FAULT MODEL

In today's world where moore's law is applied, IC consists of millions of transistors so It is a very difficult task to analyze where physical defects occur in the chip. These physical defects are very expensive or not easy to analyze because we have many possibilities of physical fault/defect in the design.

- How does a chip fail?

Usually failures are shorts between two conductors or opens in a conductor
This can cause very complicated behavior

- A simpler model: Stuck-At

Assume all failures cause nodes to be "stuck-at" 0 or 1, i.e., shorted to GND or VDD
Not quite true, but works well in practice

Design For Testability

DFT is a method for making a design testable , observable and controllable after it has been manufactured.

- Observability: ease of observing a node by watching external output pins of the chip
- Controllability: ease of forcing a node to 0 or 1 by driving input pins of the chip

It is a technique which only detects that a physical is faulty or is not faulty. It's the extra logic we add to a design throughout the design phase. Post-production testing is required because the manufacturing process is not completely error-free.

To reduce the time and effort needed to generate high-quality structural testing patterns, enhance the circuit during the design process to make the design more testable.If each register could be observed and controlled, the test problem reduces to testing combinational logic between registers. Better yet, logic blocks could enter test mode where they generate test patterns and report the results automatically.

Why DFT?

Post-production testing is necessary because the process of manufacturing is not 100% error free.
There are defects in silicon which contribute towards the errors introduced in the physical device
Only chip is qualified to shipped its customer,if physical verification will be passed

- Manufacturing test ideally would check every node in the circuit to prove it is not stuck.
- Apply the smallest sequence of test vectors necessary to prove each node is not stuck.
- Good observability and controllability reduce the number of test vectors required for manufacturing tests.
    - Reduces the cost of testing
    - Motivates design-for-test

Difference Between Normal D Flip Flop And Scan D Flip Flop

Normal D Flip Flop have two inputs " D,CLK" and one output Q while scan D Flip Flop have four inputs "D,CLK,SI(Scan In), SE(Scan Enable)" and two outputs "Q,SO(Scan Out).

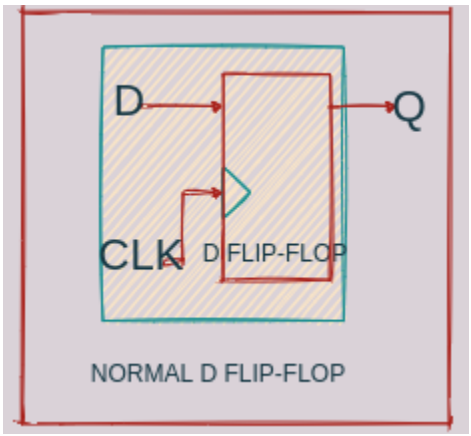

Fig 1: D-Flip Flop

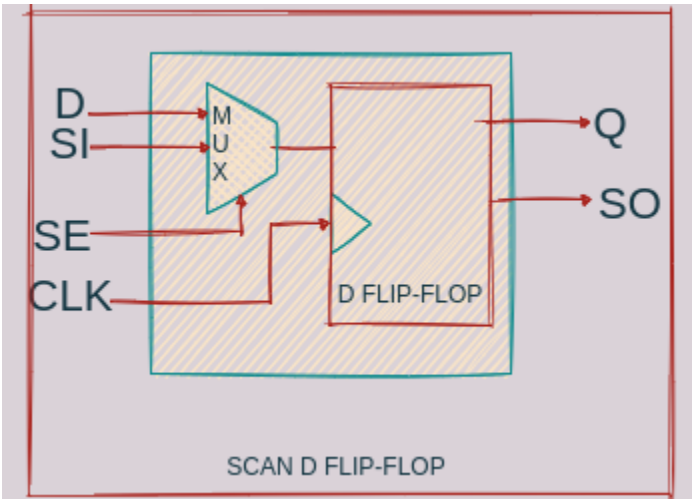

Fig 2: Scan D-Flip Flop

Working of Scan Chain

- Replaces basic-D flops with their Scan D-Flip Flops.
- Serially connecting the scan flops into scan chains
- The scan flip-flops are During scan-shift mode, test data can be shifted through the scan chains. Test data is shifted in through the SI(Scan_In) pins and shifted out through the SO(Scan_Out) pins.

