CIS 4930 Digital Circuit Testing
Functional Testing

Dr Hao Zheng
Comp. Sci. & Eng.
U of South Florida

Introduction

- Testing for SSFs is based on structural model.
  - May not be available, or
  - May be too complex.
  - Physical fault testing cannot check design errors.
- Functional testing is based on functional model.
  - Specifies design functionality.
  - Check physical faults + design errors.

Introduction – cont'd

- **Objective**: validate system implementation wrt its functional specification.
- Functional testing without fault models
  - Generate tests wrt fault free behavior
- Functional testing using specific fault models
  - Directed tests
- Exhaustive and pseudoexhaustive testing

Functional Testing without Fault Models

- Exercise specified functions.
- Ex: to test D-FFs, need to validate that
  - It can set and reset, and
  - It can hold values.
- Coverage Problem: more difficult to evaluate the quality of tests with fault models.

Heuristics for Coverage

- **Operation activation**

$$\textbf{if } x \textbf{ then } \text{operation1} \textbf{ else } \text{operation2}$$

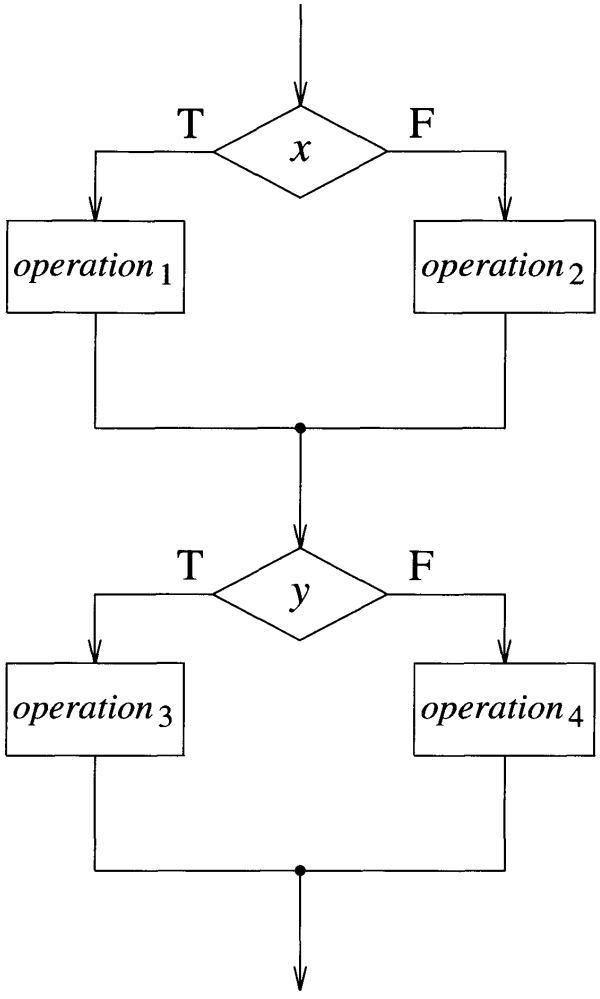
- A "complete" test should exercise both branches.
- **Decision path tracing**

$$\textbf{if } y \textbf{ then } \text{operation3} \textbf{ else } \text{operation4}$$

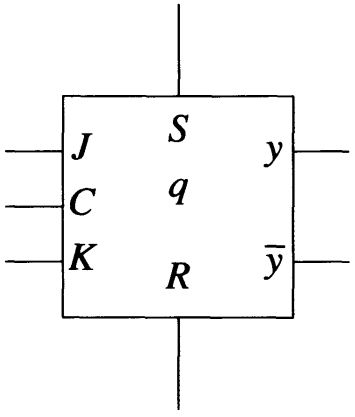
  - There are 4 paths considering those two statement

