EDA Solution for Design For Test

Navdeep Sood

cadence®

Introduction

Navdeep is a highly accomplished professional presently managing DSG DFT team for the Global Customer Success team at Cadence, Noida. With 14 years of extensive experience in the field, he possesses a comprehensive understanding of various aspects of Design-for-Test (DFT) and Test methodologies. His academic background includes an MTech degree from Punjab Engineering College. His research interests focus on cutting-edge areas such as Optimized Test Point using AI &ML, 3D IC, Physical aware DFT, and Hierarchical ATPG. He is actively involved in pushing the boundaries of semiconductor technology and addressing the challenges posed by advanced chip designs. As evidence of his expertise, he has published three papers in renowned IEEE conferences, including the prestigious International Test Conference (ITC-Anaheim). He has successfully trained hundreds of DFT Engineers across various companies on the efficient utilization of Cadence DFT and ATPG EDA Tools.

AICTE | arm Education | life.augmented

Supported by

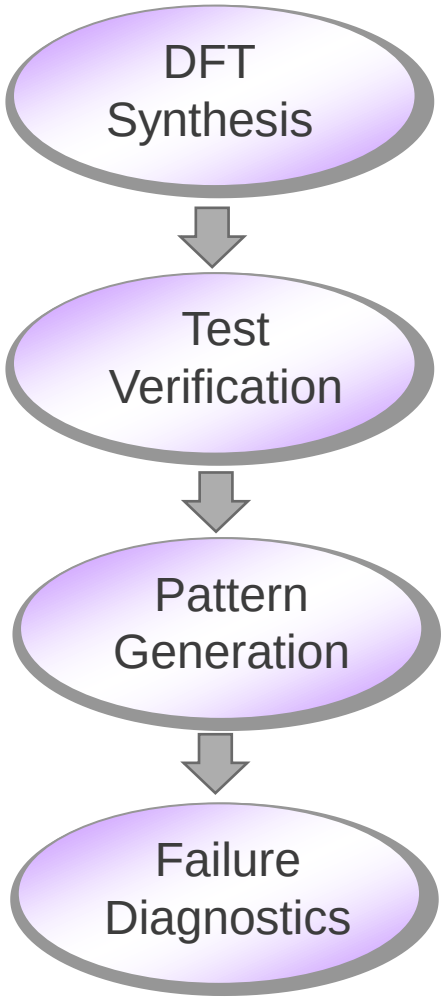
cadence

cadence

Agenda

- Fundamentals of the DFT and ATPG Flow
  - Overview of Synthesis Flow
  - Processing steps to run ATPG Flow

DFT Software Solutions



- Correct-by-design test structures using Synthesis tool.
  - Scan
  - Lbist
  - Mbist
  - Bscan
- Verify compliance with test design rules.
- Full range of tests for all types of digital testing
- Diagnostics for fault isolation, failure analysis, and process monitoring

Scan Insertion and ATPG: Inputs and Outputs

Synthesis Tool

Inputs
- Synthesis Libraries
- SDC's
- RTL
- DFT Constraints
- CTL's

Synthesis with DFT

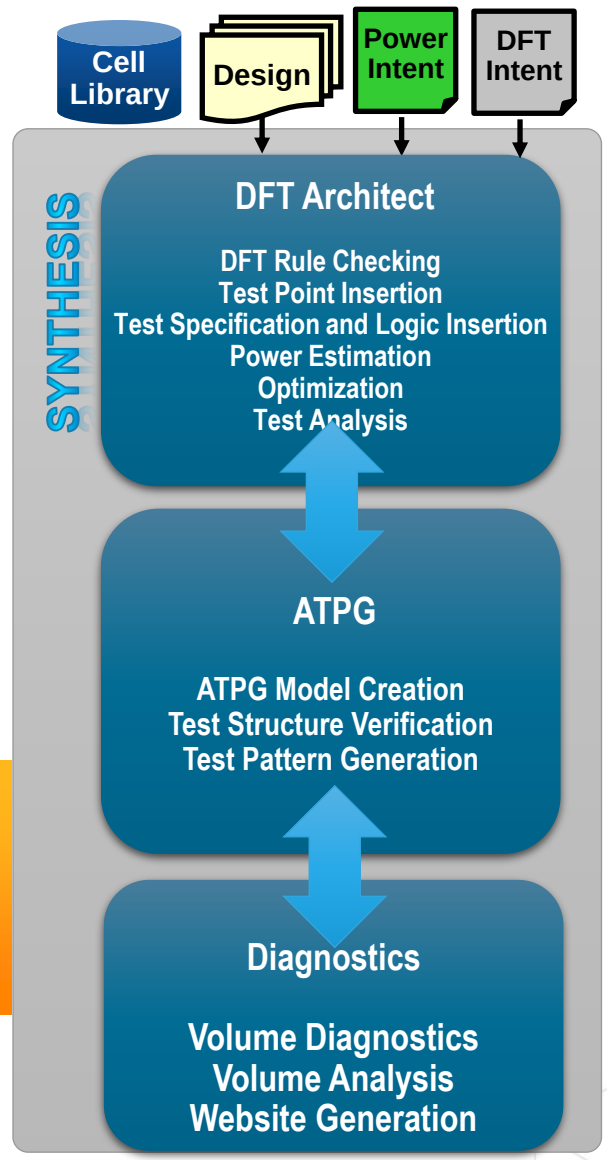
Outputs
- SDC's
- DFT Inserted Netlist
- ATPG Setup
- Scandef
- Scan Abstract Model/CTL

