SLOT: G1+TG1

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Electronics Engineering

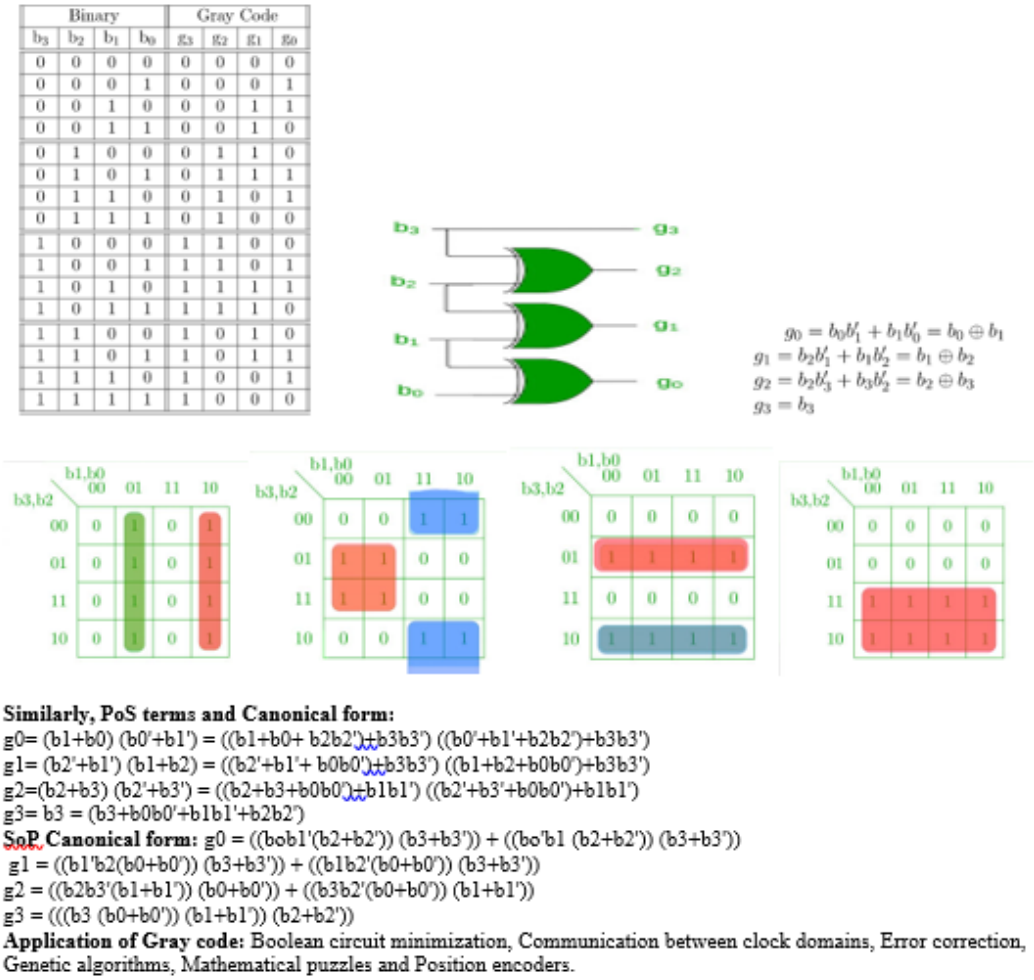
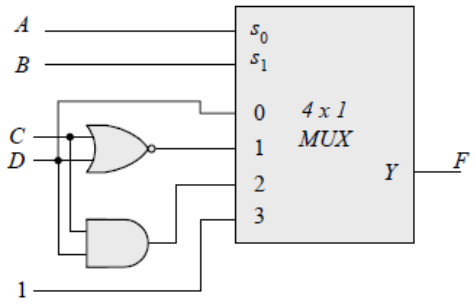
Fall Semester 2022-2023 Continuous Assessment Test – II

Programme Name & Branch : **BTECH CSE**
Course Name & code : **Digital Systems Design BECE102L**
Class Number(s): VL2022230104426; 4427; 4437; 4576; 4578; 4580; 4584; 4587; 4591; 4593
Exam Duration: 90 Min. Maximum Marks: 50

1. Use Proper pencil or black pen to draw all the circuits clearly and neatly.
2. Answer sequentially with mentioning the question number clearly outside the margin.

