# Design For Test

Dr. Paul D. Franzon

## Outline

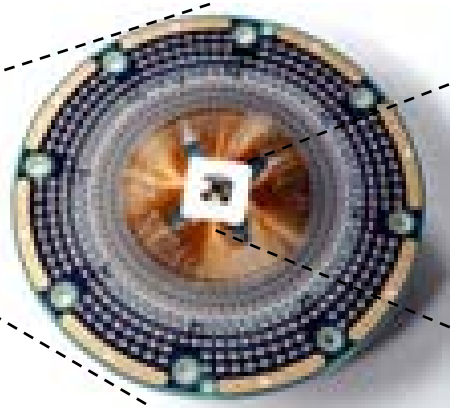
1. Scan-based testing
2. Exhaustive testing, Memory testing, and BIST
3. DFT in Synopsys Environment

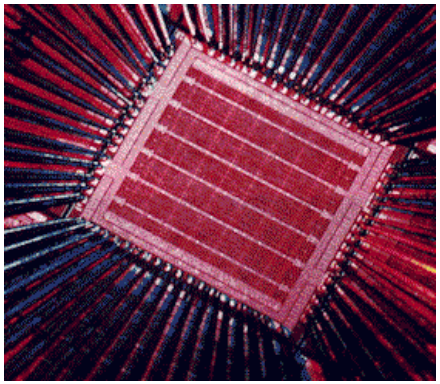
## References

1. Kurup, Chapter 6.
2. M. Smith, Chapter 14.

# Introduction

- Chip test needed after manufacturing so as to determine which parts are free of manufacturing defects
  - Goal is to discover manufacturing defects, not design bugs
  - Done before wafer dicing



# Chip test

- Goal:
  - Have a very high probability of finding manufacturing defects
    - Caused by: dust particles, fab control problems
    - Examples: Opens, shorts, transistors/wires out of specs
  - While minimizing the cost of finding the defects

# Chip Test and Cost

- Cost of test adds up to 40% to the cost of building the chip
- However, the cost of releasing a defective chip is very high
  - Packaging; Distribution; managing part return;
  - Reduction in customer satisfaction

# Chip Test Methods

- Dominant Method:
  - Scan Testing using stuck-at fault model
- Secondary Method:
  - Built In Self Test (BIST)

# Fault Models

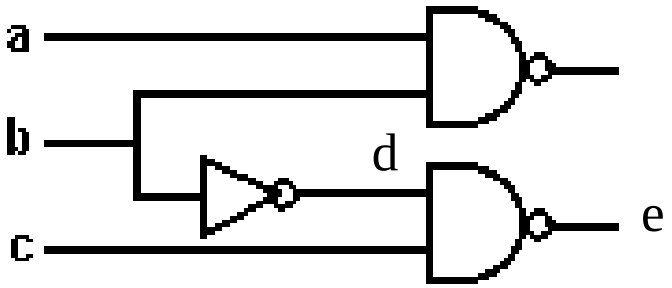
- Need a fault model so that a simulator can be used to determine the impact of a potential fault
  - i.e. Use the simulator to determine how to detect that fault
- Actual faults are mainly shorts and opens
  - Difficult and messy to model and simulate
- In practice using the stuck-at fault model captures the vast majority of “real” faults
  - i.e. Fault = Stuck-At-0 or Stuck-At-1

# *Using Stuck-At faults*

## Objectives:

- To sensitize and propagate faults to scanned outputs.
- To achieve high fault coverage (% of potential faults that are detectable)
- with fewest possible input choices

For example,




What are the test vectors that will sensitize a S-A-0 at d?

What test vector will propagate S-A-0 at d to e?

# D-algorithm

(+ its extensions)

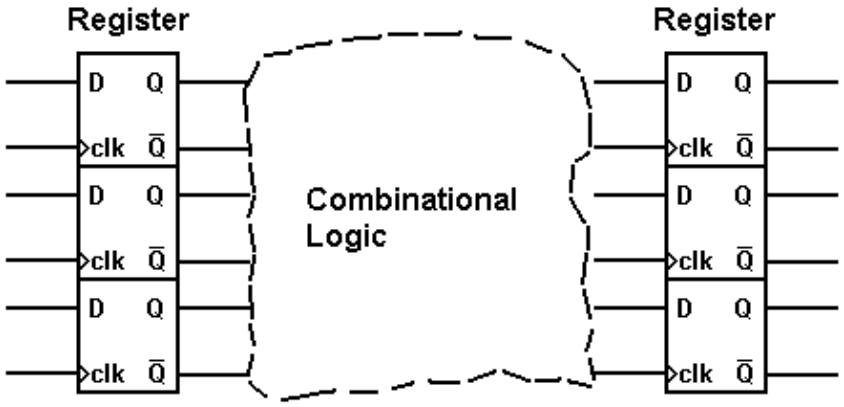
- Determines reasonably minimum test vector set that can detect a very high percentage of SA-0 and SA-1 faults at the outputs of the circuit
- % of faults detected = fault coverage

