ECE/CS 5710/6710 Digital VLSI Design

Guest Lecture

Introduction to VLSI Testing

Prof. Priyank Kalla

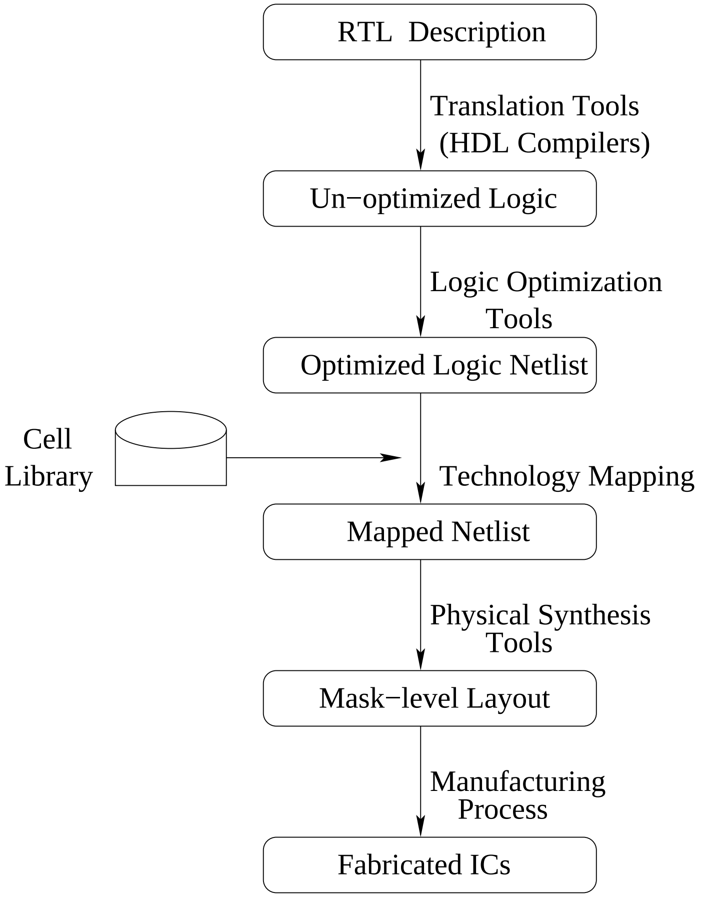
Agenda for Today

- Introduction to VLSI Testing
- Why bother? Or who cares?
  - Not just Pentium Bug!
- Design versus Testing
- Testing versus Verification
- Testing Problems: where to they come from?
- Test Apps: gate-level, ckt-level, comb./seq., memory...
- Design-for-Testability, Synthesis-for-Testability

VLSI Circuit Realization Process

- Customer's requirements
- Design Conceptualization by customers and designers
- Design Specifications (usually in English)
- Design Descriptions - VHDL, Verilog, SystemC, . . .
- Circuit realization: Synthesis, Custom Design
- Marketing.....
- Where do Test and Verification fit-in?
- **Validation v/s Verification v/s Testing**

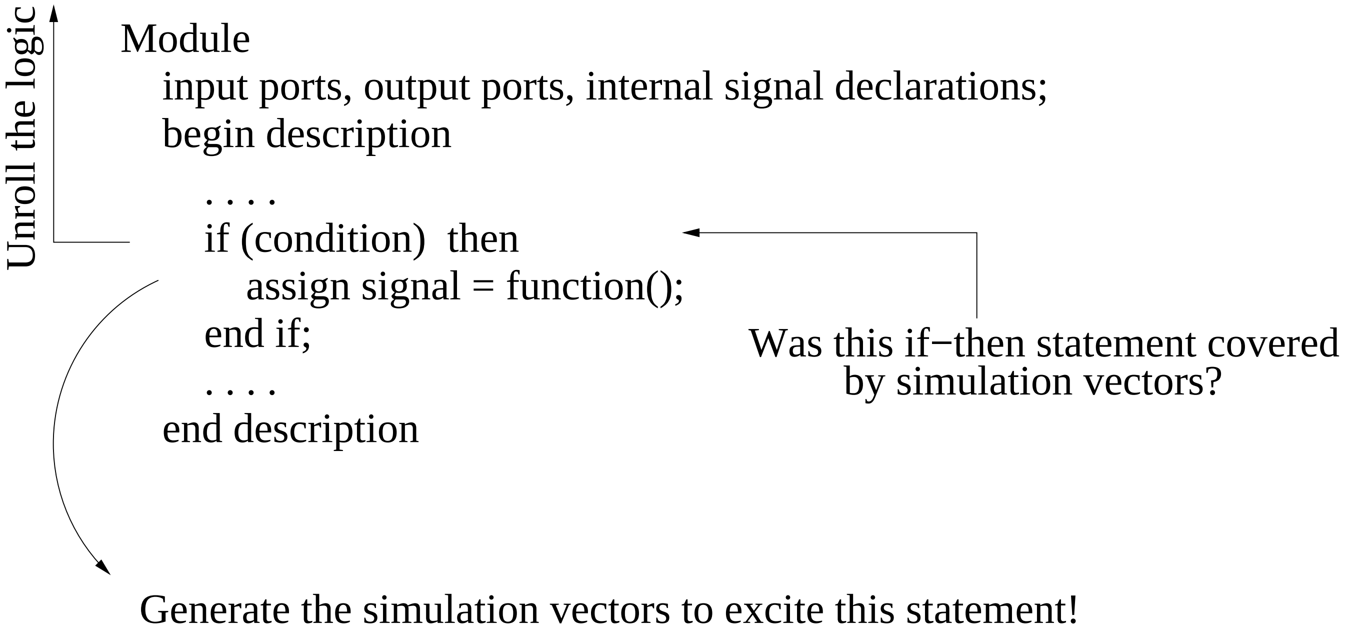
VLSI Circuit Realization Process



Design Validation

- Is the **specification** fool-proof?
  - Bugs: mis-understanding, incorrect modeling, oversight...
  - How to ascertain validity of the specs?
  - Simulate (randomly?) and observe ... too slow
  - Mathematically infer validity..... computationally infeasible
  - Not everyone knows/likes “math”
  - Today’s solutions: little-bit of math + little-bit of simulation + little-bit of luck!
- Topics on design validation: Simulation, coverage metrics, property verification, model checking, among others..

