

# Improving At-Speed DFT Coverage Using Early RTL Testability Analysis

By Chris Browy and Kai-hui Chang

## Introduction

Delay testing is widely used to check whether a manufactured chip is free of delay defects and meets its performance specification. Common delay testing methods such as logic BIST and at-speed scan testing are applied to industrial designs with good results. However, as process nodes shrink, delay defects take on a more subtle forms involving once secondary effects such as interconnect, crosstalk, process variations, and power-supply noise.

Timing-aware ATPG and logic BIST tools are addressing the need to use more sophisticated delay fault models and robust test constraints; however, engineers are burdened with significantly longer runtimes and more complex mitigation steps to raise test coverage to acceptable levels compared to previous practices.

This paper will review the basics of at-speed testing and introduce new automatic methods of analyzing and improving at-speed scan-based design for testability through early RTL analysis and design.

## Brief Overview of At-Speed Test

There are three main methods for at-speed testing

- At-speed scan test
- Faster-than-rated clock scan test
- Logic BIST

All approaches support a variety of delay fault models accurate enough to test for delay defects. Our focus is on at-speed scan-based testing.

## *Types of Test Cycles*

At-speed scan testing utilizes one of several scan test cycle approaches:

**Launch on Shift (LOS):** In LOS, the transition is launched in the last shift cycle during the scan shift operation. This activates the required transition at the target node or path which is then propagated and captured through the functional path at the observable point (Din) of any flip flop our primary output port of the chip.

**Launch off Capture (LOC):** In LOC, the transition is launched and captured through the function pin (Din) of any flop-flop in the scan chain, see Figure 1. Since the launch pattern depends on the functional response of the loaded scan pattern, the launch path is less controllable and test coverage may be more difficult to achieve. However because LOS puts more difficult timing constraints on the Scan Enable signal, LOC is more widely deployed and runs on low cost testers.

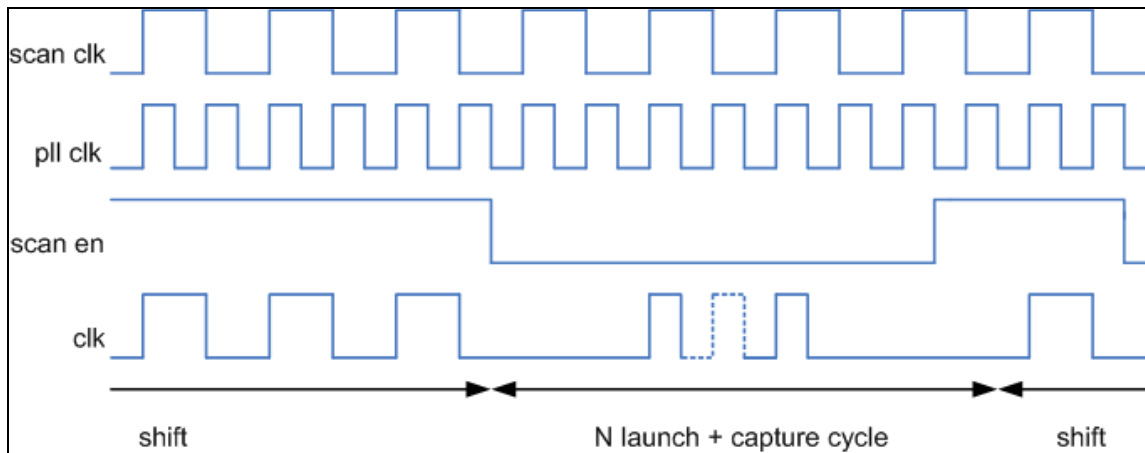


Figure 1 Launch-off Capture (LOC) test cycle timing diagram

### Test Quality

The quality of a set of tests can be classified by the concept of robustness. A test is called *robust* if, and only if, it detects the fault independent of other delay faults in the circuit. *Non-robust* tests guarantee the detection of a fault if there are no other delay faults in the circuit. If there is neither a non-robust nor a robust test, the delay fault is untestable. Robust and non-robust tests differ in the constraints on the off-path inputs of the path, notably they both must be non-controlling, however in robust test they have to be stable values throughout the test cycle. In Figure 2, the on-path transition is launched at q3 while the off-path values of q1 and q2 must be 0 throughout the test cycle. In ATPG, robust test refers to the robust fill method, while non-robust test mode refers to the random fill method.

