Chapter 4
Sequential Circuits

M. Morris Mano, Charles R. Kime. (2015). *Logic and computer design fundamentals* (5th ed.). Pearson.

Contents

1. Sequential Circuit Definitions

- Sequential circuits are formed by interconnecting **a combinational circuit** and **storage elements**.
- The storage elements are circuits that are capable of storing binary information.
  - The binary information stored in these elements at any given time defines the state of the sequential circuit at that time.
  - The outputs and the next state are a function of the inputs and the present state.

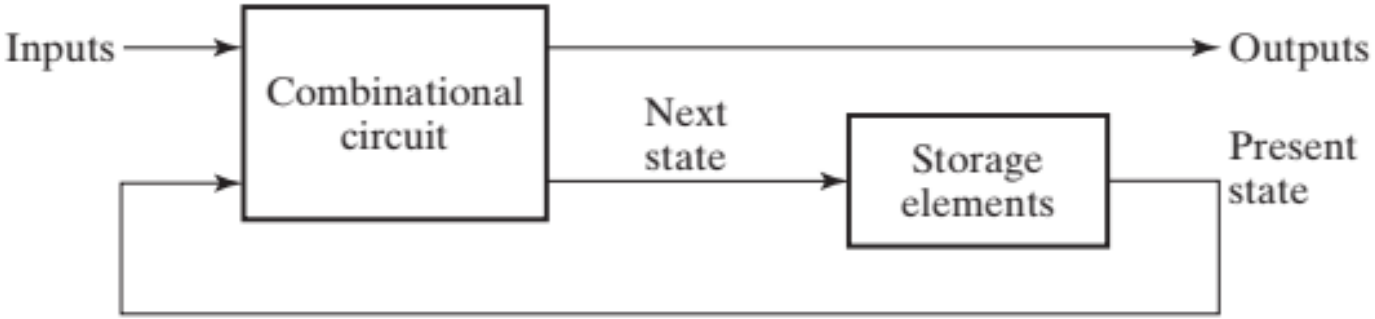


FIGURE 4-1
Block Diagram of a Sequential Circuit

1. Sequential Circuit Definitions

- Information is stored in digital systems in many ways, including the use of logic circuits.
- Figure 4-2(a) shows a buffer. This buffer has a gate delay t_G.
- Since information present at the buffer input at time t appears at the buffer output at time $t + t_G$, the information has effectively been stored for time t_G.
- But, in general, we wish to store information for an indefinite time that is typically much longer than the time delay of one or even many gates. This stored value is to be changed at arbitrary times based on the inputs applied to the circuit and the duration of storage of a value should be longer than the specific time delay of a gate.

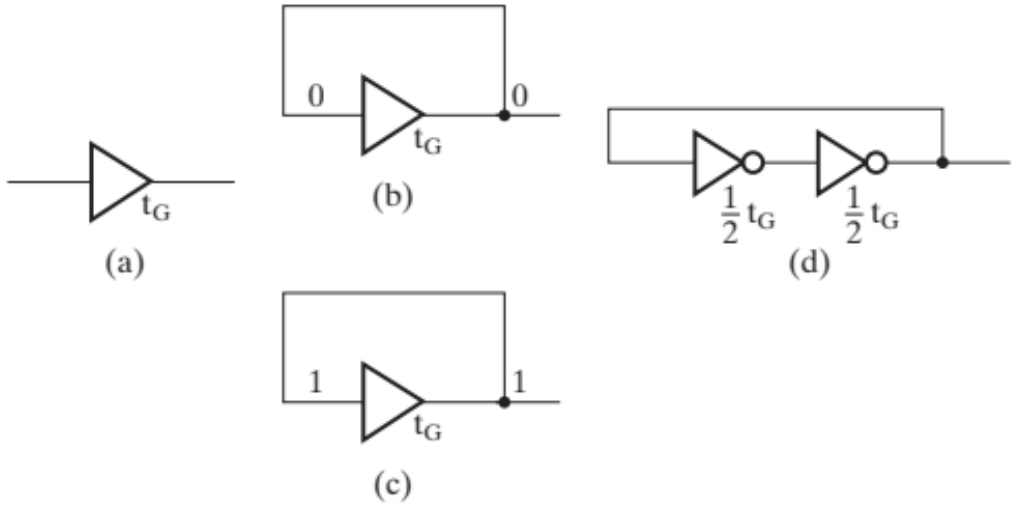


FIGURE 4-2
Logic Structures for Storing Information

1. Sequential Circuit Definitions

- Figure 4-2(b) (c) : The output of the buffer is connected to its input.
- Suppose that the value on the input to the buffer in part (b) has been 0 for at least time t_G, the delay of the buffer. Then the output produced by the buffer will be 0 at time $t + t_G$. This output is applied to the input so that the output will also be 0 at time $t + 2t_G$.
- This relationship between input and output holds for all t, so the 0 will be stored indefinitely.
- The same argument can be made for storing a 1 in the circuit in Figure 4-2(c)

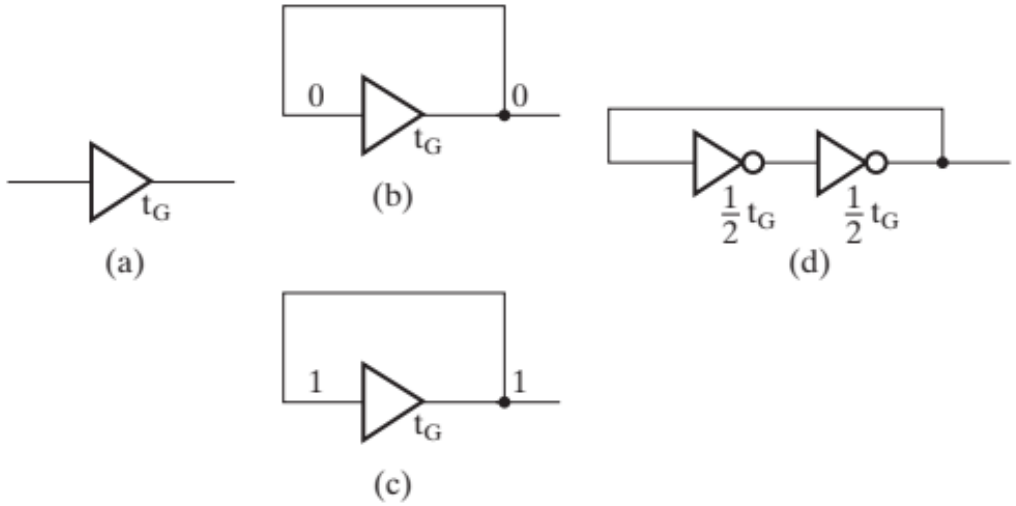
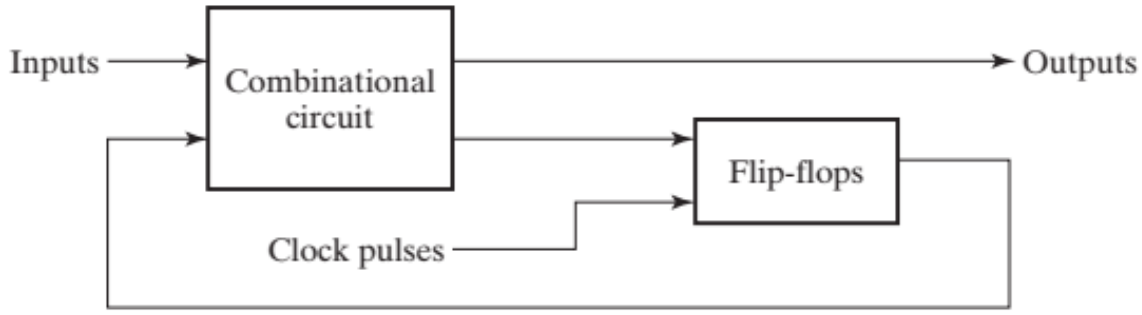


FIGURE 4-2
Logic Structures for Storing Information

Types of Sequential Circuits

- Asynchronous
  - Storage elements can change their state at any instant of time as soon as the input(s) changes. (latches)
- Synchronous
  - Storage elements observe inputs and can change state only in relation to a timing signal (clock pulses from a clock) (flip-flops)



