

DDCO

UNIT - 2

CLASS NOTES

feedback/corrections: vibha@pesu.pes.edu

Vibha Masti



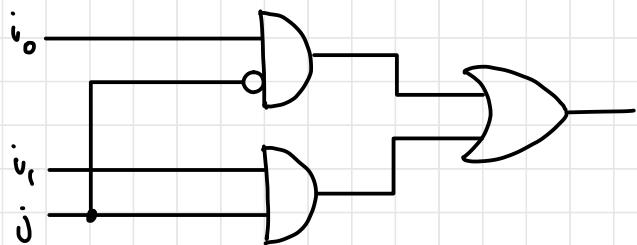
MUXERS, DECODERS & SHIFTERS

MULTIPLEXER — MUX

- multiplexes many inputs into a single output

2:1 MUX

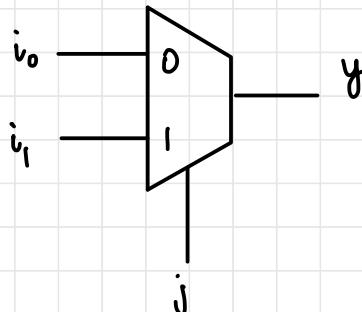
i_0	i_1	j	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



$$y = \bar{j}i_0 + ji_1$$

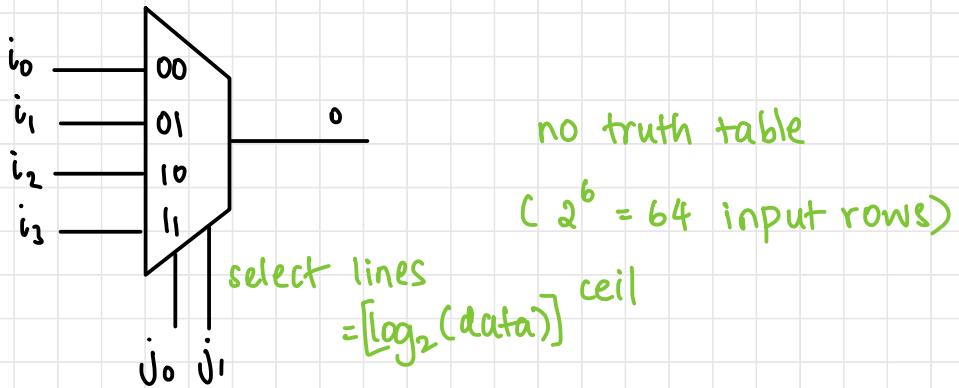
Symbol

- Acts as a selector
- if $j=0$, $O/P = i_0$
if $j=1$, $O/P = i_1$
- i_0, i_1 : data input
 j : control input

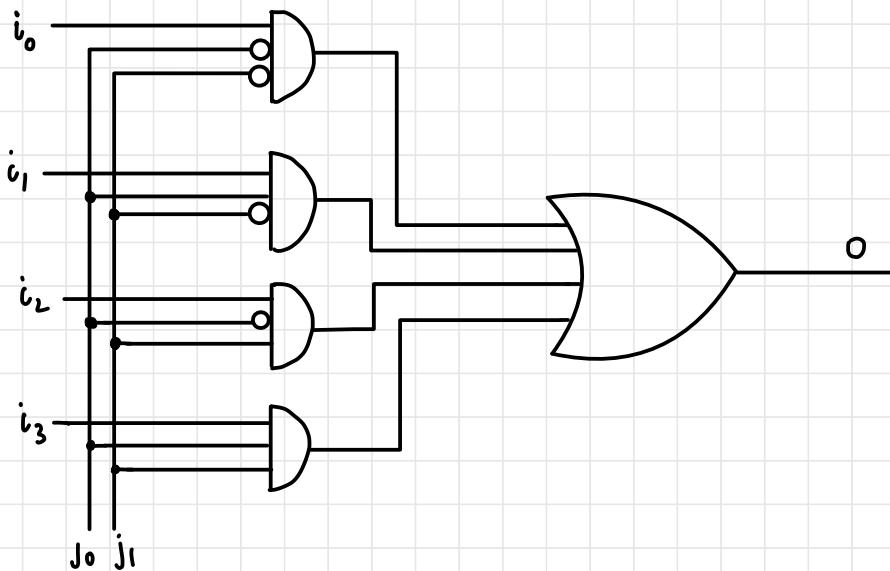


4:1 MUX

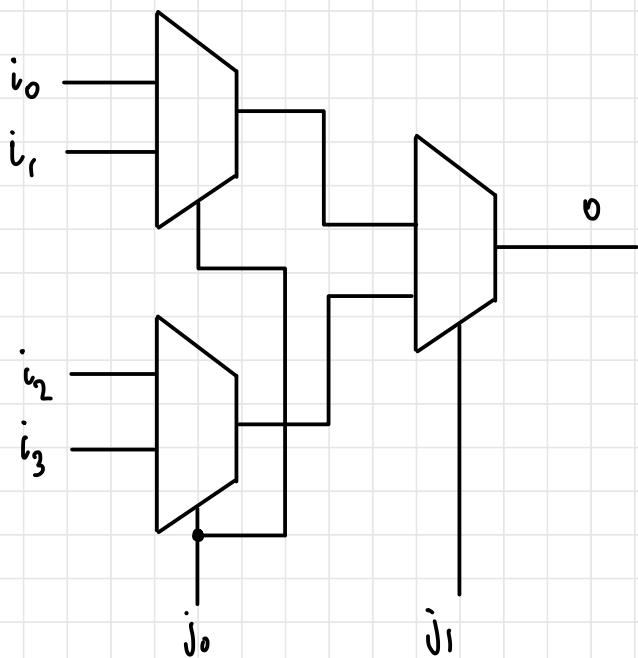
- Data inputs: i_0, i_1, i_2, i_3 2^n , $n = \text{control}$
- Control inputs: j_0, j_1 $2^n = \text{data}$



$$o = \bar{j}_1 \bar{j}_0 i_0 + \bar{j}_1 j_0 i_1 + j_1 \bar{j}_0 i_2 + j_1 j_0 i_3$$



4:1 MUX Using 2:1 MUX



if $j_1 = 0$, i_0 or i_1
 $j_1 = 1$, i_2 or i_3

n:1 MUX

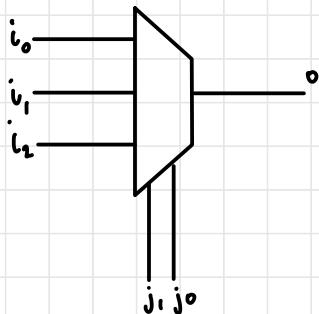
- for n data inputs, $\lceil \log_2(n) \rceil$ control inputs and one output
- data o/p to be o/p specified by control i/p
- 5-input MUX: $\lceil \log_2 5 \rceil = \lceil 2.322 \rceil = 3$
- $n:1$ MUX will have $n-1$ 2:1 MUXes

ceiling
↙

Construct 3:1 MUX using

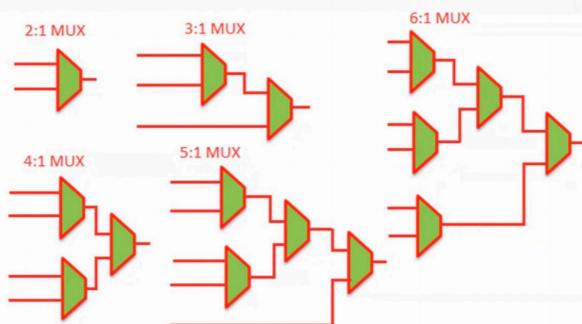
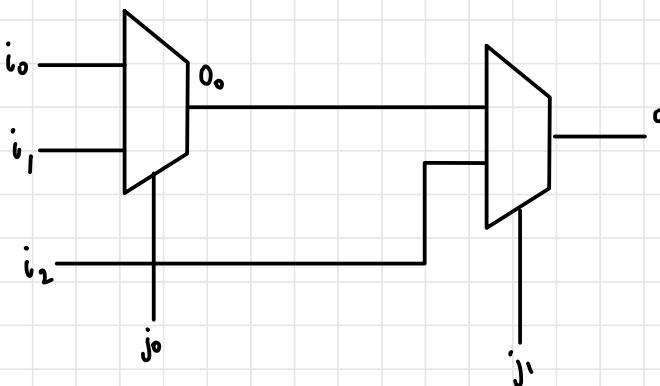
↳ 2:1 MUX

↳ AND, OR and NOT gates



$$o = \bar{j}_0 \bar{j}_1 i_0 + j_0 \bar{j}_1 i_1 + j_1 i_2$$

2:1 MUXes



DECODERS

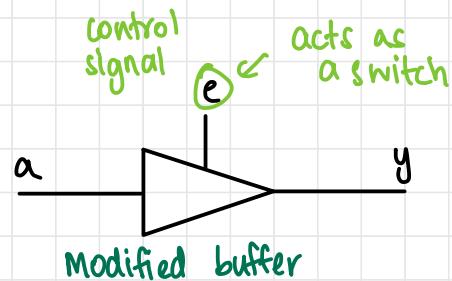
Floating Value (Z) / Tristate Buffer

- for a third state (besides 0 and 1)
- in logic circuit, 0 and 1 are voltage levels
- binary 0 — 0V, binary 1 — +1.8V (+ve voltage value)
- must be electrically connected to 0V or +1.8V using low resistance path
- What if wire disconnected from both 0V and +1.8V?
High impedance path → electrically disconnected
- neither 0 or 1; Z (third state)

