Lecture 1: Introduction to EEE/CSE 120

Bahman Moraffah
Electrical, Computer, and Energy Engineering
Arizona State University

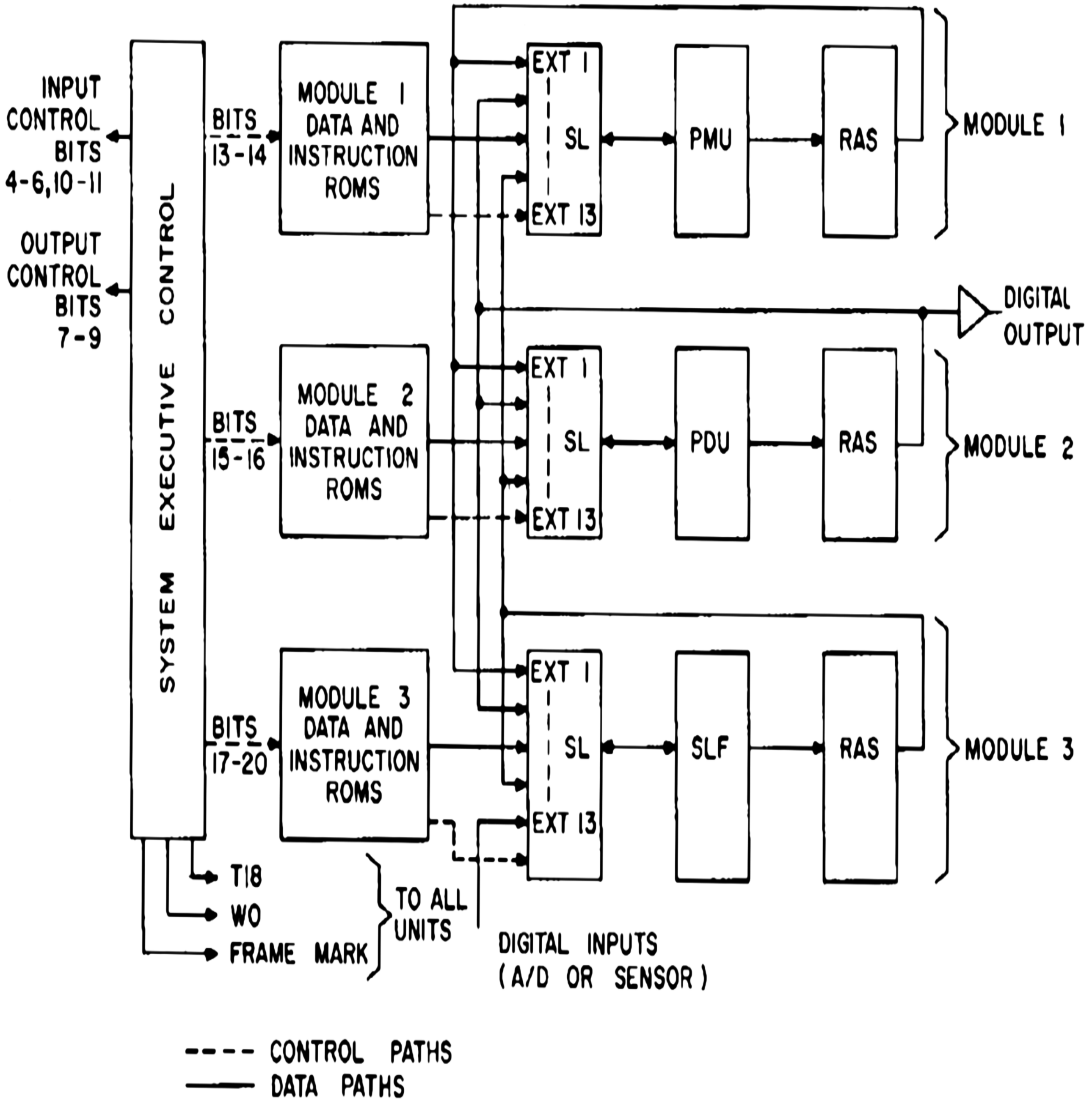
Why do we learn digital circuit analysis

- 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

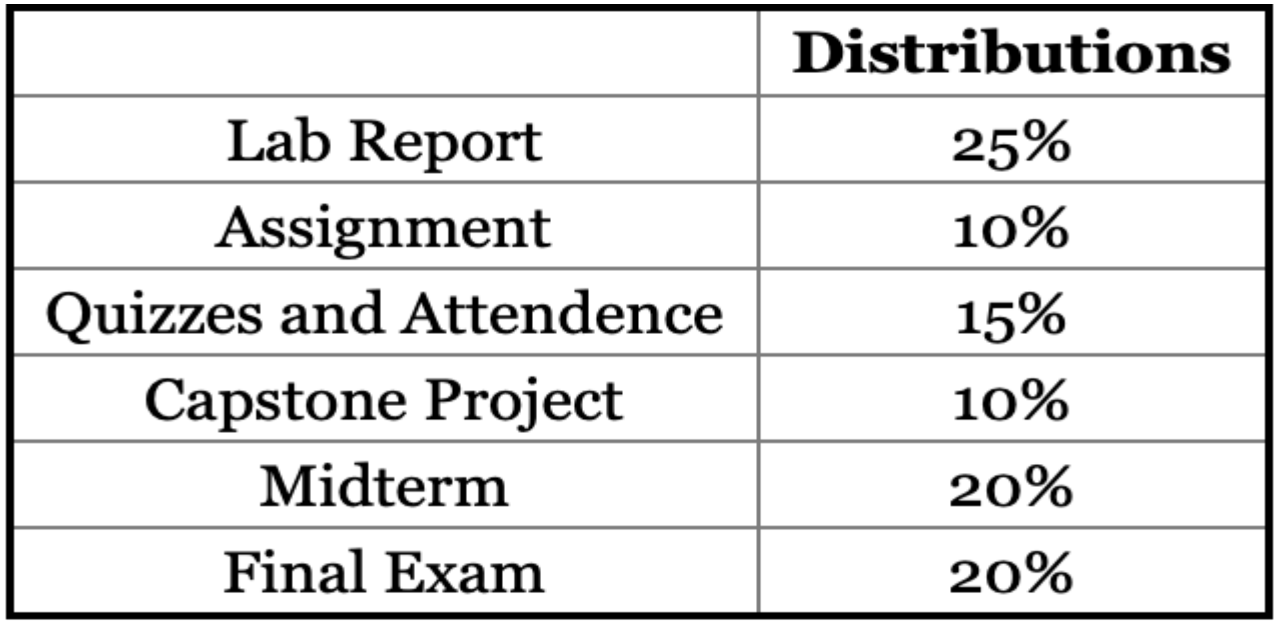
subject to change.

+ Mini quizzes

Exams will be though 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 Attendence	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	Synchrounous 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)

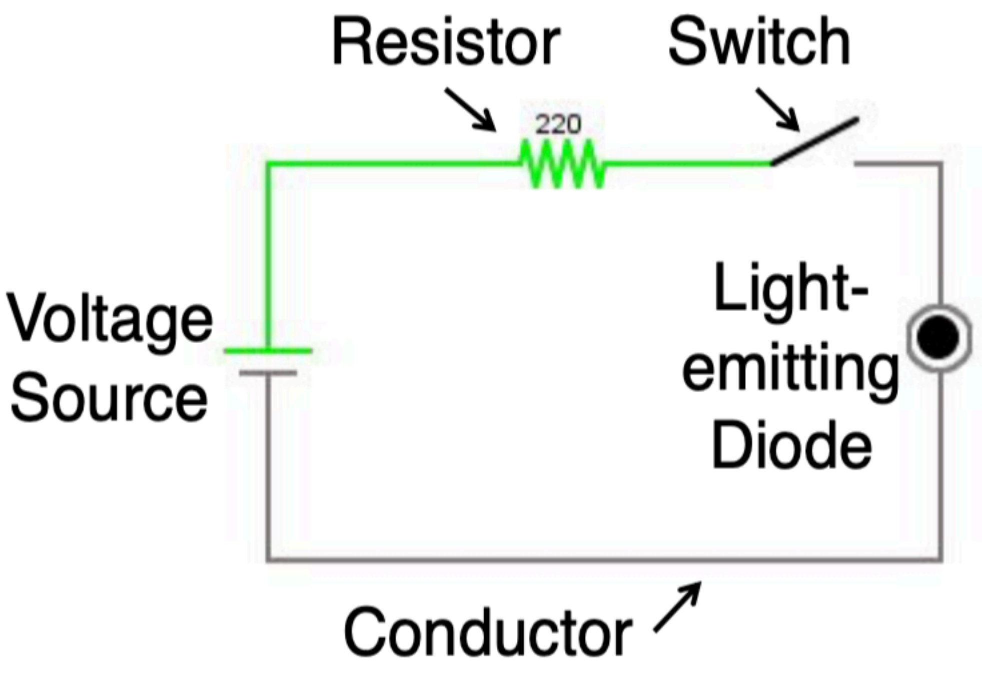
Why digital design

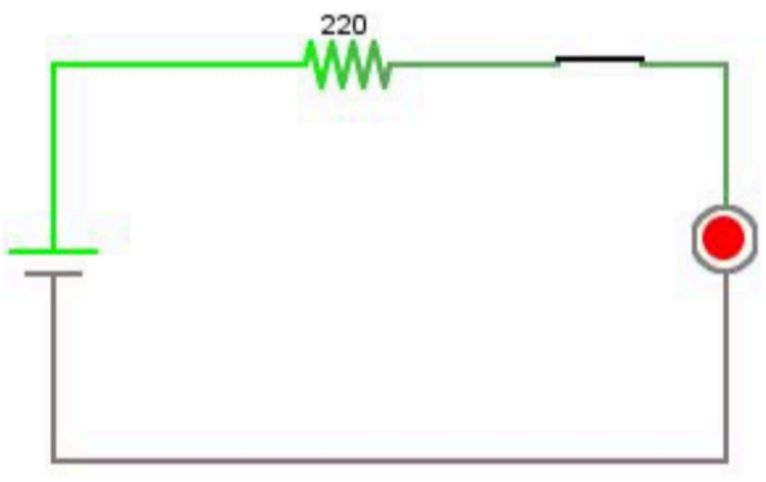
- In 17th century electricity was discovered.
- Built a switch to turn things on and off
    - They were huge switches

- 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

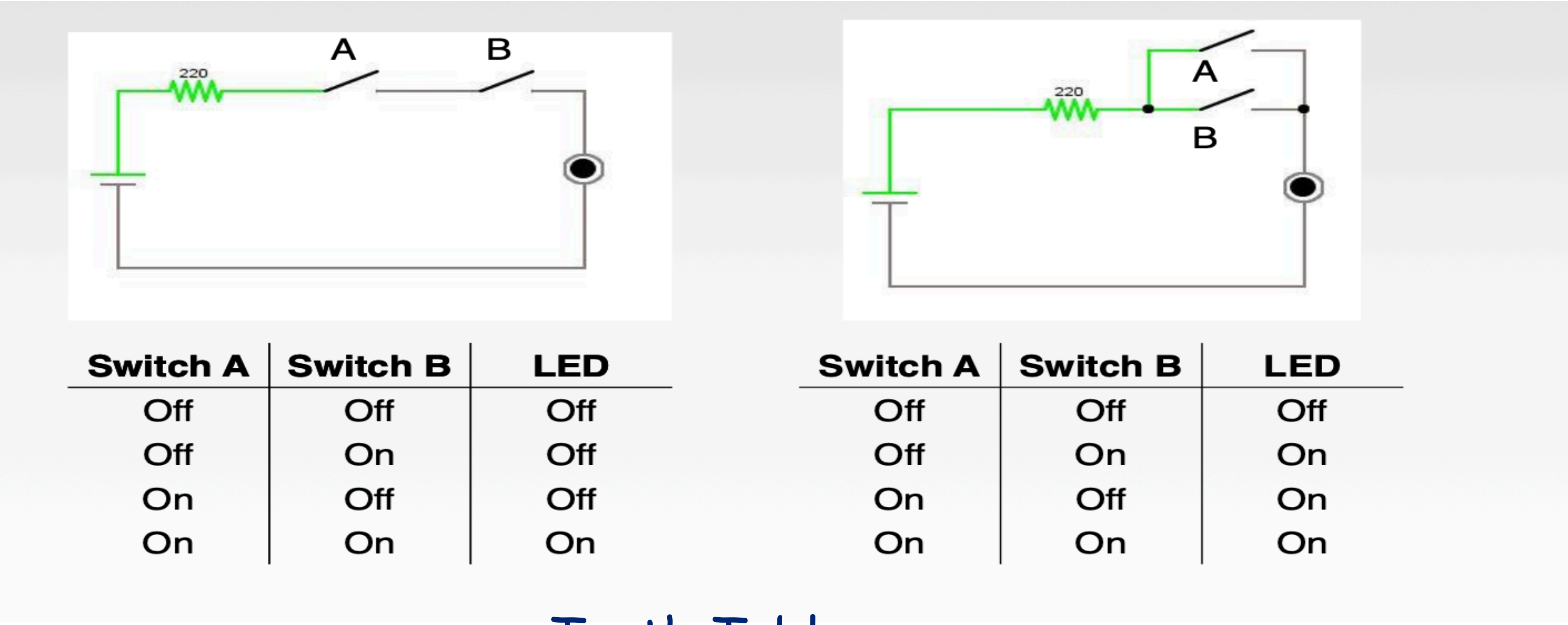
Switches: Connection

❑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

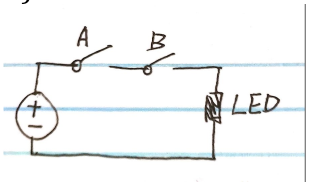
EEE/CSE 120: Digital Design Fundamentals **Fall 2020**

Lecture 2: Gates and Truth Tables

Lecturer: Dr. Bahman Moraffah Scribes: Xinyi Ma

How to connect switches?

1) . series: back to back



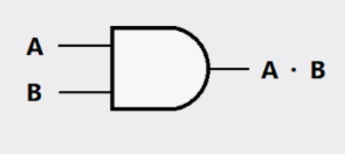
Switch off ⟺ False ⟺ 0
Switch on ⟺ True ⟺ 1

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:

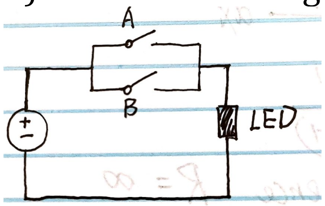
A	B	Y=Output
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table shows all possibilities of outputs.



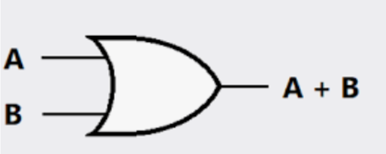
"AND" gate only true when both A and B are true.

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



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

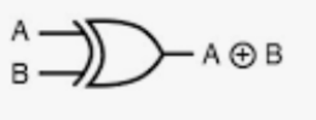
A	B	Y=Output
0	0	0
0	1	1
1	0	1
1	1	1



"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?

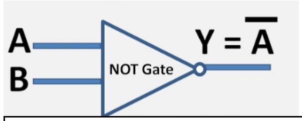
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



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

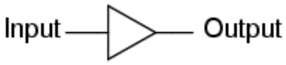
Explanation:

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

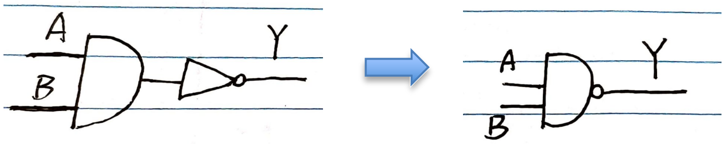
"NOT" gate is one the outputs the opposite state as what is input.



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

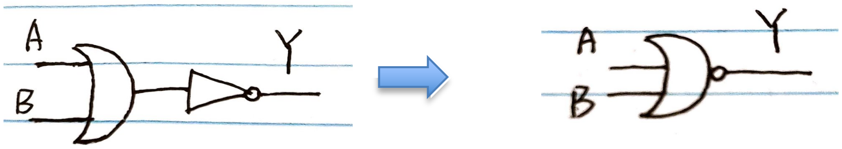
A(Input)	Y(Output)
0	0
1	1

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



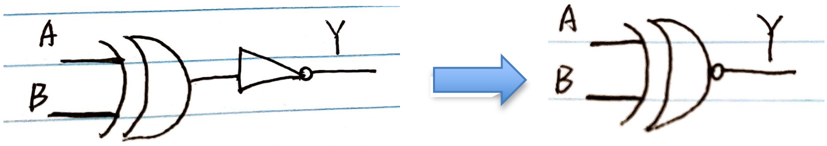
A	B	Y(final output)	C(before"not")
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

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



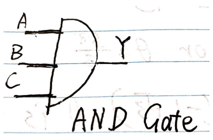
A	B	Y(final output)	C(before "not")
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

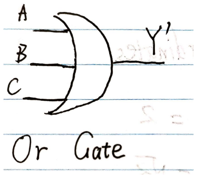
“XOR” gate combines with “NOT” gate:**”XNOR” gate**

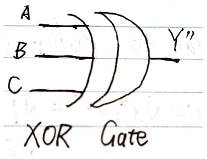


A	B	Y(final output)	C(before”not”)
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0

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 

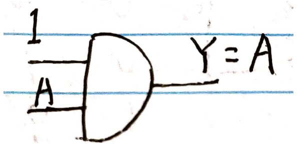
, 

and 

, 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.

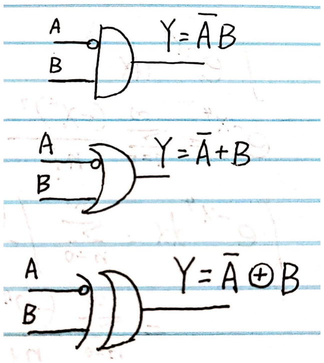


The output Y depends on A:

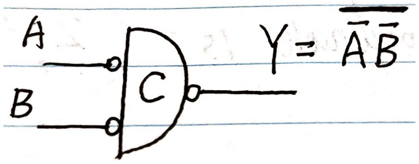
$$\begin{tikzcd} & 0 \arrow[r] & Y=0 \\ A \arrow[ur] \arrow[dr] & & \\ & 1 \arrow[r] & Y=1 \end{tikzcd}$$

so the output Y=A

Example 2.



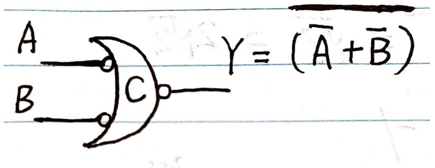
Example 3.



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)

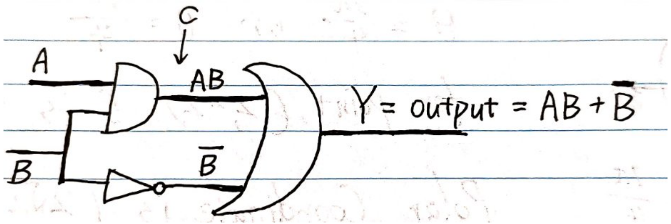
Example 4.



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

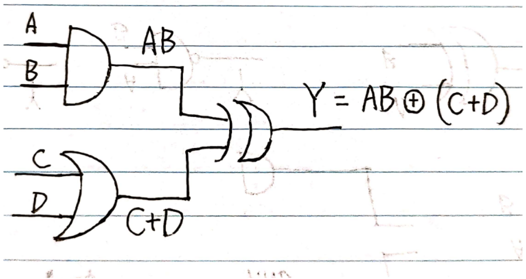
(the final outputs are the same as “AND” gate)

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.

$$(\underset{\underset{10^2}{\uparrow}}{6}\;\underset{\underset{10^1}{\uparrow}}{5}\;\underset{\underset{10^0}{\uparrow}}{4})_{10} = \underbrace{4}_{\text{coeff.}} \times 10^0 + 5 \times 10^1 + 6 \times \underbrace{10}_{\text{base}}^{\overset{\text{exponent}}{\boxed{2}}} = \sum_{i=0}^{n} c_i\, 10^i$$

coeff.

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

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

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

$\hookrightarrow (\dots)_2$

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

Q: 654 in terms of powers of Two?

(Binary rep. of 654)

Binary	Decimal	Binary	Decimal
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		

$$654 = 1\times 2^9 \underset{0\times 2^8}{\wedge} + \underline{\underline{1}}\times 2^7 + 1\times 2^3 + \underline{1}\times 2^2 + 1\times 2^1 + 0\times 2^0$$

$$\left(1\,0\,1\,0\,0\,0\,1\,1\,\underset{\underset{2^1}{\uparrow}}{1}\,\underset{\underset{2^0}{\uparrow}}{0}\right)_2$$

$\underline{\text{base} = 2^3} = 8 \longrightarrow$ octal (oct) $\longleftrightarrow$ $c_i \in \{0, 1, 2, \dots, 7\}$ $(c_i < 8)$

$\underline{\text{base} = 2^4} = 16 \longleftrightarrow$ HEXADECIMAL (HEX) $\longleftrightarrow$ $c_i < 16$

0 – 9

10 ↔ A		12 ↔ C		15 ↔ F	
11 ↔ B		13 ↔ D			
		14 ↔ E			

$$(1\,|\,0\;1\;1\,|\,\underset{2^2}{\underset{\downarrow}{0}}\;\underset{2^1}{\underset{\uparrow}{1}}\;\underset{2^0}{\underset{\uparrow}{1}})_2$$

<u>Example</u>:

$$654 = 0(01\,|\,0\;1\;0\,|\,\underbrace{0\;0\;1}\,|\,\underbrace{1\;1\;0})_2$$

Q: What is the Oct?

$$\underbrace{\quad}_{1} \qquad \underbrace{\underbrace{\quad\quad}\,0\times2^0+1\times2^1+0\times2^2}_{2} \qquad \underset{\downarrow}{}\;1 \qquad \underbrace{0\times2^0+1\times2^1+1\times2^2}_{6}$$

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

EX:

$$654 = 0(\underbrace{0\ 1\ 0}_{1\times 2^1=2} \mid \underbrace{1\ 0\ 0\ 0}_{2^3=8} \mid \underbrace{1\ 1\ 1\ 0}_{0\times 2^0+1\times 2^1+1\times 2^2+1\times 2^3=14})_2$$

HEX ?

E

$$(2\ 8\ E)_{16}$$

Example: $(B\ 3\ \underset{\sim}{6}\ \underset{\sim}{E})_{16}$ binary?

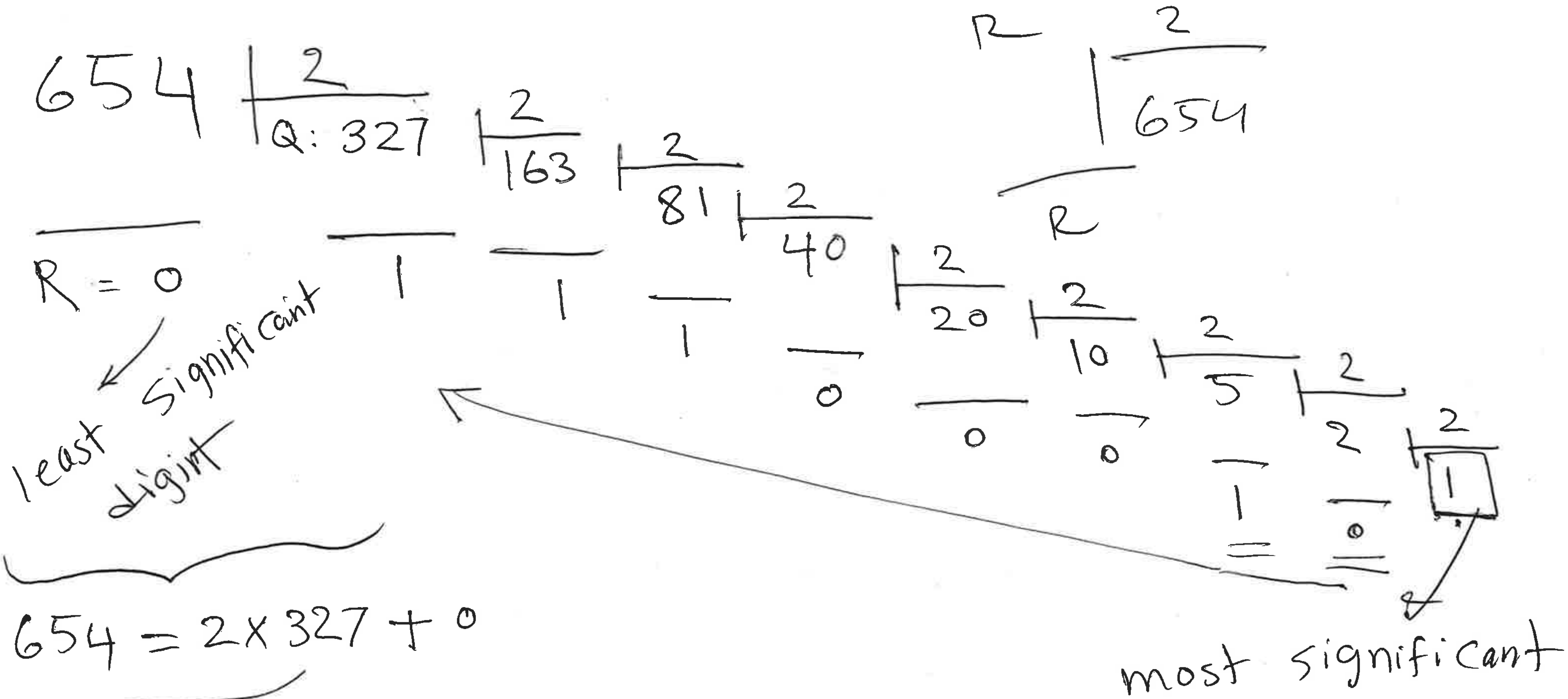
$$E = 14 = \underline{1}\times 2^3 + 1\times 2^2 + 1\times 2^1 + 0\times 2^0 \leftrightarrow (\overset{2^3}{1}\ \overset{2^2}{1}\ \overset{2^1}{1}\ \overset{2^0}{0})$$

$$6 = 1\times 2^2 + 1\times 2^1 \leftrightarrow (0\ 1\ 1\ 0)$$

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

$$B = 11 \longleftrightarrow 1\times2^3 + 1\times2^1 + 1\times2^0 \longleftrightarrow (1\ 0\ 1\ 1)$$

$$(1\ 0\ 1\ 1\quad 0\ 0\ 1\ 1\quad 0\ 1\ 1\ 0\quad 1\ 1\ 1\ 0)_2$$



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

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

$$\vdots$$

$$(1010001110)_2$$

Ex:

$$\begin{array}{r} 5 \\ +^{1}\;\; 7 \\ \hline 12 \end{array}$$

binary:

$$\begin{array}{rcccc} & & 1 & & \\ & 1 & 0 & 1 & 1 \\ + & 0 & 0 & 1 & 0 \\ \hline & 1 & 1 & 0 & 1 \end{array} \qquad \begin{array}{l} \longrightarrow 1\times2^0+1\times2^1+0\times2^2+1\times2^3=11 \\ \longrightarrow \quad 2 \qquad + \\ \hline \quad 13 \end{array}$$

$$\underbrace{1\times2^0}_{1}+0\times2^1+\underbrace{1\times2^2}_{4}+\underbrace{1\times2^3}_{8}=13$$

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

How do we define negative numbers ?

$\boxed{1}$ Signed bit.

negative ← 1 (0 0 1) → -1

positive ← 0 0 0 1 → $+1$

"0" → 1 0 0 0
0 0 0 0 } Two rep.

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

$\boxed{2}$ offsetting

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

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

$$\begin{array}{ccccc} \rightarrow & \boxed{0} & 1 & 1 & 1 \\ \nwarrow & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} \\ \hline & 1 & 1 & 1 & 1 \end{array}$$

$\boxed{3}$ 2's Complement.

$$\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \hline \boxed{1} & 0 & 0 & 1 \end{array}$$

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

$\boxed{1}$ Find a number which if added to the original number → 1 1 1

$\boxed{2}$ Add 1 to the number → 0 0 1

Lecture 4:

~~EEE~~/CSE 120 : 2's complement and it's Arithmatic

- HW 1 is due sep 3
- Lab0 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

Defening negative numbers:

$\boxed{1}$ Sign bit

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

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

Problems: 1) "0" has Two rep.

2) $1+(-1) \neq 0$

$\boxed{2}$ offsetting

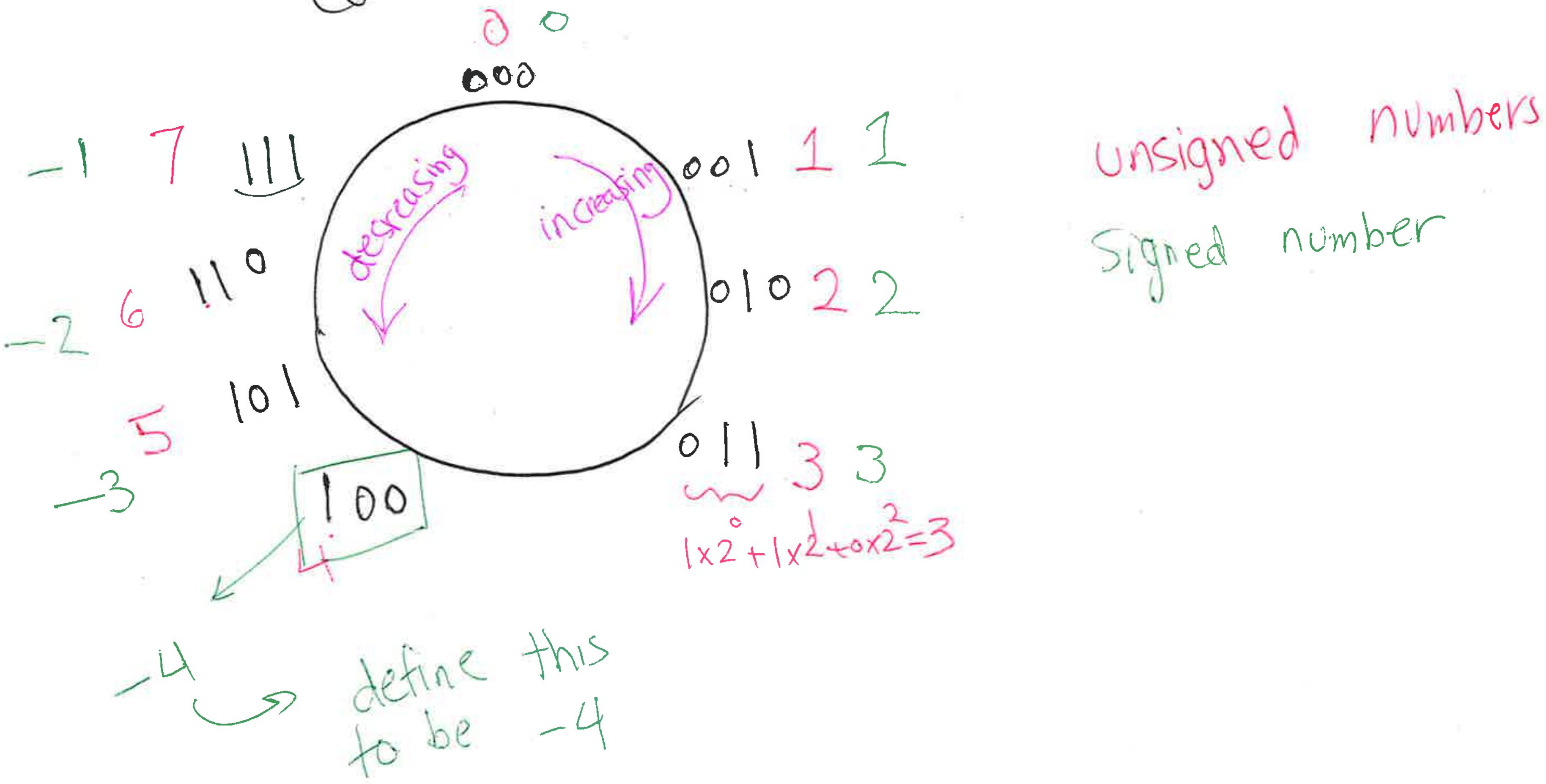
problem: $1+(-1) \neq 0$

$\boxed{3}$ 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.

$$\begin{array}{r} (+3)_{10} \\ +\ (-3)_{10} \\ \hline (0)_{10} \end{array} \longleftrightarrow \begin{array}{ccccc} & (\overset{1}{0} & \overset{1}{1} & 1)_2 & \\ & \boxed{1} & \boxed{0} & \boxed{1} & \longrightarrow (-3)_{10} \\ \boxed{1} & & & & \\ \hline \cancel{1} & 0 & 0 & 0 & \end{array}$$

ignore carry out



How to get a negative number from a positive number?

__Rule__: $\boxed{1}$ Find the number which added to the original number that results in 111

$\boxed{2}$ add 1 to that number

__Example__:

$$\begin{array}{ccccc} & & 1 & \boxed{1} & \\ & & 0 & 1 & 1 \\ + & \boxed{1} & \boxed{1} & \boxed{0} & \boxed{1} \\ \hline & \boxed{1} & 0 & 0 & 0 \end{array} = 111 + 001$$

$$\begin{array}{cccc} & 1 & 1 & \\ & 1 & 1 & 1 \\ & 0 & 0 & 1 \\ \hline \boxed{1} & 0 & 0 & 0 \end{array}$$

Another way of writhing the rule :

$\boxed{1}$ Flip all the bits of the original number (one's complement)

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

$\boxed{2}$ Add 1 to it.

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

$1\times2^0+1\times2^1=3$

① Flipping the bits : 1 0 0

② Add 1 to it

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

Example: 2's Complement of a negative number!

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

① flipping bits → $(0\ 1\ 0)_2$

② Add 1 to it

$$\begin{array}{r@{\quad}ccc} & 0 & 1 & 0 \\ + & 0 & 0 & 1 \\ \hline & \boxed{0} & 1 & 1 \end{array}$$

$$\underbrace{\boxed{0}\ 1\ 1}_{1\times 2^0 + 1\times 2^1 = 3}$$

Example: 2's Complement of 000 ?

① flipping the bits → $1\ 1\ 1$

② To add 1 to it →

$$\begin{array}{r@{\quad}cccc} & & \boxed{1} & \boxed{1} & \\ & & 1 & 1 & 1 \\ + & & 0 & 0 & 1 \\ \hline & \cancel{\boxed{1}} & 0 & 0 & 0 \end{array}$$

Example: $\boxed{1}\ 0\ 1 \quad (-3)_{10}$

① $0\ 1\ 0$

②
$$\begin{array}{r} 0\ 1\ 0 \\ 0\ 0\ 1 \\ \hline 0\ 1\ 1 \end{array} \longrightarrow +3$$

Example: $\underbrace{1\ 0\ 0\ 0}_{(-8)_{10}}$ (4 bit)

2's Compl.: ① $0\ 1\ 1\ 1$

②
$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ \ } \\ 0\ 1\ 1\ 1 \\ 0\ 0\ 0\ 1 \\ \hline 1\ 0\ 0\ 0 \end{array} \longrightarrow +8$$

Example : 0 1 1 0

$\boxed{1}$ Flip all the bits :

$\boxed{2}$ to add 1 to it

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

Quick way of Computing 2's Complement :

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

most sign. $\underbrace{0 \quad 1}_{\text{flip}} \quad \underbrace{\boxed{1} \quad 0}_{\text{unchanged}}$ least sig.

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

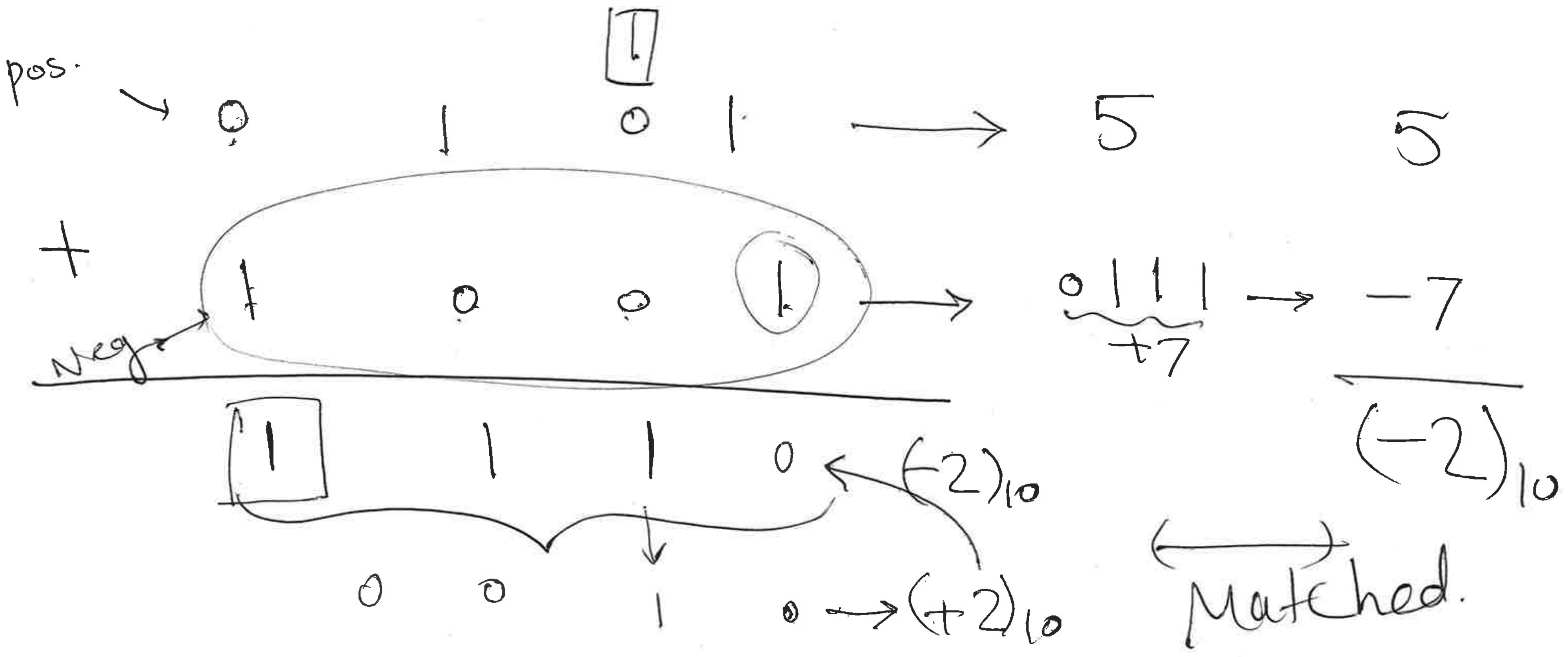
Example

Signed binary

↳ Addition :

① numbers Can be pos. or neg.

② Addition doesn't Care about sign.



Example: Neg $\boxed{0}$ ← carry in

$\boxed{1}\ \ 0\ \ 0\ \ 1 \longrightarrow \underbrace{0\,1\,1\,1}_{7} \longrightarrow (-7)$

$+$

Neg ← $\boxed{1}\ \ 1\ \ 1\ \ 0 \longrightarrow \underbrace{0\,0\,1\,0}_{2} \leftrightarrow (-2)$

carry out ← $\cancel{\boxed{1}}\ \ \underbrace{0\ \ 1\ \ 1\ \ 1}\ \ \neq\ \ -9$

Don't match

overflow.

pos Neg

pos Neg

we might have OF.

pos Neg

Neg post

No OF.

when we look at the most sign. bit

if carry in ≠ Carry out ⟹ OF.

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

$\boxed{0}$

1 0 0 0 → −8

+ 0 1 0 1 → 5

$\boxed{0}$ $\underbrace{1\ 1\ 0\ 1}_{(-3)_{10}}$ $(-3)_{10}$ ✓

Carry in = Carry out

0 0 1 1 → −3

$\boxed{1}$ → carryin

0 1 0 0 → +4

+ 0 1 1 0 +6

$\boxed{0}$ $\underbrace{1\ 0\ 1\ 0}$ $\cancel{(10)_{10}}$

↓ carryout

Carry out ≠ carry in

OF.

$\underbrace{0110}_{+6}$

1010 = −6

1 1 1 1

+ 1 1 1 0

Carry in

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

$1111 \rightarrow (0001) = 1 \rightarrow (-1)_{10}$

$1110 \rightarrow (0010) = 2 \rightarrow (-2)_{10}$

$(-1)_{10} + (-2)_{10} = (-3)_{10}$

carry out

$\underbrace{1101}$ $\leftarrow (-3)_{10}$ Match $=$ $(-3)_{10}$

Carry in = Carry out $\Longrightarrow$ NO OF.

2's comp $(0\ 0\ 1\ 1) = 3$

Lecture 5:

EEE/CSE 120: Adders:

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

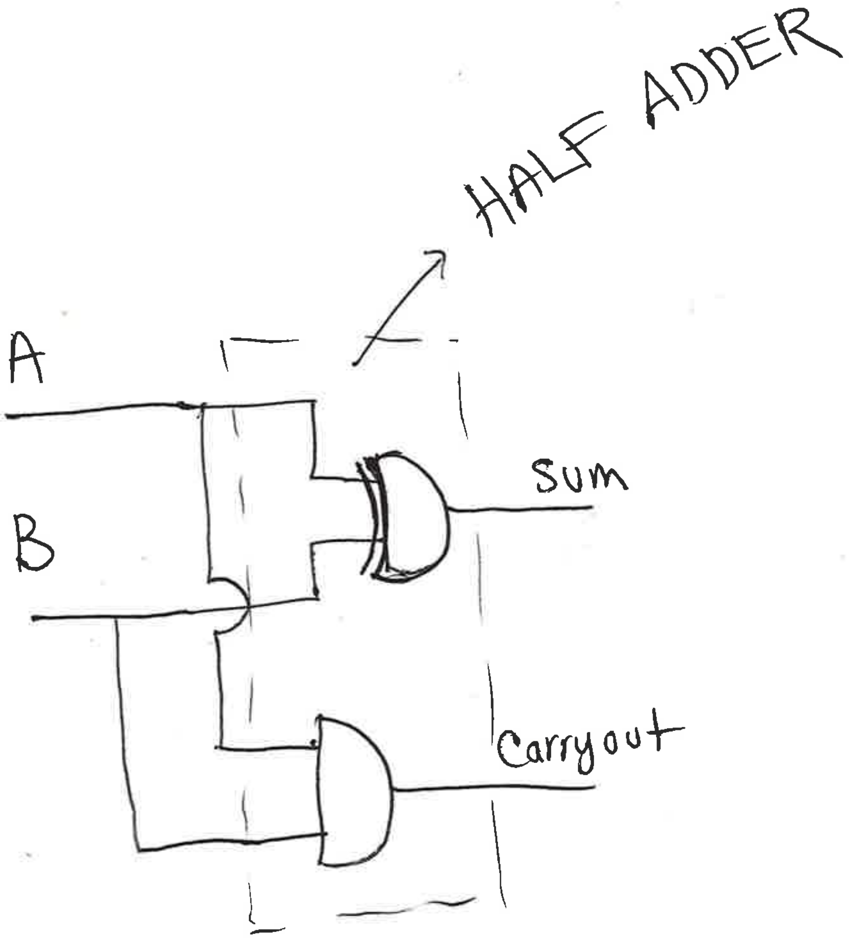
input

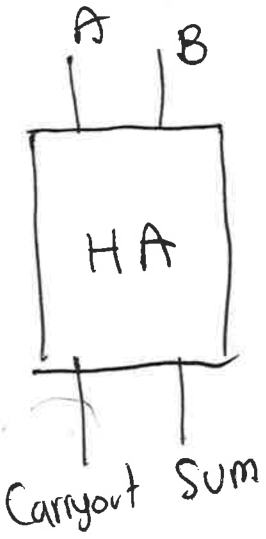
outputs

XOR

AND

"Sum ⇔ XOR"





3 bit binary

Carryin

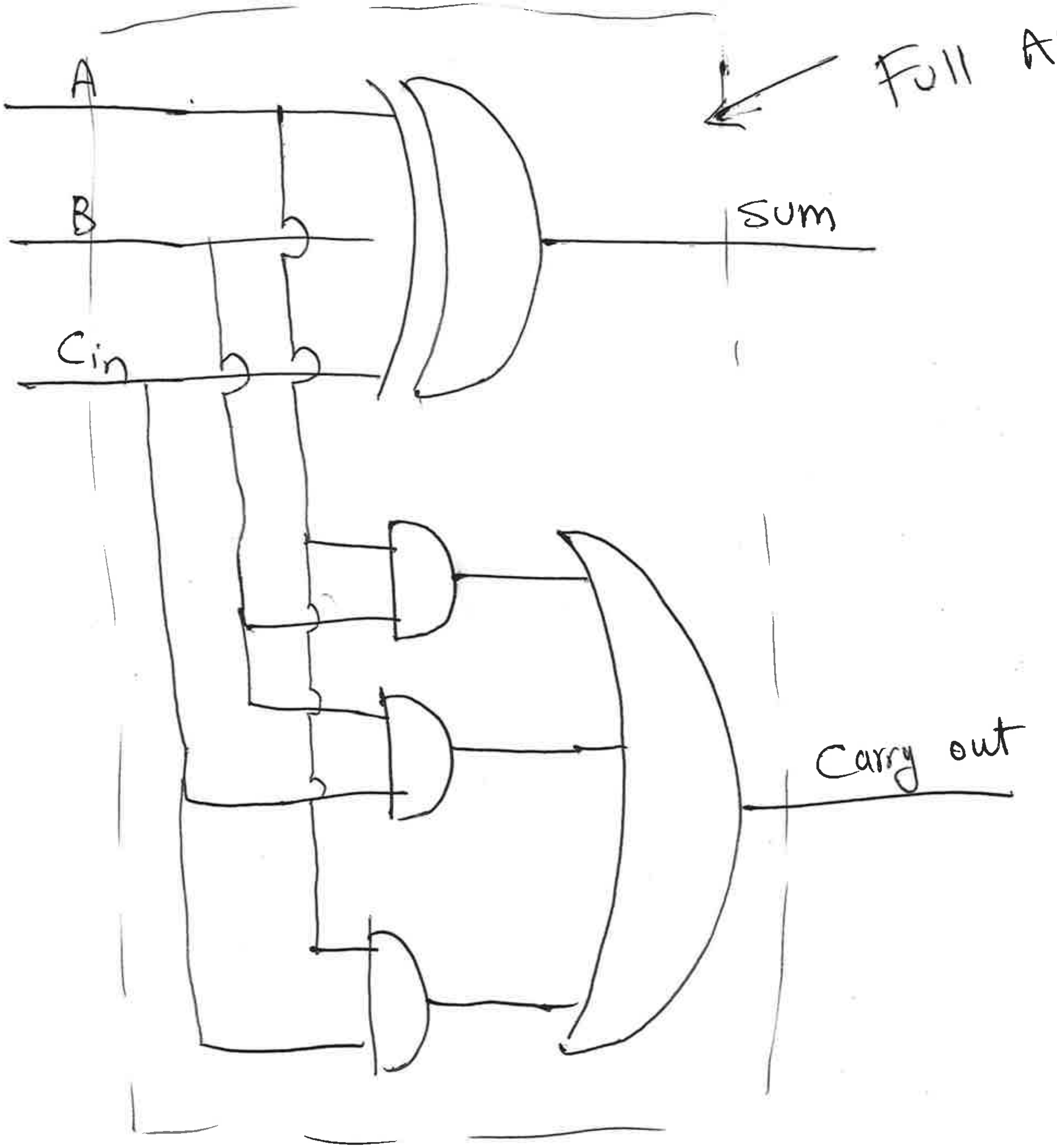
Cout

A	B	Cin	Sum	Carry out	
0	0	0	0	0	
0	0	1	1	0	
0	1	0	1	0	
0	1	1	0	1	* $\bar{A}BC_{in}$ +
1	0	0	1	0	
1	0	1	0	1	* $A\bar{B}C_{in}$ +
1	1	0	0	1	* $AB\overline{C_{in}}$ +
1	1	1	1	1	* ABC_{in}

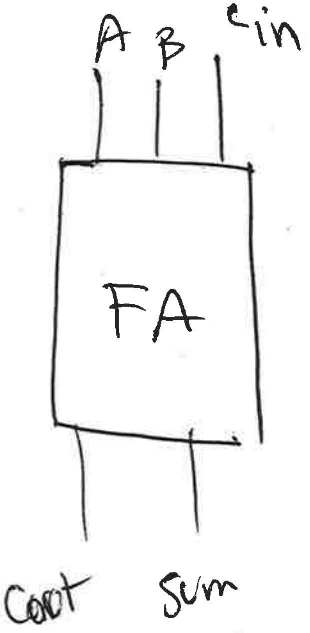
XOR

$$C_{out} = \underbrace{\bar{A}BC_{in} + A\bar{B}C_{in}}_{(\bar{A}B + A\bar{B})C_{in} = AC_{in} + BC_{in}} + \underbrace{AB\overline{C_{in}} + ABC_{in}}_{AB[\overline{C_{in}} + C_{in}] \to 1}$$

C_{in}	$\overline{C_{in}}$	$C_{in} + \overline{C_{in}}$
0	1	1
1	0	1



Full ADDER



pos. 0 0

Cin ↘ 0 0

pos. 1 0

Sum 1 0

Carryout 0 0 ✓

OF

Neg 1 1

1 1

Neg Cin → 1 0

1 0

1 1 ✓

OF

Neg 1 1

pos. 0 0

Cin 1 0

0 1

1 0 ✓ ✓

Cin	Cout	OF
0	0	0
0	1	1
1	0	1
1	1	0

XOR gate

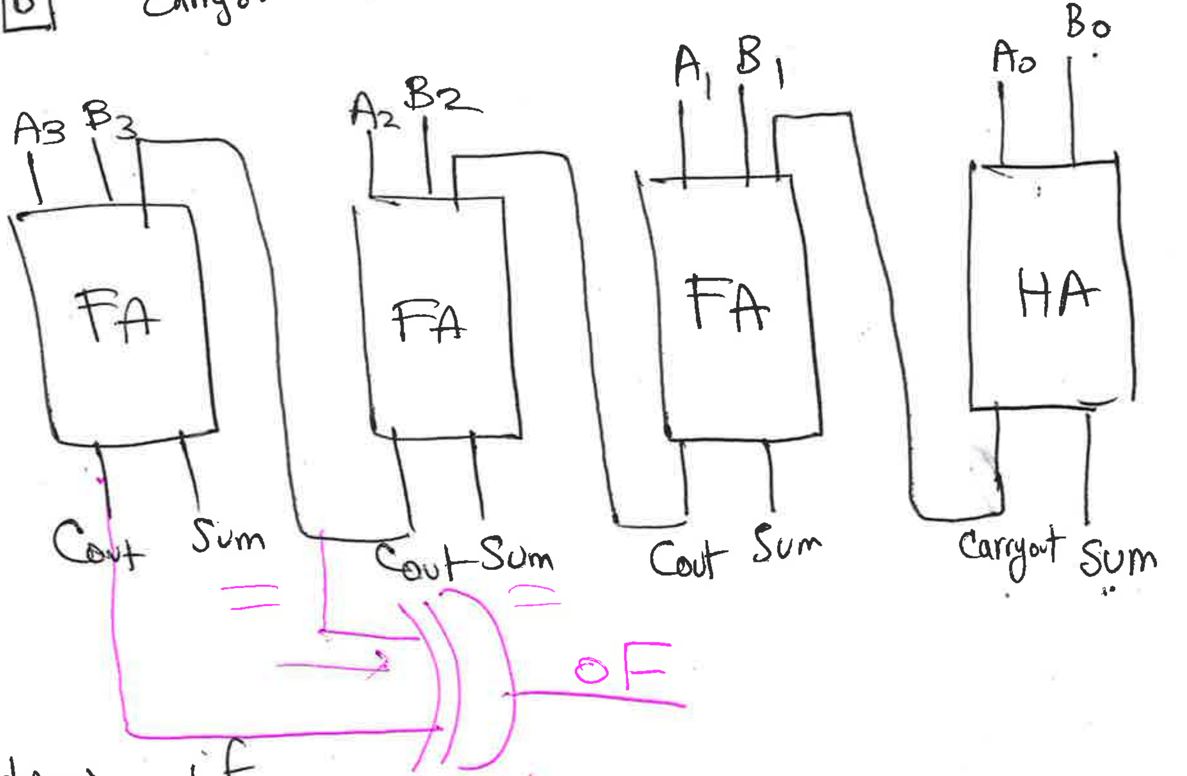
$\boxed{1}$ $\boxed{0}$

0 1 1 0

+ 1 0 1 1

0 1 Sum

$\boxed{1}$ $\boxed{0}$ Carryout



we overflow if

Carry in ≠ Carry out

$$\begin{array}{rcccc} & 0 & 1 & 0 & 1 = A = A_3 A_2 A_1 A_0 \\ + & \boxed{1} & 0 & 0 & 1 = B = B_3 B_2 B_1 B_0 \\ \hline \end{array}$$

