ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

COMPUTER ORGANISATION
(TỔ CHỨC MÁY TÍNH)

Sequential circuits

Acknowledgement

- The contents of these slides have origin from School of Computing, National University of Singapore.
- We greatly appreciate support from Mr. Aaron Tan Tuck Choy for kindly sharing these materials.

Policies for students

- These contents are only used for students PERSONALLY.
- Students are NOT allowed to modify or deliver these contents to anywhere or anyone for any purpose.

WHERE ARE WE NOW?

- Number systems and codes
- Boolean algebra
- Logic gates and circuits
- Simplification
- Combinational circuits
- Sequential circuits
- Performance
- Assembly language
- The processor: Datapath and control
- Pipelining
- Memory hierarchy: Cache
- Input/output

Preparation: 2 weeks

Logic Design: 3 weeks

Computer organisation

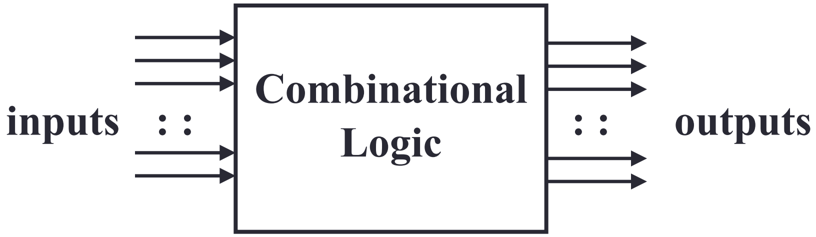
SEQUENTIAL LOGIC

- Memory Elements
- Latches: *S-R* Latch, *D* Latch
- Flip-flops: *S-R* flip-flop, *D* flip-flop, *J-K* flip-flops, *T* flip-flops
- Asynchronous Inputs
- Synchronous Sequential Circuit: Analysis and Design
- Memory
- Memory Unit
- Read/Write Operations
- Memory Arrays

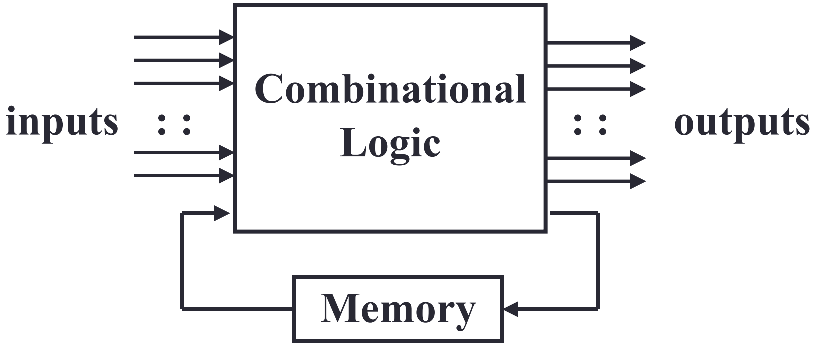
INTRODUCTION (1/2)

- 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 (2/2)

- Two types of sequential circuits:
  - Synchronous: outputs change only at specific time
  - Asynchronous: outputs change at any time
- Multivibrator: a class of sequential circuits
  - Bistable (2 stable states)
  - Monostable or one-shot (1 stable state)
  - Astable (no stable state)
- Bistable logic devices
  - Latches and flip-flops.
  - They differ in the methods used for changing their state.

MEMORY ELEMENTS (1/3)

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

- Characteristic table:

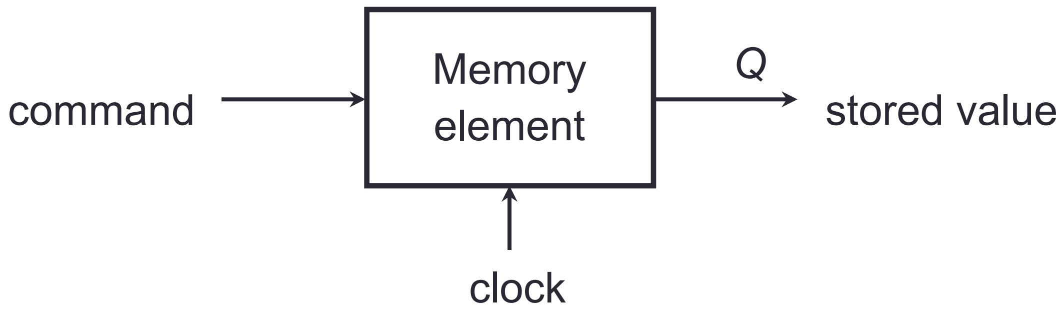
Command (at time *t*)	*Q(t)*	*Q(t+1)*
Set	X	1
Reset	X	0
Memorise / No Change	0	0
	1	1

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

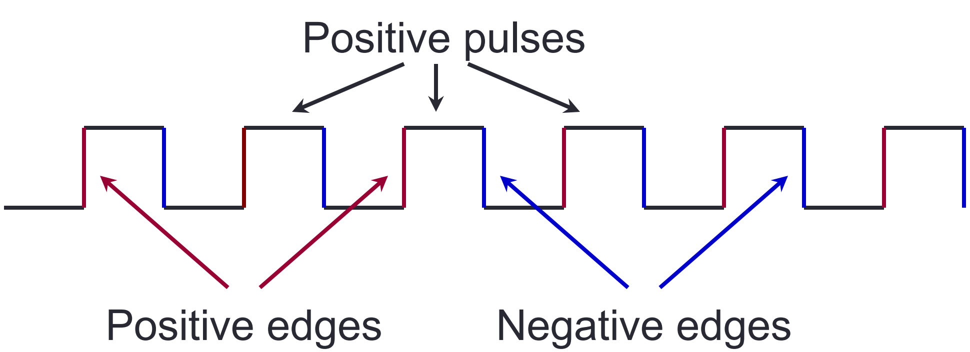
Q(t+1) or Q^+: next state

MEMORY ELEMENTS (2/3)

- Memory element with clock.

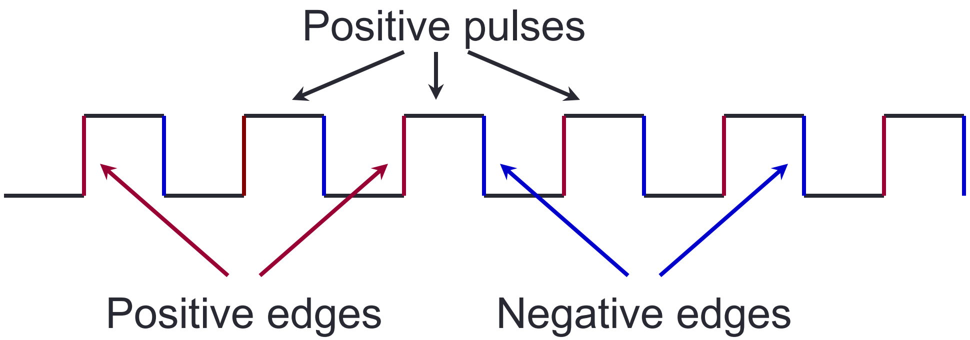


- Clock is usually a square wave.



MEMORY ELEMENTS (3/3)

- Two types of triggering/activation
  - Pulse-triggered
  - Edge-triggered



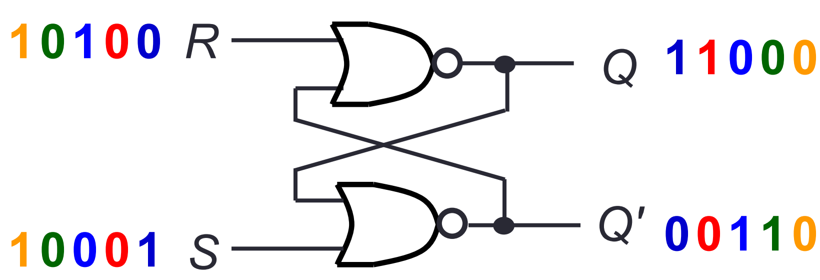
- Pulse-triggered
  - Latches
  - ON = 1, OFF = 0
- Edge-triggered
  - Flip-flops
  - Positive edge-triggered (ON = from 0 to 1; OFF = other time)
  - Negative edge-triggered (ON = from 1 to 0; OFF = other time)

S-*R* LATCH (1/3)

- 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.

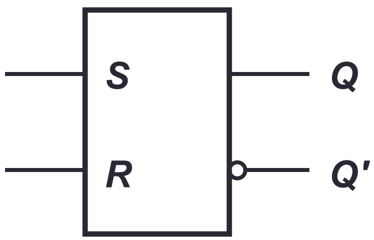
S-*R* LATCH (2/3)

- Active-high input *S*-*R* latch:



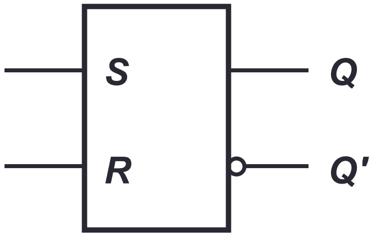
S	*R*	*Q*	*Q'*	
1	0	1	0	initial
0	0	1	0	(afer *S*=1, *R*=0)
0	1	0	1	
0	0	0	1	(after *S*=0, *R*=1)
1	1	0	0	invalid!

- Block diagram:



S-*R* LATCH (3/3)

- Characteristic table for active-high input *S*-*R* latch:



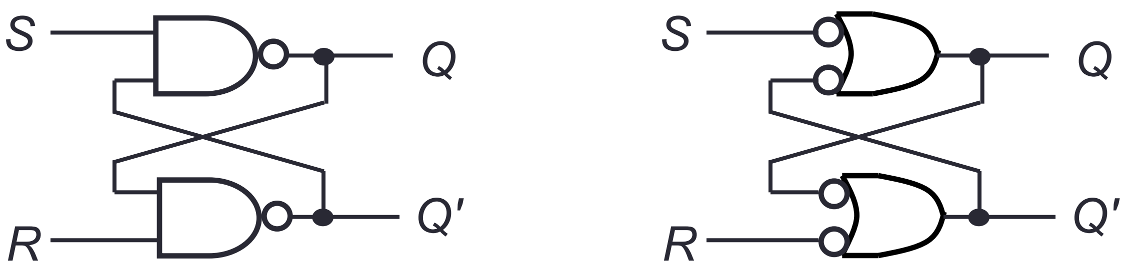
S	*R*	*Q*	*Q'*	
0	0	NC	NC	No change. Latch remained in present state.
1	0	1	0	Latch SET.
0	1	0	1	Latch RESET.
1	1	0	0	Invalid condition.

S	*R*	*Q(t+1)*	
0	0	*Q(t)*	No change
0	1	0	Reset
1	0	1	Set
1	1	indeterminate	

Q(t+1) = ?

ACTIVE-LOW *S-R* LATCH

- (You may skip this slide.)
- What we have seen is active-high input *S-R* latch.
- There are active-low input *S-R* latches, where NAND gates are used instead. See diagram on the left below.