Enhancing the Power of Simulation



Unroll the logic

```
Module
   input ports, output ports, internal signal declarations;
   begin description
      . . . .
      if (condition)  then
         assign signal = function();
      end if;
      . . . .
   end description
```

Was this if−then statement covered by simulation vectors?

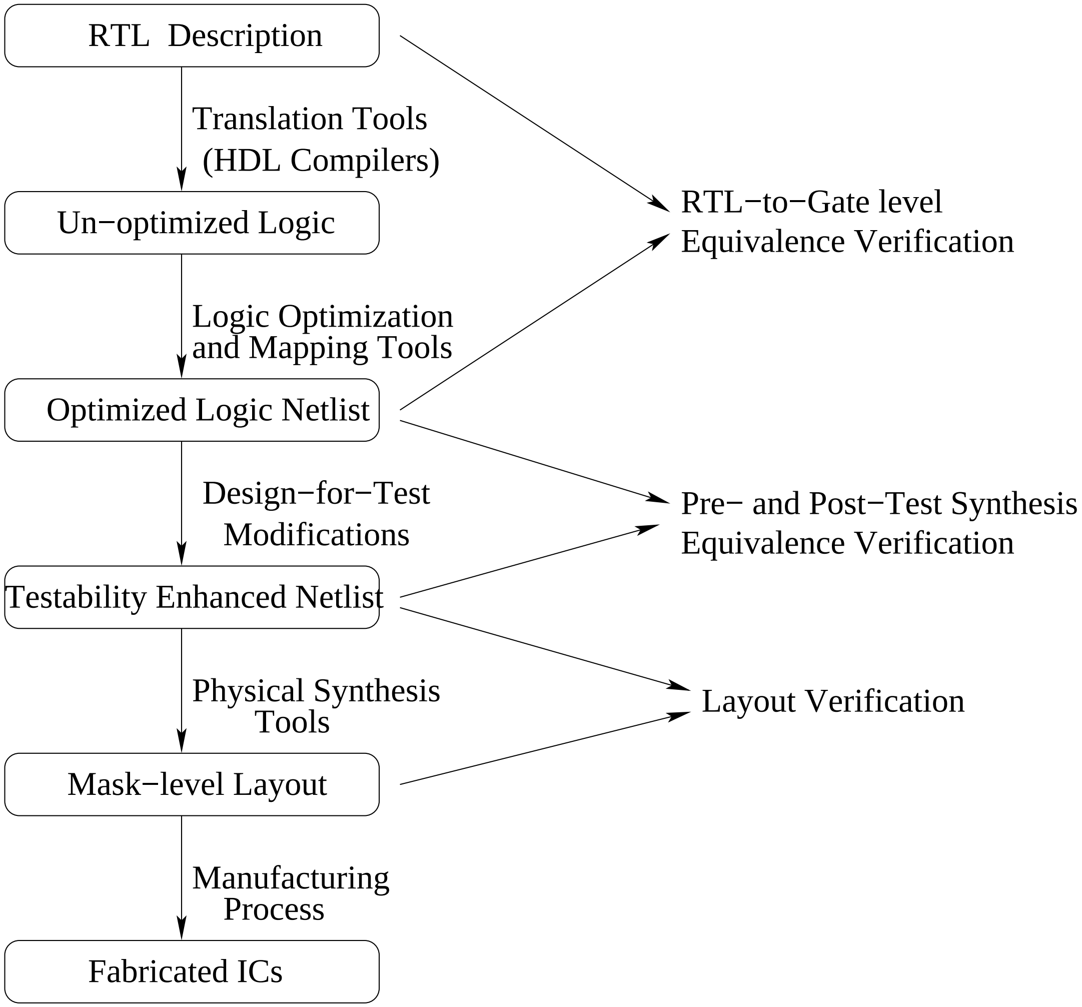
Generate the simulation vectors to excite this statement!

Figure 1: Enhancing RTL code coverage.

Formal Design Verification

- Simulation is incomplete - Remember Pentium Bug?
  - Mathematicians say: "Mathematically prove that your system is fool-proof!"
  - Theorem Proving and Model Checking.....
  - Problem: Computationally infeasible, difficult to automate
  - While mathematicians argue, industry simulates.....
  - Proof of equivalence is (somewhat) manageable
  - Implementation Verification via Equivalence Checking.

Verification: Equivalence Checking



Equivalence Checking Examples

```
module verify(a, b, c, f, y, X, Z, F)
input a, b, c; // Boolean
input [3:0] X; // Bit-vectors
output f, y;
output [3:0] F, Z;

assign f = a’bc + ab’c + abc’ + abc;
assign y = ab + ac + bc; // a(b+c) + bc

assign F[3:0] = 8 * X[3:0] * X[3:0] + 8* X[3:0];
assign Z[3:0] = 4’b0000;
```