# Test Vectors

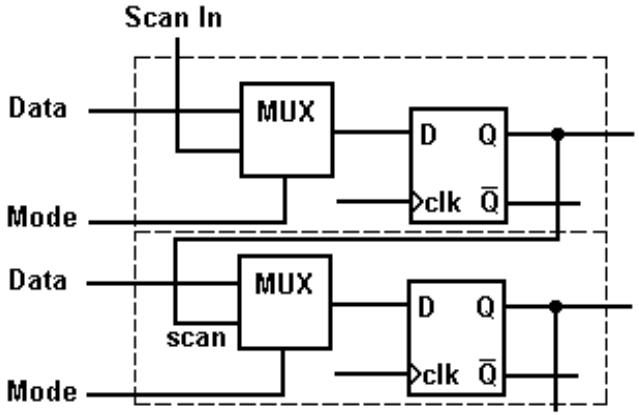
- Generate test vectors that sensitize and propagate internal faults to outputs
  - Observe outputs to determine if the results are correct or not
- Do you expect all faults to be discoverable from the chip I/O only?

# *Scan-Based Testing*

## Synchronous Logic Under Normal Mode of Operation:

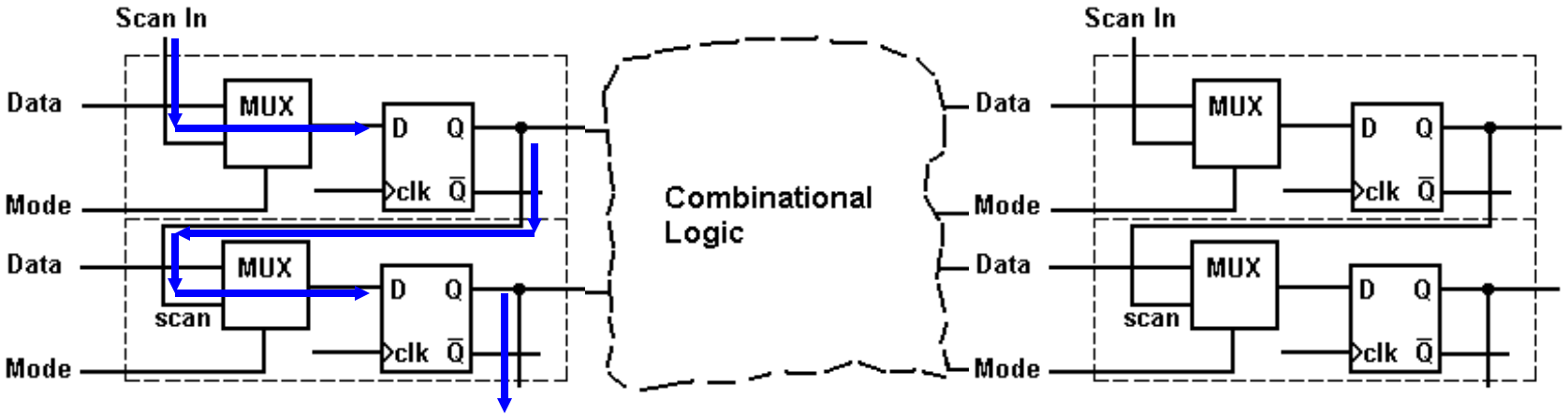



## Replace Register Cells with Scan Cell:



# *Scan-Based Testing...*

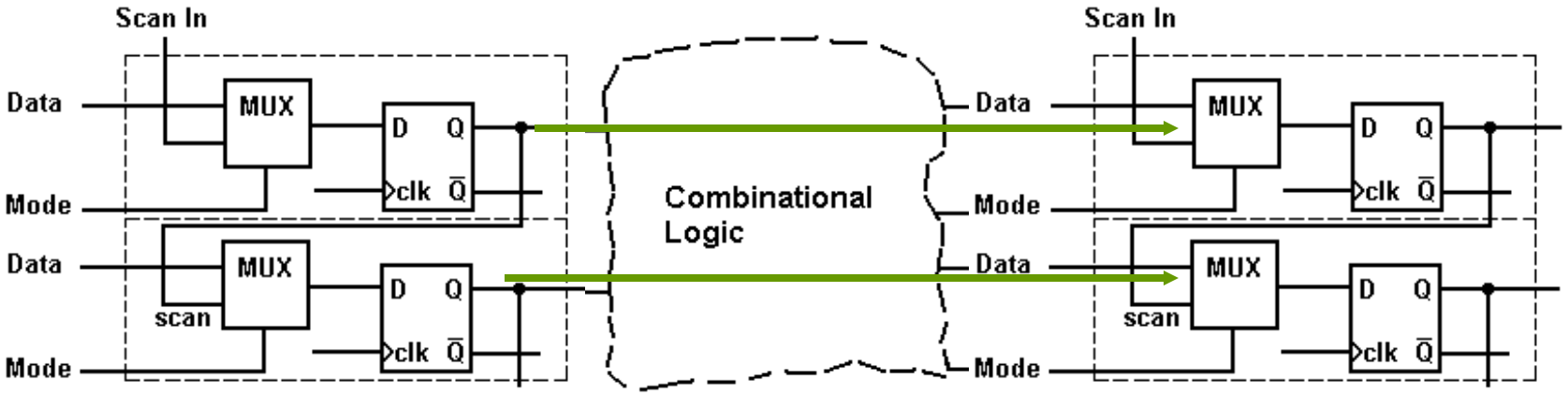
## Test Mode of Operation




Step 1 : Scan in test vector

