The Dissertation Committee for Jae Hoon Jeong
certifies that this is the approved version of the following dissertation:

Efficient Verification/Testing of System-on-Chip Through Fault Grading and Analog Behavioral Modeling

Committee:

Earl E. Swartzlander, Jr., Supervisor

Anthony P. Ambler, Co-Supervisor

Mircea D. Driga

Nur A. Touba

Lizy K. John

Efficient Verification/Testing of System-on-Chip Through Fault Grading and Analog Behavioral Modeling

by

Jae Hoon Jeong, B.S.; M.S.; M.S.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2013

Efficient Verification/Testing of System-on-Chip Through Fault Grading and Analog Behavioral Modeling

Publication No. _____

Jae Hoon Jeong, Ph.D.
The University of Texas at Austin, 2013

Supervisor: Earl E. Swartzlander, Jr.

This dissertation presents several cost-effective production test solutions using fault grading and mixed-signal design verification cases enabled by analog behavioral modeling. Although the latest System-on-Chip (SOC) is getting denser, faster, and more complex, the manufacturing technology is dominated by subtle defects that are introduced by small-scale technology. Thus, SOC requires more mature testing strategies. By performing various types of testing, better quality SoC can be manufactured, but test resources are too limited to accommodate all those tests. To create the most efficient production test flow, any redundant or ineffective tests need to be removed or minimized.

Chapter 3 proposes new method of test data volume reduction by combining the nonlinear property of feedback shift register (FSR) and dictionary

coding. Instead of using the nonlinear FSR for actual hardware implementation, the expanded test set by nonlinear expansion is used as the one-column test sets and provides big reduction ratio for the test data volume. The experimental results show the combined method reduced the total test data volume and increased the fault coverage. Due to the increased number of test patterns, total test time is increased.

Chapter 4 addresses a whole process of functional fault grading. Fault grading has always been a "desire-to-have" flow because it can bring up significant value for cost saving and yield analysis. However, it is very hard to perform the fault grading on the complex large scale SOC. A commercial tool called Z01X is used as a fault grading platform, and whole fault grading process is coordinated and each detailed execution is performed. Simulation-based functional fault grading identifies the quality of the given functional tests against the static faults and transition delay faults. With the structural tests and functional tests, functional fault grading can indicate the way to achieve the same test coverage by spending minimal test time. Compared to the consumed time and resource for fault grading, the contribution to the test time saving might not be acceptable as very promising, but the fault grading data can be reused for yield analysis and test flow optimization. For the final production testing, confident decisions on the functional test selection can be made based on the fault grading results.

Chapter 5 addresses the challenges of Package-on-Package (POP) testing. Because POP devices have pins on both the top and the bottom of the

package, the increased test pins require more test channels to detect packaging defects. Boundary scan chain testing is used to detect those continuity defects by relying on leakage current from the power supply. This proposed test scheme does not require direct test channels on the top pins. Based on the counting algorithm, minimal numbers of test cycles are generated, and the test achieved full test coverage for any combinations of pin-to-pin shortage defects on the top pins of the POP package. The experimental results show about 10 times increased leakage current from the shorted defect. Also, it can be expanded to multi-site testing with less test channels for high-volume production.

Fault grading is applied within different structural test categories in Chapter 6. Stuck-at faults can be considered as TDFs having infinite delay. Hence, the TDF Automatic Test Pattern Generation (ATPG) tests can detect both TDFs and stuck-at faults. By removing the stuck-at faults being detected by the given TDF ATPG tests, the tests that target stuck-at faults can be reduced, and the reduced stuck-at fault set results in fewer stuck-at ATPG patterns. The structural test time is reduced while keeping the same test coverage. This TDF grading is performed with the same ATPG tool used to generate the stuck-at and TDF ATPG tests.

To expedite the mixed-signal design verification of complex SoC, analog behavioral modeling methods and strategies are addressed in Chapter 7 and case studies for detailed verification with actual mixed-signal design are addressed in Chapter 8. Analog modeling effort can enhance verification quality

for a mixed-signal design with less turnaround time, and it enables compatible integration of the mixed-signal design cores into the SoC. The modeling process may reveal any potential design errors or incorrect testbench setup, and it results in minimizing unnecessary debugging time for quality devices.

Two mixed-signal design cases were verified by me using the analog models. A fully hierarchical digital-to-analog converter (DAC) model is implemented and silicon mismatches caused by process variation are modeled and inserted into the DAC model, and the calibration algorithm for the DAC is successfully verified by model-based simulation at the full DAC-level. When the mismatch amount is increased and exceeded the calibration capability of the DAC, the simulation results show the increased calibration error with some outliers. This verification method can identify the saturation range of the DAC and predict the yield of the devices from process variation.

A phase-locked loop (PLL) design cases were also verified by me using the analog model. Both open-loop PLL model and closed-loop PLL model cases are presented. Quick bring-up of open-loop PLL model provides low simulation overhead for widely-used PLLs in the SOC and enables early starting of design verification for the upper-level design using the PLL generated clocks. Accurate closed-loop PLL model is implemented for DCO-based PLL design, and the mixed-simulation with analog models and schematic designs enables flexible analog verification. Only focused analog design block is set to the schematic design and the rest of the analog design is replaced by the analog model. Then, this scaled-down SPICE simulation is performed about 10 times

to 100 times faster than full-scale SPICE simulation. The analog model of the focused block is compared with the scaled-down SPICE simulation result and the quality of the model is iteratively enhanced. Hence, the analog model enables both compatible integration and flexible analog design verification.

This dissertation contributes to reduce test time and to enhance test quality, and helps to set up efficient production testing flows. Depending on the size and performance of CUT, proper testing schemes can maximize the efficiency of production testing. The topics covered in this dissertation can be used in optimizing the test flow and selecting the final production tests to achieve maximum test capability. In addition, the strategies and benefits of analog behavioral modeling techniques that I implemented are presented, and actual verification cases shows the effectiveness of analog modeling for better quality SoC products.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recently, Very Large Scale Integrated (VLSI) or System-on-chip (SOC) circuits are getting denser, faster, and more complex. The manufacturing technology is getting smaller to meet the requirements. However, the manufacturing processes are still far from perfect and are dominated by subtle defects. The processes could introduce many kinds of manufacturing defects caused by process variation, spot defects, and so on. It is very costly to rely on the customer to identify if the shipped parts are functioning properly or not. Hence, it is crucial to test the parts before shipping them [3, 4].

VLSI circuits are tested by applying test patterns to the Circuit Under Test (CUT) and by comparing the response of the circuit with the good circuit response, which can be obtained by simulation.

There are two different test categories: functional tests and structural tests. Functional tests attempt to validate the CUT functions according to its functional specification. This is closely related to the functional verification problem of determining if the circuit specified by the netlist meets the functional specifications, assuming it is built correctly. Functional tests are very expensive and require detailed knowledge of the circuit behavior.

Structural tests make no direct attempt to determine if the overall functionality of the circuit is correct. Instead, they try to make sure that the circuit has been assembled correctly from low-level building blocks as specified in a structural netlist. For example, are all of the specified logic gates present, operating correctly, and connected correctly? The assumption is that if the netlist is correct, and structural testing has confirmed the correct assembly of the circuit elements, then the circuit should be functioning correctly.

One benefit of the structural testing paradigm is that test generation can focus on testing a limited number of relatively simple circuit elements rather than having to deal with an exponentially exploding multiplicity of functional states and state transitions. Although the task of testing a single logic gate at a time sounds simple, there is an obstacle to overcome. For highly complex designs, most gates are deeply embedded, whereas the test equipment is only connected to the primary input/outputs and/or some limited number of physical test points. The embedded gates, hence, must be manipulated through intervening layers of logic. If the intervening logic contains state elements, then the issue of an exponentially exploding state space and state transition sequencing creates an unsolvable problem for test generation. To simplify test generation, Design For Test (DFT) addresses the accessibility problem by removing the need for complicated state transition sequences when trying to control and/or observe what is happening at some internal circuit element. Depending on the DFT choices made during circuit design/implementation, the generation of structural tests for complex logic

circuits can be more or less automated.

The final production testing needs to achieve a high level of test coverage. With structural tests, it is not feasible to achieve 100% coverage for complex designs. The test coverage can be increased as much as possible by running the test generation tool for a long time, but this will increase the total test time, and it may seriously affect the test budget. Also, after passing the saturation point between pattern counts and test coverage, the efficiency of the last few patterns could be very low.

In the production testing flow, critical states or functions of the CUT need to be tested by some specific functional tests, such as memory Built-in Self Test (BIST), maximum frequency tests, and so on. These tests are the critical ones that cannot be skipped in the final production testing. There is always some coverage overlap between the functional tests and the structural tests.

To increase the testability of the CUT, there are some testing costs and parameters to be considered, such as amount of test data, test application time, area overhead, testing power, design effort, and fault coverage. For cost efficiency, those testing costs need to be reduced, but it is not possible to satisfy all these cost parameters at the same time. So there must be some trade-offs among them. Figure 1.1 shows one example of test cost parameters for a testing scheme. The bigger the gray region becomes, the better its efficiency.

Hence, one key objective of DFT methods is to allow designers to make

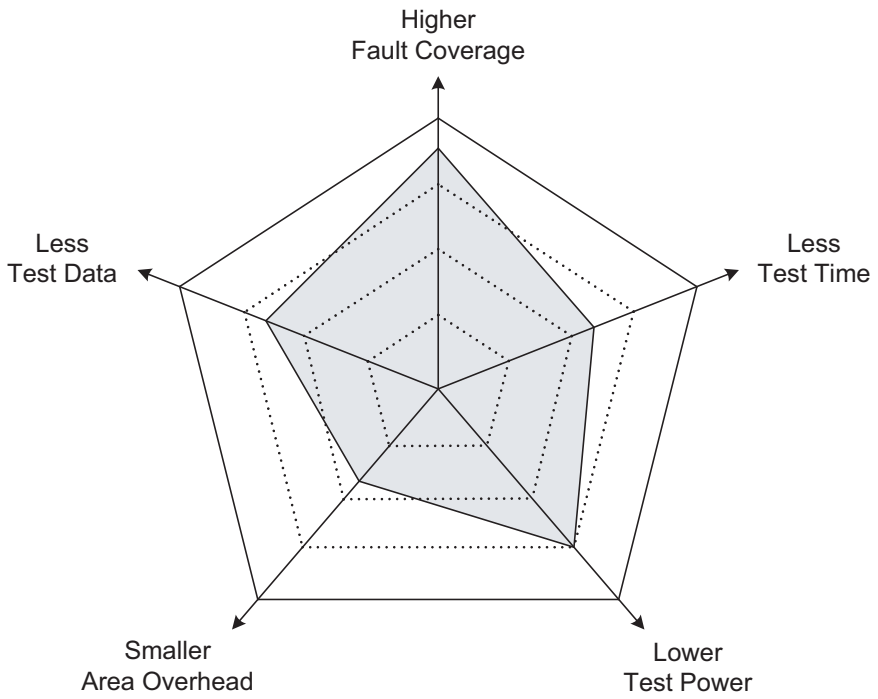

Figure 1.1: Parameters for Test Efficiency

trade-offs between the amount and the type of DFT and the cost/benefit (time, effort, and quality) of the test generation task. To optimize the efficiency of the production tests, the assessment of redundancy within test flow is very important.

1.1 Summary of Chapters in this Dissertation

This dissertation addresses the following cost-effective strategies for production testing.

- **Reduction of test data volume through nonlinear feedback shift**

register by dictionary coding

A test set expansion scheme by nonlinear feedback shift register (NFSR) is applied to reduce test data volume. All bits of the only first test pattern and the leftmost bit of all subsequent patterns from the expanded test set into the test memory. In addition, when the patterns are used for testing, the fully expanded test set is regenerated by the relationship between the patterns. This scheme uses the property of NFSR and eliminates the hardware of the NFSR. Also, the scheme needs only a small amount of memory to store the portion of the expanded test set. Furthermore, a compression scheme can be used for further reduction. A dictionary coding is applied, and this combined scheme provides big compression rate. Chapter 3 describes the detail of test data reduction by the property of NFSR and dictionary coding and experimental results.

- **Efficient Functional Test Selection by Functional Fault Grading**

  Functional testing can be very expensive and time consuming depending on the functional specification. By reducing the redundancy in test coverage between structural tests and functional tests, efficient functional tests can be selected to achieve high test coverage while reducing the total test time. To assess the redundancy, functional fault grading was used with a commercial tool. Chapter 4 provides the detail of the functional fault grading process and experimental results.

- **Contactless Leakage Test on Package-on-Package Devices**

Package-on-Package (POP) devices have Ball Grid Array (BGA) pins on both the top and the bottom of the package. The increased number of pins makes the POP testing challenging because more automatic test equipment (ATE) channels are required. The proposed testing method uses internal boundary scan cells and leakage tests to detect potential packaging defects on the top pins without having physical ATE channels. A special algorithm reduces the total test time to achieve full coverage. Also, this method can be expanded into both top and bottom pins for multi-site testing. Chapter 5 describes how the boundary scan cells work for leakage tests to screen POP package defects.

- **Reduction of Structural Test Time Through Transition Delay Fault Grading**

  Chapter 6 addresses fault grading within structural tests, whereas Chapter 4 covers fault grading to reduce the overlap between structural tests and functional tests. Two common types of faults detected by structural tests are stuck-at faults and Transition Delay Faults (TDFs). Stuck-at faults can be considered as TDFs with infinite delay. Hence, TDF test patterns can detect both TDFs and stuck-at faults. Using the TDF vectors first can reduce the total number of static undetected faults, and the reduced set of the static faults results in reduced pattern counts for the static Automatic Test Pattern Generation (ATPG) tests. This TDF grading process was performed within the ATPG tool that generated the static ATPG vectors and the TDF vectors. Experimental results show

that the test time can be reduced while keeping the same test coverage.

- **Analog Behavioral Modeling: Strategies and Methodologies**

  The analog behavioral model for mixed-signal analog cores can address the simulation speed limitation in analog simulations and the interoperability of mixed verification with analog and digital cores. In addition, the validity of the model-based simulation is contingent on the quality of behavioral models used. Depending on the level of abstraction for the behavioral modeling, there are some limitations for accurate performance analysis. Performance analysis can be covered by block-level analog simulations. In analog behavioral modeling, electrical signals such as voltage and current need to be represented as real numbers to achieve enough level of accuracy and corresponding functionality for real analog signal propagation. The interface in the analog model passes those real numbers between blocks and mimic actual signal transfer and propagation. Chapter 7 describes the strategies and methodologies of analog behavioral modeling.

- **Mixed-Signal Design Verification by Analog Behavioral Model**

  Design verification is a necessary process to ensure a quality design. Usually, digital design is accompanied with well-defined verification plan and thoroughly verified from multiple design stages. However, the verification of Mixed-signal design requires long-taking transistor-level simulations or repetitive Monte Carlo simulations. Within a given short time

for product market, it is not feasible to cover all verification aspects by those analog simulations. Whereas those analog simulation methods can cover performance-related verifications with long turnaround time, analog behavioral model can achieve fast simulations for functional verifications. By virtue of fast simulation time with analog behavioral model, such functional verification may uncover potential design bugs by applying comprehensive test cases with quick turnaround time. Chapter 8 provided the detail of mixed-signal design verification enabled by analog behavioral modeling.

Chapter 2

Background

The manufacturing test costs of System-on-Chip (SOC) threaten to increase beyond what is acceptable, if proper countermeasures are not considered. Factors that drive the production test costs up are the increases in pin count, test data volume, operation speed of SOC, and the corresponding required automatic test equipment (ATE) accuracy. In particular, the test data volume has increased dramatically because of a combination of growth in transistor count and more advanced test methods such as delay fault testing, which requires significant test resources. As a consequence, the testing of complex industry designs requires expensive ATE with a large channel count and deep test vector memory and significant test time.

There are many different approaches to reduce the test costs. By implementing a built-in self test (BIST), the SOCs themselves may eliminate the need for ATE. A BIST for embedded memories is an inevitable choice for mature production testing. However, a logic BIST is expensive to implement inside the chip, and hence its usage is typically limited to small scale designs or portions of a large design. Because all the required tests cannot be achieved by the memory BIST and the logic BIST, ATE is still required for the production

testing.

Then, each test itself can be optimized by using test data compression or compaction, but these techniques still require the presence of an ATE. The demands on both vector memory and test application time can be reduced by exploiting the many "don't care" bits in the test set [5, 6].

Another effective approach to reduce test cost is multi-site testing, in which multiple instances of the same SOC are tested in parallel on a single ATE. More sites mean more devices are tested in parallel. Multi-site testing amortizes the fixed ATE costs over multiple SOCs. High volume production testing always requires multi-site testing because the saving in test cost is maximized.

Efficient multi-site testing requires the effective management of test resources such as the number and depth of ATE channels and the on-chip DFT, while taking into account parameters such as test time, test vector memory, contact yield, etc. One way to allow an increase in the number of sites is to increase the number of ATE channels. However, this solution not only brings substantial extra costs, but also is not scalable to SOCs with high pin counts. The other way to increase the number of sites is to narrow down the test interface, i.e., the number of SOC terminals that need to be contacted during testing. Limited pin count testing [7, 8, 9] is a well-known DFT technique that does this.

Instead of reducing the direct test cost from each test, the quality of

each test can be calculated, and the test coverage overlap between different test categories can be assessed by performing fault grading. Fault grading is a procedure that rates testability by relating the number of fabrication defects that can be detected with the test vector set under consideration to the total number of conceivable faults [10]. It is used for refining both the test circuitry and the test patterns iteratively, until satisfactory fault coverage is obtained. Fault grading can be performed through fault simulation. Fault grading estimates the quality of each given test by the number of detected faults. Based on the fault grading result, the priority of the given tests can be assessed.

The most important point in the production testing is that there should be no test hole or test escape by deploying false-pass tests or by skipping critical tests with no reason. As stated previously, many different types of test optimization can be done, but the test quality should be maintained always.

Analog behavioral modeling refers to the substitution of more abstract, less computationally intensive circuit models for lower level descriptions of analog functions [11, 12, 13]. These simpler models emulate the transfer characteristics of the circuit elements. Verification simulations can be performed to verify the analog circuits. Analog SPICE simulators are much slower than digital simulators and are slower still when compared with emulators and hardware accelerators. To tackle simulation-throughput issues, the digital simulation can be performed by analog behavioral models. Various aspects of verification of mixed-signal cores can be partitioned into different simulation platforms. Per-

formance oriented verification can be performed by analog SPICE simulations although total simulation time is much longer than digital simulations. However, digital simulations using analog behavioral models can be performed to check the full functionality of the whole core and detailed operational features with quick turnaround time. After the detailed functionalities are verified first, and then the fine-tuning for analog parameters can be verified via analog simulations. These combined efforts will enhance the quality of the device and increase the efficiency of all development process and engineering time.

Chapter 3

Reduction of Test Data Volume through Nonlinear Feedback Shift Register by Dictionary Coding

Since System-on-Chip (SOC) devices are getting bigger and more complex, larger test data are required to make sure the quality of the SOC devices. This increased test data volume requires more test memory and channels, and affects testing time. This chapter describes how large test data can be handled efficiently by combining nonlinear feedback shift register (NFSR) and dictionary coding. [1]

Daehn *et al.* [1] proposed an expanded test set generated from a deterministic test set. This scheme used NFSR to generate the expanded test set for the combinational circuit under test(CUT). Hamzaoglu *et al.* [5] developed new compaction algorithms such as redundant vector elimination, essential fault reduction to reduce a given fault set. Jas *et al.* [6] introduced a statistical compression and decompression coding scheme considering testing clock, size of codeword, and multiple cores. Li *et al.* [15] proposed a dictionary coding in order to reduce test data volume. The proposed dictionary coding

[1]This chapter is based on [14].

in this chapter is developed mostly based on [6, 15]. Sun *et al.* [16] proposed a test strategy that combines a dictionary coding and a reseeding technique to reduce test data volume . Chandra *et al.* [17] used an alternating run-length code to compress test data volume. This method was proven to decrease test data volume, test application time, and power dissipation. Krishna *et al.* [18] introduced the partial dynamic linear feedback shift register reseeding method to reduce the test set size and to generate test vectors in fewer cycles. Alyamani *et al.* [19] proposed the seed ordering algorithm to decrease the required number of the seeds.

A proposed method in this chapter uses a test set expansion scheme by NFSR as described in [1]. However, the proposed scheme stores all bits of the only first test pattern and the leftmost bit of all subsequent patterns from the expanded test set into the test memory [14]. In addition, when the patterns are used for testing, the fully expanded test set is regenerated by the relationship between the patterns. This scheme uses the property of NFSR and eliminates the hardware of the NFSR. Also, the scheme needs only a small amount of memory to store the portion of the expanded test set. Furthermore, a compression scheme can be used for further reduction. A dictionary coding is applied and this combined scheme provides big compression rate [14].

3.1 Nonlinear Feedback Shift Register

This section describes how an expanded test set is generated from a deterministic test set, and the limitations of the algorithm proposed by Daehn

et al. [1]. The expanded test set generated from a deterministic test set has larger volume than the deterministic test set. Because an expanded set can be generated by the NFSR, the benefit of NFSR is to save the test data volume in [1]. However, the NFSR hardware itself is an area overhead to the chip. The proposed scheme does not require the NFSR hardware, and it stores much less amount of test data from the expanded test set.

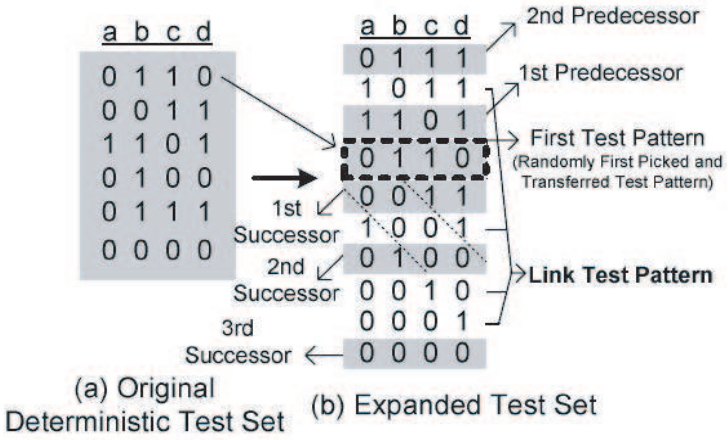

Figure 3.1: Creation of Expanded Test Set [1]

As in Figure 3.1, the six deterministic test patterns are expanded into a test set of 10 test patterns with four newly inserted test patterns. In Figure 3.1(b), the newly inserted test patterns are called "*link test patterns*." The required number of link test patterns is determined by the successor or predecessor that comes. The K_{th} successor and the K_{th} predecessor need the *K-1* link test pattern(s). The first test pattern, 0110 in Figure3.1, is randomly picked from the original test set and transferred into the expanded test set.

Successors indicate the test patterns placed after the first test pattern. On the other hand, Predecessors are placed before the first test pattern in the expanded test set. In order for any test pattern in the original deterministic test set to be transferred into the expanded test set, the following conditions should be met.

