Introduction to CMOS VLSI Design

Lecture 17: Design for Testability

Outline

- Testing
  - Logic Verification
  - Silicon Debug
  - Manufacturing Test
- Fault Models
- Observability and Controllability
- Design for Test
  - Scan
  - Scannable flip flop

Testing

- Testing is one of the most expensive parts of chips
  - Logic verification accounts for > 50% of design effort for many chips
  - Debug time after fabrication has enormous opportunity cost
  - Shipping defective parts can sink a company
- Example: Intel FDIV bug
  - Logic error not caught until > 1M units shipped
  - Recall cost $450M (!!!)

Logic Verification

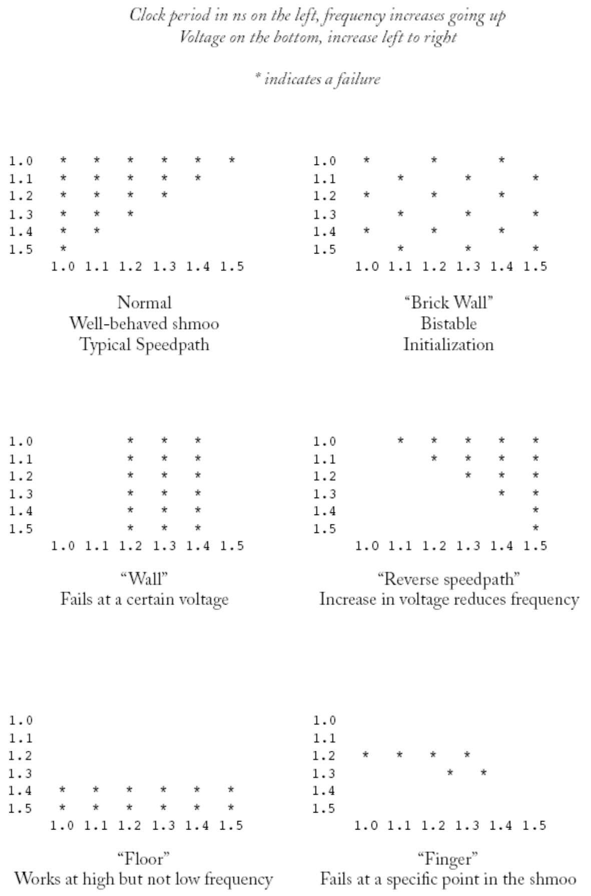
- Does the chip simulate correctly?
  - Usually done at HDL level
  - Verification engineers write test bench for HDL
    - Can’t test all cases
    - Look for corner cases
    - Try to break logic design
- Ex: 32-bit adder
  - Test all combinations of corner cases as inputs:
    - 0, 1, 2, 2^{31}-1, -1, -2^{31}, a few random numbers
- Good tests require ingenuity

Silicon Debug

- Test the first chips back from fabrication
  - If you are lucky, they work the first time
  - If not…
- Logic bugs vs. electrical failures
  - Most chip failures are logic bugs from inadequate simulation
  - Some are electrical failures
    - Crosstalk
    - Dynamic nodes: leakage, charge sharing
    - Ratio failures
  - A few are tool or methodology failures (e.g. DRC)
- Fix the bugs and fabricate a corrected chip

Shmoo Plots

- ❑ How to diagnose failures?
  - – Hard to access chips
    - Picoprobes
    - Electron beam
    - Laser voltage probing
    - Built-in self-test
- ❑ Shmoo plots
  - – Vary voltage, frequency
  - – Look for cause of electrical failures



Shmoo Plots

- ❑ How to diagnose failures?
  - – Hard to access chips
    - • Picoprobes
    - • Electron beam
    - • Laser voltage probing
    - • Built-in self-test
- ❑ Shmoo plots
  - – Vary voltage, frequency
  - – Look for cause of electrical failures

Clock period in ns on the left, frequency increases going up
Voltage on the bottom, increase left to right

** indicates a failure*

```
1.0   *   *   *   *   *   *         1.0   *       *       *
1.1   *   *   *   *   *             1.1       *       *       *
1.2   *   *   *   *                 1.2   *       *       *
1.3   *   *   *                     1.3       *       *       *
1.4   *   *                         1.4   *       *       *
1.5   *                             1.5       *       *       *
     1.0 1.1 1.2 1.3 1.4 1.5             1.0 1.1 1.2 1.3 1.4 1.5
```

