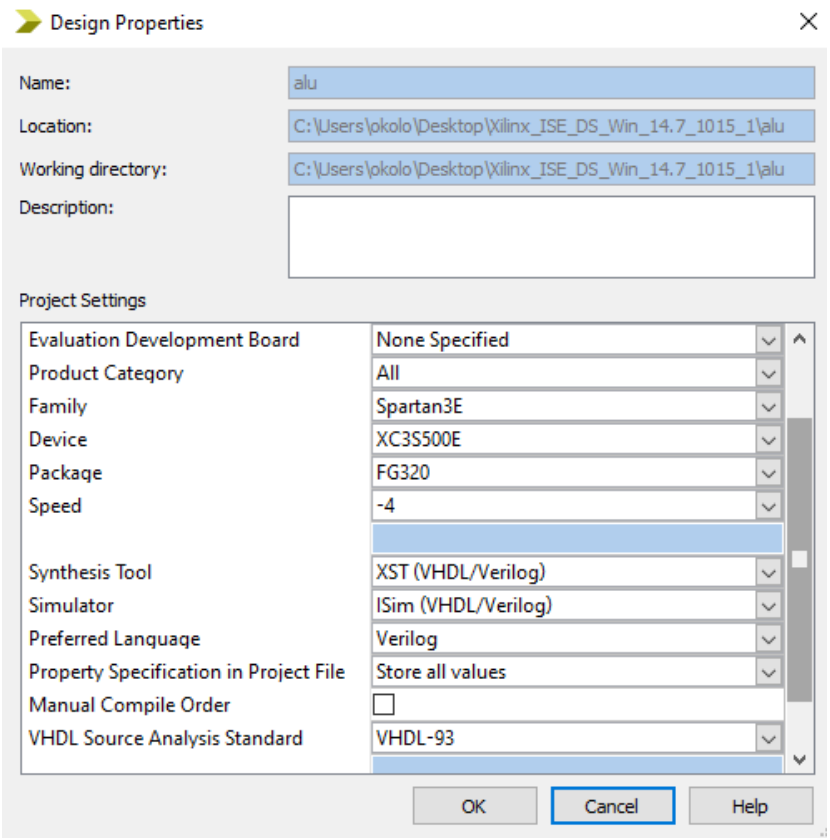
Computer Architecture and Organization 2020
Question and Answers

Student ID: 91815855E109
Student Name(in English): Miracle
Student Name(in Chinese): 奇迹

- **Summary of my work**

The first problem I encountered was downloading the software. Getting to the website page, there were a lot of files that looked like the right one, so I decided to watch a YouTube video to find the right one to download(even that was difficult too).
After downloading the software I read the Lab instruction which says that I should create a new project and call it alu and also add it to a location. When I got to the instruction of a working directory I was a little bit confused as to what the instruction meant by creating a working directory until I actually launched the software and followed does instructions. The instruction also said that I should add the following properties in the project settings: Spartan3E, XC3S500E,FG320,Speed of -4, Synthesis tool XST(VHDL/Verilog) Simulator:Ism(VHDL/Verilog), and preferred Language : Verilog.

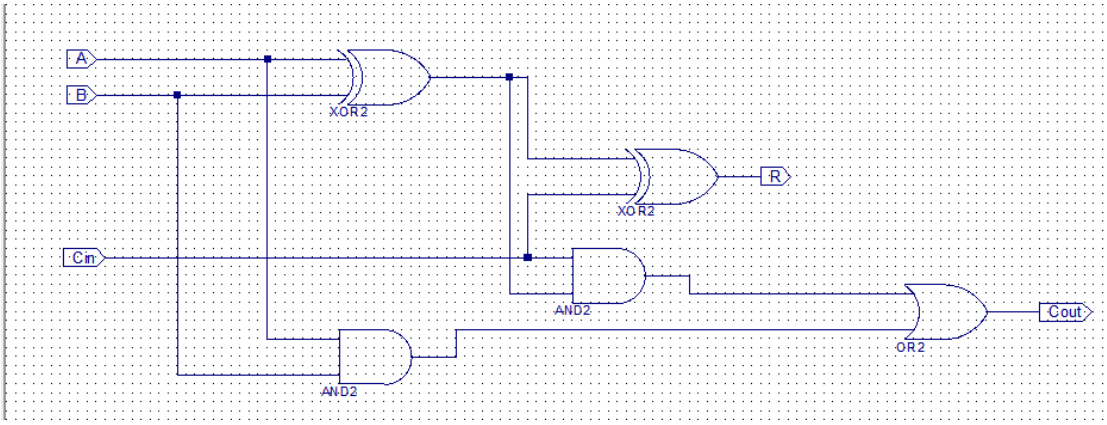


The picture of the properties

I was then instructed to create a new source file to named add1b.sch of which I did.After this instruction, came the challenge of understanding how to implement the following logic:

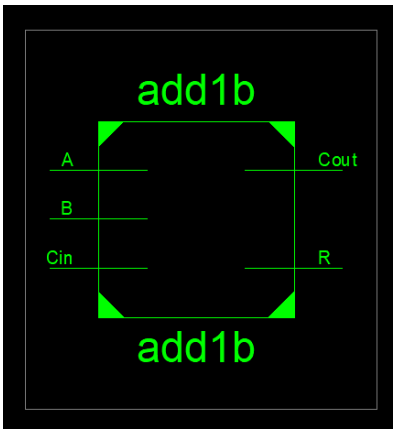
$$r = ci \cdot \bar{a} \cdot \bar{b} + \bar{ci} \cdot a \cdot \bar{b} + \bar{ci} \cdot \bar{a} \cdot b + ci \cdot a \cdot b$$
$$co = ci \cdot a + ci \cdot b + a \cdot b$$