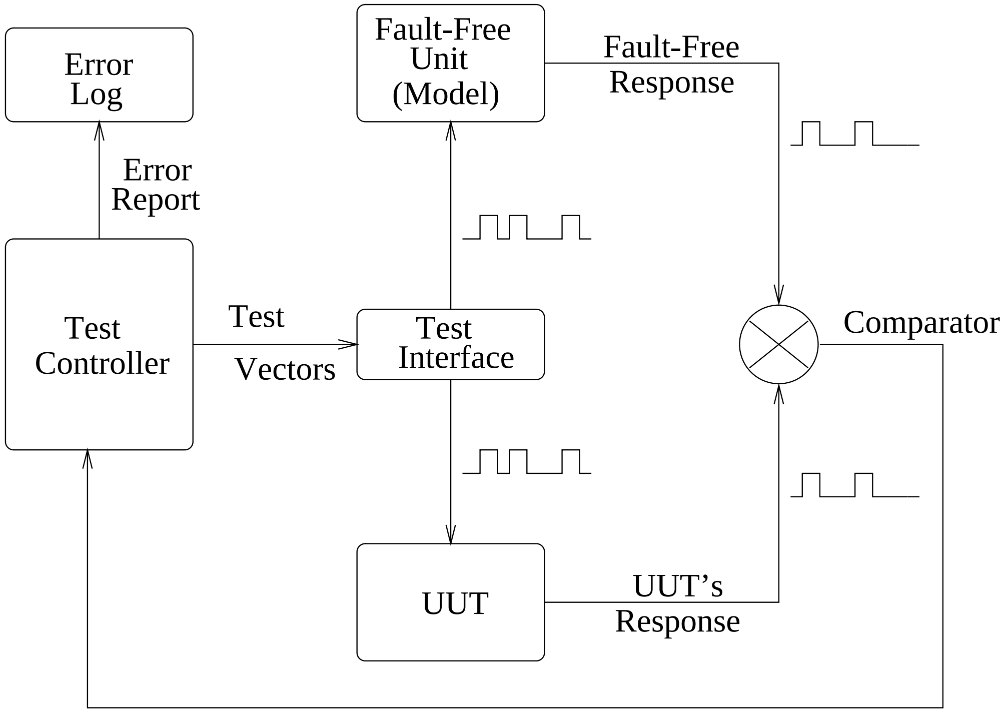
VLSI Testing

- Testing a fabricated chip for manufacturing defects.
- Physical Failures and Fabrication Defects due to:
  - Shorts or opens on wires.... unreliable metalization
  - Electromigration over a period of time
  - Shorts (fusing) of Source-Drain
  - Improper ion-implantation... slow switching
  - Ever heard of latch-up?
  - Reliability of a manufacturing process in general

VLSI Testing

- What is *testing*?
  - Examine a product
  - Ensure correct operation
  - Exhibits the properties that it was designed for
- **Detect** malfunction and incorrect behaviour
- What if malfunction exists? Diagnose it.... (not testing)
- In VLSI domain, testing has a lot of secondary applications
  - Reliability measurement of both, the product and the process
  - Quality assurance
  - Can help in verification + validation

Automatic Test Equipment & Test Environment



What to Test for and How?

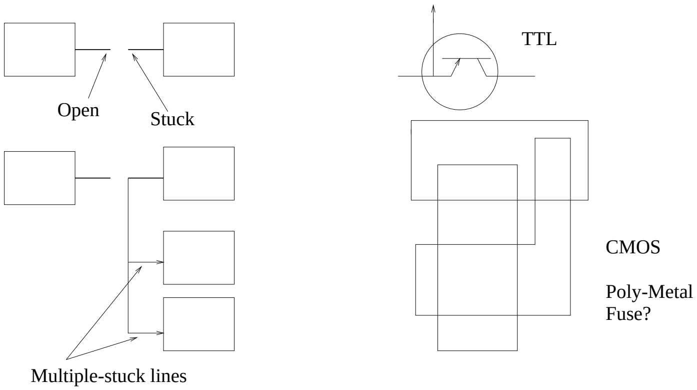
- Defects, Errors and Faults....
- Design Errors, fabrication defects, physical defects
- Design errors: Specification? Logical? Design rule violation?
- Fabrication defects: shorts, opens, incorrect wiring, improper ion-implantation, poor oxidation...
- Physical failures: Electromigration, latch-up, wear-out due to heat/humidity/vibrations...
- Types of Testing:
  - Functional Testing (Validation) and Verification
  - Characterization tests on first-pass Silicon
  - Production Tests: acceptance and rejection of product
  - Burn-in tests: “Burn it at high speed”; detect freak-errors

Logical Modeling of Physical Failures

- Physical failures → anything can fail anywhere
- “Impossible” to detect physical failures on SoC/SoS
- Model physical failures as logical faults (abstract unnecessary info)
- Logical fault model defined w.r.t. circuit structure (netlists)
- Requirement: Model should be versatile!
- Examples: Stuck-at model, bridging model, delay fault model.... (no model at all?)
- Other defects: Delay, Transistor (stuck-open), universal, functional, bridging....
- Overall Strategy: One-by-one, Test the chip w.r.t. each fault model

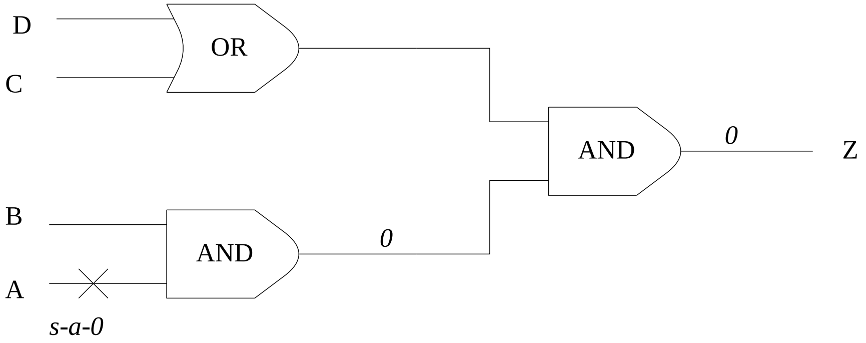
“Stuck-at” Fault Model

- Borrowed from 1960’s technology, hence appealing!
- Wire w short to ground: s-a-0 or power: s-a-1
- Shorts and opens in CMOS v/s TTL (floating v/s pull-up)
- Some s-a-0/1 can cover bridging, transistor or other faults
- Multiple versus Single Stuck-at faults



