Test Architecture for Fine Grained Capture Power Reduction

Yi Sun*, Hui Jiang*, Lakshmi Ramakrishnan*, Matan Segal*,
Kundan Nepal†, Jennifer Dworak*, Theodore Manikas*, R. Iris Bahar‡
*Southern Methodist University, Dallas, TX, USA
†University of St Thomas, MN, USA
‡Brown University, RI, USA

***Abstract*—Excessive power during in–field testing can cause multiple issues, including invalidation of the test results, overheating, and damage to the circuit. In this paper, we evaluate the reduction of capture power when specific segments of a scan chain can be kept from capturing data subject to values stored in a control register. The proposed approach requires no changes to the Automatic Test Pattern Generation (ATPG), no redesign of the circuitry to match a particular test set, and no additional patterns to maintain fault coverage. We will show that our approach can achieve very high capture power reduction—approaching 100% for multiple patterns.**

***Index Terms*—Design for Testability (DFT), Low Power Test, On-Chip Decompressor**

I. Introduction

Test power is often much higher than functional power [1]. Excess power during test can increase circuit delays, cause IR drop, overheating, and damage or reduce the long term reliability of a chip. Achieving low test power may be especially important when tests need to be applied in the field—such as in an environment where other external factors are already leading to increased temperature. Thus, effective techniques to minimize power expended during test are needed [1].

While there are many proposed approaches to reduce scan shift power, adjacent fill (e.g. [2]) is one of the easiest to implement. Using adjacent fill, don't care values in the pattern to be applied are filled with the value of an adjacent care bit. This approach is advantageous in that it allows faults to be targeted deterministically with the chosen ATPG algorithm in the normal way, and then makes use of whatever don't care values happen to remain in the test pattern.

In contrast, reducing capture power is less straightforward. Even though only a few flip-flops in the scan chain may be used to detect a new, as-of-yet undetected fault, many additional flip-flop values may actually change during capture when the patterns that are used to detect the target fault also create other propagation paths throughout the circuit. The resulting switching can create hot spots within a chip.

The test power problem becomes even harder in the presence of an on-chip decompressor. When such a decompressor is used, many of the don't care values are used to accomplish test data compression instead of reducing power draw. Commercial ATPG tools still allow patterns to be generated in a low power mode, but the overall reduction that can be achieved must also satisfy the needs of the decompressor.

In this paper, we focus on reducing capture power without requiring changes to the initial test set, any increase in test time, or any reduction in fault coverage. Furthermore, we aim to ensure that the DFT hardware is independent of the test generation procedure/test pattern set and thus can be optimized separately. Finally, we aim to make our approach equally valid in the presence of an on-chip decompressor.

To accomplish this, we break a long chain up into smaller scan segments whose ability to capture data during the capture cycle is dependent on a bit in a control register. Simple analysis of the fault dictionary for a pattern set can allow those control bits to be set to enable capture only when the corresponding segment is needed for *additional* fault detections. We will show that with very low overhead, we can achieve high toggling reduction across multiple circuits—even when we start with low-power patterns generated by commercial tools.

II. Previous work

Power reduction during both scan shift and capture haven been a focus of numerous work. Common techniques to reduce scan shift power include adjacent fill [2], test vector reordering [3],combinational logic transition masking during scan shifting [4] and reducing scan shift power in the presence of an on-chip decompressor (e.g. [5], [6]).

Capture power reduction has also been identified as being important, especially in the context of at-speed test. Some researchers have attempted to reduce capture power by intelligently filling don't cares in the test patterns to reduce power draw during the capture cycle (e.g. [7], [8]). In [9], instead of simply lowering power, an attempt is made to create pseudo-functional patterns that are more similar to those seen during normal operation. The authors of [8] apply this concept to an on-chip decompressor, where they allow the shifted out data to be fed back to the beginning of the chain so that the data shifted in can come from either the decompressor or the value that has been shifted out. When applying an X-filling encoding algorithm, [10] and [11] further modified the Embedded Deterministic Test (EDT) decompressor by adding encoded blocks and shadow registers. However, the reduced number of don't cares that result from X-filling approaches can lead to reduced efficiency of the test pattern compression algorithms.

Disabling flip-flops from capturing to reduce capture power has also been proposed. In [12], test points were added

This work was supported by NSF CCF1812777 and CCF1814928.

