King Fahd University of Petroleum & Minerals
Computer Engineering Dept

COE 202 – Fundamentals of Computer Engineering

Term 081

Dr. Ashraf S. Hasan Mahmoud

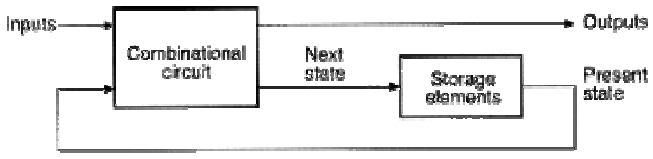
Rm 22-148-3

Ext. 1724

Email: ashraf@kfupm.edu.sa

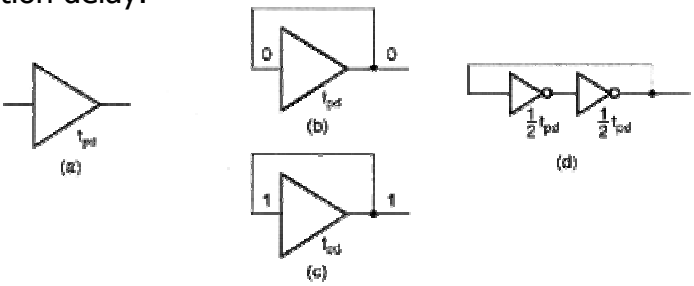
Sequential Circuit

- Definition



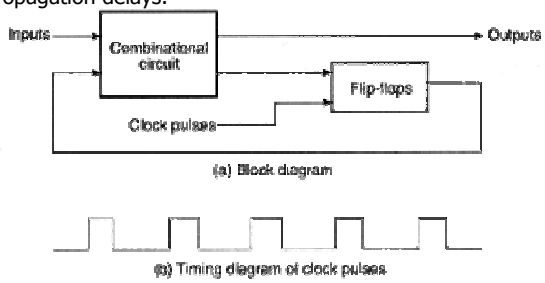
Storage Elements

- Storage can be constructed logic with delay, such as a buffer by connecting the output to the input. The signal must not undergo inversion in the loop or the system will be unstable or *astable*
- Positive Feedback – stability issues and dependence on propagation delay.



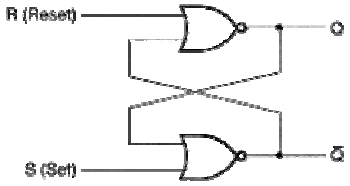
Synchronous Circuits

- Asynchronous: input changes at any time may result in the any of the outputs or internally stored information (called state) to change.
  - Such circuits are difficult to design because of dependence on propagation delays and their interaction with timing of input changes.
- Synchronous: sequential makes use of clock signals so that the storage elements (and outputs) only change at discrete instants of time in relation to the clock signal.
  - Clocked synchronous circuits provide some degree of independence on timing variations related to gate propagation delays.



SR Latch

- Definition: A storage element that maintains binary state indefinitely (as long as power is applied), until directed by an input signal to switch to its other state
- Has two inputs: S and R, and two outputs: Q and Q'
- Notes:
  - S=1, R=0 ➔ set state (i.e. Q = 1, and Q' = 0)
  - S=0, R=1 ➔ reset state (i.e. Q = 0, and Q' = 1)
  - S=0, R=0 ➔ keep state (i.e. new Q = old Q, new Q' = old Q')
  - S=1, R=1 ➔ undefined



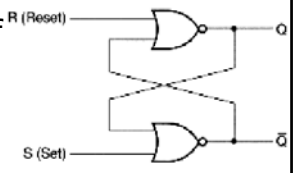
(a) Logic diagram

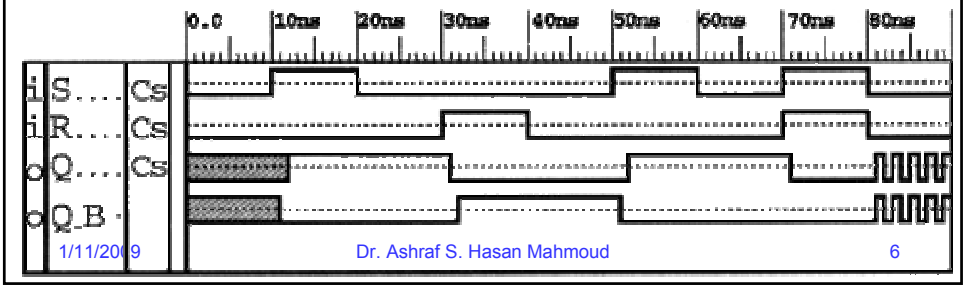
S	R	Q	$\bar{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

(b) Function table

SR Latch – Timing Diagram and Stability

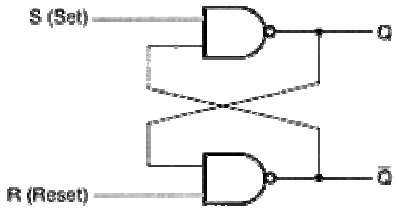
- Note: the outputs are NOW a function of time as well as the inputs
- Stability issue: When SR latch in the undefined state 00 and then both S and R are simultaneously changed to 00 – oscillations appear!!
  - What is the period of the oscillations?





SR Latch – NAND Gates Implementation

- Very similar to the NOR gates implementation
- Notes:
  - S=0, R=1 ➔ set state (i.e. Q = 1, and Q' = 0)
  - S=1, R=0 ➔ reset state (i.e. Q = 0, and Q' = 1)
  - S=1, R=1 ➔ keep state (i.e. new Q = old Q, new Q' = old Q')
  - S=0, R=0 ➔ undefined



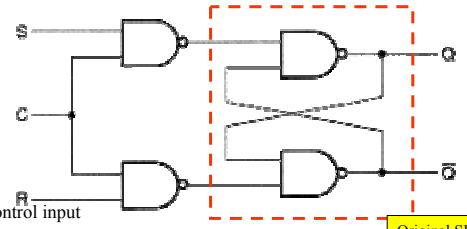
(a) Logic diagram

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

(b) Function table

SR Latch with Gating

- The inputs S and R are not always visible to the latch
  - Depending on the value of C
- Control signal C
- When enabled (i.e. C = 1):
  - You can write a 0 to Q by S=0, R=1
  - You can write a 1 to Q by S=1, R=0
  - You can keep the state by S=0, R=0
- It is still possible to drive the latch into the undefined state!!



