Digital VLSI Testing

Introduction to Testing and Diagnosis

- To determine the presence of faults, not the absence of faults, in a given circuit.
  - No amount of testing can guarantee that a circuit (chip, board or system) is fault free.
  - We carry out testing to increase our confidence in proper working of the circuit.
- Verification is an alternative to testing, used to verify the correctness of a design.
  - Simulation based methods.
  - Formal methods.

Test

- ***Test:*** A manufacturing step that ensures that the physical device, manufactured from the synthesized design, has no manufacturing defect.

Some Real Defects in Chips

- **Processing defects**
  - **Missing contact windows**
  - **Parasitic transistors**
  - **Oxide breakdown**
  - . . .
- **Material defects**
  - **Bulk defects (cracks, crystal imperfections)**
  - **Surface impurities (ion migration)**
  - . . .
- **Time-dependent failures**
  - **Dielectric breakdown**
  - **Electromigration**
  - . . .
- **Packaging failures**
  - **Contact degradation**
  - **Seal leaks**
  - . . .

Levels of Testing

- Testing can be carried out at the level of
  - Chip
  - Board
  - System
- Cost :: Rule of 10
  - It costs 10 times more to test a device as we move to the next higher level in the product manufacturing process.

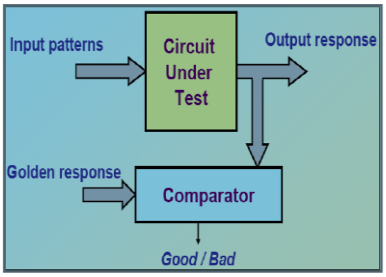
Levels of Testing

- Other ways to define levels:
  - Important to develop correct fault models and simulation models.
    - Transistor
    - Gate
    - RTL (Mux, ALU, Reg, etc.,)
    - Functional/Behavioral

Costs of Testing

- **Design For Testability (DFT)**
  - Chip area overhead and yield reduction
  - Performance overhead
- **Software processes of test**
  - Test generation and fault simulation
  - Test programming and debugging
- **Manufacturing test**
  - *Automatic test equipment* (ATE) capital cost
  - Test center operational cost

Basic Testing Principle



Fault Modeling

- I/O function tests inadequate for manufacturing (functionality versus component and interconnect testing)
- Real defects (often mechanical) too numerous and often not analyzable
- A fault model identifies targets for testing
- A fault model makes analysis possible
- Effectiveness measurable by experiments

Common Fault Models

- **Stuck-at faults**
  - **Single, multiple**
- **Transistor faults**
  - **open and short faults**
- **Memory faults**
  - **Coupling, pattern sensitive**
- **PLA faults**
  - **stuck-at, cross-point, bridging**
- **Delay faults**
  - **transition, path**
- **Functional faults (processors)**
- …..

Stuck-at Faults

- Some lines in the circuit are permanently stuck at logic 0 or logic 1.
- Two types
  - Single stuck-at faults
  - Multiple stuck-at faults

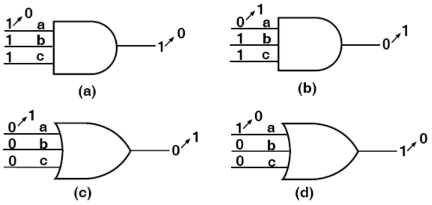
Single Stuck-at Fault

- Simpler to handle computationally
- Reasonably good fault coverage
  - A test set for detecting single stuck-at faults detects a large percentage of multiple stuck-at faults as well
- Three properties define a single stuck-at fault
  - Only one line is faulty
  - The faulty line is permanently set to 0 or 1
  - The fault can be at an input or output of a gate

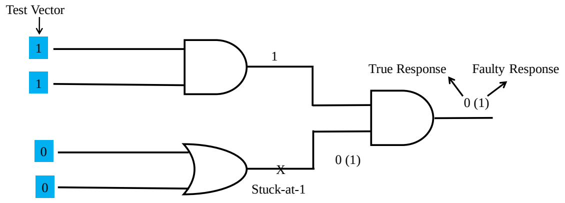
Single Stuck-at Fault

- For a circuit with k lines, the total number of single stuck-at faults possible is 2k.
- Most widely used fault model in the industry.

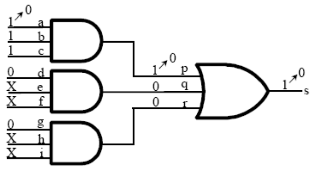
Testing AND and OR Gates for Stuck-at Faults



(a) (b) (c) (d)

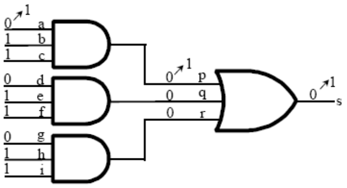


Testing an AND-OR Network



(a) stuck-at-0 test

Testing an AND-OR Network

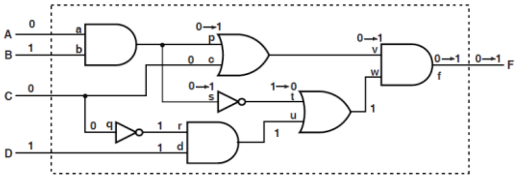


(b) stuck-at-1 test

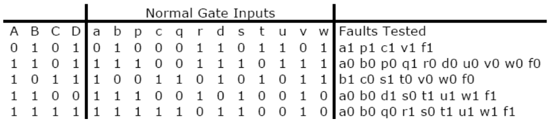
Test Vectors

a	b	c	d	e	f	g	h	i	Faults Tested
1	1	1	0	X	X	0	X	X	a0, b0, c0, p0
0	X	X	1	1	1	0	X	X	d0, e0, f0, q0
0	X	X	0	X	X	1	1	1	g0, h0, i0, r0
0	1	1	0	1	1	0	1	1	a1, d1, g1, p1, q1, r1
1	0	1	1	0	1	1	0	1	b1, e1, h1, p1, q1, r1
1	1	0	1	1	0	1	1	0	c1, f1, i1, p1, q1, r1

Example Network for Stuck-at Fault Testing



Tests for Stuck-at Faults



				Normal Gate Inputs												
A	B	C	D	a	b	p	c	q	r	d	s	t	u	v	w	Faults Tested
0	1	0	1	0	1	0	0	0	1	1	0	1	1	0	1	a1 p1 c1 v1 f1
1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	1	a0 b0 p0 q1 r0 d0 u0 v0 w0 f0
1	0	1	1	1	0	0	1	1	0	1	0	1	0	1	1	b1 c0 s1 t0 v0 w0 f0
1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	a0 b0 d1 s0 t1 u1 w1 f1
1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	0	a0 b0 q0 r1 s0 t1 u1 w1 f1

Undetectable faults

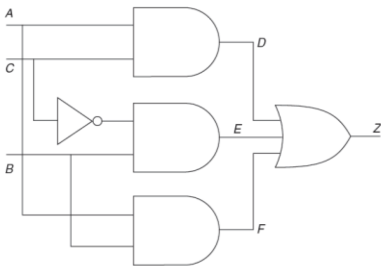
Consider the function

$$Z \ = \ A.C + B.\overline{C}$$

To avoid hazards,

$$Z \ = \ A.C + B.\overline{C} + A.B$$

Find a test for F0



Fault Equivalence

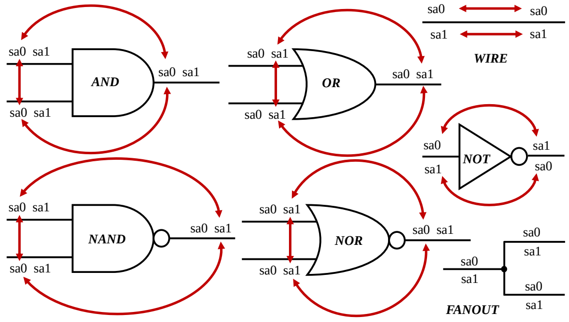
- ***Fault equivalence:*** Two faults f1 and f2 are equivalent if all tests that detect f1 also detect f2.
- If faults f1 and f2 are equivalent then the corresponding faulty functions are identical.
- *Two faults of a Boolean circuit are called equivalent iff they transform the circuit such that the two faulty circuits have identical output functions. Equivalent faults are also called indistinguishable and have exactly the same set of tests.*

