**Laboratory 3**

**External Memory Interface**

**Post-Lab**

**Team Number 5**

**Authors**

**Christopher Wilcox**

**James Nystrom**

**Christopher Wilcox**

**James Nystrom**

**LAB 3 – External Memory Interface**

For laboratory 3, we were asked to implement in the FPGA, two 8-bit memory-mapped input ports and two 8-bit memory-mapped output ports. The ports were mapped into the ARM memory space so that they are accessible in external memory region 0. We used minimal decoding as specified as to speed up the decoding logic at the cost of a larger usage of memory addresses.

Our decoding logic involves the P3\_0 pin, Read strobe, Write Strobe, and MS0. We have chosen to ignore the output of the decoder in which the read and write strobes are not simultaneously released or pressed. We are therefore using 4/8 states in which one or the other is selected. We have also accounted for the fact that the decoded output pin of the 3:8 Decoder will be driven low, where the rest will be high…

During the lab, a few problems were encountered. The use of the 3:8 decoder was initially wrong as the outputs were read ‘ABC’, not ‘CBA’ as they are in actuality. This caused errors in the decoding, and caused the board to malfunction. After correcting this mistake, the board snapped into action.

The other issue with the code constructed prior to laboratory was that the handling of the lower bound was incorrect. We are outputting bytes from the lut, and therefore, a negative 1 as represented by a byte (0xFF), is not a negative one when represented by a word (0x000000FF). To handle this case, a compare statement was used to more appropriately take care of this instance.

The last and final issue we had was an uncooperative oscilloscope that refused to connect to the computer. For this reason, we had to migrate to another scope to successfully capture the waveforms as generated by our board.

This lab proved to be educational and gave a first-hand look at how to use the external memory interface. It allowed us to further utilize the functionality of the FPGA, and to use the rotary encoder. This information will prove invaluable on the ECE 315 project.

The laboratory asks that we address the following:

1. **Describe your algorithms for determining the direction of the rotary encoder.**

The rotary encoder is decoded using the previous and current state of A and B. Using an LUT, we are able to determine the direction of the encoder, as well as the case of no movement, or the encoder being spun in a way that jumps a state. Below is a graphic description of what is done.

PREV CURR

00 00 - NOCHANGE - 0

00 01 - CCW --> Move Left +1 Position

00 10 - CW --> Move Right -1 Position

00 11 - ILLEGAL - 0

01 00 - CW --> Move Right

01 01 - NOCHANGE - 0

01 10 - ILLEGAL - 0

01 11 - CCW --> Move Left

10 00 - CCW --> Move Left

10 01 - ILLEGAL - 0

10 10 - NOCHANGE - 0

10 11 - CW --> Move Right

11 00 - ILLEGAL - 0

11 01 - CW --> Move Right

11 10 - CCW --> Move Left

11 11 - NOCHANGE – 0

1. **Describe your algorithm for manipulating the LED segment.**

To manipulate the LED Display Bar, we store a count indicating an index. This index ranges from 0 to 9, representing the segment to be off. We shift the off segment using this count to the position as indicated by the current position of the off segment + the offset as provided by the rotary encoder. We then bounds check if the user manages to leave our range, wrapping them around to the beginning or end, whichever is appropriate for the given case.

1. **Describe what the effects would be if you configured the external memory region for 16-bit operation instead of 8-bit operation. Draw a memory map of the first 16 bytes of the memory region where your ports exist. (You might want to test your answer in the lab.)**

If we configured our external memory region for 16-bit mode, we would have to change our decode logic to handle the wider bus by not using bit 0 for the decode logic. We would also have configured AD8 – AD15 to use the external memory interface. Attached is a drawing showing the first 16 bytes of the memory region where our ports exist. Below is a drawing showing the memory map using 16-bit mode.

**MEMORY MAP DRAWING**

|  |  |
| --- | --- |
| 0 | **Even** Address Shared by **Even** Input/**Even** Output, RS/WS used to determine action |
| 1 | **Even** Address Shared by **Even** Input/**Even** Output, RS/WS used to determine action |
| 2 | **Odd** Address Shared by **Odd** Input/**Odd** Output, RS/WS used to determine action |
| 3 | **Odd** Address Shared by **Odd** Input/**Odd** Output, RS/WS used to determine action |
| 4 | **Even** Address Shared by **Even** Input/**Even** Output, RS/WS used to determine action |
| 5 | **Even** Address Shared by **Even** Input/**Even** Output, RS/WS used to determine action |
| 6 | **Odd** Address Shared by **Odd** Input/**Odd** Output, RS/WS used to determine action |
| 7 | **Odd** Address Shared by **Odd** Input/**Odd** Output, RS/WS used to determine action |
| 8 | **Even** Address Shared by **Even** Input/**Even** Output, RS/WS used to determine action |
| 9 | **Even** Address Shared by **Even** Input/**Even** Output, RS/WS used to determine action |
| 10 | **Odd** Address Shared by **Odd** Input/**Odd** Output, RS/WS used to determine action |
| 11 | **Odd** Address Shared by **Odd** Input/**Odd** Output, RS/WS used to determine action |
| 12 | **Even** Address Shared by **Even** Input/**Even** Output, RS/WS used to determine action |
| 13 | **Even** Address Shared by **Even** Input/**Even** Output, RS/WS used to determine action |
| 14 | **Odd** Address Shared by **Odd** Input/**Odd** Output, RS/WS used to determine action |
| 15 | **Odd** Address Shared by **Odd** Input/**Odd** Output, RS/WS used to determine action |

**We have also attached all requested schematics, waveforms, and code. Below you will find a list of contents.**

1. Oscilloscope trace showing the operation of our decoding logic.
2. Quartus Pin-Out Diagram.
3. main.s
4. exceptions.s
5. IRQ\_Handler.s