“ADDER”, “SUBTRACTOR”, “OR”, AND “AND” CALCULATOR USING 4:1 MULTIPLEX

[Using a 4:1 multiplex to do computations]

APRIL 20, 2021

SAGE-MICHAEL BROWN
CST_213.2

Contents

Table of Tables

Table of Figures

Objectives

The objective of this project was to create a simple calculator using a multiplexer, and logic gates to create values and a calculated output to show that the calculator is performing the function as required. Then to reciprocate this action using VHDL coding to prove the results in the Multisim simulation. The values input into the multiplex can be either a value of 0 or 1, resulting in an identifier to tell that the value is high or low, for this project a blue probe was used at the end of each device for the SUM/DIFF value for each of the logic circuits, while also producing an out value which would be used in the case of multiple adders or subtractors in a row.

Parts List

Multisim, and Modelsim

Presentation link

https://drive.google.com/file/d/1WnJSLVES3OlgIeR4VZ7vRzzk8j_yzcFs/view?usp=sharing

Introduction

What is a multiplexer?

A multiplexer is an electrical device that can select either analog or digital signals, directed by a separate input resulting in a singular output. The multiplexer used in this project being the 4:1 multiplexer, uses 4 different selectors, and 2 inputs to give a signal to the 2 bulbs attached at the end for the Y and ~W portions of the device.

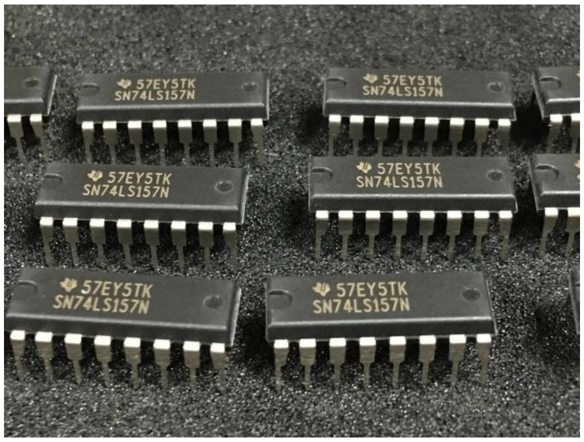


Figure 1: Physical Multiplex

Why is a multiplexer important?

When a person uses an electronic that has multiple controls and setting, the implication of the multiplexer allows for the selection of those different values to send a value whether it is a value of 1 or 0, but the main function of the output is different from the one initially selected, things such as controllers and thermostats rely on these to make selections between different functions of each device.

With the use of an adder, and subtractor the creation of a bit calculator can be made, giving the required value that is needed from the output, the more the circuit changes in terms of switches connecting the power supply to the ends of each gate, will determine the final output value once completed. This can simulate the act of using a changing variable that has only two possible outcomes, but multiple different functions to come to those values prior to it.

Procedure

Mainly because the physical components such as a motherboard and wiring are not available at the time the use of Multisim was the best substitute to replicate the project. This gives accurate custom value and allows for me as the controlling person to influence the circuit and change perform changes if need be.

Results and Analysis

Digital Circuit 1: Full Adder

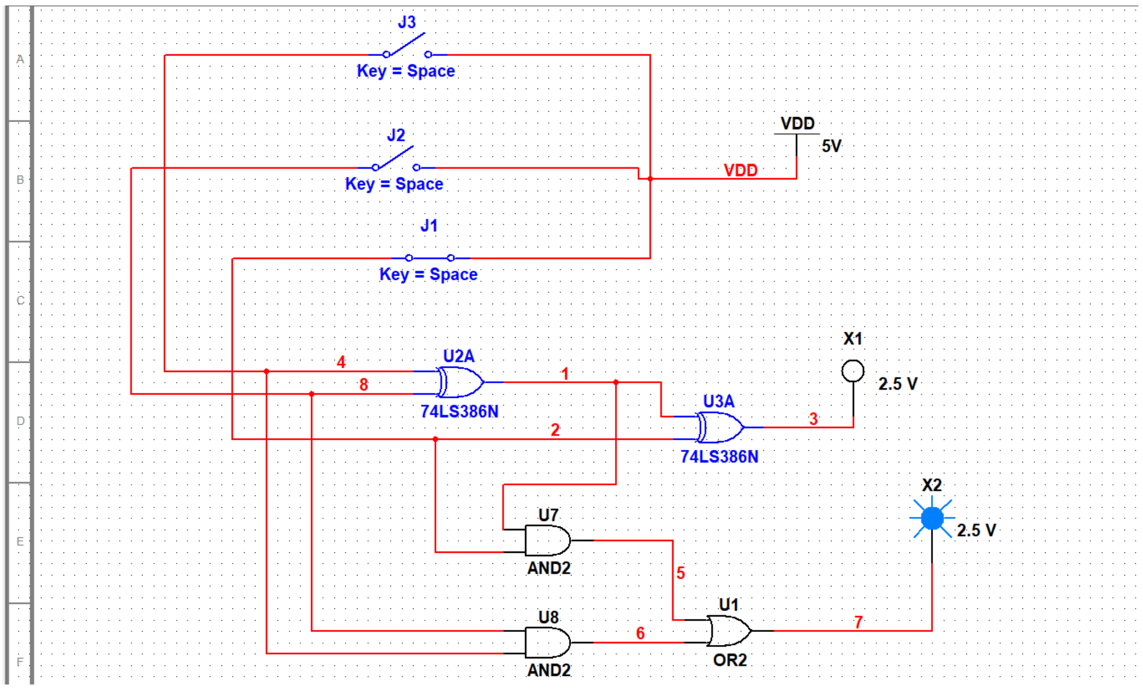


Figure 2:Full adder

The logic circuit above just like others inside of the full circuit uses a VDD power supply to power all the gates inside of the circuit. This diagram is what is used as the addition part of the calculator, it calculates the values submitted from powering the three inputs being at line numbers four, eight and two. Whenever the value submitted to line four and eight changes, the value for 1 change while line two stays constant with line 4 and line one changing only if there are changes in lines four and eight. The result then converges at line 3 in which if the value is positive the light will be lit and if negative the light will remain dormant.

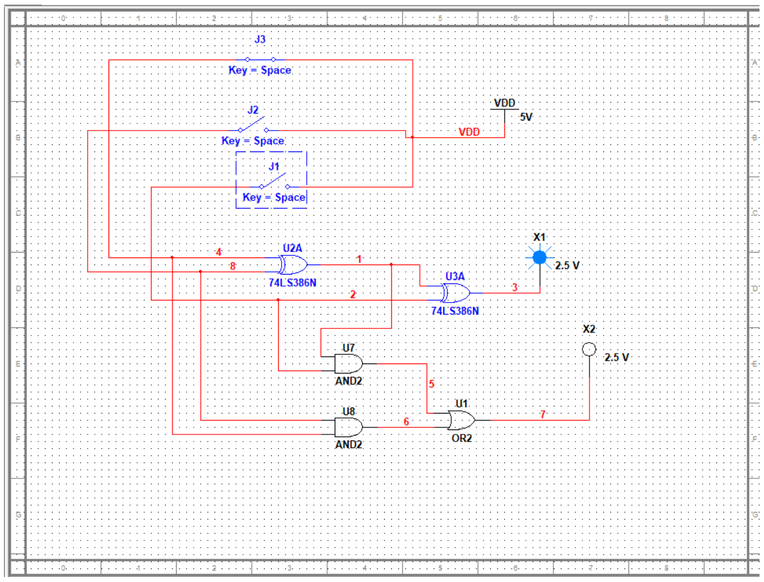


Figure 3:Full Adder Light Shine

The difference in this image being that J1 and J2 are open while J3 is closed giving a value of positive making the light shine. Using the line number as a subscript to detail which line is used.

