CompArch Lab 1

William Lu, Kathryn Hite, & Ian Hill

October 6, 2016

1 Introduction

We've implemented and tested the arithmetic logic unit subsystem for our new OCA (Olin Computer Architecture) processor.

1.1 Circuit Diagram

The final circuit in figure 1 shows each of our final logic modules along with the look up table feeding into the six input multiplexer, giving us the final result of each chosen operation.

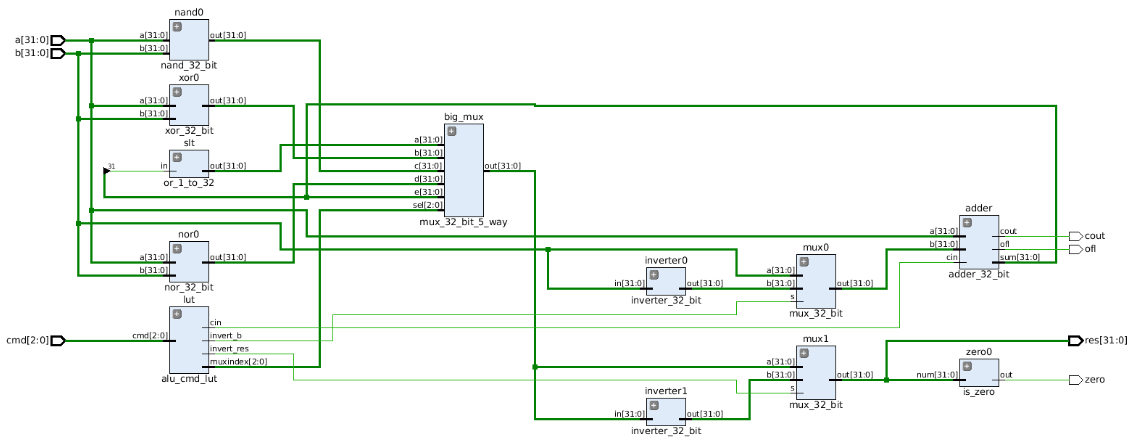

Figure 1: The above schematic is version of our full circuit diagram collapsed for readability.

The full, expanded diagram in figure 2 on the next page, while unreadable, is an interesting illustration of the large number of gates constructing the full modules for each logic and arithmetic operation.

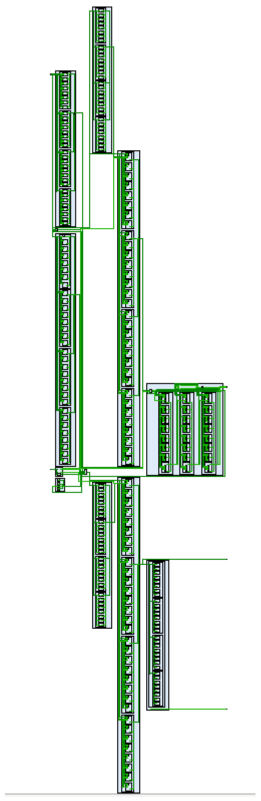

Figure 2: The above schematic shows the full, expanded curcuit diagram for our ALU.

2 Implementation

The implementation of the ALU operation modules structurally followed a bit-slice pattern, but for the sake of code readability and organization, we organized some of the modules into a 1-bit, 8-bit, and 32-bit version.

2.1 Command Look Up Table

Per Professor Hill's recommendation, we constructed a LUT in behavior Verilog using a simple switch statement. The LUT set our control signals *muxindex*, *invert_b*, *invert_res*, and *cin*. *muxindex* is a 3 bit unsigned integer which controls the primary mux which selects the result of the ALU based on the requested operation. *invert_b* is a 1 bit flag which changes whether the ALU inverts input *b* before passing it to the adder. *invert_res* is a 1 bit flag which changes whether the ALU inverts the result of the ALU after it's been selected by the primary mux in order to enable the AND and OR operation which are inversions of NAND and NOR respectively. *cin* is the carry in of the adder which is cleverly set to 1 to enable subtraction after input *b* has been inverted.

