Rodrigo Becerril Ferreyra
CECS 301 Section 03
Lab 4
Due 23 April 2020

1 Introduction

In this lab, we were tasked with modeling an arithmetic logic unit (ALU) and a register bank or register file. The ALU is a combinational block which is tasked with performing various mathematical operations, such as adding or subtracting two numbers. The ALU is capable of performing both unary and binary operations, and the operation which is selected is chosen using an opcode.

The register bank is a sequential block which contains many ragisters. In this lab, our register bank holds eight 8-bit registers, which is 64 bits of total storage. The register bank only has the ability to write to one register at a time, so both the data to be written and the address of the register must be present. However, two registers can be read asynchronously at a time.

In top-level terms, the design has five inputs—one 16-bit input and four single bit inputs—and four outputs— three single bit outputs and one 32-bit output.

2 Arithmetic Logic Unit

The ALU is similar to a multiplexer in which it has a control path input that controls what happens to the datapath inputs to produce an output; this control input is called the opcode. In this lab, the ALU can perform 8 operations; therefore the opcode must be $\lceil \log_2(8) \rceil = 3$ bits wide. The operations that the ALU can perform on the operands `A` and `B` are denoted in the table below.

Opcode	`Y` = Output
0	$A + B$
1	$A - B$
2	$A << 1$
3	$\{A[0], A[7:1]\}$
4	A AND B
5	A OR B
6	A XOR B
7	$\sim A$

Figure 1: ALU Operand Guide

The ALU also has two other outputs, both of which are one bit wide: the `zero` output is active when `Y` is equal to zero, and the `carry` output is active when the unsigned addition of `A` and `B` results in an overflow. The two operand inputs of the ALU come from the register bank, and the opcode input comes

from the top-level 16-bit output. All outputs of the ALU are sent to the top-level module outputs, and `Y` is sent back to the register file to be written.

3 Register Bank

The register bank is a collection of registers. Because it is a sequential block, a reset is needed to bring all eight registers to a known state; the positive edge of the reset signal sets all registers to zero. The register bank has a write enable input labeled `write`: if `write` is low, then no register will be written on the active edge of the clock; if `write` is high, then on the active edge of the clock, the register whose address is the value of the input `wAdrs` will copy the data on the input `wData`. Since there are 8 possible registers to write to, the input `wAdrs` is $\lceil \log_2(8) \rceil = 3$ bits wide; and since each register is 8 bits wide, the input `wData` is also 8 bits wide.

Reading from the register file is handled asynchronously: the two three-bit wide inputs `rAdrsA` and `rAdrsB` point to the addresses of the registers whose values are outputted on the two eight-bit wide outputs `rDataA` and `rDataB`. These outputs then feed into both the operand inputs of the ALU and part of the 32-bit top-level module output.