Sheet (3)

Q1-Using case statement , Write an HDL behavioral description of a eight-bit-arithmetic-logic unit (ALU).The circuit has a three-bit select bus (Sel),sixteen-bit input datapaths (A[7:0] and B[7:0]), an eight-bit output datapaths (y[7:0]),and performs the arithmetic and logic operations listed below.

Sel	Operation	Description
000	y=8'b0	
001	y=A&B	Bitwise and
010	y=A|B	Bitwise or
011	y=A^B	Bitwise exclusive or
100	y=~A	Bitwise complement
101	y=A-B	Subtract
110	y=A+B	Add (Assume A and B are unsigned)
111	y=8'hFF	

ANS

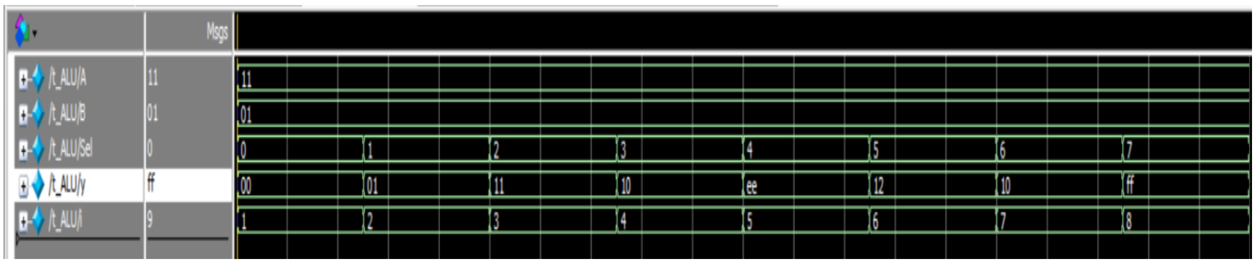
(a) Code:-

```
module ALU (output reg [7:0] y,input [7:0] A,B,input [2:0] Sel);
always @ (A,B,Sel)
case (Sel)
3'b000 : y=8'b0;
3'b001 : y=A&B;
3'b010 : y=A|B;
3'b011 : y=A^B;
3'b100 : y=~A;
3'b101 : y=A+B;
3'b110 : y=A-B;
3'b111 : y=8'hFF;
endcase
endmodule

`timescale 1ns/1ps

module t_ALU ();
reg [7:0] A,B;
reg [2:0] Sel;
wire [7:0] y;
integer i;
ALU U (y,A,B,Sel);
initial
begin
A=8'h11;B=8'h01;Sel=3'b0;
for (i=1;i<9;i=i+1)
#5 Sel=Sel+1;
end
endmodule
```

(b) Simulation:-



Q2-Develop and modify the eight-bit ALU specified in Q1.so that it has three-state output controlled by an enable input, En . Write a test bench and simulate the circuit.

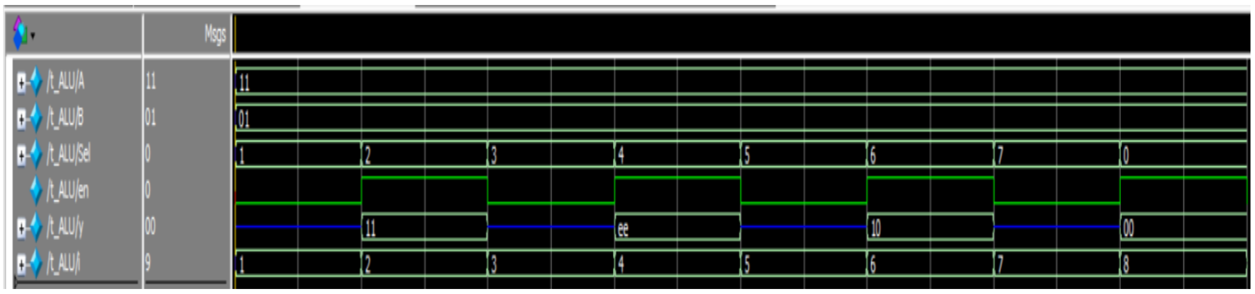
ANS

(a) Code:-

```
module ALU (output reg [7:0] y,input [7:0] A,B,input [2:0] Sel,input en);
always @ (A,B,Sel)
if (en==0)
y=8'hz;
else
case (Sel)
3'b000 : y=8'b0;
3'b001 : y=A&B;
3'b010 : y=A|B;
3'b011 : y=A^B;
3'b100 : y=~A;
3'b101 : y=A+B;
3'b110 : y=A-B;
3'b111 : y=8'hFF;
endcase
endmodule
`timescale 1ns/1ps

module t_ALU ();
reg [7:0] A,B;
reg [2:0] Sel;
reg en;
wire [7:0] y;
integer i;
ALU U (y,A,B,Sel,en);
initial
begin
A=8'h11;B=8'h01;Sel=3'b0;en=0;
for (i=1;i<9;i=i+1)
begin
Sel=Sel+1;
#20 en=~en;
end
end
endmodule
```

