Hacettepe University

Computer Engineering Department

BM233 Logic Design Lab - 2022 Fall

Assignment-1

December 11, 2022

Student name:
Mustafa Emre
Yıldırım

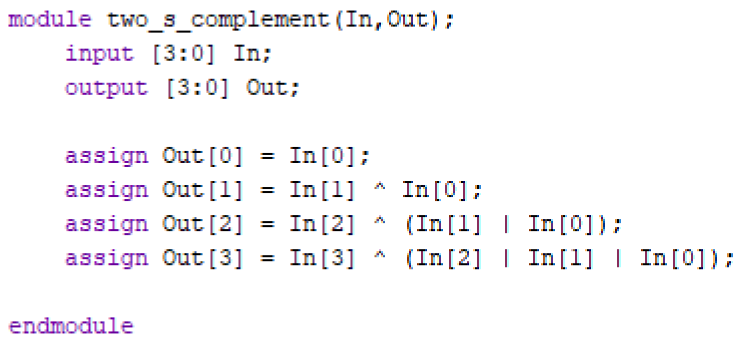
Student Number:
2200356068

1 Problem Definition

-Four bit two's complement:
We want to inverse the 4 bit binaries.
-Four bit 2x1 mux:
We want to choose one of the 2 given 4 bit binaries.
-Four bit rca:
We want to add 2 given 4 bit binaries.
-Four bit adder subtractor:
We want to select between add and subtract operation, if add is selected we add 2 given 4 bit binaries else we subtract them.

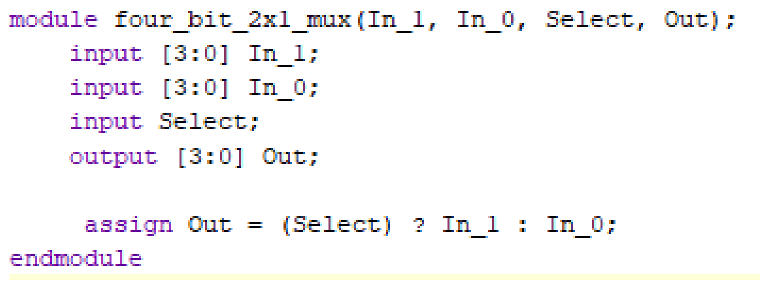
2 Solution Implementation

1- Two's Complement



```
module two_s_complement(In,Out);
    input [3:0] In;
    output [3:0] Out;

    assign Out[0] = In[0];
    assign Out[1] = In[1] ^ In[0];
    assign Out[2] = In[2] ^ (In[1] | In[0]);
    assign Out[3] = In[3] ^ (In[2] | In[1] | In[0]);

endmodule
```

2-Four Bit 2x1 Mux



```
module four_bit_2x1_mux(In_1, In_0, Select, Out);
    input [3:0] In_1;
    input [3:0] In_0;
    input Select;
    output [3:0] Out;

     assign Out = (Select) ? In_1 : In_0;
endmodule
```