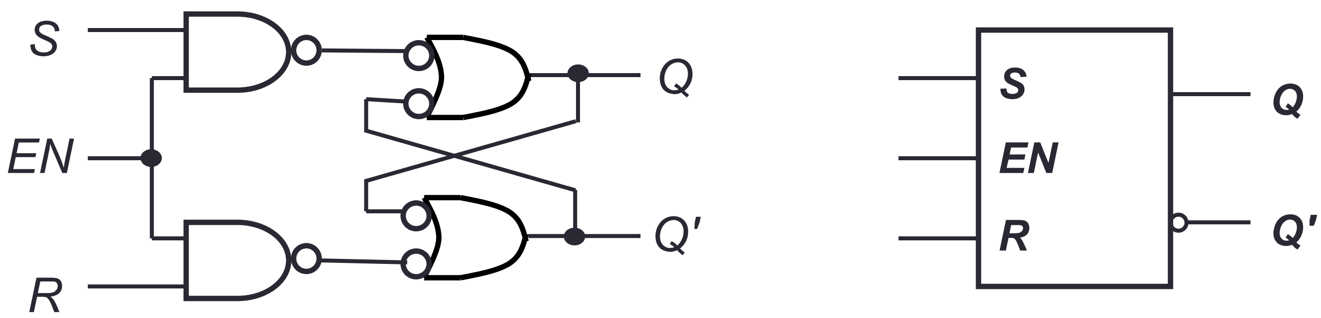
- In this case,
  - when *R*=0 and *S*=1, the latch is reset (i.e. Q becomes 0)
  - when *R*=1 and *S*=0, the latch is set (i.e. Q becomes 1)
  - when *S*=*R*=1, it is a no-change command.
  - when *S*=*R*=0, it is an invalid command.

(Sometimes, the inputs are labelled as *S'* and *R'*.)

- Sometimes, we use the alternative gate diagram for the NAND gate. See diagram on the right above. (This appears in more complex latches/flip-flops in the later slides.)

GATED *S-R* LATCH

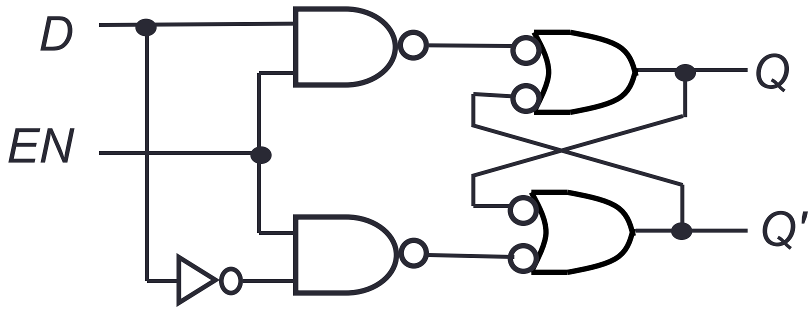
- *S-R* latch + *enable input* (*EN*) and 2 NAND gates → a gated *S-R* latch.

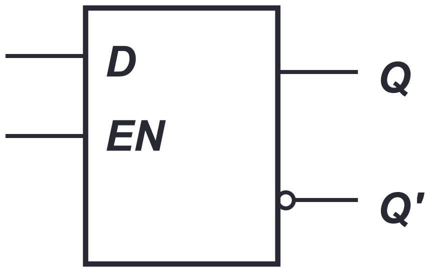


- Outputs change (if necessary) only when *EN* is high.

GATED *D* LATCH (1/2)

- 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 (2/2)

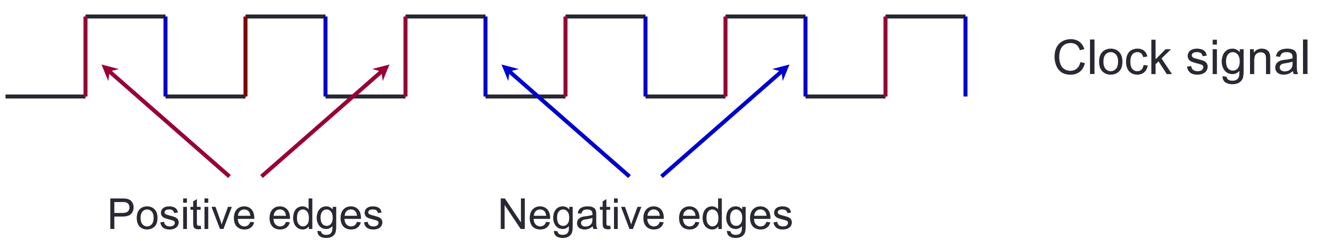
- When *EN* is high,
  - *D* = HIGH → latch is SET
  - *D* = LOW → latch is RESET
- Hence when EN is high, *Q* “follows” the *D* (data) input.
- Characteristic table:

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) = ?***

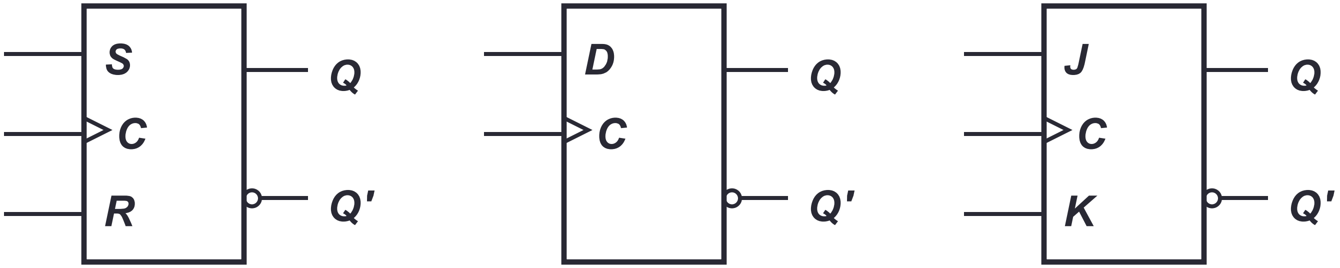
FLIP-FLOPS (1/2)

- 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.

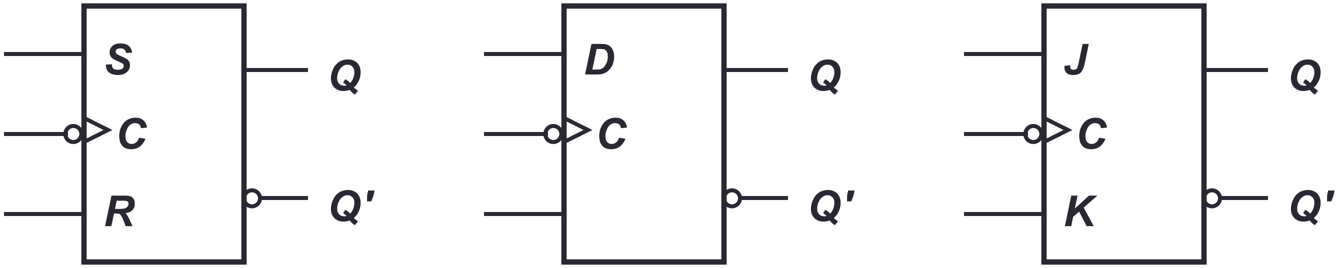


FLIP-FLOPS (2/2)

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



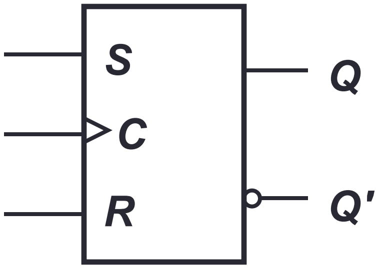
Positive edge-triggered flip-flops



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:

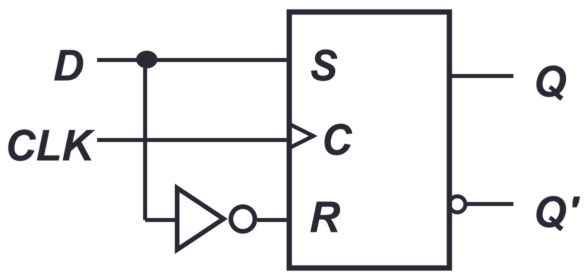


S	*R*	*CLK*	*Q(t+1)*	Comments
0	0	X	*Q(t)*	No change
0	1	↑	0	Reset
1	0	↑	1	Set
1	1	↑	?	Invalid

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

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.



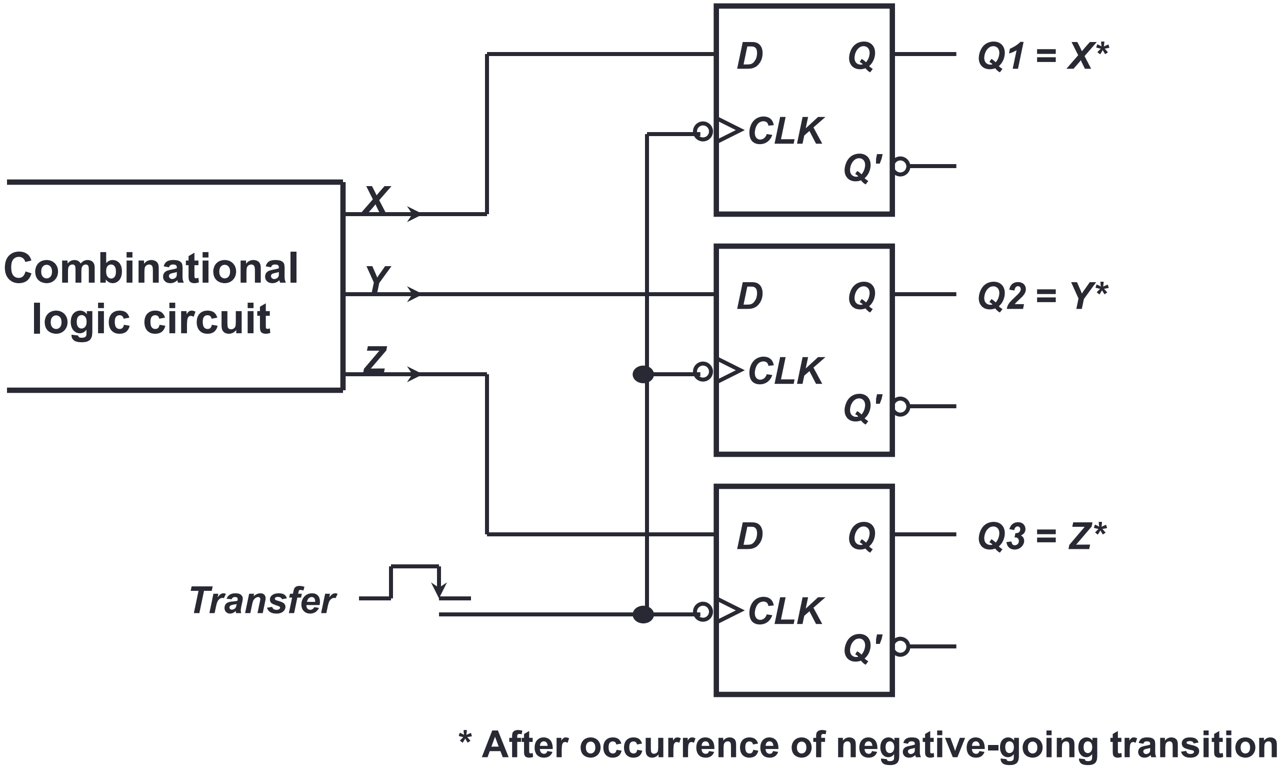
A positive edge-triggered D flip-flop formed with an S-R flip-flop.

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

↑ = clock transition LOW to HIGH

D FLIP-FLOP (2/2)

- Application: Parallel data transfer.
  - To transfer logic-circuit outputs *X*, *Y*, *Z* to flip-flops *Q1*, *Q2* and *Q3* for storage.

