Mark Lee ECE 357 November 11, 2019

Lab 3: Arithmetic Logic Unit (ALU) Design

This lab implements an arithmetic logic unit (ALU) on the BASYS 3 Artix-7 Board through VHDL code written in Vivado. The ALU operates on two 8-bit signed integer inputs stored in registers, receiving inputs and displaying outputs on the BASYS 3 board. Task 1 implements the various arithmetic operations available on the ALU and the load/read operations for register contents. Task 2 produces the same product as Task 1 but with a structural VHDL design for addition/subtraction logic.

TA Date: 11-11-19

TA Signature: 100 %

Introduction

Computer and electrical engineers are expected to be adept designers of hardware components present in modern day electronic systems. One of the most fundamental components is the arithmetic logic unit (ALU), a combinational digital circuit that performs arithmetic and binary operations on binary integers. An ALU takes two binary operands as input, as well as a special code (called an "op code") that specifies which operation is to be performed on the operands. The ALU then performs the operation and outputs the result. ALUs are vital to the function of contemporary computers as they provide essential computational capabilities, and as such are core elements for CPUs and other processing entities.

This lab exercise requires students to implement their own ALU on the FPGA. This ALU takes two 8-bit signed binary integers as input data to be operated on, and also takes various op codes to execute functions like addition, subtraction, multiplication, and boolean and bitwise operations. All actions are clocked by a button. The 8-bit binary inputs are stored in two explicitly-declared registers, and the result of the operation is stored in one of the registers as well. The BASYS 3 board can display register contents on the 7-segment displays and the sign bits of register contents on LEDs.

The lab is split into two separate parts. Task 1 implements all of the above functions in VHDL code by using VHDL operators and type cast functions to perform the op codes behaviorally. The result is a full-functional ALU system. Task 2 does the same thing as Task 1, except addition and subtraction are implemented as a structural 8-bit carry-ripple adder.

By accomplishing these tasks, electrical and computer engineering students will obtain a higher degree of competency and versatility in their hardware design abilities. Knowing how to design an ALU will grant students a deeper comprehension of processor dynamics and invaluable experience working with digital logic applications.

Procedure

To implement the ALU on the BASYS3 board I began considering my design choices. I examined the lab specifications and observed that I needed 12 arithmetic/bitwise operations and 5 load/read operations, for a total of 17 op codes. To accommodate this amount of op codes I

decided to use a 5-bit op code input using the left 5 switches (SW15-SW11) on the BASYS3 board. I encoded the op codes to correspond to the order in which the operations are presented in the lab instructions. So since addition is first in the lab write-up it is op code 00001, subtraction is 00010, multiplication is 00011… and so forth until "Read Register A0, B0" which is 10001.

Concerning my two 8-bit signed integers A and B, I used the right 8 switches (SW7-SW0) on the FPGA board to record input values. I then created two registers called A0 and B0 to store the data in memory. To create these registers I made a D flip flop entity in VHDL and created registers from it using a generate statement. Thus I ended up with 16 flip flops in total.

All ALU operations are clocked with the middle button (pin U18) on the BASYS3 board. To account for miniscule oscillations of the button when pressed down that could affect my VHDL logic, I generated a debounced pulse that effectively ignored these oscillations. Additionally, it should be noted that the clocks for registers A0 and B0 are not the same as the button but are signals declared within my VHDL code. Whenever I want to update the output stored in a register I manually enable the clock in my code. This gives my design more control, as I can keep the memory components controlled by software, not hardware.

My main VHDL logic to implement the ALU uses a case statement on different op code values. Whenever the button is pressed, the ALU performs different operations based on the input of switches 15-11. If the op code ranges from 00001-01100 (1-12) then various arithmetic and bitwise operations are performed on the values in registers A0 and B0, and the result is stored in A0. To enhance user-system interfacing, the ALU implementation also offers several functions that enable the user to input desired values or read what values are currently stored in the registers. These correspond to opcodes 01101-10001 (13-17) which allow the user to load A0 and B0 with input from switches 7-0, and also allow the user to read the values in A0, B0, or both on the seven segment display.

My 7-segment display code is almost identical to that of other labs I've done. I choose which anodes to set depending on whether I'm reading from one register or both, and then I use a clock divider to alternate anodes at a fast enough rate that they appear all turned on

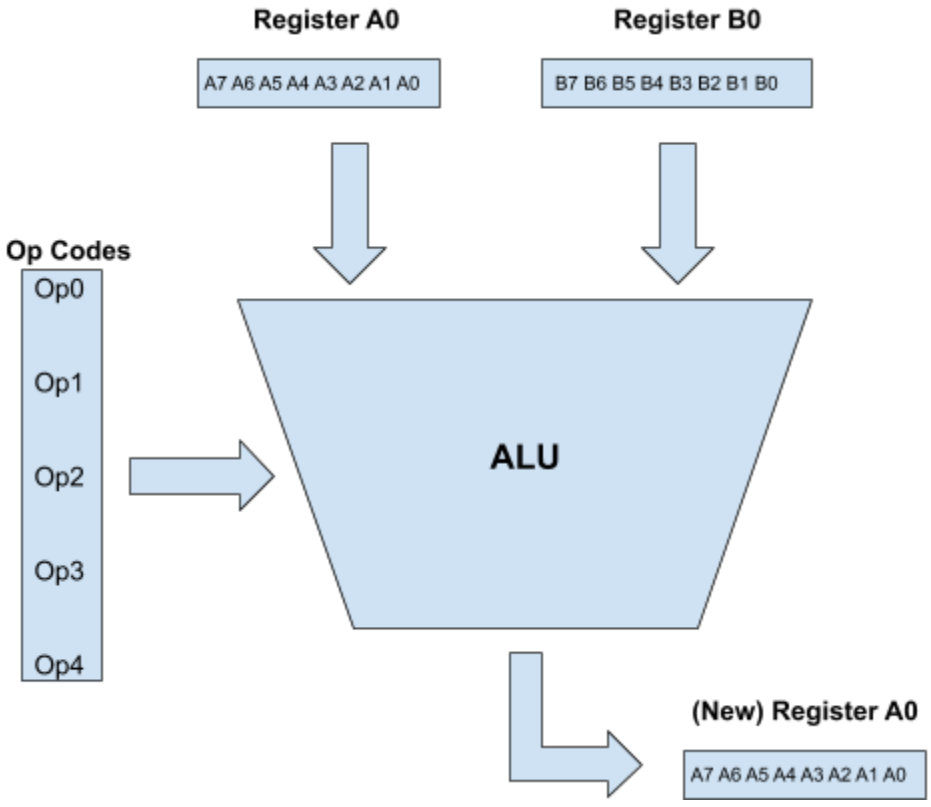
simultaneously to the human eye. Then I convert the register values to hexadecimal, and turn on the appropriate cathodes to display the desired register value.

Both inputs are signed, so when A or B is negative (i.e., the MSB is 1) then I turn LED 15 or LED 14 on, respectively. I also declare many of my signals and inputs as signed rather than std_logic_vector to easily perform arithmetic operations offered in VHDL.

In Task 1, I implemented all of the above requirements using operators available in VHDL, and using type cast functions to convert between std_logic_vector/signed/ unsigned/integer when necessary. In Task 2, my code is almost identical to Task 1, except I structurally implement addition and subtraction logic as an 8-bit ripple adder. This required me to create different components for AND, OR, XOR, NOT, and include all of these entities as components in a Full Adder entity. Finally, I added a generate statement in my main code to link 8 full adders together and successfully complete this design specification.

The below block diagram illustrates my design flow for the Task 1 and Task 2 ALU design:

ALU Block Diagram



The full op code key for my final ALU design is shown below.

Op Code					Operation
Op4	Op3	Op2	Op1	Op0	
0	0	0	0	1	Addition
0	0	0	1	0	Subtraction
0	0	0	1	1	Multiplication
0	0	1	0	0	Logical AND
0	0	1	0	1	Logical OR
0	0	1	1	0	Logical XOR
0	0	1	1	1	Logical NOT
0	1	0	0	0	Logical Shift Left
0	1	0	0	1	Logical Shift Right
0	1	0	1	0	Arithmetic Shift Right
0	1	0	1	1	Rotate Shift Left
0	1	1	0	0	Rotate Shift Right
0	1	1	0	1	Load Register A0
0	1	1	1	0	Load Register B0
0	1	1	1	1	Read Register A0
1	0	0	0	0	Read Register B0
1	0	0	0	1	Read Register A0, B0

Results

My results include the code attached at the end of the report in Appendices A, B, and C, and my successful ALU design demonstrated to the TAs during lab hours. As further evidence of my working design, I ran 4 simulations for my ALU implementation, each with different 8-bit signed integer binary inputs for variability. Featured prominently in the simulations are the register output contents after each operation is completed. Since A0 stores the result for every operation except multiplication, its value is the one changing the most, and is therefore the most important. One can also verify that the correct input values and opcodes for each operation are

indeed being input to the simulation by looking at the top of the figure, where I activate a different opcode every 5 nanoseconds to trigger either a load of A0/B0 or an operation on the register contents. This is also shown in my code for the simulation, which is shown in Appendix C. For space purposes I only include the first test case code, but the other three are identical except for the input values.

Figure 1 shows the ALU simulation for register input values of A0=00000111 (7) and B0=00000101 (5). The simulation begins with the lowest opcode (addition) on the left, and ends with the highest opcode (rotate shift right) on the right. Starting with addition, 7 + 5 = 12, and the output of register A is indeed 12 (00001100) on the very left. Moving on to subtraction, 7 - 5 = 2, which is illustrated by the next waveform to the right (00000010). This process continues for each operation, and each one can be easily validated through inspection. Notice that the only time the contents of register B0 change is after multiplication, where it holds the lower byte of the product (5 * 7 = 35, which in binary is 00100011, and the value stored in B0 according to the simulation is indeed this number).

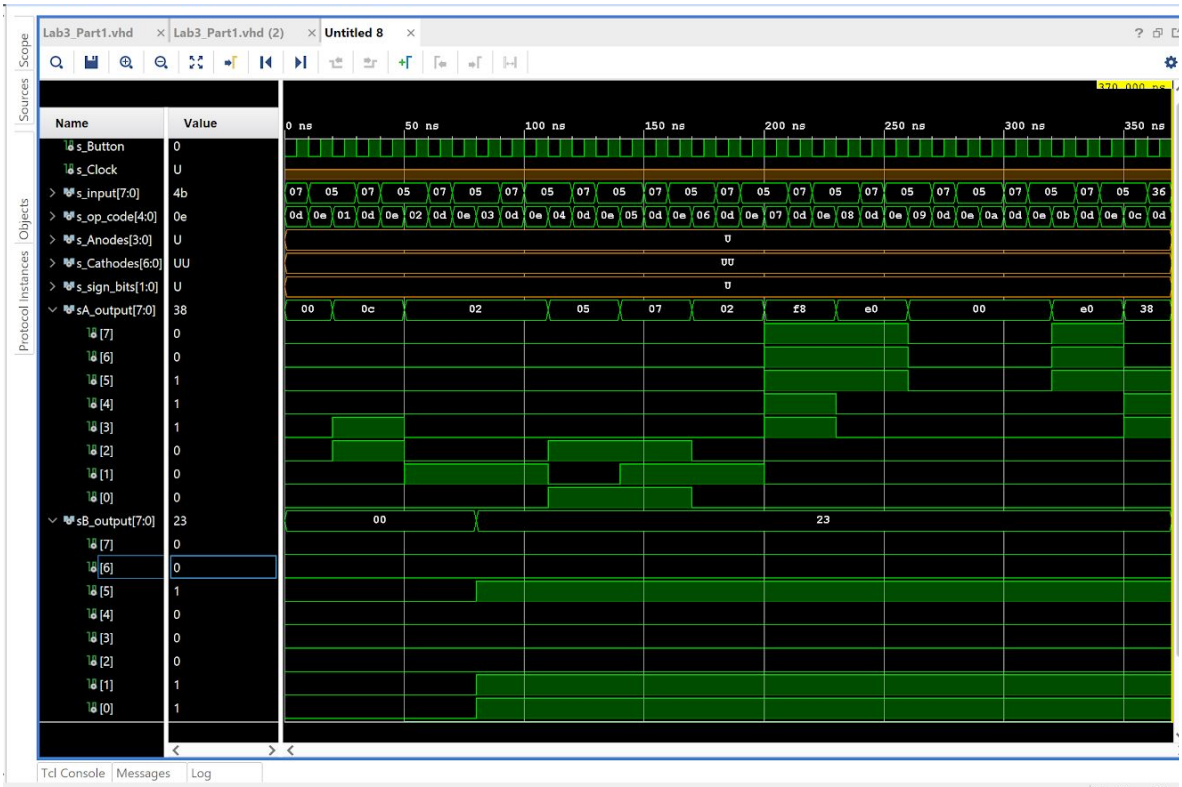


Figure 1. ALU simulation results for A0 = 7, B0 = 5.

Simulation 2 is displayed below in Figure 2. This time, the input values were A0=00110110 (54) and B0=01001011 (75). These are larger, more uncommon numbers than in the first simulation, which show that my ALU works on any two signed binary integer inputs. The input values and the result for each operation are shown below in the simulation waveform results. Picking a random operation to inspect, one could view the result of the logical XOR operation. 54 xor 75 = 125 = 01111101, which is indeed the value shown in the simulation (under the hex value "7d" near the middle). All other outputs are correct as well, but obviously due to timing constraints I will not go over all of them explicitly here.

