Contribution chart

Brian Marte – Individual contributor.

Final Project

Calculator

COURSE-CST-223001.2021

Instructor: Jannatun Naher

Due Date: 5/3/2021

By: Brian Marte

OBJECTIVES

Design a calculator which takes two inputs with 4-bit width and it can do 4 arithmetic or logic operations. The user has a 4-input button (which acts as 1 bit input) and based on that; the user can decide which operation needs to be performed. The calculator will provide the final one output with 4-bits for that specific operation, which the user has chosen.

Also, to the above make sure to include the following

- Level 0, Level 1, and/or Level 2 block diagram of the calculator
- The VHDL code for each component and the top-level code
- Simulate and design

INTRODUCTION

We will build a calculator using a top-down approach using VHDL code in the process to simulate the proposed hardware.

Level 0 of the calculator, shows the inputs from outside influence and the single 4-bit output. We can see that the input A and input B both are 4-bit as described earlier in the text.

Level 0 Diagram:

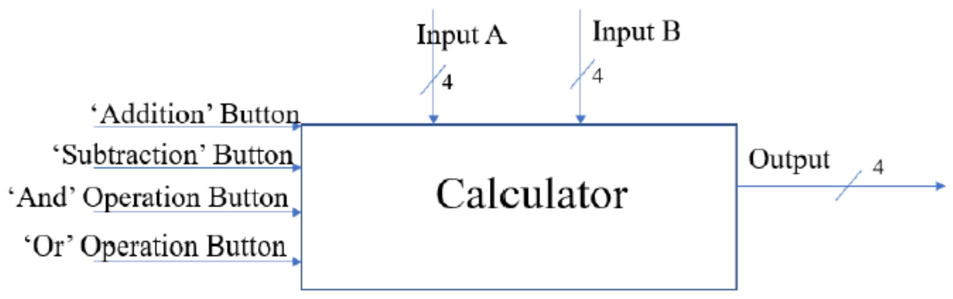


Figure 1. Level 0 of Calculator

Level 1 of the calculator is the collection of all the individual operations within the calculator. This is best described as the connections (signals) between the individual operations

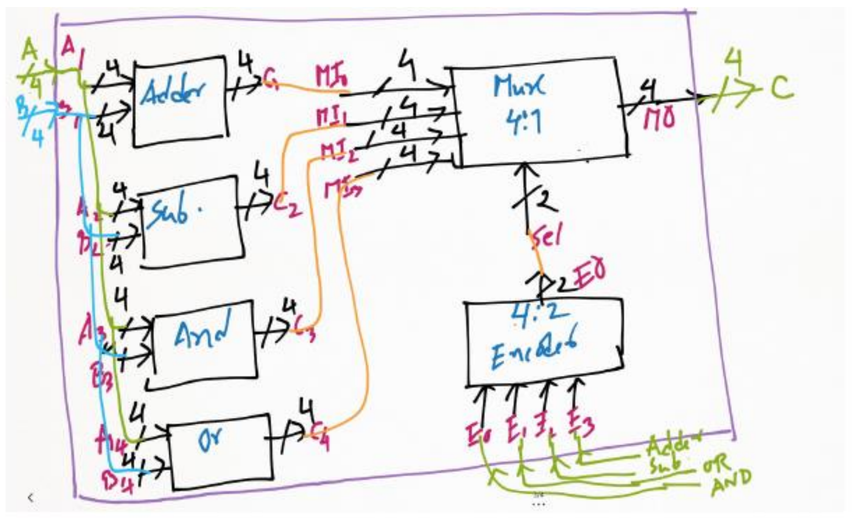


Figure 2. Level 1 of the calculator

Level 2 of the calculator are the individual operations. This will be as if the operations were standalone, and then added into a package. These are the 'Adder', 'Subtractor', 'And' operation, 'Or' operation, the 4:2 encoder, and the 4:1 Mux.

