

Scheme Of Evaluation
Internal Assessment Test 2 – October 2019

Sub:	Analog and Digital Electronics						Code:	18CS33	
Date:	15/10/2019	Duration:	90mins	Max Marks:	50	Sem:	III	Branch:	ISE

Note: Answer Any Five Questions

Question #	Description	Marks Distribution		Max Marks
1	a) What is a three state buffer? Explain the 4 types of three state buffers along with their truth tables. <ul style="list-style-type: none"> • Buffer – description + purpose of buffer • Tri-state buffer with diagram • 4 types – description with truth tables 	1M	6M	10 M
	b) Implement a 4-to-1 MUX using four three-state buffers and a decoder. <ul style="list-style-type: none"> • Show the implementation • Basic description 	3M 1M		
2	a) Implement a full adder using 3 to 8 line decoder and 2 OR gates. <ul style="list-style-type: none"> • Truth table and expression for sum and carry • Implementation 	2M 3M	5M	10 M
	b) Implement a full subtractor using 3 to 8 line decoder with inverting outputs and 2 NAND gates. <ul style="list-style-type: none"> • Truth table and expression for difference and borrow • Implementation 	2M 3M		
3	a) Derive the logic equations for a 4 to 2 line priority encoder and implement the same using basic gates. <ul style="list-style-type: none"> • Disadvantage of 4 to 2 line priority encoder with truth table • Priority encoder – truth table and description • K-Map for outputs of priority encoder + 	1M 2M	6M	10 M

		<p>Expression for each output</p> <ul style="list-style-type: none"> Implementation using basic gates 	2M 1M																																																																										
	b)	<p>Implement an 8 to 3 priority encoder using two 4 to 2 priority encoders.</p> <ul style="list-style-type: none"> Implementation Explanation for how to get the expressions of outputs and valid output indicator 	3M 1M	4M																																																																									
4		<p>Braille is a system which allows a blind person to read alpha-numeric by feeling a pattern of raised dots. Design a circuit using PLA that converts BCD to Braille. The below table shows the correspondence between BCD (denoted by ABCD) and Braille (denoted by WXYZ).</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="4"></th> <th style="border-right: 1px solid black;"><i>W</i></th> <th><i>X</i></th> </tr> <tr> <th style="border-bottom: 1px solid black;"><i>A</i></th> <th style="border-bottom: 1px solid black;"><i>B</i></th> <th style="border-bottom: 1px solid black;"><i>C</i></th> <th style="border-bottom: 1px solid black;"><i>D</i></th> <th style="border-right: 1px solid black;"><i>Z</i></th> <th><i>Y</i></th> </tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td style="border-right: 1px solid black;">.</td><td>:</td> </tr> <tr> <td>0</td><td>0</td><td>0</td><td>1</td><td style="border-right: 1px solid black;">.</td><td></td> </tr> <tr> <td>0</td><td>0</td><td>1</td><td>0</td><td style="border-right: 1px solid black;">:</td><td></td> </tr> <tr> <td>0</td><td>0</td><td>1</td><td>1</td><td style="border-right: 1px solid black;">.</td><td>.</td> </tr> <tr> <td>0</td><td>1</td><td>0</td><td>0</td><td style="border-right: 1px solid black;">.</td><td>:</td> </tr> <tr> <td>0</td><td>1</td><td>0</td><td>1</td><td style="border-right: 1px solid black;">.</td><td>.</td> </tr> <tr> <td>0</td><td>1</td><td>1</td><td>0</td><td style="border-right: 1px solid black;">:</td><td>.</td> </tr> <tr> <td>0</td><td>1</td><td>1</td><td>1</td><td style="border-right: 1px solid black;">:</td><td>:</td> </tr> <tr> <td>1</td><td>0</td><td>0</td><td>0</td><td style="border-right: 1px solid black;">:</td><td>.</td> </tr> <tr> <td>1</td><td>0</td><td>0</td><td>1</td><td style="border-right: 1px solid black;">.</td><td>.</td> </tr> </tbody> </table> <ul style="list-style-type: none"> Truth table Expressions for W, X, Y and Z using K-Map PLA table Implementation 					<i>W</i>	<i>X</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>Z</i>	<i>Y</i>	0	0	0	0	.	:	0	0	0	1	.		0	0	1	0	:		0	0	1	1	.	.	0	1	0	0	.	:	0	1	0	1	.	.	0	1	1	0	:	.	0	1	1	1	:	:	1	0	0	0	:	.	1	0	0	1	.	.	2M 4M 2M 2M	10M	10 M
				<i>W</i>	<i>X</i>																																																																								
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>Z</i>	<i>Y</i>																																																																								
0	0	0	0	.	:																																																																								
0	0	0	1	.																																																																									
0	0	1	0	:																																																																									
0	0	1	1	.	.																																																																								
0	1	0	0	.	:																																																																								
0	1	0	1	.	.																																																																								
0	1	1	0	:	.																																																																								
0	1	1	1	:	:																																																																								
1	0	0	0	:	.																																																																								
1	0	0	1	.	.																																																																								
5	a)	<p>A circuit has four inputs RSTU and four outputs VWYZ. RSTU represents a binary coded-decimal digit. VW represents the quotient and YZ the remainder when RSTU is divided by 3 (VW and YZ represent 2-bit binary numbers). Assume that invalid inputs do not occur. Realize the circuit using a PLA.</p> <ul style="list-style-type: none"> Truth table Expressions for W, X, Y and Z using K-Map PLA table Implementation 	2M 4M 2M 2M	10M	10 M																																																																								