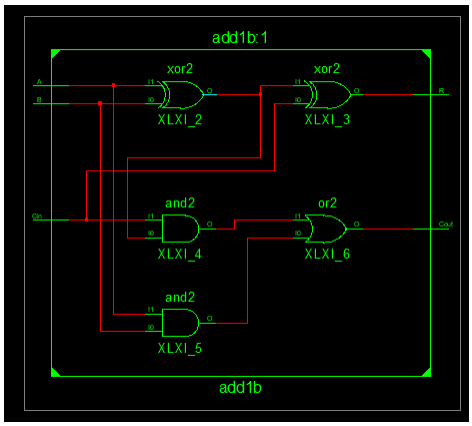
Where a and b were inputs to the adder , ci was the carry-in, r was the result, and co was the carry-out. Following the requirements that were given to me, it took a lot of time to process what I was meant to do. After lots of research from the internet I realized that what I was expected to implement was a 1-bit adder of which I did:
The specific instruction was to do this using logic gates.



My Schematic design for 1-bit adder.

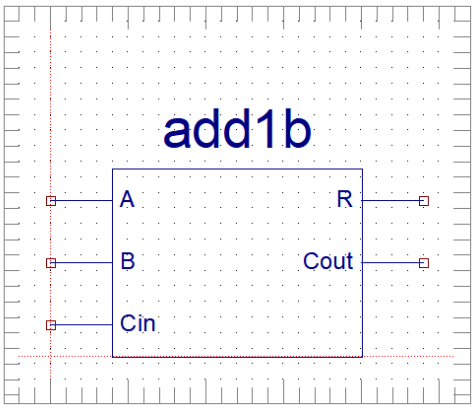
Then I was told to Synthesize my design to correct any errors. The RTL schematic is shown below: When you left-double click it looks like the picture on the right.





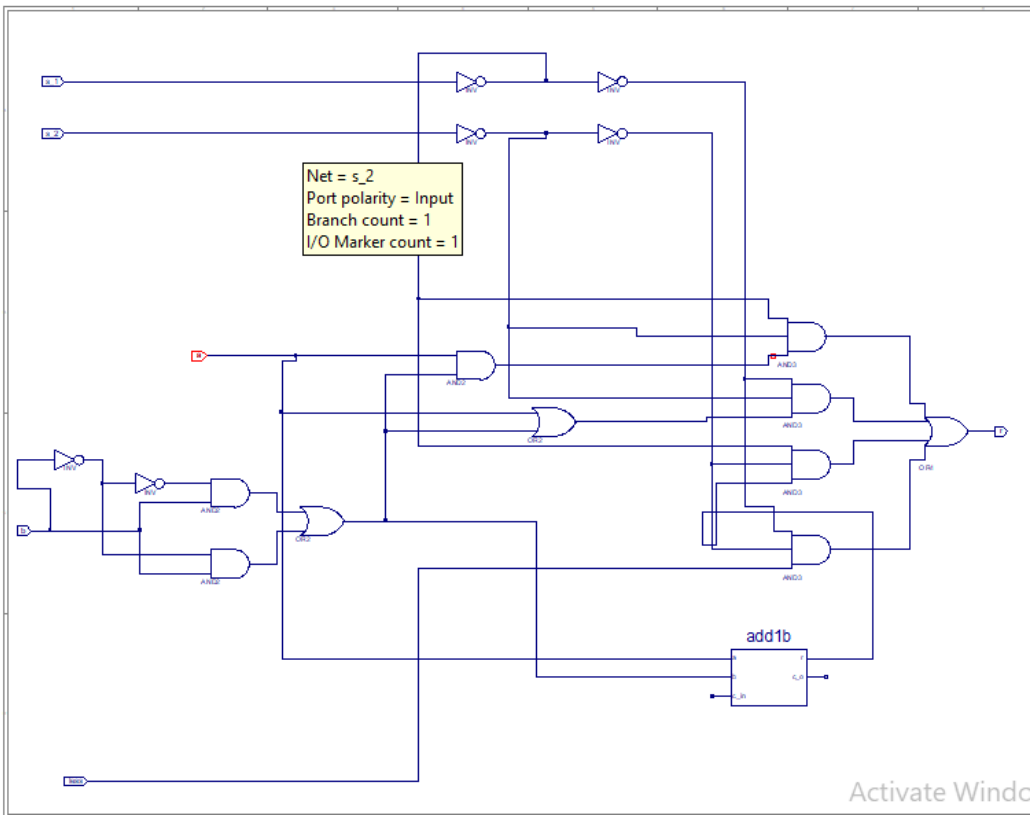
Next part of the lab was to create a test bench this part required me to understand verilog and vhdl programming .
This part will be further discussed in the next section.

I was told to create a schematic symbol for my add1b.sch of which I did.



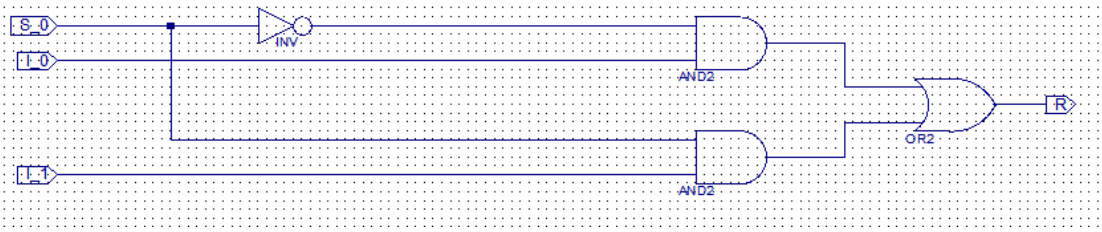
The Diagram of Schematic symbol.

