View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Texas A&M Repository

IMPROVING POWER, PERFORMANCE AND AREA WITH TEST: A CASE STUDY

A Thesis

by

IGNATIUS PRAVEEN LAWRENCE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Weiping Shi
Co-Chair of Committee,	Duncan M. H. Walker
Committee Member,	Jiang Hu
Head of Department,	Miroslav Begovic

August 2018

Major Subject: Computer Engineering

Copyright 2018 Ignatius Praveen Lawrence

ABSTRACT

As more low power devices are needed for applications such as Internet of Things, reducing power and area is becoming more critical. Reducing power consumption and area caused by full scan design-for-test should be considered as a way to help achieve these stricter requirements. This is especially important for designs that use near-threshold technology. In this work, we use partial scan to improve power, performance and area on a graphics processing unit shader block.

We present our non-scan D flip-flop (DFF) selection algorithm that maximizes non-scan DFF count while achieving automatic test pattern generation results close to those of the full scan design. We identify a category of stuck-at faults that are unique to partial scan designs and propose a check to identify and contain them. Our final test coverage of the partial scan design is within 0.1% of the full scan test coverage for both stuck-at and transition delay fault models.

In addition, we present the PPA (power, performance and area) results for both the full scan and partial scan designs. The most noteworthy improvement is seen in the hold total negative slack.

DEDICATION

I dedicate this work to my parents.

ACKNOWLEDGEMENTS

I would like to sincerely thank my academic advisor Dr. Walker for his guidance throughout my education at Texas A&M University. Through his patient and thoughtful mentorship, he helped me make the best decisions regarding my thesis and coursework. I owe my successful academic experience to him.

I would like to thank the design for testability team at Arm® for giving me the opportunity to research on partial scan. My work would not have been possible without the steadfast support from Teresa McLaurin, Richard Slobodnik and Mark Nathan.

Thanks go to my committee chair Dr. Shi and my committee member Dr. Hu for their support throughout my research.

I am also thankful to my colleagues at the EDA lab in computer science and engineering department for being supportive and making my experience a great one.

Finally, thanks to my mother and father for their encouragement.

CONTRIBUTORS AND FUNDING SOURCES

This research was conducted at and funded by Arm®, Austin TX from May through December 2017. The work was managed by Teresa McLaurin, who is a Fellow and the head of the design for testability team at Arm®. The student was mentored by Mark Nathan.

Level1 checks of the partial scan algorithm were arrived at by Teresa McLaurin, Frank Frederick and Mark Nathan. The student developed Level2 checks to make the partial scan algorithm compatible with graphics processing unit (GPU) shader cores. The idea to determine sequential redundancy from full scan automatic test pattern generation (ATPG) results were proposed by Richard Slobodnik.

Power, performance and area (PPA) results shown in Tables 8 through Table 11 were provided by Leo Prakash and Mark Appleton.

This work was supported by my thesis committee consisting of my advisor Prof. Hank Walker of the Department of Computer Science and Engineering, and Prof. Weiping Shi and Prof. Jiang Hu of the Department of Electrical and Computer Engineering.

NOMENCLATURE

SAF	Stuck-at Fault
TDF	Transition Delay Fault
DFF	D Flip-Flop
SDFF	Scan D Flip-Flop
PPA	Power, Performance and Area
TC	Test Coverage
PC	Pattern Count
OT	Overlap Threshold
SR	Sequential Redundancy

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGEMENTS iv

CONTRIBUTORS AND FUNDING SOURCES v

NOMENCLATURE vi

TABLE OF CONTENTS vii

LIST OF FIGURES ix

LIST OF TABLES x

CHAPTER I INTRODUCTION AND LITERATURE REVIEW 1

CHAPTER II PARTIAL SCAN ALGORITHM – LEVEL1 10

- Shift register check 10
- PI/PO check 11
- RAM check 11
- ICG check 11
- Self-drive check 11
- CDC check 11
- Fan-in and Fan-out check 11
- Non-scan to non-scan paths check 12

CHAPTER III SEQUENTIAL REDUNDANCY 14

CHAPTER IV PARTIAL SCAN ALGORITHM – LEVEL2 19

- Overlap threshold check 19
- Sequential redundancy check 23
- Identification of “problematic” DFFs 25
- Ordering non-scan candidate DFFs 27

CHAPTER V OTHER X-GENERATION ISSUES 31

CHAPTER VI RESULTS AND DISCUSSION 34

CHAPTER VII CONCLUSION AND FUTURE WORK 45

REFERENCES 47

LIST OF FIGURES

Page

Figure 1 Scan architecture 2

Figure 2 Hold fixing on scan paths 4

Figure 3 Partial scan design with one non-scan DFF 7

Figure 4 Back to back non-scan flip-flops 12

Figure 5 Sequential redundancy illustration 14

Figure 6 Determining sequential redundancy from full scan SAF and TDF ATPG results 17

Figure 7 Non-scan DFFs feeding in to combinational logic 20

Figure 8 Overlap threshold example 21

Figure 9 Non-scan DFFs with common fan-in and fan-out DFFs 22

Figure 10 Parallels between sequential redundancy and overlap threshold conditions 24

Figure 11 Level1 checks (except non-scan to non-scan connectivity check) 28

Figure 12 Elimination of problematic non-scan candidate DFFs (Part of Level2 checks) 29

Figure 13 Level2 checks and non-scan to non-scan connectivity check performed after problematic DFF identification 30

Figure 14 Non-scan DFFs in fan-out of last SDFF in scan chain 31

Figure 15 Reset overriding for resettable non-scan flip-flop on fan-out of last flip-flop in scan chain 32

Figure 16 Set overriding for settable non-scan flip-flop on fan-out of last flip-flop in scan chain 33

LIST OF TABLES

Page

Table 1 Truth table for circuit in Figure 5 15

Table 2 Original partial scan results 35

Table 3 Partial scan results with overlap threshold/sequential redundancy check 36

Table 4 Partial scan with semi-automatic identification of problematic non-scan DFFs 38

Table 5 Partial scan with automatic identification of problematic non-scan DFFs 39

Table 6 Partial scan results for GPU shader core 2 41

Table 7 GPU Shader Block Area/Density results 42

Table 8 GPU Shader Block Power analysis results 43

Table 9 GPU Shader Block Frequency results 43

Table 10 GPU Shader Block TNS results 44

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

Scan [1] based testing is the most commonly used structural Design for Test (DFT) technique. In this technique, all D flip-flops (DFFs) in the design are replaced with their equivalent scan DFFs (SDFFs). The difference between a non-scan DFF and its scan counterpart is that an SDFF has a two-input multiplexer at its D input. A scan enable (SE) signal selects either the functional (when SE=0) or the scan input (SI) (when SE=1). Scan input is nothing but the output of another SDFF in the design. Thus, when SE=1, all DFFs in the design are configured into scan chains, thus enabling values to be shifted in and out of these flops. When SE=0, the DFFs are configured to their functional mode letting them sample data from the combinational logic.

By effectively turning all DFFs into control and observe points, scan-based testing converts a sequential testing problem into a combinational problem. Figure 1 illustrates scan-based testing. This example shows a simple design containing 3 SDFFs. When SE=1, these 3 registers form a shift register whose values can be shifted in from the scan input pin and shifted out through the scan output pin. The primary input and output pins are under the control of the tester. Thus, all inputs to the combinational logic cloud can be controlled by the tester. When SE=0, the SDFFs sample the inputs coming from the combinational logic cloud. Then again, the SE pin is made high so that the response captured into the SDFFs is shifted out onto the tester.

While this example design has a single scan chain, large designs typically have many scan chains in order to save the time required to shift values into all SDFFs. The basic scan-based testing can be summarized as below:

i) Shift values into scan chains through scan input ports (SE=1).

ii) After all SDFFs in the design are loaded with values, set SE=0. This lets all SDFFs capture their functional inputs.

iii) Again, set SE=1 and shift all the captured values out through the scan output ports.

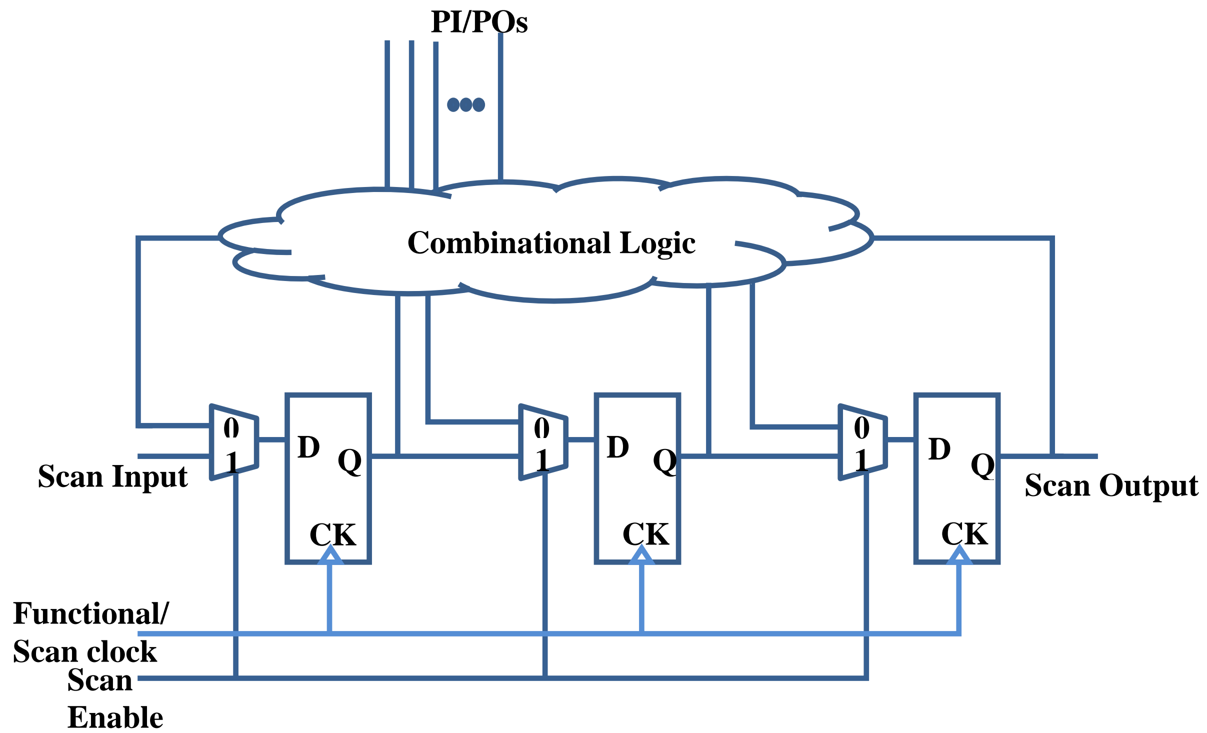

Figure 1 Scan architecture

By transforming testing of a sequential design into a combinational testing problem, scan-based architecture has proven itself to be a very convenient structural testing technique.

