

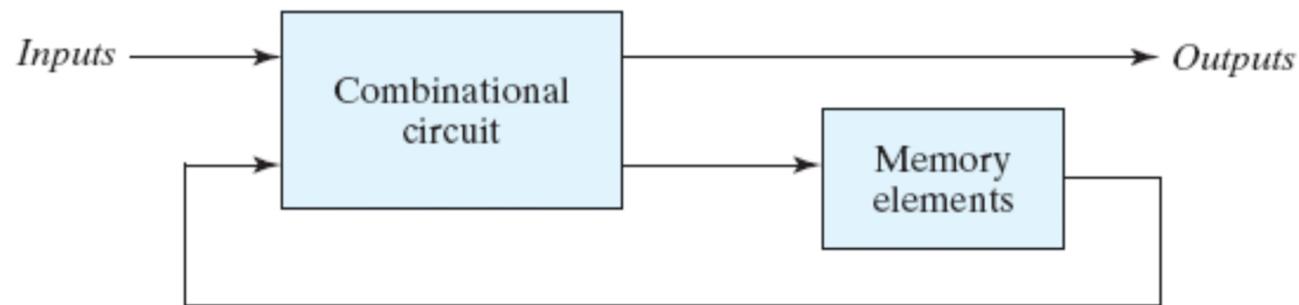
## Ch 5. Synchronous sequential logic

## 5.1 Introduction

- In order to perform useful or flexible sequences of operations, we need to be able to construct circuits that can store information between the operations.
- latches and Flip-Flops
- Sequential circuits consisting of both flip-flop and combinational logic

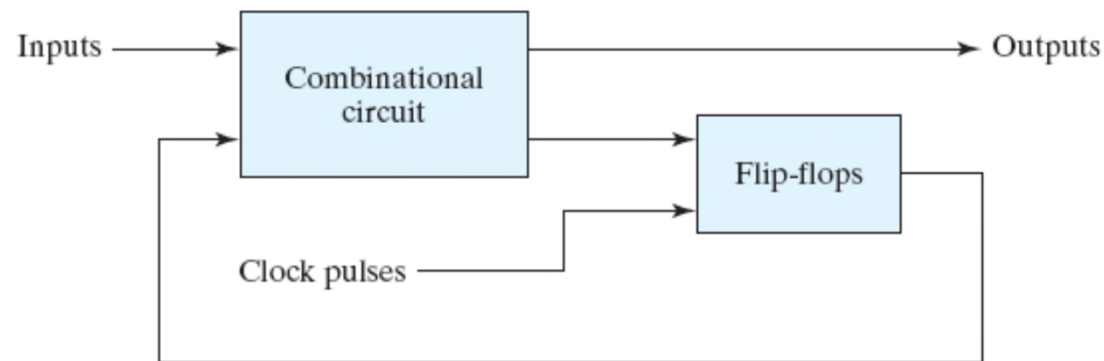
## 5.2 Sequential circuits

- Outputs are function of inputs and present states
- Present states are supplied by memory elements



## 5.2 Sequential circuits

- Two types of sequential circuit
- Synchronous : behavior depends on the signals affecting storage elements at discrete time
- Asynchronous : behavior depends on inputs at any instance of time



(a) Block diagram



(b) Timing diagram of clock pulses

## 5.3 Latches

- SR latch : consist of two cross-coupled NOR gates
- $S=1, R=0$  then  $Q=1$  (set)
- $S=0, R=1$  then  $Q=0$  (reset)
- $S=0, R=0$  then no change (keep condition)
- $S=1, R=1$   $Q=Q'=0$  (undefined)

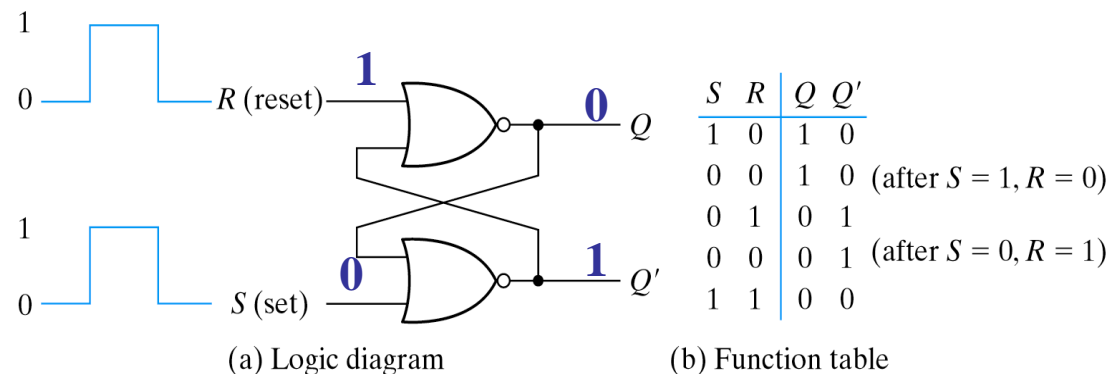


Fig. 5-3 SR Latch with NOR Gates

## 5.3 Latches - SR latch

- S'R' latch with NAND gates
  - Require the complement value of NOR latch

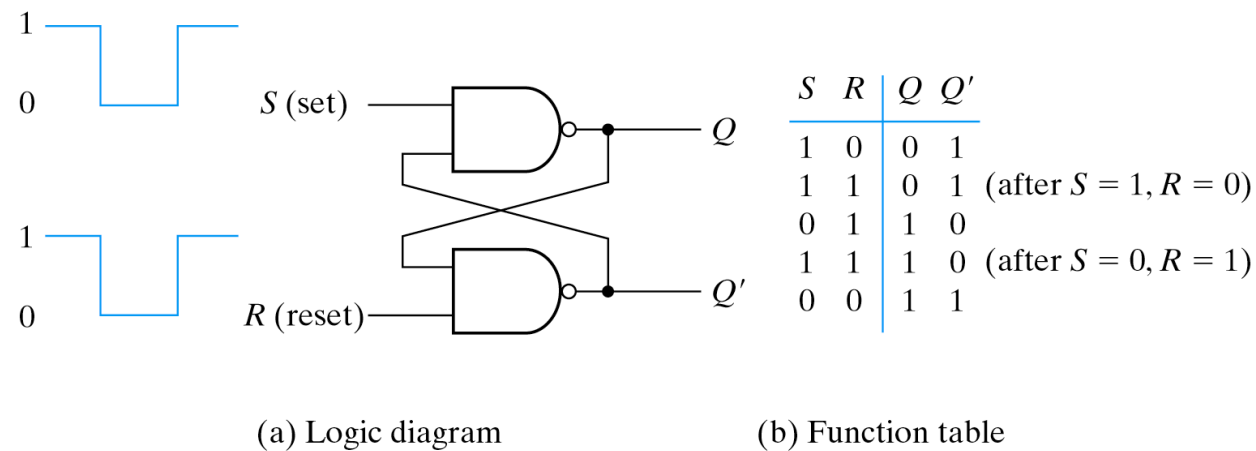
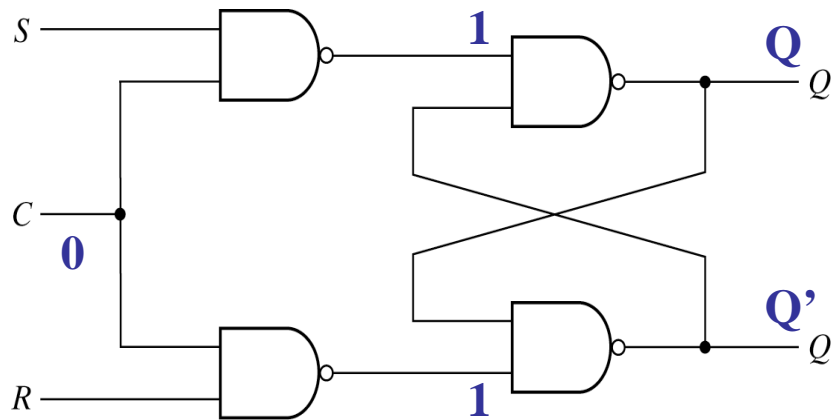


Fig. 5-4 SR Latch with NAND Gates

## 5.3 Latches - SR latch

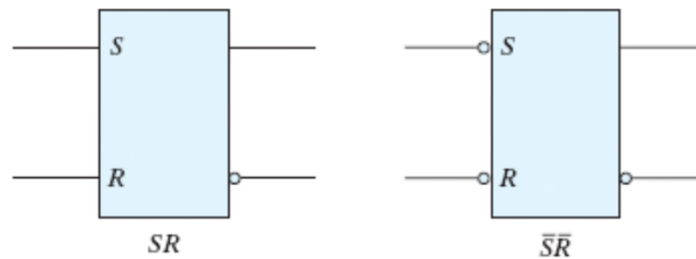
- SR latch with control input
  - Add two NAND gate and control signal
  - $C=0$ (no action),  $C=1$ (act as SR latch)



(a) Logic diagram

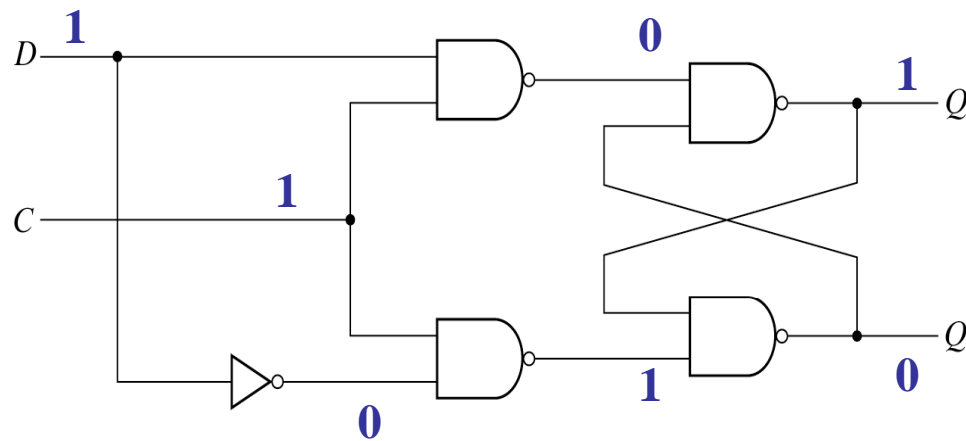
$C$	$S$	$R$	Next state of $Q$
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$ ; Reset state
1	1	0	$Q = 1$ ; set state
1	1	1	Indeterminate

