

Lecture 1: Introduction to EEE/CSE 120

Bahman Moraffah

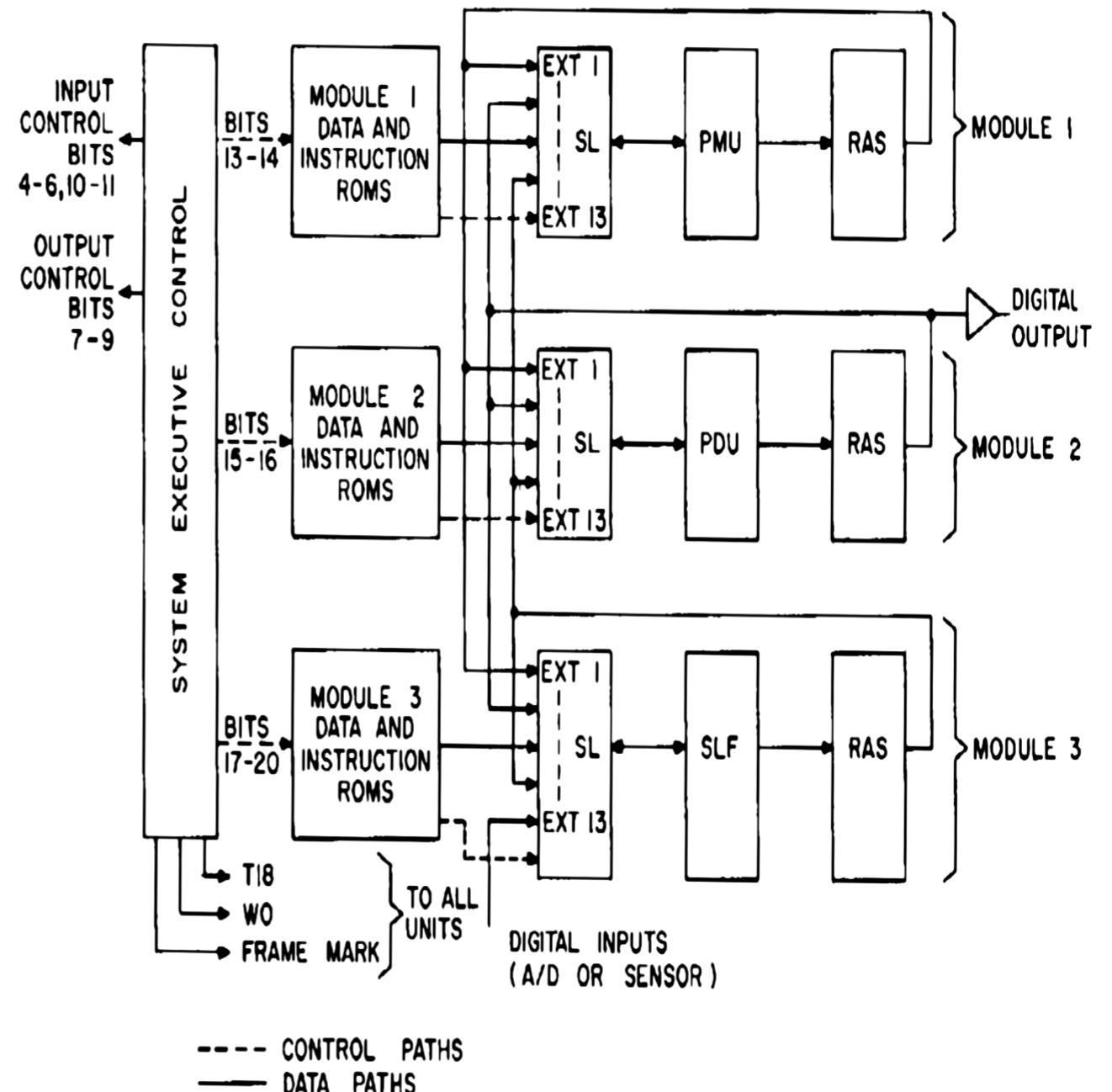
Electrical, Computer, and Energy Engineering
Arizona State University

Why do we learn digital circuit analysis

- We cannot store information in an analog way.
- Computers often chain logic gates together, by taking the output from one gate and using it as the input to another gate.
- Circuits enables computers to do more complex operations than they could accomplish with just a single gate.
- Logic gates are used to define the state of a system that has many inputs and outputs so that more complex units are created such as arithmetic units, shift registers, memory elements etc.
- Programs are written which manipulate these complex units giving what you see on the screen of your computer etc.

Cool Example: CPU

- 8 bit CPU
- Princeton Architecture
(von Neumann model)
- Few 8 bit registers
- Microcode sequence



70's CPU

How this class will work

- Instructor: Dr. Bahman Moraffah
- Grader: William Shih
- UGTA: Alden Davison
- TAs: TBD
- Class will be in-person or via Zoom
- Office Hours are virtual, and you should join using its own zoom meeting (It is different from Zoom link for the lectures)
- All of us should keep checking both Canvas and Course website for updates!
- Class is cumulative, so keep up with the materials and assignments!
- 5-6 assignments + Lab reports. They all must eb submitted on Canvas. No email or in-person submission will be accepted.

Logistics

- **Class meet:** Tuesday and Thursday
In-Person (Alphabetically) or through zoom

Find the zoom link on Canvas!

- **Office Hours:** T TH 9:30-10:15 am (Virtually through zoom).

The link for office hours is
different from the lectures!

- **Course**
Website: https://bmoraffa.github.io/EEE_CSE120_Fall2020.html
- **Canvas:** <https://canvas.asu.edu>
- **My Email:** Bahman.Moraffah@asu.edu

Exams and Quizzes

Exams/Quizzes	Date
Quiz 1	September 1
Quiz 2	October 1
Midterm	October 20
Quiz 3	November 3
Quiz 4	November 1
Final Exam	As scheduled by ASU

*subject to change
+ mini quizzes*

Exams will be through Lockdown Browser. Try to familiarize yourself with it. This method records video and sound as you take quizzes and exams. You will be required to use this method to take quizzes and exams whether you are in person or participating remotely.

All quizzes and exams are closed book and notes.
Exam and quiz dates are subject to change.

Course Grading

	Distributions
Lab Report	25%
Assignment	10%
Quizzes and Attendance	15%
Capstone Project	10%
Midterm	20%
Final Exam	20%

- For the letter grade check [syllabus](#).
- I will not curve, but I will provide a lot of opportunities to earn extra credit.
- Extra Credit:** I need volunteers to take notes each class, type it up and send it to me so it can be uploaded for the entire class. Each student can scribe at most 2 lectures.
- Incorrect Work & Correct Answer = NO CREDIT.
- No Work & Correct Answer = NO CREDIT.

ASU Sync

This course uses Sync. ASU Sync is a technology-enhanced approach designed to meet the dynamic needs of the class. During Sync classes, students learn remotely through live class lectures, discussions, study groups and/or tutoring.

- To access live sessions of this class go to the Canvas shell for this course on the side bar click on zoom and choose the lecture, you should attend.
- Classroom attendance is limited to 50% of the capacity of the room. Therefore, it may be required that you attend one day per week in person and one day per week online. For more information check the syllabus.
- If you cannot physically be on campus due to travel restrictions or personal health concerns, you will be able to attend your classes via ASU Sync during the fall semester. If you will not be on-campus for the fall semester, you are expected to contact your professors to make accommodations.
- You must attend lectures either in-person or through zoom (your choice!). Note that attendance is mandatory.

Labs

- 5 Labs + Capstone Project
- Lab instructions are posted on Canvas.
- Capstone will be posted and discussed thoroughly in class
- **Lab Reports:**
 - Lab results (schematic diagrams, timing diagrams) will be filled into a lab template. Lab templates will be posted on Canvas.
 - Lab templates have to be completed and submitted individually. No group submissions will be accepted.
 - Copying full reports or sections of other students, except for data generated as a group effort, is considered an academic integrity violation and will be reported.
 - Students must indicate their lecture session (instructor and meeting time) as well as the names of their lab partners on the lab submission.
 - Submissions must be in electronic format (doc or pdf, no individual jpegs) and must be submitted via the submission link on Canvas. No paper or email submissions of lab reports will be accepted.
 - Late lab submissions will be penalized at a rate of 10% per day late, up to a maximum penalty of 50%. No lab reports will be accepted after 5 working days, unless there is a valid excuse.
- **Capstone Project:**
 - This lab has to be performed individually, not as a group.
 - Students have to pick a one-hour time slot within their session to demonstrate a working finite state machine design, implemented in programmable logic, to the TA, and explain the operation to the TA to be graded and approved for completion.

Course in one glance

Thu, Aug 20	Syllabus, Introduction to EEE 120 & Electrical Fundamentals
Tue, Aug 25	Logical and Binary Systems, AND-OR, NAND-NOR Logic, Truth Tables, Realizations
Thu, Aug 27	Number Systems, Addition
Tue, Sep 1	Half Adder, Full Adder, Multi-bit Adder
Thu, Sep 3	2's Complement Representation, 2's Complement Arithmetic
Tue, Sep 8	Boolean Algebra I
Thu, Sep 10	Boolean Algebra II, SOP & POS Forms

Thu, Sep 17	The Uniting Theorem, Karnaugh Maps
Tue, Sep 22	Karnaugh Maps, Min SOP & Min POS, Don't Cares
Thu, Sep 24	MUX's, Decoders
Tue, Sep 29	MUX and DEC as Function Generators, PROMs
Tue, Oct 6	Tri State & Open Collector Buffers
Thu, Oct 8	Sequential Logic, Latches
Tue, Oct 13	Flip Flops

Thu, Oct 22	Synchronization and Registers
Tue, Oct 27	Synchronous Counters
Thu, Oct 29	Synchronous Machine Design, Moore Machine
Thu, Nov 5	Mealy Machines
Tue, Nov 10	Design Project
Tue, Nov 24	Brainless Microprocessor
Thu, Nov 26	No Class: Thanksgiving Recess
Tue, Dec 1	CPU Architecture and Microprocessor Systems

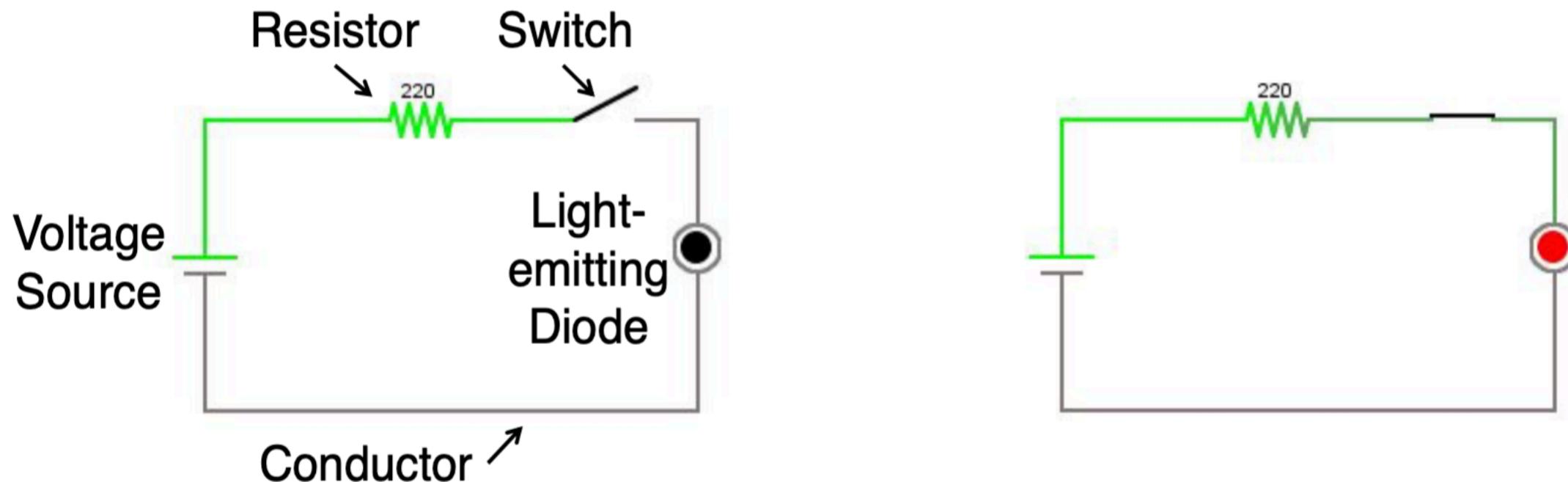
Electrical Fundamentals

- “**Charge**” carriers are the reason why we can observe “electric” phenomena
- **Electrons** are charge carriers in metals
- **Conductors**: materials having lots of freely movable charge carriers
- **Insulators**: materials that have no freely movable charge carriers
- **Energy source**: Charge carriers with a potential energy
- **Voltage**: Electrical potential – energy needed to move a unit charge from point ‘a’ to point ‘b’
- **Current**: Amount of charge that flows per second in a conductor
- Number of charges must be **constant** at all time – closed circuit necessary (Law of Conservation of Energy)

Why digital design

- In 17th century electricity was discovered.
- Built a switch to turn things on and off
 - They were huge switches
- Switches:
 - Insulators: Prevent the flow of electricity (Rubber)
 - Conductors: Allow the electricity to flow
 - Vacuum Tube: Amplify current/voltage and act like a switch
 - Very expensive and not durable
 - Semi-conductors
 - Make Transistors (Bell labs)
 - Better than Vacuum tubes most of the times
 - Disadvantage: Electromagnetic pulses don't work on vacuum tubes but works on transistors
- Transistors:
 - Analog → Continuous time
 - Digital → Discrete time (We only deal with {0,1})
- Electric circuits: Circuit diagrams are a schematic representation of the physical circuit elements and wires.

Convention



Switch off = circuit open
LED off

Switch on = circuit closed
LED on

Circuit Open (FALSE) (0)	Switch off
Circuit Closed (TRUE) (1)	Switch on

ATTN:

Active high → True = 1

Active low → True = 0

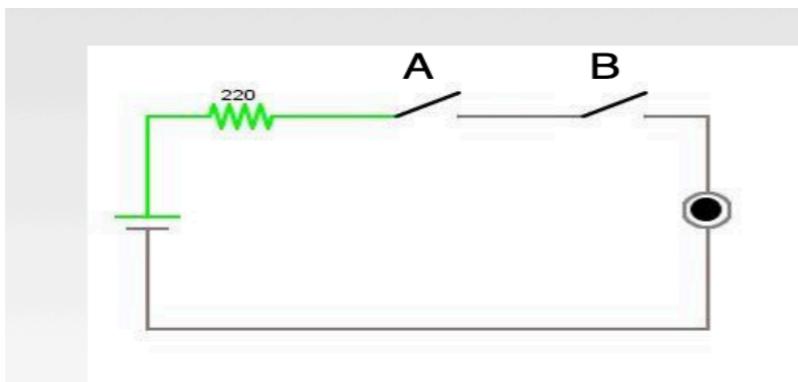
Switches: Connection

□ Series Connection

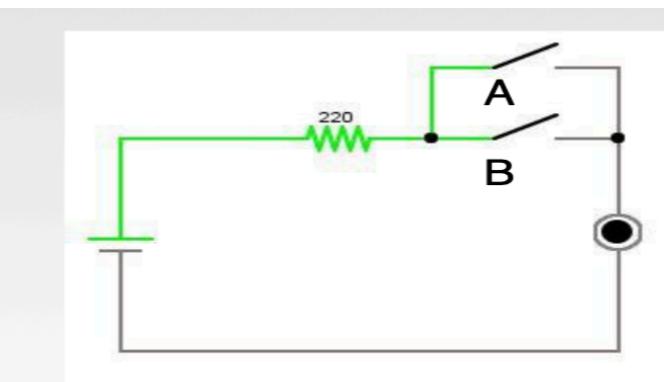
- If switches are back to back
- If two switches are in series, then circuit is closed only if switch A and switch B are both closed

□ Parallel Connection

- If switches share the same head and tail
- If two switches are in parallel, then circuit is closed if switch A or switch B is closed



Switch A	Switch B	LED
Off	Off	Off
Off	On	Off
On	Off	Off
On	On	On



Switch A	Switch B	LED
Off	Off	Off
Off	On	On
On	Off	On
On	On	On

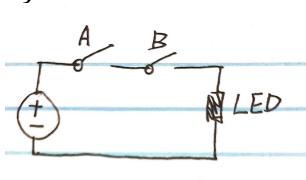
Truth Table

Next Lecture

- Truth Tables
- And/OR/NAND/ NOR/XOR/NXOR Gates
- Truth table for each of the gates
- Analysis of the gates

How to connect switches?

1) . series: back to back



Switch off \leftrightarrow False \leftrightarrow 0

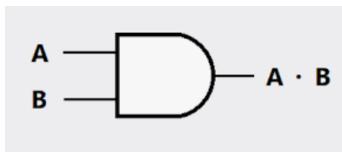
Switch on \leftrightarrow True \leftrightarrow 1

For the above series, only A open or only B open, the LED will not on. LED is on if both A and B are closed.

4 possibilities:

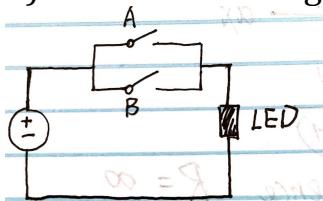
A	B	Y=Output
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table shows all possibilities of outputs.



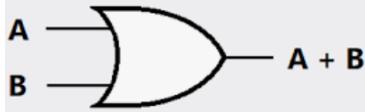
"AND" gate only true when both A and B are true.

2) . Parallel: sharing the same head and tail.



Two switches are in parallel, so output is on if the switches are both .

A	B	Y=Output
0	0	0
0	1	1
1	0	1
1	1	1



“OR” gate will be true if A is true or B is true.

Question: How do we design a truth table that allows the LED to turn off/on with each switch being independent of the other?

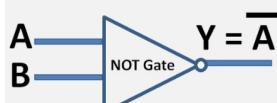
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



“XOR” gate(not series, not parallel) gives true output when the number of true inputs is odd.

Explanation:

- LED is on when 1). Switch A is on **and** Switch B is off(not on) **OR**
- 2). Switch A is off(not on) **and** Switch B is on



A	Y
0	1
1	0

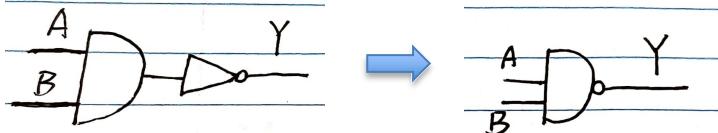
“NOT” gate is one the outputs the opposite state as what is input.



“Buffer” gate is the one that outputs equal to its inputs.

A(Input)	Y(Output)
0	0
1	1

“AND” gate combines with “NOT” gate: **“NAND” gate** (means not “AND”)



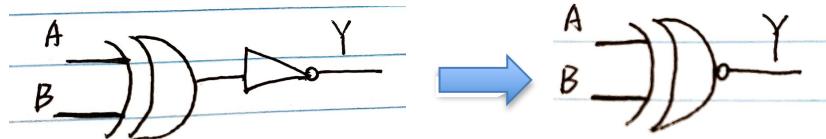
A	B	Y(final output)	C(before "not")
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

“OR” gate combines with “NOT” gate: **“NOR” gate** (means not “or”)



A	B	Y(final output)	C(before “not”)
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

"XOR" gate combines with "NOT" gate: "XNOR" gate



A	B	Y(final output)	C(before "not")
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0

The number of possibilities for outputs depends on how many input(n):

The number of output: 2^n

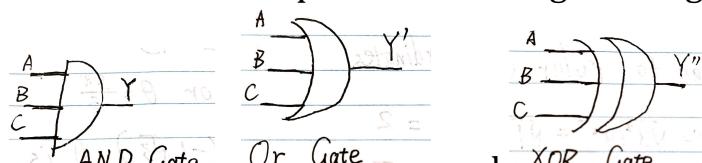
Eg. 2 inputs --- 4 outputs

3 inputs --- 8 outputs

4 inputs --- 16 outputs

Example:

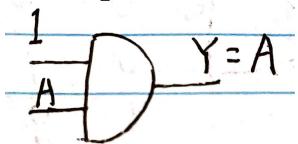
There are three inputs A, B, and C go through the following three gates



AND Gate, Or Gate and XOR Gate, list the outputs for truth table for Y, Y', and Y''.

A	B	C	Y	Y'	Y''
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	1

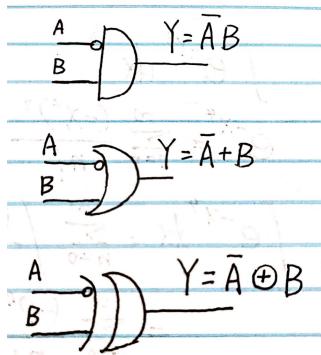
Example 1.



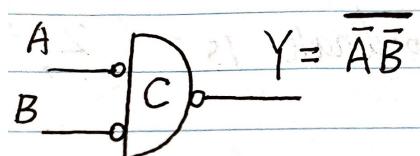
The output Y depends on A: A
so the output Y=A



Example 2.



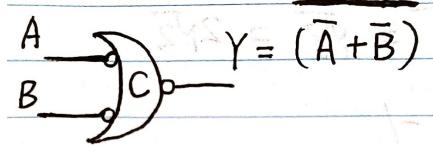
Example 3.



A	B	Y	C
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

(the final outputs are the same as "OR" gate)

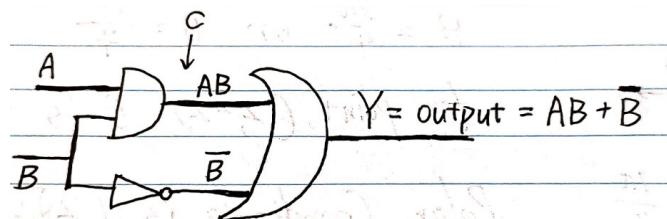
Example 4.



A	B	Y	C
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

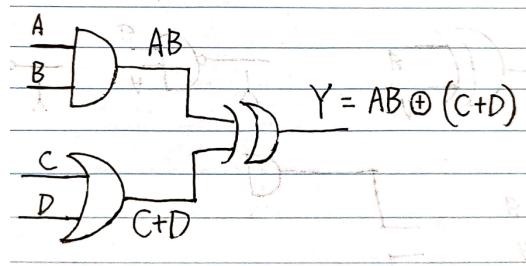
(the final outputs are the same as "AND" gate)

Example 5.



A	B	Y	C=AB
0	0	1	0
0	1	0	0
1	0	1	0
1	1	1	1

Example 6.



A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	1	0	0	0
1	0	0	0	0
1	1	0	0	1
1	0	1	0	1
1	0	0	1	1
0	1	1	0	1
0	1	0	1	1
0	0	1	1	1
1	1	1	0	0
1	1	0	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	1	0

Explanation:

- 1) . For "And" gate, its output is AB, which means only when A and B are 1, the output is 1;
- 2) . For "OR" gate, its output is C+D, which means when A is 1 and B is 0, B is 1 and A is 0, or both A and B are 1, the output is 1;
- 3) . For the final output Y, it is "XOR" gate, which means when the values of inputs are different(A is 1 and B is 0, or A is 0 and B is 1), the output Y is 1

Lecture 3

EEE/CSE 120 : Number System.

$$(654)_{10} = 4 \times 10^0 + 5 \times 10^1 + 6 \times 10^2 = \sum_{i=0}^n c_i 10^i$$

coeff. base exponent

base = 10 \leftrightarrow Decimal $\leftrightarrow c_i < 10$

$$c_i \in \{0, 1, 2, \dots, 9\}$$

base = 2 \leftrightarrow Binary $\leftrightarrow c_i < 2 \Rightarrow c_i \in \{0, 1\}$

$$(\dots)_2$$

$$\sum_{i=0}^n c_i 2^i$$

Q: 654 in terms of powers of Two?
 (Binary rep. of 654)

<u>Binary</u>	<u>Decimal</u>	<u>Binary</u>	<u>Decimal</u>
2^0	1	2^8	256
2^1	2	2^9	512
2^2	4	2^{10}	1024
2^3	8	2^{11}	2048
2^4	16	.	.
2^5	32	.	.
2^6	64		
2^7	128		

$$654 = 1 \times 2^9 + 1 \times 2^7 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

↓
 0×2^8

$$(1010001110)_2$$

$\begin{matrix} \uparrow & \uparrow \\ 2^1 & 2^0 \end{matrix}$

base $=2^3=8 \rightarrow$ octal (oct) $\leftrightarrow c_i \in \{0, 1, 2, \dots, 7\} (c_i < 8)$

base $=2^4=16 \leftrightarrow$ HEXADECIMAL (HEX) $\leftrightarrow c_i < 16$

$$\begin{matrix} 0-9 \\ 10 \leftrightarrow A \\ 11 \leftrightarrow B \end{matrix}$$

$$\begin{matrix} 12 \leftrightarrow C \\ 13 \leftrightarrow D \\ 14 \leftrightarrow E \end{matrix}$$

$$15 \leftrightarrow F$$

$$(1 \mid 0 \ 1 \ 1 \mid 0 \ 1 \ 1)_2$$

$\downarrow \uparrow \uparrow$
 $2^2 \ 2^1 \ 2^0$

Example :

$$654 = (0 \mid 1 \ 0 \ 1 \ 0 \mid 0 \ 0 \ 1 \ 1 \ 1 \ 0)_2$$

Q: What is the oct?

$$\begin{array}{r} 0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 \\ \hline 2 \end{array}$$

$$\begin{array}{r} 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 \\ \hline 6 \end{array}$$

$$(1 \ 2 \ 1 \ 6)_8$$

EX:

$$654 = (1\ 011\ 000\ 1110)_2$$

$1 \times 2^1 = 2$

$2^3 = 8$

$0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 = 14$

HEX ?

E

$$(28E)_{16}$$

Example : $(B36E)_{16}$ binary?

$$E = 14 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \leftrightarrow (1110)$$

$$6 = 1 \times 2^2 + 1 \times 2^1 \leftrightarrow (0110)$$

$$3 = 1 \times 2^1 + 1 \times 2^0 \leftrightarrow (0011)$$

$$B = 11 \longleftrightarrow 2^3 + 1 \times 2^1 + 1 \times 2^0 \longleftrightarrow (1011)$$
$$(1011 \text{ } 00 \text{ } 11 \text{ } 0 \text{ } 11 \text{ } 0 \text{ } 1110)_2$$

$$654 \begin{array}{r} | \\ \hline Q: 327 \end{array}$$

$$\begin{array}{r} | \\ R = 0 \end{array}$$

least significant
digit

$$654 = 2 \times 327 + 0$$

$$327 = 2 \times 163 + 1$$

$$(1010001110)_2$$

$$\begin{array}{r} & 2 \\ R & | \\ & 654 \end{array}$$

most significant

$$\text{Ex: } \begin{array}{r} +1 \\ 7 \\ \hline 12 \end{array}$$

binary: $\begin{array}{r} 1 \\ + 0 \\ \hline \end{array}$ $11 \rightarrow 1 \times 2^0 + 1 \times 2^1 = 11$

$+ 0 0 + 0 \rightarrow 2 +$

$\begin{array}{r} 1 \\ + 1 \\ \hline 101 \end{array}$

13

$1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 = 13$

$+ 1 0 1 1$

$+ 0 0 1 1$

$\hline 1 1 1 0$

How do we define negative numbers?

① Signed bit -

negative → 1 (0 0 1) → -1

positive → 0 0 0 1 → +1

"0" → 1 0 0 0 } two rep.
0 0 0 0

-1 → 1 0 0 1

1 → 0 0 0 1

$$\begin{array}{r} 1 0 0 1 \\ 0 0 0 1 \\ \hline 1 0 1 0 \end{array} \rightarrow 2^3 + 2^1 = 10 \quad \times$$

② offsetting

$$-8, 7 \xrightarrow{+8} 0, 15$$

We still have $a + (-a) \neq 0$

$$\rightarrow \boxed{0} \quad 1 \quad 1 \quad 1$$

③ 2's Complement.

$$\begin{array}{r} 1 \quad 0 \quad 0 \quad 0 \\ - \quad 0 \quad 0 \quad 1 \\ \hline \boxed{1} \quad 0 \quad 0 \quad 1 \end{array}$$

Rule of thumb: To get a negative number from a positive number :

① Find a number which if added to the original number $\rightarrow 111$

② Add 1 to the number $\rightarrow 001$

Lecture 4:

EEE/SE 120 : 2's complement and its Arithmetic

- HW 1 is due Sep 3
- Lab 0 is due Sep 14
- Quiz 1 is on Sep 8
- Office hours T-Th 9:30^{AM}
- Join the Lab zoom meeting if there is an issue w/ Labs.
- When you scribe you need to send the notes to me within 2 days

Defining negative numbers:

1 Sign bit

Neg \rightarrow $\boxed{1} \ 0 \ 0 \ 1 \rightarrow -1$

pos \rightarrow $\boxed{0} \ 0 \ 0 \ 1 \rightarrow +1$

problems: 1) "0" has two rep.
2) $1 + (-1) \neq 0$

2 Offsetting

problem: $1 + (-1) \neq 0$

3 2's Complement

1. zero has one rep.

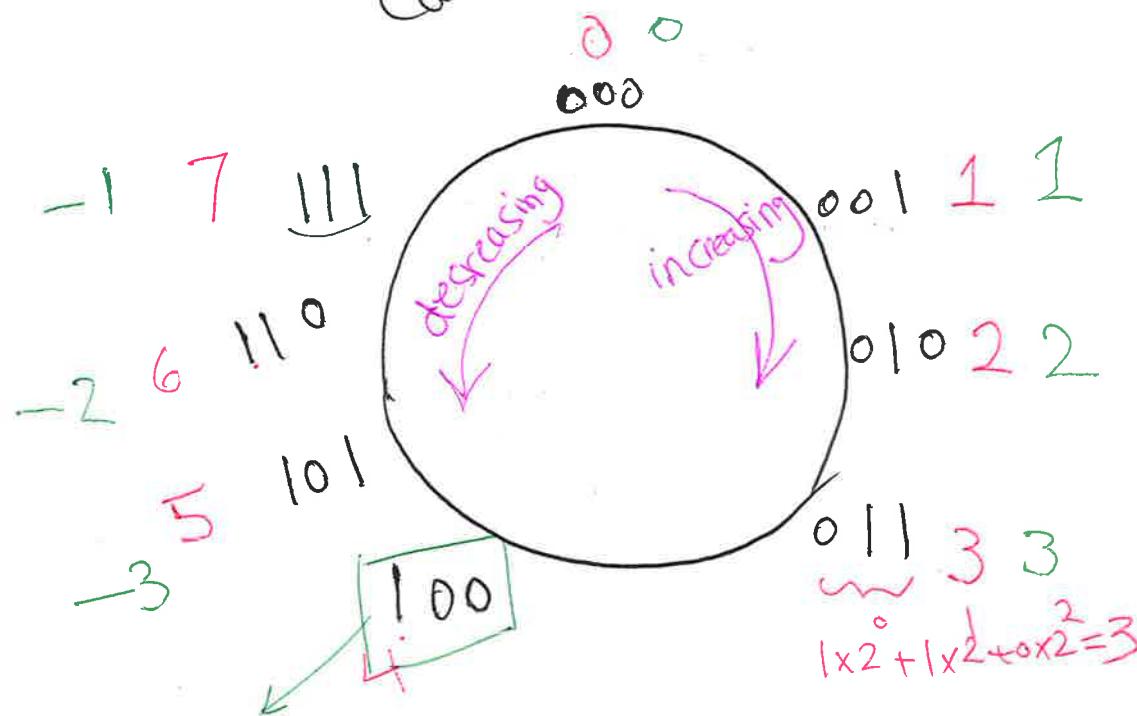
2. $1 + (-1) = 0$

3. Signed bit is part of the number

4. positive numbers stay the same.

$$\begin{array}{r}
 (+3)_{10} \\
 + (-3)_{10} \\
 \hline
 (0)_{10}
 \end{array}
 \quad \longleftrightarrow \quad
 \begin{array}{c}
 (1\ 0\ 1\ 1)_2 \\
 \boxed{1} \ \boxed{0} \ \boxed{1} \\
 \hline
 \cancel{\boxed{1}} \quad \overset{\cancel{1}}{0} \ 0 \ 0
 \end{array}
 \quad \rightarrow (-3)_{10}$$

ignore.
Carry out



-4
define this
to be -4

unsigned numbers
Signed number

How to get a negative number from a positive number?

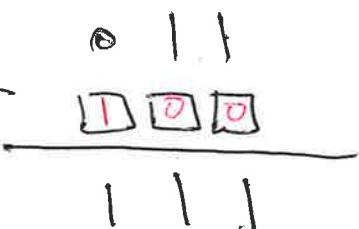
Rule: ① Find the number which added to the original number that results in 111

② add 1 to that number

Example:

$$\begin{array}{r} & \boxed{1} & \boxed{1} \\ & + & \\ 0 & | & | \\ + & \boxed{1} & \boxed{0} & \boxed{1} \\ \hline \boxed{1} & 0 & 0 & 0 = & \boxed{111} + \boxed{001} \\ & \underbrace{\quad\quad\quad}_{\boxed{111}} & & & \\ & & \boxed{001} & & \\ \hline & & \boxed{110} & & \end{array}$$

Another way of writing the rule:

- [1] Flip all the bits of the original number + (one's complement)

$$\begin{array}{r} 0 \ 1 \ 1 \\ + 1 \ 0 \ 0 \\ \hline 1 \ 1 \ 1 \end{array}$$
- [2] Add 1 to it.

Example: $(0 \ 1 \ 1) = 3_{10}$

$$1 \times 2^0 + 1 \times 2^1 = 3$$

- ① Flipping the bits : 1 0 0
- ② Add 1 to it

$$\begin{array}{r} 1 \ 0 \ 0 \\ + 0 \ 0 1 \\ \hline 1 \ 0 \ 1 \rightarrow (-3)_{10} \end{array}$$

Example: 2's complement of a negative number!

$$\underline{(1 \ 0 \ 1)}_2 = (-3)_{10}$$

① flipping bits $\rightarrow (0 \ 1 \ 0)_2$

② Add 1 to it

$$\begin{array}{r} 0 & 1 & 0 \\ + & & \\ \hline 0 & 0 & 1 \end{array}$$

$$\underline{\quad \quad \quad}$$

$$1 \times 2^2 + 1 \times 2^0 = 3$$

Example: 2's complement of 000?

① flipping the bits $\rightarrow \underline{\quad \quad \quad}$

② To add 1 to it $\rightarrow \underline{\quad \quad \quad}$

Example:

$$\begin{array}{r} \boxed{1} \ 0 \ 1 \\ \xrightarrow{\quad} 1 \ 0 \ 1 \ 0 \\ \textcircled{2} \ 0 \ 1 \ 0 \\ \qquad\quad 0 \ 0 \ 1 \\ \hline 0 \ 1 \ 1 \end{array} \rightarrow +3$$

(-3)₁₀ is converted to binary. The result is 1010. A circled 2 is above the second column from the left. The result is 011, which is +3.

Example:

$$\begin{array}{r} 1 \ 0 \ 0 \ 0 \\ \underbrace{\quad\quad\quad}_{(-8)} \\ (-8)_{10} \end{array} \quad (4 \text{ bit})$$

2's Compl.

$$\begin{array}{r} \textcircled{1} \ 0 \ 1 \ 1 \ 1 \\ \textcircled{2} \ 1 \ 1 \ 1 \ 1 \\ \hline 0 \ 0 \ 0 \ 1 \\ \hline 1 \ 0 \ 0 \ 0 \end{array} \rightarrow +8$$

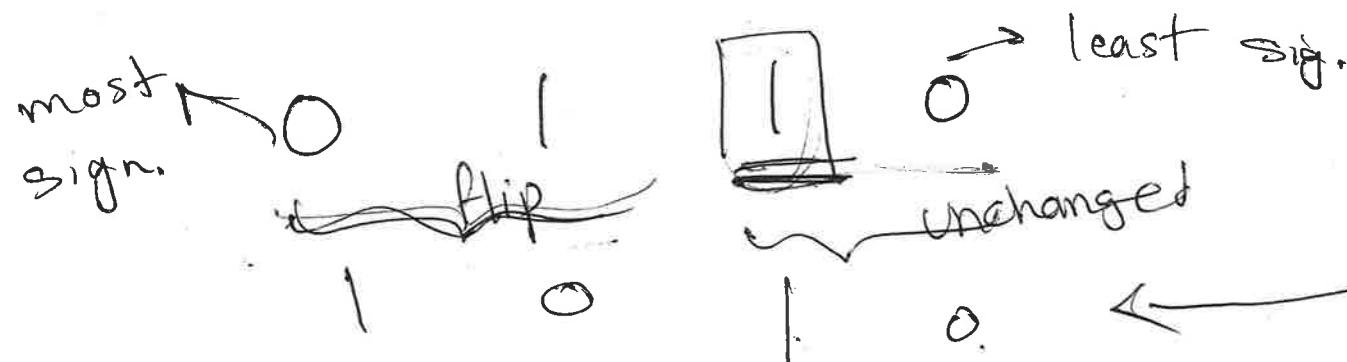
Example : 0 1 1 0

1 Flip all the bits : 1 0 0 1

2 To add 1 to it $\begin{array}{r} + \\ 0 0 0 1 \\ \hline 1 0 1 0 \end{array}$

Quick way of Computing 2's Complement :

Spot the first "1" and leave everything the same up to that "1" and then flip the rest of the bits to the left



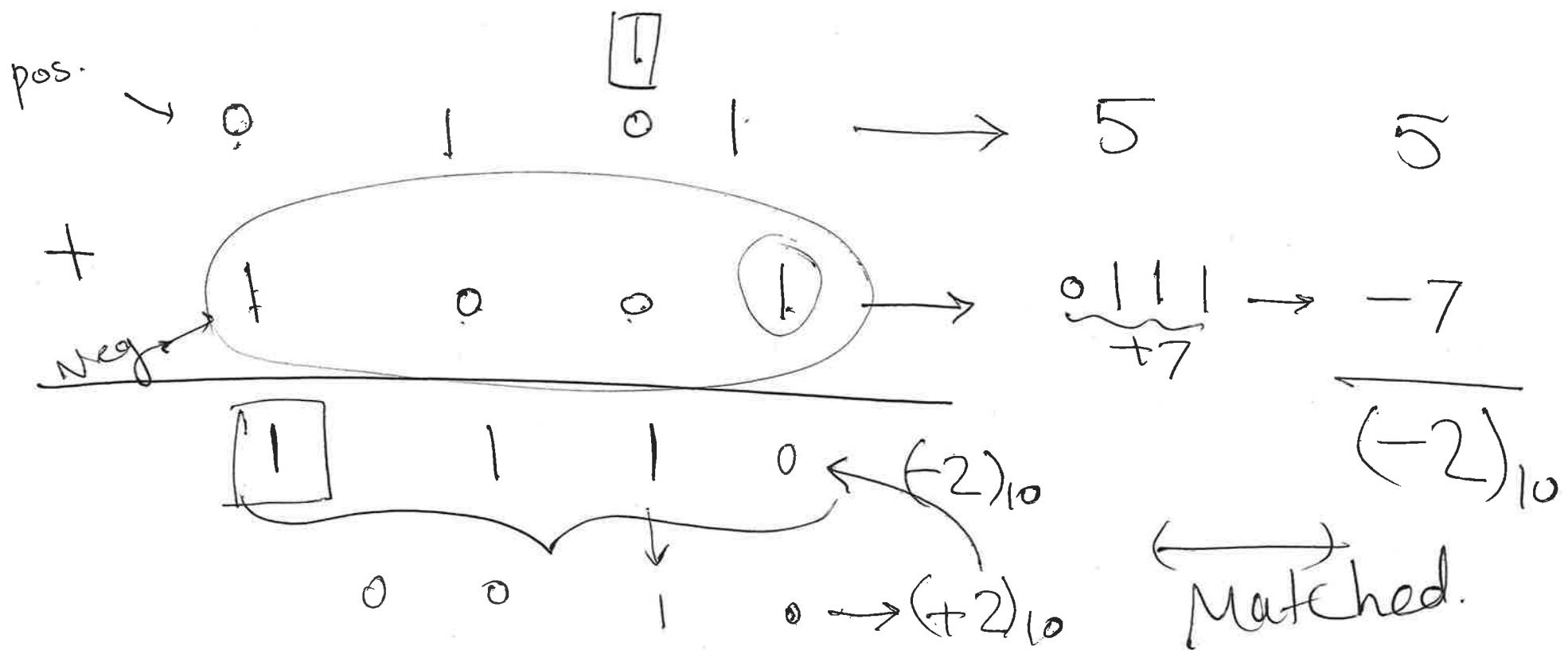
Example

Signed binary

↳ Addition :

① numbers can be pos.
or neg.

② Addition doesn't care
about sign.



Example: Neg carry in

$$\begin{array}{r} \text{Neg} \\ + \\ \text{1. } 1 \ 1 \ 0 \\ \hline \end{array} \rightarrow 0111 \rightarrow (-7)$$

 Neg

$$\begin{array}{r} \text{0. } 0 \ 1 \ 0 \\ \hline 2 \end{array} \rightarrow (-2)$$

 carry out ~~1~~
~~0. 1 1 1~~ ~~1~~ ~~-9~~
 Don't match.
 overflow.

pos	Neg	pos	Neg
pos	Neg	Neg	post

we might have
 OF.

when we look at the most sign. bit
 if carry in \neq carry out \Rightarrow OF.

Example : Add 2's complement of the following numbers and determine when we of

$$\begin{array}{r}
 \boxed{0} \\
 1 \ 0 \ 0 \ 0 \rightarrow -8 \\
 + 0 \ 1 \ 0 \ } \rightarrow 5 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 \boxed{1} \xrightarrow{\text{carryin}} \\
 0 \ 1 \ 0 \ 0 \rightarrow +4 \\
 + 0 \ 1 \ 1 \ 0 \rightarrow +6 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 1 \ 1 \ 1 \ 1 \\
 + 1 \ 1 \ 1 \ 0 \\
 \hline
 \end{array}$$

$$\begin{array}{r}
 \boxed{0} \quad 1 \ 0 \ 1 \ (\underline{-3})_{10} \\
 \text{Carry in} = \text{Carry out} \quad \boxed{0} \quad 1 \ 0 \ 1 \ 0 \ (\underline{10})_{10} \\
 \text{Carry out} \neq \text{Carry in} \\
 \downarrow \\
 0 \ 0 \ 1 \ 1 \rightarrow -3
 \end{array}$$

$$\begin{array}{r}
 \text{Carry out} \neq \text{Carry in} \\
 \downarrow \\
 \boxed{0} \ 1 \ 1 \ 0 \quad +6 \\
 \text{OF.}
 \end{array}$$

$$1010 = -6$$

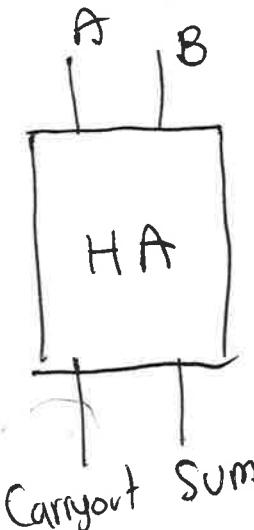
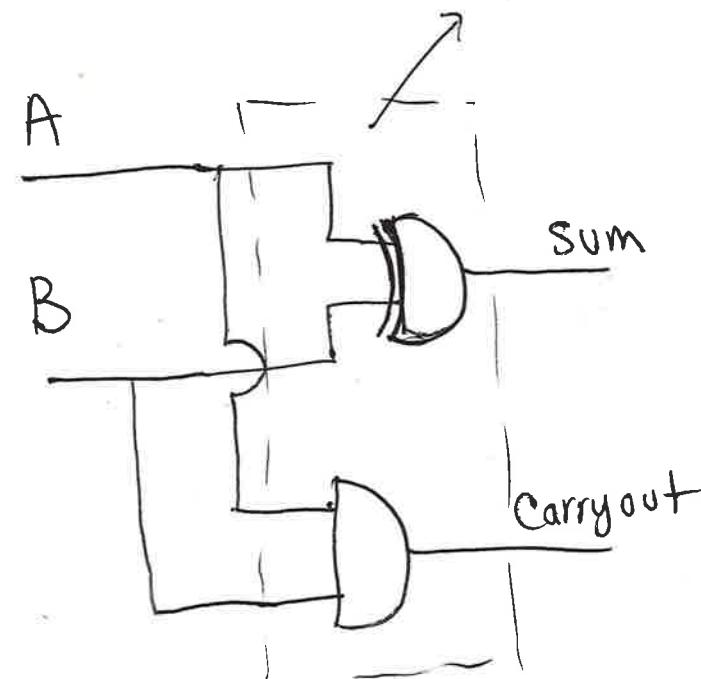
$$\begin{matrix} \text{?} \\ 2S \\ \text{comp} \end{matrix} \quad \left(\begin{matrix} 0 & 0 & 1 & 1 \end{matrix} \right) = 3$$

Lectures 5:
 EEE / KSE 120 : Adders :

A	B	Sum	Carryout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

input XOR AND

HALF ADDER



"Sum \Leftrightarrow XOR"

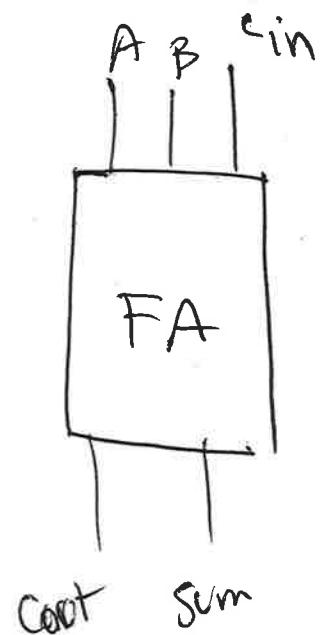
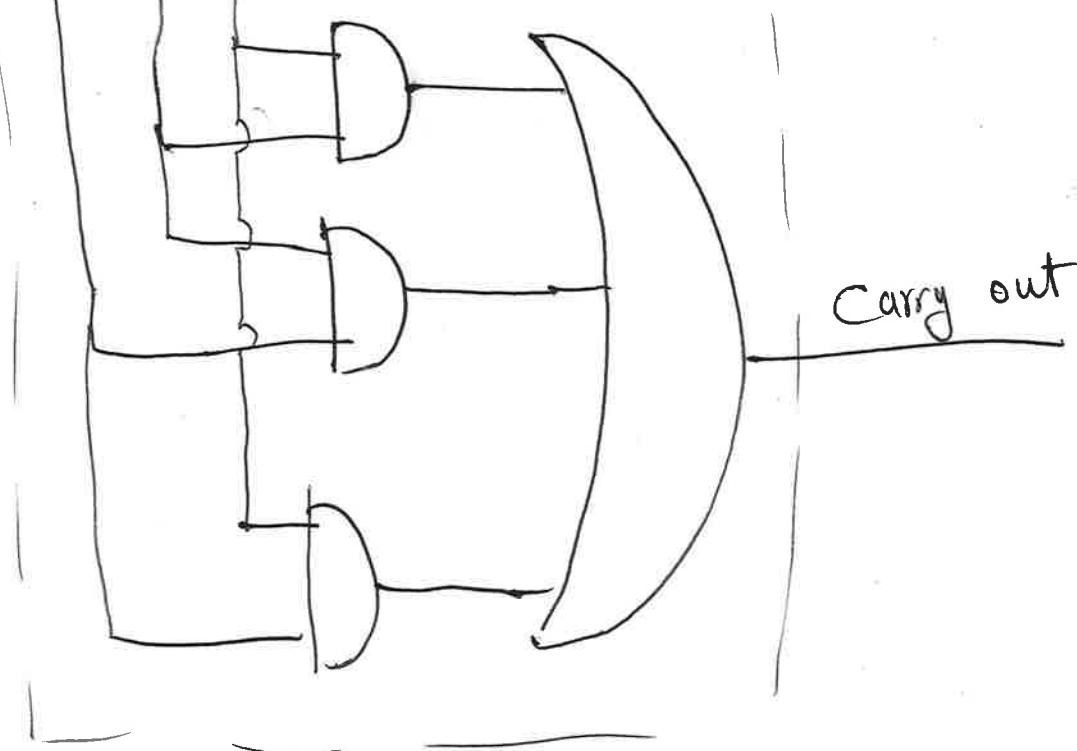
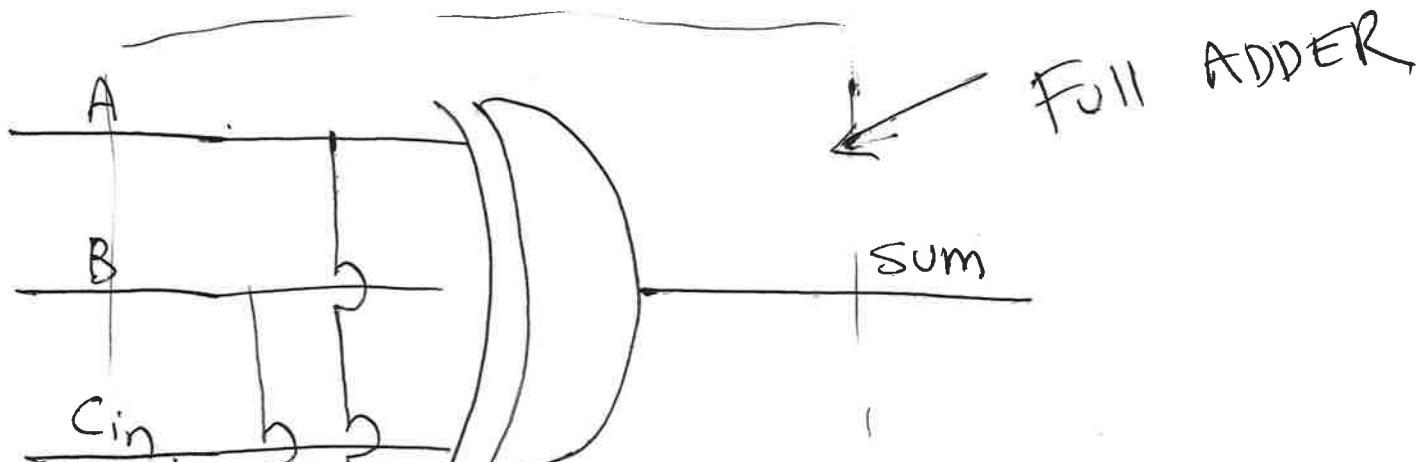
3 bit binary

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1 * $\bar{A} \bar{B} C_{in}$
1	0	0	1	0
1	0	1	0	1 * $A \bar{B} C_{in}$
1	1	0	0	1 * $A B \bar{C}_{in}$
1	1	1	1	1 * $A B C_{in}$

V.O.R

$$C_{out} = \underbrace{\bar{A} \bar{B} C_{in}} + \underbrace{A \bar{B} C_{in}} + \underbrace{A B \bar{C}_{in} + A B C_{in}} \\ (\bar{A} \bar{B} + A \bar{B}) C_{in} = A C_{in} + B C_{in}$$

C_{in}	\bar{C}_{in}	$C_{in} + \bar{C}_{in}$
0	1	1
1	0	1



$$(\bar{A}B + A\bar{B})C_{in} = AC_{in} + BC_{in}$$

$$(\bar{A}B + \underbrace{AB + A\bar{B} + A\bar{B}}_{=0})C_{in}$$

$$\cancel{(\bar{A} + A)B} + A(\cancel{B + \bar{B}})C_{in} = (A + B)C_{in} = AC_{in} + BC_{in}$$

$$C_{out} = AC_{in} + BC_{in} + AB$$

$$\begin{array}{r}
 \text{pos.} \quad 0 \quad 0 \\
 \text{Cin} \quad \downarrow \quad 0 \quad 0 \\
 \text{pos.} \quad 1 \quad 0 \\
 \hline
 \text{Sum} \quad 1 \quad 0
 \end{array}$$

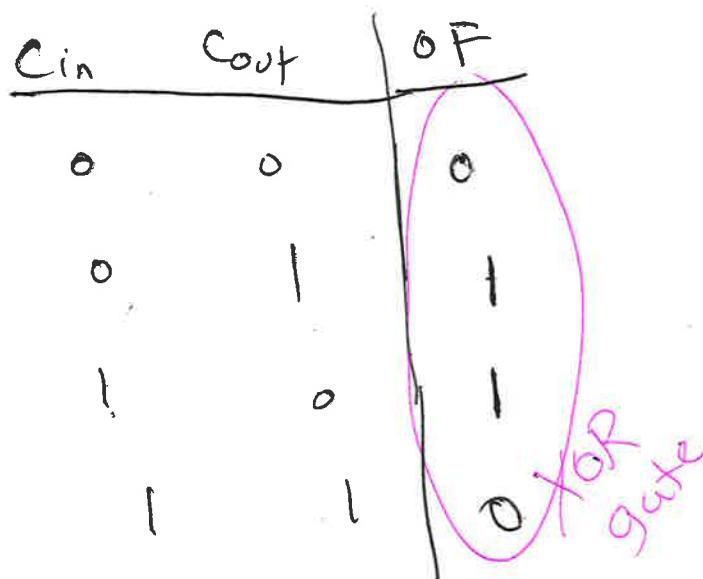
$$\begin{array}{r}
 \text{Carryout} \quad 0 \\
 \text{OF} \quad \checkmark \\
 \hline
 \end{array}$$

$$\begin{array}{r}
 \text{Neg} \quad 1 \quad 1 \\
 \text{Neg} \quad \downarrow \quad 1 \quad 1 \\
 \text{Neg} \quad 1 \quad 0 \\
 \hline
 \text{Sum} \quad 1 \quad 0
 \end{array}$$

Neg \rightarrow 1 0
1 1

$$\begin{array}{r}
 \text{Neg} \quad 1 \quad 1 \\
 \text{Neg} \quad \downarrow \quad 1 \quad 0 \\
 \text{Neg} \quad 0 \quad 0 \\
 \hline
 \text{Sum} \quad 0 \quad 1
 \end{array}$$

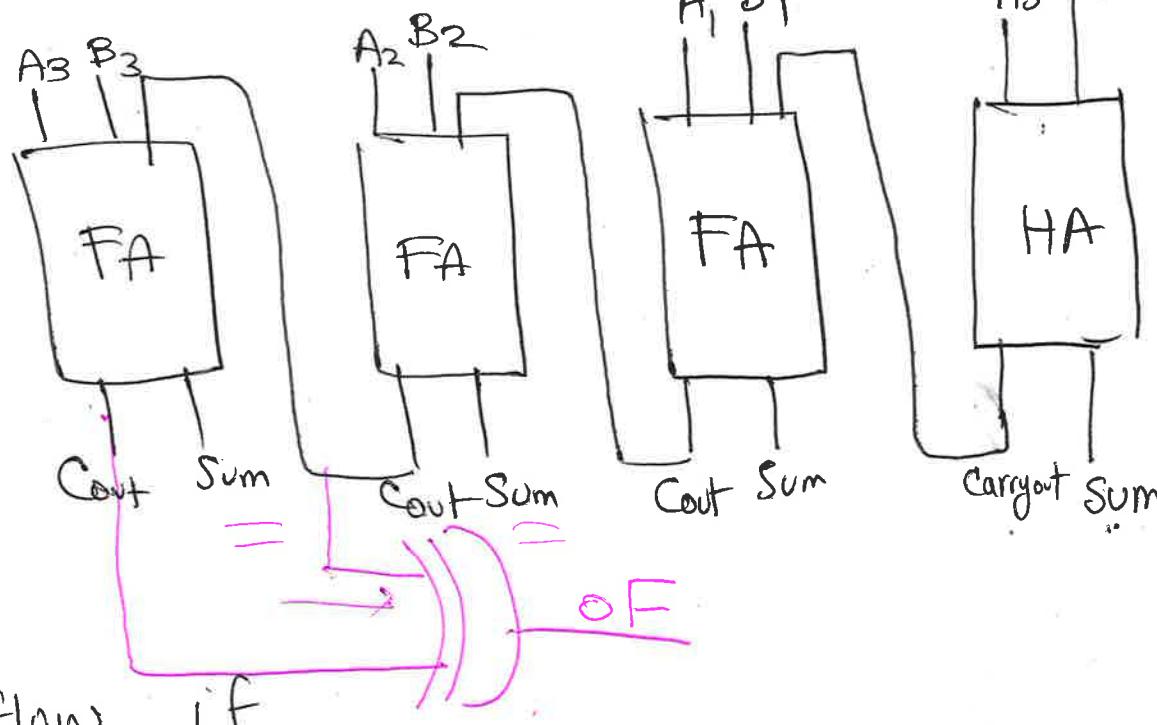
Neg \rightarrow 1 0
1 1



0 1 F 0

+ 1 0 1 1

0 1 Sum
1 0 carryout



we overflow if

Carry in \neq Carry out

$$+ \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix} = A = A_3 A_2 A_1 A_0$$

$$+ \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix} = B = B_3 B_2 B_1 B_0$$

$A + B$
 $A - B$

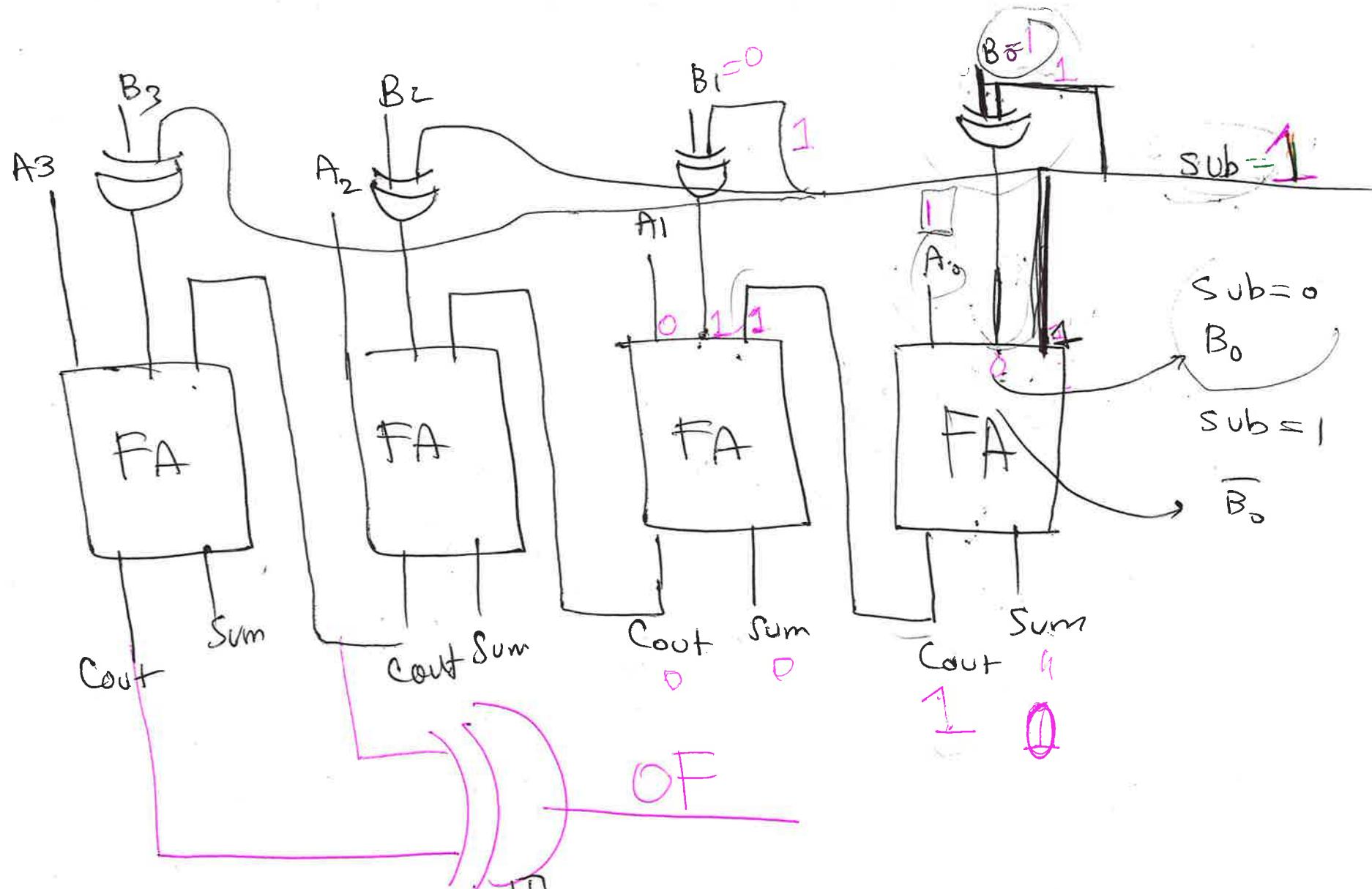
Reminder: 2's complement

① Flip all the bits

② to add 1 to it

$$\begin{array}{r}
 & 0 & 1 & 1 & 0 \\
 + & 0 & 0 & 0 & 1 \\
 \hline
 & 0 & 1 & 1 & 1
 \end{array}$$

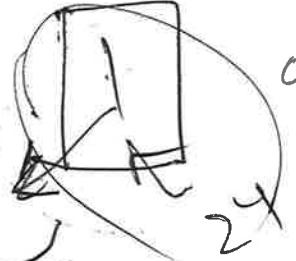
Subtractor = { Sub = 0 (\Leftrightarrow we're adding)
Sub = 1 (\Leftrightarrow subtracting)



$$\begin{array}{r}
 A = 0 \ 1 \ 0 \ 1 \\
 + B = 1 \ 0 \ 0 \ 1 \\
 \hline
 \text{Sum} = 0 \ 1 \ 1 \ 1
 \end{array}$$

Overflow (OF) is asserted (1) because the sum is 1110.

most
sig.



