# ECE 385

Fall 2019

Experiment #6

# Simple Computer SLC-3.2 in SystemVerilog

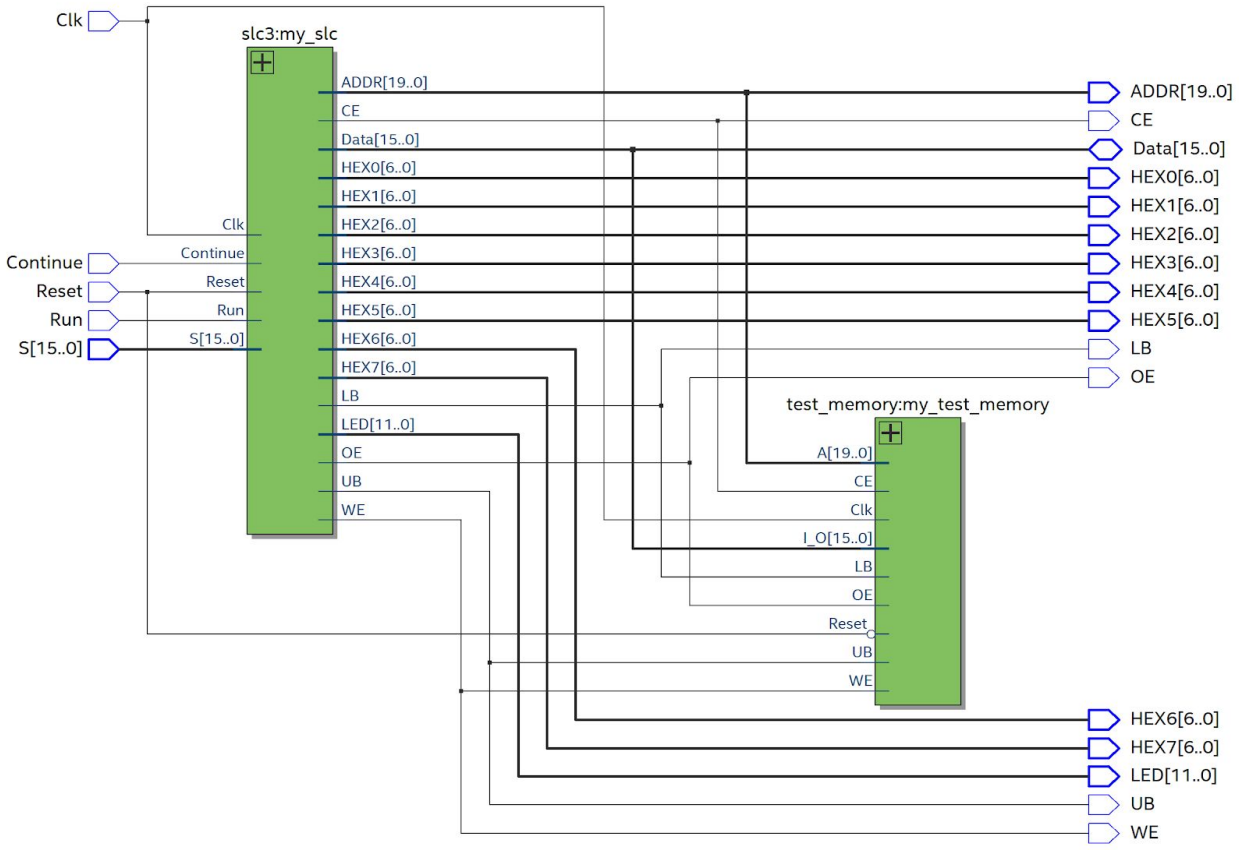
Eric Dong, Yifu Guo
ericd3, yifuguo3
Section AB3, Thursday 2pm
Gene Shiue

Introduction

In this lab, we designed a slightly altered version of the SLC3 processor. The processor is able to execute commands located in the SRAM chip. It is able to do addition, basic logical operations, load from memory, write to memory, handle conditions as well as change the PC based on the command.

