CEG2136 Lab 3

Brennan McDonald - *8195614*

09/21/2016

Table of Contents

Theoretical Part

Introduction

The purpose of this lab was to create an Arithmetic Logic Unit (ALU) that would run on the Altera FPGA board. Students were tasked with using the Quartus software to create multiple symbols and use these symbols in the creation of a fully functioning 4 bit ALU. The structure of the ALU was the combination of two main sections. The arithmetic unit and the logic unit. These were built made with the combination of adders, logic cells and basic components (AND, OR, NAND, NOR, XOR). The two of these units were combined with an 8 to 4 multiplexer. The ALU used 16 commands, given in the form of binary to select what unit to use and what function to select from each unit.

Discussion of Problem

The problem presented by this lab was the creation of the ALU which students had to design themselves. This presented a problem due to the complexity and large scale of the circuit. Since there were many small pieces and many depths of circuit diagrams it was easy to get lost attempting to create or debug the circuits. In the prelab it was required to create various sections of the circuit to overcome this problem and familiarize students with the design of the circuit. These diagrams were of the following circuits:

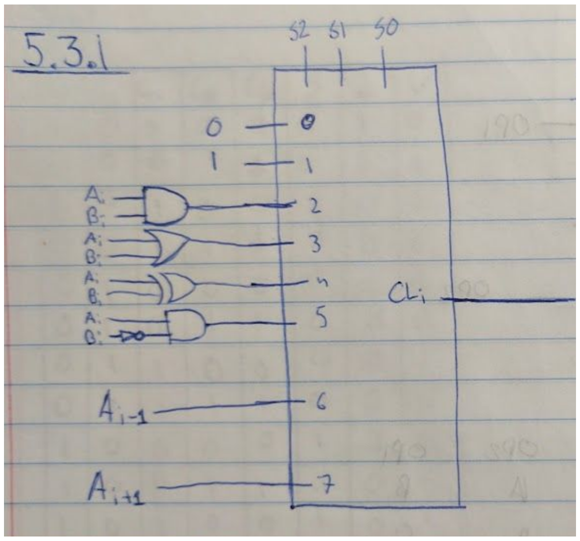

Fig 1. 1 bit logic unit.

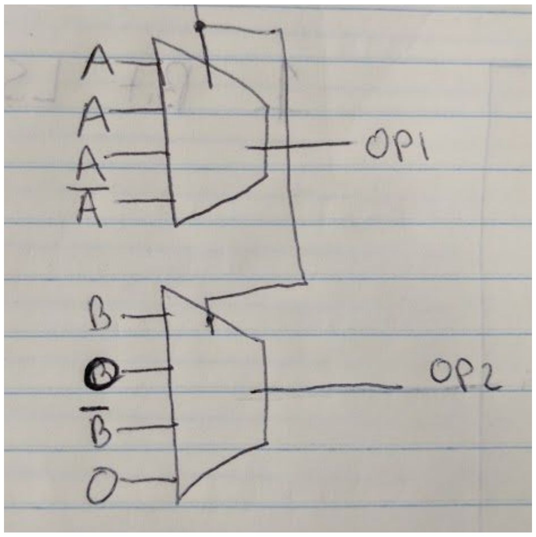

Fig 2. Overview of primary multiplexer

Discussion of Solution

The solution was quite trivial due to the explanation in the lab manual. We had to simply follow the instructions provided and use our pre-lab to create the individual circuits and then combine them in the logical order of an ALU. The circuit was made entirely out of core logic components and multiplexers.