Fault collapsing

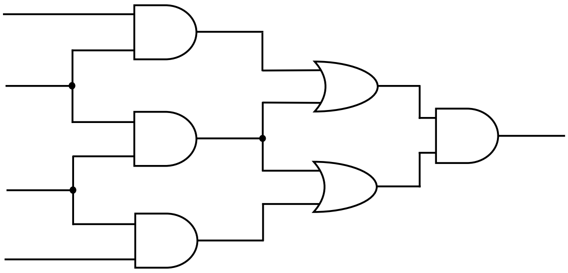
- All single faults of a logic circuit can be divided into disjoint equivalence subsets, where all faults in a subset are mutually equivalent. A collapsed fault set contains one fault from each equivalence subset.

$$\textbf{Collapse Ratio} = \frac{\textbf{Set of collapsed faults}}{\textbf{Set of all faults}}$$

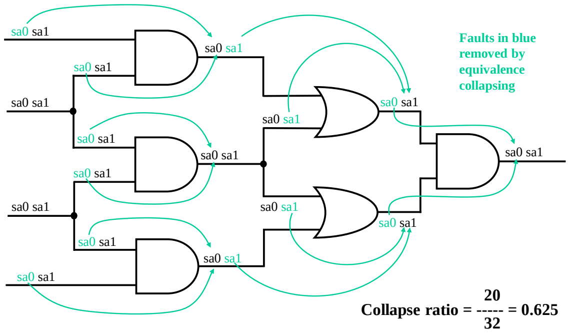
Equivalence Rules



Equivalence Example



Equivalence Example



Faults in blue removed by equivalence collapsing

$$\text{Collapse ratio} = \frac{20}{32} = 0.625$$

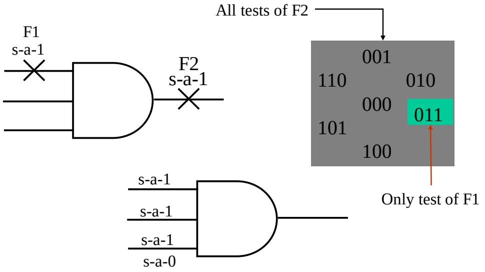
Fault Dominance

- If all tests of some fault F1 detect another fault F2, then F2 is said to dominate F1.
- Dominance fault collapsing:
  - If fault F2 dominates F1, then F2 is removed from the fault list.
- When dominance fault collapsing is used, it is sufficient to consider only the input faults of Boolean gates.

Fault Dominance

- In a tree circuit (without fanouts) PI faults form a dominance collapsed fault set.
- If two faults dominate each other then they are equivalent.

Dominance Example



A dominance collapsed fault set

Dominance Collapsing

- An *n-input Boolean gate requires n + 1 single stuck-at faults to be modeled.*
- To collapse faults of a gate, all faults from the output can be eliminated retaining one type (s-a-1 for AND and NAND; s-a-0 for OR and NOR) of fault on each input and the other type (s-a-0 for AND and NAND; s-a-1 for OR and NOR) on any one of the inputs.
- The output faults of the NOT gate, the non-inverting buffer, and the wire can be removed as long as both faults on the input are retained. No collapsing is possible for fanout.

Reference

- M. Bushnell and V. D. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits, Kluwer Academic Publishers, 2000.
- Miron Abramovici, Melvin Breuer, Arthur D Friedman ,Digital Systems Testing and Testable Design –Jaico Publishing House,2005
- Charles H Roth ,Jr , Digital Design using VHDL , Cenage Publishers , India Edition,2006

Design For Testability

Design For Testability

- **Design for testability** (DFT) refers to those design techniques that make test generation and test application cost-effective.
- Generally incorporated in design
- Goal
  - improve controllability and/or observability of internal nodes of a chip or PCB

DFT methods

- DFT methods for digital circuits:
  - Ad Hoc (Problem oriented)
  - Structured methods:
    - Scan
    - Partial Scan
    - Built-in self-test (BIST)
    - Boundary scan

Ad-Hoc DFT Methods

- Good design practices learnt through experience are used as guidelines:
  - Do-s and Don’ts
    - Avoid asynchronous (unclocked) feedback.
    - Avoid delay dependant logic.
    - Avoid self resetting logic.
    - Avoid gated clocks.
    - Avoid redundant gates.
    - Avoid large fanin gates.
    - Make flip-flops initializable.
    - Separate digital and analog circuits.
    - Provide test control for difficult-to-control signals.
    - Consider ATE requirements (tristates, etc.)

Ad-Hoc DFT Methods

- Design reviews conducted by experts or design auditing tools.
- Disadvantages of ad-hoc DFT methods:
  - Experts and tools not always available.
  - Test generation is often manual with no guarantee of high fault coverage.
  - Design iterations may be necessary.

Scan Design

- Objectives
  - Simple read/write access to all subset of storage elements in a design.
  - Direct control of storage elements to an arbitrary value (0 or 1)
  - Direct observation of the state of storage elements and hence the internal state of the circuit.
- Enhanced controlability and observability

Scan Design

- Circuit designed using pre-specified design rules.
- Test structure added to the verified design
  - Add one(or more) test control(TC) primary input.
  - Replace flip-flops by scan flip-flops and connect to form one or more shift registers in the test mode.
  - Make input/output of each scan shift register controllable/observable from PI/PO

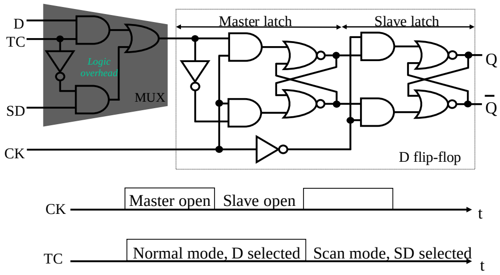
Scan Design

- Use combinational ATPG to obtain tests for all testable faults in the combinational logic.
- Add shift register tests and convert ATPG tests into scan sequences for use in manufacturing test.

Scan Design Rules

- Use only clocked D-type of flip-flops for all state variables.
- At least one PI pin must be available for test; more pins, if available, can be used.
- All clocks must be controlled from PIs.
- Clocks must not feed data inputs of flip-flops.

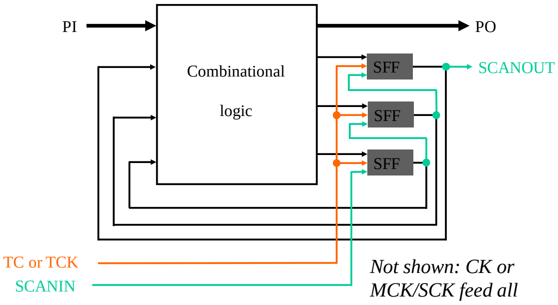
Scan Flip-Flop (SFF)



Scan Overheads

- IO pins: One pin necessary.
- Area overhead:
  - Gate overhead = [4 $n_{sff}/(n_g+10n_{ff})$] x 100%,
    - where n_g = comb. gates; n_{ff} = flip-flops;
    - Example – n_g = 100k gates, n_{ff} = 2k flip-flops, overhead = 6.7%.
  - More accurate estimate must consider scan wiring and layout area.
- Performance overhead:
  - Multiplexer delay added in combinational path; approx. two gate-delays.
  - Flip-flop output loading due to one additional fanout; approx. 5-6%.

Adding Scan Structure



Not shown: CK or MCK/SCK feed all SFFs.

