ECE 385 Lab Report 5

Experiment #5:

Simple Computer SLC-3.2 in SystemVerilog

Spring 2022

Trusha Vernekar & Soham Manjrekar

Dohun Jeong (DJ)

tnv2 & sohammm2

Introduction

For Lab 5.1 and 5.2, we had to implement an SLC-3 Processor using SystemVerilog. This is a microprocessor which is a subset of LC-3 ISA, having 16-bit Program Counter (PC), 16-bit Instructions, and Registers. This processor was able to perform several operations using a certain number of opcodes, which were a subset of the LC-3 ISA. It followed the same Datapath and the State Machine diagram as an LC-3 Processor except without the logic and states for the TRAP instructions. For Lab 5.1, we had to implement the FETCH phase. For lab 5.2, we had to implement the DECODE and EXECUTE phases. After creating all the 3 phases, our SLC-3 microprocessor was complete.

Written Description and Diagrams of SLC-3

Our SLC-3 microprocessor had 3 main phases that each instruction would have to go through. As mentioned earlier, these were FETCH, DECODE and EXECUTE. The FETCH and DECODE phase was common for all instructions. In the FETCH phase, the instruction is identified and taken from memory. The way the processor does this is by performing 4 steps:

1. MAR $\leftarrow$ PC $\Rightarrow$ The MAR stores the address that contains the instruction to be performed
2. MDR $\leftarrow$ M(MAR) $\Rightarrow$ MDR will now store the instruction to be performed
3. IR $\leftarrow$ MDR $\Rightarrow$ IR stores the instruction that needs to be decoded before performing
4. PC $\leftarrow$ PC + 1 $\Rightarrow$ PC is incremented to perform the next instruction

The next phase that each instruction must go through is the DECODE phase. In this phase, the instruction that is stored in IR (during the FETCH cycle) is put through the instruction decoder. In this phase, the control signals required for the instruction in the IR are set, for example, the select signals for the muxes or the load signals for the registers.

The final phase that every instruction must go through is the EXECUTE phase. In this phase, the operation based on the opcode in the instruction is performed depending on the control signals set in the DECODE phase. The result is then stored in memory or register, depending on the instruction. Our microprocessor could perform 8 instructions, shown below.

Instruction	Instruction(15 downto 0)	Operation
ADD	0001 DR SR1 0 00 SR2	R(DR) ← R(SR1) + R(SR2)
ADDi	0001 DR SR 1 imm5	R(DR) ← R(SR) + SEXT(imm5)
AND	0101 DR SR1 0 00 SR2	R(DR) ← R(SR1) AND R(SR2)
ANDi	0101 DR SR 1 imm5	R(DR) ← R(SR) AND SEXT(imm5)
NOT	1001 DR SR 111111	R(DR) ← NOT R(SR)

BR	0000 n z p PCoffset9	if ((nzp AND NZP) != 0)
		PC ← PC + SEXT(PCoffset9)

JMP	1100 000 BaseR 000000	PC ← R(BaseR)

JSR	0100 1 PCoffset11	R(7) ← PC;
		PC ← PC + SEXT(PCoffset11)

LDR	0110 DR BaseR offset6	R(DR) ← M[R(BaseR) + SEXT(offset6)]
STR	0111 SR BaseR offset6	M[R(BaseR) + SEXT(offset6)] ← R(SR)
PAUSE	1101 ledVect12	LEDs ← ledVect12; Wait on Continue

Table 1: SLC-3 ISA

Once the EXECUTE phase is completed for one instruction, the processor loops back to the FETCH cycle for the next instruction stored in the updated Program Counter.

