Course Name: EMBEDDED SYSTEMS I / III

Course Number and Section: **14:332:493:03**

Year: Spring 2021

Lab Report #: 6

Lab Instructor: Philip Southard

Student Name and RUID: Rishabh Chari 177007380

Date Submitted: 4/29/2021

GitHub Link: https://github.com/rishabhchari/Lab6

Purpose/Objective: <What is the intent of the lab (to design, test or implement a/ an xyz?>
The purpose of this lab is to learn how to interface different Pmods to create a system that takes external input from the RTCC pmod and process it through the FPGA to be able to output the input from the RTCC on the OLED Pmod which is a Pmod different from the RTCC.

System Design:
The system has 3 main components. It has the RTCC controller which interfaces with the RTCC Pmod to read the current time from the Pmod and the controller will output the time read in a 2-digit binary coded decimal format for hours, minutes, and seconds. For hours it is of type std_logic_vector with a width of 5 bits as it only requires digits values from 00 to 12. For minutes and seconds they are both of type std_logic_vector with a width of 7 bits as they require digit values from 00 to 59. The Pmod itself has 2 inout signals called SDA, SCL. These 2 signals must be specified in the constraint file because they are connected to the RTCC Pmod itself. SDA is serial data and SCL is serial clock. An important note about the controller is that it has an active low reset so to have it operate then the reset_n signal must be hooked up to 5V otherwise it will not work. Also, if the set_clk_en signal is equal to ‘1’ then the state machine implmented by the controller will go to state set clock. We do not want to set what the values the clock has in the Pmod instead we want to read the Pmod’s values. Therefore, we need to make sure that the set_clk_en signal is tied to ‘0’ to be able to read the current time.

After grabbing the hours, minutes, and seconds signals for the current time from the RTCC Pmod, we then must transform it to a format that the OLED can display it. To this, there is an entity created called time2vectors. Time2vectors takes 3 inputs which are the hours, minutes, and seconds signals from the RTCC controller. It has 6 outputs which are the 1st and 2nd digit of each time signal. So, for hours, minutes, and seconds each signal is split into 2 respective digits. The signal is converted from a std_logic_vector to an integer using type conversions. When it is an integer we perform the modulo 10 operation which will just give us the smallest digit of the number. For example, by taking 9 mod 10 the operation returns 9. For a double-digit number such as 12, 12 mod 10 will yield 2. So, this operation yields the right most digit of any number we apply to it. We then transform this integer back to an unsigned vector which we then add 48 to it before converting it back to a std_logic_vector. By adding 48 to the integer, it converts to the hexadecimal version of the Ascii character. To give an example, say the integer received from the modulo operation is 0. By adding 48 to 0 we are essentially adding 30 in hexadecimal because 48 in decimal is equivalent to 30 in hex. So the hex version of the number is just 30 because 0 + 30 = 30. The Ascii character related to hex 30 is 0. Therefore, this conversion gives a Ascii characters related to the 10 digits from 0 to 9. To convert the signals to the second digit we can use if statements. We check if the signal is in a certain range and depending on where in the range the number is, a specific Ascii character is put into the output. For example, seconds must be between 00 to 59. Therefore, for each 10s place the second digit must be a number from either 0 to 5. We setup the if statements as follows. If the seconds signal is less than 10, then the output for this digit is 0 because there is no second digit. If the seconds signal is less than 20 but equal to 10 or greater, then the output must be 1, and so forth up to 5. This is true for minutes as well. The second digit for hours can only be 0 or 1 so we only have if statements for if the digit should be 0 or if should be 1.

Now that the signals are converted we can know display them on the OLED display using the OLED controller. The OLED display screen is setup as a 4 by 16 array of 8-bit std_logic_vectors. So, for each element in the array, it holds a 8-bit vector. This is the reason to convert the numbers in the time2vectors entity. If we only passed the signals from RTCC output without splitting the signals into 2 digits each the OLED it would only display on 3 digits of the whole time not what we need which is 6. Also, by not converting the signals into hexadecimal corresponding to the Ascii characters, the display would not display numbers either and instead display separate characters. The controller itself works by running the initialization sequence for the OLED, which is specified in OledInit and for the OledEx entity it will make the current screen what the user specified before updating the Oled display with the actual values. We specified the screen as hours_2nd_digit, hours_1st_digit : minutes_2nd_digit, minutes_1st_digit, seconds_2nd_digit, seconds_1st_digit and for every other element in the array it is X"20" which is the Ascii value for a space, therefore the display will only show the 6 required digits we need. The last thing to be done is to create a top-level design and a proper constraint file. The top-level design will map the 3 components ports to the required inputs and outputs and hold the intermediate signals that the components share. The constraint file will allow a proper system clock as well as map the external pins to the board that the system requires.