Hacettepe University

Computer Engineering Department

BM233 Logic Design Lab - 2022 Fall

Experiment 4 - Combinational Circuits in Verilog

December 10, 2022

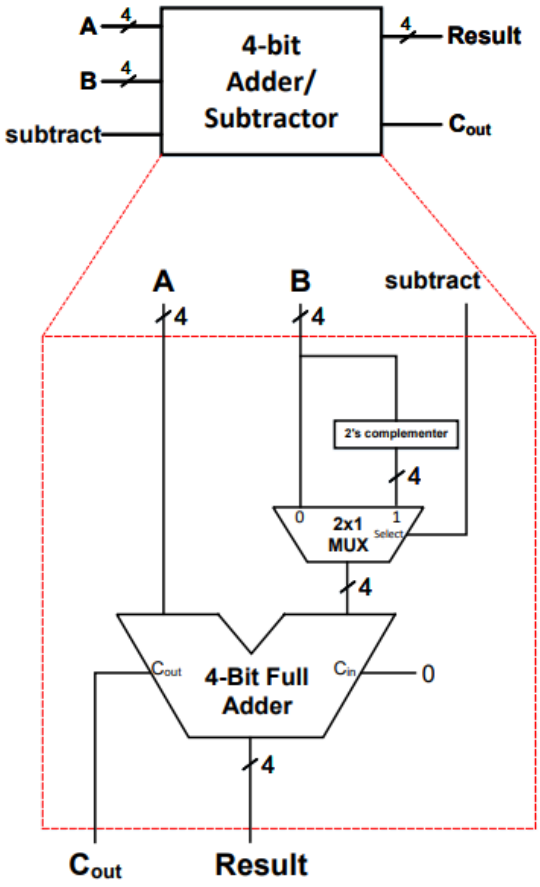
Student name:
Nurullah BAŞER

Student Number:
b2200356077

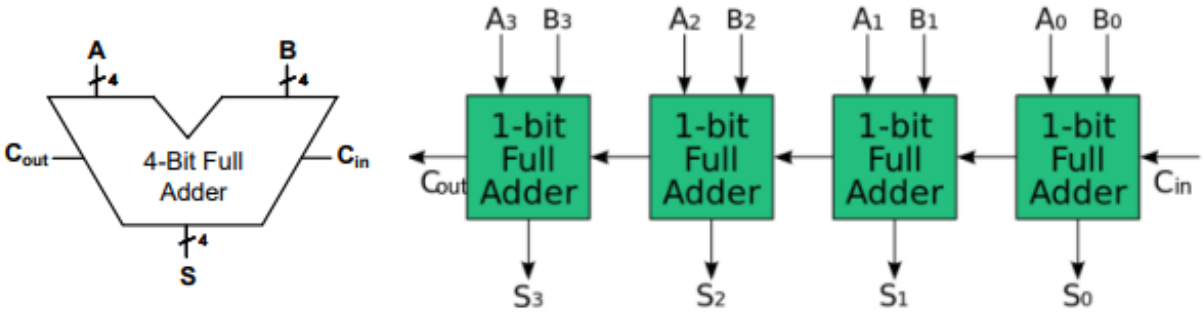
1 Problem Definition

In this assignment, a 4-bit Adder/Subtractor was done with Verilog. When designing this, 2's Complementer, 4-bit Full Adder and 4-bit 2x1 Multiplexer were also made at the same time. The accuracy of the designed circuits was tested by creating test benches.

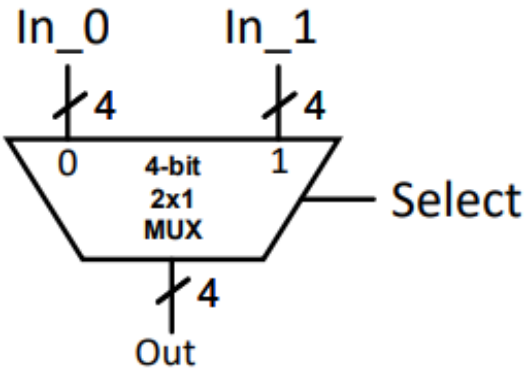
The diagram at the bottom is a schematic representation of the 4-bit Adder/Subtractor is made.