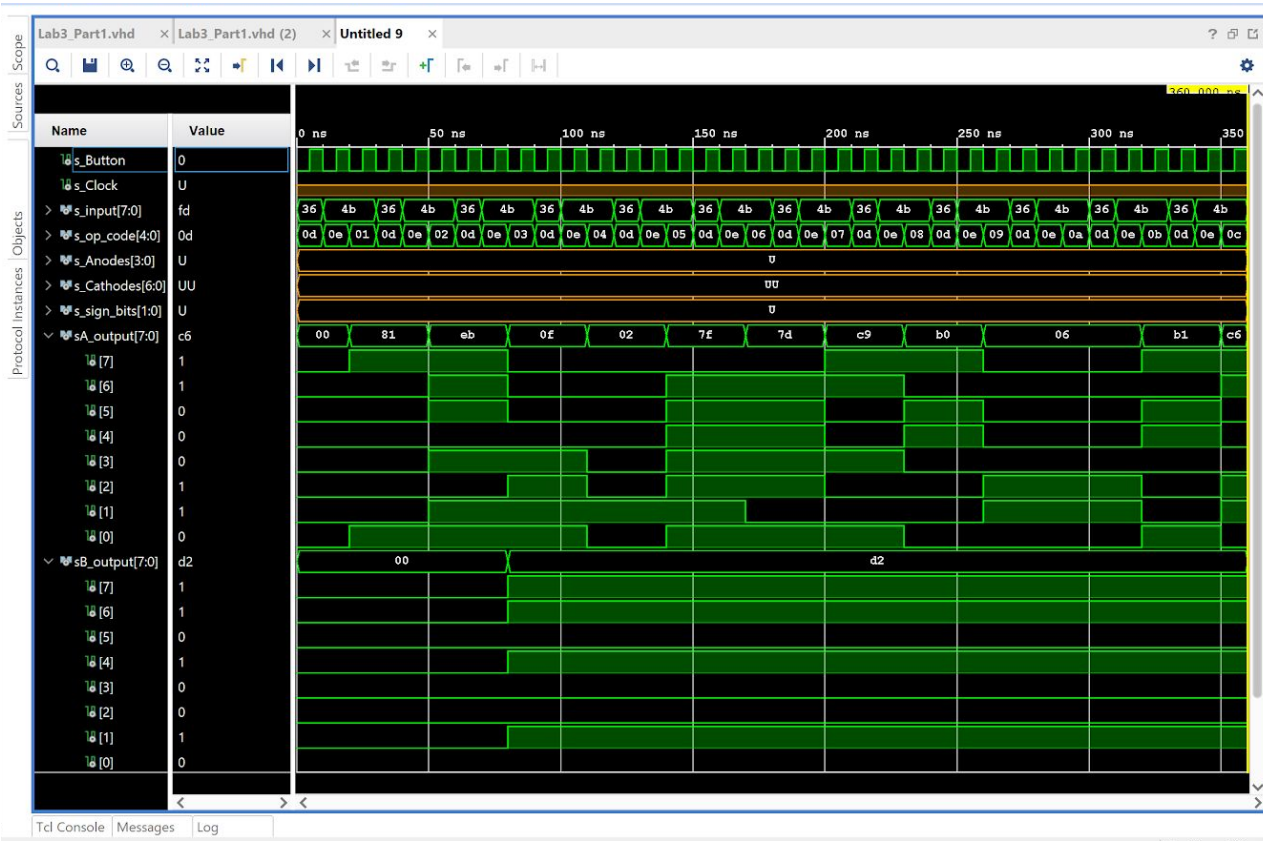


Figure 2. ALU simulation results for A0 = 54, B0 = 75.

Figure 3 shows yet another simulation waveform for the ALU. This time the inputs are A0 = 11111101 = -3, and B0 = 00001011 = 11. The most notable aspect of this simulation is that I now include a negative value, which shows my ALU can handle *all* signed numbers, not just the positive ones.

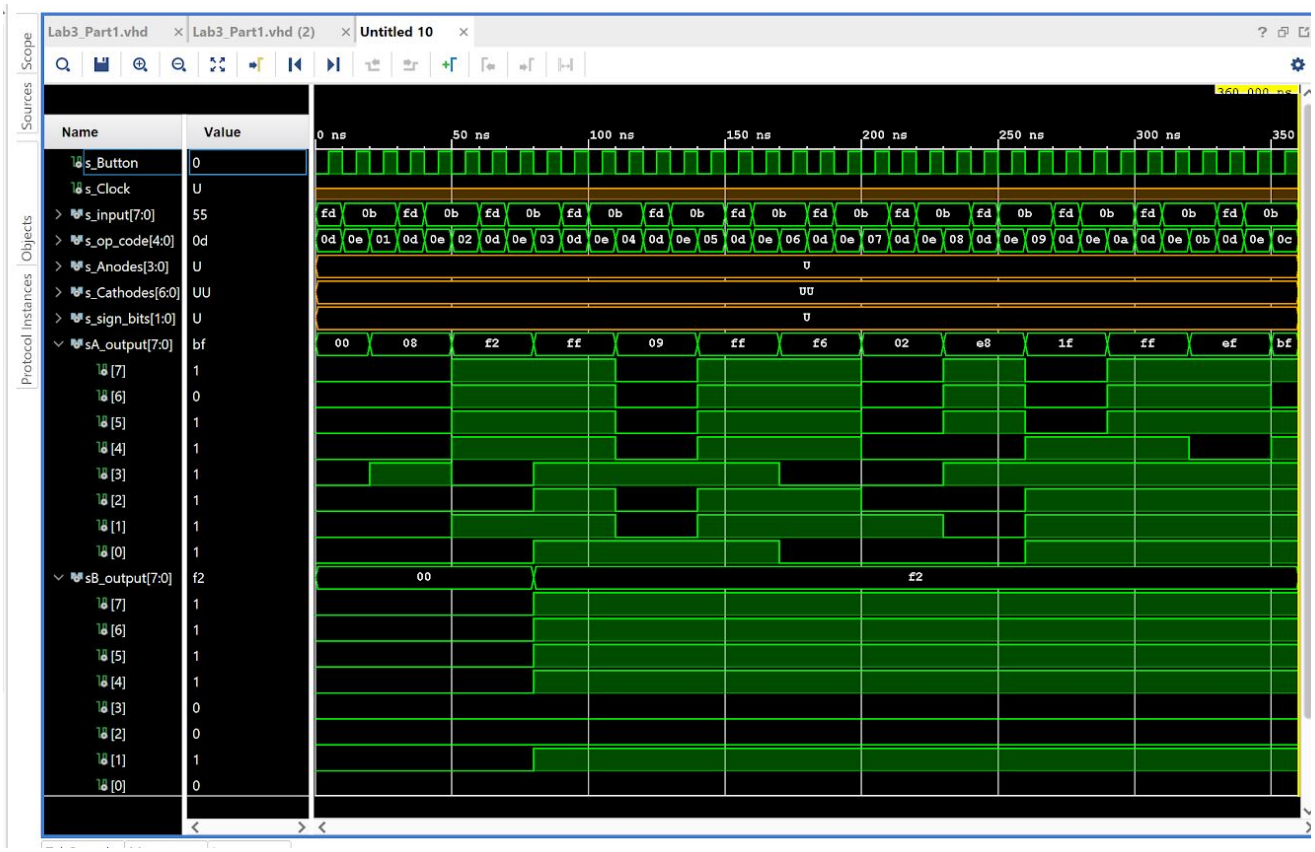


Figure 3. ALU simulation results for A0 = -3, B0 = 11.

Figure 4 below shows the simulation data for A0 = 01010101 = 85 and B0=00000001 = 1. The structure of the simulation waveforms is identical to the others above, just with different register inputs and therefore different results. These can all be verified by inspection.

Note that my simulations for Task 1 and Task 2 are the same, as I include the necessary logic for the structural addition and subtraction in my simulation code (Appendix C). The addition and subtraction logic simply come out the same, which proves the efficacy of my design for Task 2.

Beyond the simulations, another result I have are the schematics for the ALU design for Task 1 and Task 2. These are shown in Figures 5-8 below. Starting with my Task 1 schematic in Figure 5, the main component of my design is the large MUX towards the right of the figure that controls the opcode selection. This area is shown in more detail in Figure 6. Most of my opcodes are inputs to the MUX, and the output goes to register A0, showing that my design meets the

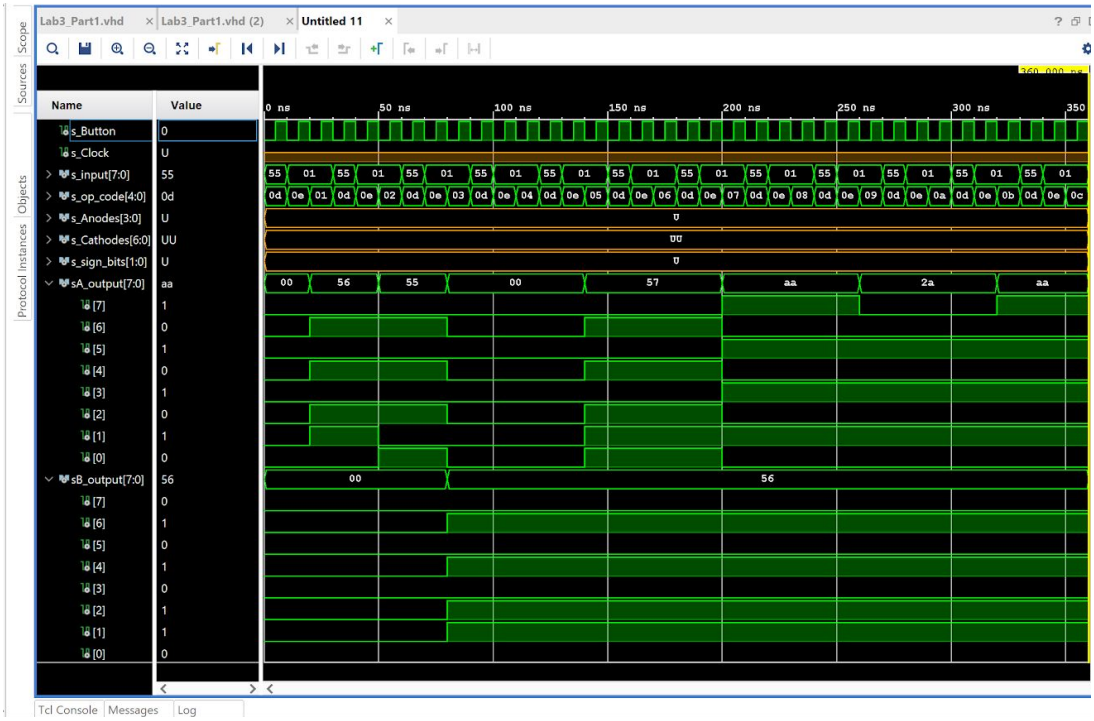


Figure 4. ALU simulation results for A0 = 85, B0 = 1.

specification that operation results be stored in register A0. Another vital aspect of this schematic are the registers I implemented explicitly, as stated in the lab instructions. This can be observed in Figure 5 and more clearly in Figure 6, where several instances of my "D_FlipFlop" entity form two registers A0 and B0. Figures 7 and 8 show the complete and enlarged views of my Task 2 schematic, respectively. The only significant difference is shown primarily in Figure 8, where my structural implementation of several full adders using a generate statement is represented by the several "Full_Adder" instances portrayed in the enlarged portion of the schematic. These perform addition and subtraction like Task 1, just in an explicitly-defined way (through Full Adders) than in a Vivado-generated behavioral manner. This structural method increases the amount of control the designer has over the hardware, and decreases overhead, so it's good practice to implement it in this way.