- K_{th} successor: A test pattern in the deterministic test set whose last consecutive (S - K) bit(s) are equal to the first consecutive (S - K) bit(s) of the last test pattern in the expanded test set. (S = size of test pattern)
- K_{th} predecessor: A test pattern in the deterministic test set whose first consecutive (S - K) bit(s) are equal to the last consecutive (S - K) bit(s) of the first test pattern in the expanded test set.

When any test pattern in the deterministic test set meets the above-mentioned conditions, the test pattern can be transferred to the expanded test set. For example, the last two bits of the second successor, 0100 in Figure 3.1b, are equal to the first two bits of the last test pattern, 0011 in Figure 3.1b. In addition, a link test pattern 1001 is to be filled between them to make a consistency among these three test patterns in the expanded test set. This example is marked with two slant dashed line in Figure 3.1b. Link test patterns, successors, and predecessors are used to make the consistency that a test pattern in the expanded test set can be obtained by shifting its preceding test pattern to the right by one bit and by inserting its leftmost bit into the

empty leftmost bit of its preceding test pattern. For example, the second test pattern 1001 in Figure 3.1b is obtained by shifting the first test pattern 0011 to the right by 1 bit and by inserting 1 to the leftmost empty bit in the second test pattern. Note that successors come without a fixed order of appearance, and any K_{th} successor may appear multiple times in the expanded test set. The operations for Predecessors are the same except the direction of shifting. The detailed process for creation of the expanded test set is presented in the next section.

To generate this expanded test set, the required hardware can be implemented by NFSR. First, the feedback function of each test pattern in the expanded test set can be determined by the leftmost bit to be filled in the next test pattern, and then the Boolean feedback function can be extracted by the Karnaugh map to satisfy all feedback functions. This Boolean feedback function is used to figure out what kind of primitive gates are needed in the feedback loop of NFSR.

In Figure 3.2a, the feedback function for each test pattern has the leftmost bit of the next test pattern as its resultant value. If there is no next test pattern, the result of the feedback function is a "don't care" state. Based on these feedback functions, the Karnaugh map can be built, as shown in 3.2b. With this Karnaugh map, the corresponding Boolean feedback function is evaluated, and then the NFSR hardware for test pattern generator is created, as shown in Figure 3.3. This test pattern generator is nonlinear because a primitive gate like an AND gate is used in its feedback loop, whereas the

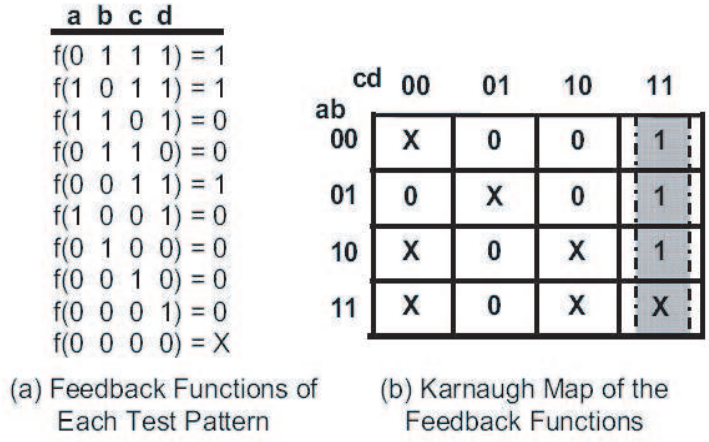

Figure 3.2: Feedback Functions of Each Test Pattern and its Karnaugh Map

linear feedback shift register uses only XOR gates in feedback loops.

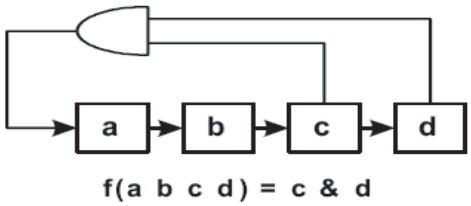

Figure 3.3: NFSR to generate an expanded test set

The generation of the expanded test set and the logic implementation for the NFSR look simple, as described previously. However, the actual implementation of NFSR has some limitations. If the given design is very large, the process identifying the feedback function for NFSR is very complex. Furthermore, although the feedback function is known, the hardware implementation requires a large area overhead because many primitive gates are usually re-

quired in the feedback loop. Hence, the above-mentioned scheme of NFSR is limited to small or simple circuits [1]. Therefore, a new efficient scheme is required to overcome these limitations. A proposed scheme is described in the next section.

3.2 Proposed Use of Nonlinear Properties of Feedback Shift Register

By minimizing the number of link test patterns, the size of expanded test set can be reduced. In addition, this would result in the reduction of test data volume. The following steps are the details of the proposed scheme.

- **Step 1:** Pick any test pattern from the deterministic test set and then transfer it into the expanded test set, which is originally empty.
- **Step 2:** Set index K to 1.
- **Step 3:** Find the K_{th} successor of the last pattern in the expanded test set from the deterministic test set. If this is found, transfer the successor into the expanded test set. This K_{th} successor becomes the new last test pattern in the expanded test set and the *K-1* link test patterns would be placed between the previous last pattern and the new last test pattern. Keep searching another K_{th} successor for a new last test pattern. If there is no matching successor, move to Step 4.
- **Step 4:** Find the K_{th} predecessor of the first pattern in the expanded test set from the deterministic test set. If this is found, transfer the

predecessor into the expanded test set. This K_{th} predecessor becomes the new first test pattern in the expanded test set and *K-1* link test patterns would be placed between the previous first pattern and the new first test pattern. Keep searching another K_{th} predecessor for a new first test pattern. If there is no matching predecessor, move to Step 5.

- **Step 5:** Increment K by 1, if neither the K_{th} successors nor the K_{th} predecessors are found from both Step 3 and Step 4, and then go back to Step 3. If any K_{th} successor or predecessor is found in either Step 3 or 4, go back to Step 2. When K reaches the "size of test pattern" or the deterministic test set is empty, finish the process.

Note that the "don't care" bits in the deterministic test set are flexibly used to minimize the required number of link test patterns in the expanded test set. For instance, when the test pattern 10110 finds its first successor, if a test pattern satisfying its condition is only 110X1 in the original deterministic test set, the "don't care" bit can be interpreted as 1. This approach decreases the number of newly created link test patterns. In some cases, some of the "don't care" bits may remain unaltered. If the original deterministic test set in Figure 3.1a is stored in the memory, it requires a 24-bit storage, which is obtained by multiplying the number of columns and the number of bits in one test pattern. Because an expanded test set is bigger than its deterministic test set, the expanded test set would require more memory than the deterministic test set. However, all bits of the expanded test set do not need to be stored.

As Figure 3.4 shows, only the first test pattern and the bits in the leftmost column of the expanded test set may be stored, and the whole expanded test set can be reproduced by a shift register. This first test pattern and the bits in the leftmost column of the expanded test set are called *one-column test set*. The first test pattern is serially fed into the shift registers, and then the following test pattern can be simply generated from the shift registers. This process can be implemented with properly designed test architecture. This pattern generation scheme can be used with either the external tester or the internal testing memory.

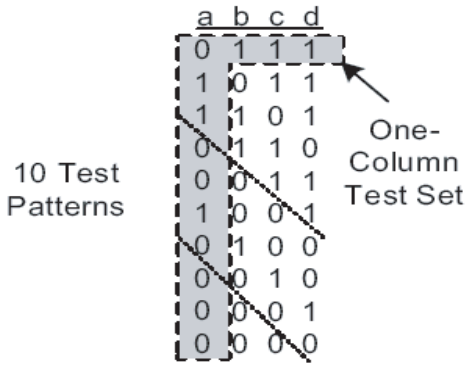

Figure 3.4: One-column test set by the proposed scheme of nonlinearity

This one-column test set reduces the required pattern bits from 24 bits to 13 bits, which is obtained by the following equation: Number of Test Pattern + (Size of Test Pattern - 1). This is a significant saving in the test set volume. Other than this, there is more room for further reduction to the one-column test set if the order of the bits in the one-column test set is kept. The next

section describes additional reduction in test set volume using a dictionary coding.

3.3 Test Architecture with Dictionary Coding

For further reduction of test data volume in the one-column test set, a dictionary coding is chosen because it provides a good compression ratio to the test set having fewer "don't care" bits [16]. Most of the work in this section is developed based on the scheme of Jas *et al.* [6] and Li *et al.* [15]. In a dictionary coding, a 1-bit prefix precedes each word regardless of compression. This prefix tells whether a word is compressed or not. The size of a codeword is determined by how many original words can be saved in the dictionary decoder memory. For example, if the decoder memory has space for only eight words regardless of their size, a 3-bit binary code can represent eight memory storage spaces. This 3-bit code is the index that comes after the prefix when a word is compressed. In this case, the size of the smallest codeword is 4 bits (i.e., 1-bit for the prefix and 3-bits for the index). When the decoder reads the prefix indicating a compressed word, it has to decode the 3-bit index that selects one of the eight memory spaces and then extracts the original word from memory. When a word is not compressed, the size of the codeword is "$m+1$" bits, where m is the size of an original word. If the decoder reads the prefix indicating an uncompressed word, then it reads the m bit uncompressed word after the prefix. This m bit uncompressed word moves directly into the m bit serializer without checking the memory. In a dictionary coding, a few critical parameters

such as the size of the original word and the appearance frequency of a word should be carefully determined. Those values cannot be determined simply because all the bits are deeply related each other. Therefore, it is an NP-hard (Non-deterministic Polynomial-time hard) problem to find an optimal solution to meet the requirements of all those parameters. Hence, a heuristic method should be used.

As mentioned by Jas *et al.* [6] and Li *et al.* [15], the tester clock for decoder and the system clock should be considered. The proposed scheme considers the case that the tester clock is slower than the system clock.

Figure 3.5 describes the test architecture for the proposed scheme with either external tester or internal test memories such as read only memory (ROM). In this architecture, the decoder is mainly related to the dictionary coding. Thus, if only the scheme of nonlinearity is applied, it requires only the shift registers, the multiple input shift register (MISR), and the memory in the external tester or inside the circuit under test (CUT). This scheme requires the same clocking scheme from the external tester or the internally generated clocks for the shift registers.

The compressed one-column test set is stored in the memory of the external tester or internal test memory. Each codeword is fed into the dictionary decoder serially, and the decoder reads the prefix first. When the prefix is 1, indicating the word is compressed, the decoder reads the index after the prefix, and then sends the word from the memory of the decoder to the m bit serializer. The serializer takes the m bit word, and the m bit word is shifted

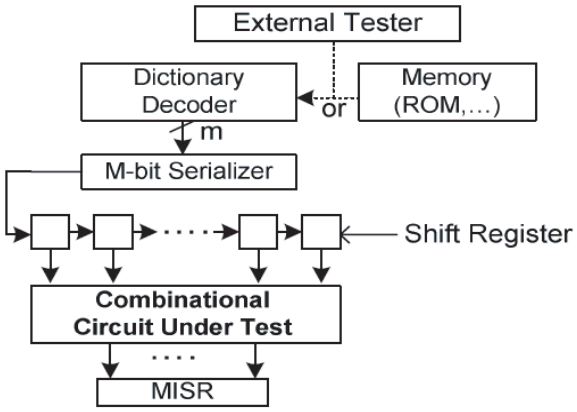

Figure 3.5: Proposed test architecture to implement the proposed scheme

into the shift registers. At every shift clock cycle, the test patterns in the shift register are applied to the CUT, and its response is captured into MISR. When the prefix is 0, the decoder simply takes an "*m+1*" bit uncompressed codeword from memory. Then, these m bit uncompressed codeword is fed into the m bit serializer. The rest of the process is the same as in the compressed word. As mentioned previously, the proposed scheme considers the case that the test clock for the decoder is slower than the system clock. In this clock system, the compressed word runs well because less operation time is required in a slow decoder clock. However, the uncompressed word is a little different. The system clock should stall until the decoder finishes its feed of the uncompressed m bit word into the serializer because the decoder spends more time with the uncompressed word [6].

3.4 Experimental Results

The most deterministic test sets for ISCAS '85 are designed to have fewer test patterns than the required amount to show the increase of fault coverage by the scheme of nonlinearity. Conversely, the test sets for ISCAS '89 are highly compacted and have almost the required number of test patterns for the desired fault coverage. Table 3.1 shows the original test data of the benchmark circuits. In the last column of Table 3.2, most of the fault coverage of ISCAS '85 is increased because of a greater number of test patterns in the expanded test set, but the fault coverage of all ISCAS '89 benchmark circuits is not increased because all undetected faults by original test set might be a random resistant fault. When implementing the dictionary coding, the program code is developed to maximize the slant in the distribution of the frequency of each word's appearance because the sharp slant gives a better compression. The word sizes from 5 to 10 bits are simulated to find the best compression rate with eight memory spaces fixed in the decoder memory. As the size of the original word increases, more memory space is required, although the number of decoder memory spaces is fixed. Note that the effect of increasing memory is not considered in computing the results. In some case, increasing the word size does not improve the reduction. This case might impact the memory requirement.

In Table 3.1, the fifth column shows the number of "don't care" bits in the test set. These numbers for ISCAS '85 are not high because the compression ratio for ISCAS '85 came out as low. Table 3.2 shows the improved data

Table 3.1: Original test data of benchmark circuits

Benchmark circuit	**# of PI* (PI+FF*)**	**# of TP***	**Original *TDV×ϒ**	**# of don't-care bits (%)**	**FC* (%)**
c432	36	85	3,060	1,683 (55)	81.00
c499	41	286	11,726	2,847 (24.3)	39.50
c1908	33	1,128	37,224	12,453 (33.5)	96.80
c2670	233	700	163,100	107,836 (66.1)	97.50
c7552	207	5	1,035	0 (0)	44.10
s9234	247	147	36,309	27,632 (76.0)	93.45
s13207	700	239	167,300	157,018 (93.8)	98.46
s15850	611	120	73,320	62,603 (85.4)	96.68
s38417	1,664	95	158,080	124,896 (79.0)	99.47
s38584	1,464	131	191,784	162,174 (84.6)	95.85

*: (PI: Primary Input, FF: Flip-Flop, TP: Test Pattern, TDV: Test Data Volume, FC: Fault Coverage)
ϒ: is obtained by multiplying the number of PI and TP

Table 3.2: Improved test data by nonlinear property of feedback shift register

Benchmark circuit	**New # of TP***	**New # of don't-care bits**	**New TDV* of the Nonlinearity Ψ**	**Improvement to Original TDV* (%)**	**New FC* (%)**
c432	1,472	526	1,507	50.75	95.00
c499	7,432	818	7,472	36.27	64.40
c1908	25,249	4,092	25,281	32.08	99.20
c2670	68,860	15,871	69,092	57.64	97.50
c7552	812	0	1,018	1.64	93.40
s9234	13,422	5,626	13,668	62.36	93.46
s13207	16,930	8,712	17,629	89.46	98.46
s15850	20,549	11,006	21,159	71.14	96.68
s38417	101,883	70,827	103,546	34.50	99.47
s38584	74,275	46,937	75,738	60.51	95.85

*: (PI: Primary Input, FF: Flip-Flop, TP: Test Pattern, TDV: Test Data Volume, FC: Fault Coverage)
Ψ: is obtained by adding the number of new TP and the number of (PI-1)

by applying only the scheme of nonlinearity. The second column indicates the number of new test patterns in the expanded test set. These increased test patterns guarantee an improvement in fault coverage when the faults detectable by random test patterns remain undetected. The improvement in fault coverage is shown in the last column of Table 3.2. Based on the expanded test

Table 3.3: Additional improved test data by dictionary coding

Benchmark circuit	TDV from Only Dictionary Coding	TDV from the Dictionary in [12]	TDV from Nonlinearity+ Dictionary	Total Improvement (%)
c432	1,938 (10)	2,072 (8)	**1,170 (9)**	**61.76**
c499	9,764 (7)	9,479 (10)	**6,834 (9)**	**41.72**
c1908	31,728 (6)	29,213 (10)	**23,364 (8)**	**37.23**
c2670	88,543 (10)	85,232 (10)	**50,922 (9)**	**68.78**
c7552	754 (9)	791 (10)	**756 (7)**	**26.96**
s9234	20,397 (10)	20,335 (10)	**11,191 (8)**	**69.18**
s13207	73,906 (10)	73,402 (10)	**13,345 (10)**	**92.02**
s15850	36,125 (10)	36,543 (10)	**15,320 (9)**	**79.11**
s38417	85,625 (10)	87,827 (10)	**65,710 (10)**	**58.43**
s38584	98,976 (10)	102,214 (10)	**53,361 (10)**	**72.18**

set, the size of the one-column test set is shown in the fourth column. The improved percentage of the one-column test set in the fifth column ranges from 32.08% to 89.46% except for the c7552. Because this c7552 benchmark circuit uses only five test patterns with no "don't care" bit, the reduction ratio is pretty low. However, the scheme of nonlinearity gives the significantly increased fault coverage for c7552 with almost the same test volume used. As mentioned previously, the reduced amount of test volume is directly proportional to the number of "don't care" bits. For instance, c432 having 55% of "don't care" bits shows 50.75% reduction. Thus, more "don't care" bits in the test set guarantee more enhancements in the reduction of test volume.

Table 3.3 shows the data for further reduction by applying dictionary coding and the cases with only dictionary coding by Li *et al.* [15]. The second column indicates the improved data when only dictionary coding is applied. The numbers in parentheses indicate the size of the original word that grants

the best reduction ratio without considering the increased required memory. In the simulation of only the dictionary coding, a "don't care" bit is simply converted to the value 0, and then the frequency of each word's appearance is counted. The result from the sole application of the dictionary coding is fairly good but much less than the improved result in the fourth column, where both the scheme of nonlinearity and the dictionary coding are applied together. The third column shows the improved result from the dictionary coding used by Li *et al.* [15]. Its results are as close as those of the sole application of the dictionary coding. One interesting point is that most of the compared data are worse than that of sole application of the scheme of nonlinearity. It means that the scheme of nonlinearity is good enough alone. The combined application of the scheme of nonlinearity and of dictionary coding provides the best compression ratio, which is listed in the fifth column of Table 3.3. However, there is no significant effect to the scheme of nonlinearity with the additional application of the dictionary coding because most of the compression has already been extracted by using the scheme of nonlinearity.

As all results indicate, both the scheme of nonlinearity and the mix of two schemes offer better performance compared with the sole application of dictionary coding or the dictionary coding used by Li *et al.* [15]. Moreover, the nonlinearity alone has a good capability to decrease the test set volume. The proposed schemes obviously show a considerable reduction in the test data volume.

Chapter 4

Effective Functional Test Selection by Functional Fault Grading

The purpose of system-on-chip (SoC) production testing is to make sure that all the manufactured devices are working correctly as a valid product. If all combinations of physical defects in the chip are tested and screened, the screened SoC devices are guaranteed not to have any manufacturing defects.

Very large-scale integrated (VLSI) circuits are tested by applying test patterns to the circuit under test (CUT) and by comparing the response of the circuit with the good circuit response, which can be obtained by simulation. There are two categories of tests: functional tests and structural tests. Because SoC devices are getting bigger and more complex, the production test flow needs to make sure there is no test hole for the product. In the meantime, test cost needs to be minimized with efficient test selections and Design For Test (DFT) choices.

4.1 Functional Tests vs. Structural Tests

Functional tests attempt to validate the CUT functions according to its functional specification. This is closely related to the functional verifica-

tion problem of determining if the circuit specified by the netlist meets the functional specifications, assuming it is built correctly. Functional tests are very expensive and require deep knowledge of the detailed circuit behavior.

Structural tests make no direct attempt to determine if the overall functionality of the circuit is correct. Instead, they try to make sure that the circuit has been assembled correctly from low-level building blocks as specified in a structural netlist. For example, are all specified logic gates present, operating correctly, and connected correctly? The assumption is that if the netlist is correct and structural testing has confirmed the correct assembly of the circuit elements, then the circuit should be functioning correctly.

One benefit of the structural testing paradigm is that test generation can focus on testing a limited number of relatively simple circuit elements rather than having to deal with an exponentially exploding multiplicity of functional states and state transitions. Although the task of testing a single logic gate at a time seems simple, there is an obstacle to overcome. For highly complex designs, most gates are deeply embedded whereas the Automatic Test Equipment (ATE) is only connected to the primary input/outputs and/or some limited number of physical test points. Hence, the embedded gates must be manipulated through intervening layers of logic. If the intervening logic contains state elements, then the issue of an exponentially exploding state space and state transition sequencing creates an unsolvable problem for test generation. To simplify test generation, the DFT process addresses the accessibility problem by removing the need for complicated state transition

sequences when trying to control and/or observe what is happening at some internal circuit element. Depending on the DFT choices made during circuit design/implementation, the generation of structural tests for complex logic circuits can be more or less automated.

The final production testing needs to achieve a high level of test coverage. With the structural tests, it is not feasible to achieve 100% test coverage for complex designs. The test coverage can be increased as much as possible by running the test generation tool for a long time, but this will increase the total test time and it may seriously affect test budget. Also, after passing the saturation point between pattern counts and test coverage, the efficiency of the last few patterns could be very low.

In the production testing flow, critical states or functions of the CUT need to be tested by some specific functional tests such as memory built-in self test (BIST), maximum frequency tests, and so on. These tests are the critical ones, which cannot be skipped in the final production testing. There is always some coverage overlap between the functional tests and the structural tests.

4.2 Functional Fault Grading

Fault grading is a procedure that rates testability by relating the number of fabrication defects that can in fact be detected with a test vector set under consideration to the total number of conceivable faults [10]. It is used for refining both the test circuitry and the test patterns iteratively, until satisfactory fault coverage is obtained.

The fault grading can be performed through fault simulation. The fault grading estimates the quality of each given test by the number of detected faults. Based on the fault grading result, the priority of the given tests can be assessed.

The structural test generation is performed by an automatic test pattern generation (ATPG) process, which is based on pseudo-random pattern generation technique. Hence, there are some random-pattern-resistant faults that prevent the perfect coverage of the ATPG process. The faults that are undetected by the ATPG process should be targeted by different types of test vectors. Functional tests are good complimentary methods, but these functional tests are very expensive in terms of test cost. Hence, all of them cannot be used in the production flow because of limited test budgets.

Some functional tests may contribute much coverage for the undetected faults from the ATPG process, but others may target faults that are already detected. Hence, the fault grading process screens out those redundant functional tests to not waste test resources.

Functional fault grading has been considered as a very significant process to assess the value of each functional test and to optimize a production testing flow, but the required effort is too intense and time consuming. Developing a new sophisticated fault grading tool is beyond the scope of this work. However, the selection of a proper commercial tool fully capable for fault grading and thoughtful arrangement of the required inputs and smart empirical decision can speed up this whole fault grading process.

In this chapter, the purpose of fault grading is to find out the best suite of functional tests based on their grade. Post-ATPG undetected faults will be targeted by functional vectors.

4.3 Comparable Works

As noted earlier, this fault grading process was performed using software-based fault simulation. Hence, the total fault grading time is dependent on the size and complexity of the target SoC, but this fault grading can be performed once, and the optimized test flow can be used for production.