A_4	B_8	$(A_4 \oplus B_8)$
0	0	0

0	1	1
1	0	1
1	1	0

Table 1:U2A Exclusive OR Gate Truth Table

$(A_4 \oplus B_8)$	C_{in}	$(A_4 \oplus B_8) \oplus C_{in}=X1$
0	1	1
1	0	1
1	1	0
0	0	0

Table 2: U3A Exclusive OR Gate Truth Table

Using the truth table, the values are factually represented proving the truth table. The second part of the circuit being the C_{out} uses two parallel AND gates in series with a OR gate, to produce to a value or 1 or 0, and if we were using multiple adders this value could be used over and over to give a value to another adder, this C_{out} now becomes the new C_{in} for the connecting device.

$(A_4 \oplus B_8)$	C_{in}	$(A_4 \oplus B_8) * C_{in}$
0	1	0
1	0	0
1	1	1
0	0	0

Table 3:U7 AND Gate Truth Table

A_4	B_8	$A_4 * B_8$
0	0	0
0	1	0

1	0	0
1	1	1

Table 4: U8 AND Gate Truth Table

$(A_4 \oplus B_8) * C_{in}$	$A_4 * B_8$	$(A_4 * B_8) + ((A_4 \oplus B_8) * C_{in}) = X2$
0	0	0
0	0	0
1	0	1
0	1	1

Table 5: U1 OR Gate Truth Table

That concludes for the Adder part of the circuit, click here for the VHDL code.

Logic Circuit 2: Subtractor

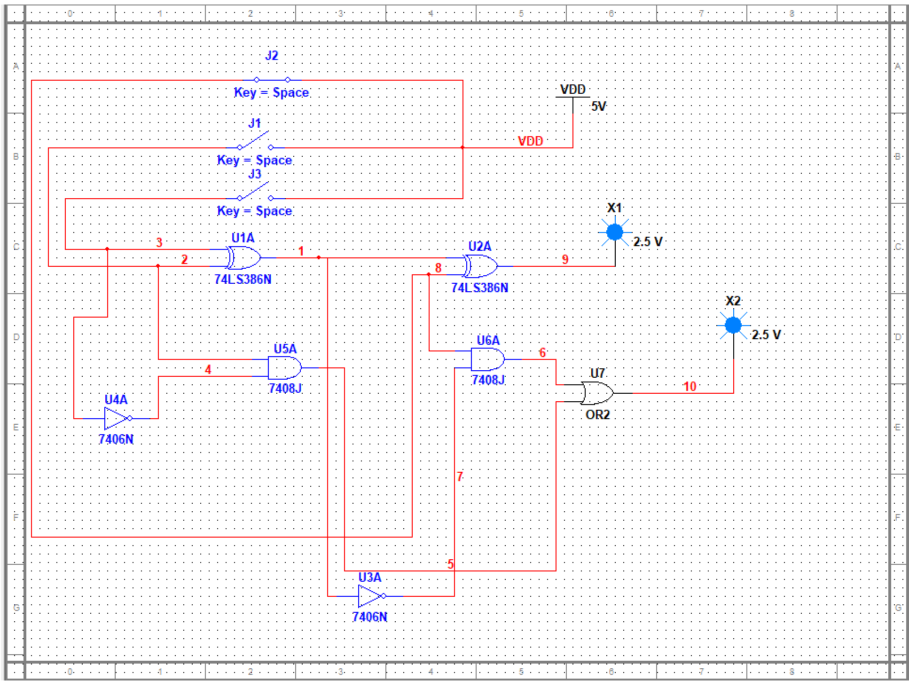


Figure 4:Subtractor Logic Gate

The subtractor aids in giving a value from the subtraction of two bits, using two exclusive OR(XOR) gates, 2 NOT gates, 2 AND gates and a single OR gate. Both the single OR gate and of the XOR gates provide a value in return to light the blue probe at the end. This is what was used to identify if the value could be a high or low output. As seen in the circuit above only 1 switch is closed, showing that the value is based on which pin receives input and will change if other switches lock.

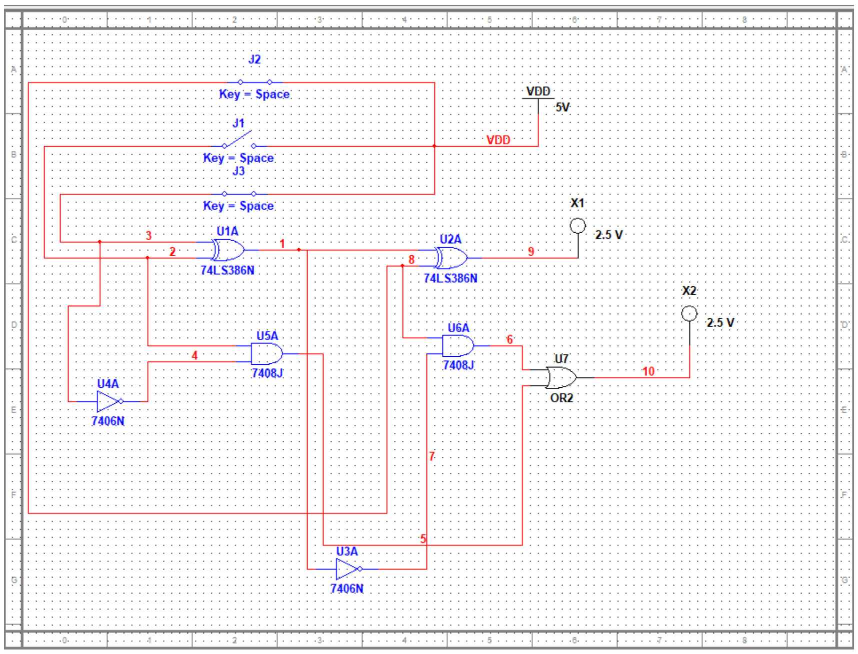


Figure 5:Full subtractor no lights

However, when two of the switches lock the light does not turn on showing the dynamic change within the circuit at the click of a button only because of the closing of a single switch on the diagram. This will be further discussed using its truth table as proof of it being the correct format.

A_3	B_2	$A_3 \oplus B_2$
0	0	0
1	1	0
1	0	1
0	1	1

Table 6:U1A Exclusive OR Gate Truth Table

$A_3 \oplus B_2$	C_{in}	$A_3 \oplus B_2 \oplus C_{in}$=DIFF.
0	0	0
0	1	1
1	0	1
1	1	0

Table 7:U2A Exclusive OR Gate Truth Table

A_3	A_3^{-1}
0	1
1	0
1	0
0	1

Table 8:U4A NOT Gate Truth Table

A_3^{-1}	B_2	A_3^{-1} * B2
1	0	0
0	1	0
0	0	0
1	1	1

Table 9:U5A AND Gate Truth Table

$(A_3 \oplus B_2)$	$(A_3 \oplus B_2)^{-1}$
0	1
0	1
1	0
1	0

Table 10:U3 NOT Gate Truth Table

$(A_3 \oplus B_2)^{-1}$	C_{in}	$(A_3 \oplus B_2)^{-1} * C_{in}$
1	0	0
1	1	1
0	0	0
0	1	0

Table 11:U6A AND Gate Truth Table

$(A_3 \oplus B_2)^{-1} * C_{in}$	$A_3^{-1} * B2$	$(A_3 \oplus B_2)^{-1} * C_{in} + A_3^{-1} * B2$
0	0	0
1	0	1
0	0	0
0	1	1

Table 12: U7 OR Gate Truth Table

That concludes for the subtractor part of the circuit, click here for the VHDL code.

