Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 1 (2025)

# Overview of Design for Testability, ATPG Flow, Pattern Generation and Translation

**Dr. Sandeep Santosh**

Department of Electronics and Communication, National Institute of Technology, Kurukshetra, Haryana, India.

Email: profsandeepkkr@gmail.com

***Article History:***

***Received: 19-01-2025***

***Revised: 24-02-2025***

***Accepted: 21-03-2025***

**Abstract:**

In today's VLSI SoC designs, Design for Testability (DFT) is a crucial methodology aimed at simplifying the testing of digital circuits during the manufacturing and debugging phases to ensure their functionality and performance. This technique primarily focuses on identifying whether a fabricated device is defective. By implementing DFT, testing costs and time can be significantly reduced, which in turn enhances manufacturing yield and accelerates time-to-market. Debugging Very Large Scale Integration (VLSI) circuits can be a complex and time-intensive task. Therefore, the strategy is to address failures early in the simulation phase, which can substantially decrease the time required for debugging compared to on-silicon debugging.

Keywords: Design for Testability (DFT), ATPG Flow, Very Large Scale Integration (VLSI), Pattern Generation, Translation.

## 1 Introduction

Design for Testability (DFT) is an essential methodology aimed at simplifying the process of testing digital circuits during manufacturing and debugging. Its primary focus is to determine whether a fabricated device is defective while ensuring that the circuit functions as intended. By incorporating DFT techniques, manufacturers can significantly reduce testing time and costs, leading to improved production yields and shorter time-to-market.

Implementing DFT requires careful consideration of several factors, including test coverage, fault models, and the trade-offs between enhancing testability and maintaining other design priorities like performance and power efficiency. As technology nodes shrink and designs grow in complexity, testing becomes increasingly challenging and time intensive. Identifying faulty chips under these conditions necessitates robust testing mechanisms.

To address these challenges, DFT introduces additional logic to the design without altering its core functionality. This added logic improves the testability of the circuit by enhancing its controllability and observability. By ensuring the quality of the manufactured chip, DFT plays a vital role in maintaining design reliability and achieving superior product performance.

Design for Testability (DFT) is a critical component in the design of integrated circuits (ICs), aimed at enhancing the ease and effectiveness of testing to ensure the reliability and functionality of chips. Fault models play a vital role in DFT by providing a framework to simulate and understand potential defects in a circuit. Here's an overview of some common DFT fault models: m, ensuring that the memory can test itself during manufacturing, in-system operation, and during periodic maintenance.[1]

## 2 Objective of the Work

a) To create or adapt system-on-chip (SoC) compatible test patterns for effective testing.

b) To reconstruct a virtual SoC-level design using netlists to ensure proper integration.

c) To validate the programming of test data registers for accurate functionality.

d) To simulate and verify both the functional and DFT logic within the design, ensuring reliability and testability.

## 3 Scan Cell and Chain

### Scan Cell

A scan flip-flop is a key element in creating a scan chain. Modern scan architecture typically employs two main types of designs: Mux-D flip-flops and Level Sensitive Scan Design (LSSD). Mux-D flip-flops are edge-triggered and incorporate a two-input multiplexer before the data input. The multiplexer is controlled by a selection signal, commonly referred to as "scan enable," "shift enable," "scan mode," or "test mode." When this signal is low, the flipflop operates in its normal mode, allowing functional data to pass through. Conversely, when the signal is high, it switches to scan mode, directing scan-in data through the flip-flop. The flip-flop uses a single clock for both normal operations and shifting scan data. Additionally, both functional and scan data are conveyed through a single output. The typical design of a Mux-D flip-flop uses "scan en" for enabling the scan mode or shift, with "SI" and "SO" representing the scan data input and output, respectively, as illustrated in Figure 1.

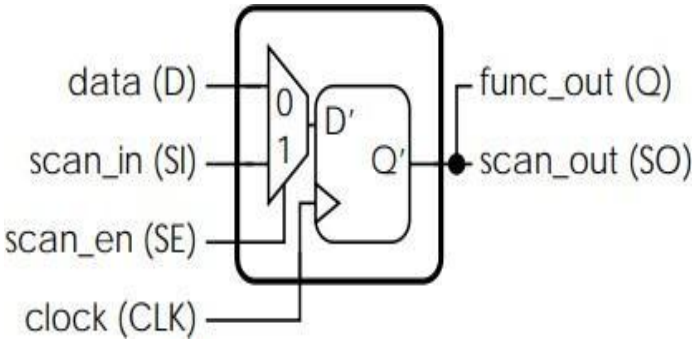


Figure 1 Scan cell design

Mux-D flip-flops are widely favored because they offer minimal area overhead, requiring only an additional selector signal for routing to each flip-flop. This selector typically has few timing constraints. However, if delay faults need to be observed by initiating transitions during the final shift cycle in scan testing, the selector should be routed similarly to a clock tree. Addressing hold-time issues in Mux-D flip-flop-based scan chains can be challenging, as simply reducing the scan shift frequency is not a viable solution, often necessitating a design overhaul. The inclusion of a multiplexer in the functional data path can increase data path delay and potentially lower the maximum frequency for functional operations. Since only one clock tree supports both functional and scan shift modes, increasing expected scan frequencies to expedite scan test execution might be labor-intensive and could negatively impact power consumption.

