**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 \*

\_Rundong Wang\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Net ID \*

\_rw310\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 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 \_\_\_1\_\_\_\_\_\_

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) \*

The key to determining whether branch of bne occurs is to compare the results of '$Rs' and' $Rt '. In order to achieve this step, the relevant control signal is' ALUinB=0 '(select the value of the register to compare),' ALUop=1 '(subtract to determine whether it is equal). Finally, we will use the result of ALU comparison 'isNotEqual' and bne's control signal 'is\_bne' to determine whether to update the address of the PC in the bne instruction. Specifically, when 'isNotEqual==0 &&is\_bne ==1', we will decide whether to update the address of the PC in BNE instruction. branch required

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

Similar to the method used to determine bne, we first control the associated control signal to compare the values of the two registers. Unlike bne, this time we refer to 'isLessThan'. If 'isLessThan == 0 &&is\_blt ==1 &&isNotEqual ==1', we create a branch

# 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) \*

It’s sw

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

50Mhz 25 ns

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

Files submitted: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 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? \_\_\_\_\_\_ Negative edge? \_\_\_\_\_\_\_

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

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

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

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

*Check all that apply.*

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

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

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

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 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) \*

First, I refer to an abstract datapath in the courseware, and roughly determine what components and how they interact. Moreover, all my subsequent designs are based on this abstract datapath in the courseware.

I think what we need to focus on in this project is the control signal and the way it changes. So early on in my design, I focused on the detailed semantics of each control signal and the scenarios in which they changed.

I set up a few checkpoints for my projects, (1) Complete and debug required components (DFFE, frequency\_divider, IMem, DMem, etc.) (2) Load incremental PCS and print the result (only PC = PC + 1 in this case) (3) Complete inst\_decode (4) Complete ALU operation ⑤ Complete the jump (determine the next PC value)

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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 \_\_\_\_\_

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

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

\_[bug] The pc value never changes

Background: The creation of imim.mIF has been completed (so there will be some instructions in IMem when the program starts)

Situation description: In the logic of pc value update, the value of pc never changes

Possible cause 1: The update of the pc value depends on the register I implemented. There may be an error in the register

Check # 1: I wrote a testbench test for the register, and the results showed that the register was fine

Possible cause 2: The new pc value needs to be calculated by the adder, which may be a problem

Check 2: Use the operator "+" instead of the adder, but it still doesn't work

Possible cause 3: The update of the pc value requires the clock to take effect. The clock may fail

Check 3-1: The clock used by the register is from stu\_processor\_clock, and this data is exposed in skeleton\_test, which I set as the output of skeleton\_test, And accepts this output(o\_stu\_processor\_clock) in our skeleton\_test\_tb that I wrote myself. I then try to print this value when o\_stu\_processor\_clock changes. The result of this is that o\_stu\_processor\_clock remains unchanged. So this is really the problem.

Troubleshooting 3-2: I printed the top clock in the same way and found no problem, so I narrowed the problem to the "clock generates stu\_processor\_clock" step, which is my frequency\_divider component.

Check 3-3: Check the frequency\_divider component and find that the implementation of frequency\_divider in the reference is reset when rst==0, but it is actually reset when rst==1