Testing of Logic Circuits

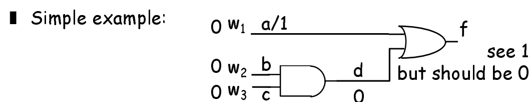
- Fault Models
- Test Generation and Coverage
- Fault Detection
- Design for Test

Fault Model

- Stuck-At Model
  - Assume selected wires (gate input or output) are “stuck at” logic value 0 or 1
  - Models curtain kinds of fabrication flaws that short circuit wires to ground or power
    - Wire w stuck-at-0: w/0
    - Wire w stuck-at-1: w/1
  - Often assume there is only one fault at a time—even though in real circuits multiple simultaneous faults are possible
  - Obviously a very simplistic model!

Fault Model

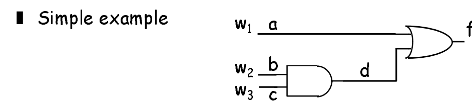
- Simple example:



- Generate a testcase to determine if a is stuck at 1
  - Try 000
  - If a stuck at 1, expect to see f = 0, but see 1 instead

Fault Model

- Simple example



Test	Fault Detected									
w1 w2 w3	a/0	a/1	b/0	b/1	c/0	c/1	d/0	d/1	f/0	f/1
000		X						X		X
001		X		X				X		X
010		X				X		X		X
011			X		X		X		X	
100	X								X	
101	X								X	
110	X								X	
111									X	

Test Set

Problems with Fault Model

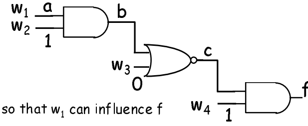
- In general, n-input circuits require much less than 2^n test inputs to cover all possible stuck-at-faults in the circuit
- However, this number is usually still too large in real circuits for practical purposes
- Finding minimum test cover is an NP-hard problem too

Path Sensitization

- Wire-at-time testing too laborious
- Better to focus on wiring paths, enabling multi-wire testing at the same time
- “Activate” a path so that changes in signal propagating along the path affects the output

Path Sensitization

- Simple Example:



To activate the path, set inputs so that w_1 can influence f

E.g., $w_2 = 1$, $w_3 = 0$, $w_4 = 1$
AND gates: one input at 1 passes the other input
NOR gates: one input at 0 inverts the other input

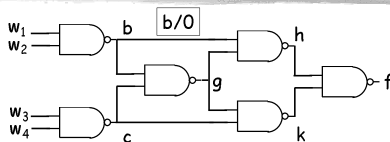
To test: w_1 set to 1 should generate f = 0 if path ok
faults a/0, b/0, c/1 cause f = 1
w_1 set to 0 should generate f = 1 if path ok
faults a/1, b/1, c/0 cause f = 0

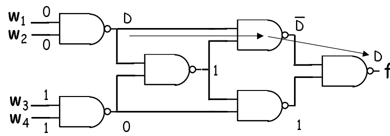
One test can capture several faults at once!

Path Sensitization

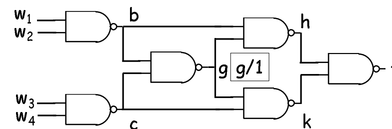
- Good news: one test checks for several faults
  - Number of paths much smaller than number of wires
  - Still an impractically large number of paths for large-scale circuits
- Path idea can be used to "propagate" a fault to the output to observe the fault
  - Set inputs and intermediate values so as to pass an internal wire to the output while setting inputs to drive that internal wire to a known value
  - If propagated value isn't as expected, then we have found a fault on the isolated wire

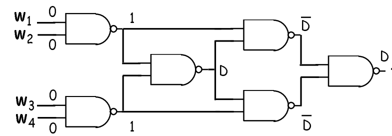
Fault Propagation



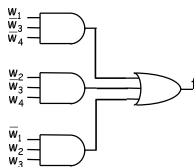


Fault Propagation



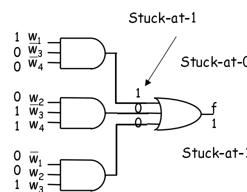


Tree Structured Circuits



- To test inputs stuck-at-0 at given AND gate
  - Set inputs at other gates to generate AND output of zero
  - Force inputs at selected gate to generate a one
  - If f is 1 then circuit ok, else fault
- To test inputs stuck-at-1 at given AND gate
  - Drive input to test to 0, rest of inputs driven to 1
  - Other gates driven with inputs that force gates to 0
  - If f is 0 then fault, else OK

