On Testing Timing-Speculative Circuits

Feng Yuan†, Yannan Liu†, Wen-Ben Jone‡ and Qiang Xu†

†CUhk REliable Computing Laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
Email: {fyuan,ynliu,qxu}@cse.cuhk.edu.hk

‡School of Electronics and Computing System
University of Cincinnati, USA
Email: jonewb@ucmail.uc.edu

ABSTRACT

By allowing the occurrence of infrequent timing errors and correcting them online, circuit-level timing speculation is one of the most promising variation-tolerant design techniques. How to effectively test timing-speculative circuits, however, has not been addressed in the literature. This is a challenging problem because conventional scan techniques cannot provide sufficient controllability and observability for such circuits. In this paper, we propose novel techniques to achieve high fault coverage for timing-speculative circuits without incurring high design-for-testability cost. Experimental results on various benchmark circuits demonstrate the effectiveness of the proposed solution.

1. INTRODUCTION

With aggressive technology scaling, the timing behavior of integrated circuits (ICs) is increasingly sensitive to process, voltage, and temperature (PVT) variations and aging effects [1, 2]. Conventional circuit design and optimization techniques guarantee that all circuit paths meet their timing requirements in all conditions to ensure error-free computing [3]. Such worst-case design methodologies, however, inevitably lead to pessimistic designs because a large design guardband needs to be incorporated to prevent any timing failure.

To address the above problem, a "better-than-worst-case (BTWC) design" methodology that allows reliability to be traded off against power and performance was proposed, which is able to dramatically improve the energy-efficiency of computation [4]. The basic idea behind BTWC design methodology is that, since circuit non-idealities mainly manifest themselves as infrequent timing errors on critical paths of the circuit (if sufficient design guardband is not incorporated) [5], we can over-clock the chip and/or reduce the supply voltage of the chip to a point where timing errors occur, and achieve resilient computation (instead of error-free computation) by performing timing error detection and correction. This approach is generally referred to as *timing speculation*, which has attracted lots of interests from both academia and industry.

To enable timing speculation, a number of timing speculators (e.g., [6, 7, 12, 13]) were presented in the literature, which capture timing errors occurred at flip-flops driven by critical paths (referred to as *suspicious FFs*) based on double sampling or late transition detection. With timing error detection capability, a circuit can react to each error quickly and recover from it by rolling back to a known-good pre-error state. By doing so, the circuit energy-efficiency can be significantly improved. Recently, Intel [7] has demonstrated in their test chip that a timing-speculative microprocessor is able to achieve more than 30% throughput gain when compared to a conventional microprocessor design under the same supply voltage. The above benefits have motivated a large amount of recent research efforts on design and optimization techniques for timing-speculative circuits (e.g., [8–11]).

While great potential for timing-speculative circuits was shown with promising test chip measurement results, there are still many challenges to overcome before they can be fully commercialized. One of the key challenges is how to conduct effective and efficient manufacturing test for timing-speculative circuits, which is fundamentally different from conventional circuit testing that targets zero timing failure. The main differences lie in the following aspects:

- Conventional VLSI testing regards a chip to be defective if it cannot pass at-speed delay tests, while timing-speculative circuits are inherently tolerant to timing errors. Therefore, the pass/fail criteria for timing-speculative circuits need to be re-examined.
- To detect timing errors, a certain period right after the clock edge (namely *detection window*) is used to monitor late transitions on suspicious FFs. Any occurrence of transitions in this detection window is regarded as a timing error. To distinguish late transitions on critical paths from those early arrivals on short paths so as to guarantee the correctness of error detection, the propagation delay on each short path driving any suspicious FF must be larger than the detection window, denoted as *min-delay constraint*. It is therefore essential to identify those defective chips that violate the above min-delay constraint during manufacturing test of timing-speculative circuits.
- the newly-introduced circuitries for timing speculation, including each timing speculator itself and the system-level error signal collection logic, need to be fully tested.

Due to the above, it is essential to develop new test methodologies for timing-speculative circuits, which are addressed in this paper. To the best of our knowledge, this is the first work that considers timing-speculative circuit testing. The contributions of this paper include the following:

- We conduct fault analysis and identify new types of faults that need to be considered for timing-speculative circuits.
- We introduce new test flow and design-for-test (DfT) structures for timing-speculative circuits.
- We present novel test solutions to achieve high fault coverage for timing speculators and timing error collection logic without incurring high DfT cost. The associated test pattern generation procedure is largely compatible with conventional ATPG techniques.

DAC'2013, May 29 - June 07 2013, Austin, TX, USA.

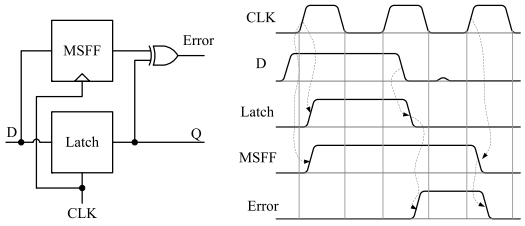
The remainder of this paper is organized as follows. Section 2 presents the preliminaries of this work on timing-speculative circuit designs. In Section 3, we conduct fault analysis for timing-speculative circuits. The proposed test methodologies and the corresponding test generation techniques are detailed in Section 4 and Section 5, respectively. Experimental results on various benchmark circuits are next presented in Section 6. Finally, Section 7 concludes this paper.