In contrast, LSSD (Level-Sensitive Scan Design) scan cells utilize level-sensitive latches and do not require a selector signal. Instead, their operation is governed by three clocks, allowing the cell to retain functional data, store scan data, or propagate scan data to a specific scan output, depending on the clock configuration. In the example LSSD cell shown in Figure 2, there are two D-latches, and the latch that retains data features two input ports: one for functional data and another for scan data.

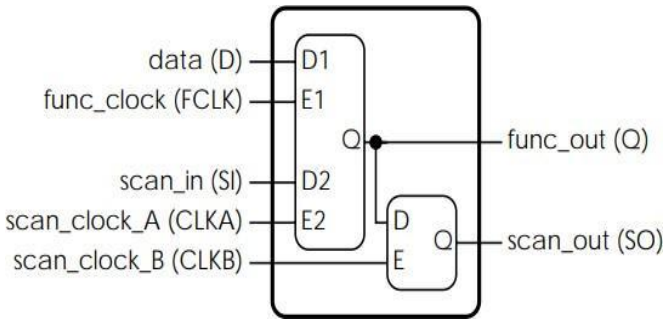


Figure 2 LSSD cell

While LSSD scan architecture can be used in high-performance systems, it is less commonly adopted. This approach comes with its own set of advantages and disadvantages:

**Advantages:**

- No extra delay in data paths.
- Avoidance of hold-time issues.
- Separate clock trees simplify timing constraints throughout the design process.

**Disadvantages:**

- Greater area overhead.
- Requirement to route two additional clock trees.

**Scan Chain**

Figure 3 illustrates a design lacking scan flip-flops, where shifting data in and out presents difficulties, thus diminishing the design's controllability and observability. This challenge led

to the introduction of scan-enabled flip-flops. In a full-scan design, scan flip-flops replace standard flip-flops. As shown in Figure 4, these scan flip-flops are linked together to form a scan chain. During test mode, with the test enable signal (SE) active, the scan chain functions as a shift register. The scan input is connected to the first flip-flop in the chain, while the scan output is linked to the final flip-flop.

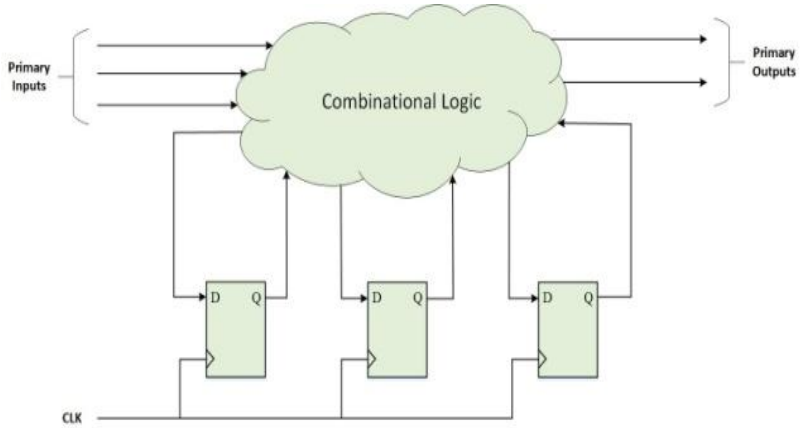


Figure 3 Scan chain without scan

Partial scan designs, on the other hand, are those that contain some flip-flops that are purposefully not modified to scan flip-flops. A design's capacity to be tested for manufacturing

