Lab 4

By

Don Dang

Brigid Kelly

May 13th, 2018

Microprocess Designs

Ken Rabold

**Section B: Table of Contents**

Section A : Title………………………………………………………………………………………………………..…..1

Section B : Table of Contents…………..……………………………………………………………………….....2

Section C : Design Descriptions…………. ………….……………………………………..……………………..3

Section D : Screenshots of Part 1 – ALU ………….……………………………………..…………………….4

Section E : Truth Table for your ALU Control Input Decoding…………………………………………8

Section F : A description of how you tested your ALU design and your ASM code…………………..9

Section G : Limit on the Fibonacci number your program can calculate? Why or why not?……10

Section H : ASM Screenshots………….…………………………………………………………………………………… 11

Section I : Lab4 link to GitHub……………………………………………………………………………………….13

**Section C: Design Descriptions**

Insert test code into the provided TestALU.vhd file from the instructor to test the following arithmetic logical operations:

1. ADD
2. ADDI
3. SUB
4. ZERO
5. AND
6. OR
7. ORI
8. SHIFT LEFT
9. SHIFT RIGHT

Each arithmetic test implementation requires three parameters: datain1, datain2 and control. The objective of the test is to pass in three hex values and observe the outcome. The test values are hard coded into the TestALU.vhd. The result of each arithmetic operation is observed from the wave output in Questa Sim. Each test operation is delayed by 20 ns.

Below is the result of each test implementation captured from the wave file.

