Chapter 5

Synchronous Sequential Logic

Introduction

- Circuits require memory to store intermediate data
- Sequential circuits use a periodic signal to determine when to store values.
  - A clock signal can determine storage times
  - Clock signals are periodic
- Single bit storage element is a flip flop
- A basic type of flip flop is a latch
- Latches are made from logic gates
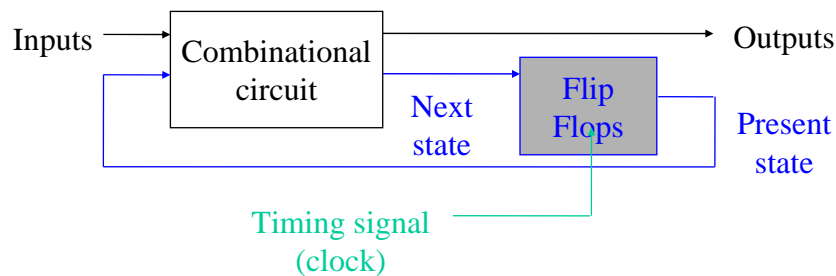 - NAND, NOR, AND, OR, Inverter

Synchronous vs. Asynchronous

- Synchronous sequential circuit is a system whose behavior can be defined from the knowledge of its signal at discrete instants of time

- Asynchronous sequential circuit is a system whose behavior can be defined from the knowledge of its signals at any point of time

Synchronous Sequential Circuit

Inputs → Combinational circuit → Outputs

Next state

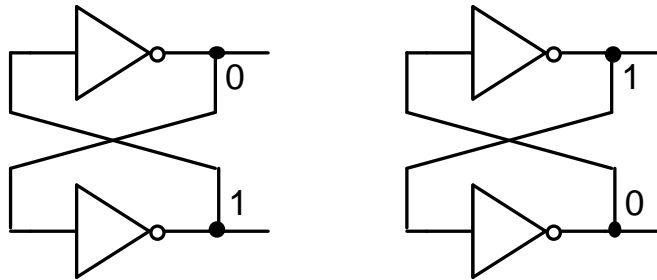Flip Flops

Present state

Timing signal (clock)

Clock

Clock

synchronizes when current state changes happen
keeps system well-behaved
makes it easier to design and build large systems
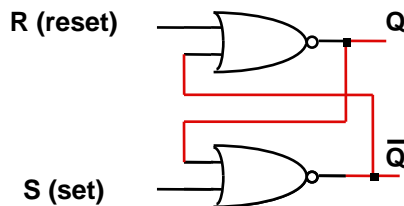
4

Cross Coupled Inverters

State 1

State 2

What if there is an inverter in the feedback path?

5

SR Latch using NOR

R (reset)   Q

$\overline{Q}$

S (set)

S	R	Q	Q'	
1	1			
1	0	1	0	Set
0	1	0	1	Reset
0	0	0	1	Stable
		1	0	

° S-R latch made from **cross-coupled** NORs

° If Q = 1, set state

° If Q = 0, reset state

° Usually S=0 and R=0

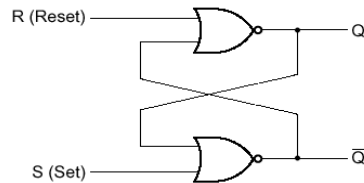° S=1 and R=1 generates unpredictable results

6

3

SR latch using NAND
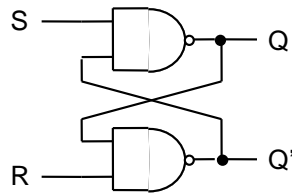
S	R	Q	Q'	
0	0	**1**	**1**	**Disallowed**
0	1	**1**	**0**	**Set**
1	0	**0**	**1**	**Reset**
1	1	**0**	**1**	**Store**

- Latch made from cross-coupled NANDs
- Sometimes called S'-R' latch
- Usually S=1 and R=1
- S=0 and R=0 generates unpredictable results

SR Latch
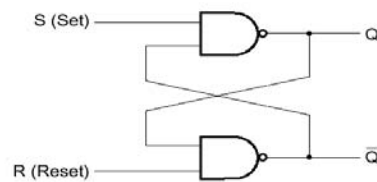
R (Reset) ——— Q

S (Set) ——— Q̄

(a) Logic diagram

S	R	Q	Q̄	
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

S (Set) ——— Q

R (Reset) ——— Q̄

(a) Logic diagram

S	R	Q	Q̄	
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
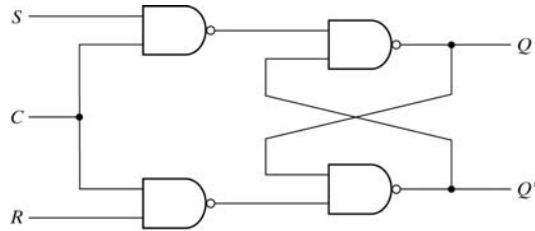
SR Latch with Control Input

| 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

(a) Logic diagram
(b) Function table

Fig. 5-5 SR Latch with Control Input

°**Occasionally, desirable to avoid latch changes**

°**C = 0 disables all latch state changes**

°**Control signal enables data change when C = 1**

°**Right side of circuit same as ordinary S-R latch.**
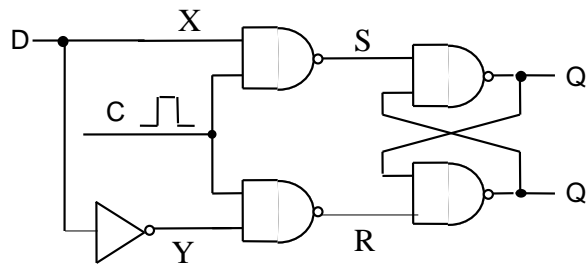

