CMPE-260 Laboratory Exercise 5
Execute Stage

Chris Larson
Performed 14 March 2019
Submitted 9 April 2019

Lab Section 2
Instructor: Jason Blocklove
TA: Priya Jain
Nabiel Kandiel
James Snedecor

Lecture Section 02
Professor: Richard Cliver

Abstract

In a MIPS processor, once an instruction has been fetched from memory and decode it needs to be executed. The execute stage consists of an ALU and several multiplexers, which determine how the instruction being passed in is executed. The ALU for this exercise is similar to the one in Lab 1 but a Ripple-Carry full adder/subtractor and a multiplier are to be added. Once the ALU has all of the components in it, the entire execute stage must be designed and written so that it can perform with previous stages.

Design Methodology

The ALU and Execute stage were drawn as a diagram to help visualize what needs to be designed. The ALU has three inputs, the ALU op-code, in1 and in2, which are inputs to the ALU which depending on the op-code are put into the module according to the op-code and the result from the ALU operation is outputted as out1. This is shown in Figure 1.

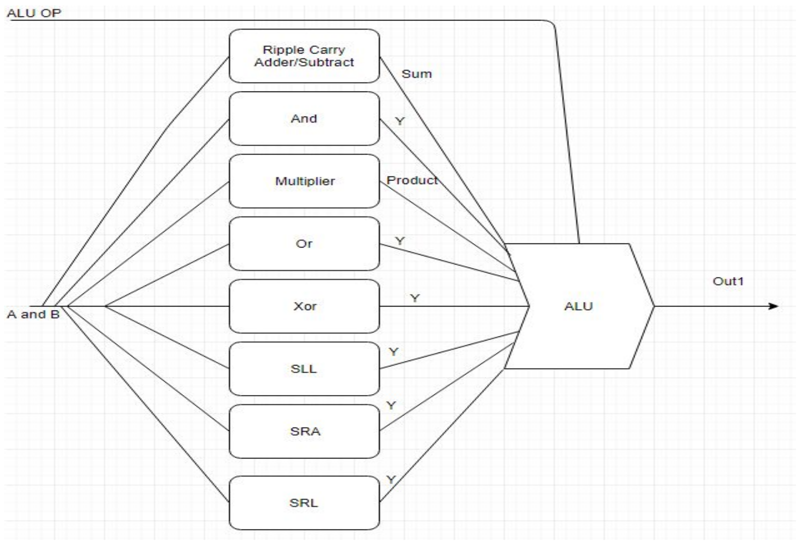

Figure 1: The ALU block diagram

The Execute Stage has seven inputs, the ALU op-code, the first and second value for the ALU to act on, the index for memory writeback, the writeback enable bit, the read and the write bit. The outputs are the data to be written to the memory stage which is always the second value for the ALU to be acted on, the result from the ALU, the writeback index, enable, read and write bit. The Execute Stage is shown in Figure 2. The And, Or, Xor, SLL, SRA, and SRL functions were taken from Lab 1, while the multiplier was designed as a generic version of the Figure 5.1 in the Manual, and the ripple carry adder/subtractor was designed as specified in the manual.

