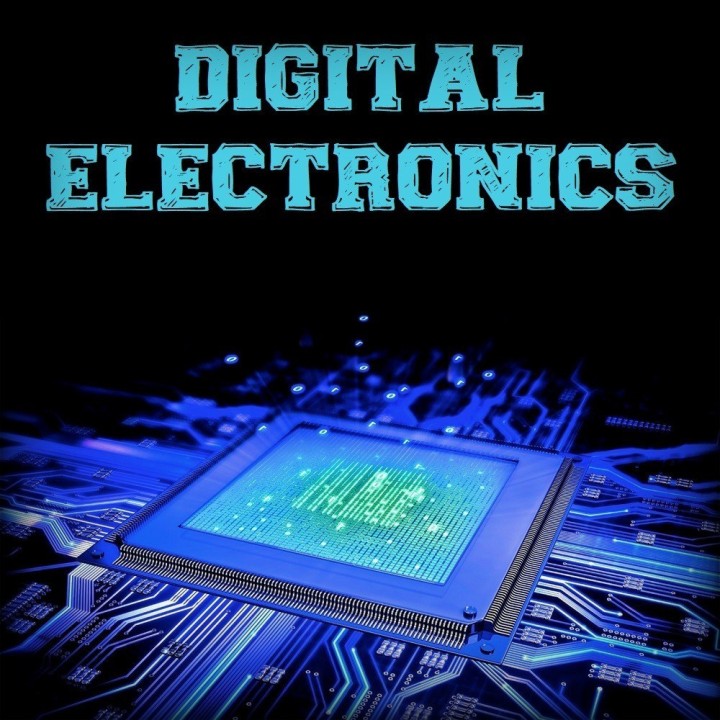
***Abanob Evram***

***Assignmen2***



A diagram of a circuit

Description automatically generated

**[Q1]**

**The design code:**

module Q1(D,A,B,C,Sel,Out,Out\_bar);

input [2:0] D;

input A,B,C,Sel ;

output Out,Out\_bar;

wire Z0,Z1,Z2;

assign Z0=D[0]&D[1];

assign Z1=Z0|D[2];

assign Z2=~(A^B^C);

assign Out=(Sel==1)?Z2:Z1;

assign Out\_bar=~(Out);

endmodule

**The testbench code:**

module Q1\_tb\_randomized();

reg [2:0] D\_tb;

reg A\_tb,B\_tb,C\_tb,Sel\_tb,Out\_excpected,Out\_bar\_excpected;

wire Out\_dut,Out\_bar\_dut;

Q1 Dut(D\_tb,A\_tb,B\_tb,C\_tb,Sel\_tb,Out\_dut,Out\_bar\_dut);

integer i ;

initial begin

for (i=0;i<99;i=i+1)begin

A\_tb=$random;

B\_tb=$random;

C\_tb=$random;

D\_tb=$random;

Sel\_tb=$random;

Out\_excpected=(Sel\_tb==1)?(~(A\_tb^B\_tb^C\_tb)):((D\_tb[0]&D\_tb[1])|(D\_tb[2]));

Out\_bar\_excpected =~(Out\_excpected);

#10

if(Out\_excpected!=Out\_dut)begin

$display("Error...");

$stop;

end

end

$stop;

end

initial begin

$monitor("Out\_excpected =%d , Out\_dut=%d",Out\_excpected,Out\_dut);

end

endmodule

A black screen with green and purple lines

Description automatically generated

A black screen with green lines

Description automatically generated

A white sheet with black text and numbers

Description automatically generated**[Q2]**

**The design code:**

module Priority\_encoder(X,Y);

input [3:0] X;

output reg [1:0] Y ;

always @(\*) begin

if (X[3]==1)

Y=3;

else if (X[2]==1)

Y=2;

else if (X[1]==1)

Y=1;

else

Y=0;

end

endmodule

**The testbench code:**

module Priority\_encoder\_tb();

reg [3:0] X\_tb;

reg [1:0] Y\_excpected;

wire [1:0] Y\_dut;

