ARITHMETIC LOGIC UNIT OPERATIONS (ALU using 4-bit microprocessor)

Dhruvi Shah DDS200004

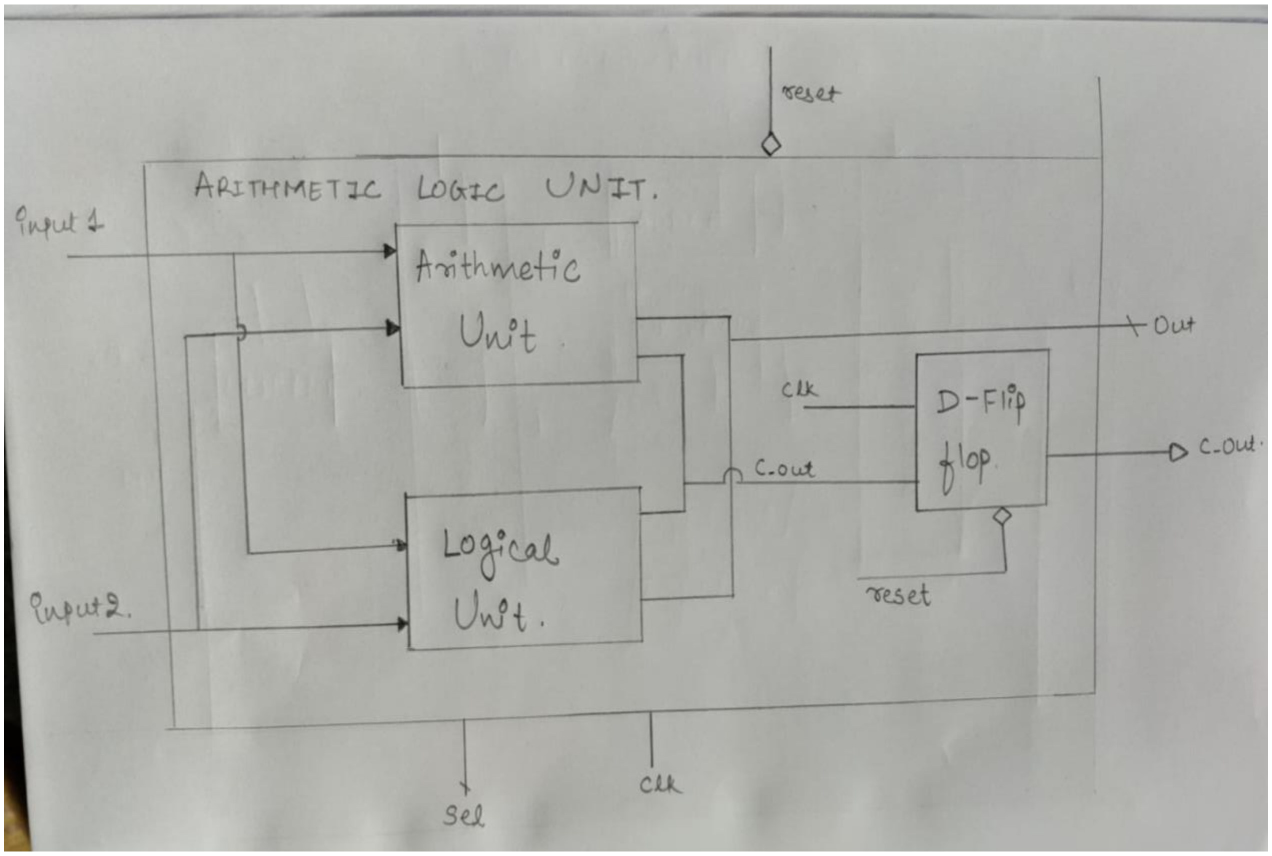
Navya Bandari NXB210004

Naga Mutya Kumar Kumtsam NXK210028

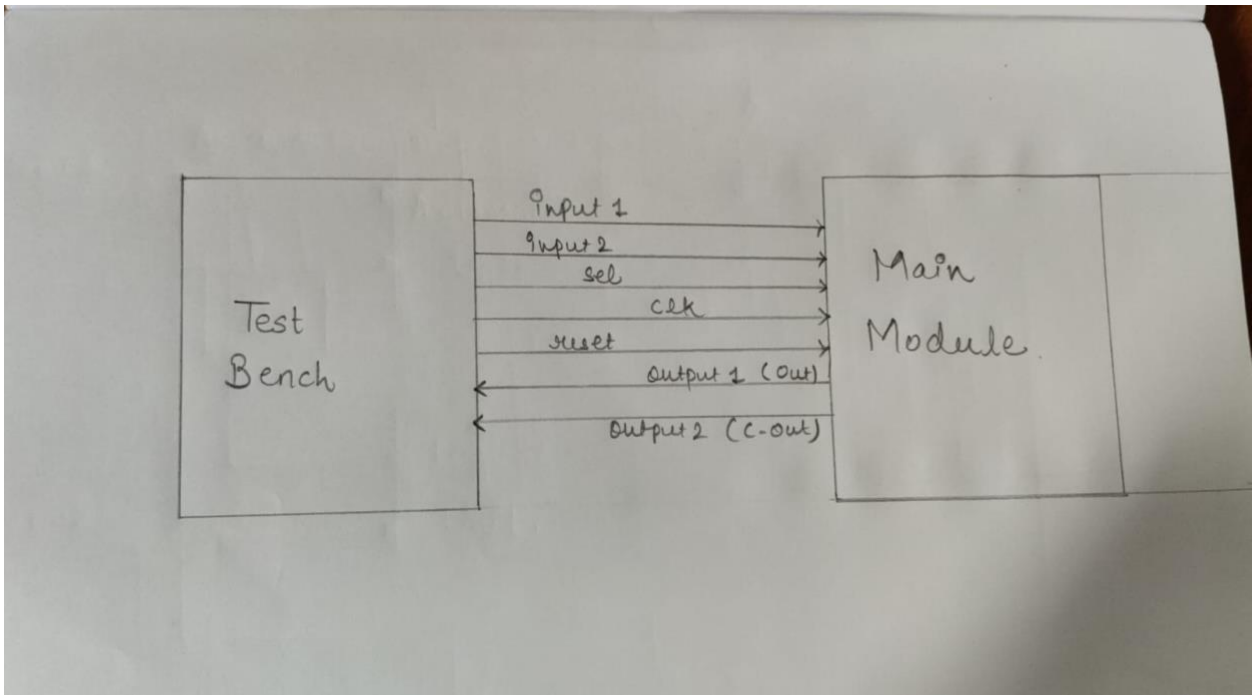
Description

The project is to design and simulate arithmetic logic unit operations using 4-bit microprocessor.

Block Diagram of Design



Connection between Test bench and Module



VERILOG CODE:

```
module MAIN(input1,input2,reset,clk,sel,out,c_out);
input[7:0] input1,input2;
input reset;
input clk;
input[3:0] sel;
output[7:0] out;
output c_out;
reg[7:0] out;
reg c_out;
always@(posedge clk)
begin
   //Implementing D-FlipFlop. If the reset signal is 1,c_out will be reset
        if(reset)
                c_out = 1'b0;
        else
                begin
                        case(sel)
                        // Arithmetic Addition
                        4'b0000: begin {c_out,out} = input1 + input2; end
                        //Arithmetic Subtraction
                        4'b0001: begin {c_out,out} = input1 - input2; end
```

```
                    //Logical AND Operation. 'c_out' is not used so it is
initialised as 0
                    4'b0010: begin out = input1 & input2; c_out = 1'b0;
end
                    //Logical OR Operation. 'c_out' is not used so it is
initialised as 0
                    4'b0011: begin out = input1 | input2; c_out = 1'b0; end
                    //Logical shift left Operation. 'c_out' is initialised as 0
                    4'b0100: begin out = input1<<1; c_out = 1'b0; end
                    //Logical Shift right 'c_out' is initialised as 0
                    4'b0101: begin out = input1>> 1; c_out = 1'b0; end
                    // Comparison Operations where 'out' is not used to
initialised to 8'd0
                    4'b0110: begin c_out = (input1>input2) ? 1'b1:1'b0; out
= 8'd0; end
                    4'b0111: begin c_out = (input1<input2) ? 1'b1:1'b0; out
= 8'd0;end
                    4'b1000: begin c_out = (input1==input2) ? 1'b1:1'b0;
out = 8'd0; end
                    endcase
                end
            end
endmodule
```