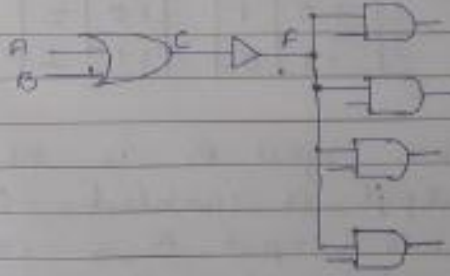
6	a)	<p>Generate the sum and carry functions of a full adder using a PAL.</p> <ul style="list-style-type: none"> • Truth table of full adder + Expressions for sum and carry • Implementation of sum and carry on PAL 	2M 4M	6M	10M
	b)	<p>What are PLDs? Distinguish between the 3 main types of PLDs.</p> <ul style="list-style-type: none"> • Description of PLDs • 3 types with general block diagram and description 	1M 3M	4M	
7	a)	<p>Explain the working of an SR latch constructed using NOR gates.</p> <ul style="list-style-type: none"> • SR Latch diagrams • Working of SR latch for each input combination 	2M 2M	4M	10 M
	b)	<p>Explain switch contact bounce circuits. How can contact bounce be eliminated in mechanical switches?</p> <ul style="list-style-type: none"> • Circuit diagrams + output waveforms • Description of contact bounce occurrence • How to avoid contact bounce 	2M 2M 2M	6M	
8		<p>With necessary diagrams and proper explanation, derive the truth table, state transition diagram, characteristic equation and excitation table for an SR flip-flop and a JK flip-flop.</p> <ul style="list-style-type: none"> • Definition of state transition diagram, characteristic equation and excitation table • Truth table, state transition diagram, characteristic equation and excitation table for SR flip-flop • Truth table, state transition diagram, characteristic equation and excitation table for JK flip-flop 	2M 4M 4M	10M	10 M

1a). What is a three state buffer? Explain the 4 types of three state buffers along with their truth tables.

papergrid
Date: / /

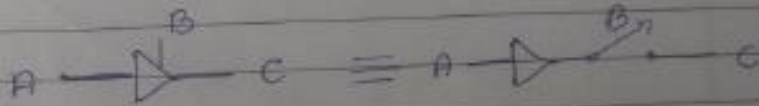
Three State Buffers

A gate output can only be connected to a limited number of other device inputs without degrading the performance of a digital system. A simple buffer may be used to increase the driving capability of a gate output. Figure below shows buffer connected between a gate output and several gate inputs. Because no bubble is present at the buffer output, this is a non-inverting buffer and the logic value of the buffer input and output are the same, that is $F = C$.



Gate circuit with added Buffer.

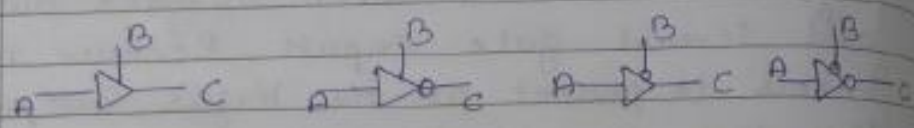
Use of three-state logic permits the outputs of two or more gates to be connected together. Figure below shows a three-state buffer and its logical equivalent.



Three State Buffer

When the enable input B is 1, the output C equals A . When B is 0, the output C acts like an open circuit.

Figure below shows the truth tables for four types of three-state buffers.



B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1

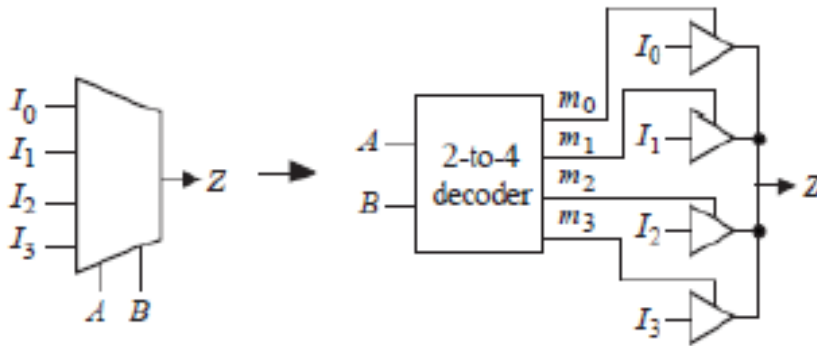
B	A	C
0	0	Z
0	1	Z
1	0	1
1	1	0

B	A	C
0	0	0
0	1	1
1	0	Z
1	1	Z

B	A	C
0	0	1
0	1	0
1	0	Z
1	1	Z

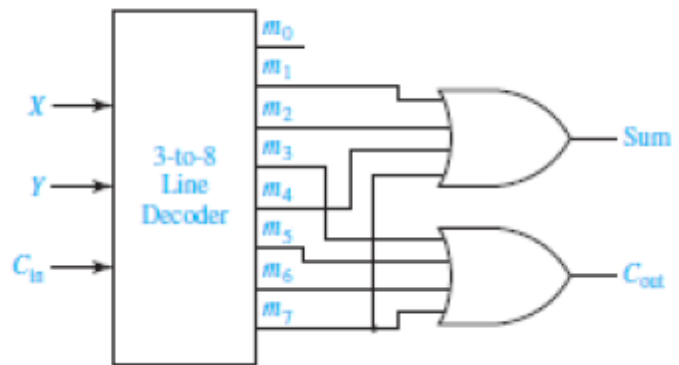
When enable input B is not inverted, buffer output is enabled, ($B=1$)
 when enable input B is inverted ($B=0$), buffer output is disabled.
 We use a symbol Z to represent high impedance state that occurs when $B=0$, i.e. B is inverted.

1b). Implement a 4-to-1 MUX using four three-state buffers and a decoder.



2a). Implement a full adder using 3 to 8 line decoder and 2 OR gates.

Input			Output	
A	B	Cin	Sum	Carry
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



$$\text{Sum} = \sum m(1, 2, 4, 7)$$

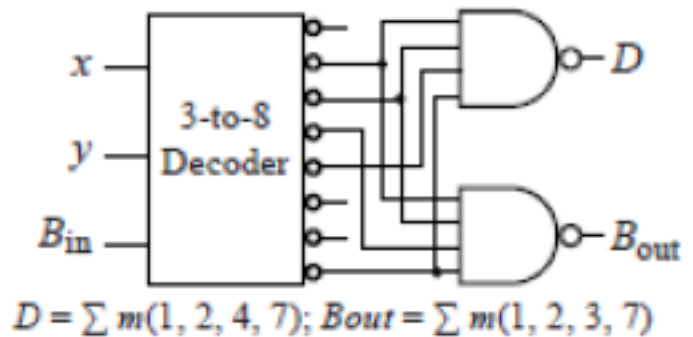
$$\text{Carry} = \sum m(3, 5, 6, 7)$$

2b). Implement a full subtractor using 3 to 8 line decoder with inverting outputs and 2 NAND gates.