But this convenience comes at a price. Below are the primary drawbacks of this technique:

i) Every DFF in the design needs to have a two-input multiplexer at its D-input. This additional multiplexer creates the following problems:

   a) Additional area

   b) Additional power consumption: The additional multiplexers will lead to more static and dynamic power consumption.

ii) Timing closure problems: The paths from the output of a DFF to the scan input of the next DFF typically cause hold timing issues. This is solved by buffering these paths, which further worsens area and power consumption.

Figure 2 shows the example design with buffering included for hold time fixing. This hold fixing problem is an overhead for the implementation team and can delay tape-out.

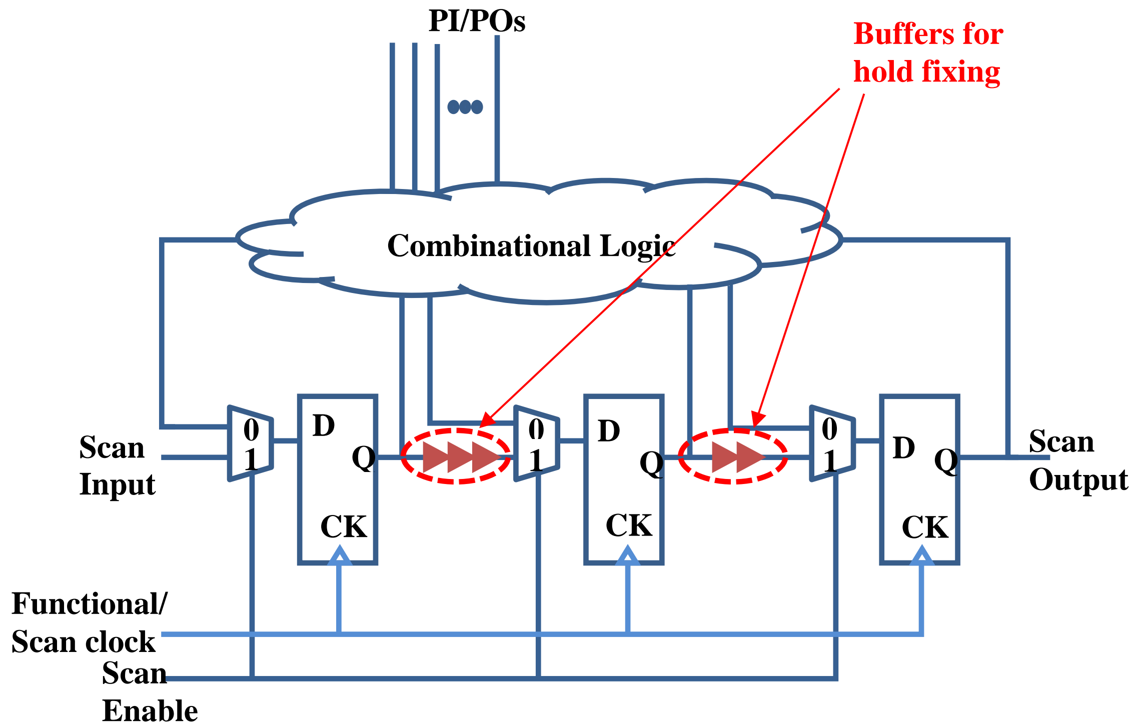

Figure 2 Hold fixing on scan paths

One promising solution that can achieve the ease of testability of a scan-based design while mitigating the above drawbacks is partial scan. Partial scan is an old topic [2] [3] [4] [5] [6] [7] [8], but with the new low power requirements of the Internet of Things (IOT), we revisit the possibilities on current designs. The idea of partial scan is to leave a subset of the DFFs as non-scan. These non-scan DFFs do not include the scan multiplexer at their D-input. Also, the scan path from their output is avoided.

In [2] [3], it is asserted that the cyclic nature of sequential circuits that occur due to feedback paths is the main reason for testability complexities in a partial scan design. Feedback

paths are broken to ease the testability of these paths i.e. DFFs that feedback are made SDFFs. This work is extended in [4] by minimizing the maximum distance between non-scan to non-scan DFFs. A minimal set of SDFFs are chosen such that cyclic paths are avoided for non-scan DFFs. The algorithm which solves the "dmax problem" then selects non-scan DFFs ensuring all non-scan to non-scan DFF path lengths are below a certain limit. This limit differed for different benchmark circuits. In the balanced structural scan test [5], SDFFs are chosen such that all paths between any two SDFFs have the same number of non-scan DFFs, i.e. the paths between any two SDFFs are balanced. Values shifted into the SDFFs are held constant for a few cycles before capturing the circuit response, to allow scan values to propagate through non-scan DFFs.

A cost function optimization-based approach was presented in [6][7]. Here the problem of selecting non-scan DFFs is approached by solving an optimization problem that considers both performance improvement and testability. Factors such as congestion and timing are considered in the cost function. The different cost factors are then scaled according to their importance. The challenge in this approach is to come up with the most optimal cost function that will work for all designs. Another problem is that it needs considerable input from the physical design team.

In [8] two approaches are presented for selecting non-scan DFFs. In the first approach, exhaustive functional fault simulation is performed on the circuit to determine all possible test vectors for every fault. The DFFs required to be set to specific values are noted in each of these tests. Final test vectors are chosen such that most faults can be covered by making the least number of DFFs as SDFFs. Such exhaustive simulations are impractical in modern designs. The second approach is similar to the first one except that an exhaustive test generation is avoided. Instead, only one test is generated per fault site. While the second approach is more computation friendly, for the same number of non-scan DFFs, it is not as effective as the first approach for ATPG results.

In this work it is assumed that non-scan DFFs are always bad control/observe points, which may not always be the case.

Figure 3 shows a three DFF design with one non-scan DFF. As shown in Figure 3, since the DFF in the middle is non-scan, scan path hold fixing may be required only for the path between the first and the last DFFs. Another noteworthy fact is that having only a subset of the DFFs as SDFFs:

i) reduces the number of scan chains

and/or

ii) reduces the number of scan cells in each scan chain.

Thus, apart from the potential power, performance and area (PPA) enhancement, a partial scan design would need fewer bits per pattern to load as compared to its equivalent full scan design.

Unlike full scan design, a single capture cycle is not sufficient for detecting stuck-at faults (SAFs) in a partial scan design. In Figure 3, we can see that two capture cycles are required to propagate faults captured by the non-scan DFF to the SDFF in its fan-out cone. Partial scan mandates multiple capture cycles for SAF testing. For SAF testing in full scan designs, multiple capture cycles are used typically to reduce pattern count [9] [10]. In [11], multiple capture cycles were used in N-detection test scenarios.

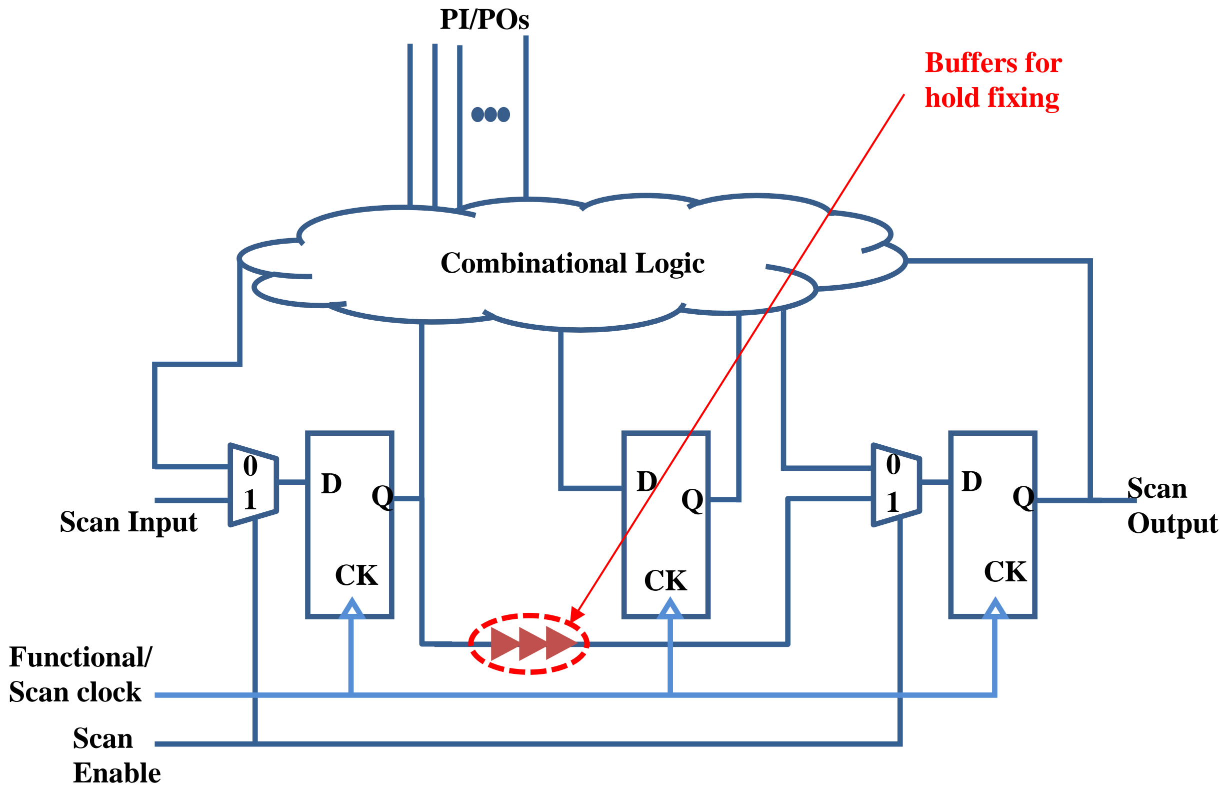

Figure 3 Partial scan design with one non-scan DFF

A non-ideal partial scan design can have negative ramifications. Unlike SDFFs that are loaded with desired values through scan chain shifting, non-scan DFFs in the design are loaded with values through the functional logic. This means all inputs to the combinational logic cloud in the fan-in of the non-scan DFF need to be set with appropriate values to load that non-scan DFFs with the desired values. Non-scan DFFs are no longer safe observe/control points unlike their scan counterparts. This can have the below negative consequences for DFT:

i) Lower Test Coverage (TC)

ii) Higher Pattern Count (PC)

To enjoy PPA enhancements without the above drawbacks, an optimum non-scan DFF selection algorithm (henceforth called partial scan algorithm) is required. For non-scan candidacy, this algorithm must identify those DFFs that are easy to control through the combinational logic fan-in and whose captured values are easy to propagate to SDFFs in the fan-out logic. The TC loss and PC increase need to be kept at minimal levels, if not completely eradicated.

In this work, we explore the idea of partial scan in the context of an Arm® GPU shader core. We split our elaborate checks for identifying non-scan DFFs into two levels of partial scan algorithm. Level1 checks are run on all DFFs in the design. Level2 checks are a more complicated and time-consuming set of checks. Non-scan candidates that pass Level2 checks are the final non-scan DFFs.

We use the ATPG tool's ability to simulate the last few shift cycles to determine the values of non-scan DFFs, mitigating the issue of unknown ("X") values in the non-scan DFFs propagating in the logic.