Logic Circuit 3: 4:2 Encoder

The Encoder functions as the way for the values to be input into the multiplex to be used in the calculation for the multiplex. This type of input has 4 different possible outcomes being, 0 0,1 1,0 1 or 1 0. This is all based on the combination and are the only possible outcomes from this part of the circuit, and because it is the input, it determines what will be the final value once passed through the multiplex of the circuit.

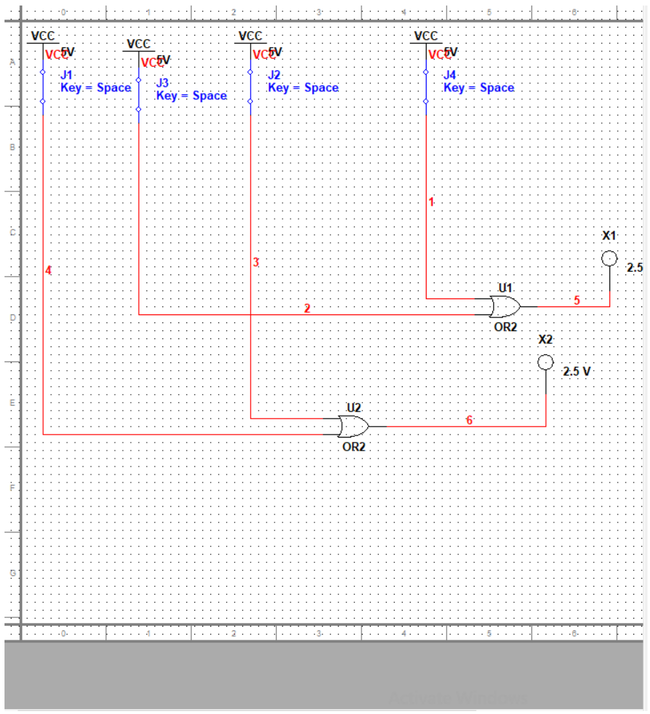


Figure 6: 4 to 2 Encoder part of circuit (paused)

The reason the values in the image above are not being lit, is due to the simulation not running now, and this is only to represent how the Encoder looks. This representation allows for the manipulation of both inputs by locking or opening each gate of each gate that is connected to the probe at the end of each.

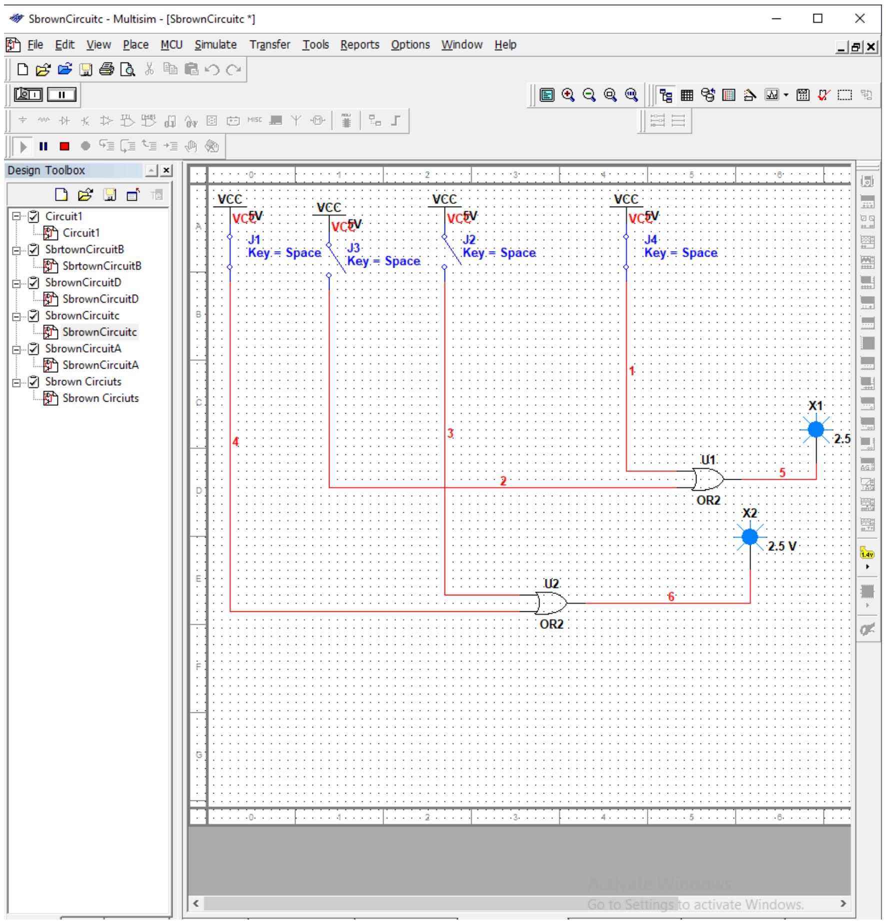


Figure 7: 4 to 2 Encoder double switch

For each probe only one switch needs to be turned on for a value of 1 to be the outcome, once realized, connected both ends to a singular switch to conserve space.

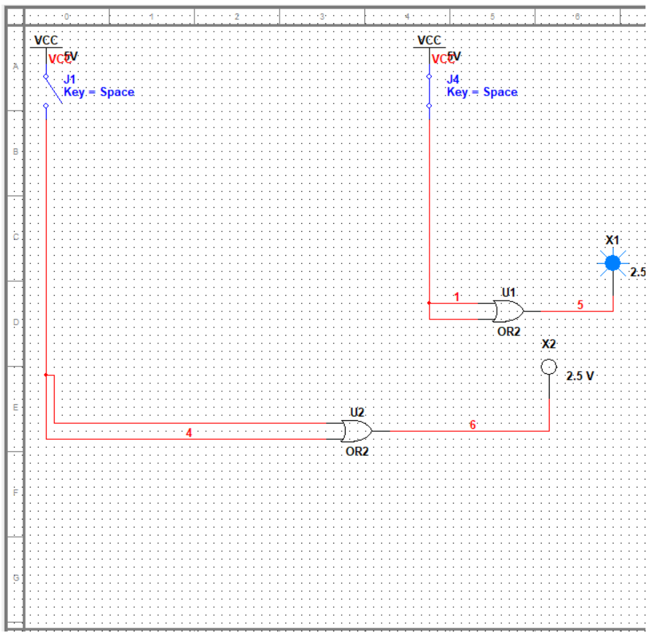


Figure 8: 4 to 2 Encoder Simple

With that the truth table can be derived and given simply using the switches as the input values, though there are only two possible outcomes from this, it does not affect the output due to it being a OR gate and the 0+1 and 1+1 will result in the same values, and only when a low is input will the output be 0.

J_1	J_1	J_1+J_1
1	1	1
0	0	0

Table 13: U1 OR Gate Truth Table

J_1	J_1	J_1+J_1
1	1	1
0	0	0

Table 14: U1 OR Gate Truth Table

That concludes for the 4:2 encoder part of the circuit, click here for the VHDL code.

Logic Circuit 4: Multiplex

This is the main part of the circuit that is where both the selector and the 4 inputs meet, to produce a single signal. This will encompass the whole circuit mainly due to the fact the multiplexer is a single item, unless made into more complex logic gate processes. The multiplex with the two values inputting in the lower selecting which mode whether it be the adder, subtractor, OR, or AND functions to process.

