DDCO

UNIT-2

CLASS NOTES

feedback/corrections: vibha@pesu.pes.edu

Vibha Masti

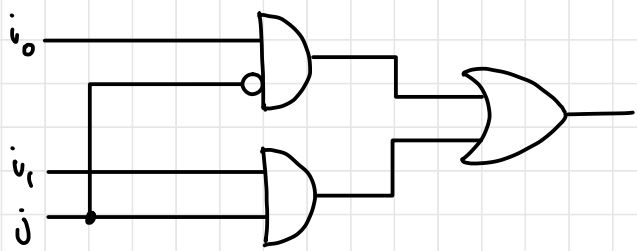
MUXERS, DECODERS & SHIFTERS

MULTIPLEXER — MUX

- multiplexes many inputs into a single output

2:1 MUX

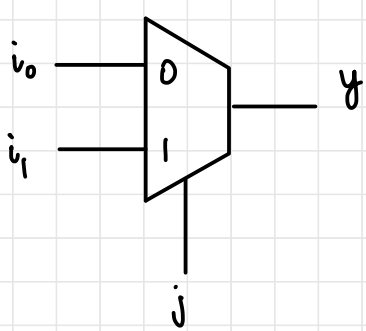
i_0	i_1	j	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



$$y = \bar{j} i_0 + j i_1$$

- 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

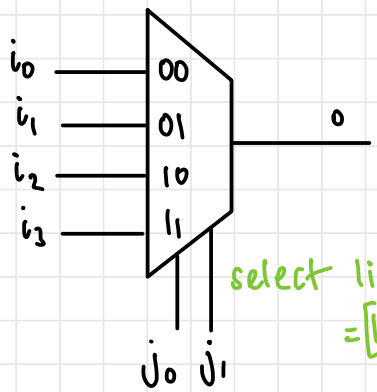
Symbol



4:1 MUX

- Data inputs: i_0, i_1, i_2, i_3
- Control inputs: j_0, j_1

2^n, n = control
2^n = data

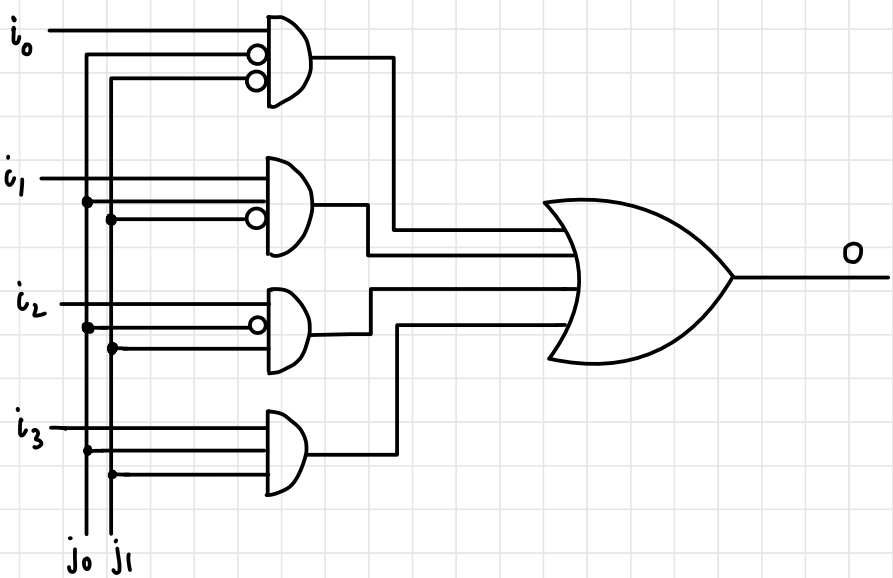


no truth table

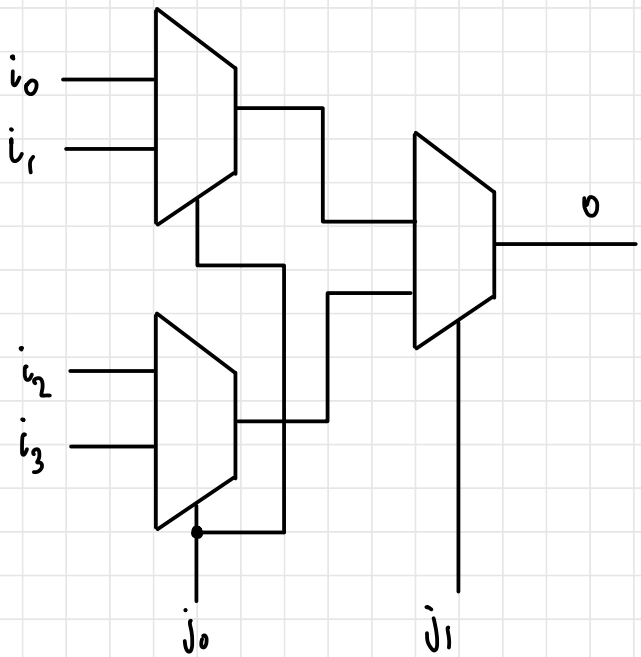
(2^6 = 64 input rows)

select lines = $\lceil \log_2(\text{data}) \rceil$ ceil

$$O = \bar{j_1}\bar{j_0}\, i_0 + \bar{j_1} j_0 i_1 + j_1 \bar{j_0}\, i_2 + j_1 j_0\, i_3$$



4:1 MUX Using 2:1 MUX



if $j_1 = 0$, i_0 or i_1

$j_1 = 1$, i_2 or i_3

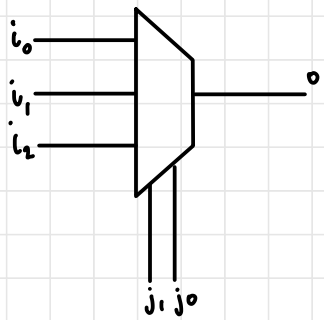
n:1 MUX

ceiling

- for n data inputs, $\lceil \log_2(n) \rceil$ control inputs and one output
- data I/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