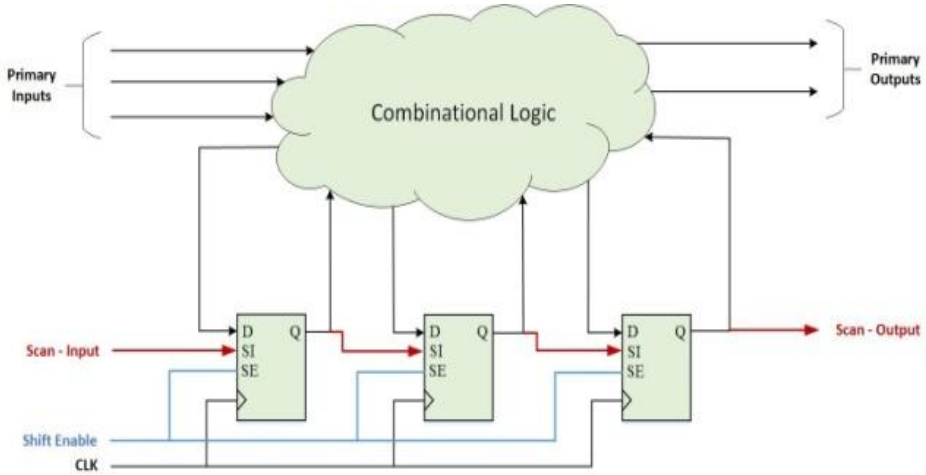


Figure 4 Design with scan

**Role of Scan Chain in DFT**

A scan chain is a series of sequential components, like flip-flops or latches, connected in sequence, each with an input and output. It allows for the recording and serial shifting out of the state of each flip-flop or latch for testing purposes. Scan chains are vital in Design for Testability (DFT) for several reasons:

**1. Improved Testability:** By providing access to the internal states of the circuit, scan chains improve the circuit's testability. This access makes it easier to detect and diagnose defects in the circuit.

**2. Reduced Test Time:** Scan chains decrease the overall number of test vectors required, allowing for more efficient testing. Test vectors can be shifted in and out serially through the scan chain, thus shortening the test duration.

**3. Simplified Evaluation:** They offer a standardized and easily automated method for assessing the circuit, which reduces both the time and cost of testing.

**4. Effective Troubleshooting:** With scan chains, designers can monitor the internal states of a circuit during operation, facilitating troubleshooting efforts.

Scan chains are a critical element in DFT, enabling the efficient and effective testing of digital circuits. Without them, testing large digital circuits would be more complex, time-consuming, and costly.

**Scan Chain Functioning**

The scan chain's functioning may be broken down into three phases, namely:

**1. Scan In:** While the design is in test timing mode, test patterns are loaded at this stage.

**2. Capture:** While the design is still in functional timing mode, the goal of this stage is to record how the design reacts to the test pattern.

**3. Scan Out:** This phase's primary goals are to unload the pattern response and return the design to the test timing mode. In rare circumstances, this step might also start the Scan In procedure, which will inject the next test pattern.

For stuck-at testing, a single clock capture pulse is adequate because stuck-at testing is done at a slower frequency. But when testing for path delay or transition defects, functional speed is employed (also known as at speed testing), and the design is placed in functional timing mode with functional frequency. In the capture mode, this calls for two or more functional clock pulses.

The execution of the capture timing mode for a scan chain differs somewhat for slow capture (for stuck-at faults) and at-speed capture (for path delay or transition faults). This will be covered in more depth during the conversation about the capture stage.

In figure 5 the scan chain functioning is shown with description of three phases.

**Stage-1 Scan In:**

The loading stage, also known as the scan-in stage, serves the primary function of loading the appropriate test vectors. In figure 2.6, the SE signal is maintained high throughout this phase (asserted) to ensure the scan flipflop only recognizes the SI signals as inputs. All of the combinational logic between the flipflops is bypassed by doing this. The next flip-flop's test patterns are then moved from SI to SO followed by SO to SI, creating a chain that serves as a shift register.

Test patterns go to the initial flip-flop in the scan chain after serially entering through the scan input port. They advance to the subsequent flip-flop stage for every active clock edge (shift register behavior), and so on, until all parts of the scan chain are loaded with the proper test patterns.

For suppose there are n flip-flops in the scan chain, the test vectors are going to reach the SI

(Q) pins and related combinational logic within the flip-flops after n-1 clock pulses. 'The loading of test patterns' is the technical term for this procedure.

**Stage-2 Capture:**

Slow-Stuck-at Capture**:** This mode operates on a slower frequency and is used to identify stuck-at problems. It only lasts for one clock cycle (CK) in this mode shown in figure 6. The flipflops function in the regular functional timing mode when the SE (enable signal) is low during this mode. The next flip-flop FF2/Q then records the test pattern response after it has been processed via the combinational logic. Prior to the next active clock edge arriving, the next flip-flop's D input receives the test pattern response after the combinational logic has processed it. The processed test pattern response is stored in the following flip-flop until the active clock edge comes, at which point it may be accessed at the Q and SO pins.

At Speed-Transition/Path delay Capture**:** Path delay or transition defects are found using capture mode at the functional frequency. Because both launch and capture operations must be completed quickly in at-speed testing. The capture mode lasts for two or more clock cycles (CK) shown in figure 7. In order to enable the flip-flops to operate in the typical timing mode, SE (enable signal) is low during this mode.

Because there are two flip-flop steps involved, creating a pattern for this type of testing is difficult. In figure 2.7 the test pattern is launched to the specified combinational logic by the first capture pulse (CK) when SE is low by the FF2, and its response at FF2/Q(SO) is captured during the functional frequency by the second pulse (CK) when SE is low by FF3. The test pattern data launches from the flop's D to Q pin with the first capture pulse, and it is captured by the following flip-flop and sent to its Q and SO pins during its second capture pulse. With this method, the test pattern response can be generated and recorded at the functional frequency. The test pattern is introduced to the intended combinational logic by the first clock pulse, and the desired combinational logic is given at-speed processing time by the second clock pulse.