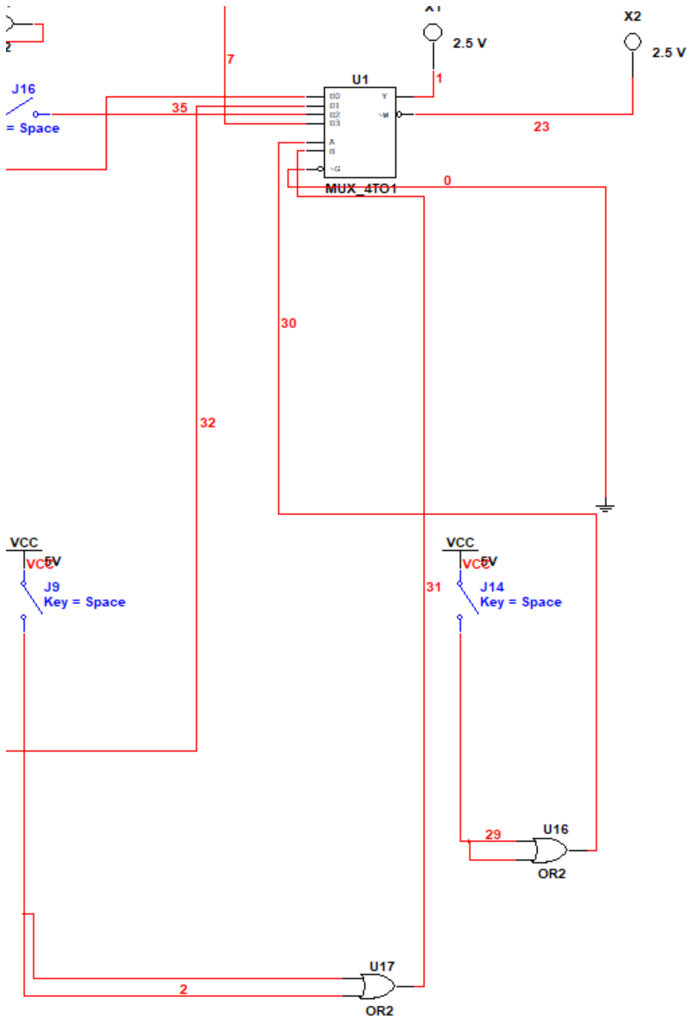


Figure 9:Multiplex With 4:2 Encoder Input

A	B	X
0	0	D_0(Adder)
0	1	D_1(Subtractor)
1	0	D_2(OR)
1	1	D_3(AND)

Table 15:4 to 2 Encoder/Selector

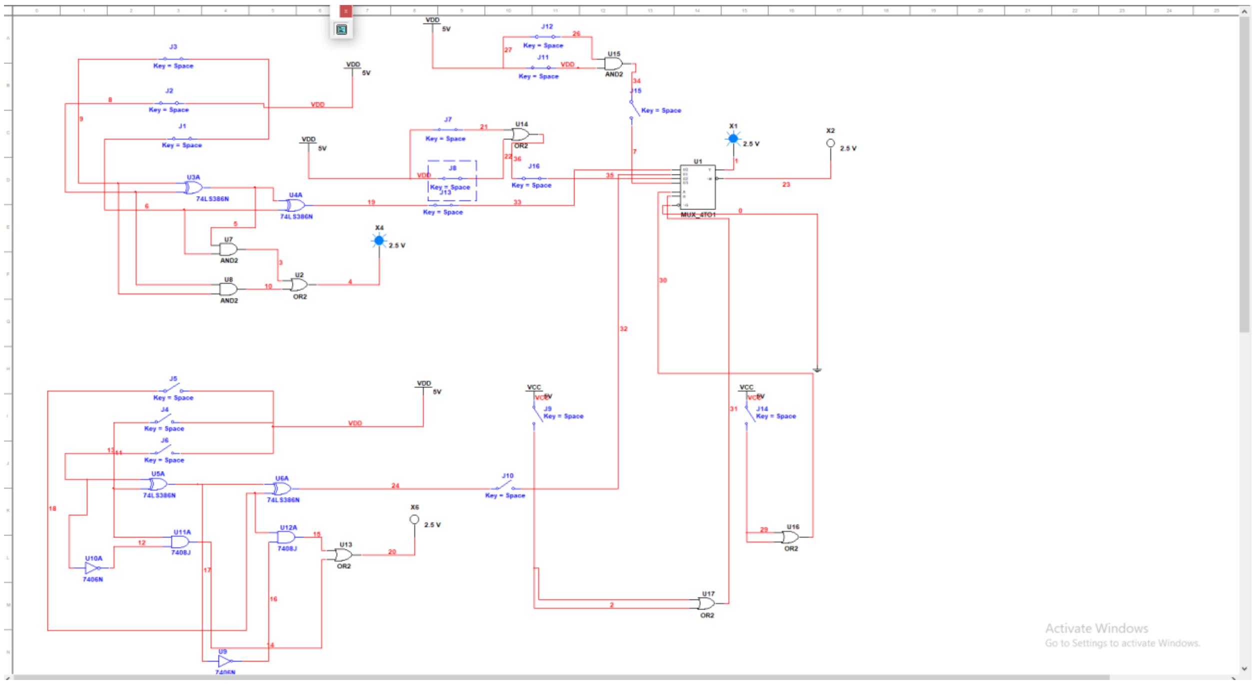


Figure 10:Fully Combined Circuit

As seen in the above circuit, the value will change based on what the selector selects, and what will be the output value from the process. It gives the user the ability to select from one of the 4 functions to get a desired value, all switch controlled, to ensure that the user has total control over what is wanted and desired. The only two logic circuit components without a part that are inside of the diagram will be the "OR" and "AND" operations, this is due to it being a single gate and with all info presented will be a given to calculate and determine output.