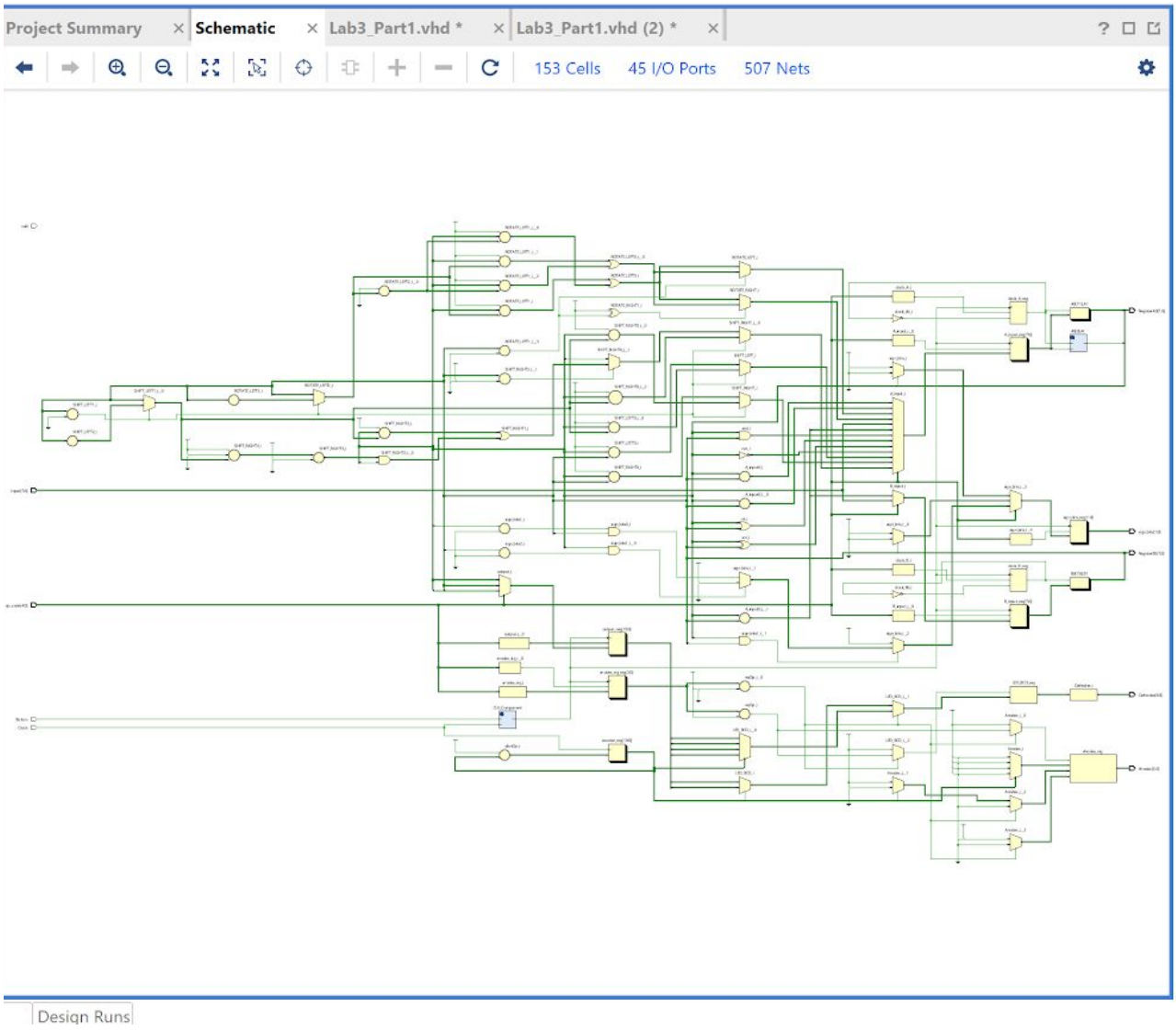


Figure 5. ALU Schematic for Task 1.

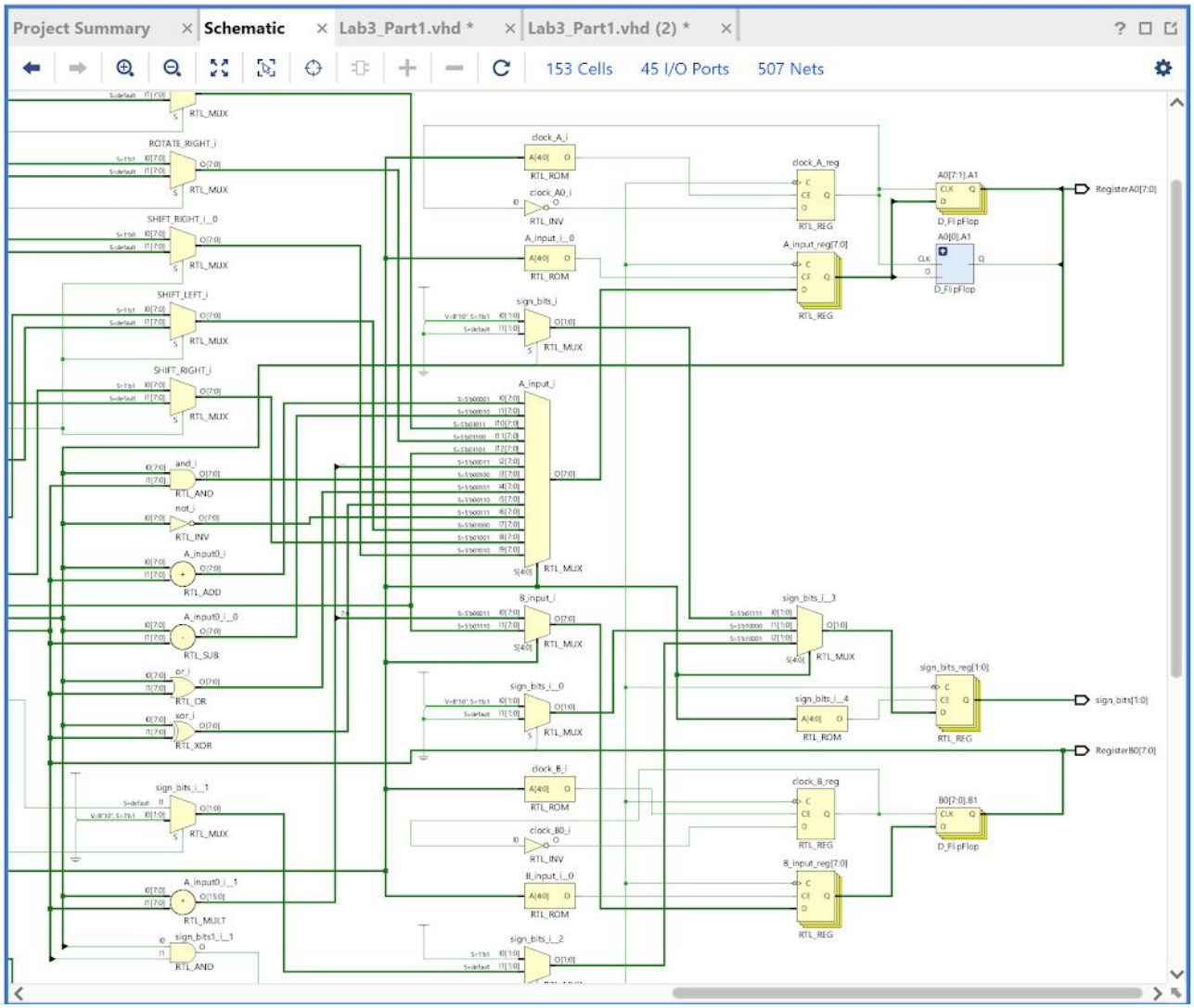


Figure 6. Enlarged ALU Schematic for Task 1, showing opcode selection MUX and explicitly-defined registers.

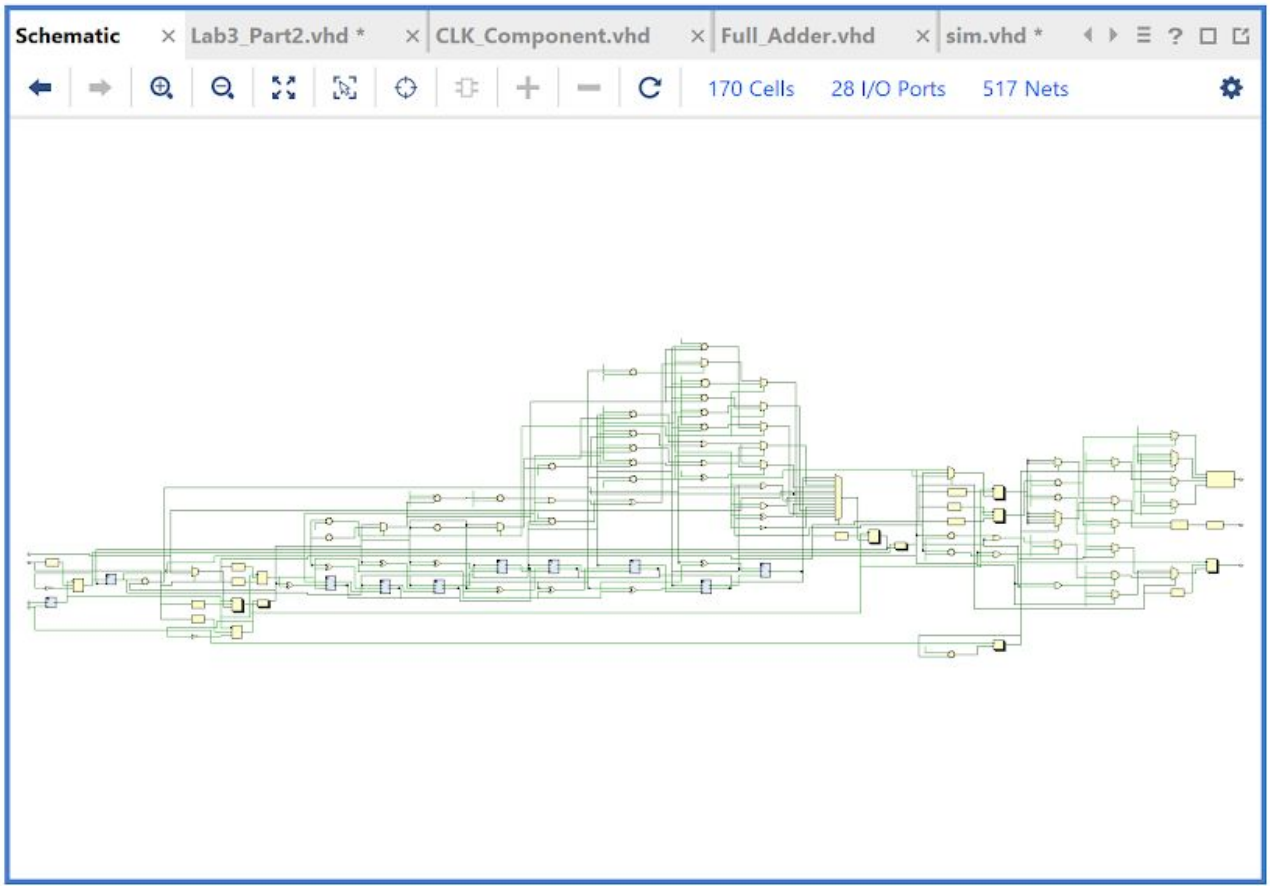


Figure 7. ALU Schematic for Task 2.

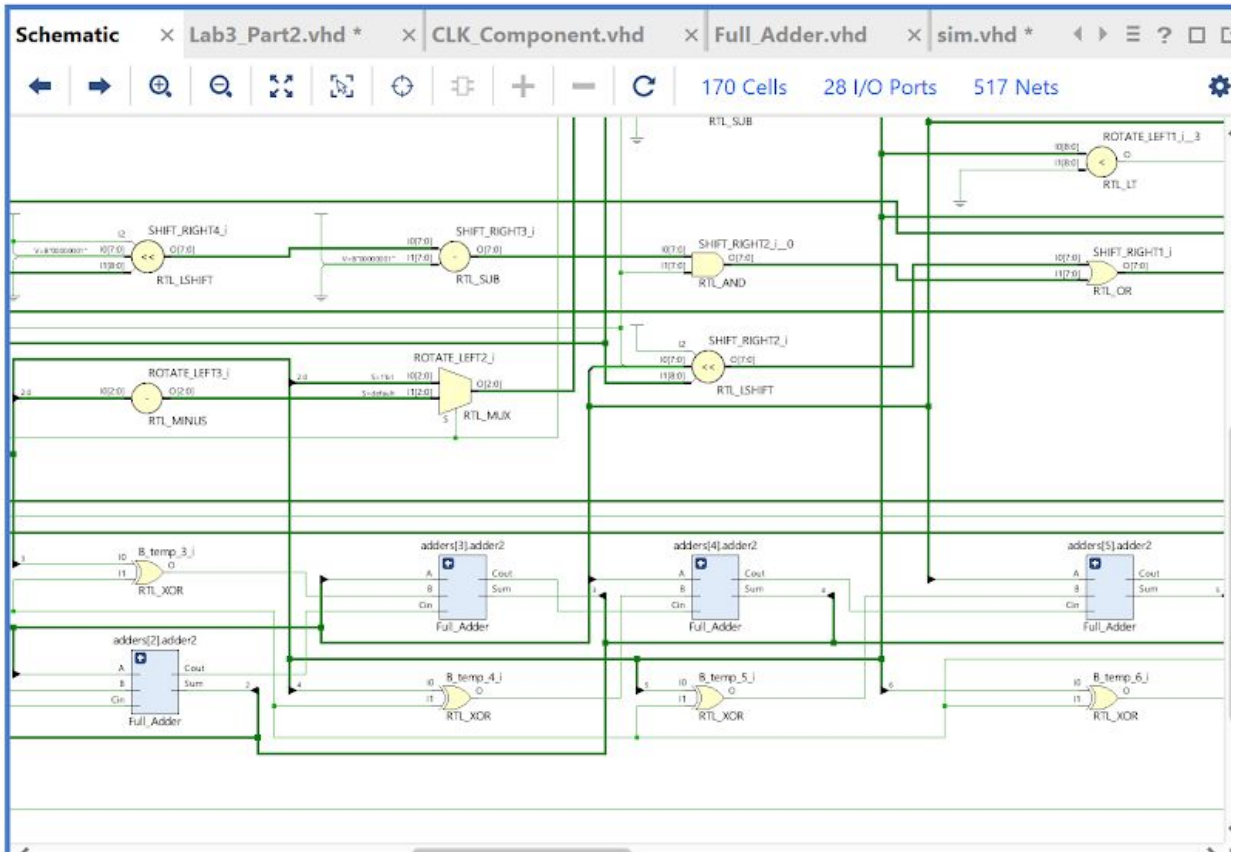


Figure 8. Enlarged ALU Schematic for Task 2, showing structurally-implemented full adder instances.

