PRE-WORK 2

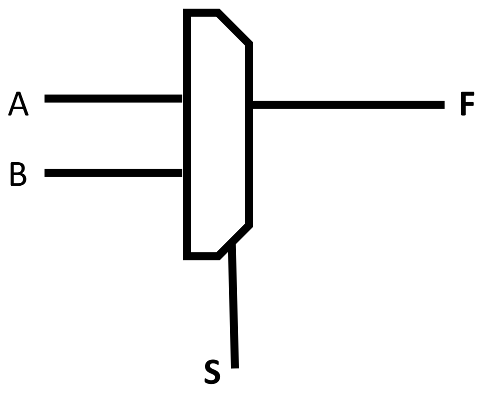
2.2 PRELIMINARY WORK

2.2.2 2-TO-1 MULTIPLEXER

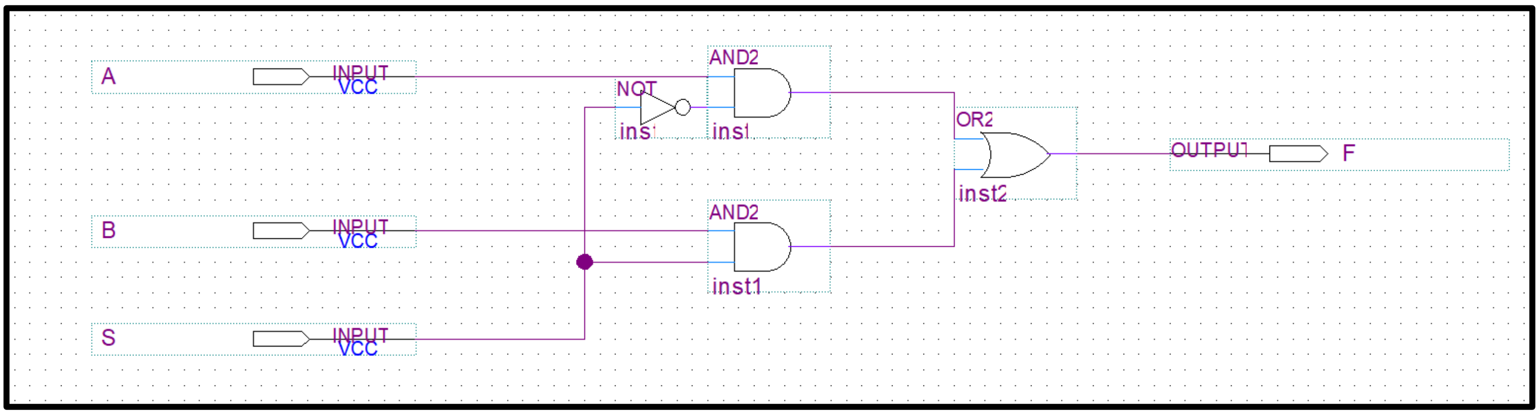
Condensed truth table:

S	**F**
0	A[3:0]
1	B[3:0]

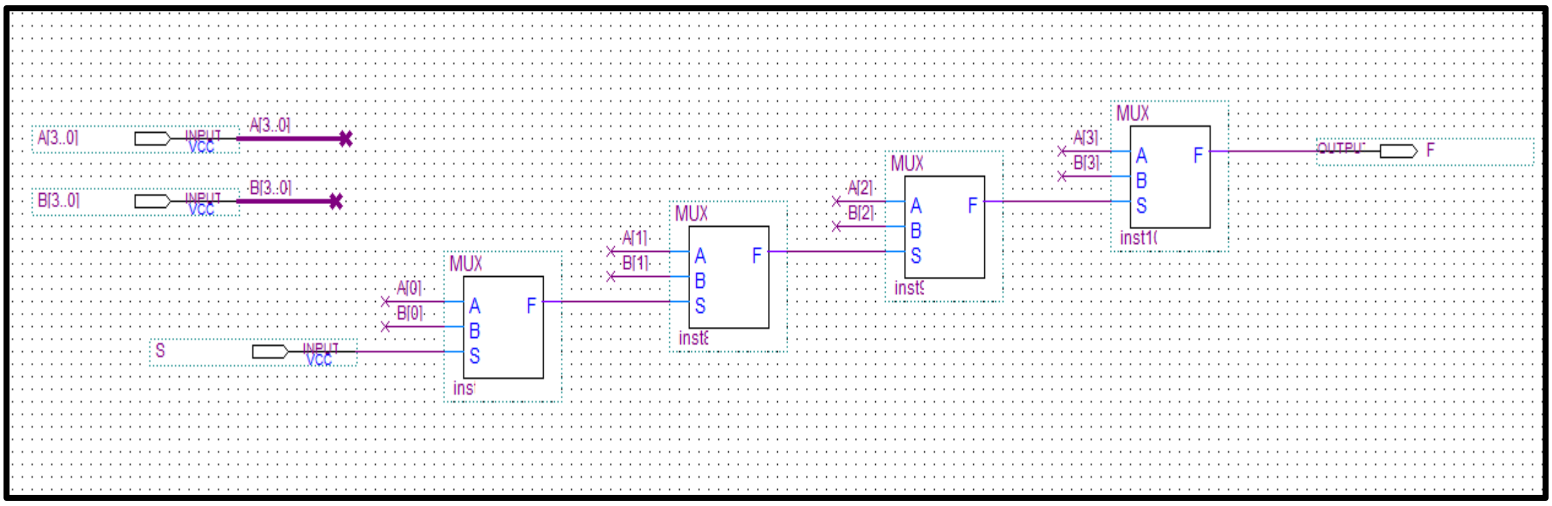
The schematic:



The schematic representation in 1-bit (MUX):



After converting to a symbol (in 4-bit) (multiplexer):

