Dual Use Circuitry for Early Failure Warning and Test

Alexander Coyle*, Hui Jiang†, Jennifer Dworak†, Theodore Manikas*, and Kundan Nepal‡
* Department of Computer Science, Southern Methodist University, Dallas Texas, 75205
† Department of Electrical and Computer Engineering, Southern Methodist University, Dallas Texas, 75205
‡Electrical and Computer Engineering Department, University of St. Thomas, Saint Paul, Minnesota 55105
Email: *acoyle@smu.edu, †huij@smu.edu, †jdworak@smu.edu, *manikas@lyle.smu.edu, ‡kundan.nepal@stthomas.edu

***Abstract*—Incorrect circuit timing often leads to errors in the field, including silent data corruption. Once a circuit violates timing, recovery can be difficult even when the violation is detected. Canary flip-flops have previously been proposed to identify when the desired slack of a path has first been violated in functional mode even before failure occurs. We show how such flip-flops can be combined in a MISR to also provide enhanced coverage during manufacturing test and scan-based field test.**

***Index Terms*—Canary Flip-flops, Design for Test, Fault Coverage, Silent Data Corruption (SDC)**

I. Introduction

DFT (Design for Testability) circuitry is often added to designs to enhance the controllability and observability of a circuit during test. Common examples of such circuitry include scan flip-flops, scan chains, on-chip decompressors, and MISRs (multiple input signature registers), among others.

Although adding circuit logic to implement scan costs area, power, and routing, the overhead cost is overshadowed by significant benefits in fault coverage and the ability to perform combinational, as opposed to sequential, ATPG (automatic test pattern generation). Similarly, the overhead cost of on-chip decompressors is offset by the great benefits in reduced test data volume that can be achieved. Circuits used to implement memory and logic BIST (built-in-self test) are also commonly included in SoCs (System on Chips) to enable automatic testing of logic and memory without requiring the use of an external tester [1], [2]. Such circuitry is especially useful when tests must be performed at regular intervals and/or upon power up or power down in the field to detect new, developing problems due to aging and latent defects. Test-per-scan, used in LBIST, is one approach that shifts the entire pseudo-random pattern into a circuit-under-test using scan chains before the test is applied. An alternate approach, test-per-clock feeds the outputs of the test pattern generator to the circuit under test on each clock and collects the results from the outputs of the circuit under test with a signature analyzer [3], [4].

While in-field tests may be used for detecting problems that have already arisen, they are not used for detecting erroneous operation of a device in functional mode. For example, it is possible for environmental factors, such as temperature, to interact with increased delays due to aging to cause timing failures in functional mode that may not have been detected during an initial in-field test session (e.g. when the circuit temperature was lower). Other approaches are needed to detect, and sometimes correct errors that occur during normal operation. Some previously proposed approaches included triple modular redundancy [5], RAZOR [6], [7], re-executing code in alternative threads [8], parity [9], and hardware assertions [10], among others.

However, while detecting errors is important, being able to predict that errors are likely to occur soon is potentially even more useful. Canary flip-flops [11]–[13] are one approach that has previously been proposed to identify when the delay of a circuit path has exceeded a predetermined slack. If that slack is well-designed, and if the delay increase is relatively gradual, then a canary flip-flop may be able to identify that a future timing failure may happen soon, and appropriate responses, such as increasing the clock period or increasing the voltage, may be employed to reduce the likelihood of such a failure before it occurs.

In the past, the use of MISRs composed of flip-flops that can capture data during scan shift to increase coverage without requiring an increase in test time have been proposed. For example, in [14] the authors proposed the use of a MISR for the detection of static cell-aware faults during scan shift that would otherwise be missed by a stuck-at test set. They showed that a high percentage of the otherwise missed cell-aware faults could be detected fortuitously during scan shift. In subsequent work, the authors of [15], [16] showed that the number of detections of the otherwise least detected stuck-at faults could also be significantly increased if a MISR were used to detect defects during scan shift. These results gave further evidence for the usefulness of such a DFT structure for fortuitous detection of defects.

However, while the MISR structure proposed in [14] and further analyzed in [16] provides significant benefits during test—including potentially during field test—the added flip-flops remain unused in the functional mode of operation. Thus, in this paper, we consider an approach that allows those added flip-flops to be used as canary flip-flops during functional mode while serving as MISR flip-flops during scan shift. Such an approach may help to reduce silent data corruption (SDC) by increasing the effectiveness of any scan-based field tests that are applied while also helping to prevent some timing related failures in functional mode.

This work was supported by NSF CCF1812777 and CCF1814928.

Of course, doubling the number of flip-flops in the circuit by including all of them in the set of flip-flops used to implement a canary structure and/or whose data will be fed into the MISR is much too expensive. As a result, we will show how the regular functional flip-flops can be selected to provide good coverage of critical paths for detection of delays by the canary structure while overall spatial coverage of the circuit (and thus coverage of the corresponding stuck-at faults) can be enhanced with the MISR. In addition, lumped delays that develop along paths that were predicted to be non-critical but that become critical in a particular chip instance due to defects and/or significant process variations are more likely to be detected with this selection of canary flip-flops as well.

II. DFT Architecture

The proposed DFT and fault tolerance architecture aims to combine canary flip-flops with a MISR that be used to detect defects during scan shift. In this section, a brief overview of the two types of components and how they can be combined in a dual use fashion is presented.