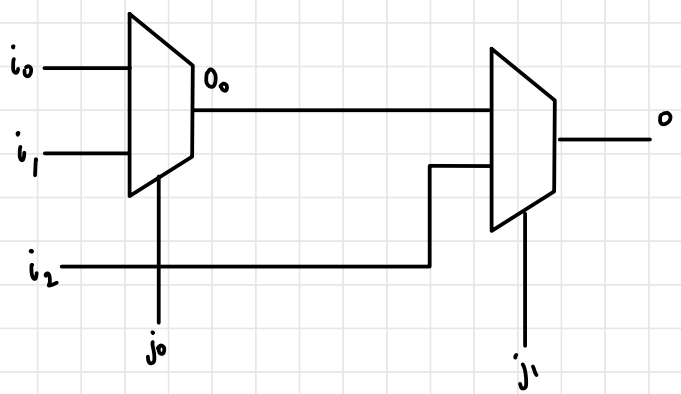
Construct 3:1 MUX using

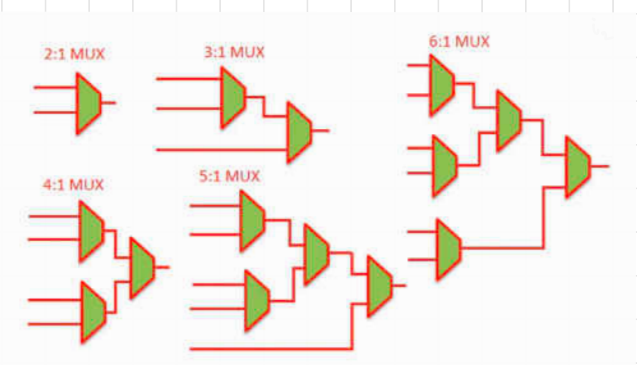
↳ 2:1 MUX
↳ AND, OR and NOT gates



$$O = \bar{j_0}\,\bar{j_1}\,i_0 + j_0\,\bar{j_1}\,i_1 + j_1$$

2:1 MUXes





DECODERS

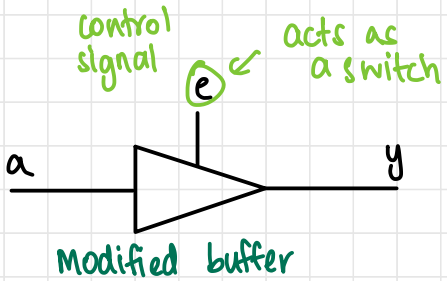
Floating Value (Z) / Tristate Buffer

- 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

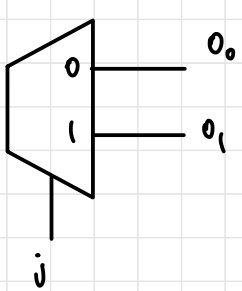
e	a	y
0	0	Z
0	1	Z
1	0	0
1	1	1

high impedance state



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

1:2 Decoder

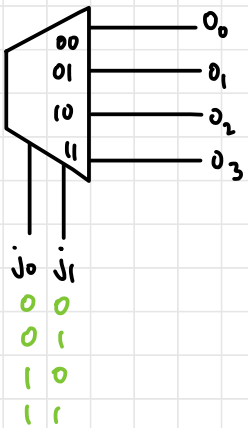


inputs: j
outputs: O_0, O_1

$O_0 = \bar{j}$
$O_1 = j$

if $j = 0$, $O_0 = 1$, $O_1 = 0$
if $j = 1$, $O_0 = 0$, $O_1 = 1$

2:4 Decoder



$O_0 = \bar{j_1}\bar{j_0}$

$O_1 = \bar{j_1} j_0$

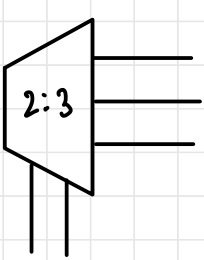
$O_2 = j_1 \bar{j_0}$

$O_3 = j_1 j_0$

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



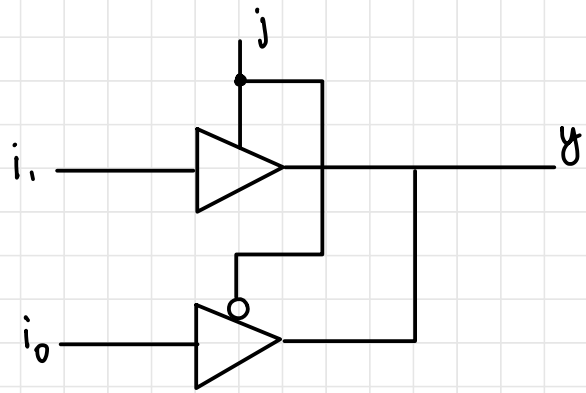
$\lceil \log_2 n \rceil : n$ decoder

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

ceiling

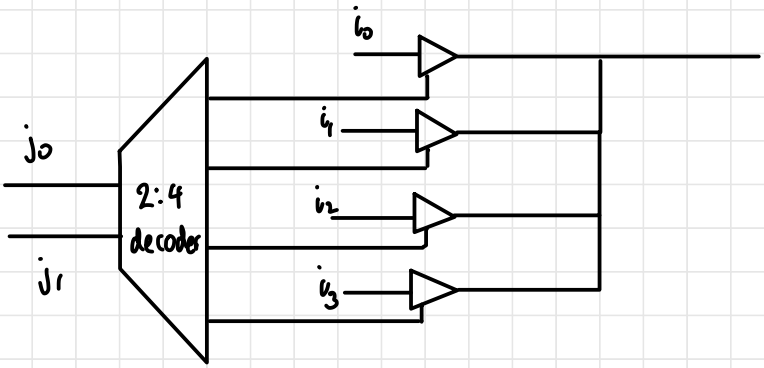
2:1 MUX Using Tristate Buffers

- 2 tristate buffers, 1 inverter



4:1 MUX

2:4 decoder, 4 tristate buffers

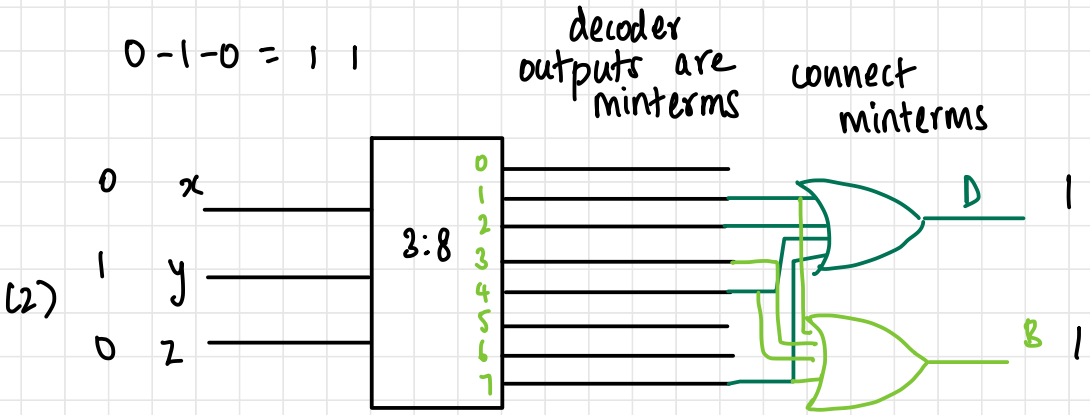


Applications of Decoders

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

$D = \Sigma(1,2,4,7)$
$B = \Sigma(1,2,3,7)$

0 - 1 - 0 = 1 1

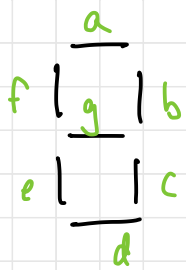


full subtractor

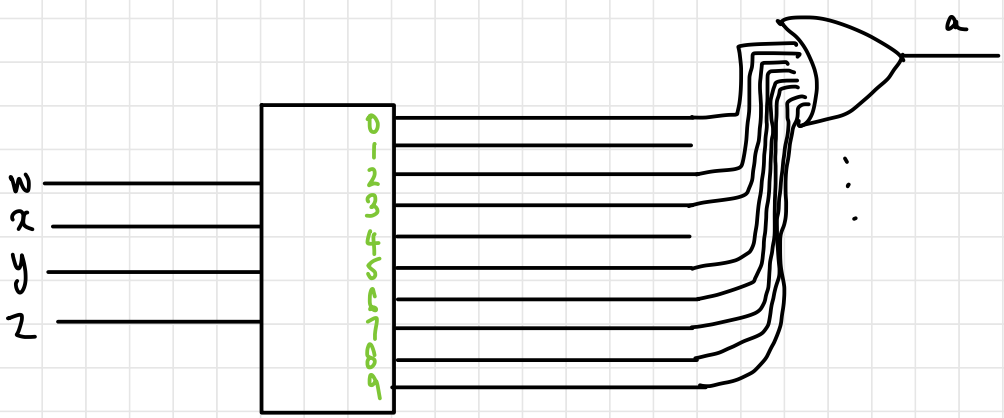
a	b	c=b_{in}	d	b_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Question 1

Construct 7-segment display using decoder



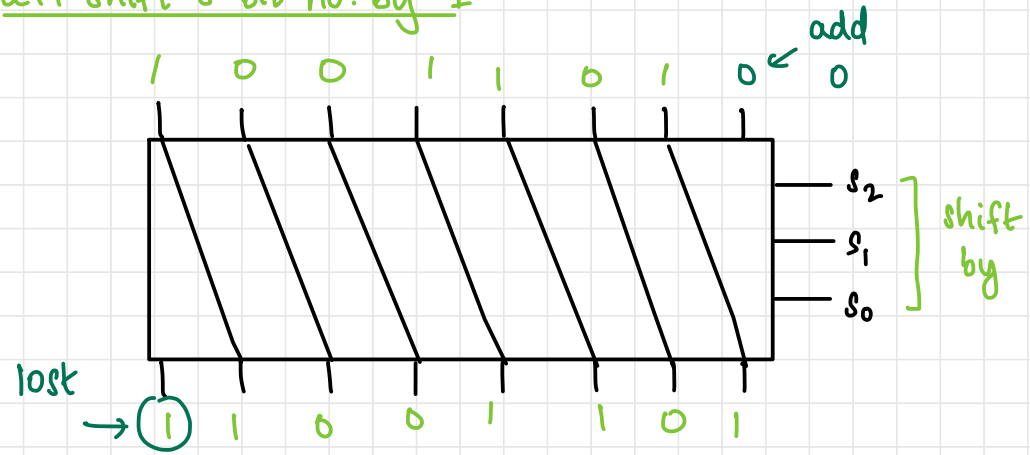
a	b	c	d	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9



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)

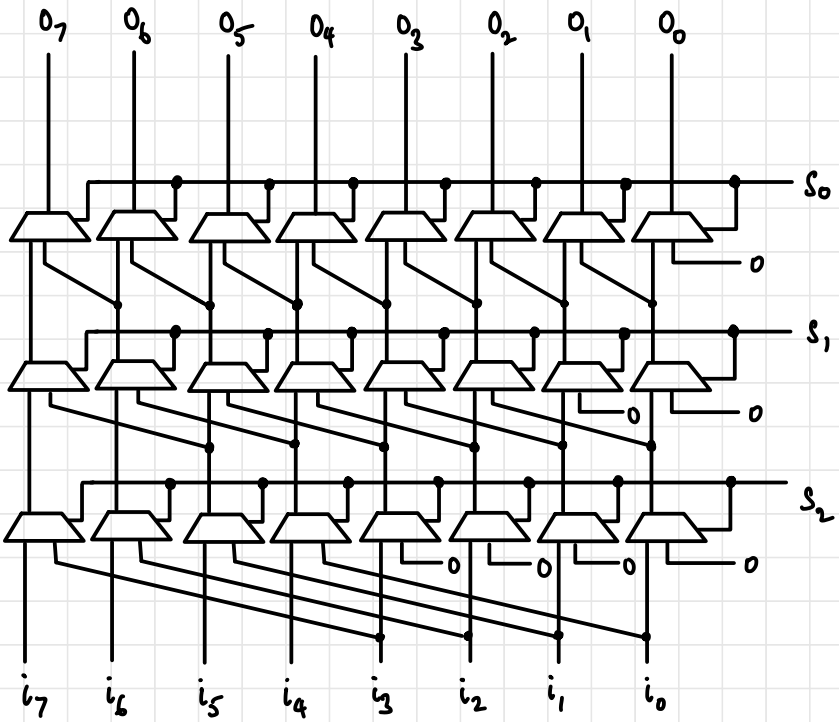
Bit	0	1
S_2	shift by 0	shift by 4
S_1	shift by 0	shift by 2
S_0	shift by 0	shift by 1

] total will be cumulative

eg: $S_2S_1S_0 = 111 \Rightarrow$ total shift is $4+2+1 = 7$ shifts
$S_2S_1S_0 = 101 \Rightarrow$ total shift is $4+0+1 = 5$ shifts