Block Diagram of slc3.sv

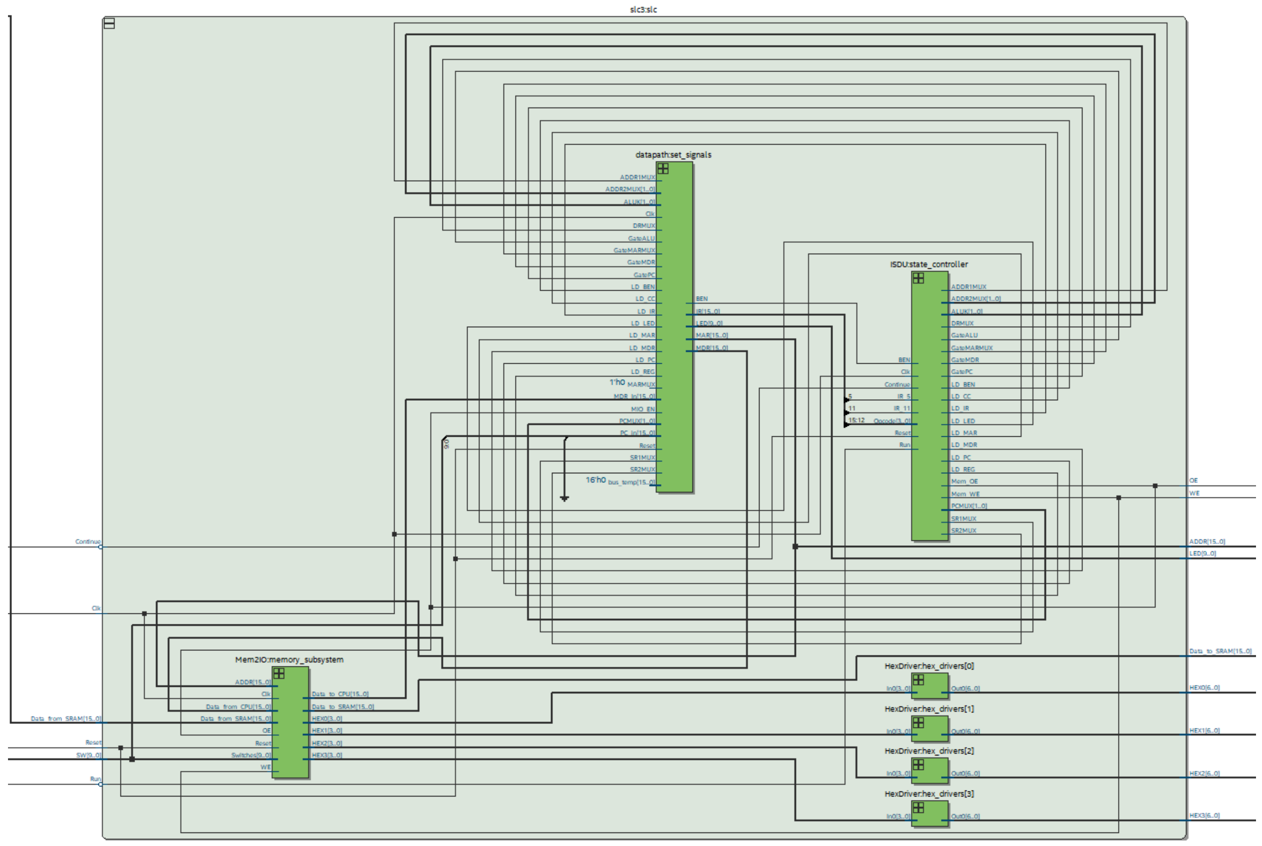


Figure 1: Block diagram of slc3.sv module

State Diagram of ISDU

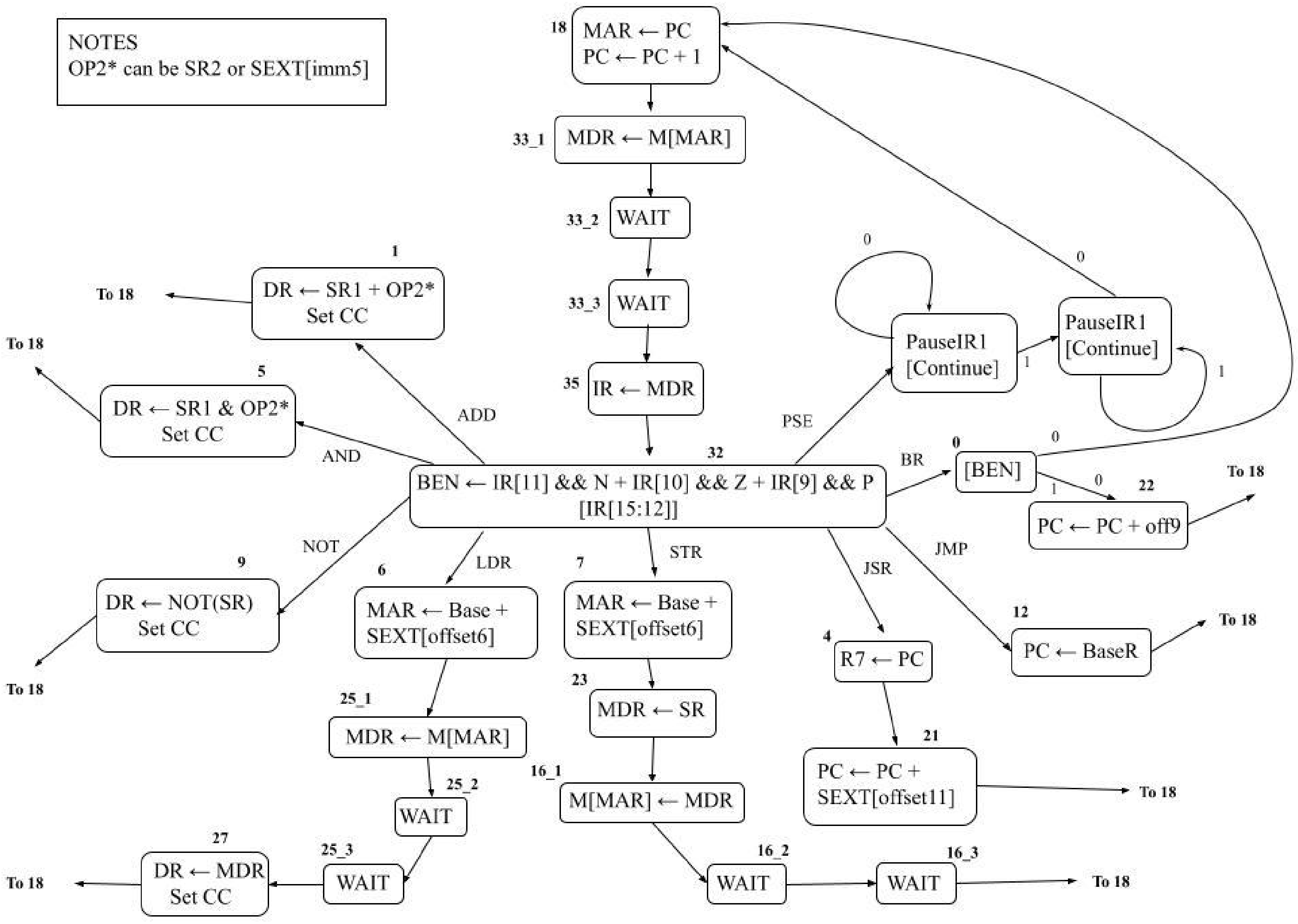
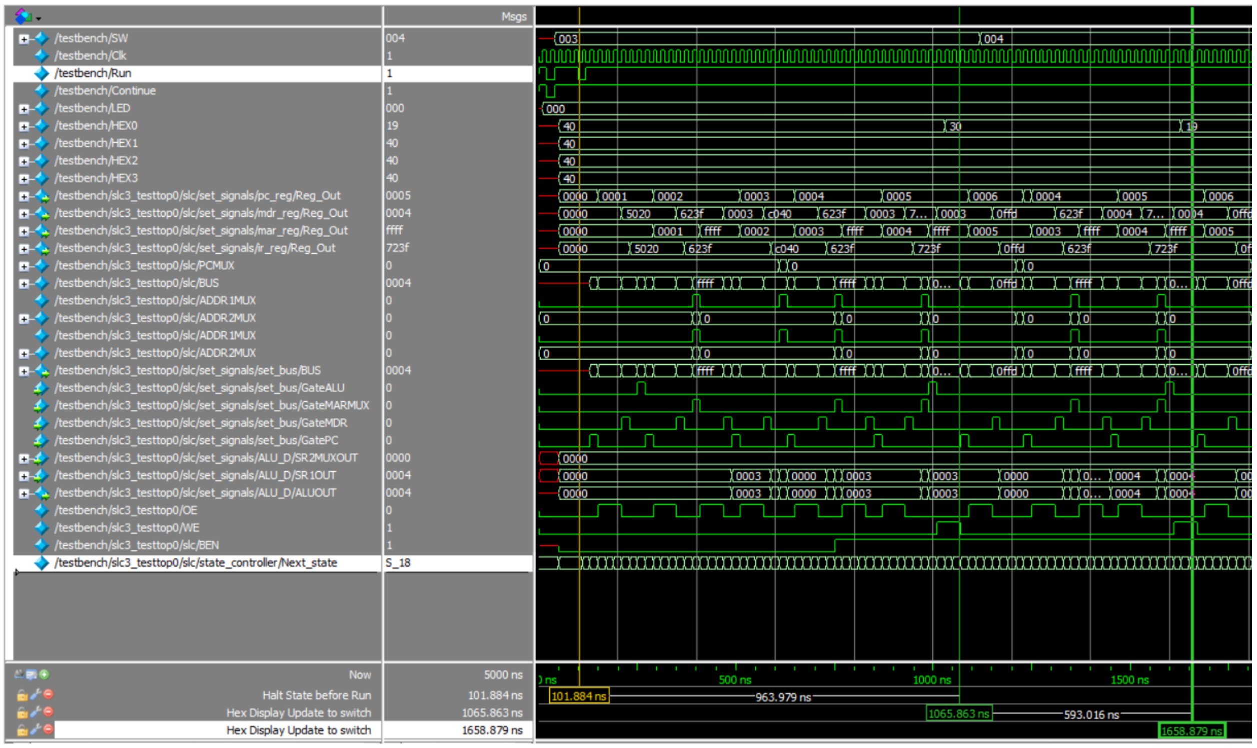


Figure 2: State diagram of ISDU

Simulations of SLC-3 Instructions



	Msgs
/testbench/SW	004
/testbench/Clk	1
/testbench/Run	1
/testbench/Continue	1
/testbench/LED	000
/testbench/HEX0	19
/testbench/HEX1	40
/testbench/HEX2	40
/testbench/HEX3	40
/testbench/slc3_testtop0/slc/set_signals/pc_reg/Reg_Out	0005
/testbench/slc3_testtop0/slc/set_signals/mdr_reg/Reg_Out	0004
/testbench/slc3_testtop0/slc/set_signals/mar_reg/Reg_Out	ffff
/testbench/slc3_testtop0/slc/set_signals/ir_reg/Reg_Out	723f
/testbench/slc3_testtop0/slc/PCMUX	0
/testbench/slc3_testtop0/slc/BUS	0004
/testbench/slc3_testtop0/slc/ADDR1MUX	0
/testbench/slc3_testtop0/slc/ADDR2MUX	0
/testbench/slc3_testtop0/slc/ADDR1MUX	0
/testbench/slc3_testtop0/slc/ADDR2MUX	0
/testbench/slc3_testtop0/slc/set_signals/set_bus/BUS	0004
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateALU	0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMARMUX	0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMDR	0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GatePC	0
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR2MUXOUT	0000
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR1OUT	0004
/testbench/slc3_testtop0/slc/set_signals/ALU_D/ALUOUT	0004
/testbench/slc3_testtop0/OE	0
/testbench/slc3_testtop0/WE	1
/testbench/slc3_testtop0/slc/BEN	1
/testbench/slc3_testtop0/slc/state_controller/Next_state	S_18

Now	5000 ns
Halt State before Run	101.884 ns
Hex Display Update to switch	1065.863 ns
Hex Display Update to switch	1658.879 ns

Figure 3: IO Test 1

- Test 1 Simulation
  - FPGA is reset
  - OPCode is set to switches and run is pressed
  - As switches are changed the value on the Hex Display is updated to be the value of the switches

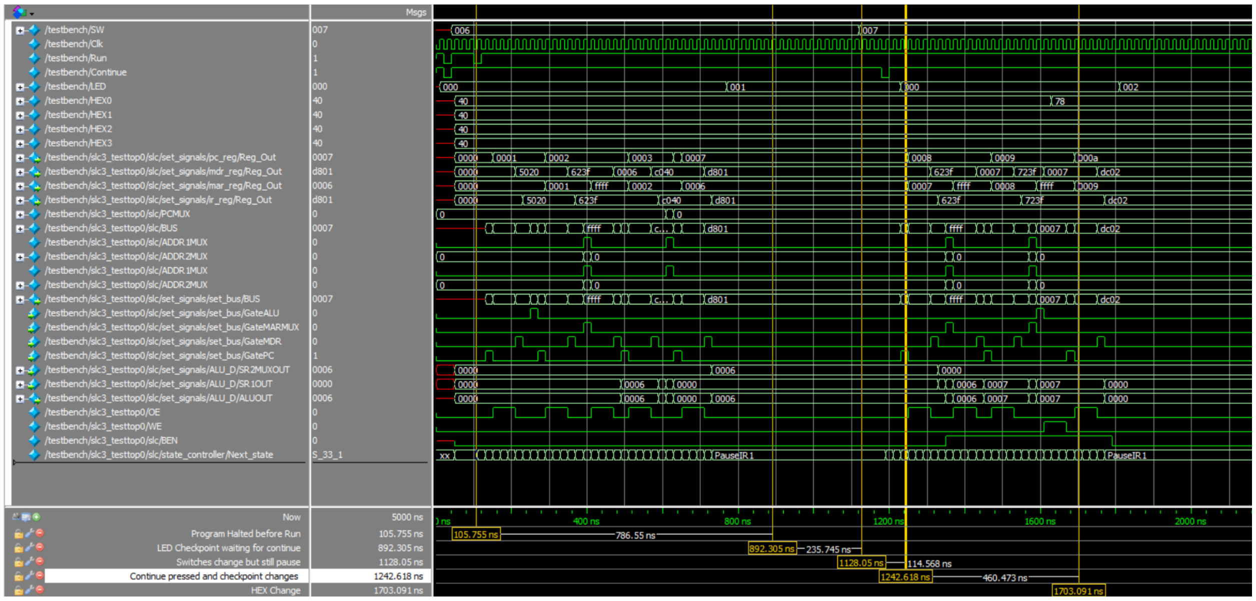

/testbench/SW
/testbench/Clk
/testbench/Run
/testbench/Continue
/testbench/LED
/testbench/HEX0
/testbench/HEX1
/testbench/HEX2
/testbench/HEX3
/testbench/slc3_testtop0/slc/set_signals/pc_reg/Reg_Out
/testbench/slc3_testtop0/slc/set_signals/mdr_reg/Reg_Out
/testbench/slc3_testtop0/slc/set_signals/mar_reg/Reg_Out
/testbench/slc3_testtop0/slc/set_signals/ir_reg/Reg_Out
/testbench/slc3_testtop0/slc/PCMUX
/testbench/slc3_testtop0/slc/BUS
/testbench/slc3_testtop0/slc/state_controller/Next_state
Program Halted before Run
LED Checkpoint waiting for continue
Switches change but still pause
Continue pressed and checkpoint changes
HEX Change

Figure 4: IO Test 2

- Test 2 Simulation
    - FPGA is reset
    - OPCode is set to switches and run is pressed
    - As switches are changed the value on the Hex Display does not update
    - Only when continue is also pressed does the Hex Display update