Inputs
- Netlist
- Test Constraints
- Cell Libraries

ATPG

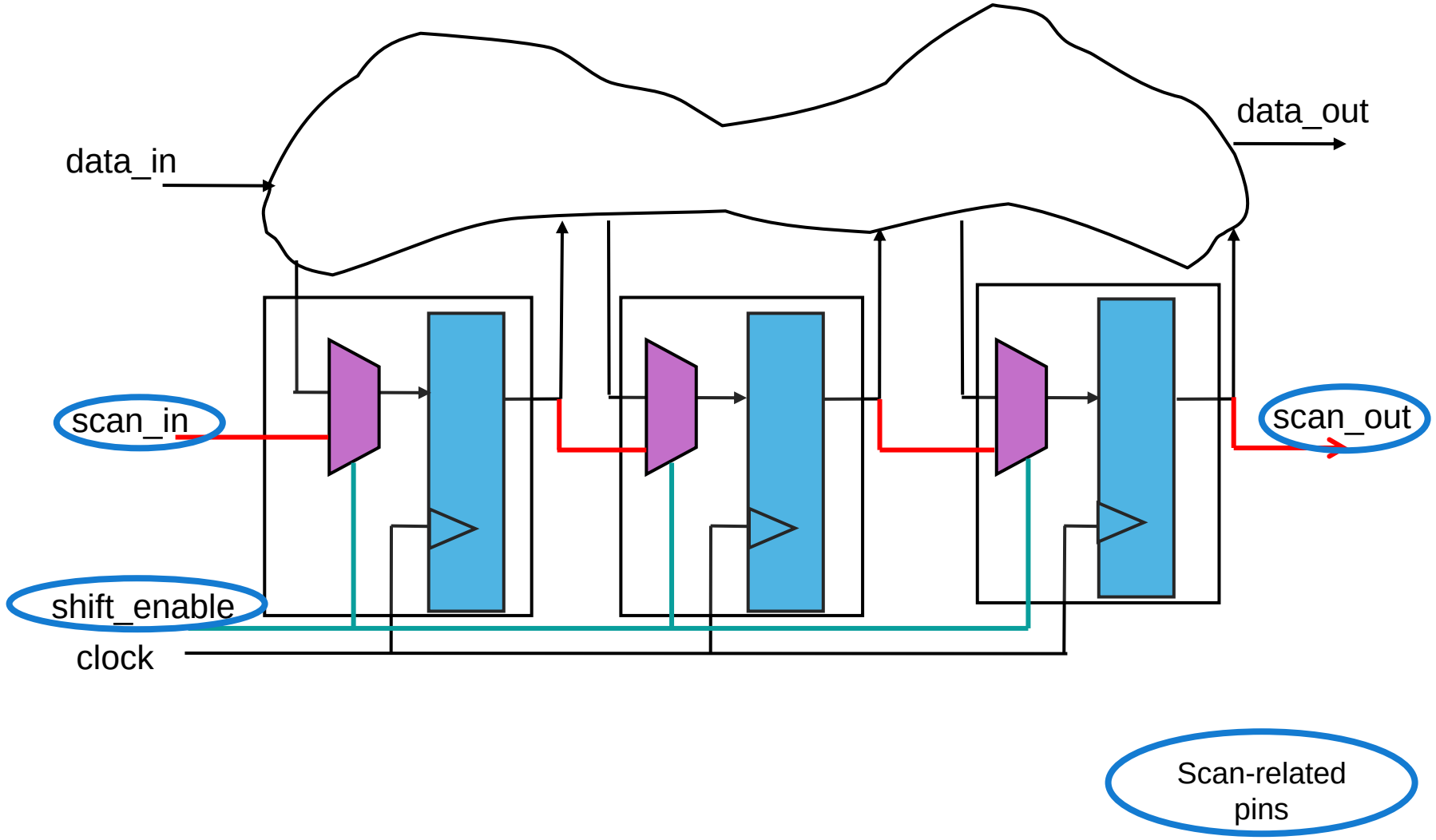
Outputs
- STIL Patterns
- Verilog Patterns

