جامعة بيرزيت
BIRZEIT UNIVERSITY

Verilog Project

Mohammad Abujaber 1190298

Yazan Shrouf 1190145

INDEX

The project's combinational circuit --------------------------------------- 1

1-bit adder/subtractor ---2

4-bit adder/subtractor ---3

BCD adder/subtractor --4

MAX ---5

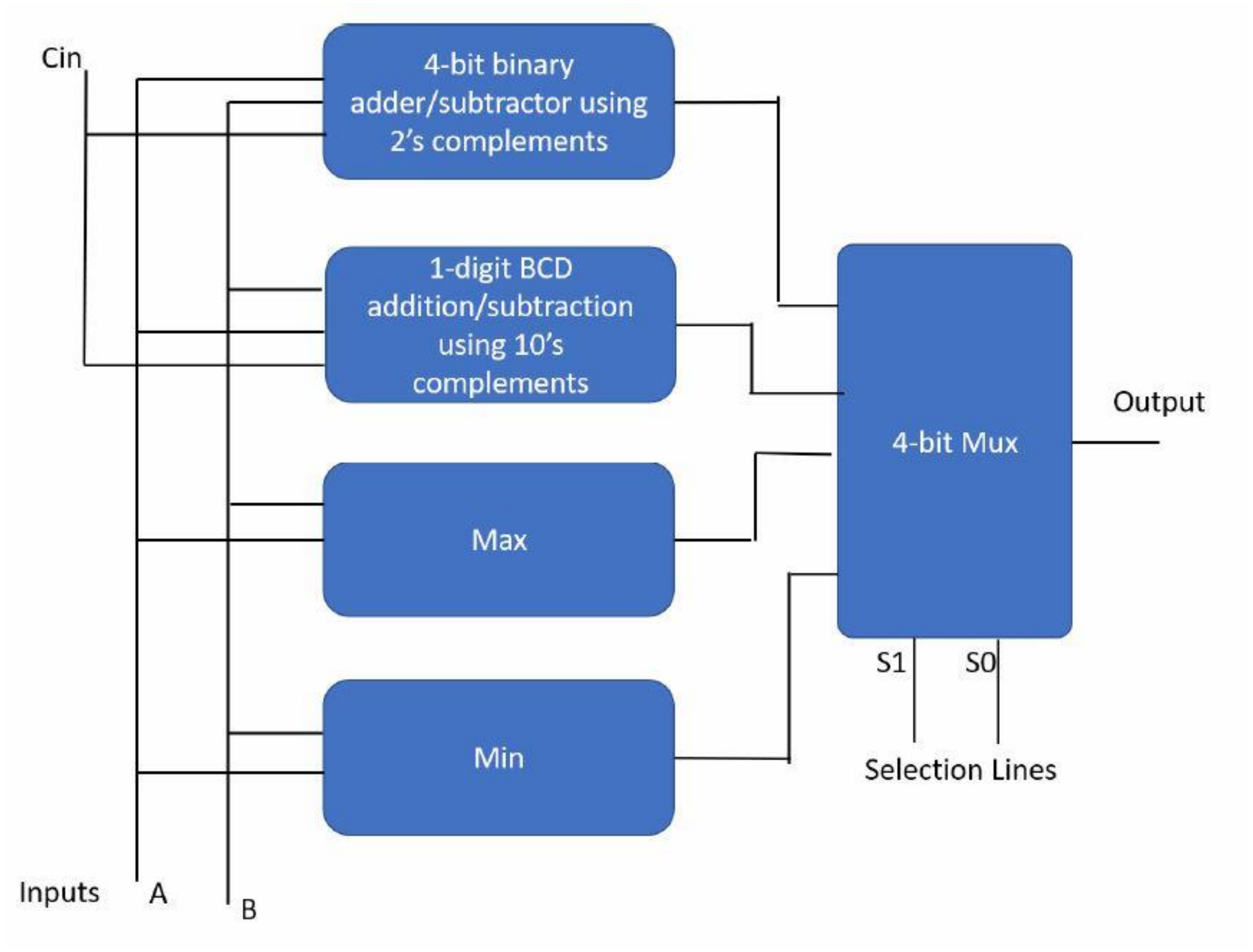
MIN ---6

MUX ---7

The full circuit --8

The teamwork ---9

The project's combinational circuit



Cin
4-bit binary adder/subtractor using 2's complements
1-digit BCD addition/subtraction using 10's complements
Max
Min
4-bit Mux
Output
S1
S0
Selection Lines
Inputs
A
B

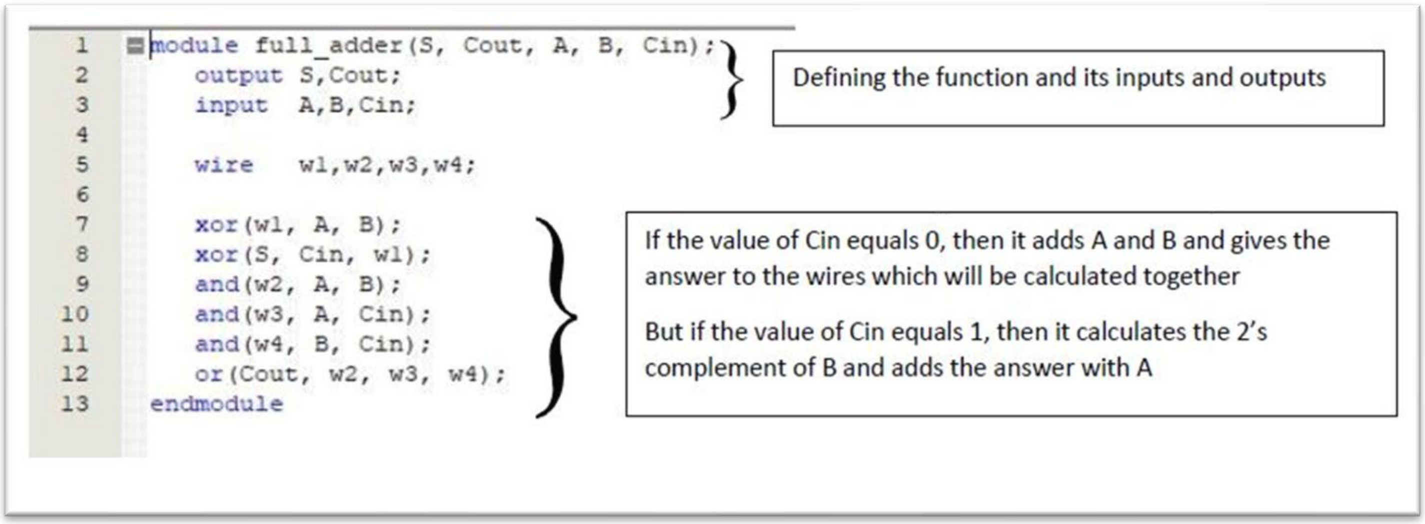
1-bit adder-sub

Consists of:

1-Inputs: A , B , Cin .