2.2 Inverter

Our 32 bit inverter (much like our other bitwise operators) was implemented as 32 figure 3 NOT gates in parallel.

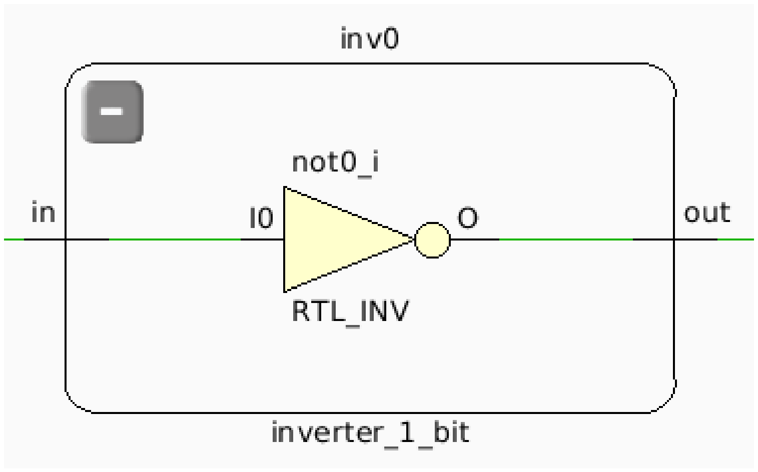

Figure 3: The 32-bit inverter is comprised of 32 parallel NOT gates.

2.3 Adder

The adder was implemented in a bit-slice fashion based on the full adder from our previous lab. These were strung together to form the byte adder shown in figure 4, and four byte adders as seen in figure 5 on the following page were connected to form the 32 bit adder. The full 32 bit adder holds the overflow logic for the adder array.

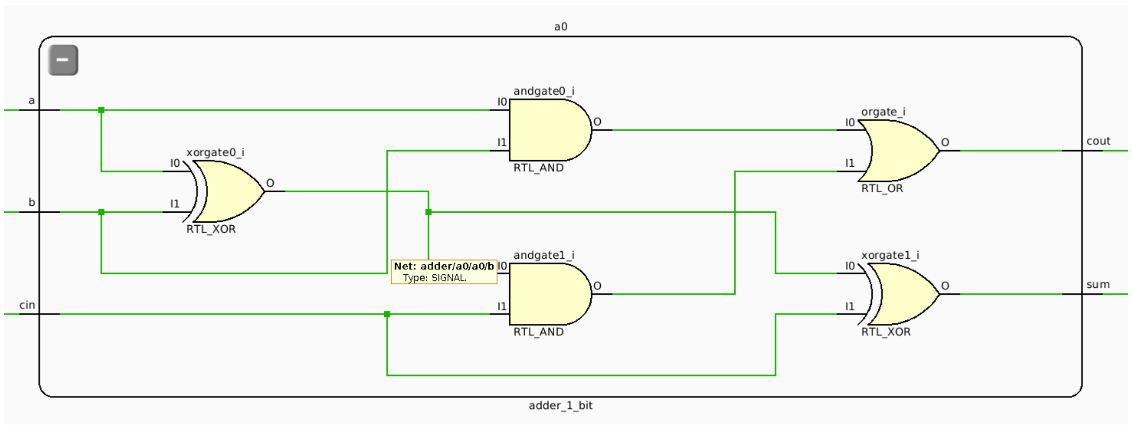

Figure 4: The building block of the full adder implementation is the 1-bit adder module, comprised of five individual gates to compute the sum and carryout.

