Hacettepe University

Computer Engineering Department

BM233 Logic Design Lab - 2021 Fall

Assignment Name Goes Here

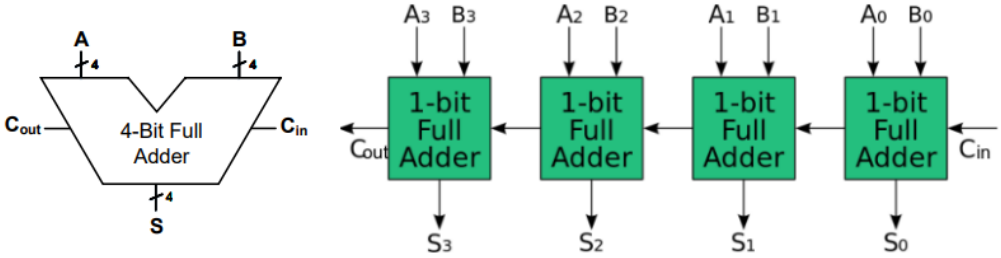
December 10, 2022

Student name:
Zeynep Yeşilkaya

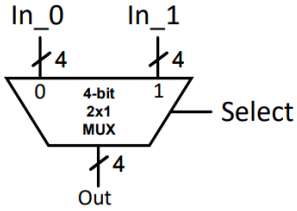
Student Number:
b2210356048

1 Problem Definition

Two's complement is the way a computer uses to represent signed (positive, negative, and zero) integers. It is a mathematical operation to reversibly convert a positive binary number into a negative binary number with an equivalent (but negative) value

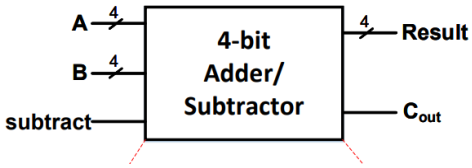


A multiplexer (MUX) is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. A MUX has a maximum of 2 n data inputs. One of the inputs is connected to the output based on the value of the selection line(s). There will be 2 n possible combinations of 1s and 0s.



Select	Out
0	In_0
1	In_1

A full adder is a combinational logic circuit that forms the arithmetic sum of three binary numbers.



2 Solution Implementation

First, a two's complementer was implemented. Then a full adder was implemented. 4bit rca was implemented using this full adder. Adder-subtractor was implemented using theese 4bit rca, multiplexer and two's complementer.

```
1  module two_s_complement(In,Out);
2      input [3:0] In;
3      output [3:0] Out;
4
5      assign Out[3] = In[3] ^ (In[2] | In[1] | In[0]);
6      assign Out[2] = In[2] ^ (In[1] | In[0]);
7      assign Out[1] = In[1] ^ In[0];
```

```
8       assign Out[0] = In[0];
9
10  endmodule
11
12
13  module full_adder(
14      input A,
15      input B,
16      input Cin,
17      output S,
18      output Cout
19  );
20      assign {Cout, S} = A + B + Cin;
21
22  endmodule
23
24
25  module four_bit_2x1_mux(In_1, In_0, Select, Out);
26          input [3:0] In_1;
27          input [3:0] In_0;
28          input Select;
29          output [3:0] Out;
30
31          assign Out = (Select) ? In_1 : In_0;
32
33  endmodule
34
35
36  module four_bit_rca(
37      input [3:0] A,
38      input [3:0] B,
39      input Cin,
40      output [3:0] S,
41      output Cout
42  );
43      wire s1,s2,s3,c1,c2,c3;
44      full_adder u1(A[0],B[0],Cin,S[0],c1);
45      full_adder u2(A[1],B[1],c1,S[1],c2);
46      full_adder u3(A[2],B[2],c2,S[2],c3);
47      full_adder u4(A[3],B[3],c3,S[3],Cout);
48
49  endmodule
50
51
52
53  module four_bit_adder_subtractor(A, B, subtract, Result, Cout);
54      input [3:0] A;
55      input [3:0] B;
```

```
56      input subtract;
57      output [3:0] Result;
58      output Cout;
59      wire [3:0] result_of_mux;
60      wire Cin=0;
61      wire [3:0] complement_B;
62
63      two_s_complement two_s_complementer(B,complement_B);
64      four_bit_2x1_mux mux(complement_B,B,subtract,result_of_mux);
65      four_bit_rca rca(A,result_of_mux,Cin,Result,Cout);
66
67  endmodule
```

3 Testbench Implementation

In this part, codes are tested. To change the values, double for was used in some parts, 8-bit count was used in some parts.

```
1
2   module two_s_complement_tb;
3       reg[3:0] In;
4       reg[3:0] count = 4'b0000;
5       wire[3:0] Out;
6       integer i;
7       two_s_complement UUT(In,Out);
8
9       initial begin
10          $dumpfile("two_s_complement.vcd");
11          $dumpvars;
12          for(i = 0;i < 16;i++) begin
13              {In[3],In[2],In[1],In[0]} = count;
14              count+=1;
15              #10;
16          end
17          $finish;
18      end
19  endmodule
```

