# CHAPTER 16

# SEQUENTIAL CIRCUIT DESIGN

# Contents

16.1 Summary of Design Procedure for Sequential Circuits
16.2 Design Example--Code Converter
16.3 Design of Iterative Circuits
16.4 Design of Sequential Circuits Using ROMs and PLAs
16.5 Sequential Circuit Design Using CPLDs
16.6 Sequential Circuit Design Using FPGAs
16.7 Simulation and Testing of Sequential Circuits
16.8 Overview of Computer-Aided Design

## 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 circuit, 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

1. Given the problem Statement, determine the relationship between the input and output sequences and derive state table. Construct a State Graph.

2. Reduce the table to a minimum number of states. Eliminate duplicates rows by row matching and then form an implication table.

3. Use Flip/flops for representing states. Assign a unique combination of F/F states corresponds to in each state in reduced table.

4. Form a transition table.

5. Plot next-state map and input maps for F/F and derive the input F/F equations.

6. Realize the F/F input equations and output equations using available logic

7. Testing your circuit

# 16.2 Design Example-Code Converter

BCD code → excess 3 code converter

TABLE 16-1

| 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 |

# 16.2 Design Example-Code Converter

TABLE-16-2 State Table for Code Converter

| 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 | - |

# 16.2 Design Example-Code Converter

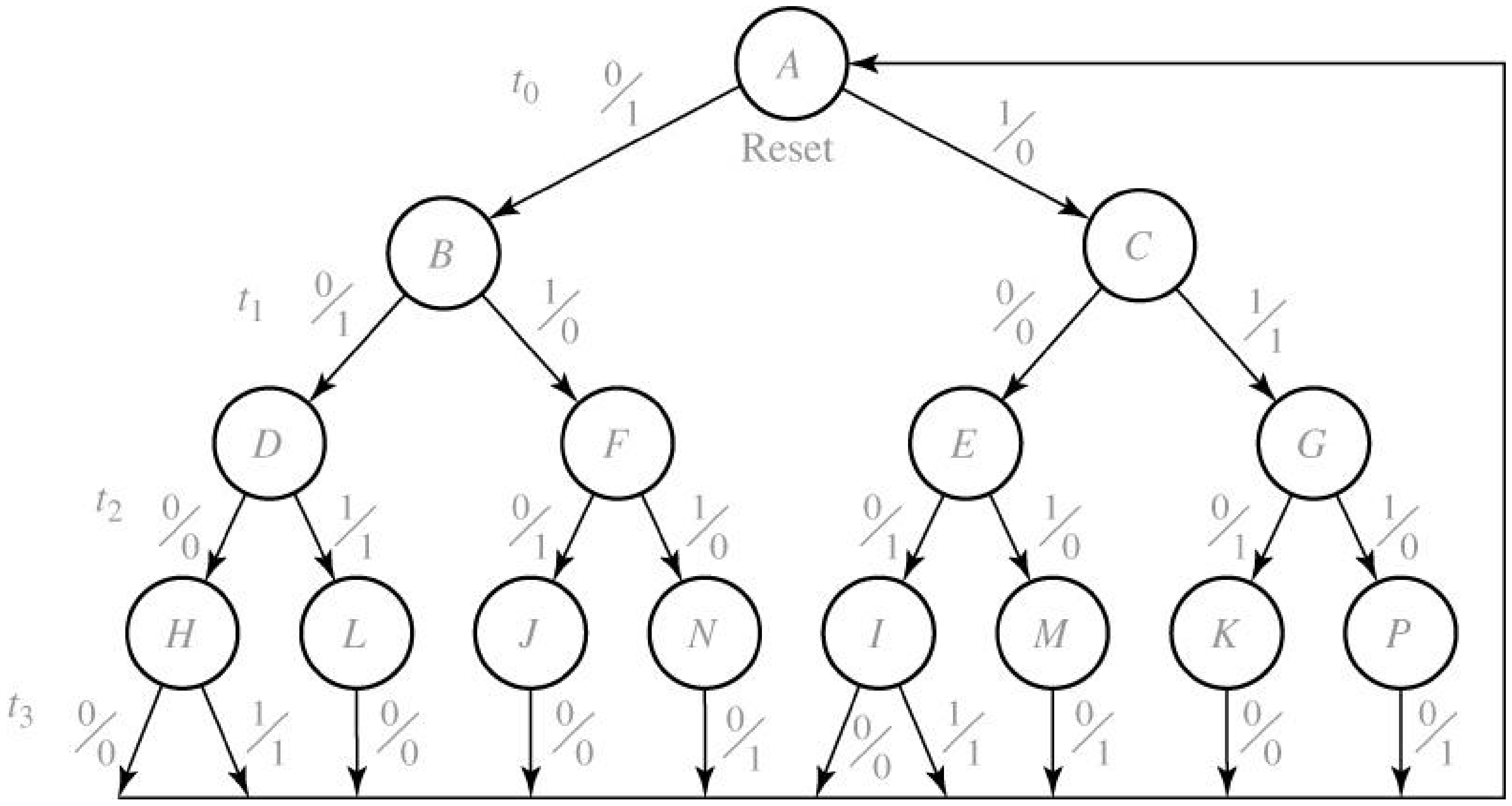
TABLE16-3 Reduced State Table for Code Converter

| Time | Present State | Next State X = 0 | Next State 1 | Present Output(Z) X = 0 | Present Output(Z) 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 | - |

$H \equiv I \equiv J \equiv K \equiv L$ , $M \equiv N \equiv P$ and $E \equiv F \equiv G$

# 16.2 Design Example-Code Converter

Figure 16-1: State Graph for Code Converter



# 16.2 Design Example-Code Converter

Figure 16-2: Assignment Map for Flip Flops

| $Q_2Q_3$ \ $Q_1$ | 0 | 1 |
|---|---|---|
| 00 | *A* | *B* |
| 01 | | *C* |
| 11 | *H* | *D* |
| 10 | *M* | *E* |

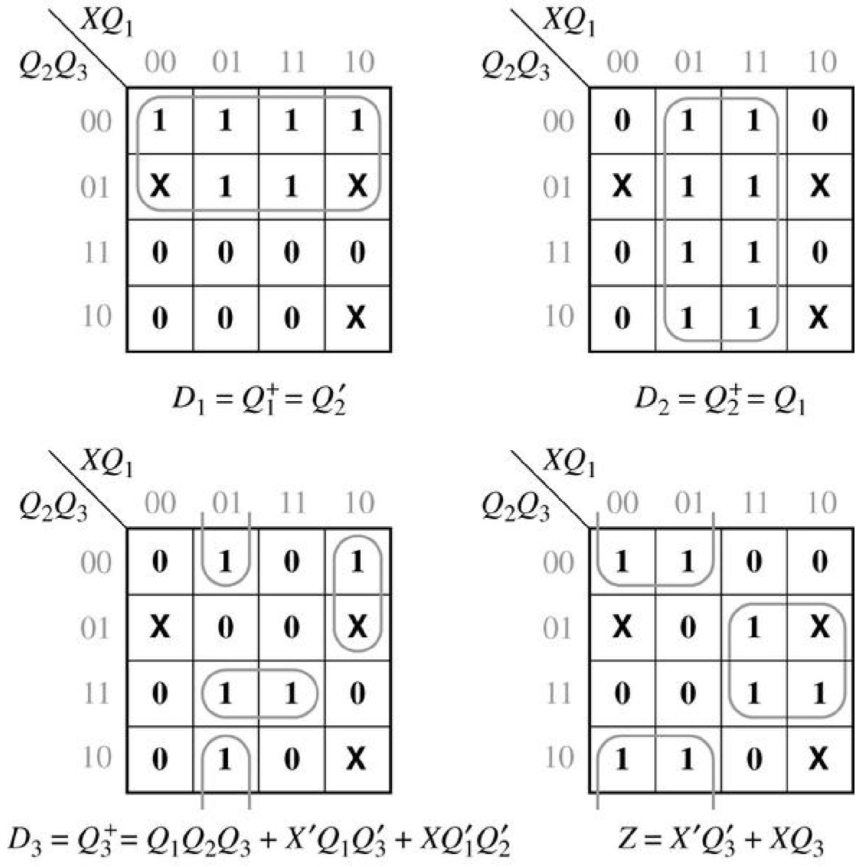
(a) Assignment map

| | | $Q_1^+Q_2^+Q_3^+$ | | Z | |
|---|---|---|---|---|---|
| | $Q_1Q_2Q_3$ | X=0 | X=1 | X=0 | X=1 |
| A | 000 | 100 | 101 | 1 | 0 |
| B | 100 | 111 | 110 | 1 | 0 |
| C | 101 | 110 | 110 | 0 | 1 |
| D | 111 | 011 | 011 | 0 | 1 |
| E | 110 | 011 | 010 | 1 | 0 |
| H | 011 | 000 | 000 | 0 | 1 |
| M | 010 | 000 | xxx | 1 | x |
| - | 001 | xxx | xxx | x | x |

(b)transition table

# 16.2 Design Example-Code Converter

Figure 16-3: Karnaugh Maps for Code Converter Design



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

$$D_2 = Q_2^+ = Q_1$$

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

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

After the state assignment has been made the transition table is filled in according to the assignment, and the next-state maps are plotted as shown in Figure 16-3