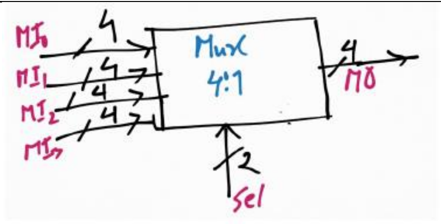
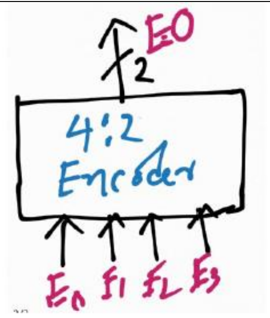
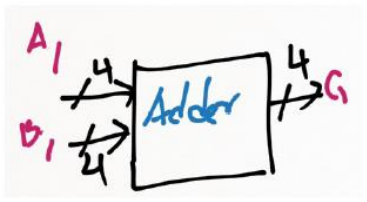
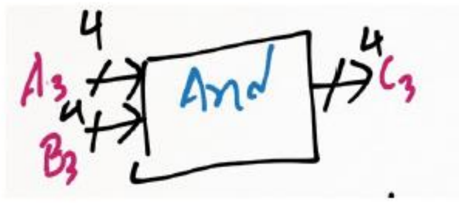
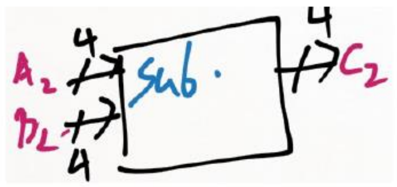
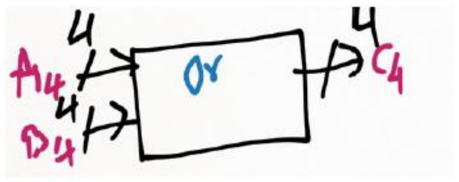
	
	
	

Figure 3. Collection of Level 2 operations

PROCEDURES

1) Draw/Write out the design of the calculator with desired functions.
2) Use the written-out design to write VHDL code in ModelSim/Quartus software
3) Use that code to simulate and analyze behavior of the design.
4) Show results of simulation and discuss.

RESULTS & ANALYSIS

CONCLUSION & DISCUSSION

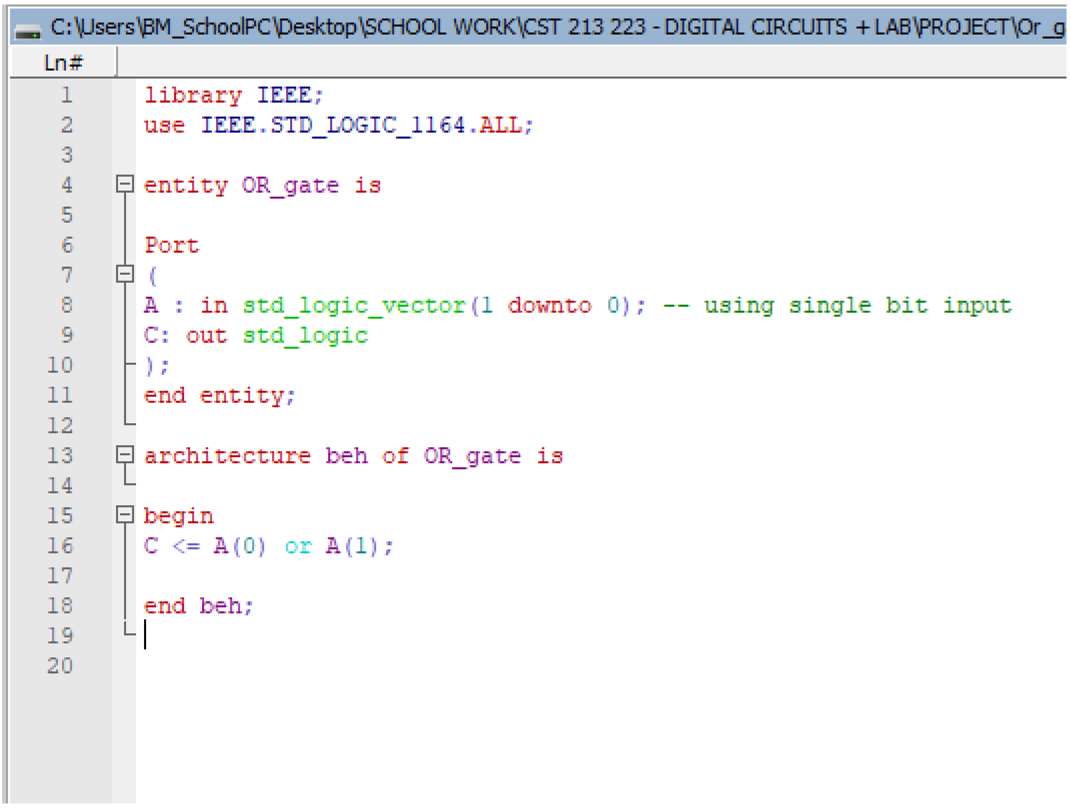
Using the top-down method of design is the most preferred in hardware design. Due to its adaptability and ease to making changes as needed dynamically. Using this approach is has its benefits of producing the desired designing faster.