Written Description and Diagrams of SLC-3

- The processor is able to pull data from the SRAM and execute the binary commands. These commands include ADD, AND, NOT, LDR, STR, BR, JSR, JMP.
- The LC3 processes its commands in 3 steps, Fetch, Decode, and Execute. In the Fetch state, the processor uses the value in the PC register as an address and stores that to MAR. Then M[MAR] is stored into MDR, which is then pulled into the IR register. To finish Fetch, the PC register is incremented. In the Decode stage, the first 4 bits of the IR is used to determine the instruction, then the rest of the bits are used to determine either which register to use or what offset is needed. Finally for the Execute stage, the operation is carried out, this can include sending the ALU which operation to execute or loading/storing to the SRAM.
-




| Module | Inputs | Outputs |
|---|---|---|

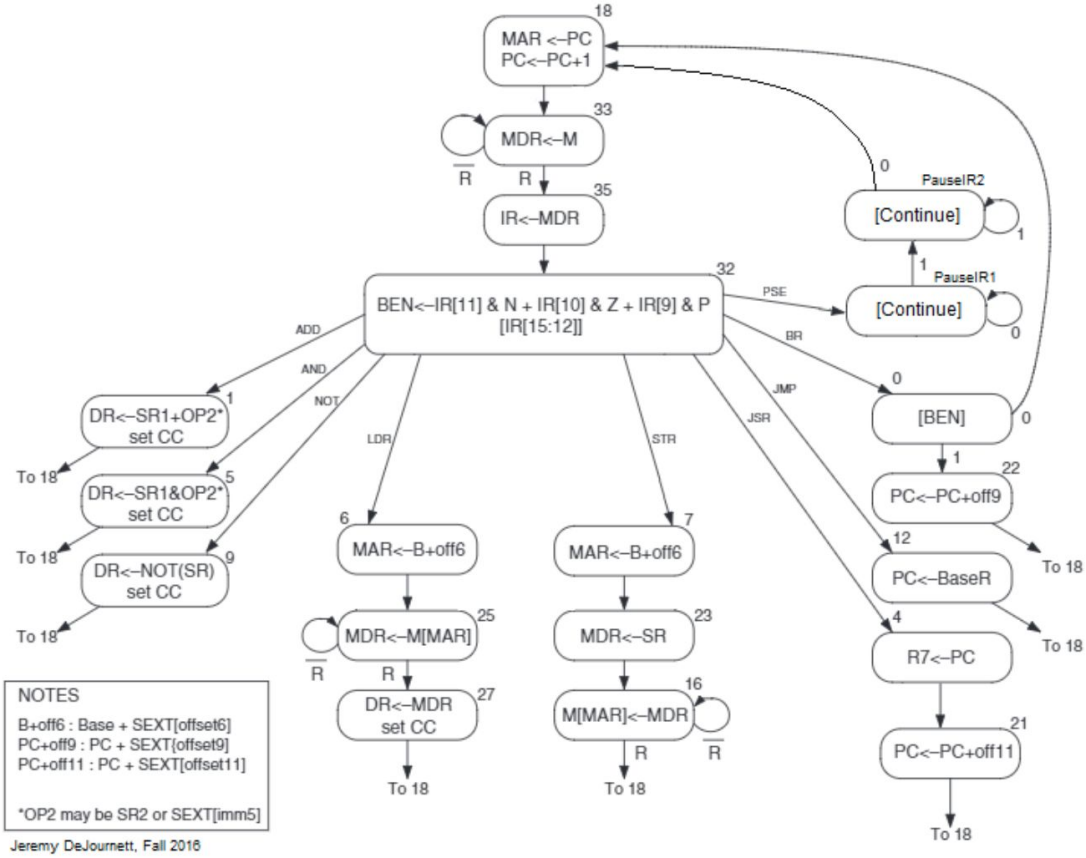
| | | |
|---|---|---|
| Mem2IO | Clk, Reset, ADDR, CE, UB, LB, OE, WE, Switches, Data_from_CPU, Data_from_SRAM, | Data_to_CPU, Data_to_SRAM, HEX0, HEX1, HEX2, HEX3 |
| ALU | A, B, aluk | ans |
| BEN | data_in, LD_CC, LD_BEN, clk, Reset, IR_in | BEN_out |
| datapath | GateMARMUX, GateMDR, GateALU, GatePC, Clk, Reset, LD_REG, LD_BEN, LD_CC, LD_IR, LD_MAR, LD_MDR, LD_PC, LD_LED, ADDR1MUX, SR2MUX, MIO_EN, SR1MUX, DRMUX, PCMUX, ADDR2MUX, ALUK, MDR_in | MDR_out, MAR_out, IR_out, PC_out, BEN, LED |
| HexDriver | In0 | Out0 |
| ISDU | Clk, Rest, Run, Continue, Opcode, IR_5, IR_11, BEN | LD_MAR, LD_MDR, LD_IR, LD_BEN, LD_CC, LD_REG, LD_PC, LD_LED, GatePC, GateMDR, GateALU, GateMARMUX, PCMUX, DRMUX, SR1MUX, SR2MUX, ADDR1MUX, ADDR2MUX, ALUK, Mem_CE, Mem_UB, Mem_LB, MEM_OE, Mem_WE |
| lab6_toplevel | S, Clk, Reset, Run, Continue, | LED, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7, CE, UB, LB, OE, WE, ADDR, Data |
| mux_21_dynamic | Sel, d0, d1 | out |
| mux_41_dynamic | Sel, d0, d1, d2, d3, | out |
| REGFILE | CLK, LDREG, Reset, bus, DR, SR1, SR2, | SR1_OUT, SR2_OUT |

