EECS 140: Lab 09 Report

Arithmetic Logic Unit

Morgan Bergen

KUID: 3073682

Date Submitted: 10/16/2022

1. Introduction and Background

Implement an arithmetic logic unit which will have the ability to take two four-bit binary numbers and depending on a two-bit selector either add, Logic AND, logic OR, or logic XOR the two inputs for a five-bit binary output. Inputs will range from the hexadecimal numbers 0 to F and outputs will range from 0 to 1E. Use Xilinx Vivado software to implement a ALU circuit. We will design the components in the (toplevel) with inputs and outputs of existing components. We will insert a ripple carry adder design and the 7-segment display unit to display hex numbers.

2. Implementation Process

Equipment used: Vivado Design Suite, DIGILENT BASYS 3 Board (Figure 1), PC and USB Communication Cable.

First, we are using a generic procedure to create VHDL Modular design. We are starting with small components and expanding to build our top-level entity.

Second, the source files have already been created and will use the ANDer.vhd, ORer.vhd, XORer.vhd, ADDer.vhd, function_select.vhd, display_driver.vhd, LEDdisplay.vhd, Bit_full_adder.vhd and toplevel.vhd. We downloaded them and then add the source files to our project.

Third, we will add 7 components to include in the toplevel.vhd file.

- the ANDer component.
- the ORer component.
- the XORer component.
- the ADDer component.
- the function_select component. o the display_driver component. o LEDdisplay component.

```
--Add component for function_select
component function_select is
    Port ( Input0 : in  STD_LOGIC_VECTOR (3 downto 0);
           Input1 : in  STD_LOGIC_VECTOR (3 downto 0);
              Input2 : in  STD_LOGIC_VECTOR (3 downto 0);
           Input3 : in  STD_LOGIC_VECTOR (4 downto 0);
           control : in  STD_LOGIC_VECTOR (1 downto 0);
           Output : out  STD_LOGIC_VECTOR (4 downto 0));
end component function_select;


-------------------------------------------------------------------------
--Add component for display_driver
component display_driver is
    Port ( inputs : in  STD_LOGIC_VECTOR (3 downto 0);
           seg_out : out  STD_LOGIC_VECTOR (6 downto 0));
end component display_driver;

-------------------------------------------------------------------------
--Add component for LEDdisplay
component LEDdisplay IS
    PORT (
          clk                   : IN  STD_LOGIC;
          seg0,seg1,seg2,seg3       : IN  STD_LOGIC_VECTOR(6 downto 0);
         seg             : OUT  STD_LOGIC_VECTOR(6  downto 0);
          an                    : OUT STD_LOGIC_VECTOR(3 downto 0));
END component LEDdisplay;
```

```
--Add component for ANDer
component ANDer is
    Port ( Input0 : in  STD_LOGIC_VECTOR (3 downto 0);
           Input1 : in  STD_LOGIC_VECTOR (3 downto 0);
           Output : out  STD_LOGIC_VECTOR (3 downto 0));
end component ANDer;


-------------------------------------------------------------
--Add component for ORer
component ORer is
    Port ( Input0 : in  STD_LOGIC_VECTOR (3 downto 0);
           Input1 : in  STD_LOGIC_VECTOR (3 downto 0);
           Output : out  STD_LOGIC_VECTOR (3 downto 0));
end component ORer;

-------------------------------------------------------------
--Add component for XORer
component XORer is
    Port ( Input0 : in  STD_LOGIC_VECTOR (3 downto 0);
           Input1 : in  STD_LOGIC_VECTOR (3 downto 0);
           Output : out  STD_LOGIC_VECTOR (3 downto 0));
end component XORer;

-------------------------------------------------------------
--Add component for ADDer
component ADDer is
    Port ( Input0 : in  STD_LOGIC_VECTOR (3 downto 0);
           Input1 : in  STD_LOGIC_VECTOR (3 downto 0);
           carry_in : in  STD_LOGIC;
           Output : out  STD_LOGIC_VECTOR (4 downto 0));
end component ADDer;

-------------------------------------------------------------
```

```
-- port mapping --
ANDer0: ANDer port map (Input0 => sw1, Input1 => sw2, Output => out_and);
ORer0: ORer port map (Input0 => sw1, Input1 => sw2, Output => out_or);
XORer0: XORer port map (Input0 => sw1, Input1 => sw2, Output => out_xor);
ADDer0: ADDer port map (Input0 => sw1, Input1 => sw2, carry_in => carry_in, Output => out_adder);

LEDdisplay0: LEDdisplay port map (seg0=> sig_segments2, seg1=> sig_segments1, seg2=> sig_segments3, seg3=> sig_segments4, seg=> segments, an=> anodes, clk=> clk);
```

Fourth, we were provided with the toplevel.vhd file and port map the missing components. Which is ORer0, XORer0, ADDer0, and the LEDdisplay0.

Fifth, created an XDC file for toplevel inputs and outputs. All ports defined in toplevel entity were declared in the constraints.

Sixth, synthesize the design, implement the design, and generate a successful bitstream generation. Then download to FPGA board.

Last, verify the board is operating correctly by testing the inputs and outputs.

3. Evaluation Process

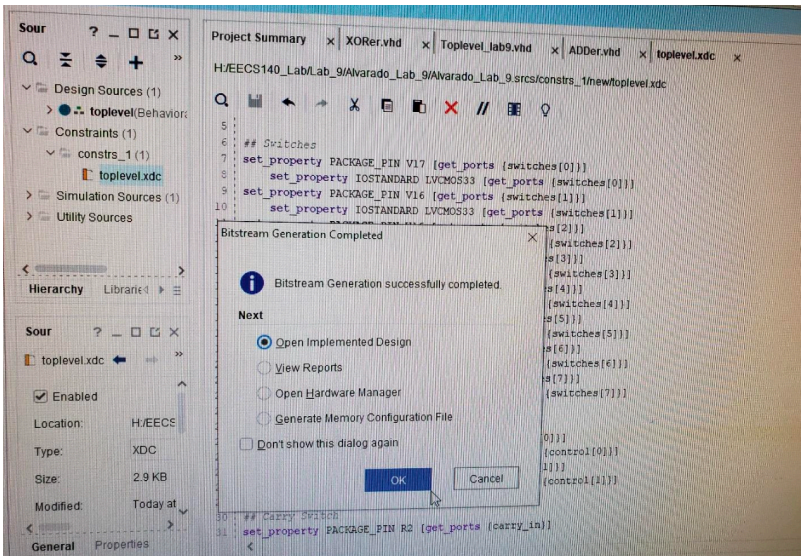
I had a successful synthesis and implementation. I did get an error when generating a bitstream. I had made a spelling error on my constraints and an error on the function select switches. I had to re-add square brackets because it was a vector. I ran a successful synthesis, implementation and bitstream.

4. Results & Discussion

Once my errors were all fixed, the 7-segment display outputs were in reverse order. I had to make changes to the LEDdisplay port mapping to display the correct output segments. Changes made were as follows:

LEDdisplay0 port map (seg0=>
sig_segments2, seg1=>
sig_segments1

Once port mapping was mapped correctly it passed the I/O test.



5. Conclusion & Recommendations

I am becoming more familiar with working with the VHDL code for the toplevel. I can interpret the VHDL code more clearly between the components, entity, and the behavior. This has been the 3rd lab that we had to work in the toplevel and I believe I understand it better.