

Chapter 5

Synchronous Sequential Logic

Chapter 5

1

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

Chapter 5

2

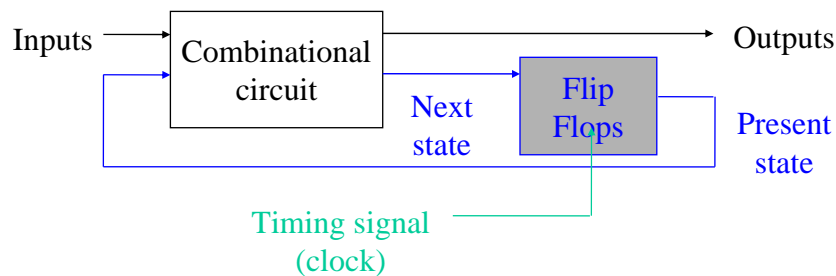
Synchronous vs. Asynchronous

- 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

Chapter 5

3

Synchronous Sequential Circuit



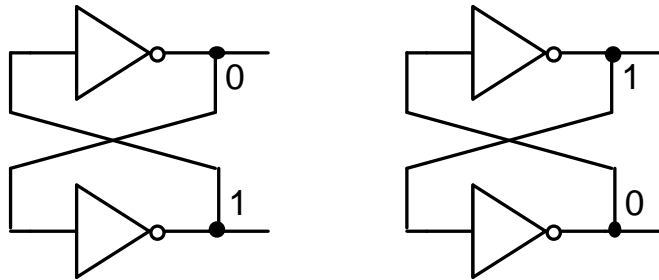
Clock

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

Clock

4

Cross Coupled Inverters



State 1

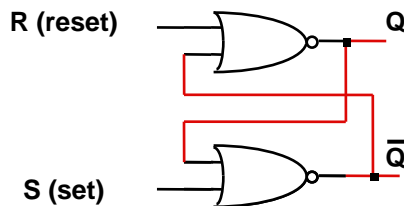
State 2

What if there is an inverter in the feedback path?

Chapter 5

5

SR Latch using NOR



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

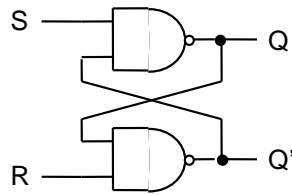
°Usually S=0 and R=0

°S=1 and R=1 generates **unpredictable results**

Chapter 5

6

SR latch using NAND



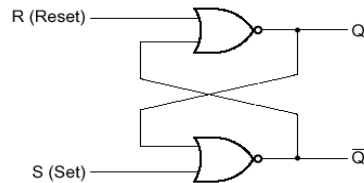
S	R	Q	Q'	
0	0	1	1	Disallowed
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	1	Store

- 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

Chapter 5

7

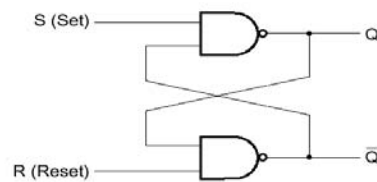
SR Latch



(a) Logic diagram

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

(b) Function table



(a) Logic diagram

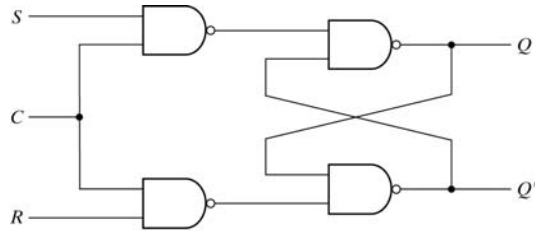
S	R	Q	\bar{Q}	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	
1	1	0	1	Reset state
0	0	1	1	Undefined

(b) Function table

Chapter 5

8

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	Indeterminate

(b) Function table

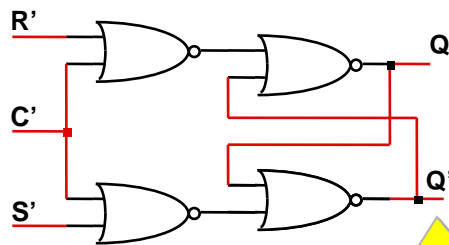
Fig. 5-5 SR Latch with Control Input

- ° 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.