All three bits are independent of the other shifts

Barrel Shifter Structure



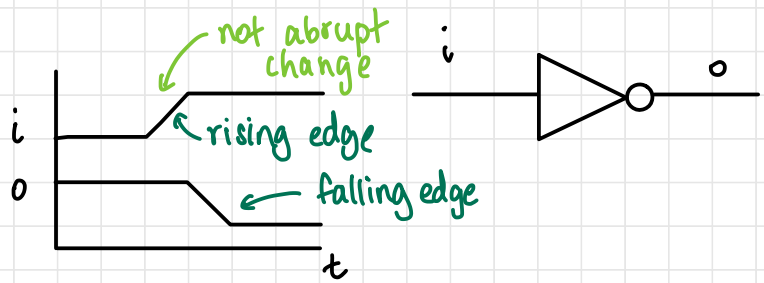
DELAYS

Propagation Delay

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

1) Gate

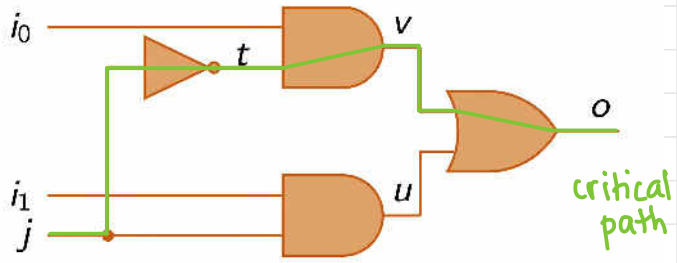
Timing diagram



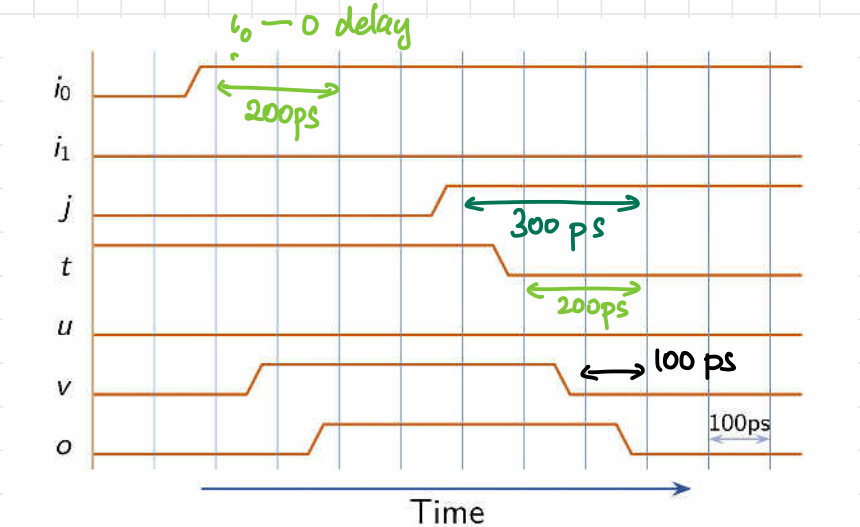
2) Logic circuit

- critical path: path whose delay takes the longest
- assume t_{pd} = 100 ps ← propagation delay
- delay due to change in any input → 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



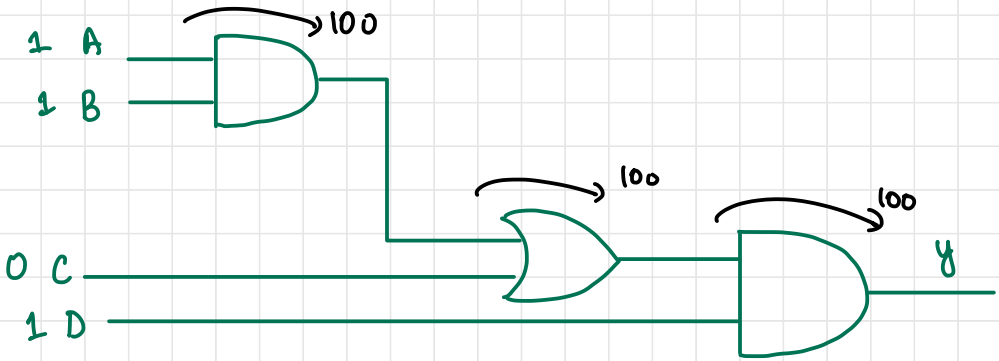
Critical path delay of MUX circuit = 300ps

How many I/P changes per second?

$1 \div (300 \times 10^{-12}) = 3.33$ GHz

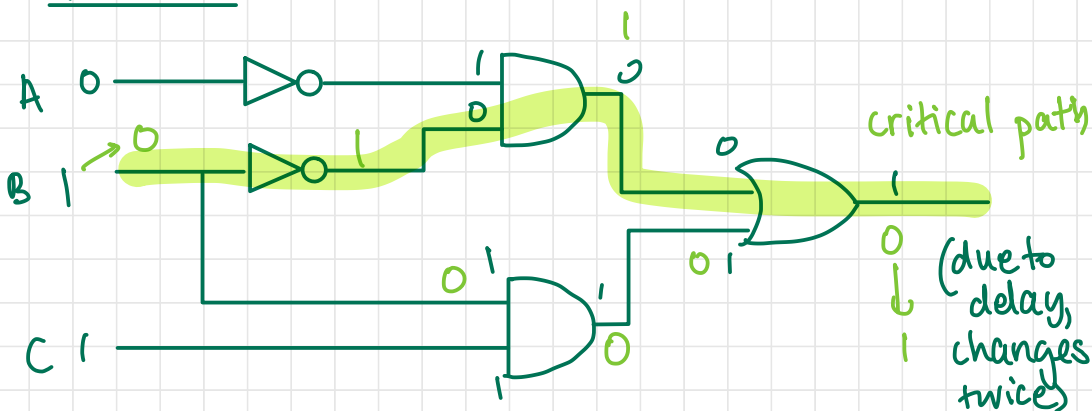
Question 2

What A : 1 → 0 , delay = 100 ps



critical delay = 300 ps

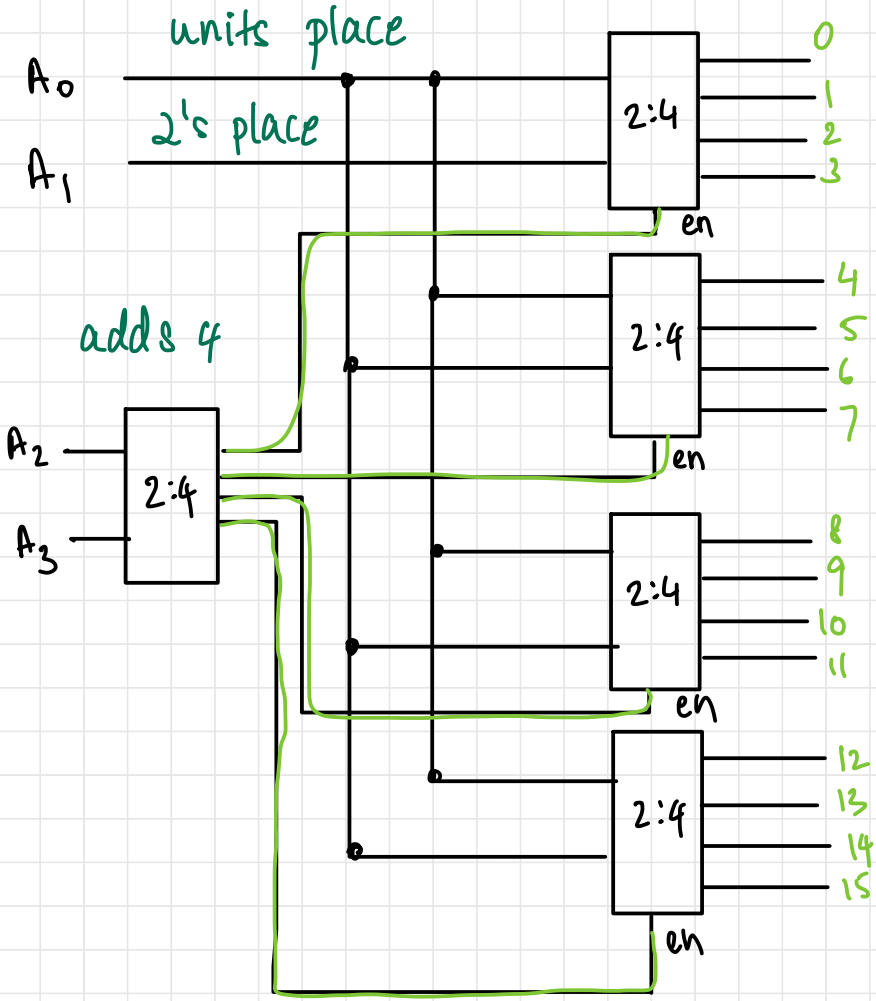
Question 3



O/P : 1 → 0 → (1) ← stable

Question 4

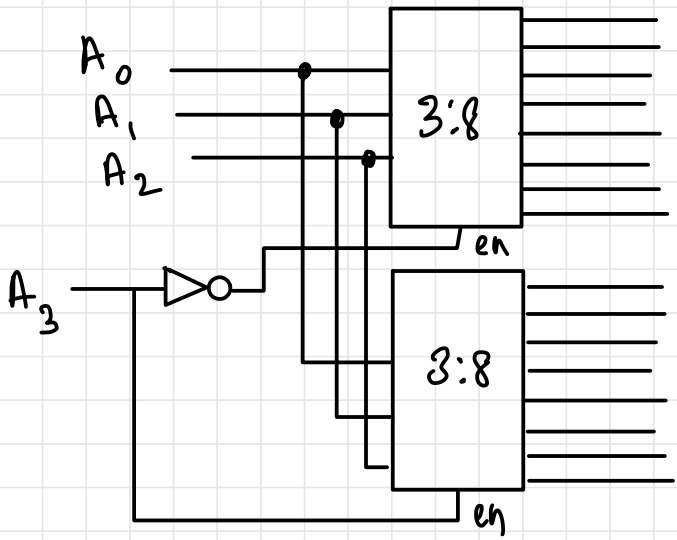
Construct 4:16 decoder with five 2:4 line decoders with enable input