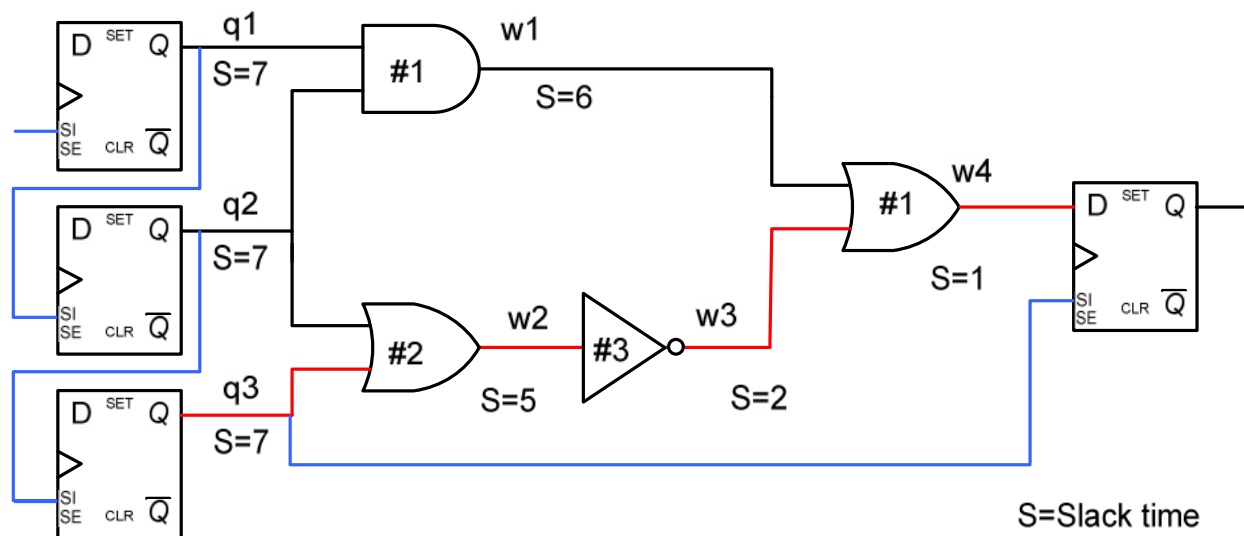


Figure 2. Path delay fault launch->capture path (q3->w4)

### At-Speed Delay Fault Models

There are several delay fault models used in at-speed ATPG and Logic BIST

- Path Delay Fault (PDF)
- Transition Delay Fault (TDF)
- Small Delay Defect Transition Delay Fault (SDD-TDF)

*Path-delay faults* (PDF) model distributed delay defects. When the cumulative delay along a targeted path exceeds the clock cycle and setup time, a path delay fault occurs. The effect of the delay may be different for a rising or falling transition so two PDFs (rising and falling transitions) are required for each physical path.

*Transition delay faults* (TDF) model a spot delay defect on a single node that is assumed to be large enough to exceed the timing slack along any propagation path to the output or flip flop. Thus, gross local faults and faults distributed across a large area are covered by this fault model. Due to differences in slow-to-rise and slow-to-fall timing, 2 transitions are necessary per node. Assuming  $m$  signals in the circuit,  $2m$  Transition Delay Faults (TDFs) can be modeled.