2-Gates: 2 XOR gates , 3 AND gates , 1 OR gate.

3-Outputs: S , Cout .

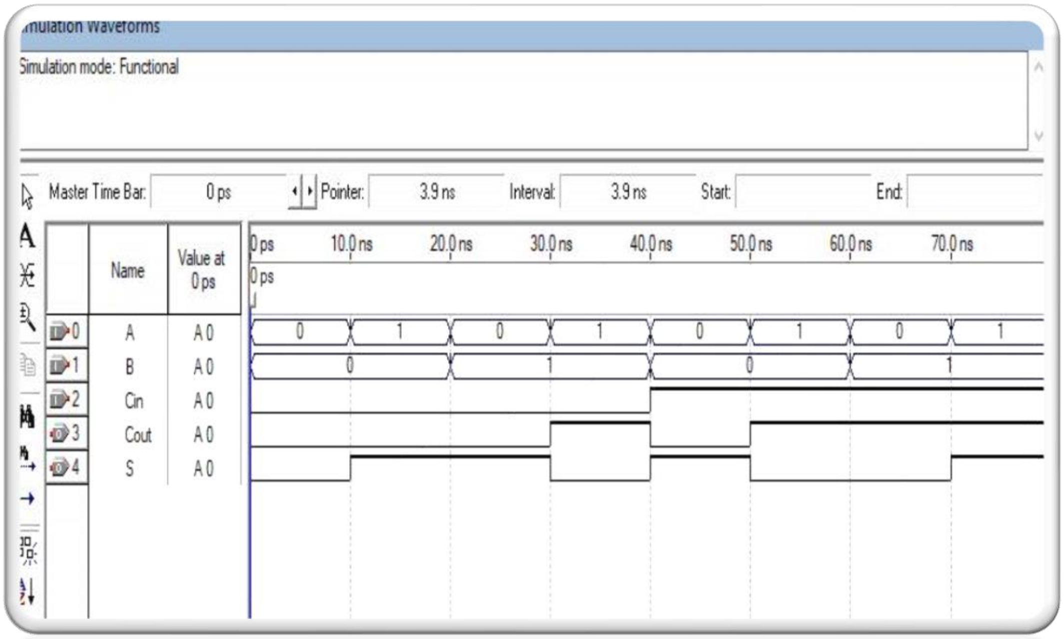


```
module full_adder(S, Cout, A, B, Cin);
    output  S,Cout;
    input   A,B,Cin;

    wire    w1,w2,w3,w4;

    xor(w1, A, B);
    xor(S, Cin, w1);
    and(w2, A, B);
    and(w3, A, Cin);
    and(w4, B, Cin);
    or(Cout, w2, w3, w4);
endmodule
```

Defining the function and its inputs and outputs

If the value of Cin equals 0, then it adds A and B and gives the answer to the wires which will be calculated together

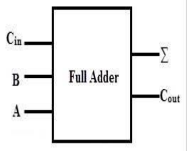
But if the value of Cin equals 1, then it calculates the 2's complement of B and adds the answer with A



Simulation Waveforms

Simulation mode: Functional

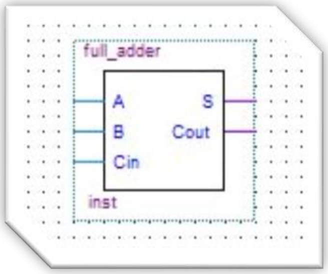
Master Time Bar: 0 ps Pointer: 3.9 ns Interval: 3.9 ns Start: End:



Block Diagram of Full Adder

C_{in}	B	A	$\sum$	C_{out}
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

Truth Table



4-bit adder-sub

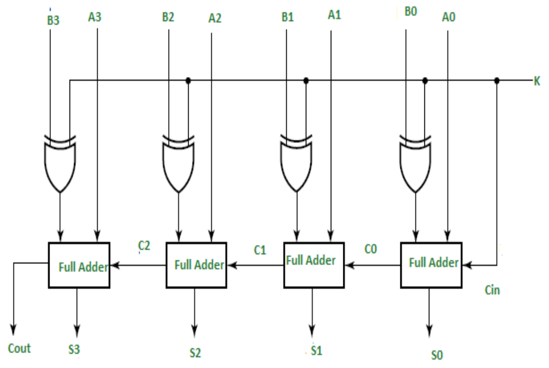
Consists of:

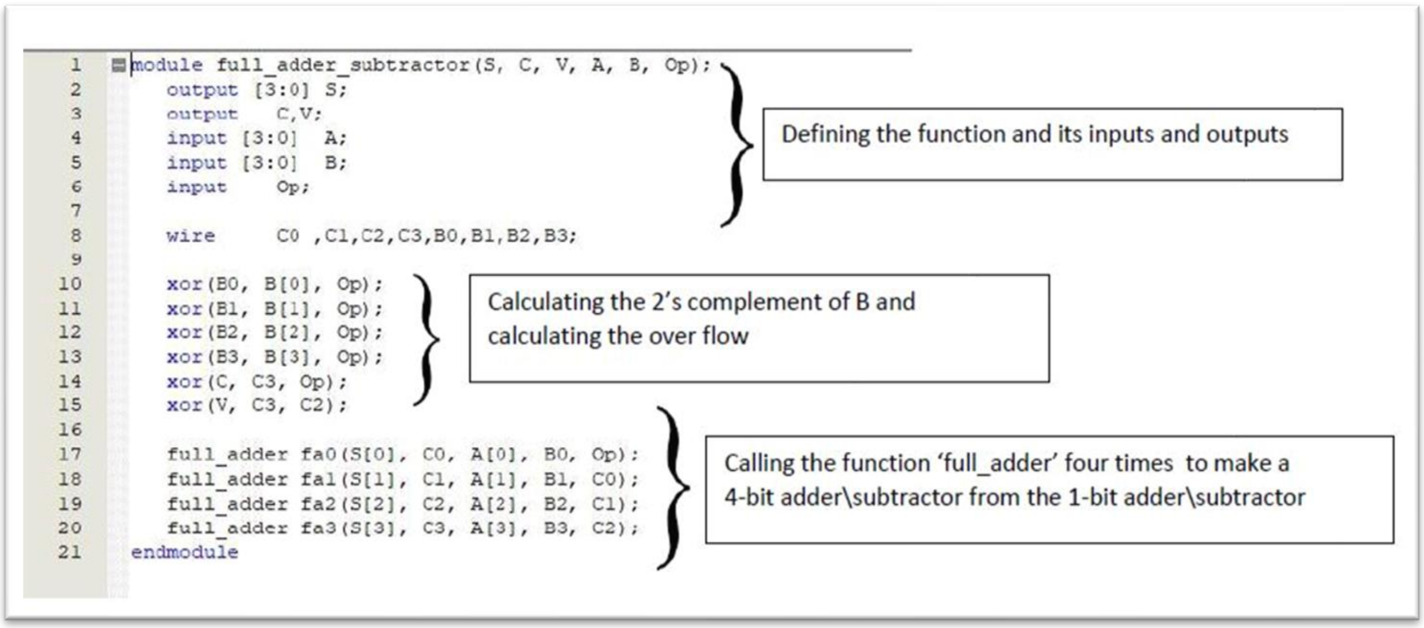
1-4*1 bit adder-sub.

2-Inputs: A , B , Op .

3-Gates: 6 XOR gates.

4-Outputs: S , C , V (over flow) .



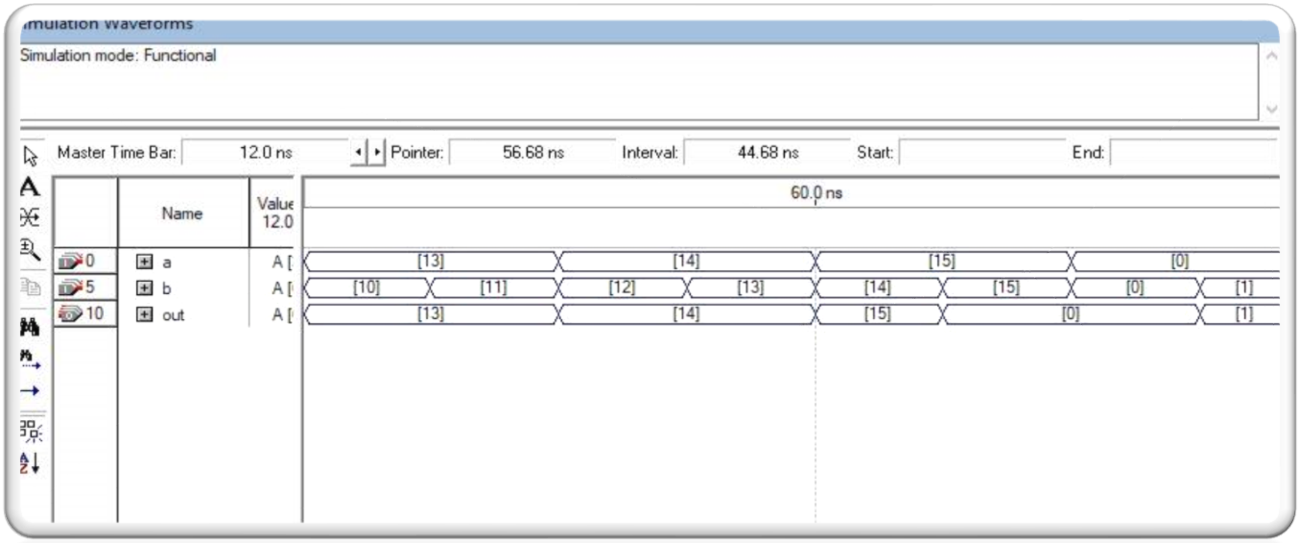


```
module full_adder_subtractor(S, C, V, A, B, Op);
   output [3:0] S;
   output    C,V;
   input [3:0]  A;
   input [3:0]  B;
   input     Op;

   wire      C0 ,C1,C2,C3,B0,B1,B2,B3;

   xor(B0, B[0], Op);
   xor(B1, B[1], Op);
   xor(B2, B[2], Op);
   xor(B3, B[3], Op);
   xor(C, C3, Op);
   xor(V, C3, C2);

   full_adder fa0(S[0], C0, A[0], B0, Op);
   full_adder fa1(S[1], C1, A[1], B1, C0);
   full_adder fa2(S[2], C2, A[2], B2, C1);
   full_adder fa3(S[3], C3, A[3], B3, C2);
endmodule
```

Defining the function and its inputs and outputs

Calculating the 2's complement of B and calculating the over flow

Calling the function 'full_adder' four times to make a 4-bit adder\subtractor from the 1-bit adder\subtractor

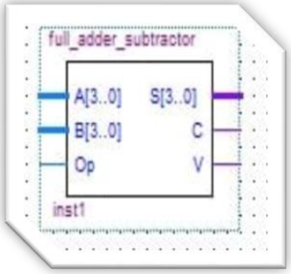


Simulation Waveforms

Simulation mode: Functional

Master Time Bar: 12.0 ns   Pointer: 56.68 ns   Interval: 44.68 ns   Start: End:

	Name	Value 12.0	60.0 ns
0	a	A [	[13] [14] [15] [0]
5	b	A [	[10] [11] [12] [13] [14] [15] [0] [1]
10	out	A [	[13] [14] [15] [0] [1]



BCD adder-sub

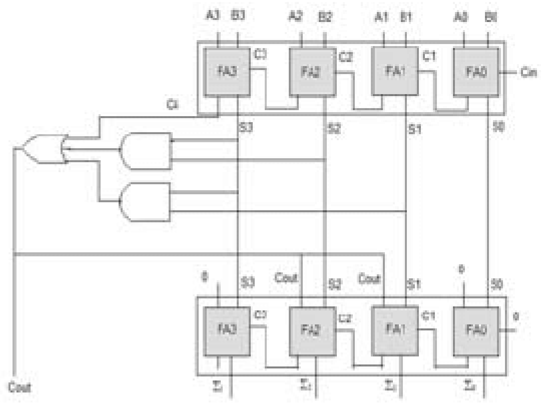
Consists of:

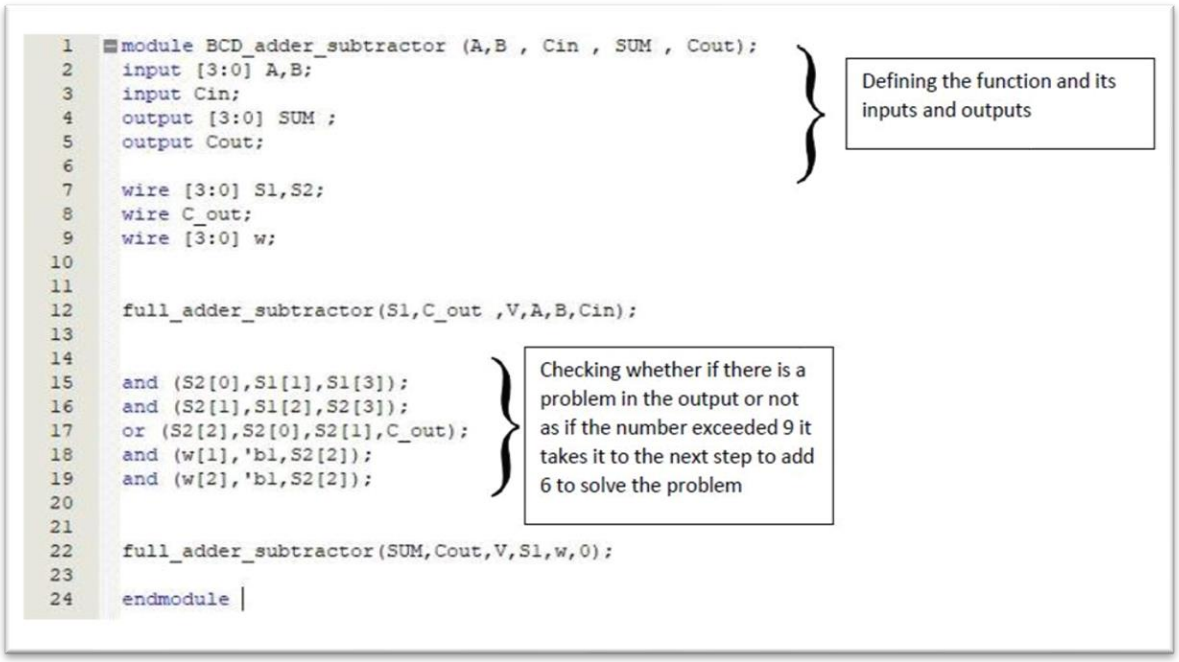
1-2*4-bit adder-sub.

2-Inputs: A , B , Cin .

3-Gates: 4 AND gates and 1 OR gate.

4-Outputs: SUM , Cout.

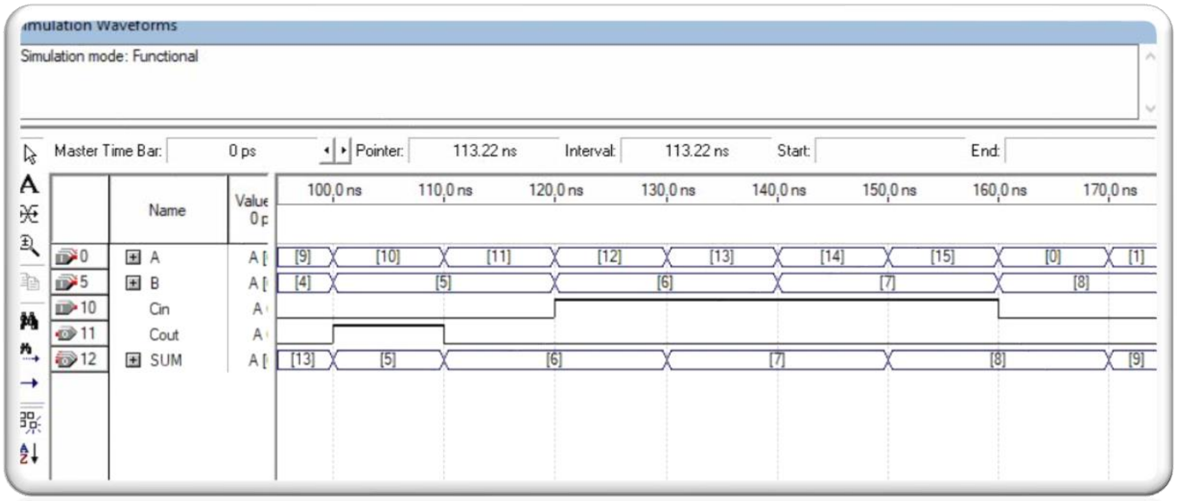




```
module BCD_adder_subtractor (A,B , Cin , SUM , Cout);
input [3:0] A,B;
input Cin;
output [3:0] SUM ;
output Cout;

wire [3:0] S1,S2;
wire C_out;
wire [3:0] w;


full_adder_subtractor(S1,C_out ,V,A,B,Cin);


and (S2[0],S1[1],S1[3]);
and (S2[1],S1[2],S2[3]);
or (S2[2],S2[0],S2[1],C_out);
and (w[1],'b1,S2[2]);
and (w[2],'b1,S2[2]);


full_adder_subtractor(SUM,Cout,V,S1,w,0);

endmodule
```

Defining the function and its inputs and outputs

Checking whether if there is a problem in the output or not as if the number exceeded 9 it takes it to the next step to add 6 to solve the problem

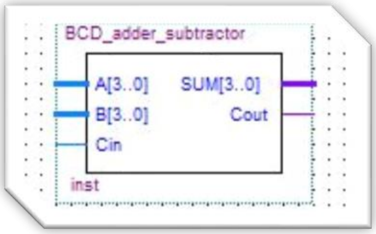


Simulation Waveforms

Simulation mode: Functional

Master Time Bar: 0 ps   Pointer: 113.22 ns   Interval: 113.22 ns   Start: End:

```
      Name    Value  100.0 ns  110.0 ns  120.0 ns  130.0 ns  140.0 ns  150.0 ns  160.0 ns  170.0 ns
0     A       A [    [9]  [10]  [11]  [12]  [13]  [14]  [15]  [0]  [1]
5     B       A [    [4]  [5]  [6]  [7]  [8]
10    Cin     A
11    Cout    A
12    SUM     A [    [13]  [5]  [6]  [7]  [8]  [9]
```