SR latch with control input

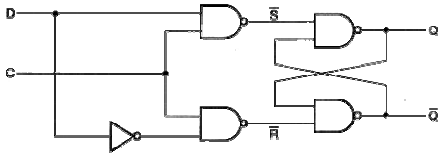
(a) Logic diagram

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

(b) Function table

Gated D-Latch

- With this controlled input D-Latch it is NOT possible to drive it into the invalid state
  - Both inputs CAN NOT be active at the same time
- When enabled (i.e. C = 1):
  - You can write a 0 to Q by D=0
  - You can write a 1 to Q by D=1



(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

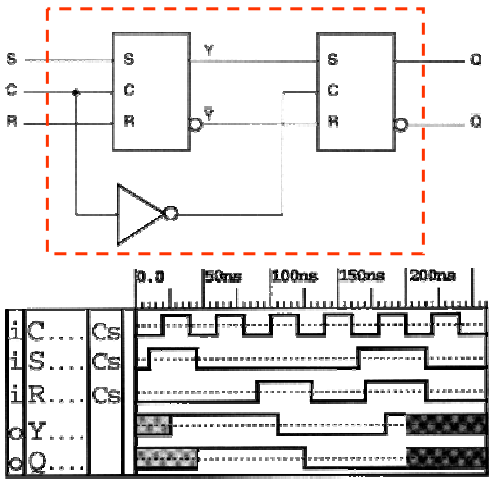
(b) Function table

Flip-Flops

- Problems with previous latches:
  - When C is enabled – the latch is in *transparent* mode. i.e. changes in S and R will be reflected in changes to its state – so it is still possible to drive the SR latch into the undefined state and then to instability.
- Solutions:
  - Master-Slave flip-flop
  - Edge-triggered flip-flop

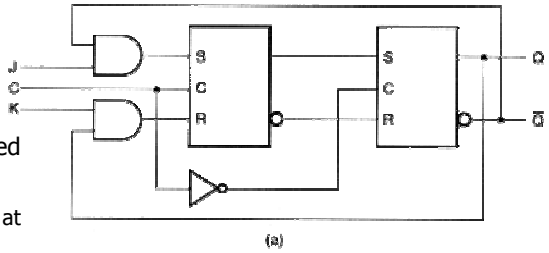
Master-Slave Flip-Flop

- Master-Slave FF: two SR latches connected as shown in figure.
- How is the transparency problem solved?
- Is the undefined state problem solved?
- Pulse-triggered flip-flop:
  - It can respond to input during the clock pulse
- You should be able to reproduce the timing diagram



Master-Slave JK Flip-Flop

- How is the transparency problem solved?
- Is the undefined state problem solved?
- Yes, it is impossible to drive the latches into the undefined state
  - The SR inputs to the first latch can never be both 1s at the same time.
- Note this flip-flop can
  - Keep the state (J=0, K=0),
  - Go into reset state (J=0, K=1),
  - Go into set state (J=1, K=0), and
  - Invert the state (J=1, K=1)



(a)

J	K	Next State of Q	
0	0	Q	Keep state
0	1	0	Reset state
1	0	1	Set state
1	1	$\bar{Q}$	Invert state

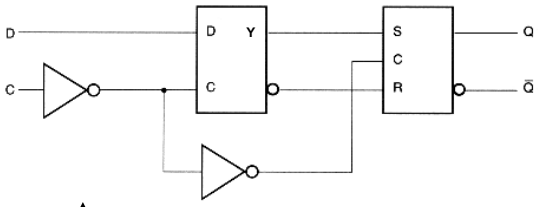
(b)

Functional Table of JK Flip-Flop

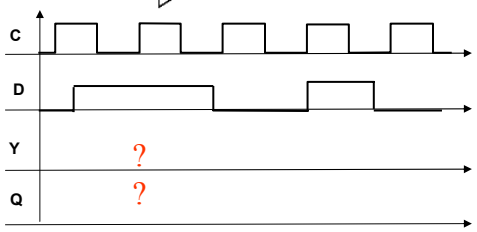
Edge-Triggered Flip-Flop

- This flip-flop responds only to the rising (+ve) edge of the clock or to the falling (-ve) edge of the clock signal.

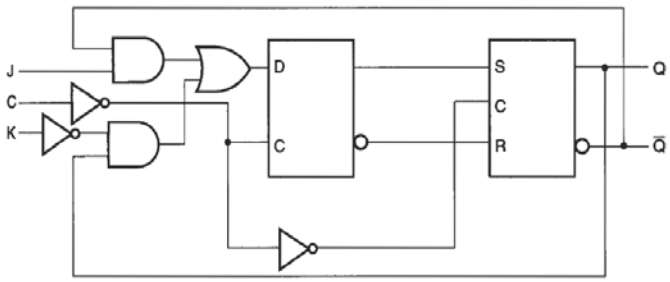
+ve edge triggered D Flip-Flop



- Can you generate the timing diagram for a given input D signal?



Positive Edge Triggered JK Flip-Flop



Timing Issues - Again

- Propagation Delay: times it takes for an input to appear at the output is called the propagation delay.
  - t_{PHL}: time it takes for an input to cause the output to change from logic-level-high to logic-level-low.
  - t_{PLH}: delay associated when an inpu change causes the output to chang from logic-level-low to logic-level-high.
- The overall propagation delay is average of t_{PHL} and t_{PLH}

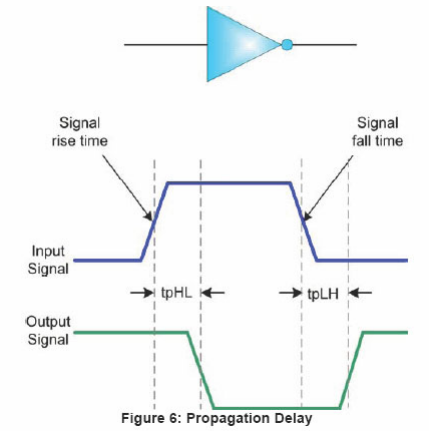
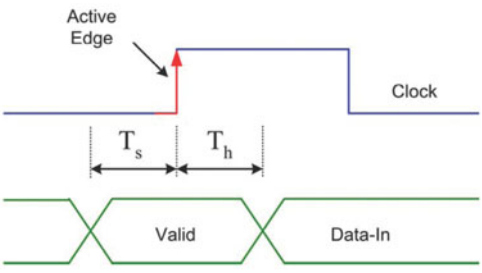


Figure 6: Propagation Delay

Timing Issues - Again

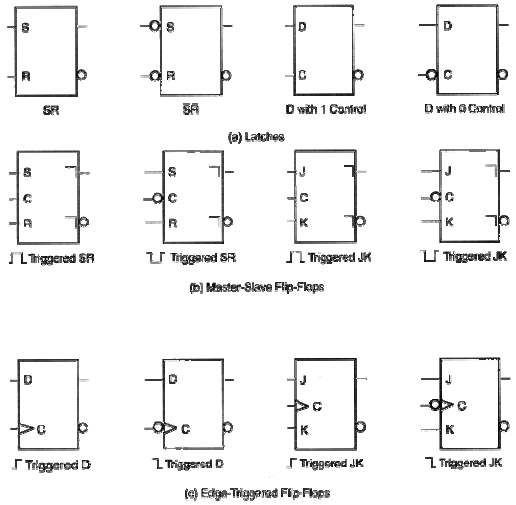
- Setup Time (Ts): a constant duration for which the inputs must be held prior to the arrival of the clock transition
- Hold Time (Th): refers to the duration for which the inputs must not change after the arrival of the transition



If the setup and hold times are violated, a gate may produce an unknown logic signal at its output. This condition is called as *meta-stability*.