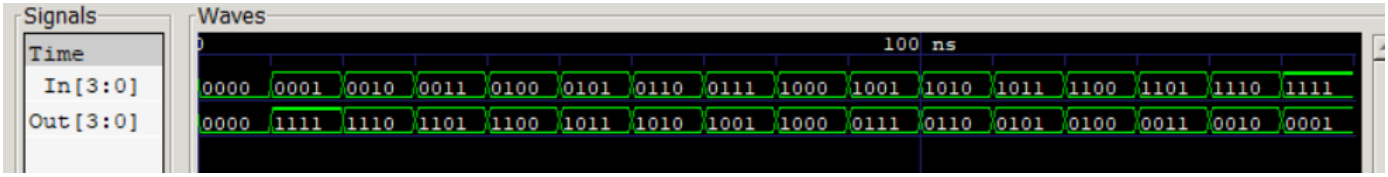


Figure 1: Two's Complement

```
module full_adder_tb;
    reg In_1;
    reg In_0;
    reg Cin=0;
    wire S;
    reg[2:0] count = 3'b000;
    wire Out;
    integer i;

    full_adder UUT(In_1,In_0,Cin,S,Out);

    initial begin
        $dumpfile("full_adder.vcd");
        $dumpvars;
        for(i = 0;i < 8;i++) begin
            {In_1,In_0,Cin} = count;
            count+=1;
            #10;
        end
        $finish;
    end

endmodule
```

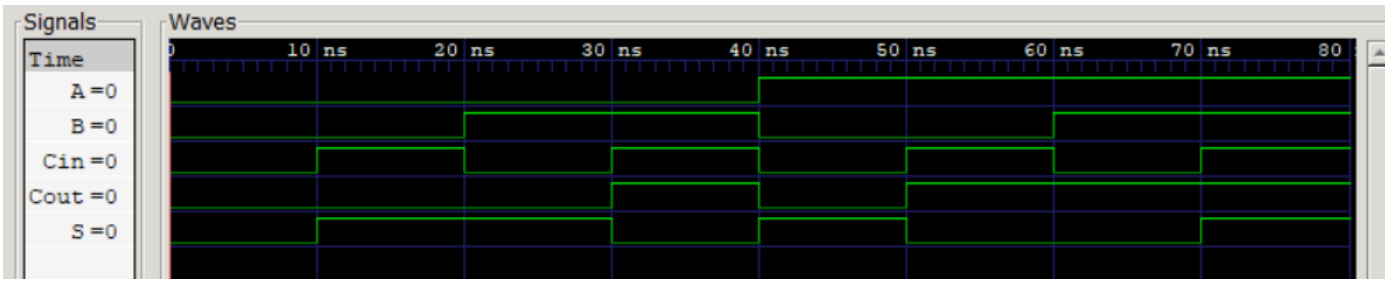


Figure 2: Full Adder

```
module four_bit_rca_tb;

    reg[3:0] In_0;
    reg[3:0] In_1;
    reg Cin=0;
    reg[7:0] count = 8'b00000000;
    wire[3:0] S;
    wire Out;
    integer i;

    four_bit_rca UUT(In_0,In_1,Cin,S,Out);

```

```
    initial begin
        $dumpfile("four_bit_rca.vcd");
        $dumpvars;
        for(i=0;i<256;i++) begin
          {In_1[3],In_1[2],In_1[1],In_1[0]
          ,In_0[3],In_0[2],In_0[1],In_0[0]} = count;
          count+=1;
          #10;
        end
        $finish;
    end
endmodule
```