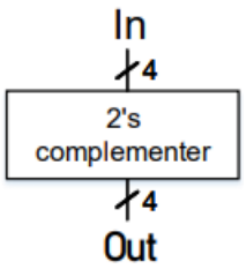
The diagram at the bottom is a schematic representation of the 4-bit Full Adder is made.



The diagram at the bottom is a schematic representation of the 4-bit Multiplexer is made.



The diagram at the bottom is a schematic representation of the 2's Complementer is made.



2 Solution Implementation

2's Complement:

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

    assign Out = (~In) + 1;
endmodule
```

1-Bit Full Adder:

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

    assign S = A ^ B ^ Cin;
    assign Cout = (A & B) | (Cin & A) | (Cin & B);

endmodule
```

4-Bit Full Adder:

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


    wire[2:0] Carries;
    full_adder A1(.A(A[0]), .B(B[0]), .Cin(Cin), .S(S[0]), .Cout(Carries[0]));
    full_adder A2(.A(A[1]), .B(B[1]), .Cin(Carries[0]), .S(S[1]), .Cout(Carries[1]));
    full_adder A3(.A(A[2]), .B(B[2]), .Cin(Carries[1]), .S(S[2]), .Cout(Carries[2]));
    full_adder A4(.A(A[3]), .B(B[3]), .Cin(Carries[2]), .S(S[3]), .Cout(Cout));

endmodule
```

4-Bit Multiplexer:

```
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
```

4-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] negatif_B;
    wire[3:0] sub_or_add_B;

    two_s_complement make_negatif(B,negatif_B);
    four_bit_2x1_mux sub_or_add(negatif_B,B,subtract,sub_or_add_B);
    four_bit_rca result(A,sub_or_add_B,1'B0,Result,Cout);

endmodule
```

3 Testbench Implementation

2's Complement Testbench:

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


   reg [3:0] In;
   reg [3:0] count = 4'b0000;
   wire [3:0] Out;

   two_s_complement DUT(.In(In), .Out(Out));

   initial begin
      $dumpfile("two_s_complement.vcd");
      $dumpvars;
      for(integer i=0;i<16;i++) begin
         {In[3],In[2],In[1],In[0]} = count;
         count += 1;
         #10;
      end
```

```
      $finish;
   end
endmodule
```

1-Bit Full Adder Testbench:

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


    reg A;
    reg B;
    reg Cin;
    wire S;
    wire Cout;

    full_adder DUT(A,B,Cin,S,Cout);

    initial begin

        $dumpfile("full_adder.vcd");
        $dumpvars;
        for(integer i = 0; i < 2; i++) begin
            Cin = i;
            for(integer j = 0; j < 2; j++) begin
                A = j;
                for(integer h = 0; h < 2; h++) begin
                    B = h;
                    #100;
                end;
            end;
        end;
    end;

endmodule
```

4-Bit Full Adder Testbench:

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

    reg[3:0] A,B;
    reg Cin;

    wire[3:0] S;
    wire Cout;

    four_bit_rca DUT(A,B,Cin,S,Cout);

    initial begin
```

```
        $dumpfile("four_bit_rca.vcd");
        $dumpvars;

        for(integer i = 0; i < 2; i++) begin
            Cin = i;
            for(integer j = 0; j < 16 ; j++) begin
                A = j;
                for(integer h = 0; h < 16; h++) begin
                    B = h;
                    #100;
                end;
            end;
        end;
    end;

endmodule
```

4-Bit Multiplexer Testbench:

```
`timescale 1ns/10ps
module four_bit_2x1_mux_tb;


        reg [3:0] In_0;
        reg [3:0] In_1;
        reg Select;
        wire [3:0] Out;

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

        initial begin
                $dumpfile("four_bit_2x1_mux.vcd");
        $dumpvars;
                for(integer s = 0; s < 2 ; s++) begin
                        Select = s;
                        for(integer i = 0; i < 16 ; i++) begin
                                In_1 = i;
                                for(integer j = 0; j < 16 ; j++) begin
                                        In_0 = j;
                                        #100;
                                end;
                        end;
                end;
        end;

endmodule
```

4-Bit Adder-Subtractor Testbench:

```
`timescale 1ns/1ps
```

```
module four_bit_adder_subtractor_tb;


    reg[3:0] A,B;
    reg subtract;
    output[3:0] Result;
    output Cout;