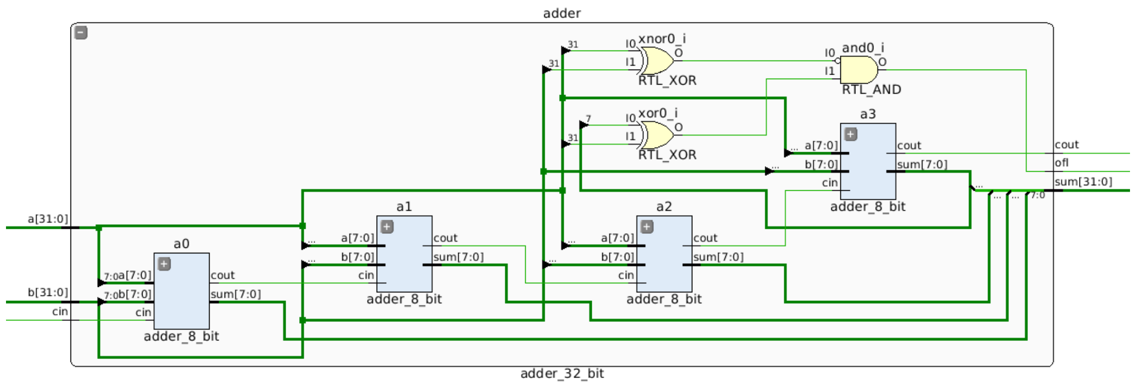

Figure 5: Four of the 1-bit adders are joined to create 4-bit adders. These in turn are strung together to form the final 32-bit adder module.

2.4 Subtractor

Subtraction is performed by converting input b to its opposite and performing normal addition with the adder. To convert input b to its opposite in 2's complement, the control LUT sets the *invert_b* flag to invert input b and sets the *cin* flag to feed the adder a carry in of 1.

2.5 Set Less Than

The set less than module takes in two 32-bit strings, A and B, and returns a 32-bit string containing all zeros other than the least significant bit. This final bit toggles to 1 if A is less than B and is zero otherwise.

The state of the final bit is computed by first inverting B. Once B is inverted, we add A and B together. If A is less than B, the resulting number will be negative as B was inverted. We can thus take the most significant bit of the adder result as the least significant bit of the slt result. In this case, the less value stored in the least significant bit of the result will be 1 to denote that $A < B$.

2.6 Bitwise AND, NAND, NOR, and OR

The bitwise AND, NAND, NOR, and OR modules were all built up starting with a single bit version using the defined structural verilog gates given custom delays. Single bit versions were utilized to build up an 8-bit verison of each gate bitwise. Four of each of the 8-bit AND, NAND, NOR, and OR modules built up the 32-bit modules using the same method.

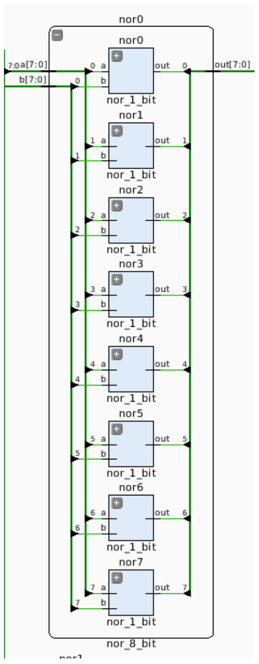

Figure 6: The 32-bit NOR gate is implemented using four 8-bit NOR modules.

