COEN316 – Lab 4

Name: Christopher Dubuc-Pesetti

Student ID: 40037815

Date submitted: November 12, 2018

"I hereby certify that this submission is my original work and meets the Faculty's Expectations of Originality."

Name: Christopher Dubuc-Pesetti ID: 40037815

Objectives:

The objective of this lab was to design and simulate the CPU datapath of the processor being designed over the semester. The components being added to this portion of the processor are the PC register, the Instruction cache, the Data cache and the sign extension of the immediate offset.

Results & Discussion:

The block diagram of the datapath connections is attached to the front of the appendix as well as the vhdl code to the datapath design.

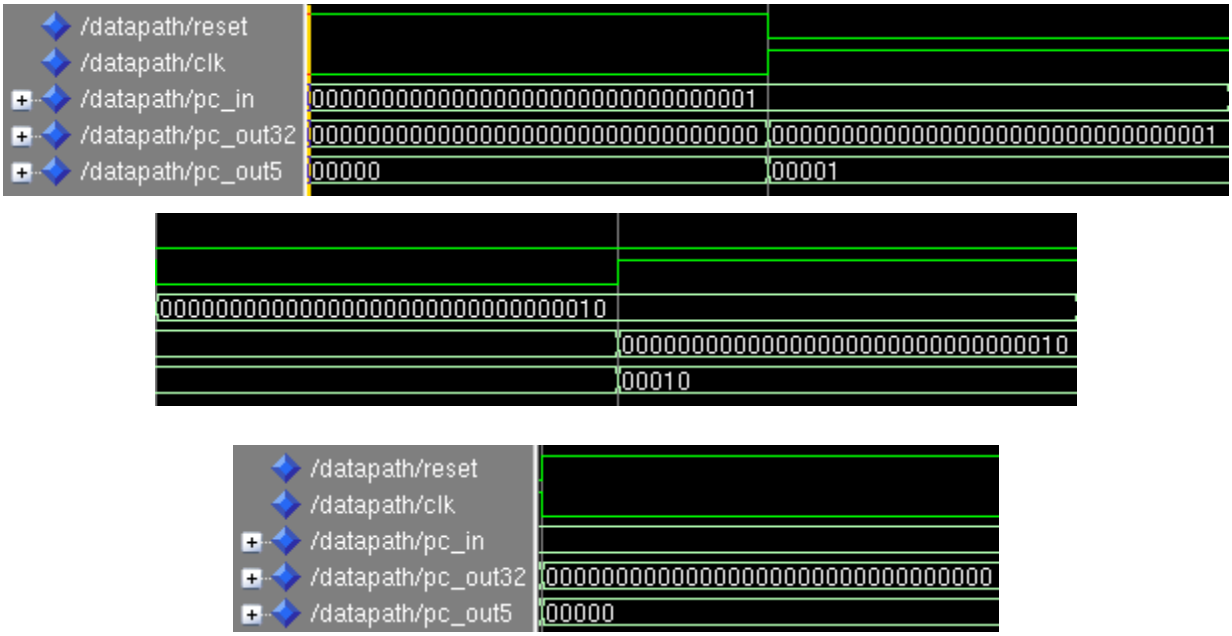

Figure 1. PC register testing

Figure 1 above shows the PC register being incremented (top two pictures) and reset (bottom picture) for the test cases. Values are in binary.

