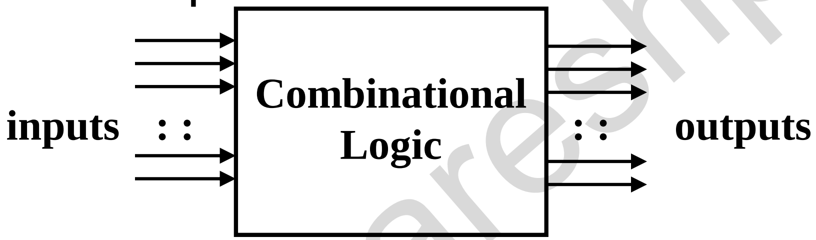
INTRODUCTION

- Two classes of logic circuits
  - Combinational
  - Sequential

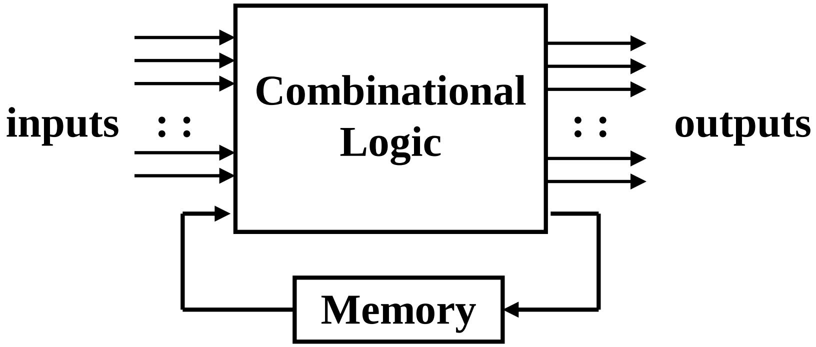
Combinational Circuit

- Each output depends entirely on the immediate (present) inputs.



Sequential Circuit

- Each output depends on both present inputs and state.

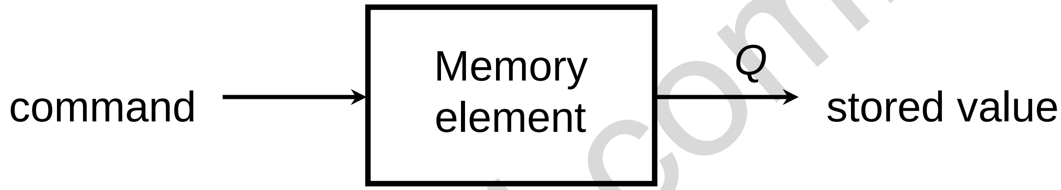


INTRODUCTION

- ❖ Two types of sequential circuits:
  - ✧ Synchronous: outputs change only at specific time
  - ✧ Asynchronous: outputs change at any time

MEMORY ELEMENTS

- **Memory element**: a device which can remember value indefinitely, or change value on command from its inputs.



- Characteristic table:

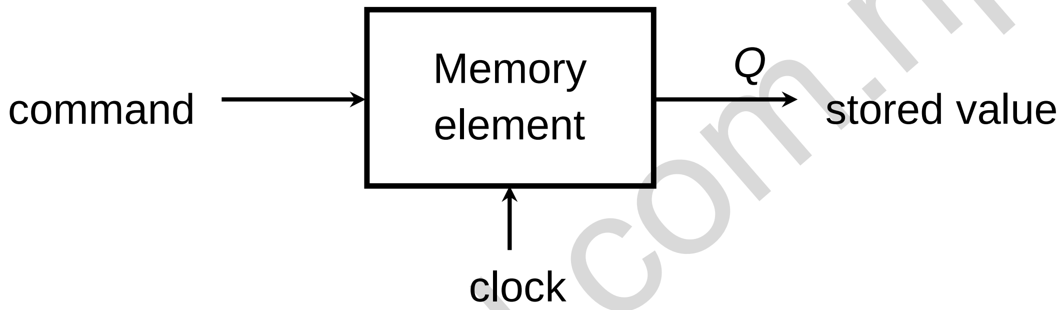
Command (at time *t*)	*Q(t)*	*Q(t+1)*
Set	X	1
Reset	X	0
Memorise / No Change	0	0
	1	1

Q(t) or ***Q***: current state

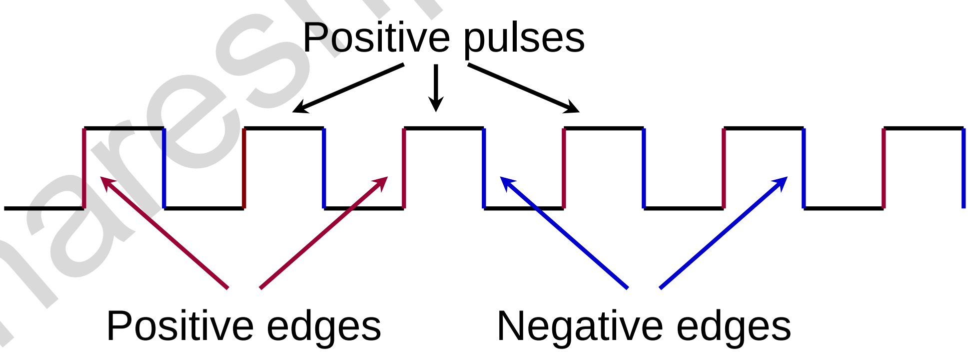
Q(t+1) or Q^+: next state

MEMORY ELEMENTS

❖ Memory element with clock.

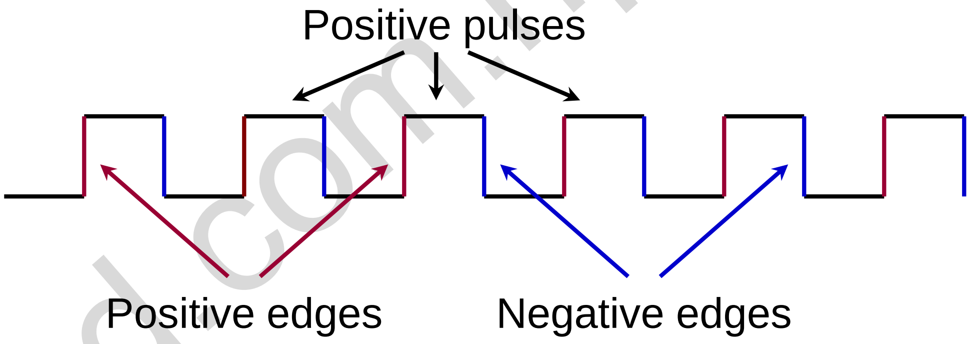


- Clock is usually a square wave.



MEMORY ELEMENTS (3/3)

- ❖ Two types of triggering/activation
  - ✧ Pulse-triggered
  - ✧ Edge-triggered
- ❖ Pulse-triggered
  - ✧ Latches
  - ✧ ON = 1, OFF = 0
- ❖ Edge-triggered
  - ✧ Flip-flops
  - ✧ Positive edge-triggered (ON = from 0 to 1; OFF = other time)
  - ✧ Negative edge-triggered (ON = from 1 to 0; OFF = other time)

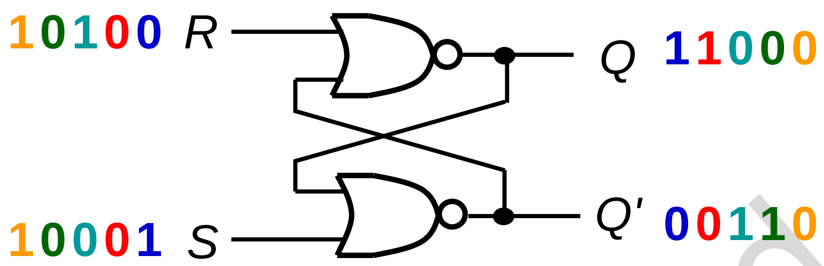


S-R LATCH

- Two inputs: *S* and *R*.
- Two complementary outputs: *Q* and *Q′*.
  - When *Q* = HIGH, we say latch is in SET state.
  - When *Q* = LOW, we say latch is in RESET state.
- For active-high input *S*-*R* latch (also known as NOR gate latch)
  - *R* = HIGH and *S* = LOW ➔ *Q* becomes LOW (RESET state)
  - *S* = HIGH and *R* = LOW ➔ *Q* becomes HIGH (SET state)
  - Both *R* and *S* are LOW ➔No change in output *Q*
  - Both *R* and *S* are HIGH ➔Outputs *Q* and *Q′* are both LOW (invalid!)
- Drawback: invalid condition exists and must be avoided.