ATPG Tool



Synthesis

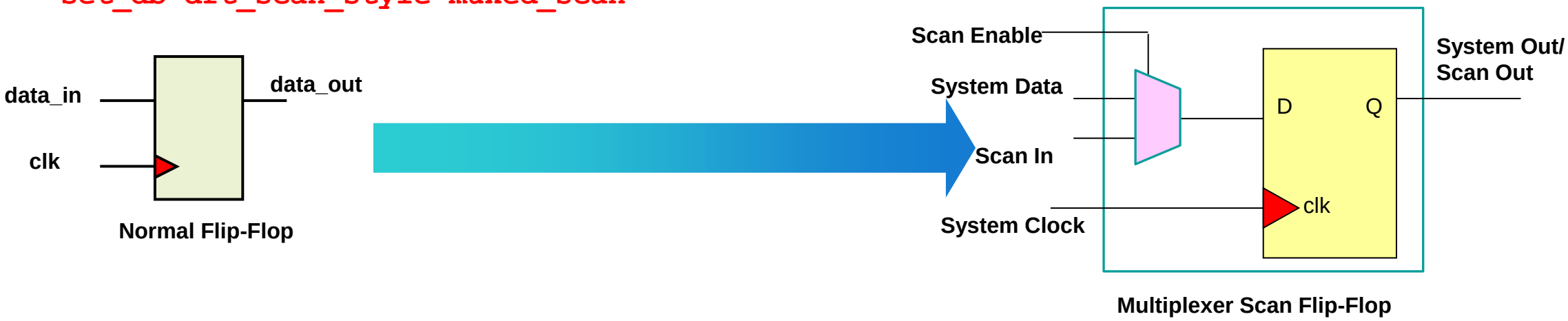
A Design with a Test Circuit



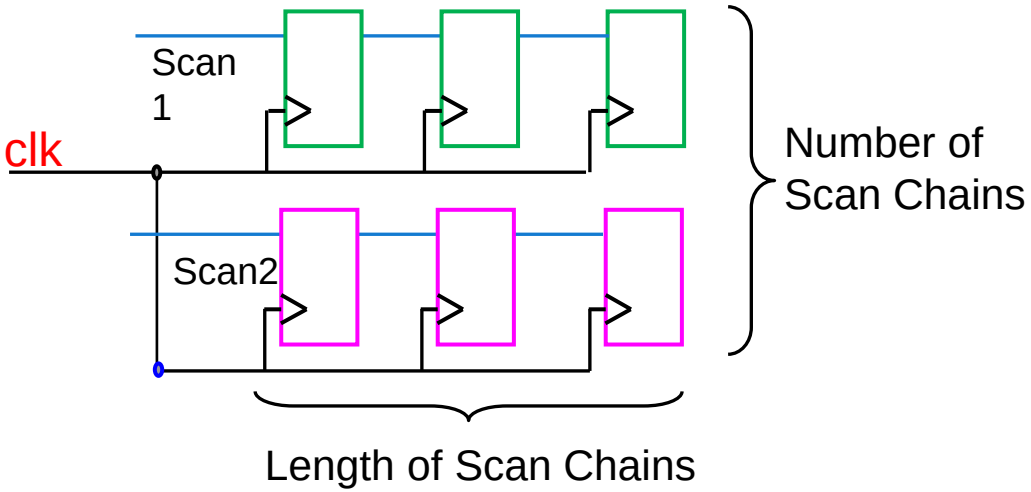
Scan Styles: MUX Scan

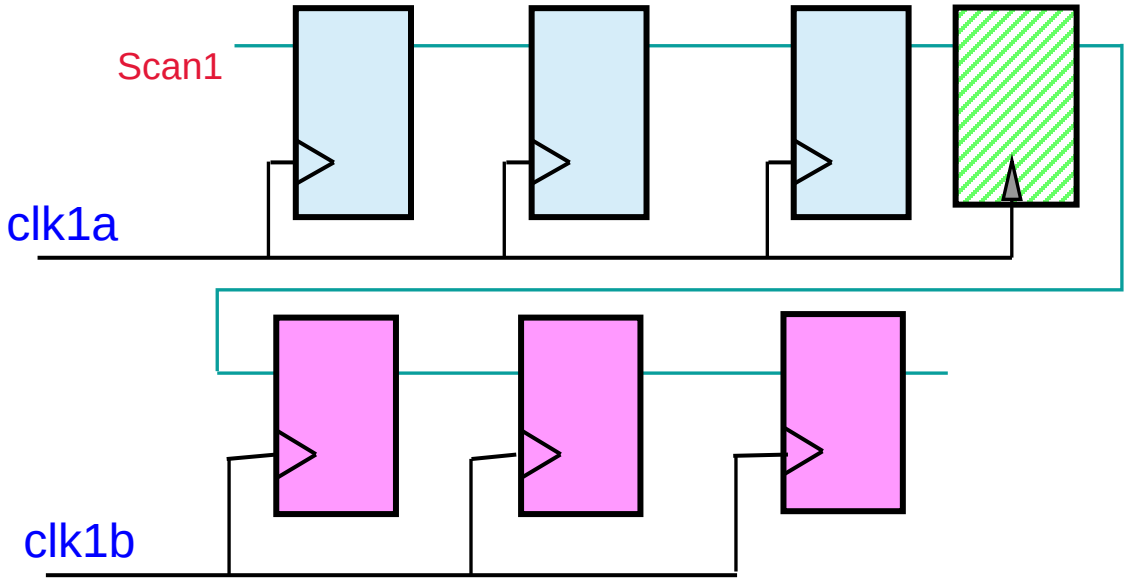
- The MUXed scan style (*muxed_scan*) is the most commonly used scan style
- In this style, an edge-triggered D-flop is replaced with its multiplexed D-flop (scan) equivalent cell
- The Scan Enable signal selects either the system or the scan data

```
set_db dft_scan_style muxed_scan
```