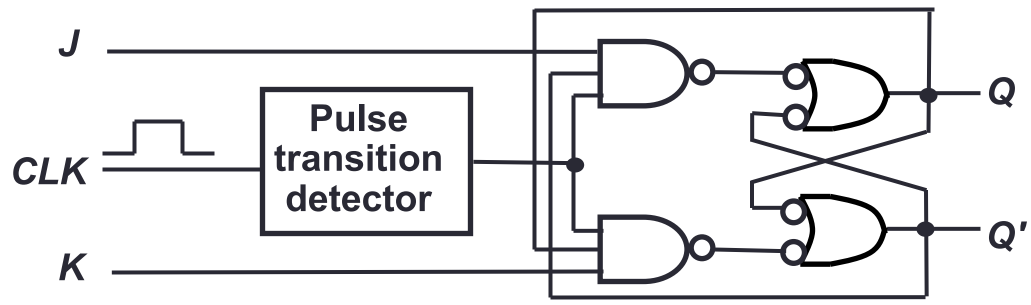


J-K FLIP-FLOP (1/2)

- *J-K* flip-flop: *Q* and *Q'* are fed back to the pulse-steering NAND gates.
- No invalid state.
- Include a toggle state
  - *J* = HIGH and *K* = LOW ➔ *Q* becomes HIGH (SET state)
  - *K* = HIGH and *J* = LOW ➔ *Q* becomes LOW (RESET state)
  - Both *J* and *K* are LOW ➔No change in output *Q*
  - Both *J* and *K* are HIGH ➔Toggle

J-K FLIP-FLOP (2/2)

- *J-K* flip-flop circuit:



- Characteristic table:

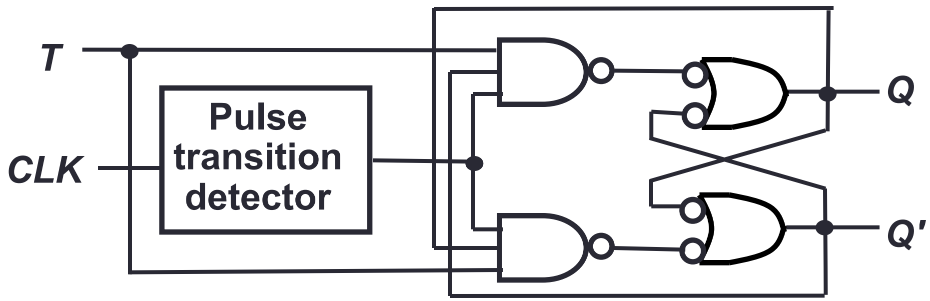
J	*K*	*CLK*	*Q(t+1)*	Comments
0	0	↑	*Q(t)*	No change
0	1	↑	0	Reset
1	0	↑	1	Set
1	1	↑	*Q(t)'*	Toggle

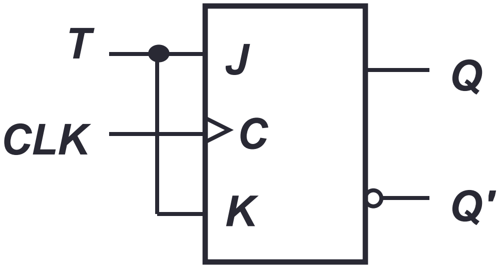
Q	*J*	*K*	*Q(t+1)*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Q(t+1) = ?

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

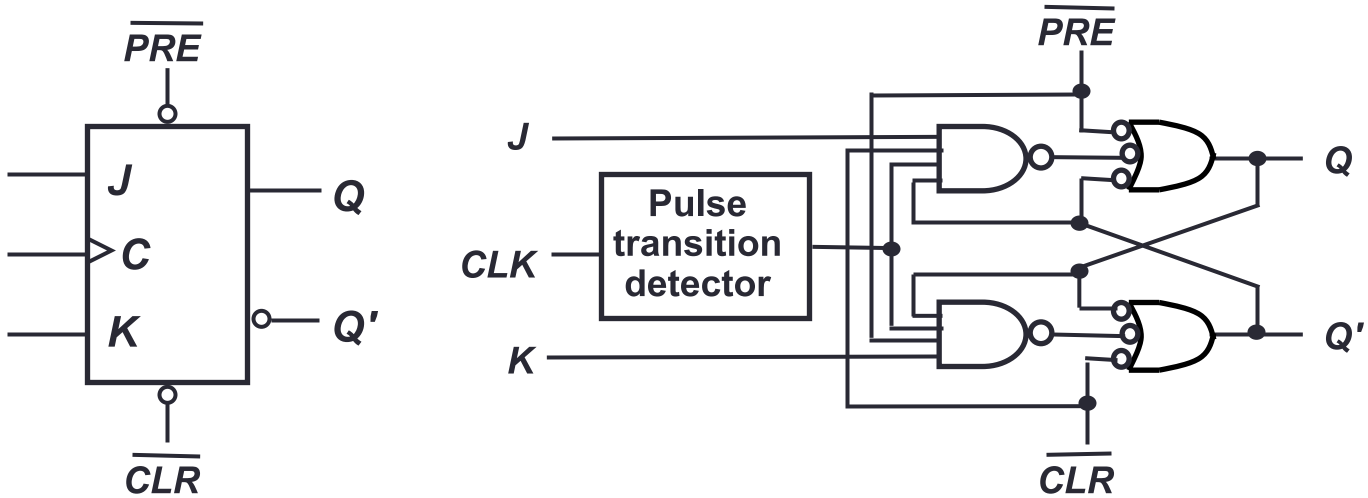
Q(t+1) = ?

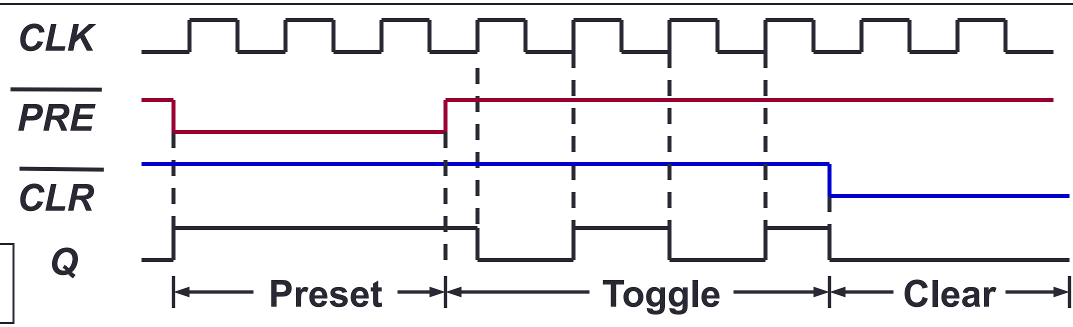
ASYNCHRONOUS INPUTS (1/2)

- *S-R*, *D* and *J-K* inputs are synchronous inputs, as data on these inputs are transferred to the flip-flop's output only on the triggered edge of the clock pulse.
- Asynchronous inputs affect the state of the flip-flop independent of the clock; example: *preset* (*PRE*) and *clear* (*CLR*) [or *direct set* (*SD*) and *direct reset* (*RD*)].
- When *PRE*=HIGH, *Q* is immediately set to HIGH.
- When *CLR*=HIGH, *Q* is immediately cleared to LOW.
- Flip-flop in normal operation mode when both *PRE* and *CLR* are LOW.

ASYNCHRONOUS INPUTS (2/2)

- A *J-K* flip-flop with active-low PRESET and CLEAR asynchronous inputs.





J = *K* = HIGH

SYNCHRONOUS SEQUENTIAL CIRCUITS

- Building blocks: logic gates and flip-flops.
- Flip-flops make up the memory while the gates form one or more combinational sub-circuits.
- We have discussed *S-R* flip-flop, *J-K* flip-flop, *D* flip-flop and *T* flip-flop.

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 (1/7)

- Given a sequential circuit diagram, we can analyze its behaviour by deriving its *state table* and hence its *state diagram*.
- Requires *state equations* to be derived for the flip-flop inputs, as well as *output functions* for the circuit outputs other than the flip-flops (if any).
- We use $\boldsymbol{A(t)}$ and $\boldsymbol{A(t+1)}$ (or simply $\boldsymbol{A}$ and $\boldsymbol{A^+}$) to represent the present state and next state, respectively, of a flip-flop represented by A.

SEQUENTIAL CIRCUITS: ANALYSIS (2/7)

- Example using *D* flip-flops

State equations:

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

$B^+ = A' \cdot x$

Output function:

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

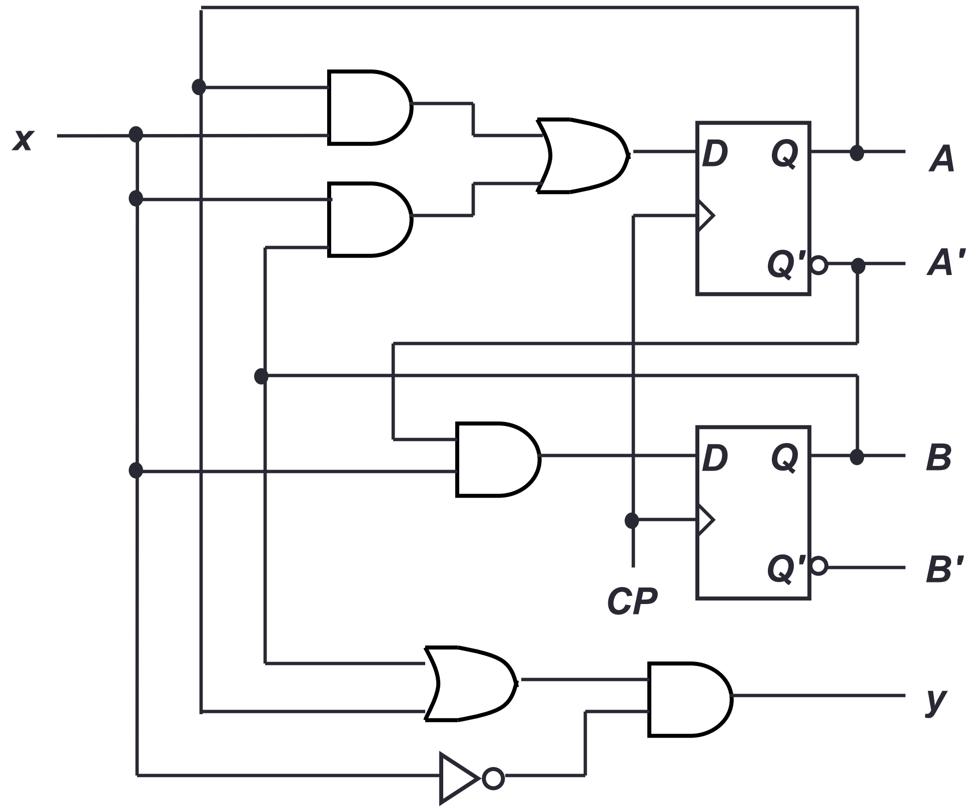


Figure 1

SEQUENTIAL CIRCUITS: ANALYSIS (3/7)

- 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 (4/7)

- ***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 (5/7)

- 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 (6/7)

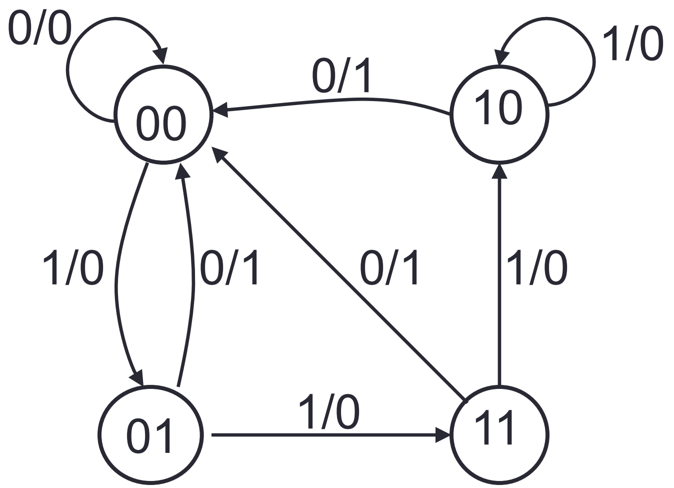
- From the *state table*, we can draw the ***state diagram***.
- State diagram
  - Each state is denoted by a circle.
  - Each arrow (between two circles) denotes a transition of the sequential circuit (a row in state table).
  - A label of the form *a*/*b* is attached to each arrow where *a* (if there 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.