3 Tests and Test Results

```
--------------------------------------------------------------------------------------
TESTS FOR AND
--------------------------------------------------------------------------------------
AND   | A: 00000000 00000000 00000000 00000000  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000000
  EXP | R: 00000000 00000000 00000000 00000000
AND   | A: 00000000 00000000 00000000 00000000  B: 01111111 11111111 11111111 11111111
  ACT | R: 00000000 00000000 00000000 00000000
  EXP | R: 00000000 00000000 00000000 00000000
AND   | A: 01111111 11111111 11111111 11111111  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000000
  EXP | R: 00000000 00000000 00000000 00000000
AND   | A: 01111111 11111111 11111111 11111111  B: 01111111 11111111 11111111 11111111
  ACT | R: 01111111 11111111 11111111 11111111
  EXP | R: 01111111 11111111 11111111 11111111
AND   | A: 01111111 11111111 11111111 11111111  B: 00000000 10000010 10110110 00110101
  ACT | R: 00000000 10000010 10110110 00110101
  EXP | R: 00000000 10000010 10110110 00110101
AND   | A: 00000000 00000000 00000000 00000000  B: 00000000 10000010 10110110 00110101
  ACT | R: 00000000 00000000 00000000 00000000
  EXP | R: 00000000 00000000 00000000 00000000
AND   | A: 00000000 10000010 10110110 00110101  B: 00000000 00000000 00000000 00000000
```

```
  ACT | R: 00000000 00000000 00000000 00000000
  EXP | R: 00000000 00000000 00000000 00000000
AND   | A: 00000000 10000010 10110110 00110101  B: 00000000 10000010 10110110 00110101
  ACT | R: 00000000 10000010 10110110 00110101
  EXP | R: 00000000 10000010 10110110 00110101
-------------------------------------------------------------------------------------
TESTS FOR NAND
-------------------------------------------------------------------------------------
NAND  | A: 00000000 00000000 00000000 00000000  B: 00000000 00000000 00000000 00000000
  ACT | R: 11111111 11111111 11111111 11111111
  EXP | R: 11111111 11111111 11111111 11111111
NAND  | A: 00000000 00000000 00000000 00000000  B: 01111111 11111111 11111111 11111111
  ACT | R: 11111111 11111111 11111111 11111111
  EXP | R: 11111111 11111111 11111111 11111111
NAND  | A: 01111111 11111111 11111111 11111111  B: 00000000 00000000 00000000 00000000
  ACT | R: 11111111 11111111 11111111 11111111
  EXP | R: 11111111 11111111 11111111 11111111
NAND  | A: 01111111 11111111 11111111 11111111  B: 01111111 11111111 11111111 11111111
  ACT | R: 10000000 00000000 00000000 00000000
  EXP | R: 10000000 00000000 00000000 00000000
NAND  | A: 01111111 11111111 11111111 11111111  B: 00000000 10000010 10110110 00110101
  ACT | R: 11111111 01111101 01001001 11001010
  EXP | R: 11111111 01111101 01001001 11001010
NAND  | A: 00000000 00000000 00000000 00000000  B: 00000000 10000010 10110110 00110101
  ACT | R: 11111111 11111111 11111111 11111111
  EXP | R: 11111111 11111111 11111111 11111111
NAND  | A: 00000000 10000010 10110110 00110101  B: 00000000 00000000 00000000 00000000
  ACT | R: 11111111 11111111 11111111 11111111
  EXP | R: 11111111 11111111 11111111 11111111
NAND  | A: 00000000 10000010 10110110 00110101  B: 00000000 10000010 10110110 00110101
  ACT | R: 11111111 01111101 01001001 11001010
  EXP | R: 11111111 01111101 01001001 11001010
-------------------------------------------------------------------------------------
TESTS FOR OR
-------------------------------------------------------------------------------------
OR    | A: 00000000 00000000 00000000 00000000  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000000
  EXP | R: 00000000 00000000 00000000 00000000