A. Canary Flip-Flops

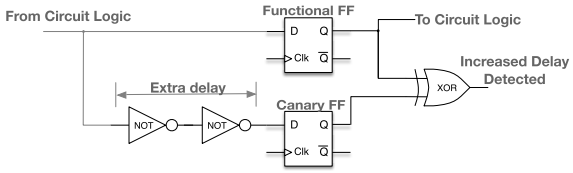


Fig. 1. Canary flip-flop design previously proposed to detect increasing delay or increase performance

Canary flip-flops, such as the flip-flop arrangement shown in Figure 1, have previously been proposed to identify that increasing delay is close to causing a timing violation. In the figure, the functional flip-flop captures the data that will be used by the circuit on the next clock cycle. The canary flip-flop captures data that travels through some extra delay elements (in this case two inverters). Ideally, it will always capture the same value as the functional flip-flop as long as the critical path delay for the path arriving at the functional flip-flop has a slack greater than the added delay element.

This is accomplished because the delay to the canary flip-flop is designed to be greater than the delay to the functional flip-flop. Unless there is a defect in the functional flip-flop itself or in the branch directly leading to the functional flip-flop, gradually increasing excessive delay should cause a failure in the canary flip-flop before the functional flip-flop experiences a failure. When the delay gets too large to meet the timing needed for correct data arrival at the canary flip-flop, but while the delay to the functional flip-flop is still acceptable, the XOR gate used to indicate that a timing failure may be imminent will be set equal to a logic one.

Canary flip-flops have previously been used for performance enhancement. In particular, they can allow the voltage applied to the circuit to be lowered (reducing power) or the clock frequency increased (increasing performance) without causing timing issues. They can also be used to detect imminent failures due to increases in temperature or due to aging [17]–[20].

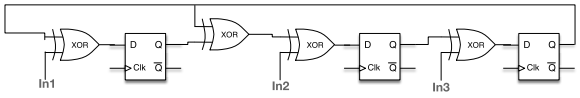


Fig. 2. 3-bit Multiple Input Signature Register (MISR)

B. Multiple Input Signature Registers

Another type of structure that is used for test and fault tolerance is a MISR. A 3-bit MISR is shown in Figure 2. Like a linear feedback shift register (LFSR), a MISR consists of a shift register composed of D flip-flops, where the output of the shift register is fed back around to the input. Selected flip-flop inputs are also connected to the feedback signal through XOR gates, where the tap points for the flip-flops connected to the feedback are selected based upon well-known primitive polynomials [21]. The primitive polynomial used is determined by the number of bits (i.e. flip-flops) in the MISR. A MISR contains additional XOR gates that allow external input data to be used to determine the values captured in the flip-flops on the next clock edge. Over multiple clock cycles, the input values coming into the MISR combine with the current MISR values to create a distinctive signature characteristic of the input value history. Ideally, a change in the input values coming into the MISR will be reflected in a different signature being stored in the MISR.

A MISR is valuable during test because the outputs of a circuit can be fed into the MISR's parallel inputs. For example, it is possible to feed the outputs of multiple scan chains into a MISR during BIST so that a single signature can be collected for the entire scan out sequence. This signature can then be compared to a good circuit signature to determine if the circuit has passed the test.

C. Using a MISR to Detect Defects during Scan Shift

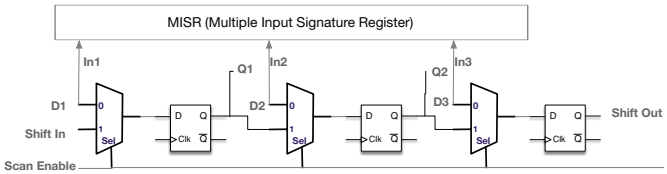


Fig. 3. Scan Chain with MISR used to detect defects during scan shift

In [14], the authors proposed connecting a MISR not only to the scan out signals of the scan chains, but to the data inputs of individual scan cells, as shown in Figure 3. The advantage of this arrangement is that it allows for the capture of data from the circuit during scan shift without overwriting the contents of the scan chain that are being shifted in or shifted out. As a result, the structure was able to fortuitously detect defects during scan shift without requiring any change in the ATPG algorithm or pattern set. The approach also can be implemented in the presence of an on-chip decompressor.

The power of their approach lies in the fact that the number of test patterns that can be applied during test can increase by a factor corresponding to the length of the scan chain because each "intermediate" state during scan shift becomes

a test pattern. As long as the changes in the circuit that arise from the "intermediate" patterns are propagated to the MISR, defective behavior may be caught.

The authors of [14] also showed that a significant fraction of the static cell-aware faults that would otherwise be missed by a stuck-at test set can be fortuitously detected during scan shift. 95% of the static cell-aware faults that would have otherwise been missed were caught fortuitously in an industrial circuit —reducing the number of "top-off" patterns that had to be added for cell-aware fault detection.

In [16], the authors explored the number of times that stuck-at faults are detected by the "intermediate" patterns. As the number of stuck-at fault detections increases, the probability of detecting whatever untargeted faults or unmodeled defects may be present at that site increases as well. However, the probability of detecting an as yet undetected defect with an additional stuck-at fault detection decreases exponentially as the number of previous detections of that stuck-at fault increases [22]. Thus, it is most critical to detect the faults that have been detected the least number of times again. In [16] the authors showed that flip-flops selected for stuck-at fault detections with a greedy algorithm could lead to high percentages of these low-detected stuck-at faults having additional detections.