(b) Function table



## 5.3 Latches - D latch

- Eliminate indeterminate state in SR latch
  - $C=1$ , output value is equal to  $D$

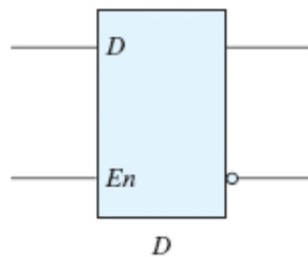


(a) Logic diagram

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

(b) Function table

Fig. 5-6 D Latch





## 5.4 Flip-Flops

- Latch : case (a), output changes as input changes
- Flip-flop : output only changes at clock edge

Latch

1



(a) Response to positive level



(b) Positive-edge response



(c) Negative-edge response

## 5.4 Flip-flops – Master-Slave Flip-Flop

- Negative edge triggered D flip-flop
- $C=0$  : master disable, slave enable
- Output has no relation with input
- $C=1$  : master enable, slave disable

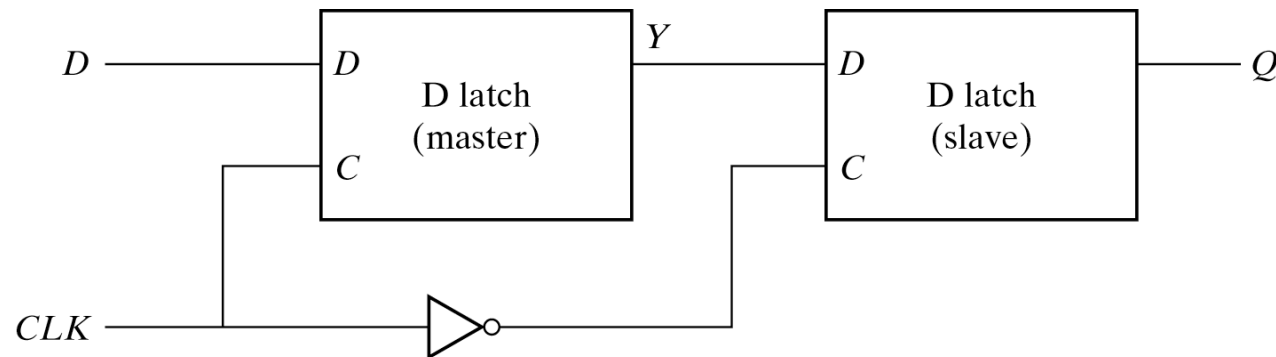
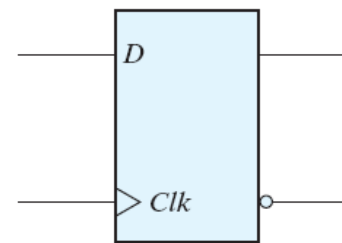
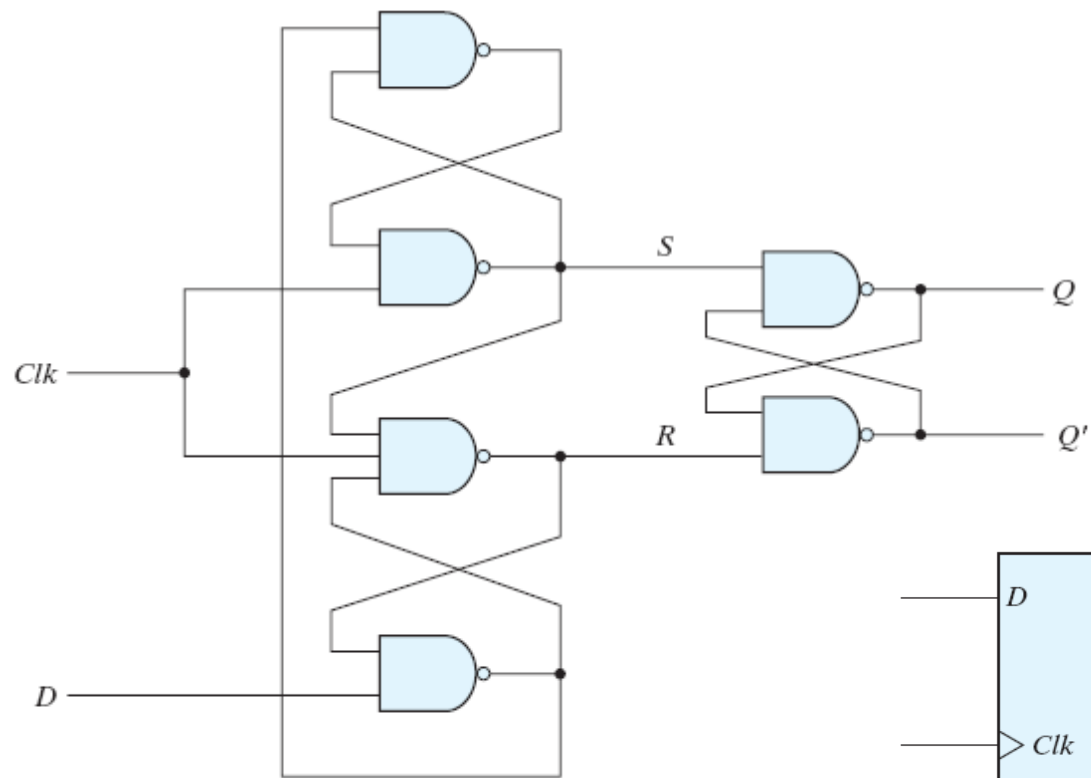


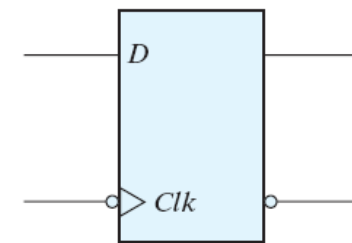
Fig. 5-9 Master-Slave  $D$  Flip-Flop

## 5.4 Flip-flops - Edge-Triggered Flip-Flop

- D-type positive edge triggered flip flop
  - Consist of 3 SR-latches
  - Q changes only when C becomes 0 to 1



(a) Positive-edge

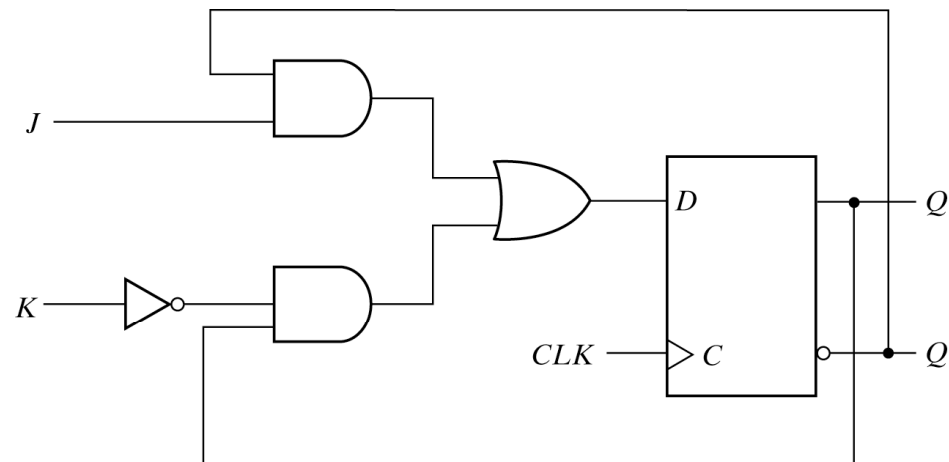


(a) Negative-edge

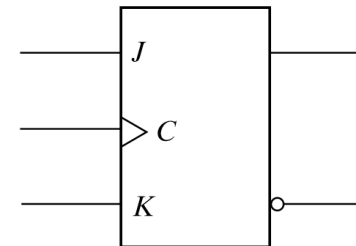
## 5.4 Flip-flops - Other Flip-Flop

### ● JK flip-flop

- Performs three operations
- Set(J), Reset(K), Complement(J=K=1)
- $D = JQ' + K'Q$



