original design can have very poor testability features that make it very hard to test and manufacture. While it can be argued that low-level design-for-testability (DFT) strategies achieve minimum testability requirements, this does not correspond to a complete exploration of the available design space. Further, the need to go through additional HLS iterations just to improve testability can prove expensive, especially in an environment where time-to-market is the only factor that determines success or failure. High-level test synthesis (HLTS) techniques incorporate testability metrics during the early stages of the design flow (at the behavioral-level and the architectural-level) to produce implementations with improved fault coverage, reduced test hardware overhead and faster turn-around times [Wagner 96]. HLTS strategies can produce inherently testable designs (at little or no area, performance, and power penalties) that require far less or no test investment at lower levels of the design process. Such implementations can have better overall area, performance, and power features than implementations made testable by lower-level DFT methodologies. Along the same lines, the necessity to incorporate reliability metrics into the design process (at all levels of design abstraction) cannot be stressed enough. By incorporating reliability metrics during the early stages of the design flow, inherently reliable designs that use a combination of fault avoidance and fault tolerance techniques can be realized.

2.2 Overview of methodology

Figure 2.1 presents the flowchart for the methodology. Given a synthesizable Verilog description, the Verilog is parsed and the CED circuitry is inserted in the design at the appropriate locations. The CED scheme that is used is selected by the designer depending on the coverage/cost requirements. After the CED circuitry has been inserted in the design, the resulting Verilog description is passed to a commercial synthesis tool to generate a gate-level implementation. Fault simulation is then performed on the gate-level implementation to evaluate the coverage that is provided by the CED circuitry. Other design criteria such as area, delay, power, etc., for the implementation are also evaluated. If the coverage that is provided by the CED circuitry meets requirements, and other design criteria are also satisfactory, the process is complete. However, if the coverage or cost is not satisfactory, the process can be reiterated with a different CED scheme. Thus, this methodology supports rapid evaluation of the design space to converge on the best implementation.

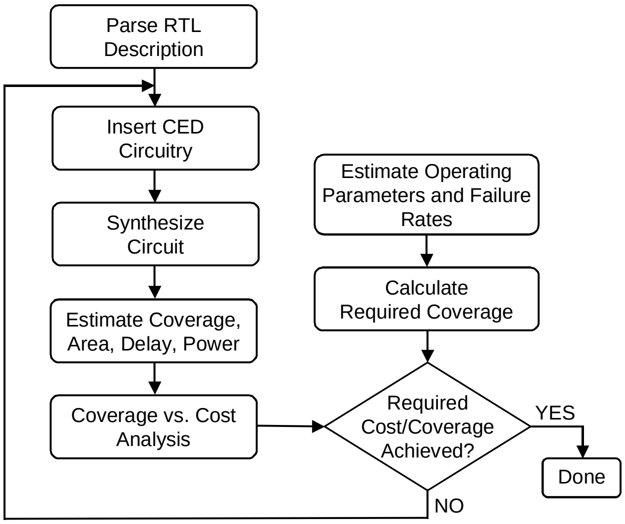


Figure 2.1 Flowchart for the proposed methodology

Two CAD tools were developed to support this methodology. The first is a CED insertion tool that takes as an input a synthesizable Verilog description and automatically inserts the CED circuitry based on the options specified by the designer. The CED insertion tool outputs a modified synthesizable Verilog description with the appropriate CED circuitry in place. The second CAD tool is a coverage evaluation tool. This tool takes as an input the gate-level implementation (after synthesis) and evaluates the coverage provided by the CED circuitry. In addition, the area, delay, and power overhead of the chosen scheme for CED is also computed. The techniques used in the CED insertion tool and the coverage evaluation tool are described in the following sections.

2.3 CED insertion tool

The goal of the CED insertion tool is to provide the designer with various options in terms of which CED schemes to use, and then automatically insert the CED circuitry into an existing synthesizable Verilog design. Synthesis is generally not used for generating an implementation for the data-path portion of a design. The data-path logic has a regular structure and is generally constructed from highly optimized components. For these components which have a regular structure, very efficient CED schemes exist (e.g., self-checking adder or multiplier designs [Gorshe 96], PLAs [Mak 82], [Nicolaidis 89]). Synthesis is generally used for obtaining an implementation of the control logic portion of a design which has an irregular multilevel structure. The CED insertion tool described here is targeted towards the control logic portion of a design.

2.3.1 CED with error detecting codes

The CED schemes that are supported by the CED insertion tool are based on error detecting codes. CED schemes based on error detecting codes use information redundancy, when the outputs of the circuit are encoded such that they have an invariant property. A checker usually monitors the outputs for the occurrence of an error. The idea is to choose a suitable error detecting code that guarantees the detection of all errors in a particular fault class.

The general approach for performing CED with a systematic error detecting code is illustrated in Fig. 2.2. Systematic codes are so called because the data can be directly extracted from the codeword. The function logic is augmented with a check symbol generator. Based upon the scheme chosen for CED, the check symbol generator can be a copy of the original circuit (duplication and compare), parity prediction logic, Berger or Bose-Lin codeword generator, etc. The function logic has *n* output bits, and the check symbol generator provides *k* check bits. Together, they form an *(n,k)* systematic code. There is a checker which monitors the encoded outputs and gives an error indication if a non-codeword occurs. The checker can be designed so that it is *self-checking*, which means that it is guaranteed to give an error indication if a fault occurs in the checker itself.

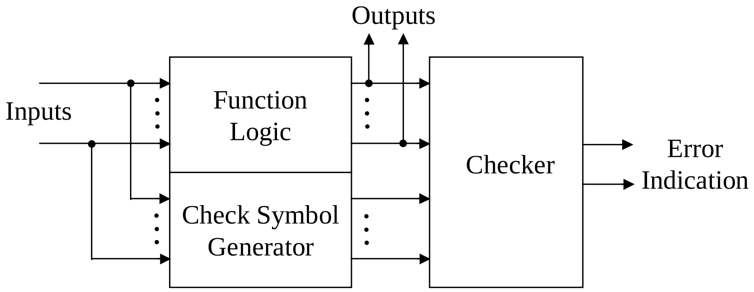


Figure 2.2 Conventional approach for CED with a systematic error detecting code

m-out-of-*n* codes are another popular class of unidirectional error detecting codes. In the *m*-out-of-*n* code, each codeword has weight *m*, i.e., exactly *m* out of *n* bits are logic ONE. Any unidirectional error destroys this property of the codeword and is guaranteed to be detected. These codes are non-systematic, since the information bits cannot be directly retrieved from the check bits. The *1*-out-of-*n* (one-hot) code is a popular choice used in the design of address decoders, and for state assignment in finite state machines (FSMs).

2.3.2 CED schemes supported

The control logic consists of finite state machines (FSMs) which control the data-path. The CED insertion tool inserts CED circuitry to detect errors in these FSMs. An FSM consists of next-state logic and output logic, in addition to the state registers. The CED insertion tool provides the designer with four different options in terms of the CED scheme used for each FSM:

1. Duplication – In this scheme, the top level module in the design hierarchy is simply duplicated and the outputs are compared with an equality checker. This requires greater than 100% area overhead. If the design has a lot of outputs, the equality checker can become very large.
2. One-Hot – In this scheme, a “one-hot” state assignment is used for the FSM (i.e., each state is encoded such that only one bit is a ‘1’ while the rest are all ‘0’). A one-hot checker is used to check the next state (as shown in Fig. 2.3 – note that the shaded areas denote the extra logic that is added for CED). One limitation in applying this scheme to an existing design is that the states need to be re-encoded. If the design uses special status bits or decodes a subset of the bits in the state register for some operations, then this scheme cannot be applied.

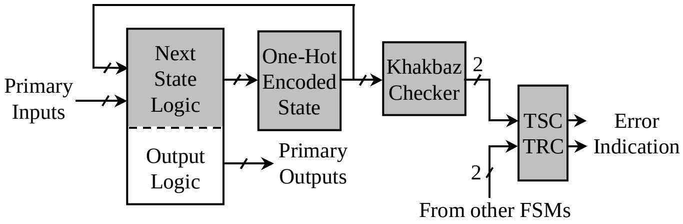


Figure 2.3 Block diagram for the one-hot scheme

3. Parity – In this scheme, a parity bit is appended to the state encoding. Unlike the one-hot scheme, the states do not need to be re-encoded. Another advantage is that if parity prediction is used on the outputs, then a single parity checker can be used to check both the present state and output parity (as shown in Fig. 2.4). This is done by generating the parity check bit so that it is the XOR of both the present state parity and output parity. The advantage of the parity scheme is that it requires less overhead than the duplication scheme, but this comes at the cost of losing coverage of errors that affect an even number of bits. However, in some designs, this loss of coverage is not substantial.

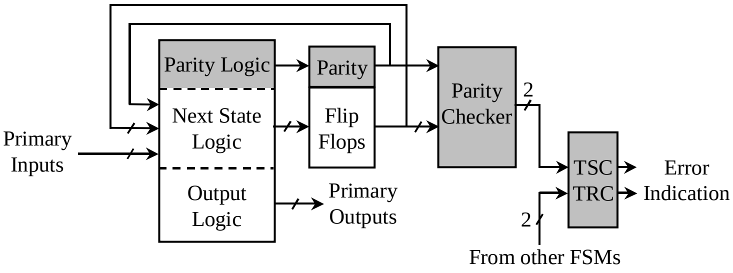


Figure 2.4 Block diagram for parity scheme

4. Hybrid Parity – In addition to parity encoding on the state of the FSM, the output logic is duplicated to provide a higher error coverage (as shown in Fig. 2.5). This approach is a hybrid between duplication and parity. It has less overhead than using full duplication, but better coverage than the parity and one-hot schemes.

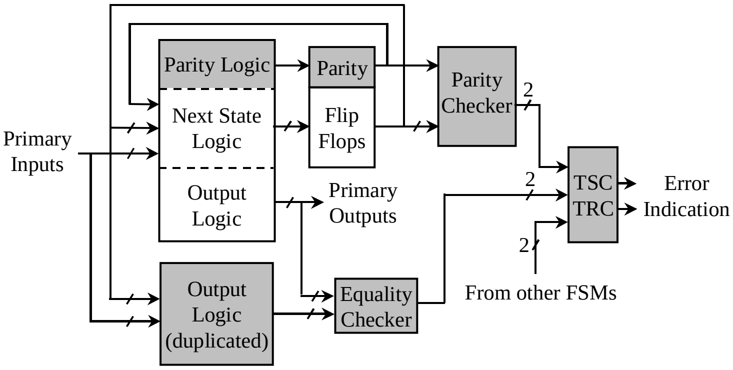


Figure 2.5 Block diagram for hybrid parity scheme

Each of the CED schemes has advantages and disadvantages. The designer has the flexibility to choose the scheme that is best suited for each FSM in the design. For FSMs with a small number of states, the one-hot scheme may provide better coverage at lower cost than parity. For larger FSMs, parity may be the most cost effective. If the coverage that is provided by parity is not sufficient, then hybrid parity or duplication can be used.

2.3.3 Pipelined checkers

Checkers usually add delay on the longest paths in the design. Since this impacts timing performance, the checkers are usually pipelined, with the introduction of latches right before the checker. By deferring the checker operation to the next clock cycle, the performance of the design remains unaltered at the cost of some latency on when the error is detected. An error occurring in one clock cycle is detected in the next clock cycle. Generally, this is early enough to prevent data corruption in most applications

2.3.4 Verilog modification

The CED insertion tool automatically modifies an existing synthesizable Verilog design to insert the CED circuitry. This is done by having the user identify each FSM and specify the CED scheme that is to be used for that FSM. The FSMs are identified by specifying the module name and the present state and next state variables. The tool then parses the module, modifies the statements containing the present state and next state variables accordingly, and adds the necessary checkers.

For the duplication scheme, the tool simply duplicates the entire FSM and adds an equality checker. For both the parity and one-hot schemes, the Verilog source has to be modified in places where the present and next state variables occur. This is explained in greater detail with respect to the example presented in Fig. 2.6:

1. Register declarations – The present and next state variables' register declarations are modified appropriately by increasing the width of the register to reflect a parity or one-hot encoded state. In the example in Fig. 2.6, the parity scheme requires that the present and next state registers change from 2 to 3 bit-wide registers. The parity bit is always the most significant bit, and the value is chosen to reflect an odd parity under normal operation. In a similar manner, the one-hot scheme would require that the registers be modified to be 4 bit-wide registers. Obviously, modifying the register width for the one-hot scheme can prove to be wasteful if all the states are not used in the FSM. Our CED insertion tool provides the user with an option to specify the exact number of states that are used in the FSM so that the present and next state registers are expanded to the optimum width when the one-hot encoding scheme is inserted.

```
reg [ 1:0 ] PS;
reg [ 1:0 ] NS;
assign sigX = ( sigY & ( PS == 2'b10 ) );
always @ ( PS or reset or …  ) begin
    if ( reset ) NS = 2'b00;
    else begin
        case ( PS )
          2'b00:
            NS = 2'b01;
            …
          default:
            NS = 2'b11;
        endcase
    end
end
```

➡

```
reg [ 2:0 ] PS;
reg [ 2:0 ] NS;
assign sigX = ( sigY & ( PS == 3'b010 ) );
always @ ( PS or reset or …  ) begin
    if ( reset ) NS = 3'b100;
    else begin
        case ( PS )
          3'b100:
            NS = 3'b001;
            …
          default:
            NS = 3'b111;
        endcase
    end
end
```

Figure 2.6 Example of modifying Verilog design for parity encoding scheme

2. <u>Equality operator</u> – The present state variable can also occur in conditional statements, for example, when the equality operator is used to check that the FSM is in a certain state. Here, the state encoding provided in the original design has to be appropriately modified to reflect the parity or one-hot encoding scheme. In the example, 2'b10 is modified to 3'b010, and the parity bit is made the most significant bit. Similarly, 2'b10 would be expanded to 4'b0100 if the one-hot encoding scheme were chosen.
3. <u>Blocking Assignments</u> – The next state variable can occur in blocking assignments when the FSM changes state depending on the combination of inputs and the present state. The changes to the Verilog source are similar to the ones described above for conditional statements.
4. <u>Output Declarations</u> – The parity encoding scheme has significant advantages over the one-hot encoding scheme in the two cases illustrated in Fig. 2.7. The first is when the contents of the present state register are exported outside the module in which the FSM is contained, by declaring all (or part) of the present state register as primary outputs of the module. The second case is when all (or part) of the present state register's contents are accessed in say a blocking assignment statement within or outside the module of occurrence of the FSM. In both these cases, the addition of the parity bit in the most significant position does not alter the functionality in any manner, and automation of the parity scheme poses no difficulties. In other words, since the parity encoding

scheme is a systematic (separable) encoding scheme, the modifications to the Verilog source are transparent to the entire design. Another advantage of parity encoding under these circumstances is that the functionality cannot be inadvertently altered during an optimization phase that follows the insertion of CED circuitry.

a) When Subset of State Register’s Contents Accessed

```
reg [ 4:0 ] present_state;
…
assign sigX [ 1:0 ] = present_state [ 3:2 ];
```

b) When State Exported Outside Module

```
module mod ( … , present_state );
   …
   output [ 4:0 ] present_state;
   reg [ 4:0 ] present_state;
   …
endmodule
```

Figure 2.7 Cases where parity encoding is advantageous over one-hot encoding

Lastly, when more than a single FSM occurs in the design, the error indication signals are stitched together during the write-back phase of the parser, and finally result in two global error indication signals that are encoded in a two-rail code. While the checker for the parity encoding scheme is straight-forward to introduce, care has to be taken to ensure that the checker for the one-hot encoding scheme is self-testing.

2.4 Coverage evaluation tool

The goal of the coverage evaluation tool is to evaluate the coverage that the CED circuitry provides. This is done at the gate-level after synthesis. The synthesis tool generates a gate-level implementation of the design which contains the CED circuitry.

We implemented the coverage evaluation tool using a commercial fault simulator. The coverage evaluation tool first generates scripts for the commercial fault simulator, which is then invoked to execute the scripts and produce output files. The coverage evaluation tool then parses the output files and computes statistics about the coverage that the chosen CED scheme provides. The patterns used for coverage evaluation can either be user-supplied, or internally randomly generated.

The coverage evaluation tool performs three runs of fault simulation. On each run, all single stuck-at faults are fault simulated. If the fault effect for each single stuck-at fault is propagated to an observation point, then the fault is considered detected. The location of the observation points is different for each of the runs. This is necessary to discount all faults that are within the checker circuitry itself, and hence propagate only to the error indication signals and not to the functional outputs. We use the fault dictionary option of the commercial fault simulator to report all the faults that are propagated to the observation points on each run. On the first run, both the error indication signals and the functional outputs are considered as observation points. On the second run, only the functional outputs are considered as observation points. Finally, on the third run, only the error indication signals are considered as observation points.

Let $f1, f2$ and $f3$ be the faults propagated to the observation points on the first, second and third runs respectively. The coverage evaluation tool computes the fault coverage as:

$$\frac{f3-(f1-f2)}{f2}$$

Note that $(f1-f2)$ gives the set of fault that propagate errors only to the error indication signals and not to the functional outputs. These are faults that occur in the error detection circuitry itself. $f3-(f1-f2)$ gives the set of faults that cause errors in the functional outputs, but are detected by the CED circuitry. These correspond to the faults in the original functional circuit that would have gone undetected in the absence of the CED circuitry. The ratio of these detected faults to all faults that cause errors in the function outputs gives the coverage that is provided by the chosen CED scheme. If this coverage does not meet requirements, then the designer can use the CED insertion tool to insert a different CED scheme that will provide better coverage.

2.5 Experimental results

Experiments were performed using the CAD tools on some synthesizable Verilog control logic designs. Information about each of the designs is given in Table 2.1

including the number of primary inputs, primary outputs, and FSMs, along with the number of states in each FSM. Designs 1, 2, and 3 were taken from the Torch processor. Design 4 was extracted from a Hewlett-Packard industrial design.

For each design, the CED insertion tool was used to automatically insert each of the four supported CED schemes: parity, one-hot, hybrid, and duplication. A commercial synthesis tool with generic libraries was then used to generate a gate-level implementation. The coverage provided by the CED circuitry for each of the schemes was measured using the coverage evaluation tool. Additionally, the area, delay, and power for the designs with the CED circuitry versus the original designs with no CED circuitry was compared. The results are presented in Table 2.2. Note that the values for the area, delay, and power, are normalized with respect to the original designs with no CED.

From the results, it can be seen that the 4 different schemes provide a broad range of cost/coverage tradeoffs that the designer can choose from. The parity scheme has the smallest area overhead in all cases, but also has the lowest coverage in all cases except for Design 4 where it provided slightly better coverage than the one-hot scheme. In general, the one-hot scheme requires a little more overhead than parity and provides a little better coverage (Design 4 is an exception). To increase the coverage, the hybrid scheme can be used. It provided over 92% coverage in all cases with less overhead than full duplication. If 100% coverage is required, then the duplication scheme can be used, although it has the largest overhead.

The faults that are missed by the parity scheme include faults that cause an even number of bits to be in error or a fault that causes an error in the next state and then a corresponding error in the output logic on the next clock cycle. The latter type of fault could be detected by using separate parity checkers for the present state and output. Faults that cause an even number of bits to be in error could be detected by using additional parity bits that check different subsets of outputs. One approach would be to analyze the gate-level circuit after synthesis and then select groups of outputs for which adding additional parity check bits would improve the coverage. Note that this would require a reiteration of the CED insertion flow.

The faults that are missed by the one-hot scheme include those that cause the next state to change from one valid one-hot value to another one (note that this requires at least a two-bit error), as well as faults that cause an even number of errors in the output logic (as previously discussed for the parity scheme).

The hybrid parity scheme uses duplication for the faults in the output logic and hence the only faults that are missed are those that cause an even number of errors in the next state.

Table 2.1 Information about designs

Design	Description	Num. PIs	Num. Pos	Num. FSMs	Num. States
1	External Interface Module	17	15	2	5, 5
2	Instruction Fetch Control Module	11	12	1	12
3	Load Store Unit Control Module	25	35	4	4, 6, 6, 15
4	Extracted from an HP Industrial Design	12	18	2	4, 12

Table 2.2 Normalized results for all the designs

Design 1				
Scheme	Area	Delay	Power	Coverage
No CED	100	100	100	0
Parity	109	101	140	75.3
One-Hot	112	101	145	77.0
Hybrid	190	152	165	93.0
Duplication	219	166	231	100

Design 2				
Scheme	Area	Delay	Power	Coverage
No CED	100	100	100	0
Parity	114	126	106	79.1
One-Hot	130	171	115	82.2
Hybrid	220	215	240	94.0
Duplication	244	232	248	100

Design 3				
Scheme	Area	Delay	Power	Coverage
No CED	100	100	100	0
Parity	140	130	120	79.4
One-Hot	170	190	133	85.0
Hybrid	215	150	230	94.1
Duplication	270	177	280	100

Design 4				
Scheme	Area	Delay	Power	Coverage
No CED	100	100	100	0
Parity	118	160	150	78.4
One-Hot	145	185	151	76.7
Hybrid	190	183	190	92.3
Duplication	234	225	200	100

2.6 Summary

In this chapter, an automated methodology for the insertion of CED circuitry into synthesizable Verilog RTL was presented. A CED insertion tool was described for modifying the Verilog source to add the CED circuitry. A coverage evaluation tool was also described for measuring the fault coverage that the CED circuitry provides in the gate-level implementation after synthesis. Using these two tools, the designer can rapidly explore various design points to find the best cost/coverage tradeoff that meets requirements. The experimental results demonstrated that the four CED schemes provide a broad range of cost/coverage tradeoffs to choose from. The methodology described here can be easily incorporated into the typical design flow used in industry and provides a quick solution for designers to use to satisfy coverage requirements.