Input			Output	
A	B	C	Difference	Borrow
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

$$\text{Difference} = \sum m(1, 2, 4, 7)$$

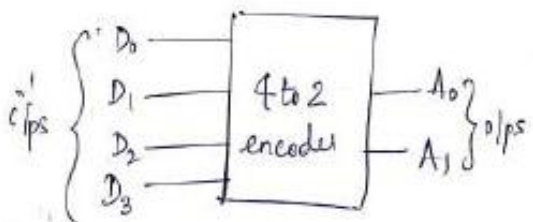
$$\text{Borrow} = \sum m(1, 2, 3, 7)$$



$$D = \sum m(1, 2, 4, 7); \text{Bout} = \sum m(1, 2, 3, 7)$$

3a). Derive the logic equations for a 4 to 2 line priority encoder and implement the same using basic gates.

4 to 2 bit binary encoder



D_3	D_2	D_1	D_0	A_1	A_0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Disadvantage:

(i) o/p code of all D_i can occur in 2 cases - when all i/p's are 0, and when only LSB is high active.

(ii) If 2 bits are active at the same time, it produces the wrong o/p code.

→ To overcome this, we go for Priority encoder.

→ In a priority encoder, we assign a priority to each i/p bit. Say, for a 4 bit priority encoder, we assign highest priority to D_3 and least priority to D_0 .

→ So if 2 i/p's are high at the same time, only the i/p with highest priority is considered.

eg:- for a 4 bit priority encoder,

D_3 → highest priority
 D_0 → lowest priority

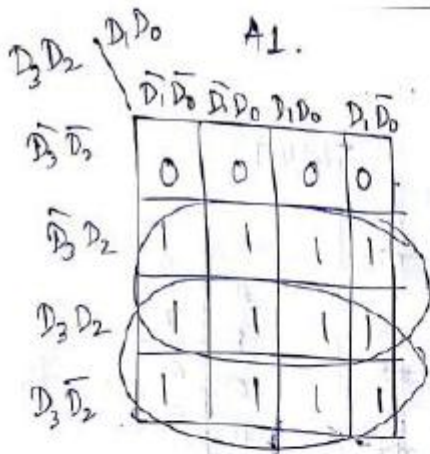
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

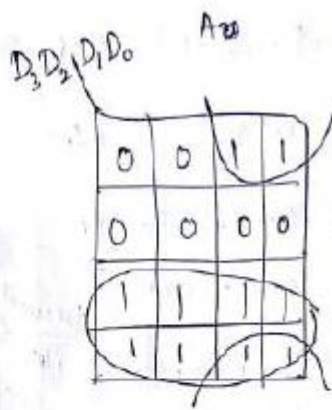
V is a valid o/p indicator.

$V=1$ only if atleast one of the i/p's is active.

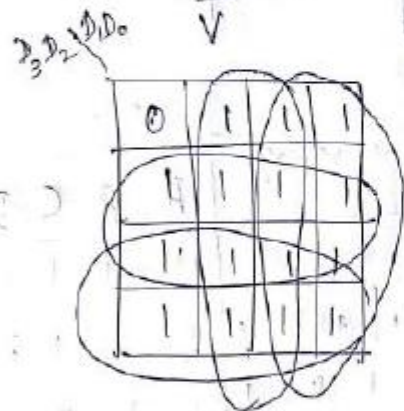
So when all i/p's are 0, V is equal to 0. Thus we can distinguish b/w 0000 and 0001 condition of i/p's.



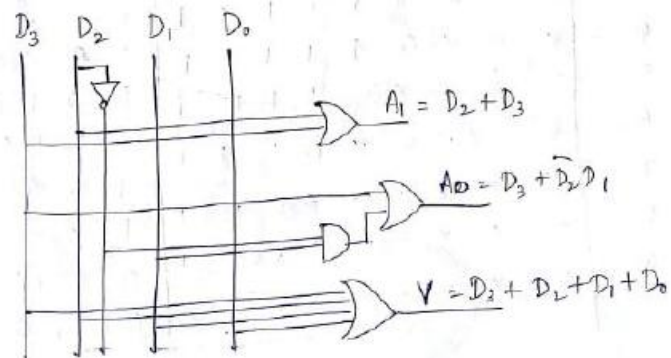
$$A_1 = \underline{D_2 + D_3}$$



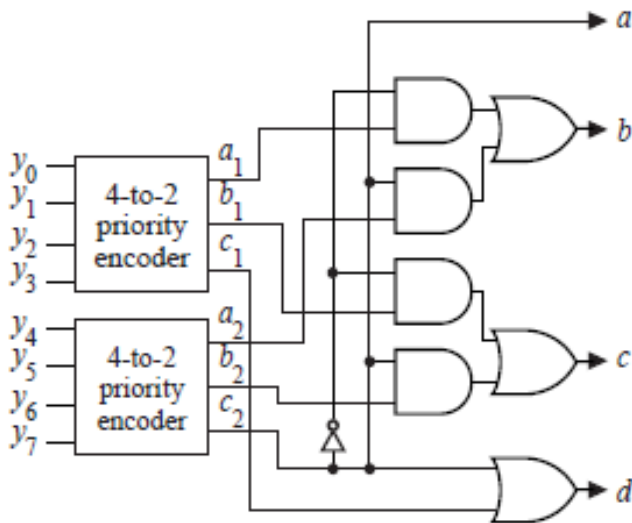
$$A_0 = \underline{D_3 + \bar{D}_2 D_1}$$



$$\Rightarrow V = \underline{D_3 + D_2 + D_1 + D_0}$$



3b). Implement an 8 to 3 priority encoder using two 4 to 2 priority encoders.



If any of the inputs y_0 through y_7 is 1, then d of the 8-to-3 decoder should be 1. But in that case, c_1 or c_2 of one of the 4-to-2 decoders will be 1. So $d = c_1 + c_2$.

If one of the inputs $y_4, y_5, y_6,$ and y_7 is 1, then a should be 1, and b and c should correspond to a_2 and b_2 , respectively. Otherwise, a should be 0, and b and c should correspond to a_1 and b_1 , respectively. So $a = c_2$, $b = c_2a_2 + c_2'a_1$, and $c = c_2b_2 + c_2'b_1$.

4. Braille is a system which allows a blind person to read alpha-numeric by feeling a pattern of raised dots. Design a circuit using PLA that converts BCD to Braille. The below table shows the correspondence between BCD (denoted by ABCD) and Braille (denoted by WXYZ).

