CIS 4930 Digital System Testing
Modeling

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

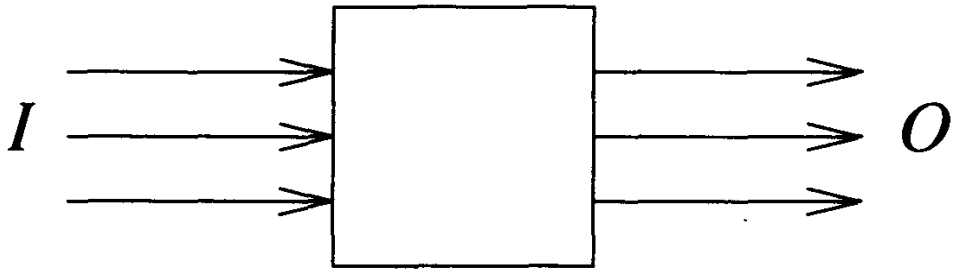
Reading

- Chapter 2

2.1 Basic Concepts

- Functional modeling at logic level
- Functional modeling at register level
- Structural models

Black Box View



- A system can be viewed as a black box i.e., generates **outputs** in response **to inputs.**
- Behavior is defined by the I/O mapping.
  - i.e. value transformation
  - Different value representations at different levels.

Behavioral/Functional Model

- System **Behavior** is defined by
  - I/O Mapping occurs over time
- A **logic function** is a I/O Mapping without timing.
- A **functional model** is a representation of logic function
- A **behavioral model** is
  - A functional Model + A representation of timing relations

Structural Model

- Collection of interconnected smaller boxes called components/elements
- Often hierarchical
  - A component itself represented by a structural model.
- Bottom-level boxes known as **primitive** elements
- Can shown as block Diagrams, or schematics
- In practice structural and functional are inter-mixed.
- Often necessary to describe large systems.

2.2 Functional Modeling at Logic Level

- Truth Tables
- Primitive Cubes
- State Tables
- Flow Tables
- BDDs – Binary Decision Diagrams
- Programs

Truth Tables & Primitive Cubes

x_1	x_2	x_3	Z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(a)

x_1	x_2	x_3	Z
x	1	0	0
1	1	x	0
x	0	x	1
0	x	1	1

(b)

Figure 2.2 Models for a combinational function $Z(x_1, x_2, x_3)$ (a) truth table (b) primitive cubes

Cubical Notation

x_1	x_2	x_3	Z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

- First two rows can be combined

  00x | 1

  where X is a don't care
- A cube of a function $Z(x_1, x_2, x_3)$ has the form

  $(v_1, v_2, v_3 \mid v_Z)$

  where $v_Z = Z(v_1, v_2, v_3)$
- A cube of *Z* can represent multiple rows in a truth table

Cubical Notation

- If cube *q* can be obtained from cube *p* by replacing one or more *x* in *p* by 0 or 1

  then *p* **covers** *q*

  Eg: 00x | 1 covers cubes 000 | 1 and 001 | 1

Implicants

- An **implicant** g of Z is a sub-function such that

  (g = 1) => (Z = 1)

- An implicant "covers" one or more minterms in a truth table.
  - A disjunction of some minterms.
- If an implicant covers at least one row that is not covered by any other implicants, it is known as a **prime implicant.**

Prime Implicant and Primitive Cube

- Cube 00x | 1 represents implicant

$$\overline{x_1}\ \overline{x_2}$$

- A cube that represents a prime implicant is known as **primitive cube.**

Primitive Cube Representation

x_1	x_2	x_3	Z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(a)

x_1	x_2	x_3	Z
x	1	0	0
1	1	x	0
x	0	x	1
0	x	1	1

(b)

Compact Representation of function Z

Figure 2.2 Models for a combinational function $Z(x_1, x_2, x_3)$ (a) truth table (b) primitive cubes

