CS2100 COMPUTER ORGANISATION

http://www.comp.nus.edu.sg/~cs2100/

# CS2100
# Recitation 10

## Sequential Circuits

31 March 2025
(Make-up on 27 March 2025)

Aaron Tan

# Contents

- Latches
- Flip-flops
- Analysis of Sequential Circuits
- Design of Sequential Circuits
- ~~Quizzes~~
- Selected Past Years' Exam Questions
  - AY2015/16 Sem1 Q7
  - AY2020/21 Sem2 Q12

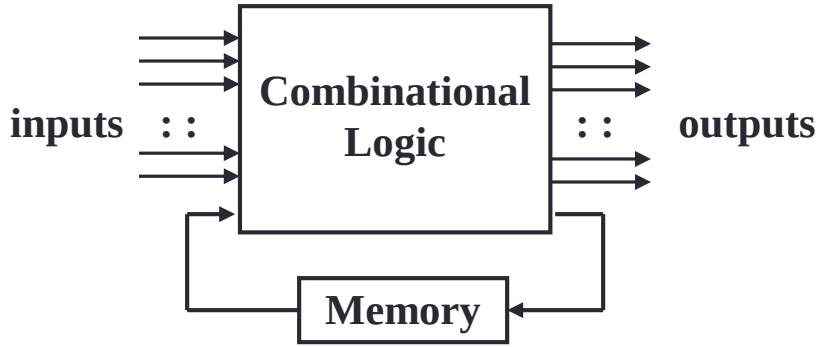
# 1. Introduction

2 weeks ago…

- 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.



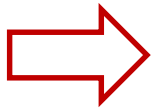

Coinadrink®
...more than just vending


50
CENTS


50
CENTS


50
CENTS


50
CENTS




Coca-Cola

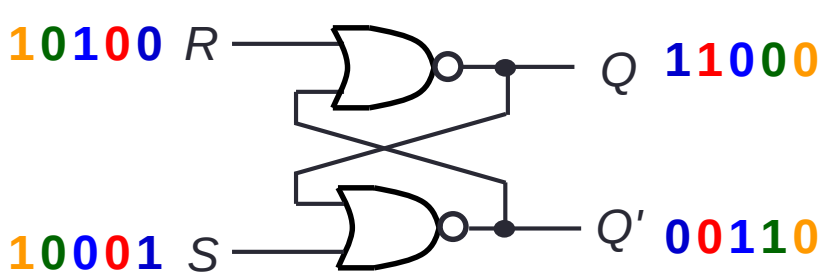
# SUMMARY

Latches

# 3.1 *S*-*R* Latch (2/3)

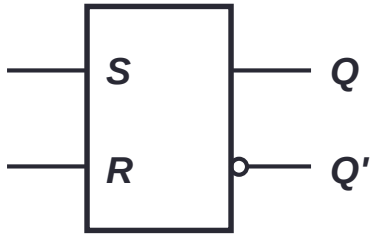
Key: When tracing a lach/flip-flop, do it 3 rounds.

- 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:

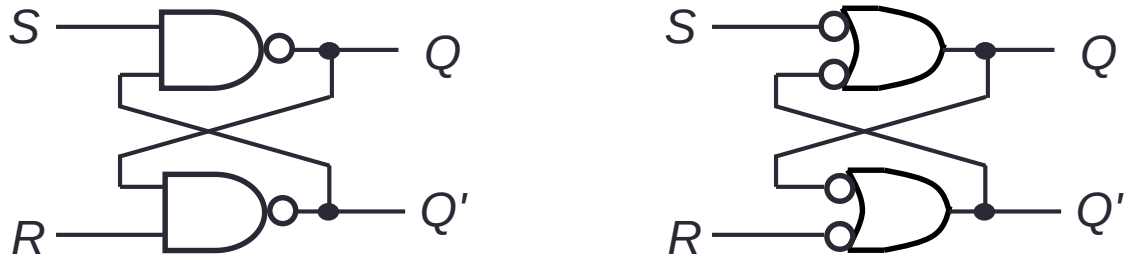



- Characteristic table:

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

# 3.1 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.



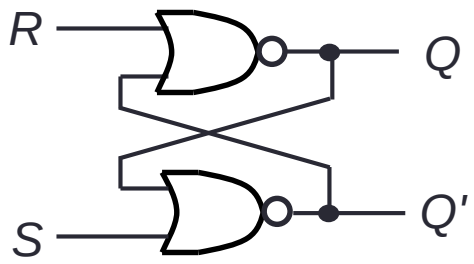

- In this case,
  - when *R*=0 and *S*=1, the latch is reset (i.e. Q becomes 0)
  - when *R*=1 and *S*=0, the latch is set (i.e. Q becomes 1)
  - when *S*=*R*=1, it is a no-change command.
  - when *S*=*R*=0, it is an invalid command.

(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.)

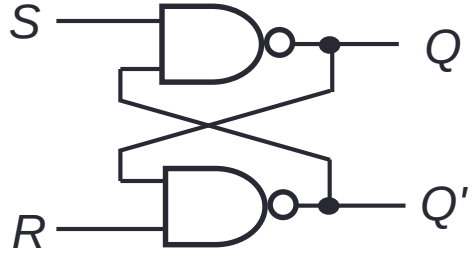
# 3.1 Active-High vs Active-Low *S-R* Latch

## (Active-high) S-R latch




| *S* | *R* | *Q*(*t*+1) | Command |
|---|---|---|---|
| 0 | 0 | *Q*(*t*) | No change |
| 0 | 1 | 0 | Reset |
| 1 | 0 | 1 | Set |
| 1 | 1 | ? | Invalid |

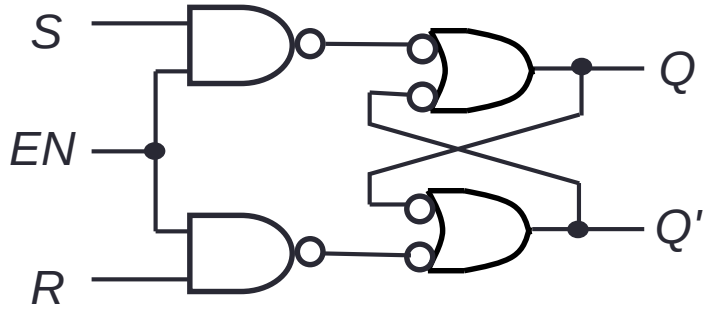
## Active-low S-R latch

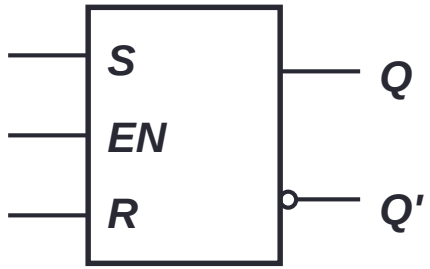



| *S* | *R* | *Q*(*t*+1) | Command |
|---|---|---|---|
| 1 | 1 | *Q*(*t*) | No change |
| 1 | 0 | 0 | Reset |
| 0 | 1 | 1 | Set |
| 0 | 0 | ? | Invalid |

# 3.1 Gated *S-R* Latch

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



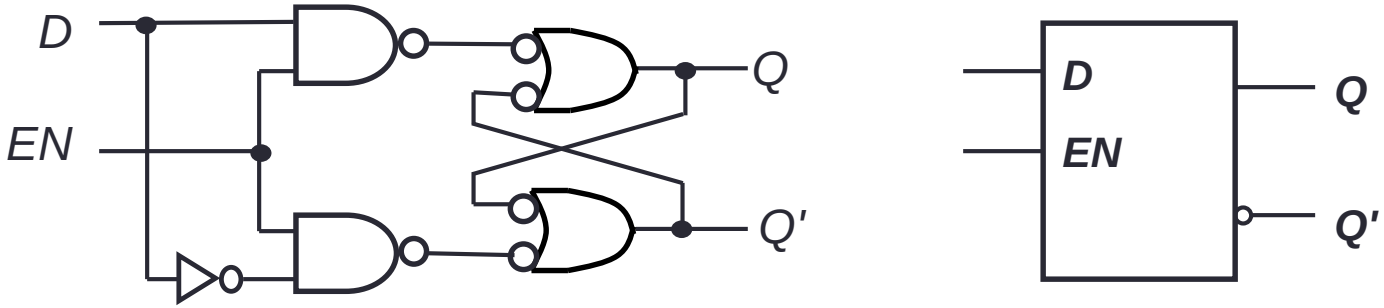




- Outputs change (if necessary) only when *EN* is high.
- Characteristic table:

| *EN* | *S* | *R* | *Q*(*t*+1) | Command |
|---|---|---|---|---|
| 1 | 0 | 0 | *Q*(*t*) | No change |
| 1 | 0 | 1 | 0 | Reset |
| 1 | 1 | 0 | 1 | Set |
| 1 | 1 | 1 | ? | Invalid |
| 0 | X | X | *Q*(*t*) | |

# 3.2 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.




- Characteristic table:

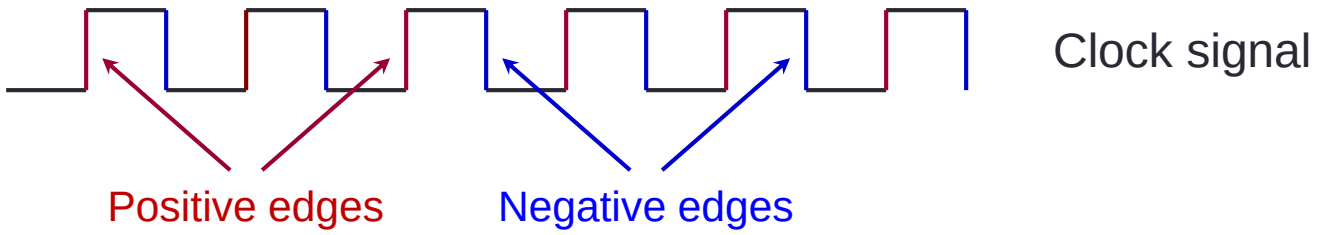
| *EN* | *D* | *Q(t+1)* | |
|---|---|---|---|
| 1 | 0 | 0 | Reset |
| 1 | 1 | 1 | Set |
| 0 | X | *Q(t)* | No change |

# SUMMARY

Flip-Flops

# 4. Flip-flops (1/2)

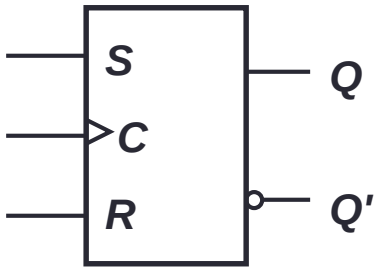
- Flip-flops are synchronous bistable devices.
- Output changes state at a specified point on a triggering input called the clock.
- Change state either at the positive (rising) edge, or at the negative (falling) edge of the clock signal.