2. PRELIMINARIES

Various timing-speculative designs have been presented in the literature (e.g., [6, 7]). While they differ in some aspects (e.g., the internal structure of timing speculators), the basic design principle is quite similar. In this work, we present the proposed test methodology based on the recent Intel timing-speculative design shown in [7], and we briefly discuss it in this section.

2.1 Timing Speculator

The timing speculator design in [7] is depicted in Fig. 1(a), which consists of a latch, a shadow master-slave FF (MSFF) and a *XOR* gate. The latch operates as a datapath state element for normal computation, while the MSFF samples the input value again for timing error detection. The timing diagram depicted in Fig. 1(b) demonstrates the operation of the timing speculator. Since the latch is on during high clock phase while the MSFF captures data at clock rising edge, normally, the input value is stable before clock rising edge and both the latch and MSFF hold the same value. When a late transition occurs, the value captured by the MSFF is different from the one just captured by the latch, and the *XOR* gate sets the error signal accordingly. Consequently, with such a timing speculator design, the high clock phase is the detection window for timing errors.



(a) Timing speculator (b) Timing diagram

Figure 1: Timing speculator in [7].

2.2 Timing-Speculative Pipeline Design

Fig. 2 presents the overall timing-speculative design in [7], which is a three-stage pipelined microprocessor with timing speculation capability. It can be observed that all the pipeline flip-flops (FFs) are replaced with timing speculators, and error signals from timing speculators in the same stage are grouped with an *OR*-gate tree, whose output is captured by a *Final Error FF*. The three pipeline error signals are propagated to the input buffer controller in one cycle, which determines the appropriate instructions to replay based on the three pipeline error signals. They are also pipelined to the output buffer controller to invalidate erroneous data. In a microprocessor, the instruction replay circuits could leverage existing circuit used to recover from a branch miss-prediction. If a timing error occurs, the input buffer signals the *clock divider* to halve the clock frequency to ensure correct operation during replay, meanwhile, to maintain a constant high clock phase to avoid min-delay constraint violation (i.e., the detection window size remains unchanged).

2.3 Clock Divider and Duty-Cycle Controller

The clock divider and duty-cycle control circuits and the corresponding conceptual timing diagram are presented in Fig. 3. A clock generator with a differential pulse-splitter creates differential inputs CLKIN and CLKIN#. The Half_Fre input is controlled by the input buffer shown in Fig. 2. The CLK output is distributed throughout the chip.

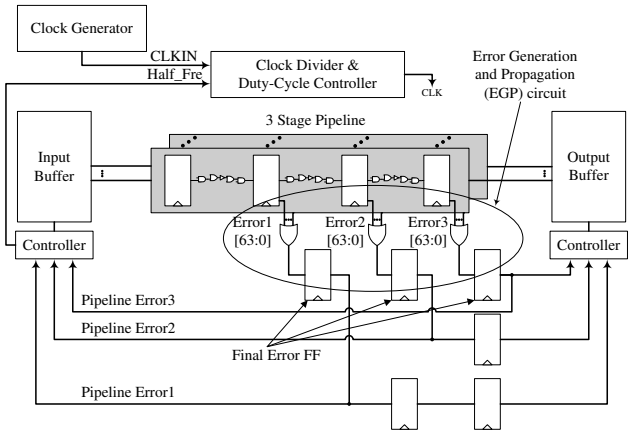


Figure 2: Timing-speculative microprocessor design in [7].

CLKIN and CLKIN# are inputs to a differential amplifier that generates an intermediate clock signal. This intermediate clock signal and the output of the falling edge-triggered MSFF in Fig. 3(a) are inputs to a logic-AND gate to produce the clock divider output (CLK0). When the Half_Fre input is logic '0' (i.e., no timing error occurs, and the circuit is running in normal functional mode), the output of the falling edge-triggered MSFF remains logic '1', and thus CLK0 and CLKIN have the same frequency. When the Half_Fre input is asserted (i.e. timing error is detected, and the circuit is running in recovery mode), the output of the falling edge-triggered MSFF toggles every other cycle, enabling the clock divider circuit to skip every other high phase of CLKIN as illustrated in Fig. 3(b). The duty-cycle control is performed with a logical-AND of CLK0 and a delayed CLK0# (i.e., inversion of CLK0) with CLK as the output, and the delayed CLK0# determines the length of the CLK high phase. With this duty-cycle control circuit, the CLK high phase delay remains constant value (T_H) at both normal and recovery modes, which is essential to ensure min-delay constraint is not violated. It is worth noting that the CLK high phase T_H is tunable with the reconfigurable duty-cycle control circuit, controlled by scan bits (see Fig. 3(a)). Considering we can use on-chip PLL to control the clock cycle time T_{cycle}, T_L is tunable as well. We leverage these tunable units in our proposed test methodologies (discussed later).