Chapter 5

9

SR Latch with Control Input

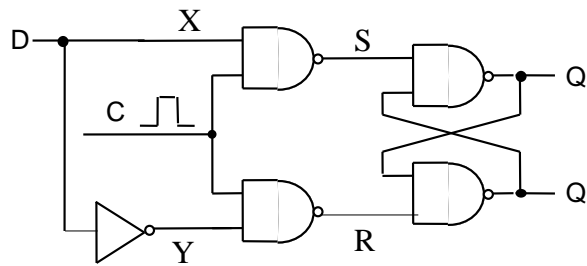


Outputs change
when C is low:
RESET and SET
Otherwise: HOLD

Chapter 5

10

D Latch



Chapter 5

11

D Latch

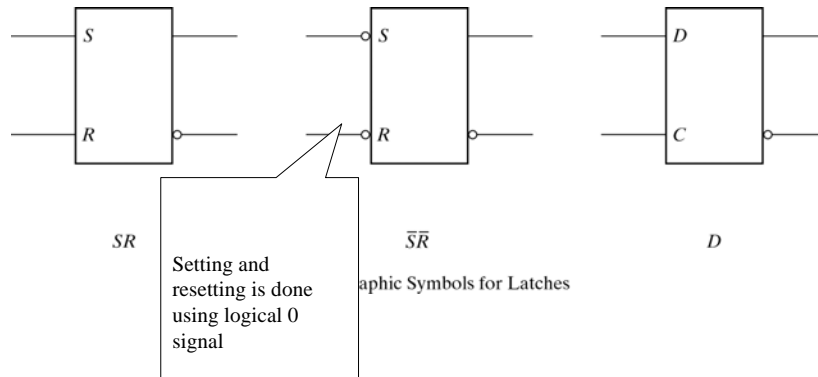
- One way to avoid the indeterminate state in the SR latch when both S and R are 1 is done by inverting S and having the inverted S as R.
- As long as $C=0$, no change. If $C=1$ D is delivered to Q and D' to Q'

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

Chapter 5

12

Symbols



Chapter 5

13

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

Chapter 5

14

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, $1 \rightarrow 0$ and $0 \rightarrow 1$.
- The first is called -ve edge, the second is positive edge.
- There are two ways to implement this, either by using master-slave or by special design of the circuit.

Chapter 5

15

Edge-Triggered Flip-Flop (Master-Slave)

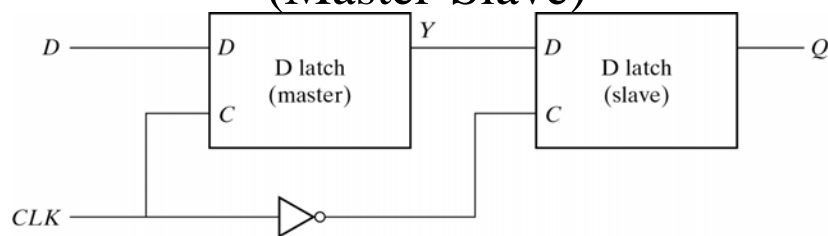


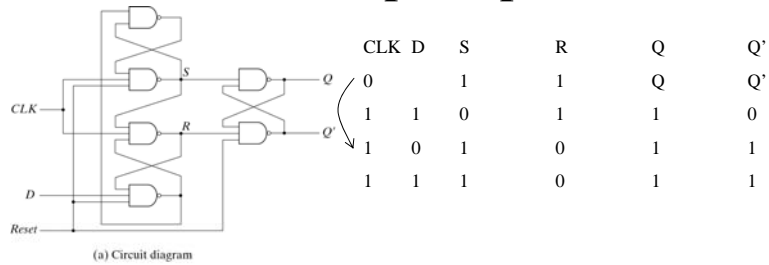
Fig. 5-9 Master-Slave *D* Flip-Flop

Any changes in the input can affect only *Y* as long as $CLK=1$; After $CLK=0$, *Y* propagates to *Q*, but the master is locked.