(a) Block diagram

(b) Timing diagram of clock pulses

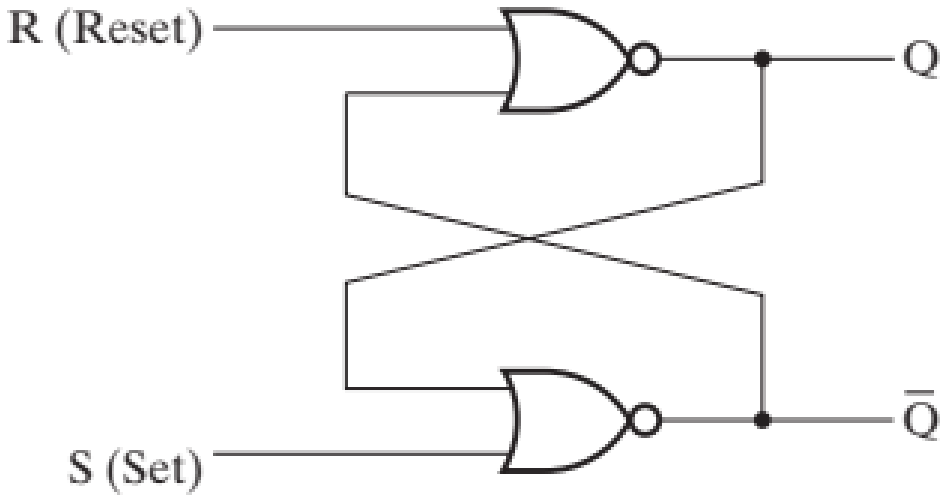
FIGURE 4-3
Synchronous Clocked Sequential Circuit

Comparison of Sequential Circuits

- Asynchronous
  - Potentially faster
  - Harder to analyze
- Synchronous
  - Easier to analyze
  - Choose the clock so that changes are only allowed to occur before next clock pulse

2. Latches

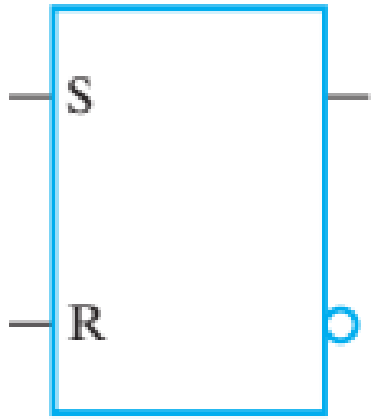
- *SR* Latches



(a) Logic diagram

S	R	Q	$\bar{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

(b) Function table

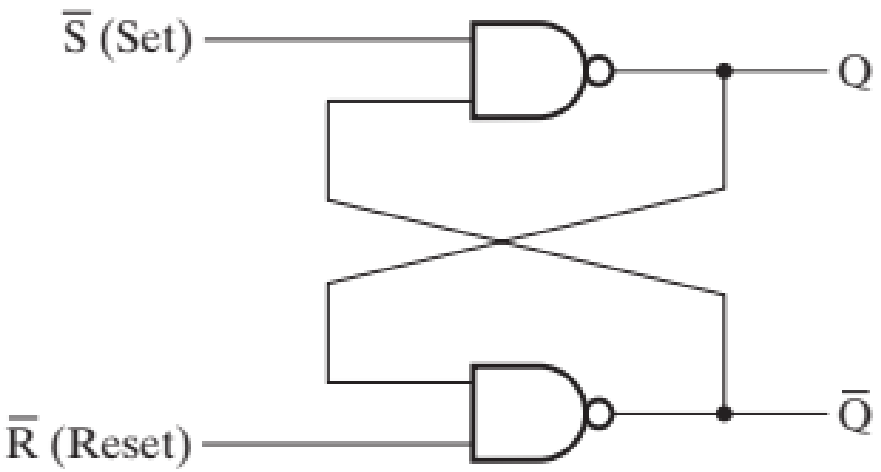


SR

FIGURE 4-4
SR Latch with NOR Gates

2. Latches

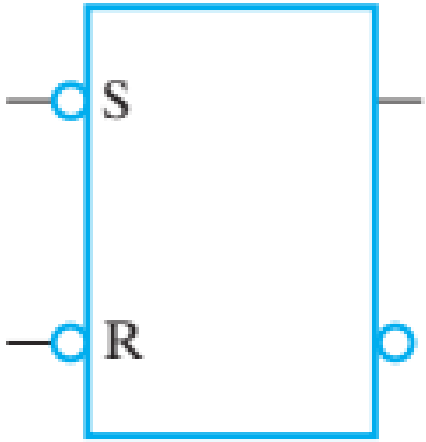
- $\overline{S}\overline{R}$ Latches



(a) Logic diagram

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

(b) Function table

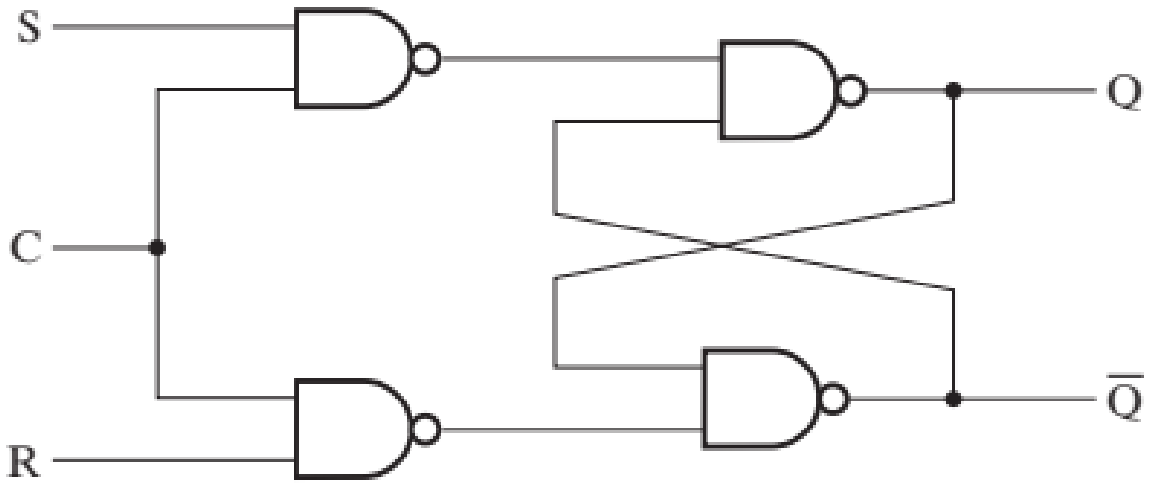


$\overline{S}\overline{R}$

FIGURE 4-6
$\overline{S}\,\overline{R}$ Latch with NAND Gates

2.Latches

- *SR* Latch with Control Input



(a) Logic diagram

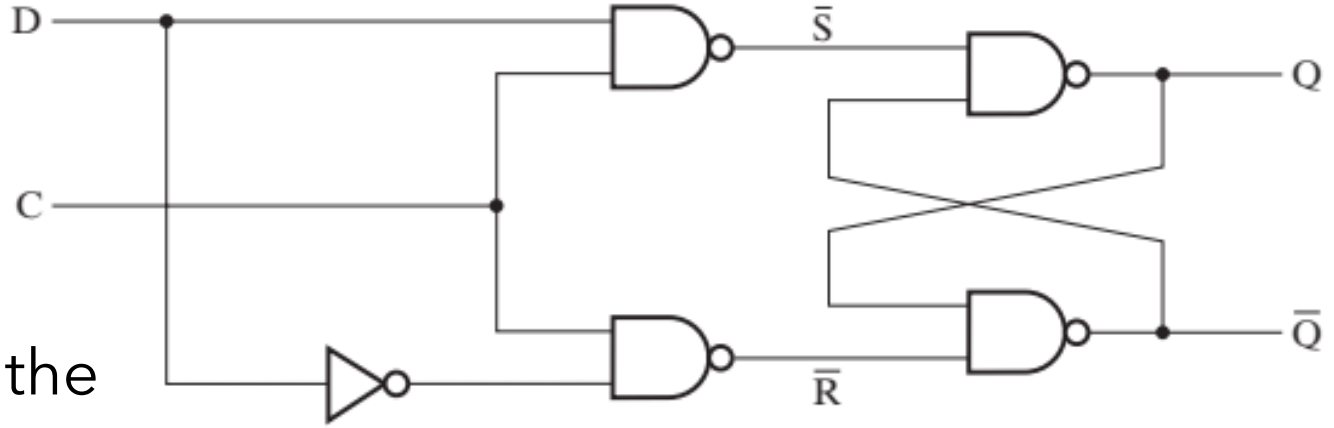
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table

FIGURE 4-7
SR Latch with Control Input

2. Latches

- *D* Latches
  - To eliminate the undesirable undefined state in the SR latch is to ensure that inputs *S* and *R* are never equal to 1 at the same time.

