DFT

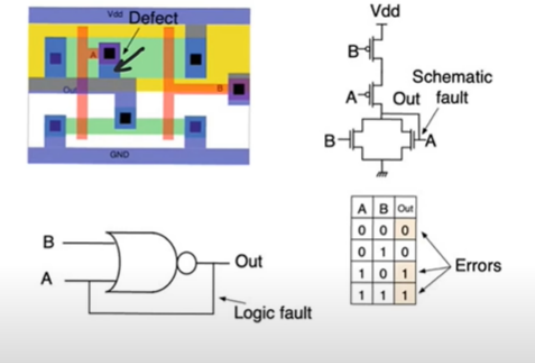
Abanoub Emad Hanna

Introduction

Design for test is the act of adding additional hardware to ensure that the chip is working correctly after manufacturing. The challenge comes from absence of direct controllability and observability of the internal nodes of the chip. The nearer the gate to the primary input, the easier it is to control but difficult to observe, and vice versa.

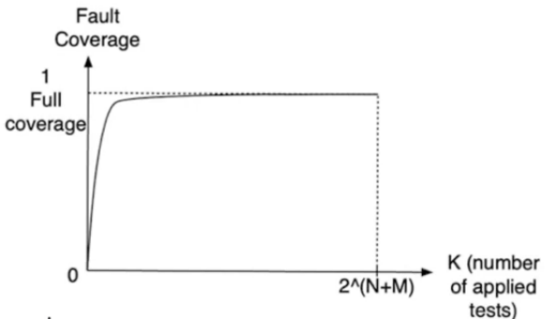
Definitions

- **Testing:** Done on every finished chip.
- **Verification:**: Done on a design before fabrication.
- **Defect:** A physical problem that happened during fabrication.
- **Fault:** Modeling the **effect** (behaviour) of a defect (not necessarily the actual defect) at a certain level of abstraction.
- **Error:** Wrong output due to a fault.



Fault Coverage

- If we have M-bit input and N flip-flops (states), number of tests = 2^{M+N}, which is impractical.
- Fault Coverage is the ratio between detected faults and total faults ($F = \frac{\text{detected faults}}{\text{total faults}}$).
- Fault coverage increases rapidly for a small number of tests, and then it starts to saturate slowly.
- Therefore, we carry out this small number of tests and tolerate the rest of the faults.

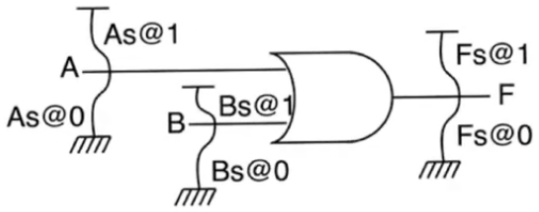


Logic Testing

Stuck-at Faults

- This models faults at the gate level.
- Any node could be either stuck-at-0 or stuck-at-1, so we have 2 possible faults per node. Total possible faults = $2 * N_{\text{gate nodes}}$.
- We assume that only one fault occurs at a time (most likely scenario).
- We fill a truth table assuming that the fault is at a certain node, and then we compare it with the original thruth table to generate a test vector.

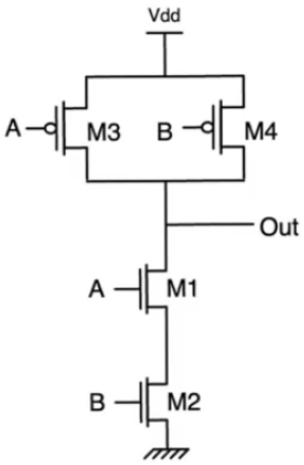
- Test vector is the smallest number of inputs that need to be applied to detect all possible faults.
- T={00,10,01} for the following example.



A	B	F	Fs@0	Fs@1	As@0	As@1	Bs@0	Bs@1
0	0	0	0	*1*	0	*1*	0	*1*
0	1	1	*0*	1	1	1	*0*	1
1	0	1	*0*	1	*0*	1	1	1
1	1	1	*0*	1	1	1	1	1

Stuck-Open/Short Faults

- This models faults at the transistor level.
- Any transistor could be either stuck-open or stuck-short, so we have 2 possible faults per transistor. Total possible faults $= 2 * N_{\text{transistors}}$.
- We assume that only one fault occurs at a time (most likely scenario).
- In order to detect a high impedance output (Z), we need to apply 2 inputs in sequence.
- For example, for M1s0, we have to apply input to drive the output to V_{DD} and then apply the input causing the high impedance. If output remains high, then the fault is detected.