Applying Tests for Scan Circuits

- Phase I (test the scan chain):
  - Shift test
  - Targets the scan flip-flops.
- Phase II (applying test patterns for combination circuits):
  - Target the single stuck-at faults in the combinational circuit.
  - Test vectors are generated by a combinational ATPG.

Phase I: Shift test

- Scan register must be tested prior to application of scan test sequences.
- A toggle sequence 00110011… of length n_{sff}+4 is scanned in. (n_{sff} is the maximum number of FFs in a scan chain.)
- Each SFF experiences all four transitions: 01, 00, 11, 10.
- The shift test covers most single stuck-at faults in the FFs.
- The shift test also verifies the correctness of the shift operation.

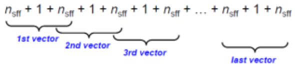
Phase II: Combinational Test

For each combinational test vector

1. Assert PI signals
2. Switch to test mode (scan)
3. Scan in
   a. Assert scan test patterns
   b. Apply test clock
   c. Repeat until all FFs are set
4. Switch to normal mode
5. Apply functional clock
6. Probe PO signals
7. Switch to test mode
8. Scan out

Testing Scan Register

- For each test vector, we need to shift in n_{sff} clock cycles (to setup the FFs) and apply one functional clock, and shift out with another n_{sff} clocks.
- The total number of clocks


$n_{sff} + 1 + n_{sff} + 1 + n_{sff} + 1 + n_{sff} + \ldots + n_{sff} + 1 + n_{sff}$
1st vector
2nd vector
3rd vector
last vector

- Scan test length for phase II= $n_{comb}(n_{sff} + 1) + n_{sff}$

n_{sff}: number of scan flip-flops;

n_{comb}: number of combinational tests.

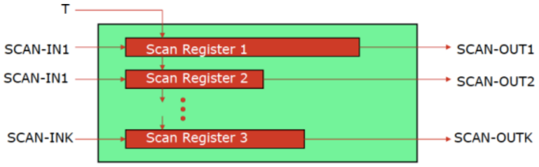
- Total scan test length for phase I and II:
  $= ((n_{sff} + 1)n_{comb} + n_{sff}) + (n_{sff} + 4)$ clock periods .
  $= (n_{comb} + 2)\ n_{sff} + n_{comb} + 4$ clock periods.

- Example: 2,000 scan flip-flops, 500 comb. vectors, total scan test length ~ 10^6 clocks.

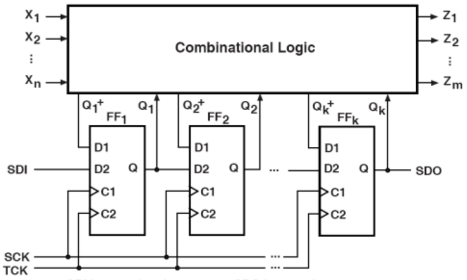
- Multiple scan registers reduce test length

Multiple Scan Registers

- Scan flip-flops can be distributed among any number of shift registers, each having a separate *scanin* and *scanout* pin.
- Test sequence length is determined by the longest scan shift register.
- Just one *test control* (TC) pin is essential.



Scan Path Test Circuit Using Two-port Flip-flops



SDI is scan data input
SCK is system clock

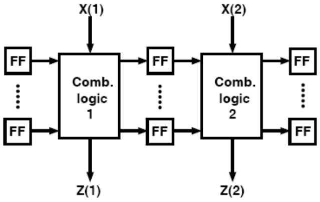
SDO is scan data output
TCK is test clock

Procedure for Scan Testing

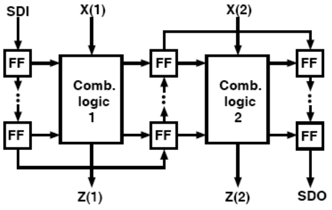
1. Scan in the test vector Q_i *values via SDI using the test clock TCK.*
2. Apply the corresponding test values to the X_i *inputs.*
3. After sufficient time for the signals to propagate through the combinational network, verify the output Z_i *values.*
4. Apply one clock pulse to the system clock *SCK to store the new values of* Q_i^+ into the corresponding flip-flops.
5. Scan out and verify the Q_i *values by pulsing the test clock TCK.*

6.Repeat steps 1 though 5 for each test vector.

System with Flip-flop Registers and Combinational Logic Blocks

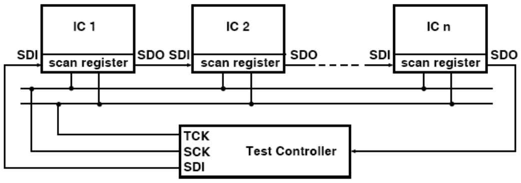


(a) Without scan chain



(b) With scan chain added

Scan Test Configuration with Multiple ICs



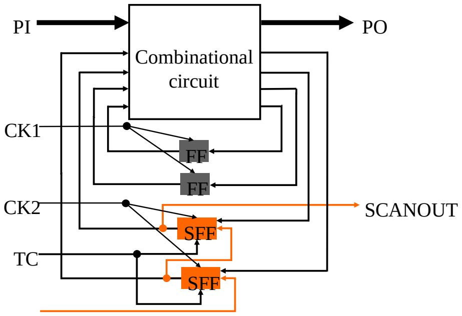
Problems with Scan Design

- Area overhead
  - Increased gate count
  - Increased routing area
  - Additional pin requirement.
- Performance degradation
  - Extra gate delay due to the multiplexer
  - Extra delay due to the capacitive loading of the scan-wiring at each flip-flop's output
- Long test application time.
- Not applicable to all designs.
  - Must follow the scan design rules.
- High power dissipation during testing.

Partial-Scan

- A subset of flip-flops is scanned.
- Objectives:
  - Minimize area overhead and scan sequence length, yet achieve required fault coverage
  - Exclude selected flip-flops from scan:
    - Improve performance
    - Allow limited scan design rule violations
  - Allow automation:
    - In scan flip-flop selection
    - In test generation
  - Shorter scan sequences

Partial-Scan Architecture



Partial-Scan -Summary

- Partial-scan is a generalized scan method; scan can vary from 0 to 100%.
- Partial-scan has lower overheads (area and delay) and reduced test length.
- Partial-scan allows limited violations of scan design rules, e.g., a flip-flop on a critical path may not be scanned.

Reference

- M. Bushnell and V. D. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits, Kluwer Academic Publishers, 2000.
- Miron Abramovici, Melvin Breuer, Arthur D Friedman ,Digital Systems Testing and Testable Design –Jaico Publishing House,2005
- Charles H Roth ,Jr , Digital Design using VHDL , Cenage Publishers , India Edition,2006

DFT - Boundary Scan

Motivation for Boundary Scan Standard

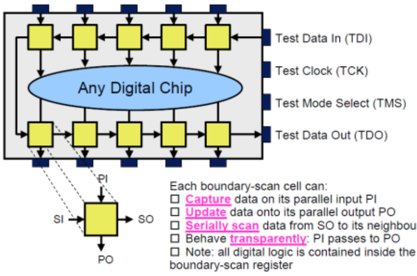
- Components put on both sides of PCB
- Reduced spacing between PCB wires
- PCB tester replaced with build-in test delivery system - JTAG
  - Need to add standard System Test Port and Bus to provide I/O communication
- Components on PCB from different vendors
  - Test bus identical for various components
  - One chip providing test hardware for other chips

Purpose of Standard

- Test instructions and data serially fed into CUT
  - Possibility to read out test results
  - Too many shifts to shift in external tests
- JTAG controlling scan operation on chip, PCB and system level
- Characteristics
  - Other chips used to collect test responses from CUT
  - System interconnects to be tested separately from components
  - Components tested separately from wires