Scan Shift Mode:

Scan Shift Mode on when SE(Shift_Enable) =1 , in which registers act as shift registers in a scan chain. Test vector data is shifted into the scan chain registers, and the captured data from the capture mode are shifted out of the scan registers.The input of the flop is the previous output of the flop in Scan Shift Mode.

System Mode:

System Mode on when SE(Shift_Enable)=0, works as a normal or intended operation of the circuit. Any logic dedicated for DFT purposes is NOT active in the system mode and work as mentioned in Fig 3.

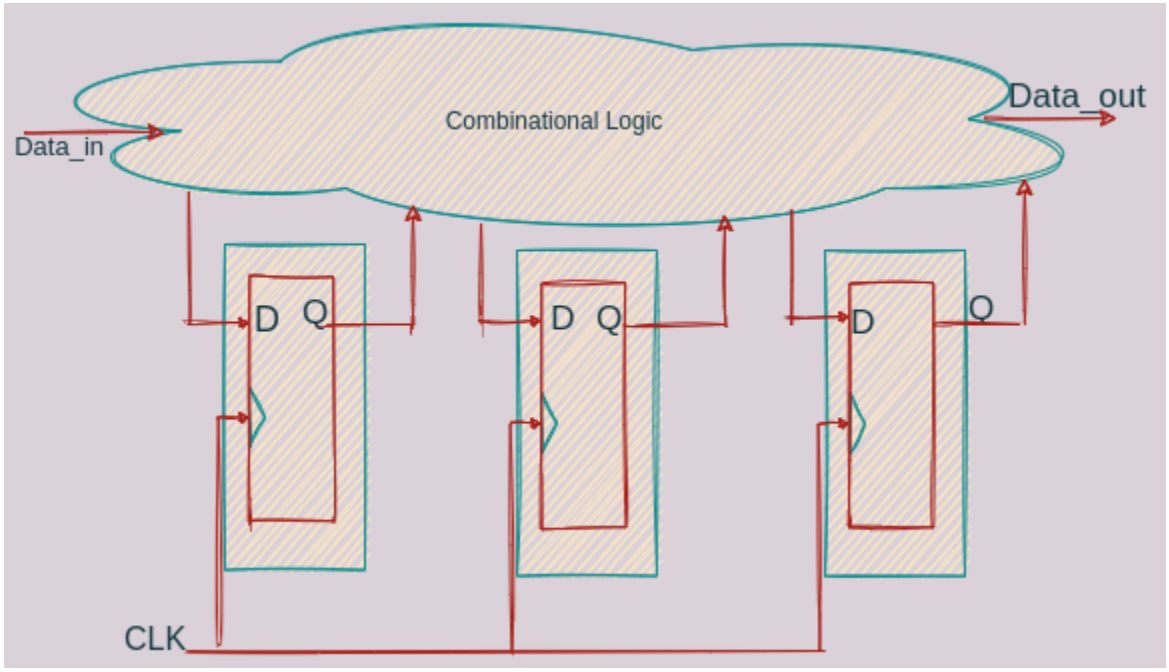

Fig 3: Without Scan Chain Logic diagram

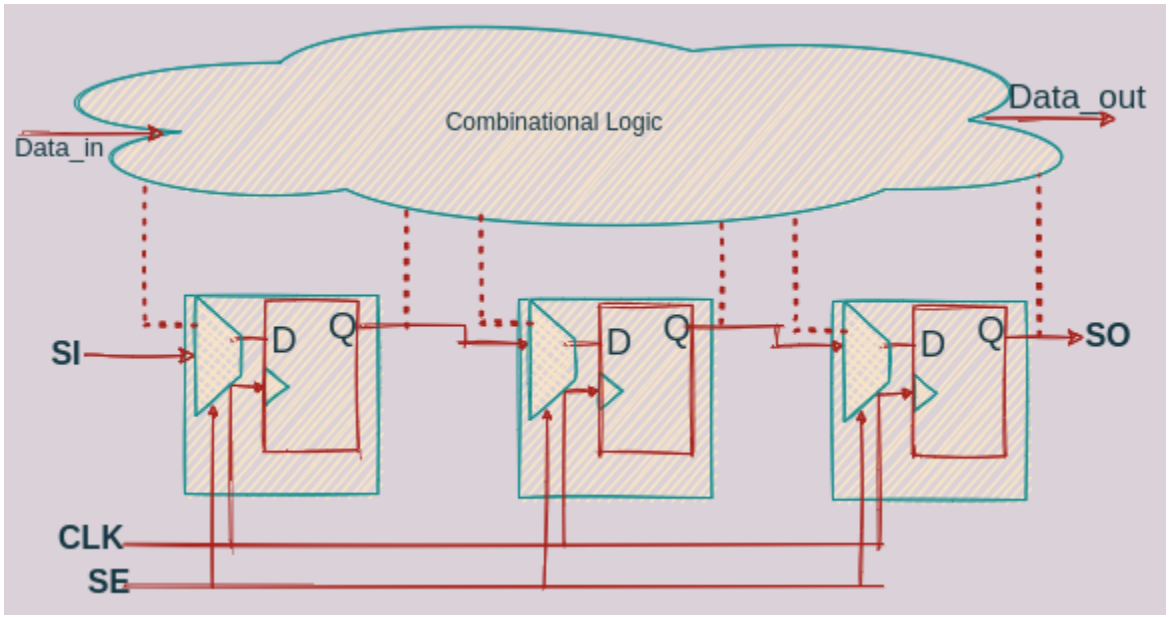

Fig 4: With Scan Chain Logic diagram

Scan Shift Mode (SE=1) System Mode (SE=0)

BOUNDARY SCAN

- Testing boards is also difficult
    - Need to verify solder joints are good
        - Drive a pin to 0 and then to 1
        - Check that all connected pins get the values
- Through-hold boards used “bed of nails”
- SMT and BGA boards cannot easily contact pins
- Build capability of observing and controlling pins into each chip to make board test easier
- Boundary scan is accessed through five pins
    - TCK: test clock
    - TMS: test mode select
    - TDI: test data in
    - TDO: test data out
    - TRST*: test reset

- Chips with internal scan chains can access the chains through boundary scan for unified test strategy.

FAULT

FAULT is an Open Source solution of DFT. Fault operates on RTL designs in Verilog and is made up to three steps

Synthesis:

1. Synthesize Flatten Netlist:

   Synth component used to synthesize flatten netlist.

```
$  fault synth -t s27 -l Tech/osu035/osu035_stdcells.lib Benchmarks/ISCAS_89/s27.v
```

```
Fault on  master [?] via  v5.6.1
 ❯ fault synth --top s27 --liberty ./Tech/osu035/osu035_stdcells.lib ./Benchmarks/ISCAS_
89/s27.v

 /-------------------------------------------------------------------------------\
 |                                                                               |
 |  yosys -- Yosys Open SYnthesis Suite                                          |
 |                                                                               |
 |  Copyright (C) 2012 - 2016  Clifford Wolf <clifford@clifford.at>              |
 |                                                                               |
 |  Permission to use, copy, modify, and/or distribute this software for any     |
 |  purpose with or without fee is hereby granted, provided that the above       |
 |  copyright notice and this permission notice appear in all copies.            |
 |                                                                               |
 |  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES     |
 |  WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF             |
 |  MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR      |
 |  ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES       |
 |  WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN        |
 |  ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF      |
 |  OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.               |
 |                                                                               |
```

Output results show in the netlist folder named as s27.netlist.v.