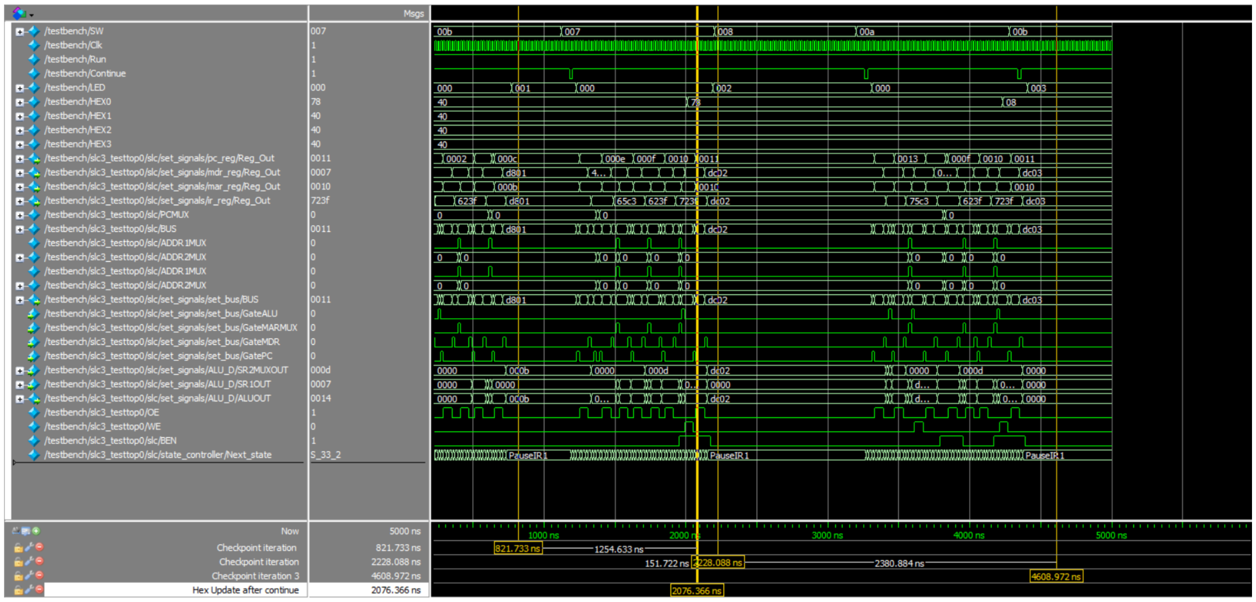
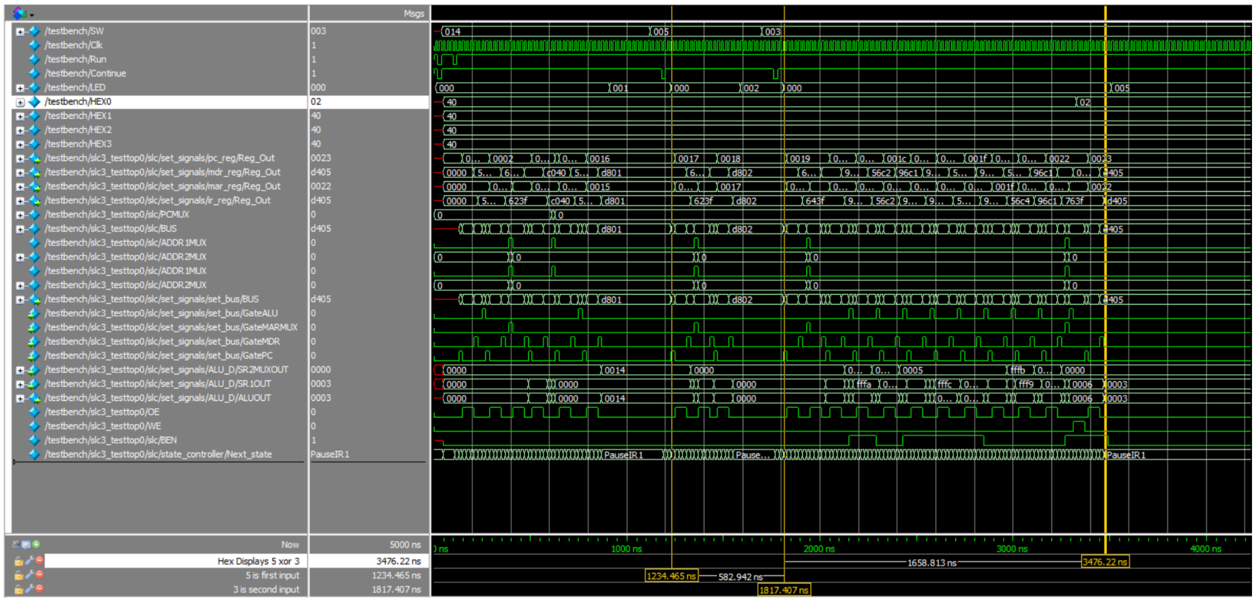


Figure 5: Self Modifying Code Test

- Test 3 Simulation
    - FPGA is reset

- OPCode is set to switches and run is pressed
- As switches are changed the value on the Hex Display does not update
- Only when continue is also pressed does the Hex Display update and is the checkpoint value displayed on the LED's iterated by 1



```
                                                                 Msgs
/testbench/SW                                                    003
/testbench/Clk                                                   1
/testbench/Run                                                   1
/testbench/Continue                                              1
/testbench/LED                                                   000
/testbench/HEX0                                                  02
/testbench/HEX1                                                  40
/testbench/HEX2                                                  40
/testbench/HEX3                                                  40
/testbench/slc3_testtop0/slc/set_signals/pc_reg/Reg_Out          0023
/testbench/slc3_testtop0/slc/set_signals/mdr_reg/Reg_Out         d405
/testbench/slc3_testtop0/slc/set_signals/mar_reg/Reg_Out         0022
/testbench/slc3_testtop0/slc/set_signals/ir_reg/Reg_Out          d405
/testbench/slc3_testtop0/slc/PCMUX                               0
/testbench/slc3_testtop0/slc/BUS                                 d405
/testbench/slc3_testtop0/slc/ADDR1MUX                            0
/testbench/slc3_testtop0/slc/ADDR2MUX                            0
/testbench/slc3_testtop0/slc/ADDR1MUX                            0
/testbench/slc3_testtop0/slc/ADDR2MUX                            0
/testbench/slc3_testtop0/slc/set_signals/set_bus/BUS             d405
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateALU         0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMARMUX      0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMDR         0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GatePC          0
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR2MUXOUT         0000
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR1OUT            0003
/testbench/slc3_testtop0/slc/set_signals/ALU_D/ALUOUT            0003
/testbench/slc3_testtop0/OE                                      0
/testbench/slc3_testtop0/WE                                      0
/testbench/slc3_testtop0/slc/BEN                                 1
/testbench/slc3_testtop0/slc/state_controller/Next_state         PauseIR1

                                                   Now           5000 ns
                                    Hex Displays 5 xor 3         3476.22 ns
                                          5 is first input       1234.465 ns
                                         3 is second input       1817.407 ns
```

Figure 6: XOR Test

- Test 4 Simulation
  - FPGA is reset
  - OPCode is set to switches and run is pressed
  - To perform XOR, the switches are set to some value and continue is pressed (0101 in this case)
  - After Pause is reached switches are set to a different value and continue is pressed once more (0011 in this case)
  - The Checkpoints on the LED's update at each pause and once the final value is outputted on the HEX Displays
  - The Checkpoint is reset and the program is ready to be run again

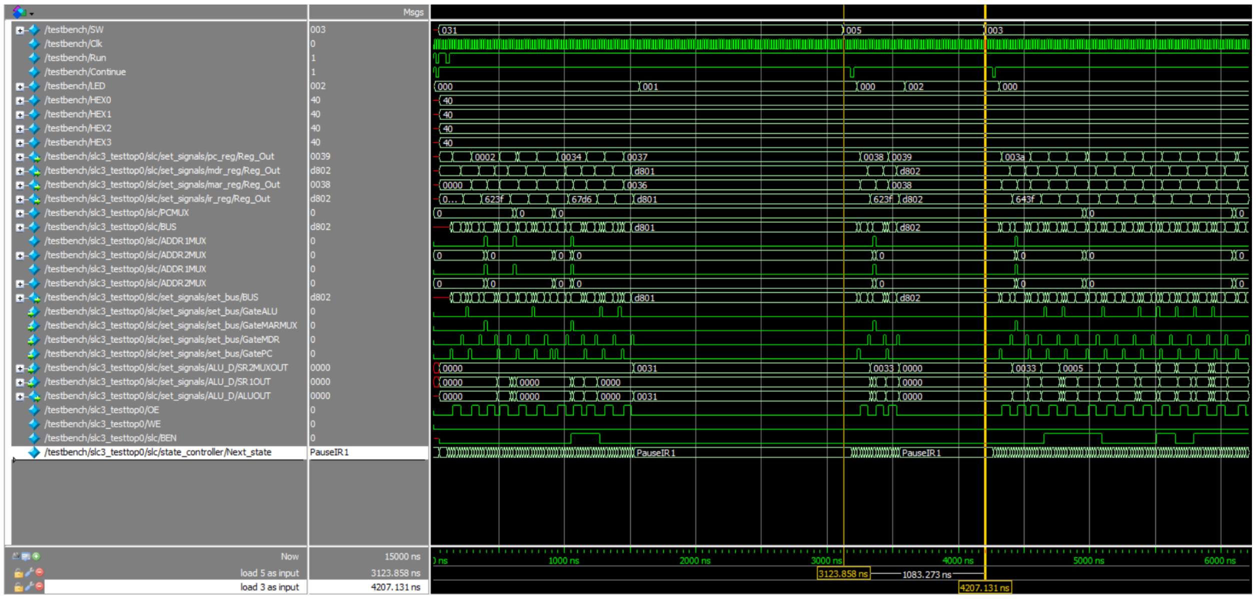

/testbench/SW 003
/testbench/Clk 0
/testbench/Run 1
/testbench/Continue 1
/testbench/LED 002
/testbench/HEX0 40
/testbench/HEX1 40
/testbench/HEX2 40
/testbench/HEX3 40
/testbench/slc3_testtop0/slc/set_signals/pc_reg/Reg_Out 0039
/testbench/slc3_testtop0/slc/set_signals/mdr_reg/Reg_Out d802
/testbench/slc3_testtop0/slc/set_signals/mar_reg/Reg_Out 0038
/testbench/slc3_testtop0/slc/set_signals/ir_reg/Reg_Out d802
/testbench/slc3_testtop0/slc/PCMUX 0
/testbench/slc3_testtop0/slc/BUS d802
/testbench/slc3_testtop0/slc/ADDR1MUX 0
/testbench/slc3_testtop0/slc/ADDR2MUX 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/BUS d802
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateALU 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMARMUX 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMDR 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GatePC 0
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR2MUXOUT 0000
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR1OUT 0000
/testbench/slc3_testtop0/slc/set_signals/ALU_D/ALUOUT 0000
/testbench/slc3_testtop0/OE 0
/testbench/slc3_testtop0/WE 0
/testbench/slc3_testtop0/slc/BEN 0
/testbench/slc3_testtop0/slc/state_controller/Next_state PauseIR1
Now 15000 ns
load 5 as input 3123.858 ns
load 3 as input 4207.131 ns
1083.273 ns