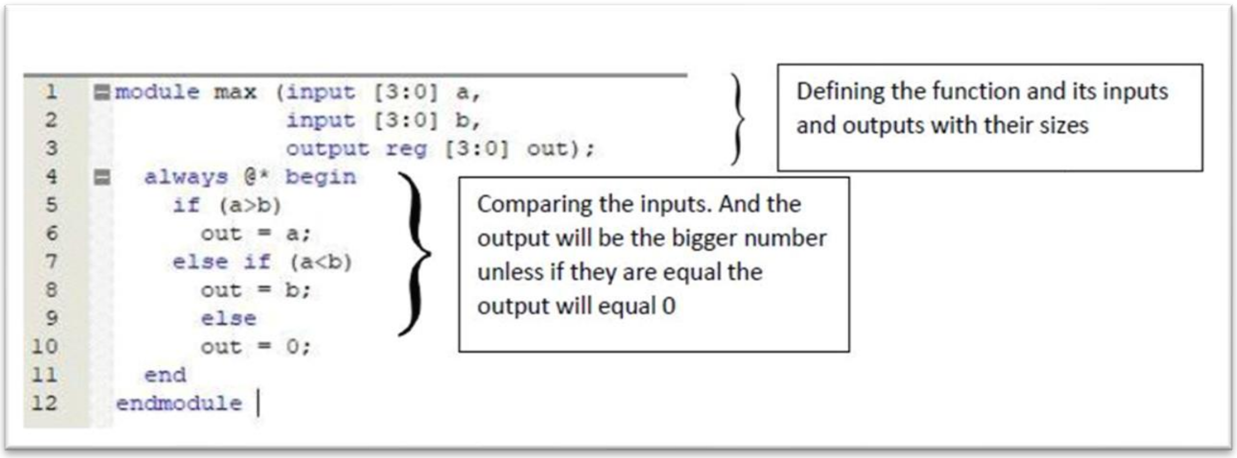
MAX

Consists of:

1-Inputs: A , B .

2- If statement .

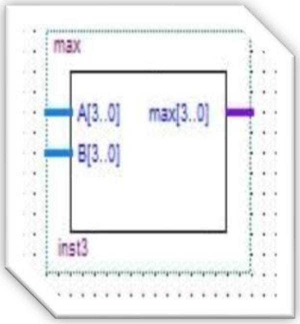
3-Outputs: Max .

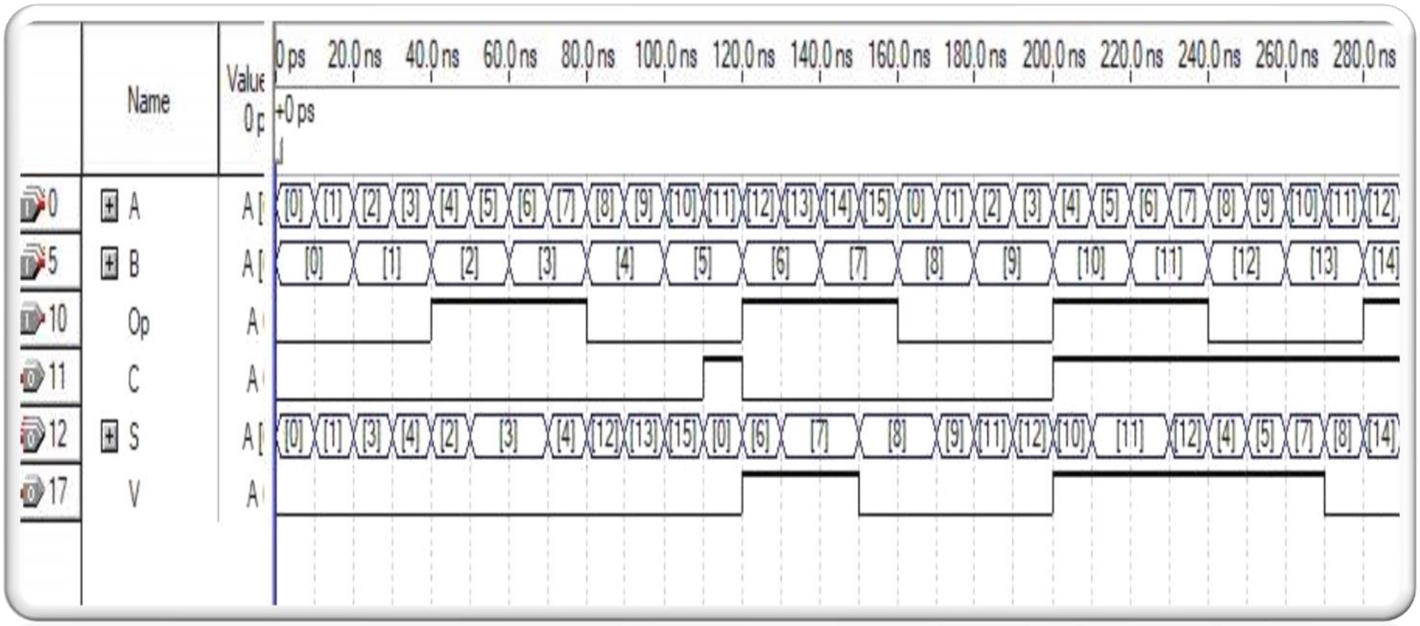


```
module max (input [3:0] a,
            input [3:0] b,
            output reg [3:0] out);
  always @* begin
    if (a>b)
      out = a;
    else if (a<b)
      out = b;
    else
      out = 0;
  end
endmodule
```

Defining the function and its inputs and outputs with their sizes

Comparing the inputs. And the output will be the bigger number unless if they are equal the output will equal 0