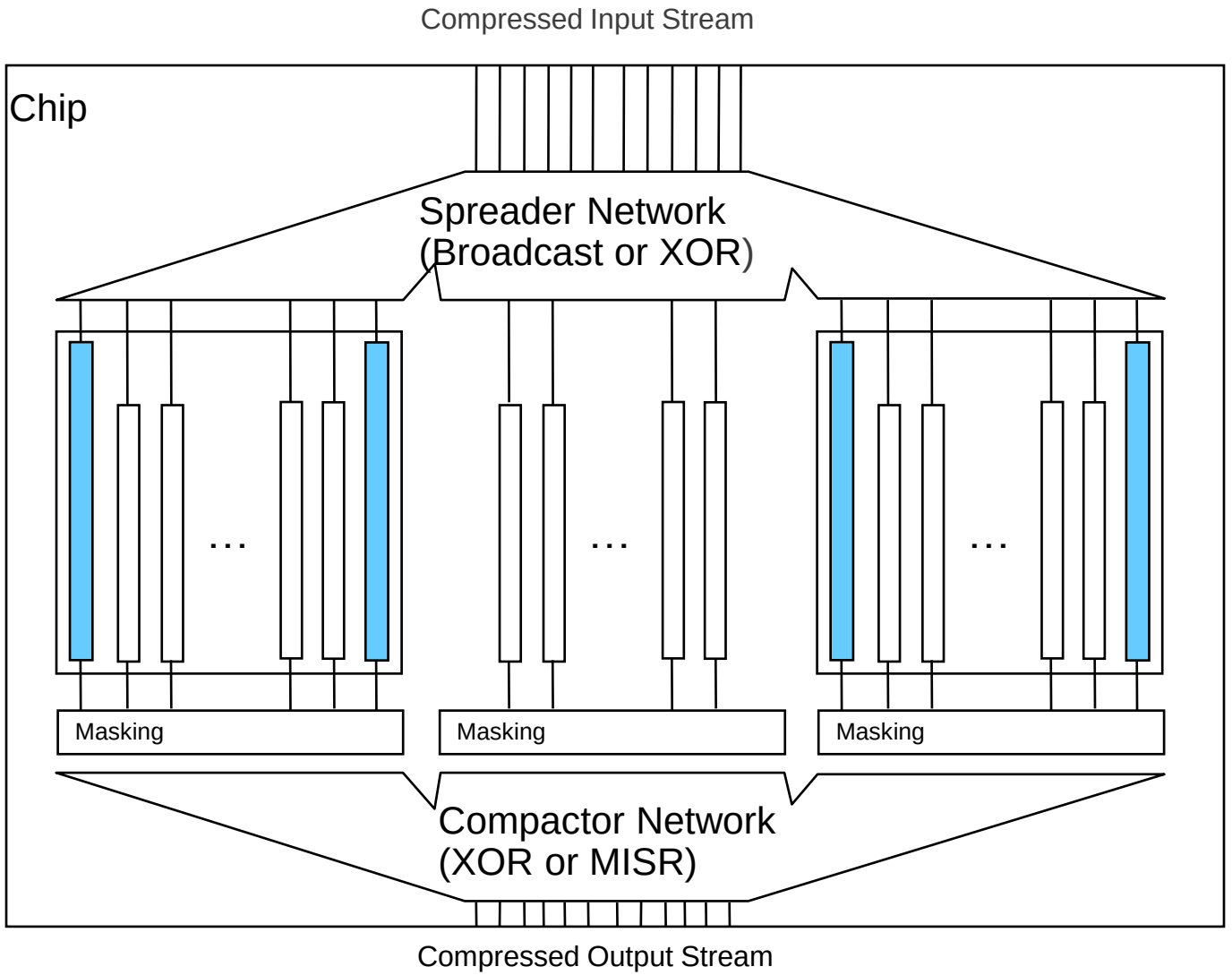


Scan Chains





Compression Structures



Input Side

- Broadcast
- XOR Spreader

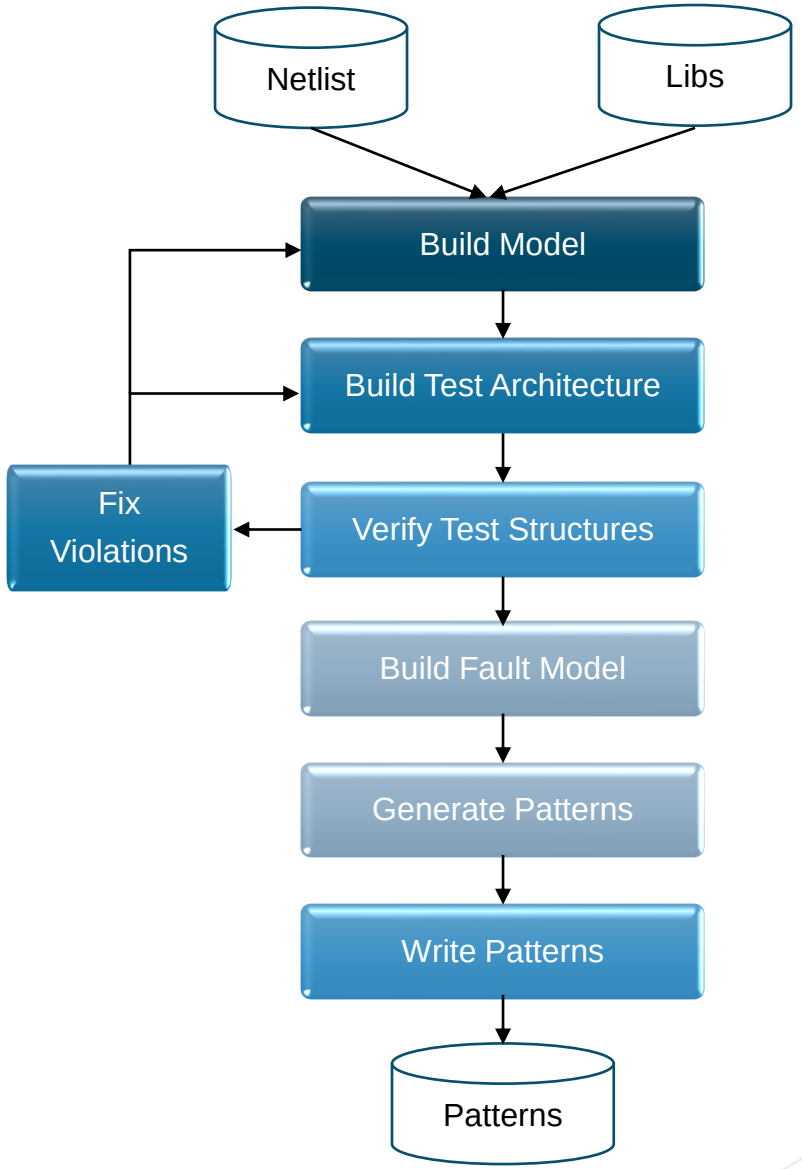
Output Side

- XOR compactor
- MISR compactor
- X-state masking

ATPG