(a) Circuit diagram



(b) Graphic symbol

Fig. 5-12 JK Flip-Flop

## 5.4 Flip-flops - Other Flip-Flop

- T flip-flop
  - Complementing flip-flop
  - $D = TQ' + T'Q$

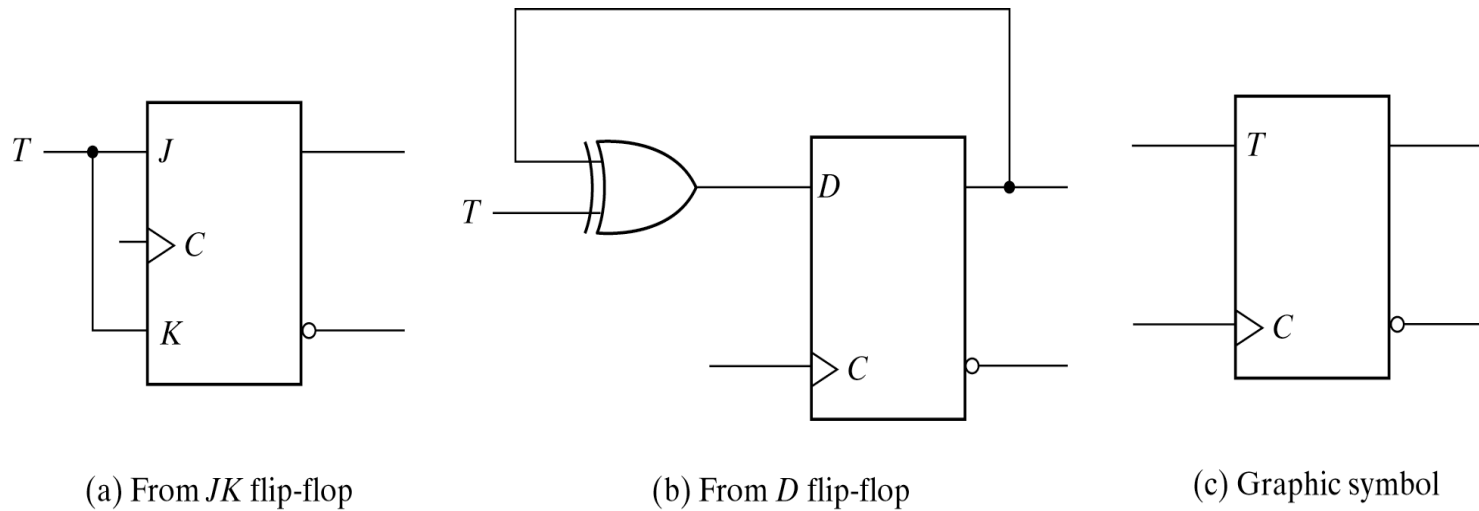


Fig. 5-13 T Flip-Flop

## 5.4 Flip-flops - Characteristic tables

### ● Flip-flop characteristic tables

$$Q(t+1) = JQ' + K'Q$$

**JK Flip-Flop**

<i>J</i>	<i>K</i>	$Q(t + 1)$	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

**SR Flip-Flop**

<i>S</i>	<i>R</i>	$Q(t+1)$	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined

**D Flip-Flop**

<i>D</i>	$Q(t + 1)$	
0	0	Reset
1	1	Set

$$Q(t+1) = D$$

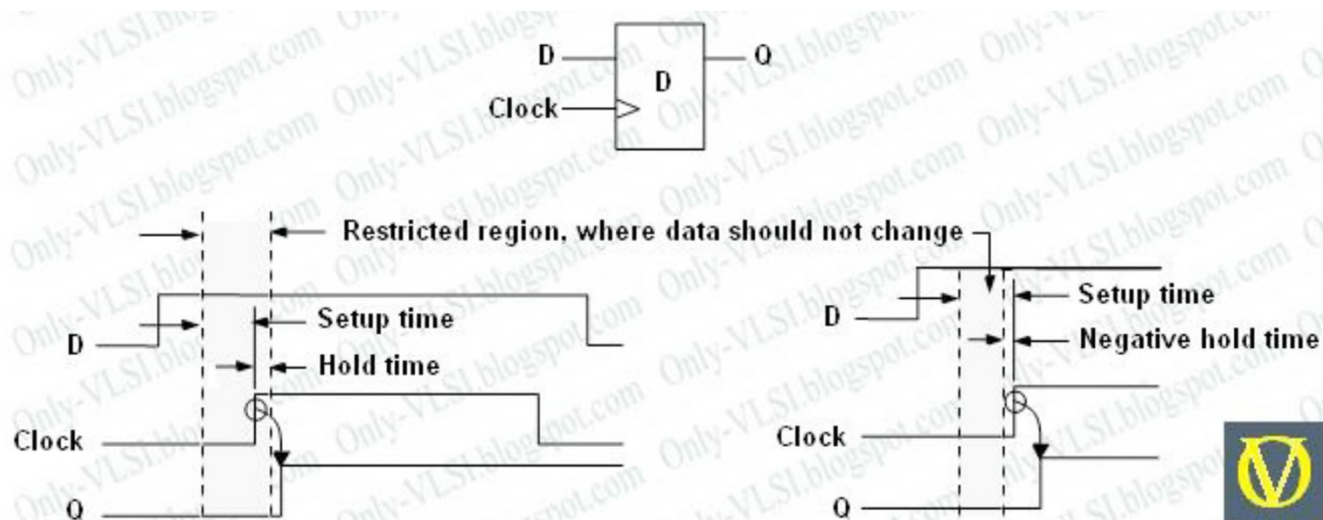
**T Flip-Flop**

<i>T</i>	$Q(t + 1)$	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

$$Q(t+1) = TQ' + T'Q$$

## 5.4 Flip-flops - Terminologies

- *Setup time*: Time in which D input must be maintained at a constant value prior to applying the clock.
- *Hold time*: Time when D input must not change after the application of the positive transition of the pulse.
- *Propagation delay time*: Time interval between the trigger edge and the stabilization of the output to the new state.



(Source: [only-vlsi.blogspot.com](http://only-vlsi.blogspot.com))

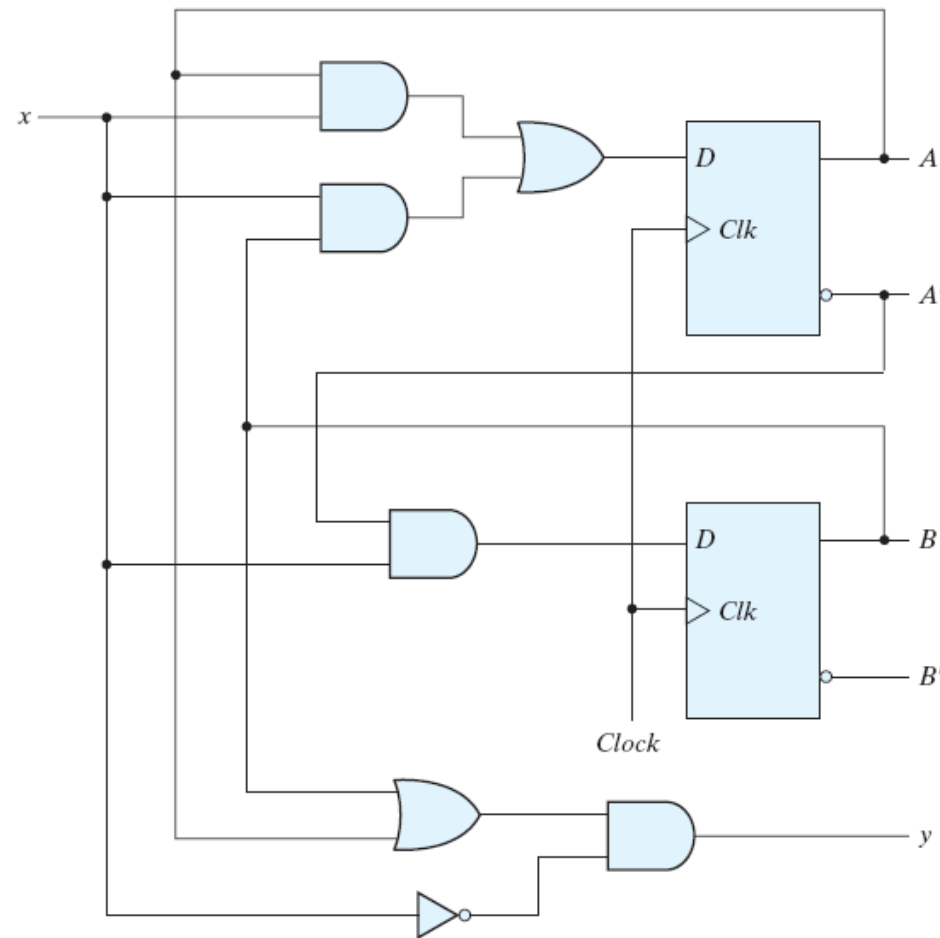
## 5.5 Analysis of clocked sequential circuits

- Behavior of clocked sequential circuit is determined from input, output and present state
- Output, next state are a function of input and present state



## 5.5 Analysis of clocked sequential circuits - State equations

- Specifies the next state and output as a function of the present state and inputs
- $A(t+1) = Ax + Bx$
- $B(t+1) = A'x$
- $Y = (A+B)x'$



## 5.5 Analysis of clocked sequential circuits - State table

- Time sequence table of inputs, outputs and flip-flop states
- two types of state table exist

**Table 5-2**  
*State Table for the Circuit of Fig. 5-15*

Present State		Input	Next State		Output
A	B		A	B	
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 5-3**  
*Second Form of the State Table*

Present State	Next State		Output	
	x = 0	x = 1	x = 0	x = 1
AB	AB	AB	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

## 5.5 Analysis of clocked sequential circuits - State diagram

- A kind of flow diagram
- Can be derived from state table
- State-circle, transition-line, I/O

**Table 5-2**  
*State Table for the Circuit of Fig. 5-15*

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

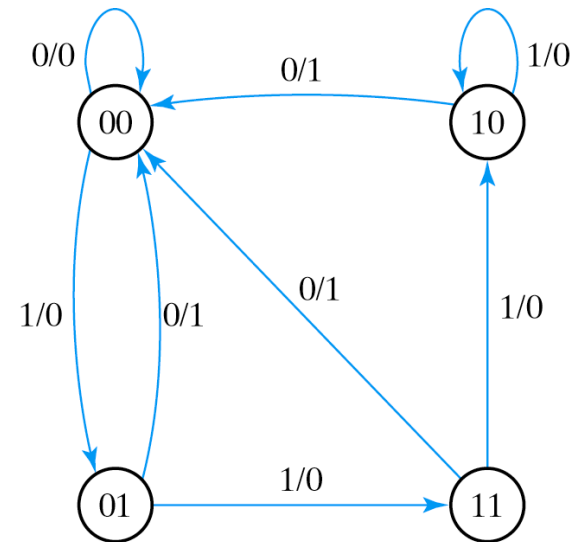
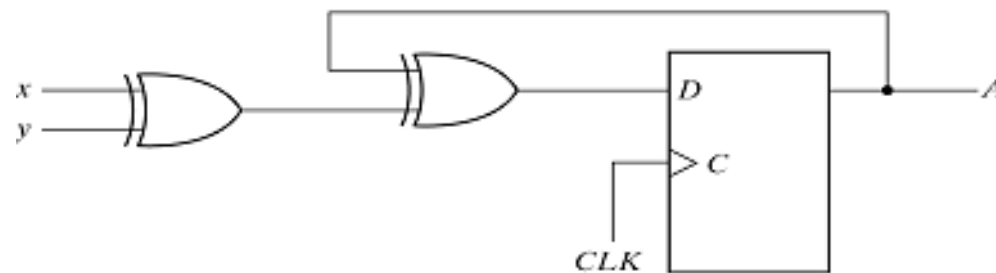