*Small delay defect transition delay faults* (SDD-TDF) model the small delay variations induced by crosstalk, process variations, power-supply noise, as well as resistive opens and shorts can potentially cause timing failures in a design, thereby leading to quality and reliability concerns. A SDD is a defect with defect size not large enough to cause a timing failure on its own. An SDD might escape during test application when a short path is sensitized since the accumulated delay of the distributed delay defect is not large enough to cause a timing violation. In contrast, the same SDD might be detected if a long path is sensitized. In Figure 3 the SDD-TDF at  $w4$  should be tested along the path from  $q3 \rightarrow w2 \rightarrow w3 \rightarrow w4$  since the measured slack time is the least. Unfortunately, common ATPG algorithms usually prefer short paths since the sensitization of these paths is typically easier.

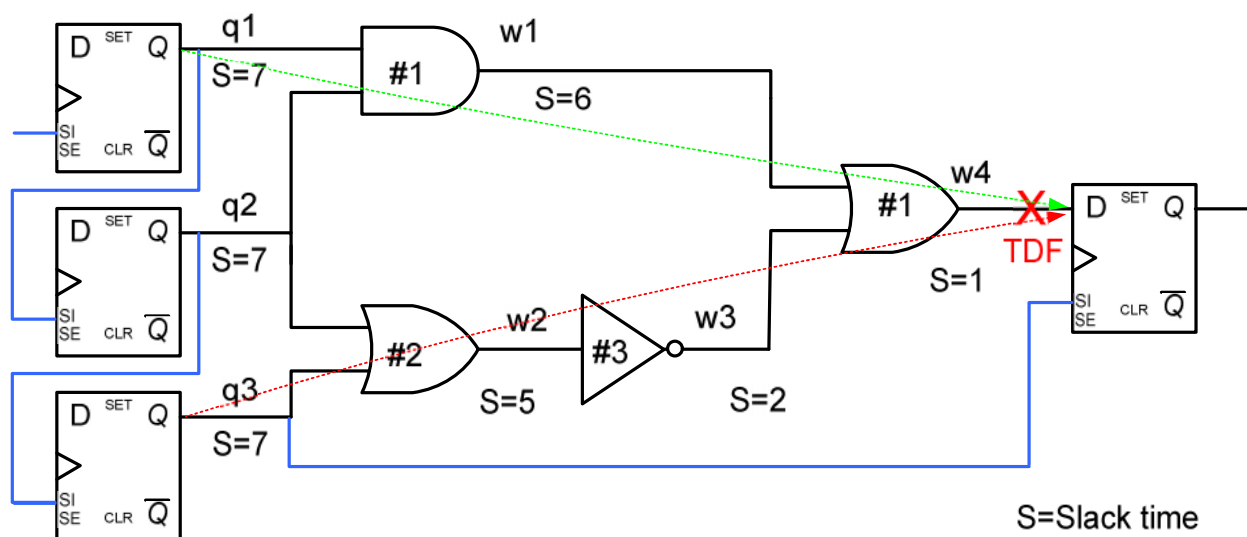
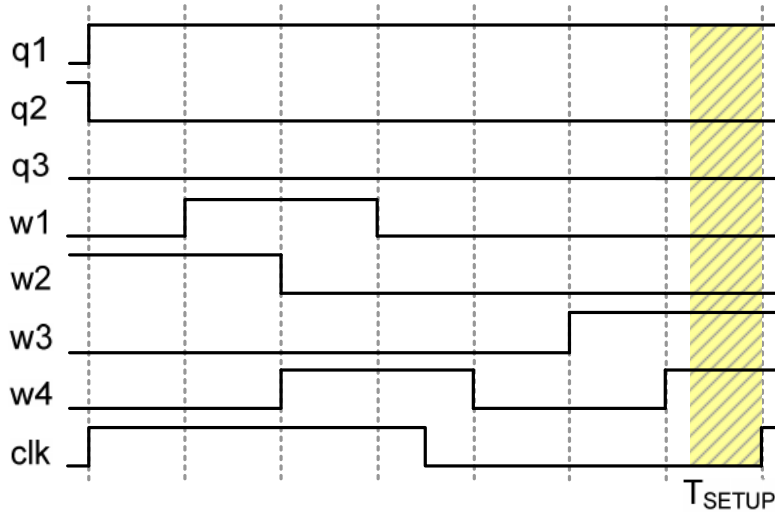