Boundary-Scan

- Consists of adding scan registers to the inputs and outputs of ICs
- In a boundary-scan device, each digital primary input signal and primary output signal is supplemented with a multi-purpose memory element called a boundary-scan cell.
- Requires the addition four I/O ports to the chip - Test Access Port (TAP)
  - TCK - test clock
  - TMS - test mode signal
  - TDI - serial test data in
  - TDO - serial test data out
- Also requires the addition of logic to control the testing process - TAP Controller



Each boundary-scan cell can:

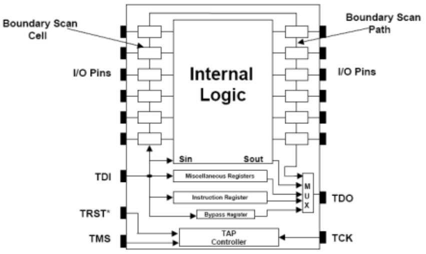
□ Capture data on its parallel input PI
□ Update data onto its parallel output PO
□ Serially scan data from SO to its neighbour's SI
□ Behave transparently: PI passes to PO
□ Note: all digital logic is contained inside the boundary-scan register

- A parallel load operation — called a **Capture** operation — causes signal values on device input pins to be loaded into input cells, and signal values passing from the internal logic to device output pins to be loaded into output cells.
- A parallel unload operation — called an **Update** operation — causes signal values already present in the output scan cells to be passed out through the device output pins. Signal values already present in the input scan cells will be passed into the internal logic.
- Data can also be Shifted around the shift register, in serial mode, starting from a dedicated device input pin called Test Data In (TDI) and terminating at a dedicated device output pin called Test Data Out (TDO).
- The Test ClocK, TCK, is fed in via yet another dedicated device input pin and the various modes of operation are controlled by a dedicated Test Mode Select (TMS) serial control signal.

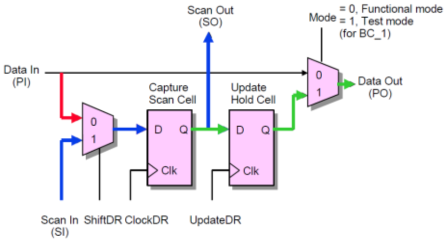
Boundary Scan

IEEE 1149.1 JTAG Boundary Scan Standard

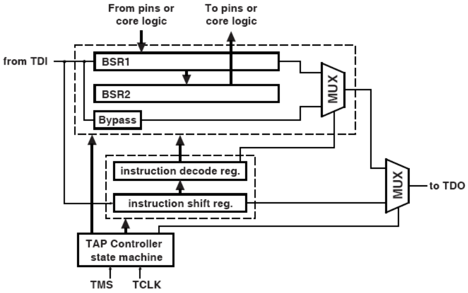
(JTAG - Joint Test Action Group)



Boundary Scan Cell



Basic Boundary Scan Architecture



Boundary-Scan Cell Modes

Normal Mode: *Mode_Control* = ‘0’
- Data passes from *IN* to *OUT*

Scan Mode: *ShiftDR* = ‘1’, *ClockDR* = scan clock
- Serial data is shifted in from *SIN* and out to *SOUT*
- TDI->...->SIN->SOUT->...TDO

Capture Mode: *ShiftDR* = ‘0’, *ClockDR* = 1 clock pulse
- Data on the *IN* line is clocked into Q_A
- IN-> QA, OUT driven by IN or QB

Update Mode: with Q_A loaded, *Mode_Control* = ‘1’, *UpdateDR* = 1 clock pulse
- Data clock into Q_A is applied to *OUT*
- QA->OUT

To shift in and apply data, the scan mode would be selected until the data is shifted in and then one cycle of update mode would be selected. To capture data and scan it out, one cycle of capture mode would be selected followed by the required number of scan cycles.

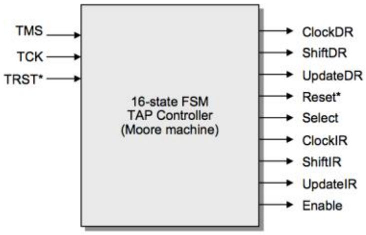
Hardware Components of 1149.1

- TAP (Test Access Port) :
  - TMS (Test Mode Select), TDI (Test Data In), TDO (Test Data Out), and TCK (Test Clock), TRST (power-up reset)
- TAP Controller :
  - A finite state machine with 16 states
  - Input : TCK, TMS
  - Output : 9 or 10 signals included ClockDR, UpdateDR, ShiftDR, ClockIR, UpdateIR, ShiftIR, Select, Enable, TCK and the optional TRST*.
- IR (Instruction Register)
- TDR (Test Data Registers) :
  - Mandatory: Boundary scan register and 1 bit Bypass register
  - Optional: Device-ID register, Design-Specific registers, etc.

Signals at Test Access Port (TAP)

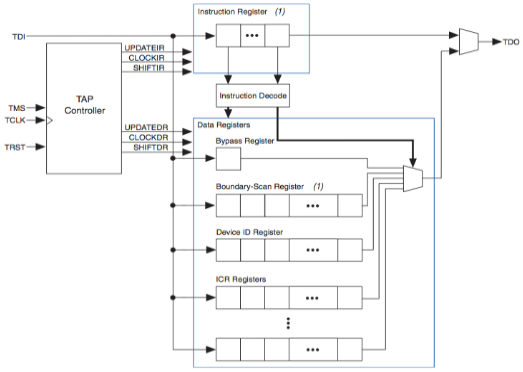
- **Test Clock Input (TCK)** - Clock for test logic
  - Possibly running at different rate than system clock
- **Test Mode Select (TMS)** - Switching system from functional to test mode
- **Test Data Input (TDI)** - Accepting serial test data and instructions
  - Used to shift-in vectors or one of many test instructions
- **Test Data Output (TDO)** - Serially shifting-out test results captured in boundary scan chain (or device ID or other internal register)
- **Test Reset (TRST)** - Optional asynchronous TAP controller reset

TAP Controller

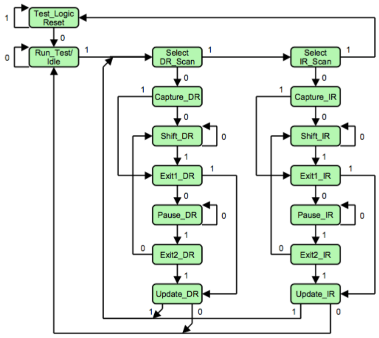


TAP Controller

- TMS and TCK (and the optional TRST*) go to a 16-state finite-state machine controller, which produces the various control signals.
- These signals include
  - Dedicated signals to the Instruction register (ClockIR, ShiftIR, UpdateIR) and
  - Generic signals to all data registers (ClockDR, ShiftDR, UpdateDR).
    - Data Register means any target register except the Instruction register
  - The data register that actually responds is the one enabled by the conditional control signals generated at the parallel outputs of the Instruction register, according to the particular instruction.

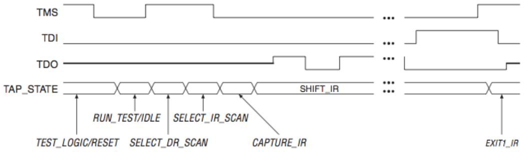


State Diagram of TAP Controller



- The state transition is controlled by the value of TMS. A state transition occurs on the positive edge of TCK and the controller output values change on the negative edge of TCK.