# 16.2 Design Example-Code Converter

Figure 16-4: Code Converter Circuit

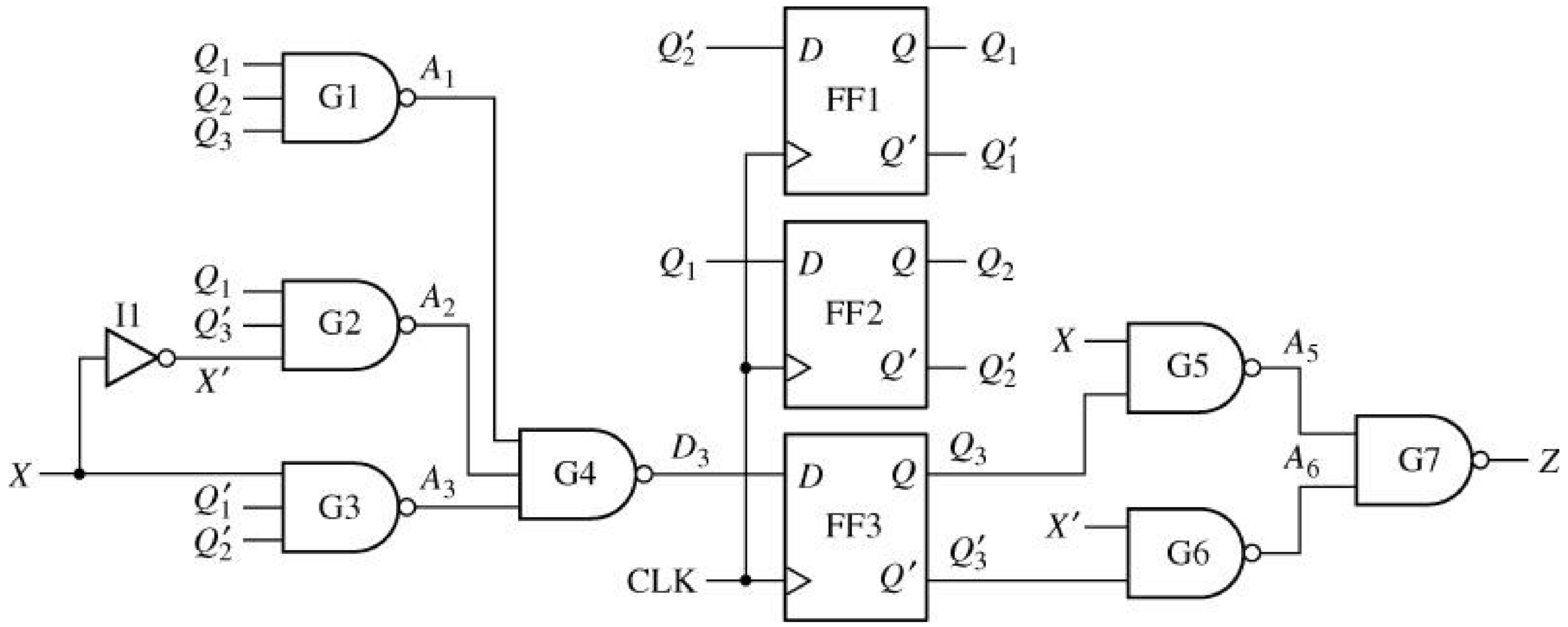
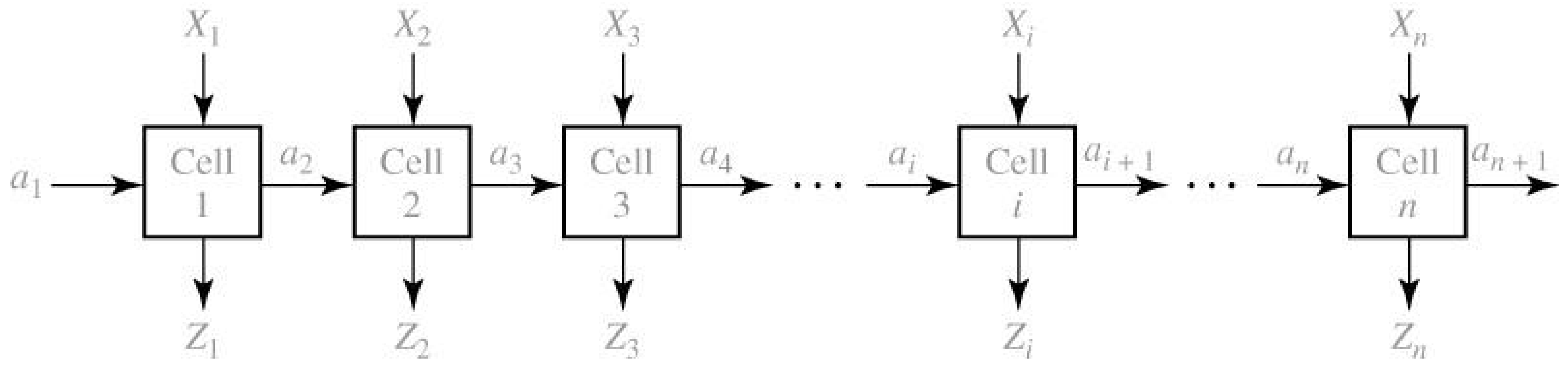


Figure 16-4 shows the resulting sequential circuit

# 16.3 Design of Iterative Circuits

Sequential Circuit Design → Iterative Design

Figure 16-5: Unilateral Iterative Circuit



The simplest form of an iterative circuit consists of a linear array of combinational cells with signals between cells traveling in only one direction.

# 16.3 Design of Iterative Circuits

## Comparator Design using Iterative Circuit

Figure 16-6: Form of Iterative Circuit for Comparing Binary Numbers

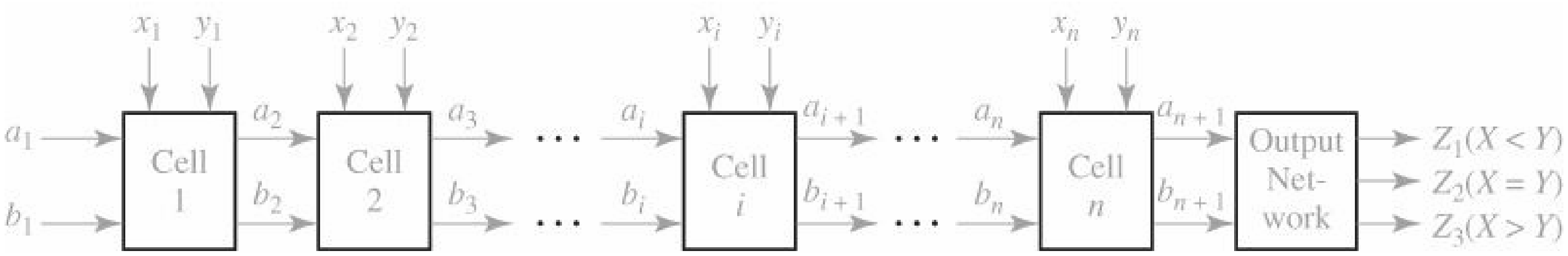


Figure 16-6 shows the form of the iterative circuit, although the number of leads between each pair of cells is not yet know.

## 16.3 Design of Iterative Circuits

TABLE 16-4 State Table for Comparator

| | $S_i$ | $X_iY_i =$ 00 | $S_{i+1}$ 01 | 11 | 10 | $Z_1$ | $Z_2$ | $Z_3$ |
|---|---|---|---|---|---|---|---|---|
| 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 |

# 16.3 Design of Iterative Circuits

TABLE 16-5 State Assignment and Transition Table for Comparator

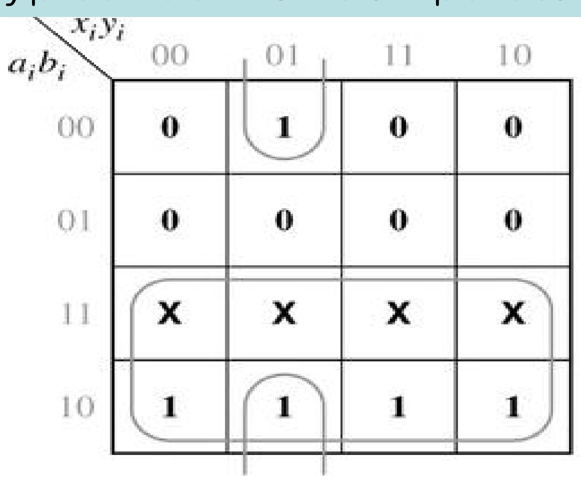
| | | | $a_{i+}$ $b_{i+}$ | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $a_i$ | $b_i$ | $x_iy_i$= 00 | 01 | 11 | 10 | $Z_1$ | $Z_2$ | $Z_3$ |
| 0 | 0 | 00 | 10 | 00 | 01 | 0 | 1 | 0 |
| 0 | 1 | 01 | 01 | 01 | 01 | 0 | 0 | 1 |
| 1 | 0 | 10 | 10 | 10 | 10 | 1 | 0 | 0 |

Equations for the first cell (a1=b1='00')

$$a_2 = a_1 + x_1' y_1 b_1' = x_1' y_1$$
$$b_2 = b_1 + x_1 y_1' a_1' = x_1 y_1'$$

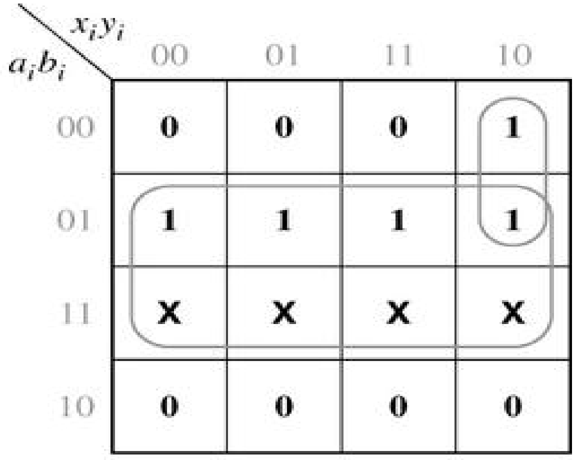
# 16.3 Design of Iterative Circuits

Figure 16-7:Typical Cell for Comparator



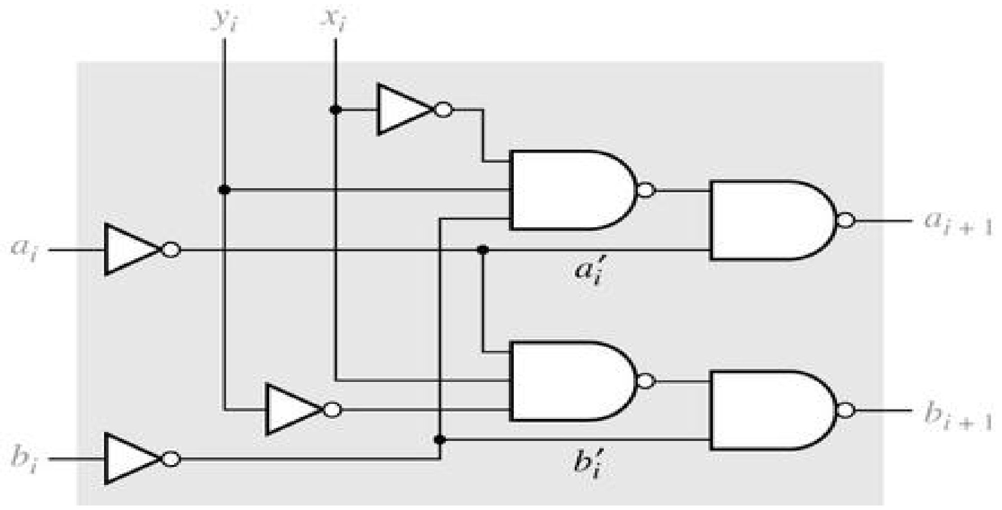
| $a_ib_i \backslash x_iy_i$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | X | X | X | X |
| 10 | 1 | 1 | 1 | 1 |

$a_{i+1} = a_i + x_i'y_ib_i'$



| $a_ib_i \backslash x_iy_i$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | X | X | X | X |
| 10 | 0 | 0 | 0 | 0 |