There is one comparable work presenting hardware-based fault emulation [20]. A Field-Programmable Gate Array (FPGA) was used to replicate the target SoC. Because all the time-consuming parts are in software-based fault grading, the overall processing time is very quick. Unfortunately, for a big industrial design, multiple FPGAs need to be placed and connected in a big emulation board. Also, the synthesized FPGA netlist is not equivalent to the actual gate-level netlist of the SoC. Finally, delay values may vary over different FPGAs through their boundaries. Hence, passing the functional test on the emulation board may fail on silicon because of those mismatches. It is not a desirable solution to decide the quality of production tests by using the nonequivalent netlists.

4.4 Functional Fault Grading Setup

Functional fault grading tools need to handle very sophisticated computing processes. Instead of developing the tool itself, a mature commercial tool was used in this study.

4.4.1 The Fault Grading Tool

Fault grading is a very intense computing simulation process. Various tools from multiple different vendors were investigated for this functional fault grading effort. Most failing tools were not able to provide full compatibility for the input functional tests and show an optimized simulation speed. After reviewing several commercial tools in the market, Z01X by WinterLogic was chosen, which represents four possible states in digital simulation. Z01X is a postsynthesis Verilog simulator for evaluating the effectiveness of manufacturing tests in chip and IC production [21]. Z01X uses traditional stuck-at fault models and transition fault models required for the detection of deep submicron manufacturing defects characterized by slow-to-rise and slow-to-fall transition delays in gates and wires. Z01X supports parallely distributed simulation, which is performed using one master process and multiple slave processes to reduce the total simulation time and to maximize the utilization of computing resource. This tool is fully scalable. Hence, the more slaves, the faster simulation can be achieved. Also, there is a companion tool called *Fault Manager*, which efficiently manages all the given faults with simulation results and quality factors.

4.4.2 Targeting Fault Categories

For the ATPG process, a commercial ATPG tool from Synopsys was used. A tool called *Tetramax* marks each fault's class to show its detection status after the completion of ATPG. Each fault was categorized as "detected," "possibly detected," "undetectable," "ATPG untestable," or "not detected" [22].

As a target for functional fault grading, the setup picked up the faults marked as "possibly detected," "not detected," and "ATPG untestable". The "possibly detected" faults are the faults not deterministically detected by the ATPG process. This fault category's credit is 50%, whereas the 'detected' faults have 100% credit for test coverage. Hence, by detecting them with functional tests, test coverage could become better.

The "undetectable" faults are the ones that cannot be detected (either hard or possible) under any conditions. When calculating test coverage, these faults are not considered because they have no logical effect on the circuit behavior and cannot cause failures. Hence, these faults are not included in the targeting faults. The "not detected" or "ATPG untestable" faults are good candidates for functional fault grading.

4.4.3 Simulation Inputs

Because the functional fault grading process handles the structural fault list, gate-level netlists are required. In addition, other simulation-specific inputs such as Read Only Memory (ROM) images or memory access interfaces

need to be prepared. These memory setups will affect the initial behavior of the design. To accelerate the fault grading process, the user-defined primitive (UDP) libraries are optimized to not require too much computation. By removing redundant hierarchies and collapsing multiple instances into a single UDP table, the total number of design units is reduced significantly. Also, the Z01X tool reads in the functional vectors in the waveform generation language (WGL) format, which is an industrial standard test format. Hence, all the functional vectors dumped from simulation are translated by translation engine into the WGL format. For transition delay fault (TDF) grading, the clock frequency of each frequency domain is prepared as a value change dump (VCD). Figure 4.1 shows the detail of the simulation inputs for the functional fault grading.

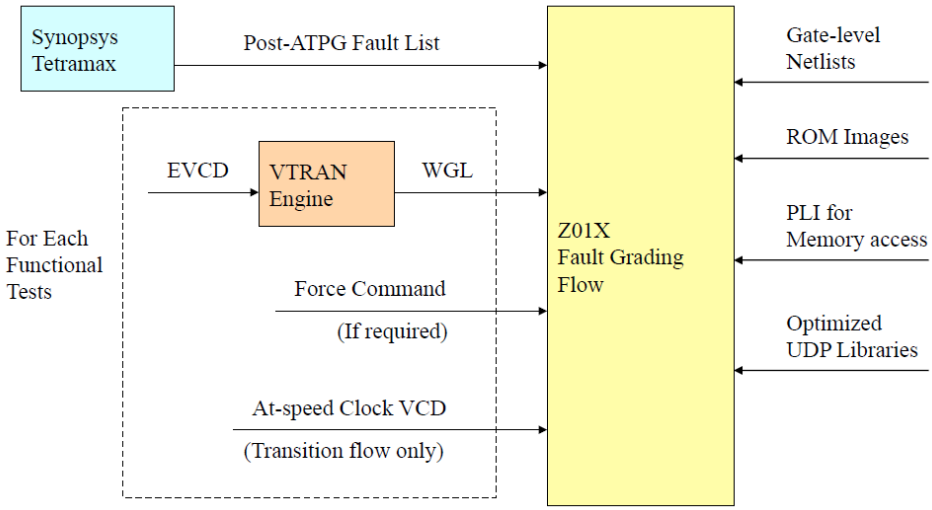

Figure 4.1: Simulation inputs for fault grading

4.5 Fault Grading Methodologies

The Z01X tool supports simulation with stuck-at, transition delay, and bridging faults. Bridging faults are not considered in this experiment because they require huge computing resources and their coverage impact is very small compared with the simulation effort. Because the nodes being detected as TDFs also can be detected as stuck-at faults, transition fault grading is applied first for the total fault list. Then, stuck-at fault grading is applied for the remaining undetected faults.

4.5.1 Fault Simulation

Fault simulation verifies the completeness of the manufacturing test sets for data and control paths within a chip [3, 4]. It works by inserting hypothetical faults into the chip design and by running the manufacturing tests against the faulty chip. The results are then compared against the unfaulted design. If the tests are able to detect the faults, then there is a high probability that the tests will detect manufacturing errors in that area of the chip.

Fault simulation, in some instances, can detect areas of untestability because of design flaws. If functional tests do not cause observable behavior in a portion of the chip during fault simulation, there may be a design problem in that part of the chip. The best place to start a good testing strategy is in the design itself, by organizing each chip design with testability in mind. Good DFT methodologies in the chip translate directly to high confidence in the manufacturing part.

Fault simulation requires target fault sets and driving tests. If the driving tests are the "must-keep" tests, the fault simulation is referred to as *fault screening* because the faults that can be detected by the "must-keep" tests should not be used to judge the effectiveness of the functional tests. If the driving tests are the "secondary" tests, it is referred to as *fault grading*. After each fault screening or fault grading, all the given faults are marked as detected by which tests or not detected by any tests.

The fault simulation algorithm that Z01X uses is the "concurrent fault simulation," which means that it is able to simulate the good machine with large numbers of faulty machines concurrently. This is accomplished by tracking differences between the faulty machine and the good machine. The concurrent fault simulation algorithm assumes that any given fault effect has a small, localized effect on the total machine. By running many faults concurrently, the time required to run all of the faults is significantly reduced. Many other researches also used these concurrent fault simulation techniques and attempted to enhance the overall performance [23, 24, 25].

4.5.2 Transition Fault Grading

The purpose of functional fault grading is to determine the functional tests that do not add value and the value-adding functional tests. The transition fault grading process is shown in Figure 4.2. There are some functional tests that are very critical. They have a unique role in functional verification and testing. These functional tests are called *must-keep tests*. All functional

tests other than the must-keep tests are referred to as *secondary tests*.

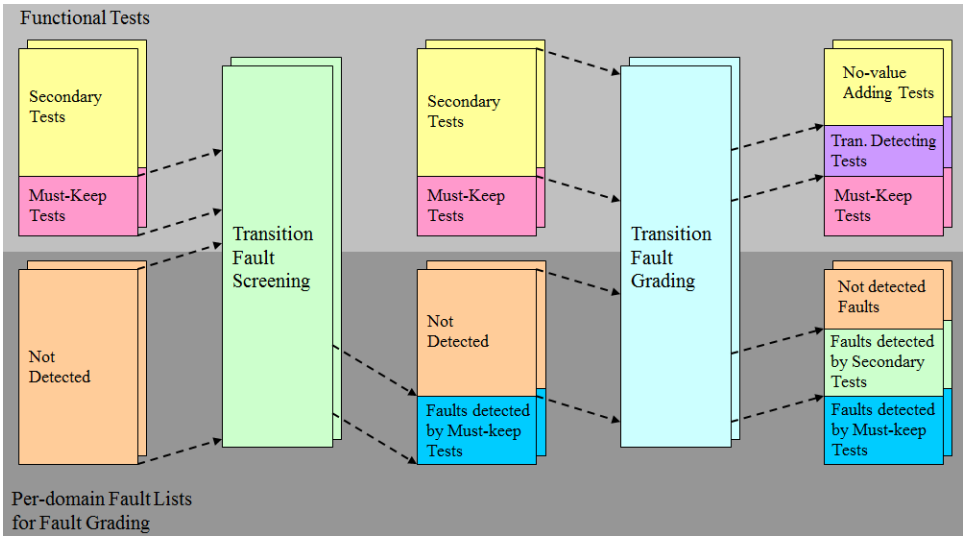

Figure 4.2: Transition delay fault grading

In the transition fault grading flow, transition fault screening is performed with the must-keep functional tests first because the faults that are detected by the must-keep tests are not supposed to be used for judging the quality of the secondary tests whether they add values for the overall test coverage or not. Some of the transition faults are detected by the must-keep tests, and the undetected transition faults are fed into transition fault grading. The secondary functional tests are applied to verify the undetected faults from the transition fault screening. The transition fault grading is repeated for each frequency domain. Also, each frequency domain has its own fault list identified by static timing analysis. The functional tests detecting some faults in transition fault grading are marked as 'transition-detecting' tests and

they are used in stuck-at fault screening. After completing the transition fault screening and transition fault grading, the tests that do not add value will be included in the secondary tests for stuck-at fault grading.

4.5.3 Stuck-at Fault Grading

In the stuck-at fault screening, the must-keep tests and transition-detecting tests are applied against the total stuck-at faults. For the remaining undetected stuck-at faults from the stuck-at fault screening, the secondary tests are applied for stuck-at fault grading. Figure 4.3 shows the stuck-at fault grading flow.

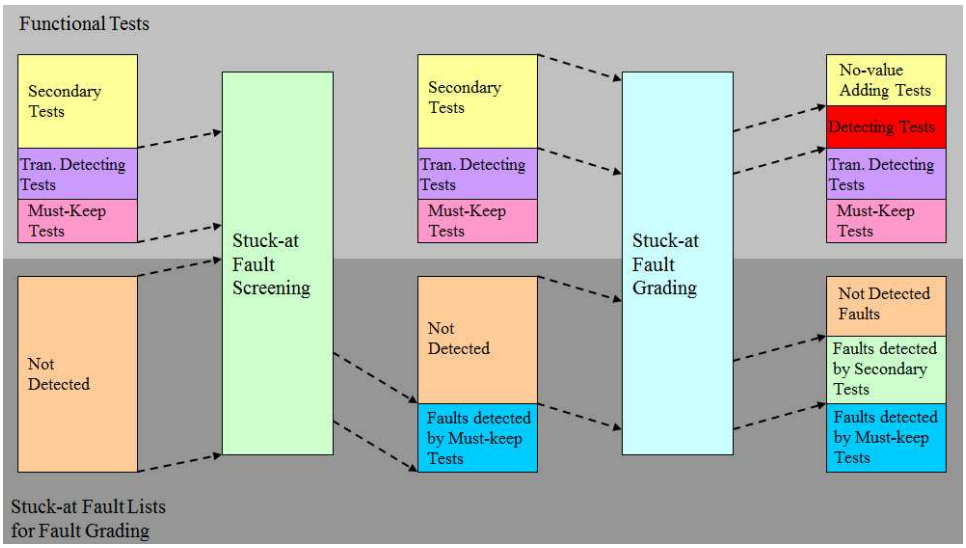

Figure 4.3: Stuck-at fault grading

This fault grading process can identify which functional tests can contribute and increase the test coverage and which functional tests do not add

any value on the test coverage.

4.6 Experimental Results

For the functional fault grading experiment, Qualcomm's 65nm SoC chip is used. The top-level stuck-at ATPG coverage of the device is 98.47%. The total number of faults is 9.1 million, and approximately 140,000 faults are targeted for fault grading with 287 functional tests. Also, 12 different frequency domains are considered as transition fault grading.

Table 4.1: Functional test breakdown

Freq. Domain	**Total Valid Tests**	**Passing Tests (P)**	**Must-Keep Tests (M)**	**Secondary Tests**	**Percentage (M/P)**
F0	58	42	16	26	38.10%
F1	68	52	26	26	50.00%
F2	19	19	19	0	100.00%
F3	19	19	19	0	100.00%
F4	71	55	29	26	52.73%
F5	35	24	20	4	83.33%
F6	29	29	25	4	86.21%
F7	12	12	12	0	100.00%
F8	23	19	4	15	21.05%
F9	9	9	4	5	44.44%
F10	0	0	0	0	0.00%
F11	48	32	17	15	53.13%
Stuck-at	287	251	91	160	36.25%

Table 4.1 shows how many functional tests are valid for each frequency domain. Each functional test was loaded into the Z01X tool and its functionality was assessed first. Some tests were not fully functional and controllable in the tool because of simulator compatibility or poor quality of the tests. Table

4.1 also shows the number of passing tests in each frequency domain. The percentage between must-keep tests and secondary tests is also described in Table 4.1. Two hundred and fifty-one functional tests are passing for stuck-at fault grading, and approximately 36.25% of the tests are considered as must-keep tests.

Table 4.2: Fault coverage status

Freq. Domain	**ATPG Coverage**	**Post FG* Coverage**	**Value-adding STs***	**No value STs***	**Coverage Increase**
F0	94.88%	95.97%	15	11	1.09%
F1	91.10%	91.84%	17	9	0.74%
F2	92.89%	No Secondary Test	0	0	0.94%
F3	96.00%	No Secondary Test	0	0	0.65%
F4	97.01%	No Detect	0	26	0.00%
F5	94.16%	94.64%	1	3	0.48%
F6	96.51%	97.25%	2	2	0.74%
F7	95.51%	No Secondary Test	0	0	0.50%
F8	94.24%	95.13%	4	11	0.89%
F9	91.90%	No Detect	0	5	0.00%
F10	98.63%	No Test	0	0	0.00%
F11	94.08%	94.16%	4	11	0.08%
Stuck-at	98.47%	98.70%	40	120	0.23%

*: FG = Fault Grading, ST = Secondary Test

Table 4.2 shows the fault grading result for each frequency domain. The F4 and F9 frequency domains did not show any detection and showed 0% coverage increase. The F10 domain did not have any valid functional tests at all. The F2, F3, and F7 frequency domains did not have any secondary test candidates for fault grading, but the coverage increase was contributed by the valid must-keep tests for each domain. All other frequency domains have both must-keep tests and secondary tests and showed some coverage increase. From

stuck-at fault grading, the enhancement of the test coverage by all detecting tests is 0.23%, but this enhancement is achieved by only 40 functional tests out of 160 secondary tests. Hence, even with all the 160 secondary tests on top of the must-keep tests, the coverage enhancement is still 0.23%, but the test time consumption is huge because of the 120 no-value-adding functional tests.

The 0.23% stuck-at coverage added by those value-adding functional tests might be considered as very minimal, but Figure 4.4 shows how much enhancement for the defective parts per million (DPM) it can achieve. It shows the cases for two different defect density (DD) processes.

Stuck-at Fault Coverage and DPM		Stuck-at ATPG Analysis	Stuck-at Fault Grading with Functional Tests	
			With Must-keep Tests only	With all detecting Tests
Coverage		98.4739%	98.5653%	98.6953%
Enhancement		-	0.0914%	0.2214%
DPM	**Typical DD (stable process)**	205	195	181
	2-Sigma DD (unstable process)	374	354	325

Figure 4.4: Stuck-at Coverage Enhancement

4.7 Conclusion

The experimental results have shown that the overall impact of functional tests on test coverage is not that significant, but the fault grading process identified the number of no-value-adding functional tests. Functional test selection can be decided based on a given test budget. Although some functional tests are not adding any value for test coverage, it is not an easy decision to skip them because there might be some test coverage hole made by the skipped tests. This experimental result can present a guideline for the functional test selection. In addition, this result can be used to assess the quality of the functional tests.

One of the shortcomings of this fault grading process is the total simulation time. Depending on the size and complexity of clock domains, the transition fault grading took a few days. The stuck-at fault grading took approximately 20 days for the worst case. Of course, the process can be accelerated by allocating more computing resources. On the basis of the simulation logs, after the first 10 days, most simulation processes were saturated and moved very slowly. Proper empirical decisions can help the setup of the tool and control the simulation time.

Although this fault grading process is still a huge task in terms of engineering resources, it is valuable to achieve efficient production testing flow especially for the latest large SoCs. This functional fault grading needs to be performed only once for the device. By spending just multiple days' effort for this functional fault grading, the final production test flow can be more

optimized in terms of test cost, and the test time reduction achieved by this functional fault grading can be accumulated over months and years for high-volume production.

Chapter 5

Contactless Leakage Test for Manufacturing Defects on Package-on-Package Devices

Recently, system-on-chip (SOC) devices require faster and more complex functions, and they need to be implemented by smaller scale technology. Because of the expanded application scopes, the latest SoC devices are equipped with bigger and faster memory. One of the most advanced packaging schemes is to stack a memory package on top of the SoC package [26]. This stacked interface provides higher density and fast signal propagation with short connections and low signal interference.

This packaging scheme is very flexible to handle many different memory requirements with an SoC package to target different vendors and various products, but it poses many testing challenges. This package is connected by a Ball Grid Array (BGA), and the increased number of pads requires the same number of test channels to access the pads. Also, higher pad density could introduce continuity failure on the pads.

To check for manufacturing defects on the pads, signal propagation needs to be checked by test channels. For high-volume production testing, all the pads on the package-on-package (POP) device may not have direct test

channels. Hence, the conventional continuity test cannot be performed for the pads not having direct test channels. Because most SOC devices support boundary scan cells for the pads compliant with IEEE 1149.1, a leakage-based testing scheme is implemented by using the internal boundary scan chain. This scheme can detect manufacturing defects on the pads with no direct test channel by measuring the leakage current of the connected power supplies.

5.1 The Package-on-package Approach

The POP approach is an integrated circuit packaging technique to allow vertically combining discrete logic and memory BGA packages [26]. This stacked interface allows higher density and fast signal propagation with short connections and low signal interference. Typically, an SoC package is sitting at the bottom, and a memory package is on the top as depicted in Figure 5.1.

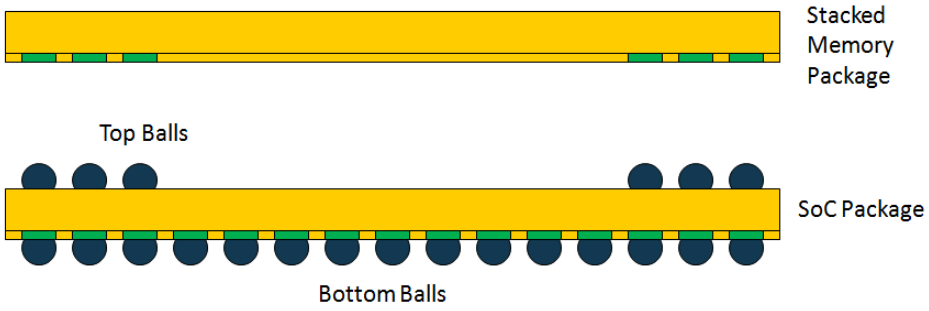

Figure 5.1: Package-on-Package

Each package can be tested separately, and if one package is malfunctioning, only that package must be replaced. Also, a family of products can be

quickly and easily implemented by changing the memory package only. The end users such as mobile phone makers control the logistics. This means memories from different suppliers can be used for different targets without changing the logic. The memory becomes a commodity to be sourced from the supplier with the lowest cost. This trait is also a benefit compared with the package-in-package (PIP), which requires a specific memory device to be designed in and sourced upstream of the end users.

Electrically, POP devices offer several benefits by minimizing track length between different interoperating parts, such as a controller and a memory. This yields better electrical performance of the devices, because the shorter route of interconnections between circuits yields faster signal propagation and reduces noise and cross-talk.

5.2 Challenges of POP Testing

POP devices have many testing challenges. Once the memory package and the SoC package are assembled together as a stacked device, there is no way to access the top balls of the SoC package or bottom balls of the memory package. Typically, the memory die in the memory package is a known-good-die (KGD), but during the packaging process, some manufacturing defects could be added. The memory package itself can be tested by itself, but it is very expensive to test the SoC package and the memory package separately. Although the SoC package is tested by itself, the increased number of pads for the top and bottom pins requires many test channels. Without having

direct test channels for the all pins, conventional continuity tests cannot be performed.

5.3 Boundary Scan-Based Test

The boundary scan description language (BSDL) is a description language for specifying the attributes of boundary scan cells in the SOC. It is part of the IEEE Standard 1149.1, and the BSDL files are well supported by various joint test action group (JTAG) tools for boundary scan applications. The boundary scan-based tests can be generated by doing JTAG programming or by some commercial tools that take the BSDL files. The tool named *Tap Checker* developed by GOEPEL is used to generate the package verification tests.

5.3.1 Test Configuration

To detect any manufacturing defects on the package pins, the continuity of the pins should be verified through the connected test channels. If no test channel is assigned to the package pins, the continuity of the package pins can be verified through the bounday scan cells beyond the package pins. Opposite values have to be applied and checked at least once to each combination of two pins. The simplest solution for this test is to apply walking zero or walking one test patterns. This test is called an asymmetric test. For N pins on a package, the N walking zero patterns or N walking one patterns have to be applied, but the test time of this asymmetric test with $2N$ patterns is huge

if N is very big. Usually, big industrial designs have at least a few hundred pins. In this case, the asymmetric tests are not feasible for production test solutions. To minimize the burden of test time, interconnect tests are required. Because most pads in the SoC support bi-directional signal propagation, both input interconnect test and output interconnect test are required. Each bi-directional boundary scan cell consists of one *input* cell, one *output* cell, and one *output-enable* cell. An output interconnect test is a test doing serial scan-in through the boundary scan chain and comparing the output values of the pads in parallel. This test can detect shorts between any output cells. An input interconnect test is a test applying the test values on the pads in parallel and scanning out the value through the boundary scan chain and comparing the values at the output of the chain. This test can detect shorts between any input cells. The number of required comparison cycles C for N pins on the package can be calculated by Equation 5.1, which is the ceiling function of the logarithm value. The C number of comparison patterns can be obtained by making array values with all possible binary combinations by the digit of C except one of the cases having all identical bits. When the N pin is a power of 2, both all zero and all one cases need to be added into the array. Each pattern can be taken by each column from the array. This is referred to as the *counting algorithm*.

$$C = \lceil \log_2 N \rceil \tag{5.1}$$

For the N pins on the package, the C input interconnect test patterns

and C output interconnect test patterns are required. If the N is big for a large design, then $2C$ calculated using Equation 5.1 is much smaller than $4N$. The $4N$ asymmetric patterns consist of $2N$ input asymmetric test patterns and $2N$ output asymmetric patterns. These interconnect tests have huge test time benefit over asymmetric tests while achieving the same test coverage, but these tests require direct test channels to check the scanned-in values.

