ECE 385

Fall 2022

Experiment #5

Lab 5

Simple Computer SLC-3.2 in System Verilog

Haocheng Yang((hy38)
Yicheng Zhou(yz69)

Introduction:

In this experiment, we designed an implemented a simple microprocessor using system Verilog. This is simplified version of LC-3 ISA, a 16-bit processor.

Written Description and Diagrams of SLC-3:

a.) Summary of operation:

a. initializing program on FPGA
b. select the function desired and enter the starting address of which in memory via switches 0-9
c. press ‘run’, the program will jump to starting address of desired function.
d. Enter user input via switches 0-9,
e. For dome functions, press ‘continue’ when a waypoint is reached, for others, program will loop/halt automatically.

b.) The SLC-3 perform its functions in three stages: “Fetch”- “Decode”- “Execute”. For Fetch stage, PC will point to next instruction’s memory address, then the instruction will be loaded into instruction register. During the decode stage, the encoded instruction will be interpreted by the decoder. Execute stage will pass the decoded information as a sequence of signals to the relevant units of the CPU, which includes ALU, register units, and CU. The result generated will be stored in main memory. The PC increments.

c.) The block diagram of slc3.sv is shown below:

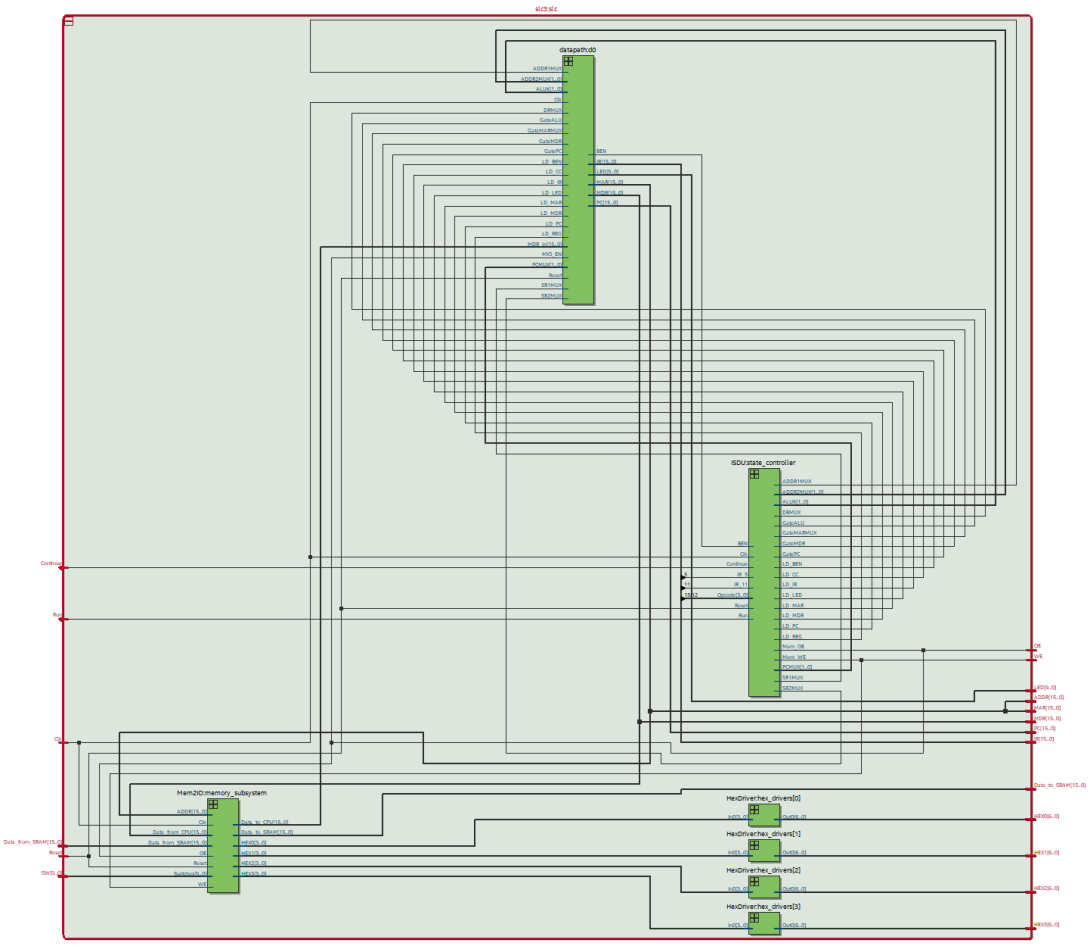


Figure 1, slc3.sv block diagram.