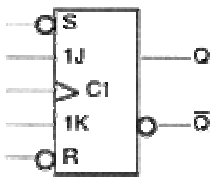
Standard Graphics Symbols

- (a) Latch Symbols
- (b) Master-Slave Flip-Flops
- (c) Edge-Triggered Flip-Flops



Asynchronous Inputs

- Special inputs for setting or resetting them asynchronously
  - Independent of the clock input
- The *direct set* and *direct reset* signals are called *preset* and *clear*, respectively
- Examining the function table:
  - When S=0, R=1, the FF is set regardless of the clock and the JK inputs
  - When S=1, R=0, the FF is reset regardless of the clock and the JK inputs
  - For the JK to operate normally, S and R should be 1 and 1.



(a) Graphic symbols

S	R	C	J	K	Q	Q̄
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

Flip-Flop Characteristic Tables

Table 6-7

(a) *JK* Flip-Flop

J	K	Q(t+1)	Operation
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q′(t)	Complement

(b) *SR* Flip-Flop

S	R	Q(t+1)	Operation
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined

(c) *D* Flip-Flop

D	Q(t+1)	Operation
0	0	Reset
1	1	Set

(d) *T* Flip-Flop

T	Q(t+1)	Operation
0	Q(t)	No change
1	Q′(t)	Complement

JK Flip-Flop Characteristic Equation

- Using table on previous slide, one can write:

KQ(t) / J	00	01	11	10
0	0	1	0	0
1	1	1	0	1

$$Q(t+1) = J\overline{Q(t)} + \overline{K}Q(t)$$

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

SR Flip-Flop Characteristic Equation

- Using table on slide 19, one can write:

RQ(t) / S	00	01	11	10
0	0	1	0	0
1	1	1	X	X

S	R	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

$$Q(t+1) = S + \overline{R}Q(t)$$

D Flip-Flop Characteristic Equation

- Using table slide 19, one can write:

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

D	Q(t)	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

T Flip-Flop Characteristic Equation

- Using table on slide 19, one can write:

$$Q(t+1) = T \oplus Q(t)$$

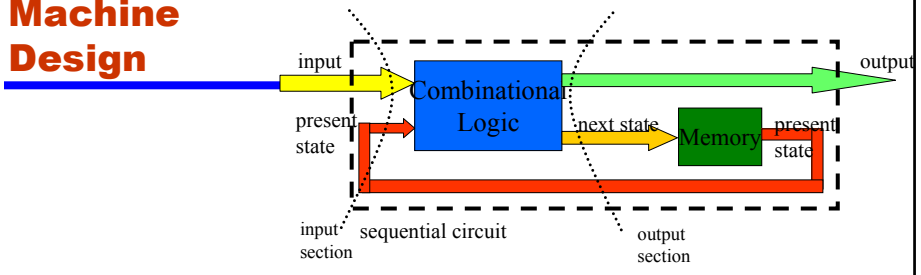
T	Q(t)	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

Flip-Flop Excitation Tables

Table 6-7

(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	

State Machine Design



Present State		Input	Output	Next State		Flip-Flop Inputs			
A	B	X	Y	A	B	J_A	K_A	J_B	K_B
0	0	0	1	0	1	0	X	1	X
0	0	1	0	0	0	0	X	0	X
0	1	0	1	1	1	1	X	X	0
0	1	1	1	1	0	1	X	X	1
1	0	0	1	1	1	X	0	1	X
1	0	1	0	1	0	X	0	0	X
1	1	0	0	0	0	X	1	X	1
1	1	1	1	1	1	X	0	X	0

input section

output section – derived from problem specification or state diagram

derived from excitation table of the particular flip-flop we use as memory

Example

- **Problem**:

a) Write characteristic equations for each type of flip-flops, using the information in Table 6-7. A characteristic equation gives the function Q(t+1) in terms of Q(t) and the input variables to the flip-flop.

b) Use the characteristic equation for the *JK* flip-flop to find equations A(t+1) and B(t+1) from the flip-flop input equations corresponding to Table shown on next slide.

- **Solution**:

a) Refer to previous slides for the development of characteristic equations

Example – cont'd

- **Solution (cont'd)**:

b)

- The columns J_A, K_A (for flip flop A) and J_B, K_B (for flip flop B) are obtained with the aid of the excitation table
- To obtain the equations for J_A, K_A, J_B, and K_B we do K-maps in terms of the inputs

Table

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	X	1	X
0	0	1	0	0	0	X	0	X
0	1	0	1	1	1	X	X	0
0	1	1	1	0	1	X	X	1
1	0	0	1	1	X	0	1	X
1	0	1	1	0	X	0	0	X
1	1	0	0	0	X	1	X	1
1	1	1	1	1	X	0	X	0

This is input

This is derived from problem specification or state diagram

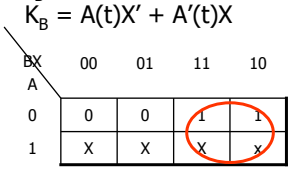
This is derived from excitation table of the particular flip-flop

Example – cont'd

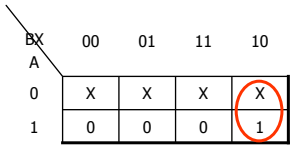
- **Solution (cont'd)**:

b) From the K-maps

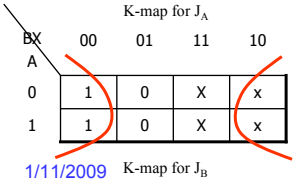
$J_A = B(t)$,
$K_A = B(t)X'$,
while
$J_B = X'$,
$K_B = A(t)X' + A'(t)X$



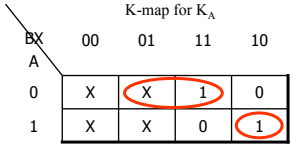
K-map for J_A



K-map for K_A



K-map for J_B



K-map for K_B

Example – cont'd

- **Solution (cont'd)**:

b) Finally, Using the characteristic equation for the A *JK* flip-flop:

$A(t+1) = J_A A(t)' + K_A' A(t)$ ➔

$A(t+1) = B(t)A(t)' + (B(t)X')'A(t)$

$= B(t)A(t)' + B(t)'A(t) + XA(t)$

- Same for the B *JK* flip-flop:

$B(t+1) = J_B B(t)' + K_B' B(t)$ ➔

$B(t+1) = X'B(t)' + (A(t)X' + A'(t)X)'B(t)$

$B(t+1) = X'B(t)' + A(t)B(t)X + A'(t)B(t)X'$

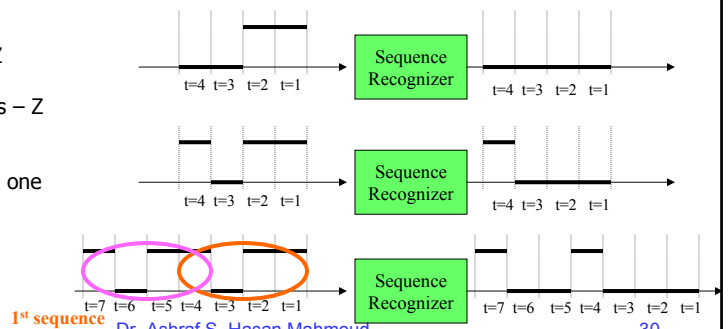
Sequence Recognizer

- **Problem**: Design a circuit to recognize the occurrence of the bits 1101 (input from left to right) on an input line X by making an output signal Z equal to 1; Otherwise Z is equal to 0
- **Solution**:

Sequential circuit with one input X and one output Z

- Examples of operation:

1. No sequence – Z remains zero
2. sequence occurs – Z is one
3. Two overlapping sequences – Z is one twice!



Sequence Recognizer – State Diagram

- **Solution (cont'd)**:

You always start from an initial state ➔ State S_0

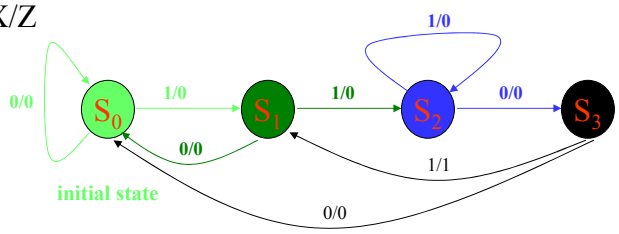
To remember first '1' of sequence ➔ State S_1

To remember two consecutive 1s of sequence ➔ State S_2

To remember '110' sequence ➔ State S_3

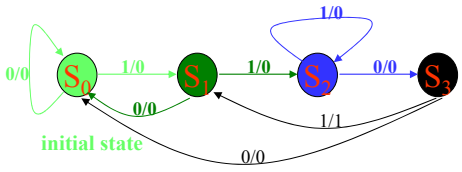
Note an arrival of S_1 while in state S_3 should make the output Z = 1, and move to state B "to remember this '1' which could be the first digit of another 1101 sequence

Arc label: X/Z



Sequence Recognizer – State Table

- **Solution (cont'd)**:



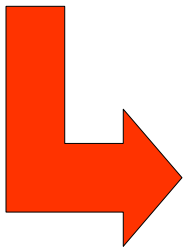
Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_2	0	0
S_3	S_0	S_1	0	1

Sequence Recognizer – State Table (2)

- **Solution (cont'd)**:

Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_2	0	0
S_3	S_0	S_1	0	1

State Code Assignment (Grey Coding):
S_0 ➔ 00
S_1 ➔ 01
S_2 ➔ 11
S_3 ➔ 10



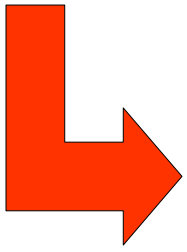
Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

Sequence Recognizer – State Table (3)

- **Solution (cont'd)**:

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

-Another way of writing the state table
- Four states ➔ we need two flip-flops A & B (in general if number of states is n, then we require $\log_2 n$ flip-flops)



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

Sequence Recognizer – Design Using *D* Flip-Flops

- **Solution (cont'd)**:

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

(c) *D* Flip-Flop		
Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

-The characteristic equation for the *D* flip-flop is Q(t+1) = D
➔The D input is the same as the desired next state

Present State		Input	Next State		Output	*D* Flip-Flops Input	
A	B	X	A	B	Z	D_A	D_B
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	0	0	0
0	1	1	1	1	0	1	1
1	0	0	0	0	0	0	0
1	0	1	0	1	1	0	1
1	1	0	1	0	0	1	0
1	1	1	1	1	0	1	1

Sequence Recognizer – Design Using *D* Flip-Flops (2)

- **Solution (cont'd)**:

Present State		Input	Next State		Output	*D* Flip-Flops Input	
A	B	X	A	B	Z	D_A	D_B
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	0	0	0
0	1	1	1	1	0	1	1
1	0	0	0	0	0	0	0
1	0	1	0	1	1	0	1
1	1	0	1	0	0	1	0
1	1	1	1	1	0	1	1

-Use K-maps to get D_A and D_B in terms of the states A and B and the input X
-Use K-map to get Z in terms of states A and B and the input X

A \ BX	00	01	11	10
0	0	0	1	0
1	0	0	1	1

$D_A = AB + BX$

A \ BX	00	01	11	10
0	0	1	1	0
1	0	1	1	0

$D_B = X$

A \ BX	00	01	11	10
0	0	0	0	0
1	0	1	0	0

$Z = AB'X$

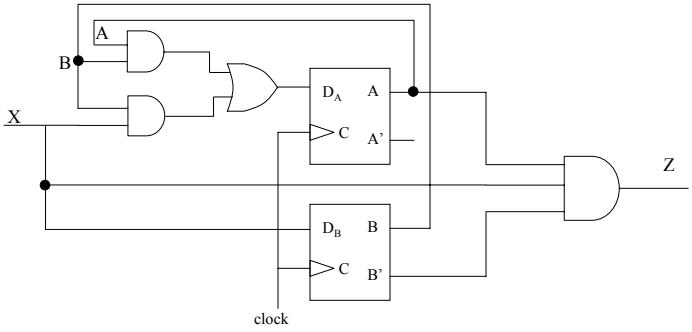
Sequence Recognizer – Design Using *D* Flip-Flops (3)

- **Solution (cont'd)**:

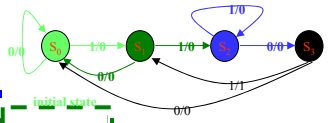
$D_A = AB + BX$

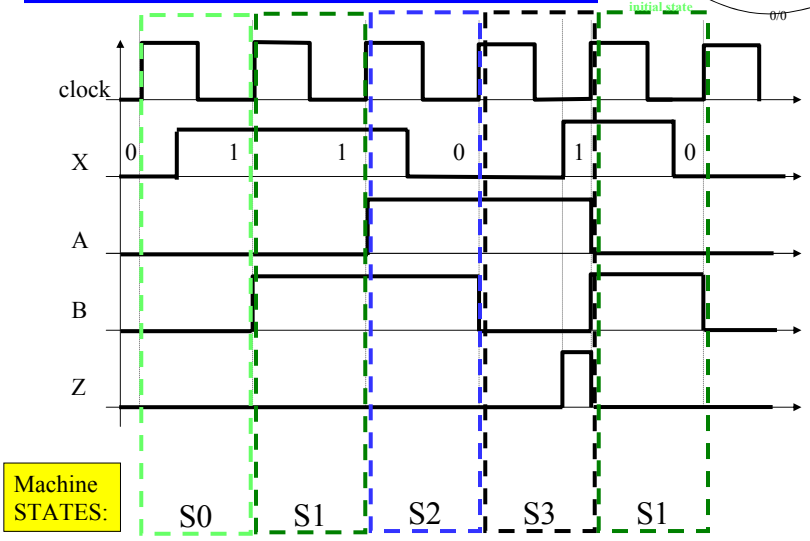
$D_B = X$

$Z = AB'X$



Let's Check Our Design – Timing Diagram





Let's Check Our Design – Timing Diagram – cont'd

- **Important Notes**
  - The value of the input prior to the positive edge is the value used to generate the rest of the outputs
    - In other words, the input signal is sampled at the positive-edge instant minus epsilon – these samples constitute the input signal X
  - Positive-edge triggered FFs respond to the input existing prior to the positive edge of the clock – and their output (state) lasts till the next positive edge at least
  - The combination logic (AND gate for this example) for producing Z responds to instantaneously to signals at the input of this combination logic – regardless of the clock signal