Intersection Operator

$\cap$	0	1	x
0	0	$\varnothing$	0
1	$\varnothing$	1	1
x	0	1	x

Figure 2.3 Intersection operator

x_1	x_2	x_3	Z
x	1	0	0
1	1	x	0
x	0	x	1
0	x	1	1

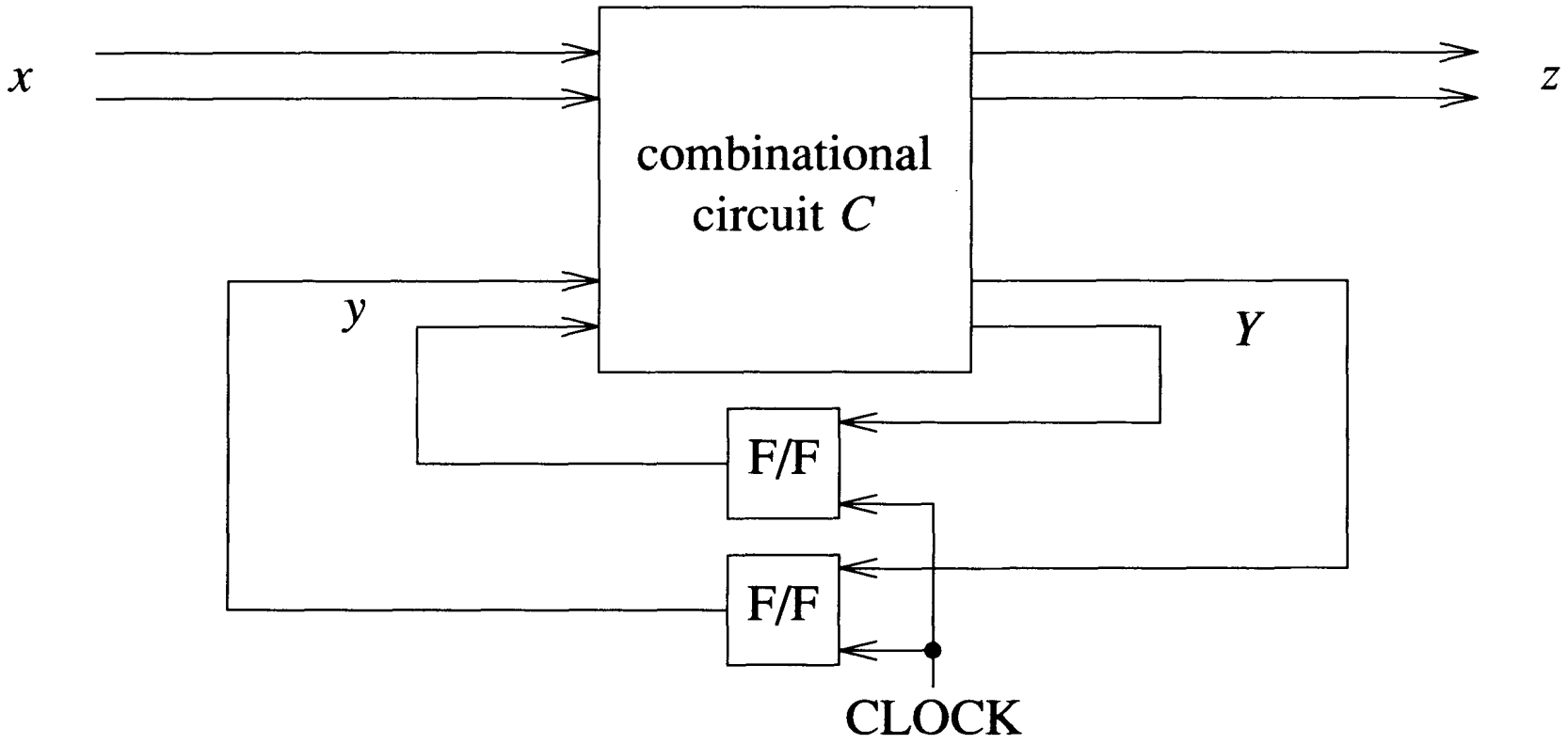
Find Z for 001

- x10 $\cap$ 001 -- Inconsistent
- 11x $\cap$ 001 -- Inconsistent
- x0x $\cap$ 001 -- Consistent
- 0x1 $\cap$ 001 -- Consistent

Output is Z = 1

Modeling Synchronous Sequential Circuit

- Canonical structure of synchronous circuits



State Table Representation

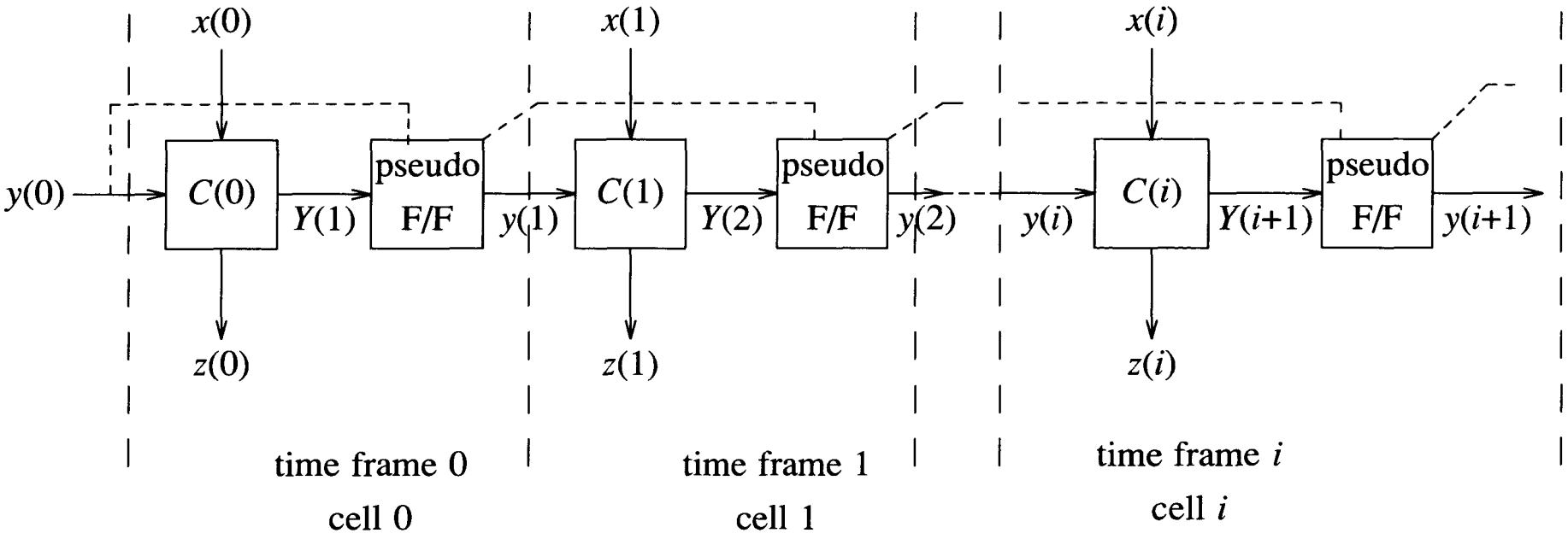
- FSM – Finite State Machine can be represented by a state table
- Inputs are sampled at regular intervals and next state and output computed