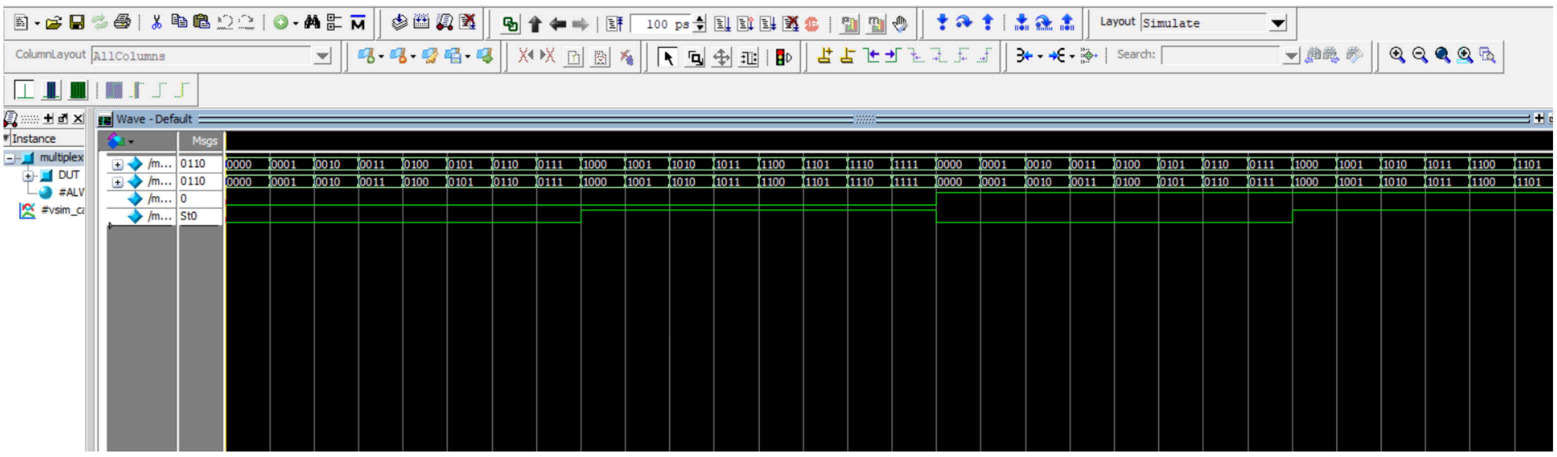


Verilog code (in procedural format):

```
module multiplexer(A, B, S, F);
        parameter size = 4;
        input [size-1:0] A, B;
        input S;
        output F;
        wire [size-2:0] w;
        genvar i;
        generate
        for (i=0 ; i<size; i=i+1) begin: multiplexer
                if(i==0)
                        MUX U1(A[i], B[i], S, w[i]);
                else if (i==size-1)
                        MUX U1(A[i], B[i], w[i-1], F);
                else
                        MUX U1(A[i], B[i], w[i-1], w[i]);
        end
        endgenerate
        endmodule
```

And Verilog Code For the MUX :

```
module MUX(A, B, S, F);
         input A, B, S;
          output F;
        wire w1, w2;
         and G1(w1, A, !S);
        and G2(w2, B, S);
         or G3(F, w1, w2);
endmodule
```



The simulation of the multiplexer in Modelsim

2.2.3 4-BIT ADDER/SUBTRACTOR

1)

Cin	**A**	**B**	**Cout**	**SUM**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\left.\vphantom{\begin{matrix}0\\0\\0\\0\\0\\0\\0\\0\end{matrix}}\right\}$$

SUM = Cin⊕A⊕B

Cout = (Cin'+B')(Cin'+A')(A'+B') = A.B + Cin(A⊕B)

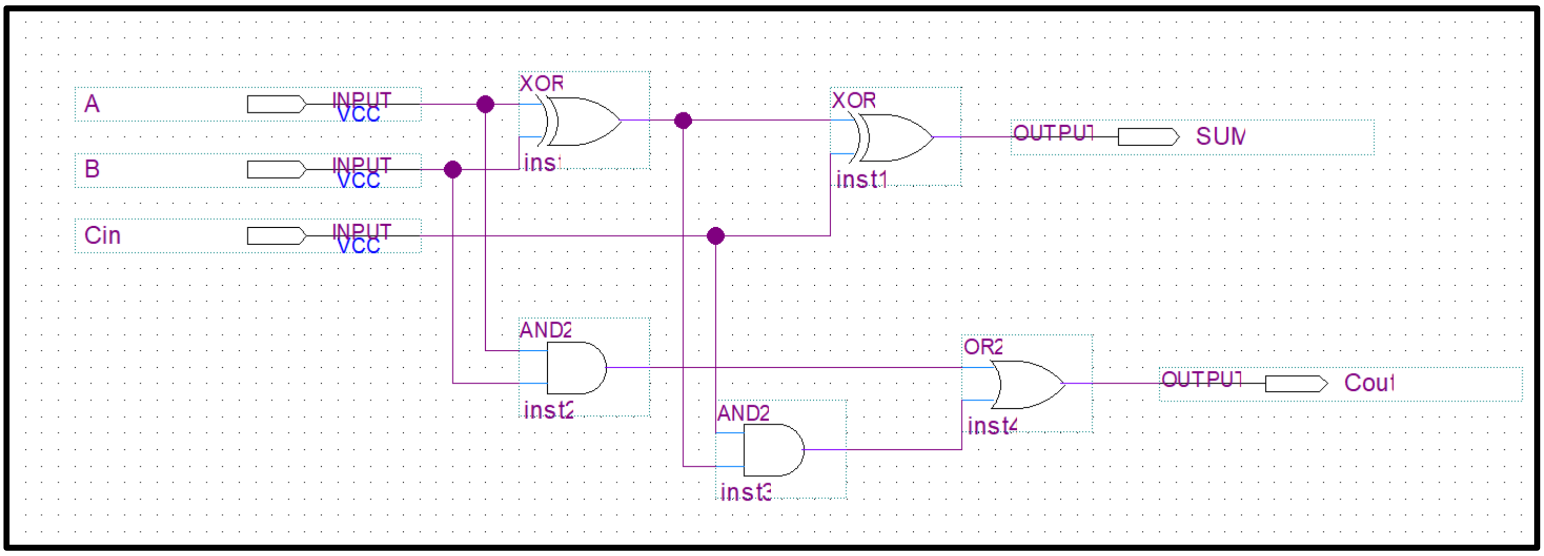
2)