**Stage-3 Scan Out:**

The scan chains are emptied in the scan out mode. To return the design into test timing mode, SE (scan enable) gets high once more. At each active clock edge, the collected data, which comprises the test pattern response processed via combinational logic, is serially shifted out on the scan chain. In this mode, the test pattern response that was recorded at the flip-flops' SI pins is carried over to the scan output port so that it may be cross-checked against the anticipated outcomes shown in figures 6 and 7 respectively.

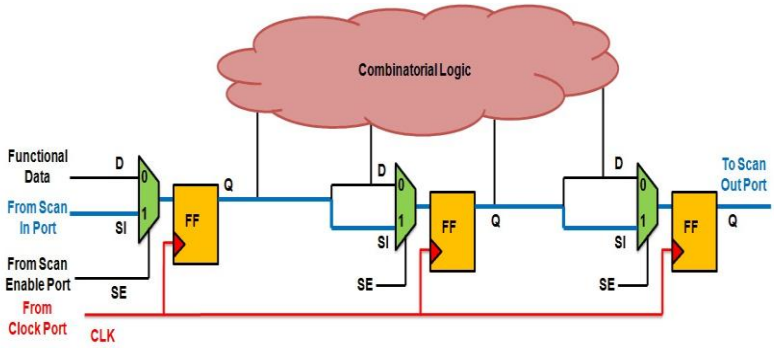


Figure 5 Scan chain functioning [5]

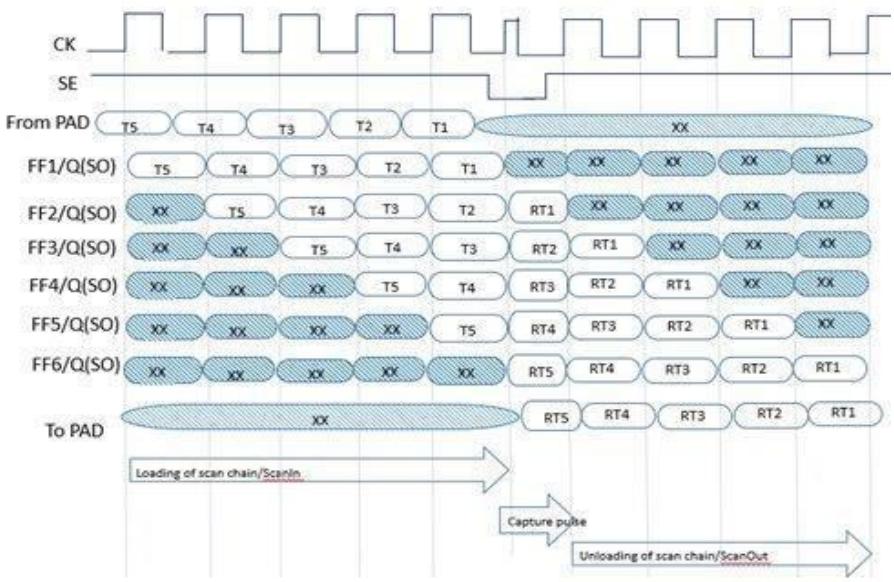


Figure 6 Stuck-at capture waves

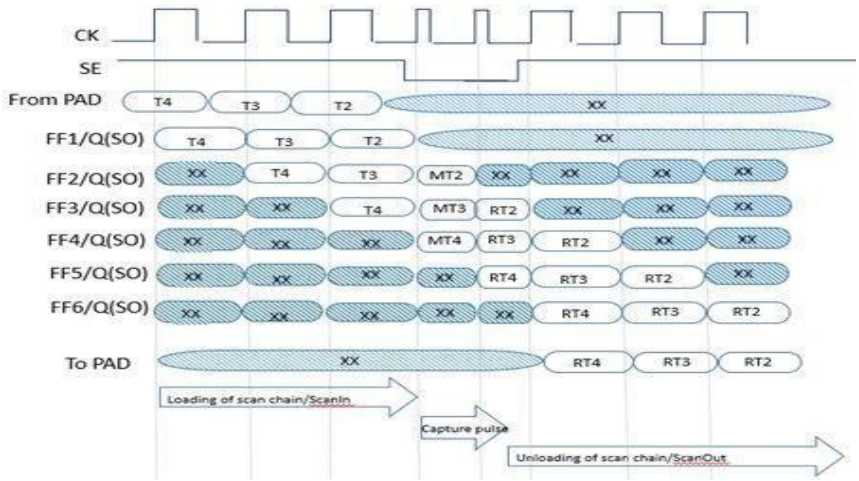


Figure 7 At-speed capture waves

**Complete Scan Design**

To obtain good fault coverage and a small test vector size, practically every flipflop in modern chip contains a significant amount of random logic and a high number of flipflops or latches are replaced with a scan cell. Then it is referred to as a "full or complete scan design".