# 4.1 *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:



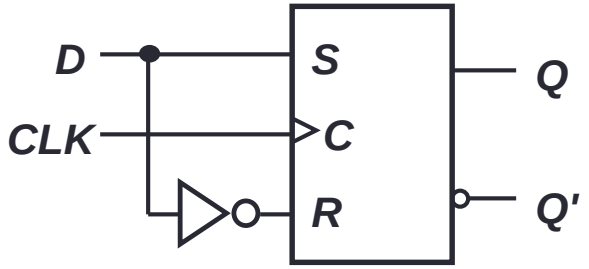

| *S* | *R* | *CLK* | *Q(t+1)* | Comments |
|---|---|---|---|---|
| 0 | 0 | X | *Q(t)* | No change |
| 0 | 1 | ↑ | 0 | Reset |
| 1 | 0 | ↑ | 1 | Set |
| 1 | 1 | ↑ | ? | Invalid |

**X = irrelevant (“don’t care”)**

**↑ = clock transition LOW to HIGH**

# 4.2 *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.



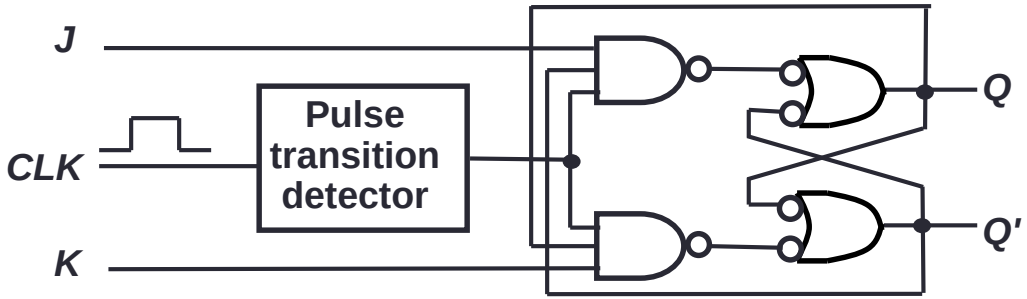

A positive edge-triggered D flip-flop formed with an S-R flip-flop.

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

↑ = clock transition LOW to HIGH

# 4.3 *J-K* Flip-flop (2/2)

- *J-K* flip-flop circuit:




- Characteristic table:

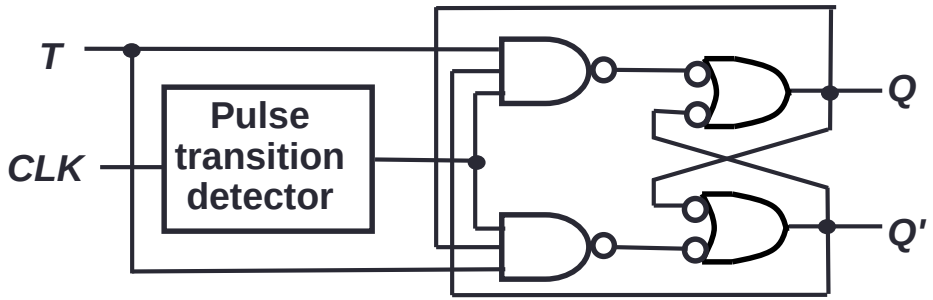
| *J* | *K* | *CLK* | *Q(t+1)* | Comments |
|---|---|---|---|---|
| 0 | 0 | ↑ | *Q(t)* | No change |
| 0 | 1 | ↑ | 0 | Reset |
| 1 | 0 | ↑ | 1 | Set |
| 1 | 1 | ↑ | *Q(t)'* | Toggle |

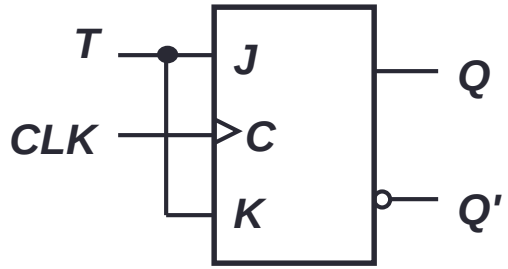
| *Q* | *J* | *K* | *Q(t+1)* |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$$Q(t+1) = J \cdot Q' + K' \cdot Q$$

# 4.4 *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 |

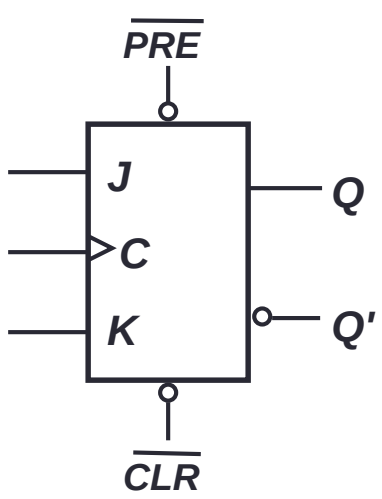
$$Q(t+1) = T \cdot Q' + T' \cdot Q$$

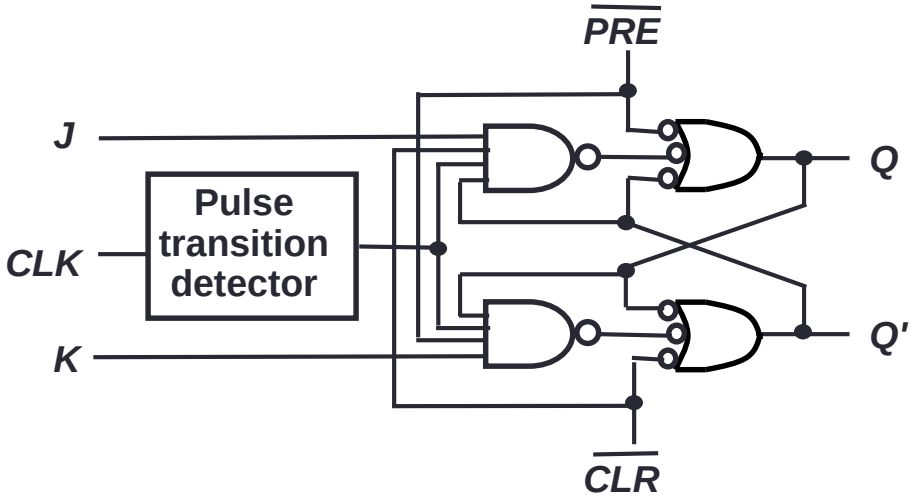
# 5. Asynchronous Inputs (1/2)

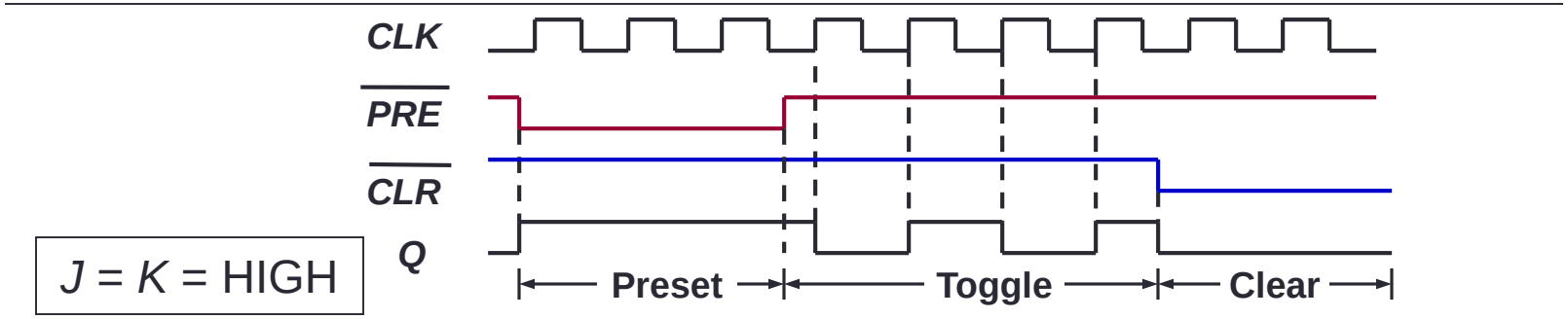
- *S-R*, *D* and *J-K* inputs are synchronous inputs, as data on these inputs are transferred to the flip-flop's output only on the triggered edge of the clock pulse.
- Asynchronous inputs affect the state of the flip-flop independent of the clock; example: *preset* (*PRE*) and *clear* (*CLR*) [or *direct 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.

# 5. Asynchronous Inputs (2/2)

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



# COMMON QUESTIONS

From previous semester

Q1: How does slide 9 of lect 19 work? Why is it that both R and S are high considered an invalid input?

A: It has been illustrated that when both R and S are high, the output Q and Q' are the same, which violates the definition of a latch/flip-flop.

Q2: Does Q(t+1), Q+ and Q' all mean the same thing?

