Brandon Rullamas – brullama

Lab section: 4 TTH TA: Shea Ellerson

Due: 4/23/2013

Lab Partner: Trieste Devlin (Fizzy)

Title:

Lab 2: Arithmetic Logic Unit and Memory

Purpose:

The purpose of this lab is to learn more about MultiMedia Logic as well as learning how memory consisting of four registers is capable of storing four bits per register. The lab also shows how arithmetic logic units operate and how they are capable of performing addition, negation, and bitwise AND.

Procedure:

The methodology for constructing the memory and arithmetic logic units followed this pattern: First, the memory unit was constructed by utilizing the pieces already supplied to us and then reproducing what was on the PDF supplied. This was done through a combination of muxes and switches that allowed for different values to be stored into the registers, allowing for the user to get a different value depending on what the switches were set to. The functionality to read a four-bit number from a register as well as write a 4-bit number into a particular register was done during this part through a series of muxes that chose what register to use through the selectors which gave the result of [0,0], [0,1], [1,0], or [1,1], which translated to register 0, 1, 2, or 3 respectively.

The arithmetic logic unit (ALU) then followed, again by utilizing what was already supplied as well as reproducing a lot of what was given in the PDFs. Using the knowledge that was learned in lab 1 with logic gates, the functionality to get the results to make the values add, not, or bitwise and was implemented. The procedure for this is trivial as it just requires a lot of comparison to see if numbers add up to a certain value, etc. The functionality to see if the value was zero, positive, negative, or overflowed was also implemented again through the use of logic gates, most of which was trivial. Muxes were used to choose between three different values (add, not, and) that were already calculated with two switches as selectors in order to give an output of whatever the user wanted. The ‘bad case’ was used as the fourth choice.

In order to combine the memory and the arithmetic logic unit, the two schematics were first copied and pasted onto multiple pages of the same document. A lot of signal senders and receivers were then renamed and reused in order to incorporate the same functionality that the memory provided with the ALU. In particular, register selector muxes were made so that the

ALU could decide between which of the registers to read from and into the ALU. Another set of muxes were created in order to choose between whether the user was trying to read between the ALU or the keypad.

Algorithms and other data:

In order to get the correct results with a lot of the arithmetic (add, not, and), multiple gates were used that utilized AND, OR, XOR, and NOT gates. For example, the bitwise NOT function simply inverted all of the values fed to it. AND used AND gates to see if the inputs shared the same values. ADD used a series of XOR and AND gates that fed into an OR gate in order to add the two values together. Overflow had a NAND gate between two inputs leading to an AND gate with the output of the NAND gate along with the added value. At the same time, an AND gate was comparing the same two values with an inverter on the ADD value and both values leading to an AND gate. The result of the two AND gates then led to an OR gate that led to the overflow result. For the rest of the combination, our knowledge of muxes was simply used that utilized a lot of the values described above along with switches that are set by the user in order to receive the correct value that they are trying to achieve.

What went wrong or what were the challenges?

The biggest problem that was encountered during the lab was the combination of the memory and the arithmetic logic unit, as the two units shared a lot of the same sender receiver and signal names. This led to a lot of renaming and trying to map out where things led to, which was challenging. It then took more work to try and get the two units to integrate with each other, which was just solved through trial and error, mostly. The biggest challenge with this phase was trying to properly select the registers for the ALU to use. This was solved by figuring out that the DR value of the memory did not need to be used, so it was pointed to a null signal receiver.

Other information:

How is the sequential logic useful; i.e. why are the inputs and outputs latched?

The sequential logic is useful because it allows for multiple inputs to be chosen from based on user input and output to a specific value that allows for the rest of the logic to work based on what the user inputs. Without sequential logic, the process to get user input and transfer that to the numbers would be impossible.

What is the clock signal used for?

The clock signal is used as a “replication” for “turning on time” for the circuit so that the values can be processed and transferred to the appropriate registers.

What number representation did you use? And how do you check for a negative number?

The number representation used was two's complement. In order to check for a negative number, the bits were looked at and if the most significant bit was a 1, then the number was a negative number.

What is the maximum and minimum number that your ALU can accept?

The maximum and minimum number that the ALU can accept is –F and F.

Your ALU only accepts 4-bits. What changes do you need so that your ALU can accept 8-bits?

The changes that need to be made so that my ALU can accept 8-bits would be that the number of muxes and sequential logic would need to be increased so that the bits can be chosen between and properly displayed. A bigger input would be needed as well as more selectors for the muxes in order to choose between all of the different values that would come from an 8-bit number instead of a 4-bit number.

Conclusion:

This lab was exceptionally harder than the previous lab and took a lot more time to complete but this lab provided a much better learning experience as much more research was needed in order to figure out what was needed to complete each unit. The lab felt very related to the real world as it gives a basic understanding of how computers operate and how much more complicated they are now than what they were in the basic opening stages of the computer.

Extra:

The TAs (Shea and Daphne) were very helpful in explaining all of the processes need to make the units work and how to utilize MultiMedia Logic for these tasks. They also helped explain a lot of typos and defined a lot of clarity in the lab 2 pdf that was confusing at first. This lab was very difficult but the use of a partner as well as the availability of the TAs made it workable.