| | | |
|---|---|---|
| slc3 | S, Clk, Reset, Run, Continue, | LED, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7, CE, UB, LB, OE, WE, ADDR, Data |
| sync | Clk, d | q |
| test_memory | Clk, Reset, I_O, A, CE, UB, LB, OE, WE | |
| Tristate | Clk, tristate_output_enable, Data_write | Data_read, Data |
| tristate_mux41 | Sel, d0, d1, d2, d3, | out |

- Mem2IO
  - Description: This module of made of several muxes and
  - Purpose: This allows data transfer between the SLC3 processor and the SRAM module. It also allows data input from the switches as well as data output to the hexdrivers which displays the data on the HEX leds.
- ALU
  - Description: This is an Arithmetic Logic Unit
  - Purpose: This unit takes in two inputs and returns an answer after processing those two inputs. The functions that we have implemented are ADD, NOT, AND and pass through one of the inputs. This allows basic arithmetic operations.
- BEN
  - Description: This is a Branch Enable unit for the instruction BR.
  - Purpose: This unit calculates whether an operation should take place given a number. With the given number, it determines whether the number is positive, zero or negative. If it matches with the conditions given in the BR instruction, then the PC is changed.
- Datapath
  - Description: This module describes the entire LC3 datapath layout.
  - Purpose: This module is used for data transfers and instruction decoding for the SLC3 processor. It contains all the individual modules such as muxes, registers, and interconnections in between.
- HexDriver
  - Description: This is a hex driver.
  - Purpose: It helps to present the output hex-value on the board.
- ISDU
  - Description: This is an FSM which controls which signals are enabled as well as which to disable. It controls what goes on in the SLC3 based on the instruction. It is connected to the different control signals on registers, muxes as well as the regfile.