Chapter 3. Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits

In this chapter, a new paradigm for designing logic circuits with CED that reduces the soft error failure rate is described [Mohanram 03b]. CED is targeted towards the nodes that have the highest soft error susceptibility to achieve cost-effective tradeoffs between overhead and reduction in the soft error failure rate. We address the issue of characterization of the soft error susceptibility of nodes in logic circuits and use the results of such a characterization to develop our low-cost CED methodology to reduce the soft error failure rate. We show that in the presence of the masking factors in logic circuits, SEUs at some internal nodes in logic circuits can have orders of magnitude greater probability of being latched and causing an error than at other nodes. By focusing CED towards the nodes that are most susceptible to SEUs, the soft error failure rate in logic circuits can be significantly reduced at a fraction of the cost of existing techniques that try to guarantee coverage of all nodes. We present an algorithm for the synthesis of CED circuitry for logic circuits based on partial duplication. The proposed algorithm achieves a very high reduction in the estimated soft error failure rate within the specified overhead constraints. We present experimental results that demonstrate the effectiveness of the proposed algorithm.

The rest of this chapter is organized as follows. Section 3.1 is an introduction. In Sec. 3.2, we describe the problem addressed in this chapter in greater detail and review some previous research in this area. In Sec. 3.3, we present the methodology for estimating the soft error susceptibility of nodes in logic circuits. In Sec. 3.4, we present a methodology for the synthesis of low-cost CED circuitry for logic circuits based on partial duplication. In Sec. 3.5, we present experimental results. Section 3.6 is a conclusion.

3.1 Introduction

When high-energy neutrons (present in terrestrial cosmic radiation) and alpha particles (that originate from impurities in the packaging materials) strike a sensitive region in a semiconductor device, they generate a dense local track of electron-hole pairs. This may be collected by a p-n junction resulting in a current pulse of very short duration termed a *single-event upset (SEU)* in the signal value. A SEU may cause a bit flip in some latch or memory element thereby altering the state of the system resulting in a *soft error*. Additionally, a SEU may occur in an internal node of combinational logic and subsequently propagate to and be captured in a latch. Soft errors in memories (both static and dynamic) have traditionally been a much greater concern than soft errors in logic circuits (for the same minimum feature size) since memories contain by far the largest number and density of bits susceptible to particle strikes. As process technology scales below 100 nanometers, studies indicate that integrated circuits will be increasingly susceptible to SEUs and that this will result in unacceptable soft error failure rates even in mainstream commercial applications [Ziegler 96], [Cohen 99]. In a recent study, it has been projected that by 2011, the soft error rate in logic circuits will be comparable to that of unprotected memory elements [Shivakumar 02].

A system or component is said to *fail* if it does not correctly perform its intended function. Whether or not a soft error causes a component or system to fail depends on its fault tolerance features. If a soft error is not detected, then it can result in a failure. The *failure rate* for a component or system is the number of failures that occur per unit time. It is generally measured in units of FIT (1 failure in 10^9 hours of operation). Note that there may be other sources of failures in a system besides soft errors (e.g., permanent faults), however, this paper just focuses on the soft error failure rate (which dominates). All sources of failures are additive, so they can be considered independently.

CED circuitry can be used to monitor the outputs of a circuit for the occurrence of an error. If an error is detected, then the system can recover thereby preventing a failure. Traditionally for mainstream applications, soft error failure rates have been tolerable even without the use of CED. However, as the soft error failure rate for memories has increased, the use of CED for memories has become more common, e.g., parity and error correcting codes (ECC). CED for logic circuits in mainstream applications has seen limited use for two reasons:

1) Very high overhead (power, area, timing, etc.) – Unlike memories, logic circuits do not have a regular structure thereby making CED more complex. Conventional techniques for CED were developed for mission critical applications where the goal is to detect all errors under a fault model. Some synthesis techniques have been developed for reducing the overhead without compromising on the error detection capability [Jha 93], [De 94], [Bolchini 97], [Touba 97], [Saposhnikov 98], [Das 99], [Zeng 99]. Some techniques based on multiple sampling of outputs have also been developed [Franco 94], [Metra 98], [Nicolaidis 99], [Favalli 02].

2) Lower susceptibility to soft errors – Logic circuits have a natural barrier to propagating SEUs to their output [Lidén 94]. When a particle strike occurs at an internal node of a logic circuit, there are three masking factors – *logical, electrical,* and *latching-window* – that may prevent it from being latched and resulting in a soft error (this is described in detail in Sec. 3.2). For larger process technologies, the soft error failure rate for logic circuits is much lower than for memories.

In the next decade, the soft error failure rate for logic circuits is projected to increase dramatically. Technology trends are causing the barriers for propagating SEUs to diminish significantly. These include smaller feature sizes, lower voltage levels, higher operating frequencies, reduced logic depth between latches, etc. (the effects of these will be described in detail in Sec. 3.2). As a result, reducing the soft error failure rate in logic circuits in mainstream applications is an important challenge for the future.

One approach for reducing the soft error failure rate in mainstream applications is to simply apply the CED techniques that have been developed for mission critical applications. However, these techniques are geared towards very high reliability and thus may be overkill. In the highly cost sensitive environment of mainstream applications, the goal is to reduce the soft error failure rate to acceptable levels at minimum cost. Thus, there is a need for a new class of CED techniques that span the middle ground between no protection/no overhead and very high protection/very high overhead. Whereas traditional CED techniques for mission critical applications target all modeled faults, the CED techniques for mainstream applications need to satisfy soft error failure rate reductions in a cost-effective manner.

In the rest of this chapter, a new paradigm for designing logic circuits with CED is described. Rather than target all modeled faults, CED is targeted towards the nodes that have the highest soft error susceptibility, i.e., the nodes that contribute the most to the soft error failure rate of the logic circuit. This allows cost-effective tradeoffs between overhead and soft error failure rate reduction. Such techniques can be used in cost-sensitive mainstream applications to satisfy soft error failure rate requirements at minimum cost. Under this new paradigm, we present one particular approach that is based on partial duplication and show that it is capable of reducing the soft error failure rate significantly with a fraction of the overhead required for full duplication. The proposed technique scales very well for large circuits and is highly compatible with synthesis flows. While some results have shown parity prediction can achieve significant reductions in overhead versus duplication for arithmetic logic units (ALUs) or small circuits with few levels of logic, it does not scale well for large multilevel circuits where the parity functions are very hard to optimize with synthesis tools.

3.2 Motivation and previous work

In this section, we discuss the factors that contribute to an increase in the soft error failure rate in logic circuits in greater detail. Specifically, we look at the masking factors and how present-day design trends are diminishing their significance leading to higher soft error failure rates. We then discuss the key idea in our work which is based upon exploiting the asymmetric soft error susceptibility of nodes in logic circuits. Lastly, we review previous work in the area of soft error failure rate estimation for integrated circuits.

3.2.1 Reasons for increase in soft error failure rate of logic circuits

Soft errors in memories have traditionally been a much greater concern than soft errors in logic circuits. Memories are very susceptible to soft errors because of their small cell size and the fact that a SEU in a memory cell can immediately result in a soft error provided it exceeds a certain minimum critical charge required to flip the value stored in the cell [May 79], [Kirkpatrick 79].

Logic circuits have been much less susceptible to soft errors than memories. If a SEU occurs at an internal node of a logic circuit, there are three factors that may prevent it from being latched and resulting in a soft error:

1) There needs to be a functionally sensitized path from the location of the SEU to a latch. This will depend on what input vector is being applied at the time of the SEU. If there is not a sensitized path, then the SEU will be logically masked.

2) The SEU must create a pulse of significant duration and amplitude to propagate through each stage of logic until it reaches a latch. The pulse will be attenuated due to the electrical properties of each gate that it passes through, so the farther away from a latch the SEU occurs, the stronger it must be to make it to the latch. If a pulse is attenuated before it propagates (along a sensitized path) and reaches a latch, then the SEU will be electrically masked.

3) The timing of the SEU must be such that it causes a pulse that arrives at a latch just as the clock transitions so that the latch captures its value. If the SEU occurs at a time outside of the “latching-window”, then it will not be captured in the latch and the SEU will be latching-window masked.

These three factors present a natural barrier to soft errors in logic circuits and have prevented soft errors in logic circuits from being a major concern [Lidén 94]. However, technology trends are causing these barriers to diminish significantly. The trend towards reduced logic depth between latches means that (1) there is less attenuation when propagating SEUs and (2) there is an increased number of sensitized paths. In addition, smaller feature sizes and lower voltage levels result in a reduction in the charged stored at a node. This allows lower energy particles to cause SEUs capable of being latched. Particles of lower energy occur much more frequently than particles of higher energy. An order of magnitude difference in energy can correspond to more than an order of magnitude larger flux for the lower energy particles. Faster gates allow SEUs of smaller pulse width to propagate through the circuit with minimum attenuation to the outputs. High operating frequencies mean that there are more latching-windows per unit time thereby increasing the probability of a SEU being latched.

Thus, as technology continues to scale, logic circuits are becoming much more susceptible to soft errors. Projections in [Shivakumar 02] indicate that for microprocessors that use ECC to reduce the soft error failure rate for memories, logic will become the dominant source of soft error failures. Thus, reducing the soft error failure rate for logic circuits is expected to emerge as a very important problem in the future.

3.2.2 Soft error susceptibility of nodes in logic circuits

A key idea in this paper is to exploit the asymmetric soft error susceptibility of internal nodes in a logic circuit. While radiation bombards a chip fairly uniformly in space and time, the probability that a SEU is latched varies greatly depending on which node it occurs at in the logic circuit. The reasons for this include the following:

1) The percentage of time that each node is functionally sensitized to a latch depends on the logic function being implemented and the distribution of input vectors that are applied while the circuit operates. Often a small subset of the input vectors can be applied for a large percentage of the clock cycles resulting in certain nodes being sensitized to a latch much more frequently than others.

2) The size of the gate driving a node and the amount of capacitance at the node affects how much particle energy is required to create a SEU of sufficient strength to be latched. Particles with lower energy have much greater flux than those of higher energy, thus this can greatly skew the probability of a SEU of sufficient strength occurring at certain nodes.

3) The logic depth of a node from a latch affects how many gates a SEU has to

propagate through to reach a latch and therefore how much attenuation will occur. The farther away a node is from a latch, the more particle energy is required to create a pulse of sufficient strength to be latched and thus the less likely it is to occur.

As a result of these factors, the soft error susceptibility of internal nodes in a logic circuit can vary by at least an order of magnitude. This provides an opportunity to significantly reduce the soft error failure rate at a reduced cost, since CED techniques can be targeted towards the nodes with high soft error susceptibility, while those with very low soft error susceptibility can essentially be ignored. This can be used to achieve a significant reduction in the soft error failure rate in a cost-effective manner.

Figure 3.1 shows the normalized soft error susceptibility distribution profile for six benchmark circuits calculated using the methodology that will be explained in Sec. 3.3. The soft error susceptibility of each node in the circuit is normalized with respect to the node with the largest soft error susceptibility. The x-axis shows increasing amount of soft error susceptibility from left to right, and the y-axis shows the number of nodes with that amount of susceptibility. These profiles illustrate the fact that certain nodes of the circuit have greater soft error susceptibility than others, and that a large number of nodes have negligible soft error susceptibility and can be effectively ignored when inserting fault tolerance features. These profiles were obtained assuming random input vectors, however, if real input traces from normal system workloads were used, the profiles would be even more skewed. Conventional approaches focus on protecting every node in the circuit, but more cost-effective approaches can be achieved by focusing only on the nodes with high soft error susceptibility.

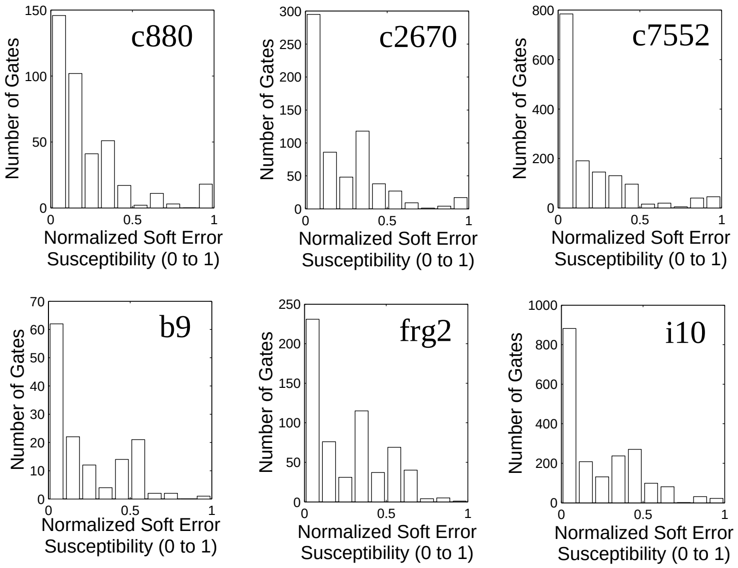


Figure 3.1 Soft error susceptibility profile for 6 benchmark circuits

3.2.3 Soft error failure rate modeling and estimation techniques

A key component to the proposed methodology is the capability to estimate the soft error susceptibility of nodes in a logic circuit and for the logic circuit as a whole. A lot of previous work has been done in this area. A comprehensive modeling program to predict the sensitivity of circuits to alpha-particles that was developed in [Sai-Halasz 82] formed the basis for many cosmic ray soft error rate modeling programs such as the Soft Error Monte Carlo Modeling Program (SEMM) [Murley 96]. A methodology to characterize the SEU rate in CMOS SRAM circuits (based on a verified empirical model for a 0.60 micron process technology) was presented in [Hazucha 00a]. A methodology to scale the results of this study to other feature sizes was presented in [Hazucha 00b]. There has been some previous work in the area of soft error susceptibility analysis of combinational and sequential circuits. A VHDL simulator that can inject and analyze soft faults in synthesized VHDL descriptions of synchronous logic circuits and the results of a study on a custom-designed bit-slice processor were presented in [Massengill 98], [Massengill 00]. An efficient transient fault injection and simulation technique that can be used to evaluate the soft error susceptibility of a design has been described in [Alexandrescu 02]. The effect of technology scaling and high-performance microprocessor design trends on the soft error rate in CMOS memory and logic circuits was presented in [Shivakumar 02]. The reader is referred to [Ziegler 96] for a comprehensive survey of the history of the study of soft errors in integrated circuits.

3.3 Proposed soft error failure rate estimation methodology

In this paper, we build on some of the previous research and develop a model that can be applied efficiently on a gate-level synthesized netlist of the design. After a design has been mapped to a standard cell library, each of the nodes (gates) in the netlist can be characterized individually to determine their soft error susceptibility. By then analyzing the interconnection of nodes (gates) in the netlist, the overall soft error susceptibility and the soft error failure rate of the design can be determined. Computing the soft error susceptibility for a node *n* with respect to latch *l* requires calculating three factors: (1) $R_{SEU}(n)$ – the rate at which a SEU of sufficient strength to change the logic value occurs at node *n*, (2) $P_{sensitized}(n,l)$ – the probability that node *n* is functionally sensitized to latch *l*, and (3) $P_{latched}(n,l)$ – the probability that the SEU at node *n* is captured in latch *l*. The soft error susceptibility for node *n* with respect to latch *l* (the rate at which soft errors are generated at latch *l* due to SEUs at node *n*) is the product of these three factors:

$$\text{Soft error susceptibility of node } n \text{ with respect to latch } l = \\ R_{SEU}(n) \bullet P_{sensitized}(n,l) \bullet P_{latched}(n,l)$$

The calculation of each of these three factors is discussed below:

1) $R_{SEU}(n)$, the rate at which a SEU of sufficient strength to change the logic value occurs at node *n:*

This depends on the device characteristics of the gate driving node *n*, the amount of capacitance at node *n*, as well as the sensitive area of node *n*. Two methods of calculating $R_{SEU}(n)$ in avionics were presented in [Normand 96]. Both methods are directly applicable to SEU rate calculations under terrestrial conditions when the variation in neutron flux at ground-level is taken into consideration. The first method, called the neutron cross-section (NCS) method, uses the neutron/proton SEU cross-section measured for the device, while the second method uses heavy-ion SEU data via the burst-generation-rate method proposed in [Ziegler 79] to calculate the SEU rate for the device. We follow the NCS method in this paper. The neutron SEU cross-section is defined as the probability that a neutron of energy E_x can produce an upset in a device in units of cm^2/device. The SEU rate for a node $R_{SEU}(n)$ using the NCS method is given by:

$$R_{SEU}\left(n\right)=\int_{E_{\min}}^{\infty}\sigma_{nSEU}\left(E_x\right)\left(\frac{dN}{dE_x}\right)dE_x$$

where $\sigma_{nSEU}(E_x)$ is the neutron SEU cross-section of the device and $\left(dN/dE_x\right)$ is the differential neutron flux (note that the flux varies depending on altitude, latitude, etc.). For a logic circuit, it is possible to obtain $\sigma_{nSEU}(E_x)$ for node n by characterizing the cell library. The integration is then done for all particle energies greater than the minimum particle energy E_{min} needed to create a voltage pulse of sufficient strength to change the logic value.

The minimum energy, E_{min}, can be determined using the charge to voltage pulse model developed in [Freeman 96]. The accuracy of the calculation of E_{min} will depend on the amount of design information available. If layout information is available, then a very accurate measure of E_{min} can be obtained from SPICE simulations where waveforms corresponding to increasing particle energies are injected at node n until it is sufficient to change the logic value. If only a technology mapped netlist is available, then the library cell that is used in the netlist can be characterized in SPICE. We use the following approximation to arrive at E_{min}. The critical charge Q_c at a node is determined using SPICE simulations. Increasing amounts of charge – modeled by current pulses of increasing magnitude and duration – are inserted till the output of the node changes. Once the critical charge is determined in this manner, we assume that all the energy deposited by the particle was used to generate the charge. Thus the critical charge Q_c is formed by deposition in the critical volume of energy E_{min} given by

$$E_{min}=\frac{3.6\text{eV}\bullet Q_c}{1.6\bullet 10^{-19}}\text{C}$$

where 3.6eV is the energy required to generate an electron-hole pair in silicon.

For the differential neutron flux $\left(dN/dE_x\right)$, we use the analytic approximation of the differential neutron flux in New York City over the 1MeV to 10000MeV range [Bradley 98]. We then use a piecewise summation of the above integral formulation of the NCS method in intervals of 10MeV starting from E_{min} to obtain $R_{SEU}(n)$.

2) $P_{sensitized}(n,l)$, the probability that node *n* is functionally sensitized to latch *l*:

Whether or not node *n* is sensitized to latch *l* depends on the input pattern being applied. Thus, the probability that node *n* is sensitized to latch *l* depends on the probability of each input pattern being applied to the circuit while it is operating. A fast and efficient way to calculate $P_{sensitized}(n,l)$ is to simply simulate the system with a typical workload for some number of clock cycles. For each clock cycle, critical path tracing [Abramovici 83] can be performed starting from each latch to identify all of the nodes that are sensitized to it (alternatively fault injection and simulation can be used for the same purpose). Note that there can be some nodes in the circuit which are only sensitized to a latch for very few input patterns and hence may not get sensitized at all during the simulation. However, these nodes will have a negligible affect on the overall soft error rate (since their probability of being sensitized is extremely low) and hence can be ignored. A less accurate alternative to simulating the system with a typical workload would be to just apply random patterns at the primary inputs to get a rough estimate.

3) $P_{latched}(n,l)$, the probability that the SEU is captured in latch *l*:

In order to be captured in latch *l*, the pulse created by the SEU must arrive at the latch during the latching-window in time. The probability of the pulse being present during the latching-window depends on the width of the pulse relative to the clock period. The width of the pulse depends on the amount of particle energy. By taking attenuation through the propagation path into consideration (through a characterization of the library cells for different voltage pulse inputs), the width of the pulse as it reaches the latch can be determined for different particle energies. By comparing this with the total clock period, the probability of the pulse being latched can be computed.

3.4 Partial duplication for CED

Conventional schemes to design circuits with CED based on error-detecting codes such as parity, duplication and compare, etc. employ checkers to monitor the outputs for the occurrence of an error. Figure 2.2 shows the structure of a circuit that has CED capability. Since the partial duplication scheme is a variant of duplication and compare, the check symbol generator is a partial copy of the original circuit and the checker is a two-rail code checker.