Tristate Truth Table

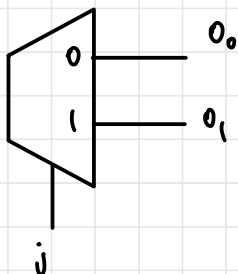
e	a	y
0	0	Z
0	1	Z
1	0	0
1	1	1

high
impedance
state



- used in implementation of shared bus
- can be used in MUXes as well

1:2 Decoder

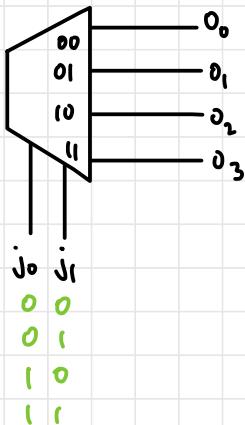


inputs: j
outputs: O₀, O₁

$$\begin{aligned} O_0 &= \bar{j} \\ O_1 &= j \end{aligned}$$

if j = 0,
O₀ = 1, O₁ = 0
if j = 1,
O₀ = 0, O₁ = 1

2:4 Decoder



$$o_0 = \bar{j}_1 \bar{j}_0$$

$$o_1 = \bar{j}_1 j_0$$

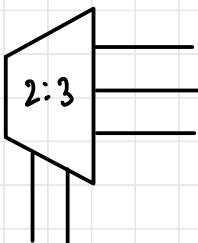
$$o_2 = j_1 \bar{j}_0$$

$$o_3 = j_1 j_0$$

j_1	j_0	o_0	o_1	o_2	o_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- all O/P's of decoder to be 0
1:4 decoder — not possible

2:3 Decoder

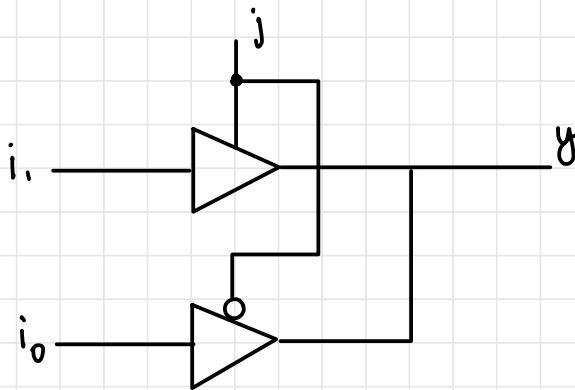


$\lceil \log_2 n \rceil : n$ decoder

- n data outputs
- $\lceil \log_2 n \rceil$ control inputs ceiling

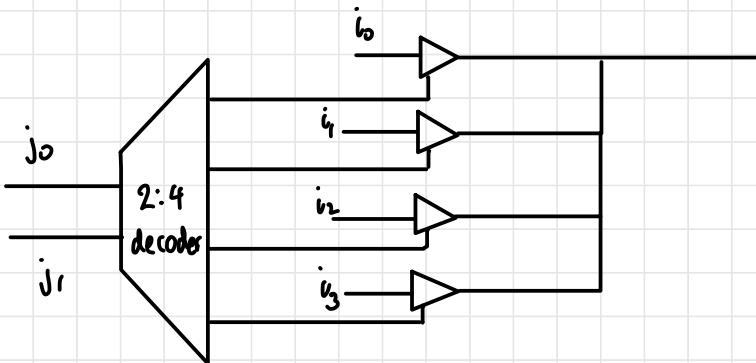
2:1 MUX Using Tristate Buffers

- 2 tristate buffers, 1 inverter



4:1 MUX

2:4 decoder, 4 tristate buffers



Applications of Decoders

eg: full subtractor $f(x,y,z)$

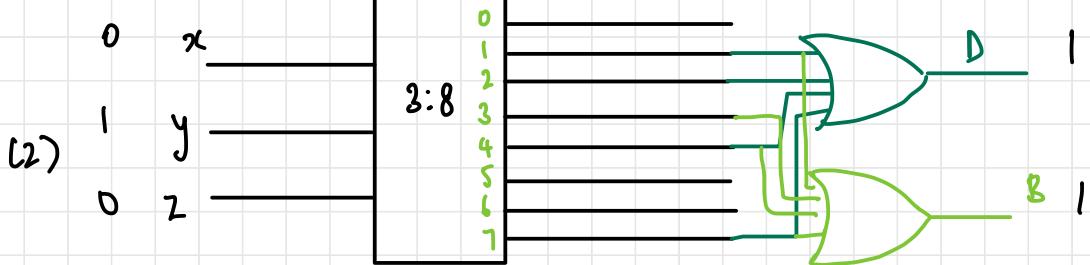
$$D = \Sigma(1, 2, 4, 7)$$

$$B = \Sigma(1, 2, 3, 7)$$

$$0 - 1 - 0 = 1 \ 1$$

decoder
outputs are
minterms

connect
minterms



full subtractor

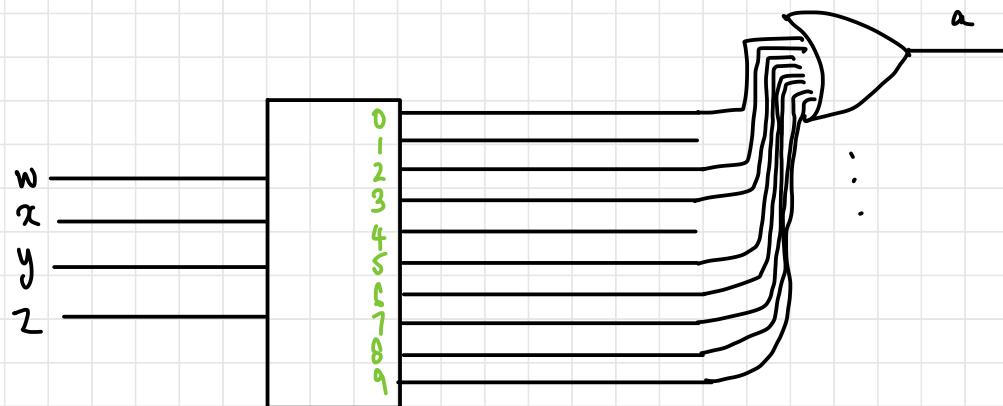
a	b	c-bin	d	b_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Question 1

construct 7-segment display using decoder

$\begin{matrix} a \\ f & g & b \\ e & c \\ \hline d \end{matrix}$

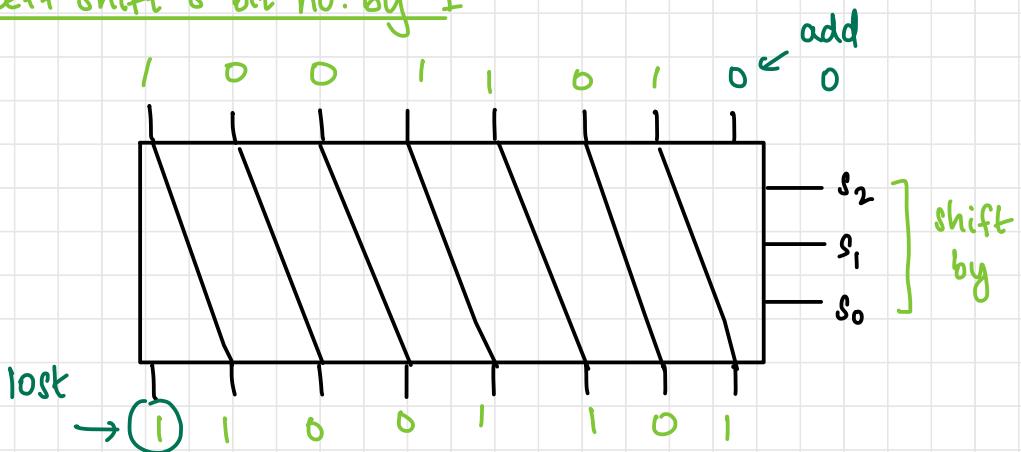
a	b	c	d	a	b	c	d	e	f	g	
0	0	0	0	1	-	-	1	1	1	0	0
0	0	0	-1	0	1	-	0	0	0	0	1
0	0	-1	0	-	-	0	-	-	0	1	2
0	0	-1	-1	-	-	-	-	-	0	0	3
0	1	0	0	0	-	-	0	0	1	-	4
0	1	0	-1	-	0	-	-	0	-	-	5
0	1	-1	0	-	0	-	-	-	-	-	6
0	-1	-1	-1	-	-	-	0	0	0	0	7
-1	0	0	0	-	-	-	-	1	1	-	8
-1	0	0	-1	-	-	-	0	1	1	-	9



Shifters

Left Shift

left shift 8-bit no. by 1



- max number of positions to left-shift = 7
- left shifting by 8 or greater: 0 (meaningless)

Bit	0	1
s_2	shift by 0	shift by 4
s_1	shift by 0	shift by 2
s_0	shift by 0	shift by 1

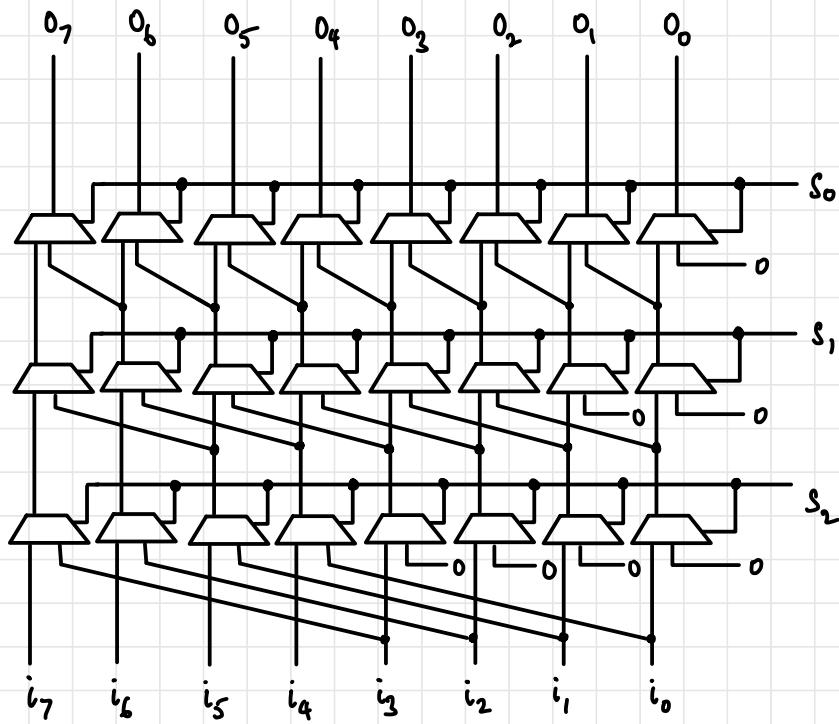
total will
be cumulative

$$\text{eg: } s_2 s_1 s_0 = 111 \Rightarrow \text{total shift is } 4+2+1 = 7 \text{ shifts}$$

$$s_2 s_1 s_0 = 101 \Rightarrow \text{total shift is } 4+0+1 = 5 \text{ shifts}$$

All three bits are independent of the other shifts

Barrel Shifter Structure



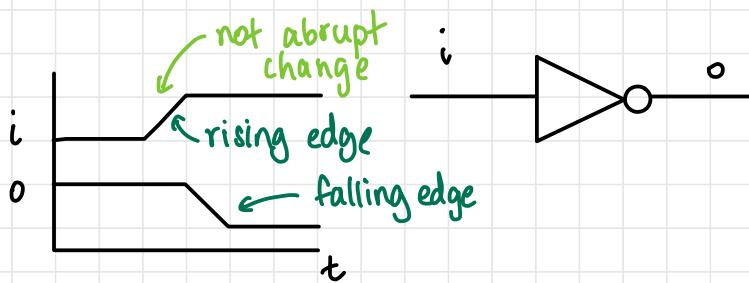
DELAYS

Propagation Delay

Time taken for change in I/P to reflect in O/P.