SEQUENTIAL CIRCUITS: ANALYSIS (7/7)

- **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!



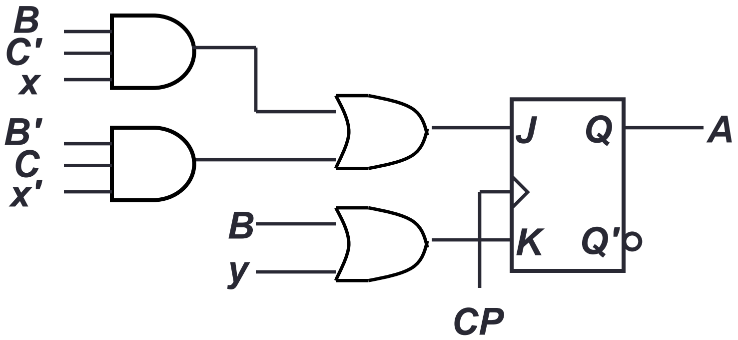
FLIP-FLOP INPUT FUNCTIONS (1/3)

- 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 (2/3)

- 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 (3/3)

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

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

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

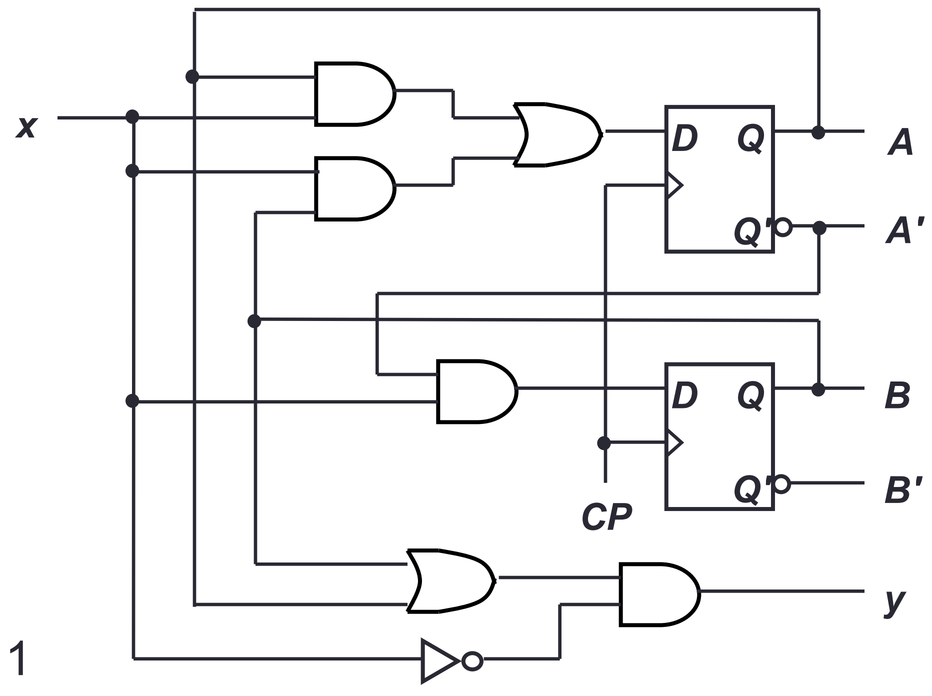


Figure 1

ANALYSIS: EXAMPLE #2 (1/3)

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

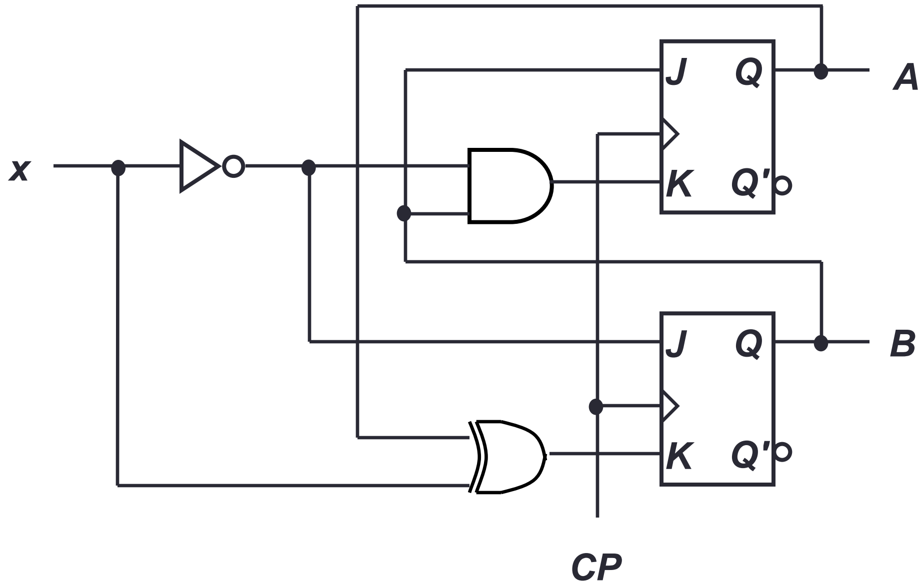


Figure 2

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

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

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

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.

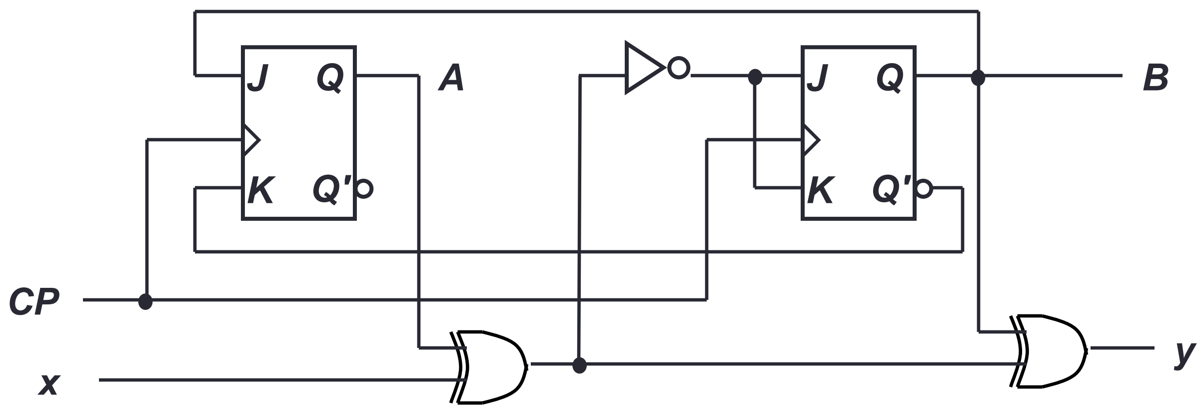


Figure 3

- Flip-flop input functions:

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

$KA = B'$

ANALYSIS: EXAMPLE #3 (2/3)

- Flip-flop input functions:

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

  $KA = B'$

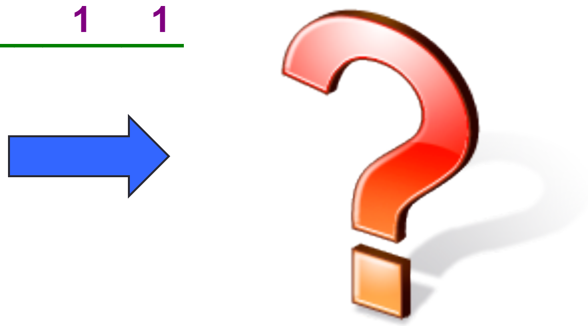
- State table:

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

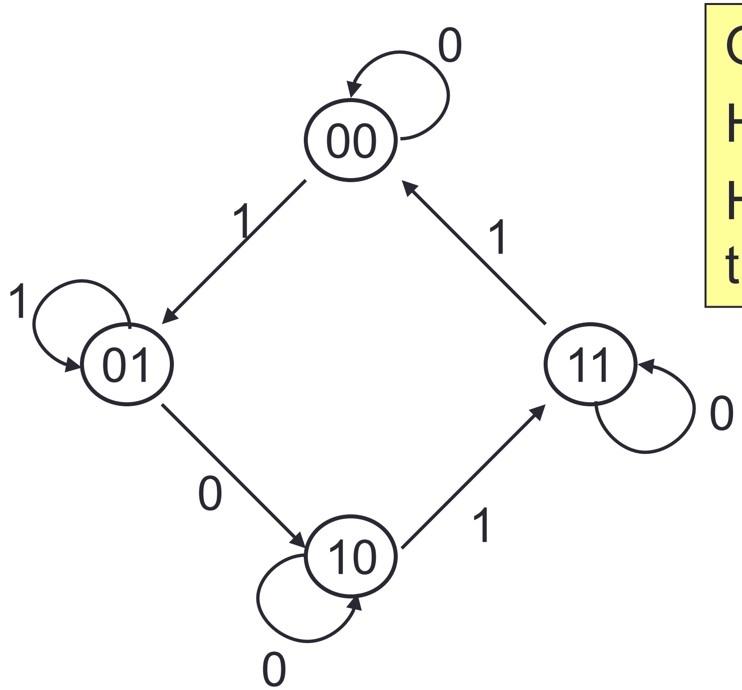
T Flip-flop

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.



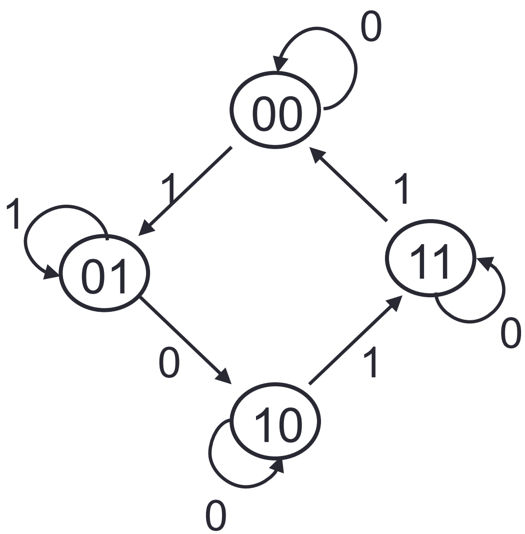
Questions:

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

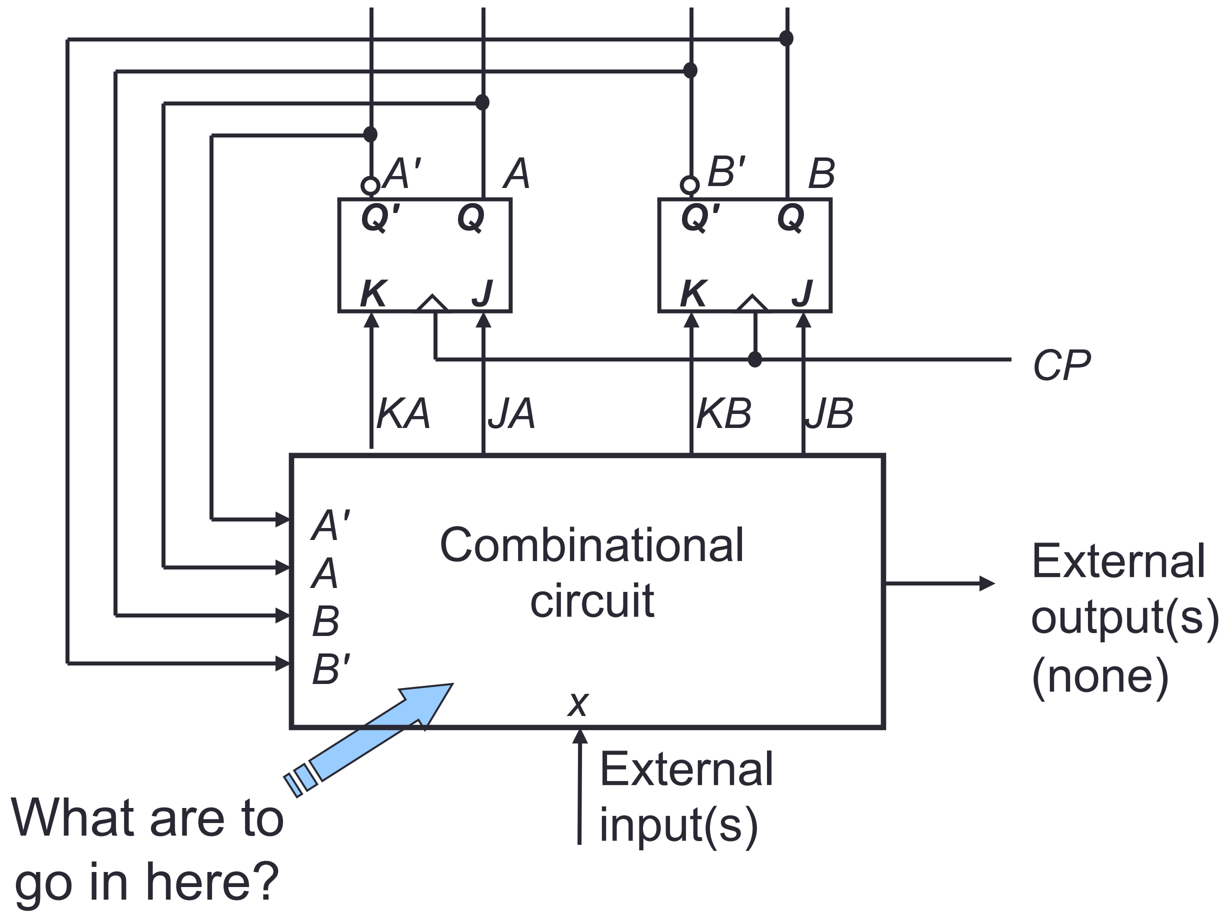
Q	Q^+	*J*	*K*
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

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)

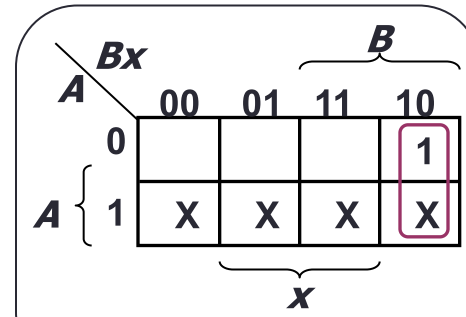
- Block diagram.



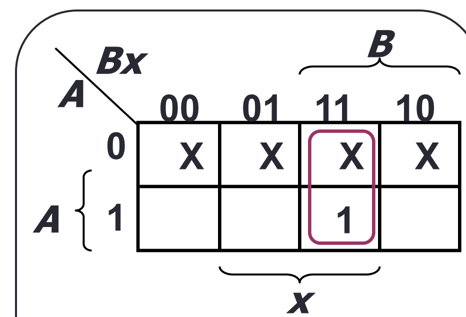
DESIGN: EXAMPLE #1 (4/5)

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

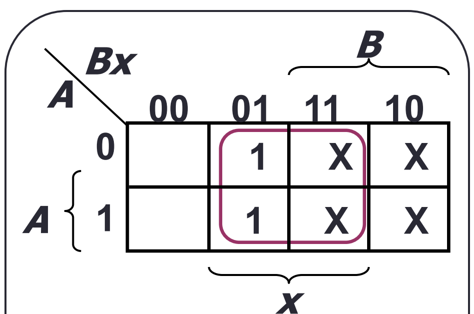
Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1



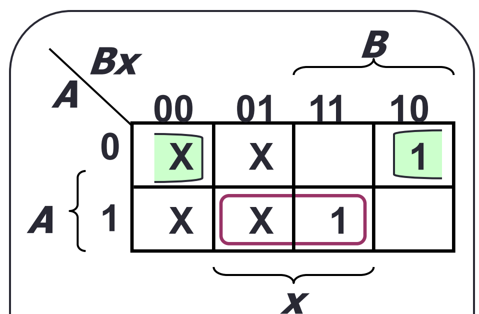
$JA = B \cdot x'$



$KA = B \cdot x$



$JB = x$



$KB = (A \oplus x)'$

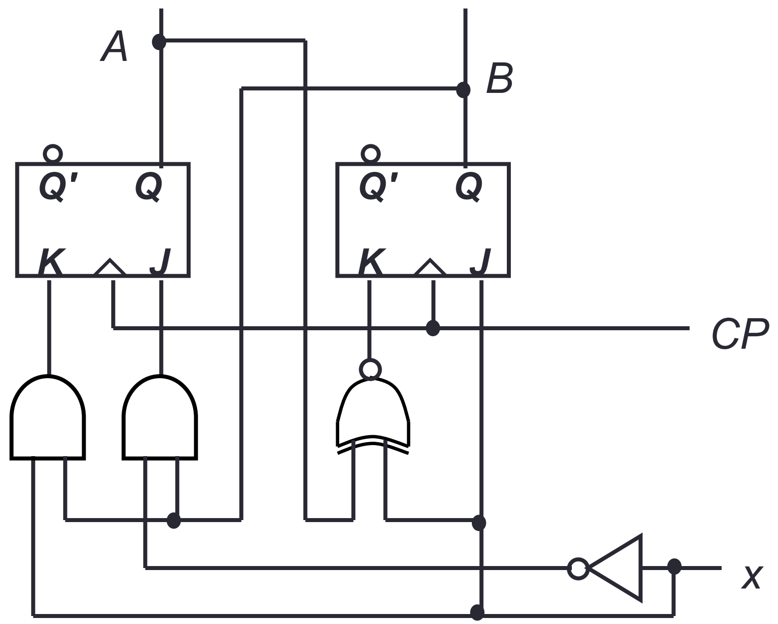
DESIGN: EXAMPLE #1 (5/5)

- 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

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

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

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

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

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

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

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

A \ Bx	00	01	11	10
0		1		
1		1		

$y = B' \cdot x$

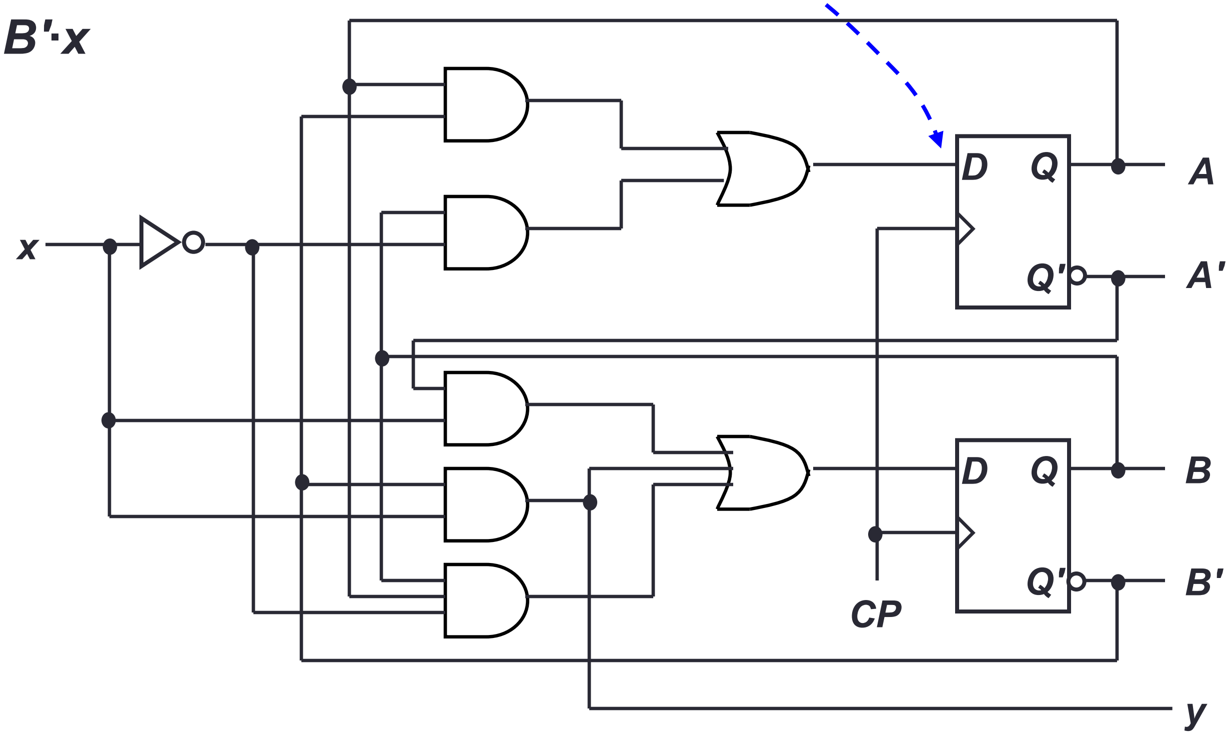
DESIGN: EXAMPLE #2 (3/3)

- 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).

SC = ?

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

RC = ?

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

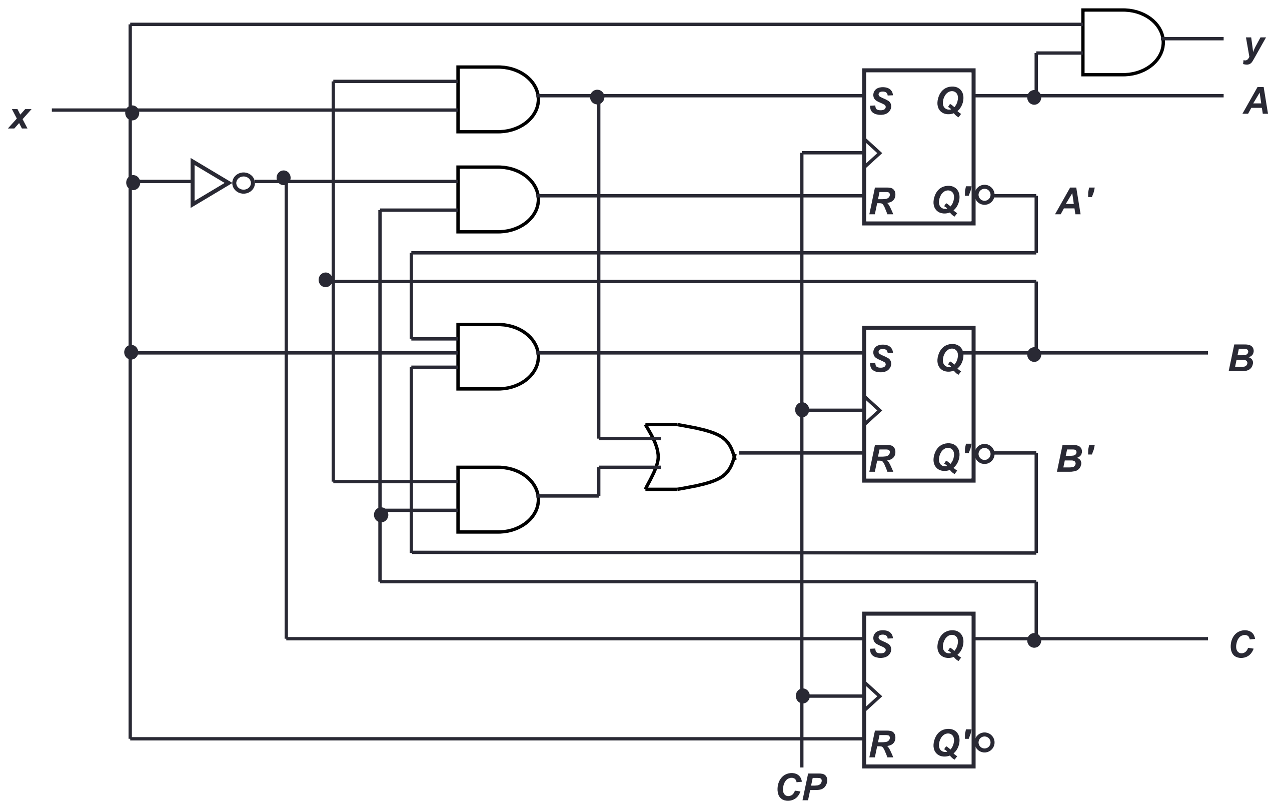
y = ?