5.3.2 Contactless Vector Generation

Most pins on an SOC package are bi-directional pins. To test for shorts on the top pins, all bi-directional top pins are redefined as output pins, and all bi-directional bottom pins are redefined as input pins in the BSDL file as shown in Figure 5.2. An output interconnect test was generated with the GOEPEL tool.

By default, this generated test has stimulus on test access port (TAP) pins at the bottom of the package and comparison cycles for all the pins on the package, but with no channel for the top pins, the comparison values for the top pins cannot be checked on the ATE, which means the functional test vector never fails for any cases, but if there are some shorts between any top pins, the leakage current on the connected power supply goes up significantly. Hence, this test vector can be used as a leakage test to detect the shorts on the top pins that have no direct test channel. For the given comparison cycles, the leakage current is compared to the nominal current, and excessive current indicates the existence of shorts on the top pins.

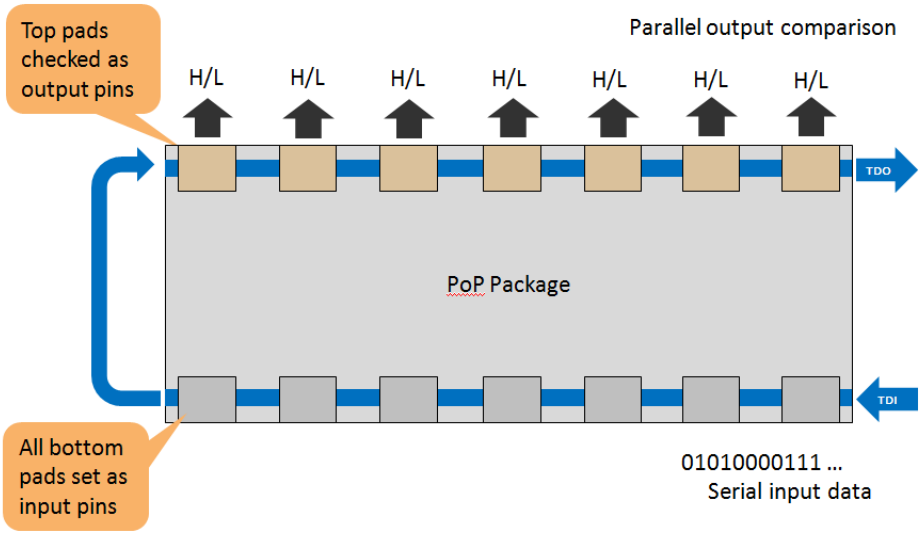

Figure 5.2: Contactless Leakage Test Configuration

Because the commercial test generation tool provides a test suite with different test features, the test generation result with interconnect test itself showed missing test pairs. As shown in Figure 5.3, the test coverage analysis flow is implemented making sure the contactless test has 100% test coverage.

5.4 Extension to Multi-site Testing

The initial test flow may be performed with all the pins on the package, but the test channels are very expensive and the number of the test channels per Circuit Under Test (CUT) should be minimized. Hence, the initial production flow is performed with a limited pin configuration, which specifies only the critical signals, but this limited pin configuration supports all the different

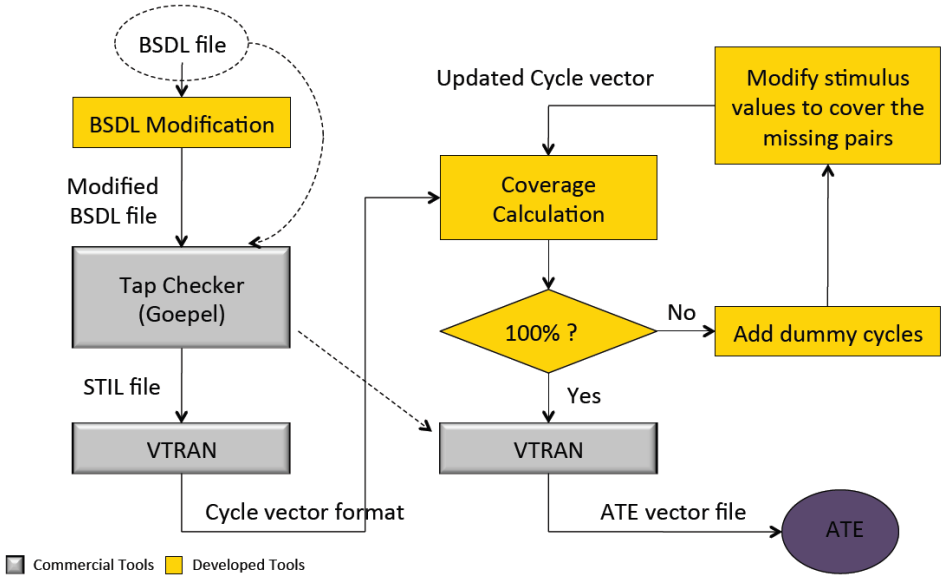

Figure 5.3: Contactless vector flow

types of structural tests and functional tests.

Once the initial production flow is fully verified, higher production volume can be achieved by testing more of the CUTs at the same time. Because the number of test channels on ATE is limited, the limited pin configuration has to be reduced further for multi-site testing. The amount of affordable multi-site testing is dependent on the size of the SOC chip and the complexity of the tests, but usually the number of multiple CUTs that can be tested ranges from 8 to 16 or more. Testing n CUTs in parallel is called *nX testing*. For example, 8X testing deals with eight CUTs in parallel, and it allowes only small number of pins per CUT because pins on ATE is limited. Also, access to the top pins is not allowed in 8X testing. Even for the bottom pins, the use

of fewer pins would be desirable. As a contactless test configuration for 8X testing, the inaccessible bottom pins could be redefined as output pins. The pins defined to have direct channels for the 8X test configuration have to be changed into input only pins. Figure 5.4 shows the multi-site test configuration.

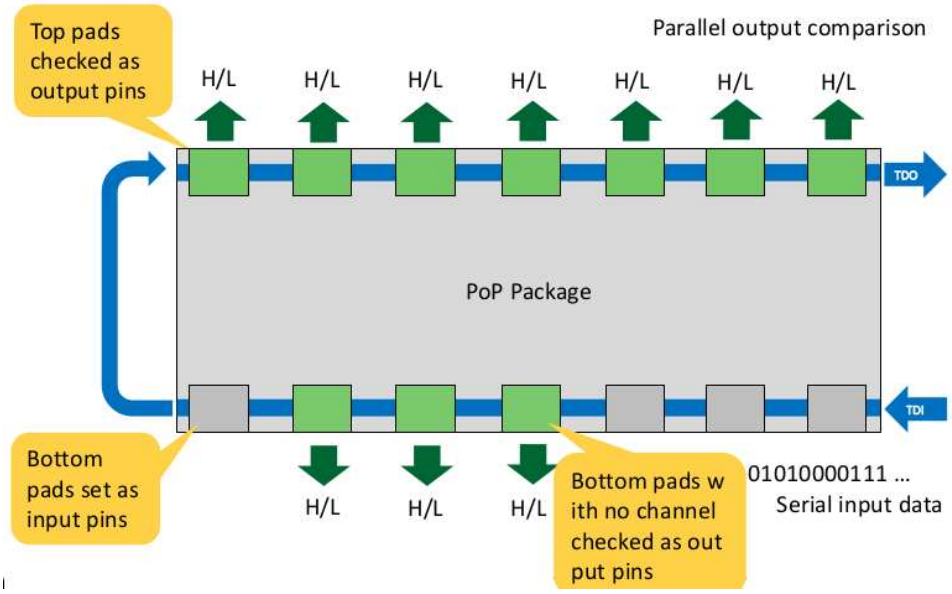

Figure 5.4: Contactless leakage test for multi-site testing

With the modified BSDL file as stated previously, the same test generation scheme can be used. For the given comparison cycles, the leakage current can be measured to check for possible shorts on the pins that have no direct test channels.

5.5 Experimental Results and Conclusion

The contactless leakage tests are developed for production test flow. To check the capability of the contactless leakage tests, artificial resistive shorts were created manually wiring all the combinations of shorts and checked by the tests. There was no test escape for any of the given combinations. Figure 5.1 shows the detailed experimental cases by a resistive short of 100 Ω. The applied resistive short caused a leakage current ranging from 10 to 100 times greater than nominal leakage current depending on the used resistance value. By setting proper threshold values, shorts on the pins that do not have direct test channels can be detected efficiently.

Table 5.1: Leakage current by artificial shorts

Experimental cases	**Leakage current on VDD_PAD1**	**Leakage current on VDD_PAD2**
No short (PASS)	1746.65	1501.944
Short 2 pins in pin group A	1747.667	11301.411
Short 2 pins in pin group B	11620.545	1556.945
Short 1 pin with power	15928.135	1577.046
Short 1 pin with ground	18778.884	1798.732

(Leakage current unit: μA)

Boundary scan chains are given to most modern designs. Key point of this chapter is to maximize the usage of the boundary scan chains in the SOC, and the testing of top pins on POP devices has been achieved by sensing leakage current by scanning in minimum sets of stimulus patterns.

Chapter 6

Structural Test Time Reduction by Transition Delay Fault Grading

As very large-scale integrated (VLSI) circuit operation speeds become faster, signal transmission in the circuit should be finished in less time. High-speed circuits with aggressive timing permit only very small delay slacks along many internal circuit paths. Hence, small process variations during VLSI fabrication may cause multiple delay faults having various magnitudes in the circuits. Hence, delay fault testing has become more important. Accompanied by this delay fault testing, conventional static testing also needs to be included in the production flow in order to assure the high quality of the tested products [27, 28].

A stuck-at fault can be considered as a Transition Delay Fault (TDF) having an infinite delay. Hence, TDF grading was performed to estimate the TDF vectors' capability to detect stuck-at faults. Because those TDF vectors can detect some or many stuck-at faults, the static Automatic Test Pattern Generation (ATPG) process can be targeted for only undetected stuck-at faults left from the TDF grading. Then, this reduced stuck-at faults set would result in fewer static ATPG patterns. The overall structural test time can be reduced

while keeping the same test coverage.

6.1 TDF grading strategy

The latest industry designs have adopted core-based DFT, and multiple core domains are isolated by a DFT design or a test access mechanism (TAM), which is compliant with IEEE P1500. Because these TDF tests are generated for each frequency domain per core, the total amount of TDF test time is very significant, but the typical delay test coverage for complex industrial design is still too tough to get 90%. Both static ATPG tests and delay ATPG tests need to be kept in a production flow to ensure high quality.

Three commonly used delay fault models are the transition fault model, the gate fault model, and the path delay fault model. The transition fault model assumes that the delay fault affects only one gate in the circuit, and the extra delay caused by the fault is large enough to prevent the transition from reaching any primary output within the specification time. This fault can be detected on any sensitized path through the fault site. The gate delay fault model captures small and large delay defects that affect single locations in the circuit. The path delay fault model captures small extra delays, such that each one by itself may not cause the circuit to fail, but their cumulative effect along a path from inputs to outputs may result in faulty behavior [29, 30]. The advantages of the TDF model are that the number of faults in the circuit is linear with the number of gates. The stuck-at fault test generation procedure can be easily modified for transition delay test generation. In this chapter,

this delay fault test will be addressed as a transition delay test.

6.2 TDF Grading Flow

The TDF grading is performed using the Synopsys ATPG tool called TetraMAX. Because this TetraMAX tool was used to generate both the stuck-at ATPG patterns and the TDF patterns, the TDF grading process is fully compatible in the tool for the same design netlist.

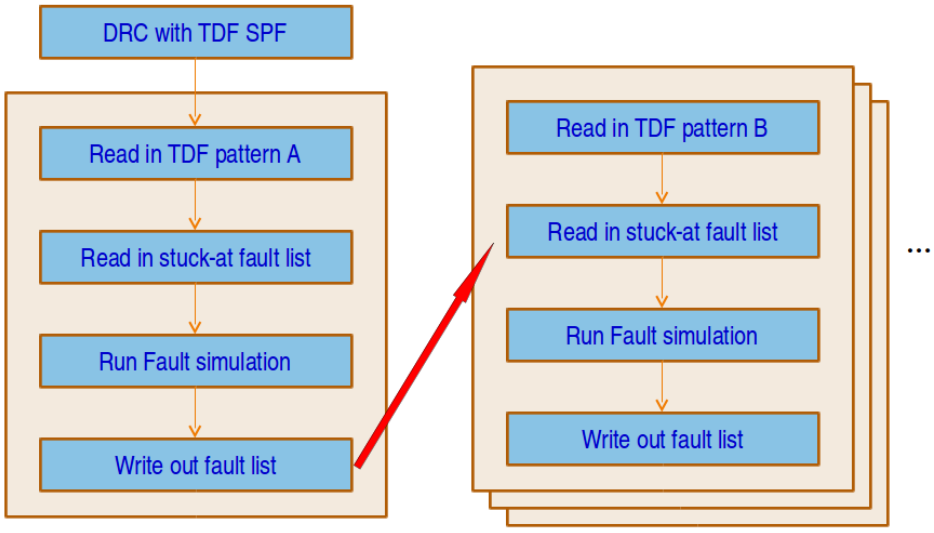

Figure 6.1: TDF grading flow

Figure 6.1 describes the flow of TDF grading. After reading in the design and completing the design rule check, one TDF test for each frequency domain is being set as test patterns. Static fault simulation is performed

with the full list of stuck-at faults. Then, each TDF test can have individual stuck-at fault coverage.

First, the TDF test having the highest individual stuck-at fault coverage is used as the test patterns for fault grading, and static fault simulation is performed with the full list of stuck-at faults. This one procedure with one TDF test and the given stuck-at fault sets can be called *one TDF grading step*. From one TDF grading step, the undetected stuck-at faults are fed into the next TDF fault grading step with different TDF patterns. Within the given core, multiple frequency domains exist. The order of frequency domains in the TDF grading can be decided by the individual stuck-at fault coverage of each domain. The higher the stuck-at fault coverage, the more stuck-at faults can be detected in the TDF grading step. By trying the better TDF tests first, more stuck-at faults can be detected at the beginning of the TDF grading steps, and the overall TDF grading process can be accelerated. These steps can be repeated for each core.

6.3 Experimental Results

For the TDF grading experiment, Qualcomm's 45nm SOC is used. This SOC has a core-based DFT design having eight separate cores isolated by IEEE P1500. Each core has different sets of fault sets and multiple frequency domains.

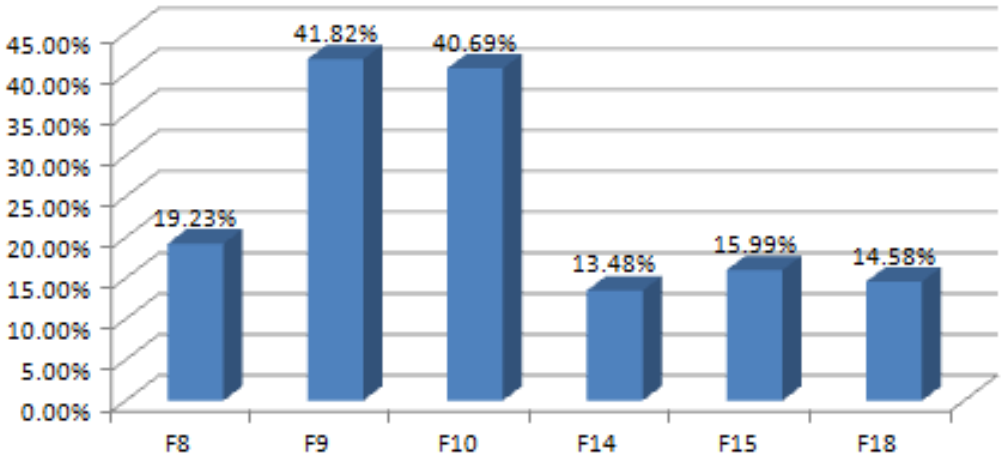

Figure 6.2: TOP: individual coverage per frequency domain

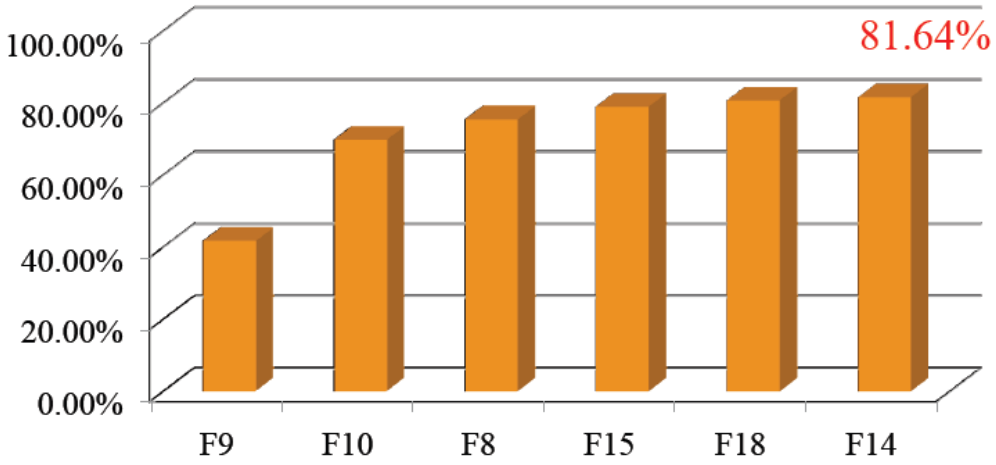

Figure 6.3: TOP: cumulative coverage per frequency domain

6.3.1 Core1 - TOP

As can be seen in Figure 6.2, the TOP core has six different frequency domains. As stated earlier, the order of TDF grading is decided based on

individual stuck-at fault coverage. Starting with the F9 domain having the highest coverage, a series of TDF grading steps is performed accordingly. The sequence is F9, F10, F8, F15, F18, and F14 in this case. Because one TDF grading step takes the undetected stuck-at fault sets from the previous TDF grading step, the undetected stuck-at faults by any of the TDF tests in the given core will remain in the stuck-at fault list as undetected. The stuck-at ATPG generation process will target this undetected fault list, and it will generate a reduced number of stuck-at ATPG patterns. Overall stuck-at coverage remains the same. Figure 6.3 shows this TOP core's cumulative coverage. With the six TDF tests in the TOP core, a stuck-at coverage of 81.64% is achieved. The original stuck-at ATPG pattern count for TOP core was 5346, but the newly generated stuck-at ATPG pattern count after TDF grading is 4,194. This results in a test time saving of 21.55%.

6.3.2 Core2 - SUBSYSTEM

As can be seen in Figure 6.4, the SUBSYSTEM core is a very big core having nine different frequency domains. Starting with the F9 domain that has the highest coverage, a series of TDF grading steps is performed accordingly. Figure 6.5 shows this SUBSYSTEM core's cumulative coverage. With the nine TDF tests in the SUBSYSTEM core, a 90.31% stuck-at coverage is achieved. The original stuck-at ATPG pattern count for SUBSYSTEM core was 5,356, but the newly generated stuck-at ATPG pattern count after TDF grading is 4,521. This results in a test time saving of 15.59%. Figure 6.6 shows the stuck-

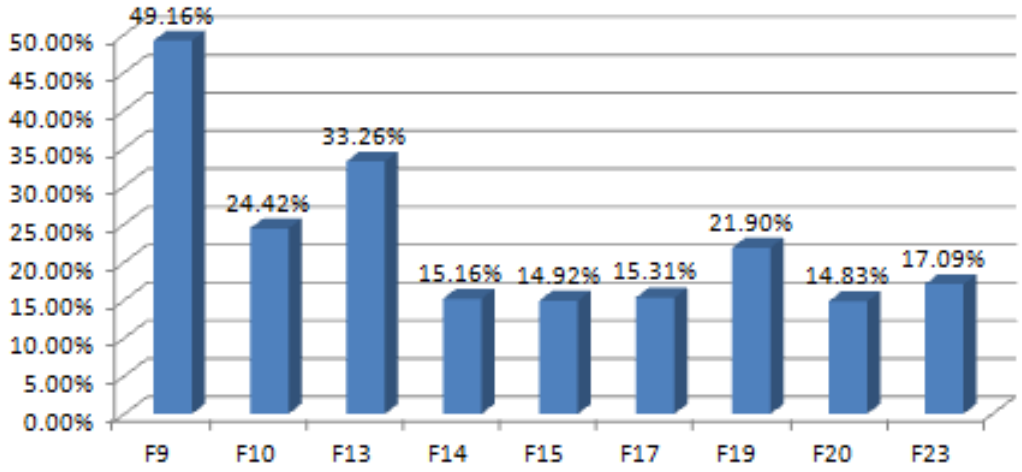

Figure 6.4: SUBSYSTEM: individual coverage per frequency domain

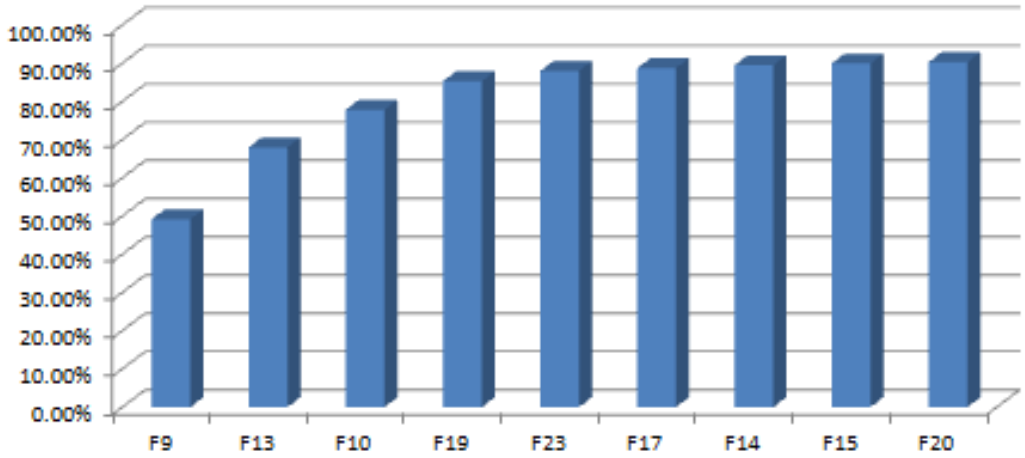

Figure 6.5: SUBSYSTEM: cumulative coverage per frequency domain

at coverage trend per pattern with and without TDF grading. The same level of test coverage is achieved with less number of patterns by TDF grading.