ATPG Flow

- Building a Test Model
- Building a Test Structure
- Verifying Test Structures
- Building a Fault Model
- Generate ATPG patterns
- Writing Patterns

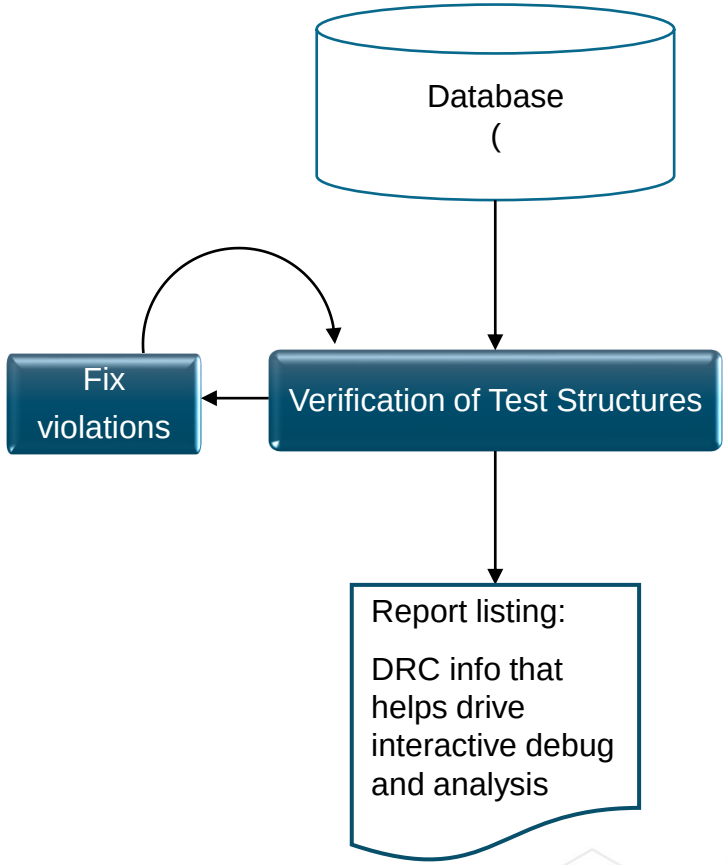


What is a Test Architecture?