978-1-7281-0996-1/19/$31.00 ©2019 IEEE

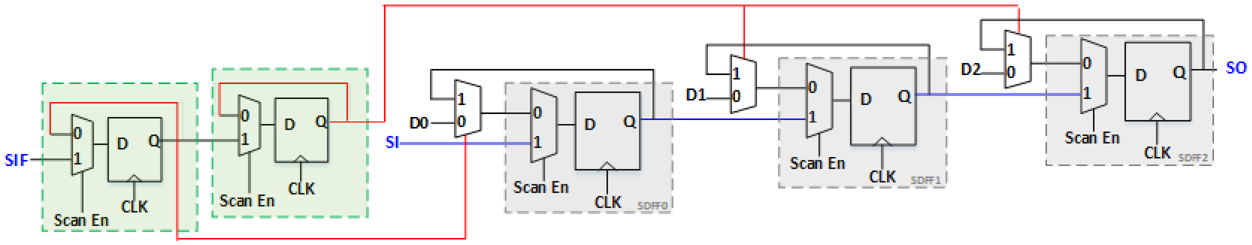

Fig. 1. Schematic design of segmented scan chain.

to independently control clock-gating circuitry and ATPG procedure modified to harness these additional test points—reducing test power at the cost of additional pattern count. The authors of [13] also attempt to reduce capture power through modifying the test generation procedure to reduce the number of flip-flops to which fault effects propagate and then cluster and reorder the flip-flops into chains based upon the optimized test patterns. A limitation of these approaches is that they both require changes to the test generation procedure.

Disabling individual chains or groups of scan chains has been proposed as well (e.g. [14], [15]). In [16]–[18], the scan chains were divided into shorter chain segments that could be shifted independently, allowing fewer simultaneous transitions during scan shift. In [17], only one chain was allowed to capture a test response at given time, further reducing overall power. The work of [17] was extended in [14] to change the test generation procedure to create special test patterns targeted toward their architecture.

Our approach, described next in Section III, builds on this previous work with the goal of significantly reducing capture power. We aim to do so in a way that is i) simple to implement, ii) scalable, and iii) valid even in the presence of an on-chip decompressor. We propose a solution that *does not*:

1) require changes in the test generation procedure carried out by a modern commercial tool;
2) force the DFT insertion process to be dependent on a particular test pattern set;
3) change the original test pattern set, the original fault coverage, test pattern count or test time;
4) require internal changes to an on-chip decompressor.

III. DFT Architecture and Methodology

Our methodology takes a fine-grained approach to preventing particular segments of the chain from capturing when those scan segments are not needed by the current pattern for fault detection. There are multiple ways to accomplish this; however, the simplest way, which does not require any changes to the test generation procedure, and which keeps the original test set intact, is to add a control chain that can be shifted in parallel with the functional scan chains themselves.

For example, consider Figure 1. In this figure, Mux-D scan flip-flops for the functional chain are shaded in grey, and the Mux-D scan flip-flops for the control chain are shaded in green. The functional chain is broken up into two segments: the first segment in the figure is only one bit long while the second segment (to the right) is two bits long.

The value in each control flip-flop determines whether the segment it controls will capture a new value or maintain its old value. There are multiple ways to implement this; however, in Figure 1, we have chosen to insert an additional mux before each scan flip-flop. The select line for this mux is connected to the control flip-flop for that segment. When the control value is equal to a logic 1, the functional flop will hold its original value on the next capture cycle because the value at the output of the flip-flop will be fed back into the input. In contrast, if the value in the control flip-flop is equal to a 0, the functional flip-flop will capture data from the circuit itself (shown as D0, D1, D2, etc.) provided that the scan enable signal, *Scan En* is set to zero, and thus the chain is not in shift mode.

There is some extra delay in the functional path due to the additional mux. If only a few flip-flops are destinations of critical paths, then this could be handled by removing the mux from those flip-flops and not disabling them. Alternatively, other approaches that disable the flip-flop by gating the clock, adding an enable input, etc. could be used instead.

In Figure 1, the control chain is fed directly from an additional scan data input. This allows the control chain to be implemented independently of any on-chip decompressor that may be present, although one could likely generate the patterns with the on-chip decompressor if one configured the control chain appropriately and had access to the on-chip decompressor ATPG tool's algorithm. Furthermore, we expect the control chain may be shorter than a normal scan chain even if it controls segments on multiple chains. Thus, only a single additional input may be needed for control if the number of segments is not too large. It could also be possible to use a de-multiplexer at this input to shift data into multiple short control chains in-turn. This could potentially allow those control chains to be located closer to the segments they control and reduce the delay between the control chain and segments. A detailed analysis of the advantages and disadvantages of such an approach is left to future work.