We identify a category of faults in SAF test, which are sequentially redundant [12] faults. These SAFs cannot be tested in partial scan designs. We provide a simple yet efficient way to identify these faults and add a check in Level2 to minimize them.

We then perform SAF and transition delay fault (TDF) ATPG to prove the effectiveness of our partial scan algorithm. Final TC is within 0.1% of the full scan design for both fault models. PPA results prove why partial scan is worth considering for modern designs, especially in areas such as near-threshold technology.

In Chapter II, we describe Level1 checks of our partial scan algorithm. We introduce sequential redundancy in Chapter III. Chapter IV details Level2 checks and how we address

sequential redundancy. Chapter V discusses other X-generation issues. We present our ATPG and PPA results in Chapter VI. Conclusions and future work are discussed in Chapter VII.

CHAPTER II

PARTIAL SCAN ALGORITHM – LEVEL1

In Level1 of our partial scan algorithm, we run some preliminary checks to identify a subset of all the DFFs on which to run the advanced Level2 checks. We refer to the DFFs that pass Level1 checks as non-scan "candidates". The non-scan candidates that pass Level2 checks are the final non-scan DFFs. Level1 checks can be summarized as:

i) Shift register check

ii) Primary input (PI) / primary output (PO) check

iii) RAM check

iv) Integrated Clock Gating (ICG) check

v) Self-drive check

vi) Clock Domain Crossing (CDC) check

vii) Fan-in and Fan-out check

viii) Non-scan to non-scan paths check

These checks are described below.

Shift register check

In the case of a shift register, only the first DFF needs to be an SDFF and the remaining DFFs can be non-scan. Since shift register DFFs are automatically handled by EDA tools and are often already non-scan DFFs, we do not need to include those in our algorithm. For this reason, shift register DFFs fail this check.

PI/PO check

DFFs that are in the fan-out of PIs and those that are in the fan-in of POs are made SDFFs. This is because, at system level, the core will be integrated with other intellectual property (IP). We do not want to jeopardize controllability/observability of logic external to our IP due to their interface with non-scan DFFs. In short, DFFs connected to PIs and POs fail this check.

RAM check

DFFs that interface with memories are made SDFFs. This is to accommodate the possibility that memories may not include an internal scan collar. DFFs in the fan-in and fan-out of memories fail this check.

ICG check

DFFs that lie in the fan-in of ICGs are made SDFFs. We avoid non-scan DFFs in clock gating logic as this becomes too complex for the ATPG tools to handle.

Self-drive check

DFFs that feedback to themselves fail this check and are scan inserted. DFFs that feedback to themselves are difficult to set to a desired value without any hardware modifications.

CDC check

Any DFF that has a fan-in DFF from a different clock domain is made an SDFF. The capture DFFs on CDC paths (that are either false or multicycle) will always be an X after the shift procedure. We do not consider such DFFs as non-scan to reduce the amount of Xs in the design before we enter into capture cycles (we try to make the design as X-free as possible).

Fan-in and Fan-out check

Fan-in and fan-out to a DFF are considered because the more complex the control or the observe logic must be, the less likely the ATPG tool will be able to get coverage with a moderate

number of patterns. Fan-in and fan-out thresholds are variables in the partial scan algorithm to achieve the minimum desired percentage of non-scan DFFs.

Non-scan to non-scan paths check

One of the Level1 rules is that there can be no paths where the launch point and capture point are both non-scan cells. This was done to help reduce the complexity of the logic for the ATPG tool. For N back-to-back non-scan DFFs, the faults captured by the first non-scan DFF need N + 1 capture cycles to be observed at an SDFF at the fan-out of the nth non-scan DFF. The extra complexity of multiple sequential non-scan DFFs may be too much for today's ATPG tools to handle.

Figure 4 shows two back-to-back non-scan DFFs. The following explanation justifies why we avoid such paths.

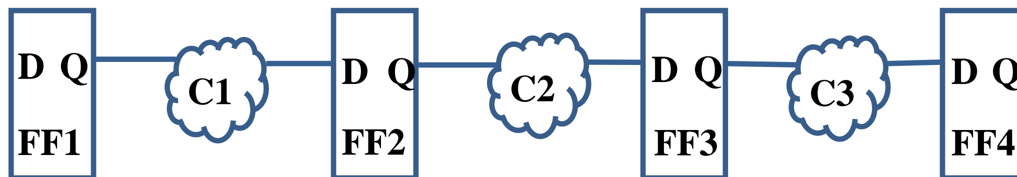

Figure 4 Back to back non-scan flip-flops

DFFs FF2 and FF3 are non-scan DFFs. Because FF2 and FF3 are connected back to back, faults in combinational logic C1 need 3 capture cycles to propagate to SDFF FF4. For N back to back non-scan DFFs, the faults captured by the first non-scan DFF need N + 1 capture cycles to be observed at an SDFF at the fan-out of the Nth non-scan DFF. As this is bad for observability, we avoid non-scan to non-scan paths. For the logic in Figure 4, the following are the only outcomes possible:

i) If FF1 is a non-scan DFF, FF2 must be SDFF.

ii) If FF2 is non-scan DFF, FF1 and FF3 must be SDFFs.

iii) If FF3 is non-scan DFF, FF2 and FF4 must be SDFFs.

iv) If FF4 is non-scan DFF, FF3 must be SDFF.

To summarize, all non-scan candidate DFFs in the fan-in and fan-out of a non-scan DFF fail the non-scan to non-scan connectivity check.

CHAPTER III

SEQUENTIAL REDUNDANCY

By introducing non-scan DFFs, we effectively revert testing from being a purely combinational problem to a sequential one. This creates a new set of SAFs that do not exist in a full-scan design. Figure 5 shows an example of a sequentially redundant [12] fault.

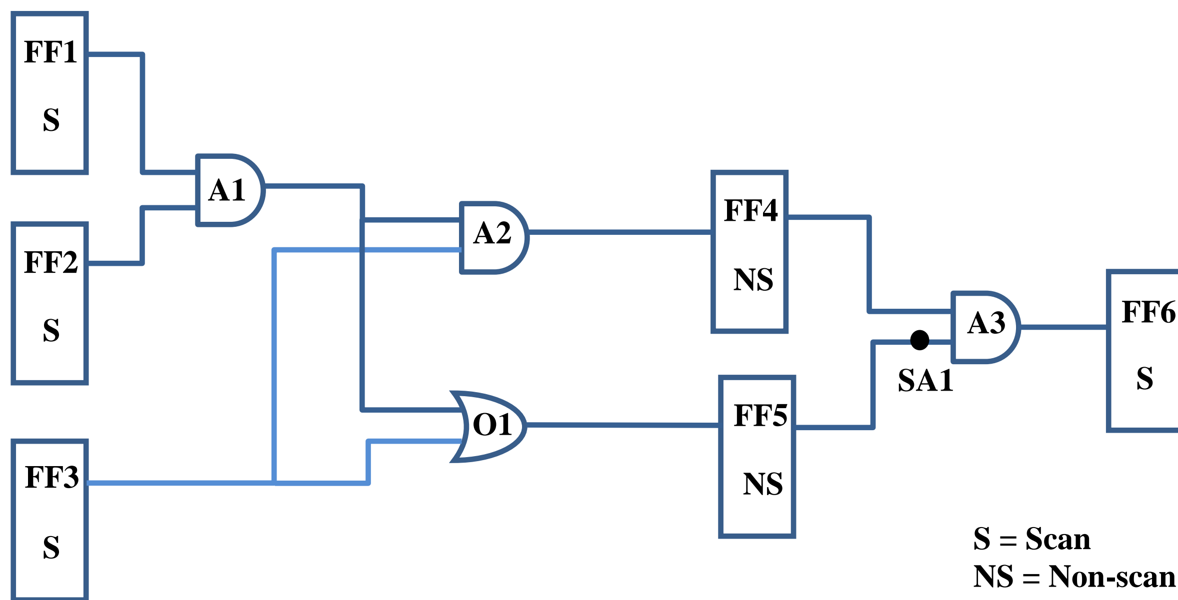

Figure 5 Sequential redundancy illustration

In Figure 5, DFFs FF1, FF2, FF3 and FF6 are SDFFs. DFFs FF4 and FF5 are non-scan DFFs. The values at the Q outputs of FF4 and FF5 in the current cycle depend on the Q outputs of FF1, FF2 and FF3 in the previous cycle. After shifting values into all SDFFs, two capture cycles are needed to propagate the faults captured by non-scan DFFs FF4 and FF5 to the SDFF FF6.

Table 1 shows the truth table for Q outputs of non-scan DFFs FF4 and FF5 as a function of Q outputs of scan DFFs FF1, FF2 and FF3.

Table 1 Truth table for circuit in Figure 5

Cycle N			**Cycle N+1**	
Q_{FF1}	Q_{FF2}	Q_{FF3}	Q_{FF4}	Q_{FF5}
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

To detect the stuck-at 1 (SA1) fault in Figure 6, we need the output values of FF4 and FF5 (QFF4, QFF5) set to 1 and 0, respectively. As shown in Table 1, (QFF4, QFF5) can never achieve the value set (1, 0). This SA1 defect is a *sequentially redundant* fault. If FF4 and FF5 had been SDFFs, they could have been set to any of the four possible values during shift mode and detected the SA1 fault.

We define a sequentially redundant fault as a fault whose presence does not affect the behavior of the sequential circuit. Combinational redundant faults (which are automatically detected by the ATPG tool) are not included in our definition of sequentially redundant faults. Sequential and combinational redundant logic can be present in a design due to improper RTL coding style or ineffective optimization by the synthesis tool. Sometimes they are deliberately intended to address timing issues. Sequentially redundant faults were found in our partial scan design and must be addressed if they are significant in number.

Classical research work on identifying sequential redundancy has focused on logic optimization during synthesis. Sequential redundancy was detected in [13] by looking for don't care conditions in state transition diagrams. In [14], a circuit was made feedback free by cutting at the feedback lines assuming they are fully controllable and observable. Test generation and fault simulation were then performed on this feedback free circuit. C-cycle redundancy was presented in [15]. In this work, an arbitrary set of inputs was provided to the circuit for C clock cycles (where C >= 0). The possible states of the circuit at the end of these C cycles were analyzed for both fault-free and faulty cases to determine sequential redundancy. If a state Sf in the faulty circuit and a state S in the fault-free circuit existed such that the response of both the circuits to any input I was the same, then the fault was considered sequentially redundant. The circuit was then optimized by

removing the region associated with that fault. It was shown that different benchmark circuits needed different values of C to detect all the redundant faults.