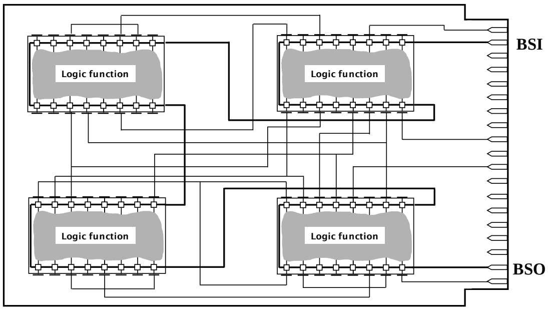
Selecting the Instruction Mode



States of TAP Controller

- Test-Logic-Reset: normal mode
- Run-Test/Idle: Awake, and do nothing state
- Select-DR-Scan: initiate a data-scan sequence
- Capture-DR: load test data in parallel
- Shift-DR: load test data in series
- Exit1-DR: finish phase-1 shifting of data
- Pause-DR: temporarily hold the scan operation (allow the bus master to reload data)
- Exit2-DR: finish phase-2 shifting of data
- Update-DR: parallel load from associated shift registers

PCB with Boundary Scan



Instruction Set

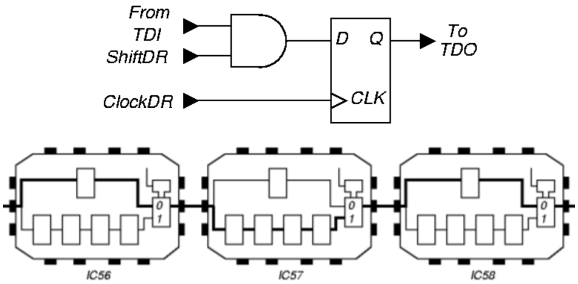
- BYPASS - Bypass data through a chip
- SAMPLE - Sample (capture) test data into BSR
- PRELOAD - Shift-in test data and update BSR
- EXTEST - Test interconnection between chips of board
- Optional - INTEST, RUNBIST, CLAMP, IDCODE, USERCODE, HIGH-Z, etc.

Optional / Required Instructions

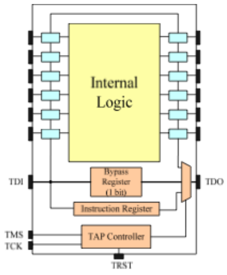
Instruction	Status
BYPASS	**Mandatory**
CLAMP	**Optional**
EXTEST	**Mandatory**
HIGHZ	**Optional**
IDCODE	**Optional**
INTEST	**Optional**
RUNBIST	**Optional**
SAMPLE / PRELOAD	**Mandatory**
USERCODE	**Optional**

BYPASS Instruction

- **Purpose: Bypasses scan chain with 1-bit register**



Execution of BYPASS Instruction



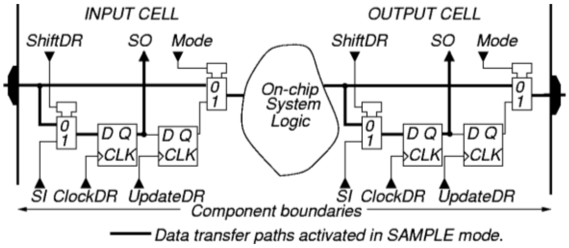
SAMPLE/PRELOAD Instruction

- The instruction sets up the boundary-scan cells either to **sample** (capture) values or to **preload** known values into the boundary-scan cells prior to some follow-on operation.
- Both instructions leave the device in **functional** mode, not test mode
  - i.e., any values preloaded or sampled into the boundary scan cells are not passed through into the device or to the device output pins, unlike ***Extest***.
  - Thus the device is still under the control of the mission mode signals i.e. is in functional mode.
  - This is an important distinction between ***Preload*** and ***Sample***, compared to ***Extest***.

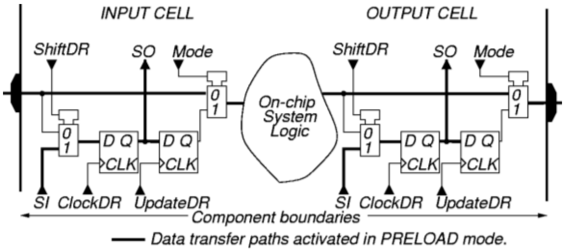
SAMPLE Instruction

Purpose:

1. used to capture mission-mode signals into the boundary-scan cells

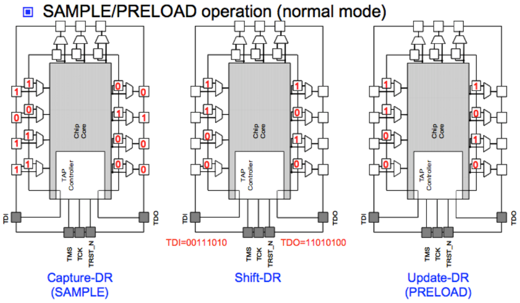


PRELOAD Instruction



SAMPLE/PRELOAD Instruction

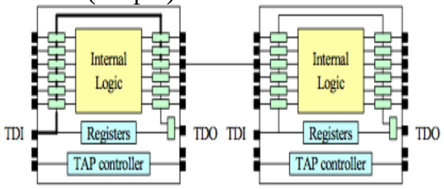
- SAMPLE/PRELOAD operation (normal mode)



EXTEST Instruction (1/3)

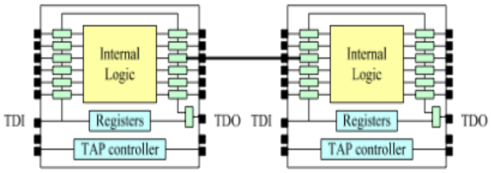
Testing the interconnect structure between two devices using the boundary-scan cells is called ***External Test***, or ***Extest***

Shift DR (Chip 1)



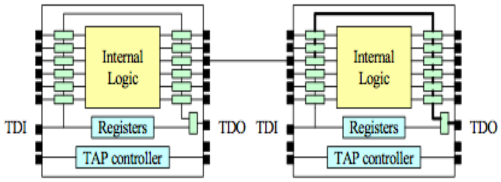
EXTEST Instruction (2/3)

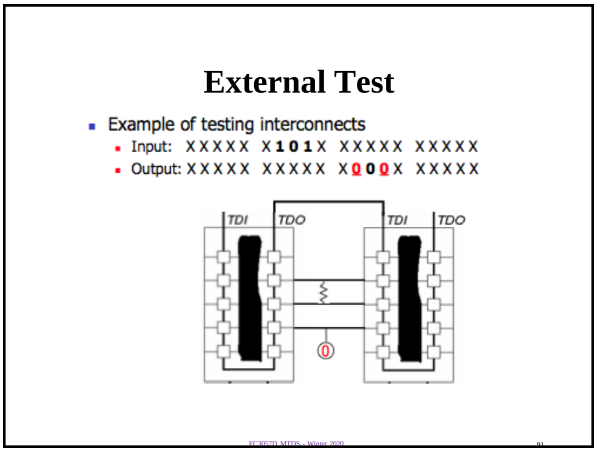
Update DR (Chip 1)
Capture DR (Chip 2)



EXTEST Instruction (3/3)

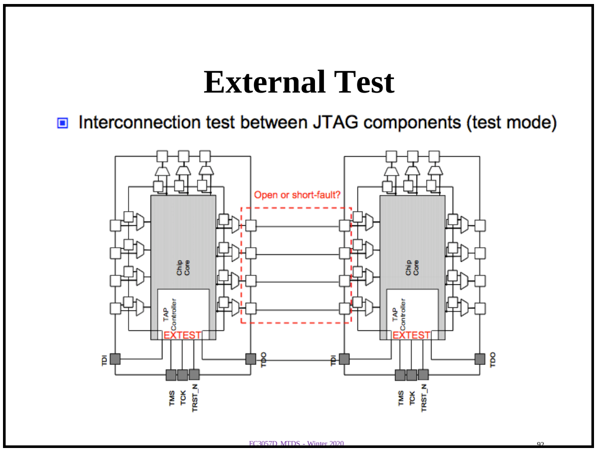
Shift DR (Chip 2)