SR Latch with Control Input

R'

C'

S'

Q

Q'
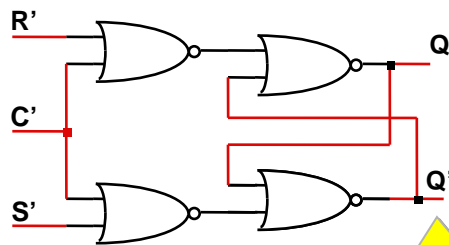
Outputs change when C is low: RESET and SET Otherwise: HOLD
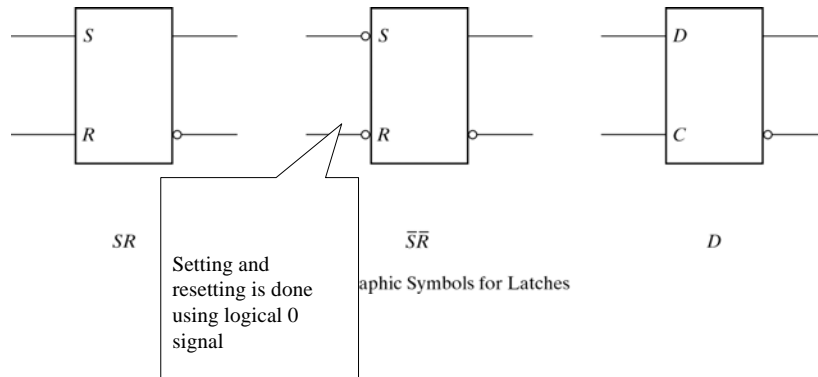
D Latch

D Latch

- One wat to avoid the indeterminate state in the SR latch when both S and R are This is done by inverting S and having the inverted S as R.

- As long as C=0, no change. If C=1 D is delivered to Q and D' to Q'

C	D	Q	Q'
0	X	Q	Q'
1	1	1	0
1	0	0	1

Symbols

SR	$\overline{S}\overline{R}$	*D*

Setting and resetting is done using logical 0 signal

aphic Symbols for Latches

Flip-Flops

- The state of the latch or flip-flop is switched by a change in the control input, this is called a *trigger*
- The D latch with pulses in its control input. As long as the pulse input is 1, any changes in the D input is transferred to the output.
- When the state of the latch changes, if the control pulse is still at logic 1, any changes to the input (possible a feedback from the output) will change the output.
- Because of this, the output of a latch can not be applied directly to the input of this or other latches

Flip-Flop

- A better way is if the flip-flop is triggered during the transition of the control input.
- A clock pulse goes through 2 transitions, 1->0 and 0->1.
- The first is called –ve edge, the second is positive edge.
- There are two ways to implement this, either by using master-salve or by special design of the circuit.

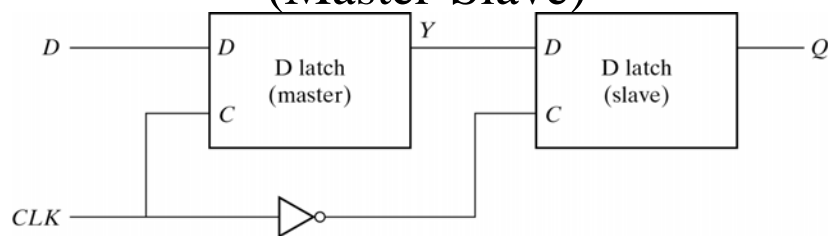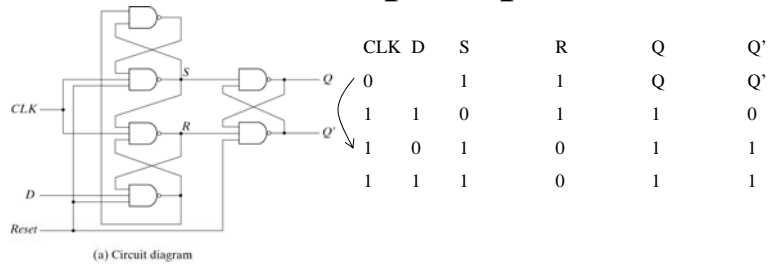# Edge-Triggered Flip-Flop (Master-Slave)

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

Any changes in the input can affect only Y as long as CLK=1; After CLK=0, Y propagates to Q, byt the master is locked.
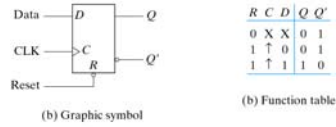
D-Type Positive-Edge-triggered Flip-Flop