		x = 0	x = 1
q	1	2,1	3,0
	2	2,1	4,0
	3	1,0	4,0
	4	3,1	3,0

$N(q,x), Z(q,x)$

Pseudo-combinational -- Iterative Array

- Unroll seq circuit → pseudocombinational cir.



Asynchronous Circuit

- Represented by a flow table
- Single input change can lead to multiple state changes until a stable configuration is reached
- Stable configurations are shown in **bold** font

	x_1x_2 = 00	01	11	10
1	**1**,0	5,1	2,0	**1**,0
2	1,0	**2**,0	**2**,0	5,1
3	**3**,1	2,0	4,0	**3**,0
4	3,1	5,1	**4**,0	**4**,0
5	3,1	**5**,1	4,0	**5**,1

Figure 2.7 A flow table

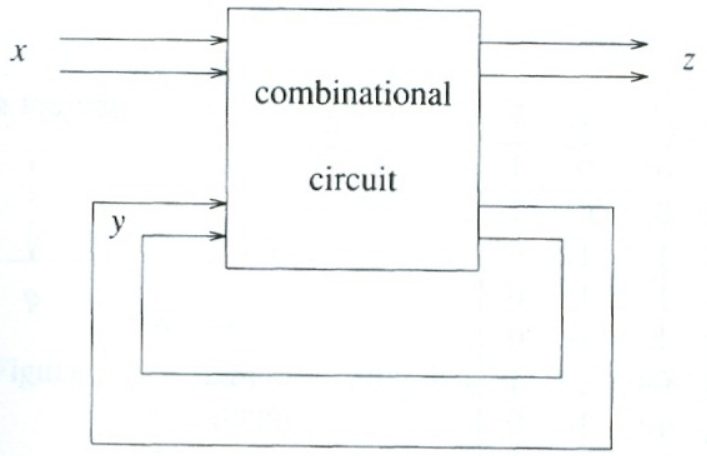


Figure 2.8 Canonical structure of an asynchronous sequential circuit

Binary Decision Diagrams (BDDs)

- A graph representation of circuit functions

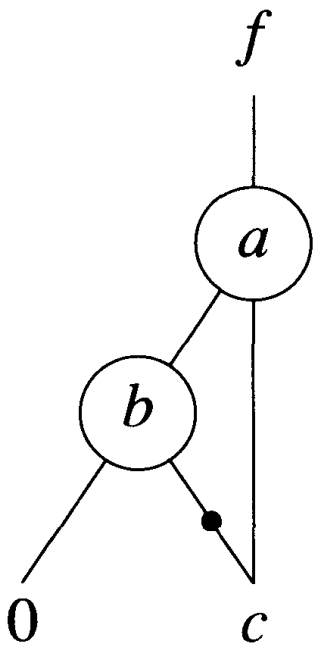
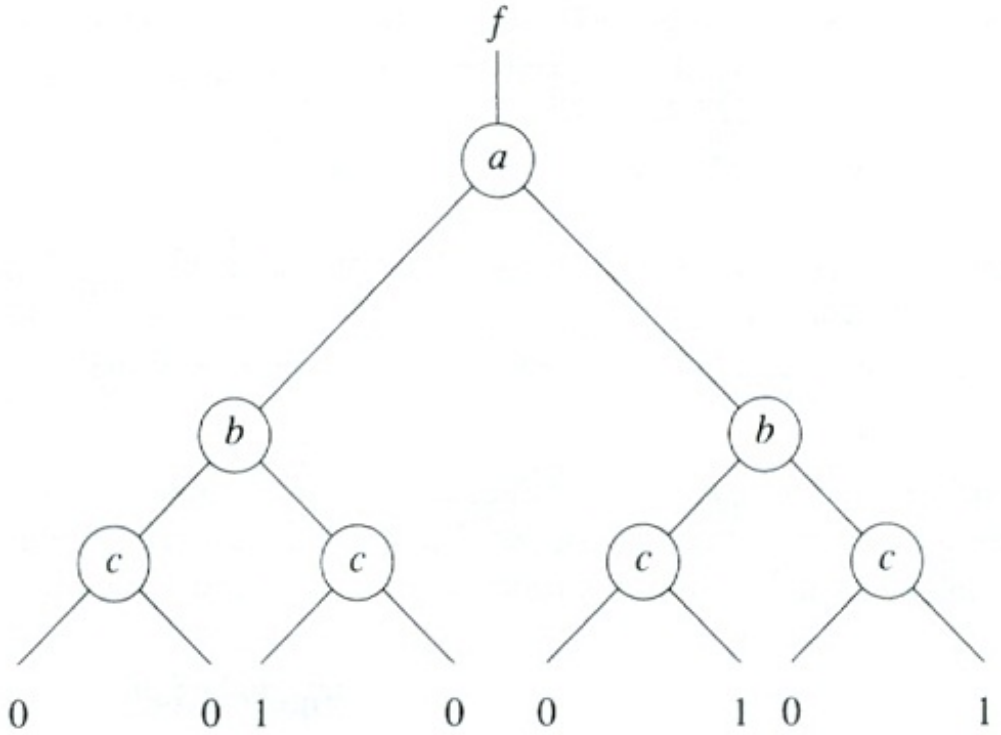


