

**Project Part II**

CSCE 230 Honors

Professor Reidesel

11/6/2015

Will Preachuk

Levi Amen

Cameron Johnson

## **Overview:**

As complicated as building a processor might seem to be, there were relatively few snags in implementing the rudimentary pieces required for basic operation. The instruction register, register file, arithmetic logic unit, control unit, various multiplexers, and various buffer registers were all strung together in VHDL as components and signals. The processor is capable of simple R-type instructions as defined in the ISA (ADD, SUB, AND, OR and XOR). As of now, the processor does not implement any memory or I/O devices. Instructions must be fed through the instructionIn pin, and outputs are only visible through the ryOut pin. Nonetheless, it is operational and properly calculates values and updates registers based on the instruction set in the ISA.

**Implemented Instructions:**

The processor currently only implements 5 basic R-type instructions: ADD, SUB, AND OR, and XOR. As documented in the processor’s ISA each of these instructions use the same OpCode (0000) due to all instructions following the same control path through the processor as set by the control unit. The difference in implementation comes at stage 3 where Opx is passed to the ALU component to determine the correct value to output via an internal MUX. The control unit also passes signals for a\_inv and b\_inv to indicate which input to invert for processing a SUB instruction, as it’s computed the same as an ADD instruction with one input inverted.

## Components Added

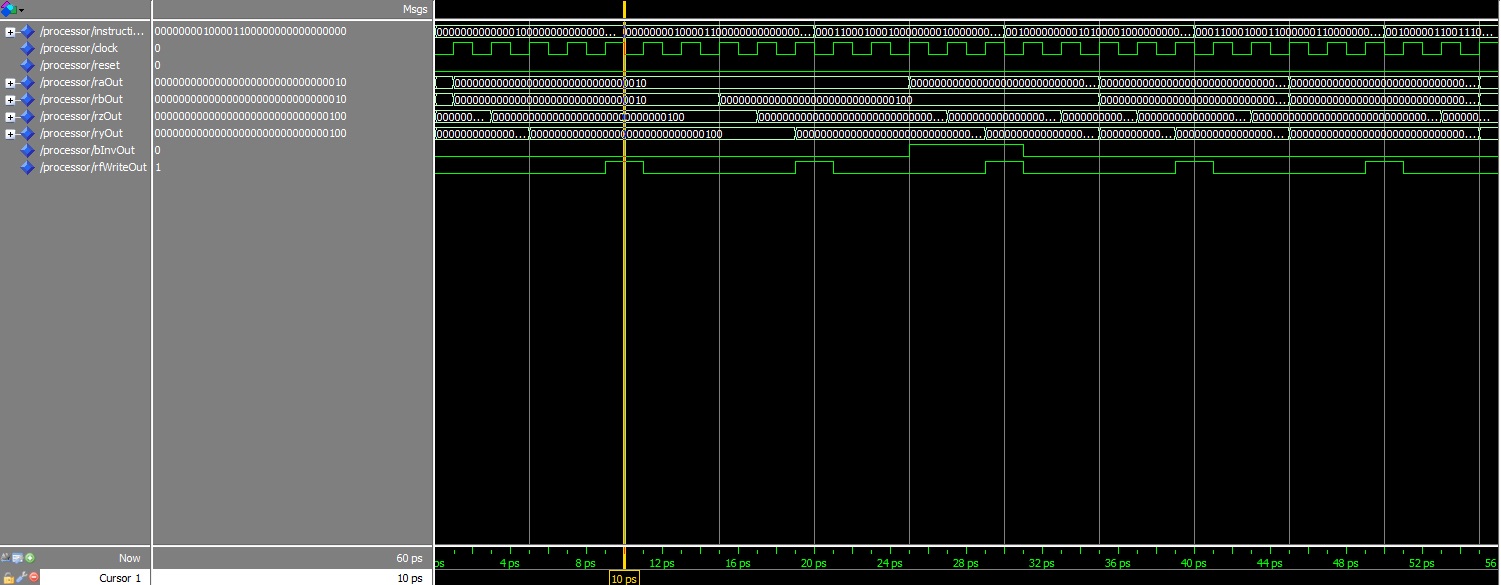
Multiple buffer registers and multiplexors were added in order to make the processor operate properly. Two buffer registers, RA and RB respectively, were added to capture the outputs from the register file. These ensure that the value coming out of the register file is properly handed off when the clock resets and will also assist in pipelining later on. A multiplexor was added to select one of the inputs to the ALU. This multiplexer selects between an immediate value and the value stored in RB. The selection is done based on a signal from the control unit, and will be used later when intermediate values are included in the instruction set. A buffer register, RZ, was used at the output of the ALU to store its value for the same reasons listed above. RZ directly connects to another multiplexor, MUXY. The control unit sends a signal to MUXY to select which value will be sent back to the register file, either the ALU output, data from memory, or a memory address. This value is then stored in another buffer register, RY, before being sent back into the register file to be stored.

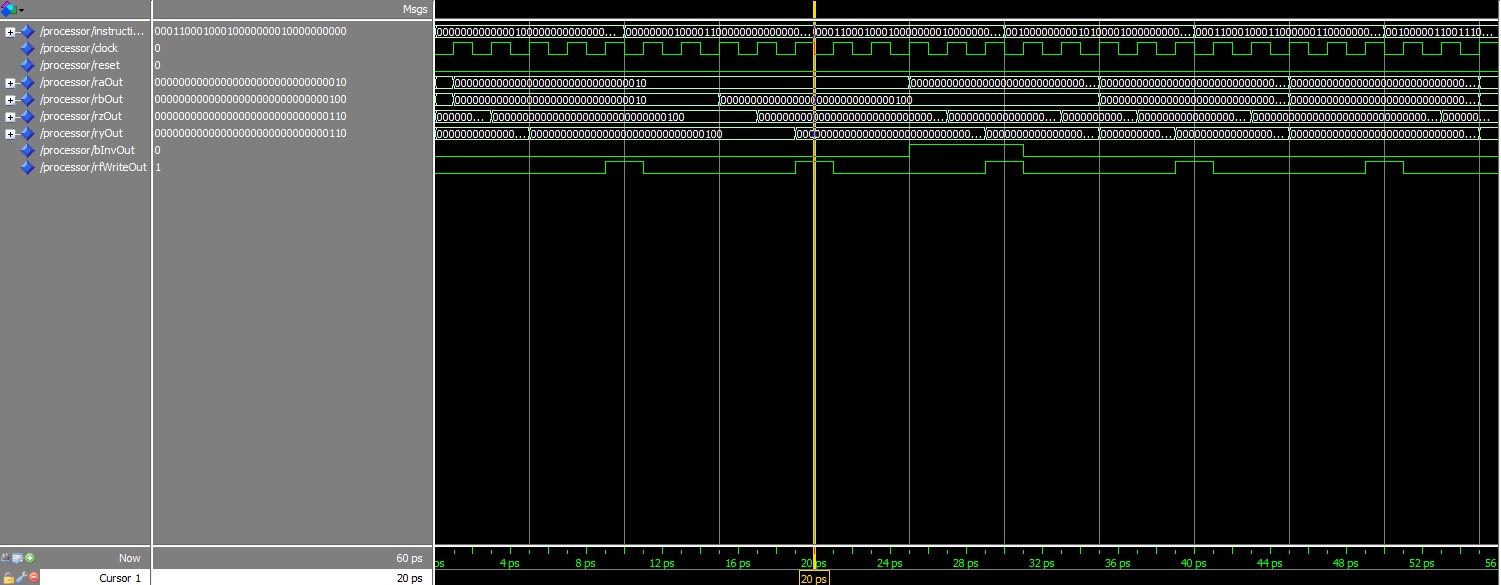
## **Processor Speed:**

The processor was successfully tested with a clock speed of 500 megahertz. The clock is simulated in a macro (.do) file using a repeat command. 500 megahertz appears to be the maximum speed capable of being used due to the constrictions of the macro file. At this speed, the command is already using a value of 1, which is the smallest value possible to use. However, when the processor is put onto a physical FPGA device, it will probably not be able to perform at these speeds. When that phase of testing is completed, more will be known. The processor can be optimized by improving various components, for example the ALU can be changed from ripple-carry to a carry-look-ahead.

## Testing:

The processor was tested using ModelSim and a macro (.do) file. The testing was done by using various instructions and observing how buffer registers changed. In order to test how the processor interacted with the register file, instructions were used that would change register values and then use those same registers later. The test was conducted by changing the value of register 0 from 0 to 2. In the following image, the instruction is adding r0 to r0 and putting the value into r2. As is evident, the output from the ALU to ryOutput is 4, the correct answer.



In the next operation, the register in which the value of 4 was stored is added to r0 again to get a value of 6, which can be seen in ryOut.

## **Group Experiences:**

We have often appreciated the fact that our group knows each other well, and that we all live near each other with similar class time schedules; this has made coordinating time to work together on this project relatively painless. Working on the processor has been a fun learning experience for all of us, with the best moment always being when we finally see our vhdl files work the way we hoped. The only issues we’ve had is working out a way to work in parallel. Currently we’ve found no efficient system to allow more than one person working on a file at a time or having more than one person testing the files at a time.