Priority\_encoder dut(X\_tb,Y\_dut);

integer i;

initial begin

for(i=0;i<99;i=i+1)begin

X\_tb=$random;

casex(X\_tb)

'b1xxx:Y\_excpected=3;

'b01xx:Y\_excpected=2;

'b001x:Y\_excpected=1;

'b000x:Y\_excpected=0;

endcase

#10

if(Y\_excpected!=Y\_dut)begin

$display("Error....");

$stop;

end

end

$stop;

end

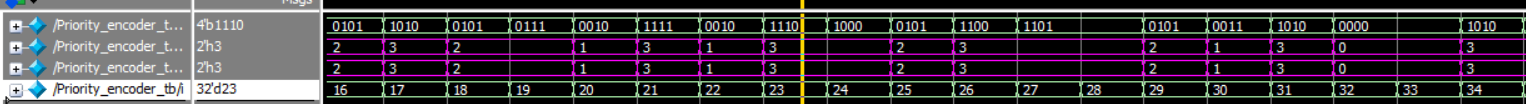
initial begin

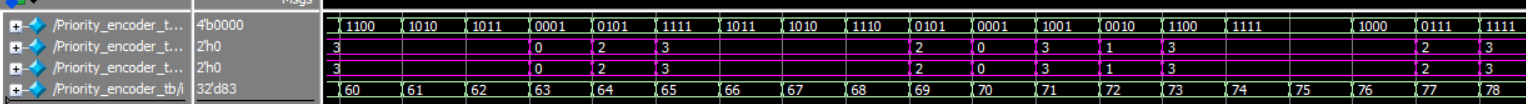
$monitor ("Y\_excpected=%d , Y\_dut=%d",Y\_excpected,Y\_dut);

end

endmodule

**Note: I do not need to use else or default because I wrote all cases**





**[Q3]**

**The design code:**

A screenshot of a computer

Description automatically generatedmodule BCD(D,Y);

input [9:0] D;

output reg [3:0] Y;

always @(\*) begin

case(D)

2:Y=1;

4:Y=2;

8:Y=3;

16:Y=4;

32:Y=5;

64:Y=6;

128:Y=7;

256:Y=8;

512:Y=9;

default: Y=0;

endcase

end

endmodule

**The testbench code:**

module BCD\_tb();

reg [9:0] D\_tb;

reg [3:0] Y\_expected;

wire [3:0] Y\_dut;

BCD dut(D\_tb,Y\_dut);

integer i;

initial begin

for(i=0;i<1000;i=i+1)begin

D\_tb=$random;

if (D\_tb==2)Y\_expected=1;

else if(D\_tb==4)Y\_expected=2;

else if(D\_tb==8)Y\_expected=3;

else if(D\_tb==16)Y\_expected=4;

else if(D\_tb==32)Y\_expected=5;

else if(D\_tb==64)Y\_expected=6;

else if(D\_tb==128)Y\_expected=7;

else if(D\_tb==256)Y\_expected=8;

else if(D\_tb==512)Y\_expected=9;

else Y\_expected=0;

#10

if (Y\_expected!=Y\_dut) begin

$display("ERROR...");

$stop;

end

end

$stop;

end

initial begin

$monitor("Y\_expected=%d , Y\_dut=%d , iteration=%d",Y\_expected,Y\_dut,i);

end

endmodule

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated**[Q4]**

**The design code:**

module N\_bit\_Adder(A,B,C);

parameter N = 1;

input [N-1:0] A,B;

output [N-1:0] C ;

assign C = A+B;

endmodule

**The testbench code:**

module N\_bit\_Adder\_tb();

parameter N\_tb =4 ;

reg [N\_tb-1:0] A\_tb,B\_tb,C\_expected;

wire[N\_tb-1:0] C\_dut;

