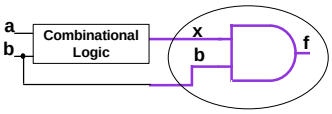
Outline

- **Last time:**
  - ➔ Deriving the State Diagram & Datapath (Cont.)
  - ➔ Mapping the Datapath onto Control
- **This lecture:**
  - **Combinational Testability and Test-pattern Generation**
  - **Faults in digital circuits**
  - **What is a test? : Controllability & Observability**
  - **Redundancy & testability**
  - **Test coverage & simple PODEM ATPG**
  - **Sequential Test: What are sequential faults?**
  - **SCAN Design**

Outline

- **Background:**
  - **Role of Don't-Cares in Logic Synthesis**
  - **Controllability & Observability**
  - **Optimality, Redundancy & Testability**
- **The Sequential Test Problem**
- **Synthesis-Directed Sequential Test**
  - **Two Approaches to Full Testability**
  - **Effectiveness and Limitations so far**

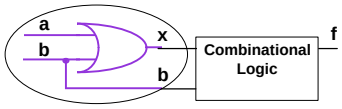
Role of Don't-Cares in Logic Synthesis



- **f = xb cannot be reduced further in isolation**

x \ ab	00	01	11	10
0				
1		1	1	

Role of Don't-Cares in Logic Synthesis

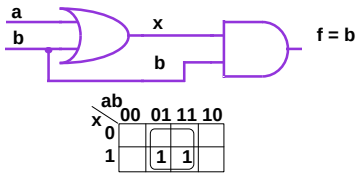


- **x ¹ a + b can never happen**
- **Don't-Care Set:**

  D = x' (a + b) + x a' b'
- **Minimize f with respect to D.**

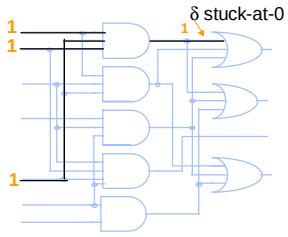
x \ ab	00	01	11	10
0				
1				

Role of Don't-Cares in Logic Synthesis



x \ ab	00	01	11	10
0				
1		1	1	

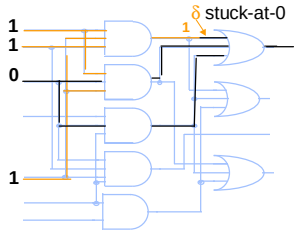
Fault Excitation



Fault Models

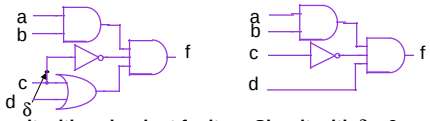
- **Input or output pin (not entire net!) stuck at logic 0 or stuck at logic 1.**
- **Open circuit**
  - **Can make a combinational circuit sequential!**
- **Short circuit**

Fault Propagation



❍ **The test is a cube, not a minterm.**

Optimality & Redundancy in Combinational Logic



Circuit with redundant fault:

δ stuck-at-0

f = (a.b).(c+d).c'
= (a.b.c + a.b.d).c'
= a.b.c.c' + a.b.d.c'
= a.b.d.c'

Circuit with δ = 0

f = (a.b).c'.d

Path-Oriented DEcision Making [Goel, 1981]

(1) Assign all Primary Inputs (PI) to the value "don't care" ( ).

(2) Given an output signal and a desired value for the output, trace a path to the PIs to obtain a PI assignment.

(3) Simulate the PI vector to see if it sets up the desired value on the output. If so, terminate.

(4) If the opposite value is set, assign an opposite value to the PI and re-simulate. If desired value is set, terminate.

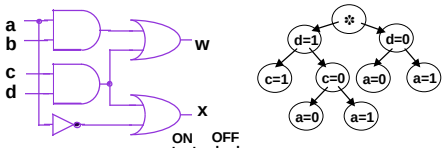
(5) If the output remains unspecified, repeat the path tracing to set another PI, as necessary.

❍ **Procedure continues until either:**

➔ A successful PI assignment has been found (circuits not equivalent).

➔ All possible PI assignments have been exhausted.