Verilog code to implement the full adder:

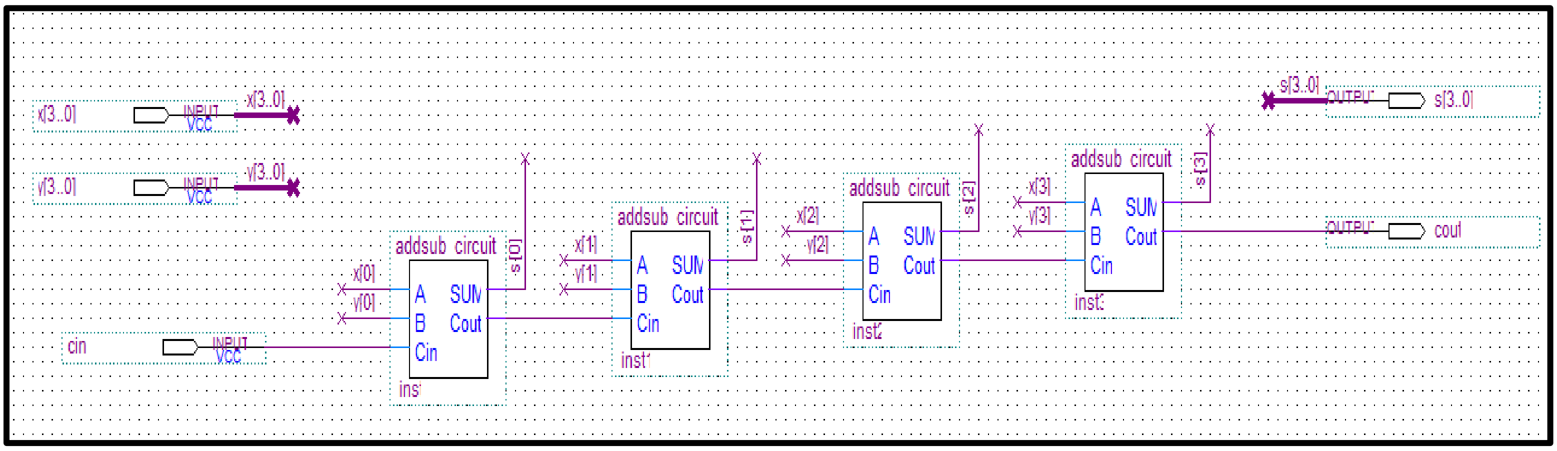
```
module add_sub_circuit(A, B, Cin, Cout, SUM);
        input A, B, Cin;
        output Cout, SUM;
        wire w1, w2, w3;

        xor G1(w1, A, B);
        and G2(w2, A, B);
        and G3(w3, Cin, w1);
         xor G4(SUM, w1, Cin);
         or G5(Cout, w2, w3);
endmodule
```

3)

The schematic for the full adder in 1-bit:



The schematic representation of 4-bit Ripple-Carry-Adder:



4)

Verilog code to implement 4-bit Ripple-Carry-Adder:

```
module add_sub (x, y, cin, cout, s);
        parameter size = 4 ;

        input [size-1:0] x, y;
        input cin;
        output [size-1:0] s;
        output cout;
        wire [size-2:0] w;

        genvar i;
```

```
generate
for (i=0 ; i<size; i=i+1) begin:add_sub
if (i==0)
            add_sub_circuit U1(x[i], y[i], cin, s[i], w[i]);
else if(i==size-1)
            add_sub_circuit U1(x[i], y[i], w[i-1], s[i], cout);
 else
            add_sub_circuit U1(x[i], y[i], w[i-1], s[i], w[i]);

end
endgenerate
endmodule
```

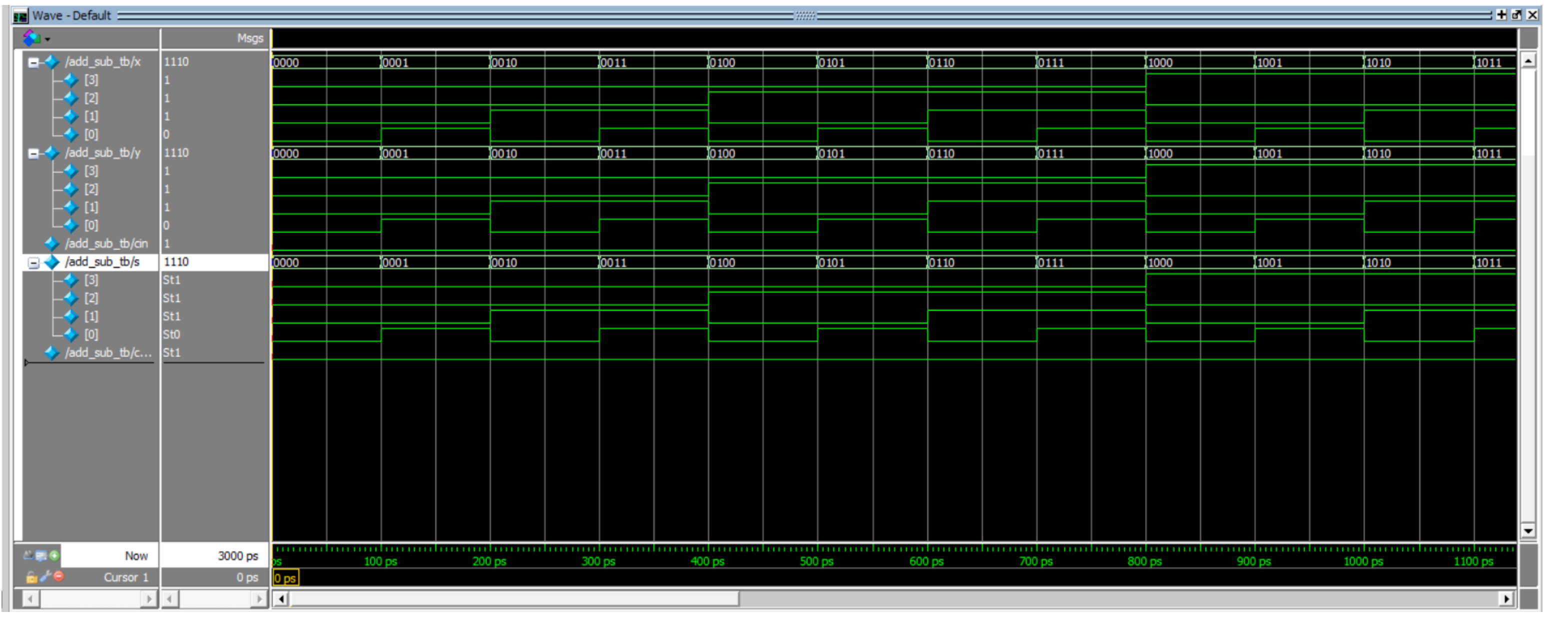
5)