Sequence Recognizer – Design Using *JK* Flip-Flops

- **Solution (cont'd)**:

-Use the excitation table for the *JK* flip-flop
-To fill the J/K entries for each flip-flop

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

(a) *JK* Flip-Flop

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

Present State		Input	Next State		Output	*JK* Flip-Flop Input		*JK* Flip-Flop Input	
A	B	X	A	B	Z	J_A	K_A	J_B	K_B
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	1	0	1	X	X	0
1	0	0	0	0	0	X	1	0	X
1	0	1	0	1	1	X	1	1	X
1	1	0	1	0	0	X	0	X	1
1	1	1	1	1	0	X	0	X	0

Sequence Recognizer – Design Using *JK* Flip-Flops (2)

- **Solution (cont'd)**:

Present State		Input	Next State		Output	*JK* Flip-Flop Input		*JK* Flip-Flop Input	
A	B	X	A	B	Z	J_A	K_A	J_B	K_B
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	1	0	1	X	X	0
1	0	0	0	0	0	X	1	0	X
1	0	1	0	1	1	X	1	1	X
1	1	0	1	0	0	X	0	X	1
1	1	1	1	1	0	X	0	X	0

-Use K-maps to get D_A and D_B in terms of the states A and B and the input X
-Use K-map to get Z in terms of states A and B and the input X

A \ BX	00	01	11	10
0	0	0	1	0
1	x	x	x	x

$J_A = BX$

A \ BX	00	01	11	10
0	x	x	x	x
1	1	1	0	0

$K_A = B'$

A \ BX	00	01	11	10
0	0	1	x	x
1	0	1	x	x

$J_B = X$

A \ BX	00	01	11	10
0	x	x	0	1
1	x	x	0	1

$K_B = X'$

A \ BX	00	01	11	10
0	0	0	0	0
1	0	1	0	0

$Z = AB'X$

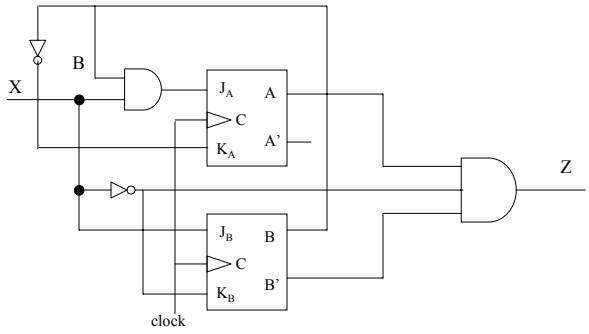
Sequence Recognizer – Design Using *JK* Flip-Flops (3)

- **Solution (cont'd)**:

$J_A = BX$ $K_A = B'$

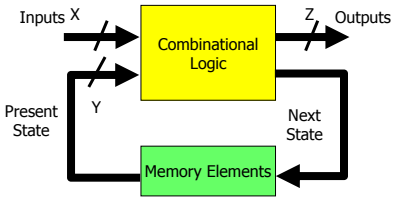
$J_B = X$ $K_B = X'$

$Z = AB'X$



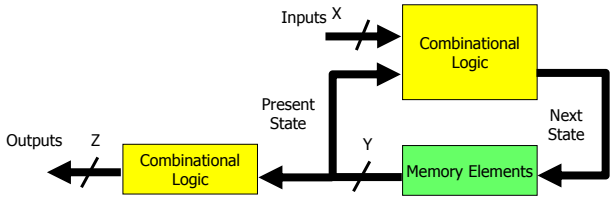
Mealy and Moore Type Finite State Machines

- Mealy Machine:
  - In a Mealy machine, the outputs are a function of the present state and the value of the inputs as shown in figure.
  - The outputs may change asynchronously in response to any change in the inputs.



Mealy and Moore Type Finite State Machines – cont'd

- Moore Machine:
  - In a Moore machine the outputs depend only on the present state as shown in figure.
  - A combinational logic block maps the inputs and the current state into the necessary flip-flop inputs to store the appropriate next state just like Mealy machine.
  - However, the outputs are computed by a combinational logic block whose inputs are only the flip-flops state outputs.
  - The outputs change synchronously with the state transition triggered by the active clock edge

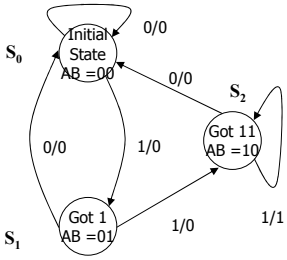


Sequence Recognizer – Mealy Machine Example

- Problem: Design a sequence recognizer (state machine) that outputs '1' if the input is '1' for three consecutive clocks – Use MEALY DESIGN – using D flip-flops

- Solution: State diagram is as shown
  State table is as shown

Arc label: X/Z

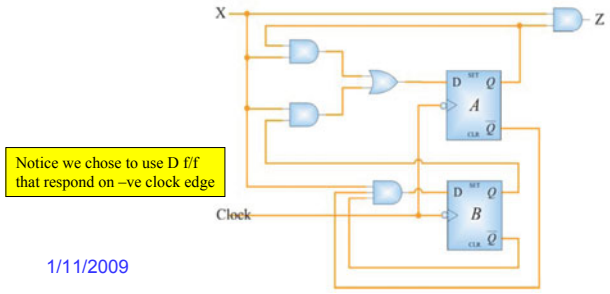


Present State		Input	Next State		Output	*D* Flip-Flops Input	
A	B	X	A	B	Z	D_A	D_B
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	0	1	1	0

Sequence Recognizer – Mealy Machine Example – cont'd

- Solution: The D flip-flops inputs are:

Present State		Input	Next State		Output	*D* Flip-Flops Input	
A	B	X	A	B	Z	D_A	D_B
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	0	1	1	0



Notice we chose to use D f/f that respond on –ve clock edge

BX / A	00	01	11	10
0	0	0	1	0
1	0	1	x	x

$D_A = AX + BX$

BX / A	00	01	11	10
0	0	1	0	0
1	0	0	x	x

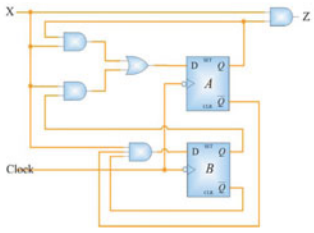
$D_B = A'B'X$

BX / A	00	01	11	10
0	0	0	0	0
1	0	1	x	x

$Z = AX$

Sequence Recognizer – Mealy Machine Example – cont'd

- Solution: Timing Diagram

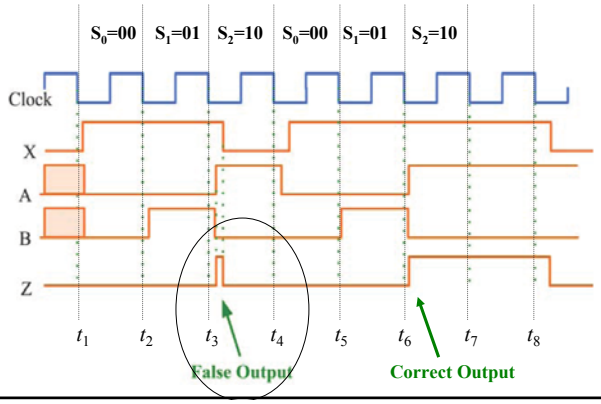


$D_A = AX + BX$

$D_B = A'B'X$

$Z = AX$

False output (or output glitch)
➔ This is a problem



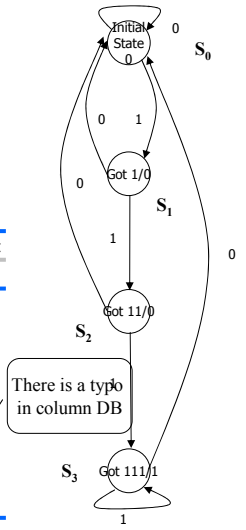
Sequence Recognizer – Moore Machine Example