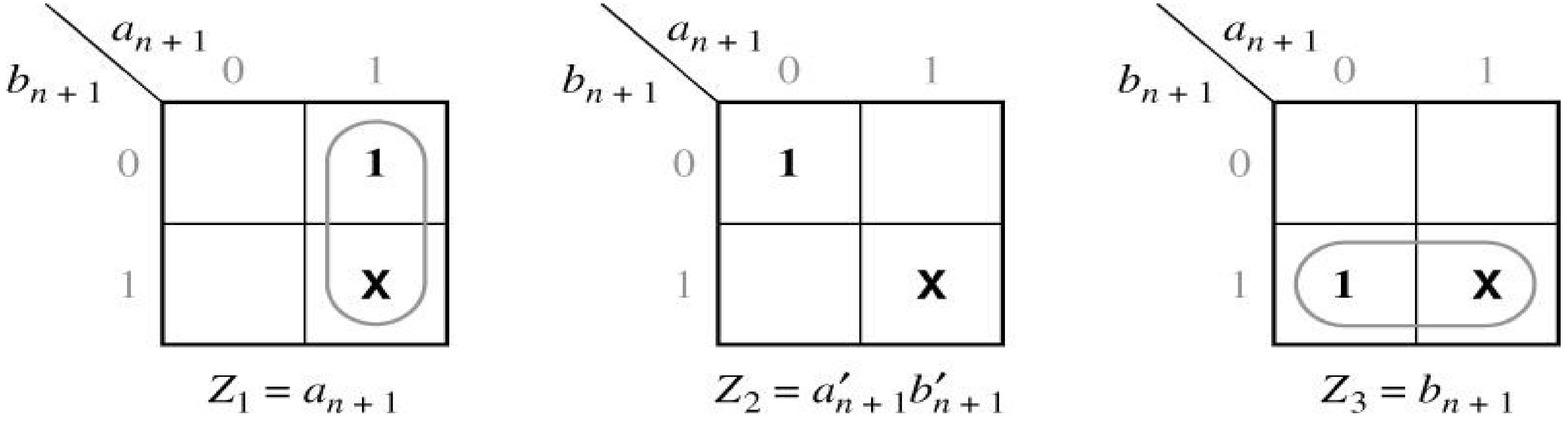
$b_{i+1} = b_i + x_iy_i'a_i'$

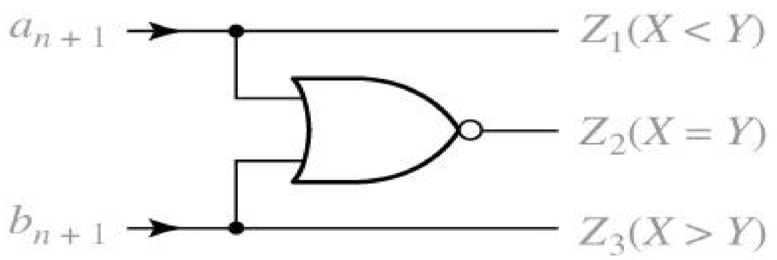


# 16.3 Design of Iterative Circuits

Figure 16-8: Output Circuit for Comparator

output maps, equations, and circuit.





$Z_1$ =1 if X < Y , $Z_2$=1 if X = Y , $Z_3$=1 if X > Y

# 16.3 Design of Iterative Circuits

Figure 16-9: Sequential Comparator for Binary Numbers

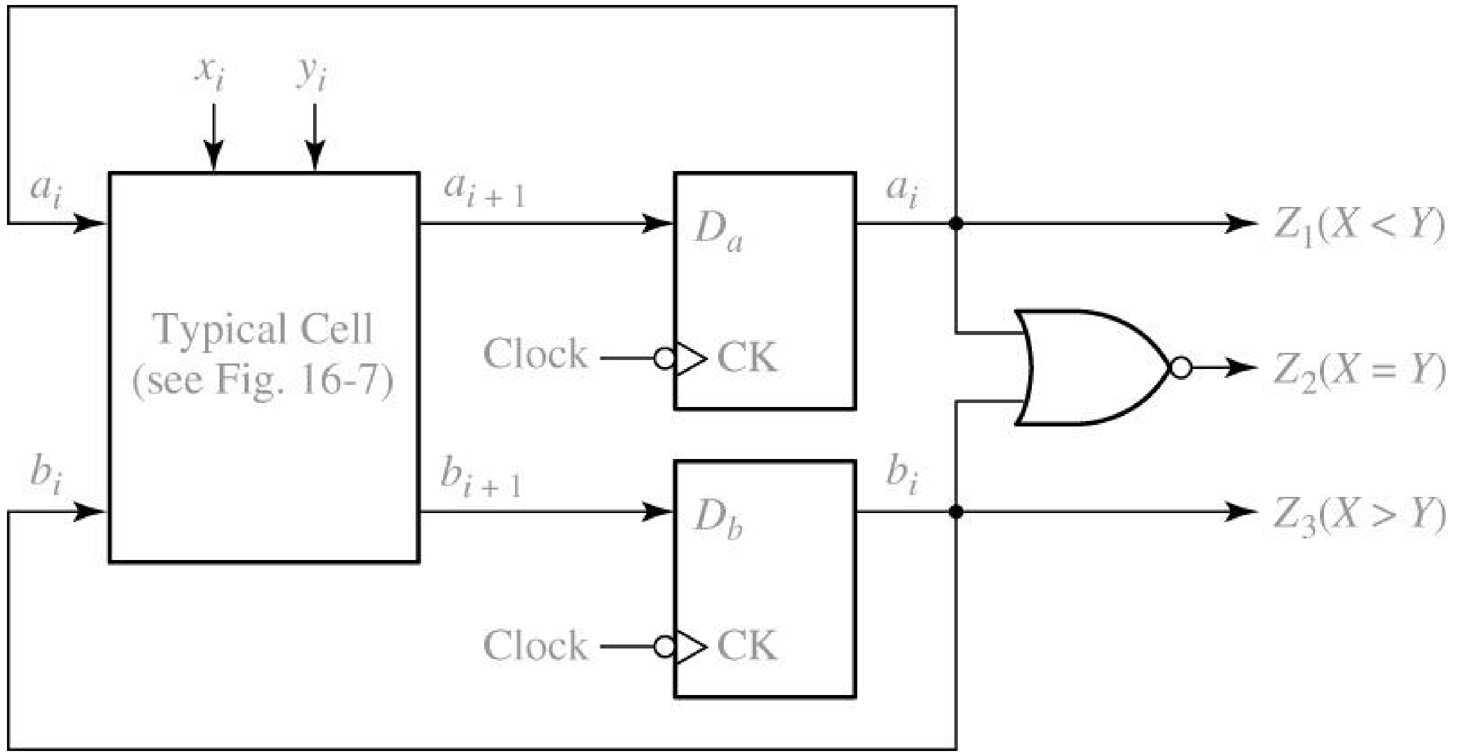


Figure 16-9 shows the resulting circuit.

# 16.4 Design of Sequential Circuits Using ROMs and PLAs

Sequential Circuit can be designed using a ROM and F/F's

TABLE 16-6: Revisit the Code Converter Design

(a)State table

| Present State | Next State X= 0 | Next State X= 1 | Present Output (Z) X= 0 | Present Output (Z) X= 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 | - |

## 16.4 Design of Sequential Circuits Using ROMs and PLAs

TABLE 16-6 (b)Transition table

|  |  |  |  | $Q_1^+Q_2^+Q_3^+$ |  | Z |  |
|---|---|---|---|---|---|---|---|
|  | $Q_1$ | $Q_2$ | $Q_3$ | X=0 | X=1 | 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 |
| K | 1 | 1 | 0 | 000 | - | 1 | - |

$$D_1 = Q_1^+, \quad D_2 = Q_2^+ \quad and \quad D_3 = Q_3^+$$

# 16.4 Design of Sequential Circuits Using ROMs and PLAs

TABLE 16-6

(c)Truth table

*ROM INPUTS

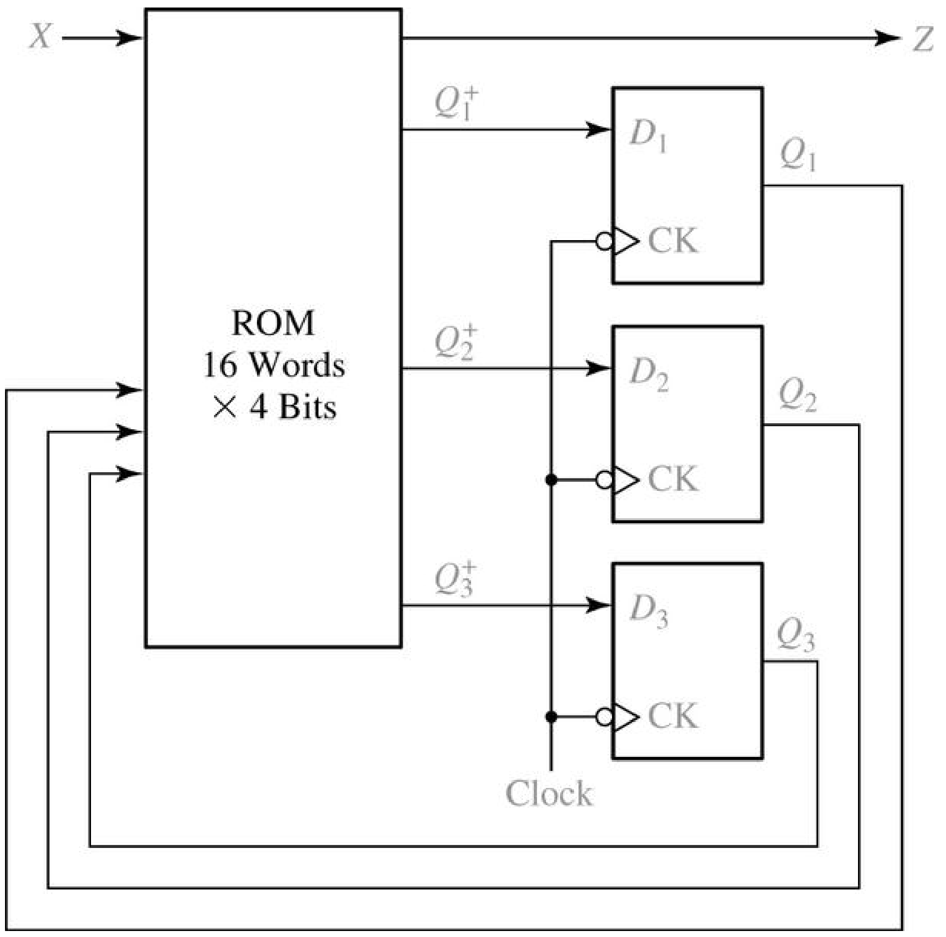
(X,$Q_1$,$Q_3$ and $Q_3$)