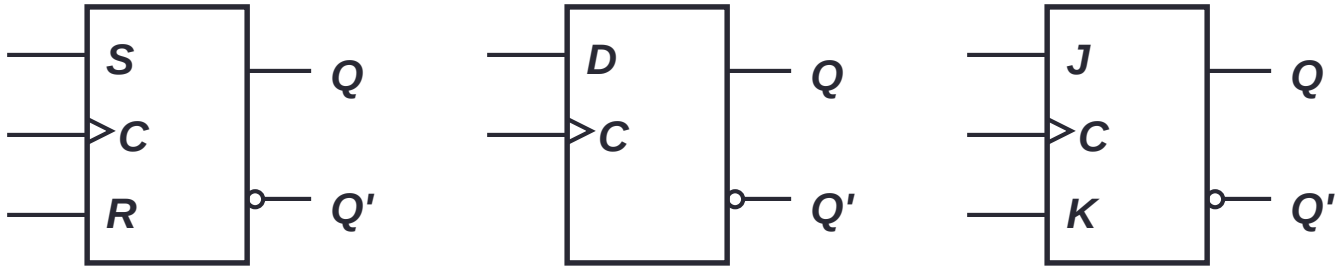
A: Q(t) and Q(t+1) mean present state and next state respectively. They can also be referred to simply as Q and Q+ respectively. So, Q = Q(t), and Q+ = Q(t+1).

Q' is the complement of Q. Therefore, Q' is not the same as Q (in fact, it is the negation/opposite of Q), nor is it the same as Q+ (it has no relationship to Q+).

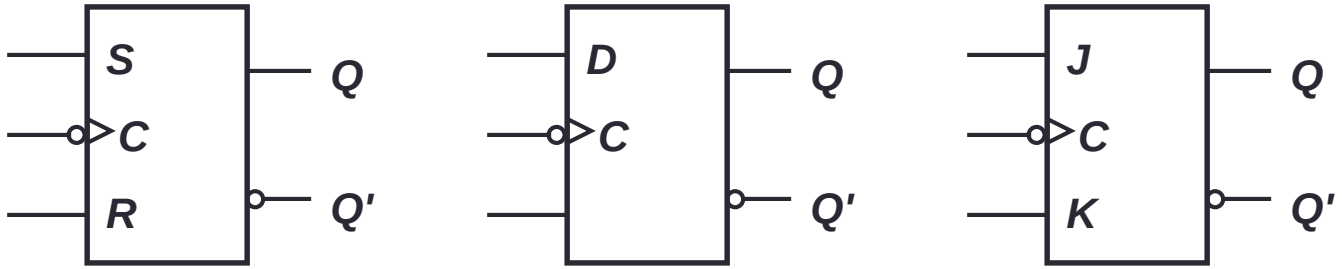
Q3: L19 slide 10, the outputs are redirected as inputs, and to get outputs we need inputs first. How does the whole thing work as it seems like a never-ending cycle.

A: If there isn't a clock, will be a "never-ending cycle". The clock provides synchronisation. The flip flop acts only when it is activated (by the rising edge or falling edge of the clock).

# Q4: Why is there a circle at Q' output on the flip flop?



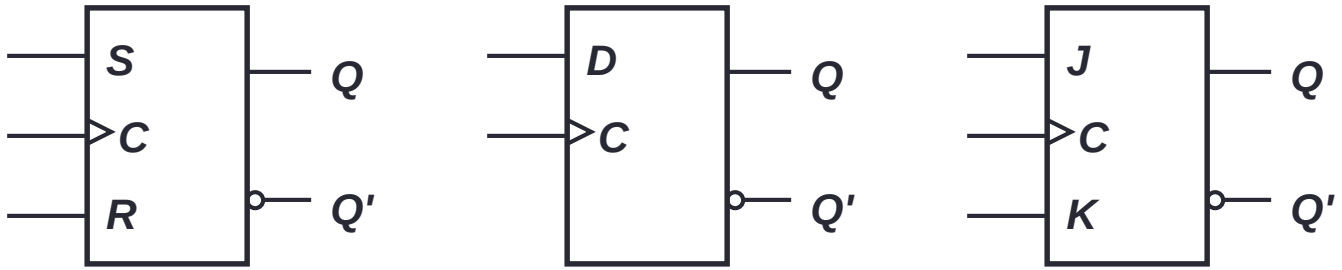

Positive edge-triggered flip-flops




Negative edge-triggered flip-flops

A: Just a labelling convention, to remind users that the output (Q') is the negation of Q.

Q5: What do Q and Q’ represent on the flip flop?




Positive edge-triggered flip-flops

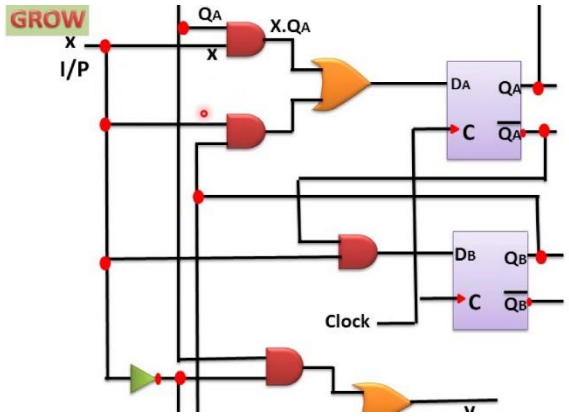
A: Q is the state of the flip-flop and Q’ is the negation of Q.

A flip-flop is a one-bit memory unit; when it stores a 0 it is said to be in the reset state; when it stores a 1 it is in the set state. The Q output of a flip-flop determines the state it is in.

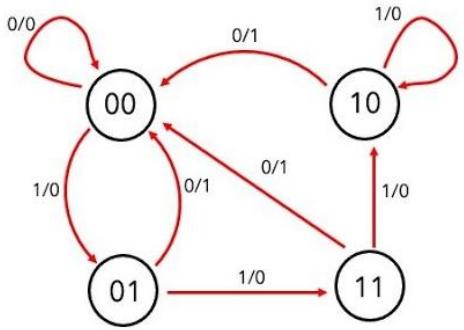
# Analysis

# Design

Uses flip-flop characteristic tables.




Logic circuit




State diagram

Uses flip-flop excitation tables.

Just for illustration. The state diagram here does not correspond to the logic circuit.

# SUMMARY

## Analysis of Sequential Circuits

# 6.1 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 |

# 6.2 Analysis: Example #2 (1/3)

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

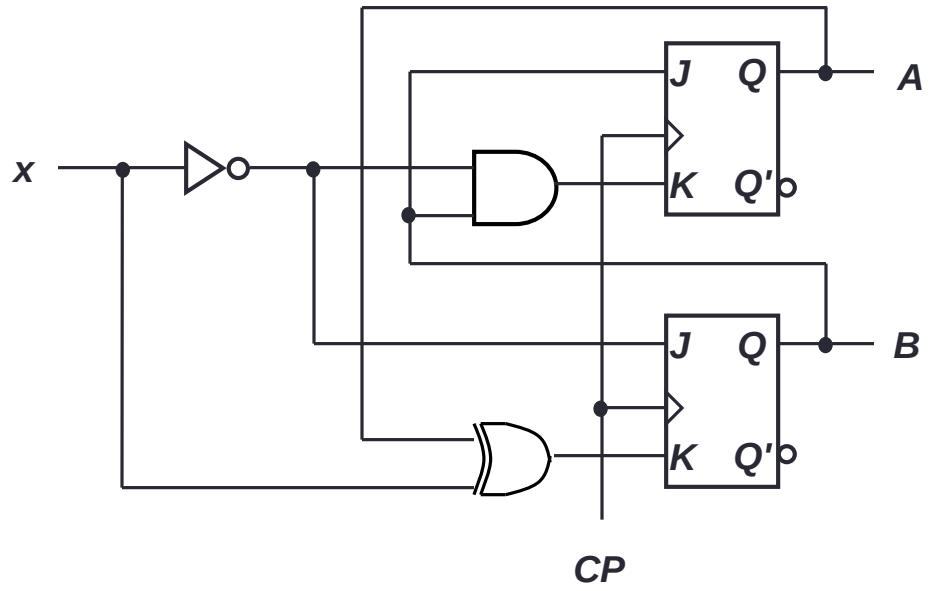



Figure 2

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

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

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

# 6.2 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.

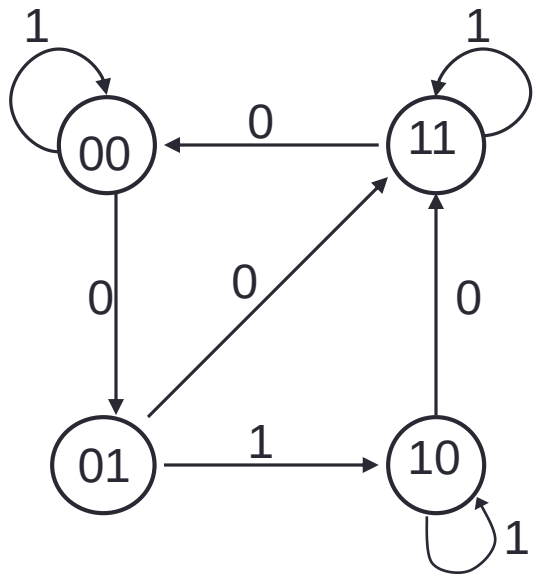
| J | K | Q(t+1) | Comments |
|---|---|---|---|
| 0 | 0 | Q(t) | No change |
| 0 | 1 | 0 | Reset |
| 1 | 0 | 1 | Set |
| 1 | 1 | Q(t)' | Toggle |