Using this schematic symbol I was to implement a 1-bit ALU which I did but this required the use of a 2 to 1 multiplexer connected to b input and selector as b-inverted for the subtraction aspect of my ALU and a 4 to 1 multiplexer for selecting which operation to be done.

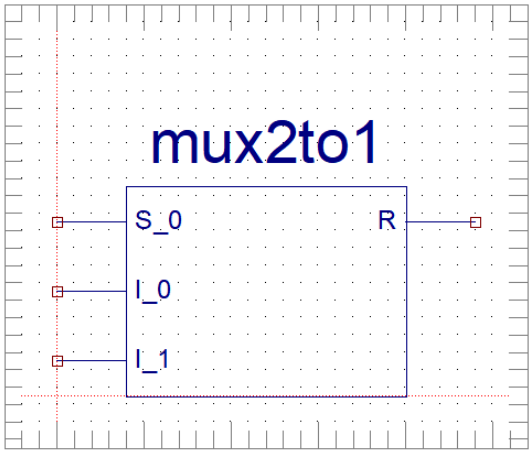


Initial Schematic of the 1-bit ALU

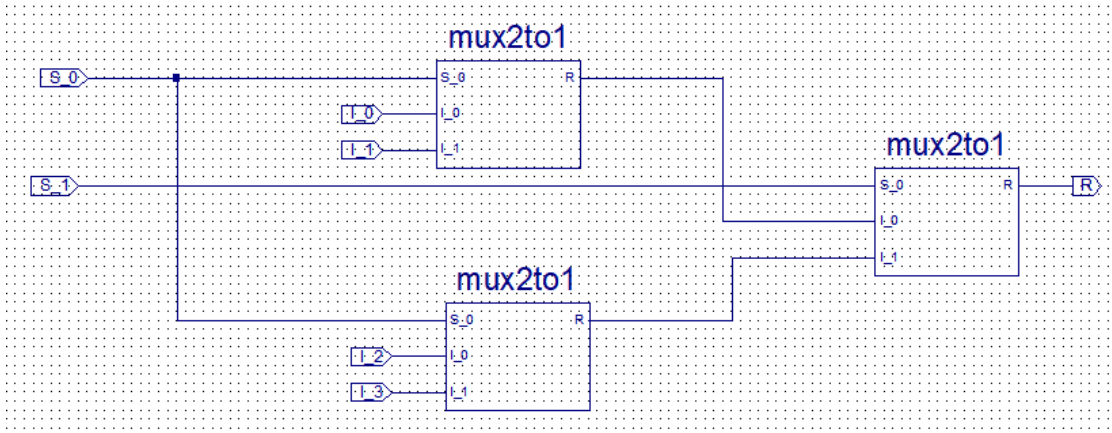
But as you can see it looks needlessly complicated, untidy and also not visible so I decided to create schematic symbols for each multiplexer.I created a new source file for each schematic For the 2 to 1 multiplexer I called mux2to1 and for the 4 to 1 multiplexer I called it mux4to1 as shown below:



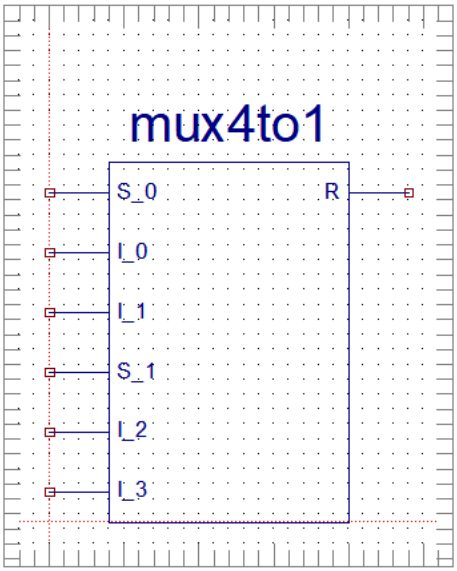
Schematic of 2 to 1 Multiplexer



Schematic Symbol of 2 to 1 multiplexer

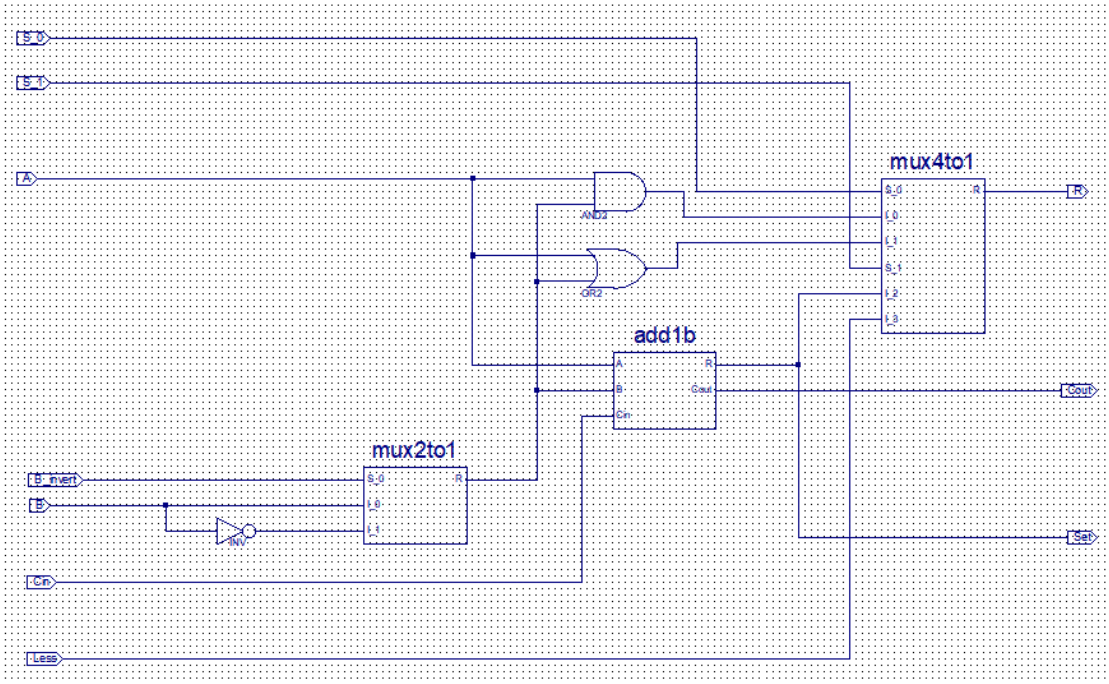


Schematic of 4 to 1 multiplexer

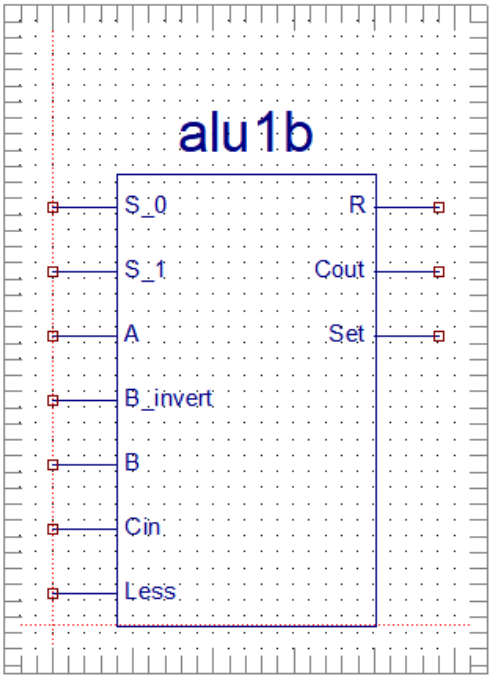


Schematic symbol of 4 to 1 multiplexer

From this I therefore restructured my ALU into a much tidy and less complicated Schematic.**I also realized that connecting the B_invert to B was wrong because it is meant to be a selector for Subtraction cases.**Less is used if the 4-bit A is less than the 4-bit B.Lastly, I added a set for my 4 -bit alu for cases to help implement the set less than operation.



Final Schematic of the 1-bit ALU



Schematic Symbol of 1-bit ALU

The test bench and Simulation will be discussed in the other sections.

- **VHDL/Verilog code**

For this section I had to do a test bench as instructed which produced the Verilog and VHDL codes of the add1b and ALU.

Add1b: Verilog

In this section the instruction said that I should replace a certain part of the verilog code form the Verilog Test Fixture to the one in the instruction manual and **also add my own code which yielded** to this:

```
`timescale 1ns / 1ps

module add1b_add1b_sch_tb();

// Inputs
  reg Cin;
  reg A;
  reg B;

// Output
  wire R;
  wire Cout;

// Bidirs

// Instantiate the UUT
  add1b UUT (
                    .Cin(Cin),
                    .A(A),
                    .B(B),
                    .R(R),
                    .Cout(Cout)
  );
// Initialize Inputs

    initial begin
                    Cin = 0;
                    A = 0;
                    B = 0;
                    #100;
                    A = 1;
                    B = 1;
                    #200;
                    A = 0;
                    B = 1;

                    #1;
                    if((R==1)&&(Cout==0))
                    $display("okay 1");
                    else
                    $display("fail 1");
end
endmodule
```

VHDL

This one is gotten from the VHDL test bench.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY add1b_add1b_sch_tb IS
END add1b_add1b_sch_tb;
ARCHITECTURE behavioral OF add1b_add1b_sch_tb IS

  COMPONENT add1b
  PORT( Cin       :          IN        STD_LOGIC;
        A  :          IN        STD_LOGIC;
        B  :          IN        STD_LOGIC;
        R  :          OUT       STD_LOGIC;
        Cout        :          OUT       STD_LOGIC);
  END COMPONENT;

  SIGNAL Cin     :          STD_LOGIC;
  SIGNAL A       :          STD_LOGIC;
  SIGNAL B       :          STD_LOGIC;
  SIGNAL R       :          STD_LOGIC;
  SIGNAL Cout    :          STD_LOGIC;

BEGIN

  UUT: add1b PORT MAP(
                    Cin => Cin,
                    A => A,
                    B => B,
                    R => R,
                    Cout => Cout
  );

-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
    WAIT; -- will wait forever
  END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

ALU: Verilog

I took the liberty of also displaying my code since I was asked to test bench.This result shall be shown in the Simulation section.

```
`timescale 1ns / 1ps

module alu1b_alu1b_sch_tb();

// Inputs
  reg S_0;
  reg S_1;
  reg A;
  reg B;
  reg Cin;
  reg Less;
  reg B_invert;

// Output
  wire Set;
  wire Cout;
  wire R;

// Bidirs

// Instantiate the UUT
  alu1b UUT (
                .Set(Set),
                .S_0(S_0),
                .S_1(S_1),
                .A(A),
                .B(B),
                .Cin(Cin),
                .Less(Less),
                .B_invert(B_invert),
                .Cout(Cout),
                .R(R)
  );
// Initialize Inputs

    initial begin
                S_0 = 0;
                S_1 = 0;
                A = 0;
                B = 0;
                Cin = 0;
                Less = 0;
                B_invert = 0;
                //AND operation
                #100;
                S_0 = 0;
                S_1 = 0;
                A = 1;
                B = 1;
                Cin = 0;
                Less = 0;
                B_invert = 0;
                //OR operation
                #200;
                S_0 = 1;
                S_1 = 0;
                A = 0;
                B = 1;
```

```
        Cin = 0;
        Less = 0;
        B_invert = 0;
        //ADD operation
        #300;
        S_0 = 0;
        S_1 = 1;
        A = 0;
        B = 1;
        Cin = 0;
        Less = 1;
        B_invert = 1;
        //Subtract operation
        #400;
        S_0 = 0;
        S_1 = 0;
        A = 1;
        B = 1;
        Cin = 1;
        Less = 0;
        B_invert = 1;


end
endmodule
```