Scan Chain:

2. Chain component is used to create scan chains through flatten netlist.a boundary scan chain is constructed through a netlist's input and output ports. An internal register chain is also constructed through the netlist's D-flip-flops.

```
$  fault chain -l Tech/osu035/osu035_stdcells.lib -c Tech/osu035/osu035_stdcells.v
--clock CK --reset reset Netlists/s27.netlist.v
```

```
Fault on  master [?] via  v5.6.1
 ❯  fault chain -l Tech/osu035/osu035_stdcells.lib -c Tech/osu035/osu035_stdcells.v --cl
ock CK --reset reset Netlists/s27.netlist.v
Generating LALR tables
WARNING: 183 shift/reduce conflicts
Chaining internal flip-flops…
Internal scan chain successfuly constructed. Length:  3
Creating and chaining boundary flip-flops…
Generating LALR tables
WARNING: 183 shift/reduce conflicts
Boundary scan cells successfuly chained. Length:  7
Total scan-chain length:  10
Resynthesizing with yosys…
Verifying scan chain integrity…
Generating LALR tables
WARNING: 183 shift/reduce conflicts
done
Scan chain verified successfully.
Done.
```

Output result will be:

```
.rw-r--r-- 8.4k root   15 Apr 19:28 s27.netlist.v.chained.v
.rw-r--r--  988 root   15 Apr 19:28 s27.netlist.v.chained.v.bsr.v
.rw-r--r-- 5.5k root   15 Apr 19:28 s27.netlist.v.chained.v.intermediate.v
.rw-r--r-- 1.8k root   15 Apr 19:28 s27.netlist.v.chained.v.tb.sv
```

3. Tap component used to add the JTAG interface to a chained netlist.

```
$  fault tap -l Tech/osu035/osu035_stdcells.lib -c Tech/osu035/osu035_stdcells.v -l Tech/osu035/osu035_stdcells.lib -c Tech/osu035/osu035_stdcells.v --clock CK --reset reset Netlists/s27.netlist.v.chained.v
```

```
Fault on  master [?] via  v5.6.1 took 9s
✦2 ❯ fault tap -l Tech/osu035/osu035_stdcells.lib -c Tech/osu035/osu035_stdcells.v -l Tech/osu035/osu035_stdcells.lib -c Tech/osu035/osu035_stdcells.v --clock CK --reset reset Netlists/s27.netlist.v.chained.v
Generating LALR tables
WARNING: 183 shift/reduce conflicts
Creating top module…
Stitching tap port…
Generating LALR tables
WARNING: 183 shift/reduce conflicts
Generating LALR tables
WARNING: 183 shift/reduce conflicts
Resynthesizing with yosys…
Verifying tap port integrity…
Generating LALR tables
WARNING: 183 shift/reduce conflicts
Done.
Tap port verified successfully.
```

Output will jtag port like this :

```
module s27(GND, VDD, CK, reset, G0, G1, G17, G2, G3, tms, tck, tdi, tdo, trst, tdo_paden_o);
  wire _000_;
  wire _001_;
  wire _002_;
```

Automatic Test Pattern Generation (ATPG):

4. Cutting out the FlipFlops and replacing them with scan equivalent flip flops.

```
Fault on  master [?] via  v5.6.1
✦2 ❯ fault cut Netlists/s27.netlist.v
Generating LALR tables
WARNING: 183 shift/reduce conflicts
```

Now we have two netlist in the Netlist folder

```
    .D(\DFF_0.D ),
    .Q(\DFF_0.Q ),
    .R(_07_),
    .S(1'b1)
  );
  DFFSR _25_ (
    .CLK(CK),
    .D(\DFF_1.D ),
    .Q(\DFF_1.Q ),
    .R(_08_),
    .S(1'b1)
  );
  DFFSR _26_ (
    .CLK(CK),
    .D(\DFF_2.D ),
    .Q(\DFF_2.Q ),
    .R(_00_),
    .S(1'b1)
  );
endmodule
```

Synthesize netlist

```
  INVX1
  _22_
  (
    .A(reset),
    .Y(_07_)
  );


  INVX1
  _23_
  (
    .A(reset),
    .Y(_08_)
  );

  assign \DFF_0.Q  = _24_;
  assign \_24_.q  = \DFF_0.D ;
  assign \DFF_1.Q  = _25_;
  assign \_25_.q  = \DFF_1.D ;
  assign \DFF_2.Q  = _26_;
  assign \_26_.q  = \DFF_2.D ;

endmodule
```

cut flip flop

Test Vector Generation:

5. By using RNG Random Number Generator test vector are generated.It also show the coverage meet.

```
$  fault -c Tech/osu035/osu035_stdcells.v -v 100 -r 50 -m 95 --ceiling 1000
Netlists/s27.netlist.v.cut.v
```

```
Fault on  master [?] via  v5.6.1
✦2 ❯ fault -c Tech/osu035/osu035_stdcells.v -v 100 -r 50 -m 95 --ceiling 1000 Netlists/s27.netlist.v.cut.v --clock CK
Generating LALR tables
WARNING: 183 shift/reduce conflicts
Processing module s27…
Found 51 fault sites in 14 gates and 15 ports.
Performing simulations…
Skipped 4 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 150…
Skipped 6 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 200…
Skipped 6 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 250…
Skipped 9 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 300…
Skipped 16 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 350…
Skipped 9 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 400…
Skipped 13 duplicate generated test vectors.
```

```
Skipped 29 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 700…
Skipped 32 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 750…
Skipped 26 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 800…
Skipped 21 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 850…
Skipped 32 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 900…
Skipped 25 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 950…
Skipped 27 duplicate generated test vectors.
Minimum coverage not met (82.35294%/95.0%,) incrementing to 1000…
Skipped 31 duplicate generated test vectors.
Hit ceiling. Settling for current coverage.
Time elapsed: 16.66s.
Simulations concluded: Coverage 82.35294%
```

Json and svf file are created

Run Fault simulation:

6. You could also run fault simulations for an input test vector set provided in json format.The simulations could also be constrained by specifying the ceiling, increment in TV count, and minimum coverage:

```
$ fault --ceiling 100 -r 10 -v 10 -m 90 --tvSet Netlists/s27.netlist.v.cut.v.tv.json Netlists/s27.netlist.v.cut.v --cellModel Tech/osu035/osu035_stdcells.v --clock CK
```

```
Fault on  master [?] via  v5.6.1 took 3s
✦2 ❯ fault --ceiling 100 -r 10 -v 10 -m 90 --tvSet Netlists/s27.netlist.v.cut.v.tv.json Ne
tlists/s27.netlist.v.cut.v --cellModel Tech/osu035/osu035_stdcells.v --clock CK
Generating LALR tables
WARNING: 183 shift/reduce conflicts
Processing module s27…
Read 647 vectors.
Found 51 fault sites in 14 gates and 15 ports.
Performing simulations…
Minimum coverage not met (82.35294%/90.0%,) incrementing to 20…
Minimum coverage not met (82.35294%/90.0%,) incrementing to 30…
Minimum coverage not met (82.35294%/90.0%,) incrementing to 40…
Minimum coverage not met (82.35294%/90.0%,) incrementing to 50…
Minimum coverage not met (82.35294%/90.0%,) incrementing to 60…
Minimum coverage not met (82.35294%/90.0%,) incrementing to 70…
Minimum coverage not met (82.35294%/90.0%,) incrementing to 80…
Minimum coverage not met (82.35294%/90.0%,) incrementing to 90…
Minimum coverage not met (82.35294%/90.0%,) incrementing to 100…
Hit ceiling. Settling for current coverage.
```

Compact:

7. You can also compact the previous Test Generation Vector.

```
$ fault compact Netlists/s27.netlist.v.cut.v.tv.json
```

```
Fault on  master [?] via  v5.6.1
✦2 ❯ fault compact Netlists/s27.netlist.v.cut.v.tv.json
Finding essential test vectors…
Found 0 essential test vectors.
Performing compaction…
Initial TV Count: 647. Compacted TV Count: 6.
Compaction is successfuly concluded with a reduction percentage of : 99.07% .
```

Post-silicon validation plan

Silicon validation is testing a chip once manufactured for functional correctness. This is done by building a real system in a lab set up to validate all features, compliance and to even run real software on the silicon before shipping to customers.

In scan shift mode we will observe the output or any fault in the device.Scan Shift Mode on when SE(Shift_Enable) =1 , in which registers act as shift registers in a scan chain. Test vector data is shifted into the scan chain registers, and the captured data from the capture mode are shifted out of the scan registers.The input of the flop is the previous output of the flop in Scan Shift Mode and with JTAG port we will analyze the output vector

For example, tested by shifting in the pattern 011111…111 in test mode, then shifting out the result, which should be 100000…000.