In order to reduce the scan's overhead area or for performance reasons, flipflops and latches may be partially replaced by scan cells but not a part of scan chains. This implies less fault coverage and more scan patterns for sequential testing with multi-cycle capture sequences. The cost of area is no longer a justification for "partial scanning" given that the cost of area and the expense of a single transistor on the die have been falling dramatically for years, while the test costs have been falling at a much slower rate. It is now more efficient to use more "cheap" area for test logic so as to save "expensive" test time and tester memory. The implementation of "partial scanning" may still be justified, though, given the performance reasons indicated above.

## 4 ATPG Pattern Generation

### 4.1 Introduction

During the fabrication of integrated circuits, the imperfect manufacturing process can introduce defects that result in malfunctioning chips. The objective of test generation in digital circuit testing is to produce a set of test vectors that can identify any defects. This process, known as Automatic Test Pattern Generation (ATPG), aims to differentiate defective chips from defect-free ones by utilizing specific inputs.

Effective test pattern generation is crucial for uncovering faults, and ATPG systems are essential for achieving this goal. The complexity of generating test patterns has led to the development of Design for Testability (DFT) methods to ease the ATPG process. In an ideal scenario, a powerful ATPG would render DFT methods unnecessary by delivering test patterns that achieve high fault coverage with minimal test sets.

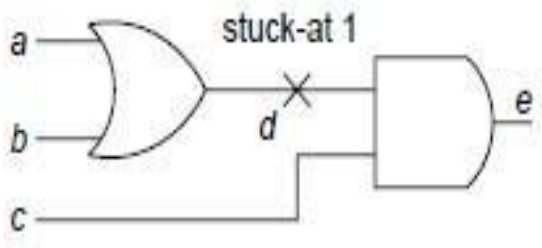


Figure 8 Example of a single stuck-at fault [3]

ATPG systems employ abstract representations of defects, typically referred to as faults, to simplify test generation. The single stuck-at fault model is widely used in this context. This model assumes that a circuit node can be tied to either logic 1 or logic 0, representing a fault. For example, in Figure 8, if a signal (d) is stuck at logic 1 (denoted as (d/1)), the test patterns

need to apply logic 0 to this node (a) and logic 0 to another node (b). To detect the fault at the output, apply logic 1 to node (c) and observe the output at node (e).

To detect all possible faults, ATPG attempts to create test vectors for each fault within the circuit. However, some faults might be logically equivalent, meaning no test can distinguish between them. Equivalent fault collapsing is a technique used to reduce the number of targeted faults by identifying these equivalences beforehand.

### 4.2 ATPG Basic Tool Flow

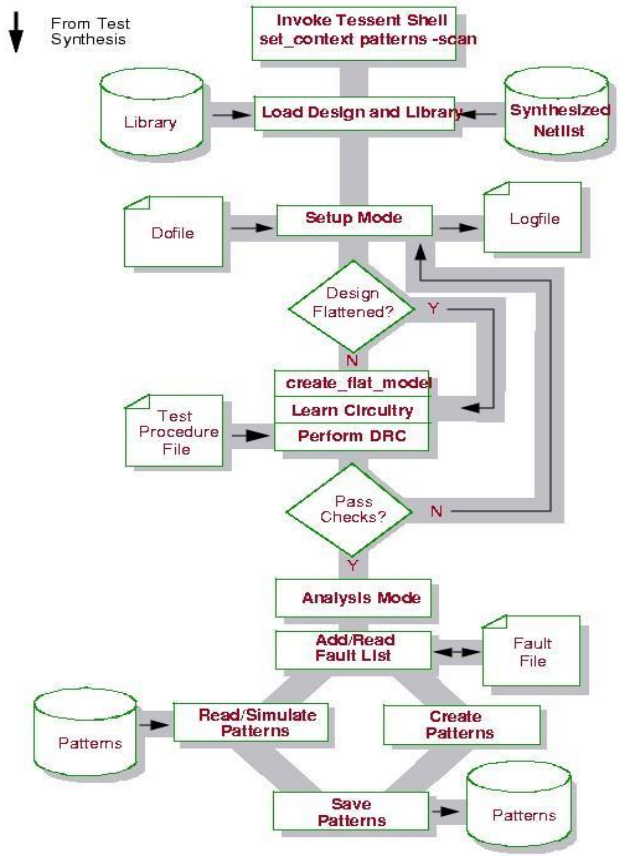


Figure 9 ATPG Tool Usage [6]

The Scan Automatic Pattern Generation and Validation Environment is an ATPG flow developed collaboratively by various teams. It utilizes Tessent software from Mentor Graphics. The process is managed using a main script that handles settings, command-line options, inputs, and outputs, along with lower-level scripts for specific tasks. This structure allows each major function to be contained within a single script, facilitating concurrent development.