3.4.1 Intuition for the partial duplication method

The proposed method is based upon the observation that in the presence of the three masking factors described in Sec. 3.2, the soft error susceptibility of certain nodes in the logic circuit can be orders of magnitude higher than that of the other nodes in the design. The second observation is that these nodes tend to be located closer to the primary outputs. Thus, nodes that are several levels of logic from the primary outputs have a comparatively lower soft error susceptibility than nodes close to the primary outputs. Thus, the proposed heuristic involves selecting a cluster of nodes (henceforth the "cutset") near the primary outputs whose logic is duplicated as shown in Fig. 3.2.

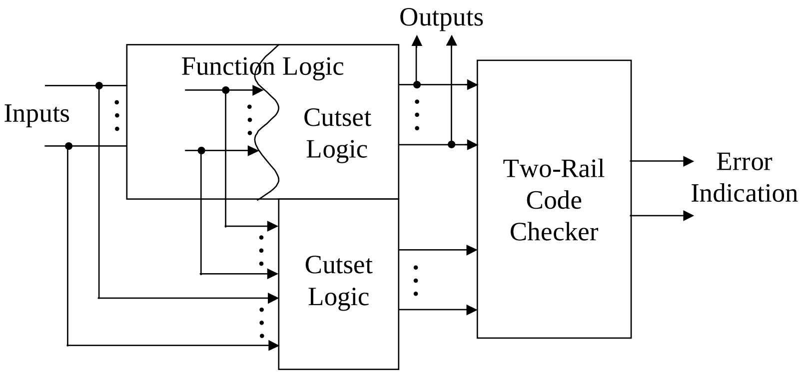


Figure 3.2 CED with partial duplication

The inputs to the duplicated logic of the cutset are taken from the design. Thus, if a particle strike occurs in the non-duplicated portion of the design and is of sufficient strength, the resulting SEU will (in the presence of a sensitized path) propagate to the outputs of both the cutset and the function logic and go undetected. However, any SEU from a particle strike that occurs at a node in the partially duplicated portion of the circuit (in either the function logic or the cutset) will be detected at the outputs of the checker. Note that if the path from the node where the SEU occurs to the outputs contains nodes that are not duplicated, the SEU propagates to both sets of outputs and is hence not detected (an example is provided in Sec. 3.4.2). By carefully selecting the cutset, the nodes with the highest soft error susceptibility will be in the partially duplicated portion of the circuit thereby giving a very cost-effective reduction in the soft error failure rate.

3.4.2 Algorithm for partial duplication

A heuristic algorithm for partial duplication that generates a cutset with specified area overhead is described below. The basic idea of the heuristic is to traverse the circuit from the primary outputs to the primary inputs in a greedy manner to generate the cutset. A priority queue (indexed by soft error susceptibility) of nodes that can be added to the current cutset of nodes in a consistent manner is maintained. At each iteration of the heuristic, the node *n* with the highest soft error susceptibility is removed from the head of the priority queue and added to the cutset. The current area overhead is updated to reflect the latest addition to the cutset and all gates that are inputs to node *n* are added to the priority queue. This process terminates when the size of the cutset equals (or just exceeds) the specified area overhead.

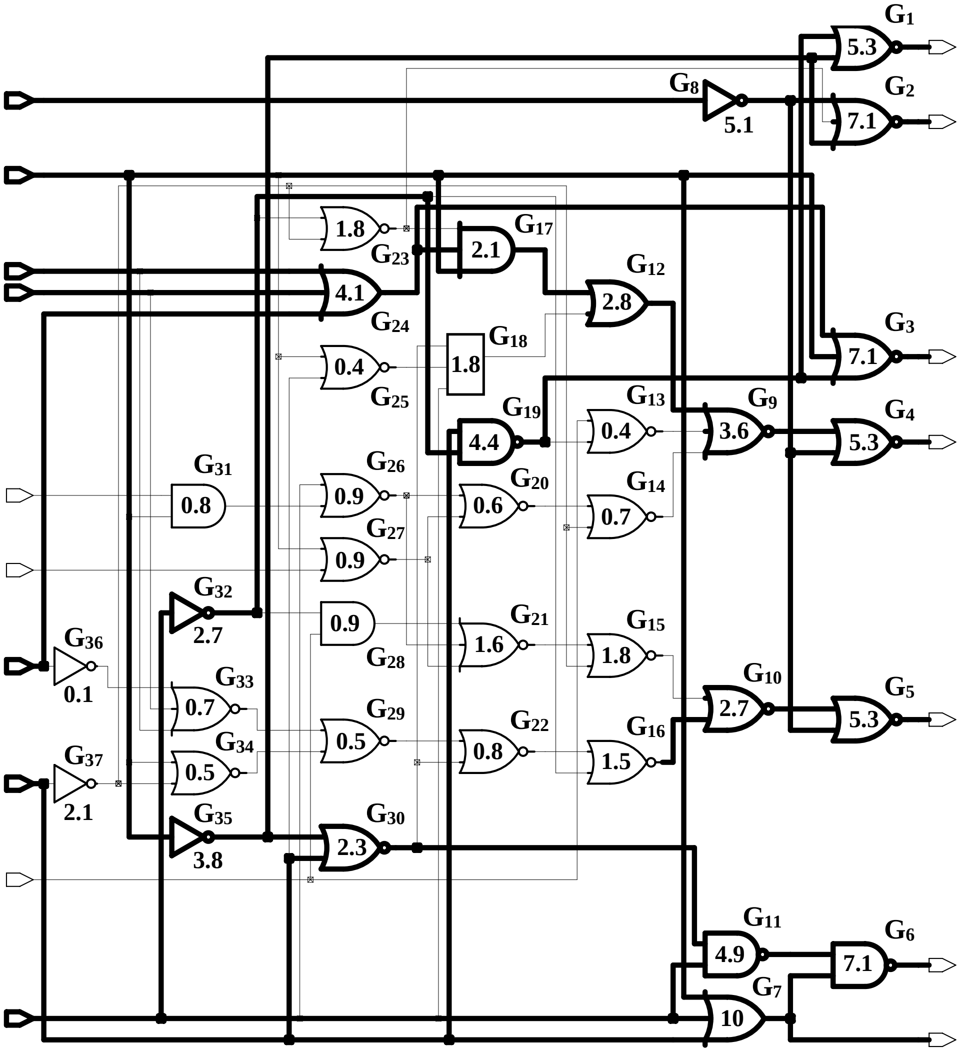


Figure 3.3 Screen shot with gates selected for partial duplication

We describe the steps of the heuristic using the following example. In Fig. 3.3, we present a screenshot of a small circuit where the gates that have been selected for partial duplication by the proposed algorithm have been highlighted. The soft error susceptibility for each of the gates in the design (to 2 significant digits) is also provided. The only constraint to the proposed algorithm is the overhead that is allowed for partial duplication. The following are the steps of the algorithm.

<u>Step 1</u> – A priority queue *gateQ* that is indexed by the soft error susceptibility of nodes is initialized. All the primary outputs of the circuit (G_1 to G_7 in Fig. 3.3) are

inserted into *gateQ*. The heuristic processes gates starting from the primary outputs for two reasons. The first is based upon the observation that the primary outputs have a very high soft error susceptibility since they are always sensitized. The second is that checking is performed on the primary outputs. As a result, the cutset has to always be "consistent" – all gates selected for partial duplication in the cutset must have at least one path to the primary outputs that is entirely contained within the cutset. Thus for a gate to be added to the cutset, at least one of the gates that it fans out to must already be a part of the cutset (and hence, by induction, have at least one path entirely contained in the cutset to the primary outputs). Adding all primary outputs to *gateQ* during initialization ensures that the cutset is generated in a consistent manner. For the example, at the end of the initialization procedure, node G_7 with a normalized soft error susceptibility of 10 is at the head of the priority queue.

```
/* netlist – technology mapped design with soft error susceptibility data
   overhead – area overhead constraint ( overhead < area ( netlist ) )
   gateQ – priority queue initialized with all primary outputs */

while ( ( is_not_empty ( gateQ ) )  ||  ( current_cost < overhead ) ) {
        node  =  top ( gateQ ) ;
        pop ( gateQ ) ;
        mark ( node ) ;
        current_cost  +=  area ( node ) ;
        for_each_fanin ( node , fanin ) {
                if_not_marked ( fanin ) {
                        insert ( gateQ , fanin , soft_error_susceptibility ( fanin ) ) ;
                }
        }
}
```

Figure 3.4 Pseudo-code for iterative phase

<u>Step 2</u> – The pseudo-code for the iterative process of growing the cutset is described in Fig. 3.4. The first two passes of the iterative phase of the heuristic are as follows. Gate G_7 is popped and marked as a node that belongs to the cutset. The cost of the cutset at this point is equal to the area of the gate G_7. Since G_7 is driven only by the primary inputs, there are no gates in its immediate fanin that need to be processed.

The next gate that is popped from *gateQ* is G_6. Note that if there are several gates with the same soft error susceptibility, the ones with the smallest area are processed first. Of the gates in G_6's immediate fanin list, G_7 has already been added to the cutset. G_{10} is inserted into *gateQ*. In this manner, the cutset grows until the cost of all the gates added to the cutset exceeds the specified area overhead constraint.

Coverage estimation is run to see if the reduction in the soft error failure rate meets requirements. Care has to be taken during soft error failure rate estimation to ensure that a SEU that is propagated along a path that does not entirely lie within the cutset is accounted for. An example of such a case is a SEU that propagates along the path $(G_{19} \rightarrow G_{13} \rightarrow G_9 \rightarrow G_4)$.

It may also be the case that a buffer or an inverter is driven by a node of considerably higher soft error susceptibility. In order to ensure that this is taken into consideration whenever a node's immediate fanin is inserted into *gateQ*, a buffer (or inverter) is indexed by the sum of the soft error susceptibilities of the buffer (or inverter) and the node that drives the buffer.

3.5 Experimental results

The synthesis tool used for all technology mapping and optimization in this paper was Synopsys' Design Analyzer. The technology library used is the 0.25 micron library distributed by Virginia Tech [Sulistyo 02]. The combinational benchmark circuits were chosen from the LGSynth91 suite [Yang 91]. A framework for the soft error failure rate estimation methodology described in Sec. 3 was implemented in C++.

Table 3.1 presents the reductions in the soft error failure rate that we achieved using the proposed partial duplication scheme. Under the first major heading, we provide details about the circuits that were chosen – name, number of primary inputs, and number of primary outputs. Under the second major heading, we present the reduction in the soft error failure rate that was observed when the area overhead constraint for CED is 20%, 33%, and 50% respectively. The soft error failure rate reduction percentage was computed as:

$$\left(\frac{\text{Original Failure Rate - Reduced Failure Rate}}{\text{Original Failure Rate}}\right)\times 100\,\%$$

The last row presents the average reduction in the soft error failure rate that is observed using the proposed scheme. Note that an order of magnitude reduction in the soft error failure rate can generally be achieved with a 50-60% overhead. A factor of 4 reduction can generally be achieved with 33% overhead, and more than a factor of 2 reduction can be achieved with 20% overhead.

Table 3.1 Soft error failure rate reduction using partial duplication

Sof Error Failure Rate Reduction (%)					
Circuit			Area Overhead		
Name	No. PIs	No. POs	20%	33%	50%
C2670	233	140	53.9	77.7	89.5
C3540	50	22	49.9	68.6	83.1
C5315	178	123	57.3	68.5	83.8
C7552	207	108	52.3	73.6	88.7
x1	51	35	73.9	90.8	95.5
c880	60	26	56.1	73.3	83.6
b9	41	21	59.8	87.9	95.0
i10	257	224	54.3	68.8	85.1
frg2	143	139	65.4	78.0	90.5
Average Reduction			58.1	76.3	88.3

In Fig. 3.5, we present a graph of the reduction in soft error failure rate that is achieved versus the allowed area overhead for all 9 benchmark circuits. The average of the reductions over all the benchmark circuits versus area overhead is provided by a continuous curve in the figure. Depending on the soft error failure rate requirements for a particular application, the appropriate point on this curve can be selected.

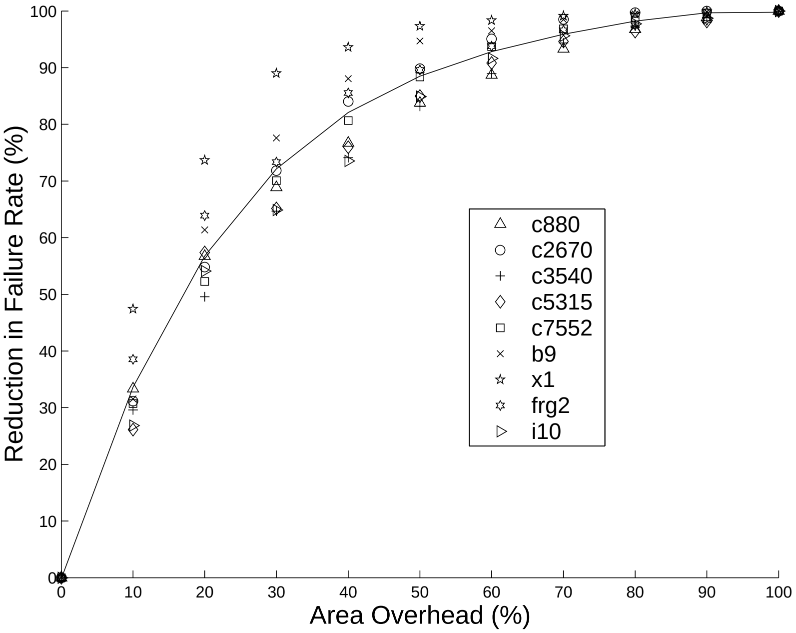


Figure 3.5 Reduction in soft error failure rate (%) for 9 benchmark circuits

3.6 Conclusions

In the future, as the soft error failure rate of logic circuits becomes unacceptably high even for mainstream applications, CED will become necessary for logic circuits. This paper described a promising new paradigm for designing logic circuits with cost-effective CED by exploiting the asymmetric soft error susceptibilities of nodes. The partial duplication approach described here is one particular approach for accomplishing this. An area for future research is to investigate other techniques that can be selectively targeted towards the most susceptible nodes.

Chapter 4. Partial Error Masking

The CED schemes discussed so far use error detection and retry with CED circuitry to reduce the soft error failure rate in logic circuits. If an error is detected using the CED circuitry, the system recovers through rollback and retry thereby preventing a failure. An alternative to error detection and retry is error masking, that involves using circuitry that masks (i.e., corrects) errors. Examples of such schemes include quadded logic [Tryon 62], interwoven logic [Pierce 65], and triple modular redundancy (TMR) [Siewiorek 98]. While error detection and retry is more commonly used, there are scenarios where error masking is advantageous. For real-time systems, it may not be possible to do retry, thus error masking is the only option. In some applications, the cost of implementing the rollback and retry functionality – state storage as well as control – may be comparable to (or exceed) that of implementing error masking. This chapter focuses on error masking techniques to reduce the soft error failure rate in logic circuits.

In Chapter 3, we presented a new paradigm to synthesize CED circuitry for error detection and retry based on partial duplication of a logic circuit. It was shown that SEUs at some internal nodes in logic circuits can have orders of magnitude greater probability of being latched and causing an error than at other nodes. By focusing CED capability towards the nodes that are most susceptible to SEUs, the soft error failure rate in logic circuits was significantly reduced at a fraction of the cost of existing techniques that try to guarantee coverage of all nodes. In this chapter, we present a new approach based on partial error masking to reduce the soft error failure rate in logic circuits [Mohanram 03c]. The proposed algorithm is composed of two reduction techniques, cluster sharing reduction and dominant value reduction, and achieves a very high reduction in the estimated soft error failure rate within the specified overhead constraints. This allows cost-effective tradeoffs between overhead and soft error failure rate reduction. The proposed technique scales very well for large circuits and is highly compatible with standard synthesis flows.

4.1 Partial error masking

The partial error masking scheme proposed here is based on TMR. TMR is the simplest error masking scheme that uses three functionally equivalent copies of the logic circuit and a 2-out-of-3 majority voter. Errors are masked and hence tolerated. Figure 4.1 shows the structure of a circuit that has error masking capability based on TMR. The associated overhead is greater than 200% when the voter circuitry is also taken into consideration. The hardware overhead of conventional TMR, that targets all modeled faults in the logic circuit, exceeds 200%. The proposed method for partial error masking uses a combination of two reduction techniques, cluster sharing reduction and dominant value reduction, to reduce a the overhead costs associated with TMR while minimizing the soft error failure rate. The reduction techniques are described separately in Secs. 4.1.1 and 4.1.2.

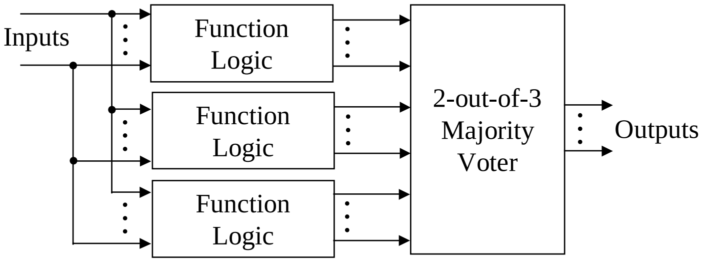


Figure 4.1 Block diagram for TMR-based error masking

4.1.1 Cluster sharing reduction

Cluster sharing reduction is based upon two observations. The first is that in the presence of the three masking factors described in Sec. 3.2, the soft error susceptibility of certain nodes in the logic circuit can be orders of magnitude higher than that of the other nodes in the design. The second is that these nodes tend to be clustered together, with low observability and controllability values. Thus, the cluster sharing reduction heuristic involves selecting such clusters of nodes (in a consistent manner), so that the logic can be shared across the three copies used to realize TMR. The clusters are removed from two out of the three copies of the TMR design and they are driven by the cluster from a single copy only. The inputs to the triplicated logic of two copies of the function logic are taken from the full copy of the design. The complementary set of nodes, i.e., the nodes that are not part of any cluster and hence actually triplicated, are the ones with a high soft error failure rate. Since they will be masked by the TMR design, it is possible to achieve a significant reduction in the soft error failure rate in this manner.

If a particle strike occurs in the non-triplicated portion of the design and is of sufficient strength, it will (in the presence of a sensitized path) propagate to the outputs of all three copies and thus go undetected. However, any particle strike that occurs on a node in the triplicated logic portion of the circuit will be masked at the 2-out-of-3 voter. By carefully selecting the clusters of nodes with low soft error susceptibility, the nodes with the highest soft error susceptibility will be in the triplicated logic portion of the circuit thereby giving a very cost-effective reduction in the soft error failure rate.

```
/* netlist – technology mapped design with soft error susceptibility data
   overhead – area overhead constraint ( overhead < area ( netlist ) ) */
   lowSusceptibilityQ – priority queue of gates indexed by soft error susceptibility */

while ( ( is_not_empty ( lowSusceptibilityQ ) )  ||  ( current_cost < overhead ) {
      node  =  top ( lowSusceptibilityQ ) ;
      pop ( lowSusceptibilityQ );
      mark ( node );
      current_cost  +=  area ( node );
      for_each_fanin ( node, fanin ) {
            if_not_marked ( fanin ) {
                  insert ( low_susceptibilityQ, fanin, soft_error_susceptibility ( fanin ) );
            }
      }
      for_each_fanout ( node, fanout ) {
            if_not_marked ( fanout ) {
                  insert ( low_susceptibilityQ, fanout, soft_error_susceptibility ( fanout ) );
            }
      }
      if ( overhead > threshold ) {
            make_consistent ( network ) ;
      }
}
```

Figure 4.2 Pseudo-code for cluster sharing reduction for error masking

A heuristic algorithm for cluster sharing to identify such clusters of logic gates with negligible soft error failure rates is presented in Fig. 4.2. The basic idea is to maintain the priority queue *lowSusceptibilityQ* (indexed by soft error susceptibility) of nodes that can be added to the current set of nodes in a consistent manner. Nodes with the lowest soft error susceptibility are at the head of *lowSusceptibilityQ*. It is possible that once a node is added to the cluster, all its fan-in nodes need to be added since they fan-out to only the selected node. This is accomplished by running a consistency routine to eliminate such orphan nodes by including them into the cluster. The consistency routine is triggered whenever the current set of nodes grows by a certain predetermined threshold.

In Fig. 4.3, we present a screenshot of a small circuit where the gates that have been selected by the proposed cluster sharing algorithm have been highlighted. The values of the soft error susceptibility for each of the gates in the design (to 2 significant digits) is also provided. G_1 is the first gate that is popped from *lowSusceptibilityQ* and added to the cluster. The cost of the cluster is equal to the area of G_1. Its fanin (only inputs that are ignored) and fanout are added to *lowSusceptibilityQ*. The gates are numbered up to 9 in the order that they are added to the cluster. The overhead limit was set to 50% and all gates that are part of the final cluster are highlighted.