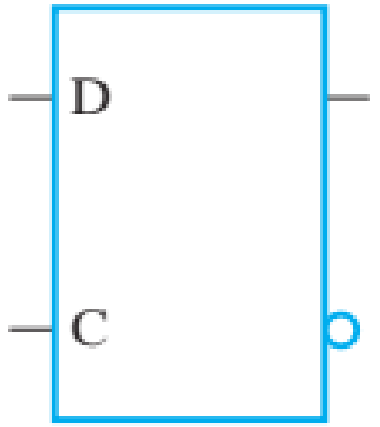


(a) Logic diagram

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

(b) Function table

FIGURE 4-8
D Latch



D with 1 Control

3. Flip-Flops

- Note that the problem with the latch is that it is transparent:
  - As soon as an input changes, shortly thereafter the corresponding output changes to match it. This transparency is what allows a change on a latch output to produce additional changes at other latch outputs while the clock pulse is at logic 1.
  - The key to the proper operation of flip-flops is to prevent them from being transparent. In a flip-flop, before an output can change, the path from its inputs to its outputs is broken.

3. Flip-Flops

- A common way to create a flip-flop is to connect two latches which is often referred to as a *master–slave* flip-flop.
  - The left latch, the master, changes its value based upon the input while the clock is high. That value is then transferred to the right latch, the slave, when the clock changes to low.

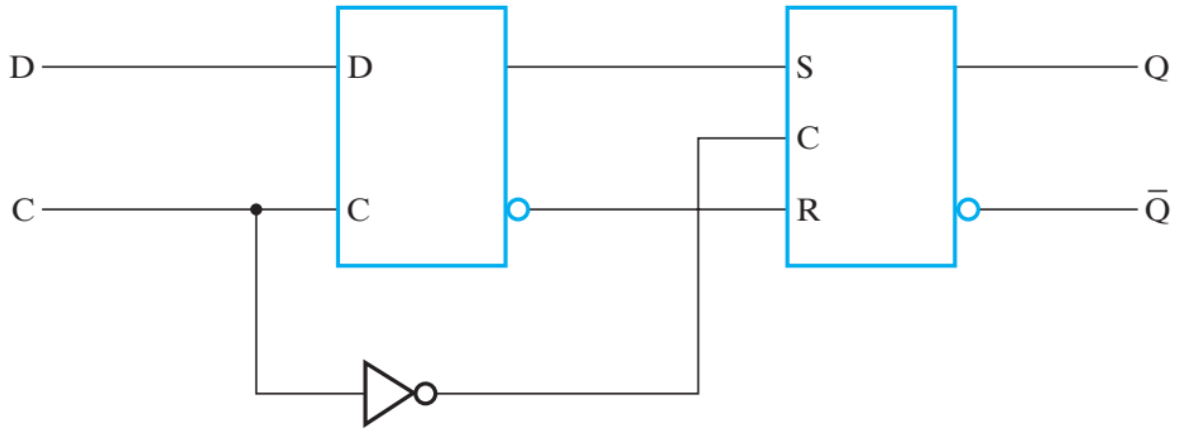
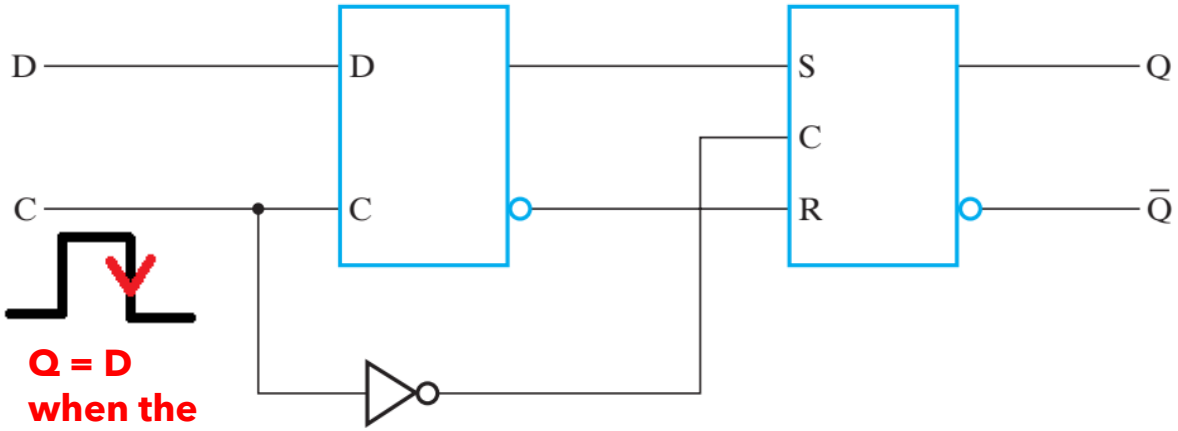


FIGURE 4-9
Negative-Edge-Triggered D Flip-Flop

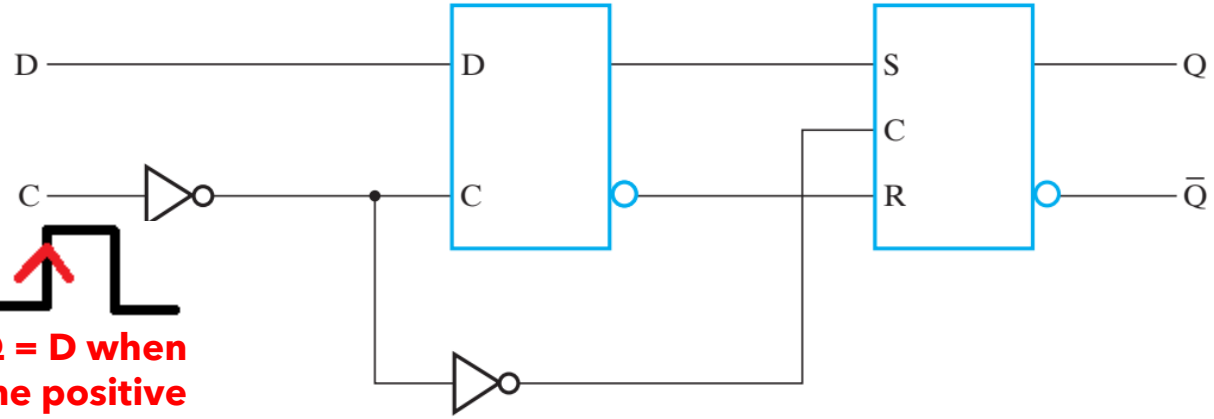
Edge-Triggered Flip-Flop

- Edge-Triggered Flip-Flop: A flip-flop that triggers only during a signal *transition* from 0 to 1 (or from 1 to 0) on the clock and that is disabled at all other times, including for the duration of the clock pulse.



Q = D when the negative edge occurs.

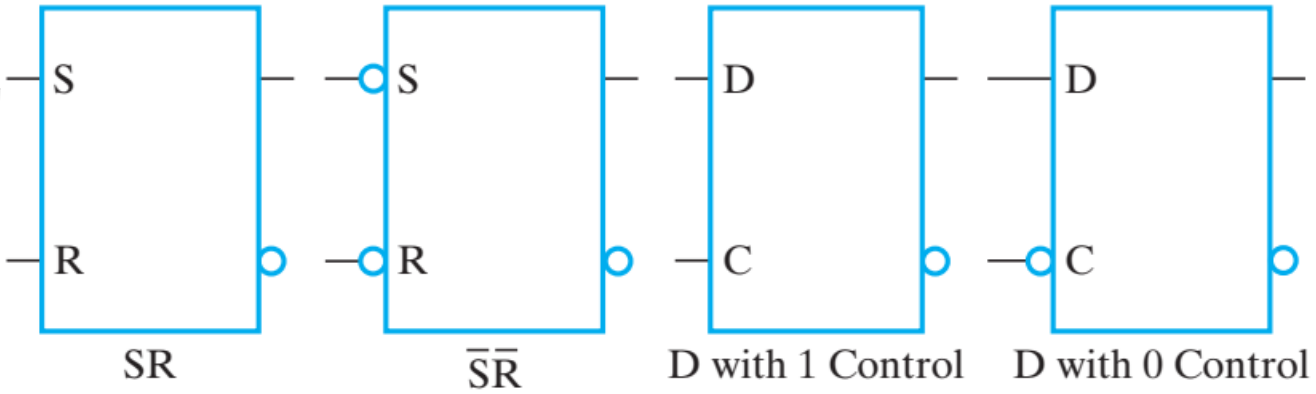
FIGURE 4-9
Negative-Edge-Triggered D Flip-Flop



Q = D when the positive edge occurs.