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

# 6.2 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 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |



# SUMMARY

## Design of Sequential Circuits

# 6.3 Flip-flop Excitation Tables (1/2)

- *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 |

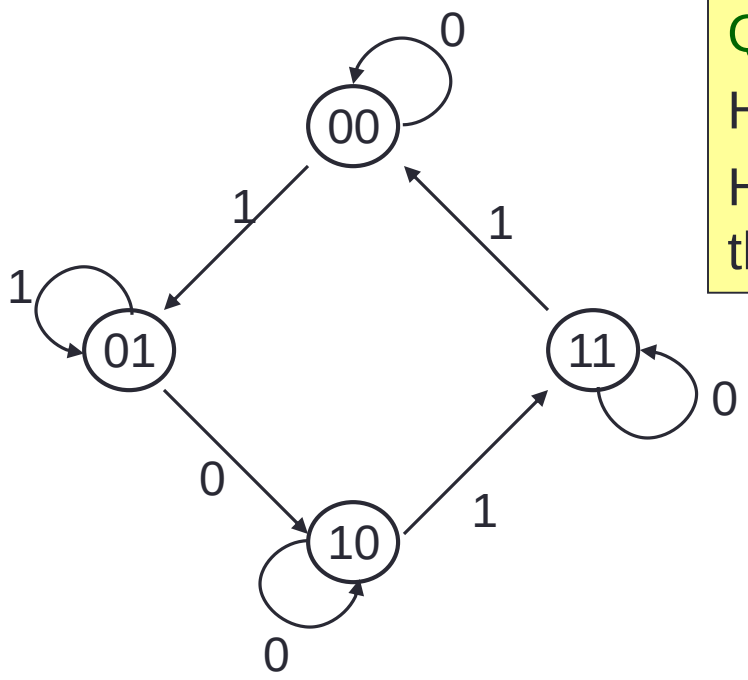
*D* Flip-flop

| $Q$ | $Q^+$ | $T$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

*T* Flip-flop

# 6.4 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?

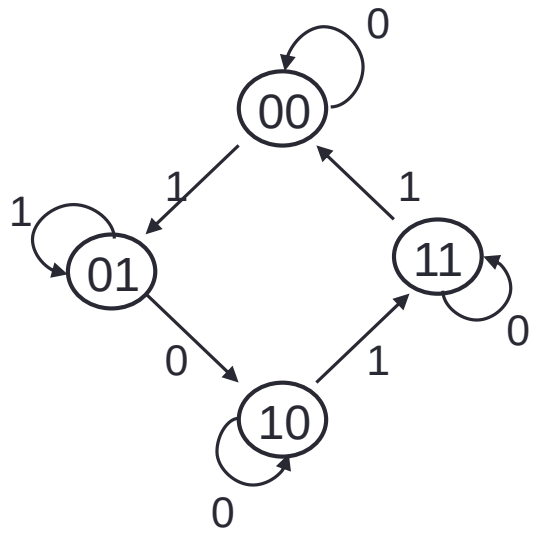
Answers:

Two flip-flops.
Let's call them *A* and *B*.

One input variable.
Let's call it *x*.

# 6.4 Design: Example #1 (2/5)

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




| $Q$ | $Q^+$ | $J$ | $K$ |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

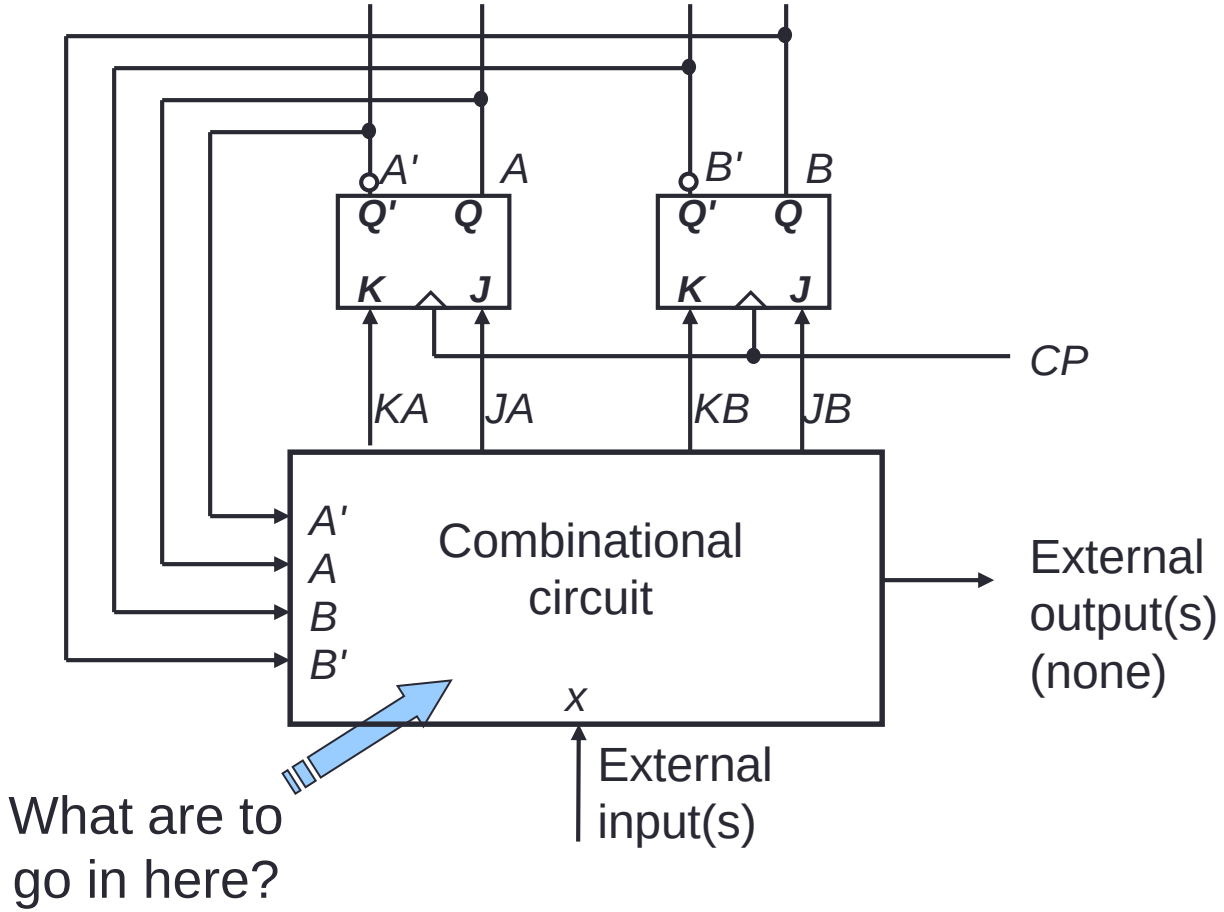
*JK* Flip-flop's excitation table.

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

| 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 |

# 6.4 Design: Example #1 (3/5)

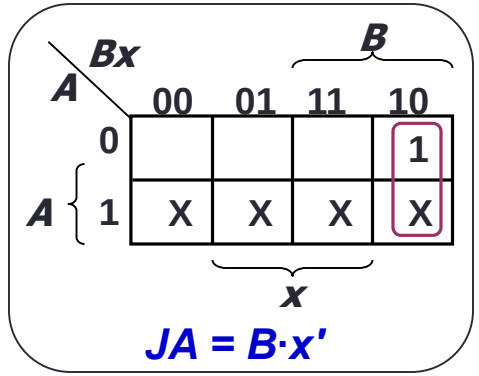
- Block diagram.

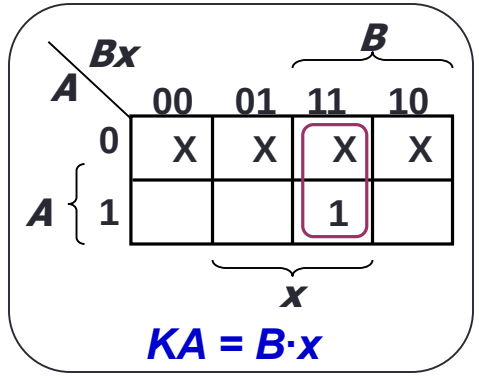


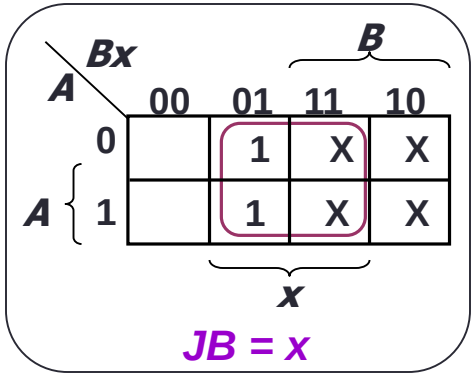
# 6.4 Design: Example #1 (4/5)

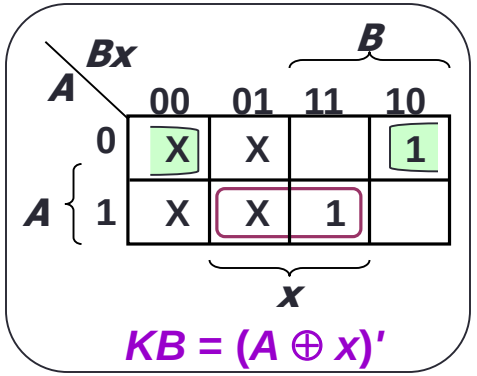
- From state table, get flip-flop input functions.

| 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 |



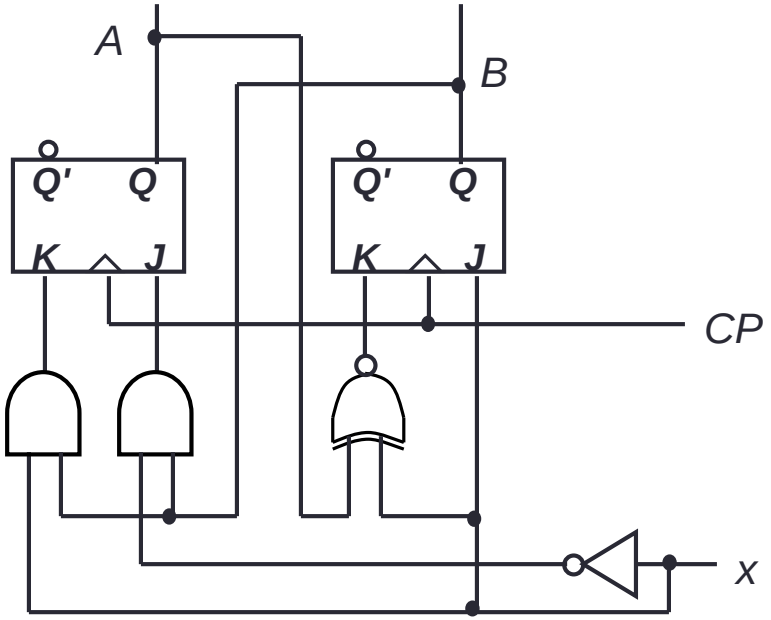
# 6.4 Design: Example #1 (5/5)