3-Four Bit Rca

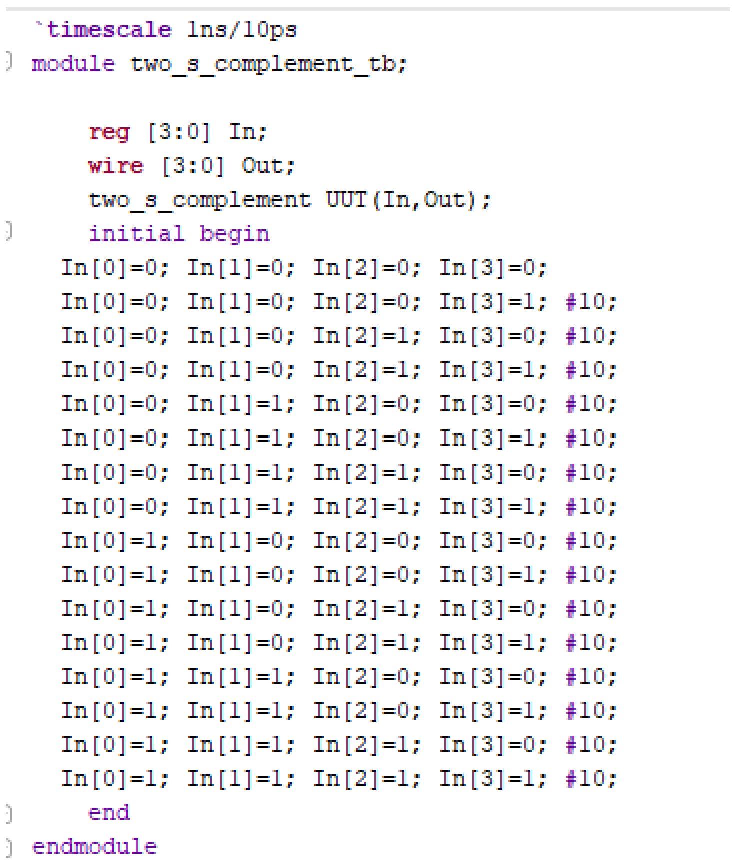
```
module four_bit_rca(
    input [3:0] A,
    input [3:0] B,
    input Cin,
    output [3:0] S,
    output Cout
);
    wire c1;
    wire c2;
    wire c3;
    full_adder fa0(.A(A[0]), .B(B[0]), .Cin(Cin), .S(S[0]), .Cout(c1));
    full_adder fa1(.A(A[1]), .B(B[1]), .Cin(c1), .S(S[1]), .Cout(c2));
    full_adder fa2(.A(A[2]), .B(B[2]), .Cin(c2), .S(S[2]), .Cout(c3));
    full_adder fa3(.A(A[3]), .B(B[3]), .Cin(c3), .S(S[3]), .Cout(Cout));

endmodule
```

4-Four Bit Adder Subtractor

```
module four_bit_adder_subtractor(A, B, subtract, Result, Cout);
    input [3:0] A;
    input [3:0] B;
    input subtract;
    output [3:0] Result;
    output Cout;

    wire [3:0] complementB,selectedBit;
    two_s_complement f1(B,complementB);
    four_bit_2x1_mux f2(complementB, B, subtract, selectedBit);
    four_bit_rca f3(A,selectedBit,0,Result,Cout);
endmodule
```