N\_bit\_Adder #(N\_tb) dut(A\_tb,B\_tb,C\_dut);

integer i ;

initial begin

for(i=0;i<99;i=i+1)begin

A\_tb=$random;

B\_tb=$random;

C\_expected=A\_tb+B\_tb;

#10

if(C\_dut!=C\_expected)begin

$display("Errror...");

$stop;

end

end

$stop;

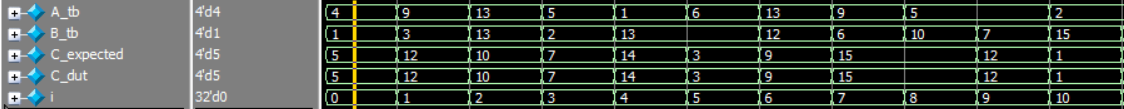
end

initial begin

$monitor("A\_tb=%d , B\_tb=%d , C\_expected=%d ,iteration=%d",A\_tb,B\_tb,C\_expected,i);

end

endmodule



A diagram of a diagram

Description automatically generated with medium confidence**[Q5]**

**The design code:**

module N\_bit\_Adder(A,B,C);

parameter N = 1;

input [N-1:0] A,B;

output [N-1:0] C ;

assign C = A+B;

endmodule

module N\_bit\_Alu(A,B,Opcode,Result);

parameter N = 4 ;

input [N-1:0] A,B;

input [1:0] Opcode ;

output reg [N-1:0] Result ;

wire [N-1:0] Adder\_result ;

N\_bit\_Adder #(N) m1(A,B,Adder\_result);

always @(\*) begin

if (Opcode==0)

Result=Adder\_result;

else if (Opcode==1)

Result=A|B;

else if (Opcode==2)

Result=A-B;

else if (Opcode==3)

Result=A^B;

end

endmodule

**The testbench code:**

module N\_bit\_Alu\_tb();

parameter N\_tb =4;

reg [N\_tb-1:0] A\_tb,B\_tb,Result\_expected;

reg [1:0] Opcode\_tb;

wire [N\_tb-1:0] Result\_dut;

N\_bit\_Alu #(N\_tb) dut(A\_tb,B\_tb,Opcode\_tb,Result\_dut);

integer i ;

initial begin

for(i=0;i<99;i=i+1)begin

A\_tb=$random;

B\_tb=$random;

Opcode\_tb=$random;

case(Opcode\_tb)

0:Result\_expected=A\_tb+B\_tb;

1:Result\_expected=A\_tb|B\_tb;

2:Result\_expected=A\_tb-B\_tb;

3:Result\_expected=A\_tb^B\_tb;

endcase

#10

if(Result\_expected!=Result\_dut)begin

$display ("Errror...");

$stop;

end

end

$stop;

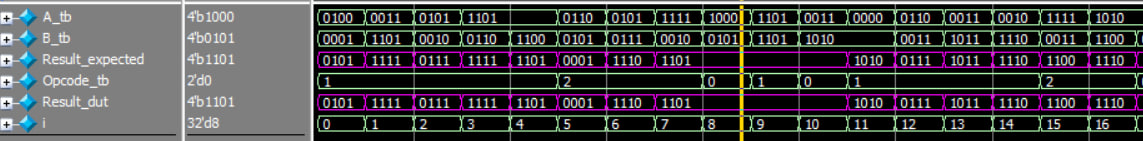
end

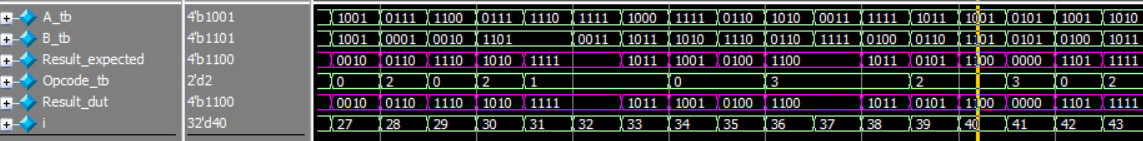
initial begin

$monitor("A\_tb=%d , B\_tb=%d , Opcode\_tb=%d , Result\_expected=%d",A\_tb,B\_tb,Opcode\_tb,Result\_expected);