To implement the proposed approach, it is necessary to a) split each scan chain into multiple segments that can be independently controlled during capture, and b) determine which segments should have capture enabled for each pattern. Intuitively, if a segment only detects faults on the current pattern that have already been detected by a previous pattern, then that segment can be disabled for the current pattern. Once the segments that should be enabled for each pattern are determined, they can be used to generate the control values that will ultimately be shifted into the control chain.

This procedure is outlined next:

1) Insert the scan chains.
2) Use commercial tool to create on-chip decompressor logic.
3) Run ATPG and record the patterns being shifted out of the

on-chip decompressor into the scan chains.

4) Determine which faults are detected by each flip-flop in each pattern.
5) Divide the scan chains into segments of equal (or approximately equal) length.
6) Iterate through *all* patterns.

a) Identify which faults have been detected by a previous pattern and remove those faults from consideration.

b) Consider each segment in turn. If the segment detects new faults, record that fault as newly detected and specify that capture should be enabled for this segment and pattern. Otherwise, disable capture for this segment on this pattern.

c) For the segments that are currently still considered *enabled* on this pattern, make further determination on which segment to keep to maintain fault coverage when faults are detected in multiple segments.

   i) Iterate through *all* of the faults detected by the current pattern in part (b) above.
   - If the current fault is detected by only one segment, we call it a *unique* fault. The segment that detects the *unique* fault is added to the list of mandatory segments for that pattern and all other faults detected by that segment are removed from the fault list.
   - If the current fault is detected by more than one segment, it is added to the *non-unique* fault list. (*Note that when a segment is identified as mandatory, faults that have not been considered yet as well as faults that are in the non-unique fault list will be dropped if they are detected by that mandatory segment.*)

   ii) Iterate through all of the remaining faults in the *non-unique* fault list.
   - Add the first segment that detects this non-unique fault to the enabled segment list. (This list was previously composed only of the *mandatory* segments.)
   - Remove all faults from the non-unique fault list that were also detected by the selected segment.

Once the entire process is complete, we know which segments should be enabled during capture for every pattern to maintain the fault coverage of the original test set. This information can then be encoded as control bit data that will be shifted into the control chain for each pattern.

IV. Results

We ran our approach on circuits from opencores.org (Table I). For each circuit, we wanted to identify which segments should capture data for each test pattern in the low power stuck-at test pattern set. The goal was to see how much *additional* capture power reduction above and beyond that which was already obtained by the commercial tool is possible with this approach. To investigate the effect that different segment lengths may have on the overall power reduction, the process was repeated for different segment lengths.

Circuit	# of scan DFFs	# of scan chains	avg. length of scan chains	# of patterns
des56	382	4	96	114
fm_receiver	521	5	104	388
colorconv	858	9	96	150
fpu_double	5434	5	1087	476

TABLE I
Benchmark circuit properties

Different segment lengths correspond to different amounts of overhead. In particular, each segment requires an additional control flip-flop to be inserted into the control bit shift register (i.e., the green flip-flops in Fig. 1). In this paper, we characterize this overhead as a percentage of the original number of flip-flops in the circuit. Thus, a chain with 10 segments and 100 flip-flops would correspond to an overhead of 10%.

We estimate the capture power by counting the number of flip-flops that toggle during capture for all of the segments for which capture is enabled. We did not simulate the effect of this toggling on the combinational logic. The corresponding toggling in the combinational logic would only add to the overall power expended. We calculate toggle reduction as:

$$\text{Toggle reduction} = \frac{\text{\# toggles in disabled segments}}{\text{\# toggles when all segments capture}} \times 100\%$$

Figure 2 shows the plot of average percent reduction in toggle count as the overhead is increased (i.e., number of segments per chain is increased). Experiments for each circuit were run until the overhead exceeded 10%. For all four circuits we see that increasing the overhead (i.e. using shorter segments and more control bits) reduces the overall toggle count—thereby reducing capture power. For smaller circuits, we see that the toggle reduction at approximately 3% scan chain overhead is about 60% for des56, 60% for fm_receiver and 70% for colorconv. For our largest circuit, fpu_double, we see that with an overhead of approximately 3%, we can achieve almost 90% toggle reduction on average, and increasing the overhead allowance to approximately 10% allows us to achieve an average toggle reduction of almost 93%.