A	B	C	D	W	X
				Z	Y
0	0	0	0	•	•
0	0	0	1	•	
0	0	1	0	•	
0	0	1	1	•	•
0	1	0	0	•	•
0	1	0	1	•	•
0	1	1	0	•	•
0	1	1	1	•	•
1	0	0	0	•	•
1	0	0	1	•	•

Handwritten student work showing a truth table, a Karnaugh map, and a PLA implementation for converting BCD to Braille.

The truth table is identical to the one above.

The Karnaugh map for W and X is shown below:

AB	CD	W	X
00	00	0	0
00	01	0	0
00	11	0	0
00	10	0	0
01	00	0	0
01	01	0	0
01	11	0	0
01	10	0	0
11	00	1	1
11	01	1	1
11	11	1	1
11	10	1	1
10	00	1	1
10	01	1	1
10	11	1	1
10	10	1	1

The PLA implementation shows the logic for W, X, Y, and Z:

$W = B + C + \bar{A}D + A\bar{D}$

$X = CD + BC + \bar{A}D + \bar{A}C\bar{D}$

	CD			
AB	00	01	11	10
00	1	0	0	0
01	1	1	1	0
11	x	x	x	x
10	1	0	x	x

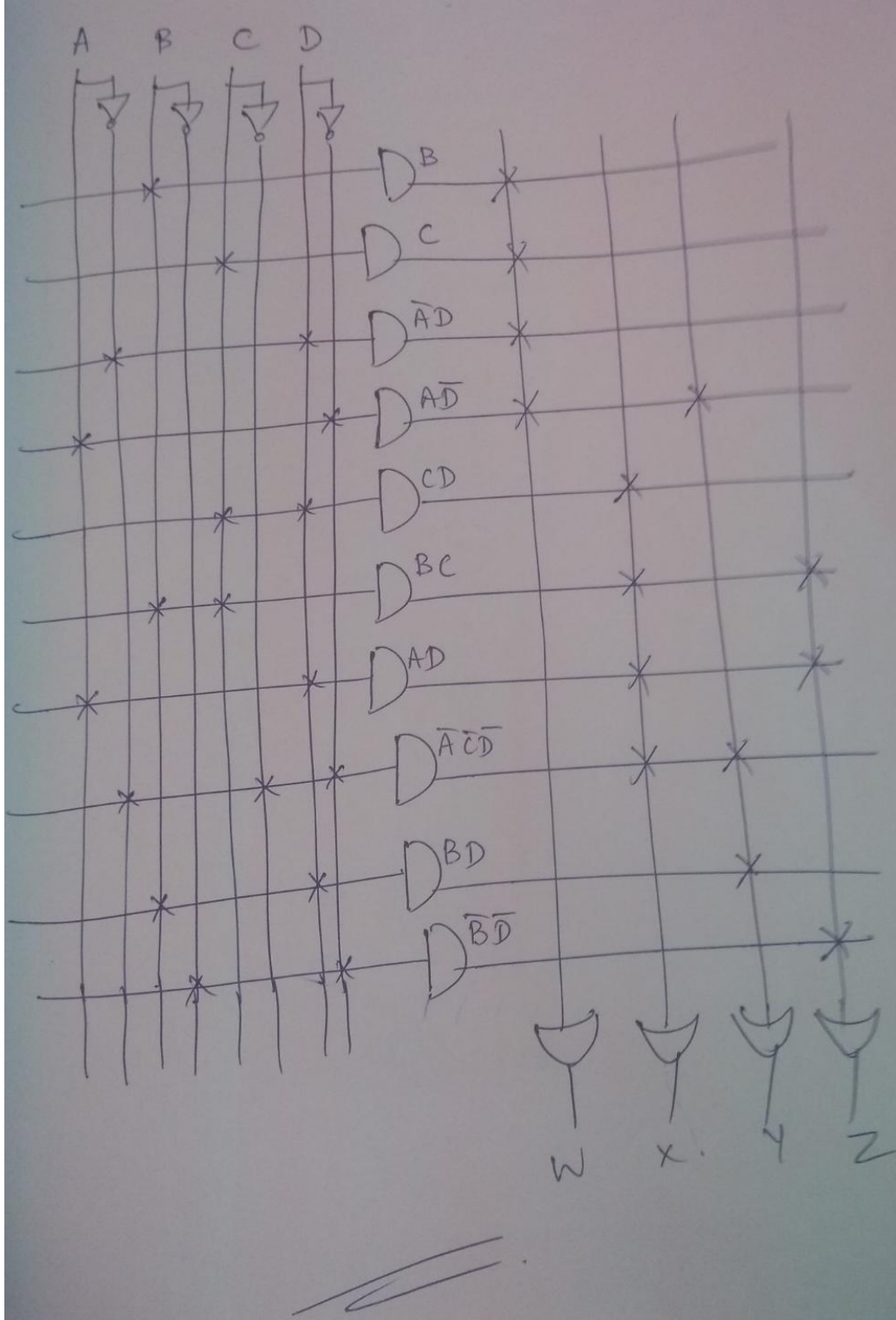
$$Y = BD + \bar{A}\bar{C}\bar{D} + A\bar{D}$$

	CD			
AB	00	01	11	10
00	1	0	0	1
01	0	0	1	1
11	x	x	x	x
10	1	1	x	x

$$Z = BC + AD + \bar{B}\bar{D}$$

PLA Program Table

Product Terms	A	B	C	D	W	X	Y	Z
B	-	1	-	-	1	0	0	0
C	-	-	1	-	1	0	0	0
$\bar{A}D$	0	-	-	1	1	0	0	0
$A\bar{D}$	1	-	-	0	1	0	0	0
CD	-	-	1	1	0	1	0	0
BC	-	1	1	-	0	1	0	1
AD	1	-	-	1	0	1	0	1
$\bar{A}\bar{C}\bar{D}$	0	-	0	0	0	1	1	0
BD	-	1	-	1	0	0	1	0
$\bar{B}\bar{D}$	-	0	-	0	0	0	0	1
					T	T	T	T



5. A circuit has four inputs RSTU and four outputs VWYZ. RSTU represents a binary coded-decimal digit. VW represents the quotient and YZ the remainder when RSTU is divided by 3 (VW and YZ represent 2-bit binary numbers). Assume that invalid inputs do not occur. Realize the circuit using a PLA.