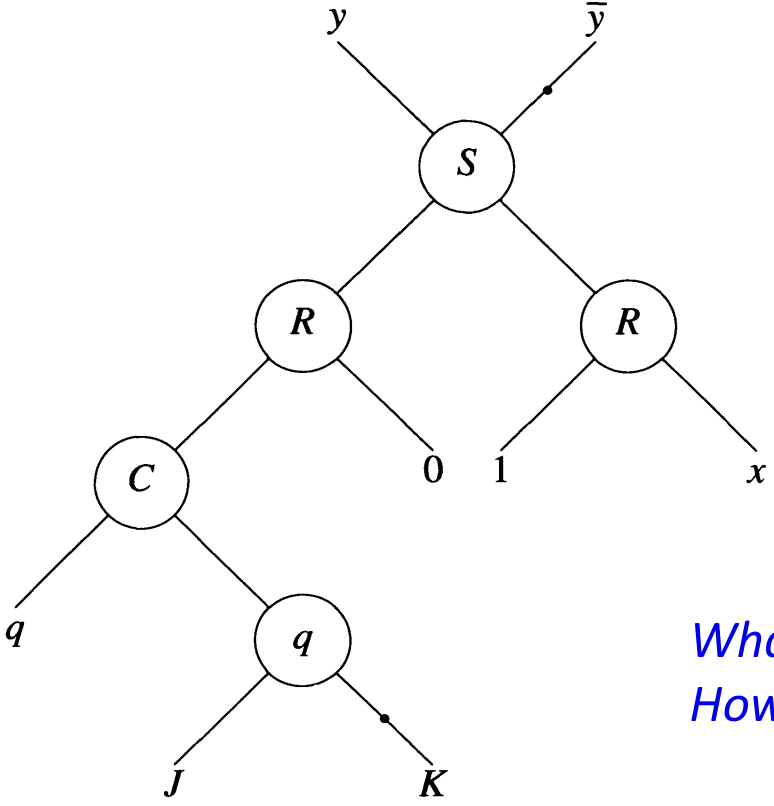
Heuristics for Coverage

- **Decision path tracing:**
  - coverage measure by the ratio of paths traversed vs total # of paths.
- **Checking unintended behavior:**
  - Possibility of writing data to R2 in addition to R1?



Functional Testing with BDDs





S	R	C	q	J	K	y
0	1	x	x	x	x	0
1	0	x	x	x	x	1
0	0	0	q	x	x	q
0	0	1	0	J	x	J
0	0	1	1	x	K	$\bar{K}$

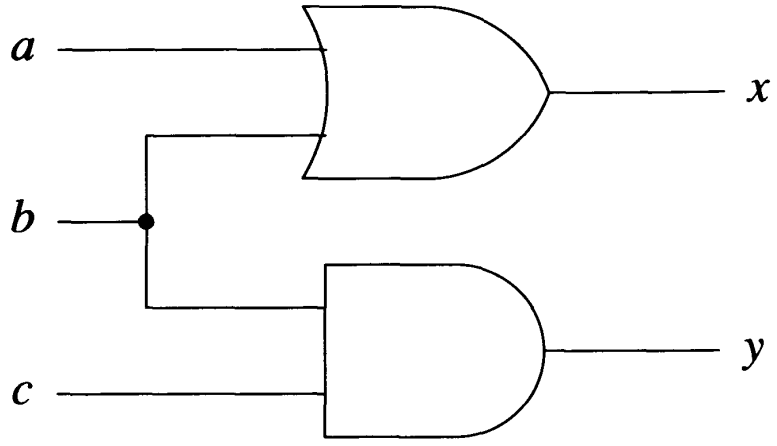
What functions to test?
How to generate tests from BDDs?

Exhaustive & PseudoExhaustive Testing

- Universal fault model
  - Any fault in a circuit is possible.
    - Any faults that changes a circuit's function
  - Need to apply all 2^n input vectors for n PIs.
  - only practical for small circuits.
- **PseudoExhaustive** Testing
  - Consider certain structural information
  - Significantly reduce the input vectors by circuit partitioning wrt POs.

Partial-Dependence Combinational Circuits

- Definition: No PO depends on all PIs.
- Only need 2^{n_i} input vectors for PO O_i with n_i PIs.



