**ECE 385**

Fall 2021

Experiment #5

**SLC-3 in System Verilog**

Steven Dimov & Owen Shin

Section ABE

TA: Abigail Wezelis

1. Introduction

a. Summarize the basic functionality of the SLC-3 processor

The objective of experiment #6 is to design the SLC-3 in system Verilog, which is a microprocessor subset of the original LC-3 ISA. Its program counter, instructions, and registers are all 16-bits. It can process basic machine language instructions for arithmetic, reading/writing to memory, and moving the program counter. It also has memory-mapped IO that can read the FPGA’s switches or control the HEX displays.

2. Written Description and Diagrams of SLC-3

a. Summary of Operation

It is able to perform eleven different operations, consisting of ADD, AND, NOT, BR, JMP, JSR, LDR, STR, and PAUSE. It does so by moving instructions and data from memory to the CPU, which can manipulate it through a combinational ALU or temporarily store it in general-use registers or the program counter. The CPU can store manipulated data back in the memory. Data travelling between most modules does so through a databus, which is contained in its own module. Mem2IO peripherals are controlled by a specific module that routes between the CPU and memory or a peripheral based on the address and read/write signal. Registers and memory operate synchronously on the clock rising edge, and other module are combinational.

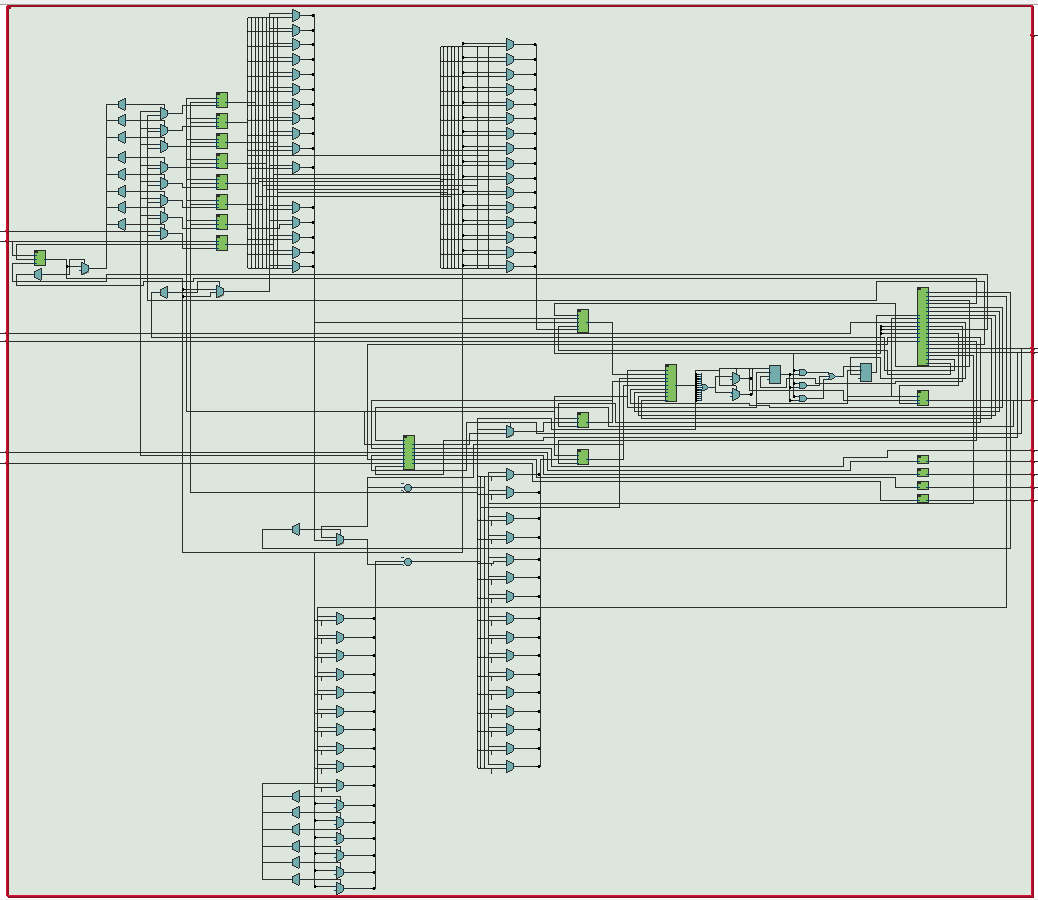
b. Describe in words how the SLC-3 performs its functions. You should describe the Fetch-Decode-Execute cycle as well as the various instructions the processor can perform.

The SLC-3 executes instructions through different states in an ISDU (Instruction Sequencer/Decoder Unit), which changes state based on previous state, the current instruction, or other signals. The ISDU sets signals for muxes, data loading, and databus control based on the current state. First, the SLC-3 fetches an instruction: the value in the PC (Program Counter) is loaded to the MAR (Memory Address Register) and that address is read in the memory. Like any other memory read, the value from memory is stored in the MDR (Memory Data Register). This value in the MDR is loaded to the IR (Instruction Register) while the PC value is incremented by one. This concludes the fetch step.