D. Combining Canary Flip-Flops with the Scan Shift MISR to Create a Dual Use DFT Structure

Both the canary flip-flop approach and the previously proposed scan shift MISR require the use of additional flip-flops connected to the same circuit signals as the functional flip-flops (possibly through some additional gates). However, while the canary flip-flops may achieve their most important use in functional mode, the MISR is used primarily in test mode. By creating a single structure that performs both functions, we can amortize the cost of the additional flip-flops over the entire lifetime of the circuit—including functional mode and test. One possible implementation of such a structure is shown in Figure 4.

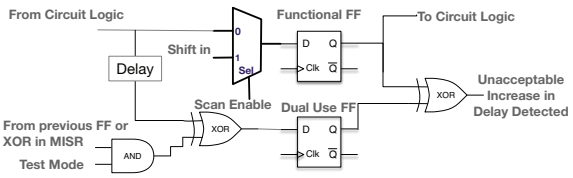


Fig. 4. Dual Use Structure in which Flip-Flops are shared between Canary FF use in Functional Mode and MISR during Scan Shift

In this figure, a Mux-D Scan Flip-Flop is shown as a functional flip-flop connected to a Multiplexer. The select line of the multiplexer is connected to the "Scan Enable" signal. The dual use flip-flop is shown below the functional flip-flop. This flip-flop will serve as a MISR flip-flop when "Test Mode" is equal to 1. It will serve as a Canary flip-flop when "Test Mode" is equal to 0.

The Canary Flip-Flop circuitry consists of the dual use flip-flop, the delay element (whose desired delay will be dependent upon the relative delay to both flip-flops in the structure and the desired slack) and the XOR gate that compares the values captured in the functional flip-flop and in the dual use flip-flop.

The MISR circuitry in this figure consists of the Dual Use Flip-Flop and the XOR gate feeding into the D input of the Dual Use Flip-Flop. The full MISR will also contain additional XORs for the taps in the feedback.

The AND gate on the left side of the figure is used to determine whether the dual-use circuitry is in Canary Flip-Flop (functional) or MISR (test) mode. When "Test Mode" is equal to a logic 1, the value of the previous flip-flop or XOR gate in the MISR will be XOR'ed with the data coming from the circuit logic—allowing the circuitry to behave as a MISR component. However, when "Test Mode" is equal to 0, the output of the AND gate will be 0, and the subsequent XOR will merely add delay to the Canary Flip-Flop path. The value captured in the Dual Use Flip-Flop should be the same as the value captured in the Functional Flip-Flop provided that the two paths are sufficiently fast. Thus, when "Test Mode" is equal to 0, the circuitry will operate in Canary Mode.

Note that because the Dual-Use Flip-Flop does not need the multiplexer required by the Scan Functional Flip-Flop, the relative delay added by the XOR in the Canary path is minimized. Thus, the desired slack difference between the canary and functional paths can be kept very small.

III. Flip-Flop Selection

While both canary flip-flops and the MISR used to capture data during scan shift are useful, they are unacceptably costly if every functional flip-flop is paired with a dual use flip-flop. In [14] and [16], the authors selected a subset of the functional flip-flops to have corresponding flip-flops inserted in the MISR. This selection was based on maximizing the detection of otherwise missed cell-aware faults or maximizing the additional detections of stuck-at faults. However, there is no guarantee that these flip-flops are the correct subset to use when excessive delay detection with canary flip-flops is part of the goal. Thus, in this section, we explore different approaches for identifying a set of flip-flops that could be used effectively as both canary flip-flops and as parts of a MISR.

Our analysis studied the effects of different approaches to flip-flop selection, analyzing the relative benefit of privileging slack length, fault detection, or a balance between the two. The nature of the circuit, its purpose, and the acceptable overhead will determine the weighing of slack and fault detection for future implementations of this approach.

A. MISR Flip-Flop Selection Using Stuck-at Fault Cones

In [14], to detect faults during scan shift, flip-flops were selected based upon maximizing the number of desired fault detections with the intermediate patterns that arise with a specific test set. Each intermediate pattern corresponded to a combination of the pattern being shifted in and the circuit responses to the last test pattern being shifted out. The number of faults detected at each scan flip-flop was recorded, and a greedy selection approach was used to maximize the detections up to a predefined flip-flop overhead. This approach produced

good results, but the simulation of the intermediate patterns to each flip-flop was resource-intensive, and the results were optimized for a specific pattern set.

As an alternative approach, the authors of [16] subsequently used fault cone analysis to identify which faults were in the fault cone of each flip-flop. Such faults were assumed to be potentially detectable at a given flip-flop, but the likelihood of that detection was not evaluated. A greedy approach was then used to select the flip-flops to be included in the MISR.

In particular, the flip-flops were initially ordered from largest to smallest fault cone, and the flip-flop with the largest number of targeted faults (in this case stuck-at faults) in its fault cone was selected for inclusion in the MISR. At that point, the faults that were potentially detectable by the selected flip-flop were removed from consideration in the other fault cones, and the flip-flops were resorted based upon their new fault cone counts. This process continued until the flip-flop overhead reached the user-defined maximum. We will use this greedy fault cone-based selection approach **(Algorithm A)** to compare to two other methods in this paper.

B. Canary Flip-Flop Selection Based on Critical Paths

In contrast, our first flip-flop selection approach that targeted canary behavior, here called **Algorithm B**, focused entirely on selecting the flip-flops based upon slack. Specifically, the slack of the longest path terminating at each flip-flop was identified, and the flip-flops were ordered in terms of increasing slack. The flip-flops with the smallest slack were then selected up to a pre-selected flip-flop overhead.

The assumption here is that, because we are proposing to use the canary flip-flops as an early warning of unacceptable path delay increases, the paths that start out with the smallest slack are likely to see an unacceptable increase first.

