**­2) Component Description**

**Phase 3:**

Phase 3 was a continuation of phase 2, where the voltage read from the ADC on the LEDs on the Nexys4 board were converted from the 12-bit number into a 16-segment sliding bar graph on the LEDs. This meant that as the potentiometer affected the voltage read from the ADC, the LEDs would act accordingly by either lighting up one at a time or turning off one at a time.

**Phase 4:**

Phase 4 involved using the Digilent PmodCLS module to display the voltage being read from the ADC. The Pmod CLS module is a liquid-crystal display that communicates with the Nexys4 board using an RS-232 UART method.

**3) Implementation**

**Phase 3:**

In order to convert the 12-bit number into a 16-segment sliding bar graph, the voltages acquired from phase 2 were placed in a range from 0 to 4095 V, with each LED representing a 255V interval. In other words, if the first LED lights up, this means that the voltage range is from 0 to 255V. If the first two LEDs light up, this means that the voltage is between 256 and 510. This process repeats until all 16 LEDs have been lit up, which means the voltage at that point is between 3840 and 4095V.

To properly perform the conversion, the toplevel was modified to include a new file labeled LED\_Slide that would map the voltage received from the 12-bit number from phase 2 to the 16-segment sliding bar graph signal, labeled LED\_Slide.

In order to store the voltage reading received from the ADC, a standard logic vector from 11 DOWNTO 8 (from the 12th LED down to the 9th LED) was created that would check and see how many LEDs had been lit up as a result of the voltage, and would adjust the numbers to fit the 16-segment sliding bar graph. For example, if no LED was turned on between the 12th and 9th LED, this meant that there was no voltage from the ADC, and this meant that the voltage was between 0 and 255V. The following is a table that shows every LED from the 12-bit number being converted to the corresponding 16-segment sliding bar graph, and the voltage range between each LED.

|  |  |  |
| --- | --- | --- |
| Number of LEDs lit up from 12-bit number (Phase 2) | Number of LEDs lit up for sliding bar graph (Phase 3) | Voltage Range  (V) |
| 0 | 1 | 0 to 255 |
| 1 | 2 | 256 to 511 |
| 2 | 3 | 512 to 767 |
| 3 | 4 | 768 to 1023 |
| 4 | 5 | 1024 to 1279 |
| 5 | 6 | 1280 to 1535 |
| 6 | 7 | 1536 to 1791 |
| 7 | 8 | 1792 to 2047 |
| 8 | 9 | 2048 to 2303 |
| 9 | 10 | 2304 to 2559 |
| 10 | 11 | 2560 to 2815 |
| 11 | 12 | 2816 to 3071 |
| 12 | 13 | 3072 to 3327 |
| 13 | 14 | 3328 to 3583 |
| 14 | 15 | 3584 to 3839 |
| 15 | 16 | 3840 to 4095 |

***Table x – Table x shows the conversion from the 12-bit number from phase 2 to the 16-segment LED sliding bar graph for phase 3.***

**Phase 4:**

In order to properly use the LCD on the Nexys4 board, the LCD was connected to the JC terminal, with pins J2 and G6 used as the RX and TX pins respectively.

Next, the toplevel was modified to include the UART signals in order to properly use the LCD display.

From there, the example LCD state machine source file was loaded into the board in order to calibrate the LCD and verify that it was able to display the “VCU RAMS” string.

Once the LCD was calibrated, the LCD state machine source file was modified to display “VIN0 = 3.0V”by finding the corresponding ASCII representation of the letters in the string. The state machine was also modified to make sure that once the end of the string had been reached, it would return to the back to the beginning of the string so that it would update the voltage as it changed.

Finally, the LED\_Slide graph was modified to take the voltage readings from phases 2 and 3 and display them on the LCD as they were changed by the potentiometer. To accomplish this, the maximum voltage read from the ADC was divided by the total number of voltage readings that would be displayed on the LCD. Since the LCD will display the voltage from 0 to 3.3V at every 0.1 interval, the total voltage readings will be 3.3(10) = 33. Therefore, This means that there are 124V per every 0.1 reading on the LCD. The following table displays the voltage at every interval for the LCD.

|  |  |
| --- | --- |
| LCD Voltage Interval (V) | Voltage (V) |
| 0 | 0 |
| 0.1 | 120 |
| 0.2 | 240 |
| 0.3 | 360 |
| 0.4 | 480 |
| 0.5 | 600 |
| 0.6 | 720 |
| 0.7 | 840 |
| 0.8 | 960 |
| 0.9 | 1080 |
| 1.0 | 1200 |
| 1.1 | 1320 |
| 1.2 | 1440 |
| 1.3 | 1560 |
| 1.4 | 1680 |
| 1.5 | 1800 |
| 1.6 | 1920 |
| 1.7 | 2040 |
| 1.8 | 2160 |
| 1.9 | 2280 |
| 2.0 | 2400 |
| 2.1 | 2520 |
| 2.2 | 2640 |
| 2.3 | 2760 |
| 2.4 | 2880 |
| 2.5 | 3000 |
| 2.6 | 3120 |
| 2.7 | 3240 |
| 2.8 | 3360 |
| 2.9 | 3480 |
| 3.0 | 3600 |
| 3.1 | 3720 |
| 3.2 | 3840 |
| 3.3 | 4080 |

***Table x – Table x shows the voltage readings from the ADC converted to the voltages to be displayed on the LCD.***

Once the voltage intervals were determined, then the logic to display the voltage on the LCD was split into two parts: the ones place and the decimals place.

For the ones place, the voltage from the ADC is checked to see if it is within the ranges of: 0 to 1199, 1200 to 2399, and 2400 to 3599V, because, from Table X, it can be seen that: if the voltage is between 0 and 1199 V, the LCD will display a 0 at the ones place; if the voltage is between 1200 and 2399V, the LCD will display a 1 at the ones place; if the voltage is between 2400 and 3599, the LCD will display a 2 at the ones place; if the voltage is between is between 3600 and 4095, the LCD will display a 3 in the ones place.

After determining the voltage range, the ADC voltage reading is subtracted from the minimum number in the voltage range. For example, if the voltage reading was 1500, this means that the voltage range is between 1200 and 2400. The LCD displays a 1, and the 1500 is subtracted from the minimum number, which in this case is 1200. So, it becomes 1500-1200 = 300.

Then, the remaining voltage is checked for the first 9 readings (0 to 0.9) in Table X and will display the corresponding 0.X number according to where it lies in the ranges. Using the same example, since the remainder was 300, the 300 is checked for every voltage range to see where it lies, and using the table, it is found that it is between 240 and 360, which places it at the 0.2 mark. Finally, the number in the decimals place is then displayed onto the LCD, which is this case is a 2, so the final reading on the LCD will be 1.2.

**4) VHDL Code**

**5) Tests**

**Phase 3:**

The main test performed was using the potentiometer to increase/decrease the voltage, and seeing how the LEDs behaved. If they did not light up as a sliding bar graph, then the logic was modified until all 16 LEDs lit up perfectly.

**Phase 4:**

Similar to phase 3, the main test performed was using the potentiometer to increase/decrease the voltage and observing what the LCD displayed. If the voltage did not change, this meant that the state machine needed to be modified to make sure it looped back to the beginning of the string as the voltage changed.

**6) Simulation Waveforms**

**7) Problems Encountered**