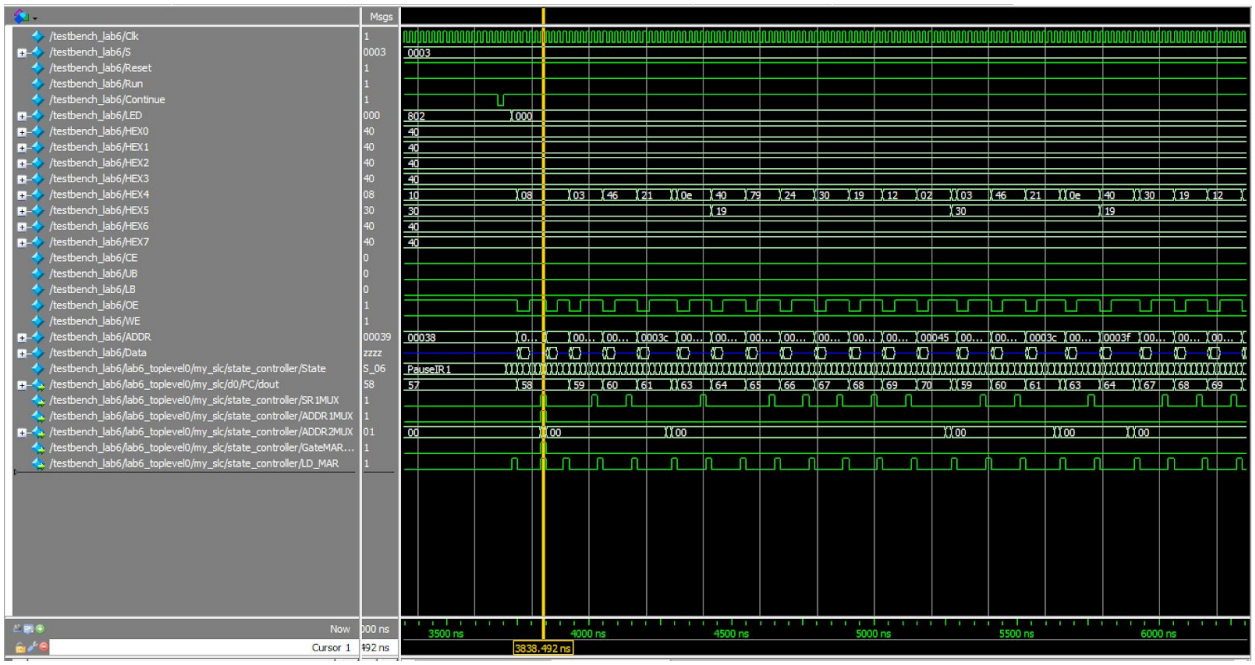
        - Purpose: For the ISDU, there are many states that it describes. For each state, the ISDU outputs different control signals in order to enable or disable some of the modules. For example for the ADD instruction, the ISDU controls which function the ALU executes and also which registers can write data to the bus and which register to store the answer to.
    - ~~Lab6_toplevel~~
        - Description: This is the top level of lab6.
        - Purpose: This combines the memory module with the SLC3 processor.
    - Mux_21_dynamic
        - Description: This is a dynamic 2 to 1 mux.
        - Purpose: Given 2 inputs of the same bit length, the control signal choses which one the mux should pass through.
    - Mux_41_dynamic
        - Description: This is a dynamic 4 to 1 mux.
        - Purpose: Given 4 inputs of the same bit length, the control signal choses which one the mux should pass through.
    - REGFILE
        - Description: This is a register file which contains 8 different 16 bit registers.
        - Purpose: This register file holds temporary values which the LC3 program can use. These registers can be used to hold values so that operations such as addition, store, load, write to display etc can be done.
    - ~~Slc3~~
        - Description: This is the layout of the entire SLC3 processor.
        - Purpose: this module is to simulate a similar behavior as a real lc3.
    - Sync
        - Description: this is a D flip-flop. The data will go from D to Q when it is on the positive edge of clock.
        - Purpose: we use this module as a synchronizer to negate the reset, execute and clearA_loadB signal synchronously.
    - Test_memory
        - Description: this is the top level of a simulating memory.
        - Purpose: this module is used to simulate on the computer and perform a similar behavior to the SRAM IC on the DE2 board.
    - Tristate
        - Description: this is a tristate buffer
        - Purpose: this is a tristate buffer connects with SRAM and Men2IO, this buffer can help to protect the data read from SRAM or write to SRAM
    - Tristate_mux41
        - Description: this is a 4 to 1 MUX
        - Purpose: we use this 4 to 1 MUX to control the signal of GateMARMUX, GatePC, GateALU and GateMDR