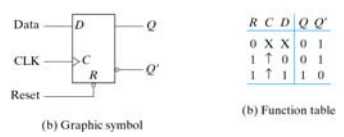
Chapter 5

16

D-Type Positive-Edge-triggered Flip-Flop



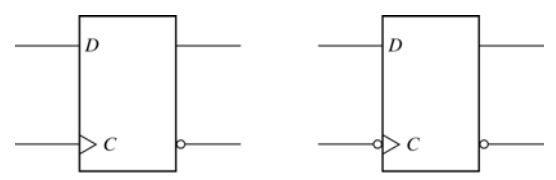
CLK	D	S	R	Q	Q'
0		1	1	Q	Q'
1	1	0	1	1	0
1	0	1	0	1	1
1	1	1	0	1	1



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

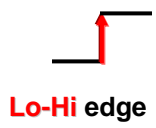
Fig. 5-14 D Flip-Flop with Asynchronous Reset

Edge-triggered Flip-Flop

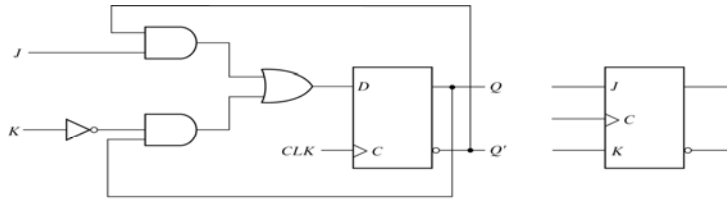


(a) Positive-edge (a) Negative-edge

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



J-K Flip-Flop



(a) Circuit diagram

(b) Graphic symbol

Fig. 5-12 JK Flip-Flop

°Created from D flop

°J sets

°K resets

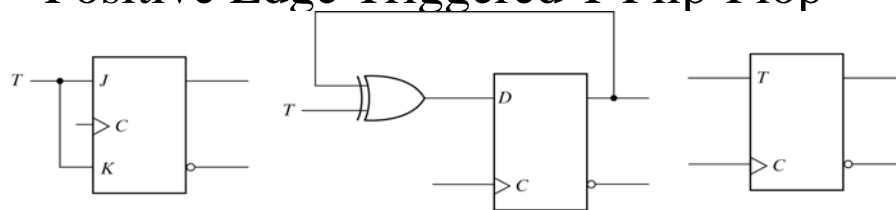
°J=K=1 -> invert output

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

Characteristic Tables

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

JK Flip Flop			D Flip Flop		F Flip Flop	
J	K	Q(t+1)	D	Q(t+1)	T	Q(t+1)
0	0	Q(t)	0	0	0	Q(t)
0	1	0	1	1	1	Q'(t)
1	0	1				
1	1	Q'(t)				

Chapter 5

21

Characteristic Equation

- For D Flip-Flop

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

- For JK Flip-Flop

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

- For T Flip-Flop

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

Chapter 5

22

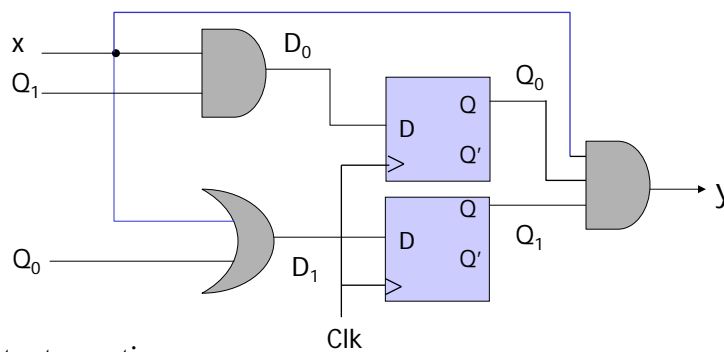
Analysis of Clocked Sequential Circuits

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

Chapter 5

23

Analysis



Output equation

$$y(t) = x(t)Q_1(t)Q_0(t)$$

$$Q_0(t+1) = D_0(t) = x(t)Q_1(t)$$

State equations

$$Q_1(t+1) = D_1(t) = x(t) + Q_0(t)$$

Chapter 5

24

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

State Table

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

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

Chapter 5
25

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:

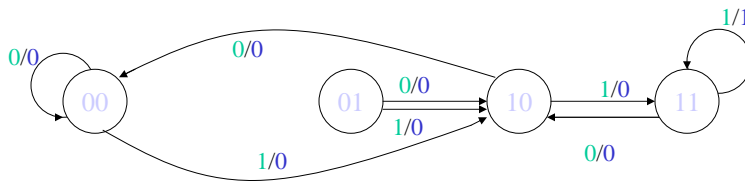
Present State	Next State		Output	
	x=0	x=1	x=0	x=1
$s_0 = 00$	s_0	s_2	0	0
$s_1 = 01$	s_1	s_2	0	0
$s_2 = 10$	s_2	s_3	0	0
$s_3 = 11$	s_3	s_2	0	1