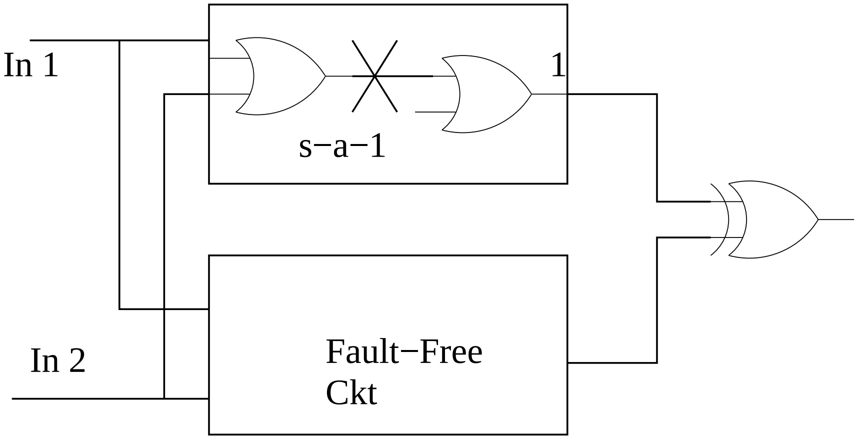
Test Generation Example

- Assume A s-a-0 (or A/0). Apply $\langle A = 0, B = 0, C = 0, D = 0 \rangle$. Z = 0
- Assume fault-free. Apply $\langle A = 0, B = 0, C = 0, D = 0 \rangle$. Z = 0
- Assume A s-a-0. Apply $\langle A = 1, B = 1, C = 1, D = 1 \rangle$. Z = 0
- Assume fault-free. Apply $\langle A = 1, B = 1, C = 1, D = 1 \rangle$. Z = 1



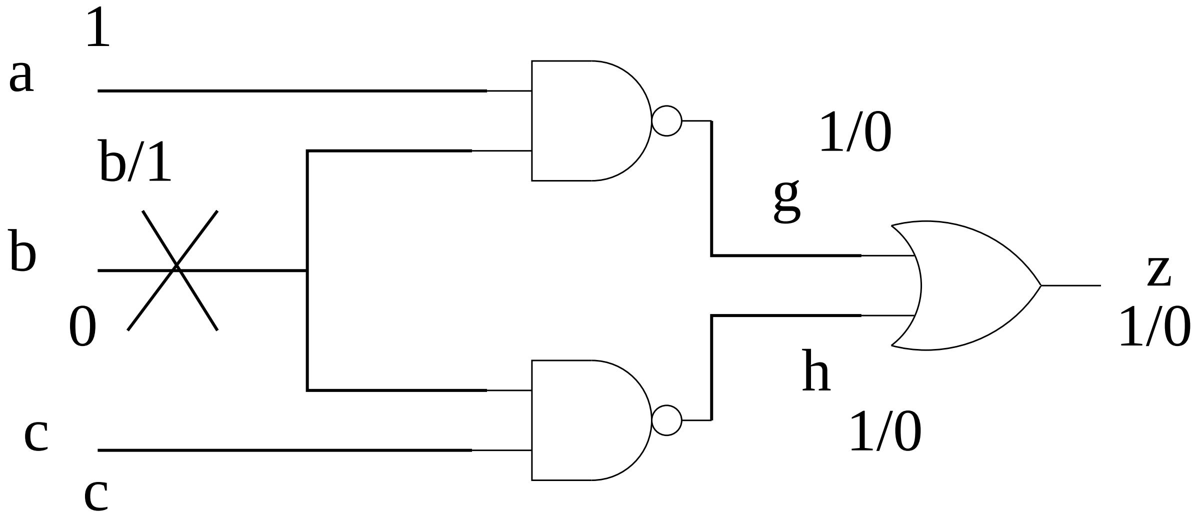
Test Generation v/s Equivalence Check

- Find "Input values" → XOR output = **1**
- If successful → test found
- If not? Complicated...