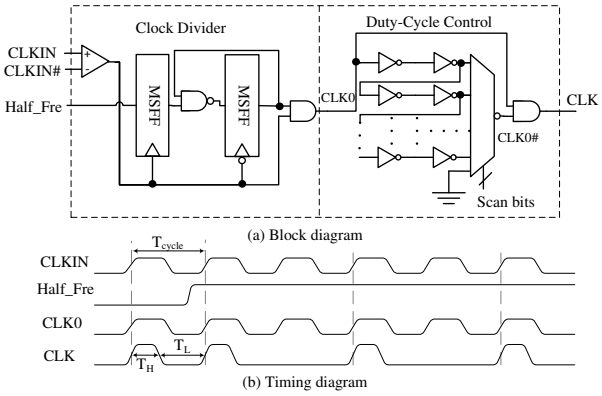


Figure 3: Illustration for Clock Divider & Duty-Cycle Controller Block Proposed in [7].

3. FAULT ANALYSIS FOR TIMING-SPECULATIVE CIRCUITS

In a timing-speculative circuit, for a regular FF that is not driven by critical paths, the following timing constraints apply:

$$\begin{aligned} T_{max\_r} &< T_{cycle} - T_{setup,clk\_r} \\ T_{min\_r} &> T_{hold,clk\_r}, \end{aligned} \tag{1}$$

where, T_{cycle} is the clock cycle time, T_{max_r} and T_{min_r} are the maximum path delay and the minimum path delay driving the regular FF,

T_{setup,clk_r} and T_{hold,clk_r} are the setup time constraint and the hold time constraint based on the clock rising edge, respectively.

For a suspicious FF, since there is an extra timing error detection window that is the high clock phase, denoted as T_H (similarly, we denote the low clock phase as $T_L = T_{cycle} - T_H$), the maximum path delay constraint T_{max_s} is as follows:

$$T_{max\_s} < T_{cycle} + T_H - T_{setup,clk\_f}, \quad (2)$$

where T_{setup,clk_f} is setup time based on the falling clock edge. As discussed earlier, in order not to treat early transitions from short paths as timing errors, we have the minimum path delay constraint T_{min_s} for a suspicious FF as

$$T_{min\_s} > T_H + T_{hold,clk\_f}, \quad (3)$$

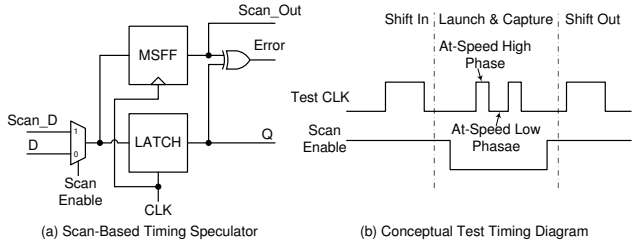
and T_{hold,clk_f} is the hold time based on the falling clock edge.

Next, let us consider error generation and propagation (EGP) circuit path (i.e., from a timing speculator to the Final Error FF, see Fig. 2). In the worst case, the timing error may appear at the end of detection window, thus the maximum error propagation delay T_{max_e} should satisfy

$$T_{max\_e} < T_{cycle} - T_{setup,clk\_r} - T_H + T_{setup,clk\_f}. \quad (4)$$

It can be observed from the above equation that the worst case error propagation time is reduced roughly by T_H.

With conventional delay testing, we usually conduct at-speed test of critical paths with functional clock only. However, for timing-speculative circuits, we need to conduct the following new types of path delay faults: (i) long path delay fault for suspicious FFs according to Equation 2; (ii) short path delay fault for suspicious FFs according to Equation 3; (iii) EGP circuit path delay fault according to Equation 4. Moreover, we need to target those static faults (e.g., stuck-at faults) that affect the logic function of the EGP circuit.



(a) Scan-Based Timing Speculator (b) Conceptual Test Timing Diagram

Figure 4: Scan Architecture for Timing-Speculative Circuits

4. PROPOSED TEST METHODOLOGY FOR TIMING-SPECULATIVE CIRCUITS

In this section, we first present the DfT structure for timing-speculated circuits. Following that, we present the proposed test methodologies.

4.1 DfT for Timing-Speculative Circuits

In order to control and observe the values of timing speculators, we need to be able to scan them. Since latches are not capable of performing shift operations properly, we propose to construct scan chains only using the shadow MSFF of each timing speculator, as shown in Fig. 4(a). By doing so, test stimuli can be shifted in to setup the logic values of both the MSFF and the latch in a timing speculator while test responses captured by the shadow MSFF will be shifted out. To perform at-speed delay test, we use functional clock during the capture phase, as shown in Fig. 4(b). Moreover, we leverage the clock divider and duty-cycle controller in timing-speculative designs (see Section 2.3) to manipulate test clocks, whenever necessary. That is, we are able to control T_H, T_L and hence T_{cycle} during testing.