- State Diagram of ISDU

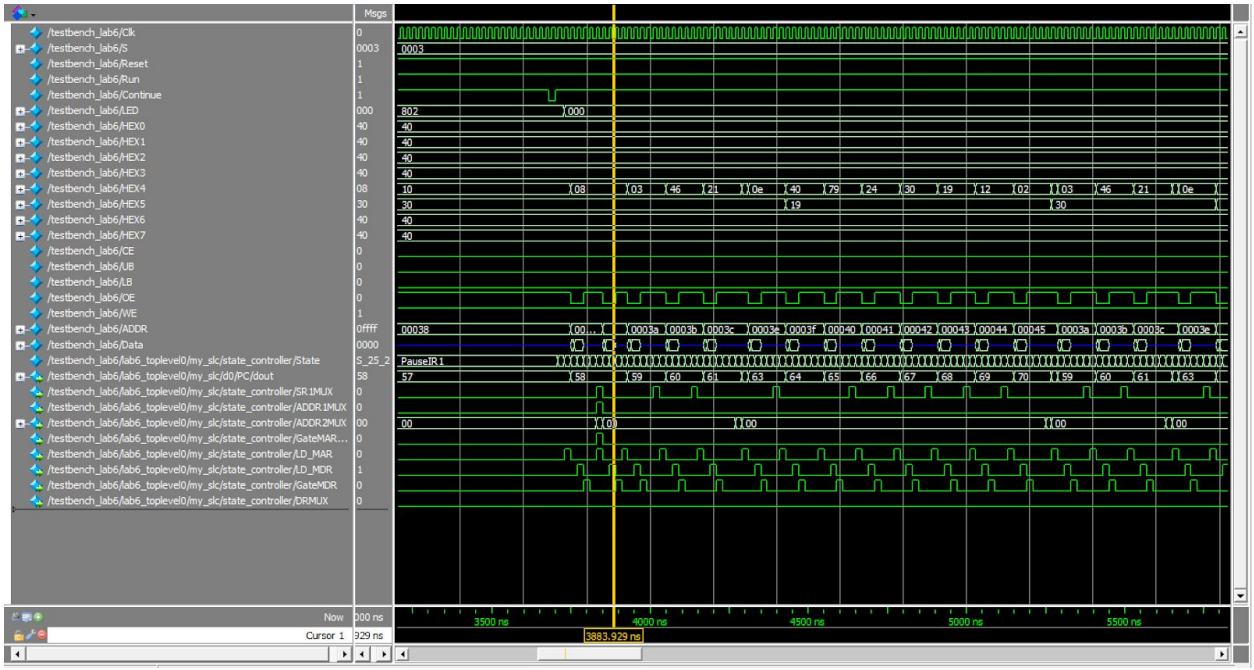



Simulations of the SLC3 Instructions

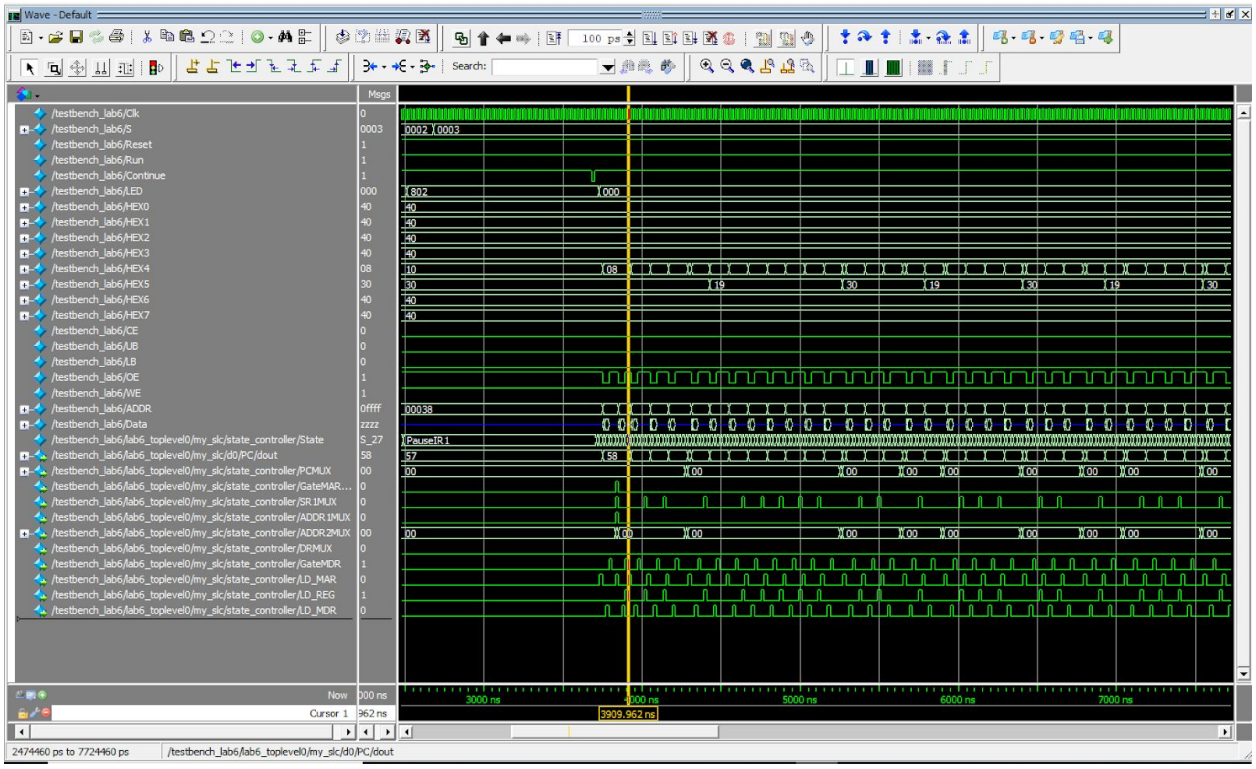
a. LDR(R2, R0, inSW):




