Modern ASIC Design

Dian Zhou (周 电)
E. E. Department
The University of Texas at Dallas
USA
Chapter 8 Verification and Testing

- Objectives
  - Fault modeling and simulation
  - Test generation
  - Automatic-test-pattern-generation
  - Built-in-self-test
  - BIST architecture
  - Scan and boundary scan
  - Scan chains
  - Digital scan standard
Introduction

- Digital system verification and testing are progressively more important, as they become major contributors to the manufacturing cost of a new IC product.
- The emphasis on the quality of the shipped products, in addition to the growing complexity of VLSI design, requires testing issues to be considered early in the design process so that the design can be adjusted to simplify testing procedures.
Figure 8-1 Verification and testing in the design flow

- Fault modeling
- Fault simulation
- Test pattern generate
- Built-in-self-test
- Design for testing
- Boundary scan method
• Figure 8-2 demonstrates a VLSI development process in hierarchies.
• Based on customer or project need, a VLSI device requirement is determined and formulated as a design specification.

• Then, the designers synthesize a circuit that satisfies the design specification and verify the design.

• Design verification is a predictive analysis that ensures the synthesized design will perform the required functions when manufactured.

• When a design error is found, modifications to the design are necessary and design verification must be repeated.
• Once verification is done, the VLSI design is ready to be fabricated.

• At the same time, test engineers develop a test procedure based on the design specification and fault models associated with the implementation technology.

• Then, the chips that pass the wafer-level test are extracted and packaged.

• The packaged devices are retested to eliminate those devices that may have been damaged during the packaging process or put into defective packages.

• Additional Quality Assurance testing is used to assure the final quality before going to market.
A typical circuit testing process is illustrated in Figure 8-3, which consists of applying a set of test stimuli to the inputs of the circuit under test (CUT) while analyzing the output responses.

Figure 8-3 Basic testing approach (Wang, Wu, & Wen, 2006)
Fault Model

• Because of the diversity of VLSI defects, it is difficult to generate tests for real defects.
• Fault models are necessary for generating and evaluating a set of test vectors.
• There are many types of fault models, like “open and short fault”, “bridging fault”, “delay fault”, “coupling fault”, and so on.
• Generally, a good fault model should satisfy two criteria:
  – It should accurately reflect the behavior of the defects.
  – It should be computationally efficient in terms of fault simulation and test pattern generation.
• Because of the diversity of VLSI defects, it is difficult to generate testing patterns for real defects.
  – Fault models are necessary for generating and evaluating a set of test vectors.

• Good fault models can abstract physical defects in the circuit at a high level and represent a high percentage of the actual physical defects that can occur in components.
  – They allow test generation and fault/coverage analysis to be done early in the design process.

• A combination of different fault models is often used in the generation and evaluation of test vectors and testing approaches developed for VLSI devices.
• An example

Figure 8-5 An example of a single stuck-at fault.
• How to find the fault

  – Testing vector

Consider the circuit in Figure 8-5. Three stuck-at faults (d, e, and f) are marked in the circuit. It is apparent that the fault-free function of output \( y = (AB) + (B + C) \). The faulty functions are (note: the fault \( d \ s-a-1 \) can also be written as “\( d \ SAI \)”, or “\( d/1 \)” for short):

\[
d/1 \text{ fault: } y = (A \cdot 1) + (1 + C) = A + 1 = 1
\]

\[
e/1 \text{ fault: } y = (A \cdot 1) + (B + C) = A + B + C
\]

\[
f/0 \text{ fault: } y = AB + 0 = AB
\]
• Fault analysis procedure
  – The stuck-at fault model, which has been successfully and most commonly used for decades, is a logical and easy-to-understand fault model.
  – The process of a stuck-at fault model testing is shown in Figure 8-4.
START

Circuit Under Test (CUT)

Fault Modeling & Collapsing

Select a fault (e.g. F/0 here)

Loop: Select another fault and repeat

Generate a Fault List that can be detected by the test vector

Test Simulation

Mark those faults detected and remove from the fault list

FINISH: when all faults are detected or proven untestable

Test Generation (for the selected fault)

Figure 8-4 Fault analysis procedure
Fault Simulation

• Logic simulation for a combinational logic circuit is the determination of steady-state logic values implied at each circuit line by the vector applied to its primary inputs.