**Section D: Screenshots of Part 1 – ALU**

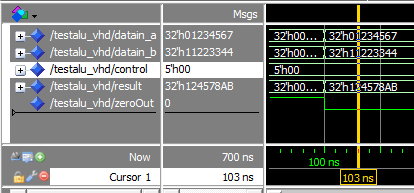


Figure 1 ADD Test

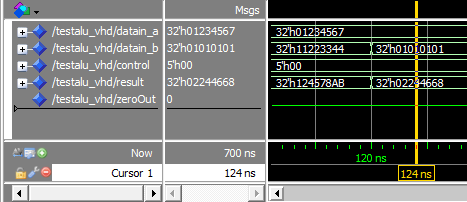


Figure 2 ADDI Test

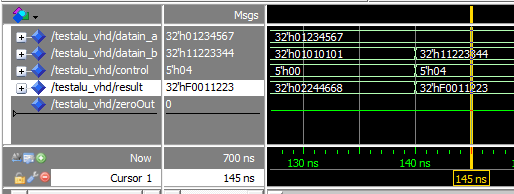


Figure 3 SUB Test

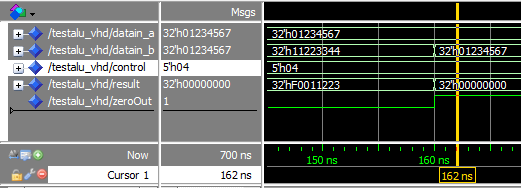


Figure 4 ZERO Test

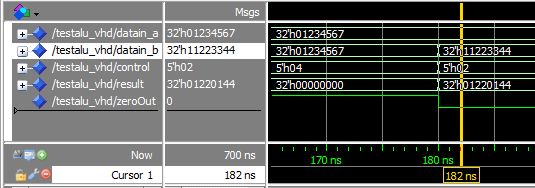


Figure 5 AND Test

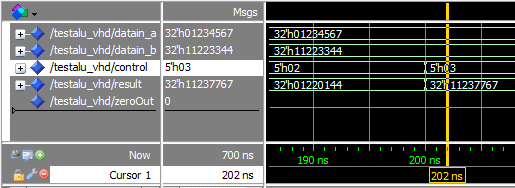


Figure 6 OR Test

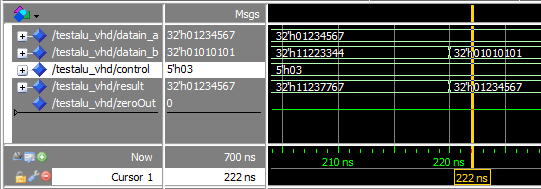


Figure 7 ORI Test

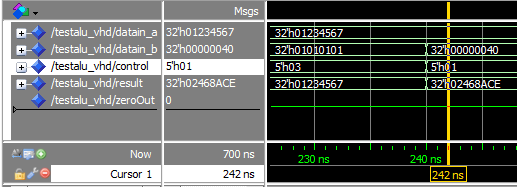


Figure 8 SHIFT LEFT Test

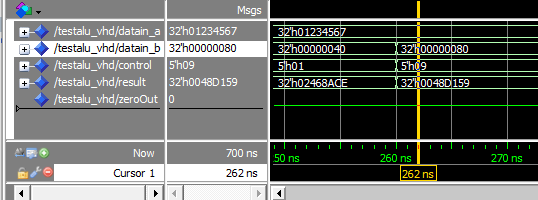


Figure 9 SHIFT RIGHT Test

**Section E : Truth Table for your ALU Control Input Decoding**

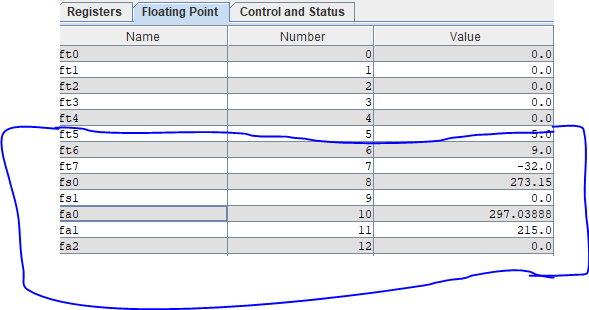
|  |  |  |  |
| --- | --- | --- | --- |
| **DataIn1** | **DataIn2** | **Opcode** | **ABI Name** |
| 0 | 0 | 000 | ADD |
| 0 | 1 | 000 | ADDI |
| 0 | 0 | 100 | SUB |
| 0 | 0 | 010 | AND |
| 0 | 1 | 010 | ANDI |
| 0 | 0 | 011 | OR |
| 0 | 1 | 011 | ORI |
| 0 | 0 | 001 | SLL |
| 1 | 0 | 001 | SRI |
| 1 | 0 | 001 | SLLI |
| 1 | 1 | 001 | SRLI |

**Section F: A description of how you tested your ALU design and your ASM code**

**Section G: Is there a limit on the Fibonacci number your program can calculate? Why or why not?**

It will depend on the value of the maximum the 32 bits register can hold. If the value is 0 x F F F F F F F F, this is equivalent to 4, 294, 967, 295.

**Section H: ASM Screenshots**



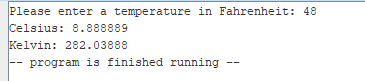


Figure 10 Display Result of Celsius and Kelvin Conversion

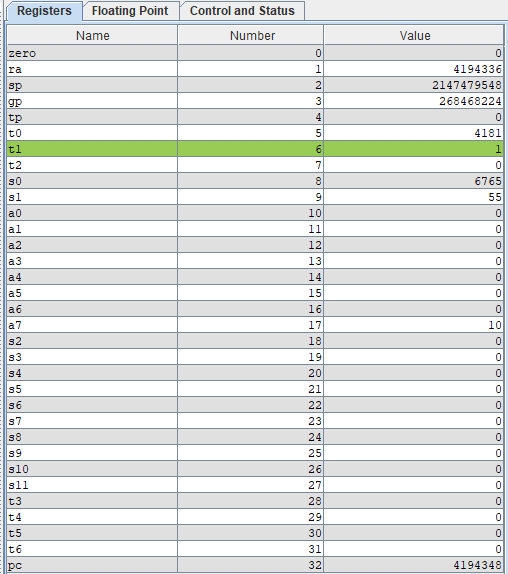


Figure 11 - Result of Fibonacci's Register

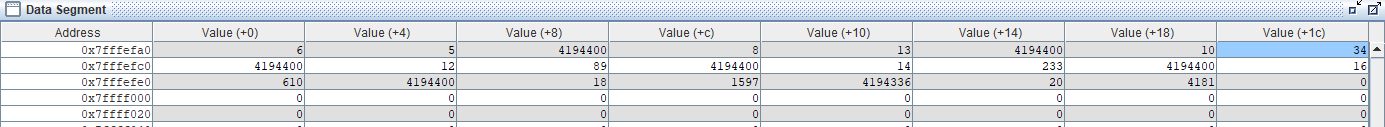


Figure 12 Result of Fibonacci's Memory

**Section H: Lab2 link to GitHub**

<https://github.com/donjuanwu/Labs/tree/master/Lab%204>