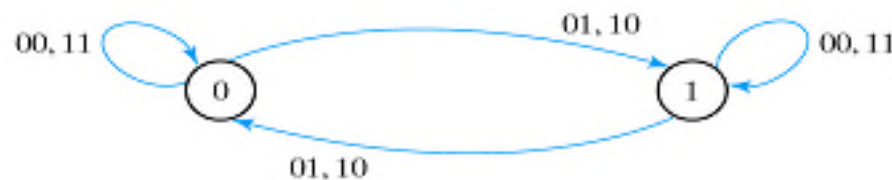
Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

## 5.5 Analysis of clocked sequential circuits - Analysis with D flip-flops

- Input equation :  $D_A = A \oplus x \oplus y$
- State equation is equal to input equation



(a) Circuit diagram



(c) State diagram

Present state	Inputs		Next state
$A$	$x$	$y$	$A$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) State table

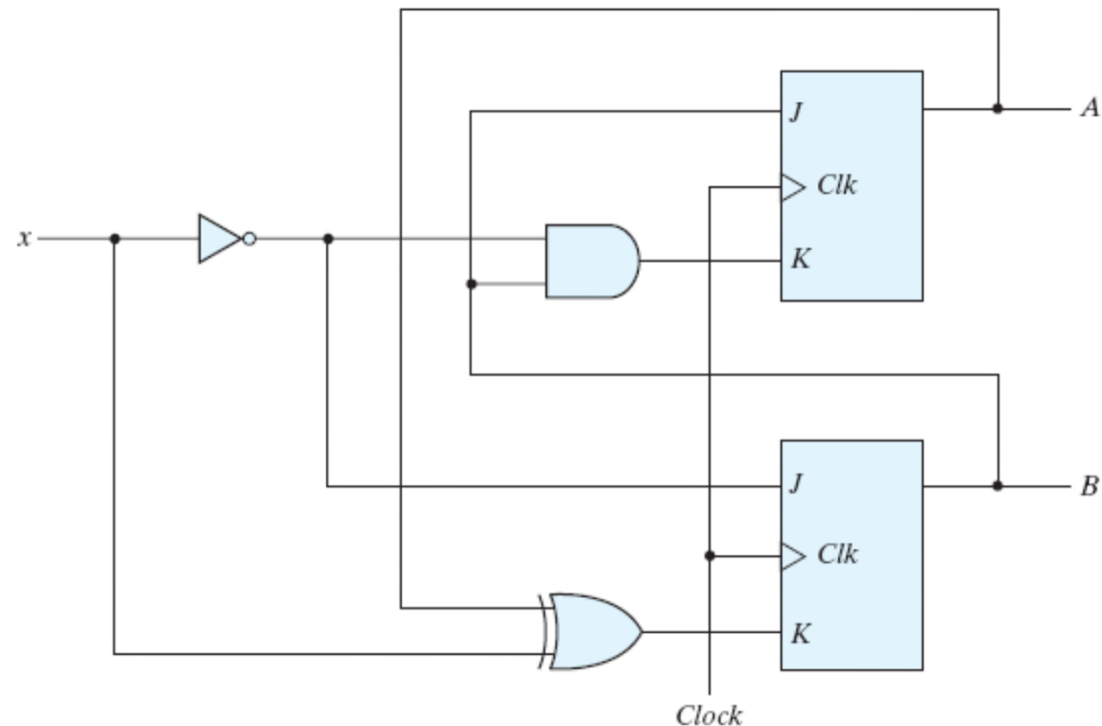
Fig. 5-17 Sequential Circuit with D Flip-Flop

## 5.5 Analysis of clocked sequential circuits - Analysis with JK flip-flops

- State equation is not the same as the input equation
- Have to refer characteristic table or characteristic equation
- Input equations

$$J_A = B \quad K_A = Bx'$$

$$J_B = x' \quad K_B = A'x + Ax'$$



## 5.5 Analysis of clocked sequential circuits - Analysis with JK flip-flops

### State table and state diagram

**Table 5-4**  
*State Table for Sequential Circuit with JK Flip-Flops*

Present State		Input	Next State		Flip-Flop Inputs			
A	B		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