Chapter 5

26

State Diagram

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

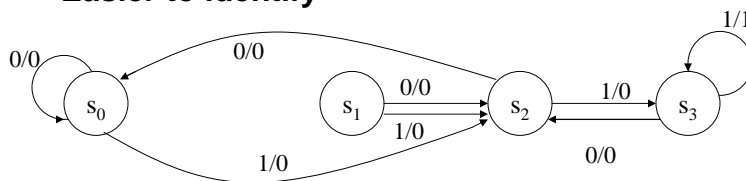


Chapter 5

27

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

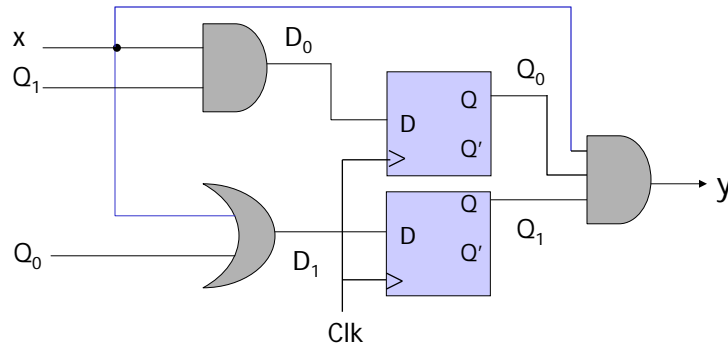


Chapter 5

28

Flip-Flop Input Equations

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



$$D_{Q0} = xQ_1$$

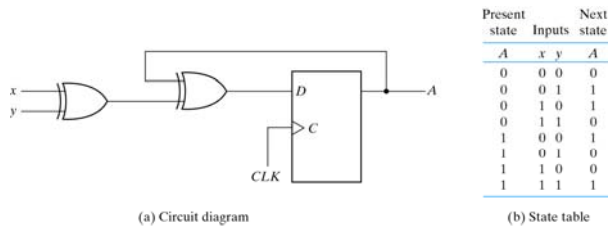
$$D_{Q1} = x + Q_0$$

Format implies type of flop used

Chapter 5

29

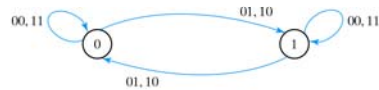
Analysis with D Flip-Flop



(a) Circuit 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



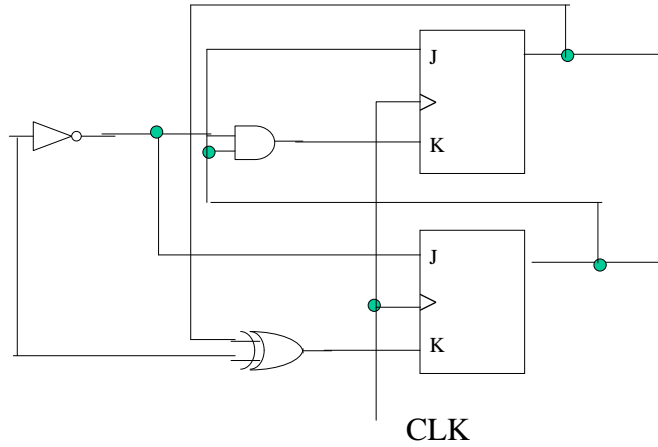
(c) State diagram

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

Chapter 5

30

J-K Flip-Flop



Chapter 5

31

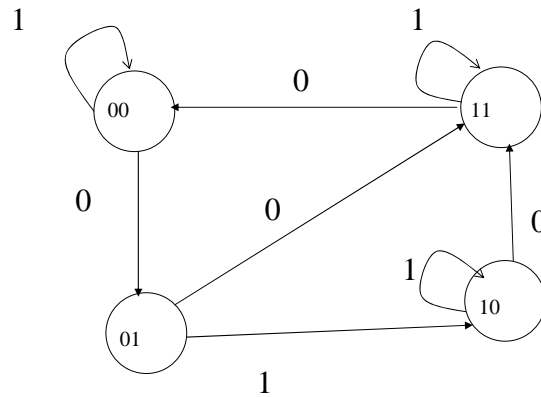
J-K Flip Flop

Present		Input	Next		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

Chapter 5

32

J-K Flip-Flop



Chapter 5

33

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.

