CompArch: Lab 1

Zoe Fiddler, Ezra Varady

October 15, 2015

1 ALU Design

For the ALU we opted to implement 32-bit logic rather than a bit slice. While the design is less flexible the implementation is similarly performant, and it allows scaling to widths that are multiples of 32 bits. This required us to implement 32-bit versions of each of the operations we support in the ALU. For the logical operations, `XOR`, `NAND` etc. this was accomplished by attaching 32 instances of the two input version of the gate, connecting the two bits in each place of the operands. The adder we used was a generalized version of the 4-bit adder we wrote previously. Subtraction was implemented as an adder with an inverter on top. SLT is calculated as `XOR` of the most significant bit of the result of subtraction and its overflow. All of these inputs connect to a 3-bit MUX where all eight inputs and the output are 32 bit values. Flags are activated only when the opcode is zero or one, with carryout and overflow being drawn from the proper adder. Zero is `NOR` of all the bits output by the MUX.

2 Test Cases

2.1 Adder Subtractor

We aimed to create test cases for each of the different black box behaviors as well as propagation through the adder. This includes testing proper sum is found in cases with no carryout or overflow, carryout and no overflow, overflow and no carryout and both carryout and overflow. The test bench produced in Modelsim can be seen in Figure 1 on the following page. The same bench was used in subtraction, with the second operand negated. This produced the exact same test bench, seen in Figure 2 on the next page. One problem we debugged using this test bench was that the flags were not being set on subtraction. The error turned out to be due to an indexing error in our code.

2.2 Logic

For the logic outputs we tested each of the different modes `AND`, `NAND`, `OR`, `NOR`, `XOR`, using the same 2 operands. By doing this we could see that each function was behaving properly. No errors were encountered in this process. The test bench can be seen in Figure 3 on the following page