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

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***	

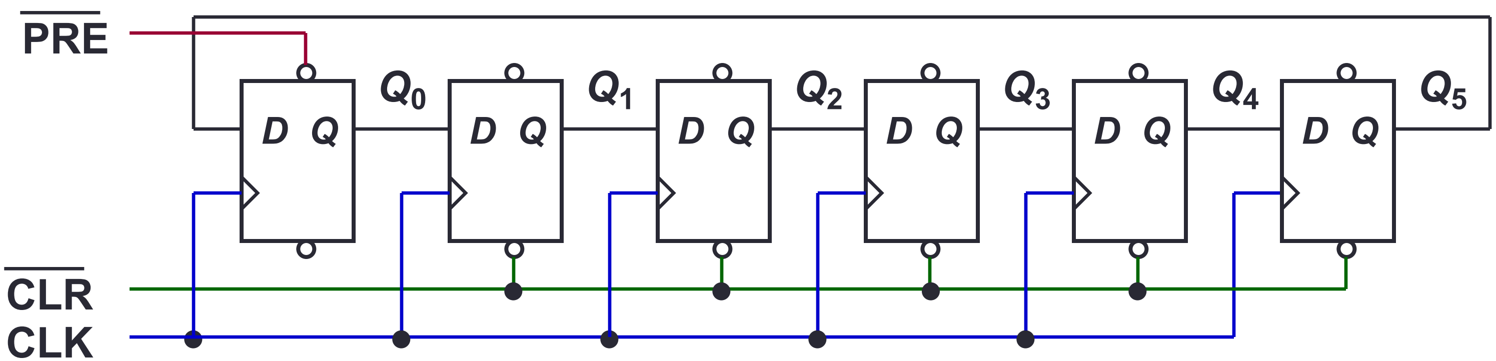


COUNTERS

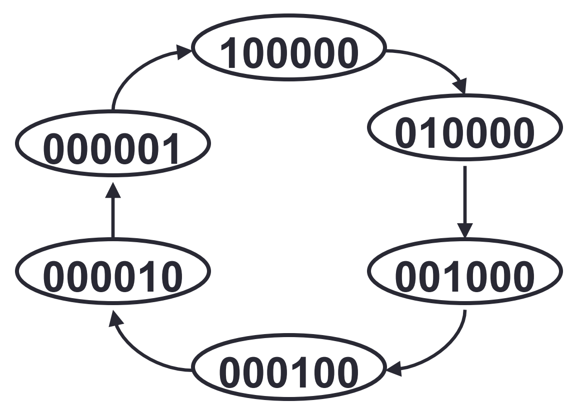
- Counters are sequential circuits that cycle through some states.
- They can be implemented using flip-flops.
- Two examples are shown: Ring counter and Johnson counter.
- Implementation is simple: using *D* flip-flops.
- (This and next few slides on ring counter and Johnson counter are just for your reading.)

RING COUNTERS

- An *n-bit* ring counter cycles through *n* states.
- Example: A 6-bit ring counter (also called mod-6 ring counter)

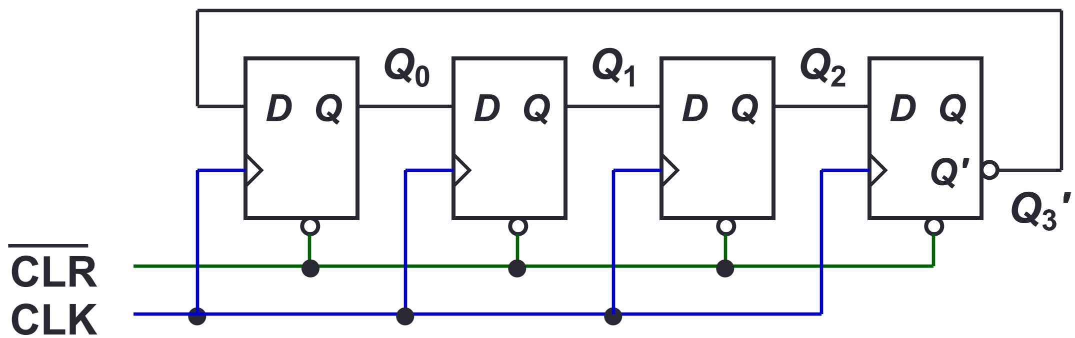


Clock	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1

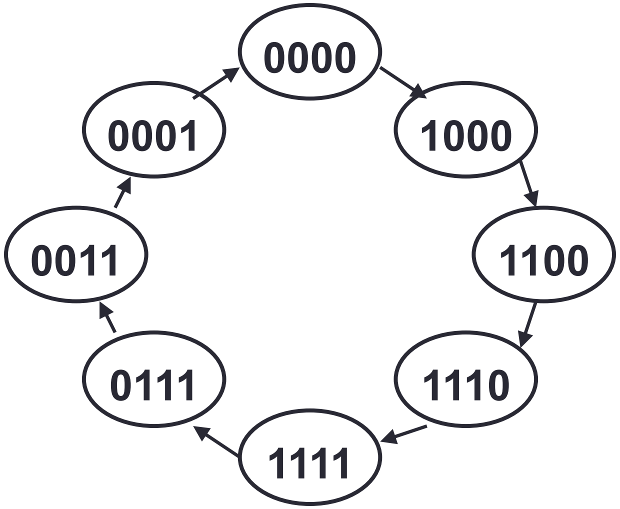


JOHNSON COUNTERS (1/2)

- An *n-bit* Johnson counter (also called twisted-ring counter) cycles through 2*n* states.
- Example: A 4-bit John counter (also called mod-8 Johnson counter)



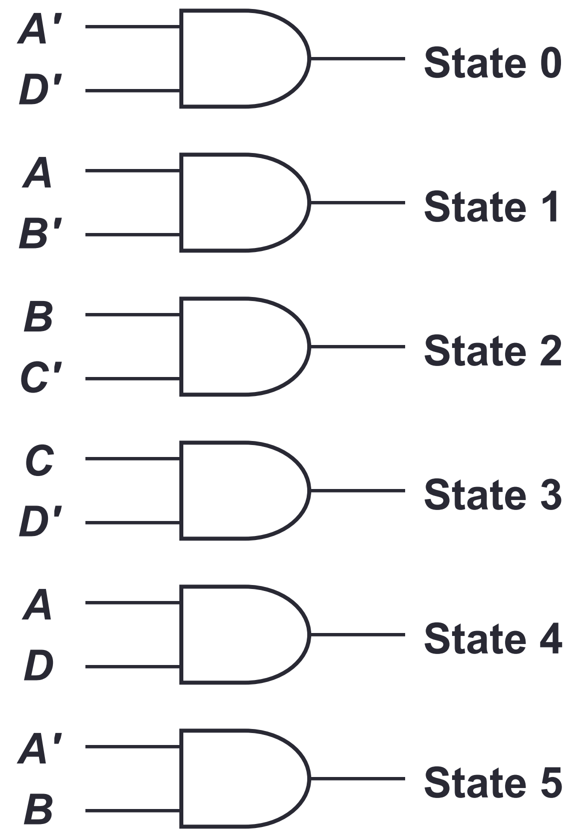
Clock	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

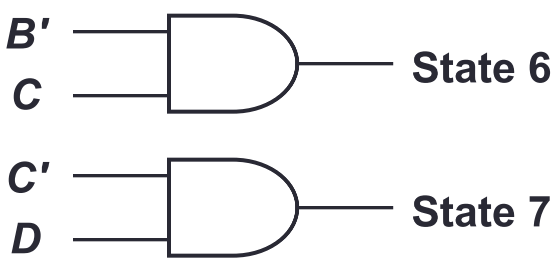


JOHNSON COUNTERS (2/2)

- Requires decoding logic for the states.
- Example: Decoding logic for a 4-bit Johnson counter.

Clock	A	B	C	D	Decoding
0	0	0	0	0	A'.D'
1	1	0	0	0	A.B'
2	1	1	0	0	B.C'
3	1	1	1	0	C.D'
4	1	1	1	1	A.D
5	0	1	1	1	A'.B
6	0	0	1	1	B'.C
7	0	0	0	1	C'.D



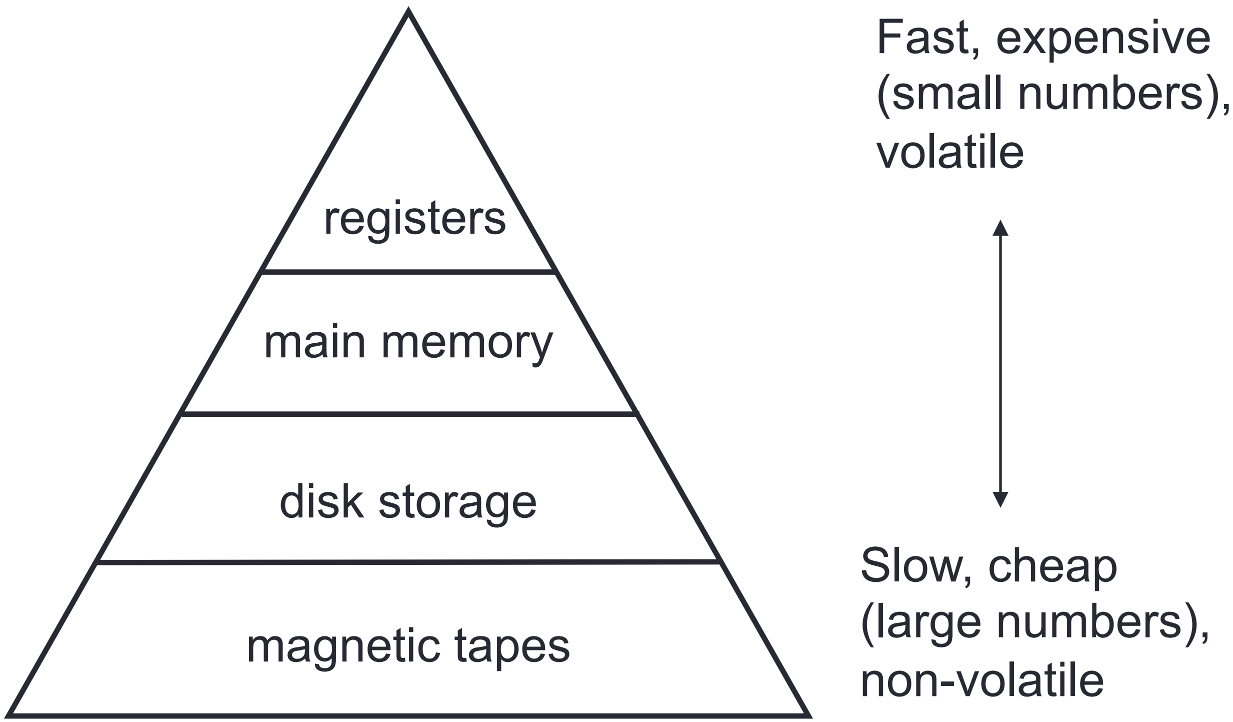


MEMORY (1/4)

- Memory stores programs and data.
- Definitions:
  - 1 byte = 8 bits
  - 1 word: in multiple of bytes, a unit of transfer between main memory and registers, usually size of register.
  - 1 KB (kilo-bytes) = 2^{10} bytes; 1 MB (mega-bytes) = 2^{20} bytes; 1 GB (giga-bytes) = 2^{30} bytes; 1 TB (tera-bytes) = 2^{40} bytes.
- Desirable properties: fast access, large capacity, economical cost, non-volatile.
- However, most memory devices do not possess all these properties.

