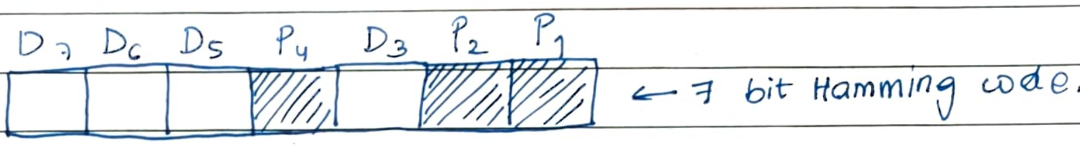
JUNAID·GIRKAR
60004190057
JAGirkar

16.12.2020

END SEM 3 EXAM
DIGITAL ELECTRONICS

Q3 Hamming code is basically a linear block code named after its inventor. It is an error correcting code.
It has parity bits that are inserted between the data as shown below. The 7-bit Hamming code is used commonly, but the concepts can be extended to any number of bits.

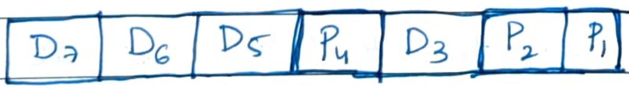


D ⟶ Data bits
P ⟶ Parity bits

Note that the parity bits are inserted at each 2^n bit where $n = 0, 1, 2, 3 \ldots$

7-BIT HAMMING CODE

A scientist named R.W Hamming developed a coding system which was easy to implement. Assuming that 4 data bits are to be transmitted, he suggested a code word pattern as shown below



$D_7, D_6, D_5, D_3 \rightarrow$ Data bits
$P_4, P_2, P_1 \rightarrow$ Parity bits

The D bits are data bits whereas P bits are parity bits.
The parity bits P_1, P_2, P_4 are adjusted in the particular way

MINIMUM NUMBER OF PARITY BITS.

The below table gives a listing of minimum number of parity bits needed for various ranges of "m" information bits

NO OF INFO BITS	NO OF PARITY BITS
2 to 4	3
5 to 11	4
12 to 26	5
27 to 57	6
58 to 120	7.

DECIDING THE VALUES OF PARITY BITS:

The below table indicates which bit positions are associated with each parity bit in order to establish required parity (even or odd) over the selected bits position.

PARITY BIT	BITS TO BE CHECKED
P_1	1,3,5,7,9,11,13,15...
P_2	2,3,6,7,10,11,14,15...
P_4	4,5,6,7,12,13,14,15...
P_8	8,9,10,11,12,13,14,15...

DECIDING THE PARITY BITS FOR A 7 BIT CODE:

selection of P_1: P_1 is adjusted to 0 or 1 so as to establish even parity over bits 1, 3, 5 and 7

7	6	5	4	3	2	1	→ consider bits 1,3,5,7 for P_1
7	6	5	4	3	2	1	→ consider bits 2,3,6,7 for P_2
7	6	5	4	3	2	1	→ consider bits 4,5,6,7 for P_4

JUNAID . GIRKAR
60004190057
JAGirkar

SELECTION OF P_2 :

P_2 is adjusted to 0 or 1 so as to set even parity over bits 2, 3, 6 and 7.

SELECTION OF P_4 :

P_4 is adjusted to 0 or 1 so as to set even parity over bits 4, 5, 6 and 7

Use the parity bits to find error bit by converting to decimal.

Received code :

7	6	5	4	3	2	1
1	0	0	0	0	1	0

Step 1 : check bits 4, 5, 6, 7 = 0001, odd parity hence error
$\therefore P_4 = 1$

Step 2 : check bits 2, 3, 6, 7 = 1001, even parity hence no error
$\therefore P_2 = 0$

Step 3 : check bits 1, 3, 5, 7 = 0001, odd parity hence error
$\therefore P_1 = 0$

$\therefore$ Error E = ~~1001~~

1	0	1
P_4	P_2	P_1

Derived equivalent of error is $(5)_{10}$. Hence fifth bit in the received code is incorrect. Hence invert the fifth ~~odd~~ bit to get the correct codeword.