Figure 2.11 Binary decision diagram of $f = \bar{a}b\bar{c} + ac$

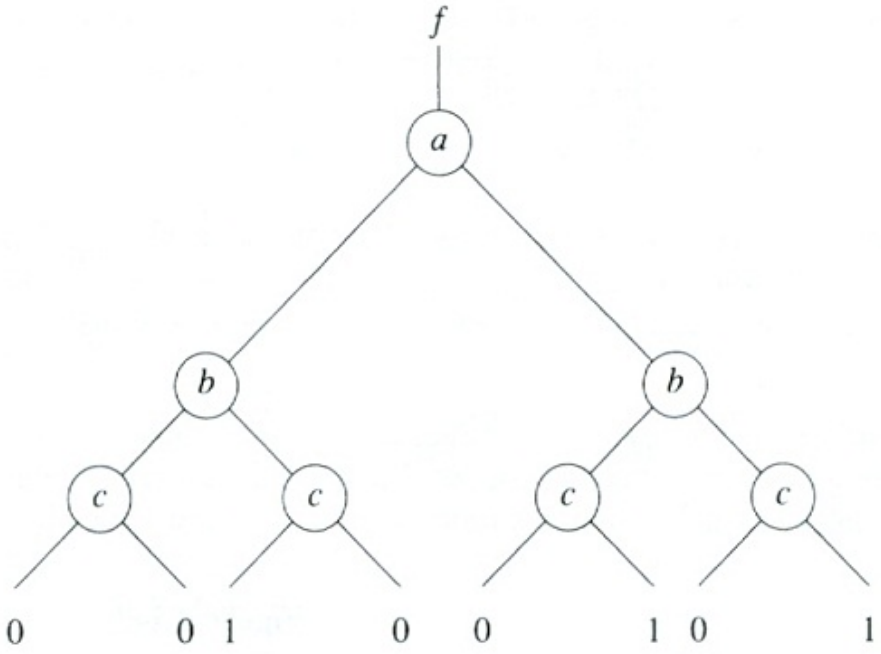
Constructing BDD

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

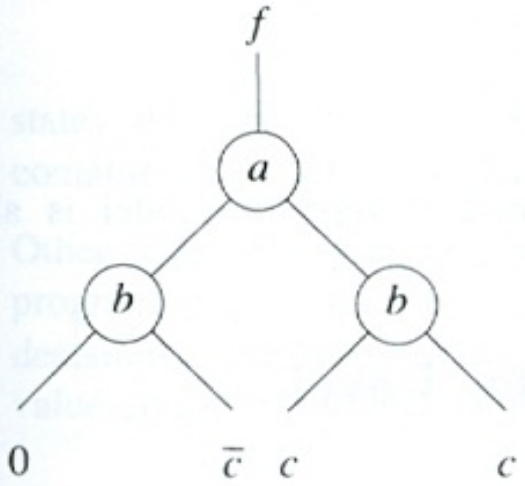
(a)



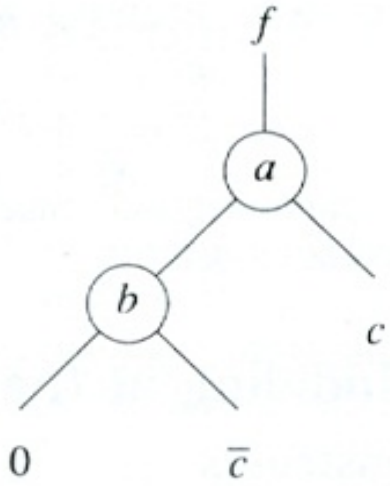
(b)



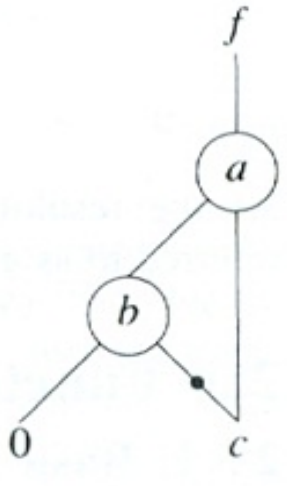
(b)