d.)Written description of all .sv modules:

1. test_memory.sv:
   Inputs: Reset, Clk, [15:0]data, [9:0] address, rden, wren
   Ouputs: [15:0]readout
   Description: This module is to simulate the behavior of the actual SDRAM that will be implemented in the full system.
   Purpose: This is used to test different system components before the whole system is operational.
2. synchronizers.sv:
   Inputs: Clk, d

Outputs: q
Description: this module is used to synchronize user input to signals in the program. It will load button inputs to various programs at every positive clock edge.
Purpose: mainly to ensure the normal operation of register units and supplementary functions.

3. SR2MUX.sv:
   Inputs: [15:0]IR, sr2_out,
   Outputs: [15:0] ALU_B,
   Description: this module is implemented as a unique case based 2:1 multiplexer.
   Purpose: the is used to implement the source register 2 mux in the lc3 cpu.
4. SR1MUX.sv:
   Inputs: SR1MUX, [15:0]IR
   Outputs: [2:1] SR1MUX_out
   Description: this module is implemented as unique case based 2:1 multiplexer
   Purpose: this is used to implement the source register1 mux in the lc3 cpu.
5. slc3_testtop.sv:
   Inputs: [9:0]SW, Clk, Run, Continue,
   Outputs: [9:0] LED, [6:0] HEX0, [6:0] HEX1, [6:0] HEX2, [6:0] HEX3, [15:0]PC, [15:0]MDR, [15:0] IR, OE, WE.
   Description: this is the top level entity written to incorporate the simulation testbench.
   Purpose: this is used to call slc3.sv and act as the I/O of LC3 while getting inputs from the digital testbench.

6. slc3_sramtop.sv:

   Inputs: [9:0]SW, Clk, Run, Continue,

   Outputs: [9:0] LED, [6:0] HEX0, [6:0] HEX1, [6:0] HEX2, [6:0] HEX3, [15:0]PC, [15:0]MDR, [15:0] IR,

   Description: this is the top level entity written to call on all function in service of the simulation of FPGA board as well as getting inputs from the FPGA borad.

   Purpose: this is used to call on slc3.sv and act as the I/O of LC3 while getting inputs from the FPGA board and displaying outputs to the FPGA board.

7. SLC3_2.sv
   Inputs: N/A
   Outputs: N/A
   Description: this acts like the compiler for LC3, which translates the assembly language back to 16-bit machine language.
   Purpose: to translate the content in memory into decodable 16-bit data for the CPU
8. slc3.sv:

   Inputs:[9:0] SW, Clk, Reset, Run, Continue, [15:0] Data_from_SRAM
   Outputs [9:0] LED, [15:0] PC, [15:0] MAR, [15:0] IR, [6:0] HEX0, [6:0] HEX1, [6:0] HEX2, [6:0] HEX3, [15:0]ADDR, [15:0]Data_to_SRAM.
   Description: This is the module in which all function “unite”, both to the testtop module will call on this module and pass down user inputs.

Purpose: this is the actual ‘top level entity’ for the system if we ignore the I/O, used to call on all functions and modules of the system to implement the LC3 CPU.