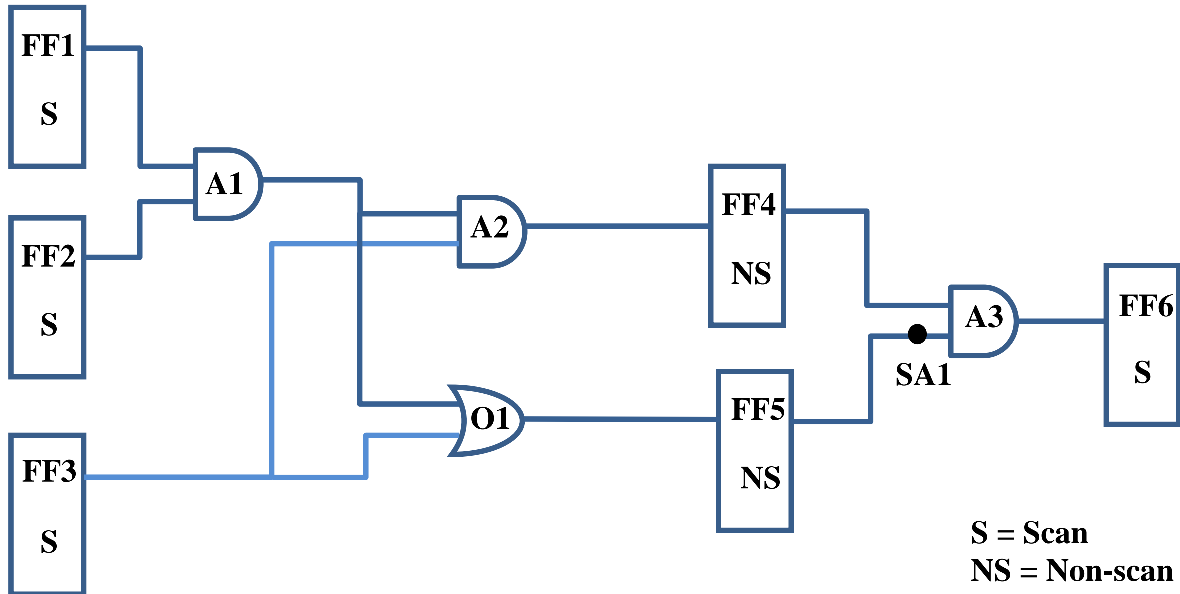

Figure 6 Determining sequential redundancy from full scan SAF and TDF ATPG results

We came up with a simple way of determining the possibility of sequentially redundant faults even before generating the partial-scan design. We first perform SAF and Launch-off Capture (LOC) [16] TDF ATPG on the full-scan design. Sequentially redundant faults are not detected during TDF ATPG. If we consider the sequentially redundant SA1 fault in Figure 6, the slow-to-fall fault on the same node will remain untested. This is because, during the launch of capture transition, all DFFs are in their functional mode and sequentially redundant faults are

exposed. During SAF ATPG, the sequentially redundant faults are hidden since there is control and observe at every DFF. We determine sequentially redundant faults as those that are undetected during full-scan transition delay ATPG and detected during full-scan SAF ATPG. This technique does not involve using synthesis or test generation/fault simulation engines. Instead, it only relies on existing full-scan ATPG results.

Determining sequentially redundant faults beforehand allows us to use that information in a Level2 check of our partial scan algorithm. This enables us to generate a partial scan netlist with a minimal amount of sequentially redundant faults.

CHAPTER IV

PARTIAL SCAN ALGORITHM – LEVEL2

Level2 checks are performed on non-scan candidate DFFs i.e. DFFs that pass Level1 checks. Level2 checks are summarized below:

i) Overlap threshold check

ii) Sequential redundancy check

iii) Identification of "problematic" DFFs

We will show why it is necessary to process the non-scan candidate DFFs in a specific order instead of a random order. At the very end, we will show our entire partial scan algorithm in the form of flowcharts.

Level2 checks are elaborated below.

Overlap threshold check

The overlap threshold check was added to control the number of non-scan DFFs that feed into a single combinational logic cloud. The GPU is a computationally intensive design, resulting in DFFs whose fan-in ranges from 1 to 18,000 and fan-out from 1 to 4,000. We looked at modules that included a high level of computational logic and identified a common problem: high fan-in from non-scan DFFs into combinational logic resulted in too many untested faults in that logic. This condition also exacerbated the propagation of faults captured by non-scan DFFs that needed to be propagated through that logic. With these observations, we concluded that in certain modules, when high fan-in includes many non-scan DFFs, this can significantly reduce test coverage. We call this the "overlap threshold" problem. Figure 7 illustrates this condition.

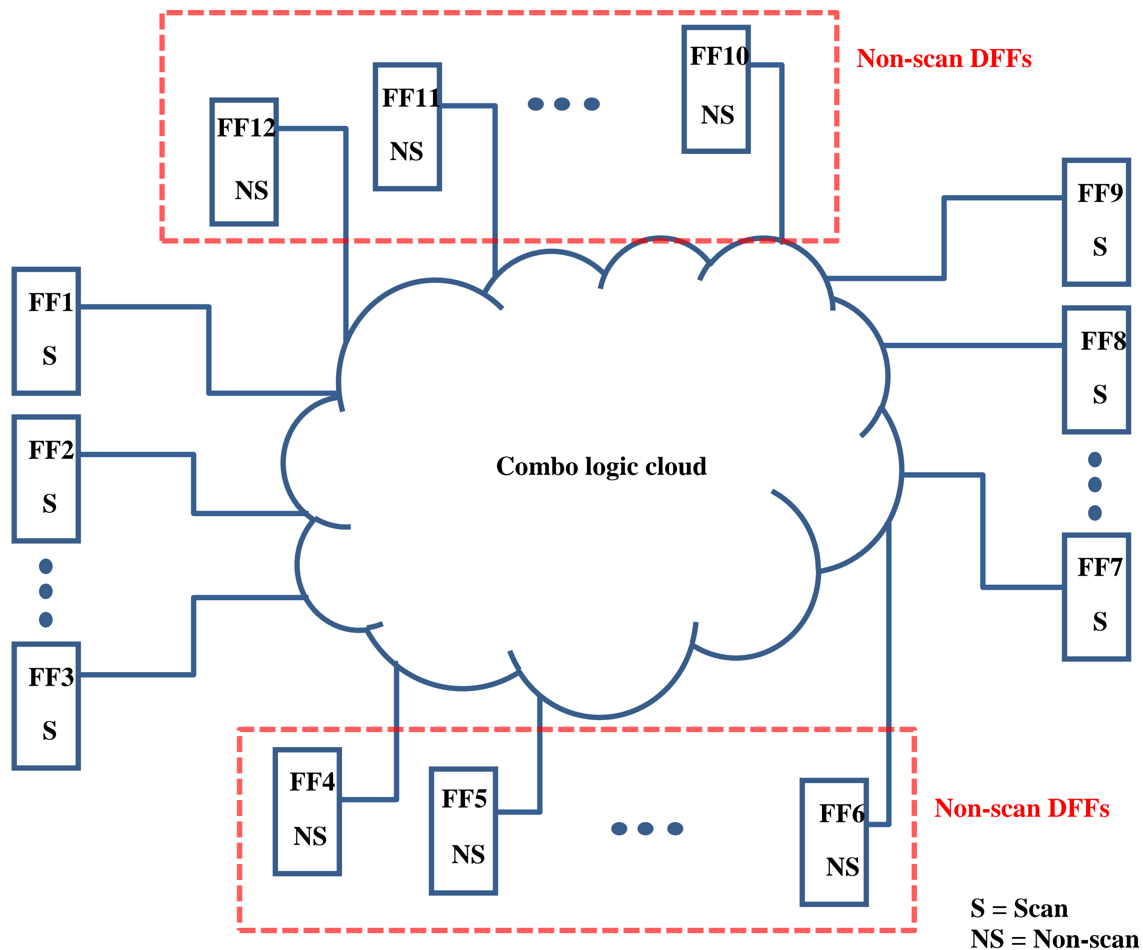

Figure 7 Non-scan DFFs feeding in to combinational logic

To limit the number of non-scan flip-flops feeding into a combinational logic cloud, we limit the number of non-scan flip-flops that fan-in to a scan flip-flop. In Figure 7, if we limit the maximum number of non-scan flip-flops that fan-in to scan flip-flops FF7, FF8 and FF9 to some value N, it is guaranteed that the combinational logic cloud shown in the figure will not have more than N non-scan flip-flops feeding in.

Figure 8 gives a more specific example. FF1, FF2, FF3, FF4, FF5, FF7 and FF8 are non-scan DFFs feeding into the combinational logic fan-in to SDFF FF6. For faults captured by FF2 to propagate to FF6, FF1, FF3, FF4 and FF8 must be a 1 while DFFs FF5 and FF7 must be a 0.

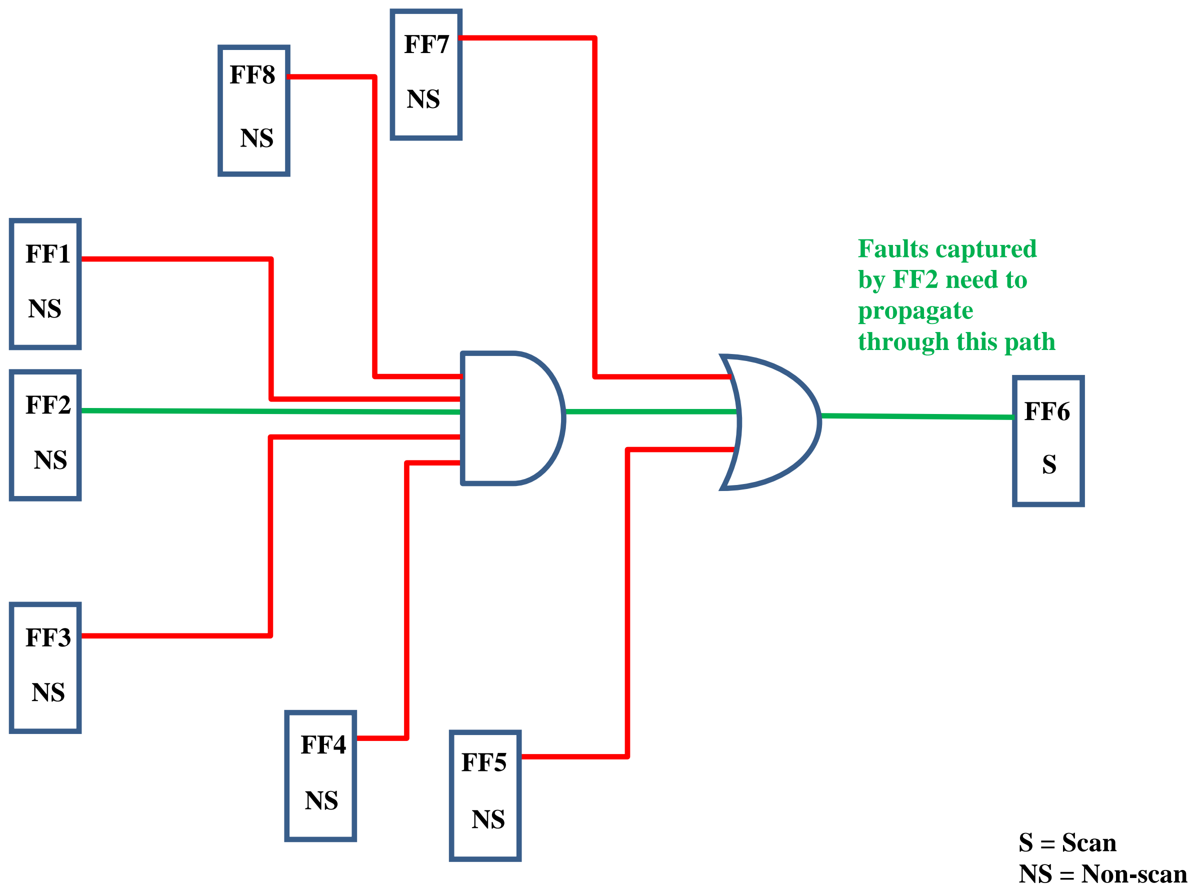

Figure 8 Overlap threshold example

Since non-scan DFFs are weak control points, it might be difficult for the ATPG to satisfy all the constraints. If we limit the maximum number of non-scan DFFs that fan-in to SDFF FF6 to some value N, it is guaranteed that the combinational logic cloud (AND gate + OR gate) shown in the figure will not have more than N non-scan DFFs feeding in. This is a simple yet effective way of implementing overlap threshold check.