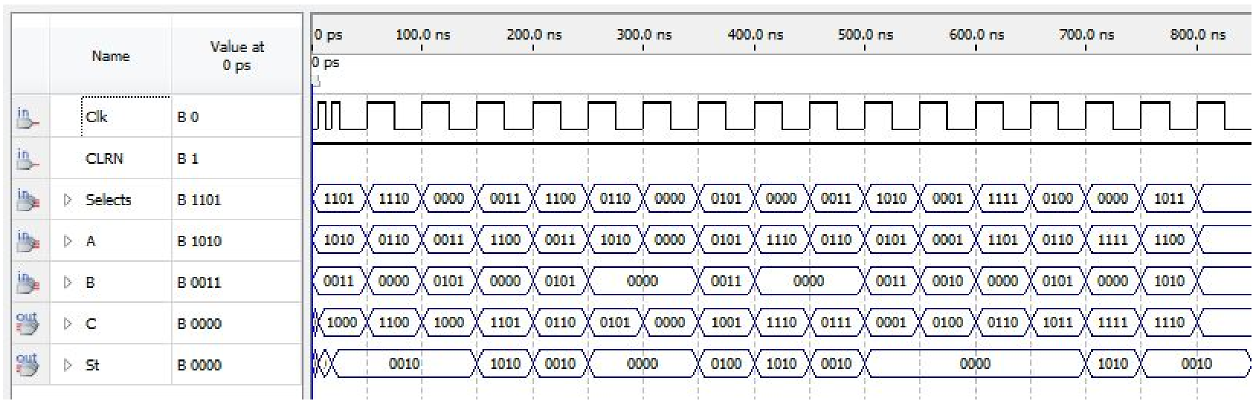

Fig 3. Simulation of the complete circuit. Note that the first few pulses are doubled to initalize registers

Design Part

Components used

At the lowest level components used were logic gates such as AND, OR, NOR, NAND, XOR and NOT gates. At this level we also used ICs such as multiplexers combined with the previous components. At the upper levels we created circuits such as 1 bit full adders and 1 bit logic units. These circuits were combined to support a bus width of 3 bits.