OP	**A**	**B**	**F**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

After simplifying from the K-map; **F = A + B + OP**

7)

Modelsim simulation for the first adder/subtractor circuit (with Cin):



Verilog code for the new circuit (with OP-for the symbol):

```
module OP_signal(B, OP, F);
        input B, OP;
         output F;
        wire w1, w2;

       and (w1, B, !OP);
       and (w2, !B, OP);
       or (F, w1, w2);

endmodule
```

Verilog code for the new circuit (with OP):

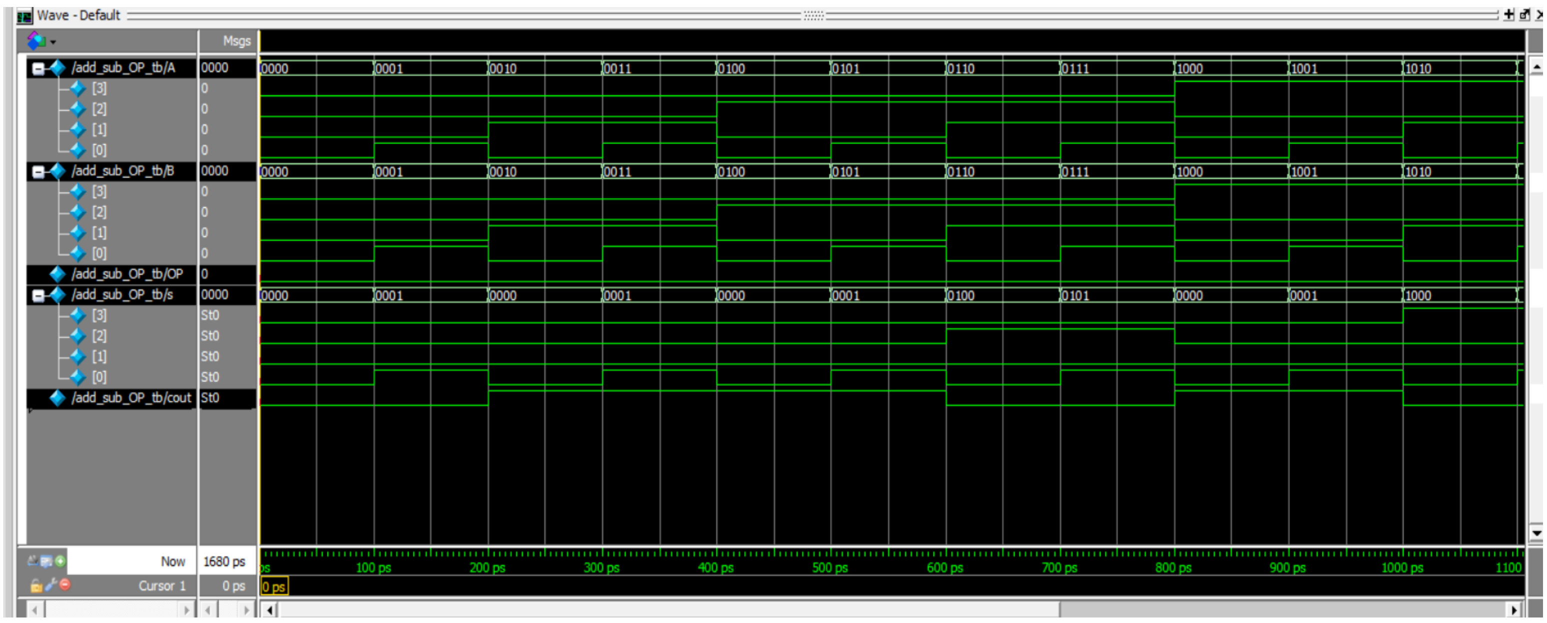
```
module add_sub_OP (A, B, OP, cout, s);
       parameter size = 4 ;

       input [size-1:0] A, B;
       input OP;
       output [size-1:0] s;
       output cout;
       wire [size-1:0] w;

       genvar i;
       generate
       for (i=0 ; i<size; i=i+1) begin:add_sub_OP
              if (i==0) begin
                     OP_signal U2(B, OP, w[i]);
                     add_sub_circuit U1(A[i], w[i], OP, s[i], w[i+1]);
                     end
              else if(i==size-1)
                     add_sub_circuit U1(A[i], w[i], OP, s[i], cout);
              else
                     add_sub_circuit U1(A[i], w[i], OP, s[i], w[i+1]);
       end
       endgenerate
       endmodule
```

