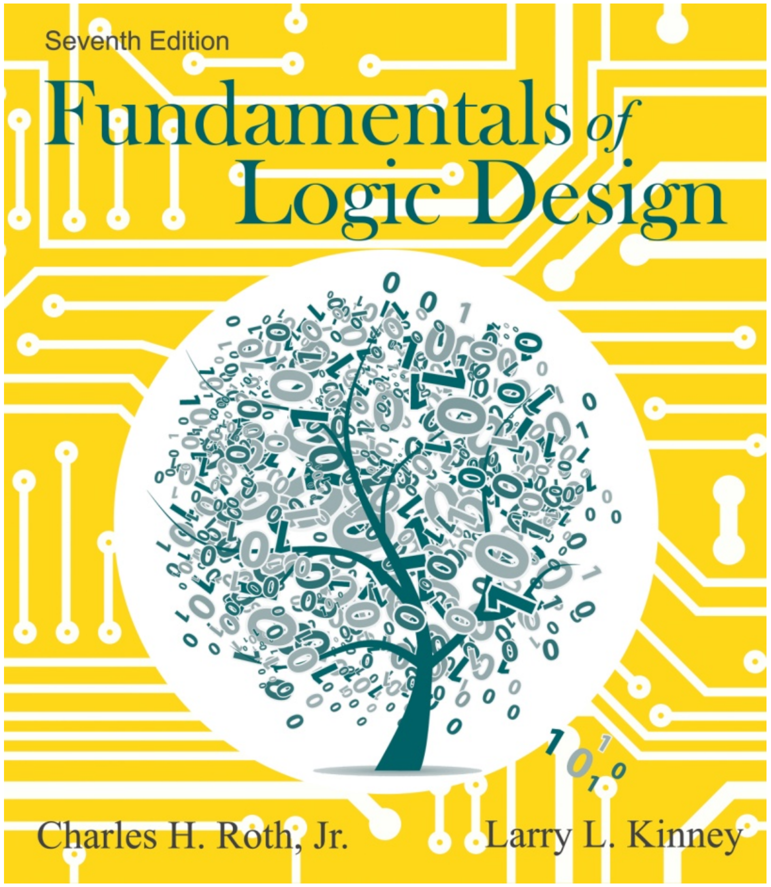


UNIT 16

Sequential Circuit Design

This chapter includes:

Learning Objectives

1. Design a sequential circuit using gates and flip-flops.
2. Test your circuit by simulating it and by implementing it in lab.
3. Design a unilateral iterative circuit. Explain the relationship between iterative and sequential circuits, and convert from one to the other.
4. Show how to implement a sequential circuit using a ROM or PLA and flip-flops.
5. Explain the operation of CPLDs and FPGAs and show how they can be used to implement sequential logic.

Summary of Design Procedure for Sequential Circuits

Design Procedure:

1. Given the problem statement, determine the required relationship between the input and output sequences and derive a state table. For many problems, it is easiest to first construct a state graph.

2. Reduce the table to a minimum number of states. First, eliminate duplicate rows by row matching and, then, form an implication table and follow the procedure in Section 15.3.

Summary of Design Procedure for Sequential Circuits

Design Procedure (continued):

3. If the reduced table has *m* states ($2^{n-1} < m <= 2^n$), *n* flip-flops are required. Assign a unique combination of flip-flop states to correspond to each state in the reduced table. The guidelines given in Section 15.8 may prove helpful in finding an assignment which leads to an economical circuit.

4. Form the transition table by substituting the assigned flip-flop states for each state in the reduced state table. The resulting transition table specifies the next states of the flip-flops, and the output in terms of the present states of the flip-flops and the input.

Summary of Design Procedure for Sequential Circuits

Design Procedure (continued):

5. Plot next-state maps and input maps for each flip-flop and derive the flip-flop input equations. (Depending on the type of gates to be used, either determine the sum-of-products form from the 1's on the map or the product-of-sums form from the 0's on the map.) Derive the output functions.

6. Realize the flip-flop input equations and the output equations using the available logic gates.

7. Check your design by signal tracing, computer simulation, or laboratory testing.

Design Example- Code Converter

BCD to excess-3 code converter:

- This circuit adds three to a binary-coded-decimal digit in the range 0 to 9.
- Table 16-1 lists the desired inputs and outputs at times t_0, t_1, t_2, and t_3.

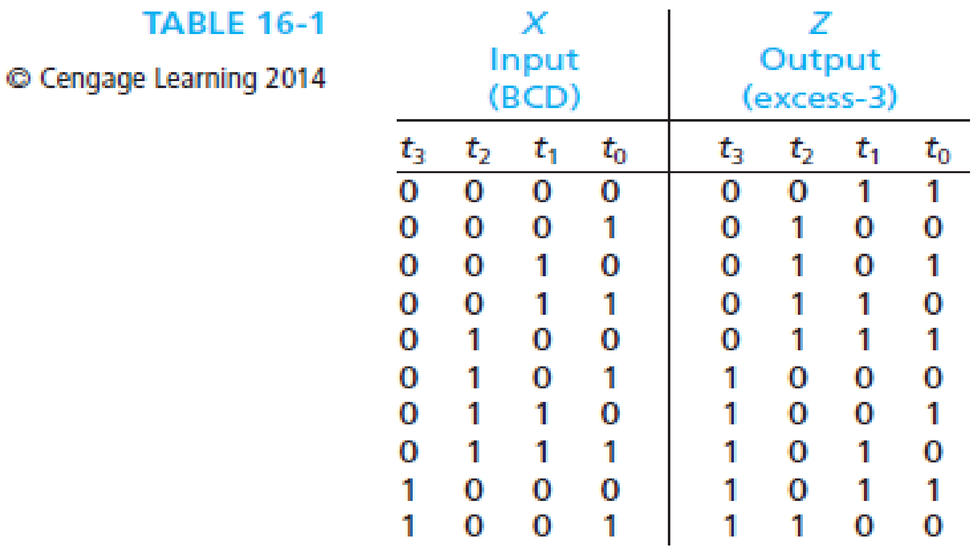


TABLE 16-1

© Cengage Learning 2014

X Input (BCD)				Z Output (excess-3)			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Design Example- Code Converter

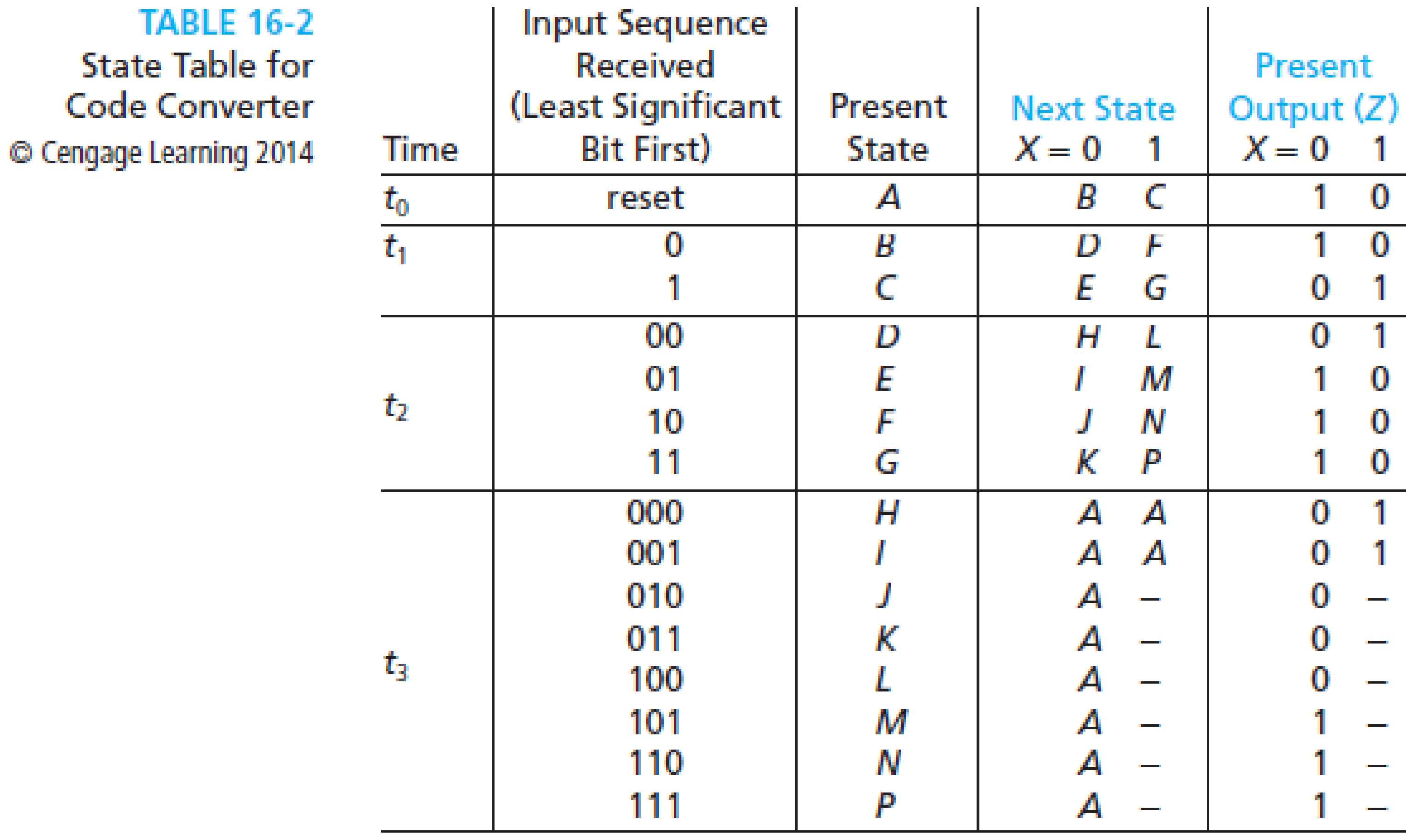


TABLE 16-2
State Table for Code Converter
© Cengage Learning 2014

Time	Input Sequence Received (Least Significant Bit First)	Present State	Next State $X = 0$	Next State $X = 1$	Present Output (Z) $X = 0$	Present Output (Z) $X = 1$
t_0	reset	A	B	C	1	0
t_1	0	B	D	F	1	0
	1	C	E	G	0	1
t_2	00	D	H	L	0	1
	01	E	I	M	1	0
	10	F	J	N	1	0
	11	G	K	P	1	0
t_3	000	H	A	A	0	1
	001	I	A	A	0	1
	010	J	A	–	0	–
	011	K	A	–	0	–
	100	L	A	–	0	–
	101	M	A	–	1	–
	110	N	A	–	1	–
	111	P	A	–	1	–

