Masking timing errors on speed-paths in logic circuits

Mihir R. Choudhury and Kartik Mohanram
Department of Electrical and Computer Engineering
Rice University, Houston, TX 77005
{mihir,kmram}@rice.edu

Abstract

There is a growing concern about timing errors resulting from design marginalities and the effects of circuit aging on speed-paths in logic circuits. This paper presents a low overhead solution for masking timing errors on speed-paths in logic circuits. Error masking at the outputs of a logic circuit is achieved by synthesis of a non-intrusive error-masking circuit that has at least 20% timing slack over the original logic circuit. The error-masking circuit can also be used to collect runtime information when the speed-paths are exercised to (i) predict the onset of wearout and (ii) assist in in-system silicon debug. Simulation results for several benchmark circuits and modules from the OpenSPARC T1 processor are presented to illustrate the effectiveness of the proposed solution. 100% masking of timing errors on all speed-paths within 10% of the critical path delay is achieved for all circuits with an average area (power) overhead of 16% (18%).

1. Introduction

It is widely acknowledged that there will be a sharp increase in hardware failures in scaled CMOS technologies, e.g., [1, 2]. *Timing errors* resulting from process variability and manufacturing defects in scaled CMOS technologies are an important — and possibly dominant — failure mode that impact hardware reliability. In addition, long-term degradation effects like hot carrier injection, electromigration, and negative-bias temperature instability are also projected to increase timing failures with technology scaling. Timing errors are most pronounced on critical or near-critical paths, also called *speed-paths*, in multi-level control logic modules in integrated circuits. Control logic, with its irregular, multi-level structure and significant number of speed-paths is usually the most challenging part of an integrated circuit (i) to achieve timing closure on, (ii) to validate and debug during the post-silicon phase, and (iii) to achieve high fault coverage on during manufacturing test. The system-level effects of errors resulting from failures in control logic can be far reaching, especially given the recent trend toward highly integrated hardware platforms with multi-threaded, parallel execution environments. These factors motivate the development of cost-effective solutions to provide online protection to timing errors on speed-paths in logic circuits.

Conventional techniques for concurrent error detection (CED) in logic circuits provide good coverage when the stuck-at fault model is used to evaluate coverage to both transient and permanent faults [3, 4], but they are poorly suited to detect and/or mask timing errors. This is because the synthesis of CED circuits usually results in an error detection circuit with a larger critical path delay than the original circuit, making the error detection circuit more susceptible to timing errors. Online delay-fault detection techniques based on monitoring the outputs of the circuit [5–7] and re-sampling values of the output [8] suffer from the inability to detect errors due to late transitions outside the stability checking period determined during design. Moreover, design effort to pad short paths is necessary to increase the stability checking period [9]. Following timing error detection, error correction based on the above techniques incurs a performance penalty when rollback to the last correctly executed instruction is initiated for re-execution of the instruction stream. Circuit-level waveform monitoring using guard-bands [10] and other architecture-level techniques [11–14] have been proposed to predict or detect slowdown of paths due to wearout or temperature and voltage variations. The techniques in [10–13] target specific sources of slowdown — either variations or wearout — but not both. The technique proposed in [14] cannot detect errors dynamically during runtime, but requires periodic offline stress testing of the system in order to be effective.

This paper proposes a low cost error-masking solution that specifically targets timing errors on speed-paths in logic circuits. To the best of our knowledge, this is the first paper that proposes runtime masking of timing errors. An error-masking circuit is designed to mask timing errors arising on the application of input patterns that sensitize speed-paths of the logic circuit. The patterns that sensitize all speed-paths in the logic circuit are represented using a speed-path characteristic function (SPCF), and an efficient algorithm to compute the SPCF based on timing analysis of the logic circuit is described. The error-masking circuit is synthesized by using the SPCF to simplify the Boolean functions of the internal nodes in the technology-independent representation of the original logic circuit. By design, the error-masking circuit has at least 20% timing slack over the original logic circuit and is hence immune to timing errors. The inputs to the error-masking circuit are the inputs of the logic circuit and errors are masked only at the outputs of the logic circuit. Error masking is thus non-intrusive, and the critical path delay of the original circuit is not altered by the addition of the error-masking circuit.

Simulation results for several benchmark circuits and modules from the OpenSPARC T1 processor illustrate that 100% masking of timing errors arising on speed-paths within 10% of the critical path delay is achieved for all circuits with an average area (power) overhead of 18% (16%) and an average slack of 57% over the original circuit. In addition to masking timing errors, the error-masking circuit can also be used to log the application of and response to inputs that sensitize speed-paths. This runtime information can be used to (i) predict the onset of wearout through periodic analysis and (ii) assist in in-system silicon debug by expanding trace buffer windows through selective capture.

This paper is organized as follows. Section 2 motivates the proposed error-masking technique. Section 3 and 4 describe the algorithm for synthesis of the error-masking circuit. Section 5 presents results for various benchmarks. Section 6 draws conclusions and summarizes directions for future work.