Chapter 5

34

HDL For Sequential Circuits

Chapter 5

35

Behavioral Modeling

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

Chapter 5

36

Behavioral Modeling

- One way to use always statement
`always @ (A or B or reset)`
- What follows will be done if any changes in A, B, or reset,
- Or, we can use
`Always @ (posedge clock or negedge reset)`

Chapter 5

37

Behavioral Modeling

- Blocking assignment
`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

Chapter 5

38

Flip-Flops and Latches

```
//Description of 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
```

Chapter 5

39

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

Chapter 5

40

Other Types of Flip-Flop

```
D.  
//D Flip-Flop (using D) //JK Flip-Flop (using D)  
Module TFF(Q,T,CLK,RST); Module JKFF(Q,T,CLK,RST);  
Output Q; Output Q;  
Input T,CLK,RST; Input JK,CLK,RST;  
Wire DT; Wire JK;  
Assign DT=Q^T; Assign JK=(J&~Q) | (~k&Q);  
DFF TF1(Q,DT,CLK,RST); DFF JK1(Q,JK,CLK,RST);  
endmodule 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  
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
```

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
endmodule
```

```
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

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
```

Chapter 5

45

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
```

Chapter 5

46


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 to 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).

Chapter 5

47

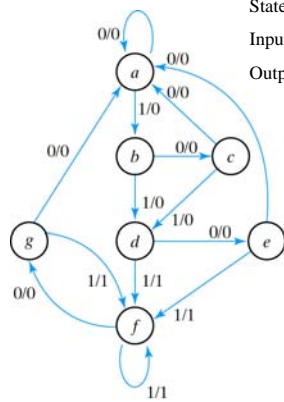
State Reduction and Assignment

- 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.

Chapter 5

48

State Reduction

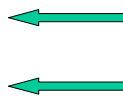


State	a	a	b	c	d	e	f	f	g	a
Input	0	1	0	1	0	1	1	0	0	0
Output	0	0	0	0	0	1	1	0	0	0

Fig. 5-22 State Diagram

State reduction

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	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		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	f	0	1
e	a	f	0	1
f	e	f	0	1

G is removed and every occurrence of g is replaced by g in the remaining of the table



States d and f are equivalent

f is removed and replaced by e

State reduction

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

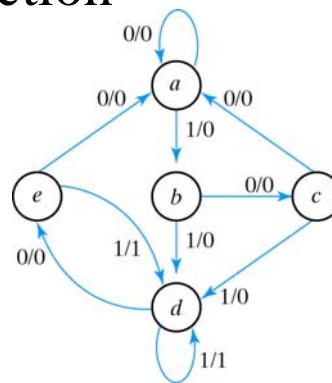


Fig. 5-23 Reduced State Diagram

State Assignment

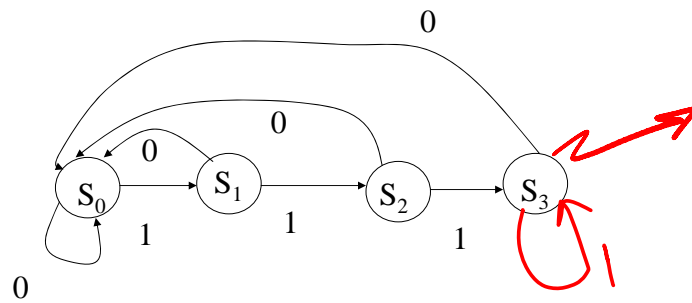
- In this stage, you have to map states to binary numbers. You can use any mapping scheme.
 - binary
 - Gray code
 - one-hot (more flip-flops)

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.

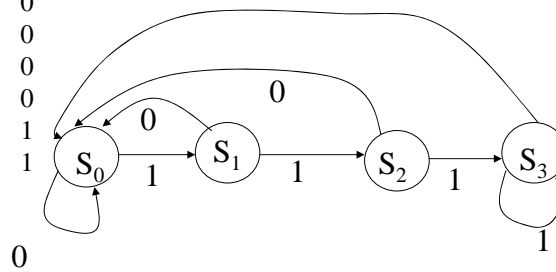


Chapter 5

55

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



Chapter 5

56

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

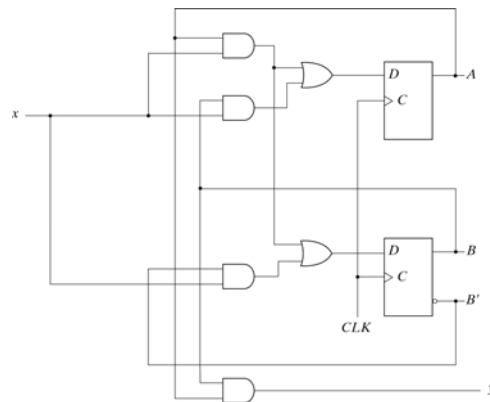


Fig. 5-26 Logic Diagram of Sequence Detector

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	0

Chapter 5

59

Synthesis with J-K Flip-Flop

Present State		Input	Next State		Output	JA	KA	JB	KB	T_A	T_B
A	B	x	A	B	y						
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		