Figure 7.1: Multiplier Test

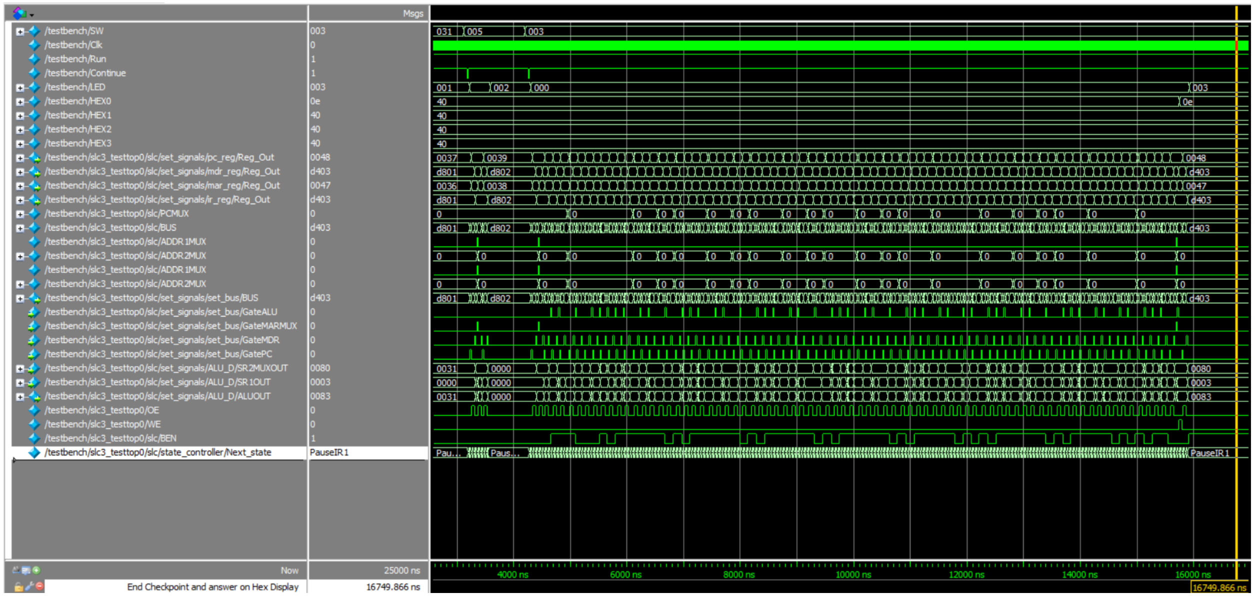
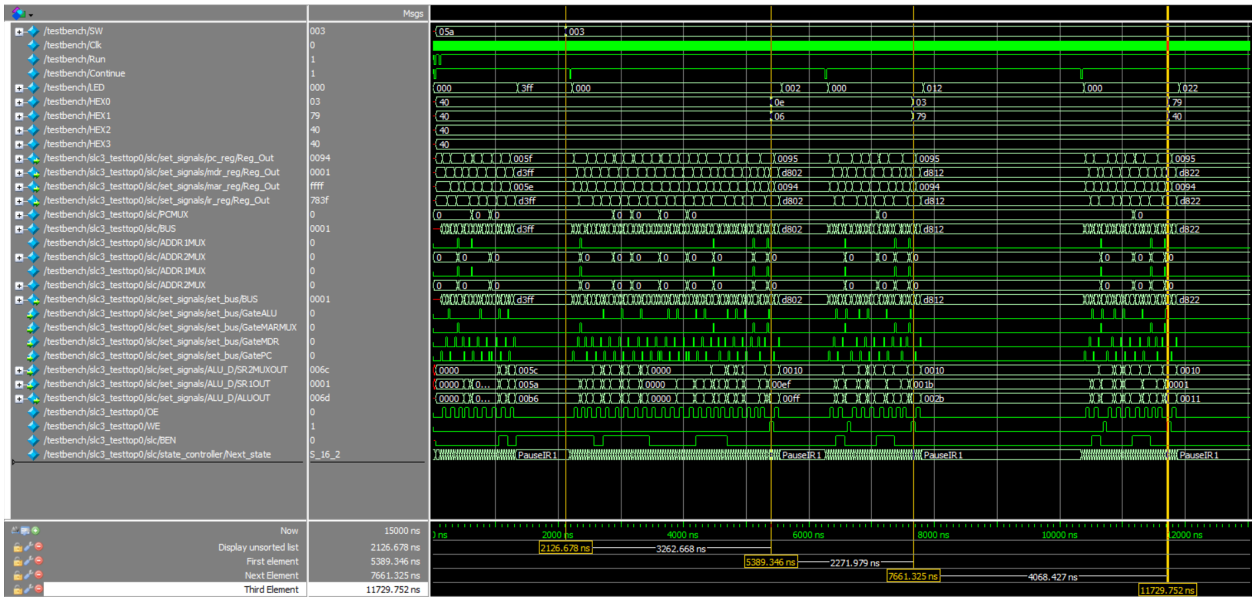

/testbench/SW 003
/testbench/Clk 0
/testbench/Run 1
/testbench/Continue 1
/testbench/LED 003
/testbench/HEX0 0e
/testbench/HEX1 40
/testbench/HEX2 40
/testbench/HEX3 40
/testbench/slc3_testtop0/slc/set_signals/pc_reg/Reg_Out 0048
/testbench/slc3_testtop0/slc/set_signals/mdr_reg/Reg_Out d403
/testbench/slc3_testtop0/slc/set_signals/mar_reg/Reg_Out 0047
/testbench/slc3_testtop0/slc/set_signals/ir_reg/Reg_Out d403
/testbench/slc3_testtop0/slc/PCMUX 0
/testbench/slc3_testtop0/slc/BUS d403
/testbench/slc3_testtop0/slc/ADDR1MUX 0
/testbench/slc3_testtop0/slc/ADDR2MUX 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/BUS d403
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateALU 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMARMUX 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMDR 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GatePC 0
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR2MUXOUT 0080
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR1OUT 0003
/testbench/slc3_testtop0/slc/set_signals/ALU_D/ALUOUT 0083
/testbench/slc3_testtop0/OE 0
/testbench/slc3_testtop0/WE 0
/testbench/slc3_testtop0/slc/BEN 1
/testbench/slc3_testtop0/slc/state_controller/Next_state PauseIR1
Now 25000 ns
End Checkpoint and answer on Hex Display 16749.866 ns

Figure 7.2: Multiplier Test

- Test 5 Simulation
  - FPGA is reset
  - OPCode is set to switches and run is pressed
  - To perform multiplication, the switches are set to some value and continue is pressed (0101 in this case)
  - After Pause is reached switches are set to a different value and continue is pressed once more (0011 in this case)

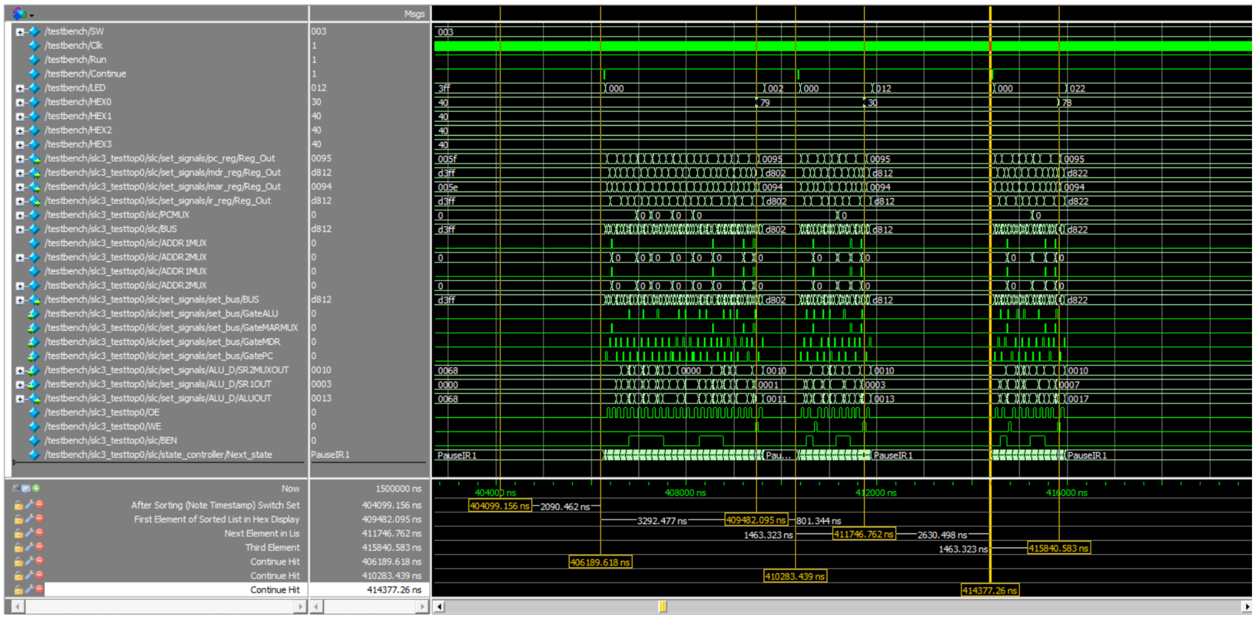
- The Checkpoints on the LED's update at each pause and once the final value is outputted on the HEX Displays
- The Checkpoint is reset and the program is ready to be run again



```
/testbench/SW                                              003
/testbench/Clk                                             0
/testbench/Run                                             1
/testbench/Continue                                        1
/testbench/LED                                             000
/testbench/HEX0                                            03
/testbench/HEX1                                            79
/testbench/HEX2                                            40
/testbench/HEX3                                            40
/testbench/slc3_testtop0/slc/set_signals/pc_reg/Reg_Out    0094
/testbench/slc3_testtop0/slc/set_signals/mdr_reg/Reg_Out   0001
/testbench/slc3_testtop0/slc/set_signals/mar_reg/Reg_Out   ffff
/testbench/slc3_testtop0/slc/set_signals/ir_reg/Reg_Out    783f
/testbench/slc3_testtop0/slc/PCMUX                         0
/testbench/slc3_testtop0/slc/BUS                           0001
/testbench/slc3_testtop0/slc/ADDR1MUX                      0
/testbench/slc3_testtop0/slc/ADDR2MUX                      0
/testbench/slc3_testtop0/slc/ADDR1MUX                      0
/testbench/slc3_testtop0/slc/ADDR2MUX                      0
/testbench/slc3_testtop0/slc/set_signals/set_bus/BUS       0001
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateALU   0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMARMUX 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMDR   0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GatePC    0
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR2MUXOUT   006c
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR1OUT      0001
/testbench/slc3_testtop0/slc/set_signals/ALU_D/ALUOUT      006d
/testbench/slc3_testtop0/OE                                0
/testbench/slc3_testtop0/WE                                1
/testbench/slc3_testtop0/slc/BEN                           0
/testbench/slc3_testtop0/slc/state_controller/Next_state   S_16_2

Now                    15000 ns
Display unsorted list  2126.678 ns
First element          5389.346 ns
Next Element           7661.325 ns
Third Element          11729.752 ns
```

