

Fig. 4-1 Block Diagram of a Sequential Circuit

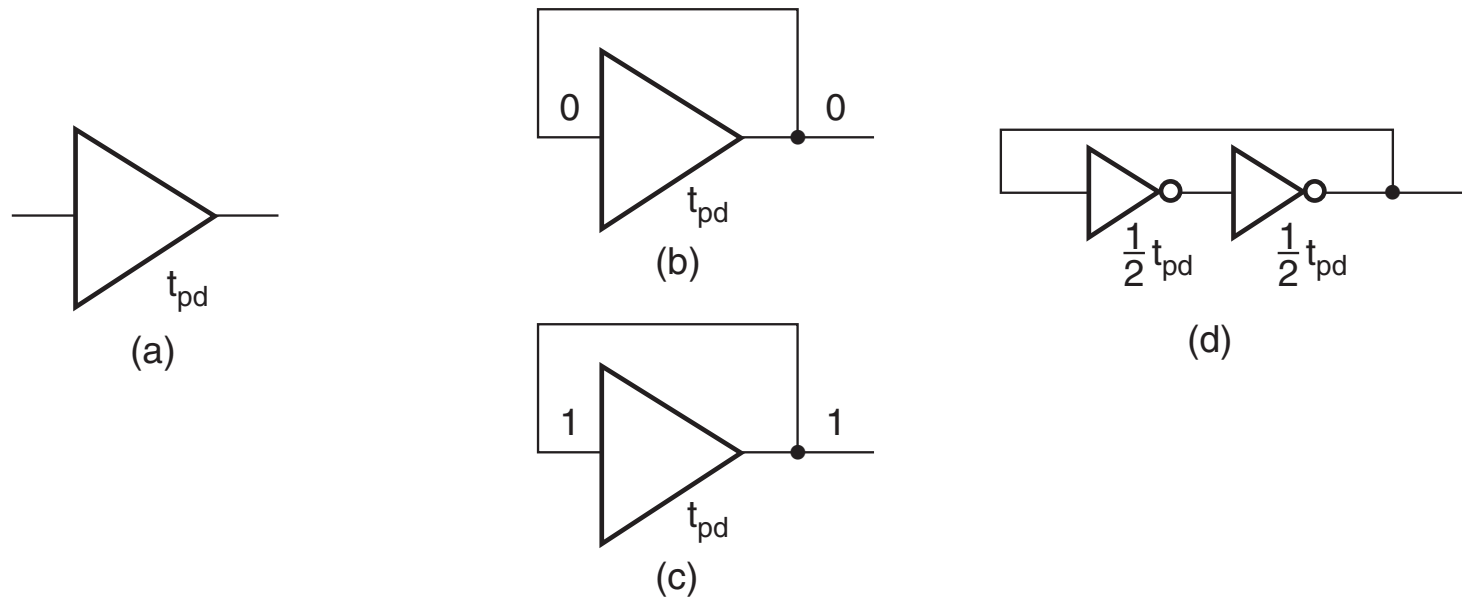
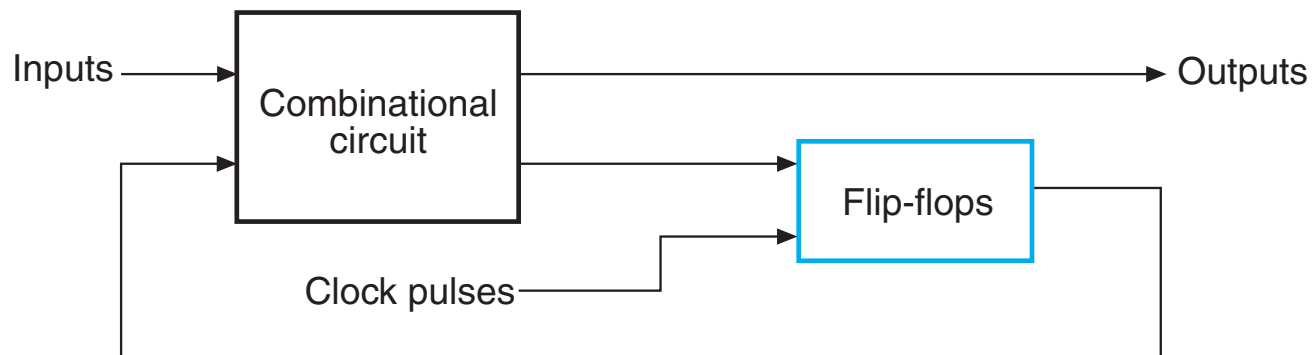


Fig. 4-2 Logic Structures for Storing Information

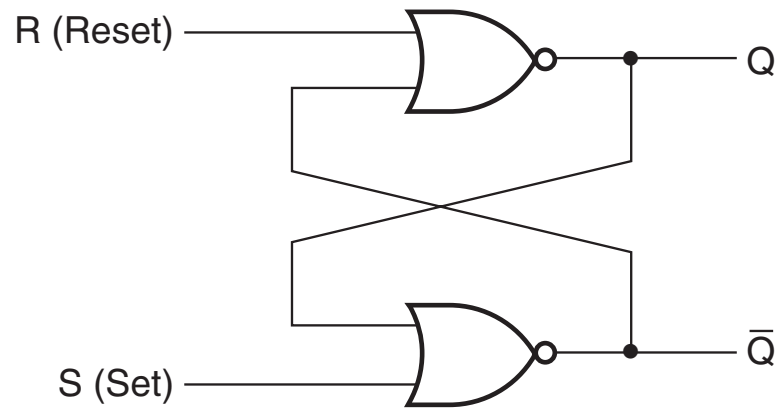


(a) Block diagram



(b) Timing diagram of clock pulses

Fig. 4-3 Synchronous Clocked Sequential Circuit



(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

Fig. 4-4 SR Latch with NOR Gates

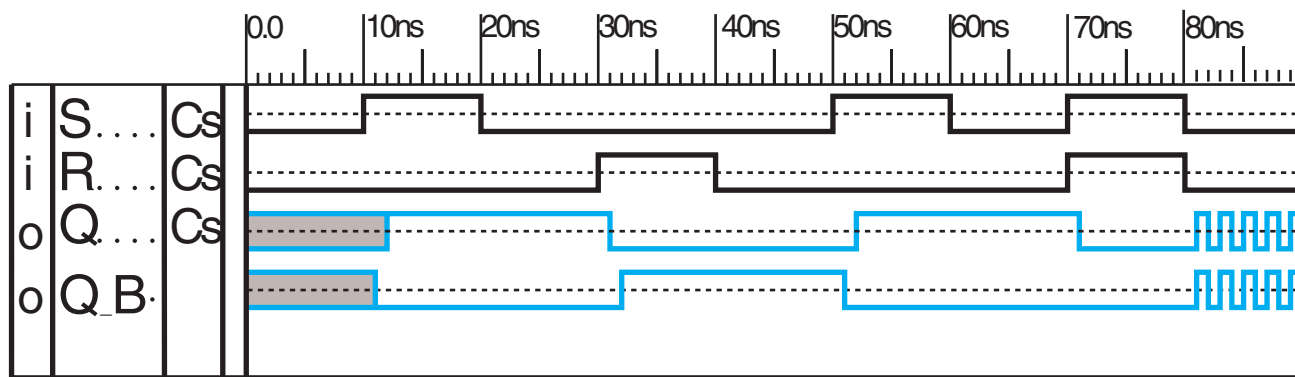
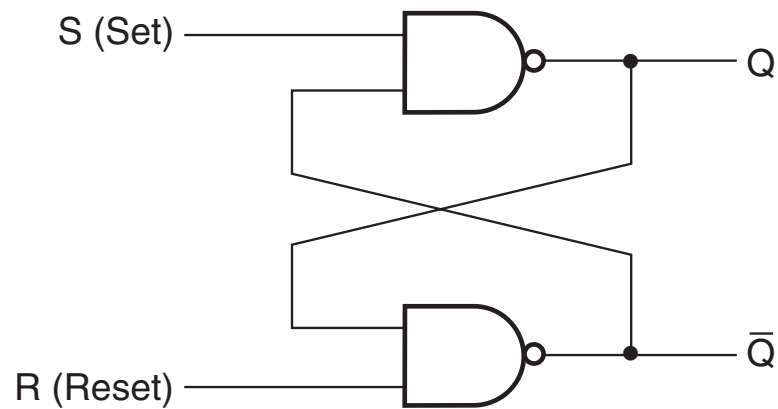