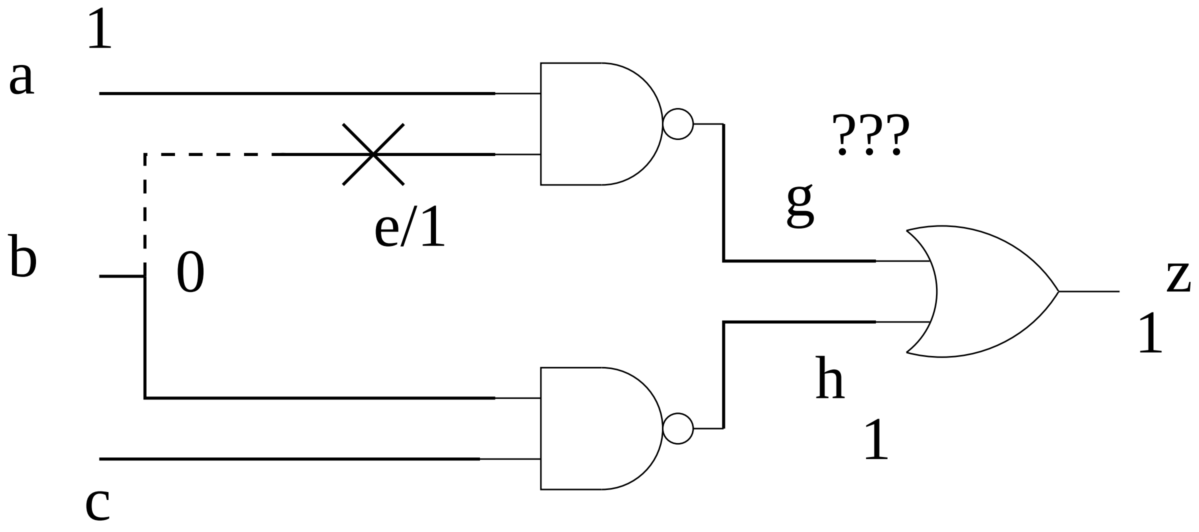
Issues in Test Generation

- Test for **b/1**
- a = 1, b = 0, c = 1

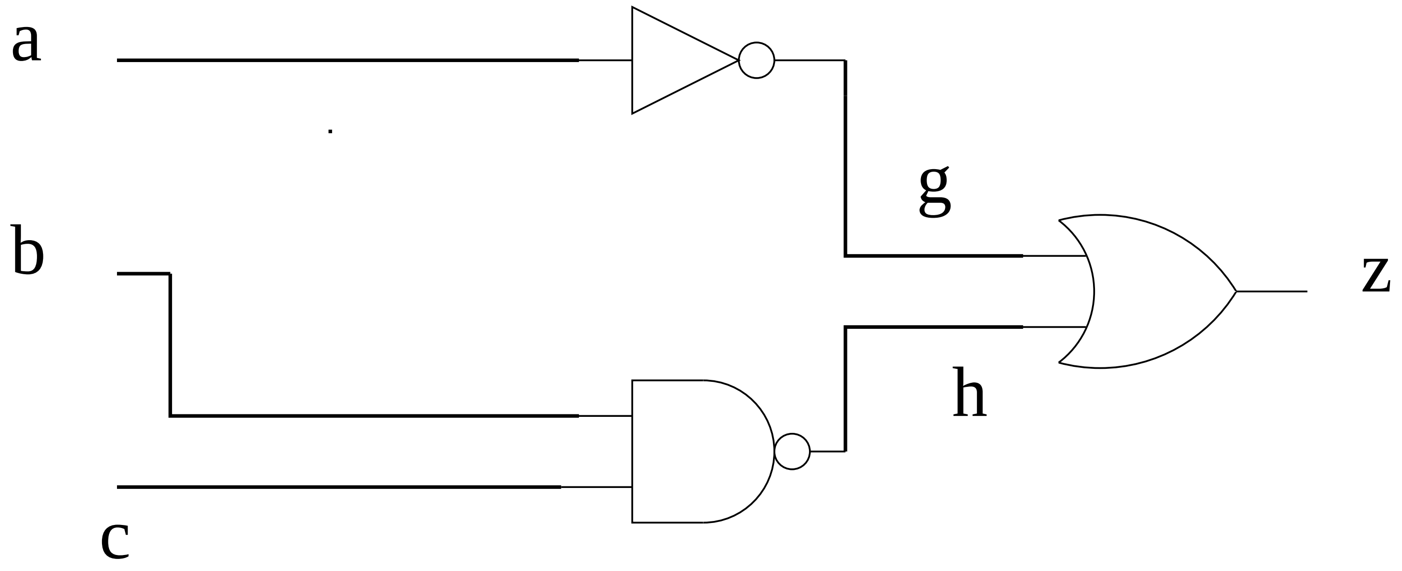


Untestable Fault

- Test for **e/1**: No Test...
- Faulty Circuit = Fault-Free Circuit!