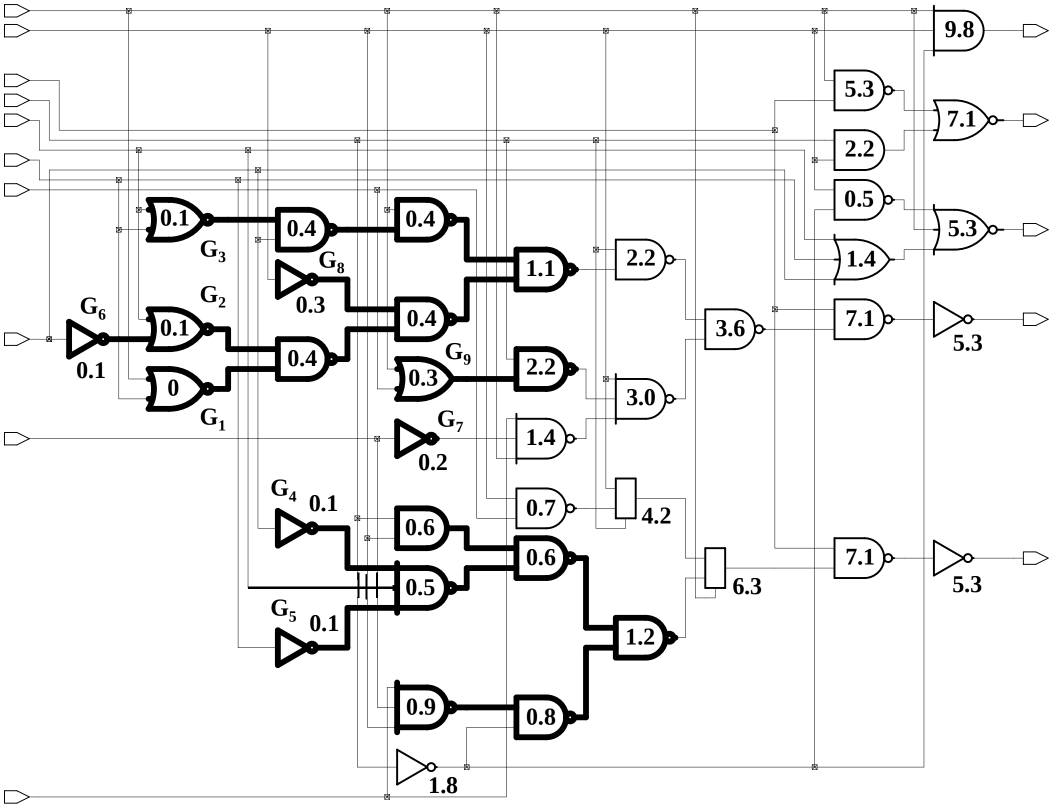


Figure 4.3 Screen shot with gates selected for cluster sharing

4.1.2 Dominant value reduction

Dominant value reduction differentiates between the logic 0 and logic 1 soft error susceptibility of a primary output. It exploits the fact that the logic 0 and logic 1 soft error susceptibility of certain primary outputs is highly skewed, i.e., the soft error failure rate at an output when it is at logic 0 (logic 1) is close to an order of magnitude higher that when it is at logic 1 (logic 0). The idea is to identify such outputs and replace triplication by duplication in such instances. The 2-out-of-3 majority voter is replaced by an AND (OR) logic gate. Note that the resultant loss in coverage is 2 times the soft error failure rate of the non-dominant logic value at the primary output, as explained below.

The heuristic algorithm for dominant value reduction is as follows. The soft error failure rate at the outputs of the circuit when each of the primary outputs has a logic 0 or logic 1 value are computed to begin with. For example, let primary output O_i have a logic

1 soft error failure rate that is an order of magnitude higher that its logic 0 soft error failure rate. The TMR-based traditional approach guarantees complete error masking to all single particle strikes to the function logic. Consider the approach to error masking where just two copies of the logic circuit are used and the 2-out-of-3 majority voter is replaced by an OR gate as shown in Fig. 4.4. Any particle strike that cause a 1 → 0 → 1 transient to appear at the output O_i is guaranteed to be masked. However, a 0 → 1 → 0 transient will not be masked. Thus, while the logic 1 soft error failure rate of the primary output O_i is reduced to 0 (through dominant value masking), the logic 0 soft error failure rate is not. Since the logic 1 soft error failure rate is an order of magnitude higher than the logic 0 soft error failure rate, the reduction in soft error failure rate is given by

$$\left(\frac{\text{Original Failure Rate} - (2 \cdot \text{Logic 0 Failure Rate})}{\text{Original Failure Rate}}\right)\times 100\,\% = \left(\frac{11 - (2 \cdot 1)}{11}\right)\times 100\,\% = 81\%$$

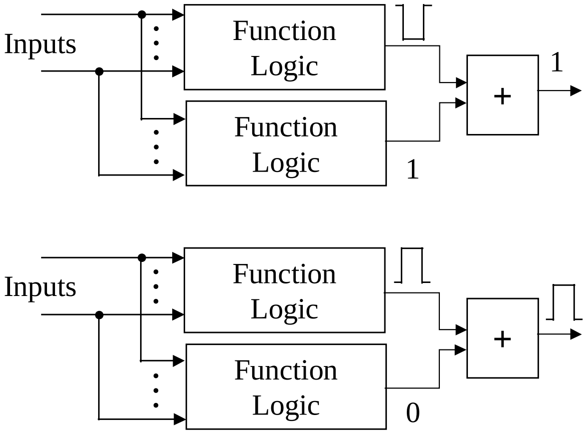


Figure 4.4 Example for dominant value reduction for error masking

Note that the original soft error failure rate is the sum of the logic 0 and logic 1 soft error failure rates for a single copy of the circuit. The logic 0 soft error failure rate is multiplied by 2 since particle strikes to either of the copies of the logic contribute to the soft error failure rate of the partially error masked implementation. The pseudo-code for this heuristic algorithm is presented in Fig. 4.5.

```
Algorithm dominant_value_reduction ( netlist, max_failure_rate ) {
    /* netlist – technology mapped design with soft error susceptibility data
       max_failure_rate – maximum acceptable failure rate */
    list skewed_list  =  sort ( primary_output_list by susceptibility skew ) ;
    for_each_primary_output ( skewed_list, primary_output ) {
        failure_rate  =  evaluate_failure_rate (netlist, dominant_value ( primary_output) ;
        if ( failure_rate > ( max_failure_rate ) ) { break; }
    }
}
```

Figure 4.5 Pseudo-code for heuristic algorithm for dominant value reduction

4.1.3 Partial error masking

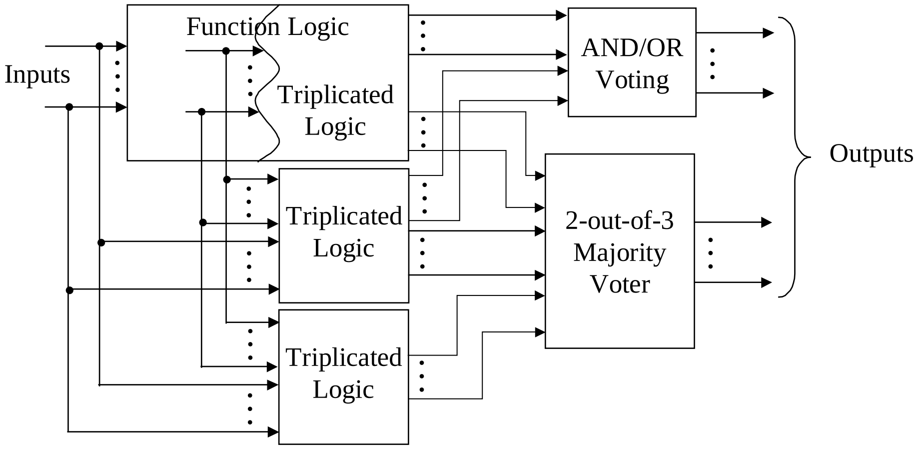


Figure 4.6 Partial error masking

The partial error masking scheme proposed in this paper utilizes a combination of the two reduction procedures that are described above to achieve a reduction in the soft error failure rate. It starts with a TMR realization for error masking that is first reduced using cluster sharing. The soft error failure rate of the resulting implementation is then estimated along with the area overhead. Note that the need to run the consistency routine (during cluster sharing reduction) does not a give precise control over the area overhead. To further reduce the area overhead, dominant value masking, where all primary outputs with a skew ratio above a specified threshold are selected is run. Since the soft error failure rate after dominant value masking is performed increases, more than one pass may be necessary before the soft error failure rate and area overhead become acceptable.

4.2 Experimental results

The synthesis tool used for all technology mapping and optimization in this paper was Synopsys' Design Analyzer. The technology library used is the 0.25 micron library distributed by Virginia Tech [Sulistyo 02]. The combinational benchmark circuits were chosen from the LGSynth91 suite [Yang 91]. The framework implementing the soft error failure rate estimation methodology described in Chapter 3 was used here.

Table 4.1 presents the reductions in the soft error failure rate that we achieved using the proposed cluster sharing reduction scheme by itself for partial error masking. Under the first major heading, we provide details about the circuits that were chosen – name, number of primary inputs, and number of primary outputs. Under the next two pairs of columns, we present the reduction in soft error failure rate that was observed along with the area overhead that was necessary to achieve this. The soft error failure rate reduction percentage was computed as:

$$\left(\frac{\text{Original Failure Rate - Reduced Failure Rate}}{\text{Original Failure Rate}}\right)\times 100\,\%$$

It is clear from the results that over an order of magnitude reduction in soft error failure rate can be achieved with 128% overhead on average (compared with conventional TMR which requires 200% overhead). Note that with not much more overhead than what is needed for duplicate-and-compare, the proposed method corrects the errors on-the-fly thereby requiring no rollback and retry mechanism as is needed for error detection and retry schemes. Thus, it is advantageous in terms of performance (and can be used for real-time systems), moreover, in some situation the overall area overhead will be less since the rollback and retry circuitry can require significant overhead itself.

Table 4.1 Soft error failure rate reduction using cluster sharing reduction

Circuit			Failure Rate Reduction	Area Overhead	Failure Rate Reduction	Area Overhead
Name	No. PIs	No. POs				
C2670	233	140	95.6	129.8	86.3	103.8
C3540	50	22	90.4	127.2	83.1	103.7
C5315	178	123	85.9	126.6	82.6	104.0
C7552	207	108	95.0	128.8	89.8	103.7
x1	51	35	95.4	128.4	93.7	103.8
c880	60	26	89.6	128	83.5	99.8
b9	41	21	96.3	128.8	91.1	101.6
Average			92.6	128.2	87.2	102.9

Figure 4.7 presents the reductions in the soft error failure rate that we achieved using the proposed dominant value reduction scheme by itself for partial error masking. On the x-axis, the number of skewed outputs that were chosen increases from 0 to 10. On the y-axis, the normalized failure rate increases from 0% (when all outputs are triplicated) to the normalized value (when 1 to 10 outputs are chosen).

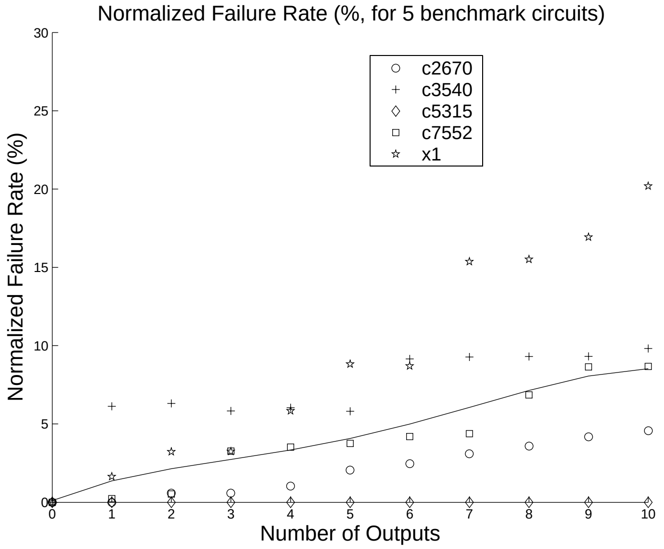


Figure 4.7 Number of outputs chosen versus soft error failure rate reduction for dominant value reduction

Figure 4.8 presents the reductions in the overhead achieved using the proposed dominant value reduction scheme by itself for partial error masking. When no output is chosen, the overhead is 200%, since two full copies of the logic circuit are necessary.

(Note that we ignore the overhead of the voter throughout.) As more outputs are selected for dominant value masking, one of the three copies of the logic circuit no longer requires the logic that belongs exclusively to the transitive fan-in cone of the dominant value masked primary outputs. Thus, the overhead ranges on the y-axis from 200% (no primary outputs chosen) to 100% (all outputs chosen). Note that the effectiveness of dominant value reduction varies considerably from circuit to circuit. This is because it depends on the extent of the signal probability skew and the amount of logic sharing between the relevant output cones of logic.

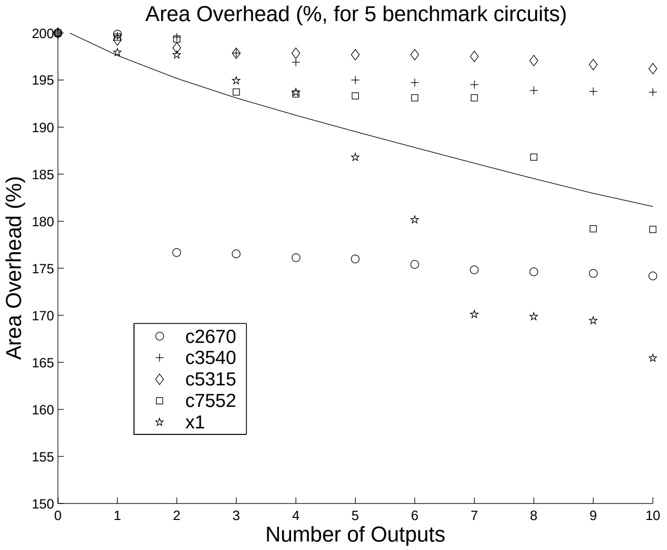


Figure 4.8 Number of outputs chosen versus overhead and soft error failure rate reduction for dominant value reduction

Table 4.2 presents the reductions in the soft error failure rate and overhead that we achieved using the partial error masking scheme described in Sec. 3.3. Under the first major heading, we provide details about the circuits that were chosen – name, number of primary inputs, and number of primary outputs. Under the next two pairs of columns, we present the reduction in soft error failure rate that was observed along with the overhead that was necessary to achieve this. In both cases, we ran cluster sharing reduction to reduce the circuit to the soft error failure rate and overhead described in Table 1. This was followed by dominant value reduction, where 10 outputs with the largest skew were selected. In our experiments, we have observed the optimum number of outputs that need to be selected for dominant value reduction varies from one logic circuit to the other. For example, C2670 showed a 86.3% reduction in the soft error failure rate with 103.8% overhead when only cluster sharing reduction is applied. However, when partial error masking is performed, we see that a 91.0% reduction in soft error failure rate can be achieved with 104% overhead. In this case, the combination of cluster sharing and dominant value reductions is clearly advantageous. A similar observation can be made for the benchmark C5315. The partial error masking algorithm proposed in Sec. 3.3 thus uses an iterative process (of cluster sharing reduction and dominant value reduction) to meet soft error failure rate and overhead requirements.

Table 4.2 Soft error failure rate reduction using partial error masking

Circuit			Failure Rate Reduction	Area Overhead	Failure Rate Reduction	Area Overhead
Name	No. PIs	No. POs				
C2670	233	140	91.0	104.0	81.7	78.0
C3540	50	22	81.6	120.9	74.7	97.4
C5315	178	123	85.9	122.8	82.6	100.2
C7552	207	108	86.6	108.0	81.2	82.9
x1	51	35	77.4	94.0	75.7	69.4
c880	60	26	85.0	124	79.0	95.8
b9	41	21	71.3	77.8	66.1	50.5
Average			82.7	107.4	77.3	82.1

4.3 Conclusions

In the future, as the soft error failure rate of logic circuits becomes unacceptably high even for mainstream applications, it will become necessary to incorporate error masking features into logic circuits. In some applications, error masking is advantageous to error detection and retry. For such applications, a partial error masking scheme to realize cost-effective fault tolerance by exploiting the asymmetric soft error susceptibilities of nodes was described in this chapter.

Chapter 5. Input Ordering in Concurrent Checkers to Reduce Power Consumption

In this chapter, a novel approach for reducing power consumption in checkers used for CED is presented [Mohanram 02b]. Spatial correlations between the outputs of the circuit that drives the primary inputs to the checker are analyzed to order them such that switching activity, and hence power consumption, in the checker is minimized. An algorithm to build a reduced cost function used in the optimization problem of determining the optimal order is presented. Use of the reduced cost function results in runtime savings of at least an order of magnitude, with minimal degradation in the quality of the solution. The reduction in power consumption comes at no additional impact to area or performance, and does not require any alteration to the design flow.

5.1 Concurrent checkers and input ordering

Conventional schemes to design circuits with CED based on error-detecting codes such as parity, duplication and compare, etc., employ checkers to monitor the outputs for the occurrence of an error. Figure 2.2 shows the structure of a circuit that has CED capability. Based upon the scheme chosen for CED, the check symbol generator can be a copy of the original circuit (duplication and compare), parity prediction logic, codeword generator (e.g., for Berger or Bose-Lin codes), etc. The check symbol generator generates check bits and the checker determines if they form a codeword. The checker usually satisfies the totally self-checking (TSC) property [Smith 78]. Examples of commonly used checkers include parity checkers, two-rail code checkers, Berger code checkers, m-out-of-n checkers, etc.

As indicated in Fig. 2.2, concurrent checkers for error detecting codes usually derive their inputs from the primary outputs of a circuit. These outputs can be connected to the inputs of the checker in any order, since most checkers have a regular structure and are functionally symmetric with respect to their inputs. In other words, the functionality of the checker is insensitive to various permutations (orderings) of the inputs.

In the presence of spatial and temporal correlations in the primary outputs of the circuit driving the checker, the input ordering presented to the checker can have a significant effect on power consumption in the checker. Correlation between vectors is of two types – spatial and temporal. Spatial correlation refers to the correlation between pairs of bits in the same input vector, while temporal correlation refers to the correlation between pairs of input vectors, spaced one or more cycles apart. Power estimation techniques usually make the spatial independence assumption about the primary inputs to a circuit and neglect spatial correlations between them [Najm 94]. A recent study of industrial circuits [Schneider 96] evaluated the accuracy of the correlation assumptions made by several power estimation methods. Large inaccuracies in total switching activity are reported when correlation between signals is neglected. Assuming spatial independence provides a very conservative estimate of power consumption for checkers. It also precludes the possibility that the inputs to the checker can be ordered to reduce power consumption. Given their symmetry, the grouping of the inputs to the checker can affect power consumption to a considerable extent due to spatial correlations in the inputs. In this paper, we address the problem of input ordering to checkers to minimize power consumption and present an algorithm that achieves this while taking spatial input correlations into consideration. We present experimental results for parity, two-rail code, and Berger code checkers over a wide range of benchmark circuits with our technique.

5.1.1 Previous work

Previous techniques that use reordering to reduce power consumption in CMOS gates work at the input and transistor levels. Input reordering methods permute only the inputs to the gates, while leaving the actual realization of the gate untouched. Transistor reordering methods modify the order in which series transistors are connected in a complex CMOS gate, in addition to input reordering. These methods list all possible configurations of a complex CMOS gate and evaluate them for power consumption. The number of configurations that are evaluated is usually small. There is usually a delay tradeoff involved in such techniques, since reordering can move late arriving inputs farther away from the output of the gate contributing to an increase in delay. In [Prasad 96], a multi-pass transistor reordering algorithm, with linear time complexity per pass, that converges to a solution in a small number of passes was presented. In [Musoll 96], an algorithm that includes transitions at the internal nodes of a complex CMOS gate to derive the optimal configuration was presented. Figure 5.1 illustrates equivalent CMOS implementations of a simple Boolean function using both input and transistor reordering techniques.

Transistor resizing techniques resize transistors subject to delay constraints to reduce power consumption. These techniques compute the slack at each gate in the circuit and process those with a positive slack. The sizes of the transistors in such gates are reduced until the slack reaches zero or the transistors reach minimum size. In [Tan 94], both the input reordering and transistor resizing approaches are combined together.

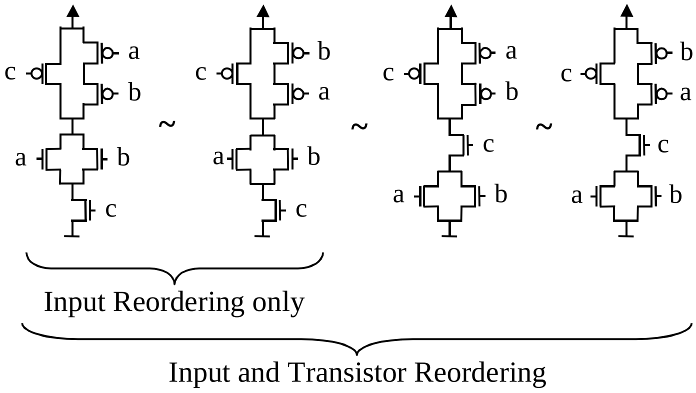


Figure 5.1 Input and transistor reordering [Musoll 96]

The proposed approach differs from previous reordering methods in several ways. Previous methods target general circuits and focus on input and transistor reordering at the individual gate level, whereas the proposed method targets checker circuits and focuses on input reordering at the module level. The magnitude of the problem addressed here is larger, since *n* inputs to a module imply *n!* possible permutations, however, the potential for improvement is much greater. Even after structural symmetry is accounted for, the number of distinct permutations renders prohibitive the costs of exhaustive enumeration and evaluation – this is discussed in greater detail in Sec. 5.1.2.

There has also been some work on the synthesis of checkers with low power consumption. In [Metra 96], a methodology to design tree checkers with low power-delay requirements was presented. In [Kavousianos 98], a methodology for designing Berger code checkers with near optimal transistor count, high speed, and low power consumption was presented. The proposed approach differs from these methods, since it does *not* consider the design of the checker in isolation from the circuit that drives it.

Thus the proposed approach can be used to obtain an optimal permutation that reduces power consumption in the checker. The reordering and checker synthesis techniques described above can be used independently – completely or partially – to obtain further reductions in power consumption, especially in the presence of structural asymmetries in the checker. The proposed approach thus complements other low power synthesis techniques, whether they target general circuits or checkers.

If the checkers are pipelined, all the inputs to the checker are ready at the same instant of time. In the absence of such pipelining, an extra dimension is added to the complexity of the problem. Even in the most balanced of designs, different paths have different delays, and the inputs to the checker will be ready at different times. This can contribute to glitches in the checker, since its inputs are not ready at the same instant, and the extra switching activity can increase power consumption. This is not addressed here.

5.1.2 Exponential complexity

In this section, we show that for a parity tree with *n* inputs, the number of unique permutations after structural symmetries are accounted for is exponential in *n*. We begin by analyzing an elementary example of a parity tree with four inputs shown in Fig. 5.2.

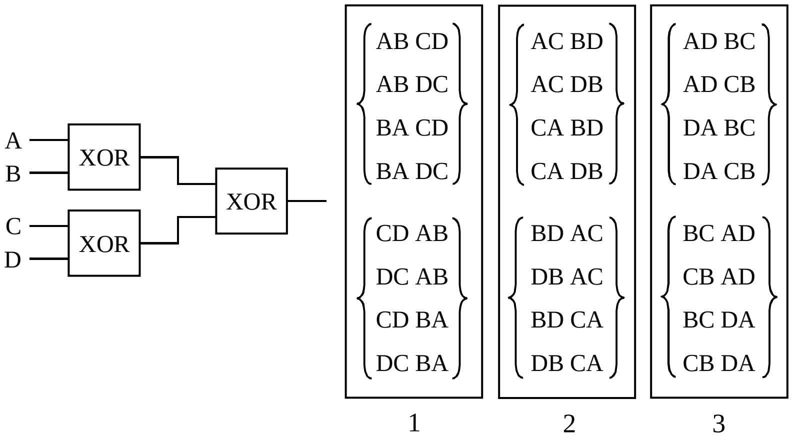


Figure 5.2 Structural symmetry in parity checkers

The number of unique permutations in the absence of any structural symmetries in the balanced parity tree for the parity checker is *4!* However, in the presence of structural symmetries at each level of logic in the checker, the number of unique permutations in actually 3. This is shown in greater detail in Fig. 5.2, where the different permutations are grouped on the basis of symmetry as observed at logic levels of the checker. It is clear that just 3 unique permutations need to be evaluated for power consumption to arrive at the optimal input order with the lowest power consumption. We now prove that for a general parity checker with n inputs realized using 2-input XOR gates, the number of unique input orders is exponential in n.

Lemma 1: For a parity tree with n inputs, the number of unique permutations is exponential in n for all $n \geq 8$.

Proof: Consider the XOR parity tree for n inputs. Without loss of generality, we will show that the number of unique permutations for i that is the largest power of 2 less than n is exponential in i, i.e., let $\left(i = 2^k\right) \leq n$. Since the number of unique permutations is a monotonically increasing function in the number of inputs to the checker, it follows that the number of permutations for arbitrary n is exponential in n, since $i \leq \left(n/2\right)$. The number of 2-input XOR gates in the balanced parity tree for i inputs is equal to i - 1. Using the notion of structural symmetry as explained above, the total number of unique permutations (input orders) is given by

$$\left(\frac{i!}{(2!)^{i/2}(2!)^{i/4}(2!)^{i/8}\cdots(2!)}\right)$$

The expression in the denominator can be further simplified by the substitution $i = 2^k$

$$\left(\frac{i!}{(2)^{\frac{2^k}{2}}(2)^{\frac{2^k}{4}}(2)^{\frac{2^k}{8}}\cdots(2)\cdot(1)}\right)=\left(\frac{i!}{\prod_{j=1}^{k}(2)^{\frac{2^k}{2^j}}}\right)=\left(\frac{i!}{(2)^{\sum_{j=1}^{k-1}2^j}}\right)=\left(\frac{i!}{(2)^{2^k-1}}\right)=\left(\frac{i!}{(2)^{i-1}}\right)$$

For $i \geq 8$, $(i!) \geq \left(2\cdot 4^{i-1}\right)$ and hence the above expression is greater than 2^i. This implies that the number of unique input orders is exponential in the number of inputs.

A similar analysis can be performed for two-rail code checkers, since the standard design for a two-rail code checker works with pairs of outputs of the driver circuit at a time.

Berger codes are a class of systematic unidirectional error detecting codes [Pradhan 86]. Berger code checkers are of two types in practice, based on parallel ones counters and sorting networks. We focus on ordering the inputs to the parallel ones counter to reduce power consumption, since they are smaller, faster, and lower on power consumption as the number of inputs increases [Piestrak 01]. Full adders are usually not structurally symmetric with respect to all their inputs. Hence, the number of equivalent permutations is bounded by the following expression for logic level 1.

$$\left(\frac{i!}{(3!)^{i/3}}\right)$$

It can be shown by an analysis similar to that above for parity checkers that the number of input orders is exponential in the number of inputs to a Berger code checker for all $n \geq 9$, since $(i!) \geq \left((3!)^{i/3}\cdot 2^i\right)$ for $i \geq 9$.

5.2 Proposed methodology

We use parity codes as an example to illustrate the key idea of our contribution. We present a small example on how spatial correlations in the inputs can affect power consumption in the checker. Consider the Boolean functions $F_1, F_2, \ldots$ and F_7 in Fig. 5.3. These (in some permutation) drive the inputs $x_1, x_2, \ldots$ and x_7 of the parity tree shown in Fig. 5.4.

F_1

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

F_2

0	0	1	1
0	0	1	1
0	0	1	1
0	1	0	1

F_3

1	1	1	1
1	1	1	1
1	0	1	1
0	0	1	1

F_4

1	1	1	1
1	1	1	1
1	0	0	0
0	0	1	1

F_5

0	1	0	1
0	0	1	1
1	0	0	0
0	0	0	0

F_6

1	0	1	0
1	0	1	0
1	1	1	0
0	1	1	0

F_7

1	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0

Figure 5.3 Boolean functions F_1, F_2, …and F_7

A naïve approach to solving the problem of determining the optimum ordering of the inputs to the checker would be to exhaustively enumerate all possible permutations and to compute the exact power consumption for each of the possible solutions. The best permutation is then chosen. For the example in Fig. 5.3, the optimum permutation obtained by exhaustive enumeration has a power consumption of 106 units. The computational costs of this method are exorbitant even for small values of *n*, since the number of possible permutations is exponential in *n*.

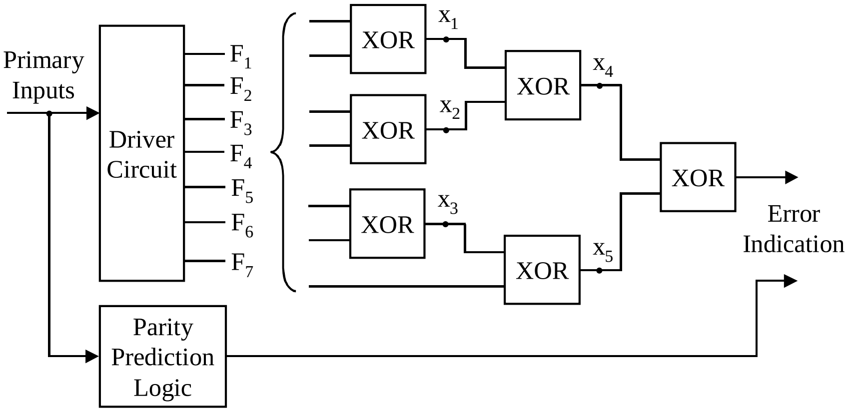


Figure 5.4 Circuit with parity encoded outputs

We propose a simulated annealing approach to solve the optimization problem of determining the best permutation. Simulated annealing that uses the Metropolis Monte Carlo algorithm and a logarithmic cooling schedule is used [Kirkpatrick 83]. We present two methods, both of which use the same simulated annealing framework, but differ in the complexity of the chosen cost function.

5.2.1 Exact simulation method

The first method, which is computationally expensive, uses the exact routine to calculate the power consumption in the checker as the cost function. This involves the use of a power estimator that simulates the checker using an output trace of the circuit driving the checker and computes the transitions at each of the internal nodes of the checker to estimate power. We term this method the “exact simulation method”. For the example in Fig. 5.3, the optimum permutation returned by the exact simulation method has a power consumption of 106 units. However, making a call to the power estimator for each permutation encountered during the simulated annealing routine is very expensive. The total number of calls is equal to the number of permutations encountered per iteration multiplied by the number of times the temperature is reduced during the simulated annealing routine. This can result in very high computational costs as the number of inputs to the checker increases.

While one approach that trades accuracy for speed would be to run the power estimation algorithm with fewer trace vectors, the computational costs are still very high as the number of inputs to the checker increases. In the next section, we describe an alternate approach that uses a simple cost function within the same simulated annealing framework to yield near optimal results.

5.3 Spatial correlation estimation method

The second algorithm proposed in this paper uses the same simulated annealing framework as the exact simulation method, but with a reduced cost function that results in a substantial decrease in the runtime complexity. We term this method the “spatial correlation estimation method”. For the example in Fig. 5.3, the optimum permutation returned by the spatial correlation estimation method has a power consumption of 106 units. This method is so called because the reduced cost function is built using the values of spatial correlation that are computed for the outputs of the circuit driving the checker. The use of the reduced cost function does not involve any circuit simulation for each permutation encountered during the simulated annealing routine. The reduced cost function is built *once* by a structural analysis of the checker, and all input orders are evaluated by simple substitution of spatial correlation values into this function. This avoids the use of the power estimator that is the computational bottleneck of the exact simulation method. The pseudo-code for the spatial correlation estimation method is presented in Fig. 5.5.

```
Algorithm  Compute_Ordering  ( checker_netlist, patterns )  {
        /* patterns  –  derived from the output trace of the circuit that drives the checker
           checker_netlist  –  technology mapped netlist of the checker */

        correlation  =  compute_correlation ( patterns ) ;
        /* compute spatial correlation between all pairs of outputs of the circuit that drives the checker */

        reduced_cost_function  =  build_cost_function ( checker_netlist ) ;
        /* build reduced cost function using structural analysis of the checker netlist */

        permutation  =  random (  ) ;   /* random initial permutation of the inputs */

        permutation  =  simulated_annealing ( reduced_cost_function, permutation ) ;
        /* simulated annealing routine to solve the optimization problem */

        return permutation ;
}
```

Figure 5.5 Pseudo-code for the spatial correlation estimation method

5.3.1 Spatial correlations and transition probability

The reduced cost function uses the notion of the transition probability at a node, used by probabilistic techniques for power estimation, as a measure of the average switching activity at that node. We first introduce some terminology and definitions, and provide an overview of probabilistic techniques for power estimation [Najm 94]. We use the terms *signal* and *node* interchangeably throughout this document.

Definition 1: (signal probability): The signal probability $P_S(x)$ of a node x in the circuit corresponds to the average fraction of clock cycles in which the node has a steady state logic value of ONE. An exact algorithm to estimate signal probabilities of all the internal signals of a circuit was proposed in [Parker 75].

Definition 2: (spatial correlation): Two (or more) signals are spatially correlated if the values that one assumes are dependent on the values of the other. Two signals are spatially independent if they are not spatially correlated.

Definition 3: (temporal correlation): A signal is temporally correlated if the value that it assumes on the next clock cycle is dependent on its present and (or) previous values.

In other words, spatial correlation refers to the correlation between pairs of bits in the same input vector, while temporal correlation refers to the correlation between pairs of input vectors, spaced one or more cycles apart. We use random pattern simulation to compute the spatial correlation between the output signals of the circuit that drives the checker. If application vectors are used instead, the values of correlation obtained will reflect both temporal and spatial correlation, since the distribution of vectors is not likely to be uniform. In other words, the distribution of vectors in the application vector set is not likely to be uniform.

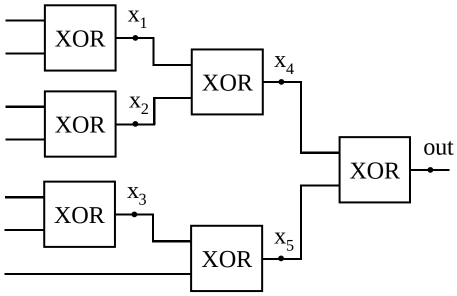
Definition 4: (transition probability): The transition probability $P_t(x)$ of a node x in the circuit corresponds to the average fraction of clock cycles in which the steady state value of the node is different from its initial value. Under the temporal independence assumption, the transition probability of a node x_i is directly obtained from its signal probability $P_S(x_i)$ as

$$P_t(x_i) = 2 \cdot P_s(x_i) \cdot P_s(\overline{x_i}) = 2 \cdot P_s(x_i) \cdot (1 - P_s(x_i))$$

Probabilistic techniques for power estimation propagate signal probabilities at the inputs into the circuit assuming spatial independence and use the temporal independence assumption to derive the transition probabilities. As discussed in Sec. 5.3.2, this can lead to large inaccuracies in the spatial correlation estimation method.

Reducing the transition probability at a node has the direct benefit of reducing the power consumption at that node. In addition, it is likely that this reduction in switching activity results in a reduction in the switching activity at the nodes that depend on this node. This concept can be extended to the checker as a whole, since an optimal permutation will certainly reduce switching activity at nodes close to the primary inputs of the checker. This has a cascading effect, in that fewer transitions occur at the outputs of the gates that use these nodes as inputs, and so on to the primary outputs of the checker.

The reduced cost function is built by a structural analysis of the checker which is decomposed using 2-input gates. To build the reduced cost function, we compute the exact transition probability (taking spatial correlation into consideration) for all those signals that depend on one or two primary inputs to the checker. This is usually possible for nodes that reach a topological depth of two or three in the checker. The transition probability at the node is weighted by the load capacitance driven by that node. The reduced cost function for the parity checker in Fig. 5.4 (assuming unit load capacitance) is shown in Fig. 5.6. Note that the reduced cost function does not include the transition probabilities at nodes x_4 and x_5, since they depend on more than two primary inputs.

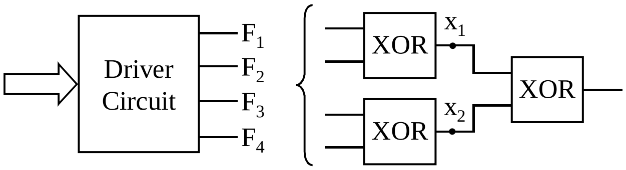


Reduced Cost Function = $P_t(x_1) + P_t(x_2) + P_t(x_3)$

Figure 5.6 Reduced cost function for the example from Fig. 5.4.

Correlations between all pairs of outputs of the circuit driving the checker are estimated by doing random vector simulation of the circuit driving the checker. Random vector simulation can be replaced by actual application vector simulation when they are available, since this best captures the correlations (both spatial and temporal) between signals. For every pair of outputs (F_i, F_j) of the driver circuit, there are four combinations of values that can occur depending on the inputs to the circuit – *00, 01, 10,* and *11* – and hence four values of spatial correlation (that sum to 1) to be computed.

Once the correlation values are known, the exact transition probability at a node (that depends on two primary inputs) can be directly computed. This is illustrated in greater detail for a parity tree with four inputs in Fig. 5.7. The four inputs are driven by the functions F_1, F_2 , F_3 , and F_4 from Fig. 5.3.



Reduced Cost Function $= P_t (x_1) + P_t (x_2)$

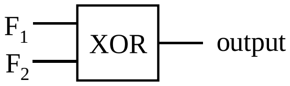
Input Order	$P_t (x_1)$	$P_t (x_2)$	Reduced Cost Function	Power
$F_1 F_2 F_3 F_4$	0.22	0.22	0.44	21
$F_1 F_3 F_2 F_4$	0.49	0.49	0.98	25
$F_1 F_4 F_2 F_3$	0.49	0.49	0.98	25

Figure 5.7 Spatial correlation estimation method for F_1, F_2 , F_3 , and F_4

The main benefit of using the reduced cost function is that many more permutations can be explored with minimal tradeoff in the accuracy and quality of the final permutation that is obtained. In Sec. 5.4 we present experimental results that indicate that the power consumption of the final permutation obtained is 16% lower than the average power consumption over 100 random input orders (when averaged over three types of checkers and all the test cases).

5.3.2 Spatial correlations and accuracy

It is important to use spatial correlation values to compute the transition probability at nodes while building the reduced cost function. It is possible to use, under the spatial independence assumption, the signal probability at a node to estimate the transition probability at that node [Najm 94]. This is not as efficient, however, since correlations between pairs of inputs are not captured with sufficient accuracy. This is illustrated with an example in Fig. 5.8. Consider an XOR gate, driven by the functions F_1 and F_2 from Fig. 5.3. In Fig. 5.8, we compare the transition probability at the output of the XOR gate computed when spatial independence is assumed with the exact transition probability. Note that there is a significant difference (0.50 versus 0.22).



Transition at output if inputs switch { 00, 11 } ⟺ { 01, 10 }

$P(\text{ output} = 1\text{ })_{\text{spatial independence}} = 0.50$ $P(\text{ output} = 1\text{ })_{\text{exact}} = 0.125$

$P_{\text{transition}}(\text{ output })_{\text{spatial independence}} = 0.50$ $P_{\text{transition}}(\text{ output })_{\text{exact}} = 0.22$

Figure 5.8 Inaccuracies when spatial independence is assumed

If the transition probabilities at the nodes are not accurate, the reduced cost function is not accurate. Such discrepancies can seriously affect the direction taken by the spatial correlation estimation method. For the example in Fig. 5.3, a solution that is not close to the optimum is obtained when spatial correlations are neglected, i.e., when spatial independence is assumed. The final permutation returned by the spatial correlation estimation method using the reduced cost function built under the spatial independence assumption has a power consumption of 119 units. This is not only considerably off the global optimum of 106 units, but also very close to the maximum power consumption that was observed (126 units).

5.4 Implementation and experimental results

The synthesis tool used for all technology mapping and power estimation in this paper is SIS [Sentovich 92]. Some combinational benchmark circuits were chosen from the LGSynth91 suite [Yang 91]. 100,000 random vectors were used to obtain an output trace from each of the circuits. This trace was used to compute the spatial correlation between the outputs, as well as the power consumption for each of the permutations that were evaluated (for the exact simulation method). The power consumption of the input order returned by the spatial correlation estimation method, as well as the average power consumption over 100 random input orders was also evaluated using this trace. Separate runs using the *minimal.genlib* and *mcnc.genlib* technology libraries were performed, since the optimal permutation obtained, as well as the reduction in power consumption achieved, varies according to the library used for technology mapping.

Table 5.1 provides details about the circuits that were chosen – name, number of primary inputs, and number of primary outputs.

Table 5.2 presents the results for parity checkers for some combinational benchmark circuits chosen from the LGSynth91 suite, mapped using the *minimal.genlib* and the *mcnc.genlib* technology library. Under the first major heading, the name of the circuit is provided. Under the second major heading, we report results when the checker is mapped using the *minimal.genlib* technology library. The average power consumption over 100 random input orderings that were used to drive the checker, the power consumption for the optimal permutation obtained using the exact simulation method, as well as the runtime, and the power consumption for the optimal permutation obtained using the spatial correlation estimation method, as well as the runtime are provided in that order. It is evident from the results that reordering the inputs results in significant power consumption reduction (12% on the average) when parity checkers are used. In addition, the spatial correlation estimation method provides a near optimal solution at a fraction of the computational cost of the exact simulation method. Note that the runtimes reported do not reflect the time taken to obtain an output trace from the driver circuit, since the same trace is used to report the power consumption in all the cases.