The operation should have the two inputs and depending on the selection of add, sub, and/or logic operation. Then the circuit should perform those operations depending on what was selected.

REFERENCES

Floyd, Thomas L., Digital Fundamentals 11th ed.

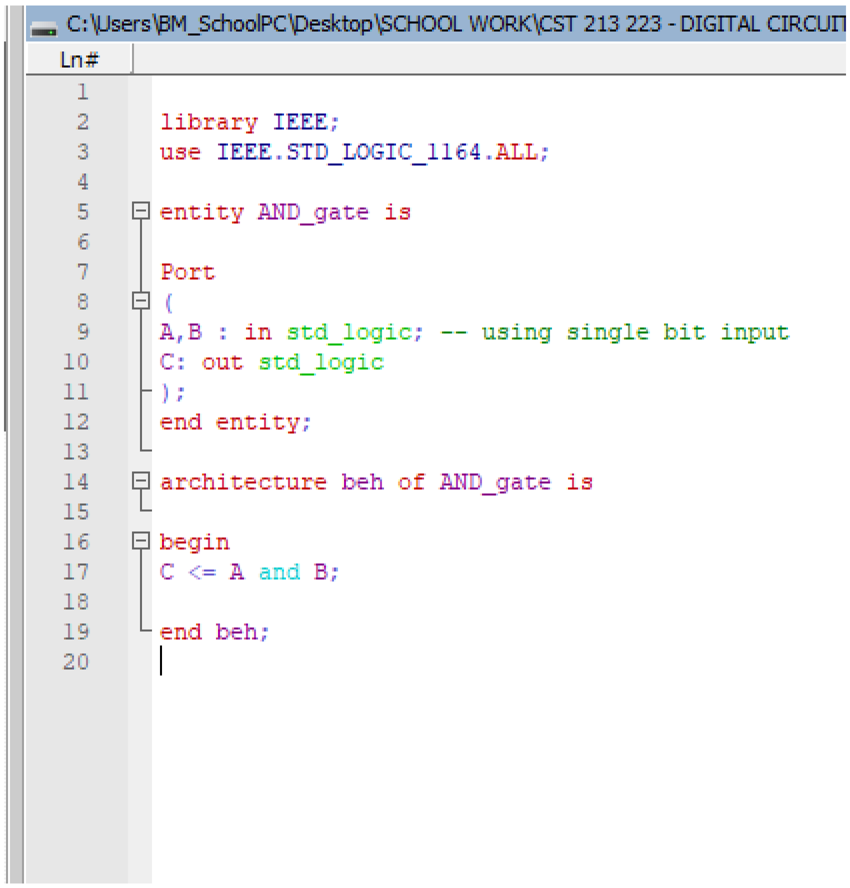
APPENDIX



C:\Users\BM_SchoolPC\Desktop\SCHOOL WORK\CST 213 223 - DIGITAL CIRCUITS + LAB\PROJECT\Or_g