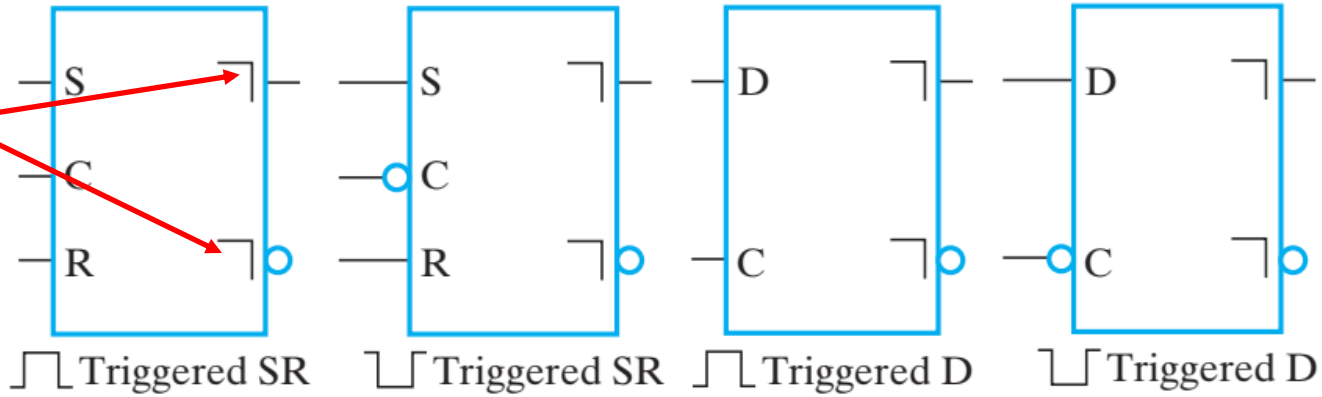
FIGURE 4-10
Positive-Edge-Triggered D Flip-Flop

Standard Graphics Symbols



(a) Latches

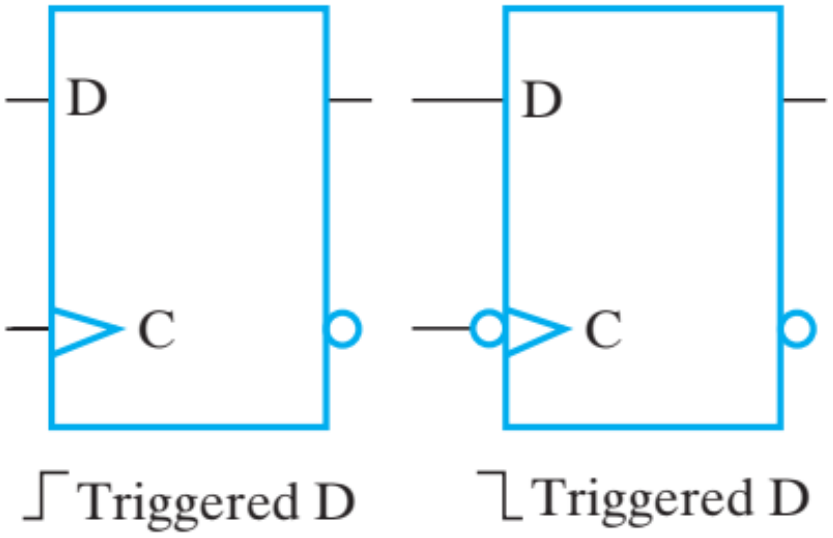
Postponed output indicators



(b) Master–slave flip-flops

The pulse-triggered flip-flop is indicated as such with a right-angle symbol called a *postponed output indicator* in front of the outputs. This symbol shows that the output signal changes at the end of the pulse.

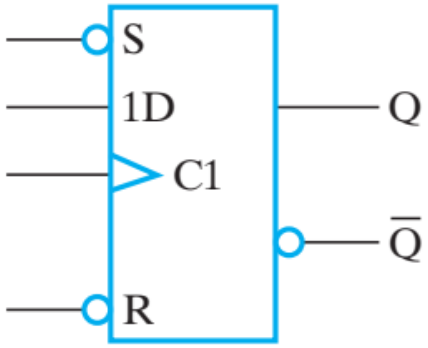
Standard Graphics Symbols



(c) Edge-triggered flip-flops

Direct Inputs

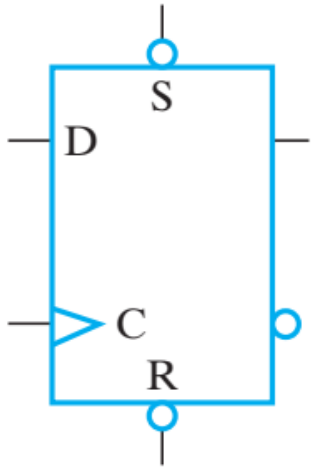
- Flip-flops often provide special inputs for setting and resetting them asynchronously (i.e., independently of the clock input *C*).
  - *Direct set* or *preset*: asynchronously set the flip-flop
  - *Direct reset* or *clear*: asynchronously reset the flip-flop



(a) Graphic symbol

S	R	C	D	Q	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	Undefined	
1	1	↑	0	0	1
1	1	↑	1	1	0

(b) Function table



(c) Simplified symbol

FIGURE 4-12
D Flip-Flop with Direct Set and Reset

4. Sequential Circuit Analysis

- The outputs and the next state are a function of the inputs and the present state.
  - Input X
  - State (A, B)
  - Output Y
- **Input Equations**
  - The part of the combinational circuit that generates the signals for the inputs of flip-flops can be described by a set of Boolean functions called *flip-flop input equations*.