$\therefore$ Correct code :

1	0	1	0	0	1	0
		↑				

5th bit inverted

JUNAID . GIRKAR
60004190057

Q4 $Y = (A + B + C' + D')(A' + C + D')(B' + C)(A + B')(B' + C')$

a) $Y = (A + B + C' + D')(A' + \cancel{C} + D' + B)(A' + \cancel{C} + D' + B')$
$(A + D' + B' + C)(A' + \cancel{D} + B' + C)(A' + \cancel{D'} + B' + C)$
$(A + \cancel{D'} + B' + C)(A + \cancel{D} + B' + C')(A' + \cancel{D} + B' + C')$
$(A + \cancel{D'} + B' + C')(A' + \cancel{D'} + B' + C')(A + \cancel{B} + C + D)$
$(A + \cancel{B'} + C' + D)(A + \cancel{B'} + C + D')(A + \cancel{B'} + C' + D')$

Simplifying.
$Y = (A + B + C' + D')(A' + B + C + D')(A' + B' + C + D')$
$(A + B' + C + D)(A' + B' + C + D)(A + B' + C + D')$
$(A + B' + C' + D)(A' + B' + C' + D)(A + B' + C' + D')$
$(A' + B' + C' + D')$

This expression is in standard POS form.

(b) 4-variable K-map.

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

$Y = (A + C' + D')(A' + C + D')(B'$)

JUNAID. GIRKAR
60004190057
JAGirkar

Q6

CONVERSION OF FLIP-FLOPS: causes one type of flip-flop to behave like another type of flip-flop.
In order to make one flip-flop mimic the behaviour of another certain addition circuitry and /or connections become necessary.

STEPS FOR CONVERSION:

STEP 1: Write the truth table of the Desired Flip-flop.

STEP 2: Obtain the Excitation Table for the given ~~flop~~ flip-flop from its truth table.

STEP 3: Append the excitation table of the given flip flop to the truth table of the desired flip-flop.

STEP 4: Simplify the expressions for the inputs of the given flip-flop using K-maps.

STEP 5: Design the necessary circuit.

JUNAID . GIRKAR
60004190057
JAGirkar

b) Conversion of SR flip-flop to JK flip-flop.

1- Truth table for JK flip flop.

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

2- Excitation table for SR flip flop.

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

JUNAID . GIRKAR
60004190057
[illegible]