A_3	A_2	A_1	A_0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Question 5

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



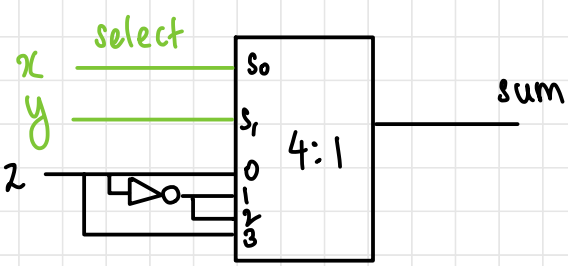
Question 6

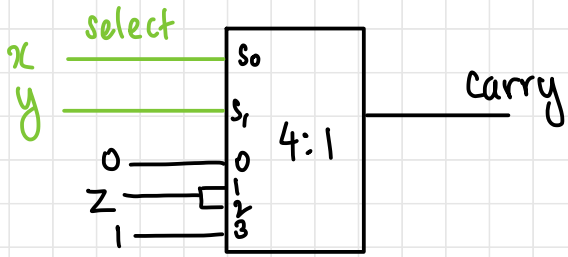
Full adder using MUX

$$S(x,y,z) = \Sigma(1,2,4,7)$$
$$C(x,y,z) = \Sigma(3,5,6,7)$$

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

z, z, z, 0, 2, 2, 1



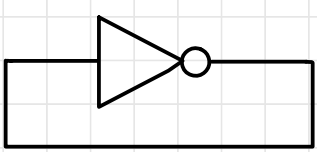


SEQUENTIAL LOGIC CIRCUITS

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

Simplest Memory Unit

Single inverter



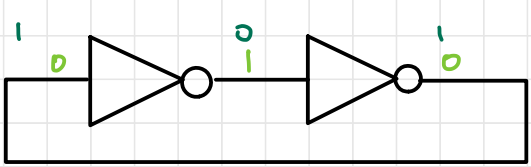
Assume O/P = 0 ⇒ I/P = 1
O/P = 1 ⇒ I/P = 0

impossible!!

astable output

- does not work

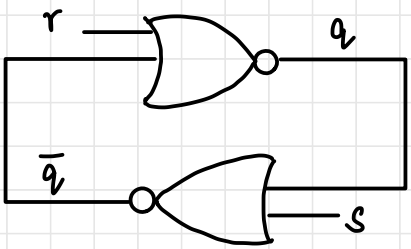
Two inverters



stable!!

SR Latch (Set-Reset Latch)

- Cross-coupled NOR gates



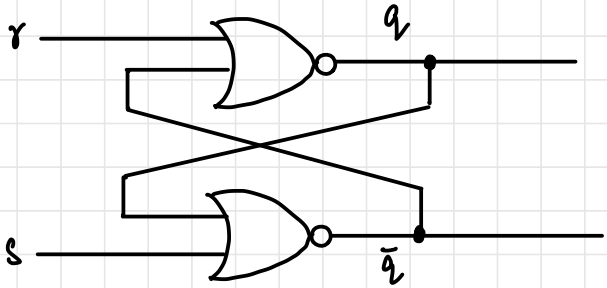
Truth table

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

stable, but to be avoided (forbidden)

also drawn like

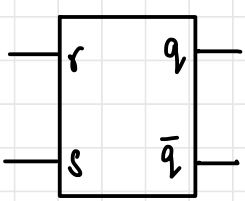


for $s=0, r=0$

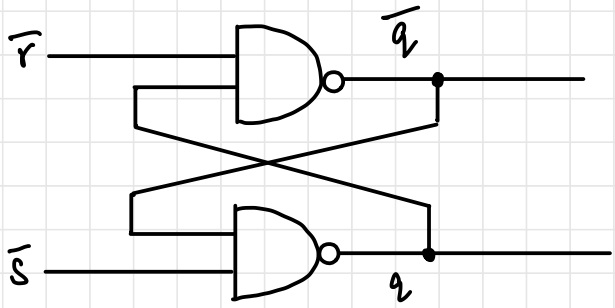
case 1: $q_{prev}=0$
$\therefore \bar{q}=1$
$q=0$

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

retains previous values



NAND-Based SR Latch

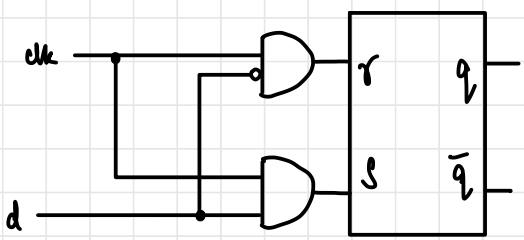


Truth table

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

forbidden

D Latch (Data)

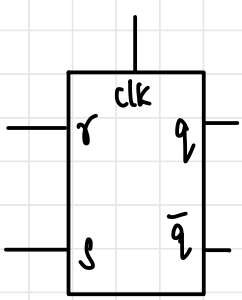


memory

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	0	1	0

transparent

- no indeterminate states
- level sensitive — changes when clk = 1

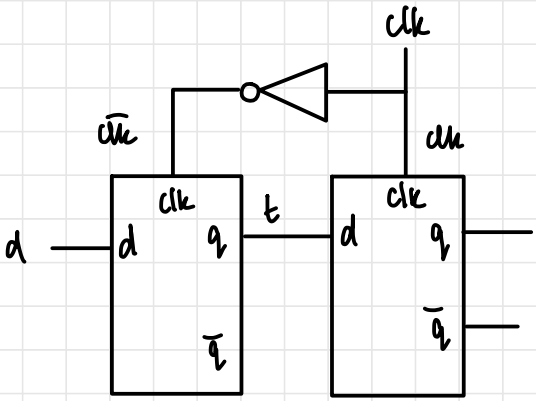


Level sensitive
- ↳ +ve level sensitive
- ↳ -ve level sensitive

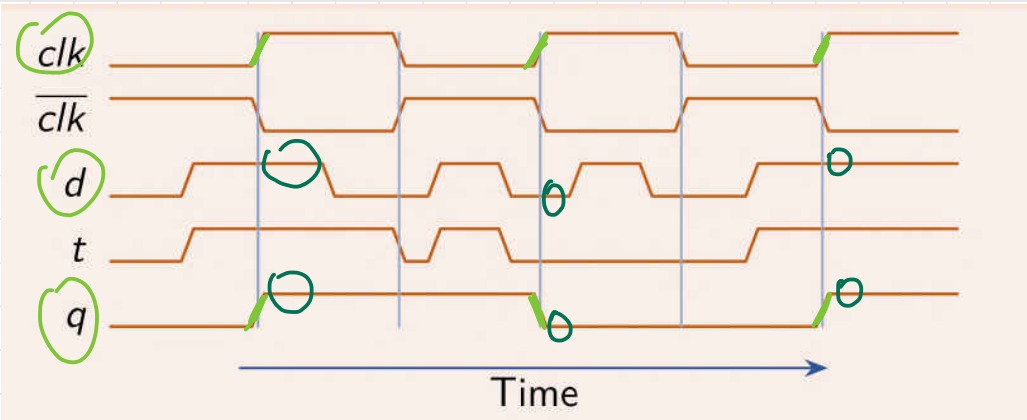
Edge sensitive
- ↳ rising edge trigger
- ↳ falling edge trigger