$$D_A = AX + BX$$

$$D_B = \overline{A}X$$

$$Y = (A + B)\overline{X}$$

D_A : Flip-flop input of FF A
D_B: Flip-flop input of FF B

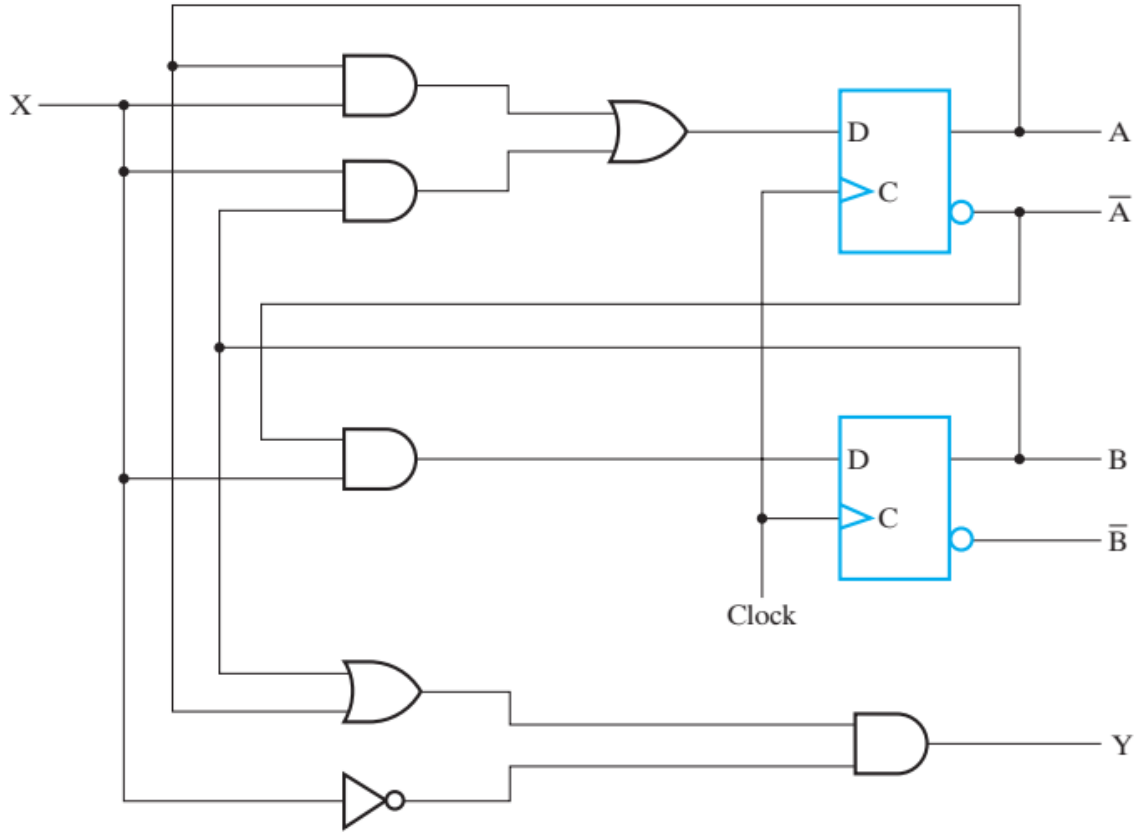
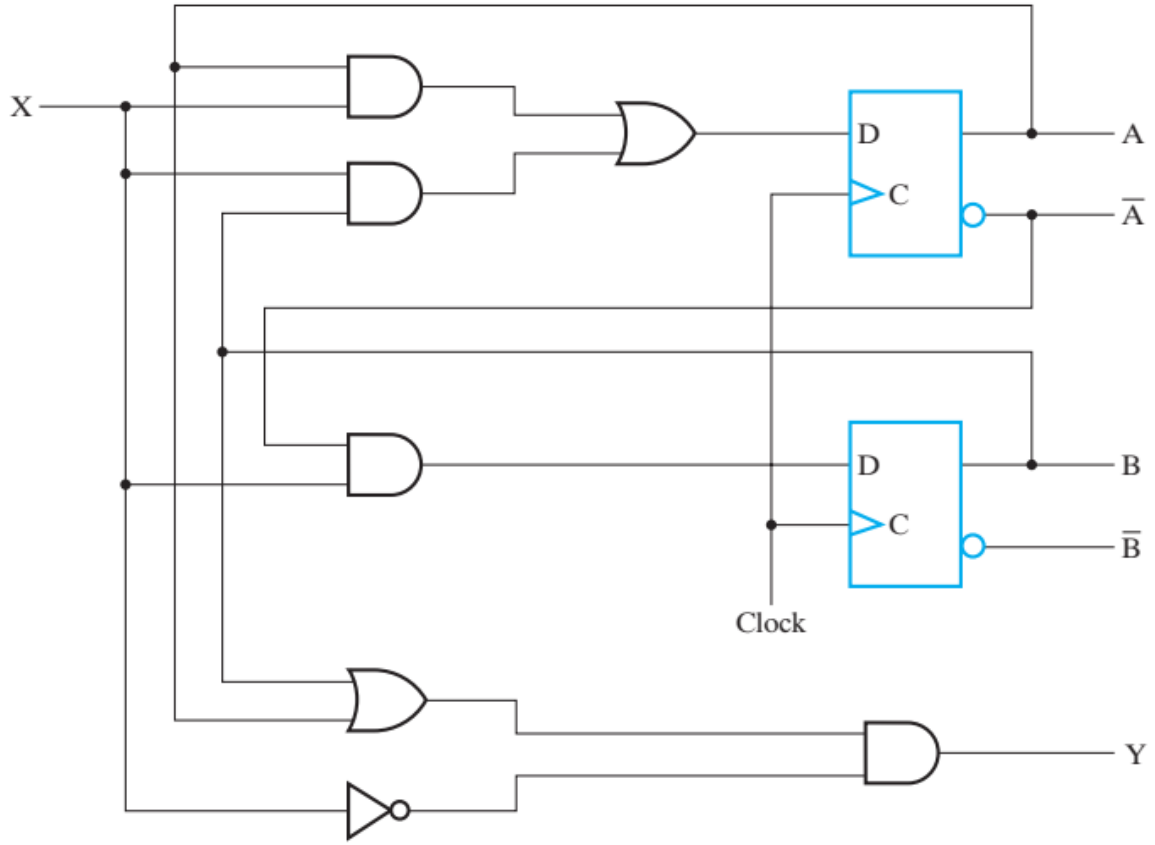


FIGURE 4-13
Example of a Sequential Circuit

4. Sequential Circuit Analysis



$$A(t+1) = D_A = AX + BX$$
$$B(t+1) = D_B = \overline{A}X$$
$$Y = A\overline{X} + B\overline{X}$$

$$D_A = AX + BX$$
$$D_B = \overline{A}X$$
$$Y = (A + B)\overline{X}$$

- **State Table**
  - The functional relationships among the inputs, outputs, and flip-flop states of a sequential circuit can be enumerated in a *state table*.

TABLE 4-1
State Table for Circuit of Figure 4-13

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

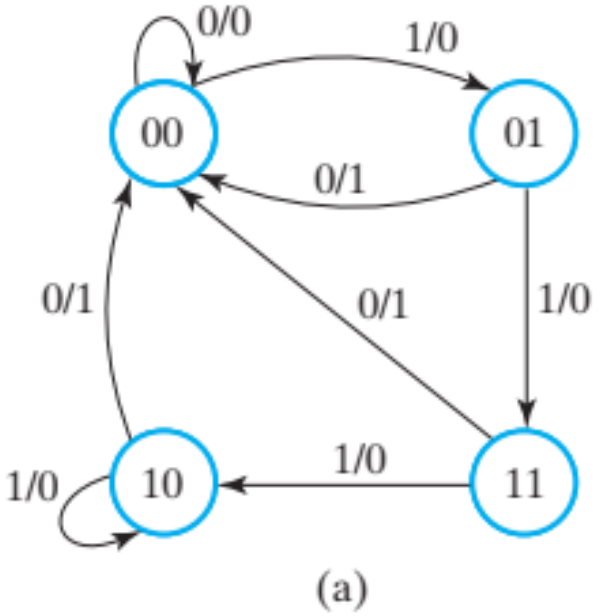
In general, a sequential circuit with m flip-flops and n inputs needs 2^{m+n} rows in the state table.

State Diagram

- The information available in a state table may be represented graphically in the form of a state diagram.
- A state is represented by a circle, and transitions between states are indicated by directed lines connecting the circles.
- The binary number inside each circle identifies the state of the flip-flops.

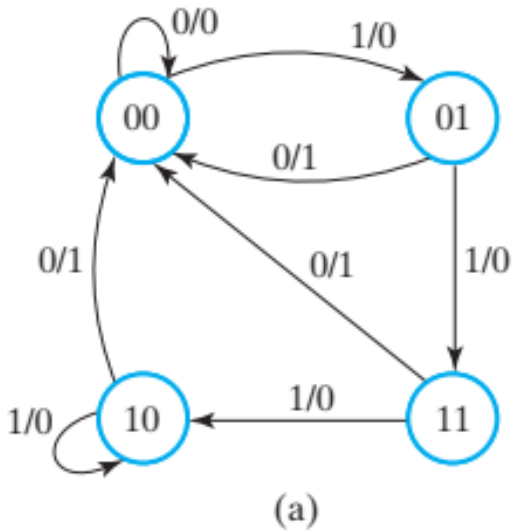
TABLE 4-1
State Table for Circuit of Figure 4-13

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



Equivalent State Definition

- Two states are *equivalent* if the output produced for each input symbol is identical and the next states for each input symbol are the same or equivalent.
  - These equivalent states can be merged into a single state.



- States 11 and 10 are equivalent.
- States 01 and 11 are equivalent.
  - Three states 11, 10, 01 are equivalent and can be merged into a single state.

5. Sequential Circuit Design

- A synchronous sequential circuit is made up of flip-flops and combinational gates.
- The minimum number of flip-flops is determined by the number of states in the circuit; *n* flip-flops can represent up to 2^n binary states.
- The combinational circuit is derived from the state table by finding the flip-flop input equations and output equations.

Design Procedure

- Design Procedure

  1. Specification: Write a specification for the circuit, if not already available.

  2. Formulation: Obtain either a state diagram or a state table from the statement of the problem.

  3. State Assignment: If only a state diagram is available from step 2, obtain the state table. Assign binary codes to the states in the table.