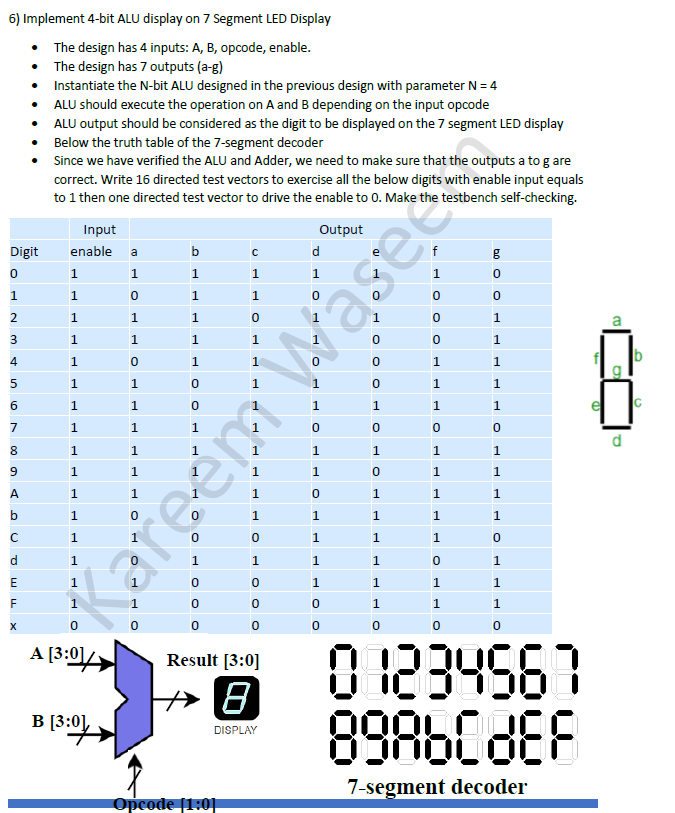
end

endmodule





**[Q6]**



**The design code:**

module N\_bit\_Adder(A,B,C);

parameter N = 1;

input [N-1:0] A,B;

output [N-1:0] C ;

assign C = A+B;

endmodule

module N\_bit\_Alu(A,B,Opcode,Result);

parameter N = 4 ;

input [N-1:0] A,B;

input [1:0] Opcode ;

output reg [N-1:0] Result ;

wire [N-1:0] Adder\_result ;

N\_bit\_Adder #(N) m1(A,B,Adder\_result);

always @(\*) begin

if (Opcode==0)

Result=Adder\_result;

else if (Opcode==1)

Result=A|B;

else if (Opcode==2)

Result=A-B;

else if (Opcode==3)

Result=A^B;

end

endmodule

module SSD(A,B,Opcode,Enable,a,b,c,d,e,f,g);

parameter N\_SSD = 4;

input [N\_SSD-1:0] A,B;

input [1:0] Opcode;

input Enable ;

output reg a,b,c,d,e,f,g;

wire [N\_SSD-1:0] Result\_alu;

N\_bit\_Alu #(N\_SSD) Alu(A,B,Opcode,Result\_alu);

always @(\*) begin

case({Result\_alu,Enable})