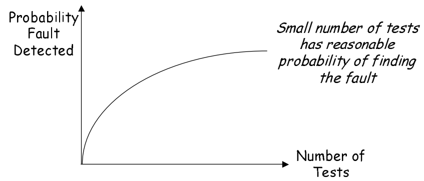
Tree Structured Circuits



	Product Term									Test			
	w_1	$\bar{w}_3$	$\bar{w}_4$	w_2	$\bar{w}_3$	w_4	$\bar{w}_1$	w_2	w_3	w_1	w_2	w_3	w_4
1	1	1	1	0	1	0	0	0	0	0	0	0	0
2	0	1	0	1	1	1	1	1	0	1	1	1	0
3	0	0	0	1	0	1	1	1	1	1	1	1	1
4	0	1	1	1	1	0	1	1	0	0	1	1	0
5	1	0	1	1	0	0	0	1	1	0	0	1	1
6	1	1	0	0	1	1	0	0	0	1	0	0	0
7	1	0	0	1	0	1	0	1	1	1	0	1	1
8	0	0	0	0	0	1	1	0	1	1	1	0	1

Random Testing

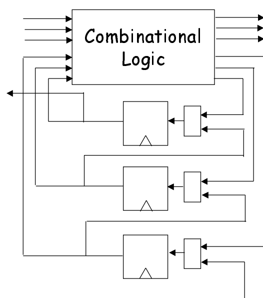
- So far: deterministic testing
- Alternative: random testing
  - Generate random input patterns to distinguish between the correct function and the faulty function



Sequential Testing

- Due to embedded state inside flip-flops, it is difficult to employ the same methods as with combinational logic
- Alternative approach: design for test
  - Scan Path technique: FF inputs pass through multiplexer stages to allow them to be used in normal mode as well as a special test shift register mode

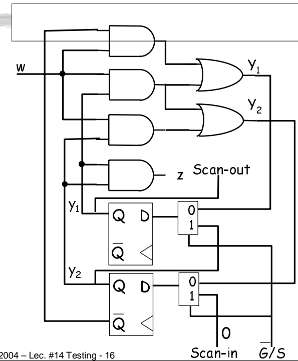
Scan Path Technique



- Configure FFs into shift register mode
- Scan in pattern of 0s and 1s
- Non-state inputs can also be on the scan path
- Run system for one clock cycle in normal mode—next state captured in scan path
- Return to shift register mode and shift out the captured state and outputs

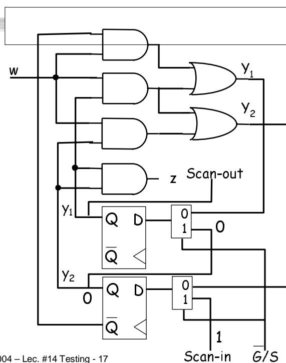
Scan Path Example

- w,y1,y2 test vector 001
  - Scan 01 into y1, y2 FFs



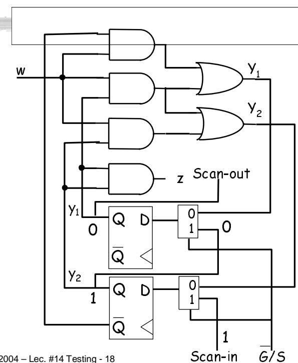
Scan Path Example

- w,y1,y2 test vector 001
  - Scan 01 into y1, y2 FFs



Scan Path Example

- w,y1,y2 test vector 001
  - Scan 01 into y1, y2 FFs



Scan Path Example

- w,y1,y2 test vector 001
  - Scan 01 into y1, y2 FFs
  - Normal w=0

Scan Path Example

- w,y1,y2 test vector 001
  - Scan 01 into y1, y2 FFs
  - Normal w=0
  - Output z=0, Y1=0, Y2=0

Scan Path Example

- w,y1,y2 test vector 001
  - Scan 01 into y1, y2 FFs
  - Normal w=0
  - Output z=0, Y1=0, Y2=0
  - Observe z directly

Scan Path Example

- w,y1,y2 test vector 001
  - Scan 01 into y1, y2 FFs
  - Normal w=0
  - Output z=0, Y1=0, Y2=0
  - Observe z directly
  - Scan out Y1, Y2