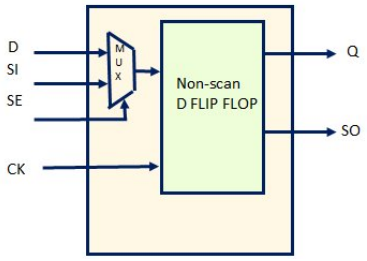
AB	Out	M1sO	M1sS	M2sO	M2sS	M3sO	M3sS	M4sO	M4sS
"00	Vdd	Vdd	Vdd	Vdd	Vdd	Vdd	Vdd	Vdd	Vdd
"01	Vdd	Vdd	Isc	Vdd	Vdd	Z	Vdd	Vdd	Vdd
"10	Vdd	Vdd	Vdd	Vdd	Isc	Vdd	Vdd	Z	Vdd
"11	0	Z	0	Z	0	0	Isc	0	Isc

Scan Registers

- Replace every single pipeline register on the design with a scan register.
- SE: Scan Enable, this is the control signal that represents mode of operation (normal/scan).
- SI: Scan In, first register in the chain gets its input from an external pin. Other registers get their input from the previous register's SO: Scan Out.
- Last register's SO is connected to external pin.
- Note that the SO connection in the scan FF is actualy Q.

Effects

- **Timing Overhead:**
  - Higher setup time due to mux delay (try KVL method also).
  - Higher clock-to-Q delay due to additional load from next scan FF.
- **Area Overhead:** Additional area of the muxes.



Operation

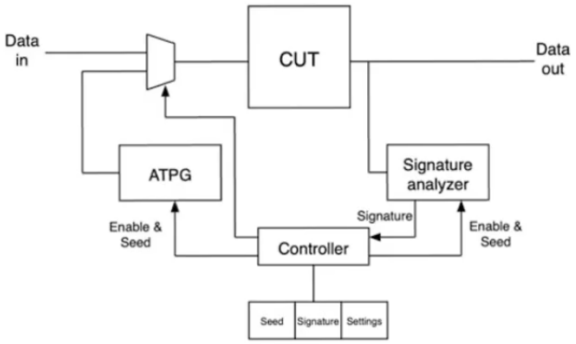
1. **Shift In:**
   - SE is asserted, so that the scan FFs are connected like a shift register.
   - We start shifting the test vector to store desired test case in the desired FF.
2. **Capture:**
   - SE is de-asserted for 1 clock cycle so that the combinational logic provide an output.
3. **Shift Out:**
   - SE is asserted again, and the value stored in the capture flop is shifted to the last FF to be obsereved at output pin.

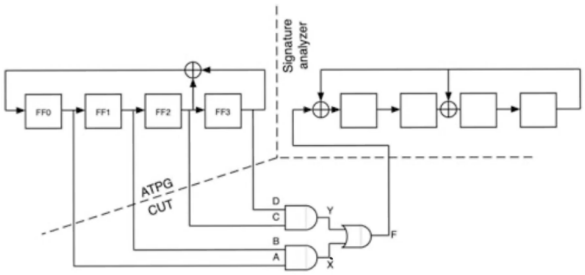
Note: The disadvantage here is the deeper the pipeline, the higher the latency as you have to wait for input to be shifted in and the output to be shifted out.

Guidelines

- All the scan inputs in the scan FF should be controlled by the ATPG tool.
- All primary input clocks, generated clocks (e.g clock dividers) and resets should be muxed with a scan clock and scan reset respectively (scan mode is usually run on a slower clock to avoid any timing violations). MUX selector is test mode signal.
- Gated clocks are not controllable and should be bypassed in test mode.
- Do not replace shift registers with scan registers, just ensure the enable control.

Built-In Self-Test (BIST)





Now we consider a setup where the test is performed within the chip itself (BIST); one of the most well known forms of BIST is Power on Self Test (POST).

- When the test signal is active, the input to the CUT (Circuit Under Test) is from the ATPG, and its output is given to a signature analyzer.
- Signature analyzer takes the outputs and automatically checks if the outputs fit with the correct outputs or not.

ATPG

- As discussed in fault coverage it's hard to cover all faults, so we should apply tests that combine the input bits in a random fashion so that it exposes as many faults as possible.
- On the other hand, the pattern applied cannot be truly random because we need to know the pattern to generate the gold standard against to compare the output.
- The sequence is pseudo-random noise (PN) sequence which is generated using linear feedback shift register (LSFR).
- The pattern of the LFSR has no input and generates a periodic sequence every $2^m - 1$ cycles based on an initial seed.
- The pattern itself can be concluded by knowing the initial pattern and the number of applied clock signal, but the bit change is random.