$A + B$

$A - B$

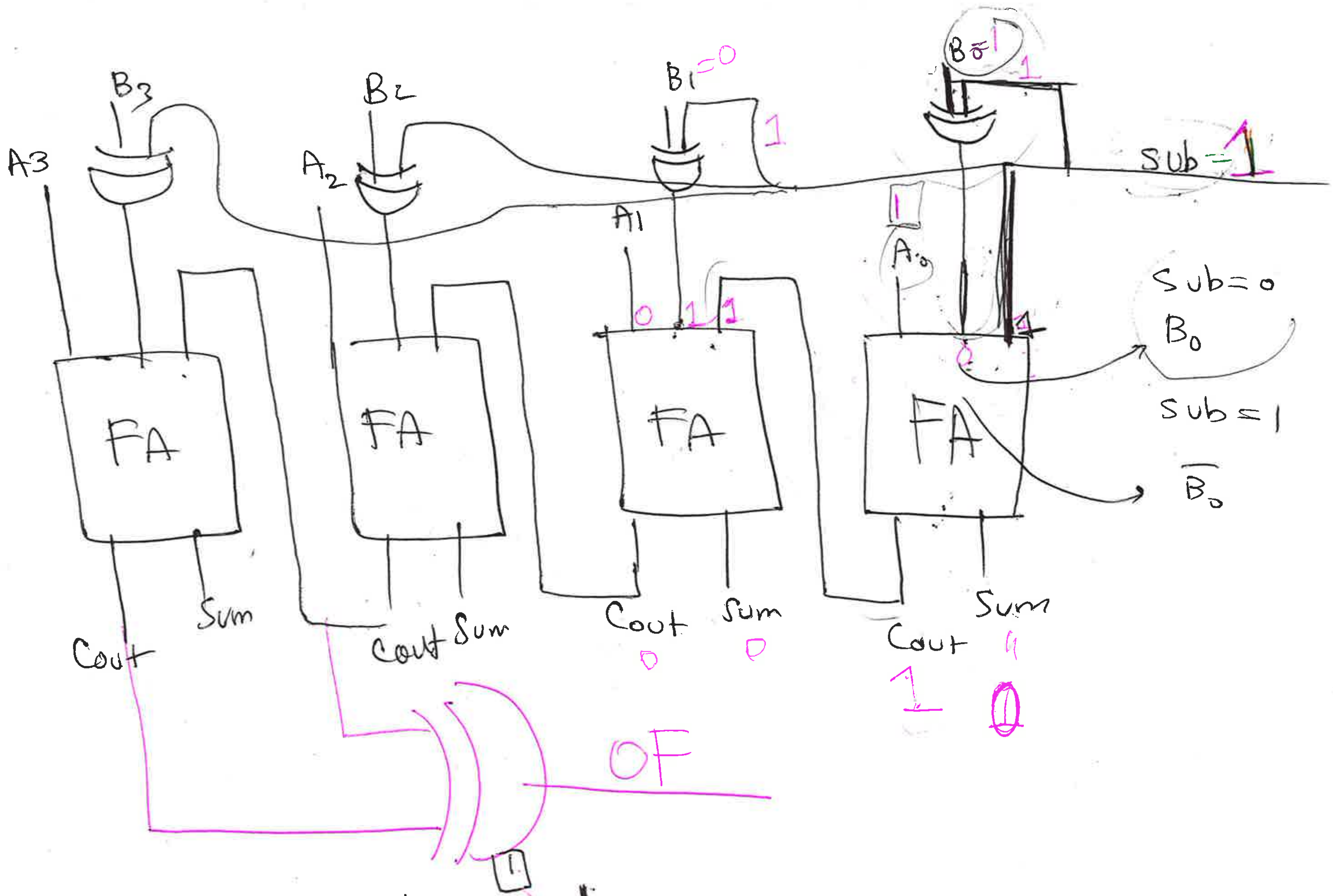
Reminder: 2s complement

① Flip all the bits

② to add 1 to it

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

$$\text{Subtractor} = \begin{cases} Sub = 0 \iff \text{we're adding} \\ Sub = 1 \iff \text{subtracting} \end{cases}$$

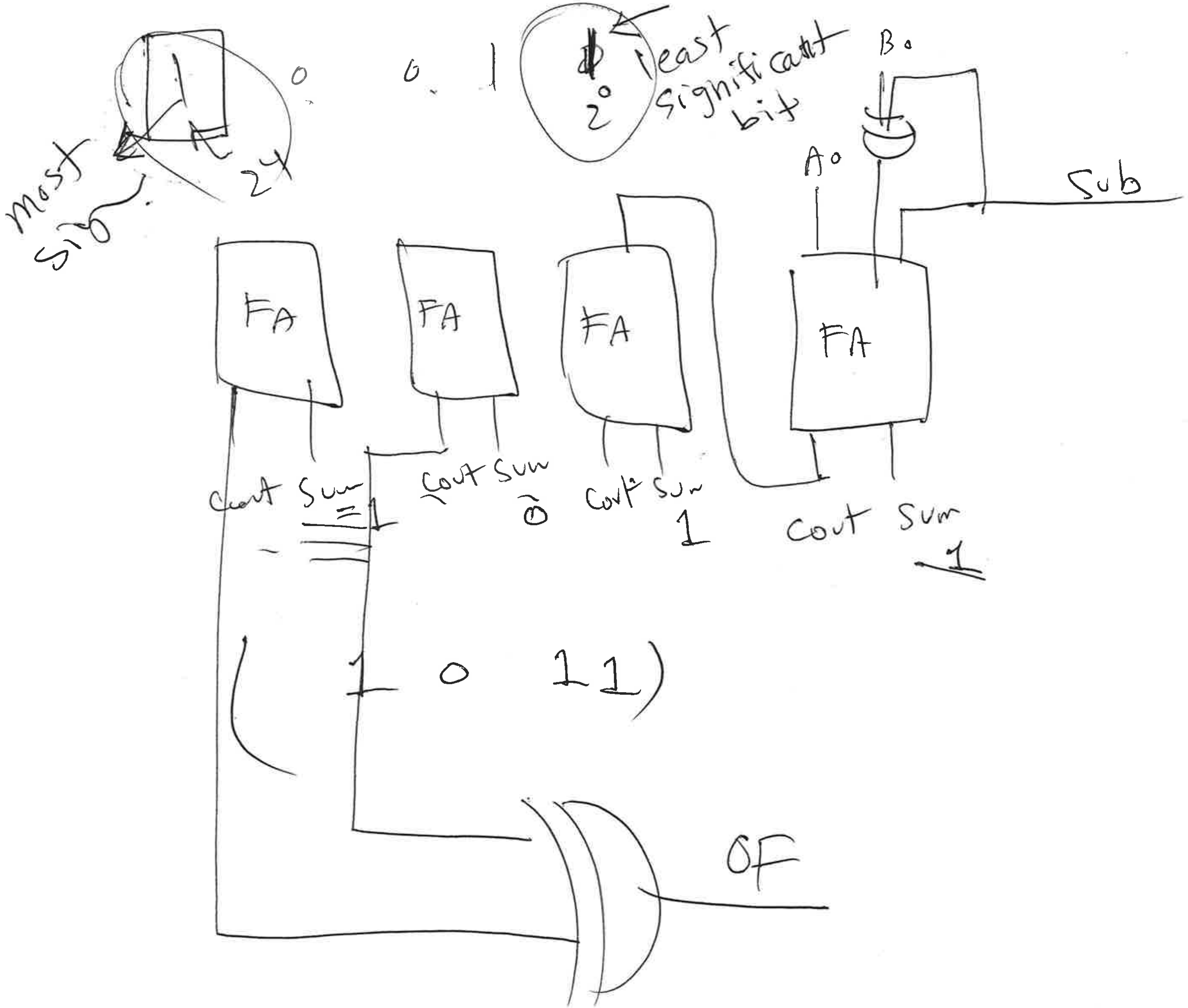


A = 0 1 0 1

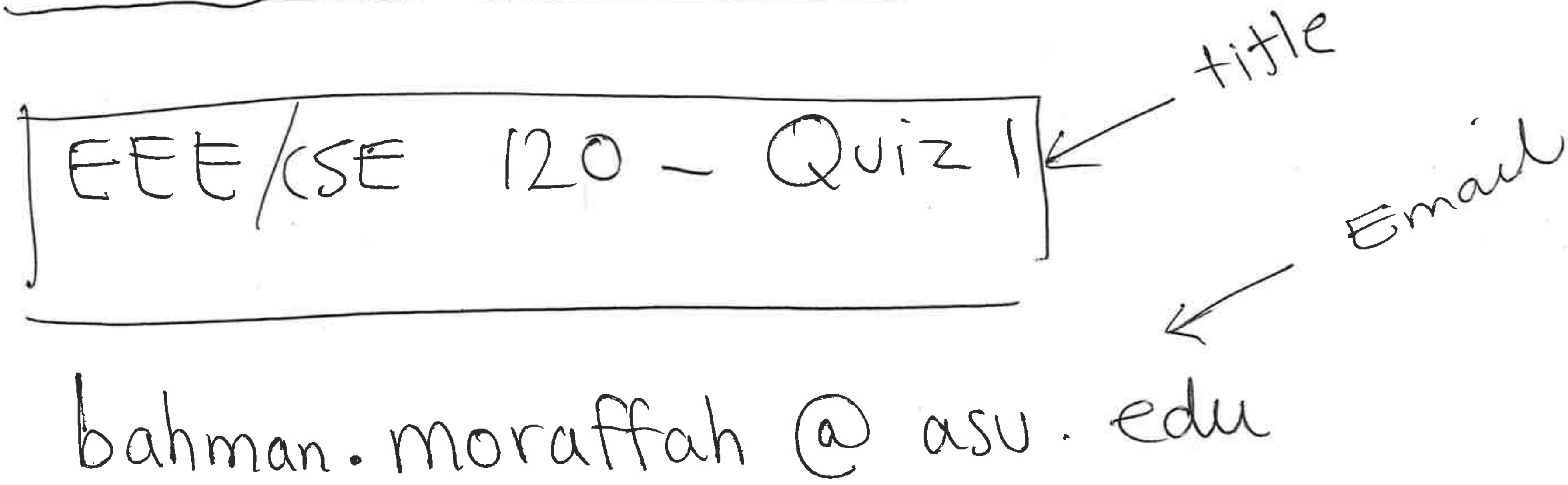
+ B = 1 0 0 1 ⟶ 1s : 0 1 1 0

0 0 0 1

0   0 1 1 1

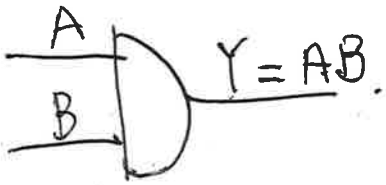

most sig
1
2⁴
0
0
1
0
2°
least significant bit
B0
A0
Sub
FA
FA
FA
FA
Cout Sum
=1
Cout Sum
0
Cout Sum
1
Cout Sum
1
(1 0 11)
OF

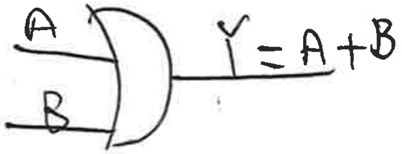
Lecture 6: EEE/CSE 120: Boolean Algebra



EEE/CSE 120 - Quiz 1

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

$1 \oplus 1 = 0$

$1 + 1 = 1$

$5 + 7 = 12$ (closure)

$5 + 0 = 5$

$5 \cdot 1 = 5$

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

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

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

"group"

① Closure :

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

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

$$A\cdot \underset{\uparrow ?}{B} = A \qquad A+\underset{\uparrow ?}{B} = A \qquad A\oplus \underset{\uparrow ?}{B} = A$$

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

$\boxed{B=1}$

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

$B=0$

EX: prove that

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

A	B	C	$A \cdot (B+C)$	$A \cdot B + A \cdot C$
0	0	0		
	1			
	1			
	1			

Summary :

$$\underbrace{\begin{array}{c} A \cdot 0 = 0 \\ A \cdot 1 = A \end{array}}_{\text{AND gate}}$$

$$\underbrace{\begin{array}{c} A \cdot A = A \\ A \cdot 0 = 0 \end{array}}_{\text{AND gate}}$$

$$\underbrace{\begin{array}{c} A + 1 = 1 \\ A + A = A \end{array}}_{\text{"OR" gate}}$$

$$A + 0 = A$$

$$1 + 1 = 1$$

Example: "OR" gate

line	A	B	A+B	
0	0	0	0	
1	0	1	1	← $\bar{A}B$ +
2	1	0	1	← $A\bar{B}$ +
3	1	1	1	← AB

$$Y = \text{output} = \underbrace{\bar{A}B + A\bar{B} + AB}_{\text{Sum of products}} = A+B$$

$$= \underset{\text{min-term}}{\sum m}(1,2,3)$$

0

1

⋮ ←

7

⋮

15

Example: $Y = \underbrace{\bar{A}B + A\bar{B} + AB}_{\text{Canonical form}} = \underbrace{A+B}_{\text{Simplified form}}$

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".

"Cananoical Form"

③ Sum them all.

EEE/CSE 120 - Quiz 1 - resubmit

Example:

A	B	$A \oplus B$	
0	0	0	
0	1	1	← $\overline{A}B$
1	0	1	← $A\overline{B}$
1	1	0	

"SOP"

$$Y = \text{output} = \overbrace{\overline{A}B + A\overline{B}}^{\text{canonical form}}$$

$$= \underbrace{A \oplus B}_{\text{Simplified.}}$$

Lecture 7

EEE/CSE 120 : Boolean Algebra II

$\boxed{1}$ closure $(1+1=1\ ,\ 1\oplus 1=0)$

$\boxed{2}$ Associativity

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

$\boxed{3}$ Neutral element , (Identity)

$$A\underset{?}{B} = A \qquad A+\underset{?}{B}=A \qquad A\oplus\underset{?}{B}=A$$

$$B=1 \qquad B=0 \qquad B=0$$

$\boxed{4}$ Inverse element

$$\underbrace{\underbrace{A\cdot\underset{?}{B}=0}_{A\bar{A}=0\ \checkmark} \qquad A+\underset{?}{B}=1 \qquad A\oplus\underset{?}{B}=1}_{B=\overline{A}}$$

⑤ Commutativity

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

$$A + B = B + A$$

⑥ Distributivity

① $$\begin{tikzcd}[column sep=small] A \cdot (\arrow[r, bend right] \arrow[rr, bend right] & B + & C) = A \cdot B + A \cdot C \end{tikzcd}$$

② $$A + (B \cdot C) = (A + B) \cdot (A + C)$$

Example : "OR" gate

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

$$Y = \underbrace{\bar{A}B + A\bar{B} + AB}_{\text{Canonical form}} \overset{?}{=} \underbrace{A+B}_{\text{Simplified form}}$$

$$\boxed{\sum m(1,2,3)}$$

min-term

$\underbrace{1\,0}$

00 → 0

01 → 1

10 → 2

11 → 3

$$Y = \underbrace{\bar{A}B + \underbrace{A\bar{B} + AB}}= \bar{A}B + A\cdot(\underbrace{\cancel{\bar{B}+B}}_{1})$$ (Distributivity)

$$= \bar{A}B + A\cdot 1$$ (Inverse)

$$= \bar{A}B + A$$ (Identity, Neutral)

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

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

$$= (B+A)$$ (Identity)

$$= \underbrace{A+B}$$ (Commutativity)

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

* $\boxed{A+(B\cdot C) = (A+B)\cdot(A+C)}$

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

$$= \bar{a}bc + a\bar{b}c + ab(\cancelto{1}{c+\bar{c}})$$

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

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

$$= \bar{a}bc + a\cdot\cancelto{1}{(\bar{b}+b)}\cdot(c+b)$$

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

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

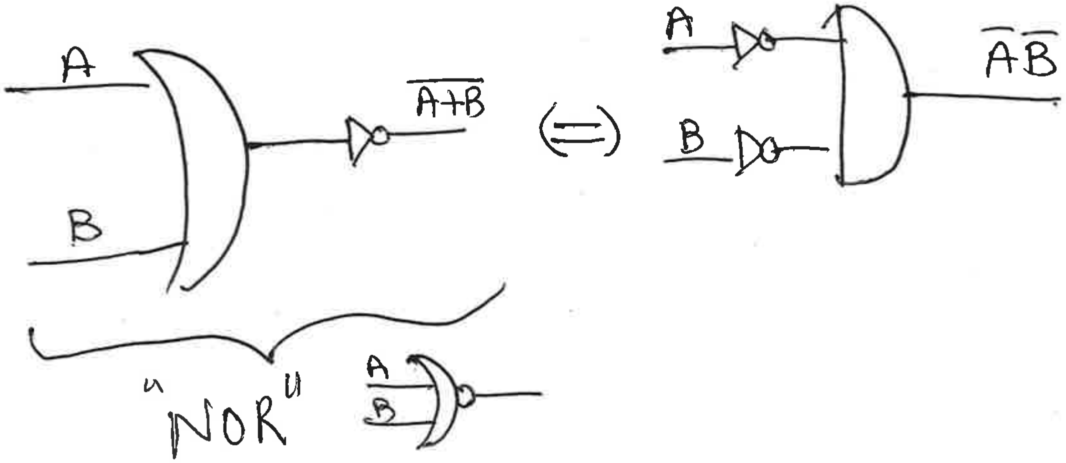
$$= \underbrace{(\bar{a}b + a)}\cdot c + a\cdot b$$

$$= \cancelto{1}{(\bar{a}+a)}\underbrace{(b+a)\cdot c} + a\cdot b$$

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

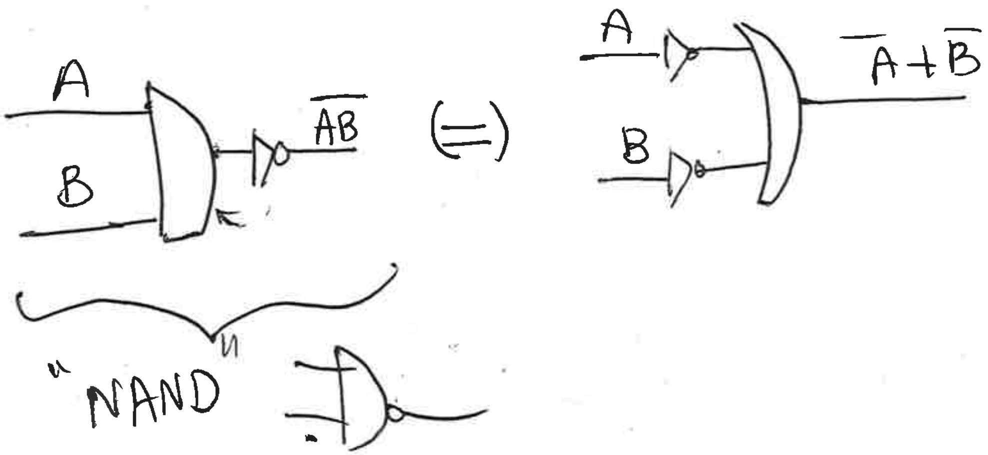
Demorgan's law :

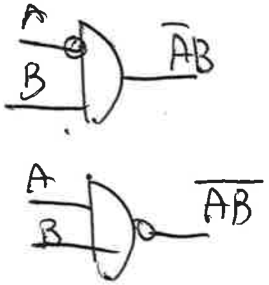
① $\overline{A+B} = \bar{A}\,\bar{B}$



② $\overline{AB} = \bar{A} + \bar{B}$

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

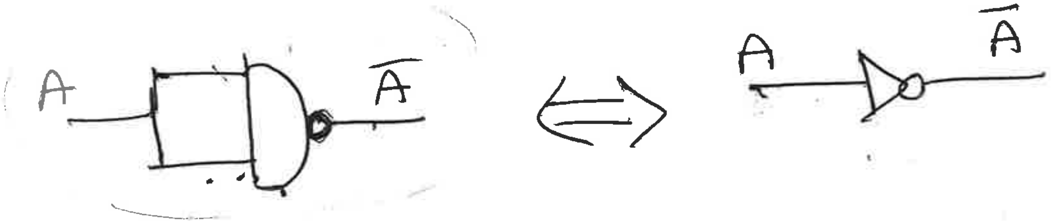




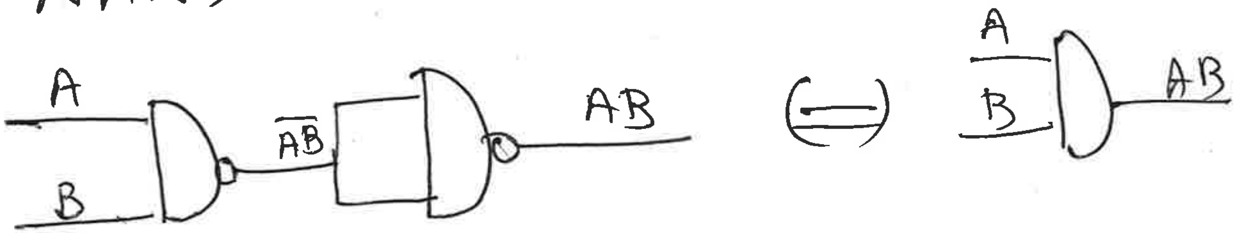
Q: Can we use "NAND" gates or "NOR" gates to build any circuit? YES!

"NAND":

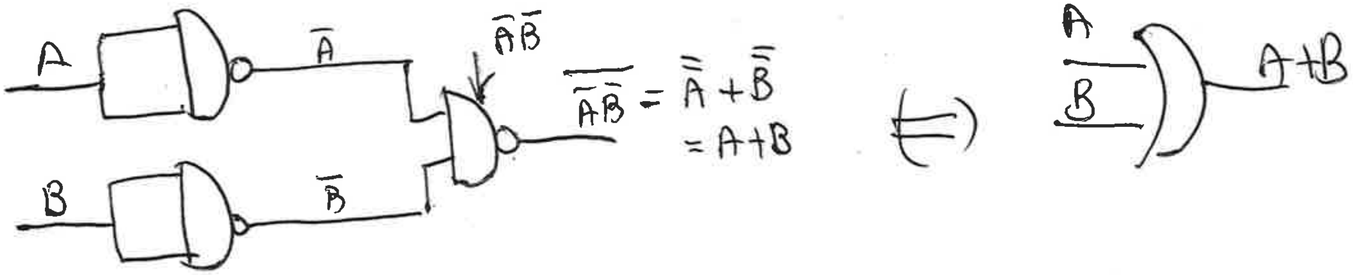
① NAND to build a NOT gate:



② NAND to build AND



③ NAND gate to build OR



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

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

Example: NAND is functionally complete.

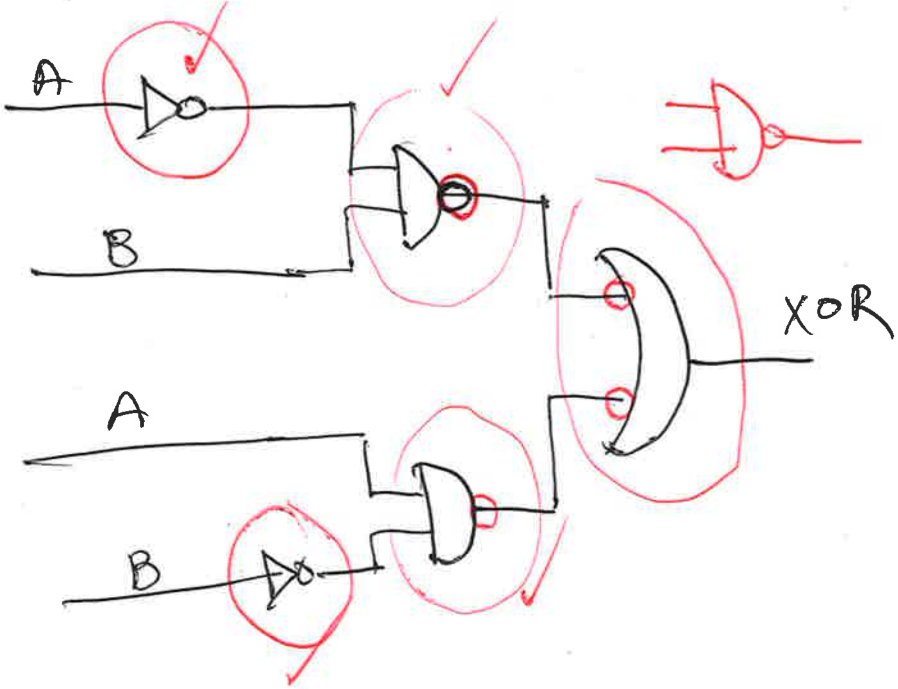
NOR is functionaly complete.

Example : XOR gate ⟶ build XOR gate in terms of NAND gates.

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

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

✓



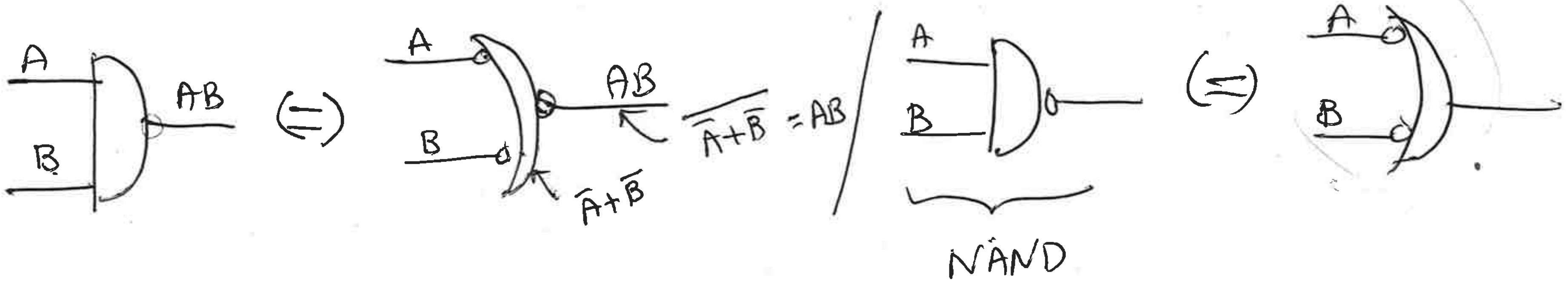
Steps to build a circuit in terms of "NAND" gates or "NOR" gates :

1. Find the equation representing the gate or gates

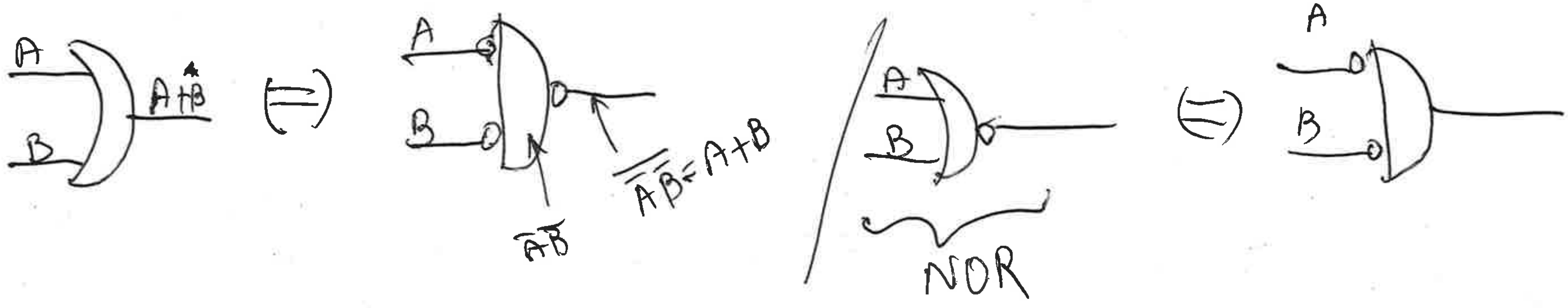
2. Draw the equation w/o paying attention to the NAND or NOR.

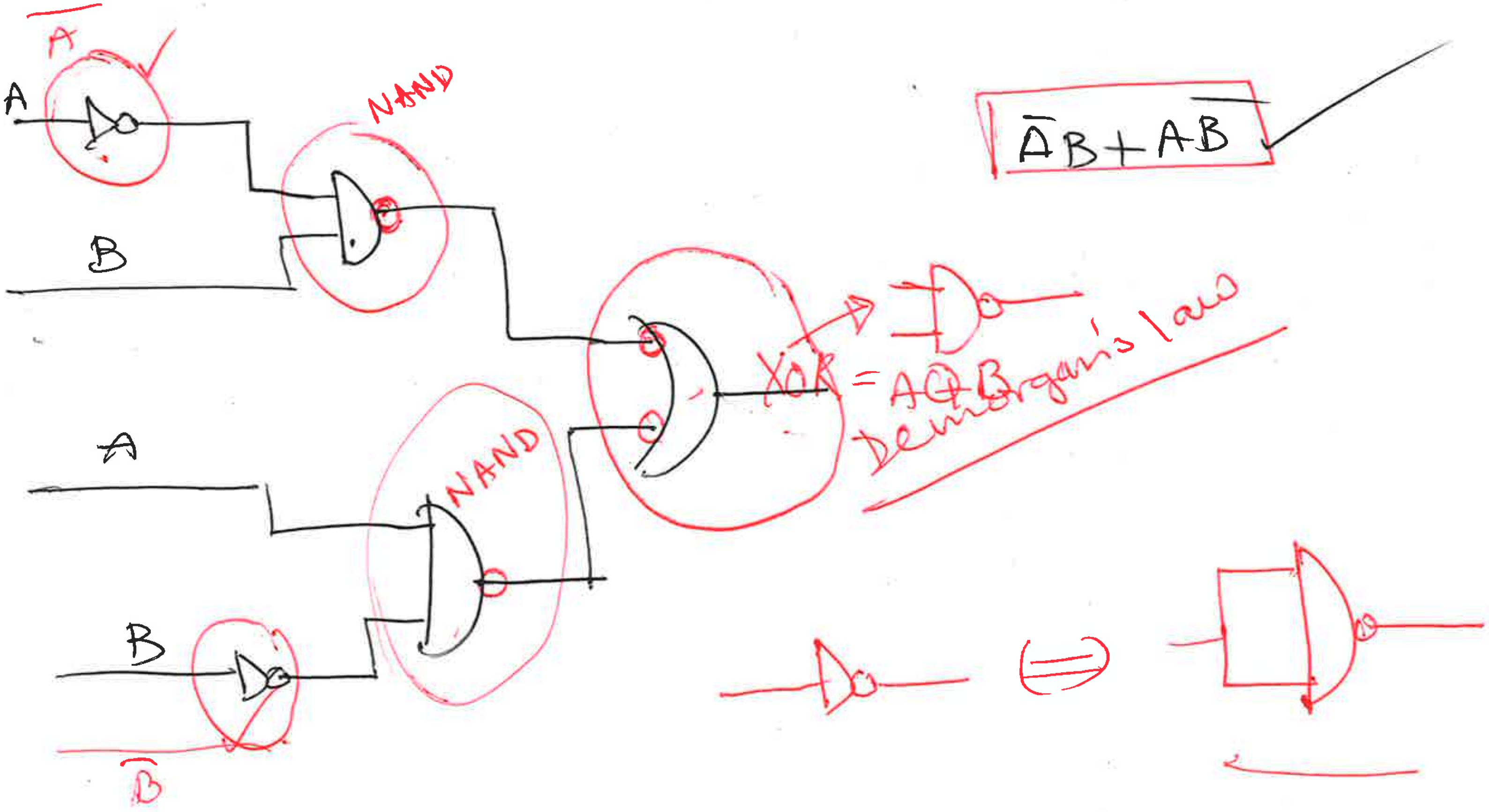
3. Play the bubble game :

(a)



(b)





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

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

$\overline{A}$

B

A

$\overline{B}$

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

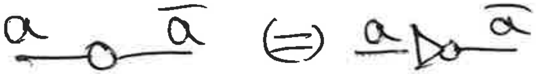
④ Short Quiz 1 is on Thursday.

Example: NAND gates, NOR gates functionally complete

How to build any circuit using NAND/NOR gates!

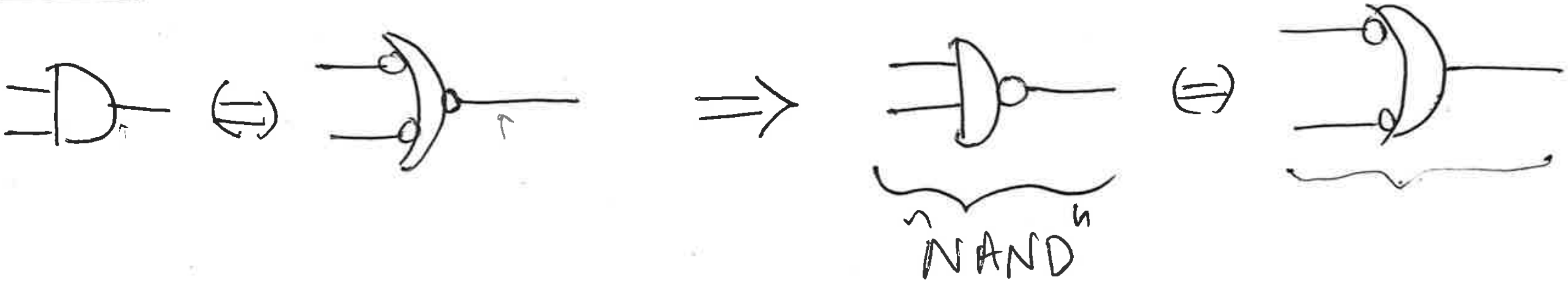
$\boxed{1}$ write down the function using truth table and draw it "normally"

$\boxed{2}$ play the bubble game

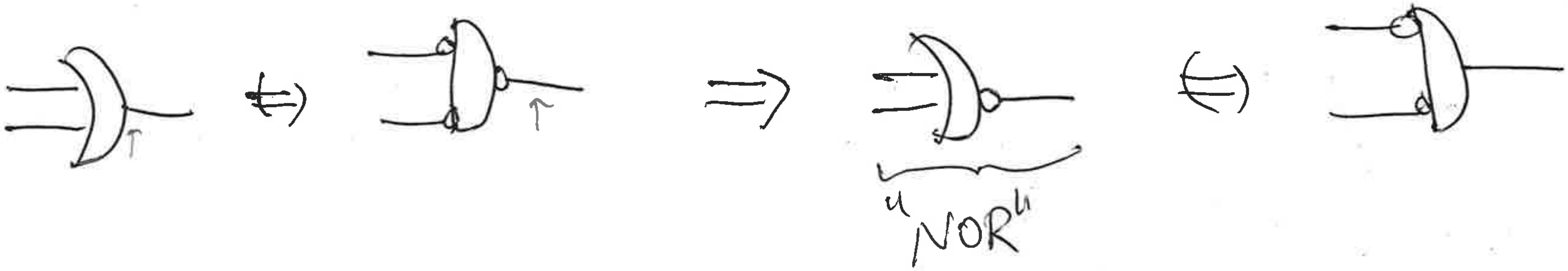


Rule:

①

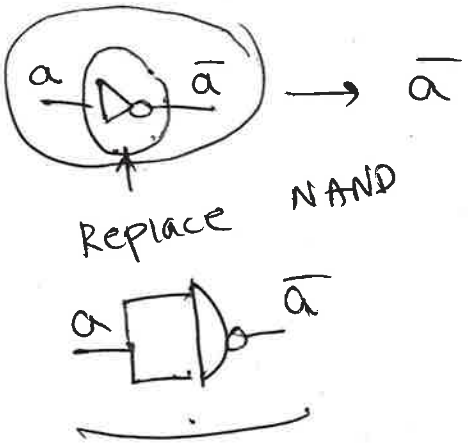


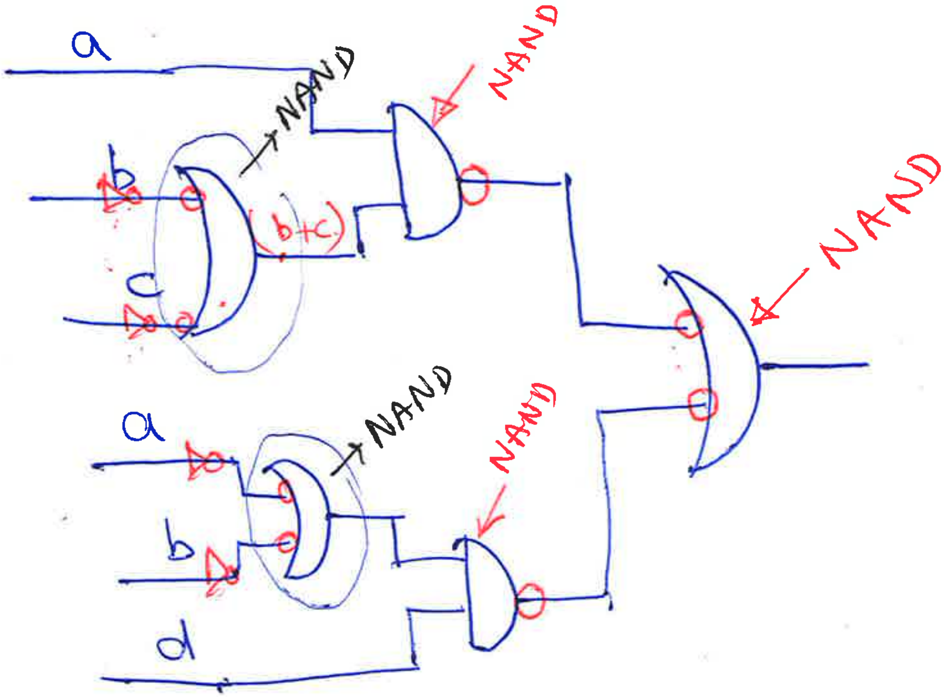
②

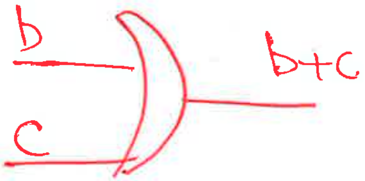


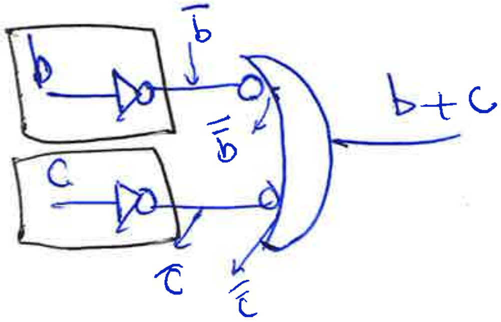
Example: Assume we have access to all inputs as well as their complement, Draw the following circuit using only "NAND" gates!

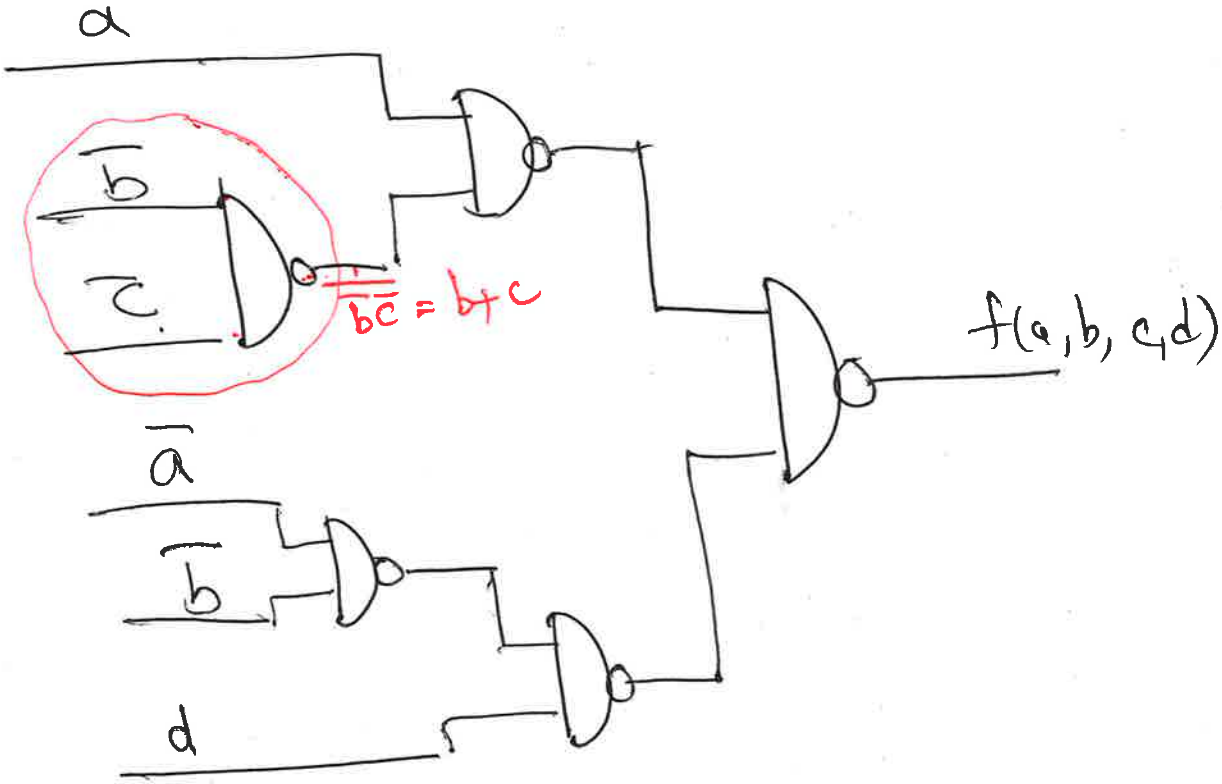
$$f(a,b,c,d) = \underline{a(b+c)} + \underline{d'(a+b)}$$



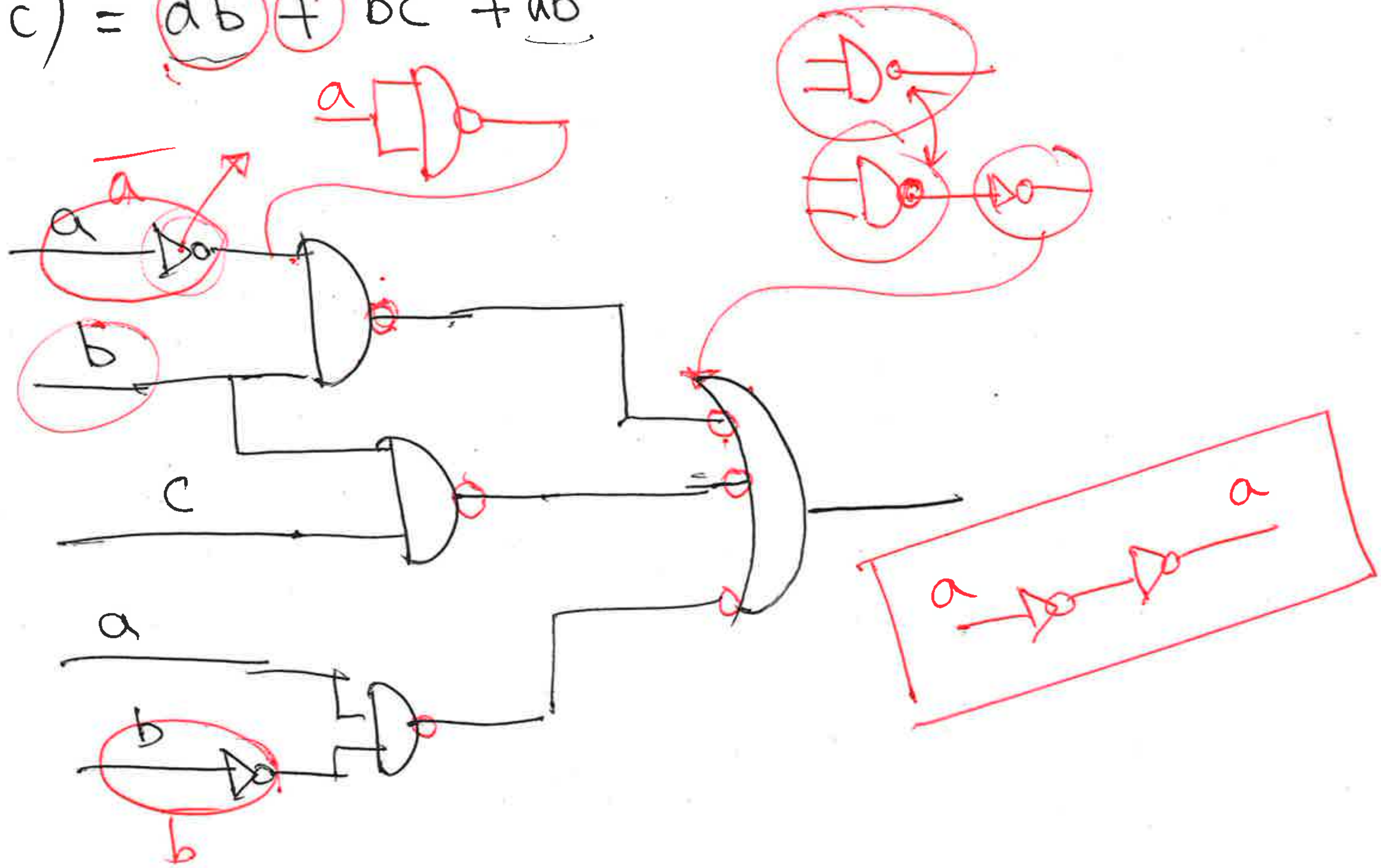


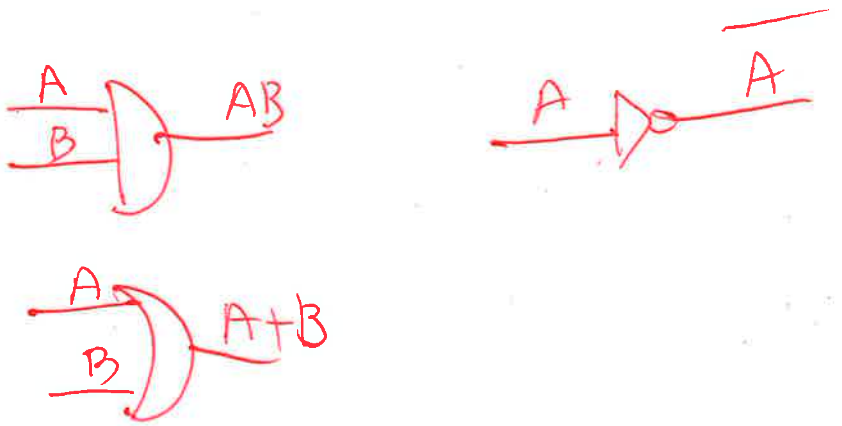





a
$\bar{b}$
$\bar{c}$
$\overline{\bar{b}\bar{c}} = b+c$
$\bar{a}$
$\bar{b}$
d
$f(a,b,c,d)$

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





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

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.

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

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

Example:

output

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

POS ?

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

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

$\leftarrow (A+B+\overline{C})$

$\leftarrow (\overline{A}+B+C)$

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

001 100

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

Max-term

Max-term repr.

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

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

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.

$\underbrace{(1+1)}_{1}+1 = 1$

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

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

$a=0, b=0, c=0 \rightarrow F = \underbrace{1\times1\times1}_{1} + \underbrace{1\times0}_{0} + \underbrace{1\times0}_{0}$
$= 1$

$F(0,1,1) = \underbrace{1\times1\times1}_{1} + \underbrace{1\times1}_{1} + \underbrace{0\times1}_{0} = 1$

$a=0, b=1, c=1$

$\underbrace{1\times0\times0}_{0} + \underbrace{1\times1}_{1} + \underbrace{0\times1}_{0} = 1$

$a=1, b=0, c=1$

$\overbrace{0\times0\times0}^{0} + \overbrace{0\times1}^{0} + \overbrace{1\times1}^{1} = 1$

ⓐ $Y = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}bc + a\bar{b}c$ (sop.)

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

ⓒ $F(a,b,c) = \sum m(0,1,3,5)$

ⓓ $F(a,b,c) = \prod M(2,4,6,7)$

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

$A B \overline{C} +$

$A B C$

$A B [C + \overline{C}] = \underline{\underline{AB}}$

A & B are NOT changing

changing variable

0	0
0	1
1	0
1	1

0	0
0	1
1	1
1	0

grey code!

$\neq \cancel{\overline{A+B}}$

Demorgan's law

$$\begin{aligned} \overline{A+B} &= \overline{A}\,\overline{B} \\ \overline{AB} &= \overline{A} + \overline{B} \end{aligned}$$

$\neq \cancel{\overline{A}\,\overline{B}}$

$y = f(x) = x^2 + 2$

$y = f(5)$

$= 5^2 + 2$

$= 27$

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

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

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

C is changing variable

A and B are not changing

AB

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

0	0
0	1
1	1
1	0

2-bit

ABC

B

K-MAP

C \ AB	00	01	11	10
0	1_{0}	1_{2}	1_{6}	0_{4}
1	0_{1}	1_{3}	1_{7}	1_{5}

$\bar{A}\bar{C}$

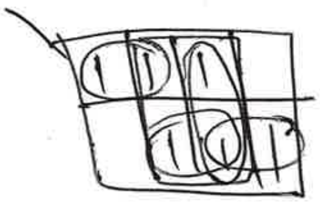
AC

011

Rules of minimum sum of product:

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

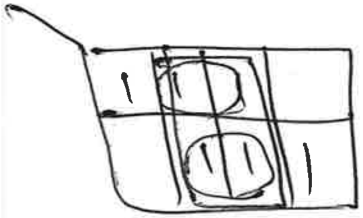
2 group all the ones into groups of 2^n elements (vertically / Horizantally but NOT diagonlly)



groups are called implicants!

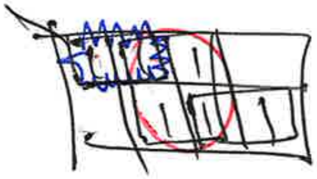
3 Find the largest groups possible

↳ prime implicants.



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

↳ essential prime implicants



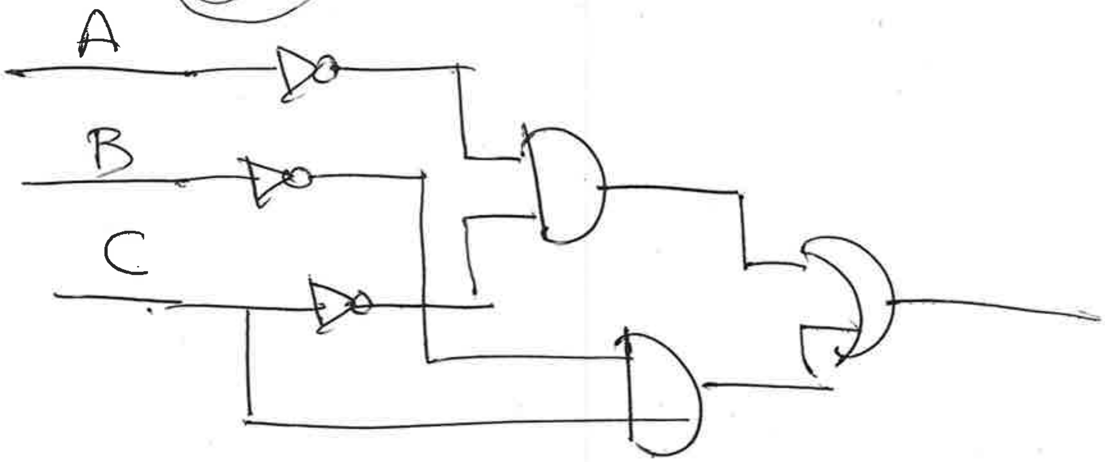
5 Add the product terms for the products of the eliminated changing variable. (essential prime implicants)

C \ AB	00	01	1ϕ	10
0	1	1 2	0 6	0 4
1	1	3	7	1 5

C \ AB	00	01	11	10
0	1	1		
1	1			1

$\bar{A}\bar{C}$ $\bar{B}C$

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



Example: Full Adder

#	A	B	C_{in}	C_{out}	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	1	1	1	← $\bar{A}BC_{in}$
4	1	0	0	0	
5	1	0	1	1	← $A\bar{B}C_{in}$
6	1	1	0	1	← $AB\overline{C_{in}}$
7	1	1	1	1	← ABC_{in}

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

Q: Simplify Y

① Boolean Alg.