(c)



(d)



(e)

BDD for a JK FF

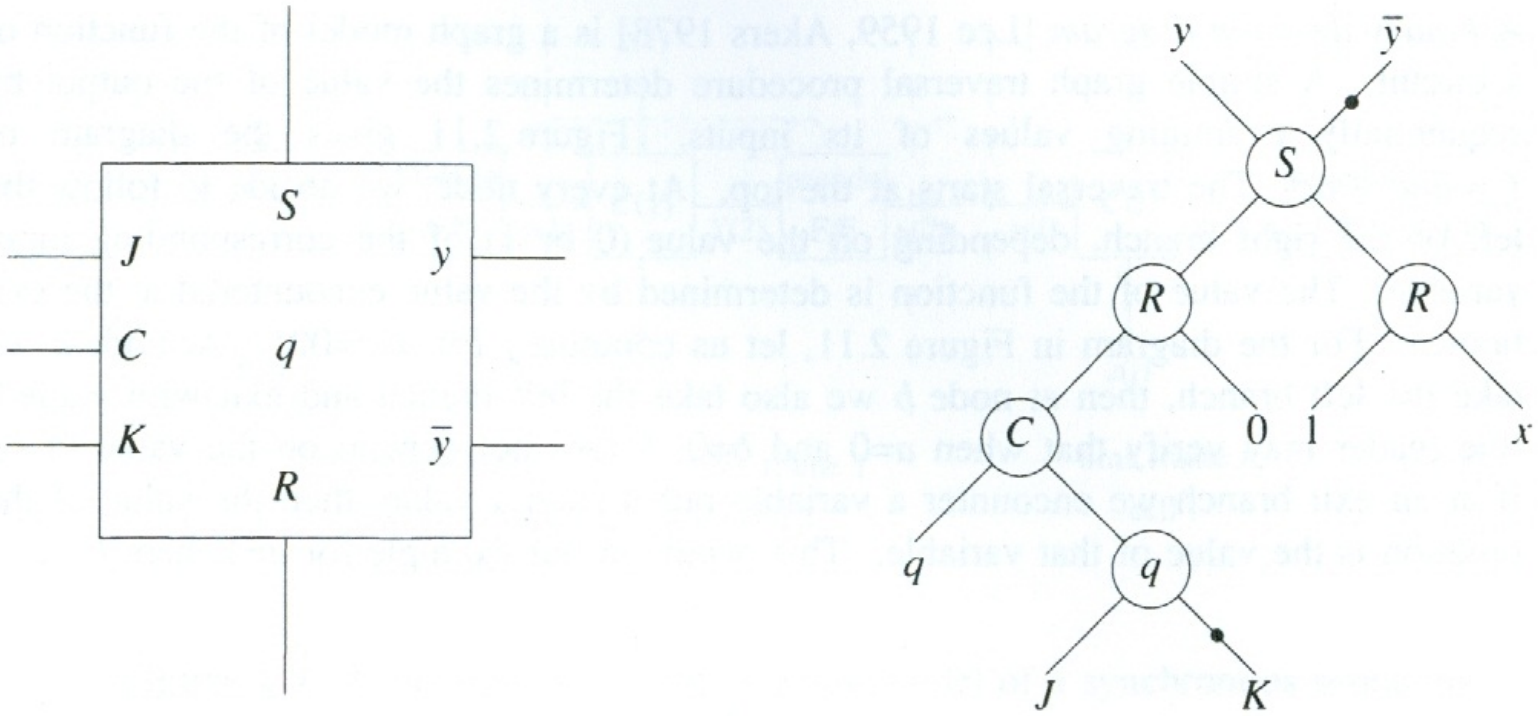


Figure 2.12 Binary decision diagram for a *JK* F/F

Program as Functional Model

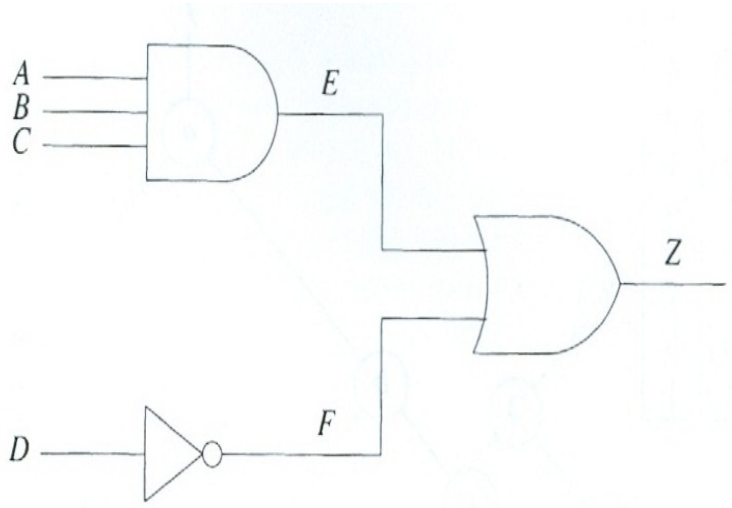


Figure 2.14

Assembly Program

```
LDA  A  /* load acc with A */
AND B  /* compute A.B    */
AND C  /* compute A.B.C  */
STA  E  /* store partial result */
LDA  D  /* load acc with D */
INV     /* compute not D */
OR   E  /* compute A.B.C + ~(D) */
STA  Z  /* store results */
```