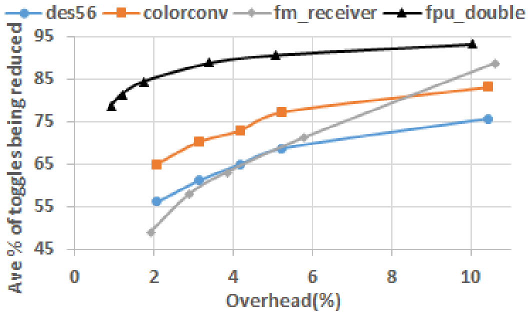

Fig. 2. Average % reduction in toggle count.

Note that increasing the overhead by decreasing the segment length allows much better toggling reduction for all circuits. This makes sense since it is easier to pinpoint a small number of flip-flops that will be used to achieve the desired fault detections when the segments are smaller. In the limit, each flip-flop could be its own segment, although the overheard would be prohibitive. Fortunately, the data shows that high capture reduction can be achieved with much lower overhead.

Also note that the proposed approach is especially effective for the fpu_double circuit because it is not only the largest circuit we studied, but it also contains many don't cares in its test patterns. Many of those test patterns are required to detect just a one or two new hard-to-detect faults. This is different from a circuit such as des56, which is more observable,

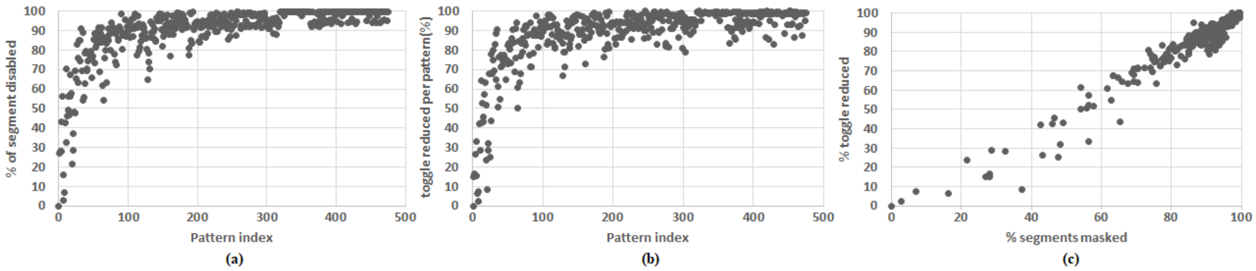

Fig. 3. (a) Percentage of segments disabled, (b) Toggle reduction per pattern; and (c) Correlation between toggle reduction and segments disabled during capture for fpu_double circuit with overhead of 3.4%

requires fewer patterns, and tends to detect more faults per pattern. Thus, des56 is more likely to have more segments enabled on each pattern.

Figure 3 summarizes results for fpu_double for an overhead of 3.4%. The data for other overheads and other circuits follow similar trends. We see that as the pattern index increases, the percentage of segments that can be disabled starts to increase quickly (Figure 3(a)). Our experiments show that for 113 patterns in the overall pattern set (i.e. 23.7% of the patterns) all but *one* segment can be disabled. We found that the average toggle reduction was 90% and the median reduction was 93.3% (Figure 3(b)). In some cases, the toggle reduction is almost 100%. We also see a clear correlation between segments disabled and toggle reduction (Figure 3 (c)).

chain #	1	2	3	4	5
original toggle	145244	202195	189093	155593	313673
removed toggle	131877	183840	171303	143222	259479
% reduction	91	91	91	92	83

TABLE II
TOGGLE COUNT DISTRIBUTION ACROSS THE 5 CHAINS IN FPU_DOUBLE.

In Table I, we saw that fpu_double was partitioned into 5 scan chains. We wanted to determine whether some chains had significantly more toggles than others. The total number of toggles during test for each chain are shown in Table II under the "original toggle" row. The next row shows the number of toggles that can be *removed* from each chain during test across all test patterns. The last row shows the percentage of the original toggling that we were able to remove with the proposed approach. We see that the toggle reduction is distributed fairly evenly across the five chains, although the last chain has less percent reduction and more toggles overall. However, that chain also had more toggles originally. If additional reduction was needed, reordering the chains would be a possibility.