  4. Flip-Flop Input Equation Determination: Select the flip-flop type or types. Derive the flip-flop input equations from the next-state entries in the encoded state table.

  5. Output Equation Determination: Derive output equations from the output entries in the state table.

  6. Optimization: Optimize the flip-flop input equations and output equations.

  7. Technology mapping: Draw a logic diagram of the circuit using flip-flops, ANDs, ORs, and inverters. Transform the logic diagram to a new diagram using the available flip-flop and gate technology.

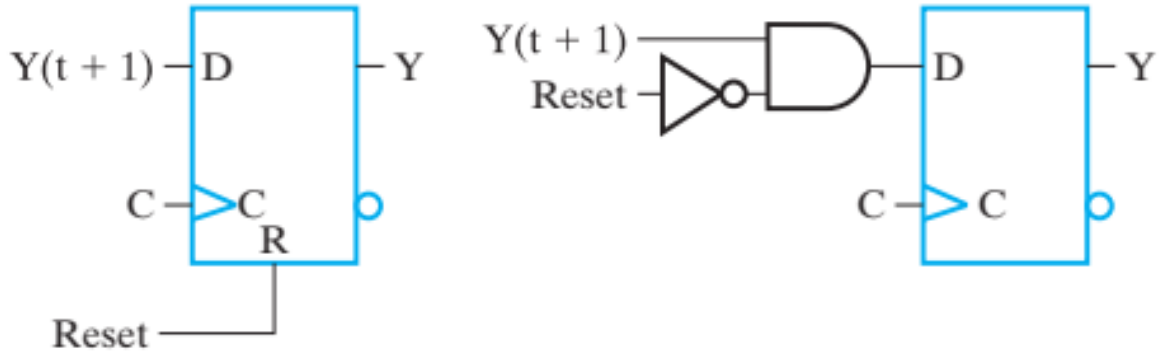
  8. Verification: Verify the correctness of the final design.

Finding State Diagrams and State Tables

- Fundamental to the formulation of state diagrams and tables is an intuitive understanding of the concept of a state.
- A state is used to “remember” something about the history of input combinations applied to the circuit either at triggering clock edges or during triggering pulses.
- In most cases, however, a state is an *abstraction* of the sequence of input combinations at the triggering points.

Finding State Diagrams and State Tables

- When the power in a digital system is first turned on, the state of the flip-flops is unknown.
  - The circuits must have a known *initial state (reset state)*.
  - Reset places the circuit in its initial state.

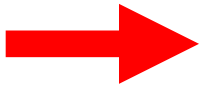


(a) Asynchronous Reset (b) Synchronous Reset

FIGURE 4-17
Asynchronous and Synchronous Reset for *D* Flip-flops

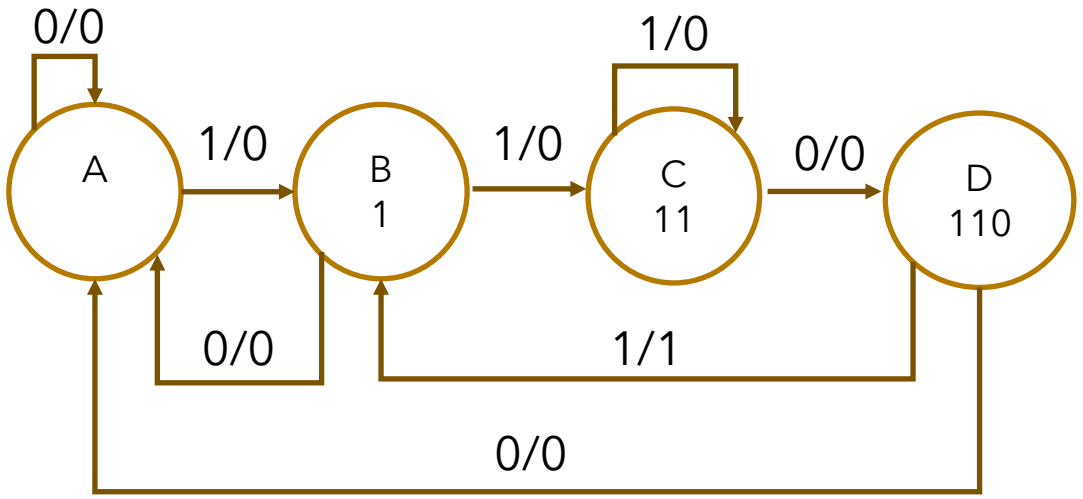
Finding State Diagrams and State Tables

- Example 4.3: Finding a State Diagram for a Sequence Recognizer
  - This "sequence recognizer" has one input X and one output Z.
  - It has **Reset** applied to the direct reset inputs on its flip-flops to initialize the state of the circuit to all zeros.
  - The circuit is to recognize the occurrence of the sequence of bits **1101** on X by making Z equal to 1 when the previous three inputs to the circuit were 110 and current input is a 1. Otherwise, Z equals 0.



Initial state *A*
States are used to "remember" the previous three inputs in the sequence.

Example 4.3: Finding a State Diagram for a Sequence Recognizer (**1101**)



Present State	X	Next State	Z
A	0	A	0
A	1	B	0
B	0	A	0
B	1	C	0
C	0	D	0
C	1	C	0
D	0	A	0
D	1	B	1

State Assignment

- In contrast to the states in the analysis examples, the states in the diagrams constructed have been assigned symbolic names rather than binary codes. It is necessary to replace these symbolic names with binary codes in order to proceed with the design.
- In general, if there are m states, then the codes must contain at least n bits, where $2^n \geq m$, and each state must be assigned a unique code. ($2^n \geq m > 2^{n-1}$)

State Assignment

- The code words are assigned in counting order. For example, for states A, B, C, and D, the codes 00, 01, 10, and 11 are assigned to A, B, C, and D, respectively.
- An alternative that is attractive, particularly if K-maps are being used for optimization, is to assign the codes in Gray code order, with codes 00, 01, 11, and 10 assigned to A, B, C, and D, respectively.
- There are a number of specialized state assignment methods, some of which are based on efficient structures for implementing at least a portion of the transitions. The most popular of these methods is **the one flip-flop per state or one-hot assignment**. This assignment uses a distinct flip-flop for each of the m states, so it generates codes that are m bits long. The sequential circuit is in a state when the flip-flop corresponding to that state contains a 1. By definition, all flip-flops corresponding to the other states must contain 0. Thus, each valid state code contains m bits, with one bit equal to 1 and all other $m - 1$ bits equal to 0.

Example 4.3: Finding a State Diagram for a Sequence Recognizer (**1101**)

- State names replaced by a 2-bit binary Gray code
  - 4 States → 2 flip-flops
  - A: 00 B: 01 C: 11 D: 10

Present State	X	Next State	Z
A	0	A	0
A	1	B	0
B	0	A	0
B	1	C	0
C	0	D	0
C	1	C	0
D	0	A	0
D	1	B	1

Present State	X	Next State	Z
00	0	00	0
00	1	01	0
01	0	00	0
01	1	11	0
11	0	10	0
11	1	11	0
10	0	00	0
10	1	01	1

Example 4.3: Finding a State Diagram for a Sequence Recognizer (**1101**)

- State names replaced by a 4-bit one-hot code
  - 4 states → 4 flip-flops
  - A: 1000 B: 0100 C: 0010 D: 0001

Present State	X	Next State	Z
A	0	A	0
A	1	B	0
B	0	A	0
B	1	C	0
C	0	D	0
C	1	C	0
D	0	A	0
D	1	B	1

Present State	X	Next State	Z
1000	0	1000	0
1000	1	0100	0
0100	0	1000	0
0100	1	0010	0
0010	0	0001	0
0010	1	0010	0
0001	0	1000	0
0001	1	0100	1

Designing with D Flip-Flops

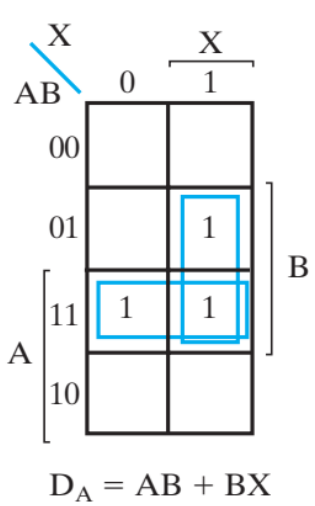
- For the Gray-coded design, two flip-flops are needed to represent the four states.
  - Flip-flop A and B