The last important result of this lab is of course the VHDL code I wrote to implement the ALU functionality. All the code I wrote is shown in Appendix A, Appendix B, and Appendix B. I described it in much detail in my procedure section so I won't reiterate here. In summary, I created a simple D Flip Flop component and instantiated it in my main .vhd program to create registers A0 and B0. Then I use a large case statement to implement my opcode selection logic, and perform the various operations on the contents of registers A0 and B0 based on the opcode. For Task 1 I use type case functions and VHDL-provided operators to do this, while for Task 2 I do almost exactly the same thing. The only difference is that I create separate components to create a full adder (NOT, AND, OR, XOR) and instantiate several of these to create an 8-bit carry ripple adder/subtractor to implement addition and subtraction logic. Then I use a few processes like a clock divider and 7-segment case statement to determine which registers to display on the BASYS3 board. Finally, my simulation code is included at the end.

Conclusion

In conclusion, my implementation for Task 1 and Task 2 was perfect. My ALU design performed as required by the lab specifications, correctly performing arithmetic, bitwise, and boolean functions on inputs A and B in registers A0 and B0. My simulations and the successful performance of my implementation on the BASYS3 board are proof of this. I ran into very few issues on this lab, as I had a good understanding of what to do from the very beginning. One obstacle I ran into was a Vivado error message that kept occurring around my generate statement, but I figured out that it was a syntax error. The biggest source of time it took to complete this lab was in the design phase. I wasn't sure exactly the best way to organize opcodes, so in the end I decided to assign them according to the order in which they are presented in the lab instructions, for simplicity purposes. I also had problems writing simulations, because I'm not as experienced writing those as I should be.

Some potential improvements I could make on my ALU design could be to create more operations like logical XNOR or maybe more complicated arithmetic like division. If I was really ambitious and had a lot of free time, I could try to offer floating point operations that could do the same computations I implemented in this lab, but on non-integers as well as integers. This would honestly be much more useful and rewarding to implement, but it would certainly be more complicated than what I have currently done in this lab.

Overall I'm really happy with how my ALU design functions. I feel like I have gained much more knowledge of VHDL too. For example, this is the first time I used type cast functions and types other than std_logic or std_logic_vector (like signed and unsigned). As I do more and more work with VHDL the more comfortable I am with it. Even as an experienced C++ programmer I really like the unique nature of how VHDL works, how it's implementing hardware concurrently and everything. I also know much more about how the internal mechanisms of ALUs work, which will be valuable information to aid me in future classes and jobs. This project could also serve as inspiration for future projects (remember that floating point idea?) and as a cool illustration of my VHDL knowledge to family, friends, and colleagues. At the completion of this lab, I ultimately feel much more competent in my digital design ability and knowledge of VHDL.

Appendix A: Task 1 Code

"Main" code from Task 1

Project Summary | Schematic | Lab3_Part1.vhd | Lab3_Part1.vhd (2) *

C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sources_1/new/Lab3_Part1.vhd

```
-- Mark Lee      ECE 351/357 Lab 3       12 November 2019
--
-- This VHDL program implements an arithmetic logic unit (ALU)
-- that performs such arithmetic, bitwise, and boolean operations
-- on two 8-bit signed binary integers. These values are stored
-- in explicitly-declared registers A0 and B0. The operations to be
-- performed are selected through various opcodes input to the BASYS3
-- board, and the program outputs results on the LEDs and 7-segment
-- displays if desired. The program also offers load/read functions
-- to enable more user control of register contents.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.ALL;
use ieee.std_logic_unsigned.all;

entity Lab3_Part1 is
    Port (Button, Clock: in std_logic;
          input: in signed(7 downto 0);
          op_code: in std_logic_vector(4 downto 0);
          Anodes: out std_logic_vector(3 downto 0);
          Cathodes: out std_logic_vector(6 downto 0);
          sign_bits: out std_logic_vector(1 downto 0));
end Lab3_Part1;

architecture Behavioral of Lab3_Part1 is

--Necessary signals for ALU function
signal DB_pulse: std_logic;
signal clock_A, clock_B: std_logic;
signal A_input, B_input, A_output, B_output: signed(7 downto 0);
signal LED_BCD: signed(3 downto 0);
signal counter: std_logic_vector(19 downto 0);
signal output: signed(15 downto 0);
signal anodes_sig: std_logic_vector(3 downto 0);
```

```
signal output: signed(15 downto 0);
signal anodes_sig: std_logic_vector(3 downto 0);

--Instantiate the debounced pulse functionality
component DB_CLK
    port(CLK, BTNC: in std_logic;
         DB: out std_logic);
end component;

begin
    --Instantiate the debounced pulse functionality and generate 16 D flip flop
    --instances for the two registers. Map pins to above signals.
    CLK_Component : DB_CLK port map(CLK => Clock, BTNC => Button, DB => DB_pulse);
    A0: for i in 0 to 7 generate
        A1: entity work.D_FlipFlop(Behavioral)
            port map(CLK => clock_A, D => A_input(i), Q => A_output(i));
        end generate;
    B0: for i in 0 to 7 generate
        B1: entity work.D_FlipFlop(Behavioral)
            port map(CLK => clock_B, D => B_input(i), Q => B_output(i));
        end generate;

    --This process is the main "logic" of the code. It features a case statement that
    --selects different operations to perform on register contents based on the code.
    --Results typically stored in register A0.
    process is
        variable product: signed(15 downto 0);
    begin
        wait until falling_edge(DB_pulse);

        case op_code is
            when "00001" =>
                A_input <= A_output + B_output;
                clock_A <= not clock_A;
```

```
68          when "00010" =>
69              A_input <= A_output - B_output;
70              clock_A <= not clock_A;
71          when "00011" =>
72              product := A_output * B_output;
73              A_input <= product(15 downto 8);
74              B_input <= product(7 downto 0);
75              clock_A <= not clock_A;
76              clock_B <= not clock_B;
77          when "00100" =>
78              A_input <= A_output and B_output;
79              clock_A <= not clock_A;
80          when "00101" =>
81              A_input <= A_output or B_output;
82              clock_A <= not clock_A;
83          when "00110" =>
84              A_input <= A_output xor B_output;
85              clock_A <= not clock_A;
86          when "00111" =>
87              A_input <= not A_output;
88              clock_A <= not clock_A;
```

```
89          when "01000" =>
90              A_input <= signed(shift_left(unsigned(A_output), to_integer(B_output)));
91              clock_A <= not clock_A;
92          when "01001" =>
93              A_input <= signed(shift_right(unsigned(A_output), to_integer(B_output)));
94              clock_A <= not clock_A;
95          when "01010" =>
96              A_input <= shift_right(signed(A_output), to_integer(B_output));
97              clock_A <= not clock_A;
98          when "01011" =>
99              A_input <= rotate_left(A_output, to_integer(B_output));
100             clock_A <= not clock_A;
101         when "01100" =>
102             A_input <= rotate_right(A_output, to_integer(B_output));
103             clock_A <= not clock_A;
104         when "01101" =>
105             --A_signed <= signed(input);
106             A_input <= input;
107             clock_A <= not clock_A;
108         when "01110" =>
109             --B_signed <= signed(input);
110             B_input <= input;
111             clock_B <= not clock_B;
112         when "01111" =>
113             output(7 downto 0) <= A_output;
114             anodes_sig <= "0011";
115
116             if(A_output(7) = '1') then
117                 sign_bits <= "10";
118             else
119                 sign_bits <= "00";
120             end if;
```

```
122          when "10000" =>
123              output(7 downto 0) <= B_output;
124              anodes_sig <= "0011";
125
126              if(B_output(7) = '1') then
127                  sign_bits <= "10";
128              else
129                  sign_bits <= "00";
130              end if;
131
132          when "10001" =>
133              output(7 downto 0) <= B_output;
134              output(15 downto 8) <= A_output;
135              anodes_sig <= "1111";
```

```
136
137                 if(A_output(7) = '1' and B_output(7) = '1') then
138                     sign_bits <= "11";
139                 elsif(A_output(7) = '1' and B_output(7) = '0') then
140                     sign_bits <= "10";
141                 elsif(A_output(7) = '0' and B_output(7) = '1') then
142                     sign_bits <= "01";
143                 else
144                     sign_bits <= "00";
145                 end if;
146
147             when others =>
148         end case;
149     end process;
150
151     --Increment counter until counter(18) changes value --> approximately 10ms has passed
152     process(Clock)
153     begin
154         if(rising_edge(Clock)) then
155             counter <= counter + 1;
156         end if;
157     end process;
158
```

```
159     --When the value for LED_BCD changes, change the output on the display
160     process(LED_BCD)
161     begin
162         case LED_BCD is
163             when "0000" => Cathodes <= "0000001"; -- "0"
164             when "0001" => Cathodes <= "1001111"; -- "1"
165             when "0010" => Cathodes <= "0010010"; -- "2"
166             when "0011" => Cathodes <= "0000110"; -- "3"
167             when "0100" => Cathodes <= "1001100"; -- "4"
168             when "0101" => Cathodes <= "0100100"; -- "5"
169             when "0110" => Cathodes <= "0100000"; -- "6"
170             when "0111" => Cathodes <= "0001111"; -- "7"
171             when "1000" => Cathodes <= "0000000"; -- "8"
172             when "1001" => Cathodes <= "0000100"; -- "9"
173             when "1010" => Cathodes <= "0000010"; -- "A"
174             when "1011" => Cathodes <= "1100000"; -- "b"
175             when "1100" => Cathodes <= "0110001"; -- "C"
176             when "1101" => Cathodes <= "1000010"; -- "d"
177             when "1110" => Cathodes <= "0110000"; -- "E"
178             when "1111" => Cathodes <= "0111000"; -- "F"
179             when others =>
180         end case;
181     end process;
182
```

```
182
183     --When 10ms has passed, alternate displays.
184     --Additional logic added based on whether the user wants to display only
185     --the lower 2 displays (one register), or all 4 (both registers)
186     process(counter(18))
187     begin
188         if(anodes_sig = "0011") then
189             case counter(19) is
190                 when '0' =>
191                     Anodes <= "1110";
192                     LED_BCD <= output(3 downto 0);
193                 when '1' =>
194                     Anodes <= "1101";
195                     LED_BCD <= output(7 downto 4);
196                 when others =>
197             end case;
198         elsif(anodes_sig = "1111") then
199             case counter(19 downto 18) is
200                 when "00" =>
201                         Anodes <= "1110";
202                         LED_BCD <= output(3 downto 0);
203                 when "01" =>
204                         Anodes <= "1101";
205                         LED_BCD <= output(7 downto 4);
206                 when "10" =>
207                         Anodes <= "1011";
208                         LED_BCD <= output(11 downto 8);
209                 when "11" =>
210                         Anodes <= "0111";
211                         LED_BCD <= output(15 downto 12);
212                 when others =>
213             end case;
214         end if;
215     end process;
216 end Behavioral;
217
```

D Flip Flop code used to implement registers

Project Summary | Schematic | Lab3_Part1.vhd | Lab3_Part1.vhd (2) * | D_FlipFlop.vhd

C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sources_1/new/D_FlipFlop.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity D_FlipFlop is
5      Port (CLK, D: in std_logic;
6            Q: out std_logic);
7  end D_FlipFlop;
8
9  architecture Behavioral of D_FlipFlop is
10 begin
11     process(CLK)
12     begin
13         Q <= D;
14     end process;
15 end Behavioral;
16
```

Appendix B: Task 2 Code

"Main" program code ALTERED from Task 1 (Full Adder Generation)

```
41 begin
42     --Instantiate the debounced pulse functionality and 16 D Flip Flops for the registers
43     CLK_Component : DB_CLK port map(CLK => Clock, BTNC => Button, DB => DB_pulse);
44     A0: for i in 0 to 7 generate
45         A1: entity work.D_FlipFlop(Behavioral)
46             port map(CLK=>clock_A, D=>A_input(i), Q=>A_output(i));
47         end generate;
48     B0: for i in 0 to 7 generate
49         B1: entity work.D_FlipFlop(Behavioral)
50             port map(CLK=>clock_B, D=>B_input(i), Q=>B_output(i));
51         end generate;
52
53     --Instantiate 8 full adders to implement an 8-bit carry ripple adder with subtraction
54     --capability. Map signals as appropriate
55     B_temp(0) <= B_output(0) xor sub;
56     adder1: entity work.Full_Adder(Behavioral)
57             port map(A => A_output(0), B => B_temp(0), Cin => sub, Cout => C_temp(0), Sum => sum(0));
58     adders: for i in 1 to 7 generate
59                 B_temp(i) <= B_output(i) xor Sub;
60                 adder2: entity work.Full_Adder(Behavioral)
61                         port map(A => A_output(i), B => B_temp(i), Cin => C_temp(i-1), Cout => C_temp(i), Sum => sum(i));
62             end generate;
63
64     process is
65         variable product: signed(15 downto 0);
66     begin
67         wait until falling_edge(DB_pulse);
68
```