In the next section, we will reveal how we addressed both sequential redundancy and the overlap threshold problem by using only the overlap threshold check.

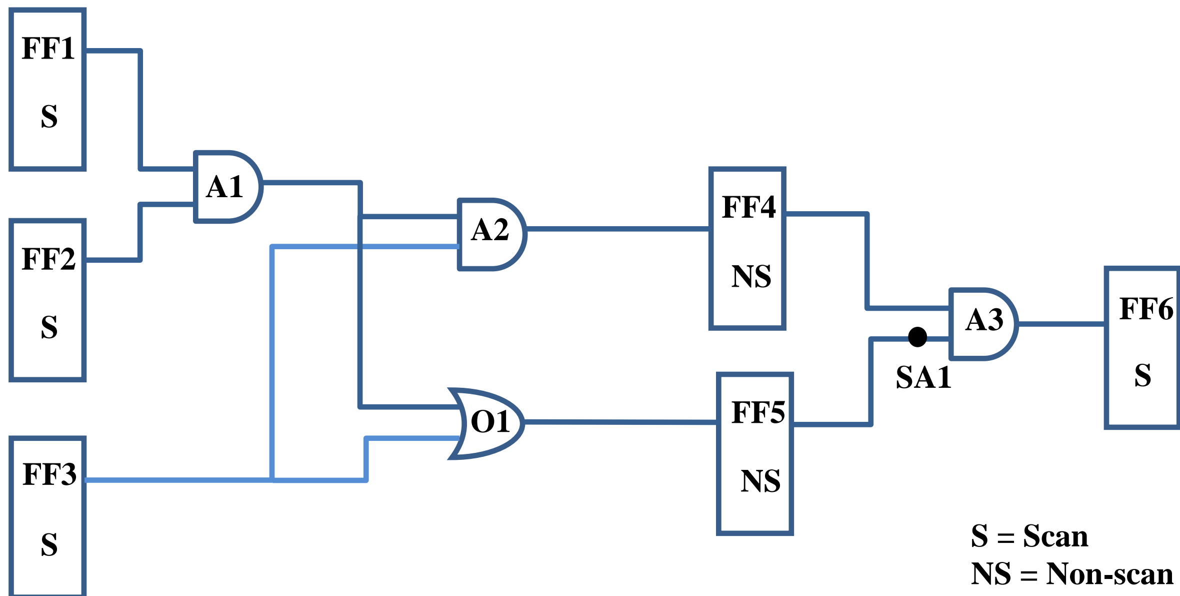

Figure 9 Non-scan DFFs with common fan-in and fan-out DFFs

Sequential redundancy check

Sequential redundancy was described in Chapter III. In this section, we will discuss how the sequential redundancy check was implemented using the overlap threshold check. To create a simpler implementation, we chose an aggressive way to tackle sequential redundancy.

Non-scan DFFs that illuminate sequential redundancy have common fan-in and fan-out DFFs. This is evident in Figure 9. In the event of two or more non-scan candidate DFFs having common fan-in and fan-out DFFs, only one of those candidates will pass the sequential redundancy check. In Figure 9, only FF4 or FF5 will end up passing the check. Just because there are non-scan candidates that have common fan-in and fan-out DFFs does not mean there will be sequential redundancy. However, the simplified flow is more aggressive in replacing these DFFs with SDFFs. Figure 10 illustrates the parallels between sequential redundancy and the overlap threshold problem.

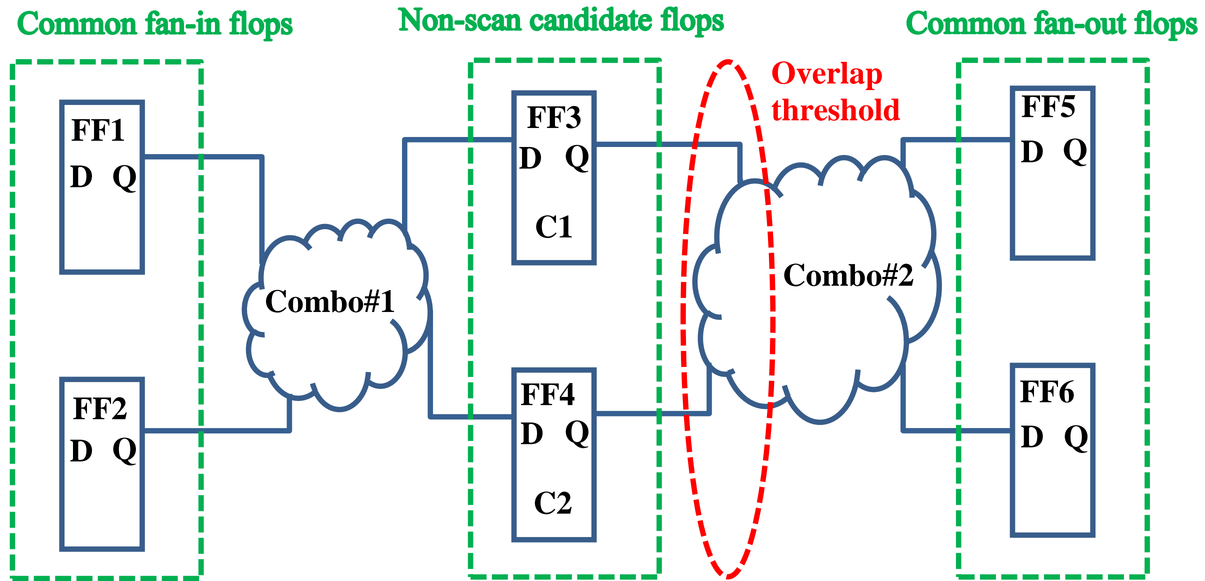

Figure 10 Parallels between sequential redundancy and overlap threshold conditions

In Figure 10, there are two non-scan DFF candidates FF3 and FF4 that need to be evaluated for sequential redundancy and the overlap threshold condition. The candidates have common fan-in DFFs: FF1 and FF2, and common fan-out DFFs: FF5 and FF6.

Hence, only one of them will end up passing the sequential redundancy check based on the order in which they are processed. If FF3 is processed before FF4, FF4 will fail the sequential redundancy check. While checking for the overlap threshold condition, we consider the DFFs in the fan-out of the candidate and ensure that the number of non-scan DFFs that fan-in is below the set threshold.

When FF3 is processed, we look at the number of non-scan DFFs that fan-in to DFFs FF5 and FF6; the same is true for FF4. Now, if we set the overlap threshold limit to one (i.e. only one non-scan DFF can fan-in to a combinational logic cloud), FF5 and FF6 will have only one non-scan DFF in their fan-in. Assuming FF3 is processed before FF4, FF3 will end up passing the overlap threshold check and FF4 will not. Thus, the overlap threshold limit of one meets the sequential redundancy requirements.

Implementing an overlap threshold limit of 1 is very aggressive and we ended up losing a considerable number of non-scan DFFs. Our non-scan DFF count dropped from ~30% to ~20% after adding this check. But by losing only a third of the non-scan DFFs, we regained two-thirds of the lost SAF coverage.

The overlap threshold check enables a lot of flexibility. We can set the overlap threshold limit to one to push the coverage as high as possible. An overlap threshold limit of one is the most conservative value: it completely gets rid of sequential redundancy and the overlap threshold problem. However, the threshold can be varied anywhere from one to the maximum fan-in minus one. With higher threshold values, we can get a higher non-scan DFF count usually at the cost of lower coverage and/or a higher pattern count due to an increase in sequential redundancy and the overlap threshold problem. However, as in LOC transition delay testing, one could argue that the lost coverage is for faults that can never affect functional operation.

Identification of "problematic" DFFs

After addressing the overlap threshold and sequential redundancy problems, we still saw around 0.1% lower test coverage as compared to the full scan design. The non-scan DFFs that affected these faults, either in the observe or control paths of the faults, were collected. We refer to these non-scan DFFs as "problematic" DFFs. We associated a weight with each problematic

DFF. The weight is the number of untested faults on a control or observe path from that DFF. We ranked these problematic DFFs in decreasing weight order to identify the worst ones. DFFs with higher weight are associated with more untested faults.

We identified the following properties of the problematic DFFs:

i) Many were high fan-in/fan-out DFFs. High fan-in/fan-out DFFs affect more logic, hence can be problematic. Based on this, we adjusted the maximum fan-in threshold to 120 and maximum fan-out threshold to 100 on the GPU shader block. DFFs with thresholds above these levels will be non-scan DFF candidates. Note that these thresholds have nothing to do with overlap threshold. These are the thresholds described in Chapter II.

ii) There were low fan-in/fan-out problematic DFFs. These DFFs were involved in arithmetic intensive logic and the combinational logic depth between these DFFs and the SDFFs in their fan-out was high.

There are two ways to address the low fan-in/fan-out problematic DFFs problem:

i) We select the problematic DFFs whose weight is above a certain value and exclude them while processing non-scan DFF candidates. This is a more accurate approach with some iterative work involved. It is iterative since making a problematic DFF a SDFF may allow another DFF to become non-scan. We call this the "semi-automatic" way of identifying problematic DFFs. For the GPU shader block, we selected 100 as the cut-off weight. So, any problematic DFF whose weight was above or equal to 100 was excluded and made SDFF.

ii) We check if a candidate DFF is involved in arithmetic logic and exclude it if none of its fan-out DFFs have a fan-in below a certain threshold. To check if a candidate

is involved in arithmetic logic, we look for adder cells in the fan-in/fan-out. By having at least one fan-out DFF with low fan-in, the non-scan DFF has at least one easy path to propagate captured faults. Here, we assume that a low fan-in DFF has less combinational logic in its fan-in. This is an automatic but less accurate technique than i).

Ordering non-scan candidate DFFs

In three of our checks: overlap threshold check, sequential redundancy check and non-scan to non-scan connectivity check, the order in which we process the non-scan candidate DFFs has a significant impact on the non-scan DFF count and ATPG results. Non-scan candidates that are processed early and pass all checks jeopardize the chances of other candidate DFFs that overlap with them. Besides, it is guaranteed that the non-scan candidates in their fan-in and fan-out will be made SDFFs. This can be disastrous to non-scan DFF count if high fan-in/fan-out DFFs are processed early on.