	Answer all the following Questions (5*10=50)			
S. No.	Question	Marks	Course Outcome (CO)	BL

1.	Check whether both the Verilog code generates the same circuit. If so, from the following Verilog code (i) write the standard SOP & POS form Boolean expression (ii) write the canonical SOP & POS form Boolean expression (iii) draw the circuit diagram for standard SOP expression (iv) draw truth table (v) identify the application of the circuit.
	// Code 1 //
	module identify (input [3:0] x, output [3:0] y);
	assign y[3] = x[3];
	assign y[2] = x[3] ^ x[2];
	assign y[1] = x[2] ^ x[1];
	assign y[0] = x[1] ^ x[0];
	endmodule
	// Code 2 //
	module identify (input [3:0] x, output [3:0] y);
	wire inv;
	not (inv, x[3]);
	not (y[3] , inv);
	xor xor1(y[2], x[3] ,x[2]);
	xor xor1(y[1], x[2] ,x[1]);
	xor xor1(y[0], x[1] ,x[0]);
	endmodule
	
	--
	Soln: Binary to Gray (b=x; g=y)
		10	CO3	1

	 **Similarly, PoS terms and Canonical form:** g0= (b1+b0) (b0'+b1') = ((b1+b0+ b2b2')+b3b3') ((b0'+b1'+b2b2')+b3b3') g1= (b2'+b1') (b1+b2) = ((b2'+b1'+ b0b0')+b3b3') ((b1+b2+b0b0')+b3b3') g2=(b2+b3) (b2'+b3') = ((b2+b3+b0b0')+b1b1') ((b2'+b3'+b0b0')+b1b1') g3= b3 = (b3+b0b0'+b1b1'+b2b2') **SoP Canonical form:** g0 = ((bob1'(b2+b2')) (b3+b3')) + ((bo'b1 (b2+b2')) (b3+b3')) g1 = ((b1'b2(b0+b0')) (b3+b3')) + ((b1b2'(b0+b0')) (b3+b3')) g2 = ((b2b3'(b1+b1')) (b0+b0')) + ((b3b2'(b0+b0')) (b1+b1')) g3 = (((b3 (b0+b0')) (b1+b1')) (b2+b2')) **Application of Gray code:** Boolean circuit minimization, Communication between clock domains, Error correction, Genetic algorithms, Mathematical puzzles and Position encoders.			
2.	Write the sum of minterms for the given circuit. Also write using K map, write SOP and POS Boolean expression for the circuit in canonical and standard form. Design the given circuit using 4X16 decoder constructed using 2x4 decoders (use only two input OR gates). Construct the internal circuit of 2x4 decoder using two input NAND logic.  --Soln $F_1(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15)$	10	CO3	1

Inputs ABCD	F	
0000	0	
0001	1	$AB = 00$
0010	0	$F = D$
0011	1	
0100	1	
0101	0	$AB = 01$
0110	0	$F = C'D'$
0111	0	$= (C + D)'$
1000	0	
1001	0	$AB = 10$
1010	0	$F = CD$
1011	1	
1100	1	
1101	1	$AB = 11$
1110	1	$F = 1$
1111	1	

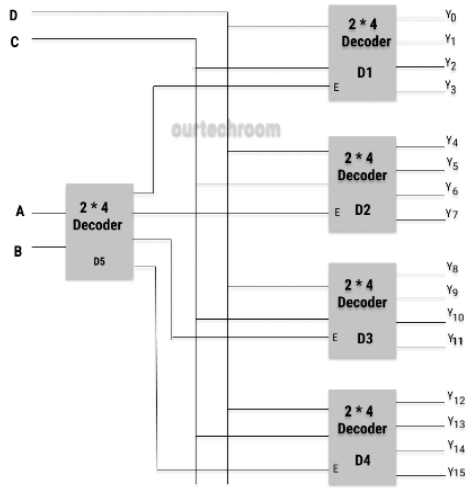
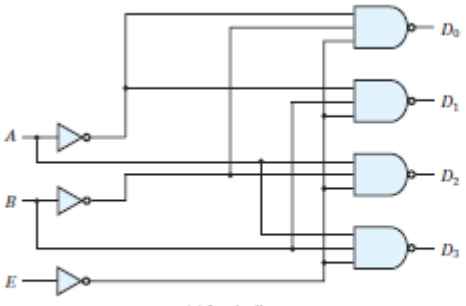


fig. Implementing 4*16 Decoder using 2*4 Decoder



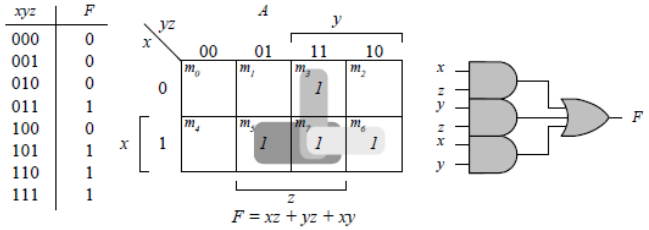
(a) Logic diagram

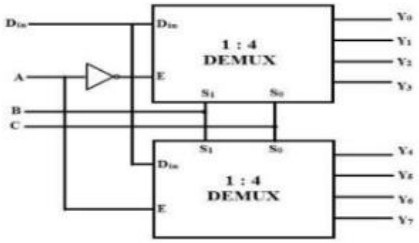
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	1

(b) Truth table

FIGURE 4.19
Two-to-four-line decoder with enable input

3.	Design a combinational circuit whose output (F) is equal to 1 if the three input variables have more 1's than 0's. The output is 0 otherwise. Assume three inputs, x, y, and z. (i) Implement the above circuit using logic gates and (ii) using 1x8 De-Multiplexer which is constructed using two 1X4 De-Multiplexers.	10	CO3	





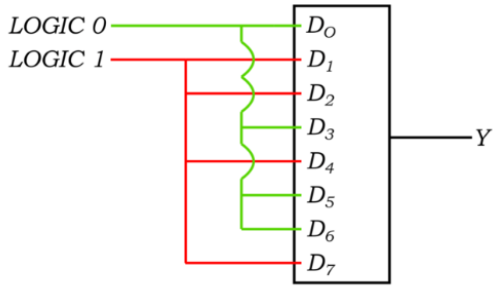
4.	Design Full adder and Full Subtractor using MUX. Write the mixed style Verilog code and test bench with line by line comments for Full subtractor implementation using MUX. **Implement full adder using multiplexer** **Truth table for full adder is shown below.**	10	CO3

A	B	C_{in}	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

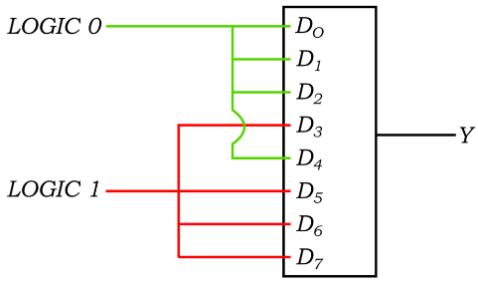
Full Adder using 8:1 mux

Now we will implement this full adder in 8:1 MUX. We will need to have two different multiplexers. The first one for implementing sum and the other one for implementing carry.

The implementation of Sum is shown in the circuit below



FOR SUM



FOR CARRY

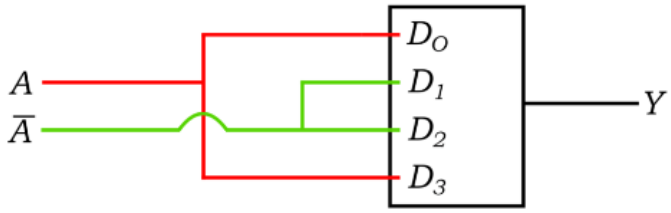
Full Adder using 4:1 mux

Since the 4:1 MUX has only four inputs and the truth table shows 8 inputs. The truth table shown above needs to be converted. Again we will do separate conversion for both sum and carry. Also there will be separate circuit diagrams.