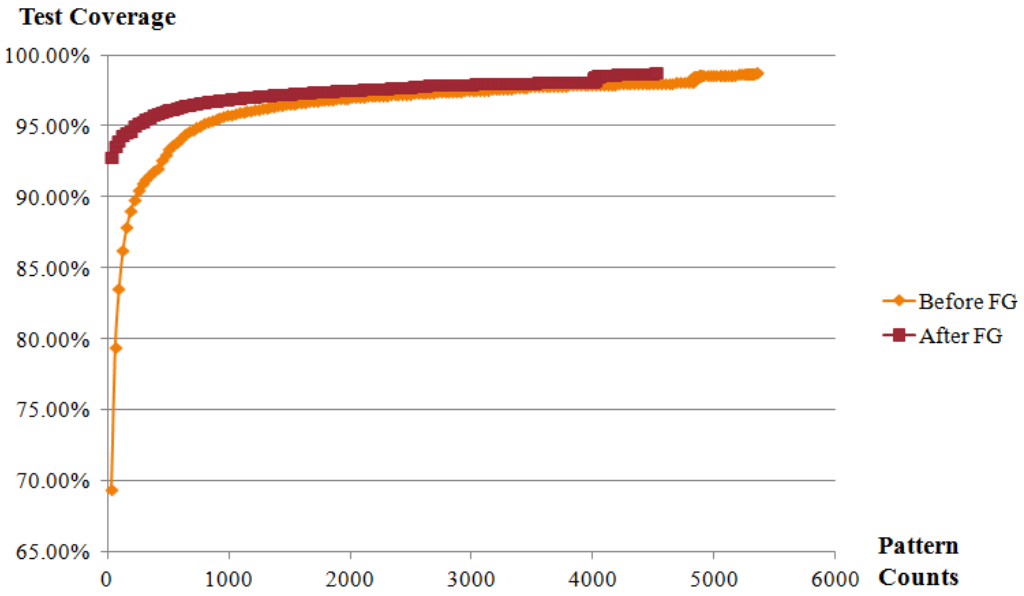

Figure 6.6: SUBSYSTEM: coverage comparison

6.3.3 Core3 - GRAPHICS 2D

The GRAPHICS 2D core has only two frequency domains. One TDF test has almost no test coverage, but the other TDF test has shown 90.28% coverage. The TDF test having higher coverage is fed into TDF grading step. The original stuck-at ATPG pattern count for GRAPHICS 2D core was 1,204, but the newly generated stuck-at ATPG pattern count after TDF grading is 1,066. This results in a test time saving of 11.46%. Figure 6.7 shows the stuck-at coverage trend per pattern with and without TDF grading.

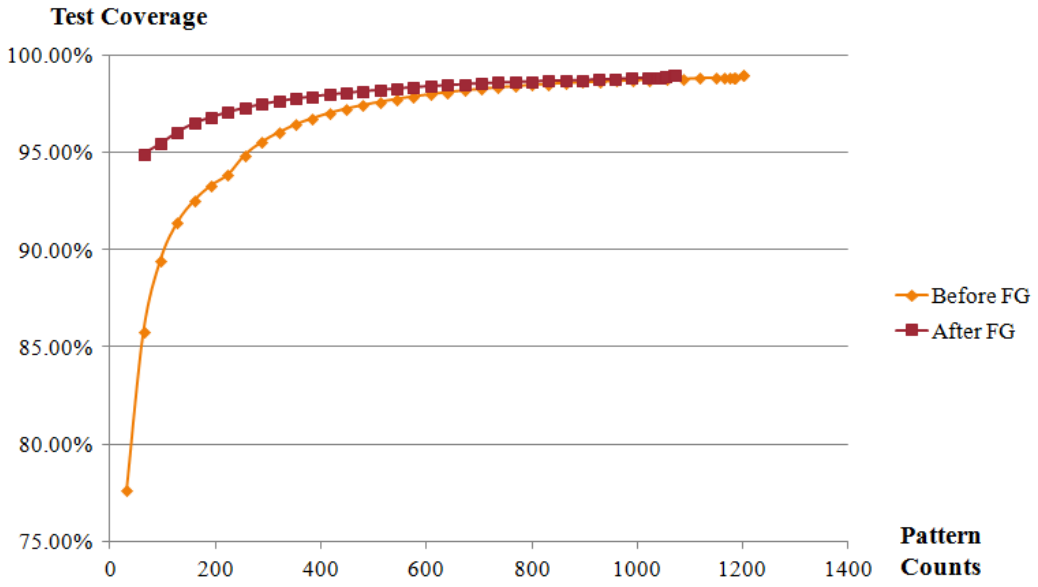

Figure 6.7: GRAPHICS 2D: coverage comparison

6.3.4 Core4 - GRAPHICS 3D

The GRAPHICS 3D core also has only two frequency domains. Those two TDF tests are fed into TDF grading steps accordingly, and the cumulative stuck-at coverage is 94.54%. The original stuck-at ATPG pattern count for the GRAPHICS 3D core was 2,420, but the newly generated stuck-at ATPG pattern count after TDF grading is 1,798. This results in a test time saving of 25.70%. Figure 6.8 shows the stuck-at coverage trend per pattern with and without TDF grading.

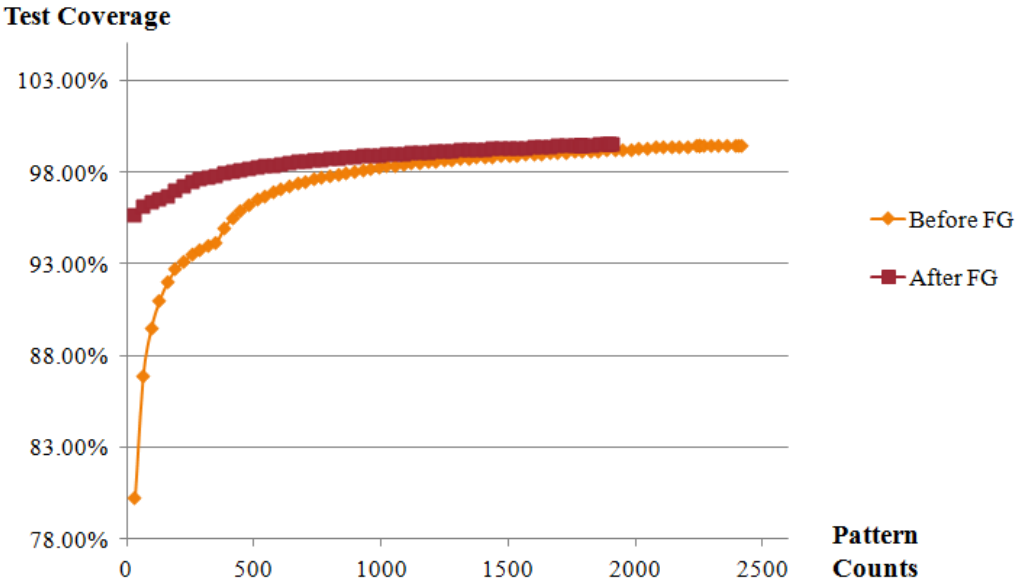

Figure 6.8: GRAPHICS 3D: coverage comparison

6.3.5 Core5 - MODEM

As can be seen in Figure 6.9, the MODEM core is also a very big core having nine different frequency domains. Starting with the F6 domain having the highest coverage, a series of TDF grading steps is performed accordingly.

Figure 6.10 shows the MODEM core's cumulative coverage. With the nine TDF tests in the MODEM core, a 59.48% stuck-at coverage is achieved. Compared with the SUBSYSTEM core, this MODEM core has less stuck-at coverage with the nine TDF tests. The original stuck-at ATPG pattern count for the MODEM core was 10309, but the newly generated stuck-at ATPG pattern count after TDF grading is 9,832. This results in a test time saving of

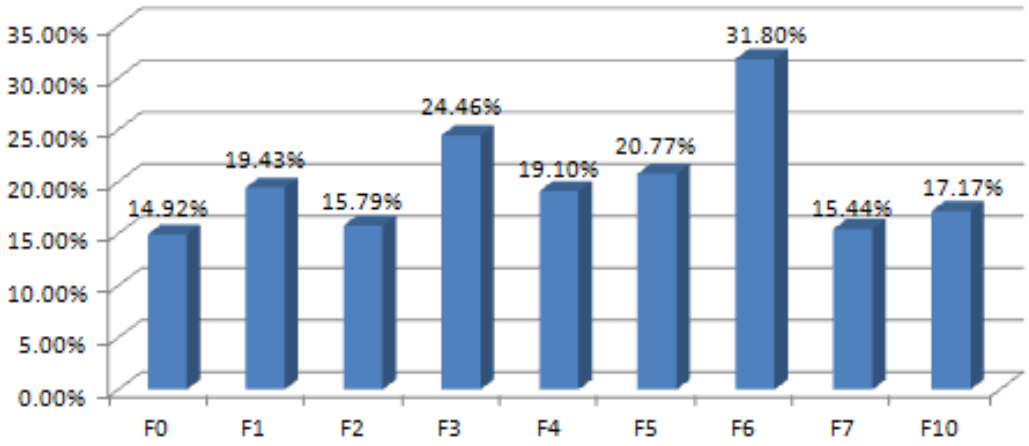

Figure 6.9: MODEM: individual coverage per frequency domain

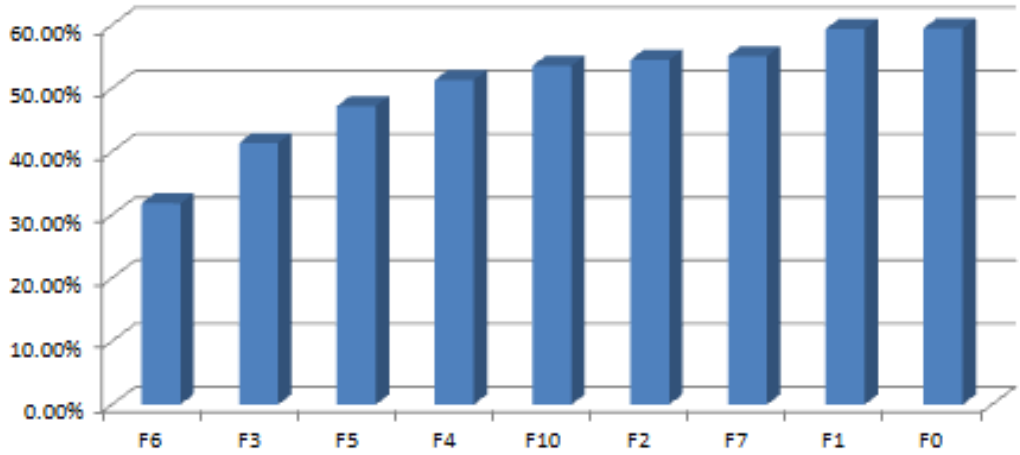

Figure 6.10: MODEM: cumulative coverage per frequency domain

4.63%. This MODEM core has the most complex logic and huge gate count. The lack of a fully controllable DFT design in this core resulted in this low coverage.

6.3.6 Core6 - ARM

The ARM core has only two frequency domains. Those two TDF tests are fed into TDF grading steps accordingly, and the cumulative stuck-at coverage is 92.32%. The original stuck-at ATPG pattern count for the ARM core was 1,416, but the newly generated stuck-at ATPG pattern count after TDF grading is 1,134. This results in a test time saving of 19.92%. Figure 6.11 shows the stuck-at coverage trend per pattern with and without TDF grading.

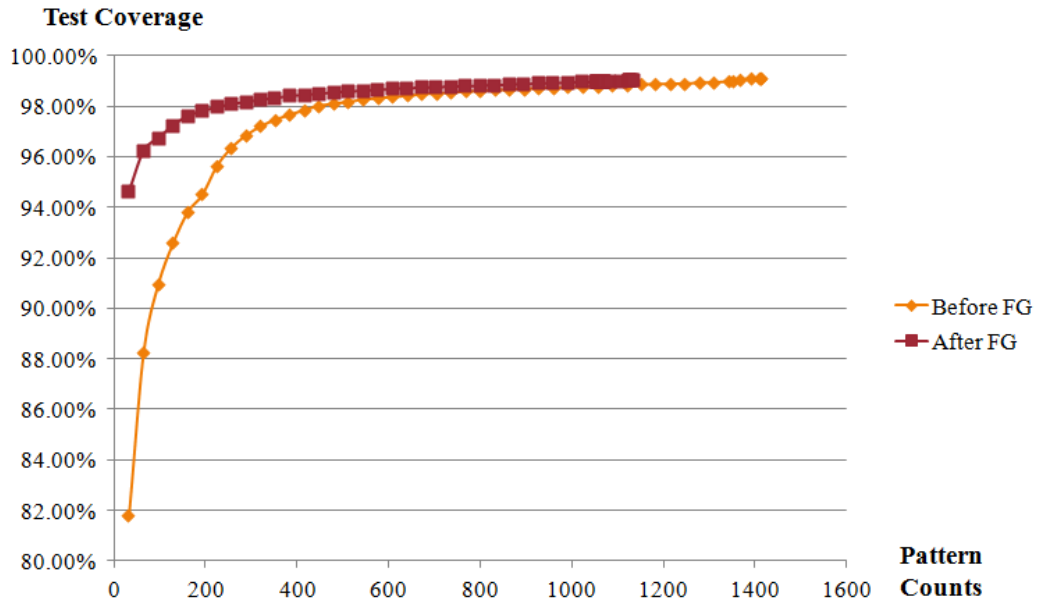

Figure 6.11: ARM: coverage comparison

6.3.7 Core7 - CPU

The CPU core has only three frequency domains. Those three TDF tests are fed into TDF grading steps accordingly, and the cumulative stuck-at

coverage is 87.25%. The original stuck-at ATPG pattern count for CPU core was 2,795, but the newly generated stuck-at ATPG pattern count after TDF grading is 2,421. This results in a test time saving of 13.38%. Figure 6.12 shows the stuck-at coverage trend per pattern with and without TDF grading.

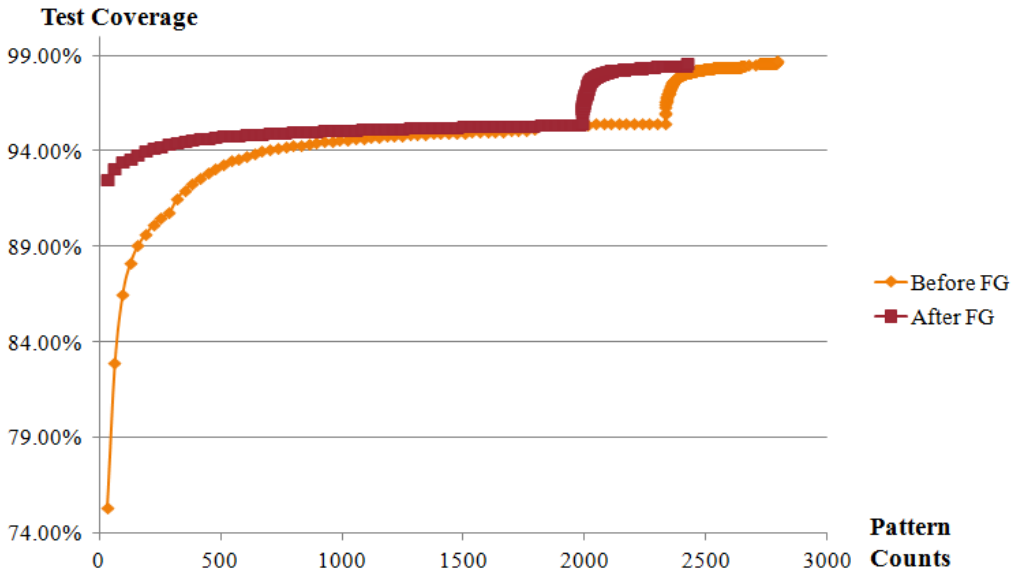

Figure 6.12: CPU: coverage comparison

6.3.8 Core8 - VIDEO

The VIDEO core has only two frequency domains. One TDF test has almost no test coverage, but the other TDF test has shown 87.03% coverage. The TDF test having higher coverage is fed into the TDF grading step. The original stuck-at ATPG pattern count for the VIDEO core was 2,919, but the newly generated stuck-at ATPG pattern count after TDF grading is 1,854.

This results in a test time saving of 36.49%. Figure 6.13 shows the stuck-at coverage trend per pattern with and without TDF grading.

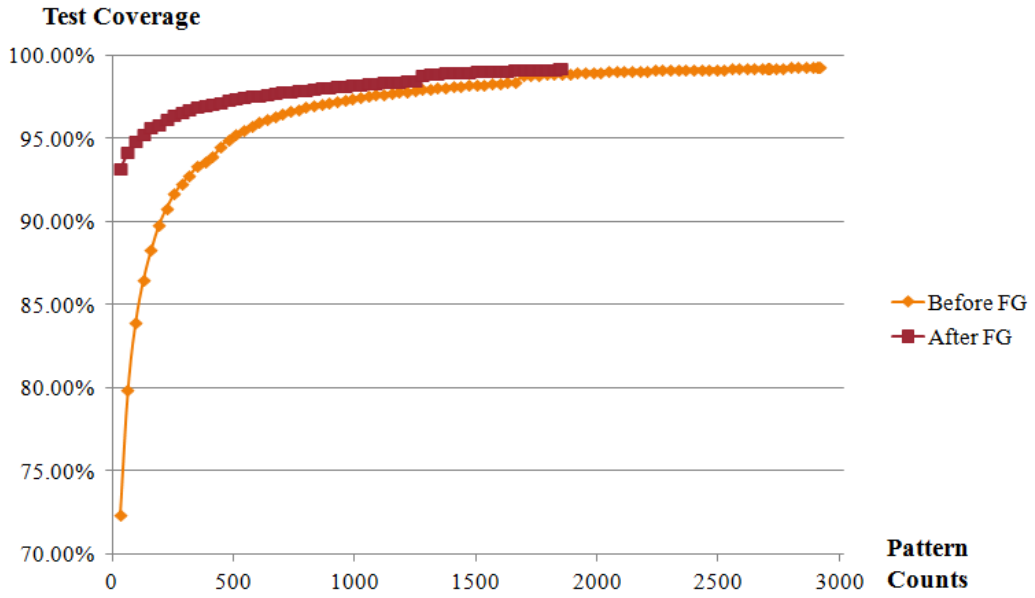

Figure 6.13: VIDEO: coverage comparison

6.4 Conclusion

TDF grading has been performed for all eight cores in the SOC. Some cores show less coverage than the other cores, but most cores have shown a significant test time reduction. Table 6.1 shows each core's pattern count and test cycles. To calculate the actual test time per core, the pattern count is transformed into test cycles based on the longest scan chain length in the core. The scan clock is running at 20 MHz.

Table 6.1: Test time reduction

Core	**Pattern Count**	**Test Cycles**	**Reduced Pattern Count**	**Reduced Test Cycles**
TOP	5,346	2,170,581	4,194	1,702,846
SUB SYSTEM	5,356	2,099,915	4,521	1,772,538
ARM	1,416	432,342	1,134	346,240
GRP2D	1,204	195,886	1,066	173,434
GRP3D	2,420	694,469	1,798	515,973
MODEM	10,309	5,020,212	9,832	4,787,926
CPU	2,795	985,130	2,421	853,309
VIDEO	2,919	544,526	1,854	345,855
Total Test Cycles		12,143,061		10,498,122
Total Test Time		0.607 sec		0.525 sec

Hence, the actual test time is calculated by multiplying the scan clock cycle time by the number of test cycles. Without considering TDF grading, the total test time for the static ATPG test was 0.607 seconds. After TDF grading, the total test time for static ATPG test came down to 0.525 seconds. This is a 13.55% saving in the test time. There is a minimal coverage impact by TDF grading. TDF grading provides a test time reduction while keeping the same test coverage. When the initial production flow is being set up, this TDF grading flow can be performed once to remove the overlap between stuck-at ATPG tests and TDF ATPG tests. Then, the benefit of the reduced test time will be accumulated and become a big saving as the production goes through.

Chapter 7

Analog Behavioral Modeling: Strategies and Methodologies

Complex system-on-chip (SOC) consists of many various kinds of digital and analog cores to achieve highly complicated features. All those cores should be verified for their functionalities at the core level before they are integrated into the SOC-level design. Once they are integrated into SOC level, the design scale becomes bigger and it is harder to debug any potential problems. Analog circuits are composed of basic devices such as transistors, resistors, capacitors, and inductors. Transistor-level custom design implementation is required for analog circuits in mixed-signal cores, radio-frequency (RF) cores, power management cores, etc. These cores cannot be verified by the same way as digital cores are being verified via quick simulation. Analog design verification requires circuit-level simulation using the "Simulation Program with Integrated Circuit Emphasis (SPICE) " model. This process takes an unacceptable amount of time when its scale is not small. In particular, the verification based on the SPICE model cannot be run together with other digital verification methodologies. Analog behavioral models for those analog cores can address the simulation speed issues and interoperability of mixed verification with analog and digital cores. In addition, the validity of

the model-based simulation is contingent on the quality of behavioral models used. Depending on the level of abstraction for the behavioral modeling, there are some limitations for accurate performance analysis. Executing core-level or block-level SPICE simulations may compensate for the lack of accuracy in the analog behavioral model for performance analysis.

7.1 Behavioral Modeling Strategies

Analog signals can change in almost infinitely small increments in terms of time and amplitude. To describe analog functionality, nonlinear equations are required. Verification of analog circuits has traditionally been performed using SPICE simulation, capable of iteratively solving a set of nonlinear equations. SPICE was developed at the University of California, Berkeley in the early nineteen seventies, and since then many enhanced variations have been produced by academia and commercial companies [2].

Behavioral modeling places the substitution of more abstract, less computationally intensive circuit models to describe lower level functions of analog circuits. These simplified models mimic the transfer characteristics of the circuit elements, but with increased efficiency, they lead to substantial reduction in the actual simulation time per circuit. When considering the whole design and total simulation time, this reduction in elapsed time per simulation can lead to a tremendous increase in design and verification efficiency as well as possible reduction in the time necessary to take a design from a concept to a marketable product. Figure 7.1 shows simulation speed results among dif-

ferent simulation approaches. The simulation driven by hardware description language (HDL) consumes linearly scaled time by design complexity whereas SPICE simulation consumes exponentially increased simulation time as design becomes complex. Advancements in SPICE simulation, such as Fast-SPICE, provide additional speed and capacity while sacrificing some accuracy. In addition, distributed computing engines have further increased capacity and performance limits, but not enough to keep up with growing size and complexity of the mixed-signal designs.

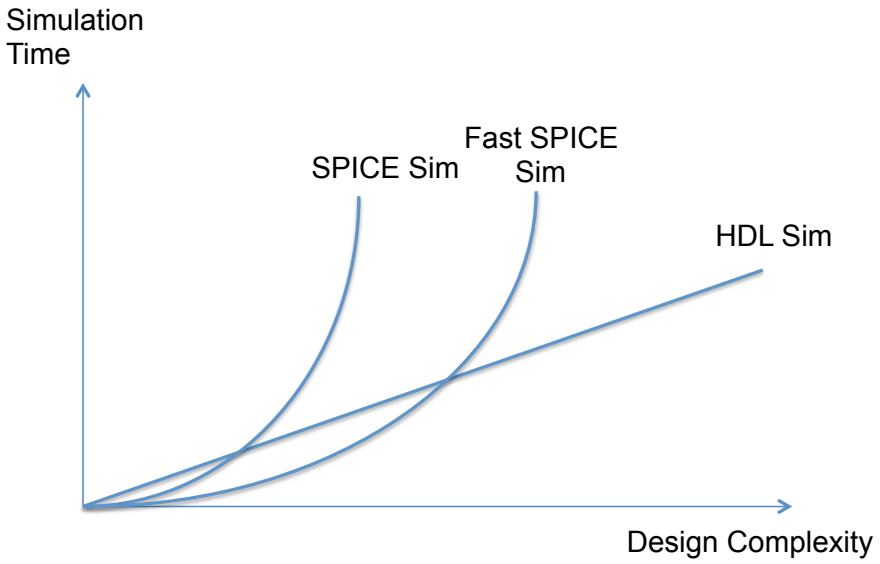

Figure 7.1: Simulation time comparison

Figure 7.2 shows two main different modeling approaches for the analog behavioral modeling. One is a top-down modeling approach, and the other is a bottom-up modeling approach.