Design Example- Code Converter

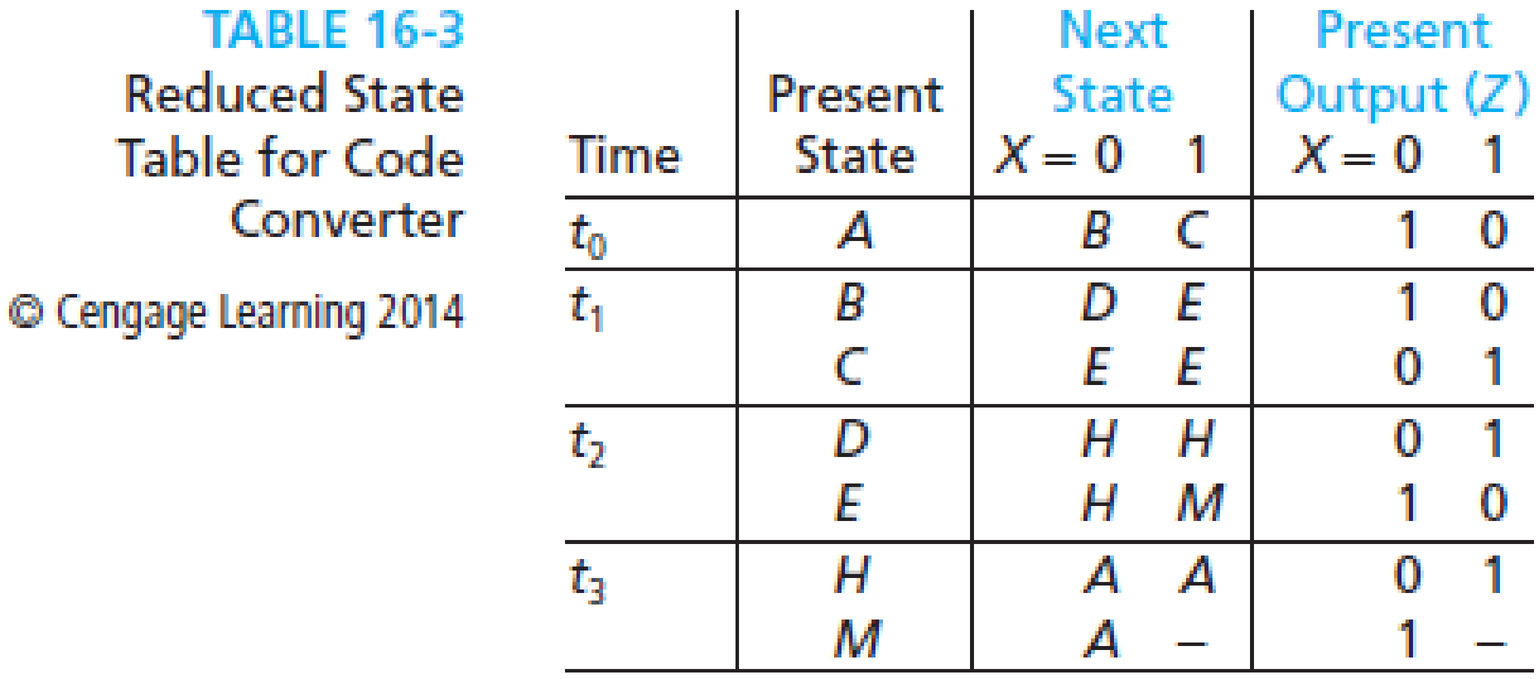


TABLE 16-3 Reduced State Table for Code Converter

© Cengage Learning 2014

Time	Present State	Next State $X = 0$	Next State $X = 1$	Present Output (Z) $X = 0$	Present Output (Z) $X = 1$
t_0	A	B	C	1	0
t_1	B	D	E	1	0
	C	E	E	0	1
t_2	D	H	H	0	1
	E	H	M	1	0
t_3	H	A	A	0	1
	M	A	–	1	–

Design Example- Code Converter

State Graphs For Code Converter:

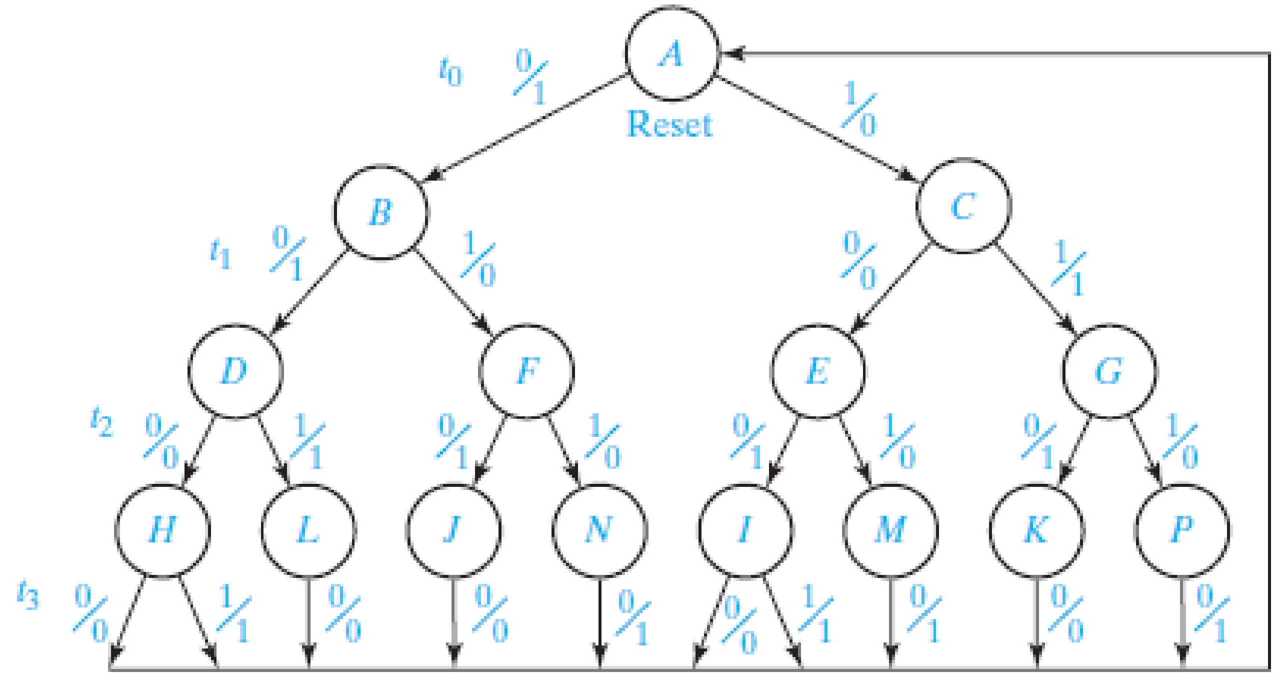
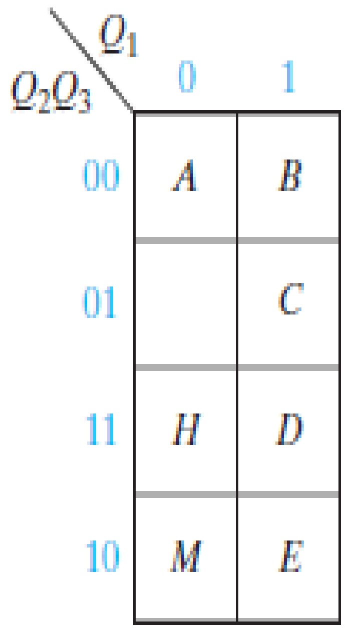


FIGURE 16-1 State Graph for Code Converter

Design Example- Code Converter

FIGURE 16-2
Assignment Map and Transition Table for Flip-Flops

© Cengage Learning 2014



(a) Assignment map

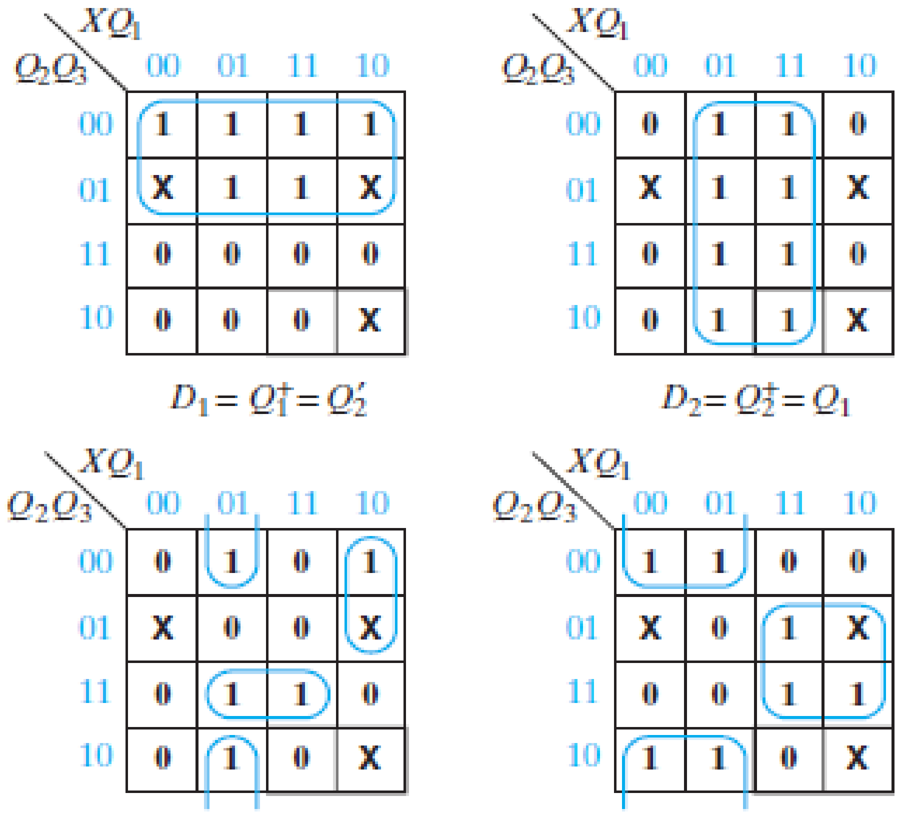
	$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ $X = 0$	$X = 1$	Z $X = 0$	$X = 1$
A	0 0 0	1 0 0	1 0 1	1	0
B	1 0 0	1 1 1	1 1 0	1	0
C	1 0 1	1 1 0	1 1 0	0	1
D	1 1 1	0 1 1	0 1 1	0	1
E	1 1 0	0 1 1	0 1 0	1	0
H	0 1 1	0 0 0	0 0 0	0	1
M	0 1 0	0 0 0	x x x	1	x
–	0 0 1	x x x	x x x	x	x

(b) Transition table

Design Example- Code Converter

FIGURE 16-3 Karnaugh Maps for Code Converter Design

© Cengage Learning 2014



$D_1 = Q_1^+ = Q_2'$

$D_2 = Q_2^+ = Q_1$

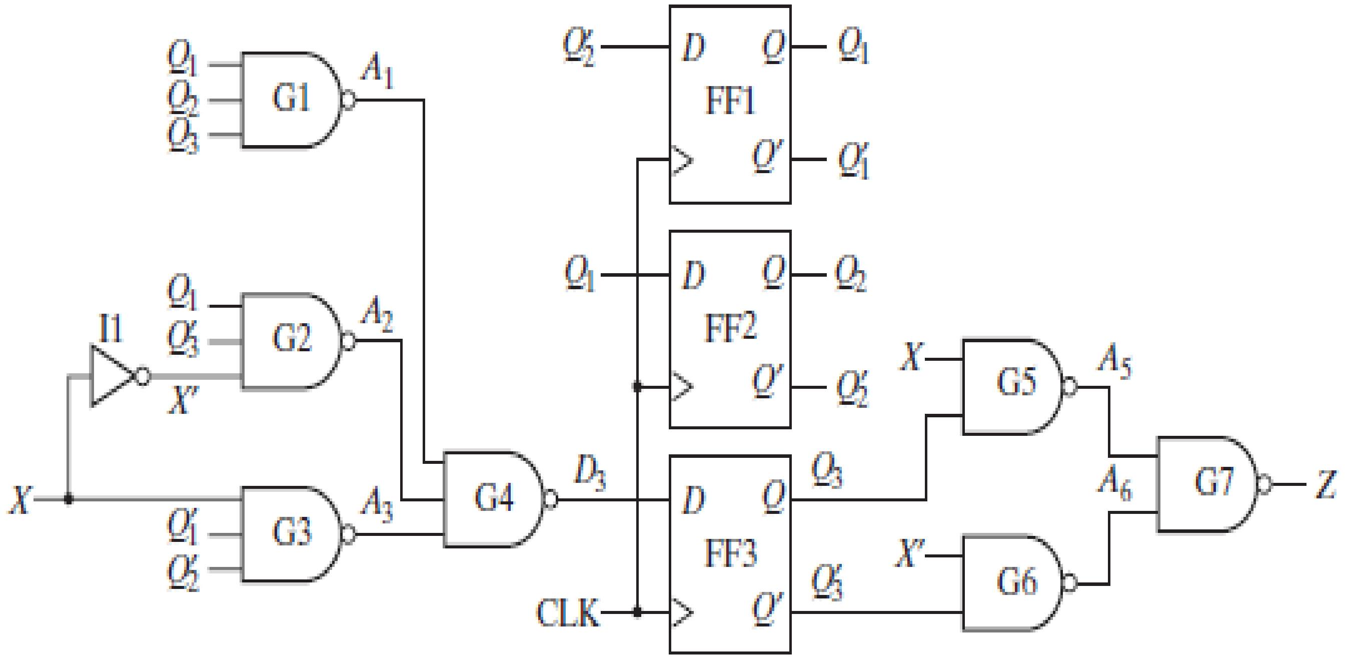
$D_3 = Q_3^+ = Q_1 Q_2 Q_3 + X' Q_1 Q_3' + X Q_1' Q_2'$