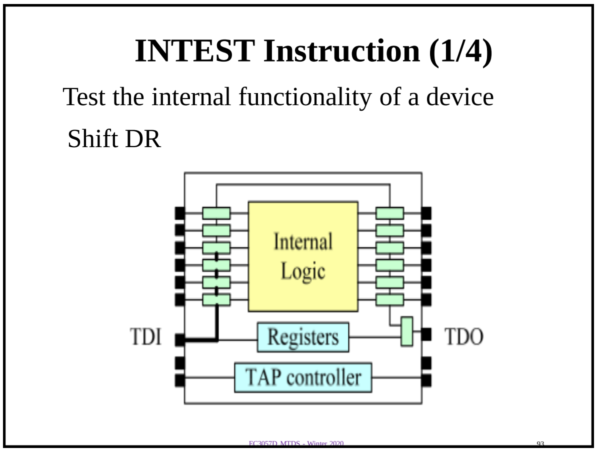
External Test

- Example of testing interconnects
  - Input: XXXXX X**101**X XXXXX XXXXX
  - Output: XXXXX XXXXX X**000**X XXXXX



External Test

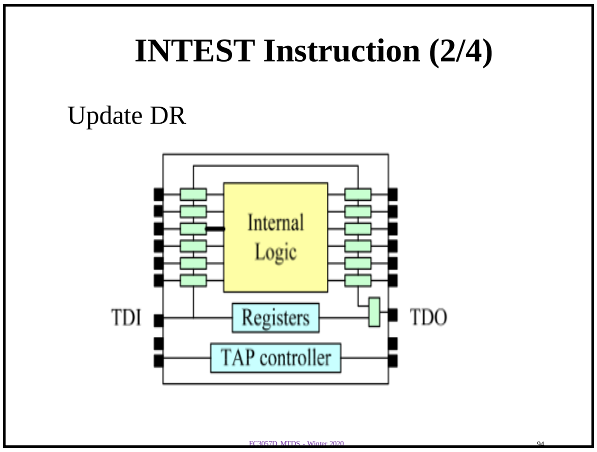
- Interconnection test between JTAG components (test mode)



INTEST Instruction (1/4)

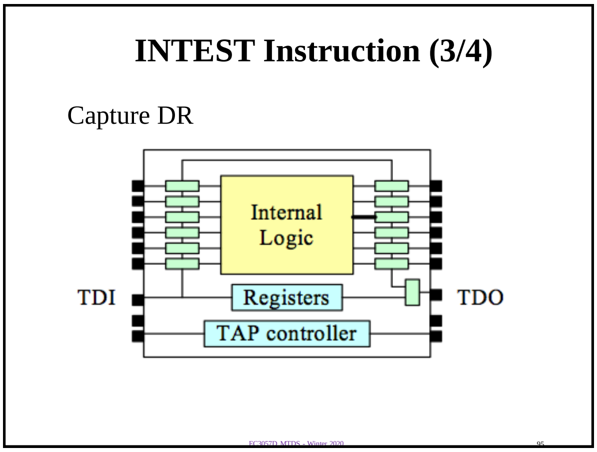
Test the internal functionality of a device

Shift DR



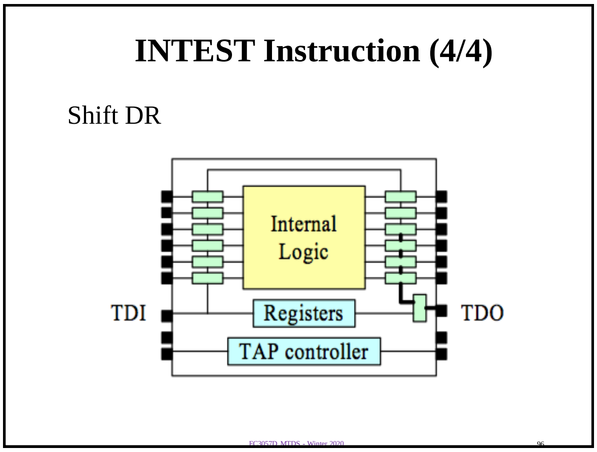
INTEST Instruction (2/4)

Update DR



INTEST Instruction (3/4)

Capture DR

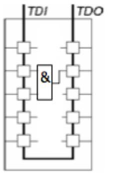


INTEST Instruction (4/4)

Shift DR

Internal Test

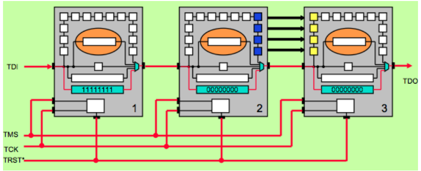
- Example of testing chip logic
  - Input: X 1 0 X X X X X X X
  - Good output: X 0 X X X X X X X X



Basic operations

1. Instruction sent (serially) through TDI into instruction register
2. Selected test circuitry configured to respond to the instruction
3. Test pattern shifted into selected data register and applied to logic to be tested
4. Test response captured into some data register
5. Captured response shifted out; new test pattern shifted in simultaneously
6. Steps 3-5 repeated until all test patterns are applied

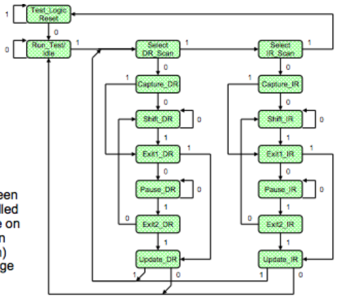
Example



Aim : Test the four interconnects between Chips 2 and 3.

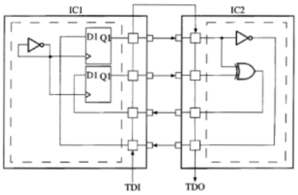
- Chip 1 has to be in ***Bypass*** mode and chips 2 and 3 have to be in ***Extest*** mode to set up tests to check the four interconnects between Chips 2 and 3.
- This is done by loading the ***Bypass*** instruction (all-1s) into the Instruction register of chip 1, and the ***Extest*** instruction (assumed to be all-0s) into the Instruction registers of Chips 2 and 3.

- Step 1 - Connect the Instruction registers of all three devices between their respective TDI and TDO pins.
  - This is achieved by a special sequence of values on the board control line TMS going to each TAP controller in each device.
- Step 2 – Load the appropriate instructions into the various Instruction registers via the global connection of Instruction registers.
  - If we assume simple two-bit Instruction registers per device, this operation amounts to a 6 x TCK serial load of the sequence **110000** into the edge-connector TDO entry to place **00** in the Instruction registers of Chips 2 and 3, and **11** in the Instruction register of Chip 1.
- Step 3 - Continue with values on TMS to cause each TAP controller to issue the control-signal values to transfer the instruction codes in the scan sections of the Instruction registers to the hold sections where they become the current instruction. This is the Update operation.
- At this point, the various instructions are ***executed*** — that is, Chip 1 deselects the Instruction register and selects the Bypass register between its TDI and TDO (***Bypass*** instruction), and Chips 2 and 3 deselect their Instruction registers and select their Boundary-Scan registers between their TDI and TDO (***Extest*** instruction). Devices 2 and 3 are now set up ready for ***Extest*** operation.



Transitions between states are controlled by the logic value on TMS (as shown in the state diagram) plus the rising edge of TCK.

Example – Interconnection Testing



- IC1 has clock oscillator and two FFs
- IC2 has an inverter and XOR gate
- The two ICs are connected to form an 2 bit binary counter

Example – Interconnection Testing

- Reset the TAP state machine to the test-logic-reset state by applying a sequence of five 1’s on TMS or by applying TRST
- Scan in SAMPLE/PRELOAD instruction (001) to both IC’s using TMS and TDI as shown below

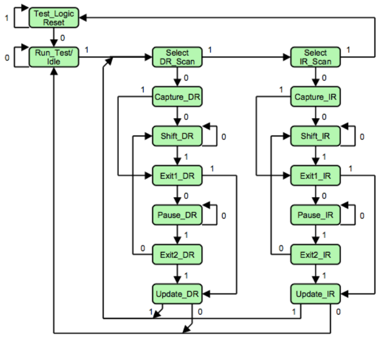
```
State: 0   1   2   9   10 11  11  11  11  11  11  12  15  2
TMS:   0   1   1   0   0   0   0   0   0   0   1   1   1
TDI:   -   -   -   -   -   1   0   0   1   0   0   -   -
```

- Preload the first test data into the IC’s using TMS and TDI as shown below

```
State: 2 3 4 4 4 4 4 4 4 4 5 8 2
TMS:   0 0 0 0 0 0 0 0 0 1 1 1
TDI:   - - 0 1 0 0 0 1 0 0 - -
```

  - Data is shifted into boundary scan cells in the Shift-DR state and will be shifted out in the update-DR state

State Diagram of TAP Controller



- The state transition is controlled by the value of TMS. A state transition occurs on the positive edge of TCK and the controller output values change on the negative edge of TCK.