The above scan design, however, is not sufficient to test the EGP circuit due to the difficulty in error signal generation. To be specific, asserting the error signal of a particular timing speculator requires careful timing control so that the values latched in the MSFF and the latch are different, which is quite difficult to achieve. It is even harder to control the timing of a late signal transition at the end of T_H (see Equation 4), which is required to tolerate dynamic variations. A straightforward method to resolve this issue would be adding a dedicated falling edge-triggered shadow flip-flop to generate an artificial *error signal* directly when testing the EGP circuits. However, this method incurs high DfT cost and also complicates the design of timing speculators. Instead of doing so, we propose *non-intrusive* test solutions to address this problem, as discussed in the following subsection.

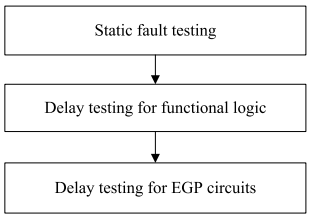


Figure 5: The overall flow for testing timing-speculative circuits

4.2 Proposed Test Methodologies

According to the timing-related fault analysis discussed in Section 3, for a timing-speculative circuit, we need to test long paths driving suspicious FFs, long paths driving regular FFs, short paths driving suspicious FFs, as well as EGP circuit testing.

The test flow is presented in Fig. 5. First, we conduct static fault testing (e.g., stuck-at faults) to ensure that circuit does not contain logic errors, including static faults in EGP circuits (discussed later). Next, we perform the above-mentioned delay tests. To test a short path driving a specific suspicious FF, we apply its test patterns and observe the corresponding error signal result at the end of the timing error detection window T_H. To ensure that the error can be correctly captured, we fix T_H as its functional value $T_{H\_functional}$ and prolong T_{cycle} to remove the impact of possible EGP circuit delay fault (which is addressed later in the test flow). The test results are then captured into Final Error FFs and scanned out for observation. Long path delay test for regular FFs can be performed with conventional delay testing methods using functional clock cycle $T_{cycle\_functional}$. Finally, to test a long path driving a suspicious FF, we configure the test clock cycle to be $T_{cycle} = 2T_{H\_functional} + T_{L\_functional}$ (due to timing speculation), and then apply its test patterns and observe the test response. Any error identified during the above at-speed delay testing process means the circuit is defective and the chip under test must be abandoned.

As shown in Fig. 5, the last step in the test flow is to conduct delay testing for EGP circuits. Without adding dedicated DfT circuits for such faults, we discuss how to achieve high fault coverage by novel test application techniques after introducing our test strategies for static faults in EGP circuits.

4.2.1 Static Fault Testing for EGP Circuits

In order to test static faults in EGP circuits, we intentionally generate and capture each error signal with a low-speed test clock, as illustrated in Fig. 6. This figure shows the scenario where a $0 \rightarrow 1$ transition propagates through two consecutive scan cells. It can be observed that, starting from Scan_Out0, the transition takes delay D_{sc} to arrive at Scan_D1 because of the scan chain delay. Since the test clock is low-speed, we can assume T_H is much larger than D_{sc}, and thus logic '1' will be captured by Latch1 first. MSFF1 will latch the transition at the next clock's rising edge, different logic values occur on Latch1 and MSFF1, and Error1 is set to logic '1'. Then, we set Scan_Enable to logic '0', so that the circuit is applied in functional mode and Error1 will be captured by the corresponding Final Error FF.

Test pattern generation is quite straightforward. Based on the static fault model, we only need to control the proper transition position in the scan data to align with the targeted suspicious FF position in the scan chain. Nevertheless, one thing needs to be pay attention to is that two timing speculators belonging to the same error group cannot be tested by the same pattern, because their error signals will mask each other with the *OR*-gate tree used for error grouping and propagation.

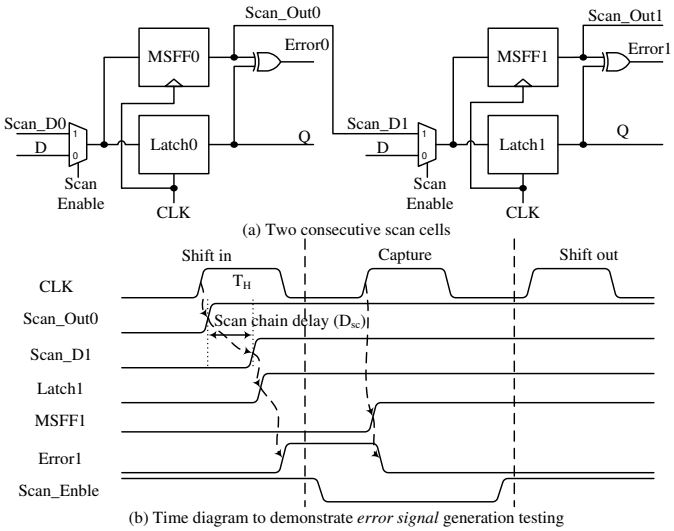


Figure 6: The proposed error signal generation method.