$Z = X' Q_3' + X Q_3$

Design Example- Code Converter

FIGURE 16-4
Code Converter Circuit

© Cengage Learning 2014



Design of Iterative Circuits

Iterative Circuits:

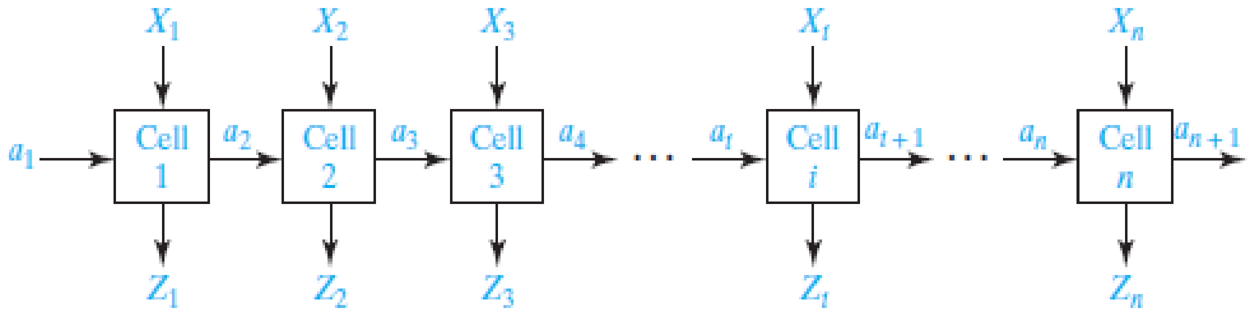
❖An iterative circuit consists of a number of identical cells interconnected in a regular manner.

❖Some operations, such as binary addition, naturally lend themselves to realization with an iterative circuit because the same operation is performed on each pair of input bits.

❖The simplest form of an iterative circuit consists of a linear array of combinational cells with signals between cells traveling in only one direction (Figure 16-5).

FIGURE 16-5 Unilateral Iterative Circuit

© Cengage Learning 2014

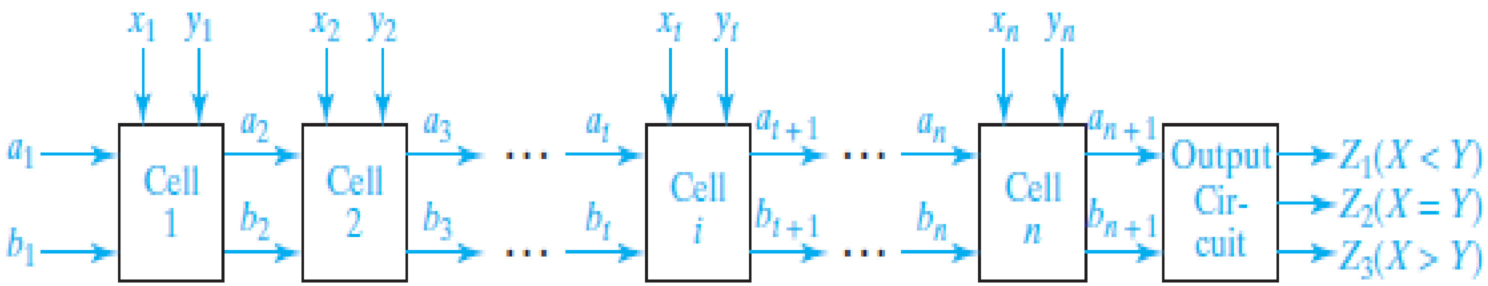


Design of Iterative Circuits

Design of a Comparator:

- We will design a circuit which compares two n-bit binary numbers and determines if they are equal or which one is larger if they are not equal.
- Designate the two binary numbers to be compared as $X = x_1x_2 \dots x_n$ and $Y = y_1y_2 \dots y_n$

FIGURE 16-6 Form of Iterative Circuit for Comparing Binary Numbers



Design of Iterative Circuits

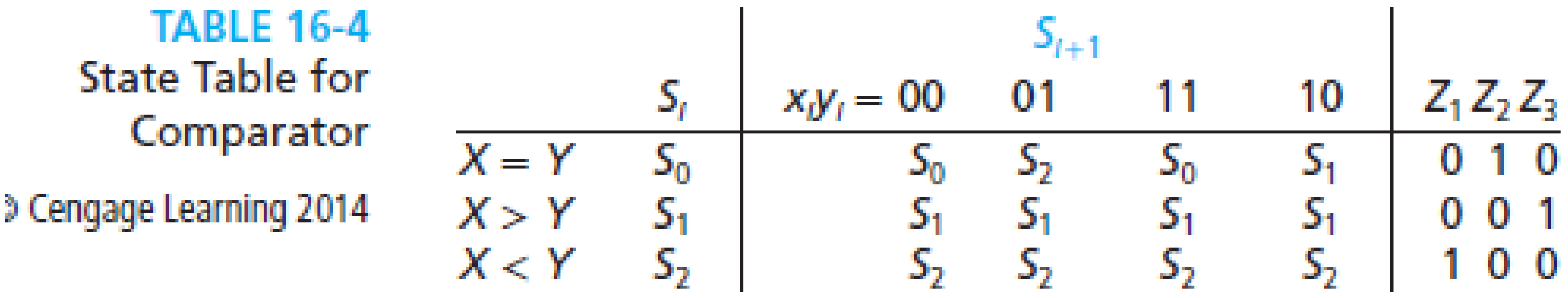


TABLE 16-4 State Table for Comparator

	S_i	S_{i+1}				$Z_1 Z_2 Z_3$
		$x_i y_i = 00$	01	11	10	
$X = Y$	S_0	S_0	S_2	S_0	S_1	0 1 0
$X > Y$	S_1	S_1	S_1	S_1	S_1	0 0 1
$X < Y$	S_2	S_2	S_2	S_2	S_2	1 0 0

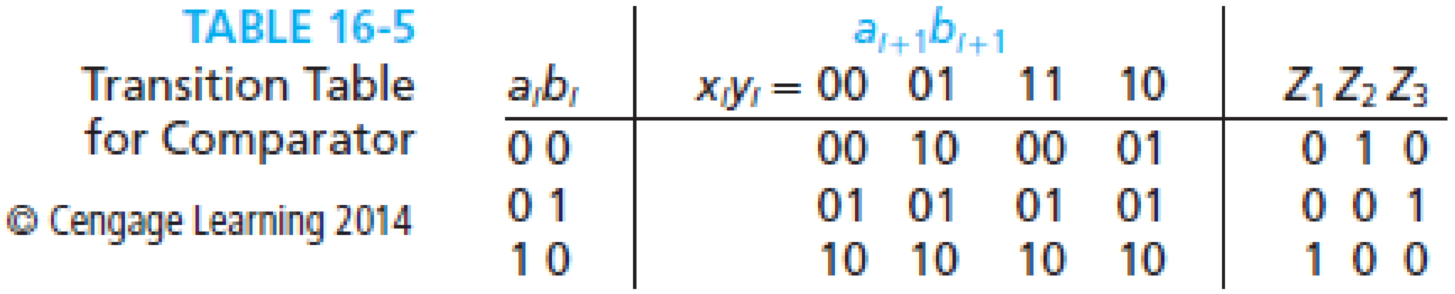
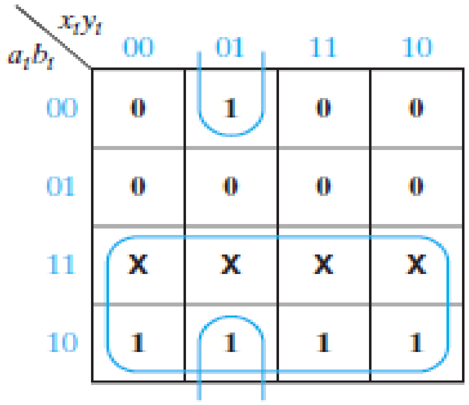


TABLE 16-5 Transition Table for Comparator

$a_i b_i$	$a_{i+1} b_{i+1}$				$Z_1 Z_2 Z_3$
	$x_i y_i = 00$	01	11	10	
0 0	00	10	00	01	0 1 0
0 1	01	01	01	01	0 0 1
1 0	10	10	10	10	1 0 0

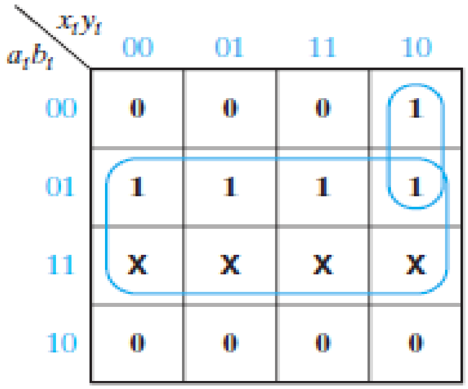
Design of Iterative Circuits

FIGURE 16-7
Typical Cell for Comparator



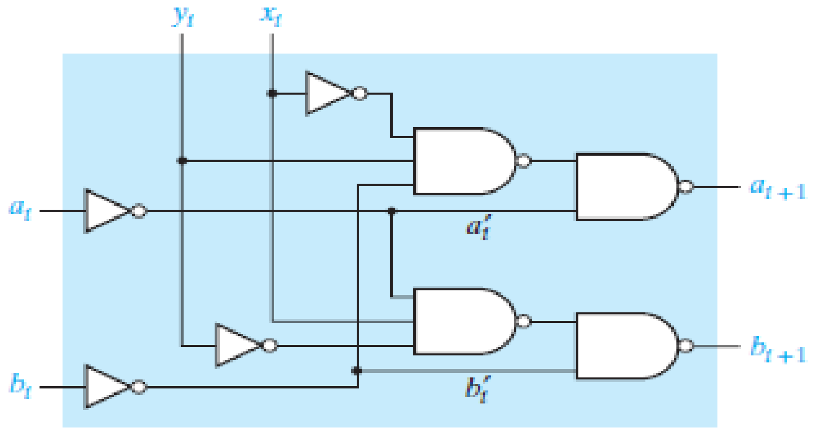
$a_t b_t \backslash x_t y_t$	00	01	11	10
00	0	1	0	0
01	0	0	0	0
11	X	X	X	X
10	1	1	1	1