② K-MAP.

3 inputs ⇒ 3-variable K-MAP.

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

① $$Y = \underline{\underline{\bar{A}BC_{in}}} + A\bar{B}C_{in} + \underset{\checkmark}{AB\bar{C}_{in}} + \underline{\underline{ABC_{in}}} + ABC_{in} + \underset{\checkmark}{ABC_{in}}$$

$$A + A = A$$

$$\underbrace{BC_{in}[\bar{A} + A]}_{1} + \underbrace{AC_{in}[\bar{B} + B]}_{1} + \underbrace{AB[\bar{C}_{in} + C_{in}]}_{1}$$

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

②

C_{in} \ AB	00	01	11	10
0	0	2	1	4
1	1	1 3	1	1 5

AB

BC_{in}

AC_{in}

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

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	

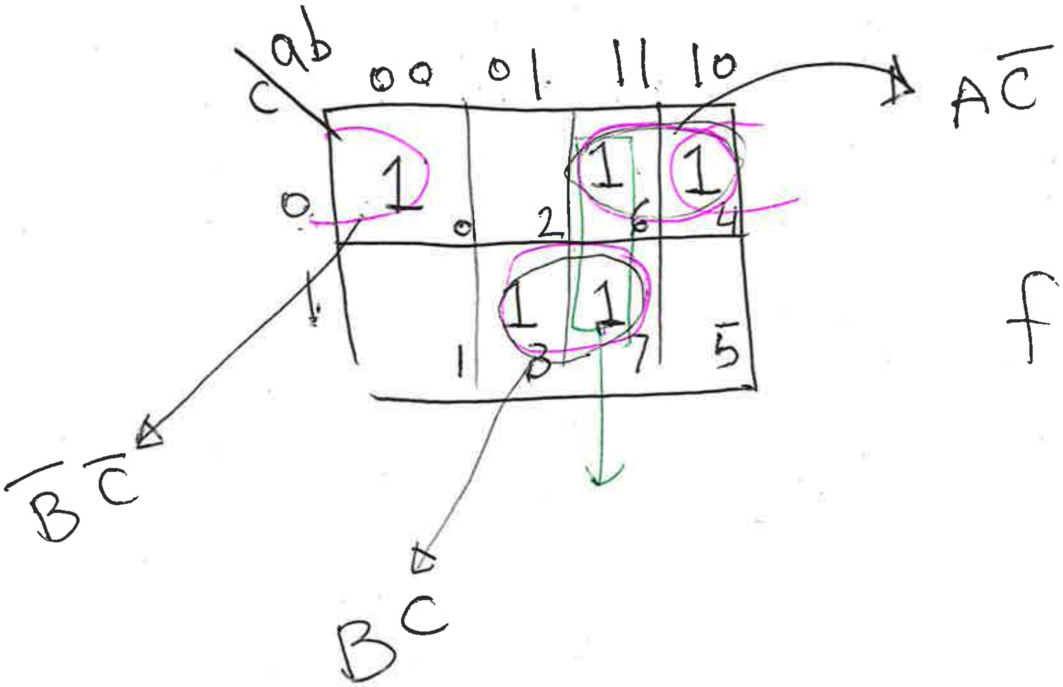
2. office hours 9:30 - 10:15 Am (T/TH)

3. Have Quartus installed on your computer.

Example: minimum sum of product of

$$f(a,b,c) = \Sigma m(\underbrace{0,3,4,6,7})$$

K-MAP → 3-var

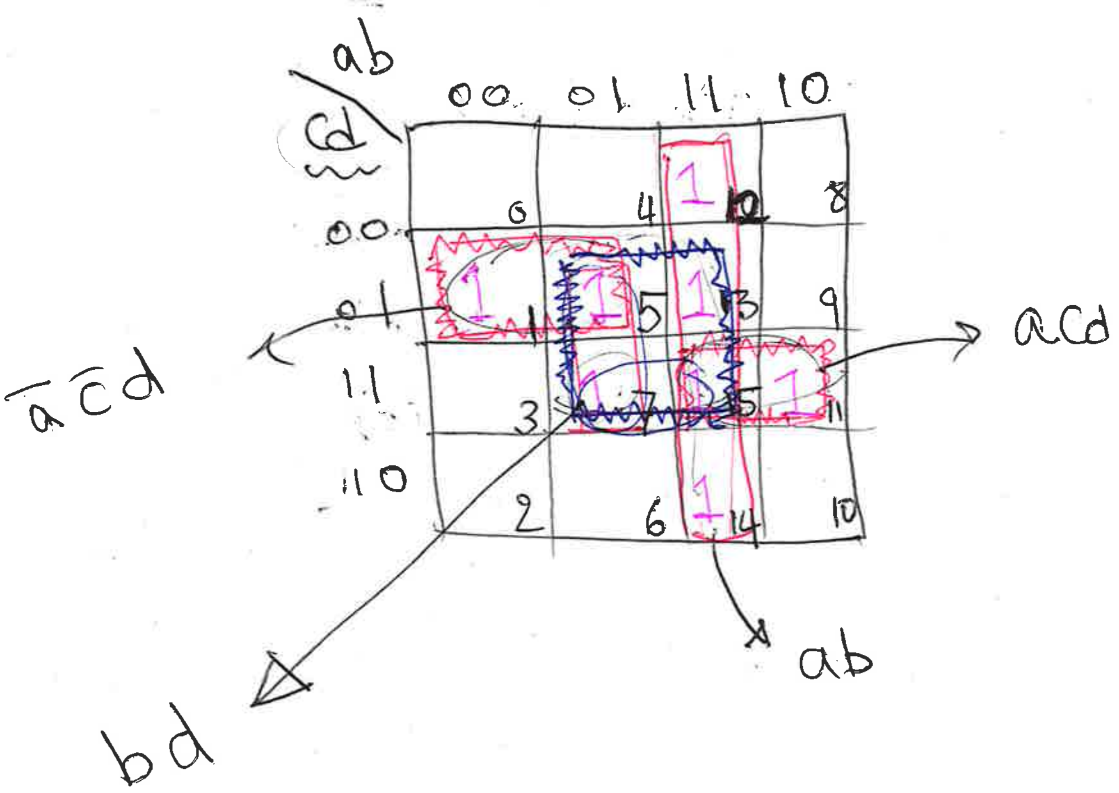


$$f(a,b,c) = \underbrace{\bar{B}\bar{C} + BC + A\bar{C}}$$

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

4-var K-MAP

$11\ 00 \rightarrow 12$



cd \ ab	00	01	11	10
00	0	4	1 12	8
01	1 1	1 5	1 13	9
11	3	1 7	1 15	1 11
10	2	6	1 14	10

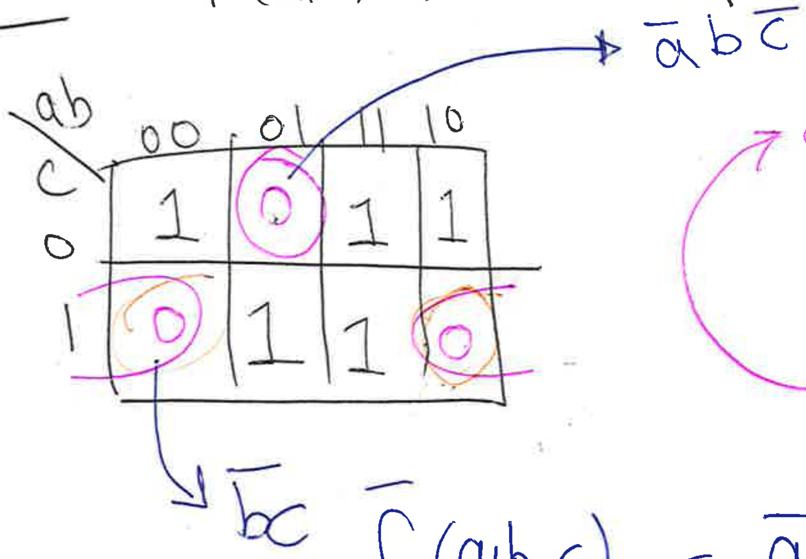
→ minimum sum of product

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

$\cancel{b(a+d)}$
sum

$$\bar{\bar{F}} = f$$

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



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

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

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

Demorgan's law:

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

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

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

How to write min POS :

1. group all "Zeros" → essential Prime implicants and write the product

2. writing the product terms and ignoring the changing variable → sum them up $\Rightarrow \overline{f}$

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!

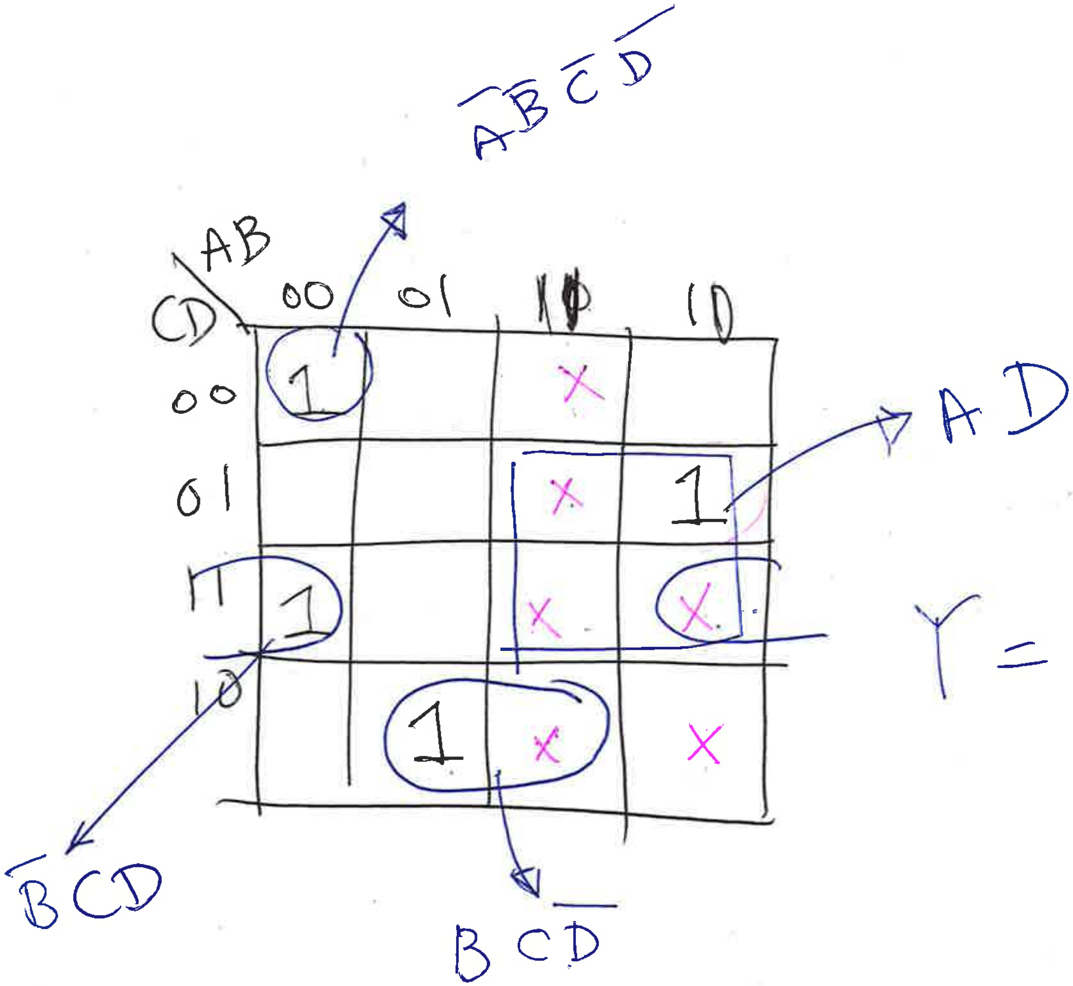
#	A	B	C	D	Y=output
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	X
11	1	0	1	1	X
12	1	1	0	0	X
13	1	1	0	1	X
14	1	1	1	0	X
15	1	1	1	1	X

SOP

$$Y = \sum m(0,3,6,9)$$

$$+ \underbrace{\sum d(10,11,12,13,14,15)}_{\text{"don't care"}}$$

can either be "1" or "0"



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

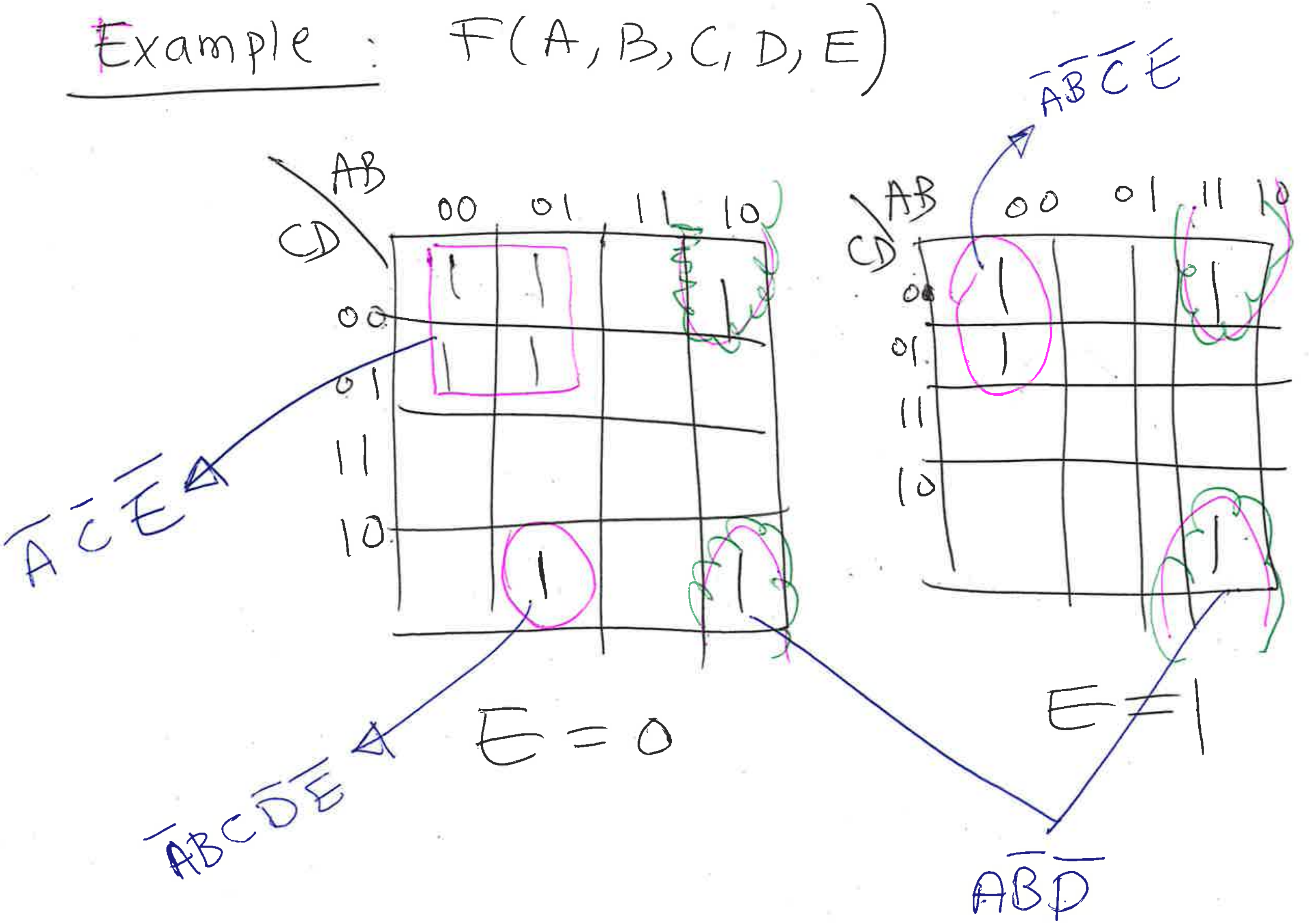
$$Y = \underline{AD} + \underline{\bar{B}CD} + \underline{BC\bar{D}} + \underline{\bar{A}\bar{B}\bar{C}\bar{D}}$$

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

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

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

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



E = 0

CD \ AB	00	01	11	10
00	1	1		1
01	1	1		
11				
10		1		1

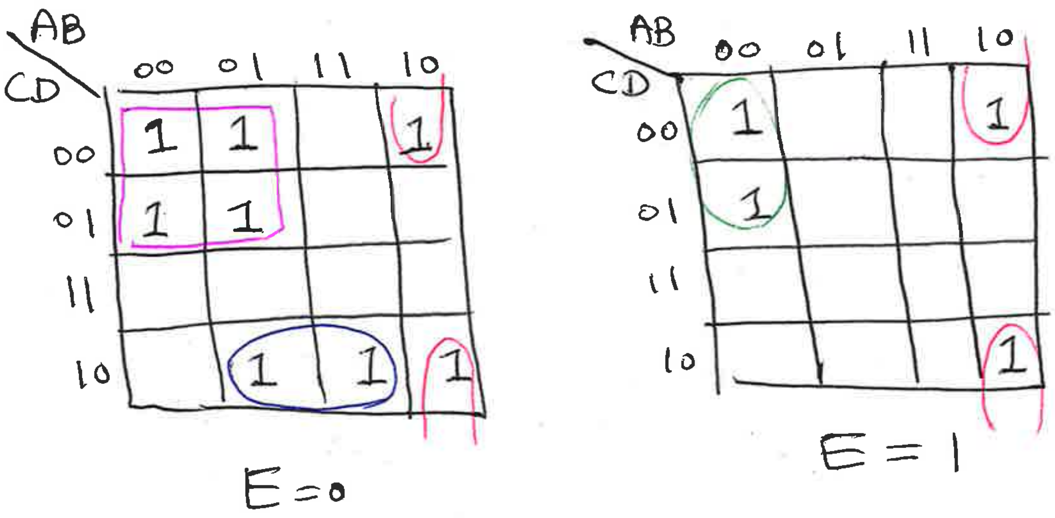
E = 1

CD \ AB	00	01	11	10
00	1			1
01	1			
11				
10				1

$$F(A,B,C,D,E) = \bar{A}\bar{C}\bar{E} + \bar{A}BC\bar{D}\bar{E} + \bar{A}\bar{B}\bar{C}E + A\bar{B}\bar{D}$$

Karnough MAP (5 variables):

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



CD \ AB	00	01	11	10
00	1	1		1
01	1	1		
11				
10		1	1	1

E=0

CD \ AB	00	01	11	10
00	1			1
01	1			
11				
10				1

E = 1

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

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

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

1. What are the Multiplexers (Encoders) ?
2. what are the demoltiplexers (Decoders) ?
3. How to use them ?

Annocement :