V. Conclusions and Future work

In this paper, we have shown that very high toggling reduction can be achieved during capture cycles even when we start with patterns that have been created by a commercial tool to achieve low power. Note that because we are disabling capture in scan chain segments, any effort made by the ATPG algorithm to reduce shift power for those segments for the pattern shifted into the chain will perform a "double duty". A reduced number of transitions during scan in will lead to a reduced number of transitions during scan out for the bits in those disabled segments because the values will be identical during shift in and shift out.

While our experiments were for stuck-at faults, we expect the results to be just as good, if not better, in the case of transition faults. Transition test sets often detect even fewer faults per pattern than stuck-at test sets—possibly leading to even fewer enabled segments on most patterns. However, some (relatively minor) modifications to the design shown in Figure 1 will be needed if we want to implement Launch-on-Capture and Launch-on-Shift patterns. We will address dynamic patterns in future work.

References

[1] A. Bosio, P. Girard, and A. Virazel, "Test of low power circuits: Issues and industrial practices," in *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec 2016, pp. 524–527.

[2] K. Butler *et. al.*, "Minimizing power consumption in scan testing: pattern generation and dft techniques," in *IEEE Intl. Conf. on Test*, 2004.

[3] J. T. Tudu, E. Larsson, V. Singh, and V. D. Agrawal, "On minimization of peak power for scan circuit during test," in *2009 14th IEEE European Test Symposium*, May 2009, pp. 25–30.

[4] S. Bhunia *et. al.*, "Low-power scan design using first-level supply gating," *IEEE Tran. VLSI Systems*, vol. 13, no. 3, pp. 384–395, 2005.

[5] M. Filipek *et. al.*, "Low power decompressor and prpg with constant value broadcast," in *Asian Test Symposium (ATS)*, Nov 2011, pp. 84–89.

[6] D. Czysz, M. Kassab, X. Lin, G. Mrugalski, J. Rajski, and J. Tyszer, "Low-power scan operation in test compression environment," *IEEE Trans. CAD*, vol. 28, no. 11, pp. 1742–1755, Nov 2009.

[7] X. Wen *et. al.*, "Low-capture-power test generation for scan-based at-speed testing," in *IEEE Intl. Conf. on Test*, Nov 2005.

[8] E. K. Moghaddam *et. al.*, "Low capture power at-speed test in edt environment," in *IEEE Intl. Test Conference*, Nov 2010, pp. 1–10.

[9] Z. Zhang, S. M. Reddy, and I. Pomeranz, "On generating pseudo-functional delay fault tests for scan designs," in *IEEE Intl. Symp.on Defect and Fault Tolerance in VLSI Systems (DFT)*, 2005, pp. 398–405.

[10] W. Wang, J. Kuang, and Z. You, "Achieving low capture and shift power in linear decompressor-based test compression environment," *Microelectronics Journal*, vol. 43, pp. 134–140, Dec. 2012.

[11] G. Mrugalski *et. al.*, "Compression based on deterministic vector clustering of incompatible test cubes," in *Intl. Test Conf. (ITC)*, Nov 2009.

[12] R. Shaikh, P. Wilson, K. Agarwal, H. V. Sanjay, R. Tiwari, K. Lath, and S. Ravi, "At-speed capture power reduction using layout-aware granular clock gate enable controls," in *International Test Conference*, Oct 2014.

[13] L. Lee, C. He, and W. Tseng, "Deterministic ATPG for low capture power testing," in *13th Intl. Workshop on Microprocessor Test and Verification (MTV)*, Dec 2012, pp. 24–29.

[14] Z. You, J. Huang, M. Inoue, J. Kuang, and H. Fujiwara, "Capture in turn scan for reduction of test data volume, test application time and test power," in *IEEE Asian Test Symposium*, Dec 2010, pp. 371–374.

[15] D. Xiang, X. Wen, and L. Wang, "Low-power scan-based built-in self-test based on weighted pseudorandom test pattern generation and reseeding," *IEEE Tran. VLSI Systems*, vol. 25, no. 3, pp. 942–953, 2017.

[16] L. Whetsel, "Adapting scan architectures for low power operation," in *Intl. Test Conference*, Oct 2000, pp. 863–872.

[17] Z. You, T. Iwagaki, M. Inoue, and H. Fujiwara, "A low power deterministic test using scan chain disable technique," *IEICE Transactions on Information and Systems*, vol. E89D, 06 2006.

[18] E. Arvaniti and Y. Tsiatouhas, "Low-power scan testing: A scan chain partitioning and scan hold based technique," *Journal of Electronic Testing*, vol. 30, no. 3, pp. 329–341, Jun 2014.