0000 → 000
 T_A
 T_B

Chapter 5

60

Synthesis with J-K Flip Flop

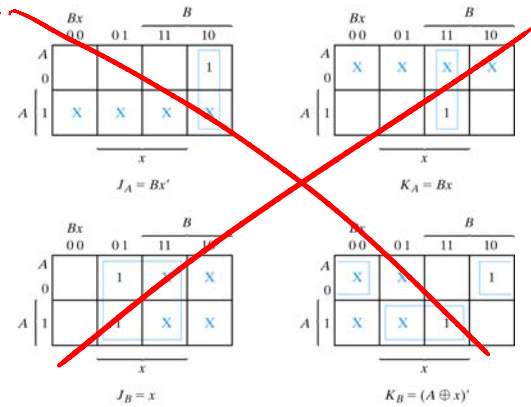


Fig. 5-27 Maps for J and K Input Equations

Synthesis with J-K Flip-Flop

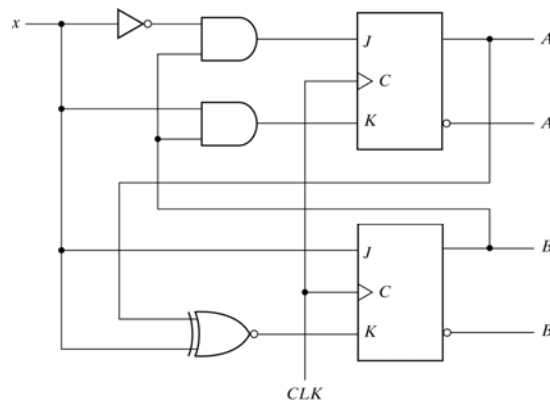
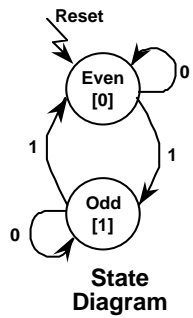


Fig. 5-28 Logic Diagram for Sequential Circuit with JK Flip-Flops

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

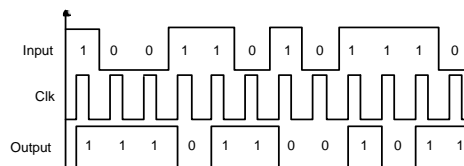
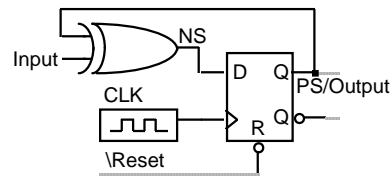
- **Note: Present state and output are the same value**

Odd Number of 1's

Example: Odd Parity Checker

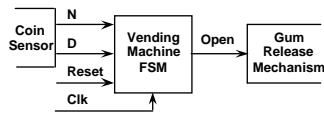
Next State/Output Functions

$$NS = PS \text{ xor } PI; \quad OUT = PS$$



Vending Machine

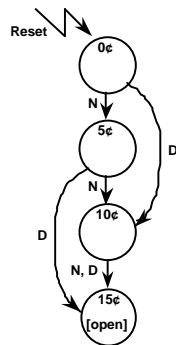
- ❑ Deliver package of gum after 15 cents deposited
- ❑ Single coin slot for dimes, nickels
- ❑ No change
- ❑ Design the FSM using combinational logic and flip flops



Chapter 5

65

Vending Machine



**Reuse states
whenever possible**

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

Symbolic State Table

Chapter 5

66

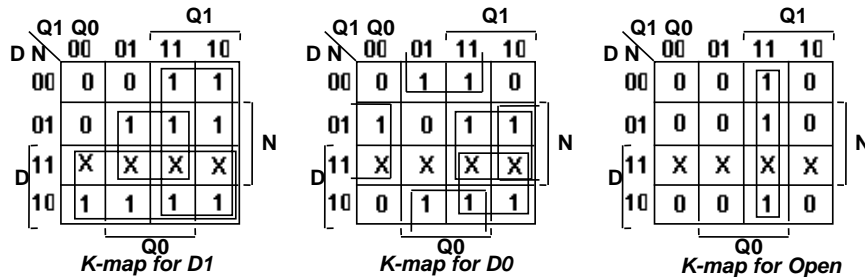
State encoding

Present State		Inputs		Next State		Output
Q_1	Q_0	D	N	D_1	D_0	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