S_4

C_0

0. 1

least significant bit



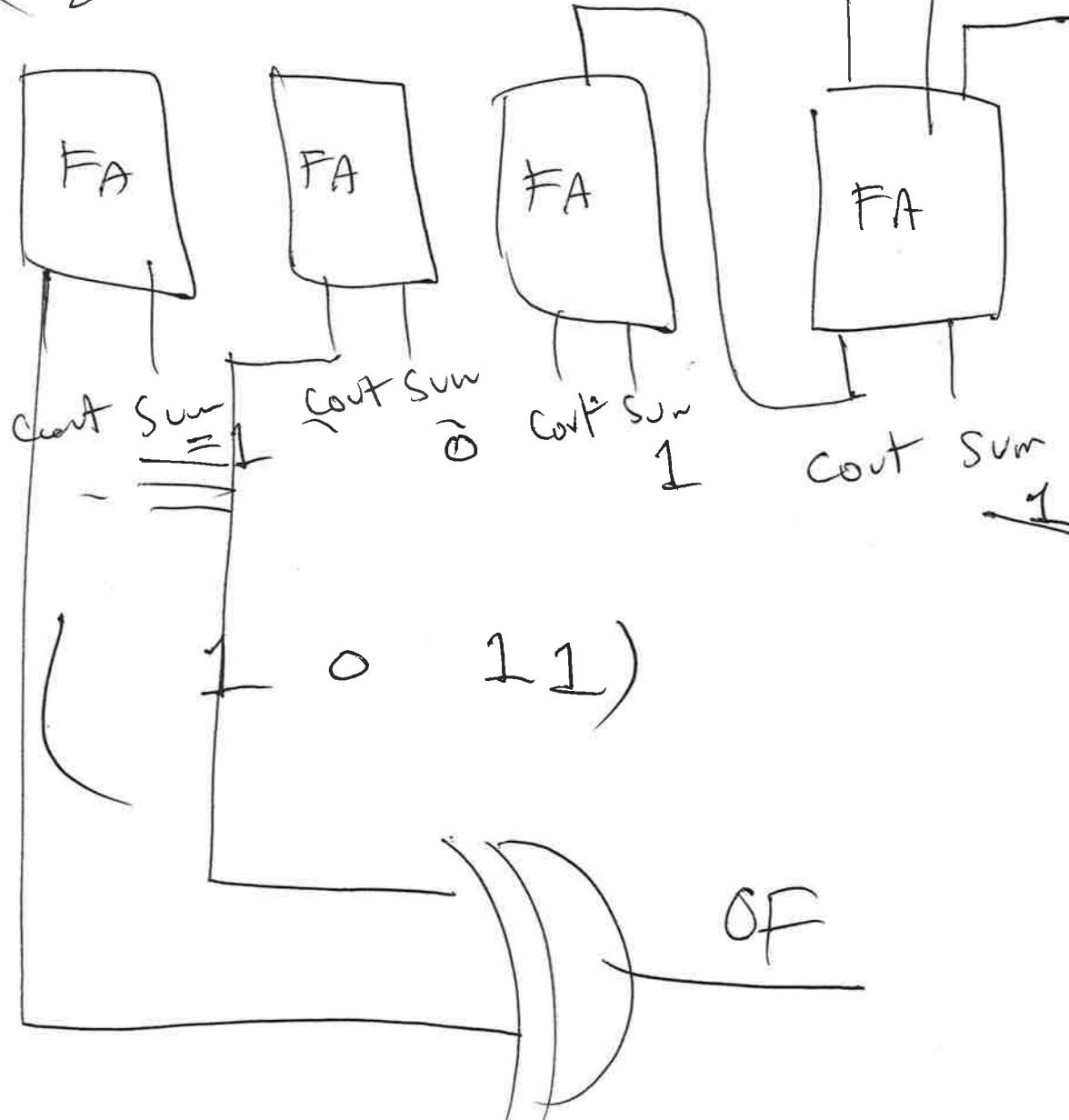
S_3

C_1

B_0

A_0

Sub

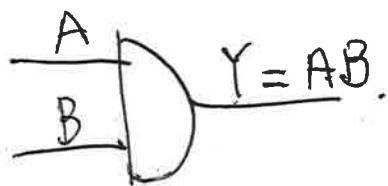


Lecture 6:

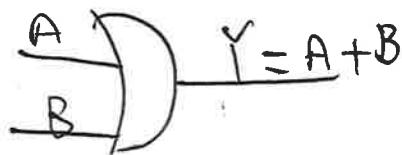
EEE/CSE 120 : Boolean Algebra

[EEE/CSE 120 - Quiz 1] ← title
← Email

bahman.moraffah @ asu . edu



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

$$1 \oplus 1 = 0$$

$$1 + 1 = 1$$

$$5 + 7 = 12 \text{ (closure)}$$

$$5 + 0 \leftarrow = 5$$

$$5 \cdot 1 = 5$$

$$\rightarrow 2 \cdot 3 \cdot 4 = 2 \cdot (3 \cdot 4) \neq (2 \cdot 3) \cdot 4$$

$$= (2 \cdot 4) \cdot 3$$

$$2 \cdot (3 + 4) = 2 \cdot 3 + 2 \cdot 4$$

"group"

① Closure :

$$\underbrace{1 + 1 = 1}_{\text{"OR" gate}}, \quad \underbrace{1 \oplus 1 = 0}_{\text{XOR gate}}$$

② Associativity

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C) = (C \cdot A) \cdot B$$

③ Neutral element

$$A \cdot B = A$$

?

A	B	$\#AB$
0	0	0
0	1	0
1	0	0
1	1	1

$B=1$

$$A + B = A$$

?

A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

$$A \oplus B = A$$

?

$$B = 0$$

EX: prove that $A \cdot (B+C) = A \cdot B + A \cdot C$

$$A \cdot (B+C) = (A+B) \cdot (A+C)$$

A	B	C	<u>$A \cdot (B+C)$</u>	$A \cdot B + A \cdot C$
0	0	0	0	0
1	0	0	1	0
1	1	0	1	1
1	0	1	1	1

Summary :

$$A \cdot 0 = 0$$

$$\underbrace{A \cdot 1 = A}_{\text{AND gate}}$$

$$A \cdot A = A$$

$$\underbrace{A \cdot 0 = 0}_{\text{AND Gate}}$$

$$A + 1 = 1$$

$$\underbrace{A + A = A}_{\text{"OR" gate}}$$

$$A + 0 = A$$

$$1 + 1 = 1$$

Example: "OR" gate

line	A	B	$A+B$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

$$Y = \text{output} = (\overline{A}\overline{B}) + (\overline{A}\overline{B}) + (AB) = A+B$$

Sum of products

$$= \sum m(1, 2, 3)$$

min-term

Example:

$$Y = (\overline{A}\overline{B}) + (\overline{A}\overline{B}) + (AB) = A+B$$

Canonical form

Simplified form

Sum of Products Rule : (SOP)

① Find the lines in truth table for which the output is "1".

② Write down the product of "ALL" input variables and invert those are "zero".

" ALL " input variables
" zero " 
" Canonical Form "

③ Sum them all .

EECE 120 - Quiz 1 - resubmit

Example:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

"SOP"

Canonical form

$$Y = \text{output} = \overline{A} \overline{B} + A \overline{B}$$
$$= A \oplus B$$

Simplified.

Lecture 7
EEE/CSE 120: Boolean Algebra II

1 Closure $(1+1=1, 1 \oplus 1=0)$

2 Associativity

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C) = (C \cdot A) \cdot B$$

3 Neutral element, (Identity)

$$AB = A$$

?

$$B = 1$$

$$A+B = A$$

$$B = 0$$

$$A \oplus B = A$$

$$B = 0$$

4 Inverse element

$$A \cdot B = 0$$

?

$$A\bar{A} \neq 0$$

$$A+B = 1$$

?

$$B = \overline{A}$$

$$A \oplus B = 1$$

?

⑤ Commutativity

$$A \cdot B = B \cdot A$$

$$A + B = B + A$$

⑥ Distributivity

$$\textcircled{1} \quad A \cdot (B + C) = A \cdot B + A \cdot C$$

$$\textcircled{2} \quad A + (B \cdot C) = (A + B) \cdot (A + C)$$

Example : "OR" gate

	A	B	$Y = A + B$
0	0	0	0
1	0	1	1 $\leftarrow \bar{A}B$
2	1	0	1 $\leftarrow A\bar{B}$
3	1	1	1 $\leftarrow AB$

$$\begin{aligned} 00 &\rightarrow 0 \\ 01 &\rightarrow 1 \\ 10 &\rightarrow 2 \\ 11 &\rightarrow 3 \end{aligned}$$

$$Y = (\bar{A}B + A\bar{B} + AB) = ?$$

Canonical form

$$\sum m(1, 2, 3)$$

min-term

$\downarrow 0$

$\underbrace{A + B}$
Simplified form

$$Y = \bar{A}B + \underbrace{A\bar{B} + AB}_{\downarrow} = \bar{A}B + A \cdot (\cancel{\bar{B} + B})$$

(Distributivity)

$$= \bar{A}B + A \cdot 1$$

(Inverse)

(Identity, Neutral)

$$= \bar{A}B + A$$

(Distributivity)

$$\Rightarrow = (\bar{A} + A) \cdot (B + A)$$

$$= 1 \cdot (B + A)$$

(Inverse)

$$= (B + A)$$

(Identity)

$$= \underbrace{A + B}$$

(Commutativity)

** $A \cdot (B + C) = A \cdot B + A \cdot C$*

** $A + (B \cdot C) = (A + B) \cdot (A + C)$*

Example : $F(a, b, c) = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$

$$= \bar{a}bc + a\bar{b}c + ab(c + \cancel{\bar{c}})$$

$$= \bar{a}bc + \underbrace{a\bar{b}c + ab}$$

$$= \bar{a}bc + a \cdot (\bar{b}c + b)$$

$$= \bar{a}bc + a \cdot (\bar{b} + b) \cdot (c + b)$$

$$= \bar{a}bc + a \cdot (c + b)$$

$$= \underbrace{\bar{a}bc + a \cdot c}_{+ a \cdot b}$$

$$= (\bar{a}b + a) \cdot c + a \cdot b$$

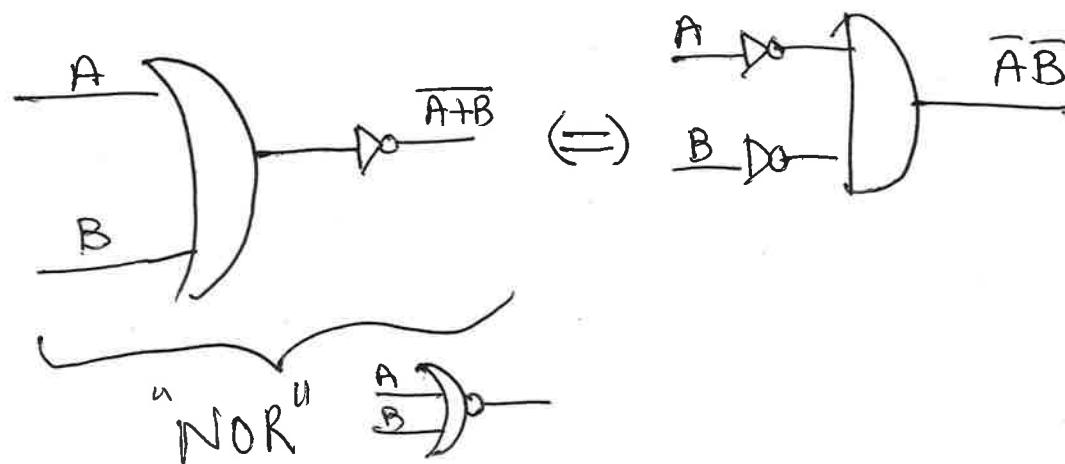
$$= (\cancel{\bar{a}} + a) (\cancel{b} + a) \cdot c + a \cdot b$$

$$= b \cdot c + a \cdot c + a \cdot b$$

DeMorgan's Law:

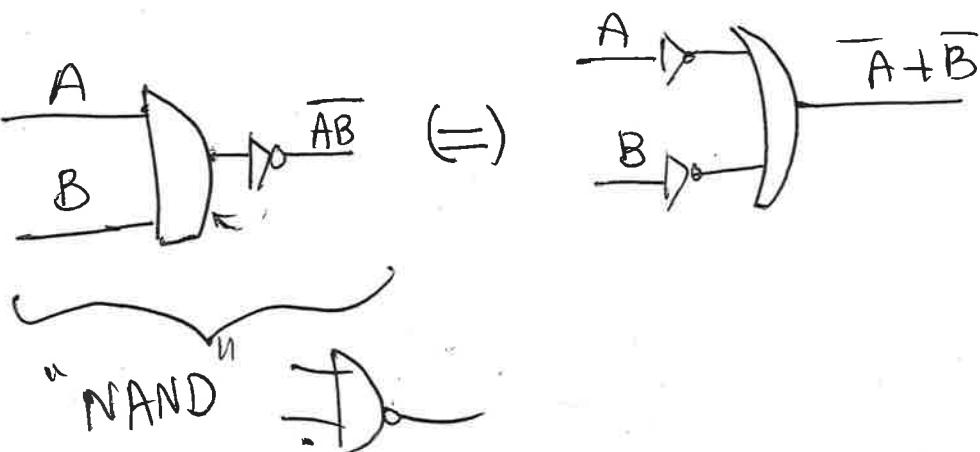
①

$$\overline{A+B} = \overline{A}\overline{B}$$

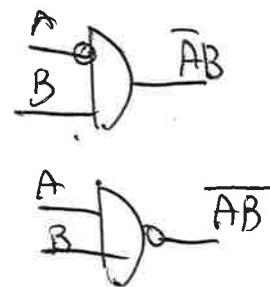


②

$$\overline{AB} = \overline{A} + \overline{B}$$



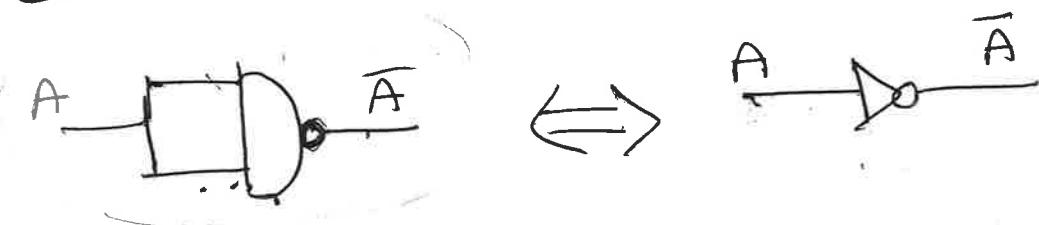
$$\overline{AB} \neq \overline{A}\overline{B}$$



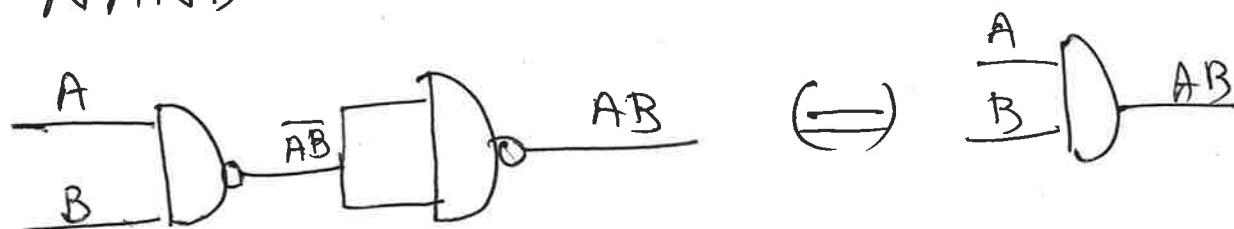
Q: Can we use "NAND" gates or "NOR" gates to build any circuit? YES!

"NAND":

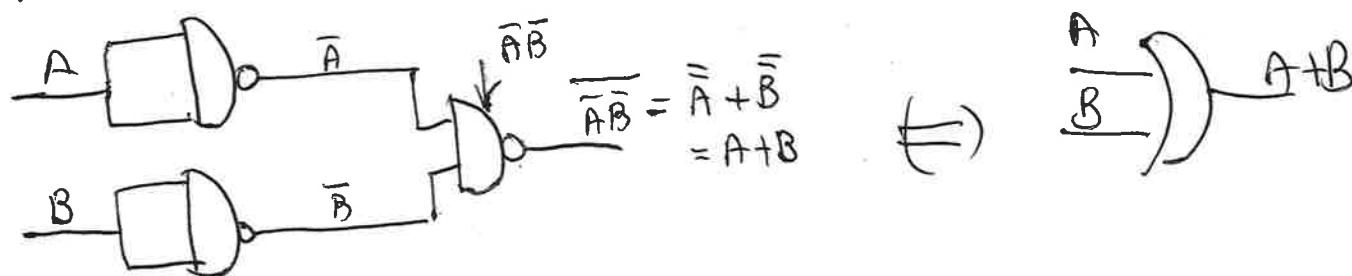
- ① NAND to build a NOT gate:



- ② NAND to build AND



- ③ NAND gate to build OR



Def: We call a gate "Functionally complete" if we can use that gate to build "AND", "OR", "NOT" gates.

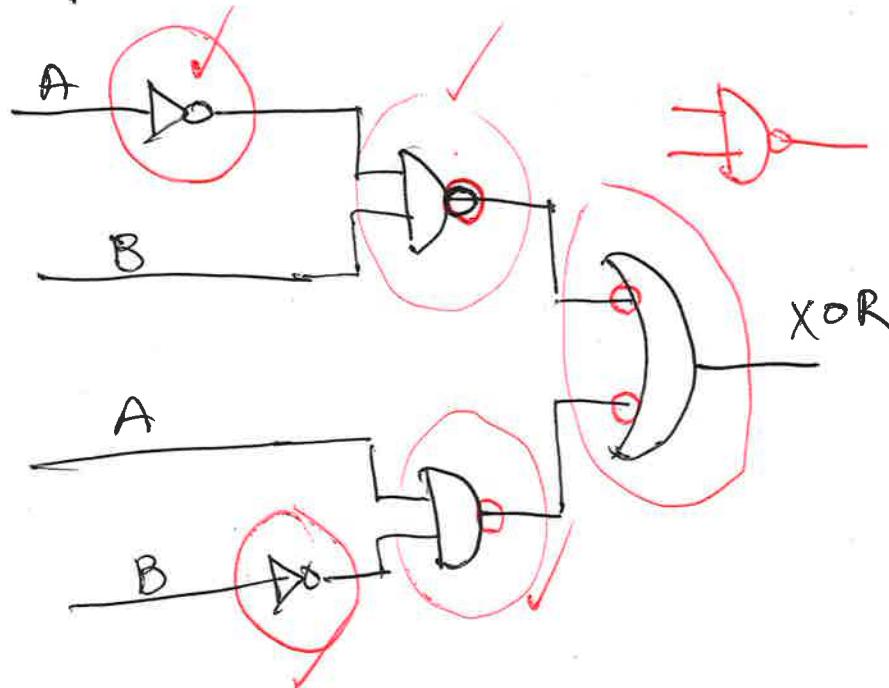
Example : NAND is functionally complete.

NOR is functionally complete.

Example : XOR gate \rightarrow build XOR gate in terms of NAND gates.

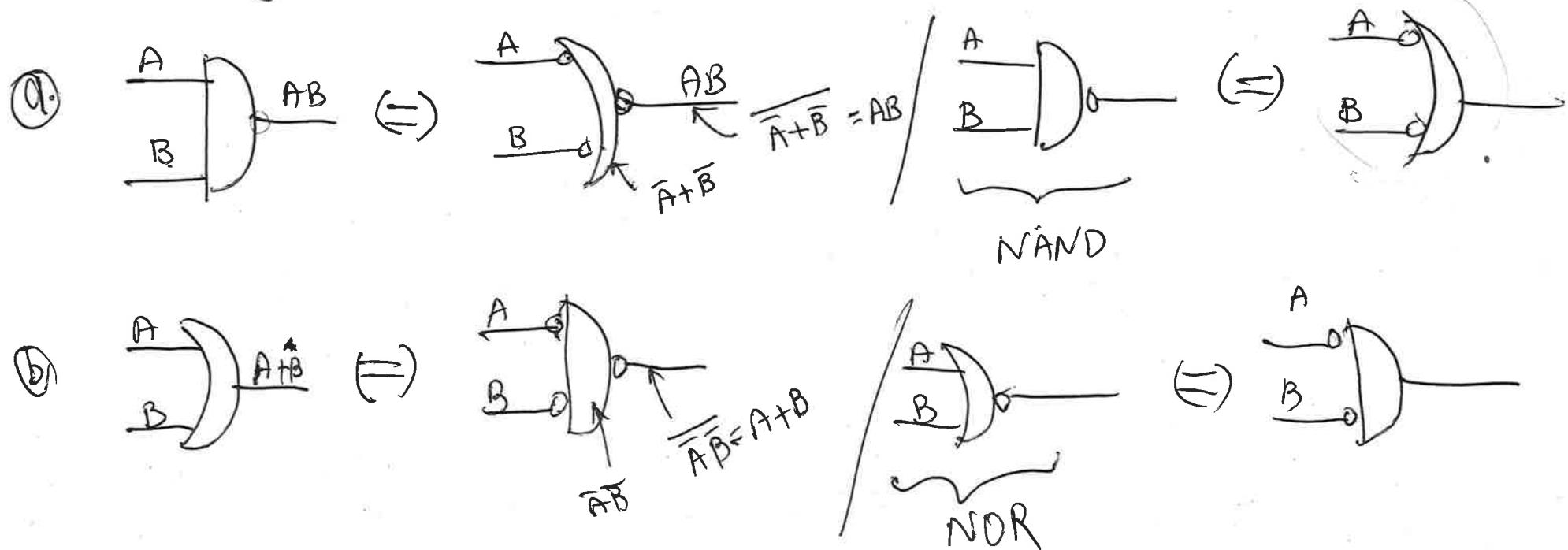
A	B	XOR
0	0	0
0	1	1 $\leftarrow \bar{A}B$
1	0	1 $\leftarrow A\bar{B}$
1	1	0

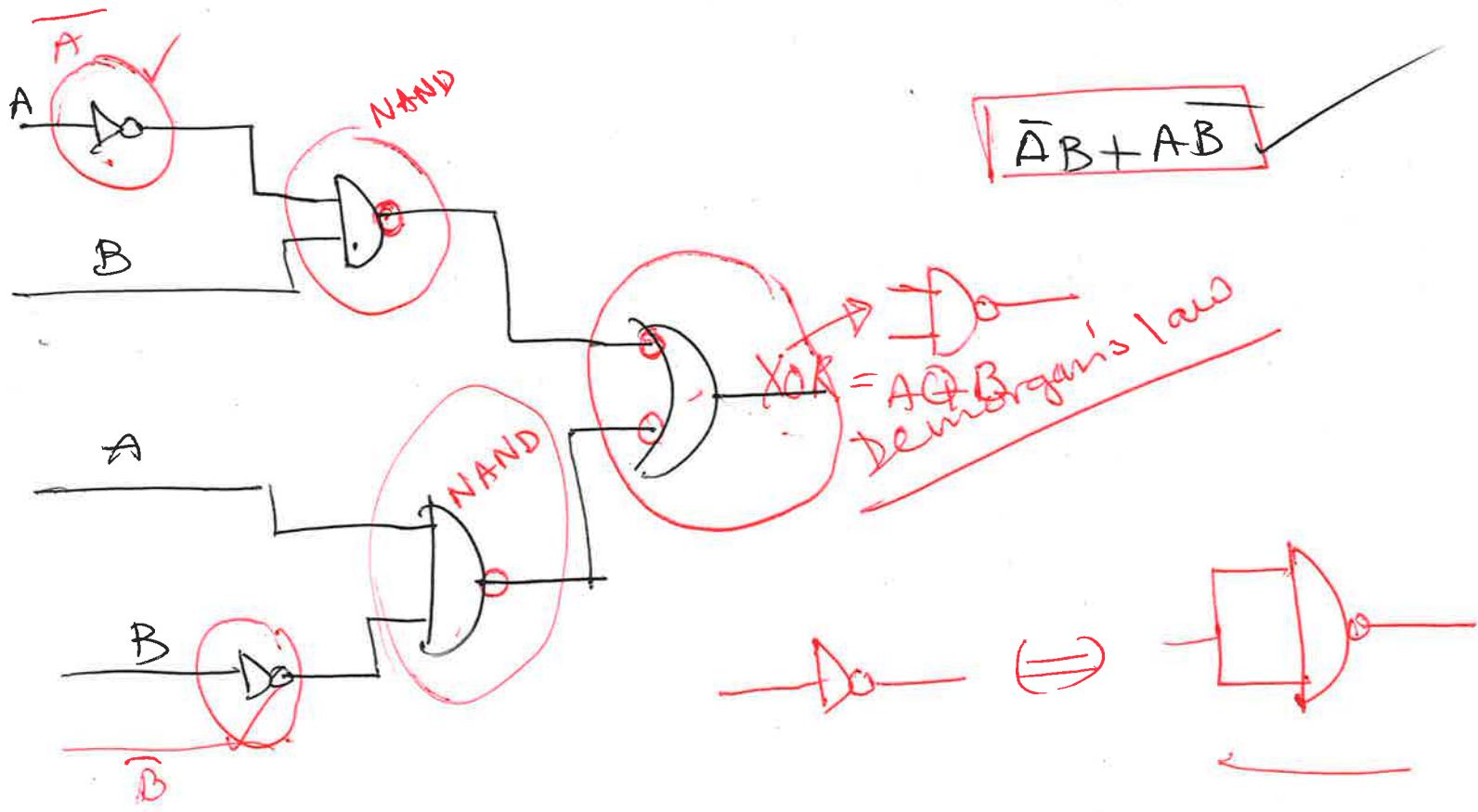
$$\text{XOR} = \bar{A}B + A\bar{B}$$



Steps to build a circuit in terms of "NAND" gates or "NOR" gates:

- ① Find the equation representing the gate or gates
- ② Draw the equation w/o paying attention to the NAND or NOR.
- ③ Play the bubble game :

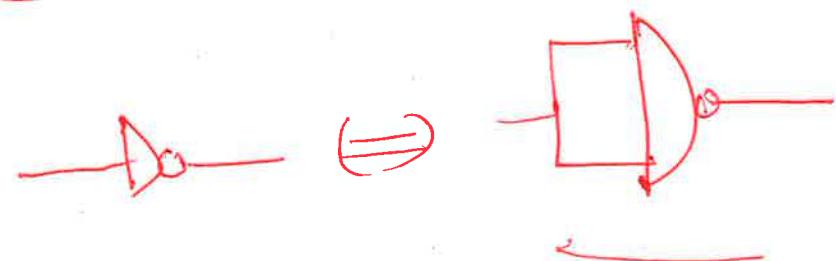




$$\bar{A}B + A\bar{B}$$

$$\text{XOR} = \bar{A}B + A\bar{B}$$

= ~~AB~~ De Morgan's law



$$\bar{A}B + A\bar{B} = A \oplus B$$

$$\bar{A}$$

$$\bar{B}$$

$$A$$

$$\bar{B}$$

Lecture 8:

EEE/CSE 120:

"Sum of Product" & Product of sum

- ① HW2 is due today → HW3 is up tonight
- ② Lab 0 is due today
- ③ office hours T/TH 9:30 - 10:15 AM
- ④ Short Quiz 1 is on Thursday

Example: NAND gates , NOR gates functionally complete

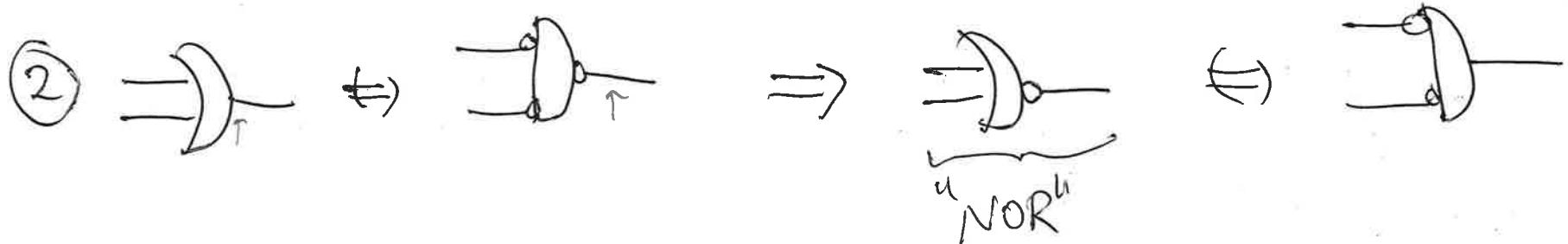
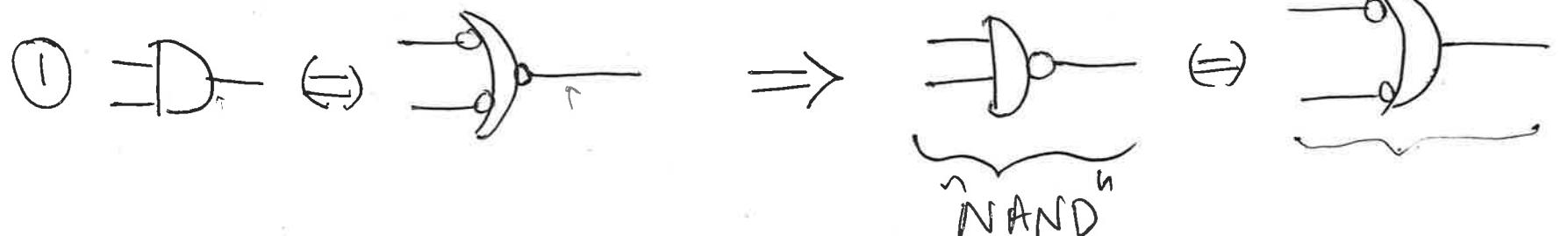
How to build any circuit using NAND /NOR gates!

1 write down the function using truth table and draw it "normally"

2 play the bubble game

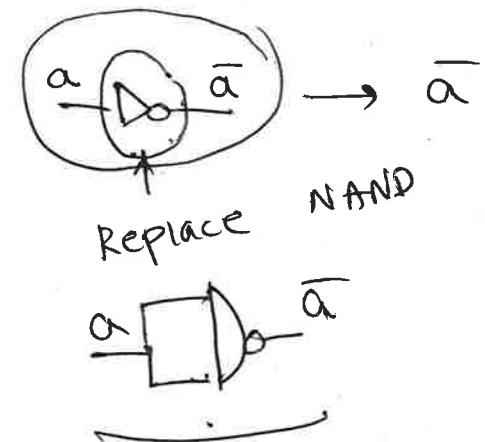
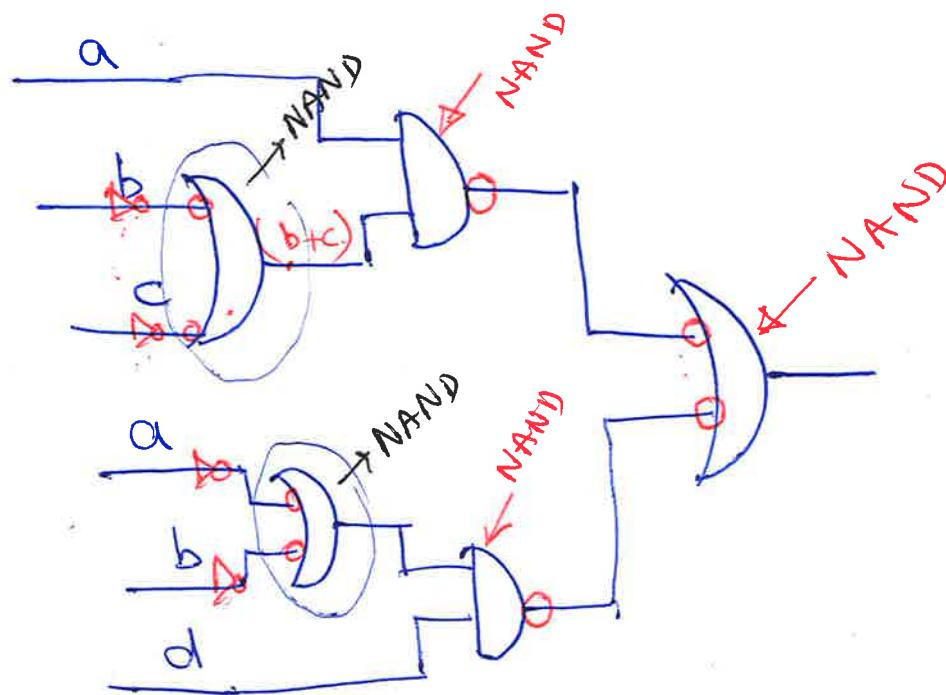
$$a \odot \bar{a} \Leftrightarrow a \triangleright \bar{a}$$

Rule:

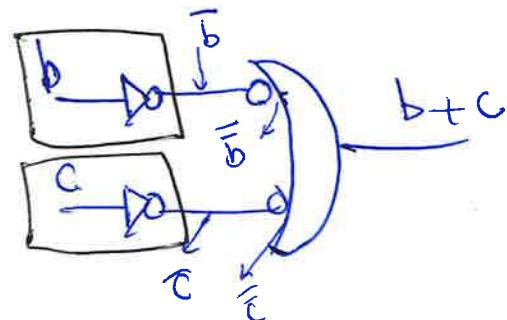


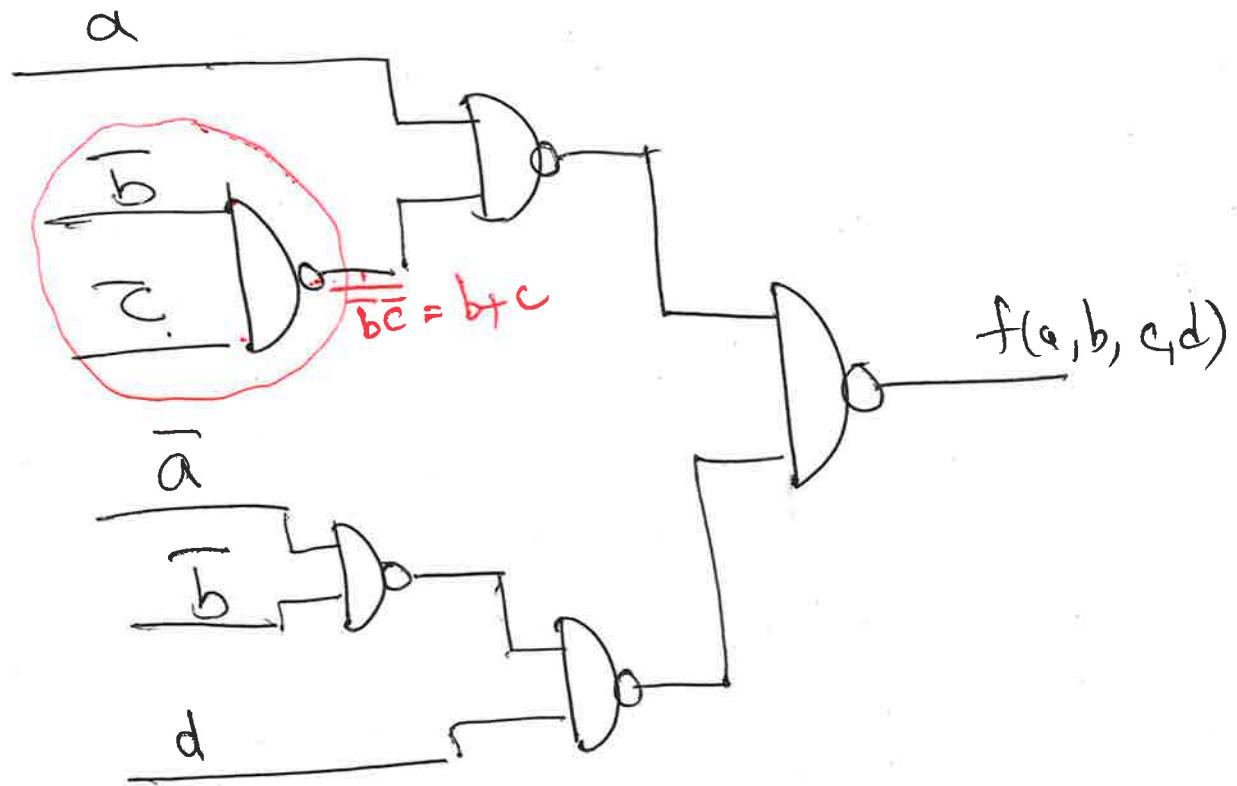
Example : Assume we have access to all inputs as well as their complement, draw the following circuit using only "NAND" gates!

$$f(a, b, c, d) = \underline{a(b+c)} + \underline{d(a+b)}$$

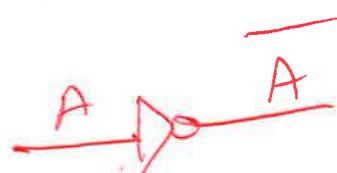
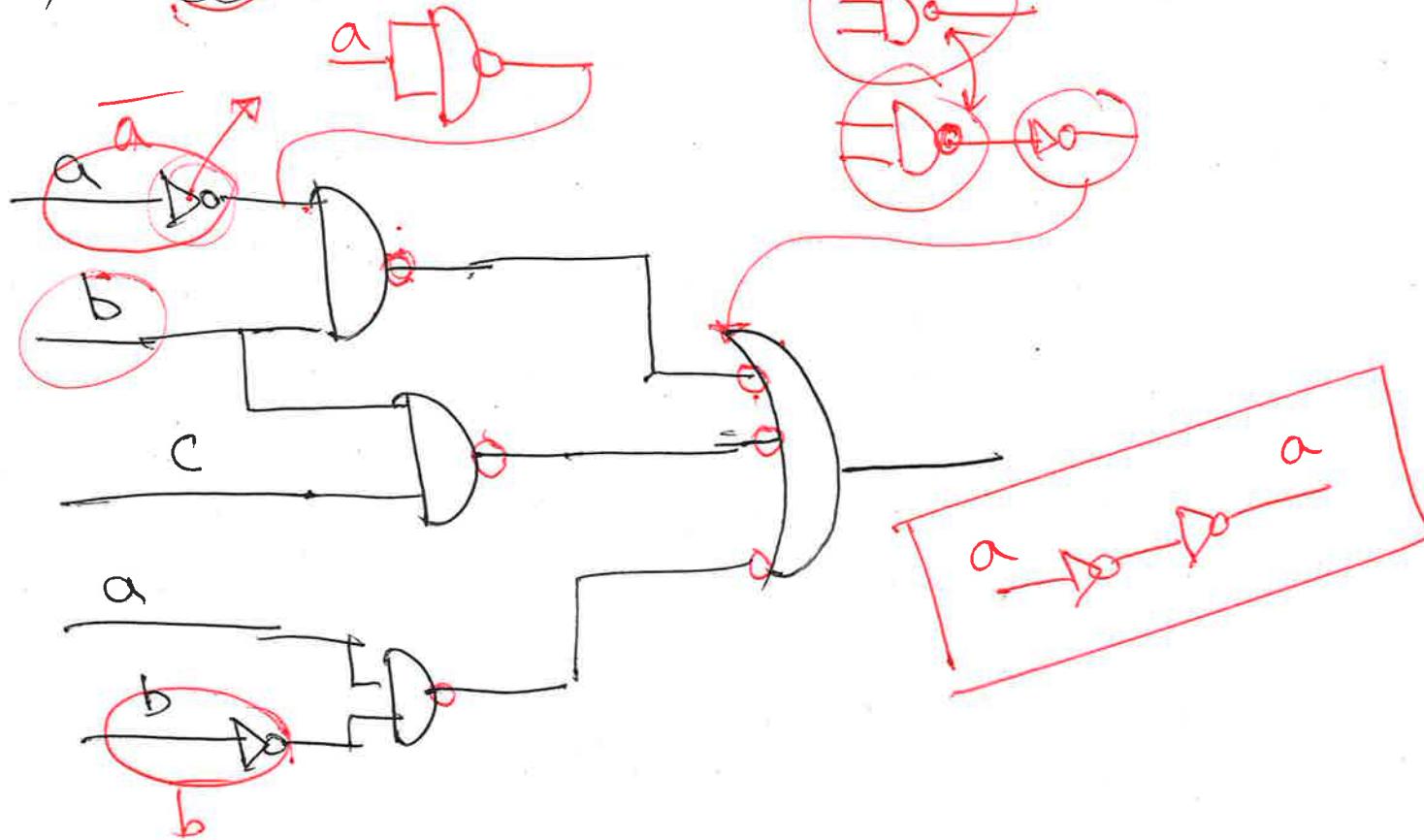


$$\overline{b+c}$$

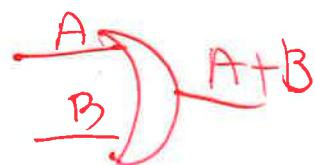




$$f(a, b, c) = \overline{ab} + bc + \overline{ab}$$



$$\overline{AB} = \overline{A} + \overline{B}$$



How to do product of sum:

- ① look at the lines of the truth table for which function is "0".
- ② write the sum of the input variables and invert those which are "one".
- ③ multiply all the sums.

$$f(a,b,c) = \underbrace{ab\bar{c}} + \underbrace{a\bar{b}\bar{c}} + \underbrace{\bar{a}\bar{b}\bar{c}} \quad (\text{canonical})$$

$$f(a,b,c) = a\bar{b}\bar{c} + \underbrace{bc}$$

Example:

line #	A	B	C	Y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

OUTPUT

pos?

$$(A + B + \bar{C})$$

$$(\bar{A} + B + C)$$

$$Y = (A + B + \bar{C})(\bar{A} + B + C)$$

$$Y = \prod M(1, 4)$$

max-term

Max-term repr.

$$\sum_{i=1}^n a_i = a_1 + a_2 + \dots + a_n$$

$$\prod_{i=1}^n a_i = a_1 \cdot a_2 \cdot \dots \cdot a_n$$

$$1+2+\dots+n = \sum_{i=1}^n i$$

$$n! = 1 \times 2 \times \dots \times n = \prod_{i=1}^n i$$

Example : $F(a,b,c) = \bar{a}\bar{b}\bar{c} + \bar{a}c + \bar{b}c$

- a) write the Canonical SOP.
- b) write the Canonical POS.
- c) write the function as min-term expression.
- d) write the function as max-term expression.

$$(1+1)^{+1} = 1$$

line#	a	b	c	$Y = F(a,b,c)$
0	0	0	0	1 $\leftarrow \bar{a}\bar{b}\bar{c}$
1	0	0	1	1 $\leftarrow \bar{a}\bar{b}c$
2	0	1	0	0 $\leftarrow (\bar{a}+\bar{b}+c)$
3	0	1	1	1 $\leftarrow \bar{a}bc$
4	1	0	0	0 $\leftarrow (\bar{a}+b+c)$
5	1	0	1	1 $\leftarrow a\bar{b}c$
6	1	1	0	0 $\leftarrow (\bar{a}+\bar{b}+c)$
7	1	1	1	0 $\leftarrow (\bar{a}+b+\bar{c})$

$$F(a,b,c) = \bar{a}\bar{b}\bar{c} + \bar{a}c + \bar{b}c$$

$$\begin{aligned} a=0, b=0, c=0 \rightarrow F &= \underbrace{1 \times 1 \times 1}_1 + \underbrace{1 \times 0}_0 + \underbrace{1 \times 0}_0 \\ &= 1 \end{aligned}$$

$$\begin{aligned} a=0, b=1, c=1 \rightarrow F &= \underbrace{1 \times 1 \times 1}_1 + \underbrace{1 \times 1}_1 + \underbrace{0 \times 1}_0 \\ &= 1 \end{aligned}$$

$$\underbrace{1 \times 0 \times 0}_0 + \underbrace{1 \times 1}_1 + \underbrace{0 \times 1}_0 = 1$$

$$\begin{aligned} a=1, b=0, c=1 \rightarrow F &= \underbrace{0 \times 0 \times 0}_0 + \underbrace{0 \times 1}_0 + \underbrace{1 \times 1}_1 \\ &= 1 \end{aligned}$$

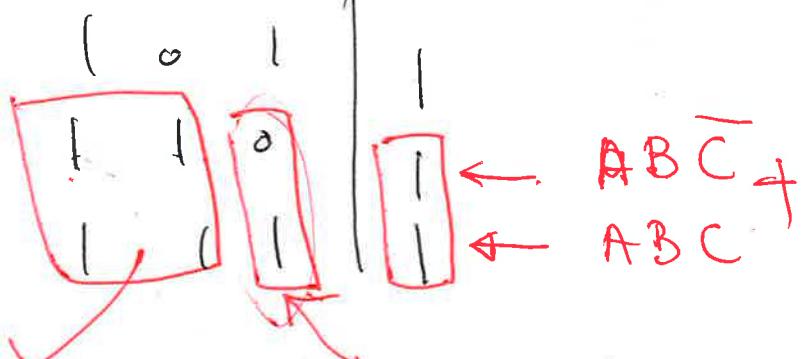
a) $Y = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}bc + ab\bar{c}$ (SOP)

b) $Y = (a+b+c)(\bar{a}+b+c)(\bar{a}+\bar{b}+c)$ (POS)

c) $F(a, b, c) = \sum m(0, 1, 3, 5)$

d) $F(a, b, c) = \prod M(2, 4, 6, 7)$

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1

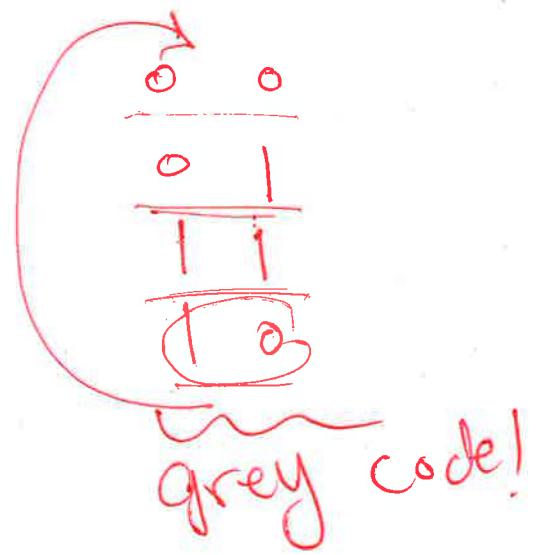


$A \oplus B$ are
not changing variable

Demorgan's
law

$$\overline{AB} + \overline{AC} = \overline{AB}$$

$$\overline{\overline{AB}} = \overline{A} + \overline{B}$$



~~\overline{AB}~~

$$\overline{A} + \overline{B} = \overline{AB}$$

~~$\overline{AB} \neq \overline{A} + \overline{B}$~~

$$y = f(x)$$

$$\begin{aligned} y &= f(5) \\ &= 5^2 + 2 \\ &= 27 \end{aligned}$$

Short Quiz 1 :

Assume both inputs and their complement are given.

Draw the following circuit using only "NAND" gates

$$F(a,b,c,d,e,f) = a(bc + de) + f(cd + be)$$

Lecture 9 :

EEE | CSE 120 : Karnough Map (k-map)

- ① HW3 is up on Canvas (due: Oct 1)
- ② Lab 1 is up on Canvas (due: Sep 28)
- ③ office hours T/TH 9:30 - 10:15 AM.
- ④ Start installing Lockdown browser for exams.

Example: line

	A	B	C	Y
0	0	0	0	1 $\leftarrow \bar{A}\bar{B}\bar{C}$
1	0	0	1	0
2	0	1	0	1 $\leftarrow \bar{A}B\bar{C}$
3	0	1	1	1 $\leftarrow \bar{A}BC$
4	1	0	0	0
5	1	0	1	1 $\leftarrow A\bar{B}C$
6	1	1	0	1 $\leftarrow ABC$
7	1	1	1	1 $\leftarrow A\bar{B}\bar{C}$

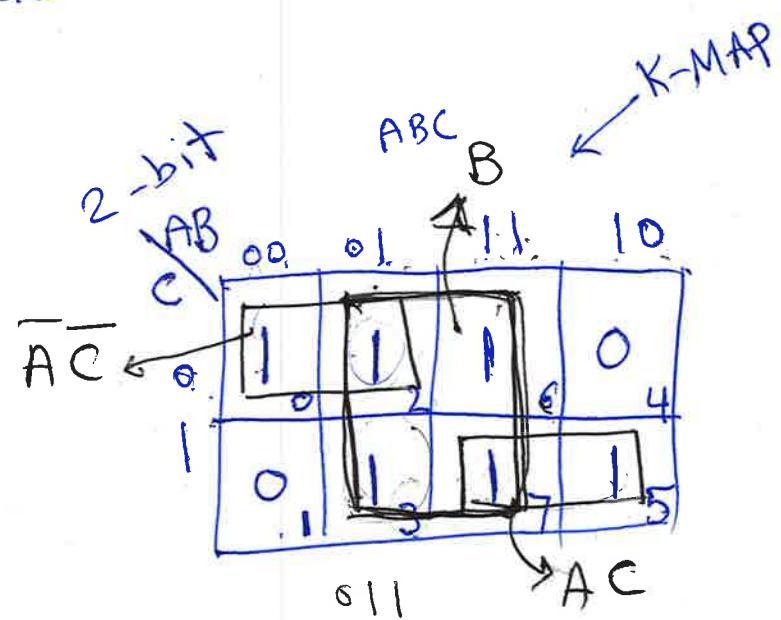
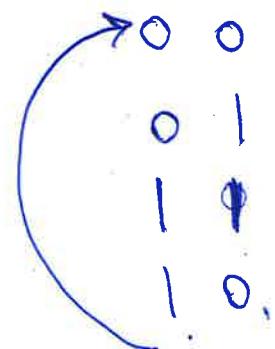
A and B are not changing
AB

C is changing variable

$$Y = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + AB\bar{C} + ABC$$

$$ABC + A\bar{B}C = AB[C + \bar{C}] = AB$$

$$Y = AC + \bar{A}\bar{C} + B$$

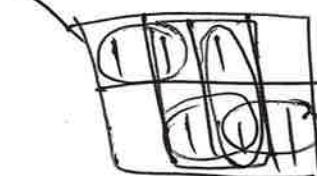


Rules of minimum sum of product:

1 Find all the ones on the map and form the K-map.

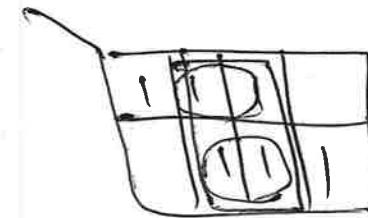
2 group all the ones into groups of 2^n elements
(vertically / Horizontally but NOT diagonally) ↴

groups are called implicants!



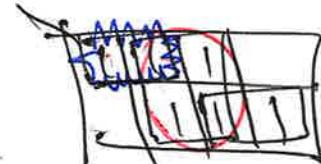
3 Find the largest groups possible

↳ prime implicants.

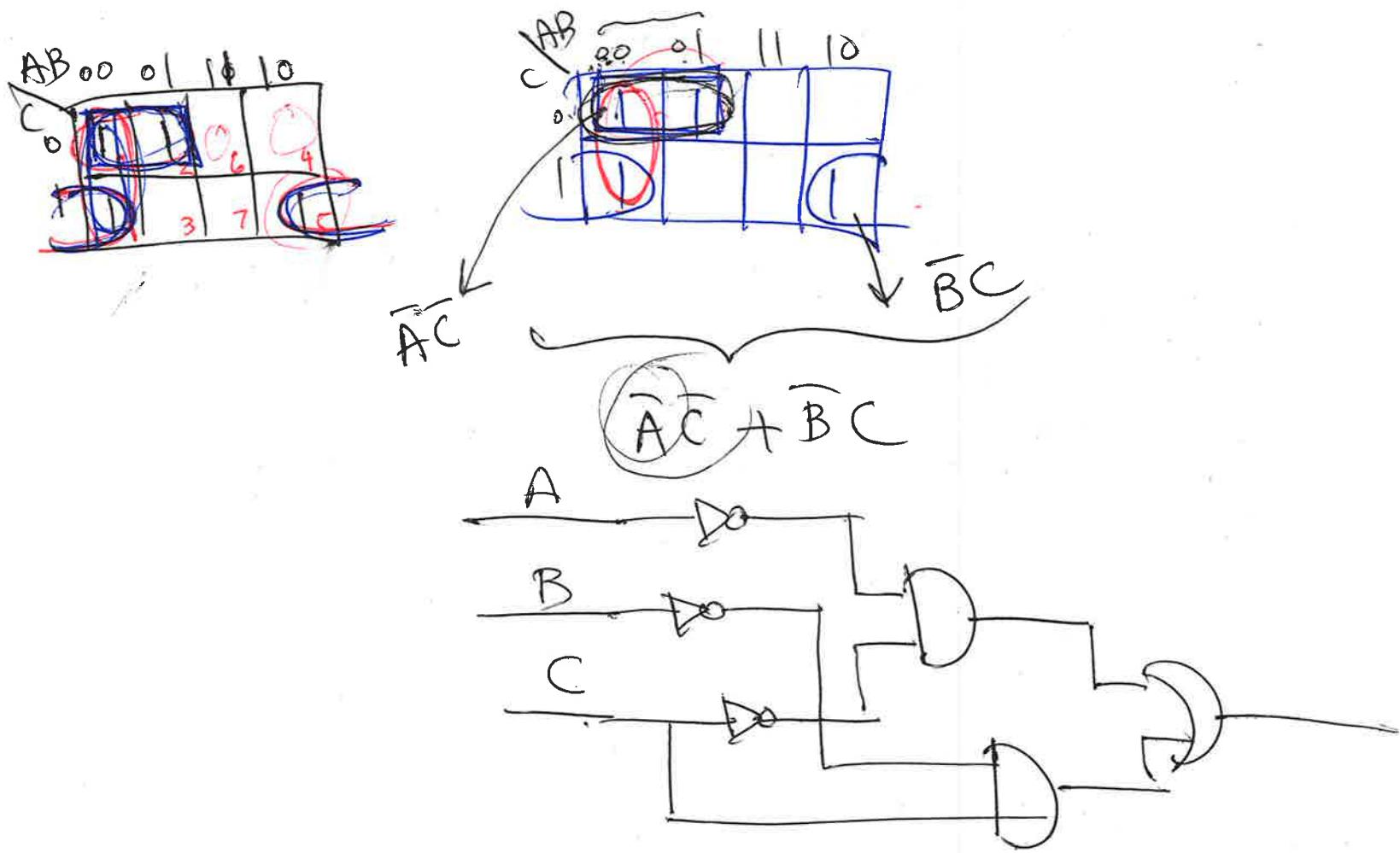


4 Find groups that have at least a single "one" that is NOT shared w/ another group.

↳ essential prime implicants



5 Add the product terms for the products of the eliminated changing variable. (essential prime implicants)



Example: Full Adder

#	A	B	ein	Cout
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1 $\leftarrow \bar{A}BC_{in}$
4	1	0	0	0
5	1	0	1	1 $\leftarrow A\bar{B}C_{in}$
6	1	1	0	1 $\leftarrow AB\bar{C}_{in}$
7	1	1	1	1 $\leftarrow ABC_{in}$

$$Y = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$$

Q: Simplify Y

- ① Boolean Alg.
- ② K-MAP

3 inputs \Rightarrow 3-variable K-MAP.