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


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

        for(integer s = 0; s < 2; s++) begin
            subtract = s;
            for(integer j = 0; j < 16; j++) begin
                A = j;
                for(integer h = 0; h < 16; h++) begin
                    B = h;
                    #100;
                end;
            end;
        end;
    end;


endmodule
```

4 Results

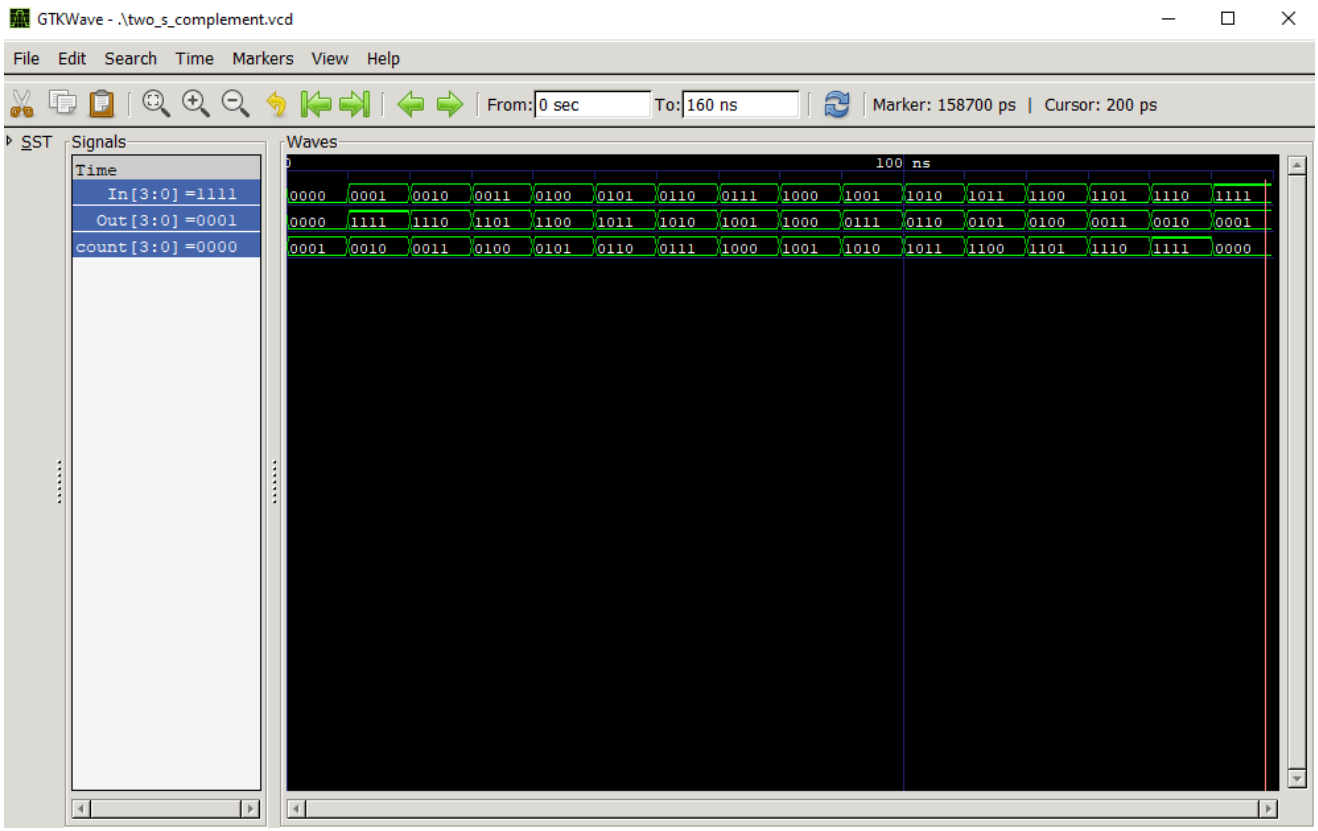


Figure 1: Resulting Waveform from 2's Complement

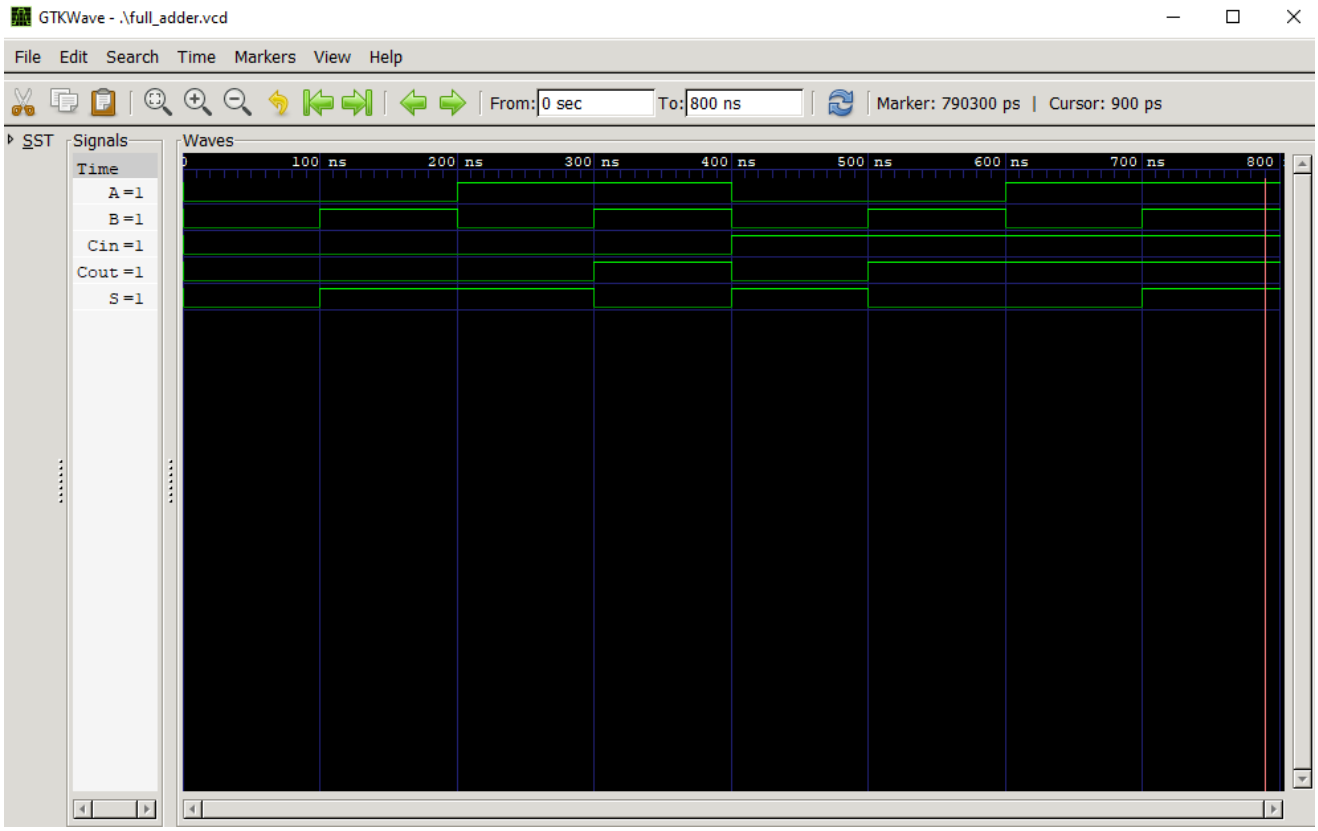