- Flip-flop input functions:

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

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

- Logic diagram:



# 6.4 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

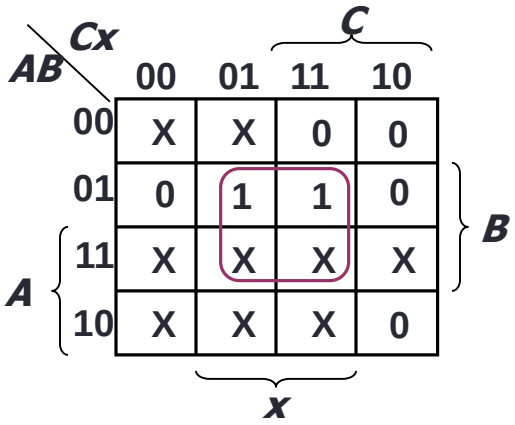
Are there other unused states?

Unused state 000:

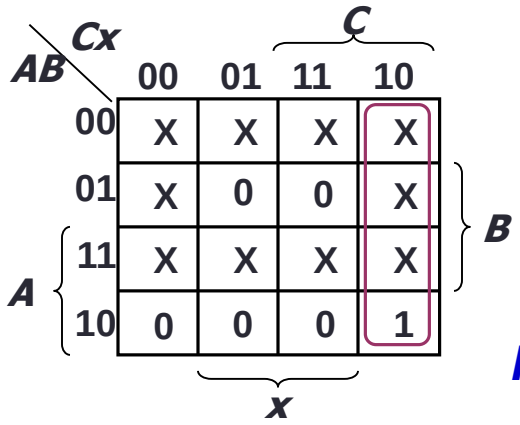
| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | X | X | X |
| 0 | 0 | 0 | 1 | X | X | X | X | X | X | X | X | X | X |

# 6.4 Design: Example #3 (2/4)

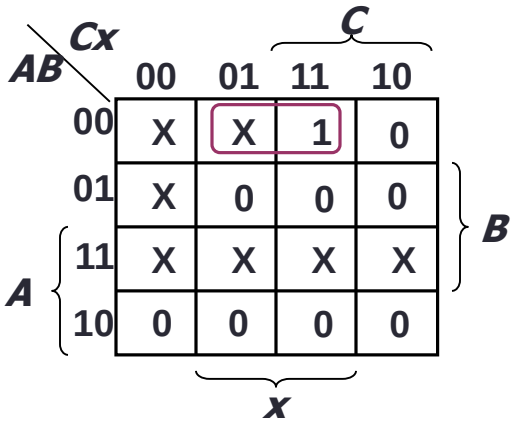
- From state table, obtain expressions for flip-flop inputs.



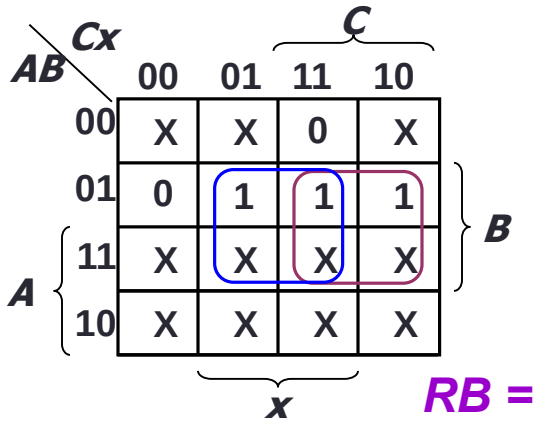

$SA = B \cdot x$




$RA = C \cdot x'$



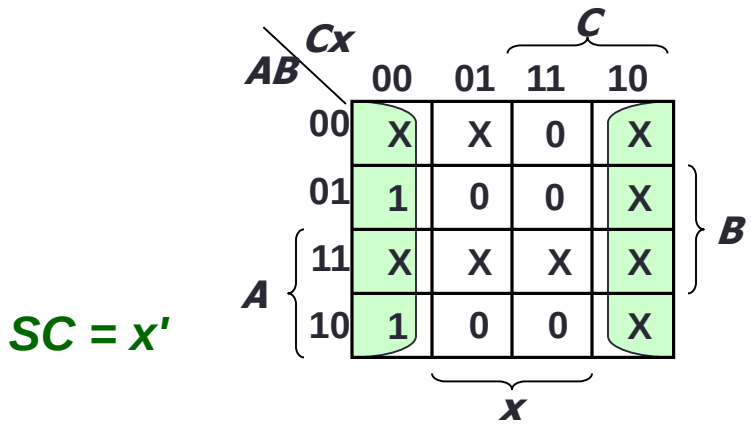

$SB = A' \cdot B' \cdot x$



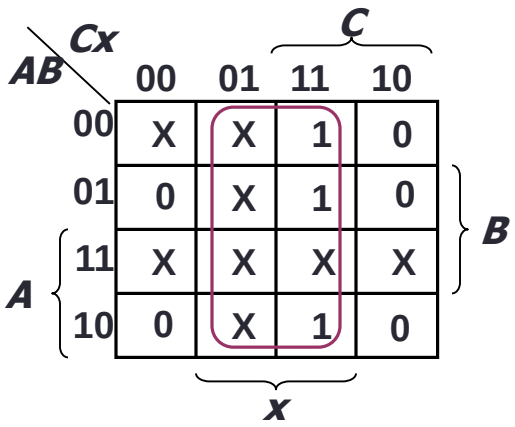

$RB = B \cdot C + B \cdot x$

# 6.4 Design: Example #3 (3/4)

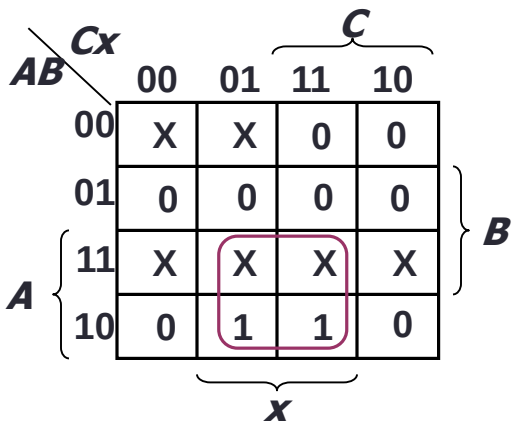
- From state table, obtain expressions for flip-flop inputs (cont'd).




$SC = x'$




$RC = x$

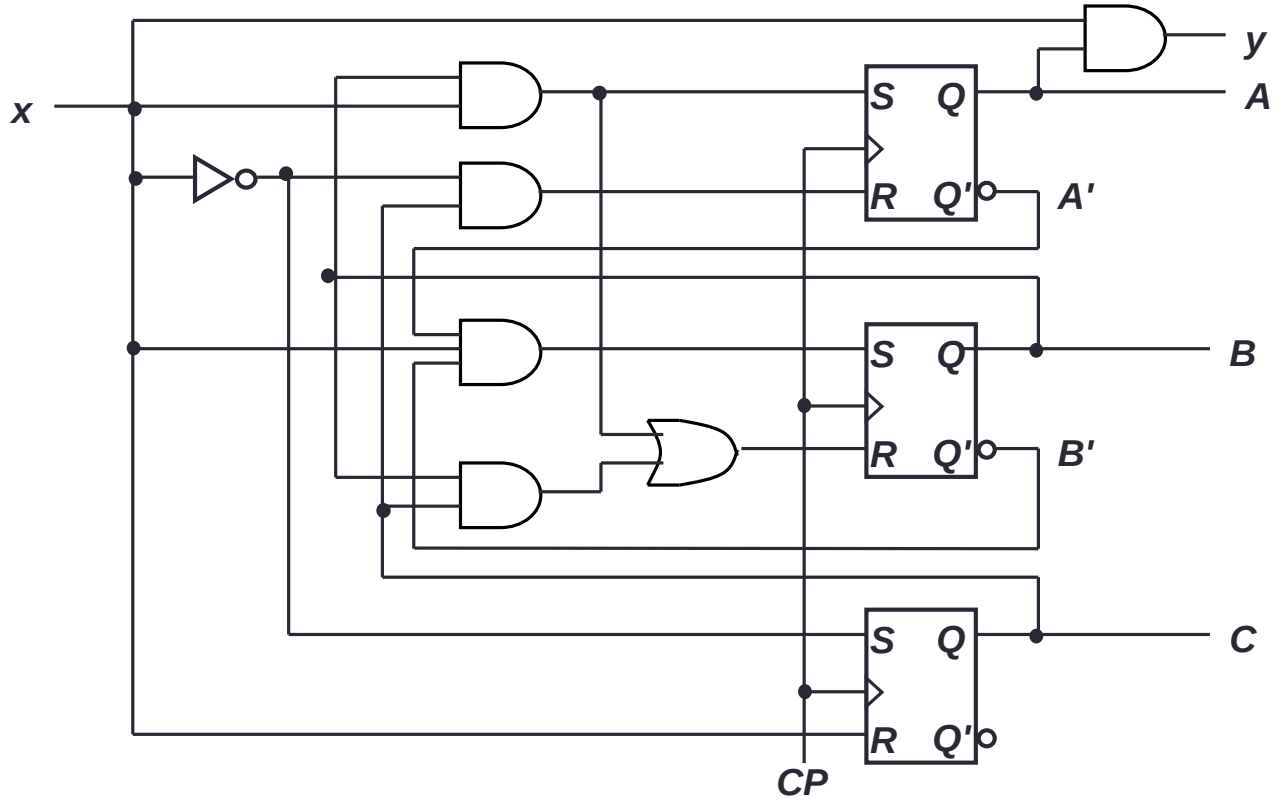



$y = A \cdot x$

# 6.4 Design: Example #3 (4/4)

- From derived expressions, draw the logic diagram:

| | | | |
|---|---|---|---|
| $SA = B \cdot x$ | $SB = A' \cdot B' \cdot x$ | $SC = x'$ | $y = A \cdot x$ |
| $RA = C \cdot x'$ | $RB = B \cdot C + B \cdot x$ | $RC = x$ | |