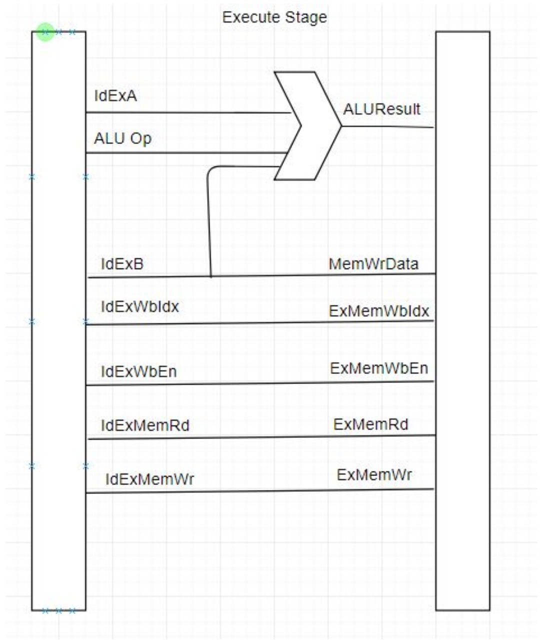

Figure 2: Execute stage diagram

The full ALU was created with the components and a behavioral simulation, post-implementation timing simulation , post-synthesis timing simulation was performed and then the ALU was demonstrated on the hardware chip. The execute stage without the multiplier had a behavioral simulation done, then the multiplier was simulated. Finally the entire execute stage with full ALU was put together and simulated.

Results and Analysis

A testbench was made to ensure the correctness of all of the operations and components of the exercise as well as compare the waveforms to expected results. A behavioral testbench was created to tested each of the operations, and changing the inputs. The ALU behavioral simulation is shown in Figure 1.

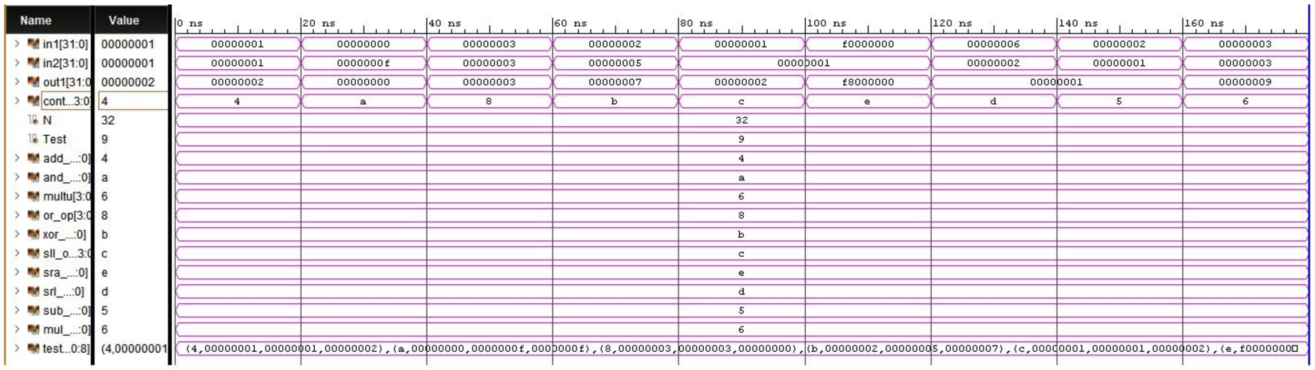
Name
Value
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
in1[31:0] 00000001 | 00000001 00000000 00000003 00000002 00000001 f0000000 00000006 00000002 00000003
in2[31:0] 00000001 | 00000001 0000000f 00000003 00000005 00000001 00000002 00000001 00000003
out1[31:0 00000002 | 00000002 00000000 00000003 00000007 00000002 f8000000 00000001 00000009
cont...3:0] 4 | 4 a 8 b c e d 5 6
N 32 | 32
Test 9 | 9
add_...:0] 4 | 4
and_...:0] a | a
multu[3:0 6 | 6
or_op[3:0 8 | 8
xor_...:0] b | b
sll_o...3:0 c | c
sra_...:0] e | e
srl_...:0] d | d
sub_...:0] 5 | 5
mul_...:0] 6 | 6
test...0:8] (4,00000001 | (4,00000001,00000001,00000002),(a,00000000,0000000f,0000000f),(8,00000003,00000003,00000000),(b,00000002,00000005,00000007),(c,00000001,00000001,00000002),(e,f0000000

Figure 1: ALU Behavioral Simulation

Each of the different functions in the ALU were tested once and the results matched the expected results. The ALU Op code was changed to go through each function and that worked exactly as it was supposed to. Figure 2 shows the Behavioral Simulation for the Execute Stage.