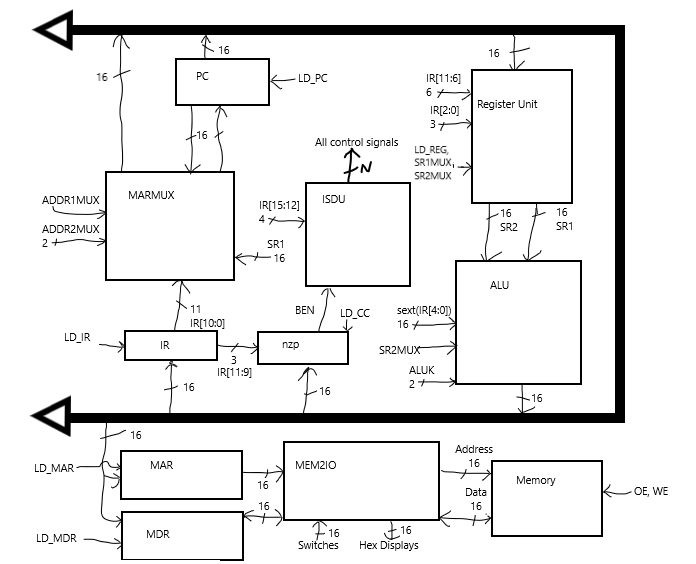
Next, the instructions are decoded. During this step, the BEN (Branch Enable) signal is set by comparing the nzp (negative-zero-positive) bits denoting the sign of the last arithmetic/logical operation or load from memory with the nzp bits in the IR. Then, the next state is determined by the four most significant bits of the IR (known as the opcode) and the execution of the instruction begins.

During execution, the ISDU moves through a sequence of states based on the given opcode. The 12 least significant bits of the IR determine which registers, address offsets, constants, or branch conditions will be used during the execution. During arithmetic/logical or load operations, the nzp register is set based on the relevant calculated/loaded value. At the end of an instruction being executed, the ISDU fetches the next instruction.

c. Block Diagram of slc3.sv



RTL viewer diagram of the slc3 module. Some registers or muxes were made in this module for debugging purposes.



Due to the arrangement of modules, the rtl diagram is mostly unreadable. This diagram is meant to clarify how the hardware specifically from our group works. The “datapath” module is the bold data bus in the diagram.

Modules:

HexDriver

inputs: [3:0] data

outputs: [6:0] hex

description: This module was provided by the 385 team. It uses combinational logic to output a seven-bit word based on a four-bit hexadecimal word.

purpose: The hex driver module translates the hex word into a word that makes it visually presentable on a seven-segment display.

Datapath

inputs: Clock, 4x [15:0] data, 4x select signals

output: [15:0] databus

description: This module synchronously outputs a 16-bit word from one of the four word inputs based on which select signal is high (one-hot).

purpose: The datapath module replaces the buffers in the original LC3, sending only one word at a time from the available words based on select signals from the ISDU.

f. Description of the operation of the ISDU (Instruction Sequence Decoder Unit)

i. Named ISDU.sv, this is the control unit for the SLC-3. Describe in words how the ISDU controls the various components of the SLC-3 based on the current instruction.

The ISDU, or the (Instruction Sequence Decoder Unit) serves to take in the 16-bit instructions and decode them appropriately so that the other modules can carry out their functions accordingly.

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

MEM2IO routes the memory writes and reads to/from either peripherals or the memory. Specifically, reads from the address 0xFFFF read the values of the switches and writes to the value 0xFFFF control the hex display. All other reads/writes access the memory at the given address.

• What is the difference between BR and JMP instructions?

The difference between the BR (branch) and JMP (jump) instructions are that the BR takes in a positive or negative address offset to add to the current PC value, while JMP sets PC to the value in a register. Also, BR is conditional based on the bits [11:9] of the instruction register ANDed with the nzp (negative-zero-positive) value of the previous arithmetic/logical operation or load operation.

• What is the purpose of the R signal in Patt and Patel? How do we compensate for the lack

of the signal in our design? What implications does this have for synchronization?

The R signal, or the ready bit is an input to the control unit from the memory during a read/write (between MDR and the memory itself). This indicates whether the read/write is complete, and the control unit only transitions from a read/write state once it is complete. This prevents the previous value of MDR being mistaken for the current value, as MDR may not be loaded immediately. It also serves to protect a signal to load MDR into memory not being held long enough, resulting in the write not being completed.

Since this signal is not available, the read/write states are copied into multiple sequential states that hold the memory access and the write signal, in the case of a write. At the end of a read, the signal to load MDR is pulsed to load the then-available data from memory. Even though the time it takes to read/write is not minimized and there’s a chance some reads/writes could not have enough time, for the most part the CPU will work with the memory.

5. Conclusion

a. Discuss functionality of your design. If parts of your design did not work, discuss what could be done to fix it.

The initial demonstration functionality of our design was limited. It was able to

b. Was there anything ambiguous, incorrect, or unnecessarily difficult in the lab manual or given materials which can be improved for next semester? You can also specify what we did right, so it doesn’t get changed.

Can the lab manual be changed to emphasize more that both demos are on hardware?