CECS 440 Lab 5 Report - MIPS Datapath for R-type and I-type Instructions

Group members: Veronica Fuentes

Len Quach

1. Method of designing the datapath

PC is the starting point for all instructions and provides the instruction memory with the instruction address. The register operands used by an instruction are defined by fields of that instruction after it has been fetched. After fetching the register operands, they can be used to compute a memory address, compute an arithmetic result, or perform a compare. If the instruction is an arithmetic-logical instruction, the output of the ALU must be written to a register. If the operation is a load or store, the ALU result is used as an address to either store a value from the registers or load a value from memory into the registers. The ALU or memory's output is written back into the register file. ALU performance is needed for branches to decide the next instruction address, which can come from the ALU or from an adder that increases the current PC by 4.

2. HDL codes developed for each datapath component

There are 9 units to design the MIPS datapath: program counter, instruction memory, register file, ALU, data memory, control unit, ALU control unit, 32-bit adder, and multiplexers.

Program counter: is a 32-bit wide register which stores the address of the instruction currently being executed. It provides the address needed to fetch an instruction from the instruction memory. The instructions being fetched are available at the start of every clock cycle and will occur after the falling edge of the clock.

32-bit adder: is an adder that computes the sum of two 32-bit operands. The first adder calculates the next sequential instruction address by adding 4 to the address stored in the PC. The second adder computes the branch target address by adding the next sequential address (PC + 4) to the signed extended 16-bit immediate field from the instruction shifted two positions to the left.

Control unit: takes as input the most significant six bits of the 32-bit instruction output by the instruction memory. This part is the OPCODE field which specifies the type of operation encoded in an instruction. The circuit has several control lines as outputs, which supervise the operation of the various CPU components.

ALU control unit: The inputs to this ALU Control are the FUNCT field of the instruction (the least significant six bits) and the 2-bit ALUOp outputs from the main control unit. The 4-bit output line is determined by these two signals.

Multiplexers: There are four muxes to choose from among the various outputs produced by the various components that make up the CPU. A 5-bit wide 2 to 1 mux selects between the RD and RT fields of an instruction to choose a destination register in the register format. RegDst is the control line that governs the selection process. When an instruction is of the R-type, RegDst is asserted, and when an instruction is of the I-type format, it is deserted. Furthermore, the second register defined by the field RT and the signed extended 16-bit immediate from the instruction are two options for a 32-bit wide 2 to 1 mux to pick the second operand to the ALU.

3. Required modification on each component

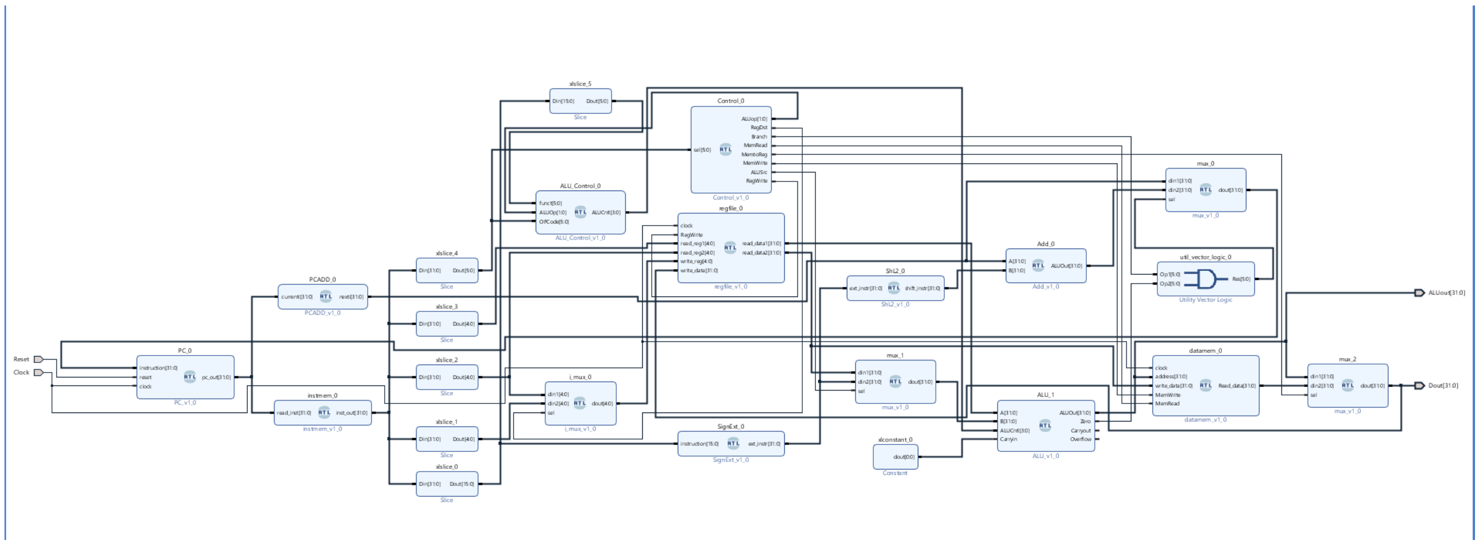
To add the functionality for the slt instruction, we add a 32-bit slt signal (R-Type instruction) to the ALU module. If the contents of the first source register (input A) are less than the contents of the second source register (input B), the slt instruction sets the destination register's content to 1. Otherwise, it's set to 0. Also, we added the ALUCntl signal for slt which is 0111.