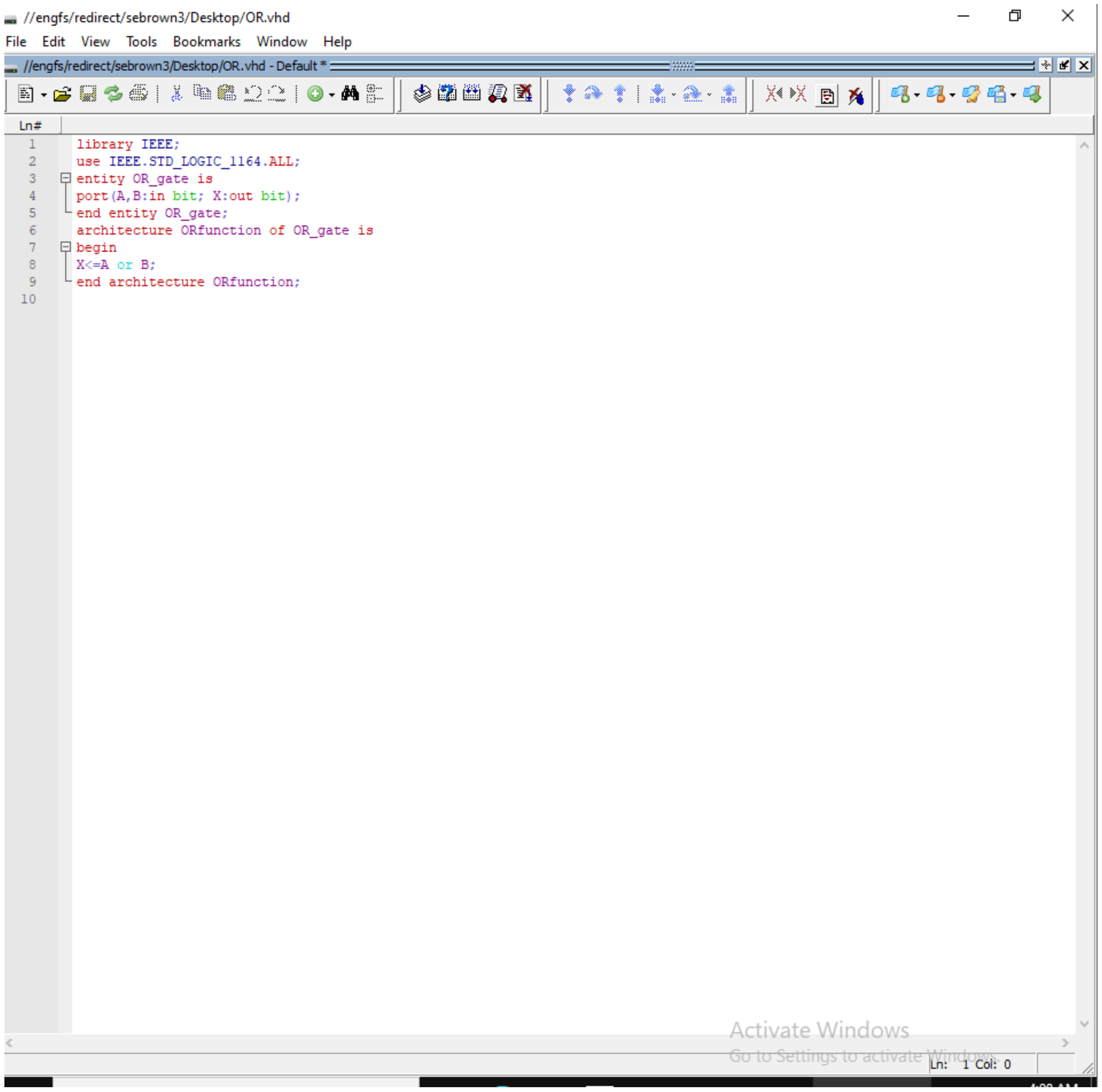
Conclusion

In conclusion the findings represented through out the project match up to those of other sources, whether from PowerPoints published or blogs online. I have been trying to put everything in VHDL together but to no luck, I constantly got error and tried to find more notes on it, but there was not a lot of info online to use. The multiplex calculator provides a simple yet convenient way for user input through changing of methods in which a variable can be achieved. As shown in this lab and in my presentation the varying between each device in the project is justifiable and correct, both equation and logic wise.

In terms of modeling and replicating it was successful and plan to input that into presentation to submit with this file. The applications and uses of this calculator, because of its basic functionality and usefulness makes it a highly valuable item with immeasurable uses.

Appendix

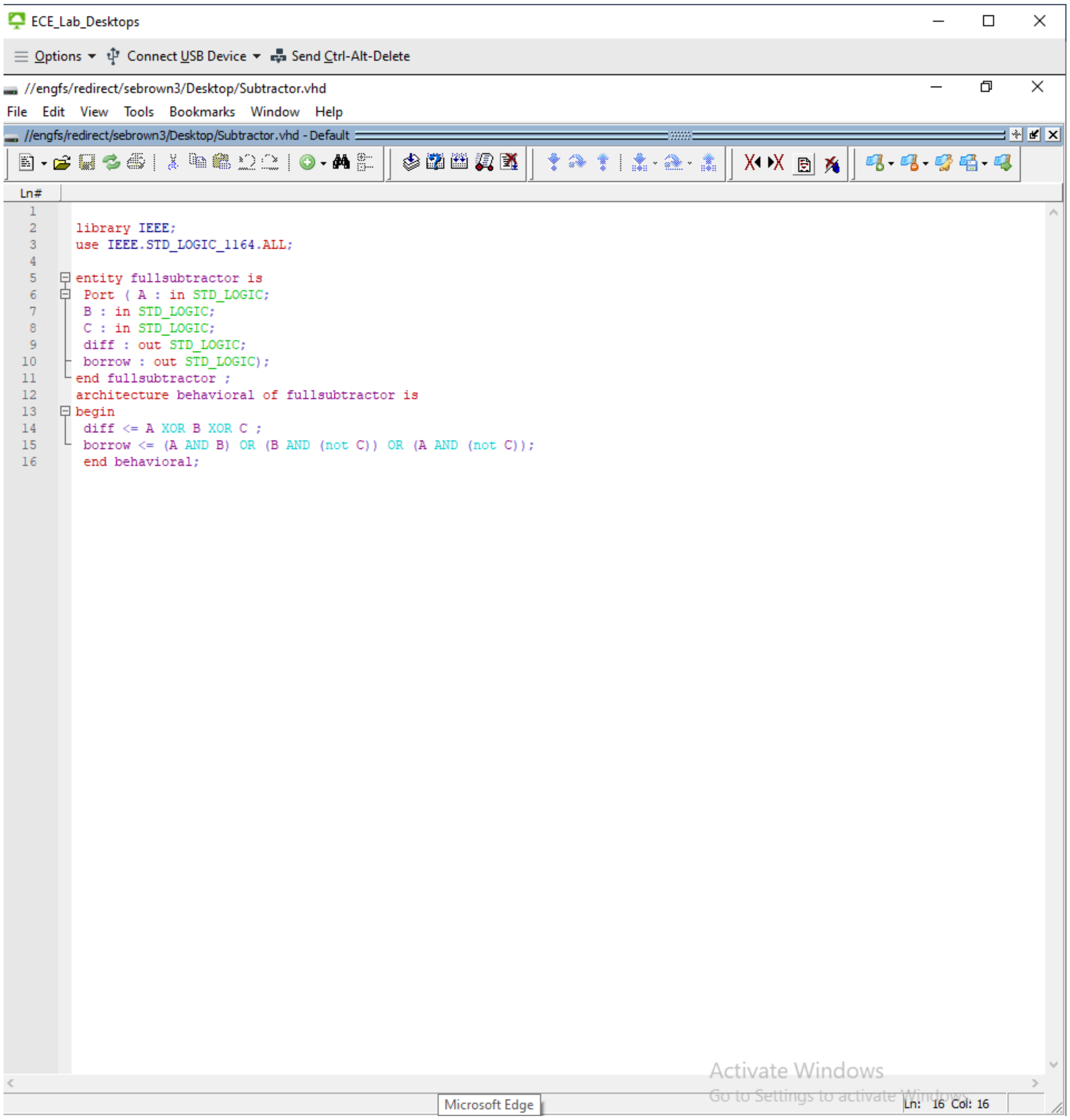
Images


//engfs/redirect/sebrown3/Desktop/OR.vhd
File Edit View Tools Bookmarks Window Help
//engfs/redirect/sebrown3/Desktop/OR.vhd - Default *
Ln#