# *Scan-Based Testing...*

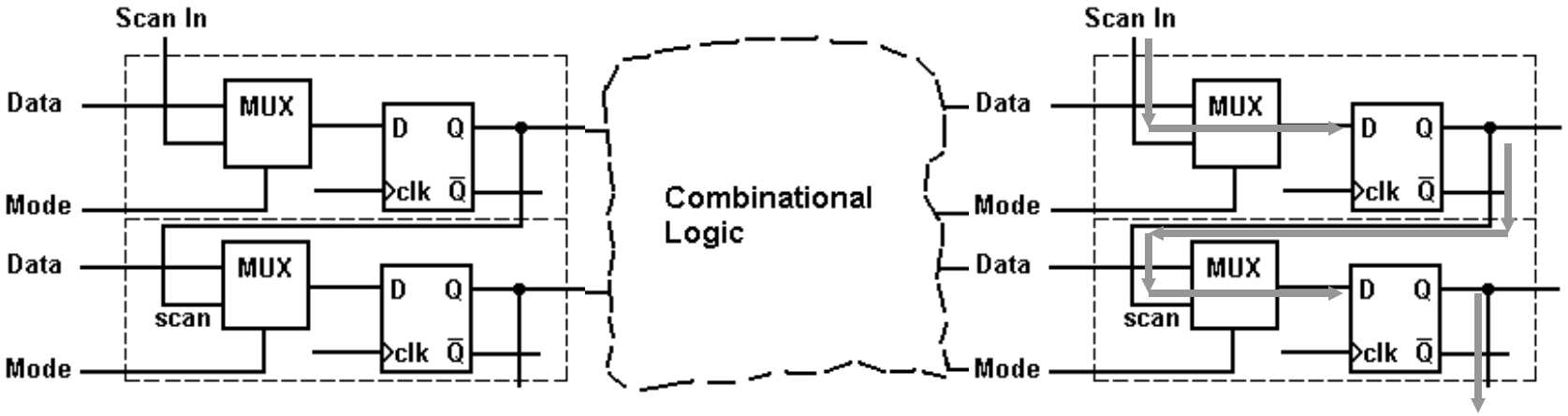
## Test Mode of Operation




Step 2 : Operate for one clock cycle

# *Scan-Based Testing...*

## Test Mode of Operation




Step 3 : Scan out result vector

Permits testing of all combinational logic in a design.

- Flip-flops tested through the scan chain itself

# Test Quality

## Test Escape

It can be shown that the percentage of parts that are still faulty but are not detected as being faulty, is given as [1]:

$$TE = 1 - Y^{(1 - T)}$$

where $TE$ is the test escape, $T$ is fault coverage, $Y$ is the yield.