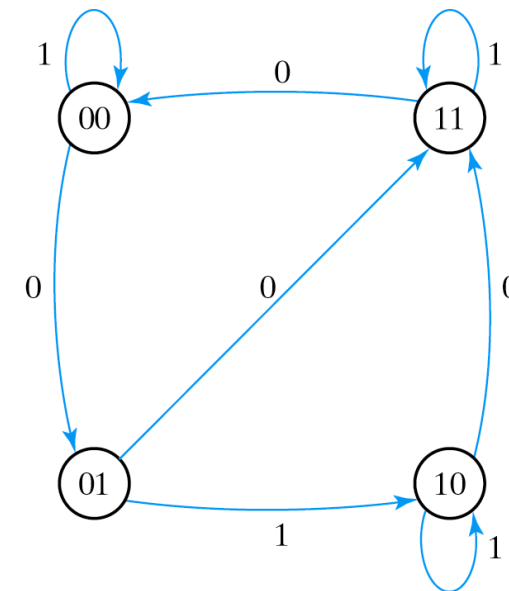


Fig. 5-19 State Diagram of the Circuit of Fig. 5-18

## 5.5 Analysis of clocked sequential circuits - Analysis with T flip-flops

- Input equations and output equation

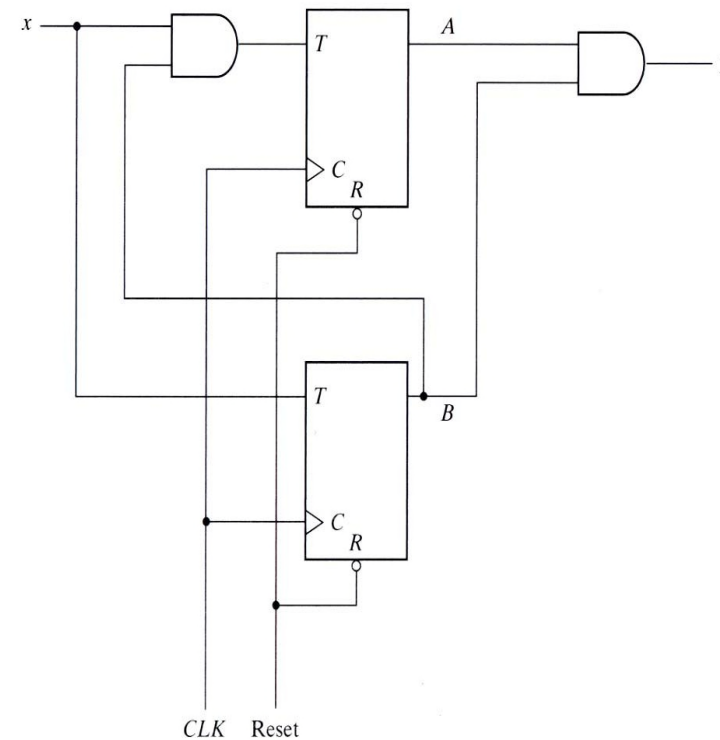
$$T_A = Bx, \quad T_B = x$$

$$y = AB$$

- State equations are derived from characteristic equation

$$A(t+1) = T_A A' + T_A' A$$

$$B(t+1) = T_B B' + T_B' B$$

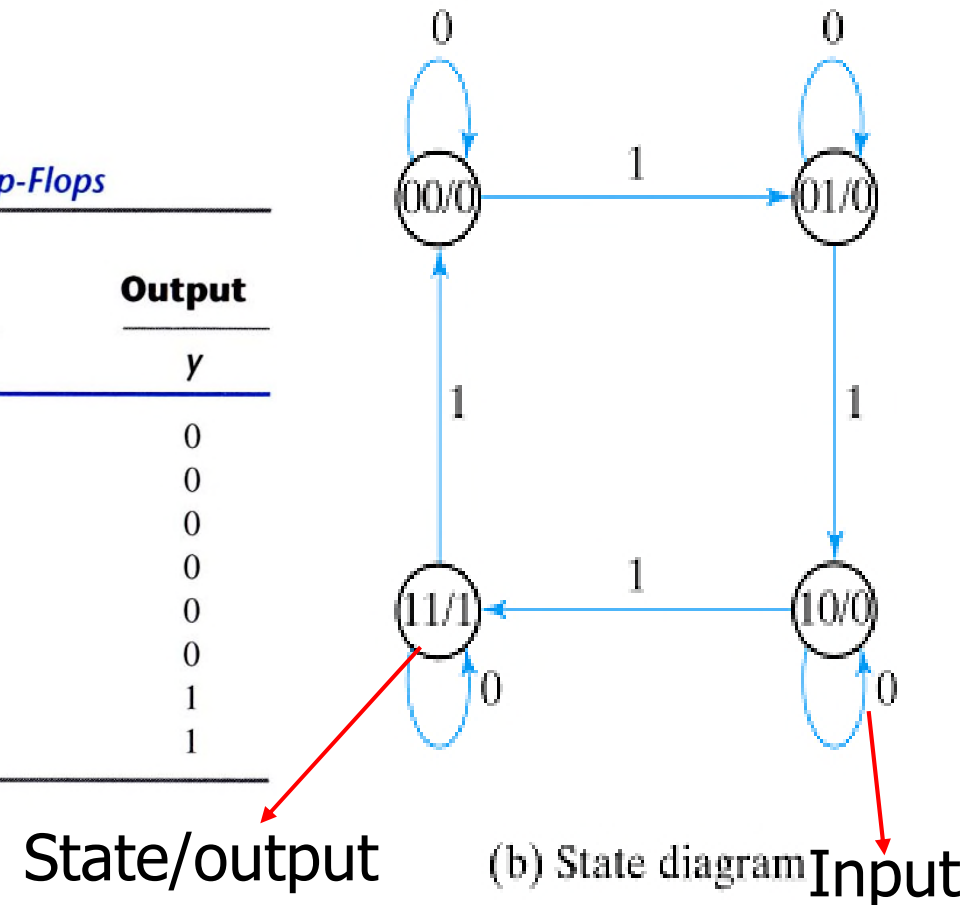


(a) Circuit diagram

## 5.5 Analysis of clocked sequential circuits - Analysis with T flip-flops

**Table 5-5**  
*State Table for Sequential Circuit with T Flip-Flops*

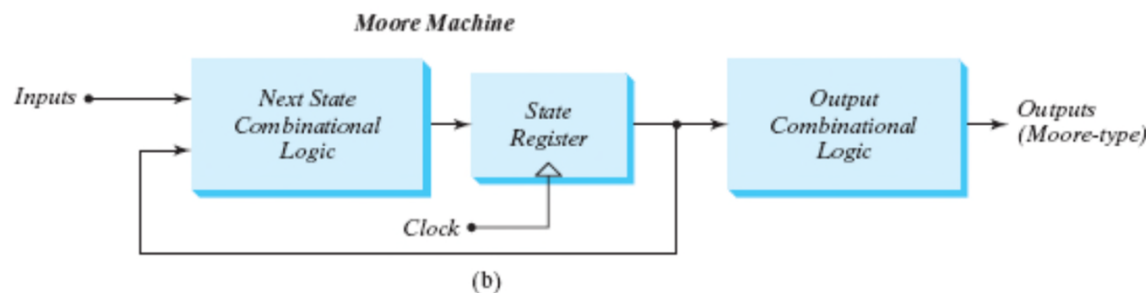
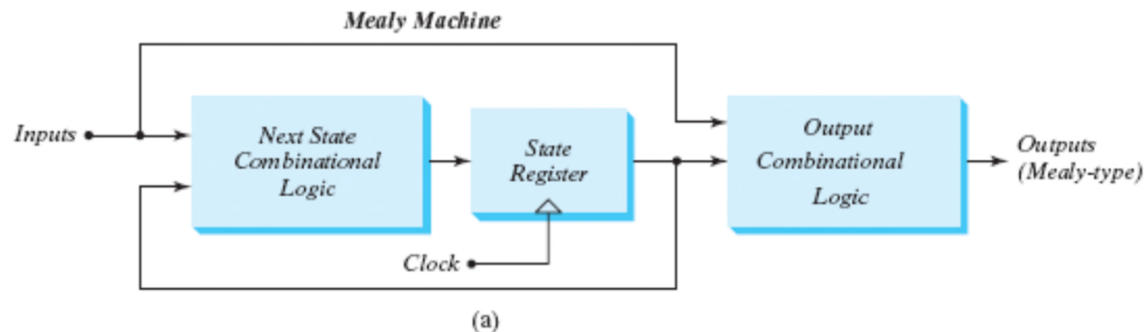
Present State		Input $x$	Next State		Output $y$
$A$	$B$		$A$	$B$	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1





## 5.5 Analysis of clocked sequential circuits - Mealy and Moore models

- Mealy model : output is a function of the present state and input
- Inputs must be synchronized with the clock
- Outputs must be sampled at the clock edge
- Moore model : output is a function of the present state only
- Outputs are synchronized with the clock

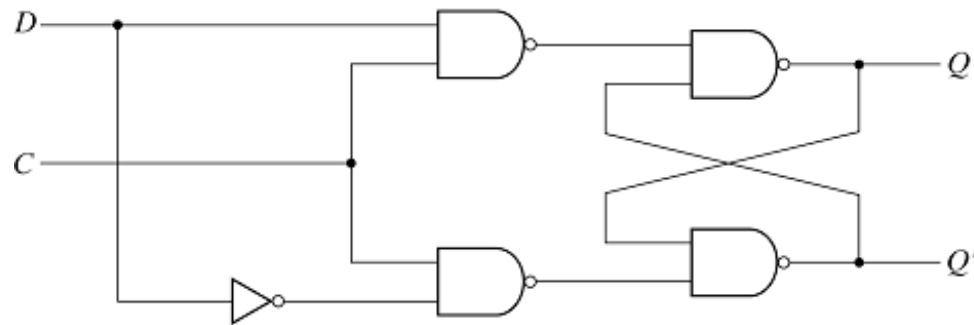


## 5.6 HDL for sequential circuits

- Two kinds of behavioral statements
- Initial : executes only once
- Always : executes repeatedly until the simulation terminates

## 5.6 HDL for sequential circuits - Flip-flops and latches

### ● D-latch



#### HDL Example 5-1

```
//Description of D latch (See Fig. 5-6)
module D_latch (Q,D,control);
    output Q;
    input D,control;
    reg Q;
    always @ (control or D)
        if (control) Q = D;      //Same as: if (control == 1)
endmodule
```

## 5.6 HDL for sequential circuits - Flip-flops and latches

### ● D flip-flop

#### - D flip-flop with asynchronous reset

---

```
//D flip-flop
module D_FF (Q,D,CLK);
    output Q;
    input D,CLK;
    reg Q;
    always @ (posedge CLK)
        Q = D;
endmodule

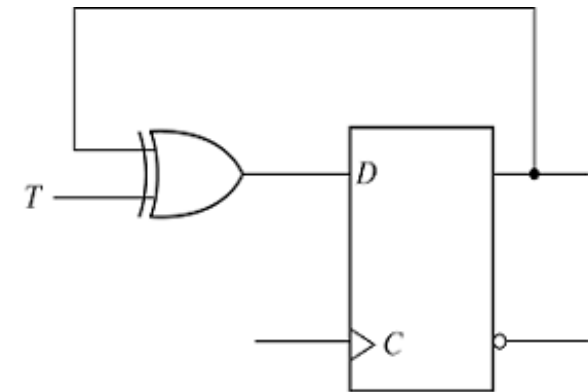
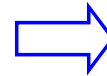
//D flip-flop with asynchronous reset.
module DFF (Q,D,CLK,RST);
    output Q;
    input D,CLK,RST;
    reg Q;
    always @(posedge CLK or negedge RST)
        if (~RST) Q = 1'b0;    // Same as: if (RST == 0)
        else Q = D;
endmodule
```

---

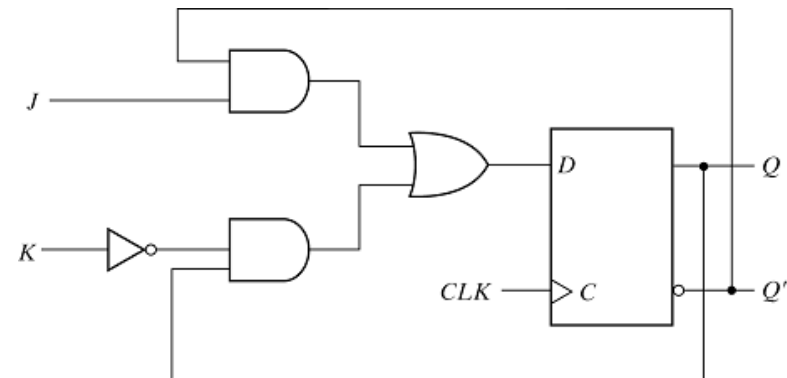
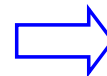
## 5.6 HDL for sequential circuits - Flip-flops and latches

### ● T flip-flop from D flip-flop

```
//T flip-flop from D flip-flop and gates
module TFF (Q,T,CLK,RST);
    output Q;
    input T,CLK,RST;
    wire DT;
    assign DT = Q ^ T ;
    //Instantiate the D flip-flop
    DFF TF1 (Q,DT,CLK,RST);
endmodule
```

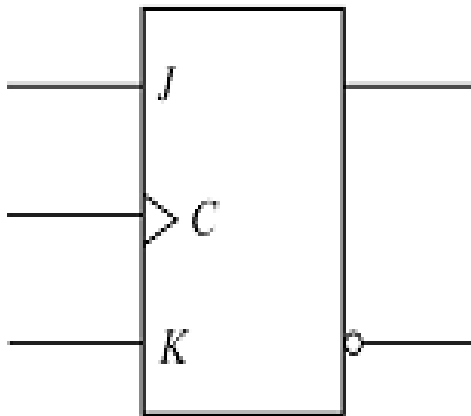


```
//JK flip-flop from D flip-flop and gates
module JKFF (Q,J,K,CLK,RST);
    output Q;
    input J,K,CLK,RST;
    wire JK;
    assign JK = (J & ~Q) | (~K & Q);
    //Instantiate D flipflop
    DFF JK1 (Q,JK,CLK,RST);
endmodule
```



## 5.6 HDL for sequential circuits - Flip-flops and latches

### ● JK flip-flop



#### HDL Example 5-4

```
// Functional description of JK flip-flop
module JK_FF (J,K,CLK,Q,Qnot);
    output Q,Qnot;
    input J,K,CLK;
    reg Q;
    assign Qnot = ~ Q ;
    always @ (posedge CLK)
        case ({J,K})
            2'b00: Q = Q;
            2'b01: Q = 1'b0;
            2'b10: Q = 1'b1;
            2'b11: Q = ~ Q;
        endcase
endmodule
```