4.2.2 *Delay Testing for EGP Circuits with Long Path Sensitization*

To apply traditional long path delay patterns to detect delay faults in EGP circuits, the basic principle is to purposely generate a timing error on a particular long path, and make it propagate through the OR tree under the at-speed clock condition. Due to design guardband and process variation, however, there is no guarantee for the occurrence of timing errors, since whether the targeted path delay is larger than functional clock cycle ($T_{cycle\_functional}$) is unknown for us. Therefore, both the logic value captured by the MSFF of a targeted suspicious FF and its corresponding Final Error FF state have to be observed to make the pass/fail decision, and the criteria are listed in Table 1. It is important to emphasize that, the logic value stored in the MSFF of the suspicious FF appears in the second at-speed test clock cycle, while the error signal can only be latched into its Final Error FF in the next clock cycle. Therefore, to observe both of them, we have to apply the same test patterns twice: one is to capture the MSFF state with two at-speed clock cycles, while the other is to capture the Final Error FF state with three at-speed clock cycles.

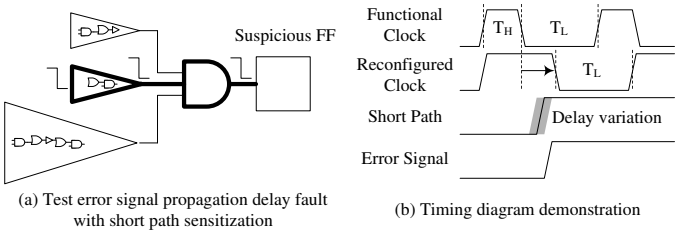
Case ID	Captured Value vs Golden Value	Final Error FF State	Result
A	Same	0	Unknown
B	Same	1	Fail
C	Different	0	Fail
D	Different	1	Pass

Table 1: Pass/Fail criteria for EGP circuit delay testing with long path sensitization.

As shown in the table, there are four possible test results. Case A means no timing error occurs, and hence EDP circuit is not tested. Case B presents the scenario that the targeted path delay is smaller than one clock cycle while an error signal is still generated. Such kind of result may occur when static faults exists in EGP circuit or relevant short paths violating the min-delay constraint are sensitized (due to incomplete testing with previous test steps in our test flow). Therefore, it is a test fail. Case C represents the situation where a timing error occurs but its error signal is not captured because of a delay fault in the EGP circuit, and thus it is a failed test. Case D is a passed test, and it implies that the timing error is correctly captured under the at-speed test clock.

Due to the existence of Case A, delay testing for EGP circuits with long path sensitization cannot guarantee high fault coverage[1]. We therefore propose another method to test such faults using short path sensitization, as discussed in the following.

[1]Note that, tightening the clock period with smaller T_{cycle} may facilitate to set the error signal, but it is not trustworthy due to possible glitches on long paths.



(a) Test error signal propagation delay fault with short path sensitization

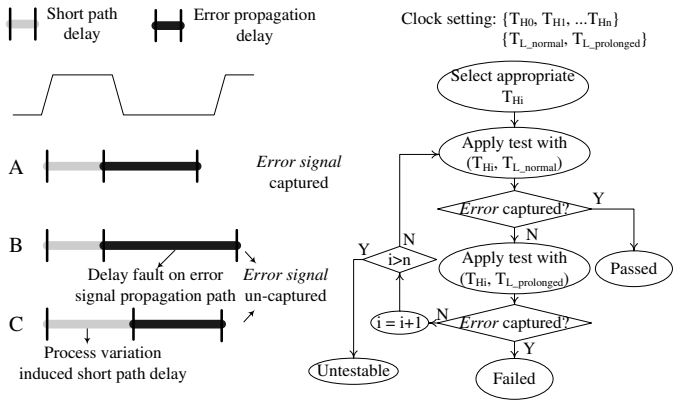
(b) Timing diagram demonstration

Figure 7: EGP circuit delay testing with short path sensitization

4.2.3 *Delay Testing for EGP Circuits with Short Path Sensitization*

In circuit normal function mode, all short paths have to obey the min-delay constraint, i.e., $T_{H\_functional}$ must be smaller than all short path delays in order to guarantee that transitions on the short paths are not accidentally treated as timing errors. By making T_H longer than $T_{H\_functional}$ with the help of the clock divider and duty-cycle controller, however, we can purposely select a short path to drive each suspicious FF to generate an "early transition", thus generating the error signal, as shown in Fig. 7. We use this method for delay testing of EGP circuits.

Due to process variation, the estimated short path delay based on timing analysis is inherently inaccurate and it is very difficult to setup test clock configurations for error signal generation. We use Fig. 8(a) for illustration. Cases A, B and C depict three possible situations after test clock reconfiguration, wherein the light line denotes a short path delay and the dark line depicts an error signal propagation delay. As shown in Case A, the error signal generated by a timing speculator will be captured by its corresponding Final Error FF, if we correctly estimate the short path delay and no delay fault occurs on the error signal propagation path, the circuit passes the test. However, if no error signal is captured by the Final Error FF, there are two possibilities: (i) there exists a delay fault on the error signal propagation path (e.g., Case B); (ii) the short path fails to generate an error signal (e.g., Case C due to process variation). Without distinguishing these two cases, we cannot make a correct test decision.