- 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
```

} Switch

select

I_0	I_1	S	Y = output	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	←
1	0	0	1	←
1	0	1	0	
1	1	0	1	←
1	1	1	1	←

S \ I_0I_1	00	01	11	10
0			1	1
1		1	1	

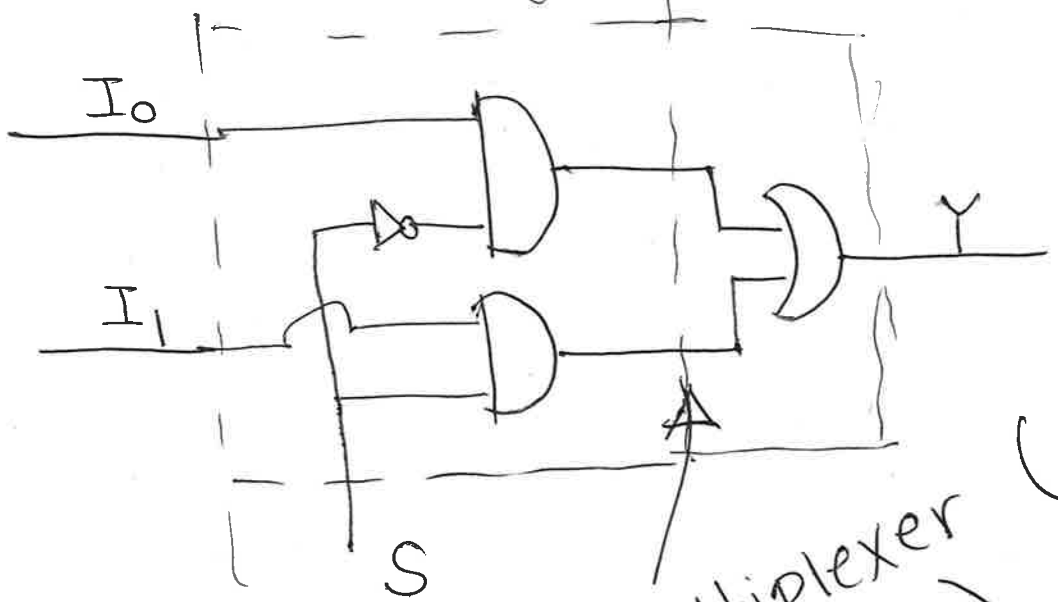
$I_0\bar{S}$

I_1S

$$Y = I_0\bar{S} + I_1S$$

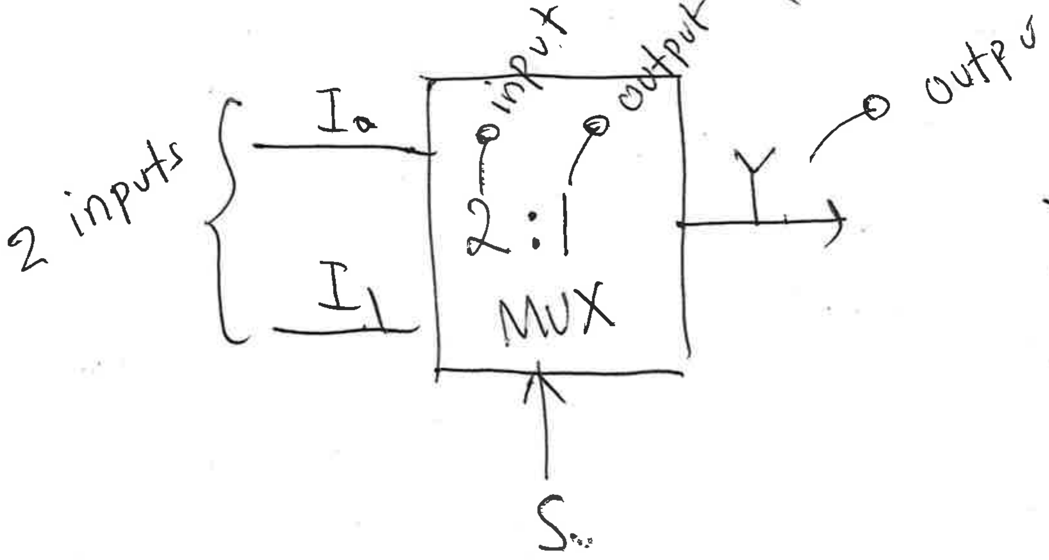
binary number represents the index

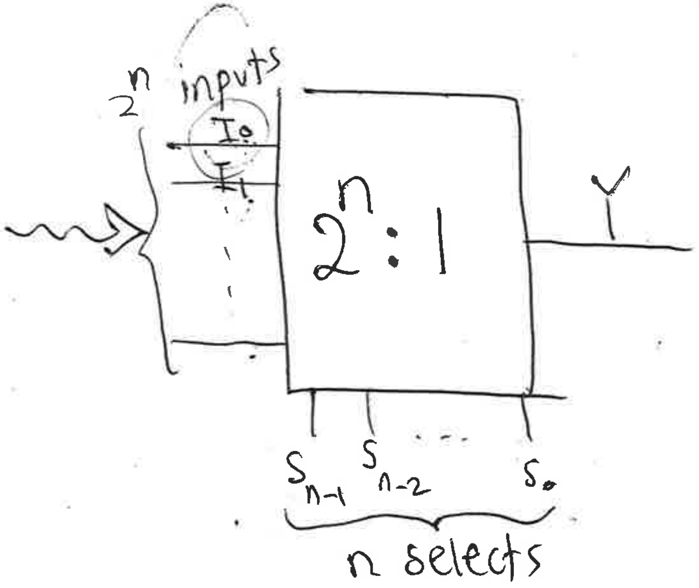
$$Y = I_0 \overline{S} + \underbrace{I_1 S}$$



Multiplexer (encoder)

(MUX)





Think of Multiplexers as signal routing devices.

<u>Application</u>: CPU

How to build $\underset{2^n}{4}:1$ MUX? (I_0, I_1, I_2, I_3)

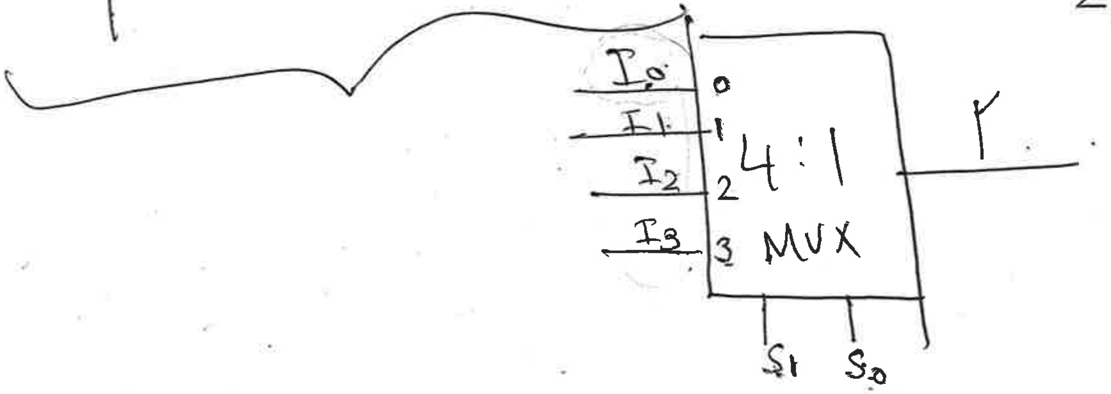
2^n ← # of select lines

MSB → S_1, LSB → S_0

# line	S_1	S_0	Y=output
0	0	0	I_0
1	0	1	I_1
2	1	0	I_2
3	1	1	I_3

$$Y = I_0 \underbrace{\overline{S_1}\,\overline{S_0}}_{0\ 0} + I_1 \underbrace{\overline{S_1}\,S_0}_{0\ 1} +$$

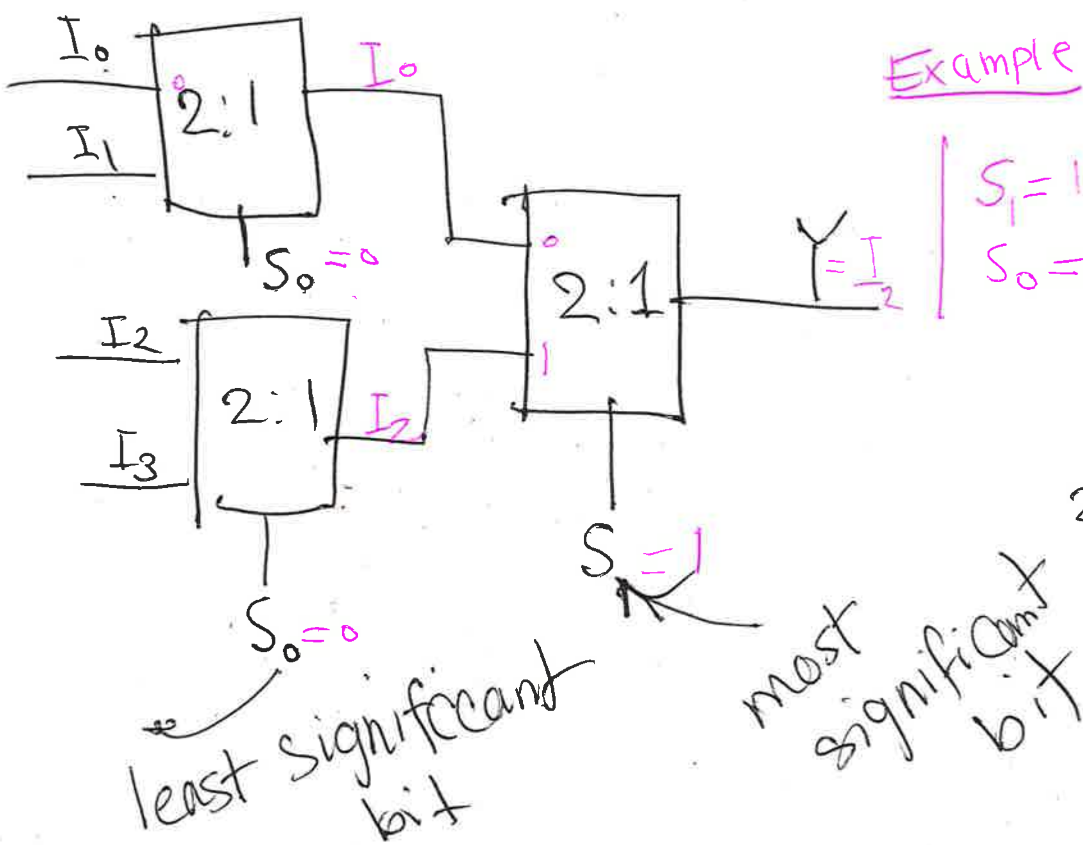
$$I_2 \underbrace{S_1\,\overline{S_0}}_{1\ 0} + I_3 \underbrace{S_1\,S_0}_{1\ 1}$$



$$f(a,b,c) = \underbrace{\sum m(1,2,4,5) + \sum d(0,6)}_{\text{represent the output}}$$

not input

Q: Can we build a 4:1 Mux using 2:1 Muxes?



Example:

$S_1 = 1$
$S_0 = 0$ → I_2

4:1 Mux

LSB

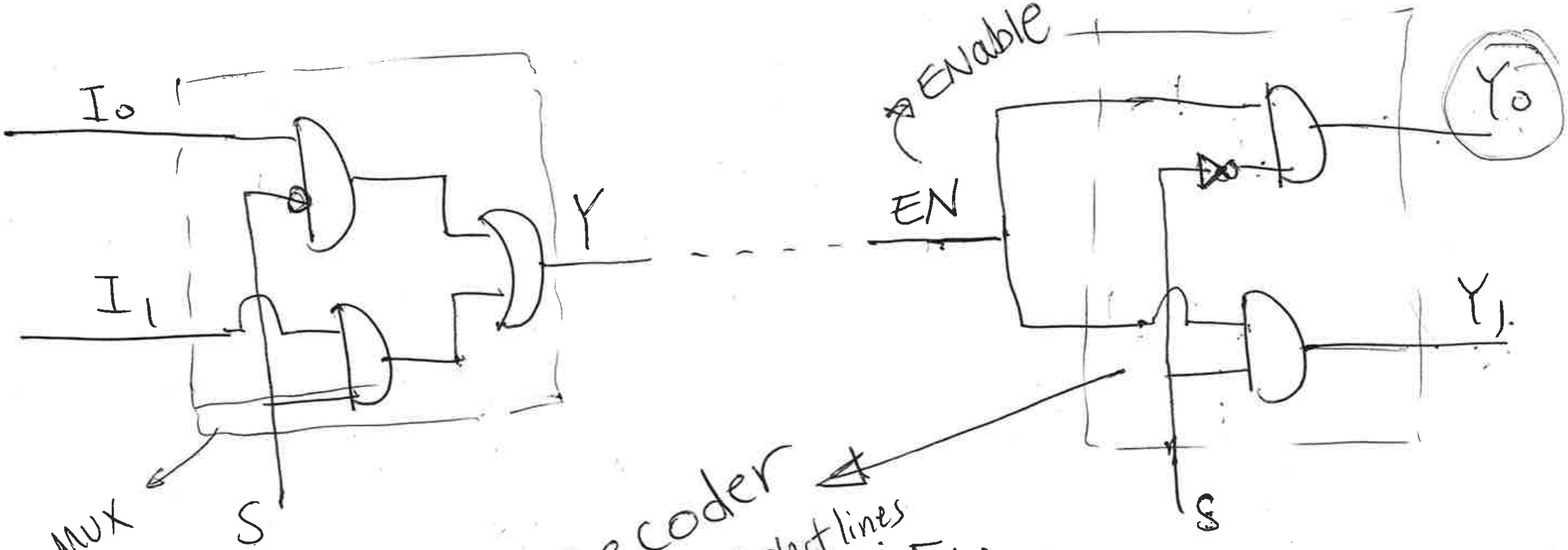
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

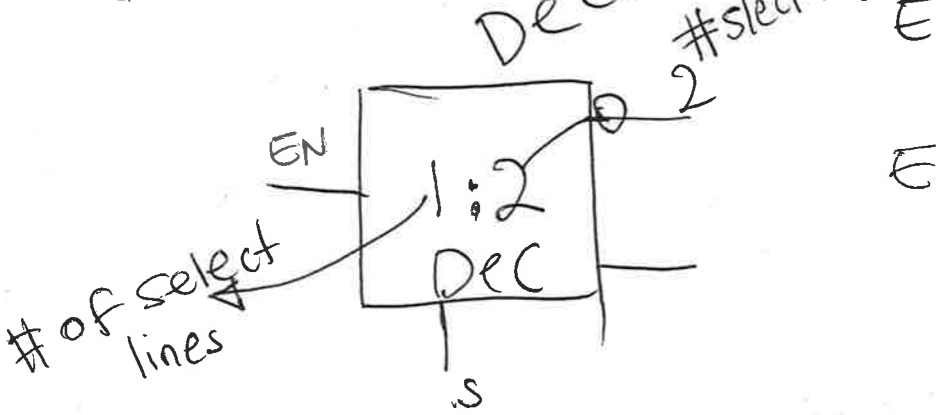
2:1 Mux

S	
0	I_0
1	I_1

Demultiplexers (Decoders)

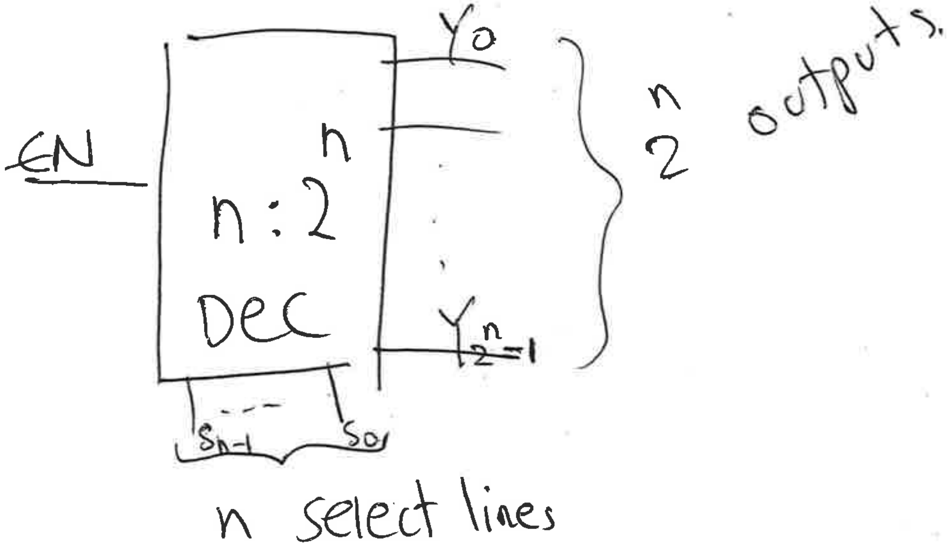
1. one input & multiple outputs
2. select line(s) determine which output is active





EN=0 ⇒ Decoder is not on

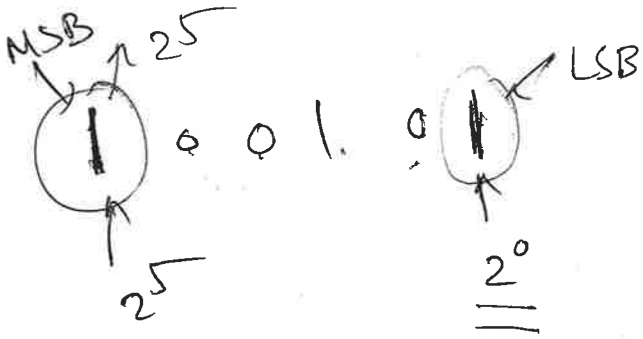
EN=1 ⇒ { S=0 ⇒ Y_0
S=1 ⇒ Y_1

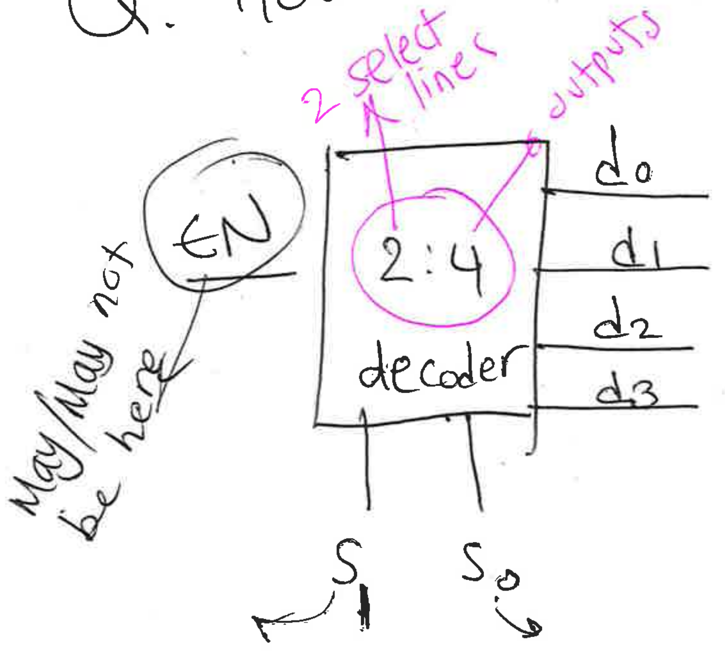


EN=1 says, we generate output.

EN=0 ⇒ output = 0

Q: How to build 2:4 decoders?



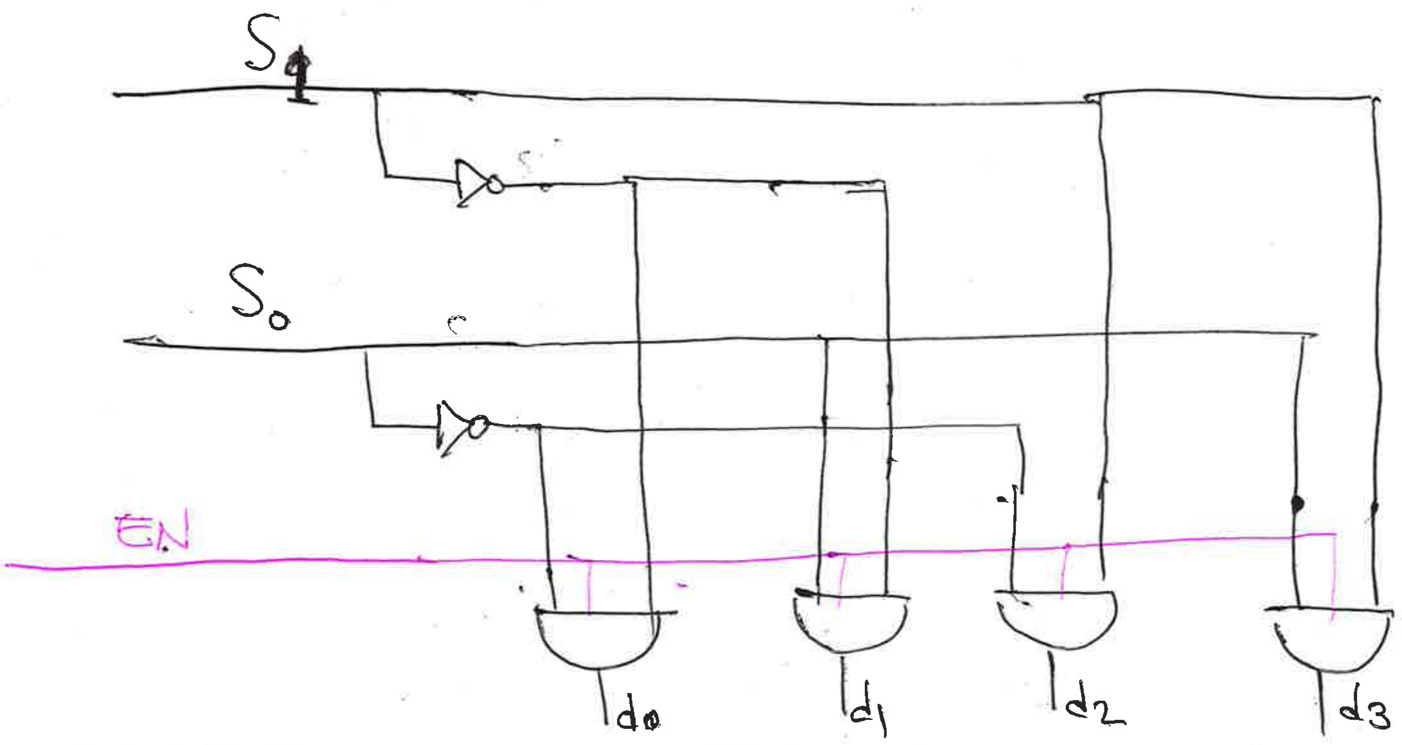


$d_0 \leftrightarrow \overline{S_1}\,\overline{S_0}$

$d_1 \leftrightarrow \overline{S_1}\,S_0$

$d_2 \leftrightarrow S_1\,\overline{S_0}$

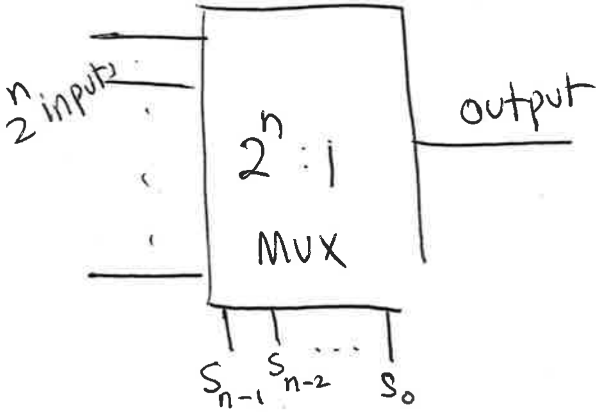
$d_3 \leftrightarrow S_1\,S_0$

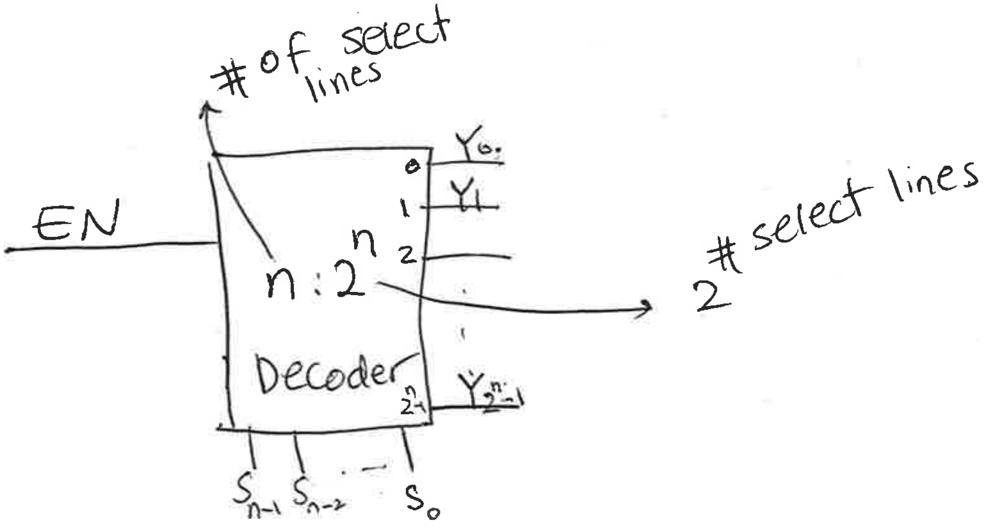


Sep 29, 2020

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

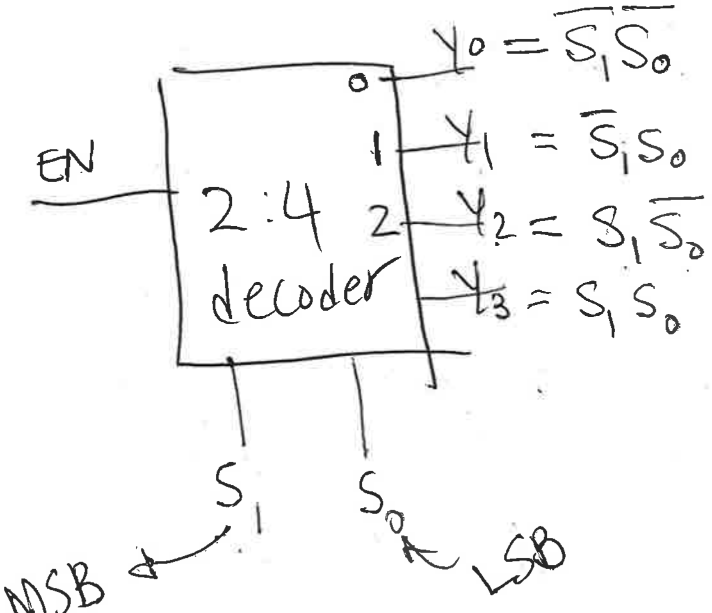
- HW3 is due Oct 1.
- Short Quiz 2 is on Oct 1.
- Lab 1 due is extended.





2^n inputs ⟺ only one output

Q: How to build a 2:4 decoder

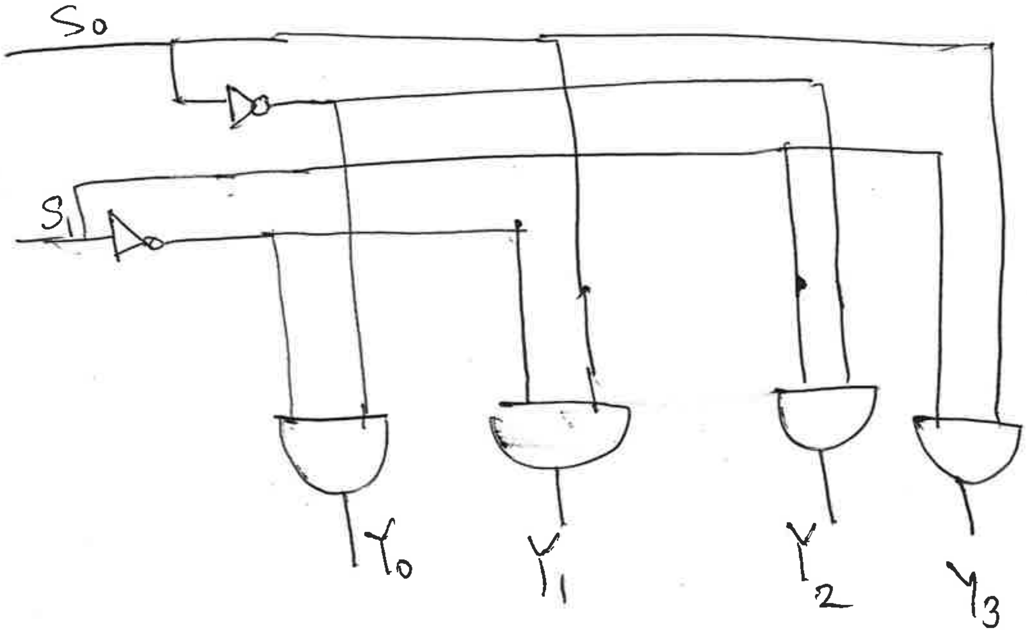


$Y_0 = \overline{S_1}\,\overline{S_0}$

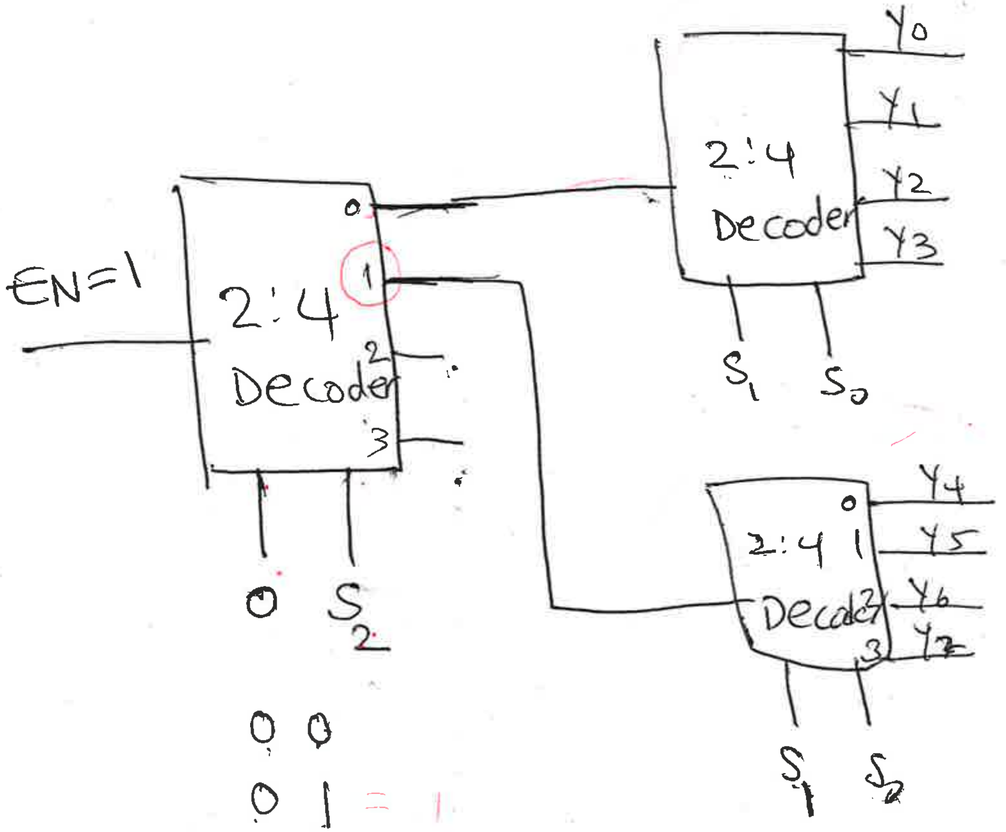
$Y_1 = \overline{S_1} S_0$

$Y_2 = S_1 \overline{S_0}$

$Y_3 = S_1 S_0$



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



	S_2	S_1	S_0	Output
0	0	0	0	Y_0
0	0	0	1	Y_1
0	0	1	0	Y_2
0	0	1	1	Y_3
0	1	0	0	Y_4
0	1	0	1	Y_5
0	1	1	0	Y_6
0	1	1	1	Y_7

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

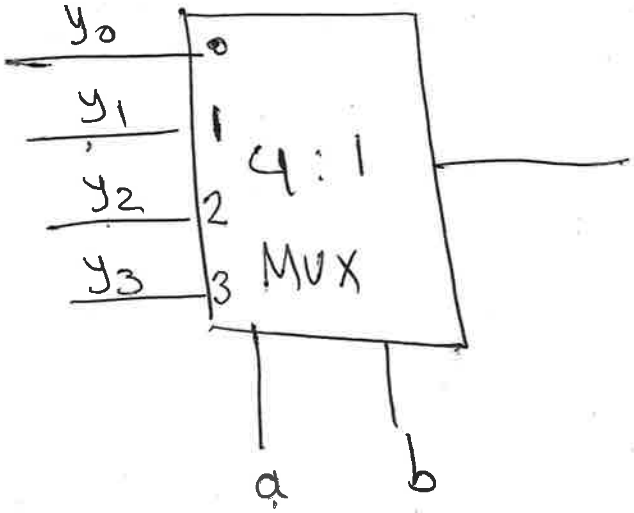
$\underbrace{0S_2} = S_2$

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

Example:

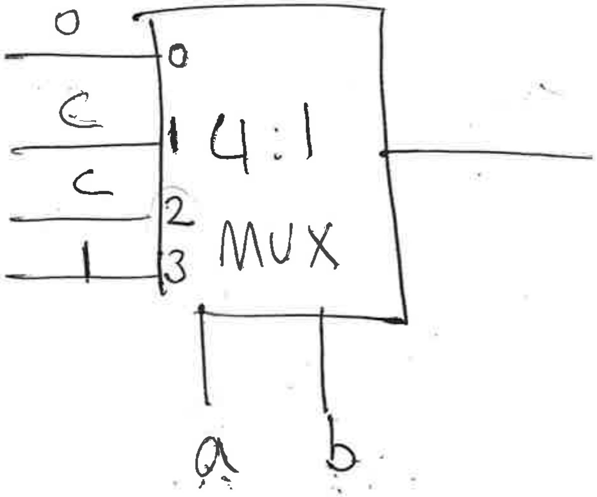
a	b	Y=output
0	0	y_0
0	1	y_1.
1	0	y_2
1	1	y_3.

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



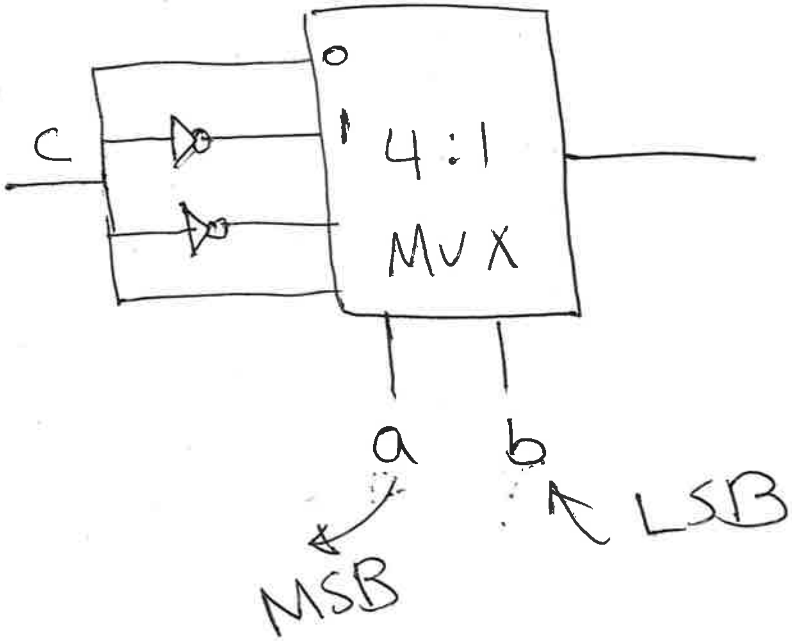
Example : 3 inputs , output = majority voting ⟹ use 4:1 Mux to build the circuit

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	1
1	1	1	1	1

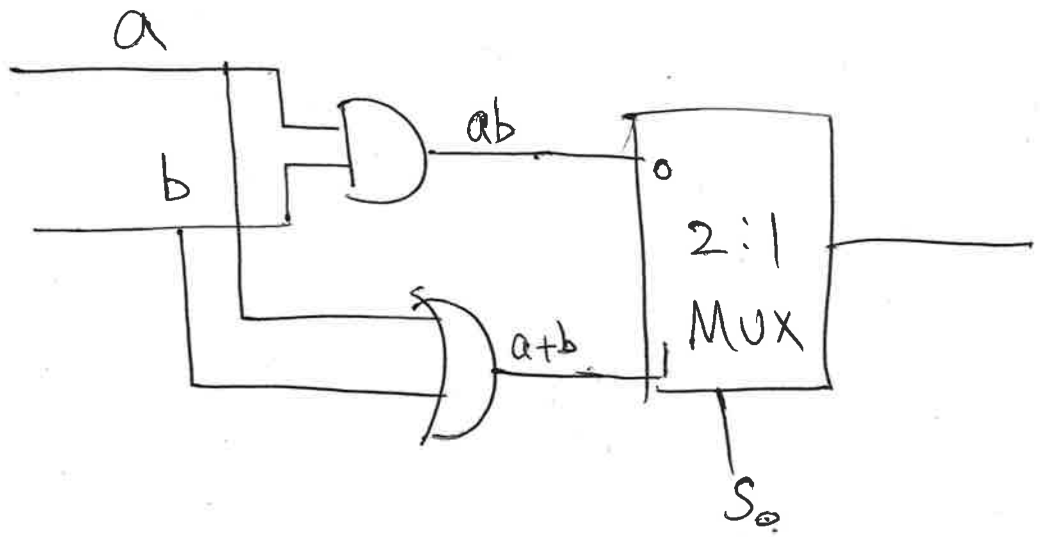


Example : (parity)

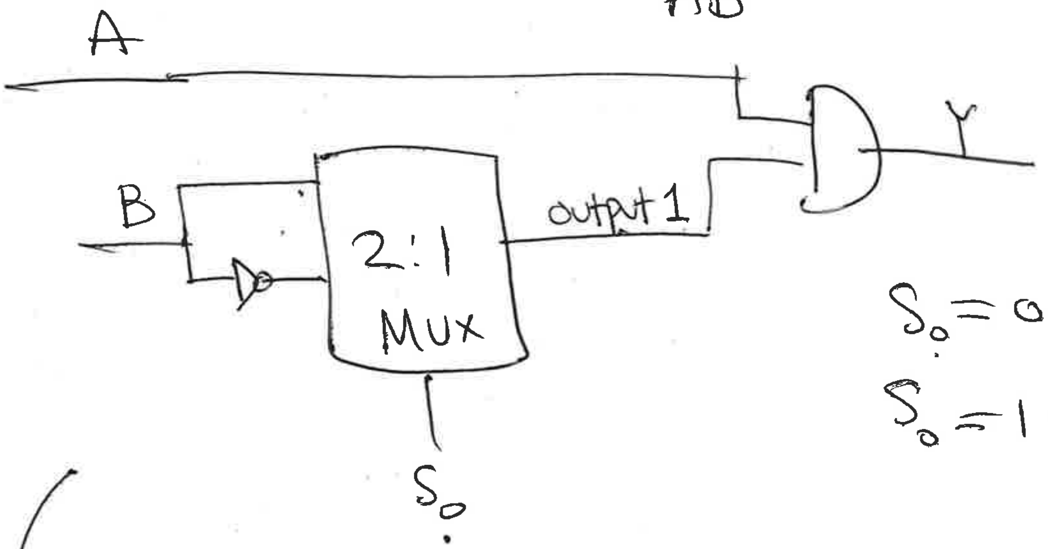
a	b	c	Y=output	
0	0	0	0	c
0	0	1	1	c
0	1	0	1	$\overline{c}$
0	1	1	0	$\overline{c}$
1	0	0	1	$\overline{c}$
1	0	1	0	$\overline{c}$
1	1	0	0	c
1	1	1	1	c



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 $\underbrace{\text{"A and B"}}_{AB}$ or $\underbrace{\text{"A and not B"}}_{A\overline{B}}$.



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

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

→ we build ALU

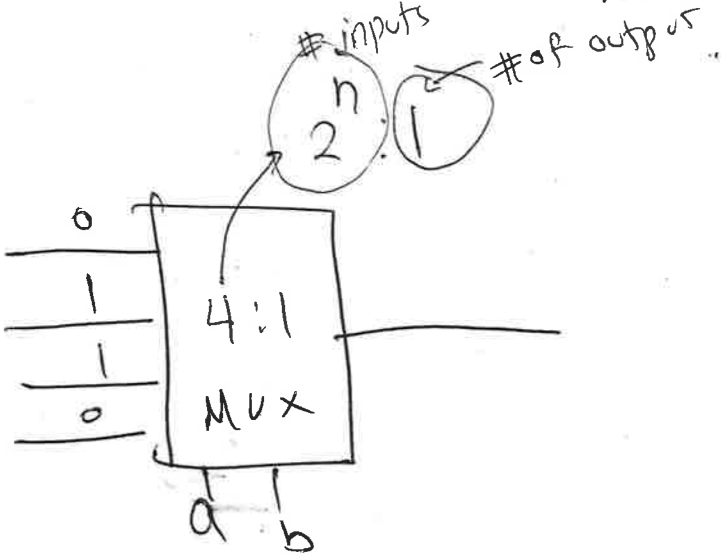
→ Arithmitic logic unit

→ carry out a range of calculations on its input.

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

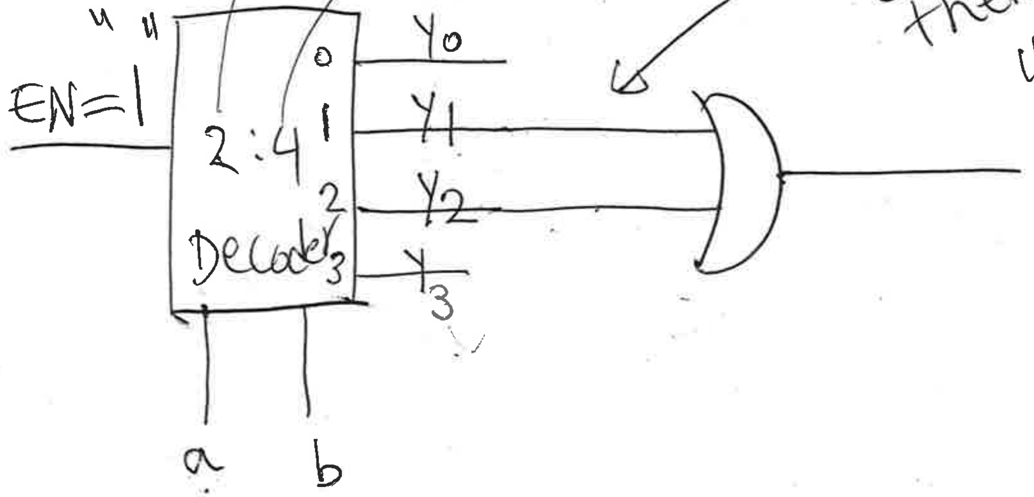
	a	b	Y = "XOR"
0	0	0	0
1	0	1	1 ← $\bar{a}b$
2	1	0	1 ← $a\bar{b}$
3	1	1	0

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



# of select lines

#select lines 2



wire the ones and input them to an "OR" gate.

Oct 1, 2020

Lecture 13 : Programmable logic devices and Tri-state Buffer

programmable logic devices (PLD)

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

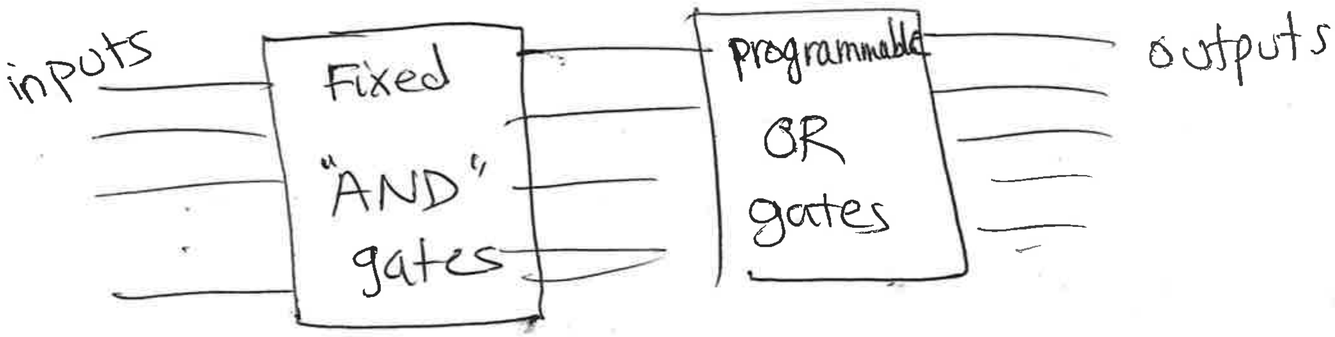
Programming : entring info into these devices is Called Programmig

X : Indicate programmig

Types of PLD :

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

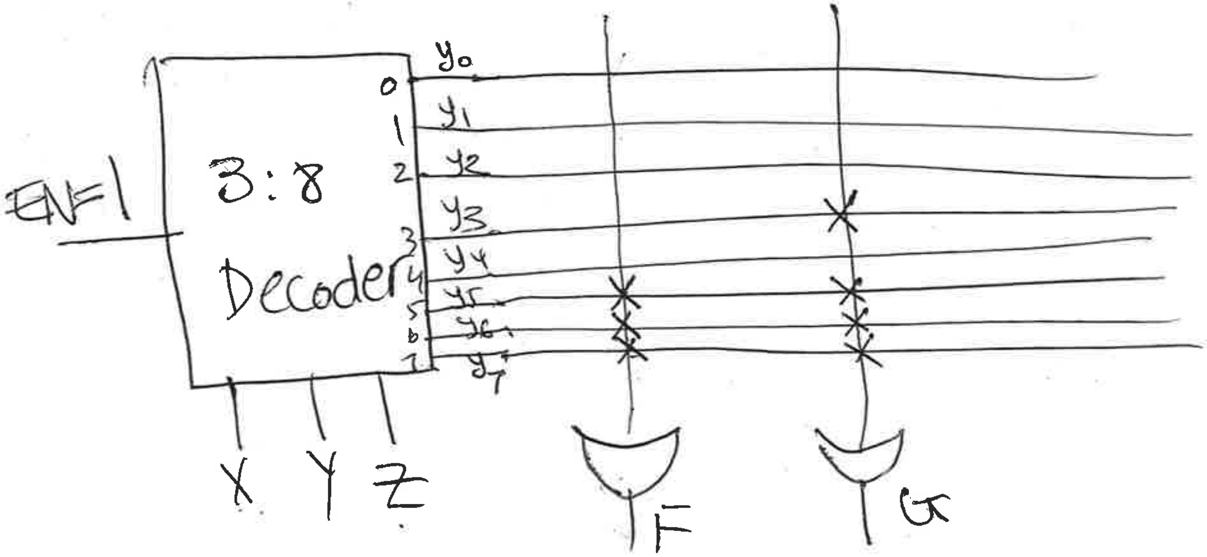
1 PROM :



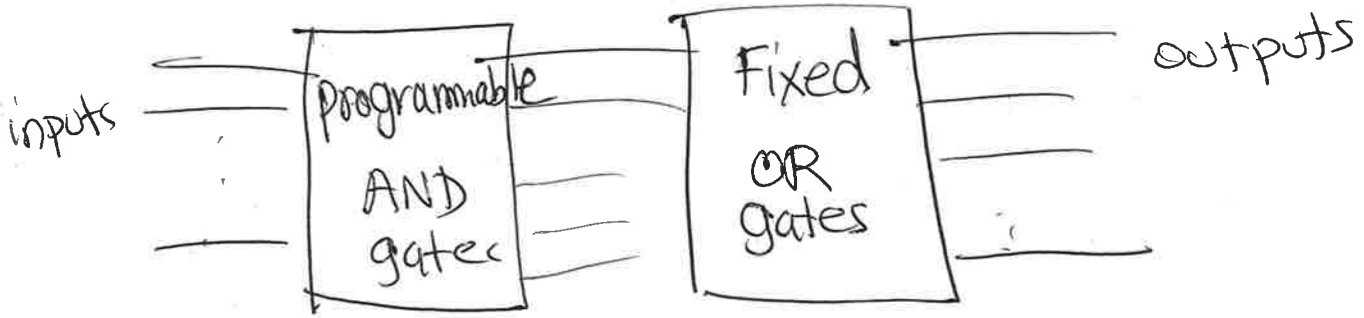
Example : Implement the following outputs using PROM

$$F(X,Y,Z) = \sum m(5,6,7)$$

$$G(X,Y,Z) = \sum m(3,5,6,7)$$



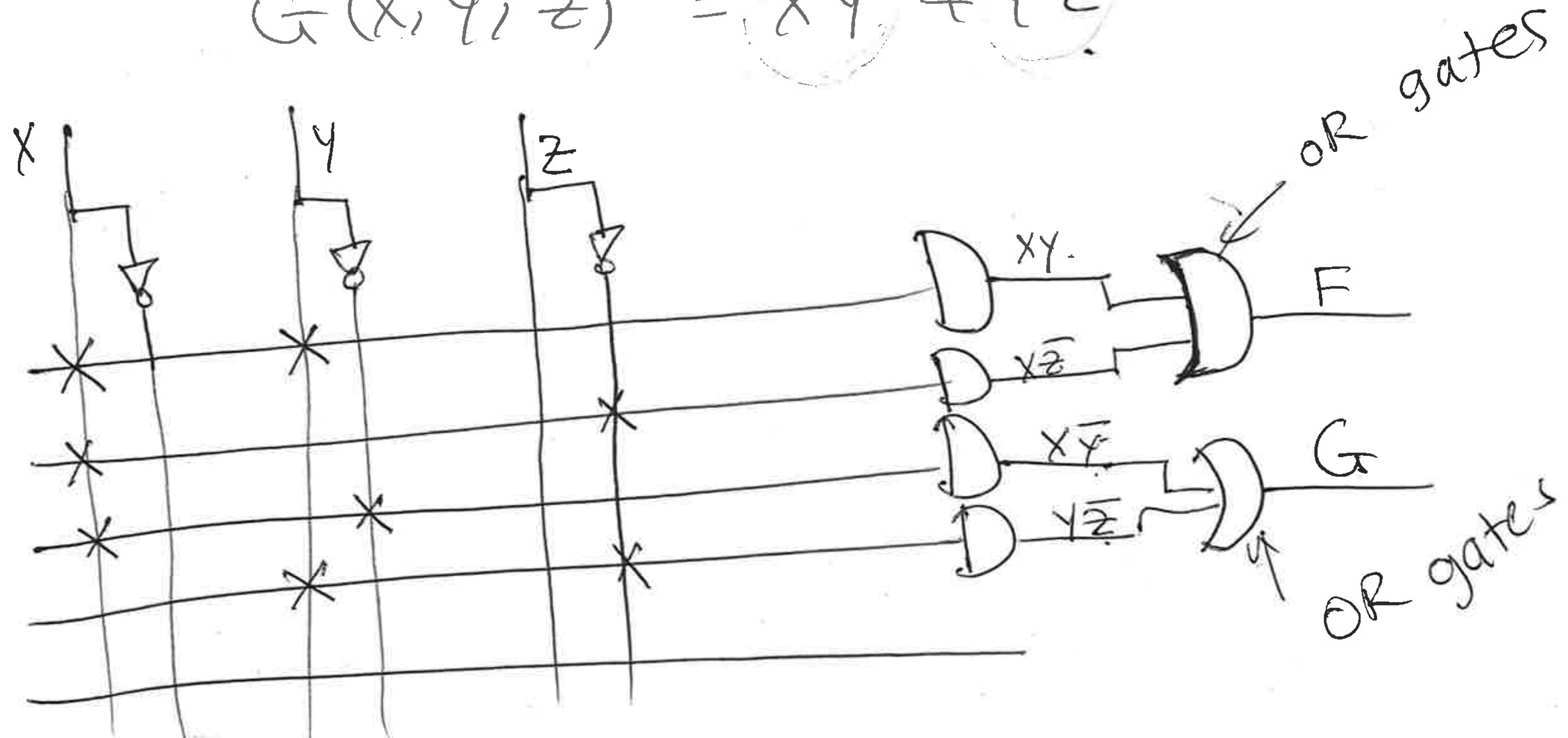
$\boxed{2}$ PAL



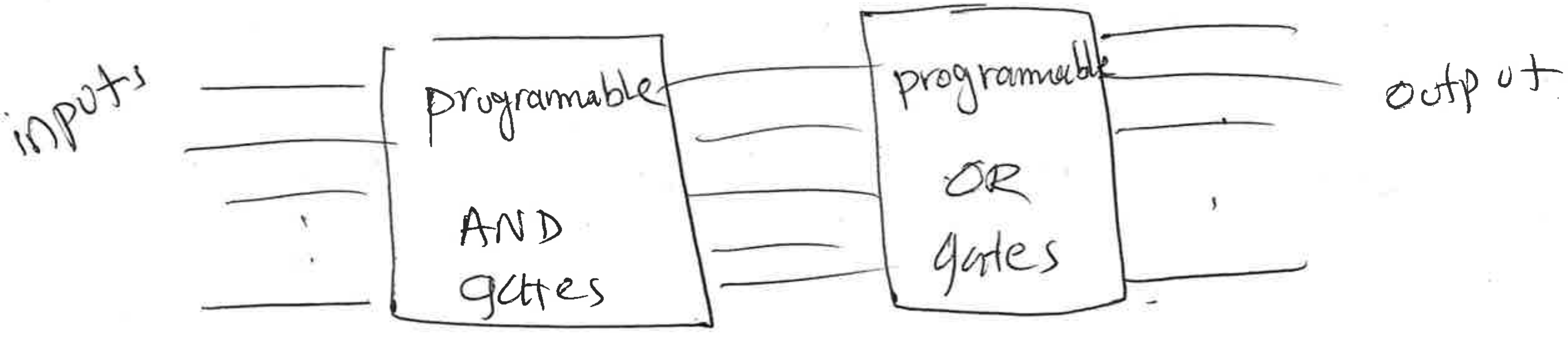
<u>Example</u>: Build the following functions using PAL.

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

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

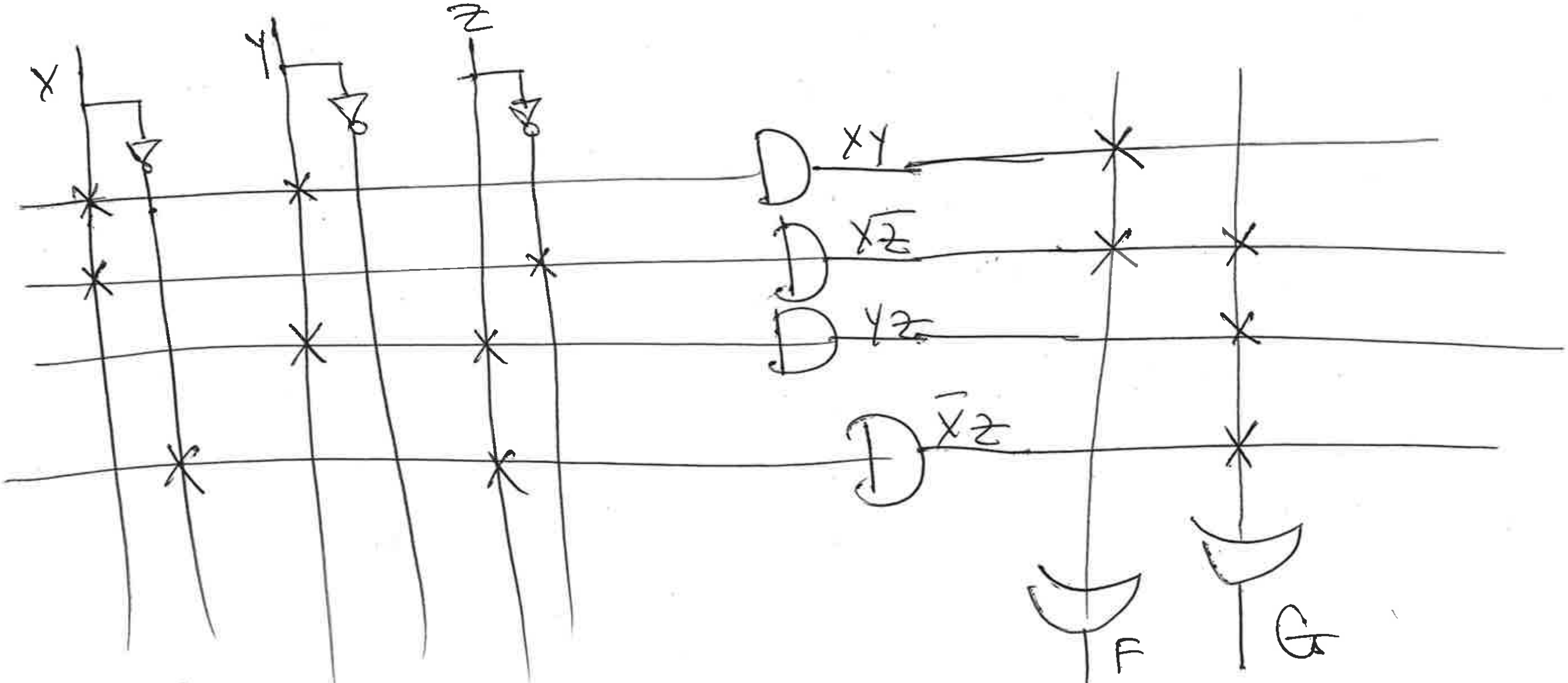


$\boxed{3}$ PLA

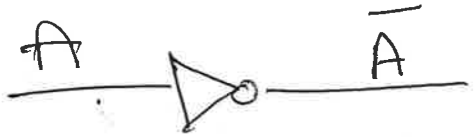


Example : $F(X,Y,Z) = XY + X\overline{Z}$

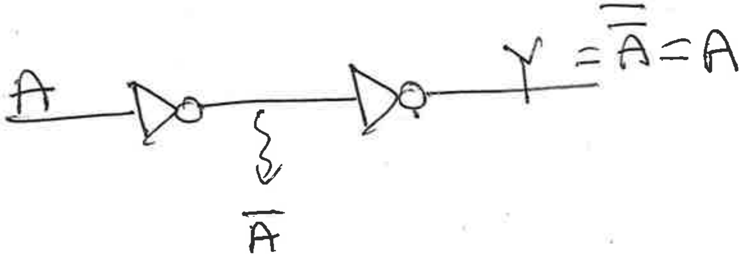
$G(X,Y,Z) = X\overline{Z} + YZ + \overline{X}Z$



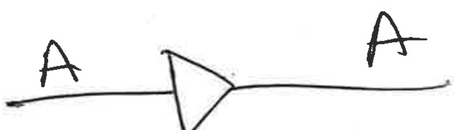
Tri-State Buffer:



A	Y = output
0	1
1	0

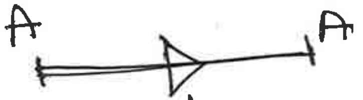


$\Rightarrow$

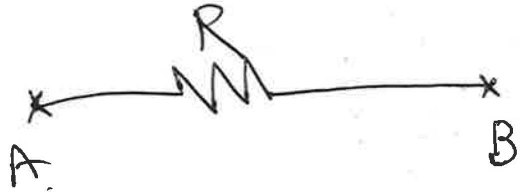


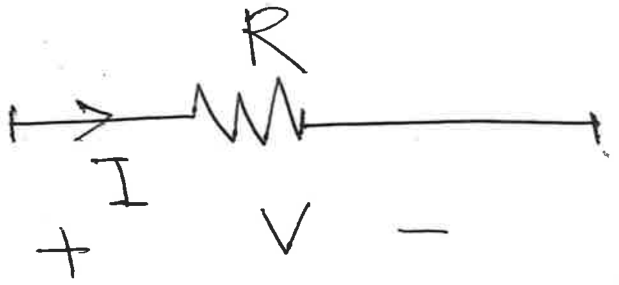
Buffer

A	Y
0	0
1	1

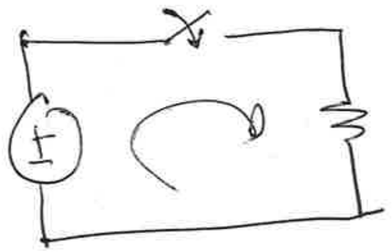


digital amplification.





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



switch open $\Rightarrow I = 0$

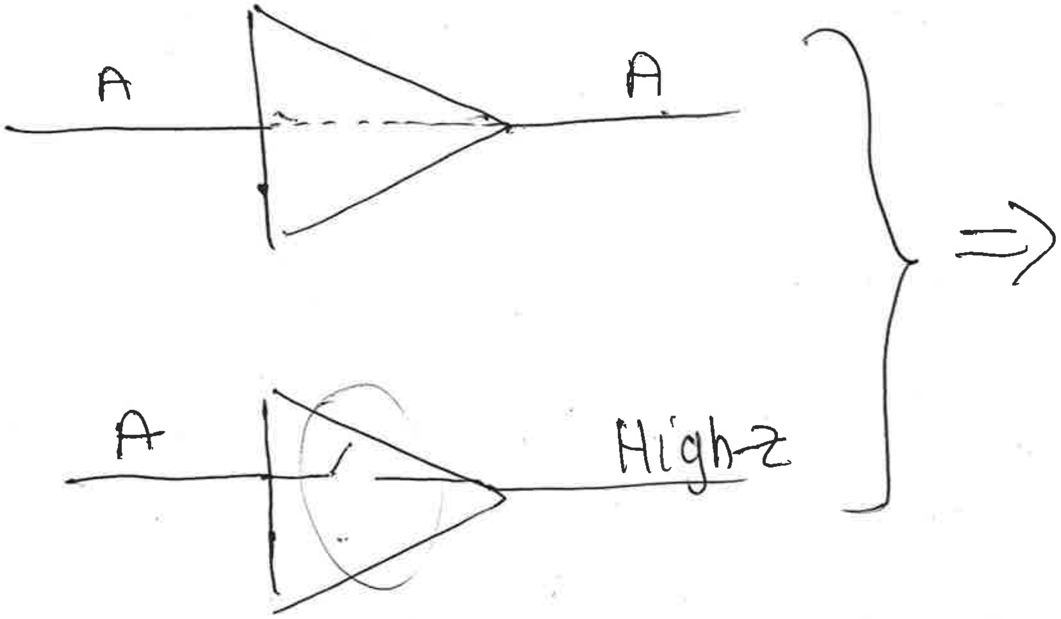
$$R = \frac{V}{0} = +\infty$$

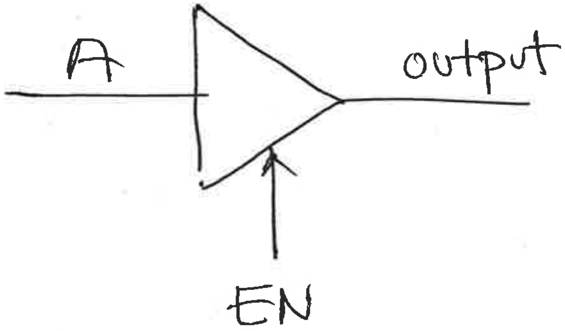
$$\text{O.C.} \Rightarrow \underbrace{R = \infty}_{\text{high } R \text{ "}\Omega\text{"}}$$

high-z ← impedence

Tri-state Buffer :

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





EN = 1 ⇒ output = A

EN = 0 ⇒ output = High z

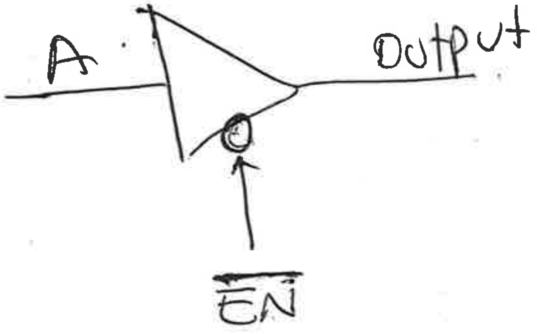
EN	A	output
0	0	High-z
0	1	High z
1	0	0
1	1	1

"Active-high" Tri-state Buffer

Active low Tri-state Buffer

EN=0 ⇒ output = A

EN=1 ⇒ ouput = High-Z

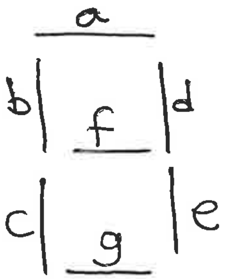


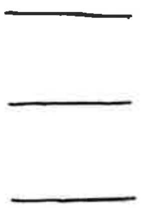
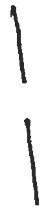
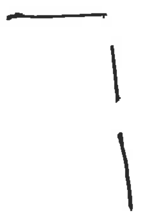
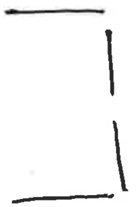
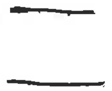
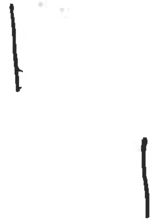
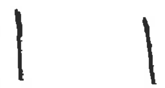
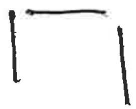
EN	A	output
0	0	0
0	1	1
1	0	High-Z
1	1	High-Z

Short Quiz 2 g

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) $f(x,y,z) = \sum m(0,2,3,7)$

$Z \backslash XY$	00	01	11	10
0	1	1		
1		1	1	

$\bar{X}\bar{Z}$

YZ

$f(x,y,z) = \bar{X}\bar{Z} + YZ$

(B) $g(x,y,z) = \sum m(0,4)$

$Z \backslash XY$	00	01	11	10
0	1			1
1				

$\bar{Y}\bar{Z}$

$g(x,y,z) = \bar{Y}\bar{Z}$

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 ⇒ output

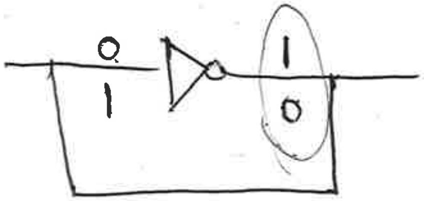
↳ Store the logic value!

- ability to program storage module

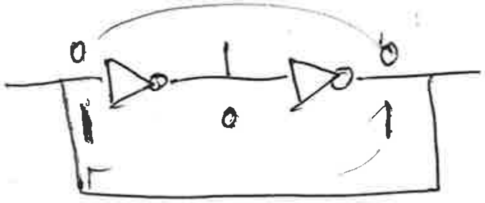
- ability to read the stored value.

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

☐1 First attempt:

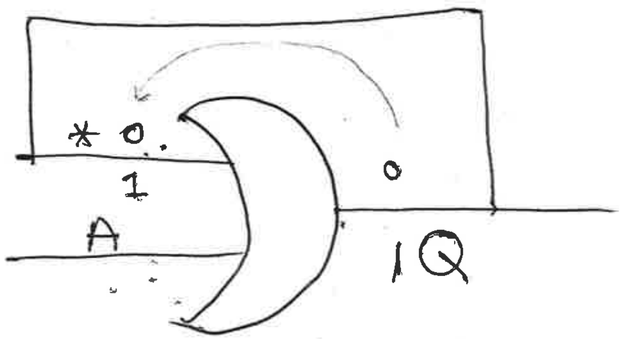


oscillates between "0" & "1"
⇒ unstable



⇒ Stable

$\boxed{2}$ Second attempt



$\underline{A=0}$: $* = 0 \Rightarrow Q = 0$

$* = 1 \Rightarrow Q = 1$

bi-stable

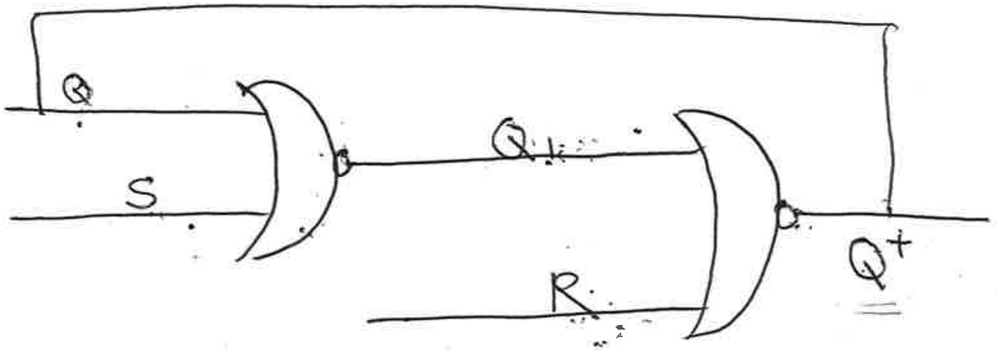
$\underline{A=1}$: $* = 0 \Rightarrow Q = 1$

$* = 1 \Rightarrow Q = 1$

output switches to 1

⇒ cannot program "0".

3 Third Attempt



Set $S=1$:

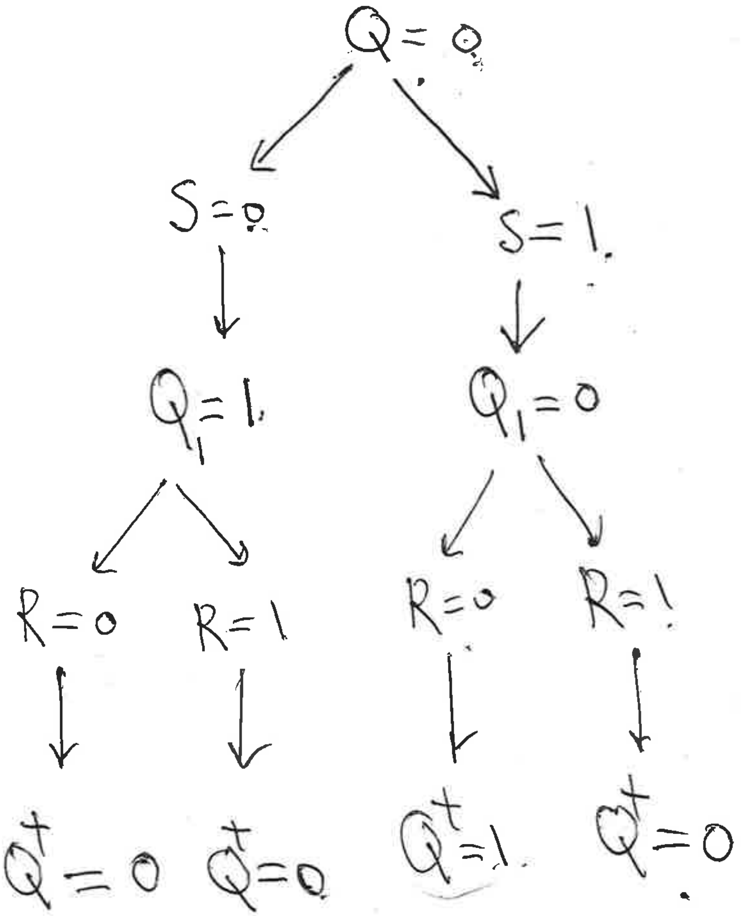
$Q_1 = 0$

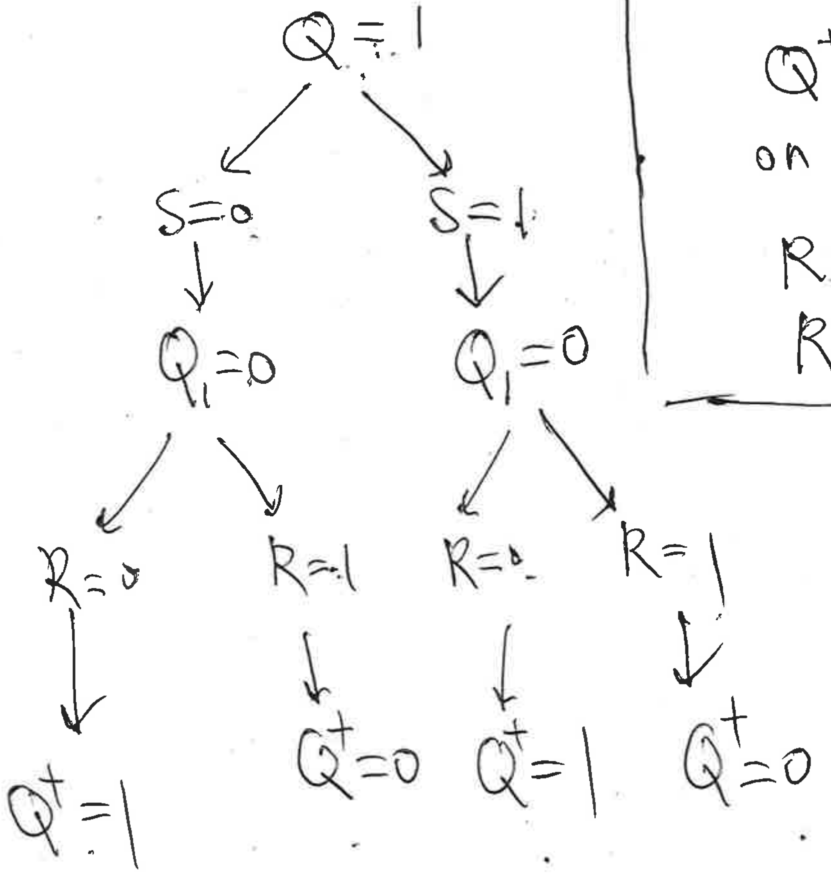
⇓

Q^+ only depends on R

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

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

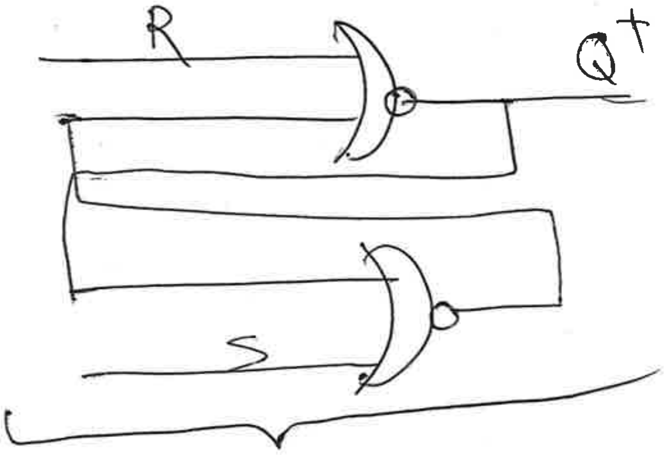




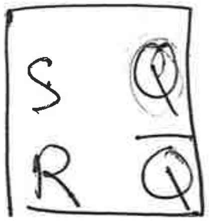
S	R	Q	Q^+ ← Qnext	
0	0	0	0	bi-stability => Storage part
0	0	1	1	
0	1	0	0	← Program "0"
0	1	1	0	
1	0	0	1	← Program "1"
1	0	1	1	
1	1	0	0	Exclude R=1, S=1
1	1	1	0	

State transition table

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



SR latch

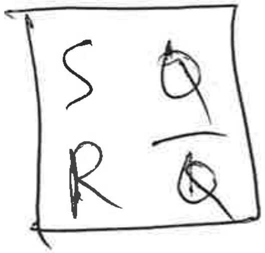


SR_latch

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

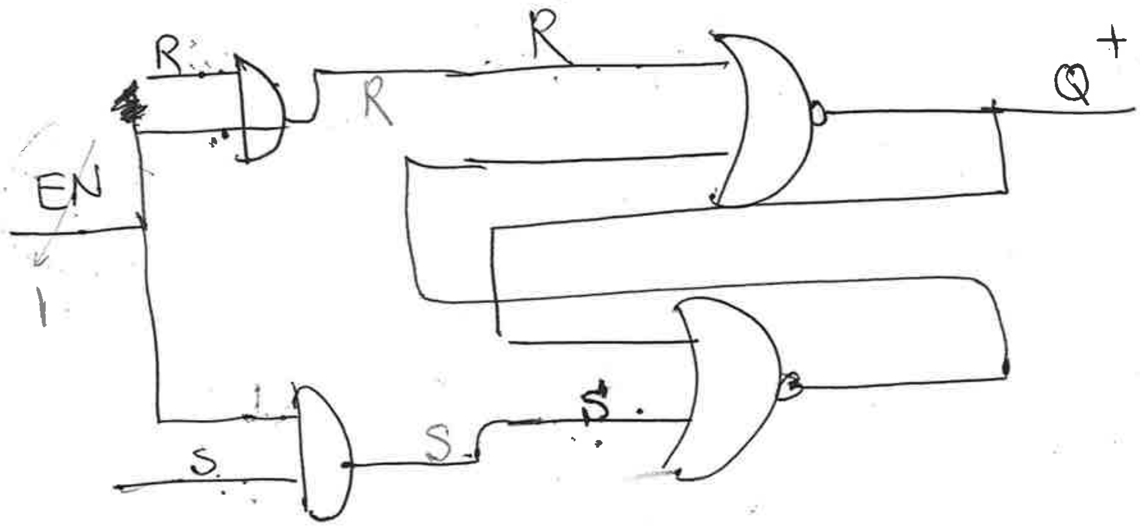
Observations :

- If $S = R = 0$ ⇒ stable states
- If switch S to 1 (temporarily) ⇒ $Q^+ \rightarrow 1$
  If we switch S back to zero ⇒ Q^+ stays at 1 } ⇒ "Set"
- If switch R to 1 (temprorily) ⇒ $Q^+ = 0$
  switch back to 1 ⇒ Q^+ stays at " " } ⇒ Reset



SR- latch

SR latches w/ enable :



$EN = 1 \Rightarrow$ SR-latch

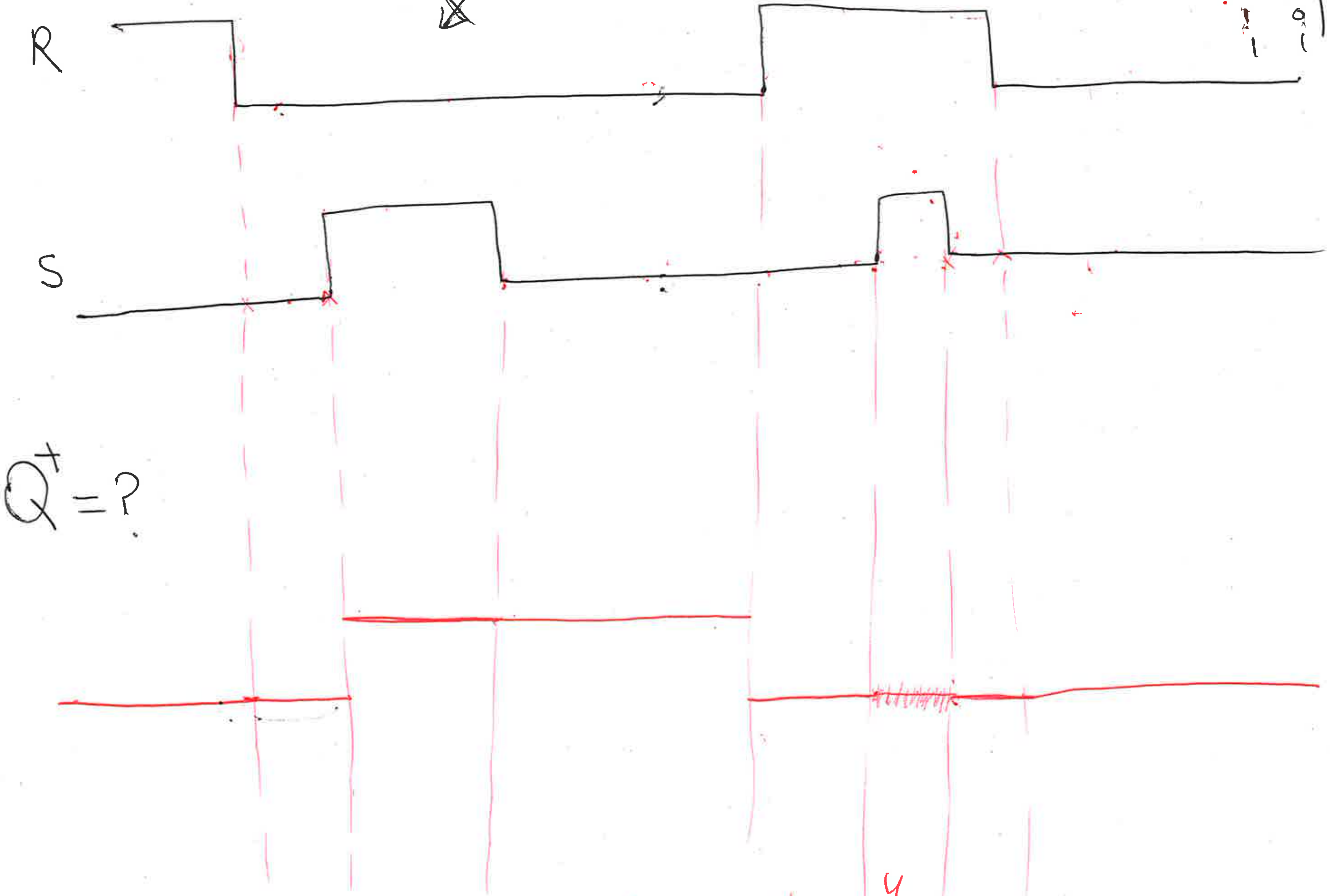
EN=0 : storage mode

EN	S	R	Q^+
0	X	X	storage mode
1	0	0	storage
1	0	1	0
1	1	0	1
1	1	1	X Forbidden.

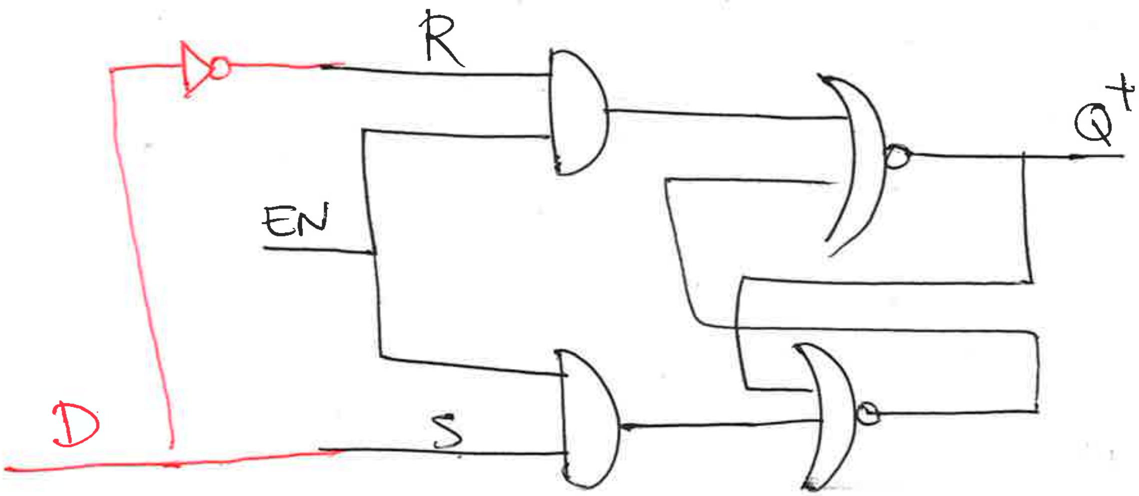
Example:

Timing diagrams

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



"level sensitive"



EN	S	R	Q^+
0	X	X	Storage "Q"
1	0	0	Q (storage)
1	0	1	0
1	1	0	1
1	1	1	X

EN	D	Q^+
0	x	Storage
1	0	0
1	1	1

=>

EN	D	Q^+
0	X	storage
1	D	D

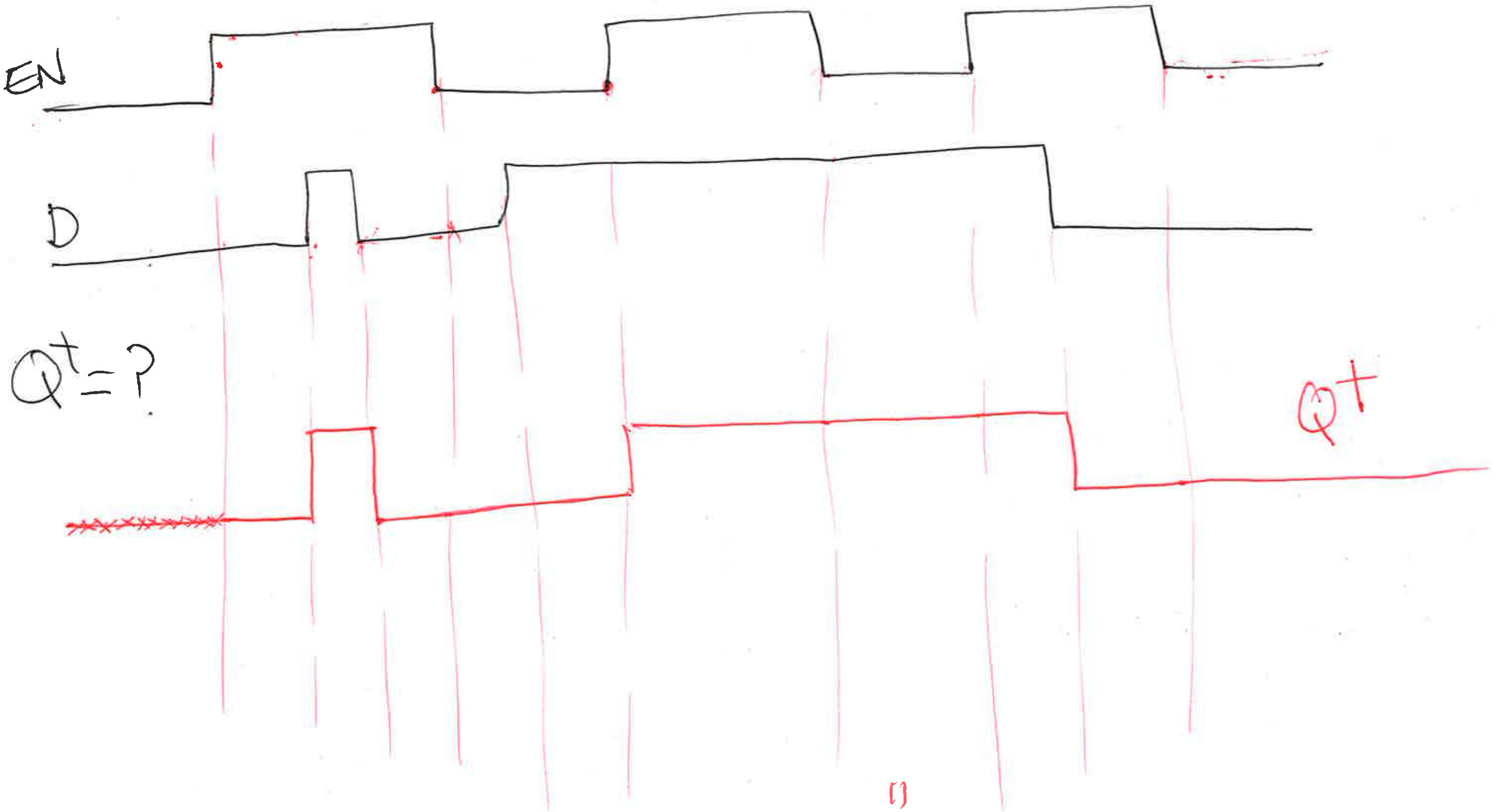
state transition table

D-latch

$$\boxed{Q^+ = D}$$

behavioral eq.

Example:



"level sensitive"

Lecture 15 : Flip Flops

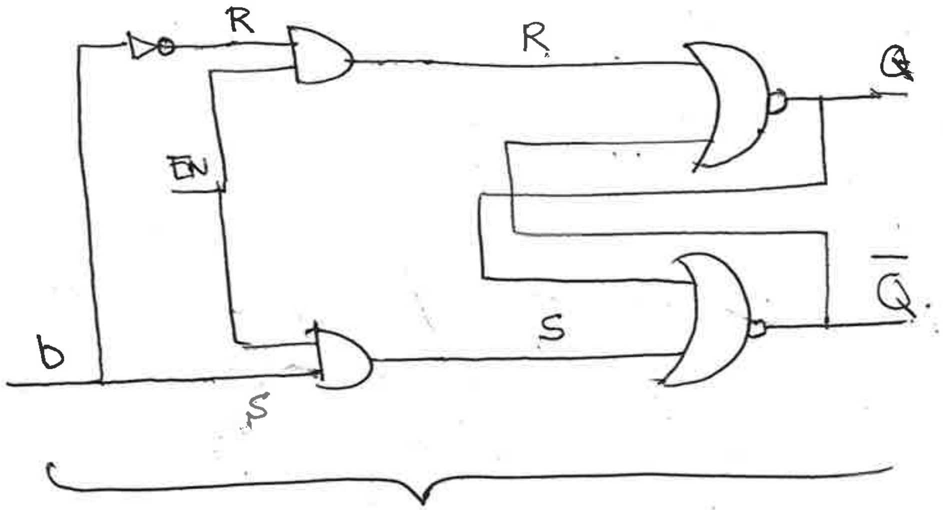
oct 8, 2020

- Lab office hours next week :

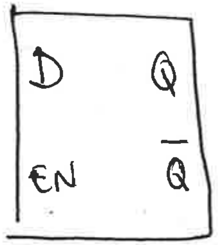
Monday	5-6 pm
Wednesday	12-1:30 pm
Friday	2:30- 4:30pm

- Make sure you have lockdown browser

Last time:



D - latch



EN=1 ⇒ Active high
high ~~D~~-latch

EN=0 ⇒ low D-latch

Next state

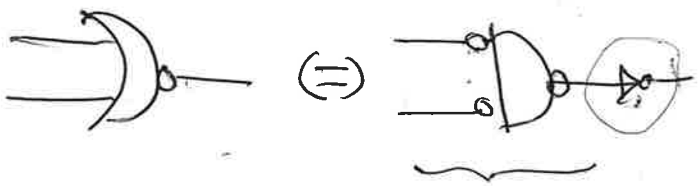
EN	D	Q^+
0	X	storage Q
1	0	0
1	1	1

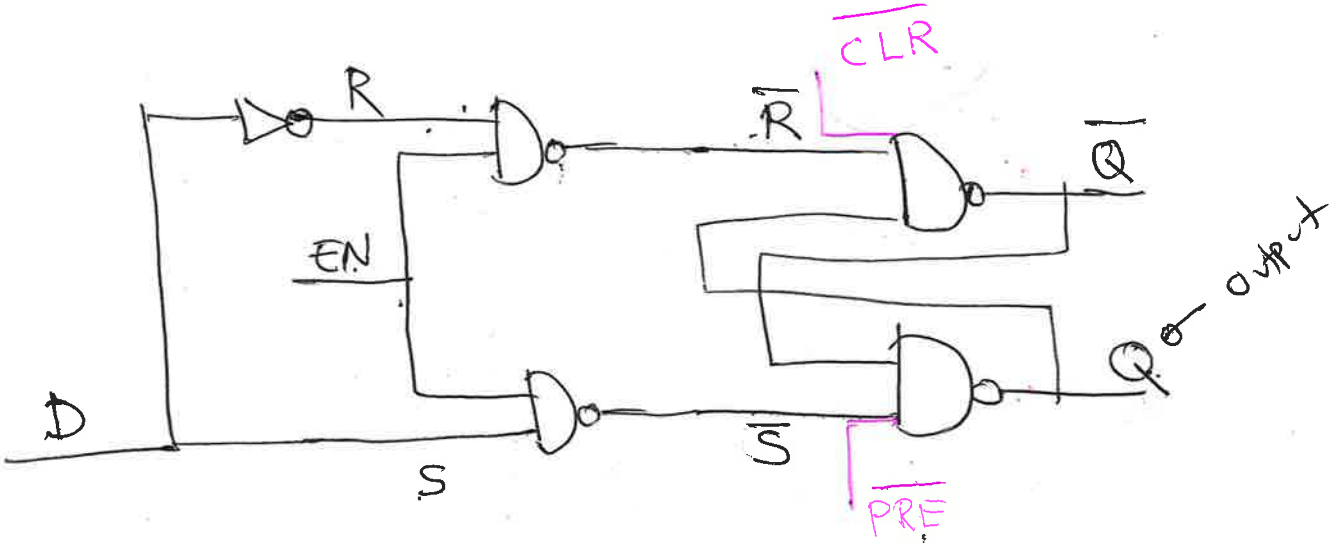
D D

high D-latch

$$\Rightarrow \underbrace{Q^+ = D}_{\text{behavioral eq.}}$$

D-latch using "NAND" gates





EN=1 ⇒ latch working

EN=0 ⇒ Storage mode

EN	D	Q^+
0	x	Q
1	D	D

$\overline{CLR}=0$ ⇒ $\overline{Q}=1$

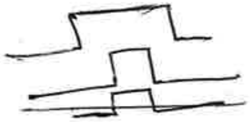
⇕ ⇓

CLR=1 $\boxed{Q=0}$

$\overline{PRE}=0$ ⇒ $\boxed{Q=1}$

⇕

PRE=1

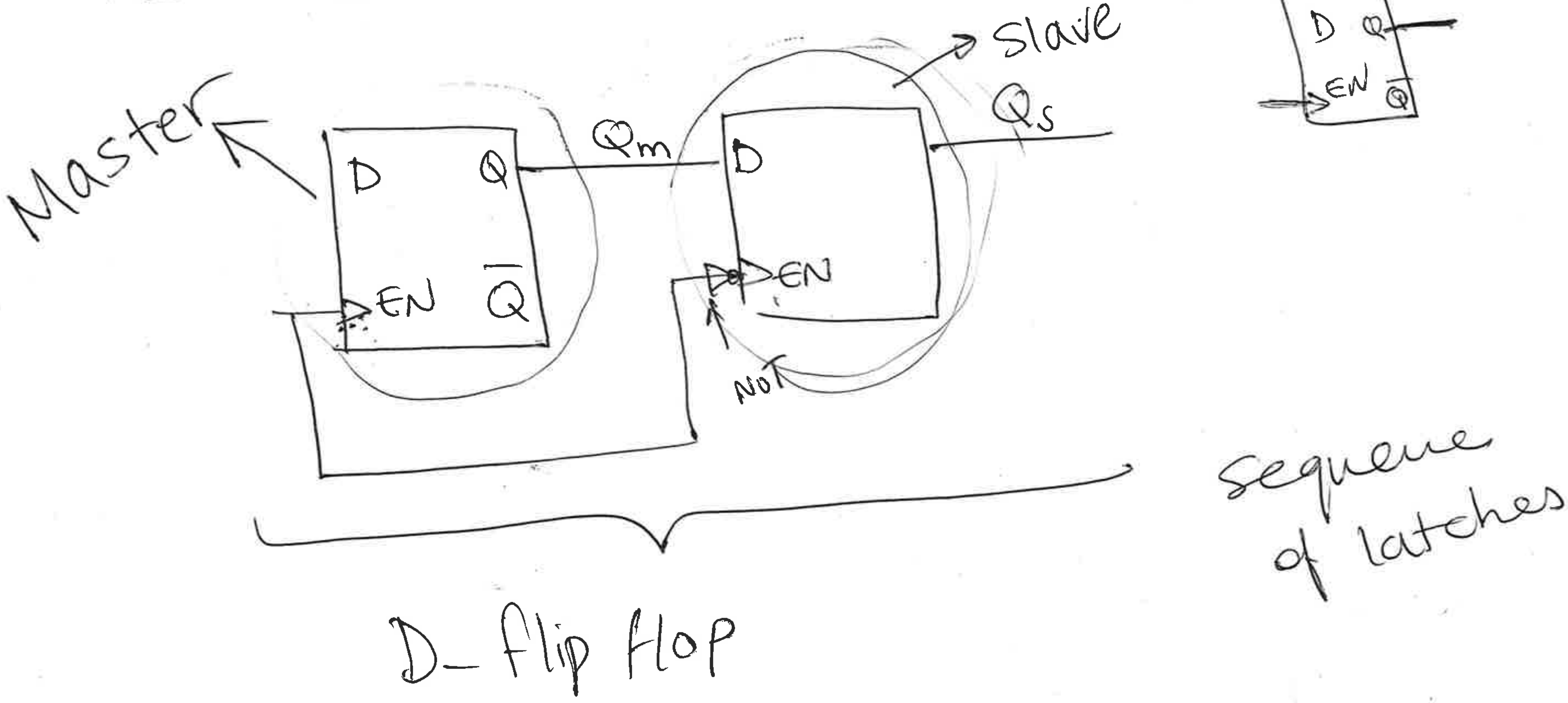


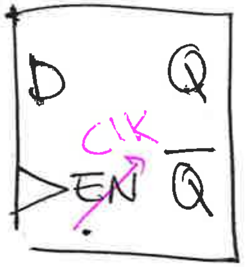
D-latches are level sensitive ⇒ output will be proprogrammed "EN=1"

Transparency.

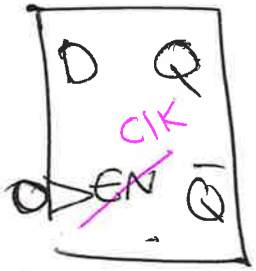
Q: Can we design an "EN" that only enables when signal changes $\underbrace{0 \rightarrow 1}_{\downarrow \text{ positive edge}}$ or $\underbrace{1 \rightarrow 0}_{\text{Negative edge}}$

How do we do this ?





positive edge – D-flip-flop



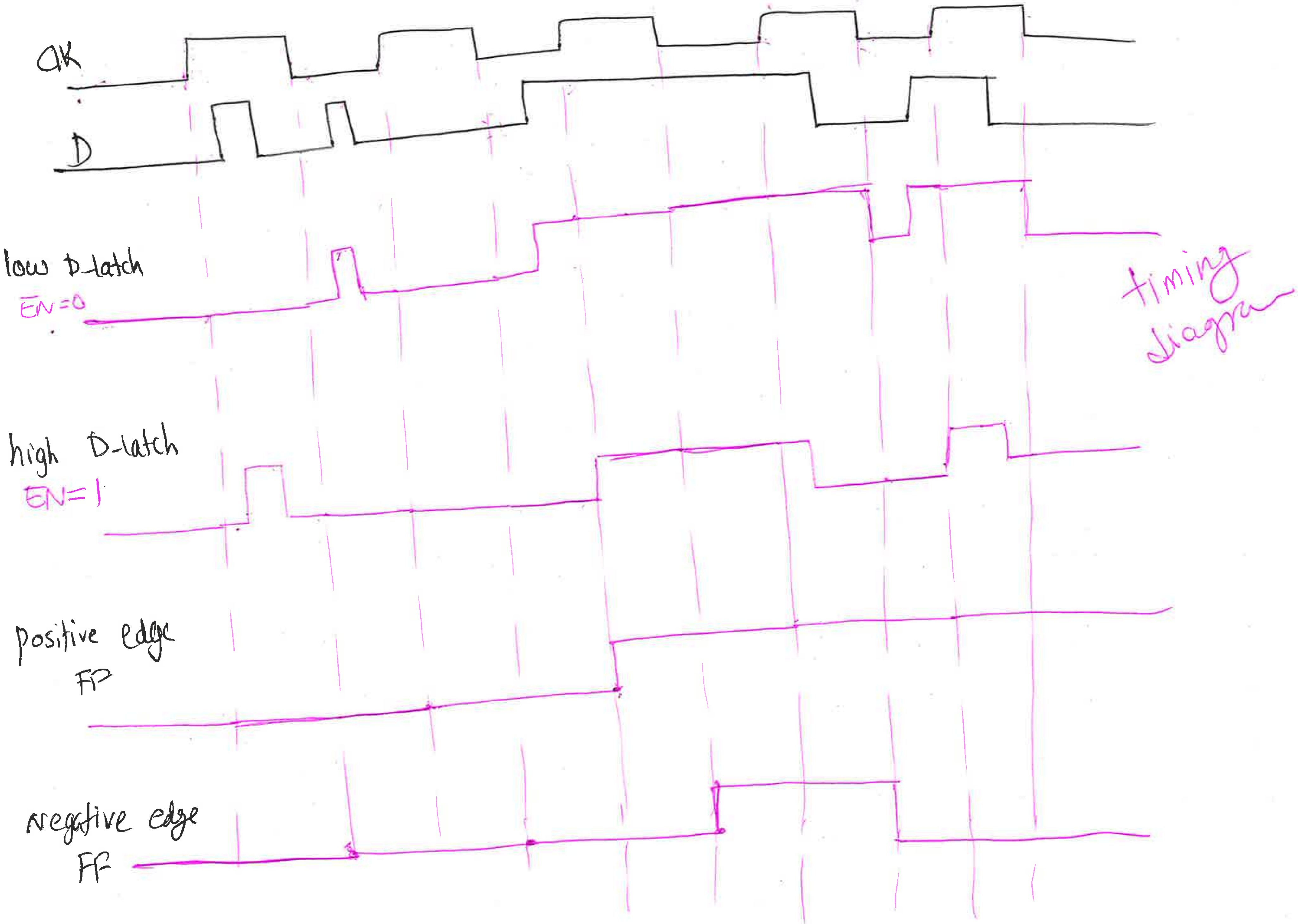
negative edge – D-flip flop

$$Q^{+} = D$$

as long as ~~neg~~ rising edge (0 → 1)

EN	D	
0	x	storage
1	D	D

Example: D-flip flop



Other types of FFs

① JK flip flop

J	K	Q^+
0	0	Storage mode "Q"
0	1	0
1	0	1
1	1	$\overline{Q}$

Q^+ ← Next state

State transition table

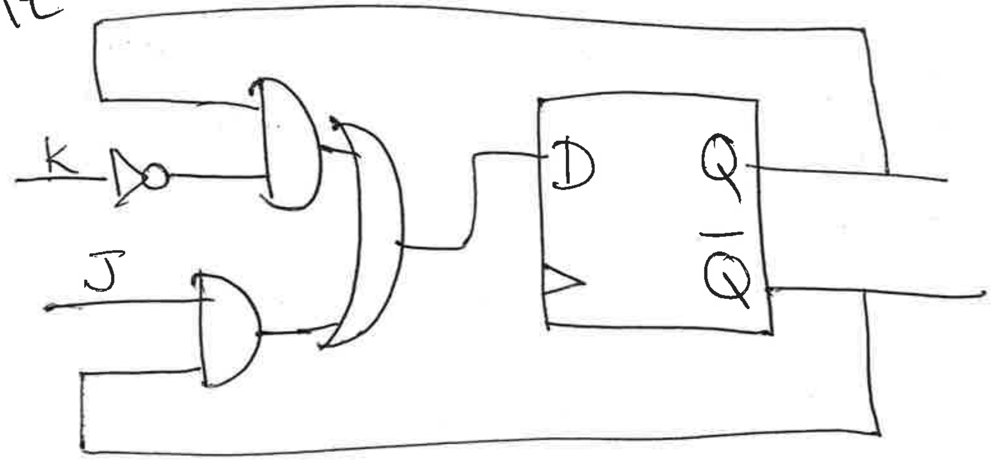
K-MAP for Q^+

Q \ JK	00	01	11	10
0			1	1
1	1			1

$J\overline{Q}$

$\overline{K}Q$

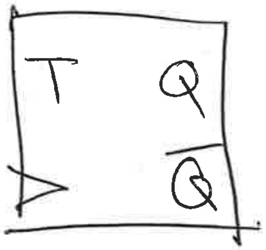
$$\boxed{Q^+ = J\overline{Q} + \overline{K}Q}$$



JK flip-flop

② Toggle flip flop

T	Q^+
0	Q
1	$\bar{Q}$

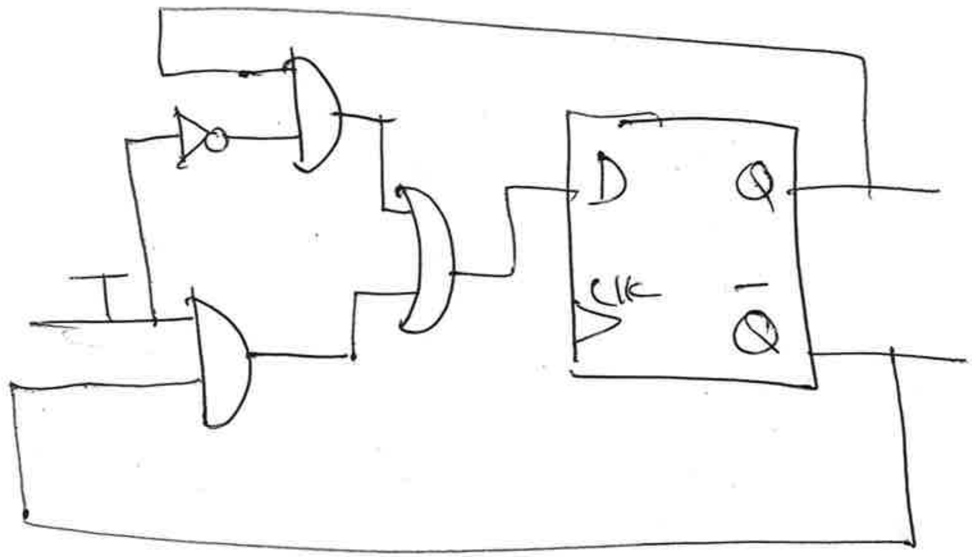


Q \ T	0	1
0		1
1	1	

$T\bar{Q}$

$\bar{T}Q$

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



Toggle flip flop.

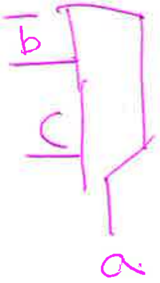
Example :

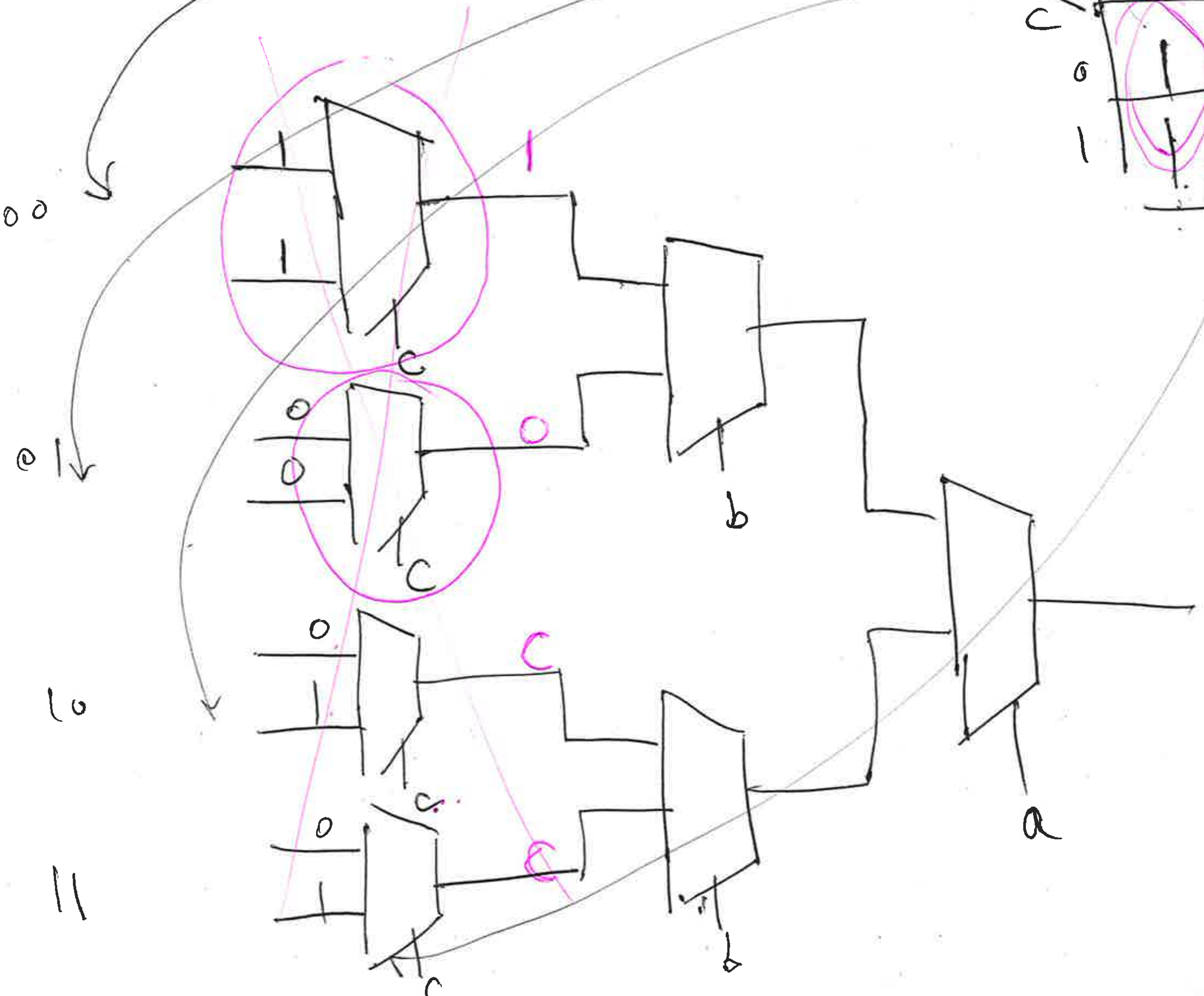
$f(a,b,c) = \sum m(0,1,5,7)$

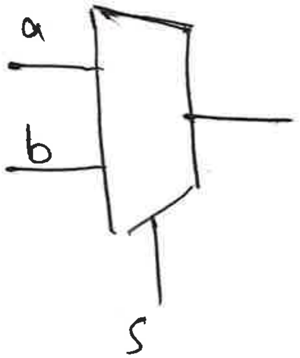
Using MUX to build this function.

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

$\bar{a}\bar{b}+ac$

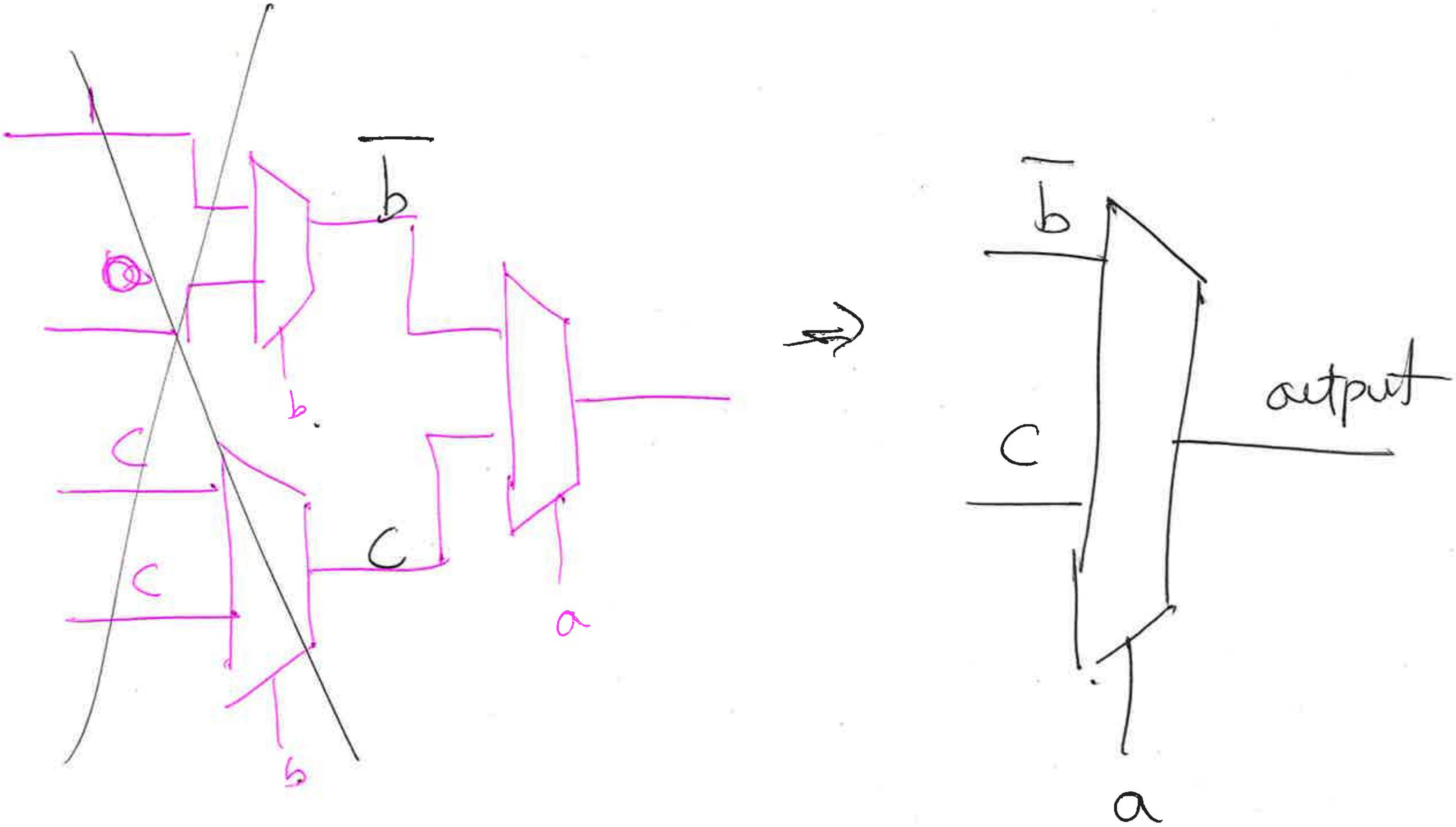






$s=0 \Rightarrow Y=a$

$s=1 \Rightarrow Y=b$


1
0
$\bar{b}$
b
c
c
c
b
a
$\bar{b}$
c
output
a

oct 13, 2020

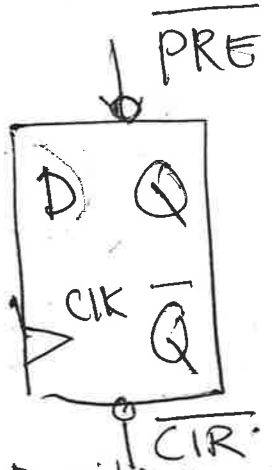
EEE/CSE 120 : Registers

- 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 :

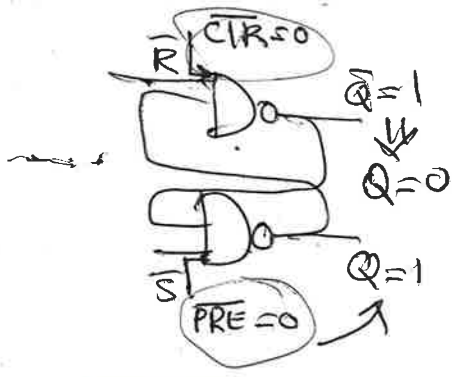


positive edge FF

0 → 1

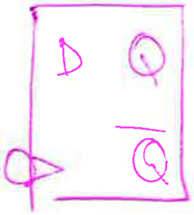
$\overline{ClR} = 0 \Rightarrow Q = 0$

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



$$Q^+ = D$$

Next State

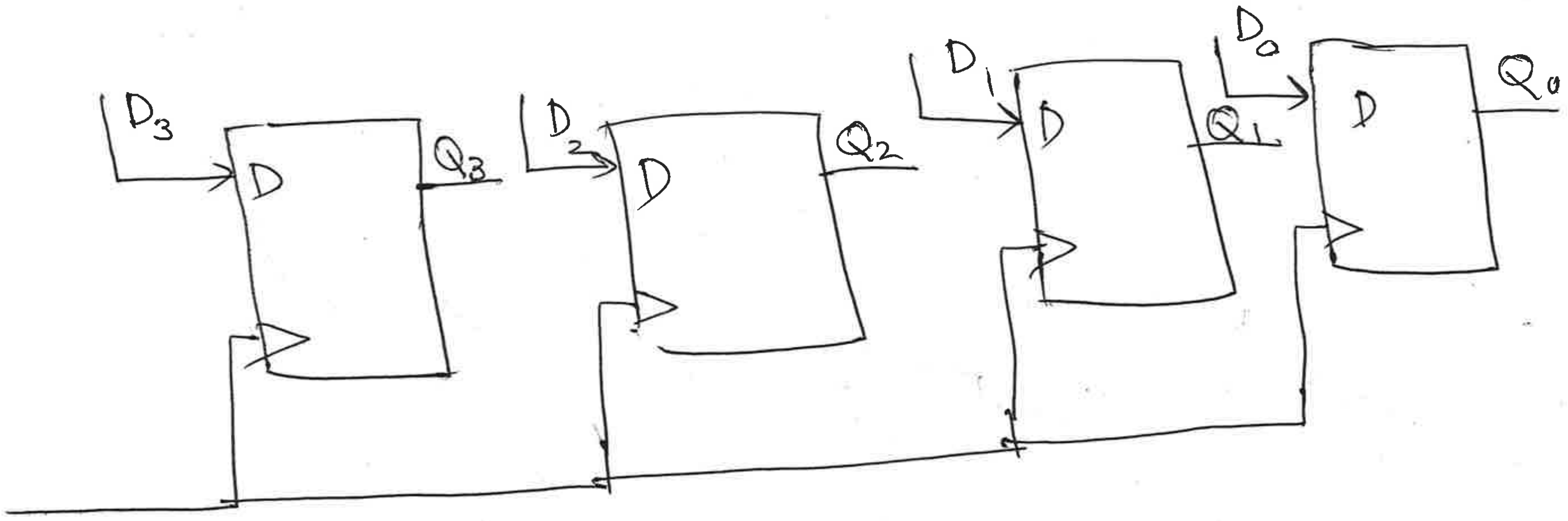


Negative edge

1 → 0

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

Example : (parallel in - parcallel out)



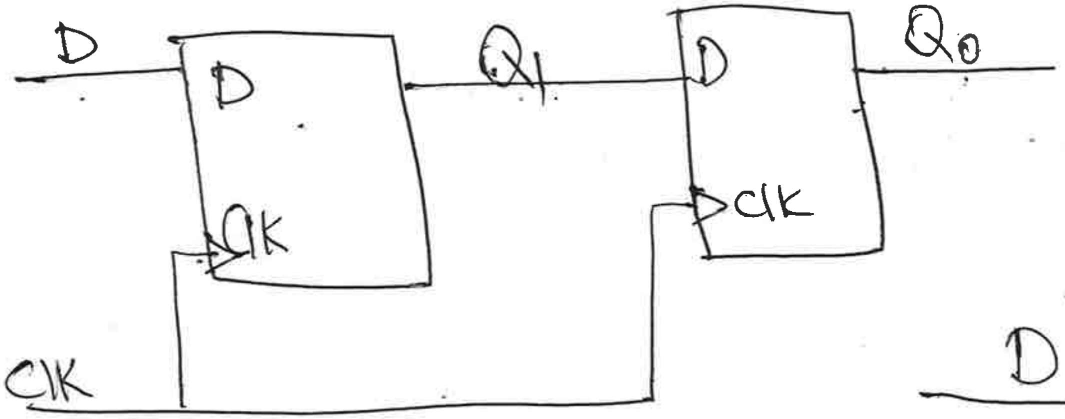
Same clock ⟹ Synchrous.

$D_3D_2D_1D_0$ (4 bit)

$Q_3Q_2Q_1Q_0$ (output)

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

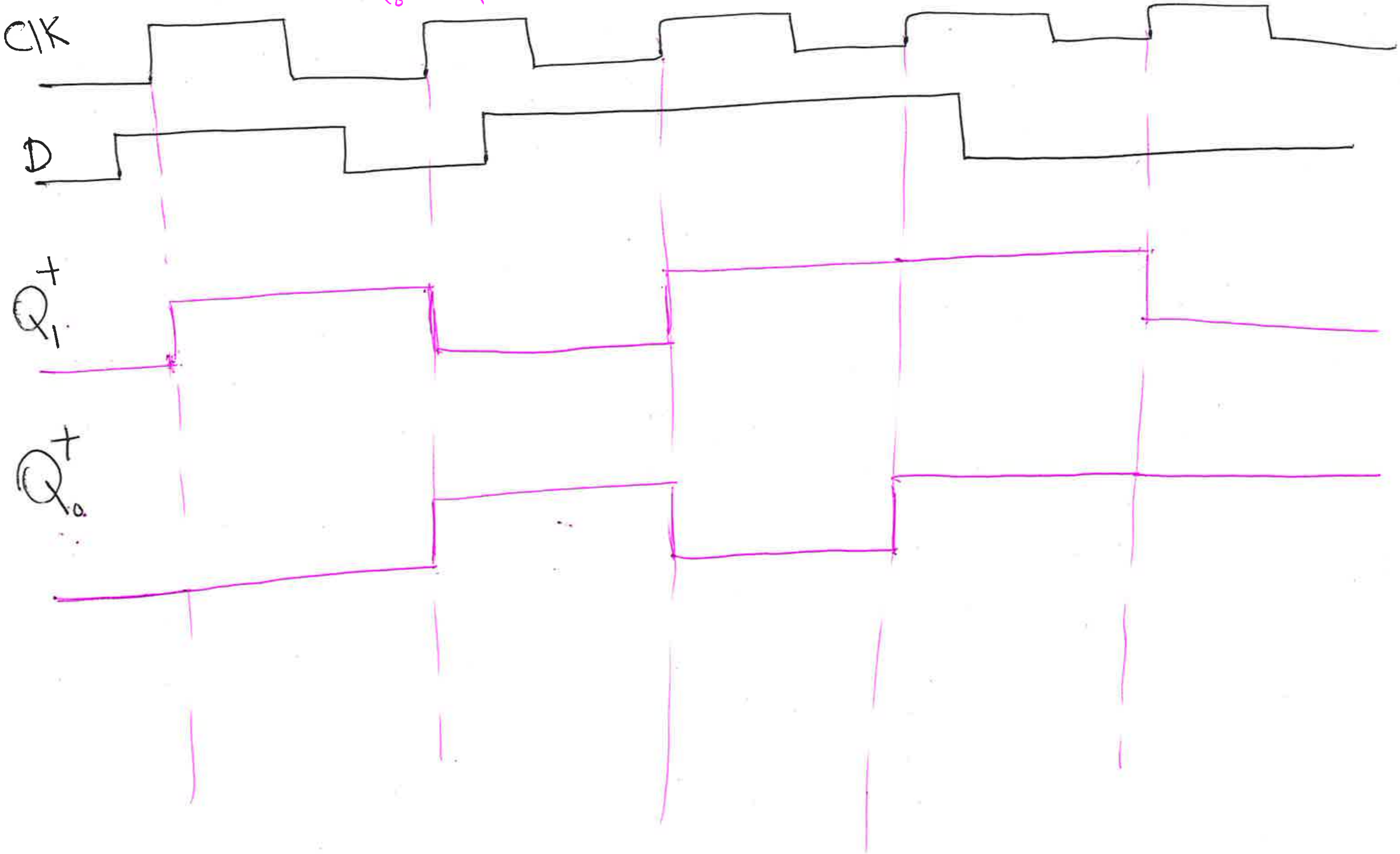
Example : (Shift register)



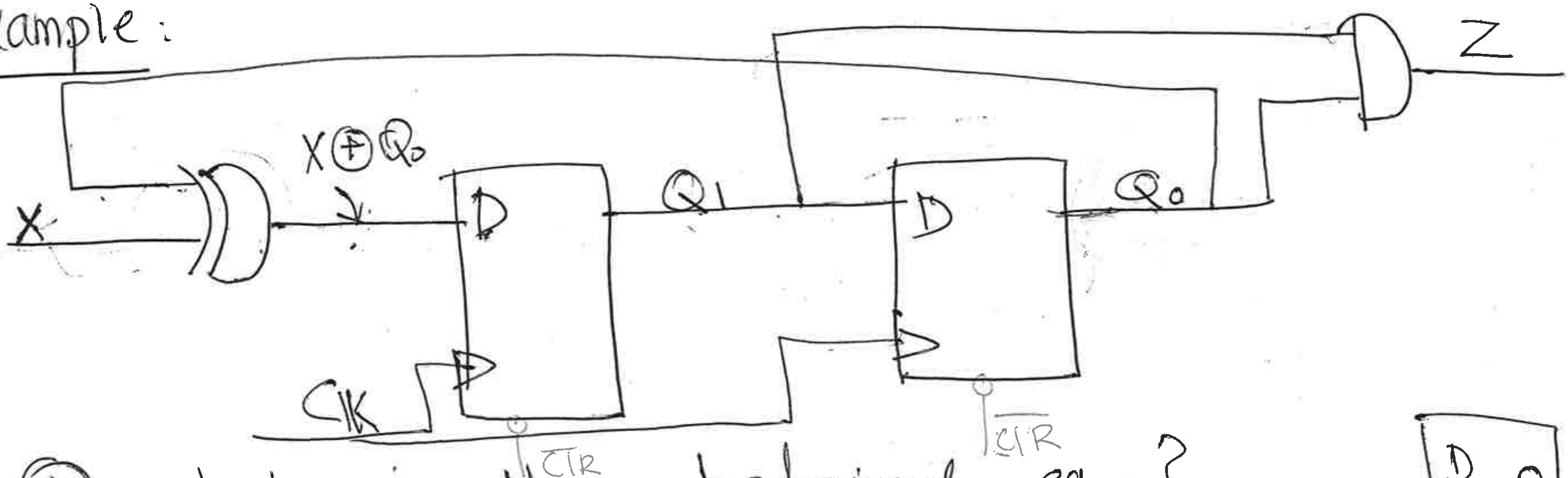
$$\begin{cases} Q_1^+ = D \\ Q_0^+ = Q_1 \end{cases}$$

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

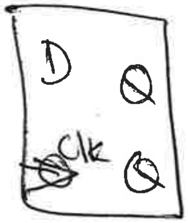
$Q_1^+ = D$, $Q_0^+ = Q_1$



Example:



① what is the behaivival eq · ?

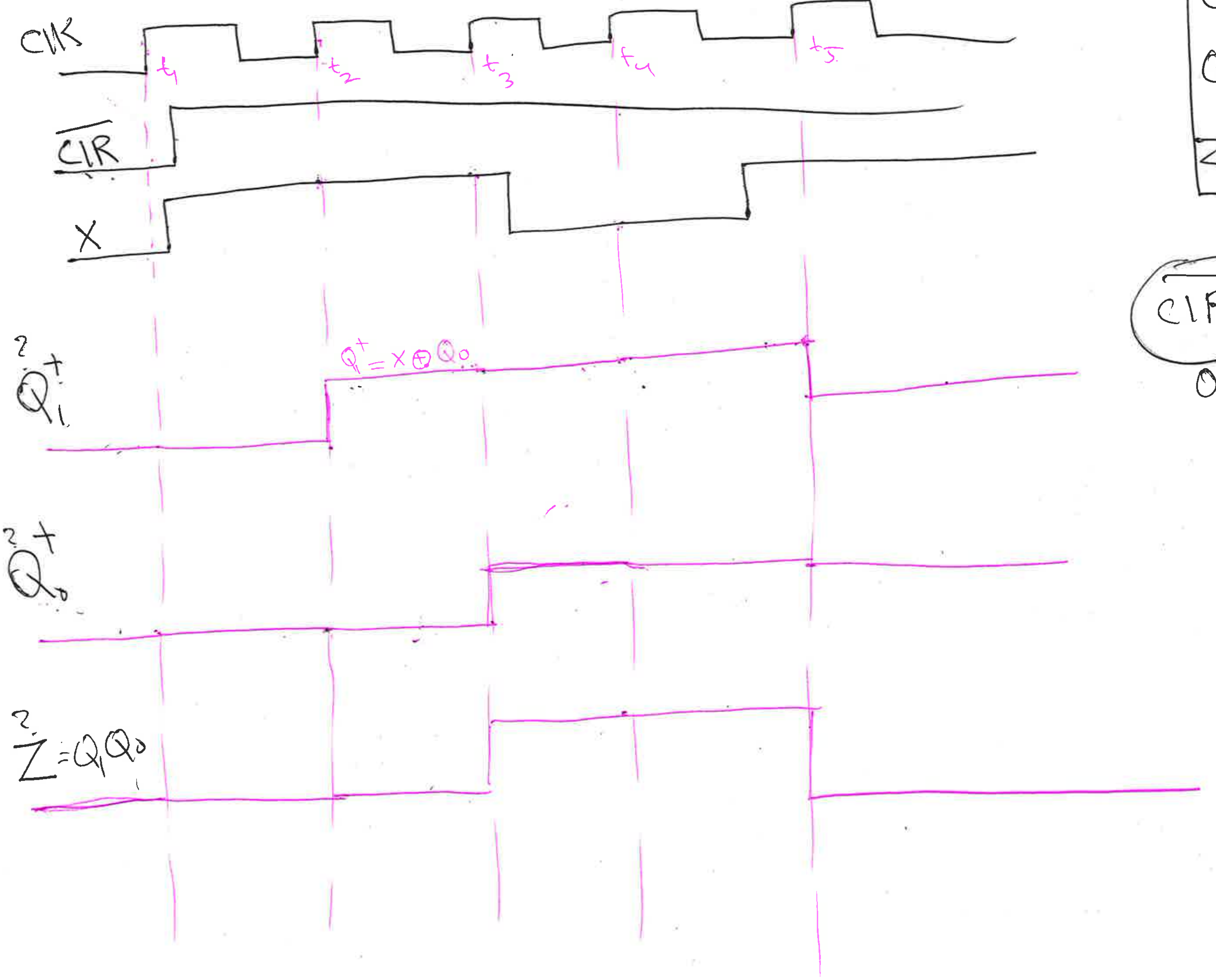


$Q^+ = D$

next states

$$\begin{cases} Q_1^{\overset{\text{next}}{+}} = X \oplus Q_0 \\ Q_0^+ = Q_1 \end{cases}$$

output ← $Z = Q_1 Q_0$



$Q_1^+ = X \oplus Q_0$

$Q_0^+ = Q_1$

$Z = Q_1 Q_0$

CIR=1

$\overline{CIR}=0$ ⟹

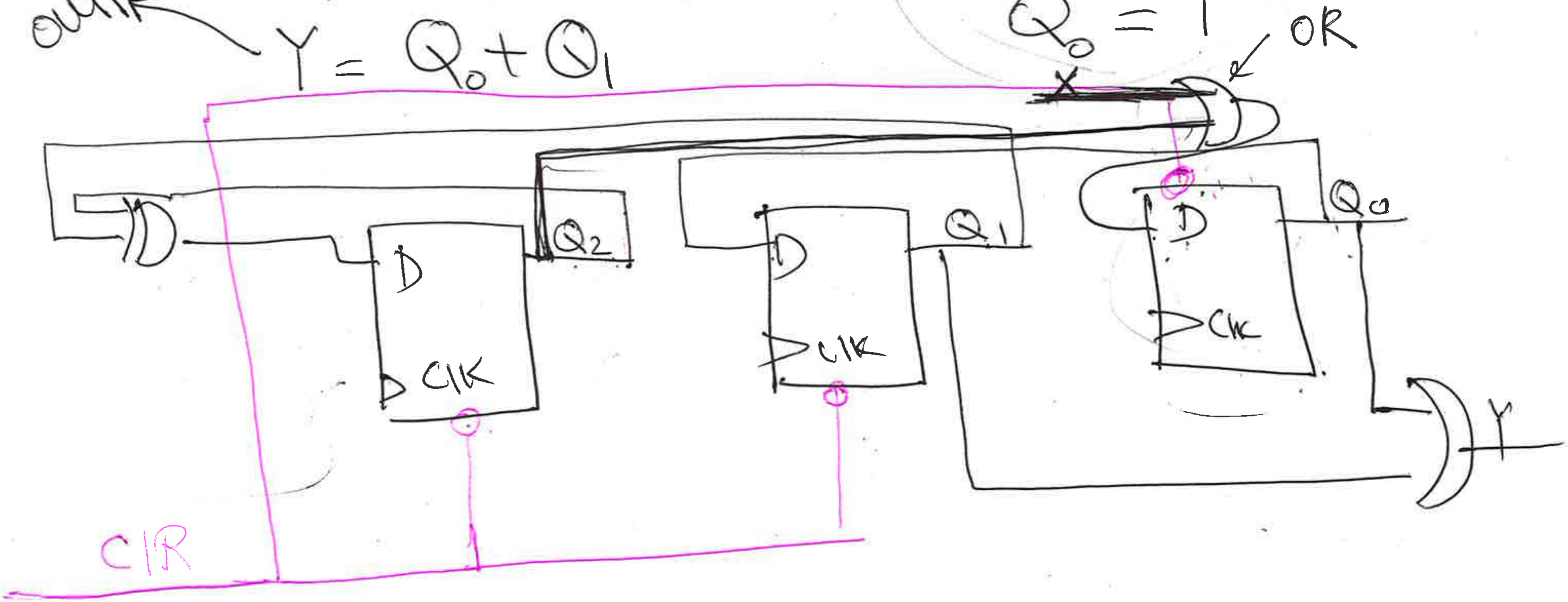
output = 0

Example: Register that follows these equations:

Next State
$$Q_2^+ = Q_1 \oplus Q_2$$
$$Q_1^+ = Q_0$$
$$Q_0^+ = X + Q_2$$

output
$$Y = Q_0 + Q_1$$

Assuming
$$Q_2 = 0$$
$$Q_1 = 0$$
$$Q_0 = 1$$



ClR = 1

(Synchrous, ClKs are connected to the same clk.

EEE/CSE 120 : Review for Midterm

office hours :

Monday 3:00 - 4:00 pm

Example1: Add # (Signed)

Cin

$\boxed{1}$ (1) (1) (0)

$$\begin{array}{rccccccl} & 1 & 0 & 0 & 1 & 1 & \longrightarrow & (-13)_{10} \\ + & 0 & 1 & 1 & 1 & 0 & & (14)_{10} \\ \hline \boxed{1} & (0 & 0 & 0 & 0 & 1) & & +1 \end{array}$$

Cout

$(01101) \to +13$

$Cin = Cout \Rightarrow$ NO OF.

cin

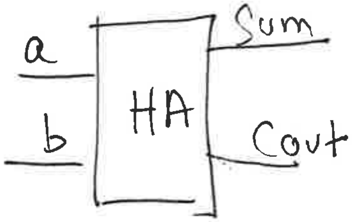
$\boxed{0}$ (0) (0)

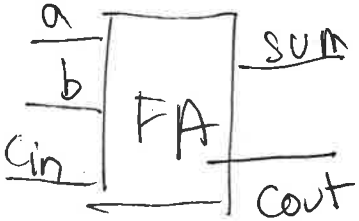
$$\begin{array}{rccccccl} & 1 & 0 & 0 & 1 & 0 & \longrightarrow & \overbrace{(01110)}^{14} \to -14 \\ + & 1 & 1 & 0 & 0 & 1 & \longrightarrow & \underbrace{(00111)}_{7} \to -7 \\ \hline \boxed{1} & 0 & 1 & 0 & 1 & 1 & & \end{array}$$

cout

2^4 2^3 2^2 2^1 2^0

$Cin \neq Cout \Rightarrow$ OF.





Example 2 :

$(213)_{16} \longrightarrow (\qquad\qquad)_{10}$ ← Decimal

$$\begin{tikzcd} 2 \arrow[d] & 1 \arrow[d] & 3 \arrow[d] \\ 16^2 & 16^1 & 16^0 \end{tikzcd}$$

$3\times16^0 + 1\times16^1 + 2\times16^2$

$(327)_{10} \longrightarrow$ HEX

$$\begin{array}{r|l} 327 & 16 \\ \hline 7 & 20 \end{array} \qquad \begin{array}{r|l} 20 & 16 \\ \hline 4 & 1 \end{array}$$

R ← 7

$(147)_{16}$

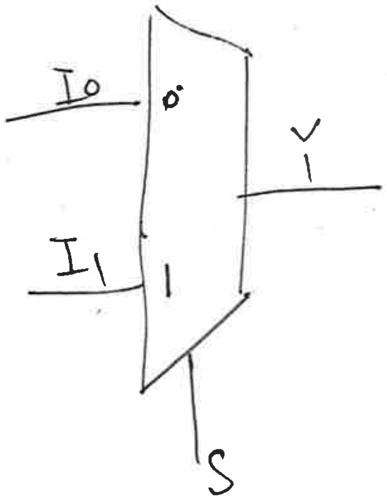
$(A)_{10} = \sum_{i=1} c_i a^i = (\ ,\ ,\ ,\ )_a$

$\underbrace{1\times16^2 + 4\times16^1 + 7\times16^0}_{327}$

$(F729)_{16} \longleftrightarrow$ binary

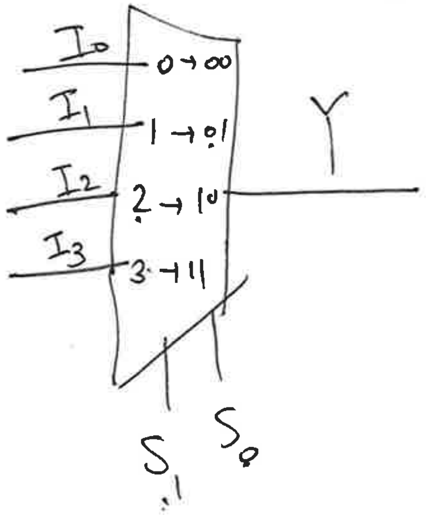
$16 \to 2^4$

$(1111\ 0111\ 0010\ 1001)_2$



$$\underbrace{\begin{aligned} S=0 &\Rightarrow Y=I_0 \\ S=1 &\Rightarrow Y=I_1 \end{aligned}}$$

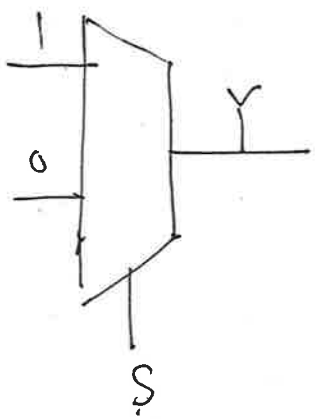
$$Y=\underline{\underline{\overline{S}}}\,I_0+\underline{\underline{S}}\,I_1$$