Figure 8.1:Sort Test - Unsorted Elements



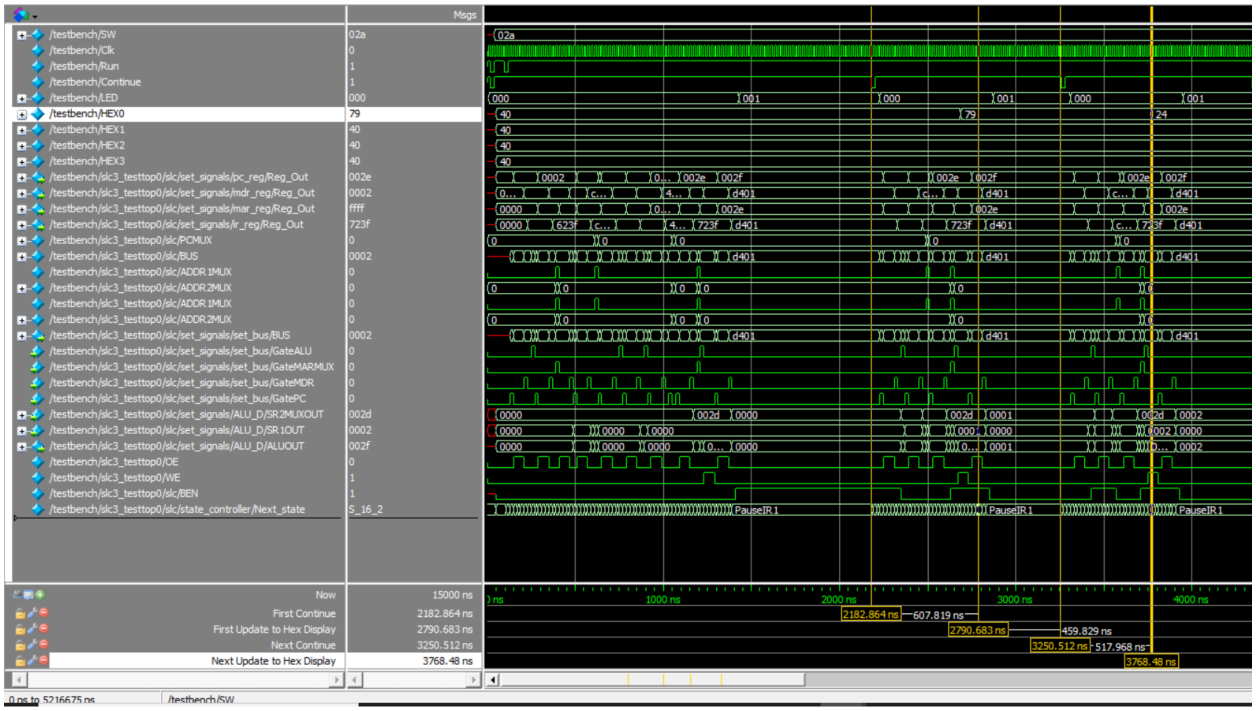
```
/testbench/SW                                              003
/testbench/Clk                                             1
/testbench/Run                                             1
/testbench/Continue                                        1
/testbench/LED                                             012
/testbench/HEX0                                            30
/testbench/HEX1                                            40
/testbench/HEX2                                            40
/testbench/HEX3                                            40
/testbench/slc3_testtop0/slc/set_signals/pc_reg/Reg_Out    0095
/testbench/slc3_testtop0/slc/set_signals/mdr_reg/Reg_Out   d812
/testbench/slc3_testtop0/slc/set_signals/mar_reg/Reg_Out   0094
/testbench/slc3_testtop0/slc/set_signals/ir_reg/Reg_Out    d812
/testbench/slc3_testtop0/slc/PCMUX                         0
/testbench/slc3_testtop0/slc/BUS                           d812
/testbench/slc3_testtop0/slc/ADDR1MUX                      0
/testbench/slc3_testtop0/slc/ADDR2MUX                      0
/testbench/slc3_testtop0/slc/ADDR1MUX                      0
/testbench/slc3_testtop0/slc/ADDR2MUX                      0
/testbench/slc3_testtop0/slc/set_signals/set_bus/BUS       d812
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateALU   0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMARMUX 0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMDR   0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GatePC    0
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR2MUXOUT   0010
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR1OUT      0003
/testbench/slc3_testtop0/slc/set_signals/ALU_D/ALUOUT      0013
/testbench/slc3_testtop0/OE                                0
/testbench/slc3_testtop0/WE                                0
/testbench/slc3_testtop0/slc/BEN                           0
/testbench/slc3_testtop0/slc/state_controller/Next_state   PauseIR1

Now                                            1500000 ns
After Sorting (Note Timestamp) Switch Set      404099.156 ns
First Element of Sorted List in Hex Display    409482.095 ns
Next Element in Lis                            411746.762 ns
Third Element                                  415840.583 ns
Continue Hit                                   406189.618 ns
Continue Hit                                   410283.439 ns
Continue Hit                                   414377.26 ns
```

Figure 9: Sort Test - Sorted Elements

- Test 6 Simulation
  - FPGA is reset
  - OPCode is set to switches and run is pressed

- First switches are set to 3 and continue is pressed continuously to view unsorted list
- Once 3FFF is reached on the LED's, switches are set to 2, and continue is pressed once again, this sorts the list
- Finally switches are set to 3 and continue is pressed continuously to view the sorted list



```
/testbench/SW                                                   02a
/testbench/Clk                                                  0
/testbench/Run                                                  1
/testbench/Continue                                             1
/testbench/LED                                                  000
/testbench/HEX0                                                 79
/testbench/HEX1                                                 40
/testbench/HEX2                                                 40
/testbench/HEX3                                                 40
/testbench/slc3_testtop0/slc/set_signals/pc_reg/Reg_Out         002e
/testbench/slc3_testtop0/slc/set_signals/mdr_reg/Reg_Out        0002
/testbench/slc3_testtop0/slc/set_signals/mar_reg/Reg_Out        ffff
/testbench/slc3_testtop0/slc/set_signals/ir_reg/Reg_Out         723f
/testbench/slc3_testtop0/slc/PCMUX                              0
/testbench/slc3_testtop0/slc/BUS                                0002
/testbench/slc3_testtop0/slc/ADDR1MUX                           0
/testbench/slc3_testtop0/slc/ADDR2MUX                           0
/testbench/slc3_testtop0/slc/ADDR1MUX                           0
/testbench/slc3_testtop0/slc/ADDR2MUX                           0
/testbench/slc3_testtop0/slc/set_signals/set_bus/BUS            0002
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateALU        0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMARMUX     0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GateMDR        0
/testbench/slc3_testtop0/slc/set_signals/set_bus/GatePC         0
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR2MUXOUT        002d
/testbench/slc3_testtop0/slc/set_signals/ALU_D/SR1OUT           0002
/testbench/slc3_testtop0/slc/set_signals/ALU_D/ALUOUT           002f
/testbench/slc3_testtop0/OE                                     0
/testbench/slc3_testtop0/WE                                     1
/testbench/slc3_testtop0/slc/BEN                                1
/testbench/slc3_testtop0/slc/state_controller/Next_state        S_16_2

Now                              15000 ns
First Continue                   2182.864 ns
First Update to Hex Display      2790.683 ns
Next Continue                    3250.512 ns
Next Update to Hex Display       3768.48 ns
```

Figure 10: Act Once Test

- Test 7 Simulation
  - FPGA is reset
  - OPCode is set to switches and run is pressed
  - To iterate through the counter continue is pressed continuously
  - The checkpoint is continuously set to 1 as the Hex Display value increases by one each button press of continue

Post-Lab Questions

1) Refer to the Design Resources and Statistics in IQT.17-19 and complete the following design statistics table.

LUT	866
DSP	0
Memory (BRAM)	18432
Flip-Flop	268
Frequency	71.98 MHz
Static Power	89.94 mW
Dynamic Power	0.00 mW
Total Power	98.63 mW

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

Answer: MEM2IO is the interface between the FPGA's memory and the switches, it treats the address of xFFFF as a read from the switches and treats the instruction xFFFF as a write to the Hex seven segment display to output values. This is important for the LDR and STR functions, if the address or instruction is not xFFFF the FPGA will read and write data to memory.

3) What is the difference between BR and JMP instructions?

Answer: Both of these instructions directly affect the PC. However, for the JMP instruction, the PC is changed to a PC of a subroutine. The PC changes back to the PC where the JMP was executed once the subroutine is fully executed and the RET (return) instruction is executed towards the end. For the BR instruction, the change is PC only occurs if the NZP signal in the BR instruction matches the condition codes set in the previous instruction. The PC can be changed by a small, limited number and it does not return to the PC where the BR instruction was executed. For JMP, the PC can be changed to any 16-bit value stored in the base register, and for BR, the PC can only be changed by a 9-bit offset.

4) 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?