However, the flip-flops, ranked by least amount of slack, had critical paths that had significant overlap in their gates in our experiments. This seemed to indicate that we were not getting good spatial coverage of the circuit overall. This is potentially problematic if we want to use those same flip-flops for a MISR to detect defects during scan shift. However, it is also potentially problematic even for detecting increasing delays as a warning of failures.

Clearly, the capture of the longest path(s) is important because, if the timing issue is the result of an even aging of the chip, it will likely be detectable along the longest path(s) first. However, the actual path lengths for a particular chip may not perfectly match the lengths predicted by timing analysis tools. For example, process variations may make different parts of a chip slower or faster than expected. Aging may not be uniform across the chip, and latent defects could cause an unexpectedly large delay in a part of the chip not covered by the predicted longest paths. To address this issue, a new flip-flop selection algorithm was devised specifically for the proposed dual use approach.

C. Dual Purpose Flip-Flop Selection Based on Fault Cones and Critical Paths — Greedy Slack

This algorithm targets having a diverse set of gates along the paths ending at selected flip-flops to not only increase the fault coverage for the flip-flops in the MISR during test, but also to improve the benefit derived in functional mode by having the selected canaries cover a larger area of the chip. We will refer to this Greedy Slack algorithm as **Algorithm C**.

Algorithm C: Greedy Slack

1: Run timing analysis on circuit to generate 10 longest paths that terminate at each of the circuit's flip-flops
2: Add {Flip-flop, Slack amount, and Gates in Path} to Path Array
3: Pop the first State (longest path) off of Path Array and add to a Results Array
4: Add the path's gates to a Visited Gates array
5: Remove the other paths terminating at the selected flip-flop from the Path Array
6: Score the remaining flip-flops in the Path Array by adding one point for each unvisited gate along the path
7: Order the path array by score to find the path that visits the most new gates
8: If two paths visit the same number of new gates, the longer path will be chosen
9: Pull this State off the Path Array and add to the Results Array
10: If the number of flip-flops does not equal the maximum number, go to step 4

To maximize our coverage, we ran static timing analysis on our circuits, but this time we generated the 10 longest paths that terminated at each of the circuit's flip-flops. These paths were processed by the *Greedy Slack Algorithm* shown here. This greedy algorithm assessed not only the slack length of each path but also the number of previously unvisited gates along that path.

Beginning by choosing the longest path from the timing report, the algorithm recorded each gate along this path. Each subsequent path in the report was scored based on how many new gates it traverses, a point for each. The path with the highest score was selected, its gates were recorded, and the paths were scored again, each time against the expanded list of visited gates. We select the top flip-flops using this combination of coverage and slack, which we called the Greedy Slack flip-flops, up to a predefined flip-flop overhead.

The steps of the algorithm can be illustrated with a brief example. Figure 5 shows the path array as it arrives from the timing analysis, with the 10 paths to each flip flop arranged in ascending order of slack amount. The greedy slack algorithm takes the longest path (shortest slack) from the top of the array, in this case **FF1**. Adding the flip-flop to the selection list (the selected FFs array), the algorithm takes the gates visited on this path to FF1 and adds them to an array of visited gates, here {A,B,C,D}. We remove the other paths that share FF1 as an endpoint.

After passing the first flip-flop to the results array, the algorithm scores each of the paths in the array according to how many new gates are visited on the way to the endpoint (Part 2). The path ending at FF2 with slack 2.4, while it is longer than FF3, visits only one new gate and thus receives

Part 1: Initial Visited Gates={ }
Selected FFs ={ }

Flip-Flop Endpoint	Score	Slack	Path Gates
FF1	-	2.0	A, B, C, D
FF1	-	2.2	A, B, C, X
FF2	-	2.4	A, B, C, E
FF3	-	2.5	B, E, F, G
FF2	-	2.7	C, D, H, I
:	:	:	:

Part 2: Score Visited Gates={**A, B,C, D** }
Selected FFs ={**FF1** }

Flip-Flop Endpoint	Score	Slack	Path Gates
FF2	1	2.4	A, B, C, E
FF3	3	2.5	B, E, F, G
FF2	2	2.7	C, D, H, I
:	:	:	:

Part 3: Sort Visited Gates={A, B,C, D }
Selected FFs ={FF1 }

Flip-Flop Endpoint	Score	Slack	Path Gates
FF3	3	2.5	B, E ,F, G
FF2	2	2.7	C, D, H, I
FF2	1	2.4	A, B, C, E
:	:	:	:

Part 4: Rescore Visited Gates={A, B,C, D, **E, F, G** }
Selected FFs ={FF1, **FF3** }

Flip-Flop Endpoint	Score	Slack	Path Gates
FF2	2	2.7	C, D, H, I
FF2	0	2.4	A, B, C, E
:	:	:	:

Fig. 5. Example of Algorithm C (Greedy Slack)

a score of 1. FF3 visits 3 new gates, so when the Path Array is ordered by score (Part 3), **FF3** is pulled off of the Path Array and its gates are added to the Visited Gates array, now {A,B,C,D,E,F,G}. Part 4 shows how the remaining paths in the Path Array are re-scored according to the expanded array of visited gates. Although it had a score of 1 in the previous iteration of the algorithm's loop, the path to FF2 along gates A,B,C,E now has a score of 0 because these gates are already contained in the Visited Gates array. FF2 is chosen for the selected flip-flop list, the gates {H,I} are added to the Visited Gates, and the other paths to FF2 are removed from the Path Array. The process repeats, updating the scores of the paths against the expanding Visited Gates array, until the desired number of flip-flops has been collected in the Results Array.