CLK	D	S	R	Q	Q'
0		1	1	Q	Q'
1	1	0	1	1	0
1	0	1	0	1	1
1	1	1	0	1	1

(a) Circuit diagram

Data — D   Q
CLK — C
      R   Q'
Reset —

(b) Graphic symbol

R	C	D	Q	Q'
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(b) Function table

Fig. 5-14 *D* Flip-Flop with Asynchronous Reset

Edge-triggered Flip-Flop
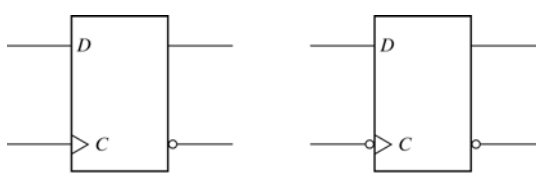
(a) Positive-edge   (a) Negative-edge

Fig. 5-11 Graphic Symbol for Edge-Triggered *D* Flip-Flop

Lo-Hi edge   **Hi-Lo edge**

J-K Flip-Flop

(a) Circuit diagram   (b) Graphic symbol

Fig. 5-12 *JK* Flip-Flop

° **Created from D flop**

° **J sets**

° **K resets**

° **J=K=1 -> invert output**

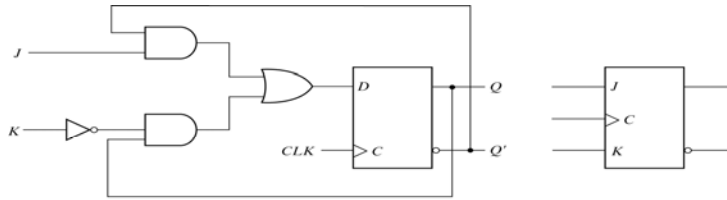| J	K	CLK	Q	Q'
0	0	↑	Q_0	Q_0'
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	TOGGLE	

Chapter 5   19

Positive Edge Triggered T Flip-Flop
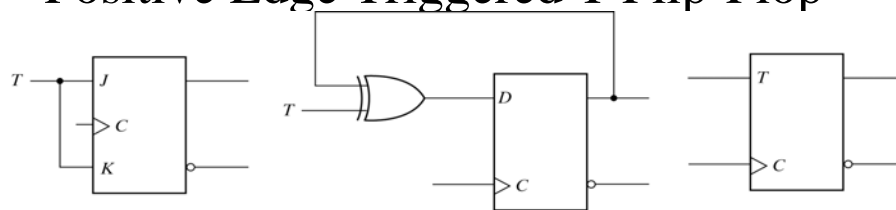
(a) From *JK* flip-flop   (b) From *D* flip-flop   (c) Graphic symbol

Fig. 5-13 T Flip-Flop

° **Created from D flop**

° **T=0 -> keep current**

° **K resets**

° **T=1 -> invert current**

T	C	Q	Q'
0	↑	Q_0	Q_0'
1	↑	TOGGLE	

Chapter 5   20

10

Characteristic Tables

- Characteristic tables Describes the operation of the flip-flop in a tabular form

JK Flip Flop

J K	Q(t+1)
0 0	Q(t)
0 1	0
1 0	1
1 1	Q'(t)

D Flip Flop

D	Q(t+1)
0	0
1	1

F Flip Flop

T	Q(t+1)
0	Q(t)
1	Q'(t)

Characteristic Equation

- For D Flip-Flop

  Q(t+1)=D

- For JK Flip-Flop

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

- For T Flip-Flop

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

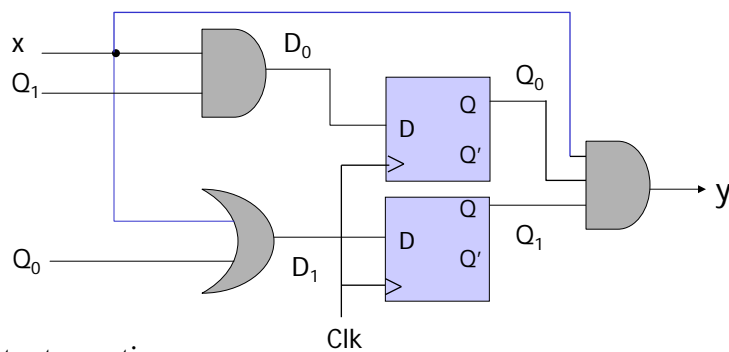# Analysis of Clocked Sequential Circuits

- State equation, state table
- State diagram: states are represented by circuits, transitions by arcs labeled I/O
- Flip-Flop Equations:

Analysis

x
Q_1
D_0
Q_0
D   Q
> Q′
Q
D   Q
Q_1
> Q′
Q_0
D_1
y
Clk

Output equation
$$y(t) = x(t)Q_1(t)Q_0(t)$$
$$Q_0(t+1) = D_0(t) = x(t)Q_1(t)$$
$$Q_1(t+1) = D_1(t) = x(t) + Q_0(t)$$
State equations

State Table

- Sequence of outputs, inputs, and flip flop states enumerated in state table
- Present state indicates current value of flip flops
- Next state indicates state after next rising clock edge
- Output is output value on current clock edge

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
0 0	00	10	0	0
0 1	10	10	0	0
1 0	00	11	0	0
1 1	10	11	0	1

State Table

$Q_1(t)\ Q_0(t)$ $Q_1(t+1)\ Q_0(t+1)$

State Table

- All possible input combinations enumerated
- All possible state combinations enumerated
- Separate columns for each output value.
- Sometimes easier to designate a symbol for each state.