9. REGFILE.sv:
   Inputs: Clk, Reset, LD_REG, [2:0] DRMUX_out, SR1MUX_out,[15:0] IR, BUS,
   Outpus: [15:0] ALU_A, SR2_out
   Description: we implemented register 0-7 in LC3 here. These are 16-bit register controlled by Reset, L3_REG signals. Read is effectively enabled at all time.
   Purpose: this is the register unit in LC3, all load, store, and other operation must go through this module. The SR1, SR2,DR multiplexer will specify their respective register and signal the register to operate ‘load’ or ‘read’
10. PCMUX.sv:
    Inputs:[15:0]PC_PP,BUS.MARMUX, [1:0] PCMUX,
    Outputs: [15:0] PC_in
    Description: this module will call on allMUX.sv to implement a 4:1 multiplexer for PC mux function.
    Purpose: this is a 4:1 mux used to determine PC inputs, where PC_PP is the increment, or if load pc is enabled, this function will output BUS or MARMUX
11. PC.sv:
    Inputs: Clk, Reset, LD_PC, [15:0] PC_in,
    Outputs: [15:0] PC, PC_PP

Description: this module calls on the reg_16 module to implement a 16-bit register as the PC register. This program also implemented the natural increment of PC as the PC_PP as inputs for the PCmux, which in turns directly decides the next PC value.
Purpose: the program counter, which essentially is which memory address the system will get the next instruction from.

12. MIOEN.sv:
    Inputs: MIO_EN, [15:0] MDR_In, BUS
    Outputs: [15:0] MIO_out
    Description: this is a unique case implemented gate between bus and MIO.
    Purpose: this is to control when and if to load the content of MIO to the BUS
13. Memory_content.sv:
    Inputs: N/A
    Outputs: N/A
    Description: this is the memory of the system, and the instruction used to run tests are loaded into this module.
    Purpose: the slc3 will read the content of this module, combined with SLC3-2.sv, the instruction can be passed to fetch phase and start executing.
14. Mem2IO.sv:

    Inputs: Clk, Reset, OE, WE, [9:0] Switches, [15:0] ADDR, [15:0] Data_from_CPU, Data_from_SRAM,

Outputs: [3:0] HEX0, HEX1, HEX2, HEX3, [15:0] Data_to_CPU, Data_to_SRAM,

Description: this, as the name suggests is the IO for CPU and SRAM. This will pass memory content, user inputs, and CPU outputs to the correct locations.

Purpose: this module is used to pass information from CPU to SRAM and from SRAM to CPU.

15. MDR.sv:
    Inputs: Clk, LD_MDR, [15:0] MIO_out,
    Outputs: [15:0] MDR
    Description: this will read the data stored in the designated register.
    Purpose: this module is used to store data required for computation from source register
16. MARMUX.sv:
    Inputs: ADDR1MUX, [1:0] ADDR2MUX, [15:0] PC, IR, ALU_A,
    Outputs: [15:0] MARMUX
    Description: this module contains 2 sub-modul: ADDR1MUX and ADDR2MUX. Addr1mux is implemented as unique case 2:1 multiplexer and addr2mux is implemented as unique case 4:1 multiplexer
    Purpose: This module takes 2 control signal and 3 inputs to determine the correct value to load in to MARMUX.
17. MAR.sv:

Inputs: Clk, LD_MAR, [15:0] BUS,
Outputs: [15:0] MAR
Description: a 16-bit register
Purpose: to implement the memory address register, the content to be loaded will first be loaded into BUS and then loaded in to MAR.

18. ISDU.sv:
    Inputs: Clk, Reset, Run, Continue, [3:0] Opcode, IR_5, IR_11, BEN
    Outputs LD_MAR, LD_MDR, LD_IR, LD_BEN, LD_CC, LD_REG, LD_PC, LD_LED, GatePC, GateMDR, GateALU, Gate MARMUX, [1:0] PCmux, DRmux, SR1MUX,SR2MUX,ADDR1MUX,[1:0]ADDR2MUX, ALUK, MEM-OR, MEM-WE
    Description: this is the state machine that controls the operation of the execute phase. This module will take in the opcode and relevant information and issue control signal to all relevant unit according to state.
    Purpose: This module is the control unit for the SLC-3
19. IR.sv:
    Inputs: Clk, LD_IR, [15:0] BUS,
    Outputs: [15:0] IR
    Description: a 16-bit register
    Purpose: this is the instruction register, at fetch phase, the instruction in binary will be loaded into the bus and then loaded in to IR for future use.
20. Instantiateram.sv:

Inputs: Reset, Clk
Output: wren, [15:0]ADDR, data
Description: this contains the instantiated memory content.
Purpose: This is where the data from memory ‘meets’ SLC3-2.sv and translates into binary machine language for program execution.