Here's a simplified breakdown of the tasks involved in generating test patterns as shown in Figure 9:

**1. Starting Tessent Shell:** Launch the Tessent Shell using the command **tessent -shell**. Set the context to use ATPG functionality by running **set_context patterns -scan**.

**2. Loading Design and Library:** The ATPG tool needs a gate-level netlist and a Design for Testability (DFT) library. Use the commands **read_cell_library** and **read_verilog** to load these files. Every element in the netlist must have a corresponding definition in the DFT library.

**3. Setup Mode:** After loading the netlist and library, the tool enters setup mode. Here, you can configure details about the design and its scan circuits either by interactive commands or by using a script file (dofile). This stage also includes setting parameters that influence model creation during design flattening.

**4. Exiting Setup Mode:** After setting up, exiting the setup mode triggers several operations. If it's the first exit attempt, a flattened design model gets created. This model represents a simplified version of the design and is used in later steps. If the model already exists, perhaps due to a previous session or using the **create_flat_model** command, no new model is created.

**5. Learning Analysis:** Once the model is flattened, the tool performs a learning analysis to understand the design better.

**6. Design Rule Checking:** After learning analysis, the tool checks the design rules to ensure everything is in order.

**7. Analysis Mode:** If the design passes the rule check, it moves into analysis mode, allowing you to simulate the pattern set applicable to the design.

**8. Pattern Creation:** At this stage, you can generate test patterns. Additional setup might be needed, such as specifying a list of faults. ATPG can then be executed on this fault list, and during this process, fault simulation is done to confirm that the patterns can detect the anticipated faults.

This workflow helps in efficiently generating test patterns to ensure that the integrated circuit functions correctly by identifying and addressing potential manufacturing defects.

**ATPG Tool Inputs and Outputs**

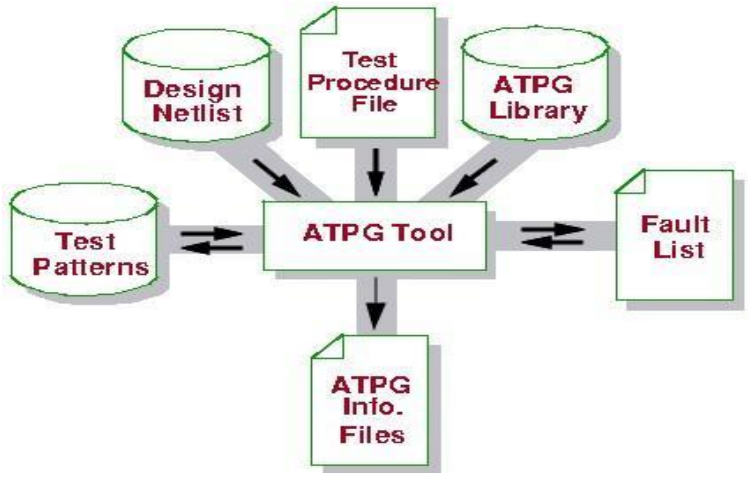


Figure 10 ATPG Tool Inputs and Outputs [6]

As per Figure 9 **ATPG Inputs:**

**1. Design:** The design data must be provided in gate-level Verilog format. Additional inputs include:

- A cell model from the design library.
- A previously saved, flattened model.

**2. Test Procedure File:** This file outlines how the scan circuitry operates in your design. You can either create this file manually or have Tessent Scan automatically generate it using the **write_atpg_setup** command.

**3. Library:** The design library contains details about all the cells used in the design. The tool leverages this library to convert the design data into a simplified, gate-level model that can be used by both the fault simulator and the test generator.

**4. Fault List:** The tool can use an external fault list, which includes information and current states of faults, as a foundation for initiating the test generation process.

**ATPG Outputs:**

**1. Test Patterns:** The tool produces files with test patterns, available in different formats compatible with various simulators and ASIC vendors.

**2. ATPG Information Files:** This collection of files contains data from the ATPG session, such as the creation of a session log.

**3. Fault List:** An ASCII-readable file is generated, detailing internal fault information using the standard Mentor Graphics fault format.

**Results**

To generate test patterns for detecting stuck-at faults, the following command is utilized: **create_patterns**

This command enables the automatic initiation of the ATPG process for stuck-at faults. By default, the tool generates these test patterns internally, meaning they are created within the ATPG system rather than being imported from external files. This approach streamlines the test pattern generation process and ensures integrated handling of design data.