For example, for $Y$ = 50% and $T$ = 95%, what is the test escape?

Typically, using the verification vectors (from your Verilog test fixture) for test gives a 80% coverage. What is the test escape then?

Typically, fault coverages above 99% give acceptably low test escapes.

[1] Source: Williams and Brown ("Defect level as a function of fault coverage," IEEE Trans Comp., C-30(12), December 1981, pp. 987-988.

# Partial Scan

Full Scan:

- Every flip-flop is included in a scan chain (generally several scan chains on the chip)

Partial Scan:

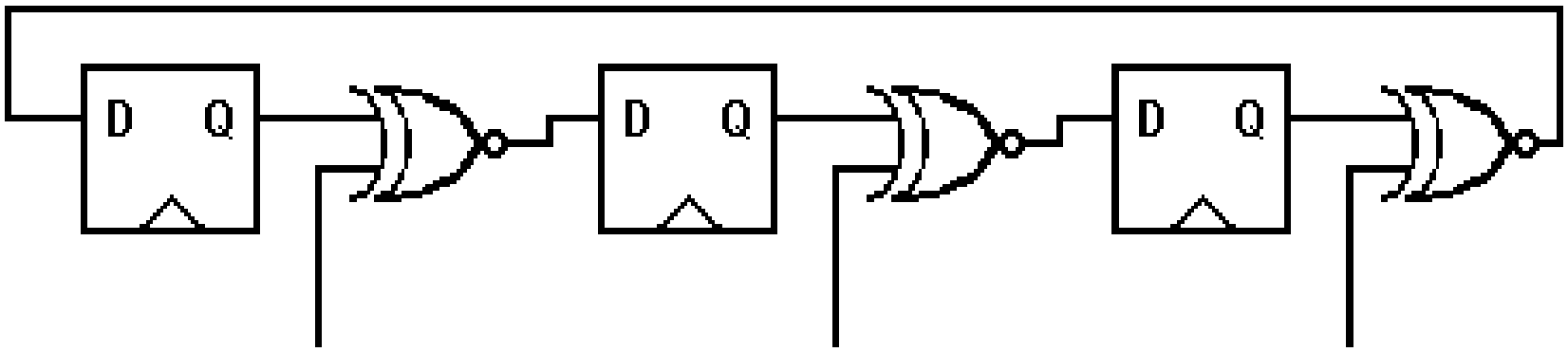
- Some flip-flops left out
- Works best on synchronous circuits without feedback
  - i.e. Excluded flip-flops treated as pass-throughs with a single cycle delay

# *Memory Testing*

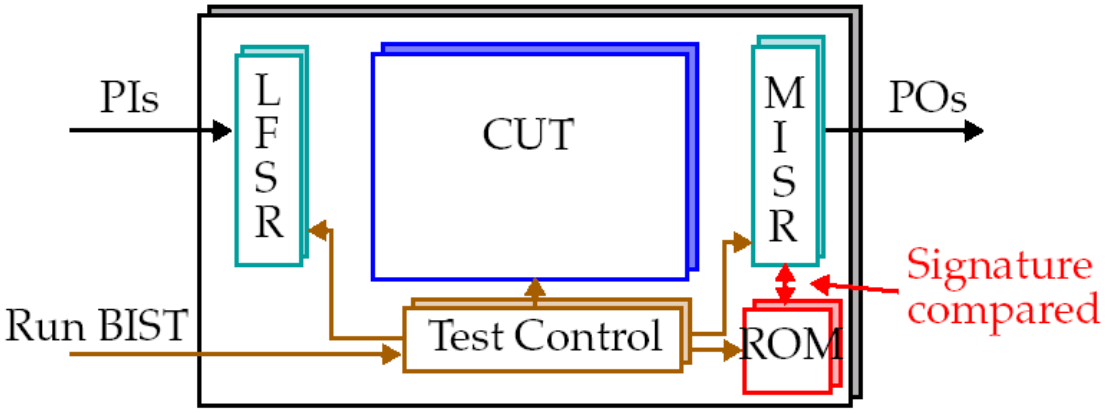
- Memories are exhaustively tested.
- Memories are tested for opens, shorts at individual locations and shorts between neighboring locations by **marching** test patterns through them.
- Memories are isolated by scan registers for testing purposes.