Normal
Well-behaved shmoo
Typical Speedpath

"Brick Wall"
Bistable
Initialization

```
1.0           *   *   *             1.0       *   *   *   *   *
1.1           *   *   *             1.1           *   *   *   *
1.2           *   *   *             1.2               *   *   *
1.3           *   *   *             1.3                   *   *
1.4           *   *   *             1.4                       *
1.5           *   *   *             1.5                       *
     1.0 1.1 1.2 1.3 1.4 1.5             1.0 1.1 1.2 1.3 1.4 1.5
```

"Wall"
Fails at a certain voltage
Coupling, charge share, races

"Reverse speedpath"
Increase in voltage reduces frequency
Speedpath, leakage

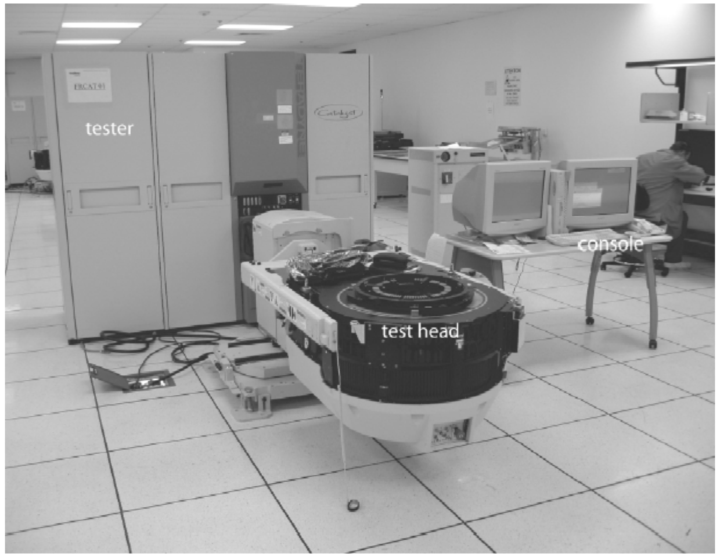
```
1.0                                 1.0
1.1                                 1.1
1.2                                 1.2   *   *   *   *
1.3                                 1.3             *   *
1.4   *   *   *   *   *   *         1.4
1.5   *   *   *   *   *   *         1.5
     1.0 1.1 1.2 1.3 1.4 1.5             1.0 1.1 1.2 1.3 1.4 1.5
```

"Floor"
Works at high but not low frequency
Leakage

"Finger"
Fails at a specific point in the shmoo
Coupling

Manufacturing Test

- A speck of dust on a wafer is sufficient to kill chip
- *Yield* of any chip is < 100%
  - Must test chips after manufacturing before delivery to customers to only ship good parts
- Manufacturing testers are very expensive
  - Minimize time on tester
  - Careful selection of *test vectors*