Redundancy Creates Testing Problems

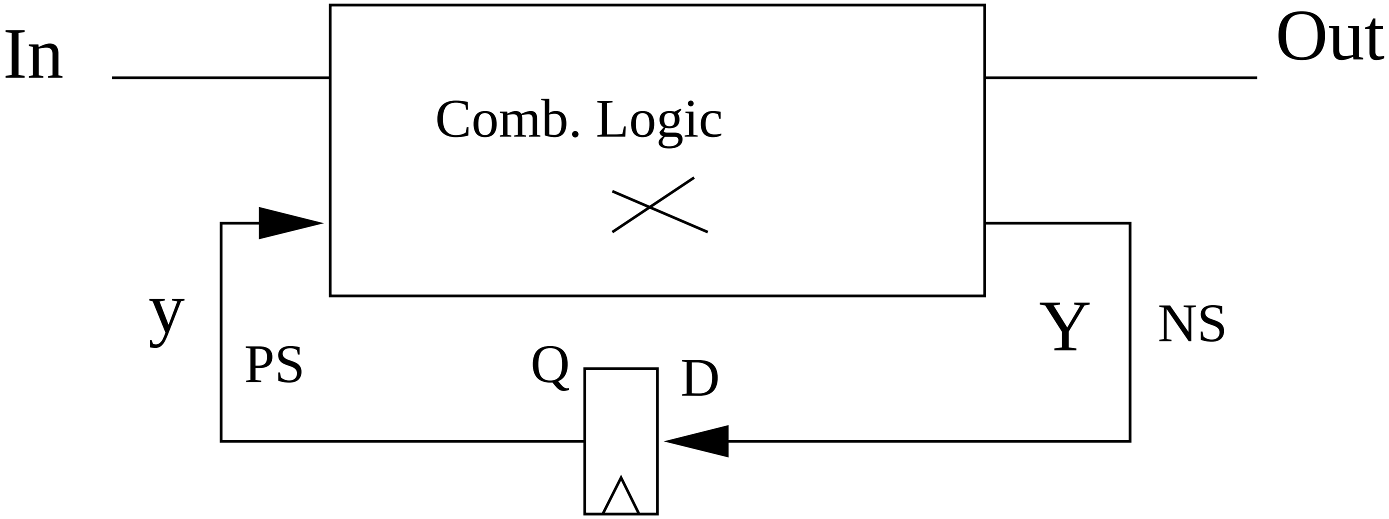


z = !a + !b + !b + !c = !a + !b + !c

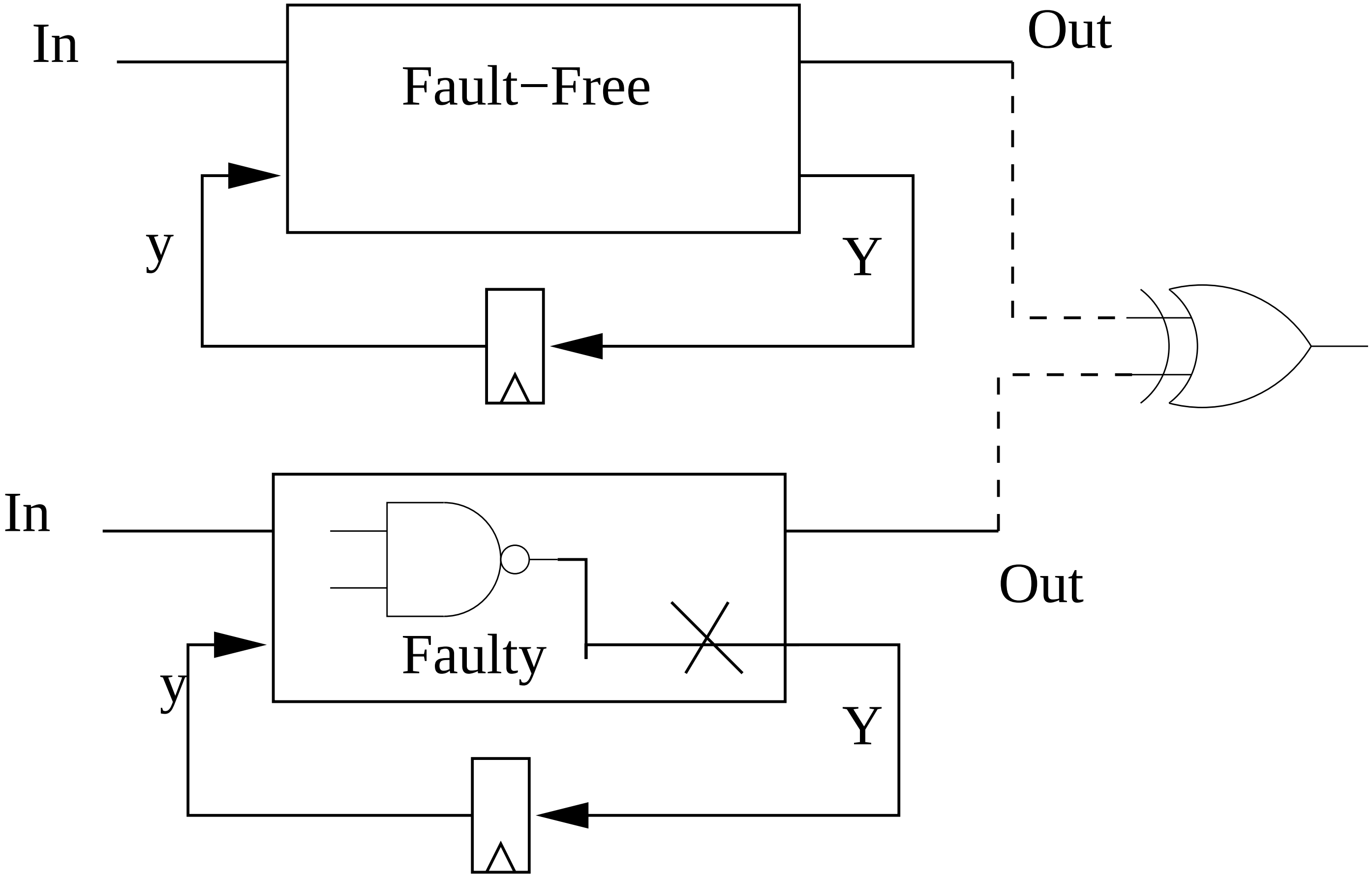
How to test for Single Stuck-Faults

- Any and all lines can be faulty....
- Using the "computer" representation of your circuit:
- Pick a line, assume it has s-a-0 fault, derive a test that detects that fault
- Take the same line, assume it has s-a-1 fault, derive a test for it
- One by one, go through each line, derive both tests for it
- Collect a **set** of "test vectors" and apply them as test stimulus, using ATE, to the fabricated chip.
- n lines in a ckt. → $3^n - 1$ faults. Test for everything?
- Issues: Testing Cost, Testing Efficiency, Testing Coverage.....
- How to reduce Test Generation Effort?