Answer: The R signal is known as the Ready signal. This signal, once asserted, indicates that the memory access is completed, and the data is ready for computation. Since we lack this signal in our SLC-3 microprocessor, we compensate by having 3 wait or pause states for all instructions that require the system to wait for the R signal. Since the states don't need to wait for an R signal to be raised, all the states that require the system to wait, wait for the same number of cycles, meaning our processor is asynchronous.

Conclusion

Given that this is considered one of the hardest labs in this class, we were extremely happy to have a fully functioning SLC-3 Microprocessor. The two most difficult and important parts of the lab were creating the Datapath and setting the right signals for all the phases and opcodes according to the State Diagram. The most difficult path in the lab was to understand what each signal did and what signals need to be on for each state in the state diagram. In our opinion, more than the experiment documentation, the additional resources on the Lab 5 page such as the resources from ECE 120 were more helpful in creating the microprocessor.

The lab was an amazing way for us to understand how a simple microprocessor such as the SLC-3 microprocessor works. We learned how the Datapath is created, the State diagram, and how the States control the microprocessor. Overall, our objective of creating an SLC-3 Microprocessor, a subset of an LC-3 processor was complete.

Appendix A

Module: hexdriver.sv

Inputs: [3:0] In0,

Outputs: [6:0] Out0

Description: Takes a nibble or 4 bits and converts it to a 7 bit equivalent that is displayable on the HEX# output LEDs of the FPGA.

Purpose: To allow the seven segment displays to display what the input and output of our code are.

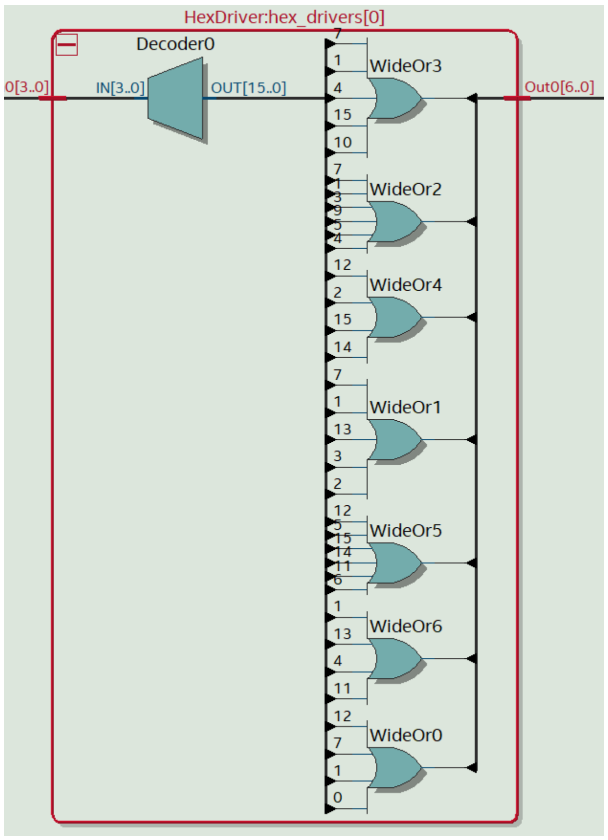


Figure 11: RTL Block diagram for hexdriver.sv

Module: datapath.sv

Inputs: Clk, Reset, LD_MAR, LD_MDR, LD_IR, LD_BEN, LD_CC, LD_REG, LD_PC, LD_LED, SR2MUX, ADDR1MUX, MARMUX, GatePC, GateMDR, GateALU, GateMARMUX, MIO_EN, DRMUX, SR1MUX, PCMUX, ADDR2MUX, ALUK, MDR_In, PC_in, bus_temp,

Outputs: LED, MAR, MDR, IR, PC, PC_Now, BUS, BEN

Description: Construction of the LC3 datapath, a combination of MUXes and register signals connected to the bus via tristate gate,

Purpose: To implement the SLC3 the datapath handles directing all the signals in the correct way such that the outputs and inputs through MEM2IO are correct.

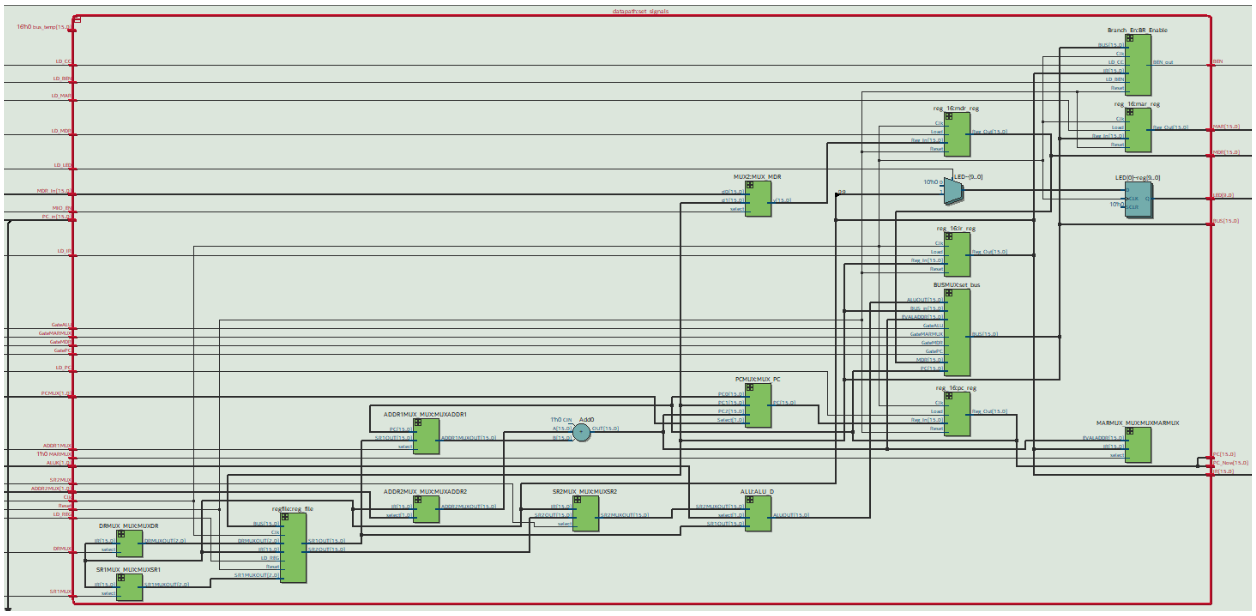


Figure 12: RTL Block diagram datapath.sv

Module: slc3.sv

Inputs: Clk, Reset, LD_MAR, LD_MDR, LD_IR, LD_BEN, LD_CC, LD_REG, LD_PC, LD_LED, SR2MUX, ADDR1MUX, MARMUX, GatePC, GateMDR, GateALU, GateMARMUX, MIO_EN, DRMUX, SR1MUX, PCMUX, ADDR2MUX, ALUK, MDR_In, PC_in, bus_temp,

Outputs: LED, MAR, MDR, IR, PC, PC_Now, BUS, BEN

Description: instantiates datapath, MEM2IO, and the ISDU, as well as the WE register, and constructs the HexDriver variables that are used to condense the instructions.

Purpose: To start the LC3 so that it can begin to run instructions

Module: SEXT.sv

Inputs: signed in, parameter size

Outputs: signed out

Description: Takes the signed value of the inputted value of size (parameter - 1) and extends the output to be the 16-bit sign extension of the input

Purpose: To make sign extension easy with a simple call of a module rather than manually extending each signal.

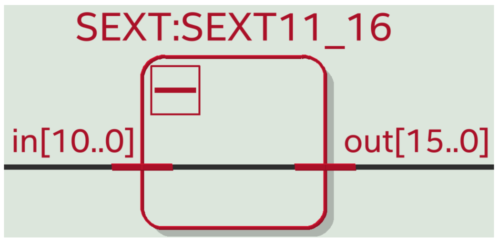


Figure 13: RTL Block diagram for SEXT.sv

Module: MUX2.sv

Inputs: d0, d1, select

Outputs: y

Description: parameterized 2 bit mux, used for MIO.En MUX to get MDR value.

Purpose: to implement MIO MUX in the datapath

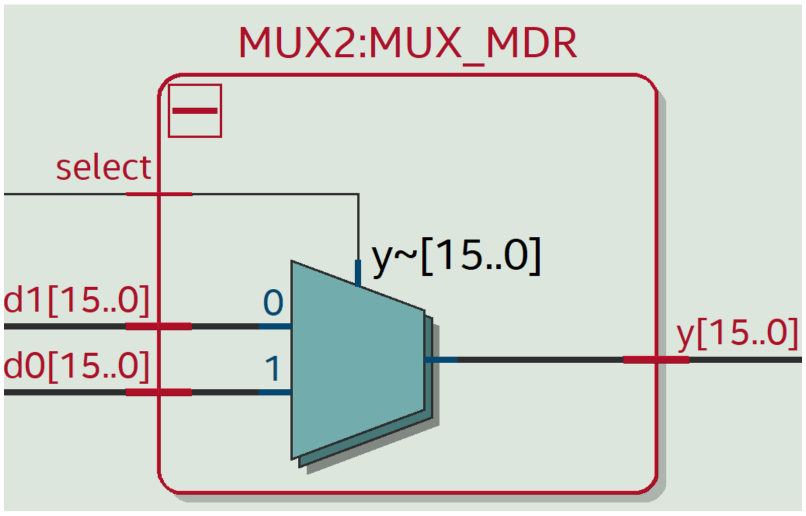


Figure 14: RTL Block diagram for MUX2.sv

Module: PCMUX.sv

Inputs: PC0, PC1, PC2,select

Outputs: PC

Description: MUX, used to choose the input for PC, and what will get passed on to the BUS if GatePC is High.