- A test architecture defines a specific configuration of a design consisting of:
  - How the test structures are accessed and how clocking is controlled
  - Scan chains, clocks, and control pins
  - Initialization, scan, and test sequences (can be customized)
  - Fixed-Value registers, inactive logic, etc.
- Any change in a test configuration requires the definition of a new test mode
- Different types of test modes include:
  - FULLSCAN, the traditional scan test with scan-in and scan-out pairs
  - COMPRESSION are the Modus compression test modes
  - 1149 is a scan mode using the IEEE 1149.1 Standard scan protocol
    - Used for our MBIST, LBIST, and/or Boundary Scan in chip manufacturing
    - Often used as a parent mode to set up other test modes

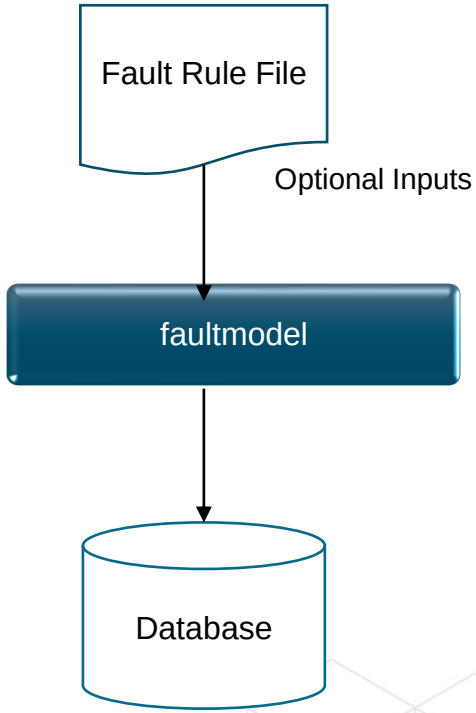
Verification of Test Structures

- This step is to analyzes scan-based design rules for proper shifting and capturing
  - Identify scan elements and verify their controllability and observability
  - Identify logic structures that may cause manufacturing test problems
  - Identify logic structures that cause test generation problems
- Some examples:
  - Clock races
  - Tri-state contention
  - Broken scanchains
  - X-state propagation