TEST BENCH CODE:

```
module MAIN_TB;
reg[7:0] input1,input2;
reg clk,reset;
reg[3:0] sel;

wire[7:0] out;
wire c_out;



MAIN DUT(input1,input2,reset,clk,sel,out,c_out);


initial begin
        #0 input1 = 8'd250; input2 = 8'd200;
        #0 sel = 4'b0010;
        reset = 0;
        clk =0;
end
always begin
        #30 clk = !clk;
end
endmodule
```

SIMULATION WAVEFORM OUTPUT:

Addition:

input 1: (250)=11111010

input 2: (200)=11001000

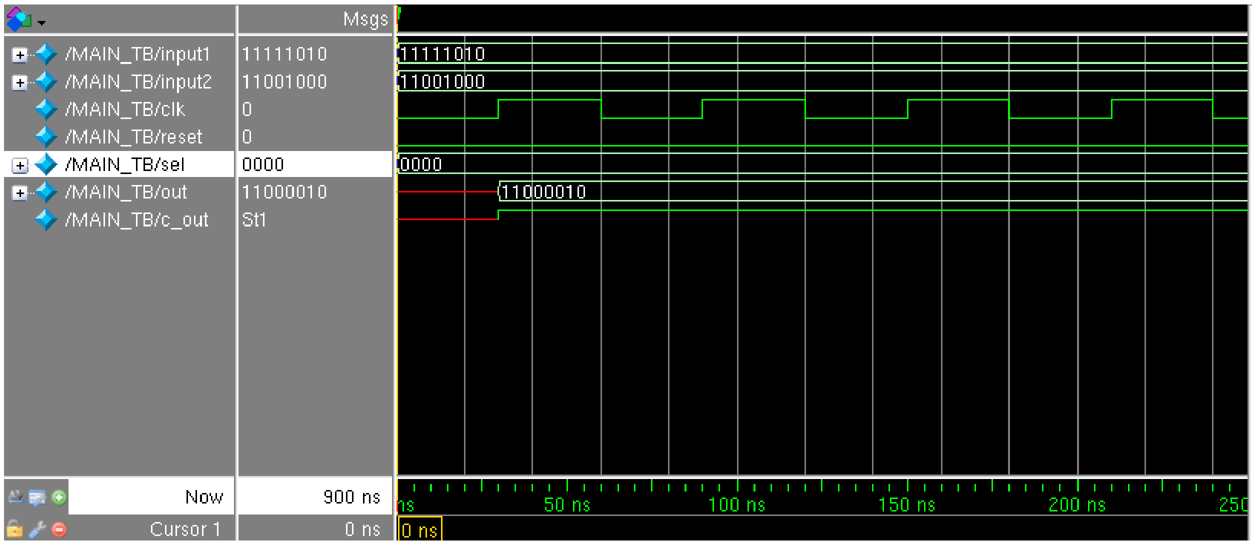
sel: (0000)

reset : 0

clk : 1

out :input1+input2=11000010

c_out: 1



Subtraction:

input 1: (250)=11111010

input 2: (200)=11001000

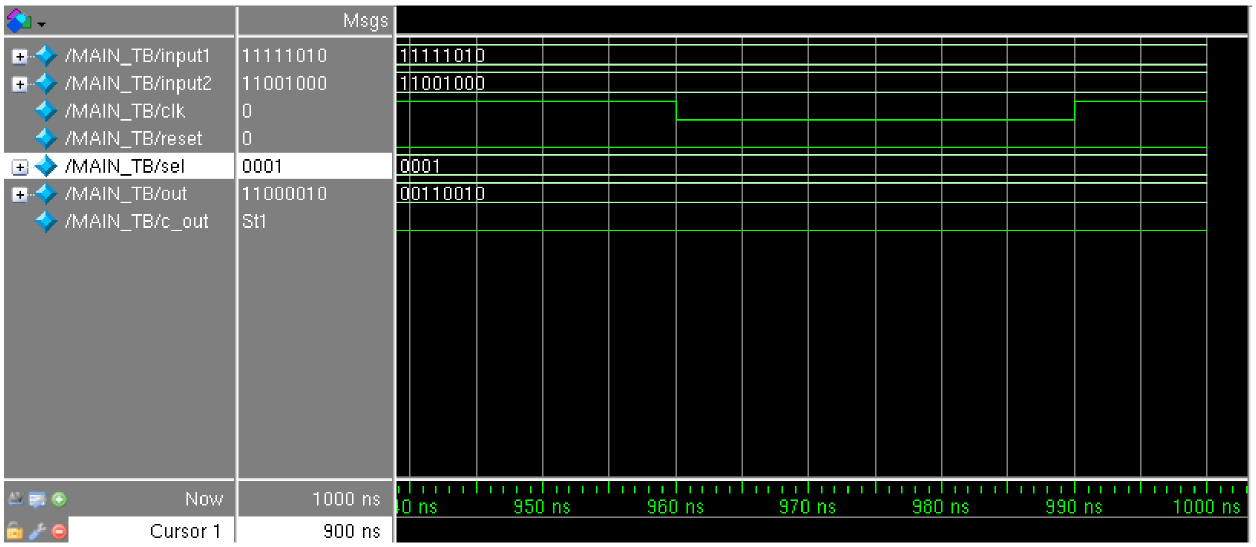
sel: (0001)

reset : 0

clk : 0

out :input1-input2=11000010

c_out: 1



```
                    Msgs
/MAIN_TB/input1     11111010    11111010
/MAIN_TB/input2     11001000    11001000
/MAIN_TB/clk        0
/MAIN_TB/reset      0
/MAIN_TB/sel        0001        0001
/MAIN_TB/out        11000010    00110010
/MAIN_TB/c_out      St1

Now                 1000 ns     0 ns   950 ns   960 ns   970 ns   980 ns   990 ns   1000 ns
Cursor 1            900 ns
```

Logical AND:

input 1: (250)=11111010

input 2: (200)=11001000

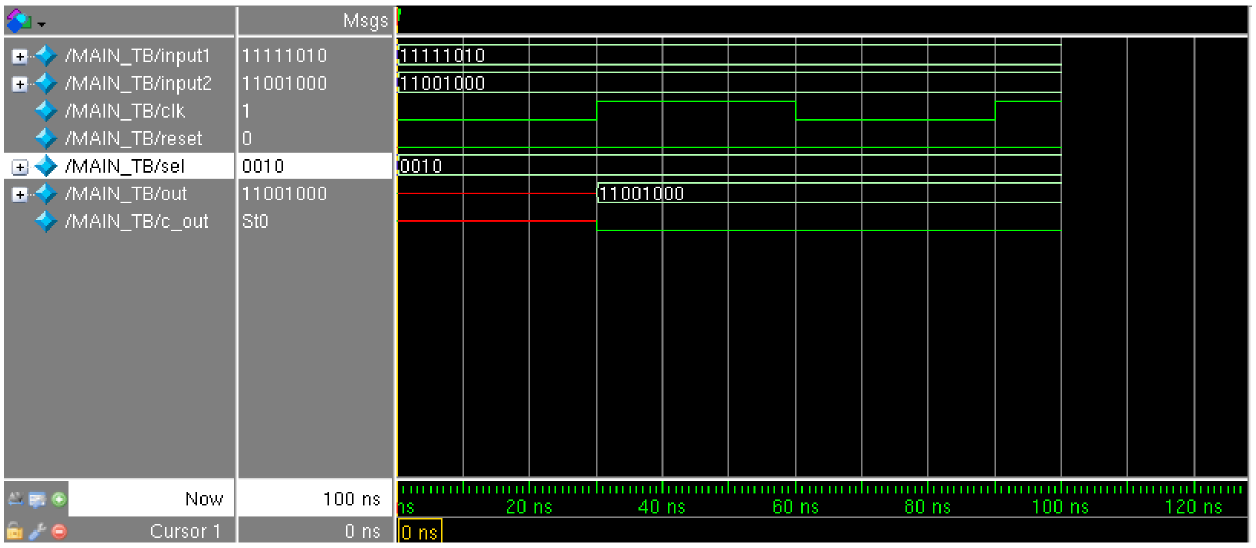
sel: (0010)

reset : 0

clk : 1

out :input1&input2=11001000

c_out: 0



```
/MAIN_TB/input1   11111010
/MAIN_TB/input2   11001000
/MAIN_TB/clk      1
/MAIN_TB/reset    0
/MAIN_TB/sel      0010
/MAIN_TB/out      11001000
/MAIN_TB/c_out    St0
Now               100 ns
Cursor 1          0 ns
```

Logical OR:

input 1: (250)=11111010

input 2: (200)=11001000

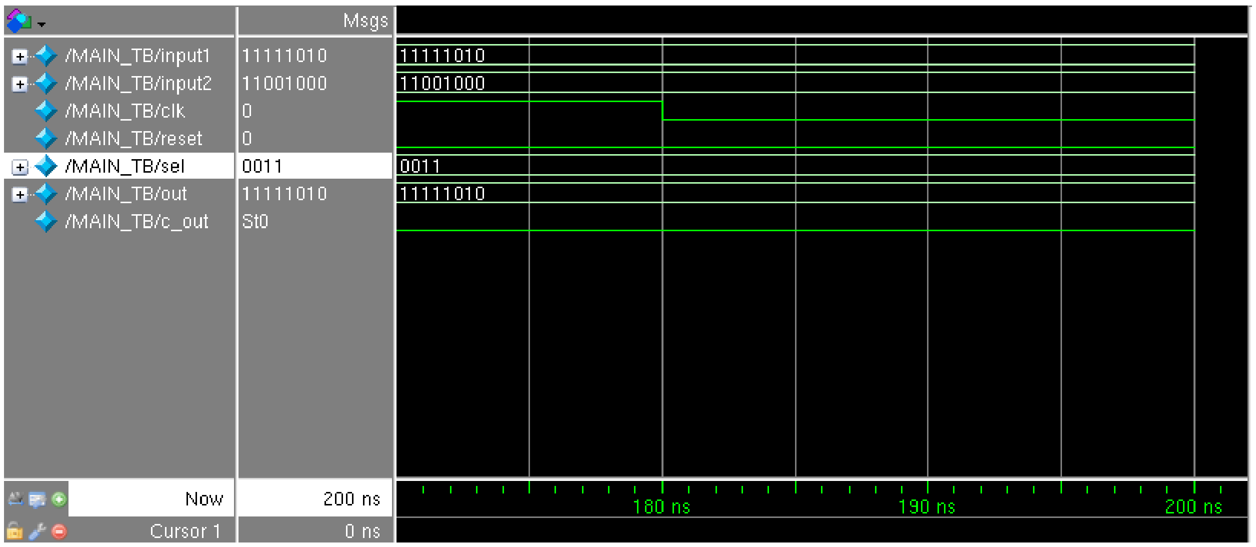
sel: (0011)

reset : 0

clk : 0

out :input1|input2=11111010

c_out: 0



```
/MAIN_TB/input1   11111010
/MAIN_TB/input2   11001000
/MAIN_TB/clk      0
/MAIN_TB/reset    0
/MAIN_TB/sel      0011
/MAIN_TB/out      11111010
/MAIN_TB/c_out    St0
```

Shift left:

input 1: (250)=11111010

input 2: (200)=11001000

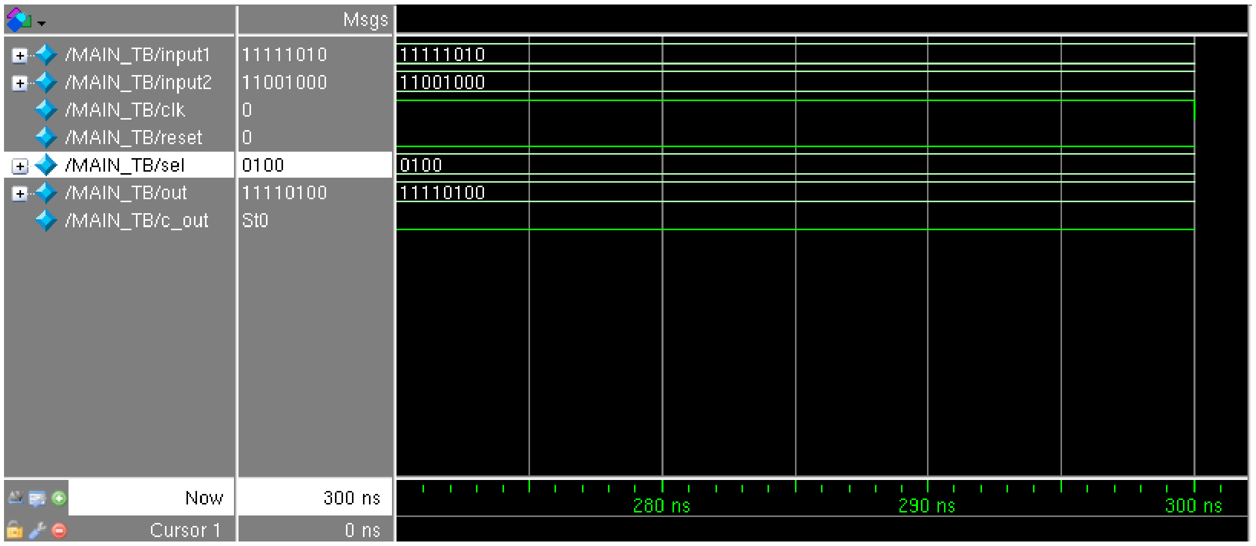
sel: (0100)

reset : 0

clk : 0

out :input1<<1=11110100

c_out: 0



```
                 Msgs
/MAIN_TB/input1  11111010   11111010
/MAIN_TB/input2  11001000   11001000
/MAIN_TB/clk     0
/MAIN_TB/reset   0
/MAIN_TB/sel     0100       0100
/MAIN_TB/out     11110100   11110100
/MAIN_TB/c_out   St0

Now              300 ns     280 ns   290 ns   300 ns
Cursor 1         0 ns
```

Shift right:

input 1: (250)=11111010

input 2: (200)=11001000

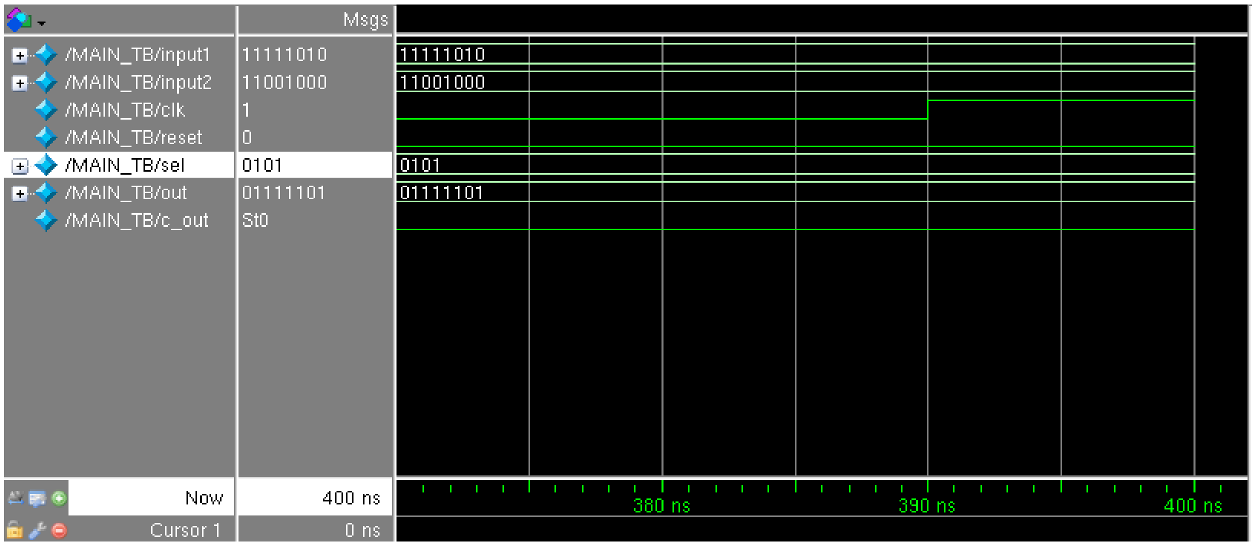
sel: (0101)

reset : 0

clk : 1

out :input1>>1=0111101

c_out: 0



Comparision :