This research was supported by NSF CAREER Award CCF-0746850.
.

978-3-9810801-5-5/DATE09 © 2009 EDAA

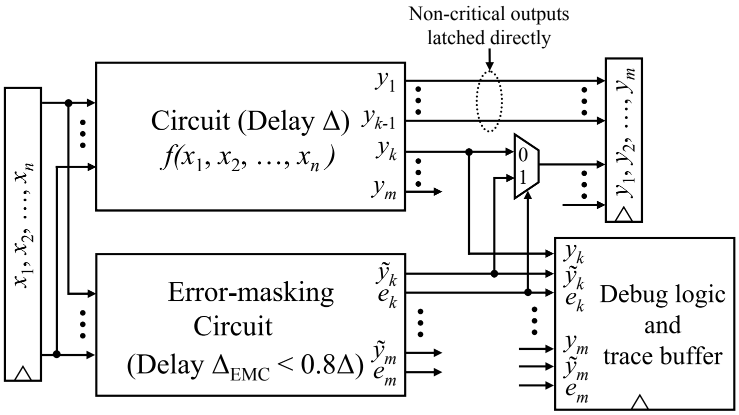

Figure 1: Proposed error-masking mechanism

2. Background and motivation

Several techniques have been proposed in literature to achieve resilience to timing errors arising due to design marginalities, environmental variations like temperature and voltage, and aging. Existing solutions can be broadly classified into techniques that require (i) logic redundancy or logic restructuring, (ii) output waveform monitoring, or (iii) architectural support.

Concurrent error detection is a logic redundancy based technique that traditionally uses the stuck-at fault model to evaluate and improve coverage to transient and permanent faults, e.g., [3,4,15–17]. However, these techniques are either intrusive, i.e., they require making modifications to the original circuit, or they result in an error detection circuit with long critical paths that is more vulnerable to timing errors. Although a totally self-checking concurrent delay testing technique was proposed in [18], it is based on duplication of the original logic circuit and hence vulnerable to common-mode timing errors. Logic restructuring techniques based on rewiring [19] and implication [20] transformations have also been proposed. These techniques target soft error rate reduction and their coverage of timing errors has not been demonstrated.

One class of waveform monitoring techniques monitor the outputs for late transitions that occur after the clock edge. These techniques are limited in their effectiveness since they cannot detect errors due to late transitions outside the monitoring period. Further, they are intrusive since short paths in the circuit must be padded to extend the monitoring period. Since such techniques focus on detecting timing errors, the errors are corrected through rollback to the last correctly executed instruction followed by re-execution of the instruction stream, which imposes control overhead and impacts performance. Examples of these techniques include monitoring using a sensing circuit [5–7] and output re-sampling after a certain delay [5, 8]. Re-sampling techniques also need to address data path metastability and increased clock energy due to the addition of an extra latch, which require design effort and overhead [9]. Output waveform monitoring that provides a guard-band at the outputs before the clock edge for wearout prediction has also been proposed in [10]. Since this technique is based on collection of timing data using sensors, it is specific to prediction of timing errors arising from gradual slowdown of speed-paths due to aging.

Several architecture-level techniques for resilience to variations and lifetime reliability, including dynamic on-chip verification [11], lifetime-reliability tracking based on technology parameters [12], wearout detection circuitry to predict the onset of wearout [13], periodic stress testing [14], and on-chip temperature and voltage sensors to predict temperature surges and voltage droop [21] have been proposed. A major drawback of these techniques is that they either target only specific sources of timing failures or require periodic offline stress testing in order to be effective.

This paper proposes a low cost online error-masking technique that specifically targets failures arising from timing errors on speed-paths in logic circuits. To the best of our knowledge, this is the first paper that proposes a solution for masking timing errors during runtime without any rollback. The proposed error-masking solution has several advantages. First, since a timing error is logically masked, unlike periodic monitoring techniques, the proposed technique is not restricted to any specific source of variation and can be used to mask timing errors due to temperature and voltage variations, aging and wearout related slow down of speed-paths, and errors arising in early lifetime due to latent defects and design marginalities. Second, the error-masking circuit is designed to have at least 20% timing slack over the original circuit. Hence, the error-masking circuit is immune to timing errors. Finally, by design, the error-masking circuit is non-intrusive since errors are masked directly at the primary outputs using a 2-to-1 multiplexer, Hence, there is a marginal, quantifiable impact on the critical path delay of the original circuit, which can be easily compensated for during synthesis.