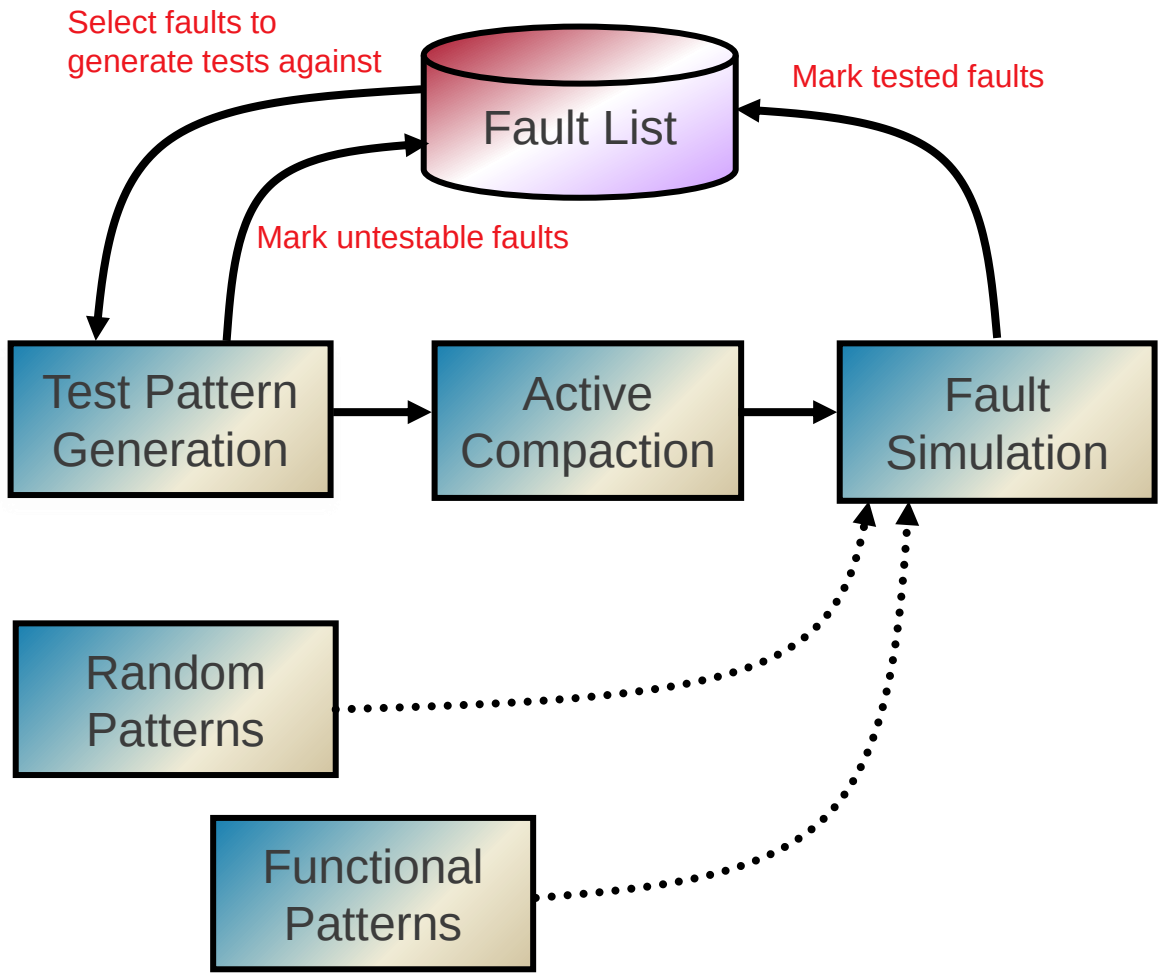


Build Fault Model Overview

- This step creates a fault database for ATPG processing.
- Some examples of fault types: static, dynamic(transition), iddq, pattern, pin, path, gate-exhaustive
- Prepares a fault list for Modus ATPG fault processing
  - Stuck-at (static) pin faults (stuck-at 1, stuck-at 0)
  - Transition (dynamic) pin faults (slow to rise, slow to fall)
  - Pattern faults (static and dynamic)
  - Path delay faults
- Optimizes the fault list
  - Faults are collapsed
    - Faults that have the same effect on the circuit
  - Faults are grouped for more efficient test generation
    - For example, Register Arrays where there is known regularity in the logic structure
- Prepares
  - Repository of all fault status information
  - Mechanism to "keep score" (multi-mode fault coverages)
  - Permanent pattern set