Testbench for the new circuit with all possibilities:

```
module add_sub_OP_tb();
      parameter size = 4;
      reg [size-1:0] A, B;
      reg  OP;
      wire [size-1:0] s;
      wire cout;

      add_sub_OP DUT (A, B, OP, cout, s);

      always
      begin
      A=4'b0000; B=4'b0000; OP=1'b0; #100;
   A=4'b0001; B=4'b0001; OP=1'b0; #100;
   A=4'b0010; B=4'b0010; OP=1'b0; #100;
   A=4'b0011; B=4'b0011; OP=1'b0; #100;
   A=4'b0100; B=4'b0100; OP=1'b0; #100;
   A=4'b0101; B=4'b0101; OP=1'b0; #100;
   A=4'b0110; B=4'b0110; OP=1'b0; #100;
   A=4'b0111; B=4'b0111; OP=1'b0; #100;
   A=4'b1000; B=4'b1000; OP=1'b0; #100;
   A=4'b1001; B=4'b1001; OP=1'b0; #100;
   A=4'b1010; B=4'b1010; OP=1'b0; #100;
   A=4'b1011; B=4'b1011; OP=1'b0; #100;
   A=4'b1100; B=4'b1100; OP=1'b0; #100;
   A=4'b1101; B=4'b1101; OP=1'b0; #100;
   A=4'b1110; B=4'b1110; OP=1'b0; #100;
   A=4'b1111; B=4'b1111; OP=1'b0; #100;

   A=4'b0000; B=4'b0000; OP=1'b0; #100;
   A=4'b0001; B=4'b0001; OP=1'b0; #100;
   A=4'b0010; B=4'b0010; OP=1'b0; #100;
   A=4'b0011; B=4'b0011; OP=1'b0; #100;
   A=4'b0100; B=4'b0100; OP=1'b0; #100;
   A=4'b0101; B=4'b0101; OP=1'b0; #100;
   A=4'b0110; B=4'b0110; OP=1'b0; #100;
   A=4'b0111; B=4'b0111; OP=1'b0; #100;
   A=4'b1000; B=4'b1000; OP=1'b0; #100;
   A=4'b1001; B=4'b1001; OP=1'b0; #100;
   A=4'b1010; B=4'b1010; OP=1'b0; #100;
```

```
    A=4'b1011; B=4'b1011; OP=1'b0; #100;
    A=4'b1100; B=4'b1100; OP=1'b0; #100;
    A=4'b1101; B=4'b1101; OP=1'b0; #100;
    A=4'b1110; B=4'b1110; OP=1'b0; #100;
    A=4'b1111; B=4'b1111; OP=1'b0; #100;
end
endmodule
```

The simulation for the new circuit (with OP):



2.2.4 4-BIT COMPARATOR

1)

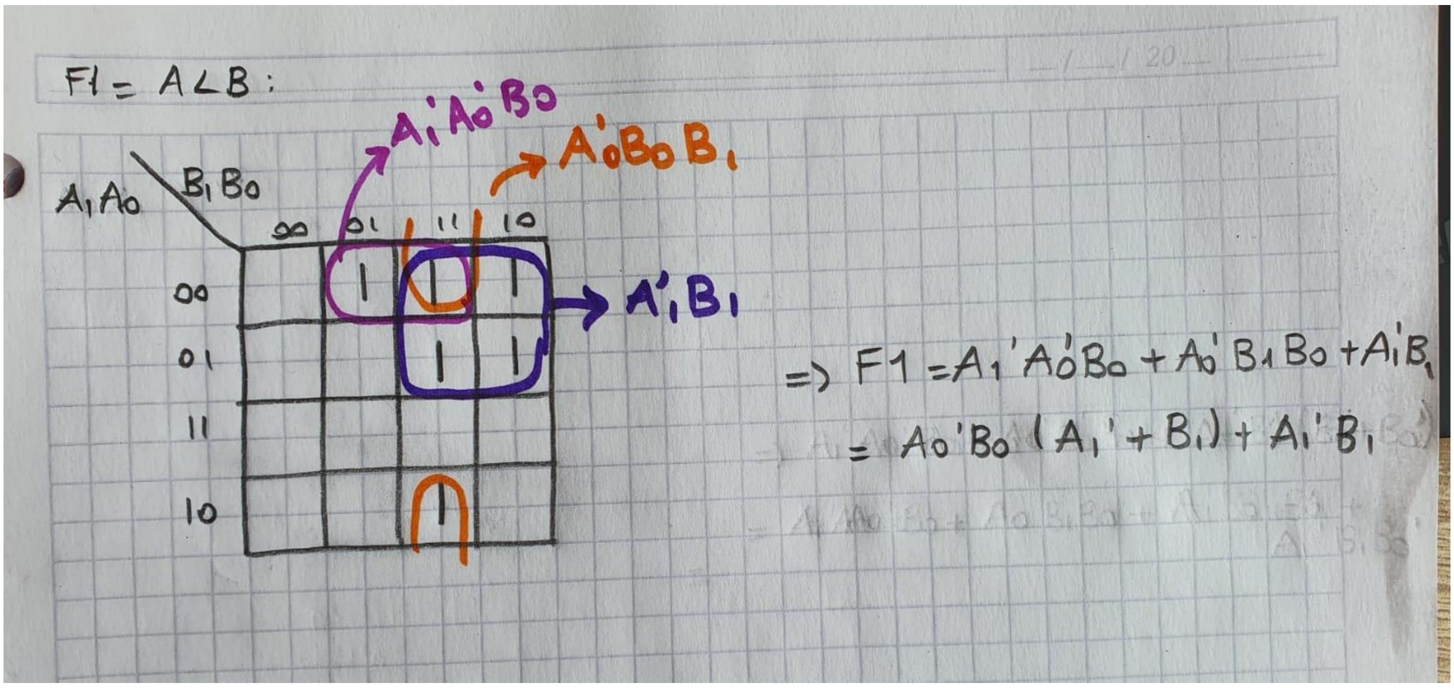
The truth table:

A1	A0	B1	B0	A<B	A>B	A=B
0	0	0	0	0	0	1
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	0	0	1
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	0