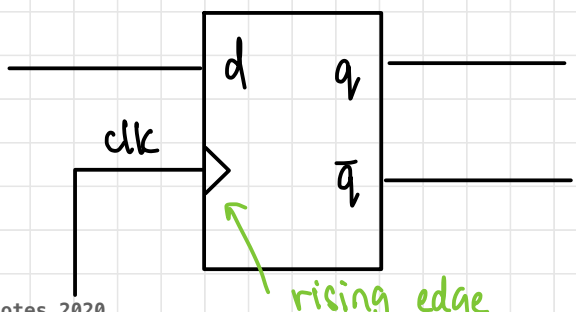
D - Flip Flop

- Edge triggered - flip flops



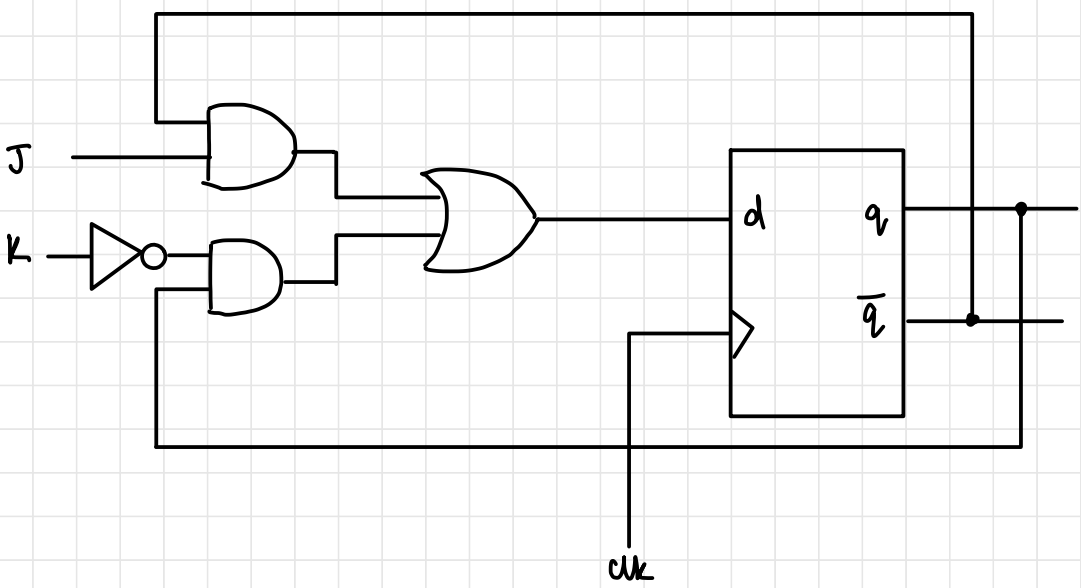
- when $\overline{clk}$ is 1, t reflects d
- when clk is 1, changes in d are ignored and prev t value is retained





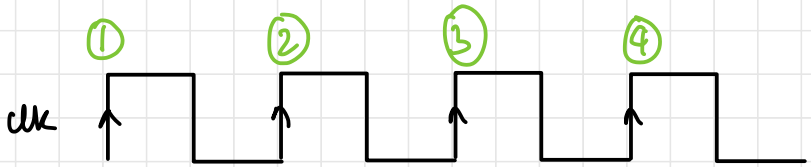
J-K Flip Flop

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



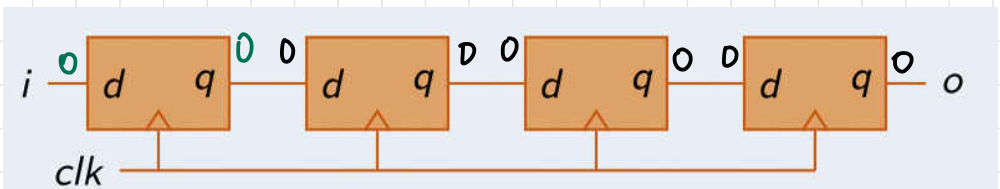
4-Bit Shift register

SISO

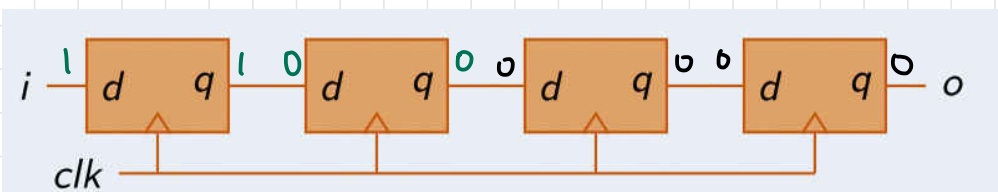


Input: 1010

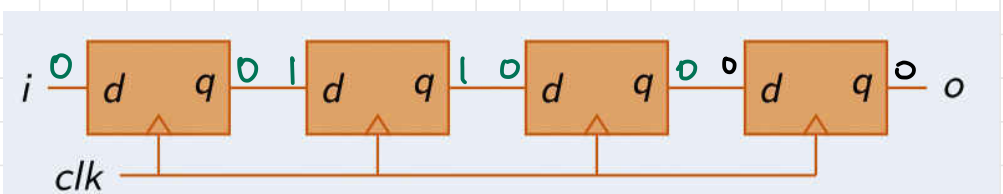
1



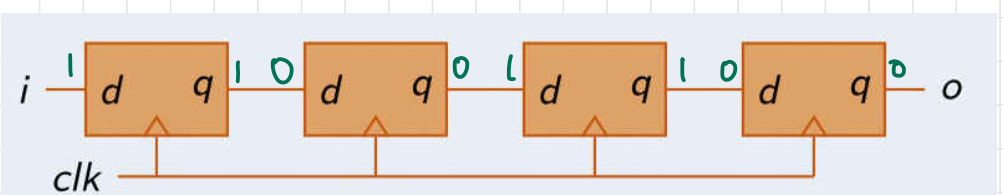
2



3

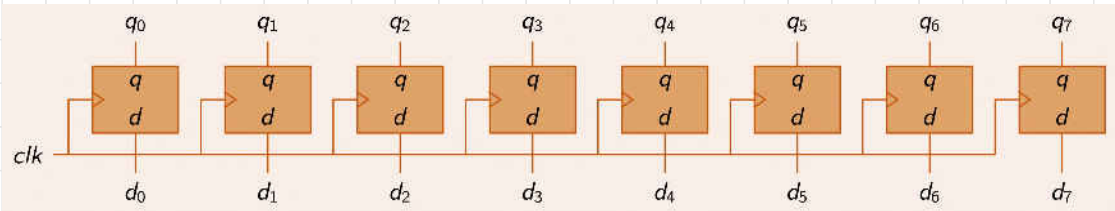


4

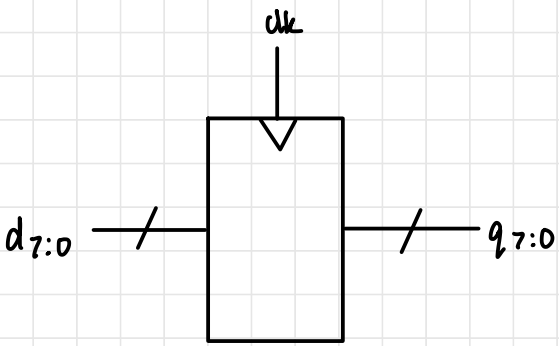


8-Bit Shift Register

PIPO (parallel in, parallel out)

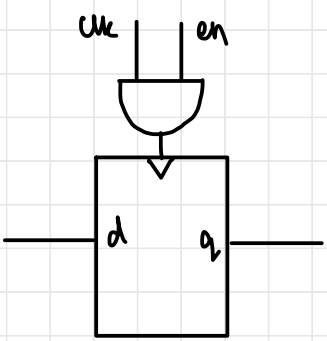


Symbol



Flip Flop with Enable

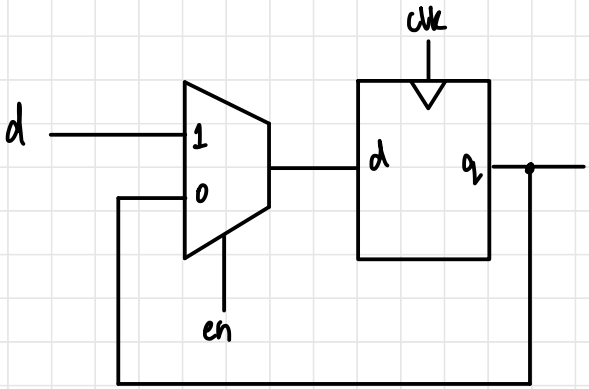
1) Clock Gating Approach

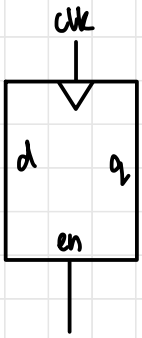


- clk signal is gated
- can cause timing problems
- careful use can reduce power consumption – useful here
- usually avoided

2) MUX

Symbol



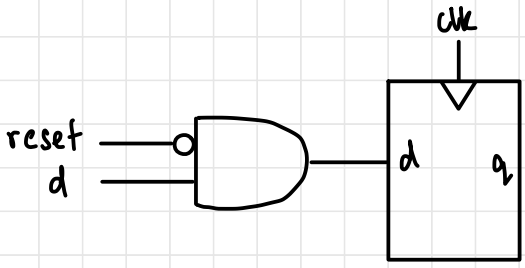


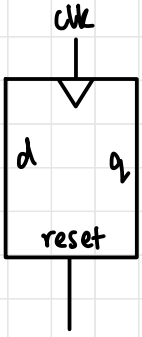
- At rising edge of clock

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

Resettable Flip Flop

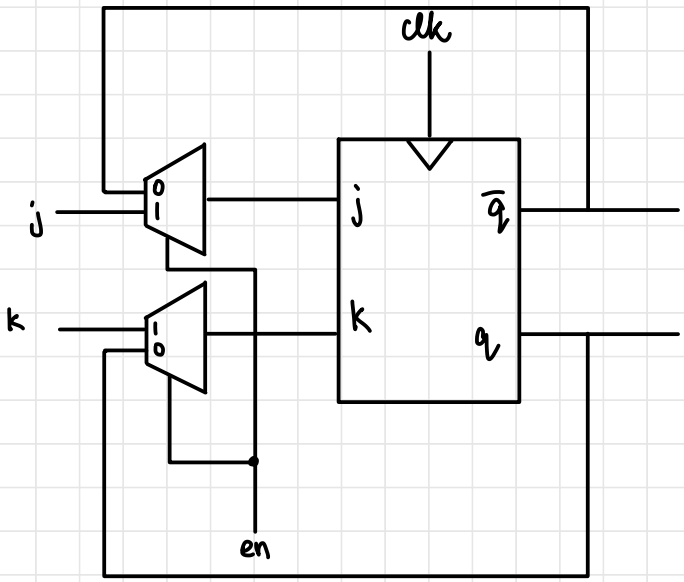
- When reset = 0, d-flip flop normal
- When reset = 1, d-flip flop stores 0
- Force memory elements into known state
- Initial power supply — prevent randomness