Let:
$s_0 = 00$
$s_1 = 01$
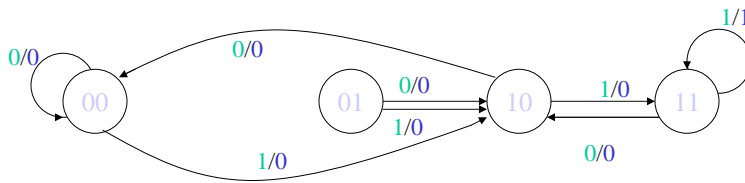$s_2 = 10$
$s_3 = 11$

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
s_0	s_0	s_2	0	0
s_1	s_2	s_2	0	0
s_2	s_0	s_3	0	0
s_3	s_2	s_3	0	1

State Diagram

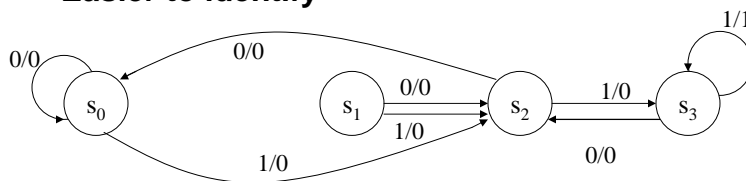| Present State	Next State		Output	
	x=0	x=1	x=0	x=1
0 0	00	10	0	0
0 1	10	10	0	0
1 0	00	11	0	0
1 1	10	11	0	1

State Diagram

° **Each state has two arrows leaving**

  ° **One for x = 0 and one for x = 1**

° **Unlimited arrows can enter a state**

° **Note use of state names in this example**

  ° **Easier to identify**
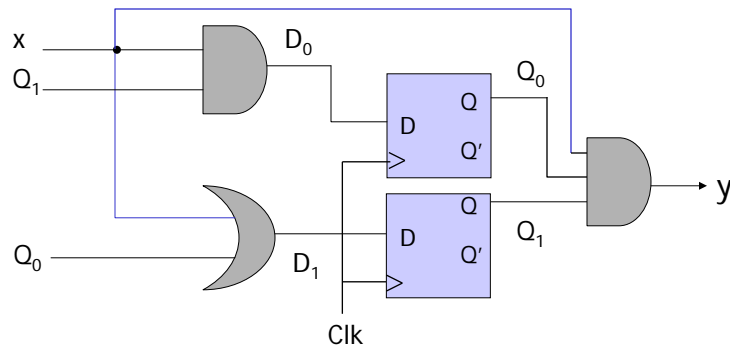
Flip-Flop Input Equations

° **Boolean expressions which indicate the input to the flip flops.**

x

Q_1

D_0

Q_0

Q

D

Q'

Q

D

Q_1

Q'

y

Q_0

D_1

Clk

$D_{Q0} = xQ_1$

$D_{Q1} = x + Q_0$

Format implies type of flop used

Analysis with D Flip-Flop
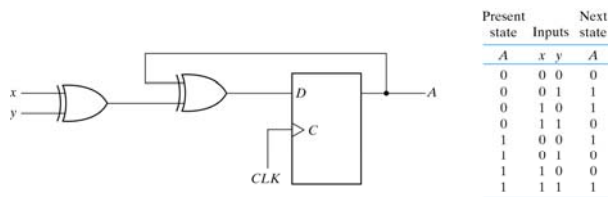
Present state	Inputs	Next state
A	x y	A
0	0 0	0
0	0 1	1
0	1 0	1
0	1 1	0
1	0 0	1
1	0 1	0
1	1 0	0
1	1 1	1

x
y
D
A
C
CLK

(a) Circuit diagram

(b) State table

00, 11

01, 10

00, 11

0

1
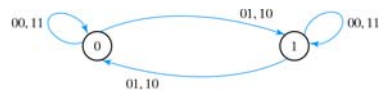
01, 10

(c) State diagram

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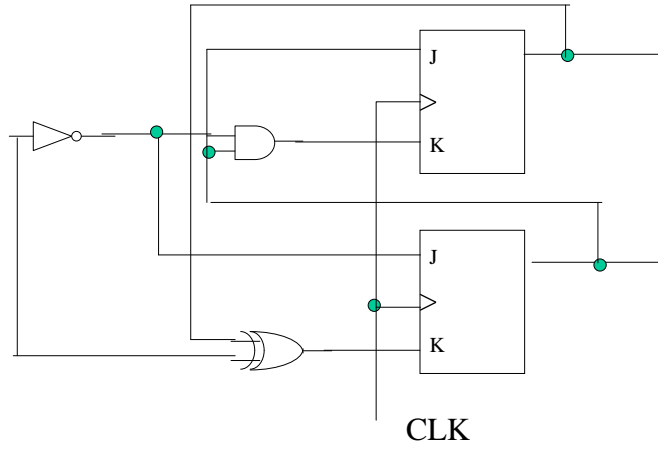
J-K Flip-Flop