Fig. 4-5 Logic Simulation of *SR* Latch Behavior

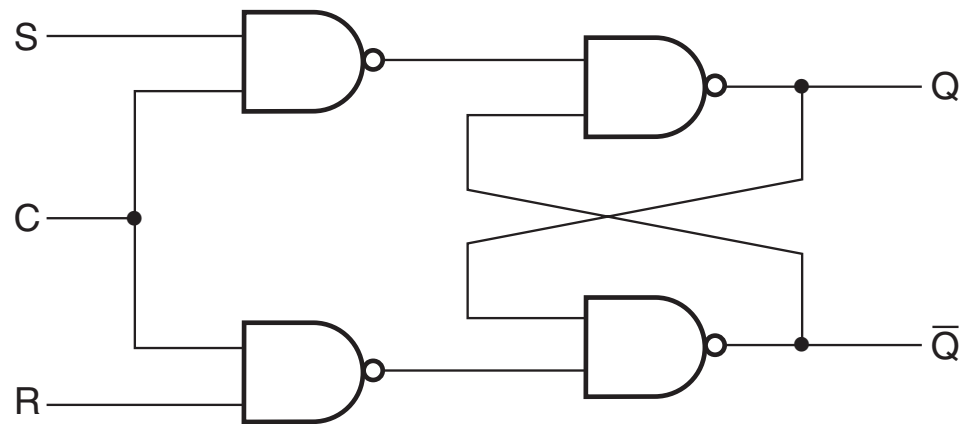


(a) Logic diagram

S	R	Q	\bar{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

Fig. 4-6 $\bar{S}\bar{R}$ Latch with NAND Gates

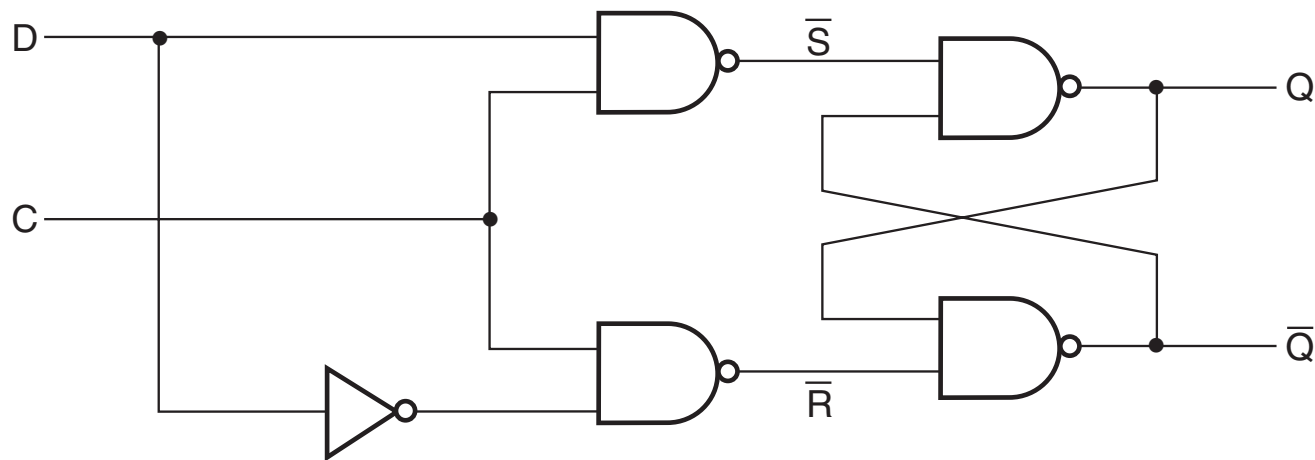


(a) Logic diagram

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

Fig. 4-7 SR Latch with Control Input



(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

Fig. 4-8 D Latch

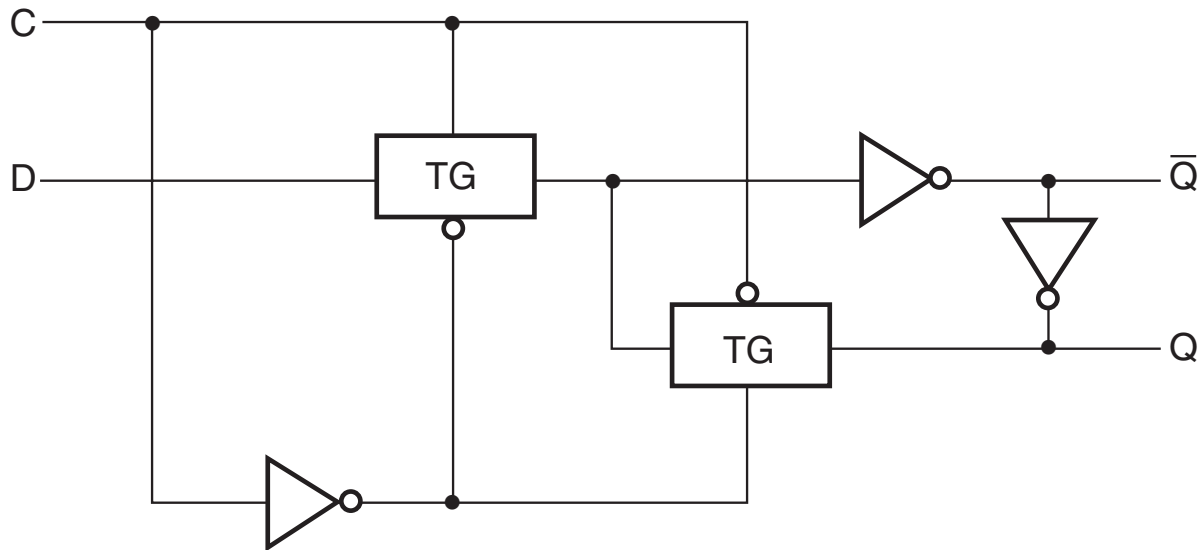


Fig. 4-9 *D* Latch with Transmission Gates

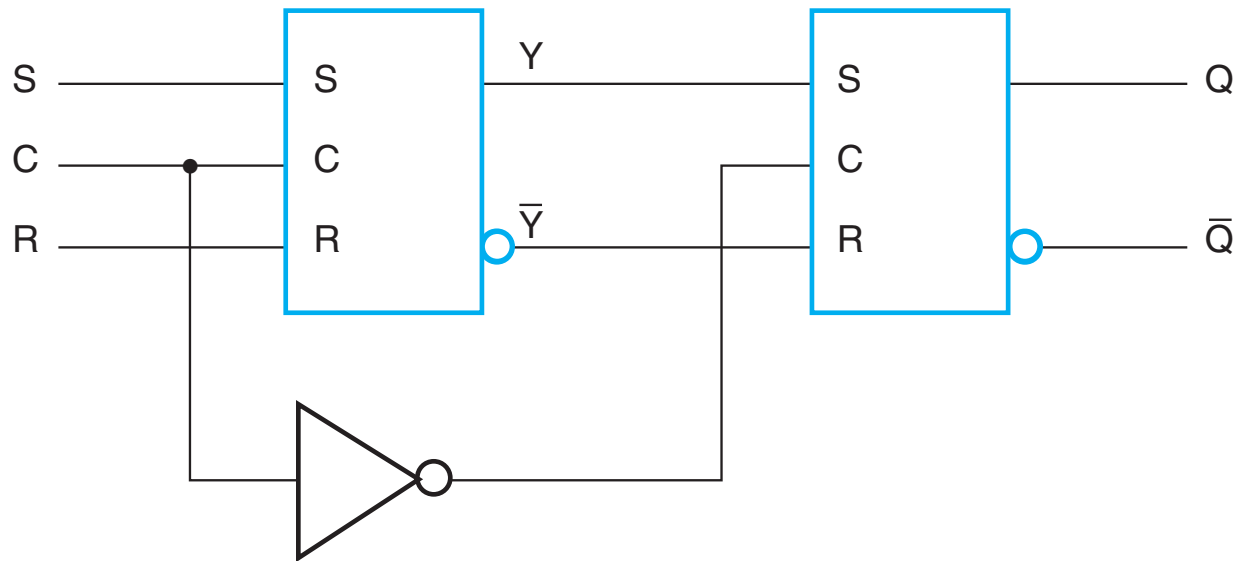


Fig. 4-10 SR Master-Slave Flip-Flop

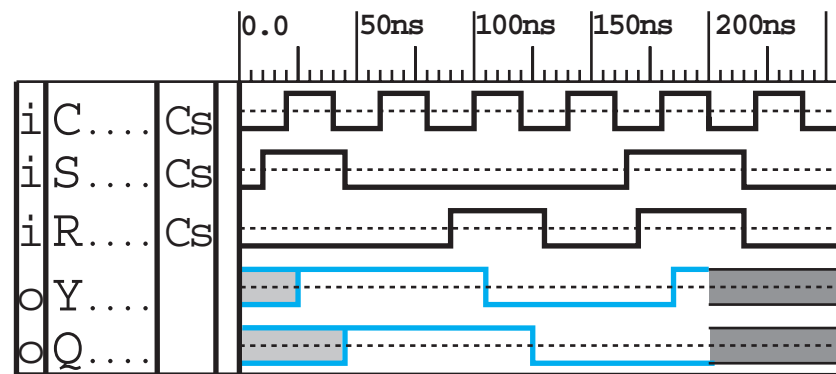
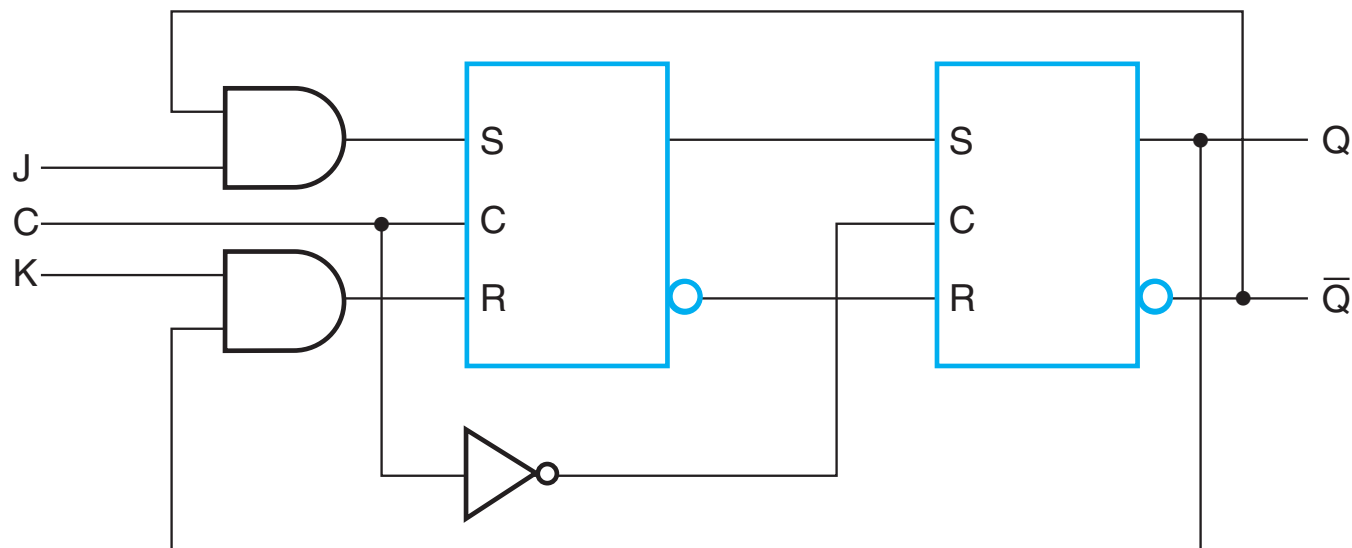