## 5.6 HDL for sequential circuits - State diagram

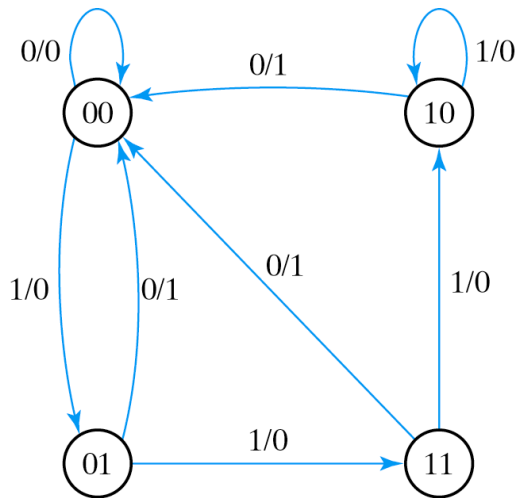


Fig. 5-16 State Diagram of the Circuit of Fig. 5-15

(Mealy state diagram)

```
module Mealy_md1 (x,y,CLK,RST);
  input x,CLK,RST;
  output y;
  reg y;
  reg [1:0] Prstate, Nxtstate;
  parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
  always @ (posedge CLK or negedge RST)
    if (~RST) Prstate = S0; //Initialize to state S0
    else Prstate = Nxtstate; //Clock operations
  always @ (Prstate or x) //Determine next state
    case (Prstate)
      S0: if (x) Nxtstate = S1;
          else Nxtstate = S0;
      S1: if (x) Nxtstate = S3;
          else Nxtstate = S0;
      S2: if (~x) Nxtstate = S0;
          else Nxtstate = S2;
      S3: if (x) Nxtstate = S2;
          else Nxtstate = S0;
    endcase
  always @ (Prstate or x) //Evaluate output
    case (Prstate)
      S0: y = 0;
      S1: if (x) y = 1'b0; else y = 1'b1;
      S2: if (x) y = 1'b0; else y = 1'b1;
      S3: if (x) y = 1'b0; else y = 1'b1;
    endcase
endmodule
```

## 5.6 HDL for sequential circuits - State diagram

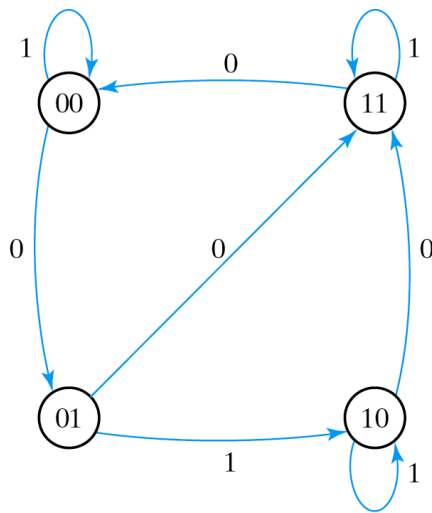


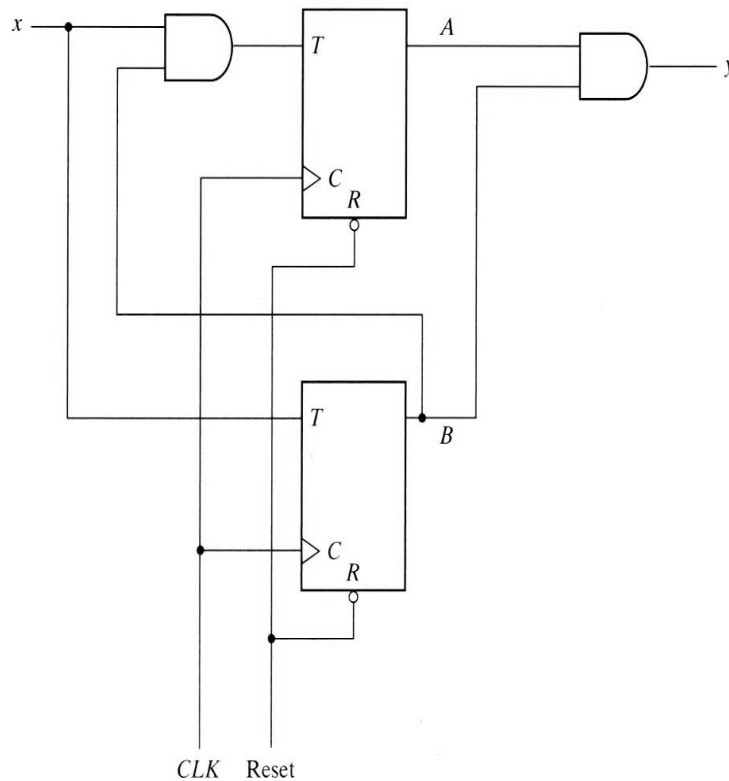
Fig. 5-19 State Diagram of the Circuit of Fig. 5-18

(Moore state diagram)

```
//Moore state diagram (Fig. 5-19)
module Moore_mdl (x,AB,CLK,RST);
  input x,CLK,RST;
  output [1:0]AB;
  reg [1:0] state;
  parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
  always @ (posedge CLK or negedge RST)
    if (~RST) state = S0; //Initialize to state S0
    else
      case (state)
        S0: if (~x) state = S1; else state = S0;
        S1: if (x) state = S2; else state = S3;
        S2: if (~x) state = S3; else state = S2;
        S3: if (~x) state = S0; else state = S3;
      endcase
  assign AB = state; //Output of flip-flops
endmodule
```



## 5.6 HDL for sequential circuits - Structural description



(a) Circuit diagram

```
module Tcircuit (x,y,A,B,CLK,RST);  
    input x,CLK,RST;  
    output y,A,B;  
    wire TA,TB;  
    //Flip-flip input equations  
    assign TB = x,  
           TA = x & B;  
    //Output equation  
    assign y = A & B;  
    //Instantiate T flip-flops  
    T_FF BF (B,TB,CLK,RST);  
    T_FF AF (A,TA,CLK,RST);  
endmodule
```

```
module T_FF (Q,T,CLK,RST);  
    output Q;  
    input T,CLK,RST;  
    reg Q;  
    always @ (posedge CLK or negedge RST)  
        if (~RST) Q = 1'b0;  
        else Q = Q ^ T;  
endmodule
```

## 5.6 HDL for sequential circuits - Structural description

```
module testTcircuit;
  reg x,CLK,RST; //inputs for circuit
  wire y,A,B;    //output from circuit
  Tcircuit TC (x,y,A,B,CLK,RST); // instantiate circuit
  initial
    begin
      RST = 0;
      CLK = 0;
      #5 RST = 1;
      repeat (16)
        #5 CLK = ~CLK;
      end
  initial
    begin
      x = 0;
      #15 x = 1;
      repeat (8)
        #10 x = ~ x;
      end
  end
endmodule
```

(Stimulus and output)

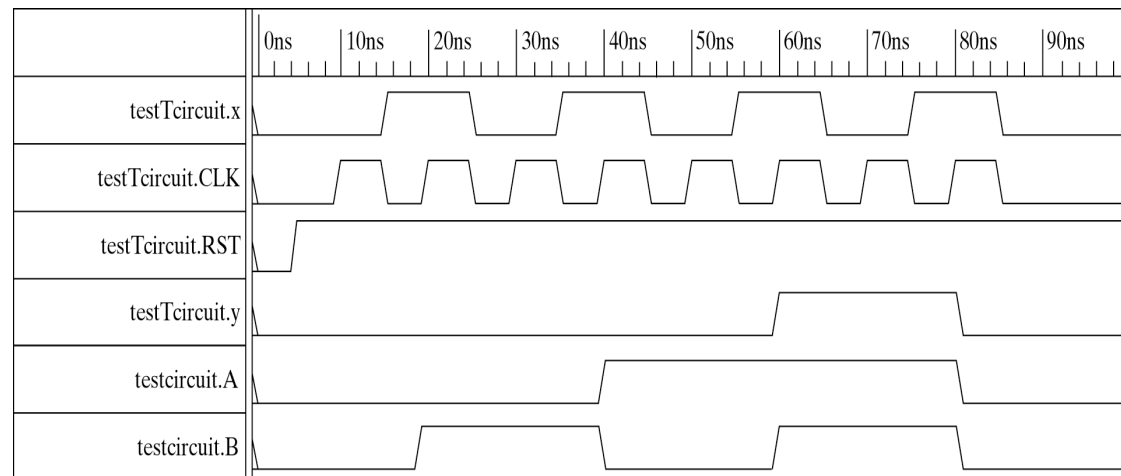


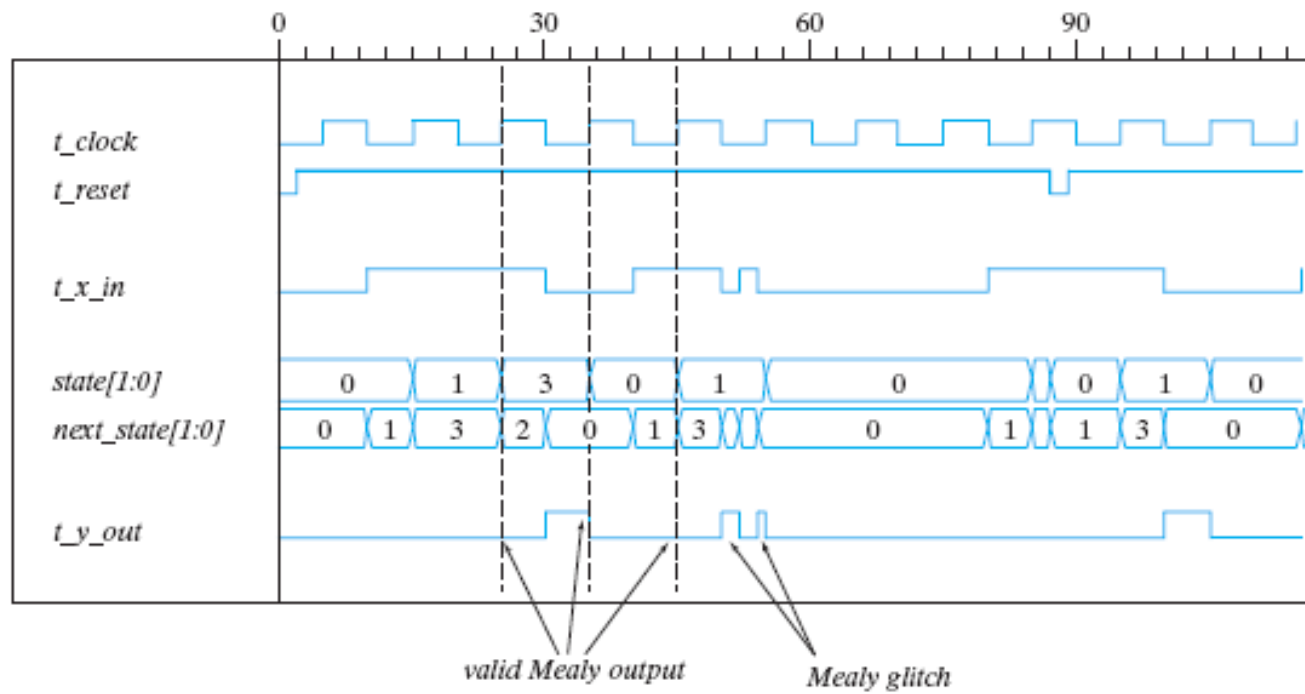
Fig. 5-21 Simulation Output of HDL Example 5-7