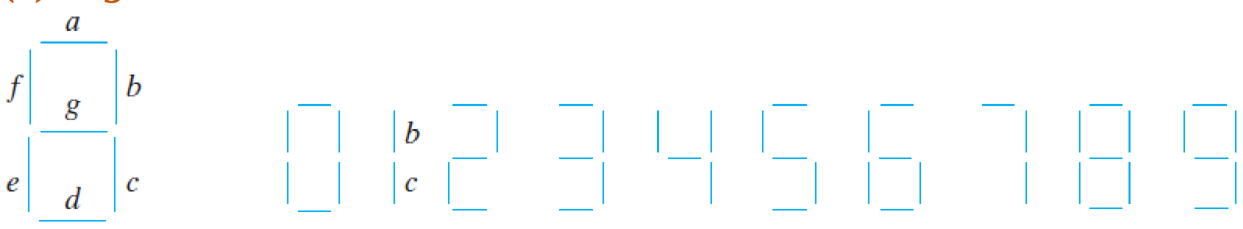
(b) Simulation:-



Q3-Develop and simulate a behavioral model of the ABCD-to-seven-segment decoder described in Fig

ANS

(a) Fig:-



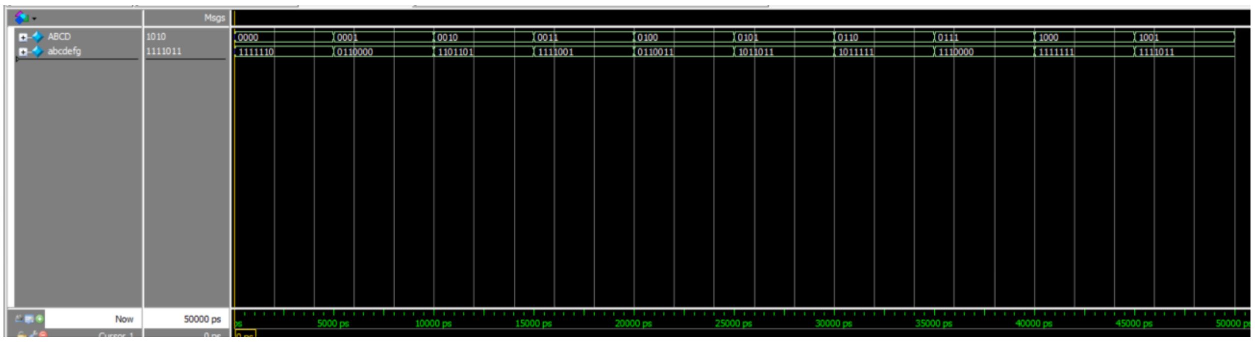
(b) Code:-

```
module ABCD_7_segment (output reg [6:0] abcdefg,input [3:0] ABCD);
always @ (ABCD)
case (ABCD)
0 : abcdefg=7'b1111110;
1 : abcdefg=7'b0110000;
2 : abcdefg=7'b1101101;
3 : abcdefg=7'b1111001;
4 : abcdefg=7'b0110011;
5 : abcdefg=7'b1011011;
6 : abcdefg=7'b1011111;
7 : abcdefg=7'b1110000;
8 : abcdefg=7'b1111111;
9 : abcdefg=7'b1111011;
endcase
endmodule
`timescale 1ns/1ps

module tb_7_segment ();
reg [3:0] ABCD;
wire [6:0] abcdefg;
ABCD_7_segment S (abcdefg,ABCD);
initial
begin
ABCD=0;
repeat (10)
#5 ABCD=ABCD+1;
end
```

endmodule

(c) Simulation:-



Q4-Using a continuous assignment , develop and simulate a dataflow model of

(a) The four-bit incrementer.

(b) The four-bit decrementer.