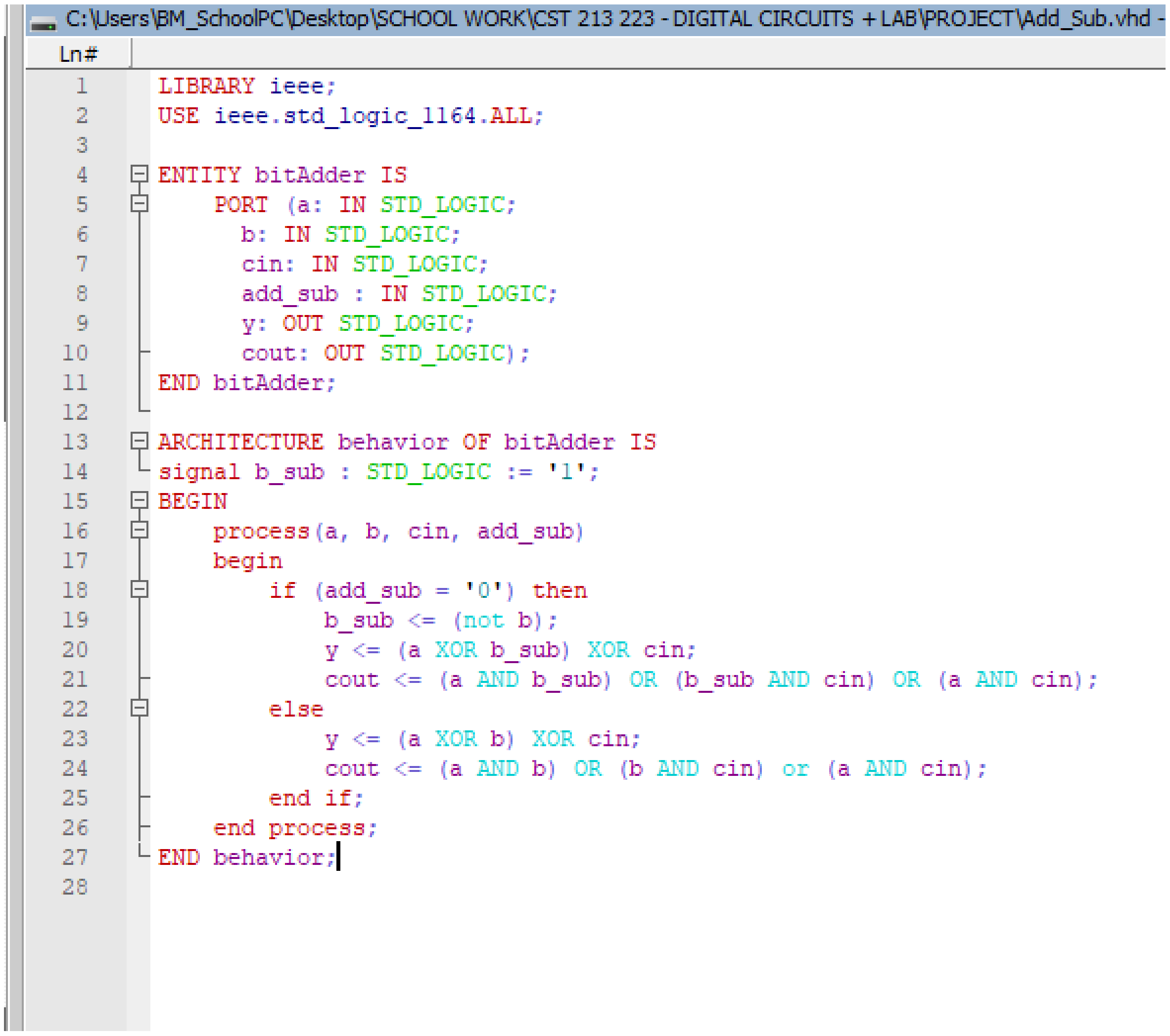
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity OR_gate is