Present State AB	X	Next State AB	Z
00	0	00	0
00	1	01	0
01	0	00	0
01	1	11	0
11	0	10	0
11	1	11	0
10	0	00	0
10	1	01	1

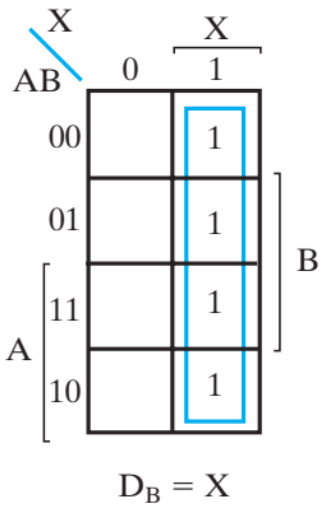
$$A(t+1) = D_A(A, B, X) = \Sigma m(3, 6, 7)$$

$$B(t+1) = D_B(A, B, X) = \Sigma m(1, 3, 5, 7)$$

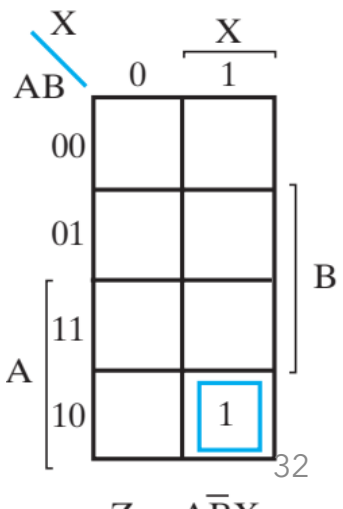
$$Z(A, B, X) = \Sigma m(5)$$



$D_A = AB + BX$



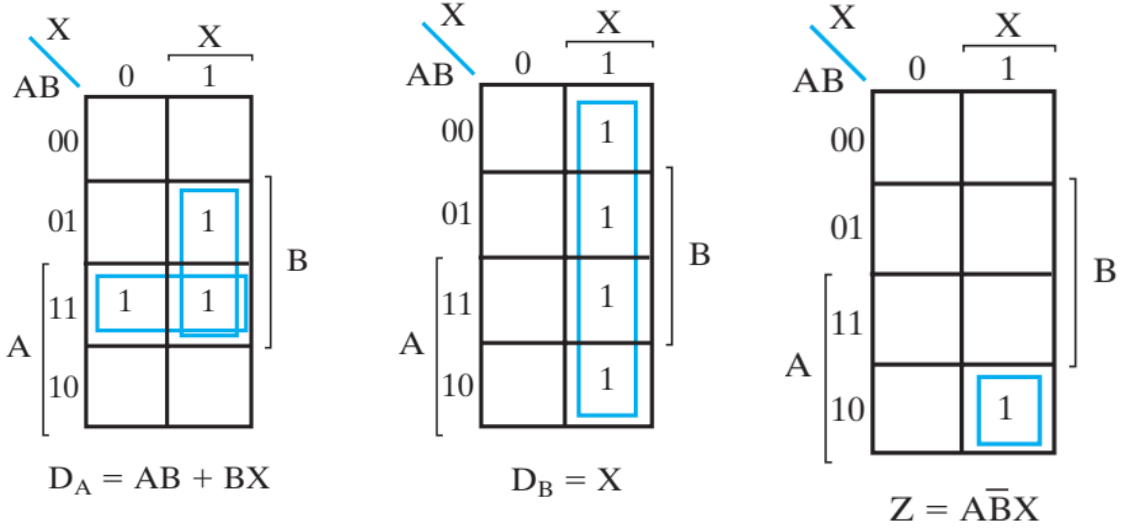
$D_B = X$



$Z = A\bar{B}X$

Designing with D Flip-Flops

- For the Gray-coded design, two flip-flops are needed to represent the four states.
  - Flip-flop A and B



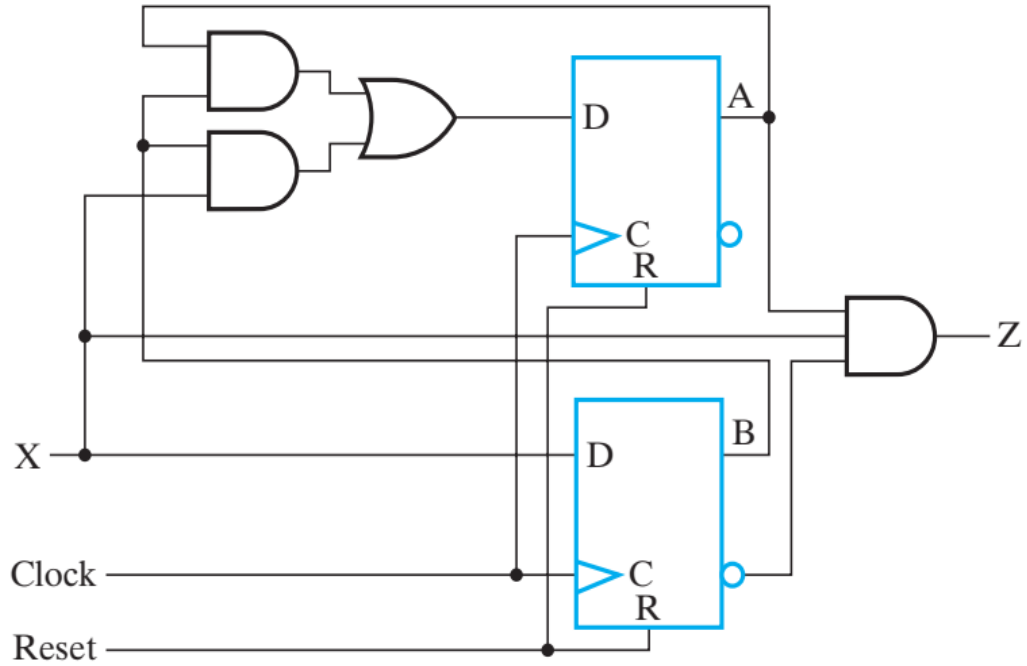
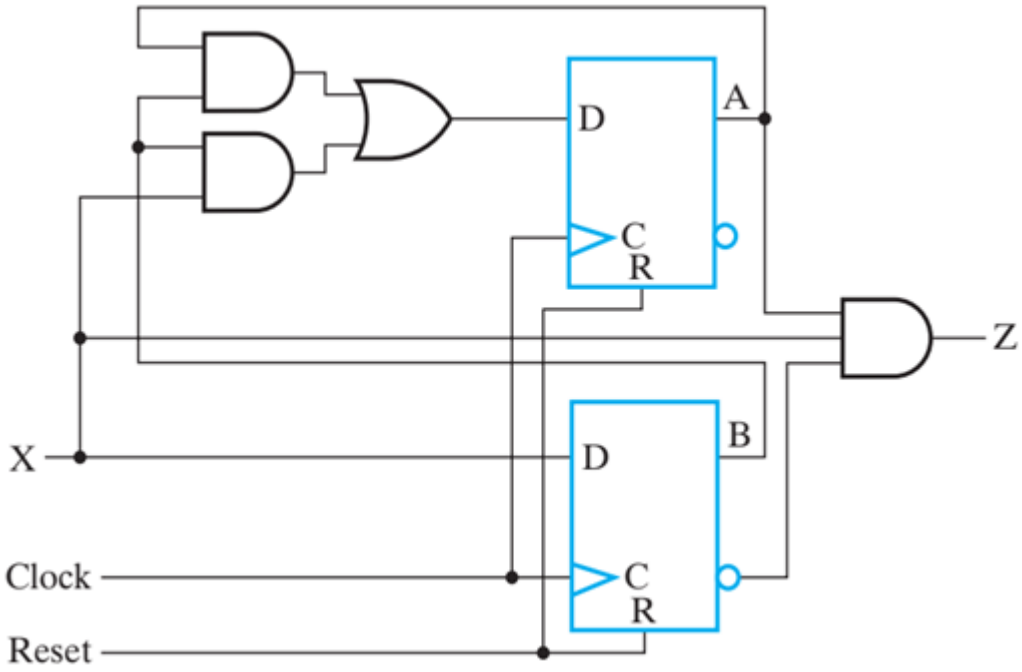
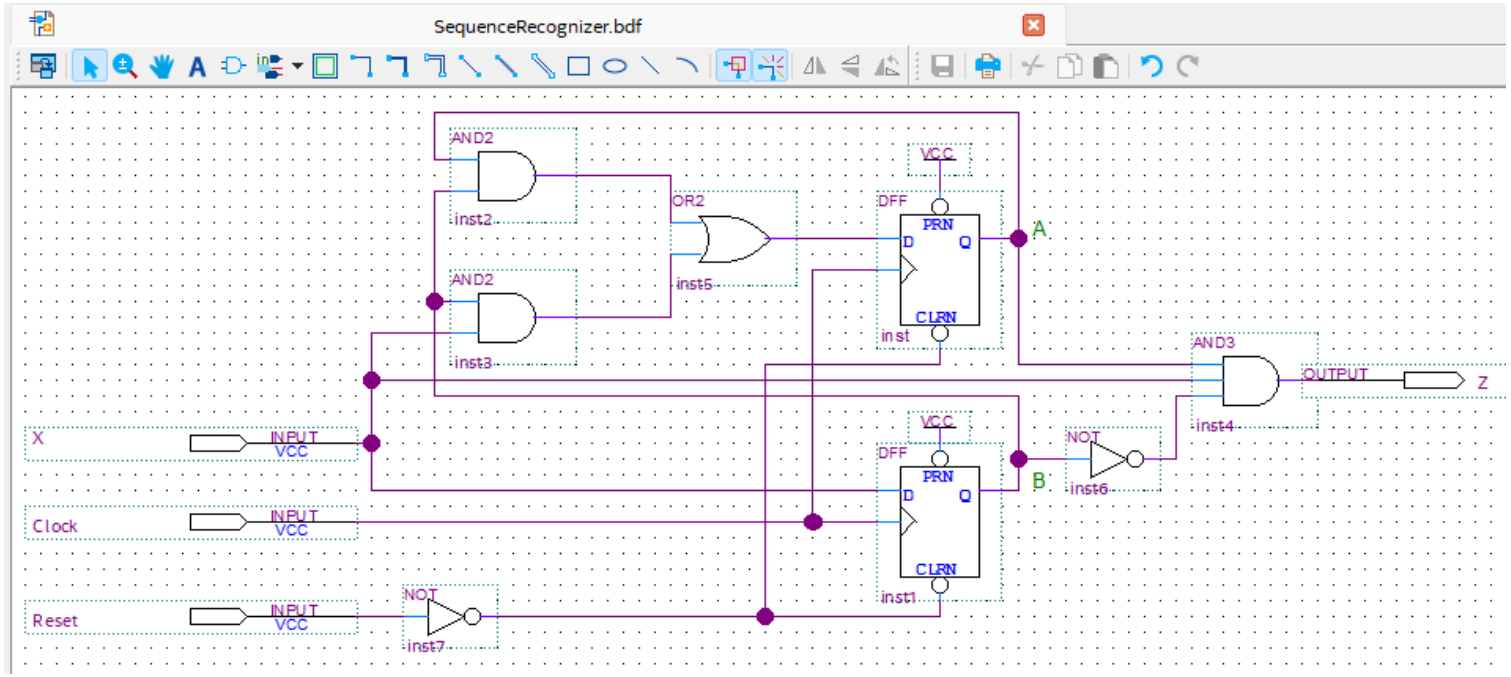