## 5.6 HDL for sequential circuits

```
module Mealy_Zero_Detector (  
    output reg y_out,  
    input      x_in, clock, reset  
);  
    reg [1: 0]    state, next_state;  
    parameter     S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;  
  
    always @ (posedge clock, negedge reset)    // state transition  
        if (reset == 0) state <= S0;  
        else state <= next_state;  
  
    always @ (state, x_in) // Form the next state  
        case (state)  
            S0:    if (x_in) next_state = S1; else next_state = S0;  
            S1:    if (x_in) next_state = S3; else next_state = S0;  
            S2:    if (~x_in) next_state = S0; else next_state = S2;  
            S3:    if (x_in) next_state = S2; else next_state = S0;  
        endcase  
  
    always @ (state, x_in) // Form the output  
        case (state)  
            S0:    y_out = 0;  
            S1, S2, S3: y_out = ~x_in;  
        endcase  
endmodule  
  
module t_Mealy_Zero_Detector;  
    wire t_y_out;  
    reg t_x_in, t_clock, t_reset;  
  
    Mealy_Zero_Detector M0 (t_y_out, t_x_in, t_clock, t_reset);  
  
    initial #200 $finish;  
    initial begin t_clock = 0; forever #5 t_clock = ~t_clock; end
```

```
initial fork  
    t_reset = 0;  
    #2 t_reset = 1;  
    #87 t_reset = 0;  
    #89 t_reset = 1;  
    #10 t_x_in = 1;  
    #30 t_x_in = 0;  
    #40 t_x_in = 1;  
    #50 t_x_in = 0;  
    #52 t_x_in = 1;  
    #54 t_x_in = 0;  
    #70 t_x_in = 1;  
    #80 t_x_in = 1;  
    #70 t_x_in = 0;  
    #90 t_x_in = 1;  
    #100 t_x_in = 0;  
    #120 t_x_in = 1;  
    #160 t_x_in = 0;  
    #170 t_x_in = 1;  
  
join  
endmodule
```

## 5.6 HDL for sequential circuits



## 5.7 State reduction and assignment

- State reduction is used to reduce the number of flip-flop
- Only input/output sequences are important
- Interested in present states that go to the same next state and have the same output

## 5.7 State reduction and assignment - State reduction

**Table 5-6**  
*State Table*

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
<i>a</i>	<i>a</i>	<i>b</i>	0	0
<i>b</i>	<i>c</i>	<i>d</i>	0	0
<i>c</i>	<i>a</i>	<i>d</i>	0	0
<i>d</i>	<i>e</i>	<i>f</i>	0	1
<i>e</i>	<i>a</i>	<i>f</i>	0	1
<i>f</i>	<i>g</i>	<i>f</i>	0	1
<i>g</i>	<i>a</i>	<i>f</i>	0	1

*Reducing the State Table*

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
<i>a</i>	<i>a</i>	<i>b</i>	0	0
<i>b</i>	<i>c</i>	<i>d</i>	0	0
<i>c</i>	<i>a</i>	<i>d</i>	0	0
<i>d</i>	<i>e</i>	<i>f</i>	0	1
<i>e</i>	<i>a</i>	<i>f</i>	0	1
<i>f</i>	<i>e</i>	<i>f</i>	0	1

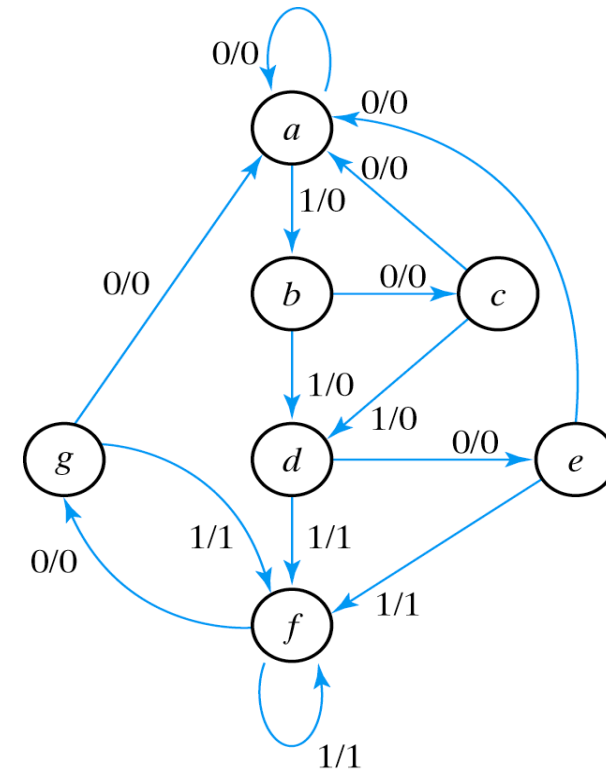


Fig. 5-22 State Diagram

## 5.7 State reduction and assignment - State reduction

**Table 5-8**  
*Reduced State Table*

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
$a$	$a$	$b$	0	0
$b$	$c$	$d$	0	0
$c$	$a$	$d$	0	0
$d$	$e$	$d$	0	1
$e$	$a$	$d$	0	1

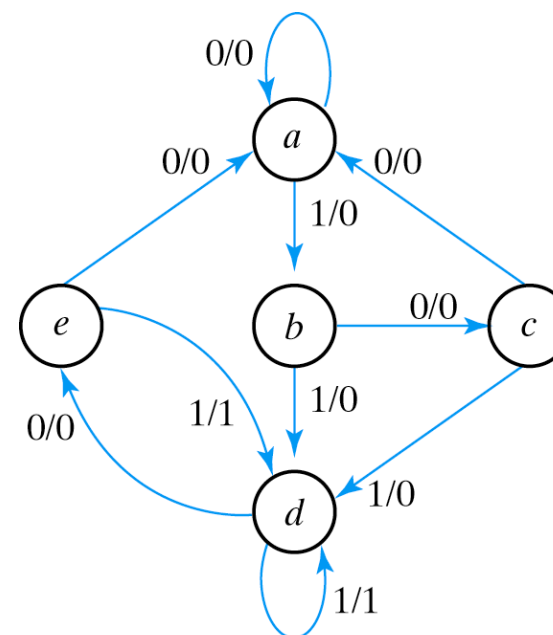


Fig. 5-23 Reduced State Diagram

## 5.7 State reduction and assignment - State assignment

- $m$  states circuit, codes must contain  $n$  bits where  $2^n \geq m$
- Three possible binary state assignments

**Table 5-9**  
*Three Possible Binary State Assignments*

State	Assignment 1 Binary	Assignment 2 Gray code	Assignment 3 One-hot
<i>a</i>	000	000	00001
<i>b</i>	001	001	00010
<i>c</i>	010	011	00100
<i>d</i>	011	010	01000
<i>e</i>	100	110	10000



## 5.8 Design procedure

- Sequential circuit design : requires state table  
⇒ Combinational circuit : truth table
- The number of flip-flop is determined from the number of states in circuit
- If  $2^n$  states exist, there are  $n$  flip-flops

## 5.8 Design procedure

- Design steps

- 1) Derive a state diagram or state table
- 2) Reduce the number of states if necessary
- 3) Assign binary code to the state
- 4) Choose the type of flip-flops to be used
- 5) Derive the flip-flop input equations and output equations
- 6) Draw the logic diagram

## 5.8 Design procedure

- Derive a state diagram
  - Sequential detector
  - Three or more consecutive 1's in a string of bits coming through an input line