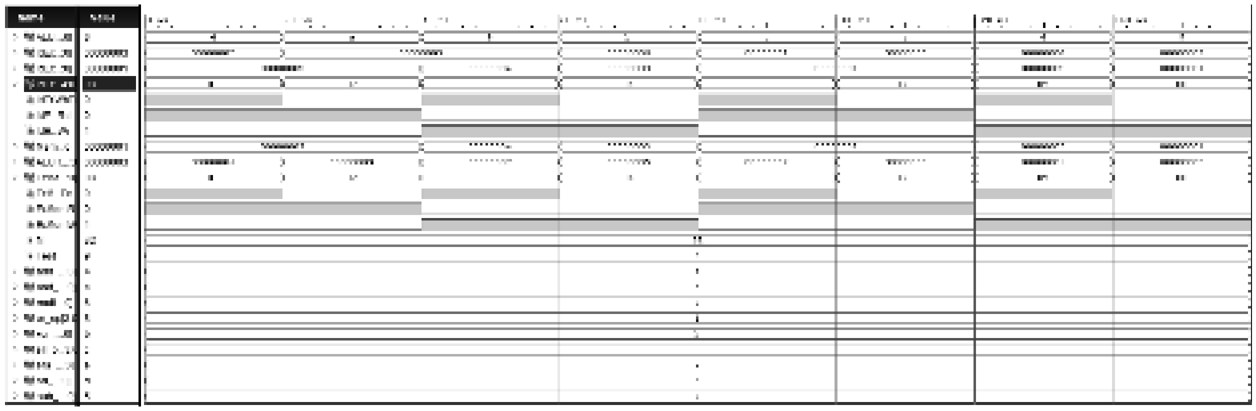

Figure 2: Execute Stage Behavioral Simulation

The Execute Stage was given different inputs to see if the outputs would match the correct value and each of the different functions in the ALU were tested, the results matched the expected results. A Post-Implementation Timing Simulation was done on the ALU to test the functions as if they were on a board. Figure 3 shows the Post-Implementation Timing Simulation for the ALU.

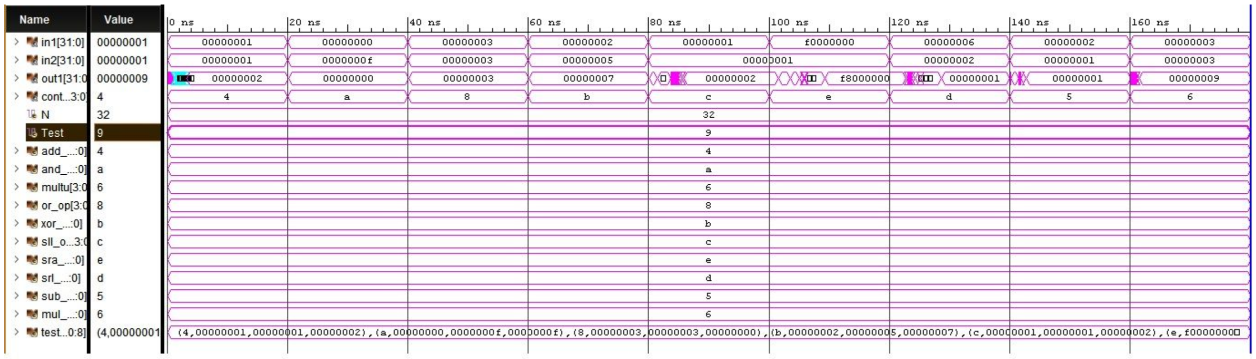

Figure 3: ALU PITS

Each of the functions of the ALU were tested again with more realistic timing, as the PITS builds the ALU as if it were on an actual piece of hardware. The results from the simulations exactly matched the expected results, this means that the ALU and the Execute stage were designed and written correctly.

Conclusion

The Execute stage is important for the computation of the processor, this is where the arithmetic, logic, and shifting is done. It also passed bits through to the writeback and memory stage. The exercise was a success because all of the testbench simulations matched expected results and all of the functions of the ALU worked as intended.