OR    | A: 00000000 00000000 00000000 00000000  B: 01111111 11111111 11111111 11111111
  ACT | R: 01111111 11111111 11111111 11111111
  EXP | R: 01111111 11111111 11111111 11111111
OR    | A: 01111111 11111111 11111111 11111111  B: 00000000 00000000 00000000 00000000
  ACT | R: 01111111 11111111 11111111 11111111
  EXP | R: 01111111 11111111 11111111 11111111
OR    | A: 01111111 11111111 11111111 11111111  B: 01111111 11111111 11111111 11111111
```

```
  ACT | R: 01111111 11111111 11111111 11111111
  EXP | R: 01111111 11111111 11111111 11111111
OR    | A: 01111111 11111111 11111111 11111111  B: 00000000 10000010 10110110 00110101
  ACT | R: 01111111 11111111 11111111 11111111
  EXP | R: 01111111 11111111 11111111 11111111
OR    | A: 00000000 00000000 00000000 00000000  B: 00000000 10000010 10110110 00110101
  ACT | R: 00000000 10000010 10110110 00110101
  EXP | R: 00000000 10000010 10110110 00110101
OR    | A: 00000000 10000010 10110110 00110101  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 10000010 10110110 00110101
  EXP | R: 00000000 10000010 10110110 00110101
OR    | A: 00000000 10000010 10110110 00110101  B: 00000000 10000010 10110110 00110101
  ACT | R: 00000000 10000010 10110110 00110101
  EXP | R: 00000000 10000010 10110110 00110101
-------------------------------------------------------------------------------------
TESTS FOR NOR
-------------------------------------------------------------------------------------
NOR   | A: 00000000 00000000 00000000 00000000  B: 00000000 00000000 00000000 00000000
  ACT | R: 11111111 11111111 11111111 11111111
  EXP | R: 11111111 11111111 11111111 11111111
NOR   | A: 00000000 00000000 00000000 00000000  B: 01111111 11111111 11111111 11111111
  ACT | R: 10000000 00000000 00000000 00000000
  EXP | R: 10000000 00000000 00000000 00000000
NOR   | A: 01111111 11111111 11111111 11111111  B: 00000000 00000000 00000000 00000000
  ACT | R: 10000000 00000000 00000000 00000000
  EXP | R: 10000000 00000000 00000000 00000000
NOR   | A: 01111111 11111111 11111111 11111111  B: 01111111 11111111 11111111 11111111
  ACT | R: 10000000 00000000 00000000 00000000
  EXP | R: 10000000 00000000 00000000 00000000
NOR   | A: 01111111 11111111 11111111 11111111  B: 00000000 10000010 10110110 00110101
  ACT | R: 10000000 00000000 00000000 00000000
  EXP | R: 10000000 00000000 00000000 00000000
NOR   | A: 00000000 00000000 00000000 00000000  B: 00000000 10000010 10110110 00110101
  ACT | R: 11111111 01111101 01001001 11001010
  EXP | R: 11111111 01111101 01001001 11001010
NOR   | A: 00000000 10000010 10110110 00110101  B: 00000000 00000000 00000000 00000000
  ACT | R: 11111111 01111101 01001001 11001010
  EXP | R: 11111111 01111101 01001001 11001010
NOR   | A: 00000000 10000010 10110110 00110101  B: 00000000 10000010 10110110 00110101
  ACT | R: 11111111 01111101 01001001 11001010
  EXP | R: 11111111 01111101 01001001 11001010
-------------------------------------------------------------------------------------
TESTS FOR XOR
-------------------------------------------------------------------------------------
XOR   | A: 00000000 00000000 00000000 00000000  B: 00000000 00000000 00000000 00000000
```

```
  ACT | R: 00000000 00000000 00000000 00000000
  EXP | R: 00000000 00000000 00000000 00000000
XOR   | A: 00000000 00000000 00000000 00000000  B: 01111111 11111111 11111111 11111111
  ACT | R: 01111111 11111111 11111111 11111111
  EXP | R: 01111111 11111111 11111111 11111111
XOR   | A: 01111111 11111111 11111111 11111111  B: 00000000 00000000 00000000 00000000
  ACT | R: 01111111 11111111 11111111 11111111
  EXP | R: 01111111 11111111 11111111 11111111