As the Greedy Slack algorithm begins with the longest path provided by the static timing analysis tool, the most critical path identified in the timing report will always be included in the selected flip-flops. The capture of the longest path or paths is important because, if the developing timing issue is the result of an even aging of the chip, it will likely be detectable along the one of the longest paths first. At the same time, we may not need to detect it along the absolute longest path as long as enough extra delay is present in the canary flip-flop structure to account for the fact that the first canary flip-flop to fail may not necessarily correspond to the first functional flip-flop whose arrival time begins to enter its slack region. To address this, different canary flip-flops could be created with their different delay elements designed to account for the different expected slacks along their respective critical paths.

IV. Experimental Setup

Our experiments were carried out on five circuits obtained from opencores.org. Some characteristics of these circuits are shown in Table I.

TABLE I
Characteristics of Circuits

Circuit	# Flip -Flops	# Stuck-at Faults	# ATPG Patterns	# Intermediate Patterns
Quadratic	120	8166	36	1044
Des56	193	13788	119	2856
Fm_rec	501	19888	406	10962
Colorconv	584	38518	98	3136
Fpu	5231	297358	538	17216

Previous results on cell-aware and stuck-at fault detection during scan shift indicate that circuit size is not a reliable indicator of the applicability of the approach to a circuit. Instead, structural and functional characteristics of the circuit are considerably more important, and these circuits were selected because they do a good job of spanning some of those characteristics.

In particular, the overlap of the fault cones is very important. For example, Des56 has many gates that are in the fault cones of only one or a few flip-flops with limited overlap. As a result, the percentage of flip-flops that need to be included in a MISR to get very high additional fault coverage is large. In contrast, Fm_rec has a small number of flip-flops that include a very high percentage of the faults in their fault cones. As a result, a relatively small percentage of flip-flops need to be included in a MISR to get high fault coverage for Fm_rec.

Another important characteristic of the circuit is the inherent observability of the circuit sites. Although, Des56 has fault cones with smaller overlap, the circuit sites in that fault cone generally appear to be highly observable at the corresponding flip-flop. This helps to improve coverage provided that the corresponding flip-flop is selected for inclusion. In contrast, Fpu appears to have some circuit sites that are significantly less observable at the selected flip-flops. This reduces the coverage that can be achieved. While this observability may be related to circuit depth, it is also related to functionality, and it is not automatically related to circuit size.

These circuits were synthesized using a NanGate 45nm library. Using a commercial static timing analysis tool, we gathered data on the path to each flip-flop in the circuit with the least amount of slack. With the greatest distance to travel within a clock cycle, these critical paths are the most likely routes along which one would first see timing errors. A commercial software tool was also used to obtain fault cone analysis data for flip flop selection.

In each case, a maximum flip-flop overhead of 8% was set as the predetermined limit. This overhead was selected based on the work in [23], which proposed the replacement of 7% and 21% of flip-flops for two circuits, respectively. We chose 8% to remain closer to the lower replacement rate found in the paper in an attempt to maximize the timing and fault-detection efficacy with a reasonably small overhead.

The Greedy Flip-flop selection based only on fault cones as described in Section III-A used all stuck-at faults as the target fault model. The flip-flop selection methods described in Section III-B and III-C were based upon the timing analysis report from a static timing analysis tool.

To reveal the effect of these flip-flops on the detection of low-detected stuck-at faults, fault simulation was done with intermediate patterns. First, a commercial software tool was used to generate stuck-at ATPG patterns. Then, the scan shift intermediate patterns were identified. Each intermediate pattern is a combination of the current pattern being shifted in and the results from the previous pattern being shifted out. The number of stuck-at faults, the number of stuck-at ATPG patterns, and the number of intermediate patterns for each circuit are shown in Table I.

To identify which fault would be considered to have a "low" number of detections if a MISR were not used, first all the stuck-at faults were simulated with the stuck-at ATPG patterns in normal mode (without MISR). The number of times each stuck-at fault was detected by the ATPG test set was then obtained. We define low-detected faults as being stuck-at faults whose original detection count was 1 to 15. The number 15 was selected because the authors of [24] showed that, for their experiments, detecting all faults at least 15 times at-speed led to no missed defect detections.

Then, the intermediate patterns were simulated to determine how many extra detections could potentially be obtained with a MISR for these low-detected faults. (We did not determine how much aliasing would have occurred in the MISR, but in our previous MISR work, aliasing was reasonably small). Note that these simulations were performed with all the flip-flops that were not selected to be included in the MISR masked.

V. Results

A. Detection of the Otherwise Least Detected Stuck-at Faults During Scan Shift

Figure 6 shows what percentage of the low-detected stuck-at faults (i.e. those that originally had 1 to 15 detections by the stuck-at test set) could achieve additional detections in the MISR during scan shift, where the flip-flops included in the MISR corresponded to those selected by one of the three previously described algorithms: A, B, and C. On the x-axis, each cluster of bars shows the results for the three different groups of flip-flops of each circuit, chosen by the three algorithms. The legend indicates the algorithm that corresponds to each bar of each cluster. The y-axis, by percentage, shows the percent of low-detected stuck-at faults that gained extra detections with the MISR and intermediate patterns (i.e. detections during scan chain shifting).