(a) Ambiguous test result

(b) Process variation aware test flow

Figure 8: Short path variation-aware test methodology

One observation to solve this problem is that, for Case C, once T_H is fixed we can never capture the error signal even if a prolonged T_L clock setting is applied, while we can still capture the error signal by prolonging T_L for Case B. Based on this observation, we propose a process-variation-aware test flow to distinguish the above two cases, as shown in Fig. 8(b). Here, we assume there exists a set of clock configuration $\{T_{H0}, T_{H1}, ..., T_{Hn}\}$ and $\{T_{L\_functional}, T_{L\_prolonged}\}$, where T_{Hi} is sorted in an ascending order, $T_{L\_functional}$ is simply the functional T_L, and $T_{L\_prolonged}$ is a much larger value than $T_{L\_functional}$. Considering a single suspicious FF, we first select an appropriate T_{Hi} based on the static timing analysis result of a sensitized short path, and apply at-speed delay testing under the clock setting $(T_{Hi}, T_{L\_functional})$. If an error signal is captured by the Final Error FF, the delay test passes successfully (Case A). Otherwise, we reconfigure the clock as $(T_{Hi}, T_{L\_prolonged})$ and

apply the same test again. A delay fault on the propagation path can be detected if the error signal is captured (Case B). If the error signal is not captured, we first check whether we can further prolong T_H with another configuration. If yes, we sweep to $T_{H(i+1)}$ and repeat the test flow again, otherwise the suspicious FF is untestable even with short path sensitization.

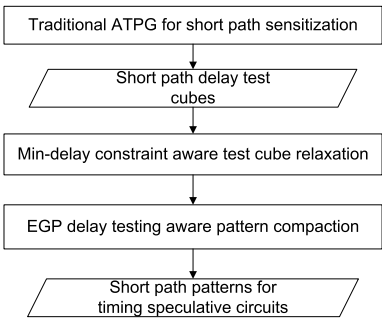


Figure 9: Test pattern generation flow

5. TEST PATTEN GENERATION

In this section, we introduce test pattern generation algorithm for short path sensitization, and the generated patterns are used for EGP delay testing and short path min-delay constraint checking.

As discussed earlier, testing EGP circuits with short path sensitization usually require multiple test clock configurations. To reduce the associated testing time cost, we propose several techniques to enhance test parallelism (i.e., test multiple faults concurrently whenever possible), and the test generation flow is shown in Fig. 9. We first acquire test cubes for short paths with conventional ATPG tool. Next, we propose a novel test cube relaxation technique by taking advantage of the min-delay constraint checking requirement. By doing so, we are able to have a more compacted test pattern set, and our test compaction algorithms are described afterwards.

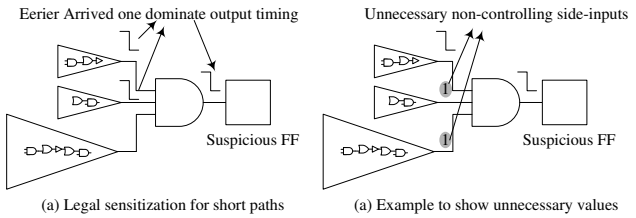


Figure 10: Example to motivate short path test cube relaxation

5.1 Test Cube Relaxation

As the objective of short path testing is to check min-delay constraint violations, we try to use one pattern to sensitize multiple short paths simultaneously. An example is shown in Fig. 10(a). As can be observed for this example circuit, the falling transition at the *AND* gate's output will be dominated by the earliest arrived transition since logic '0' is the controlling value of each *AND* gate. Consequently, the delay of the shortest path determines the transition time and it does not matter which short path generates the earliest transition. With conventional ATPG, however, the side-inputs are set to be non-controlling values to test each individual path, as shown in Fig. 10(b). This is not necessary and we use this feature for test cube relaxation and we use the algorithm proposed in [14] to achieve the above objective.

5.2 Test Pattern Compaction

By merging compatible test cubes into a single test pattern, both test pattern count and testing time can be reduced. However, as the testing time is mainly determined by the number of test clock configurations for EGP circuit delay testing, it is essential to take this into consideration when conducting test pattern compaction. Let us consider the following example with two short paths, e.g., path A with 0.25ns propagation delay and path B with 1.25ns propagation delay. According to the test flow in Fig. 8(b), we need to set different clock configurations to test these two paths, due to the large gap between their propagation delays. In this case, even though these two paths can be sensitized simultaneously by the same pattern, we still need to apply this pattern twice with our test application strategy. From this perspective, it is preferable to sensitize multiple paths with close delay values simultaneously by a single test pattern, so that they have a high chance to share the same clock configuration when sweeping the T_H value.

Based on the above observation, we propose to divide the sensitized short paths into several groups according to their propagation delay values (obtained with static timing analysis), and two test cubes can be compacted into one pattern if their sensitized paths belong to the same short path delay group. We denote the above constraint as *delay group constraint* during test compaction. In addition, as we need to observe the error signal for making test pass/fail decision, all short paths whose driven suspicious FFs belong to the same error group (see Fig. 2) cannot be compacted into one test pattern either, because otherwise error signals may cancel each other. We name this rule as the *error group constraint* during test compaction. With the above two constraints, we use the algorithm in [15] for test pattern compaction.

benchmark	SFF (#)	SFF (#) detectable using long path	SFF(#) detectable using short path	
			One config.	Multiple config.
s38417	469	36.113	340.16	469
s38584	424	3.5616	348.99	424
ethernet	21	18.00	3.73	21
wb_conmax	86	15.1618	62.245	86
des_perf	161	160.678	115.095	161

Table 2: Results for EGP circuit coverage with different sensitization methods.