2.1 Error-masking mechanism

The proposed solution for error-masking is based on synthesizing a circuit, referred to as the error-masking circuit, that correctly predicts the outputs of the circuit upon application of inputs that sensitize the speed-paths of the circuit. The basic mechanism of the proposed error-masking approach is shown in Fig. 1. The original logic circuit has inputs $x_1, x_2, ..., x_n$, outputs $y_1, y_2, ..., y_m$, and a critical path delay Δ. Errors are masked only at the critical outputs, i.e., outputs at which one or more speed-paths terminate. In Fig. 1, outputs $y_k, ..., y_m$ are critical and the error-masking circuit is used to mask timing errors only at these outputs. For each critical output y_i, $k \leq i \leq m$, the error-masking circuit produces two outputs $\tilde{y_i}$ and e_i. The first output $\tilde{y_i}$ predicts the correct value of y_i when a speed-path is sensitized in the fanin cone of y_i. The second output e_i is used to indicate that a speed-path is sensitized, i.e., e_i is 1 *if* a speed-path is sensitized to output y_i. The logic for the outputs $\tilde{y_i}$ and e_i are designed so that $\tilde{y_i}$ correctly predicts y_i when e_i is 1. Error masking at the output is performed using a 2-to-1 multiplexer. The indicator output is connected to the select input, y to the 0-input, and $\tilde{y_i}$ to the 1-input of the multiplexer. Thus, when a speed-path is sensitized, the select input (e_i) routes

Table 1: Accuracy vs. runtime for computing speed-path characteristic function with different approaches

Circuit	I/O	Area	Node-based approach [22]		Proposed path-based extension of [22]		Proposed short-path-based approach	
			Over-approximation		Exact		Exact	
			Critical patterns	Runtime (s)	Critical patterns	Runtime (s)	Critical patterns	Runtime (s)
C432	36/7	147	4.4×10^{10}	0.82	3.3×10^{7}	4.96	3.3×10^{7}	1
C2670	233/140	568	9.9×10^{67}	1.1	8×10^{66}	1.6	8×10^{66}	0.58
sparc_ifu_dec	131/146	887	6.4×10^{38}	0.01	4.2×10^{31}	0.07	4.2×10^{31}	0
sparc_ifu_invctl	173/115	442	3.04×10^{63}	0.4	3.46×10^{62}	0.59	3.46×10^{62}	0.32
lsu_stb_ctl	182/169	810	6.7×10^{52}	0.18	3.8×10^{50}	0.36	3.8×10^{50}	0.13

the 1-input ($\tilde{y_i}$) of the multiplexer to the output; otherwise e_i is 0 and the multiplexer routes the original output y_i to the output. Note that the error-masking circuit is designed so that it has at least 20% smaller critical path delay than the original circuit. Hence, the error-masking circuit is itself immune to timing errors on its speed-paths.

Wearout detection: The error-masking circuit can be used to detect the onset of wearout. As speed-paths slow down due to wearout and aging, timing errors at the critical outputs $y_k, ..., y_m$ start to increase. With the proposed error-masking circuit in place, these timing errors will be masked. However, the information that a timing error occurred, indicated by $e_i(y_i \oplus \tilde{y_i})$, can be recorded and analyzed offline periodically. For instance, a high timing error rate observed during offline analysis can predict the onset of wearout and the system can be designed to adapt dynamically to reduce the timing error rate.

Debug information: The error-masking circuit can also assist post-silicon at-speed in-system debug by guiding selective capture of debug information in trace buffers. Trace buffers are very useful because they can be used for real-time at-speed observation of limited signals during in-system debug [23,24]. However, trace buffers can only store a limited amount of data in one debug session. To optimize usage of trace buffers, selective storage of signal values on only a few suspect clock cycles has been proposed in [25]. Since errors occur mainly as timing errors on speed-paths, we propose that the error-masking circuit can provide runtime information to selectively store debug information. For a critical output y_i, the output e_i of the error-masking circuit indicates the application of an input pattern that sensitizes speed-paths that terminate at y_i. By storing debug information only when y_i is vulnerable to timing errors in the trace buffers, the window size of the trace buffers can be expanded significantly. This runtime identification of the application of patterns that sensitize speed-paths also increases the ability to debug unreproducible bugs.

With this background, we describe the proposed algorithm for computing the speed-path characteristic function in Sec. 3 and the synthesis of the error-masking circuit in Sec. 4.

3. Speed-path characteristic function

Consider a technology-mapped circuit C with primary inputs $x_1, x_2, ..., x_n$ and primary outputs $y_1, y_2, ..., y_m$. For ease of notation and without loss of generality, we will use an output $y \in \{y_1, y_2, ..., y_m\}$ for illustration, although our implementation considers all outputs simultaneously. For a given input pattern I, let y stabilize to the correct value after a finite non-zero delay Δ_I. The value of Δ_I depends on the applied pattern I, gate delays, and circuit structure. Given a target arrival time at output y, Δ_y, pattern I is a *speed-path activation pattern iff* $\Delta_I > \Delta_y$.