In Figure 6, we can see that Algorithm A was the best choice for maximizing the detection of the low-detected stuck-at faults in three of the five circuits. Choosing flip-flops solely on the criterion of the longest path, as Algorithm B did, entailed a significant drop in the detection of low-detected stuck-at faults for Quadratic, Des56, and Fpu. Our Greedy

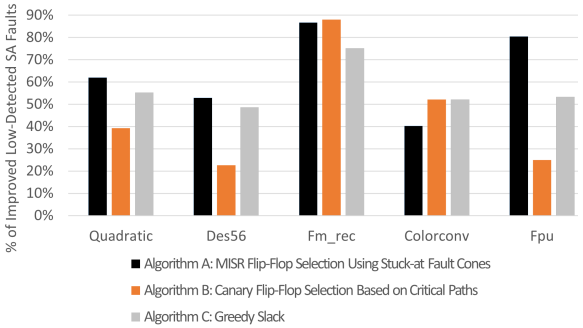


Fig. 6. Percent of low-detected stuck-at faults that gained extra detections with flip-flops chosen by the three Algorithms A, B, and C

Slack algorithm, proposed here as Algorithm C, worked effectively as a compromise between the two: in three of the five circuits, it detected only a somewhat lower percentage of the otherwise low-detected stuck-at faults when compared to Algorithm A. Furthermore, even in the case of FPU, which had a more significant drop versus Algorithm A, over 50% of the otherwise least-detected stuck-at faults still obtained additional detections during scan shift. Finally, in the case of Colorconv, Algorithm C actually obtained better fault detection results than Algorithm A.

We were surprised to find that Algorithm B had very good fault coverage for two circuits, Fm_rec and Colorconv. We believe that this may be due to a correspondence between why a stuck-at fault is hard to detect and its location relative to a critical path. In particular, if the least detected stuck-at faults were hard to detect because they were hard to observe along long paths, then there would naturally be overlap in the flip-flops required for both purposes. In contrast, because Algorithms A and C looked at fault cones and gates without regard to the difficulty of detection/observation, they didn't focus on the least-detected faults.

B. Critical Path Length of Paths Ending at Selected Flip-Flops

TABLE II
Z-Score of Mean Slack Length for Selected Paths

Circuit	Algorithm A	Algorithm B	Algorithm C
Quadratic	-0.4	-2.2	-1.3
Des56	-1.2	-1.8	-1.2
Fm_rec	-1.4	-2.7	-1.9
Colorconv	0.1	-1.7	-0.8
Fpu	-0.7	-2.8	-0.8

Table II provides summary information regarding the slacks of the paths ending at the flip-flops selected by each of the three algorithms. Specifically, it shows the Z-score of the mean slack length for the paths selected by each algorithm for each circuit. The Z-score, also known as the standard score, is a statistical measure that indicates how many standard deviations a data point is from the mean of a population. A positive Z-score indicates that the observation is above the mean, while a negative Z-score indicates that the observation is below the mean. A Z-score of 0 indicates that the observation is equal to the mean. For the three different algorithms, we found the mean slack length for each of the paths selected by the algorithms and calculated the Z-score of that mean relative to the total distribution of slack lengths.

Note that because a larger slack indicates that the path is less critical, the slacks of the most critical paths in a circuit will be below the mean slack for that circuit, and their Z-scores will be negative. A more negative Z-score will indicate that the paths are farther from the mean and therefore more critical.

The slacks for the paths selected by Algorithm B were the smallest of the three algorithms we tested, and its Z-scores were the most negative. Each path was technically unique because it terminated at a different flip-flop; however, the top paths, ranked by least amount of slack, had significant overlap, traversing many of the same gates. As already stated, this can adversely affect the spatial coverage achieved by the canary flip-flops and MISR.

The Z-scores of the mean slack lengths of Algorithm A tended to be closest to the overall mean of 0 (i.e. the Z-scores were the least negative), when compared to the other two algorithms. This is unsurprising given that these flip-flops were not chosen with consideration of their effectiveness in a canary structure for catching timing problems.

Algorithm C, the Greedy Slack algorithm, as with the detection of stuck-at faults, tended to work as a compromise between the two algorithms, choosing many of the longest paths in the circuits while still trying to get good spatial coverage. This is indicated by the fact that the Z-scores from the paths selected by Algorithm C generally lay between Algorithm A's and Algorithm B's.

However, while the summary statistics provided by the Z-scores are useful, even better intuition can be obtained by looking at the distribution of the slacks at each flip-flop selected by one of the three algorithms.

In Figure 7, we see a line graph of the smallest slack per path to each of the selected flip-flops, with each line representing a different algorithm. The x-axis corresponds to the flip-flop/path ID sorted from lowest to highest slack. The y-axis corresponds to the slack of each path. (Note that we didn't attempt to scale the slack for a fast clock for these circuits. As a result, the slack numbers are large. However, we are not interested in the raw numbers, but in the relative slacks of the paths. Changing the clock periods would merely shift these curves up and down the y-axis.)

In general, the paths chosen by Algorithm C have slack lengths between those of Algorithms A and B, Except in the case of Des56, the smallest slack paths of Algorithm C overlap well with those of Algorithm B, the latter choosing flip-flops solely based on the smallest slack, or longest path. This means that in most cases, Algorithm C, our Greedy Slack algorithm, shares not only the absolute longest path with the slack-only Algorithm B, but it also selects flip-flops corresponding to many of the other longest paths of the circuit. This overlap is important because the timing analysis tool's estimate of path length may not perfectly match the results of physical tests of the circuit once it is manufactured, as has been shown in [25]. Thus, it is good to include multiple long paths, any one of which could actually be the longest in a particular instance of the circuit.