C Program

```
E = A & B & C ;
F = ~ D;
Z = E | F ;
```

"Compiled Code Model"

2.3 Functional Modeling at Register Level

- System model at register and instruction set levels
- Storage declarations:

  register *IR [0 → 7]* *//8-bit register*

  memory *ABC [0 →255; 0 → 15]* *// 256 word memory*
- Data paths implicitly defined with operations on registers.
- Data processing by primitive operators:

  C = A + B // add two registers A and B

RTL constructs

- Control flow:
  - **if** X **then** C = A + B
  - **if (** *CLOCK* ***and*** *(AREG < BREG))* ***then*** *AREG = BREG*
  - **test** (*IR [0* $\rightarrow$ *3])*

    case 0*: operation0*

    case 1: *operation1*

    :

    testend

FSM modeling

- One block per state
- Only one state is active/current at any time

```
state  S1, S2, S3  /* state register */
S1:  if X then
        begin
              P = Q + R
              go to S2
        end
     else
              P = Q – R
              go to  S3
S2: …
```

Procedural and Non-procedural RTLs

- Procedural RTLs (sequential semantics, ex. C)
  - Use a sequential programming language
  - implicit cycle-based timing model (i.e., assures that the state of the model accurately reflects state of processor at the end of a cycle)
- Non-procedural RTLs (concurrent semantics, ex. VHDL)
  - Statements conceptually executed in parallel
- Example: A = B

  C = A
  - if the above is procedural code, register C gets value B after execution
  - If the above is non-procedural code, then C gets old value of A

Timing Modeling

- Explicitly specify delay to add more details.
  - Make a model more accurate.
- Examples:

```
// delay updating C by 100.
C  = A + B, delay = 100


// delay every update to C by 100.
delay C 100
```

2.4 Structural Models

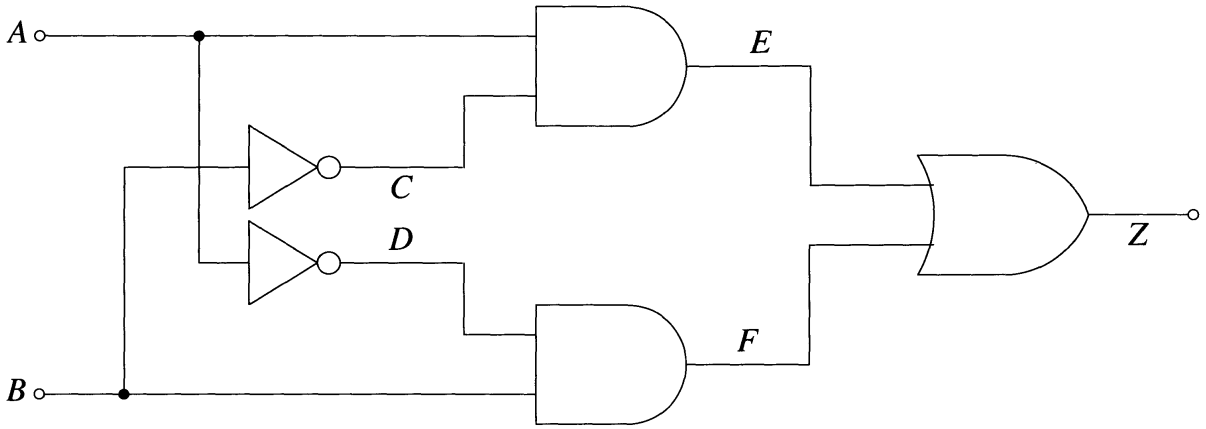


Figure 2.15 Schematic diagram of an XOR circuit

```
CIRCUIT XOR
INPUTS = A,B
OUTPUTS = Z

NOT   D, A
NOT   C, B
AND   E, (A, C)
AND   F, (B, D)
OR    Z, (E, F)
```

Connections are implicitly done via name matching.

Fanout-Free Circuit

- If a gate drives more than one gate then it has a **fanout**.
- A fanout-free circuit can be represented as a tree