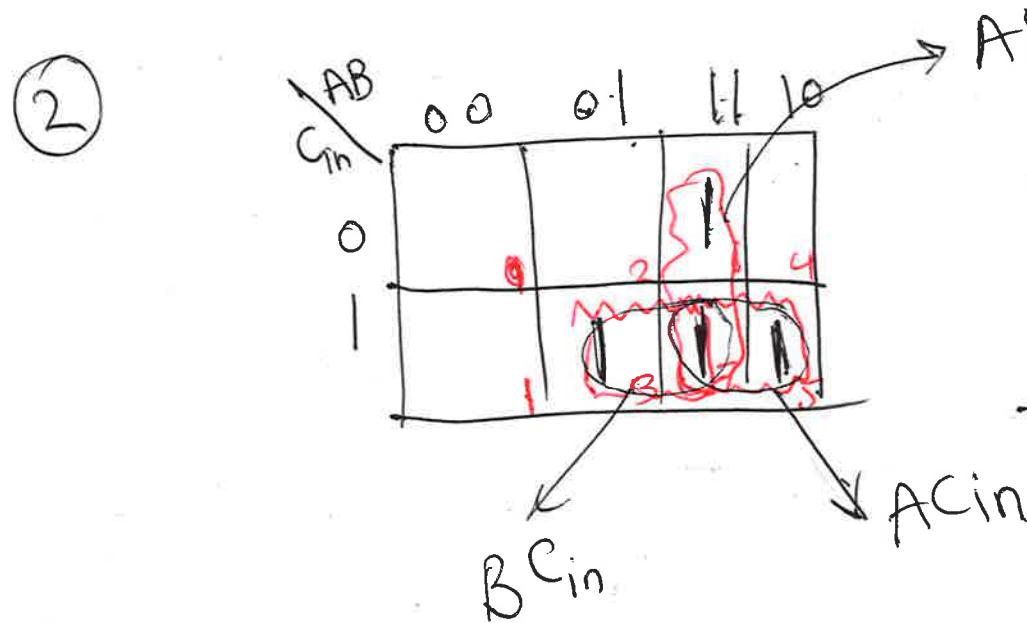
$$\sum m(3, 5, 6, 7)$$

$$\textcircled{1} \quad Y = \underline{\underline{\bar{A}B C_{in}}} + \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + \underline{\underline{ABC_{in}}} + \bar{A}\bar{B}C_{in} + \underline{\underline{ABC_{in}}} + \bar{A}BC_{in}$$

$\cancel{A + A = A}$

$$BC_{in} [\bar{A} + \cancel{A}] + AC_{in} [\bar{B} + \cancel{B}] + AB [\bar{C}_{in} + \cancel{C}_{in}]$$

$$= BC_{in} + AC_{in} + AB$$



$$Y = AB + BC_{in} + AC_{in}$$

Sep 22, 2020

Lecture 10 : Karnough Map (K-MAP) cont.

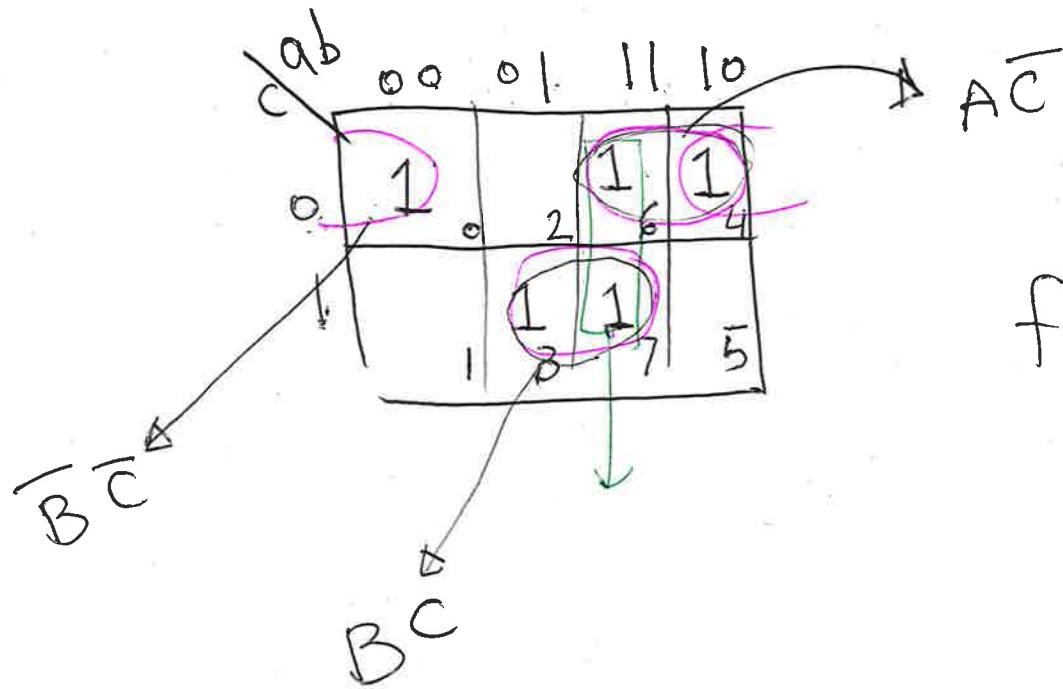
- ① Lab office hours |

Monday	5:00-6:00 PM	(AZ time)
Wednesday	12-1:00 PM	
Friday	2:30-4:00 PM	
- ② Office hours 9:30 - 10:15 Am (T/TH)
- ③ Have Quartus installed on your computer.

Example : minimum sum of product of f

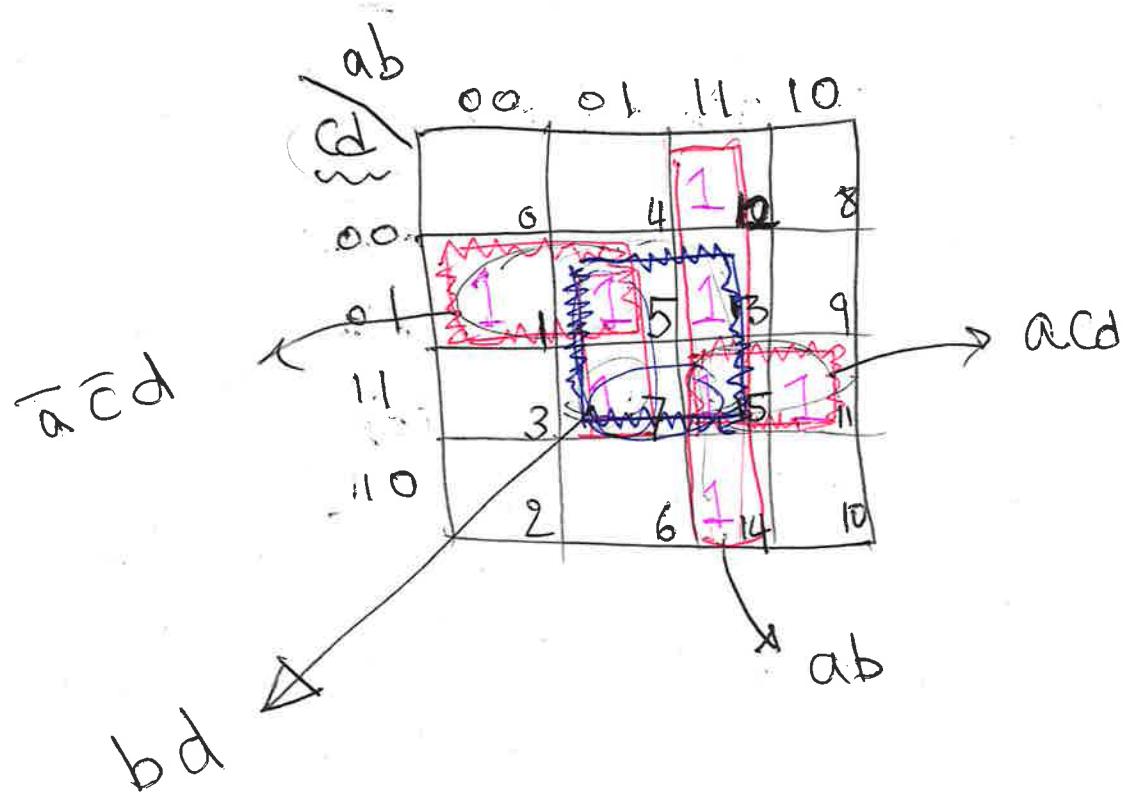
$$f(a,b,c) = \sum m(0, 3, 4, 6, 7)$$

K-MAP \rightarrow 3-var



$$f(a,b,c) = \overline{B}\overline{C} + Bc + A\overline{C}$$

Example : $f(a,b,c,d) = \sum m (1, 5, 7, 11, 12, 13, 14, 15)$



4-var K-MAP

$$11\ 00 \rightarrow 12$$

$$f(a,b,c,d) = \underline{acd} + \underline{\bar{a}\bar{c}d} + \underline{bd} + \underline{ab}$$

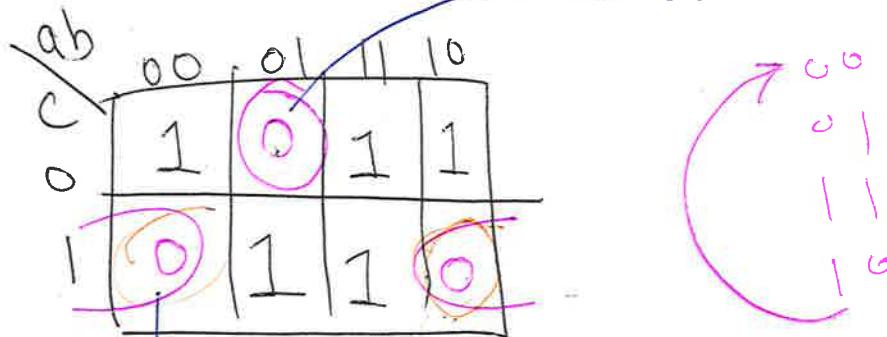
$$\overline{\overline{f}} = f$$

~~$b(\bar{a}+\bar{d})$~~

sum

minimum sum of product

Example: $f(a,b,c) = \sum m(0, 3, 4, 6, 7)$



$$\bar{b}\bar{c} \quad \bar{f}(a,b,c) = \bar{a}b\bar{c} + \bar{b}c$$

$$f(a,b,c) = \overline{\bar{f}(a,b,c)} = \overline{(\bar{a}b\bar{c} + \bar{b}c)}$$

Demorgan's law:

$$\begin{cases} \overline{A+B} = \bar{A}\bar{B} \\ \overline{AB} = \bar{A}+\bar{B} \end{cases}$$

$$f(a,b,c) = (\overline{\bar{a}b\bar{c}})(\overline{\bar{b}c}) = (a+\bar{b}+c)(\bar{b}+\bar{c})$$

How to write min Pos :

- ① group all "zeros" \rightarrow essential Prime implicants
and write the product
- ② writing the product terms and ignoring the
changing variable \rightarrow sum them up $\Rightarrow \bar{f}$
- ③ Use Demorgan's law to get $f \Rightarrow f$

Example :

1	2	3
4	5	6
7	8	9
0		

Output = 1 , if we press
multiples of 3 (including "0")

Q: Design a circuit that does that !

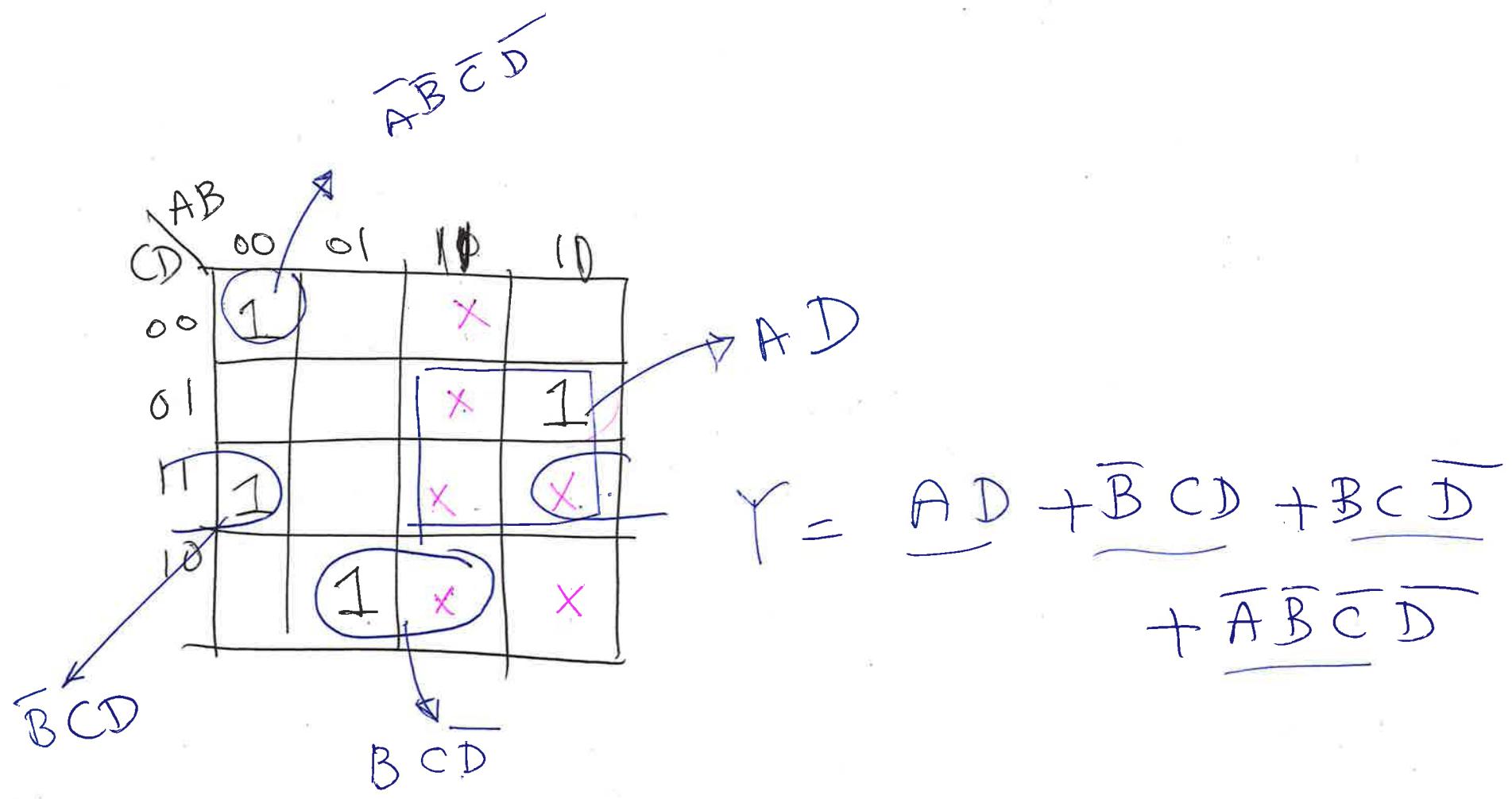
#	A	B	C	D	Y=output
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	X
11	1	0	1	1	X
12	1	1	0	0	X
13	1	1	0	1	X
14	1	1	1	0	X
15	1	1	1	1	X

SOP

$$Y = \sum m(0, 3, 6, 9) + \sum d(10, 11, 12, 13, 14, 15)$$

"don't care"

can either be "1" or "0"

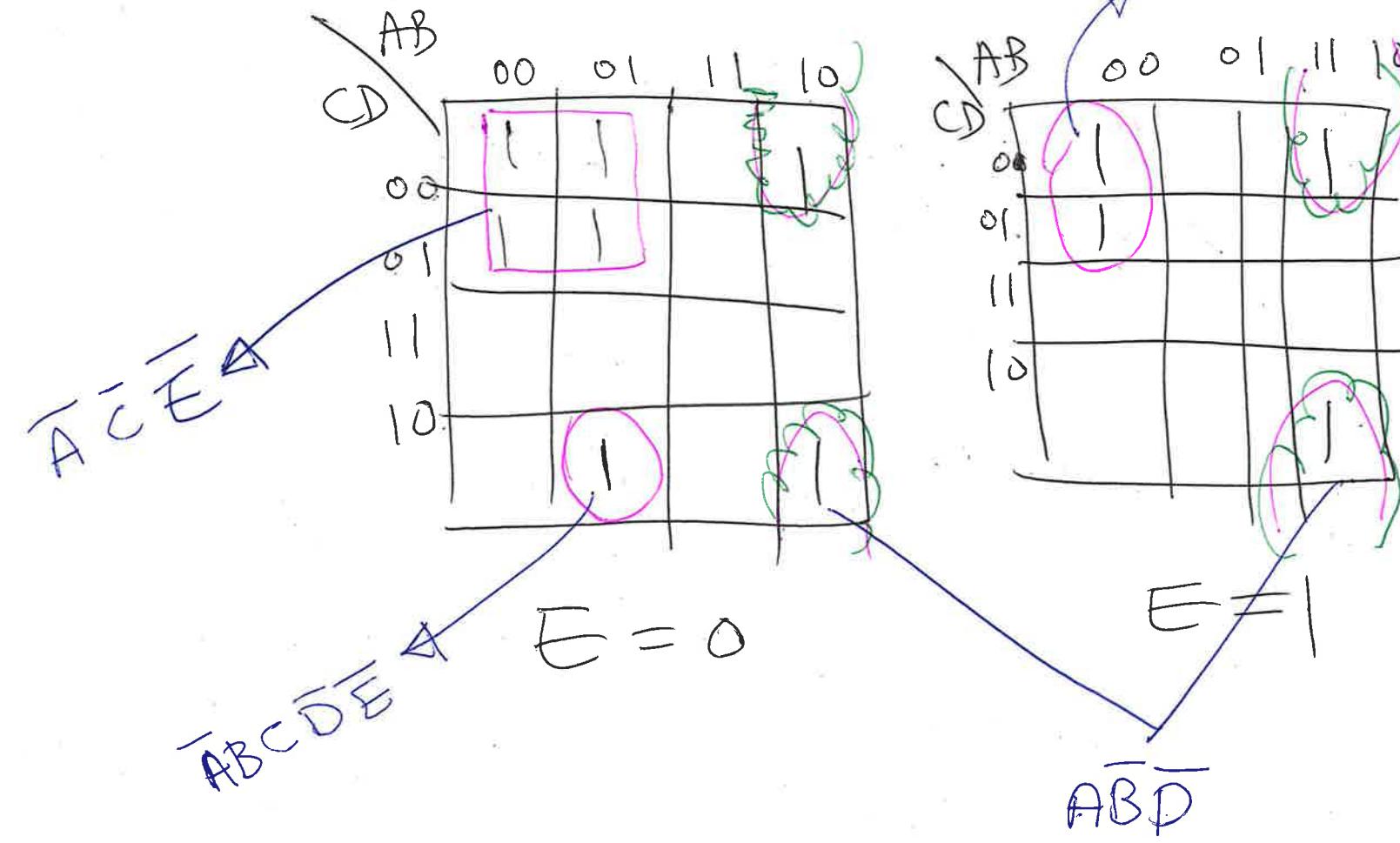


$$f(a, b, c) = \sum m(4, 6, 7) + \sum d(1, 5)$$

ab\c	00	01	11	10
0			X	
1	X			X

$$f(a, b, c) = a \quad \underline{=}$$

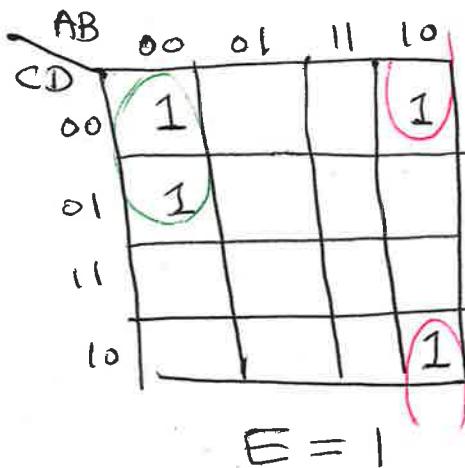
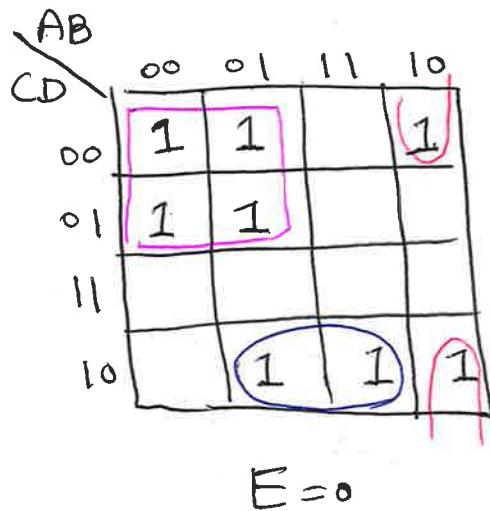
Example : $F(A, B, C, D, E)$



$$\begin{aligned} F(A, B, C, D, E) &= \overline{A}\overline{C}\overline{E} + \overline{A}\overline{B}\overline{C}\overline{D}\overline{E} + \overline{A}\overline{B}\overline{C}E \\ &\quad + \overline{A}\overline{B}\overline{D} \end{aligned}$$

Karnaugh MAP (5 variables)

Find minimum SOP for the following 5-variable K-MAP: $F(A, B, C, D, E) = ?$



$$F = \overline{A}\overline{C}\overline{E} + BC\overline{D}\overline{E} + \overline{A}\overline{B}\overline{C}E + A\overline{B}\overline{D}$$

(pay attention to color code.
Each color represent an essential prime implicant.)

Lecture 11 : Multiplexers & Decoders (Sep 24, 2020)

- ① what are the Multiplexers (Encoders) ?
- ② what are the demultiplexers (Decoders) ?
- ③ How to use them ?

Announcement :

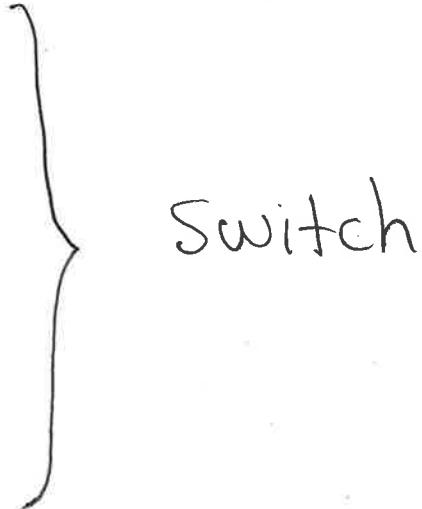
- HW3 is due Oct 1.
- Short Quiz 2 is on Oct 1.
 - It is exactly the same as short quiz 1 and must be submitted on canvas!

If (select 0)

$$Y = I_0$$

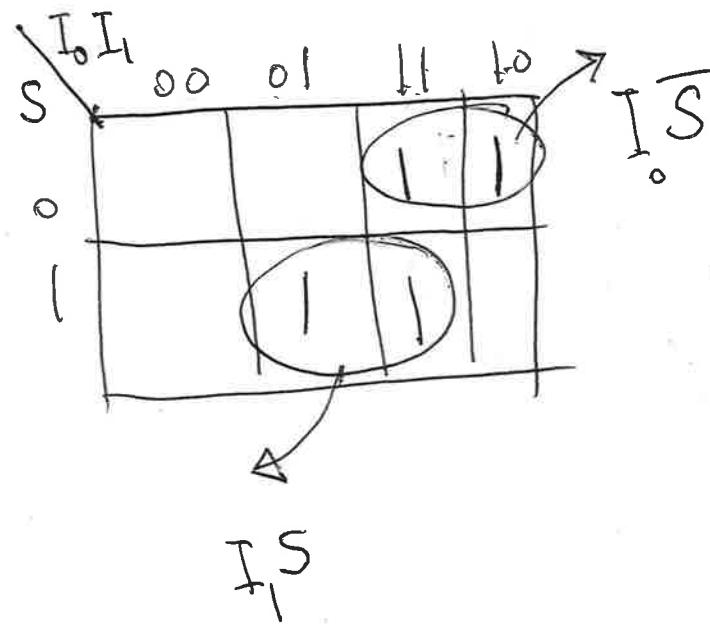
else (select 1)

$$Y = I_1$$



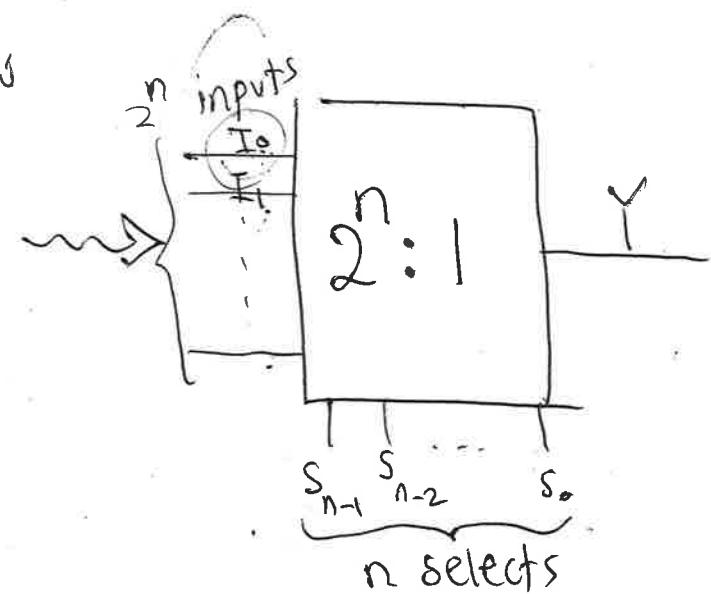
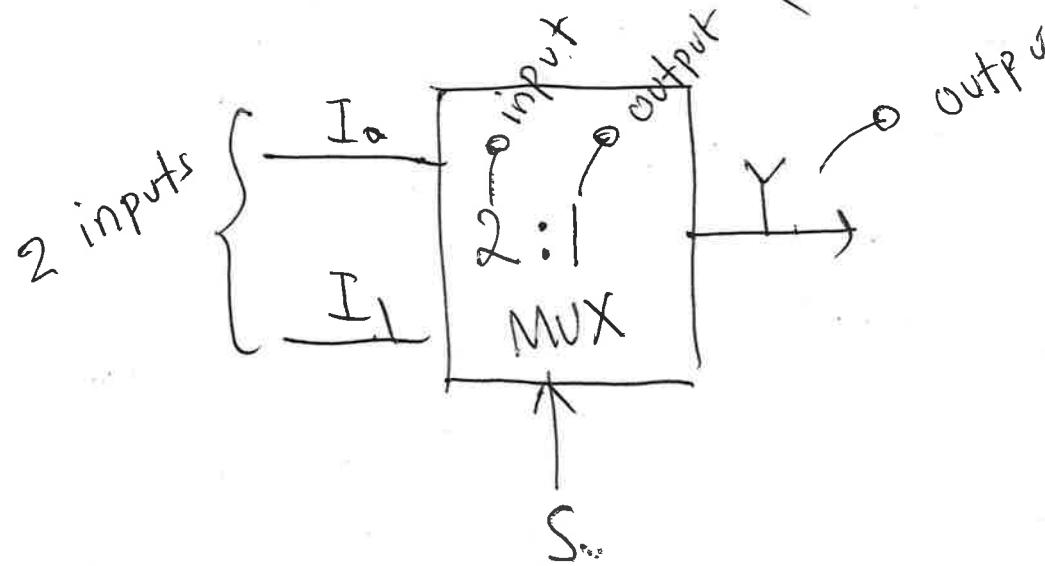
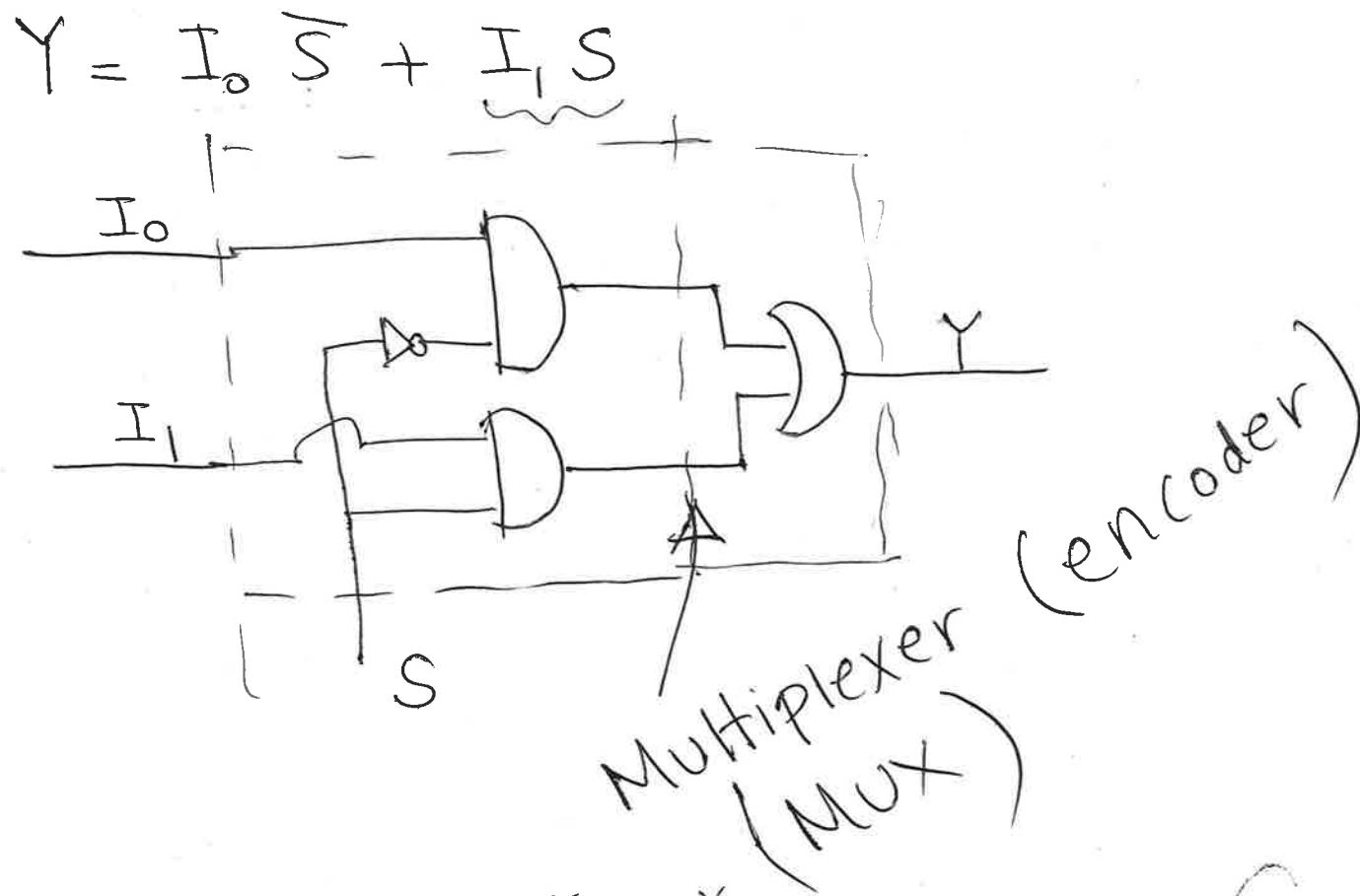
select

I ₀	I ₁	S	Y = output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1 (T)	1 ↑
1	0	0	1 ↑
1	0	1	0
1	1	0	1 ↑
1	1	1	1 ↑



$$Y = I_0 \bar{S} + I_1 S$$

binary number
represents the index



Think of Multiplexers as signal routing devices.

Application: CPU

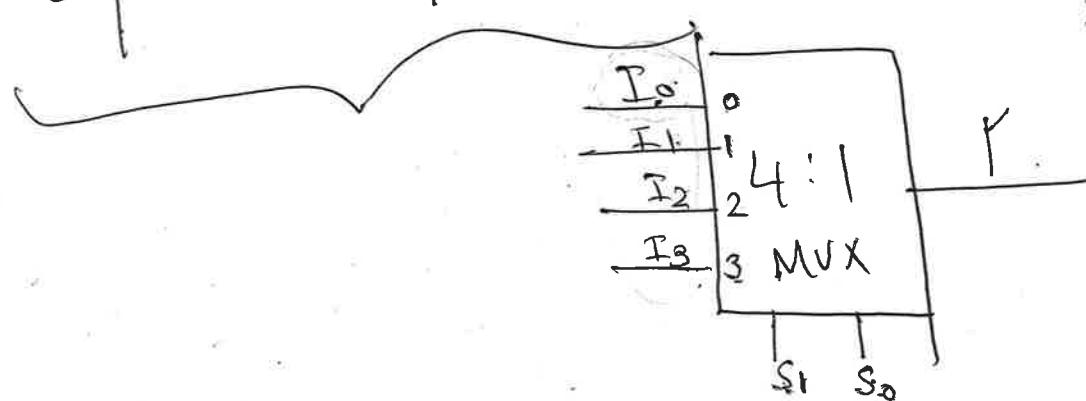
How to build $4:1$ MUX?

(I_0, I_1, I_2, I_3)

#line	MSB S_1	LSB S_0	$Y = \text{output}$
0	0	0	I_0
1	0	1	I_1
2	1	0	I_2
3	1	1	I_3

$$Y = I_0 \overline{S_1} \overline{S_0} + I_1 \overline{S_1} S_0 +$$

$$I_2 S_1 \overline{S_0} + I_3 S_1 S_0$$

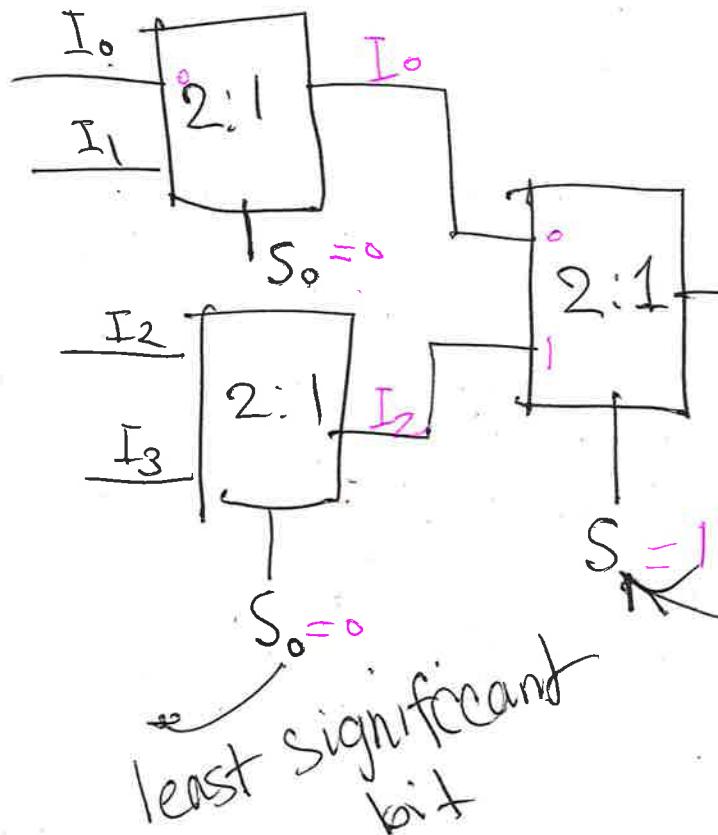


$$f(a_1, b, c) = \underbrace{\sum m(1, 2, 4, 5)}_{\text{represent the output}} + \sum d(6, 7)$$

represent the output

not input

Q: Can we build a 4:1 MUX using 2:1 Mixers?



Example:

$$\left| \begin{array}{l} S_1 = 1 \\ S_0 = 0 \end{array} \right. \rightarrow I_2$$

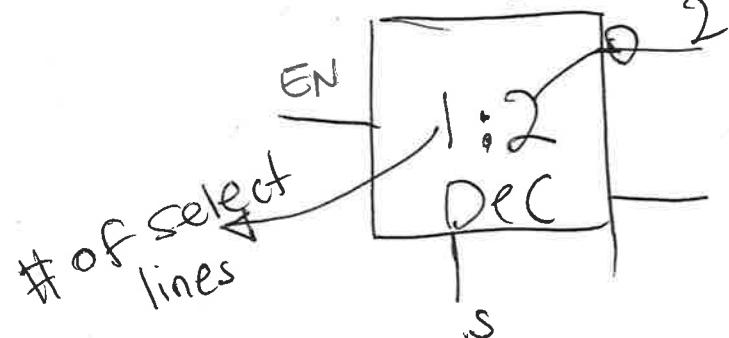
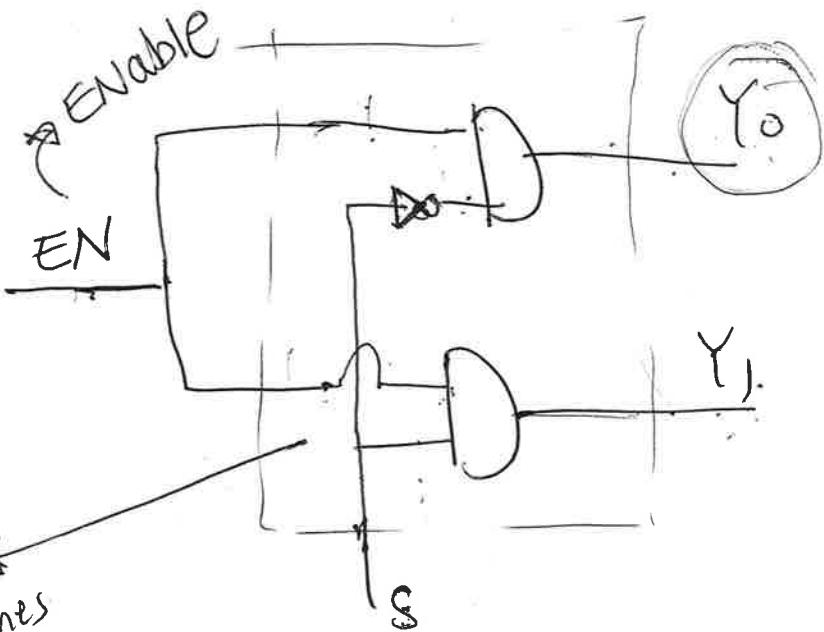
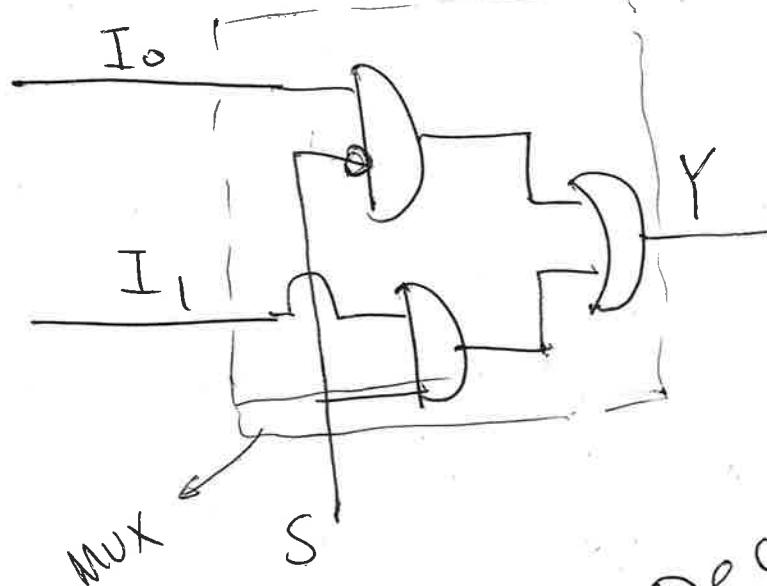
		S_1	S_0	Y
0	0			I_0
0	1			I_1
1	0			I_2
1	1			I_3

		S
0		I_0
1		I_1

Demultiplexers (Decoders)

① one input & multiple outputs

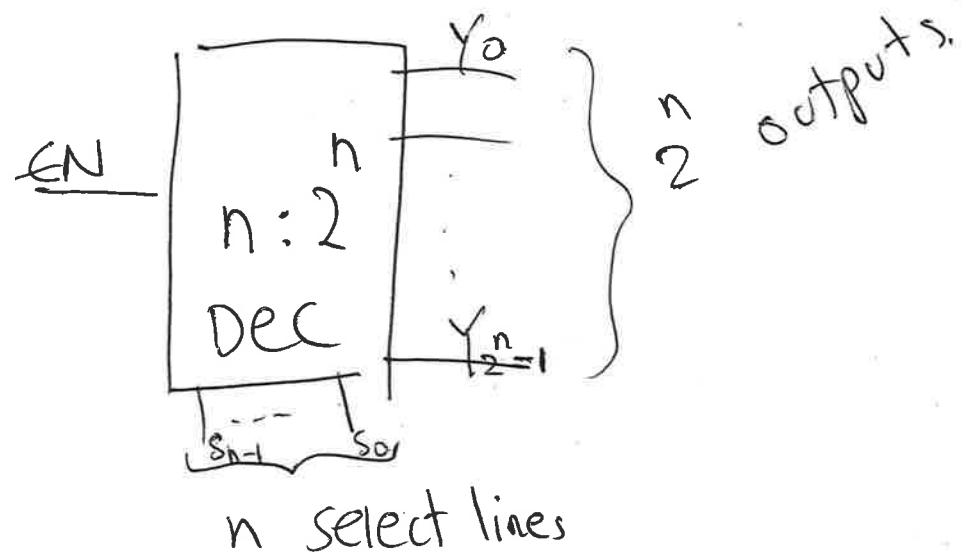
② select line(s) determine which output is active



Decoder
#select lines

$EN = 0 \Rightarrow$ Decoder is not on

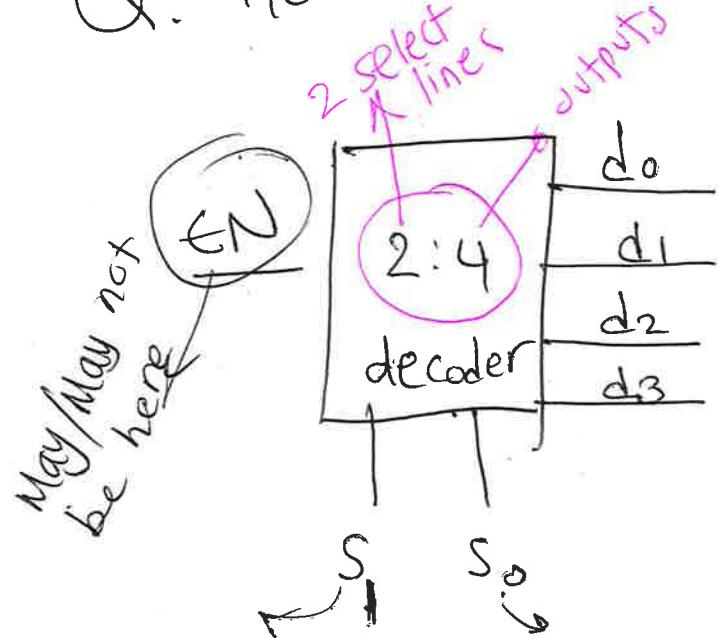
$EN = 1 \Rightarrow \begin{cases} S = 0 \Rightarrow Y_0 \\ S = 1 \Rightarrow Y_1 \end{cases}$



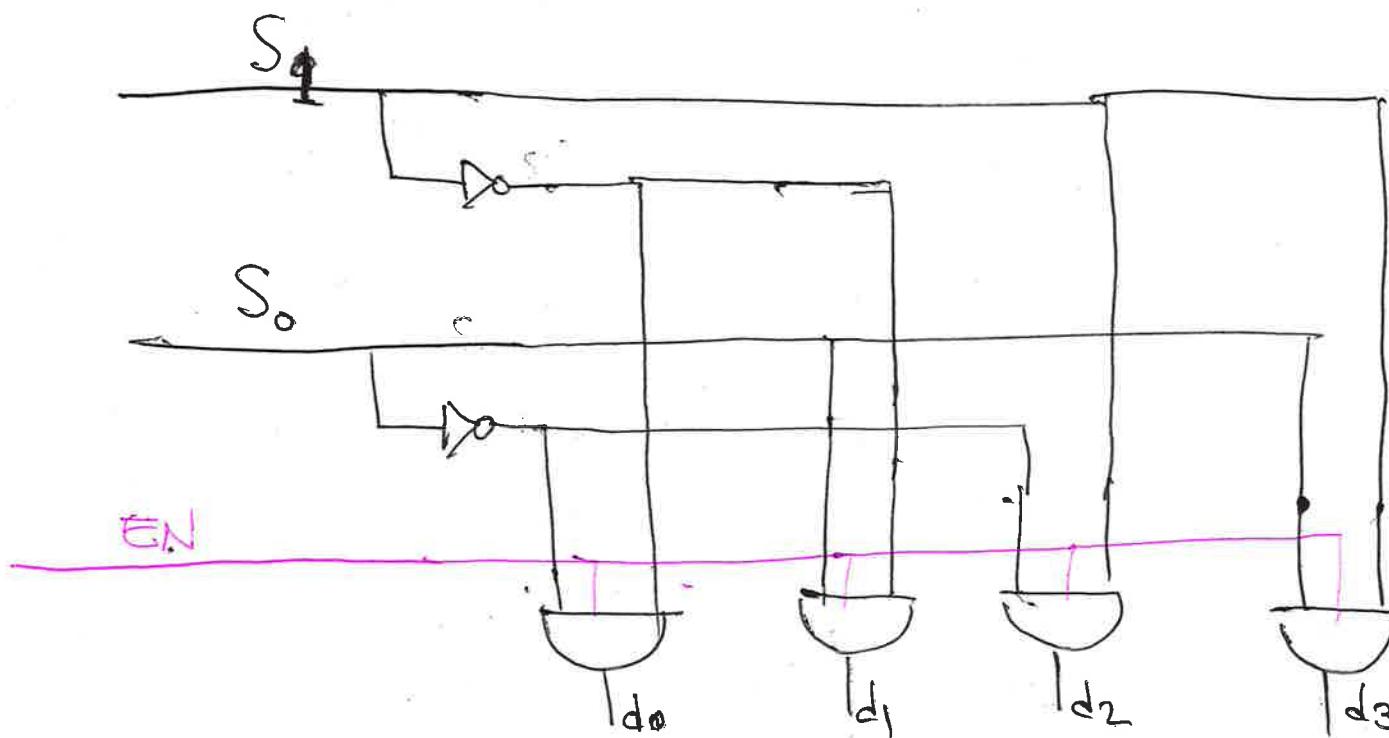
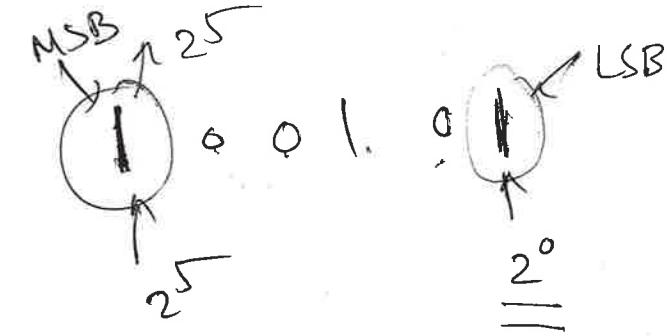
$\bar{EN}=1$ says, we generate output.

$\bar{EN}=0 \Rightarrow$ output = 0

Q: How to build 2:4 decoders?



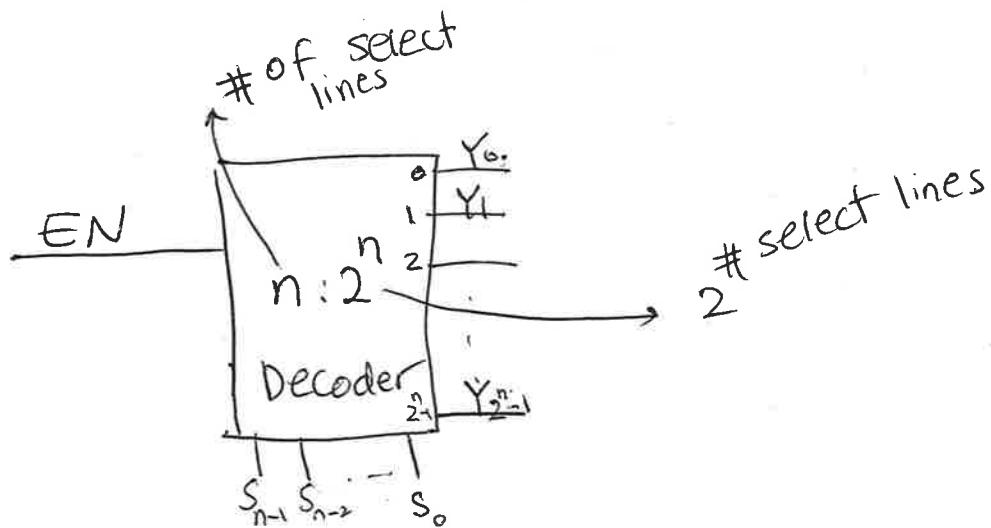
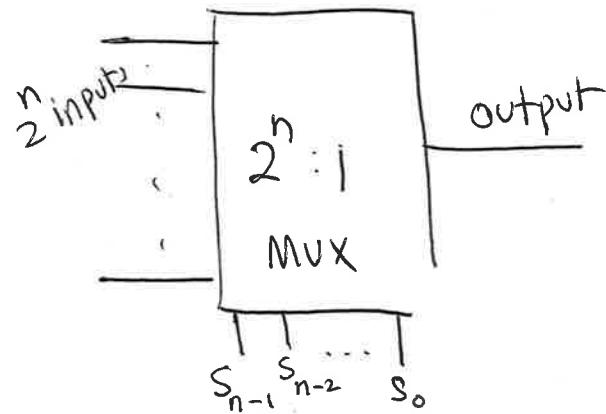
$$\begin{aligned} d_0 &\leftarrow \overline{S_1} \overline{S_0} \\ d_1 &\leftarrow \overline{S_1} S_0 \\ d_2 &\leftarrow S_1 \overline{S_0} \\ d_3 &\leftarrow S_1 S_0 \end{aligned}$$



Sep 29, 2020

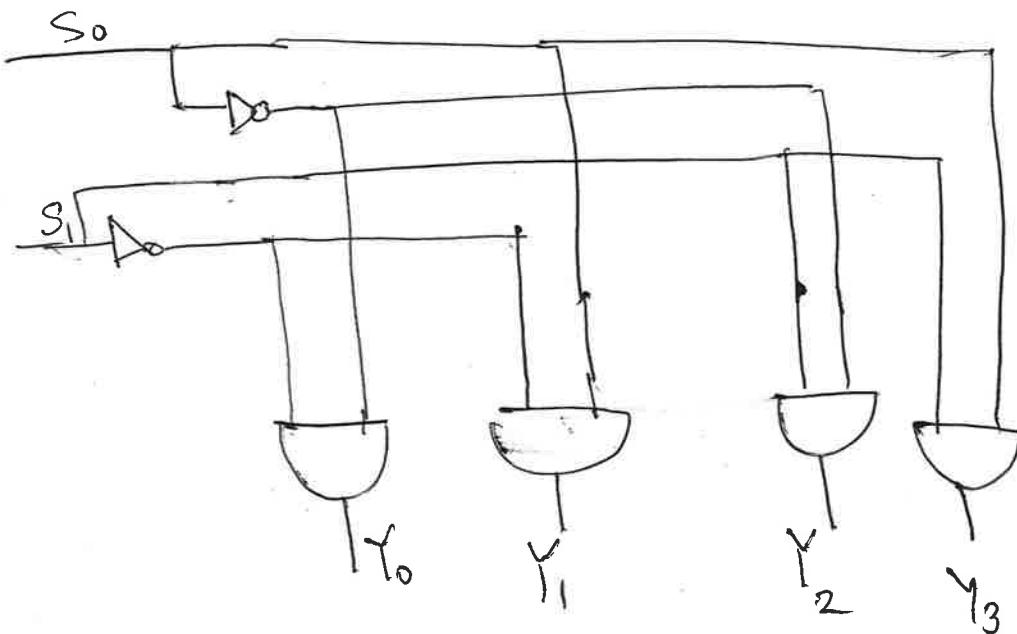
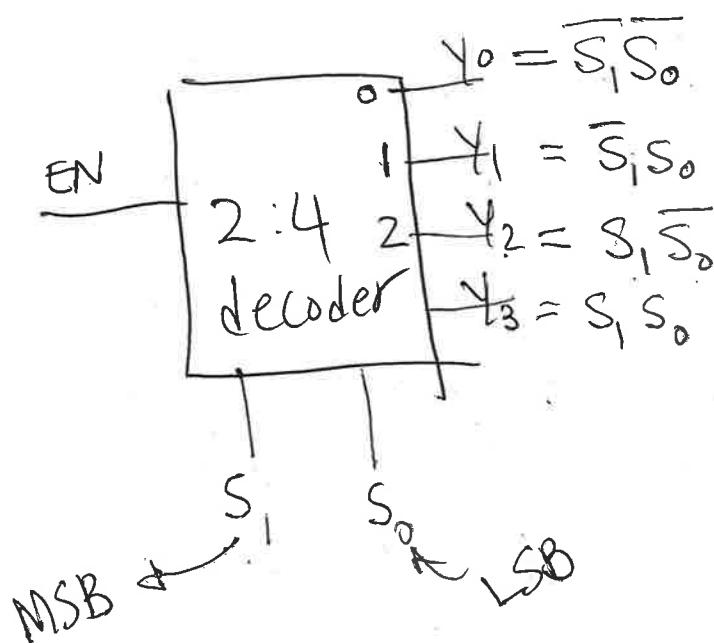
Lecture 12: What are the applications of MUXes & Decoders?

- HW 3 is due Oct 1.
- Short Quiz 2 is on Oct 1.
- Lab 1 due is extended.

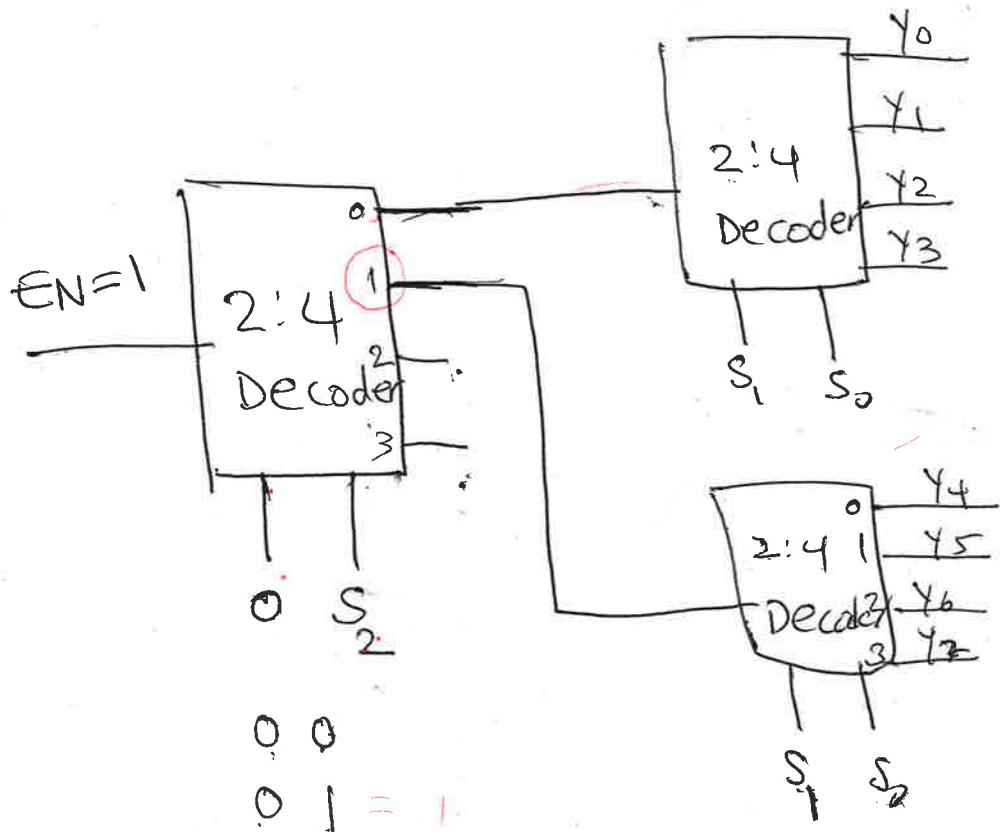


2^n inputs \Leftrightarrow only one output

Q: How to build a 2 : 4 decoder



Q: Can we build a 3:8 decoder using 2:4 decoders?



S_2	S_1	S_0	Output
0	0	0	y_0
0	0	1	y_1
0	1	0	y_2
0	1	1	y_3
1	0	0	y_4
1	0	1	y_5
1	1	0	y_6
1	1	1	y_7

$y_i \in \{0, 1\}$

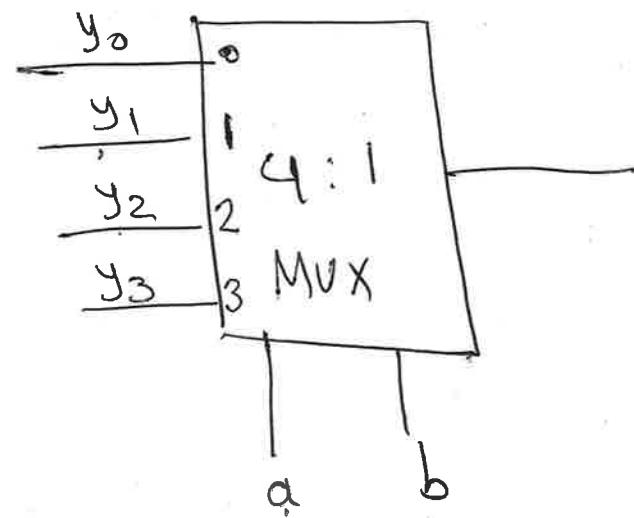
$$\underbrace{OS_2}_2 = S_2$$

Any logic function w/ "n" inputs can be implemented w/
 2^n :1 Multiplexer.