As per Figure 10 it will Performs test generation on selected faults from current fault list.

```
//  ---------------------------------------------------------------------
//  Simulation performed for #gates = 586  #faults = 1664
//  system mode = analysis    pattern source = internal patterns
//  ---------------------------------------------------------------------
//  #patterns  test      #faults  #faults    # eff.    # test         process      RE/AU/AAB
//  simulated  coverage  in list  detected   patterns  patterns       CPU time
//
//  Test generation with capture procedure "OneCycle_CLK"
//   ---        ------      ---       ---      ---       ---          0.25 sec     0/130/0
//   64         88.07%      119      1517       50        50          0.26 sec
//   ---        ------      ---       ---      ---       ---          0.27 sec     0/130/0
//   118        93.68%        2       117       41        91          0.27 sec
//
//  Test generation with capture procedure "TwoCycle_CLK"
//   ---        ------      ---       ---      ---       ---          0.27 sec     0/130/0
//  ---------------------------------------------------------------------
//  Performing redundant fault identification for 132 faults
//  ---------------------------------------------------------------------
//  deterministic ATPG invoked with abort limit = 300
//  # red.    # non-red.   # abort     # remn.   progress   test        process
//  faults      faults      faults      faults              coverage    CPU time
//       0         132          0           0   100.00%    93.68%       0.00 sec
```

Figure 10 Pattern generation report in log

### 4.4 Coverage Analysis

Analyzing the coverage report and addressing low test coverage are crucial elements of ATPG execution. A fundamental step in this process involves understanding the various fault-related terms presented in the statistics reports produced by the ATPG tool. As illustrated in Figure 11, different fault classes are categorized, while Figure 12 provides an example of coverage statistics generated after ATPG's pattern generation phase. By reviewing these figures, users can pinpoint specific faults that need attention to enhance the overall coverage percentage.

```
                    Statistics Report
                     Stuck-at Faults
-------------------------------------------------------------
Fault Classes                                  #faults
                                               (total)
---------------------------------  -------------------------
  FU (full)                           2146
  ---------------------------------  -------------------------
  DS (det_simulation)                1634      (76.14%)
  DI (det_implication)                321      (14.96%)
  UU (unused)                          58      ( 2.70%)
  TI (tied)                             1      ( 0.05%)
  AU (atpg_untestable)                132      ( 6.15%)
-------------------------------------------------------------
Fault Sub-classes
  -------------------------------
  AU (atpg_untestable)
    PC*   (pin_constraints)           100      ( 4.66%)
    SEQ   (sequential_depth)            2      ( 0.09%)
    Unclassified                       30      ( 1.40%)
  *Use "report_statistics -detailed_analysis" for details.
-------------------------------------------------------------
Coverage
  -------------------------------
  test_coverage                                  93.68%
  fault_coverage                                 91.10%
  atpg_effectiveness                            100.00%
-------------------------------------------------------------
#test_patterns                                     91
#simulated_patterns                               118
CPU_time (secs)                                  86.7
-------------------------------------------------------------
```

Figure 11 Coverage Statistics

Untestable faults can be categorized into four subgroups: tied, blocked, redundant, and unused. These types of faults do not lead to functional failures and are therefore excluded from the calculations for test coverage. The most widely recognized metric for measuring coverage is test coverage itself. To enhance test coverage, focusing on faults that are undetected and ATPG-untestable is often most effective.

In figure 12 fault patterns are applied and simulation with expected response is observed, whereas in figure 12 mismatches are there between simulated and expected values at multiple time stamps resulting in simulation failure.

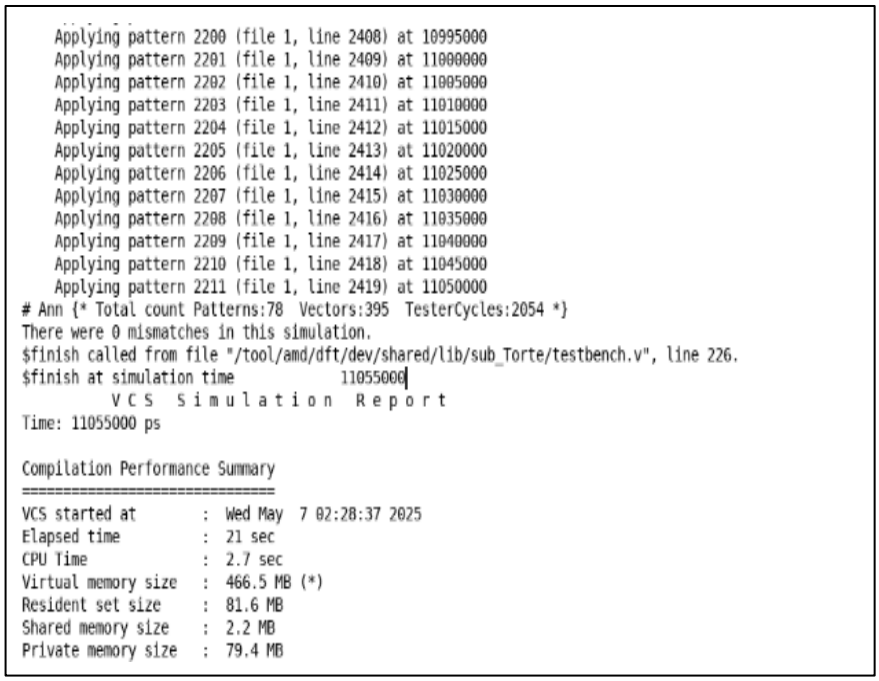


Figure 12 No mismatch in simulation

## 5 Conclusion & Future Scope

In electronic design, Design for Testability (DFT) plays a crucial role in ensuring efficient and effective testing processes. This work explores the foundational concepts, methodologies, and techniques of DFT, emphasizing its importance in developing reliable and robust electronic systems. By adopting DFT principles, engineers and designers can address potential testing challenges, enhance test coverage, and improve overall product quality.