*ROM OUTPUTS
(Z,$D_1$,$D_2$ and $D_3$)

| 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 |

## 16.4 Design of Sequential Circuits Using ROMs and PLAs

Figure 16-10:Realization of Table 16.6(a) Using a ROM



A ROM with four input($2^4$ words) and four outputs is required, as shown in Figure16-10

## 16.4 Design of Sequential Circuits Using ROMs and PLAs

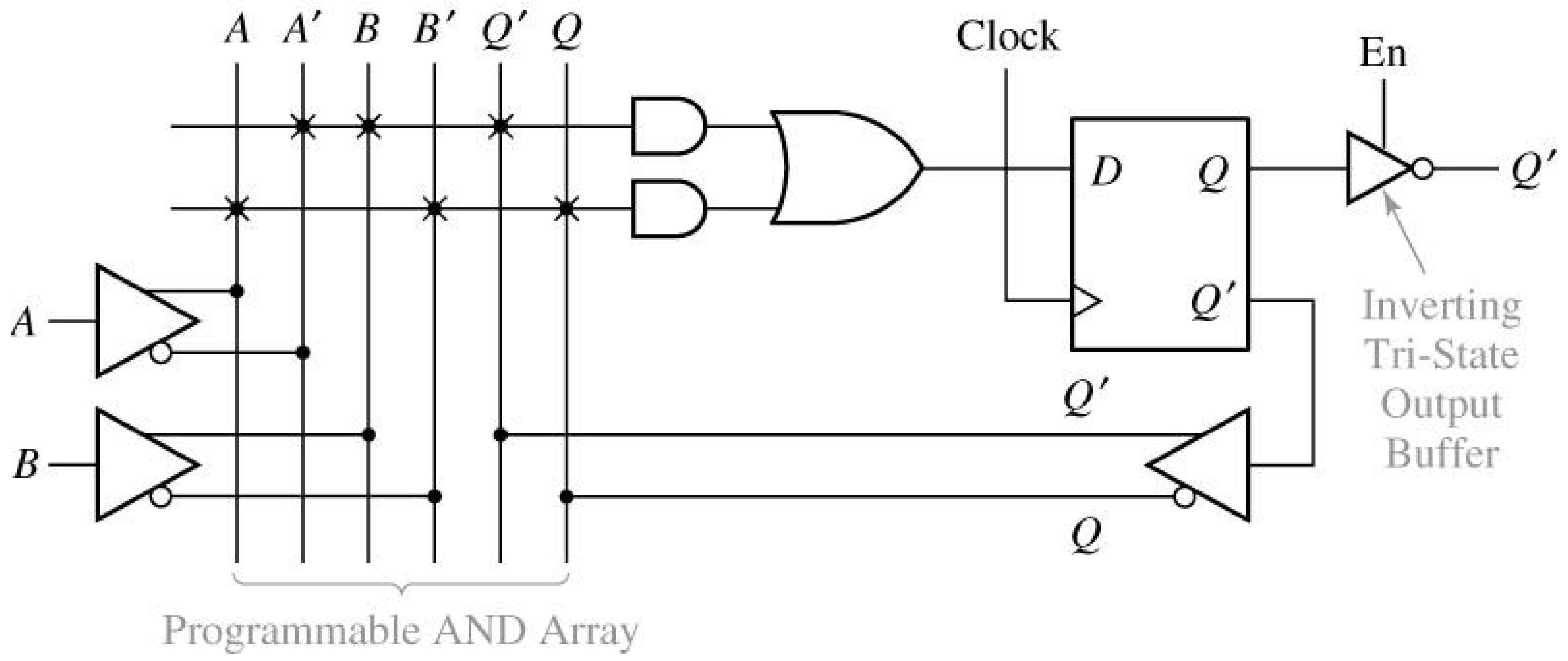
$$D_1 = Q_1^+ = Q_2'$$
$$D_2 = Q_2^+ = Q_1$$
$$D_3 = Q_3^+ = Q_1Q_2Q_3 + X'Q_1Q_3' + XQ_1'Q_2'$$
$$Z = X'Q_3' + XQ_3$$

TABLE 16-7

| X | $Q_1$ | $Q_2$ | Q3 | 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 |

## 16.4 Design of Sequential Circuits Using ROMs and PLAs

Figure 16-11: Segment of Sequential PAL



$$Q^+ = D = A'BQ' + AB'Q$$

## 16.5 Sequential Circuit Design Using CPLDs

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

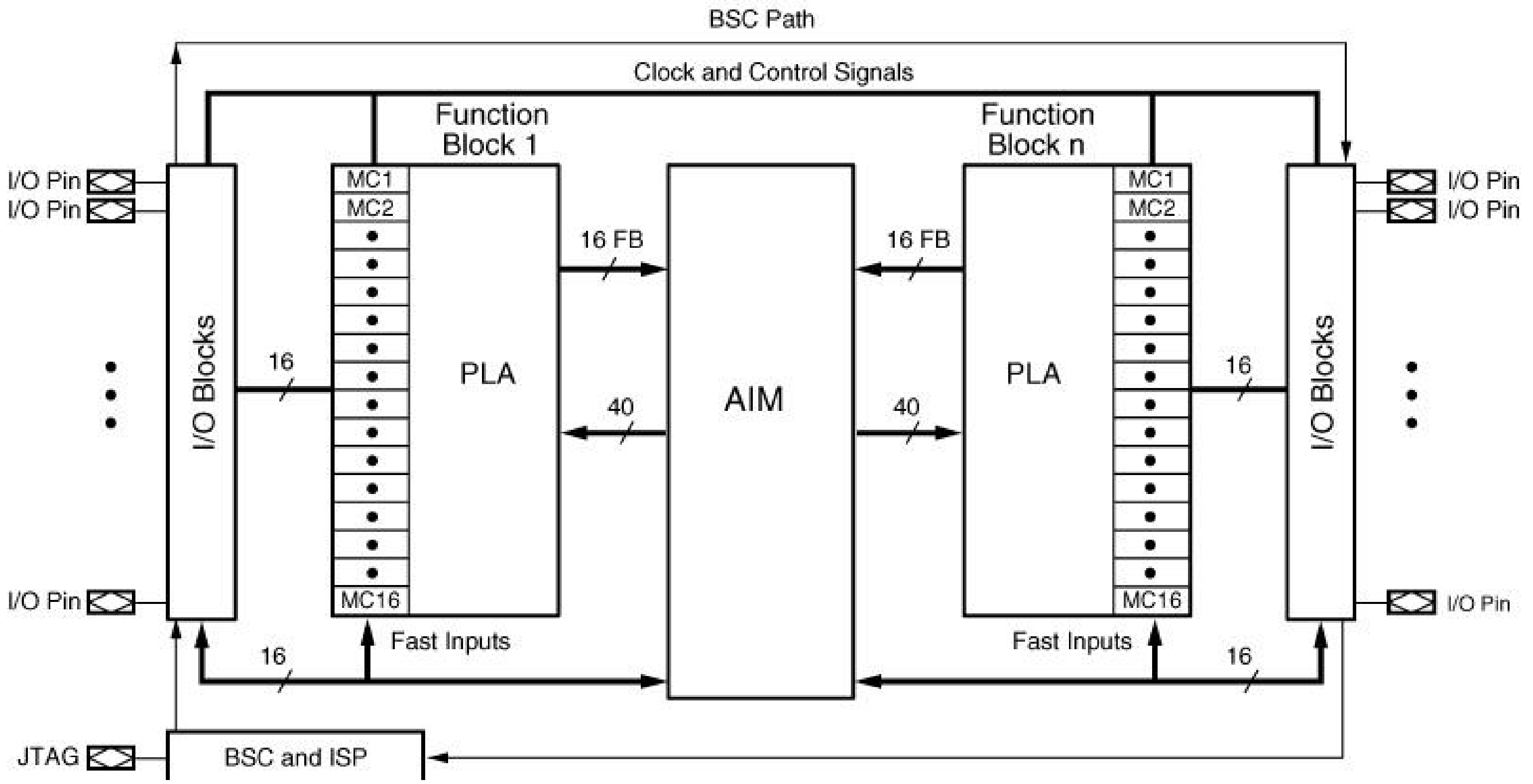


Figure 16-12 shows the structure of a Xilinx CoolRunner II CPLD, which uses a PLA in each function block.