Purpose: to implement the PCMUX in datapath

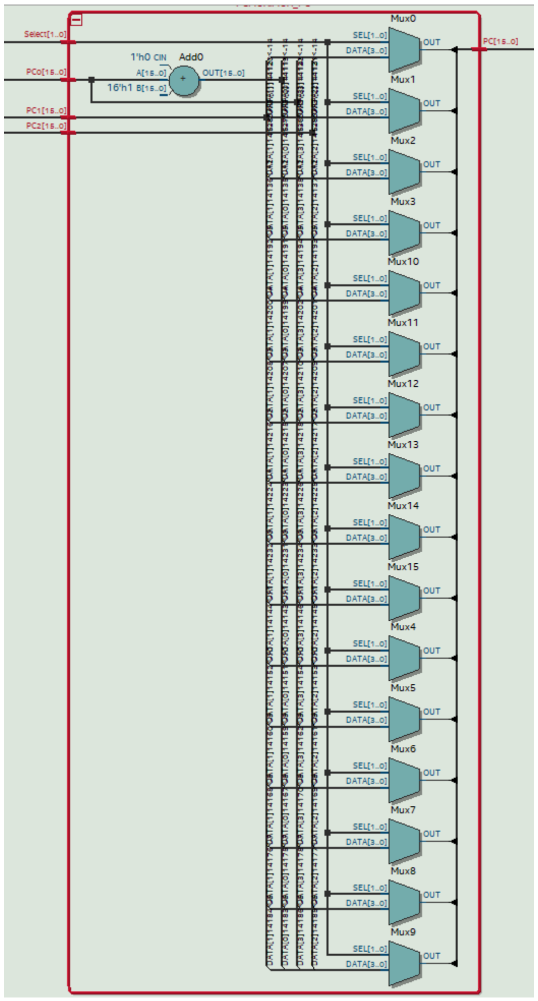


Figure 15: RTL Block diagram for PCMUX.sv

Module: BUSMUX.sv

Inputs: GateALU, GateMARMUX, GateMDR, GatePC,MDR, PC, BUS_in, EVALADDR, ALUOUT,

Outputs: BUS

Description: Tristate buffer, used to choose which signal from MAR, MDR, PC, or ALU gets to be on the BUS

Purpose: to implement the tristate buffer in a way that is executable on the FPGA

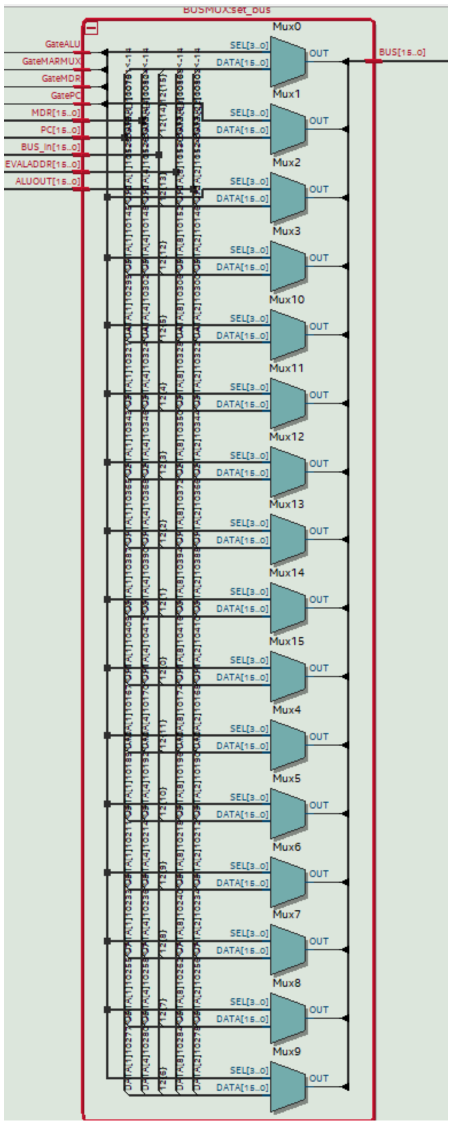


Figure 16: RTL Block diagram for BUSMUX.sv

Module: SR1MUX_MUX.sv

Inputs: IR, select

Outputs: SR1MUXOUT

Description: MUX that chooses the value of the SR1out that goes into the regfile

Purpose: to implement the SR1MUX in datapath

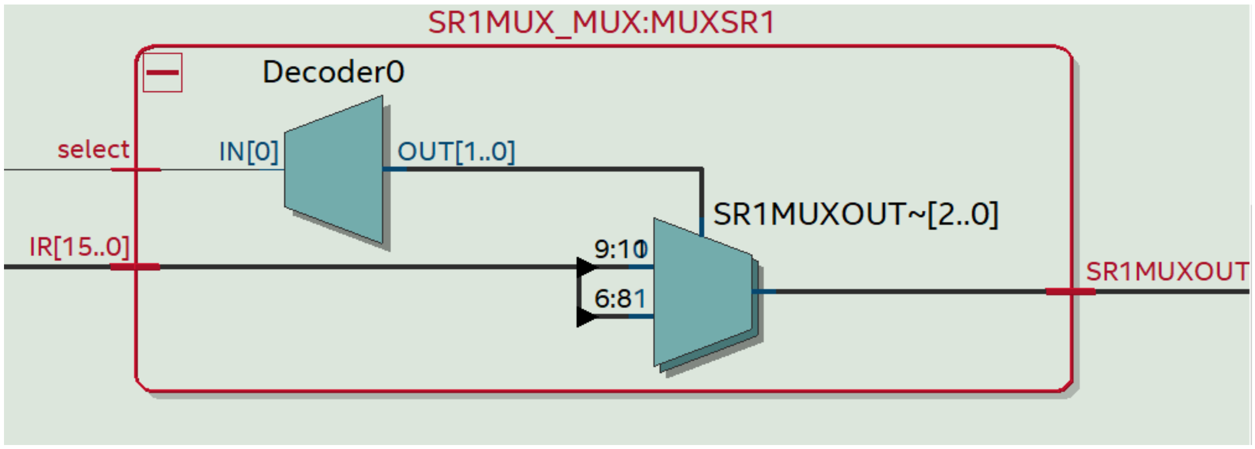


Figure 17: RTL Block diagram for SR1MUX_MUX.sv

Module: DRMUX_MUX.sv

Inputs: IR, select

Outputs: DRMUXOUT

Description: MUX to choose the value of the DR between 0-7 to be used in the regfile later

Purpose: to implement DRMUX in datapath

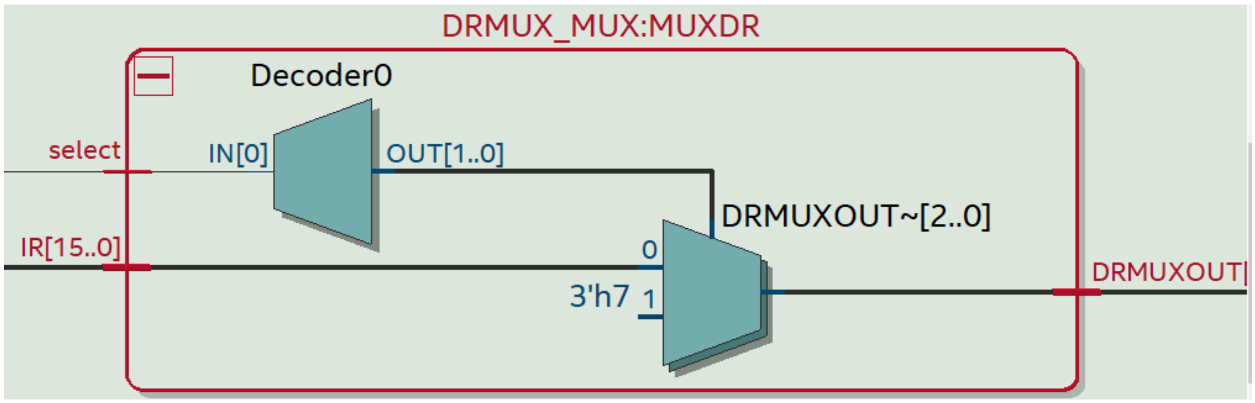


Figure 18: RTL Block diagram for DRMUX_MUX.sv

Module: SR2MUX_MUX.sv

Inputs: IR, SR2OUT, select

Outputs: SR2MUXOUT

Description: 2 bit MUX to choose the value of the B input for the ALU to be used in the ALU module later

Purpose: to implement SR2MUX in datapath

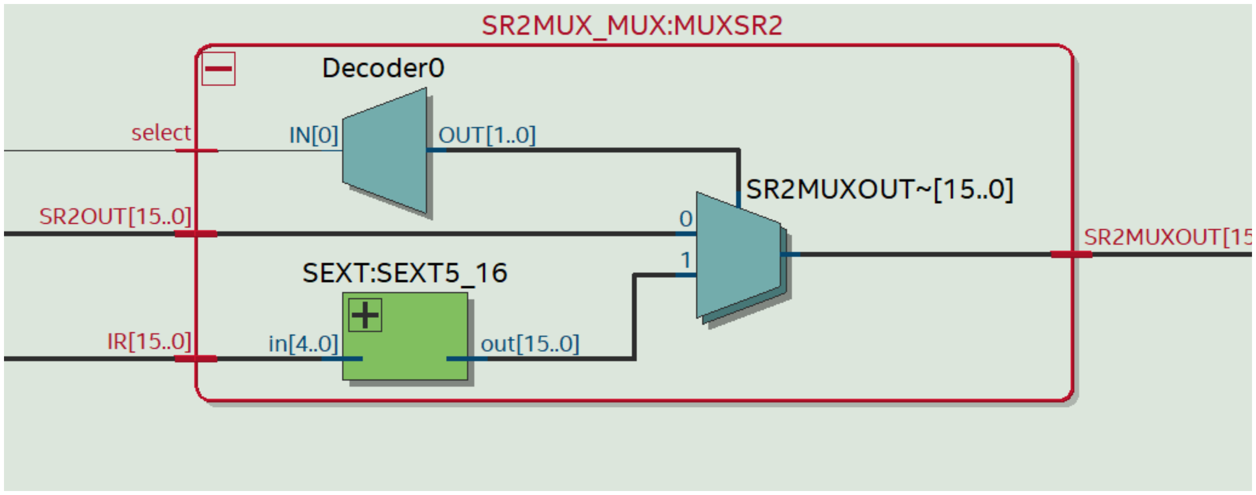


Figure 19: RTL Block diagram for SR2MUX_MUX.sv

Module: ADDR2MUX_MUX.sv

Inputs: IR, select

Outputs: ADDR2MUXOUT

Description: 4 bit MUX that chooses either 0 or some sign extended value of the IR, to be used in the Address addition