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity OR_gate is
port(A,B:in bit; X:out bit);
end entity OR_gate;
architecture ORfunction of OR_gate is
begin
X<=A or B;
end architecture ORfunction;
```

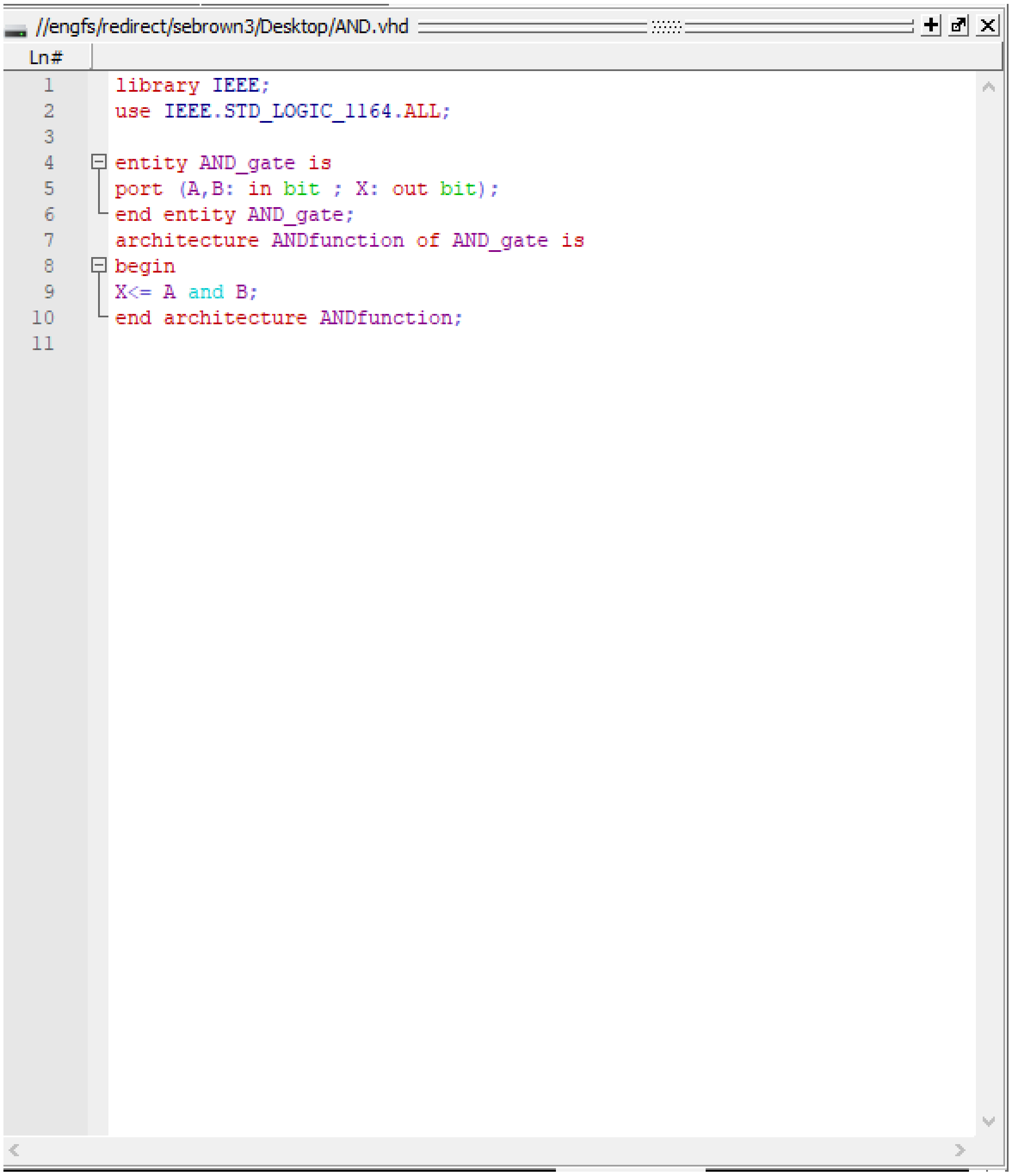
Activate Windows
Go to Settings to activate Windows.
Ln: 1 Col: 0

Figure 11:ORgate VHDL code



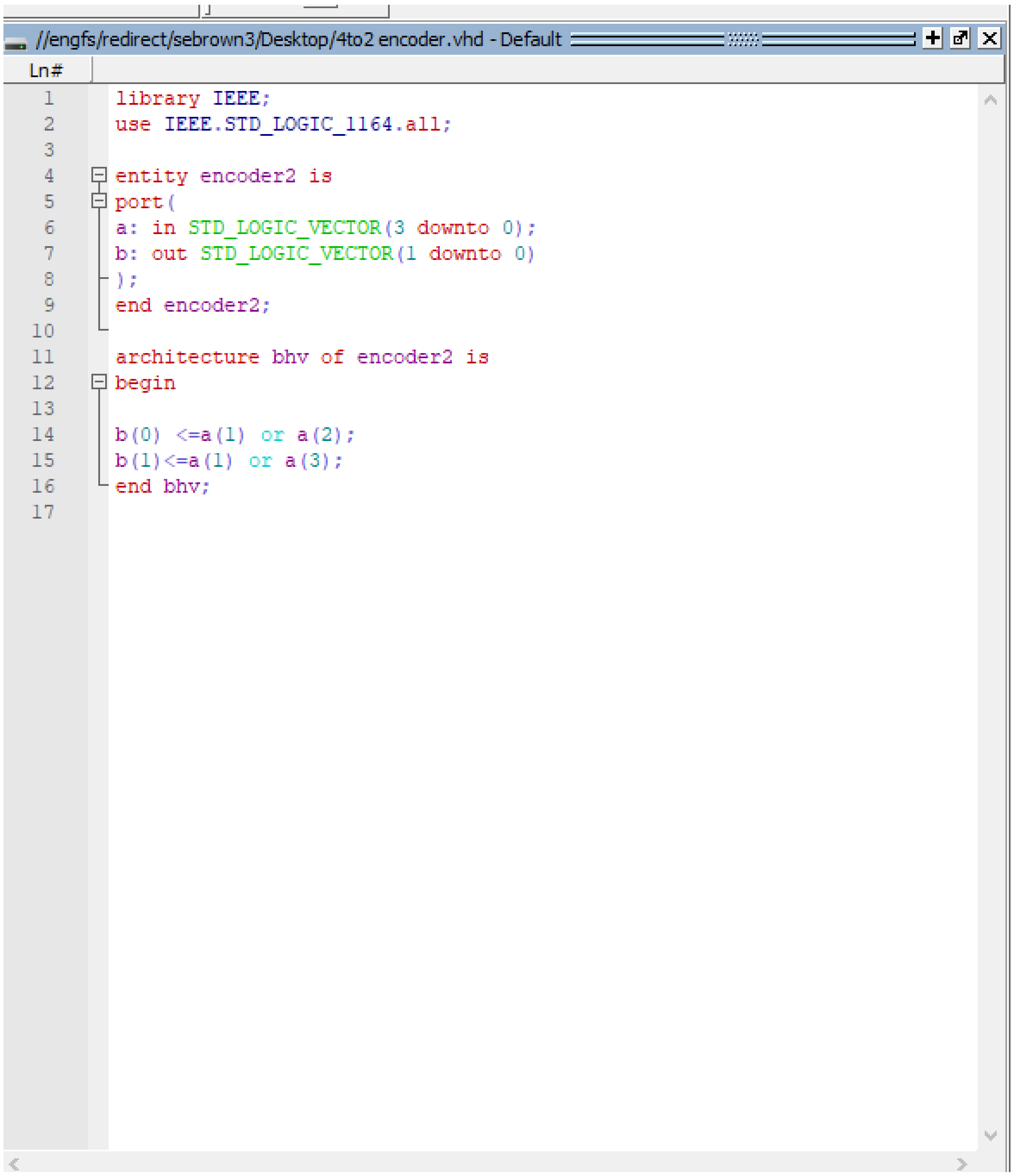
```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fullsubtractor is
 Port ( A : in STD_LOGIC;
 B : in STD_LOGIC;
 C : in STD_LOGIC;
 diff : out STD_LOGIC;
 borrow : out STD_LOGIC);
end fullsubtractor ;
architecture behavioral of fullsubtractor is
begin
 diff <= A XOR B XOR C ;
 borrow <= (A AND B) OR (B AND (not C)) OR (A AND (not C));
 end behavioral;
```

Figure 12:Subtractor VHDL Code


//engfs/redirect/sebrown3/Desktop/AND.vhd

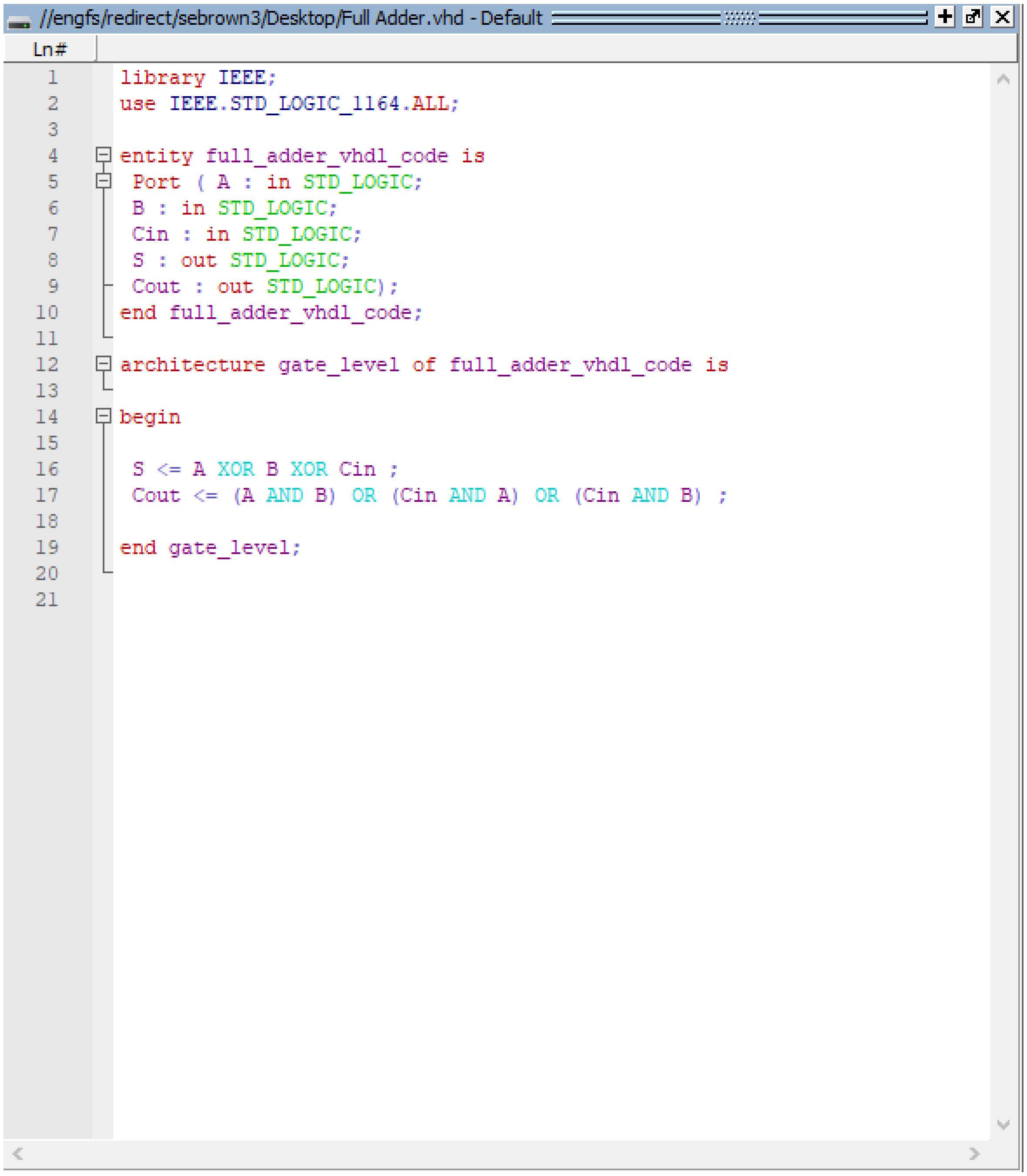
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity AND_gate is
port (A,B: in bit ; X: out bit);
end entity AND_gate;
architecture ANDfunction of AND_gate is
begin
X<= A and B;
end architecture ANDfunction;