# 16.5 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.)

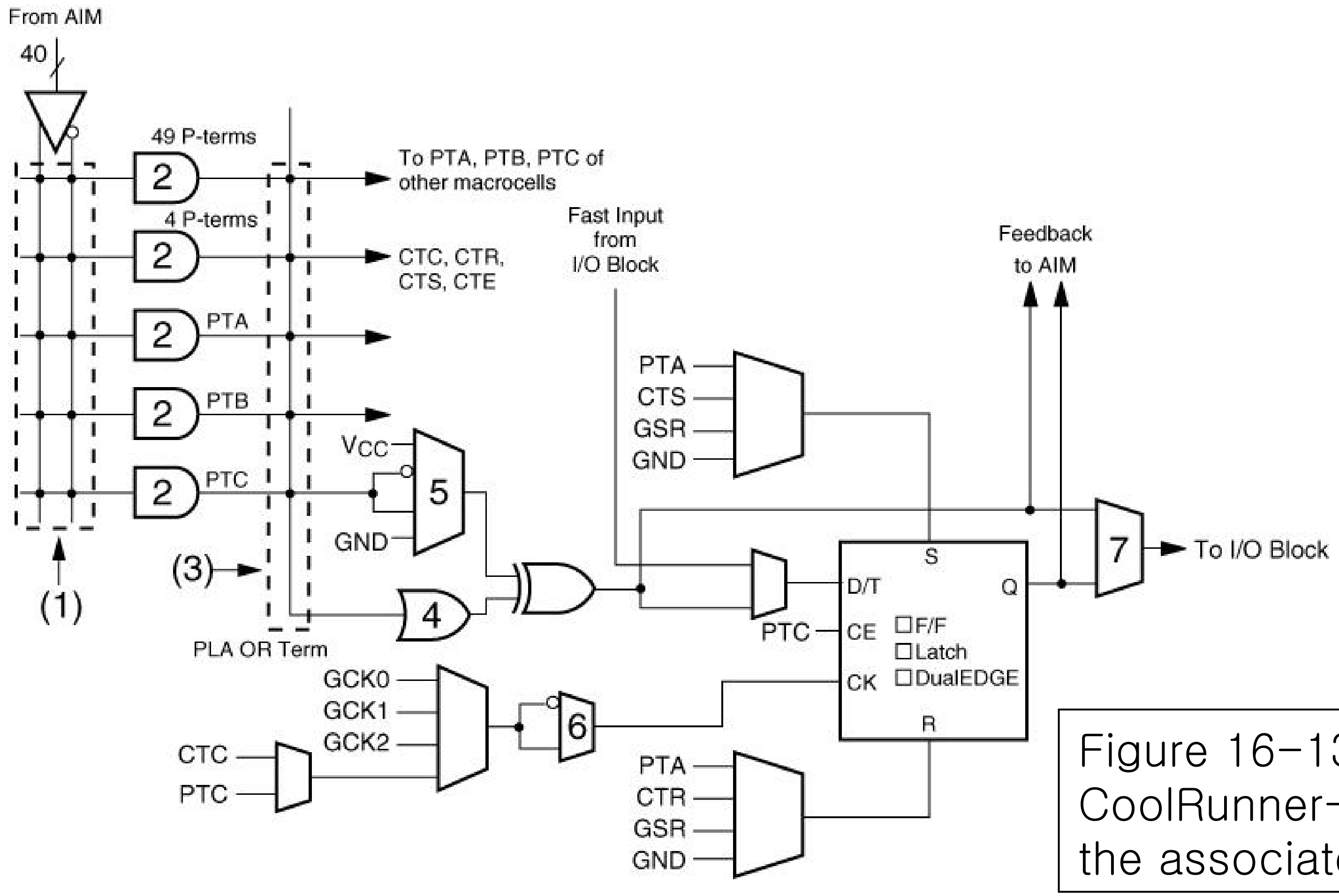


Figure 16-13 represents a CoolRunner-II macrocell and the associated AND array.

## 16.5 Sequential Circuit Design Using CPLDs

Figure 16-14: CPLD Implementation of a Mealy Machine

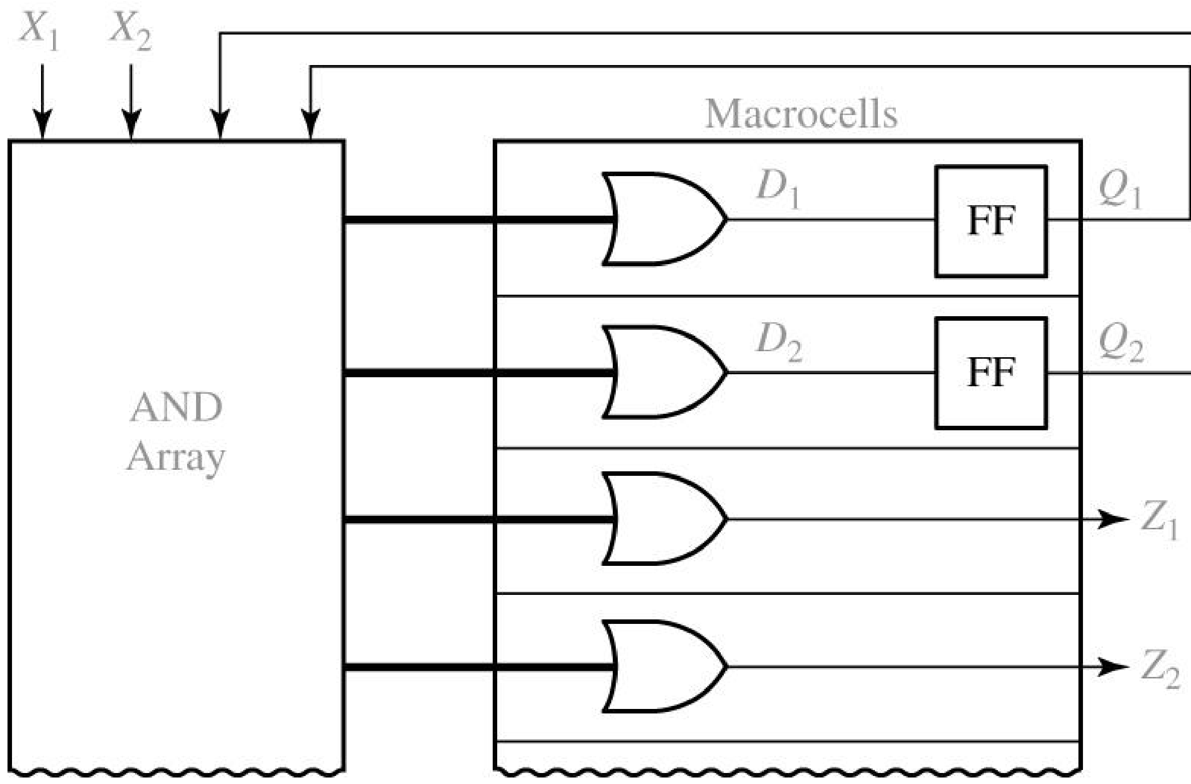


Figure 16-14 shows how a Mealy sequential machine with two input, two outputs, and two flip-flops can be implemented by a CPLD

## 16.5 Sequential Circuit Design Using CPLDs

Figure 16-15: CPLD Implementation of a Shift Register

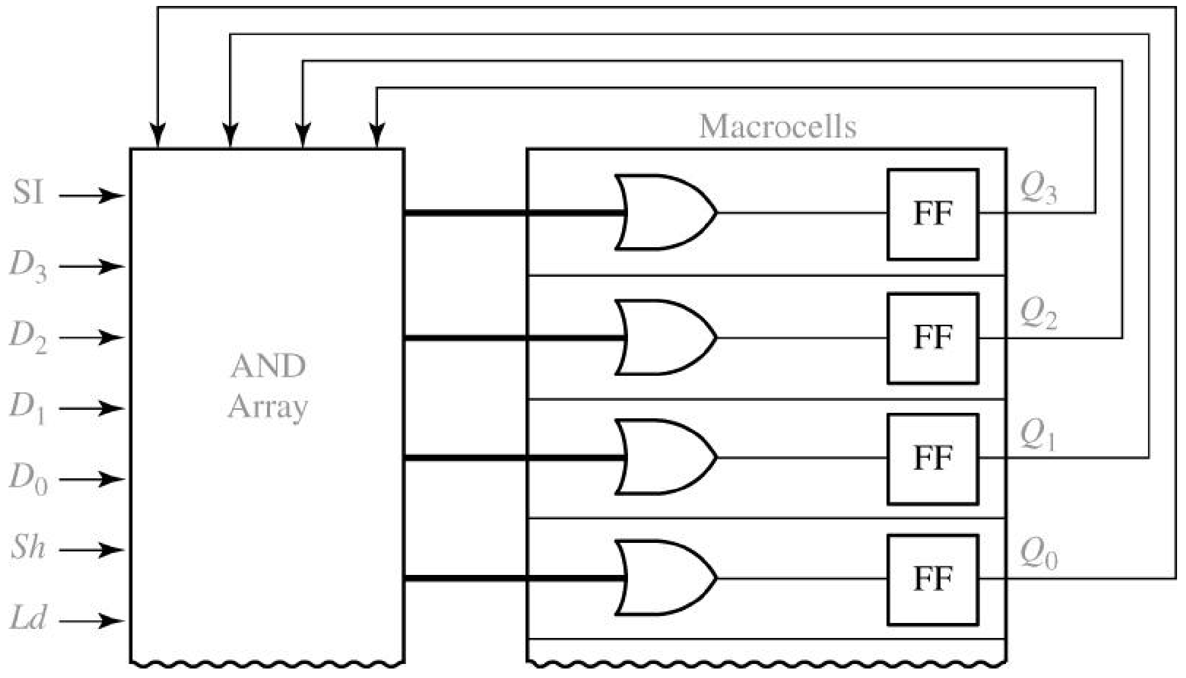
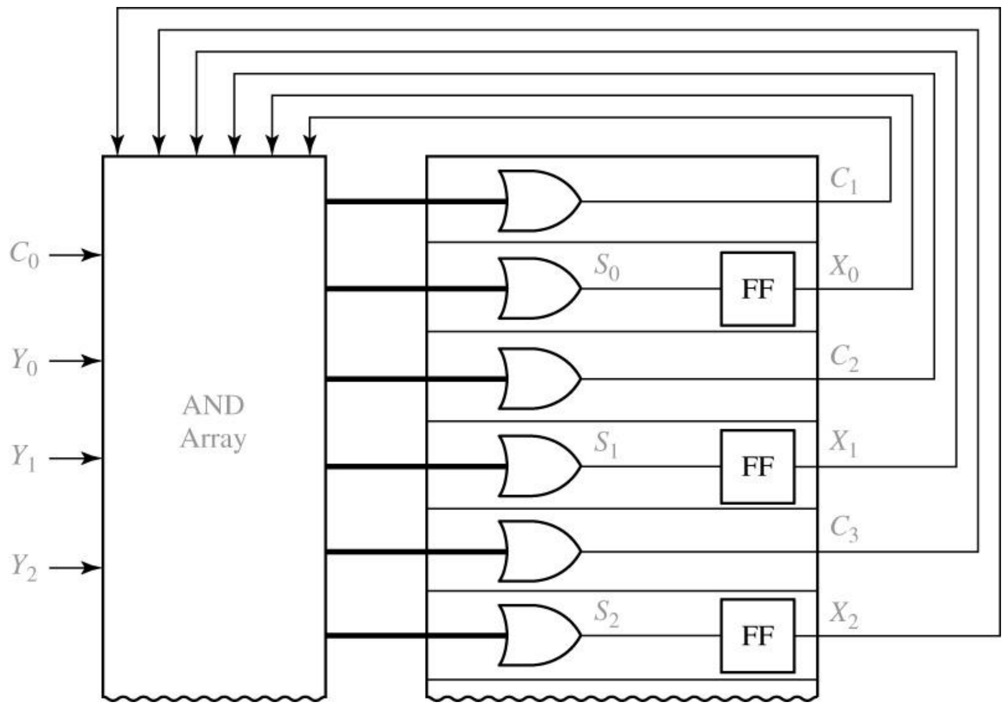


Figure 16-15 shows how the 4-bit loadable right-shift register of Figure 12-15 can be implemented using four macrocells of a CPLD