CLK

J-K Flip Flop

Present A	B	Input X	Next A	B	J_A	K_A	J_B	K_B
					Flip-Flop Inputs			
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

J-K Flip-Flop
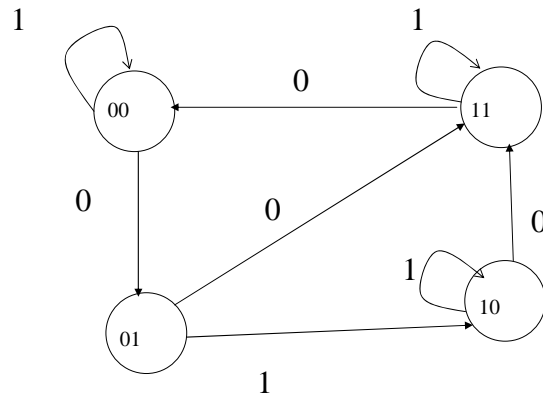
Mealy and Moore Models

- Mealy model: the output is a function of both the present state and the input.
- Moore model: The output is a function of the present state only.
- In a Moore model, the output is synchronized with the clock (since it changes only if the state changes).
- In a Mealy model the output may change if the input changes during the clock cycle.

HDL For Sequential Circuits

Behavioral Modeling

iitial	**initial**
begin	**begin**
clock=1'b0;	clock = 1'b0;
repeat(30)	#300 $finish;
#10 clock = ~clock	**end**;
end	**always**
	#10 clock=~clock;

Behavioral Modeling

- One ways to use always statement

    always @ (A **or** B **or** reset)

- What fellows will be done if any changes in A, B, or reset,

- Or, we can use

    Always @ (**posedge** clock **or negedge** reset)

Behavioral Modeling

- Blocking assignmnet

    B=A;
    C=B+1;

- A is assigned to B, and the new value is incremented

- Non-blocking assignment

    B<=A; C<=B+1;

- Incrementing old value of A, because assignment is done after all the values in the block are done

Flip-Flops and Latches

//Description od D latch
module D_Latch (Q,D,control);
   output Q;
   input D, control;
   reg Q;
   always @ (control **or** D)
   if (control) Q=D;
endmodule

//D Flip-Flop
module D_FF(Q, D, CLK);
   output Q;
   input D, CLK;
   reg Q;
   always @ (posedge CLK)
     Q=D;
endmodule

Flip-Flops and Latches

//D Flip-Flop with Asynchronous reset
module DFF(Q,D,CLK,RST);
       output Q;
       input D, CLK,RST;
       reg
       always @(**posedge** CLK or **negedge** RST);
        if(~RST) Q=1'b0;
        else Q=D;
endmodule

Only if RST=1 can posedge clock
event synchronously D->Q

Other Types of Flip-Flop

```
//F Flip-Flop (using D)
Module TFF(Q,T,CLK,RST);
Output Q;
Input T,CLK,RST;
Wire DT;
Assign DT=Q^T;
DFF TF1(Q,DT,CLK,RST);
endmodule
```

```
//JK Flip-Flop (using D)
Module JKFF(Q,T,CLK,RST);
Output Q;
Input JK,CLK,RST;
Wire JK;
Assign JK=(J&~Q)|(~k&Q);
DFF JK1(Q,JK,CLK,RST);
endmodule
```

Other Types of Flip-Flop

```
//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
allways @ (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
```

State Diagram

```
//Mealy state diagram
module Mealy_Model(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;
          else Prstate=Nxtstate;
 always @ (Prstate or x)
          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)
  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;
  endcsde
endmodule
```

Chapter 5   43

State Diagram

```
//Moore state diagram
module Mealy_Model(x,AB,CLK,RST);
 input x,CLK,RST;
 output [: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) Prstate = S0;
          else Prstate=Nxtstate;
 always @ (Prstate or x)
          case(Prstate)
                 S0: if(~x) State = S1;
                          else State = S0;
                 S0: if (x)  State = s2;
                          else State = S3;

                 S0: if (x) State = S3;
                          else State = S2;

                 S0: if(x) State = S0;
                          else State = S3;
          endcase
assign AB=State;
endmodule
```

Chapter 5   44

22

Structural Description

```
//Figure 5-20 in the text book
module TCircuit(x,y,A,B,CLK,RST);
        input x. CLK, RST;
        output y,A,B;
        wire TA,TB;
        //Flip-Flop input equation
        assign TB=x,
                TA=x & B;
        assign y = A & B;
// Instantiate 2 ff's
        T_FF BF (B,TB,CLK,RST);
        T_FF AF (A,TA,CLK,RST);
endmodule
```

Structural Description

```
//test fixture for the previous design
module testTFF;
        reg x, CLK, RST;
        wire y,A,B;
        TCircuit (x,y,A,B,CLK,RST);
        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;
endmodule
```

State Reduction and Assignment

- In this part, we study some properties of the sequential circuits in order to reduce the number of gates or flip-flops.
- If we have m flip-flops, we can have up tp 2^m states.
- Thus reducing the number of states, may result in reduction of the number of flip-flops.
- Sometimes, we care only about the output produced by a specific input sequence, while in others (counters) the states are important (considered as the output).