The results here on Fpu are especially good for all three algorithms. There is a big discrepancy in the path lengths between the longest paths and the majority of the paths. All three algorithms consistently selected the flip-flops at the ends of the very longest paths. Also, note that because obtaining flip-flops for Algorithm B is so simple, it is very easy to compare Algorithm C's path length results to those of Algorithm B's. Thus, for circuits such as Des56, if the flip-flops selected by Algorithm C do not have enough overlap with the longest paths, it is easy to swap out some of the flip-flops selected last by Algorithm C with flip-flops corresponding to the first paths selected by Algorithm B.

VI. Conclusions and Future Work

In this paper, we explored three approaches for selecting dual use flip-flops to detect timing violations via a canary structure and stuck-at faults as part of a MISR. We introduced a Greedy Slack algorithm that is an effective compromise between these two functions, capturing flip-flops at the ends of critical paths that also deliver decent additional fault detection for stuck-at faults. We chose as a maximum overhead approximately 8% of a circuit's total flip-flops, in keeping with previous research by other authors, although the specific overhead tolerance will vary based on the specific circuit and its application. Regardless of the algorithm chosen, our dual-use circuitry combining the MISR and the canary structure has a compelling benefit for the detection of both timing failure and stuck-at faults: even selecting only 8% of the flip-flops while solely considering the longest paths (i.e. Algorithm B) detects 22-88% of the low-detected stuck-at faults additional times during scan shift as shown in Figure 6.

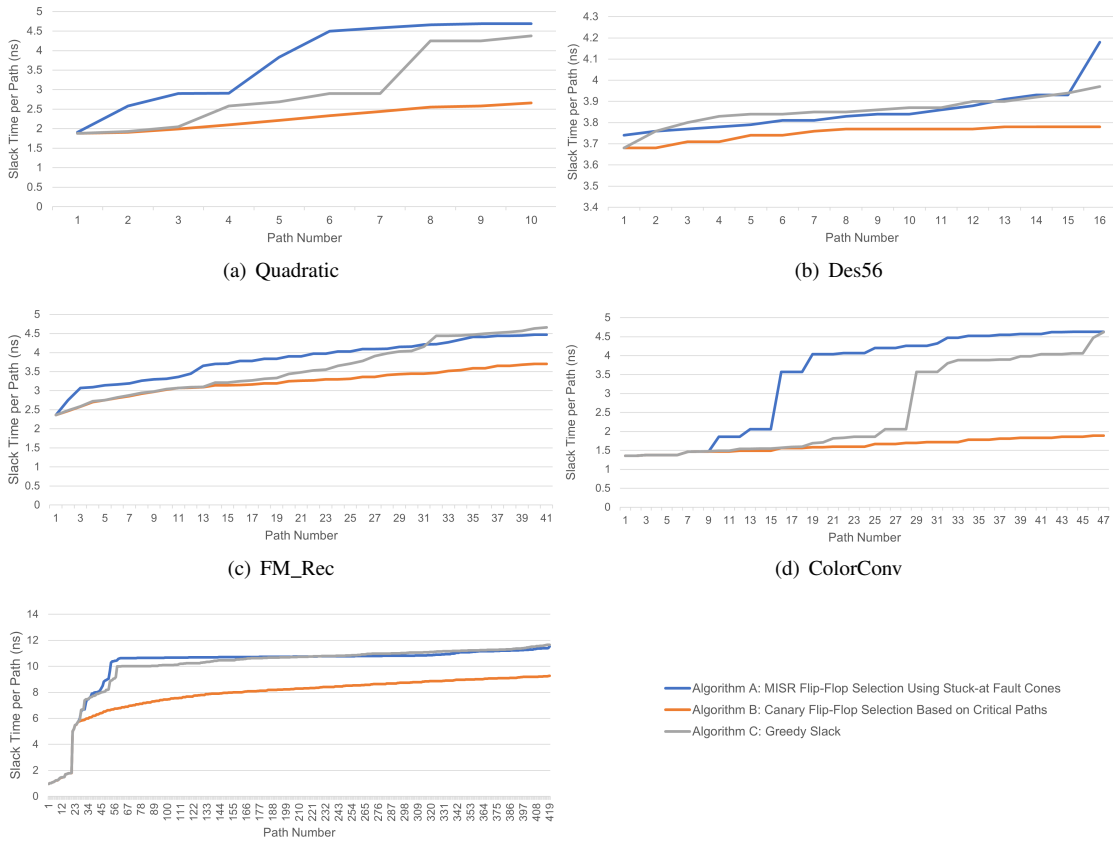
Future work will study the flip-flops' efficacy in achieving other types of fault coverage (e.g. cell-aware and transition faults). We will also investigate refinements to the Greedy Slack algorithm in order to improve its fault detection and area coverage. For example, when targeting stuck-at faults, we can also consider altering the scoring in the algorithm to give more value to gates that correspond to the least detected faults. We can also consider looking at the distribution of the slacks in the circuit overall when selecting or optimizing a particular algorithm. For example, because Fpu contains a small number of paths with a much smaller slack than most of the paths in the circuit, we can make sure that those paths are included in the flip-flop selection.

We can also consider modifications to the dual-purpose structure itself, including considering what benefits may be obtained and costs incurred by designing the delay element in the canary portion of the circuit to better match the expected path lengths so that larger increases in delay even along shorter paths can be detected in real time.

References

[1] N. Mukherjee *et al.*, "Test time and area optimized BIST scheme for automotive ICs," in *Intl. Test Conf. (ITC)*, 2019, pp. 1–10.