# 16.5 Sequential Circuit Design Using CPLDs

Figure 16-16: CPLD Implementation of a Parallel Adder with Accumulator



$$X_i^+ = X_i \oplus Y_i \oplus c_i$$

$T \quad INPUT \quad is$

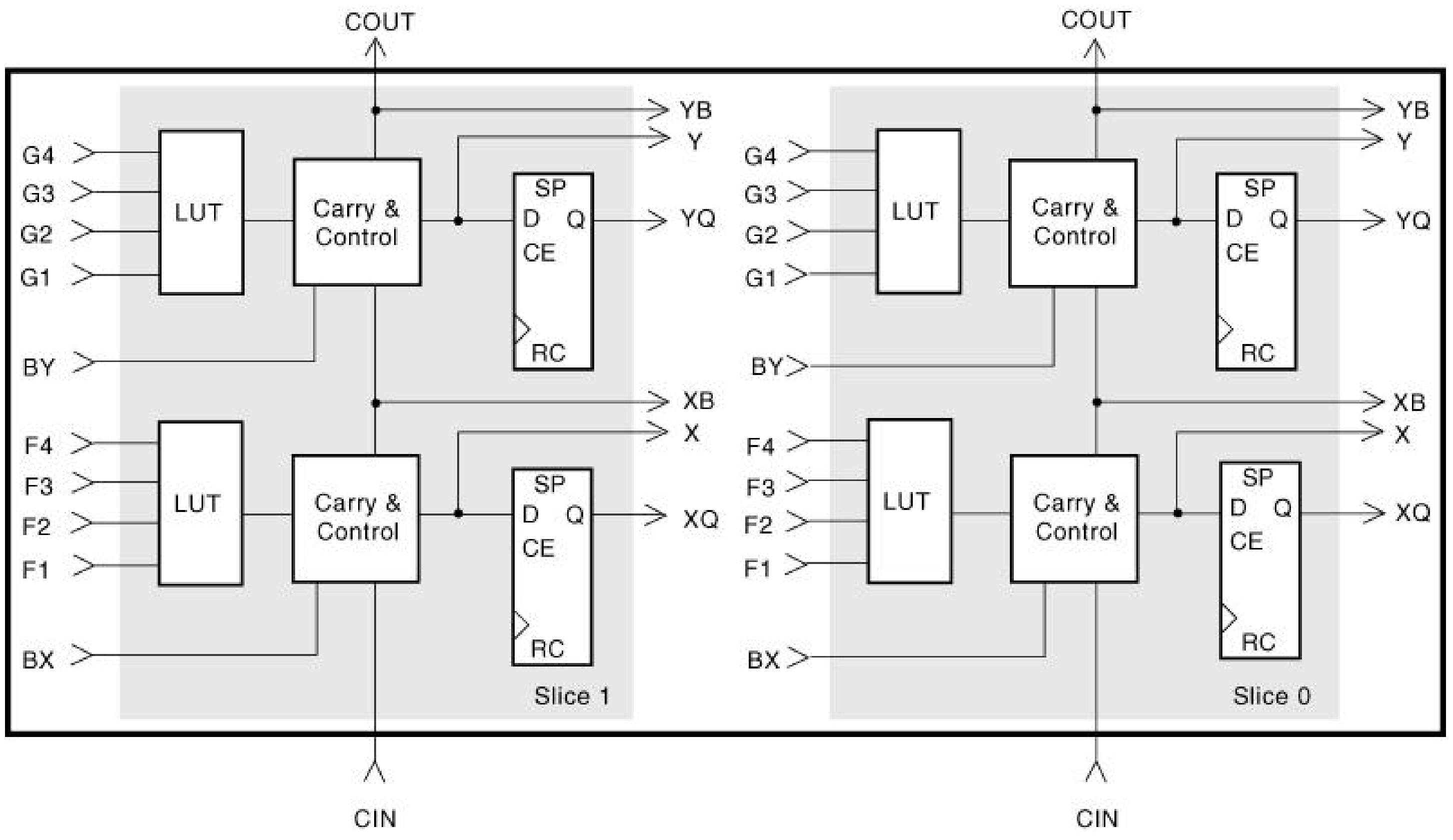
$$T_i = X_i^+ \oplus X_i = Y_i \oplus C_i$$

Figure 16-16 shows how three bit of the parallel adder with accumulator of Figure 12-5 can be implemented using a CPLD.

## 16.6 Sequential Circuit Design Using FPGAs

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.)



# 16.6 Sequential Circuit Design Using FPGAs

Figure 16-18: FPGA Implementation of a Mealy Machine

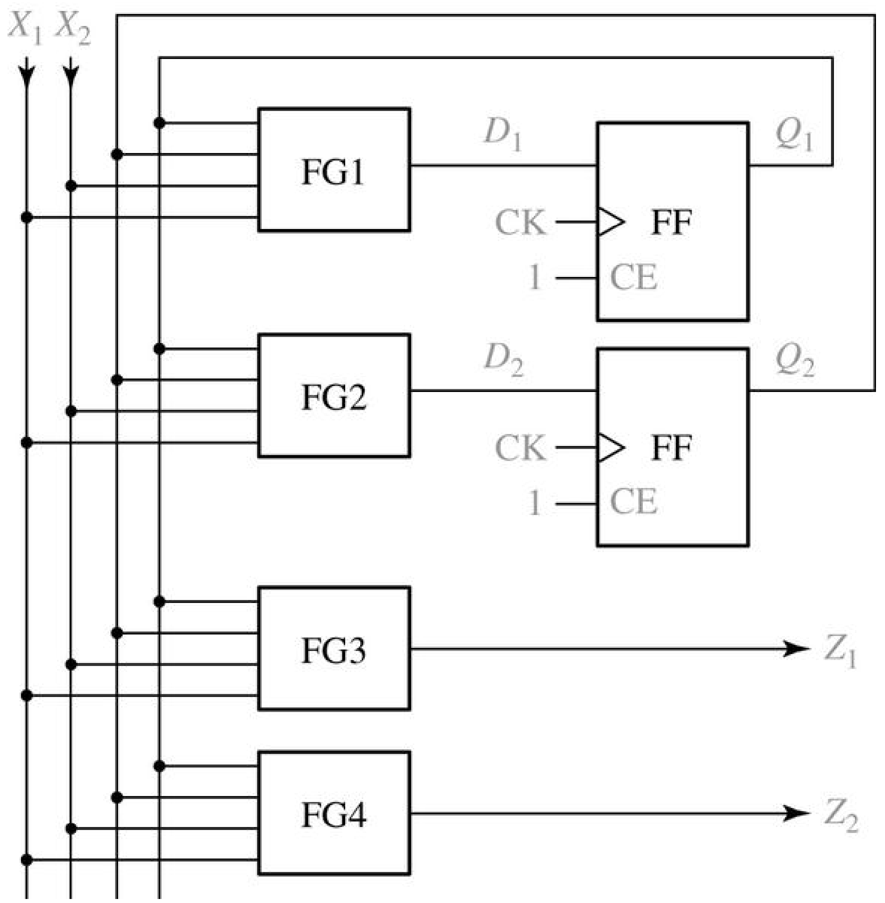


Figure 16-18 shows how a Mealy sequential machine with two inputs, two output, and two flip-flops can be implemented by a FPGA.

# 16.6 Sequential Circuit Design Using FPGAs

Figure 16-19: FPGA Implementation of a Shift Register

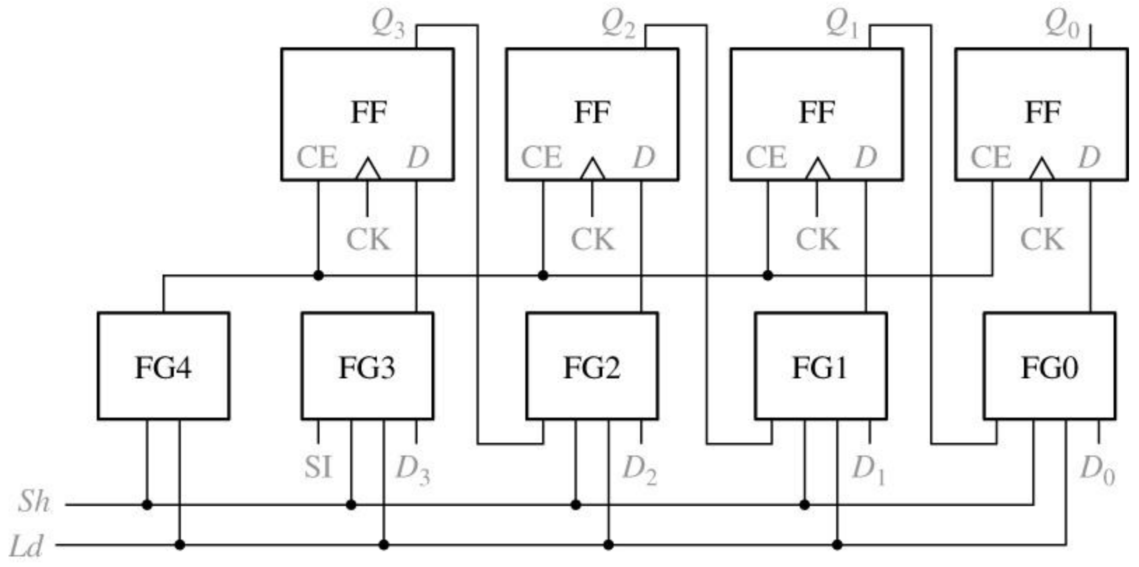


*Figure 16-19 shows how the 4-bit loadable right-shift register Figure 12-15 can be implemented using an FPGA.

$$Q_3^+ = CE'Q_3 + CED_{3f} = (Ld + Sh)(Sh'D_3 + ShSI)$$

$D_{3f}$ *is the D input to flip-flop3 therefore*

$$D_{3f} = Sh'D_3 + ShSI$$

# 16.6 Sequential Circuit Design Using FPGAs

Figure 16-20: FPGA Implementation of a Parallel Adder with Accumulator

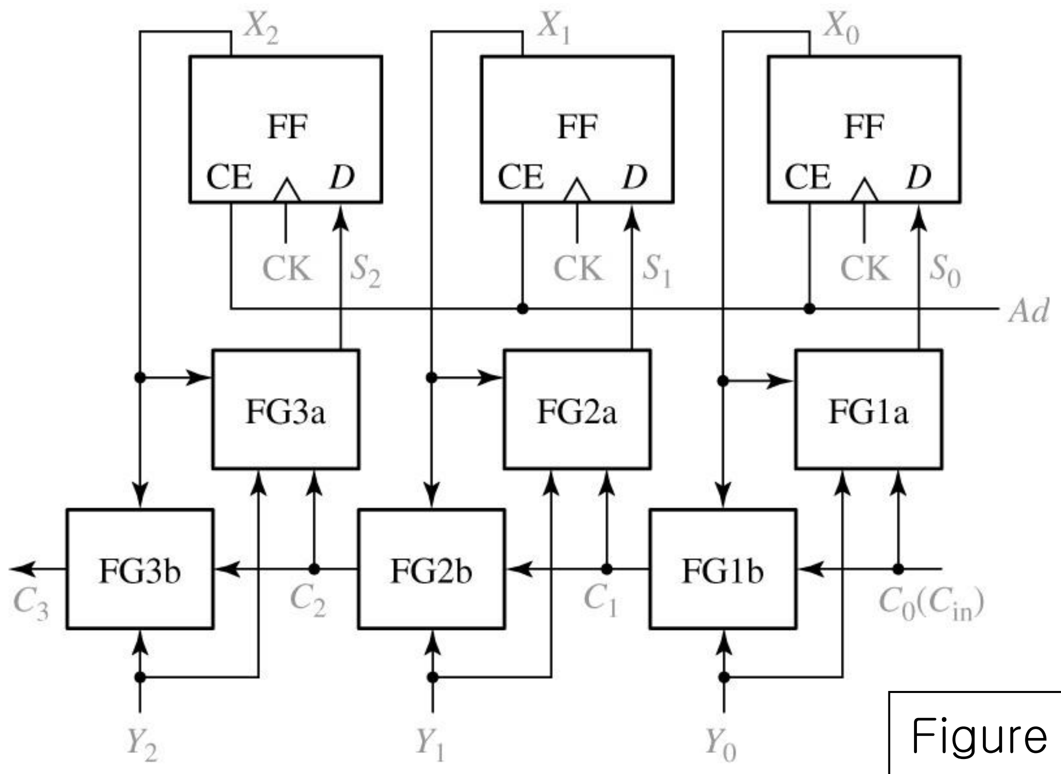


Figure 16-20 shows how three bits of the parallel adder with accumulator of Figure 12-5 can be implemented using an FPGA