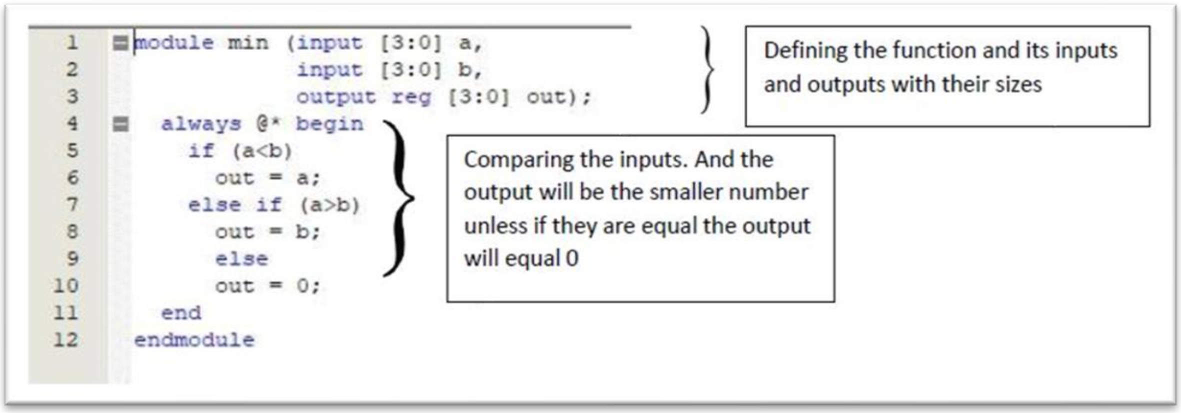
M I N

Consists of:

1-Inputs: a , b .

2-If statement .

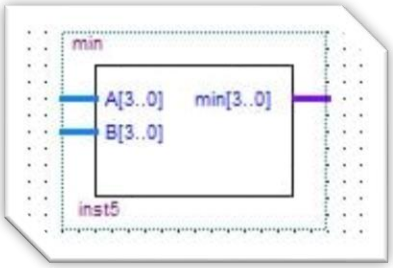
3-Outputs: Min .

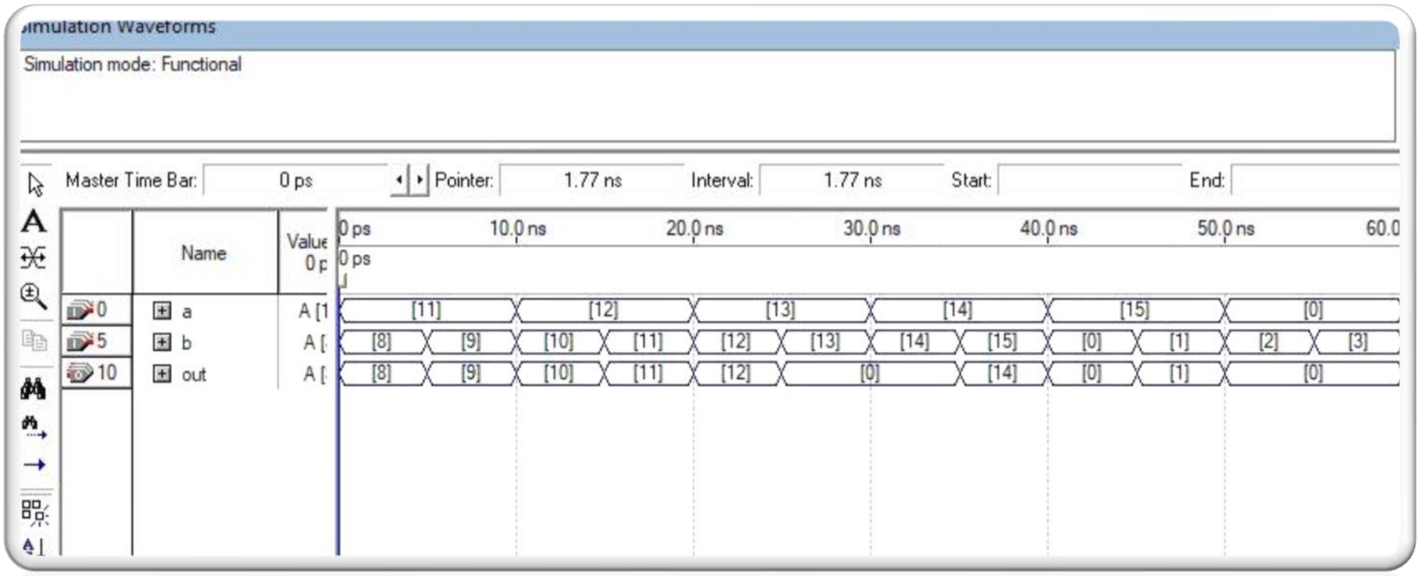


```
module min (input [3:0] a,
            input [3:0] b,
            output reg [3:0] out);
  always @* begin
    if (a<b)
      out = a;
    else if (a>b)
      out = b;
      else
      out = 0;
  end
endmodule
```

Defining the function and its inputs and outputs with their sizes

Comparing the inputs. And the output will be the smaller number unless if they are equal the output will equal 0





Simulation Waveforms

Simulation mode: Functional

Master Time Bar: 0 ps Pointer: 1.77 ns Interval: 1.77 ns Start: End:

```
     Name   Value  0 ps   10.0 ns   20.0 ns   30.0 ns   40.0 ns   50.0 ns   60.0
0    a      A [1   [11]  [12]  [13]  [14]  [15]  [0]
5    b      A [    [8] [9] [10] [11] [12] [13] [14] [15] [0] [1] [2] [3]
10   out    A [    [8] [9] [10] [11] [12] [0] [14] [0] [1] [0]
```