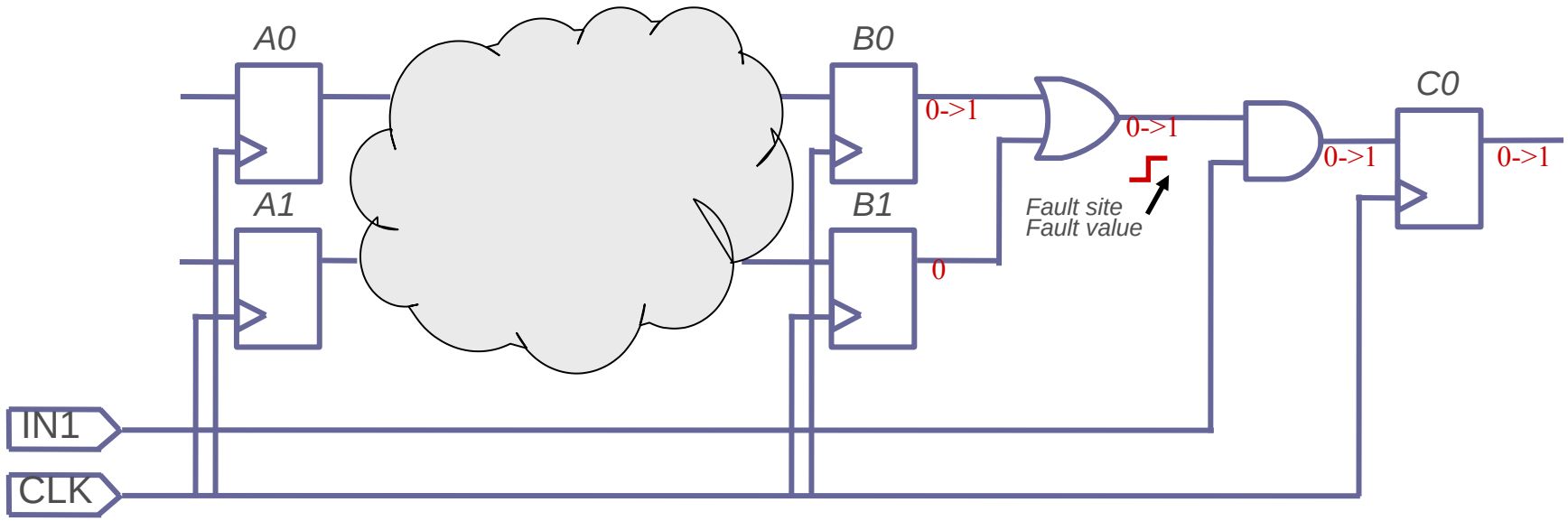


Generation of Patterns: ATPG Flow



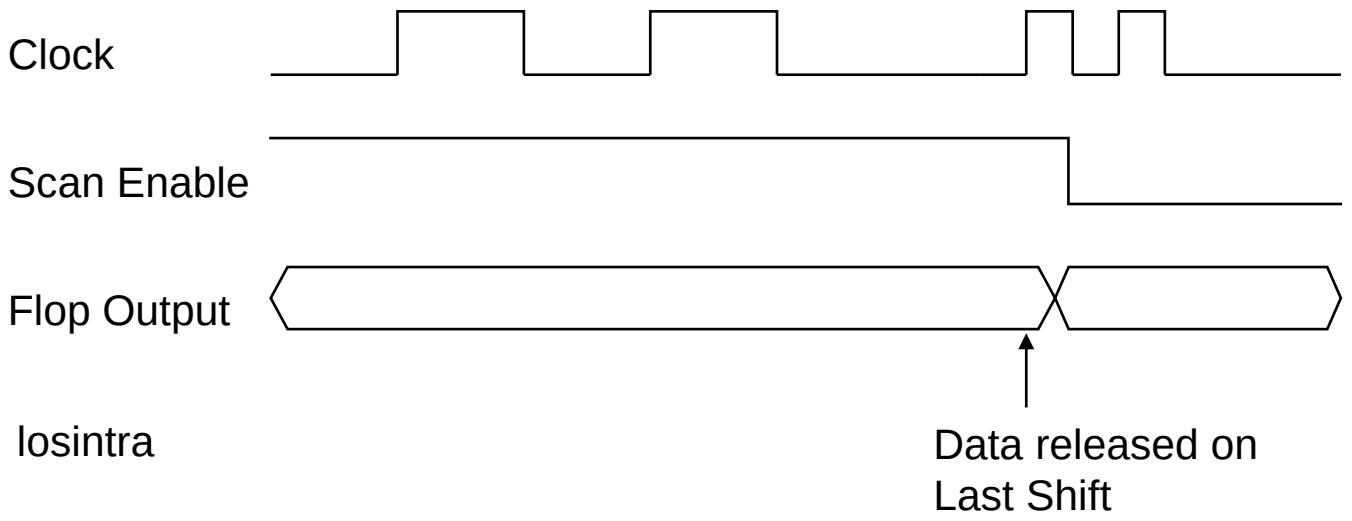
Dynamic Fault Testing

- Delay test pattern generation
  - Generate a transition through the fault site in two time frames
    - First time frame: generate transition
    - Second time frame: capture results
- More difficult and processing-intensive than stuck-at test

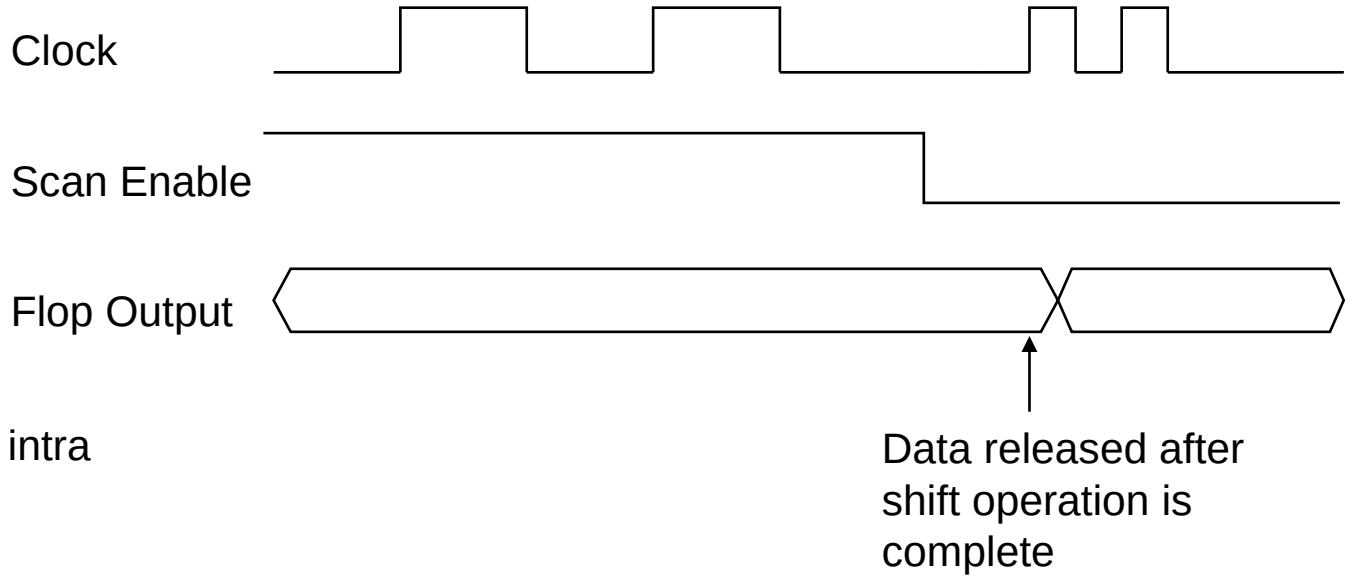


Creating a Transition

- Launch on Last Shift



- Launch on Capture / Functional Release



Test Coverage Calculations

- Standard calculations:
  - $Test\ Coverage = \frac{Number\ of\ detected\ Faults}{Total\ \ Faults}$
  - $Adjusted\ Test\ Coverage = \frac{Number\ of\ detected\ Faults}{Total\ \ Faults - Redundant\ faults}$

Write Patterns

- EDA tools writes out patterns in different formats which can be consumed either by tester or for simulation
- Formats: Verilog, WGL, STIL, TDL