Example:

a	b	Y=output
0	0	y_0
0	1	y_1
1	0	y_2
1	1	y_3

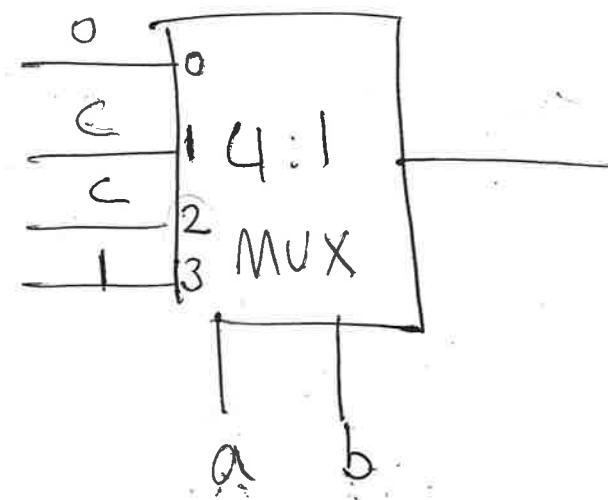
$$y_i \in \{0,1\}$$



Example : 3 inputs , output = majority voting \Rightarrow use 4:1 MUX

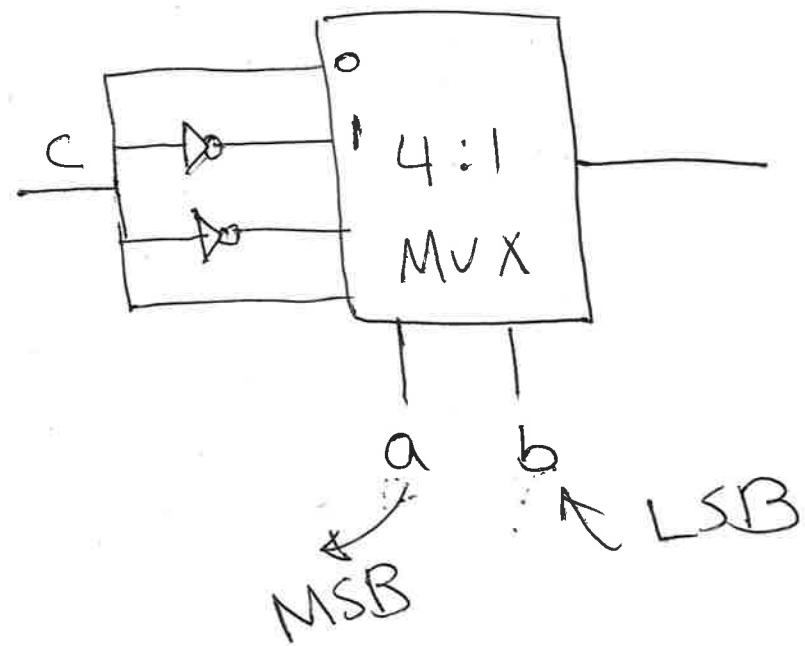
to build the circuit

a	b	c	output=Y output
0	0	0	0 0
0	0	1	0 0
0	1	0	0 C
0	1	1	1 C
1	0	0	0 C
1	0	1	1 C
1	1	0	1 C
1	1	1	1 1

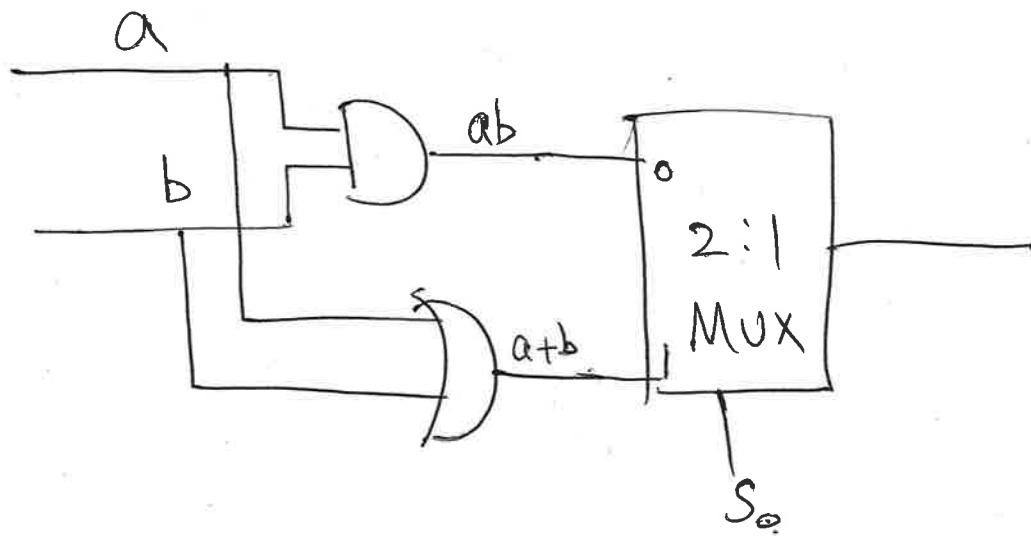


Example : (parity)

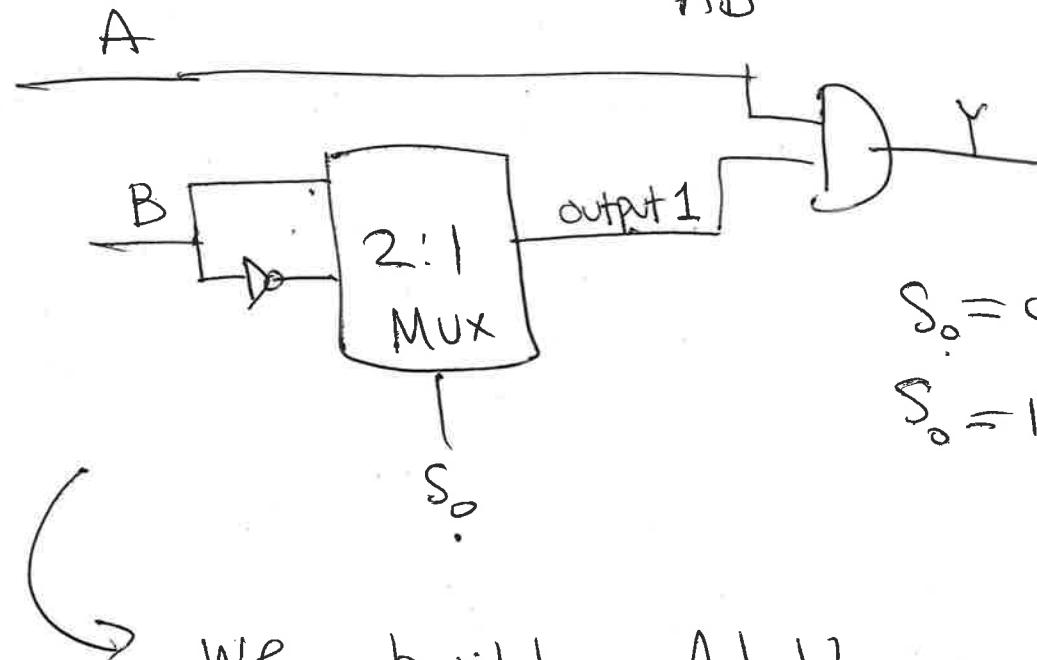
a	b	c	$y = \text{output}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



Example : use a mux to build a circuit that allows one of the operations ("AND", "OR") to be chosen and applied to an input.



Example: Use a Mux to build a circuit that evaluates either "A and B" or "A and not B".



$$S_0 = 0 \Rightarrow \text{output 1} = B \Rightarrow Y = AB$$

$$S_0 = 1 \Rightarrow \text{output 1} = \overline{B} \Rightarrow Y = A\overline{B}$$

we build

ALU logic unit

Arithmetic

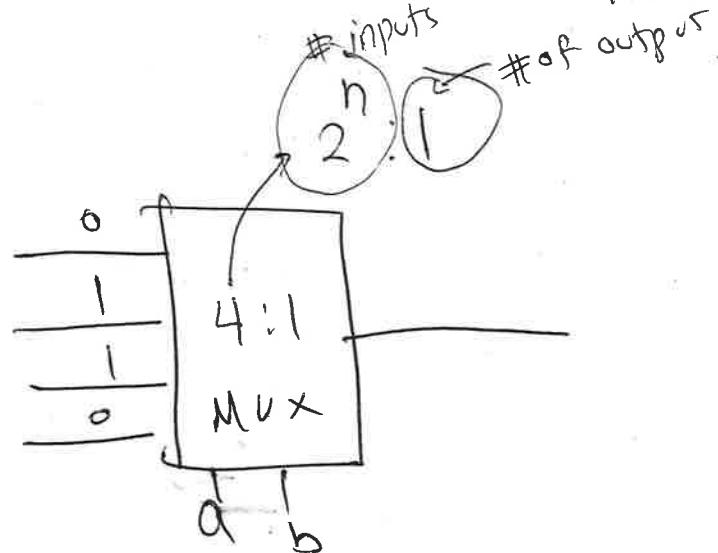
Carry out a range of Calculations

on its input.

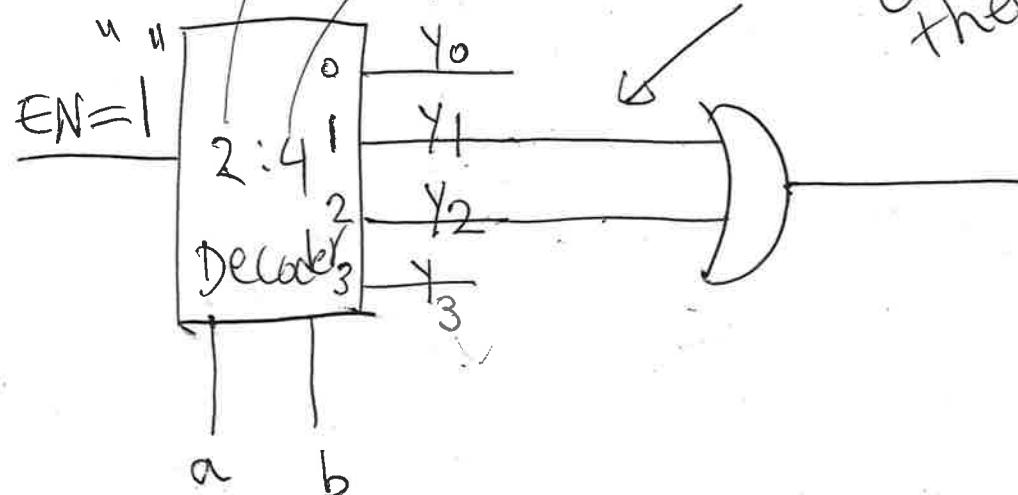
Example: Use 2:4 decoder to build XOR function.

	a	b	$Y = \text{XOR}^u$
0	0	0	0
1	0	1	1 $\leftarrow \bar{ab}$
2	1	0	1 $\leftarrow ab$
3	1	1	0

$$Y = \bar{a}\bar{b} + a\bar{b} + \bar{a}b$$



of select lines
2
select line(s)



wire ones them and input
to an
XOR^u gate.

Oct 1, 2020

Lecture 13 : Programmable logic devices and Tri-state Buffer

programmable logic devices (PLD)

↳ Integrated circuits that contain "AND", "OR" gates

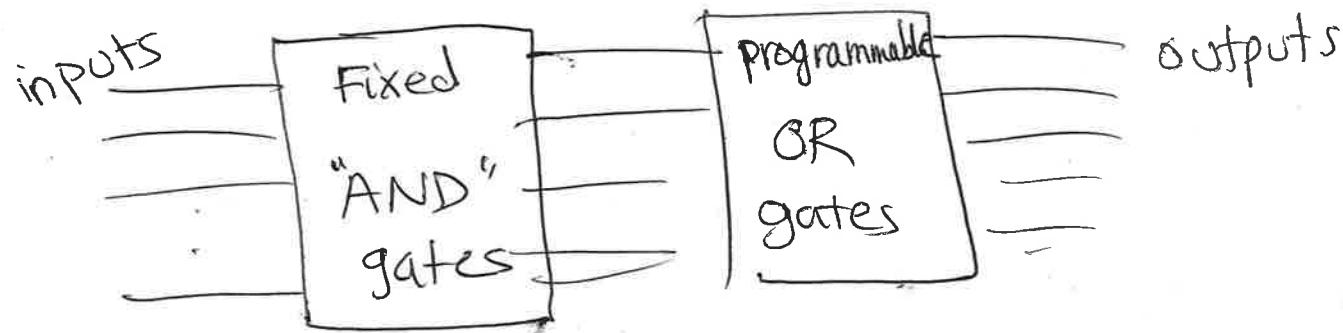
Programming : entering info into these devices
is called Programming

X : Indicate Programming

Types of PLD's

- 1 Programmable read only memory (PROM)
- 2 Programmable array logic (PAL)
- 3 Programmable logic array (PLA)

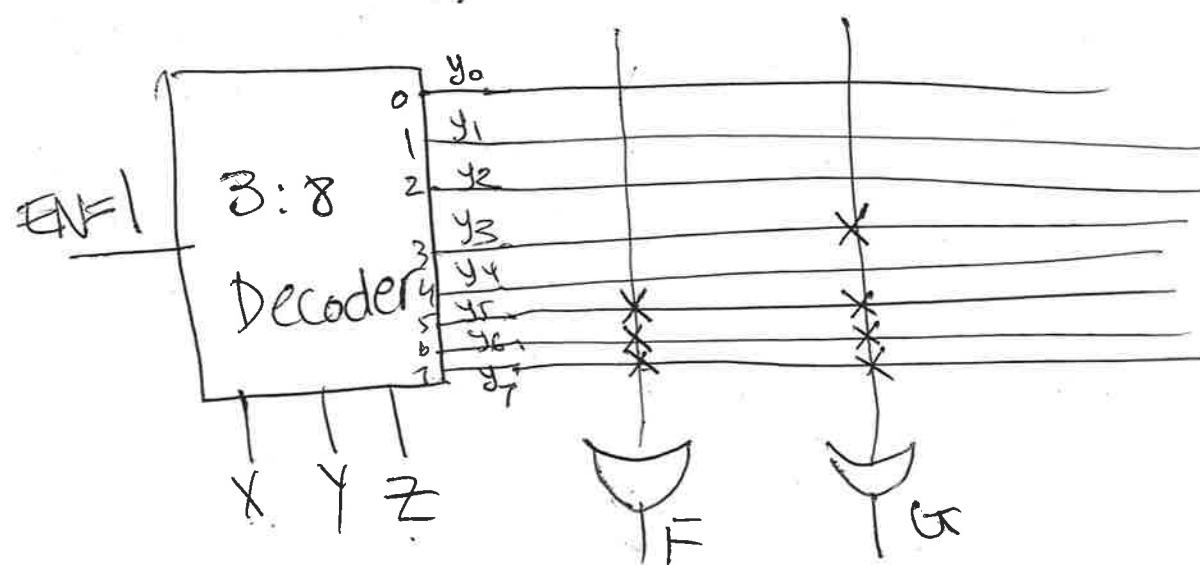
PROM :



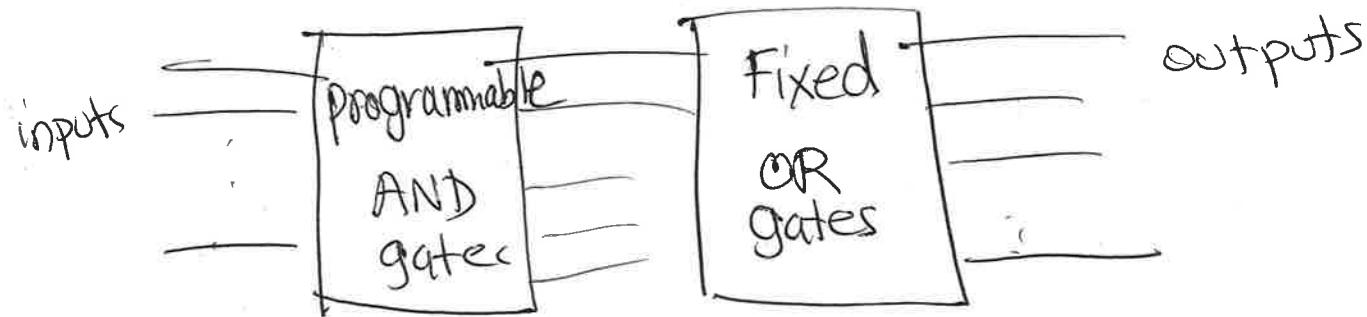
Example : Implement the following outputs using PROM

$$F(x,y,z) = \sum m(5,6,7)$$

$$G(x,y,z) = \sum m(3,5,6,7)$$



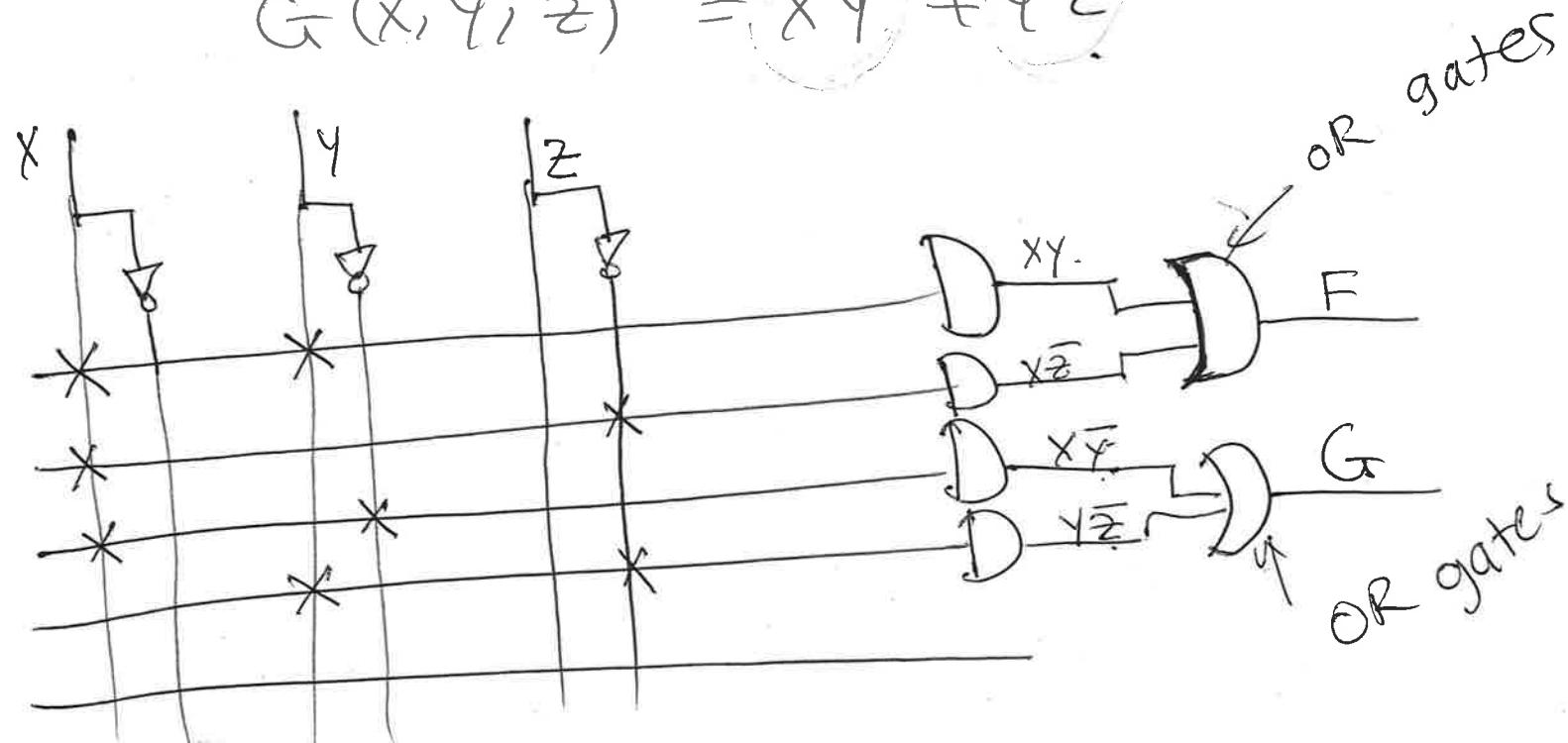
2 PAL



Example: Build the following functions using PAL.

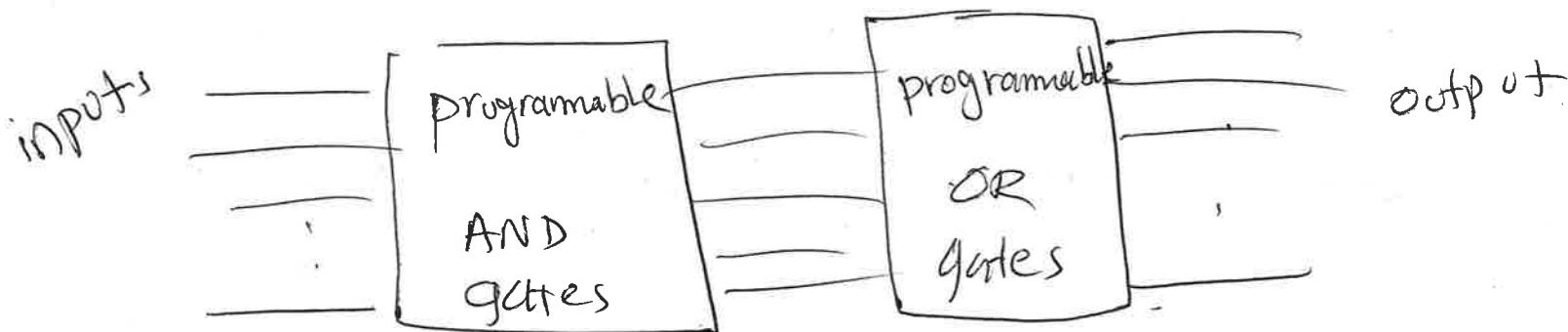
$$F(X, Y, Z) = \overline{XY} + X\overline{Z}$$

$$G(X, Y, Z) = \overline{X}\overline{Y} + Y\overline{Z}$$



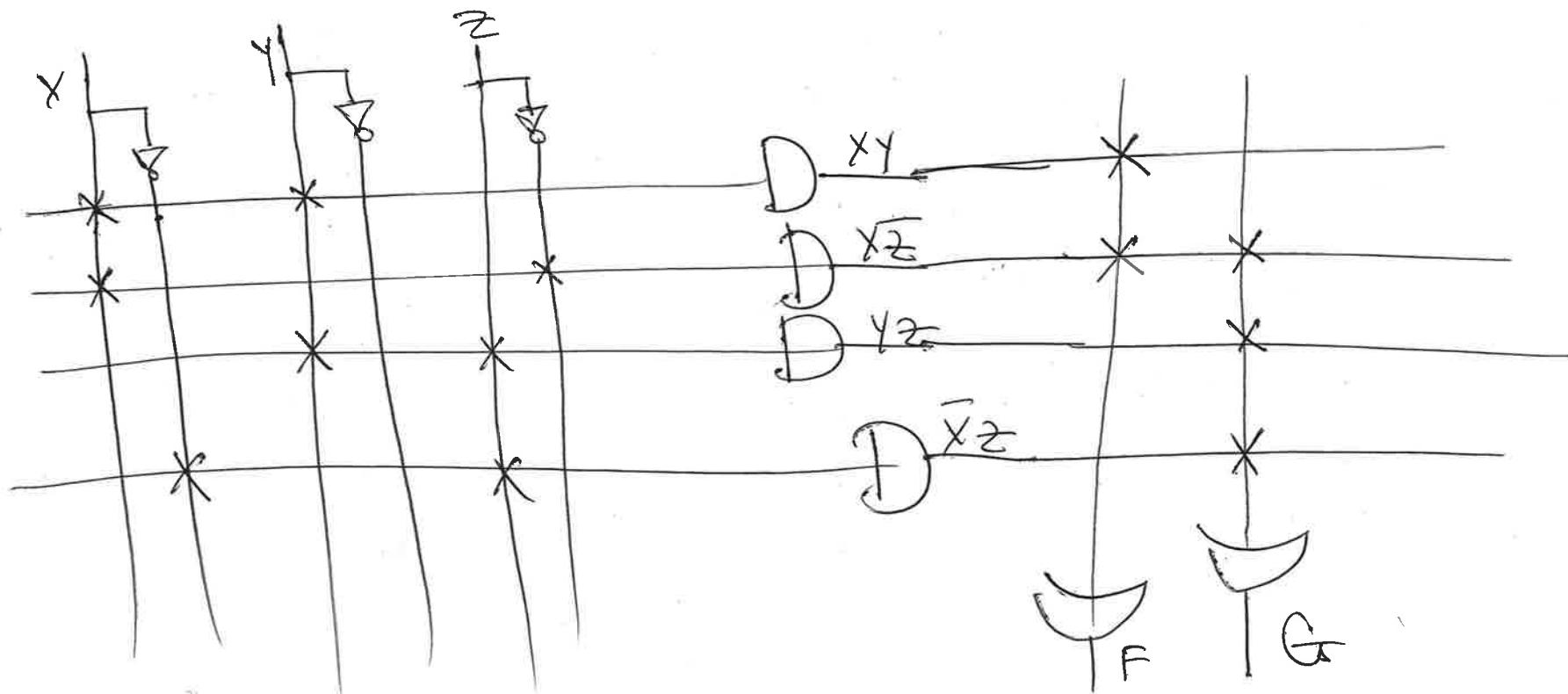
3

PLA

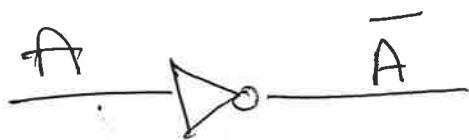


Example : $F(x,y,z) = xy + x\bar{z}$

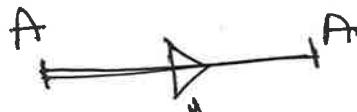
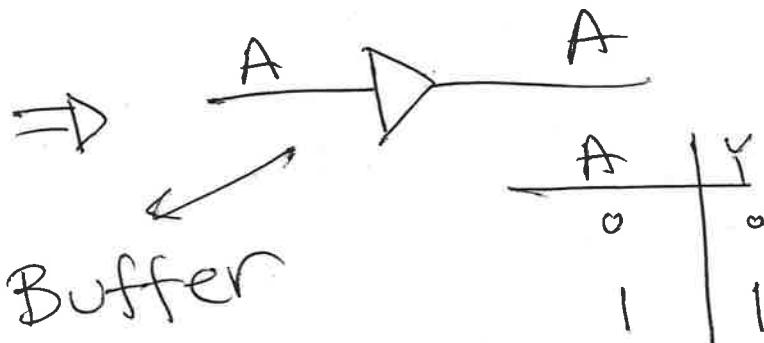
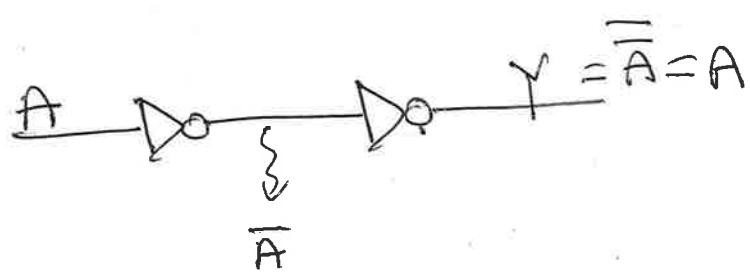
$$G(x,y,z) = \bar{x}\bar{z} + yz + \bar{x}z$$



Tri - State Buffer:

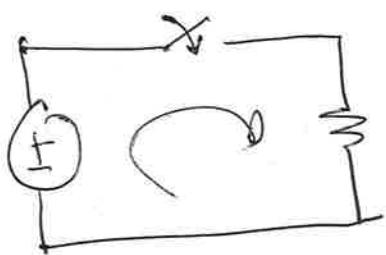
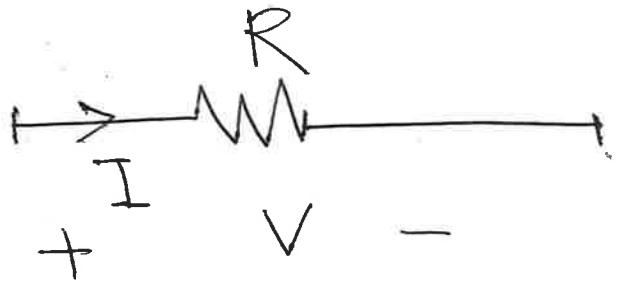


A	Y = \text{Output}	
	0	1
1	0	



digital amplification.





$$V = RI \Rightarrow R = \frac{V}{I}$$

switch open $\Rightarrow I = 0$

$$R = \frac{V}{0} = +\infty$$

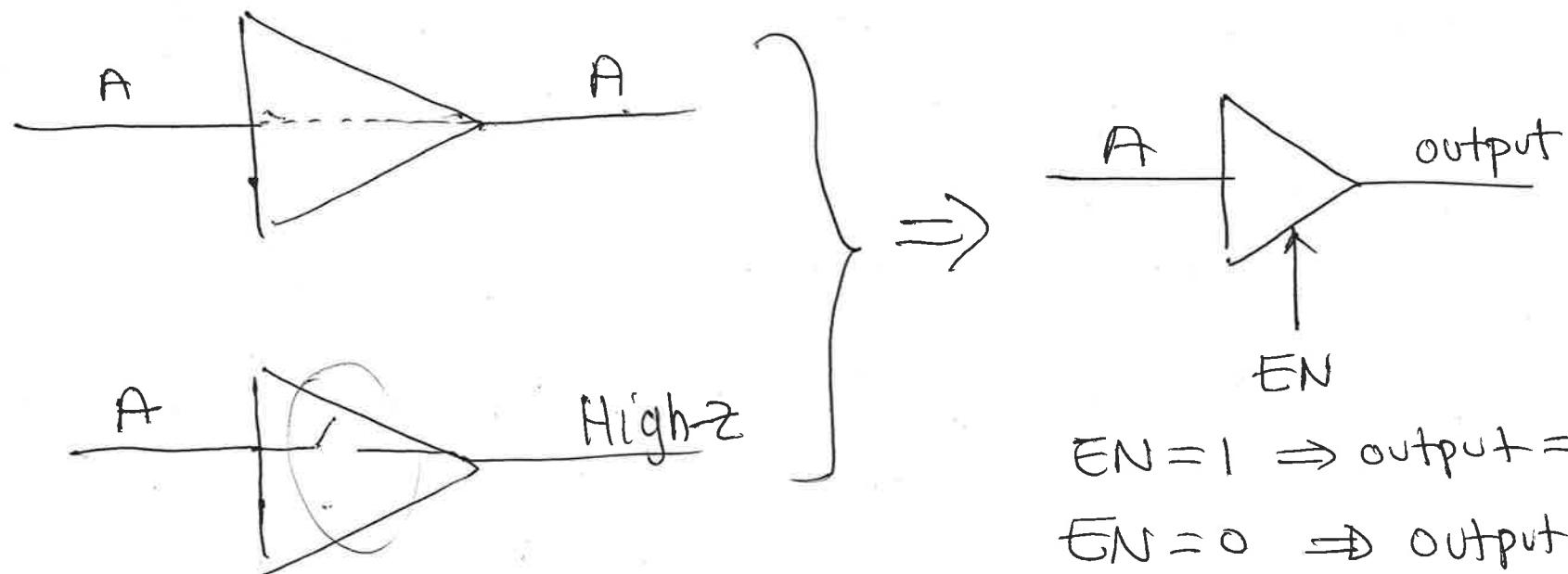
O.C. $\Rightarrow R = \infty$

high $R \Omega'$

high - Z
impedance

Tri-state Buffer :

→ Can be thought of as an input controlled switch w/ an output that can electronically be turned "ON" or "off".



$$\begin{aligned} EN = 1 &\Rightarrow \text{output} = A \\ EN = 0 &\Rightarrow \text{output} = \text{High-Z} \end{aligned}$$

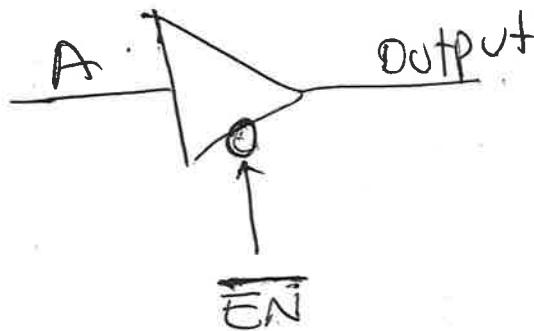
EN	A	output
0	0	High-Z
0	1	High-Z
1	0	0
1	1	1

"Active-high
Tri-state Buffer"

Active low Tri-state Buffer

$EN = 0 \Rightarrow \text{output} = A$

$EN = 1 \Rightarrow \text{output} = \text{High-Z}$

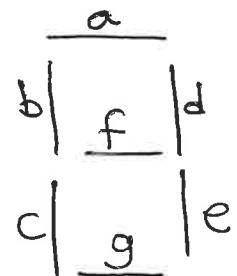


<u>EN</u>	<u>A</u>	<u>output</u>
0	0	0
0	1	1
1	0	High-Z
1	1	High-Z

Short Quiz 2

We'd like to design a display that can show the required output:

The display looks like



Input XYZ	0	1	2	3	4	5	6	7
display Symbol	---		-	--	-			-

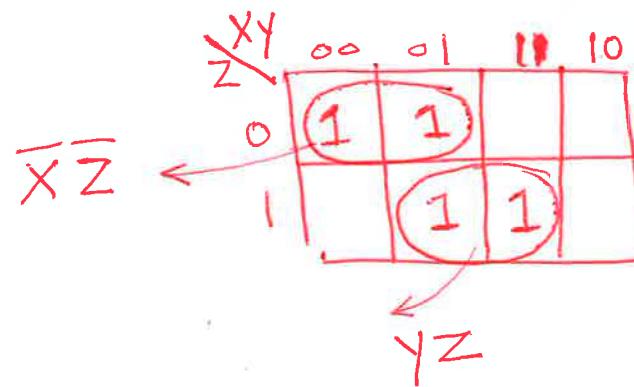
- (A) What is minimum sum of product to display segment "a"?
- (B) What is minimum sum of Product to display segment "f"?

Quiz2:

Solution

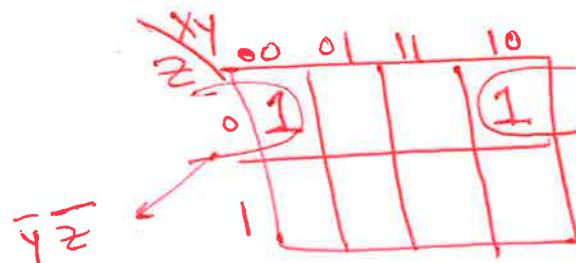
Indecies that contain segment "a".

$$(A) \quad f(x_1, y_1, z) = \sum m(0, 2, 3, 7)$$



$$f(x_1, y_1, z) = \bar{x}\bar{z} + yz$$

$$(B) \quad g(x_1, y_1, z) = \sum m(0, 4)$$



$$g(x_1, y_1, z) = \bar{y}\bar{z}$$

Lecture 14 : Oct. 6, 2020

EEE/CSE 120: Latches & flipflops

- HW 4 is uploaded
- Lab 2 is on canvas
- office hours T/TH 9:30 - 10:15 AM
- Get ready for mid-term
 - | • lockdown browser
 - | • Review your notes

So far; input \Rightarrow output

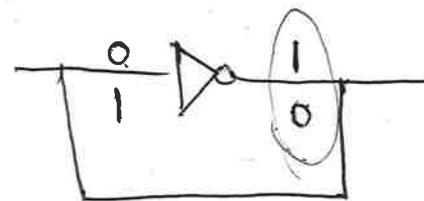
\hookrightarrow Store the logic value!

— ability to program storage module

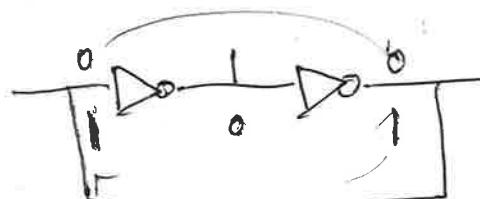
— ability to read the stored value.

Idea: Feed the output back to the input
"feed back"

1 First attempt:

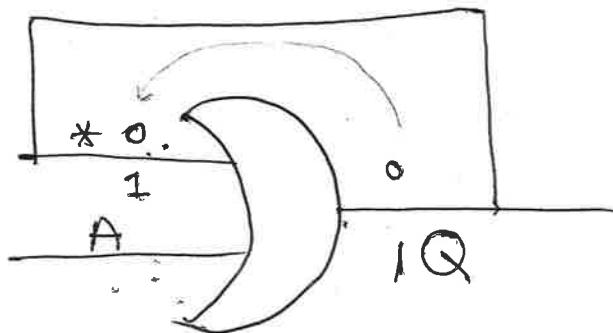


oscillates between "0" & "1"
 \Rightarrow Unstable



\Rightarrow Stable

2 Second attempt

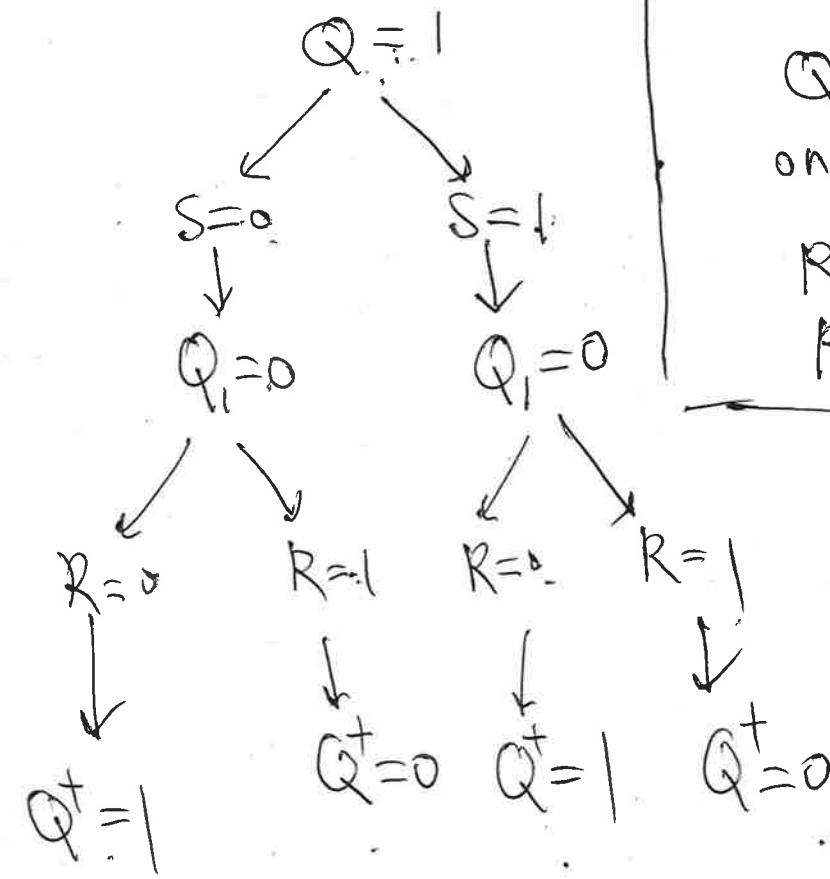
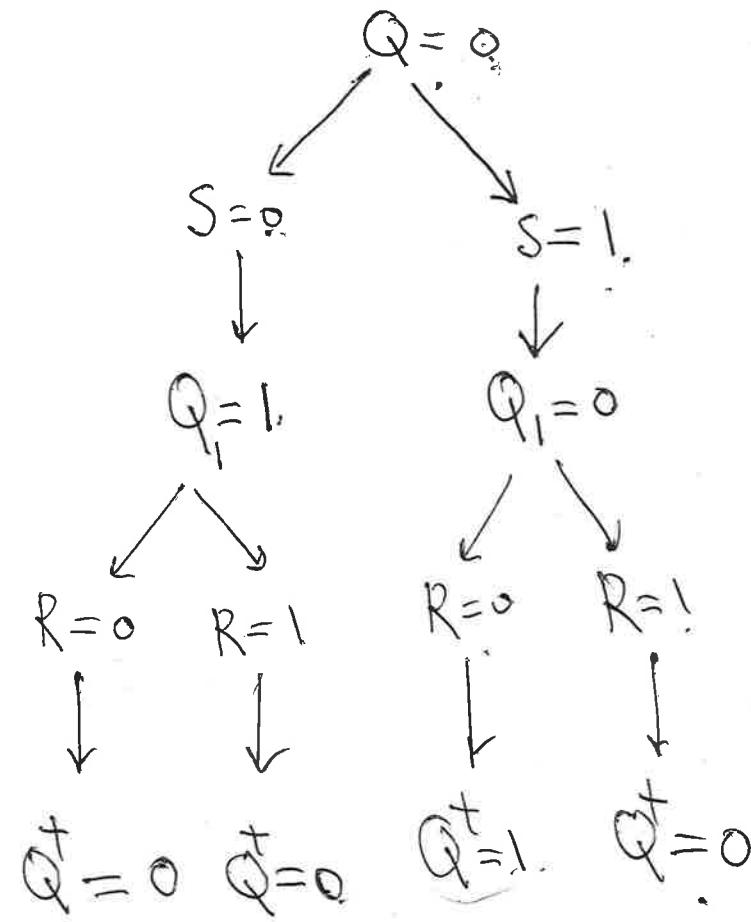
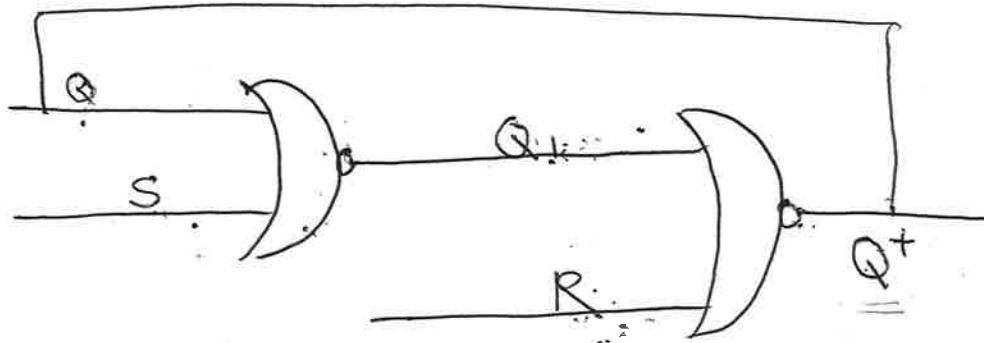


$$\begin{aligned} \underline{A = 0} : * = 0 &\Rightarrow Q = 0 \} \\ * = 1 &\Rightarrow Q = 1 \\ \underline{\text{bi-stable}} \end{aligned}$$

$$\begin{aligned} \underline{A = 1} : * = 0 &\Rightarrow Q = 1 \} \\ * = 1 &\Rightarrow Q = 1 \end{aligned}$$

output switches to 1
 \Rightarrow cannot program "0".

3 Third Attempt



Set $S=1$

Q^+ only depends on R

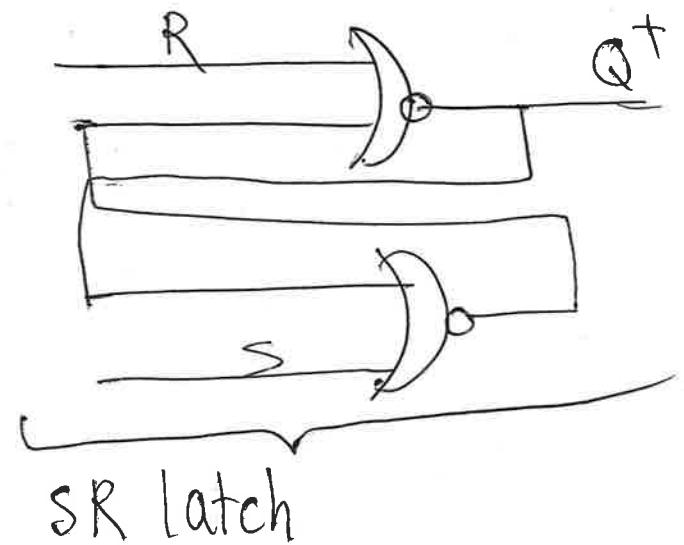
$R=0 \Rightarrow Q^+=1$

$R=1 \Rightarrow Q^+=0$

S	R	Q	$Q^+ \leftarrow Q_{next}$
0	0	0	0 } bi-stability \Rightarrow Storage part
0	0	1	1 }
0	1	0	0 } \leftarrow program "0"
0	1	1	0 }
1	0	0	1 } \leftarrow program "1"
1	0	1	1 }
1	1	0	0 } Exclude $R=1, S=1$
1	1	1	0 }

State transition table

S	R	Q^+
0	0	Hold data (Q)
0	1	0
1	0	1
1	1	X



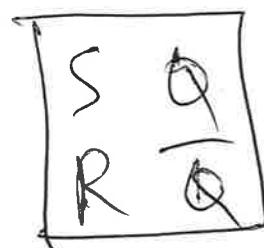


SR-latch

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

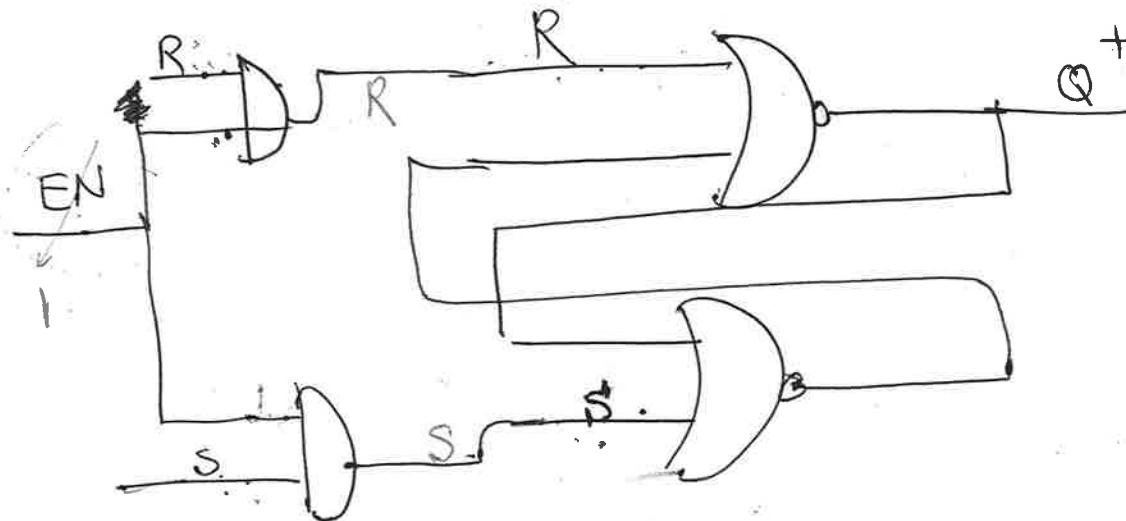
Observations :

- If $S = R = 0 \Rightarrow$ Stable states
- If switch S to 1 (temporarily) $\Rightarrow Q^+ \rightarrow 1$ } " set
If we switch S back to zero $\Rightarrow Q^+$ stays at 1
- If switch R to 1 (temporarily) $\Rightarrow Q^+ = 0$ } Reset
Switch back to 0 $\Rightarrow Q^+$ stays at 0



SR-latch

SR latches w/ enable :



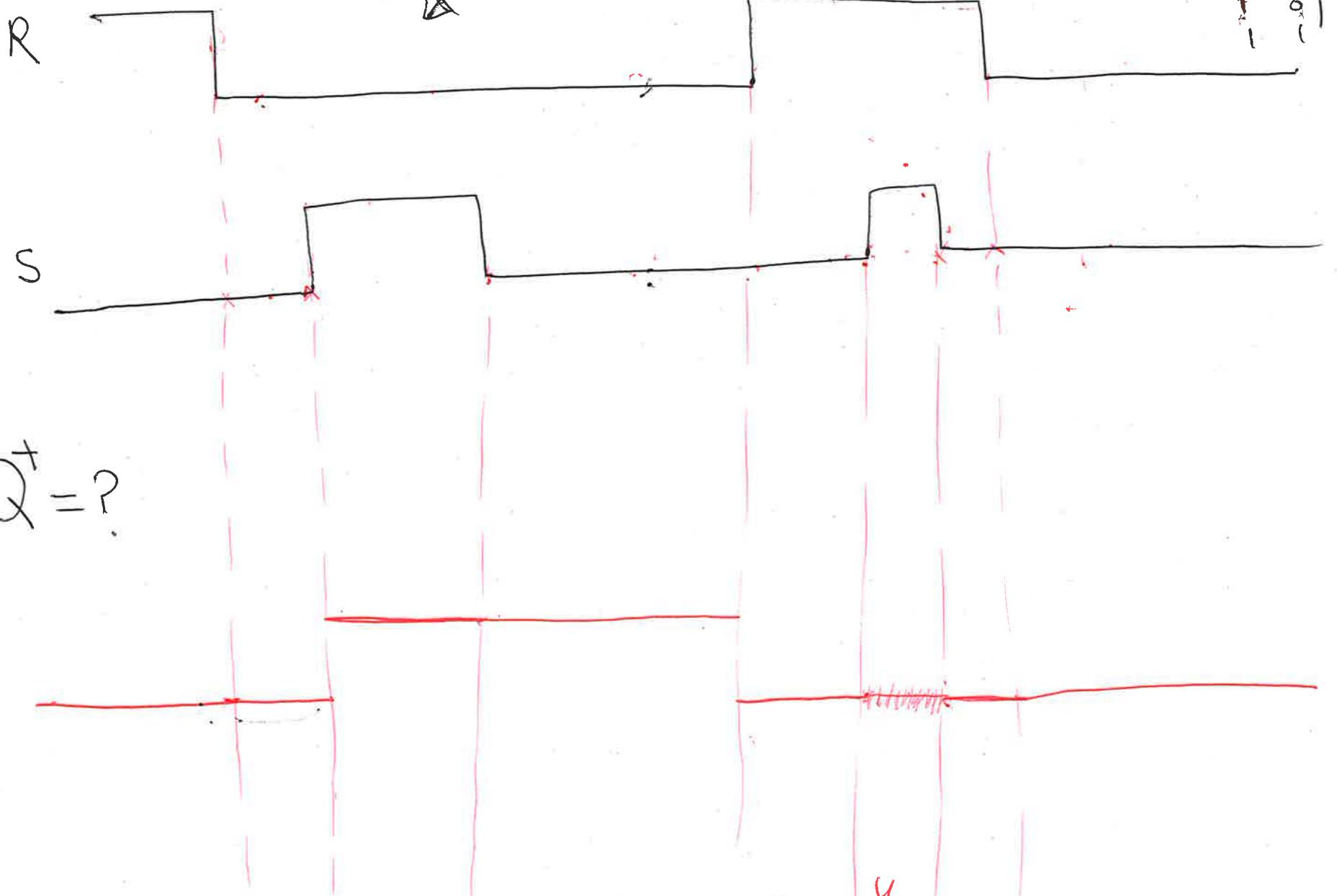
$\text{EN} = 1 \rightarrow \text{SR-latch}$

$\text{EN} = 0$: storage mode

<u>EN</u>	<u>S</u>	<u>R</u>	<u>Q^+</u>
0	X	X	storage mode
1	0	0	storage
1	0	1	0
1	1	0	1
1	1	1	X forbidden

Example:

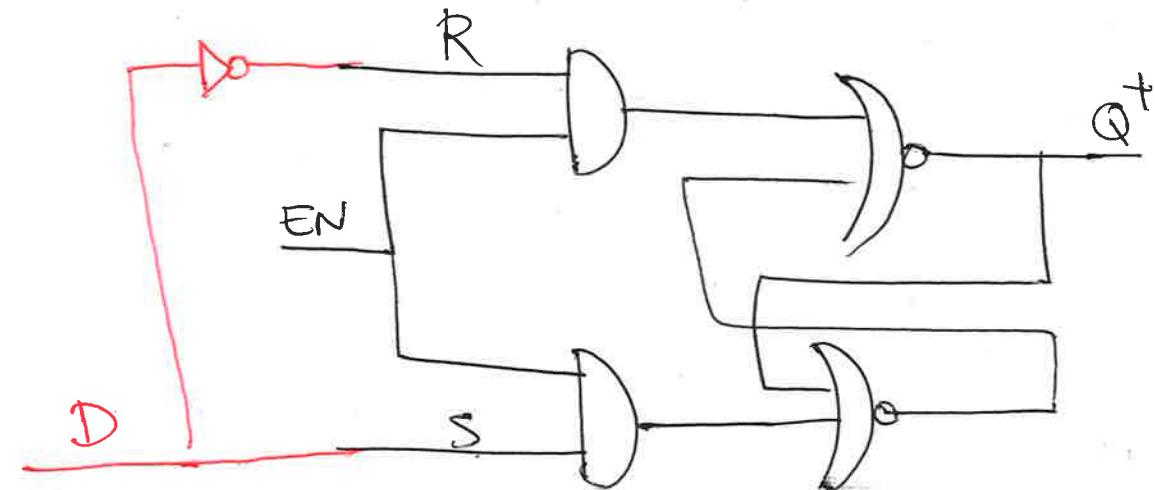
Timing diagrams



$Q^+ = ?$

!! (level) sensitive !!