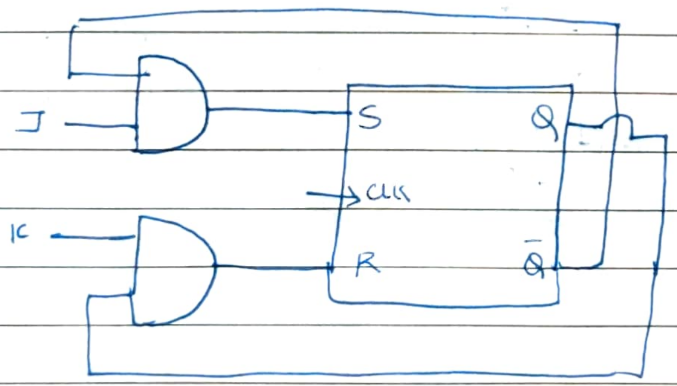
3. Conversion Table

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

4. K-MAP simplification.

J \ KQ_n	00	01	11	10
0	0	X	0	0
1	1	X	0	1

J \ KQ_n	00	01	11	10
0	X	0	1	X
1	0	0	1	0



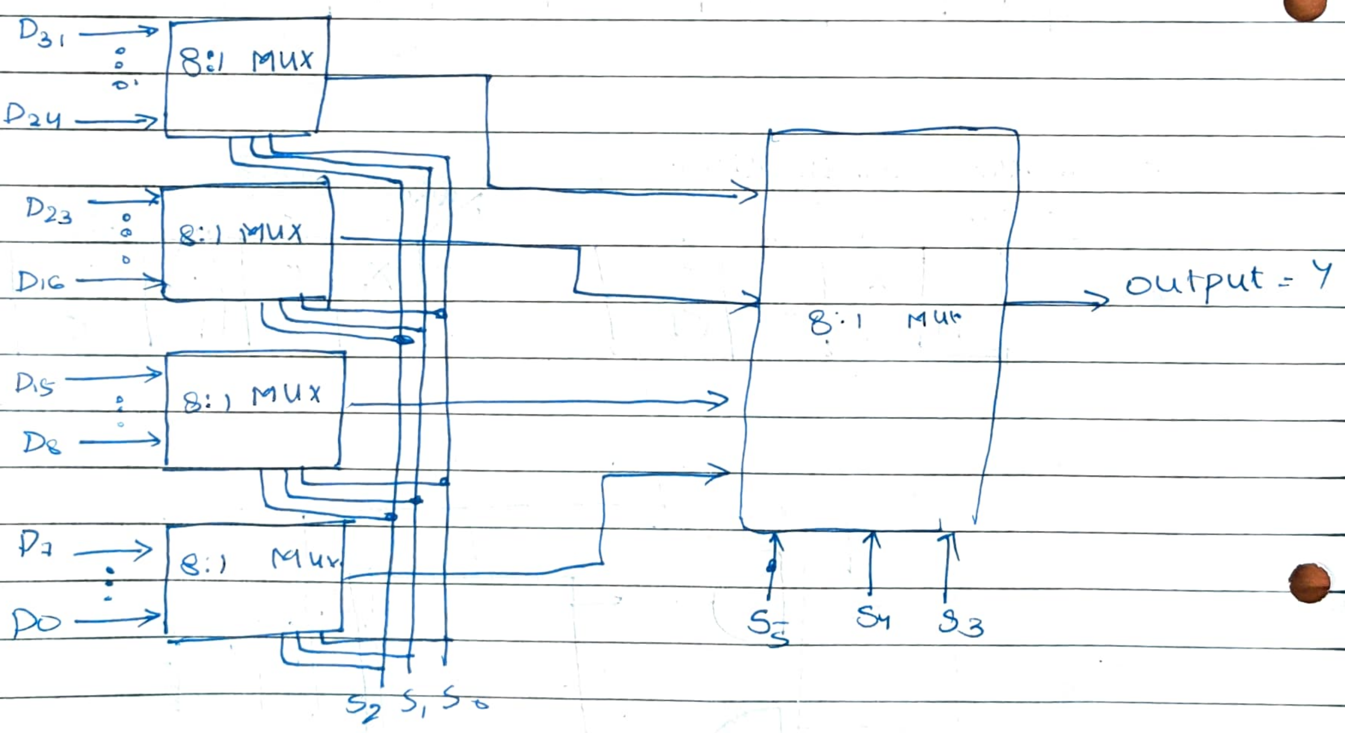
5 CIRCUIT DIAGRAM.

Q5

MULTIPLEXER TREES

A number of m-to-1 multiplexers can be arranged in a tree topology to obtain a bigger n-to-1 multiplexer is called a multiplexer tree where n > m.

Basically bigger multiplexers are obtained by combining smaller multiplexers.



5

JUNAID . GIRKAR
60004190057
JAGirkar

$F(A,B,C,D) = \Sigma m(0,1,3,6,7,9,10,13,15)$

Step 1: write the design table.

	D_0	D_1	D_2	D_3
$\bar{A}\bar{B}$	⓪	①	②	③
$\bar{A}B$	4	5	⑥	⑦
$A\bar{B}$	8	⑨	⑩	11
AB	12	⑬	14	⑮