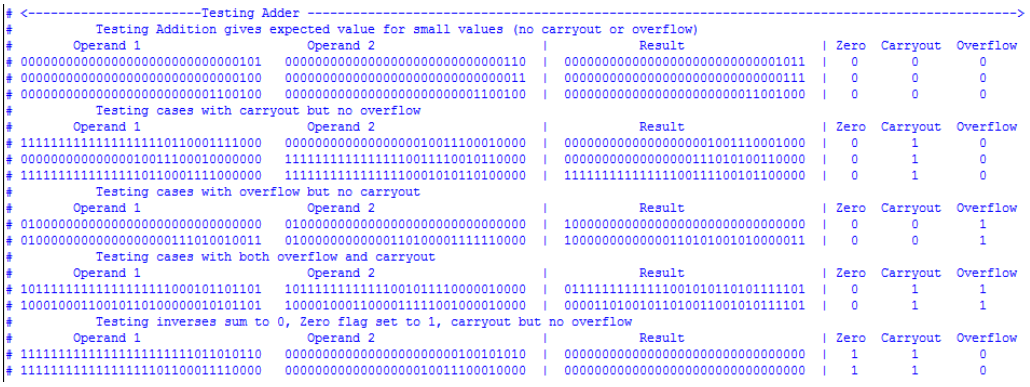

Figure 1: Test bench for 32-bit adder functionality of an ALU

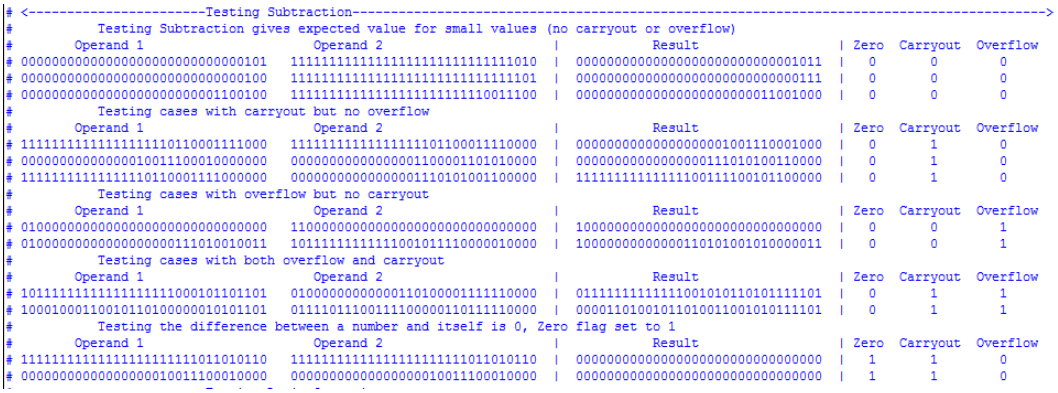

Figure 2: Test bench for 32-bit subtractor functionality of an ALU

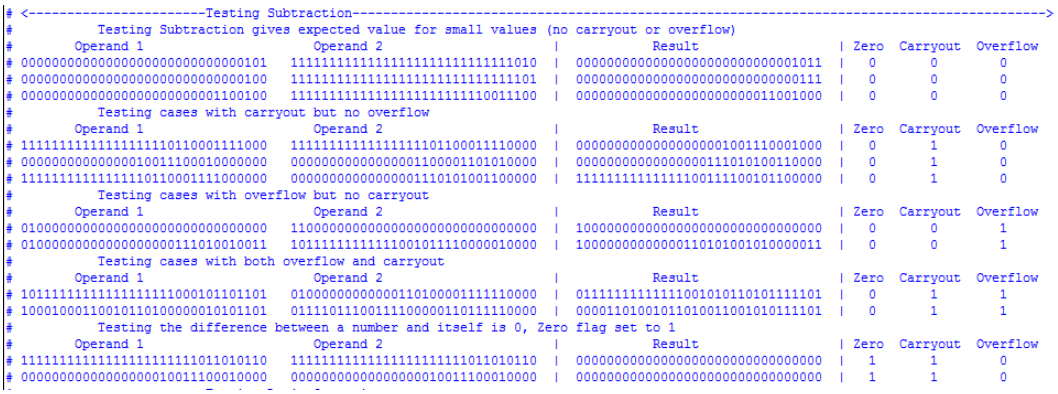

Figure 3: Test bench for 32-bit logic functionalities of an ALU

2.3 SLT

For the set less than function we tested a variety of cases with both positive and negative numbers, which can be seen in Figure 4. In all of these cases, the flags should be set to 0 at all times. Using this test bench we discovered a wiring problem which caused an error in the lsb of the output.

```
# <----------------------Testing SLT ------------------------------------------------------------------------------------------>
#        Operand 1                       Operand 2                       |            Result                  | Expected | Zero  Carryout  Overflow
# 00000000000000000000000000000101   00000000000000000000000000001010  |  00000000000000000000000000000001  |    1     |  0       0        0
# 00000000000000000000000000001010   00000000000000000000000000000101  |  00000000000000000000000000000000  |    0     |  0       0        0
# 00000000000000000000000000000101   00000000000000000000000000000101  |  00000000000000000000000000000000  |    0     |  0       0        0
# 11111111111111111111111111111011   11111111111111111111111111110110  |  00000000000000000000000000000000  |    0     |  0       0        0
# 11111111111111111111111111110110   11111111111111111111111111111011  |  00000000000000000000000000000001  |    1     |  0       0        0
```

Figure 4: Test bench for 32-bit SLT functionality of ALU

Table 1: Worst case propagation delays

Operation	Propagation Delay
Logical operations	160ns
Add	1820
Sub	1830
SLT	1470

3 Work Plan Reflection

We followed our work plan exceedingly well. The day before the lab, we sat down and did most of it. We then, as planned, smoked several (3-7) cigarettes. After a brief nap, as the kids are calling it, we wrote this report. Some may attribute our success to the nonexistance of a work plan before the writing of this report, however we *vehemently* deny that this was the case.