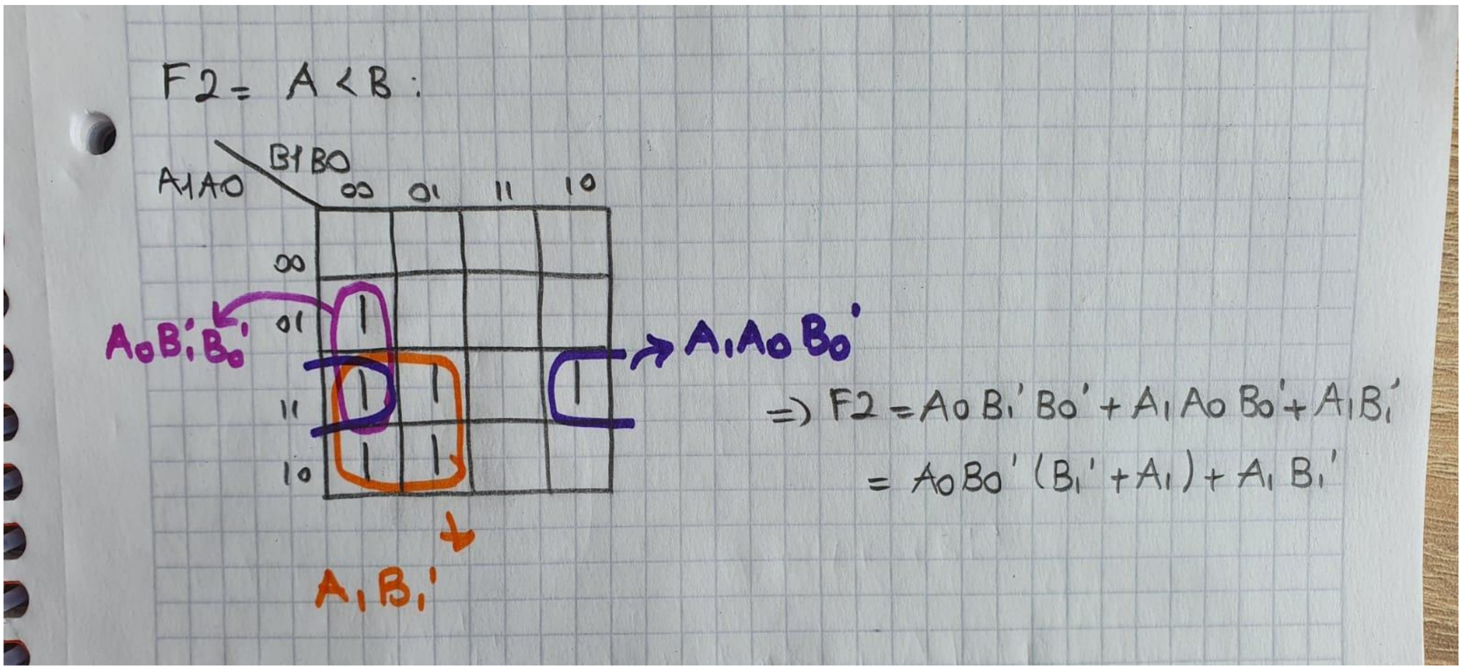
1	0	1	0	0	0	1
1	0	1	1	1	0	0
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	1	0
1	1	1	1	0	0	1

K-Maps:

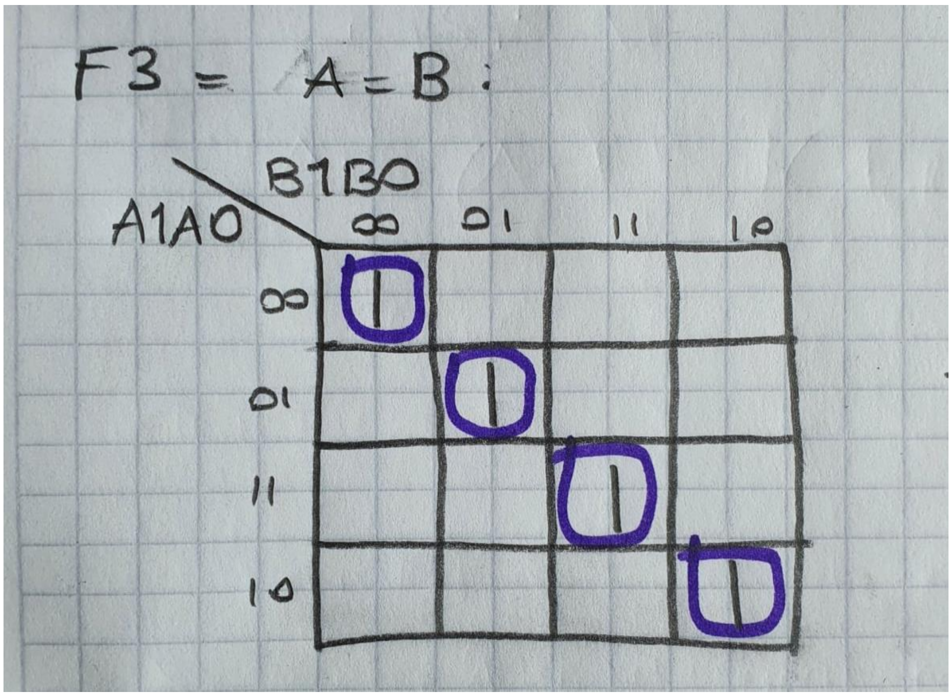
For the output F1 = A<B:



For F2 = A>B:



For F3 = A=B:



$$
\begin{aligned}
F3 &= A1'A0'B1'B0' + A1'A0B1'B0 + A1A0B1B0 + A1A0'B1B0' \\
&= A0'B0'(A1'B1' + A1B1) + A0B0(A1'B1' + A1B1) \\
&= (A1'B1' + A1B1)(A0'B0' + A0B0) \\
&= (A1 \oplus B1)'(A0 \oplus B0)'
\end{aligned}
$$

2)

```
module comparator(A, B, F1, F2, F3);
       parameter size = 4;
        input [size-1:0] A, B;
        output F1, F2, F3;
       wire w1, w2, w3, w4, w5, w6;

        or (F1, w1, w4);
       or (F2, w2, w5);
       or (F3, w3, w6);

        genvar i;
         generate
        for (i=0 ; i<size ; i=i+1) begin: comparator
                   if (i==0 || i==1)
                               comparator_circuit U1(A[i], B[i], w4, w5, w6);
                    else
                               comparator_circuit U1(A[i], B[i], w1, w2, w3);
       end
  endgenerate
```

```
endmodule
```