1) Gate

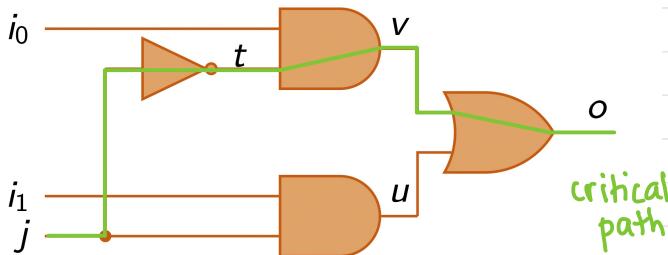
Timing diagram



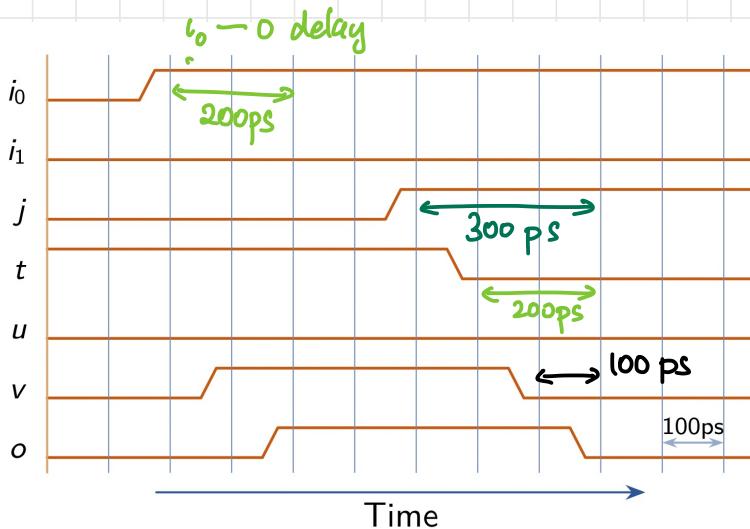
2) Logic circuit

- critical path: path whose delay takes the longest
 - propagation delay
- assume $t_{pd} = 100 \text{ ps}$
- delay due to change in any input \rightarrow depends on no. of gates

Example: MUX Circuit



- change in i_0 : 2 gates = 200ps
 - change in i_1 : 2 gates = 200ps
 - change in j : 3 gates = 300ps
- ← critical path delay

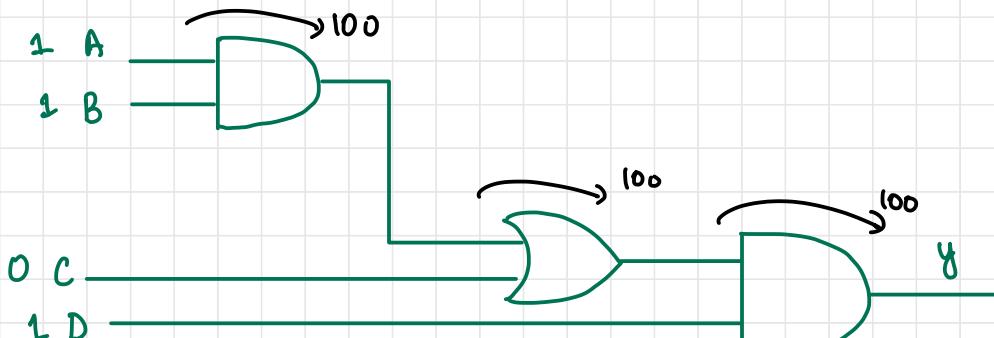


Critical path delay of MUX circuit = 300ps

How many 1/P changes per second?
 $1 \div (300 \times 10^{-12}) = 3.33 \text{ GHz}$