VHDL

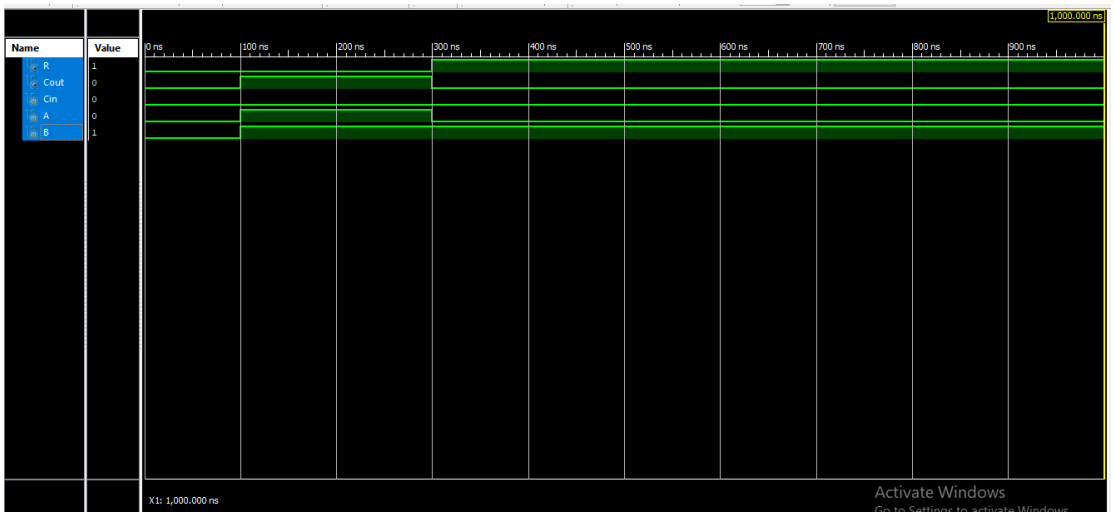
This is the Vhdl code also from the test bench.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY alu1b_alu1b_sch_tb IS
END alu1b_alu1b_sch_tb;
ARCHITECTURE behavioral OF alu1b_alu1b_sch_tb IS

  COMPONENT alu1b
  PORT( Set       :          OUT     STD_LOGIC;
        S_0       :          IN      STD_LOGIC;
        S_1       :          IN      STD_LOGIC;
        A  :       IN      STD_LOGIC;
        B  :       IN      STD_LOGIC;
        Cin       :          IN      STD_LOGIC;
        Less      :       IN      STD_LOGIC;
        B_invert  :       IN      STD_LOGIC;
        Cout      :       OUT     STD_LOGIC;
        R  :       OUT     STD_LOGIC);
  END COMPONENT;

  SIGNAL Set    :       STD_LOGIC;
  SIGNAL S_0    :       STD_LOGIC;
  SIGNAL S_1    :       STD_LOGIC;
  SIGNAL A      :       STD_LOGIC;
  SIGNAL B      :       STD_LOGIC;
  SIGNAL Cin    :       STD_LOGIC;
  SIGNAL Less   :       STD_LOGIC;
  SIGNAL B_invert       :       STD_LOGIC;
  SIGNAL Cout   :       STD_LOGIC;
  SIGNAL R      :       STD_LOGIC;

BEGIN

  UUT: alu1b PORT MAP(
                    Set => Set,
                    S_0 => S_0,
                    S_1 => S_1,
                    A => A,
                    B => B,
                    Cin => Cin,
                    Less => Less,
                    B_invert => B_invert,
                    Cout => Cout,
                    R => R
  );

-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
    WAIT; -- will wait forever
  END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

- **Simulation Results**

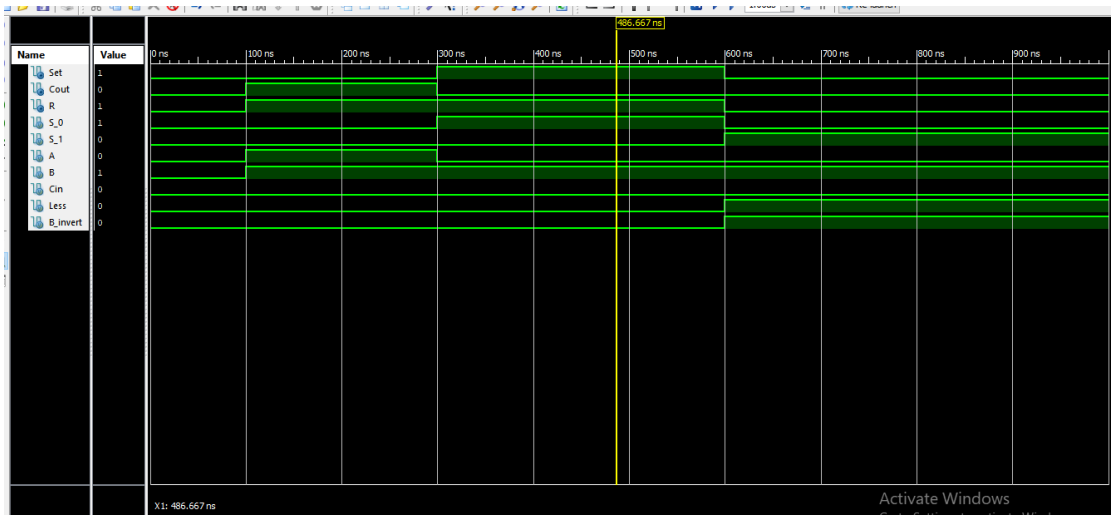
Add1b:

The result below is gotten from the Verilog code of the 1-bit adder



Wave form

1-bit ALU:



Wave form

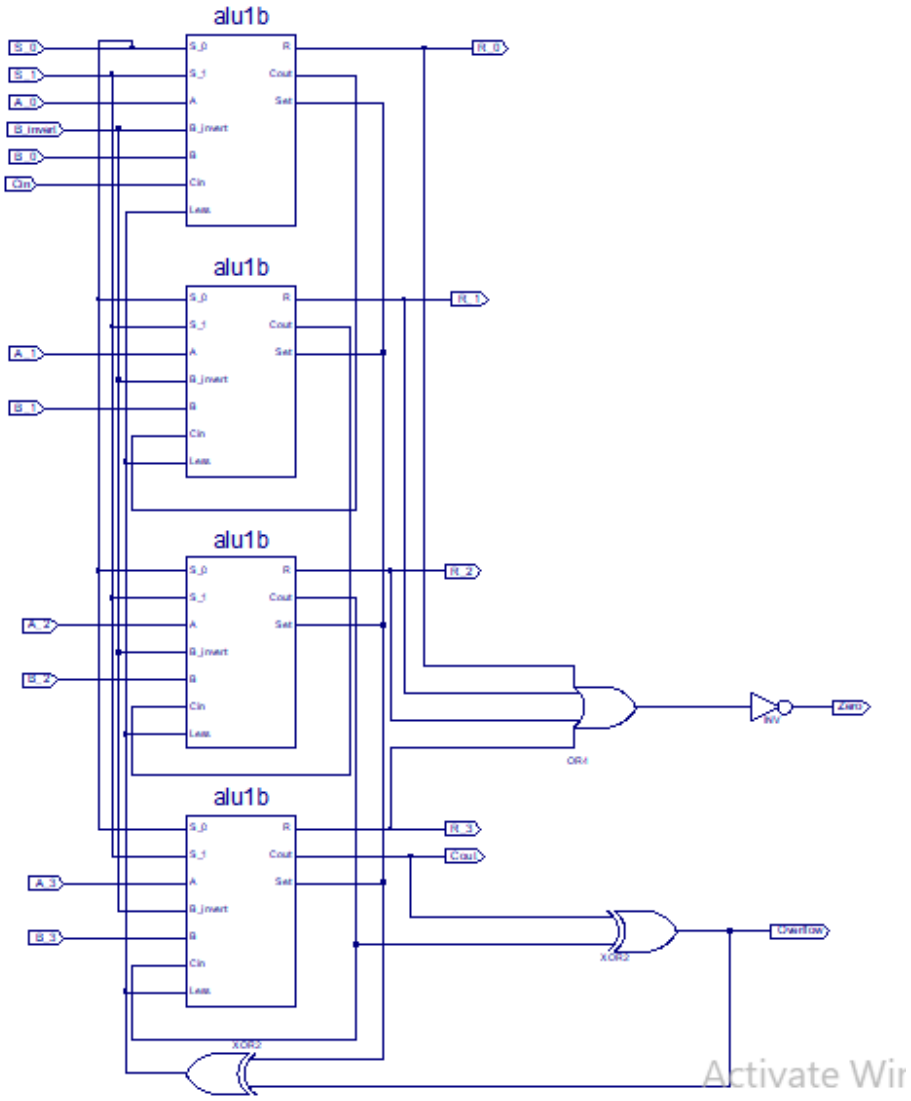
4-bit ALU

The instruction asked for:

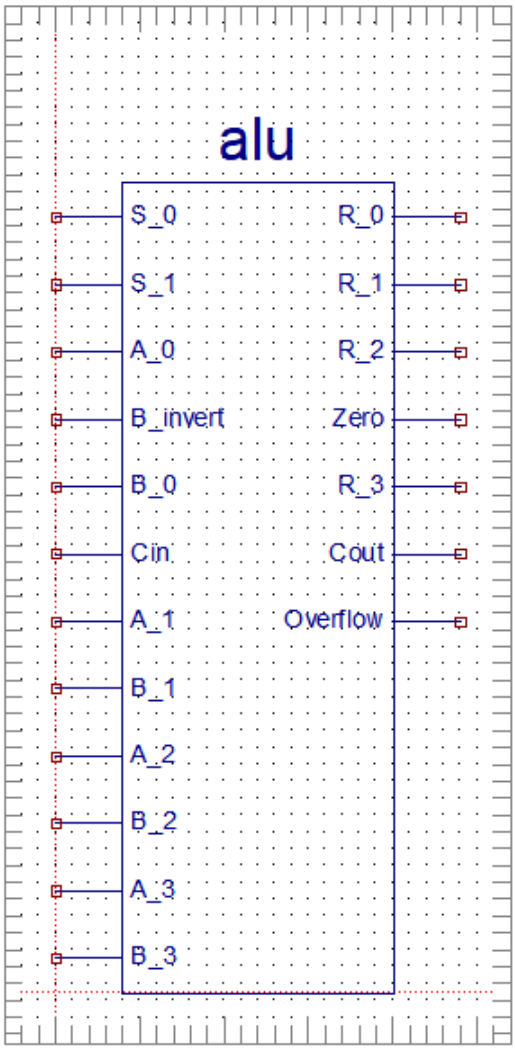
- Two 4-bit inputs which are A_0 to A_3 and B_ to B_3
- 3-bit input operation which B_invert ,S_0 and S_1
- 4 - bit output to R_0 to R_3
- Two 1-bit output Zero and Overflow

Less and Set is not shown because they are connected to each other

Note: The Set is JOINED with Overflow in XOR and then their Output is joined with the Less.