R	S	T	U	V	W	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	1	0	0	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

VW → quotient of $RSTU/3$
YZ → remainder when $RSTU/3$
RSTU → BCD code

Since BCD code exists only from 0 to 9.

RS \ TU	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	X	X	X	X
10	1	1	X	X

$$V = R + ST$$

RS \ TU	00	01	11	10
00	0	0	0	1
01	0	1	0	0
11	X	X	X	X
10	1	0	X	X

$$Y = \underline{\underline{STU}} + \underline{\underline{RU}} + \underline{\underline{\bar{S}TU}}$$

RS \ TU	00	01	11	10
00	0	0	1	0
01	1	1	0	0
11	X	X	X	X
10	0	1	X	X

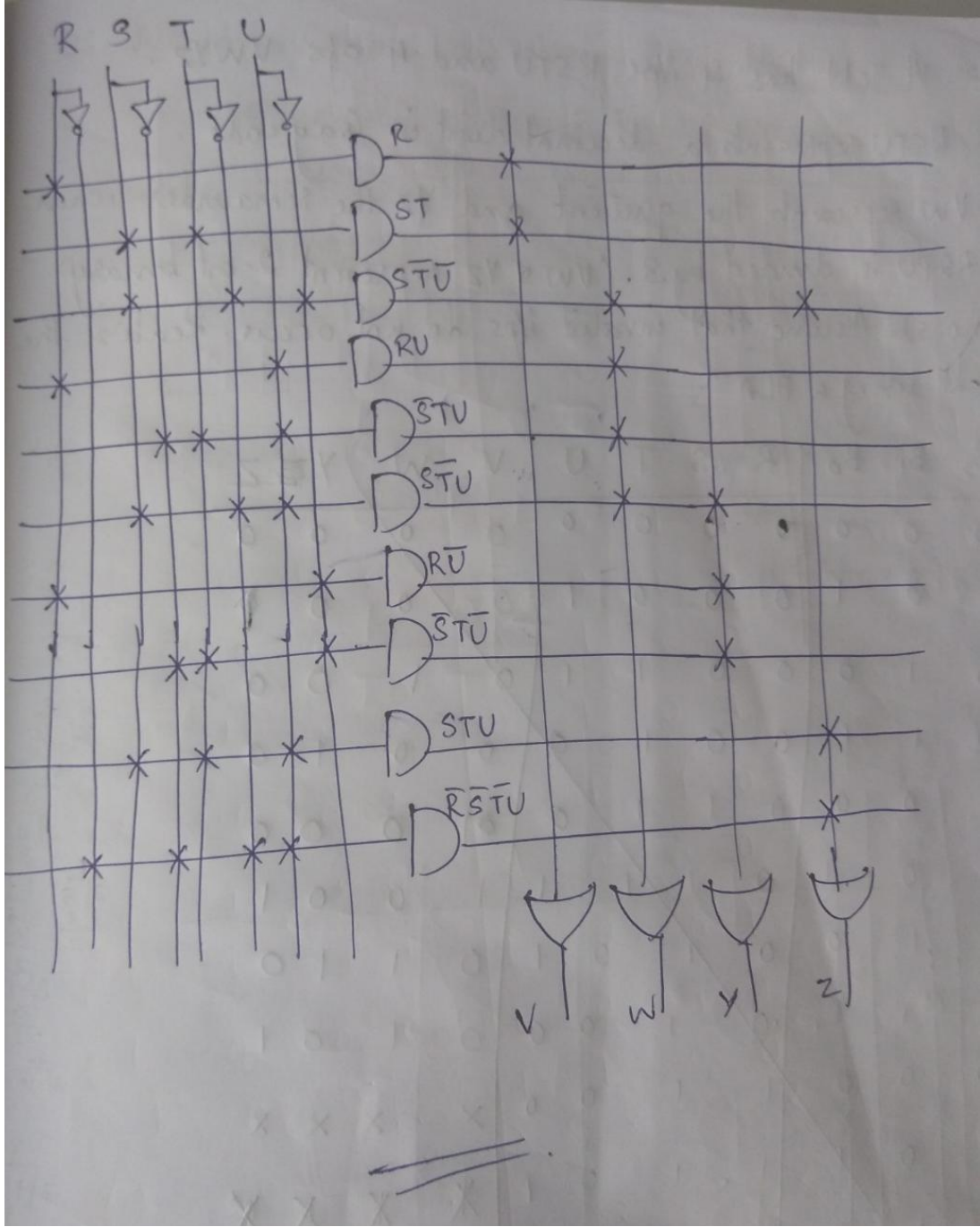
$$W = \underline{\underline{STU}} + \underline{\underline{RU}} + \underline{\underline{\bar{S}TU}} + \underline{\underline{STU}}$$

RS \ TU	00	01	11	10
00	0	1	0	0
01	1	0	1	0
11	X	X	X	X
10	0	0	X	X

$$Z = \underline{\underline{STU}} + \underline{\underline{STU}} + \underline{\underline{\bar{R}STU}}$$

PLA program table

Product Terms	R	S	T	U	V	W	Y	Z
R	1	-	-	-	1	0	0	0
ST	-	1	1	-	1	0	0	0
$\bar{S}U$	-	1	0	0	0	1	0	1
RU	1	-	-	1	0	1	0	0
$\bar{S}TU$	-	0	1	1	0	1	0	0
$\bar{S}U$	-	1	0	1	0	1	1	0
$\bar{R}U$	1	-	-	0	0	0	1	0
$\bar{S}TU$	-	0	1	0	0	0	1	0
STU	-	1	1	1	0	0	0	1
$\bar{R}\bar{S}U$	0	0	0	1	0	0	0	1