```
68
69      case op_code is
70          when "00001" =>
71              sub <= '0';
72              A_input <= sum;
73              clock_A <= not clock_A;
74          when "00010" =>
75              sub <= '1';
76              A_input <= sum;
77              clock_A <= not clock_A;
78          when "00011" =>
79              product := A_output * B_output;
80              A_input <= product(15 downto 8);
81              B_input <= product(7 downto 0);
82              clock_A <= not clock_A;
83              clock_B <= not clock_B;
84          when "00100" =>
85              A_input <= A_output and B_output;
86              clock_A <= not clock_A;
87          when "00101" =>
88              A_input <= A_output or B_output;
89              clock_A <= not clock_A;
90          when "00110" =>
91              A_input <= A_output xor B_output;
92              clock_A <= not clock_A;
93          when "00111" =>
94              A_input <= not A_output;
95              clock_A <= not clock_A;
96          when "01000" =>
```

Full Adder code and sub-components

art2.vhd * | CLK_Component.vhd | Full_Adder.vhd | sim.vhd * | NOT_Gate.vhd

C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part2/Lab3_Part2.srcs/sources_1/new/NOT_Gate.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity NOT_Gate is
5      Port (A: in std_logic;
6            Q: out std_logic);
7  end NOT_Gate;
8
9  architecture Behavioral of NOT_Gate is
10
11 begin
12     Q <= not A;
13 end Behavioral;
```

Component.vhd | Full_Adder.vhd | sim.vhd * | NOT_Gate.vhd | **AND_Gate.vhd**

C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part2/Lab3_Part2.srcs/sources_1/new/AND_Gate.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity AND_Gate is
5      Port (A, B: in std_logic;
6            Q: out std_logic);
7  end AND_Gate;
8
9  architecture Behavioral of AND_Gate is
10
11 begin
12     Q <= A and B;
13 end Behavioral;
14
```

sim.vhd * | NOT_Gate.vhd | AND_Gate.vhd | XOR_Gate.vhd | **OR_Gate.vhd**

C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part2/Lab3_Part2.srcs/sources_1/new/OR_Gate.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity OR_Gate is
5      Port (A, B: in std_logic;
6            Q: out std_logic);
7  end OR_Gate;
8
9  architecture Behavioral of OR_Gate is
10
11 begin
12     Q <= A or B;
13 end Behavioral;
14
```

Full_Adder.vhd | sim.vhd * | NOT_Gate.vhd | AND_Gate.vhd | XOR_Gate.vhd

C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part2/Lab3_Part2.srcs/sources_1/new/XOR_Gate.vhd

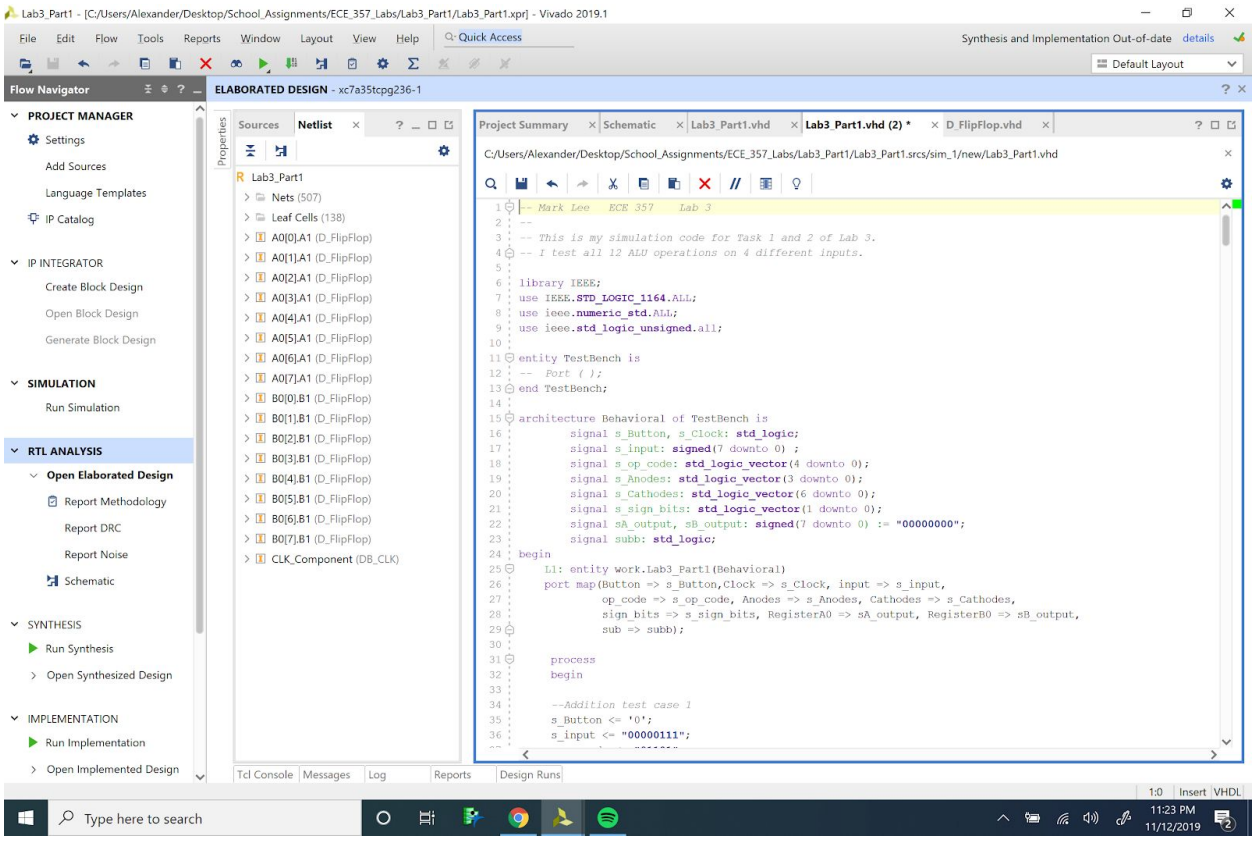
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity XOR_Gate is
5      Port (A, B: in std_logic;
6            Q: out std_logic);
7  end XOR_Gate;
8
9  architecture Behavioral of XOR_Gate is
10
11 begin
12     Q <= A xor B;
13 end Behavioral;
14
```

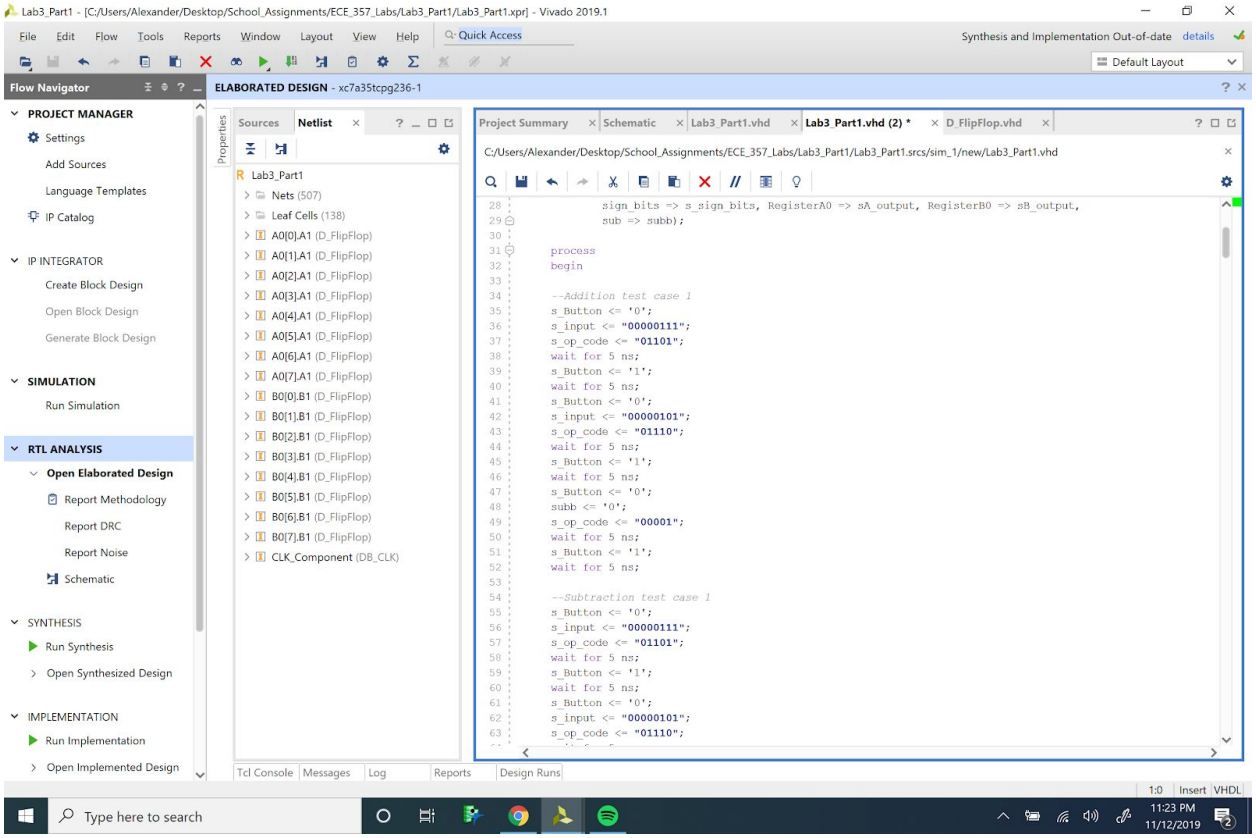
Lab3_Part2.vhd * | CLK_Component.vhd | Full_Adder.vhd | sim.vhd * | NOT_Gate.vhd | AND_Gate.vhd | XOR_Gate.vhd | OR_Gate.vhd

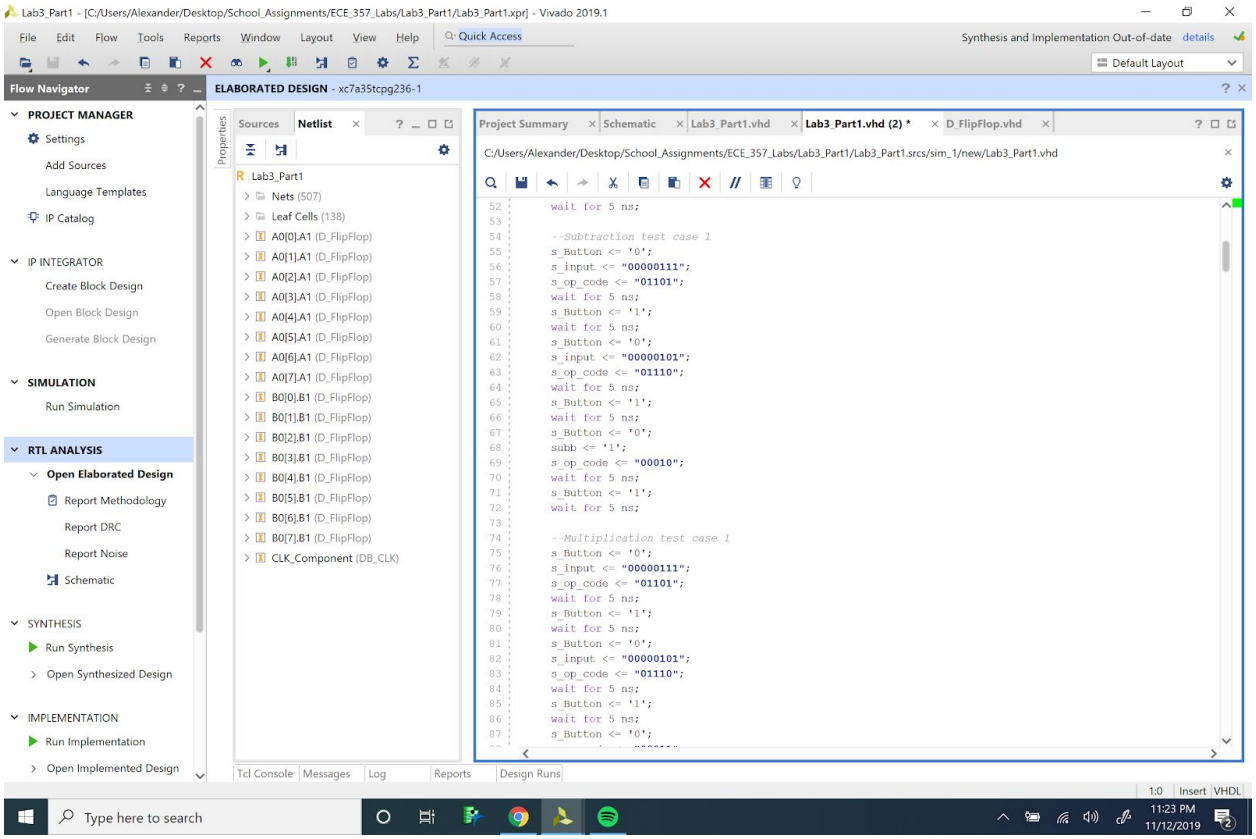
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part2/Lab3_Part2.srcs/sources_1/new/Full_Adder.vhd

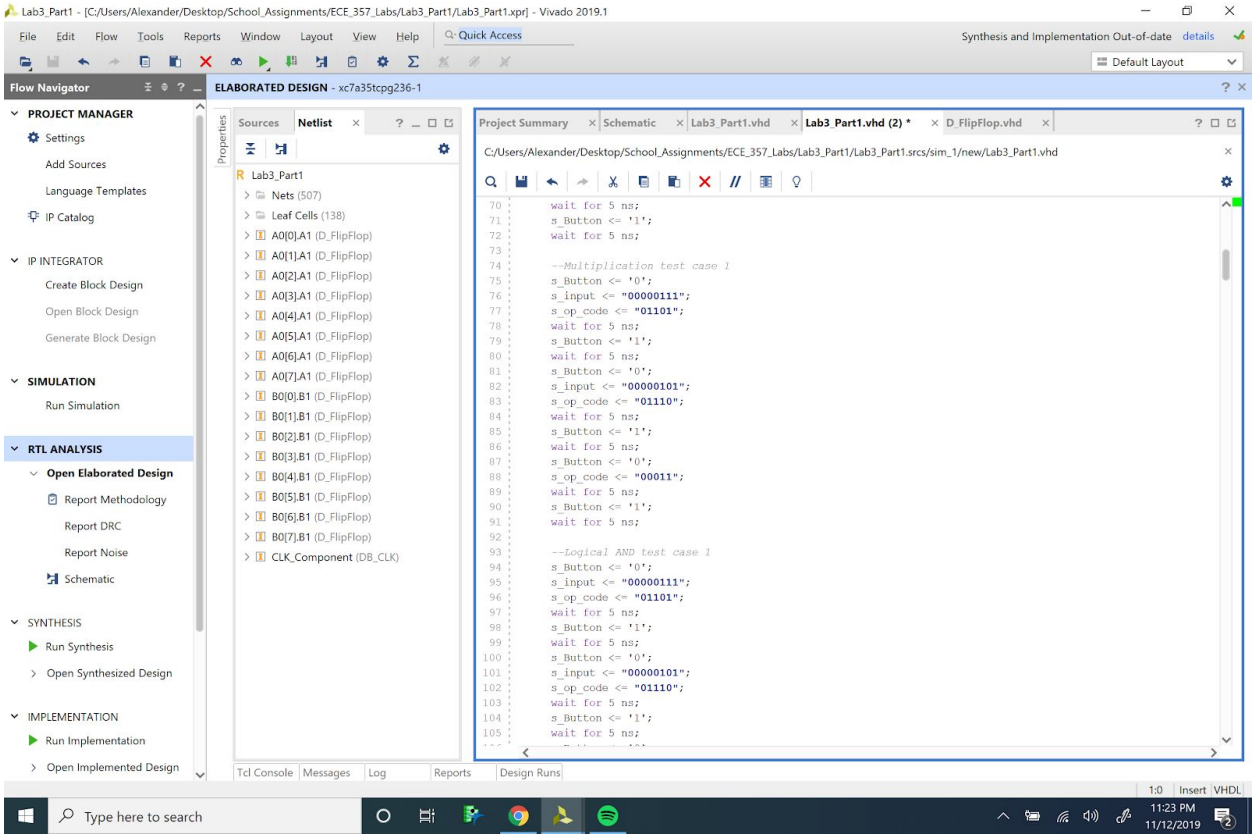
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Full_Adder is
5      Port (A, B, Cin: in std_logic;
6            Sum, Cout: out std_logic);
7  end Full_Adder;
8
9  architecture Behavioral of Full_Adder is
10
11     signal temp1, temp2, temp3: std_logic;
12
13     component AND_Gate
14         port(A, B: in std_logic;
15              Q: out std_logic);
16     end component;
17
18     component OR_Gate
19         port(A, B: in std_logic;
20              Q: out std_logic);
21     end component;
22
23     component XOR_Gate
24         port(A, B: in std_logic;
25              Q: out std_logic);
26     end component;
27
28 begin
29
30    AND1: AND_Gate port map(A => A, B => B, Q => temp1);
31    AND2: AND_Gate port map(A => temp3, B => Cin, Q => temp2);
32    XOR1: XOR_Gate port map(A => A, B => B, Q => temp3);
33    XOR2: XOR_Gate port map(A => temp3, B => Cin, Q => Sum);
34    OR1: OR_Gate port map(A => temp1, B => temp2, Q => Cout);
35
36 end Behavioral;
```

