Homework 6: Single Cycle MIPS ALU

For design of my ALU I had a similar idea in the structure as shown in the class overview. I created the simpliest circuits first to generate the overall structure for my ALU. It utilizes a mux to decide between the input bits and which path it will allow through. This meant I just needed to connect all of my 32b logical operations in the correct order to get the desired result from my ALU.

A diagram of a circuit

Description automatically generated

**EXTRA CREDIT:** For my adder and subtractor I decided to take the more efficient route of utilizing a carry-lookahead adder. The difference between this and the standard ripple-carry adder design (which I also made and included) is that it takes the bits from operations and moves them forward to the end of the circuit to “look ahead”. This helps mitigate the delay associated with large ripple-carry additions as they simply connect full adders directly to one another. I figured the logic would be incredibly difficult for all 32b so I utilized class slides and trial-error to get the 4b adders working and strung them together for the full circuit..

Pictured below is a 4b carry-lookahead that I used to make the full 32b carry-lookahead (the logic for a single carry-lookahead unit was too large for one circuit)

A diagram of a computer

Description automatically generatedA computer screen shot of a diagram

Description automatically generated

A diagram of a computer

Description automatically generated

For the shifter design I also referenced class slides to complete my design. The mux design mentioned in class can easily be scaled up to 31b of shifting through use of five rows (just like binary). Using mux signals and input bits you can tell the circuit which rows you want to shift and have the waterfall-effect take place letting you shift any amount between 0-31. For arithmetic shift you just need an extra input such that you can tell when a 1 is in the first spot and alter accordingly. And a convenient fact of this design is that swapping the input bit stream lets you dictate which direction you want the shift to take place.

A computer screen shot of a computer

Description automatically generatedA diagram of a computer

Description automatically generated

**EXTRA CREDIT**: The rest of the circuits took quite a long time to create but are quite simple. It involved mostly just matching bits up between two inputs and making sure that result leaves the circuit in the correct output bit. Below are my XOR, OR, NOR, AND, NOT, 2’s C, and Comparator circuits.

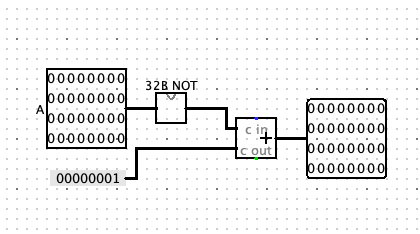
A green lines with white dots

Description automatically generated with medium confidenceA green and white diagram

Description automatically generated with medium confidenceA green lines with arrows

Description automatically generated with medium confidenceA green circuit board with white circles and white dots

Description automatically generated with medium confidenceA diagram of a computer

Description automatically generatedA diagram of a diagram

Description automatically generatedA drawing of a curtain

Description automatically generated

As a whole I have learned a lot about what ALU’s do and how critical it is that they are made correctly. The sheer number of signals they need to interpret and data they work with is incredible. I learned about how each subcircuit involved in an ALU is designed and how it decides which ones to use. The most essential part (in my opinion) is the deciding logic within an ALU. When it’s dissected it is really just a lot of relatively straightforward circuits that are combined into one highly functional unit. Overall I learned a lot about design process, ALU circuits, and the final logic behind ALU’s work.