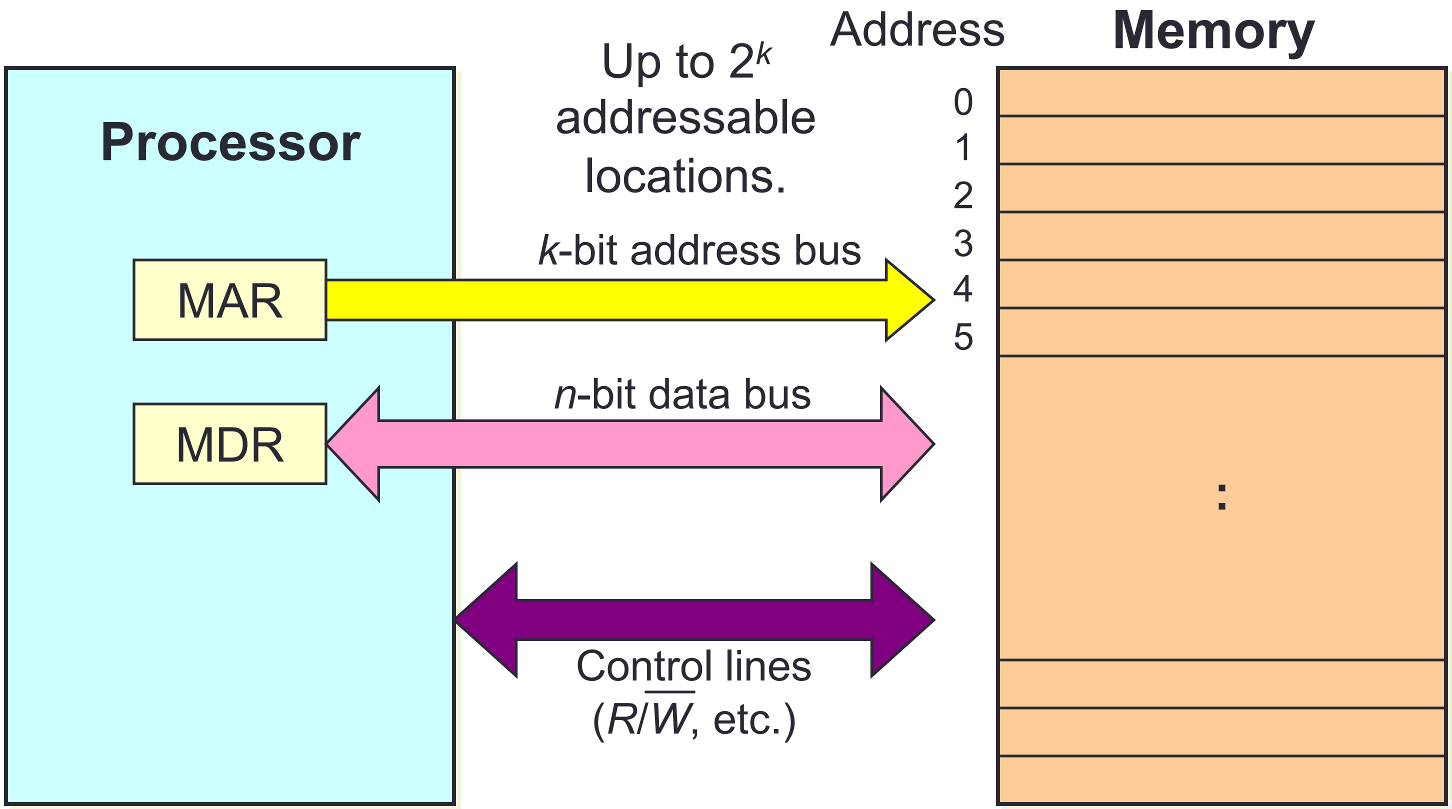
MEMORY (2/4)

- Memory hierarchy



MEMORY (3/4)

- Data transfer

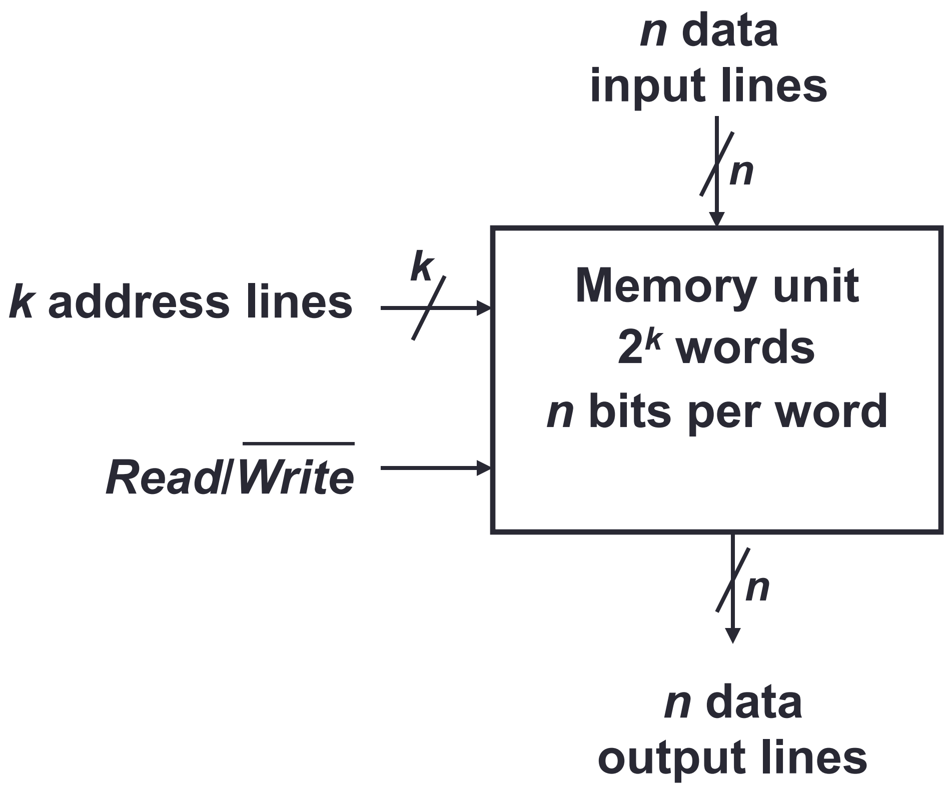


MEMORY (4/4)

- A memory unit stores binary information in groups of bits called *words*.
- The data consists of *n* lines (for *n*-bit words). Data input lines provide the information to be stored (*written*) into the memory, while data output lines carry the information out (*read*) from the memory.
- The address consists of *k* lines which specify which word (among the 2^k words available) to be selected for reading or writing.
- The control lines *Read* and $\overline{\textit{Write}}$ (usually combined into a single control line $\textit{Read}/\overline{\textit{Write}}$) specifies the direction of transfer of the data.

MEMORY UNIT

- Block diagram of a memory unit:



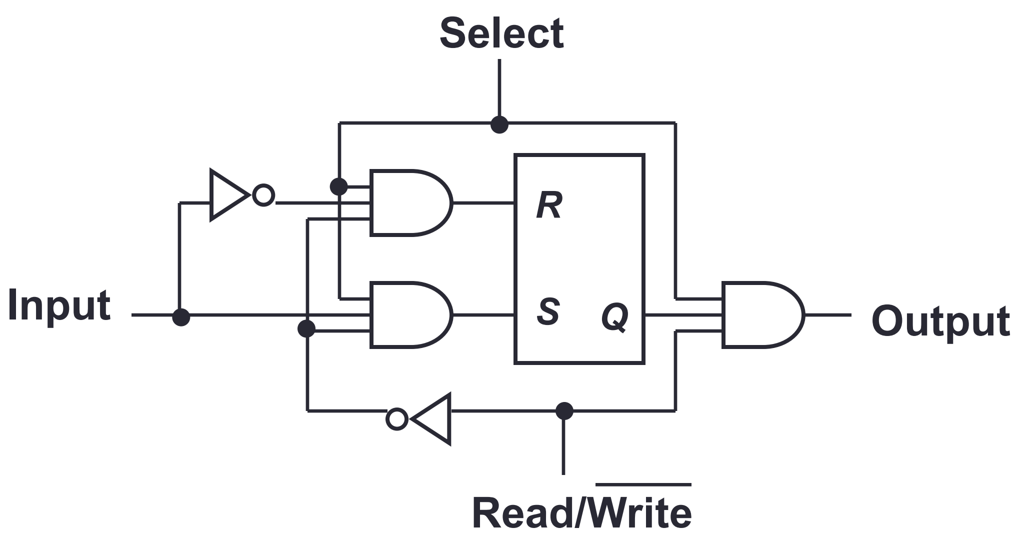
READ/WRITE OPERATIONS

- Write operation:
  - Transfers the address of the desired word to the address lines.
  - Transfers the data bits (the word) to be stored in memory to the data input lines.
  - Activates the *Write* control line (set *Read*/$\overline{\textit{Write}}$ to 0).
- Read operation:
  - Transfers the address of the desired word to the address lines.
  - Activates the *Read* control line (set *Read*/$\overline{\textit{Write}}$ to 1).