Port
(
A : in std_logic_vector(1 downto 0); -- using single bit input
C: out std_logic
);
end entity;

architecture beh of OR_gate is

begin
C <= A(0) or A(1);

end beh;
```

Figure 4. Or Gate VHDL Code


C:\Users\BM_SchoolPC\Desktop\SCHOOL WORK\CST 213 223 - DIGITAL CIRCUIT
Ln#

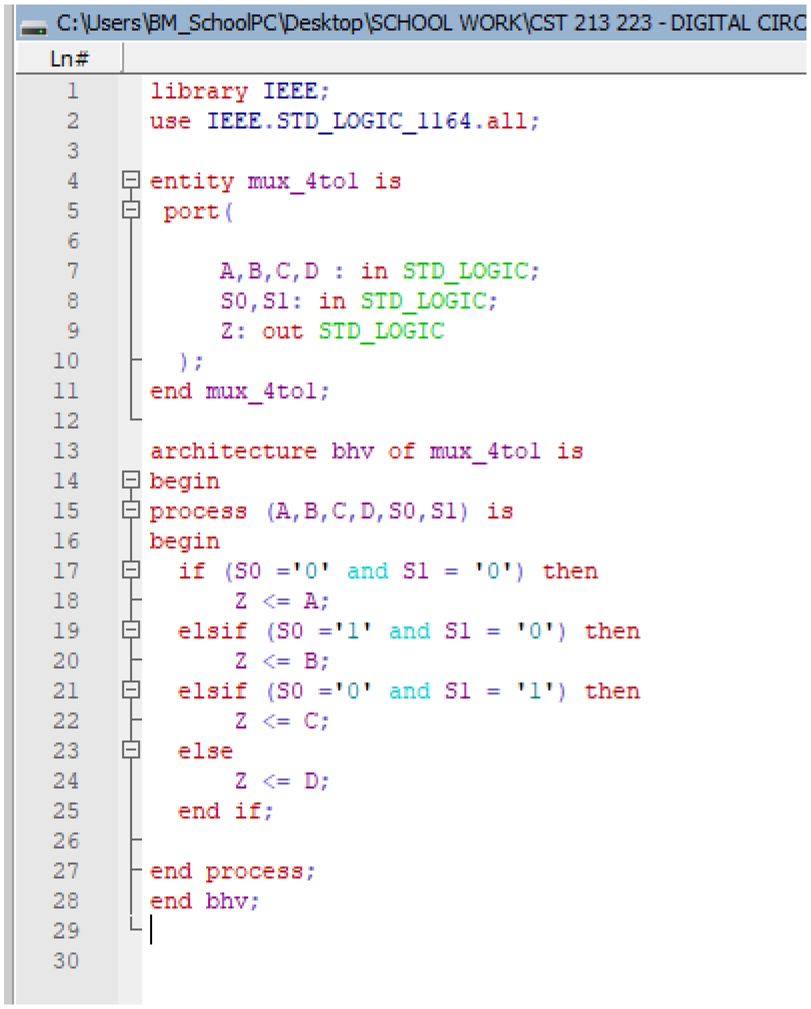
```vhdl