The primary objectives of DFT include simplifying defect detection and diagnosis, reducing test time and costs, and enabling the development of efficient test programs. Techniques such as boundary scan, built-in self-test (BIST), hierarchical DFT, and scan-based methods were examined, highlighting their advantages and limitations. The influence of DFT spans all stages of the design process, from initial design to manufacturing and post-production testing.

Key factors affecting DFT effectiveness include design complexity, area overhead, power consumption, and testability metrics. Implementing DFT involves trade-offs, requiring careful consideration of design constraints and goals. Challenges such as rising design

complexity, the advent of new technologies, and the need for standardized approaches were also discussed.

Looking ahead, DFT will continue to evolve to address the growing complexity of electronic systems. Potential advancements include:

**1. Automation:** Leveraging artificial intelligence (AI) and machine learning (ML) can transform DFT by automating tasks like test generation, fault diagnosis, and optimization. AI-based algorithms can analyze vast amounts of design and test data to detect patterns, predict potential failures, and streamline testing processes.

**2. Testability-Driven Design:** Incorporating testability early in the design process can create inherently testable structures, reducing the need for complex DFT techniques and improving overall testability.

**3. System-Level Testability:** As electronic systems become increasingly interconnected, DFT techniques should extend beyond individual components to include entire systems, encompassing interconnects, interfaces, and communication protocols.

**4. Standardization and Collaboration:** Collaboration among designers, test engineers, and manufacturers can promote standardization, improving the integration of DFT techniques within design and testing environments.

**5. Integration with Design for Manufacturability (DFM):** Aligning DFT with DFM principles ensures a smooth transition from design to production. By considering manufacturing constraints and testing requirements simultaneously, designers can optimize both manufacturability and testability.

In summary, DFT remains a cornerstone of electronic design, facilitating effective testing processes. By embracing technological advancements, focusing on early-stage testability, expanding to system-level testing, standardizing practices, and integrating DFT with DFM, designers can address future challenges and drive innovation in testability for electronic systems.

Future research in the field of Design for Testability (DFT) focuses on enhancing automation and collaboration. The integration of artificial intelligence (AI) and machine learning (ML) has the potential to transform DFT by automating key processes such as test generation, fault diagnosis, and optimization. AI-driven algorithms can analyze large volumes of design and test data to uncover patterns, predict potential failures, and streamline testing procedures.

Additionally, fostering collaboration between designers and test engineers will play a pivotal role in advancing DFT practices. Establishing standard interfaces and protocols can simplify the integration of DFT into the design and testing workflows, enabling a more cohesive and efficient process. This collaborative approach is expected to enhance the adoption of DFT strategies, paving the way for more reliable and testable electronic systems.

**References**

1. "Scan and ATPG Process Guide", Mentor Graphics, 2006.
2. Lo, H. H., Lee, W. F., Reaz, M. B. I., Hisham, N., & Shakaff, A. Y. M. (2008). *Design methodology to achieve good testability of VLSI chips: An industrial perspective*. In Proceedings of the International Conference on Electronic Design (ICED) IEEE.
3. Wang, L. T., Wu, C. W., & Wen, X. *VLSI Test Principles and Architectures.* Elsevier.
4. *VCS Functional Verification User Guide, Version X2005.06* Synopsys, 2005.
5. Chauhan, J., Panchal, C., and Suthar, H., 2017. "*Scan methodology and atpg dft techniques at lower technology node*". In 2017 International Conference on Computing Methodologies and Communication (ICCMC).
6. *Tessent Scan and ATPG User's Manual*, Mentor Graphics, Version 2016.3.
7. *Design-for-Test: Scan and ATPG Training Student Workbook,* Mentor Graphics. AMD Internal Documents.
8. Harishchandra Patel, "Impedance Control in HDI and Substrate-Like PCBs for AI Hardware Applications" (2024). Journal of Electrical Systems, 20(11s), 5109-5115.
9. F.-H. Tang, H. -Y. Kao, S. -H. Huang and J. -F. Li, "3D Test Wrapper Chain Optimization with I/O Cells Binding Considered," *2019 International 3D Systems Integration Conference (3DIC)*, Sendai, Japan, 2019, pp. 1-4, doi: 10.1109/3DIC48104.2019.9058794.