$$Y=\overline{S}_1\overline{S}_0 I_0+\overline{S}_1 S_0 I_1+S_1\overline{S}_0 I_2$$
$$+\underline{\underline{S_1 S_0}}\,I_3$$

Is a Mux functionally complete?

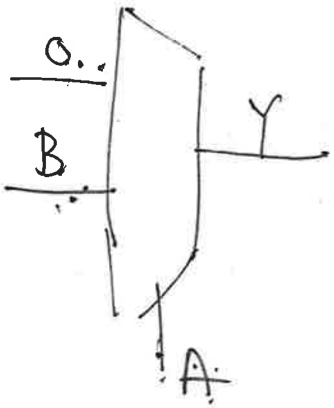
① Invertor



$S=0 \Rightarrow Y=1$

$S=1 \Rightarrow Y=0$ $\rightsquigarrow Y=\overline{S}$ ✓

② AND

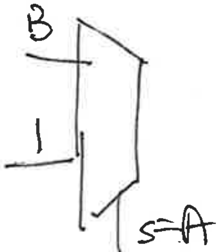


$A=0 \Rightarrow Y=0B=AB$

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

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

③ OR



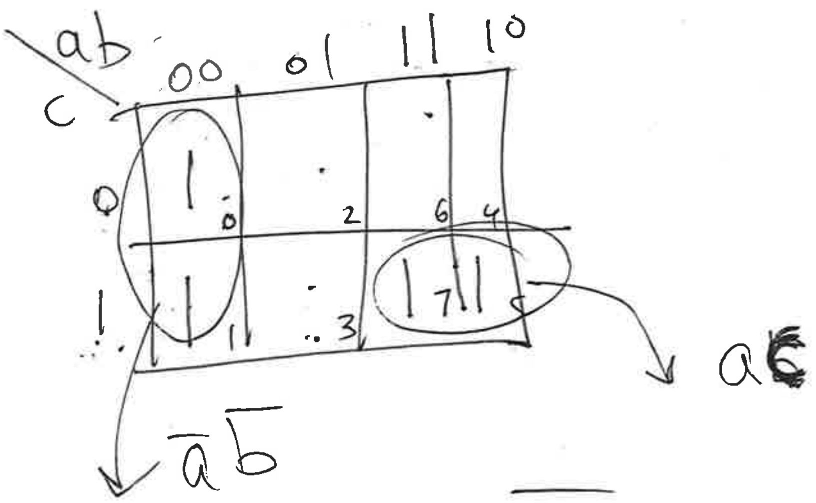
$A=0 \Rightarrow Y=B$

$A=1 \Rightarrow Y=1$ $\Big\}$ $A+B$

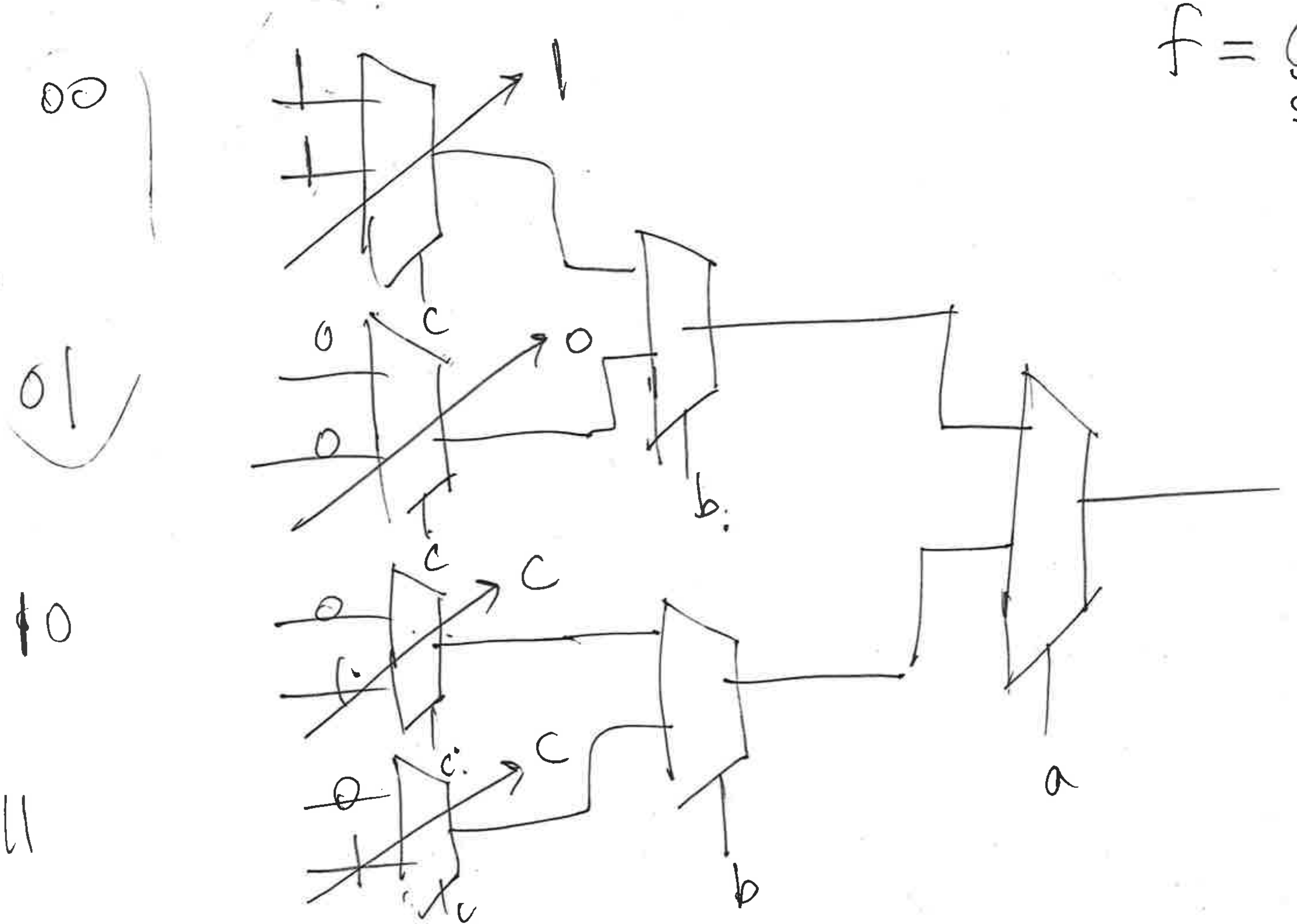
0	0	0
0	1	1
1	0	1
1	1	1

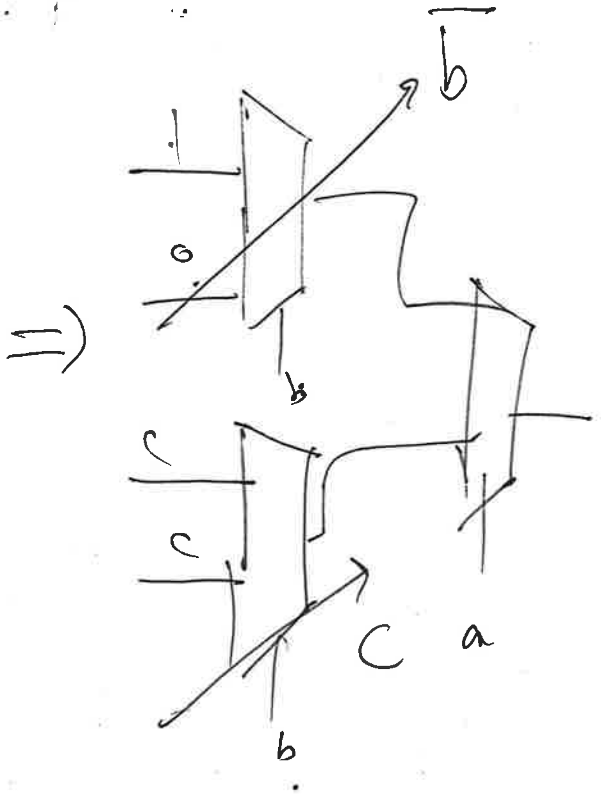
$f(a,b,c) = \Sigma m(0,1,5,7)$

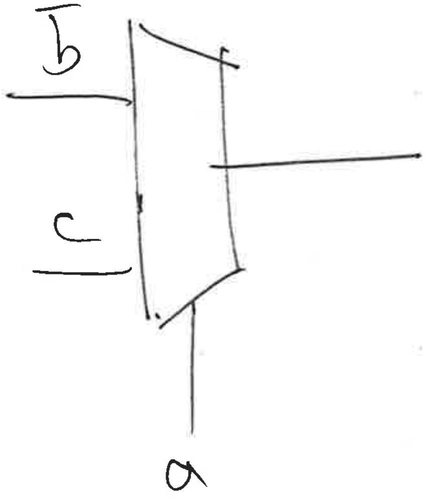
- use 7 multiplexers
- use 3 muxes
- use 1 mux.



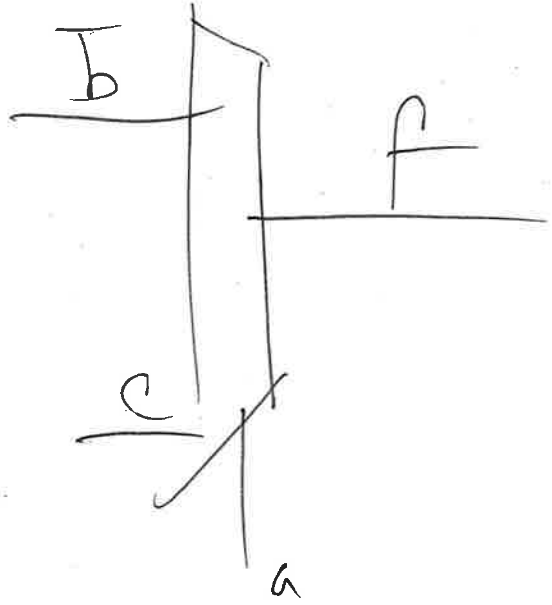
$$f = \underbrace{a}_{S_0}\underbrace{c}_{I_1} + \underbrace{\bar{a}}_{\bar{S}_0}\underbrace{\bar{b}}_{I_0}$$







$$f = \underbrace{a}_{s}\overset{I_1}{c} + \underbrace{\bar{a}}_{\bar{s}}\overset{I_0}{\bar{b}}.$$



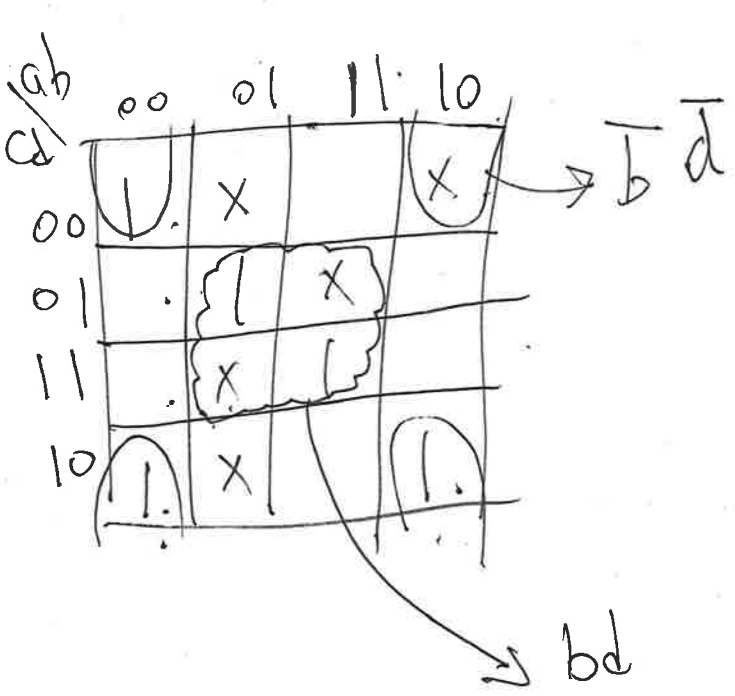
Example 4 :

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

① Find minimum SoP ?

② Find minimum PoS ?

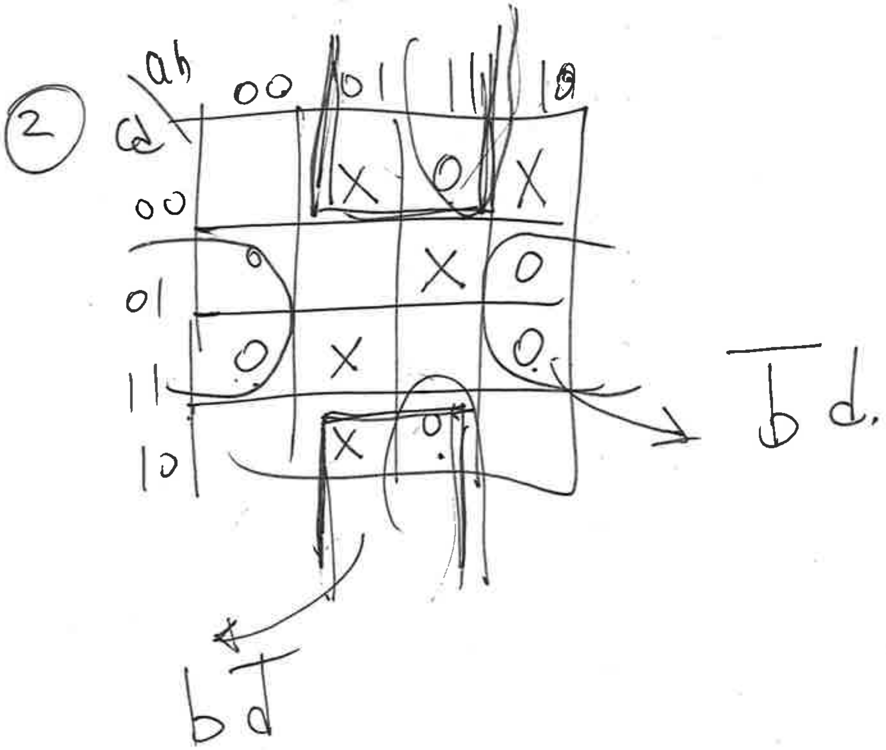
①



cd \ ab	00	01	11	10
00	1	X		X
01		1	X	
11		X	1	
10	1	X		1

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

②



cd \ ab	00	01	11	10
00		X	0	X
01	0		X	0
11	0	X		0
10		X	0	

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

$$f = \overline{\overline{f}} = \overline{\left(\overline{b}d + b\overline{d}\right)}$$

$$= \left(b + \overline{d}\right)\left(\overline{b} + d\right)$$

oct 22, 2020

EEE/CSE 120 : Design sequential Finite state Machines

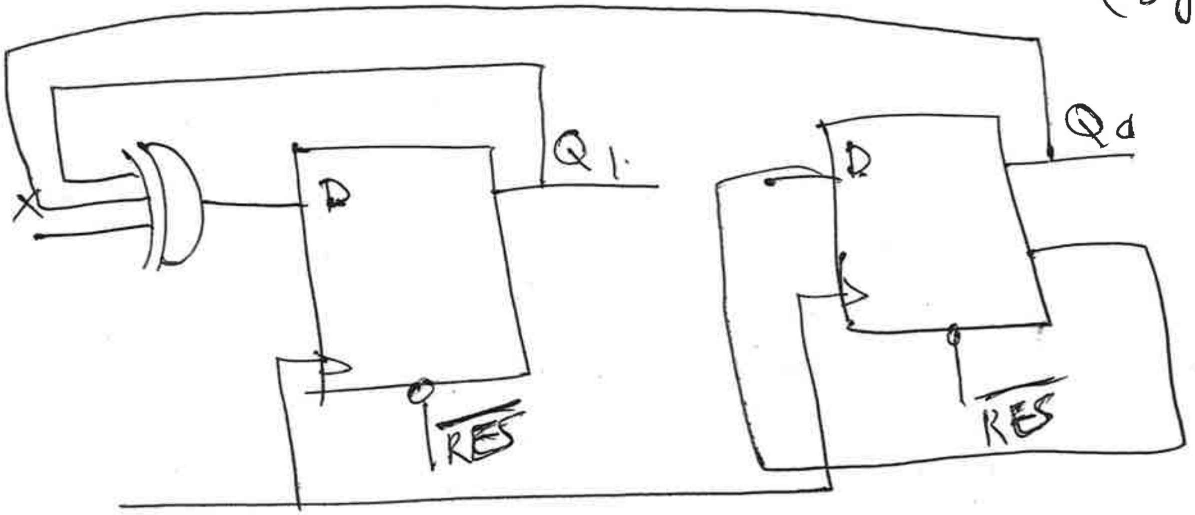
- office Hrs T/TH 9:30-10:15 AM
- Midterm redo for extra credit up to 10 points
  starting today at 6pm
  Till Friday (tomorrow) at 5:59 pm.

- 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.

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

3 FFs ⇝ $2^3 = 8$ 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 $\underbrace{3 \text{ to } 0}_{\text{Decimal \#}}$ stopping at 0

3, 2, 1, 0, 0, . . .

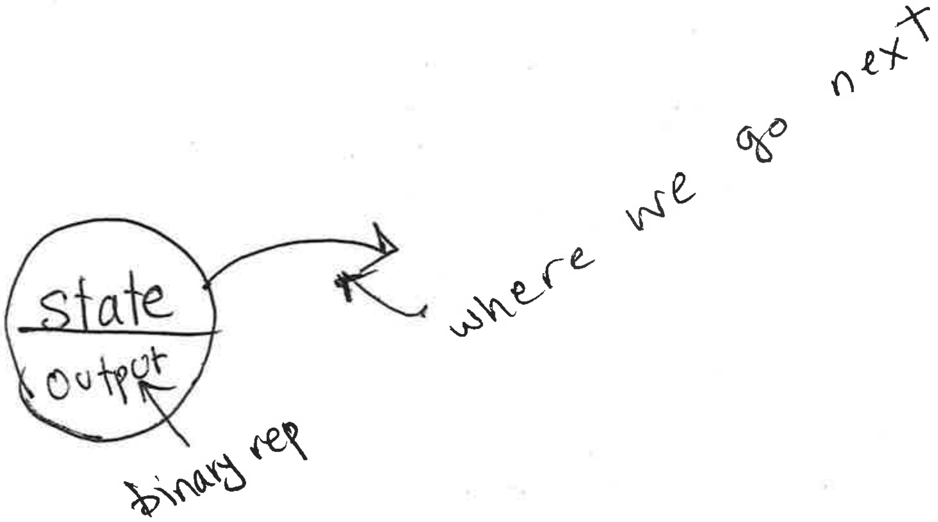
(1)

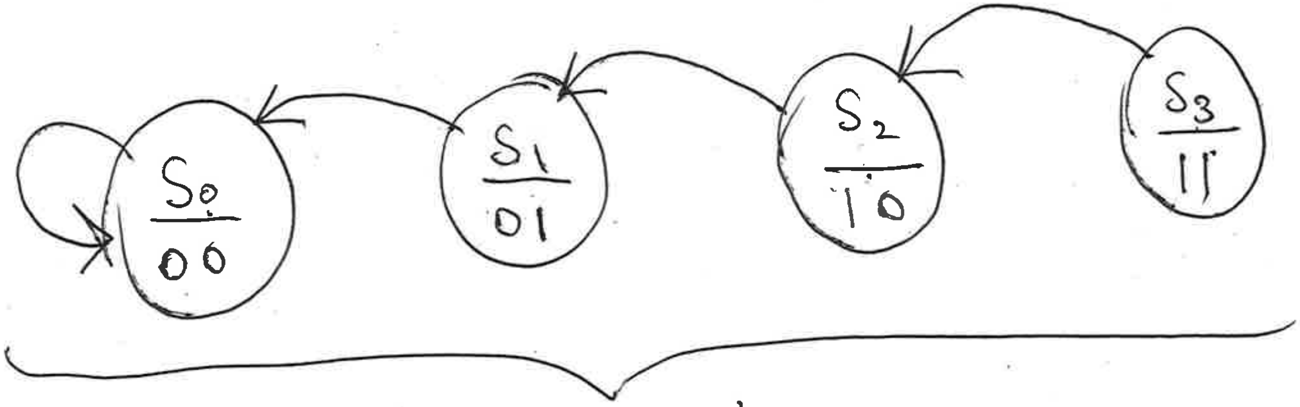
States	Definition	Binary rep. of states	
S_0	Count = 0.	0	0
S_1	Count = 1	0	1
S_2	Count = 2	1	0
S_3	Count = 3	1	1
		Q_1	Q_0

Two digits needed.

state def. table

② State transition diagram.





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

K-MAP for Q_0^+

$Q_0 \backslash Q_1$	0	1
0		1
1		

$Q_1\overline{Q_0}$

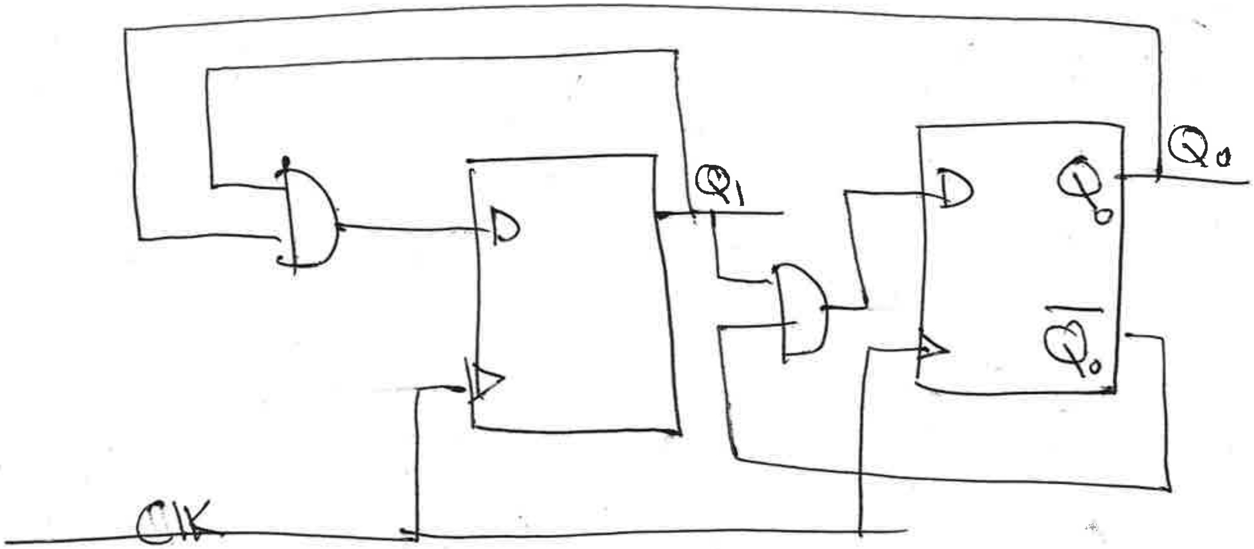
K-Map for Q_1^+

$Q_0 \backslash Q_1$	0	1
0		
1		1

$Q_1 Q_0$

④ Design the circuit

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



Example : Build a circuit that can count up / down

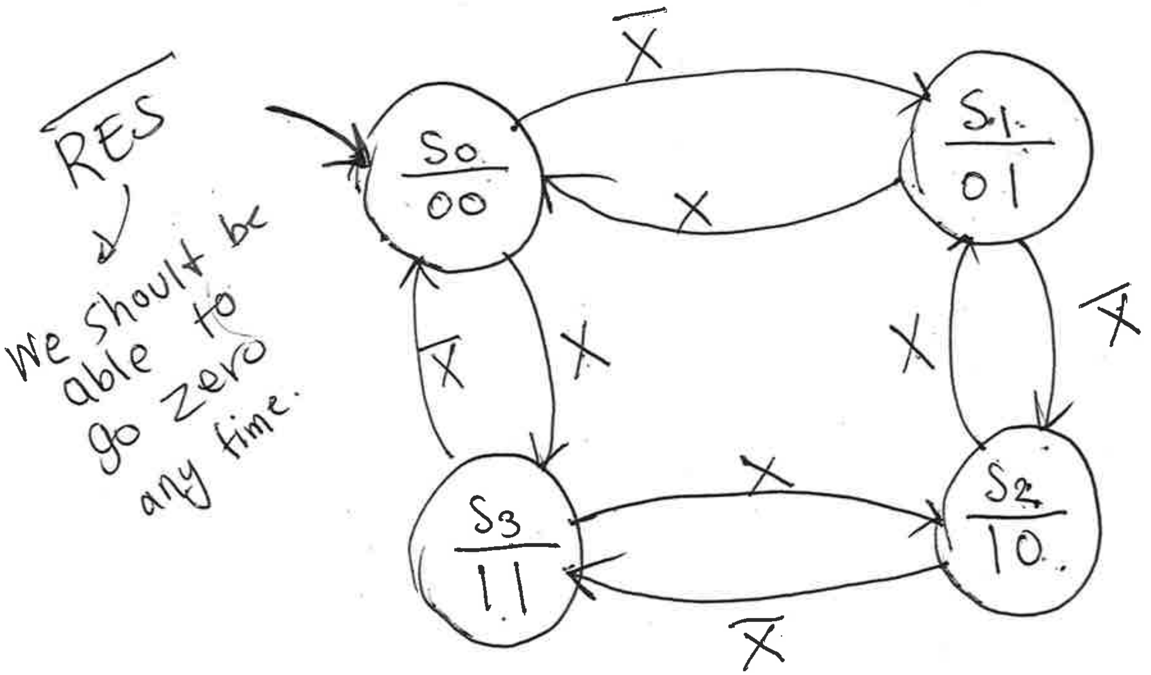
① State def. table

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 ⇝ $X = 0$: Counting up

$X = 1$: Counting down

② Draw transition diagram.



We should be able to go zero any time.

- $\underline{\text{prob. def}}$

  $3 \xrightarrow[\text{up}]{\text{counting}} 0$

- $\underline{\text{prob. def.}}$

  $0 \xrightarrow[\text{down}]{\text{Counting}} 3$

③ state transition table.

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

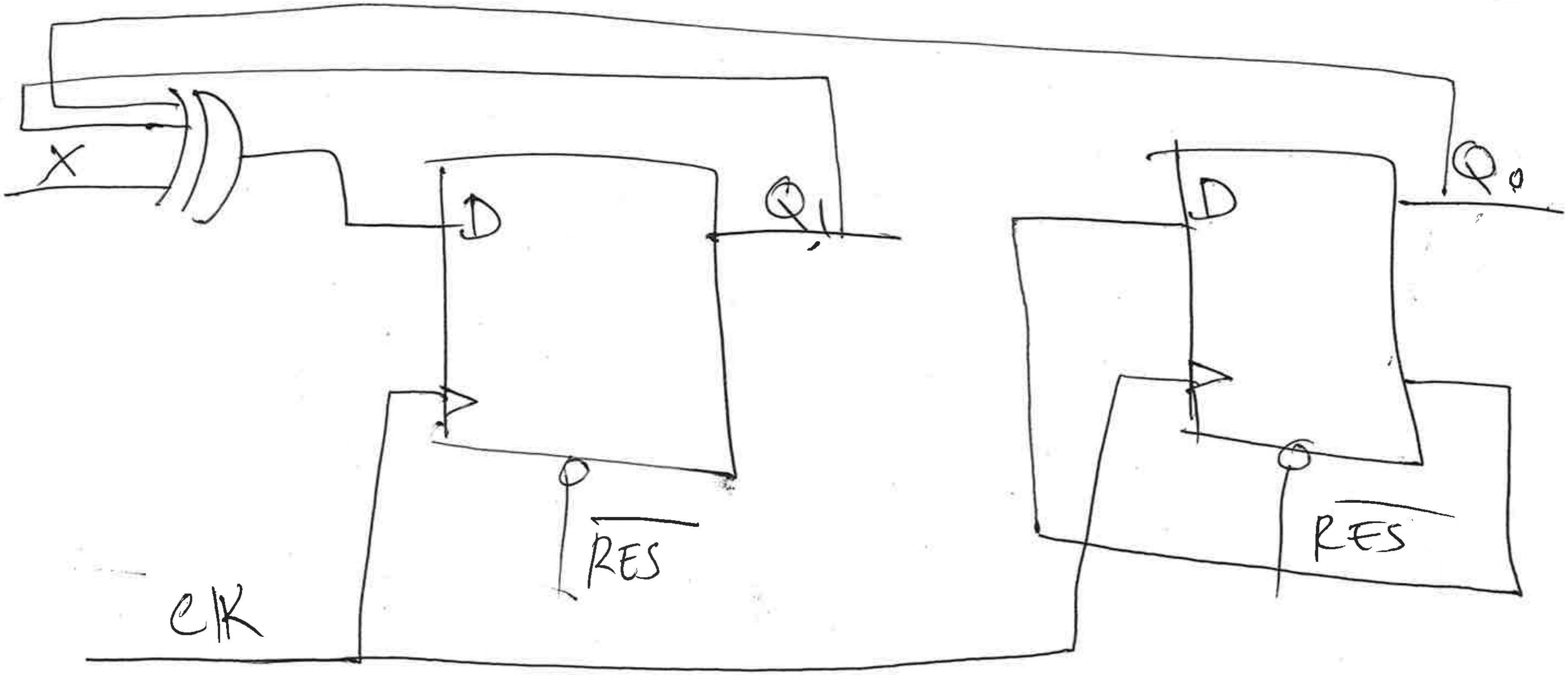
Design :

Q_0 \ XQ_1	00	01	11	10
0		1		1
1	1		1	

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

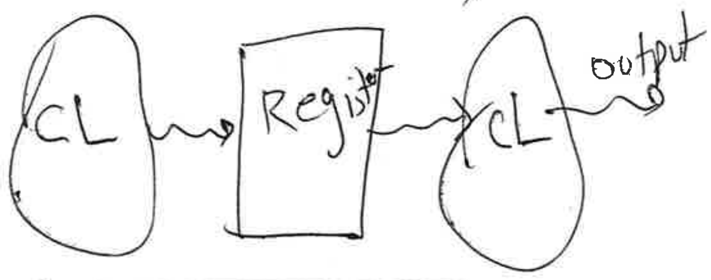
Q_0 \ XQ_1	00	01	11	10
0	1	1	1	1
1				

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



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

$$\begin{tikzcd}
 & \text{Moore Machine} \\
\text{FSM} \arrow[ur] \arrow[dr] & \\
 & \text{Mealy Machine}
\end{tikzcd}$$

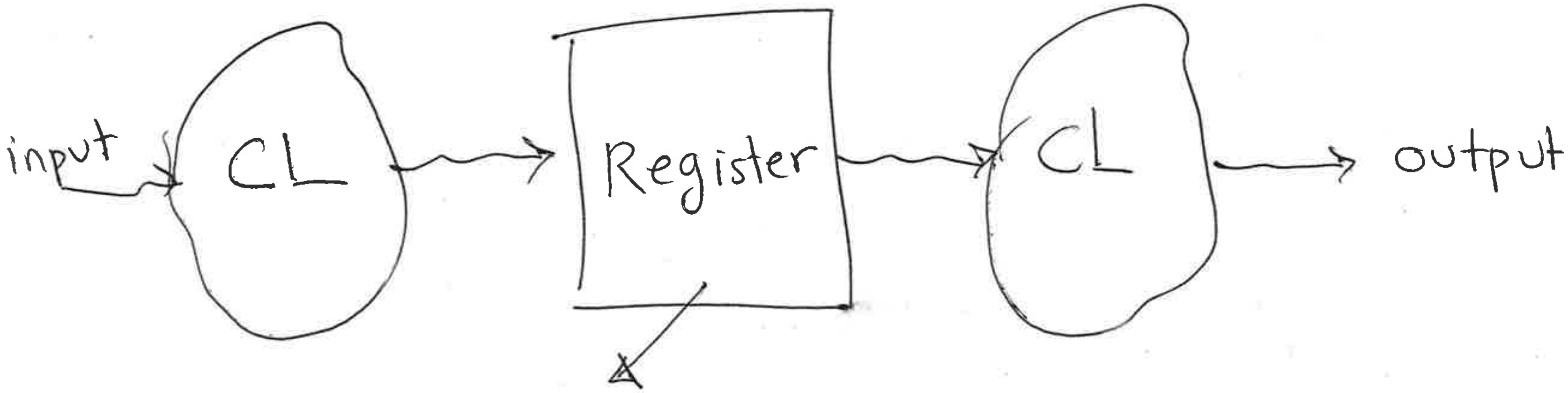


EEE/CSE 120 : Finite State Machines

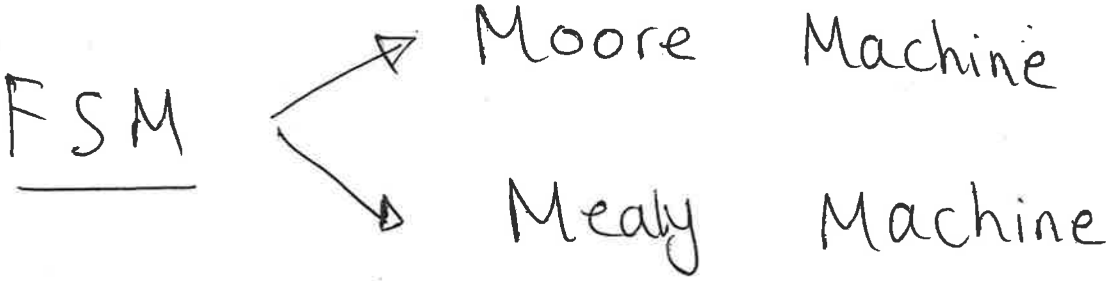
* HW 5 is due oct 29

* poll → due oct 31

↳ Canvas → Quizzes → poll.



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



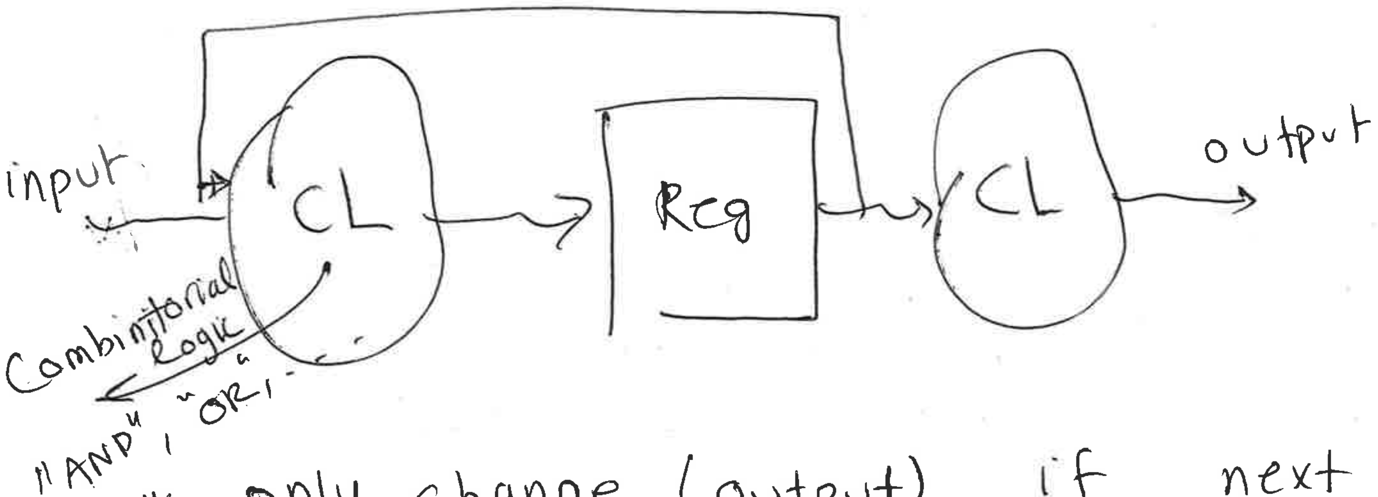
FSM

Moore Machine

Mealy Machine

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/ $\left\{\begin{array}{l} X=0 \\ X=1 \end{array}\right.$

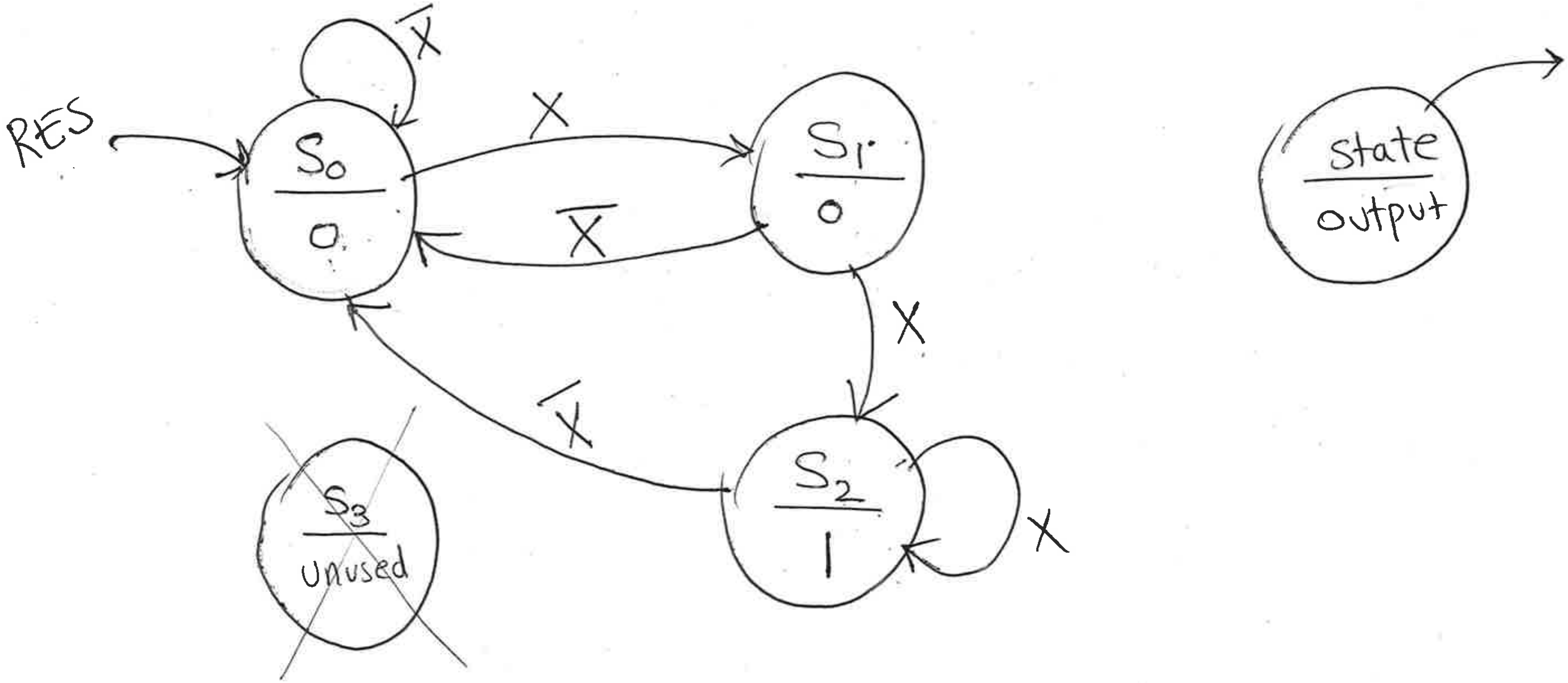
press "1" Twice in a row then the Safe opens up.

① State def. table:

states	name (def)	binary rep.		output
S_0	IdLE	0	0	0
S_1	One - one	0	1	0
S_2	two - ones	1	0	1
S_3	unused	1	1	X
		Q_1	Q_0	

input: $\left\{\begin{array}{l} X=0 \\ X=1 \end{array}\right.$

② Draw the state diagram



③ State transition table

states	input=X	Q_1	Q_0	Q_1^+	Q_0^+	Output
S_0	0	0	0	0	0	0
S_0	1	0	0	0	1	0
S_1	0	0	1	0	0	0
S_1	1	0	1	1	0	0
S_2	0	1	0	0	0	1
S_2	1	1	0	1	0	1
S_3	0	1	1	X	X	X
S_3	1	1	1	X	X	X

→ depends on the current states and not the future states!

④ Design the circuit

For Q_1^+

$Q_0 \backslash XQ_1$	00	01	11	10
0			1	
1		X	X	1

XQ_1

XQ_0

$$Q_1^+ = XQ_0 + XQ_1$$

For Q_0^+

$Q_0 \backslash XQ_1$	00	01	11	10
0				1
1		X	X	

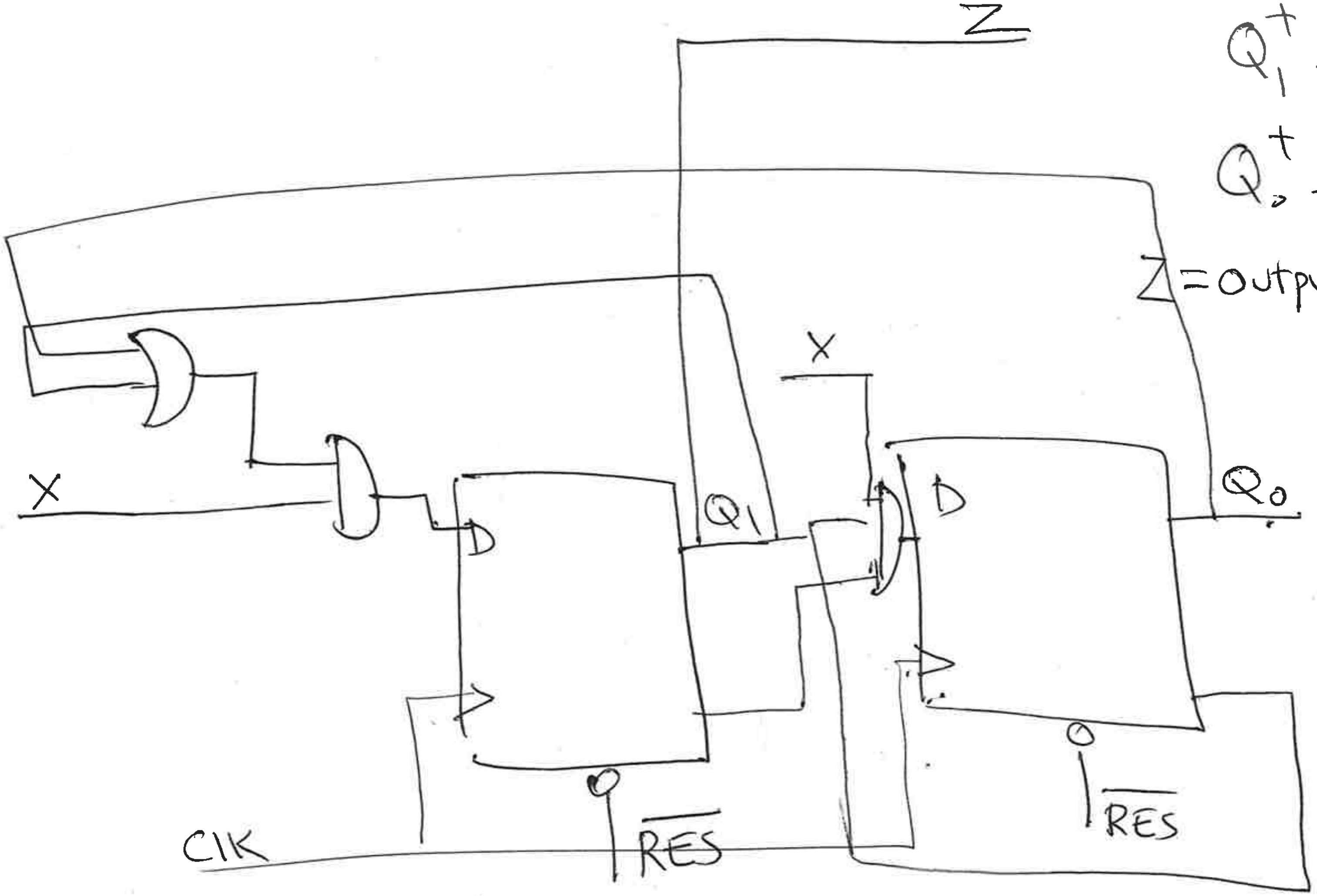
$$Q_0^+ = X\overline{Q}_1\overline{Q}_0$$

For output

$Q_0 \backslash XQ_1$	00	01	11	10
0		1	1	
1		X	X	

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

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



$$Q_1^{+} = XQ_0 + XQ_1$$

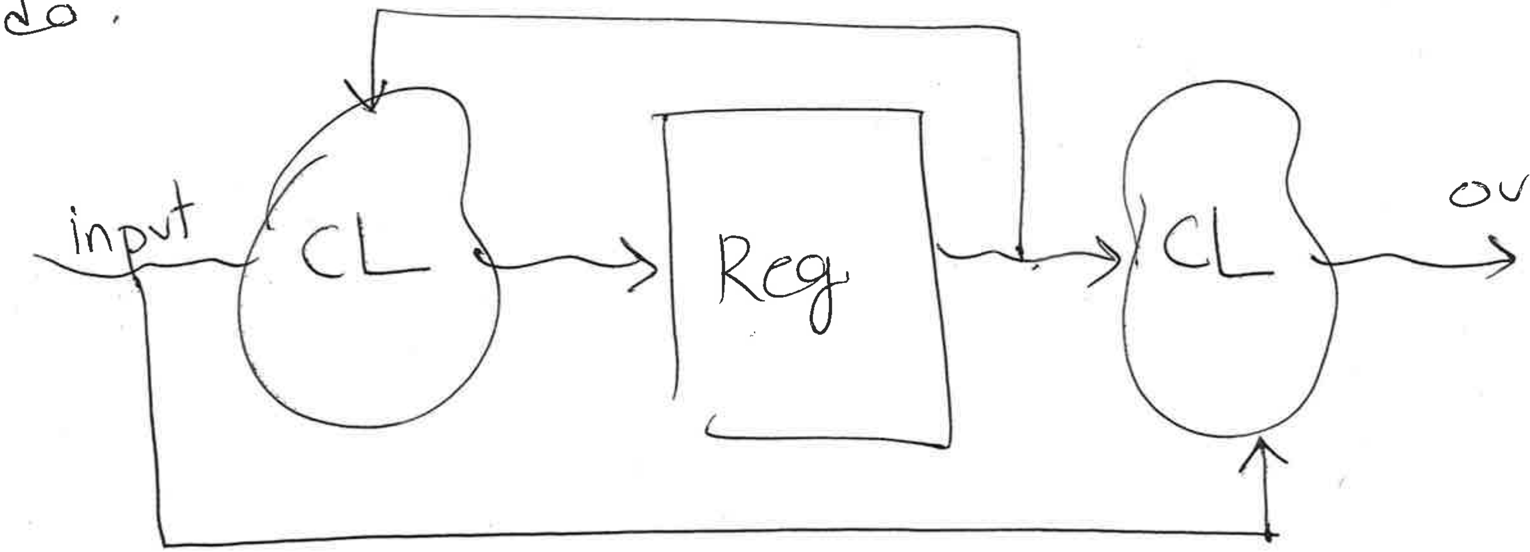
$$Q_0^{+} = X\overline{Q_1}\,\overline{Q_0}$$

Z = output = 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 ⇒ for output to change we have to wait for the next clock

Mealy Machine : Input change ⇒ output change.

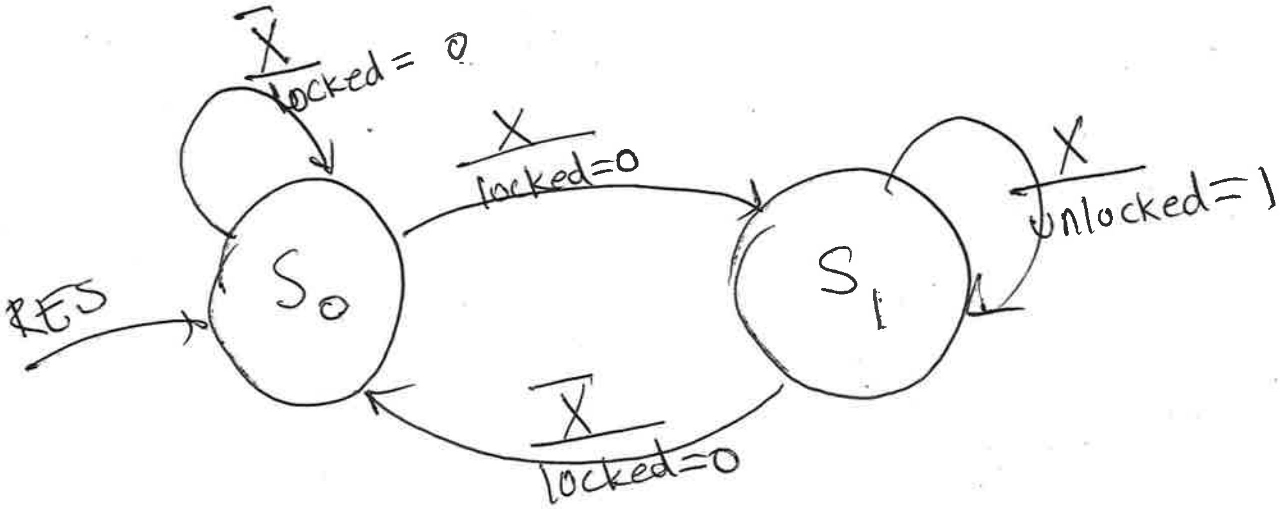
Example : Key pad $\begin{cases} X=0 \\ X=1 \end{cases}$, we press two ones in a row, then we open up the safe.

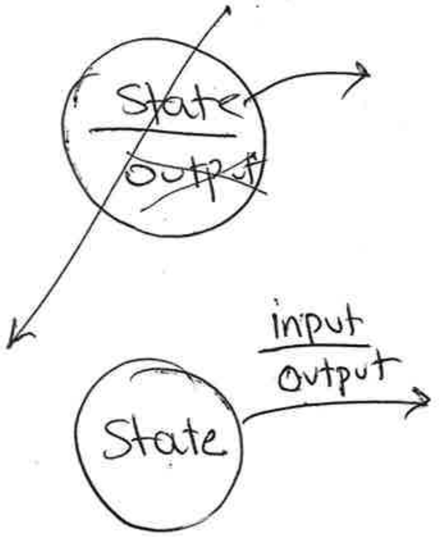
Design a Mealy Machine.

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

- Megly machine the output is a function of inputs.

③ State transition table

output → This depends on input

input X	Q	Q^+	output
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	1

④ Design the circuit

For Q^+

Q\X	0	1
0		1
1		1

→ $Q^+ = X$

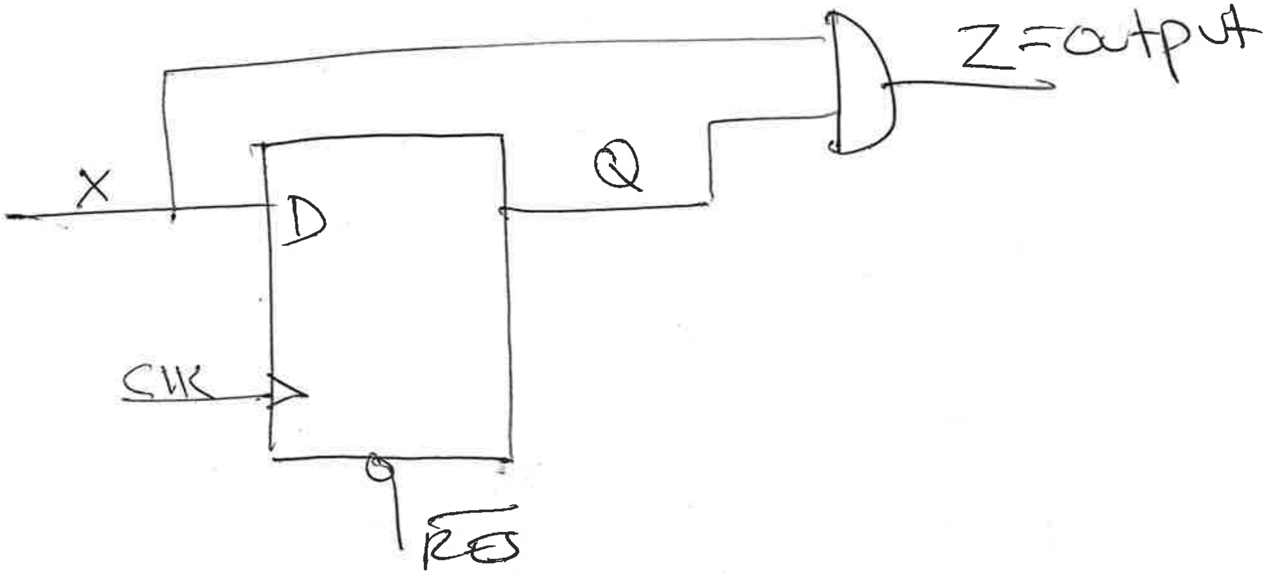
For output

Q\X	0	1
0		
1		1

output $= XQ$

$Q^+ = X$

$Z = \text{output} = XQ$



- There is timing issue

Synchrounization

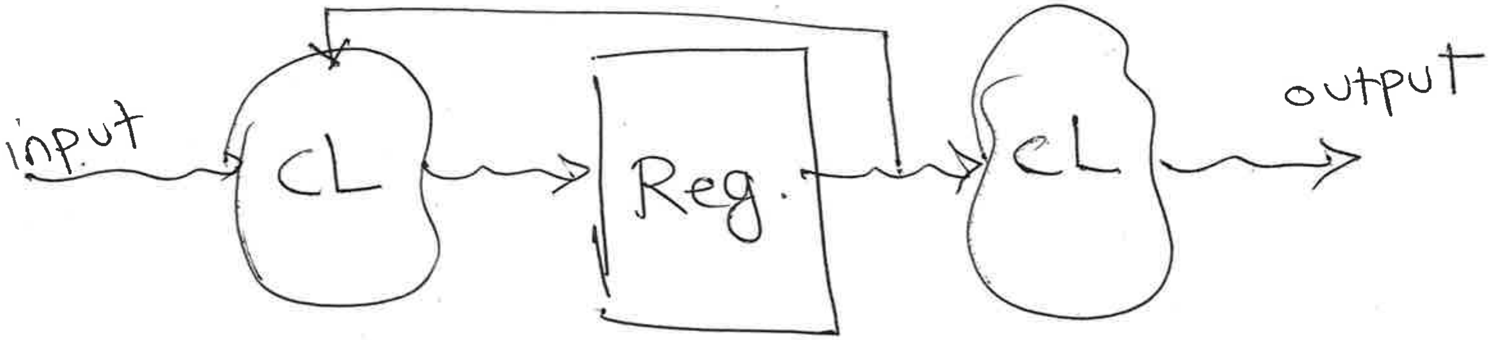
Adding one ff.

EEE/CSE 120: Moore and Mealy Machines

Oct 29, 2020

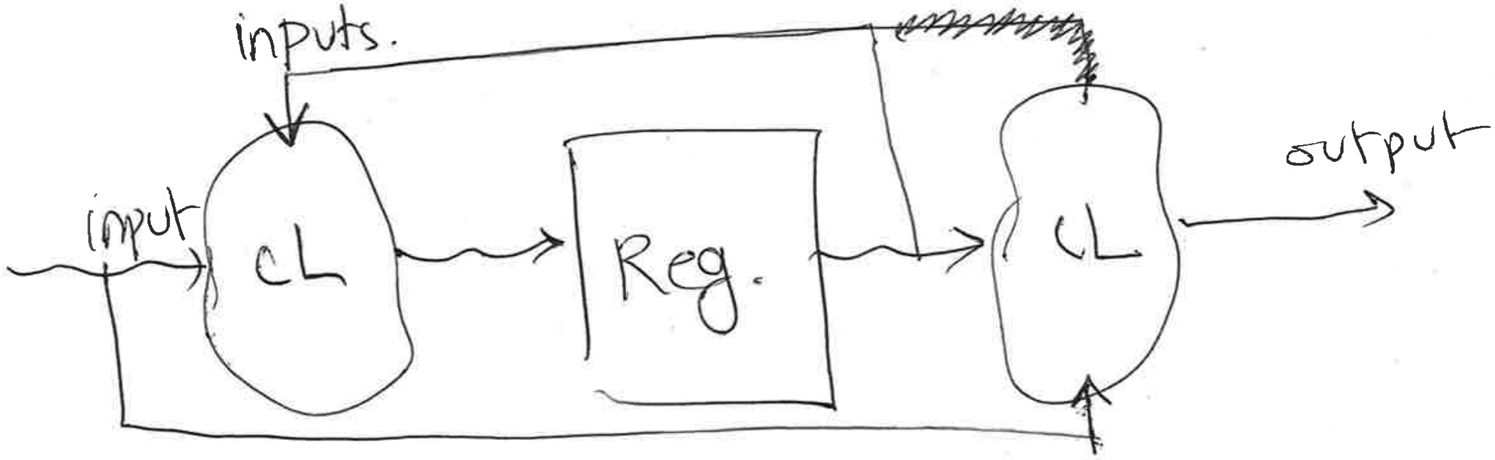
① Moore Machine:

FSM, there is no direct relationship between input & output



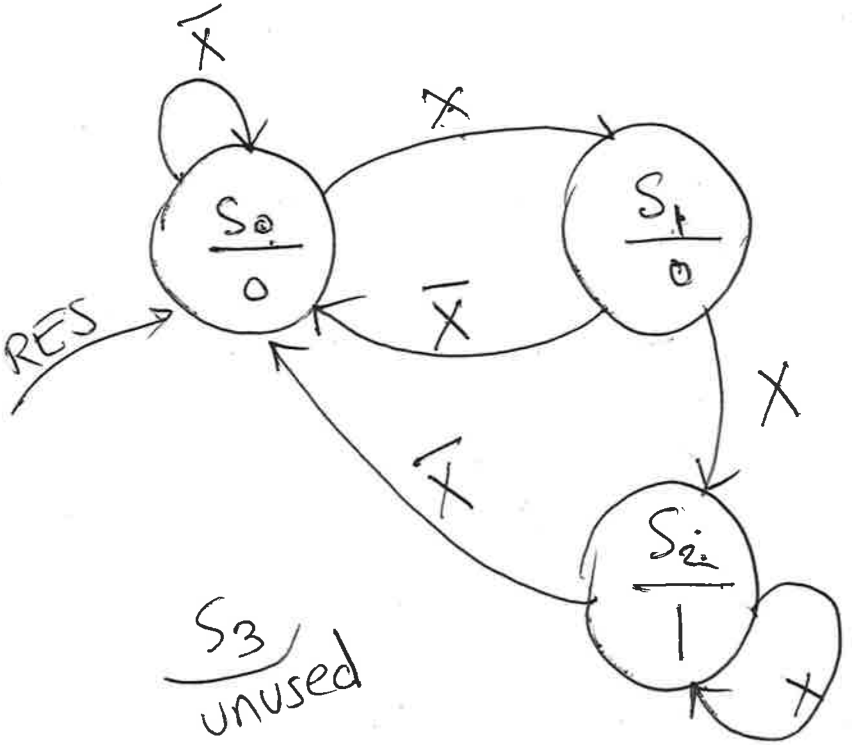
② Mealy Machine:

FSM: output change depends on both states & inputs.

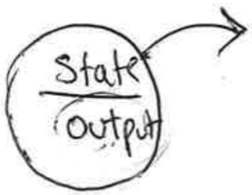


Review of the example: Keypad $\begin{cases} X=0 \\ X=1 \end{cases}$, safe opens up if we have two "1"s in a row.

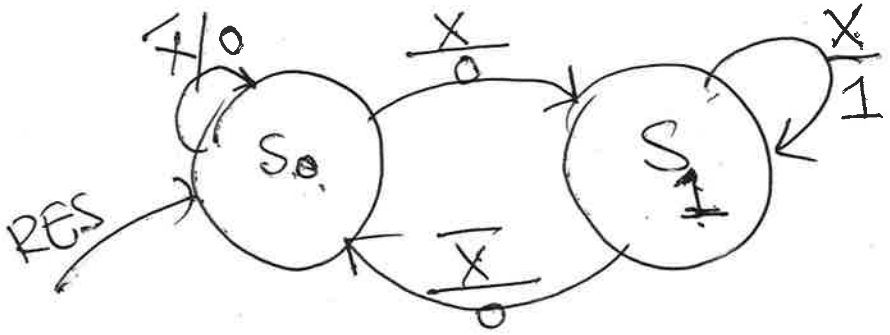
Moore Machine



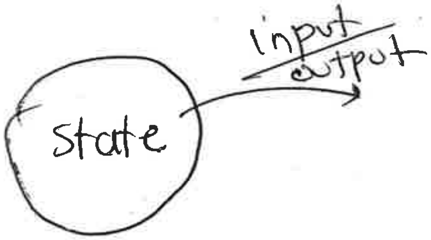
2 Flip flops

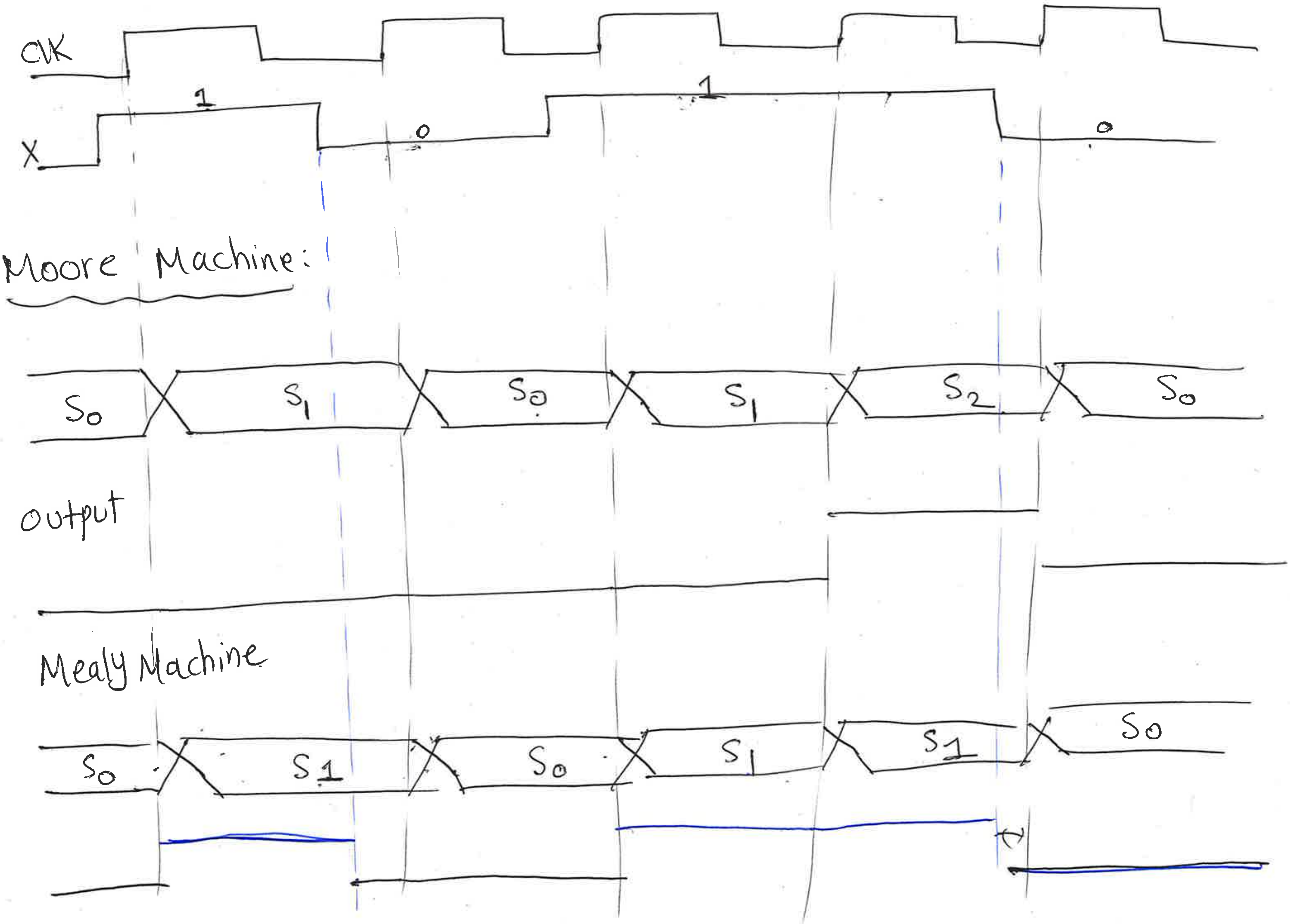


Mealy Machine



1 Flip flop.




CLK
X
1
0
1
0
Moore Machine:
S0
S1
S0
S1
S2
S0
output
Mealy Machine
S0
S1
S0
S1
S1
S0

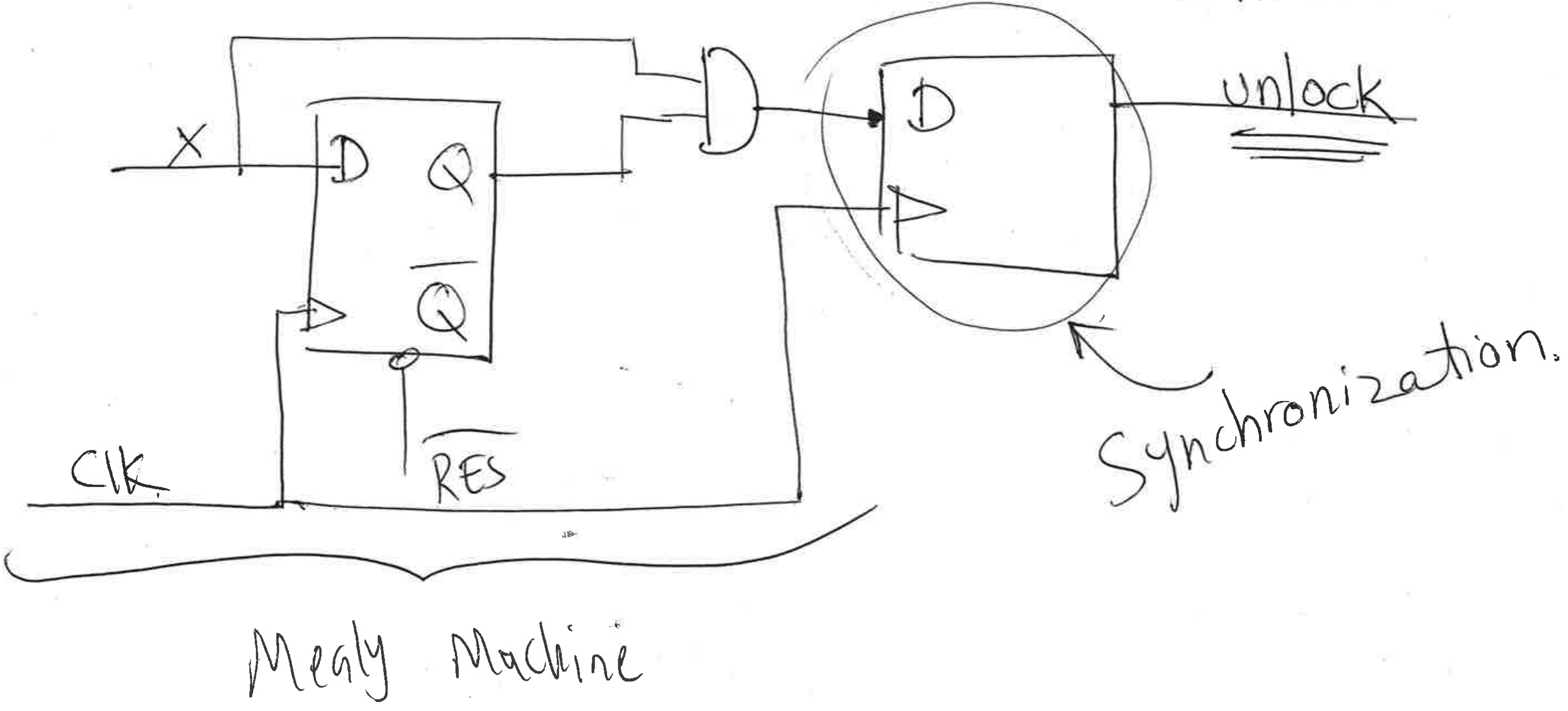
- Mealy Machines use less number of ffs.,

↳ 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

$$\text{inputs} = \begin{cases} e & \text{entering} \\ X & \text{exiting} \end{cases}$$

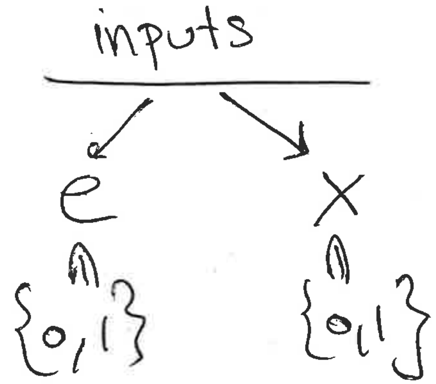
- Max # of people in the room = 3
- only one person at a time can leave /enter.
- If the room is full, red light on

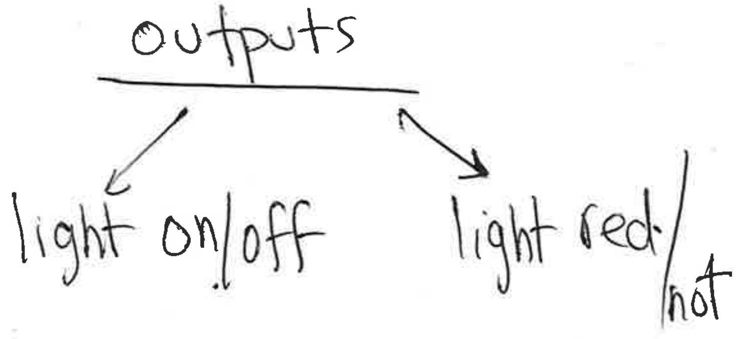
  If the room is empty, light is off