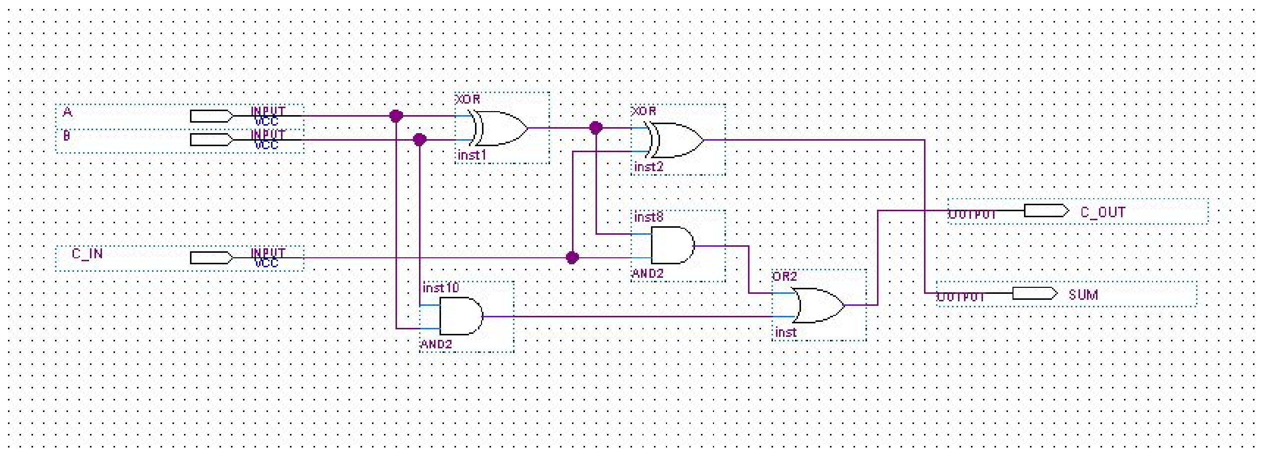

Fig 3. 1 bit full adder circuit block diagram

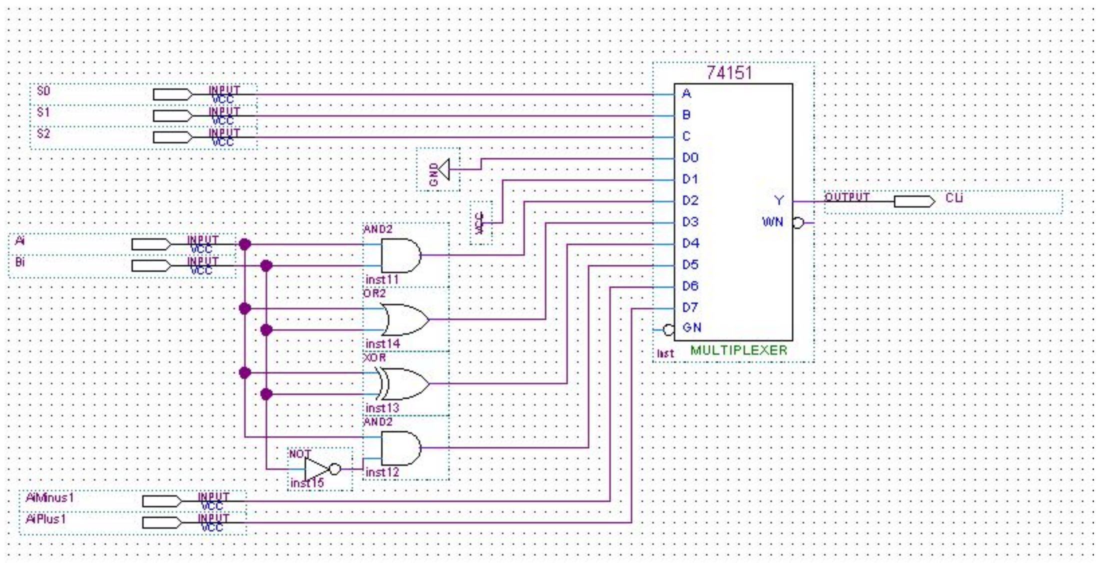

Fig 4. 1 bit logic unit circuit block diagram.

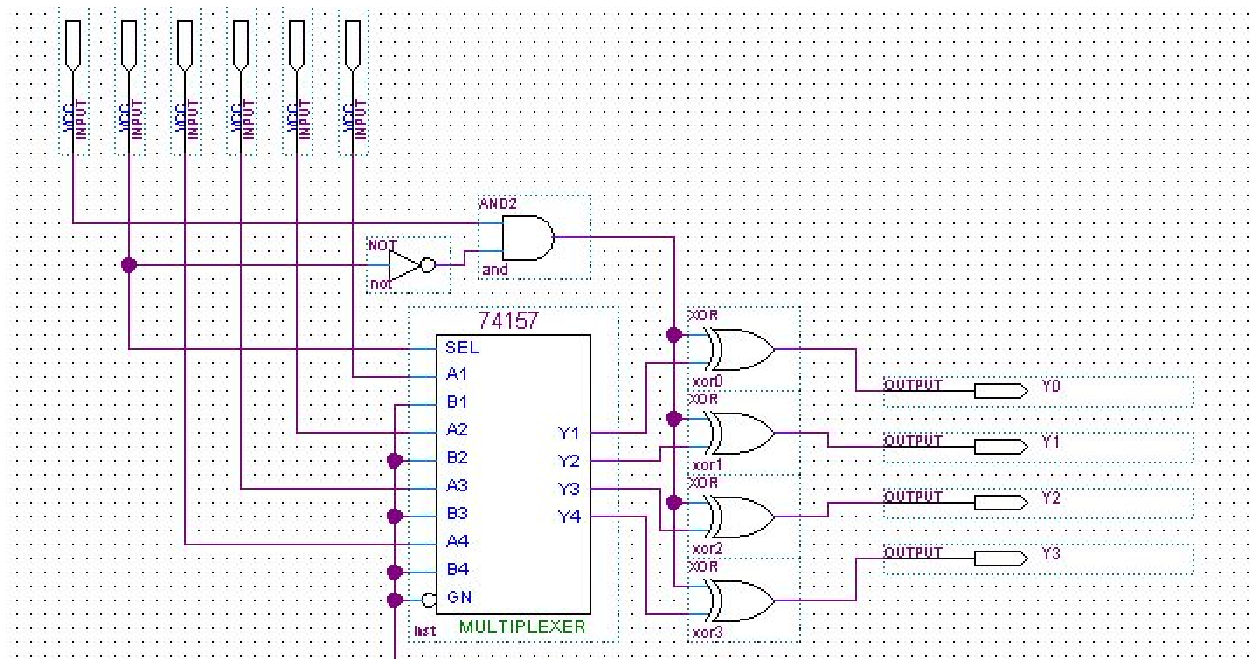

Fig 5. 4 bit arithmetic unit circuit block diagram.