We process non-scan candidate DFFs in order of increasing fan-in. All candidates with a given fan-in are processed in order of increasing fan-out. We gave preference to low fan-in over low fan-out DFFs because it is easier to set desired values in low fan-in DFFs due to smaller combinational logic in the fan-in cone.

Our partial scan algorithm is shown in the below flow charts. Note that the flow charts show the various checks in the order in which they are implemented. Non-scan to non-scan paths check which is a Level1 check is moved to the end of the flow. Figure 11 shows the Level1 checks (except non-scan to non-scan paths check).

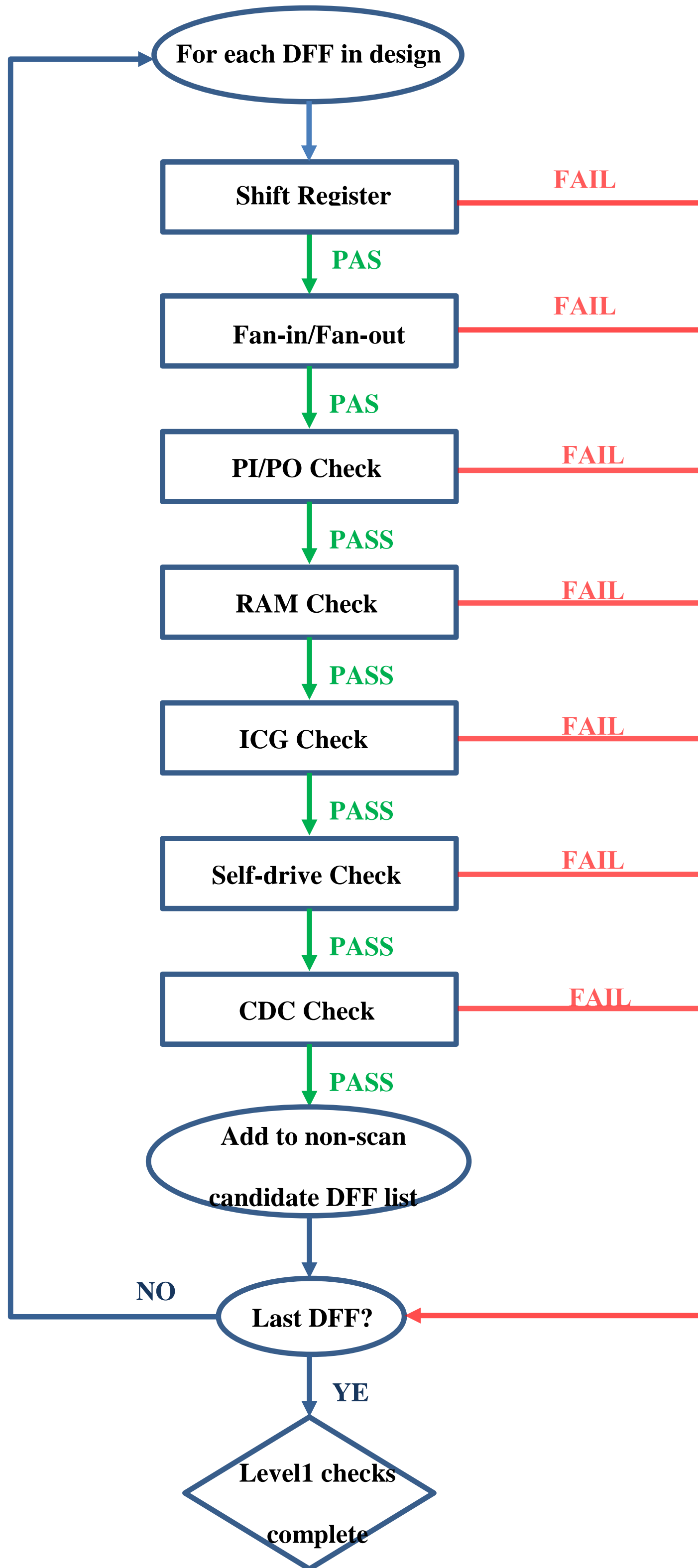

Figure 11 Level1 checks (except non-scan to non-scan connectivity check)

As shown in Figure 12, problematic non-scan candidate DFFs are eliminated and the remaining candidates are eligible for the remaining Level2 checks. As mentioned previously, there are two ways to identify problematic non-scan candidates.

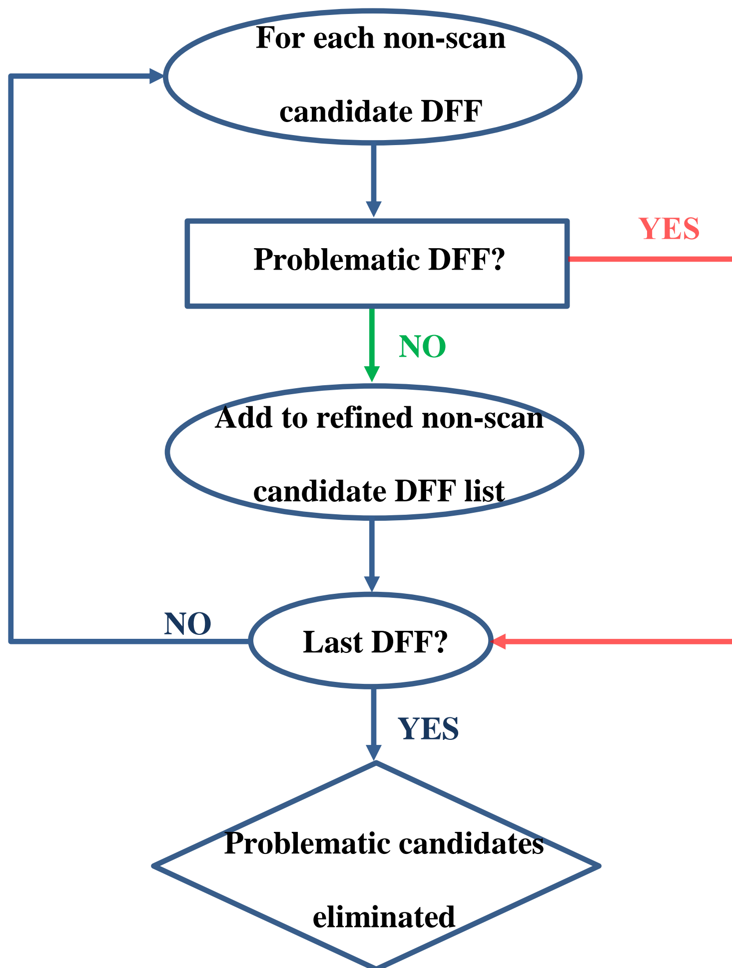

Figure 12 Elimination of problematic non-scan candidate DFFs (Part of Level2 checks)

Figure 13 shows the Level2 checks and non-scan to non-scan connectivity check.

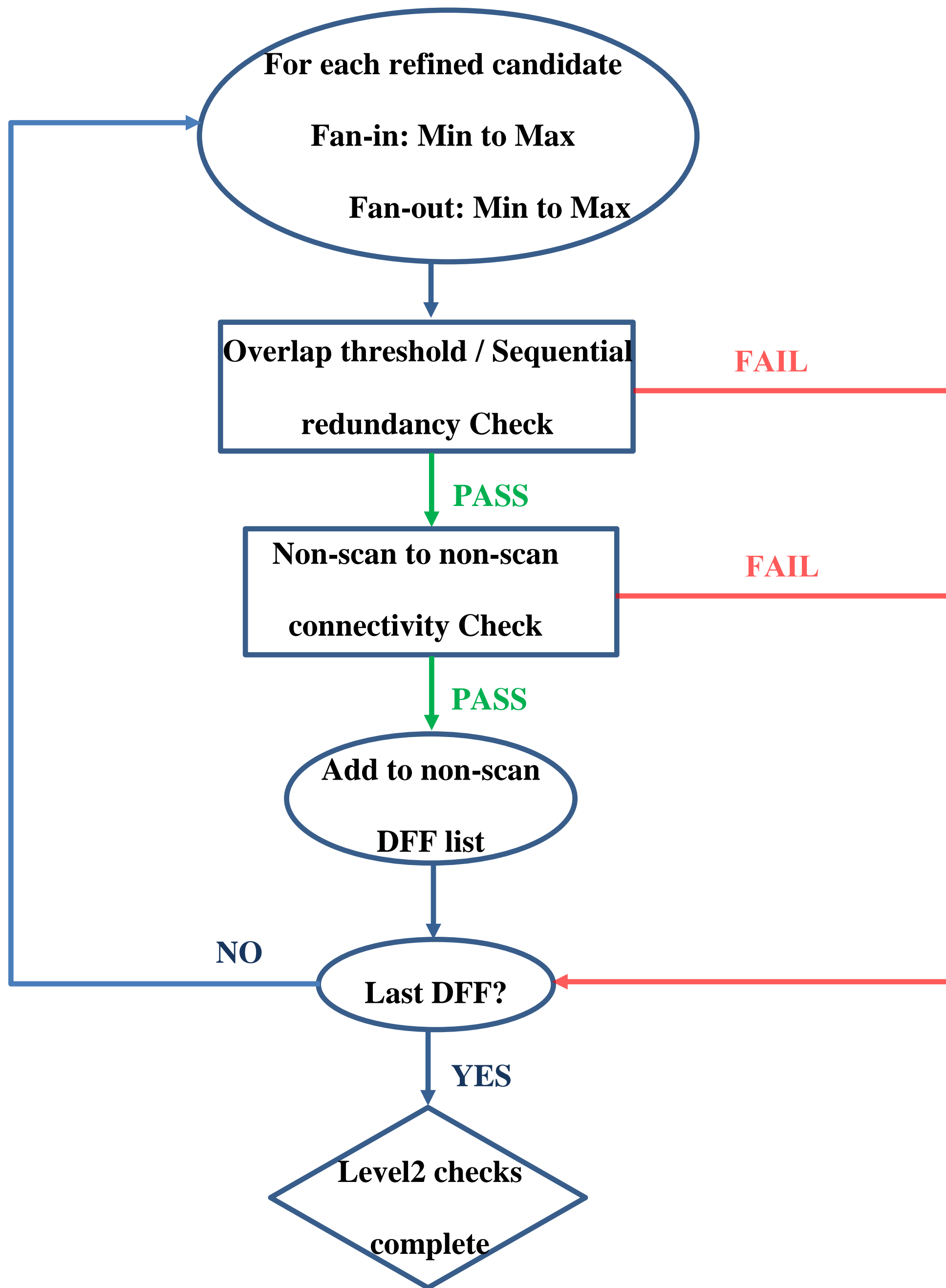

Figure 13 Level2 checks and non-scan to non-scan connectivity check performed after problematic DFF identification

CHAPTER V

OTHER X-GENERATION ISSUES

Even though we simulate the last few shift cycles, we found that there was still some X generation. Since the value in the last DFF of a scan chain is known only after all shift cycles, the non-scan DFFs that lie in the fan-out of these DFFs propagate X's during the capture phase. Figure 14 shows two non-scan DFFs in the fan-out of last SDFF in a scan chain.

