University of Tehran
College of Engineering
School of Electrical and Computer Engineering

Digital Systems 1

Dr.Navabi

Homework 6

Soroush Mesforush Mashhad

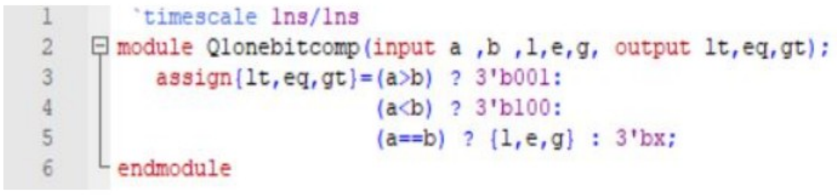
SN:810198472

Ordibehesht 00

Question 1

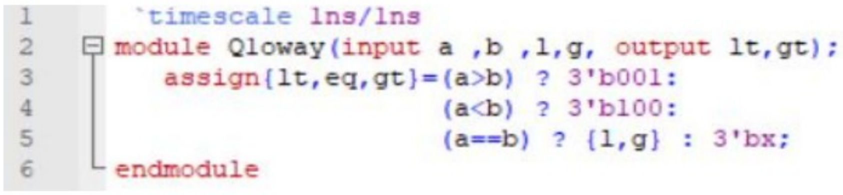
A

The system verilog code for a 1-bit comparator is as follows:



```
 `timescale 1ns/1ns
module Q1onebitcomp(input a ,b ,l,e,g, output lt,eq,gt);
   assign{lt,eq,gt}=(a>b)  ? 3'b001:
                     (a<b)  ? 3'b100:
                     (a==b)  ? {l,e,g} : 3'bx;
endmodule
```

Figure 1: 1-bit comparator

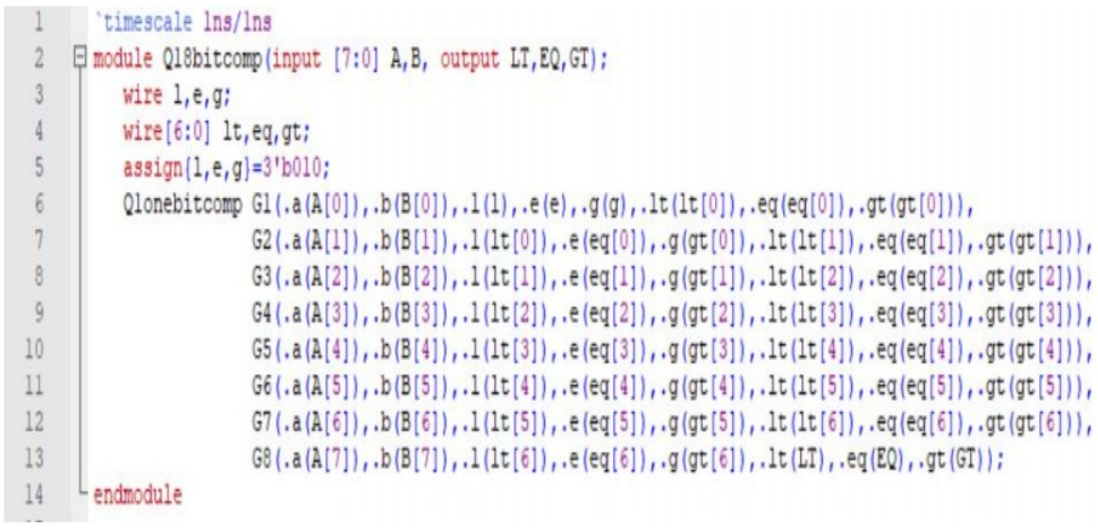


```
 `timescale 1ns/1ns
module Q1oway(input a ,b ,l,g, output lt,gt);
   assign{lt,eq,gt}=(a>b)  ? 3'b001:
                     (a<b)  ? 3'b100:
                     (a==b)  ? {l,g} : 3'bx;
endmodule
```