ABCD	F	As@0	Xs@1	Xs@0	Cs@1	Ys@0	Fs@0	Bs@1
"1000"	0	0	*1*	0	0	0	0	*1*
"0100"	0	0	*1*	0	0	0	0	0
"0010"	0	0	*1*	0	0	0	0	0
"1001"	0	0	*1*	0	*1*	0	0	*1*
"1100"	1	*0*	1	*0*	1	1	*0*	1
"0110"	0	0	*1*	0	0	0	0	0
"1011"	1	1	1	1	1	*0*	*0*	1
"0101"	0	0	*1*	0	*1*	0	0	0
"1010"	0	0	*1*	0	0	0	0	*1*
"1101"	1	*0*	1	*0*	1	1	*0*	1
"1110"	1	*0*	1	*0*	1	1	*0*	1
"1111"	1	1	1	1	1	1	*0*	1
"0111"	1	1	1	1	1	*0*	*0*	1
"0011"	1	1	1	1	1	*0*	*0*	1
"0001"	0	0	*1*	0	*1*	0	0	0
"1000"	0	0	*1*	0	0	0	0	*1*

Signature Analyzer

- Testing all possible faults would require 2^m bit subtractors, which is impractical.
- The signature anayzer is similar to LFSR but has an external input and can have linear operations (XORs) between the registers of the shift register.
- The analyzer compresses the output sequence of CUT (2^m) into an n-bit signature that can specify the fault occured.
- Sequence of operation:
  1. Initial seed is loaded to the registers (from designer).
  2. The output of the CUT (F) is fed to the signature analyzer.
  3. At the end of 2^m cycles, the contents of the register is the signature.
  4. Compare the signature with the golden signature (1000 in this example).
- However, the compressed signature may not be unique, so there is a trade-off between the number of bits (size of analyzer) and the uniqueness of the signature.

	F	As@0	Xs@1	Xs@0	Cs@1	Ys@0	Fs@0
Seed	"0001"	"0001"	"0001"	"0001"	"0001"	"0001"	"0001"
1	"1010"	"1010"	"0010"	"1010"	"1010"	"1010"	"1010"
2	"0101"	"0101"	"1001"	"0101"	"0101"	"0101"	"0101"
3	"1000"	"1000"	"0110"	"1000"	"1000"	"1000"	"1000"
4	"0100"	"0100"	"1011"	"0100"	"1100"	"0100"	"0100"
5	"1010"	"0010"	"0111"	"0010"	"1110"	"1010"	"0010"
6	"0101"	"0001"	"0001"	"0001"	"0111"	"0101"	"0001"
7	"0000"	"0010"	"0010"	"0010"	"0001"	"1000"	"1010"
8	"0000"	"0001"	"1001"	"0001"	"0010"	"0100"	"0101"
9	"0000"	"1010"	"0110"	"1010"	"0001"	"0010"	"1000"
10	"1000"	"0101"	"1011"	"0101"	"0010"	"1001"	"0100"
11	"1100"	"1000"	"0111"	"1000"	"1001"	"0110"	"0010"
12	"1110"	"1100"	"0001"	"1100"	"0110"	"1011"	"0001"
13	"1111"	"1110"	"0010"	"1110"	"1011"	"1111"	"1010"
14	"0101"	"1111"	"1001"	"1111"	"0111"	"1101"	"0101"
15	"1000"	"1101"	"0110"	"1101"	"0001"	"1100"	"1000"
Signature	"1000"	"1101"	"0110"	"1101"	"0001"	"1100"	"0100"

Memory Testing

Memory March is a testing approach where we test a specific cell in the memory and we march to the next address until we reach the end of the memory. The high density of memories makes each cell affect the other and can't be tested in isolation.

Stuck-at Faults

- A zero is applied and read, then a one is applied and read.
- If the output is not the same as the input, then the cell is stuck at 0 or 1.

Transition Faults

- The cell can store a 0 or a 1, but it has a fault in the transition from 0 to 1 or from 1 to 0.
- **1-to-0 Transition:** Write 1, write 0, read 0.
- **0-to-1 Transition:** Write 0, write 1, read 1.

Coupling Faults

Faults that occur due to the high density of the memory cells, where the fault involves two cells.

Briding Faults

- Two cells are bridged (shorted) to each other (bidirectional).
- If you change one the other changes too.

Inversion Faults

- Directional fault (the effect goes only in one direction); one cell has an impact on the other but not vice versa.
- a transition in one cell causes the other cell to invert its stored value.

Idempotent Faults

- It is a directional fault, where a transition in one cell sets a value in the other cell regardless of its previous value.

Coupling Faults Testing

Idempotent fault (0 to 1 A transition sets B to 0) example:

- Initialize A to 0 and B to 1.
- Write 1 to A and read B.
- If 1 is read, then this specific fault isn't present.
- If 0 is read, then we have to check if the fault is idempotent or inversion.
- Write 0 to both A and B.
- Write 1 to A and read B.
- If 0 is read, then this specific idempotent fault is present.
- If 1 is read, then this is an inversion fault.

NPSF (Neighbor Pattern Specific Fault)

It is the most common fault and most difficult to detect. The fault here involves all neighboring cells (8 cells in 2D) affecting the central cell to have any fault from the stated before.

Passive

The presence of a specific pattern in all 8 neighboring cells, causes an erroneous behavior in the central cell.

Active

Occurs when a pattern in the neighboring cells and a transition in one of them causes erroneous behavior in the central cell.