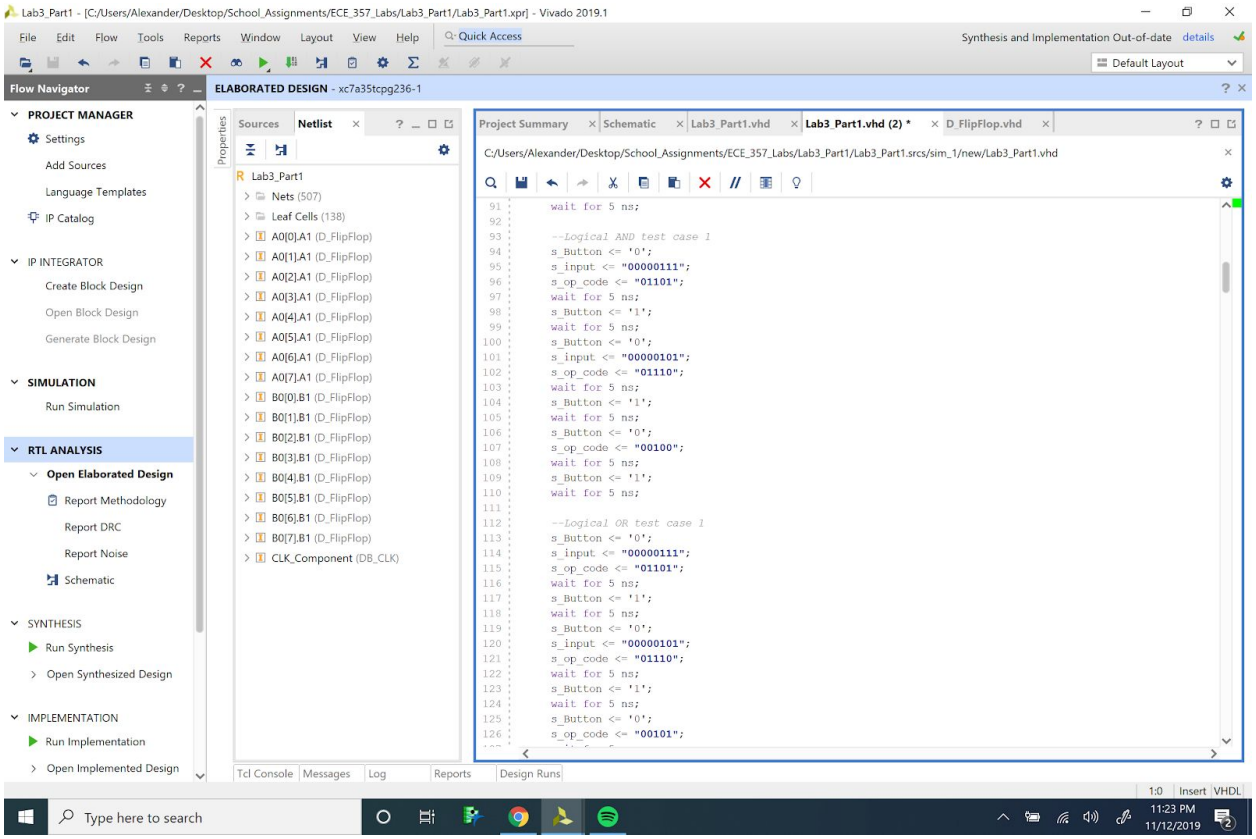
Appendix C: Simulation Code for Task 1 and Task 2 (First input only, for space purposes)

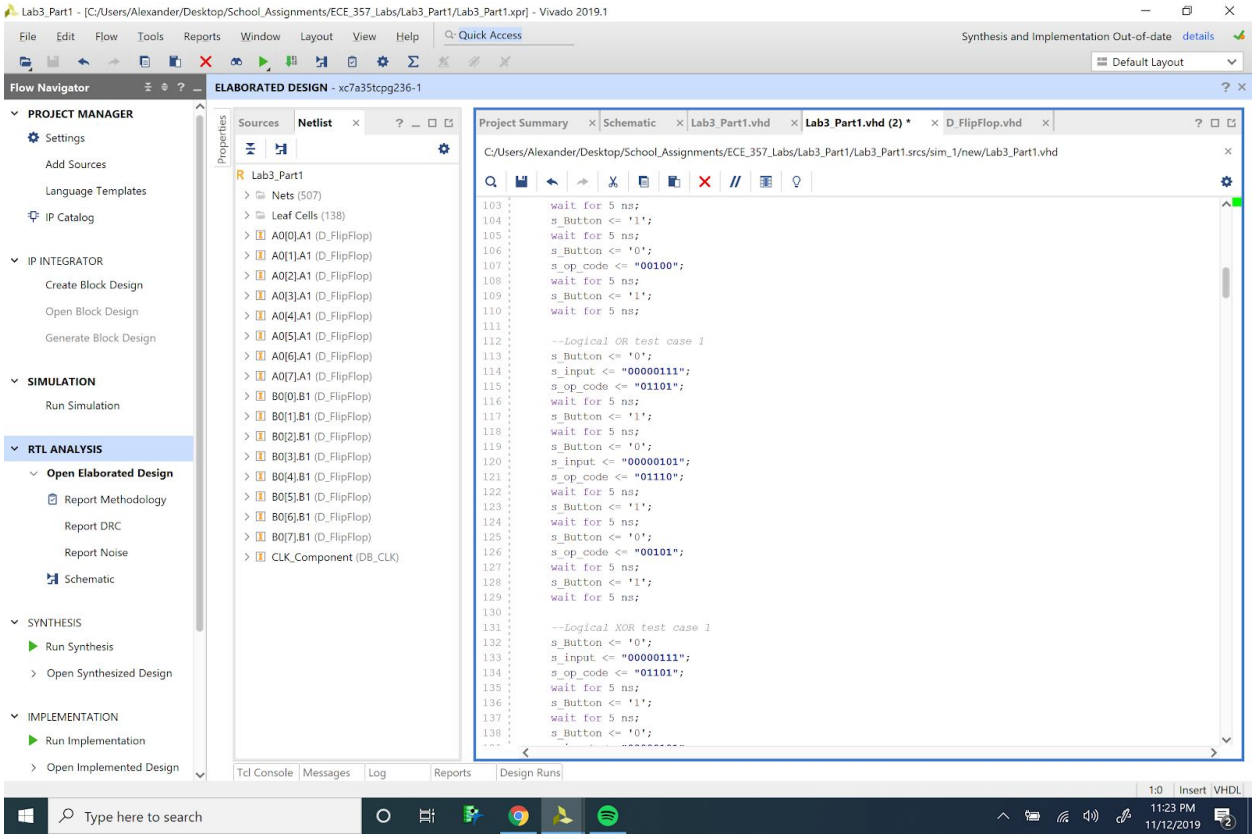


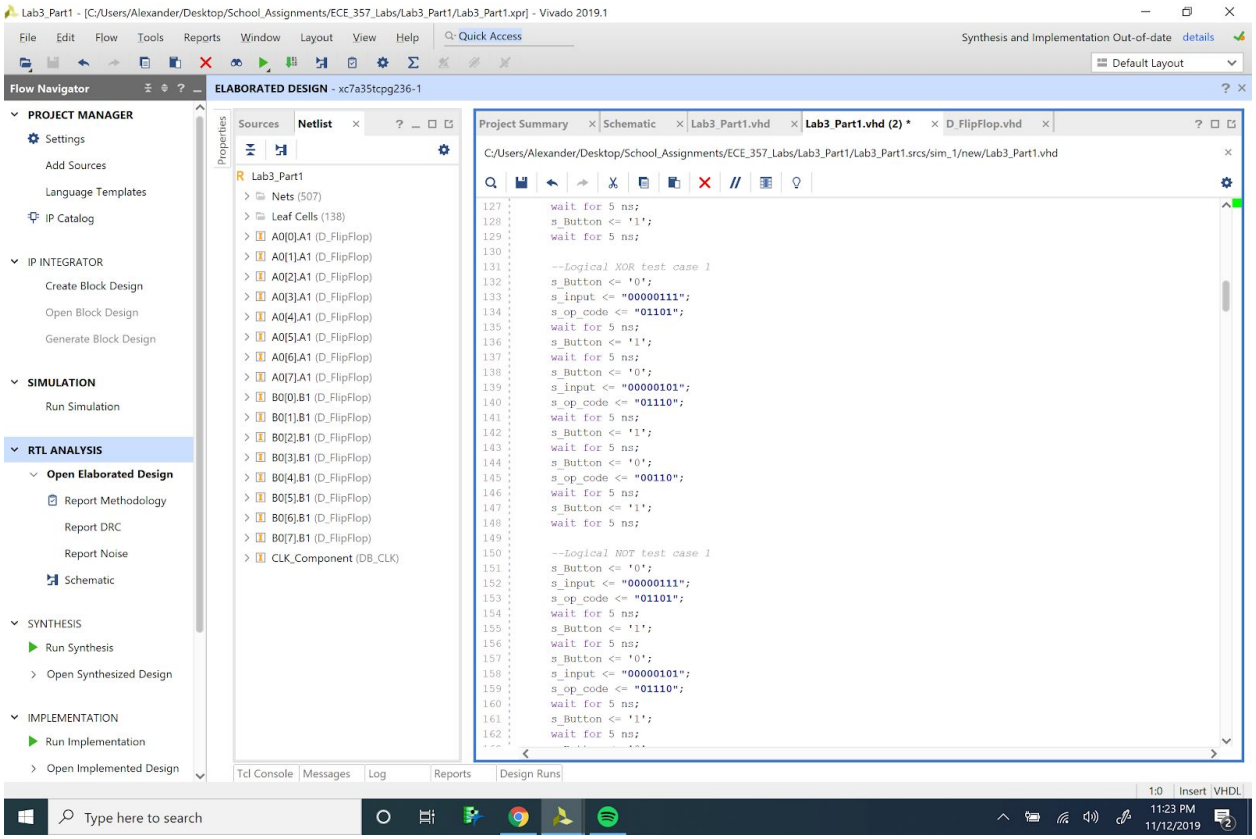

Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
ELABORATED DESIGN - xc7a35tcpg236-1
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
sign_bits => s_sign_bits, RegisterA0 => sA_output, RegisterB0 => sB_output,
sub => subb);
process
begin
--Addition test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
subb <= '0';
s_op_code <= "00001";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Subtraction test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019