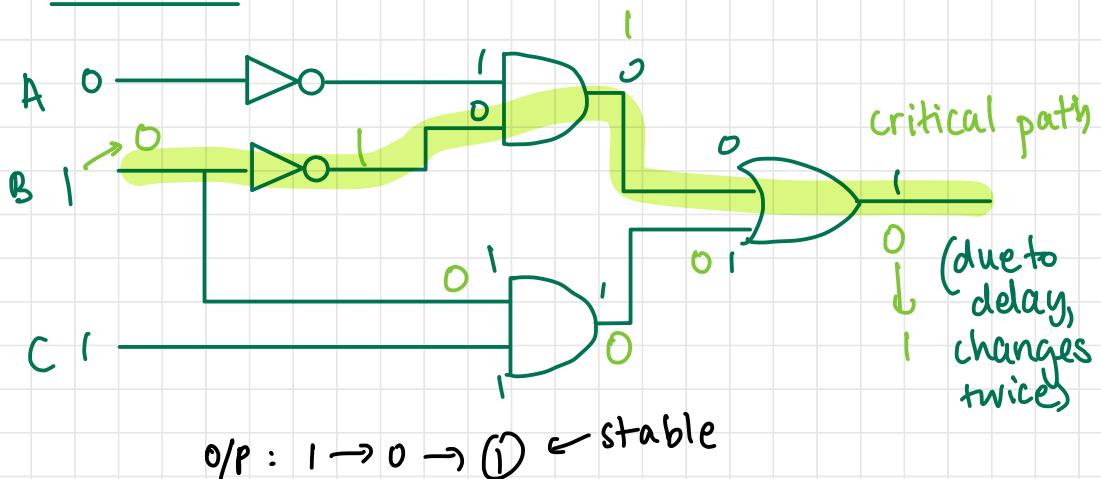
Question 2

What $A : 1 \rightarrow 0$, delay = 100ps



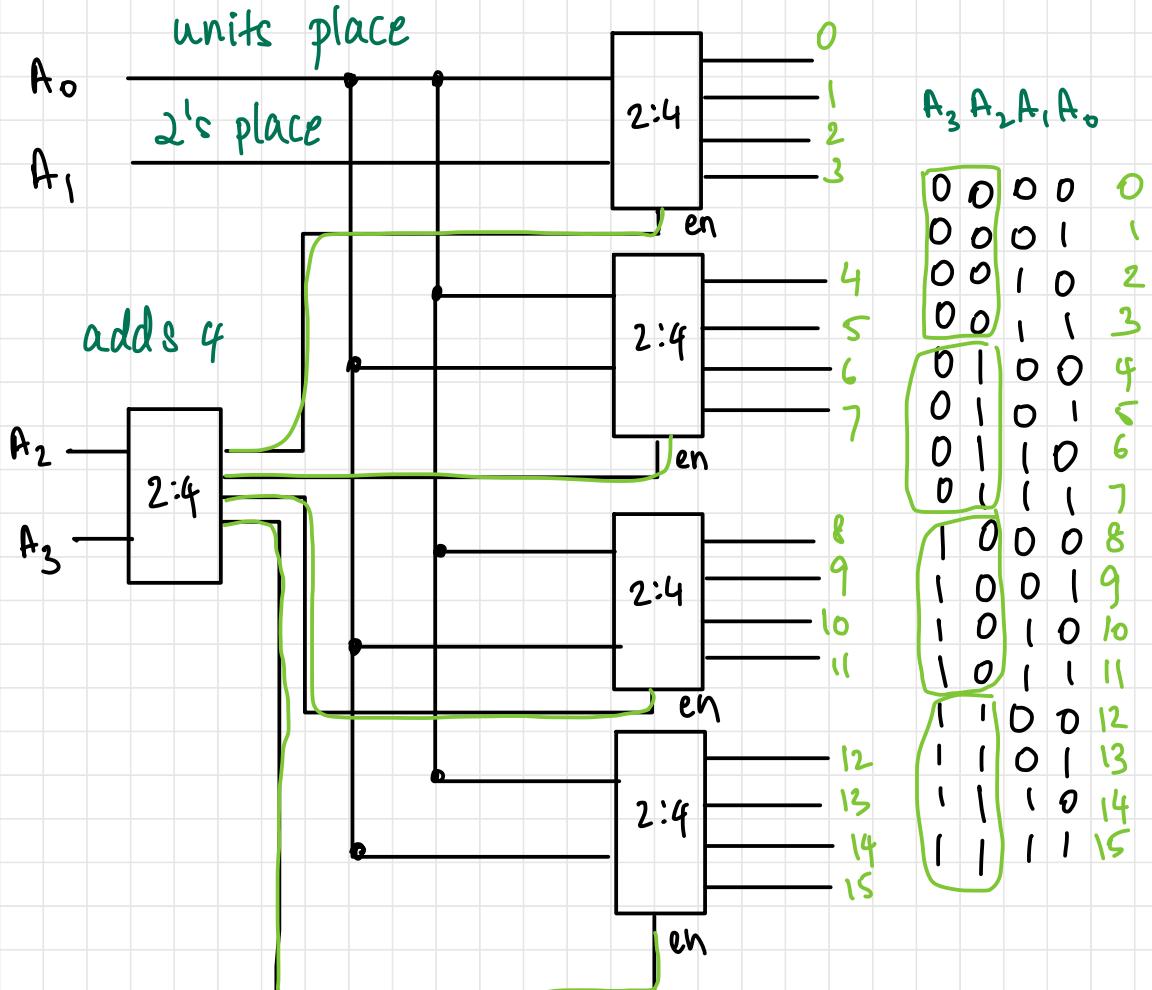
critical delay = 300ps

Question 3



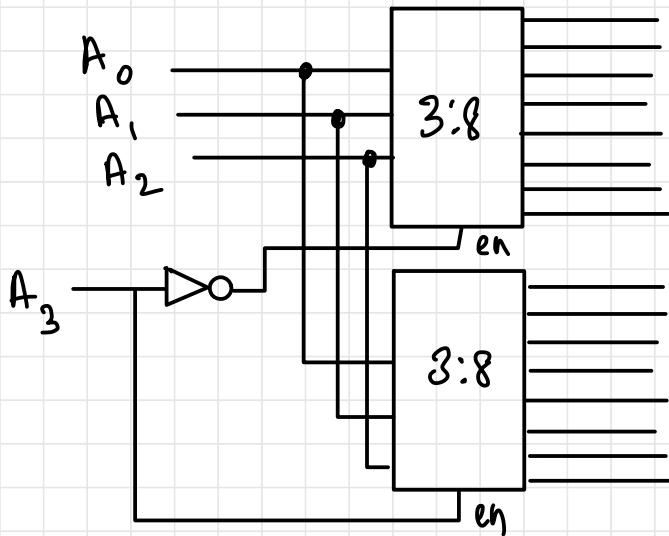
Question 4

Construct 4:16 decoder with five 2:4 line decoders with enable input



Question 5

Construct 4:16 decoder with two 3:8 decoders and a not gate

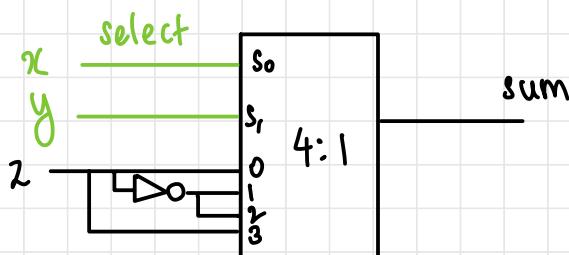


Question 6

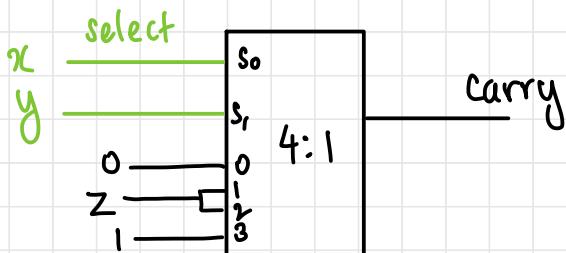
Full adder using MUX

$$s(x, y, z) = \sum(1, 2, 4, 7)$$

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



x	y	z	s	c
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

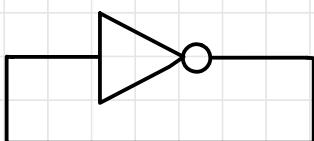


SEQUENTIAL LOGIC CIRCUITS

- logic circuits that have memory/storage elements (state of logic circuit)
- Output fed back to input

Simplest Memory Unit

single inverter



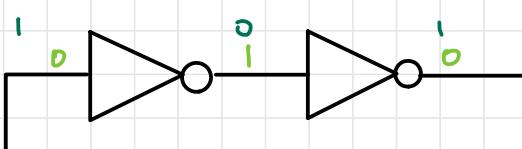
Assume $O/P = 0 \Rightarrow I/P = 1$
 $O/P = 1 \Rightarrow I/P = 0$

impossible!!

astable output

- does not work

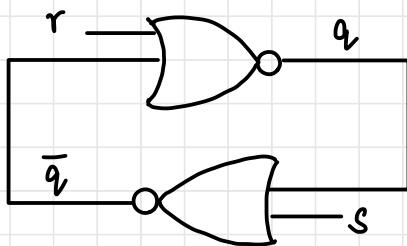
Two inverters



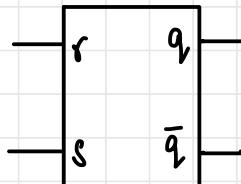
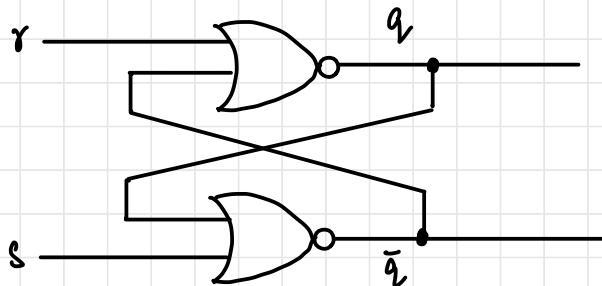
stable!!

SR Latch (Set-Reset Latch)

- Cross-coupled NOR gates



also drawn like



stable, but
to be
avoided
(forbidden)

Truth table

S	R	Q	\bar{Q}
0	0	q_{prev}	\bar{q}_{prev}
0	1	0	1
1	0	1	0
1	1	0	0

Memory
↙

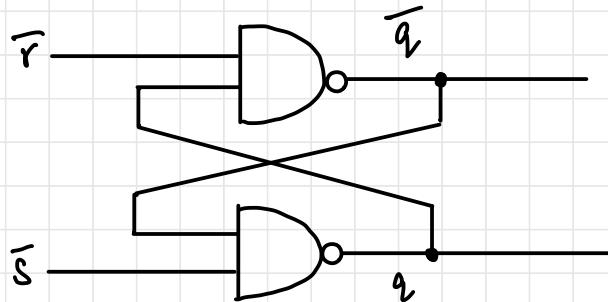
for $s=0, r=0$

case 1: $q = D$
 $\therefore q_{\text{prev}} = 1$
 $q = 0$

case 2: $q = \bar{D}$
 $\therefore q_{\text{prev}} = 0$
 $q = 1$

retains
previous
values

NAND-Based SR Latch

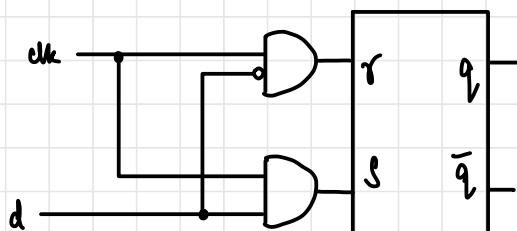


Truth table

\bar{S}	\bar{R}	q	\bar{q}
0	0	q_{prev}	\bar{q}_{prev}
0	1	1	0
1	0	0	1
1	1		

forbidden

D Latch (Data)

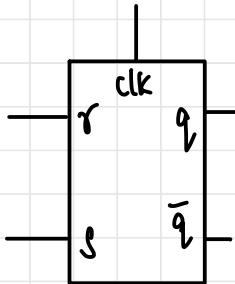


clk	d	s	r	q	\bar{q}
0	0	0	0	q_{prev}	\bar{q}_{prev}
0	1	0	0	q_{prev}	\bar{q}_{prev}
1	0	0	1	0	1
1	1	1	1	0	1

memory

transparent

- no indeterminate states
- level sensitive — changes when $\text{clk} = 1$

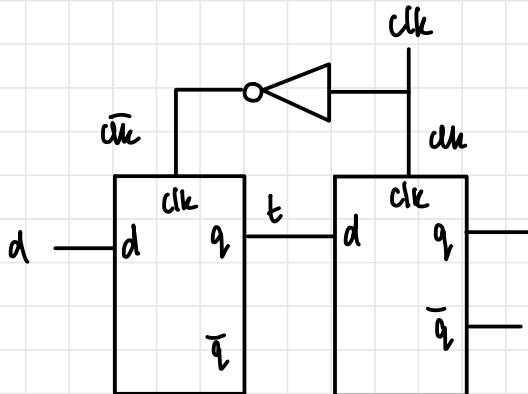


Level sensitive
 ↳ +ve level sensitive
 ↳ -ve level sensitive

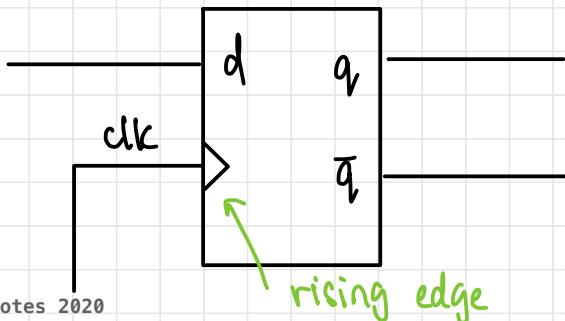
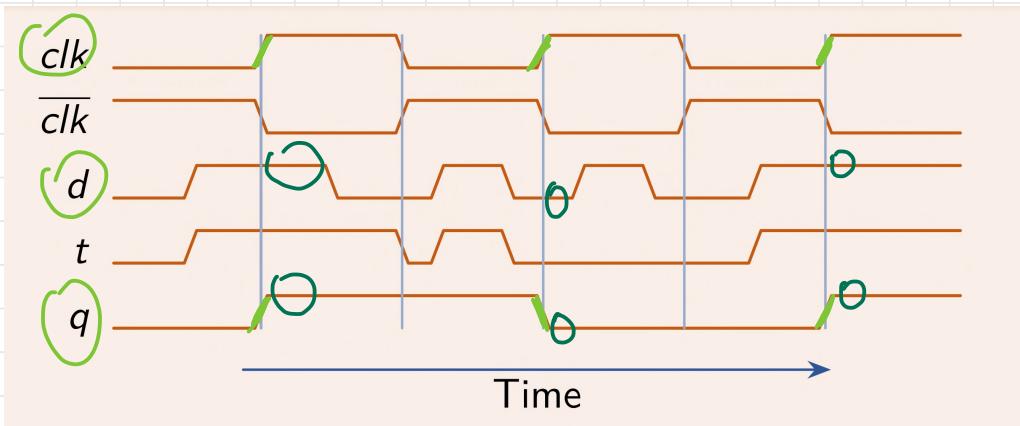
Edge sensitive
 ↳ rising edge trigger
 ↳ falling edge trigger

