Processor Design Report

This document must be submitted in addition to the AG350 submission by the project deadline. The purpose of this document is to explain your processor design and the process you went through to produce it. It is composed of four segments: Construction; Implementation; Integration; and Process.

* Required

1. Name *

_Wei Xue_____

2. Net ID *

_wx59_____

3. Honor Code: I have neither given nor received unauthorized help in completing this assignment and I have conducted myself within the guidelines of the Duke Community Standard. *

Mark only one oval.

Yes __✓_____
No _____

Construction

This aspect considers the logical soundness of your processor.

4. How do you determine a branch should occur for bne? (200 characters max) *

As shown in the project document, 'bne' instruction is to compare the data from two registers 'rd' and 'rs' and then jump to the branch if they are not equal. To accomplish the goal, first I set two read control signals of Regfile to 'rs' and 'rd'. And then send the read data from Regfile into ALU module. Since ALU has an output, 'isNotEqual', if it is high, then the branch jump of 'bne' operation should occur. Then I could use the signal 'isNotEqual' as a selection signal of mux to select the right signal for next PC value, which is (original PC) + N + 1. If the result is false, I will choose the results of other signals or PC+1 as next PC value.

5. How do you determine a branch should occur for blt? (200 characters max) *

As shown in the project document, 'blt' instruction is to compare the data from two registers 'rd' and 'rs' and jump to the branch when 'rd' value less than 'rs' value. The former steps are the same as 'bne'. As for 'isNotEqual' signal, if it is high, the result shows 'rs' is less than 'rd', which is opposite to the requirement. So I should use the signal 'isLessThan' together with the signal 'isNotEqual'. Then if (not 'rs' is less than 'rd') and (not 'rs' is equal to 'rd'), then 'rd' must be less than 'rs'. I can use such result as a selection signal of mux to select the right signal for next PC value, which is (original PC) + 1 + N. If it is false, I will choose the result of other signals or PC+1 as next PC value.

Implementation

This aspect considers efficiency, cost, simplicity, and elegance of your processor.

6. What is the slowest instruction in your processor? Explain why you know this. (200 characters max) *

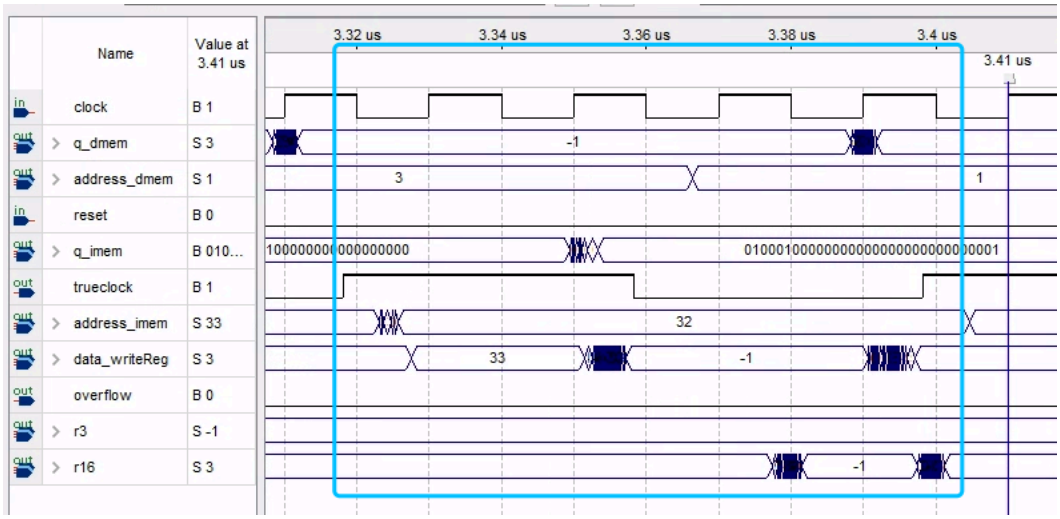
As far as I concerned, 'lw' instruction may be the slowest instruction in my processor design. As I tested and compared, I-Mem and D-Mem take much more time than any other modules. Every instruction needs to read from I-Mem but only 'lw' and 'sw' need to use D-Mem module. Compared with 'sw', actually compared with any other instructions, 'lw' needs to take one extra steps, which is writing the fetched data from D-Mem into registers. And to write to registers, I must wait until next clock edge to trigger the write operation. So 'lw' may be the slowest instruction in my processor design.

