Processor Components: ALU, Memories, and control logic

Design

In this lab, I designed and tested more of the basic components of an ARM processor. I designed the following modules: ADD, ALU Control, ALU, CPU Control, Instruction Memory (read-only), Register File, and Data Memory. See the following figure:

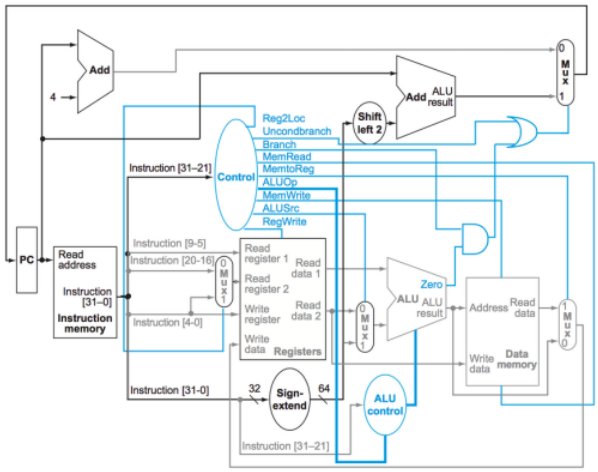

Figure 1: The ARM processor components designed in this lab

ADD

This adder design is a full-adder: The top-level module is ADD which has the sub-module: fullAdder which takes inputs from add and computes the sum.

ALUControl

ALU Control is a multiplexer that selects between different operations to perform based on the ALUOP and Opcode. The actual operations themselves are done in the ALU module.

ALU

As seen in Figure 1, the output operation word from ALU Control is sent to the ALU to do the computation of the inputs from the register file. The ALU module is a multiplier of multiplexers for the three outputs: ALU result, Zero, and Overflow. Based on the control line we perform AND, OR, Add, or Subtract.

CPU Control

The CPU Control is used to determine how we write to the register file, data memory, the ALU control input, and branching (conditional and unconditional). CPU Control is a 11:10 Mux.

Registers

This is the register file for the processor. It holds 32 registers: X0-XZR (X32). We have a write-enable signal as well and use a LUT. Writes are falling clock edge triggered, while reads are asynchronous to the clock. Also, note that we don't write to the XZR register which is hardcoded to '0'.

DMEM

This module was already provided. We changed the size of the memory to hold 1KB of data. I also initialized the Debug signal provided.

Results and Discussion

To test the components I designed, I wrote a testbench for each of them.

ADD:

I varied the two inputs to the adder over time and checked if the output was the correct sum:

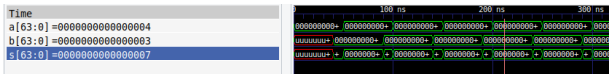

Figure 2: here $4 + 3 = 7$ as expected

ALU Control:

Here I just wanted to see if the ALU Control word, "operation", was correct based on the operation. I set the opcode to different words and checked to see if the operation was the one I wanted.

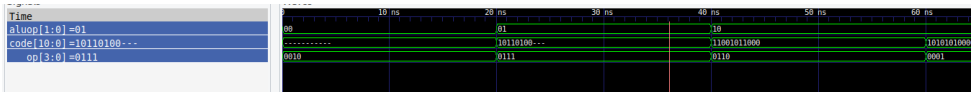

Figure 3: When the aluOp is 10 and the opcode is 1100101100, the operation is SUB as expected

ALU:

I tested this in the same way as ALUControl: I varied input to the mux and also varied the input signals and checked to see if the computation was correct and if the control flags were set correctly.

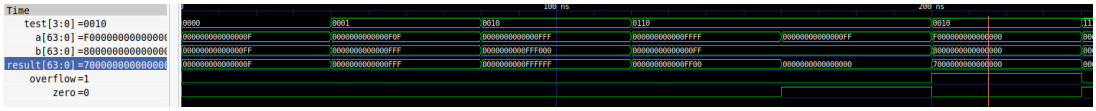

Figure 4: Here the select line is "0010" so do addition and you can see the overflow is set when you add $f + e$

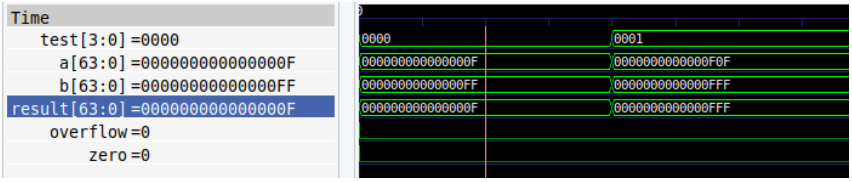

Figure 5: Here the select line is "0000" so do bitwise AND: 0f $\wedge$ ff = 0f

Registers:

Here I tested readings from memory and wrote to it separately. I varied by having regWrite on or off (allowing me to write or not) and trying to write under those conditions.

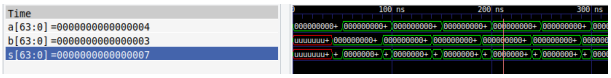

Figure 6: here $4 + 3 = 7$ as expected