$a_{t+1} = a_t + x_t' y_t b_t'$



$a_t b_t \backslash x_t y_t$	00	01	11	10
00	0	0	0	1
01	1	1	1	1
11	X	X	X	X
10	0	0	0	0

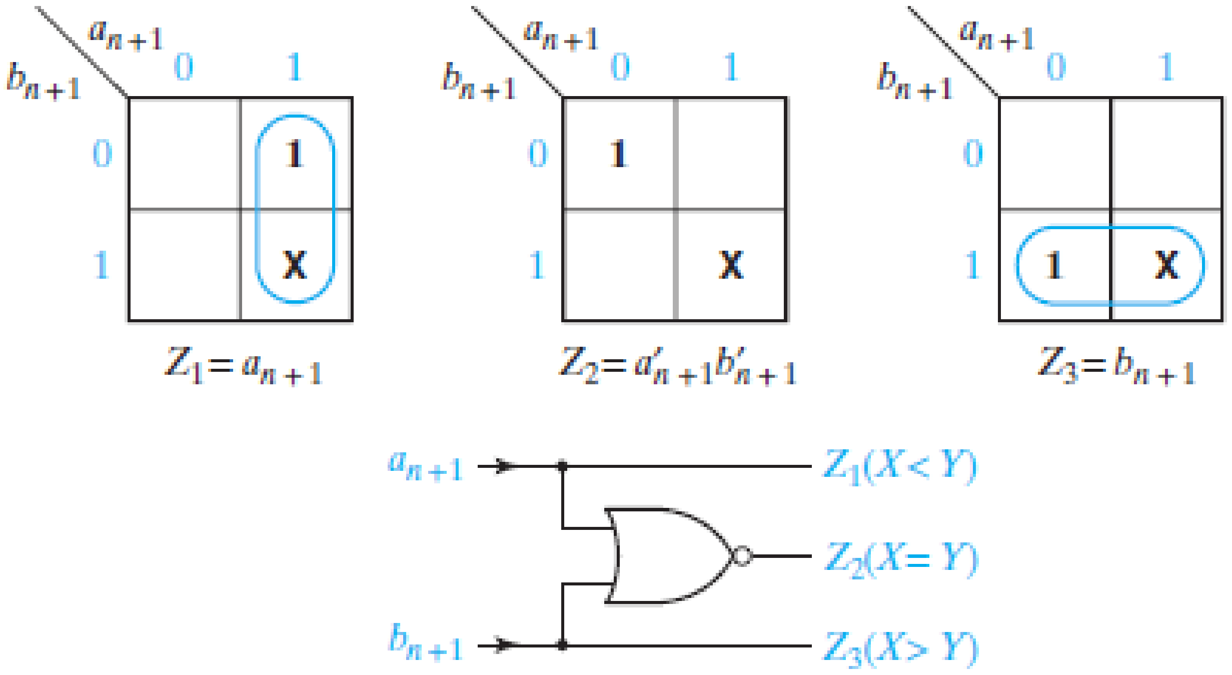
$b_{t+1} = b_t + x_t y_t' a_t'$



Design of Iterative Circuits

FIGURE 16-8
Output Circuit for Comparator

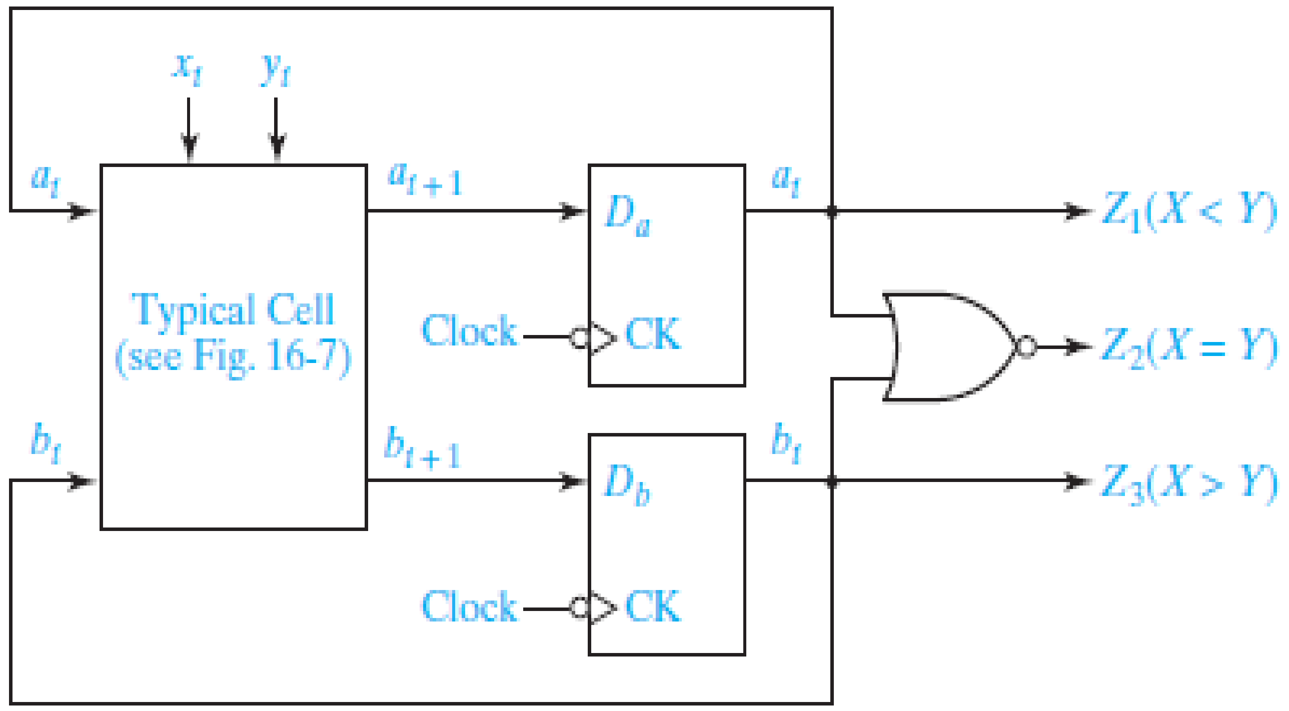
© Cengage Learning 2014



Design of Iterative Circuits

FIGURE 16-9 Sequential Comparator for Binary Numbers

© Cengage Learning 2014



Design of Sequential Circuits Using ROMs and PLAs

Code Converter Using ROM and D Flip-Flops:

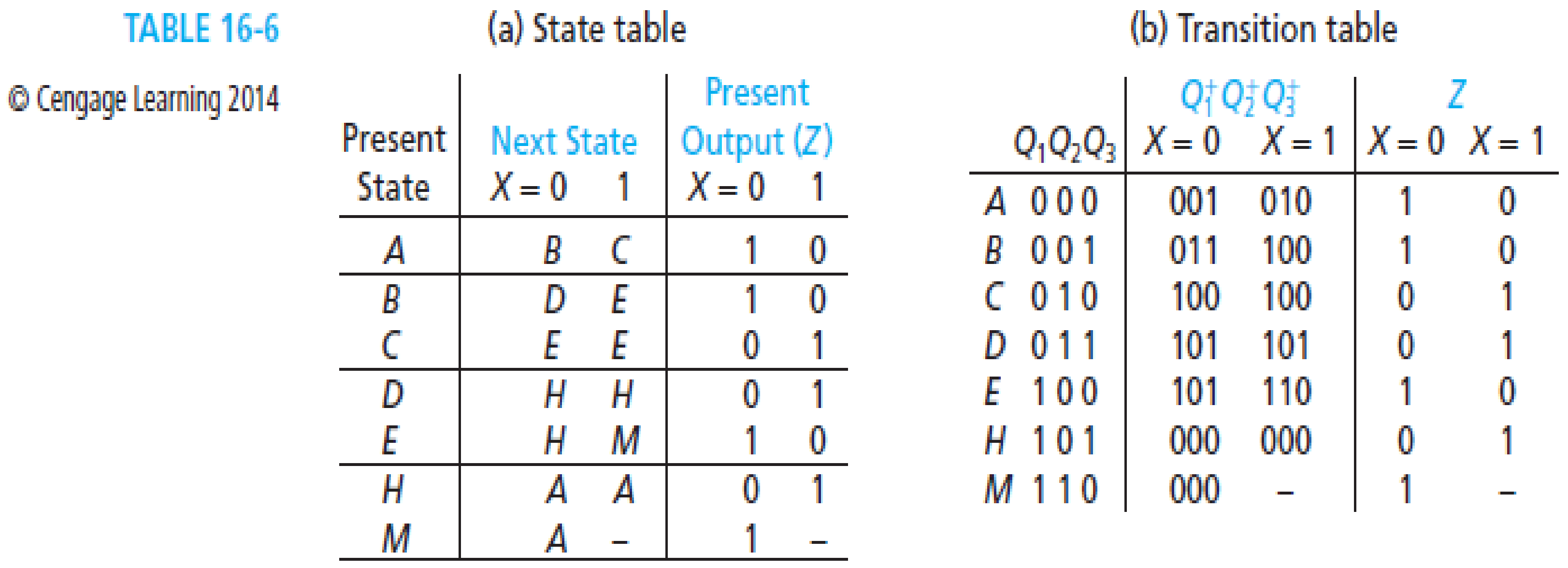


TABLE 16-6

© Cengage Learning 2014

(a) State table

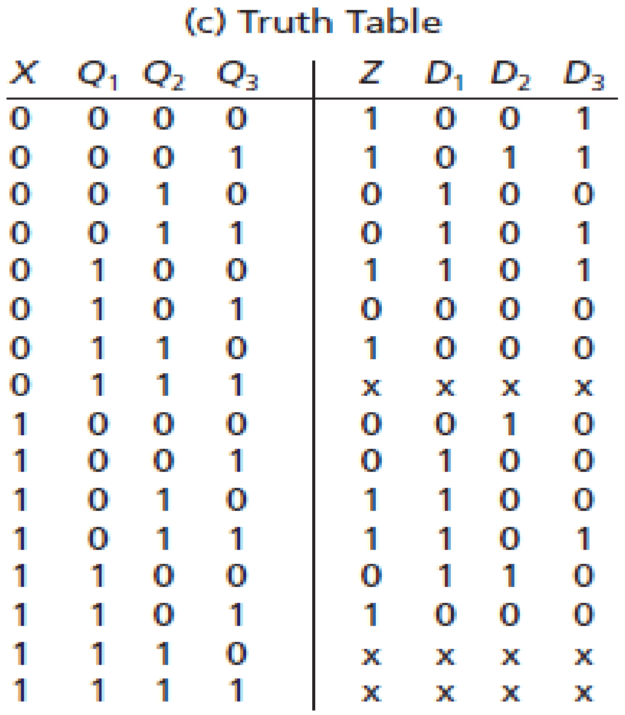
Present State	Next State $X = 0$	1	Present Output (Z) $X = 0$	1
A	B	C	1	0
B	D	E	1	0
C	E	E	0	1
D	H	H	0	1
E	H	M	1	0
H	A	A	0	1
M	A	–	1	–

(b) Transition table

	$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ $X = 0$	$X = 1$	Z $X = 0$	$X = 1$
A	0 0 0	001	010	1	0
B	0 0 1	011	100	1	0
C	0 1 0	100	100	0	1
D	0 1 1	101	101	0	1
E	1 0 0	101	110	1	0
H	1 0 1	000	000	0	1
M	1 1 0	000	–	1	–

Design of Sequential Circuits Using ROMs and PLAs

Code Converter Using ROM and D Flip-Flops (continued):