Figure 3. Transition delay fault long path and short paths



**Figure 4. Critical timing path waveform**

Timing-aware ATPG is needed for PDF and SDD-TDF testing including the robust test method. Here pre-calculated timing information is used during structural ATPG to guarantee sensitization of the longest path and will be more likely to detect SDDs. However, timing-aware ATPG is a computationally intensive task, since the search space is very large. As a result, the run time of timing-aware ATPG increases significantly compared to regular ATPG.

### Insight DFT Overview

Insight DFT provides an at-speed DFT testability analysis solution that considers DFT starting at the RT-level and can lead to higher ATPG test coverage in less time. The overall solution, as shown in Figure 5, is comprised of:

- Automatic DFT testbench generation
- DFT design rule checks
- Robustness checks
- PDF testability coverage
- Automatic RTL repair

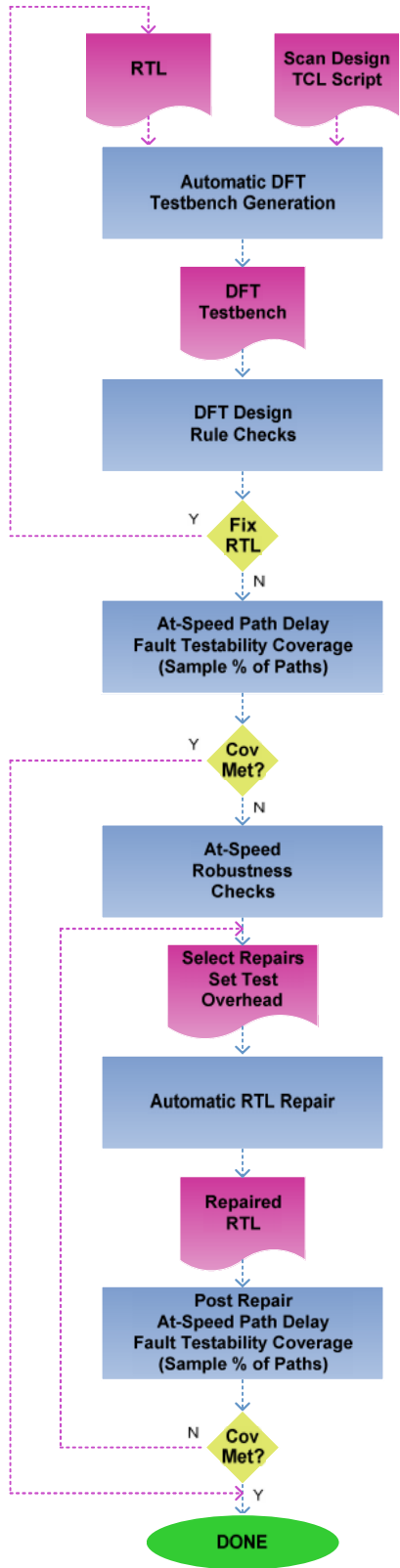


Figure 5. Insight RT-level at-speed DFT flow

## Scan Design Intent

Performing accurate testability analysis on RTL requires knowledge of how the design will operate in at-speed test mode. Insight supports Tcl commands to express the full scan design intent including defining scannable and non-scannable registers, X-generators, test mode clock schemes, IO pin modes, memory bypass modes, external clock sources and test pins, and false and multicycle paths. The following is a typical Insight DFT setup script.

```
# Set DUT (design under test)
insight_dft_set_dut dut1

# Set inputs: get all inputs and then remove clk, clk2 and reset (remove those
# that are explicitly controlled
set inputs [insight_get_inputs]
set dft_inputs [set_difference $inputs {clk reset clk2}]
insight_dft_set_input $dft_inputs

# Generate lists of different scan, non-scan, and X generators
set all_regs [insight_get_registers]
set noscan_regs {dut1.b dut1.f}
set x_gen {dut1.x}
set exclock {clk2}

# Generate scannable register list by removing non-scannable registers and
# x-generators from the all-register list
set scan_regs [set_difference $all_regs $noscan_regs]
set scan_regs [set_difference $scan_regs $x_gen]
# Set non-scannable registers
insight_dft_set_noscan $noscan_regs
# Set scannable registers
insight_dft_set_scan $scan_regs

# Set X generators
insight_dft_set_xgenerator $x_gen

# Set external clocks
insight_dft_set_exclock $exclock

# Set test clock signal
insight_dft_set_clock clk 10 posedge

# Set values for constant inputs like reset, scan enable, etc.
insight_dft_set_input_value reset 0
# DFT Options, please refer to user's guide for more details
# Use path-delay source trigger robust-testing model
insight_dft_option 3 'b1000
# Set SAT time-out to 120 seconds
insight_dft_option 6 120

# Set multi-cycle analysis. Here no multi-cycle is used
insight_dft_set_extra_cycle_analysis_no 0