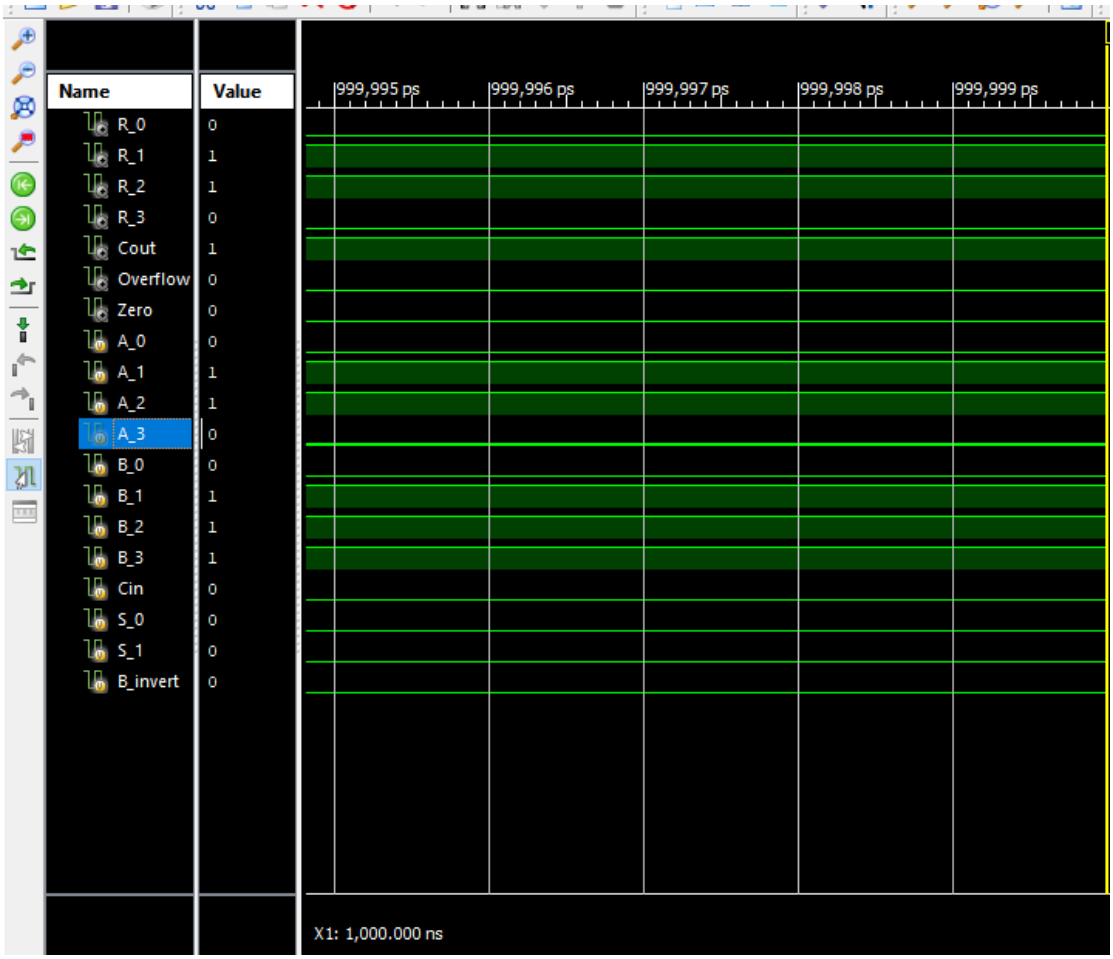
Schematic Diagram



Schematic Symbol

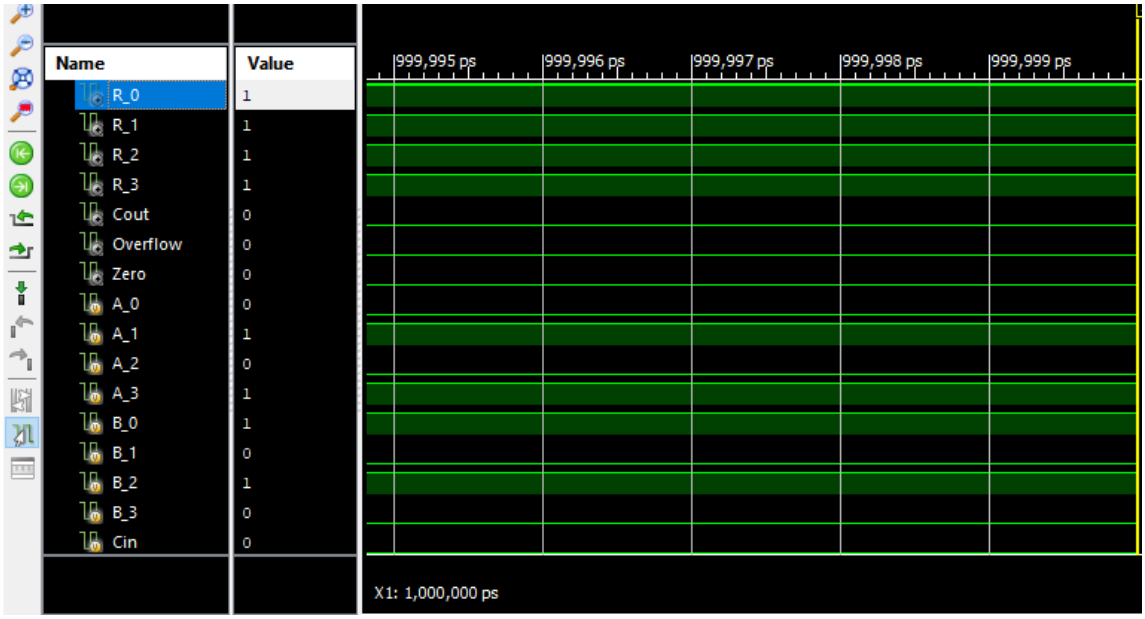
- **Operations**

AND(000):
I shall use **A_3** = 0,**A_2** = 1,**A_1** = 1,**A_0** = 0 and **B_3** = 1,**B_2** = 1,**B_1** = 1,**B_0** = 0.
This IS 0110 AND 1110 which the result is 0110.