XOR   | A: 01111111 11111111 11111111 11111111  B: 01111111 11111111 11111111 11111111
  ACT | R: 00000000 00000000 00000000 00000000
  EXP | R: 00000000 00000000 00000000 00000000
XOR   | A: 01111111 11111111 11111111 11111111  B: 00000000 10000010 10110110 00110101
  ACT | R: 01111111 01111101 01001001 11001010
  EXP | R: 01111111 01111101 01001001 11001010
XOR   | A: 00000000 00000000 00000000 00000000  B: 00000000 10000010 10110110 00110101
  ACT | R: 00000000 10000010 10110110 00110101
  EXP | R: 00000000 10000010 10110110 00110101
XOR   | A: 00000000 10000010 10110110 00110101  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 10000010 10110110 00110101
  EXP | R: 00000000 10000010 10110110 00110101
XOR   | A: 00000000 10000010 10110110 00110101  B: 00000000 10000010 10110110 00110101
  ACT | R: 00000000 00000000 00000000 00000000
  EXP | R: 00000000 00000000 00000000 00000000
--------------------------------------------------------------------------------------
TESTS FOR ADDER
--------------------------------------------------------------------------------------
ADD   | A: 00000000 00000000 00000000 00000000  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000000  C: 0  O: 0  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 0  O: 0  Z: 1
ADD   | A: 00000000 00000000 00000000 00000001  B: 00000000 00000000 00000000 00000001
  ACT | R: 00000000 00000000 00000000 00000010  C: 0  O: 0  Z: 0
  EXP | R: 00000000 00000000 00000000 00000010  C: 0  O: 0  Z: 0
ADD   | A: 00111111 11111111 11111111 11111111  B: 01000000 00000000 00000000 00000000
  ACT | R: 01111111 11111111 11111111 11111111  C: 0  O: 0  Z: 0
  EXP | R: 01111111 11111111 11111111 11111111  C: 0  O: 0  Z: 0
ADD   | A: 10011001 10011001 10011001 10011001  B: 00010001 00010001 00010001 00010001
  ACT | R: 10101010 10101010 10101010 10101010  C: 0  O: 0  Z: 0
  EXP | R: 10101010 10101010 10101010 10101010  C: 0  O: 0  Z: 0
ADD   | A: 11000000 00000000 00000000 00000000  B: 11000000 00000000 00000000 00000000
  ACT | R: 10000000 00000000 00000000 00000000  C: 1  O: 0  Z: 0
  EXP | R: 10000000 00000000 00000000 00000000  C: 1  O: 0  Z: 0
ADD   | A: 11111111 11111111 11111111 11111111  B: 11111111 11111111 11111111 11111111
  ACT | R: 11111111 11111111 11111111 11111110  C: 1  O: 0  Z: 0
  EXP | R: 11111111 11111111 11111111 11111110  C: 1  O: 0  Z: 0
ADD   | A: 01100110 01100110 01100110 01100110  B: 11111111 11111111 11111111 11111111
```

```
  ACT | R: 01100110 01100110 01100110 01100101  C: 1  O: 0  Z: 0
  EXP | R: 01100110 01100110 01100110 01100101  C: 1  O: 0  Z: 0
ADD   | A: 01000000 00000000 00000000 00000000  B: 11000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
ADD   | A: 01000000 00000000 00000000 00000000  B: 01000000 00000000 00000000 00000000
  ACT | R: 10000000 00000000 00000000 00000000  C: 0  O: 1  Z: 0
  EXP | R: 10000000 00000000 00000000 00000000  C: 0  O: 1  Z: 0
ADD   | A: 01111111 11111111 11111111 11111111  B: 01111111 11111111 11111111 11111111
  ACT | R: 11111111 11111111 11111111 11111110  C: 0  O: 1  Z: 0
  EXP | R: 11111111 11111111 11111111 11111110  C: 0  O: 1  Z: 0
ADD   | A: 01100110 01100110 01100110 01100110  B: 01111111 11111111 11111111 11111111
  ACT | R: 11100110 01100110 01100110 01100101  C: 0  O: 1  Z: 0
  EXP | R: 11100110 01100110 01100110 01100101  C: 0  O: 1  Z: 0
ADD   | A: 01010101 01010101 01010101 01010101  B: 01100110 01100110 01100110 01100110
  ACT | R: 10111011 10111011 10111011 10111011  C: 0  O: 1  Z: 0
  EXP | R: 10111011 10111011 10111011 10111011  C: 0  O: 1  Z: 0
ADD   | A: 10000000 00000000 00000000 00000000  B: 10000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000000  C: 1  O: 1  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 1  O: 1  Z: 1