# If additional environment setup is required, uncomment the following two lines
# to include additional Verilog code and to set DFT start delay
#
# insight_add_include environment_setup.h
# insight_dft_start_delay #100

# Generate testbench for DFT path testability analysis
insight_dft_tbench_gen dut1_tb.v
```

## DFT Design Rule Checks

Insight performs a series of DFT rule checking to find obvious testability issues associated with operating in test mode. Repairing the design for these situations will improve overall test coverage. These checks focus on mixing clock and data signals and X generators.

```
-----
DFT Rule Check Summary Report
-----
Rule                                code    #
-----
data driven clock                    DC      0
clock contains X                     XC      1
data contains clock                  QC      0
data contains X                      XD      1
-----
```

**Listing 2. DFT rule checks for clock and data**

## At-Speed PDF Testability Analysis

Insight performs SDD-TDF and PDF test coverage estimation based on targeting the high fanout critical paths in the design. Assessing what paths are most critical in the RTL is based on combinational logic complexity heuristics. Because the PDF model generates an exponentially large number of register to register paths to analyze, Insight uses a sampling approach of the highest fanout paths to analyze for at-speed testability. Insight will generate a PDF summary report that identifies which paths are detected robustly along with the number of paths that are not detected for any number of testability issues. Additionally a detailed untestable path report ranked by combination logic complexity is also generated for detailed investigation.

```
-----
Path Delay Fault Summary Report
-----
fault class                          code    #paths
-----
detected robustly                    DR      463
not detected                          ND     6943
  X generator                          XG      0
  non-scannable register                UC      0
  unobservable capture register         OB      0
  cross clock domain unblocked          CD      0
  launch register non-scannable         NS      0
  false path unblocked                  FP      0
  multicycle path unblocked             MP      0
  non-controllable                      NC     6943
timeout                                TO      0
-----
sampled path analyzed                  7406
total flops                            186
total scan flops                        186
total non-scan flops                    0
test coverage                           6.25%
-----
```

**Listing 3. PDF summary report**

```

=== Begin of At-Speed Path Report (Untestable) ===
dft_testbench.dut.opcode[1] =>
  dft_testbench.dut.wdatahold2[2]
  (AIG nodes = 243): 01 (NC), 10 (NC)
dft_testbench.dut.opcode[0] =>
  dft_testbench.dut.wdatahold2[2]
  (AIG nodes = 123): 01 (NC), 10 (NC)
dft_testbench.dut.opcode[7] =>
  dft_testbench.dut.wdatahold2[2]
  (AIG nodes = 43): 01 (NC), 10 (NC)
=== End of At-Speed Path Report ===