- Scan in the EXTEST instruction to both IC’s

```
State: 2   9   10 11  11  11  11  11  11  12  15  2
TMS:   1   0   0   0   0   0   0   0   1   1   1
TDI:   -   -   -   0   0   0   0   0   0   -   -
```

  - The EXTEST instruction (000) is scanned into the instruction register in Shift-IR state and loaded into the instruction decode register in the state Update-IR.
  - At this time, the preloaded test data goes to the output pins and is transmitted to the adjacent IC input pins.
- Capture the test results from the IC inputs. Scan this data out to TDO and scan in the second test data.

```
State: 2 3 4 4 4 4 4 4 4 4 5 8 2
TMS:   0 0 0 0 0 0 0 0 0 1 1 1
TDI:   - - 1 0 0 0 1 0 0 0 - -
TDO:   - - x x 1 0 x x 1 0 - -
```

  - The new data is loaded into the boundary scan-cell in the Update-DR state

- Capture the test results from the IC inputs. Scan this data out to TDO and scan in the second test data.

```
State: 2 3 4 4 4 4 4 4 4 4 5 8 2 9 0
TMS:   0 0 0 0 0 0 0 0 0 1 1 1 1 1
TDI:   - - 0 0 0 0 0 0 0 0 - - - -
TDO:   - - x x 0 1 x x 0 1 - - - -
```

Built-In Self-Test (BIST)

BIST Motivation

- ❑ Useful for field test and diagnosis (less expensive than a local automatic test equipment)
- ❑ Software tests for field test and diagnosis:
  - ▪ Low hardware fault coverage
  - ▪ Low diagnostic resolution
  - ▪ Slow to operate
- ❑ Hardware BIST benefits:
  - ▪ Lower system test effort
  - ▪ Improved system maintenance and repair
  - ▪ Improved component repair
  - ▪ Better diagnosis at component level

Costly Test Problems Alleviated by BIST

- Increasing chip logic-to-pin ratio – harder observability
- Increasingly dense devices and faster clocks
- Increasing test generation and application times
- Increasing size of test vectors stored in ATE
- Expensive ATE needed for GHz clocking chips
- Hard testability insertion – designers unfamiliar with gate-level logic, since they design at behavioral level
- *In-circuit testing* no longer technically feasible
- Circuit testing cannot be easily partitioned

Benefits and Costs of BIST with DFT

Level	Design and test	Fabri-cation	Manuf. Test	Maintenance test	Diagnosis and repair	Service interruption
Chips	+ / -	+	-			
Boards	+ / -	+	-		-	
System	+ / -	+	-	-	-	-

+ Cost increase
- Cost saving
+/- Cost increase may balance cost reduction

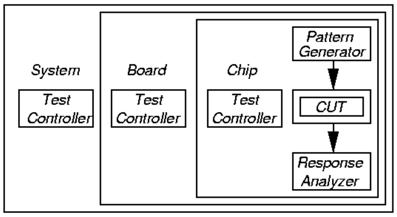
Economics – BIST Costs

- **Chip area overhead for:**
  - **Test controller**
  - **Hardware pattern generator**
  - **Hardware response compacter**
  - **Testing of BIST hardware**
- **Pin overhead -- At least 1 pin needed to activate BIST operation**
- **Performance overhead – extra path delays due to BIST**
- ***Yield loss* – due to increased chip area or more chips In system because of BIST**
- **Reliability reduction – due to increased area**
- **Increased BIST hardware complexity – happens when BIST hardware is made testable**

BIST Benefits

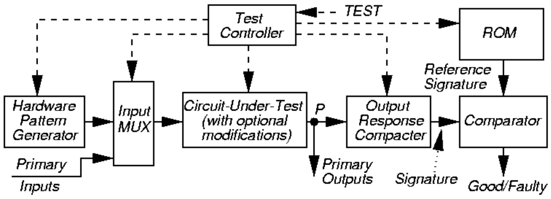
- **Faults tested:**
  - **Single combinational / sequential stuck-at faults**
  - **Delay faults**
  - **Single stuck-at faults in BIST hardware**
- **BIST benefits**
  - **Reduced testing and maintenance cost**
  - **Lower test generation cost**
  - **Reduced storage / maintenance of test patterns**
  - **Simpler and less expensive ATE**
  - **Can test many units in parallel**
  - **Shorter test application times**
  - **Can test at functional system speed**

BIST Process



- ***Test controller* – Hardware that activates self-test simultaneously on all PCBs**
- **Each board controller activates parallel chip BIST Diagnosis effective only if very high fault coverage**

BIST Architecture

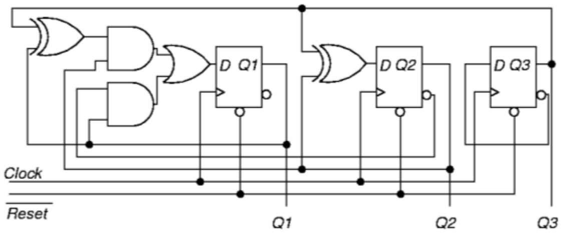


- **Note: BIST cannot test wires and transistors:**
  - **From PI pins to Input MUX**
  - **From POs to output pins**

Pattern Generation

- **Store in ROM – too expensive**
- ***Exhaustive***
- ***Pseudo-exhaustive***
- ***Pseudo-random* (LFSR) – Preferred method**
- **Binary counters – use more hardware than LFSR**
- **Modified counters**
- **Test pattern *augmentation***
  - **LFSR combined with a few patterns in ROM**
  - ***Hardware diffracter* – generates pattern cluster in neighborhood of pattern stored in ROM**

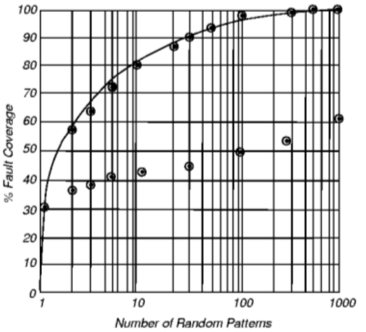
Exhaustive Pattern Generation (A Counter)



- **Shows that every state and transition works**
- **For *n*-input circuits, requires all 2^n vectors**
- **Impractical for large *n* (> 20)**

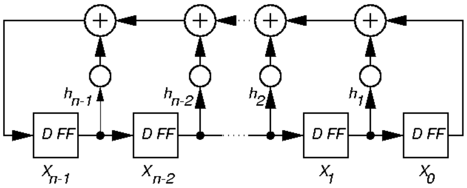
Random Pattern Testing

Bottom: Random-Pattern Resistant circuit



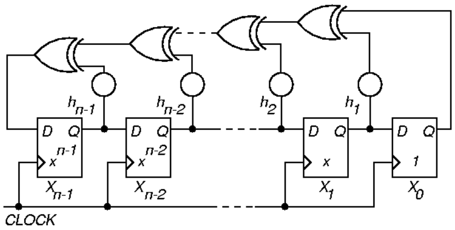
(a) Top curve -- random pattern testing with acceptable fault coverage.
(b) Bottom curve -- unacceptable random pattern testing.