```

Figure 13:AND gate VHDL code


//engfs/redirect/sebrown3/Desktop/4to2 encoder.vhd - Default

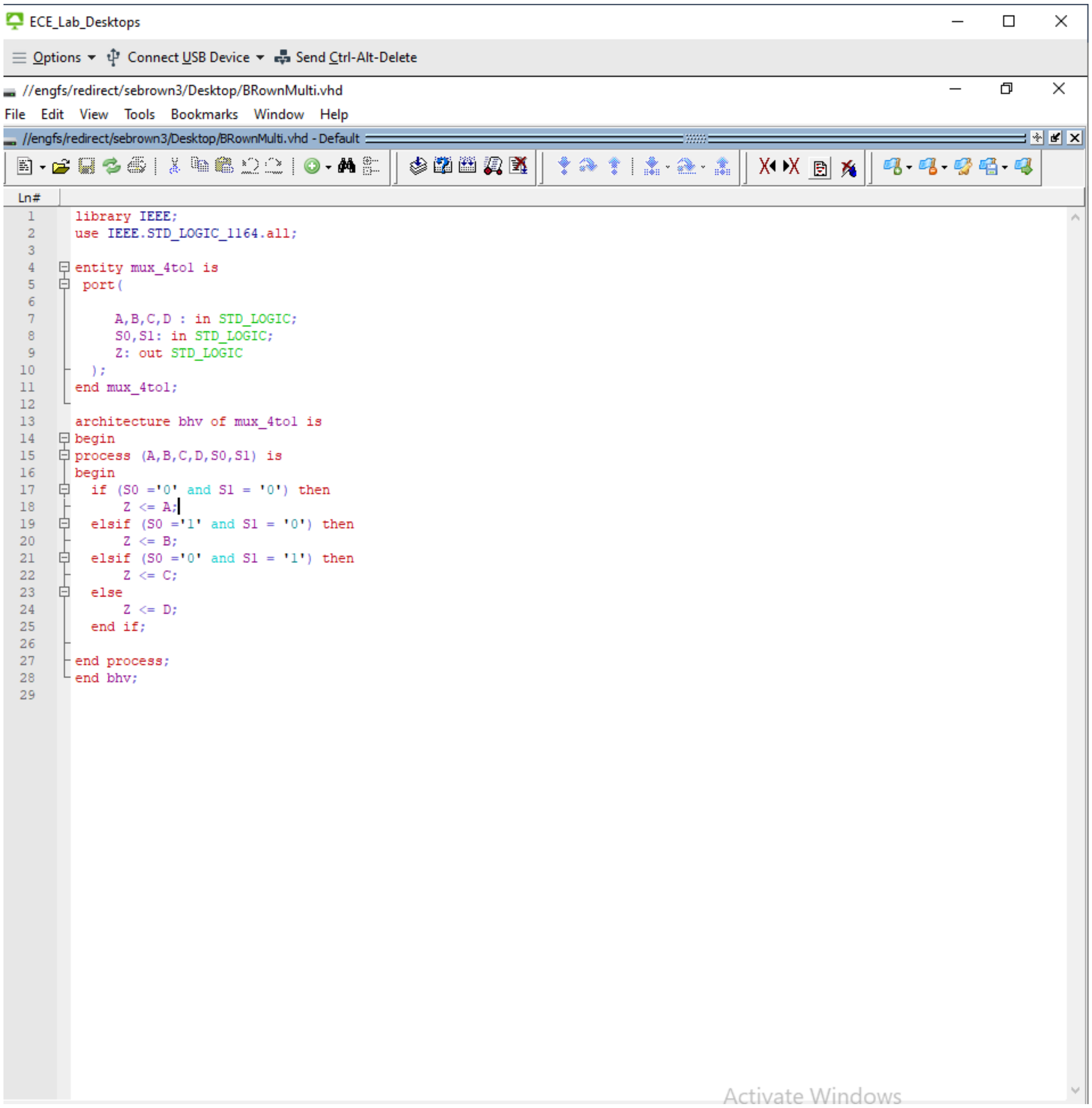
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity encoder2 is
port(
a: in STD_LOGIC_VECTOR(3 downto 0);
b: out STD_LOGIC_VECTOR(1 downto 0)
);
end encoder2;