Cover Extraction



ON abcd	OFF abcd
✲✲11	1**✲**01
0**✲**01	1**✲✲**0
0**✲✲**0	

❍ **Covers can be generated with as many "don't cares" in the present state part as possible.**

Testability and Logic Synthesis

❍ **Important Issue:**

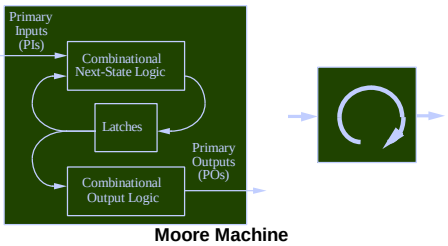
Generating tests for circuits with redundancies is very difficult.

➔ **Must use algorithms which decrease the number of redundancies or eliminate them completely during synthesis.**

Test Generation for Finite-State Machines

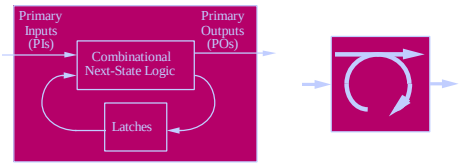
- Irredundant combinational logic does not imply 100% sequential testability
- Sequential Faults: Faults may not be excited ("controlled") by primary inputs; faults may not be propagated to primary outputs ("observed").

Finite-State Machines



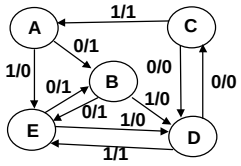
Moore Machine

Finite-State Machines

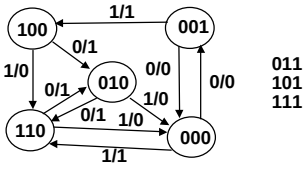


Mealy Machine

Example Finite-State Machine: State Transition Diagram



Example Finite-State Machine: Encoded States



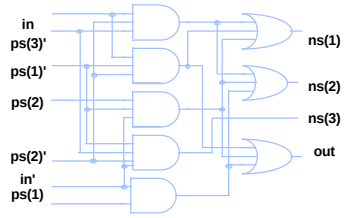
State Assignment

- Find a binary encoding of states which minimizes the eventual area (or delay) of the FSM after combinational logic optimization of NSL and OL.

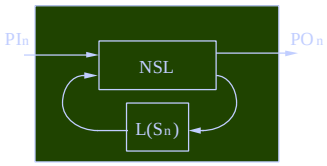
FSM → ENCODE → OPTIMIZE →

- Need to predict and model the optimization
- State assignment has major effect on testability.

Example Finite-State Machine: Next-State Logic

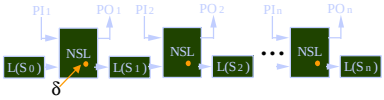


Mealy Machine at Time t_n



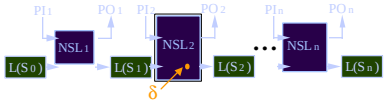
- S_n is state of latches at time t_n

Finite-State Machine as Iterated Array



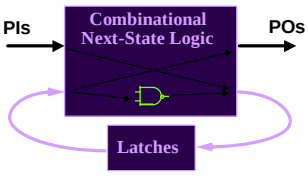
- Fault is present in all copies of NSL
- Fault may mask excitation or propagation
- More likely, fault may cause next-state N to be invalid state Nf.

Ideal Iterated Array



- In an ideal situation, the NSL would be optimized separately for each possible state transition!
- Each NSL block would be made prime and irredundant separately.

Sequential Circuits: Controllability & Observability



A test:

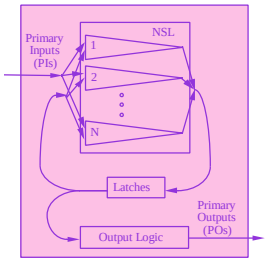
001 1:1
010 1:1
010 0:0

111 1:1
110 0:1

Scan Design

- Make all flip-flops scan (i.e. direct read and write access)
  - ➔ All inputs to the combinational logic can be set and all outputs can be read.
  - ➔ The sequential testing problem becomes a combinational testing problem.
- "Overkill" in virtually all cases.
- Area and time penalty; often a longer testing time.
- *But* scan can be inserted automatically.