D - Flip Flop

- Edge triggered - flip flops

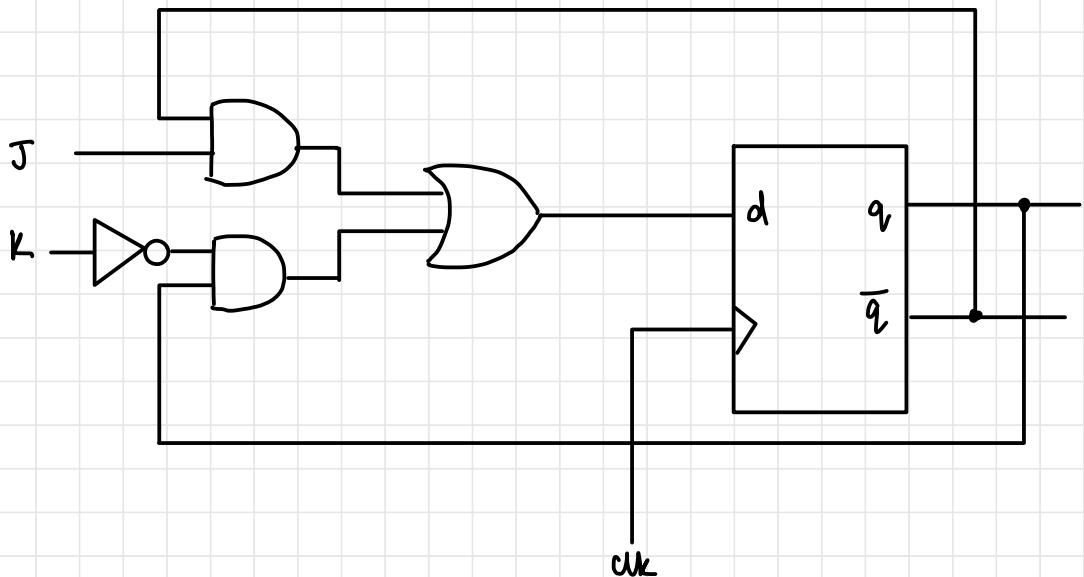


- When \bar{clk} is 1, it reflects d
- when \bar{clk} is 1, changes in d are ignored and previous t value is retained



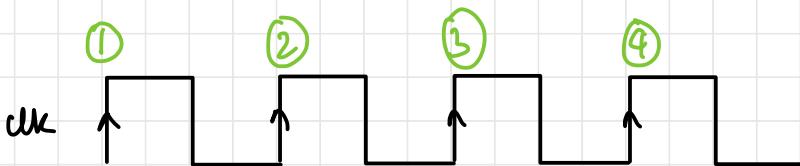
J-K Flip Flop

j	k	q	\bar{q}
0	0	q_{prev}	\bar{q}_{prev}
0	1	0	1
1	0	1	0
1	1	\bar{q}_{prev}	q_{prev}



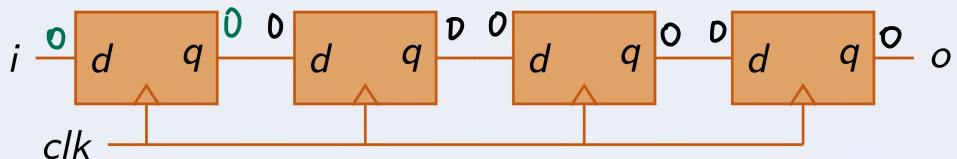
4-Bit Shift register

8180

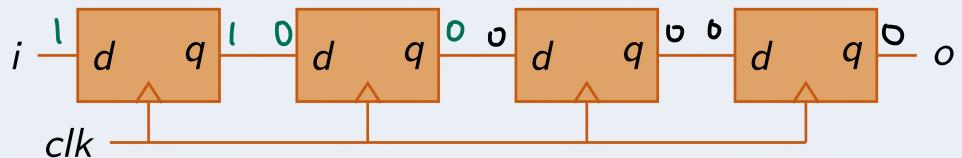


Input: 1010

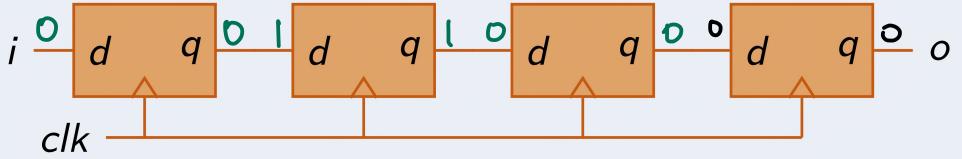
1



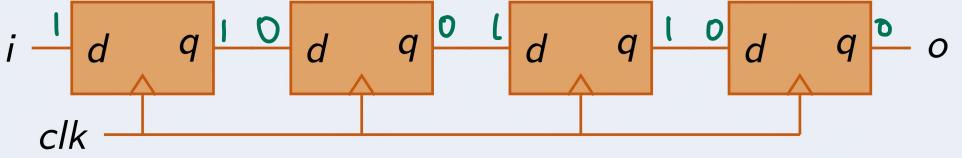
2



3

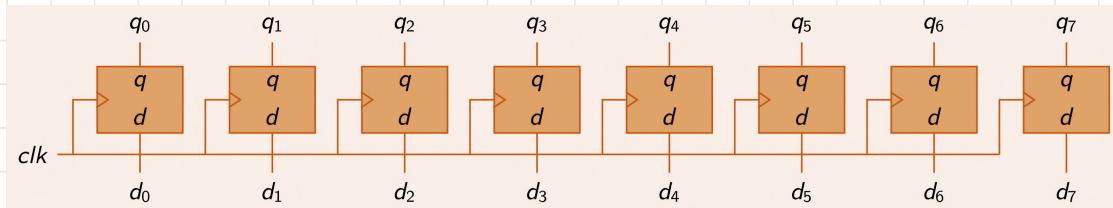


4

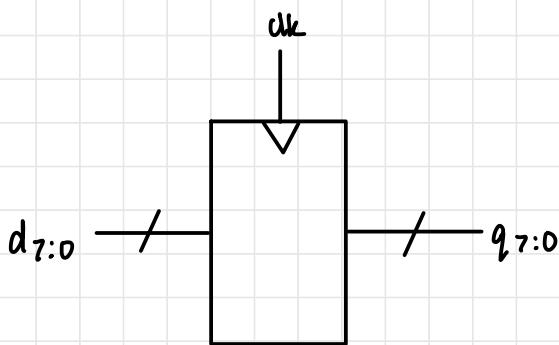


8-Bit Shift Register

PIPO (parallel in, parallel out)

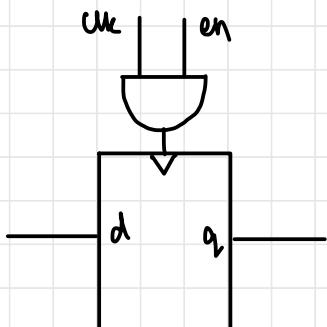


Symbol



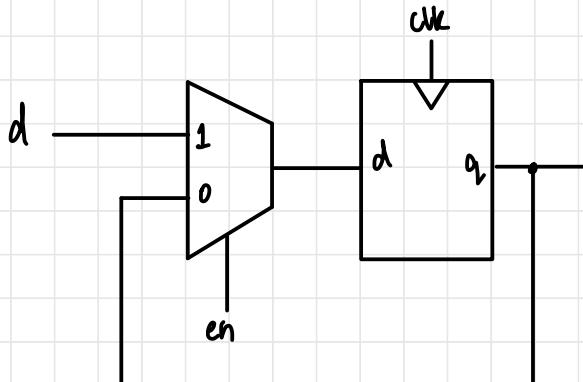
Flip Flop with Enable

I) Clock Gating Approach

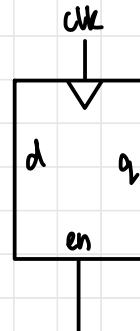


- clk signal is gated
- can cause timing problems
- careful use can reduce power consumption - useful here
- usually avoided

2) MUX



Symbol

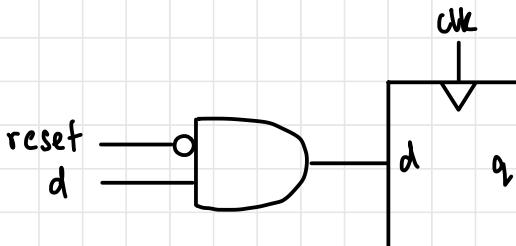


- At rising edge of clock

en	d	q
0	0	q_{prev}
0	1	q_{prev}
1	0	0
1	1	1

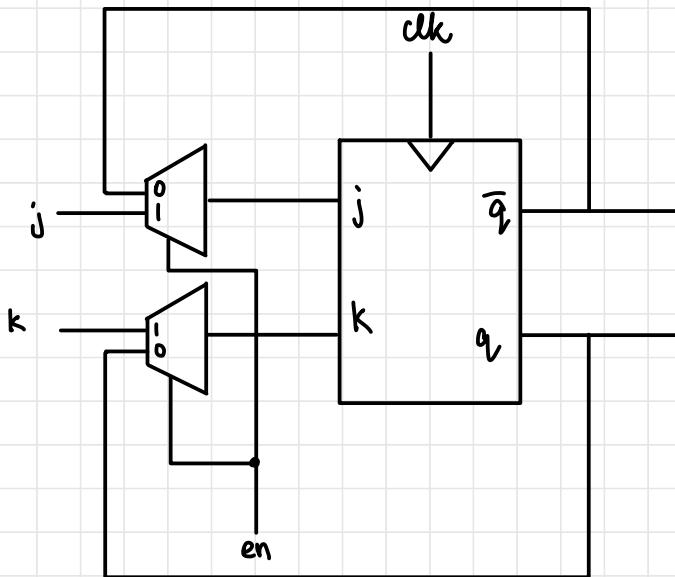
Resettable Flip Flop

- When $reset = 0$, d-flip flop normal
- When $reset = 1$, d-flip flop stores D
- Force memory elements into known state
- Initial power supply - prevent randomness