Construct JK Flip Flop Using enable / reset

Flip Flop with Enable



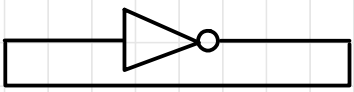
j-k flip-flop truth table

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

Synchronous Sequential Logic Circuits

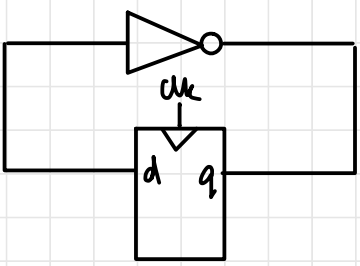
2) Loops

- inverter loop is a problem



unstable

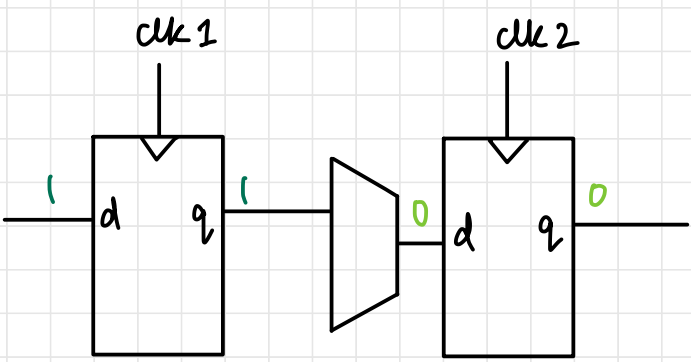
- solution: insert flip flop



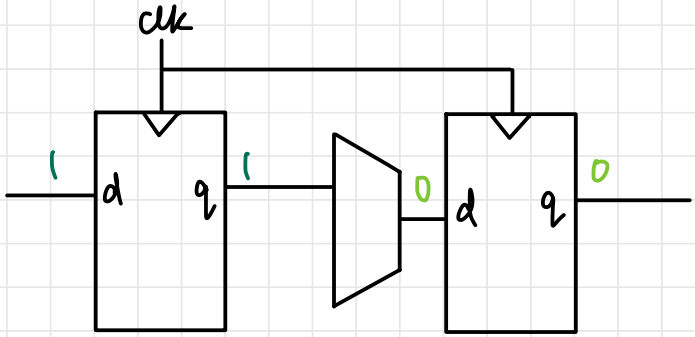
stable: toggles at every clock rising edge

1) Race Condition

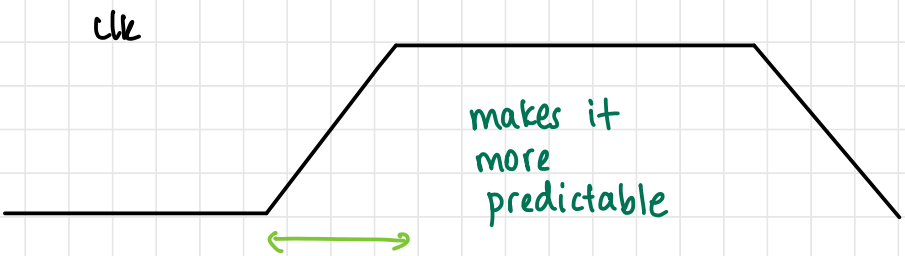
- Memory elements can suffer from race conditions (race hazards)
- depends on relative timing of clks
- if clk 1 arrives before clk 2, race condition



- solution: common clock synchronisation



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



if this time is greater than critical path delay, both happen in one cycle

Synchronous Sequential Logic Circuit

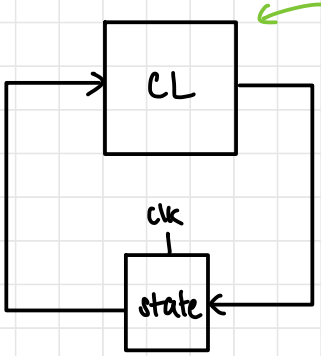
1) Every circuit element is either a register (flip flop) or a combinational circuit
   - ↳ no latches alone, only flip flops
   - ↳ think of flip flops as indivisible units
2) At least one circuit element is a register
3) All registers receive same clock signal
4) Every cyclic path has at least one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

Classify the following logic circuits as combinational or synchrnous 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



only combinational logic elements

each arrow represents wires

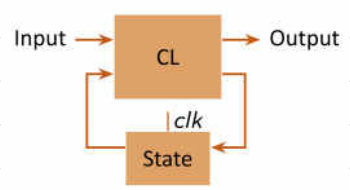
can contain any memory elements

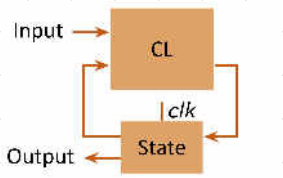
in this course, only synchronous, seq. logic circuits

specifically, D-flip flops

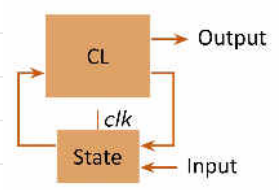
- Any sequential logic circuit (from a counter to a complex microprocessor) can be represented as an FSM
- Fundamental concept in CSE
- In AFLL, nodes and edges, but to implement, this

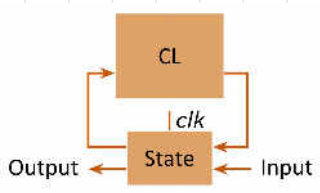
- This diagram lacks input & output
- Two types – Mealy type and Moore type FSMs
- Four input/output possibilities





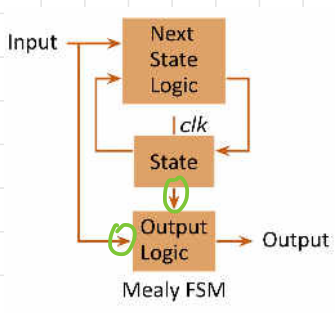
- if I/P given to state block, delayed by one clock cycle and therefore neglected





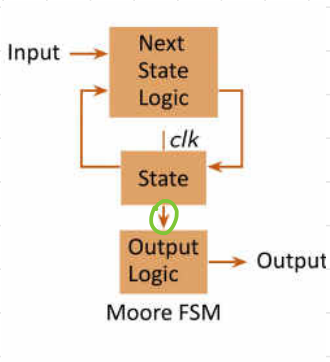
Mealy FSM

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



Moore FSM

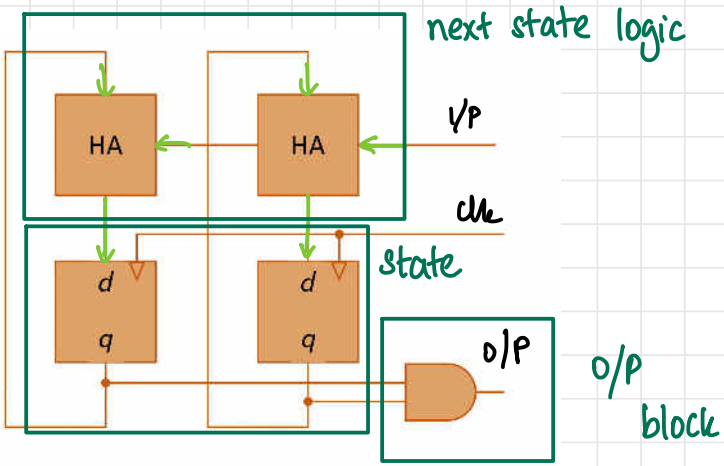
- Output depends only on current state



Question 7

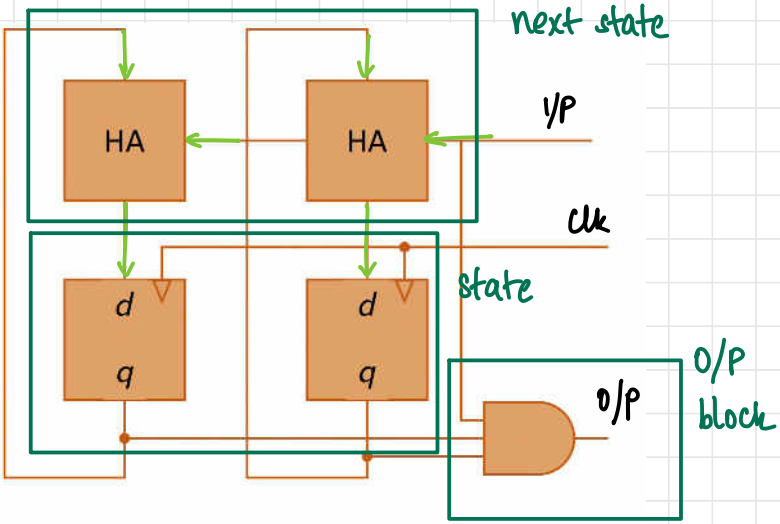
Identify as Mealy or Moore & make into blocks

(a)



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] ~truth tables
5. Output table]
6. Logic minimisation
   - ↳ Boolean formulas
7. Logic circuit construction

Question 8

Lift / elevator control logic (2 floors)