Figure 2: Resulting Waveform from 1-bit Full Adder

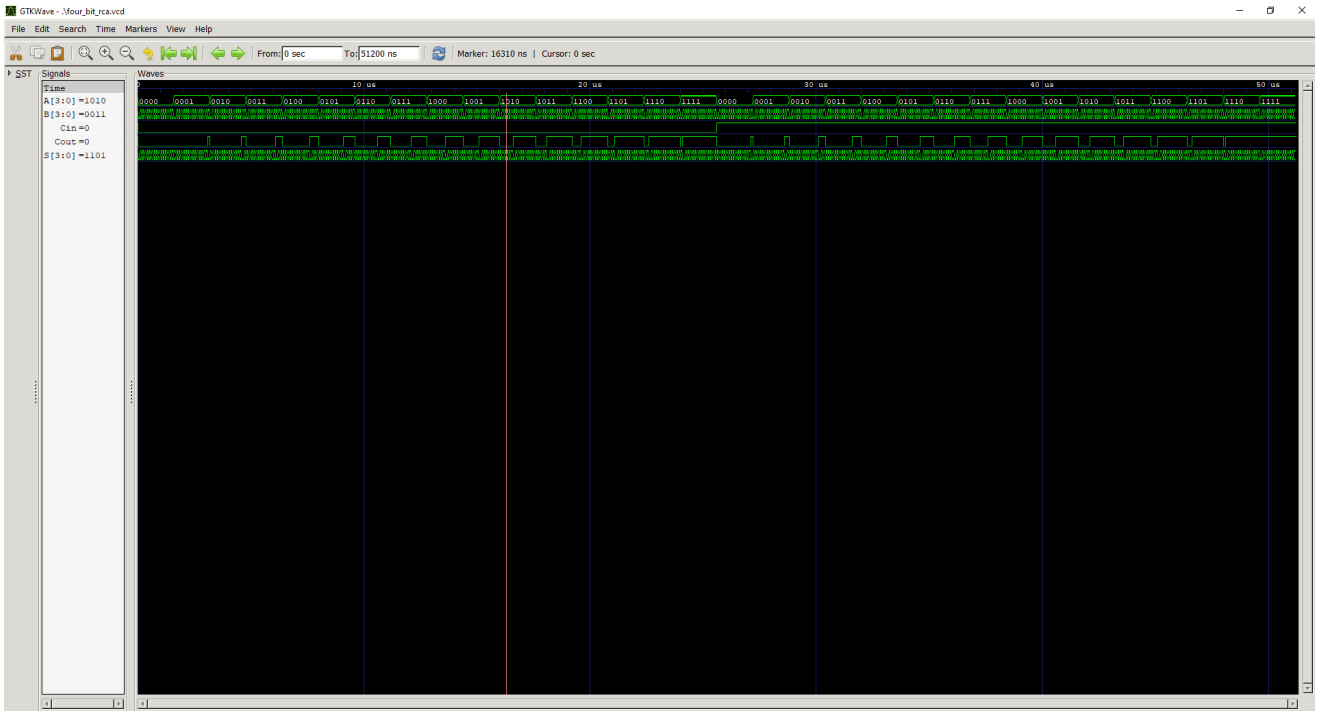


Figure 3: Resulting Waveform from 4-bit Full Adder

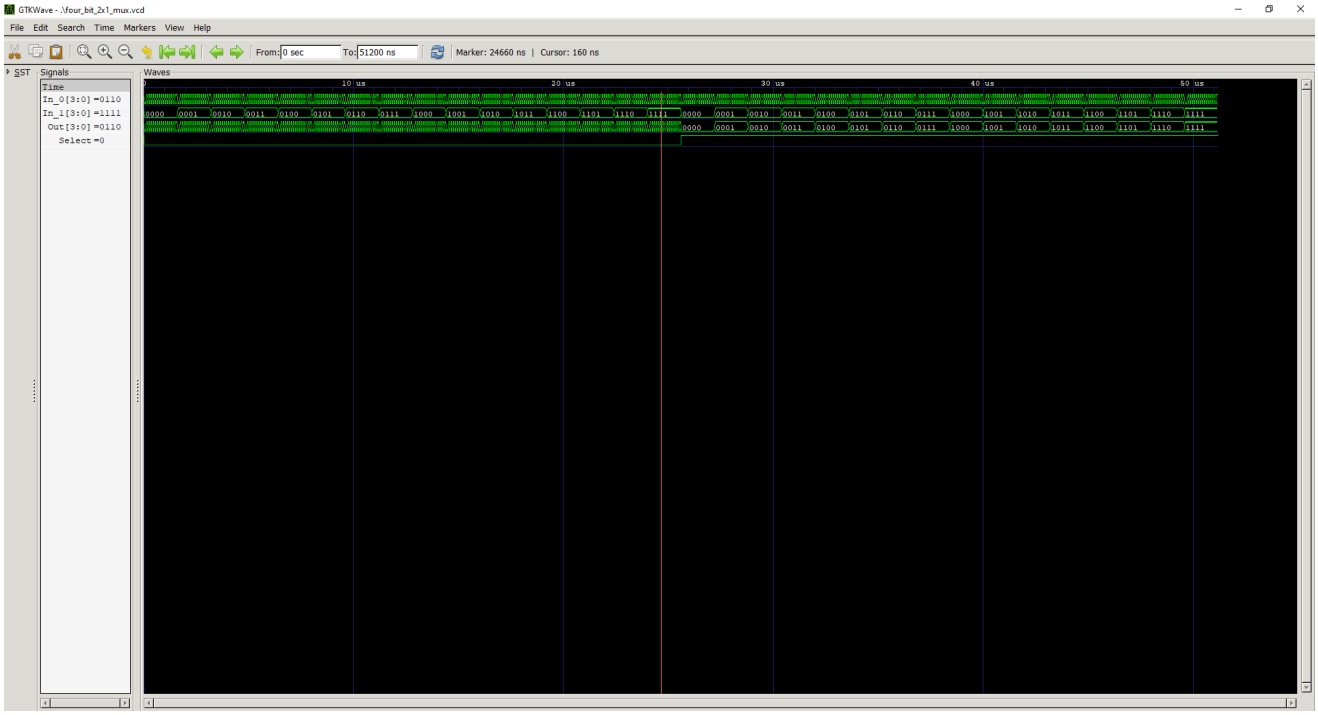


Figure 4: Resulting Waveform from 4-bit Multiplexer

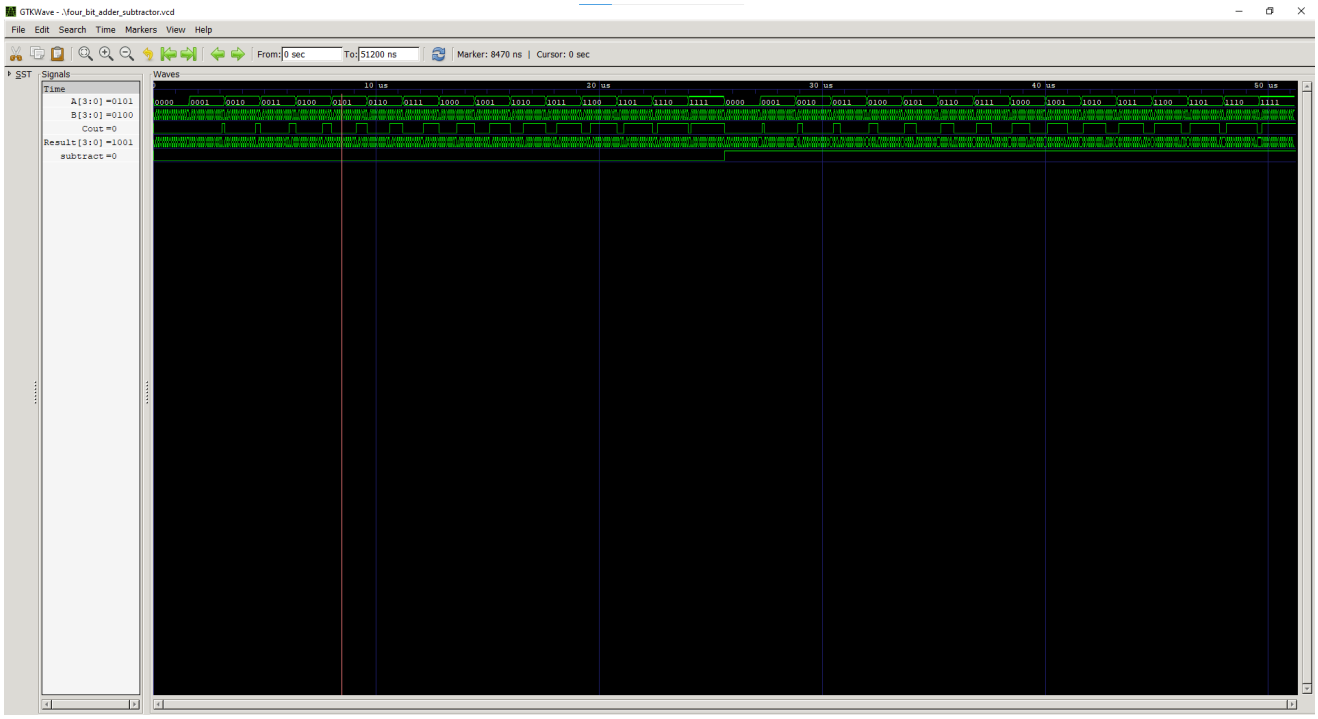