ANS

(a) Incrementer:

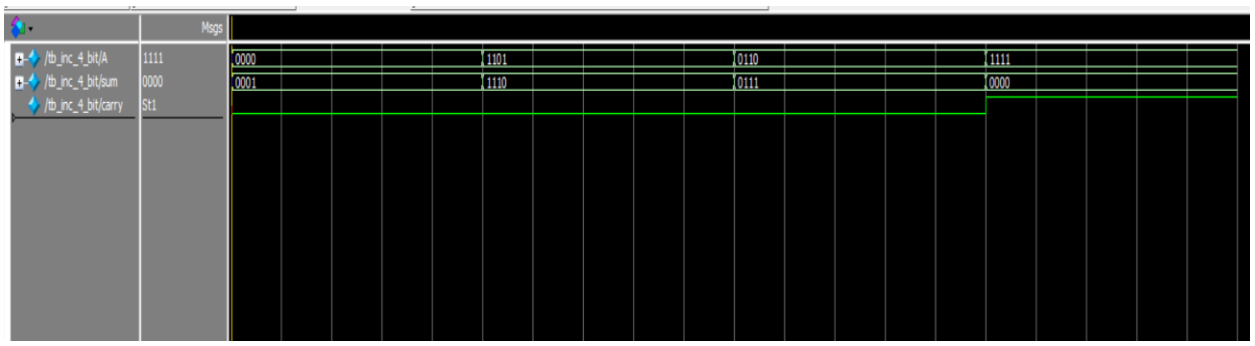
(a) Code:-

```
module inc_4_bit (output [3:0] sum,output carry,input [3:0] A);
assign {carry,sum}=A+1;
endmodule

`timescale 1ns/1ps

module tb_inc_4_bit ();
reg [3:0] A;
wire [3:0] sum;
wire carry;
inc_4_bit inc (sum,carry,A);
initial
begin
A=4'b0000;
#10 A=4'b1101;
#10 A=4'b0110;
#10 A=4'b1111;
end
endmodule
```

(b) Simulation:



(b) Decrementer:

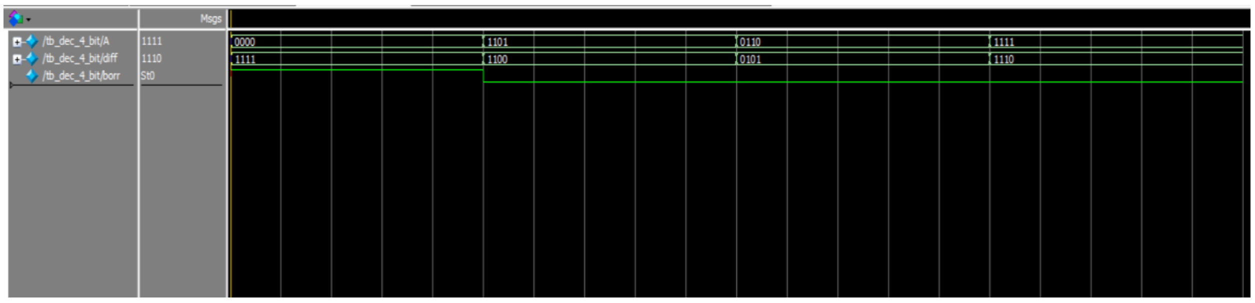
(a) Code:-

```
module dec_4_bit (output [3:0] diff,output borr,input [3:0] A);
assign {borr,diff}=A-1;
endmodule

`timescale 1ns/1ps

module tb_dec_4_bit ();
reg [3:0] A;
wire [3:0] diff;
wire borr;
dec_4_bit V (diff,borr,A);
initial
begin
A=4'b0000;
#10 A=4'b1101;
#10 A=4'b0110;
#10 A=4'b1111;
end
endmodule
```