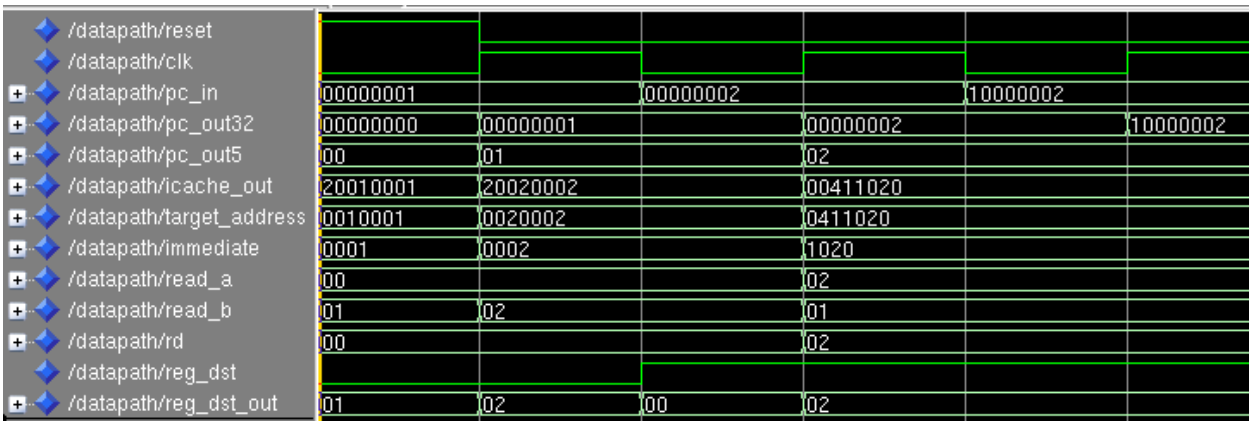

Figure 2. I-Cache testing

Figure 2 above shows the I-cache testing. All values from this point forward are shown in hexadecimal format.

Pc_out32 is the 32 bit signal of the pc counter.

Pc_out5 is the lower 5 bits of the same signal which is used as the icache address.

Icache_out is the 32 bit output of the instruction cache at the pc_out5 address.

Target_address is the lower 26 bits of the instruction output which will be sent to the Next Address Unit.

Immediate is the lower 16 bits of the instruction output to be used as offset.

Read_a is rs (25 downto 21 bits) of instruction output, to be sent to the register file.

Read_b is rt (20 downto 16 bits) of instruction output, to be sent to the register file.

Rd is bits (15 downto 11) of instruction output, sent to the “reg_dst” mux. This mux determines the input to the register file address input.

Reg_dst is the control signal for the “reg_dst” mux.

Reg_dst_out is the output of the “reg_dst” mux.

The hardcoded instruction data used was the same as the values given in the lab manual as an example.

Rs, rd, rt all receive the proper values from the icache output. The immediate and target_address signals are changed properly as well.

The multiplexer directs the proper inputs to the output when the control signal is changed.