Figure 2: 1-bit comparator

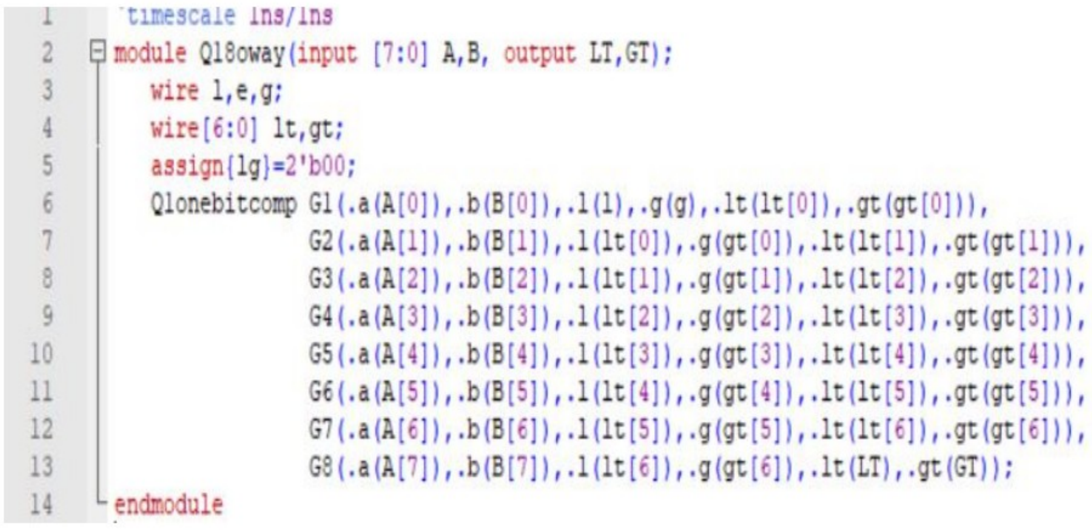
B

The system verilog code for a 8-bit comparator is as follows:



```verilog
`timescale 1ns/1ns
module Q18bitcomp(input [7:0] A,B, output LT,EQ,GT);
   wire l,e,g;
   wire[6:0] lt,eq,gt;
   assign{l,e,g}=3'b010;
   Q1onebitcomp G1(.a(A[0]),.b(B[0]),.l(l),.e(e),.g(g),.lt(lt[0]),.eq(eq[0]),.gt(gt[0])),
                G2(.a(A[1]),.b(B[1]),.l(lt[0]),.e(eq[0]),.g(gt[0]),.lt(lt[1]),.eq(eq[1]),.gt(gt[1])),
                G3(.a(A[2]),.b(B[2]),.l(lt[1]),.e(eq[1]),.g(gt[1]),.lt(lt[2]),.eq(eq[2]),.gt(gt[2])),
                G4(.a(A[3]),.b(B[3]),.l(lt[2]),.e(eq[2]),.g(gt[2]),.lt(lt[3]),.eq(eq[3]),.gt(gt[3])),
                G5(.a(A[4]),.b(B[4]),.l(lt[3]),.e(eq[3]),.g(gt[3]),.lt(lt[4]),.eq(eq[4]),.gt(gt[4])),
                G6(.a(A[5]),.b(B[5]),.l(lt[4]),.e(eq[4]),.g(gt[4]),.lt(lt[5]),.eq(eq[5]),.gt(gt[5])),
                G7(.a(A[6]),.b(B[6]),.l(lt[5]),.e(eq[5]),.g(gt[5]),.lt(lt[6]),.eq(eq[6]),.gt(gt[6])),
                G8(.a(A[7]),.b(B[7]),.l(lt[6]),.e(eq[6]),.g(gt[6]),.lt(LT),.eq(EQ),.gt(GT));
endmodule
```

Figure 3: 8-bit comparator



```verilog
`timescale 1ns/1ns
module Q18oway(input [7:0] A,B, output LT,GT);
   wire l,e,g;
   wire[6:0] lt,gt;
   assign{lg}=2'b00;
   Q1onebitcomp G1(.a(A[0]),.b(B[0]),.l(l),.g(g),.lt(lt[0]),.gt(gt[0])),
                G2(.a(A[1]),.b(B[1]),.l(lt[0]),.g(gt[0]),.lt(lt[1]),.gt(gt[1])),
                G3(.a(A[2]),.b(B[2]),.l(lt[1]),.g(gt[1]),.lt(lt[2]),.gt(gt[2])),
                G4(.a(A[3]),.b(B[3]),.l(lt[2]),.g(gt[2]),.lt(lt[3]),.gt(gt[3])),
                G5(.a(A[4]),.b(B[4]),.l(lt[3]),.g(gt[3]),.lt(lt[4]),.gt(gt[4])),
                G6(.a(A[5]),.b(B[5]),.l(lt[4]),.g(gt[4]),.lt(lt[5]),.gt(gt[5])),
                G7(.a(A[6]),.b(B[6]),.l(lt[5]),.g(gt[5]),.lt(lt[6]),.gt(gt[6])),
                G8(.a(A[7]),.b(B[7]),.l(lt[6]),.g(gt[6]),.lt(LT),.gt(GT));
endmodule
```

Figure 4: 8-bit comparator

Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW6	Due Date: 00 / 2 / 12	$\frac{1}{6}$

2- First I draw the truth table of the 4 to 2 Encoder then by using K-maps we Shall design the Circuit handsomely.

S_3	S_2	S_1	S_0	N_1	N_0	A_S	EO
1	–	–	–	1	1	0	1
0	1	–	–	1	0	0	1
0	0	1	–	0	1	0	1
0	0	0	1	0	0	0	1
0	0	0	0	0	0	ϕ	0

(The circuit only works if Ei = 0)

So if Ei = 1 we deactivate N_0, N_1 by turning them to 0.

N_0

S_1S_0 \ S_3S_2	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	0	1	1
10	1	0	1	1

$\downarrow$

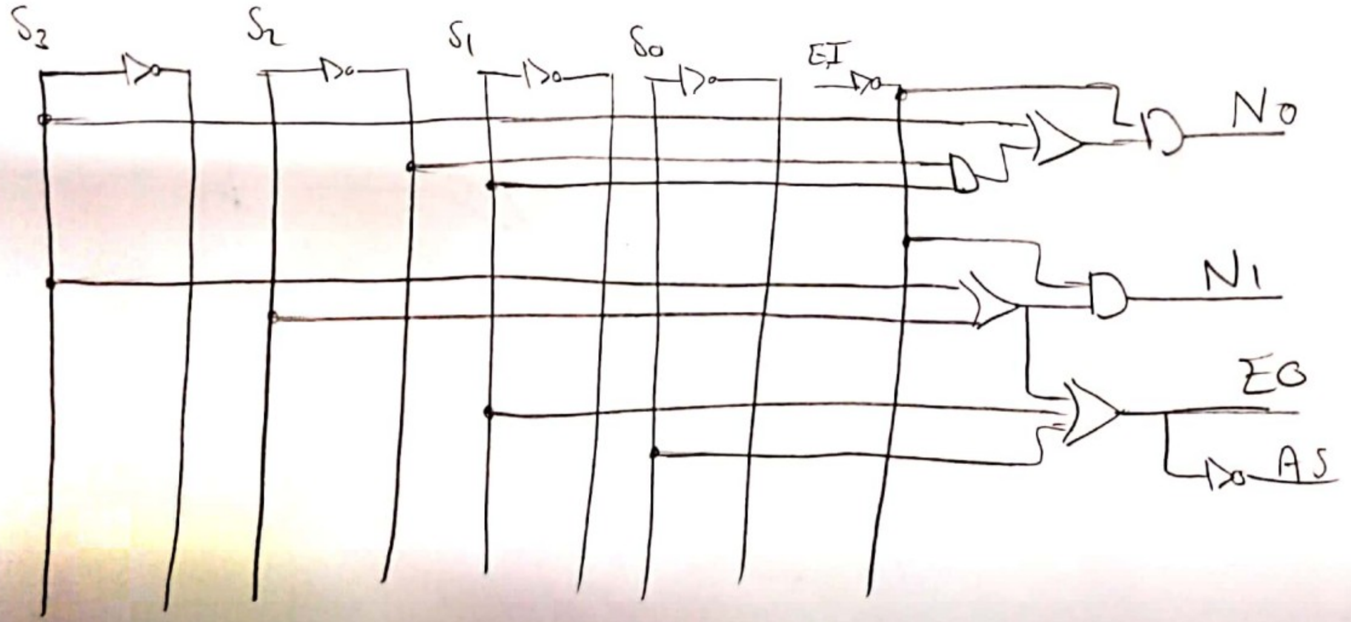
$S_3 + \bar{S_2} S_1$

N_1

S_1S_0 \ S_3S_2	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

$\downarrow$

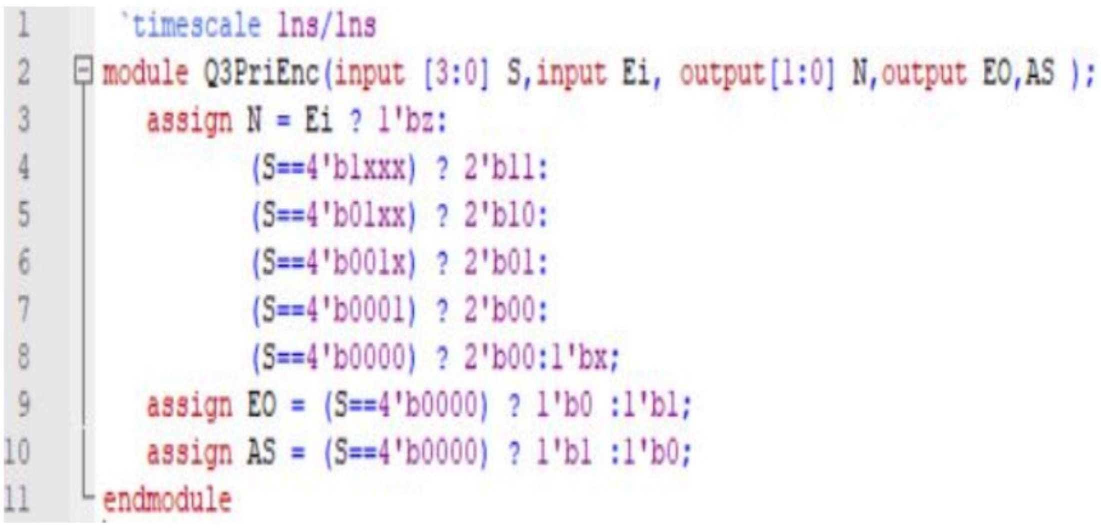
$S_3 + S_2$



Scanned by CamScanner

Question 3

The system verilog code of the encoder is as follows:

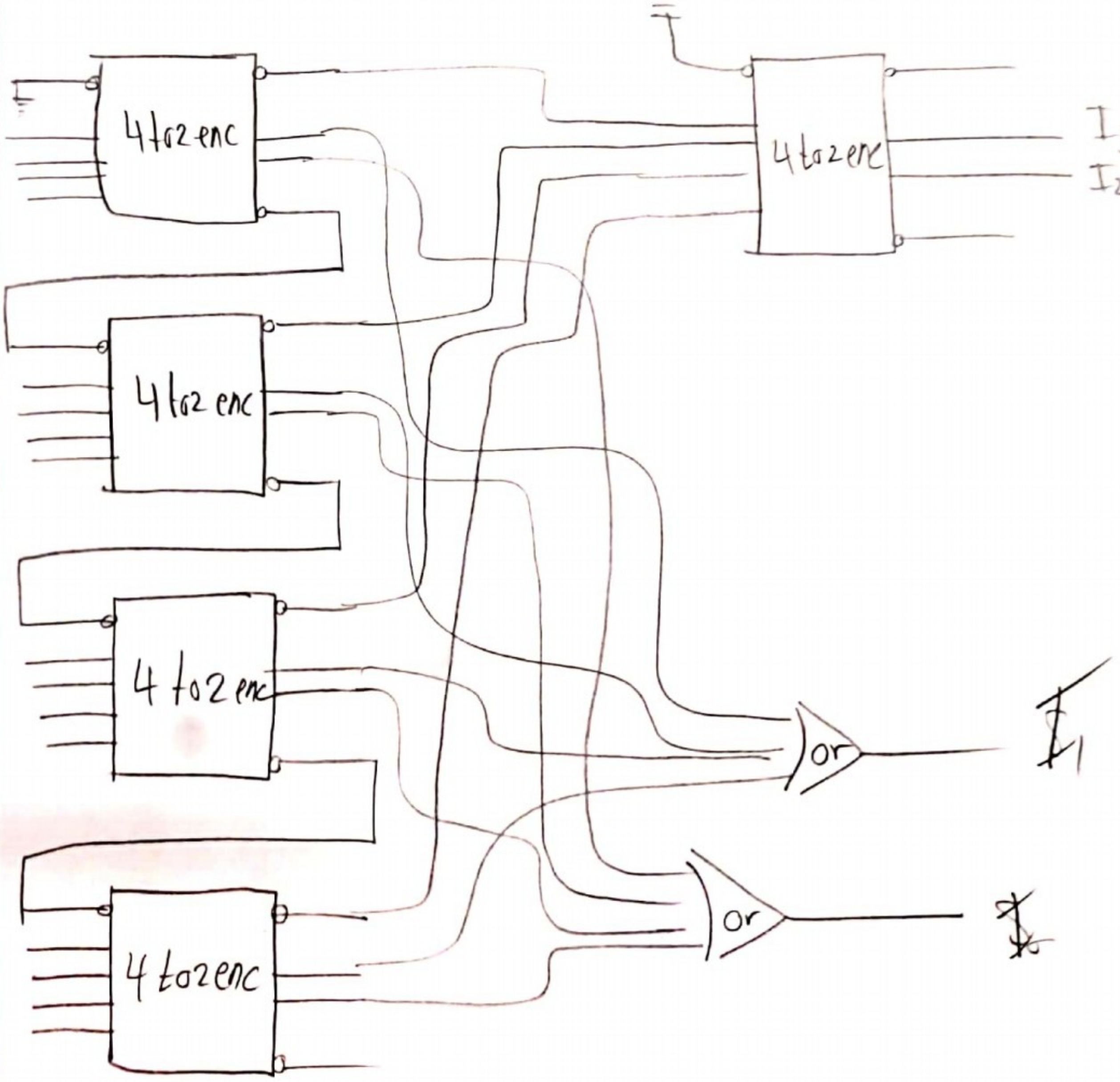


```
 `timescale 1ns/1ns
module Q3PriEnc(input [3:0] S,input Ei, output[1:0] N,output EO,AS );
   assign N = Ei ? 1'bz:
           (S==4'b1xxx) ? 2'b11:
           (S==4'b01xx) ? 2'b10:
           (S==4'b001x) ? 2'b01:
           (S==4'b0001) ? 2'b00:
           (S==4'b0000) ? 2'b00:1'bx;
   assign EO = (S==4'b0000) ? 1'b0 :1'b1;
   assign AS = (S==4'b0000) ? 1'b1 :1'b0;
endmodule
```

Figure 5: Priority Encoder

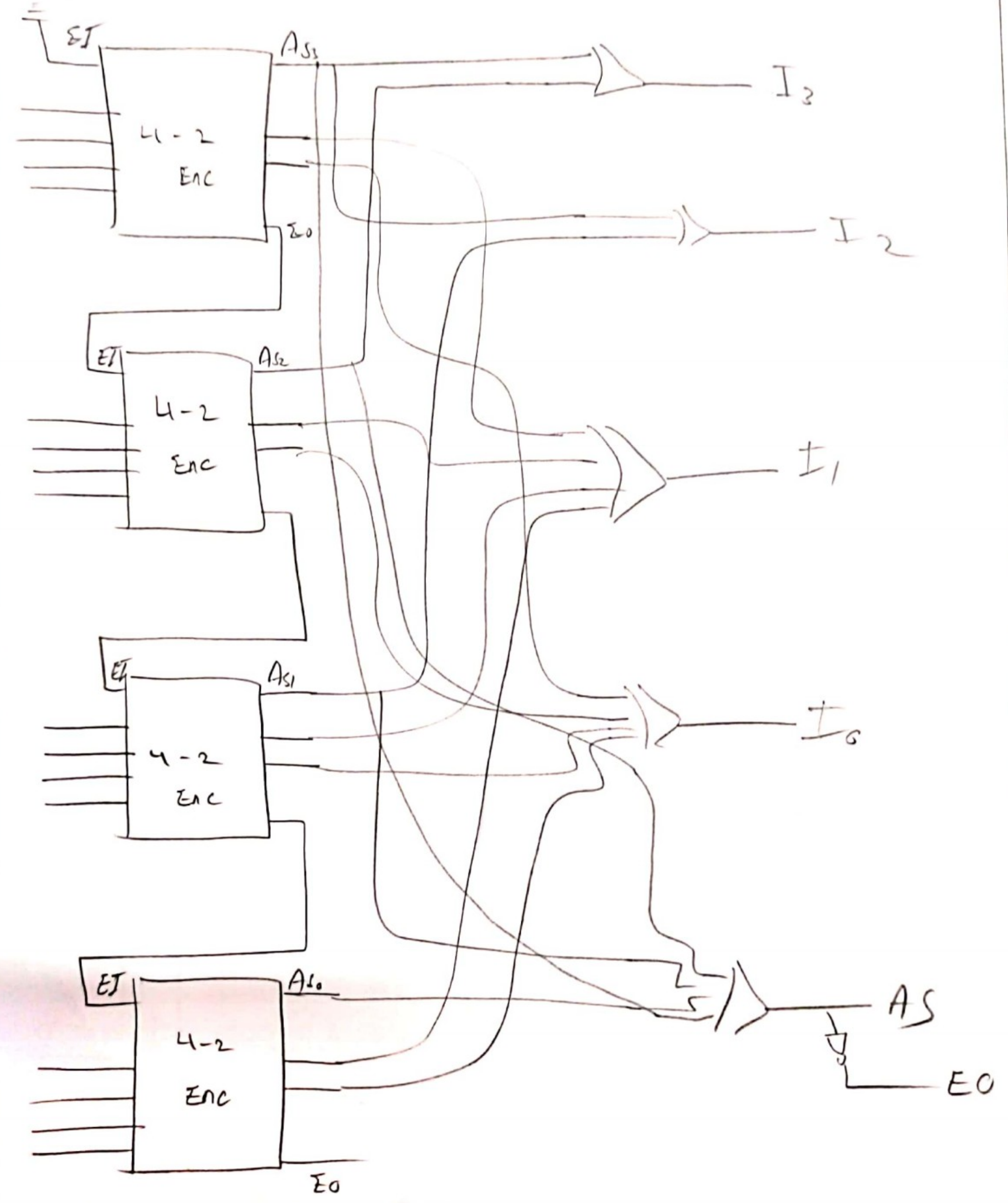
Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW6	Due Date: 00 / 2 / 12	2/6

4- In order to implement this encoder, five 4to 2 encoders, 4 to connect the preliminary 16-bits, another to connect the As outputs consecutively in order to find the MSB and the bit after it, in order to find the last two bits of the output we can use 4input or gates, the logic behind this if the lower 4to2 encoders have 1 as one of their outputs then the higher ones certainly must have it.



Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW6	Due Date: ∞ / 2 / 12	$\frac{6}{6}$

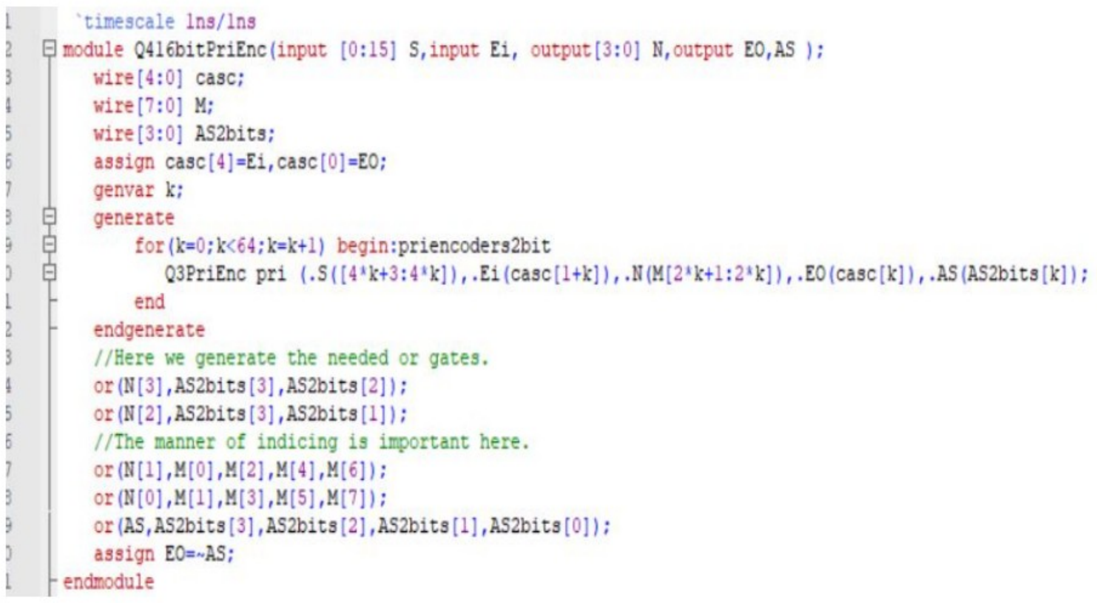
4- (Approach 2) Another approach for this question is used with 4, 2bit



Scanned by CamScanner

Question 4

The system verilog code is as follows:



```
 `timescale 1ns/1ns
module Q416bitPriEnc(input [0:15] S,input Ei, output[3:0] N,output EO,AS );
   wire[4:0] casc;
   wire[7:0] M;
   wire[3:0] AS2bits;
   assign casc[4]=Ei,casc[0]=EO;
   genvar k;
   generate
      for(k=0;k<64;k=k+1) begin:priencoders2bit
         Q3PriEnc pri (.S([4*k+3:4*k]),.Ei(casc[1+k]),.N(M[2*k+1:2*k]),.EO(casc[k]),.AS(AS2bits[k]);
      end
   endgenerate
   //Here we generate the needed or gates.
   or(N[3],AS2bits[3],AS2bits[2]);
   or(N[2],AS2bits[3],AS2bits[1]);
   //The manner of indicing is important here.
   or(N[1],M[0],M[2],M[4],M[6]);
   or(N[0],M[1],M[3],M[5],M[7]);
   or(AS,AS2bits[3],AS2bits[2],AS2bits[1],AS2bits[0]);
   assign EO=~AS;
endmodule
```

Figure 6: 16-bit priority encoder

Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW6	Due Date: 00 / 2 / 12	$\frac{3}{6}$

5- The outputs of such Multiplication is between 0 to 9 hence:

M_3

$B_1B_0 \backslash A_1A_0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

$\rightarrow A_1A_0B_1B_0$

M_2

$A_1B_0 \backslash A_1A_0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	1
10	0	0	1	1

$\rightarrow A_1\bar{A}_0B_1 + A_1B_1\bar{B}_0$

M_1

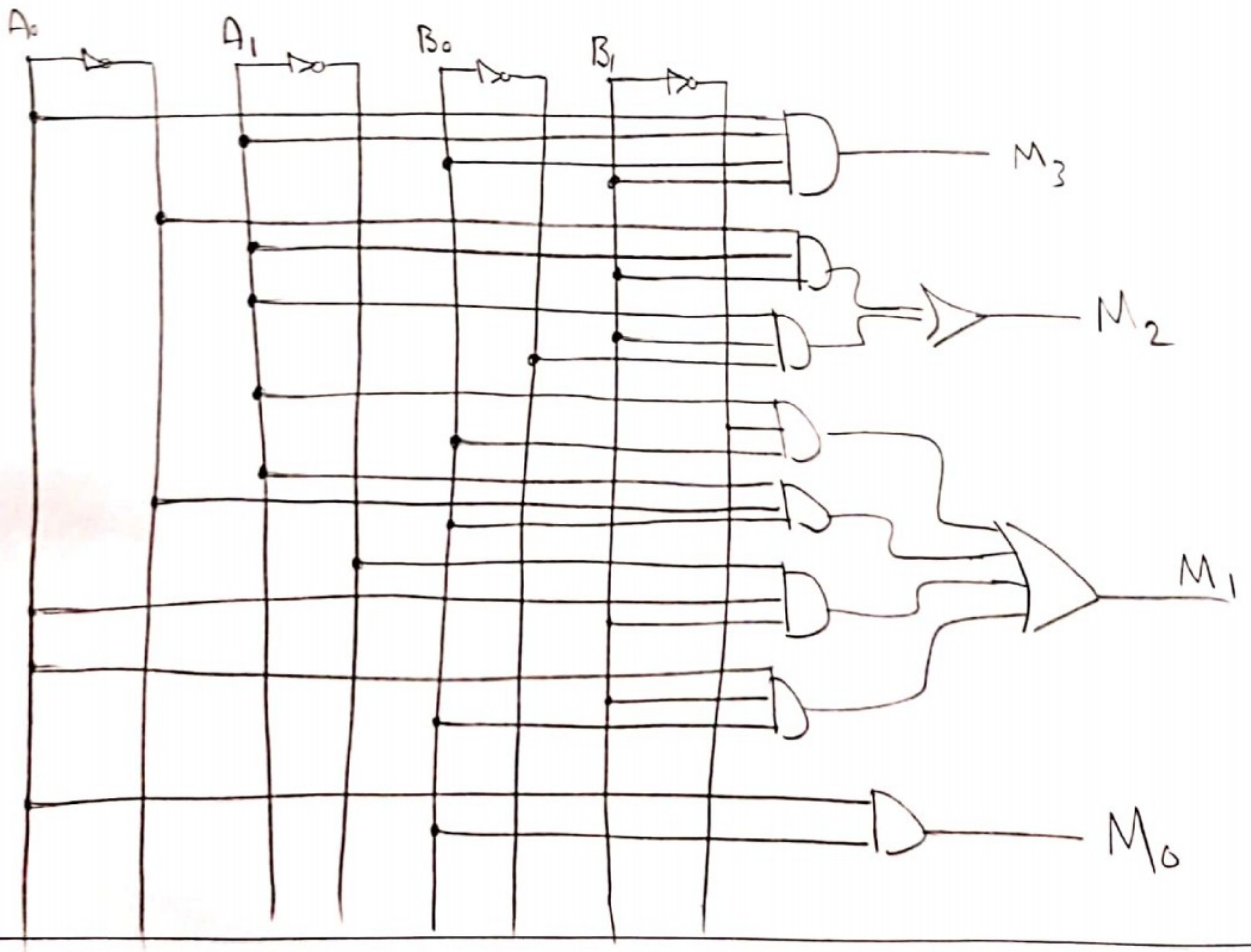
$B_1B_0 \backslash A_1A_0$	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	1	0	1
10	0	1	1	0

$\rightarrow A_1\bar{B}_1B_0 + A_1\bar{A}_0B_0 + \bar{A}_1A_0B_1 + A_0B_1\bar{B}_0$

M_0

$B_1B_0 \backslash A_1A_0$	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

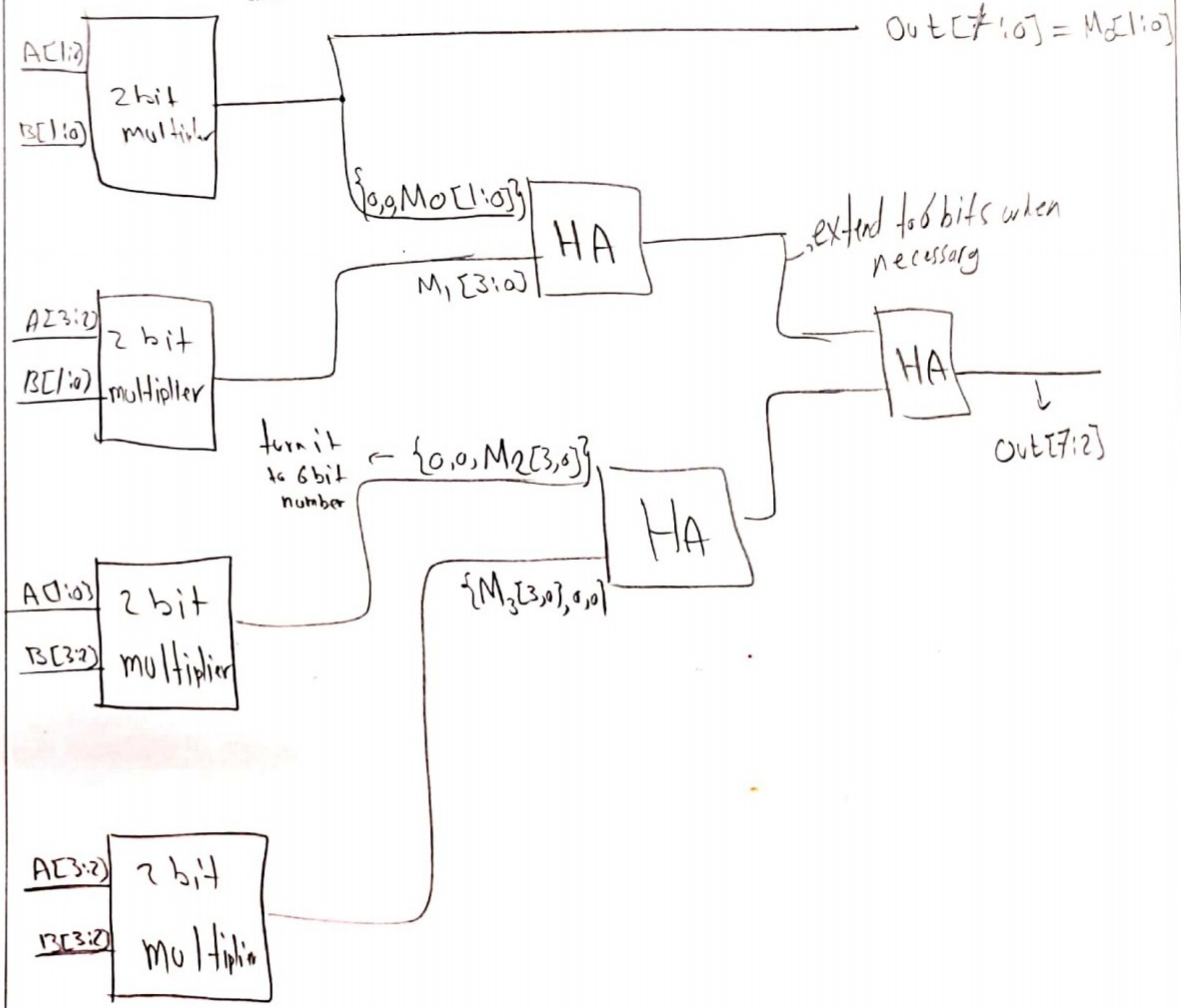
$\rightarrow A_0B_0$



Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW6	Due Date: 00/2/12	4/6

IN order to make a 4 bit multiplier we use three half adders (carries don't matter here) and 4, 2 bit multipliers. we multiply the 2 first and last bits of both numbers (all combinatorial possibilities) when we multiply the two lesser bits of the numbers logically the two last bits of it shall be the last two bits of the multiplied numbers the rest of the design comes as follows

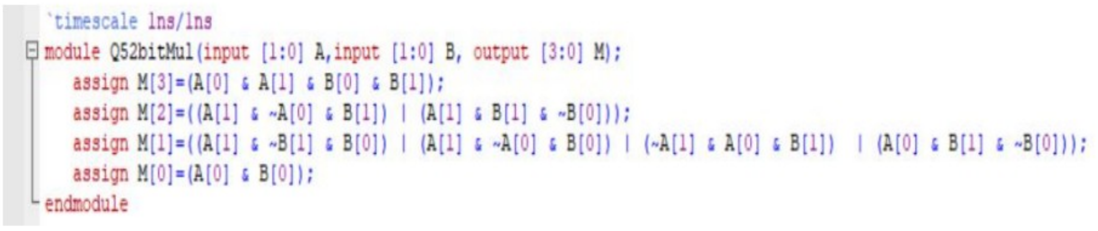
A[3:0], B[3:0] are inputs Out[7:0] → output Mi, output of 2-bit multipliers



* in order to answer this question I made some searches in the internet and got inspired by some designs

Question 6

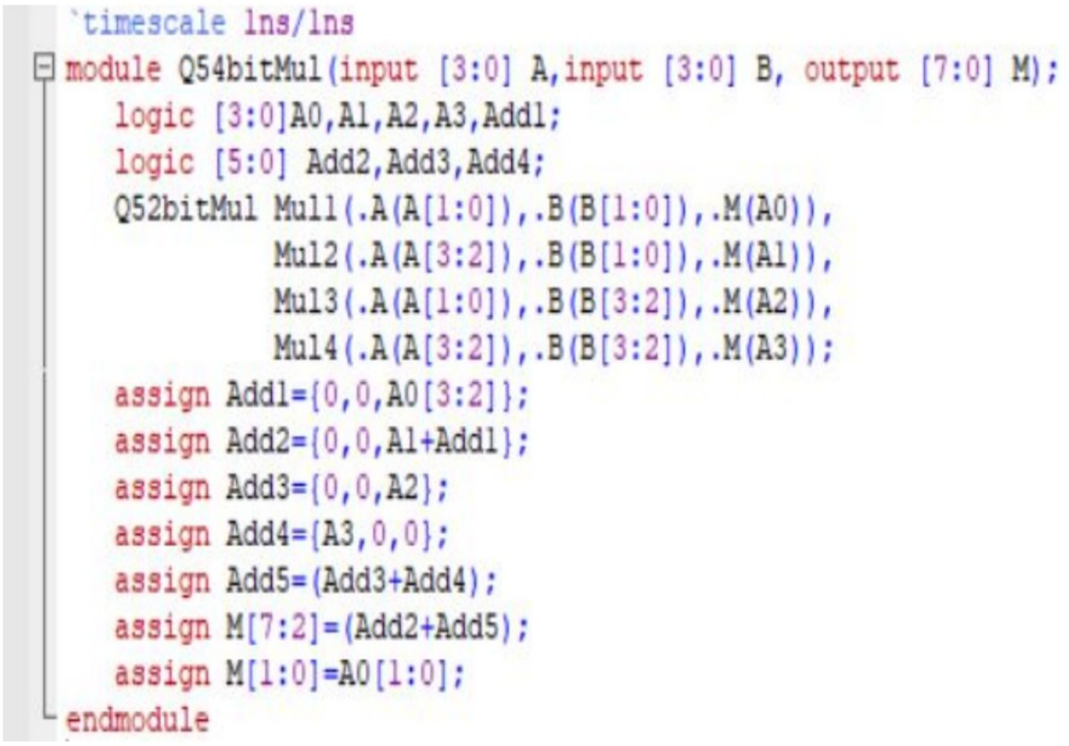
The system verilog code for a 2-bit multiplier is as follows:



```
`timescale 1ns/1ns
module Q52bitMul(input [1:0] A,input [1:0] B, output [3:0] M);
   assign M[3]=(A[0] & A[1] & B[0] & B[1]);
   assign M[2]=((A[1] & ~A[0] & B[1]) | (A[1] & B[1] & ~B[0]));
   assign M[1]=((A[1] & ~B[1] & B[0]) | (A[1] & ~A[0] & B[0]) | (~A[1] & A[0] & B[1])  | (A[0] & B[1] & ~B[0]));
   assign M[0]=(A[0] & B[0]);
endmodule
```

Figure 7: 2-bit multiplier

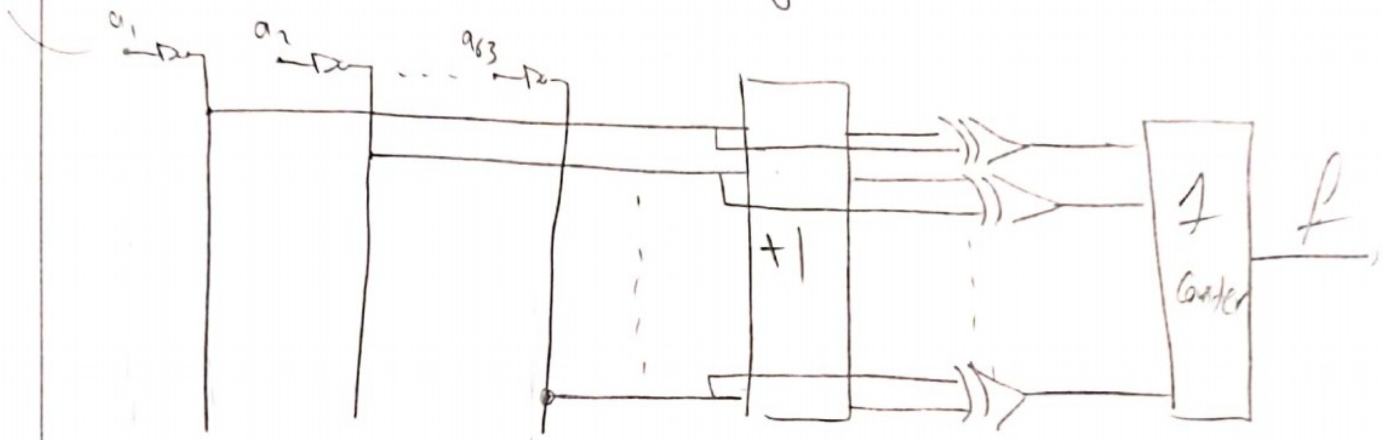
The system verilog code for a 4-bit multiplier is as follows:



```
`timescale 1ns/1ns
module Q54bitMul(input [3:0] A,input [3:0] B, output [7:0] M);
   logic [3:0]A0,A1,A2,A3,Add1;
   logic [5:0] Add2,Add3,Add4;
   Q52bitMul Mul1(.A(A[1:0]),.B(B[1:0]),.M(A0)),
             Mul2(.A(A[3:2]),.B(B[1:0]),.M(A1)),
             Mul3(.A(A[1:0]),.B(B[3:2]),.M(A2)),
             Mul4(.A(A[3:2]),.B(B[3:2]),.M(A3));
   assign Add1={0,0,A0[3:2]};
   assign Add2={0,0,A1+Add1};
   assign Add3={0,0,A2};
   assign Add4={A3,0,0};
   assign Add5=(Add3+Add4);
   assign M[7:2]=(Add2+Add5);
   assign M[1:0]=A0[1:0];
endmodule
```

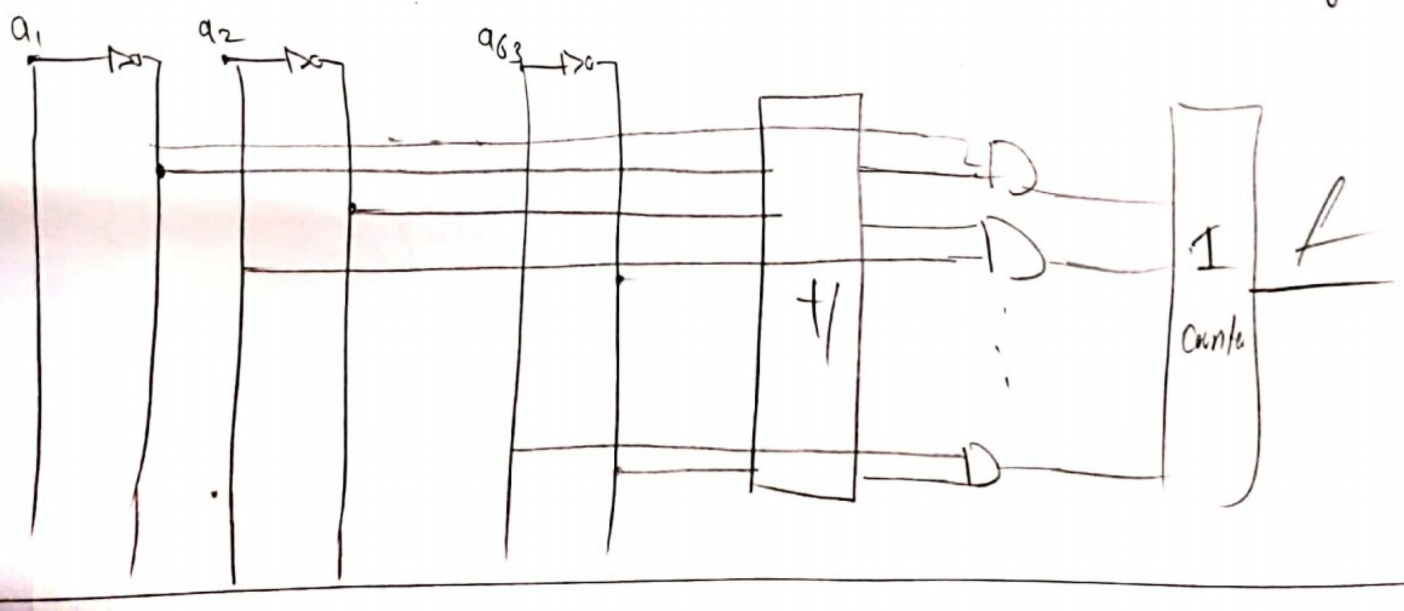
Figure 8: 4-bit multiplier

7- This question wants us to take a number, Calculate its 2's complement, then check how many of the bits are 1 in the Same place for example : A= 10011 → 2's complement: 01101 → overlap= 1 So the design shall be something like this



To implement the 1's Counter we can Simply add the bits of the output XOR's

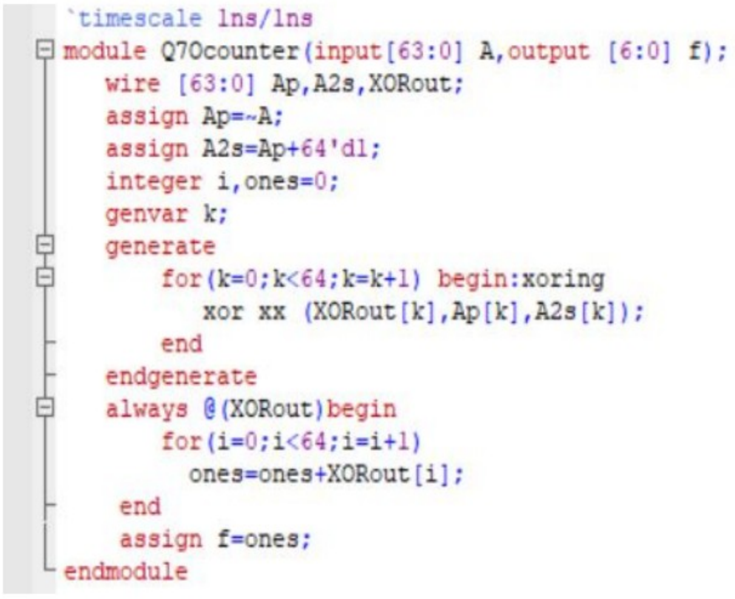
Another approach which can be taken is to and the bits of the original number and it's compliment in the following manner,



Scanned by CamScanner

Question 8

The system verilog code for the OC2B is as follows:



```
`timescale 1ns/1ns
module Q7Ocounter(input[63:0] A,output [6:0] f);
   wire [63:0] Ap,A2s,XORout;
   assign Ap=~A;
   assign A2s=Ap+64'd1;
   integer i,ones=0;
   genvar k;
   generate
       for(k=0;k<64;k=k+1) begin:xoring
          xor xx (XORout[k],Ap[k],A2s[k]);
       end
   endgenerate
   always @(XORout)begin
       for(i=0;i<64;i=i+1)
         ones=ones+XORout[i];
    end
    assign f=ones;
endmodule
```

Figure 9: OC2B