Under the third major heading, we report results when the checker is mapped using the *mcnc.genlib* technology library. The results are similar to those obtained when the *minimal.genlib* technology library is used – there is a reduction in power consumption, and the spatial correlation estimation method returns a solution that is close to that returned by the exact simulation method at a fraction of the computational cost.

Table 5.3 presents the results for two-rail-code checkers for some combinational benchmark circuits chosen from the LGSynth91 suite, mapped using the *minimal.genlib* and *mcnc.genlib* technology library respectively. The results are similar to those obtained for parity checkers – there is a reduction in power consumption, and the spatial correlation estimation method returns a solution that is close to that returned by the exact simulation method at a fraction of the computational cost.

Table 5.4 presents the results for Berger code checkers for some combinational benchmark circuits chosen from the LGSynth91 suite, mapped using the *minimal.genlib* and *mcnc.genlib* technology library respectively. The results are similar to those obtained for parity checkers – there is a reduction in power consumption, and the spatial correlation estimation method returns a solution that is close to that returned by the exact simulation method at a fraction of the computational cost.

The runtime of the simulated annealing routine for the exact simulation method was chosen to be between one to two orders of magnitude greater than that of the spatial correlation estimation method, depending on the complexity of the checker. If the execution time exceeded this predetermined limit, the exact simulation method was aborted and the best solution seen up to that point is reported *only* if it is better than that returned by the spatial correlation estimation method – this appears as (power consumption, *) in the third major column of all the tables. If the solution provided by the spatial correlation method is better, the solution is reported as (*, *). Cases where the solution of the exact simulation method is poorer than that obtained using the spatial correlation estimation method and where the exact simulation method was not aborted are given by (*, execution time).

5.4.1 Runtime versus accuracy tradeoffs

Our implementation decomposes the checker using 2-input gates when building the reduced cost function. If correlation values for signals considered three at a time were available, it would be possible to write the transition probabilities for some more gates when building the reduced cost function. This can improve the accuracy of the proposed procedure, and better results could be obtained by decomposing the checker to up to 3-input gates. Our experimental results indicate that it is sufficient to consider correlations between pairs of signals when building the reduced cost function.

It is also interesting to note that the computational complexity of the spatial correlation estimation method lies mainly in the estimation of the spatial correlation between pairs of outputs of the driver circuit. It is not affected by the choice of the library that is used for technology mapping or the complexity of the checker. In other words, for the same number of inputs, parity checkers are less complex than two-rail code checkers that are in turn less complex than Berger code checkers. The exact simulation method takes least time for parity checkers and the most time for Berger code checkers for the same circuit. However, the spatial correlation estimation method takes the same time for all the three checkers for any given circuit. Thus, the spatial correlation estimation method provides a very fast alternative to computing a good input order as the complexity of the checker (and hence the exact simulation method) increases.

Table 5.1 Benchmark combinational circuits [Yang 91]

Circuit	Num PIs	Num POs	Circuit	Num PIs	Num POs
x2	10	7	x1	51	35
cu	14	11	vda	17	39
sct	19	15	k2	45	45
b9	41	21	i5	133	66
seq	41	35	i6	49	67

Table 5.2 Power consumption reduction for parity checkers

Circuit	*minimal.genlib*					*mcnc.genlib*				
	Average Power	Exact Simulation Method		Spatial Correlation Estimation Method		Average Power	Exact Simulation Method		Spatial Correlation Estimation Method	
		Power	Time	Power	Time		Power	Time	Power	Time
x2	80	72	1066	73	16	44	35	188	36	16
cu	76	60	2368	63	31	43	31	374	34	31
sct	289	262	3286	274	53	131	111	468	118	53
b9	432	394	6119	425	97	207	140	769	152	97
seq	455	347	13400	347	246	212	113	1300	122	246
x1	830	725	13410	746	249	374	315	1298	349	249
vda	867	710	14672	728	303	381	275	1394	298	303
k2	572	*	*	444	396	281	188	1579	198	396
i5	2230	*	*	2118	872	786	762	4150	764	872
i6	2334	*	*	2109	886	857	*	4600	776	886

Table 5.3 Power consumption reduction for two-rail code checkers

Circuit	*minimal.genlib*					*mcnc.genlib*				
	Average Power	Exact Simulation Method		Spatial Correlation Estimation Method		Average Power	Exact Simulation Method		Spatial Correlation Estimation Method	
		Power	Time	Power	Time		Power	Time	Power	Time
x2	156	118	1096	131	16	209	150	636	175	16
cu	139	82	1802	106	31	182	111	981	143	31
sct	396	331	2351	354	53	517	439	1156	468	53
b9	568	401	3458	403	97	815	560	2293	608	97
seq	560	309	6133	317	246	772	395	2306	501	246
x1	972	765	5900	771	249	1344	1110	3370	1124	249
vda	993	734	7030	735	303	1335	998	3377	1010	303
k2	762	483	8349	493	396	1035	667	4323	703	396
i5	2173	2101	14712	2111	872	2878	*	5273	2800	872
i6	2201	1910	15109	1953	886	3063	2700	6232	2712	886

Table 5.4 Power consumption reduction for Berger code checkers

Circuit	*minimal.genlib*					*mcnc.genlib*				
	Average Power	Exact Simulation Method		Spatial Correlation Estimation Method		Average Power	Exact Simulation Method		Spatial Correlation Estimation Method	
		Power	Time	Power	Time		Power	Time	Power	Time
x2	268	230	3510	250	16	290	254	2900	269	16
cu	267	210	19140	210	31	283	212	24486	224	31
sct	1234	1101	22510	1134	53	1355	1099	29772	1282	53
b9	2405	1987	77400	2161	97	2690	2046	*	2429	97
seq	2501	*	*	1765	246	2829	*	*	1909	246
x1	7602	*	*	7363	249	7602	6800	*	7363	249
vda	6789	*	*	5729	303	6769	*	*	5729	303
k2	3234	*	*	2668	396	3667	*	*	3166	396
i5	28178	*	*	27813	872	28173	*	*	27826	872
i6	26110	*	*	24462	886	26787	*	*	24919	886

5.5 Summary

In this chapter, a novel approach for reducing the power consumption in checkers used for CED was presented. The basic idea is to analyze spatial correlations between the outputs of the circuit that drives the checker, and to order them such that switching activity (and hence power consumption) in the checker is minimized. The computational costs of determining the optimum order can be very expensive, since the number of possible permutations increases exponentially in the number of inputs to the checker. A technique to build a reduced cost function to solve the optimization problem was presented. The reduction in power consumption comes at no additional impact to area or performance and does not require any alteration to the design flow. The only cost is the computational time to determine the optimal input ordering to the checker.

Chapter 6. Eliminating Non-determinism During Test of High-Speed Source Synchronous Differential Buses

The increasing demand for throughput has accelerated the move away from hierarchical shared bus architectures like PCI and PCI-X towards high performance, packet switched architectures like RapidIO^{TM} [Bouvier 00], Infiniband^{TM} [Infiniband 01], and HyperTransport^{TM} [HyperTransport 01]. These protocols span a wide range of applications, both general purpose and otherwise.

Hierarchical shared bus topologies have failed to meet the data bandwidth requirements and increased system concurrency needs in today's demanding high-speed, high-performance processing environments. The fact that a device has to wait for the bus arbiter to complete an existing transaction before serving the next request poses a severe limitation on the available bandwidth on shared buses. Improvements like increasing the bus width and the bus frequency not only counter each other due to skew between signals but also increase the pin count on the device. Moreover, routing issues constrain the number of devices that can directly communicate with each other as the complexity of shared bus protocols increases. It is interesting to note that today's 133 MHz PCI-X standard allows only 2 devices per bus segment, and is for all practical purposes a point-to-point connection [Jaenicke 01].

These limitations have led to the emergence of high-speed, source synchronous, low power-low voltage differential signaling (LP-LVDS) parallel (or serial) buses such as the RapidIO^{TM} interconnect architecture (optimized for intra-system interconnections). Such high-performance interconnect technologies employ packet switching to transfer data from source to destination thereby providing a high degree of scalability and are projected to attain performance levels scaling to 10 Gbps and beyond. Other examples of emerging high-speed interconnect architectures include Infiniband^{TM} (optimized for system area networks) and HyperTransport^{TM} (optimized for system level interconnections).

The development of packet switched interconnect architectures is not without its share of challenges for the test community. The challenges are manifold, requiring innovative ideas not only from an automated test equipment (ATE) hardware perspective but also from a design-for-test (DFT) and test generation perspective. There has been some work in the area of ATE hardware development to meet the requirements of supply and capture of high-speed differential stimuli to the device-under-test (DUT). In [Oshima 01], a pin electronics integrated circuit capable of testing high-speed differential buses up to 3 Gbps is presented. In [Keezer 01], a multiplexing/de-multiplexing approach to realize higher performance from existing ATE is presented.

The presence of multiple clock domains and internal asynchronous boundaries in high-performance designs necessitates the application of a full complement of “at-speed” functional tests in test mode. Functional tests for high-speed I/O ports use a stream of stimulus packets sent to the receive port of the DUT. The DUT processes these packets and the tester monitors the response packet stream from the DUT. However, non-determinism can be introduced by external inputs as well as the presence of asynchronous boundaries internal to the DUT such that the output from the DUT is not cycle deterministic. One such asynchronous boundary inherent to the DUT is between the receive logic clock domain of the high-speed I/O port and the core logic clock domain. The receive logic is driven by the source synchronous clock (henceforth the Rx clock) from the tester, that is asynchronous with respect to the core clock of the DUT. It is important that the tester not classify a defect-free device as a failure just because the received response did not match the expected one due to non-determinism. Non-determinism can be introduced due to jitter and limited ATE edge placement accuracy (EPA) in the source synchronous clock of the stimulus stream to the DUT from the tester. The frequency of operation of high-speed differential buses has increased at a rate much faster than the rate of improvement in ATE EPA, eroding large margins of comfort that were previously available. As this trend continues, the ambiguity of just when the ATE generated Rx clock is captured by the DUT’s core clock domain is increasing and guaranteeing cycle-for-cycle deterministic behavior of the DUT is becoming a challenge.

In this chapter, we present a DFT technique aimed at removing the non-deterministic behavior in the response stream from the DUT at high performance rates [Mohanram 03a]. We also provide an analysis of the probability of non-determinism as a function of clock speed and EPA and show that as the frequency of operation of high-speed I/Os continues to rise, non-determinism will become a significant problem that can result in an unacceptable yield loss.

The rest of this chapter is organized as follows. In Sec. 6.1, we describe the test environment for a high-speed I/O port and discuss the problem of non-determinism in greater detail. In Sec. 6.2, we present the proposed solution to the problem. In Sec. 6.3, we present a model for the EPA of a tester, and analyze the problem of non-determinism using an analytical framework. Section 6.4 is a conclusion.

6.1 Problem of non-determinism

The DUT in the test environment is shown in Fig. 6.1. The DUT has a core clock, a Tx (transmit) clock, and an Rx (receive source synchronous) clock. The core clock is created by an on-DUT PLL from an ATE supplied source and is phase aligned with the incoming reference clock from the ATE. The core clock and the Tx clock may be integer multiples of each other (including 1-1) and may also include non-integer modes such as 3-2, 5-2, etc. It is usually the case that the core clock is a slow one while the Tx clock, used by the DUT to transmit source synchronous data, is a fast one.

The Tx clock going into the ATE from the DUT is used by the ATE to capture the Tx data from the DUT source synchronously. We assume throughout that the tester is able to receive the source synchronous clock (the Tx clock of the DUT) of the response stream, capture the Tx data, and that there is no problem of resolution here. Some recent ATE provide the ability to capture a source synchronous bus with its own clock at frequencies up to 2.5Gbps [Teradyne 00], [Agilent 01], [Schlumberger 02].

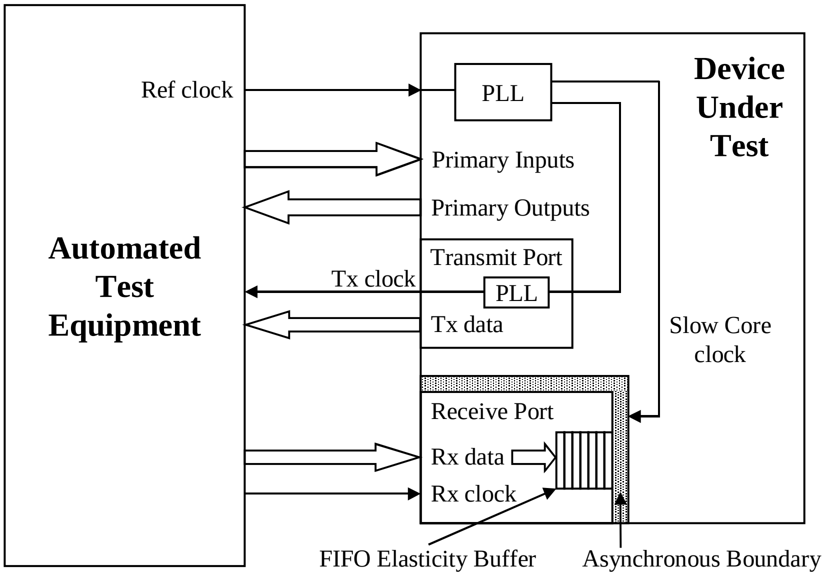


Figure 6.1 DUT in the test environment

Packets to the high-speed I/O port can be classified into two types – data and control. Control symbols are issued to initiate a request for and to acknowledge the receipt of packets in the system, as well as for system maintenance. Control symbols are also used for flow control, when retry and idle symbols are inserted to manage the flow of packets in the system. LP-LVDS buses are never tri-stated, but issue idle control symbols into the transmit stream when not processing any data.

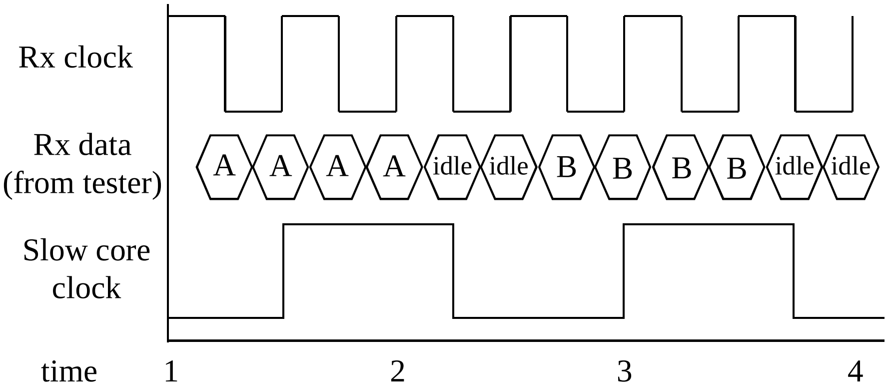


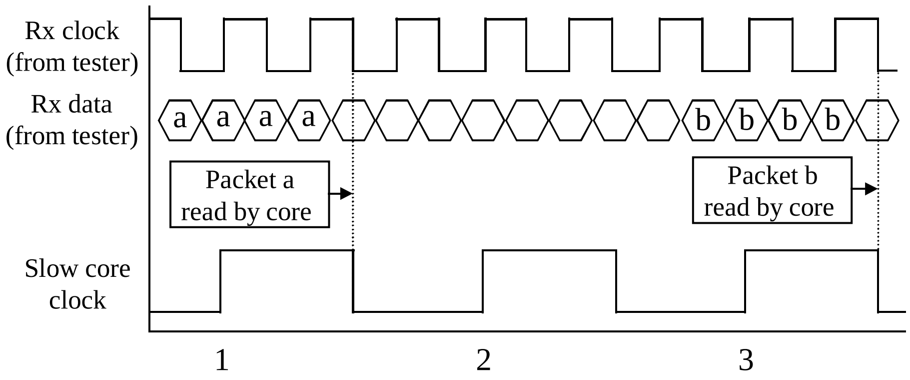
Figure 6.2 Sample input stream from ATE to DUT

Most packet switched I/O architectures use an elasticity buffer in the receive circuitry to temporarily store packets before transferring them into the core clock domain. The source synchronous data (Rx data) that is embedded in the stimulus stream from the ATE is de-serialized as it is clocked into the elasticity buffer with the source synchronous clock (Rx clock). Several cycles later, once the data is stable, it is pulled out of the elasticity buffer using the slow core clock. An asynchronous boundary is explicitly crossed when the packet is transferred into the slow core clock domain of the DUT. Figure 6.2 shows a sample input stream to the DUT from the tester. Note that the I/O port functions in double-data-rate (DDR) mode, since packets are clocked on the rising as well falling edges of the Rx clock.

The packet switched nature of the I/O protocol can complicate test response analysis in the presence of asynchronous boundaries internal to the DUT. In an ideal environment, as modeled in simulation, all events on the tester and the DUT occur at known absolute times. However, in the “real world”, there is the issue of ATE EPA which causes variability in when exactly events occur. The EPA is a measure of the accuracy with which the ATE can place an input drive edge at exactly the same point in time relative to a common reference. While these problems have always existed, they were never a cause for concern because the EPA of the ATE was so small compared to cycle times such that a “sweet spot” was always found for timing edge sets where the DUT would respond in a deterministic manner. However, in [ITRS 01], it is estimated that off-chip I/O speeds have improved at a rate of 30% per year, while ATE EPA has improved at a rate of 12% per year. Thus, as cycle times of high-speed buses are rapidly decreasing and EPAs are not decreasing at the same rate, we are approaching a point where non-determinism becomes an issue. A detailed analysis of the probability of non-determinism with respect to clock frequencies and tester EPA is provided in Sec. 6.4.

Non-determinism due to limitations in tester EPA occurs because of ambiguity in just when the source synchronous clock from the tester arrives at the DUT. When crossing the asynchronous boundary internal to the DUT, the cycle in which the packet is read from the elasticity buffer by the core logic could be misaligned by one internal core clock cycle. This variability in when the packet is received can cause the subsequent output stream to be different than expected (without the presence of a defect).

(a) Simulation predicted order of events



(b) Non-determinism due to limitations in ATE EPA

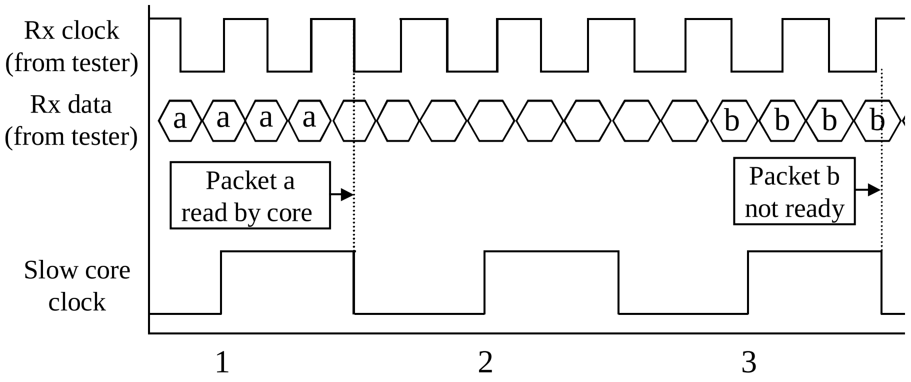


Figure 6.3 Example of non-determinism where packet *b* is read in a different core clock cycle

Non-determinism is illustrated in Fig. 6.3. Packet *a* is clocked into the core clock domain in cycle 1 as expected. Now consider packet *b* from the tester, which is expected to be clocked in from the elasticity buffer in cycle 3 of the DUT’s slow core clock. Limitations in the EPA of the tester clock cause it to arrive later such that it is clocked into the elasticity buffer in cycle 4 of the DUT’s core clock (instead of cycle 3 as originally expected). This causes the test stimuli that are synchronized to the core clock of the DUT when packet *b* is received to be offset by one core clock cycle – something that simulation did not predict, and something that can cause a deviation from the expected response (as will be elaborated on shortly). The ATE can end up classifying the part as defective simply because the packet was received a cycle later than intended. Variability in packet arrival times is not a problem in normal operation because the functional logic can handle packets in any order and will provide a correct output response for the order it receives the packets in. The problem exists in test because it is necessary that the output response match what is expected by the tester so that the part can be correctly classified.

6.1.1 Types of non-determinism

Two forms of the non-determinism in the output stream are possible. The first is that the response from the DUT occurs in the same order as the expected response, but is simply delayed by an extra idle in the Tx stream. This can be handled by “match mode” on testers today where they automatically compare and synchronize the actual and expected data streams. The second, and more difficult, case to handle is when the DUT issues all the test response packets and controls correctly, but in an order other than that predicted by simulation. This can occur because the order in which things are processed by the core logic is altered due to the variability in when the packet is received with respect to the other test stimuli synchronized to the core clock.