[2] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 776–792, 2004.



(a) Quadratic (b) Des56

(c) FM_Rec (d) ColorConv

(e) FPU

Fig. 7. Slack time per path for the paths to flip-flops chosen by the three selection algorithms discussed in the paper, one graph for each of the five circuits.

[3] O. Novak and H. Nosek, "Test-per-clock testing of the circuits with scan," in *7th Intl On-Line Testing Workshop*. IEEE, 2001, pp. 90–92.

[4] M. Bushnell and V. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Springer, 2004, vol. 17.

[5] M. Santhiya *et al.*, "Application of voter insertion algorithm for fault management using triple modular redundancy (TMR) technique," in *Intl. Conf. Intelligent Comm. Tech. and Virtual Mobile Networks (ICICV)*, 2021, pp. 578–583.

[6] D. Enst *et al.*, "Razor: a low-power pipeline based on circuit-level timing speculation," in *IEEE/ACM Intl. Symp on Microarchitecture, 2003. MICRO-36.*, Dec 2003, pp. 7–18.

[7] S. Das *et al.*, "RazorII: In situ error detection and correction for PVT and SER tolerance," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 32–48, Jan 2009.

[8] M. Rahman, B. R. Childers, and S. Cho, "COMeT+: Continuous online memory testing with multi-threading extension," *IEEE Transactions on Computers*, vol. 63, no. 7, pp. 1668–1681, 2014.

[9] R. T. Veetil, R. Sharma, and S. Gundeboyina, "Comprehensive in-field memory self-test and ECC self-checker -minimal hardware solution for FuSa," in *IEEE Intl. Test Conf. India (ITC India)*, 2021, pp. 1–6.

[10] B. S. Chaithanya, G. Gopakumar, D. K. Krishnan, S. K. Rao, and B. C. Oommen, "Assertion based reconfigurable testbenches for efficient verification and better reusability," in *2nd Intl. Conf. on Electronics and Communication Systems (ICECS)*, 2015, pp. 1511–1514.

[11] B. Calhoun and A. Chandrakasan, "Standby voltage scaling for reduced power," in *Proc. Custom Integrated Circ. Conf.*, Sep. 2003, pp. 639–642.

[12] ——, "Standby power reduction using dynamic voltage scaling and canary flip-flop structures," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1504–1511, Sep. 2004.

[13] T. Sato and Y. Kunitake, "A simple flip-flop circuit for typical-case designs for DFM," in *Intl. Symp. on Quality Electronic Design (ISQED)*, March 2007, pp. 539–544.

[14] F. Zhang, D. Hwong, Y. Sun, A. Garcia, S. Alhelaly, G. Shofner, L. Winemberg, and J. Dworak, "Putting wasted clock cycles to use: Enhancing fortuitous cell-aware fault detection with scan shift capture," in *Intl. Test Conf. (ITC)*, 2016, pp. 1–10.

[15] H. Jiang, F. Zhang, Y. Sun, and J. Dworak, "One more time! increasing fault detection with scan shift capture," in *2018 IEEE 27th North Atlantic Test Workshop (NATW)*. IEEE, 2018, pp. 1–7.

[16] H. Jiang, F. Zhang, J. Dworak, K. Nepal, and T. Manikas, "Increased detection of hard-to-detect stuck-at faults during scan shift," *Journal of Electronic Testing: Theory and Applications (JETTA)*, Apr 2023.

[17] A. Muramatsu *et al.*, "12% power reduction by within-functional-block fine-grained adaptive dual supply voltage control in logic circuits with 42v domains," in *Eur. Conf Solid-State Circuits*, Sep. 2011, pp. 191–194.

[18] C. Kim, M. Chung, Y. Cho, M. Konijnenburg, S. Ryu, and J. Kim, "ULP-SRP: Ultra low-power samsung reconfigurable processor for biomedical applications," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 3, sep 2014.

[19] A. Lu, H. He, and J. Hu, "Proximity optimization for adaptive circuit design," in *ACM Intl. Symp on Physical Design (ISPD)*, 2016, p. 91–97.

[20] G. Sai, B. Halak, and M. Zwolinski, "A cost-efficient delay-fault monitor," in *Intl. Symp. Circuits & Systems(ISCAS)*, May 2017, pp. 1–4.

[21] K. N. Devika and R. Bhakthavatchalu, "Programmable MISR modules for logic BIST based VLSI testing," in *Intl. Conf. on Control, Instrumentation, Comm. and Comp. Technologies (ICCICCT)*, 2016, pp. 699–703.

[22] J. Dworak, M. Grimaila, S. Lee, L.-C. Wang, and M. Mercer, "Enhanced DO-RE-ME based defect level prediction using defect site aggregation-MPG-D," in *Intl. Test Conf. (ITC))*, 2000, pp. 930–939.

[23] Y. Kunitake, T. Sato, H. Yasuura, and T. Hayashida, "A selective replacement method for timing-error-predicting flip-flops," in *IEEE Intl. Midwest Symp. on Circuits and Systems (MWSCAS)*, 2011, pp. 1–4.

[24] S. C. Ma, P. Franco, and E. J. McCluskey, "An experimental chip to evaluate test techniques experiment results," in *Intl. Test Conf. (ITC)*. IEEE, 1995, pp. 663–672.

[25] L. Lee, L.-C. Wang, P. Parvathala, and T. M. Mak, "On silicon-based speed path identification," in *IEEE Symp. on VLSI Test (VTS)*, USA, 2005, p. 35–41.