ADD   | A: 10000000 00000000 00000000 00000000  B: 11111111 11111111 11111111 11111111
  ACT | R: 01111111 11111111 11111111 11111111  C: 1  O: 1  Z: 0
  EXP | R: 01111111 11111111 11111111 11111111  C: 1  O: 1  Z: 0
ADD   | A: 10101010 10101010 10101010 10101010  B: 10011001 10011001 10011001 10011001
  ACT | R: 01000100 01000100 01000100 01000011  C: 1  O: 1  Z: 0
  EXP | R: 01000100 01000100 01000100 01000011  C: 1  O: 1  Z: 0
ADD   | A: 10111011 10111011 10111011 10111011  B: 10101010 10101010 10101010 10101010
  ACT | R: 01100110 01100110 01100110 01100101  C: 1  O: 1  Z: 0
  EXP | R: 01100110 01100110 01100110 01100101  C: 1  O: 1  Z: 0
--------------------------------------------------------------------------------------
TESTS FOR SUBTRACTOR
--------------------------------------------------------------------------------------
SUB   | A: 00000000 00000000 00000000 00000000  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
SUB   | A: 00000000 00000000 00000000 00000001  B: 00000000 00000000 00000000 00000001
  ACT | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
SUB   | A: 01111111 11111111 11111111 11111111  B: 00000000 00000000 00000000 00000001
  ACT | R: 01111111 11111111 11111111 11111110  C: 1  O: 0  Z: 0
  EXP | R: 01111111 11111111 11111111 11111110  C: 1  O: 0  Z: 0
SUB   | A: 00000000 00000000 00000000 00000001  B: 01111111 11111111 11111111 11111111
  ACT | R: 10000000 00000000 00000000 00000010  C: 0  O: 0  Z: 0
  EXP | R: 10000000 00000000 00000000 00000010  C: 0  O: 0  Z: 0
--------------------------------------------------------------------------------------
```

```
TESTS FOR SLT
-------------------------------------------------------------------------------------
SLT   | A: 00000000 00000000 00000000 00000000  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
SLT   | A: 01111111 11111111 11111111 11111111  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
SLT   | A: 00000000 00000000 00000000 00000000  B: 01111111 11111111 11111111 11111111
  ACT | R: 00000000 00000000 00000000 00000001  C: 0  O: 0  Z: 0
  EXP | R: 00000000 00000000 00000000 00000001  C: 0  O: 0  Z: 0
SLT   | A: 01111111 11111111 11111111 11111111  B: 00000000 10000010 10110110 00110101
  ACT | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
SLT   | A: 00000000 10000010 10110110 00110101  B: 01111111 11111111 11111111 11111111
  ACT | R: 00000000 00000000 00000000 00000001  C: 0  O: 0  Z: 0
  EXP | R: 00000000 00000000 00000000 00000001  C: 0  O: 0  Z: 0
SLT   | A: 11111111 01111101 01001001 11001011  B: 00000000 10000010 10110110 00110101
  ACT | R: 00000000 00000000 00000000 00000001  C: 1  O: 0  Z: 0
  EXP | R: 00000000 00000000 00000000 00000001  C: 1  O: 0  Z: 0
SLT   | A: 11111111 01111101 01001001 11001011  B: 00000000 00000000 00000000 00000000
  ACT | R: 00000000 00000000 00000000 00000001  C: 1  O: 0  Z: 0
  EXP | R: 00000000 00000000 00000000 00000001  C: 1  O: 0  Z: 0
SLT   | A: 11111111 01111101 01001001 11001011  B: 10000000 00000000 00000000 00000001
  ACT | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 1  O: 0  Z: 1
SLT   | A: 10000000 00000000 00000000 00000001  B: 11111111 01111101 01001001 11001011
  ACT | R: 00000000 00000000 00000000 00000001  C: 0  O: 0  Z: 0
  EXP | R: 00000000 00000000 00000000 00000001  C: 0  O: 0  Z: 0
SLT   | A: 11111111 01111101 01001001 11001011  B: 01111111 11111111 11111111 11111111
  ACT | R: 00000000 00000000 00000000 00000000  C: 1  O: 1  Z: 1
  EXP | R: 00000000 00000000 00000000 00000000  C: 1  O: 1  Z: 1
```