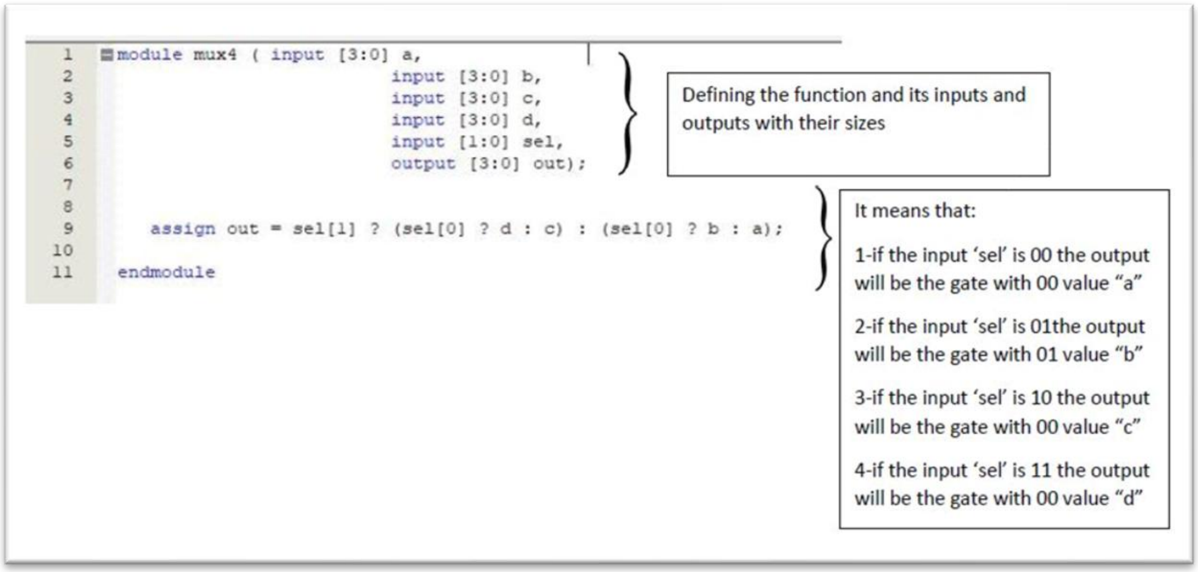
MUX

Consists of:

1-Inputs: b , c , d , sel .

2-If statement .

3-Outputs: out.



```
module mux4 ( input [3:0] a,
              input [3:0] b,
              input [3:0] c,
              input [3:0] d,
              input [1:0] sel,
              output [3:0] out);


    assign out = sel[1] ? (sel[0] ? d : c) : (sel[0] ? b : a);

endmodule
```

Defining the function and its inputs and outputs with their sizes

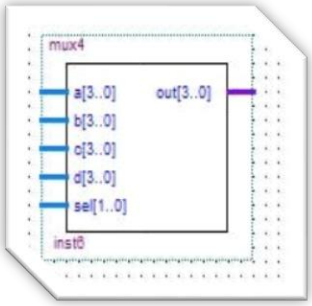
It means that:

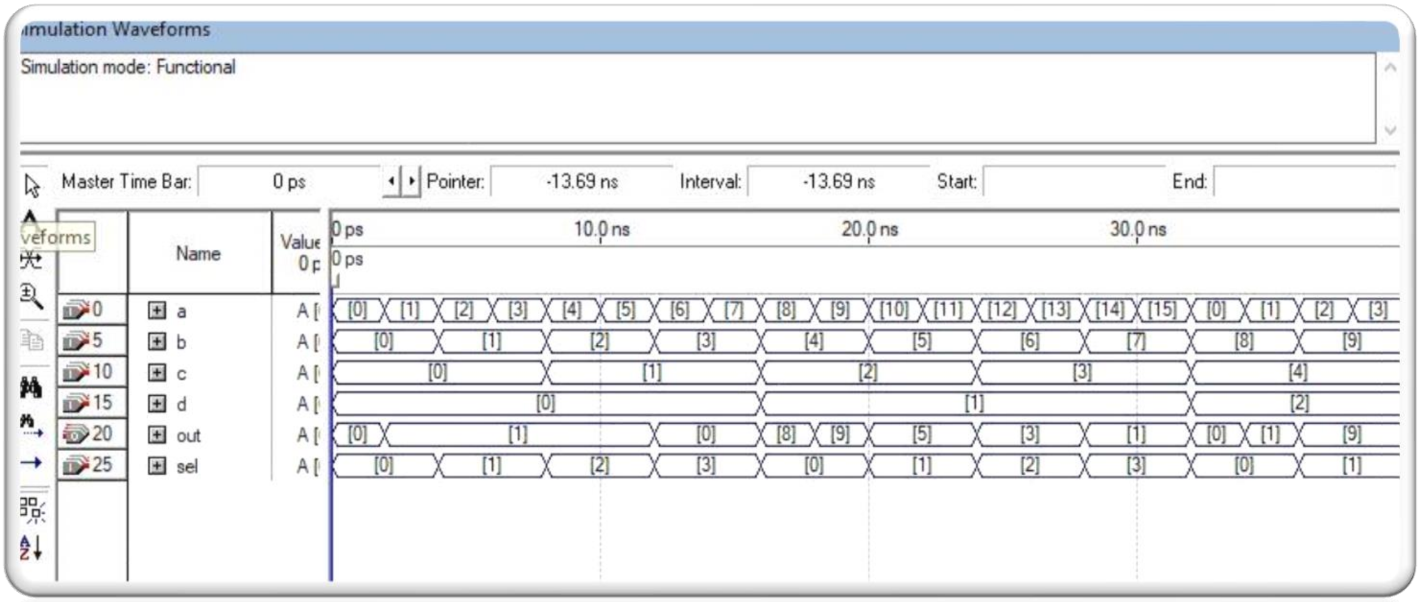
1-if the input 'sel' is 00 the output will be the gate with 00 value "a"

2-if the input 'sel' is 01the output will be the gate with 01 value "b"

3-if the input 'sel' is 10 the output will be the gate with 00 value "c"

4-if the input 'sel' is 11 the output will be the gate with 00 value "d"