Fig. 4-11 Logic Simulation of a Master-Slave Flip-Flop



(a)

		Next State of Q
J	K	
0	0	Q
0	1	0
1	0	1
1	1	\overline{Q}

(b)

Fig. 4-12 Master-Slave *JK* Flip-Flop

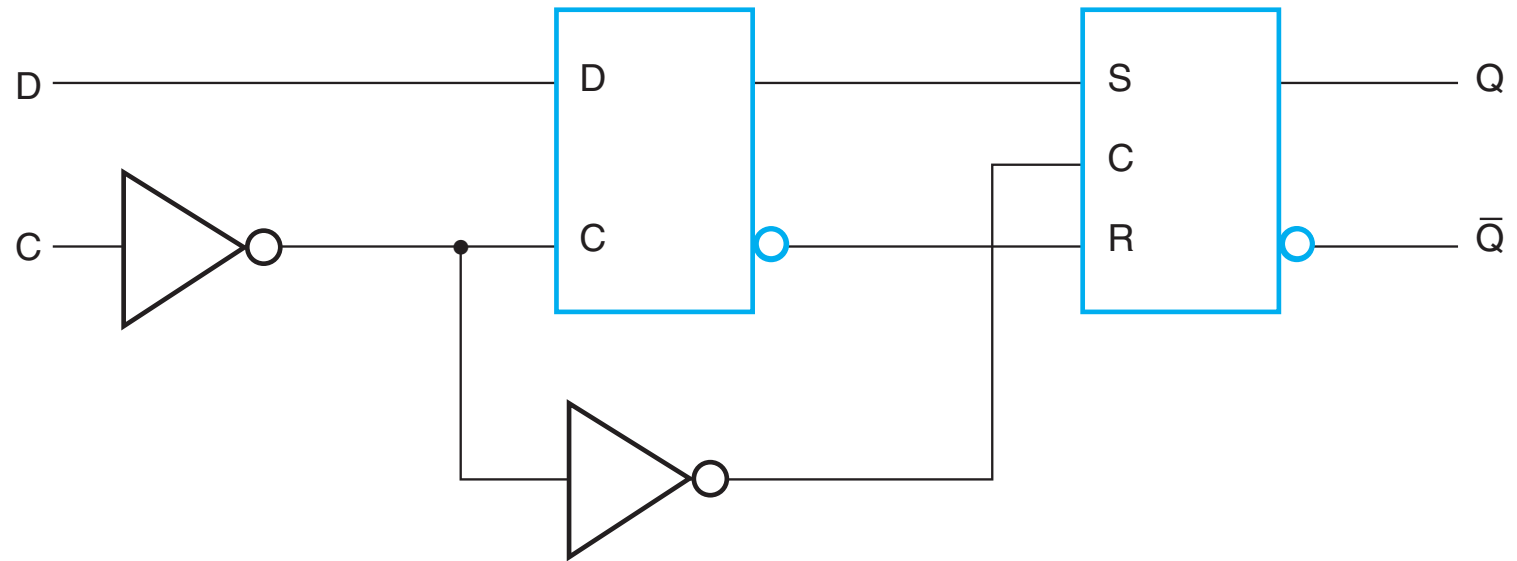


Fig. 4-13 *D*-Type Positive Edge-Triggered Flip-Flop

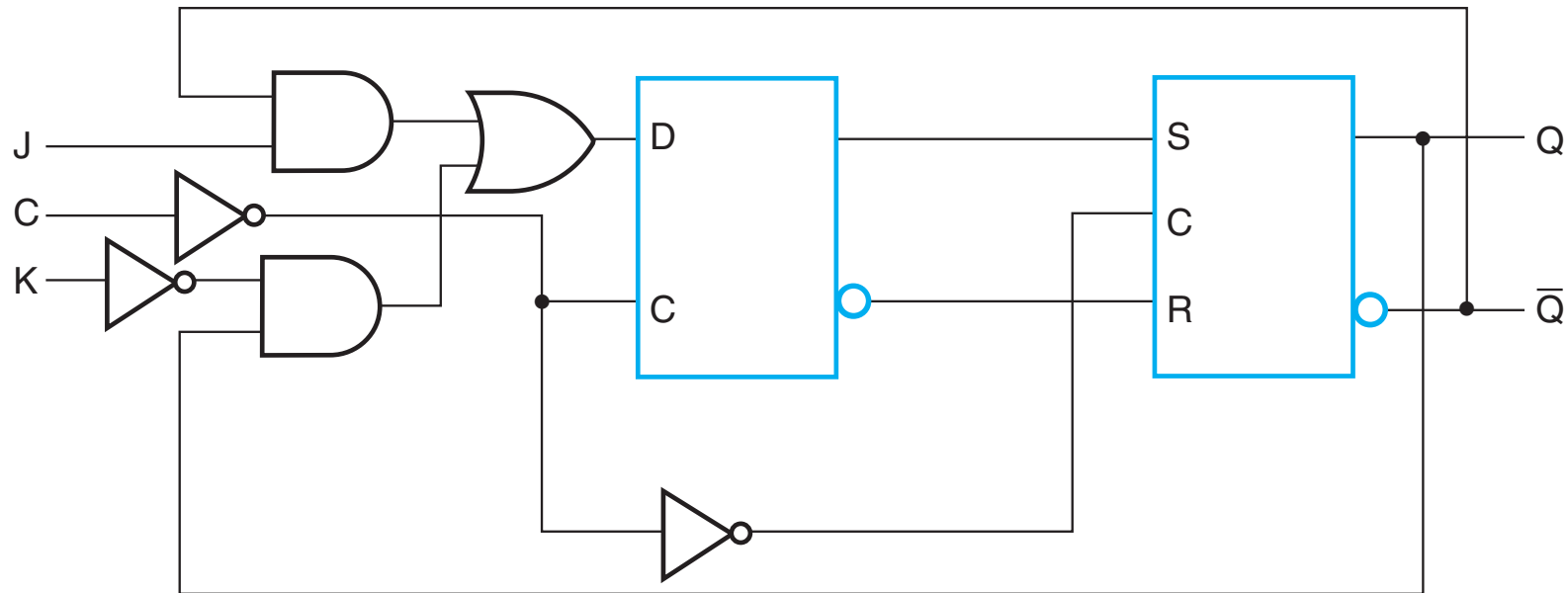
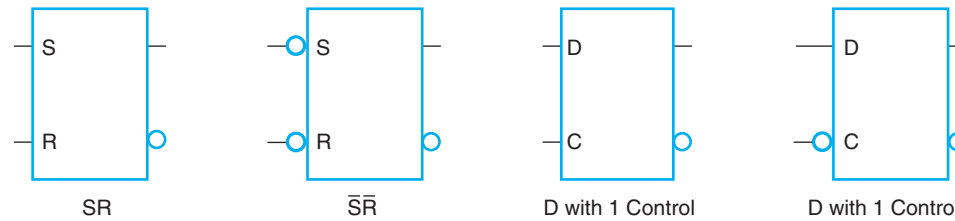
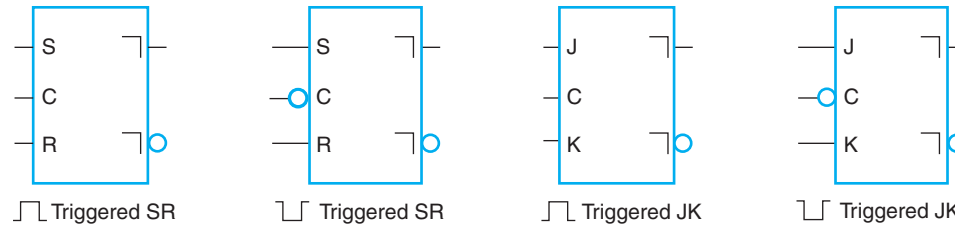


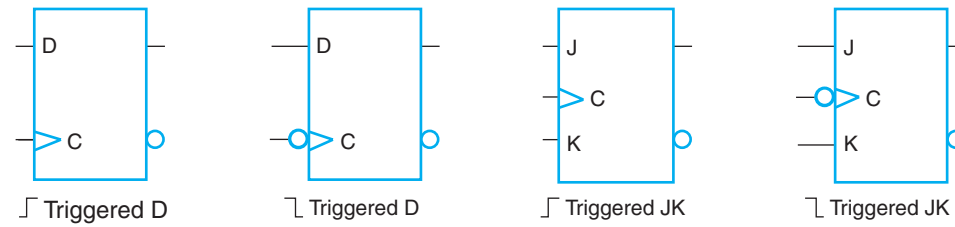
Fig. 4-14 Positive Edge-Triggered *JK* Flip-Flop



(a) Latches



(b) Master-Slave Flip-Flops



(c) Edge-Triggered Flip-Flops

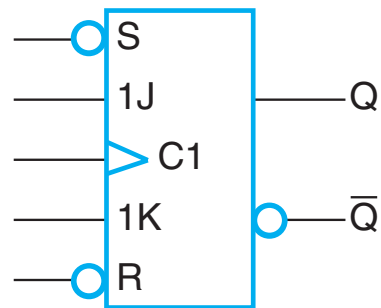
Fig. 4-15 Standard Graphic Symbols for Latch and Flip-Flops