input 1: (250)=11111010

input 2: (200)=11001000

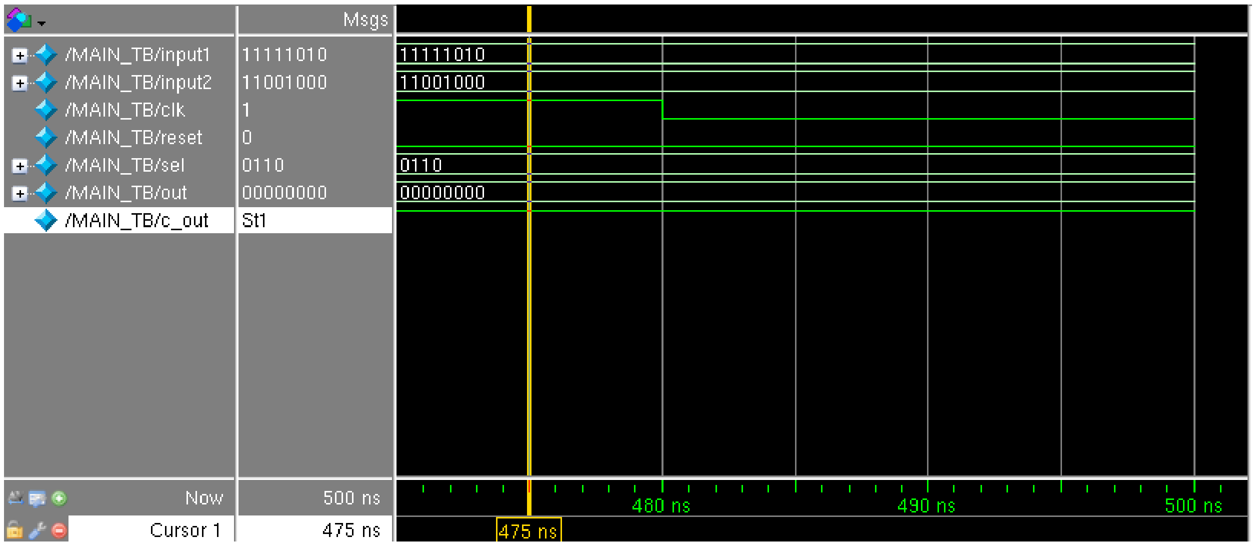
sel: (0110) for greater than

reset : 0

clk : 1

out :input1>input2=00000000

c_out: 1



```
/MAIN_TB/input1   11111010   11111010
/MAIN_TB/input2   11001000   11001000
/MAIN_TB/clk      1
/MAIN_TB/reset    0
/MAIN_TB/sel      0110       0110
/MAIN_TB/out      00000000   00000000
/MAIN_TB/c_out    St1
Now               500 ns
Cursor 1          475 ns
```

Lesser than:

input 1: (250)=11111010

input 2: (200)=11001000

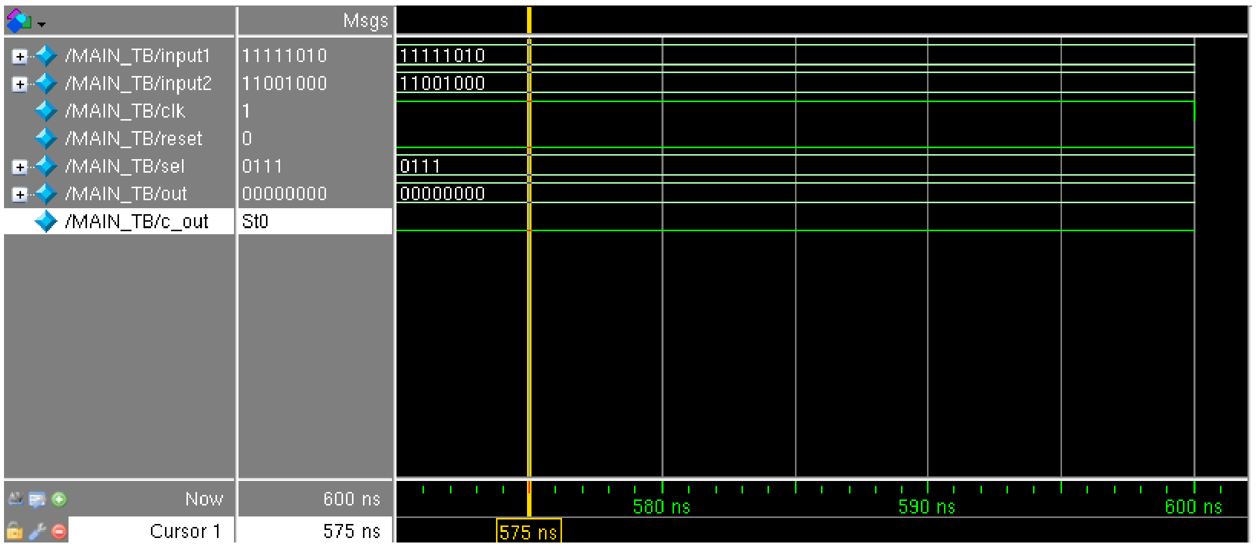
sel: (0111) for lesser than

reset : 0

clk : 1

out :input1<input2=00000000

c_out: 0



```
/MAIN_TB/input1   11111010
/MAIN_TB/input2   11001000
/MAIN_TB/clk      1
/MAIN_TB/reset    0
/MAIN_TB/sel      0111
/MAIN_TB/out      00000000
/MAIN_TB/c_out    St0
Now               600 ns
Cursor 1          575 ns
```

Equality:

input 1: (250)=11111010

input 2: (200)=11001000

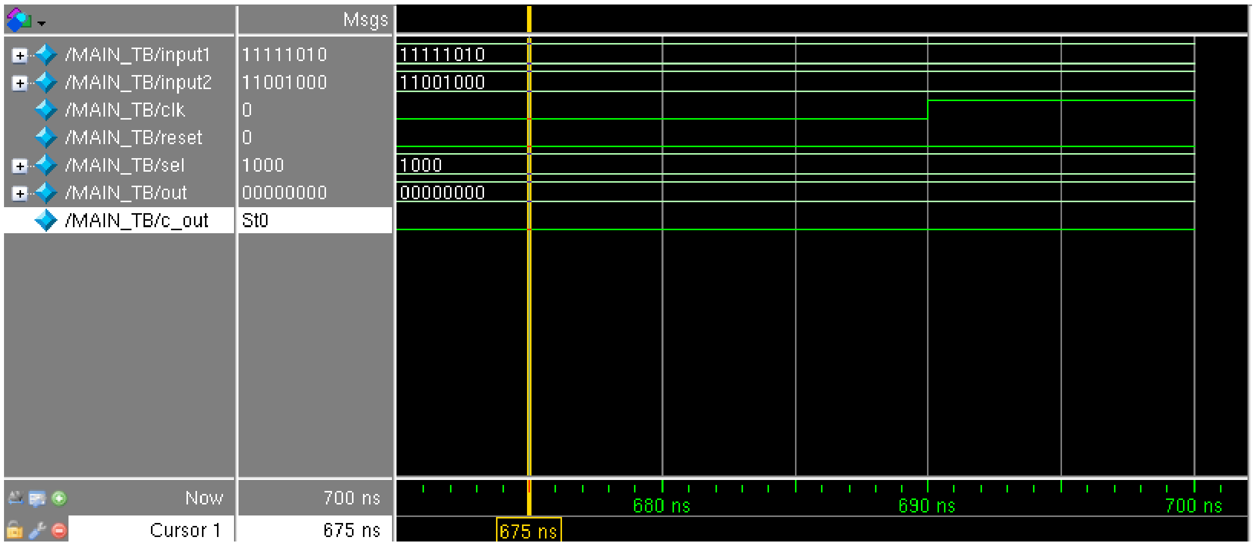
sel: (1000)

reset : 0

clk : 0

out :input1==input2=00000000

c_out: 0



```
                       Msgs
/MAIN_TB/input1   11111010     11111010
/MAIN_TB/input2   11001000     11001000
/MAIN_TB/clk      0
/MAIN_TB/reset    0
/MAIN_TB/sel      1000         1000
/MAIN_TB/out      00000000     00000000
/MAIN_TB/c_out    St0

Now               700 ns       680 ns      690 ns      700 ns
Cursor 1          675 ns       675 ns
```