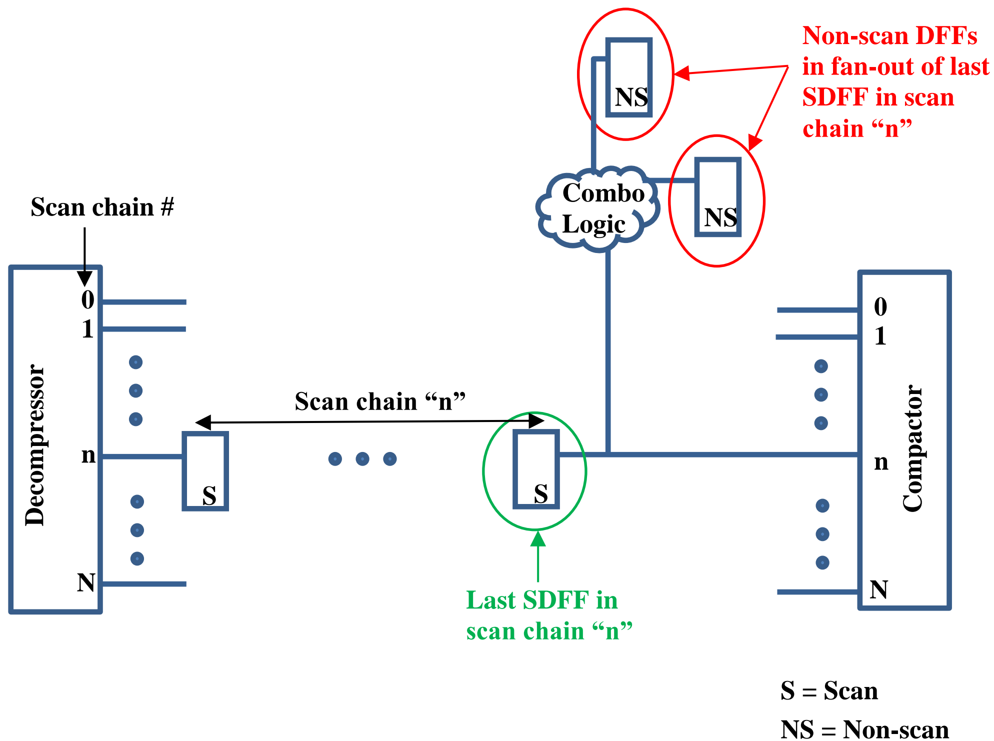

Figure 14 Non-scan DFFs in fan-out of last SDFF in scan chain

These DFFs will capture X's during the first capture cycle since the tool is unable to simulate shift values for the last DFF in a scan chain. To prevent this, we override the set/reset ports of all non-scan DFFs that are functionally adjacent to the last SDFF of each scan chain with the scan enable signal so that they are either 1 or 0 at the end of shift instead of X as described in [17] . If these non-scan DFFs are neither set nor reset DFFs, then we use a set or reset DFF to be able to initialize them during shift. Figure 15 shows an example of how a resettable non-scan DFF on the fan-out of last DFFs in scan chains is handled. A settable DFF shown in Figure 16 is handled in the same manner where shift enables the set, rather than the reset. This resolved the issue that was causing the X-generation.

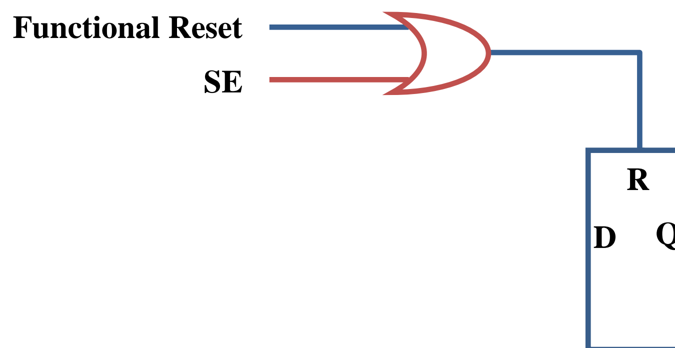

Figure 15 Reset overriding for resettable non-scan flip-flop on fan-out of last flip-flop in scan chain

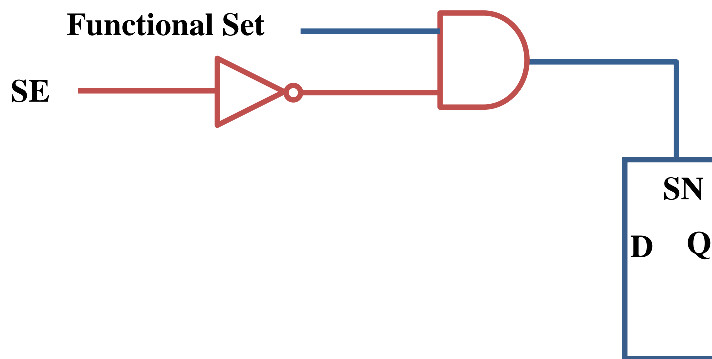

Figure 16 Set overriding for settable non-scan flip-flop on fan-out of last flip-flop in scan chain

During shift, SE=1. This guarantees the non-scan DFF in Figure 15 to be reset during and at the end of the shift phase. The non-scan DFF in Figure 16 is guaranteed to be set during and at the end of the shift phase. During functional mode, SE=0, so the overriding shown in Figure 15 and Figure 16 never affect functional mode of operation. In case of non-scan DFFs on the fan-out of last DFFs in scan chains that have both set and reset pins, either set or reset can be overridden.

Non-scan DFFs on the fan-out of last DFFs in scan chains that are neither set nor reset DFFs are dealt with in a unique way. We make these DFFs set or reset DFFs depending on whether they fan-out to AND/NAND or OR/NOR gate. We configure these DFFs to be non-controlling to ease the detection of all faults propagated through the other input of the AND/NAND or OR/NOR gate. So, if the Q output of the DFF fans out to an AND gate without any inversion, we make it a set DFF. If the Q output of the flip-flop fans out to a NOR gate without inversion, we make it a reset DFF.

CHAPTER VI

RESULTS AND DISCUSSION

All ATPG results shown in this section are with compression enabled. The same compression hardware configuration is used for both full scan and partial scan.

Since the number of scan chains in the design is fixed, there will be fewer DFFs per scan chain in partial scan as compared to full-scan. This means fewer shift cycles per pattern. Hence, to make a fair comparison w.r.t. pattern count, we introduce a new parameter: pattern volume which is defined below:

Pattern volume = (Pattern count) x (Number of flip-flops in longest scan chain)

Table 2 shows the full scan and original partial scan results for GPU shader core 1. The original partial scan netlist is the one which does not include overlap threshold / sequential redundancy checks. Also, the problematic DFFs are not excluded. The order of processing non-scan candidate DFFs is increasing fan-out, increasing fan-in.

Table 2 Original partial scan results

Netlist	Non-scan DFF % w.r.t. total DFFs (%)	Stuck-at ATPG			Transition ATPG		
		Test coverage (%)	Pattern Count	Pattern volume (K)	Test coverage (%)	Pattern count	Pattern volume (K)
Full scan		99.75	11573	11631	97.46	27200	27336
Original Partial scan	29.87	99.01	22396	15946	95.84	54243	38622

SAF TC drops by 0.74% and TDF TC by 1.62%. PC nearly doubles in both cases. Pattern volume increases by 37% and 41% for SAF and TDF ATPG respectively.

Next, we determine the modules that have sequential redundancy and overlap threshold problems. We determine modules that exhibit sequential redundancy by determining the full scan faults that are both uncovered in TDF ATPG and covered in SAF ATPG. We then report these faults hierarchically to determine the modules where sequential redundancy is concentrated.

It is important to understand that that not all sequentially redundant faults determined using full scan TDF and SAFATPG results exist in the partial scan netlist. The sequentially redundant faults determined this way give us all possible sequentially redundant faults. Sequentially redundant faults contributed to ~0.26% (74K) of the total faults in the GPU shader core.

Next, we need to determine the possibility of overlap threshold problem and the modules where it is concentrated. For this, we take all uncovered faults from partial scan and delete all sequentially redundant faults determined earlier. We also delete all untested faults from the full

scan ATPG run. This is done because the untested faults in full scan cannot be detected in partial scan.

Now that we have the untested faults from partial scan that are not sequentially redundant, these need to be debugged manually to check for the overlap threshold problem. There is no automation for this. We manually picked faults across different modules and determined the overlap threshold problem.

We found out that the same modules exhibited both sequential redundancy and overlap threshold problems. We applied an overlap threshold limit of 1 for these modules. Table 3 shows the results with overlap threshold (OT)/sequential redundancy (SR) check applied. Here again, the non-scan candidates are processed in increasing fan-out, increasing fan-in order.

Table 3 Partial scan results with overlap threshold/sequential redundancy check

Netlist	Non-scan DFF % w.r.t. total DFFs (%)	Stuck-at ATPG			Transition ATPG		
		Test coverage (%)	Pattern Count	Pattern volume (K)	Test coverage (%)	Pattern count	Pattern volume (K)
Full scan		99.75	11573	11631	97.46	27200	27336
Original Partial scan	29.87	99.01	22396	15946	95.84	54243	38622
Partial scan w/ OT/SR check	19.98	99.54	17194	13910	96.98	45101	36487

The overlap threshold/sequential redundancy checks cause a loss of one-third of the non-scan DFFs, but we gain 0.53% SAF TC and 1.14% TDF TC w.r.t. the original partial scan netlist. Also, the pattern volume drops by 13% and 5.5% for SAF and TDF ATPG respectively w.r.t. original partial scan. While we lose about a third of non-scan DFFs, we regain about two-thirds of the lost fault coverage.

Next, we determined the problematic non-scan DFFs in the partial scan netlist with overlap threshold/sequential redundancy check. As described previously, there is a semi-automatic and an automatic way to determine problematic non-scan DFFs. Based on the report given by our fault debug script, fan-in and fan-out of non-scan DFFs have an upper limit of 120 and 100 respectively. Also the order of processing non-scan DFFs is changed to increasing fan-in, increasing fan-out order. This order was determined to be better based on the problematic DFFs report. Table 4 shows the results with semi-automatic identification of problematic DFFs.

W.r.t. partial scan with only OT/SR check, we gain 0.14% SAF TC and 0.42% TDF TC. With 20% non-scan DFFs, we are within 0.1% of TC w.r.t. full-scan ATPG. The pattern volume increase is 8% and 24% for SAF and TDF ATPG respectively w.r.t. full-scan.