Listing 1: Text results for the ALU.

3.1 Results

Our comprehensive but non-exhaustive tests show that our ALU functions as expected. Each of the "basic" gates (AND, NAND, OR, NOR, XOR) function properly, and the adder, subtractor, and SLT behave as expected. Tests were chosen to evaluate edge cases (e.g. 0, large numbers, negative computations) as well as more "normal" scenarios (e.g. numbers that are not 0 and 2147483647_{10}).

Because the SLT is based off of the 32 bit adder in our ALU, comparing two numbers that are large in magnitude cause overflow, resulting in an invalid comparison.

4 Timing Analysis

The results of the timing analysis are shown in table 1. The worst case propagation delay is 4060 time units and occurs in the SLT path. The main reason for this large delay is the adder module, which is a propogation of 32 modules requiring 120 time units each.

Table 1: ALU Module Timing Analysis

Module	Compenent Timing	Total
SUB	70 + 3840 + 70 + 70	4050
SUB	ADD + 10	4060
SLT	SUB + 30 + 210	240
NAND	20 + 210 + 70	300
AND	NAND + 10	400
NOR	20 + 210 + 170	400
OR	NOR + 10	410
XOR	60 + 210 + 70	420

5 Work Plan Reflection

5.1 Work Plan

```
------------------------
CompArch Lab 1 Work Plan
------------------------

(1.5) Build test bench
   (1)   Finding and choosing appropriate test cases
   (0.5) Implement test cases
(3)   Build modules
   (1)   AND, NAND, NOR, OR
   (1)   Adder, Subtractor
   (1)   SLT
(2.5) Report

William:  Start coming up with test cases
Ian:      Start writing the test bench
William:  Write AND, NAND, NOR, OR modules
Katie:    Write Adder, Subtractor, and SLT modules
Everyone: Write report

------------------------
```

```
         Notes
------------------------


Verilog module
    -Adder: reuse 4 bit adder from Lab1
    -Subtractor: tweak 4 bit adder from Lab1
    -XOR: Modified OR gate
    -SLT: Build 1 bit, string 32 of them together
    -AND: Use structural AND
    -NAND: Use structural NAND
    -NOR: Use structural NOR
    -OR: Use structural OR
Test bench
    -Test cases
    -Implementation
Report
    -Implementation
    -Test Results
    -Timing analysis
    -Reflection
```

5.2 Reflection

The total times spent on each portion of the work plan deviated from our initial estimates as follows:

```

(1.5) Build test bench
   (1)   Finding and choosing appropriate test cases
        It took 2 hours rather than the initial estimate of 1 to select appropriate
            test cases for each of the modules.
   (0.5) Implement test cases
        Implementing the test case file also took approximately double the estimated
            time to complete.
(3)   Build modules
    The amount of time spent building the individual modules totaled to 5 hours.  The
        initial estimate of 3 hours total for the three subsets of modules did not
        take into consideration aspects including the main ALU module and updating the
         timing of the structural verilog gates.
(2.5) Report
    Generating the documentation, figures, and explanations for the report was
        approximately within the time given in our workplan.
```