Definition: The *speed-path characteristic function (SPCF)* for an output y, denoted by $\Sigma_y(x_1, x_2, ..., x_n, \Delta_y)$, is the characteristic function for the set of all speed-path activation patterns. Thus, if speed-paths within 10% of the critical path delay Δ are targeted, $\Delta_y = 0.9\Delta$. In the rest of this paper, the SPCF at y is denoted by $\Sigma_y(\Delta_y)$ for brevity. Traditionally, the SPCF has been used in timing-driven optimization during logic synthesis [26] and in variable latency designs [27]. Many algorithms for computing the SPCF have been proposed in literature. In [27], a path-based algorithm for computation of the exact SPCF was proposed using an ADD-based timing analysis framework. The ADD-based approach traverses all paths in a circuit and is hence memory and time intensive, especially when a complex and realistic gate delay model is used. An over-approximation algorithm that traverses nodes on the critical paths to compute a super-set of the SPCF, instead of the exact SPCF, was proposed in [28]. Comparing the over-approximated SPCF, computed using the node-based algorithm presented in [28], to the exact SPCF indicates that the node-based approach presented in [28] may lead to large over-approximations of the SPCF for most circuits [22]. Hence, an extension of the node-based approach to reduce the over-approximation in the SPCF was presented in [22].

The extended node-based approach of [22] uses arrival and required time information to mark gates with a negative slack as critical. Using two functions, the long path activation function and the short path activation function, both statically and dynamically sensitizable patterns are computed in a single topological pass through the circuit. The algorithm is node-based because the critical gates are marked statically, i.e., before the topological pass through the circuit. Thus, if a gate has more than one fanout and the gate lies on a critical path only along one fanout, the gate is marked critical and input patterns that sensitize any path through this gate are included in the SPCF. Although node-based traversal makes it computationally efficient, the over-approximation in the SPCF arises as a consequence of node-based traversal, i.e., *statically* marking critical gates *before* the topological pass to compute the SPCF.

Our work extends the node-based algorithm from [22] to a path-based algorithm that computes the SPCF exactly. In our path-based approach, gates are not marked as critical based on required and arrival time information. Instead, a gate is marked as critical in the context of the path on which it lies. This enables the exact computation of the SPCF using a path-based algorithm. However, the accuracy of the path-based algorithm comes at the cost of computational complexity. The trade-off between accuracy and runtime for the node-based approach of [22] and the proposed path-based approach is illustrated in Table 1. The first 3 columns report the name, number of inputs and outputs, and area of the circuit. The SPCF is computed as the set of all patterns that sensitize speed-paths within 10% of the critical path delay. The number of critical patterns, i.e., the number of input patterns in the SPCF and the runtime for computing the set of critical patterns for the node-based approach [22] and the path-based extension are shown in columns 4 and 5. Note that the set of critical patterns computed using the node-based approach is always a super-set of the set of critical patterns computed using the proposed path-based approach. However, the path-based approach is, on average, 3.5 X slower than the node-based approach.

The computational complexity of the path-based extension of [22] can be attributed to the path traversals for the computation of the long path and short path activation functions. In this paper, we show that the computational complexity can be reduced significantly by computing the SPCF based on the short path activation function only. Consider a gate g with a single output z in a technology-mapped circuit with inputs $l_1, l_2, ..., l_k$. Let $f(l_1, l_2, ..., l_k)$ denote the Boolean function at z. Let δ_{l_i} denote the delay of input l_i to output z and Δ_z denote the target arrival time at z. Let $\overline{\Sigma}_z(\Delta_z)$ denote the complement of the SPCF at z, i.e., $\overline{\Sigma}_z(\Delta_z)$ is the set of all input patterns such that the value at z stabilizes before the target arrival time Δ_z. Let P be the set of all prime implicants in the on-set and off-set of f. Let $L(p)$ denote the set of literals in each prime implicant p, where $L(p) \subseteq \{l_1, l_2, ..., l_k\}$. The target arrival time at z is met *iff* the target arrival time is met for all the literals in at least one prime implicant in the off-set or on-set of f. Thus, $\overline{\Sigma}_z(\Delta_z)$ is given by

$$\overline{\Sigma}_z(\Delta_z) = \vee_{p \in P} \left(\wedge_{l \in L(p)} \overline{\Sigma}_l(\Delta_z - \delta_l) \right) \qquad (1)$$

Eqn. 1 can be used to recursively compute $\overline{\Sigma}_y$ for each primary output y of the circuit that contains speed-paths. The runtime for the proposed path-based algorithm is shown in column 6 of Table 1. Note that for runtimes that are comparable to the node-based approach, the proposed algorithm computes the SPCF exactly. The next section describes a synthesis algorithm for the error-masking circuit using the SPCF.

4. Synthesis of the error-masking circuit

Given a technology-mapped circuit C with inputs $x_1, x_2, ..., x_n$ and outputs $y_1, y_2, ..., y_m$. For ease of notation and without loss of generality, we will use an output $y \in \{y_1, y_2, ..., y_m\}$ for illustration, although our implementation considers all outputs simultaneously. Let $\Sigma_y(\Delta_y)$ denote the SPCF of output y, where Δ is the critical path delay for the design. Thus, if the speed-paths in the top 10% of Δ are targeted for protection by the error-masking circuit, then Δ_y is set to 0.9Δ. Note that if an output has a slack greater than 0.1Δ, the output is not critical. The objective of an error-masking circuit $\tilde{C}$ is to predict the correct output of C for patterns in $\Sigma_y(\Delta_y)$. For patterns not in $\Sigma_y(\Delta_y)$, the circuit $\tilde{C}$ need not predict the outputs of circuit C correctly, i.e., the patterns not in $\Sigma_y(\Delta_y)$ lie in the input don't care space for circuit $\tilde{C}$. Since the error-masking circuit $\tilde{C}$ must correctly predict the outputs of circuit C only on patterns in $\Sigma_y(\Delta_y)$, the circuit $\tilde{C}$ must also indicate when an output is correctly predicted. Thus, for an output y in circuit C, the error-masking circuit $\tilde{C}$ produces two outputs: (i) $\tilde{y}$ that predicts the correct value of y and (ii) e that indicates when the prediction of the error-masking circuit $\tilde{C}$ is correct. The logic functions for $\tilde{y}$ and e are designed so that e is 1 when a speed-path is sensitized, i.e., a pattern from the SPCF is applied and $\tilde{y}$ predicts the correct value of y when e is 1. Although the specifications of $\tilde{y}$ and e have a rich input don't care space to be exploited during synthesis, the Boolean functions of $\tilde{y}$ and e are inter-dependent (since $\tilde{y}$ must correctly predict y whenever e is 1) and this makes synthesis of $\tilde{C}$ challenging.

The rest of this section describes an algorithm for the synthesis of the error-masking circuit. The synthesis algorithm exploits the don't care space in the specification of circuit $\tilde{C}$ to optimize the area and delay of $\tilde{C}$. Hence, the error-masking circuit $\tilde{C}$ has a small area-power overhead (16–18% on average) and greater than 20% timing slack over the original circuit, as presented in Sec. 5. The large timing slack ensures that the error-masking circuit is itself immune to timing errors on its speed-paths. Before presenting the proposed algorithm for the synthesis of the error-masking circuit $\tilde{C}$, we will briefly describe two simple synthesis algorithms along with their limitations to motivate the proposed synthesis algorithm.

Bottom-up synthesis: The bottom-up approach involves direct synthesis, i.e., two level minimization of the incompletely specified function, followed by multi-level optimization of the logic for $\tilde{y}$ and e. Although, this approach can leverage the rich space of don't cares in the specification of $\tilde{y}$ and e for optimizing the logic, it is not scalable to circuits with more than 15–20 inputs. Further, the inter-dependence of the Boolean functions of $\tilde{y}$ and e makes bottom-up synthesis even more computationally demanding.

Top-down synthesis: A top-down approach uses circuit C as a starting point and simplifies the circuit for synthesizing the logic for $\tilde{y}$. This approach has several disadvantages: (i) it is not flexible because the synthesis is tied to the implementation of circuit C, (ii) this approach may yield circuits that are susceptible to common-mode timing failures, since the implementation of circuit $\tilde{C}$ will be structurally similar to circuit C, and (iii) since the Boolean functions of $\tilde{y}$ and e are inter-dependent, top-down synthesis from C may not effectively use the input don't care space. The top-down approach — in the extreme case — is based on duplication of the critical paths of C. This approach is ineffective because the duplicated paths will be as susceptible to timing errors as the critical paths in the original circuit.

4.1 Proposed synthesis algorithm

The technique proposed in this paper uses an intermediate representation — in the form of the technology-independent representation of the original circuit C — as a starting point for the synthesis of the error-masking circuit $\tilde{C}$. A technology-independent network is an intermediate representation of a circuit in which the internal nodes can have complex Boolean functions (with 10–15 inputs). The main advantage of starting with a technology-independent network is that the don't care space in the specification of the Boolean function for the error-masking circuit $\tilde{C}$ can be exploited effectively to simplify the Boolean expressions of the internal nodes and thus reduce the overhead of the error-masking circuit $\tilde{C}$. In addition, working with a technology-independent representation does not suffer from the scalability issues of the bottom-up approach described above because the Boolean expression of the internal nodes are limited to 10–15 inputs. We will now describe the technique for simplification of the technology-independent representation, T, of circuit C to obtain the technology-independent representation, $\tilde{T}$, of the error-masking circuit $\tilde{C}$.