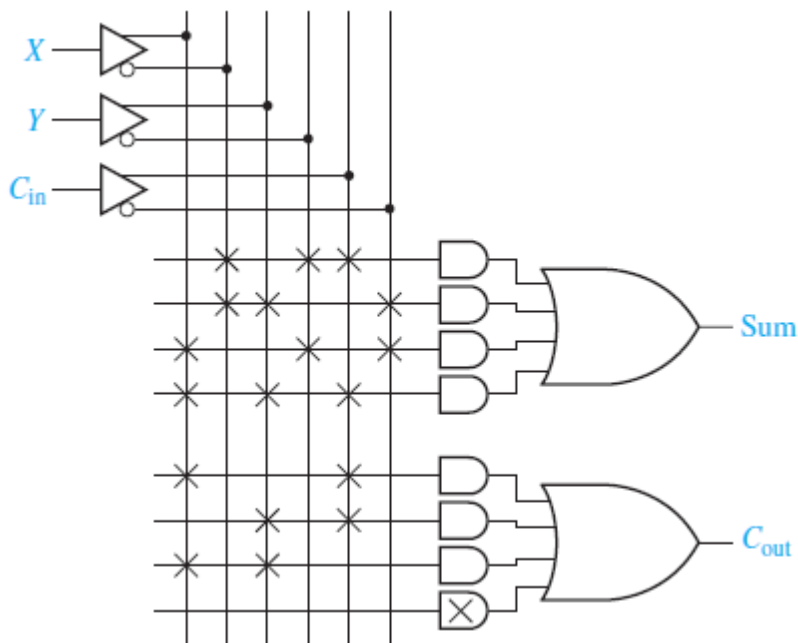


6a). Generate the sum and carry functions of a full adder using a PAL.

Inputs			Outputs	
x	y	c _{in}	c _{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$Sum = X'Y'C_{in} + X'YC'_{in} + XY'C'_{in} + XYC_{in}$$

$$C_{out} = XC_{in} + YC_{in} + XY$$



6b). What are PLDs? Distinguish between the 3 main types of PLDs.

PLDs (Programmable Logic Devices)

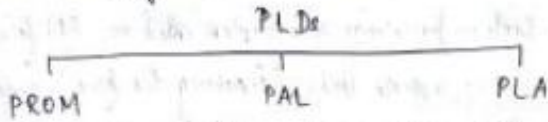
- An IC that contains large no. of gates, flipflops etc that can be configured by the user to perform different functions. refers to hardware programming
- This configuration is done using a programming process. The simplest programming technologies use fuses. In the original state of the device, all the fuses are intact. Programming the device involves blowing those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function.
- PLDs are typically built with an array of AND gates (AND-array) and an array of OR gates (OR-array)



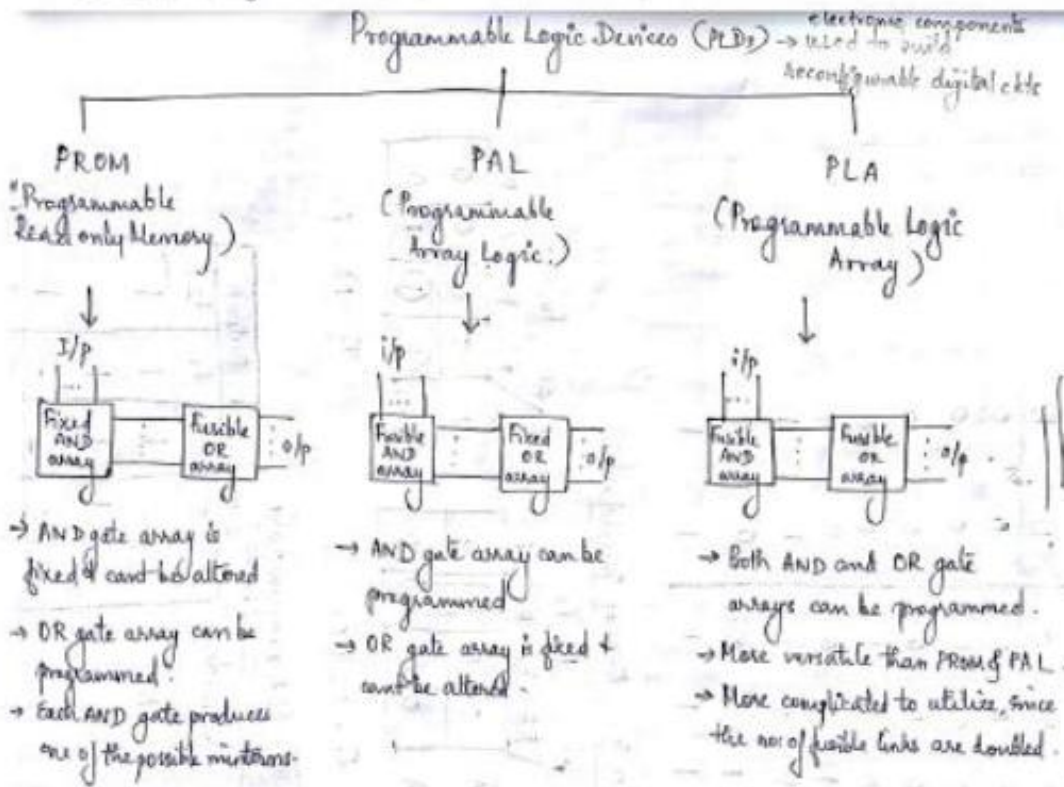
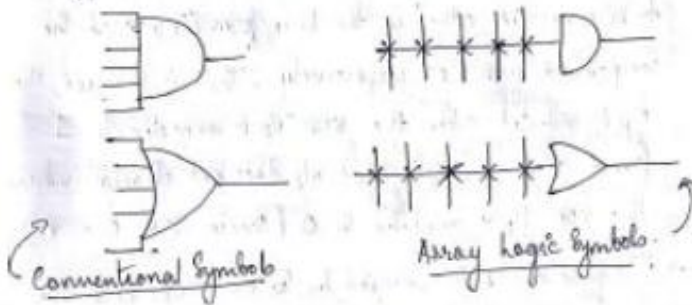
Advantages of PLDs :-

- While using standard ICs in logic design, hundreds or thousands of such ICs would be required, which take up considerable amount of circuit board space, more time and cost in inserting, soldering and testing. Also it requires keeping a significant inventory of ICs.
- PLDs take up less board space, they are faster and have lower power requirements (i.e. smaller power supplies). They have less costly assembly processes, higher reliability (fewer ICs)

and ckt connections means easier troubleshooting) and availability of design software.



→ A typical PLD may have hundreds to millions of gates. So in order to show the internal logic diagram for such technologies in a concise form, special symbols for array logic are used.



7a). Explain the working of an SR latch constructed using NOR gates.