21. HexDriver.sv:
    Inputs: [3:0]In0
    Outputs:[6:0] Out0
    Description: This module converts 4bit-binry input and convert it to 7 bit for hex-display.
    Purpose: This module converts 4bit biary numbers from registers into hexadecimal so LEDs can display outputs.

22. DRMUX.sv:
    Inputs: DRMUX, [15:0]IR
    Outputs: [2:0] DRMUX_out
    Description: an unique case implemented 2:1 multiplexer
    Purpose: using drmux logic to decide which input should be outputted.
23. Datapath.sv:
    Inputs: Clk, Reset, LD_REG, LD_PC, LD_MDR, LD_MAR, LD_IR, LD_BEN, LD_CC, LD_LED, GatePC, GateMDR, GateALU, GateMARMUX, [1:0] ALUK, PCMUX, ADDR2MUX, DRMUX,

SR1MUX, SR2MUX, ADDR1MUX, MIO_EN, [15:0] MDR_In,
Outputs: BEN, [9:0] LED, [15:0] IR, PC, MAR, MDR
Description: the is a hub where signals get passed around.
Purpose: although in theory we can skip this module and connect the ports directly between function modules, but implementing a ‘hub’ such that all signals/ports are located makes debugging much easier.

24. BUS.sv:
    Inputs: [15:0] PC, ALU, MDR, MARMUX, GateMARMUX, GatePC, GateALU, GateMDR,
    Outputs: [15:0] BUS
    Description: the data bus that load in values form different registers according to gate value. The output will go into high impedance mode if multiple gate are opened simultaneously.
    Purpose: this module function as the 16-bit data bus for the system, basically a general purpose shared data space.
25. BEN.sv:
    Inputs: Clk, Reset, LD_BEN, LD_CC, [15:0] IR, BUS,
    Outputs: BEN
    Description: this is a logic circuit to enable branch enable condition.

Purpose: this is a module when called, will run through the branching condition data which are loaded based on CC and IR.

26. ALU.sv:
    Inputs: [1:0] ALUK, [15:0] ALU_A, ALU_B,
    Outputs: [15:0] ALU
    Description: a bit wise ALU
    Purpose: the mode of ALU is selected via ALUK, this module is used for bit-wise computation.
27. allMUX.sv:
    Inputs: Select, [15:0] In_0, In_1,
    Outputs: [15:0] MUX2_out
    Description: there are 3 modules integrated within this file. 3 general purpose mux are implemented, a 2:1 mux, 4:1mux, 8:1 mux.
    Purpose: this module is for the general design of different types of mux(s) used in the system, and to be called on for different mux usage.
28. 16_bit_register.sv:
    Inputs: Clk, Reset, Load, [15:0] D_in,
    Ourputs: [15:0]D_out
    Description: a basic design of a 16-bit register
    Purpose: this module is for the general design of a 16-bit register and to be called on for different register usage.

e.) Description of the operation of the ISDU

the 16bit instruction are stored in IR for decoding, each containing and op code and operand. The ISDU will select the correct state according to opcode and issue command signals to the function units according to state & transition.

The state diagram of ISDU is as follows:

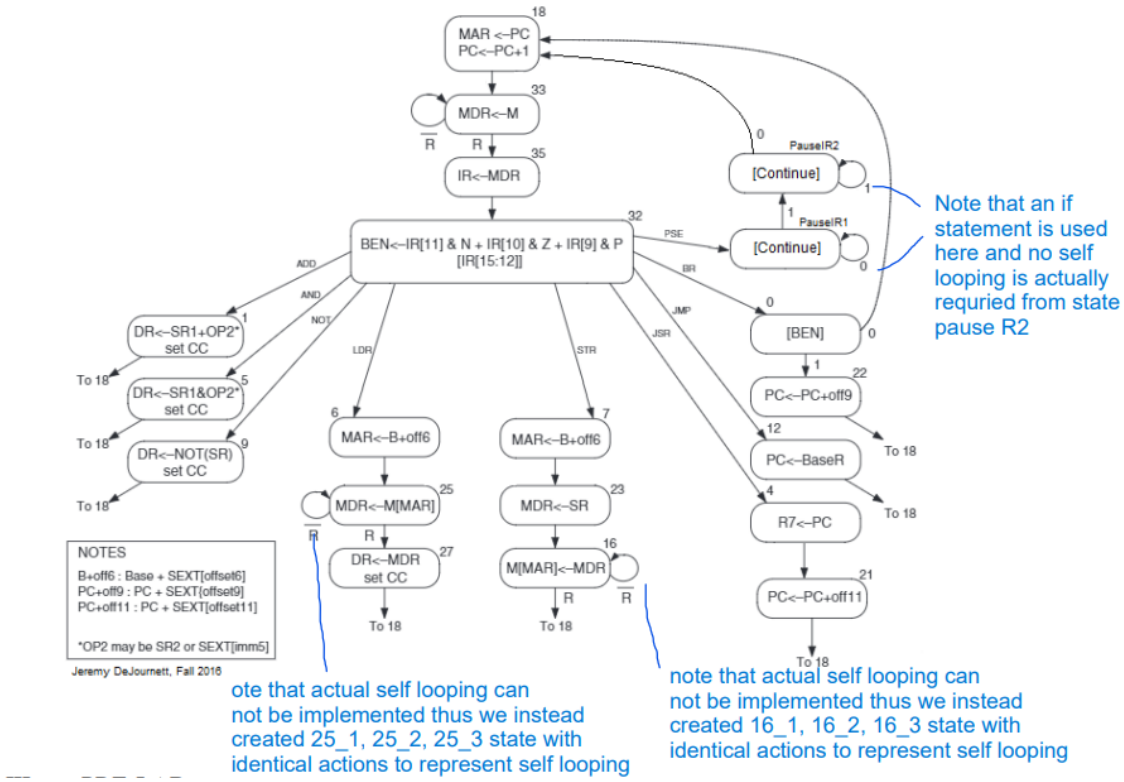
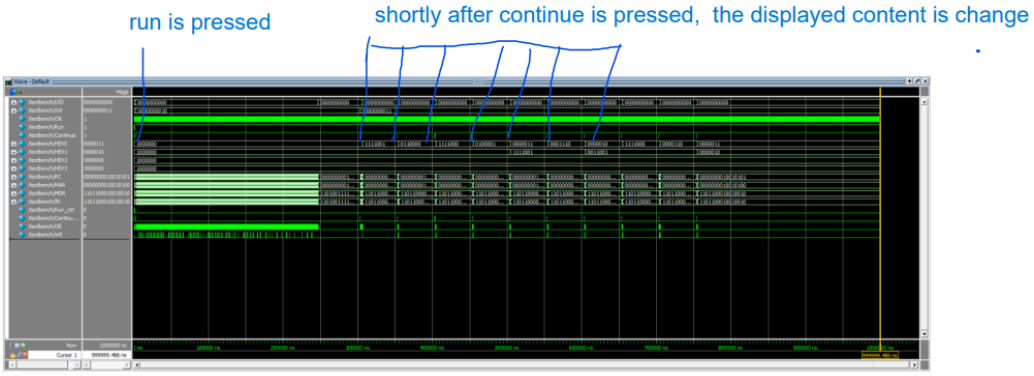


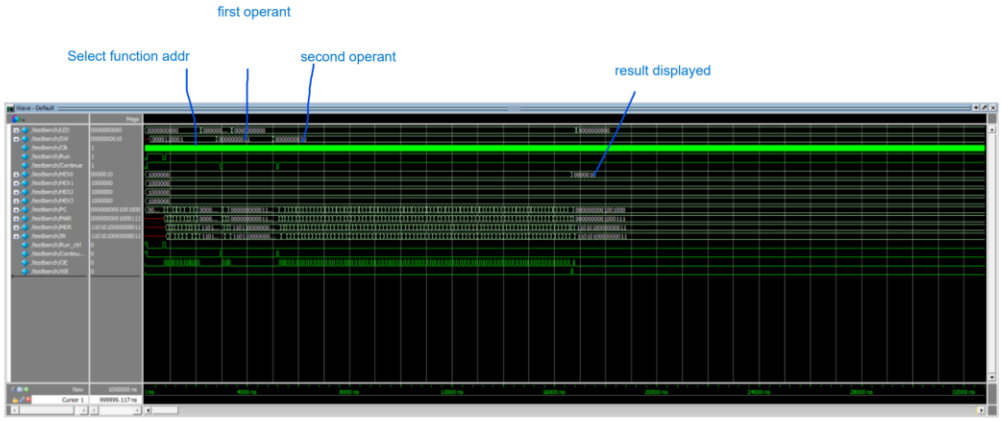
Figure 2, state diagram of ISDU.

