

Faculty of Engineering and Technology Department of Electrical and Computer Engineering

Digital System
ENCS234
Verilog project

Student's Name: Hazar Michael

ID: #1201838

Section: #4

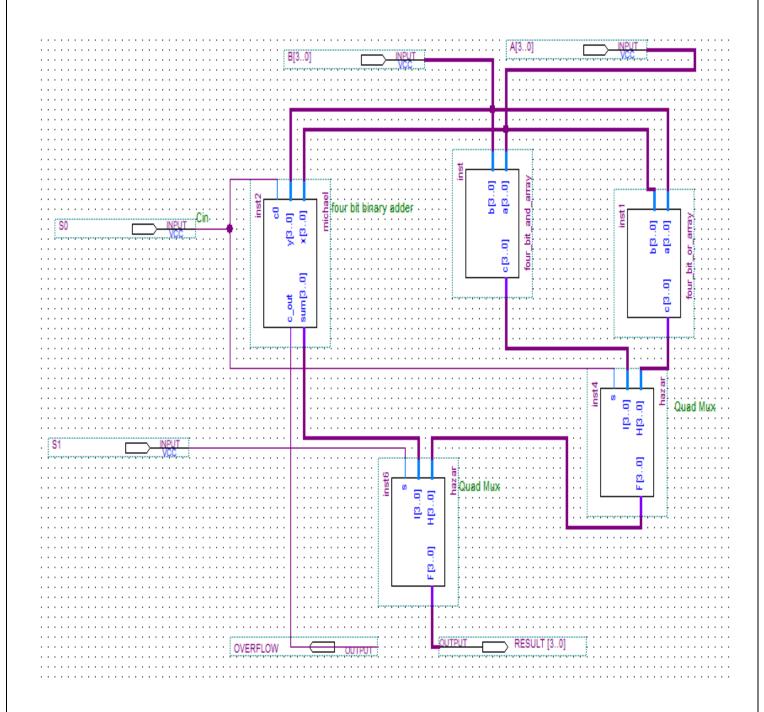
Instructor: Dr. Ismail Khater

June 2022

Contents

Whole system design:	3
One-Bit binary adder:	
Four-Bit binary adder:	
Mux 2x1:	6
Quad mux 2x1:	
Four bit OR array:	8
Four bit AND array:	

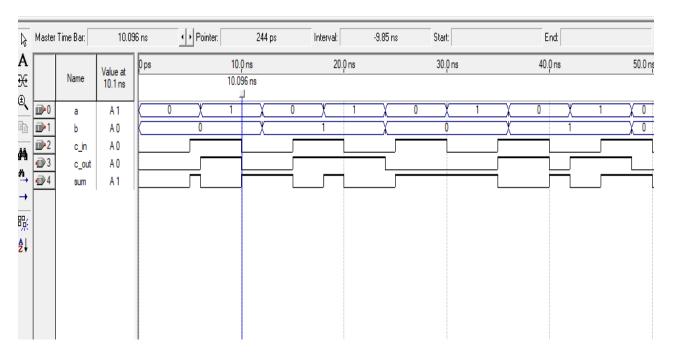
Whole system design:



One-Bit binary adder:

• Verilog code:

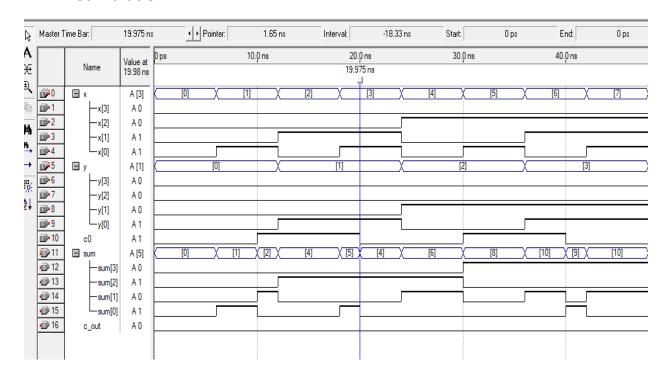
```
one_bit_adder.v
                 //hazar michael 1201838
2
#4 😘
            3
               module one_bit_adder(a,b,c_in,sum,c_out);
            4
₹}
            5
                 input a,b,c in;
            6
                 output sum, c_out;
賃 賃
                 assign sum = a^b^c in;
16 %
            8
                 assign c_out = (a&&b) | | (a&&c_in) | | (b&&c_in);
            9
                 endmodule
% %
           10
0 5
€2
267
268 ab/
```



Four-Bit binary adder:

• Verilog code:

```
nichael.v
           1
---
           2
                 //hazar michael 1201838
# 1
           3
                 //four bit binary adder
           4
7
           5
               module michael (x,y,c0,sum,c out);
           6
                 input c0;
+≡ +≡
           7
                 input [3:0] x, y;
16 %
           8
                output [3:0] sum;
           9
                output c out;
% %
          10
                 wire cl,c2,c3;
          11
7 0
          12
                 one bit adder Gl (x[0],y[0], c0, sum[0], c1);
€2
          13
                one bit adder G2 (x[1],y[1], c1, sum[1], c2);
          14
                 one_bit_adder G3(x[2],y[2], c2, sum[2], c3);
267
268 ab/
          15
                 one bit adder G4(x[3],y[3], c3, sum[3], c4);
          16
          17
  - 52
          18
                 endmodule
```



Mux 2x1:

• Verilog code:

```
    mux_2x1.v

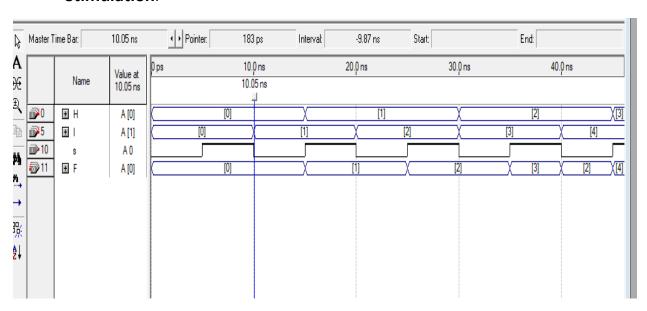
             1
                    //hazar michael 1201838
---
             2
                  module mux_2x1 (I0,I1,s,F);
#4 😘
             3
             4
                   input IO, I1, s;
7
             5
                   output F;
             6
賃 賃
             7
                    assign \cdot F = \cdot (I0\&\&!s) \mid \mid (I1\&\&s);
16 %
             8
                    endmodule
7& X
0 5
267
268 ab/
 | | .....
≣ 🖺
```



Quad mux 2x1:

• Verilog code:

```
nazar.v
                 //hazar michael 1201838
---
            2
                 //Quad mux
#4 1.5
            3
            4
               module hazar (H, I, s, F);
7
            5
                 input [3:0] H, I;
            6
                 input s ;
賃 賃
            7
16 %
            8
                 output [3:0] F;
            9
                 mux 2x1 G1(H[0], I[0], s, F[0]);
% %
           10
                 mux 2x1 G2(H[1], I[1], s, F[1]);
                 mux 2x1 G3(H[2], I[2], s, F[2]);
2
           11
           12
                 mux_2x1 G4(H[3], I[3], s, F[3]);
€2
           13
           14
                 endmodule
267
268 ab/
           15
```

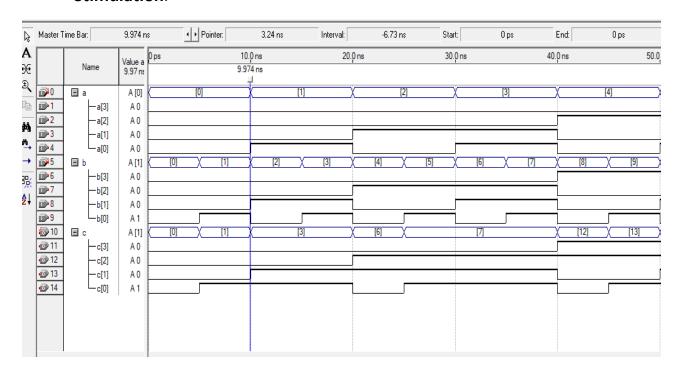


Four bit OR array:

• Verilog code:

```
four_bit_or_array.v
                 //hazar michael 1201838

            2
               module four_bit_or_array (a,b,c);
44 €,8
            3
                 input [3:0]a,b;
            4
                 output [3:0] c;
{}
            5
            6
賃 賃
                 assign c[0] = a[0] | | b[0]; --
            7
                 assign c[1] = a[1] | | b[1]; --
            8
                 assign c[2] = a[2] | | b[2]; -
            9
                 assign c[3] = a[3] | | b[3]; -
% %
           10
           11
                 endmodule
0 2
           12
267
268 ab/
```



Four bit AND array:

• Verilog code:

```
four_bit_and_array.v
                   //hazar michael 1201838
             1
----
             2
                 module four bit and array (a,b,c);
#4 🛵
             3
                   input [3:0]a,b;
             4
                   output [3:0] c;
{}
             5
             6
                   assign c[0] = a[0] && b[0]; ...
擅 賃
             7
                   assign \cdot c[1] \cdot = \cdot a[1] \cdot & & \cdot b[1];
16 %
             8
                   assign c[2] = a[2] && b[2]; --
             9
                   assign c[3] = a[3] && b[3]; ...
            10
                   endmodule
            11
Z
            12
₩
267 ab/
   | ....:
```