Construct JK Flip Flop Using enable / reset

Flip Flop with Enable



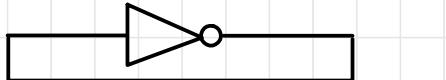
j-k flip-flop truth table

j	k	q	\bar{q}
0	0	q_{prev}	$\overline{q_{\text{prev}}}$
0	1	0	1
1	0	1	0
1	1	$\overline{q_{\text{prev}}}$	q_{prev}

Synchronous Sequential Logic Circuits

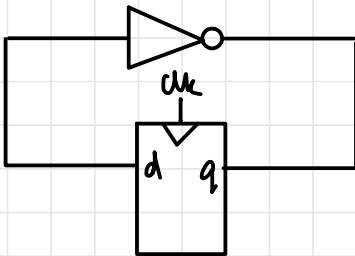
2) Loops

- inverter loop is a problem



unstable

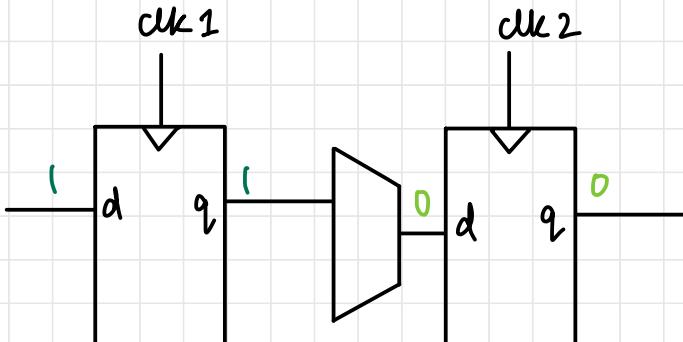
- solution: insert flip flop



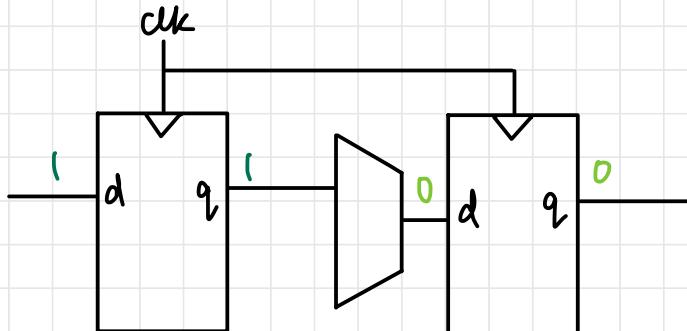
stable: toggles at every clock rising edge

i) Race Condition

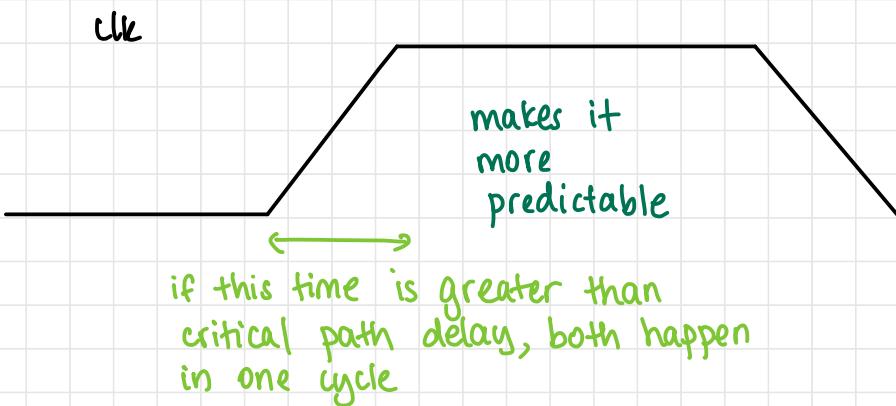
- Memory elements can suffer from race conditions (race hazards)
- depends on relative timing of clks
- if clk 1 arrives before clk 2, race condition



- solution: common clock synchronisation



- if clk time period is greater than critical path delay
race condition eliminated



Synchronous Sequential Logic Circuit

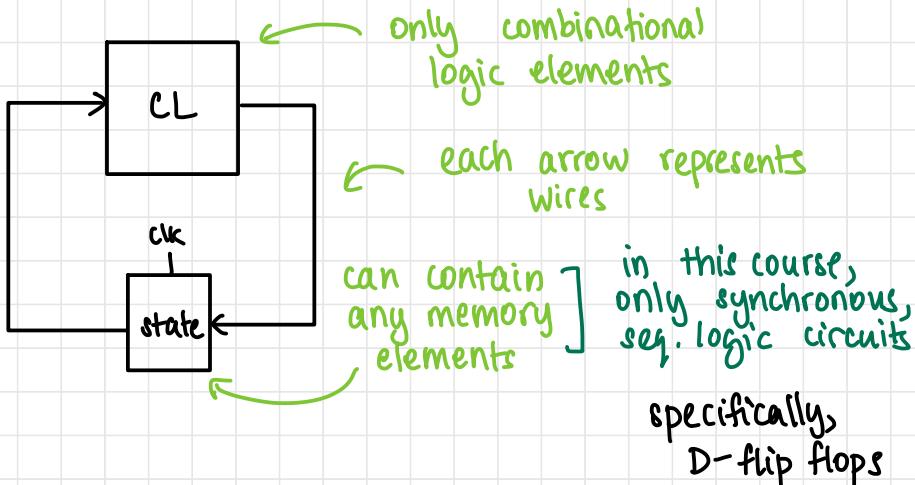
- 1) Every circuit element is either a register (flip flop) or a combinational circuit
 - ↳ no latches alone, only flip flops
 - ↳ think of flip flops as indivisible units
- 2) At least one circuit element is a register
- 3) All registers receive same clock signal
- 4) Every cyclic path has at least one register
 - ↳ no inverter loops

Classify the following logic circuits as combinational or synchronous sequential or neither

- ▶ AND gate (output not connected to own input) **combinational logic circuit**
- ▶ AND gate (output connected to own input) **neither**
- ▶ D flip-flop **synchronous sequential logic circuit**
- ▶ SR latch **neither**
- ▶ Series of D flip-flops (with common clock) with output of one connected to the input of the next via an inverter **synchronous sequential logic circuit**

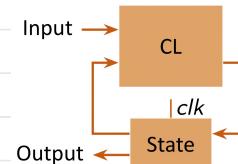
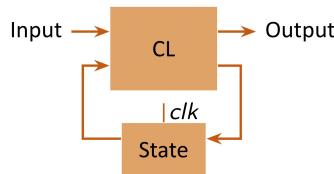
FINITE STATE MACHINES

- Mathematical foundation for sequential logic circuits
- Consists of two blocks - CL and state block

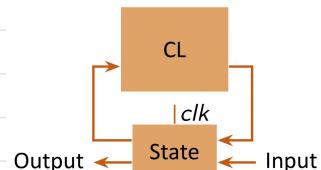
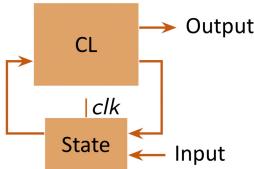


- Any sequential logic circuit (from a counter to a complex microprocessor) can be represented as an FSM
- Fundamental concept in CSE
- In APLL, nodes and edges, but to implement, this

- This diagram lacks input & output
- Two types - **Mealy** type and **Moore** type FSMs
- Four input/output possibilities

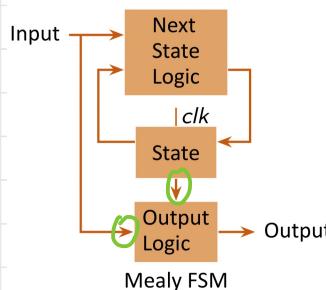


- if I/P given to state block, delayed by one clock cycle and therefore neglected



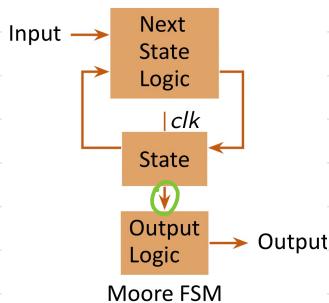
Mealy FSM

- O/P depends on current state as well as input



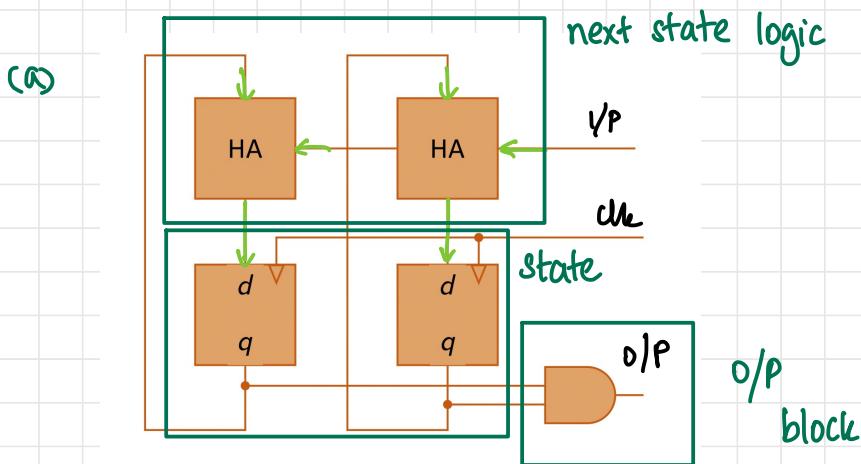
Moore FSM

- Output depends only on current state



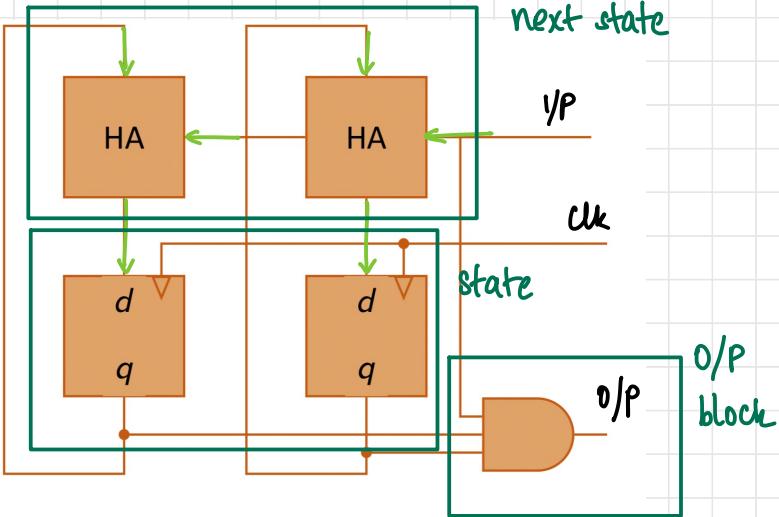
Question 7

Identify as Mealy or Moore & make into blocks



Moore FSM

(b)



Mealy FSM

Designing an FSM (Moore)

1. Determine input / output
2. State transition diagram
 - ↳ no. of states
 - ↳ draw diagram
3. Encoding tables
 - ↳ state d-flip flops
 - ↳ output
4. State transition table
5. Output table
6. Logic minimisation
 - ↳ Boolean formulas
7. Logic circuit construction

] ~truth
tables

Question 8

Lift / elevator control logic (2 floors)

Inputs

- Lift has a switch which can be in the down or up position, indicating that the lift should go down or up respectively.
- The switch signal switch-up is 0 when lift should go down and is 1 when the lift should go up.
- Lift system has a sensor which indicates when the lift is stationary or moving between floors.
- Signal on-floor is 1 when lift is at ground or first floor and is 0 when in between.