Chapter 5

67

Vending Machine



$$D_1 = D + Q_1 + NQ_0$$

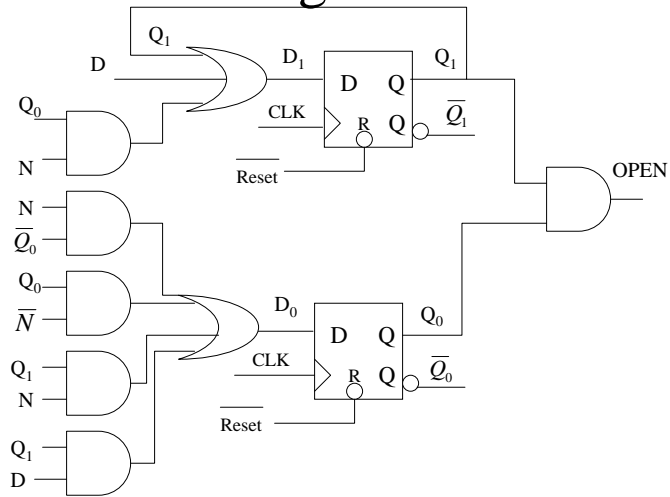
$$D_0 = NQ_1 + NQ_0' + Q_0N' + DQ_1$$

$$Q_1Q_0$$

Chapter 5

68

Vending Machine



Chapter 5

69

3-bit Binary Counter

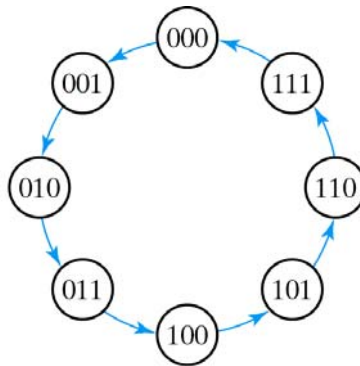


Fig. 5-29 State Diagram of 3-Bit Binary Counter

Chapter 5

70

3-Bit Binary Counter

Present State	Next State	T-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
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 0 1
1 1 1	0 0 0	1 1 1

Chapter 5

71

3-Bit Binary Counter

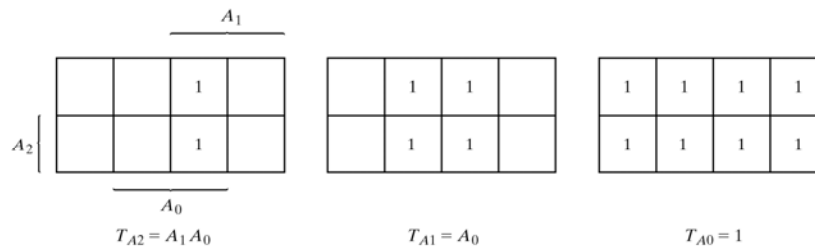
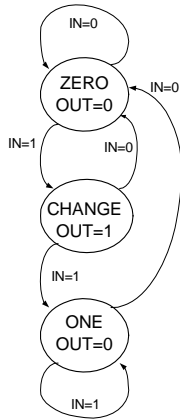