Let $\Sigma_y(\Delta_y)$ denote the SPCF of an output y of C. Since the patterns in $\Sigma_y(\Delta_y)$ are targeted for error masking, $\Sigma_y(\Delta_y)$ is the input care-set for the logic cone of y. Let n_j be an internal node in the fanin cone of y in the technology independent network T. Let $a_1, a_2, ..., a_k$ be the inputs of n_j. In order to predict the output y correctly for input patterns in $\Sigma_y(\Delta_y)$, the output of n_j must be correctly predicted for the minterms in the satisfiability-care set induced by $\Sigma_y(\Delta_y)$ at the inputs of n_j. Let s_0 and s_1 denote the satisfiability-care minterms in the off-set and on-set of n_j. The Boolean expression of n_j can be simplified to ensure the correct

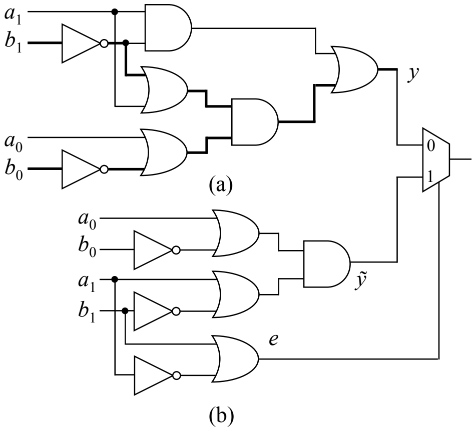

Figure 2: (a) 2-bit comparator with two speed-paths, (b) error-masking circuit for timing errors.

prediction of minterms in s_0 and s_1. Since n_j can have up to 10–15 inputs, the exact computation of the satisfiability-care minterms, s_0 and s_1, is computationally intensive. Hence, we propose a technique based on eliminating cubes from the sum-of-product (SOP) expressions of the on-set (n_j) and off-set ($\overline{n_j}$) to obtain reduced on-set (n_j^0) and reduced off-set (n_j^1) as follows: (i) the cubes in the SOP are arranged in the ascending order of number of literals, (ii) the *essential weight* of the j^{th} cube, c_j, is computed with respect to $\Sigma_y(\Delta_y)$ as the fraction of input patterns in $\Sigma_y(\Delta_y)$ covered by the c_j and not covered by $c_1, c_2, ..., c_{j-1}$, and (iii) cubes with a non-zero essential weight are selected and other cubes are discarded. Thus, n_j^0 covers all the minterms in s_0 and n_j^1 covers all minterms in s_1. Note that the proposed technique is computationally efficient because it uses a coarser granularity, i.e., the cubes of the SOP, rather than the individual minterms, to compute the cover. After generating the covers n_j^0 and n_j^1, the outputs $\tilde{n}_j$ and e_{n_j} for node n_j are given by,

$$\begin{aligned} \tilde{n}_j &= \overline{n_j^0} \text{ or } \tilde{n}_j = n_j^1 \\ e_{n_j} &= n_j^0 \oplus n_j^1 \end{aligned} \tag{2}$$

The output e_{n_j} is 1 when an input pattern from $\Sigma_y(\Delta_y)$ is applied and output $\tilde{n}_j$ predicts the correct value of n_j when e_{n_j} is 1. Note that e_{n_j} may be 1 for many more minterms apart from the minterms in $\Sigma_y(\Delta_y)$. Hence, the Boolean expression for e_{n_j} can be simplified further by elimination of cubes in its on-set that are not essential for covering minterms in $\Sigma_y(\Delta_y)$. The indicator output e_y for primary output y is 1 when all internal nodes in the fanin cone of y predict their outputs correctly. Thus, e_y is generated as a Boolean `and` of the indicator outputs, e_{n_j}, of all internal nodes n_j in the fanin cone of y. The simplified technology-independent network $\tilde{T}$ is then synthesized, optimized, and mapped to produce the error-masking circuit $\tilde{C}$.

4.2 2-bit Comparator

A 2-bit comparator is used to illustrate the proposed error-masking technique. Consider the 2-bit comparator shown in Figure 2(a) with inputs a_0, a_1, b_0, b_1 and output y. The comparator output, y, is 0 when the decimal equivalent of the binary number a_1a_0 is less than the decimal equivalent of b_1b_0. The optimal factored form for the off-set and on-set of y is,

$$\begin{aligned} y &= a_1\overline{b_1} + (a_0 + \overline{b_0})(a_1 + \overline{b_1}) \\ \overline{y} &= \overline{a_1}b_1 + \overline{a_0}b_0(\overline{a_1} + b_1) \end{aligned} \tag{3}$$

Assuming unit delay for an inverter and a delay of two units for 2-input gates, the critical path delay of the 2-bit comparator is 7. Suppose all speed-paths within 10% of the critical path delay are susceptible to timing errors, then Δ_y is 6.3. The speed-paths within 10% of the critical path delay have been highlighted in Fig. 2(a). The SPCF, $\Sigma_y(a_0, a_1, b_0, b_1, \Delta_y)$, is

$$\Sigma_y(a_0, a_1, b_0, b_1, \Delta_y) = \overline{a_1} + \overline{a_0}b_1$$

The satisfiability care-set induced by $\Sigma_y(a_0, a_1, b_0, b_1, \Delta_y)$ in the off-set and on-set of y, represented by s_0 and s_1, respectively, is

$$\begin{aligned} s_0 &= \overline{a_1}b_1 + \overline{a_0}b_0(\overline{a_1} + b_1) \\ s_1 &= \overline{a_1}\overline{b_1}(a_0 + \overline{b_0}) + \overline{a_0}\overline{b_0}a_1b_1 \end{aligned}$$