(a)

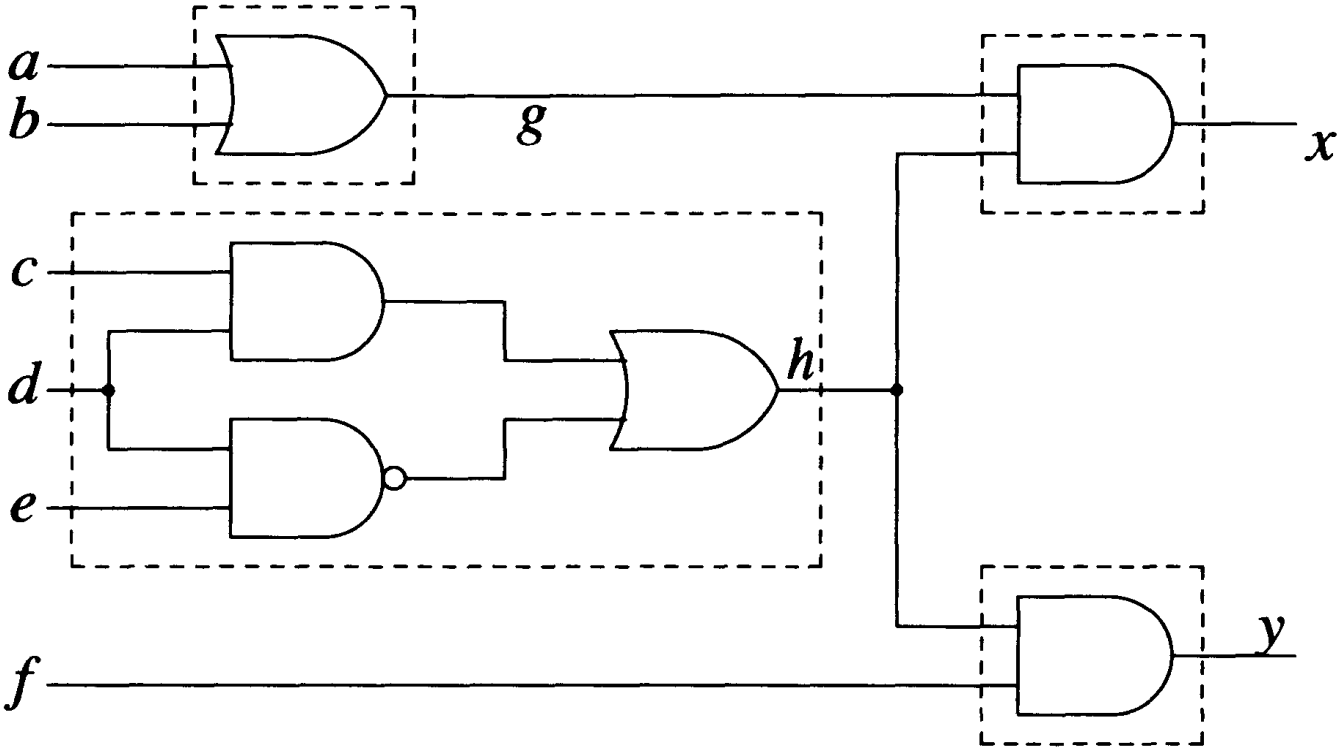
a	b	c
0	0	0
0	1	0
1	0	1
1	1	1

(b)

