1. project\_arithmatic.circ - The arithmatic circuit deals with adding/subtracting two 8bit values (remade after class, as I found we could use the arithmatic logic tools). This circuit also has a decoder to allow me to add the multiplication and subtraction circuits as well. The default input is applied north of the circuit. If it is necessary to subtract the two initial values a side bit applies a multiplexer that inverts each of the bits for one of the values. When the subtraction enabler is active (1) it also adds 1 to the inverted value via the Full\_Adder carry value in the first adder. This successfully applies the “flip the bits and add one” requirements to invert for subtraction. If a final carry is ever present it is discarded.
2. project\_equalto.circ - The equalto circuit relies on sequential XNOR gates that compare two 32bit values in equal measure. If any value in the same position as the corresponding value in the second number does not match the XNOR gate will be set to 0. All the XNOR gates link to a single 32bit AND gate so if any of the previous values come up as 0, the and gate will pass 0 signifying the values don’t match. If all the values match, the XNOR gates will pass 1’s, and all 1’sat the AND gate will pass a 1 for equality verification.
3. proj\_regustry.circ - the registry circuit is a 2\*\*2-by-32bit registry with Write Enabling. Using a two-input decoded, the appropriate address is selected from the four possible locations. When an address is selected, a multiplexer linked to the decoder returns the values saved in the rs-latches at each memory position to the output south of the circuit. Selecting an address, providing an input, then setting the Write Enable bit to 1 will overwrite the data in that location. The same is true if you leave Write Enabled set to 1, then select a location, but this is not preferable as it is easier to accidentally overwrite the wrong data if the intended location is not properly selected. Best practice would be to find the location then flip Write Enabling on then off again.  
     
    Each address location is separated into two rows of 16bits each. The values count up from right to left as this example shows. The output is ordered in a similar fashion for better readability.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ... | 21 | 20 | 19 | 18 | 17 |
| ... | 5 | 4 | 3 | 2 | 1 |

1. Division Circuit Not Completed - I was unable to find an example of how I would set up a division circuit online or in the book. Were I to hazard a custom circuit without studying ideas ahead of time I think there would be WAY too many AND/OR/NOR gates and processing would be limited with greatly extended clock times as the circuit would be so large. Adding to that, I am using 32bits for my example function which would greatly increase the required gates and logic complexity.