The cubes from Eqn. 3 that are selected to cover all the satisfiability care minterms, s_0 and s_1, are $y^0 = \overline{a_1}b_1 + \overline{a_0}b_0(\overline{a_1} + b_1)$ and $y^1 = (a_0 + \overline{b_0})(a_1 + \overline{b_1})$. Using y^0 and y^1, the Boolean function for the outputs of the error-masking circuit $\tilde{y}$ and e, as shown in Eqn. 2 are,

$$\begin{aligned} \tilde{y} &= y^1 = (a_0 + \overline{b_0})(a_1 + \overline{b_1}) \\ e &= y^0 \oplus y^1 = \overline{a_0} + b_0 + \overline{a_1} + b_1 \end{aligned} \tag{4}$$

Note that the error-masking circuit specified by Eqn. 4 covers extra minterms in addition to the minterms in $\Sigma_y(a_0, a_1, b_0, b_1, \Delta_y)$. The Boolean expression for e is simplified further by elimination of cubes in the on-set that are not essential to cover minterms in $\Sigma_y(a_0, a_1, b_0, b_1, \Delta_y)$. The simplified Boolean expression for e is

$$e = \overline{a_1} + b_1$$

This error-masking circuit is shown in Fig. 2(b).

5. Results

This section presents simulation results for the proposed error-masking technique. The simulations were run on a 64-bit 2.4 GHz Opteron-based system with 6 GB memory. The benchmark circuits were synthesized using Synopsys Design Compiler and mapped with the `lsi_10k` gate library.

The SPCF was computed for a target arrival time of 0.9Δ, where Δ is the critical path delay of the circuit. Table 2 presents area-power overhead of the proposed technique. Columns 1–3 report the name, number of inputs and outputs, and number of gates for each circuit. Column 4 reports the number of critical primary outputs, i.e., primary outputs that contain speed-paths. The number of input patterns in the SPCF over all critical primary outputs is reported in column 5. We observed that on average, about 20% of the primary outputs were critical primary outputs. The slack in the critical path delay of the error-masking circuit over the original circuit is reported in column 6. The area and power overhead of the error-masking circuit is reported in columns 7 and 8. For every benchmark circuit, 100% coverage for masking of timing errors was achieved, i.e., all the input patterns in the SPCF were covered by the error-masking circuit. The average area (power) overhead of the error-masking circuit is 16% (18%) and the average timing slack is 57%.

6. Conclusions

Timing errors resulting from process variability, manufacturing defects, and long-term degradation effects on logic circuit speed-paths are an important — and possibly dominant — failure mode that impact hardware reliability. This paper described the synthesis of a low-cost error-masking circuit to mask timing errors on speed-paths in a logic circuit. The error-masking circuit is itself immune

Table 2: Area and power overhead for 100% masking of timing errors on speed-paths

Circuit	I/O	No.	Critical	Critical	Slack	Overhead	
		gates	POs	minterms	(in %)	Area	Power
i1	25/16	33	3	5.6×10^6	30	34	30
cmb	16/4	13	1	4×10^3	52	26.1	19.1
x2	10/7	26	1	16	48.8	11.9	7.4
cu	14/11	26	4	3.6×10^3	60.8	6.25	5.4
too_large	38/3	230	2	8.7×10^7	44.5	25.3	21.5
k2	45/45	649	8	1.9×10^9	68.4	6.5	10.4
alu2	10/6	190	2	15	86	4	3
alu4	14/8	355	3	4	83.6	3.2	2.1
apex4	9/19	973	13	15	78	6.7	13.3
apex6	135/99	392	4	2.5×10^{36}	56.3	4.7	3.4
frg1	28/3	56	8	2.6×10^6	42.3	33	26
C432	36/7	95	4	3.3×10^7	67.1	45	40
C880	60/26	180	3	1.2×10^{13}	49.2	35	32
C2670	233/140	369	1	8×10^{66}	48.7	30	27
sparc_ifu_dec	131/146	556	3	4.2×10^{31}	22.5	12	10.5
sparc_ifu_invctl	212/72	312	22	3.5×10^{62}	55	18.5	12
sparc_ifu_ifqdp	882/987	1974	165	8.8×10^{107}	44.8	27.8	30.2
sparc_ifu_dcl	136/94	301	6	7.9×10^5	65.3	12	8
lsu_stb_ctl	182/169	506	5	3.8×10^{50}	65.9	9.8	8.4
sparc_exu_ecl	572/634	1515	211	1.4×10^{101}	75.4	7.8	6.7
Average	–	–	–	–	**57**	**18**	**16**

to timing errors, and can also be used in wearout prediction and post-silicon debug. Other potential applications of error-masking circuits, e.g., (i) adaptive speed-up of critical gates using body bias and (ii) aggressive dynamic voltage scaling by masking timing errors are areas for future research.