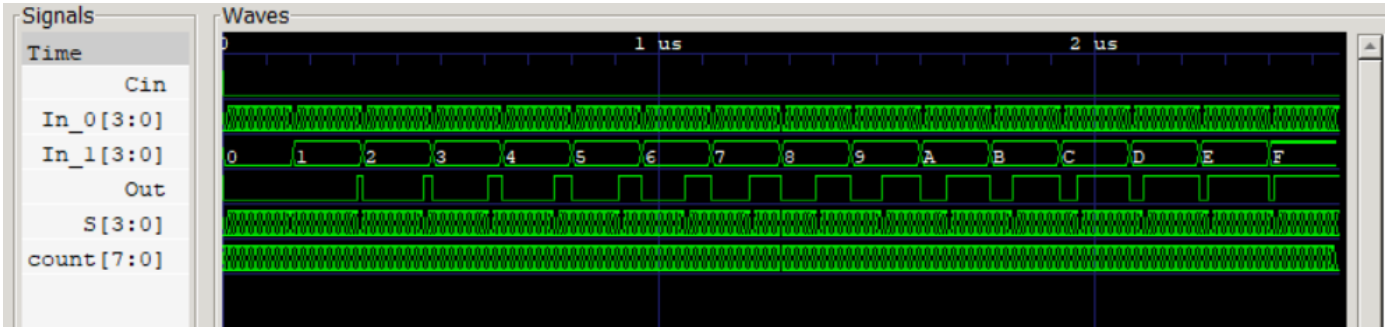


Figure 3: Four Bit Rca

```
module four_bit_2x1_mux_tb;
        reg [3:0] In_1;
        reg [3:0] In_0;
        reg Select=0;
        wire [3:0] Out;
        reg[7:0] count = 8'b00000000;
        integer i;

        four_bit_2x1_mux UUT(In_1, In_0, Select, Out);

        initial begin
                $dumpfile("four_bit_2x1_mux.vcd");
                $dumpvars;
                for(i=0;i<256;i++) begin
                        {In_1[3],In_1[2],In_1[1],In_1[0]
                ,In_0[3],In_0[2],In_0[1],In_0[0]} = count;
                        count+=1;
                        #10;
                end
                Select=1;
                for(i=0;i<256;i++) begin
```

```
                        {In_1[3],In_1[2],In_1[1],In_1[0]
                ,In_0[3],In_0[2],In_0[1],In_0[0]} = count;
                        count+=1;
                        #10;
                end
                $finish;
        end
endmodule
```

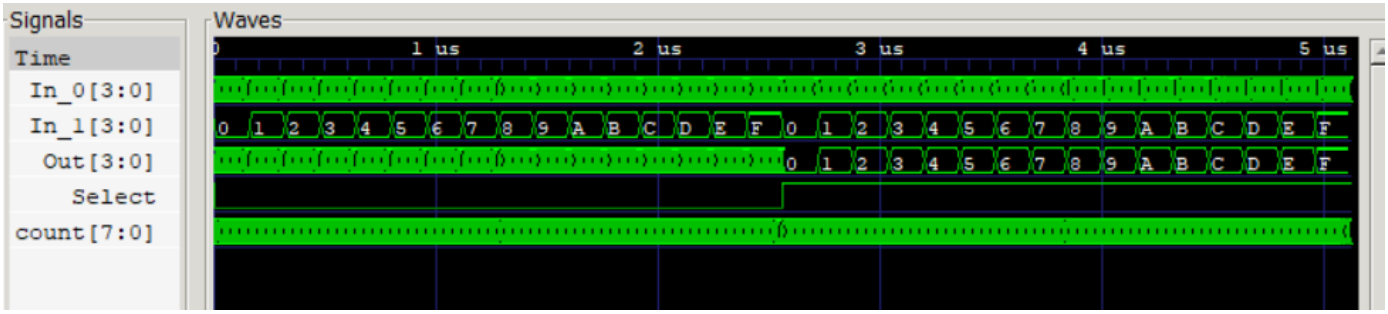


Figure 4: Four Bit 2x1 Mux

```
module four_bit_adder_subtractor_tb;

    reg [3:0] A;
    reg [3:0] B;
    reg subtract;

    wire [3:0] Result;
    wire Cout;

    integer i;
    integer j;
    integer k;


    four_bit_adder_subtractor UUT(A, B, subtract, Result, Cout);

    initial begin
        $dumpfile("four_bit_adder_subtractor.vcd");
        $dumpvars;

        for(i=0;i<2;i++)begin
            subtract=i;
            for(j=0;j<16;j++)begin
                A = j;
                for(k=0;k<16;k++)begin
                    B = k;
```

```
27                          #10;
28                      end
29                  end
30              end
31              $finish;
32          end
33
34  endmodule
```

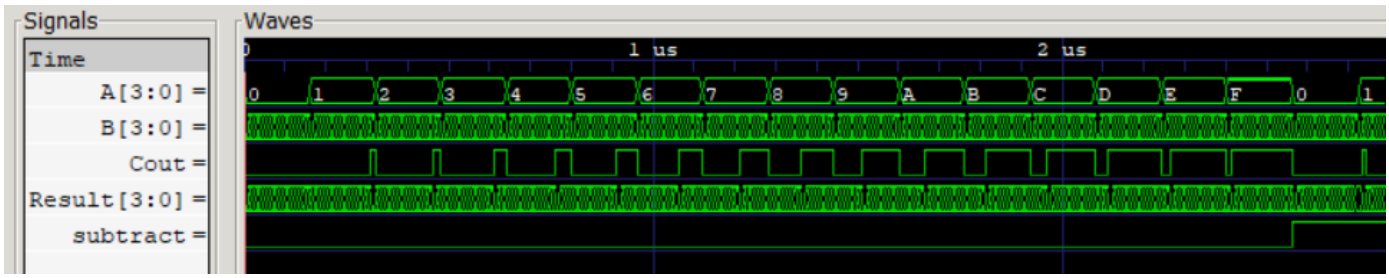


Figure 5: Four Bit Adder Subtractor

References

- http://www.asic-world.com/examples/verilog/mux.html
- https://cdn-uploads.piazza.com/paste/itmvemweb267cd/20ca9bf7d8564d6df9dde83eddff107faef5
  IcarusVerilogGuide2022.pdf