The conversion for Sum is shown in the figure below

	D_0	D_1	D_2	D_3
$\overline{A}$	0	1	1	0
A	1	0	0	1
	A	$\overline{A}$	$\overline{A}$	A

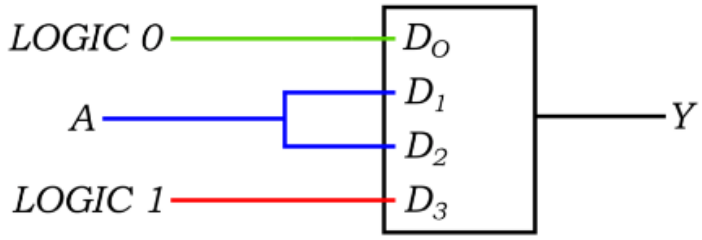
Implementation of Full adder (Sum) in 4:1 MUX is shown in the Circuit below



The conversion for Carry is shown in the figure below

	D_0	D_1	D_2	D_3
$\overline{A}$	0	0	0	1
A	0	1	1	1
	0	A	A	1

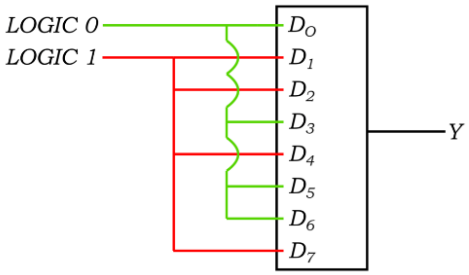
Implementation of Full adder (Carry) in 4:1 MUX is shown in the Circuit below



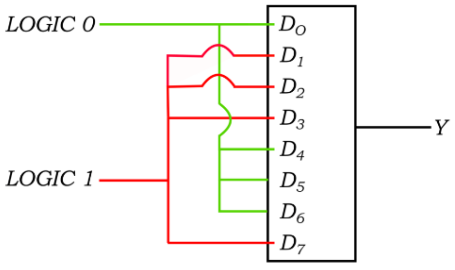
Full subtractor using multiplexer

A	B	C_{in}	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

The implementation of difference is shown in the circuit below



FOR DIFFERENCE



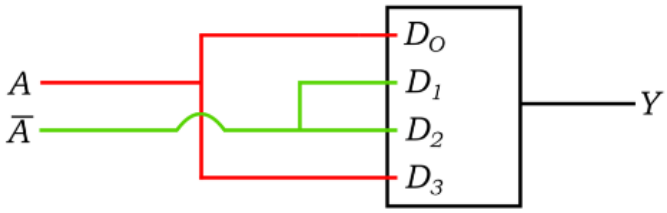
FOR BORROW

Full Subtractor using Multiplexer (4:1)

The conversion for difference is shown in the figure below

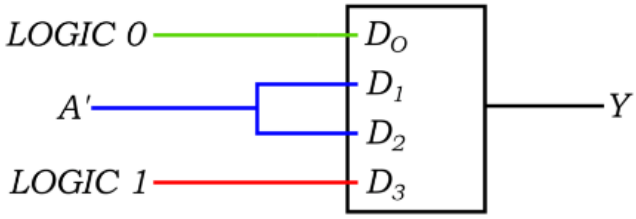
	D_O	D_1	D_2	D_3
$\overline{A}$	0	1	1	0
A	1	0	0	1
	A	$\overline{A}$	$\overline{A}$	A

Implementation of Full subtractor (Difference) in 4:1 MUX is shown in the Circuit below