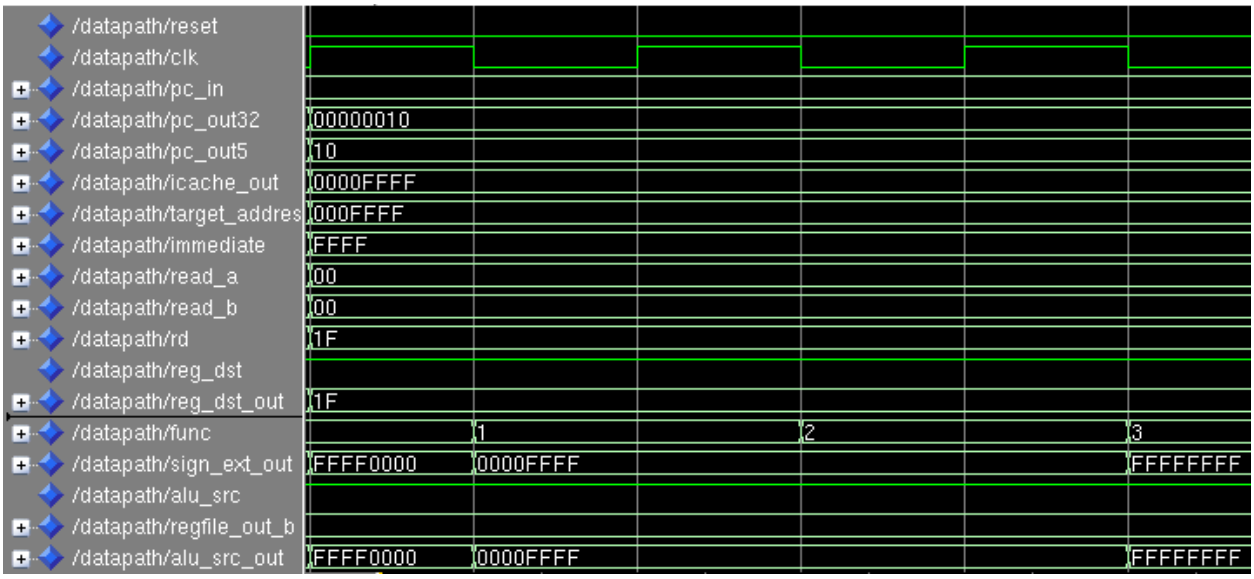

Figure 3. Sign extend testing

Figure 3 shows the sign extend functionality. The func signal determines what type of sign extension occurs. The sign extension is performed asynchronously. Load upper immediate, logical and regular sign extensions are performed properly.

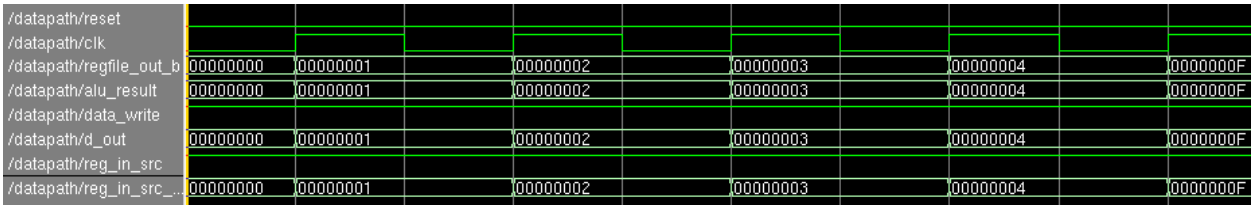

Figure 4. Data cache writing test

Figure 4 above shows the data cache writing test being performed. Sequential writes similar to the register file were performed. Regfile_out_b is the signal sent to the d_in of the data cache. The address is the lower 5 bits of the ALU_result. Reg_in_src is the control signal of the "reg_in_src_out" mux whose output is sent to din of the register file.

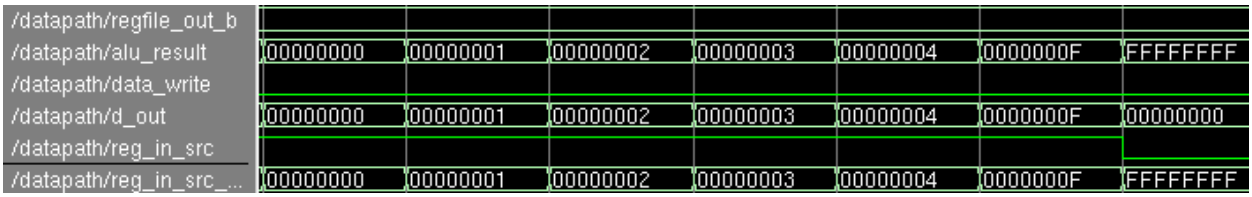

Figure 5. Data cache read test

Sequential reads performed asynchronously showing the previously written values with the reg_in_src mux being tested at the end. (Note: Since the data cache values are all initialized to hold zeroes, and since the 32nd data location was not written to, it has its default value of 0x00000000 as seen at the d_out when ALU_result is changed to 0xFFFFFFFF).

An issue was encountered in this testing phase. Initially, intermediary signals were used to hold the lower 5 bits of ALU_result and the rt output of the register file but this gave erroneous write values to improper data cache addresses. They were delayed by 1 clock cycle. Removing the signals and taking the values directly from the outputs of the ALU and rt solved the problem. Intermediary signals were not used in the register file, so the problem did not occur in that lab.

Conclusion:

There was some trouble involved with troubleshooting the data cache that proved time consuming. Using signals to connect component outputs to the d_in and dcache_address caused a delay in the data write giving erroneous values.

Appendix