Synthesis Procedure for Fully-Testable Non-Scan Finite-State Machine (Devadas, et.al. 1988)



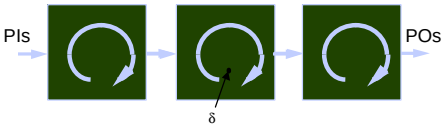
- Partition NSL into single-cone circuits
- Single stuck-fault ⇒ correct & incorrect next-state differ by exactly one bit.
- Perform state assignment such that all states differing in one bit assert different outputs ⇒ one-step propagation

Synthesis Procedure for Fully-Testable Non-Scan Finite-State Machines

Given a state-transition graph (STG) of a FSM, a 100%-testable logic-level implementation of the machine is produced

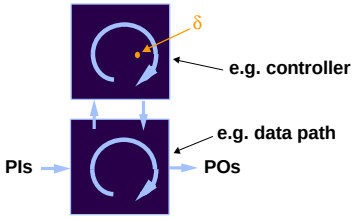
- No scannable latches required
- Uses partitioned logic approach and constrained state assignment
- Small penalty

Cascaded Finite-State Machines



- Is it possible to synthesize a cascade of FSM's such that all embedded faults are detectable non-scan from the external inputs only?
- What is the penalty in real cases?

Coupled Finite-State Machines



Example FSMs

Name	No. Inputs	No. Outputs	No. States	No. Edges
sse	7	7	13	59
tbk	6	3	16	787
dfile	2	1	24	99
planet	7	19	48	118
scf	27	54	97	168

Constrained State Assignment (single cones)

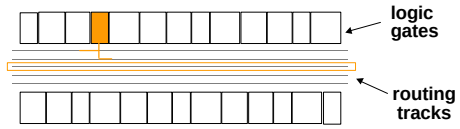
Name	Optimized only			Optimized and Testable		
	No. Gates	Fault Cover (%)	TPG time	No. Gates	Fault Cover (%)	TPG time
sse	91	84.6	70s	129	100	5s
tbk	181	98.6	72s	231	98.6	4s
dfile	124	96.9	104s	144	100	2s
planet	417	98.8	373s	449	100	14s
scf	502	96.1	83m	541	100	71s

Output logic block contained combinational redundancies

Constrained State Assignment (single cones)

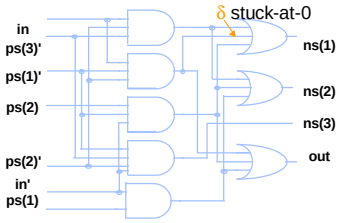
Name	Optimized only No. Gates	Optimized only Normalized Area	Optimized and Testable No. Gates	Optimized and Testable Normalized Area
sse	91	1.00	129	1.34
tbk	181	1.00	231	1.10
dfile	124	1.00	144	0.98
planet	417	1.00	449	0.86
scf	502	1.00	541	1.01

Effect of Gate Duplication in Standard-Cell Layout

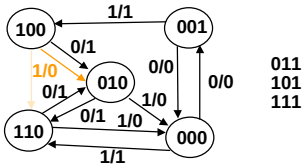


- Duplication of gates reduces routing congestion and may save a routing track.
- The area of a routing track is usually >> the area of a gate.
- Reducing maximum gate fanout *may* improve performance.

Example Finite-State Machine with Fault d

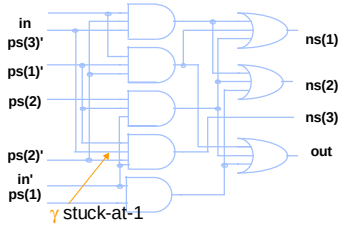


Example Finite-State Machine Effect of Fault d

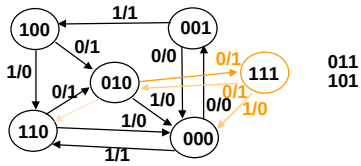


- N f ° N, where N f is a valid state.

Example Finite-State Machine with Fault g

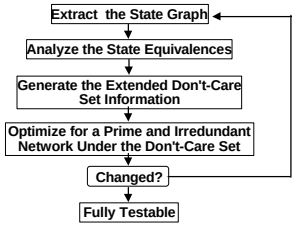


Example Finite-State Machine Effect of Fault g



- N f ° N, where N f is a invalid state (state splitting has occurred).