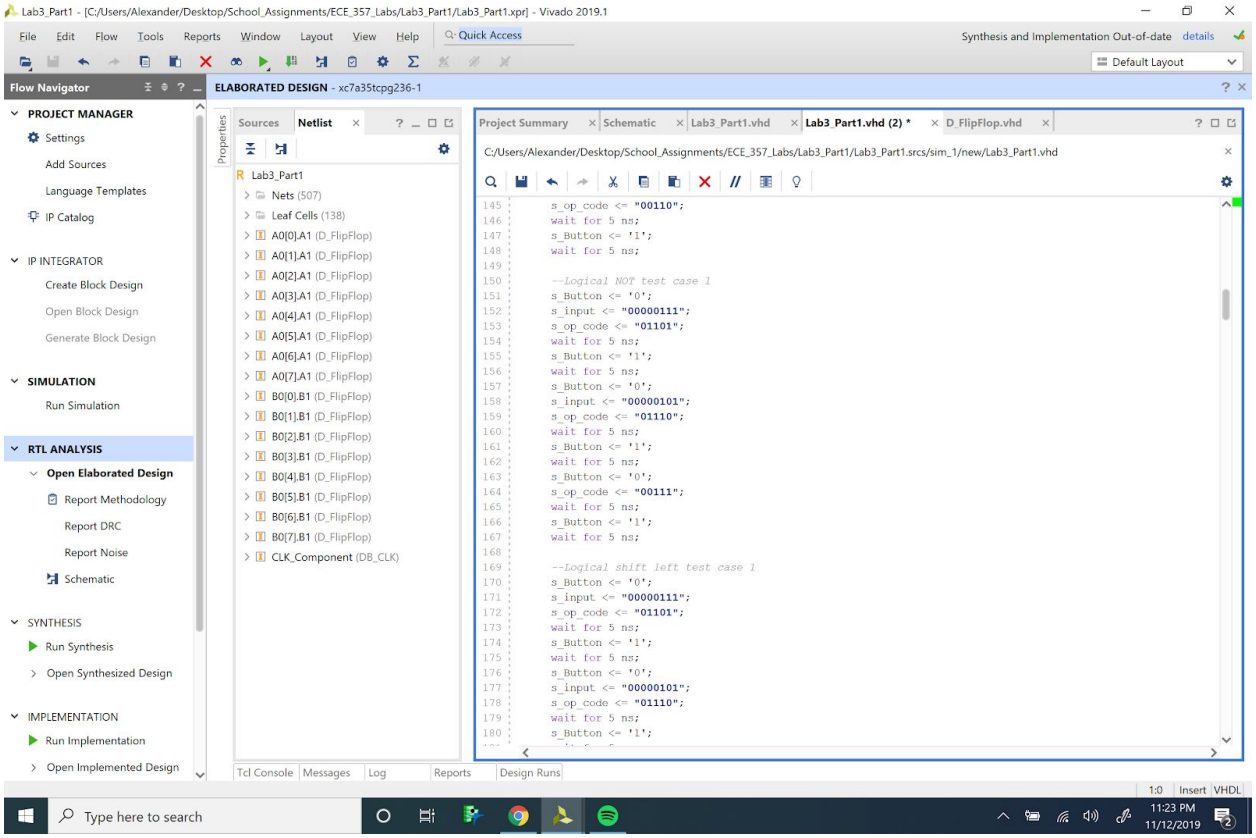

Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
ELABORATED DESIGN - xc7a35tcpg236-1
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
wait for 5 ns;
--Subtraction test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
subb <= '1';
s_op_code <= "00010";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Multiplication test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019

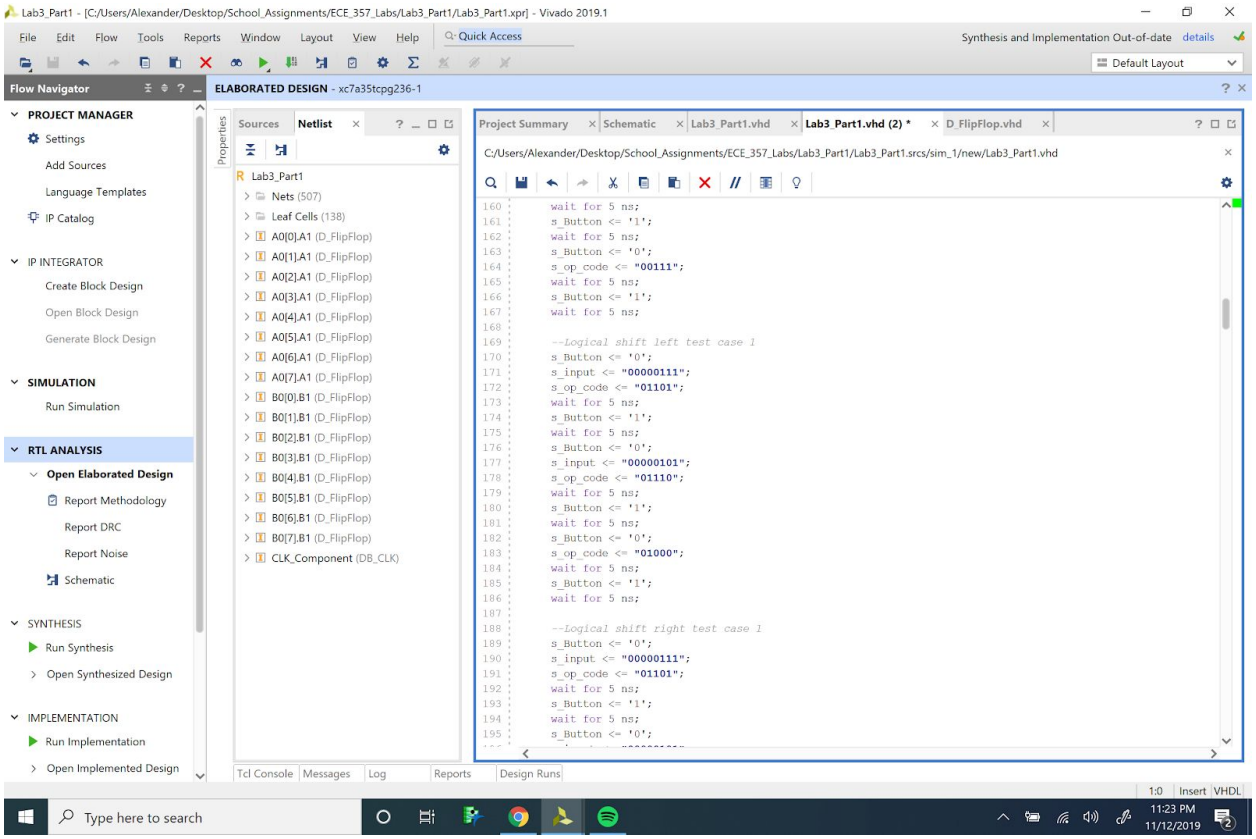

Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
ELABORATED DESIGN - xc7a35tcpg236-1
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
70 wait for 5 ns;
71 s_Button <= '1';
72 wait for 5 ns;
73
74 --Multiplication test case 1
75 s_Button <= '0';
76 s_input <= "00000111";
77 s_op_code <= "01101";
78 wait for 5 ns;
79 s_Button <= '1';
80 wait for 5 ns;
81 s_Button <= '0';
82 s_input <= "00000101";
83 s_op_code <= "01110";
84 wait for 5 ns;
85 s_Button <= '1';
86 wait for 5 ns;
87 s_Button <= '0';
88 s_op_code <= "00011";
89 wait for 5 ns;
90 s_Button <= '1';
91 wait for 5 ns;
92
93 --Logical AND test case 1
94 s_Button <= '0';
95 s_input <= "00000111";
96 s_op_code <= "01101";
97 wait for 5 ns;
98 s_Button <= '1';
99 wait for 5 ns;
100 s_Button <= '0';
101 s_input <= "00000101";
102 s_op_code <= "01110";
103 wait for 5 ns;
104 s_Button <= '1';
105 wait for 5 ns;
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019

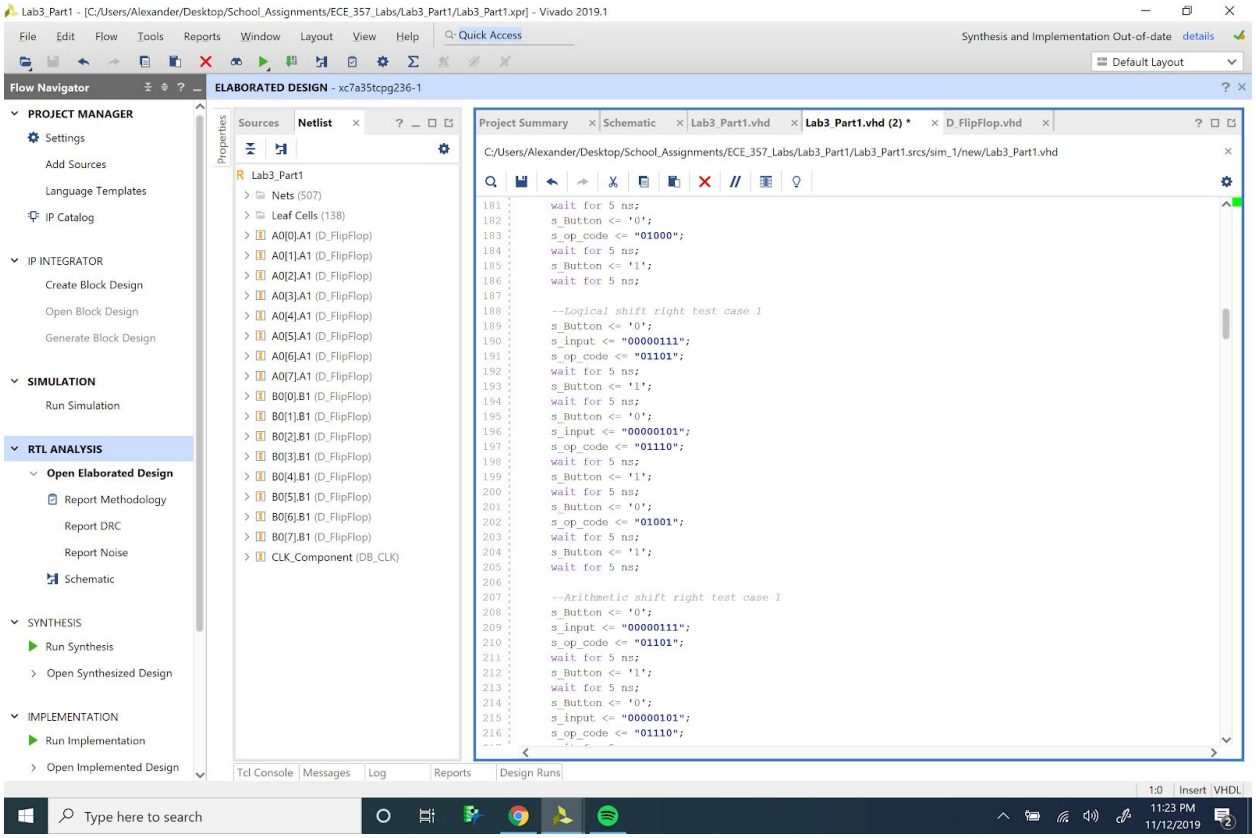

Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
ELABORATED DESIGN - xc7a35tcpg236-1
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
91 wait for 5 ns;
92
93 --Logical AND test case 1
94 s_Button <= '0';
95 s_input <= "00000111";
96 s_op_code <= "01101";
97 wait for 5 ns;
98 s_Button <= '1';
99 wait for 5 ns;
100 s_Button <= '0';
101 s_input <= "00000101";
102 s_op_code <= "01110";
103 wait for 5 ns;
104 s_Button <= '1';
105 wait for 5 ns;
106 s_Button <= '0';
107 s_op_code <= "00100";
108 wait for 5 ns;
109 s_Button <= '1';
110 wait for 5 ns;
111
112 --Logical OR test case 1
113 s_Button <= '0';
114 s_input <= "00000111";
115 s_op_code <= "01101";
116 wait for 5 ns;
117 s_Button <= '1';
118 wait for 5 ns;
119 s_Button <= '0';
120 s_input <= "00000101";
121 s_op_code <= "01110";
122 wait for 5 ns;
123 s_Button <= '1';
124 wait for 5 ns;
125 s_Button <= '0';
126 s_op_code <= "00101";
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019