Simulations of SLC-3 instructions:

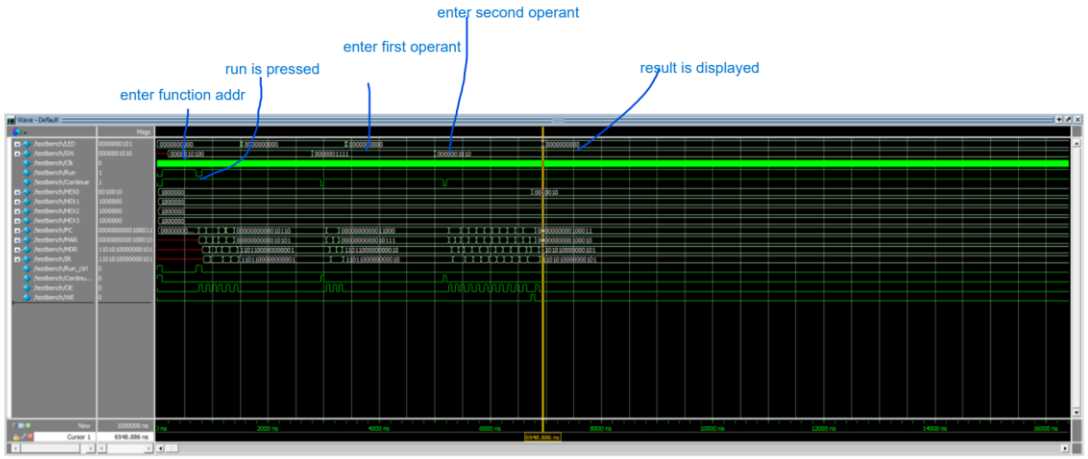
Sort:



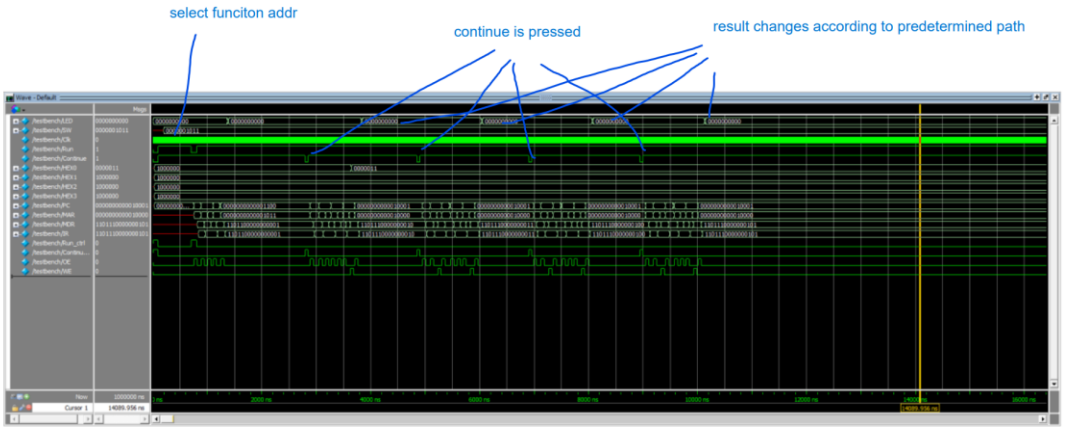
MULT:



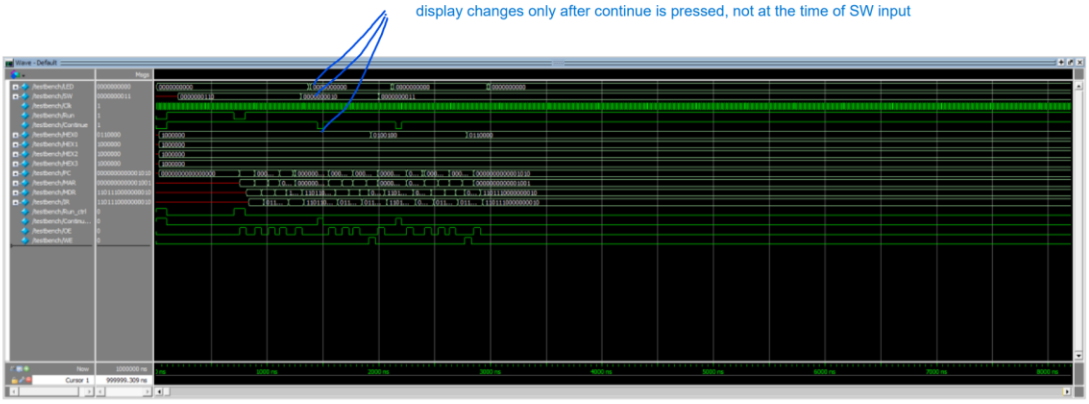
XOR:



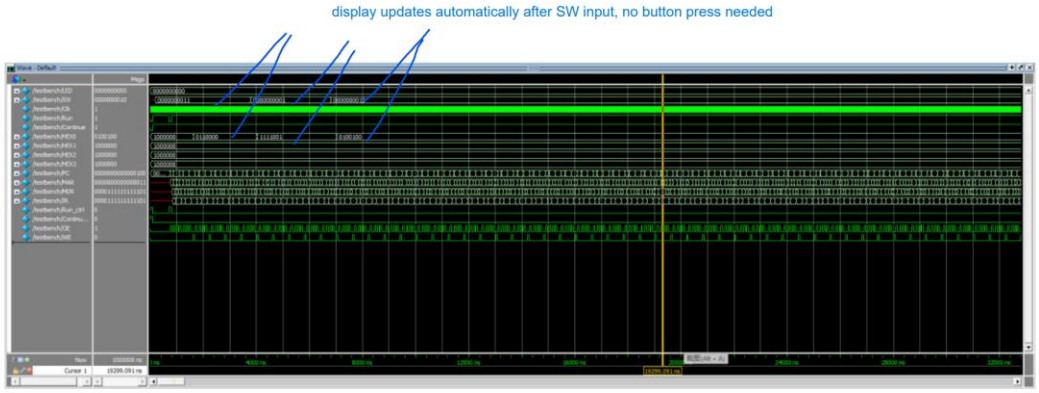
Self Mod Code:



I/O test2:



I/O test1:



Post-Lab Questions:

A.

LUT	838
DSP	0
Memory (BRAM)	16384 bits
Flip-Flop	269
Frequency	94.7 MHz
Static Power	89.94mW
Dynamic Power	0
Total Power	98.92mW

B. 1. What is MEM2IO used for:

MEM2IO is used for passing data between memory and CPU. And also gather user inputs from SW.

2. What is the difference between BR and JMP?

JMP tend to unconditional where BR is conditional,

3. 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 signal R means memory ready, which is used to make the state machine stay at the correct stage until memory is ready and loaded. Due to the lack of such signal, for each self looping state we have to ‘assume’ the amount of time (times of self looping) and implement accordingly, for example, for state 16, we assumed that the memory will be ready in 3 cycles thus in reality we implemented 3 identical state 16: 16_1, 16_2 ,16_3. The implications this have for synchronization is that instructions involving R signal tend to take longer to complete.

Conclusion:

Our design functioned as expected. However one obstacle we

Encountered is that we tried to implement datapath.sv during week 1. Due to the system being incomplete, a lot of port does not exist, and the design can’t compile. We tried to fix this by creating tempory ports and directly assigning values to them, which caused the signal BUS going into high-inpedance mode. The rest of the design process went smoothly.

The instruction given are clear enough that a student can be reasonably expected to complete the lab with given instruction.