- Problem: Design a sequence recognizer (state machine) that outputs '1' if the input is '1' for three consecutive clocks – Use MOORE DESIGN – using D flip-flops

Sequence Recognizer – Moore Machine Example – cont'd

- Solution – State diagram is as shown
- Observations:
  - States: are of the form AB/Z where Z is the output
  - Arc format: X where X is the input
- States codes:
  - S0 = 00, S1 = 01, S2 = 10, S3 = 11

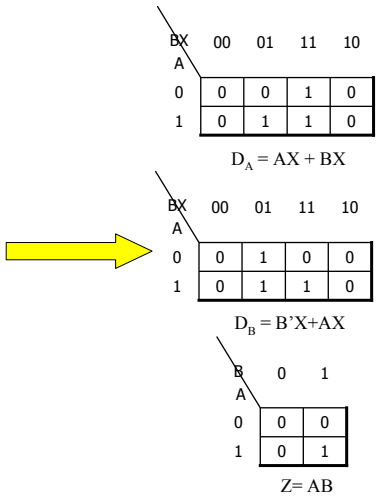
	Present State		Input	Next State		Output	*D* Flip-Flops Input	
	A	B	X	A	B	Z	D_A	D_B
S_0	0	0	0	0	0	0	0	0
	0	0	1	0	1	0	0	1
S_1	0	1	0	0	0	0	0	0
	0	1	1	1	0	0	1	0
S_2	1	0	0	0	0	0	0	0
	1	0	1	1	1	0	1	1
S_3	1	1	0	0	0	1	0	0
	1	1	1	1	1	1	1	1



Sequence Recognizer – Moore Machine Example – cont'd

- Solution:

Present State		Input	Next State		Output	*D* Flip-Flops Input	
A	B	X	A	B	Z	D_A	D_B
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	0	0	1	0	0
1	1	1	1	1	1	1	1



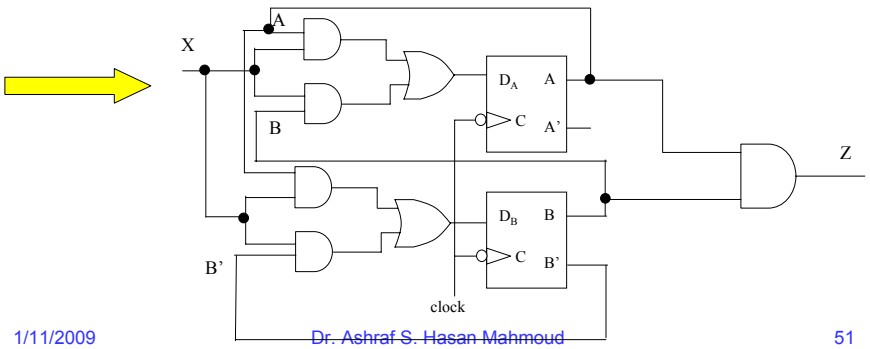
Sequence Recognizer – Moore Machine Example – cont'd

- Solution: circuit

$D_A = AX + BX$

$D_B = B'X + AX$

$Z = AB$



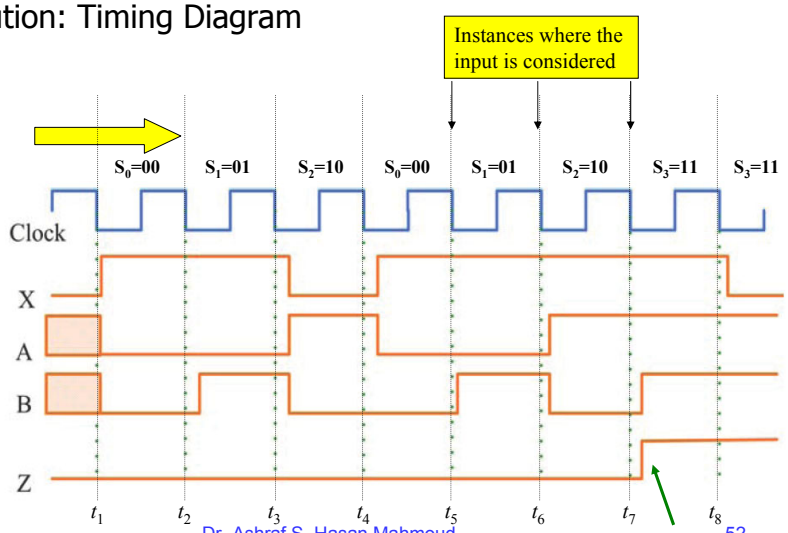
Sequence Recognizer – Moore Machine Example – cont'd

- Solution: Timing Diagram

$D_A = AX + BX$

$D_B = B'X + AX$

$Z = AB$



Serial Two's Complementer – Problem 6-15

- **Problem**: A serial two's complementer is to be designed. A binary integer of arbitrary length is presented to the serial two's complementer least significant bit first on input X. When a given bit is presented on input X, the corresponding output bit is to appear during the same clock cycle on output Z. To indicate that a sequence is complete and that the circuit is to be initialized to receive another sequence, input Y becomes 1 for one clock cycle. Otherwise, Y is 0

  a) Find the state diagram for the serial two's complementer

  b) Find the state table for the serial two's complementer

  c) Design the circuit using *D* flip-flops

  d) Design the circuit using *JK* flip-flops

Serial Two's Complementer – Problem 6-15

- **Solution**:

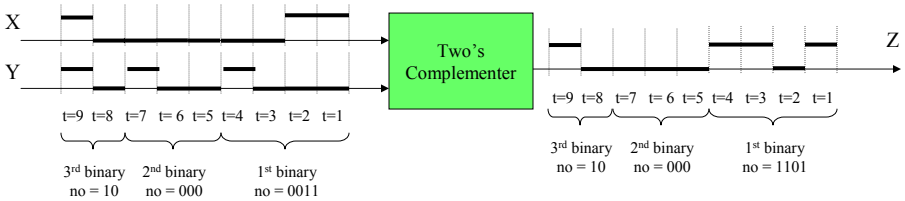
  Remember to complement $A_nA_{n-1}...A_1A_0$, we scanned the binary digits from LSB to MSB, skipping all zeros and passing the first 1 bit. All subsequent bits are complemented. The result is the two's complement of $A_nA_{n-1}...A_1A_0$