Actual Solution

The solution I used to create the final circuit was a combination of the block diagrams pictured above to create the complete ALU. I started the circuit with a multiplexer that would select between the arithmetic side or the logic side of the circuit. In these sub-circuits the exact command that we wanted to use was selected by the 3 least significant bits of the 4 select bits. At the end of the circuit

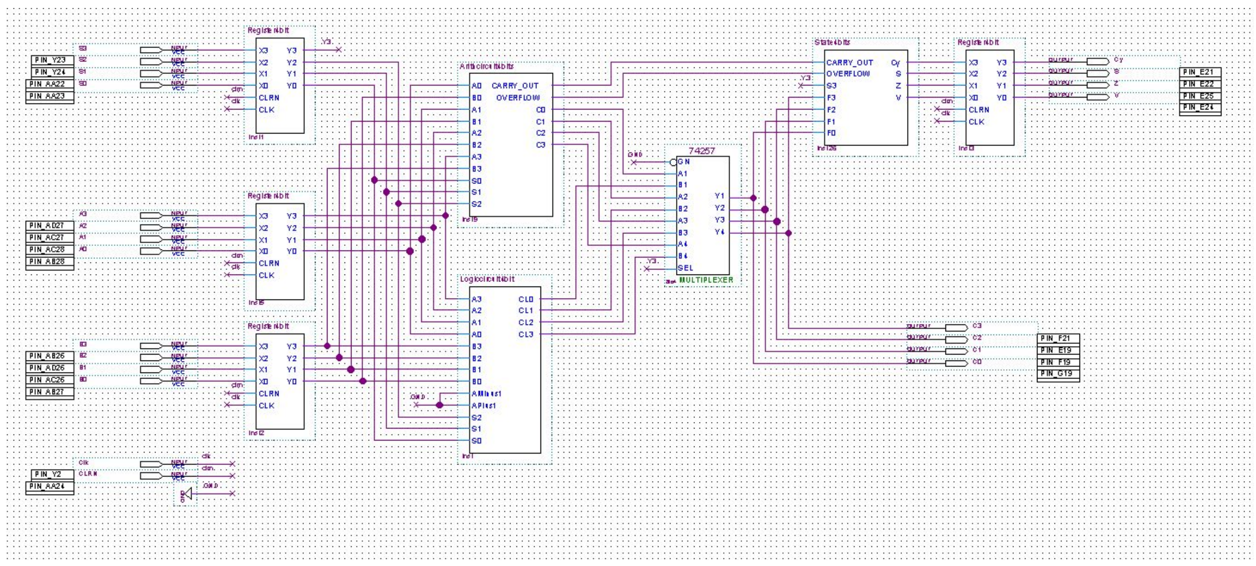

Fig 6. Top down view of the full 3 bit ALU

Tools Used

This lab was completed with minimal tools. These tools included the computer, the Altera board, the Quartus software, and the USB cable.

Discussion

Errors

Errors I encountered in this lab were mostly with bussing and design. Due to the high volume of components and wires, there were a lot of locations in the design that were too dense. I was able to solve most of these errors by not connecting nodes and connecting them via naming the ends. I also ran into errors in the simulation and running it on the board. Mostly these errors were simple to fix due to the straight forward logic of the circuit. The simulation I was supposed to provide was also quite vague and due to this I did not quite know what was required of me.

Discrepancies

Since I planned my circuit before performing the actual implementation I was able to create the circuit to mostly what I had imagined and designed. The only minor discrepancies between the circuit and design was where I used named nodes to connect the components instead of actual whole wires. This was only done to preserve space and had no change on the actual behaviour of the circuit.