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



D-latch

EN	D	Q^+
0	X	storage
1	0	0
1	1	1

behavioral eq:

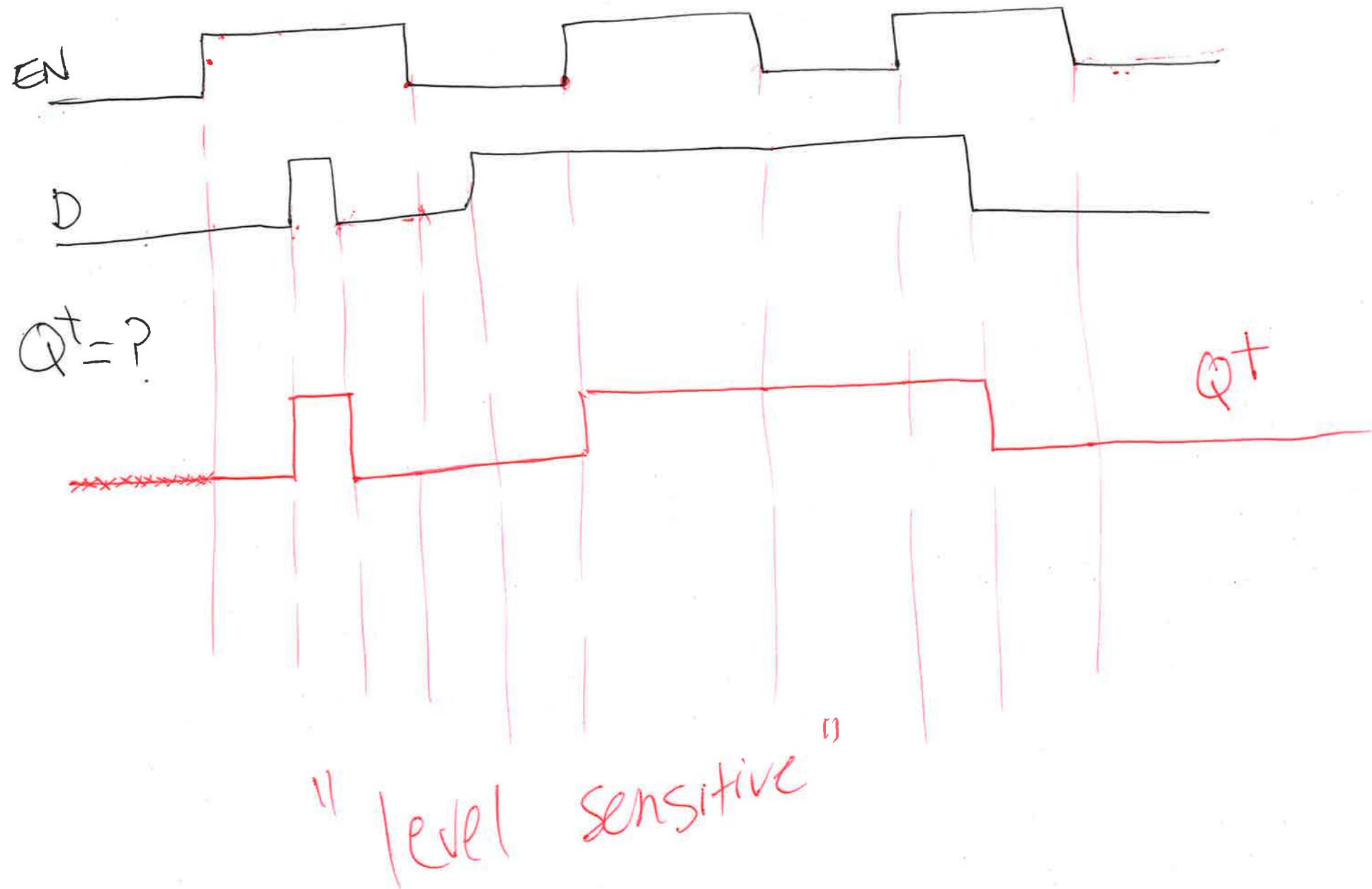
EN	S	R	Q^+
0	X	X	Storage ' Q '
1	0	0	$\neg Q$ (storage)
1	0	1	0
1	1	0	1
1	1	1	X

state transition table:

EN	D	Q^+
0	X	storage
1	D	D

$$Q^+ = D$$

Example



Lecture 15 : Flip Flops

Oct 8, 2020

- Lab office hours next week :

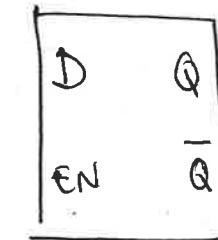
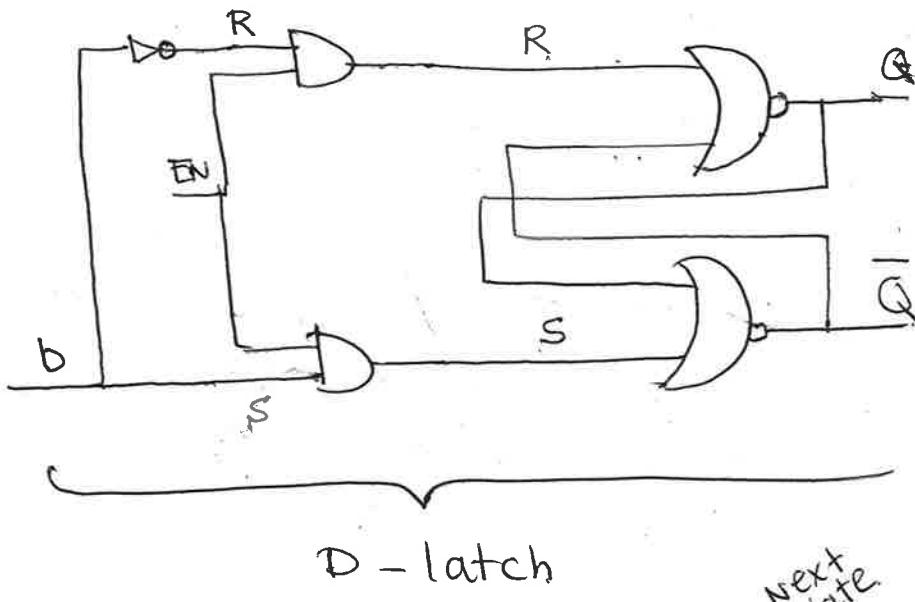
Monday 5-6 pm

Wednesday 12-1:30 pm

Friday 2:30- 4:30 pm

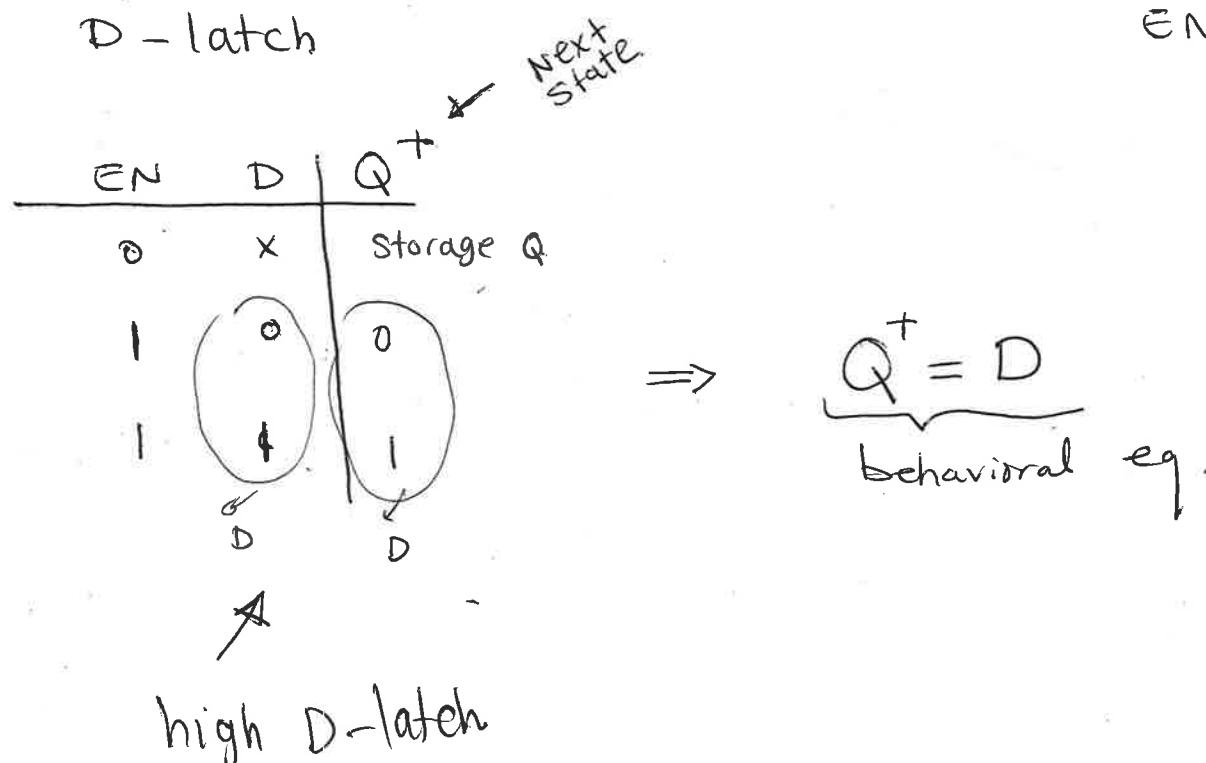
- Make sure you have lockdown browser

Last time :

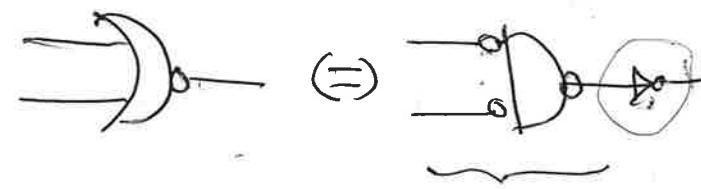
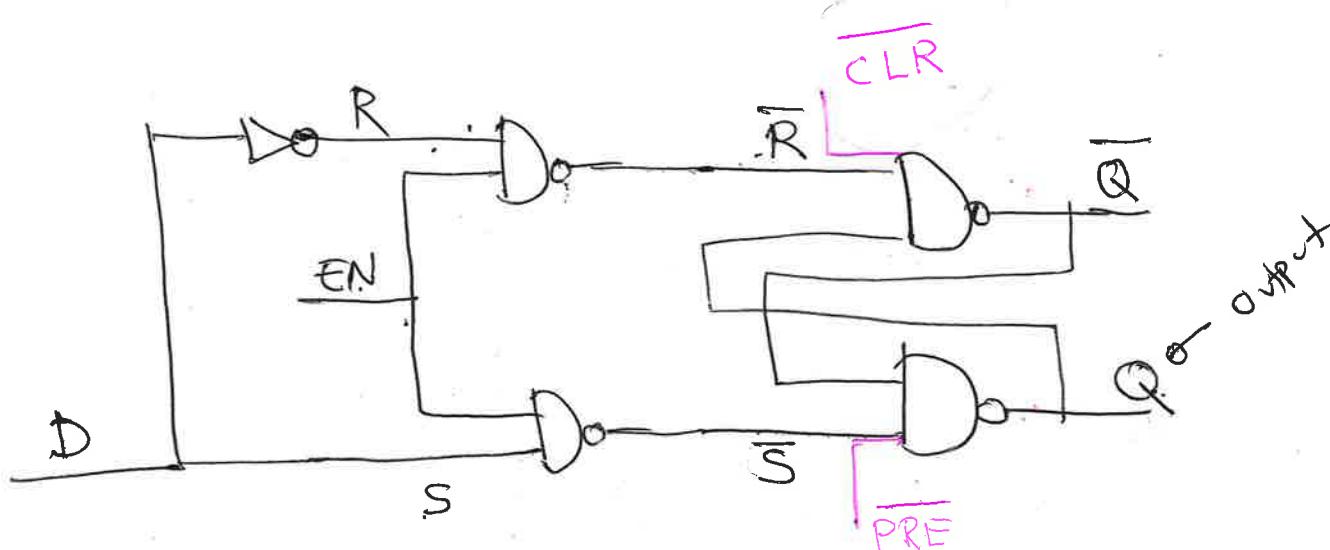


$EN=1 \Rightarrow$ Active high
high D-latch

$EN=0 \Rightarrow$ low D-latch



D-latch using "NAND" gates



$EN = 1 \Rightarrow$ latch working

$EN = 0 \Rightarrow$ storage mode

EN	D	Q^+
0	x	\bar{Q}
1	D	D

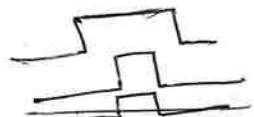
$$\overline{CLR} = 0 \Rightarrow \bar{Q} = 1$$

$$CLR = 1$$

$$\boxed{\bar{Q} = 0}$$

$$\overline{PRE} = 0 \Rightarrow \boxed{Q = 1}$$

$$PRE = 1$$



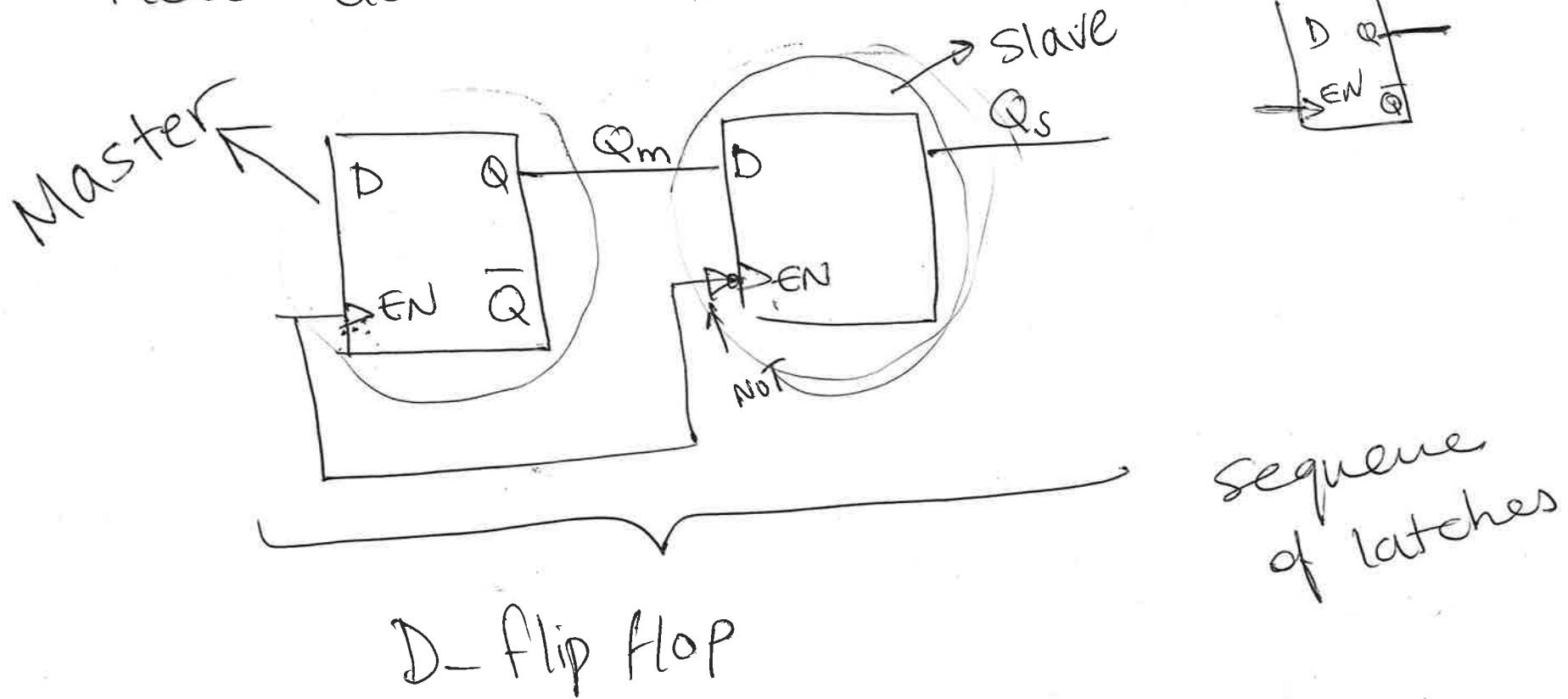
D-latches are level sensitive \Rightarrow output will be programmed
 Transparency.

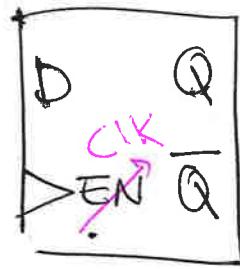
" $EN = 1$ "

Q : Can we design an "EN" that only enables when Signal changes $0 \rightarrow 1$ or $1 \rightarrow 0$

\downarrow
positive edge Negative edge

How do we do this?





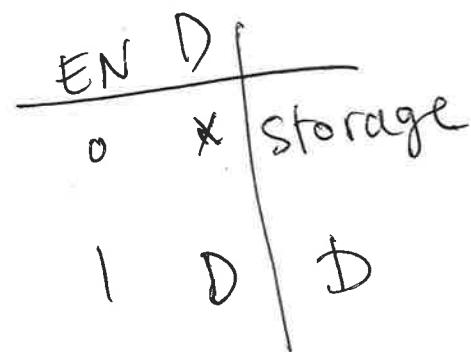
positive edge - D-flip-flop



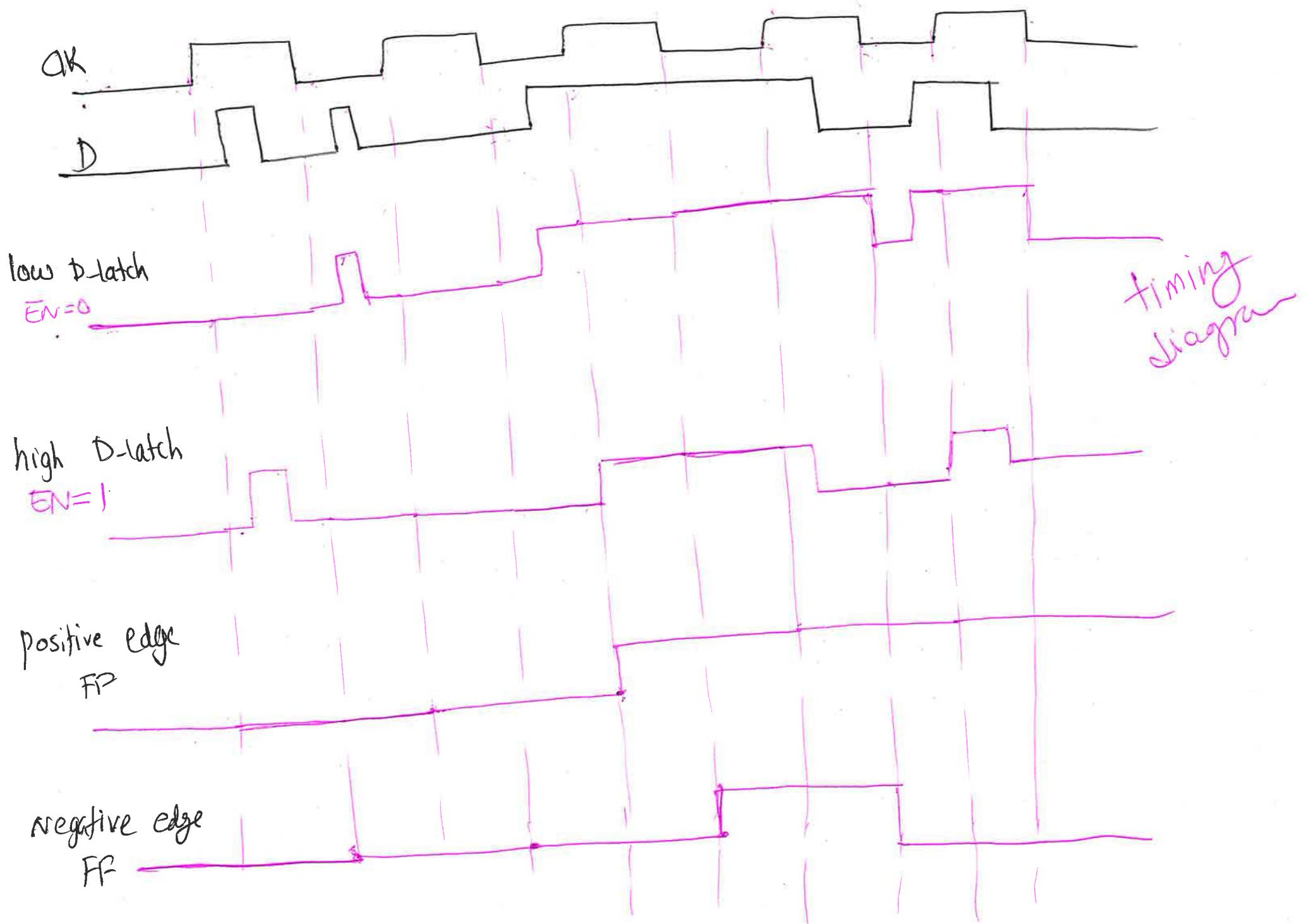
negative edge - D-flip-flop

$$Q^+ = D$$

as long as ~~next~~ rising
edge ($0 \rightarrow 1$)



Example : D- flip flop



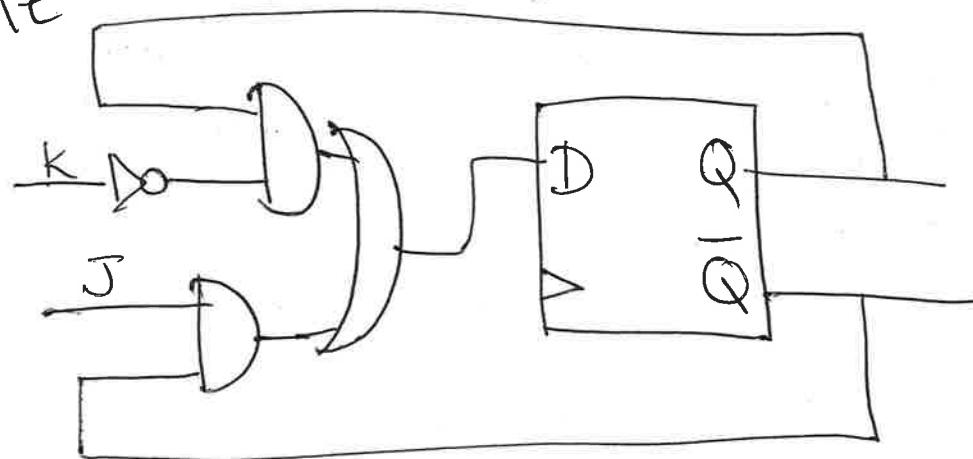
Other types of FFs

① JK flip flop

JK flip flop		Q^+	Next state
J	K		
0	0	0	Storage mode " Q "
0	1	1	
1	0	0	
1	1	1	\bar{Q}

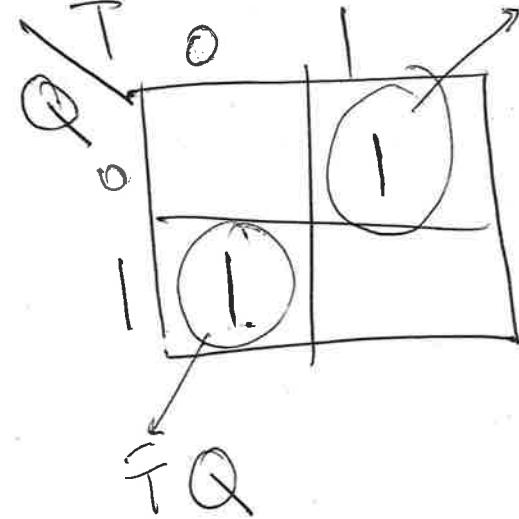
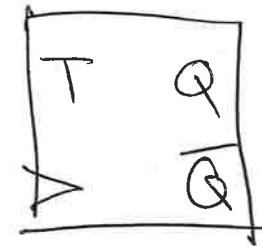
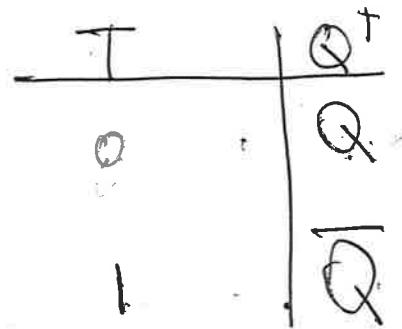
State transition table

$$Q^+ = J Q^- + \pi Q$$



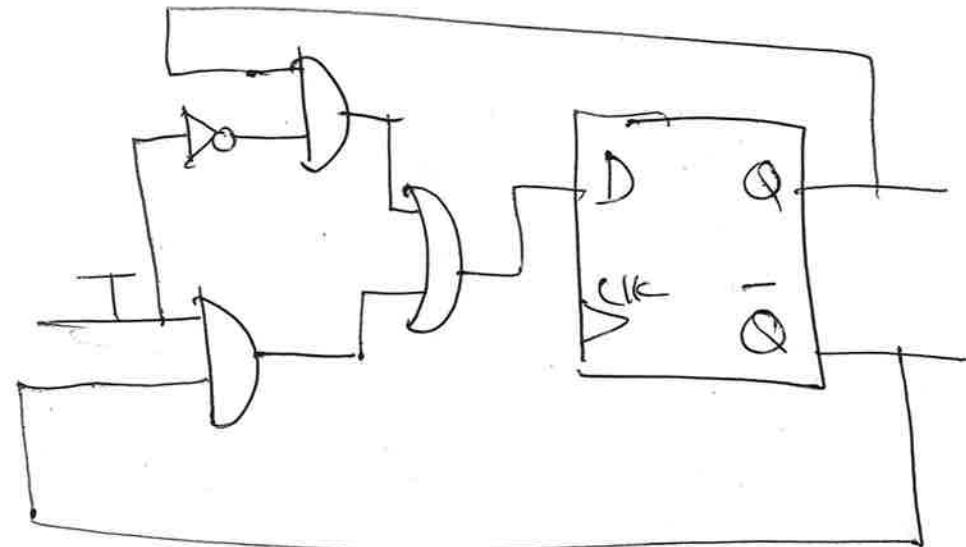
JK flip-flop

② Toggle flip flop



$T\bar{Q}$

$$Q^+ = \bar{T}\bar{Q} + \bar{T}Q$$

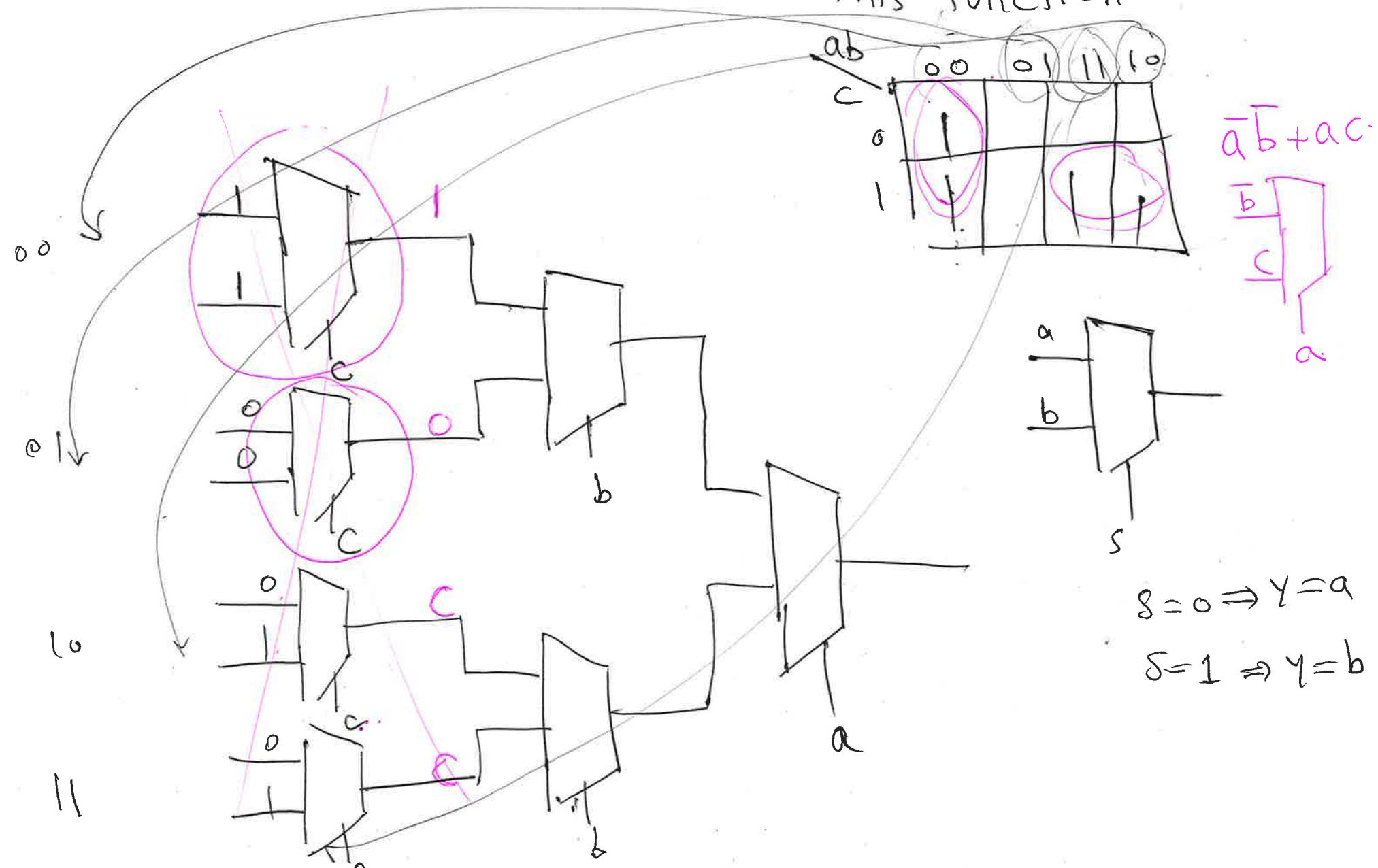


Toggle flip flop .

Example :

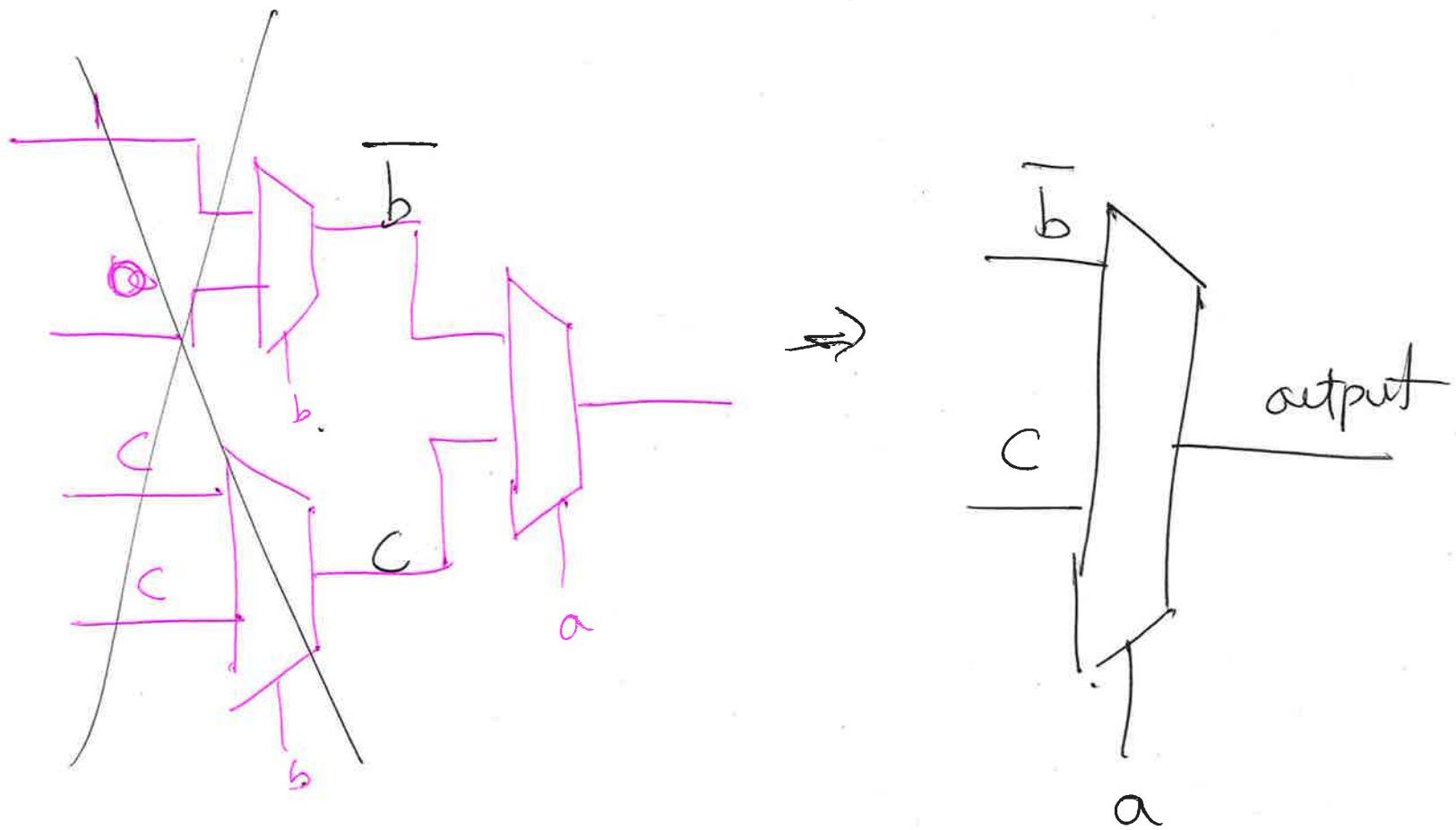
$$f(a, b, c) = \sum m(0, 1, 5, 7)$$

Using MUX to build
this function.



$$s=0 \Rightarrow y=a$$

$$s=1 \Rightarrow y=b$$

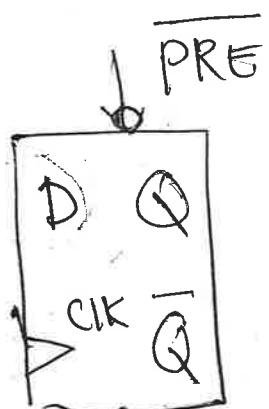


Oct 13, 2020

EEE/CSE 120 : Registers

- Office hours T/TH 9:30 - 10:15 AM
- Lab office hours | wed: 12 - 1:30 pm
| Fri: 2:30 - 4:30 pm
- Lab 3 is uploaded
- Make sure you have lock-down
 - = Mock exam!

FF 8



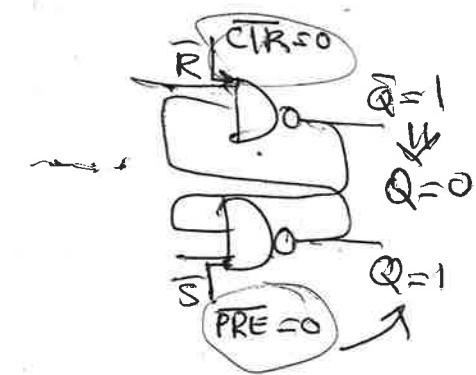
positive edge

FF

0 → 1

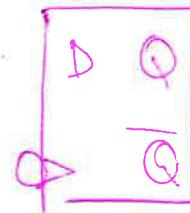
$$\overline{\text{CIR}} = 0 \Rightarrow Q = 0$$

$$\overline{\text{PRE}} = 0 \Rightarrow Q = 1$$



$$Q^+ = D$$

Next State

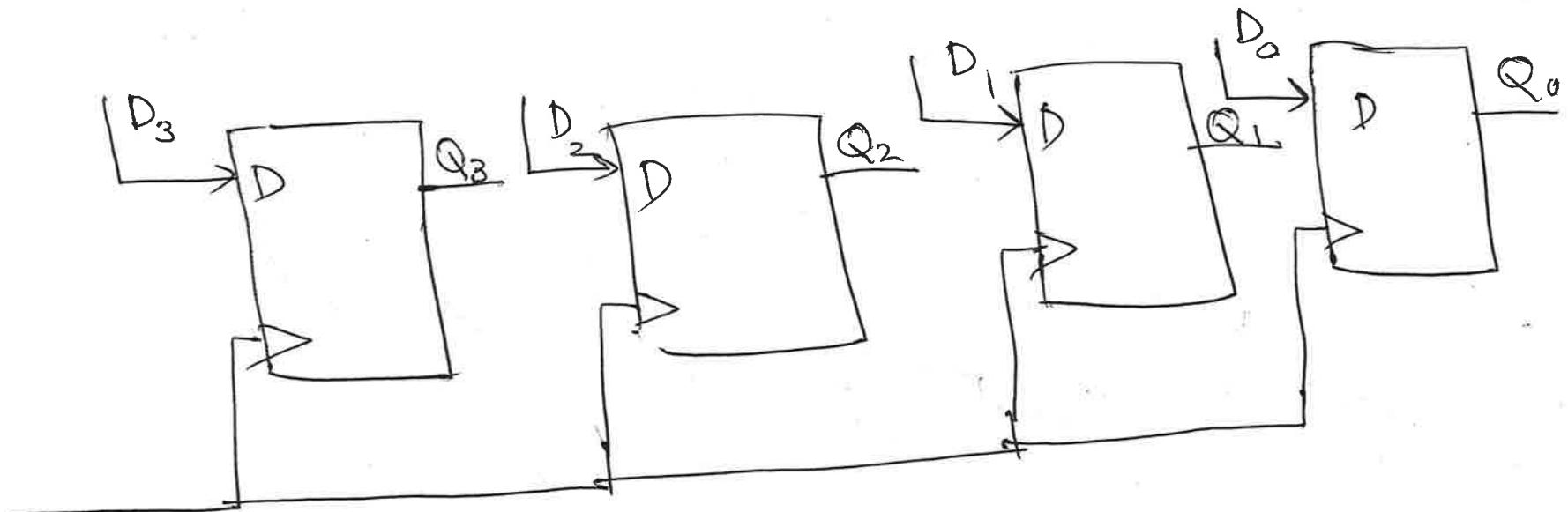


Negative edge

1 → 0

Register : Any combination of flip flops is called a register !

Example : (parallel in - parallel out)



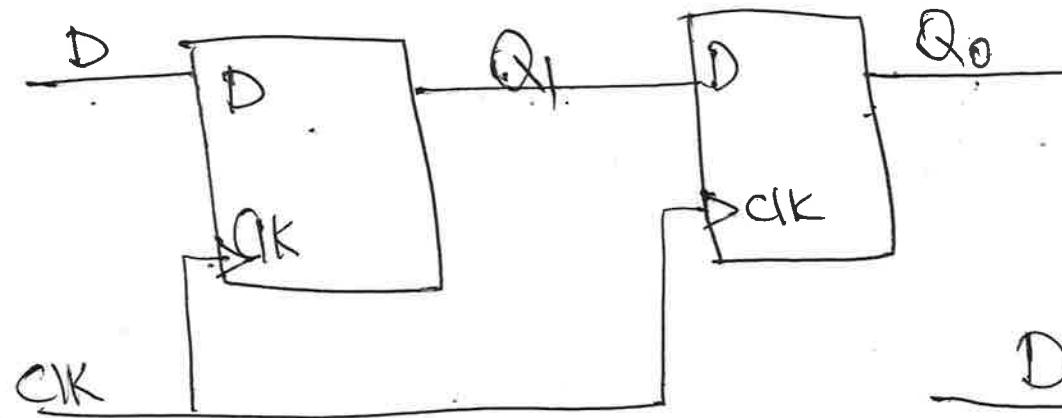
Same clock \Rightarrow Synchronous.

$D_3 D_2 D_1 D_0$ (4 bit)

$Q_3 Q_2 Q_1 Q_0$ (output)

Each clock comes in
4-bit binary number
and loads 4 bit
out.

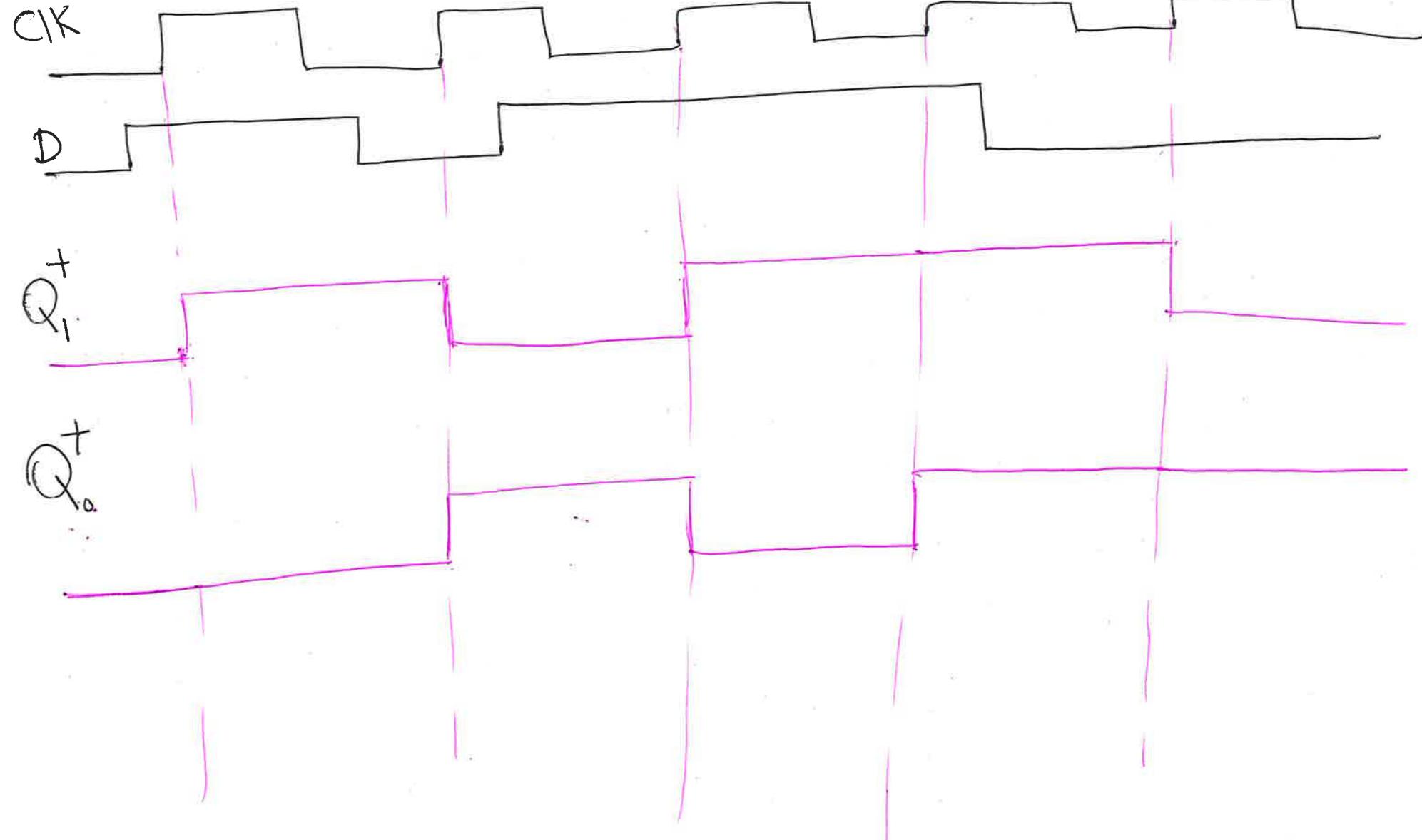
Example : (Shift register)



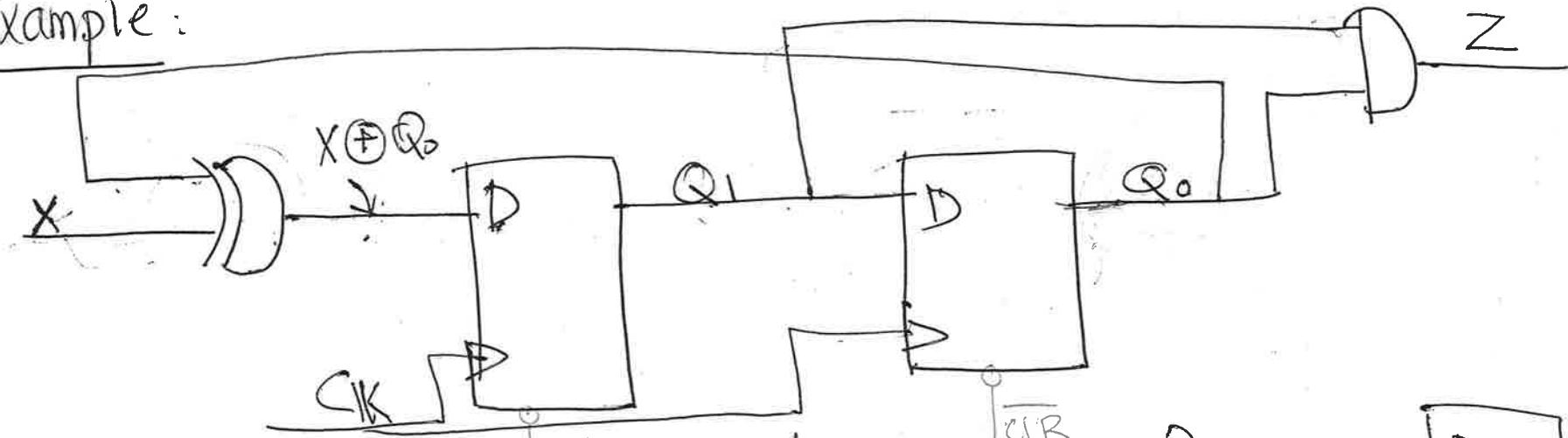
$$\begin{cases} Q_1^+ = D \\ Q_0^+ = Q_1 \end{cases}$$

D	Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

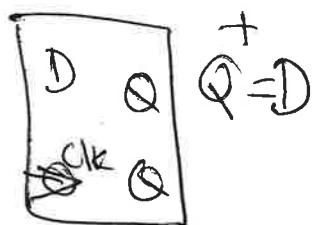
$$Q_i^+ = D \quad | \quad Q_o^+ = Q_F$$



Example :



① what is the behavioral eq. ?



next states

$$\left\{ \begin{array}{l} Q_1^+ = X \oplus Q_0 \\ Q_0^+ = Q_1 \end{array} \right.$$

output

$$Z = Q_1 Q_0$$



$$Q_1^+ = X \oplus Q_0$$

$$Q_0^+ = \overline{Q_1}$$

$$Z = Q_1 Q_0$$

$\text{CIR} = 1 \Rightarrow$
 $\text{CIR} = 0 \Rightarrow$
 $\text{output} = 0$

Example: Register that follows these equations:

Next Step

$$Q_2^+ = Q_1 + Q_2$$

$$Q_1^+ = Q_0$$

outp_X

$$Q_0^+ = X + Q_2$$

$$Y = Q_0 + Q_1$$

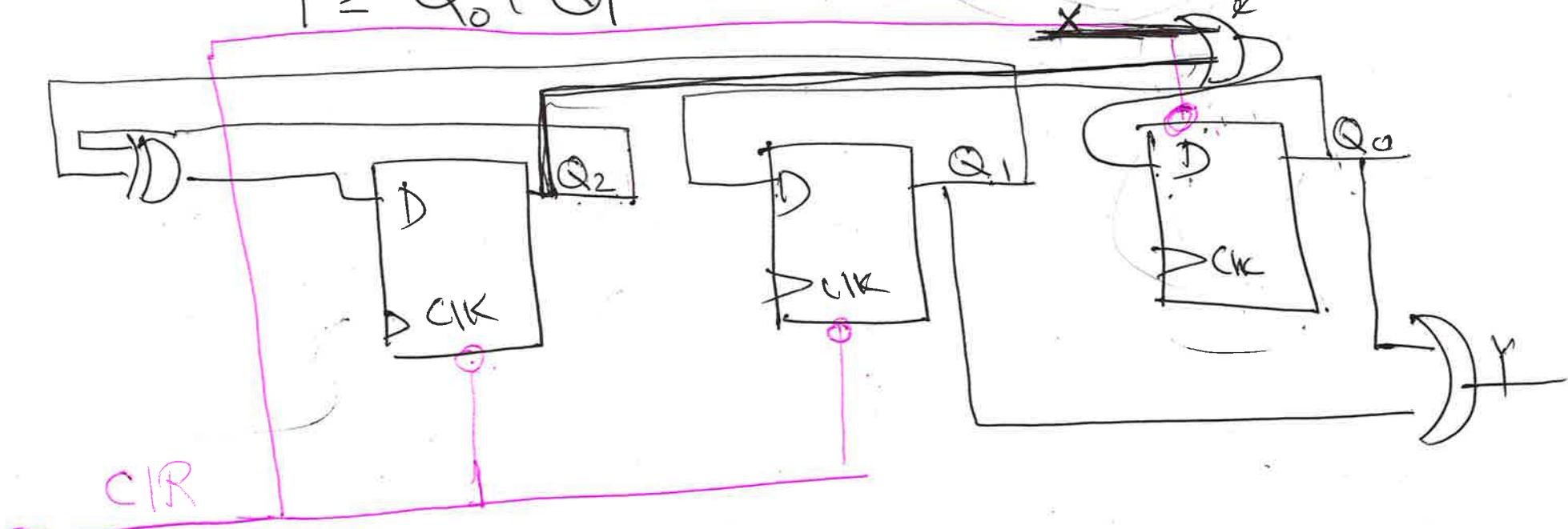
Assuming

$$Q_2 = 0$$

$$Q_1 = 0$$

$$Q_0 = 1$$

OR



CIR = 1

(Synchronous CKs are connected to the same)

EEE/CSE 120 : Review for Midterm

office hours :

{ Monday 3:00 - 4:00 pm }

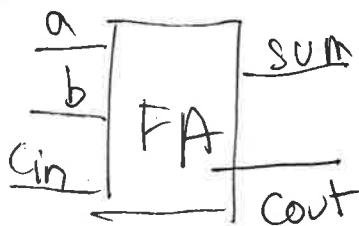
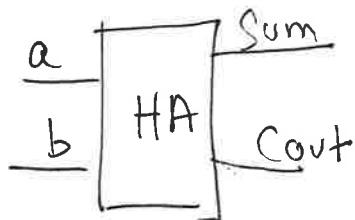
Example I: Add # (Signed)

$$\begin{array}{r}
 \text{cin} \\
 \boxed{1} \quad 0 \quad 1 \quad 0 \quad | \quad \xrightarrow{\text{+13}} \quad (01101) \\
 | \quad 0 \quad 0 \quad 1 \quad | \quad \rightarrow (-13)_{10} \\
 + \quad . \quad 1 \quad 1 \quad 1 \quad 0 \quad (14)_{10} \\
 \hline
 \text{cout} \\
 \boxed{1} \quad (0 \quad 0 \quad 0 \quad 0 \quad 1) \quad \cancel{+1}
 \end{array}$$

$\text{cin} = \text{cout} \Rightarrow \text{NO OF.}$

$$\begin{array}{r}
 \text{cin} \\
 \boxed{0} \quad 0 \quad 0 \quad 1 \quad | \quad \xrightarrow{\text{-14}} \quad (01110) \\
 | \quad 1 \quad 1 \quad 0 \quad | \quad \rightarrow (-14)_{10} \\
 + \quad . \quad 1 \quad 0 \quad 0 \quad 1 \quad \xrightarrow{\text{-7}} \quad (00111) \rightarrow -7 \\
 \hline
 \text{cout} \\
 \boxed{1} \quad (0 \quad 1 \quad 0 \quad 1 \quad 1) \quad 7
 \end{array}$$

$\text{cin} \neq \text{cout} \Rightarrow \text{OF.}$



Example 2 :

$$(213)_{16} \rightarrow (\quad)_{10}^{\text{decimal}}$$

↓ ↓ ↓
 16² 16¹ 16⁰
 3 × 16⁰ + 1 × 16¹ + 2 × 16²

$$(327)_{10} \rightarrow \text{HEX}$$

$$\begin{array}{r} 327 \\ \hline R \end{array} \quad \begin{array}{r} 16 \\ \hline 20 \end{array} \quad \begin{array}{r} 16 \\ \hline 1 \end{array}$$

← 7 ← 4

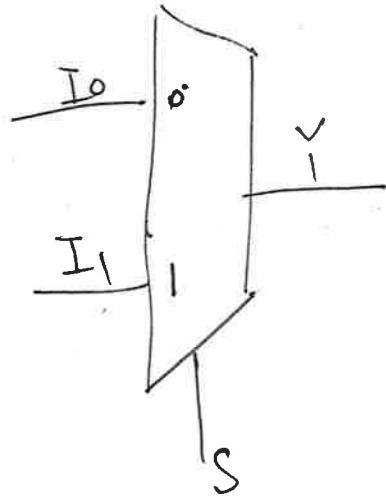
$$(147)_{16}$$

$$(A)_{10} = \sum_{i=1}^n c_i a^i \equiv (\quad)_a$$

$$1 \times 16^2 + 4 \times 16^1 + 7 \times 16^0 \equiv 327$$

$$(F729)_{16} \rightarrow \text{binary}$$

$$\begin{array}{ccccccc} & & & & 1 & 0 & 0 1 \\ & & & & \swarrow & & \downarrow \\ 1 & 1 & 1 & 1 & 0 & 0 1 0 & 1 0 0 1 \end{array}_2$$

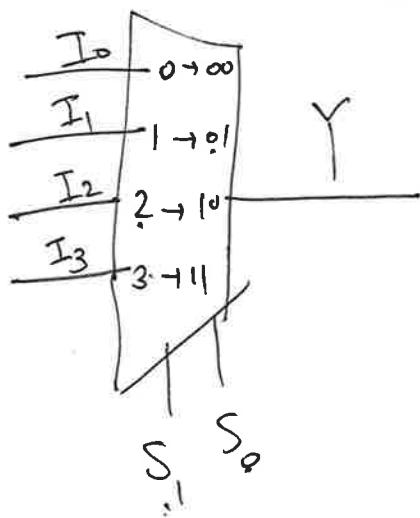


$$S=0 \Rightarrow Y = I_0$$

$$S=1 \Rightarrow Y = I_1$$

$\underbrace{\hspace{10em}}$

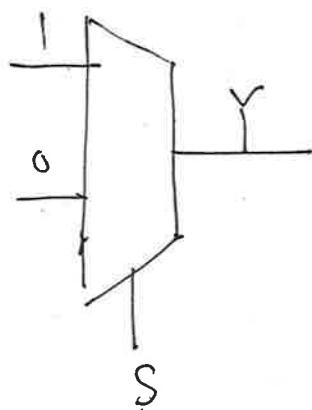
$$Y = \overline{S} I_0 + \underline{S} I_1$$



$$Y = \overline{S_1} \overline{S_0} I_0 + \overline{S_1} S_0 I_1 + S_1 \overline{S_0} I_2 + S_1 S_0 I_3$$

IS a MUX functionally complete?

① Invertor

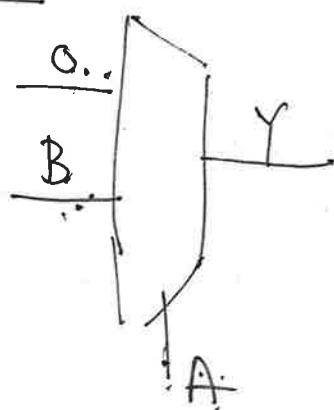


$$S=0 \Rightarrow Y=1$$

$$S=1 \Rightarrow Y=0 \Rightarrow Y=\bar{S}$$



② AND

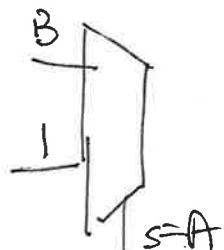


$$A=0 \Rightarrow Y=0 \quad B=AB$$

$$A=1 \Rightarrow Y=B = \underbrace{1 \cdot B}_{A} = AB$$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

③ OR



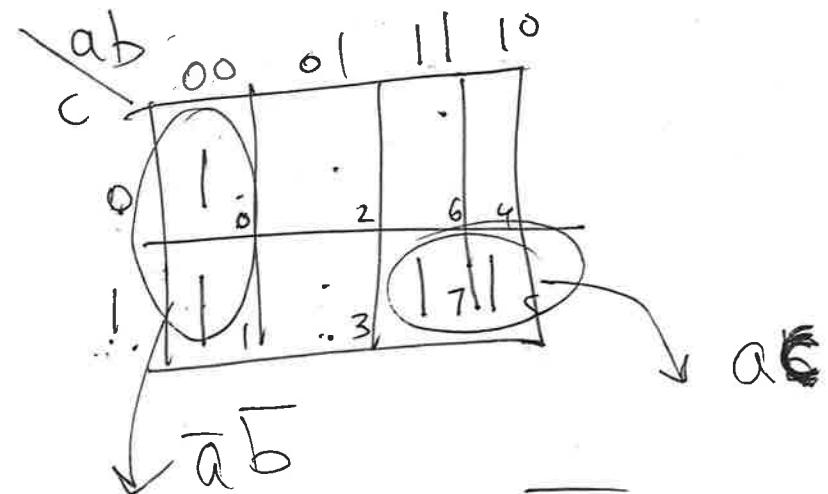
$$A=0 \Rightarrow Y=B$$

$$A=1 \Rightarrow Y=1$$

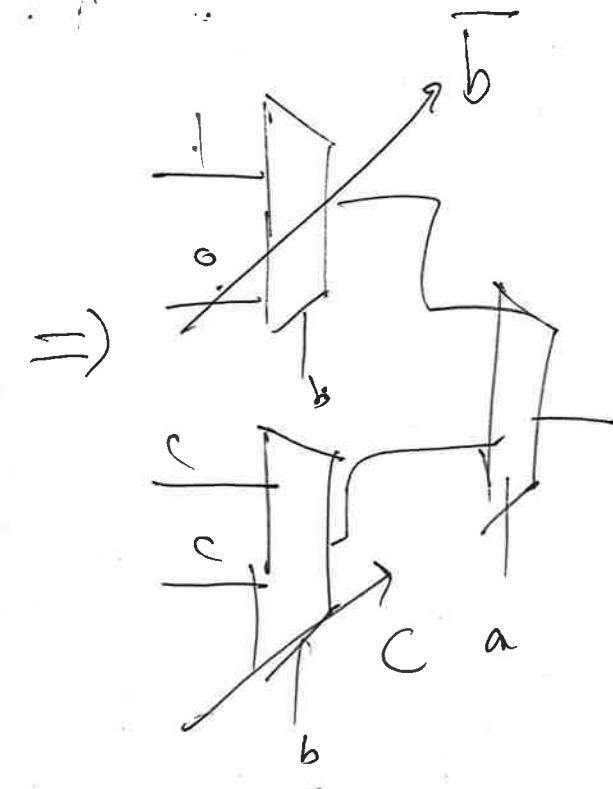
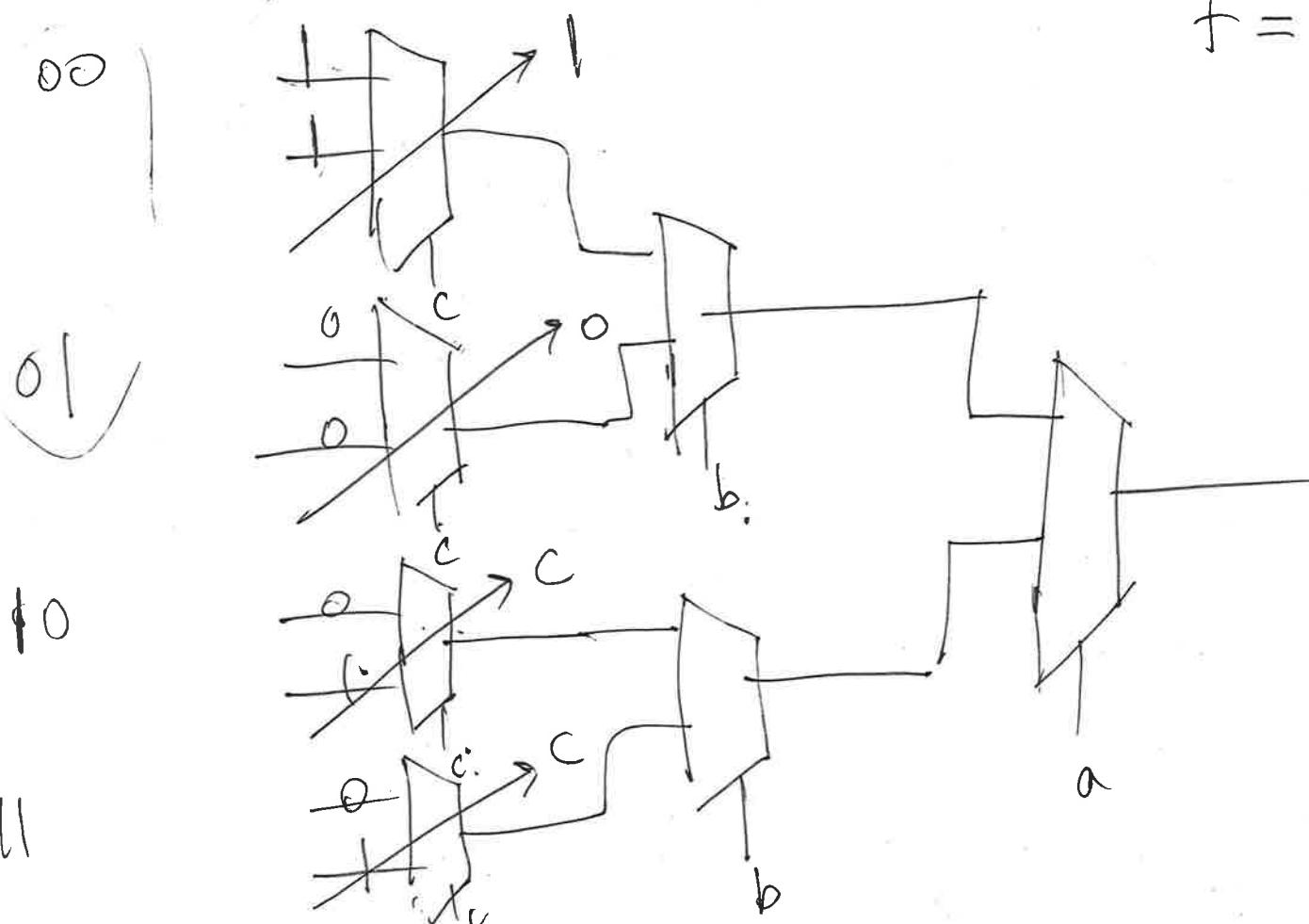
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

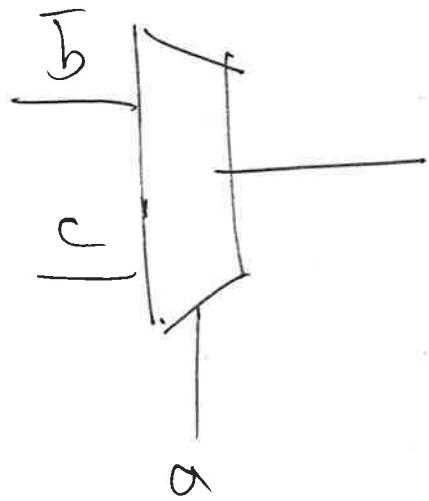
$$f(a, b, c) = \sum m(0, 1, 5, 7)$$

- USE 7 multiplexers
- USE 3 muxes
- USE 1 mux

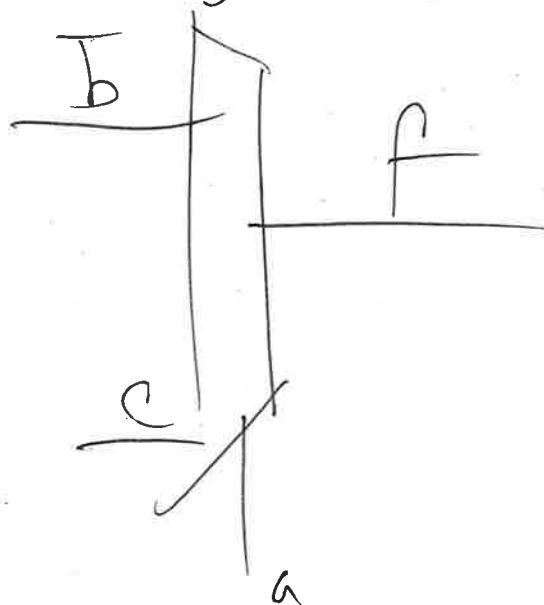


$$f = \underbrace{ac}_{S_0} + \overline{\underbrace{ab}_{S_1}} + \overline{\underbrace{b}_{S_2}}$$





$$f = \frac{ac}{s} I_1 + \frac{\bar{a}b}{\bar{s}} I_0$$



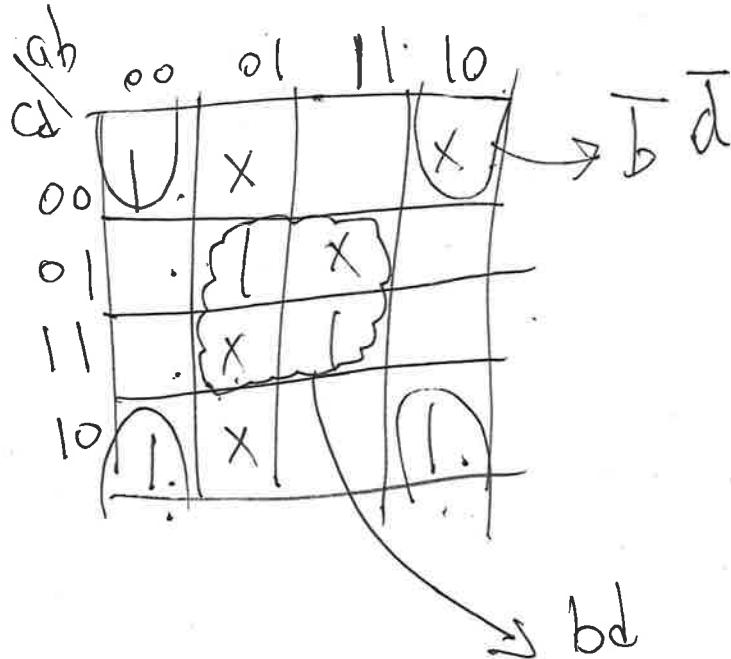
Example 4 :

$$f(a, b, c, d) = \sum m(0, 2, 5, 10, 15) + \sum d(4, 6, 7, 8, 13)$$

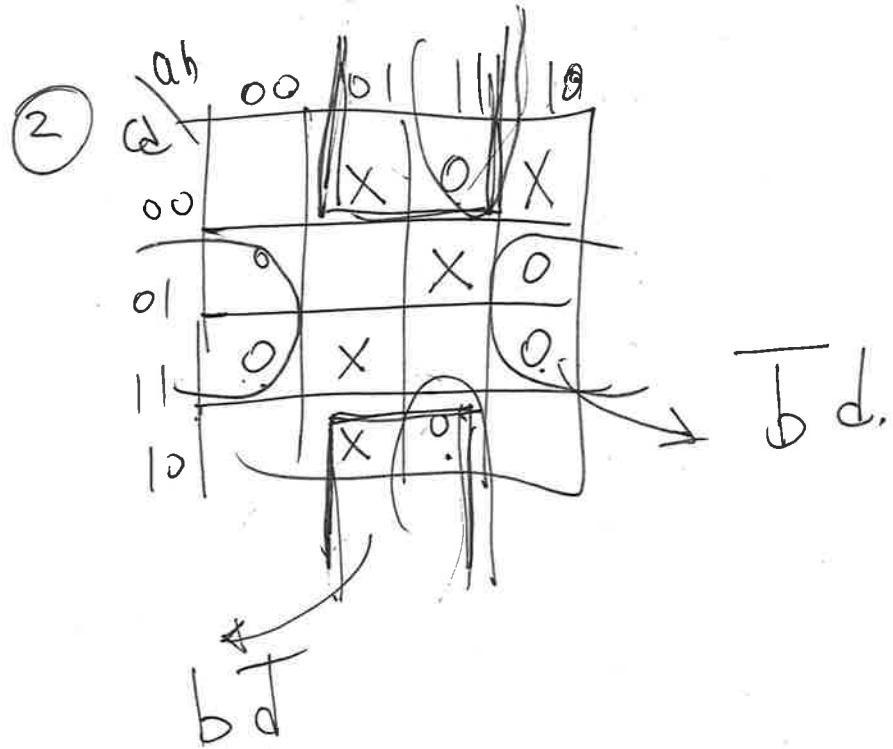
① Find minimum SOP?