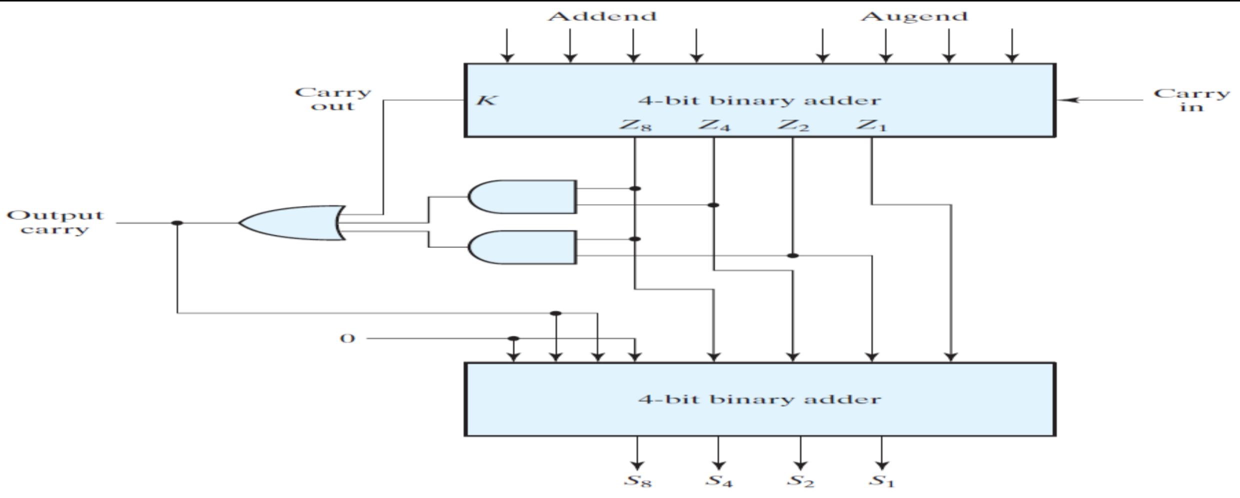
(b) Simulation:-



Q5-Develop and simulate a structural model of the decimal adder shown in Fig.

ANS

(a) Block diagram:-



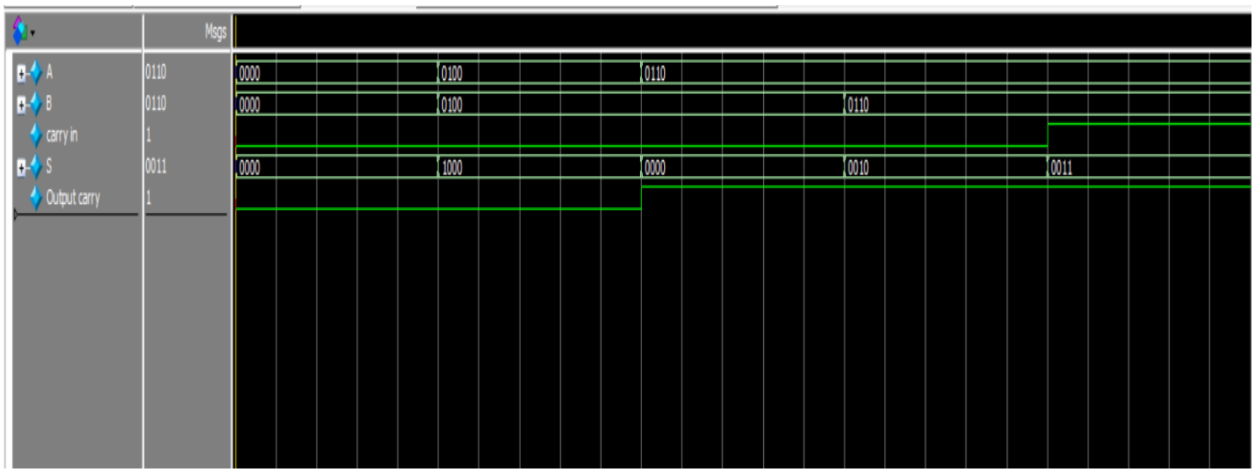
(b) Code:-

```
module decimal_adder (output [3:0] S,output Output_carry,input [3:0] A,B,input carry_in);
wire [3:0] S_Z,A_d;
wire carry_Z,And_1,And_2;
assign {carry_Z,S_Z}=A+B+carry_in;
assign And_1=S_Z[3]&S_Z[2];
assign And_2=S_Z[3]&S_Z[1];
assign Output_carry=And_1|And_2|carry_Z;
assign A_d={1'b0,Output_carry,Output_carry,1'b0};
assign {carry,S}=S_Z+A_d;
endmodule

`timescale 1ns/1ps

module tb_decimal_adder ();
reg [3:0] A,B;reg carry_in;
wire [3:0] S;wire Output_carry;
decimal_adder U (S,Output_carry,A,B,carry_in);
initial
begin
A=4'd0;B=4'd0;carry_in=0;
#10 A=4'd4;B=4'd4;
#10 A=4'd6;B=4'd4;
#10 A=4'd6;B=4'd6;
#10 carry_in=1;
end
endmodule
```

(c) Simulation:-



Msgs
A 0110
B 0110
carry in 1
S 0011
Output carry 1
0000
0100
0110
0000
0100
0110
0000
1000
0000
0010
0011