6. EXPERIMENTAL RESULTS

6.1 Experimental Setup

To evaluate the effectiveness of our proposed timing-speculative circuit testing technique, we conduct experiments on two large ISCAS'89 benchmark circuits, *s38417* and *s38584*, as well as three large IWLS benchmark circuits, *wb_conmax*, *ethernet* and *des_perf*, which are the largest benchmark circuits available to the public domain.

In the experimental flow, we first synthesize benchmark circuits with commercial tools to obtain the optimized circuit netlist. Next, based on timing analysis results reported with Synopsys PrimeTime, we set 70% of the circuit's longest path delay as the threshold to differentiate regular FFs and suspicious FFs. That is, for those FFs whose driving path delays are larger than this threshold, they are replaced with timing speculators. These timing speculators are randomly grouped for error propagation, and each group holds up to 32 error signals. After that, we manually insert *OR* gate trees and Final Error FFs to obtain the timing-speculative designs. Finally, path delay patterns generated with Synopsys TetraMax are converted to timing-speculative circuit test patterns with our proposed pattern generation algorithms. To evaluate the impact of process variation on our proposed solution, we conduct Monte Carlo simulation to generate 10,000 circuits for each benchmark under 10% gate delay variation based on Gaussian distribution.

6.2 Results and Analysis

In the first experiment, we show the EGP circuit delay fault coverages achieved by long path sensitization and short path sensitization. The experiments are conducted with 10,000 circuits under process variation, and results are shown in Table 2. For each benchmark circuit, the total number of suspicious FFs is shown in Column 2. Columns 3-5 present the average number of detectable circuit paths following suspicious FFs in the EGP circuit, with long path sensitization, short path sensitization without test clock reconfigurations, and short path sensitization with test clock reconfigurations, respectively. It can be seen from the table, the proposed method using short paths to sensitize EGP circuits with test clock reconfigurations is able to achieve 100% coverage for EGP circuits, while the other two methods cannot achieve this objective.

benchmark	Short path (#)	Compacted pattern count				
		Ori.	Ori. (DG & EG)		Relaxed (DG & EG)	
			#	INC	#	INC
s38417	3445	126	197	56.35%	129	-34.51%
s38584	2521	111	152	36.94%	116	-23.68%
ethernet	33	12	23	91.67%	21	-8.70%
wb_conma	841	21	40	90.48%	30	-0.25%
des_perf	300	3	10	233.33%	10	0.00%
Avg.				101.75%		-18.38%

Table 3: Test pattern count results for EGP circuit testing

benchmark	Compacted pattern count			Testing time ($\times 10^3$ clock cycles)		
	EG	DG & EG	INC	EG	DG & EG	INC
s38417	86	129	50.00%	1449.916	985.6	-32.02%
s38584	78	116	48.72%	936.48	774.435	-17.30%
ethernet	21	21	0.00%	125.425	114.88	-8.41%
wb_conmax	19	30	57.89%	398.86	274.215	-31.25%
des_perf	10	10	0.00%	59.405	57.805	-2.69%
Avg.			31.32%			-18.34%

Table 4: Testing time results for EGP circuit testing

In the next experiment, we show the test pattern count for EGP circuits with our proposed solution, as shown in Table 3. Column 2 presents the number of sensitized short paths for each benchmark circuit. Without considering both the error group constraint (*EG*) and delay group constraint (*DG*), we conduct test compaction on the raw patterns generated by the ATPG tool directly, denoted as *Ori.*. When taking both *DG* and *EG* constraints into consideration for test compaction, we can see from Columns 4 and 5 that the pattern count increases 101% on average. With the proposed min-delay constraint aware test cube relaxation technique, we are able to mitigate the negative effect caused by these constraints. The average saving is about 18% . In particular, when the number of original test patterns is high, the proposed method is able to achieve better savings, e.g., for s38417 and s38584.

Table 4 evaluates the testing time with the proposed methodology that intentionally sensitizes paths with close delay values. Because the number of test patterns is determined by the real circuit short path delays, this experiment is also conducted with 10,000 circuits under process variation. Two test sets are compacted from the relaxed patterns, where one of them only considers the *EG* constraint and the other one considers both *EG* and *DG* constraints. Despite that the pattern count increases 31% on average (Column 4) after considering *DG* constraint, on average we can reduce testing time by 18% (Column 7).

In the last experiment, we plot the relationship between EGP test coverage with the number of performed test clock reconfigurations, as shown in Fig. 11. For all benchmark circuits, the trend is that the coverage increases significantly in the first few configurations and quickly saturates to a stable value. To be specific, all benchmarks can achieve more than 90% suspicious FF coverage after 5 test clock configurations. To achieve 100% coverage for the EGP circuits, however, the average number of configurations is 5 for all the benchmarks, and the best case is benchmark des_perf using 2 configurations and the worst case is benchmark wb_conmax using 10 configurations.