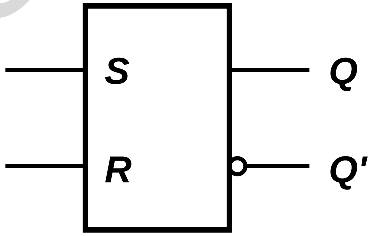
S-R LATCH

❖ Active-high input *S*-*R* latch:



S	*R*	*Q*	*Q'*	
1	0	1	0	initial
0	0	1	0	(afer *S*=1, *R*=0)
0	1	0	1	
0	0	0	1	(after *S*=0, *R*=1)
1	1	0	0	invalid!

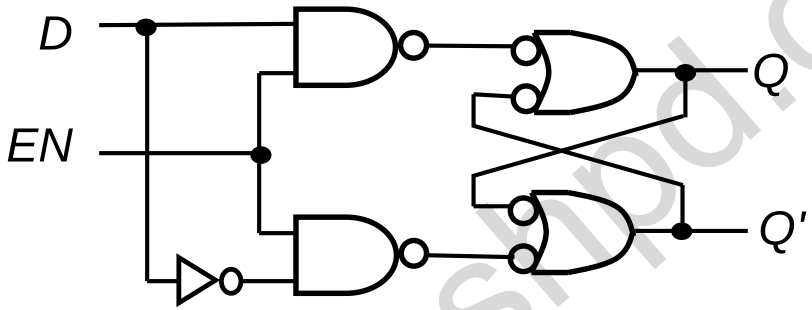
- Block diagram:

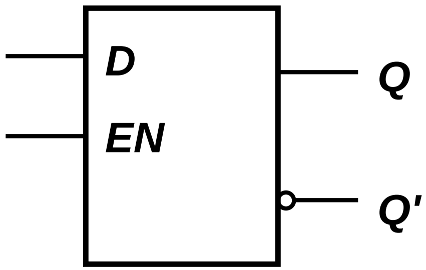


GATED D LATCH

❖ Make input *R* equal to *S'* → gated *D* latch.

❖ *D* latch eliminates the undesirable condition of invalid state in the *S*-*R* latch.





GATED D LATCH

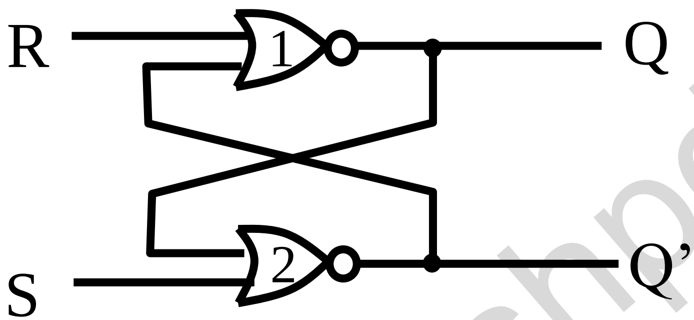
- When *EN* is high,
  - *D* = HIGH → latch is SET
  - *D* = LOW → latch is RESET
- Hence when EN is high, *Q* “follows” the *D* (data) input.
- Characteristic table:

EN	***D***	***Q(t+1)***	
1	**0**	**0**	**Reset**
1	**1**	**1**	**Set**
0	**X**	***Q(t)***	**No change**

When *EN*=1, ***Q(t+1) = ?***

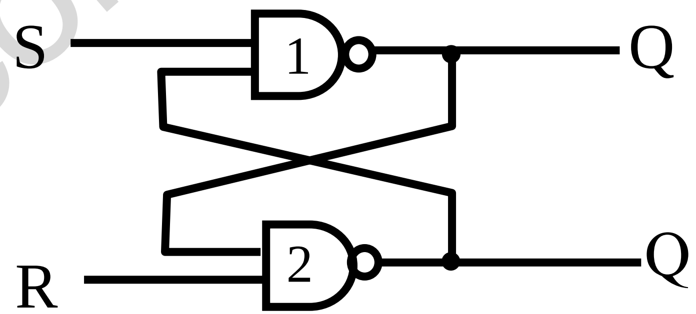
Basic Flip-Flop

- Two Output (Q and Q')
- Various Ways to Feed Flip-Flops
- NOR Gate Flip-Flops



S	R	Q	Q'
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1

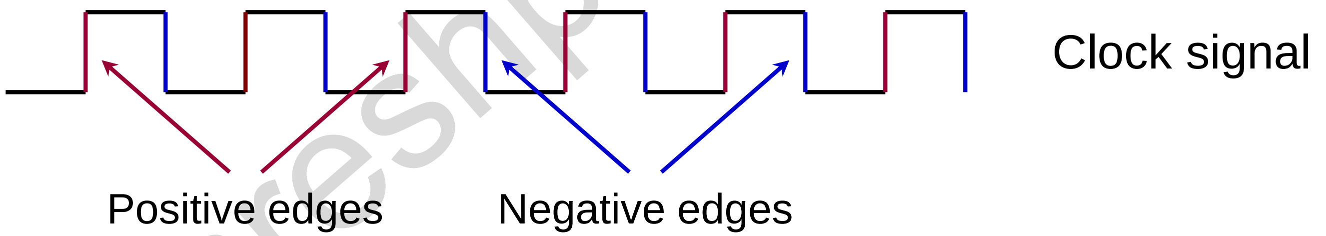
- NAND Gate Flip-Flops



S	R	Q	Q'
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0

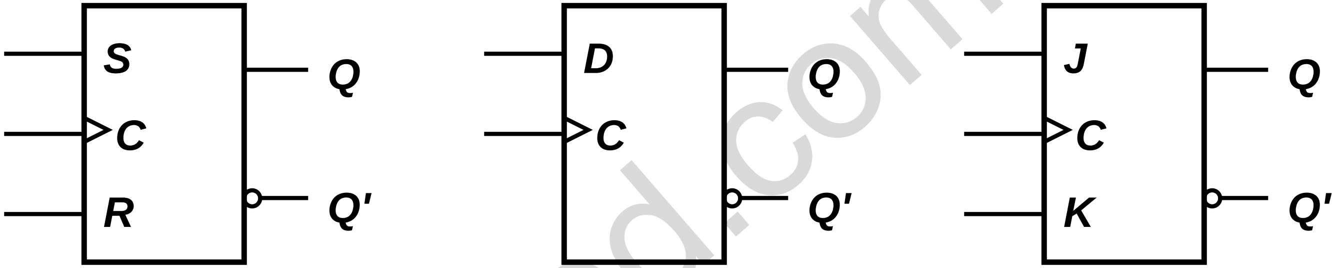
FLIP-FLOPS

- Flip-flops are synchronous bistable devices.
- Output changes state at a specified point on a triggering input called the clock.
- Change state either at the positive (rising) edge, or at the negative (falling) edge of the clock signal.