(c) Truth Table

X	Q_1	Q_2	Q_3	Z	D_1	D_2	D_3
0	0	0	0	1	0	0	1
0	0	0	1	1	0	1	1
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	1
0	1	0	0	1	1	0	1
0	1	0	1	0	0	0	0
0	1	1	0	1	0	0	0
0	1	1	1	x	x	x	x
1	0	0	0	0	0	1	0
1	0	0	1	0	1	0	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	0	1	1	0
1	1	0	1	1	0	0	0
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

Design of Sequential Circuits Using ROMs and PLAs

Code Converter Using ROM and D Flip-Flops (continued):

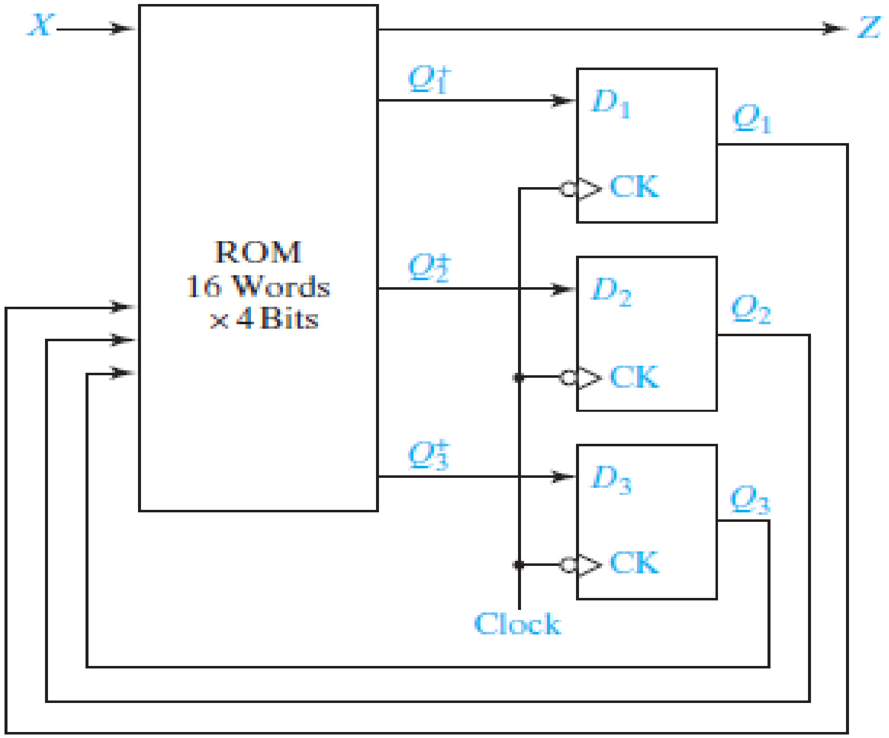


FIGURE 16-10 Realization of Table 16-6(a) Using a ROM

Design of Sequential Circuits Using ROMs and PLAs

Converter Circuit Using PLAs and D Flip-Flops:

- Same circuit configuration as Figure 16-10 but ROM is replaced with a PLA of appropriate size.
- Input equations: PLA Table:

$$D_1 = Q_1^+ = Q_2'$$

$$D_2 = Q_2^+ = Q_1$$

$$D_3 = Q_3^+ = Q_1Q_2Q_3 + X'Q_1Q_3' + XQ_1'Q_2'$$

$$Z = X'Q_3' + XQ_3$$

TABLE 16-7

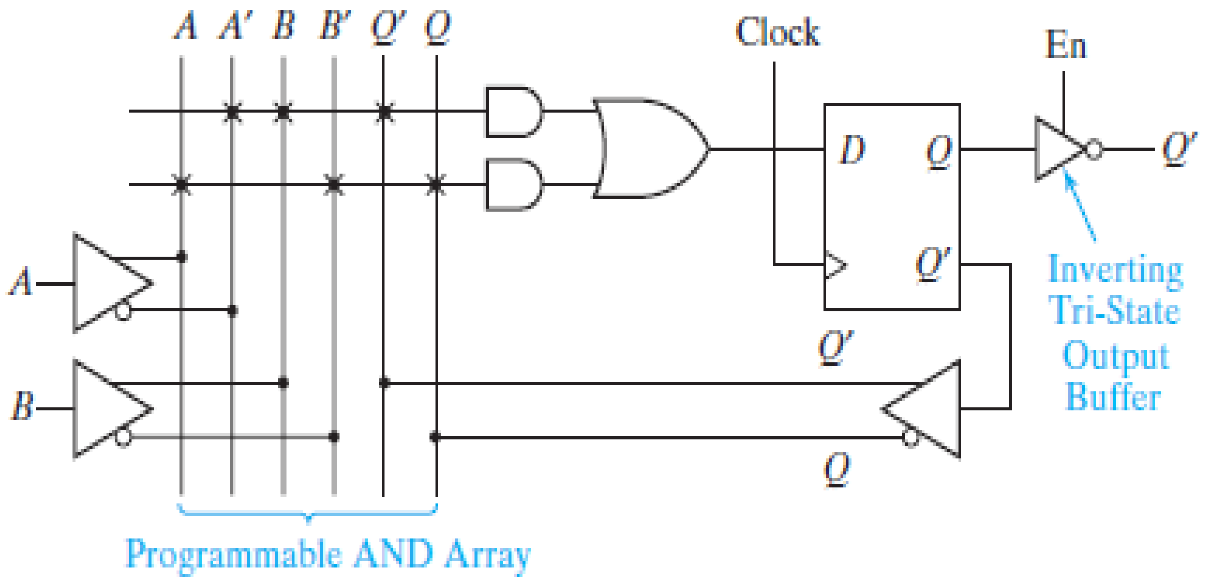
X	Q_1	Q_2	Q_3	Z	D_1	D_2	D_3
–	–	0	–	0	1	0	0
–	1	–	–	0	0	1	0
–	1	1	1	0	0	0	1
0	1	–	0	0	0	0	1
1	0	0	–	0	0	0	1
0	–	–	0	1	0	0	0
1	–	–	1	1	0	0	0

Design of Sequential Circuits Using ROMs and PLAs

Realization of Sequential Circuits By PAL:

FIGURE 16-11 Segment of a Sequential PAL

© Cengage Learning 2014



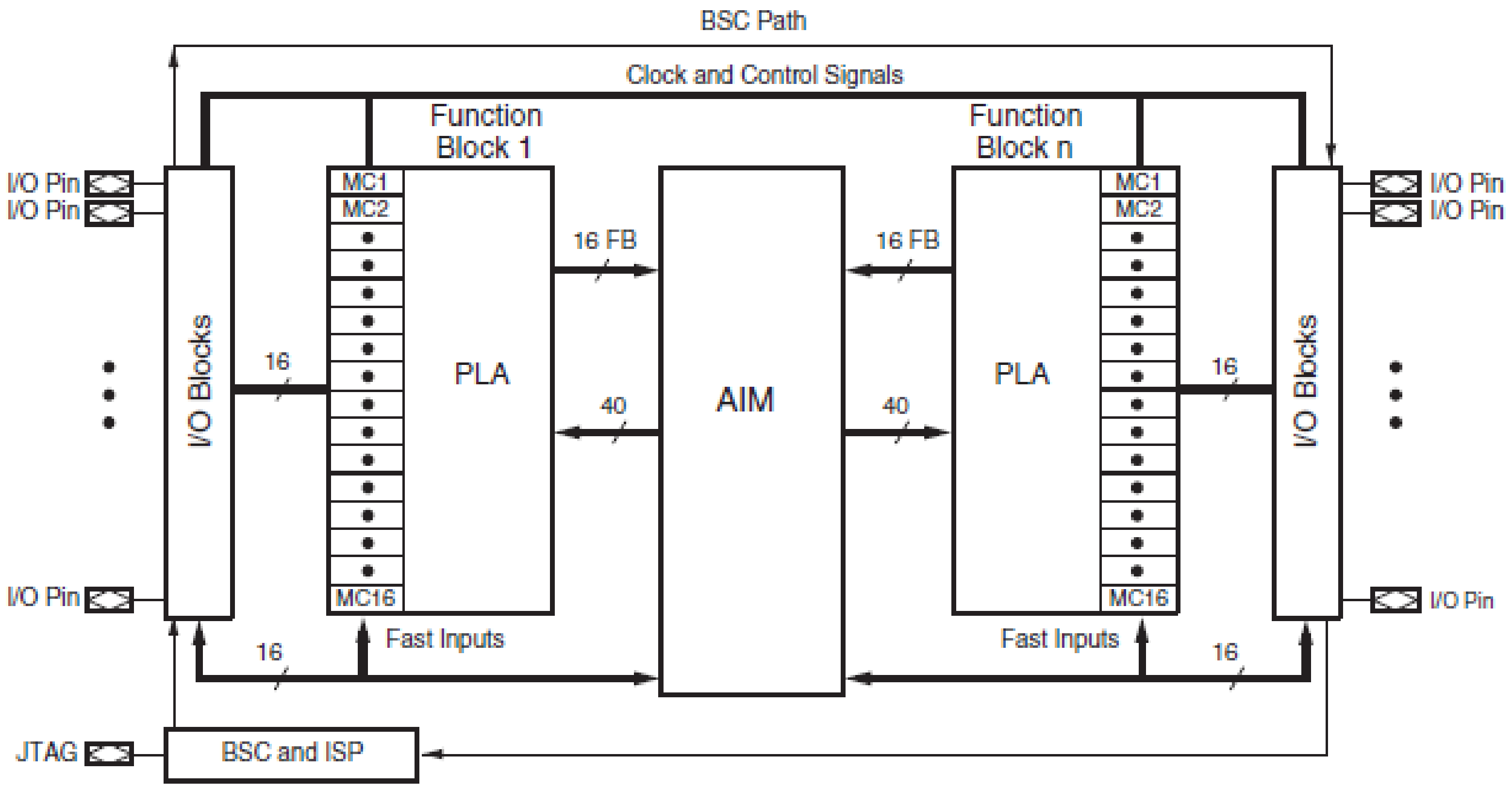
Sequential Circuit Design Using CPLDs

CPLD Sequential Circuit:

- A typical CPLD contains a number of macrocells that are grouped into function blocks, which are connected through an interconnection array.
- Each macrocell contains a flip-flop and an OR gate, which has its inputs connected to an AND gate array.
- Some CPLDs are based on PALs, in which case each OR gate has a fixed set of AND gates associated with it.
- Other CPLDs are based on PLAs, in which case any AND gate output within a function block can be connected to any OR gate input in that block.

Sequential Circuit Design Using CPLDs

FIGURE 16-12 CoolRunner-II Architecture[1] (Figure based on figures and text owned by Xilinx, Inc., Courtesy of Xilinx, Inc. © Xilinx, Inc. 1999–2003. All rights reserved.)