Inputs

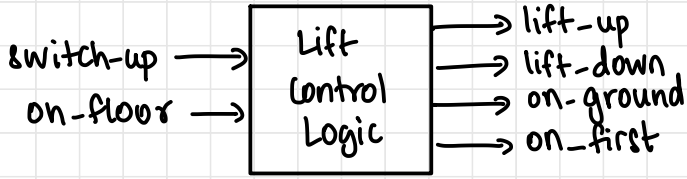
- Lift has a switch which can be in the down or up position, indicating that the lift should go down or up respectively
- The switch signal switch-up is 0 when lift should go down and is 1 when the lift should go up
- Lift system has a sensor which indicates when the lift is stationary or moving between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

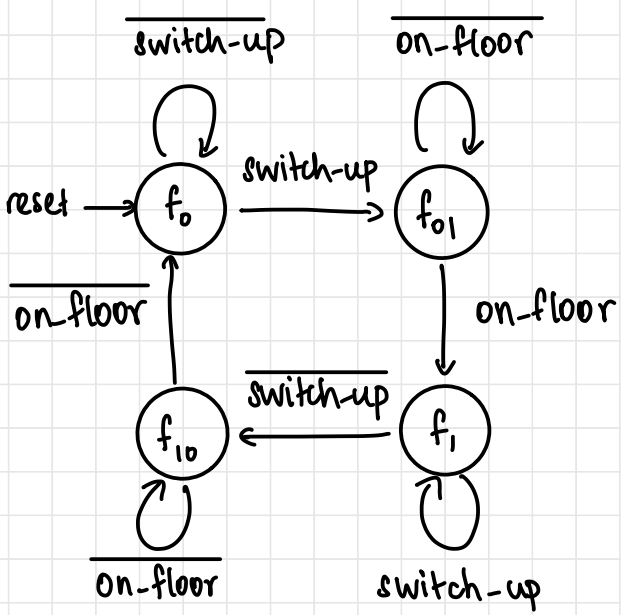
Reset / start state

- 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)
f_0	00
f_{01}	01
f_1	11
f_{10}	10

State	Encoding (s_1 s_0)
f0	00
f01	01
f1	11
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 Sate	
	s_1	s_0	switch_up	on_floor	s_1'	s_0'
f_0	0	0	0	0	0	0
	0	0	0	1	0	0
	0	0	1	0	0	1
	0	0	1	1	0	1
f_{01}	0	1	0	0	0	1
	0	1	0	1	1	1
	0	1	1	0	0	1
	0	1	1	1	1	1
f_{10}	1	0	0	0	1	0
	1	0	0	1	0	0
	1	0	1	0	1	0
	1	0	1	1	0	0
f_1	1	1	0	0	1	0
	1	1	0	1	1	0
	1	1	1	0	1	1
	1	1	1	1	1	1

no change

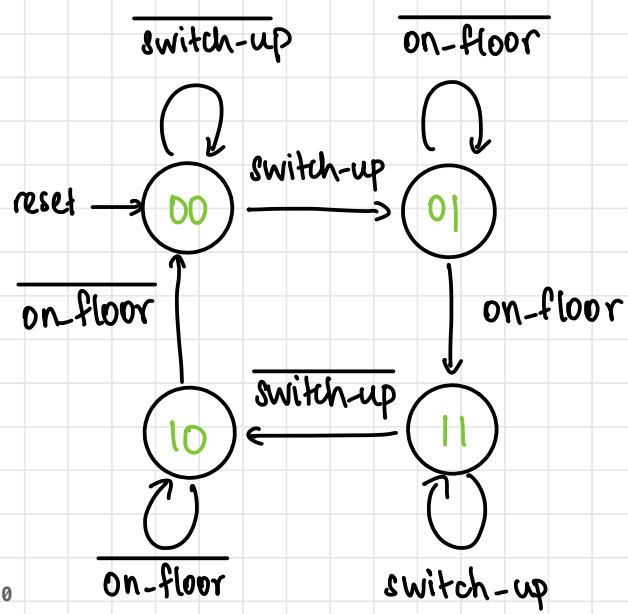
go to f_{01}

no change

go to f_0

go to f_{10}

no change



Output Table

State		Outputs			
s_1	s_0	on_ground	on_first	lift_up	lift_down
0	0	1	0	0	0
0	1	0	0	1	0
1	0	0	0	0	1
1	1	0	1	0	0

on floor

in trans.

K-map for s_1'

$s_1 s_0$ \ switch_up on_floor	00	01	11	10
$\bar{s}_1\bar{s}_0$ 00	0	0	0	0
$\bar{s}_1 s_0$ 01	0	1	1	0
$s_1 s_0$ 11	1	1	1	1
$s_1\bar{s}_0$ 10	1	0	0	1

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

K-map for s_0'

$s_1 s_0$ \ switch_up on_floor	00	01	11	10
$\bar{s}_1\bar{s}_0$ 00	0	0	1	1
$\bar{s}_1 s_0$ 01	1	1	1	1
$s_1 s_0$ 11	0	0	1	1
$s_1\bar{s}_0$ 10	0	0	0	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

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

$$s_0' = \bar{s}_1 s_0 + \text{switch\_up} \cdot \bar{s}_1 + \text{switch\_up} \cdot s_0$$

Outputs

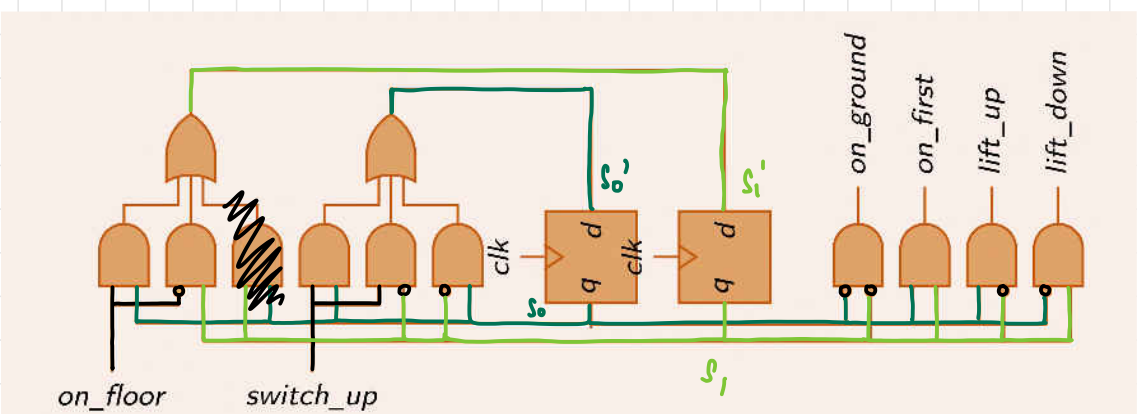
$$\text{on\_ground} = \bar{s}_1 \bar{s}_0$$

$$\text{on\_first} = s_1 s_0$$

$$\text{lift\_up} = \bar{s}_1 s_0$$

$$\text{lift\_down} = s_1 \bar{s}_0$$

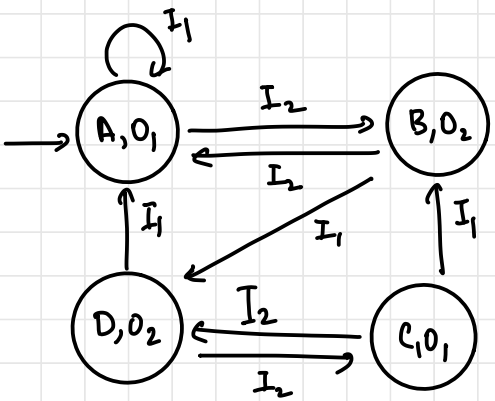
Logic Circuit



Moore FSM

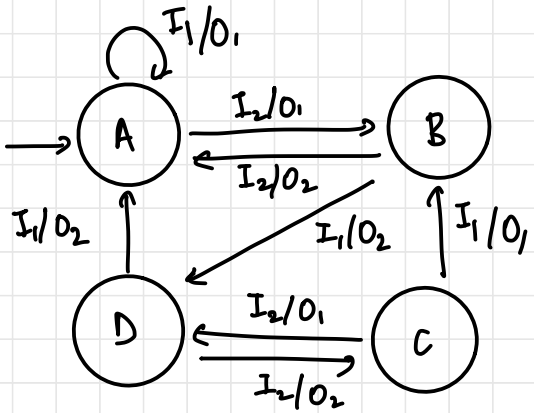
State table for Moore Machine

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



Mealy Machine

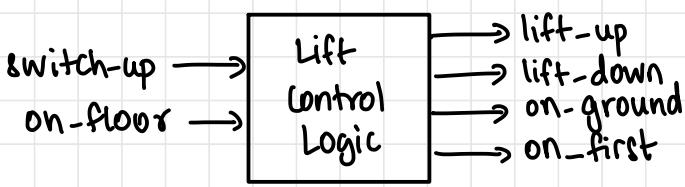
Present state	Next state, output Input I_1	Next state, output Input I_2
A	A,0	B,O_1
B	D,O_2	A,O_2
C	B,O_1	D,O_1
D	A,O_2	C,O_2