Scan Path Example

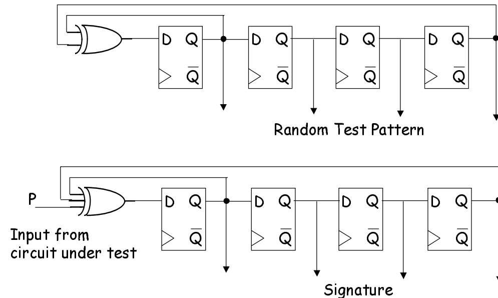
- w,y1,y2 test vector 001
  - Scan 01 into y1, y2 FFs
  - Normal w=0
  - Output z=0, Y1=0, Y2=0
  - Observe z directly
  - Scan out Y1, Y2

Built-in Self-Test (BIST)

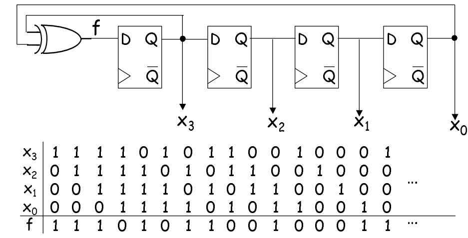
Test Vector Generator → x_0 . . . x_{n-1} → Circuit Under Test → P_0 . . . P_{m-1} → Test Response Compressor → Signature

- Test Vector Generator
  - Pseudorandom tests with a feedback shift register
  - Seed generates a sequence of test patterns
  - Outputs combined using the same technique
  - Generates a unique signature that can be checked to determine if the circuit is correct

Linear Feedback Shift Register



Linear Feedback Shift Register

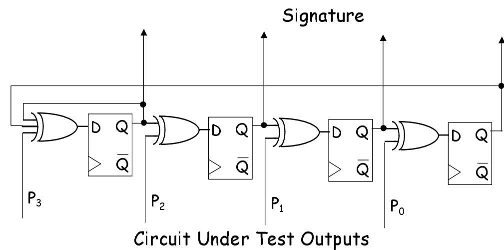


x_3	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1	
x_2	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0	
x_1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0	...
x_0	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0	
f	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1	...

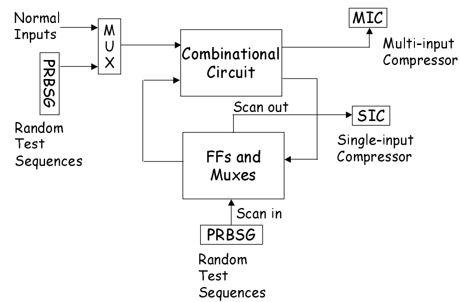
- Starting with the pattern 1000, generates 15 different patterns in sequence and then repeats

Linear Feedback Shift Register

- Multi-input Compressor

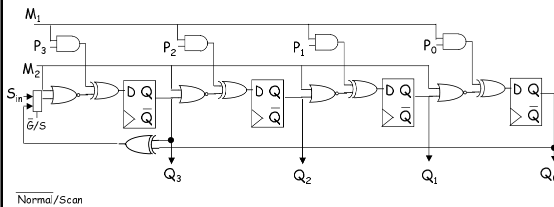


Complete Self-Test System



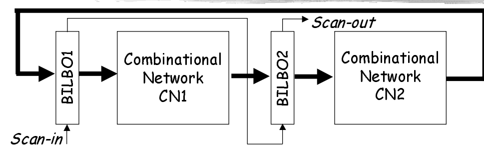
Built-in Logic Block Observer (Bilbo)

- Test generation and compression in a single circuit!
  - M_1, M_2 = 11: Regular mode
  - M_1, M_2 = 00: Shift register mode
  - M_1, M_2 = 10: Signature generation mode
  - M_1, M_2 = 01: Reset mode



$\overline{\text{Normal}}$/Scan

Bilbo Architecture



- Scan initial pattern in Bilbo1, reset FFs in Bilbo2
- Use Bilbo1 as PRBS generator for given number of clock cycles and use Bilbo2 to produce signature
- Scan out Bilbo2 and compare signature; Scan in initial test pattern for CN2; Reset the FFs in Bilbo1
- Use Bilbo2 as PRBS generator for a given number of clock cycles and use Bilbo1 to produce signature
- Scan out Bilbo1 and compare signature;

Summary

- Fault models
  - Approach for determining how to develop a test pattern sequence
  - Weakness is the single fault assumption
- Scan Path
  - Technique for applying test inputs deep within the system, usually for asserting state
  - Technique for getting internal state to edges of circuit for observation
- Built-in Test
  - Founded on the approach of random testing
  - Generate pseudo random sequences; compute signature; determine if signature generated is same as signature of a correctly working circuity