# *Built-In Self Test (BIST)*

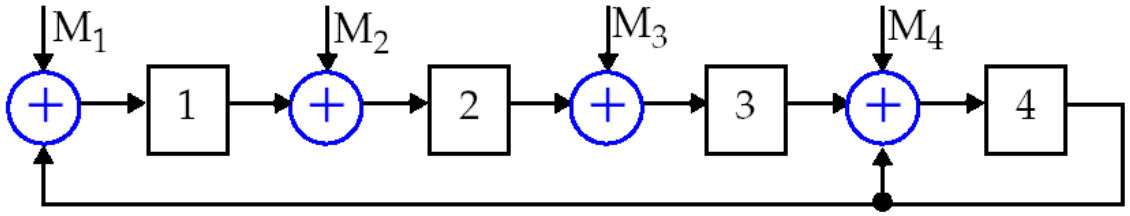
- In the most common BIST approach, the scan cells are modified to generate psuedorandom test vectors at the input to a logic block, and then to collect a signature at the output (using a linear feedback shift register).
- BIST takes more silicon area and more cycles (as pseudorandom) but saves on the cost of generating and storing test vectors.
- Also often takes less elapsed time as can often be run at full clock rate
- Automatic CAD support starting to appear



# BIST

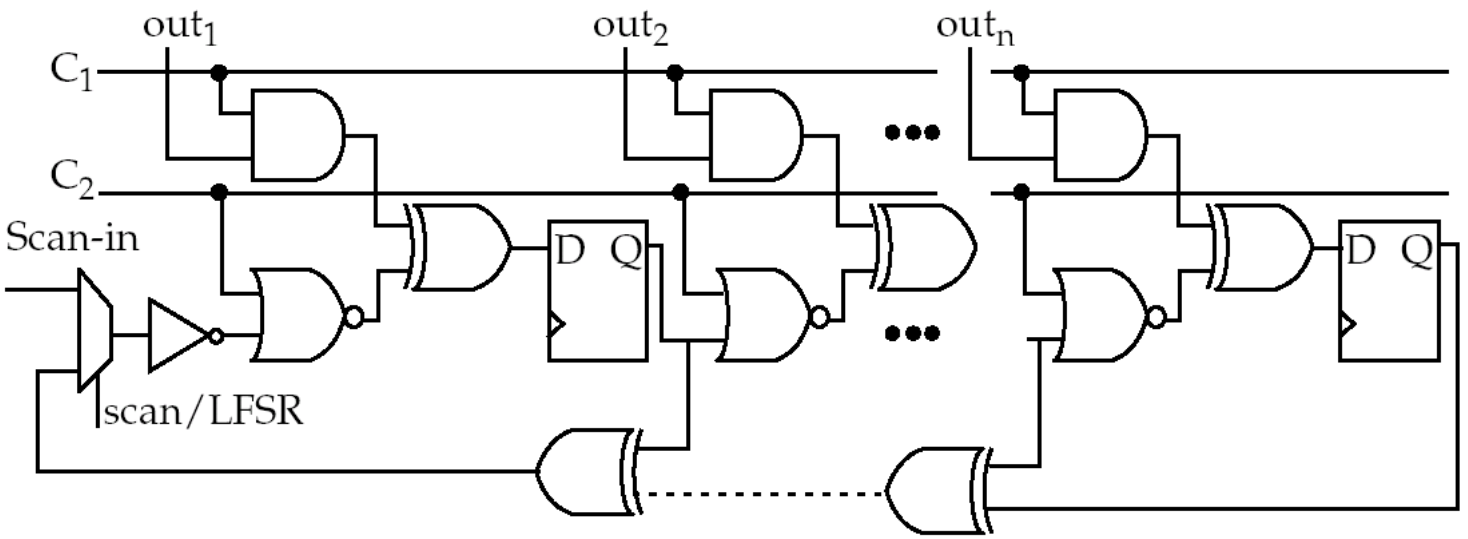



- LFSR:
  - Linear Feedback Shift Register
  - Used to generate pseudo-random sequence
- MISR
  - Multiple Signature Input Register
  - Used to generate signature of tested circuit



