PROJECT 1: Part 1

Section 1:

Design a 4-to-1 multiplexer that uses 8-bit buses on the inputs (instead of single bits). The output will be a single 8-bit bus.

Documentation:

A multiplexer is used in many applications as a way to do quick computations and placing the focus on a single computation one at a time. The use of the Multiplexer I had designed (see Fig 1.1) involves a large number of variables and an even larger of inputs. The 8-bit buses along with the expander and bit mergers do help in the organization of this project. The 8 boxes at the bottom represent a different 4-to-1 in which I have designed myself, which has been sub-designed for simplicity of the larger 4-to-1 mux involving 8 bit buses. The 4-to-1 multiplexer (Fig 1.2) that I had designed was selector focused for best efficiency by segmenting. By adding inverters near the start of the selector, it cleaned up the end of inputs into the AND gates as well as changing the inverters earlier do computations a lot quicker then putting them randomly by the AND gates, or even worse, putting them between the AND gates and 4-input OR gate.

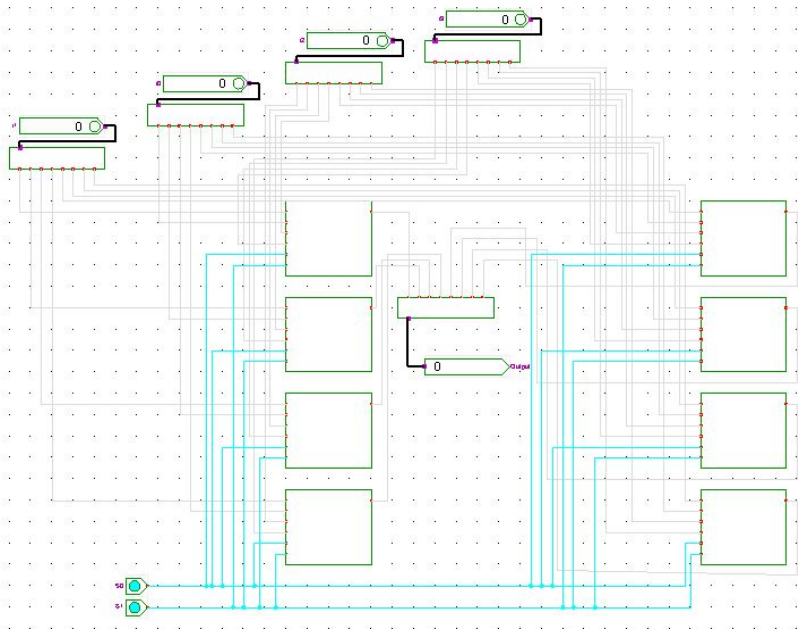

Fig 1.1: 4 to 1 Mux with 8 bit buses and single 8 bit output. Selectors at the bottom, and 8 4to1 muxes run by 8 bit Buses

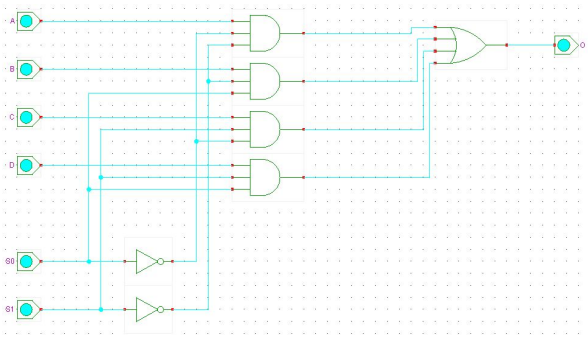

Fig 1.2: 4 to 1 Mux. Notice the inverter at the bottom by the selectors

Testing:

The testing phase was fairly simple, monitoring the behavior of selectors and the output given by the Output Pin Vector. The wiring for the 4 to 1 multiplexer followed the following formula: *F = (A*S1*S0) + (B*S1*~S0) + (C*~S1*S0)*. The was demonstrated in Figure 1.2

For the test shown below a small output should be given due to the equally small inputs added to the various boxes. For this test (Fig 1.3), I did 2 types. The first involved the 8bit buses turned to the inputs as follows: A = 1, B = 4, C = 3, D = 4. The results by selector are as follows: S0 = 4, S0 and S1 = 4, and S1 = 3. Afterwards the 8bit bus inputs were: A = 2, B = 6, C = 6, D = 8. Its outputs were: 6, 8, and 6 (follows same format as previous). Also the wave forms (1.4) preformed perfectly to the selectors involved.

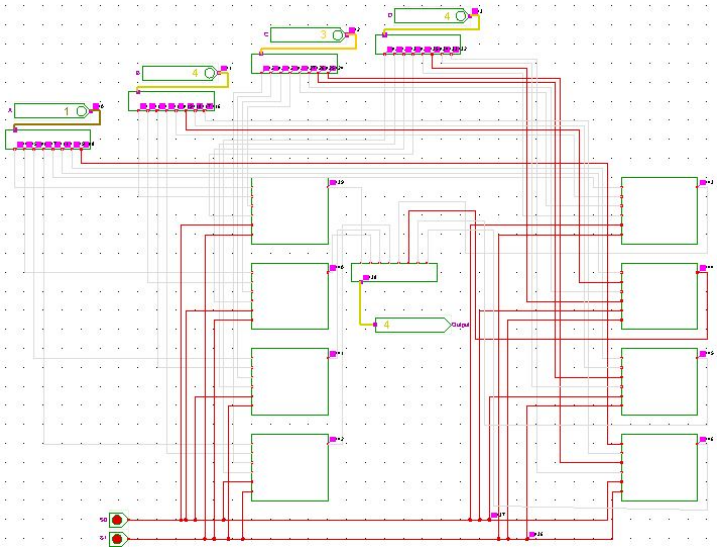

Fig 1.3: The output of the first test with both selectors chosen

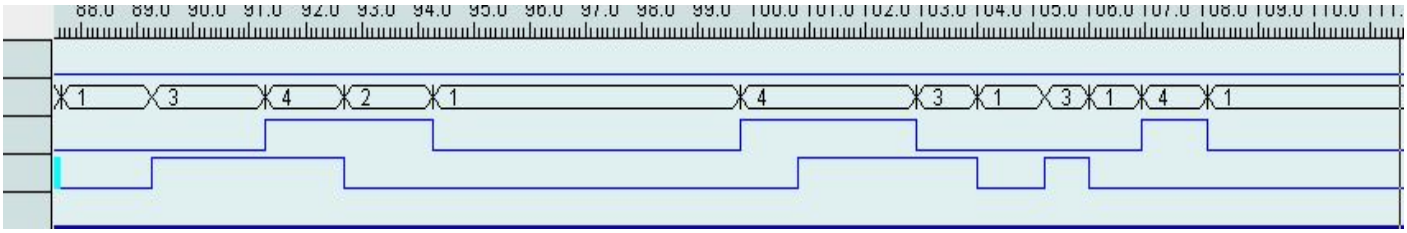

Fig 1.4: Wave forms, Selector 0 on top and selector 1 at the bottom, at the very top are the outputs for each

Section 2:

Build an 8-bit adder. There should be two 8-bit values as input, and a single 8-bit value produced as a result. If you feel there should be additional outputs, please discuss in the report what they are and their intended use.

Documentation:

The 8-bit adder seemed fairly easy at the start but several problems did arose for me. Namely the single 2 bit adder (Fig 2.1). I had used my original adder that I had created simply could not connect with wires, thus I was not able to create a full adder. I did eventually find out my problem, the reason is that Hades simply did not allow for “Y” to be a variable, as a result I changed my variables with in my 2 bit adder to say “Yadder” and for consistency changed “X” to “Xadder”. My wiring also caused problems within organization, while some wires do cross over, my 8 but adder (Fig 2.2) does fully work to my understanding.

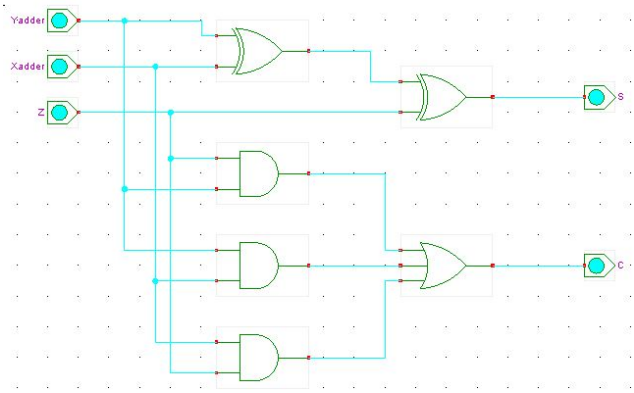

Fig 2.1: 2 bit adder, it is hard to tell but my input names are now longer

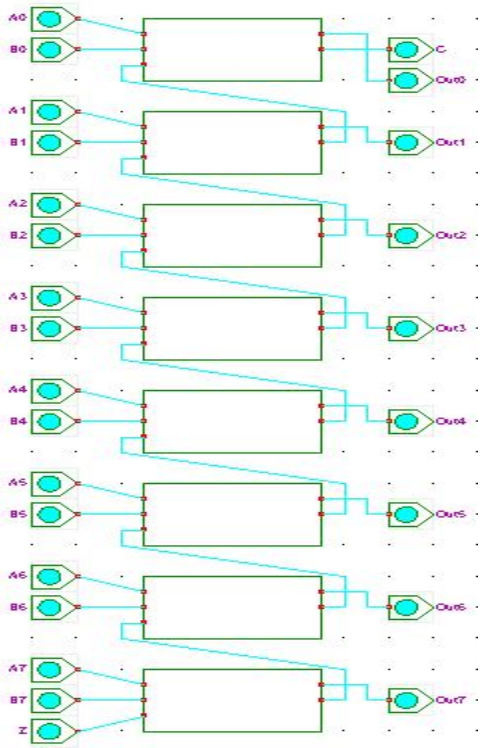

Fig 2.2: the finished 8 bit adder, the boxes represent the 2-bit ones already made

Testing:

Testing involved, the testing of 2 8 bit binary numbers and receiving an output of the correct 8 bit answer. I found this out by looking at the formula for the 2 bit full adder: $S = A \otimes B \otimes C$. The more important part of this formula is the A and B parts as shown in this truth table (source: Wikipedia):

Inputs			Outputs	
A	B	C_i	C_o	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

The testing went as follows I had added two binary numbers. For my adder, due to my

personal placement of inputs, follows the binary from the bottom up when they are entered. My two binary numbers are as follows: 00100100 (36) + 10001001 (137) which should equal 10101101 (173). Sure enough, my answer was proven by the Adder (Fig 2.3). And my wave forms were completed as well.

As a whole I found my project to be a complete success as each test was done efficiently and correctly as per the guidelines were given.

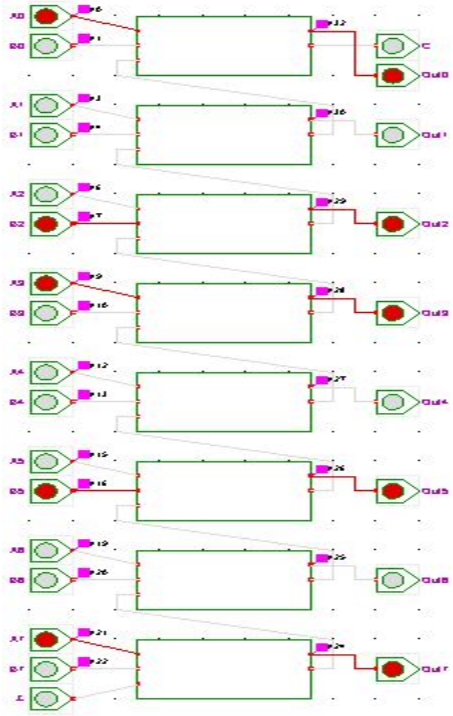

Fig 2.3: The answered adder, the inputs must be followed from the bottom up, while the answer is from top to bottom

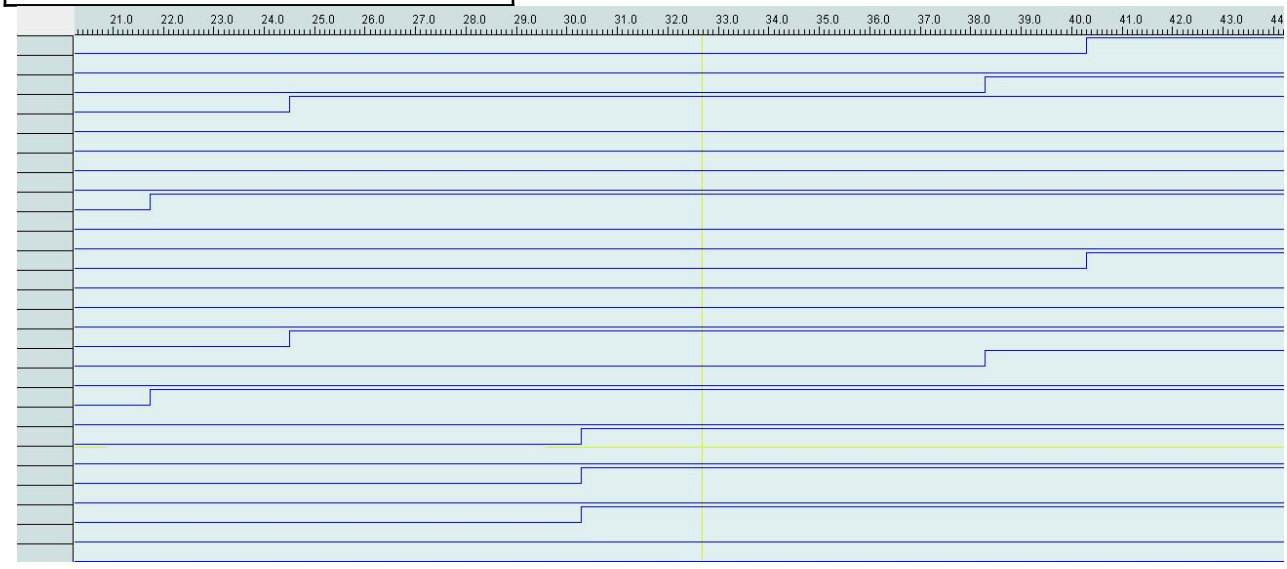

Fig 2.4 The completed wave form, this is only half, the switches (top rows) and the results (bottom rows) are shown