The conversion for borrow is shown in the figure below

	D_O	D_1	D_2	D_3
$\overline{A}$	0	1	1	1
A	0	0	0	1
	0	A'	A'	1



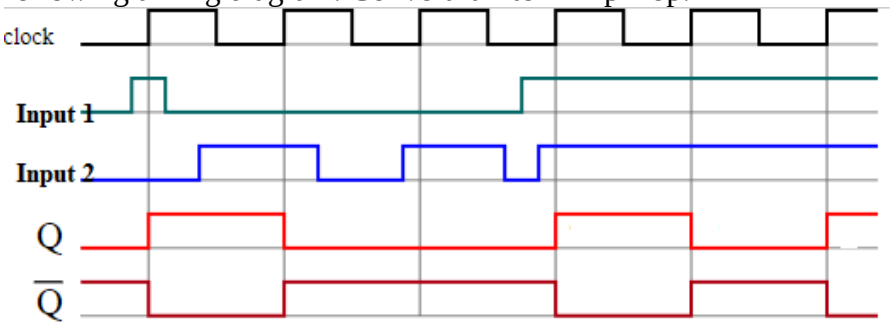
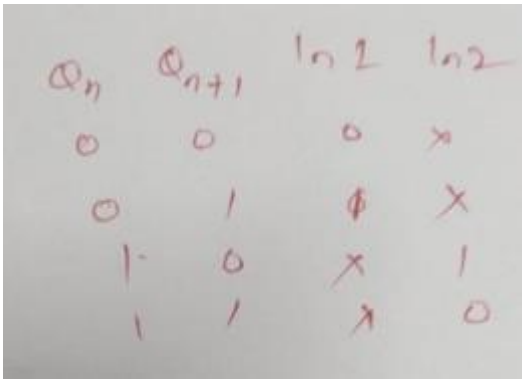
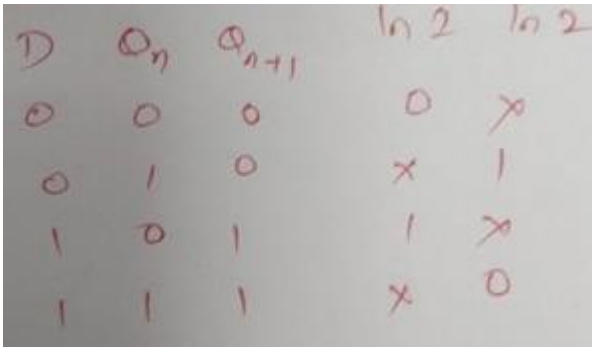
Verilog Code for Full Subtractor Using MUX

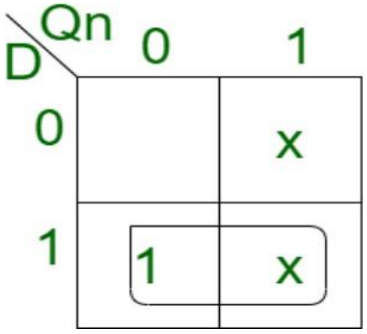
```
module Mulitplexer(d0,d1,d2,d3,d4,d5,d6,d7,sel,out);
input d0,d1,d2,d3,d4,d5,d6,d7;
input [2:0] sel;
output reg out;
always@(sel)
begin
case(sel)
3'b000:out=d0;
3'b001:out=d1;
3'b010:out=d2;
3'b011:out=d3;
3'b100:out=d4;
3'b101:out=d5;
3'b110:out=d6;
3'b111:out=d7;
```

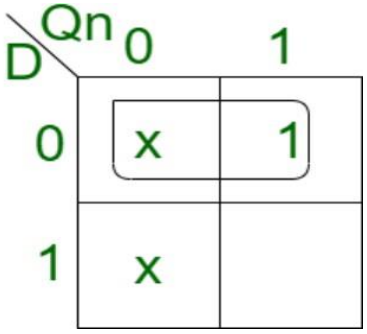
```
endcase
end
endmodule

module TestModule;
// Inputs
reg d0;
reg d1;
reg d2;
reg d3;
reg d4;
reg d5;
reg d6;
reg d7;
reg [2:0] sel;
// Outputs
wire out;
// Instantiate the Unit Under Test (UUT)
Mulitplexer uut
(.d0(d0),.d1(d1),.d2(d2),.d3(d3),.d4(d4),.d5(d5),.d6(d6),.d7(d7),.sel(sel),.out(out));
initial begin
// For Sum
d0 = 0;
d1 = 1;
d2 = 1;
d3 = 0;
d4 = 1;
d5 = 0;
d6 = 0;
d7 = 1;
 sel = 0;
#10 sel = 1;
#10 sel = 2;
#10 sel = 3;
#10 sel = 4;
#10 sel = 5;
#10 sel = 6;
#10 sel = 7;
#20;
// For Borrow
d0 = 0;
d1 = 1;
d2 = 1;
d3 = 1;
d4 = 0;
d5 = 0;
d6 = 0;
d7 = 1;

 sel = 0;
```

	#10 sel = 1; #10 sel = 2; #10 sel = 3; #10 sel = 4; #10 sel = 5; #10 sel = 6; #10 sel = 7; #20; end endmodule			
5.	Identify the flip-flop and write it's characteristic table, and equation from the following timing diagram. Convert it into D flip-flop.   	10	CO4	





$$J = D$$

$$K = D'$$