Date: / /

SR Flip Flop using NOR gate.

S R Flip Flop

Here, two inputs S and R
Two outputs Q & \bar{Q}

Positive edge triggered state table

ck.	S	R	Q	\bar{Q}
0/1	X	X	NC	NC
↑	0	0	NC	NC
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	invalid	invalid

Characteristic equation:

$Q(t)$	SR	00	01	11	10
0		0	0	X	1
1		1	0	X	1

$R \cdot S = 0$

$Q(t+1) = R \cdot Q(t) + S$

Flip flop changes state after the rising edge of the clock.

Set State :-

As shown in state table, if the R input is at low level and S input at high level ($RS = 01$), during a low-to-high transition on clock pulse SR Flip flop is said to be in Set State and output of SR flip flop is set to 1 ($Q = 1$)

Reset State:

If the R input is at high level and S input at low level ($RS = 10$), during a low-to-high transition on clock pulse SR flip flop is said to be in Reset state and output of SR flip flop is reset to 0 ($Q = 0$)

Nochange state:

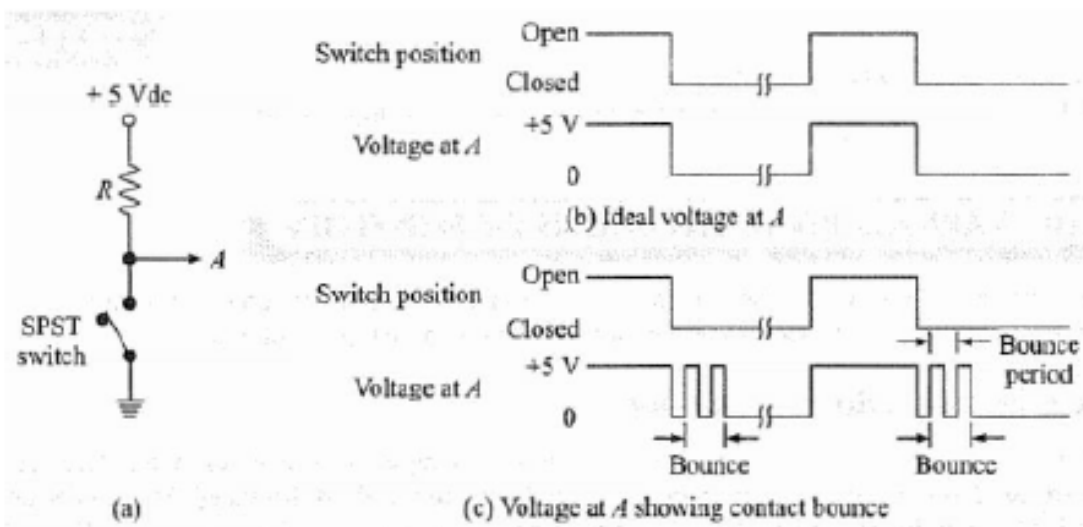
When both S & R equal to zero ($SR = 00$), then output remains in previous state.

Invalid State:

When both S & R equal to one ($SR = 11$), then output will be invalid.

7b). Explain switch contact bounce circuits. How can contact bounce be eliminated in mechanical switches?

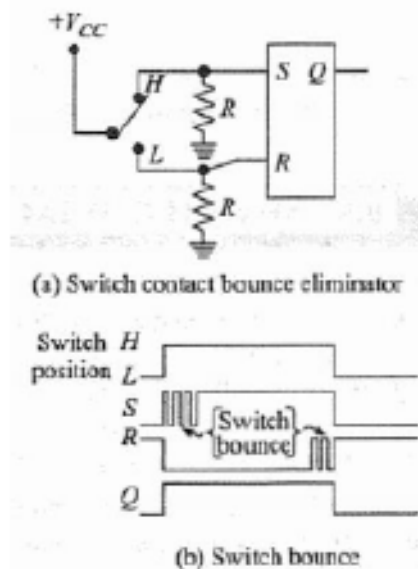
In nearly every digital system there will be occasion to use mechanical contacts for the purpose of conveying an electrical signal; examples of this are the switches used on the keyboard of a computer system. In each case, the intent is to apply a high logic level (usually +5 Vdc) or a low logic level (0 Vdc). The single-pole single-throw (SPST) switch shown in the figure (a) below is one such example. When the switch is open, the voltage at point A is +5 V dc; when the switch is closed, the voltage at point A is 0 V dc. Ideally, the voltage waveform at A should appear as shown in figure (b) as the switch is moved from open to closed, or vice versa.



In actuality, the waveform at point A will appear more or less as shown in figure (c), as the result of a phenomenon known as contact bounce. Any mechanical switching device consists of a moving contact arm restrained by some sort of a spring system. As a result, when the arm is moved from one stable position to the other, the arm bounces, much as a hard ball bounces when dropped on a hard surface. The number of bounces that occur and the period of the bounce differ for each switching device. Notice carefully that in this particular instance, even though actual physical contact bounce occurs each time the switch is opened or closed, contact bounce appears in the voltage level at point A only when the switch is closed.

If the voltage at point A is applied to the input of a TTL circuit, the circuit will respond properly when the switch is opened, since no contact bounce occurs. However, when the switch is closed, the circuit will respond as if multiple signals were applied, rather than the single-switch closure intended - the undesired result of mechanical contact bounce. There is a need here for some sort of electronic circuit to eliminate the contact bounce problem.

RS Latch Debounce Circuit



The RS latch in the figure above will remove any contact bounce due to the switch. The output (Q) is used to generate the desired switch signal. When the switch is moved to position H, $R = 0$ and $S = 1$. Bouncing occurs at the S input due to the switch. The flip-flop "sees" this as a series of high and low inputs, settling with a high level. The flip-flop will immediately be set with $Q = 1$ at the first high level on S. When the switch bounces, losing contact, the input signals are $R = S = 0$, therefore the flip-flop remains set ($Q = 1$). When the switch regains contact, $R = 0$ and $S = 1$; this causes an attempt to again set the flip-flop. But since the flip-flop is already set, no changes occur at Q. The result is that the flip-flop responds to the first, and only to the first, high level at its S input, resulting in a "clean" low-to-high signal at its output (Q).