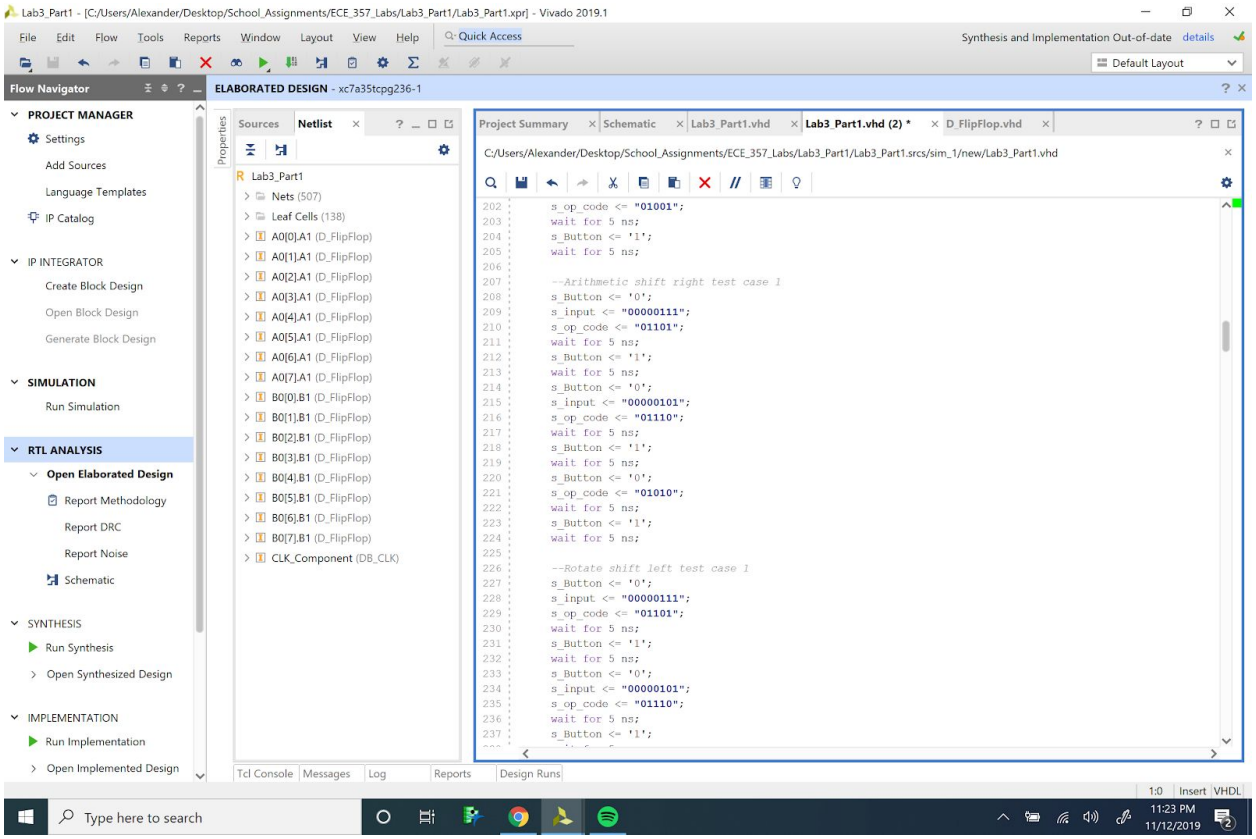

Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
ELABORATED DESIGN - xc7a35tcpg236-1
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "00100";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Logical OR test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "00101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Logical XOR test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019

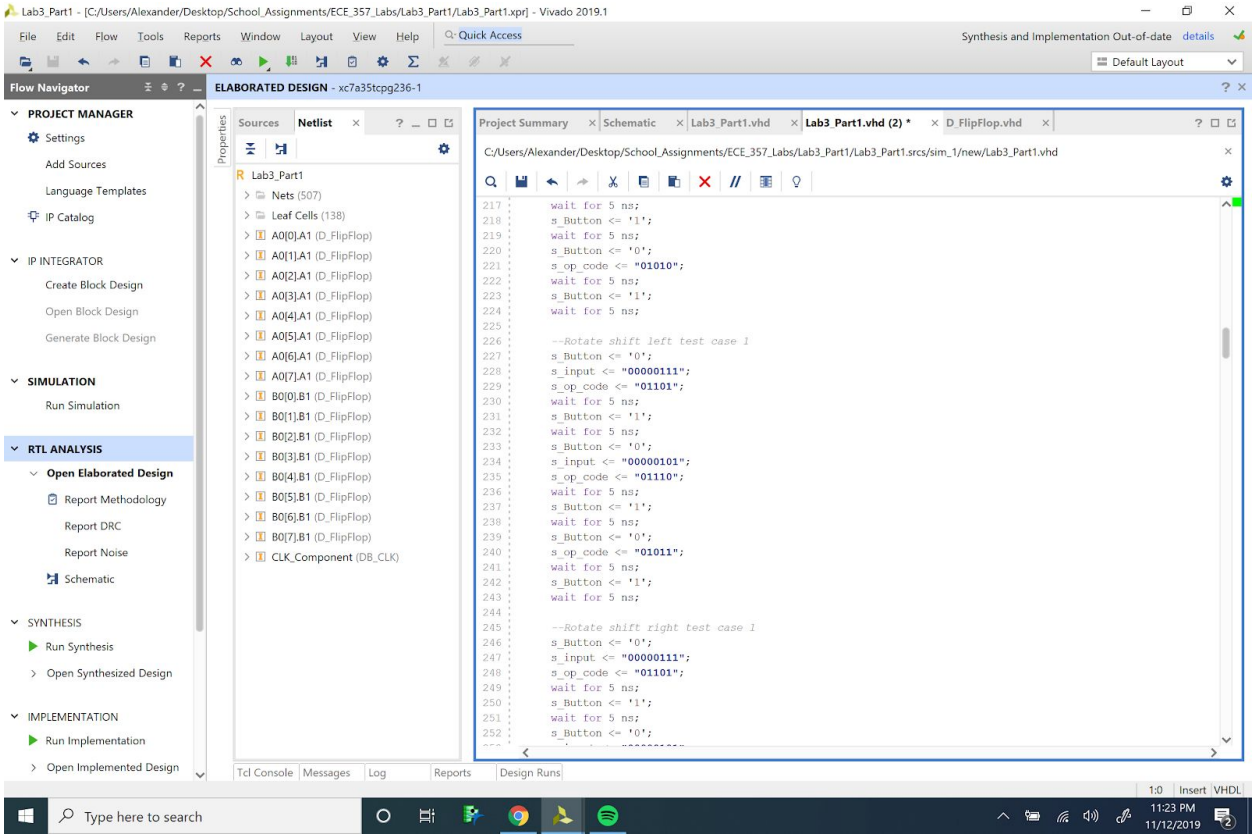

Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
ELABORATED DESIGN - xc7a35tcpg236-1
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Logical XOR test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "00110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Logical NOT test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019

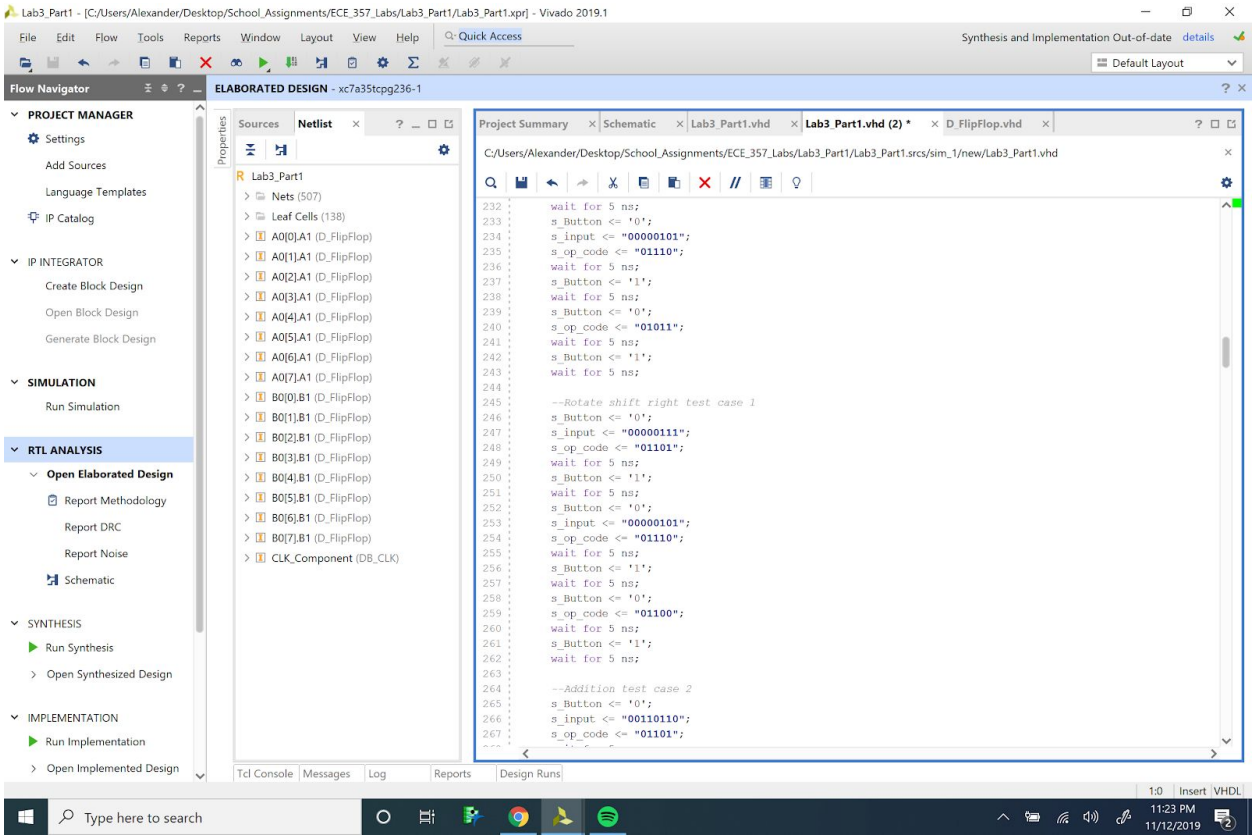

Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
ELABORATED DESIGN - xc7a35tcpg236-1
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
s_op_code <= "00110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Logical NOT test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "00111";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Logical shift left test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019


Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
ELABORATED DESIGN - xc7a35tcpg236-1
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "00111";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Logical shift left test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "01000";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Logical shift right test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019


Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
ELABORATED DESIGN - xc7a35tcpg236-1
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
wait for 5 ns;
s_Button <= '0';
s_op_code <= "01000";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Logical shift right test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "01001";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Arithmetic shift right test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019


Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
ELABORATED DESIGN - xc7a35tcpg236-1
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
s_op_code <= "01001";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Arithmetic shift right test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "01010";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Rotate shift left test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019


Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
ELABORATED DESIGN - xc7a35tcpg236-1
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "01010";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Rotate shift left test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "01011";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Rotate shift right test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019


Lab3_Part1 - [C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis and Implementation Out-of-date details
Default Layout
Flow Navigator
ELABORATED DESIGN - xc7a35tcpg236-1
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
Sources
Netlist
Lab3_Part1
Nets (507)
Leaf Cells (138)
A0[0].A1 (D_FlipFlop)
A0[1].A1 (D_FlipFlop)
A0[2].A1 (D_FlipFlop)
A0[3].A1 (D_FlipFlop)
A0[4].A1 (D_FlipFlop)
A0[5].A1 (D_FlipFlop)
A0[6].A1 (D_FlipFlop)
A0[7].A1 (D_FlipFlop)
B0[0].B1 (D_FlipFlop)
B0[1].B1 (D_FlipFlop)
B0[2].B1 (D_FlipFlop)
B0[3].B1 (D_FlipFlop)
B0[4].B1 (D_FlipFlop)
B0[5].B1 (D_FlipFlop)
B0[6].B1 (D_FlipFlop)
B0[7].B1 (D_FlipFlop)
CLK_Component (DB_CLK)
Project Summary
Schematic
Lab3_Part1.vhd
Lab3_Part1.vhd (2) *
D_FlipFlop.vhd
C:/Users/Alexander/Desktop/School_Assignments/ECE_357_Labs/Lab3_Part1/Lab3_Part1.srcs/sim_1/new/Lab3_Part1.vhd
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "01011";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Rotate shift right test case 1
s_Button <= '0';
s_input <= "00000111";
s_op_code <= "01101";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_input <= "00000101";
s_op_code <= "01110";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
s_Button <= '0';
s_op_code <= "01100";
wait for 5 ns;
s_Button <= '1';
wait for 5 ns;
--Addition test case 2
s_Button <= '0';
s_input <= "00110110";
s_op_code <= "01101";
Tcl Console
Messages
Log
Reports
Design Runs
1:0
Insert
VHDL
Type here to search
11:23 PM
11/12/2019