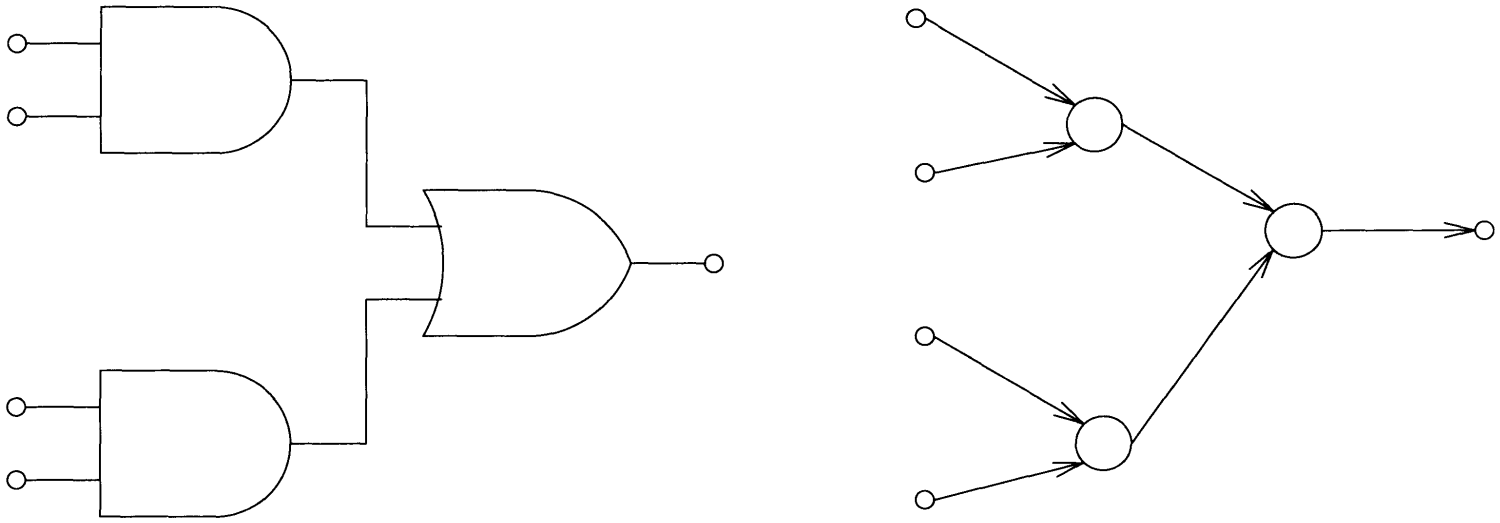


Figure 2.17 Fanout-free circuit and its graph (tree) representation

Reconvergent Fanout

- A signal fans out into two or more paths which later converge at a gate.
- Common in real circuits
- Makes fault detection problem more difficult

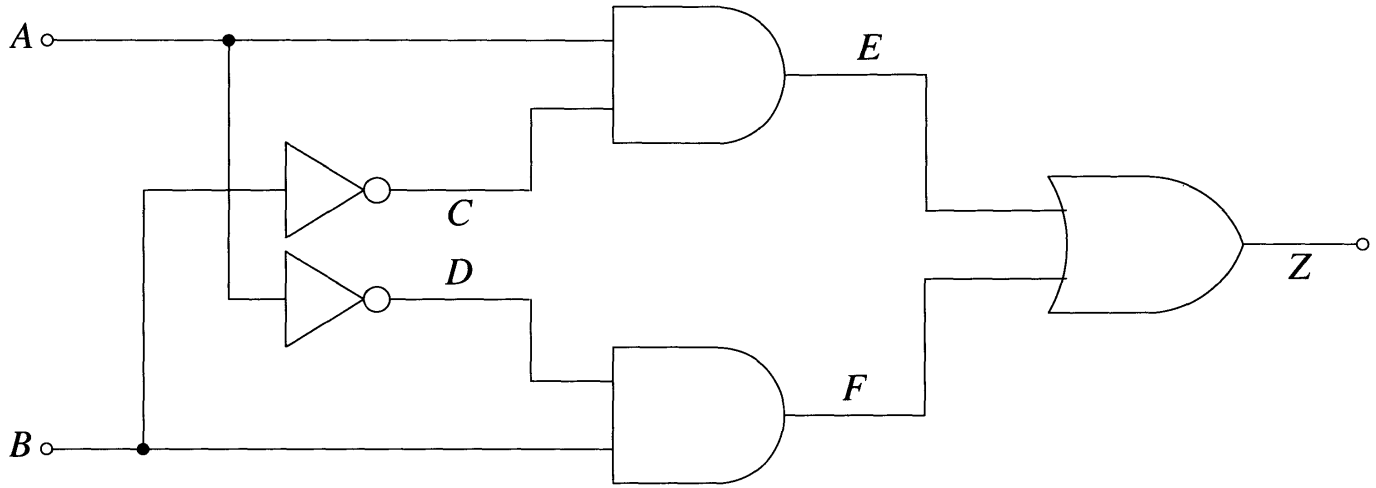
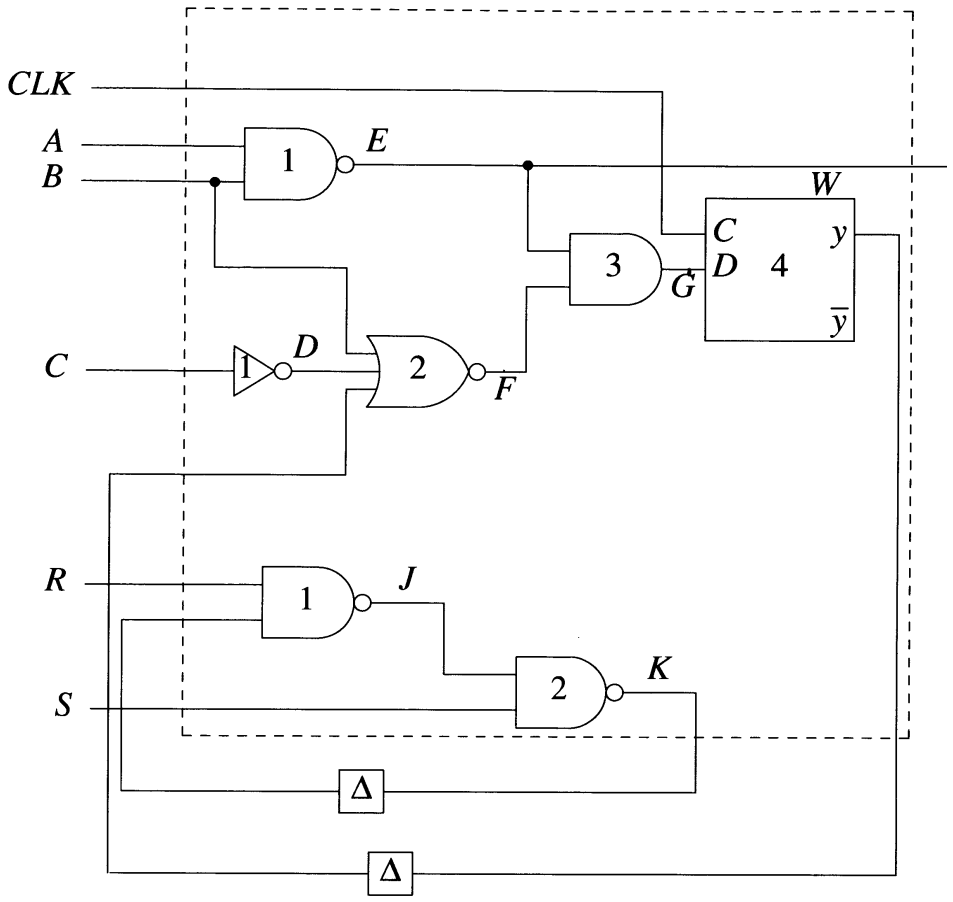