7. CONCLUSION

Without developing efficient and effective test methods for timing-speculative circuits, it is impossible to push forward for volume production. The major difficulty in dealing with timing-speculative circuit testing comes from the fact that the timing behavior of such circuits is non-deterministic. In this paper, for the first time, novel test solutions were proposed to address the above problem. With the proposed test flow and the novel test pattern generation techniques, we are able to achieve high fault coverage for timing-speculative circuits without incurring high DfT cost.

8. ACKNOWLEDGEMENT

This work was supported in part by the Hong Kong SAR Research Grants Council under General Research Fund No. CUHK418111 and No. CUHK418812.

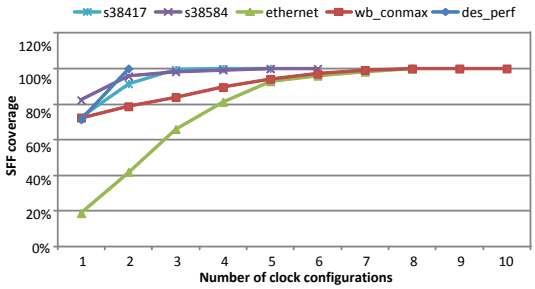


Figure 11: EGP circuit coverage vs. the number of test clock configurations

9. REFERENCES

[1] S. Borkar, *et al*., Parameter variations and impact on circuits and microarchitecture. In *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 338–342, 2003.

[2] D. Frank, R. Puri, and D. Toma, Design and CAD challenges in 45nm CMOS and beyond. In *Proc. ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*, pp. 329–333, 2006.

[3] S. Borkar, Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. In *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[4] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, Opportunities and challenges for better than worst-case design. In *Proc. IEEE/ACM Asia South Pacific Design Automation Conference (ASP-DAC)*, pp. 2–7, 2005.

[5] S. R. Sarangi, *et al*., VARIUS: A model of process variation and resulting timing Errors for microarchitects. In *IEEE Transactions on Semiconductor Manufacturing*, vol. 21, no. 1, pp. 3–13, February 2008.

[6] D. Ernst, *et al*., Razor: a low-power pipeline based on circuit-level timing speculation. In *Proc. IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 7-18, 2003.

[7] K. Bowman, *et al*., Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance. In *IEEE Journal of Solid-State Circuits*, pp. 49-63, 2009.

[8] B. Greskamp, *et al*., Blueshift: Designing processors for timing speculation from the ground up. *Proc. IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 213-224, 2009.

[9] A. B. Kahng, *et al*., Slack redistribution for graceful degradation under voltage overscaling. *Proc. IEEE/ACM Asia South Pacific Design Automation Conference (ASP-DAC)*, pp. 825-831, 2010.

[10] R. Ye, F. Yuan and Q. Xu. Online clock skew tuning for timing speculation. *Proc. ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*, pp. 442–447, 2011.

[11] Y. Liu, *et al*., On logic synthesis for timing speculation. *Proc. ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*, pp. 591–596, 2012.

[12] M. Favalli and C. Metra, Sensing circuit for on-line detection of delay faults. In *IEEE Transactions on VLSI Systems*, pp. 130-133, 1996.

[13] Y. Tsiatouhas *et al*.,A sense amplifer based circuit for concurrent detection of soft and timing errors in CMOS ICs. In *IEEE International On-Line Testing Symposium (IOLTS)*, pp. 12-16, 2003.

[14] A. EI-Maleh and A. AI-Suwaiyan, An efficient test relaxation technique for combinational & full-scan sequential circuits. In *IEEE VLSI Test Symposium (VTS)*, pp. 53-59, 2002.

[15] R.K. Roy, J.H. Patel and J.A. Abraham, Test Compaction for sequential circuits. In *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, pp. 206-267, 1992.

[16] The International Technology Roadmap for Semiconductors: 2011 Edition - Design. In *Semiconductor Industry Association*, (http://public.itrs.net), San Jose, CA, 2011.

[17] E. S. Fetzer, Using adaptive circuits to mitigate process variations in a microprocessor design. In *IEEE Design & Test*, vol. 23, no. 6, pp. 476-483, Nov. - Dec. 2006.

[18] V. Chickermane *et al*., A power-aware test methodology for multi-supply multi-voltage designs. In *International Test Conference*, paper 9.1, 2008.

[19] X. Kavousianos *et al*., Test schedule optimization for multicore SoC: handling dynamic voltage scaling and multiple voltage islands. In *IEEE Transactions on Computer-Aided Design*, vol. 31, no. 11, pp. 1754-1766, Nov. 2012.

[20] N. B. Z. Ali *et al*., Dynamic voltage scaling aware delay fault testing. In *Proc. of 11th European Test Sym.*, pp. 15-20, 2006.

[21] S. Khursheed *et al*., Gate-sizing-based single test for bridging defects in multivoltage designs. In *IEEE Transactions on Computer-Aided Design*, vol. 27, no. 2, pp. 1117-1127, June 2010.