Pseudo-Random Pattern Generation



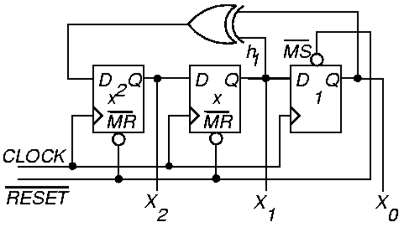
- ***Standard Linear Feedback Shift Register* (LFSR)**
  - **Normally known as *External XOR* type LFSR**
  - **Produces patterns algorithmically – repeatable**
  - **Has most of desirable random # properties**
- **Need not cover all 2^n input combinations**
- **Long sequences needed for good fault coverage**

Standard *n*-Stage LFSR

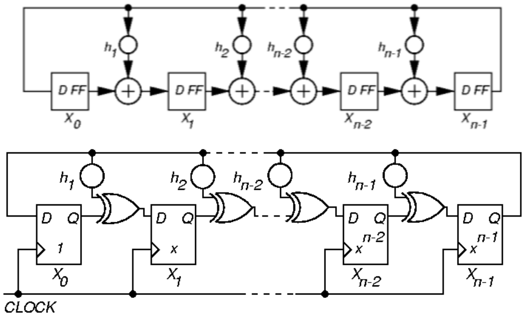


- **If h_i = 0, that XOR gate is deleted**

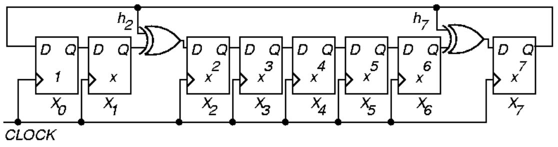
Example External XOR LFSR



Generic Modular (Internal XOR) LFSR



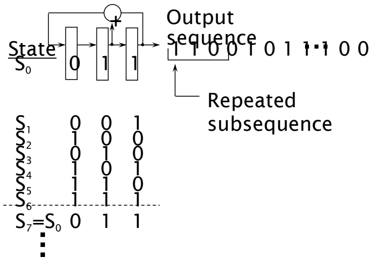
Example Modular LFSR



- ***f*** **(*x*) = 1 + *x*^{2} + *x*^{7} + *x*^{8}**
- **Read LFSR tap coefficients from left to right**

Maximal Length LFSR

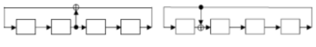
Linear Feedback Shift Register



State			
S_0	0	1	1
S_1	0	0	1
S_2	1	0	0
S_3	0	1	0
S_4	1	0	1
S_5	1	1	0
S_6	1	1	1
S_7=S_0	0	1	1

Generates a cyclic state sequence of length 2^n - 1 (no all zeros case)

4-stage standard and modular LFSRs



(a) A 4-stage standard LFSR (b) A 4-stage modular LFSR

(c) Test sequence generated by (a)	(d) Test sequence generated by (b)
0 0 0 1	**0 0 0 1**
1 0 0 0	1 1 0 0
0 1 0 0	0 1 1 0
1 0 1 0	0 0 1 1
0 1 0 1	1 1 0 1
0 0 1 0	1 0 1 0
0 0 0 1	0 1 0 1
1 0 0 0	1 1 1 0
0 1 0 0	0 1 1 1
1 0 1 0	1 1 1 1
0 1 0 1	1 0 1 1
0 0 1 0	1 0 0 1
0 0 0 1	1 0 0 0
1 0 0 0	0 1 0 0
0 1 0 0	0 0 1 0
1 0 1 0	**0 0 0 1**

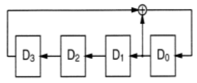
- 4-stage Standard LFSR
  f (x) = 1 + x^2 *+ x*^{4}
- 4-stage Modular LFSR
  f (x) = 1 + x + x^4

LFSR as TPG

- Algorithmically generated test patterns gives good fault coverage, but have some disadvantages.
  - Test pattern generation will be time consuming
  - Storage is needed to hold the test patterns
  - The speed at which tests can be applied is limited
- But LFSR can be clocked at very high rate
- Test patterns generated by LFSR need not be stored since they are reproducible.

LFSR as Exhaustive Sequence Generator

If LFSR is designed as maximum length sequence generator, it will generate all patterns except all zero pattern



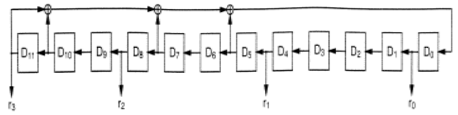
Primitive Polynomial = x^4+x^3+1

D_3	D_2	D_1	D_0
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	1
1	1	1	0
1	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0
0	1	0	0
1	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

LFSR as Random Sequence Generator

- When it is used as the previous case, they are generated in random manner, but they repeat exactly at the intervals of length $2^n – 1$
- In a truly random sequence, vectors repeat at varying intervals and the same vector can appear twice.

LFSR as Random Sequence Generator



r_3	r_2	r_1	r_0
1	0	0	1
0	1	0	0
1	0	1	1
0	1	0	1
1	0	1	0
0	0	0	0
1	1	1	1
0	0	1	0
0	1	0	0
1	0	0	0

⋮

Sequence generated with initial contents 101010100101

Primitive polynomial = $x^{12}+x^6+x^4+x+1$
All zero pattern may also occur

Output Response Analysis (ORA)

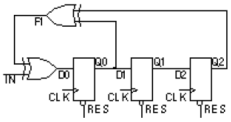
- For BIST operations, it is impossible to store all output responses on-chip, on-board, or in-system to perform bit by-bit comparison. Instead, output responses compacted into a signature and compared with a golden signature
  - Compaction signature: lossy
  - Compression signature: loss-less
  - Error masking: the faulty and fault-free signatures are the same
  - Alias: erroneous output response is said to be an alias of the correct output response
- Three output response compaction techniques
  - Ones count testing
  - Transition count testing
  - Signature analysis

Signature Analysis

- Compresses the stream of inputs to the length of LFSR. The LFSR contents are called Signature of input stream
- Test Patterns for BIST can be generated at-speed by an LFSR with only a clock input
- The outputs of the circuit-under-test must be compared to the known good response
- In general, collecting each output response and off-loading it from the CUT for comparison is too inefficient to be practical
- The general solution is to compress the entire output stream into a single *signature* value
- Signature Analysis is a compression technique based on the concept of *cyclic redundancy checking* (CRC)
  - The simplest form of this technique is based on a single input LFSR

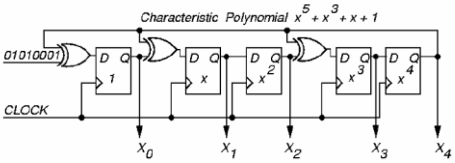
Signature Analysis

- Signature analysis is the most popular compaction technique used today, based on cyclic redundancy checking.

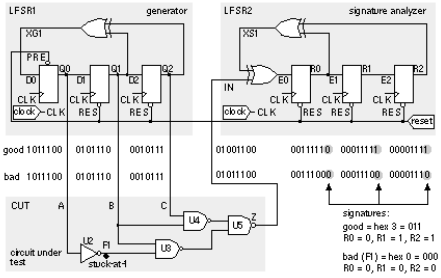


- If we apply a binary input sequence to IN, the shift register will perform **data compaction (or compression)** on the input sequence.
- At the end of the input sequence the shift-register contents, Q0Q1Q2 , will form a pattern that is known as **signature**

Example



A Simple BIST Example



Reference

- M. Bushnell and V. D. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits, Kluwer Academic Publishers, 2000.

- Miron Abramovici, Melvin Breuer, Arthur D Friedman ,Digital Systems Testing and Testable Design –Jaico Publishing House,2005

- Charles H Roth ,Jr , Digital Design using VHDL , Cenage Publishers , India Edition,2006