② Find minimum POS?

①



$$f(a, b, c, d) = bd + \bar{b}\bar{d}$$



$$\begin{aligned}\bar{f}(a_1, b_1, c_1, d_1) &= \bar{b}d + b\bar{d} \\ f = \bar{\bar{f}} &= \overline{(\bar{b}d + b\bar{d})} \\ &= (\bar{b} + \bar{d})(\bar{b} + d)\end{aligned}$$

OCT 22, 2020

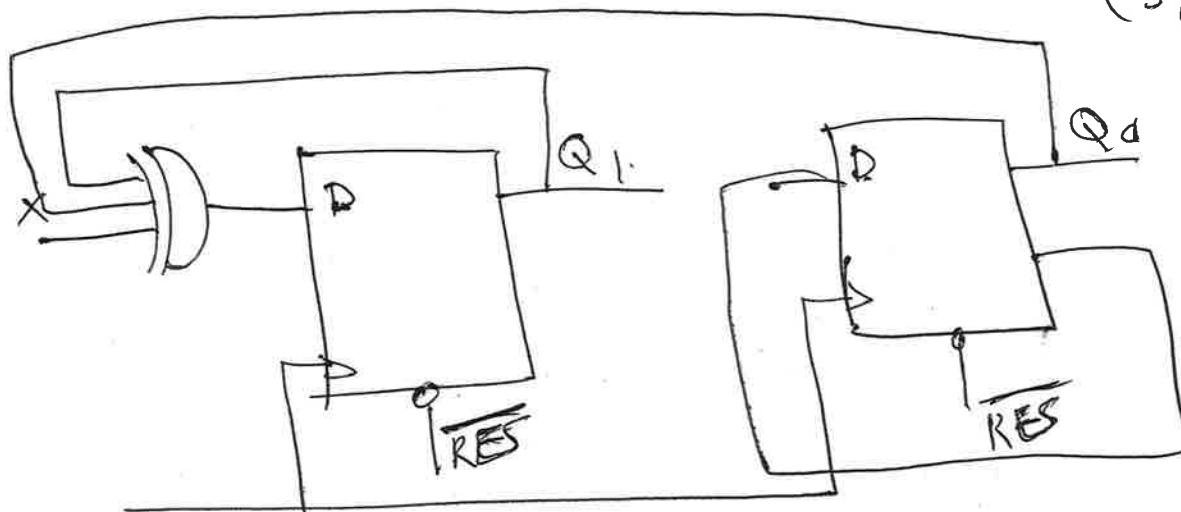
EEE/CSE 120 : Design sequential Finite state Machines

- Office Hrs T/TH 9:30-10:15 AM
- Midterm redo for extra credit up to 10 points
starting today at 6pm.
Till Friday (tomorrow) at 5:59 pm.
- HW 5 is due on oct 29

Sequential Circuit

↳ A bunch of ffs together.

↳ all ffs get the same
Clks at the same time
(Synchronous)



↳ ("counter")

$$\begin{cases} Q_1^+ = X \oplus Q_1 \oplus Q_0 \\ Q_0^+ = \overline{Q}_0 \end{cases}$$

Designing a Sequential FSM :

① State definition table.

states: Any combination of "0"s and "1"s

$$3 \text{ FFs} \rightarrow 2^3 = 8 \text{ states}$$

* Sometimes we don't use some of these states (Don't Cares)

② Draw state transition diagram.

↳ graphically represents how one state goes to another state encountering a clock pulse.

③ State transition table

↳ tells us if these are inputs and current states
where we would go next !

④ Design the next state logic.

Example: Build a counter that counts down from
 3 to 0 stopping at 0
 Decimal # 3, 2, 1, 0, 0, ...

①

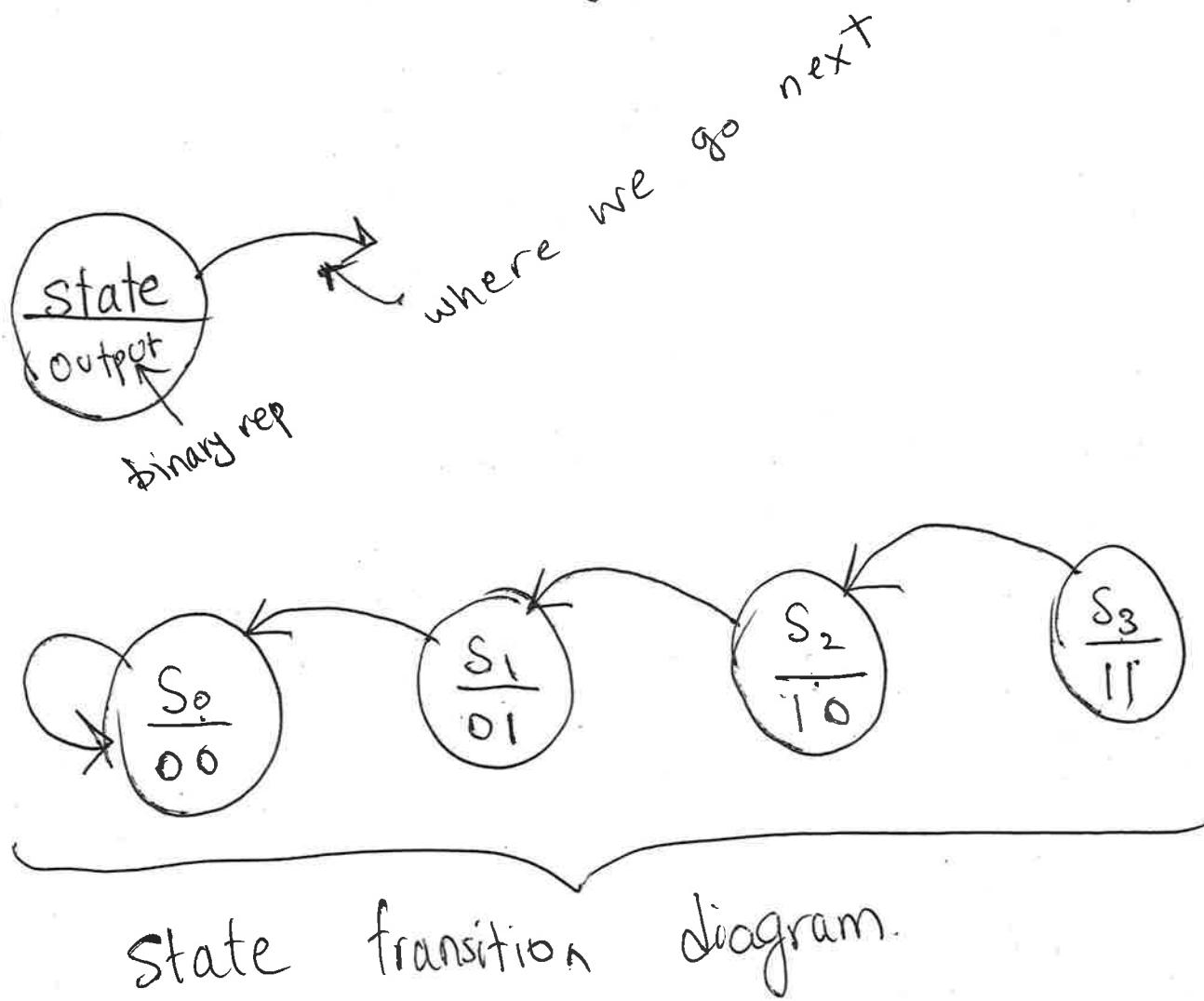
States	Definition	Binary rep. of states
S_0	Count = 0.	0 0
S_1	Count = 1	0 1
S_2	Count = 2	1 0
S_3	Count = 3	1 1

two digits needed



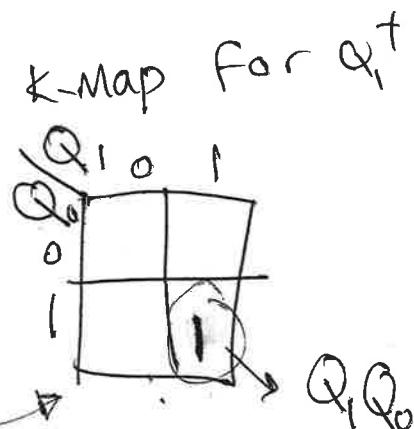
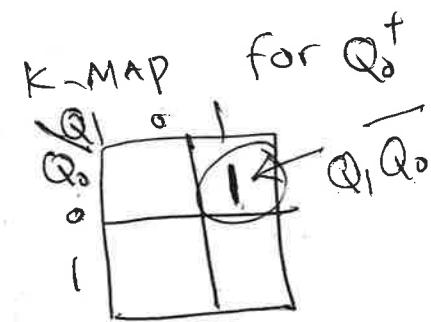
state def. table

② State transition diagram.



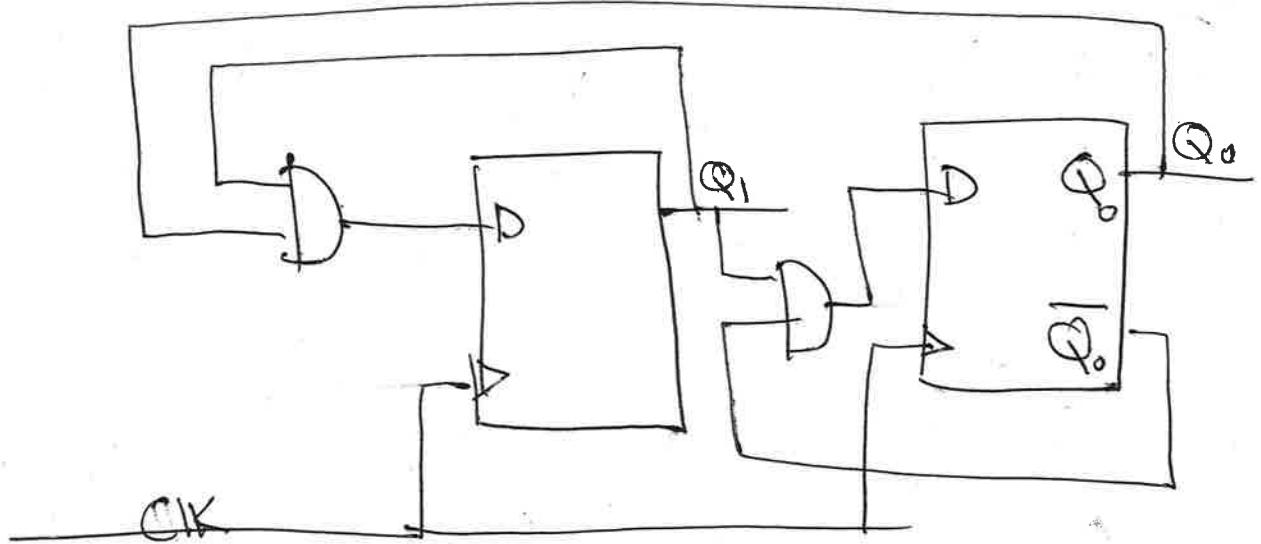
③ State transition table

State	Q_1	Q_0	Q_1^+	Q_0^+	next state
S_0	0	0	0	0	S_0
S_1	0	1	0	0	S_0
S_2	1	0	0	1	S_1
S_3	1	1	1	0	S_2



④ Design the circuit

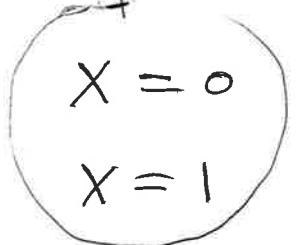
$$\begin{cases} Q_1^+ = Q_1 Q_0 \\ Q_0^+ = Q_1 \bar{Q}_0 \end{cases}$$



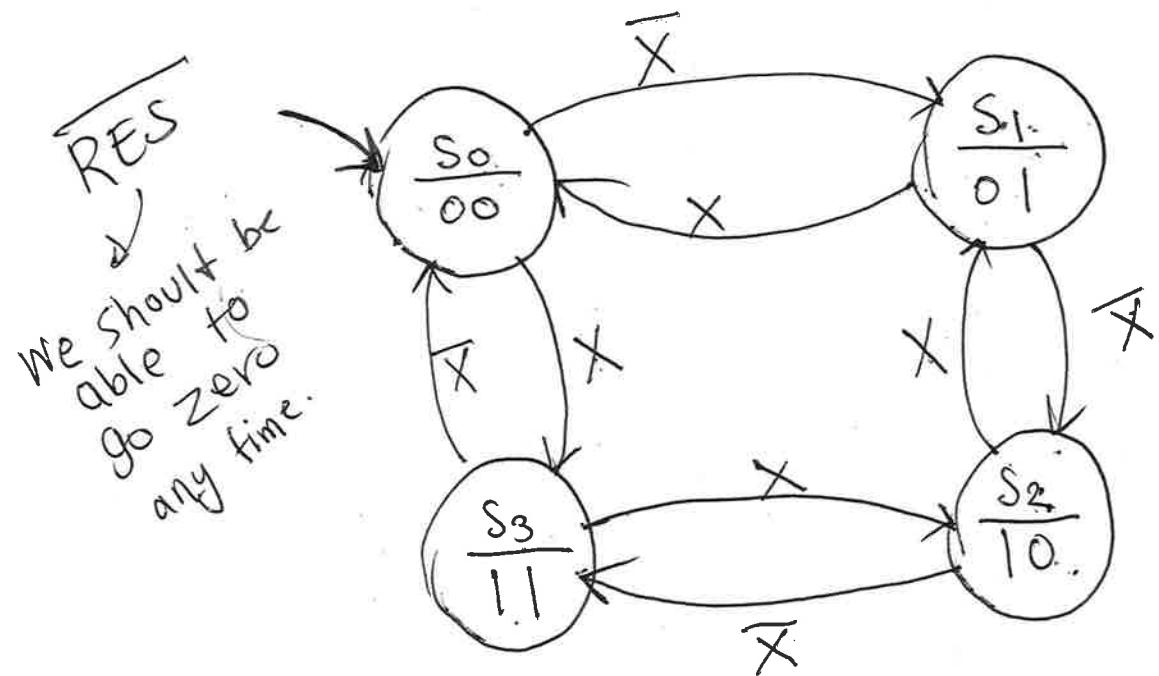
Example : Build a circuit that can count up /down

① State def. table

States	Def	binary rep
S_0	Count=0	0 0
S_1	Count<1	0 1
S_2	Count=2	1 0
S_3	Count=3	1 1

Input \rightarrow  : counting UP
 : counting down

② Draw transition diagram.



③ State transition table.

- prob. def
 $\begin{array}{c} \xrightarrow{\text{Counting}} 0 \\ 3 \xrightarrow{\text{up}} \end{array}$
- prob. def
 $\begin{array}{c} \xrightarrow{\text{Counting}} 3 \\ 0 \xrightarrow{\text{down}} \end{array}$

States	input X	Q_1	Q_0	Q_1^+	Q_0^+
S_0	0	0	0	0	1
S_0	1	0	0	1	1
S_1	0	0	1	1	0
S_1	1	0	1	0	0
S_2	0	1	0	1	1
S_2	1	1	0	0	1
S_3	0	1	1	0	0
S_3	1	1	1	1	0

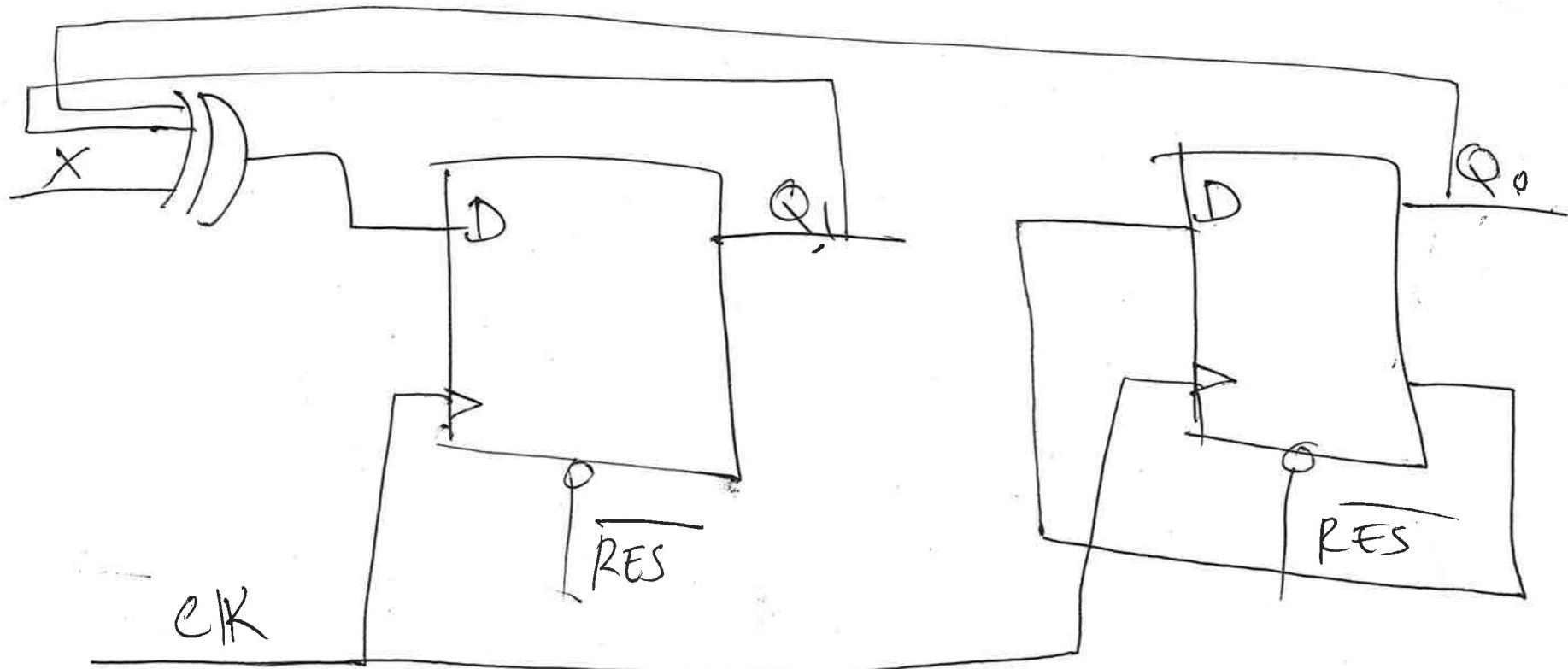
Design :

$X\bar{Q}_1$	00	01	10	11
\bar{Q}_0	0	1	1	1
0	1	1	1	1
1	1	1	1	1

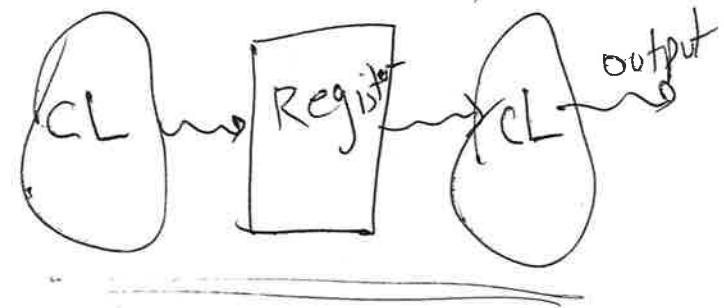
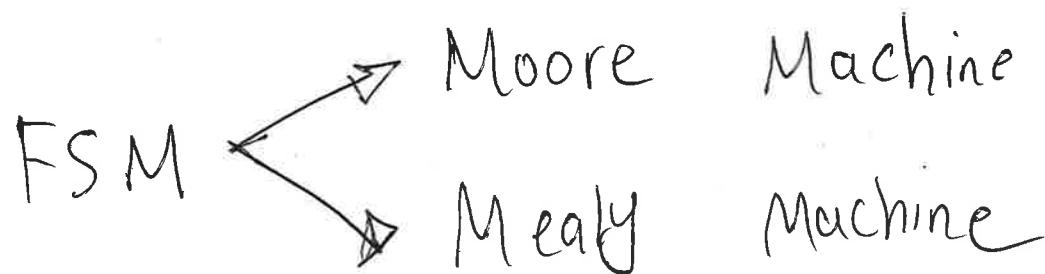
$$\text{For } Q_1^+ = X \oplus Q_1 \oplus Q_0$$

$X\bar{Q}_1$	00	01	11	10
\bar{Q}_0	0	1	1	1
0	1	1	1	1
1	1	1	1	1

for $Q_0^+ = \bar{Q}_0$



- $2^{\# \text{FFS}_{\text{Finite}}} = \# \text{of states}_{\# \text{Finite}} \Rightarrow$ called Finite state machine (FSM)

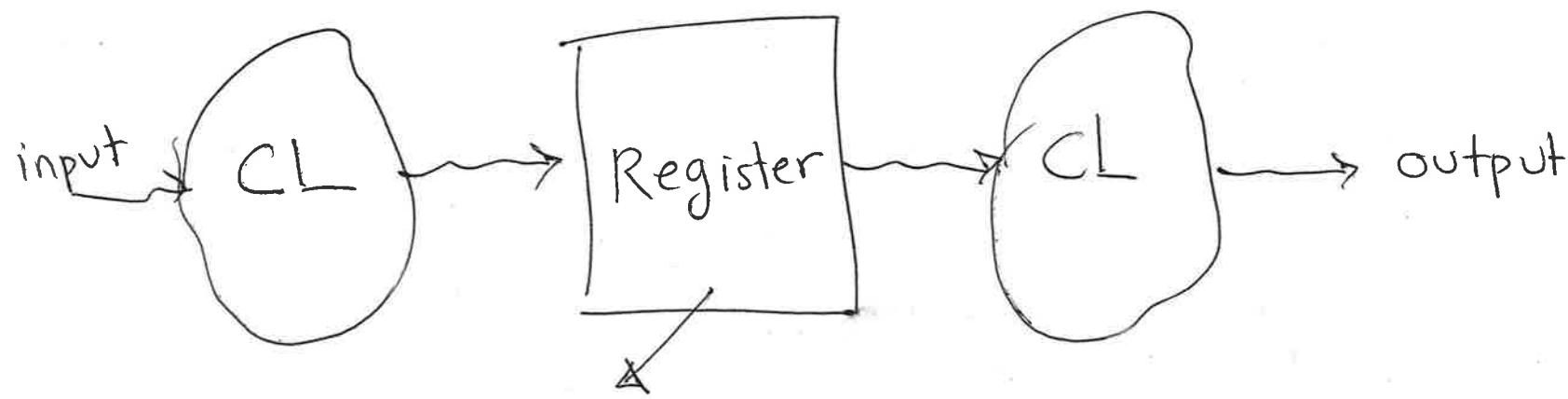


EEE/CSE 120 : Finite State Machines

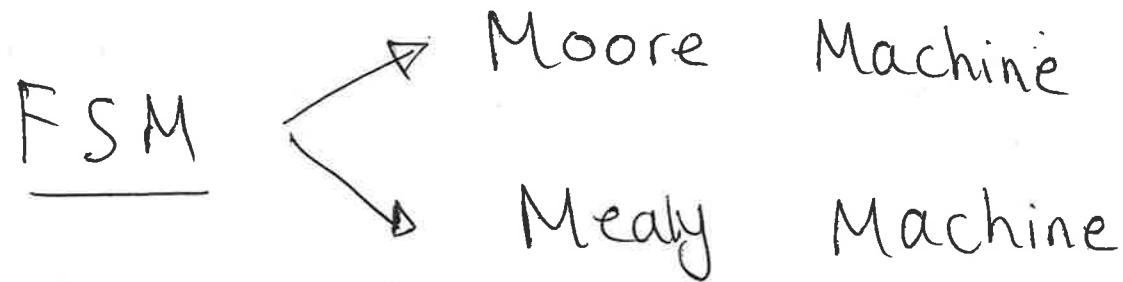
* HW 5 is due Oct 29

* Poll \rightarrow due Oct 31

\hookrightarrow Canvas \rightarrow Quizzes \rightarrow Poll.

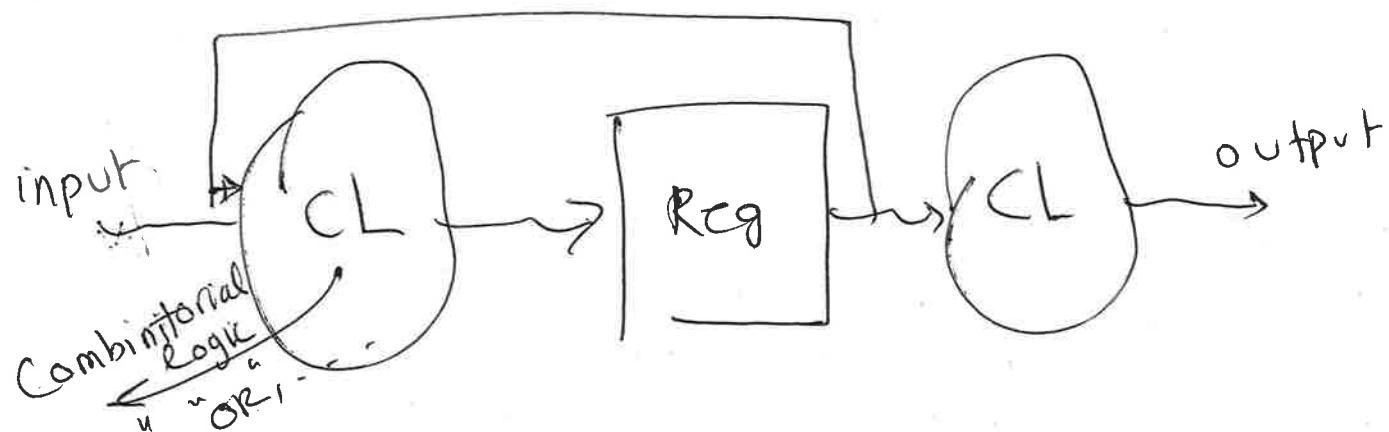


$$2^{\# \text{ of FFs}} = \# \text{ of states}$$



Moore Machine:

A FSM is Moore if there is no direct connection from input to output.



* only change (output) if next clock happens.
(unless there is a reset)

Example : Design a Moore machine Such that

Key pad w/ $\begin{cases} X=0 \\ X=1 \end{cases}$

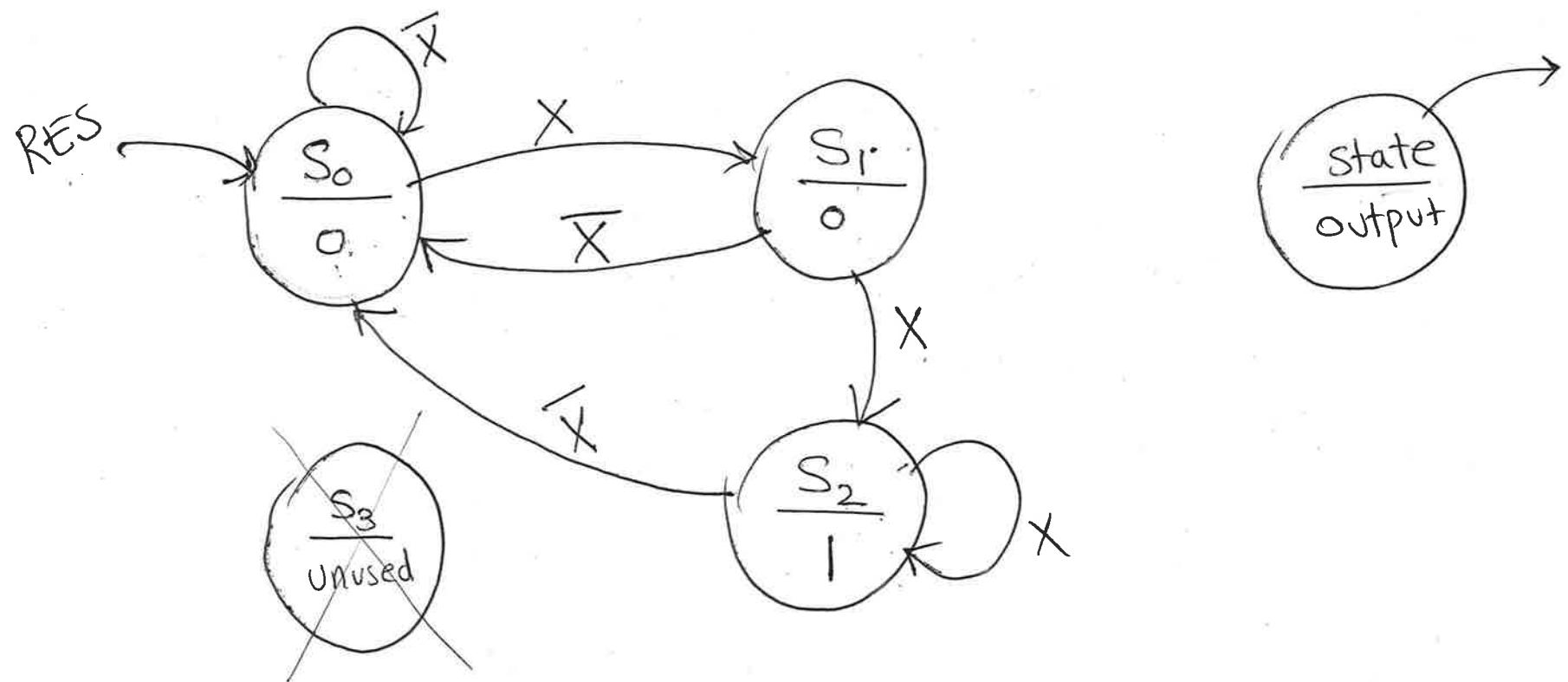
press "1" twice in a row then the
Safe opens up.

① State def. table:

states	name (def)	binary rep.	output
S_0	IDLE	0 0	?
S_1	One-one	0 1	?
S_2	two - Ones	1 0	1
S_3	UNUSED	1 1 Q_1 Q_0	X

input:
 $\begin{cases} X=0 \\ X=1 \end{cases}$

② Draw the state diagram



③ State transition table

states	input=x	Q_1		Q_0		Q_1^+	Q_0^+	output
		0	1	0	1			
S_0	0	0	0	0	0	0	0	0
S_0	1	0	0	0	1	1	0	0
S_1	0	0	1	0	0	0	0	0
S_1	1	0	1	1	0	0	0	0
S_2	0	1	0	0	0	0	1	1
S_2	1	1	0	1	0	1	0	1
S_3	0	1	1	x	x	x	x	x
S_3	1	1	1	x	x	x	x	x

depends on
the current states
and not the
future states!

④ Design the circuit

For Q_1^+

\bar{Q}_0	Q_0	Q_1	XQ_1
0	00	01	11
1	10	X	X

$$Q_1^+ = XQ_0 + XQ_1$$

For output

\bar{Q}_0	Q_0	Q_1	XQ_1
0	00	01	11
1	10	X	X

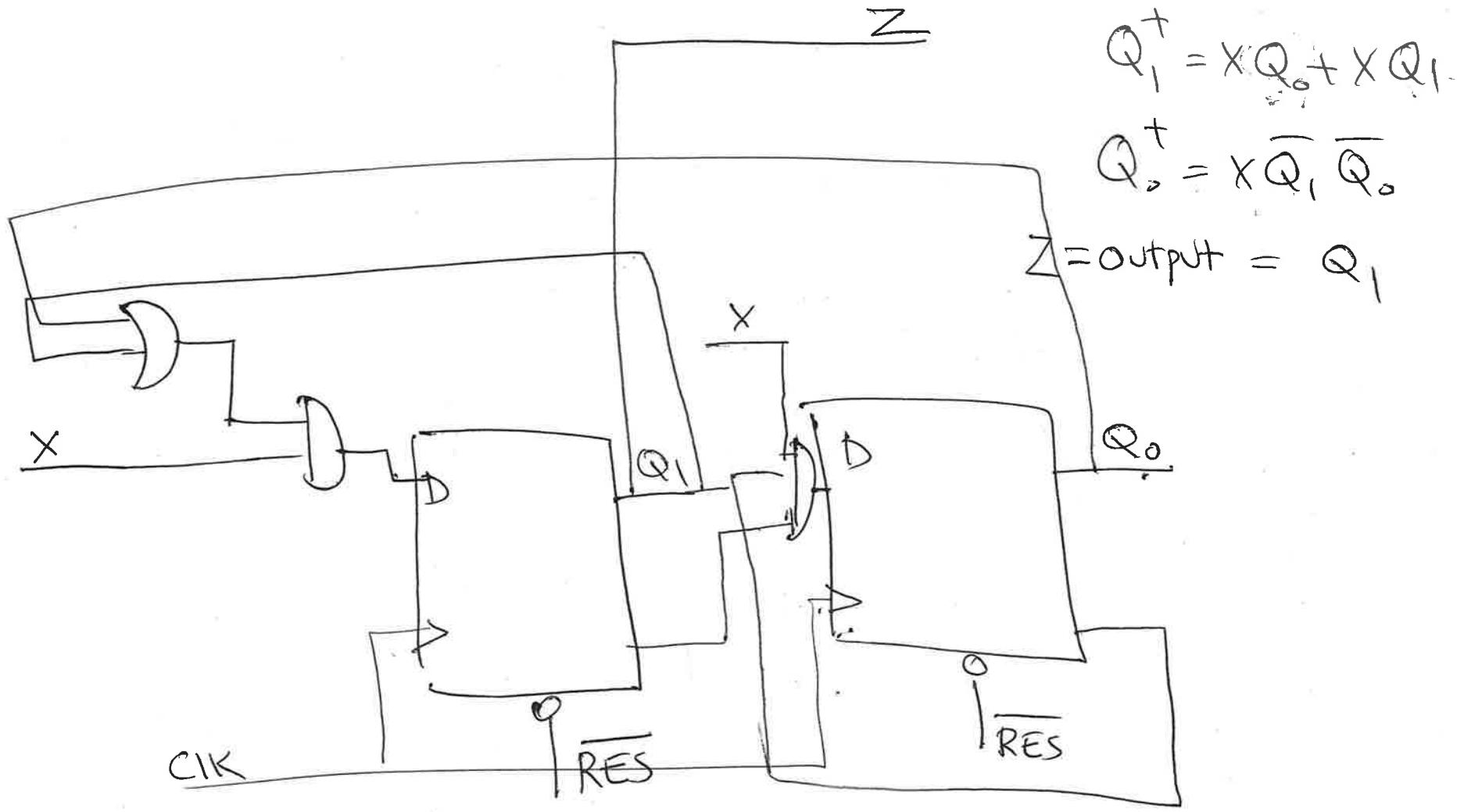
For Q_0^+

\bar{Q}_0	Q_0	Q_1	XQ_1
0	00	01	11
1	10	X	X

$$Q_0^+ = X\bar{Q}_1\bar{Q}_0$$

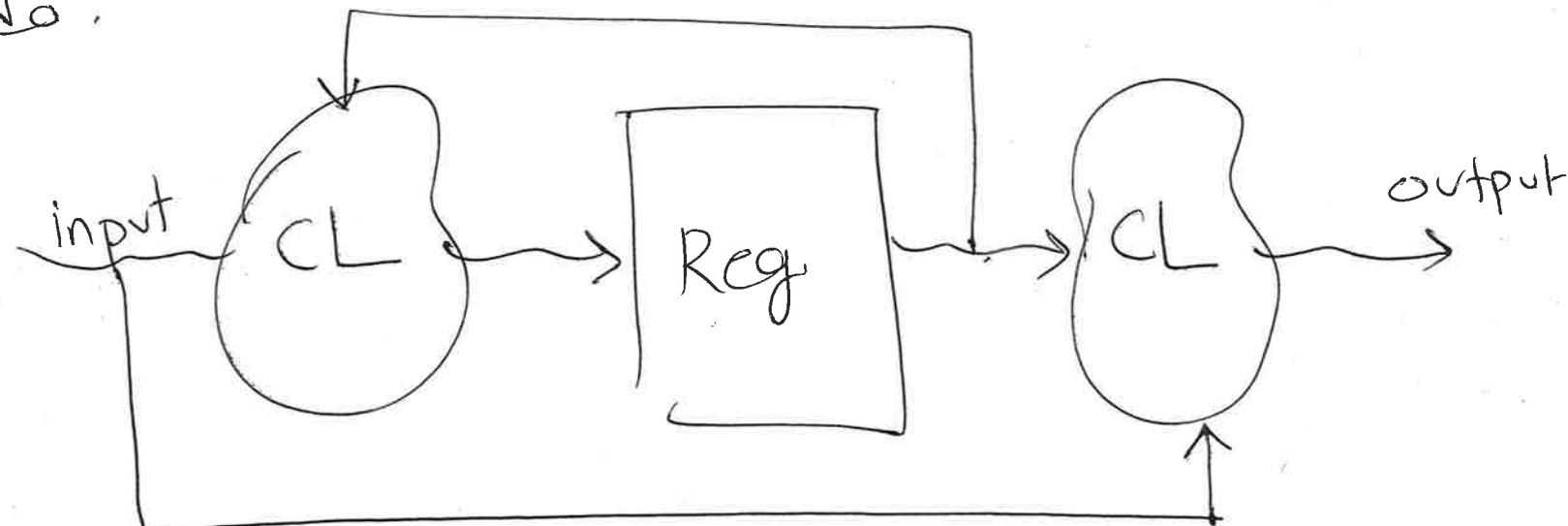
$$\text{output} = \underbrace{Q_1}_{1}$$

is only a funct.
of current state
 Q_0, Q_1 .



Mealy Machine :

- Is a FSM that the output depends on the states of the flip flops as well as the input.
- Mealy machine can do whatever moore machines can do.



{ Moore Machine : Input change \Rightarrow for output to change we have to wait for the next clock

Mealy Machine : Input change \Rightarrow output change.

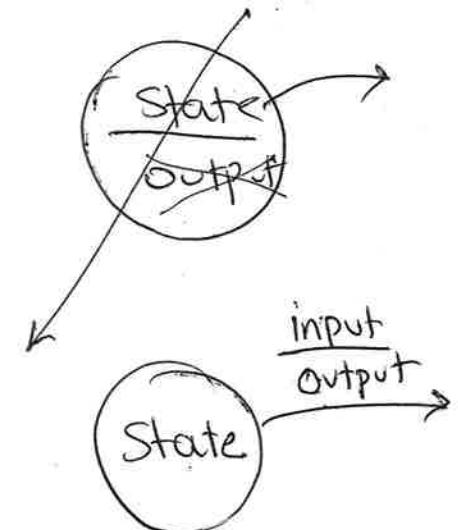
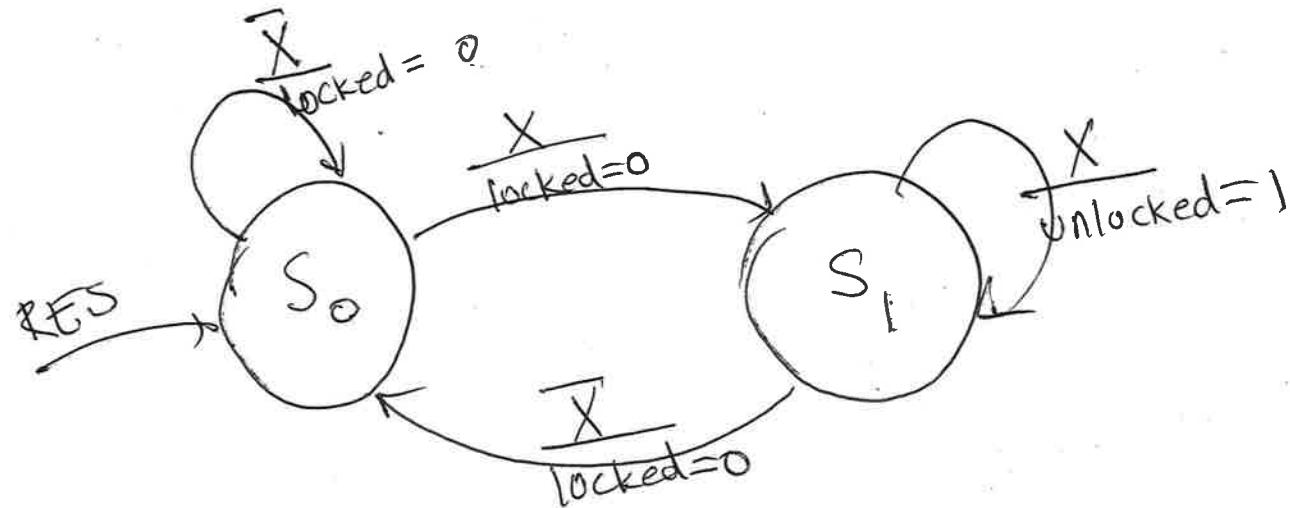
Example: Key pad $\begin{cases} X=0 \\ X=1 \end{cases}$, we press two ones in a row, then we open up the safe.

Design a Mealy Machine.

① State def. table.

States	State def	binary rep.
S_0	IDLE	0
S_1	Count the first one	1

② Draw state diagram



- Moore output is the output of flip flop and the output state
- Mealy Machine the output is a function of inputs.

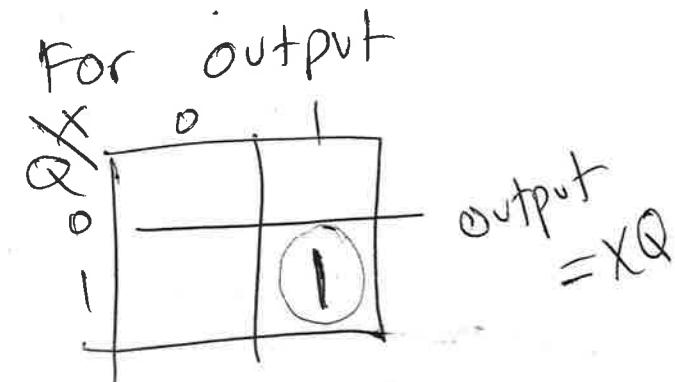
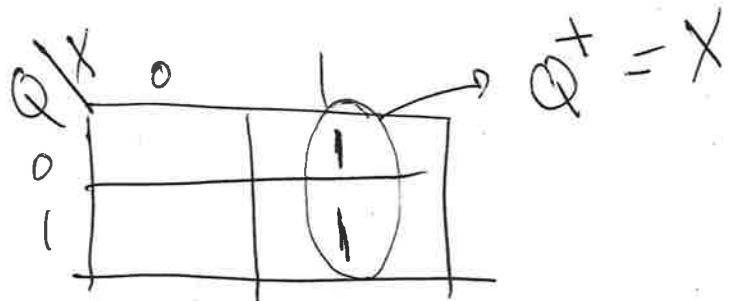
③ State transition table

<u>input X</u>	<u>Q</u>	<u>Q^+</u>	<u>output</u>
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	1

This depends
on input

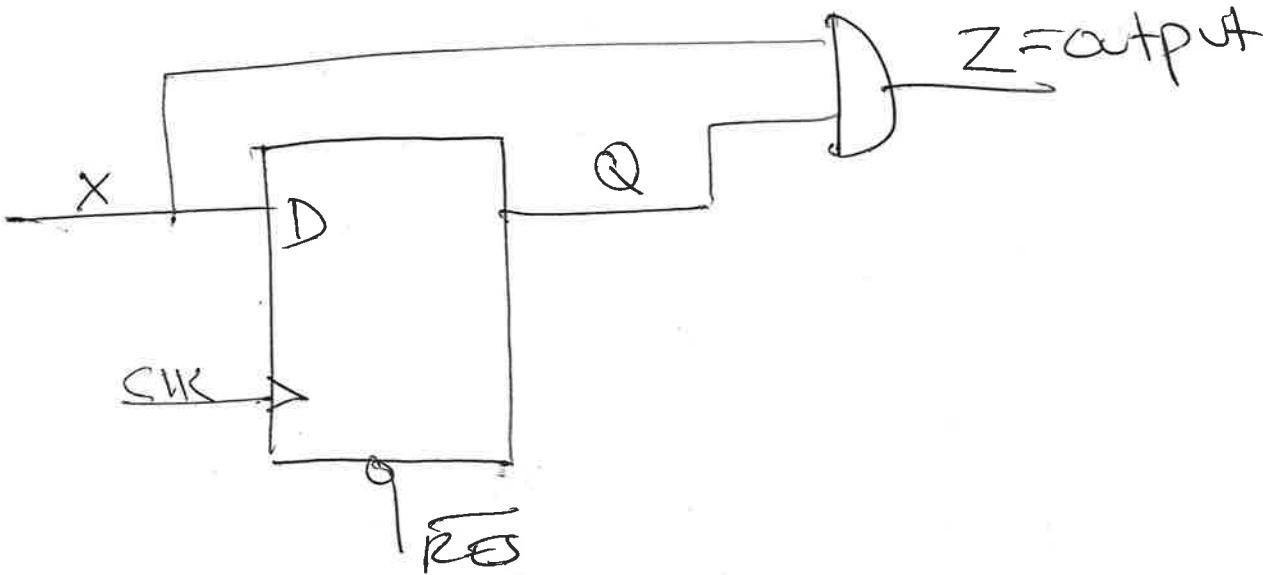
④ Design the circuit

For Q^+



$$Q^+ = X$$

$$Z = \text{output} = XQ$$



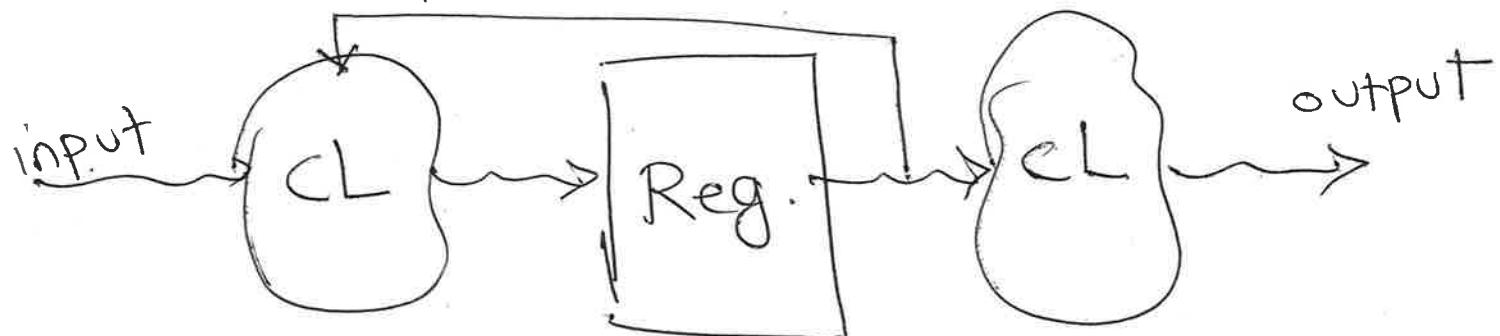
- There is timing issue

→ Synchronization

→ Adding one
ff.

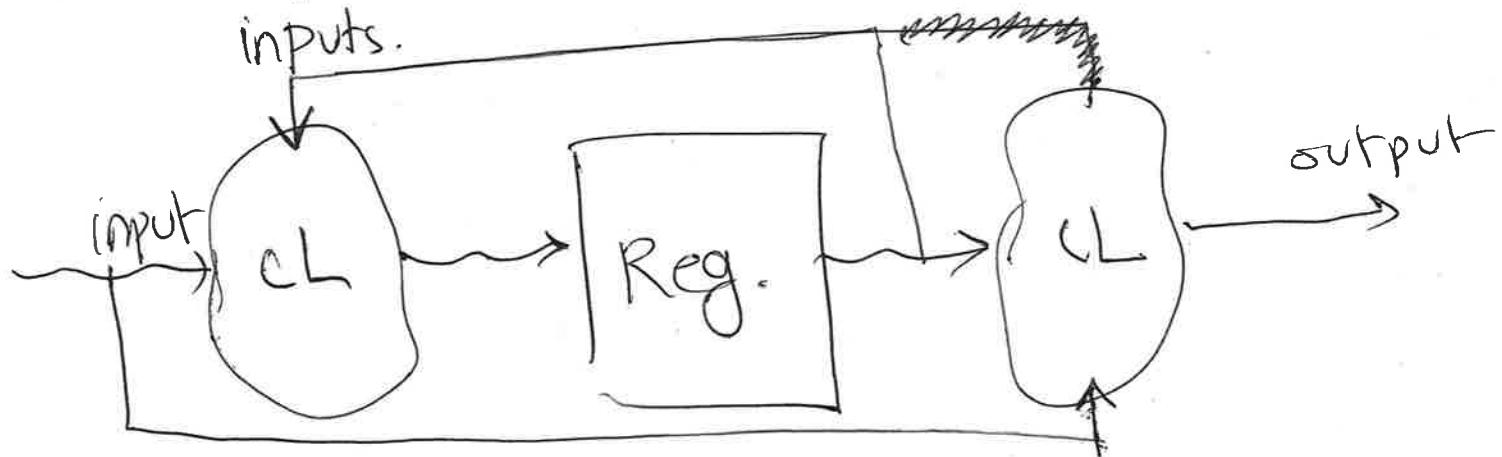
① Moore Machine:

FSM, there is no direct relationship between input & output



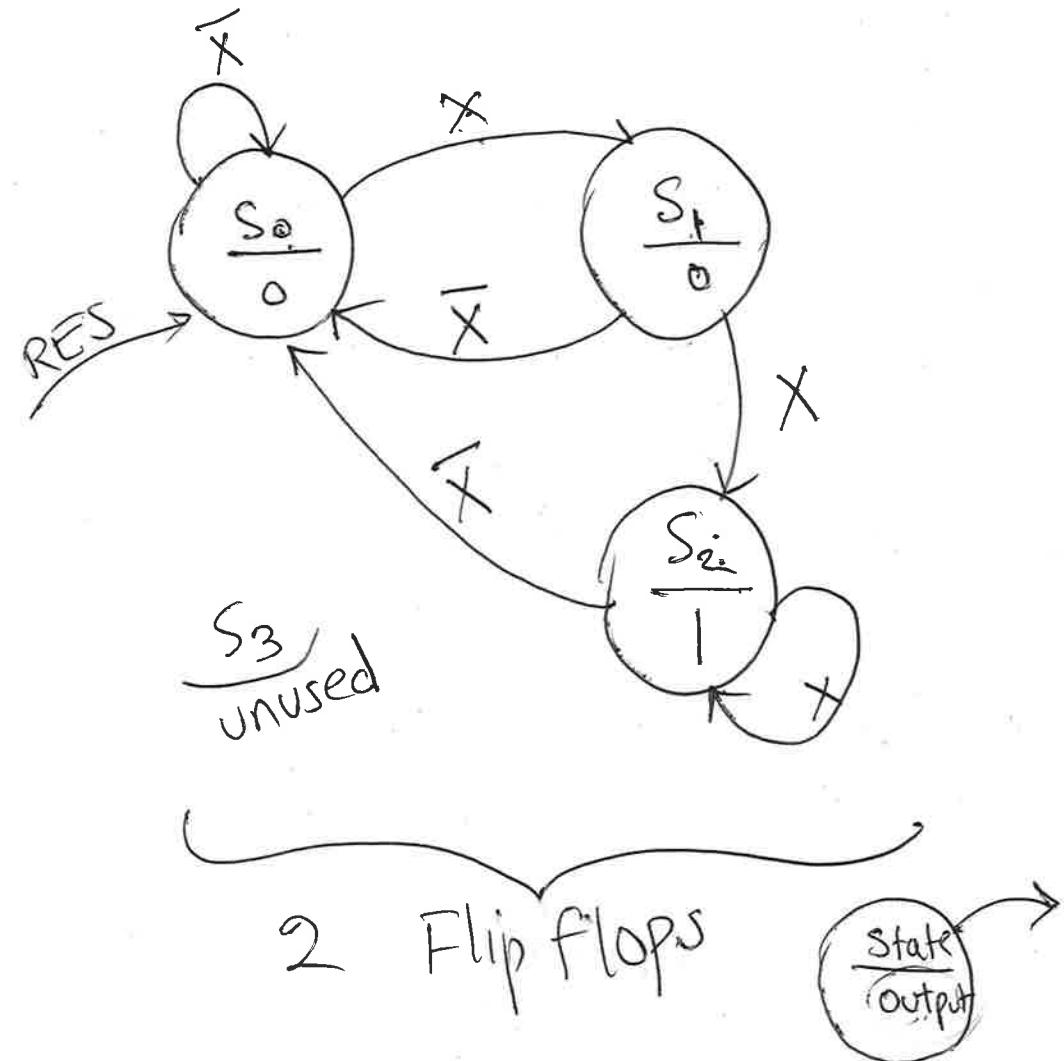
② Mealy Machine:

FSM: output change depends on both states & inputs.