• A high-quality fault simulation to predict the faulty circuit behavior can greatly improve testing and diagnosis.

• The purpose of fault simulation is to evaluate the effectiveness of a set of test patterns in detecting manufacturing defects.

• The quality of a test set is expressed in terms of fault coverage, the percentage of faults that causes an incorrect output if the test set is applied.

• Furthermore, it helps identify undetected faults. In this case, the test designer has to generate additional test vectors to improve the fault coverage.
Automatic-Test-Pattern-Generation

• Automatic test equipment (ATE) is computer-controlled equipment used in the production testing of ICs (both at the wafer level and in packaged devices) and PCBs.

• Test patterns are applied to the CUT and the output responses are compared to stored responses for the fault-free circuit.

• Generating effective test patterns efficiently for a digital circuit is thus the goal of any Automatic-Test-Pattern-Generation (ATPG) system.

• A powerful ATPG can be regarded as the Holy Grail in testing, with which all Design-for-testability (DFT) methods could potentially be eliminated.
Built-In-Self-Test (BIST)

- As mentioned before, testing ought to be quick and have very high fault coverage. One approach is to inset a testing circuit as one of the system functions, so it becomes capable of self-test.
- Built-In-Self-Test (BIST) refers to techniques and circuit configurations that enable a chip to test itself.
- BIST techniques can be classified into two categories, online-BIST and offline-BIST.
  - Online-BIST includes concurrent and nonconcurrent BIST, whereas offline-BIST consists of functional and structural approaches.
The Concept of BIST

- Traditional test techniques using ATPG software to target single faults for digital circuit testing have become quite expensive and can no longer provide adequately high fault coverage for deep submicron or nanometer designs.

- One practical approach to solve these testing problems is to incorporate BIST features into a digital circuit.

- Logic BIST is a design for testability (DFT) technique in which a portion of a circuit on a chip, board, or system is used to test the digital logic circuit itself.
  
  - In this methodology, test patterns are generated on-chip and test responses are also analyzed on-chip. The basic of BIST designs has a Test pattern generator (TPG) and an output response analyzer (ORA) as shown in Figure 8-23.
- BIST system structure

Figure 8-23 A structural offline-BIST system
• BIST offers several advantages over testing using automatic test equipment (ATE).
  
  — First, in BIST the test circuit is integrated on-chip so that no external tester is required.
  
  — Second, a self-testable chip enables self-test to execute even after it is built into a system. This can be used either for periodic testing or to diagnose system failures.
  
  — Third, self-test can be performed at the circuit’s normal clock rate, since it’s getting more and more difficult for ATE to keep pace with the increasing circuit speeds.
Figure 8-24 gives a typical BIST hardware in more detail.

- A comparator compares the signature produced by the data compacter with a reference signature stored in a ROM during BIST.

![Figure 8-24 BIST Process](image)
Scan and Boundary Scan

• Testability is a design feature which influences the cost of testing.

• Two important attributes, controllability and observability, are highly associated with testability.
  
  – Controllability for a digital circuit is the difficulty of setting a particular logic signal to ‘0’ or ‘1’.
  
  – Observability is the challenge of observing the state of a logic signal at a particular point in the circuit.
• Those circuits have difficulty to control are decoders, circuits with feedback, oscillators, and so on; whereas low observability circuits are sequential circuits, embedded RAM, ROM or PLAs, etc.

• Control logic, random logic, and asynchronous design are more difficult to test than the combinational logic, data-path logic, and synchronous design.
• The main idea in scan DFT design is to obtain controllability and observability for flip-flops.
  
  – This is done by adding a test mode to the circuit, in addition to its normal mode. In the normal mode, the flip-flops are connected as shown in Figure 8-46.

  – During this mode, the response at the state outputs ($Y_1$ to $Y_k$) is captured in the flip-flops. These values can be observed by switching the circuit to test mode, whose flip-flops are reconfigured as one or more shift-registers, called scan registers or scan chains.