TABLE 4-1
Flip-Flop Characteristic Tables

(a) <i>JK</i> Flip-Flop				(b) <i>SR</i> Flip-Flop			
J	K	$Q(t + 1)$	Operation	S	R	$Q(t + 1)$	Operation
0	0	$Q(t)$	No change	0	0	$Q(t)$	No change
0	1	0	Reset	0	1	0	Reset
1	0	1	Set	1	0	1	Set
1	1	$\overline{Q}(t)$	Complement	1	1	?	Undefined

(c) <i>D</i> Flip-Flop			(d) <i>T</i> Flip-Flop		
D	$Q(t + 1)$	Operation	T	$Q(t + 1)$	Operation
0	0	Reset	0	$Q(t)$	No change
1	1	Set	1	$\overline{Q}(t)$	Complement

Table 4-1 Flip-Flop Characteristic Tables



(a) Graphic symbols

S	R	C	J	K	Q	\bar{Q}
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	Undefined	
1	1	↑	0	0	No change	
1	1	↑	0	1	0	1
1	1	↑	1	0	1	0
1	1	↑	1	1	Complement	

(b) Function table

Fig. 4-16 *JK* Flip-Flop with Direct Set and Reset

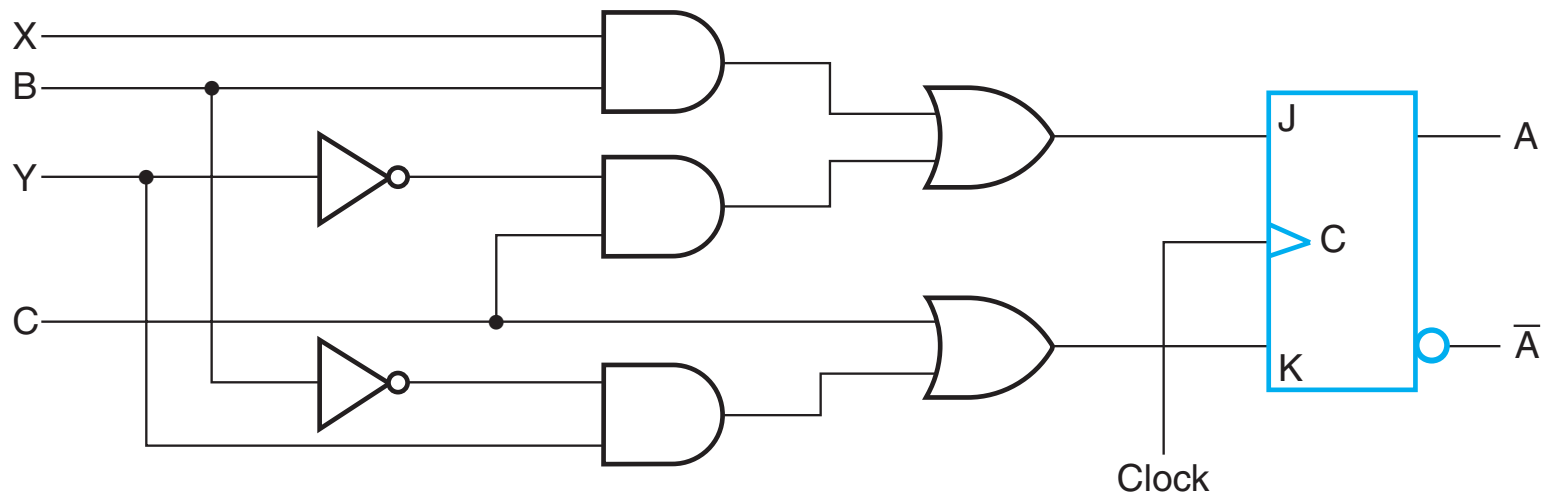


Fig. 4-17 Implementing Input Equations

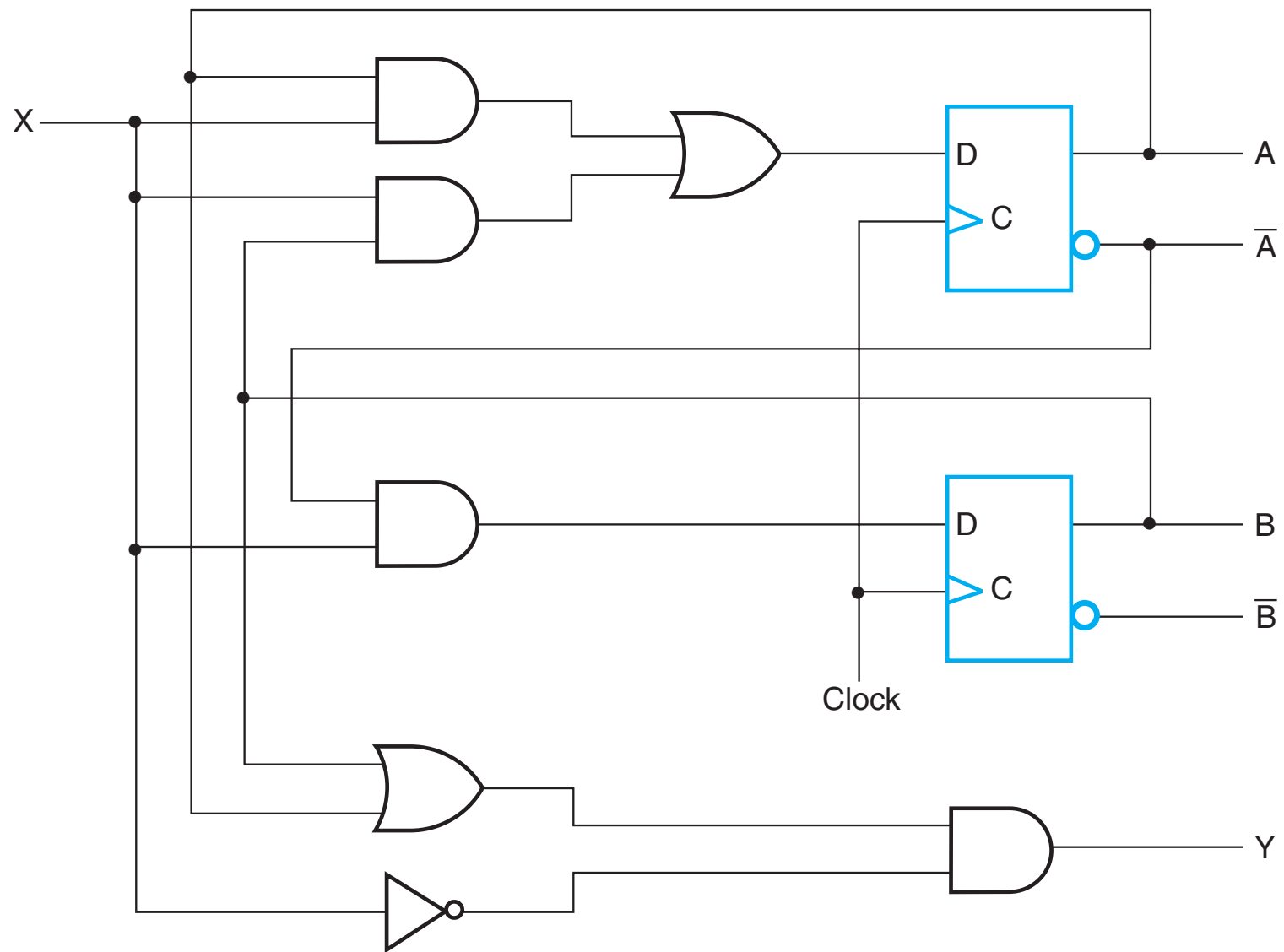


Fig. 4-18 Example of a Sequential Circuit

TABLE 4-2
State Table for Circuit of Figure 4-18

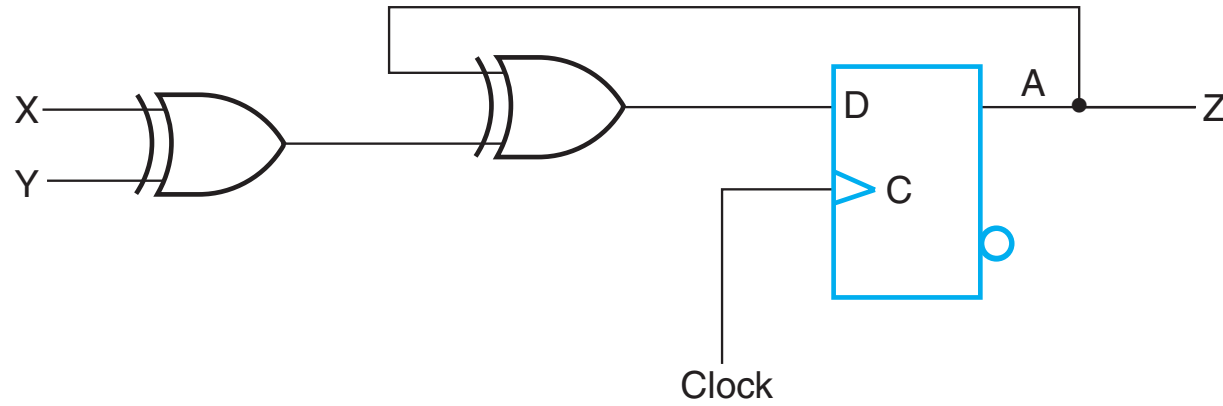
Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Table 4-2 State Table for Circuit of Figure 4-18

TABLE 4-3
Two-Dimensional State Table for the Circuit in Figure 4-18

Present state		Next state				Output	
		X = 0		X = 1		X = 0	X = 1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

Table 4-3 Two-Dimensional State Table for the Circuit in Figure 4-18



(a)

Present state	Inputs		Next state	Output
A	X	Y	A	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(b) State table

Fig. 4-19 Logic Diagram and State Table for $D_A = A \oplus X \oplus Y$

TABLE 4-4
State Table for Circuit with *JK* Flip-Flops

Present state		Input	Next state		Flip-flop inputs			
A	B	X	A	B	J _A	K _A	J _B	K _B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

Table 4-4 State Table for Circuit with *JK* Flip-Flops

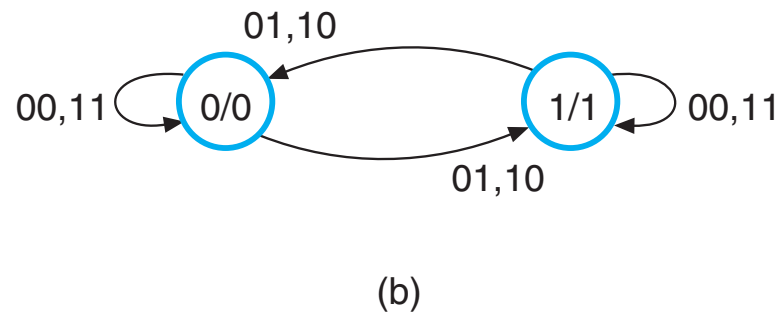
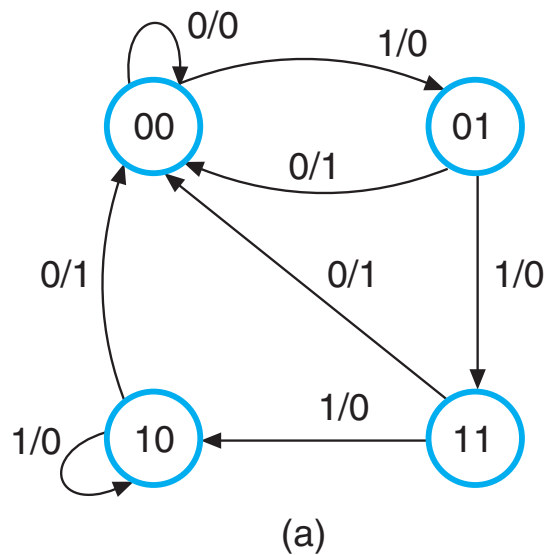


Fig. 4-20 State Diagrams

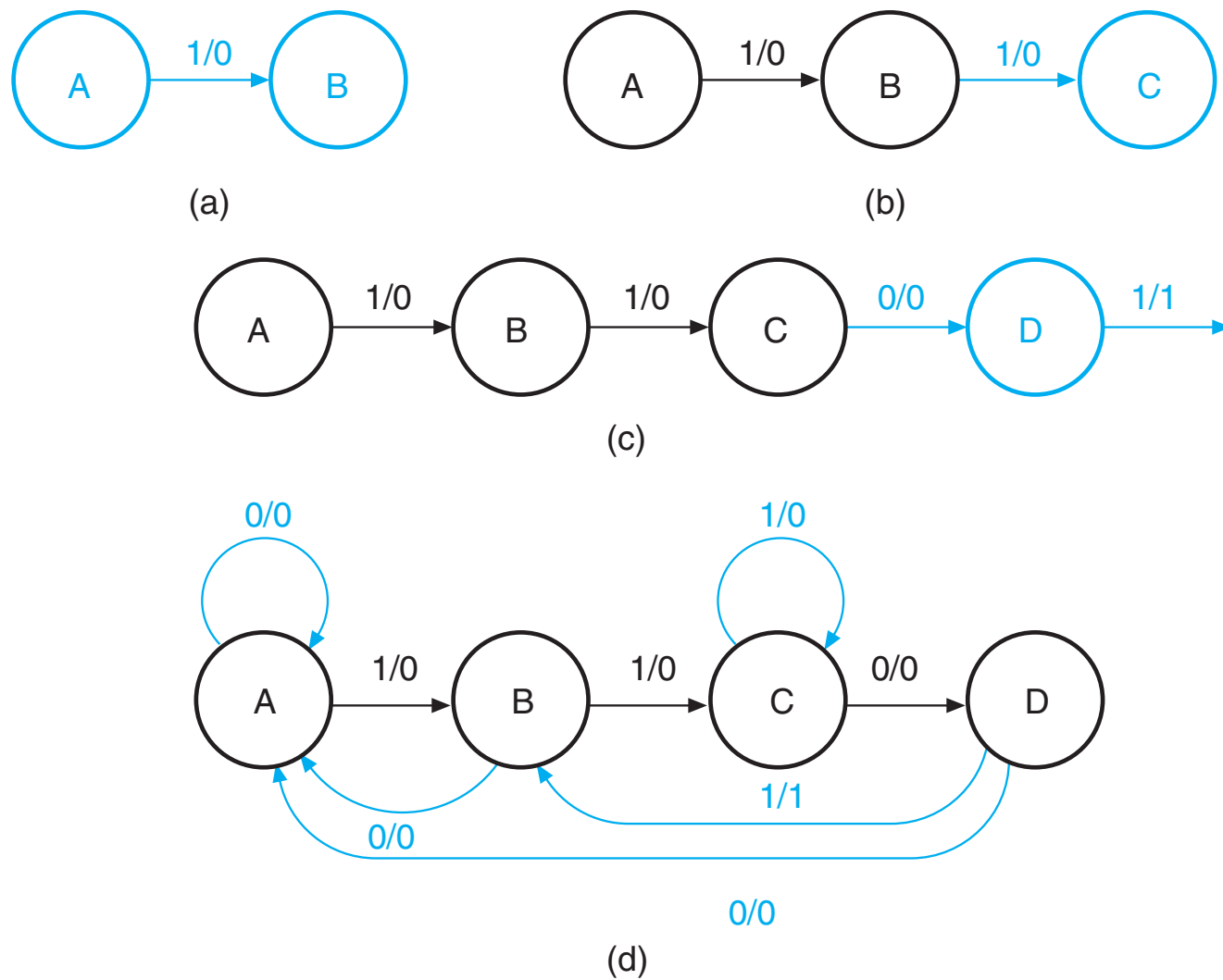


Fig. 4-21 Construction of a State Diagram

TABLE 4-5
State Table for State Diagram in Figure 4-21

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

Table 4-5 State Table for State Diagram in Figure 4-21

TABLE 4-6
Sequence Tables for Code Converter Example

Sequences in Order of Digits Represented								Sequences in Order of Common Prefixes							
BCD Input				Excess-3 Output				BCD Input				Excess-3 Output			
1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0
1	0	0	0	0	0	1	0	0	0	0	1	1	1	0	1
0	1	0	0	1	0	1	0	0	0	1	0	1	1	1	0
1	1	0	0	0	1	1	0	0	1	0	0	1	0	1	0
0	0	1	0	1	1	1	0	0	1	1	0	1	0	0	1
1	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0
0	1	1	0	1	0	0	1	1	0	0	1	0	0	1	1
1	1	1	0	0	1	0	1	1	0	1	0	0	0	0	1
0	0	0	1	1	1	0	1	1	1	0	0	0	1	1	0
1	0	0	1	0	0	1	1	1	1	1	0	0	1	0	1