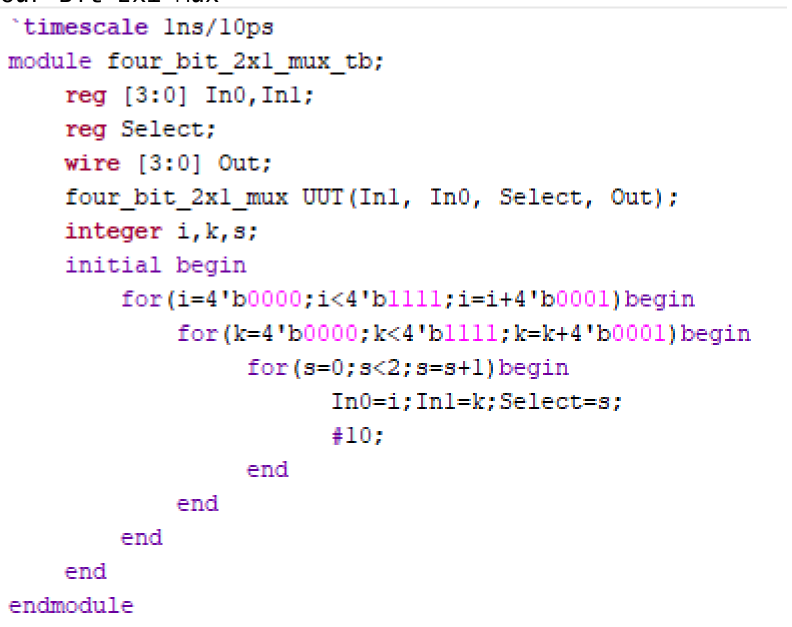
3 TestBench Implementation

1- Two’s Complement



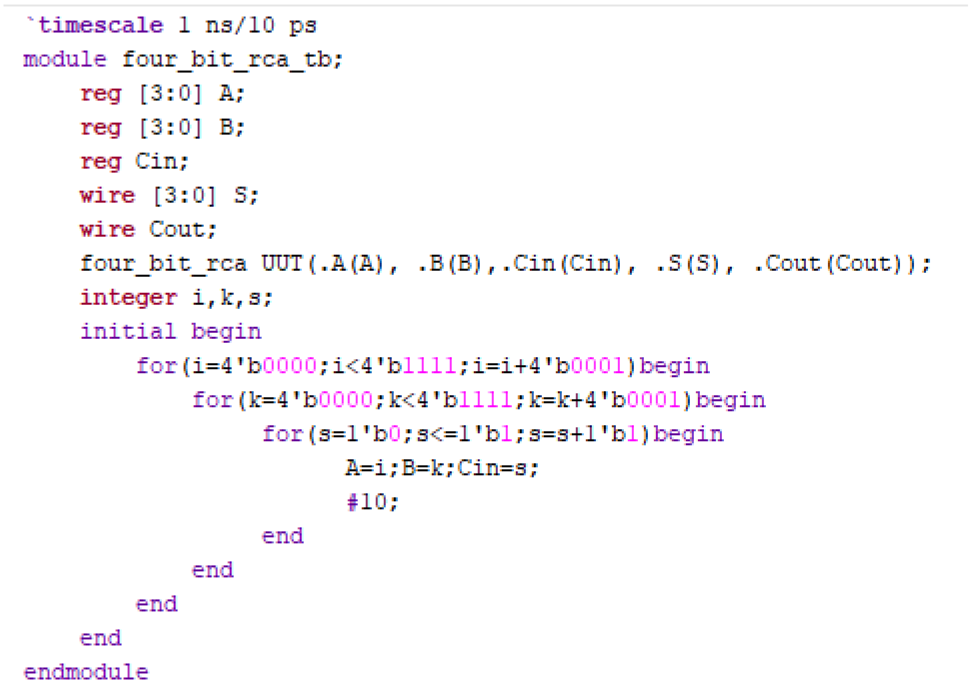
```
`timescale 1ns/10ps
module two_s_complement_tb;

    reg [3:0] In;
    wire [3:0] Out;
    two_s_complement UUT(In,Out);
    initial begin
  In[0]=0; In[1]=0; In[2]=0; In[3]=0;
  In[0]=0; In[1]=0; In[2]=0; In[3]=1; #10;
  In[0]=0; In[1]=0; In[2]=1; In[3]=0; #10;
  In[0]=0; In[1]=0; In[2]=1; In[3]=1; #10;
  In[0]=0; In[1]=1; In[2]=0; In[3]=0; #10;
  In[0]=0; In[1]=1; In[2]=0; In[3]=1; #10;
  In[0]=0; In[1]=1; In[2]=1; In[3]=0; #10;
  In[0]=0; In[1]=1; In[2]=1; In[3]=1; #10;
  In[0]=1; In[1]=0; In[2]=0; In[3]=0; #10;
  In[0]=1; In[1]=0; In[2]=0; In[3]=1; #10;
  In[0]=1; In[1]=0; In[2]=1; In[3]=0; #10;
  In[0]=1; In[1]=0; In[2]=1; In[3]=1; #10;
  In[0]=1; In[1]=1; In[2]=0; In[3]=0; #10;
  In[0]=1; In[1]=1; In[2]=0; In[3]=1; #10;
  In[0]=1; In[1]=1; In[2]=1; In[3]=0; #10;
  In[0]=1; In[1]=1; In[2]=1; In[3]=1; #10;
    end
endmodule
```