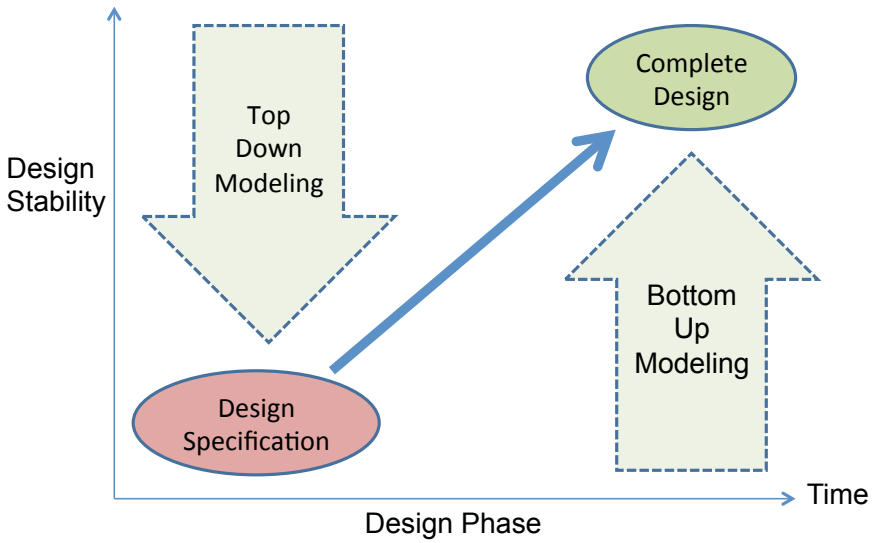

Figure 7.2: Modeling strategy

7.1.1 Top-Down Modeling

The analog behavioral model by top-down modeling approach can be implemented by the specification of analog design blocks. The development and the implementation of analog blocks take longer time and more effort than digital blocks. In the beginning of design phase, only design specification is decided for analog blocks. Although the analog development is ongoing, the implementation of other upper-level system design can be started in parallel to complete the final integrated product earlier. This upper-level system design requires the function of the analog blocks to verify the system functionalities. The key functional analog blocks are a phase-locked loop (PLL) for clock generation and an analog-to-digital converter (ADC) or a digital-to-analog converter (DAC) for data conversion. In the digital domain of the system

design, signal propagation needs to be hooked up between multiple blocks. Top-down modeling can be used in this phase. The top-down models can replace the analog blocks under development in the system design. In addition, they can enable the early start of system design and verification. The top-down model can simply represent the function of the modeling target, but if oversimplification is done, the model may lose verification confidence. It is important to select the proper level of modeling boundary and scope.

7.1.2 Bottom-up Modeling

Bottom-up modeling can be started after a certain phase of the analog design when stable schematic design is ready for critical analog blocks. A detailed modeling boundary needs to be decided for all sub-blocks. In addition, each analog block in deep design hierarchy is replaced by a behavioral model representing its behavior. In this time, all the signal connections should remain the same as the schematic connection. This approach can detect overlooked errors in the schematic connection and potential logical bugs. Because this bottom-up modeling introduce less abstraction, the model is much closer to the actual design, and it is easier to verify the analog design. In some cases, over-modeling at the deep design hierarchy may affect simulation efficiency for SoC-level simulation when the model is integrated into the SoC. Thus, it is crucial to decide the proper level of modeling boundary.

7.2 Industrial Trends

Mixed-signal content, in most of today's SOCs, has increased from 10-20% to 50% or more due to increased needs for mobility, higher performance and integration of interfaces. Similarly, what used to be pure analog blocks now include significant amounts of digital logic either to increase functionality or to assist the analog portions of the design achieve target performance [2].

Cummings presented proper ways to handle delays in circuits as Verilog behavioral models [31, 32]. In the analog circuit design to be modeled for behavioral model, critical delay lines have to be modeled correctly. Otherwise, the improperly modeled delay may cause the analog core to malfunction. There are various delay coding styles in hardware description languages, but very few of the permitted coding styles actually model realistic hardware delays. The inertial delay and transport delay scheme are widely used in the modeling cases described in the next chapter.

One of key factors to be guaranteed by analog behavioral model is to match and correlate between the analog model and the actual analog design. One of the recommended methods to address this challenge is to co-simulate the SPICE circuit with rest of the system [33, 34, 35]. Sharma *et al.* [36] proposed a verification methodology to establish equivalence of analog behavioral model and the SPICE circuit being modeled. The validity of the simulations driven by behavioral models is contingent on the quality of the behavioral models used. In the proposed scheme, SPICE-on-top co-simulation environment to simulate the behavioral model in the same SPICE testbench that is

used for circuit characterization using SPICE simulations, and circuit characteristics/metrics of interest are defined and checkers in the co-simulation environment are developed to measure them.

7.3 Analog Traffic Modeling

In analog behavioral modeling, electrical signals such as voltage and current need to be represented as real numbers to achieve accuracy and corresponding functionality for real analog signal propagation. This is in contrast to digital circuits that can be handled by four value logic (0, 1, Z, X). The interface in the analog model passes those real numbers between blocks and mimics actual signal transfer and propagation.

Analog behavioral models are typically written in Verilog, Verilog-AMS, Verilog-A, VHDL-AMS, or SystemVerilog. The Verilog-A and Verilog-AMS represent real electrical properties of the analog circuit in detail, and they are usually used for performance verification. However, circuit performance can be verified more accurately by circuit-level simulation without having behavioral models.

Figure 7.3 shows modeling accuracy and performance gain for the various real number modeling methods. In real number modeling, analog voltages or currents are represented as a time-varying sequence of real values, This is actually very similar to what analog simulators do. The difference is that in a typical analog simulator, the models define a set of equations and the simulator solves the overall constrained system of simultaneous equations at

each timestamp to compute the voltage or current from those equations. In a discrete real environment such as Verilog-AMS, VHDL-AMS, SystemVerilog, there is no voltage or current equations, and there is no simultaneous equation solution step. The output is directly computed from the input by ignoring the voltage or current and other feedback mechanisms that could have caused interdependencies between drive and load in an electrical environment [2].

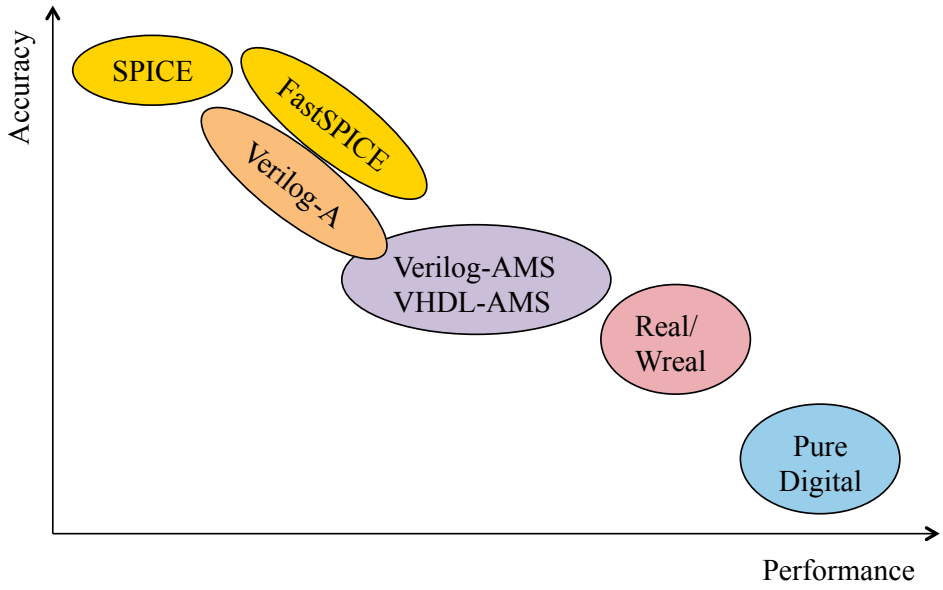

Figure 7.3: Modeling accuracy versus performance gain compared to transistor-level simulation for various modeling styles [2]

When the analog models are integrated into the SOC level, the majority of the SOC design consists of digital designs implemented by Verilog or VHDL. Also, those analog cores are needed in many different cores or sub-systems. On the digital design side, extensive verification effort is applied to the digital

design and analog models. The models developed by Verilog-A or Verilog-AMS do not provide full compatibility with digital design code. Although those models can be used, it may drag down the efficiency of the SOC level verification. In other words, the challenge is to model analog blocks for a digital simulator. In addition, the model should support multiple different digital simulators in the verification chain as shown in Figure 7.4.

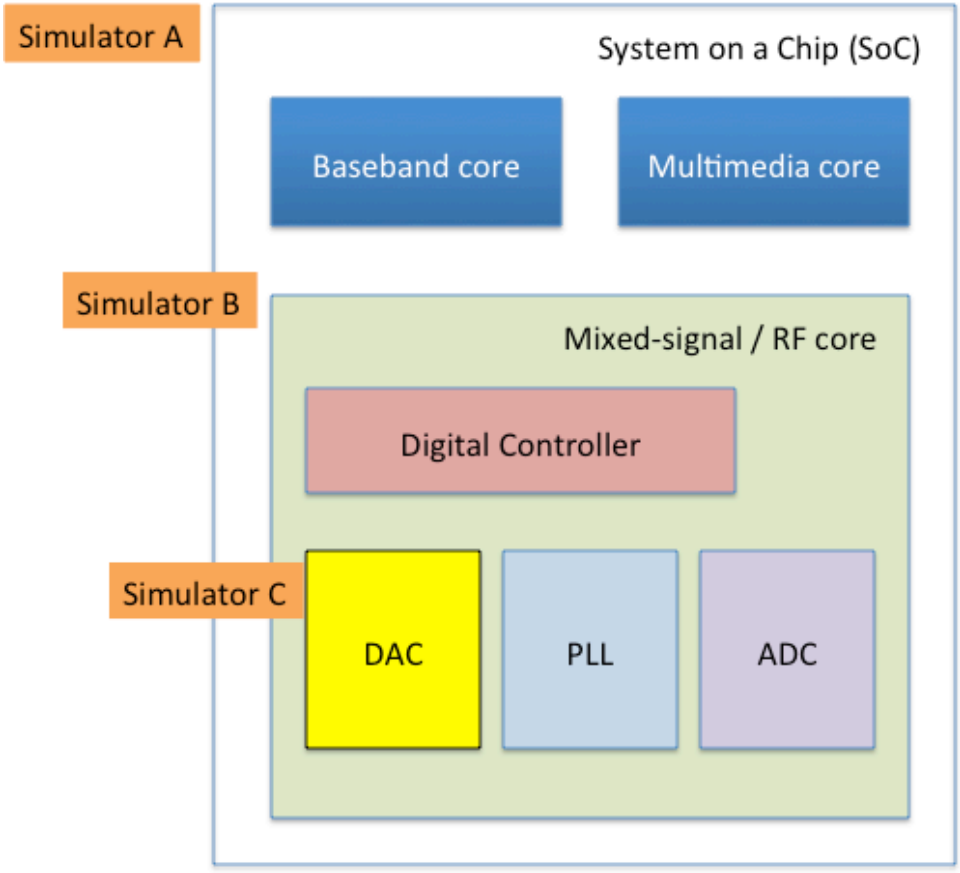

Figure 7.4: Simulators in verification chain

Hence, the proper solution is to restrict the analog model to be de-

veloped by compatible HDL languages with digital design such as Verilog or SystemVerilog. The analog model developed by these HDL languages may not have enough accuracy and variety to support all different corner conditions as SPICE model does. However, the main goal of the analog model is to verify the correctness of the analog design and accelerate the verification processes and enable compatible integration for the upper-level digital design systems. The analog model is mainly targeted for typical corner of what actual analog design covers. For the critical timing of the analog circuits, the typical delay values are extracted from sub-block level SPICE simulations and the values are embedded into the analog model of the analog block. All other general logics in the analog design are implemented as non-timing logics in the analog model, and for the timing critical sections such as scan logic involving analog logics, the timing verification can be performed with the extracted timing from the layout of the analog design. The next section shows multiple different methods used in this research to represent the analog traffic by Verilog code.

7.3.1 Analog Wire Interface

Analog Wire Interface (AWI) is an analog interface module developed as a Verilog programming language interface (PLI) function. This AWI module is developed by the Australian Semiconductor Technology Company (ASTC). Between the AWI modules, the analog signals representing electrical property are transferred through telegraph coding. Figure 7.5 shows the structure of the AWI modules. The analog signal cannot be probed at the ports having the

AWI module behind, but the analog signal values in the AWI can be referenced in the manipulation of the analog signal activities from the analog model.

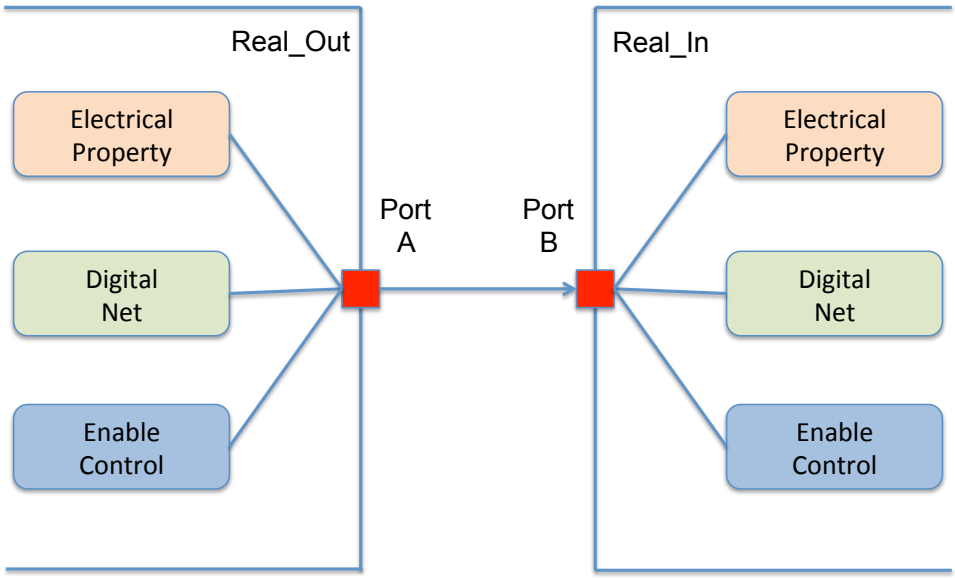

Figure 7.5: Real signal traffic model

There are two types of AWI modules. The *AWI_Realout*, which sends out the real value of analog signal, and the *AWI_Realin*, which receives the real value of the analog signal being sent by the *AWI_Realout*. With these AWI modules, current or voltage values are treated as just a real number. The actual conversion between voltage and current by Ohm's law has to be performed in the analog model code by a user.

In analog circuit cases, some shared nodes have to be resolved for voltage division or current summation. This AWI module cannot handle Kirchhoff's laws. However, this AWI module is quite flexible and has good compat-

ibility with digital design code. This AWI is solid enough to handle the analog signal traffic in large scale design.

7.3.2 Virtual Verilog Wire

Verilog virtual wire (VVW) is also a Verilog PLI module developed by ASTC. Compared with the AWI in the previous section, this VVW can handle more realistic analog signal traffic. There are multiple different kinds of VVW modules to handle voltage and current, input and output, baseband signals, and radio frequency signals. There are a total of 12 different VVW modules. Shared nodes in analog circuits can be modeled by VVW modules to implement divided voltage into multiple load, and merged nodes can be modeled for current summation. The modeling by VVW modules is suitable for small scale design with very detailed analog modeling.

7.4 Modeling Flow

Analog modeling can be started with modeling boundary identification and analysis of each modeling target block. The scope of analog design is given as the hierarchical layers of schematics from analog designers based on the design readiness. The goal of analog behavioral modeling is to verify the given analog design. Majority of the analog block analysis is performed by an analog modeler (me) as a part of modeling process. Since the analog modeler (me) is not as familiar with the detailed analog circuits as the analog designers who designed the circuits, some of modeling directions and approaches for

intensive analog blocks are based on the comments and analog simulation results performed by analog designers.

Once the modeling boundary is identified, the developed analog model would be aligned with the schematic at each modeling boundary. The top-level pin interface of the analog model should match with the pin interface of the actual analog design blocks. Although the functionality of some pins are not modeled by the in-necessity from the model requirements, the pin interface has to be matching each other. Otherwise, the model may end up some warnings in the simulation by the unconnected ports or mismatching pins.

After defining the model boundary, the model skeleton needs to be written with the matching pin names and directions for the modeling target block. Because this model skeleton should match with the actual design ports, a script-based automation is preferred to avoid any human mistakes or errors. During the multiple design phases, it is very likely to have the input/output interface of the design and model to be updated multiple times. This script-based model skeleton generation is one of the key factors to speed up the modeling process. For this script-based model skeleton generation, a customized in-house tool is used. Some feature enhancement of the in-house tool is achieved by me. Also, some health check signals can enhance the quality of the model and verification for confirming supply and ground connection and the enabling conditions for the modeling target block. The in-house tool provides an automated template to easily build those health check signals. Those health checks can be confirmed by an assertion, which is a positive

statement about the given property of a design. When this statement is found to be false, it indicates an error. Those assertion-based checks can enhance the model's capability. On top of those health check signals, the actual model body code that represents the behavior of the modeling target block needs to be developed by modelers. This step is the main critical stream of the modeling process and requires deeply experienced skills.

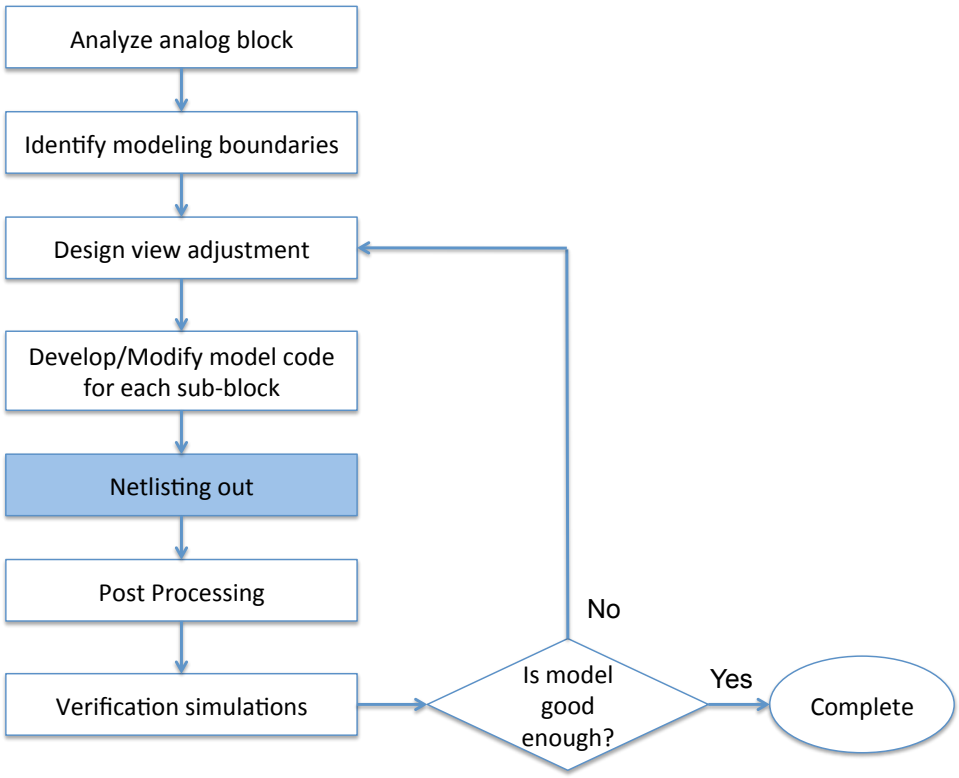

Figure 7.6: Behavioral modeling flow

When the models for all critical blocks are ready, the whole design can be extracted by an electronic design automation (EDA) tool. This extrac-

tion process is referred as "Netlisting". Through this netlisting process, the connections in the schematic design can be extracted as a readable netlist including the developed models and standard cell libraries used in the analog designs. In this research, Cadence OSS netlister is used. Since the output of the netlister is a Verilog AMS code, a post-processing is required to convert the Verilog AMS code into a Verilog code. And then, this generated Verilog model has to be verified through various test cases and validated by multiple iterations of modification, generation, and verification. This verification procedures are also the responsibility of the modelers, because the modelers know the detailed modeling scope and modeled design specifications. This model-based verification process was done by me, since it requires a significant experience. Figure 7.6 shows the modeling flow described so far. In this diagram, only netlisting step relies on a commercial tool. All other steps are completed by the modeler which I contributed for this research. In order to get a complete model of the given analog core, the model of each sub-block at the identified modeling boundary has to be implemented and verified by looping the sequence described in Figure 7.6.

7.5 Model Verification

Without validating the model by extensive verification cases for all specifications, the model is not guaranteed to function correctly and match with the actual design. Without enough level of verification, the model should not be deployed to upper-level design blocks for integration.

Assertions in the model flag simple low-level problems that would be tedious to check visually. For example, power supply connections can be visually checked for each block, but if the model has the assertions for the power supply connections, the power line connectivity checks can be easily performed. Simulation symptoms by verification cases would flag high level problems showing real issues in the design or model. Through these model-based verifications, any undetected bugs in the design or testbench can be revealed before the design goes to actual fabrication. The most common problems in the analog design are incorrect MUX selection logic, swapped connections in the bused signals, inverted reset polarity, and so on.

The development of verification cases can be generated by going through the key critical specifications. However, some generic process should make sure there is no holes in the model-based verification. Code coverage analysis can find out any potential holes in the verification plans or any logical redundancy or uncovered logical implementation. Code coverage analysis checks various aspects of coverage numbers such as statement coverage, conditional coverage, and branch coverage. This code coverage analysis was performed by me using the supporting feature in commercial EDA tools. The next Chapter shows more detailed verification cases with actual large scale mixed-signal design.

Chapter 8

Mixed-Signal Design Verification by Analog Behavioral Model

Verification of mixed-signal designs with clearly separated analog and digital sections was possible in the past, using independent analog and mixed-signal methodologies. Today, analog and digital functionality is tightly integrated throughout the entire design at different levels of hierarchy, and cannot be verified separately [2, 37]. In comparison with the total chip development effort, the portion of effort spent in design verification is growing at a faster rate and consuming a significantly larger portion of the development cost.