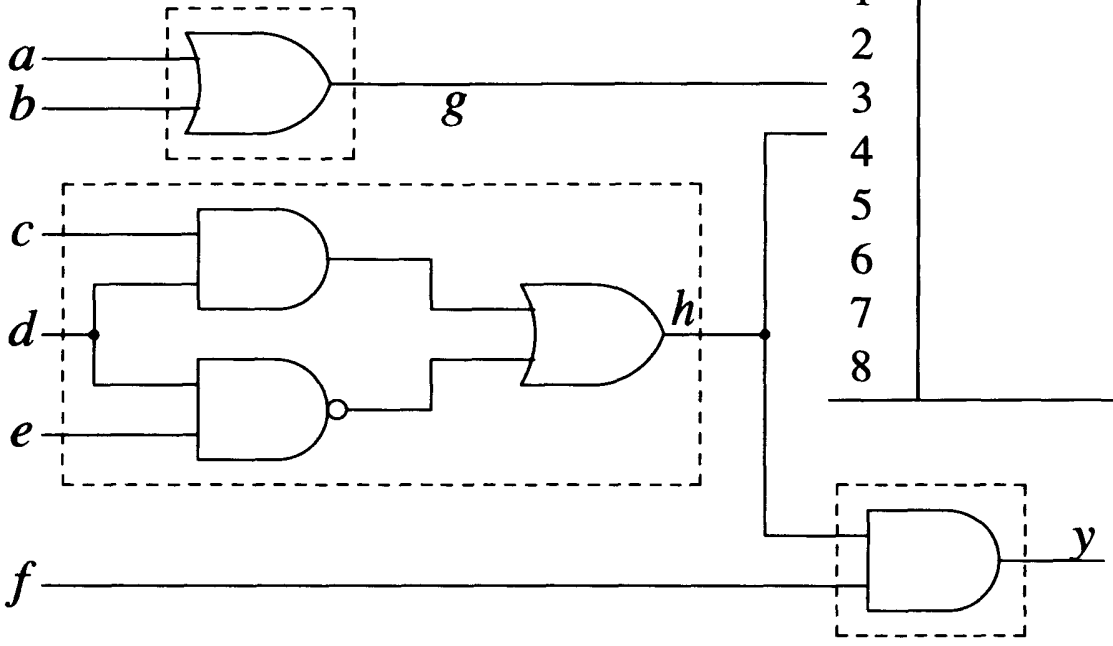
Circuit Partitioning

- Pseudo-exhaustive testing still not practical for large n_i
  - Or total dependence circuits
- Circuit partitioned into *segments* with limited # of inputs.
- If inputs/outputs of a segment are not PIs/POs,
  - Need to control segment's inputs from PIs,
  - Need to observe segment's outputs on POs.

Circuit Partitioning



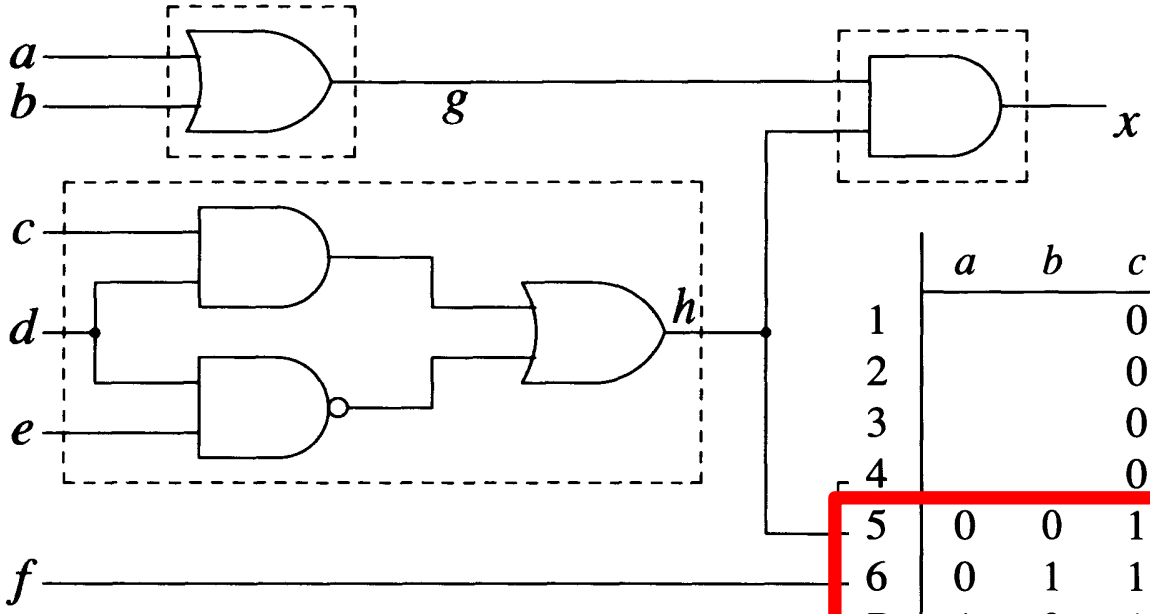
Circuit Partitioning – Vectors for Testing h



	a	b	c	d	e	f	g	h	x	y
1			0	0	0	1		1		1
2			0	0	1	1		1		1
3			0	1	0	1		1		1
4			0	1	1	1		0		0
5			1	0	0	1		1		1
6			1	0	1	1		1		1
7			1	1	0	1		1		1
8			1	1	1	1		1		1

