Labs 11 and 12 Write up

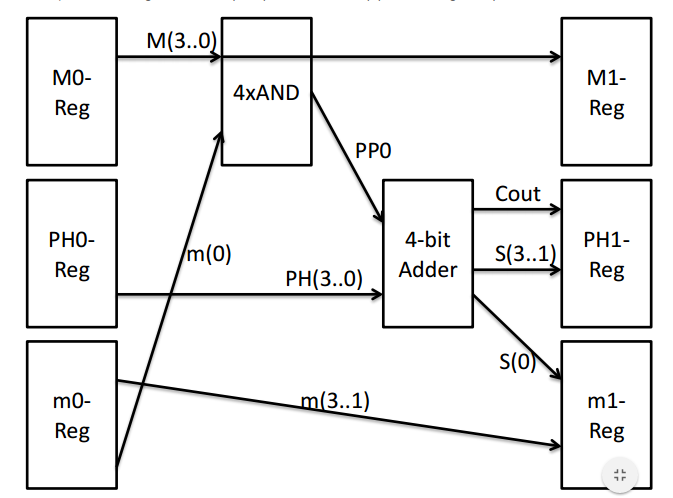
Evan Holmes

Lab 11:

In this lab, the student is to create a multiplier using a structural data path and a sequential control unit to control all of the components. There are three stages, the last two repeat several times (4) until the process is complete. MC0, or the reset stage, resets all components and loads the multiplier and multiplicand registers. MC1, or the start stage, loads the PH register (the modified multiplicand) and loads the D Flip Flop. MC2, or the shift stage, shifts the PH and multiplier registers. The input to the PH is complicated. The multiplicand register is one of two inputs into a MUX, alongside “0000” where the select is the shift out of the multiplier register. The Mux output is then input into an adder, alongside the output of the PH register from the previous clock cycle. The Carry out of the adder is the input to the DFF which is then the Shift in of the PH\_register. After MC1 and MC2 are ran 4 times, the output (combined output of PH register and multiplier register) should be the product of the multiplier and the multiplicand.

Lab 12:

In this lab, the student is to create a 4 bit pipeline multiplier. The pipeline multiplier works in 4 sections. Each sections takes an input of M0 (Multiplicand), PH0 (Register), m0(multiplier). PH0 is usually set to “0000”. The first stage is shown below. In the 4xAND module, all of the Multiplicand bits are ANDed with the least significant bit of the multiplier. The Process repeats with the same design 3 more times. The output[7..4] is the output of the PH4 register and output[3..0] is the multiplier4 register.



Preprocessing and Post processing:

For both labs, there are processing procedures to be done in the code to the multiplier and the multiplicand. My number given to me in class is 13, or “00001101”. According to the chart given to us in the lab 11 description. The multiplicand is 5 bits. For each number, I must OR the multiplier with 0xC “1101” for preprocessing and subtract the product by 0xA “1010” for post processing. For that, I just added the 2’s compliment of 1010. Below is the table of values and how I got them.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Multiplicand | Multiplier | Pre P Mplier | Before Post P | Final Product |  |
| 0000 | 0000 | 1101 | 000000000 | 11110110 |  |
| 1111 | 1111 | 1111 | 11100001 | 011010111 |  |
| 0101 | 0101 | 1101 | 01000001 | 00110111 |  |
| 1010 | 1010 | 1111 | 10010110 | 01000001 |  |
| 0101 | 1010 | 1111 | 01001011 | 01000001 |  |

Lab 11:

Design Choices:

In terms of design, the multiplier is created by changing the output of the PH register and shifting in 0s or 1s. Most of the design is described in the design specifications. However, I did have to experiment a bit to get the products to be correct. Because the machine cycles 1 and 2 are iterated 4 times, a for loop must be created. Initially, I made the for loop in the testbench, but I realized that this would not work when implementing with the FPGA. Therefore, I had to change control unit, and have only one start button that would reset and then do MC1 and 2 four times to get the final product.

Problems Encounters:

I came into many problems with this lab. I found the structure to be straight forward, but spent several hours on the control unit to no avail. I also built the testbench with relatively few problems. I have been very successful in quickly getting rid of compiler error, but logic errors are what is really troubling. I eventually decided to take what I learned and start from scratch. I made slight changes to some of the modules and made a much cleaner, more organized code. Currently, it multiplies but often produces the wrong result, which is a problem.

Unresolved Issues:

Currently, there is some small error which I soon hope to have fixed with the output of the D flip flop for certain values. For some values, the code works and successfully multiplies. But for values in which the Carry out of the Adder is one at any point, there comes small problems and the answers are slightly off. There is also a problem of consistency of when the output is correct. Sometimes the output is correct during the final cycle and sometimes the second to last cycle.

Lab 12:

Design Choices:

For this lab, I only created one register and then I instantiated it 15 times. I tried to sort my code to be as organized as possible. Because each of the four sections are almost identical, I created signals and structural data paths for the first section with “0” at the end of each signal/stage, and then copied and pasted for the next four sections, and changing the number to the respective stage. The first section is slightly different than the others as it takes actual inputs as opposed to other signals, so I had to modify sections 1-3 slightly.

Problems Encountered:

For the most part, I didn’t encounter too many problems. I first created the multiplier without using clocks or registers, which was wrong. but it worked correctly, which was a good start. Then I had to recreate the multiplier from scratch with the registers using a clock. This was significantly more work, and was a much longer code. The first major error I encountered was that I simply was not getting an output. All my signals were working correctly, but no output was ever read. Then I realized that it was simply due to my testbench having too few clock cycles. I added two more clock cycles and my multiplier worked. My final issue was that I had the output set to the multiplicand register and the multiplier register instead of the PH register and the multiplier register, which meant that the four most significant bits of the output were constantly equal to the multiplicand.