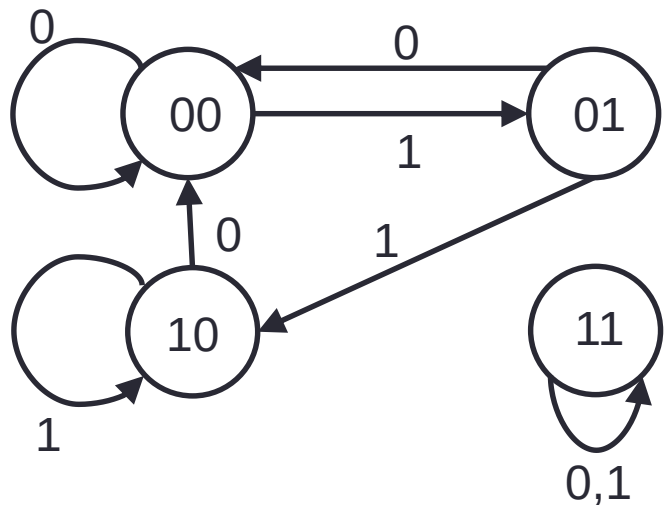


# Self-Correcting Circuits (1/4)

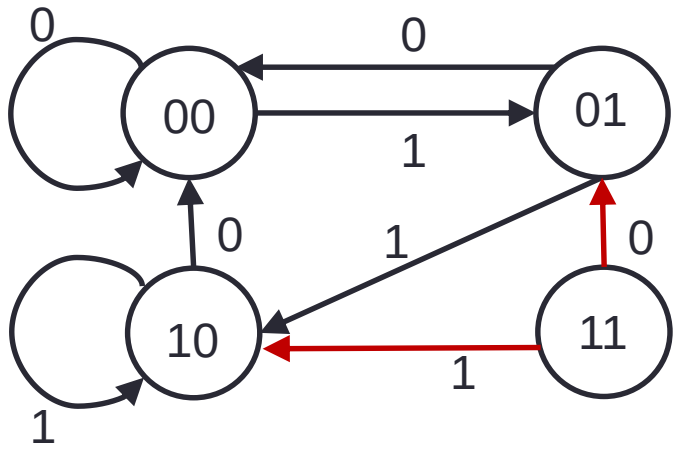
- When a circuit has unused states, these unused states are considered to be “invalid”.
- Suppose we have a circuit with 2 flip-flops where states 0 to 2 are used. State 3 is unused and considered to be “invalid”.
- On the following pages we see two possible state diagrams for this circuit.

# Self-Correcting Circuits (2/4)

- Case 1: Non-self-correcting Circuit:
  - If the circuit finds itself in state 3 (e.g. due to a glitch), it will never transit into a valid state (state 0, 1, or 2).




- Case 2: Self-Correcting Circuit
  - A circuit that will (eventually) find its way back to a valid state.
  - Here even if the circuit finds itself in state 3, it will transit to a valid state (either state 1 or 2)



# Self-Correcting Circuits (3/4)

- Is Design Example #3 self-correcting?
  - We recreate the state table for the invalid states and see if they lead to any valid states:

SA = B.x
SB = A'.B'.x
SC = x'

RA = C.x'
RB = B.C + B.x
RC = x

Valid states: 1, 2, 3, 4, 5.
Invalid states: 0, 6, 7.

(y is omitted since we don't need it to check for self-correction)

| A | B | C | X | SA | RA | SB | RB | SC | RC | A+ | B+ | C+ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

# Self-Correcting Circuits (4/4)

- Is Design Example #3 self-correcting?
  - The transitions for the invalid states is summarized below:

| ABC | (ABC)+ | |
|---|---|---|
| | x=0 | x=1 |
| 000 | 001 | 010 |
| 110 | 111 | 100 |
| 111 | 001 | 100 |

Valid states: 1, 2, 3, 4, 5.
Invalid states: 0, 6, 7.

  - We can see that every invalid state eventually transits to some valid state, hence this circuit is **self-correcting**.

QnA Q2:

Some of the tutorial qns require very long truth table + kmap which I feel is almost impossible to draw quickly during exam.

Any tips to be faster?

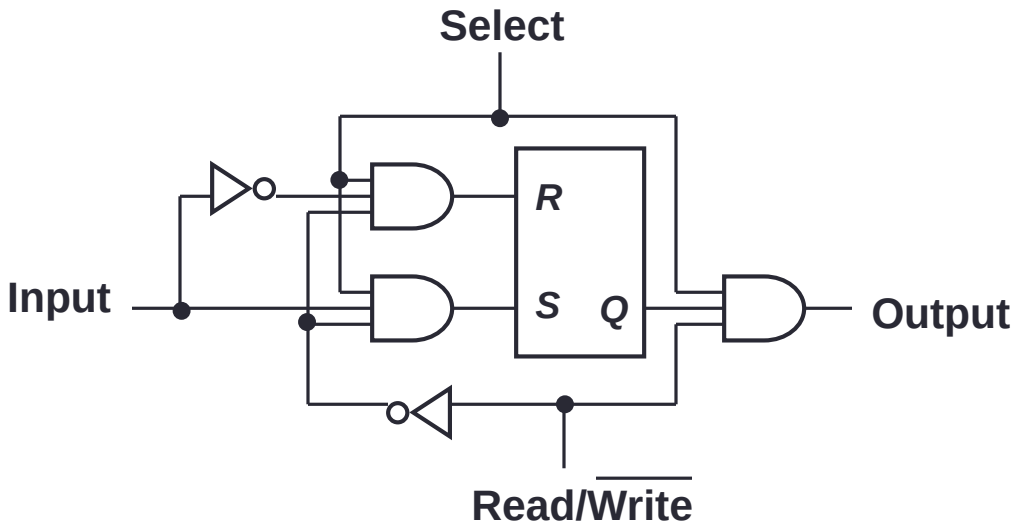
# QNA TOPIC 8

From previous semester:

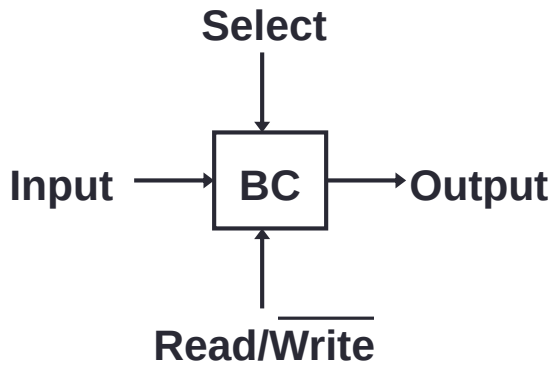
Q7. hi prof, can you go through the memory array portion the 4k x 8 ram portion

# 7.3 Memory Cell

- A single memory cell of the static RAM has the following logic and block diagrams:




Logic diagram

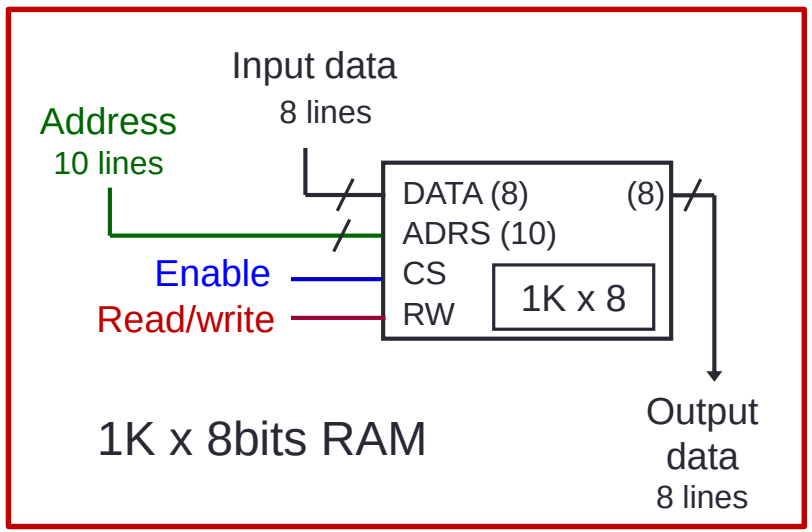


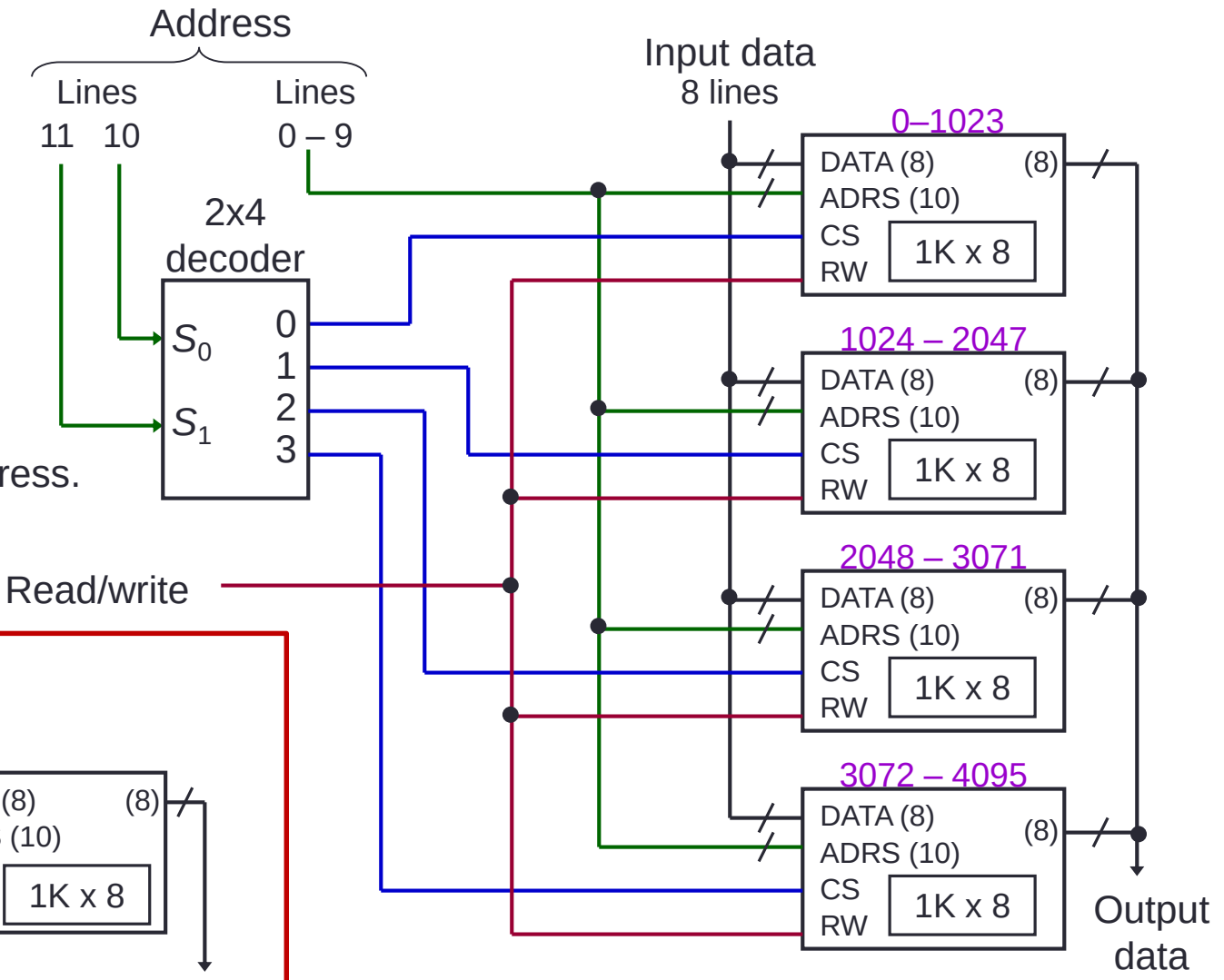

Block diagram

# 7.4 Memory Arrays (3/4)

Building a 4K x 8bits RAM using 1K x 8bits RAMs.

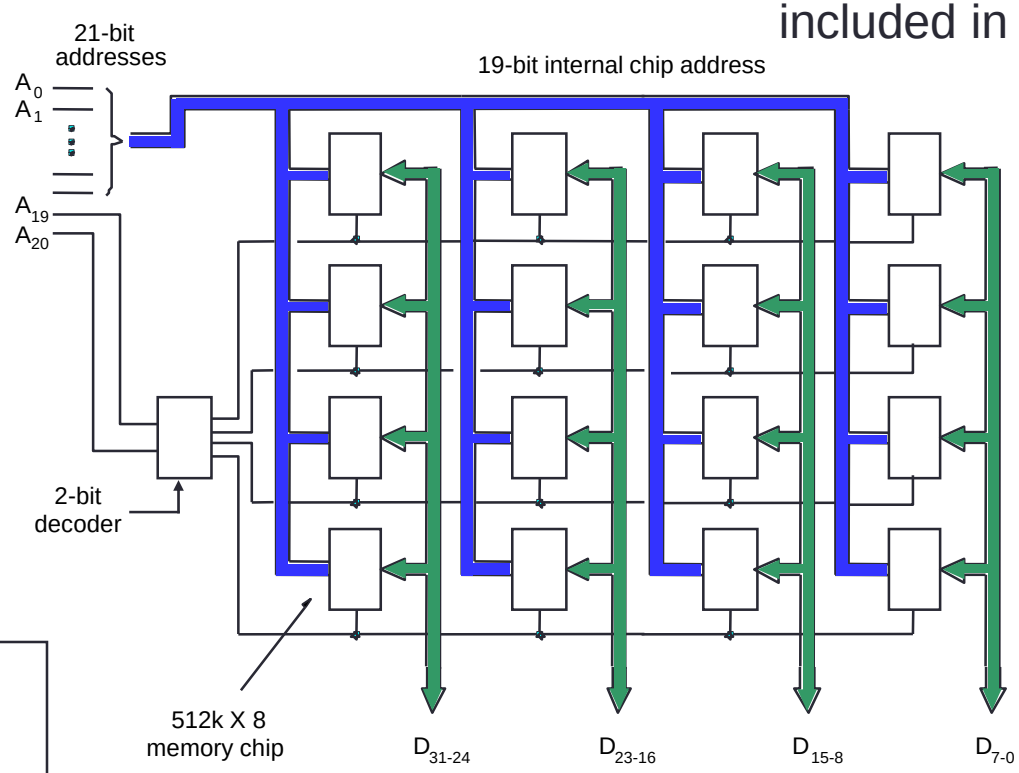
4K locations →12-bit address.

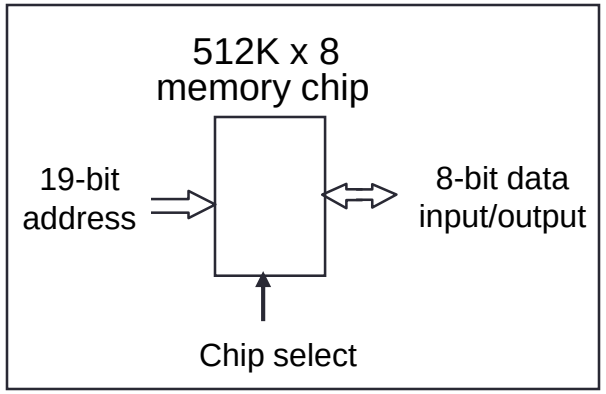


# 7.4 Memory Arrays (4/4)

Read/write control line not included in this diagram.




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

# PAST YEARS' QUESTIONS

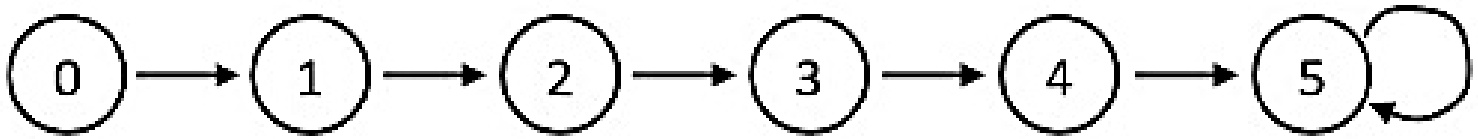
Sequential Circuits

- AY2015/16 Sem1 Q7
- AY2020/21 Sem2 Q12

# AY2015/16 Sem1 Q7

7. **[8 marks]**

A sequential circuit goes through the following states, whose state values are shown in decimal, as shown below:




The states are represented by 3-bit values *ABC*. Implement the sequential circuit using a *JK* flip-flop for *A*, a *T* flip-flop for *B*, and a *D* flip-flop for *C*.

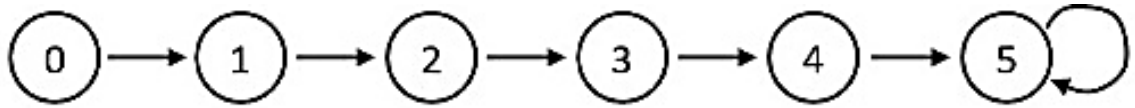
a. Write out the **simplified SOP expressions** for all the flip-flop inputs. Note that the simplified expression for *KA* has been done for you (*KA* = 0). [3 marks]

b. Complete the logic diagram on the answer booklet, by adding one inverter and a minimum number of logic gates of another type. [2 marks]

c. Complete the given state diagram on the answer booklet, by indicating the next state for each of the two unused states. [2 marks]

d. Is the circuit self-correcting? Explain your answer. (No mark will be awarded if there is no explanation or the explanation is wrong.) [1 mark]

# AY2015/16 Sem1 Q7



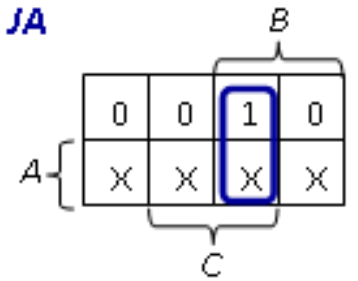

The states are represented by 3-bit values *ABC*. Implement the sequential circuit using a *JK* flip-flop for *A*, a *T* flip-flop for *B*, and a *D* flip-flop for *C*.