State Reduction and Assignment

- There are infinite number of sequences that could be applied to the circuit, producing some output.
- Two circuits (may have different states) will produce the same output for the same input are considered equivalent (from the input output point of view).
- We want to find a way to reduce the number of states and keeping the same input-output equivalence.
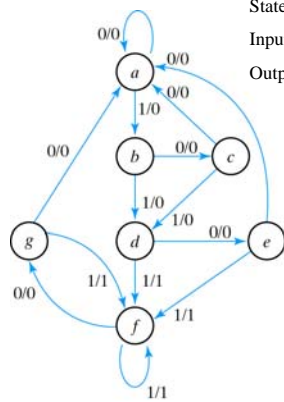
State Reduction

State	a	a	b	c	d	e	f	g	f	g
Input	0	1	0	1	0	1	1	0	0	0
Output	0	0	0	0	0	1	1	0	0	0

Fig. 5-22 State Diagram

State reduction

	next state		Output	
Present State	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	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

look for any two state that go to the same next state and have the same output for both input combination (states g and e), remove one and replace it by the other

State reduction

Present State	next state x=0	x=1	Output x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

States d and f are equivalent

State reduction

Present State	next state x=0	x=1	Output 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
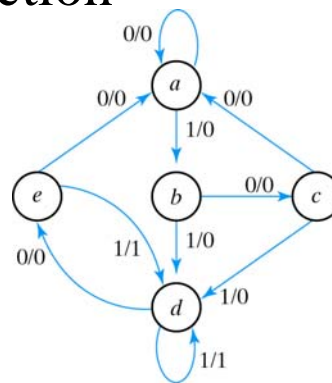
Fig. 5-23   Reduced State Diagram

State Assignment

- In this stage, you have to map states to binary numbers. You can use any mapping scheme.
  - binary
  - Gray code
  - one-hot (more flip-flops)

Design Procedure

- From the word description Derive the state diagram
- Reduce the number of states if necessary
- Assign Binary values to states.
- Obtain the binary-coded state table
- Choose the type of flip-flop
- Derive the simplified flip-flop input and output equations
- Draw the logic diagram

Design

- Design a circuit that detects three or more consecutive ones.
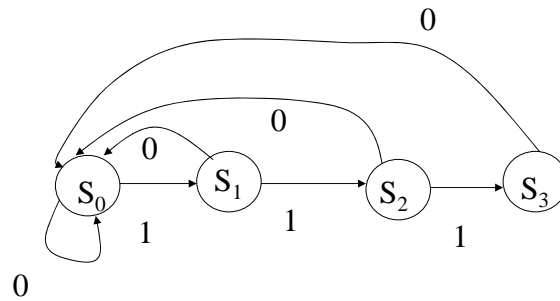
State table for sequence detector

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	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Synthesis with D Flip-Flops

- Construct the state table

$$A(t+1) = D_A(A, B, X) = \sum_{ABX}(3,5,7)$$

$$B(t+1) = D_B(A, B, x) = \sum_{ABX}(1,5,7)$$

$$y(A, B, x) = \sum_{ABx}(6,7)$$

$$D_A = Ax + Bx$$

$$D_B = Ax + B'x$$

$$y = AB$$

Synthesis with D Flip-Flops
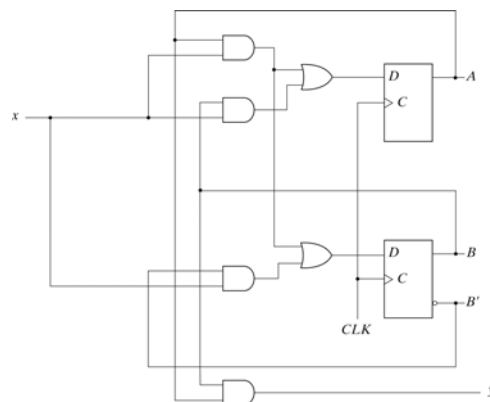
Fig. 5-26 Logic Diagram of Sequence Detector

Synthesis with J-K Flip-Flop

- With J-K flip-flop, it is not as easy as D, since the output is not the same as the previous input, we need an excitation table

Q(t)	Q(t+1)	J	K	Q(t)	Q(t+1)	T
0	0	0	X	0	0	0
0	1	1	X	0	1	1
1	0	X	1	1	0	1
1	1	X	0	1	1	1

Synthesis with J-K Flip-Flop

Present State		Input	Next State		Output	JA	KA	JB	KB
A	B	x	A	B	y	JA	KA	JB	KB
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	0	X	1	X
0	1	0	0	0	0	0	X	X	1
0	1	1	1	0	0	1	X	X	1
1	0	0	0	0	0	X	1	0	X
1	0	1	1	1	0	X	0	1	X
1	1	0	0	0	1	X	1	X	1
1	1	1	1	1	1	X	0	X	0