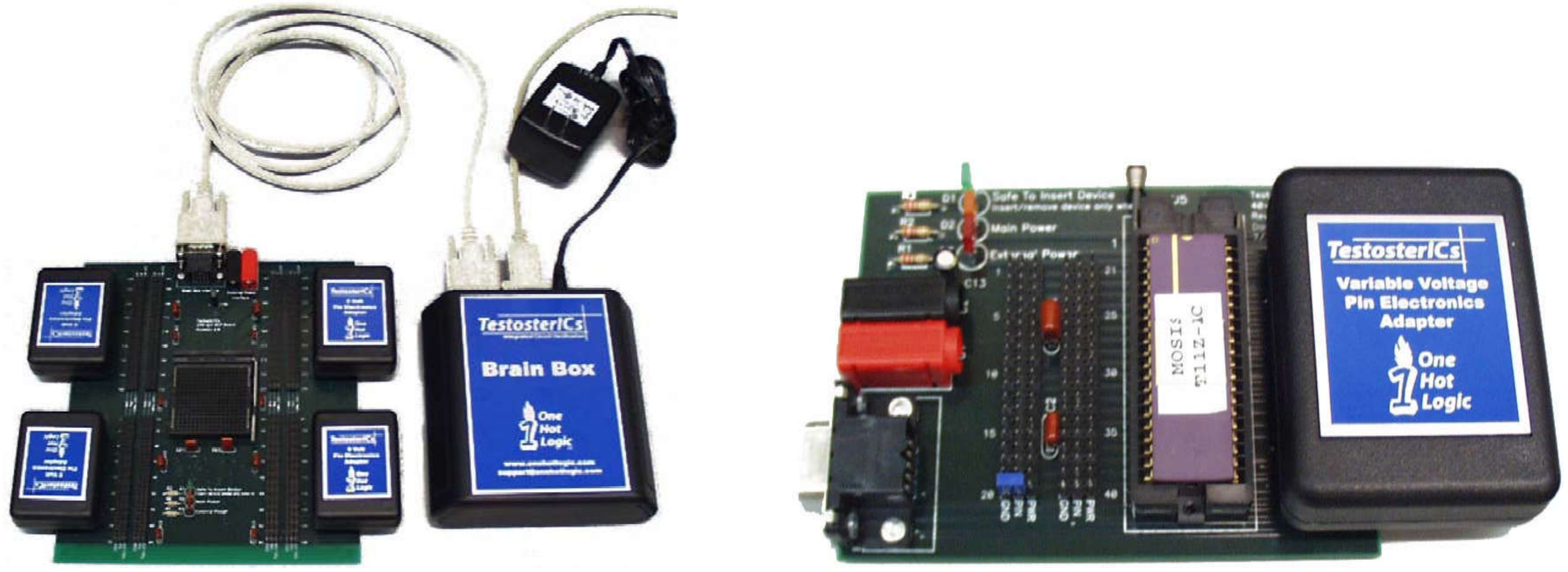


Testing Your Chips

- If you don't have a multimillion dollar tester:
  - Build a breadboard with LED's and switches
  - Hook up a logic analyzer and pattern generator
  - Or use a low-cost functional chip tester

TestosterICs

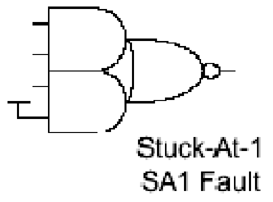
- Ex: TestosterICs functional chip tester
  - Designed by clinic teams and David Diaz at HMC
  - Reads your IRSIM test vectors, applies them to your chip, and reports assertion failures

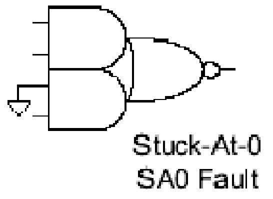


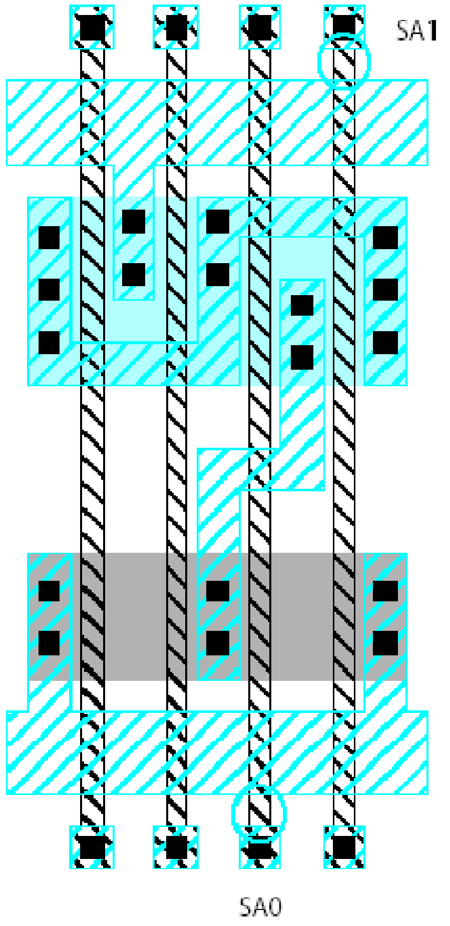
Stuck-At Faults

- 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 V_{DD}
  - – Not quite true, but works well in practice