Table 4-6 Sequence Tables for Code Converter Example

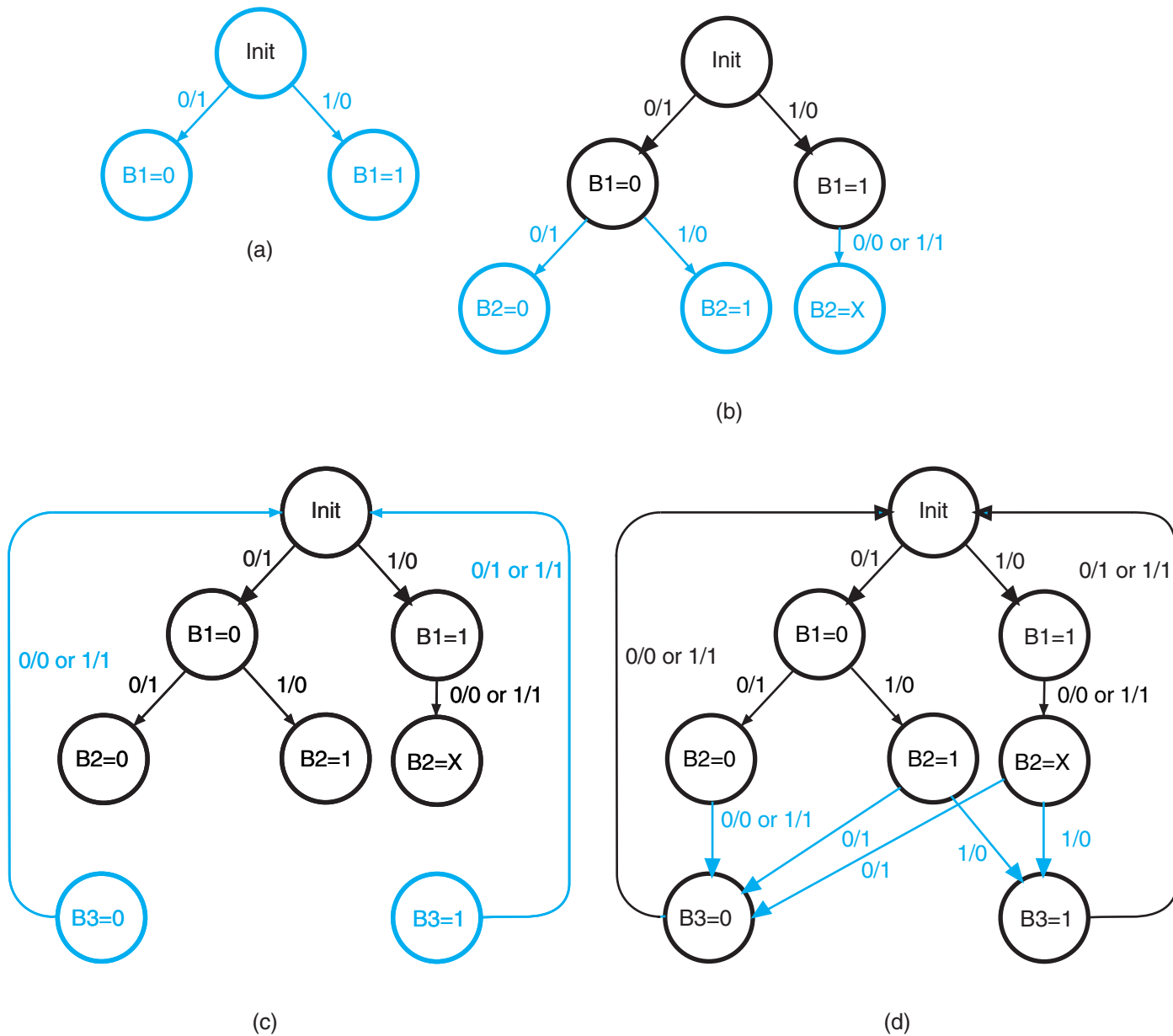


Fig. 4-22 Construction of a State Diagram

□ **TABLE 4-7**

Table 4-5 with Names Replaced by Binary Codes

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

Table 4-7 Table 4-5 with Names Replaced by Binary Codes

TABLE 4-8
State Table for Design Example

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

Table 4-8 State Table for Design Example

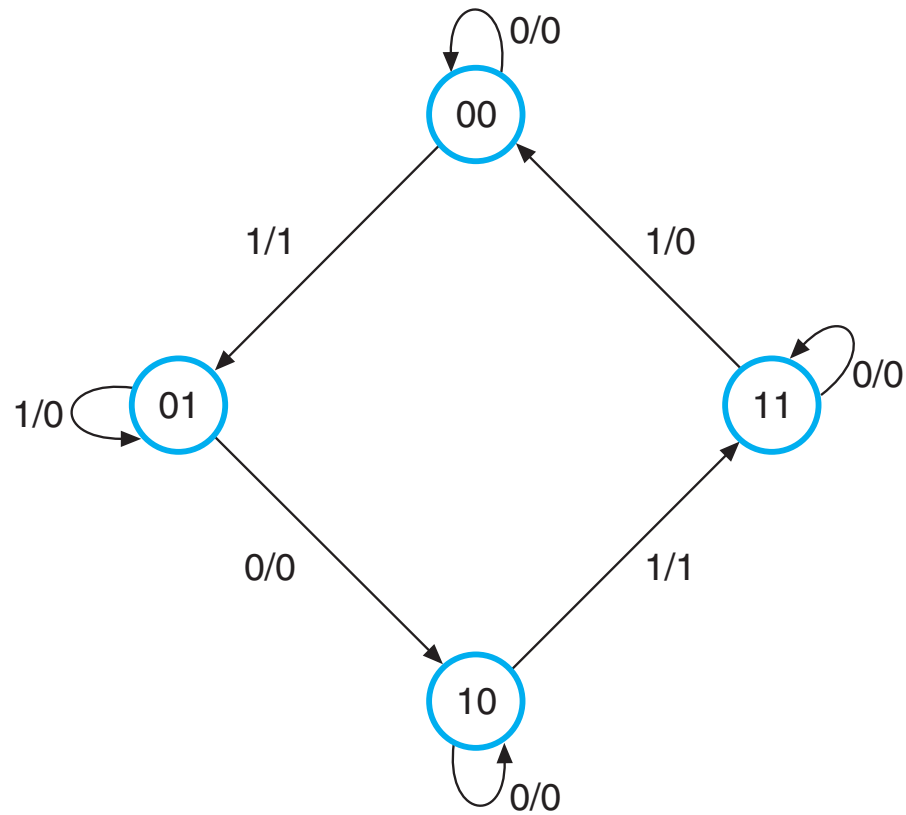


Fig. 4-23 State Diagram for Design Example

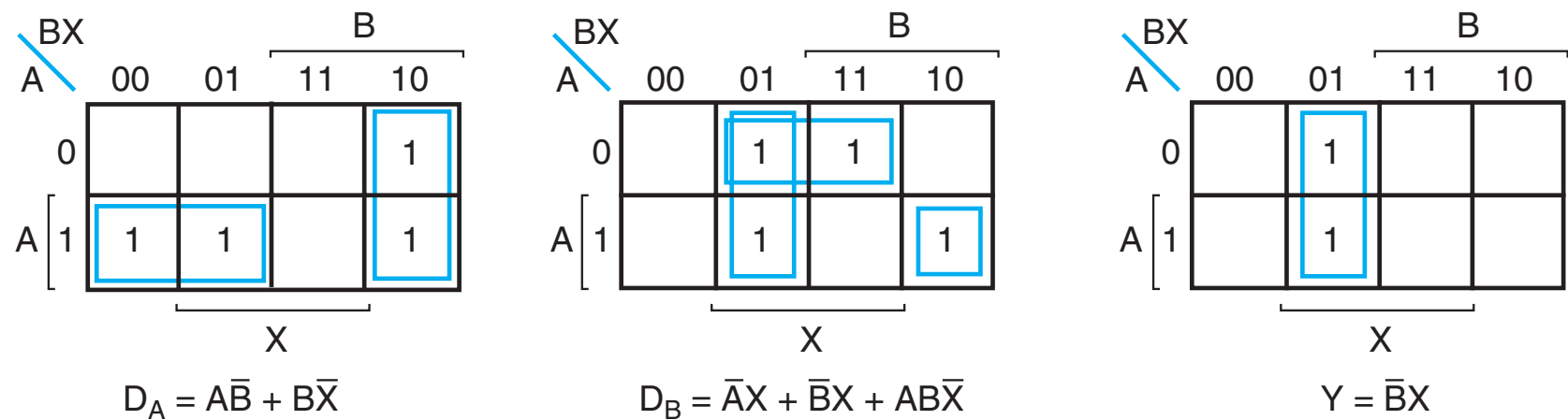


Fig. 4-24 Maps for Input Equations and Output Y

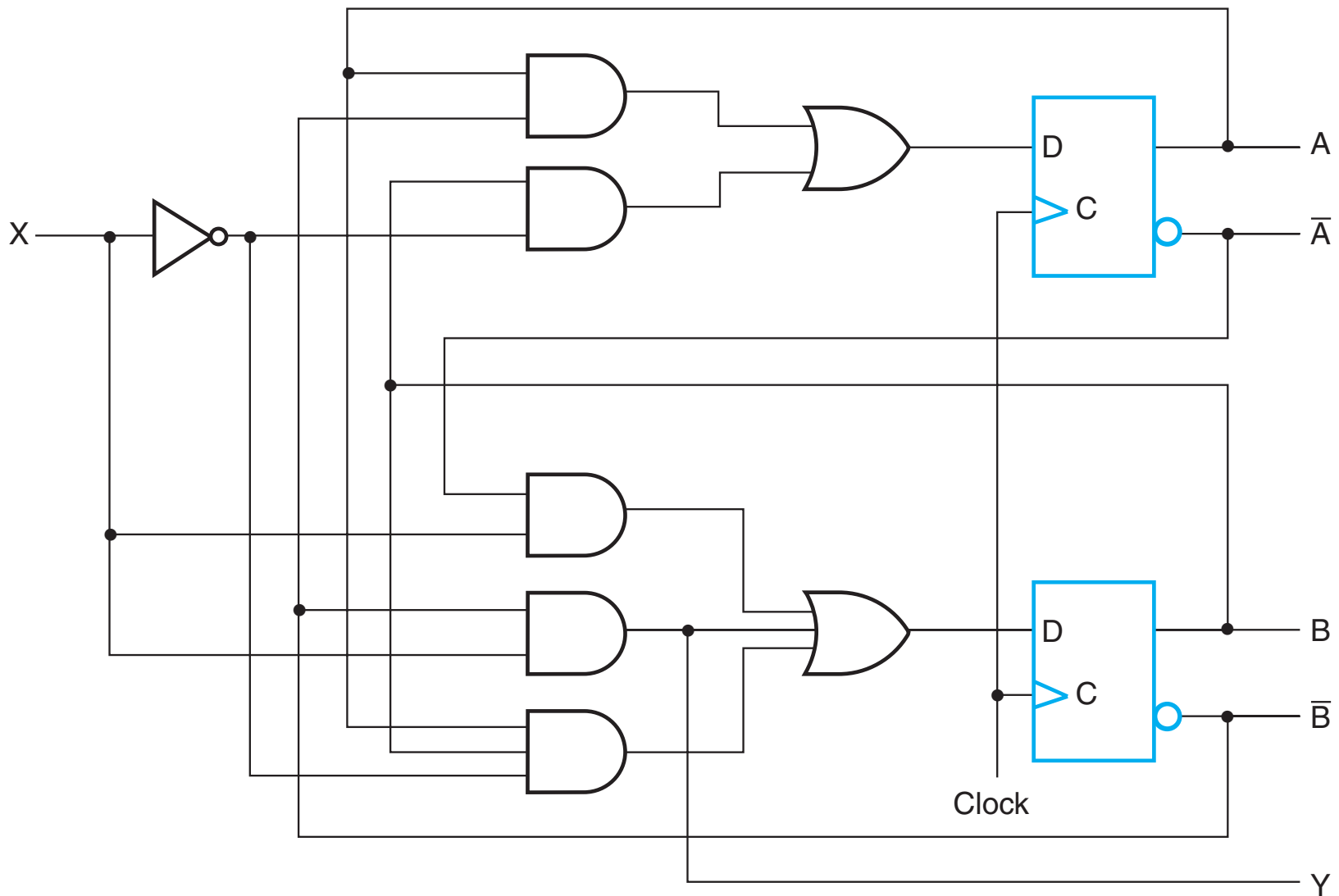


Fig. 4-25 Logic Diagram for Sequential Circuit with D Flip-Flops

TABLE 4-9
State Table for Second Design Example

Present State			Input	Next State		
A	B	C	X	A	B	C
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	1	0	0
1	0	0	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	1	0	0