2-Four Bit 2x1 Mux



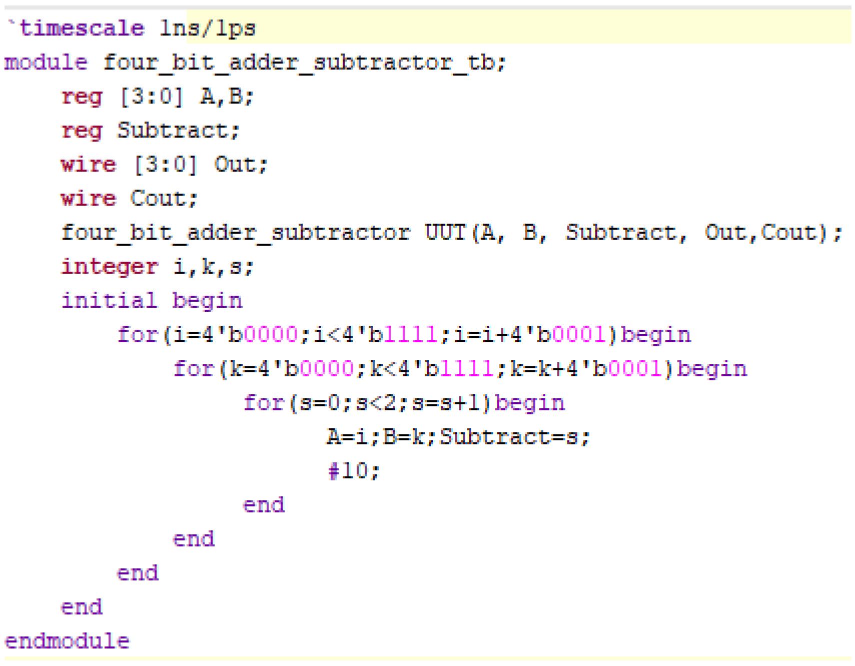
```
`timescale 1ns/10ps
module four_bit_2x1_mux_tb;
    reg [3:0] In0,In1;
    reg Select;
    wire [3:0] Out;
    four_bit_2x1_mux UUT(In1, In0, Select, Out);
    integer i,k,s;
    initial begin
        for(i=4'b0000;i<4'b1111;i=i+4'b0001)begin
            for(k=4'b0000;k<4'b1111;k=k+4'b0001)begin
                 for(s=0;s<2;s=s+1)begin
                       In0=i;In1=k;Select=s;
                       #10;
                 end
            end
        end
    end
endmodule
```

3-Four Bit Rca



```
`timescale 1 ns/10 ps
module four_bit_rca_tb;
    reg [3:0] A;
    reg [3:0] B;
    reg Cin;
    wire [3:0] S;
    wire Cout;
    four_bit_rca UUT(.A(A), .B(B),.Cin(Cin), .S(S), .Cout(Cout));
    integer i,k,s;
    initial begin
        for(i=4'b0000;i<4'b1111;i=i+4'b0001)begin
            for(k=4'b0000;k<4'b1111;k=k+4'b0001)begin
                 for(s=1'b0;s<=1'b1;s=s+1'b1)begin
                        A=i;B=k;Cin=s;
                        #10;
                 end
            end
        end
    end
endmodule
```

4-Four Bit Adder Subtractor



```
`timescale 1ns/1ps
module four_bit_adder_subtractor_tb;
    reg [3:0] A,B;
    reg Subtract;
    wire [3:0] Out;
    wire Cout;
    four_bit_adder_subtractor UUT(A, B, Subtract, Out,Cout);
    integer i,k,s;
    initial begin
        for(i=4'b0000;i<4'b1111;i=i+4'b0001)begin
            for(k=4'b0000;k<4'b1111;k=k+4'b0001)begin
                 for(s=0;s<2;s=s+1)begin
                        A=i;B=k;Subtract=s;
                        #10;
                 end
            end
        end
    end
endmodule
```