6.1.2 Possible solutions

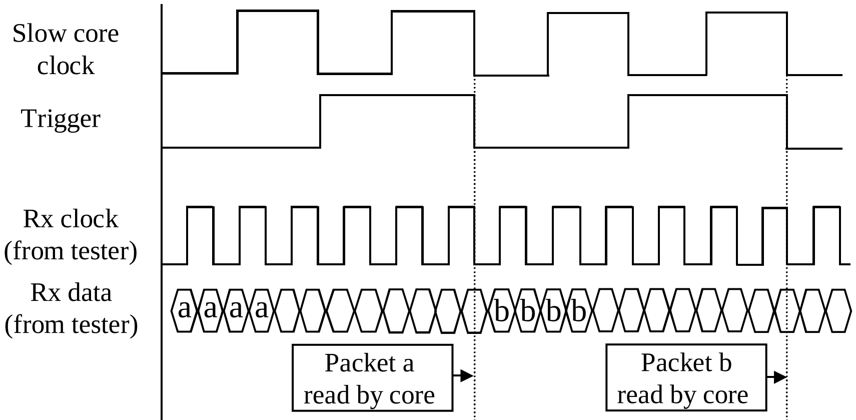
A naïve solution to this problem would be to test the DUT at a slower speed so that the input stream is less likely to be affected by any jitter in the fast source synchronous clock from the tester and the slow core clock of the DUT. Although a viable option, a direct testing of the DUT’s capabilities through “at-speed” functional tests is indispensable especially since such tests can uncover complex defects undetectable at lower speeds of operation.

A second option would be to include an explicit post-processing phase, where an algorithm is used to examine the contents of the capture RAM of the ATE to determine if the response stream can be matched to that from a non-defective part. The algorithm would try to match the actual response to one that could have been obtained from a non-defective part by examining the places where mismatch occurred, and try to fit that with a possible sequence of valid events in the DUT. Depending on the extent of non-determinism in the response, the test data volume (both input and output), and the complexity of the algorithm used, the costs associated with this approach could be prohibitive.

6.2 Proposed solution to eliminate non-determinism

The basic idea is to ensure that the DUT receives data from the tester on deterministic cycle boundaries of the slow core clock, i.e., to ensure that the DUT clocks in packets from the elasticity buffer into the slow core clock domain in a deterministic fashion. Since the DUT processes test packets in a deterministic way once they enter the slow core domain, the output response sent to the tester will also be deterministic.

(a) Simulation predicted order of events



(b) Actual order of events

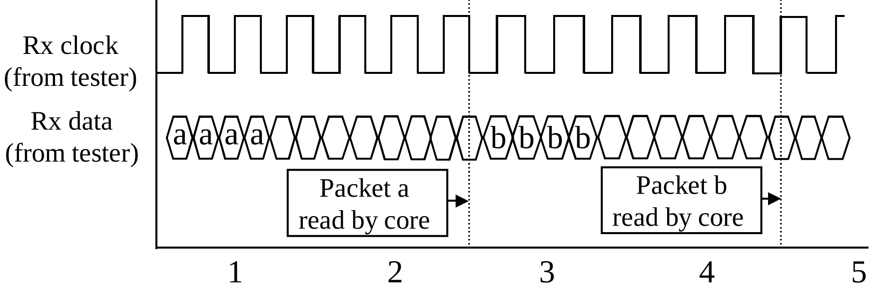


Figure 6.4 Use of the trigger signal to synchronize the reading of packets into the core clock domain to eliminate non-determinism

We propose the use of a trigger signal between the DUT and the tester that is another primary input synchronized to the slow core clock. Whenever a test packet is to be clocked in across the asynchronous boundary, the trigger signal is asserted a fixed number of cycles before the desired event. When the trigger is asserted, a counter is started on the DUT that prompts the reading of the elasticity buffer a fixed number of cycles later. The timing of the trigger signal is computed so that the packet will always be ready in the elasticity buffer even with the worst-case arrival time taking the EPA of the source synchronous clock from the tester to the DUT into consideration. If the packet arrives earlier, it is just held in the elasticity buffer until the predetermined core clock cycle. The trigger signal ensures that the packet crosses the asynchronous boundary in a deterministic fashion.

Figure 6.4 shows how the proposed solution will work to resolve the problem presented in Fig. 6.3. In this case, the trigger is asserted such that packets are clocked in cycles 2 and 4 of the slow core clock of the DUT. If the packet arrives at the expected time, it is held in the elasticity buffer for one core clock cycle before it is read (as shown in the upper part of the diagram). However, if jitter and limited tester EPA cause the packet arrival time to be delayed, it is still read in during the same core clock cycle as the previous case (as shown in Fig. 6.4(b)). Hence the behavior is deterministic irregardless of the packet arrival time.

6.3 Overhead of the proposed solution

The DFT hardware for implementing the trigger is shown in Fig. 6.5. The trigger signal initiates the counter, and the counter gates the packet-ready signal when in test mode. The delay in clocking packets across the asynchronous boundary leads to the introduction of extra idle symbols in the response stream from the DUT. However, this is something than can be deterministically predicted. Note that there is a minor latency penalty to be paid for using the trigger scheme (one extra core clock cycle for each packet read). An additional drawback is that the DUT operation is not completely a normal functional mode, but the difference in the logic used in normal and test mode is minimal.

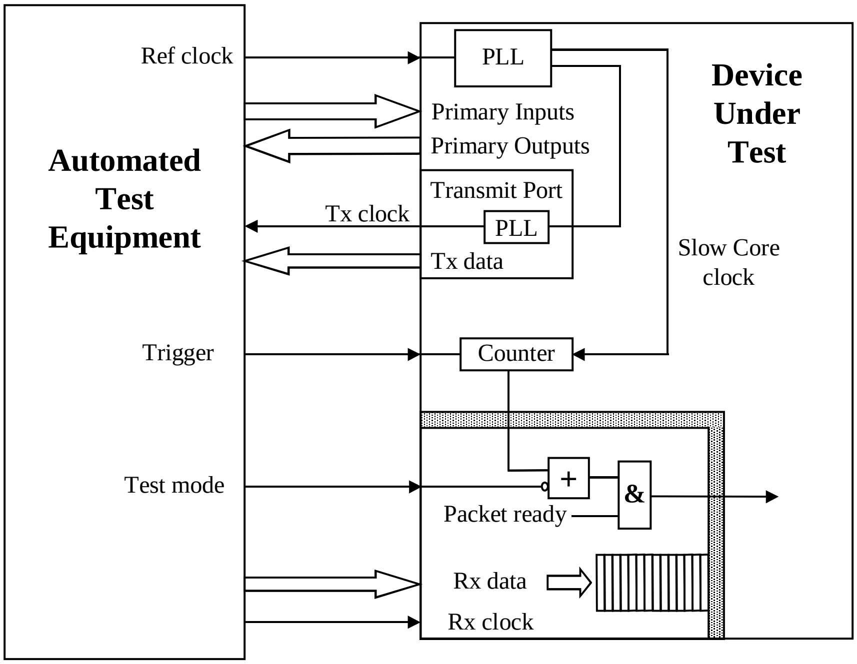


Figure 6.5 Proposed solution

This scheme also comes at the expense of increased hardware complexity on the DUT. The receive circuitry of the high-speed I/O port of the DUT has to be modified to realize the proposed solution in test-mode. A counter that initiates the transfer of test packets a fixed number of cycles after the trigger is asserted needs to be implemented. Note that an extra pin is not required on the DUT to support the trigger signal since one of the primary inputs of the DUT can be multiplexed internally to function as the trigger signal in test mode.

6.4 Analysis of probability of non-determinism

In this section, we analyze the probability of non-determinism in the output response of the DUT as a function of the Rx clock frequency and the EPA of the ATE. We address the important issue of when the problem of non-determinism becomes significant enough to warrant the implementation of the proposed solution.

6.4.1 EPA model

Since the EPA of the tester is a measure of the accuracy with which the tester can position any driven edge with respect to an absolute reference in time, it is possible to model the distribution of the placement of the driven edge by a Gaussian distribution about the reference point [Dalal 99]. The EPA of the tester is equal to three times the variance (σ) of this Gaussian distribution. Note that this translates to a very high probability (greater than 0.99) that the driven edge is within ±3σ of the reference point. This is illustrated in Fig. 6.6.

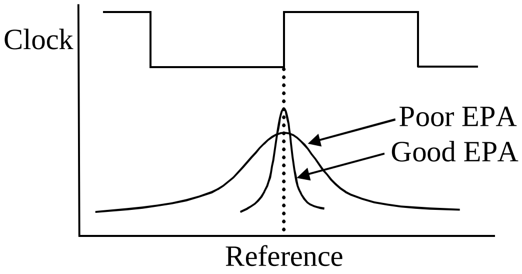


Figure 6.6 Gaussian distribution for driven edge and EPA

Thus, it is apparent that a high variance for this distribution corresponds to a poor tester EPA and vice versa. The Gaussian distribution for both the Rx clock (supplied by the ATE) and the core clock (of the DUT) can be used to tabulate the probability of occurrence of non-determinism as a function of both tester EPA and the speed of operation of the high-speed I/O port. Note that the core clock is derived from the reference clock which comes directly from the tester and hence is affected by the EPA of the tester.

6.4.2 Probability of non-determinism

Consider the final packet *f* of an input stream that is to be clocked in with the falling edge A_1 in Fig. 6.7. The corresponding core clock cycle in which this packet will be read from the elasticity buffer will depend on the relative position of the falling edges A_1 and A_2. If, say, A_1 occurs before A_2, the packet will be read into the core clock domain on a core clock cycle *i*. However, if A_1 occurs after A_2, the packet will be read into the core clock domain on core clock cycle *i+1*. Thus there is a one cycle variability in the core cycle on which the packet enters the core clock domain which can lead to non-determinism. As the absolute time difference between the occurrence of the falling edges

A_1 and A_2 shrinks (and becomes comparable to the EPA of the tester), the probability that the order of occurrence of the falling edges A_1 and A_2 differs from run to run increases.

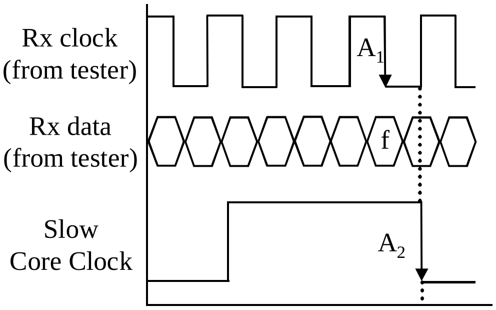


Figure 6.7 Analysis of non-determinism

Let the absolute time displacement between the occurrence of the two events A_1 and A_2 (in the simulation environment) be Δ. σ_{Rx} and σ_{core} are the variance of the Rx and the core clock probability distributions (Gaussian), respectively. Let the order of events in the simulation environment be A_1 precedes A_2 ($A_1 \rightarrow A_2$). Then, the probability distribution (Gaussian) of the placement of the edge A_1 (assumed to occur at an instant *y*), relative to the placement of the edge A_2 (which is used as the reference, with mean $\mu = 0$), is given by:

$$P\,(A_1\ occurs\ at\ y) = \exp\left(\frac{-(y-(-\Delta))^2}{2\cdot\sigma_{Rx}^2}\right)$$

If the event A_2 occurs at an instant *x*, the probability that A_1 occurs after *x* (and hence, in an order other than the simulation predicted one) is given by the integral:

$$P\,(A_1\ occurs\ after\ x) = \int_x^{\infty} \exp\left(\frac{-(y+\Delta)^2}{2\cdot\sigma_{Rx}^2}\right)dy$$

The above integral is then evaluated over the interval [-∞, +∞] to compute the total probability of interchange of the events A_1 and A_2. Thus:

$$P\,(A_2 \rightarrow A_1) = \int_{-\infty}^{\infty}\int_x^{\infty} \exp\left(\frac{-(y+\Delta)^2}{2\cdot\sigma_{Rx}^2}\right)dy\,\exp\left(\frac{-x^2}{2\cdot\sigma_{core}^2}\right)dx$$

When Δ is 0, i.e., when the time difference between the events A_1 and A_2 in the simulation environment is 0, $P(A_2 \rightarrow A_1)$ evaluates to 0.5. We show the value of

$P(A_2 \rightarrow A_1)$ as a function of the time difference Δ in the semi-log plot in Fig. 6.8.

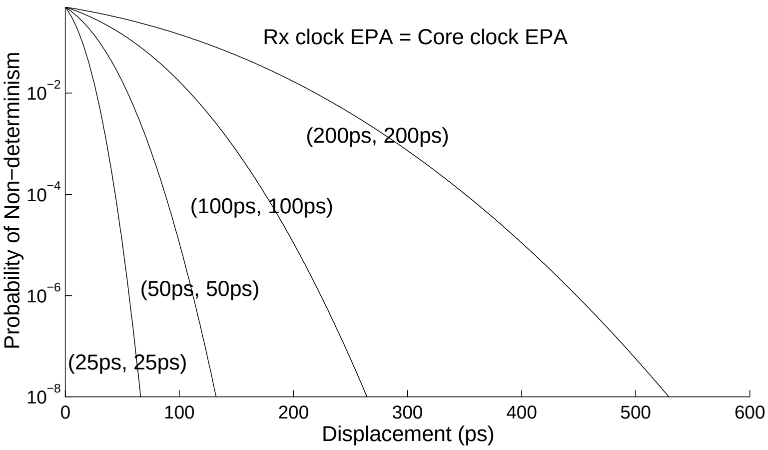


Figure 6.8 Semi-log plot of $P(A_2 \rightarrow A_1)$ as a function of displacement Δ.
(Rx clock EPA, core clock EPA) varies from (25ps, 25ps) to (200ps, 200ps)

To plot the graph in Fig. 6.8, the EPA of both the Rx clock and the core clock was varied from 25ps to 200ps. On the X-axis, the displacement Δ was varied from 0ps to 600ps in increments of 1ps. For each pair of values of the EPA of the Rx clock and the core clock, the log (base 10) of the probability of interchange of events A_1 and A_2 as a function of increasing displacement Δ is graphed. Note that the probability of non-determinism (i.e., $P(A_2 \rightarrow A_1)$) diminishes rapidly as the displacement Δ between the events A_1 and A_2 increases. In addition, for a fixed value of Δ, there is a significant increase in the probability of non-determinism as the EPA becomes less precise.

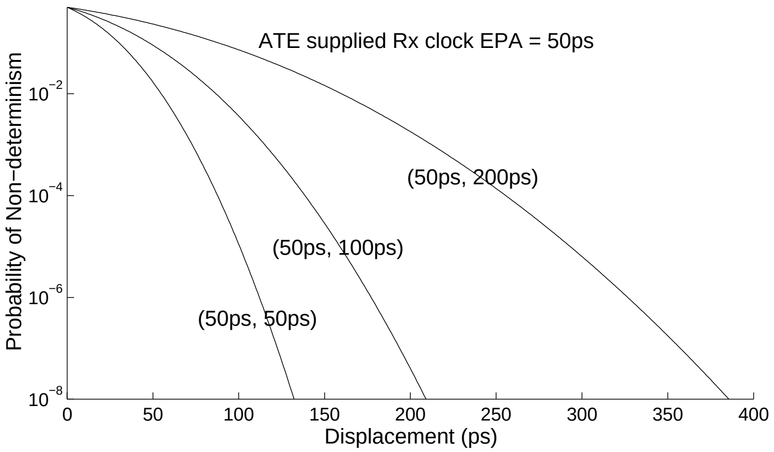


Figure 6.9 Semi-log plot of $P(A_2 \rightarrow A_1)$ as a function of Δ, when Rx clock EPA is 50ps and core clock EPA varies from 50ps to 200ps

Often the ATE has only a limited number of pins that have very precise EPA, while the other pins have a less precise EPA. In some cases, it may not be possible to use the very precise EPA pins to drive both the Rx clock and the core clock. In that case, the core clock is driven with a less precise EPA. We show a semi-log plot in Fig. 6.9 where the Rx clock EPA is 50ps while the core clock EPA varies from 50ps to 200ps. In Fig. 6.10, the Rx clock EPA is 25ps while the core clock EPA varies from 25ps to 200ps. Note that the probability of non-determinism is significantly increased as the core clock EPA becomes less precise.

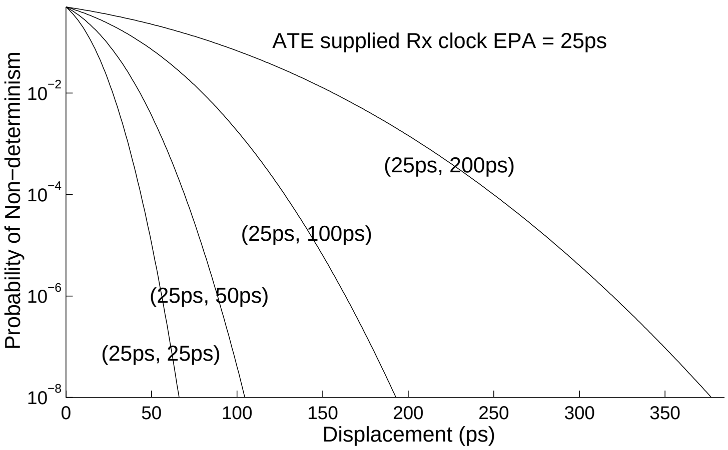


Figure 6.10 Semi-log plot of $P(A_2 \rightarrow A_1)$ as a function of Δ, when Rx clock EPA is 25ps and core clock EPA varies from 25ps to 200ps

6.4.3 Probability of non-determinism for the test session

Packets are latched on both edges of the Rx clock (since it is DDR). There are several edges on the Rx clock during each core clock period. The Rx clock edge that is nearest the falling edge of the core clock will have the smallest displacement Δ (such Rx clock edges will be referred to as *critical edges*). All the other edges during the core clock period will have a much larger displacement and hence a negligible probability of non-determinism. The best-case value for the displacement Δ for the critical edges is equal to half the period of the Rx clock. Let P_{nd} be the probability that a packet on a critical edge causes non-determinism. Then the probability of non-determinism for the entire test session is is equal to $[1 - (1 - P_{nd})^m]$ where m is the total number of packets that are received on critical edges.

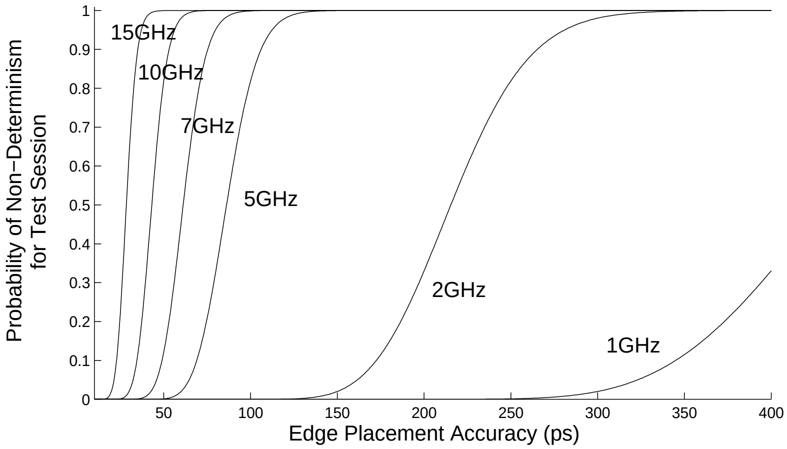


Figure 6.11 Probability of non-determinism for test session as a function of EPA for different I/O frequencies

In Figs. 6.11 and 6.12, we show the probability of non-determinism for the entire test session if m is 100 and the displacement Δ for the critical edges is equal to half the period of the Rx clock. Figure 6.11 shows how the probability of non-determinism for the test session varies with EPA for a given I/O frequency. As the I/O frequency continues to increase, the EPA must rapidly improve to keep the probability of non-determinism low.

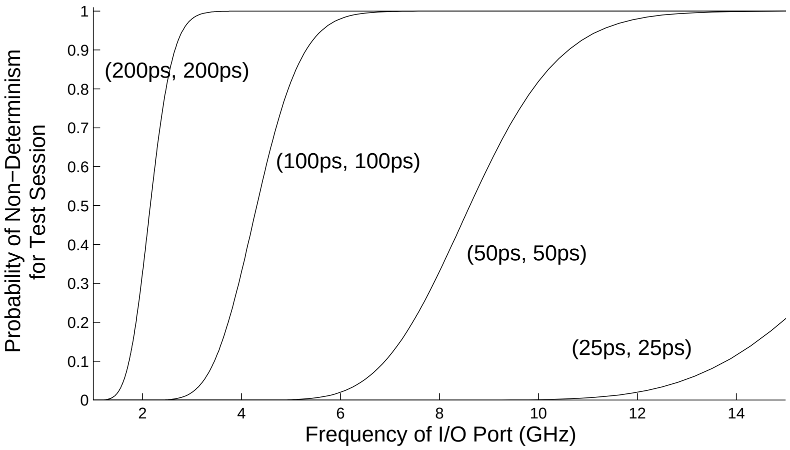


Figure 6.12 Probability of non-determinism for test session as a function of I/O frequency for different EPAs

Figure 6.12 shows how the probability of non-determinism for the test session varies with I/O frequency for a given EPA. If a tester with a particular EPA is to be used to test successive generations of chips, the probability of non-determinism will rapidly increase as the I/O frequency improves.