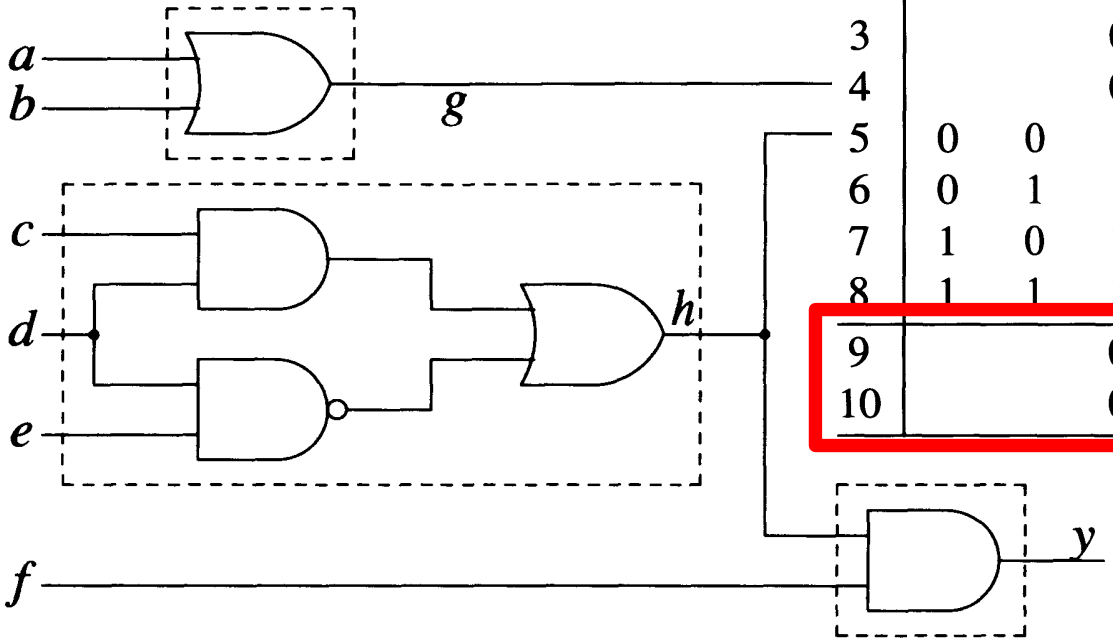
Test vectors for *h* which is observable at *y*

Circuit Partitioning – Vectors for Testing g



	a	b	c	d	e	f	g	h	x	y
1			0	0	0	1		1		1
2			0	0	1	1		1		1
3			0	1	0	1		1		1
4			0	1	1	1		0		0
5	0	0	1	0	0	1	0	1	0	1
6	0	1	1	0	1	1	1	1	1	1
7	1	0	1	1	0	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1
9			0	1	1	0		0		0
10			0	0	0	0		1		0

Circuit Partitioning – Vectors for Testing y



	a	b	c	d	e	f	g	h	x	y
1			0	0	0	1		1		1
2			0	0	1	1		1		1
3			0	1	0	1		1		1
4			0	1	1	1		0		0
5	0	0	1	0	0	1	0	1	0	1
6	0	1	1	0	1	1	1	1	1	1
7	1	0	1	1	0	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1
9			0	1	1	0		0		0
10			0	0	0	0		1		0

Testing Sequential Circuits

- **Fault assumption**: state table modified w/o increasing # of states
  - Add/remove state transitions
- **Problem**: finding input sequences that distinguish a circuit with *n* states from all other *n*-state circuits
  - Such sequences exist for reduced and strong connected sequential circuits

Testing Sequential Circuits – Three Phases

- 1: **initialization** – bring circuit to a known state
- 2: Verify that circuit has *n* states
- 3: Verify that every entry in the state table

Functional Fault Models

- Functional faults represent effects of physical faults on the functions of a system.
  - Behavior due to functional faults should match the behavior due to physical faults.
  - Tests to detect functional faults have high coverage for the SSFs in the structural model.

Example – Addressing Faults

- **Addressing decoding – Functions**
  - Addressing a word in memory
  - Selecting a register in processor
  - Decoding an instruction to determine operations to perform
- **Functional faults**
  - Selecting no item
  - selecting item *i* instead of *j*
  - Selecting item *j* in addition to item *i*.
- Test generation concerns generating a program that produces wrong results.

Summary

- Test if circuit functions are implemented correctly
- Functional Fault Model: effects of physical faults on functional behavior
- Functional testing:
  - Pseudo-exhaustive
  - Test generation w or w/o fault models