Figure 2.15 Schematic diagram of an XOR circuit

Logic Levels

- Logic level of a gate is it's distance from primary inputs
- Can be computed in a breadth-first manner



Summary

- Digital system model
  - representation of the actual system
  - can be done at logic and RT-levels
- Behavioral model = Functional Model + Timing
- Models for combinational circuits: truth table, primitive cubes, BDDs, programs (compiled-code models)
- Sequential circuits: state tables, RT level case statement
- Reconvergent Fanout – signal branches and later converges on a gate
- Non-procedural language used for RTL modeling to mimic concurrent execution of the hardware

Backup

Fanout Terminology

- Stem and fanout branches

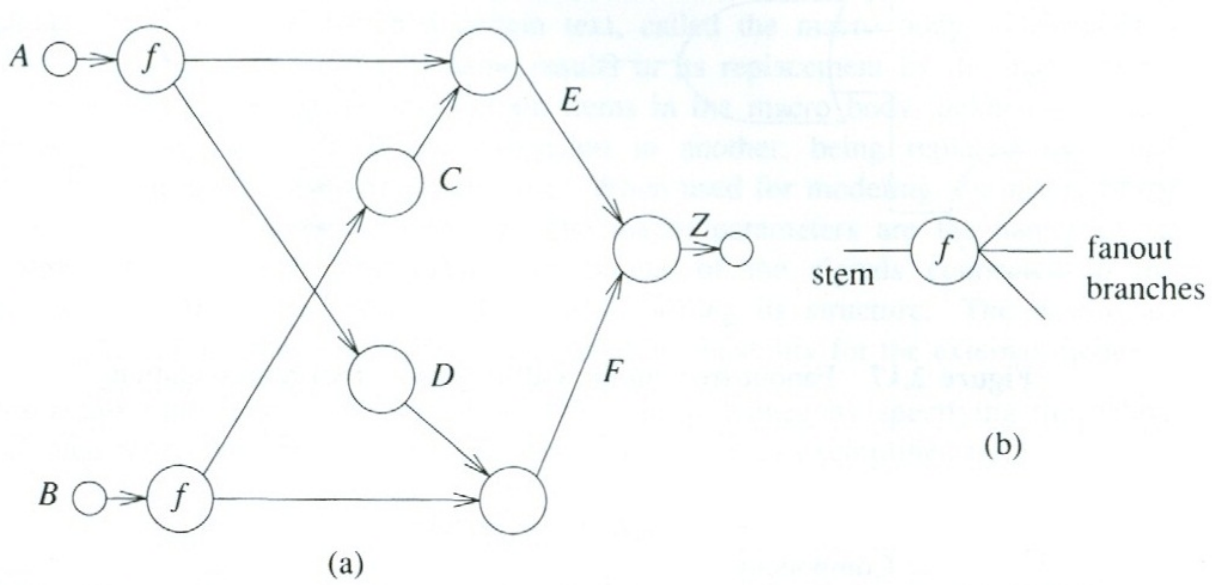


Figure 2.19 (a) Graph model for the circuit of Figure 2.15 (b) A fanout node