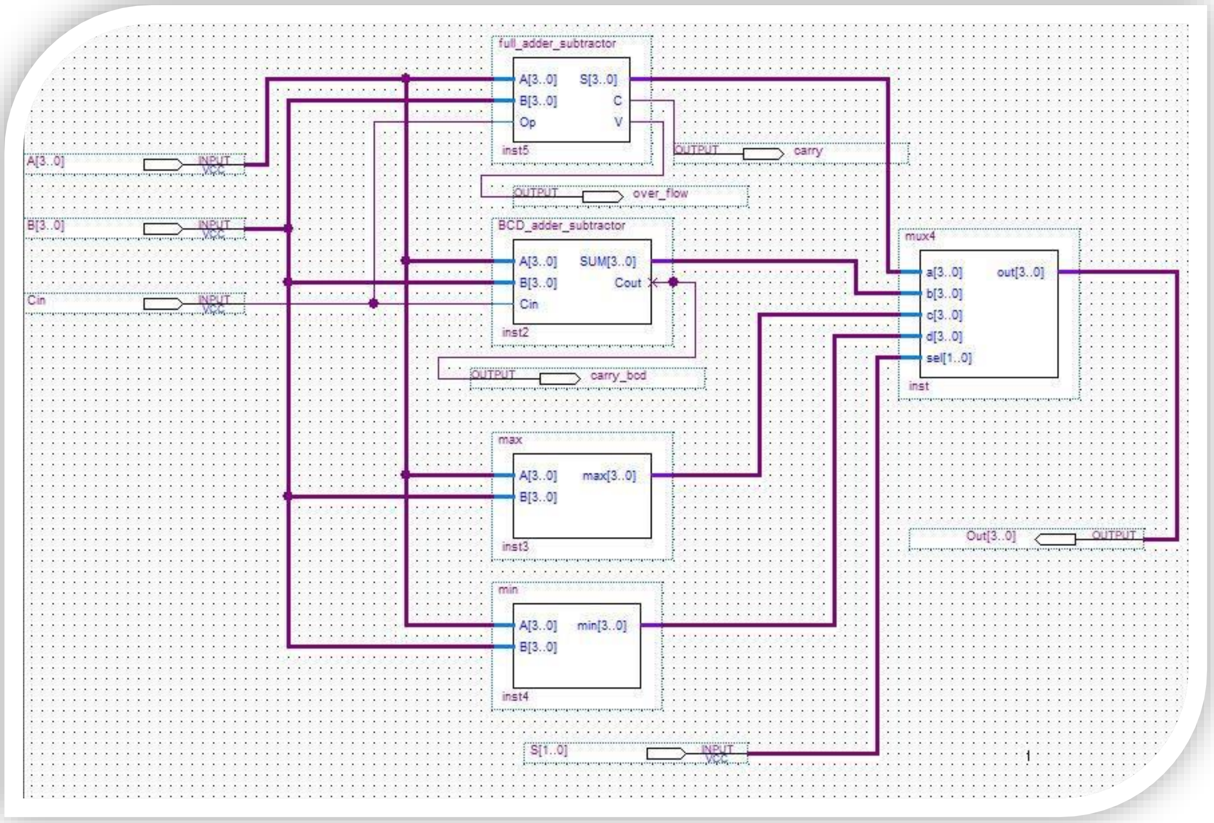
Simulation Waveforms

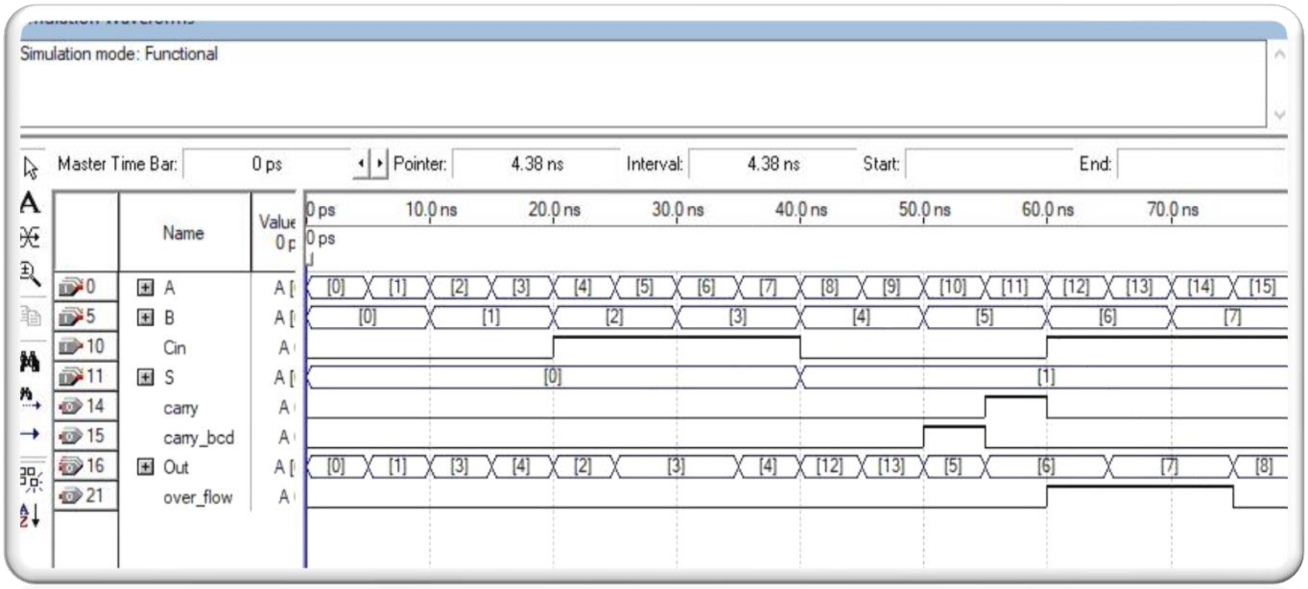
Simulation mode: Functional

Master Time Bar: 0 ps   Pointer: -13.69 ns   Interval: -13.69 ns   Start: End:

veforms
Name
Value 0 p
0 ps
10.0 ns
20.0 ns
30.0 ns
0 ps
0 a A [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [0] [1] [2] [3]
5 b A [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
10 c A [0] [1] [2] [3] [4]
15 d A [0] [1] [2]
20 out A [0] [1] [0] [8] [9] [5] [3] [1] [0] [1] [9]
25 sel A [0] [1] [2] [3] [0] [1] [2] [3] [0] [1]

THE COMPLETED CIRCUIT




Simulation mode: Functional

Master Time Bar: 0 ps Pointer: 4.38 ns Interval: 4.38 ns Start: End:

0 ps
10.0 ns
20.0 ns
30.0 ns
40.0 ns
50.0 ns
60.0 ns
70.0 ns
Name
Value
0 A [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]
5 B [0] [1] [2] [3] [4] [5] [6] [7]
10 Cin
11 S [0] [1]
14 carry
15 carry_bcd
16 Out [0] [1] [3] [4] [2] [3] [4] [12] [13] [5] [6] [7] [8]
21 over_flow

THE TEAMWORK

Mohammad: {1-bit + 4-bit} adder/subtractor (coding) + BCD adder/subtractor (simulating + testing) + min (coding) + max (simulating + testing) + mux(coding)

Yazan: {1-bit + 4-bit} adder/subtractor (simulating + testing) + BCD adder/subtractor(coding) + max (coding) + min (simulating + testing) + mux (simulating + testing)

Putting all the circuit together and testing it was made by both of us while using zoom.

The report was written using a sharable document done by both of us.