Examples







Observability & Controllability

- *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

- Combinational logic is usually easy to observe and control
- Finite state machines can be very difficult, requiring many cycles to enter desired state
  – Especially if state transition diagram is not known to the test engineer

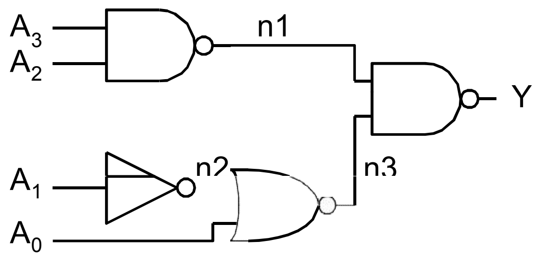
Test Pattern Generation

- ❑ 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 reduces number of test vectors required for manufacturing test.
  - – Reduces the cost of testing
  - – Motivates design-for-test

Test Example

	SA1	SA0
A_3	{0110}	{1110}
A_2	{1010}	{1110}
A_1	{0100}	{0110}
A_0	{0110}	{0111}
n1	{1110}	{0110}
n2	{0110}	{0100}
n3	{0101}	{0110}
Y	{0110}	{1110}



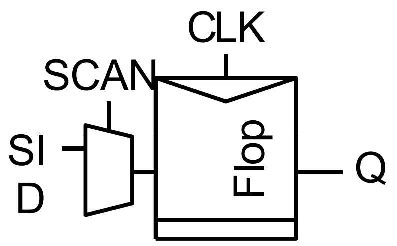
- Minimum set: {0100, 0101, 0110, 0111, 1010, 1110}

Design for Test

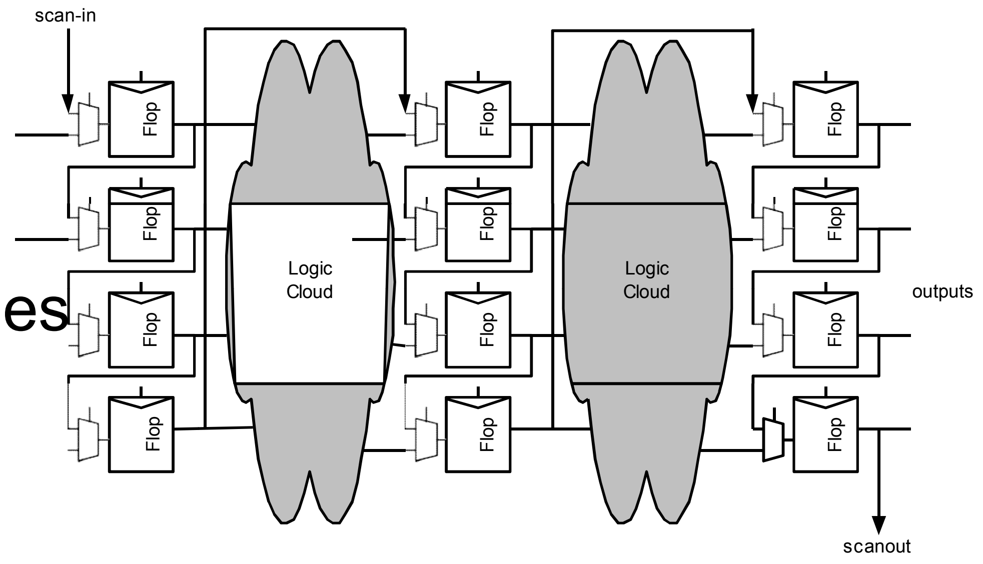
- ❑ Design the chip to increase observability and controllability
- ❑ If each register could be observed and controlled, test problem reduces to testing combinational logic between registers.
- ❑ Better yet, logic blocks could enter test mode where they generate test patterns and report the results automatically.