3)

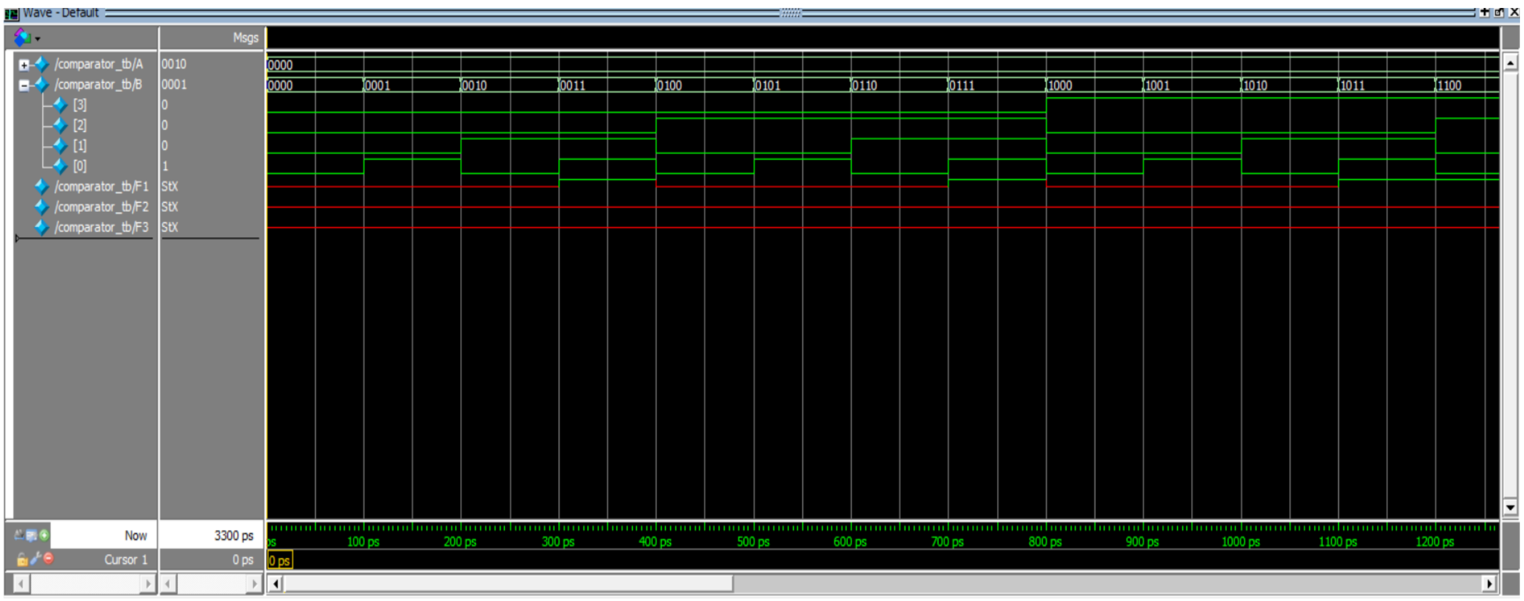
```
module comparator_tb();
        parameter size = 4;
        reg [size-1:0] A, B;
       wire F1, F2, F3;

comparator DUT(A, B, F1, F2, F3);

  always
  begin
    A=4'b0000; B=4'b0000; #100;
    A=4'b0000; B=4'b0001; #100;
    A=4'b0000; B=4'b0010; #100;
    A=4'b0000; B=4'b0011; #100;
    A=4'b0000; B=4'b0100; #100;
    A=4'b0000; B=4'b0101; #100;
    A=4'b0000; B=4'b0110; #100;
    A=4'b0000; B=4'b0111; #100;
    A=4'b0000; B=4'b1000; #100;
    A=4'b0000; B=4'b1001; #100;
    A=4'b0000; B=4'b1010; #100;
    A=4'b0000; B=4'b1011; #100;
    A=4'b0000; B=4'b1100; #100;
    A=4'b0000; B=4'b1101; #100;
    A=4'b0000; B=4'b1110; #100;
    A=4'b0000; B=4'b1111; #100;

    A=4'b0001; B=4'b0000; #100;
    A=4'b0001; B=4'b0001; #100;
    A=4'b0001; B=4'b0010; #100;
    A=4'b0001; B=4'b0011; #100;
    A=4'b0001; B=4'b0100; #100;
    A=4'b0001; B=4'b0101; #100;
    A=4'b0001; B=4'b0110; #100;
    A=4'b0001; B=4'b0111; #100;
    A=4'b0001; B=4'b1000; #100;
    A=4'b0001; B=4'b1001; #100;
    A=4'b0001; B=4'b1010; #100;
    A=4'b0001; B=4'b1011; #100;
```

```
A=4'b0001; B=4'b1100; #100;
A=4'b0001; B=4'b1101; #100;
A=4'b0001; B=4'b1110; #100;
A=4'b0001; B=4'b1111; #100;

A=4'b0010; B=4'b0000; #100;
A=4'b0010; B=4'b0001; #100;
A=4'b0010; B=4'b0010; #100;
A=4'b0010; B=4'b0011; #100;
A=4'b0010; B=4'b0100; #100;
A=4'b0010; B=4'b0101; #100;
A=4'b0010; B=4'b0110; #100;
A=4'b0010; B=4'b0111; #100;
A=4'b0010; B=4'b1000; #100;
A=4'b0010; B=4'b1001; #100;
A=4'b0010; B=4'b1010; #100;
A=4'b0010; B=4'b1011; #100;
A=4'b0010; B=4'b1100; #100;
A=4'b0010; B=4'b1101; #100;
A=4'b0010; B=4'b1110; #100;
A=4'b0010; B=4'b1111; #100;

A=4'b0011; B=4'b0000; #100;
A=4'b0011; B=4'b0001; #100;
A=4'b0011; B=4'b0010; #100;
A=4'b0011; B=4'b0011; #100;
A=4'b0011; B=4'b0100; #100;
A=4'b0011; B=4'b0101; #100;
A=4'b0011; B=4'b0110; #100;
A=4'b0011; B=4'b0111; #100;
A=4'b0011; B=4'b1000; #100;
A=4'b0011; B=4'b1001; #100;
A=4'b0011; B=4'b1010; #100;
A=4'b0011; B=4'b1011; #100;
A=4'b0011; B=4'b1100; #100;
A=4'b0011; B=4'b1101; #100;
A=4'b0011; B=4'b1110; #100;
A=4'b0011; B=4'b1111; #100;

A=4'b0101; B=4'b0000; #100;
A=4'b0101; B=4'b0001; #100;
```

```
A=4'b0101; B=4'b0010; #100;
A=4'b0101; B=4'b0011; #100;
A=4'b0101; B=4'b0100; #100;
A=4'b0101; B=4'b0101; #100;
A=4'b0101; B=4'b0110; #100;
A=4'b0101; B=4'b0111; #100;
A=4'b0101; B=4'b1000; #100;
A=4'b0101; B=4'b1001; #100;
A=4'b0101; B=4'b1010; #100;
A=4'b0101; B=4'b1011; #100;
A=4'b0101; B=4'b1100; #100;
A=4'b0101; B=4'b1101; #100;
A=4'b0101; B=4'b1110; #100;
A=4'b0101; B=4'b1111; #100;

A=4'b0111; B=4'b0000; #100;
A=4'b0111; B=4'b0001; #100;
A=4'b0111; B=4'b0010; #100;
A=4'b0111; B=4'b0011; #100;
A=4'b0111; B=4'b0100; #100;
A=4'b0111; B=4'b0101; #100;
A=4'b0111; B=4'b0110; #100;
A=4'b0111; B=4'b0111; #100;
A=4'b0111; B=4'b1000; #100;
A=4'b0111; B=4'b1001; #100;
A=4'b0111; B=4'b1010; #100;
A=4'b0111; B=4'b1011; #100;
A=4'b0111; B=4'b1100; #100;
A=4'b0111; B=4'b1101; #100;
A=4'b0111; B=4'b1110; #100;
A=4'b0111; B=4'b1111; #100;

A=4'b1000; B=4'b0000; #100;
A=4'b1000; B=4'b0001; #100;
A=4'b1000; B=4'b0010; #100;
A=4'b1000; B=4'b0011; #100;
A=4'b1000; B=4'b0100; #100;
A=4'b1000; B=4'b0101; #100;
A=4'b1000; B=4'b0110; #100;
A=4'b1000; B=4'b0111; #100;
A=4'b1000; B=4'b1000; #100;
```

```
      A=4'b1000; B=4'b1001; #100;
      A=4'b1000; B=4'b1010; #100;
      A=4'b1000; B=4'b1011; #100;
      A=4'b1000; B=4'b1100; #100;
      A=4'b1000; B=4'b1101; #100;
      A=4'b1000; B=4'b1110; #100;
      A=4'b1000; B=4'b1111; #100;

      A=4'b1001; B=4'b0000; #100;
      A=4'b1001; B=4'b0001; #100;
      A=4'b1001; B=4'b0010; #100;
      A=4'b1001; B=4'b0011; #100;
      A=4'b1001; B=4'b0100; #100;
      A=4'b1001; B=4'b0101; #100;
      A=4'b1001; B=4'b0110; #100;
      A=4'b1001; B=4'b0111; #100;
      A=4'b1001; B=4'b1000; #100;
      A=4'b1001; B=4'b1001; #100;
      A=4'b1001; B=4'b1010; #100;
      A=4'b1001; B=4'b1011; #100;
      A=4'b1001; B=4'b1100; #100;
      A=4'b1001; B=4'b1101; #100;
      A=4'b1001; B=4'b1110; #100;
      A=4'b1001; B=4'b1111; #100;

    end
  endmodule
```