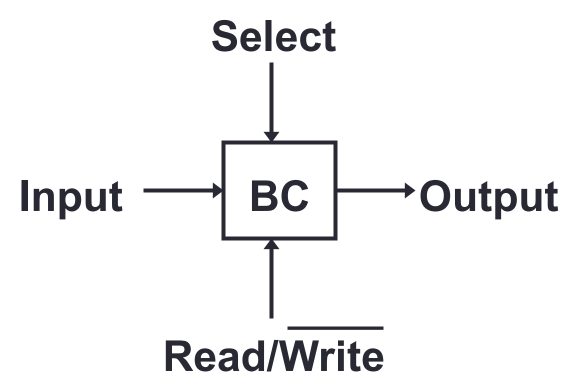
Memory Enable	*Read*/$\overline{\textit{Write}}$	Memory Operation
0	X	
1	0	
1	1	

MEMORY CELL

- Two types of RAM
  - Static RAMs use flip-flops as the memory cells.
  - Dynamic RAMs use capacitor charges to represent data. Though simpler in circuitry, they have to be constantly refreshed.
- A single memory cell of the static RAM has the following logic and block diagrams:



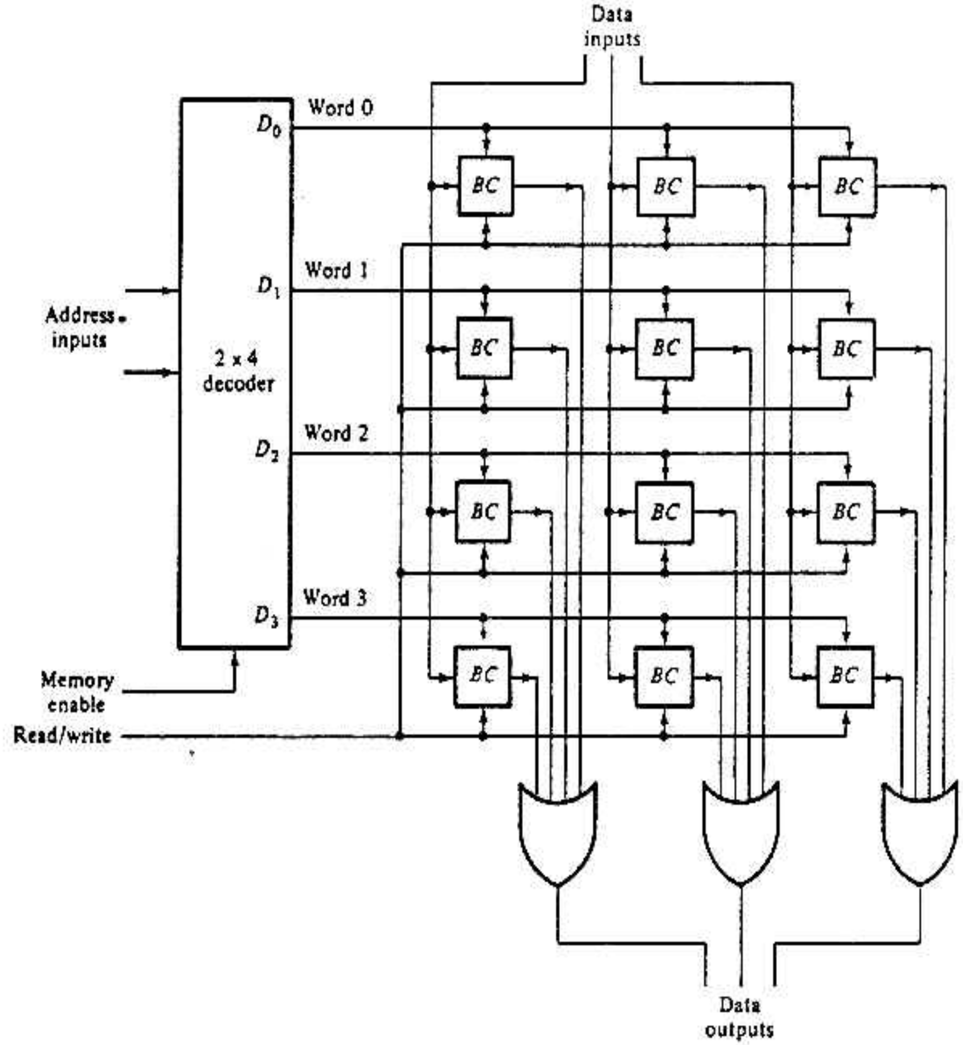
Logic diagram



Block diagram

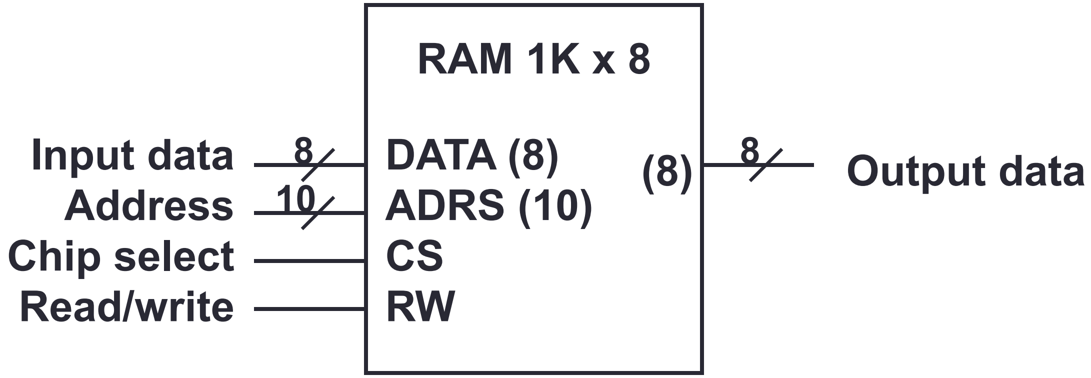
MEMORY ARRAYS (1/4)

- Logic construction of a 4×3 RAM (with decoder and OR gates):



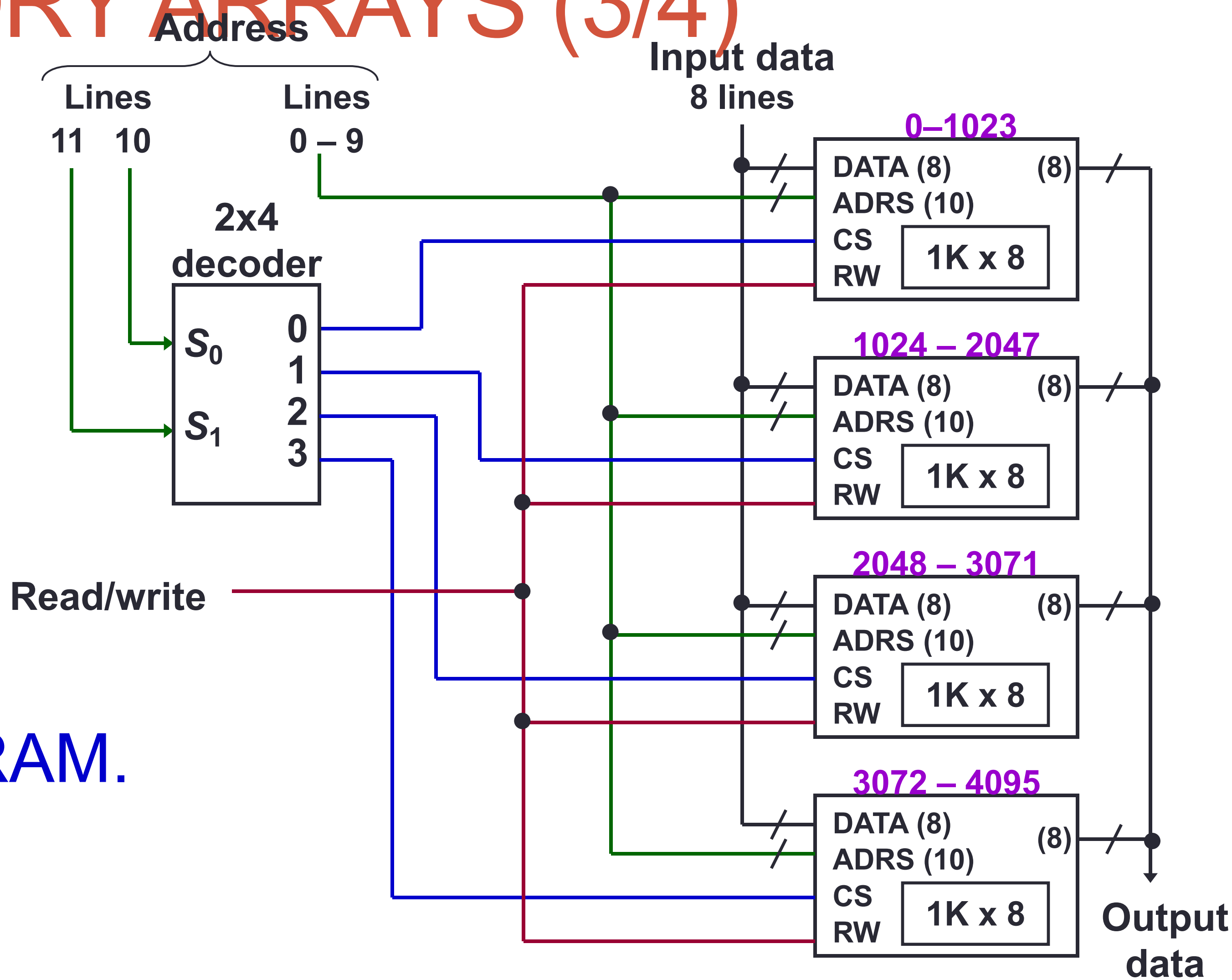
MEMORY ARRAYS (2/4)

- An array of RAM chips: memory chips are combined to form larger memory.
- A 1K × 8-bit RAM chip:



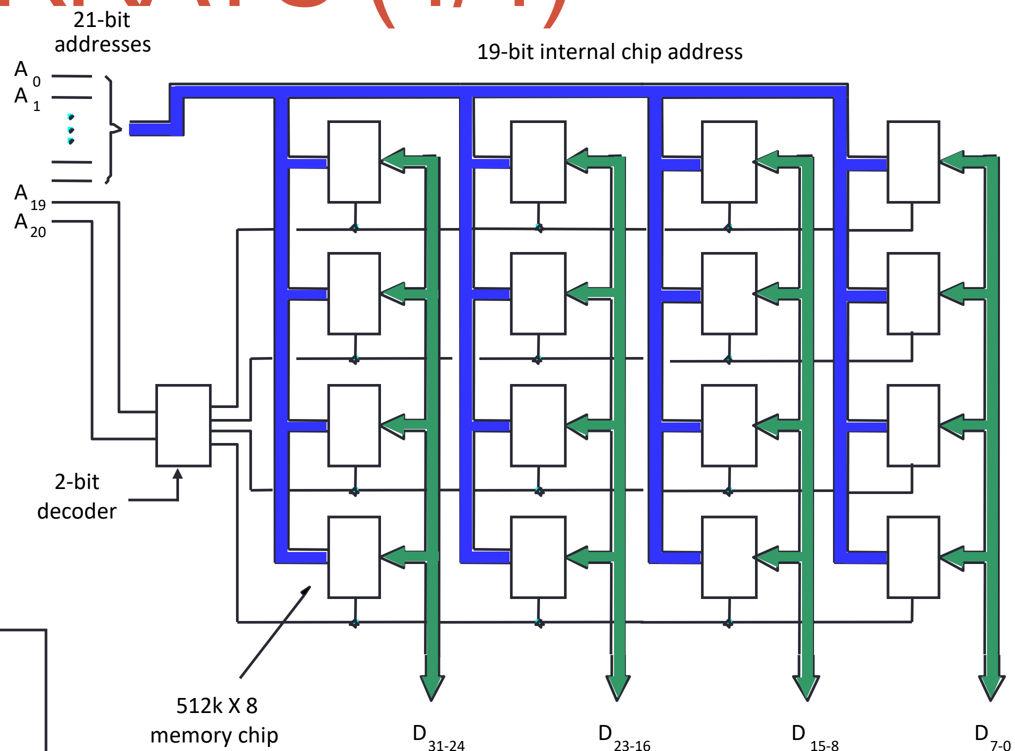
Block diagram of a 1K x 8 RAM chip

MEMORY ARRAYS (3/4)

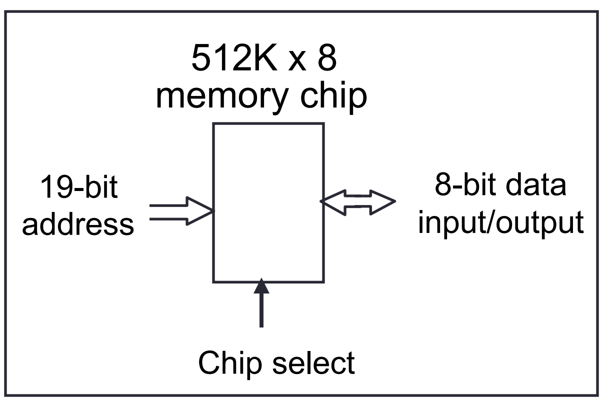


- $4K \times 8$ RAM.

MEMORY ARRAYS (4/4)



- 2M × 32 memory module
  - Using 512K × 8 memory chips.



Q&A