Q: Design a light Controller that Can do the above.

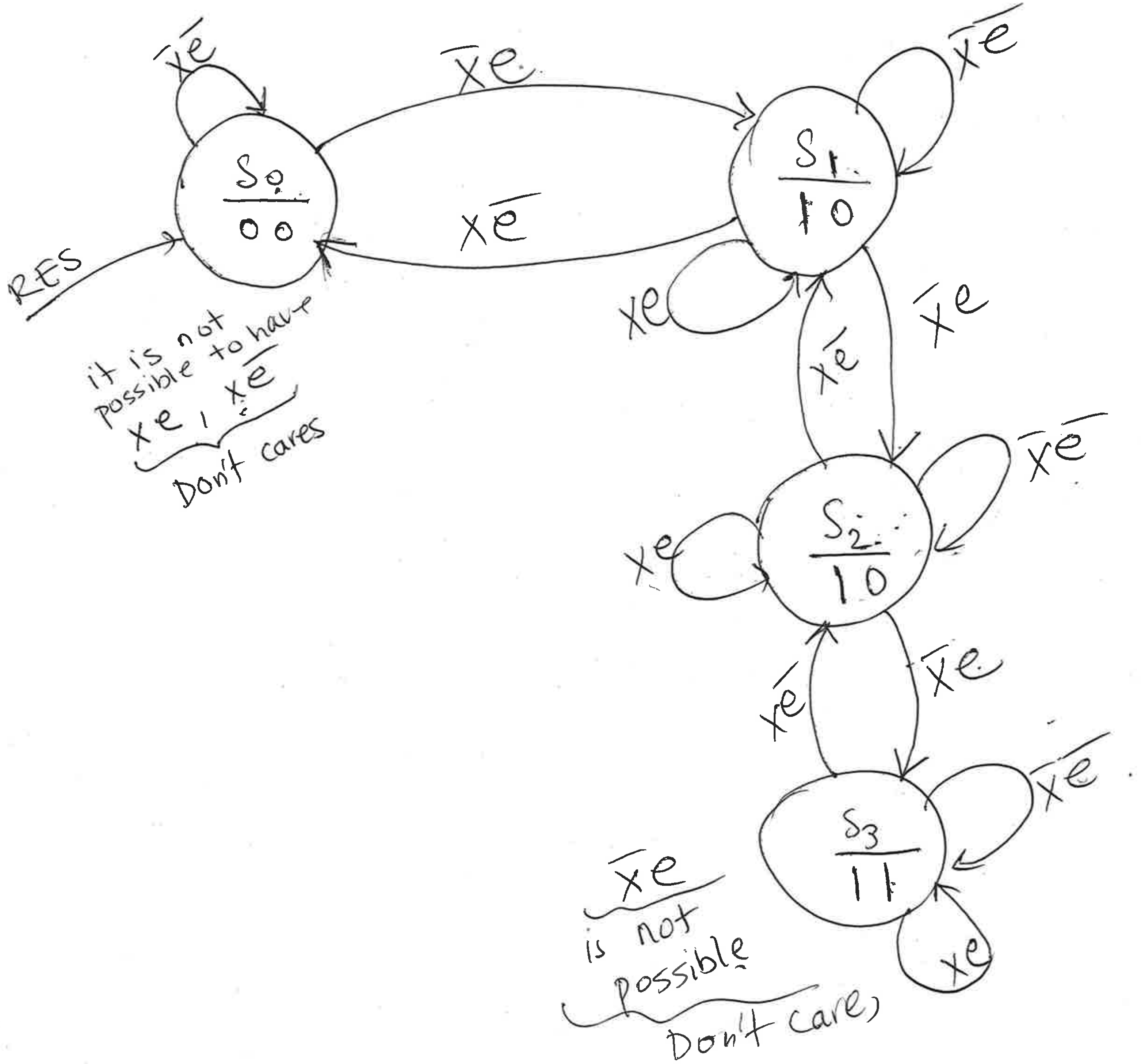
① state def. table:

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	1
		Q_1	Q_0





② state diagram



③ State transition table.

							light	red light
	Q_1	Q_0	X	E	Q_1^+	Q_0^+	L	R
State S_0	0	0	0	0	0	0	0	0
	0	0	0	1	0	1	0	0
	0	0	1	0	X	X	~~X~~ 0	~~X~~ 0
	0	0	1	1	X	X	~~X~~ 0	~~X~~ 0
S_1	0	1	0	0	0	1	1	0
	0	1	0	1	1	0	1	0
	0	1	1	0	0	0	1	0
	0	1	1	1	0	1	1	0
S_2	1	0	0	0	1	0	1	0
	1	0	0	1	1	1	1	0
	1	0	1	0	0	1	1	0
	1	0	1	1	1	0	1	0
S_3	1	1	0	0	1	1	1	1
	1	1	0	1	X	X	X	X
	1	1	1	0	1	0	1	1
	1	1	1	1	1	1	1	1

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

④ For Q_1^+

$XE \backslash Q_1Q_0$	00	01	11	10
00			1	1
01		1	X	1
11	X		1	1
10	X			

$Q_1\bar{X}$

Q_1E

$Q_0\bar{X}E$

$$\boxed{Q_1^+ = Q_1\bar{X} + Q_1E + Q_0\bar{X}E}$$

For Q_0^+

$XE \backslash Q_1Q_0$				
		1	1	
	1		X	1
	X	1	1	
	X		1	1

Q_1Q_0

$\bar{Q}_0\bar{X}\bar{E}$

$Q_1X\bar{E}$

$\bar{Q}_0XE$

$$\boxed{Q_0^+ = Q_1Q_0 + \bar{Q}_0XE + Q_1X\bar{E} + \bar{Q}_0\bar{X}E}$$

For L

$XE \backslash Q_1Q_0$	00	01	11	10
00		1	1	1
01		1	X	1
11	X	1	1	1
10	X	1	1	1

Q_0

Q_1

$$\boxed{L = Q_1 + Q_0}$$

For R

$XE \backslash Q_1Q_0$	00	01	11	10
00			1	
01			X	
11	X		1	
10	X		1	

$$\boxed{R = Q_1Q}$$

Lecture 21

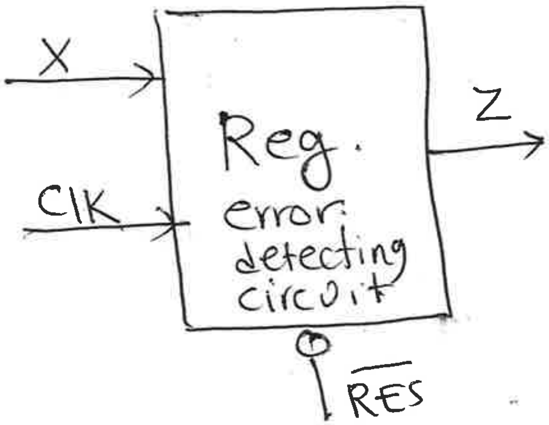
EEE | CSE 120 : Capstone project

(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 "0 0" or three ones in a row "1 1 1"

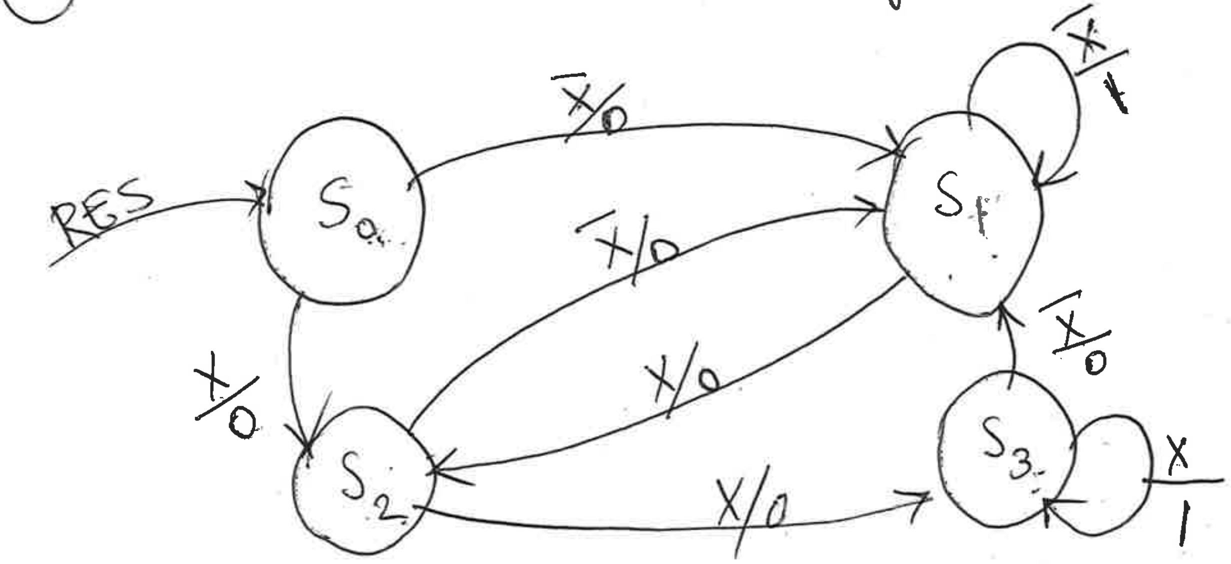
Example :

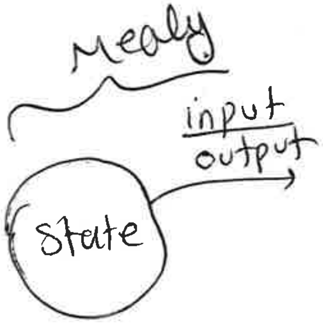
X	0	0	1	1	1	1	1	0	0	0	1	1	1	0	0
Z	0	1	0	0	1	1	1	0	1	1	0	0	1	0	1

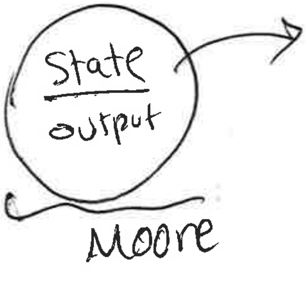
① State def. table

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 =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_1Q_0 / X	00	01	11	10
0				
1	1	1	1	1

$Q_1^+ = X$

K-Map Q_0^+

Q_1Q_0 / X	00	01	11	10
0	1	1	1	1
1			1	1

$\overline{X}$

Q_1

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

For output Z

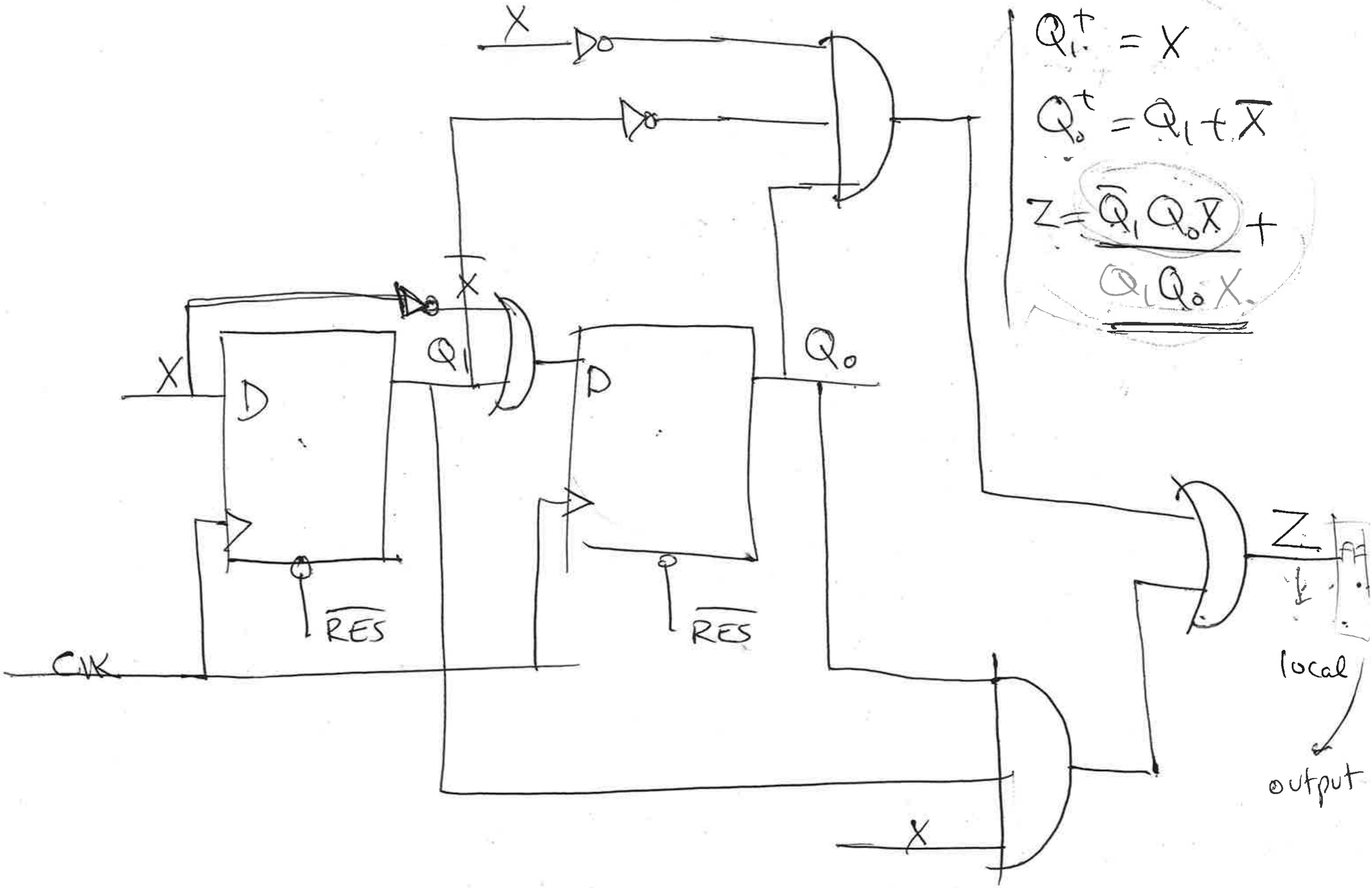
Q_1Q_0 / X	00	01	11	10
0		1		
1			1	

$\overline{Q}_1 Q_0 \overline{X}$

$Q_1 Q_0 X$

$Z = \overline{Q}_1 Q_0 \overline{X} +$

$Q_1 Q_0 X$



$Q_1^+ = X$

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

$Z = \overline{Q}_1 Q_0 \overline{X} +$

$Q_1 Q_0 X$

EEE/CSE 120 : More examples on FSMs

NOV 5, 2020

- office hrs "today" → 3:15 – 4:00 pm
- Capstone project is uploaded
- Grades are uploaded
- Assignment 6 is uploaded.
- Quiz 2 → NOV 12 → { Registers, timing diagrams }
  - D flip flops
  - JK flip flops, Toggle flip flops
  - behavioral eq.

Q: Design an alarm system.

$$\text{Alarm disarmed} \xrightarrow{010} \text{Alarm armed}$$

$$\text{Alarm armed} \xrightarrow{010} \text{Alarm disarmed}$$

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

- Inputs : $\begin{cases} e : \text{input value} \\ V = \begin{cases} 0 & \text{If } c \text{ is invalid} \\ 1 & \text{If } c \text{ is valid} \end{cases} \end{cases}$ ⇝ translation : press a key $\Rightarrow V = 1$ otherwis $V = 0$

- Output = Alarm is armed = A

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

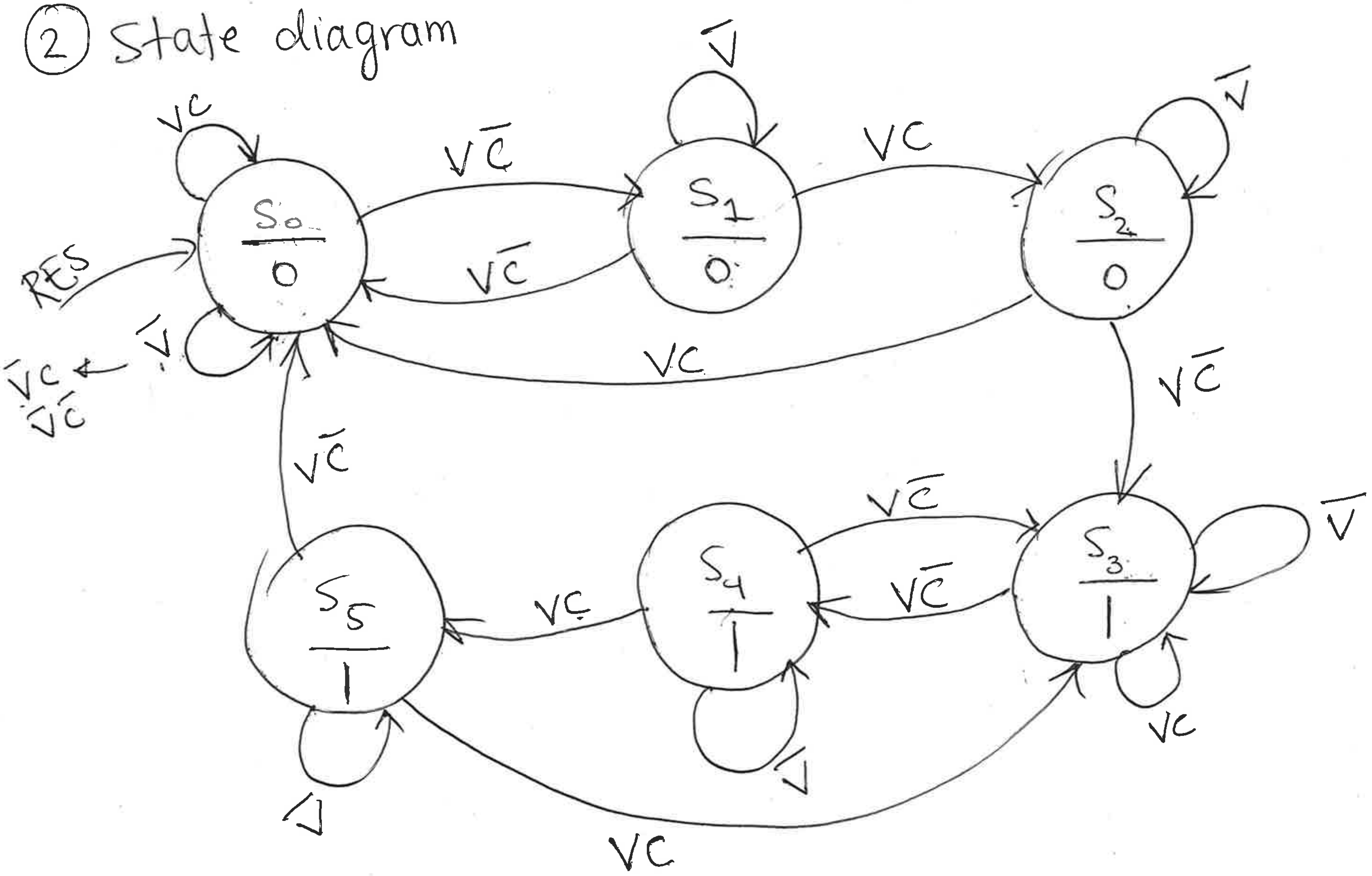
① State def. table

states	Def.	Binary rep.		
S_0	disalarmed	0	0	0
S_1	disalarmed / 0	0	0	1
S_2	disalarmed / 01	0	1	0
S_3	armed	1	0	0
S_4	armed / 0.	1	0	1
S_5	armed / 01.	1	1	0
		Q_2	Q_1	Q_0

3 flip flop

output $= Q_2$

② State diagram



③ State transition table

output only depends on the current state

Q_2	Q_1	Q_0	V	C	Q_2^+	Q_1^+	Q_0^+	A
0	0	0	0	1	0	0	0	0
	S_0							
0	0	0	1	0	0	0	1	0
	S_0							
0	1	0	1	0	1	0	0	0 only depends on S_2
	S_2					S_3		
1	0	1	1	1	1	1	0	1
	S_4					S_5		
0	1	1	1	1	X	X	X	X
1	1	1	0	0	X	X	X	1

$$\boxed{A=Q_2}$$

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

(A) Overlapping is allowed

(B) overlapping is "NOT" allowed.

$0\ 1\ \overset{\downarrow}{0}\ 1\ 0$ (overlapping is allowed)

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

Hint :

State	Def	binary rep.
S_0	IdLE	00
S_1	get 0	01
S_2	get 1	10
S_3	unused	11

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 "0 1 0"

A) overlapping is allowed

B) overlapping is "NOT" allowed

0 1 0 1 0 (overlapping)

$\underbrace{0\,1\,0}\;\underbrace{0\,1\,0}$ (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	1
		Q_1	Q_0

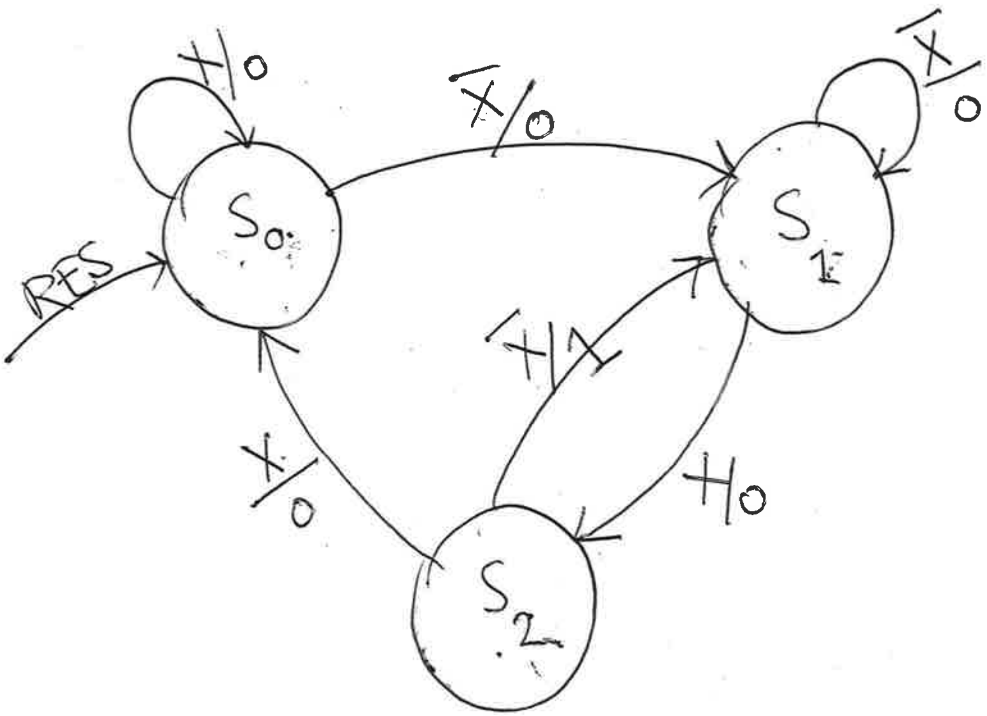
4 States ⇒ 2 FFs

⇓

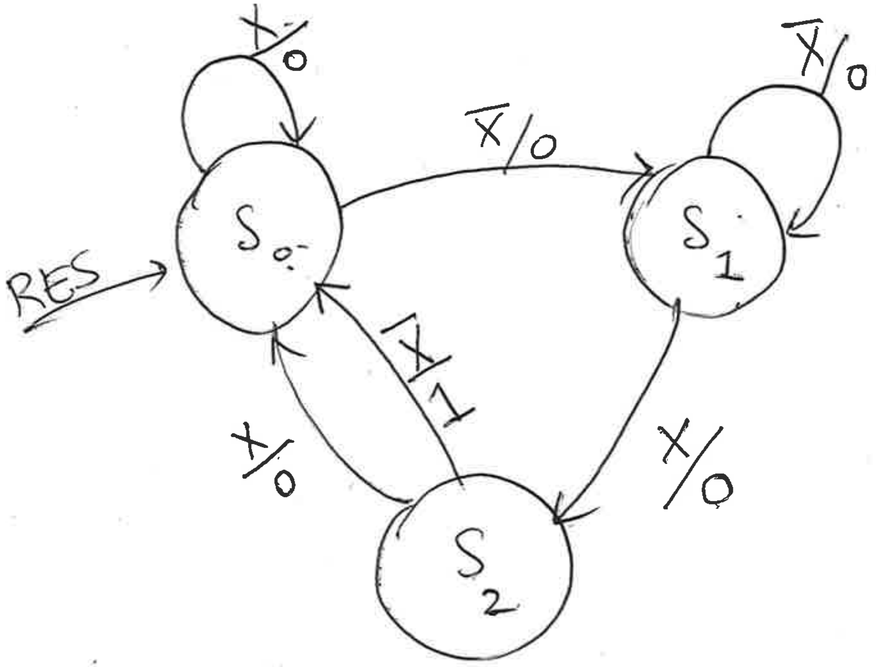
$Q_1 \; Q_0$

(2) Draw State diagram.

Overlapping



Non-overlapping



3 State transition table

sequence detector

overlapping

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

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

④ Design the circuit

Over lapping :

K-MAP For Q_1^+

$X \backslash Q_1Q_0$	00	01	11	10
0			x	
1		1	x	

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

K-MAP for Q_0^+

$X \backslash Q_1Q_0$	00	01	11	10
0	1	1	X	1
1			X	

$$\boxed{Q_0^+ = \overline{X}}$$

K-MAP for output

$X \backslash Q_1Q_0$	00	01	11	10
0			X	1
1			X	

$$\boxed{\text{output} = Q_1 \overline{X}}$$

Non-overlapping

K-MAP for Q_1^+

X \ Q_1Q_0	00	01	11	10
0			x	
1		1	x	

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

K-MAP for Q_0^+

X \ Q_1Q_0	00	01	11	10
0	1	1	X	
1			X	

$$\boxed{Q_0^+ = \overline{Q_1}\,\overline{X}}$$

K-MAP for output

X \ Q_1Q_0	00	01	11	10
0			X	1
1			X	

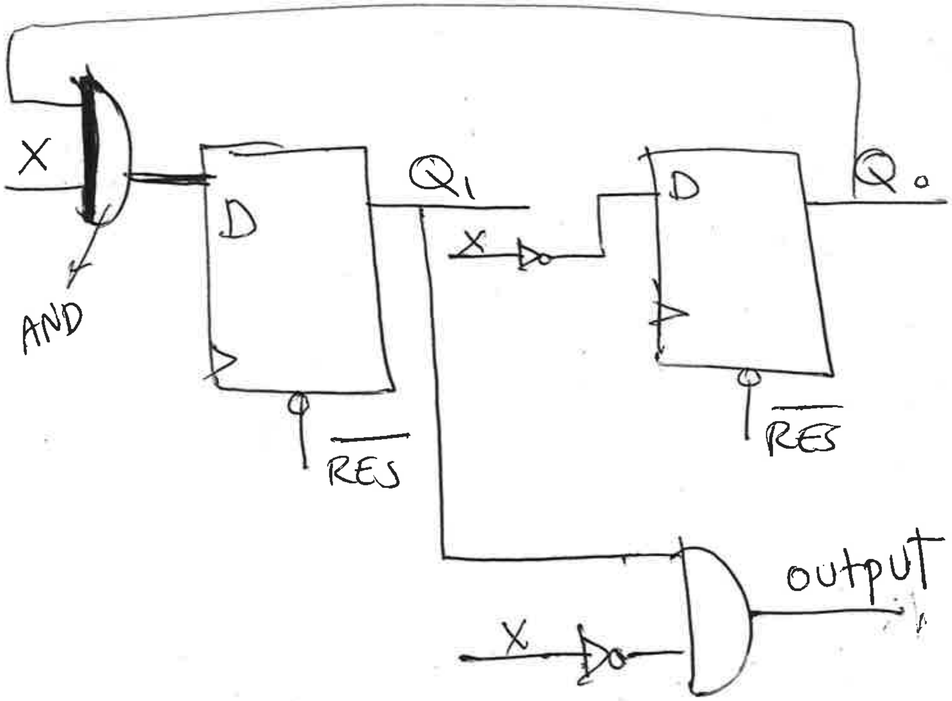
$$\boxed{\text{output} = Q_1 \overline{X}}$$

overlapping

$Q_1^+ = Q_0 X$

$Q_0^+ = \overline{X}$

output $= Q_1 \overline{X}$

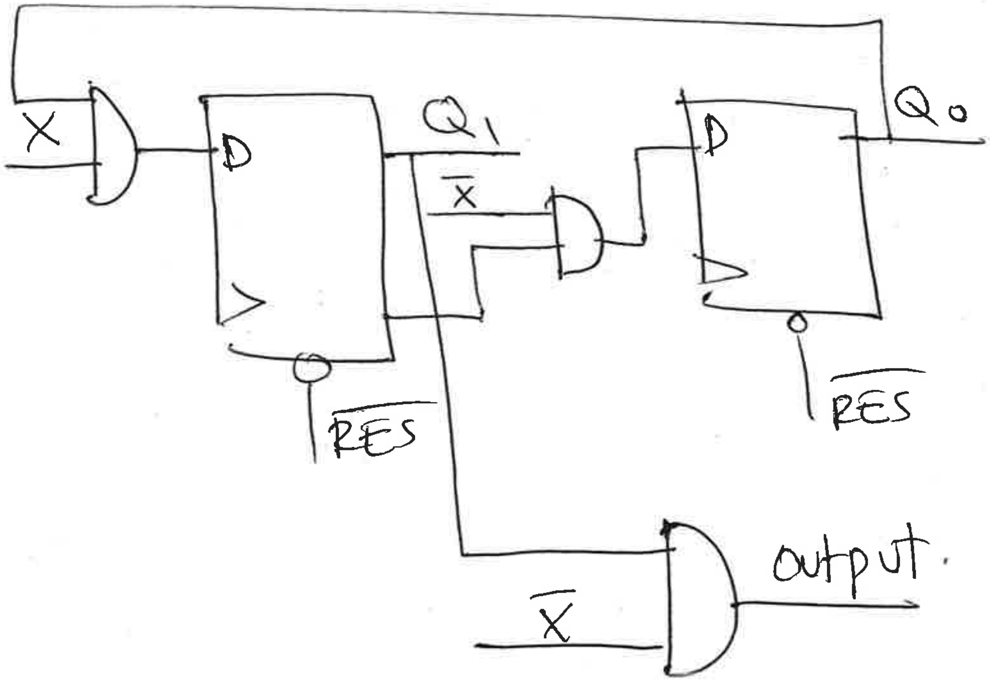


Non-overlapping

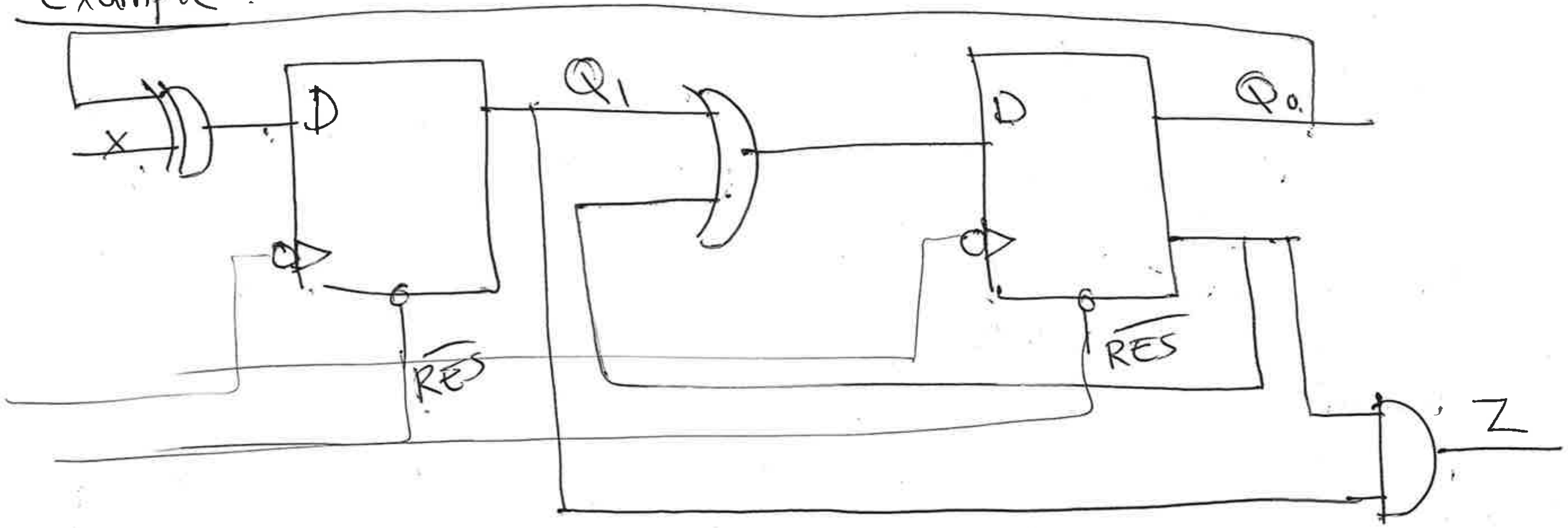
$Q_1^+ = Q_0 X$

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

output $= Q_1 \overline{X}$



Example :



① Behavioral eq.

$Q_1^+ = Q_0 \oplus X$

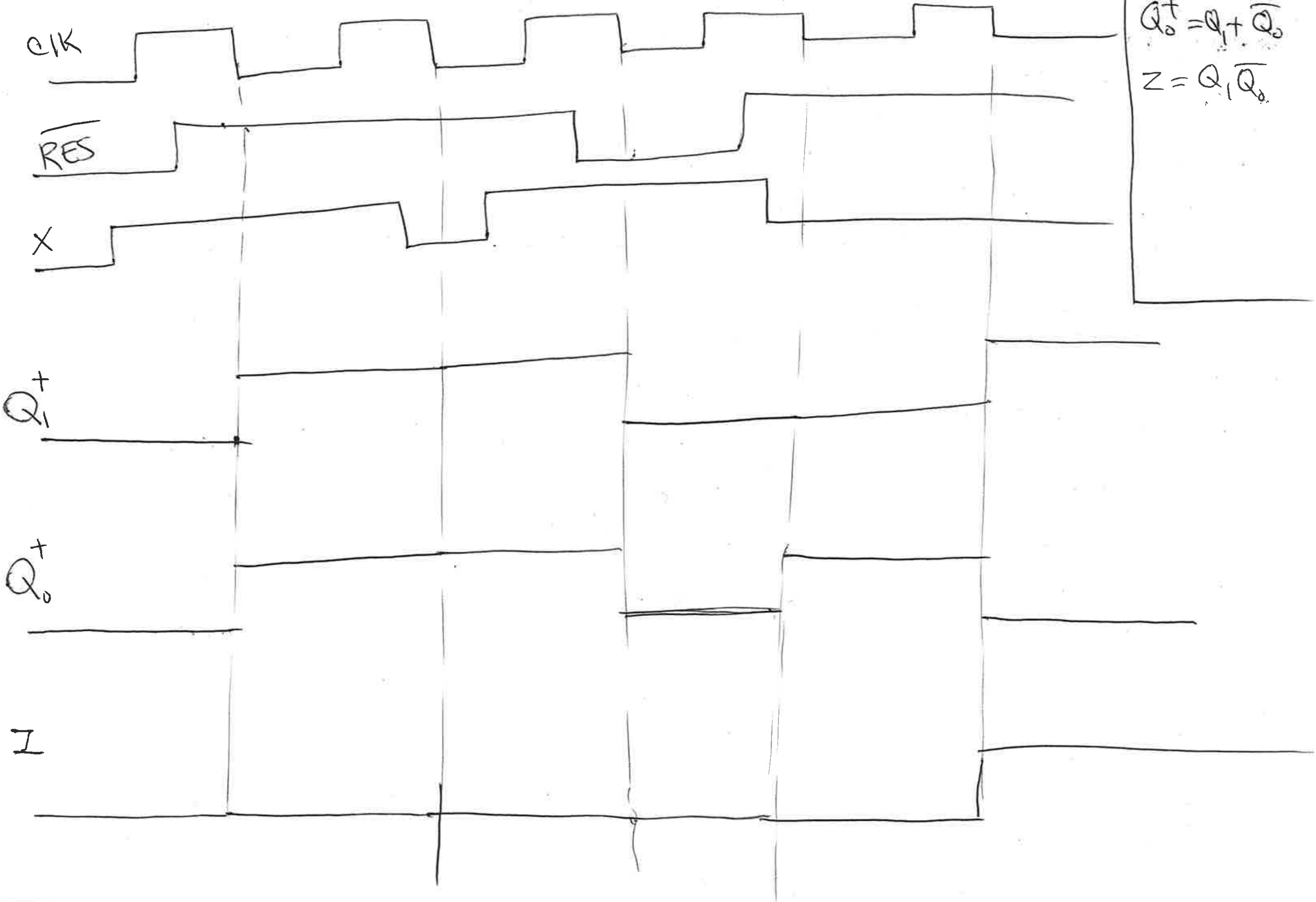
$Q_0^+ = Q_1 + \overline{Q_0}$

$Z = Q_1 \overline{Q_0}$

$\overline{CLR} = 0$ ⟹ output = 0

$\overline{PRE} = 0$ ⟹ output = 1

② Complete timing diagram.



$Q_1^+ = Q_0 \oplus X$

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

$Z = Q_1 \overline{Q_0}$

EEE/CSE 120 : CPU

Lecture 24 NOV 17

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.

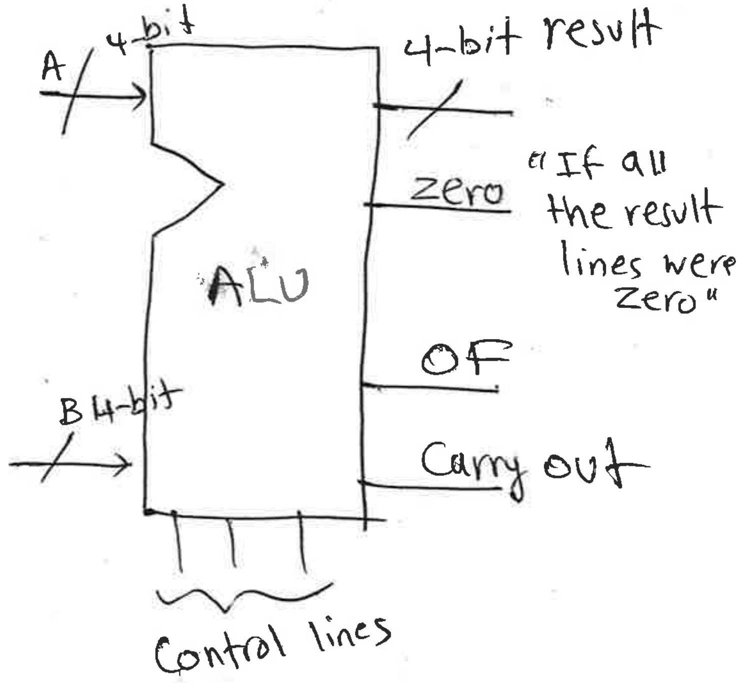
CPU : central Processing unit

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

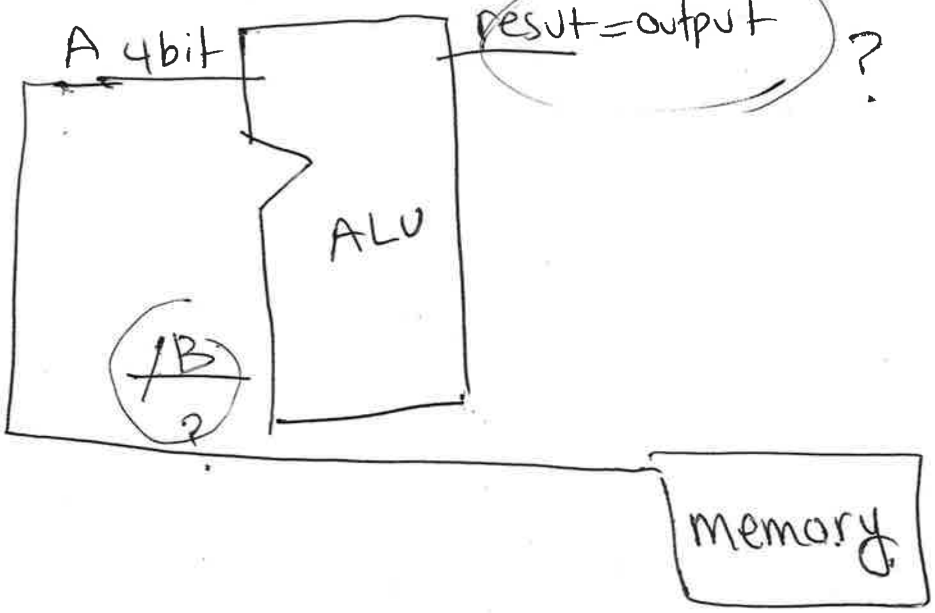


② Memory is needed to build a CPU.

↳ – 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



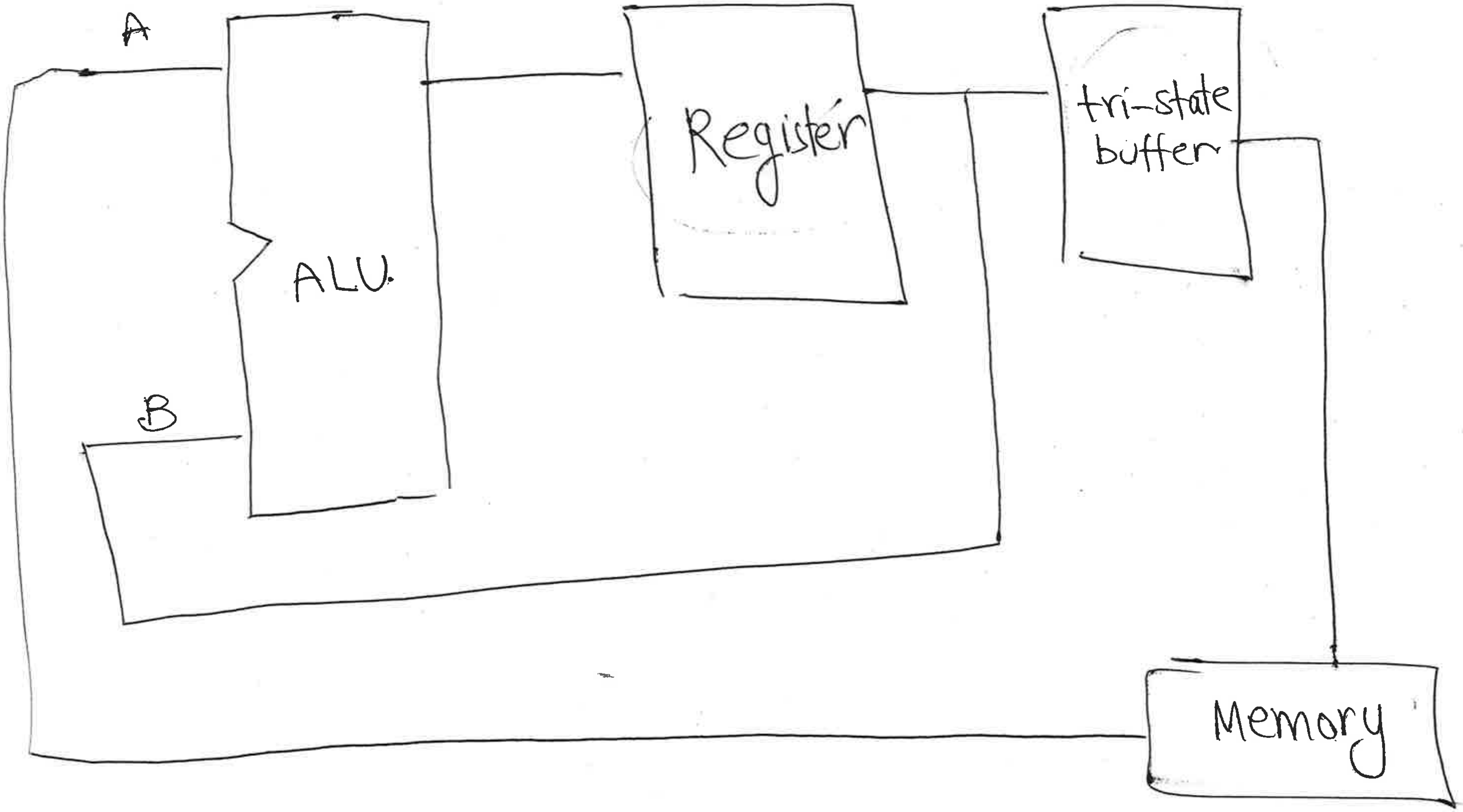
Designing CPU :

① # of connections is minimized

② Temporary storage (Registers)

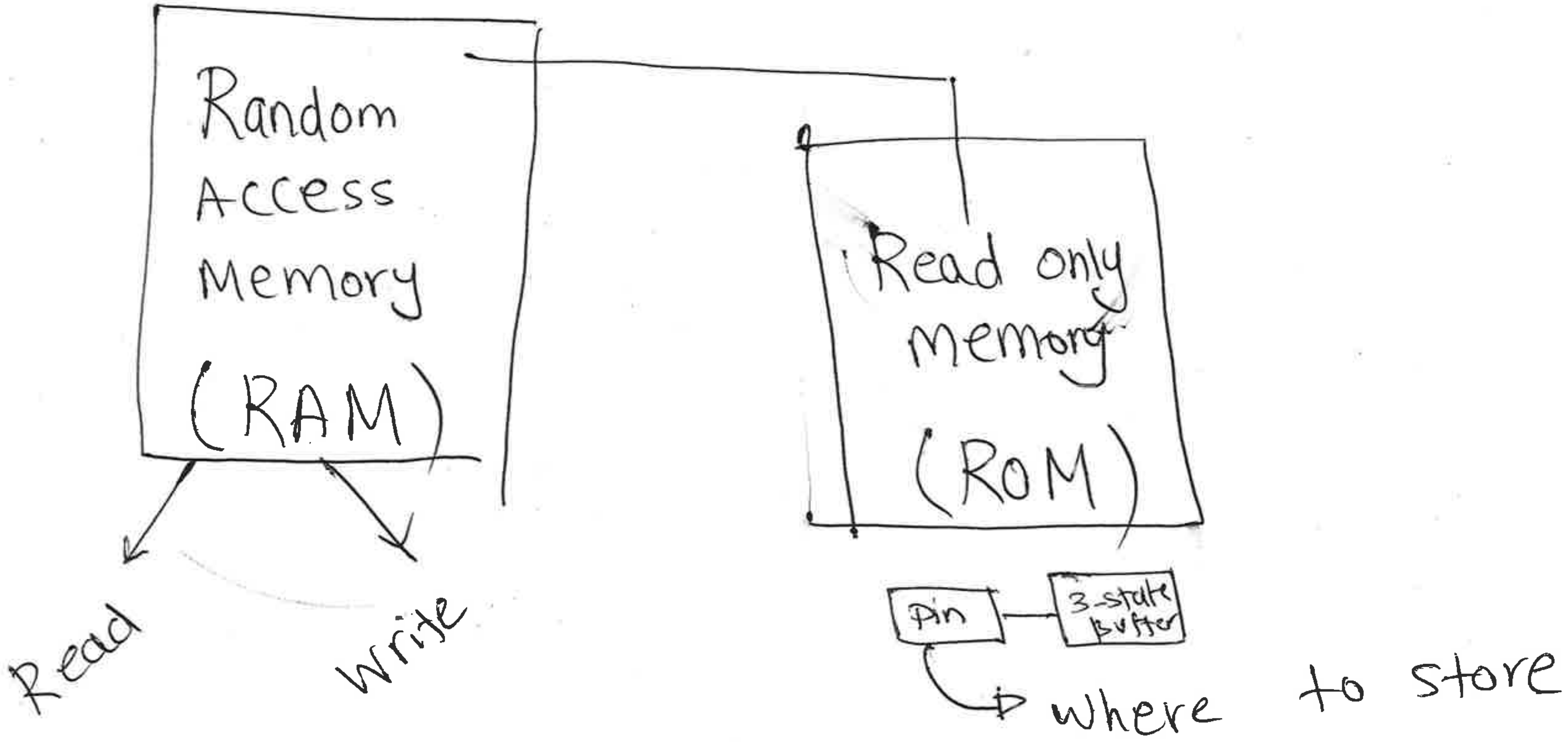
③ use loopback connectin to the input B. (Multiplexing our connection)

↙ tri-state buffer.

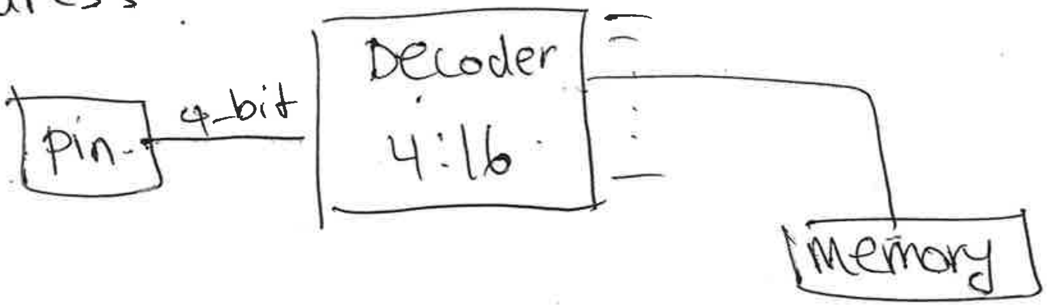


Accumulator.

How to address in a memory ?



How to address ?

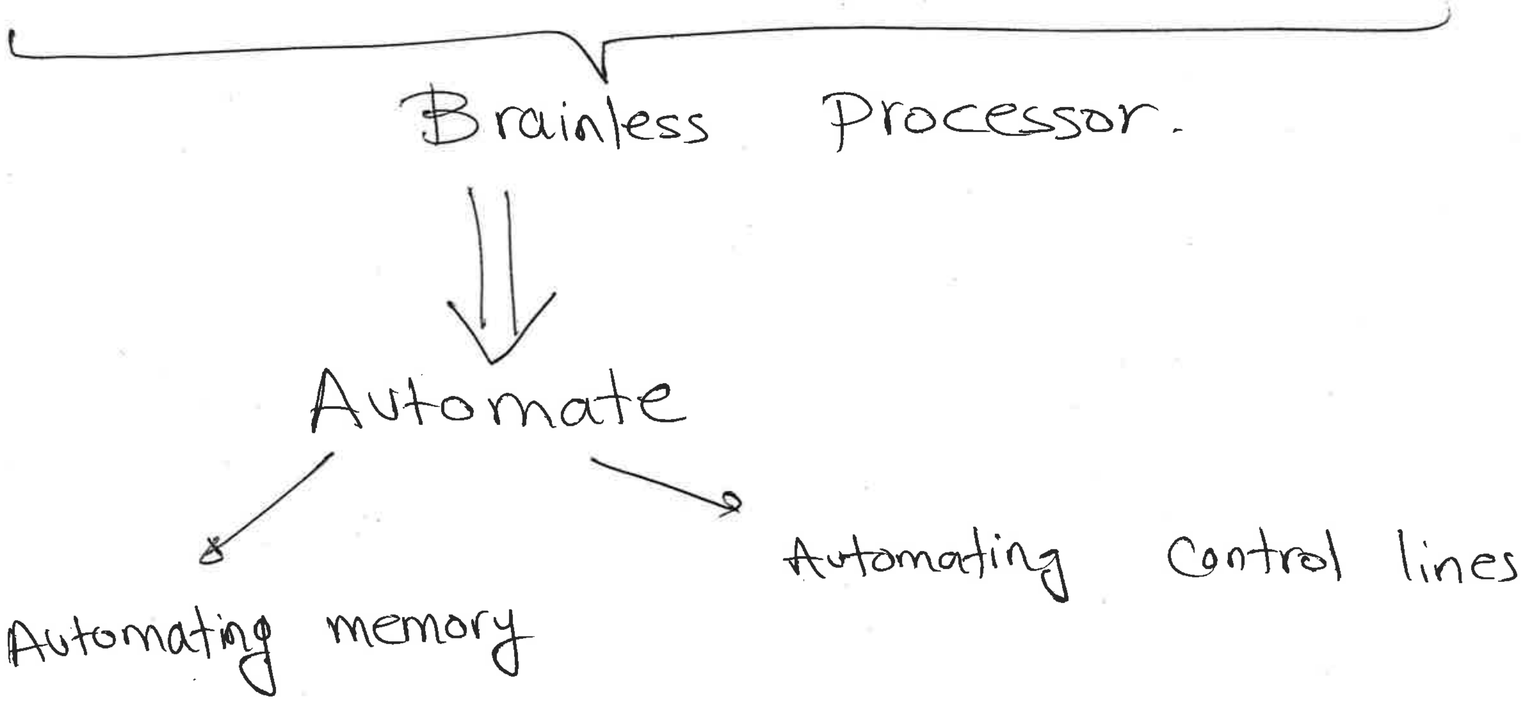


Summary:

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.

↓ op-code (operation Code)

↳ Hexa # that forces the Control line to be what's needed.

Encode the Control line

↓ use decoder

translate op-code into actual Control Signal

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 Comman 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 seperate instruction memory and seperate 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 princton 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

1 0 0 1 1 0 0 1

↑

= 0 1 1 0 0 1 1 1

0 1 0 1 1 0 0

↑

= 1 0 1 0 1 0 0

Find $(-21)_{10}$

- Find binary rep. of (21)