Outputs

- Signal lift-up when lift should move up
- Signal lift-down when lift should move down
- Signal on-ground is 1 when on ground floor and 0 when on first floor
- Signal on-first is 1 when on first floor and 0 when on ground floor

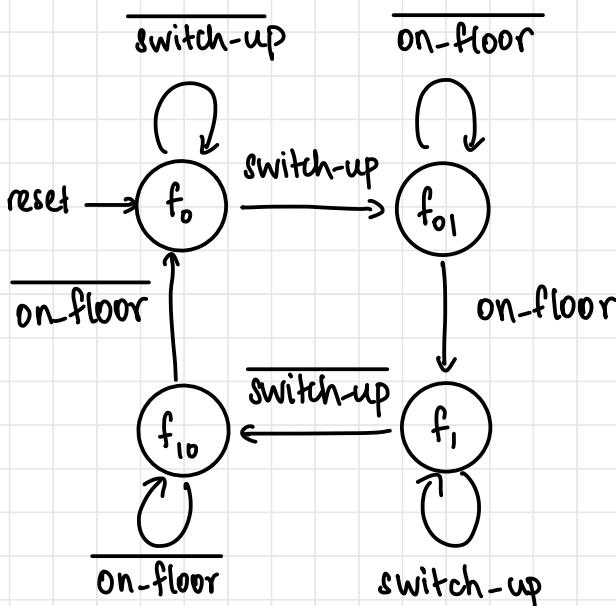
Reset / start state

- Elevator initially on ground floor.



State Transition Diagram

- Visual FSM
- states / nodes / vertices → circles
- Transition between states → line segments / arrows / arcs / edges
- Each contains Boolean formula of inputs as label
- Determine no. of states (2 in transition, 2 stationary)



State Encoding Table

State	Encoding ($s_1 s_0$)	State	Encoding ($s_1 s_0$)
f0	00	f0	00
f01	01	f01	01
f1	11	f1	11
f10	10	f10	10

Output Encoding Tables

- on-ground

Meaning	Encoding
Lift on ground floor	1
Lift anywhere else	0

- on-first

Meaning	Encoding
Lift on first floor	1
Lift anywhere else	0

- lift-up

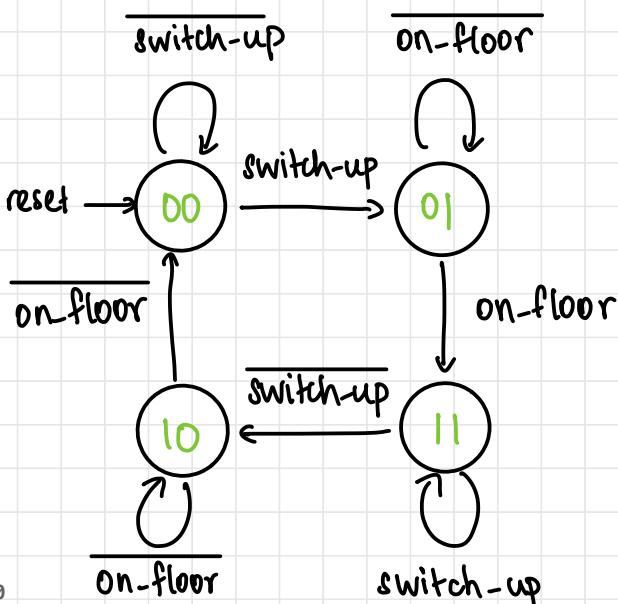
Meaning	Encoding
Lift going from ground to first floor	1
Lift anywhere else	0

- lift-down

Meaning	Encoding
Lift going from first to ground floor	1
Lift anywhere else	0

State Transition Table

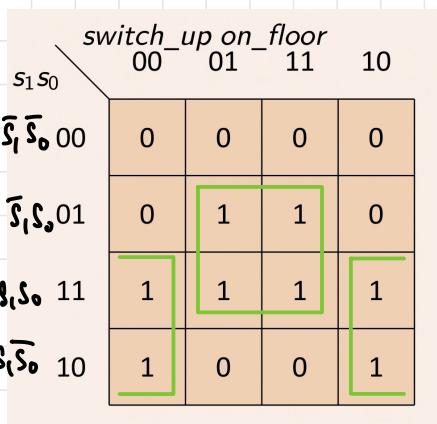
Current State		Inputs		Next State	
s_1	s_0	switch_up	on_floor	s'_1	s'_0
f_0	0	0	(0) no change	0	0
	0	0	(0)	0	0
	0	0	(1) goto f_{01}	0	1
	0	0	(1)	0	1
f_{01}	0	1	0	0	1
	0	1	0	1	1
	0	1	1	0	1
	0	1	1	1	1
f_{10}	1	0	0	0	no change
	1	0	0	1	0
	1	0	1	0	0
	1	0	1	1	0
f_1	1	1	(0) go to f_{10}	0	0
	1	1	(0)	1	0
	1	1	(1) no change	1	0
	1	1	(1)	1	1



Output Table

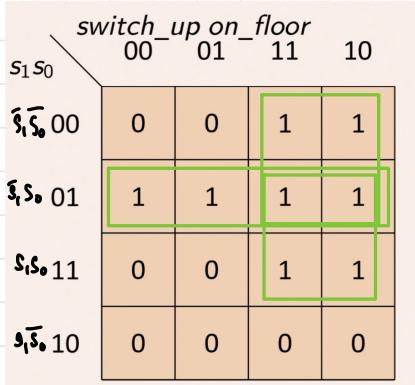
State		Outputs			
s_1	s_0	on_ground	on_first	$lift_up$	$lift_down$
on floor	0 0	1	0	0	0
in trans.	0 1	0	0	1	0
	1 0	0	0	0	1
	1 1	0	1	0	0

K-map for s'_1



$$s'_1 = \overline{on_floor} \cdot s_1 + s_0 \cdot \overline{on_floor}$$

K-map for s'_0



$$s'_0 = \bar{s}_1 s_0 + \text{switch-up} \cdot \bar{s}_1 + \text{switch-up} \cdot s_0$$

Boolean Formulas

Next state

$$s_1' = \overline{\text{on_floor}} \cdot s_1 + s_0 \cdot \overline{\text{on_floor}}$$

$$s_0' = \overline{s_1} s_0 + \text{switch-up} \cdot \overline{s_1} + \text{switch-up} \cdot s_0$$

Outputs

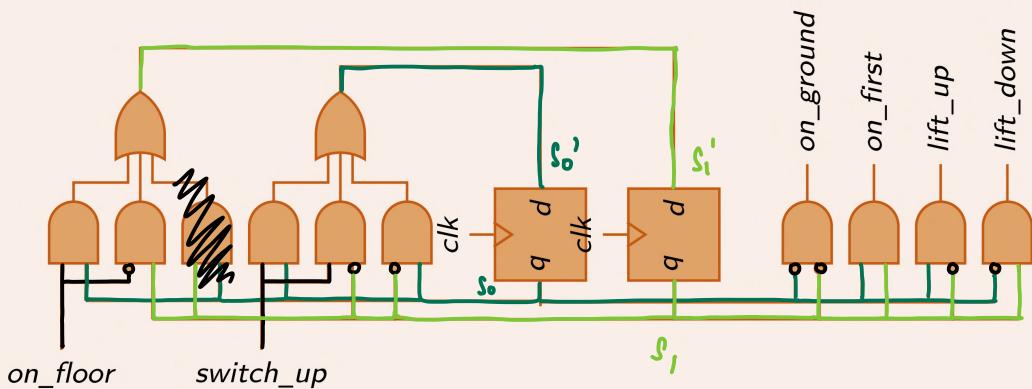
$$\text{on-ground} = \overline{s_1} \overline{s_0}$$

$$\text{on-first} = s_1 s_0$$

$$\text{lift-up} = \overline{s_1} s_0$$

$$\text{lift-down} = s_1 \overline{s_0}$$

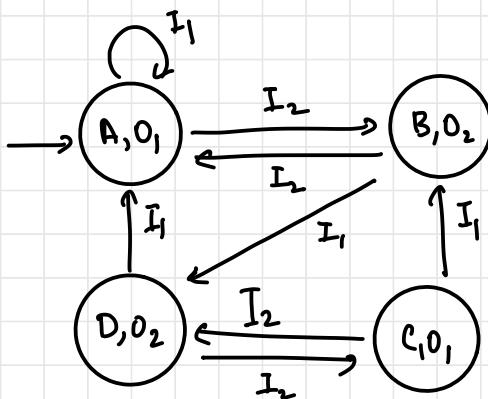
Logic Circuit



Moore FSM

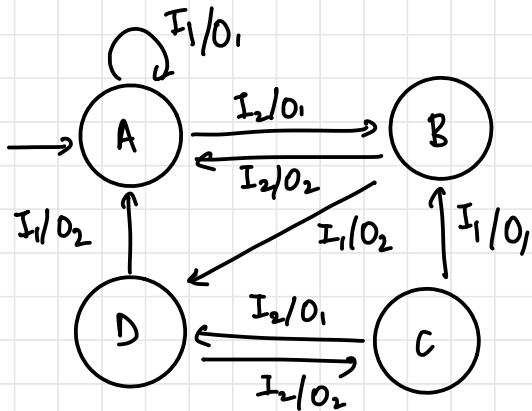
State table for Moore Machine

Present state	Next state		Output
	Input I_1	Input I_2	
A	A	B	O_1
B	D	A	O_2
C	B	D	O_1
D	A	C	O_2