  Example: 2's complement of (10110100) is equal to (01001100)

  Example: 2's complement of (0011) is equal to (1101)

  Example: 2's complement of (000) is equal to (000)

  Example: 2's complement of (10) is equal to (10)



Serial Two's Complementer – Problem 6-15 – State Diagram

- **Solution (cont'd)**:

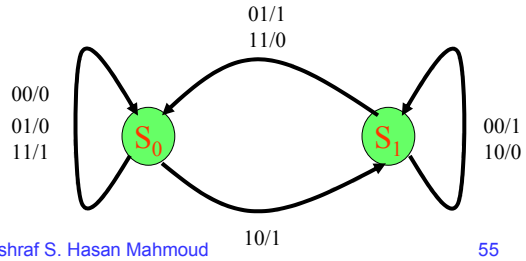
  Two inputs X: the binary bits in serial
  Y: indicator when number is complete

  Scanning the binary number, we switch between two modes:
  copying binary digits till first 1 is found
  inverting subsequent bits

  Hence TWO states are needed – need to remember that we passed the one

  Because we have four inputs, each state has FOUR departing arcs

Arc label: XY/Z



Serial Two's Complementer – Problem 6-15 – State Diagram (2)

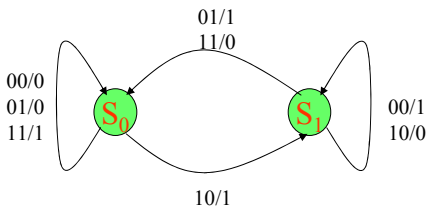
- **Solution (cont'd)**:

  State S_0: initial state (copying X to Z without inverting bits)

  1. if zero arrives (input patterns 00 or 01) on X it is copied to Z –
  2. if one arrives (input patter 11) on X it is also copied to Z if Y is 1 (i.e last bit of number)
  3. if one arrives and it is not last bit (input pattern 10) then it is copies to Z but circuit moves to the other state – to start complementing bits

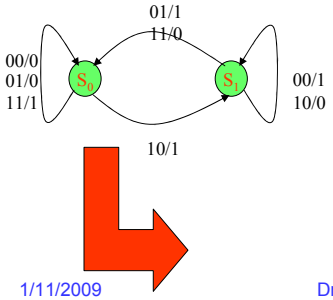
  State S_1: (copying X to Z while inverting bits) till Y = 1

  when Y = 1, another number is about to start – move to initial state S_0



Serial Two's Complementer – Problem 6-15 – State Table

- **Solution (cont'd)**:
  2 States ➔ need one flip-flop
  Let S_0 = 0, while S_1 = 1



Present State	Inputs		Next State	Output
Q(t)	X	Y	Q(t+1)	Z
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	0	1
1	0	0	1	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0

Serial Two's Complementer – Problem 6-15 – Implementation Using *D* Flip-Flops

- **Solution (cont'd)**:

Present State	Inputs		Next State	Output	*D*-Flip-Flop Input
Q(t)	X	Y	Q(t+1)	Z	D_Q
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	0	0	0

(c) *D* Flip-Flop

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

XY / Q(t)	00	01	11	10
0	0	0	0	1
1	1	0	0	1

$D_Q = QY' + XY'$

XY / Q(t)	00	01	11	10
0	0	0	1	1
1	1	1	0	0

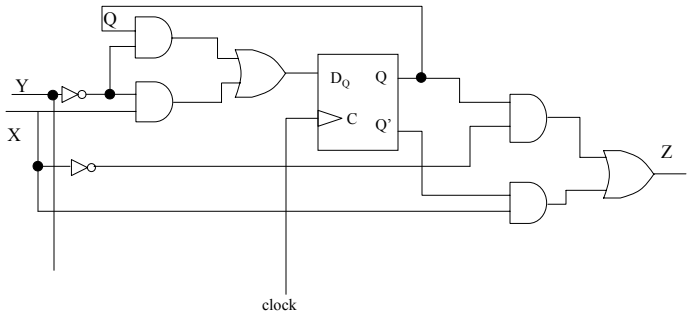
$Z = Q'X + QX'$

Serial Two's Complementer – Problem 6-15 – Implementation Using *D* Flip-Flops (2)

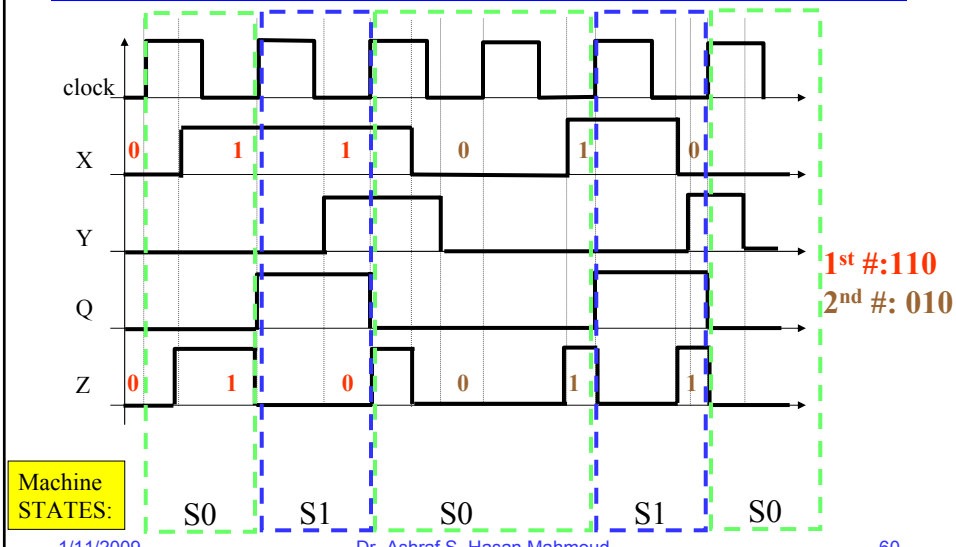
- **Solution (cont'd)**:

$D_Q = QY' + XY'$

$Z = Q'X + QX'$



Let's Check Our Design – Timing Diagram For Serial Complementer



Serial Two's Complementer – Problem 6-15 – Implementation Using *JK* Flip-Flops

- **Solution (cont'd)**:

Present State	Inputs		Next State	Output	*JK*-Flip-Flop Input	
Q(t)	X	Y	Q(t+1)	Z	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	0	0	X
0	1	0	1	1	1	X
0	1	1	0	1	0	X
1	0	0	1	1	X	0
1	0	1	0	1	X	1
1	1	0	1	0	X	0
1	1	1	0	0	X	1