Design verification is a necessary process to ensure a quality design. Usually, digital design is accompanied with well-defined verification plan and thoroughly verified from multiple design stages. However, the verification of mixed-signal design requires slow transistor-level simulations or repetitive Monte Carlo simulations. Within a given short time for product market, it is not feasible to cover all verification aspects by those analog simulations. Although those analog simulation methods covers performance related verifications, analog behavioral model can achieve fast simulations for functional verifications. By virtue of fast simulation time with analog behavioral model,

such functional verification may uncover potential design bugs by applying comprehensive test cases with quick turnaround time.

As the design specifications are getting complex and support many features with complicated setups, the functionalities of those mixed-signal cores are partitioned into multiple different sub-blocks and implemented by suitable design flows such as finite state machine (FSM) based digital designs for complex controllability and transistor based analog designs for highly tunable designs. At the end, these sub-blocks need to be integrated together and verified for its functionality at the full design level. In complex mixed-signal design cases, comprehensive verification at the full design level is not doable simply. This chapter describes how the analog modeling can be beneficial for mixed-signal design verification.

8.1 Calibration Verification for Digital-to-Analog Converter

[1]A system-on-chip (SOC) supporting modern wireless communication systems includes a baseband Modem. A digital-to-analog Converter (DAC) core in the SOC converts modulated baseband wireless digital data streams into analog waveforms. Since the latest SoC with a modem supports many different communication specifications, it is very challenging to meet the dynamic requirement for the spectral quality of the converted analog waveform

[1]The work in this section was performed by me using the procedure that is described here.

from the DAC. Although the DAC core includes many advanced features to achieve the high quality of data conversion, a solid calibration capability is a key feature that cannot be skipped. The DAC design referred to in this chapter includes FSM-driven calibration features. Figure 8.1 shows key blocks in the DAC design.

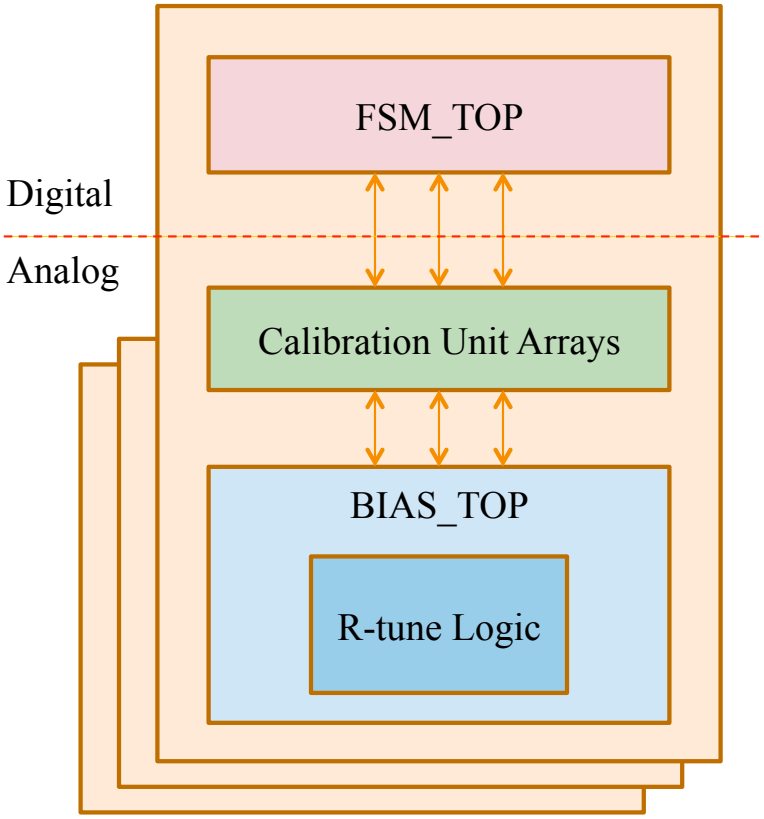

Figure 8.1: DAC block diagram

The FSM design is a purely digital design block and given by digital designers, and the DAC analog model includes the RTL code or synthesized netlist of the FSM block. The bias generation block is an analog circuit block

that generates and controls the required bias current and voltage for other DAC blocks. Too much detailed deep-down modeling of all the bias generation signals would slow down the model-based simulation performance. Hence, a simple equation based bias generation model and the model for R-tune logic are implemented for the bias generation block. The DAC core includes a deep hierarchical mixed-signal design including current source cells for each data bit. The modeling boundary is lowered as much as I can and the hierarchical schematic connections are fully preserved in the DAC analog model. All the blocks impacting key functionalities of the DAC core are implemented at each modeling boundary.

8.1.1 Silicon Mismatch Modeling

In the DAC design, the converted analog outputs are represented by the total summed current output from the current source circuits representing each data bit. The DAC output has two channels and each channel has 64 current sources inside. Then, those current values are added together to represent the analog output of the DAC. Although the current source circuit is identical for all data bits, process variations from manufacturing process may introduce some mismatches in the gain of the current source cells. The gate dimensions of metal oxide semiconductor field effect transistors (MOSFETs) suffer from random, microscopic variations, and hence, mismatches between the equivalent lengths and widths of two transistors that are identically laid out. Also, metal oxide semiconductor (MOS) devices exhibit a threshold volt-

age mismatch because the threshold voltage is a function of the doping levels in the channel and the gate, and these levels vary randomly from one device to another [38].

These random mismatches degrade the DAC's performance and impact the quality of the DAC output. In particular, the mismatches on the P-type MOS (PMOS) transistor of the current sources are dominant for the DAC output quality. With the calibration feature in the DAC design, those mismatches need to be compensated, and each device needs to be tuned to create an accurate analog output. To verify the full calibration features, the DAC-level simulations need to be performed by incorporating both digital FSM design and whole analog blocks. The digital FSM design code can be hooked up with the analog design, and if these full scale simulations with the FSM design and the analog circuits are performed in analog SPICE simulator, it may require a long repetitive Monte-Carlo simulations, and its simulation time can be very long from few hours to few days.

Figure 8.2 shows the PMOS-based current source design. The BIAS_TOP block in Figure 8.1 generates a bias current of 125 μA, and the ideal output of current source with no mismatch is 31.25 μA. In real silicon, this output value might vary because of the random mismatches on the PMOS transistors. Hence, the models for the PMOS-based current source design, compensation current generation block, and random mismatch insertion scheme are implemented. The calibration logic generates a compensation current based on the given calibration data and makes the summed current output to be tuned

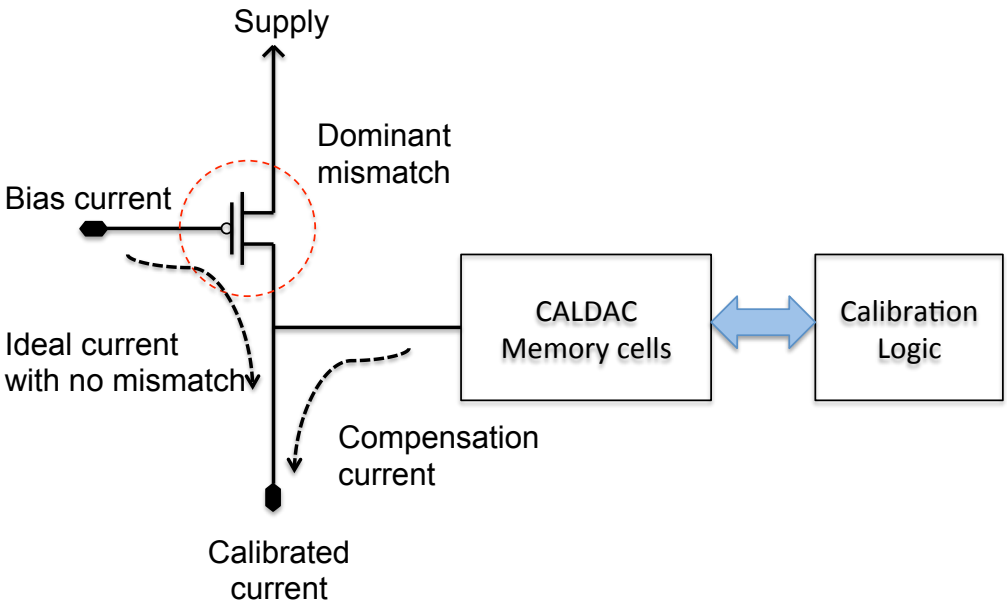

Figure 8.2: Modeling of mismatches in current source

within error range. Also, the calibrated data can be read back and stored into nonvolatile memory.

To model the random mismatch, random values having normal distribution are generated by Verilog built-in function, *$dist_normal* and added into the implemented Verilog model of the current source in Figure 8.2. This random mismatch inserted case is compared with the ideal case with no mismatch. The detailed simulation results are described in the next section.

8.1.2 Simulation Results

Figure 8.3 shows the calibration result by model-based calibration simulation. The X axis is the index of each current source, and the Y axis is the

current output value of each current source.

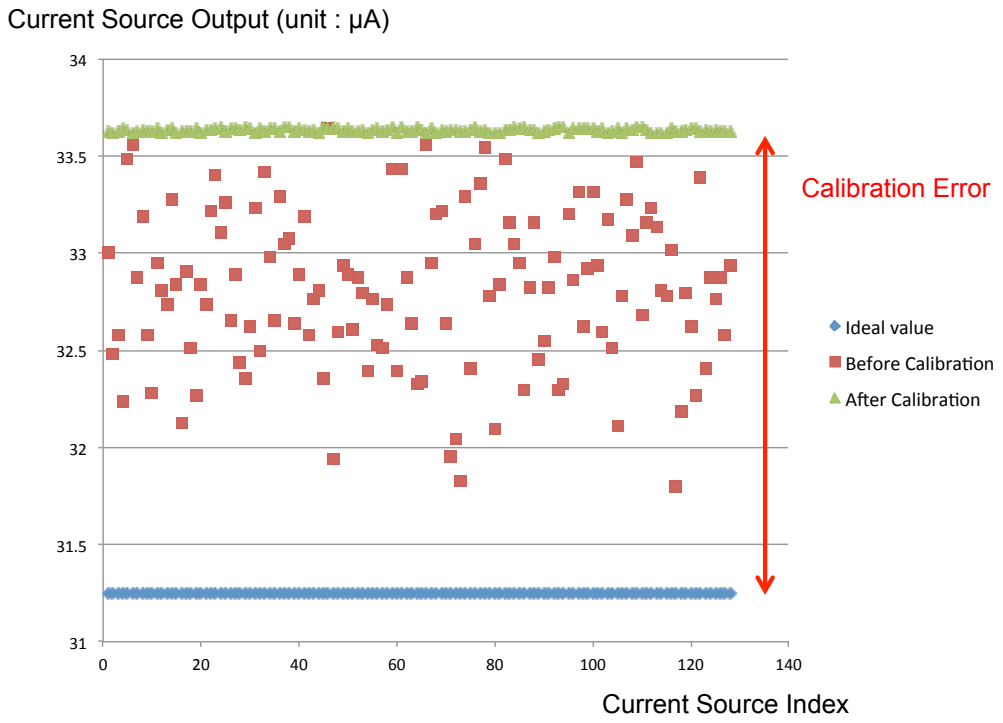

Figure 8.3: Offset errors from calibration

The initial gain values of the current sources in the DAC design are contaminated by random mismatches, and the required calibration sequences are performed by the DAC model. Because the random mismatch may have certain offset and variance, the ideal value at 31.25 μA is spread by the given distribution. Then, most significant bit (MSB) calibration is performed, and all those values are calibrated into the maximum value of the mismatched values. This MSB calibration adjusts only the variance of the mismatch values. In real silicon, the offset amount might not be big, but a little excessive offset

is used in this experiment for better visualization. As shown in Figure 8.3, if only MSB calibration is performed, calibration offset error can be left. This offset also needs to be calibrated by combining other types of calibrations.

To adjust the offset of mismatches, the resistance-tuning (R-tuning) calibration is combined with the MSB calibration. Figure 8.4 shows the combined result of MSB calibration and R-tuning calibration. The MSB calibration adjusts the variance of mismatches, whereas the R-tuning calibration adjusts the offset of mismatches. In this case, two different types of calibrations are applied in the following order:

- First R-tuning calibration
- MSB calibration
- Second R-tuning calibration

The first R-tuning calibration compensated the offset of the mismatches values, and the calibrated values by the first R-tuning calibration are spread with variance around the ideal values. Then, the MSB calibration is applied, and the variance of the mismatched values is compensated. As shown in Figure 8.4, the calibrated values by the MSB calibration still have some offset error because the MSB calibration converged the mismatch values into the maximum value among the mismatch values. To compensate this remaining offset further, the second R-tuning calibration is applied, and the final calibrated values are almost perfectly aligned with the ideal values. This multiple

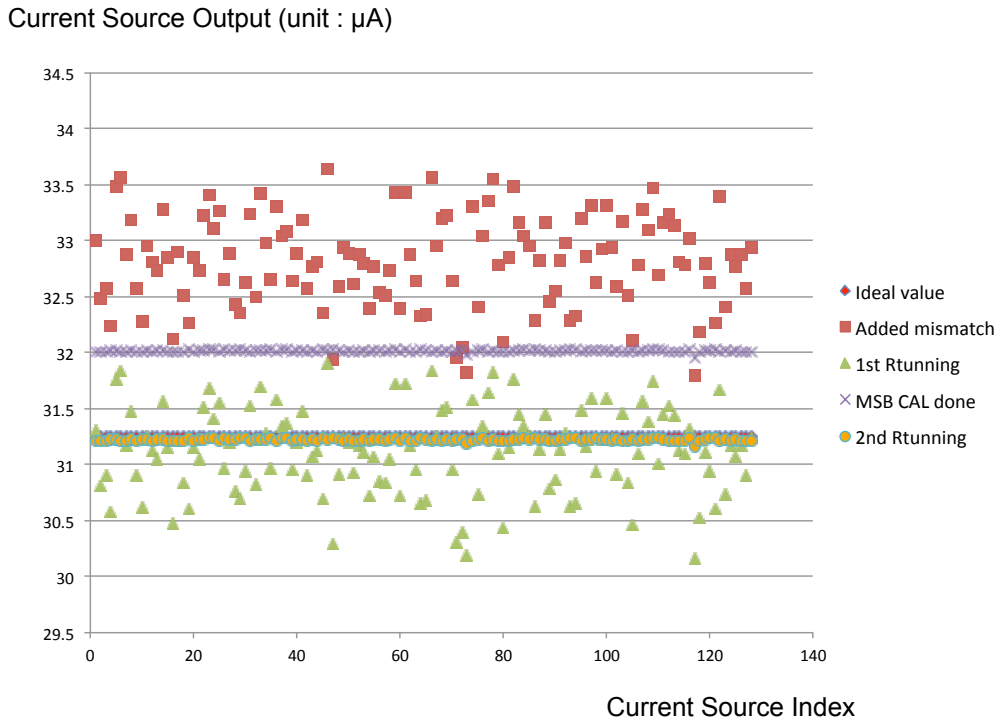

Figure 8.4: Calibration results

calibration sequence is controlled by the FSM block based on the programmed register values on the DAC core. The verification of this full calibration sequence is performed with the implemented behavioral model of the DAC core, and provided perfect calibration result as shown in Figure 8.4.

When the variance of the mismatch values is increased more, such highly varying values would exceed the DAC's calibration capability. Because this model-based simulation can be performed with a short turnaround time, different mismatch values can be applied to identify the saturation range of the DAC. With this excessive mismatch values, some outliers are shown as plotted

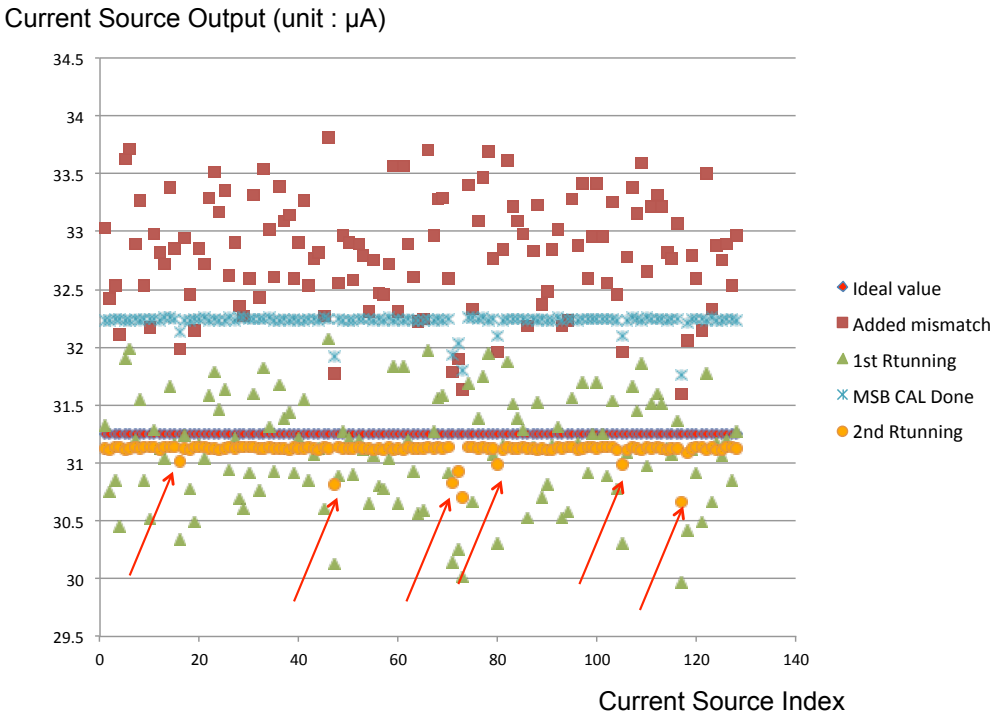

Figure 8.5: Calibration results with excessive mismatches

in Figure 8.5. By running these verification approaches, this saturation range of the DAC can be identified and used to predict the yield of the devices from process variation.

Figure 8.6 shows the analog outputs of the DAC from model-based verification simulation converted from digitized sine wave input under different gain settings. The adjusted bias current is generated from the bias block model for different gain settings, and the corresponding amplitude of the sine wave is measured for each case. In the waveform, the I channel output, "*i1*" and the inverted output, "*i1b*" show exactly opposite shapes and the sum of those

two current outputs from the DAC model always stays consistent as intended by the DAC design. The other Q channel has the same result.

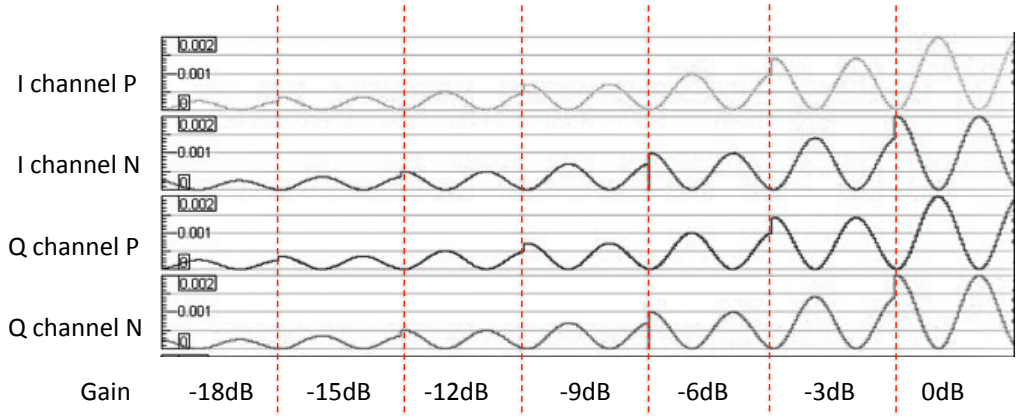

Figure 8.6: Converted DAC output from digitized sine wave

This section has described how the analog model of the DAC can accelerate the full verification of calibration processes by incorporating both digital FSM design and analog schematic designs. Furthermore, potential silicon mismatches are modeled and inserted into the model, and the convergence of the calibrated current values at each current source inside the design is verified. When this calibration verification is performed by SPICE simulation, it takes from few hours to few days. However, the model-based verification can be performed within few seconds. This model-based verification approach can be used to verify the validity of the calibration algorithm in the early design stage at architectural phase and the correctness of the initial design implementation.

8.2 Clock Generation Verification for Phase-Locked Loop

[2]The Phase-Locked Loop (PLL) is widely used to generate and distribute clocks in most high-performance digital systems. The PLL is an analog design core, but it has to be integrated with various digital blocks to enable the synchronous operation of the digital design blocks. Functional simulation and verification of the digital blocks require to have a compatible representation of the PLLs. The analog behavioral model of the PLL can be used for the development and verification of the digital systems. Because the latest PLL design supports many different configurable features, the PLL design itself needs to be verified via an analog model before the PLL is integrated into a whole system for manufacturing.

The PLL model can be beneficial from both top-down modeling and bottom-up modeling. In the early stage of design, the block-level digital design is ongoing, whereas the analog implementation of PLLs is not quite completed. In this case, the specification of the PLL is ready. Hence, top-down modeling can be performed based on the PLL specification and enables clocks for the digital design blocks. After a certain stage of the PLL design, the top-down model can be replaced by a more detailed and closer model to the actual design by bottom-up modeling.

[2]The work in this section was performed by me using the procedure that is described here.

8.2.1 Open-loop PLL model

As shown in Figure 8.7, a typical PLL design is controlled by a closed loop. In the diagram, the charge pump, loop filter, and voltage-controlled oscillator (VCO) blocks are implemented by extensive analog circuit. In addition, other blocks are implemented by mixed design.

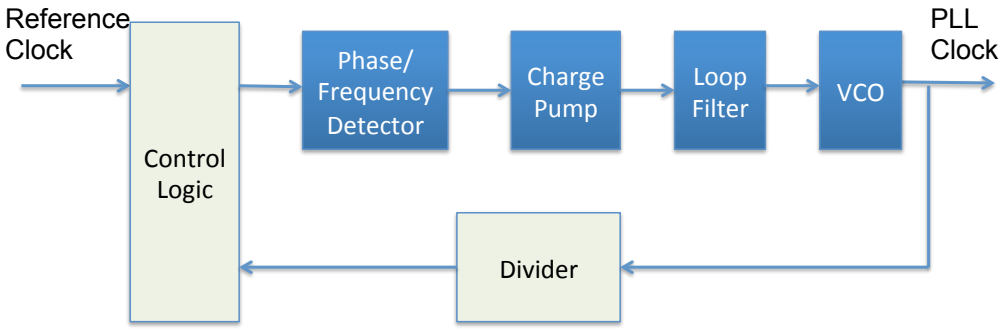

Figure 8.7: PLL block diagram

In this VCO-based PLL design, the detailed modeling of each analog block does not have much benefit. If this PLL design is hierarchically modeled with a closed-loop signal propagation, the model itself would add a significant simulation overhead to the SOC-level simulations. Especially, when many instances of PLL cores are integrated into the SOC level, this simulation overhead might be worsened. The accuracy and performance of the analog design portion of the PLL can be verified by analog simulation, and the behavioral model of the PLL can be achieved by open-loop control with full functionality via behavioral abstraction. The ultimate goal of the PLL model is to generate the correct PLL output clock with enough level of accuracy. This high-level

abstraction as an open-loop PLL model can be implemented quickly and enable early starting of design verification for upper-level systems. Figure 8.8 shows the diagram of the open-loop PLL model.