When the switch is moved to position L, $S = 0$ and $R = 1$. Bouncing occurs at the R input due to the switch. Again, the flip-flop "sees" this as a series of high and low inputs. It simply responds to the first high level, and ignores all following transitions. The result is a "clean" high-to-low signal at the flip-flop output.

8. With necessary diagrams and proper explanation, derive the truth table, state transition diagram, characteristic equation and excitation table for an SR flip-flop and a JK flip-flop.

Characteristic Equations of Flip-flops

- Used for analyzing ckt's made of flip-flops.
- The next s/p, Q_{n+1} , is expressed as a function of present Q_n and i/p to the flip-flops.
- K-maps are used to get the optimized expression and the truth table of each FF is mapped into it.

• The states of sequential ckt's are represented using binary variables. For a sequential ckt with n state variables (where n is possibly large), the nos of possible states (2^n) is finite. Hence sequential ckt's are called as finite state machines.

• State transition diagram is a very convenient tool to describe a FSM. It is obtained from the truth table of a FF or its characteristic equation.

SR FF

• Each FF can be at either 0 state or 1 state defined by its stored value at any given time. Application of i/p may change the stored value, i.e., state of the FF. This is shown by a directional arrow and the corresponding i/p is written alongside it.

• The timing relation implicit in FF truth tables are brought to the forefront by the FSM concept of state transition diagram.

Flip-flop Excitation Table → It is like looking at the truth table in a reverse way.

→ Used in ckt synthesis or design problems.

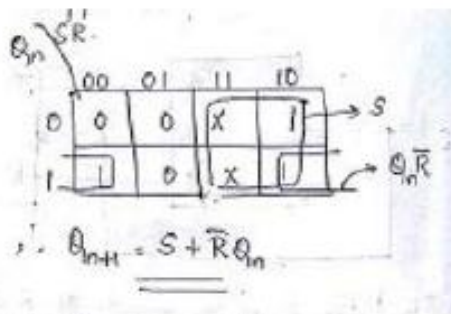
→ Excitation table shows the flip-flop i/p as a dependent function of transition $Q_n \rightarrow Q_{n+1}$, i.e., it shows what all i/ps must be applied to a FF in order to achieve a desired state transition from Q_n to Q_{n+1} .

→ It is obtained directly from state transition diagram. It can also be obtained from the characteristic table/eqn or truth table.

(a) SR flip-flop

Characteristic equation of SR FF

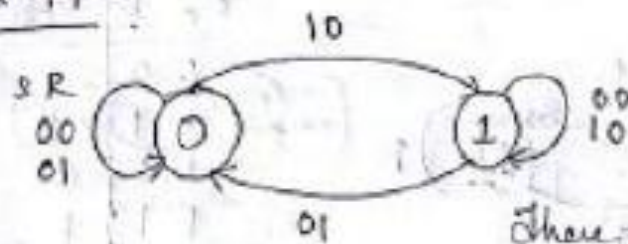
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	X
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	X



The logic equations are presented in SOP form by forming the largest group of 1s for each FF. For SR-FF, since $S=R=1$ i/p is not allowed, we put don't care cases (X) in the corresponding K-map locations, i.e., whether Q_{n+1} is 0 or 1, any i/p combination of $S=R=1$ will never arise.

State transition diagram of SR FF

SR-FF :



This is the state transition diagram for SR-FF

There are 2 possible states 0 & 1.

- While at 0, the state may have a next state of 0 for i/p $S=0, R=0$ (represented as 00) and $S=0, R=1$ (denote as 01). This is shown by the arrows starting from and ending at state 0.
- While at 1, the state may have a next state of 1 for i/p $S=0, R=0$ and for $S=1, R=0$. This is shown by the arrow starting from and ending at 1.
- While at ^{state} 0, the state may change to a next state of 1 for the i/p $S=1, R=0$ and while at state 1, the state may change to a next state of 0 for the i/p $S=0, R=1$. These are shown by the arrows from 0 to 1 and 1 to 0 respectively.
- The i/p of $S=R=1$ is not allowed in a SR-FF, hence not shown on the state transition diagram.

Excitation Table of SR FF

Characteristic table of SR-FF

S	R	Q_{n+1}
0	0	0 \rightarrow 0
		1 \rightarrow 1
0	1	0 \rightarrow 0
		1 \rightarrow 0
1	0	0 \rightarrow 1
		1 \rightarrow 1
1	1	0 \rightarrow ?
		1 \rightarrow ?

Excitation table of SR-FF

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

The excitation table can be explained as follows:

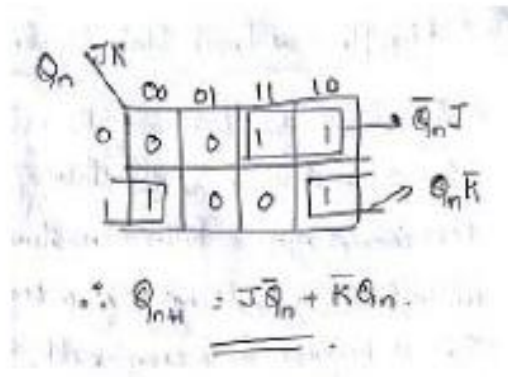
- \rightarrow If $Q_n = 0$, then on applying $S=0$ and $R=0$ or 1 (i.e., X), the Q_{n+1} becomes 0.
- \rightarrow If $Q_n = 0$, applying $S=1$ and $R=0$ makes $Q_{n+1} = 1$.
- \rightarrow If $Q_n = 1$, applying $S=0$ and $R=1$ makes $Q_{n+1} = 0$.
- \rightarrow If $Q_n = 1$, applying $S=0$ or 1 (i.e., X) and $R=0$ makes $Q_{n+1} = 1$.

Excitation tables can be obtained for the other FFs in a similar manner.

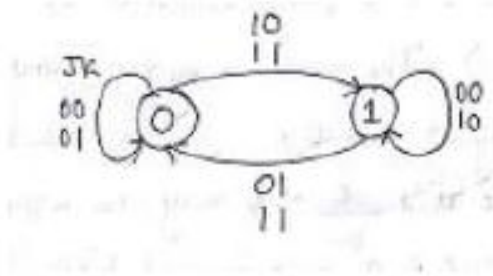
(b) JK flip-flop

Characteristic equation of JK FF

Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



State transition diagram of JK FF



Excitation Table of JK FF

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0