$A\bar{B}$

$D_0 = \bar{A}\bar{B}$

$D_1 = \bar{A}\bar{B} + A\bar{B} + AB = \bar{B}(A+\bar{A}) + AB$

$= \bar{B} + AB \qquad (\because A+\bar{A} = 1)$

$= \bar{B} + A$

$D_2 = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$

$= \bar{A}(B+\bar{B}) + A\bar{B} \qquad (\because B+\bar{B} = 1)$

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

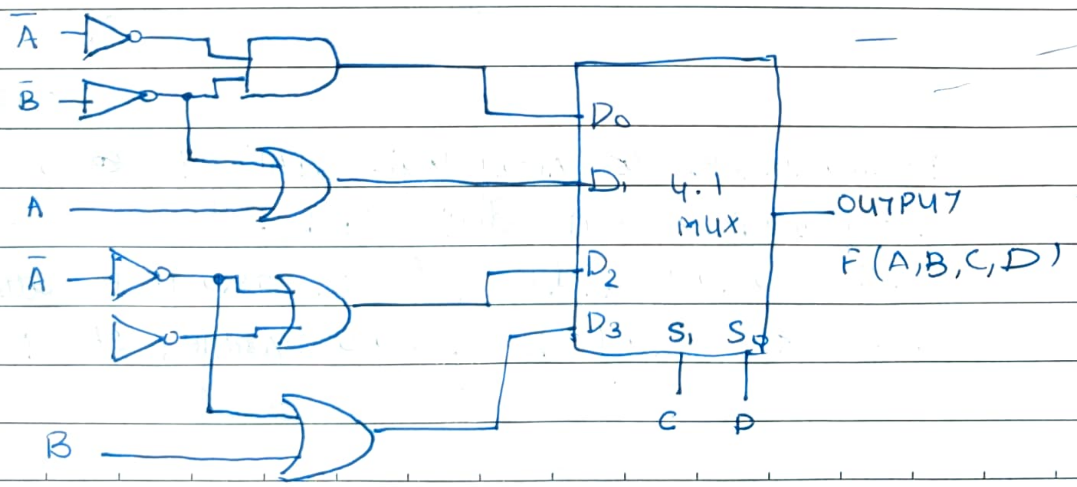
$D_3 = \bar{A}\bar{B} + \bar{A}B + AB$

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

$= \bar{A} + AB$

$= \bar{A} + B$

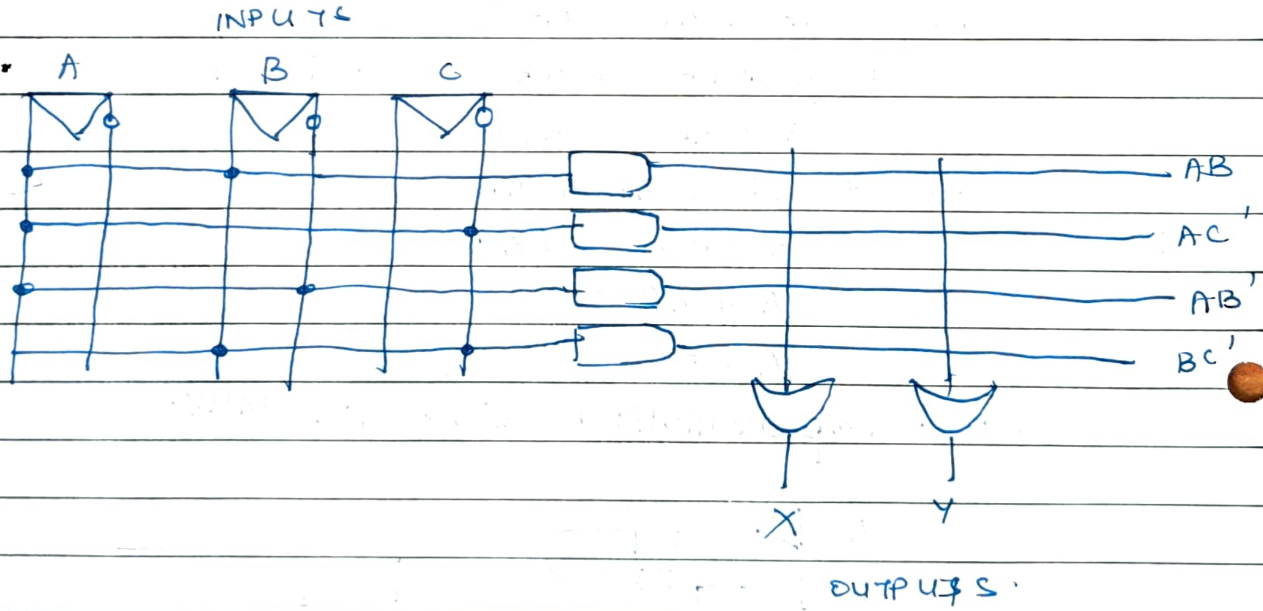
STEP 2: Implementation using 4:1 MUX.



Q e

$X = AB + AC'$
$Y = AB' + BC'$

The above given two boolean functions are in the form of SOP (sum of products)
The product terms present in the Boolean expressions are X & Y, and one product term that is AC' is common in every equation. So, the total required logic gates for generating the above 2 equations is AND gates ~~[illegible]~~ - 4 OR programmable gates - 2.
The equivalent PAL logic diagram is shown below



Programmable ~~log~~ Array Logic (PAL) is ~~a~~ a commonly used programming logic device (PLD). It has programmable (AND) array and fixed OR array. Because only the AND ~~go~~ array is programmable, it is easier to use

JUNAID. GH.
60004190055
JAGIRKAR

but not flexible as compared to programmable logic array (PLA).
PAL's limitation is number of AND gates

PAL consists of small programmable read only memory (PROM) and additional output logic used to implement a particular desired logic function with limited components