Fig. 5-30 Maps for 3-Bit Binary Counter

Chapter 5

72

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

Chapter 5

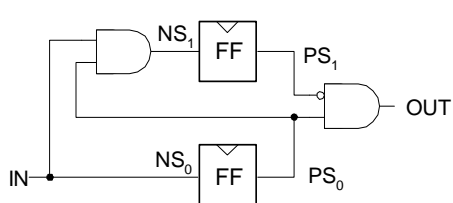
75

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

IN	PS				
	00	01	11	10	
0	0	0	0	-	$NS_1 = IN \text{ PS}_0$
1	0	1	1	-	

IN	PS				
	00	01	11	10	
0	0	0	0	-	$NS_0 = IN$
1	1	1	1	-	

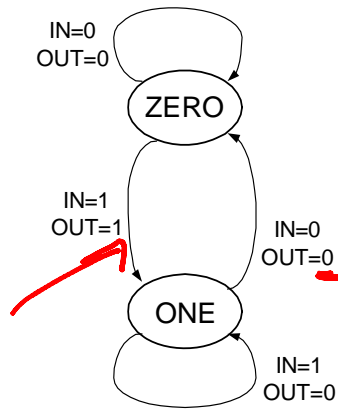


IN	PS				
	00	01	11	10	
0	0	1	0	-	$OUT = \overline{PS_1} \text{ PS}_0$
1	0	1	0	-	

Chapter 5

76

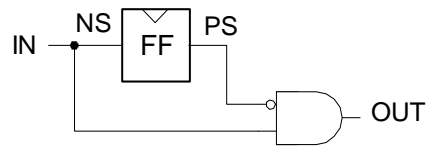
Solution 2



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

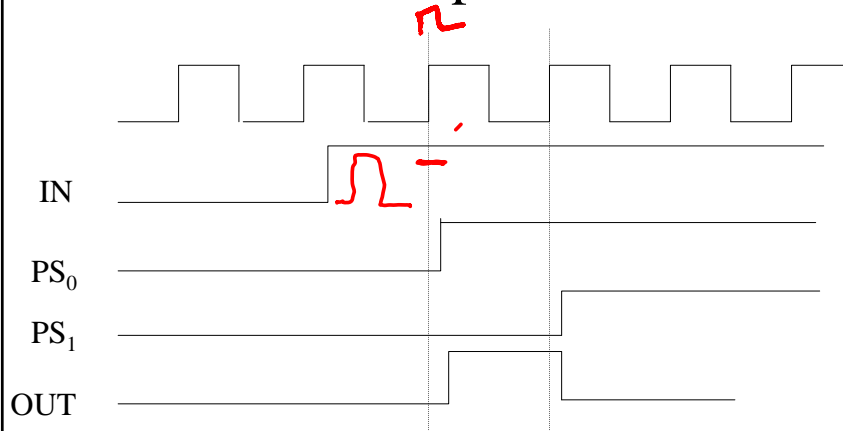
NS = IN, OUT = IN PS'



Chapter 5

77

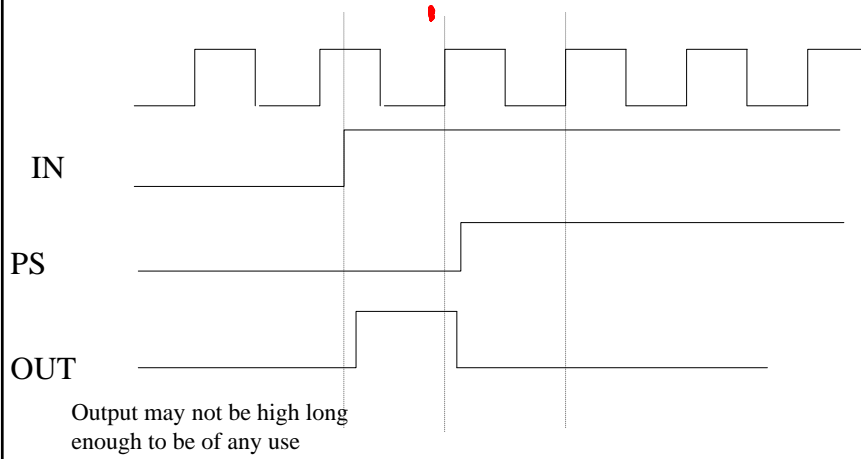
Comparison



Chapter 5

78

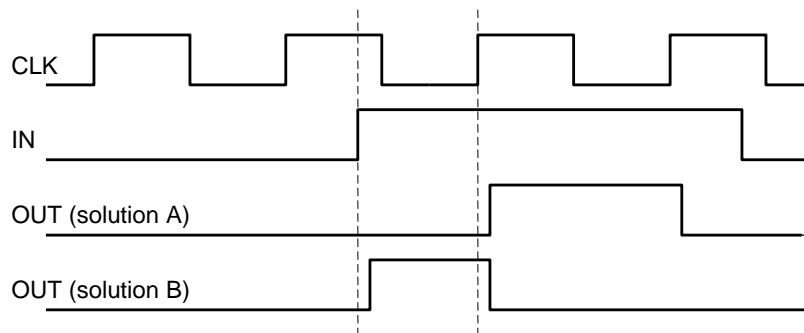
Comparison



Chapter 5

79

Comparison



Chapter 5

80