architecture bhv of encoder2 is
begin

b(0) <=a(1) or a(2);
b(1)<=a(1) or a(3);
end bhv;

```

Figure 14: Encoder 4:2 VHDL code



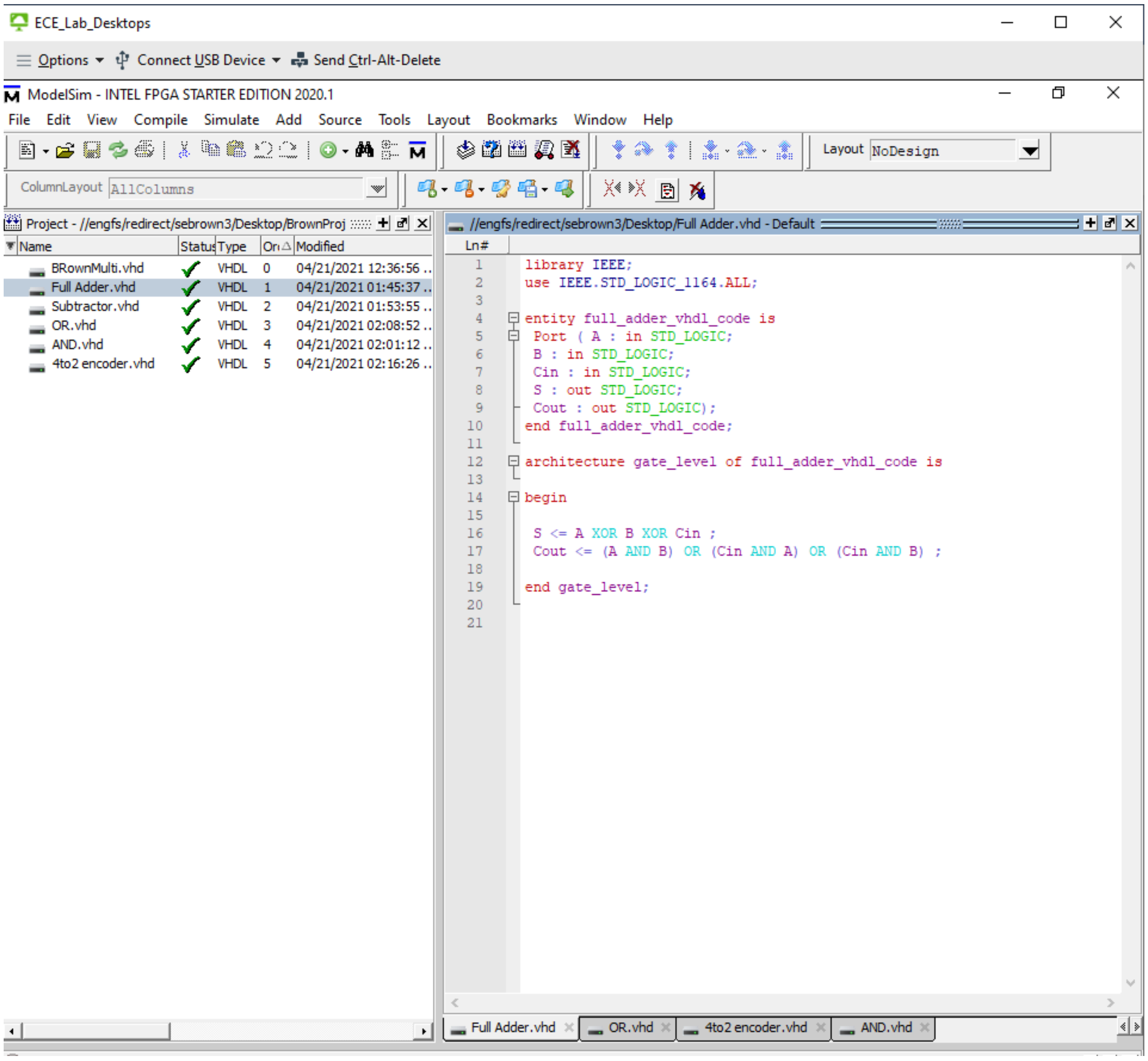
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity full_adder_vhdl_code is
 Port ( A : in STD_LOGIC;
 B : in STD_LOGIC;
 Cin : in STD_LOGIC;
 S : out STD_LOGIC;
 Cout : out STD_LOGIC);
end full_adder_vhdl_code;

architecture gate_level of full_adder_vhdl_code is

begin

 S <= A XOR B XOR Cin ;
 Cout <= (A AND B) OR (Cin AND A) OR (Cin AND B) ;

end gate_level;
```

Figure 15:Full Adder VHDL code


ECE_Lab_Desktops
Options
Connect USB Device
Send Ctrl-Alt-Delete
//engfs/redirect/sebrown3/Desktop/BRownMulti.vhd
File Edit View Tools Bookmarks Window Help
//engfs/redirect/sebrown3/Desktop/BRownMulti.vhd - Default
Ln#
Activate Windows

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux_4tol is
 port(

     A,B,C,D : in STD_LOGIC;
     S0,S1: in STD_LOGIC;
     Z: out STD_LOGIC
   );
end mux_4tol;

architecture bhv of mux_4tol is
begin
process (A,B,C,D,S0,S1) is
begin
  if (S0 ='0' and S1 = '0') then
      Z <= A;
  elsif (S0 ='1' and S1 = '0') then
      Z <= B;
  elsif (S0 ='0' and S1 = '1') then
      Z <= C;
  else
      Z <= D;
  end if;

end process;
end bhv;
```

Figure 16:Multiplex VHDL code


ECE_Lab_Desktops
Options Connect USB Device Send Ctrl-Alt-Delete
ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Source Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Project - //engfs/redirect/sebrown3/Desktop/BrownProj
//engfs/redirect/sebrown3/Desktop/Full Adder.vhd - Default
Full Adder.vhd OR.vhd 4to2 encoder.vhd AND.vhd

Name	Status	Type	Order	Modified
BRownMulti.vhd	✓	VHDL	0	04/21/2021 12:36:56 ..
Full Adder.vhd	✓	VHDL	1	04/21/2021 01:45:37 ..
Subtractor.vhd	✓	VHDL	2	04/21/2021 01:53:55 ..
OR.vhd	✓	VHDL	3	04/21/2021 02:08:52 ..
AND.vhd	✓	VHDL	4	04/21/2021 02:01:12 ..
4to2 encoder.vhd	✓	VHDL	5	04/21/2021 02:16:26 ..

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity full_adder_vhdl_code is
 Port ( A : in STD_LOGIC;
 B : in STD_LOGIC;
 Cin : in STD_LOGIC;
 S : out STD_LOGIC;
 Cout : out STD_LOGIC);
end full_adder_vhdl_code;

architecture gate_level of full_adder_vhdl_code is

begin

 S <= A XOR B XOR Cin ;
 Cout <= (A AND B) OR (Cin AND A) OR (Cin AND B) ;

end gate_level;
```

Figure 17: Proof of Compilation Success