(a) *JK* Flip-Flop

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

XY / Q(t)	00	01	11	10
0	0	0	0	1
1	x	x	x	x

$J_Q = XY'$

XY / Q(t)	00	01	11	10
0	x	x	x	x
1	0	1	1	0

$K_Q = Y$

XY / Q(t)	00	01	11	10
0	0	0	1	1
1	1	1	0	0

$Z = Q'X+QX'$

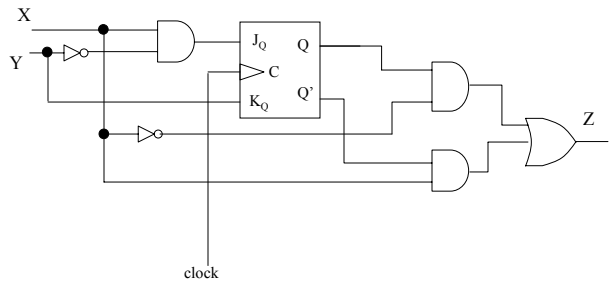
Serial Two's Complementer – Problem 6-15 – Implementation Using *JK* Flip-Flops (2)

- **Solution (cont'd)**:

$J_Q = XY'$

$K_Q = Y$

$Z = Q'X+QX'$

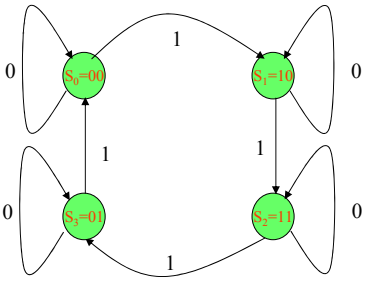


More Examples: Problem 6-14

- **Problem**: Design a sequential circuit with two *D* flip-flops A and B and one input X. When X = 0, the state of the circuit remains the same. When X = 1, the circuit goes through the state transitions 00 to 10 to 11 to 01, and back to 00, and then repeats.

Problem 6 -14 – State Diagram / Table

- **Solution**:



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

A \ BX	00	01	11	10
0	0	1	0	0
1	1	1	0	1

$D_A = AX' + B'X$

A \ BX	00	01	11	10
0	0	0	0	1
1	0	1	1	1

$D_B = AX + BX'$

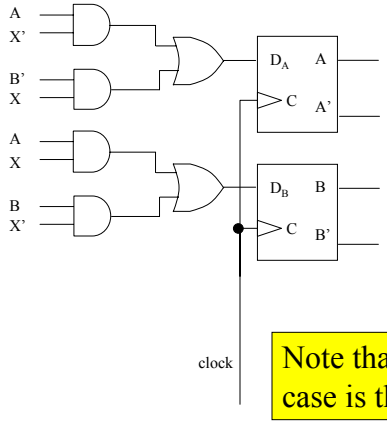
Note that the output in this case is the states AB

Problem 6-14 – Circuit Implementation

- **Solution**:

$D_A = AX' + B'X$

$D_B = AX + BX'$



Note that the output in this case is the states AB

Another Example: Problem 6-5

- **Problem**: A sequential circuit with two D flip-flops A and B, two inputs X and Y, and one output Z is specified by the following input equations:

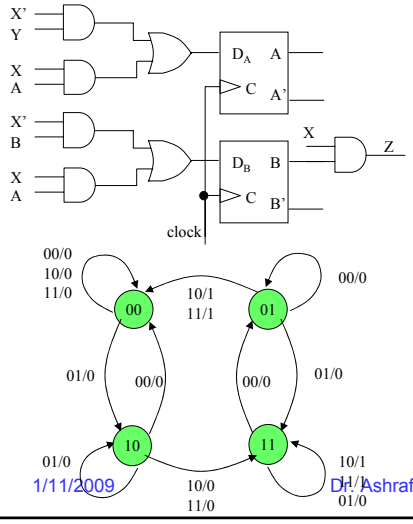
$$D_A = X'Y+XA;\ D_B = X'B+XA;\ Z = XB$$

a) Draw the logic diagram of the circuit
b) Derive the state table
c) Derive the state diagram

This is NOT a design problem – should be much easier than the ones presented earlier!

Problem 6-5:

- **Solution**:



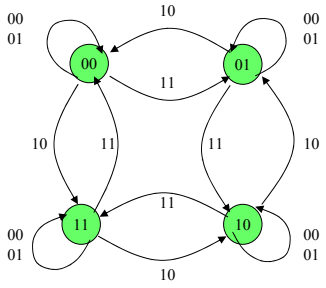
Present State		Inputs		Next State		Output
A	B	X	Y	A	B	Z
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	0	1	1	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	0	0
1	0	0	1	1	0	0
1	0	1	0	1	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Yet Another Example: Up/Down Counter with Enable

- **Problem**: Design a sequential circuit with two *JK* flip-flops A and B and two inputs X and E. If E = 0, the circuit remains in the same state, regardless of the input X. When E = 1 and X = 1, the circuit goes through the state transitions from 00 to 01 to 10 to 11, back to 00, and then repeats. When E = 1 and X = 0, the circuit goes through the state transitions from 00 to 11 to 10 to 01, back to 00 and then repeats.

Example: State Diagram/Table

- **Solution**:



Arc Label: EX

(a) *JK* Flip-Flop

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

Present State		Inputs		Next State		FF Inputs			
A	B	E	X	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	0	1	0	0	0	X	0	X
0	0	1	0	1	1	1	X	1	X
0	0	1	1	0	1	0	X	1	X
0	1	0	0	0	1	0	X	X	0
0	1	0	1	0	1	0	X	X	0
0	1	1	0	0	0	0	X	X	1
0	1	1	1	1	0	1	X	X	1
1	0	0	0	1	0	X	0	0	X
1	0	0	1	1	0	X	0	0	X
1	0	1	0	0	1	X	1	1	X
1	0	1	1	1	1	X	0	1	X
1	1	0	0	1	1	X	0	X	0
1	1	0	1	1	1	X	0	X	0
1	1	1	0	1	0	X	0	X	1
1	1	1	1	0	0	X	1	X	1

Example – Logic Circuit

- **Solution**:

EX / AB	00	01	11	10
00	0	0	0	1
01	0	0	1	0
11	x	x	x	x
10	x	x	x	x

$J_A = BEX + B'EX'$

EX / AB	00	01	11	10
00	x	x	x	x
01	x	x	x	x
11	0	0	1	0
10	0	0	0	1

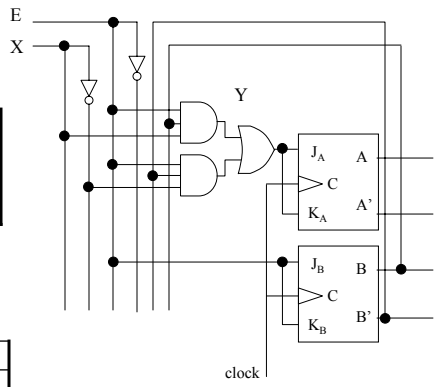
$K_A = BEX + B'EX'$

EX / AB	00	01	11	10
00	0	0	1	1
01	x	x	x	x
11	x	x	x	x
10	0	0	1	1

$J_B = E$

EX / AB	00	01	11	10
00	x	x	x	x
01	0	0	1	1
11	0	0	1	1
10	x	x	x	X

$K_B = E$



Recommended Set of Problems

- **Problems Of INTEREST:**

- Problems with "s" are solved in this slides package