7. Estimate your processor's maximum clock speed in ns (i.e. the fastest clock speed that allows the slowest instruction to correctly execute). *

My processor can work on the 50MHz input system clock and the real working clock for every instruction is 12.5MHz. As for the whole processor design, there are two aspects that need to be taken into consideration. First, some memory modules has relatively long latency, such as I-Mem and D-Mem. Second, several modules are edge-triggered, so I need to assign different edges for those modules to write or latch the correct data in those modules. For the slowest 'lw' instruction, it will go through I-Mem and D-Mem modules. As I tested, these two modules require about 20ns (1 system cycle) to get the output data under the 50MHz system clock. And the other parts actually work so fast that it nearly takes less than 10ns to get the output of every module(i.e. PC, Regfile, ALU). Then 'lw' may take 20 + 20 + 10*(3~4) $\approx$ 70ns~80ns (including the edges to trigger different modules) to finish all the steps, which would be the real fastest clock speed under such situation and design. And after the below testing, the clock frequency could match all the aspects of my design.

8. Prove the above clock estimation by attaching a waveform of your processor functioning at this speed. *

Files submitted: ___See attached test.vwf and the picture below__



As the above waveform timing simulation figure, the instruction at the address 32 is actually:

```
0032 : 01000100000000000000000000000001; -- lw $16 1($0)    # r16 = dmem[1] = 3
```

The instruction starts from PC at the first rising edge of the 'trueclock', then get the address number '32' from the PC. After '32' becomes the input address of I-Mem, it would take about one system clock cycle to get the instruction. Then the instruction would come out before the falling edge of the 'trueclock' which means before the half of the real clock cycle. And the address of the D-Mem will be calculated out right after the falling edge. Then it would take about one system clock cycle to get the data from D-Mem. After that, the data would be sent to the data_writeReg, waiting for the next system clock rising edge to write into the register. Finally, As showing in the waveform figure, register 16's value becomes 3 which is the expected result.

Integration

This aspect considers how well you tie together your memory elements, regfile, ALU, and other elements.

9. What clocking scheme does your processor follow, i.e., how is each clocked element in your design clocked? *

Check all that apply.

PC Register	Positive edge? __✓___	Negative edge? _____
Regfile	Positive edge? __✓___	Negative edge? _____
Dmem	Positive edge? _____	Negative edge? __✓___
Imem	Positive edge? _____	Negative edge? __✓__

10. Why did you choose the above clocking scheme? (200 characters max) *

First, PC is the start point of every new instruction. So I set PC Register positive edge triggered which is the first positive edge of a new clock cycle. Then I set I-Mem negative edge triggered which is the first negative edge right after PC, since there is nearly no delay in the circuit between PC and I-Mem. I-Mem may take 1 system clock cycle to get the output data, the instruction to be executed. Then I set D-Mem also negative edge triggered, which is actually faster than positive edge triggered as I tested. Besides, the Regfile read operation and calculation in ALU would finish within 0.5-1 system clock cycle. Then the following negative edge will be the fastest one for D-Mem. After that, D-Mem may take about 1 system clock cycle to get the output data. If I need to write the data into Regfile, I need to set the Regfile positive edge triggered which may be the rising edge of next new clock cycle in the slowest situation, which is the nearest clock edge after the output of D-Mem data as I tested.