Table 4-9 State Table for Second Design Example

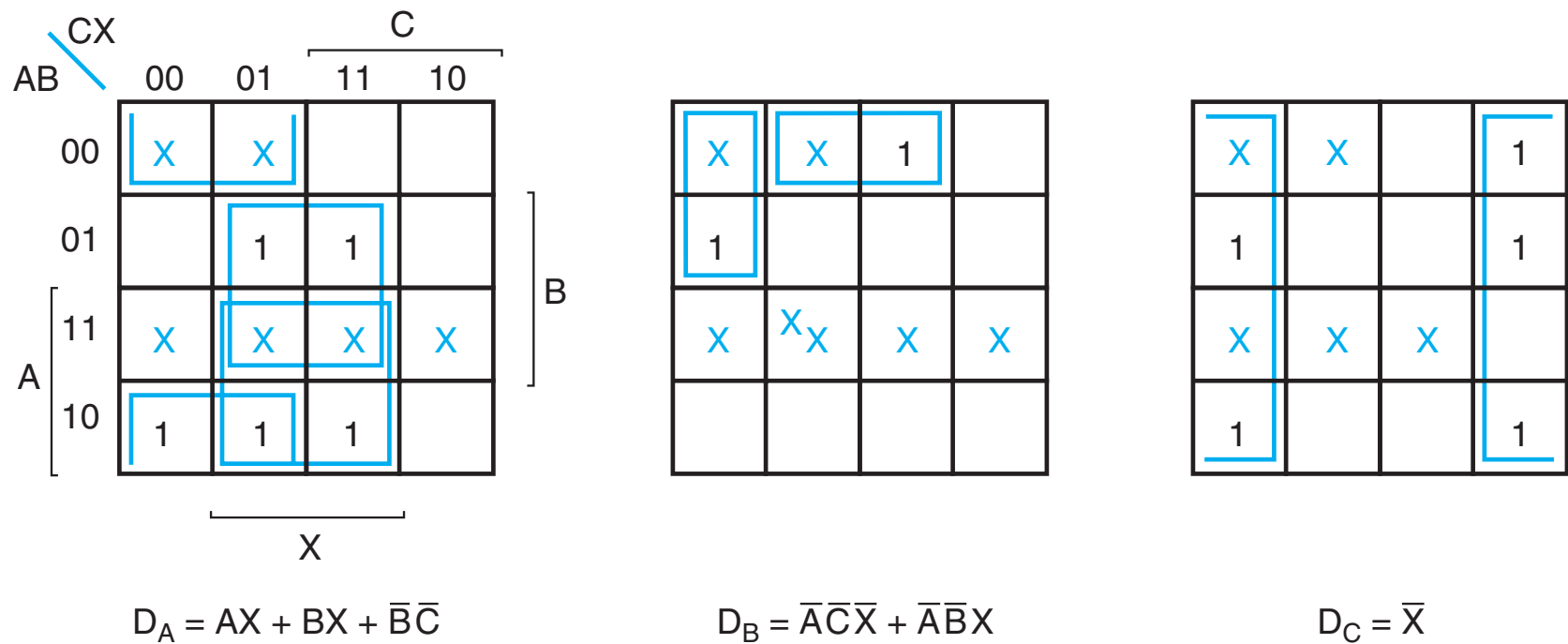


Fig. 4-26 Maps for Simplifying Input Equations

TABLE 4-10
Flip-Flop Excitation Tables

(a) JK Flip-Flop				(b) SR Flip-Flop			
$Q(t)$	$Q(t + 1)$	J	K	$Q(t)$	$Q(t + 1)$	S	R
0	0	0	X	0	0	0	X
0	1	1	X	0	1	1	0
1	0	X	1	1	0	0	1
1	1	X	0	1	1	X	0

(c) D Flip-Flop			(d) T Flip-Flop		
$Q(t)$	$Q(t + 1)$	D	$Q(t)$	$Q(t + 1)$	T
0	0	0	0	0	0
0	1	1	0	1	1
1	0	0	1	0	1
1	1	1	1	1	0

Table 4-10 Flip-Flop Excitation Tables

TABLE 4-11
State Table with *JK* Flip-Flop Inputs

Present State		Input	Next State		Flip-Flop Inputs			
A	B	X	A	B	J _A	K _A	J _B	K _B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Table 4-11 State Table with *JK* Flip-Flop Inputs

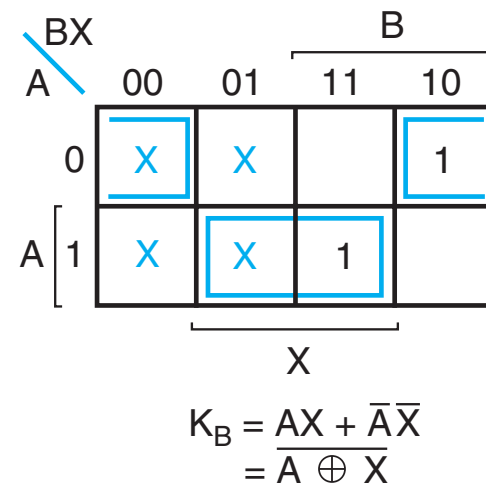
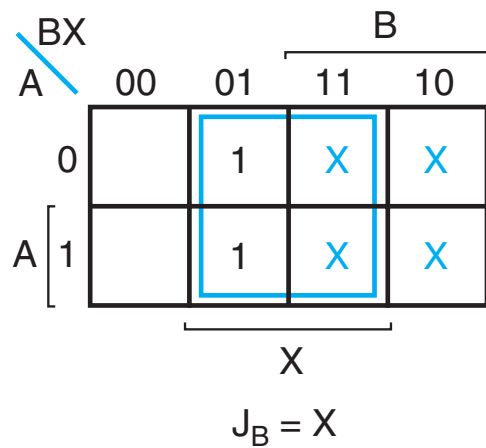
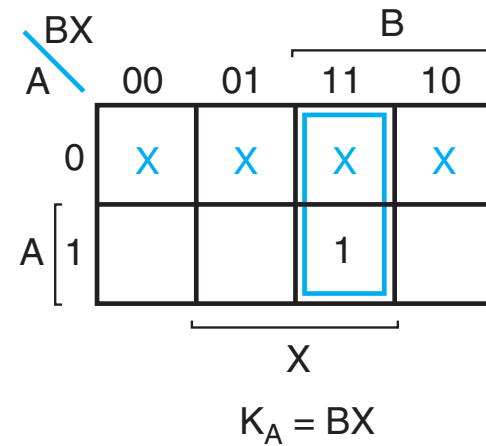
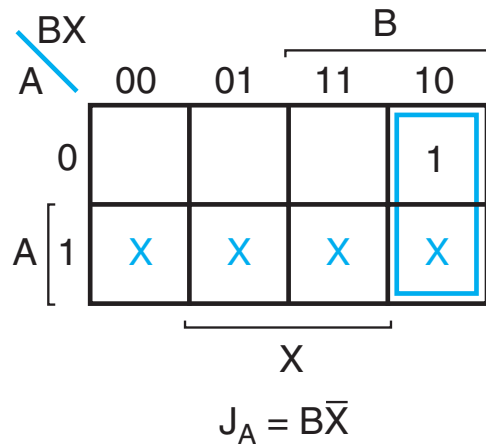