Table 4 Partial scan with semi-automatic identification of problematic non-scan DFFs

Netlist	Non-scan DFF % w.r.t. total DFFs (%)	Stuck-at ATPG			Transition ATPG		
		Test coverage (%)	Pattern Count	Pattern volume (K)	Test coverage (%)	Pattern count	Pattern volume (K)
Full scan		99.75	11573	11631	97.46	27200	27336
Original Partial scan	29.87	99.01	22396	15946	95.84	54243	38622
Partial scan w/ OT/SR check	19.98	99.54	17194	13910	96.98	45101	36487
Partial scan w/ OT/SR and semi-automatic problematic DFF check	20.36	99.68	15667	12612	97.40	42167	33945

Table 5 shows results with automatic identification of problematic non-scan DFFs. The results of automatic and semi-automatic identification of problematic non-scan DFFs are very close. W.r.t. semi-automatic problematic non-scan DFF identification, automatic identification results in a 0.03% and 0.26% drop in stuck-at and transition TC respectively. Fan-in of the fan-out flip-flops was limited to 70. As expected, semi-automatic identification can pin point the problematic DFFs more accurately.

Table 5 Partial scan with automatic identification of problematic non-scan DFFs

Netlist	Non-scan DFF % w.r.t. total DFFs (%)	Stuck-at ATPG			Transition ATPG		
		Test coverage (%)	Pattern Count	Pattern volume (K)	Test coverage (%)	Pattern count	Pattern volume (K)
Full scan		99.75	11573	11631	97.46	27200	27336
Original Partial scan	29.87	99.01	22396	15946	95.84	54243	38622
Partial scan w/ OT/SR check	19.98	99.54	17194	13910	96.98	45101	36487
Final partial scan w/ OT/SR and semi-automatic problematic DFF check	20.36	99.68	15667	12612	97.40	42167	33945
Final partial scan w/ OT/SR and automatic problematic DFF check	20.34	99.65	16077	12959	97.14	37250	30024

We took all the learnings and attempted to apply them to another Arm® GPU shader core block. Table 6 summarizes all results for shader core 2. Based on the problematic non-scan DFF report given by our fault debug script, no fan-in/fan-out limit was required.

Partial scan results of both the shader core blocks are similar in a number of ways. The original partial scan netlist has 31% non-scan DFFs and one third of the non-scan DFFs are lost by applying overlap threshold/sequential redundancy check. But by losing a third of the non-scan DFFs, we gain half of the lost fault coverage. The coverage is bumped up further by eliminating problematic DFFs. For the final partial scan netlist (with semi-automatic problematic non-scan DFF identification), ATPG TC is within 0.1% of the full scan TC for both SAF and TDF.

In the original partial scan netlist, SAF and TDF ATPG TC are 0.55% and 1.25% below the corresponding full scan ATPG TC respectively. W.r.t. full scan, pattern volume increases by 117% for SAF and 108% for TDF ATPG.

In the partial scan netlist with the OT/SR check, SAF and TDF ATPG TC are 0.22% and 0.36% below the corresponding full scan ATPG TC. W.r.t. full scan, pattern volume increases by 90% for SAF and 89% for TDFATPG.

In the partial scan netlist with OT/SR check and semi-automatic problematic non-scan DFF identification, SAF ATPG TC is 0.07% below full scan SAF ATPG TC. However, the transition TC is 0.06% more than the corresponding full scan TC. W.r.t. full scan, pattern volume increases by 80% for SAF and 87% for TDF ATPG.

Table 6 Partial scan results for GPU shader core 2

Netlist	Non-scan DFF % w.r.t. total DFFs (%)	Stuck-at ATPG			Transition ATPG		
		Test coverage (%)	Pattern Count	Pattern volume (K)	Test coverage (%)	Pattern count	Pattern volume (K)
Full scan		99.78	3956	3217	96.82	10624	8638
Original Partial scan	31.39	99.23	12377	6994	95.57	32190	17914
Partial scan w/ OT/SR check	21.07	99.56	9458	6120	96.46	25212	16313
Final partial scan w/ OT/SR and semi-automatic problematic DFF check	20.21	99.71	8880	5799	96.89	24703	16132
Final partial scan w/ OT/SR and automatic problematic DFF check	20.19	99.60	9073	5934	96.5	22143	14482

For automatic problematic non-scan DFF identification, fan-in of the fan-out DFFs was limited to 100. Unlike the earlier shader core 1, automatic detection of problematic non-scan DFFs is not as effective as semi-automatic problematic non-scan DFF identification. SAF ATPG TC is 0.18% below full scan SAF ATPG TC. TDF TC is 0.32% below the corresponding full scan TC. W.r.t. full scan the pattern volume increases by 85% for SAF and 68% for TDF ATPG.

PPA analysis was performed for the final partial scan netlist (using semi-automatic problematic non-scan DFF identification) for GPU shader core 1. The PPA numbers shown in this section are percentage change in partial scan netlist with respect to the full-scan netlist, due to the proprietary nature of the data. Area and density results are shown in Table 7. Standard cell area dropped by 0.92% and this is reflected on the total cell area. There is no change in RAM area which is expected. No work was done to reduce the floorplan, so there is no change in total die area. Physical utilization of the standard cells has dropped by 0.9%. We learned from our standard cell team that due to their limited usage, the non-scan DFF standard cells are not optimized to the same extent as their SDFF counterparts.

Table 7 GPU Shader Block Area/Density results

Area/Density results	
	Difference w.r.t. full-scan
Standard cell area	-0.92%
Total Area	-0.66%
Density	-0.90%

Our vector-less power analysis results depicted in Table 8 show that leakage power dropped by 0.5% while dynamic power dropped by 1.3%. Dynamic power simulations with vectors were not run on this design.

Table 8 GPU Shader Block Power analysis results

Power analysis results	
	Difference w.r.t. full-scan
Dynamic power	-1.30%
Static power	-0.50%

Frequency results are shown in Table 9. Register to register (reg2reg) paths can now be clocked at a frequency that is 2.09% higher with respect to full-scan. Maximum frequency for input to register (in2reg) and register to output (reg2out) paths increased by 1.45%.

Table 9 GPU Shader Block Frequency results

Frequency results	
	Difference w.r.t. full-scan
regreg paths	2.09%
in2reg/reg2out paths	1.45%

The biggest benefit of partial scan is reflected in the total negative slack (TNS) results shown in Table 10. On reg2reg paths, TNS improved by 77.5%. On in2reg paths, TNS improved by 33.33%. There is no change in TNS for reg2out paths.

Table 10 GPU Shader Block TNS results

Hold TNS results	
	Difference w.r.t. full-scan
regreg paths	-77.50%
in2reg paths	-33.33%
reg2out paths	0.00%

No change was seen in worst negative slack (WNS) result. According to the implementation team that worked on shader core 1, achieving these PPA results with full scan would require about 2 weeks of effort. Partial scan saved those 2 weeks of effort.

CHAPTER VII

CONCLUSION AND FUTURE WORK

We presented our partial scan algorithm as a complex amalgamation of several checks. We introduced sequentially redundant faults that manifest themselves during partial scan SAF ATPG and presented a way not only to determine them but also eliminate/contain them. We presented two different ways to identify problematic non-scan DFFs.

With 20% non-scan DFFs our ATPG TC lies within 0.1% of the full scan TC for both SAF and TDF. We have proven the effectiveness of this algorithm on two different GPU shader cores. In Chapter VI, we saw PPA improvements in the partial scan design, the most noteworthy improvement being in the hold TNS.

One way to extend this work would be to analyze the presence of overlap threshold problem in an automated way. In our work, overlap threshold problem was detected manually. Another area for future work would be to involve test point analysis to determine locations of problematic DFFs.

In our work, we did not have any PPA targets. If DFFs on critical paths or congestion prone areas are made non-scan, PPA improvements can be even better. However, consolidating desired PPA and testability is a challenge. Also, physical design information will be needed before performing scan insertion.

One downside of using partial scan is the reduced debug capability on silicon. The area of post silicon debug on a partial scan design must be investigated. Also, if Logic Built-In Self-Test (LBIST) is used, partial scan might mandate more test points thus negating the benefits. Partial scan for designs with LBIST would be a challenging area for future research.

Near Threshold Voltage (NTV) designs are being considered for ultra-low power applications. Partial scan benefits should be explored in such designs.

REFERENCES

[1] L. -T. Wang, C. E. Stroud, N. A. Touba (Eds.), "System on Chip Test Architectures", Morgan Kaufmann, Burlington, MA, 2008

[2] K. -T. Cheng, V. D. Agrawal, "A partial scan method for sequential circuits with feedback", IEEE Transactions on Computers, Vol. 39, No. 4, pp. 544-548, Washington, DC, 1990

[3] S. T. Chakradhar, A. Balakrishnan, V. D. Agrawal, "An Exact Algorithm for Selecting Partial Scan DFFs", Design Automation Conference, pp. 81-86, San Diego, CA, 1994

[4] D. H. Lee, S. M. Reddy, "On determining scan DFFs in partial-scan designs", IEEE International Conference on Computer-Aided Design, pp. 322-325, Santa Clara, CA, 1990

[5] R. Gupta, M. A. Breuer, "The Ballast methodology for structured partial scan design", IEEE Transactions on Computers, Vol. 39, No. 4, pp. 538-544, Washington, DC, 1990

[6] V. Chickermane, J. H. Patel, "An optimization based approach to the partial scan design problem", International Test Conference, pp. 377-386, Washington, DC, 1990

[7] V. Chickermane, J. H. Patel, "A fault oriented partial scan design approach", IEEE International Conference on Computer-Aided Design, pp. 400-403, Santa Clara, CA, 1991

[8] V. D. Agrawal, K. -T. Cheng, D. Johnson, T. Lin, "Designing circuits with partial scan", IEEE Design and Test of Computers, Vol. 5, No. 2, pp. 8-15, 1988

[9] V. Jain, J. Waicukauski, “Scan Test Data Volume Reduction in Multi-Clocked Designs with Safe Capture Technique”, International Test Conference, pp. 148-153, Baltimore, MD, 2002

[10] I. Pomeranz, S. M. Reddy, “On Test Data Compression and n-Detection Test Sets”, Design Automation Conference, pp. 748-751, Anaheim, CA, 2003

[11] G. Bhargava, D. Meehl, J. Sage, “Achieving serendipitous N-detect mark-offs in Multi-Capture-Clock scan patterns”, International Test Conference, pp. 1-7, Santa Clara, CA, 2007

[12] M. A. Iyer, D. E. Long, M. Abramovici, “Surprises in Sequential Redundancy Identification”, European Design and Test Conference, pp. 88-94, Paris, France, 1996

[13] M. Damiani, G. D. Micheli, “Synchronous Logic Synthesis: Circuit Specifications and Optimization Algorithms”, IEEE International Symposium on Circuits and Systems, pp. 2566-2570, New Orleans, LA, 1990

[14] K. -T. Cheng, “On removing redundancy in sequential circuits”, Design Automation Conference, pp. 164-169, San Francisco, CA, 1991

[15] M. A. Iyer, D. E. Long, M. Abramovici, “Identifying Sequential Redundancies Without Search”, Design Automation Conference, pp. 457-462, Las Vegas, NV, 1996

[16] J. Savir, S. Patil, “On broad-side delay test”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 2 Issue 3, pp. 368-372, Piscataway, NJ, 1994

[17] M. Abramovici, P. S. Parikh, B. Mathew, D. G. Saab, "On selecting DFFs for partial reset", International Test Conference, pp. 1008-1012, Baltimore, MD, 1993