From this waveform, we can see that the state is s_06, which is the first state of the operation LDR. also, the SR1MUX select bit is 1, which means now the lc3 is reading the IR[8:6] to select the SR1. what's more, the ADDR1MUX is 1 and ADDR2MUX is 01, which means the SEXT[5:0] + SR1. Also, the GateMAR and LD_MAR is 1, which means the SEXT[5:0] + SR1 goes to the MAR.

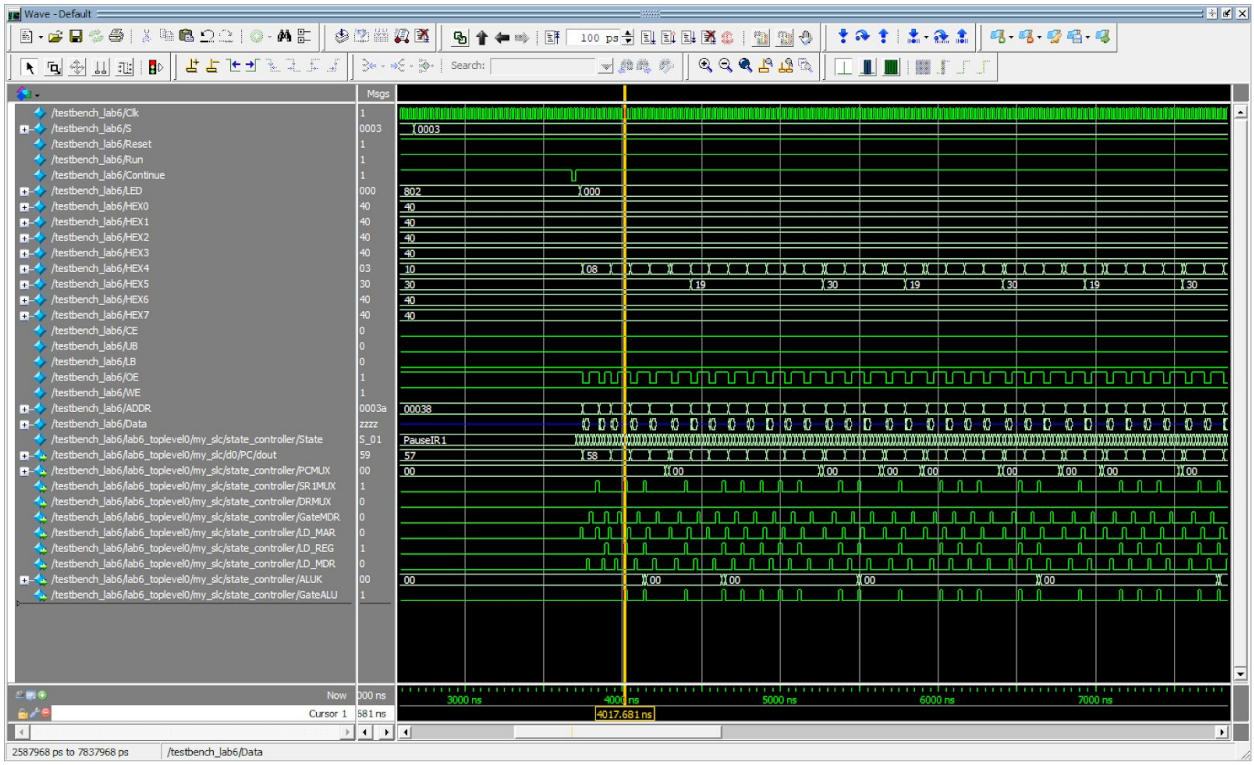



At this state, we can see the LD_MDR is 1and the OE is 0, which means the the M[MAR] is read to the MDR.