References

[1] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, pp. 10–16, 2005.

[2] D. Frank *et al.*, "Design and CAD challenges in 45nm CMOS and beyond," in *Proc. Intl. Conference Computer-aided Design*, pp. 329–333, 2006.

[3] D. K. Pradhan, ed., *Fault-tolerant computing: Theory and techniques*, vol. 1. Prentice-Hall, NJ, USA, 1986.

[4] M. Gössel and S. Graf, *Error detection circuits*. McGraw-Hill Book Company, London, UK, 1993.

[5] P. Franco *et al.*, "On-line delay testing of digital circuits," in *Proc. VLSI Test Symposium*, pp. 167–173, 1994.

[6] M. Favalli *et al.*, "Sensing circuit for on-line detection of delay faults," *IEEE Trans. VLSI Systems*, vol. 4, pp. 130–133, 1996.

[7] Y. Tsiatouhas *et al.*, "A sense amplifier based circuit for concurrent detection of soft and timing errors in CMOS ICs," in *Proc. Intl. On-line Testing Symposium*, pp. 12–16, 2003.

[8] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. Intl. Symposium on Microarchitecture*, pp. 7–18, 2003.

[9] K. Bowman *et al.*, "Energy-efficient and metastability-immune timing-error detection and instruction-replay-based recovery circuits for dynamic-variation tolerance," in *Proc. Intl. Solid-state Circuits Conference*, pp. 402–403,623, 2008.

[10] M. Agarwal *et al.*, "Circuit failure prediction and its application to transistor aging," in *Proc. VLSI Test Symposium*, vol. 32, pp. 277–286, 2007.

[11] T. Austin *et al.*, "DIVA: A reliable substrate for deep submicron microarchitecture design," in *Proc. Intl. Symposium on Microarchitecture*, pp. 196–207, 1999.

[12] J. Srinivasan *et al.*, "The case for lifetime reliability-aware microprocessors," in *Proc. Intl. Symposium on Computer Architecture*, pp. 276–287, 2004.

[13] J. Blome *et al.*, "Online timing analysis for wearout detection," in *Workshop on Architectural Reliability*, pp. 51–60, 2006.

[14] J. Smolens *et al.*, "Detecting emerging wearout faults," in *Workshop on Silicon Errors in Logic — System Effects*, pp. 276–287, 2007.

[15] S. Kundu and S. M. Reddy, "On symmetric error correcting and all unidirectional error detecting codes," *IEEE Trans. Computers*, vol. 39, pp. 752–761, 1990.

[16] N. K. Jha and S. Wang, "Design and synthesis of self-checking VLSI circuits," *IEEE Trans. Computer-aided Design*, vol. 2, pp. 878–887, 1993.

[17] N. A. Touba and E. J. McCluskey, "Logic synthesis of multilevel circuits with concurrent error detection," *IEEE Trans. Computer-aided Design*, vol. 16, pp. 783–789, 1997.

[18] A. Paschalis *et al.*, "Concurrent delay testing in totally self-checking systems," in *Journal of Electronic Testing: Theory and Applications*, vol. 12, pp. 55–61, 1998.

[19] S. Almukhaizim *et al.*, "On the minimization of potential transient errors and SER in logic circuits using SPFD," in *Proc. Intl. On-line Testing Symposium*, pp. 123–128, 2008.

[20] S. Krishnaswamy *et al.*, "Enhancing design robustness with reliability-aware resynthesis and logic simulation," in *Proc. Intl. Conference Computer-aided Design*, pp. 149–154, 2007.

[21] J. Tschanz *et al.*, "Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging," in *Proc. Intl. Solid-state Circuits Conference*, pp. 292–293,604, 2007.

[22] Y.-S. Su *et al.*, "An efficient mechanism for performance optimization of variable-latency designs," in *Proc. Design Automation Conference*, pp. 976–981, 2007.

[23] A. Hopkins *et al.*, "Debug support for complex systems on-chip: A review," in *Proc. Computers and Digital Techniques*, pp. 197–207, 2006.

[24] M. Abramovici, "In-system silicon validation and debug," *IEEE Design and Test of Computers*, vol. 25, no. 3, pp. 216–223, 2008.

[25] J.-S. Yang *et al.*, "Expanding trace buffer observation window for in-system silicon debug through selective capture," in *Proc. VLSI Test Symposium*, pp. 345–351, 2008.

[26] A. Saldanha *et al.*, "Performance optimization using exact sensitization," in *Proc. Design Automation Conference*, pp. 425–429, 1994.

[27] L. Benini *et al.*, "Telescopic units: A new paradigm for performance optimization of vlsi designs," *IEEE Trans. Computer-aided Design*, vol. 17, pp. 220–232, 1998.

[28] L. Benini *et al.*, "Automatic synthesis of large telescopic units based on near-minimum timed supersetting," *IEEE Trans. Computers*, vol. 48, pp. 769–779, 1999.