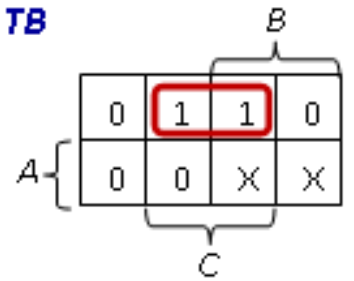
a. Write out the **simplified SOP expressions** for all the flip-flop inputs. Note that the simplified expression for *KA* has been done for you (*KA* = 0). [3 marks]

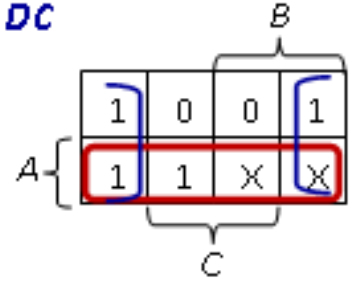
| $Q$ | $Q^+$ | $J$ | $K$ |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

*JK* flip-flop excitation table

| A | B | C | $A^+$ | $B^+$ | $C^+$ | JA | KA | TB | DC |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | X | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | X | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | X | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | X | 0 | 0 | 1 |
| 1 | 1 | 0 | X | X | X | X | X | X | X |
| 1 | 1 | 1 | X | X | X | X | X | X | X |




$JA = B \cdot C$

$TB = A' \cdot C$

$DC = A + C'$

$KA = 0$

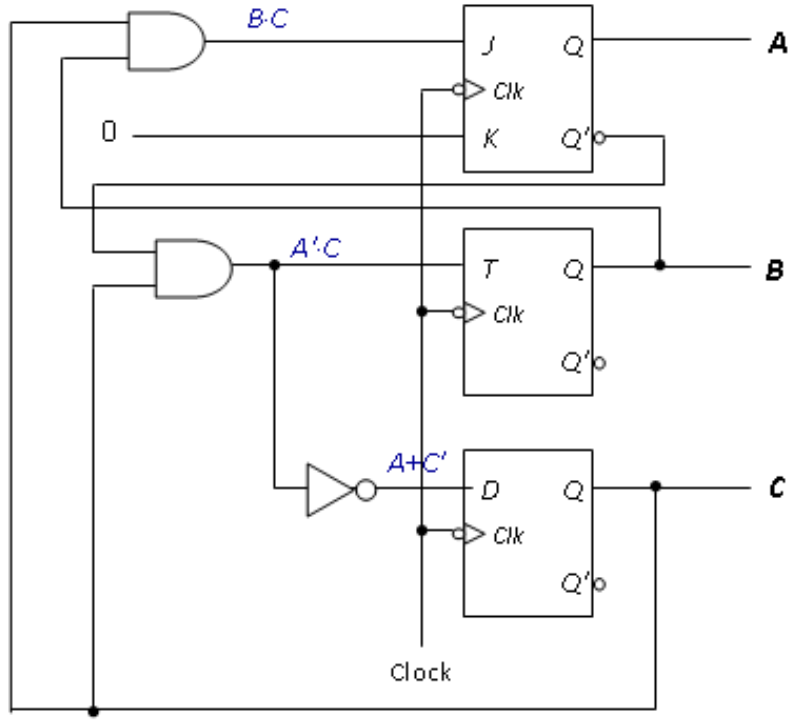
# AY2015/16 Sem1 Q7

b. Complete the logic diagram on the answer booklet, by adding one inverter and a minimum number of logic gates of another type. [2 marks]

$JA = B \cdot C$ $TB = A' \cdot C$ $DC = A + C'$

$KA = 0$

| A | B | C | $A^+$ | $B^+$ | $C^+$ | JA | KA | TB | DC |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | X | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | X | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | X | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | X | 0 | 0 | 1 |
| 1 | 1 | 0 | X | X | X | X | X | X | X |
| 1 | 1 | 1 | X | X | X | X | X | X | X |

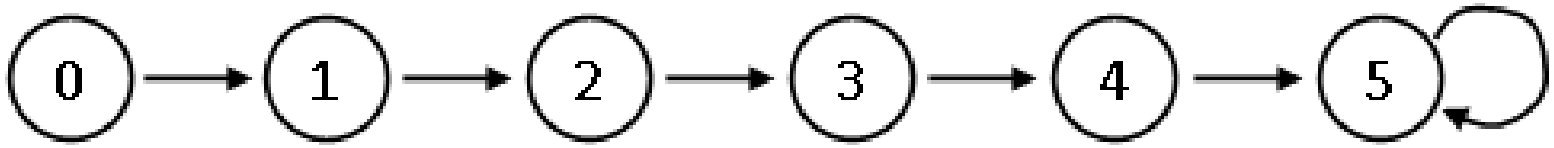


# AY2015/16 Sem1 Q7

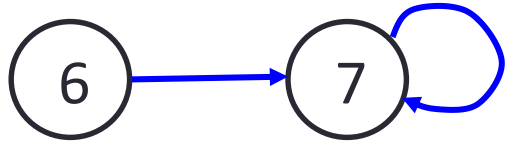
c. Complete the given state diagram on the answer booklet, by indicating the next state for each of the two unused states. [2 marks]

*JA = B·C* *TB = A'·C* *DC = A+C'*

*KA = 0*




| A | B | C | $A^+$ | $B^+$ | $C^+$ | JA | KA | TB | DC |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | X | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | X | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | X | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | X | 0 | 0 | 1 |
| 1 | 1 | 0 | X(1) | X(1) | X(1) | X(0) | X(0) | X(0) | X(1) |
| 1 | 1 | 1 | X(1) | X(1) | X(1) | X(1) | X(0) | X(0) | X(1) |



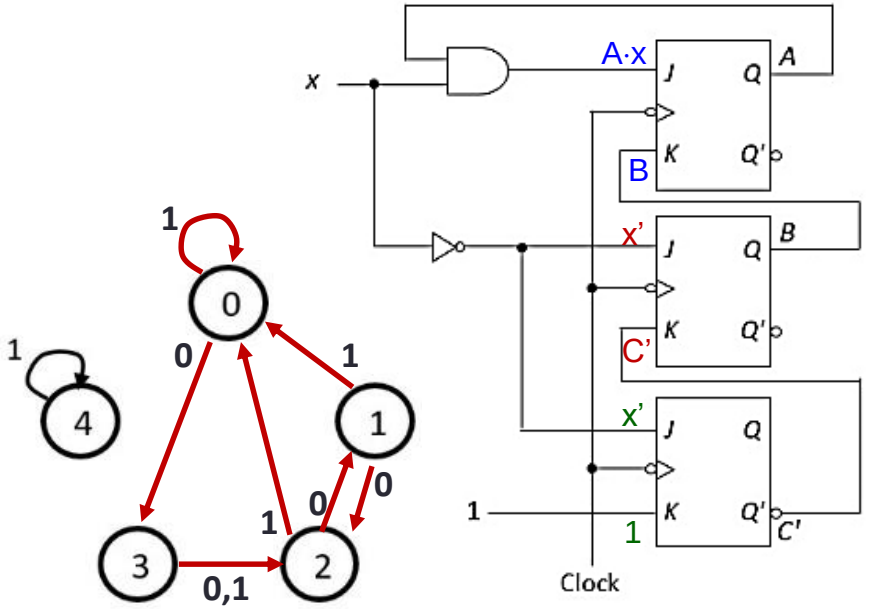

d. Is the circuit self-correcting? Explain your answer. (No mark will be awarded if there is no explanation or the explanation is wrong.) [1 mark]

No

# AY2020/21 Sem2 Q12

**Q12. Sequential circuits [12 marks]**

Study the following sequential circuit with three JK flip-flops and an external input *x*. The states are represented by *ABC*. This circuit implements a machine with 5 valid states *ABC* = 000 to 100 (or 0 to 4 in decimal). The other 3 states (*ABC* = 101 to 111, or 5 to 7 in decimal) are unused.




JA = A·x
KA = B

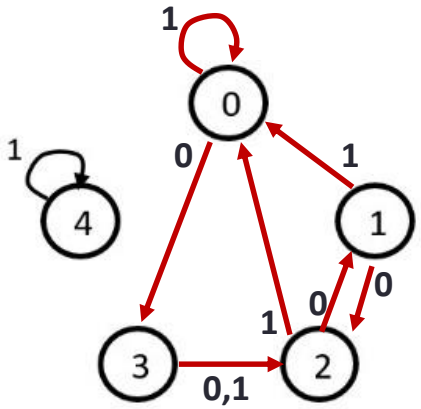
JB = x'
KB = C'

JC = x'
KC = 1

(a) Complete the state diagram below by drawing the arrows and labelling each arrow with the value of *x*. States are shown in decimal values. You need only fill in the transitions to the used states 0, 1, 2, 3 and 4. You do not need to include the unused states 5, 6 and 7 in your diagram. The transition from state 4 has been drawn for you. [8]

| A | B | C | x | JA | KA | JB | KB | JC | KC | A+ | B+ | C+ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

# AY2020/21 Sem2 Q12




JA = A·x
KA = B

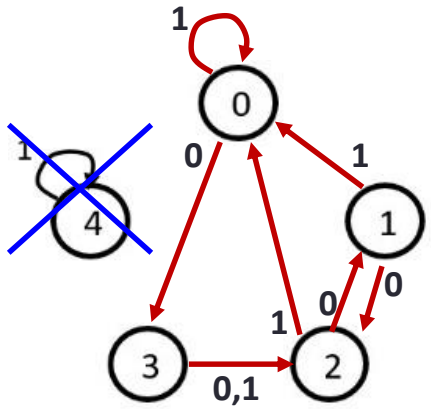
JB = x'
KB = C'

JC = x'
KC = 1

| A | B | C | x | JA | KA | JB | KB | JC | KC | A+ | B+ | C+ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

# AY2020/21 Sem2 Q12

(b) If you did your state table correctly, you would see that state 4 is unreachable from any of the other used states 0, 1, 2 and 3. Redesign the above sequential circuit by removing state 4, using two D flip-flops to replace the 3 JK flip-flops. Call you flip flops *F* and *G*. Your circuit now has 4 used states *FG* = 00 to 11 (or 0 to 3 in decimal). What are your flip-flop inputs ***DF*** and ***DG***? [4]




| F | G | x | $F^+$ | $G^+$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Note that $Q^+ = D$ for D flip-flops.

So $F^+ = DF$ and $G^+ = DG$.

***DF***

| F \ Gx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |

$DF = F'\cdot x' + F\cdot G$

***DG***

| F \ Gx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |

$DG = G'\cdot x'$

End of File