```

**Listing 4. Untestable PDF path report**

***At-Speed Robustness Checks***

SDD-TDF and PDF require using the robust fill ATPG test generation algorithm to ensure these delay fault types can be detected independently of other delay faults in the circuit. This mode of ATPG is very time consuming because of the additional constraints compared to random fill ATPG. Insight is able to identify the most interesting circuit paths in the design and analyze them for robustness, specifically that certain flops in the logic cone of the path under test can remain stable throughout the at-speed test cycle. If the design’s logic cannot be controlled deterministically in test mode then ATPG test coverage will suffer. Insight will identify the high fanout paths in the design and analyze them for adherence to robustness constraints. Once identified these paths can be repaired and result in higher SDD-TDF and PDF test coverage.

```

-----
Robustness Check Summary Report
-----
Uncontrollable Summary          #
-----
total source flops analyzed      186
uncontrollable source flops     154
modules to repair                1
sampled paths analyzed          3703
total paths predicted to fix     1211
-----
Fixed path fanout histogram
#flops                          #paths
-----
1                                93
2                                19
3                                978
4                                 2
5                                 0
<=10                             8
<=100                             111
>100                              0
-----

```

**Listing 5. Robustness check report**

## **Automatic RTL Repair**

Fixing testability issues in the RTL is tedious and error prone. Insight can automatically generate RTL repairs that will result in higher testability under test overhead constraints set by the designer. The top at-speed robustness checks can be targeted for repair and evaluated for the estimated number of paths fixed and additional logic overhead. Insight optimizes the test logic overhead by reusing existing scannable flops to add the new logic paths to ensure improved controllability.

Here is the result of running the `dft_rtl_fix.pl` repair script. If the test overhead is too high the repair list can be reduced until it is within acceptable bounds. The estimated number of paths fixed for a repair solution is also provided. Note repairs already take into account register reuse.

```
Number of registers inserted: 4
Number of paths repaired: 9
Fixing dut1.counter, file dut1.v
Module dut1 found
* Fixed successfully
```

**Listing 6. Automatic RTL robustness fix summary**

Here is an example of how the repairs are implemented. The off-path registers that are repaired to have a new logical path muxed on their output which is controlled by a scannable control register. This repair is enough to provide the robust controllability that is required.

```
reg avery_val_0_state;
reg avery_val_1_state;
reg avery_val_2_state;
assign state[0] = avery_ctrl_state[0] ?
    avery_val_0_state :
    avery_orig_state[0] ;
assign state[1] = avery_ctrl_state[1] ?
    avery_val_1_state :
    avery_orig_state[1] ;
assign state[2] = avery_ctrl_state[2] ?
    avery_val_2_state :
    avery_orig_state[2] ;
```

**Listing 7. Automatic RTL robustness fix**

## **Post Repair At-Speed PDF Testability Analysis**

After RTL repairs are made, Insight performs another SDD-TDF and PDF test coverage estimation based on targeting the same high fanout critical paths in the design as the initial run. This run confirms the increased number of paths that are detected robustly as a result of the repairs.

```

-----
Path Delay Fault Summary Report
-----
fault class                code    #paths
-----
detected robustly         DR      3378
not detected              ND      4028
  X generator              XG       0
  non-scannable register  UC       0
  unobservable capture register OB       0
  cross clock domain unblocked CD       0
  launch register non-scannable NS       0
  false path              FP       0
  non-controllable       NC      4028
timeout                  TO       0
-----
sampled path analyzed          7406
total flops                   186
total scan flops              186
total non-scan flops          0
test coverage                  45.61%
-----

```

**Listing 8. Post-repair PDF testability coverage**

### ***At-Speed ATPG***

After the RT-level DFT analysis and repairs have been completed, the design is ready to go through logic and test synthesis and ATPG.

### **Conclusion**

Timing-aware ATPG are necessary to address the need to use more sophisticated delay fault models such as SDD-TDF and PDF. However, engineers are burdened with significantly longer runtimes and more complex mitigation steps to raise test coverage to acceptable levels compared to previous practices. Insight DFT provides early RT-level testability analysis and repair which can analyze and diagnose test issues dealing with robust test generation and yield better 1<sup>st</sup> pass ATPG results.