# EEL 3701 – Digital Logic and Computer Systems Lab 5

# Greg Bolling

# UF ID 64911960

## Problem Statement

The goal of the lab is to design, develop, and implement using VHDL/Quartus, a small GCPU that is a four bit ALU with a controller. The controller is a Moore state machine.

## Design

The design of the ALU is simple and is from the template provided for the class. The data path is also started from the class and expanded to included the muxes and an extra output register at the output of the ALU before going to the Data Out.

Problem 1: ALU

The ALU was designed as:

Problem 2: GCPU Data Path

The GCPU data path was designed as:

Problem 2 Questions:

1. All computations. The computation must operate on the data in the registers and loaded first.

Since the ALU has a 2’s complement, the input can be loaded and inverted so a NOT is possible for a bit if its one bit in a specific location (can’t be all 1’s or all 0’s and must have lowest bit as a 0 to begin but it could do this in the middle two bits of the four bit ALU.) So not all operations would work on all four bits, but there are a number of operations possible. The registers can then be AND’d, OR’d, added (XOR’d). This means that it could do a lot of logical operations such as NAND, NOR, XOR, XNOR, AND, OR all on the middle two bits given the 2’s complement won’t affect those bits if the upper and lower bit are both 0’s. The ALU can Add, Subtract, 2’s complement. Since it can AND and OR and Invert (using middle two bits) it can really do about any function of two bits. It does not have a shift capability so it cannot shift the bits but it does have a carry. Since the operands must all be in the register, it’s hard to force a carry without loading a third value. Since it can add, it could do multiple adds which is kind of like a multiply and could be made to multiply by 2 or 4 or 3 by repeatedly adding the same value around and around the ALU. The ALU could isolate a single bit by loading 0001, 0010, 0100 or 1000 in A and AND’ing it with B, to isolate a given bit in B. Same could be done to remove a single bit from A by AND’ing it with 0111, 1011, 1101 or 1110 to remove a bit from a register. It can test if a value is close to a limit (greater than) by adding another value such as unknown + 4 and if there is a carry out, clearly the value is greater than 3 (so it can compare two values by testing the carry out bit and using the adding capability.) Using this logic, it can test if a value is negative by adding 1000 to the value and seeing if there is a carry out.

1. A single operand instruction, using the Moore state machine, takes two clock cycles. One to go to the instruction state and a second to decide where to store the output. If this were a Mealy state machine, it might be done in one less clock cycle. If you count the idle clock cycle that has a “start” it takes three clock cycles. Depends if this idle clock cycle is part of the total or part of the lost time.

A two operand instruction first must load the first value into the register. Since there is no need to return to the idle state, it could take two clock cycles to perform the load (ignoring the starting idle state) then two more clock cycles to load the second operand into the second register as the ALU must use values from the registers. Then another clock cycle to load the result from the operation into a register (A, B, or send to the output.) So it would take 5 clock cycles to do the two operand operations (ignoring the extra clock cycle lost in the idle waiting state waiting for the “start” command.) If this extra “start” command was counted, it would take six clock cycles.

Problem 3: GCPU Controller

The GCPU Controller was designed as:

The diagram of the controller is:

## Conclusions

Appendix