4 Results

1-Two’s Complement

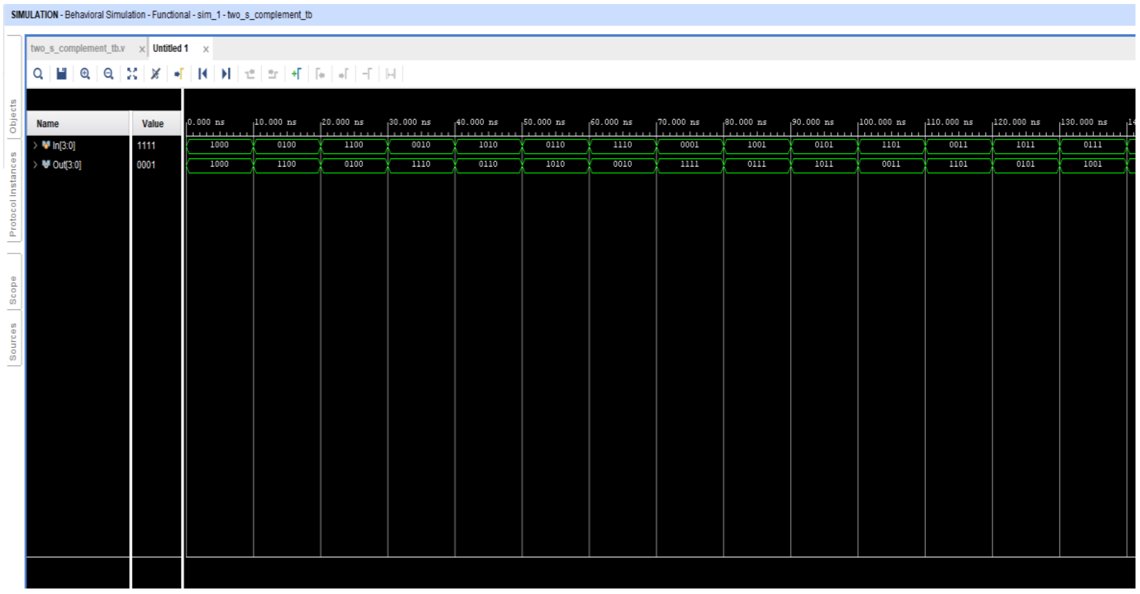
I used k-mapping method for two’s complement’s derive equations

Out[0] formula: F(In[3], In[2], In[1], In[0]) =P(1, 3, 5, 7, 9, 11, 13, 15)

Out[1] formula: F(In[3], In[2], In[1], In[0]) =P(1, 2, 5, 6, 9, 10, 13, 14)

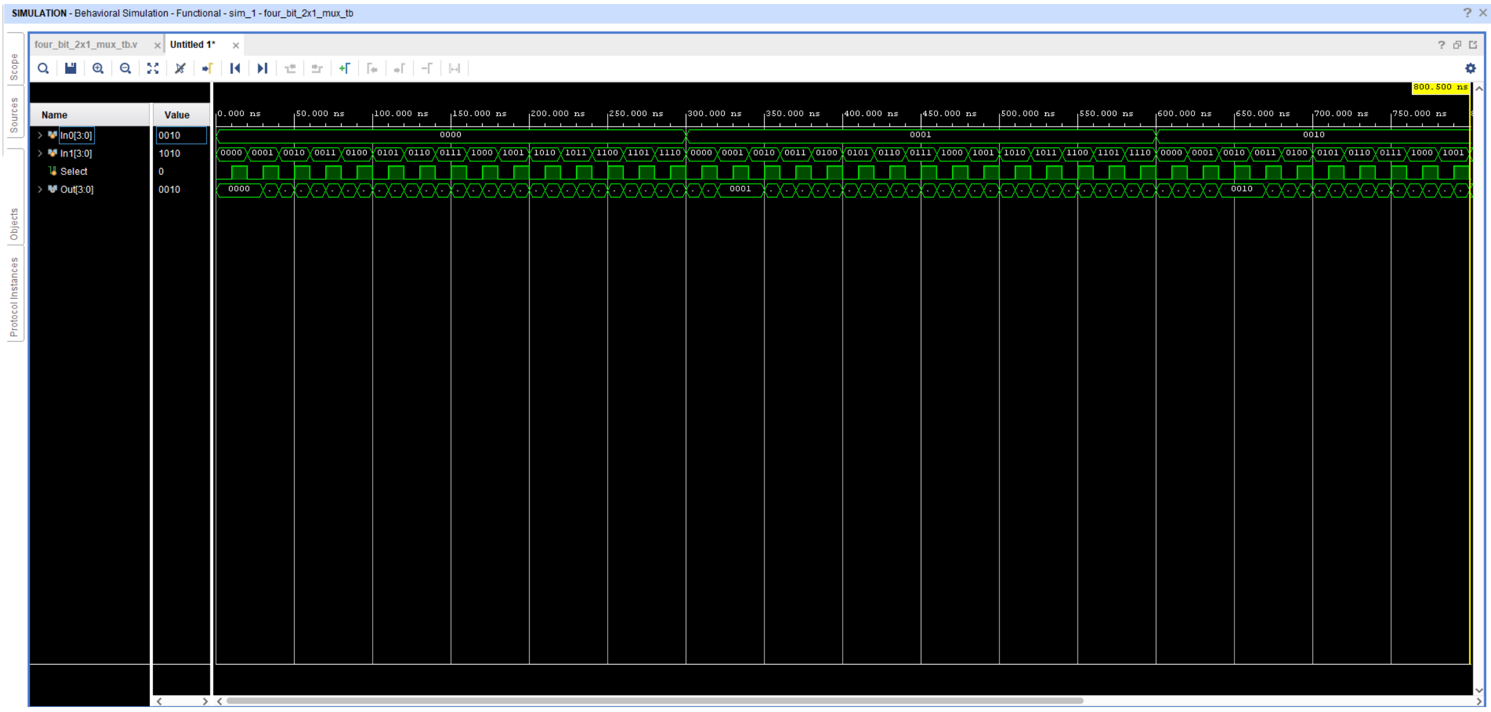
Out[2] formula: F(In[3], In[2], In[1], In[0]) =P(1, 2, 3, 4, 9, 10, 11, 12)

