Aryan Gupta
Marcos Rodrigues
Micaela Coco De Marco

Project 2 Writeup

To simplify reading, I have added the full list of the code to the bottom of this page.

Part 1

Part one was straight forward, however understanding what and how structural VHDL worked was very confusing. During the first part, we googled a lot and used the slides to understand this. The first part to understand what structural VHDL is understanding that structural VHDL defines a physical implementation. In this project we declared the components that we used in our implementation, this is C++ analogy of function prototypes, it tells VHDL that this black box is used somewhere, and here are the input and output. This was defined after the architectural statement and before the begin statement. An example is shown here.

```
architecture Behavioral of top is
    component encoder
        Port (
              hex_in : in  STD_LOGIC_VECTOR(3 DOWNTO 0);
              a      : out STD_LOGIC;
              b      : out STD_LOGIC;
              c      : out STD_LOGIC;
              d      : out STD_LOGIC;
              e      : out STD_LOGIC;
              f      : out STD_LOGIC;
              g      : out STD_LOGIC
        );
    end component;
begin
```

After we declared the components, we needed to create wires or traces that would connect the components together. From using the 4-bit adder implementation used in the slides we, created signals that represented a `STD_LOGIC_VECTOR` for each input and output.

```
signal a_in, b_in, c_in, d_in, e_in, f_in, g_in : std_logic_vector(3 downto 0);
signal seg_out : std_logic_vector(6 downto 0);
signal dp_out : std_logic;
signal an_out : std_logic_vector(3 downto 0);
```

We then instantiated the parts and connected each encoder to the respective input value in the ssd_muxer. You can see an example here:

```
    REGE1: encoder port map(
        hex_in => sw(15 downto 12),
        a => a_in(3),
        b => b_in(3),
        c => c_in(3),
        d => d_in(3),
        e => e_in(3),
        f => f_in(3),
        g => g_in(3)
    );
```

Aryan Gupta
Marcos Rodrigues
Micaela Coco De Marco

Project 2 Writeup

After this, we came to the test bench. Because the directions in the pdf was not very clear on what we had to do specifically, we ran multitude of trails, hoping to get the same waveform in the pdf. We attempted to change the clock signal but that did not really change anything. We also tried using a counter, similar to project 1 and decided to go through all the values that a 16bit value can hold, but quickly realized that it may not be the correct path. After many failed attempt, the simplest solution worked out and we created a process that assigned the proper vale to sw and waited for 10ns.

```
sw_proc: process
begin
    sw <= x"7b10";
    wait for clk_per;
end process sw_proc;
```

This concluded our first part of the project. We were surprised on how long this took us, especially trying to figure out how to get the test bench to work. The 4-bit encoder in the slides, helped a lot with understanding the syntax of Structural VHDL

Part 2

Part 2 was by far the hardest part of the project, just simply because we had no idea what we had to do. When we read that we had to create an accumulator, we assumed we had to create an adder, which by pressings one of the buttons adds the binary value of the switches to the current output of the display then displays the new number. The other button would clear the display to 0. We also though that we needed to use decimal to output to the display, so we went on a google spree learning BCD (binary coded decimal) and how to implement it. We just gave up on this part and decided that it would be best if we moved to the 3rd part which seemed easier and was a larger part of our grade. 2 days before the project was due, one of our group members was talking about the project and then realized that we were entirely wrong about everything and we quickly raced against the clock to fix the project and get it to work. Instead of making an 'accumulator', we had accumulated the switches, where the 7th switch was -2^7. In essence we were making a 2's complement display. After that part was cleared up, it made coding the project much simpler and didn't take us very long to get it working. The other part that was difficult was the negative sign, we realized that our encoder was not capable of taking any more inputs (`hex_in` was a 4-bit value, and we were using all those bits to encode the number). We then realized that we can reserve one of the displays solely for the negative and manually set that display to a negative if we want.

```
    g_in(2) <= '1';
else -- not negative
    accum <= sw;
    g_in(2) <= '0';
end if;
```

Aryan Gupta
Marcos Rodrigues
Micaela Coco De Marco

Project 2 Writeup

The largest part of VHDL is trying to understand the syntax, and the limits of the programing language.

Part 3

I believe part 3 was by far the easiest part of the project, mostly because after we did part 1, we had a working knowledge of VHDL syntax and what structural VHDL is, and after delving into processes and for loops and if statements from our attempt at a BCD in part 2, we knew most of the syntax this part required. The logic was fairly simple, we originally had the top entity divided into two processes but kept getting weird errors and critical warnings, so we switched to a single process that handled everything. We also learned about variables vs. signals. Variables are used in processes while signals are used in the actual definition of the entity. We created a temp variable to store our next value for the 15th LED then at the end of the process 'copied' the value from the variable to the actual signal.

```
GUESS: process (test)
    variable led_15 : std_logic := '0';
begin
    if (reset = '1') then
        value <= sw;
        led_15 := '0';
    end if;
```

We did have this weird issue where the board would crash and disconnect from the computer, but after testing on a friend's board, we chalked it up to the USB wire and not the code.