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity AND_gate is

Port
(
A,B : in std_logic; -- using single bit input
C: out std_logic
);
end entity;

architecture beh of AND_gate is

begin
C <= A and B;

end beh;
```

Figure 5. AND Gate VHDL code


C:\Users\BM_SchoolPC\Desktop\SCHOOL WORK\CST 213 223 - DIGITAL CIRCUITS + LAB\PROJECT\Add_Sub.vhd -

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY bitAdder IS
    PORT (a: IN STD_LOGIC;
      b: IN STD_LOGIC;
      cin: IN STD_LOGIC;
      add_sub : IN STD_LOGIC;
      y: OUT STD_LOGIC;
      cout: OUT STD_LOGIC);
END bitAdder;

ARCHITECTURE behavior OF bitAdder IS
signal b_sub : STD_LOGIC := '1';
BEGIN
    process(a, b, cin, add_sub)
    begin
        if (add_sub = '0') then
            b_sub <= (not b);
            y <= (a XOR b_sub) XOR cin;
            cout <= (a AND b_sub) OR (b_sub AND cin) OR (a AND cin);
        else
            y <= (a XOR b) XOR cin;
            cout <= (a AND b) OR (b AND cin) or (a AND cin);
        end if;
    end process;
END behavior;
```

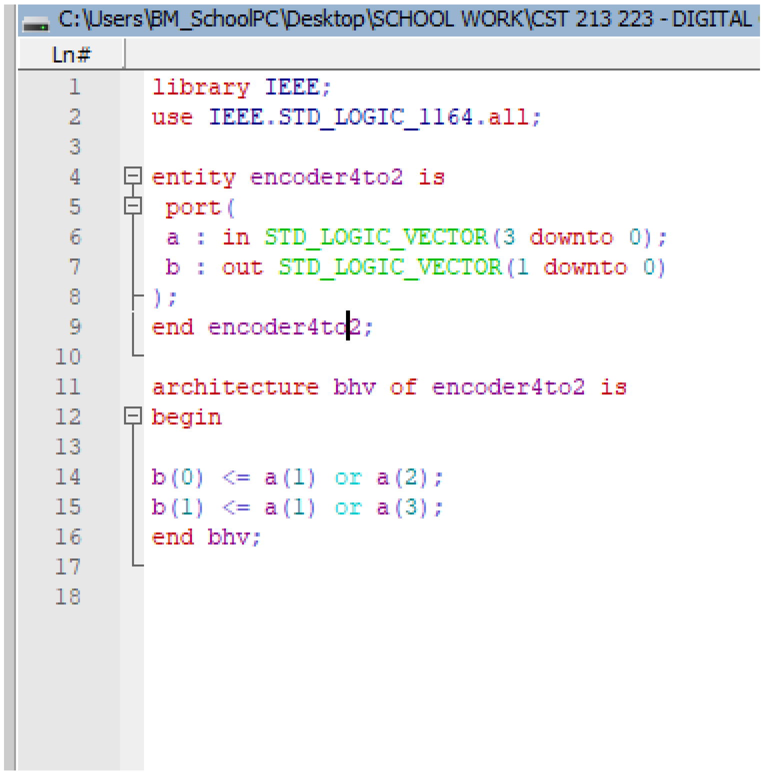
Figure 6. Add-Sub VHDL code



C:\Users\BM_SchoolPC\Desktop\SCHOOL WORK\CST 213 223 - DIGITAL CIRC

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

entity mux_4to1 is
 port(

     A,B,C,D : in STD_LOGIC;
     S0,S1: in STD_LOGIC;
     Z: out STD_LOGIC
  );
end mux_4to1;

architecture bhv of mux_4to1 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 7. 4:1 Mux VHDL Code



C:\Users\BM_SchoolPC\Desktop\SCHOOL WORK\CST 213 223 - DIGITAL

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

entity encoder4to2 is
 port(
 a : in STD_LOGIC_VECTOR(3 downto 0);
 b : out STD_LOGIC_VECTOR(1 downto 0)
);
end encoder4to2;

architecture bhv of encoder4to2 is
begin

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

Figure 8. 4:2 Encoder VHDL code