Out[3] formula: F(In[3], In[2], In[1], In[0]) =P(1, 2, 3, 4, 5, 6, 7, 8, 10)



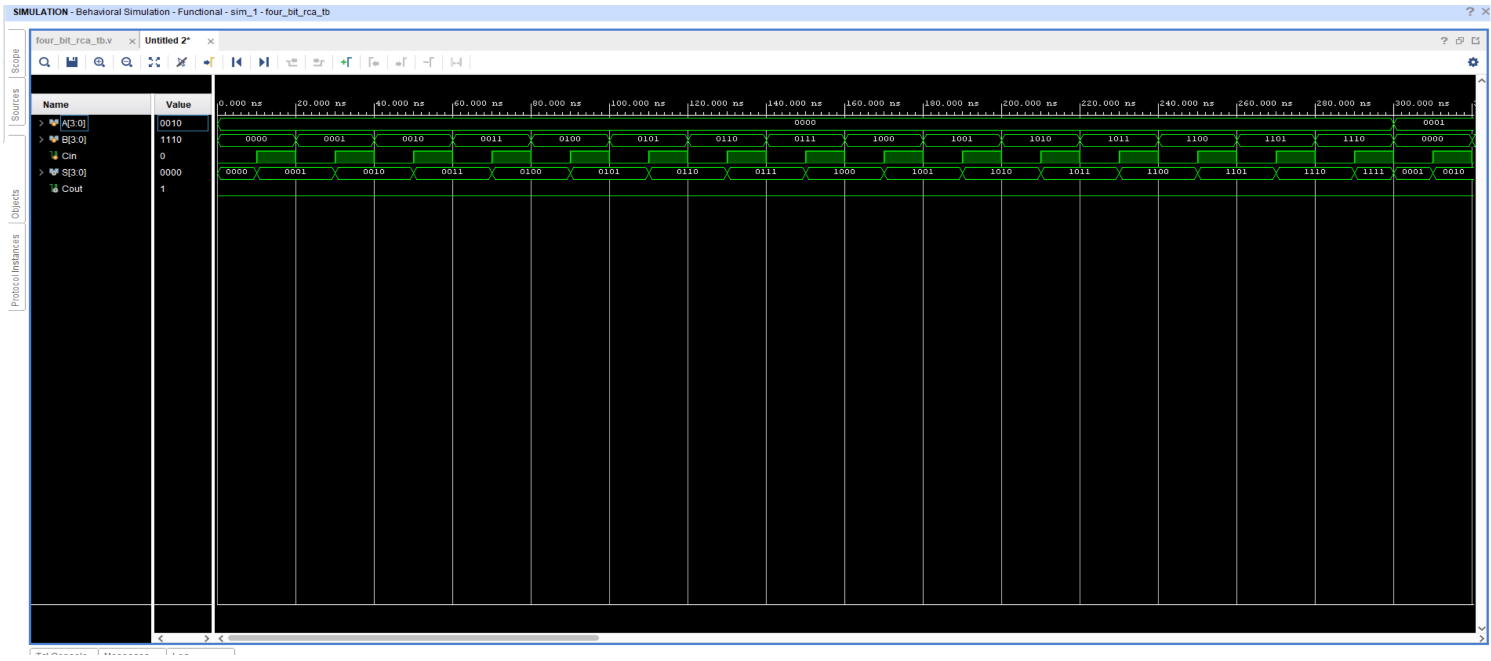
2-Four Bit 2x1 Mux:

I used this link [http://www.asic-world.com/examples/verilog/mux.html for 2x1](http://www.asic-world.com/examples/verilog/mux.html) Mux



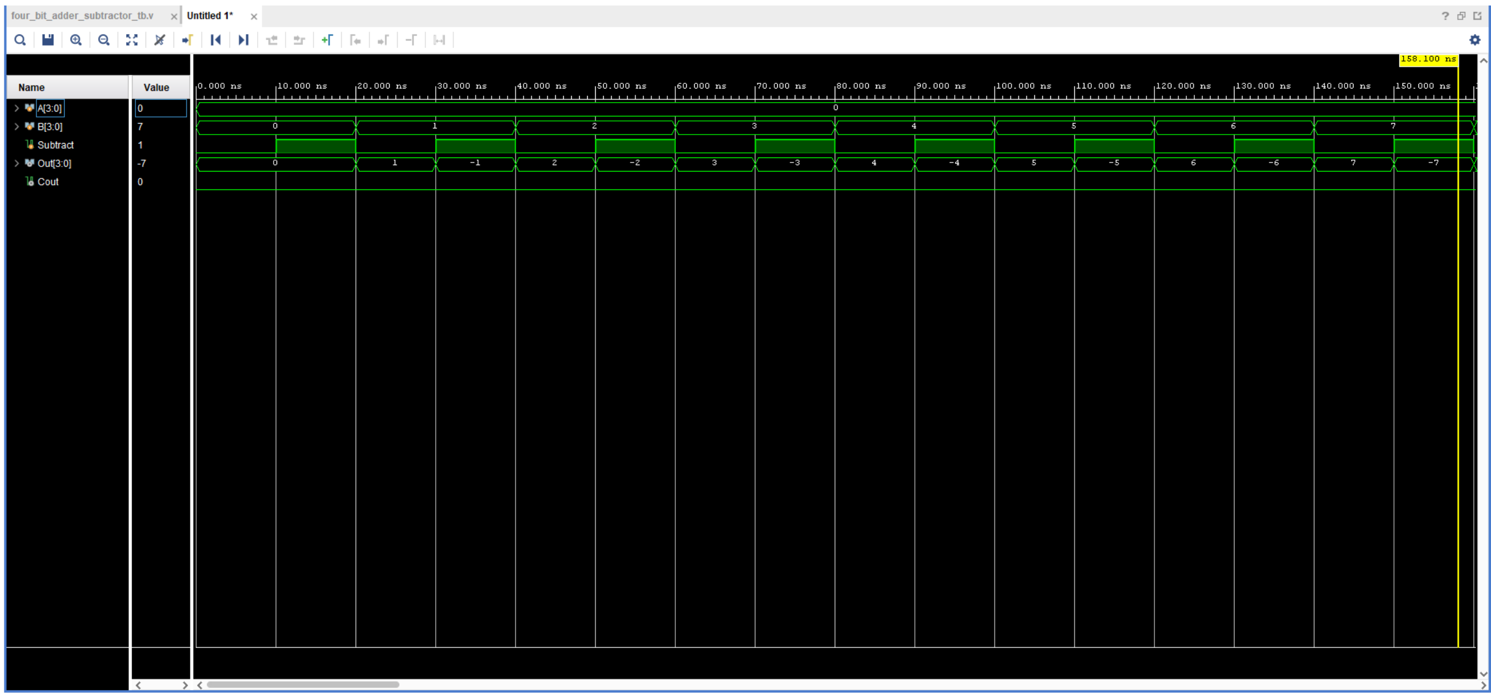
3-Four Bit Rca

I used 4 Full Adder for Fout Bit Rca



4-Four Bit Adder Subtractor:

I used two's complement , four bit 2x1 mux and fou



Name	Value
A[3:0]	0
B[3:0]	7
Subtract	1
Out[3:0]	-7
Cout	0

```
A[3:0]:   0
B[3:0]:   0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
Out[3:0]: 0 | 1 | -1 | 2 | -2 | 3 | -3 | 4 | -4 | 5 | -5 | 6 | -6 | 7 | -7
```