Mealy Machine

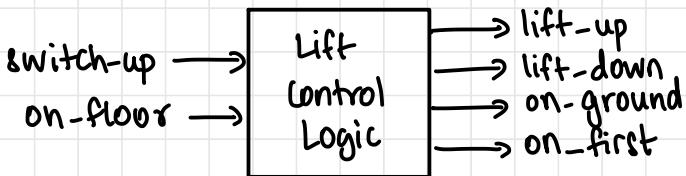
Present state	Next state, output	
	Input I_1	Input I_2
A	A, O1	B, O1
B	D, O2	A, O2
C	B, O1	D, O1
D	A, O2	C, O2



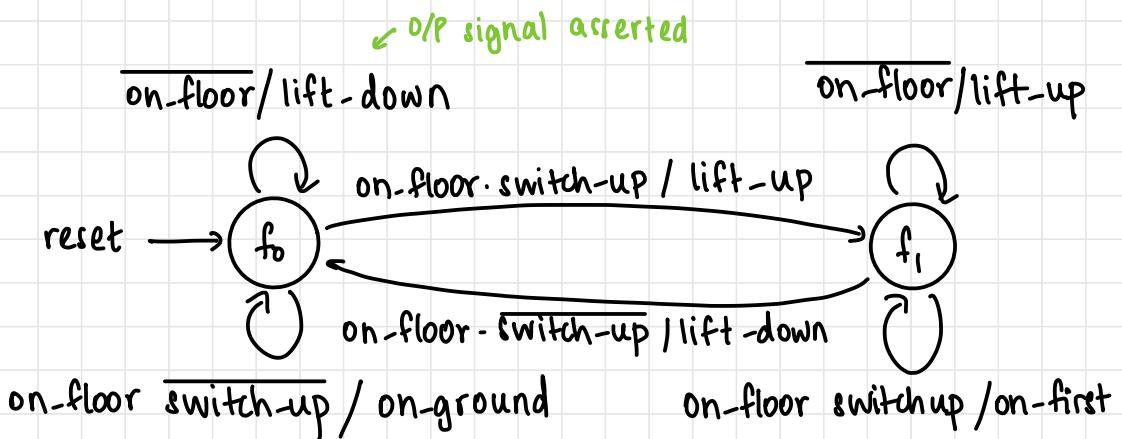
Question 9

Elevator Problem as a Mealy Machine

- depends on YP



- Determine no. of states — here only 2 states
- state f_0 : lift on ground floor or in transit to ground floor
- state f_1 : lift on first floor or in transit to first floor



State Encoding Table

State	Encoding (s_1, s_0)
f_0	0
f_1	1

Output Encoding Tables

- on-ground

Meaning	Encoding
Lift on ground floor	1
Lift anywhere else	0

- on-first

Meaning	Encoding
Lift on first floor	1
Lift anywhere else	0

- lift-up

Meaning	Encoding
Lift going from ground to first floor	1
Lift anywhere else	0

- lift-down

Meaning	Encoding
Lift going from first to ground floor	1
Lift anywhere else	0

State Transition & Output Table — Mealy

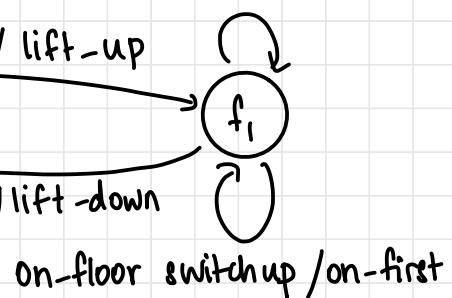
Current State	Inputs		Next State	Outputs			
	on_floor	switch_up		on_ground	on_first	lift_up	lift_down
fo (0)	not (0)	doesn't (0)	stays (0)	(0) not yet	(0) no	(0) no	(1) yes
fo (0)	on gnd (1)	matter (1)	stay (0)	(0) stays	(1) on gnd	(0) no	(0) no
fo (0)	on gnd (1)		go up (1)	(1) move	(0) no	(0) no	(1) yes
f ₁ (1)	not (0)	doesn't (0)	stays (1)	(0) no	(0) not yet	(1) yes	(0) no
f ₁ (1)	on first (0)	matter (1)	go down (0)	(0) move	(0) no	(0) no	(1) yes
f ₁ (1)	on first (1)		stay (1)	(1) stays	(0) no	(1) yes	(0) no

on-floor / lift-down



on-floor switch-up / on-ground

on-floor / lift-up



K-maps

K-map for s'

		on_floor	switch_up	
		00	01	11
		0	1	0
s	0	0	0	1
	1	1	1	0

$$s' = \overline{\text{on_floor}} \cdot s + \text{on_floor} \cdot \text{switch_up}$$

$$\text{on_ground} = \overline{s} \cdot \overline{\text{on_floor}} \cdot \overline{\text{switch_up}}$$

$$\text{on_first} = s \cdot \text{on_floor} \cdot \text{switch_up}$$

K-map for lift-up

		on_floor	switch_up	
		00	01	11
		0	1	0
s	0	0	0	1
	1	1	1	0

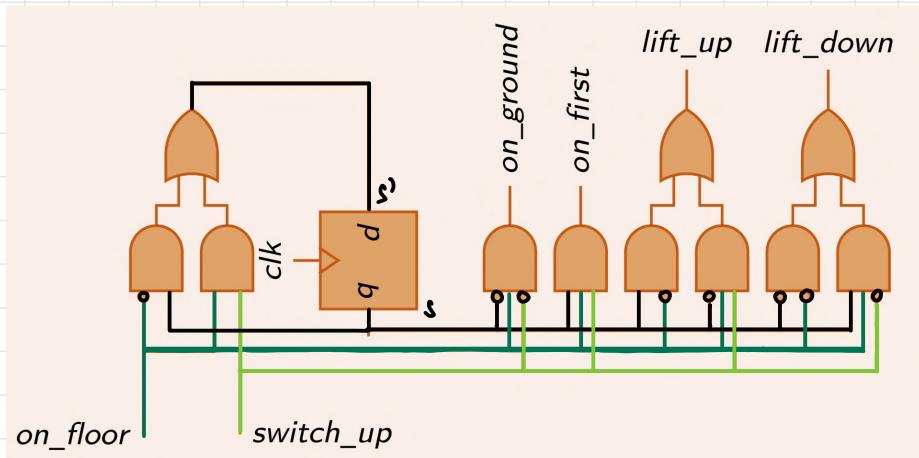
$$\text{lift-up} = s \cdot \overline{\text{on_floor}} + \overline{s} \cdot \text{on_floor} \cdot \text{switch_up}$$

K-map for lift-down

		on_floor switch_up				
		00	01	11	10	
s		0	1	1	0	0
1		0	0	0	0	1

$$\text{lift-down} = \overline{\text{on-floor}} \cdot \overline{s} + \text{on-floor} \cdot \overline{\text{switch-up}} \cdot s$$

Logic Circuit



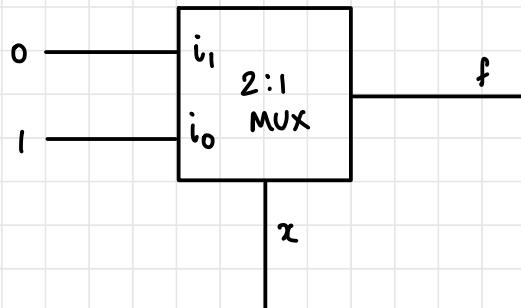
Mealy FSM

Revision Questions

Q1. Implement NOT, AND and OR using 2:1 MUX

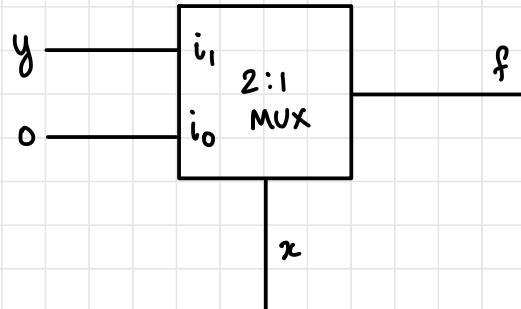
NOT

x	f
0	1
1	0



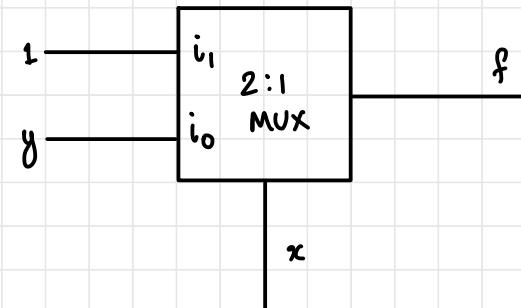
AND

x	y	f
0	0	0
0	1	0
1	0	0
1	1	1



OR

x	y	f
0	0	0
0	1	1
1	0	1
1	1	1

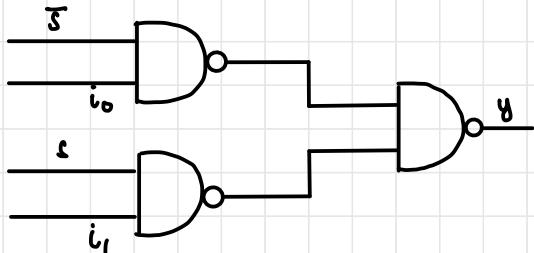


Q2. Construct 2:1 MUX using only 2-input NAND gates. How many gates are required?

selection	Output
0	i_0
1	i_1

$$y = \overline{s}i_0 + si_1$$

$$= (\overline{\overline{s}i_0} \cdot \overline{si_1})$$



Q3. Implement a decoder for the equation $f(x,y,z) = \sum(1,2,4,7)$