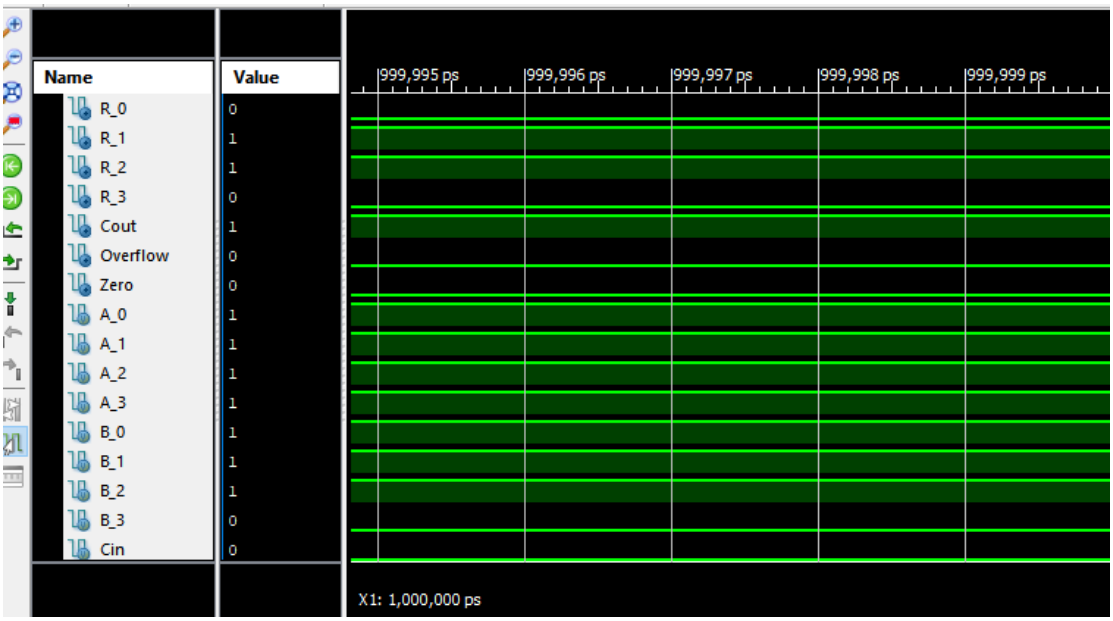
Wave Form of Operation Result

OR(001):
I shall use **A_3** = 1,**A_2** = 0,**A_1** = 1,**A_0** = 0 and **B_3** = 0,**B_2** = 1,**B_1** = 0,**B_0** = 1.
This IS 1010 OR 0101 which the result is 1111.



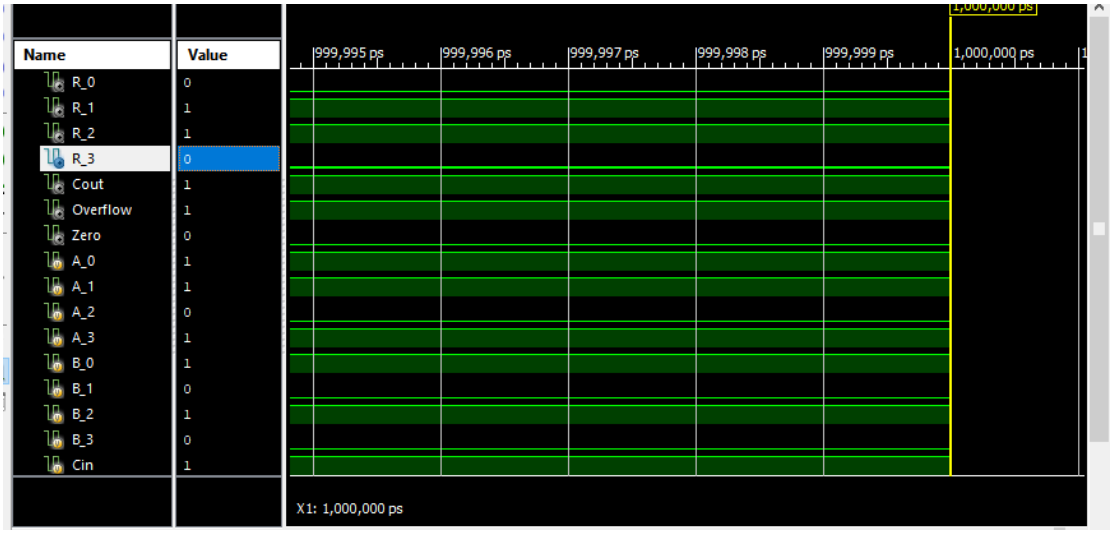
Wave Form of Operation Result

ADDITION(010):
I shall use **A_3** = 1,**A_2** = 1,**A_1** = 1,**A_0** = 1 and **B_3** = 0,**B_2** = 1,**B_1** = 1,**B_0** = 1.
This IS 1111 ADD 0111 which the result is 0110 with Carry-out 1.



Wave Form of Operation Result

SUBTRACTION(110):
I shall use **A_3** = 1,**A_2** = 0,**A_1** = 1,**A_0** = 1 and **B_3** = 0,**B_2** = 1,**B_1** = 0,**B_0** = 1.
This IS 1011 SUBTRACT 0101 which the result is 0110.For this B_invert is equals 1 and Carry_in is equals 1.



SET LESS THAN(111): This operation is done internally.Please take a look at the Schematic.
(If A < B then Set =1 else Set =0)