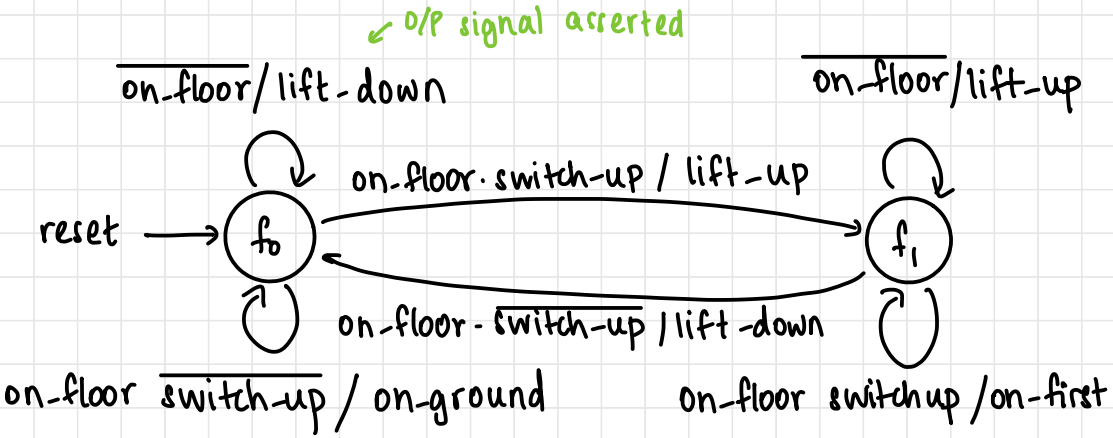
Question 9

Elevator Problem as a Mealy Machine

- depends on I/P



- 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_0	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

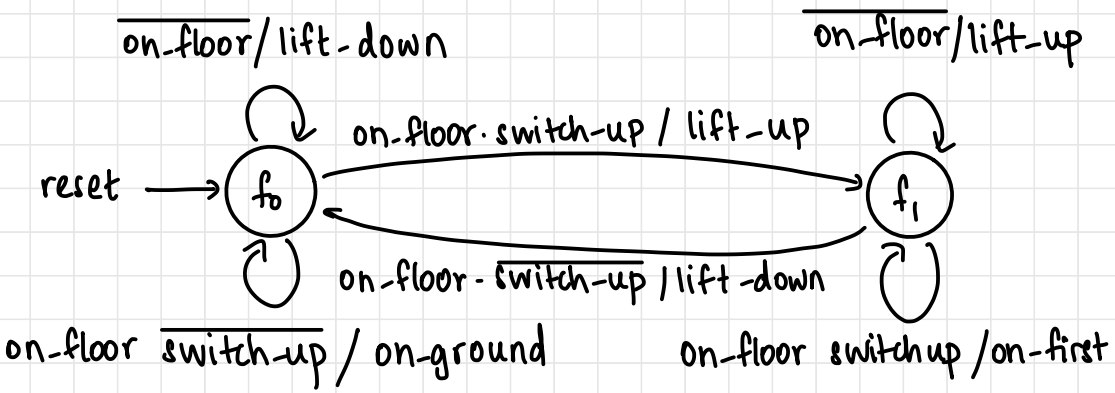
- lift_down

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

State Transition & Output Table — Mealy

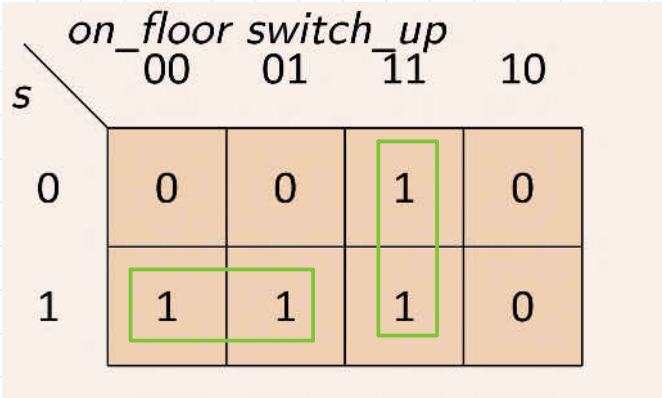
state output

Current State	Inputs		Next State	Outputs			
s	*on_floor*	*switch_up*	*s'*	*on_ground*	*on_first*	*lift_up*	*lift_down*
f_0 0	0 not on gnd	0 doesn't matter	0 stays	0 not yet	0 no	0 no	1 yes
f_0 0	0 not on gnd	1 doesn't matter	0 stays	0 not yet	0 no	0 no	1 yes
f_0 0	1 on gnd	0 stay	0 stays	1 on gnd	0 no	0 no	0 no
f_0 0	1 on gnd	1 go up	1 move	0 no	0 no	1 yes	0 no
f_1 1	0 not on first	0 doesn't matter	1 stays	0 no	0 not yet	1 yes	0 no
f_1 1	0 not on first	1 doesn't matter	1 stays	0 no	0 not yet	1 yes	0 no
f_1 1	1 on first	0 go down	0 move	0 no	0 no	0 no	1 yes
f_1 1	1 on first	1 stay	1 stays	0 no	1 yes	0 no	0 no



K-maps

K-map for s'



$s' = \overline{on\_floor} \cdot s + on\_floor \cdot switch\_up$

$on\_ground = \bar{s} \cdot on\_floor \cdot \overline{switch\_up}$

$on\_first = s \cdot on\_floor \cdot switch\_up$

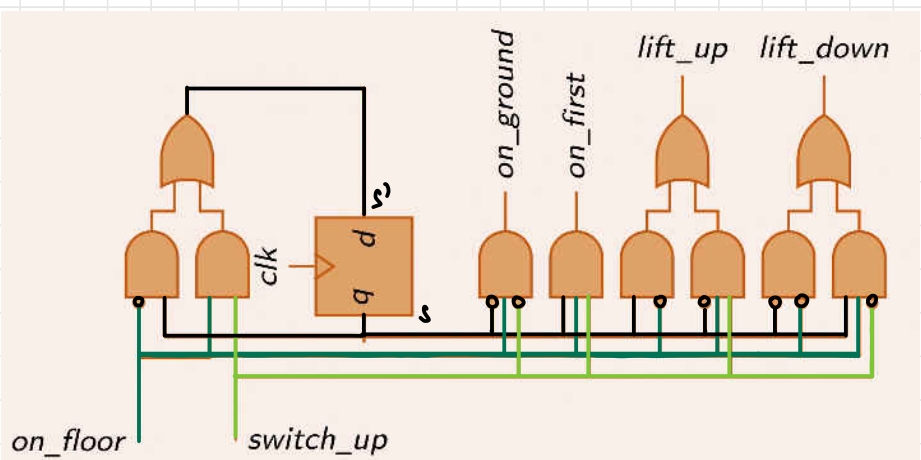
K-map for lift_up

$lift\_up = s \cdot \overline{on\_floor} + \bar{s} \cdot on\_floor \cdot switch\_up$

K-map for lift_down

$$lift\_down = \overline{on\_floor} \cdot s + on\_floor \cdot \overline{switch\_up} \cdot s$$

Logic Circuit



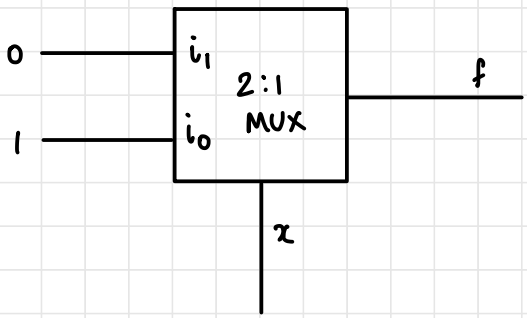
Mealy FSM

Revision Questions

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

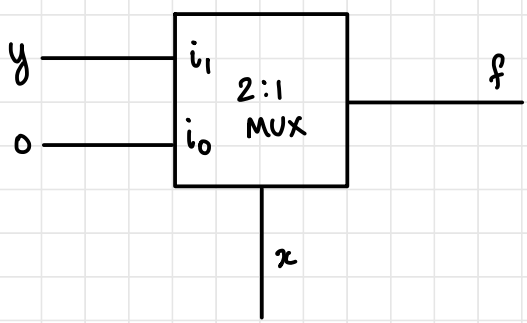
NOT

x	f
0	1
1	0



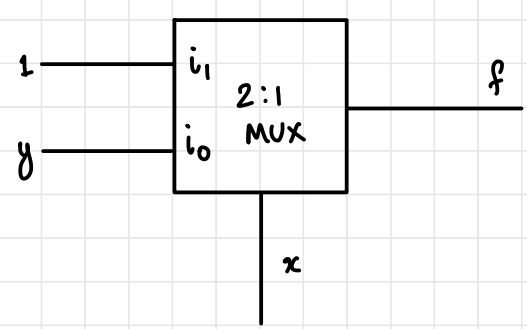
AND

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



OR

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

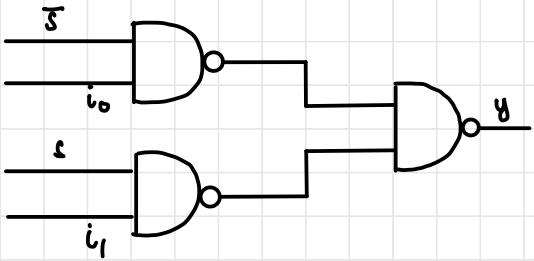


Q2. Construct 2:1 MUX using only 2-input NAND gates. How many gates are required?

selection	Output
0	i_0
1	i_1

$$y = \bar{s} i_0 + s i_1$$

$$= \overline{(\overline{\bar{s} i_0} \cdot \overline{(s i_1)})}$$



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