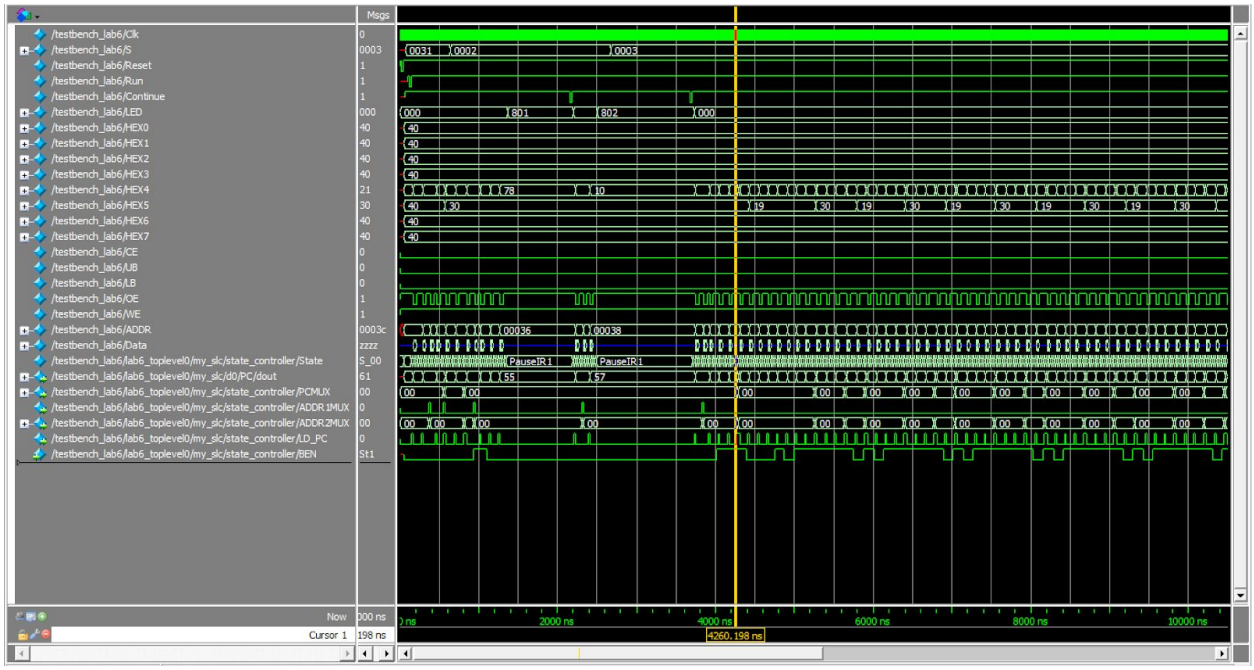

At this state, we can see that GateMDR is 1, which means the data from MDR is going to bus, and DRMUX is 0, which means the lc3 is read the IR[11:9] to decide the DR. also, LD_REG is 1, which means the data from bus is loaded to the DR.

b. ADD(R5, R5, R5) :