11. What module(s) do you use to compute the next PC? *
Check all that apply.

ALU _____✓_____
32 bit Adder _____
Other: __sign-extension_____

12. Regarding the question above, why did you choose these modules for computing the next PC? (200 characters max) *

To calculate the next PC, the only operation is addition. Since the given optimized behavioral ALU module could achieve nearly the same or even better performance in both speed and power, there's no need writing extra new adder modules. So I just re-use the given ALU module as a new instance to calculate next PC. There are three situations for next PC. First, next PC = PC + 1, in such situation I just use (original PC) and 1 as two operands and perform addition in a new ALU instance, then I can get the value of next PC. Second, next PC = PC + 1 + N. Then I use the result of the first step and the immediate value N(after sign extension) as two operands and perform addition in a new ALU instance, then I can get the value of next PC. Third, I just use the higher 5 bits of original PC and immediate value T(27 bits) to combine into next PC value. After doing the calculation in above three situation, I use the opcode and control signals with mux to choose the right value for next PC.

Process

This aspect considers how you went about designing the processor, i.e. you own work ethic and operations.

13. How did you prepare your processor design? Ex. You created an initial visual model through Logisim or by hand, you broke down the processor into modules you wanted to design, etc. (200 characters max) *

At first, I tried to start my design from the data-path shown in the course slides. I tried to construct some fundamental modules for the data-path, such as DFFE, clock divider, two RAMs and mux. Then I tried to break the whole data-path down into some sub-modules. First sub-module is PC part, I tried to construct a PC together with an I-Mem and an ALU(Adder). After finishing and testing first sub-module, I tried to add Regfile part into the design. Most parts are similar to the slides, but the read and write controls are a little different. In our processor, 'rd' is usually used as write control signal but sometimes used as one of the read control signals. So I need to add mux to select it. After this part, I added ALU and D-Mem into my design. These parts are identical to the slides. Then, the most confusing part is the implementation of branch and jump. I tried to list the situations for different branch and jump instructions into a table, and used these situations into signals to choose from different next PC values. Finally, send the correct next PC value to the input of PC. Besides, about timing sequential, I tried the latency of every sub-modules and assigned different clock edges to make the whole data-path run correctly.

14. Which of the following testing methods did you use throughout your design process? (Select all that apply) *

Check all that apply.

Waveform tests of single modules _____✓_____
Personally created testbenches for single modules _____
Waveform tests of subgroups of modules _____✓_____
Personally created testbenches for subgroups of modules _____
Waveform tests of cumulative processor design _____✓_____
Personally created testbenches for cumulative processor design _____
Independent web research and resolution of error/warning messages _✓__
Submitting to AG350 and receiving feedback on failed test cases ___✓___
Other: _I tried to write some mips codes to cover every sub-modules of my data-path for testing. _

15. Describe one time when you found a bug in your processor: how did you find, isolate, and resolve it? (300 characters max) *

One of bugs I found during my design and test is that my data-path couldn't work correctly when reading from and writing to the same register. Since the provided Regfile has a special design that it would set the read data to 'Z' when reading and writing to the same register, I need to do some special designs to avoid the problem. Actually I found the bug during some random tests. Some values would go wrong after the instruction like 'add $1, $3, $1'. Then I connected some inner signals like the two read ports of Regfile and write data of Regfile to the output of top level

entity ‘skeleton’, and then observed the outputs in waveform. I found the bug was from Regfile module. Sometimes the read port would output ‘Z’. To resolve the bug, I firstly checked the design and logic in the Regfile module. Then I noticed such special design. Since I couldn’t modify the given behavioral Regfile module, I must deal with the bug in my processor module. I tried some different solutions by hand on the paper. Finally I found a reasonable solution that I just set the ctrl_writeenable singal to be high at the downside level of every real clock cycle. That is to say, I make my processor read the data from Regfile in the former part of the real clock cycle and write to the Regfile in the latter part of the real clock cycle. Such design matches the sequence of every given instructions. To implement the design, I add an ‘and’ gate to do an logic ‘and’ calculation on the ctrl_writeenable singal and the down level of the clock signal and set the signal to be the true ctrl_writeenable for Regfile. After some functional and timing tests, I got the correct output with my design. I successfully resolved this bug.