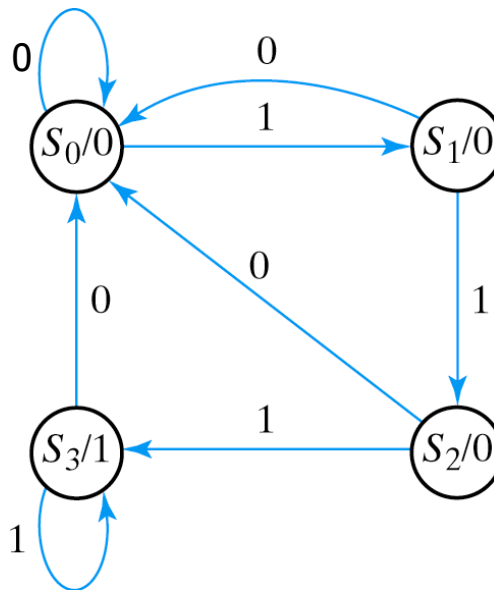


Fig. 5-24 State Diagram for Sequence Detector

## 5.8 Design procedure - Synthesis using D flip-flops

- Input equations are obtained directly from the next states

**Table 5-11**  
*State Table for Sequence Detector*

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

## 5.8 Design procedure - Synthesis using D flip-flops

### ● K-maps and logic diagram

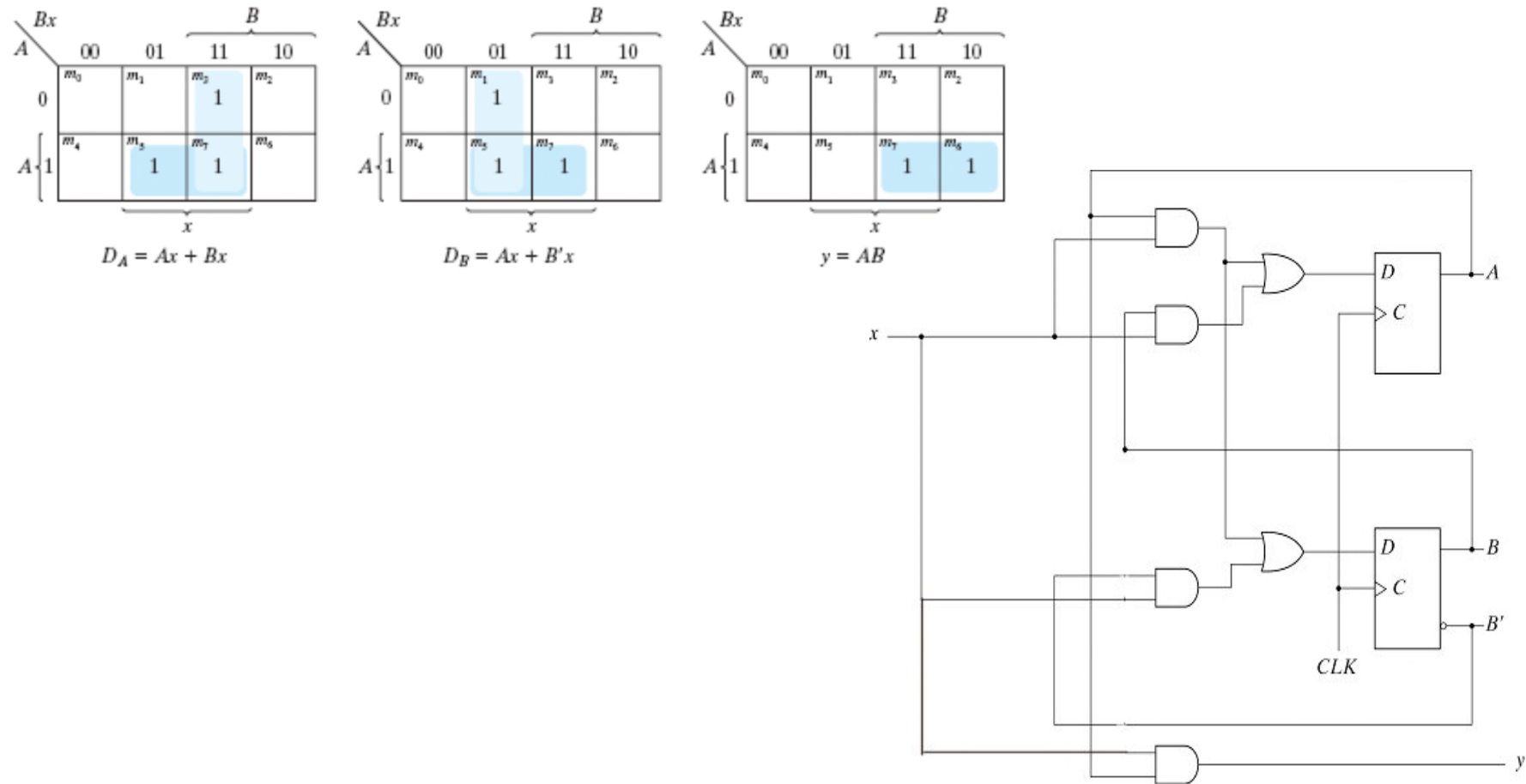


Fig. 5-26 Logic Diagram of Sequence Detector

## 5.8 Design procedure - Excitation table

### ● Excitation Table

**Table 5.12**

*Flip-Flop Excitation Tables*

$Q(t)$	$Q(t = 1)$	$J$	$K$
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(a)  $JK$

$Q(t)$	$Q(t = 1)$	$T$
0	0	0
0	1	1
1	0	1
1	1	0

(b)  $T$

## 5.8 Design procedure - Synthesis with JK flip-flops

- Input equations evaluated from the present state to next state transition

**Table 5-13**  
*State Table and JK Flip-Flop Inputs*

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J <sub>A</sub>	K <sub>A</sub>	J <sub>B</sub>	K <sub>B</sub>
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

## 5.8 Design procedure - Synthesis with JK flip-flops

### ● K-maps and logic diagram

	$Bx$		$B$	
	00	01	11	10
$A$				
0				1
1	X	X	X	X

$x$

$$J_A = Bx'$$

	$Bx$		$B$	
	00	01	11	10
$A$				
0	X	X	X	X
1			1	

$x$

$$K_A = Bx$$

	$Bx$		$B$	
	00	01	11	10
$A$				
0		1	X	X
1		1	X	X

$x$

$$J_B = x$$

	$Bx$		$B$	
	00	01	11	10
$A$				
0	X	X		1
1	X	X	1	

$x$

$$K_B = (A \oplus x)'$$

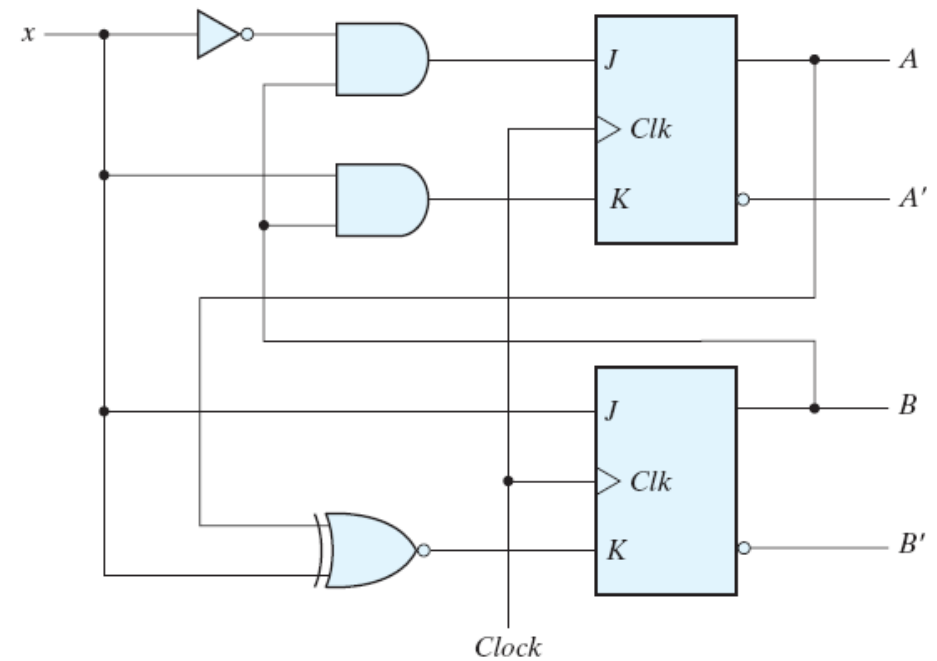


Fig. 5-27 Maps for J and K Input Equations



## 5.8 Design procedure - Synthesis using T flip-flops

- 3-bit binary counter
- 3-bit counter has 3 flip-flops and can count from 0 to  $2^n - 1$  ( $n=3$ )

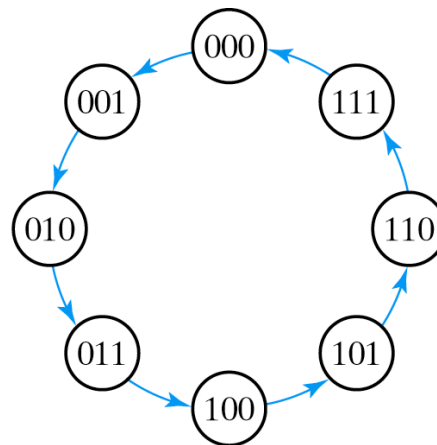


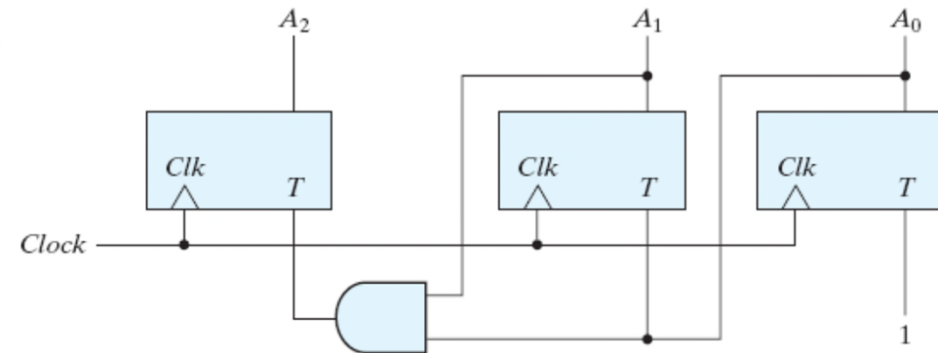
Fig. 5-29 State Diagram of 3-Bit Binary Counter

## 5.8 Design procedure - Synthesis using T flip-flops

### State table and logic diagram

**Table 5-14**  
State Table for 3-Bit Counter

Present State			Next State			Flip-Flop Inputs		
$A_2$	$A_1$	$A_0$	$A_2$	$A_1$	$A_0$	$T_{A2}$	$T_{A1}$	$T_{A0}$
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1



$$T_{A2}=A_1A_0, \quad T_{A1}=A_0, \quad T_{A0}=1$$