Synthesis with J-K Flip Flop

$J_A = Bx'$

$K_A = Bx$

$J_B = x$

$K_B = (A \oplus x)'$
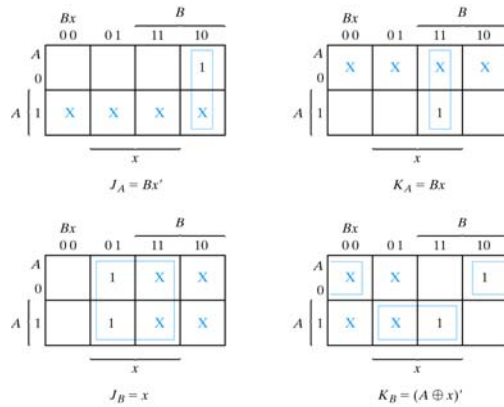
Fig. 5-27 Maps for J and K Input Equations
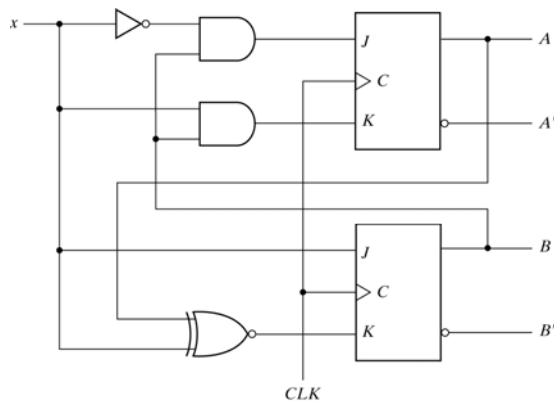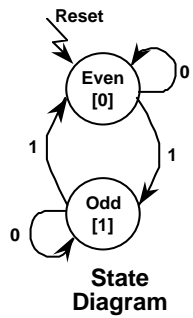
Synthesis with J-K Flip-Flop

Fig. 5-28 Logic Diagram for Sequential Circuit with JK Flip-Flops

Example: Odd Number of 1's

Assert output whenever input bit stream has odd # of 1's

Present State	Input	Next State	Output
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

State Diagram

° **Note: Present state and output are the same value**

Odd Number of 1's

Example: Odd Parity Checker

Next State/Output Functions
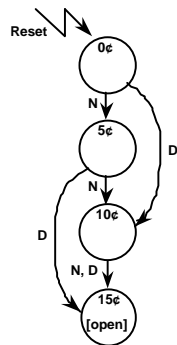
NS = PS xor PI;   OUT = PS
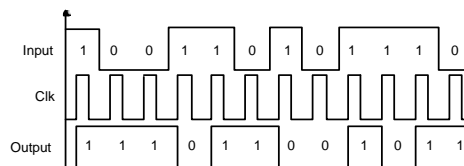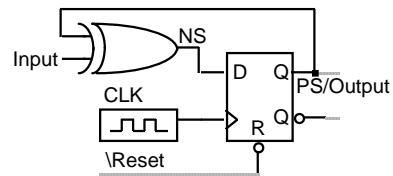
Vending Machine

❑**Deliver package of gum after 15 cents deposited**

❑**Single coin slot for dimes, nickels**

❑**No change**

❑**Design the FSM using combinational logic and flip flops**

Vending Machine

Present State	Inputs D	N	Next State	Output Open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	X	X
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	X	X
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	X	X
15¢	X	X	15¢	1
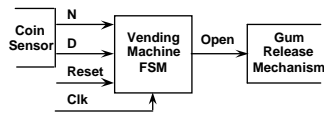
Reuse states whenever possible

Symbolic State Table

State encoding

Present State Q_1 Q_0		Inputs D N		Next State D_1 D_0		Output Open
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	X	X	X
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	0
		1	1	X	X	X
1	0	0	0	1	0	0
		0	1	1	1	0
		1	0	1	1	0
		1	1	X	X	X
1	1	0	0	1	1	1
		0	1	1	1	1
		1	0	1	1	1
		1	1	X	X	X

Vending Machine

K-map for D1   **K-map for D0**   **K-map for Open**
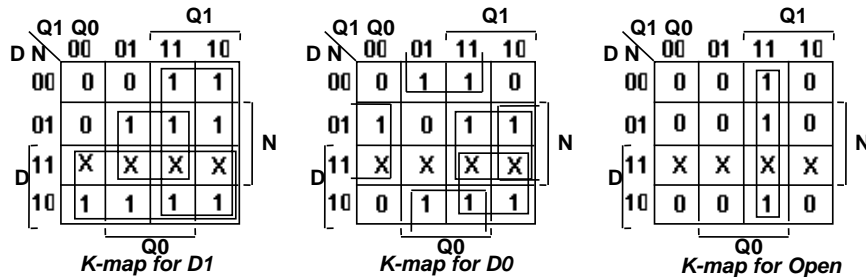
$D_1 = D + Q_1 + NQ_0$