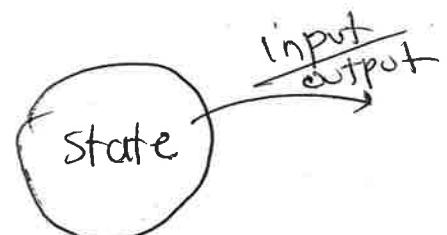
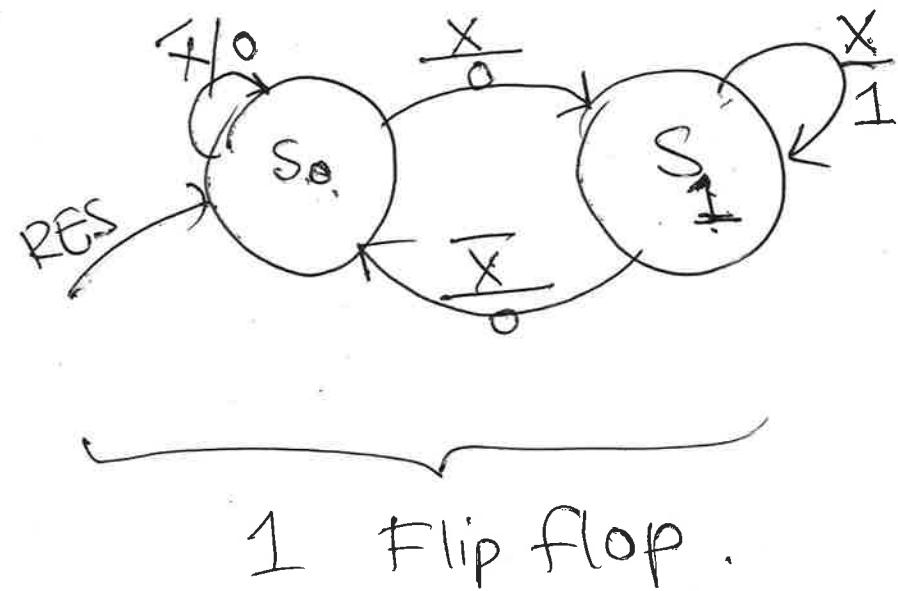


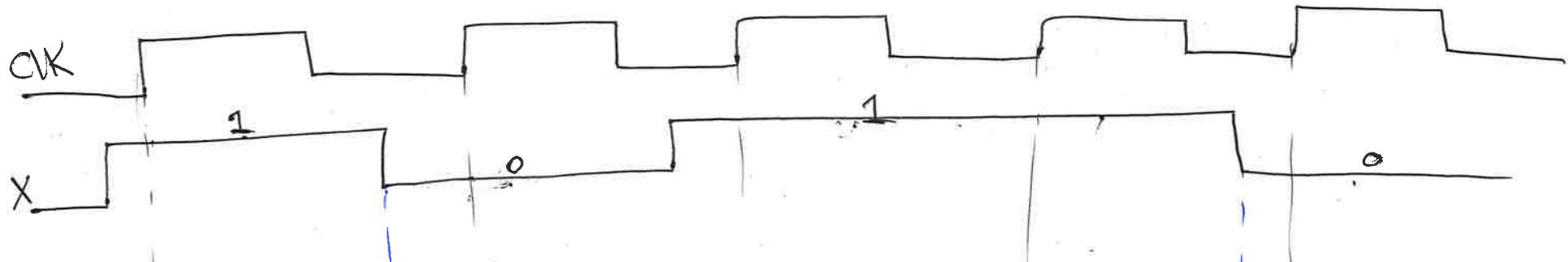
Review of the example : Keypad $\begin{cases} X=0 \\ X=1 \end{cases}$, safe opens up if we have two "1"s in a row.

Moore Machine

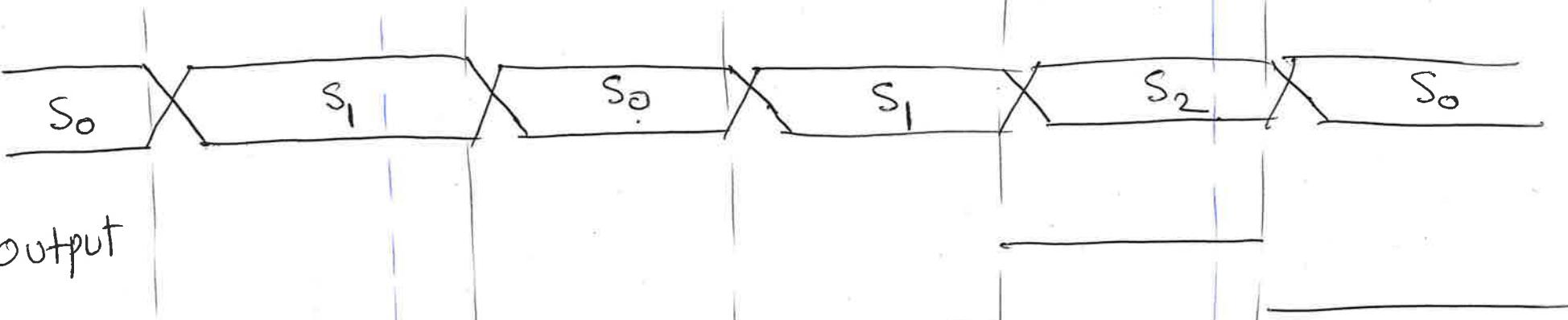


Mealy Machine



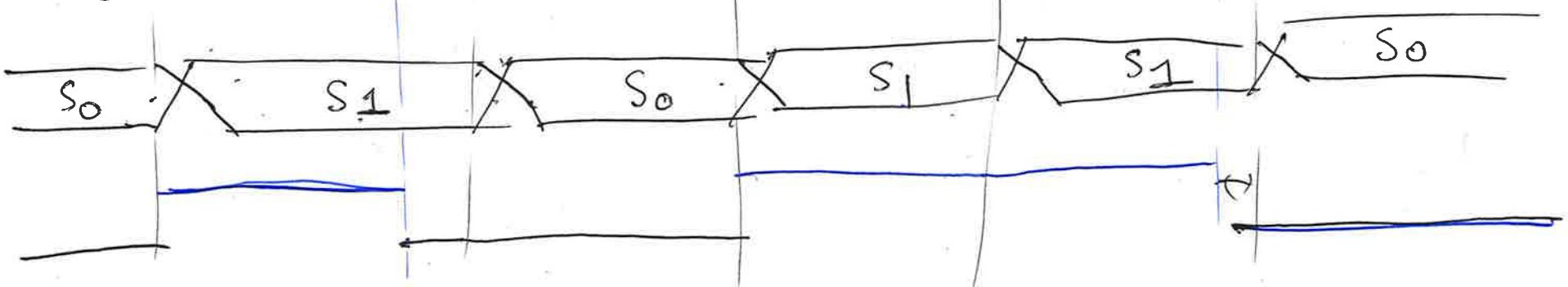


Moore Machine:

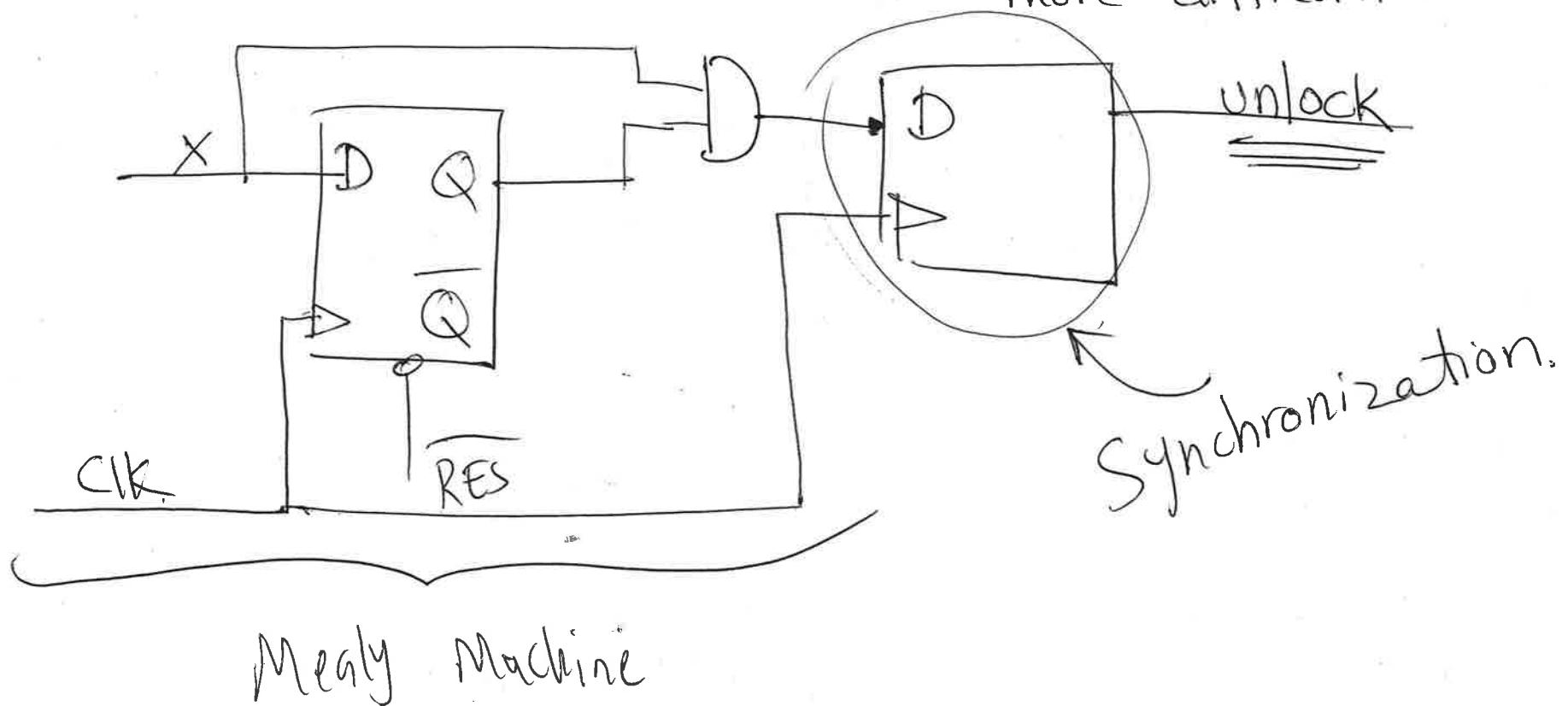


output

Mealy Machine



- Mealy Machines use less number of ffs,
 - ↳ however, we often have timing issue!
 - ↳ Asynchronous.
 - ↳ by adding another FF.
 - ↳ Makes analysis much more difficult.



Example: Lets say we have sensor that can tell us if someone enters or exists

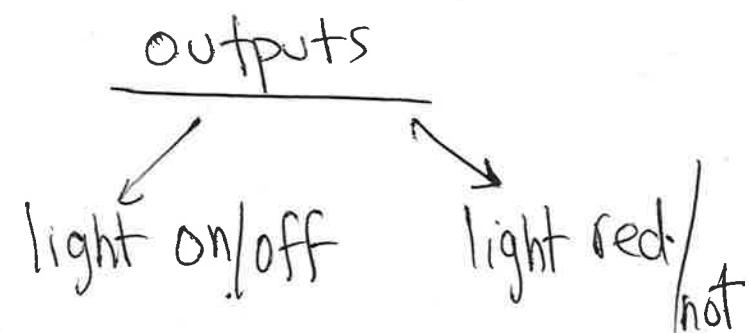
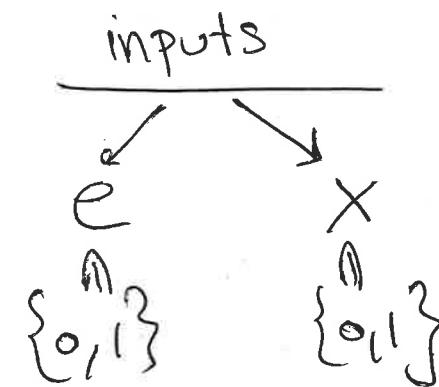
inputs = {
e entering
x exiting}

- Max # of people in the room = 3
- only one person at a time can leave / enter.
- If the room is full , red light on
- If the room is empty , light is off

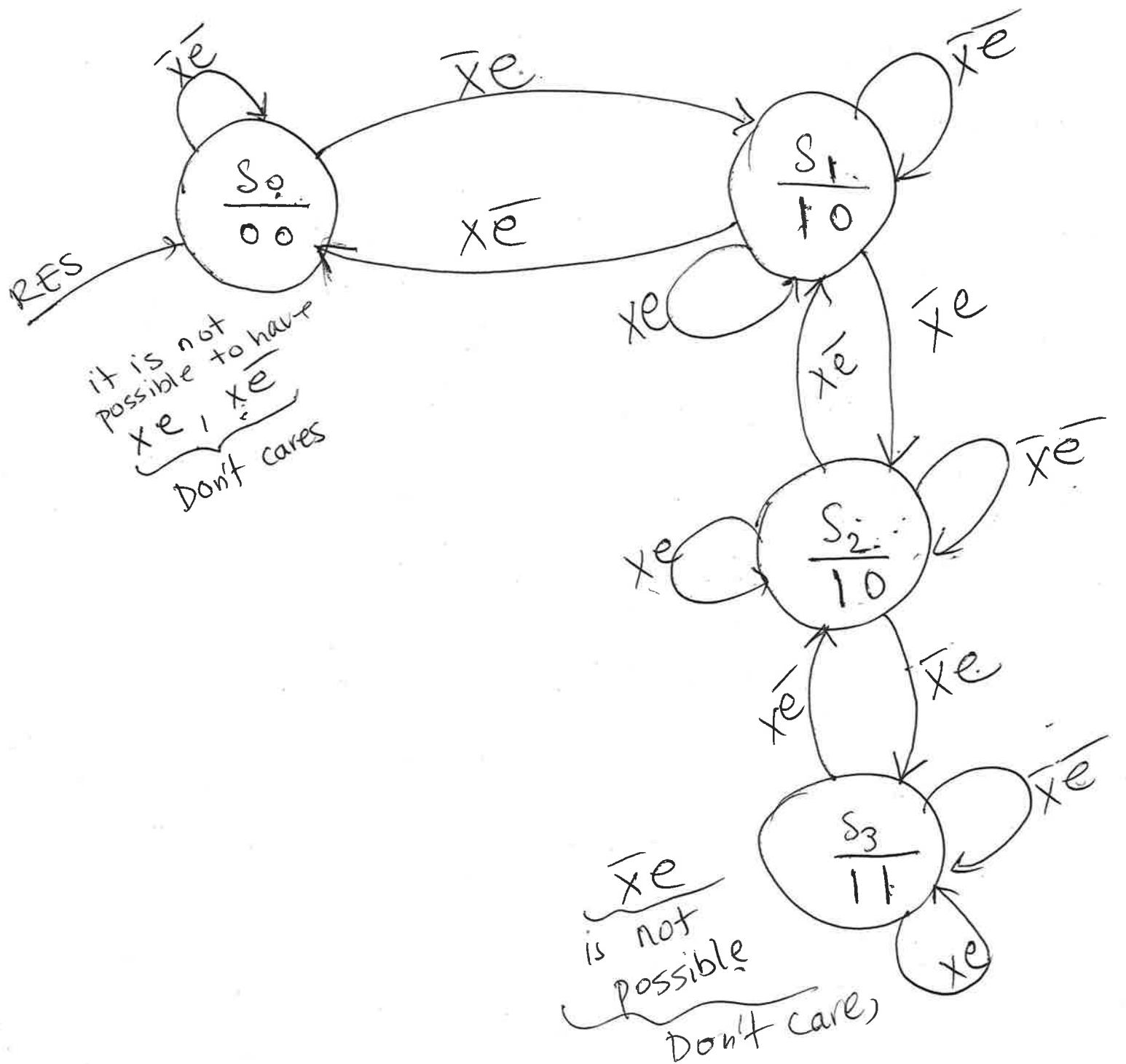
Q: Design a light controller that can do the above .

① State def. table:

state	def.	binary rep.
S_0	light off & room empty	0 0
S_1	light on & one person	0 1
S_2	light on & two people	1 0
S_3	light on & 3 people	1 \bar{Q}_1 \bar{Q}_0



② state diagram

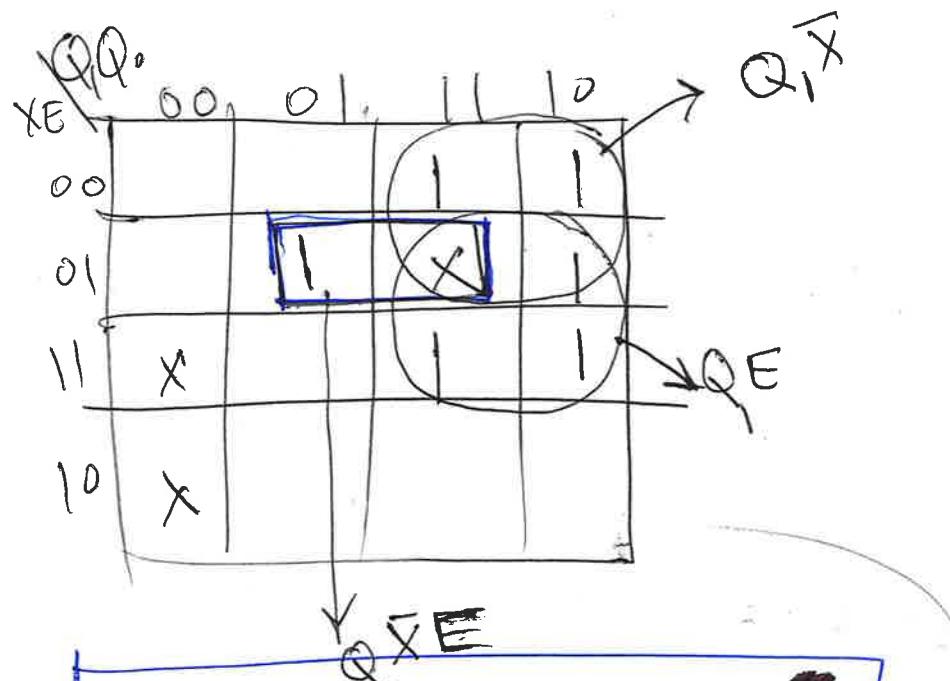


③ State transition table.

	Q_1	Q_0	X	E	Q_1^+	Q_0^+	L	light	red light
state s_0	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	1	0	0	0
	0	0	1	0	X	X	X	X	X
	0	0	1	1	X	X	X	X	X
s_1	0	1	0	0	0	1	1	0	0
	0	1	0	1	1	0	1	0	0
	0	1	1	0	0	0	1	0	0
	0	1	1	1	0	1	1	0	0
s_2	1	0	0	0	1	0	1	0	0
	1	0	0	1	1	1	1	0	0
	1	0	1	0	0	1	1	0	0
	1	0	1	1	1	0	1	0	0
s_3	1	1	0	0	1	1	1	1	1
	1	1	0	1	X	X	X	X	X
	1	1	1	0	1	0	1	1	1
	1	1	1	1	1	1	1	1	1

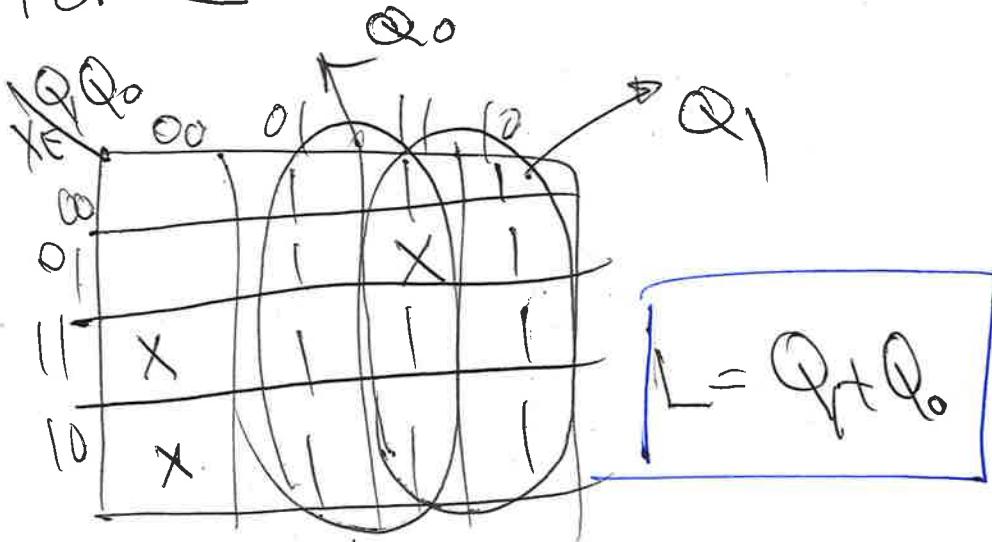
output only
depends on
current stat
"NOT"
the
next
state

④ For Q_1^+



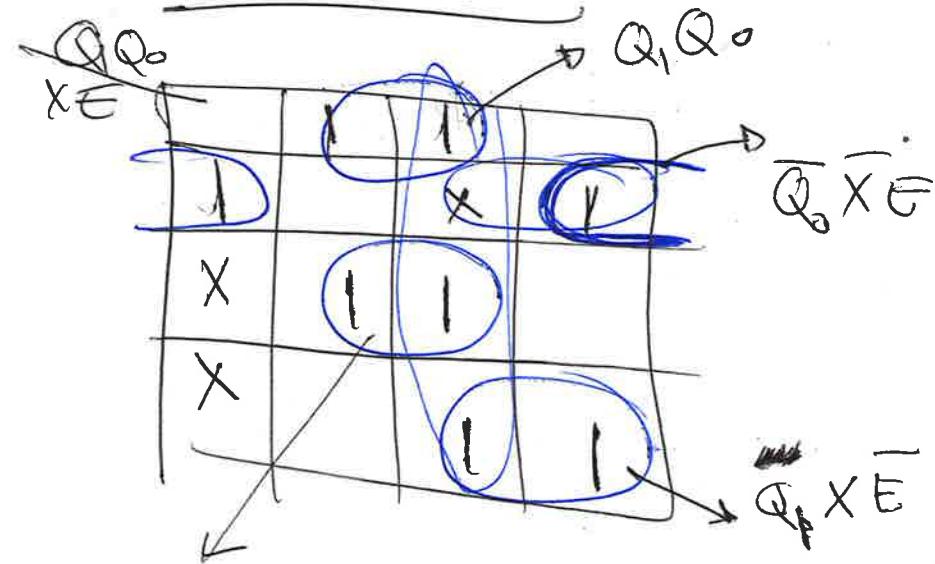
$$Q_1^+ = Q_1 \bar{X} + Q_1 E + Q_0 \bar{X} E$$

For L



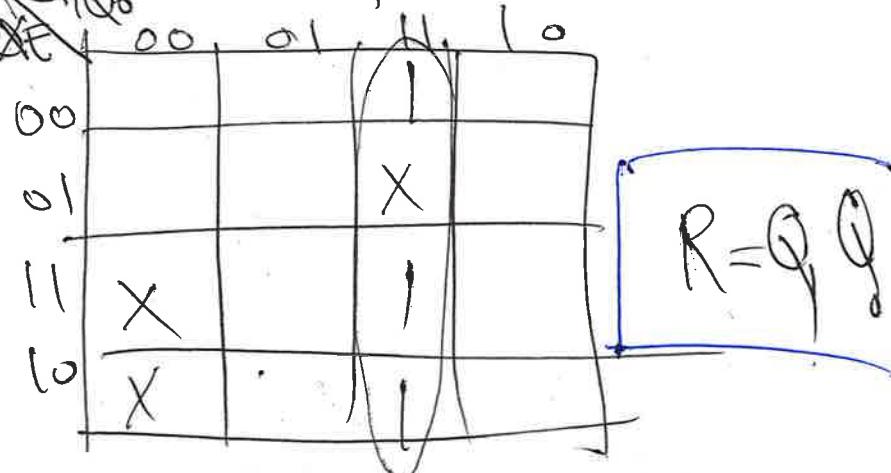
$$L = Q_1 + Q_0$$

For Q_0^+



$$Q_0^+ = Q_1 Q_0 + \bar{Q}_0 X E + Q_1 \bar{X} E + \bar{Q}_0 \bar{X} E$$

For R



$$R = Q_1 Q_0$$

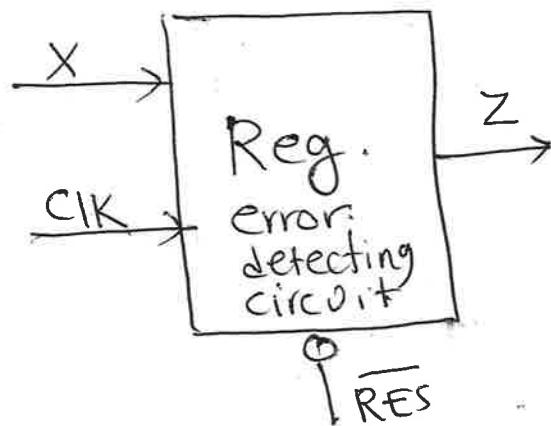
Lecture 21

EEE/CSE 120 : Capstone project

(Nov 3, 2020)

- Office Hrs { T : 9:30 - 10:15 AM
 } TH 3:15 - 4:00 PM (only this week)
- Capstone project is uploaded
- HW6 is uploaded.
- Get ready for Quiz 2
 - {-Registers
 - timing diagram
- Midterm is graded and grades will be uploaded soon.

Example : Design a Mealy machine for an error detector.



- Single input
- Single output.

output $Z = 1$ (error)

error: Two zeros in a row
"00" or three ones
in a row "111"

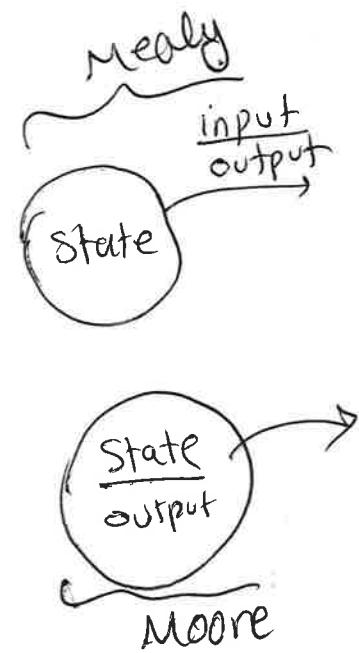
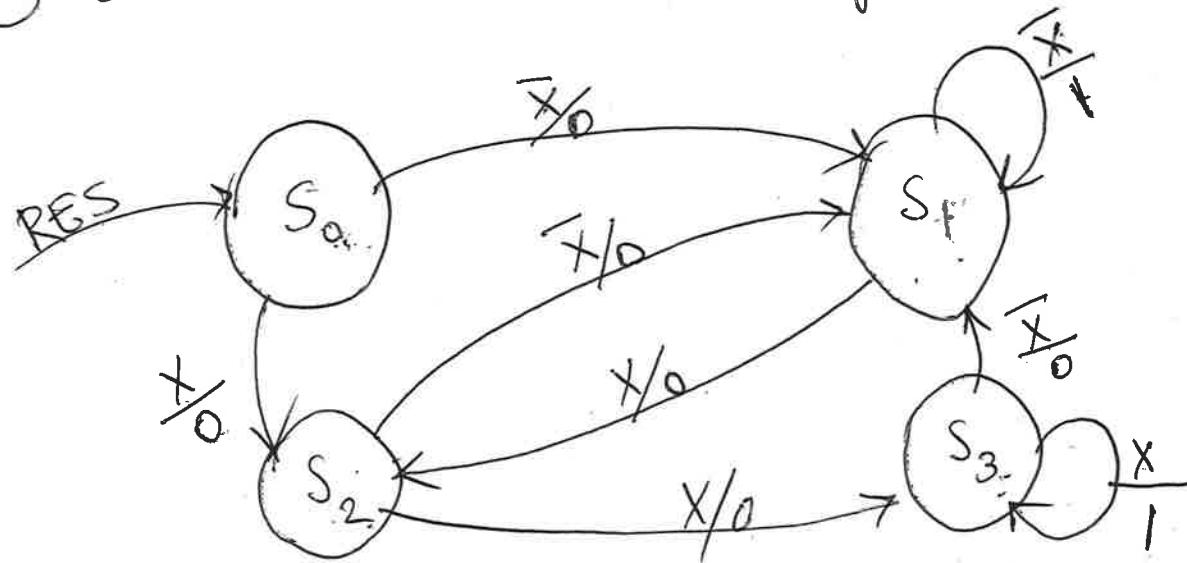
Example :

X	0	0	1	1	1	1	1	1	0	0	0	1	1	1	0	0
Z	0	1	0	0	1	1	1	1	0	1	1	0	0	1	0	1

① State def. table

State	Def.	binary rep.
S_0	IDLE	0 0
S_1	One-zero	0 1
S_2	One-one	1 0
S_3	two-ones	1 1 Q ₁ Q ₀

② State transition diagram

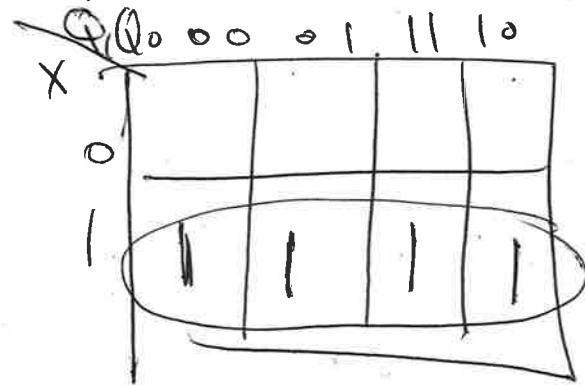


③ State transition table

	Q_1	Q_0	X	Q_1^+	Q_0^+	Z	$= \text{output}$
S_0	0	0	0	0	1.	0	
	0.	0.	1.	1.	0	0	
S_1	0	1	0	0	1	1	
	0	1	1	1	0	0	
S_2	1	0	0	0	1.	0	
	1	0	1	1	1.	0	
S_3	1	1	0	0	1	0	
	1	1	1	1	1	1	

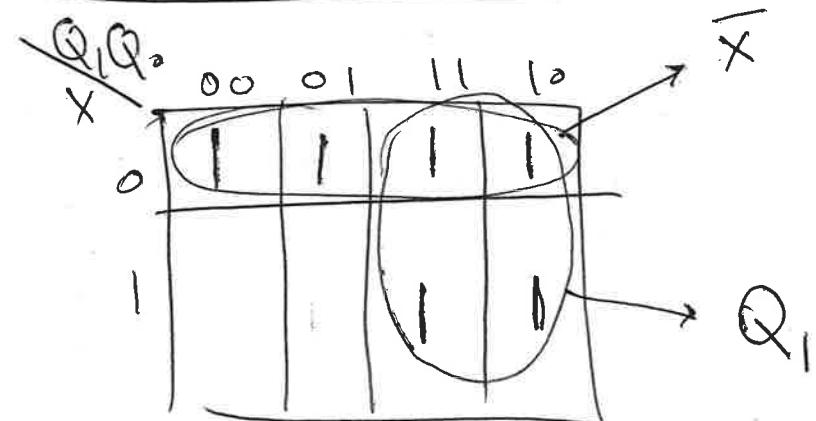
④ Design

K-Map for Q_1^+



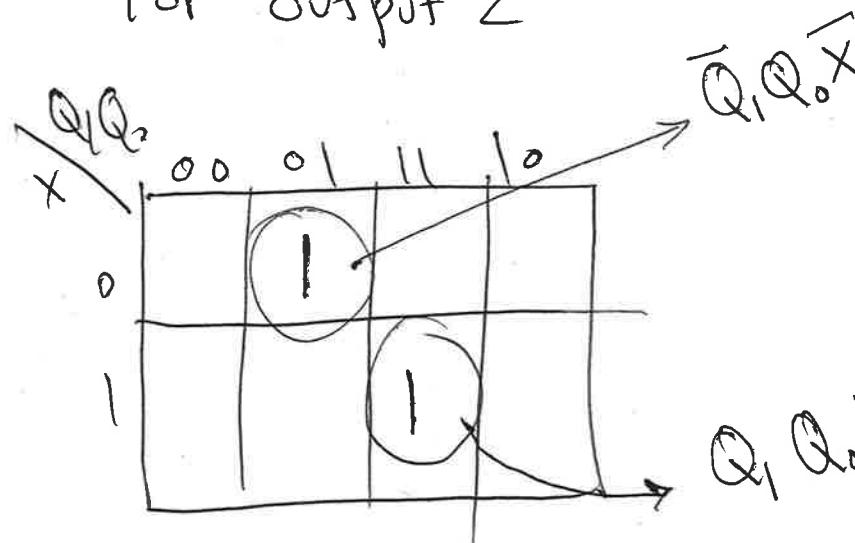
$$Q_1^+ = X$$

K-Map Q_0^+



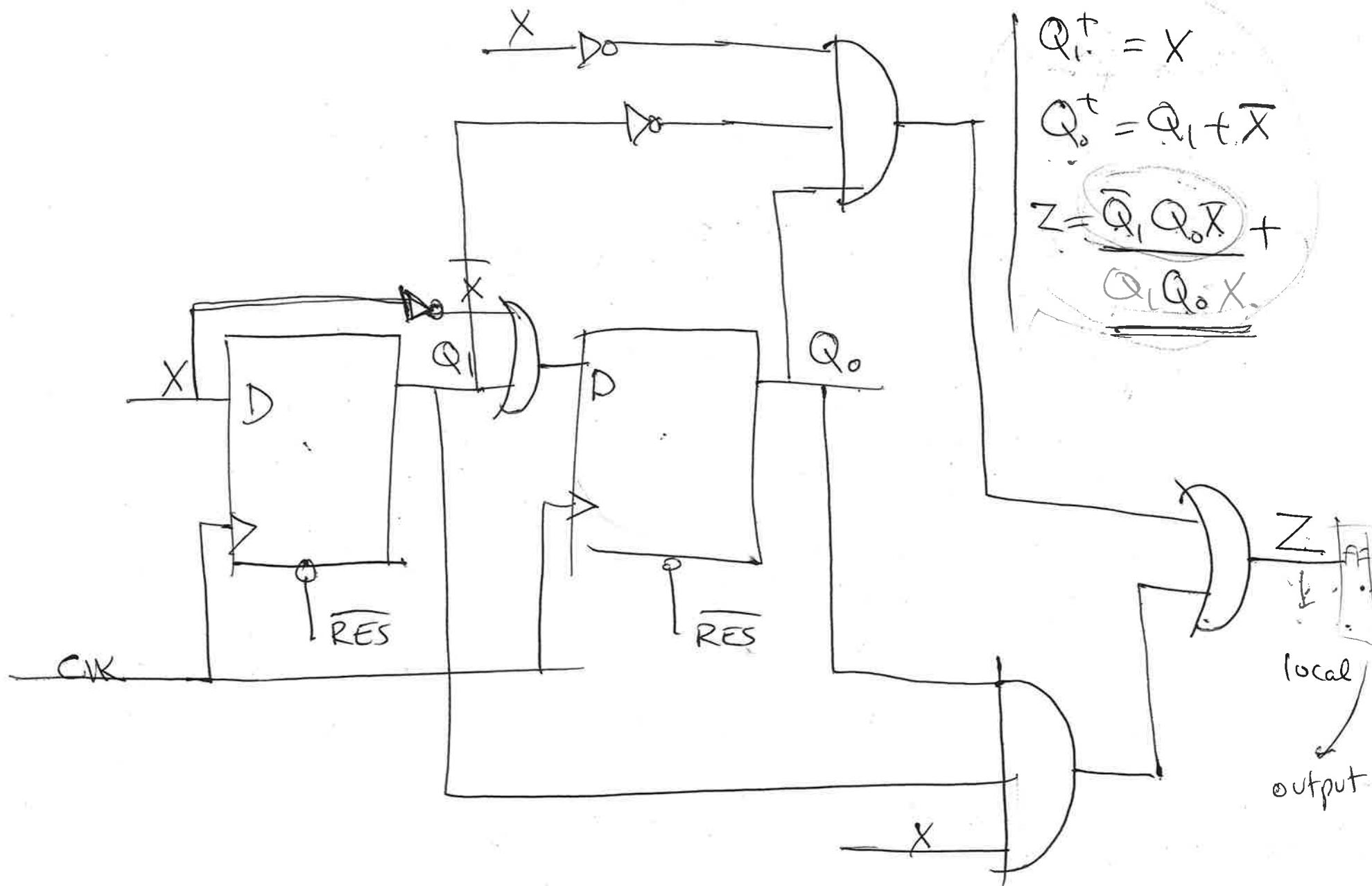
$$Q_0^+ = Q_1 + \bar{X}$$

For output Z



$$Z = \bar{Q}_1 \bar{Q}_0 X +$$

$$Q_1 Q_0 X$$



EEE/CSE 120 : More examples on FSMs

NOV 5, 2020

- Office hrs "today" \rightarrow 3:15 - 4:00 pm
- Capstone project is uploaded
- Grades are uploaded
- Assignment 6 is uploaded.
- Quiz 2 \rightarrow NOV 12 \rightarrow Registers

timing diagrams

- D flip flops

- JK flip flops, Toggle flip flops

- behavioral eq.

Q: Design an alarm system.

Alarm disarmed $\xrightarrow{0 \mid 0}$ Alarm armed

Alarm armed $\xrightarrow{0 \mid 0}$ Alarm disarmed

- If we press a wrong number, we go back to the beginning.

- Inputs : { c : input value

$v = \begin{cases} 0 & \text{If } c \text{ is invalid} \\ 1 & \text{If } c \text{ is valid} \end{cases}$

transation :

press a key

$$\Rightarrow v = 1$$

otherwise
 $v = 0$

- Output = Alarm is armed = A

Remark: States are input of ffs and ffs are storage elements. \Rightarrow states are for what's to remember the assignments.

① State def. table

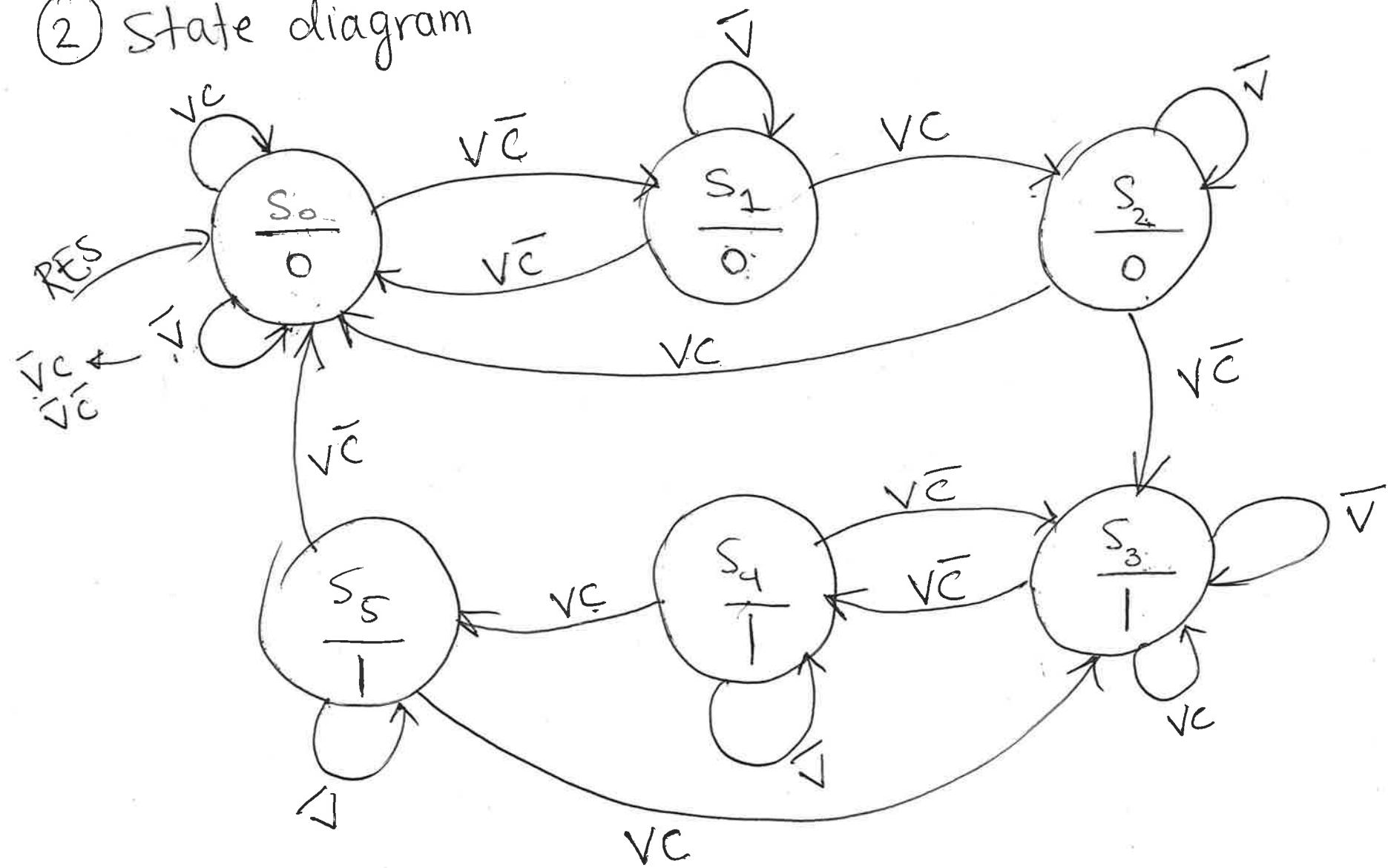
States	Def.	Binary rep.
S_0	disalarmed	0 0 0
S_1	disalarmed / 0	0 0 1
S_2	disalarmed / 0 /	0 1 0
S_3	armed	1 0 0
S_4	armed / 0	1 0 1
S_5	armed / 0 /	1 1 0

3 flip flop

output = Q_2

$Q_2 \quad Q_1 \quad Q_0$

② State diagram



③ State transition table

Q_2	Q_1	Q_0	V	C	Q_2^+	Q_1^+	Q_0^+	A
0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	1	0
0	1	0	1	0	1	0	0	0
1	0	1	1	1	1	1	0	1
0	1	1	1	1	X	X	X	X
1	1	1	0	0	X	X	X	V

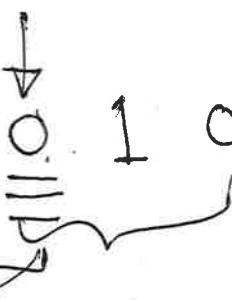
output only depends on the current state

$A = Q_2$

Example : Design a Mealy machine that detects a sequence "0 1 0".

(A) Overlapping is allowed

(B) overlapping is "NOT" allowed.

0 1.  0 1 0 (overlapping is allowed)

0 1 0, 0 1 0 (overlapping is not allowed)

Hint :

<u>State</u>	<u>Def</u>	<u>binary rep</u>
S_0	IDLE	0 0
S_1	get 0	0 1
S_2	get 1	1 0
S_3	unused	1 1

EEE/CSE 120 : Examples

- office hrs 9:30 - 10:15 AM

- Lab office hrs : | M : 5-6 pm
| W : 12-1 pm
| F : 2:30 - 4 pm

- Quiz 2 on Thursday (submission on canvas)

75 minutes

70 min exam

5 min submission.

Example : Design a Mealy Machine that detects
a sequence "010"

- A) Overlapping is allowed
- B) Overlapping is "NOT" allowed

0 1 0 1 0

(overlapping)

0 1 0 0 1 0

(Non-overlapping)

① State def. table

State	Def.	Binary rep.	
S_0	Idle (nothing)	0	0
S_1	get "0"	0	1
S_2	get "1"	1	0
S_3	unused.	1 on Q_1	1 on Q_0

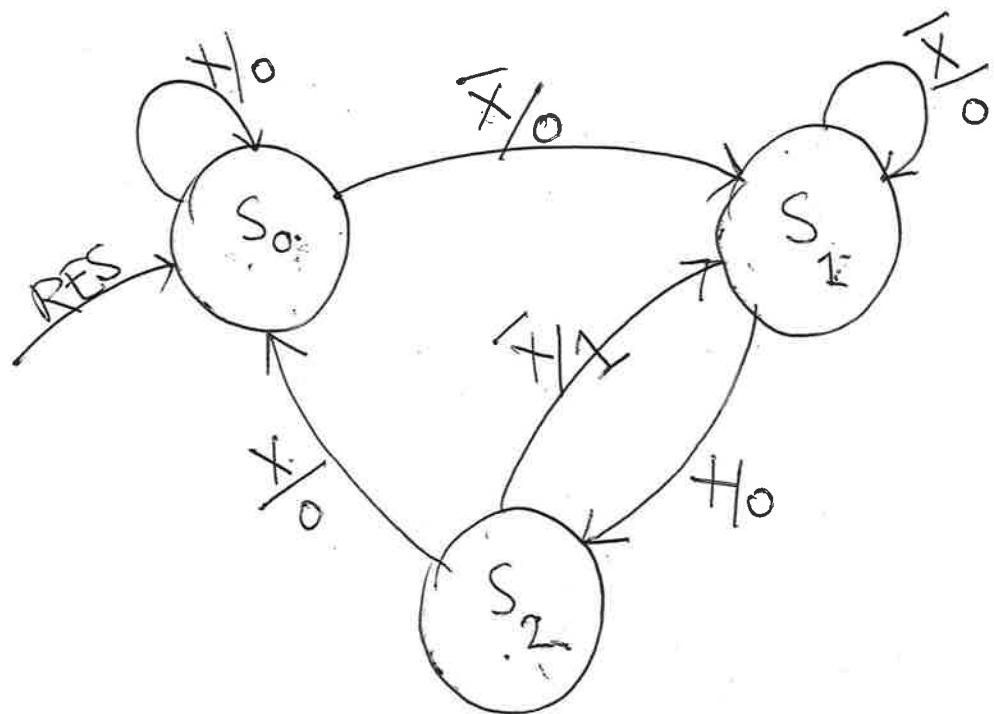
4 States \Rightarrow 2 FFS



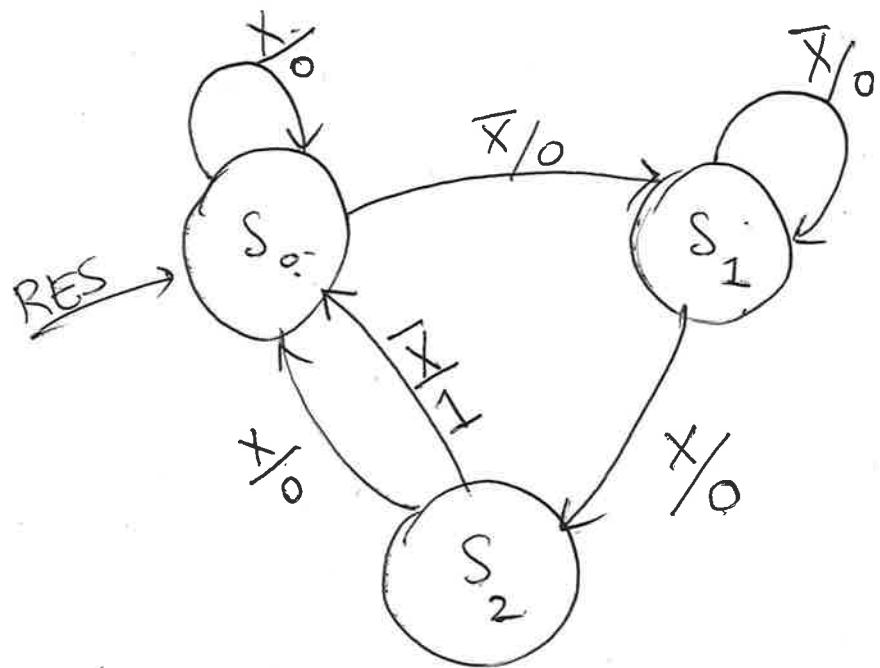
Q_1, Q_0

② Draw State diagram.

overlapping



Non-overlapping



③ State transition table

overlapping

Q_1	Q_0	X	Q_1^+	Q_0^+	output
S_0	0	0	0	1	0
	0	0	0	0	
S_1	0	1	0	1	0
	0	1	1	0	
S_2	1	0	0	1	1
	1	0	0	0	
S_3	1	1	x	x	x
	1	1	x	x	x

sequence
detector

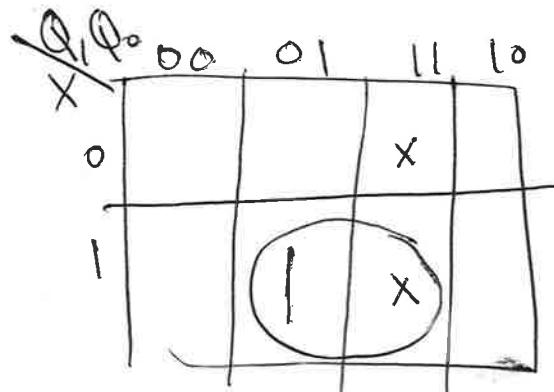
Non-overlapping

Q_1	Q_0	X	Q_1^+	Q_0^+	output
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	1	1	0
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	x	x	x
1	1	1	x	x	x

④ Design the circuit

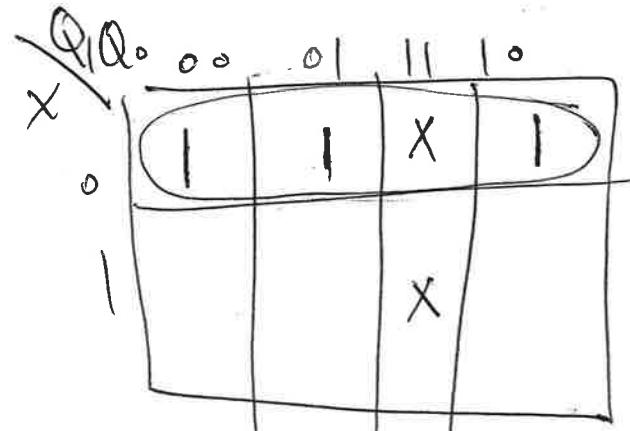
Overlapping :

K-MAP for Q_1^+



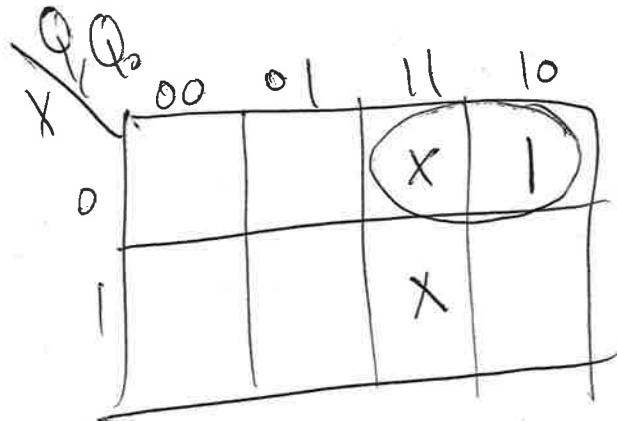
$$Q_1^+ = Q_0 X$$

K-MAP for Q_0^+



$$Q_0^+ = \bar{X}$$

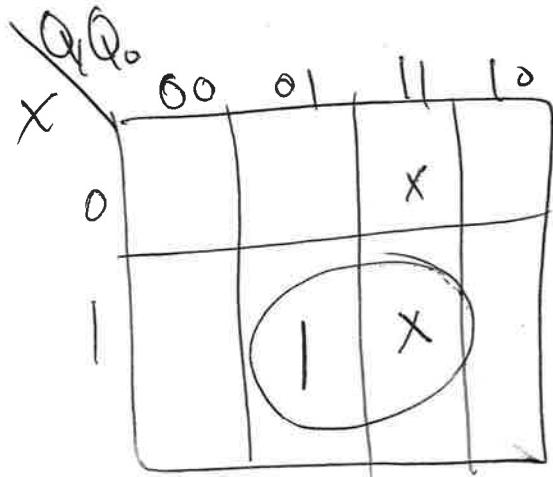
K-MAP for output



$$\text{output} = Q_1 \bar{X}$$

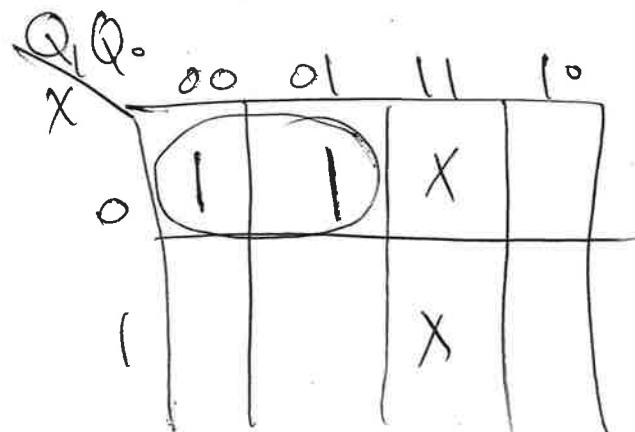
Non-overlapping

K-MAP for Q_1^+



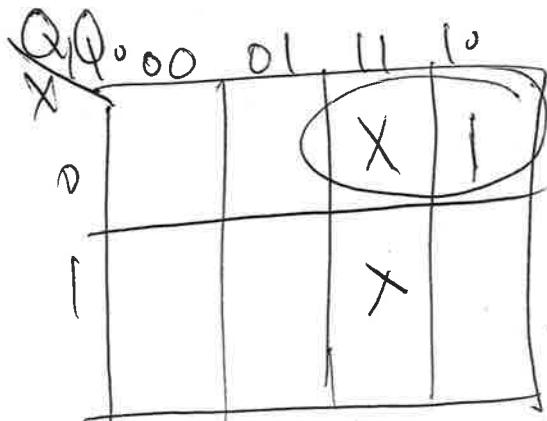
$$Q_1^+ = Q_0 X$$

K-MAP for Q_0^+



$$Q_0^+ = \bar{Q}_1 \bar{X}$$

K-MAP for output



$$\text{Output} = Q_1 \bar{X}$$

overlapping

$$Q_1^+ = Q_0 X$$

$$Q_0^+ = \bar{X}$$

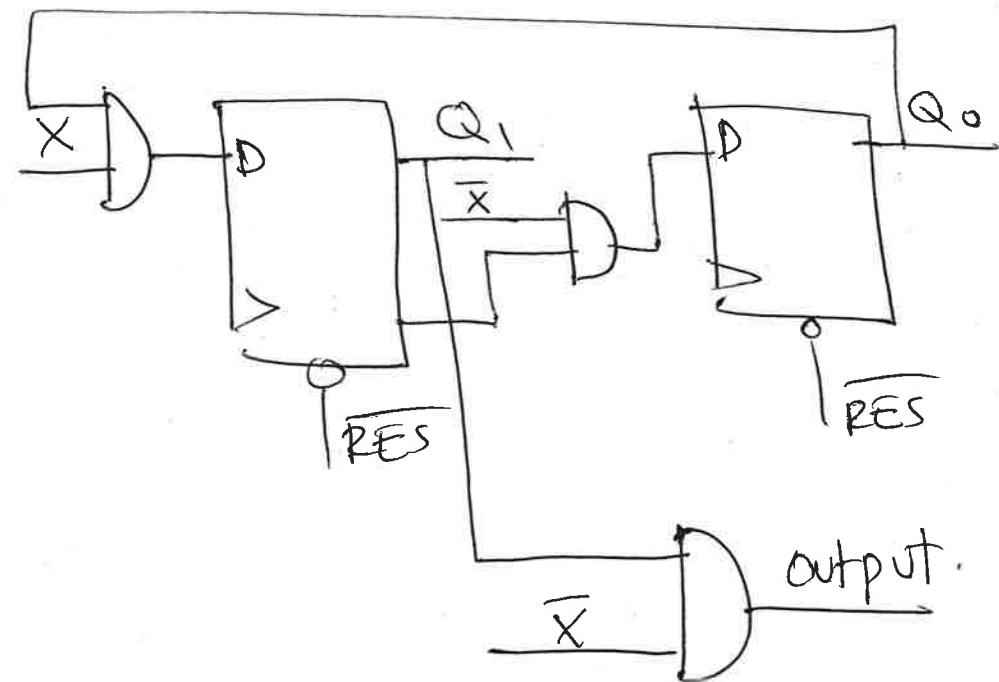
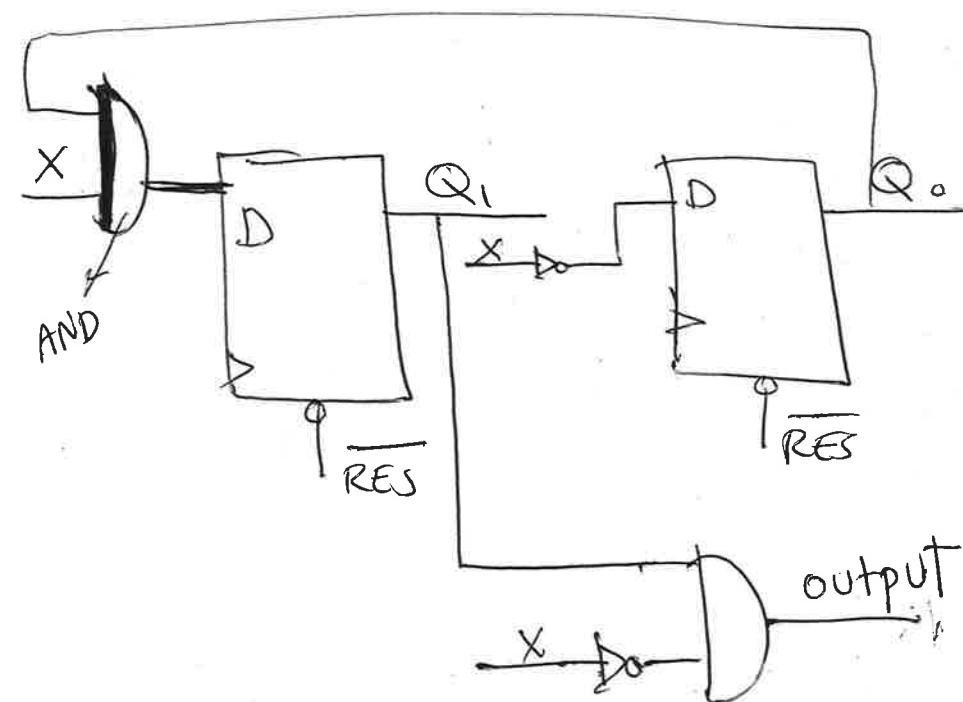
$$\text{output} = Q_1 \bar{X}$$

non-overlapping

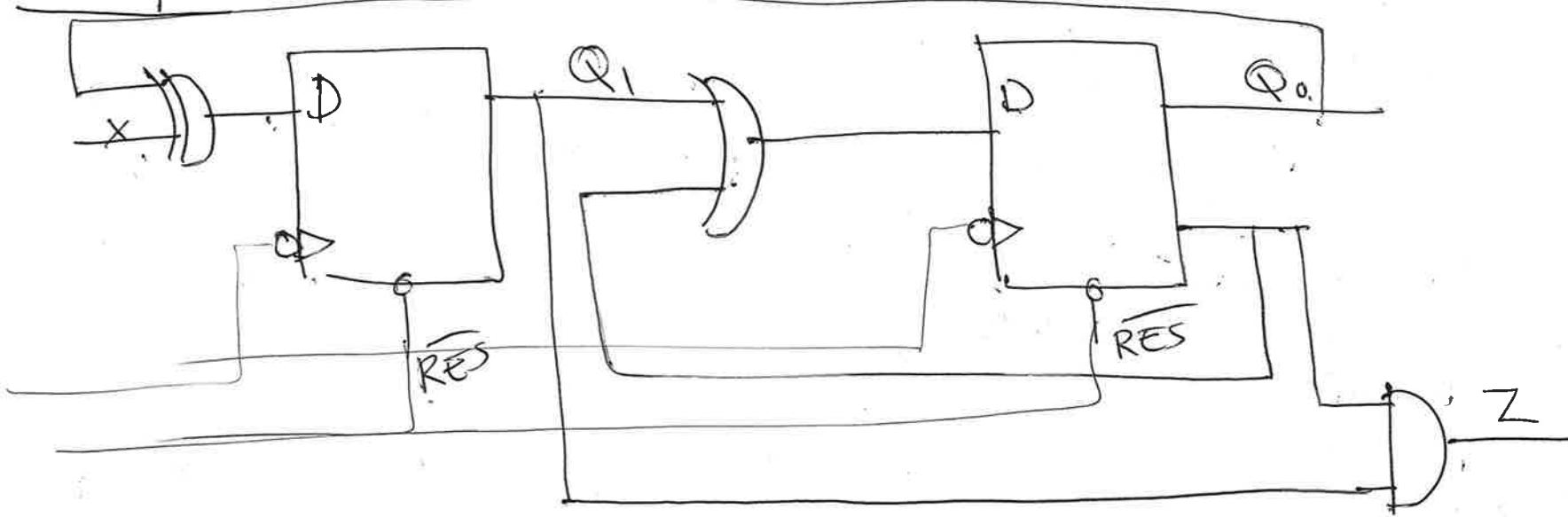
$$Q_1^+ = Q_0 X$$

$$Q_0^+ = \bar{Q}_1 \bar{X}$$

$$\text{output} = Q_1 \bar{X}$$



Example :



① Behavioral eq.

$$Q_1^+ = Q_0 \oplus X$$

$$Q_0^+ = Q_1 + \bar{Q}_0$$

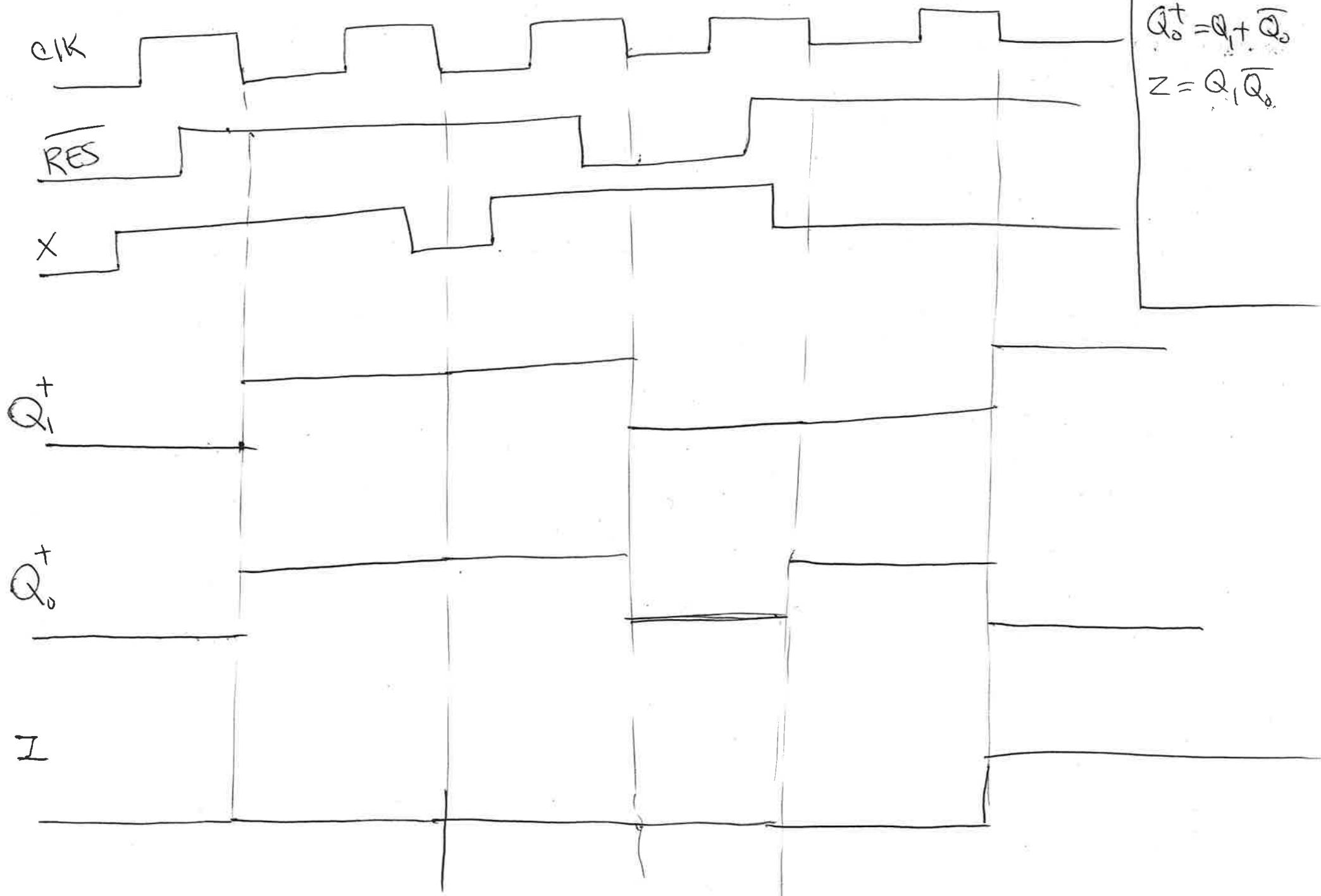
$$Z = Q_1 \bar{Q}_0$$

$$\overline{CTR} = 0 \Rightarrow \text{output} = 0$$

$$\overline{PRE} = 0 \Rightarrow \text{output} = 1$$

② Complete timing diagram.

$$\begin{aligned}Q_1^+ &= Q_0 \oplus X \\Q_0^+ &= Q_1 + \overline{Q}_0 \\Z &= Q_1 \overline{Q}_0\end{aligned}$$



- ① Quiz 3 is next tuesday via zoom / submitted on canvas
Topic: FSMs
- ② Useful Info. for final exam is posted on canvas.
- ③ Office Hours T/TH 9:30 - 10:15 AM.
- ④ Lab 4 is up on canvas and is due on NOV 30.