Table 6.1 Probability of non-determinism for test session if $m = 100$

Rx Clock Frequency	Δ ps	EPA (Rx Clock, Core Clock) (ps, ps)				
		(25, 25)	(50, 50)	(100,100)	(150,150)	(200,200)
1 GHz	500	0	0	0	0	$5.6 \cdot 10^{-6}$
2 GHz	250	0	0	$5.6 \cdot 10^{-6}$	0.0201	0.3306
5 GHz	100	0	0.0011	0.8190	0.9997	0.9999
7 GHz	71	0	0.1216	0.9989	0.9999	0.9999
10 GHz	50	0.0011	0.8190	0.9999	0.9999	0.9999
15 GHz	33	0.2256	0.9997	0.9999	0.9999	0.9999

Table 6.1 gives the probability of non-determinism for a test session ($m = 100$) for a few example cases. As can be seen from the table, if a tester with a 50ps EPA is used to test a chip with a 7 GHz I/O port, the probability of non-determinism is 0.1216. This could result in a significant yield loss. From this data, it is apparent that as the rapid increase in I/O frequencies continues to outstrip improvements in tester EPA, non-determinism in the output response of the DUT will be a significant problem. The DFT

technique presented in Sec. 6.3 can be used to address this problem.

6.5 Summary

As bus frequencies continue to rise and ATE edge placement accuracy becomes relatively less precise, techniques for limiting sources of non-deterministic behavior in the DUT will be needed. In this chapter, we have presented a DFT technique for the elimination of non-determinism that arises due to ATE EPA limitations in the source synchronous clock of the stimulus stream to a high-speed I/O port from the tester. The proposed solution allows the application of at-speed functional patterns to the DUT while incurring a very small hardware overhead and trivial increase in test application time.

Chapter 7. Conclusions and Future Directions

7.1 Concurrent error detection

As the failure rate due to temporary faults of logic circuits becomes unacceptably high even for mainstream applications, there will be a need for techniques to reduce the failure rate for such high-volume markets at minimum cost. This thesis described approaches for automated design of logic circuits that meet failure rate requirements while minimizing the impact to area, performance, and power. The primary emphasis in this thesis has been on reducing the soft error failure rate (which dominates). An immediate direction for future research is to incorporate other temporary fault models into the framework described in Sec. 3.3 and to evaluate the effectiveness of the CED techniques described in this thesis for such fault models. Other directions include (1) the development of efficient quantitative models (high-level as well as gate-level) to compute the temporary fault susceptibility in logic circuits and (2) the use of this fine-grain information to direct the insertion of fault tolerance features in the design to satisfy failure rate requirements in a cost-effective manner. Figure 7.1 shows an integrated logic synthesis environment, where reliability metrics are incorporated in iterative phases to realize reliable designs that meet area, delay, power, and testability metrics. It is expected that such techniques will be needed in the next 10 years to address the rapid increase in temporary fault failure rates for logic circuits.

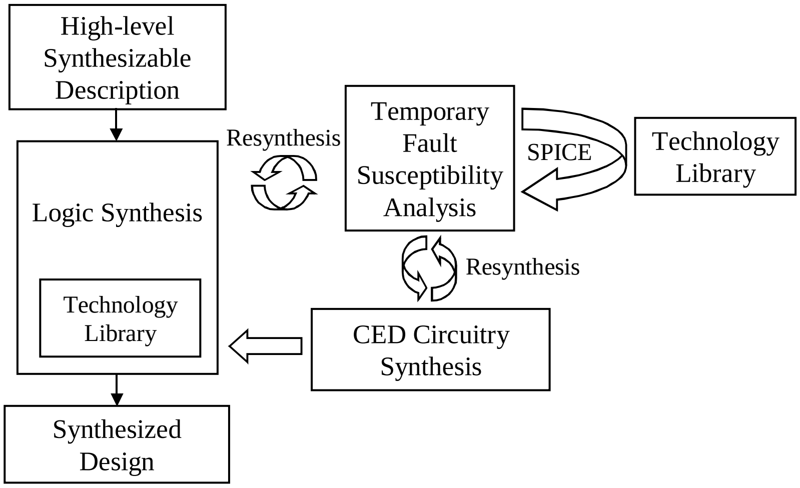


Figure 7.1 Integrated synthesis environment

Future directions for research to reduce the soft error failure rate in logic circuits will include techniques to augment a technology library with robust versions of cells (radiation hardening). During technology mapping, once temporary fault susceptibility analysis is performed and the nodes with high susceptibility are identified, the cells driving the most susceptible nodes can be replaced with a more robust version of the cell. Additionally, resizing techniques for gates can also be employed (subject to constraints) to increase their robustness.

7.2 Test of high-speed I/O ports

In this thesis, a low-cost trigger-based solution to eliminate the problem of non-determinism that may arise due to limitations in ATE EPA during at-speed functional test of high-speed source synchronous I/O ports was described. An analysis of when the problem of non-determinism becomes significant enough to warrant the implementation of the proposed solution was also provided. Future directions include the development of solutions from the DUT perspective in the form of DFT structures that eliminate the effects of non-determinism. These DFT structures should allow the application of functional test patterns to the DUT without any degradation in performance (in test mode or otherwise). There is also a need for innovative test methodologies that allow for the application of at-speed functional test patterns to the DUT. An example would include padding stimulus packets to avoid critical edges and race conditions (internal to the DUT). All the solutions need to be efficient from test application time, hardware overhead, and coverage perspectives to ensure the effectiveness of the test process.

Bibliography

[Abramovici 83] M. Abromovici, P. R. Menon, and D. T. Miller, "Critical Path Tracing – An Alternative to Fault Simulation", *Proc. Design Automation Conference*, pp. 214-220, 1983.

[Agilent 01] Technical Specifications, *Agilent 93000 SOC Series NP-models*, Agilent Technologies, 2001.

[Alexandrescu 02] D. Alexandrescu, L. Anghel, and M. Nicolaidis, "New Methods for Evaluating the Impact of Single Event Transients in VDSM ICs," *Proc. Defect and Fault Tolerance Symposium*, pp. 99-107, 2002.

[Bolchini 97] C. Bolchini, F. Salice, and D. Sciuto, "A Novel Methodology for Designing TSC Networks based on the Parity Bit Code," *Proc. European Design and Test Conference*, pp. 440-444, 1997.

[Boudjit 93] M. Boudjit, M. Nicolaidis, and K. Torki, "Automatic Generation Algorithms, Experiments and Comparisons of Self-Checking PLA Schemes Using Parity Codes," *Proc. European Conference on Design Automation*, pp. 144-150, Feb. 1993

[Bouvier 00] D. Bouvier, "RapidIO^{TM}: An Embedded System Component Network Architecture," RapidIO^{TM} Trade Association, 2001.

[Bradley 98] P. D. Bradley and E. Normand, "Single Even Upsets in Implantable Cardioverter Defibrillators," *IEEE Trans. Nuclear Science*, Vol. 45, No. 6, pp. 2929-2940, Dec. 1998.

[Cohen 99] N. Cohen, *et al.*, "Soft Error Considerations for Deep-Submicron CMOS Circuit Applications," *International Electron Devices Meeting*, 1999.

[Dalal 99] W. Dalal and S. Miao, "The Value of Tester Accuracy," *Proc. International Test Conference*, pp. 518-523, 1999.

[Das 99] D. Das and N. A. Touba, "Synthesis of Circuits with Low-Cost Concurrent Error Detection Based on Bose-Lin Codes," *Journal of Electronic Testing: Theory and Applications*, Vol. 15, Nos. 1/2, pp. 145-155, Aug. 1999.

[De 94] K. De, *et al.*, "RSYN: A System for Automated Synthesis of Reliable Multilevel Circuits," *IEEE Trans. VLSI Systems*, Vol. 2, Nos. 2, pp. 186-195, Jun. 1994.

[Favalli 02] M. Favalli and C. Metra, "Online Testing Approach for Very Deep-Submicron ICs," *IEEE Design and Test of Computers*, Vol. 19, No. 2, pp. 16-23, Mar. 2002.

[Franco 94] P. Franco and E. J. McCluskey, "On Line Delay Testing of Digital Circuits," *Proc. VLSI Test Symposium*, pp. 167-173, 1994.

[Freeman 96] L. B. Freeman, "Critical Charge Calculations for a Bipolar SRAM Array," *IBM Journal Research and Development*, Vol. 40, No. 1, pp. 119-129, Jan. 1996.

[Gorshe 96] S. Gorshe and B. Bose, "A Self-Checking ALU Design with Efficient Codes," *Proc. IEEE VLSI Test Symposium*, pp. 157-161, 1996.

[Gössel 93] M. Gössel and S. Graf, *Error Detection Circuits*, McGraw-Hill Book Company, London, 1993.

[Hazucha 00a] P. Hazucha and C. Svensson, "Cosmic Ray Neutron Multiple-Upset

Measurements in a 0.6-/spl mu/m CMOS Process," *IEEE Trans. Nuclear Science*, Vol. 47, No. 6, pp. 2595-2602, Jul. 2000.

[Hazucha 00b] P. Hazucha and C. Svensson, "Impact of CMOS Technology Scaling on the Atmospheric Neutron Soft Error Rate," *IEEE Trans. Nuclear Science,* Vol. 47, No. 6, pp. 2586-2594, Dec. 2000.

[HyperTransport 01] "HyperTransport™ Technology I/O Link: A High Bandwidth I/O Architecture," Advanced Micro Devices, Inc., 2001.

[Infiniband 01] Infiniband™ Architecture Specification, Vols. 1 and 2, Infiniband™ Trade Association, 2001.

[ITRS 01] The International Technology Roadmap for Semiconductors, Test and Test Equipment, 2001.

[Jaenicke 01] R. Jaenicke, "RapidIO raises PCI stature in the box," *EETIMES*, April 2001.

[Jha 93] N. K. Jha and S. Wang, "Design and Synthesis of Self-Checking VLSI Circuits," *IEEE Trans. Computer-Aided Design,* Vol. 12, pp. 878–887, Jun. 1993.

[Kavousianos 98] X. Kavousianos and D. Nikolos, "Novel Single and Double Output TSC Berger Code Checkers," *Proc. VLSI Test Symposium,* pp. 348-353, 1998.

[Keezer 01] Keezer, *et al.*, "Terabit-per-second Automated Digital Testing," *Proc. International Test Conference,* pp. 1143-1151, 2001.

[Kirkpatrick 79] S. Kirkpatrick, "Modeling Diffusion and Collection of Charge from Ionizing Radiation in Silicon Devices," *IEEE Trans. Electron Devices,* No. 11, pp. 1742-1753, Nov. 1979.

[Kirkpatrick 83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi Jr., "Optimization by Simulated Annealing," *Science*, pp. 671-680, May. 1983.

[Laprie 92] Laprie, J. C. (ed.), Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese, Springer-Verlag, 1992.

[Lidén 94] P. Lidén, *et al.*, "On Latching Probability of Particle Induced Transients in Combinational Networks," *Proc. Fault-Tolerant Computing Symposium,* pp. 340-349, 1994.

[Mak 82] G. P. Mak, J. A. Abraham, and E. S. Davidson, "The Design of PLAs with Concurrent Error Detection," *Proc. Intl. Symposium on Fault Tolerant Computing,* pp. 303-310, Jun. 1982.

[Massengill 98] L. W. Massengill, *et al.*, "Single-Event Upset Cross-Section Modeling in Combinational CMOS Logic Circuits," *Journal of Radiation Effects, Research, and Engineering*, Vol. 16, No. 1, Jan. 1998.

[Massengill 00] L. W. Massengill, *et al.*, "Analysis of Single-Event Effects in Combinational Logic-Simulation of the AM2901 Bitslice Processor", *IEEE Trans. Nuclear Science*, Vol. 47, No. 6, pp. 2609-2615, Dec. 2000.

[Maxwell 00] Maxwell, P., Hartanto, I., and Bentz, L., "Comparing functional and structural tests," *Proc. International Test Conference*, pp. 400-407, 2000.

[May 79] T. C. May and M. H. Woods, "Alpha-Particle-Induced Soft Errors in Dynamic

Memories," *IEEE Trans. Electron Devices*, Vol. 26, No. 1, pp. 2-9, Jan. 1979.

[Metra 96] C. Metra, M. Favalli, and B. Ricco, "Tree Checkers for Applications with Low Power-Delay Requirements," *Proc. Intl. Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 213-220, 1996.

[Metra 98] C. Metra, M. Favalli, and B. Ricco, "Online Detection of Logic Errors Due to Crosstalk, Delay and Transient Faults," *Proc. International Test Conference,* pp. 524-533, 1998.

[Mohanram 02a] K. Mohanram, C. V. Krishna, and N. A. Touba, "A Methodology for Automated Insertion of Concurrent Error Detection Hardware in Synthesizable Verilog RTL," *Proc. International Symposium on Circuits and Systems (ISCAS)*, Vol. 1, pp. 577-580, 2002.

[Mohanram 02b] K. Mohanram and N. A. Touba, "Input Ordering in Concurrent Checkers to Reduce Power Consumption," *Proc. Intl. Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 87-95, 2002.

[Mohanram 03a] K. Mohanram and N. A. Touba, "Eliminating Non-Determinism During Test of High-Speed Source Synchronous Differential Buses," *Proc. VLSI Test Symposium (VTS)*, pp. 121-127, 2003.

[Mohanram 03b] K. Mohanram and N. A. Touba, "Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits," (to appear in) *International Test Conference (ITC)*, 2003.

[Mohanram 03c] K. Mohanram and N. A. Touba, '"Partial Error Masking to Reduce Soft Error Failure Rate in Logic Circuits,'" (to appear in) *International Symposium on Defect and Fault Tolerance in VLSI Systems (DFTS)*, 2003.

[Murley 96] P. C. Murley and G. R. Srinivasan, "Soft-Error Monte Carlo Modeling Program, SEMM," *IBM Journal of Research and Development,* Vol. 40, No. 1, pp. 109-118, 1996.

[Mudge 01] T. Mudge, "Power: A First-Class Architectural Design Constraint," *IEEE Computer*, Vol. 34, Iss. 4, pp. 52-58, Apr. 2001.

[Musoll 96] E. Musoll and J. Cortadella, "Optimizing CMOS Circuits for Low Power Using Transistor Reordering," *Proc. European Design and Test Conference,* pp. 219-223, Mar. 1996.

[Najm 94] F. N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Trans. Very Large Scale Integration,* Vol. 2, No. 4, pp. 446-455, Dec. 1994.

[Normand 96] E. Normand, "Single-Event Effects in Avionics," *IEEE Trans. Nuclear Science,* Vol. 43, No. 2, pp. 461-474, Apr. 1996.

[Nicolaidis 93] M. Nicolaidis, "Efficient Implementations of Self-Checking Adders and ALUs," *Proc. Intl. Symposium on Fault Tolerant Computing,* pp. 586-595, Jun. 1993.

[Nicolaidis 98] M. Nicolaidis and Y. Zorian, "Online Testing for VLSI – A Compendium of Approaches," *Journal of Electronic Testing: Theory and Applications (JETTA),* Vol. 12, Nos. 1/2, pp. 7-20, Feb.-Apr. 1998.

[Nicolaidis 99] M. Nicolaidis, "Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies," *Proc. of VLSI Test Symposium*, pp. 86-94, 1999.

[Oshima 01] Oshima, A., Poniatowski, J., and Nomura, T., "Pin Electronics IC for High Speed Differential Devices," *Proc. International Test Conference*, pp 1128-1133, 2001.

[Pierce 65] W. H. Pierce, *Failure-Tolerant Computer Design*, Academic Press, New York, 1965.

[Piestrak 01] S. J. Piestrak, D. Bakalis, and X. Kavousianos, "On the Design of Self-Testing Checkers for Modified Berger Codes," *Proc. IEEE Intl. Online Testing Workshop,* pp. 153-157, 2001.

[Pradhan 86] D. K. Pradhan, *Fault Tolerant Computing – Theory and Techniques,* Vol. 1, Prentice-Hall, NJ, 1986.

[Prasad 96] S. C. Prasad and K. Roy, "Transistor Reordering for Power Minimization Under Delay Constraint," *ACM Trans. Design Automation of Electrical Systems,* Vol. 1, No. 2, pp. 280-300, Apr. 1996.

[Saposhnikov 98] V. V. Saposhnikov, *et al.*, "A New Design Methodology for Self-Checking Unidirectional Combinational Circuits," *Journal on Electronic Testing: Theory and Applications*, Vol. 12, Nos. 1/2, pp. 41-53, Feb. 1998.

[Sai-Halasz 82] G. A. Sai-Halasz, M. R. Wordeman, and R. H. Dennard, "Alpha-Particle-Induced Soft Error Rate in VLSI Circuits," *IEEE Journal of Solid-State Circuits,* Vol. 17, No. 2, pp. 355-361, Apr. 1982.

[Saposhnikov 98] V. V. Saposhnikov, *et al.*, "A New Design Methodology for Self-Checking Unidirectional Combinational Circuits," *Journal on Electronic Testing: Theory and Applications,* Vol. 12, Nos. 1/2, pp. 41-53, Feb. 1998.

[Schlumberger 02] Technical Specifications, *ITS9000ZX Test System,* Schlumberger Semiconductor Solutions, 2002.

[Schneider 96] P. H. Schneider and S. Krishnamoorthy, "Effects of Correlations on Accuracy of Power Analysis – An Experimental Study," *Proc. Intl. Symposium on Low Power Electronics and Design (ISLPED),* pp. 113-116, 1996.

[Sentovich 92] E. M. Sentovich, *et al.*, "SIS: A System for Sequential Circuit Synthesis," Technical Report Memorandum No. UCB/ERL M92/41, University of California, Berkeley, 1992.

[Shivakumar 02] P. Shivakumar, *et al.*, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," *Proc. International Conference on Dependable Systems and Networks*, pp. 389-398, 2002.

[Siewiorek 98] D. P. Siewiorek and R. S. Swarz, *Reliable Computer Systems: Design and Evaluation 3rd edition,* A. K. Peters, 1998.

[Sulistyo 02] J. B. Sulistyo and D. S. Ha, "Developing Standard Cells for TSMC 0.25µ Technology under MOSIS DEEP Rules", Department of Electrical and Computer Engineering, Virginia Tech, TR VISC-2002-01, Jan. 2002.

[Smith 78] J. E. Smith and G. Metze, "Strongly Fault Secure Logic Networks," *IEEE Trans. Computers,* Vol. C-27, pp. 491-499, Jun. 1978.

[Tan 94] C. Tan and J. Allen, "Minimization of Power in VLSI Circuits Using Transistor Sizing, Input Ordering, and Statistical Power Estimation," *Proc. Intl. Workshop on Low Power Design*, pp. 75–80, Apr. 1994.

[Teradyne 00] Technical Specifications, *J973-400 VLSI Test System*, Teradyne, Inc., 2000.

[Touba 97] N. A. Touba and E. J. McCluskey, "Logic Synthesis of Multilevel Circuits with Concurrent Error Detection", *IEEE Trans. Computer-Aided Design*, Vol. 16, No. 7, pp. 783-789, Jul. 1997.

[Tryon 62] J. G. Tryon, "Quadded Logic," *Redundancy Techniques for Computing Systems*, pp. 205-228, Spartan, 1962.

[Wagner 96] K. D. Wagner and S. Dey, "High-Level Synthesis for Testability: A Survey and Perspective," *Proc. Design Automation Conference*, pp. 131-136, Jun. 1996.

[Yang 91] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide," Technical Report 1991-IWLS-UG-Saeyang, MCNC, Research Triangle Park, NC, Jan. 1991.

[Zeng 99] C. Zeng and E. J. McCluskey, "Finite State Machine Synthesis with Concurrent Error Detection", *Proc. International Test Conference*, pp. 672-679, 1999.

[Ziegler 79] J. F. Ziegler and W. A. Lanford, "Effect of Cosmic Rays on Computer Memories," *Science*, Vol. 206, pp. 776-788, Nov. 1979.

[Ziegler 96] J. F. Ziegler, *et al.*, "IBM Experiments in Soft Fails in Computer Electronics (1978-1994)," *IBM Journal of Research and Development*, Vol. 40, pp. 3-18, 1996.

Vita

Kartik Mohanram was born in Hyderabad, Andhra Pradesh, India on September 14th 1976, the son of Pushpa Mohanram and P. J. Mohanram. He received the Bachelor of Technology (B. Tech.) degree in Electrical Engineering from the Indian Institute of Technology, Bombay in May 1998. He joined the department of Electrical and Computer Engineering at the University of Texas, Austin in September 1998. He will join the department of Electrical and Computer Engineering at Rice University, Houston as an assistant professor in September 2003.

Permanent Address: 4, 1st Block, HMT Layout,
Vidyaranyapura,
Bangalore 560097
India

This dissertation was typed by the author.