Fig. 4-27 Maps for J and K Input Equations

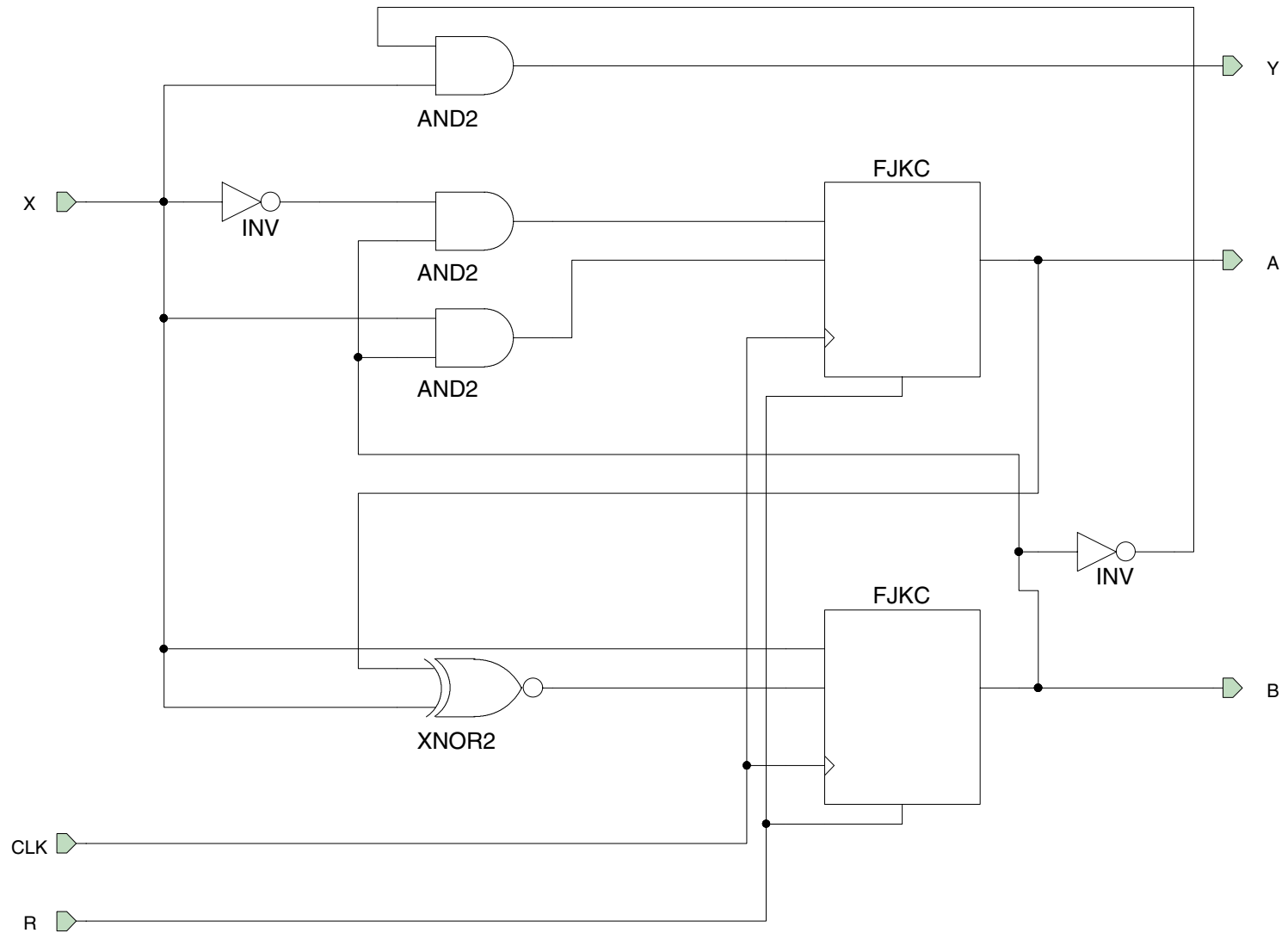
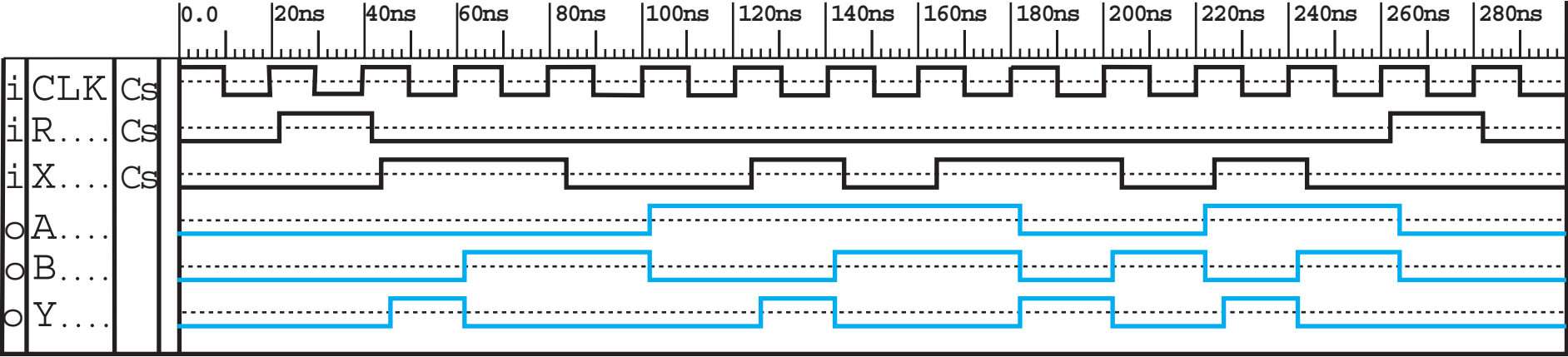


Fig. 4-28 Logic Diagram for Sequential Circuit with *JK* Flip-Flops

R:	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
X:	0	0	0	1	1	0	0	1	0	1	1	0	1	0	0
A:	X	X	0*	0	0	0	1	1	1	1	0	0	1	1	0*
B:	X	X	0*	0	1	1	0	0	1	1	0	1	0	1	0*
Y:	0	0	0*	1	0	0	0	1	0	0	1	0	1	0	0*

* These responses are asynchronous with the clock and thus do not wait for the next positive clock edge.

(a) Circuit test and expected results



(b) Simulation results

Fig. 4-29 Logic Simulation Verification for the Circuit in Figure 4-28


```

-- Positive Edge-Triggered D Flip-Flop with Reset:
-- VHDL Process Description
library ieee;
use ieee.std_logic_1164.all;
entity dff is
    port(CLK, RESET, D : in std_logic;
        Q : out std_logic);
end dff;

architecture pet_pr of dff is
    -- Implements positive edge-triggered bit state storage
    -- with asynchronous reset.

    begin
    process (CLK, RESET)
    begin
        if (RESET = '1') then
            Q <= '0';
        elsif (CLK'event and CLK = '1') then
            Q <= D;
        end if;
    end if;
    end process;
end;

```

Fig. 4-30 VHDL Process Description of Positive Edge-Triggered Flip-Flop with Reset

```

-- Sequence Recognizer: VHDL Process Description
-- (See Figure 4-21 for state diagram)
library ieee;
use ieee.std_logic_1164.all;
entity seq_rec is
    port(CLK, RESET, X: in std_logic;
         Z: out std_logic);
end seq_rec;

architecture process_3 of seq_rec is
    type state_type is (A, B, C, D);
    signal state, next_state : state_type;
begin

    -- Process 1 - state_register: implements positive edge-triggered
    -- state storage with asynchronous reset.
    state_register: process (CLK, RESET)
    begin
        if (RESET = '1') then
            state <= A;
        elsif (CLK'event and CLK = '1') then
            state <= next_state;
        end if;
    end if;
end process;

    -- Process 2 - next_state_function: implements next state as
    -- a function of input X and state.
    next_state_func: process (X, state)
    begin
        case state is
            when A =>
                if X = '1' then
                    next_state <= B;
                else
                    next_state <= A;
                end if;
            when B =>
                if X = '1' then
                    next_state <= C;
                else
                    next_state <= A;
                end if;
        end case;
    end process;
end architecture;

```

Fig. 4-31 VHDL Process Description of a Sequence Recognizer

```

-- Sequence Recognizer: VHDL Process Description (continued)
when C =>
    if X = '1' then
        next_state <= C;
    else
        next_state <= D;
    end if;
    when D =>
        if X = '1' then
            next_state <= B;
        else
            next_state <= A;
        end if;
    end case;
end process;

-- Process 3 - output_function: implements output as function
-- of input X and state.
output_func: process (X, state)
begin
    case state is
        when A =>
            Z <= '0';
        when B =>
            Z <= '0';
        when C =>
            Z <= '0';
            when D =>
                if X = '1' then
                    Z <= '1';
                else
                    Z <= '0';
                end if;
            end case;
        end process;
end;

```

Fig. 4-32 VHDL Process Description of a Sequence Recognizer (continued)

TABLE 4-12
Illustration of generation of storage in VHDL

Inputs			Action
RESET = 1	CLK = 1	CLK'event	
FALSE	FALSE	FALSE	Unspecified
FALSE	FALSE	TRUE	Unspecified
FALSE	TRUE	FALSE	Unspecified
FALSE	TRUE	TRUE	$Q \leq D$
TRUE	—	—	$Q \leq '0'$

Table 4-12 Illustration of generation of storage in VHDL

```
// Positive Edge-Triggered D Flip-Flop with Reset:  
// Verilog Process Description  
  
module dff_v(CLK, RESET, D, Q);  
    input CLK, RESET, D;  
    output Q;  
    reg Q;  
  
    always @(posedge CLK or posedge RESET)  
    begin  
        if (RESET)  
            Q <= 0;  
        else  
            Q <= D;  
        end  
    endmodule
```

Fig. 4-33 Verilog Process Description of Positive Edge-Triggered Flip-Flop with Reset

```

// Sequence Recognizer: Verilog Process Description
// (See Figure 4-21 for state diagram)
module seq_rec_v(CLK, RESET, X, Z);
    input CLK, RESET, X;
    output Z;
    reg [1:0] state, next_state;
    parameter A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;
    reg Z;
    // state register: implements positive edge-triggered
    // state storage with asynchronous reset.
    always @(posedge CLK or posedge RESET)
    begin
        if (RESET == 1)
            state <= A;
        else
            state <= next_state;
    end
    // next state function: implements next state as function
    // of X and state
    always @(X or state)
    begin
        case (state)
            A:  if (X == 1)
                    next_state <= B;
                else
                    next_state <= A;
            B:  if(X) next_state <= C; else next_state <= A;
            C:  if(X) next_state <= C; else next_state <= D;
            D:  if(X) next_state <= B; else next_state <= A;
        endcase
    end
    // output function: implements output as function
    // of X and state
    always @(X or state)
    begin
        case (state)
            A: Z <= 0;
            B: Z <= 0;
            C: Z <= 0;
            D: Z <= X ? 1 : 0;
        endcase
    end
end
endmodule

```

Fig. 4-34 Verilog Process of a Sequence Recognizer

TABLE 4-13
Illustration of generation of storage in Verilog

Inputs		Action
posedge RESET and RESET = 1	posedge CLK	
FALSE	FALSE	Unspecified
FALSE	TRUE	$Q \leq D$
TRUE	FALSE	$Q \leq 0$
TRUE	TRUE	$Q \leq 0$

Table 4-13 Illustration of generation of storage in Verilog