5'b00001:{a,b,c,d,e,f,g}=7'b1111110;//0

5'b00011:{a,b,c,d,e,f,g}=7'b0110000;//1

5'b00101:{a,b,c,d,e,f,g}=7'b1101101;//2

5'b00111:{a,b,c,d,e,f,g}=7'b1111001;//3

5'b01001:{a,b,c,d,e,f,g}=7'b0110011;//4

5'b01011:{a,b,c,d,e,f,g}=7'b1011011;//5

5'b01101:{a,b,c,d,e,f,g}=7'b1011111;//6

5'b01111:{a,b,c,d,e,f,g}=7'b1110000;//7

5'b10001:{a,b,c,d,e,f,g}=7'b1111111;//8

5'b10011:{a,b,c,d,e,f,g}=7'b1111011;//9

5'b10101:{a,b,c,d,e,f,g}=7'b1110111;//a

5'b10111:{a,b,c,d,e,f,g}=7'b0011111;//b

5'b11001:{a,b,c,d,e,f,g}=7'b1001110;//c

5'b11011:{a,b,c,d,e,f,g}=7'b0111101;//d

5'b11101:{a,b,c,d,e,f,g}=7'b1001111;//e

5'b11111:{a,b,c,d,e,f,g}=7'b1000111;//f

default : {a,b,c,d,e,f,g}=0;

endcase

end

endmodule

**The testbench code:**

module SSD\_tb();

parameter N\_SSD\_tb = 4;

reg [N\_SSD\_tb-1:0] A\_tb,B\_tb;

reg [1:0] Opcode\_tb;

reg Enable\_tb,a,b,c,d,e,f,g; //a:g expected

wire a\_dut,b\_dut,c\_dut,d\_dut,e\_dut,f\_dut,g\_dut;

SSD #(N\_SSD\_tb) S0(A\_tb,B\_tb,Opcode\_tb,Enable\_tb,

a\_dut,b\_dut,c\_dut,d\_dut,e\_dut,f\_dut,g\_dut);

**//Note OP=0:ADD,OP=1:OR,OP=2:SUB,OP=3:XOR**

initial begin

#0 A\_tb=10;B\_tb=10;Opcode\_tb=2;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1111110;//0

#10 A\_tb=1;B\_tb=0;Opcode\_tb=1;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b0110000;//1

#10 A\_tb=10;B\_tb=8;Opcode\_tb=3;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1101101;//2

#10 A\_tb=1;B\_tb=2;Opcode\_tb=0;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1111001;//3

#10 A\_tb=2;B\_tb=2;Opcode\_tb=0;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b0110011;//4

#10 A\_tb=7;B\_tb=2;Opcode\_tb=2;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1011011;//5

#10 A\_tb=8;B\_tb=2;Opcode\_tb=2;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1011111;//6

#10 A\_tb=5;B\_tb=2;Opcode\_tb=1;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1110000;//7

#10 A\_tb=15;B\_tb=7;Opcode\_tb=3;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1111111;//8

#10 A\_tb=4;B\_tb=5;Opcode\_tb=0;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1111011;//9

#10 A\_tb=5;B\_tb=5;Opcode\_tb=0;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1110111;//a

#10 A\_tb=12;B\_tb=1;Opcode\_tb=2;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b0011111;//b

#10 A\_tb=15;B\_tb=3;Opcode\_tb=2;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1001110;//c

#10 A\_tb=5;B\_tb=8;Opcode\_tb=1;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b0111101;//d

#10 A\_tb=7;B\_tb=7;Opcode\_tb=0;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1001111;//e

#10 A\_tb=15;B\_tb=0;Opcode\_tb=2;Enable\_tb=1;{a,b,c,d,e,f,g}=7'b1000111;//f

#10 A\_tb=5;B\_tb=5;Opcode\_tb=0;Enable\_tb=0;{a,b,c,d,e,f,g}=0;

#10

if({a\_dut,b\_dut,c\_dut,d\_dut,e\_dut,f\_dut,g\_dut}!={a,b,c,d,e,f,g})begin

$display("Errrrror...");

$stop;

end

$stop;

end

initial begin

$monitor("A\_tb=%d,B\_tb=%d,Opcode\_tb=%d,Enable\_tb=%d,{a,b,c,d,e,f,g}=%b",

A\_tb,B\_tb,Opcode\_tb,Enable\_tb,{a,b,c,d,e,f,g});

end

endmodule

A screen shot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated