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

Digital Systems 1

Dr.Navabi

Homework 5

Soroush Mesforush Mashhad

SN:810198472

Ordibehesht 00

Question 1

A

The verilog and synthesized code are depicted below:

```
  `timescale 1ns/1ns
module Q1(input a,b,c,d , output f);
assign f = (~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);
endmodule
```

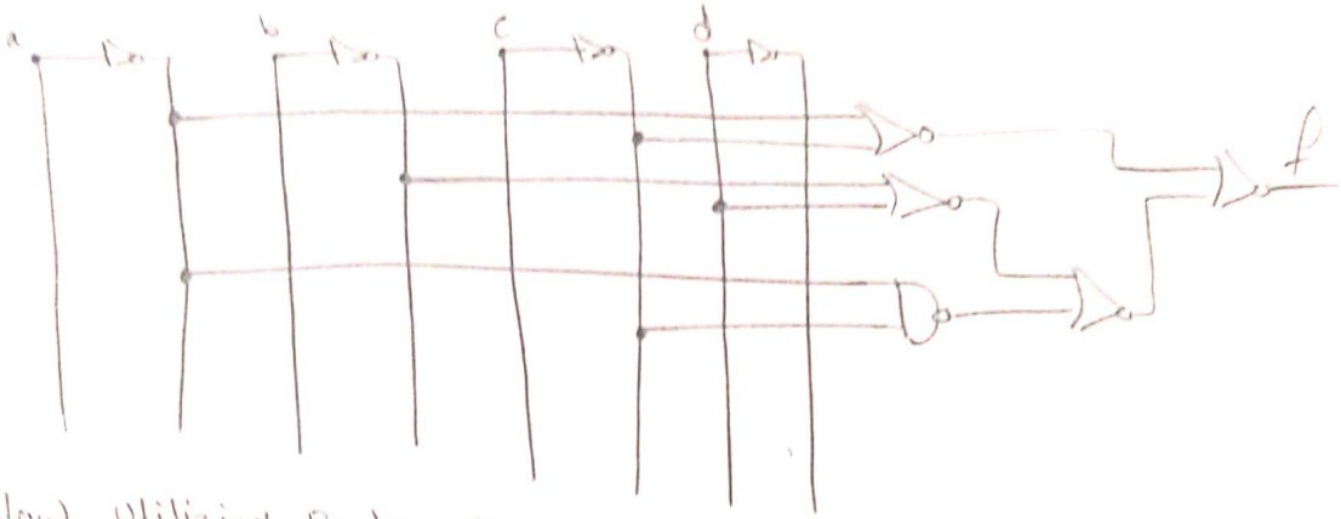
Figure 1: Original Verilog

```
module Q1(a, b, c, d, f);
  wire _00_;
  wire _01_;
  wire _02_;
  wire _03_;
  wire _04_;
  wire _05_;
  wire _06_;
  wire _07_;
  wire _08_;
  wire _09_;
  wire _10_;
  wire _11_;
  wire _12_;
  wire _13_;
  wire _14_;
  wire _15_;
  wire _16_;
  wire _17_;
  wire _18_;
  wire _19_;
  wire _20_;
  wire _21_;
  wire _22_;
  wire _23_;
  wire _24_;
  wire _25_;
  wire _26_;
  wire _27_;
  wire _28_;
  wire _29_;
  input a;
  input b;
  input c;
  input d;
  output f;
  NOT _30_ (.A(_18_),.Y(_25_));
  NOT _31_ (.A(_24_),.Y(_26_));
  NOR _32_ (.A(_26_),.B(_25_),.Y(_28_));
  NOT _33_ (.A(_19_),.Y(_29_));
  NOR _34_ (.A(_27_),.B(_29_),.Y(_20_));
  NAND _35_ (.A(_26_),.B(_25_),.Y(_21_));
  NOR _36_ (.A(_21_),.B(_20_),.Y(_22_));
  NOR _37_ (.A(_22_),.B(_28_),.Y(_23_));
  assign _18_ = c;
  assign _19_ = b;
  assign _24_ = a;
  assign _27_ = d;
  assign f = _23_;
endmodule
```

Figure 2: Synthesized Verilog

1. B) The SOP is: $\bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + \bar{a}b\bar{c}\bar{d} + \bar{a}bcd + \bar{a}bc\bar{d} + ab\bar{c}\bar{d} + ab\bar{c}d$

$+ ab\bar{c}\bar{d} + a\bar{b}\bar{c}d + a\bar{b}\bar{c}\bar{d}$.



Now Utilizing Boolean Algebra we have:

$f: (\bar{a}+\bar{c}) + (\bar{a}.\bar{c}) + (\bar{b}+d) = ac + \overline{(a+c)+(b\bar{d})}$

$= (\bar{a}+\bar{c}).\overline{(a+c)+(b\bar{d})} = (\bar{a}+\bar{c}).((a+c)+b\bar{d})$

$= \bar{a}a + \bar{a}c + a\bar{c} + c\bar{c} + b\bar{c}\bar{d} + \bar{a}b\bar{d} = a\bar{c} + \bar{a}c + b\bar{c}\bar{d} + \bar{a}b\bar{d}$

This is exactly correct because our minimized function is $a\bar{c} + \bar{a}c + \begin{Bmatrix} b\bar{c}\bar{d} \\ \bar{a}b\bar{d} \end{Bmatrix}$

Hence the synthesis is correct

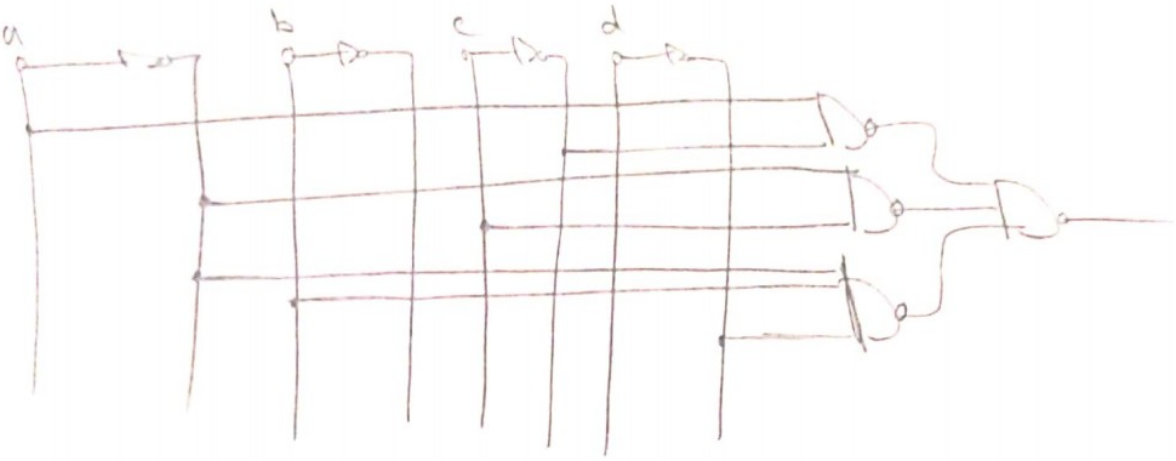
c) $f = \sum_m (2,3,4,6,7,8,9,12,13)$

cd \ ab	00	01	11	10
00	0 (0)	1 (4)	1 (12)	1 (8)
01	0 (1)	0 (5)	1 (13)	1 (9)
11	1 (3)	1 (7)	0 (15)	0 (11)
10	1 (2)	1 (6)	0 (14)	0 (10)

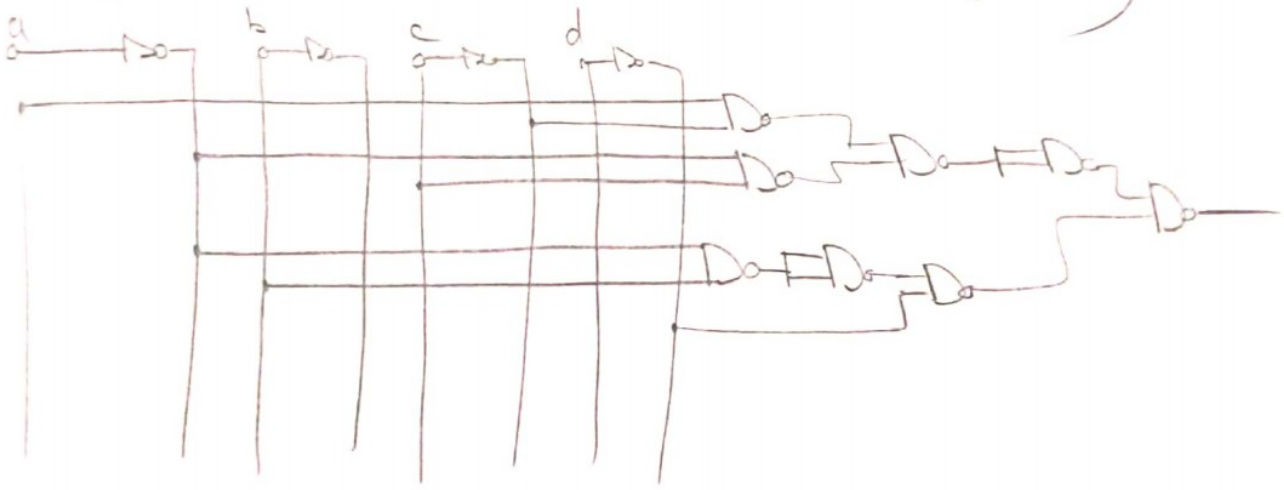
$\longrightarrow f(a,b,c,d) = a\bar{c} + \bar{a}c + \begin{Bmatrix} b\bar{c}\bar{d} \\ \bar{a}b\bar{d} \end{Bmatrix}$

Scanned by CamScanner

If we are allowed to use 3-input NAND gate, $f = a\bar{c} + \bar{a}c + a\bar{b}\bar{d}$



In this case I dare say that this circuit has less gates a Probably a lesser delay but if we use 2 input gates only:



In this case our number of gates increase drastically and I think the delay is Probably more than the Synthesized circuit.

2- b) After synthesizing the Verilog code I've designed this circuit exactly from the synthesis results.

Scanned by CamScanner

Question 2

A

The verilog and synthesized code are depicted below:

```
`timescale 1ns/1ns
module Q2p1(input  A,B,input Cin, output S, output Co);
assign {Co,S}=A+B+Cin;
endmodule
```

Figure 3: Original Verilog

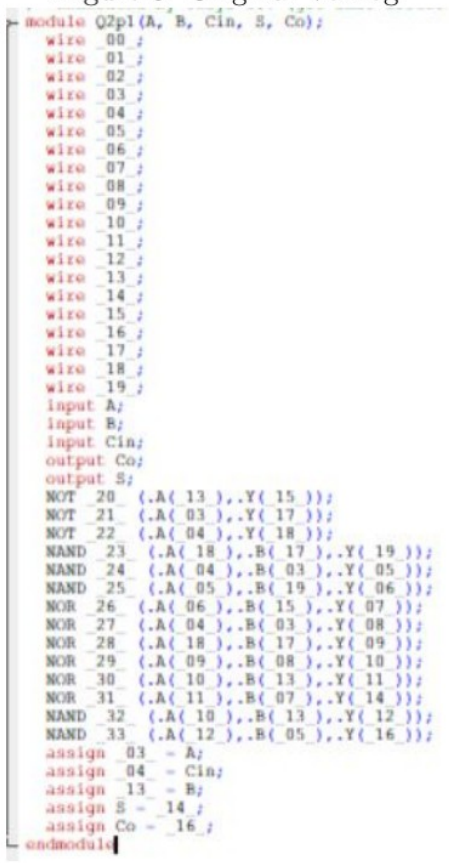
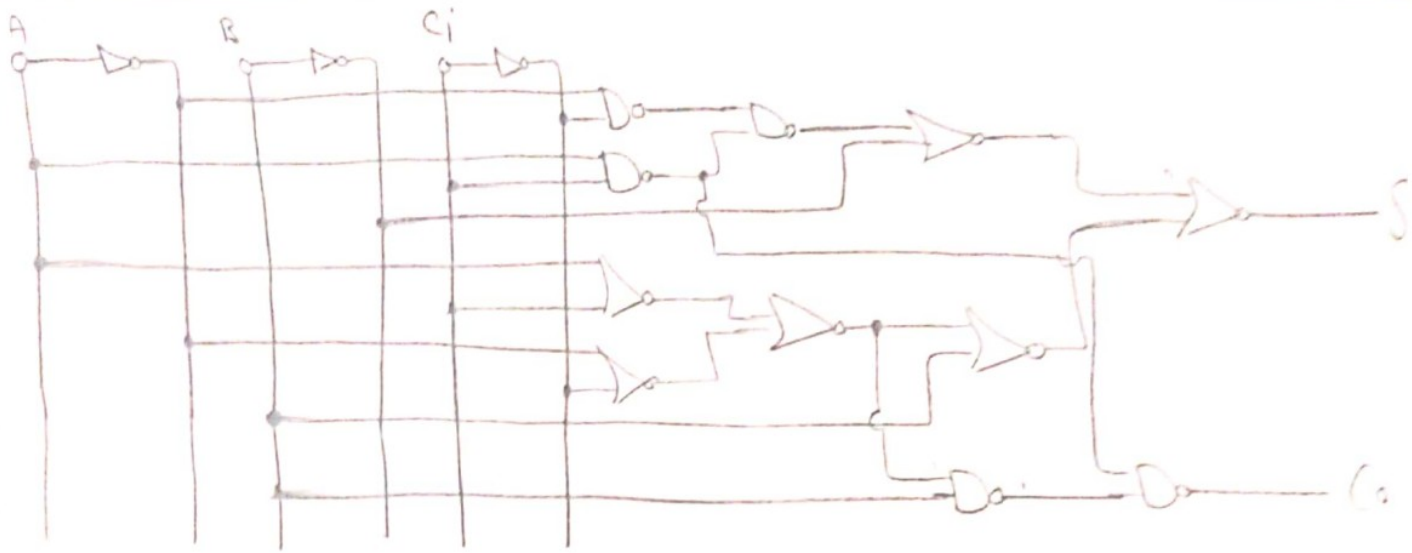


Figure 4: Synthesized Verilog

Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW 5	Due Date: 2/1/14	3/10

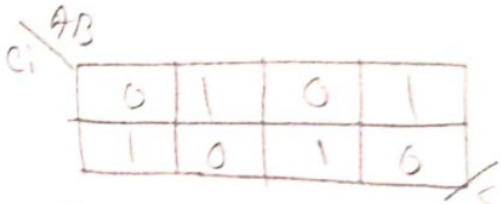


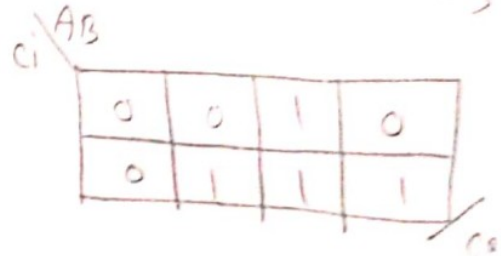
(Full Adder with yosys..)

I checked the truth table for this circuit very carefully and obtained the following results.

A	B	Ci	S	Co
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

K-Map Representation





The Above table and k-maps is exactly the same as the full Adder talked about in class, hence the synthesis has worked smoothly.

Scanned by CamScanner

C

Firstly, I have included the photos for the number of gates and cells of Part b before and after re-integration:

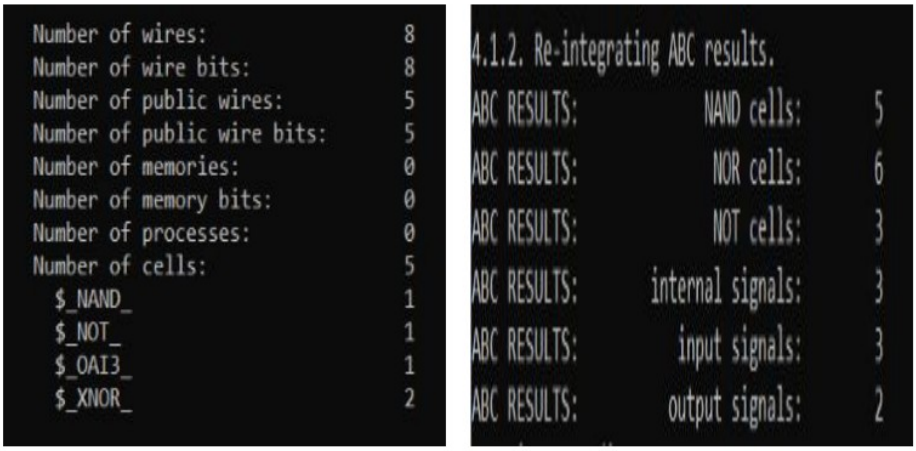


Figure 5: Part B

and now part c:

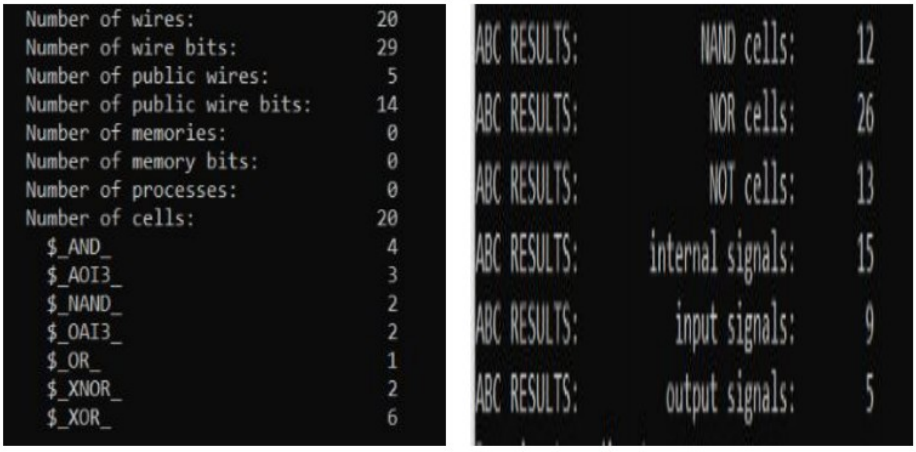


Figure 6: Part C

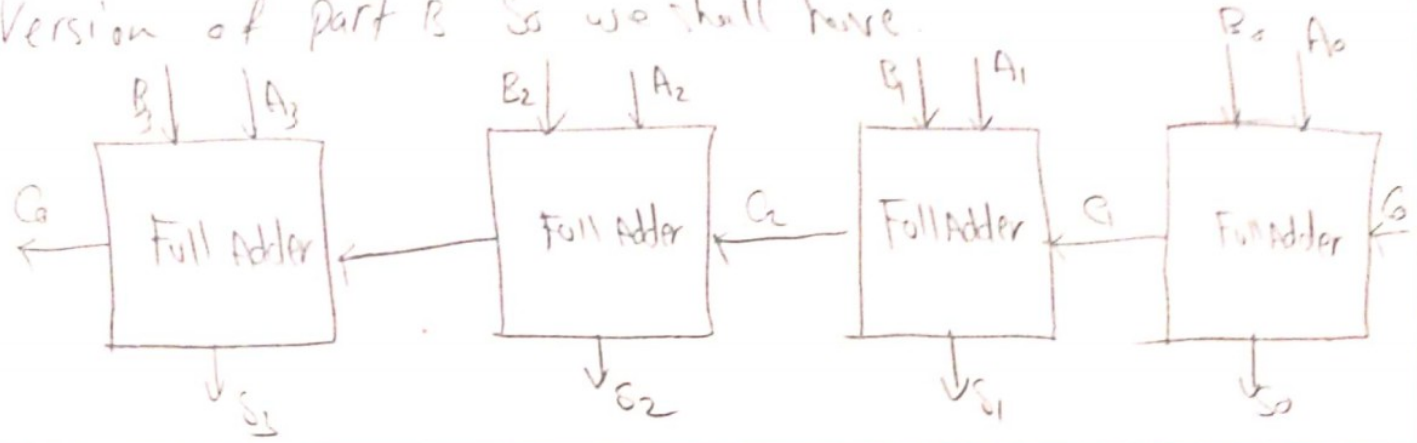
As we expected in the 4-bit full adder we have more gates than the one in part b, the exact values(after reintegration) are depicted in the table below:

B	C
5 NAND's	12 NAND's
6 NOR's	26 NOR's
3 NOT's	13 NOT's

As we can see the patterns in part b have repeated themselves in this adder, for example in both case the number of NOR cells is twice the number of NOT cells and etc.

Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW 5	Due Date: 01 / 2 / 14	4/10

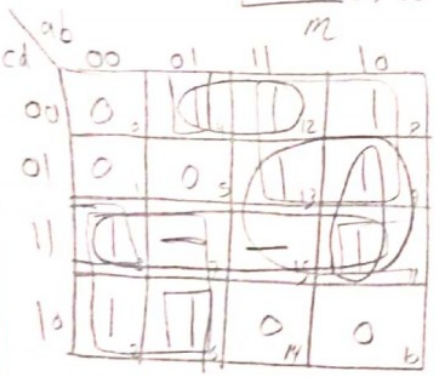
D) We endure that the 4-bit adder should kind of be the cascaded version of part B So we shall have



3- A) First of all we find the PI's and EPI's by using K-maps

$f_{EO}(a,b,c,d) = \sum_m (2,3,4,6,8,9,11,12,13) + d(7,15)$

cubical form: a b c d



PI: $a\bar{c}$, $\bar{a}c$, $\bar{b}cd$, $\bar{a}b\bar{d}$, $a\bar{b}d$, cd, $b\bar{c}\bar{d}$, ad

EPI: $a\bar{c}$, $\bar{a}c$

Now I shall show the PI's in the cubical form they are: PI: 1x0x, 0x1x, x011, 01x0, 10x1, xx11, x100

B) For the EPIs we have: EPIs: 1x0x, 0x1x

C) $f(a,b,c,d) = a\bar{c} + \bar{a}c + cd + \bar{a}b\bar{d}$

$f(a,b,c,d) = a\bar{c} + \bar{a}c + cd + b\bar{c}\bar{d}$

$f(a,b,c,d) = a\bar{c} + \bar{a}c + ad + \bar{a}b\bar{d}$

$f(a,b,c,d) = a\bar{c} + \bar{a}c + ad + b\bar{c}\bar{d}$

Scanned by CamScanner

Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW5	Due Date: 00/12/14	5/10

4 a) $f(a,b,c,d) = \sum_m (2,3,4,6,7,8,9,12,13)$

cd \ ab	00	01	11	10
00	0_0	1_1	1_{12}	1_8
01	0_1	0_5	1_{13}	1_9
11	1_3	1_7	0_{15}	0_{11}
10	1_2	1_6	0_{14}	0_{10}

	0 cubes	1 Cubes	2 cubes
0011	0010 ✓	001X ✓	0X1X
0010	0100 ✓	0X10 ✓	1X0X
0100	1000 ✓	01X0	
0111	0011 ✓	X100	
0110	0110 ✓	100X ✓	
1101	1021 ✓	1X00 ✓	
1100	1100 ✓	0X11 ✓	
1003	0111 ✓	011X ✓	
1001	1101 ✓	1X01 ✓	
		110X ✓	

The PIs are:

01X0 , X100 , 0X1X , 1X0X

b) The EPIs are: 1X0X , 0X1X

c)

PI \ cell	0010	0011	0100	0110	0111	1000	1001	1100	1101
01X0			✓	✓					
X100			✓					✓	
0X1X	✓	✓		✓	✓				
1X0X						✓	✓	✓	✓
f	✓	✓		✓	✓	✓	✓	✓	✓

$f(a,b,c,d) = a\bar{c} + \bar{a}c + \bar{a}b\bar{d}$

$f(a,b,c,d) = a\bar{c} + \bar{a}c + b\bar{c}\bar{d}$

Scanned by CamScanner

Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW5	Due Date: 00/12/4	6/10

5- Firstly we shall check the table for a 3-to-8 binary decoder: (Active high of course)

en (active low)	A_0	A_1	A_2	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	0	0	0	1	0
0	1	1	1	0	0	0	0	0	0	0	1
1	-	-	-	0	0	0	0	0	0	0	0

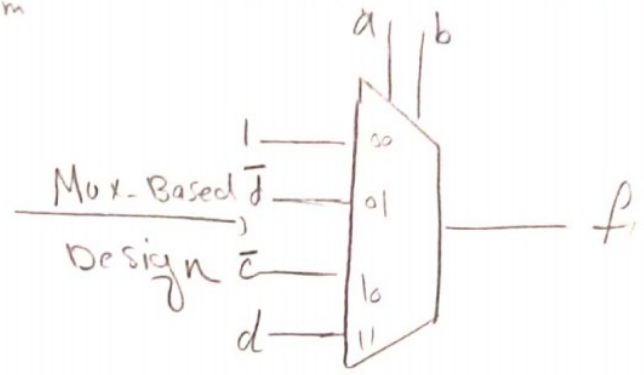
System Verilog description:

```
`timescale 1ns / 1ns
module DCD3to8 (input [2:0] A, input en, output [0:7] F);
    assign F = en ? 8'd0 :
        (A==0) ? 8'b10000000 :
        (A==1) ? 8'b01000000 :
        (A==2) ? 8'b00100000 :
        (A==3) ? 8'b00010000 :
        (A==4) ? 8'b00001000 :
        (A==5) ? 8'b00000100 :
        (A==6) ? 8'b00000010 :
        (A==7) ? 8'b00000001 :
        8'bxxxxxxxx
endmodule
```

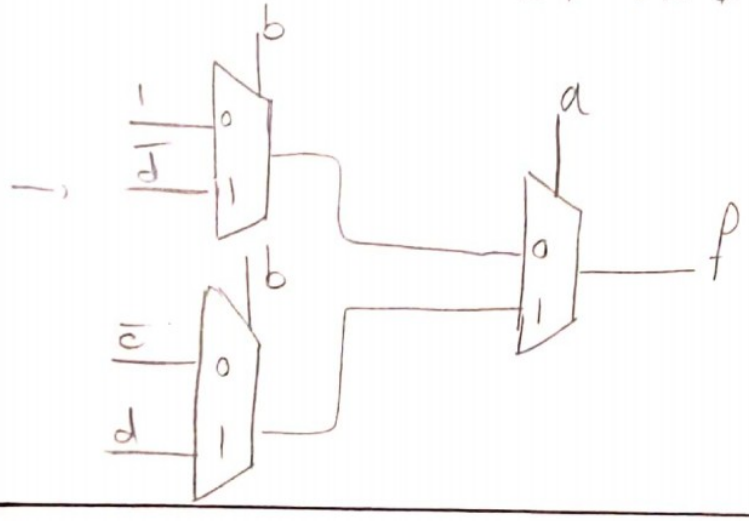

Scanned by CamScanner

6- $f(a,b,c,d) = \sum_m (0,1,2,3,4,6,8,9,13,15), d(5,11,14)$

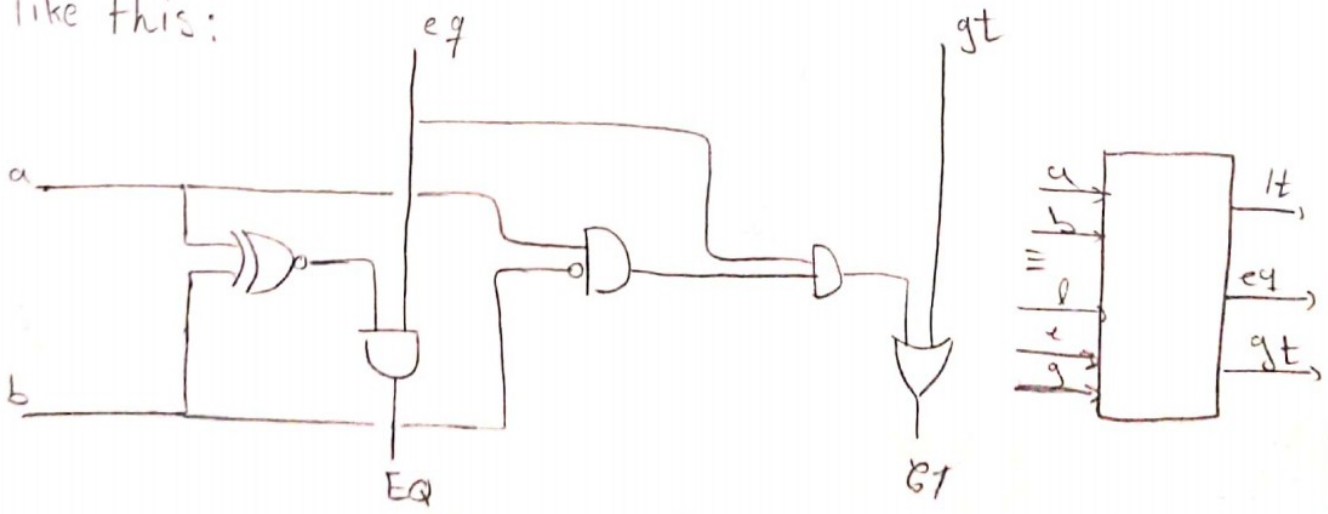
ab \ cd	00	01	11	10
00	1 (0)	1 (4)	0 (12)	1 (8)
01	1 (1)	d (5)	1 (13)	1 (9)
11	1 (3)	0 (7)	1 (15)	d (11)
10	1 (2)	1 (6)	d (14)	0 (10)



Now we most break that 4-to-1 MUX, to, 2-to-1 MUX's



8- As discussed in the lectures the design is something like this:



Now we shall answer the question:

Scanned by CamScanner

Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW5	Due Date: 00/2/4	8/10

A) ~timescale 1ns/1ns

```
module CMP1bit (input a,b,l,e,g, output lt,gt,eq);
    assign {lt,eq,gt} = (a<b)? 3'b100 :
                        (a>b)? 3'b001 :
                        (a==b)? {l,e,g} : 3'bxxx
endmodule
```

b) In order to create an 8-bit comparator I am forced to use 8, of the 1 bit Comparators Cascaded.

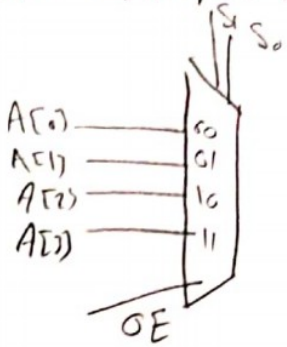
~timescale 1ns/1ns

```
module CMP8bit (input [7:0] A, input [7:0] B, output lt,gt,eq);
    logic [7:0] l,e,g;
    assign {l[0],g[0],e[0]} = 3'b001;   // like in the lecture.
    CMP1bit c1(A[0],B[0],l[0],e[0],g[0], l[1],g[1],e[1]),
            c2(A[1],B[1],l[1],e[1],g[1], l[2],g[2],e[2]),
            c3(A[2],B[2],l[2],e[2],g[2], l[3],g[3],e[3]),
            c4(A[3],B[3],l[3],e[3],g[3], l[4],g[4],e[4]),
            c5(A[4],B[4],l[4],e[4],g[4], l[5],g[5],e[5]),
            c6(A[5],B[5],l[5],e[5],g[5], l[6],g[6],e[6]),
            c7(A[6],B[6],l[6],e[6],g[6], l[7],g[7],e[7]),
            c8(A[7],B[7],l[7],e[7],g[7], lt,gt,eq);
endmodule
```


Scanned by CamScanner

7-a) The codes are as follows.

```
module mux4to1 low (input [3:0] A, input [1:0] S, input oe, output f);
    assign f = oe ? 1'bz :
        (S = 2'b00) ? A[0] :
        (S = 2'b01) ? A[1] :
        (S = 2'b10) ? A[2] :
        (S = 2'b11) ? A[3] : 1'bx;
endmodule
```




```
module mux4to1 high (input [3:0] A, input [1:0] S, input oe, output f);
    assign f = ~oe ? 1'bz :
        (S = 2'b00) ? A[0] :
        (S = 2'b01) ? A[1] :
        (S = 2'b10) ? A[2] :
        (S = 2'b11) ? A[3] : 1'bx
endmodule
```

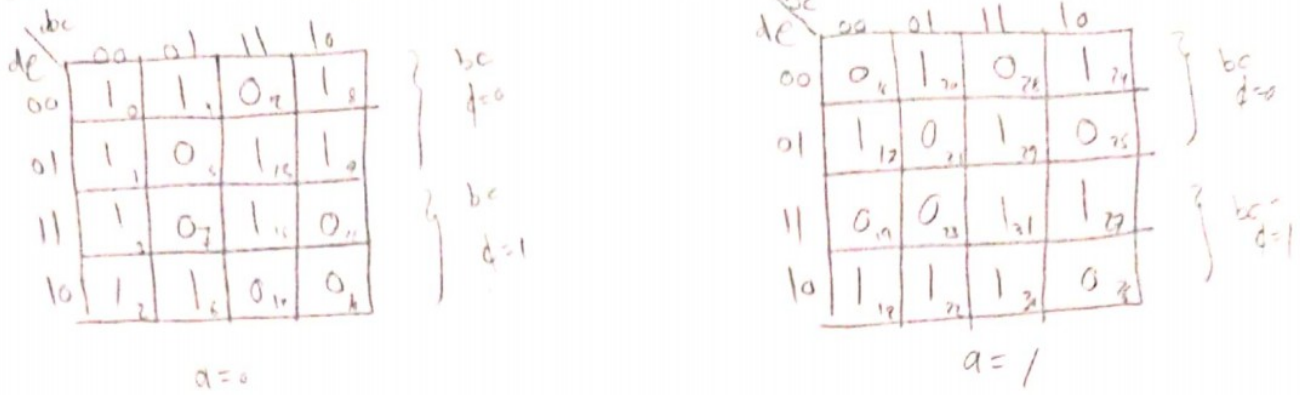
b) Similar to the project I shall create a 16-to-1 MUX

```
~ time scale 1ns / 1ns
module mux16to1 (input [0:15] J, input [3:0] S, oe, output f);
    wire f1, f2, f3, f4;
    mux4to1 low M1 (.A({J[0], J[1], J[2], J[3]}), .S({S[1], S[0]}), .oe(oe), .f(f1));
    M2 (.A({J[4], J[5], J[6], J[7]}), .S({S[1], S[0]}), .oe(oe), .f(f2));
    M3 (.A({J[8], J[9], J[10], J[11]}), .S({S[1], S[0]}), .oe(oe), .f(f3));
    M4 (.A({J[12], J[13], J[14], J[15]}), .S({S[1], S[0]}), .oe(oe), .f(f4));
    M5 (.A({f1, f2, f3, f4}), .S({S[2], S[3]}), .oe(oe), .f(f));
endmodule.
```

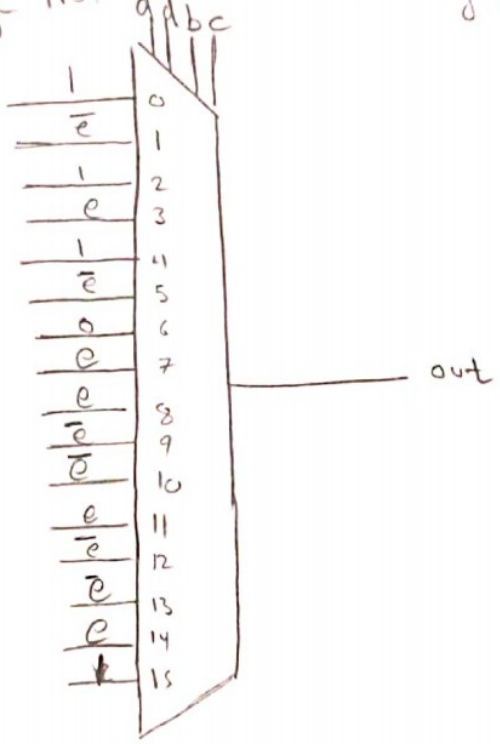

Scanned by CamScanner

Soroush Mesforush Mashhad Student Number: 810198472	Digital Logic Design HW5	Due Date: 00/12/4	10/10

c) with the help of two K-maps we have:



using our knowledge from mux based designs we have



~timescale 1ns/1ns

```
module funct (input a,b,c,d,e,f,output out f);
    wire J[0,15];
    wire S[3,0];
    assign J[0,15] = {1'b1, ~e, 1'b1, e, 1'b1, ~e, 1'b0, e, e, ~e, ~e, e, ~e, ~e, e, 1'b1};
    assign S[3:0] = {a,d,b,c};
    mux16to1 M1(.J(J), .S(S), .oe(oe), .f(f));
endmodule
```


Scanned by CamScanner