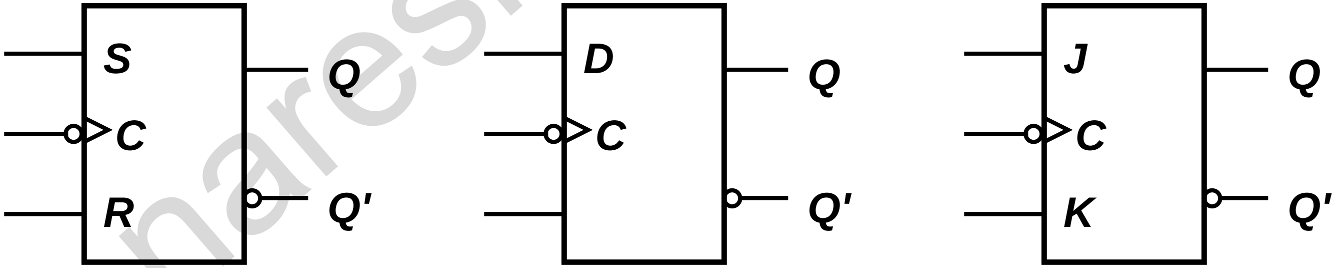


FLIP-FLOPS

- *S-R* flip-flop, *D* flip-flop, and *J-K* flip-flop.
- Note the ">" symbol at the clock input.



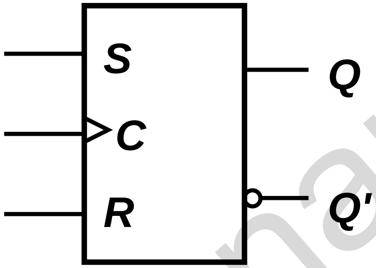
Positive edge-triggered flip-flops



Negative edge-triggered flip-flops

S-R FLIP-FLOP

- *S-R* flip-flop: On the triggering edge of the clock pulse,
  - *R* = HIGH and *S* = LOW ➔ *Q* becomes LOW (RESET state)
  - *S* = HIGH and *R* = LOW ➔ *Q* becomes HIGH (SET state)
  - Both *R* and *S* are LOW ➔ No change in output *Q*
  - Both *R* and *S* are HIGH ➔ Invalid!
- Characteristic table of positive edge-triggered *S-R* flip-flop:



S	*R*	*CLK*	*Q(t+1)*	Comments
0	0	X	*Q(t)*	No change
0	1	↑	0	Reset
1	0	↑	1	Set
1	1	↑	?	Invalid

X = irrelevant ("don't care")
↑ = clock transition LOW to HIGH

RS Flip-Flop (cont'd)

❖ Characteristic Equation:

$Q(t+1) = F(Q(t), S(t+1), R(t+1))$

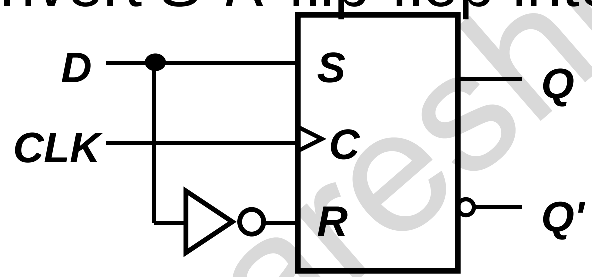
$= S + R'Q$

❖ Characteristic Table:

S	R	Q	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	i.d.
1	1	1	i.d.

D FLIP-FLOP (1/2)

- *D* flip-flop: Single input *D* (data). On the triggering edge of the clock pulse,
  - *D* = HIGH ➔ *Q* becomes HIGH (SET state)
  - *D* = LOW ➔ *Q* becomes LOW (RESET state)
- Hence, *Q* "follows" *D* at the clock edge.
- Convert *S-R* flip-flop into a *D* flip-flop: add an inverter.

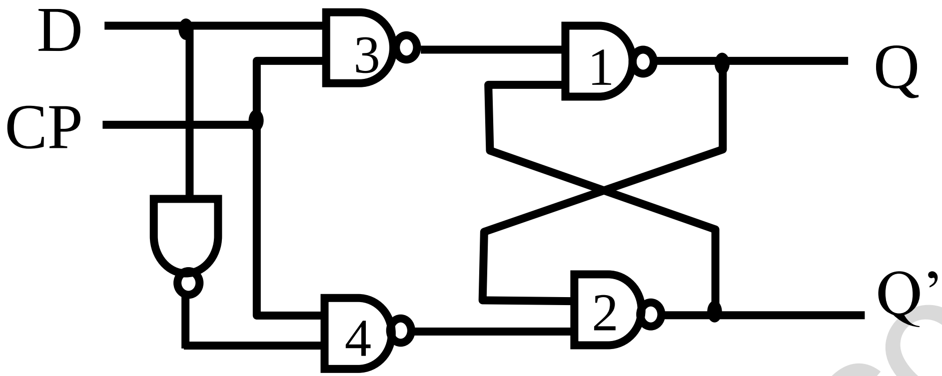


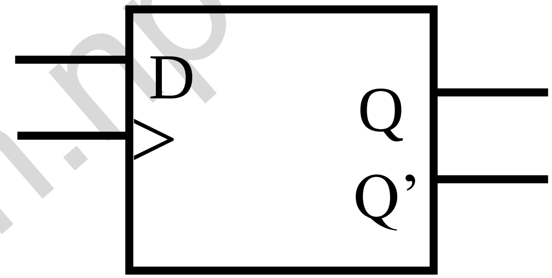
A positive edge-triggered D flip-flop formed with an S-R flip-flop.

D	*CLK*	*Q(t+1)*	Comments
1	↑	1	Set
0	↑	0	Reset

↑ = clock transition LOW to HIGH

D Flip-Flop





- Two Inputs:
  - CP: Clock Pulse
  - D: Set input

    D’: Reset input

- **Two States:**
  - **Set state: D=1, CP=1**
  - **Reset state: D=0, CP=1**
- **Characteristic Equation:**

  Q(t+1) = F(Q(t), D(t+1)) = D

JK Flip-Flop (cont'd)

❖ Characteristic Equation:

$Q(t+1) = F(Q(t), J(t+1), K(t+1))$

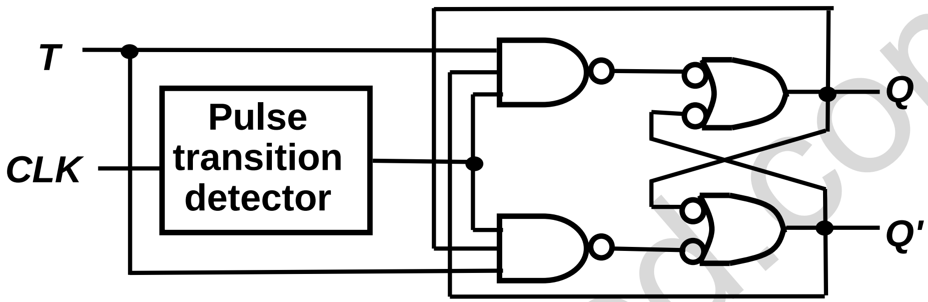
$= JQ' + K'Q$

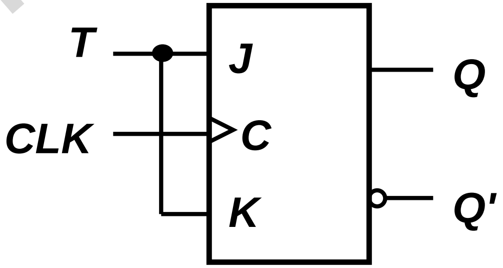
❖ Characteristic Table:

J	K	Q	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

T FLIP-FLOP

- *T* flip-flop: Single input version of the *J-K* flip-flop, formed by tying both inputs together.





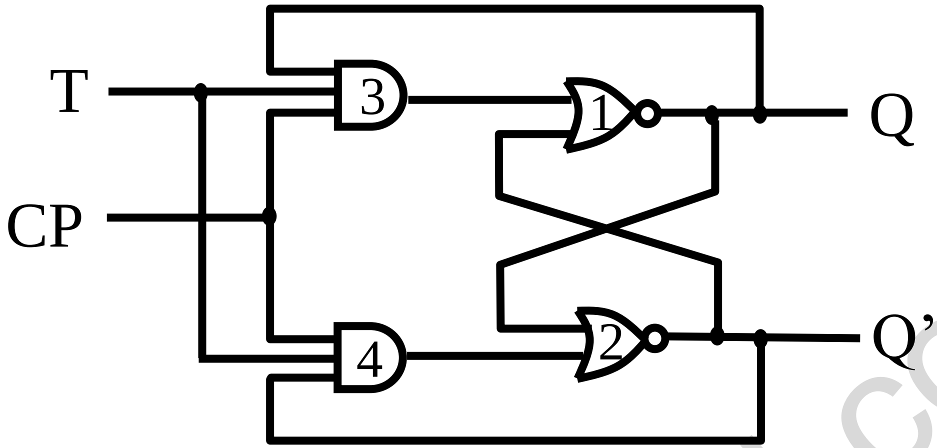
- Characteristic table:

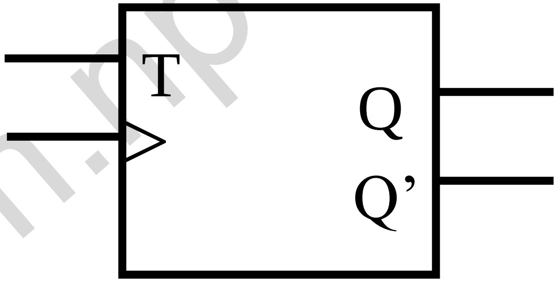
T	*CLK*	*Q(t+1)*	Comments
0	↑	*Q(t)*	No change
1	↑	*Q(t)'*	Toggle

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

Q(t+1) = ?

T Flip-Flop





- One input JK flip-flop
- Two States:
  - No Change: T=0, CP=1
  - Complement: T=1, CP=1
- Characteristic Equation:
  - $Q(t+1) = F(Q(t), T(t+1)) = TQ' + T'Q$

FLIP-FLOP CHARACTERISTIC TABLES

- Each type of flip-flop has its own behaviour, shown by its characteristic table.

J	*K*	*Q(t+1)*	Comments
0	0	*Q(t)*	No change
0	1	0	Reset
1	0	1	Set
1	1	*Q(t)'*	Toggle

S	*R*	*Q(t+1)*	Comments
0	0	*Q(t)*	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Unpredictable

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

T	*Q(t+1)*	
0	*Q(t)*	No change
1	*Q(t)'*	Toggle

SEQUENTIAL CIRCUITS: ANALYSIS

❖ Example using *D* flip-flops

State equations:

$A^{+} = A \cdot x + B \cdot x$

$B^{+} = A' \cdot x$

Output function:

$y = (A + B) \cdot x'$

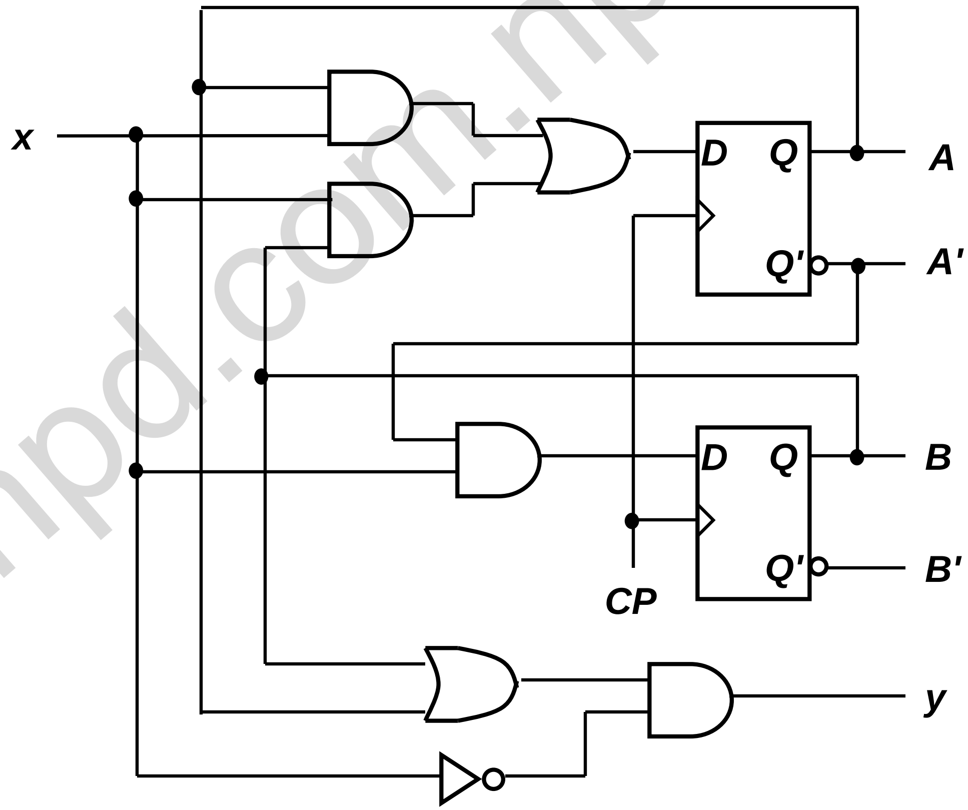


Figure 1

SEQUENTIAL CIRCUITS: ANALYSIS

- From the *state equations* and *output function*, we derive the ***state table***, consisting of all possible binary combinations of present states and inputs.
- State table
  - Similar to truth table.
  - Inputs and present state on the left side.
  - Outputs and next state on the right side.
- *m* flip-flops and *n* inputs $\rightarrow 2^{m+n}$ rows.

SEQUENTIAL CIRCUITS: ANALYSIS

❖ ***State table*** for circuit of Figure 1:

State equations:

$A^+ = A \cdot x + B \cdot x$

$B^+ = A' \cdot x$

Output function:

$y = (A + B) \cdot x'$

Present State		Input	Next State		Output
A	***B***	***x***	A^+	B^+	***y***
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

SEQUENTIAL CIRCUITS: ANALYSIS

❖ Alternative form of state table:

Full table

Present State		Input	Next State		Output
A	*B*	*x*	A^+	B^+	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Compact table

Present State	Next State		Output	
	x=0	*x*=1	*x*=0	*x*=1
AB	A^+B^+	A^+B^+	*y*	*y*
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

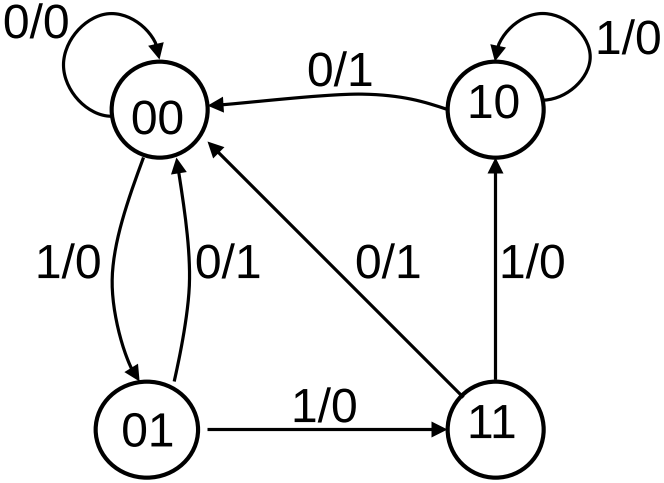
SEQUENTIAL CIRCUITS: ANALYSIS

- From the *state table*, we can draw the ***state diagram***.
- State diagram
  - Each state is denoted by a circle.
  - Each arrow (between two circles) denotes a transition of the sequential circuit (a row in state table).
  - A label of the form *a*/*b* is attached to each arrow where *a* (if there is one) denotes the inputs while *b* (if there is one) denotes the outputs of the circuit in that transition.
- Each combination of the flip-flop values represents a state. Hence, *m* flip-flops $\rightarrow$ up to 2^m states.

- **State diagram** of the circuit of Figure 1:

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
AB	A^+B^+	A^+B^+	*y*	*y*
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

DONE!



FLIP-FLOP INPUT FUNCTIONS

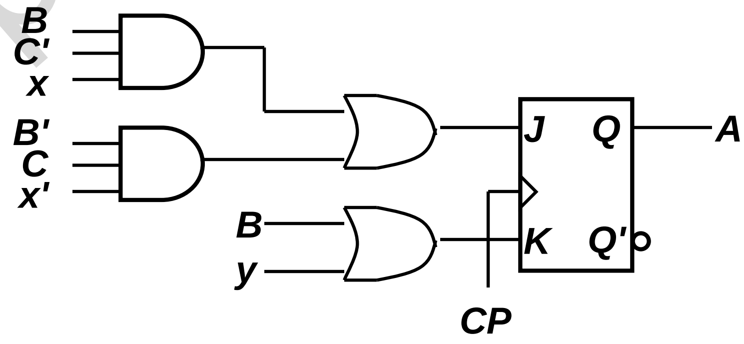
- The outputs of a sequential circuit are functions of the present states of the flip-flops and the inputs. These are described algebraically by the *circuit output functions*.
  - In Figure 1: $y = (A + B) \cdot x'$
- The part of the circuit that generates inputs to the flip-flops are described algebraically by the *flip-flop input functions* (or *flip-flop input equations*).
- The flip-flop input functions determine the next state generation.
- From the flip-flop input functions and the characteristic tables of the flip-flops, we obtain the next states of the flip-flops.

FLIP-FLOP INPUT FUNCTIONS

- Example: circuit with a *JK* flip-flop.
- We use 2 letters to denote each flip-flop input: the first letter denotes the input of the flip-flop (*J* or *K* for *J*-*K* flip-flop, *S* or *R* for *S*-*R* flip-flop, *D* for *D* flip-flop, *T* for *T* flip-flop) and the second letter denotes the name of the flip-flop.

$JA = B \cdot C' \cdot x + B' \cdot C \cdot x'$

$KA = B + y$



FLIP-FLOP INPUT FUNCTIONS

- In Figure 1, we obtain the following state equations by observing that $Q^+ = DQ$ for a D flip-flop:

$\mathbf{A^+ = A \cdot x + B \cdot x}$ (since $DA = A \cdot x + B \cdot x$)

$\mathbf{B^+ = A' \cdot x}$ (since $DB = A' \cdot x$)

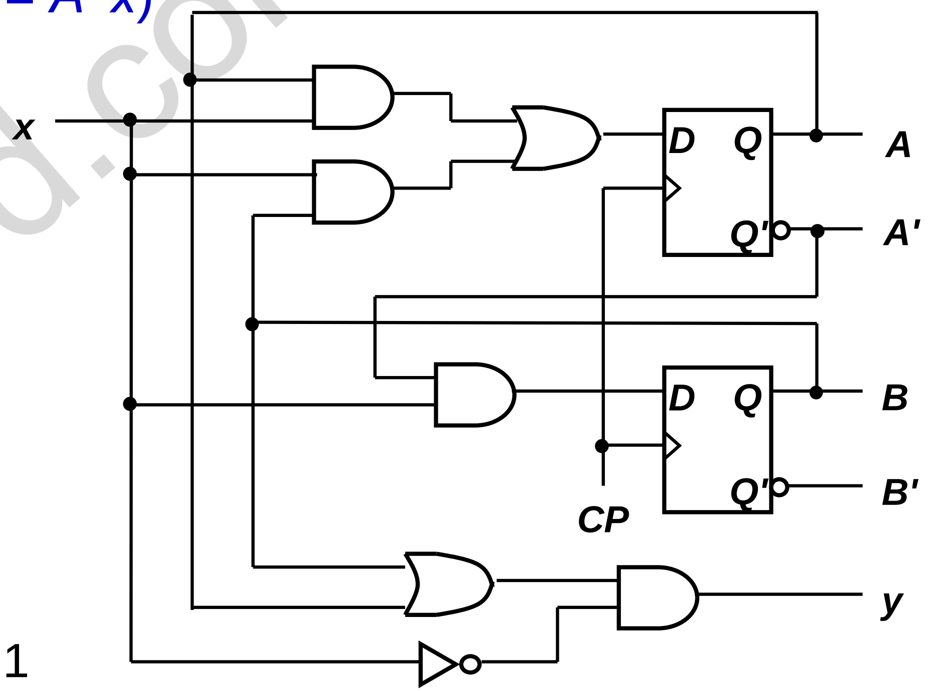


Figure 1

ANALYSIS: EXAMPLE #2 (1/3)

- Given Figure 2, a sequential circuit with two *J-K* flip-flops *A* and *B*, and one input *x*.

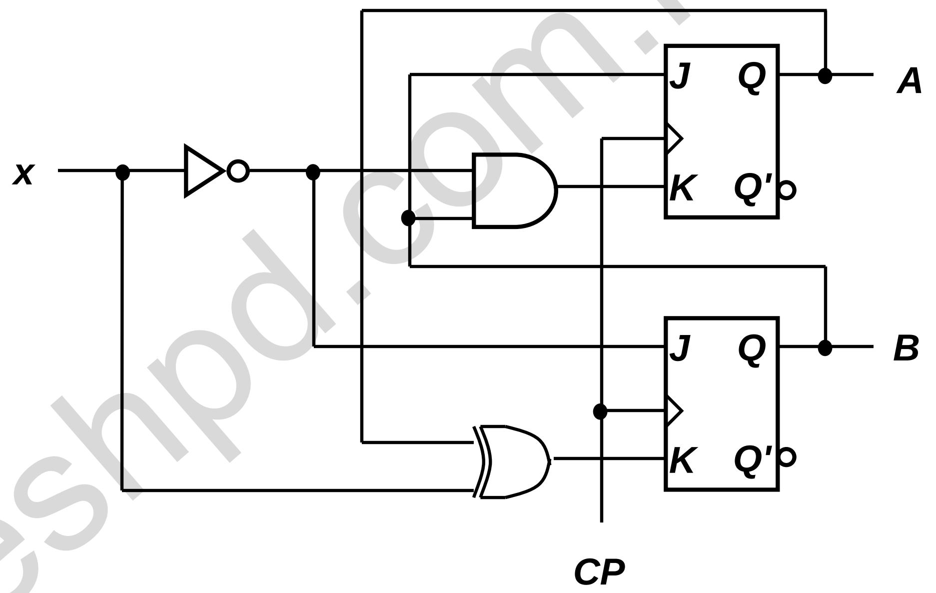


Figure 2

- Obtain the flip-flop input functions from the circuit:

$JA = B$ $\quad JB = x'$

$KA = B \cdot x'$ $\quad KB = A' \cdot x + A \cdot x' = A \oplus x$

ANALYSIS: EXAMPLE #2 (2/3)

$JA = B$ $JB = x'$

$KA = B \cdot x'$ $KB = A' \cdot x + A \cdot x' = A \oplus x$

- Fill the state table using the above functions, knowing the characteristics of the flip-flops used.

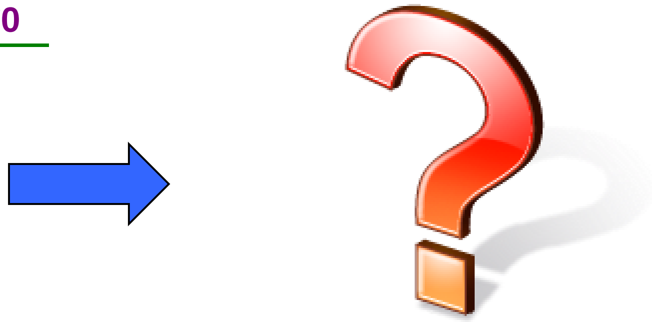
J	K	Q(t+1)	Comments
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q(t)'	Toggle

Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0			0	0	1	0
0	0	1			0	0	0	1
0	1	0			1	1	1	0
0	1	1			1	0	0	1
1	0	0			0	0	1	1
1	0	1			0	0	0	0
1	1	0			1	1	1	1
1	1	1			1	0	0	0

ANALYSIS: EXAMPLE #2 (3/3)

- Draw the state diagram from the state table.

Present state		Input	Next state		Flip-flop inputs			
A	*B*	*x*	A^+	B^+	*JA*	*KA*	*JB*	*KB*
0	0	0			0	0	1	0
0	0	1			0	0	0	1
0	1	0			1	1	1	0
0	1	1			1	0	0	1
1	0	0			0	0	1	1
1	0	1			0	0	0	0
1	1	0			1	1	1	1
1	1	1			1	0	0	0



ANALYSIS: EXAMPLE #3 (1/3)

❖ Derive the state table and state diagram of this circuit.

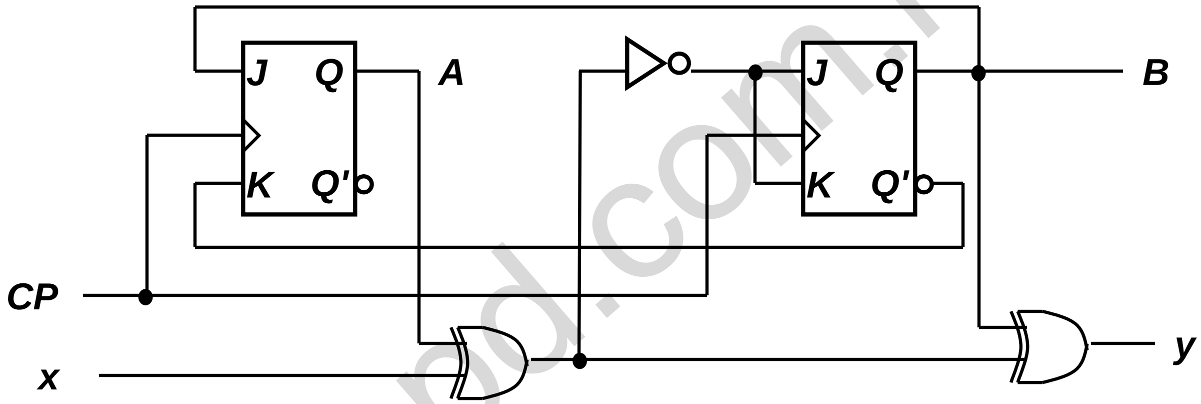


Figure 3

- Flip-flop input functions:

  $JA = B$ $\quad$ $JB = KB = (A \oplus x)' = A \cdot x + A' \cdot x'$

  $KA = B'$

ANALYSIS: EXAMPLE #3 (2/3)

- Flip-flop input functions:

  $JA = B$ $\qquad JB = KB = (A \oplus x)' = A \cdot x + A' \cdot x'$

  $KA = B'$

- State table:

Present state		Input	Next state		Output	Flip-flop inputs			
A	*B*	*x*	A^+	B^+	*y*	*JA*	*KA*	*JB*	*KB*
0	0	0			0	0	1	1	1
0	0	1			1	0	1	0	0
0	1	0			1	1	0	1	1
0	1	1			0	1	0	0	0
1	0	0			1	0	1	0	0
1	0	1			0	0	1	1	1
1	1	0			0	1	0	0	0
1	1	1			1	1	0	1	1

ANALYSIS: EXAMPLE #3 (3/3)

❖ State diagram:

Present state		Input	Next state		Output	Flip-flop inputs			
A	*B*	*x*	A^+	B^+	*y*	*JA*	*KA*	*JB*	*KB*
0	0	0			0	0	1	1	1
0	0	1			1	0	1	0	0
0	1	0			1	1	0	1	1
0	1	1			0	1	0	0	0
1	0	0			1	0	1	0	0
1	0	1			0	0	1	1	1
1	1	0			0	1	0	0	0
1	1	1			1	1	0	1	1

FLIP-FLOP EXCITATION TABLES (1/2)

- *Analysis*: Starting from a circuit diagram, derive the state table or state diagram.
- *Design*: Starting from a set of specifications (in the form of state equations, state table, or state diagram), derive the logic circuit.
- *Characteristic tables* are used in analysis.
- *Excitation tables* are used in design.

FLIP-FLOP EXCITATION TABLES (2/2)

- *Excitation tables*: given the required transition from present state to next state, determine the flip-flop input(s).

Q	Q^+	*J*	*K*
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK Flip-flop

Q	Q^+	*S*	*R*
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

SR Flip-flop

Q	Q^+	*D*
0	0	0
0	1	1
1	0	0
1	1	1

D Flip-flop

Q	Q^+	*T*
0	0	0
0	1	1
1	0	1
1	1	0

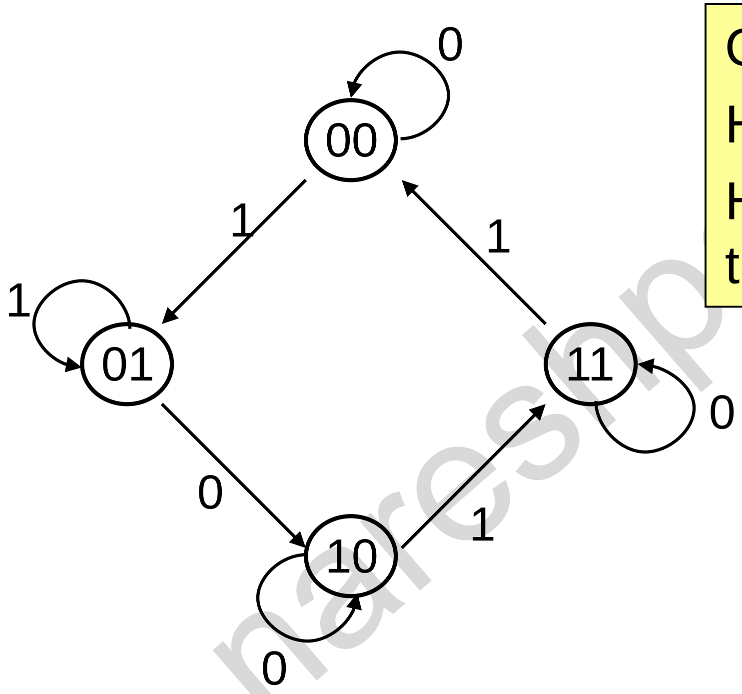
T Flip-flop

SEQUENTIAL CIRCUITS: DESIGN

- Design procedure:
  - Start with circuit specifications – description of circuit behaviour, usually a state diagram or state table.
  - Derive the state table.
  - Perform state reduction if necessary.
  - Perform state assignment.
  - Determine number of flip-flops and label them.
  - Choose the type of flip-flop to be used.
  - Derive circuit excitation and output tables from the state table.
  - Derive circuit output functions and flip-flop input functions.
  - Draw the logic diagram.

DESIGN: EXAMPLE #1 (1/5)

- Given the following state diagram, design the sequential circuit using *JK* flip-flops.



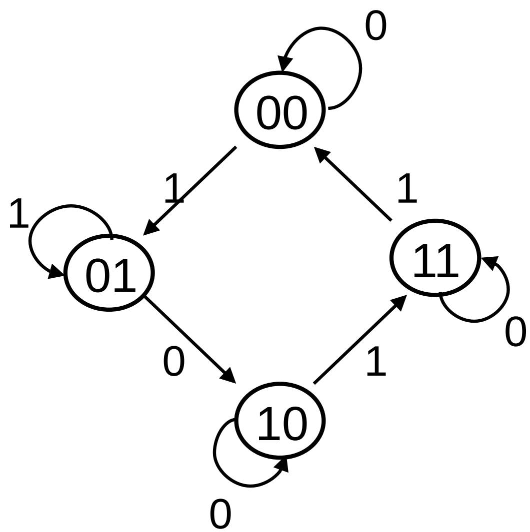
Questions:

How many flip-flops are needed?

How many input variable are there?

DESIGN: EXAMPLE #1 (2/5)

❖ Circuit state/excitation table, using *JK* flip-flops.



Present State	Next State	
	x=0	*x*=1
AB	A^+B^+	A^+B^+
00	00	01
01	10	01
10	10	11
11	11	00

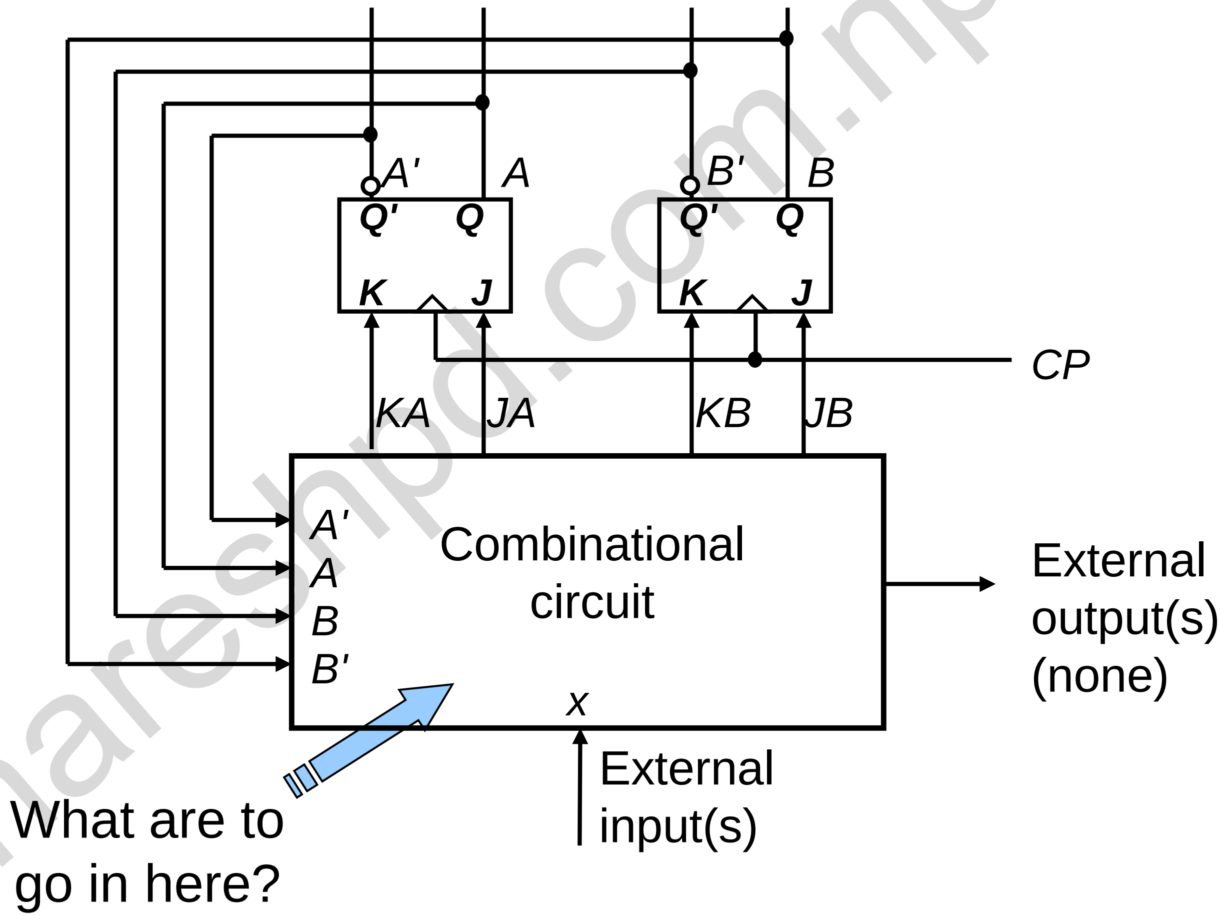
Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK Flip-flop's excitation table.

Present state		Input	Next state		Flip-flop inputs			
A	*B*	*x*	A^+	B^+	*JA*	*KA*	*JB*	*KB*
0	0	0	0	0				
0	0	1	0	1				
0	1	0	1	0				
0	1	1	0	1				
1	0	0	1	0				
1	0	1	1	1				
1	1	0	1	1				
1	1	1	0	0				

DESIGN: EXAMPLE #1 (3/5)

❖ Block diagram.



DESIGN: EXAMPLE #1 (4/5)

❖ From state table, get flip-flop input functions.

Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

A \ Bx	00	01	11	10
0				1
1	X	X	X	X

$JA = B \cdot x'$

A \ Bx	00	01	11	10
0	X	X	X	X
1			1	

$KA = B \cdot x$

A \ Bx	00	01	11	10
0		1	X	X
1		1	X	X

$JB = x$

A \ Bx	00	01	11	10
0	X	X		1
1	X	X	1	

$KB = (A \oplus x)'$

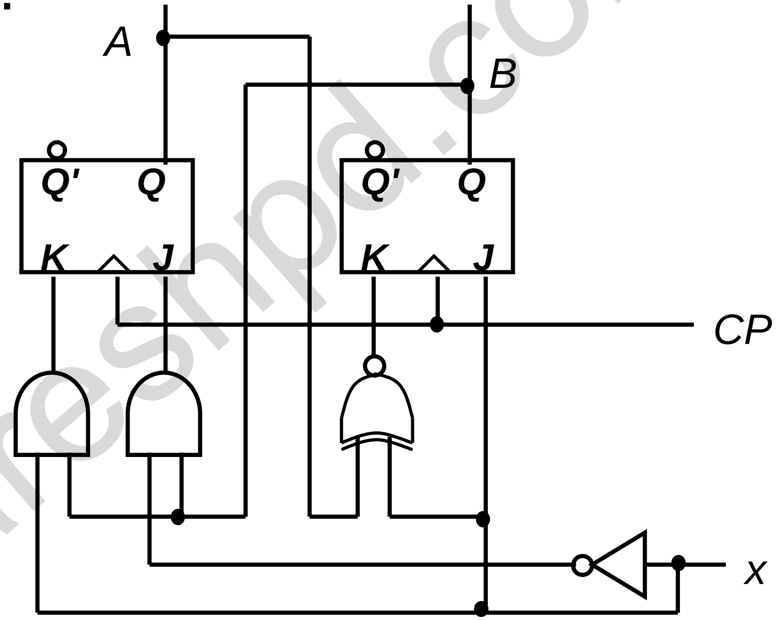
DESIGN: EXAMPLE #1 (5/5)

❖ Flip-flop input functions:

$JA = B \cdot x'$ $JB = x$

$KA = B \cdot x$ $KB = (A \oplus x)'$

❖ Logic diagram:



DESIGN: EXAMPLE #2 (1/3)

❖ Using *D* flip-flops, design the circuit based on the state table below. (Exercise: Design it using *JK* flip-flops.)

Present state		Input	Next state		Output
A	*B*	*x*	A^+	B^+	*y*
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

DESIGN: EXAMPLE #2 (2/3)

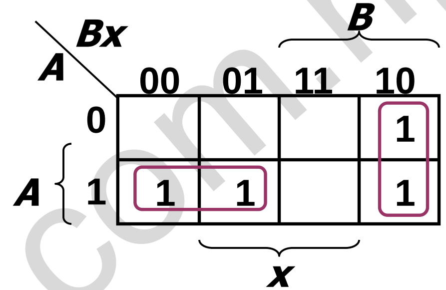
- Determine expressions for flip-flop inputs and the circuit output *y*.

Present state		Input	Next state		Output
A	*B*	*x*	A^+	B^+	*y*
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

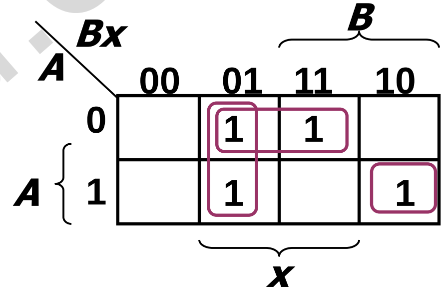
$DA(A,B,x) = \Sigma\ m(2,4,5,6)$

$DB(A,B,x) = \Sigma\ m(1,3,5,6)$

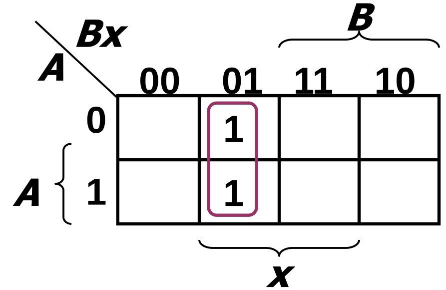
$y(A,B,x) = \Sigma\ m(1,5)$



$DA = A{\cdot}B' + B{\cdot}x'$



$DB = A'{\cdot}x + B'{\cdot}x + A{\cdot}B{\cdot}x'$



$y = B'{\cdot}x$

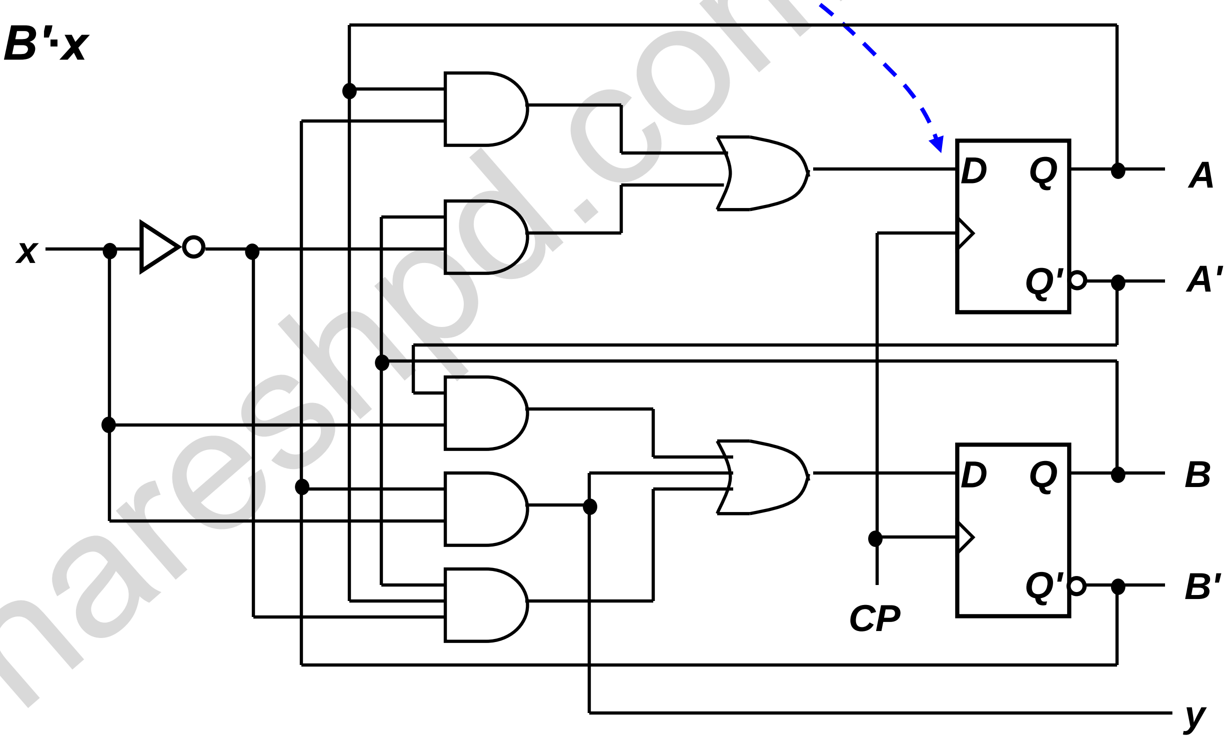
DESIGN: EXAMPLE #2 (3/3)

❖ From derived expressions, draw logic diagram:

$DA = A \cdot B' + B \cdot x'$

$DB = A' \cdot x + B' \cdot x + A.B \cdot x'$

$y = B' \cdot x$



DESIGN: EXAMPLE #3 (1/4)

❖ Design involving unused states.

Present state			Input	Next state			Flip-flop inputs						Output
A	*B*	*C*	*x*	*A*⁺	*B*⁺	*C*⁺	*SA*	*RA*	*SB*	*RB*	*SC*	*RC*	*y*
0	0	1	0	0	0	1	0	X	0	X	X	0	0
0	0	1	1	0	1	0	0	X	1	0	0	1	0
0	1	0	0	0	1	1	0	X	X	0	1	0	0
0	1	0	1	1	0	0	1	0	0	1	0	X	0
0	1	1	0	0	0	1	0	X	0	1	X	0	0
0	1	1	1	1	0	0	1	0	0	1	0	1	0
1	0	0	0	1	0	1	X	0	0	X	1	0	0
1	0	0	1	1	0	0	X	0	0	X	0	X	1
1	0	1	0	0	0	1	0	1	0	X	X	0	0
1	0	1	1	1	0	0	X	0	0	X	0	1	1

Given these — Derive these

Are there other unused states?

Unused state 000:

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

DESIGN: EXAMPLE #3 (2/4)

❖ From state table, obtain expressions for flip-flop inputs.

AB \ Cx	00	01	11	10
00	X	X		
01		1	1	
11	X	X	X	X
10	X	X	X	

SA = ?

AB \ Cx	00	01	11	10
00	X	X	X	X
01	X			X
11	X	X	X	X
10				1

RA = ?

AB \ Cx	00	01	11	10
00	X	X	1	
01	X			
11	X	X	X	X
10				

SB = ?

AB \ Cx	00	01	11	10
00	X	X		X
01		1	1	1
11	X	X	X	X
10	X	X	X	X

RB = ?

DESIGN: EXAMPLE #3 (3/4)

❖ From state table, obtain expressions for flip-flop inputs (cont'd).

AB \ Cx	00	01	11	10
00	X	X		X
01	1			X
11	X	X	X	X
10	1			X

SC = ?

AB \ Cx	00	01	11	10
00	X	X	1	
01		X	1	
11	X	X	X	X
10		X	1	

RC = ?

AB \ Cx	00	01	11	10
00	X	X		
01				
11	X	X	X	X
10		1	1	

y = ?

DESIGN: EXAMPLE #3 (4/4)

❖ From derived expressions, draw the logic diagram:

SA = B·x	***SB = A'·B'·x***	***SC = x'***	***y = A·x***
RA = C·x'	***RB = B·C + B·x***	***RC = x***