$$\begin{array}{ccccc} 21 \vdash \overline{2} & & & & \\ & 10 \vdash \overline{2} & & & \\ \underline{} & & 5 \vdash \overline{2} & & \\ 1 & \underline{} & & 2 \vdash \overline{2} & \\ & 0 & \underline{} & & 1 \\ & & 1 & \underline{} & \\ & & & 0 & \end{array}$$

$$\begin{tikzcd} 0\,1\,0\,1\,0\,1 \arrow[d, "2\text{'s Complement}"] \\ 1\,0\,1\,0\,1\,1 \end{tikzcd}$$

↑

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

← Verification.

② Add these 2's Complement <u>Signed</u> binary numbers

$\boxed{0}$ cin

$1\ 0\ 0\ 0 \rightsquigarrow -8$

$+\ 1\ 0\ 1\ 0 \rightsquigarrow 0110 \rightsquigarrow +6 \rightsquigarrow -6$

$\boxed{1}\ \underbrace{0\ 0\ 1\ 0}_{+2} \neq -14$

Cout

OF

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

$\boxed{0}$

$\quad\quad 1$

$1\ 0\ 0\ 1 \rightsquigarrow (0111) \rightsquigarrow -7$

$+\ 0\ 1\ 0\ 1 \rightsquigarrow 5$

$\boxed{0}\ 1\ 1\ 1\ 0 \qquad -2 \leftrightarrow (0010) \rightsquigarrow +2 \rightsquigarrow -2$ ✓

$C_{in} = C_{out} \Rightarrow$ OF X

③ Do Conversion.

- $(\overset{}{\underset{\uparrow}{2}}\,\underset{\uparrow}{0}\,\underset{\uparrow}{3})_{16}$ ⇝ Decimal ?

$$3\times16^{0} + 0\times16^{1} + 2\times16^{2} = 515$$

- $(186)_{10}$ ⇝ HExa ?

$$\begin{array}{r|l} 186 & \boxed{16} \\ \cline{2-2} & \boxed{11} \\ \hline 10 & \end{array} \qquad (BA)_{16}$$

- $(\underbrace{0001}_{1}\,|\,\underbrace{1010}_{A}\,|\,\underbrace{0011}_{3}\,|\,\underbrace{0111}_{7})_{2}$ ⇝ Hexa ?

$$(1\;A\;3\;7)_{16}$$

$*(3A09)_{16} \rightsquigarrow$ Binary

$$\begin{tikzcd}
3 \arrow[d] & A \arrow[d] & 0 \arrow[d] & 9 \arrow[d] \\
0011 & 1010 & 0000 & 1001
\end{tikzcd}$$

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

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

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

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

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

$\left(ab + c \overset{\uparrow}{=} (a+c)(b+c)\right.$ distributivity $\left.\right)$

$$= \left((\bar{a} + \bar{b})\cancelto{1}{(b + \bar{b})}\right)c + \bar{c}$$

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

$\left((a+b)c = ac + bc\right)$

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

$$= \bar{a}c + (\bar{b} + \bar{c})\left(\cancelto{1}{c + \bar{c}}\right)$$

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

$$= (\bar{a} + \bar{c})\cancelto{1}{(c + \bar{c})} + \bar{b} = \bar{a} + \bar{b} + \bar{c} \overset{\text{Demorgan's}}{=} \overline{abc}$$

* Find Canonical form for SOP.

	a	b	c	$f(a,b,c) = \overline{abc}$	
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	← $\bar a b c$
4 →	1	0	0	1	← $a \bar b \bar c$
5 →	1	0	1	1	← $a \bar b c$
6 →	1	1	0	1	← $a b \bar c$
7 →	1	1	1	0	↑ $\bar a + \bar b + \bar c$

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

* Find Canonical form for POS.

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

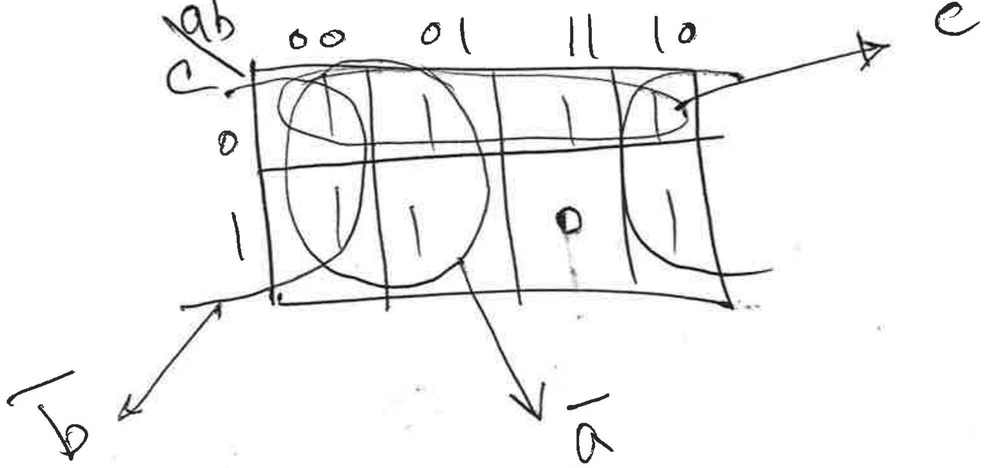
* Find min-term expression

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

* Find the max-term expression

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

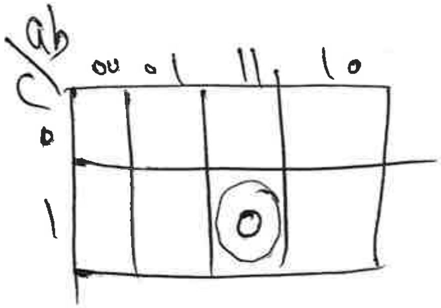
* Find the minimum Sop:



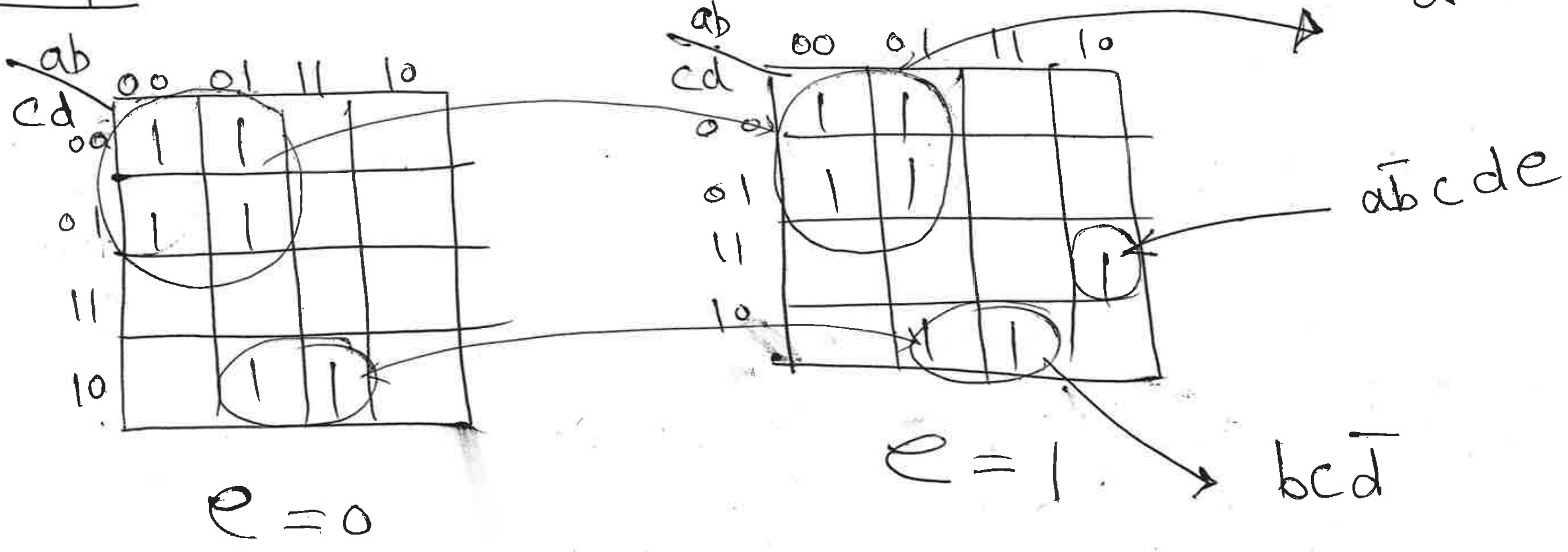
$$f(a,b,c) = \bar{a} + \bar{b} + \bar{c}$$

* Find the minimum product of sum.

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



5 Example:



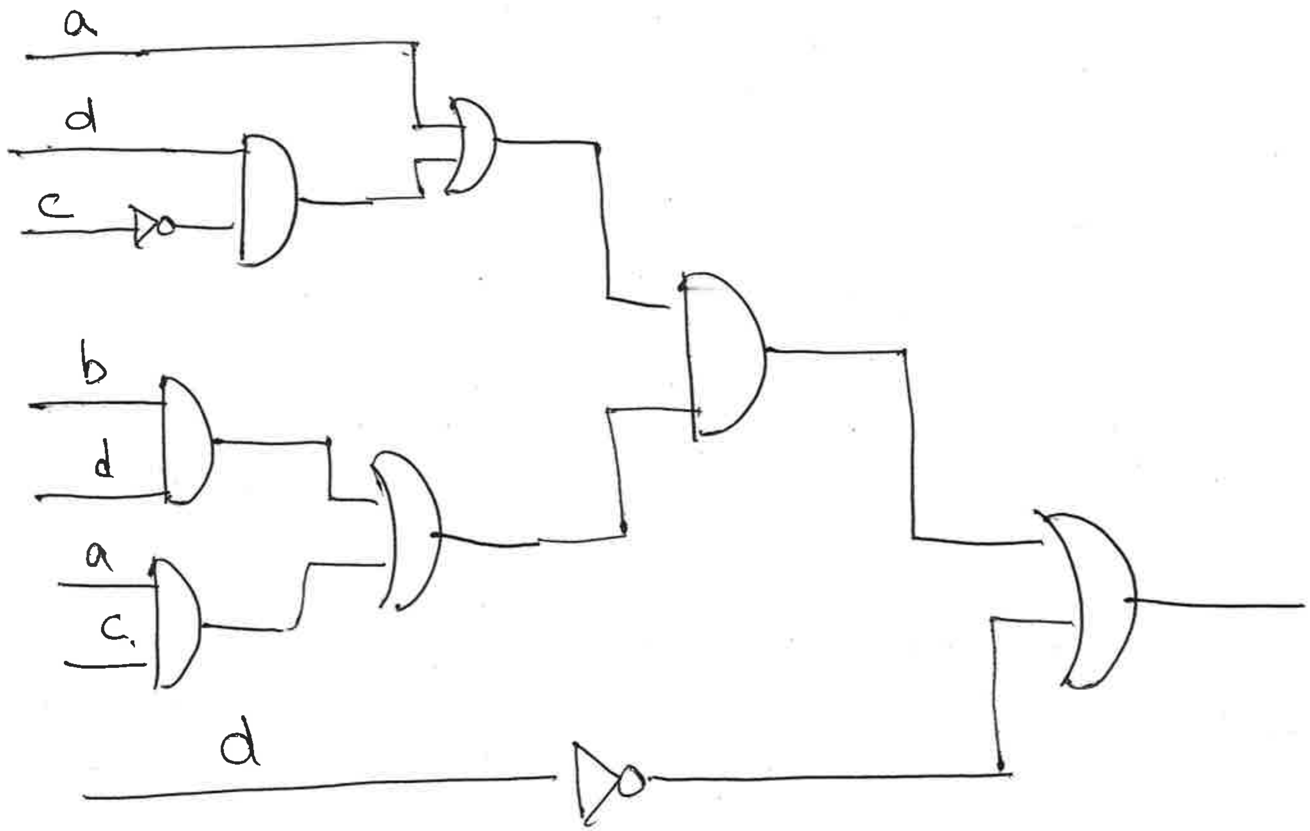
$f(a,b,c,d,e) = ?$

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

⑥ — • Design a circuit that represents a function

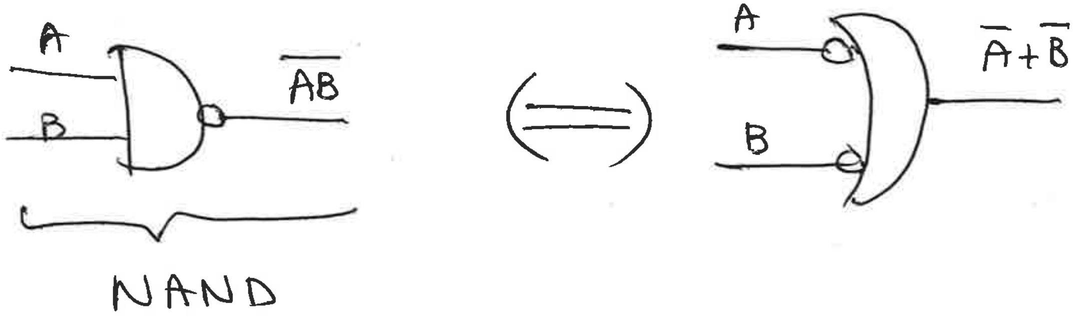
• Design a circuit using only "NAND" or "NOR".

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



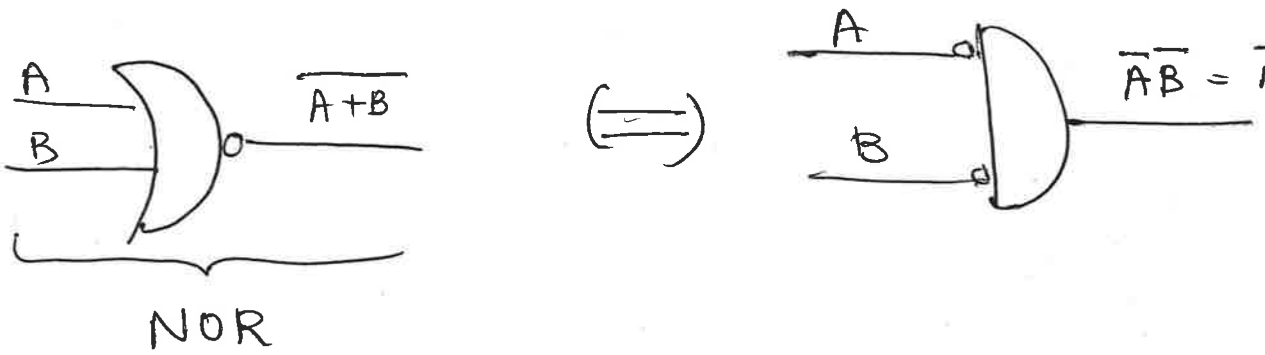
- Build a circuit using "NAND" / "NOR"
  - Build the circuit the way it is
  - Play the bubble game

①



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

②



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

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 → can do it through Zoom or any app you'd like

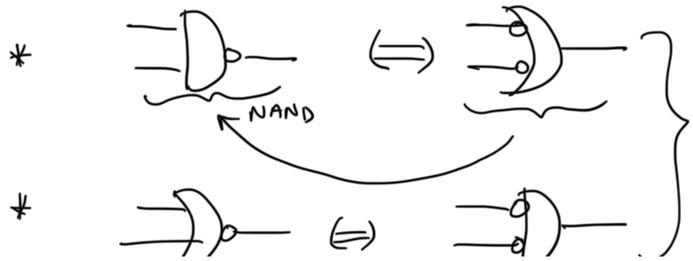
■ Make sure you've read the info for final before taking the exam.

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



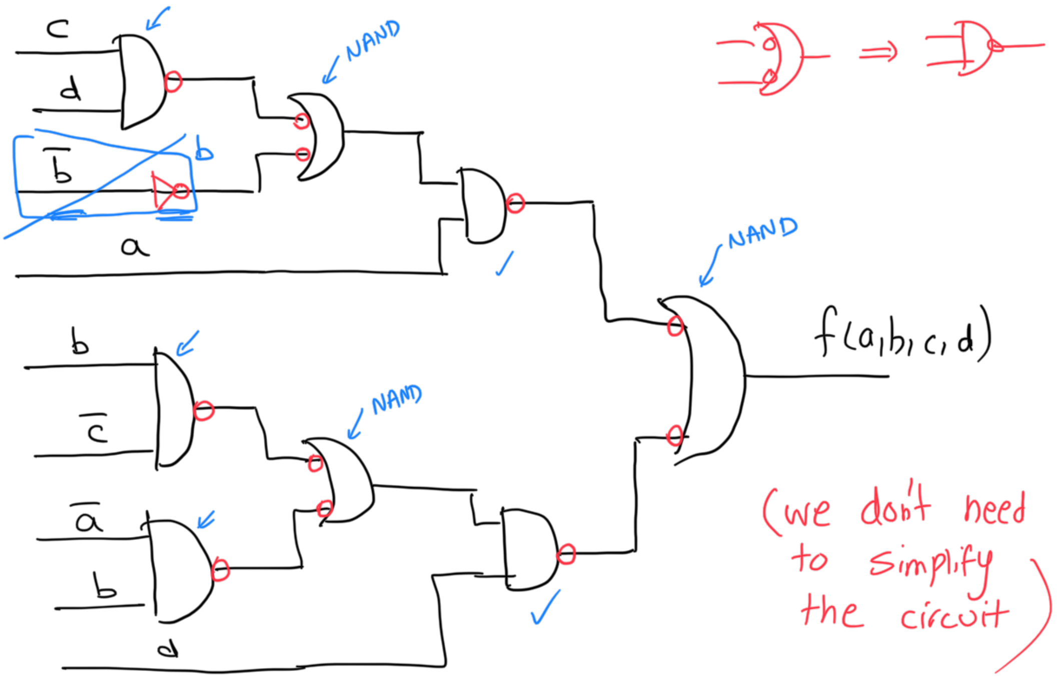
Demorgan's law.

Example 7: Implement $f(a,b,c,d) = a(\underbrace{cd+\bar{b}}) + d(\underbrace{b\bar{c}+\bar{a}b}$

using only "NAND" gates.

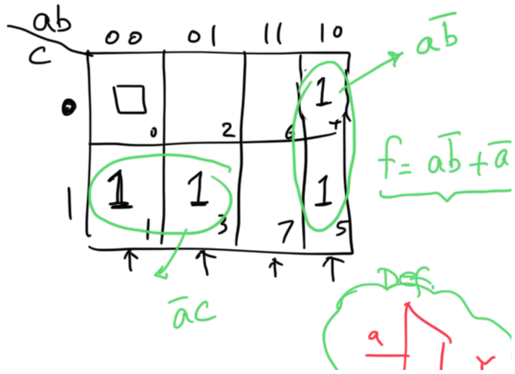
— Assumption: both Complemented and uncompleme

Variables are available.



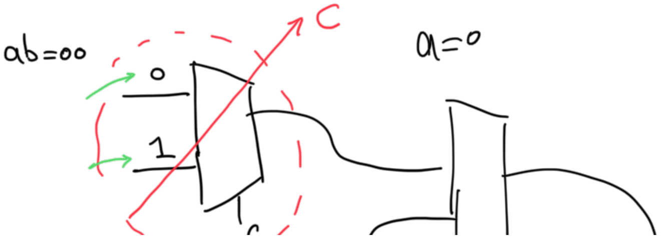
(we don't need to simplify the circuit)

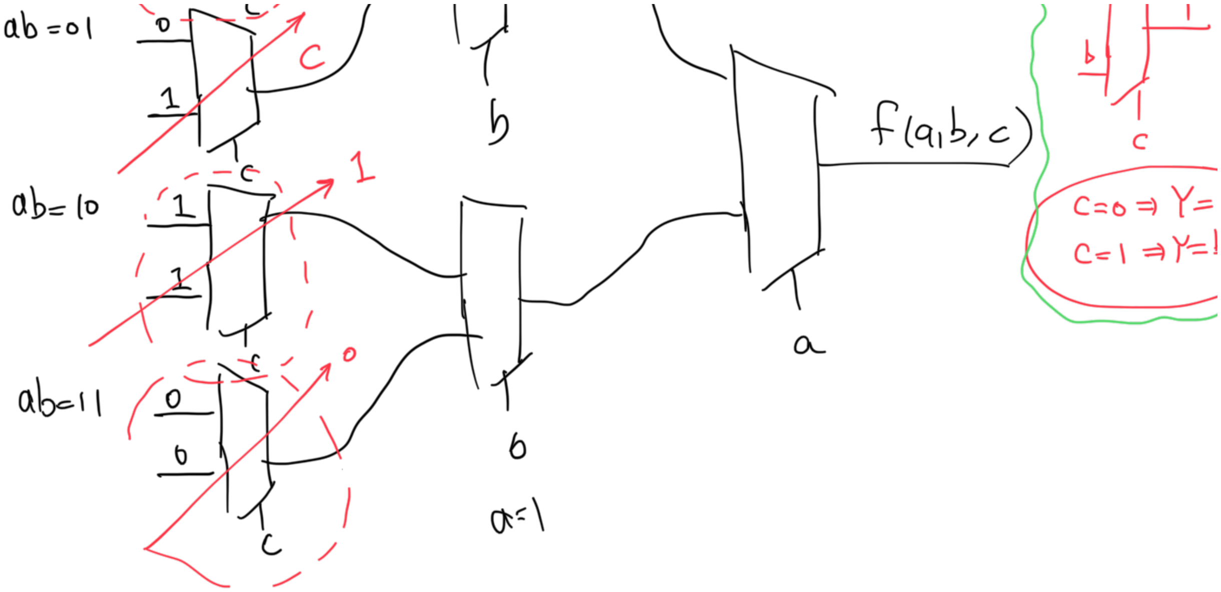
Example 8: Implement $f(a,b,c) = \sum m(1,3,4,5)$ using 2:1 MUXs.

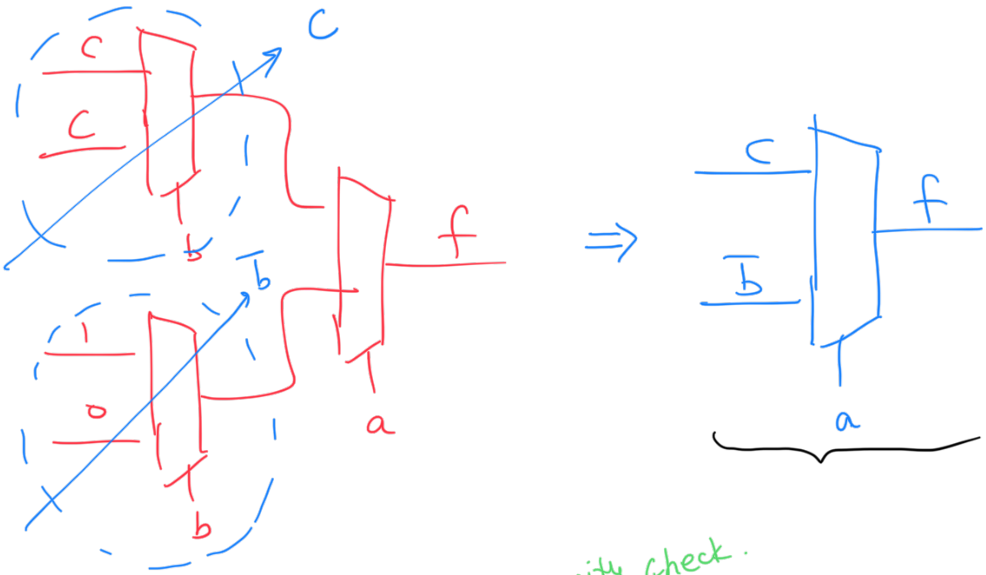


c \ ab	00	01	11	10
0	□			1
1	1	1		1

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

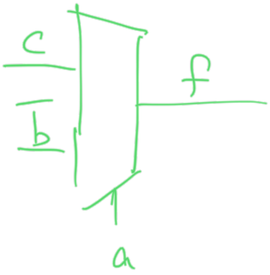






sanity check.

$$f = \underbrace{\underline{\underline{a}}\,\bar{b} + \underline{\underline{\bar{a}}}\,c}_{\text{eq. for 2:1 mux.}} \Rightarrow$$



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

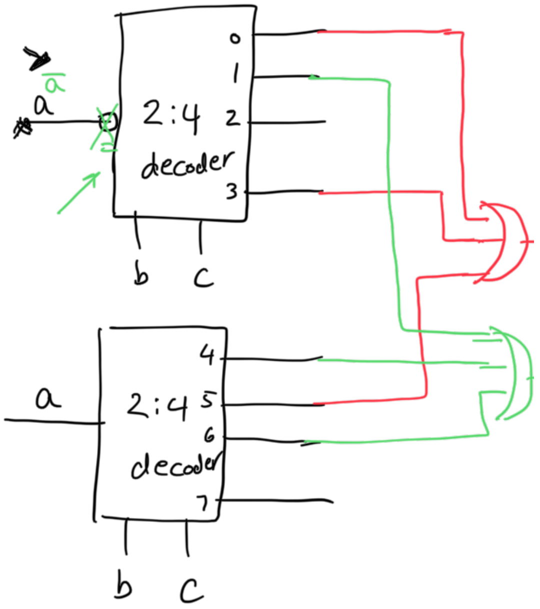
$g(a,b,c) = \sum m(1,4,6)$

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

Assumption : • we have access to both Complemented and uncomplemented variab

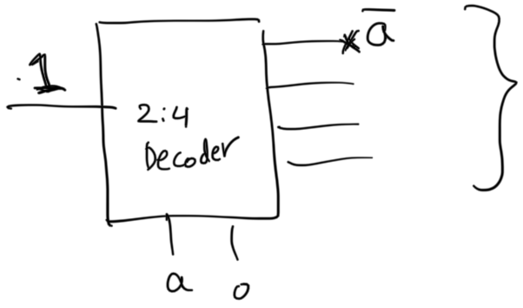
• use Active high decoders.

	input				
#line	a	b	c	f	g
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 $\bar{a}$. ⇝ 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" th is not shared between that group and others.

Moore Machine : A FSM where the output doesn 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 for the clock depending on the input.
- Mealy Machine : output changes immediately w/ the input change.
- Mealy machine we have timing issue ⟹ Synchronization FF.

Functionally Compelete : A gate is called functionally compelete if we could use that gate to build {AND, OR, NOT}

Example : Is "OR" functionally Compelete?

↪ Cannot build "NOT" gate

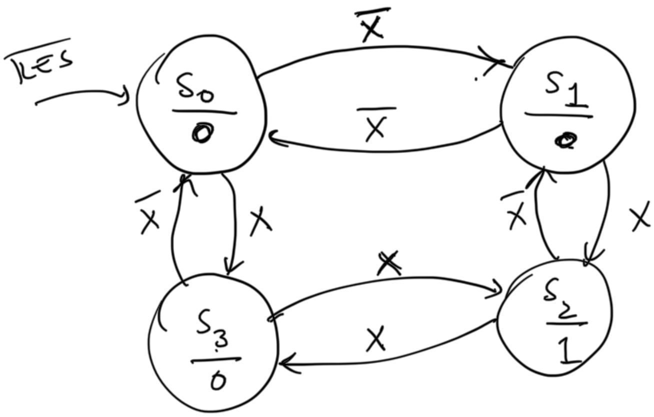
⟹ Not functionally Compelete

<u>Example</u>: Are "NAND" and "NOR" functionall Compelete?

both NAND, NOR are functionally compelete (Justify)

<u>Example 11</u>: Design a FSM (Moore / Mealy)

<u>Example</u>: you're given the following diagram



1) what type machine doe this diagram represent?

"Moore"

↓

the output only depends on the current state an not the next st

2) How many ffs do we need? $2^{\# \text{FFs}} = \# \text{states}$ ⟹ 2 FFs

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
				↑	↑	↑

<u>Example</u>: Design a <u>Mealy machine</u> 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

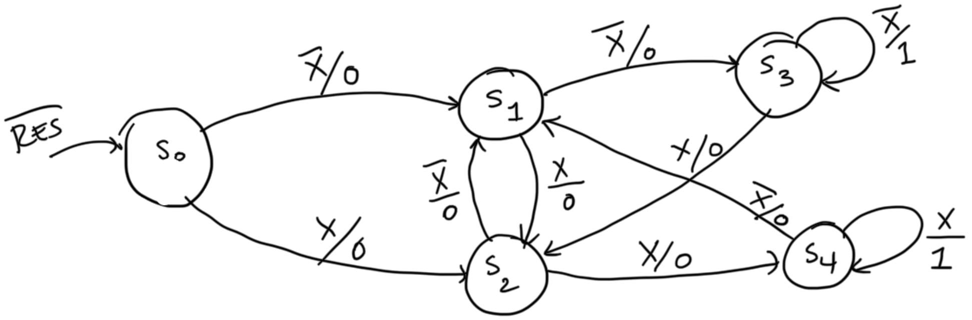
<u>Step1</u>: State def. table

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

# of ffs : $2^3 = \underline{\underline{8}} \longrightarrow 3$ ffs $\rightsquigarrow Q_2 Q_1 Q_0$

Step 2: Draw the transition diagram

Step 2 : Draw the transition diagram.



Step3 : 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	x	α	α
S_7					α	α	α	α

← unl

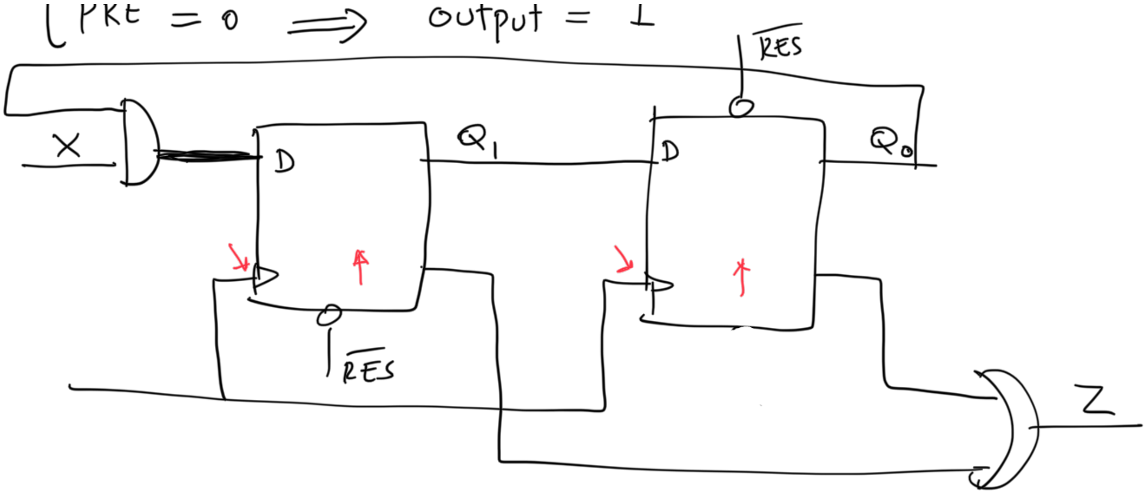
Example 12 : Timing diagram

- behaivioral eq. ⤳ Draw Circuit ✓
- Circuit ⤳ behaivoral eq. ⤳ Timing diagram

$\overline{\text{CIR}} = 0 \Longrightarrow$ output $= 0$

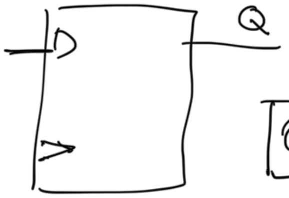
$\overline{\text{PRE}}$ 1

(PRE = 0 ⟹ output = 1



① Behaivioral eq:

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

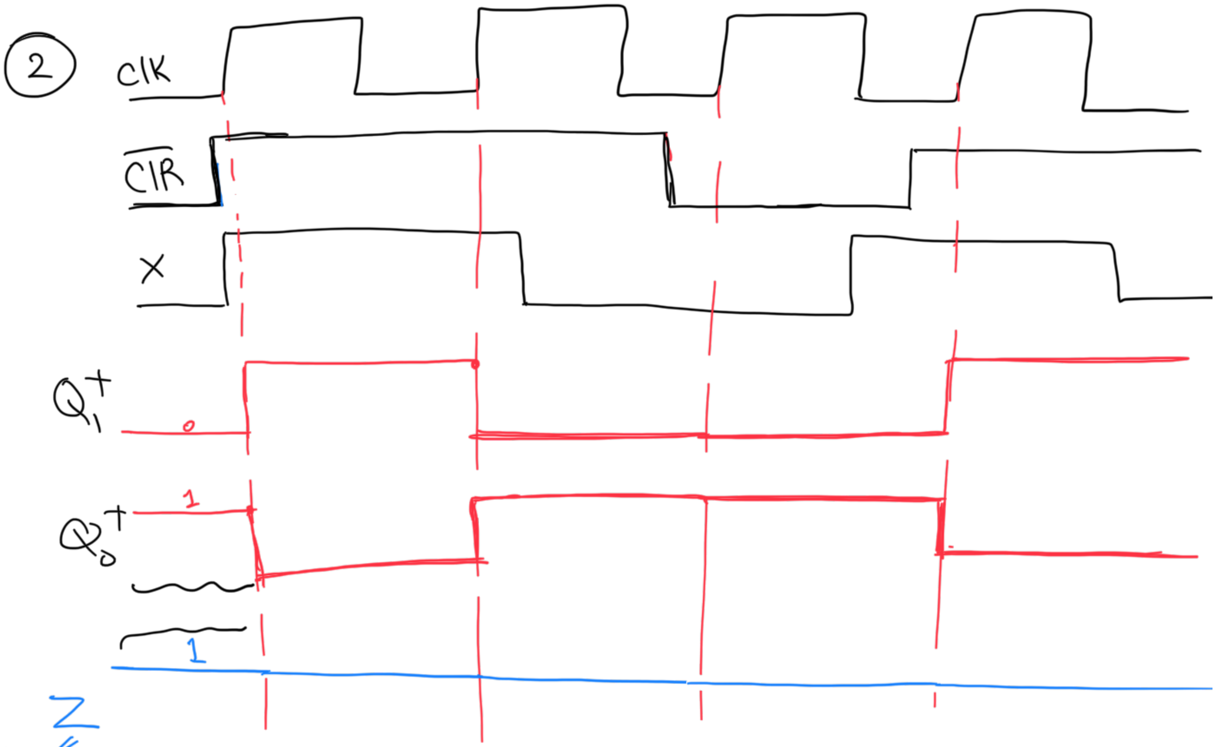


$\boxed{Q^+ = D}$

① Determine PE ff o
NE ff.

② look at behuivoral e

②



Z

$\overline{Q_1} + \overline{Q_0}$ $\overline{0} + \overline{1} = 1$