## 16.7 Simulation and Testing of Sequential Circuits

Figure 16-21: Simulator Output for an Inverter

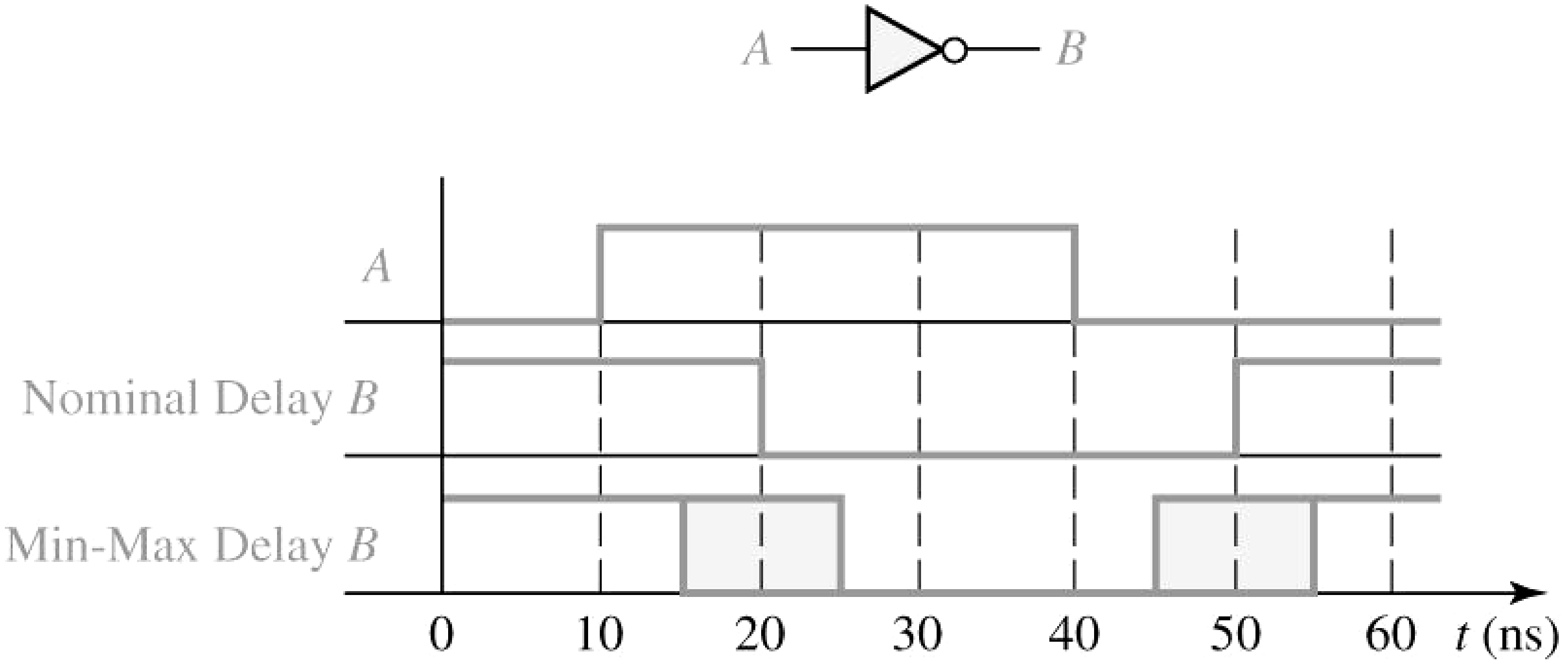


Figure 16-21 shows the output from an inverter which has a nominal delay of 10 ns, a minimum delay of 5 ns, and a maximum delay of 15 ns.

## 16.7 Simulation and Testing of Sequential Circuits

Figure 16-22: Simulation Screen for Figure 13-7

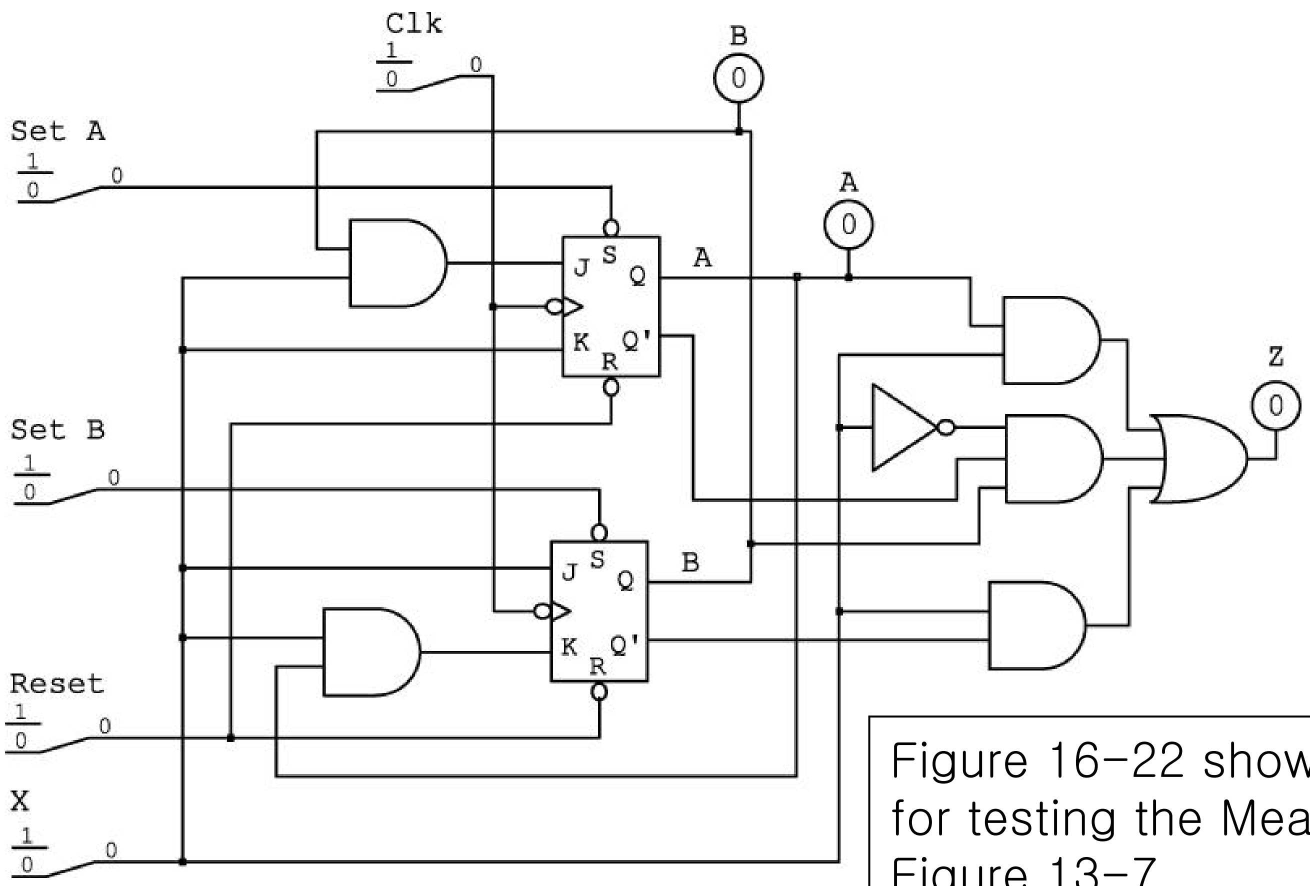
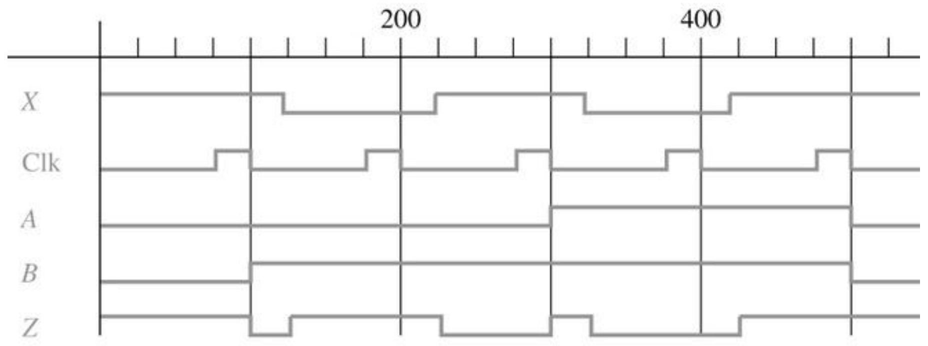


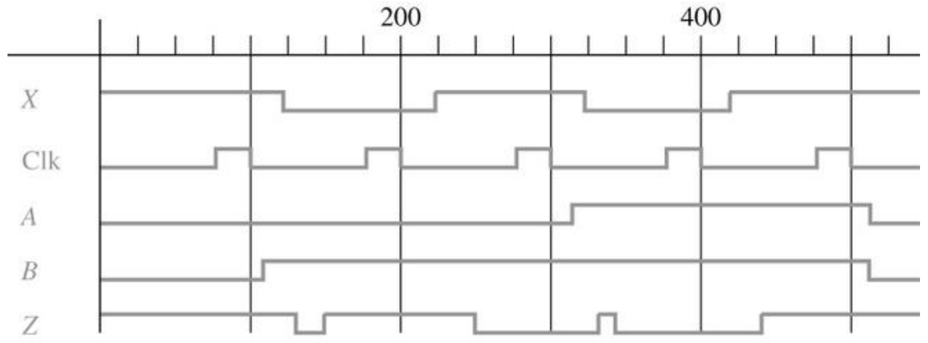
Figure 16-22 shows a simulator screen for testing the Mealy sequential circuit of Figure 13-7.