# BILBO

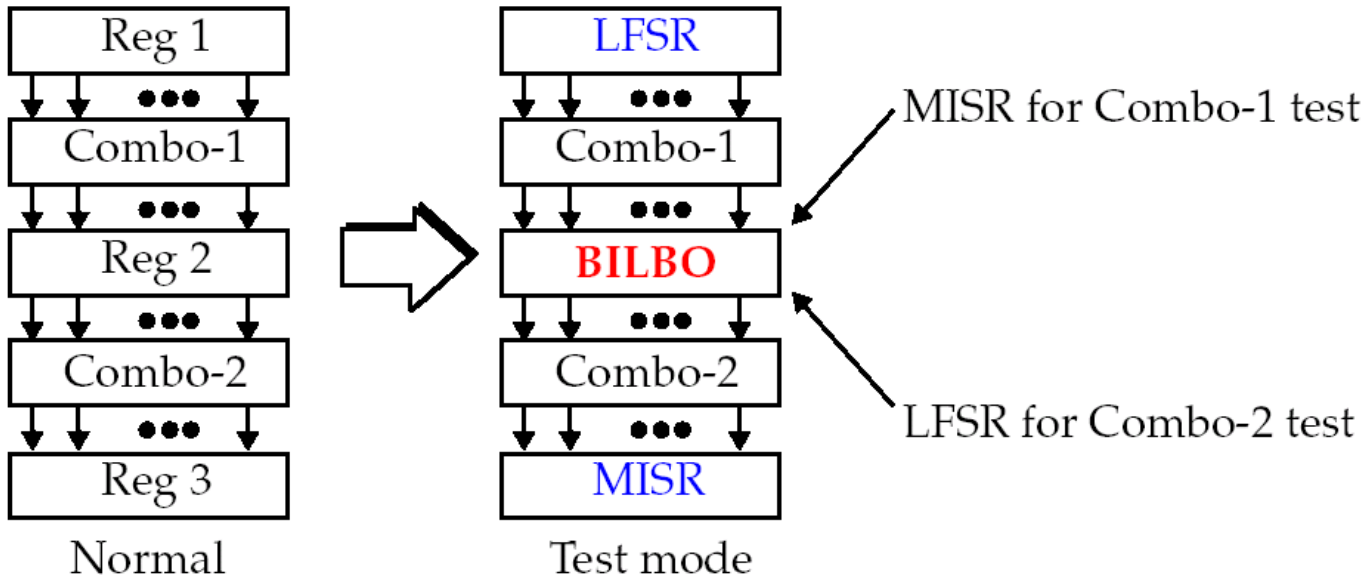
- Built-In Logic Block Observor:
  - Incorporates BIST into a scan architecture




$C_1$ and $C_2$ configure as a **shift register** for scan (00), an **LFSR** (00), **MISR** (10) a **Normal** (11).

# BILBO

- Scan-out signatures from BILBO registers after being configured as MISRs
  - Compare with expected result



# *Scan DFT in the Synopsys Environment*

Sample Synopsys script with a concentration on DFT. (Please see the tutorial [available in Iview] and manual for more details.)

```
read -f Verilog fsm.v
link_library = target_library = lsi_10k.db
create_clock clock81 -period 12.3

/* must expand design and click on clock81 first */
set_input_delay -clock clock81 -max -rise 2 “RW”
set_test_methodology full_scan

/* menu: attributes -> optimize directives -> design */
set_scan_style multiplexed_flip_flop

/* compile including scan */
compile map_effort low
```

# *Scan DFT in the Synopsys Environment...*

```
/* check for testability analysis -- look at result */
insert_test
check_test

/* create test patterns to check for ATPG conflicts (additional option in */
/* create pattern menu), also checks fault coverage */
create_test_patterns -sample 5

/* full test pattern creation and scan insertion are done complete chip at */
/* the end but try them if you want */
insert_test -scan_chains 1
create_test_patterns -output fsm.vdb \
-compaction_effort low \
-check_contention_float true -backtrack_effort low \
-random_pattern_failure_limit 64 -sample 100
```

# Advanced Topics in Test

- Memory Test
  - Different fault models, including bridging faults (write in one cell, and the neighbor also changes)
  - Usually relies on pre-determined patterns applied to memory
- Speed Testing
  - Scan chains normally run slowly, and can not detect delay faults
  - Can modify clock so that step 2 above can operate at full clock speed (just for one cycle)
  - Have to add test vectors designed to exercise critical paths
- Hierarchical Testing
  - To handle complexity of multicore SOCs
- Test Compression
  - To keep test vector set within size manageable by memory available in test equipment
- Debug
  - Using DFT features to help in debug of first chip

# Summary

- What is the purpose of DFT?
- What is the purpose of a fault model?
- What fault model is usually used?
- What is the value of scan based testing?