ECE385 Experiment #6

Eric Meyers, Ryan Helsdingen

Section ABG; TAs: Ben Delay, Shuo Liu

March 9th, 2016

emeyer7, helsdin2

I. Introduction

The purpose of this lab is to create a very primitive processing unit designed around the Little Computer 3 (LC3) that was explored during previous ECE curriculum. This will be referred to the SLC3 Processing Unit throughout this lab. The SLC3 is a condensed version of the LC3 that allows user interfacing through memory-mapped I/O on board the Altera Cyclone IV SRAM Module, along with switches and LED indicators to show the user the status of the data within registers of the SLC3.

II. Description of Circuit

The circuit consists of several modules specifically the high level SLC3 module, the register file, the datapath, the Instruction Decoder/Sequencer Unit (IDSU), the Arithmetic and Logic Unit (ALU), many 16-bit registers (MAR, MDR, IR, and PC), several multiplexers, and some tristate buffers. The datapath and ISDU (Control) are shown in Figure 2. This Figure directly below this in the same section (Figure 3) is the memory interface of the SLC3.

All of these modules work together with one another to form the top level SLC3. The SLC3 will perform a total of

TODO

III. Purpose of Modules

As stated in the previous section there are many modules that work together in this system to form the top level SLC3. The following modules were created:

16-bit Shift Register

These modules are 16 inputs and 16 outputs with a load_enable line that determines if the data-in is sent to the data-out. The Instruction Register, Memory Address Register, Memory Data Register, Program Counter Register, and Register File will all be utilizing this module.

Multiplexers

A 16-bit 2-to-1 MUX will be used for the ADDR1MUX and the SR2MUX. A 3-bit 2-to-1 MUX will be used for the DRMUX and the SR1MUX.

A 16-bit 4-to-1 MUX will be used for the ADDR2MUX only.

A 16-bit 3-to-1 MUX will be used for the PCMUX. This will take inputs of the databus, the 16-bit adder output, and the PC+1.

Instruction Sequencer and Decoder

This module will contain the state machine for the entire SLC3. It will have the ability to implement 11 states and cycle through all of them in a cyclic fashion.

NZP

TODO

Datapath

TODO

Arthmetic and Logic Unit

ALUK	OUTPUT FUNCTION
00	A ADD B
01	A AND B
10	NOT A
11	PASS A

The ALU module does the brunt arithmetic work of the SLC3 machine. The ALU takes in two 16 values (A and B) and does one of four things with the data:

bitwise ADD, bitwise AND, bitwise NOT, or passes the A input through to the output terminal.

SEXT/ZEXT
The SEXT/ZEXT modules take in data that is M bits long and extend it to N bits long where M¡N by adding bits to the front of the input. ZEXT extends the input with purely zeros added to the front of it, whereas SEXT depends on the sign of the value. If the MSB=1, the input is negative and gets extended with the proper number of 1s. On the contrary, if MSB=0, the input is positive and gets extended with the proper number of zeros.

Register File
TODO

16-bit Adder
TODO

Tristate Buffer
The tristate buffer module is used to protect the bus from multiple signals entering the bus. The tristate buffer takes in gate signals determined by the state of the machine and sends the respective data to the proper databus input.

Tristate
TODO

Mem2IO
TODO

IV. State Diagram

The State Diagram can be found on Figure **??** in ”Section XI: Figures”.

V. Instruction Sequencer / Decoder

ERIC SECTION

VI. Schematic/Block Diagram

RYAN SECTION

The Schematic / Block Diagrams can be found on Figure **??** in ”Section XI: Figures”.

VII. Pre-Lab Simulation Waveforms

The Pre-Lab Simulation Waveforms can be found on Figure **??** in ”Section XI: Figures”.

VIII. Design Statistics

ERIC SECTION

IX. Post Lab

1.) What is MEM2IO used for, i.e. what is its main function?

MEM2IO acts as the middle-man between the data going to/from the SRAM and CPU and the I/O of the system. For this lab, the I/O includes the switches as input and the Hex displays as output.

2.) What is the difference between BR and JMP instructions?