# 16.7 Simulation and Testing of Sequential Circuits

Figure 16-23



(a) Simulator output with a unit delay model

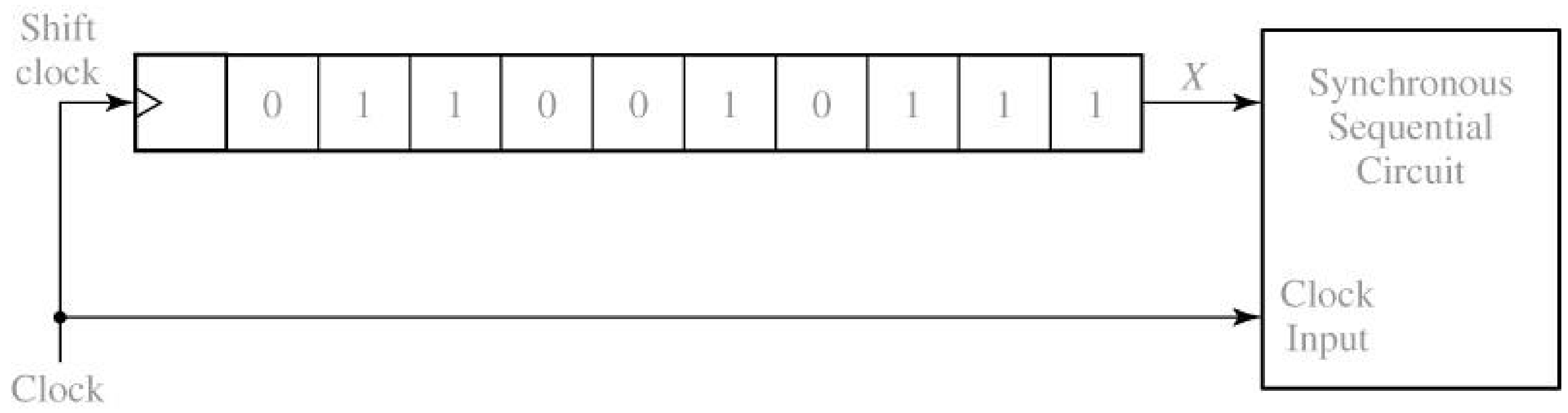


(b) Simulator output with a nominal delay of 10 ns

Figure 16-23 shows the simulator input waveform for the example of Figure 16-22,using the test sequence X=10101.

## 16.7 Simulation and Testing of Sequential Circuits

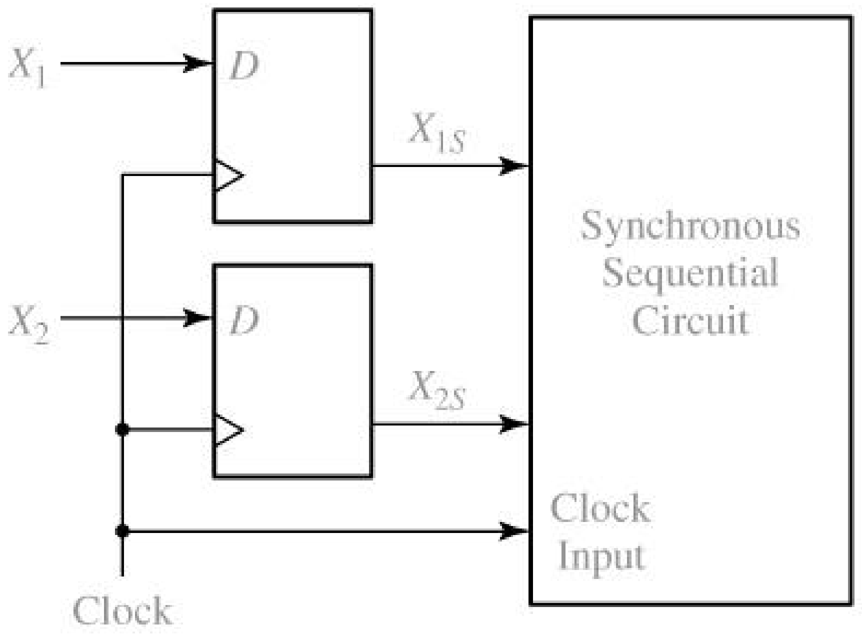
Figure 16-24: Using a Shift Register to Generate Synchronized Inputs



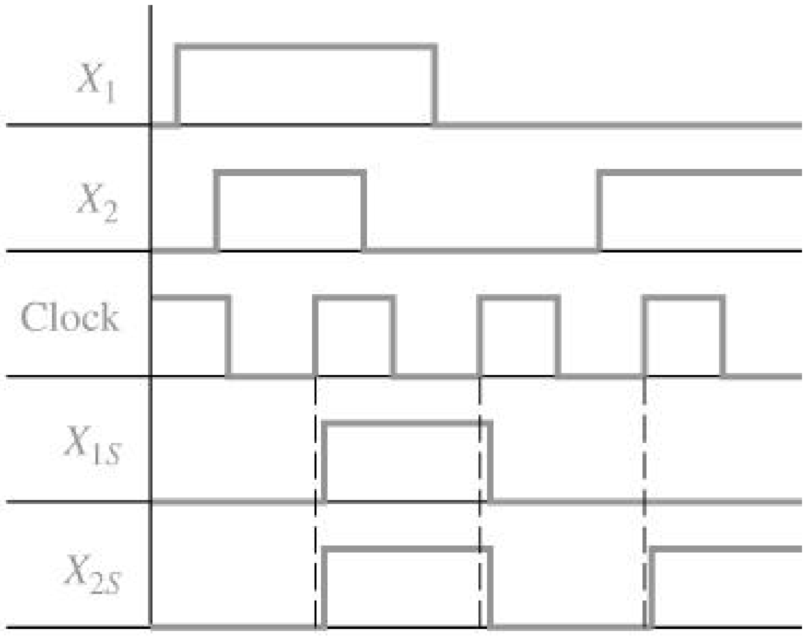
The former can be accomplished by loading the inputs into a shift register, and then using the circuit clock to shift them into the circuit one at a time, as shown in figure 16-24.

## 16.7 Simulation and Testing of Sequential Circuits

Figure 16-25



(a) Synchronizer circuit



(b) Synchronizer inputs and outputs

# 16.7 Simulation and Testing of Sequential Circuits

Figure 16-26: Synchronizer with Two D Flip-Flops

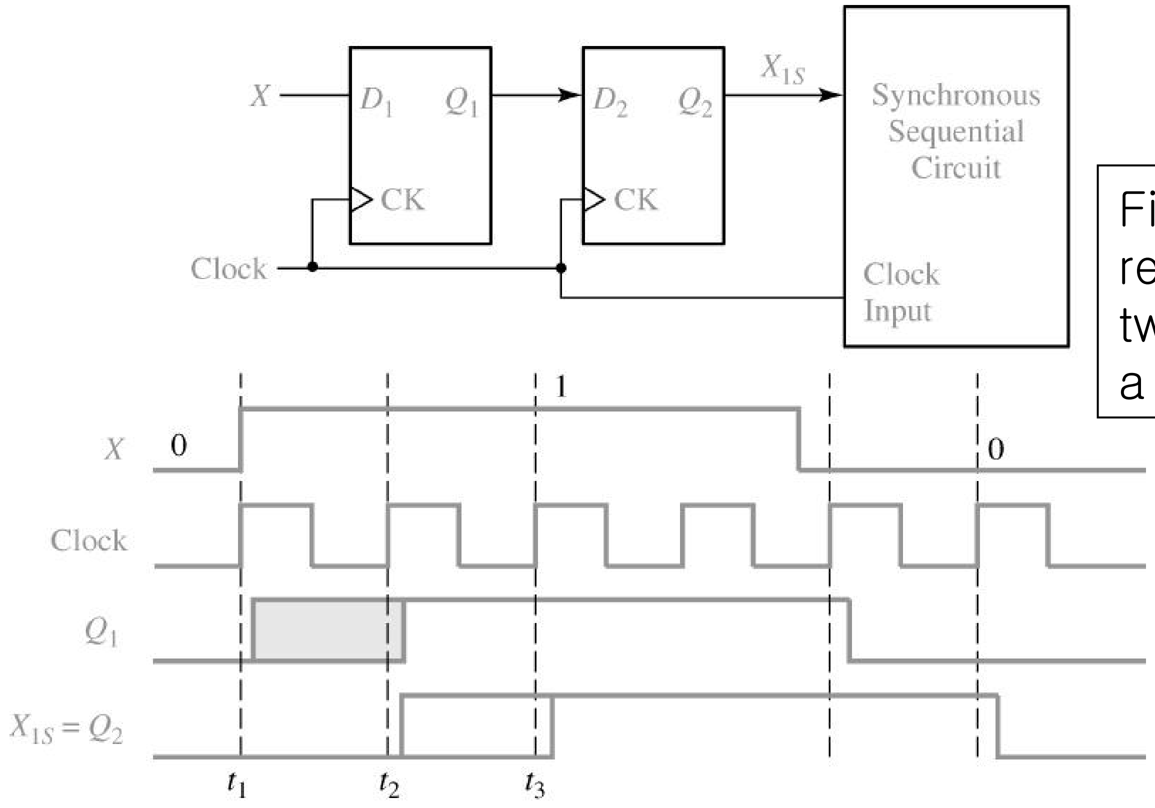


Figure 16-26 shows a more reliable synchronizer that uses two D flip-flop to synchronize a single asynchronous input,X

## 16.7 Overview of Computer-Aided Design

Functions performance of CAD tools

- Generation and Minimization of logic equation
- Generation of bit patterns for programming PLD's
- Schematic Capture
- Simulation
- Synthesis tools
- IC design and Layout
- Test Generation
- PC board Layout

## 16.7 Simulation and Testing of Sequential Circuits

Design a small digital systems with an FPGA

1. Draw a block diagram of the digital system. Define the required control signals and construct state graph and describes the required sequence of operations
2. Workout a detailed logic design using gates, F/F,register, counter,adders, etc... (HDL)
3. Construct a logic diagram using a schematic capture program(HDL)
4. Simulate and debug the logic diagram and make any necessary corrections to the design(HDL)
5. Run an implementation program that fits the design into the target FPGA
6. Simulation and verifying
7. Download the bit pattern into FPGA and test.