Use of Extended Don't-Care Set to Guarantee Testability (Devadas & Keutzer, '90)



Example FSMs

Name	No. Inputs	No. Outputs	No. States	No. Edges
ex1	2	2	6	24
ex2	2	1	13	57
s1	8	6	20	110
dfile	2	1	24	96
keyb	7	2	19	170

Results of Synthesis Procedure

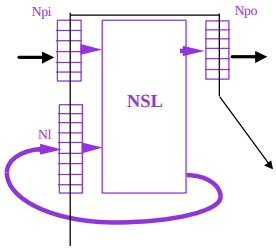
Name	No. Latches	No. Gates	Fault Cover (%)	Optimize	TPG	identify redund.	remove redund.
ex1	3	23	97.9	0.5s	2.0s	1.1s	2.0s
ex2	5	35	98.2	2.2s	42s	6.1s	1.8m
s1	5	105	99.8	5.5s	303s	4.0s	303s
dfile	6	77	97.8	6.2s	332s	42s	>1h
keyb	5	146	98.7	29.5s	21m	1.2m	>1h

Results Using Extended Don't-Care Sets During Synthesis

Name	State Enum.	Optimize Time	TPG	No. Logic Optimize	Fault Cover (%)
ex1	0.5s	0.5s	2.1s	1	100.0
ex2	6.5s	22.4s	41s	7	100.0
s1	1.0s	6.1s	298s	1	100.0
dfile	10.2s	25.5s	747s	3	100.0
keyb	14.6s	27.8s	22m	2	100.0

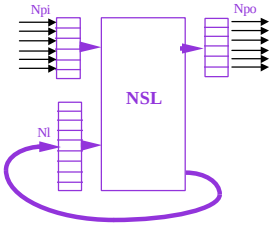
- 2%-5% smaller designs than without don't-care sets.

Test Procedure: Scan



- Nl+Npi input bits per test
- One tester clock tick per bit

Test Procedure: Non-Scan



- Npi input bits per test
- One tester clock tick per Npi bits

What About Testing Time?

(Ghosh et. al. 1989)

Number of Test Bits

Name	Scan	S for T
ex1	4,032	13,728
ex2	29,696	111,510
ex3	55,680	134,290
des	8768	22,826
key	11,856	51,968
viterbi	15,168	224,950

Viterbi Chip

- Part of a system for real-time continuous speech recognition developed by Prof's. Broderson & Rabaey at Berkeley.
- Largest chip in the chip-set for the system.
- Implements the Viterbi algorithm for mapping an observation (some speech) into the most likely sequence of states in the speech model being used.
- Chip Statistics:
  - 25,000 transistors
  - 116 inputs, 44 outputs.
  - 10 x11.5 mm die size

What About Testing Time?

	Tester Cycles		Test Bits	
Name	Scan	S for T	Scan	S for T
ex1	4,032	208	4,032	13,728
ex2	29,696	590	29,696	111,510
ex3	55,680	1,033	55,680	134,290
des	8,768	202	8768	22,826
key	11,856	203	11,856	51,968
viterbi	15,168	2,045	15,168	224,950

A Revolution in Test in the Late 1990s?

- Can Synthesize a Guaranteed Fully-Testable, Non-Scan Implementation of Any Collection of FSMs.
  - ✓ Almost always requires fewer gates or *less area than full scan.*
  - ✓ Almost always requires *shorter tester times* (in many cases by one or two orders of magnitude) than full scan.
  - ✓ Can handle faults in embedded machines, machines with feedback, etc. - any topology of interconnected machines.
  - ✓ Test patterns generated as a by-product of the synthesis, so synthesis time represents a saving of ATPG time.

Synthesis-Directed Sequential Test

- Entire-chip full-scan-based design-for-test will be *obsolete* by the end of the 1990s
  - ➔ Will be used for some very-specific on-chip structures (e.g. ROM, RAM, maybe Datapath) and for some chip boundaries.
- Circuit-structure-specific and BILBO-like test styles will continue to be used for go-nogo tests.
- Architectural memory structures will continue to be accessible directly for the pins.

Synthesis-Directed Sequential Test

- Test will be incorporated directly into the synthesis process
  - ➔ Guaranteed fully-testable non-scan or partial-scan designs will be produced by the synthesis process.
  - ➔ A complete set of test patterns will be a by-product of the process