Modelsim simulation of the comparator



2.2.6 DECODER

The Verilog code:

```
   module decoder (num, left_out, right_out);
         input [3:0] num;
        output [7:0] left_out, right_out;
         reg [7:0] left_out, right_out;

    always @(num)
      case (num)
        4'b0000 : right_out = 8'b00000001;
        4'b0001 : right_out = 8'b01100000;
        4'b0010 : right_out = 8'b11011010;
        4'b0011 : right_out = 8'b11110010;
        4'b0100 : right_out = 8'b01100110;
        4'b0101 : right_out = 8'b10110100;
        4'b0110 : right_out = 8'b10111110;
        4'b0111 : right_out = 8'b11100000;
        4'b1000 : right_out = 8'b01100000;
        4'b1001 : right_out = 8'b11011010;
        4'b1010 : right_out = 8'b11110010;
        4'b1011 : right_out = 8'b01100110;
        4'b1100 : right_out = 8'b10110100;
        4'b1101 : right_out = 8'b10111110;
        4'b1110 : right_out = 8'b11100000;
        4'b1111 : right_out = 8'b11111110;

        4'b0000 : left_out = 8'b00000000;
        4'b0001 : left_out = 8'b00000000;
        4'b0010 : left_out = 8'b00000000;
        4'b0011 : left_out = 8'b00000000;
        4'b0100 : left_out = 8'b00000000;
        4'b0101 : left_out = 8'b00000000;
        4'b0110 : left_out = 8'b00000000;
        4'b0111 : left_out = 8'b00000000;
        4'b1000 : left_out = 8'b00000010;
        4'b1001 : left_out = 8'b00000010;
        4'b1010 : left_out = 8'b00000010;
        4'b1011 : left_out = 8'b00000010;
        4'b1100 : left_out = 8'b00000010;
      4'b1101 : left_out = 8'b00000010;
      4'b1110 : left_out = 8'b00000010;
      4'b1111 : left_out = 8'b00000010;

    endcase
endmodule
```