In this lab, we added four modules for Shift left 2, Signed-extended, Add, and four multiplexers to the previous lab. In the ALU Control, we added more ALUOp and OPcode cases for I-type instructions, as well as more cases for I-type selectors in the Control Unit.

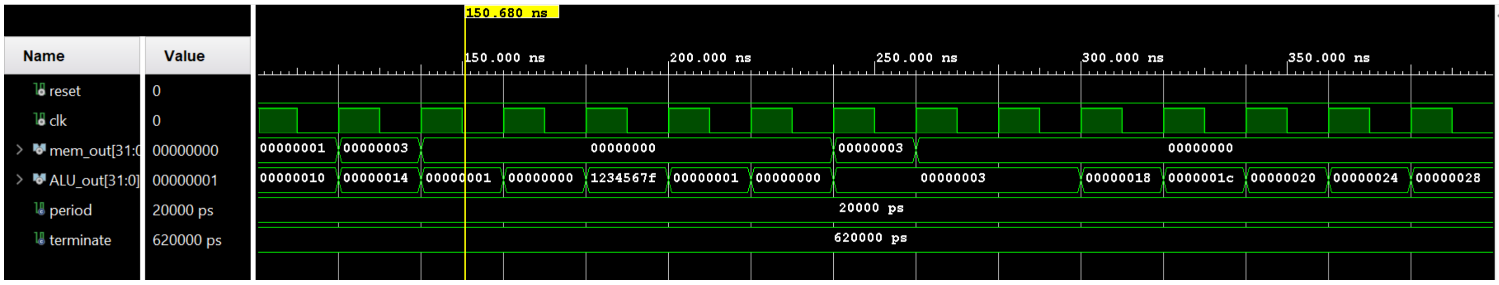
4. Testbench

We made 2 testbenches: alu_tb and Datapath_tb. alu_tb performs a NOR operation of two numbers A and B (32-bit wide each) with Carryin is 0 and produces ALUOut. Datapath_tb is built with a clock period of 20ns and runs for 1 clock cycle with reset = 1 and for 30 clock cycles with reset = 0. This testbench includes 2 input signals: reset and clock, and 2 output signals: mem_out and ALU_out (32-bit wide each). The program will test the 20 given instructions.

5. Snapshot of datapath design



6. Final simulation waveform



```
Name            Value
reset           0
clk             0
mem_out[31:0    00000000     00000001 | 00000003 | 00000000 | 00000003 | 00000000
ALU_out[31:0]   00000001     00000010 | 00000014 | 00000001 | 00000000 | 1234567f | 00000001 | 00000000 | 00000003 | 00000018 | 0000001c | 00000020 | 00000024 | 00000028
period          20000 ps     20000 ps
terminate       620000 ps    620000 ps
```