Substantially different designs

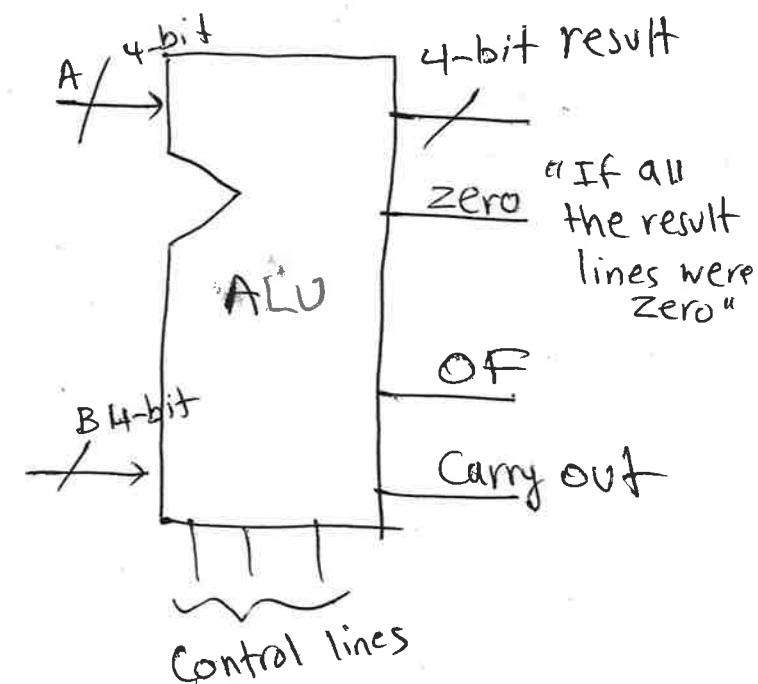
CPU : Central Processing unit

How to build CPU?

① Main part of any CPU is the ALU.

Arithmetic logic unit

- ALU :
- AND
 - OR
 - Summation
 - Subtraction
 - set less than (Comparison)
 - NOR

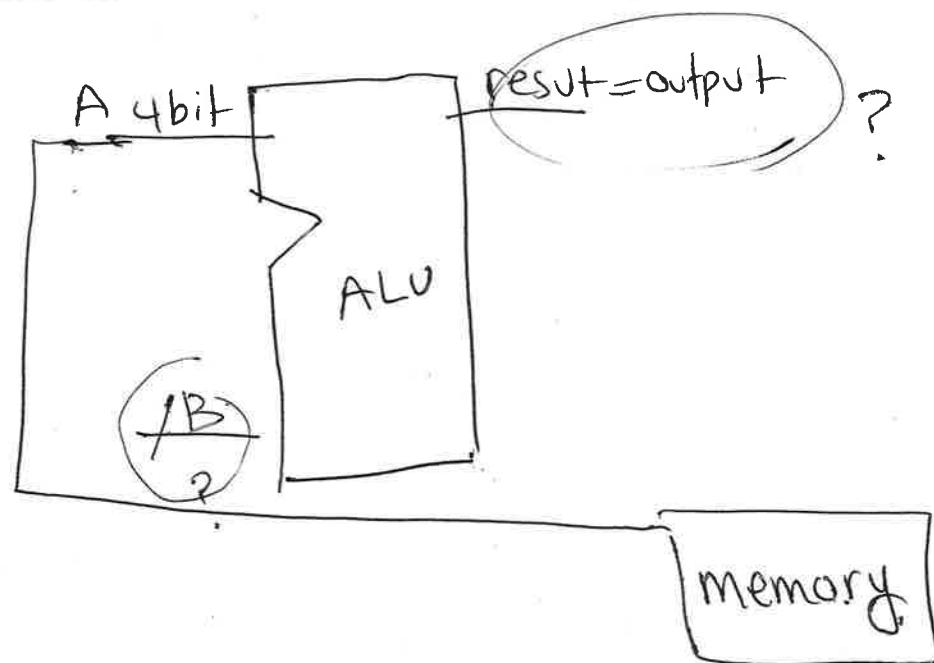


② Memory is needed to build a CPU.

→ - ALU needs data at its input which needs to be stored in the memory

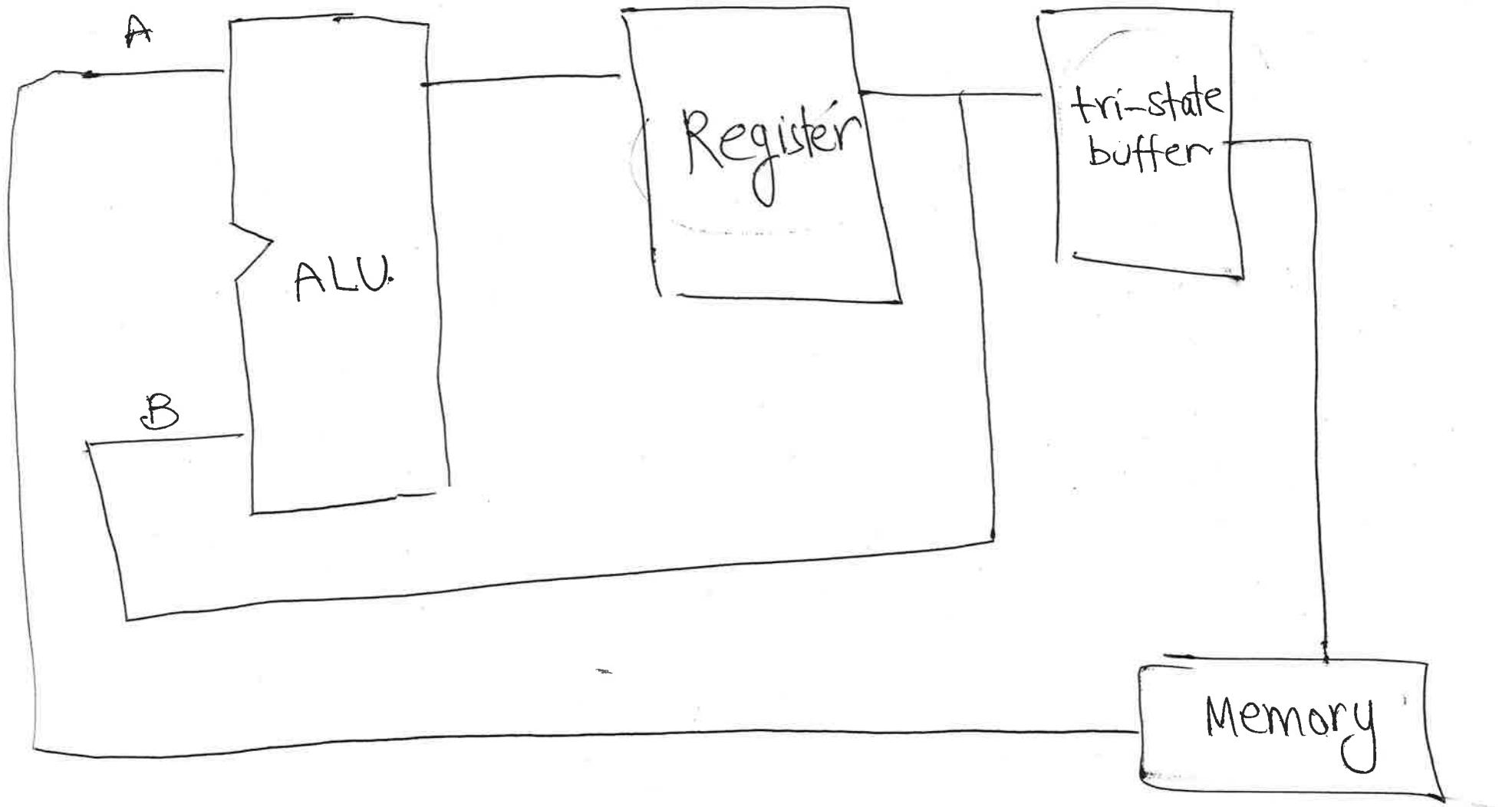
- ALU needs to store its result, specially if it is supposed to be reused.

CPU : ALU + Memory



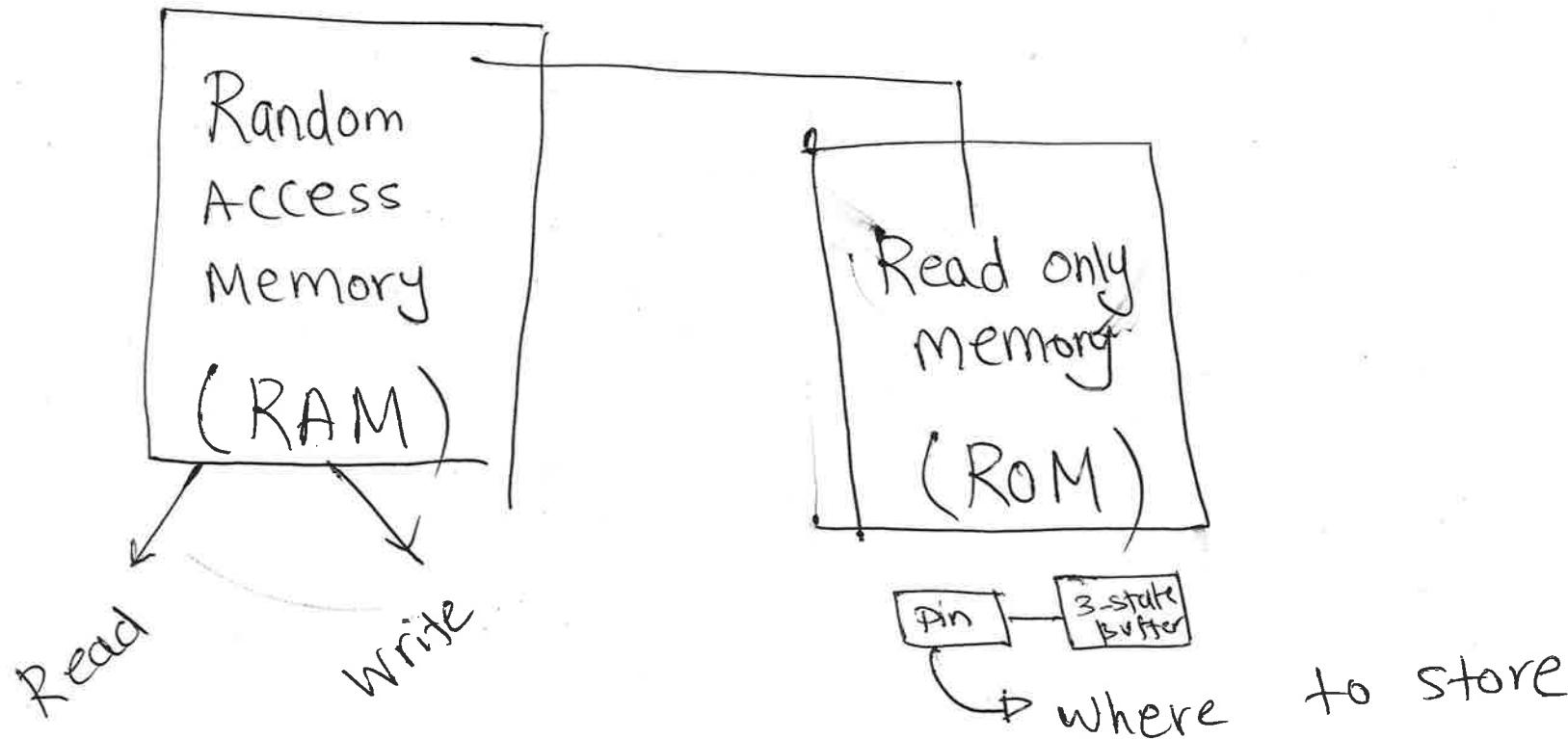
Designing CPU :

- ① # of connections is minimized
- ② Temporary storage (Registers)
- ③ use loop back connection to the input B.
(Multiplexing our connection)
↓
tri-state buffer.

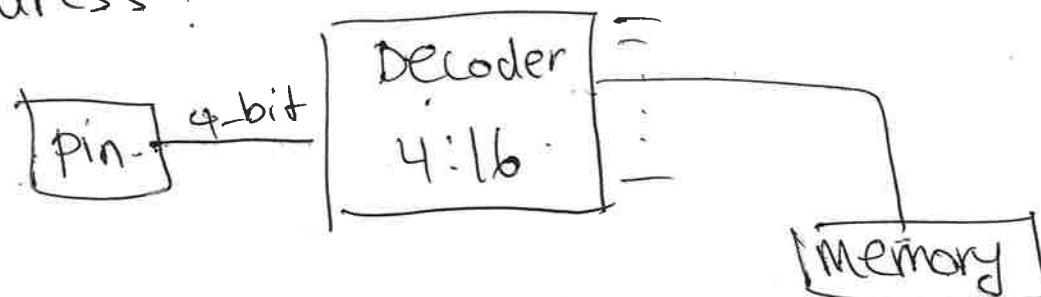


Accumulator.

How to address in a memory?



How to address?



Summary:

- ① We designed a processor using an ALU & a memory!
- ② we used "Accumulator" to store ALU results temporarily
- ③ ISSUE: Need to take care of a lot of control lines and memory storage manually.

Brainless Processor.



Automate

Automating memory

Automating control lines

① Automating memory

we need to have an automatic memory address incrementer.

memory address incrementer

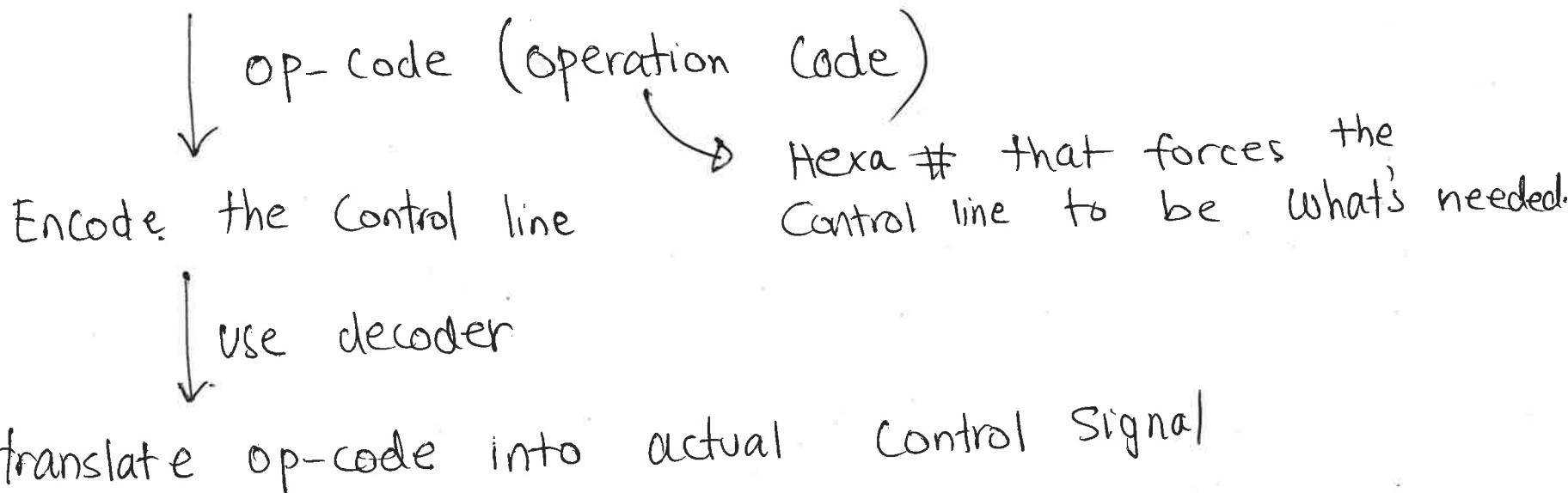
- 1 Count up
- 2 Jump around (write in specific part)
- 3 Stop adder
(not read/write in memory)

Idea: Design a FSM that can

- ① increment
- ② jump around
- ③ stop

② Automating the control lines

— Single bit Control lines depend on the operation we're performing.



Issue : Single op-code may not be enough to perform an operation (need more than cycle)

↓

Design a FSM for Macro-op (μ-op)
→ Mealy Machine

CPU Architecture

Princeton (Von-Neumann)

Princeton response was a computer that had a Common memory for storing programs as well as variables and other data structure.

- use memory interface unit to access memory spaces.

Harvard

Harvard designed a computer that had a separate instruction memory and separate program bus.

- Princeton won the competition because it was better at the time (technology issue)
 - ↳ It had fewer things that could go wrong.
- Harvard was ignored till 1970, then people realized the advantages of Harvard design:
 - ① Faster
 - ② In parallel
 - ③ Fewer instruction cycles compared to Princeton design.

EEE/CSE 120 : Review

- Reminder: Quiz 3 is on Tuesday (Next week)
 - Finite state machines
- office hours 9:30 - 10:15 AM. (T/TH)

① Determine 2's complement

$$\begin{array}{r} 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \\ \quad \uparrow \\ \underline{1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1} \end{array}$$

$$\begin{array}{r} 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \\ \quad \uparrow \\ \underline{1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0} \end{array}$$

Find $(-21)_{10}$

- Find binary rep. of (21)

$$\begin{array}{r} 21 \longdiv{2} \\ \overline{1} \quad \begin{array}{r} 10 \longdiv{2} \\ \overline{0} \quad \begin{array}{r} 5 \longdiv{2} \\ \overline{1} \quad \begin{array}{r} 2 \longdiv{2} \\ \overline{0} \quad \begin{array}{r} 1 \end{array} \end{array} \end{array} \end{array}$$

$$0 \ 1 \ 0 \ 1 \ 0 \ 1$$

↓ 2's Complement

$$\begin{array}{r} 1 \ 1 \ 1 \ 1 \ 1 \\ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \\ + 1 \ 0 \ 1 0 1 1 \\ \hline \square \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \end{array} \quad \leftarrow \text{Verification}$$

$$1 \ 0 \ 1 \ 0 \ 1 \ 1$$

② Add these 2's complement Signed binary numbers

cin

$$\begin{array}{r} + \\ \begin{array}{r} 1 \ 0 \ 0 \ 0 \\ 1 \ 0 \ 1 \ 0 \end{array} \\ \hline \end{array} \rightarrow 8 \quad 0110 \rightsquigarrow +6 \rightsquigarrow -6$$

$$\begin{array}{r} \boxed{1} \\ \text{cout} \\ \hline \begin{array}{r} 0 \ 0 \ 1 \ 0 \\ +2 \end{array} \end{array} \neq \begin{array}{r} -14 \\ OF \end{array}$$

$$C_{in} \neq C_{out} \Rightarrow OF.$$

$$\begin{array}{r} \boxed{0} \\ \text{cout} \\ \hline \begin{array}{r} 1 \ 0 \ 0 \ 1 \\ 0 \ 1 \ 0 \ 1 \end{array} \end{array} \rightarrow (0111) \rightsquigarrow -7$$

$$\begin{array}{r} + \\ \begin{array}{r} 0 \ 1 \ 1 \ 1 \ 0 \\ 5 \end{array} \\ \hline \end{array} \rightarrow -2$$

$$C_{in} = C_{out} \Rightarrow OF \times (0010) \rightsquigarrow +2 \rightsquigarrow -2$$

③ Do Conversion.

- $(203)_{16} \rightarrow$ Decimal ?

$$3 \times 16^0 + 0 \times 16^1 + 2 \times 16^2 = 515$$

- $(186)_{10} \rightarrow$ Hexa ?

$$\begin{array}{r} 186 \\ \text{---} \\ 10 \\ \text{---} \\ 8 \\ \text{---} \\ 0 \end{array}$$

$(BA)_{16}$

- $0(01|1\ 0\ 1\ 0|0\ 0\ 1\ 1|0\ 1\ 1\ 1)_2 \rightarrow$ Hexa ?

$(1\ A\ 37)_{16}$

* $(3\ A\ 0\ 9)_{16}$ \rightsquigarrow Binary

0011
1010 0000 1001

$(0011\ 1010\ 0000\ 1001)_2$

④ Given $f(a,b,c) = \overline{\bar{a}bc + \bar{b}c + \bar{c}}$

* Use Boolean Algebra to show $f(a,b,c) = \overline{abc}$

$$\begin{aligned} f(a,b,c) &= \overbrace{\bar{a}bc + \bar{b}c + \bar{c}}^{\text{1}} \\ &= (\underbrace{\bar{a}b + \bar{b}}_{\text{1}})c + \bar{c} \end{aligned}$$

$$(ab + c = (a+c)(b+c))$$

↑
distributivity

$$= ((\bar{a} + \bar{b})(b + \cancel{b}))c + \bar{c}$$

↓
1

$$= (\bar{a} + \bar{b})c + \bar{c}$$

$$((a+b)c = ac + bc)$$

$$= \bar{a}c + \overbrace{\bar{b}c + \bar{c}}^{\text{1}}$$

$$= \bar{a}c + (\bar{b} + \bar{c})(c + \cancel{c})$$

↑
1

$$= \bar{a}c + \bar{b} + \bar{c}$$

$$= (\bar{a} + \bar{c})(c + \cancel{c}) + \bar{b} = \bar{a} + \bar{b} + \bar{c} = \overline{abc}$$

Demorgan's

* Find Canonical form for SOP.

$$f(a,b,c) = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \\ \bar{a}b\bar{c} + \bar{a}bc + \\ a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c}$$

<u>a</u>	<u>b</u>	<u>c</u>	<u>f(a,b,c) = $\bar{a}\bar{b}\bar{c}$</u>
0 → 0	0	0	1 ← $\bar{a}\bar{b}\bar{c}$
1 → 0	0	1	1 ← $\bar{a}\bar{b}c$
2 → 0	1	0	1 ← $\bar{a}b\bar{c}$
3 → 0	1	1	1 ← $\bar{a}bc$
—	—	—	—
4 → 1	0	0	1 ← $a\bar{b}\bar{c}$
5 → 1	0	1	1 ← $a\bar{b}c$
6 → 1	1	0	1 ← $ab\bar{c}$
7 → 1	1	1	0 ↑ $\bar{a} + \bar{b} + \bar{c}$

* Find Canonical form for POS.

$$f(a,b,c) = (\bar{a} + \bar{b} + \bar{c})$$

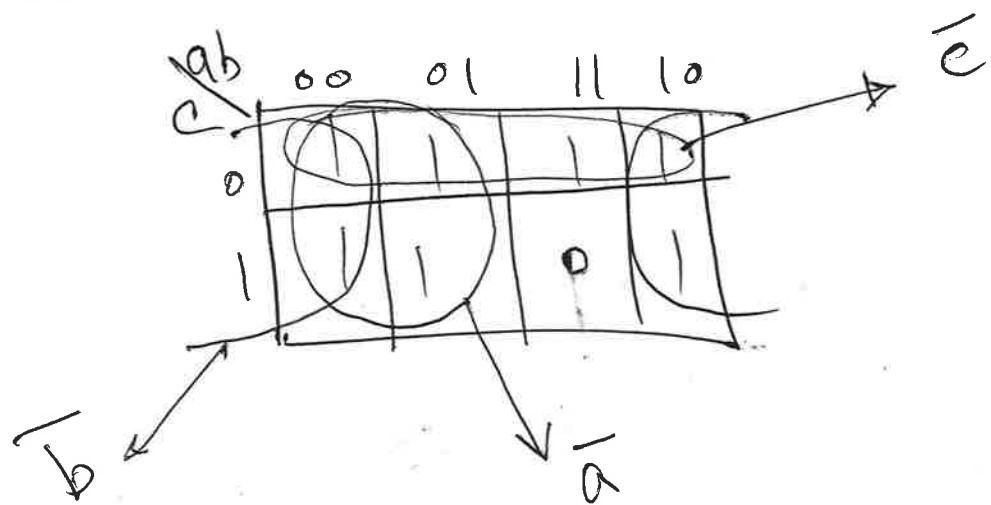
* Find min-term expression

$$f(a,b,c) = \sum m(0,1,2,3,4,5,6)$$

* Find the max-term expression

$$f(a,b,c) = \prod M(7)$$

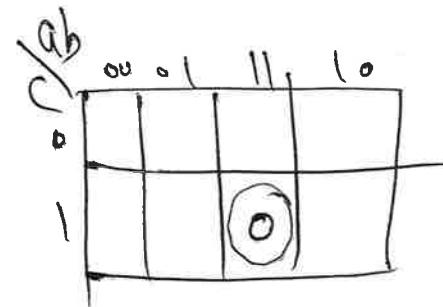
* Find the minimum SOP:



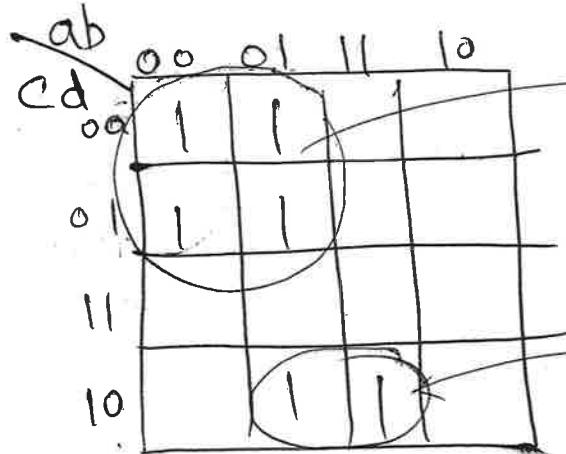
$$f(a,b,c) = \bar{a} + \bar{b} + \bar{c}$$

* Find the minimum product of sum:

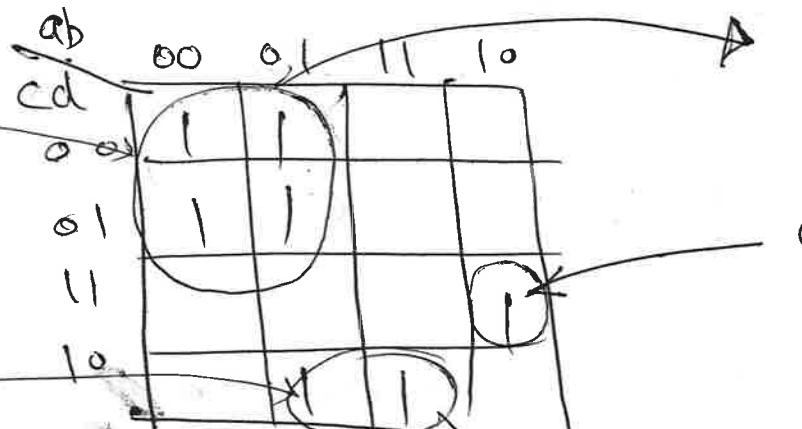
$$f = abc \Rightarrow f = \overline{f} = \overline{abc}$$



⑤ Example:



$$e = 0$$



$$\bar{a}\bar{c}$$

$$\bar{a} \bar{c} d e$$

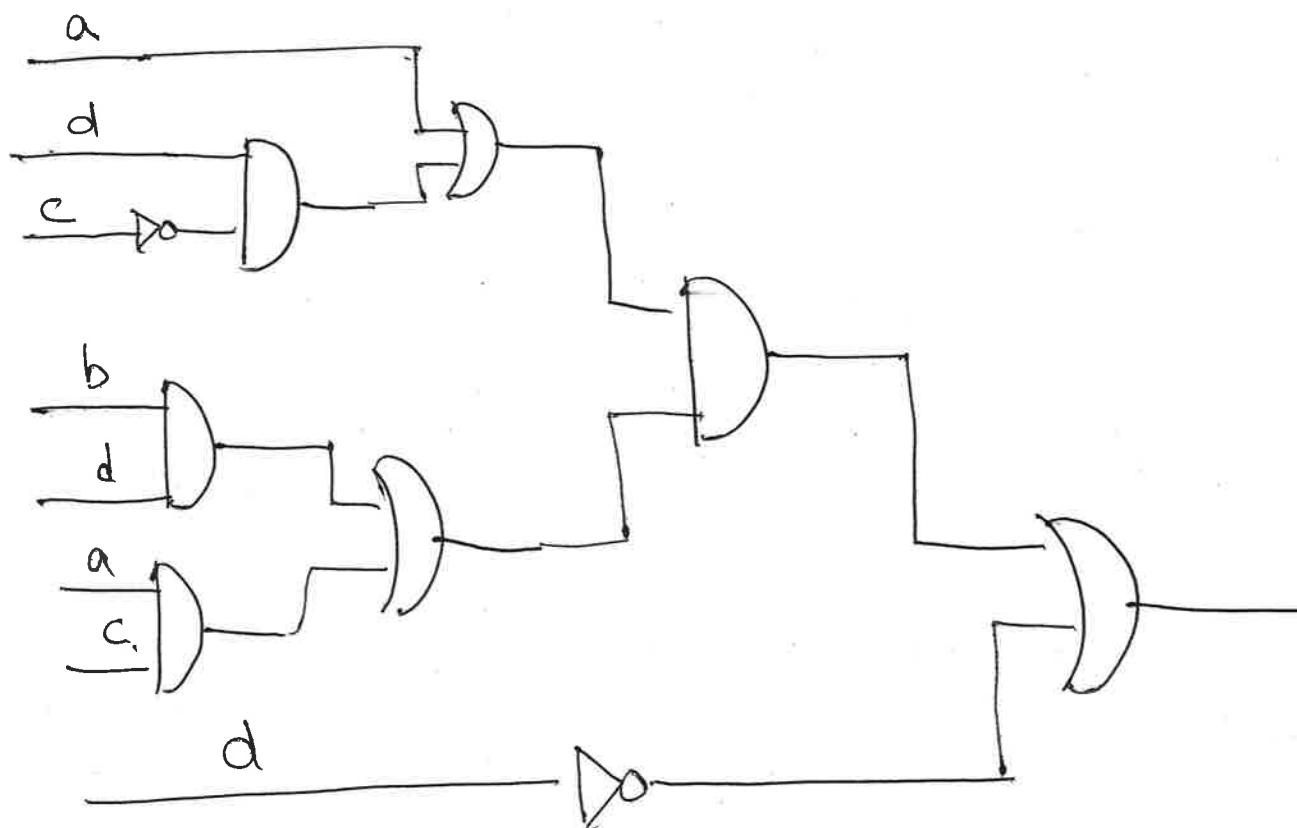
$$e = 1 \rightarrow b\bar{c}\bar{d}$$

$$f(a, b, c, d, e) = ?$$

$$f(a, b, c, d, e) = \bar{a}\bar{c} + b\bar{c}\bar{d} + \bar{a}\bar{b}cde$$

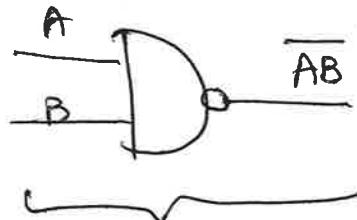
- ⑥ - {
- Design a circuit that represents a function
 - Design a circuit using only "NAND" or "NOR".

$$f(a, b, c, d) = \underbrace{(a + d\bar{c})}_{\text{underbrace}} \underbrace{(b\bar{d} + ac)}_{\text{underbrace}} + \bar{d}$$

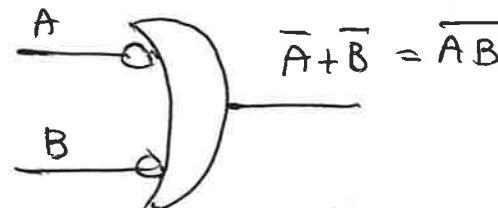


- Build a circuit using "NAND" / "NOR"
 - Build the circuit the way it is
 - Play the bubble game

①

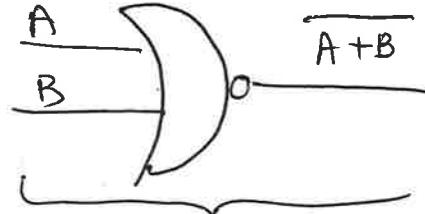


\Leftrightarrow

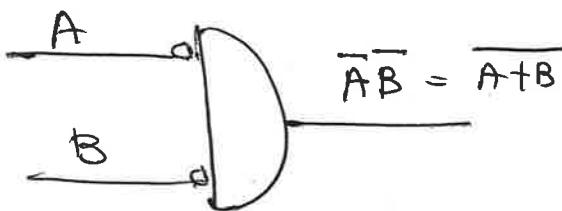


NAND

②



\Leftrightarrow



NOR

EEE/CSE 120 : Review 2 (Last lecture)

■ For the exam :

- 1) Examples on the notes (will be updated shortly)
- 2) Assignments
- 3) All the quizzes
- 4) Midterm
- 5) Examples we've been doing in review lectures

■ 3 double-sided cheat sheet

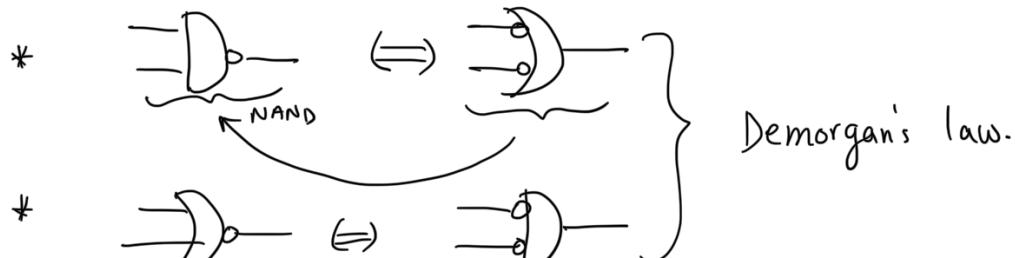
(Cannot have worked-out examples)

- Make sure you're recording \Rightarrow can do it through zoom or any app you'd like
- Make sure you've read the info for final before taking the exam.

- Example 6 :
- Draw a circuit given a function
 - Draw a circuit given a function using only "NAND" / NOR.

* Draw the circuit w/o paying attention to NAND / NOR

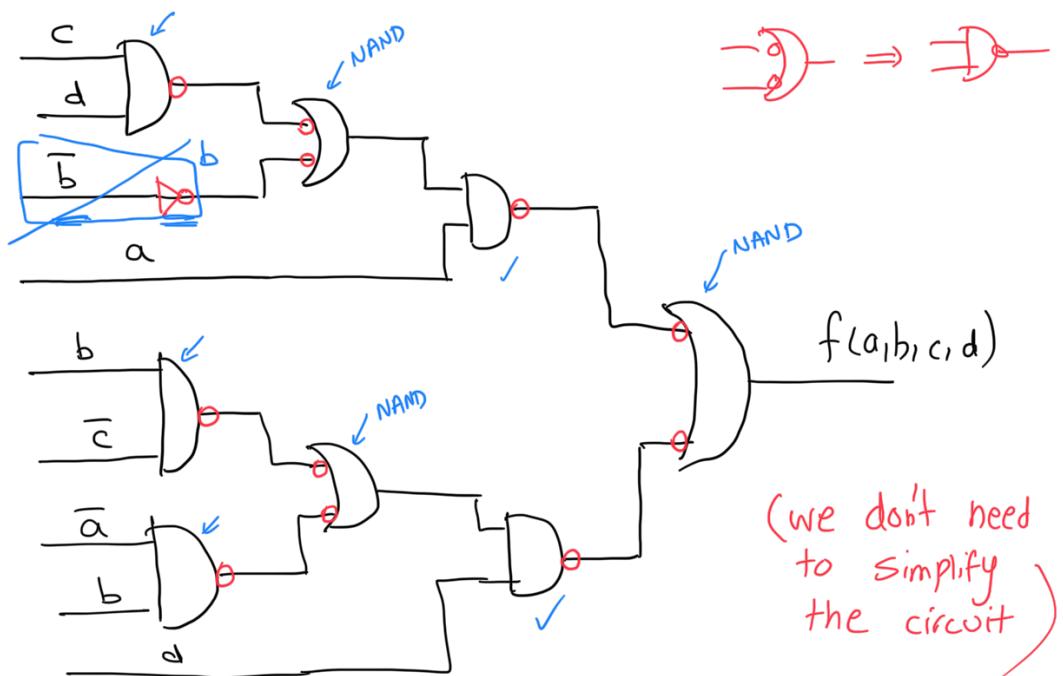
* play the bubble game



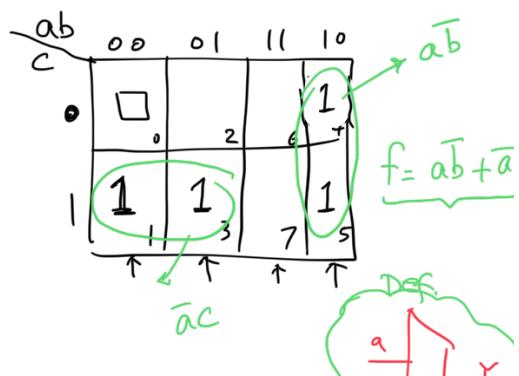
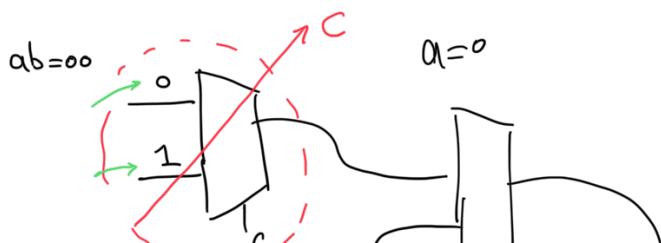


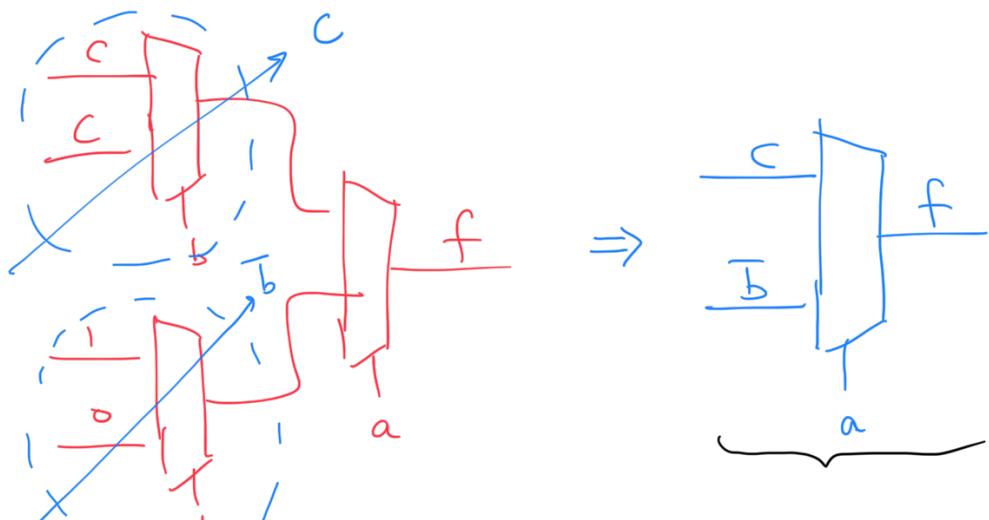
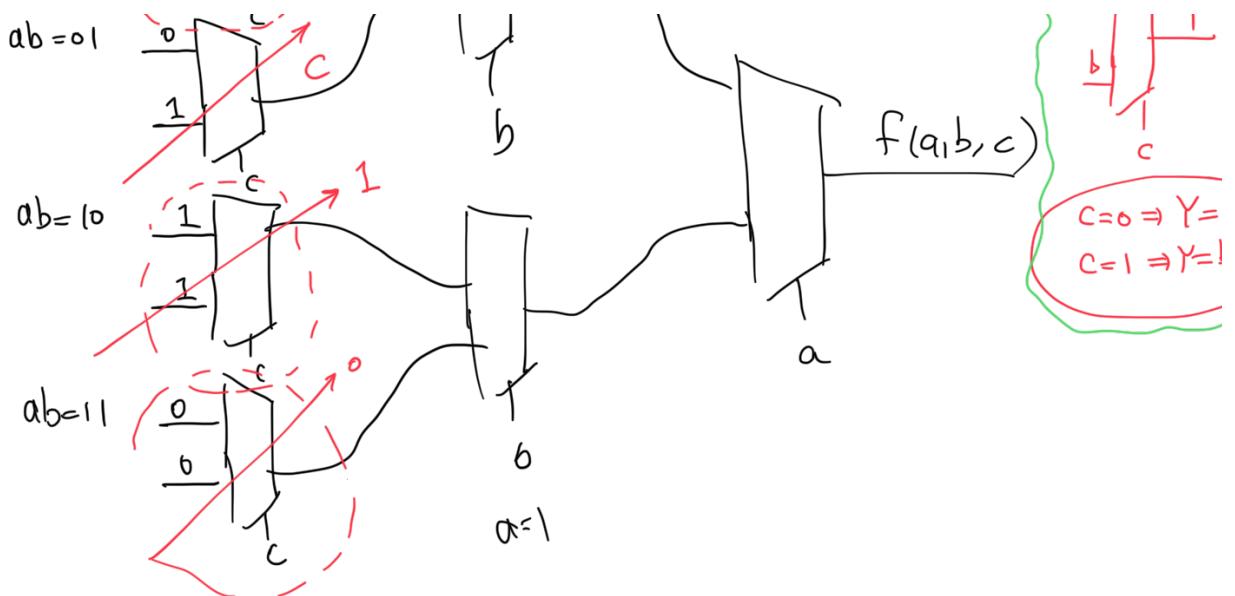
Example 7 : Implement $f(a, b, c, d) = a(c\bar{d} + \bar{b}) + d(\underbrace{b\bar{c}}_{\text{NAND}} + \underbrace{\bar{a}b}_{\text{NAND}})$ using only "NAND" gates.

- Assumption: both Complemented and unComplemented variables are available.



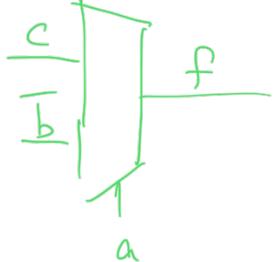
Example 8 : Implement $f(a, b, c) = \sum m(1, 3, 4, 5)$ using 2:1 MUXs.





$$f = \overline{a}\overline{b} + \overline{a}c \Rightarrow \text{eq. for 2:1 mux.}$$

sanity check.

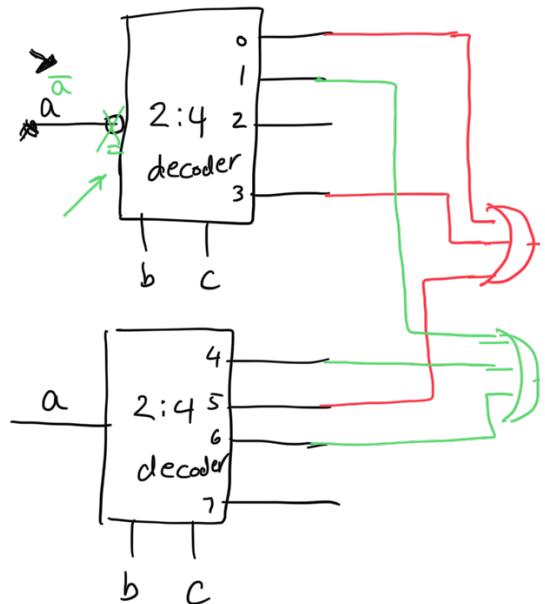


Example 9 : Implement $f(a, b, c) = \sum m(0, 3, 5)$
 $g(a, b, c) = \sum m(1, 4, 6)$

Using two "2:4 decoder" and Two "OR".

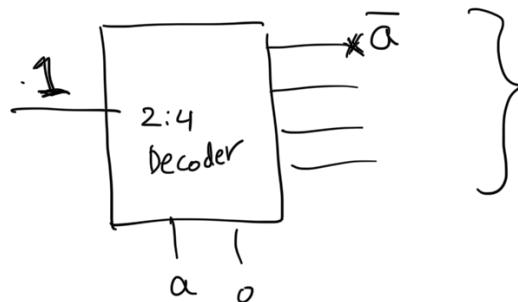
- Assumption:
- we have access to both Complemented and uncomplemented variab
 - use Active high decoders.

#line	input			f	g
	a	b	c		
0	0	0	0	1	0
1	0	0	1	0	1
2	0	1	0	0	0
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	0	1
7	1	1	1	0	0



* If we don't have access to \bar{a} :

we have to use another decoder to build \bar{a} . \rightarrow Replace \bar{a} by another decoder.



Example 10: Definition question:

Implicant: group of 1's which has size of 2^n .

Prime Implicant: is an implicant w/ the largest siz

Essential Prime implicant : is a prime implicant that has at least a single "1" + is not shared between that group and others.

Moore Machine : A FSM where the output doesn't depend on the input directly

Mealy Machine : A FSM where output directly depends on the input

what is the diff. between Moore & Mealy?

- . Moore Machine : output changes should follow the clock depending on the input.
- . Mealy Machine : output changes immediately w/ the input change.
- . Mealy Machine we have timing issue
 \Rightarrow Synchronization FF.

Functionally Complete : A gate is called functionally complete if we could use that gate to build $\{\text{AND}, \text{OR}, \text{NOT}\}$

Example : Is "OR" functionally complete?
 \hookrightarrow Cannot build "NOT" gate

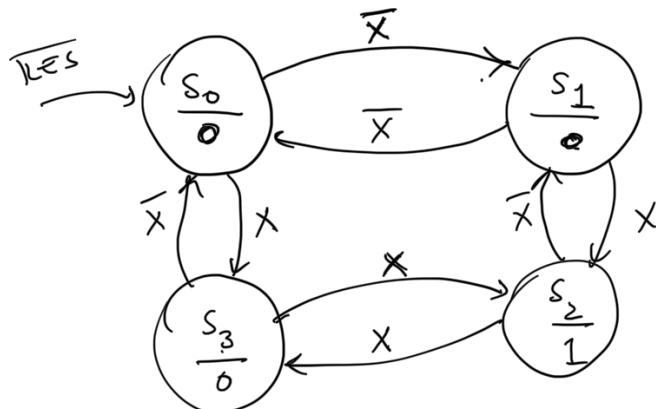
\Rightarrow Not functionally complete

Example: Are "NAND" and "NOR" functionally complete?

both NAND, NOR are functionally complete (Justify)

Example II: Design a FSM (Moore / Mealy)

Example: you're given the following diagram



1) what type machine does this diagram represent?

"Moore"
↓

- 2) How many ffs do we need? $\frac{\# \text{fbs}}{\# \text{states}} = \frac{2}{4} = 2 \text{ ffs}$ the output only depends on the current state or not the next state
- 3) Complete the transition table

	Q_1	Q_0	X	Q_1^+	Q_0^+	Z
S_0	0	0	0	0	1	0
	0	0	1	1	1	0
S_1	0	1	0	0	0	0
	0	1	1	1	0	0
S_2	1	0	0	0	1	1
	1	0	1	1	1	1

Output depends only on current state.

S_3	1	1	0	0	0	0
	1	1	1	1	0	0
				↑	↑	↑

Example: Design a Mealy machine that produces "1" output iff

- input has been 1 for three or more consecutive clock times
- input has been 0 for three or more consecutive clock times

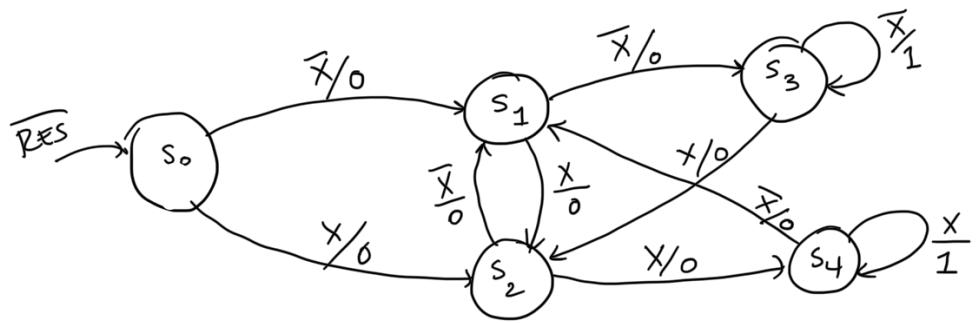
Step 1 : State def. table

state	Def.	binary rep.
S_0	Idle	0 0 0
S_1	Single "0" received	0 0 1
S_2	Single "1" received	0 1 0
S_3	two zeros or more	0 1 1
S_4	two ones or more	1 0 0

$$\# \text{ of ffs} : 2^3 = \underline{\underline{8}} \rightarrow 3 \text{ ffs } \Rightarrow Q_2, Q_1, Q_0$$

Step 2 : Draw the transition diagram

Step 1: draw the transition diagram.



Step 3: Converting transition diagram into transition table.

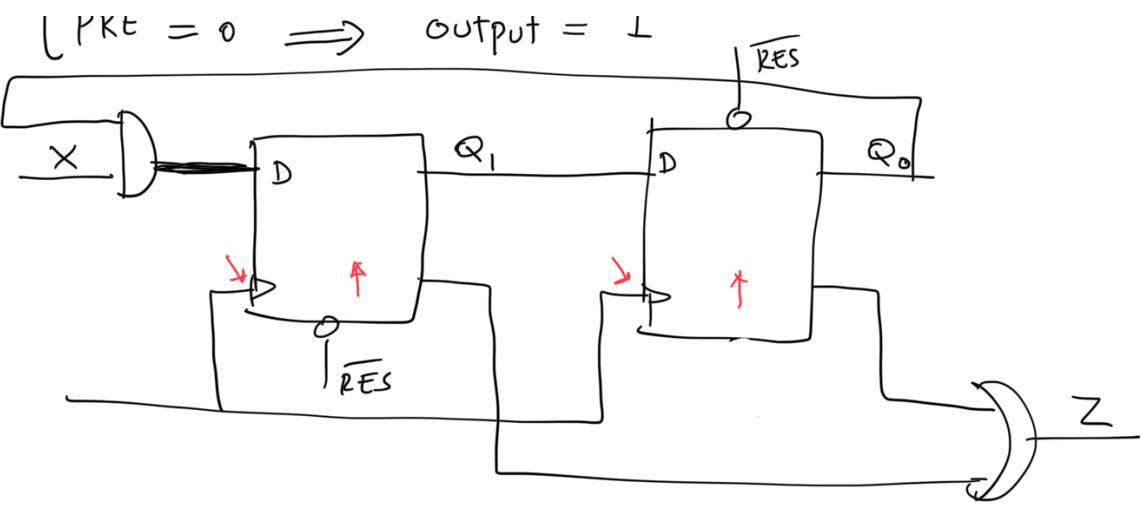
	Q_2	Q_1	Q_0	X	Q_2^+	Q_1^+	Q_0^+	Z
s_0	0	0	0	0	0	0	1	0
	0	0	0	1	0	1	0	0
:								
s_5					x	x	x	x
s_6					x	a	a	a
s_7					a	a	a	a

Example 12 & Timing diagram

→ behavioral eq. → Draw Circuit ✓

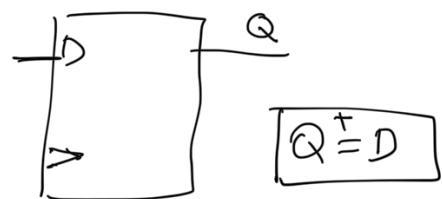
→ Circuit → behavioral eq. → Timing diagram

$$\left\{ \begin{array}{l} \overline{CIR} = 0 \Rightarrow \text{output} = 0 \\ \dots \end{array} \right.$$



① Behavioral eq:

$$\begin{cases} Q_1^+ = Q_0 X \\ Q_0^+ = Q_1 \\ Z = \overline{Q}_1 + \overline{Q}_0 \end{cases}$$



① Determine PE ff & NE ff.

② look at behavioral e

