**Processor Design Report**

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 \*

\_\_\_Xiang Ji\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Net ID \*

\_\_\_ xj52\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. 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 \_\_\_X\_\_\_

No \_\_\_\_\_\_\_\_\_

# Construction

This aspect considers the logical soundness of your processor.

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

For bne, I need to compare $rs and $rd, if they are not equal, it should jump to specific instruction. First, I “assign is\_bne = ~Opcode[4] & ~Opcode[3] & ~Opcode[2] & Opcode[1] & ~Opcode[0]; ”, because the opcode of bne is 00010. Then, I choose the $rd rather than $rt to pass out as data\_readRegB, using “assign ctrl\_readRegB = is\_bex? 5'd0 : (is\_bne | is\_blt | is\_sw ? rd : rt);”. Next, if the isnotequal port of ALU is hot, then it will pass to the PC controller. Finally, update the pc register value with pc = pc + N +1 and thus can jump to another instruction, using “assign ctrl\_pc\_N\_1 = (is\_bne & isNotEqual)| (is\_blt & ~isLessThan & isNotEqual);”.

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

For blt, I need to justify whether $rs is less than $rd, if so, it should jump to specific instruction. Firstly, I “assign is\_blt = ~Opcode[4] & ~Opcode[3] & Opcode[2] & Opcode[1] & ~Opcode[0];”, because the opcode of blt is 00110.

Then, I choose the $rd instead of $rt to be passed out as data\_readRegB, using “assign ctrl\_readRegB = is\_bex? 5'd0 : (is\_bne | is\_blt | is\_sw ? rd : rt);”. Next, if the isnotequal port of alu is hot and the lessthan port is hot, pass it to the PC controller. Finally, update the pc register value with pc = pc + N +1, using “assign ctrl\_pc\_N\_1 = (is\_bne & isNotEqual)| (is\_blt & ~isLessThan & isNotEqual);”.

# Implementation

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

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

The lw instruction is the slowest instruction. For the lw instruction, the processor has to read instruction from imem, it has to read data from regfile for $rs, and it has to pass to ALU with the offset. After ALU’s calculation, the processor will take the data out of dmem and write the dmem data into regfile. Thus, the lw instruction involves the largest number of procedures.

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

1/48ns = 20.8MHz by looking at lw instruction

图形用户界面, 应用程序

描述已自动生成

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

Files submitted: \_\_\_\_\_\_ Self\_test.vwf\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Integration

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

1. 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? \_\_X\_\_ Negative edge? \_\_\_\_\_\_\_

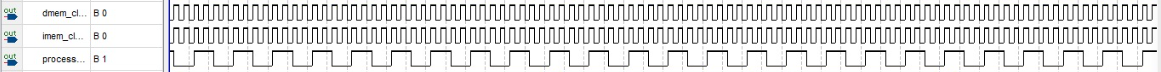
Regfile Positive edge? \_\_X\_\_ Negative edge? \_\_\_\_\_\_\_

Dmem Positive edge? \_\_X\_\_ Negative edge? \_\_\_\_\_\_\_

Imem Positive edge? \_\_X\_\_ Negative edge? \_\_\_\_\_\_\_

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







For a datapath to control timing, we have to consider the order of data passed.

For imem, it should fetch the data after the register write in Regfile at the last circle.

For the dmem, it uses the clock which is the same as input clock.

The processor and regfile can be triggered late so divide the frequency by 4 and then negate them, because the processor has to deal with the whole operation of an instruction in one cycle.

1. What module(s) do you use to compute the next PC? \*

*Check all that apply.*

ALU \_\_\_\_\_\_\_\_\_\_\_\_\_

32 bit Adder \_\_\_\_\_\_\_

Other: \_I use syntactic sugar.

assign ctrl\_pc\_T = is\_j | is\_jal | (is\_bex & isNotEqual);

assign ctrl\_pc\_N\_1 = (is\_bne & isNotEqual)| (is\_blt & ~isLessThan & isNotEqual);

assign pc\_next = is\_jr? data\_readRegA : (ctrl\_pc\_T? sx\_immed\_T : (ctrl\_pc\_N\_1? pc+sx\_immed\_N+32'd1 : pc+32'd1));

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

Because the syntactic sugar is the easiest. I only need to check if the instruction is jr or if the instruction is (is\_j | is\_jal | (is\_bex & isNotEqual)) or if the instruction is (is\_bne & isNotEqual)| (is\_blt & ~isLessThan & isNotEqual).

# Process

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

1. 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) \*

I created the initial visual model by hand. I developed the details according to sections, including Instruction Fetch, Instruction Decode, Operand Fetch, Execute, Result Store, what to write and Next Instruction. And I used syntactic sugar to figure out my code.

1. 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 \_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Personally created testbenches for single modules \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Waveform tests of subgroups of modules \_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Personally created testbenches for subgroups of modules \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Waveform tests of cumulative processor design \_\_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_\_\_\_

Personally created testbenches for cumulative processor design \_\_\_\_\_\_\_\_\_

Independent web research and resolution of error/warning messages \_\_X\_\_

Other: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

When I first use waveform to test my instructions, I found that my setx and bex instructions were incorrect. Besides, I wanted some wires to be the output of my waveform. Therefore, I added some output in my processor and my skeleton test 2 file. Besides, I fixed my assign instructions to resolve the problem about setx and bex. In the end, it seems that there is no bug.