While the BR branch instruction and the JMP jump instruction pushes the PC to a non-sequential location as a result, they are completely different functions in design. The JMP instruction is simple in design and function. It takes the value in BaseR and shoves it into the PC regardless of any other conditions. The BR instruction is conditional. It relies on the current values of n, z, and p. If any of the values match their equivalents in the status register, the branch is enabled and the value of PC is updated with the new address. If none of the values match, the branch instruction is ignored and the next instruction is called.

X. Conclusion

Week 1 was successful for the team as we finished the Fetch command with relative ease. The team had minor problems with understanding the datapath vs. databus, minimizing delay in data movement, and properly implementing test_memory.sv into the design for simulation and testing. Initially too many modules were created building a high enough delay to the point that signals were missing their proper clock cycles.

Week 2 gave the team much more trouble. Each function required some sort of debugging after the first failed attempt. The largest source of errors was improper connections between modules. Unfortunately, not all of the functions were able to be completely debugged in time for the lab section.

Beyond the problems experienced in lab, understanding the creation of the simple SLC3 processor in System Verilog is a valuable tool for a computer engineer moving foward.

XI. Figures

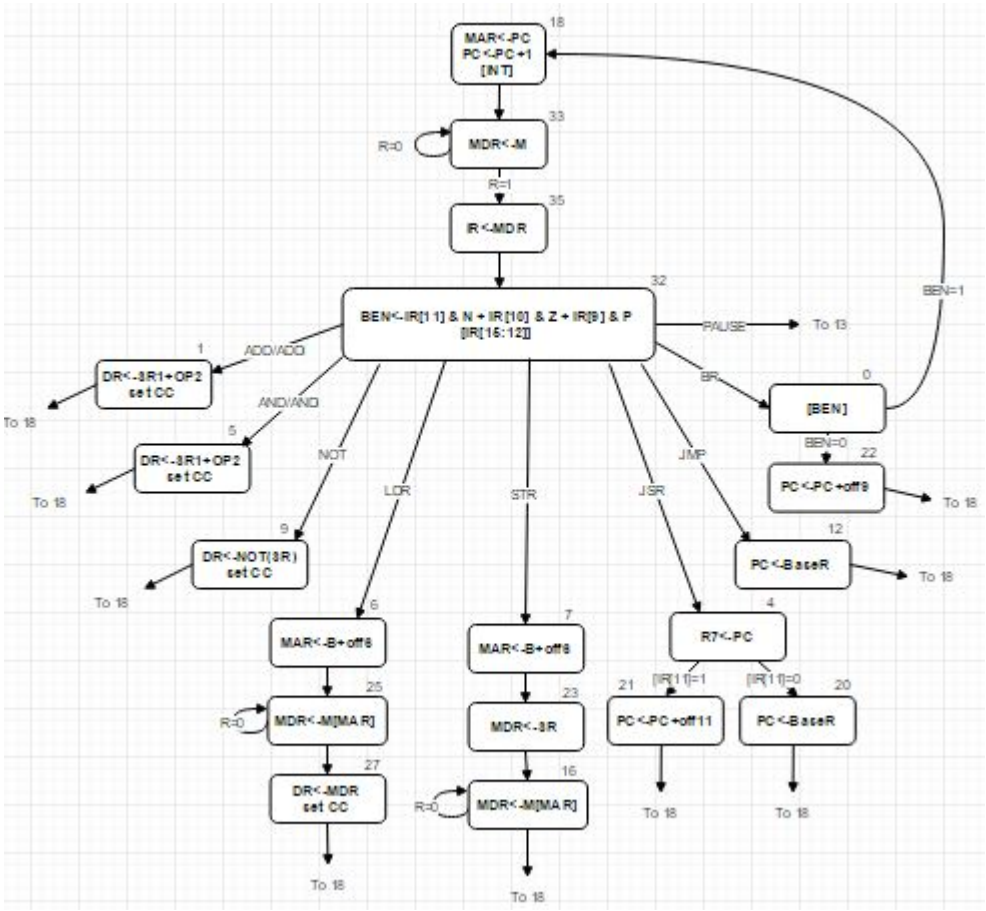

Fig. 1: SLC3 Machine State Diagram

Fig. 2: SLC3 CPU

Fig. 3: Memory, MAR, MDR, Mem2IO Configuration