Sequential Circuit Design Using CPLDs

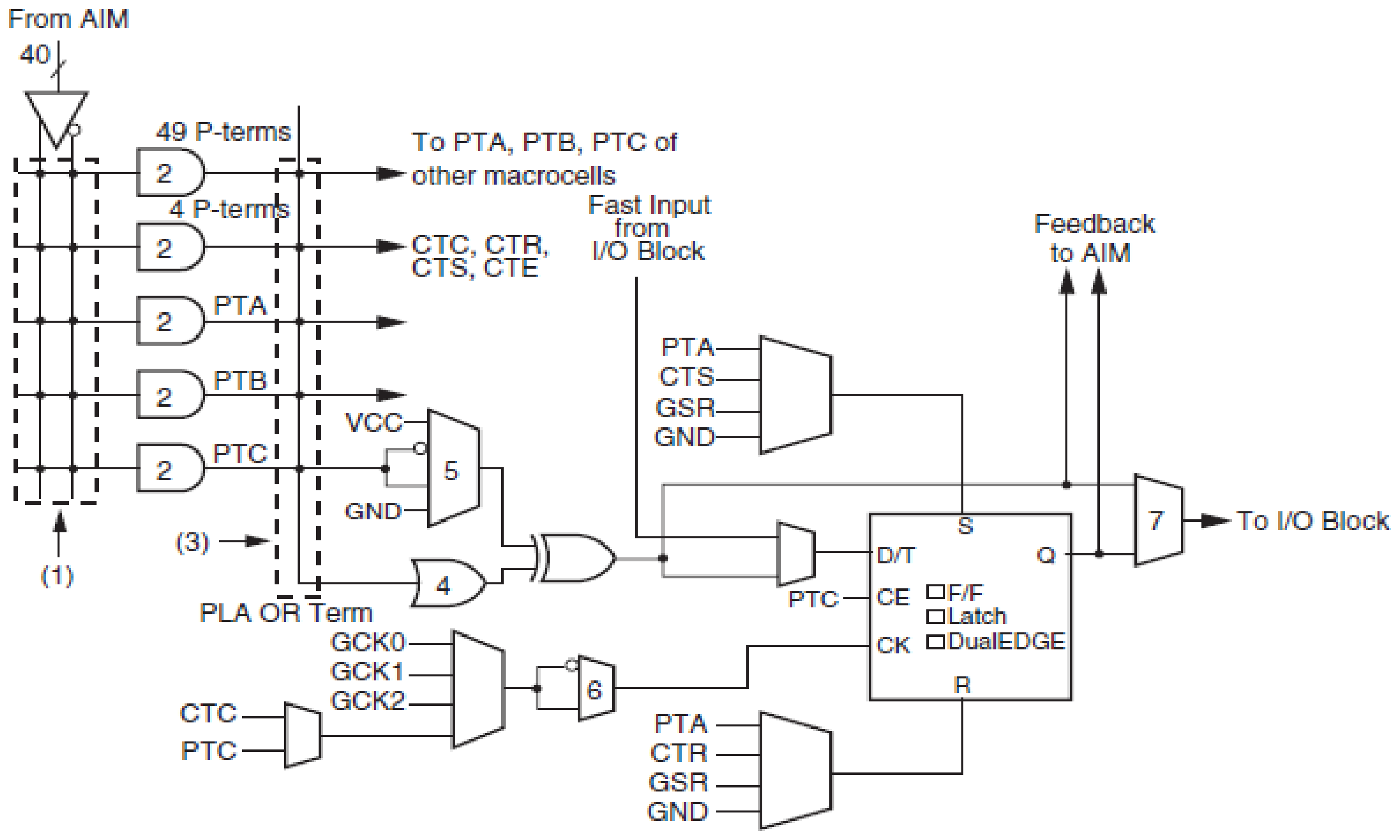
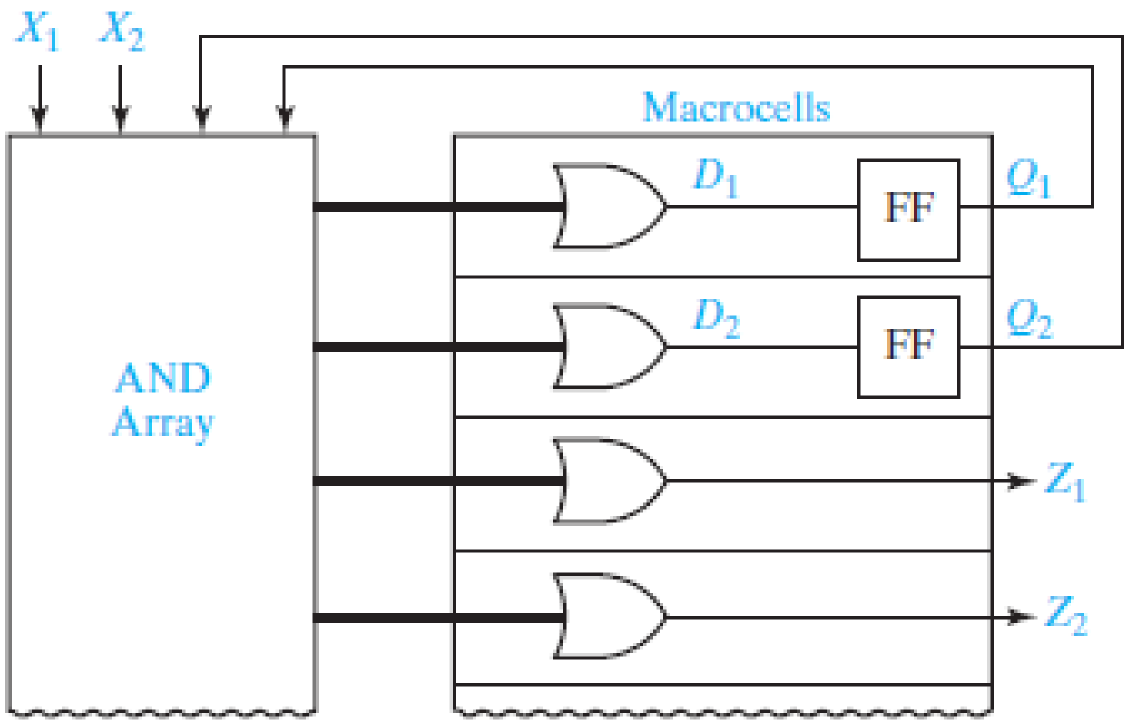


FIGURE 16-13 CoolRunner-II Macrocell (Figure based on figures and text owned by Xilinx, Inc., Courtesy of Xilinx, Inc. © Xilinx, Inc. 1999–2003. All rights reserved.)

Sequential Circuit Design Using CPLDs

CPLD Mealy Machine Implementation:

FIGURE 16-14 CPLD Implementation of a Mealy Machine

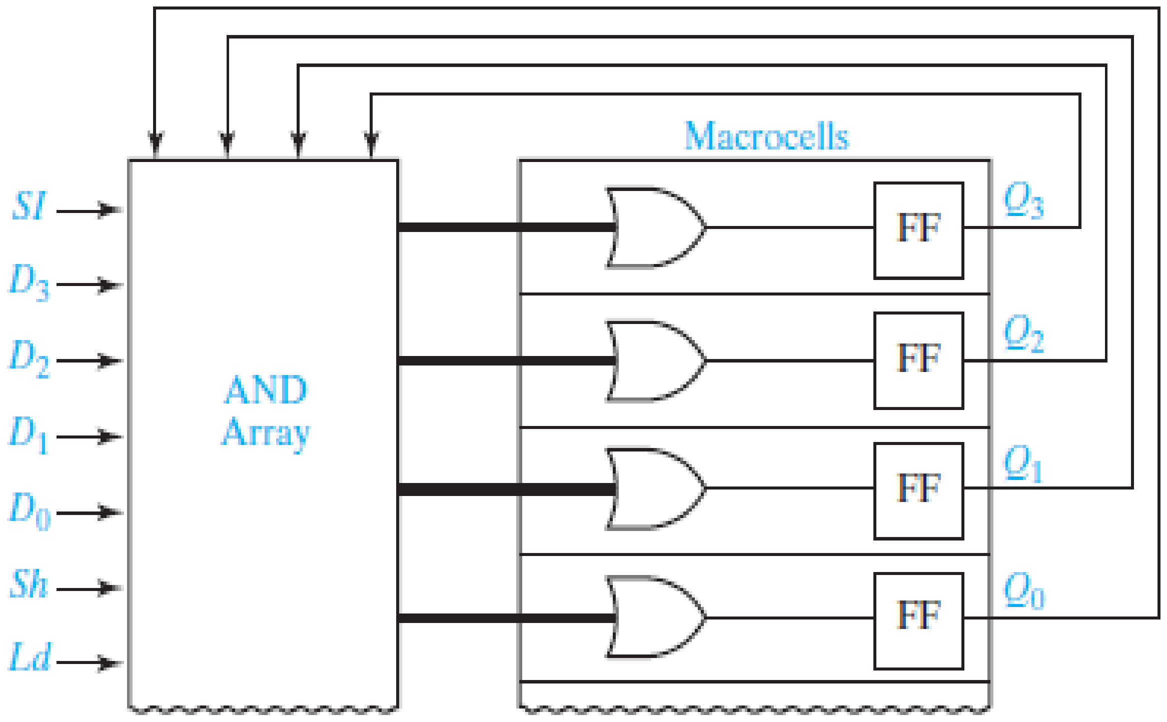


Sequential Circuit Design Using CPLDs

CPLD Implementation of a Shift Register:

FIGURE 16-15 CPLD Implementation of a Shift Register

© Cengage Learning 2014



Sequential Circuit Design Using CPLDs

CPLD Implementation of a Parallel Adder With Accumulator:

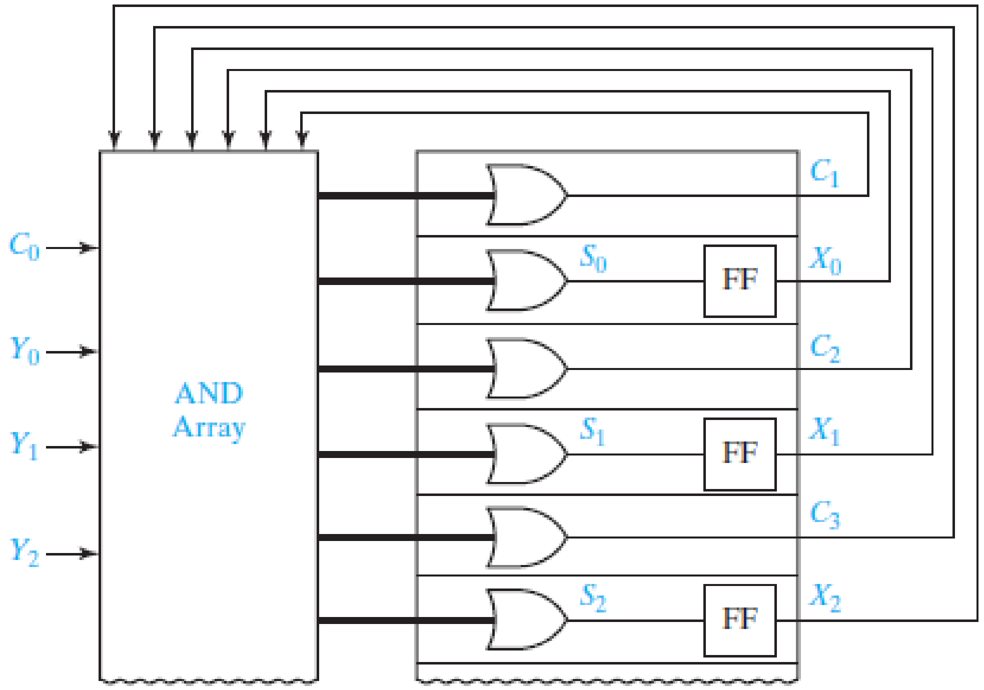


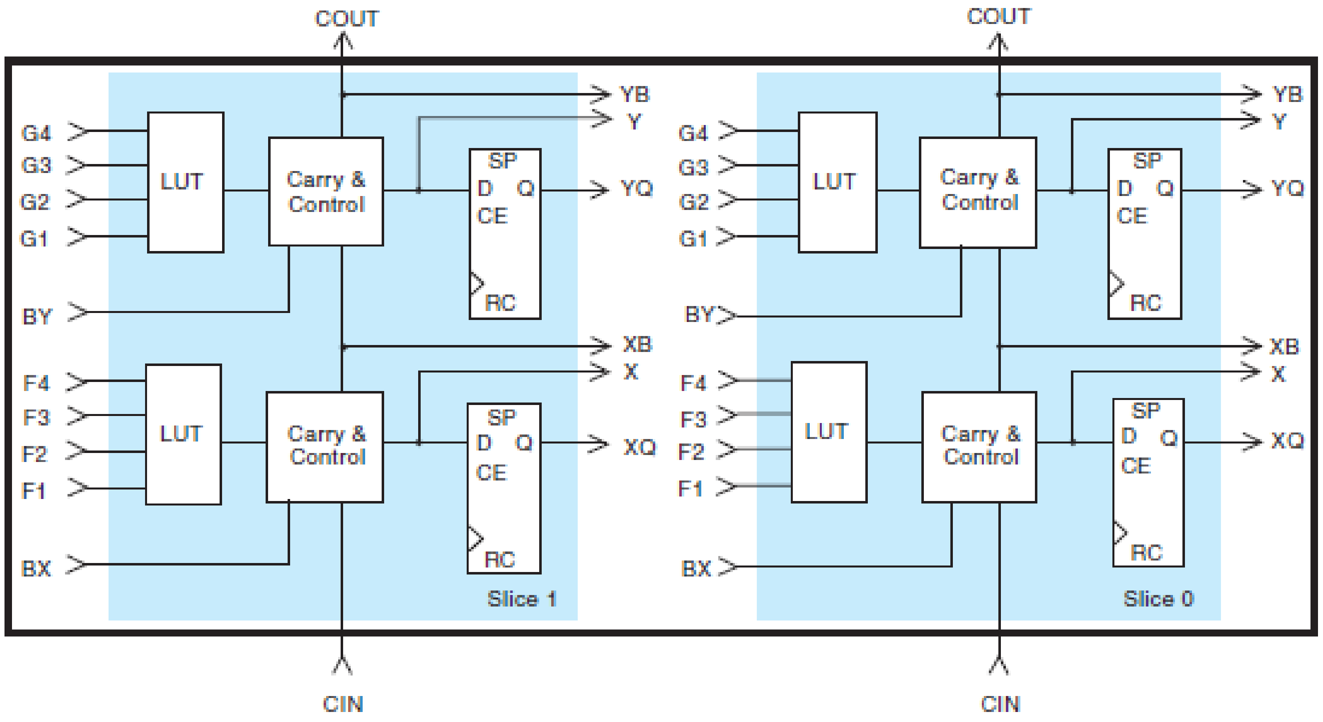
FIGURE 16-16 CPLD Implementation of a Parallel Adder with Accumulator

© Cengage Learning 2014

Sequential Circuit Design Using FPGAs

Simplified Block Diagram of Xilinx Virtex/ Spartan II CLB:

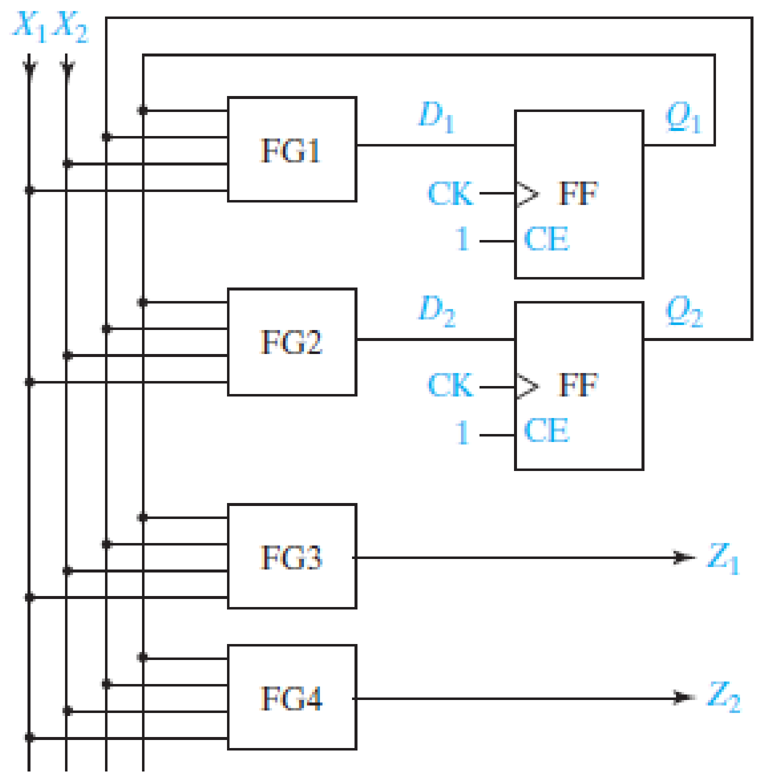
FIGURE 16-17 Xilinx Virtex/ Spartan II CLB (Figure based on figures and text owned by Xilinx, Inc., Courtesy of Xilinx, Inc. © Xilinx, Inc. 1999–2003. All rights reserved.)



Sequential Circuit Design Using FPGAs

FPGA Implementation of Mealy Machine:

FIGURE 16-18 FPGA Implementation of a Mealy Machine



Sequential Circuit Design Using FPGAs

FPGA Implementation of Shift Register:

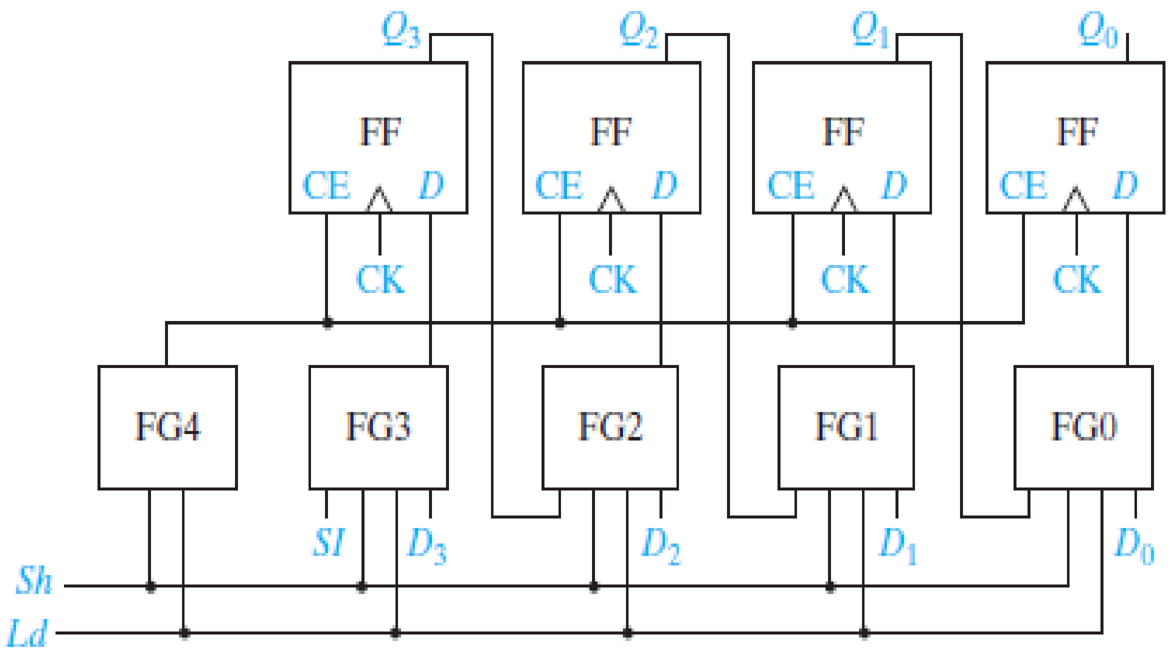


FIGURE 16-19 FPGA Implementation of a Shift Register

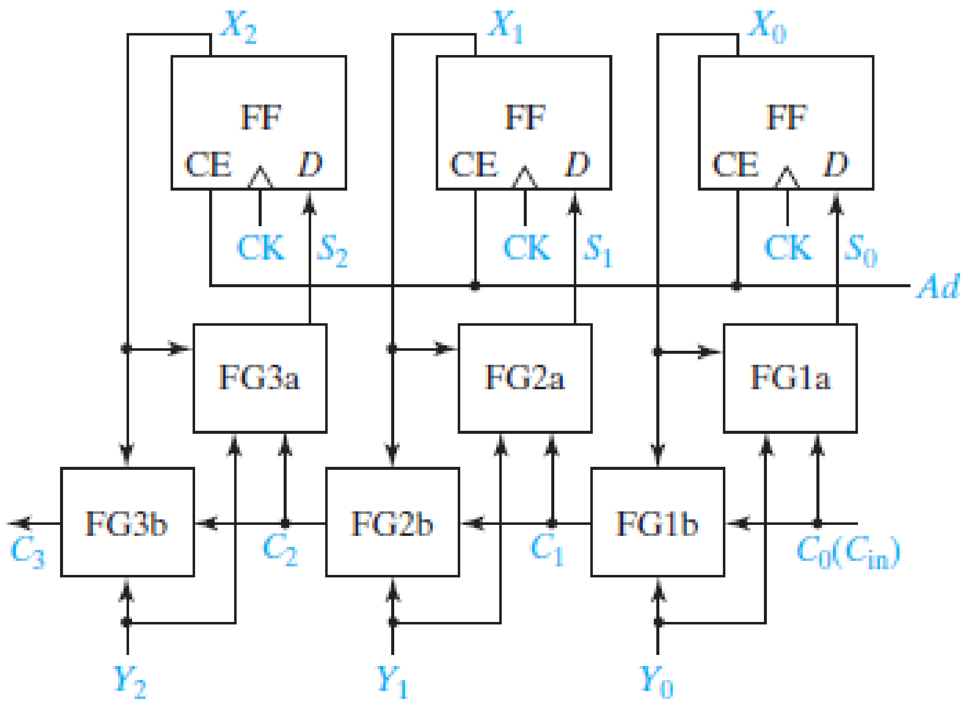
© Cengage Learning 2014

Sequential Circuit Design Using FPGAs

FPGA Implementation of Parallel Adder with Accumulator:

FIGURE 16-20 FPGA Implementation of a Parallel Adder with Accumulator

© Cengage Learning 2014



Simulation and Testing of Sequential Circuits

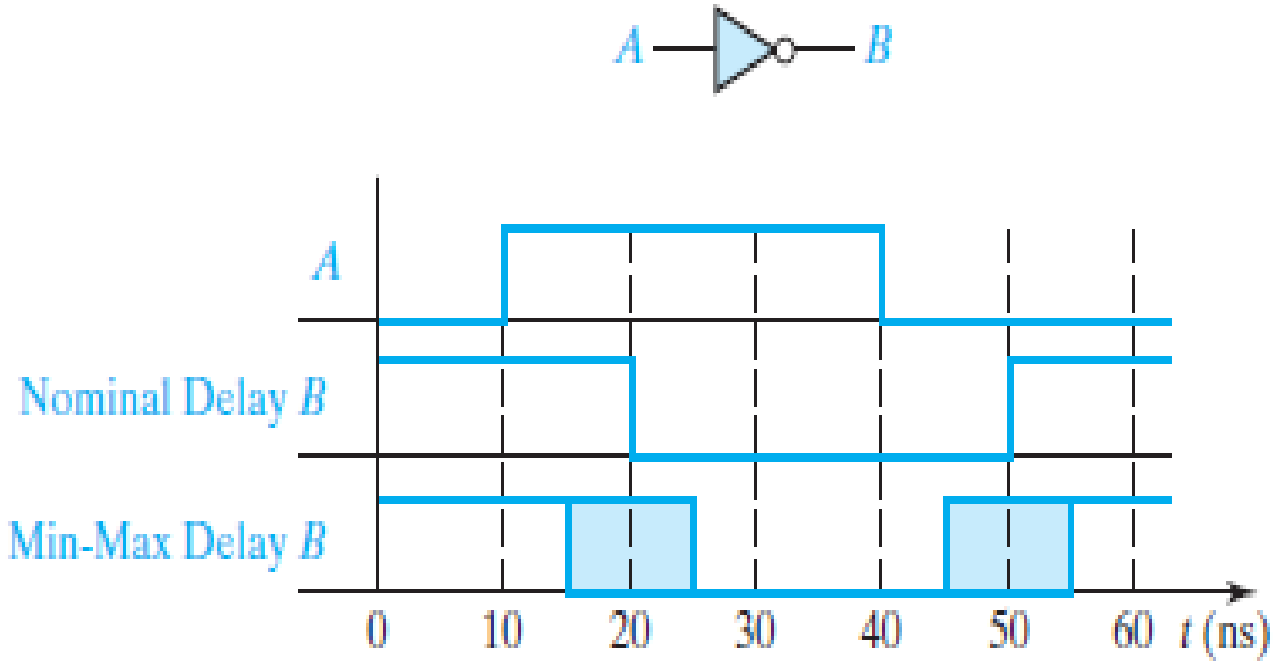
Simulation of Sequential Circuits:

- For sequential circuits, the propagation delays associated with the individual logic elements must be taken into account, and the presence of feedback may cause complications.
- The simulator output usually includes timing diagrams which show the times at which different signals in the circuit change.

Simulation and Testing of Sequential Circuits

FIGURE 16-21
Simulator Output for an Inverter

© Cengage Learning 2014



Simulation and Testing of Sequential Circuits

Testing Sequential Circuits Procedure:

1. Using the direct set and clear inputs, set the flip-flop states to correspond to one of the present states in the table.

2. For a Moore machine, check to see that the output is correct. For a Mealy machine, check to see that the output is correct for each input combination.

3. For each input combination, clock the circuit and check to see that the next state of the flip-flops is correct. (Reset the circuit to the proper state before each input combination is applied.)

4. Repeat steps 1, 2, and 3 for each of the present states in the table.

Simulation and Testing of Sequential Circuits

Using a Shift Register to Generate Synchronized Inputs:

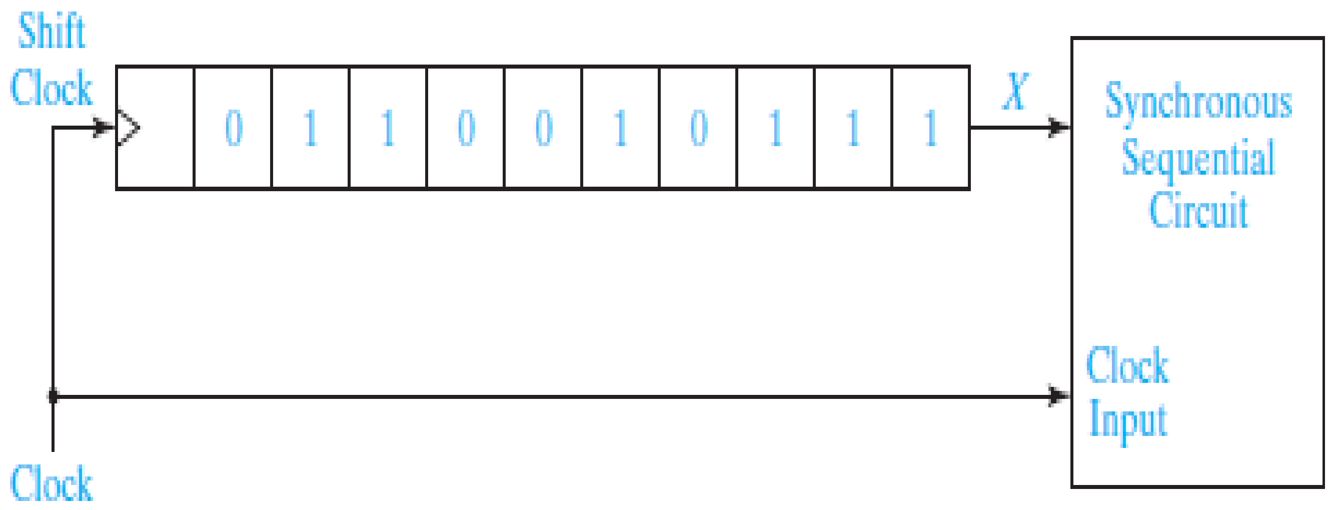


FIGURE 16-24 Using a Shift Register to Generate Synchronized Inputs

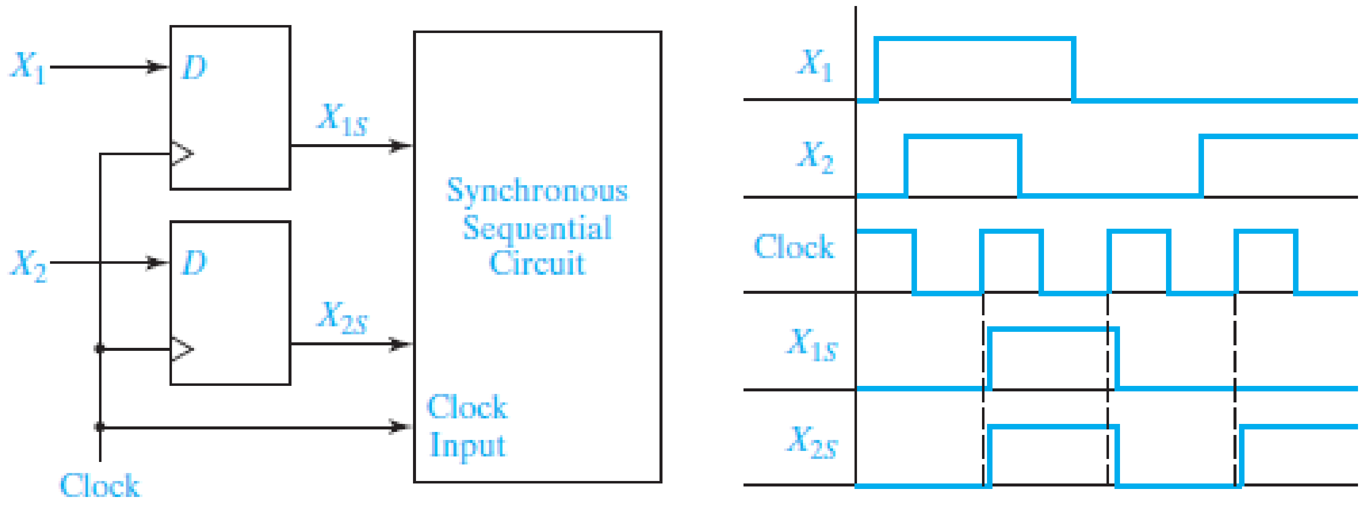
© Cengage Learning 2014

Simulation and Testing of Sequential Circuits

Synchronizer Circuit:

FIGURE 16-25

© Cengage Learning 2014

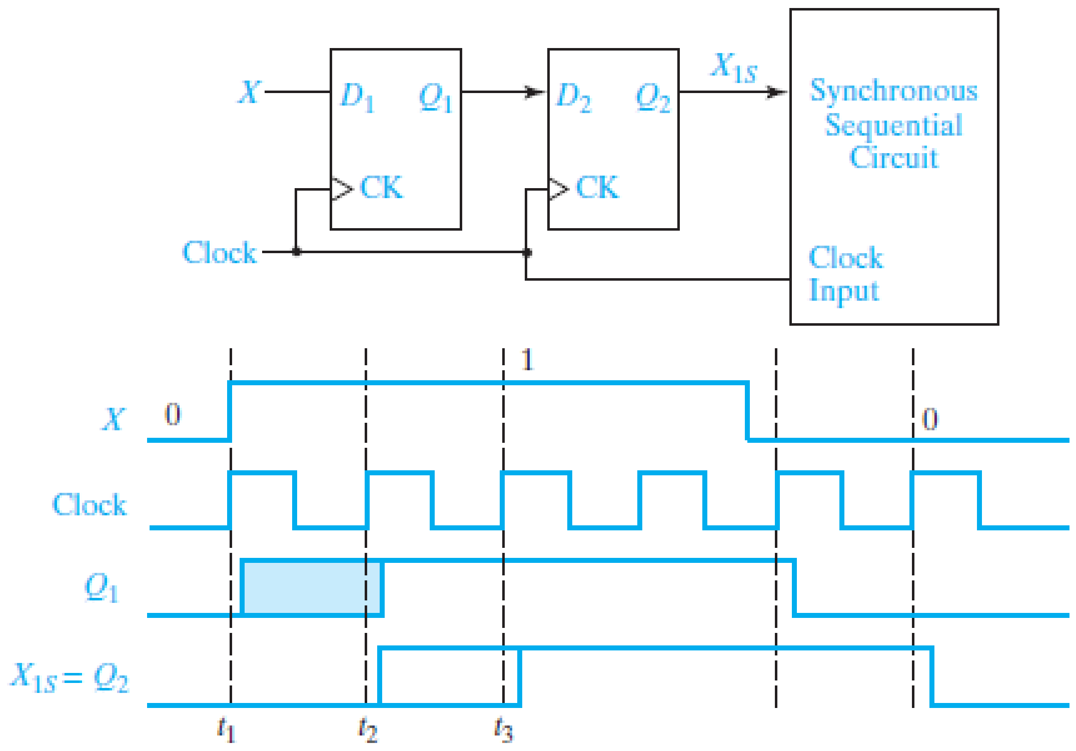


(a) Synchronizer circuit (b) Synchronizer inputs and outputs

Simulation and Testing of Sequential Circuits

Synchronizer Circuit Using Two D Flip-Flops:

FIGURE 16-26
Synchronizer with Two D Flip-Flops



Overview of Computer-Aided Design

Several functions of CAD tools:

- Functions of CAD tools include:
  - Generation and minimization of logic equations
  - Generation of bit patterns for programming PLDs
  - Schematic capture
  - Simulation
  - SimUaid
  - Synthesis tools.
  - IC design and layout
  - Test generation
  - PC Board Layout

Overview of Computer-Aided Design

Design of a Small Digital System Using FPGAs:

1. Draw a block diagram of the digital system. Define the required control signals and construct a state graph that describes the required sequence of operations.
2. Work out a detailed logic design of the system using gates, flip-flops, registers, counters, adders, etc.
3. Construct a logic diagram of the system, using a schematic capture program.
4. Simulate and debug the logic diagram and make any necessary corrections to the design.

Overview of Computer-Aided Design

Design of a Small Digital System Using FPGAs (continued):

5. Run an implementation program that fits the design into the target FPGA. This program carries out the following steps: (a) Partition the logic diagram into pieces that will fit into CLBs of the target FPGA.

(b) Place the CLBs within the logic cell array of the FPGA and route the connections between the logic cells.

(c) Generate the bit pattern necessary to program the FPGA.

Overview of Computer-Aided Design

Design of a Small Digital System Using FPGAs (continued):

6. Run a timing simulation of the completed design to verify that it meets specifications. Make any necessary corrections and repeat the process as necessary.

7. Download the bit pattern into the internal configuration memory cells in the FPGA and test the operation of the FPGA.