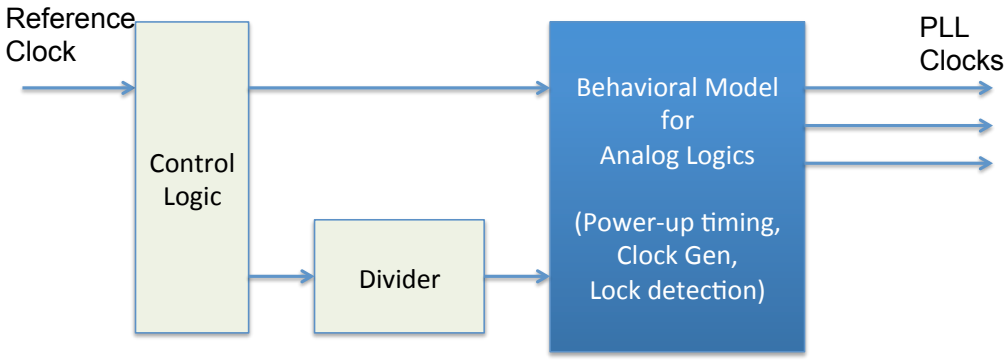

Figure 8.8: Open-loop PLL model

The abstracted PLL analog model includes all possible checks for the functional and electrical specification of the PLL analog features, such as power up timing sequence, PLL lock time, allowed voltage controlled oscillation (VCO) frequency range per given VCO mode, jitter filter, and analog behavior for losing PLL lock from changing the multiplier value, or drift of the reference clock. All these checks are implemented with various assertion schemes in the PLL model. When the PLL model is used in the stimulus-driven simulations, any incorrect sequences or settings can be filtered by the assertion messaging schemes. This can minimize the unnecessary debugging time for incorrectly generated test vectors that fail on silicon.

The PLL lock time is usually approximately 50 microseconds based on the design specification. Just for faster simulation speed, the PLL model

supports fast mode by adjusting the PLL lock time to 1 microsecond. Then, all the simulation-based verifications can be done quicker. Only the final simulation to generate a functional test vector needs to use the actual PLL lock time. Figure 8.9 shows the PLL frequency outputs by the PLL model within the allowed frequency range.

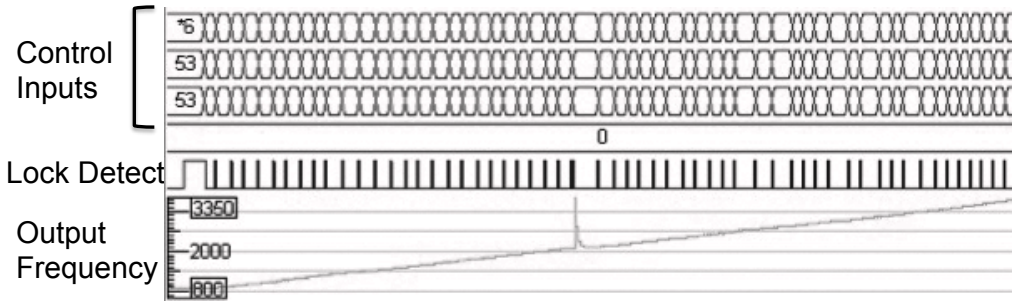

Figure 8.9: Frequency sweep with PLL model

Fast clocking for high-speed memory requires very accurate clock generation. In this case, the fractional portion of PLL multiplier is enabled to generate accurate frequency. To get enough resolution in the generated clock period, the simulation resolution in the Verilog model can be set to picosecond or femtosecond, but this finer resolution in the PLL model may slow down SoC-level simulations including the PLL model. Hence, proper scheme is included in the PLL model to avoid rounding error accumulation. If the simulation resolution is picosecond, a variation of 1 picosecond might happen depending on the given PLL setup. This is within the jitter spec of the PLL.

8.2.2 Closed-loop PLL model

For portable or mobile applications, lock-in time is very important since the PLL must support fast entry and exit from power management techniques [39]. PLLs often operates in a very noisy environment, and the digital switching noise coupled through power supply and substrate induces considerable noise into noise-sensitive analog circuits and results in the PLL output with high jitter. [40]. To improve the jitter performance of the PLLs, a narrow bandwidth was selected or a low-gain VCO was used. However, these approaches often result in long lock-in time and increasing design complexity of the PLLs.

Recently, all-digital PLLs have become more suitable because they yield better testability, programmability, stability, and portability over different processes [40]. Instead of using VCO, the all-digital PLLs use digitally-controlled oscillator (DCO) with fine-tuning capabilities. If the partitioning between analog blocks and digital blocks is planned well, a closed-loop PLL model can be implemented efficiently and enhance the flexibility and compatibility for better verification. Figure 8.10 shows the block diagram of DCO-based PLL design. In this design case, digital control logic requires accurate digital outputs from analog-to-digital converter (ADC) in analog blocks and generates corresponding coarse tune and fine tune as inputs to DCO. By virtue of these detailed modeling, each analog block is modeled at its boundary and fully closed-loop signal propagation is implemented. For this better modeling of the PLL, well-partitioned PLL design is provided by the analog designer of

the PLL.

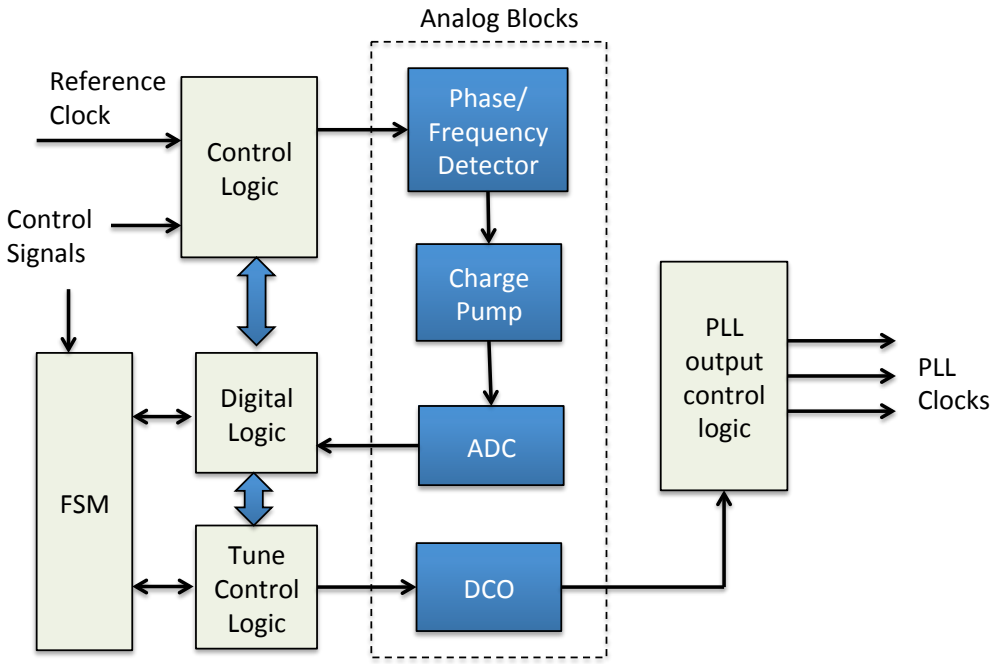

Figure 8.10: DCO-based PLL : open-loop PLL model

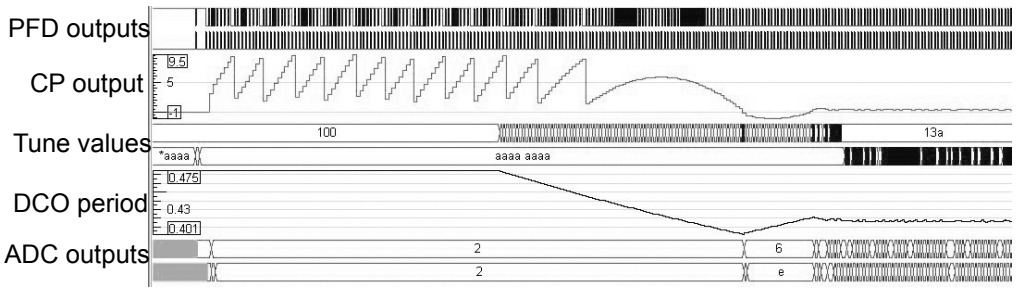

Figure 8.11: PLL waveform from analog blocks

All four analog blocks in Figure 8.10 are modeled with the behavior of each analog block, and the communicated signals between the charge pump

and the ADC are modeled as real numbers, which represent analog signals. All other signals between the analog blocks are modeled as digital signals and this closed-loop model provides quick lock-in time and accurate full functionality of the PLL.

A PLL locking simulation by the analog model of the PLL is performed. Figure 8.11 shows simulation waveform of the signals between analog blocks in Figure 8.10. For performance analysis of the analog blocks in the PLL, a full SPICE simulation for whole PLL design might not be feasible to be completed in reasonable time. When the SPICE simulation for the whole PLL with many different programmed configurations, it takes few hours to few days. In this design case, the PLL has well-partitioned fully hierarchical model. The analog model of the PLL enables a mixed-simulation with the model and design. The performance of each analog block can be assessed by placing actual schematic design for the target analog block and analog models for the rest of the design. This mixed simulation method provides faster simulation time with enough accuracy. The reduction of simulation time varies by the design size of the focused analog block, but it fairly reduced down the simulation time by 10 times or 100 times. The quality of the analog model of the focused block can be compared with the scaled-down SPICE simulation result and the quality of the model can be enhanced. Eventually, the mixed-simulation elevates both the quality of the model and accuracy of the design. Hence, the analog model is beneficial to both compatible integration and flexible analog design verification.

Chapter 9

Conclusion

This dissertation has presented several cost-effective production test solutions and mixed-signal design verification driven by analog behavioral modeling. Although the latest system-on-chip (SOC) is getting denser, faster, and more complex, the manufacturing technology is dominated by more subtle defects introduced by smaller scale technology and requires more mature testing strategies. By performing various different types of tests, better quality SoC can be manufactured, but test resources are too limited to accommodate all the required tests. To create the most efficient production test flow, any redundant or non-effective tests are removed or minimized. Testing of mixed-signal cores is becoming harder as their features and capabilities has grown and their scale is getting bigger. Before the testing phase of the design, better design verification can be enabled for mixed-signal cores by achieving comprehensive behavioral modeling.

Chapter 3 has proposed new method of test data volume reduction by combining the nonlinear property of feedback shift register (FSR) and dictionary coding. Instead of using the nonlinear FSR for actual hardware implementation, the expanded test set by nonlinear expansion is used as the

one-column test sets and provides big reduction ratio for the test data volume. The experimental results shows the combined method reduced the total test data volume and increased the fault coverage. Due to the increased number of test patterns, total test time is increased.

Chapter 4 has addressed a whole process of functional fault grading. Structural tests are widely used to cover the stuck-at and transition delay faults (TDFs) on the circuit under test (CUT), but for highly complex designs, deeply embedded logic may not be covered by the automatically generated tests. In particular, functional tests can be used to target critical functions and paths, but these functional tests require a big effort to generate and consume a lot of test time and test memory. By keeping the most critical functional tests, the remaining functional tests can be skipped if the test quality of each functional test can be assessed. Simulation-based functional fault grading is performed to identify the quality of the given functional tests against static faults and TDFs. With structural tests and functional tests, functional fault grading can indicate the way to achieve the same test coverage by spending minimal test time. For the final production testing, a confident decision on the functional test selection can be performed based on the fault grading results. Fault grading has always been a "desire-to-have" flow because it can bring up significant value for cost saving and yield analysis. However, it is very hard to perform the fault grading on the complex large scale SOC. A commercial tool called Z01X was used, but overall fault grading planning was organized and detailed execution was performed. Compared to the consumed time and

resource for fault grading, the contribution to the test time saving might not be acceptable as promising, but those fault grading data can be reused for yield analysis and test flow optimization.

Chapter 5 has addressed the challenges of package-on-package (POP) testing. Because POP devices have pins on both the top and the bottom of the package, the increased test pins require more test channels to detect packaging defects. Boundary scan chains are used to detect those continuity defects by relying on leakage current from the power supply. This proposed test scheme does not require direct test channels on the top pins. Based on the counting algorithm, minimal numbers of test cycles are generated, and the test achieved full test coverage for any combinations of pin-to-pin short defects on the top pins of the POP package. Also, it can be expanded to multi-site testing with fewer test channels for high-volume production. Boundary scan chains are given to most modern designs. Key point of this chapter is to maximize the usage of the boundary scan chains in the SOC, and the testing of top pins on POP devices has been achieved by sensing leakage current by scanning in minimum sets of stimulus patterns.

In Chapter 6, fault grading is applied within different structural test categories. Stuck-at faults can be considered as TDFs having infinite delay. Hence, the TDF ATPG tests can detect both TDFs and stuck-at faults. By removing the stuck-at faults detected by the TDF tests, the total test time for stuck-at faults is reduced, and the reduced stuck-at faults test set resulted in smaller number of stuck-at ATPG patterns. This TDF grading was performed

in the same ATPG tool that was used to generate the stuck-at and TDF ATPG tests. Without using other commercial tools for fault grading, the methodology proposed in this chapter was able to remove redundant test patterns and test time. By the proposed TDF grading, the reduced patter sets are generated while achieving the same test coverage.

In Chapter 7 and 8, the strategies and methodologies of analog behavioral modeling are addressed, and actual mixed-signal verification cases are presented. Recent complex SOC design includes various mixed-signal cores to implement highly complicated features to meet the needs of the latest mobile devices. Analog behavioral modeling enables wider scope of verification for the SOC having better quality. In the meantime, the modeling process reveals any potential design errors or incorrect test bench setup, and minimizes unnecessary debugging time with silicon triggered by the uncovered problems in analog cores. From the actual cases from digital-to-analog converter (DAC) and phase-locked loop (PLL), successful verification results are presented.

The calibration verification of DAC requires full design scale of digital finite state machine design and detailed representation of analog blocks. Fully hierarchical model for the DAC has proven quick simulation verification, and the actual calibration algorithm was verified through the modeling of silicon mismatches. The simulation by analog model is more than 100 times faster than SPICE simulation. This model-based verification can be used to verify the calibration algorithm in the early design stage for architectural phase and minimize unnecessary engineering time to identify escaped issues in the actual

design implementation.

Two different types of PLL model has been presented. Quick bring-up of open-loop PLL model has provided low simulation overhead for widely used PLLs in the SOC and enables early starting of design verification for the upper-level design using the PLL generated clocks. Accurate closed-loop PLL model had been achieved by DCO-based PLL design and sophisticated partitioning of digital and analog logics in the PLL design. By virtue of properly identified modeling boundary, the simulation overhead of the closed-loop model was not big enough to impact the SOC-level simulation and the mixed simulation with analog models and schematic designs enables both prompt verification and flexible performance analysis.

This dissertation contributes to reduce test time and to enhance test quality and helps to set up efficient production testing flow. Depending on the size and performance of the device, properly used testing schemes can maximize the efficiency of the production testing. The topics covered in this dissertation can be used in optimizing the test flow and making the list of final production tests that will achieve maximum test capability. Also, wider and better mixed-signal verification enabled by analog behavioral modeling elevates the quality of the device, and minimizes expensive engineering time caused by the errors in the device found too late in the development stage.

Bibliography

[1] W. Daehn and J. Mucha, "Hardware test pattern generation for built-in testing," in *Proceedings of the IEEE International Test Conference*, 1981, pp. 110 – 113.

[2] J. Chen, M. Henrie, M. F. Mar, and M. Nizic, *Mixed-signal Methodology Guide*, B. Bailey, Ed. San Jose: Cadence, 2012.

[3] M. L. Bushnell and V. D. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits.* Boston: Kluwer, 2000.

[4] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability.* San Francisco: Morgan Kaufmann, 2006.

[5] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuirs," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 19, no. 8, August 2000, pp. 957 - 963.

[6] A. Jas, J. Ghosh-Dastidar, and N. A. Touba, "Scan vector compression/decompression using statistical coding," in *Proceedings of the IEEE VLSI Test Symposium*, 1999, pp. 114 – 120.

[7] R. W. Basset, B. J. Butkus, S. L. Dingle, M. R. Faucher, P. S. Gillis, J. H. Panner, J. G. Petrovick, and D. L. Wheater, "Low-cost testing of

high-density logic components," in *Proceedings of the IEEE International Test Conference*, 1989, pp. 550–557.

[8] ——, "Low-cost testing of high-density logic components," in *IEEE Design & Test of Computers*, vol. 7, no. 2, 1990, pp. 15–28.

[9] H. Vranken, T. Waayers, H. Fleury, and D. Lelouvier, "Enhanced reduced pin-count test for full scan design," in *Proceedings of the IEEE International Test Conference*, 2001, pp. 738 – 747.

[10] H. Kaeslin, *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication.* Cambridge: Cambridge University Press, 2008.

[11] M. Cotorogea, "Using analog behavioral modeling in pspice for the implementation of subcircuit-models of power devices," in *IEEE International Power Electronics Congress*, 1998, pp. 158–163.

[12] X.-D. Tan and C.-J. Shi, "Parametric analog behavioral modeling based on cancellation-free ddds," in *IEEE International Workshops on Behavioral Modeling and Simulation*, 2002, pp. 25–31.

[13] H. Li, M. Mansour, S. Maturi, and L.-C. Wang, "Analog behavioral modeling flow using statistical learning method," in *11th International Symposium on Quality Electronic Design (ISQED)*, 2010, pp. 872–878.

[14] I.-S. Lee, J. H. Jeong, and A. P. Ambler, "Using the nonliear property of FSR and dictionary coding for reduction of test volume," in *Proceedings of the IEEE Symposium on VLSI*, 2005, pp. 194 – 199.

[15] L. Li and K. Chakrabarty, "Test data compression using dictionaries with fixed-length indices," in *Proceedings of the IEEE VLSI Test Symposium*, 2003, pp. 219 – 224.

[16] X. Sun, L. Kinney, and B. Vinnakota, "Combining dictionary coding and LFSR reseeding for test data compression," in *Proceedings of the IEEE Design and Automation Conference*, 2004, pp. 944 – 947.

[17] A. Chandra and K. Chakrabarty, "Reduction of SOC test data volume, scan power and testing time using alternating run-length codes," in *Proceedings of the IEEE Design and Automation Conference*, 2002, pp. 673 – 678.

[18] C. V. Krishna, A. Jas, and N. A. Touba, "Test vector encoding using partial LFSR reseeding," in *Proceedings of the IEEE International Test Conference*, 2001, pp. 885 – 893.

[19] A. A. Al-yamani, S. Mitra, and E. J. McCluskey, "BIST reseeding with very few seeds," in *Proceedings of the IEEE VLSI Test Symposium*, 2003, pp. 69 – 74.

[20] K.-T. Cheng, S.-Y. Huang, and W.-J. Dai, "Fault emulation: A new methodology for fault grading," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 18, 1999, pp 1487-1495.

[21] WinterLogic, *WinterLogic Z01X Manual.* WinterLogic, 2007.

[22] Synopsys, *Synopsys Tetramax Manual.* Synopsys, 2011.

[23] E. M. Rudnick, T. M. Niermann, and J. H. Patel, "Methods for reducing events in sequential circuit fault simulation," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 1991, pp. 546–549.

[24] M. S. Hsiao and J. H. Patel, "A new architectural-level fault simulation using propagation prediction of grouped fault-effects," in *Proceedings of the IEEE International Conference on Computer Design*, 1995, pp. 628–635.

[25] Y. E. Osais and A. H. El-Muleh, "A static test compaction technique for combinational circuits based on independent fault clustering," in *Proceedings of the International Conference on Electronics, Circuits, and Systems*, 2003, pp. 1316–1319.

[26] P. Sun, V. C. K. Leung, D. Yang, and D. X. Q. Shi, "Development of a novel cost-effective package-on-package (pop) solution," in *Proceedings of the International Conference on Electronic Packaging Technology & High Density Packaging*, 2009, pp. 46–51.

[27] N. Ahmed and M. Tehranipoor, "Improving transtion delay fault coverage using hybrid scan-based tech," in *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2005, pp. 187–195.

[28] M. M. V. Kumar, S. Tragoudas, S. Chakravarty, and R. Jayabharathi,

"Exact at-speed delay fault grading in sequential circuits," in *Proceedings of the IEEE International Test Conference*, 2006, pp. 1–10.

[29] L.-T. Wang, C. E. Stroud, and N. A. Touba, *System on Chip Test Architectures: Nanometer Design for Testability.* San Francisco: Morgan Kaufmann, 2007.

[30] I. Pomeranz and S. M. Reddy, "Transition path delay faults: A new path delay fault model for small and large delay defects," *IEEE Transactions on VLSI Systems*, vol. 16, pp. 98–107, 2008.

[31] C. E. Cummings, "Verilog nonblocking assignments demystified," in *Proceedings of Verilog HDL Conference and VHDL International Users*, 1998, pp. 67–69.

[32] ——, "Correct methods for adding delay to Verilog behavioral models," in *Proceedings of Verilog HDL Conference and VHDL International Users*, 1999, pp. 1–8.

[33] S. Joeres and S. Heinen, "Functional verification of radio frequency socs using mixed-mode and mixed-domain simulations," in *IEEE International Workshop on Behavioral Modeling and Simulation*, September 2006, pp. 144–149.

[34] B. Foret, L. Rolindez, C. Adobati, and S. Engels, "Unified environment for mixed signal top-level soc verification," in *IEEE Journal of Solid-state Circuits*, vol. 42, no. 5, May 2007, pp. 992–1002.

[35] G. Bonfini, M. Chiavacci, R. Mariani, and E. Pescari, "A mixed-signal verification kit for verification of analogue-digital circuits," in *IEEE Design Automation and Test in Europe*, Munich, Germany, March 2006, pp. 88–93.

[36] V. Sharma, G. Lakshmanan, S. Tare, and S. Dhamankar, "Predicting the correlation between analog behavioral models and SPICE circuits for robust SOC verification," in *IEEE International Behavioral Modeling and Simulation Workshop*, 2008, pp. 130–135.

[37] K. Kundert and H. Chang, "Verification of complex analog integrated circuits," in *IEEE Custom Integrated Circuits Conference*, September 2006, pp. 177–184.

[38] B. Razavi, *Design of Analog CMOS Integrated Circuits*. New York: McGraw-Hill, 2001.

[39] J. Dunning, G. Garcia, J. Lundberg, and E. Nuckolls, "An all-digital phase-locked loop with 50-cycle lock time suitable for high-performance microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 412–422, April 1995.

[40] C.-C. Chung and C.-Y. Lee, "An all-digital phase-locked loop for high-speed clock generation," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 347–351, February 2003.