$D_0 = NQ_1 + NQ'_0 + Q_0 N' + DQ_1$

$Q_1 Q_0$

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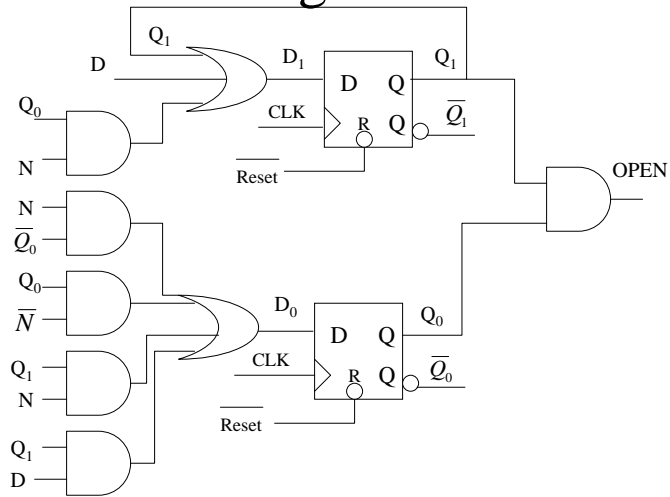
3-bit Binary Counter
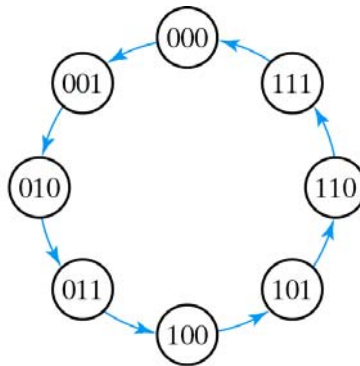
Fig. 5-29 State Diagram of 3-Bit Binary Counter

3-Bit Binary Counter

Present State	Next State	F-F 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

3-Bit Binary Counter
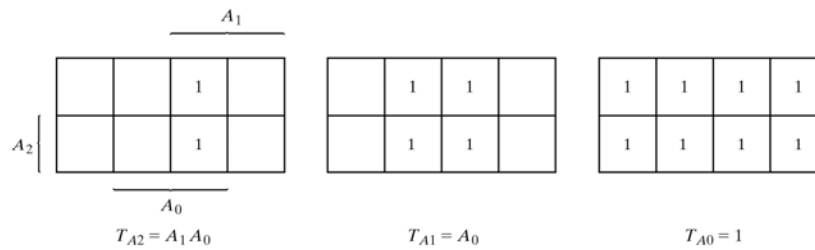
Fig. 5-30 Maps for 3-Bit Binary Counter

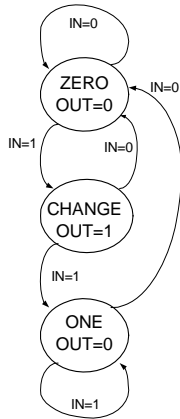$T_{A2} = A_1 A_0$ $T_{A1} = A_0$ $T_{A0} = 1$

3-Bit Binary Counter
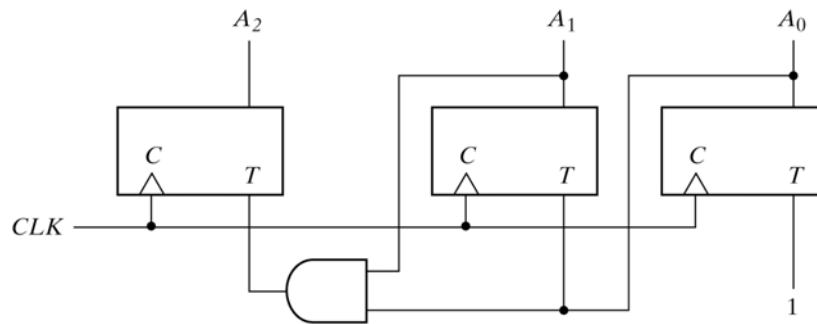
Fig. 5-31 Logic Diagram of 3-Bit Binary Counter

Example (Mealy vs. Moore)

- Design a circuit that asserts its output for one cycle for every change in the input from 0 to 1
- We will try two different approaches.
- Compare between them

Solution 1

IN=0

ZERO
OUT=0

IN=0

IN=1   IN=0

CHANGE
OUT=1

IN=1

ONE
OUT=0

IN=1

	IN	PS	NS	OUT
ZERO	0	00	00	0
	1	00	01	0
CHANGE	0	01	00	1
	1	01	11	1
ONE	0	11	00	0
	1	11	11	0

Solution 1

	IN	PS	NS	OUT
ZERO	0	00	00	0
	1	00	01	0
CHANGE	0	01	00	1
	1	01	11	1
ONE	0	11	00	0
	1	11	11	0

PS

	00	01	11	10
IN 0	0	0	0	-
1	0	1	1	-

$NS_1 = IN\, PS_0$

PS

	00	01	11	10
IN 0	0	0	0	-
1	1	1	1	-

$NS_0 = IN$

PS

	00	01	11	10
IN 0	0	1	0	-
1	0	1	0	-

$OUT = \overline{PS_1}\, PS_0$

NS₁ → FF → PS₁

OUT

NS₀ → FF → PS₀

IN

Solution 2

IN=0
OUT=0

ZERO

Let ZERO=0,
ONE=1

IN	PS	NS	OUT
0	0	0	0
0	1	0	0
1	0	1	1
1	1	1	0

IN=1
OUT=1

IN=0
OUT=0

NS = IN, OUT = IN PS'

ONE

IN=1
OUT=0

IN → NS FF PS

OUT

Comparison

IN

PS$_0$

PS$_1$

OUT

Comparison

IN

PS

OUT

Output may not be high long
enough to be of any use

Comparison

CLK

IN

OUT (solution A)

OUT (solution B)