Scan

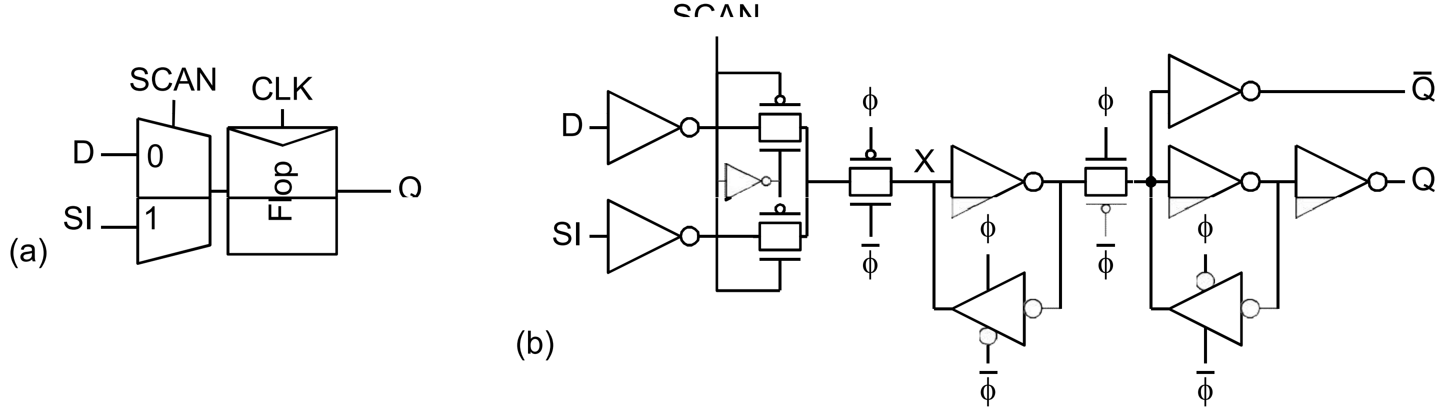
- Convert each flip-flop to a scan register
  - – Only costs one extra multiplexer
- Normal mode: flip-flops behave as usual
- Scan mode: flip-flops behave as shift register

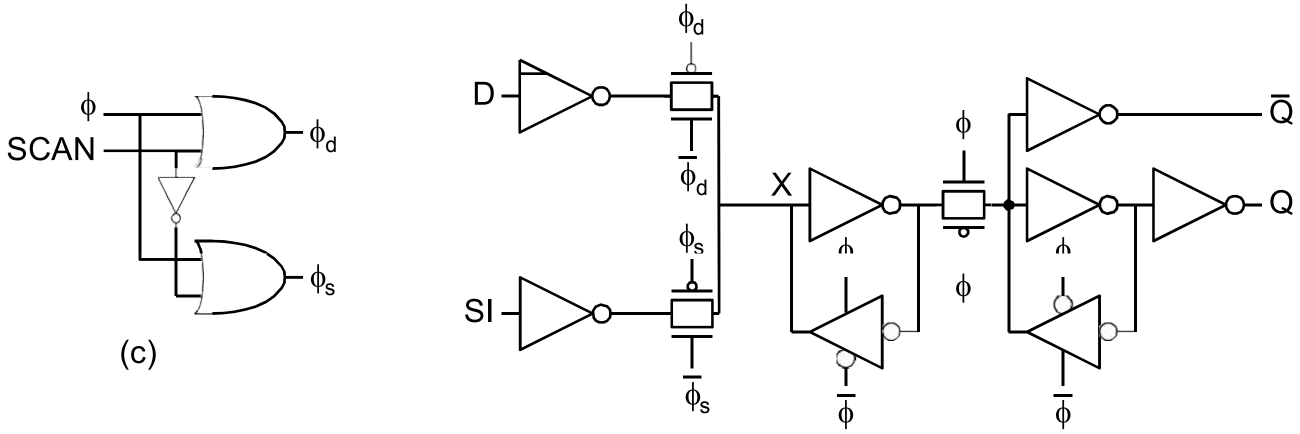


- Contents of flops can be scanned out and new values scanned in



Scannable Flip-flops





Summary

- ❑ Think about testing from the beginning
  - – Simulate as you go
  - – Plan for test after fabrication

- ❑ "If you don't test it, it won't work! (Guaranteed)"
- ❑ Testing
  - – Logic Verification
  - – Silicon Debug
  - – Manufacturing Test
- ❑ Fault Models
- ❑ Observability and Controllability
- ❑ Design for Test
  - – Scan
  - – Scannable flip flop