  – In addition, values to be applied at the state inputs in the subsequent test may be simultaneously shifted into the flip-flops.

  – Thus, for the purposes of test development, the state inputs and outputs can be treated as being similar to primary inputs and outputs, respectively.
• Normal sequential logic structure

Figure 8-46 A special view at a general model of a sequential circuit (Jha & Gupta, 2003)
• Replace the normal flip-flop by scan flip-flop

(a) A view of scan flip-flop
- A sequential logic with scan flip-flop

Figure 8-47 (a) A view of scan flip-flop, (b) a scan structure working under normal mode and (c) a scan structure working under test mode.
• General logic structure with “scan capability”
Operation of the circuit with scan flip-flop

Figure 8-49 Combinational test vectors ($x_i$, $y_i$, $z_i$, and $Y_i$ have the same meaning as it in Figure 8-46)
Scan Chains

- Scan testing is very useful, but it brings higher costs, which can be categorized into two types.
- First, the scan hardware increases the chip size (area overhead) and second, it slows the signals down (performance overhead).
- Area overhead is typically found to be 5% to 10%, because of using “scan flip-flops” and wiring them.
- Adding multiplexer delay into the combinational path results in approximately two gate-delays, and flip-flop output loading due to one additional fanout produces approximately 5% to 6%.
- Overall, scan design can cause a 5 to 10% reduction in speed.
To overcome the above shortcomings, scan flip-flops can be organized into chains by using Partial Scan, partitioning them into multiple chains, ordering flip-flops within each chain, and using a reconfiguration circuit.

For ease of representation, combinational circuit elements are combined into a number of combinational blocks using the following procedure.

- First, each combinational circuit element is treated as an individual block. If any element in one block is hooked up via a combinational connection to any element in another block, then these two are combined into a single block.

- This process is repeated until no two blocks can be combined, i.e., blocks of maximal size are obtained. Let $CLB_1, CLB_2, \ldots, CLB_N$ be the Combinational Logic Blocks obtained in this methodology. Subsequently, the flip-flops may be combined into multi-bit parallel-load registers (Jha & Gupta, 2003).
An example

Figure 8-51 An example circuit with two CLBs and five registers configured as a single chain (Jha & Gupta, 2003)
• An outline of a typical test procedure using a boundary scan is as follows:

  – A boundary-scan test instruction is shifted into the IR through the TDI.
  – The instruction is decoded by the decoder associated with the IR to generate the required control signals so as to properly configure the test logic.
  – A test pattern is shifted into the selected data register through the TDI and then applied to the logic to be tested.
  – The test response is captured into some data register.
  – The captured response is shifted out through the TDO for observation and, at the same time, a new test pattern can be scanned in through the TDI.
  – Steps 3 to 5 are repeated until all test patterns are shifted in and applied, and all test responses are shifted out.
Figure 8-57 Boundary-scan architecture

Note: The four terminals—test data input (TDI), test data output (TDO), test mode select (TMS), and test clock (TCK)—are mandatory terminals of TAP.
Summary

• In this section we have discussed the verification and testing.
• Each of these two subjects itself is a deep and broad area in VLSI design.
  – What has been discussed in this chapter is only introductory material, but is self-contained.
• Today, most testing circuits and testing vectors are generated automatically by CAD tools.
• It is necessary for a designer to understand what circuit has been inserted into the chip for testing and how it works.
• Readers can try to design and insert the testing circuit in the project MSDAP.
1. Develop a fault collapsing in an exclusive-OR (XOR) circuit, and see how many faults can be removed from the six total faults.

2. Perform the equivalence and dominance fault collapsing for the following circuit, with Checkpoint Theorem.

Figure 8-58 Problem from (Bushnell & Agrawal, 2002)
3. Insert the scan train into the design of MSDAP.

4. Use fault collapsing to find the fault list of the circuit, and then use Non-Structural Test Generation (Boolean Algebra, XOR) to find the least number of test patterns to detect all faults.