FIGURE 4-21
Logic Diagram for the Gray-Coded Sequence Recognizer with *D* Flip-Flops

Designing with D Flip-Flops

- For the Gray-coded design, two flip-flops are needed to represent the four states.
  - Flip-flop A and B





Designing with D Flip-Flops

- For the one-hot code design, four flip-flops are needed to represent the four states.
  - Flip-flop A, B, C and D

Present State	X	Next State	Z
1000	0	1000	0
1000	1	0100	0
0100	0	1000	0
0100	1	0010	0
0010	0	0001	0
0010	1	0010	0
0001	0	1000	0
0001	1	0100	1

$$A(t+1) = D_A = A\overline{X} + B\overline{X} + D\overline{X} = (A + B + D)\overline{X}$$

$$B(t+1) = D_B = AX + DX = (A + D)X$$

$$C(t+1) = D_C = BX + CX = (B + C)X$$

$$D(t+1) = D_D = C\overline{X}$$

$$Z = DX$$

Designing with D Flip-Flops

- For the one-hot code design, four flip-flops are needed to represent the four states.
  - Flip-flop A, B, C and D

$$A(t+1) = D_A = A\overline{X} + B\overline{X} + D\overline{X} = (A + B + D)\overline{X}$$

$$B(t+1) = D_B = AX + DX = (A + D)X$$

$$C(t+1) = D_C = BX + CX = (B + C)X$$

$$D(t+1) = D_D = C\overline{X}$$

$$Z = DX$$

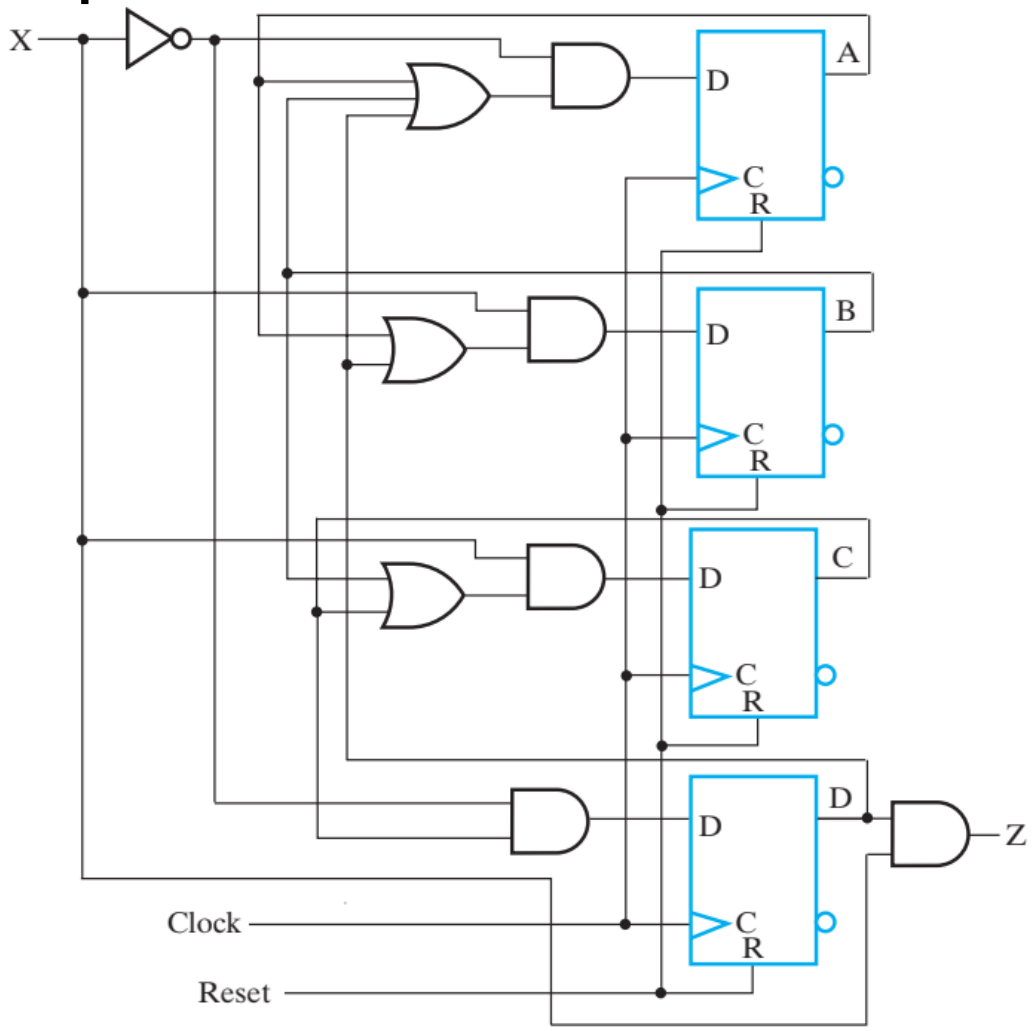


FIGURE 4-22
Logic Diagram for the One-Hot Coded Sequence Recognizer with *D* Flip-Flops