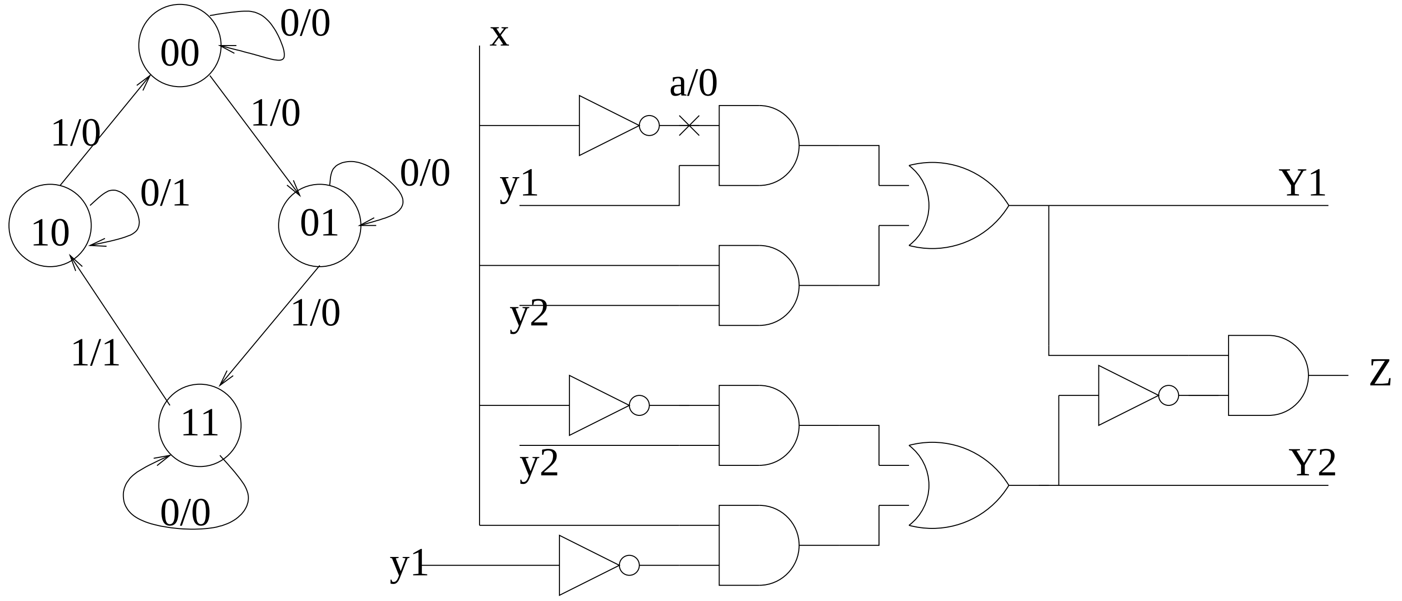
Sequential Circuit



Sequential Circuit Testing

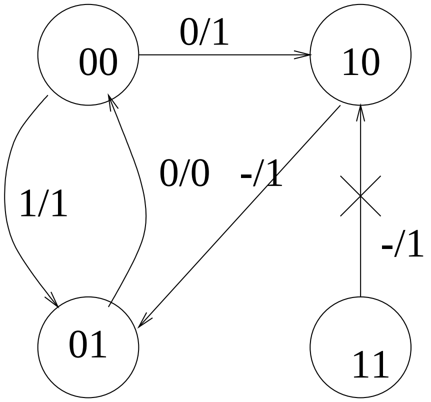


Sequential Circuit Testing Example



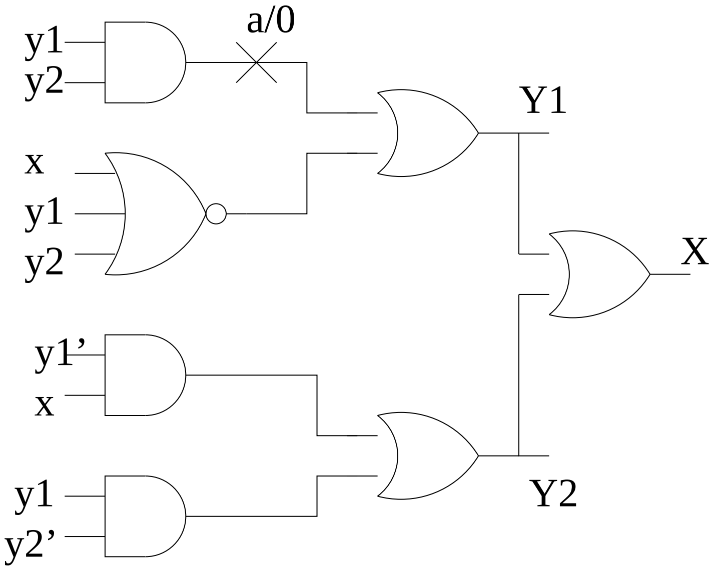
- Reset state: (00). Start in (00), and traverse the machine until...
- : $x_0 = 1, x_1 = 1, x_2 = 0, x_3 = 1$

Untestable faults due to unreachable states

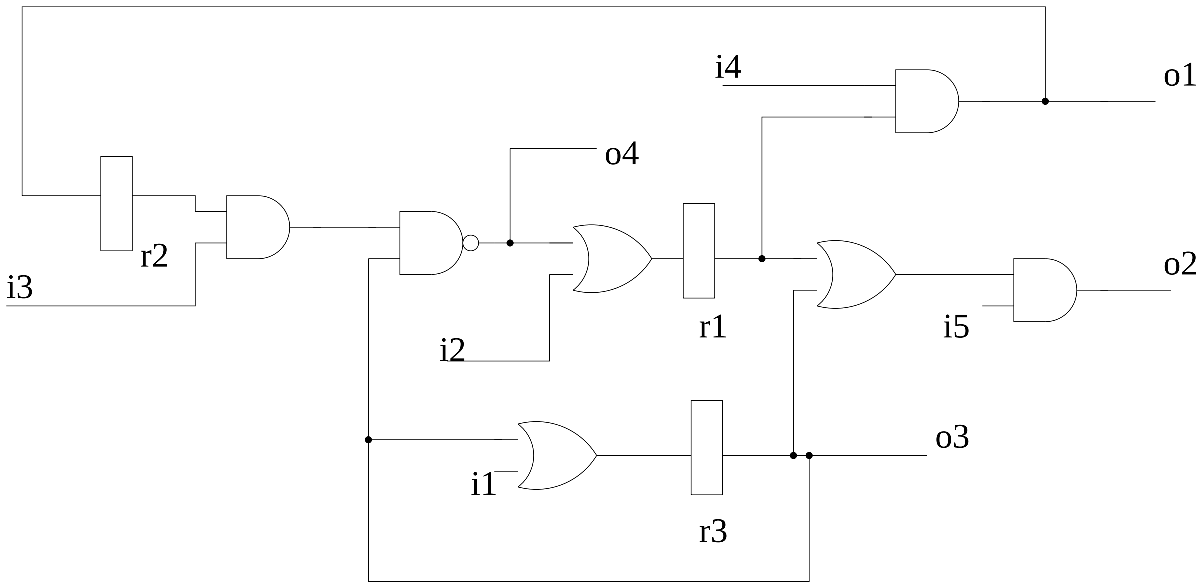


Y1 = y1 y2 +y1’ y2’ x’

Y2 = y1’x + y1 y2’



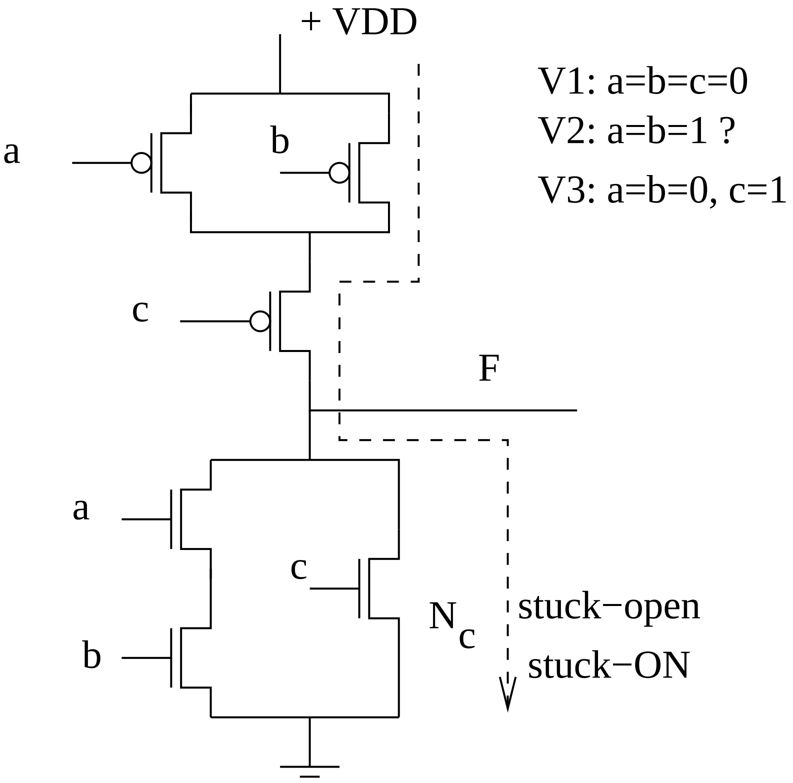
Logic Synthesis can cause Testing Problems



- Input i_3 to output o_2 cannot be "sensitized"
- Multi-Cycle "False Path"
- "Synthesize" circuit to make it easily testable!

IDDQ Testing

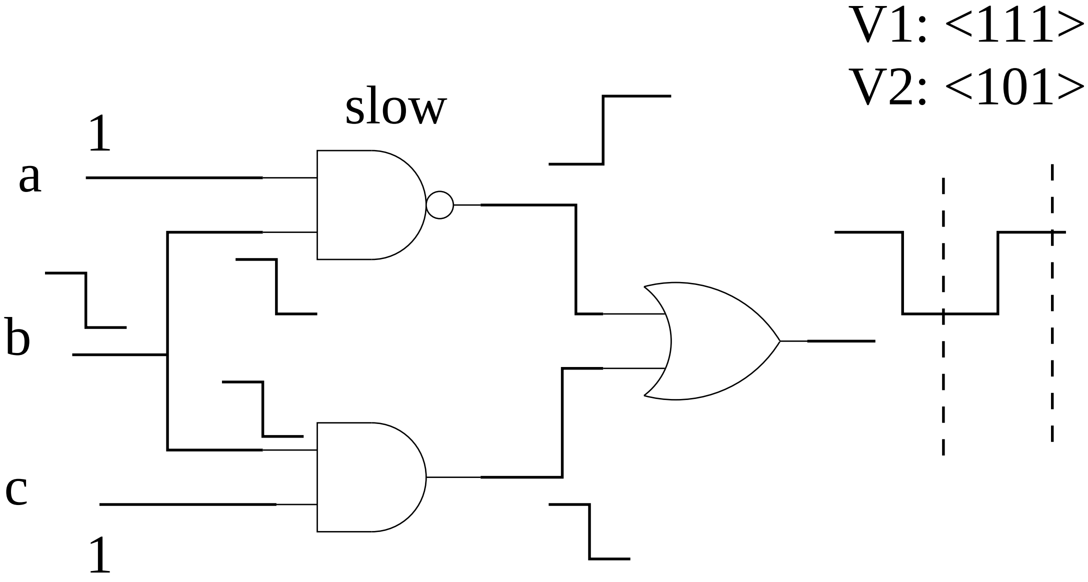
- Fault Location in Transistors: Test I_{DDQ}



V1: a=b=c=0
V2: a=b=1 ?
V3: a=b=0, c=1

Delay-Fault Testing

- Gates do not switch at desired speed
- Find vectors that detect delay faults!



Conclusion

- CAD Tools for Testing
- Test Generation, Test Application, Power issues in Testing, Test Time, Test Efficiency, Test Planning and Management
- TetraMax (Synopsys), HITEC, Mentor Graphics..
- Well Studied Problems, Good solutions
- Testing headache will always remain
- Courses:
  - ECE/CS 5740/6740: CAD of Digital Circuits, S'08
  - ECE/CS 5745/6745: Test & Verif. Digital Circuits, F'08