Figure 5: Resulting Waveform from 4-bit Adder-Subtractor (Subtract = 0)

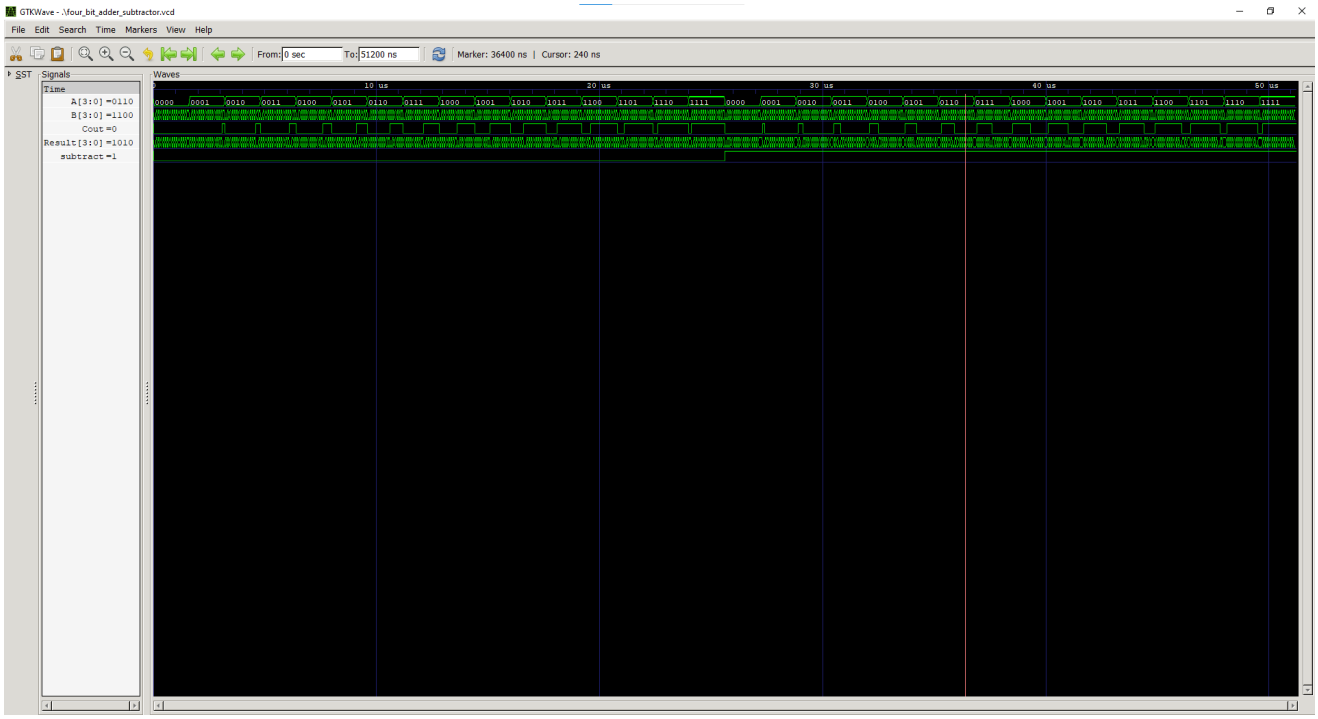


Figure 6: Resulting Waveform from 4-bit Adder-Subtractor (Subtract = 1)

Some input-output examples made with 4-bit Adder-Subtractor

A[3:0] =0011	A[3:0] =0111	A[3:0] =0011	A[3:0] =1100
B[3:0] =0100	B[3:0] =1100	B[3:0] =0111	B[3:0] =1000
Cout =0	Cout =1	Cout =0	Cout =1
Result[3:0] =0111	Result[3:0] =0011	Result[3:0] =1100	Result[3:0] =0100
subtract =0	subtract =0	subtract =1	subtract =1

References

- Part 4 - Combinational Circuits, BBM231 Lecture Notes
- Verilog HDL A Brief Introduction Fall 2022, BBM233 Lecture Notes