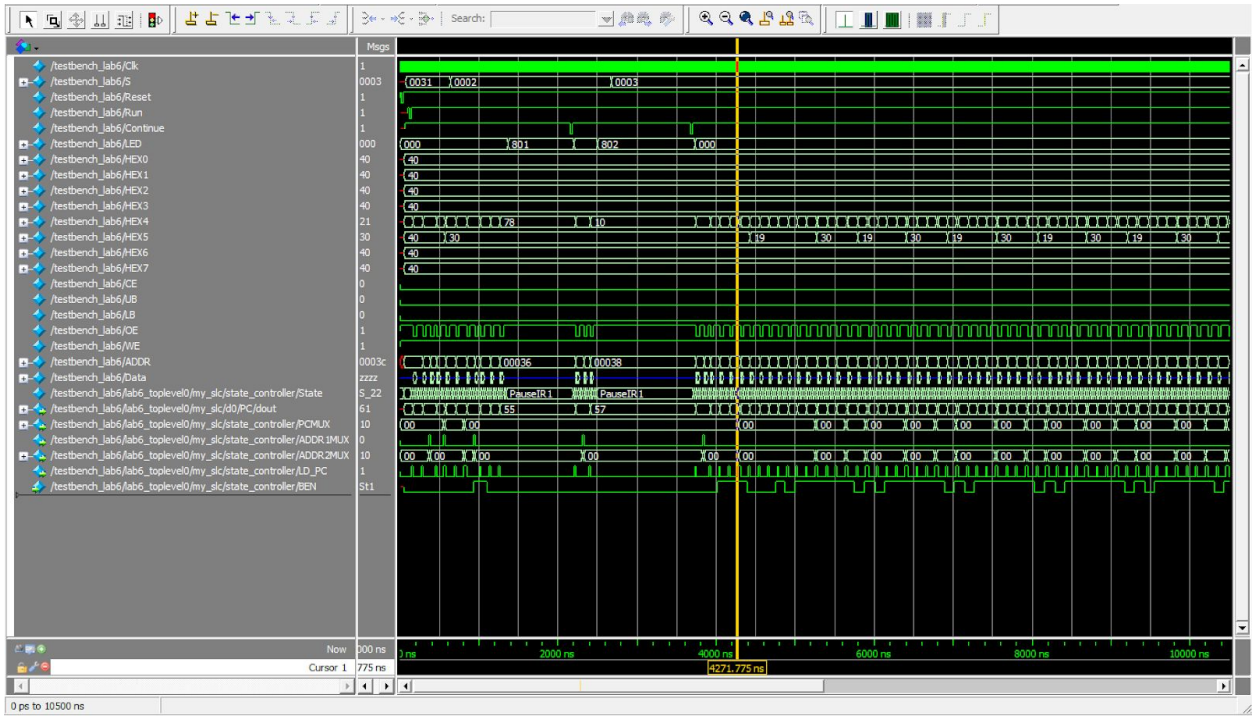



From the above waveform, we can see that the state is s_01, which is the state of ADD. we can see that SR1MUX is 1 and SR2MUX is 0, which means that two operands are both from registers. Also, the ALUK is 00 and GateALU is 1, which means two operands are added and the output goes to the bus. Meanwhile, we see the LD_REG is 1 and DRMUX is 0, which means the output data is stored to the selected DR.

c. BR(z, 1) :




From the above waveform, we can see that the state is s_00, which is the BR state. Because BEN is 1, so it will jump to the s_22 state.(showed in the following waveform)




At this waveform, the state comes to the s_22, and we can see that the LD_PC is 1 and PC_MUX is 10, which shows that the new PC is from the ADDR1MUX + ADDR2MUX. Compared to the old PC, we can see the new PC increments one, which is exactly the meaning of  BR(z, 1).

Design Resources:

| | |
|---|---|
| LUT | 735 |
| DSP | 0 |
| Memory | 0 |
| Flip-Flop | 283 |
| Frequency | 65.94 MHz |
| Static Power | 98.65 mW |
| Dynamic Power | 7.92 mW |
| Total Power | 178.55 mW |

Post Lab Questions:

1. MEM2IO is the bridge between the SRAM module and the SLC3 processor. It also allows us to read in the switch value as well as output data to the HEX display.
2. The JMP instruction changes PC without any condition while the BR instruction needs a condition to be met before the PC changes.
3. The R signal is used to make sure that the memory is ready. To compensate for that, we use two cycles for the state 33 so we have 33_1 and 33_2, two clock cycles to ensure that the memory is ready. The data is guaranteed to be correct so we only need a longer wait time and don't need synchronizers.

Conclusion

- Our project functions perfectly, everything works. During the process of building the project, we ran into several small roadblocks which includes the SRAM not being flashed perfectly as well as we were confused on some of the instructions.
- I think the lab manual should be split into two, one for week one and one for week 2 since I was really confused about what has to be done in week 1 and ended up doing much more than I needed. But the overall experience was really rewarding since I learned so much about LC3 that I didn't know before.