Purpose: to implement ADDR2 in datapath

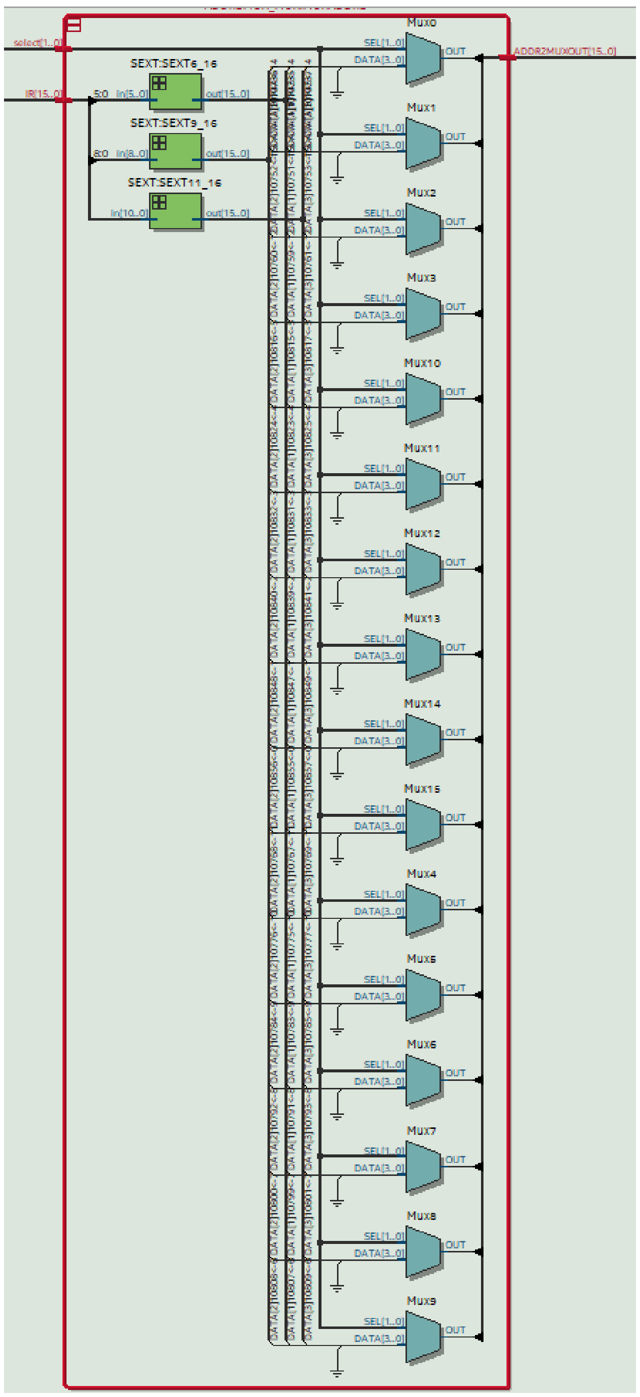


Figure 20: RTL Block diagram for ADDR2MUX_MUX.sv

Module: ADDR1MUX_MUX.sv

Inputs: SR1OUT, PC, , select

Outputs: ADDR1MUXOUT

Description: 2 bit MUX that chooses either the PC or the SR1OUT value from the reg file for the address addition.

Purpose: to implement ADDR1 in datapath

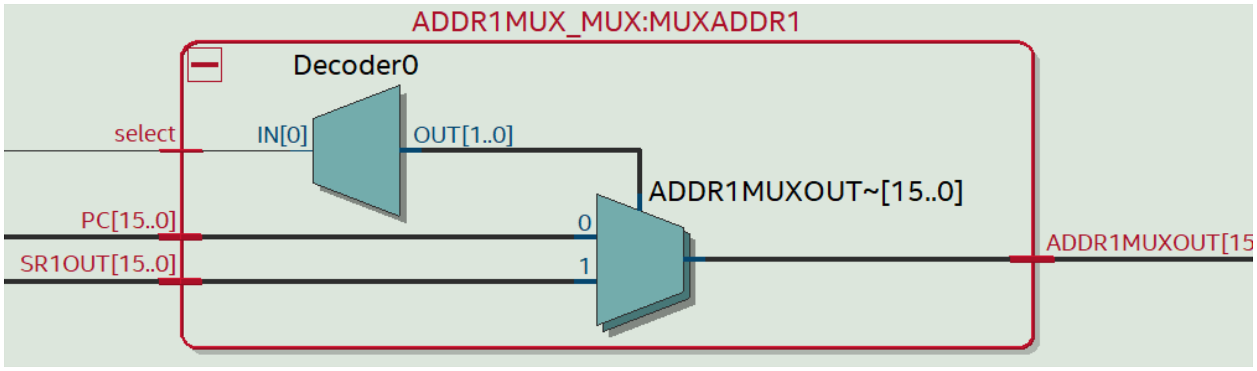


Figure 21: RTL Block diagram for ADDR1MUX_MUX.sv

Module: MARMUX_MUX.sv

Inputs: EVALADDR, IR, select

Outputs: MARMUXOUT

Description: MUX that chooses the value of MAR from either memory or the BUS.

Purpose: to implement the MUX feeding into MAR

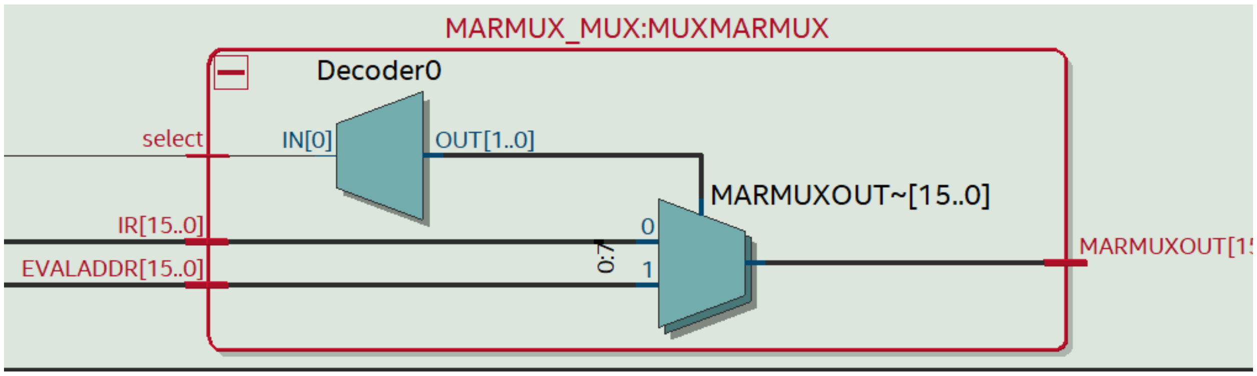


Figure 22: RTL Block diagram for MARMUX_MUX.sv

Module: regfile.sv

Inputs: LD_REG, Clk, Reset, IR, BUS, DRMUXOUT, SR1MUXOUT,

Outputs: SR1OUT, SR2OUT, R0, R1, R2, R3, R4, R5, R6, R7

Description: creates 8 registers 0-7 that are reset on reset, and are loaded with the value of the BUS if LD_REG is high, otherwise the outputs from the regfile are defined as the register corresponding to the output of the SR MUXes.

Purpose: to create the temp registers that the LC3 needs to run its functions.

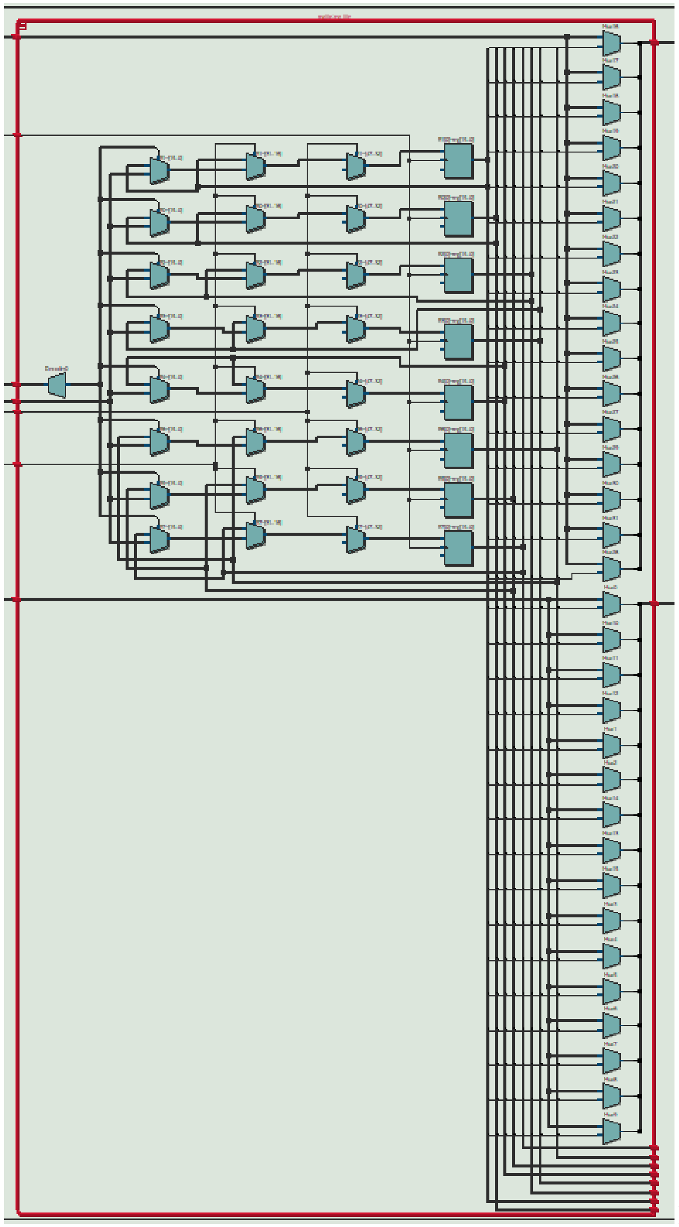


Figure 23: RTL Block diagram for regfile.sv

Module: reg_16.sv

Inputs: Clk, Reset, Load, RegIn

Outputs: RegOut

Description: 16 bit registers that construct MAR, PC, MDR, and IR, the registers reset to 0 on reset and are filled with the input value taken off the bus.

Purpose: To properly store the values of PC, MDR, MAR and IR

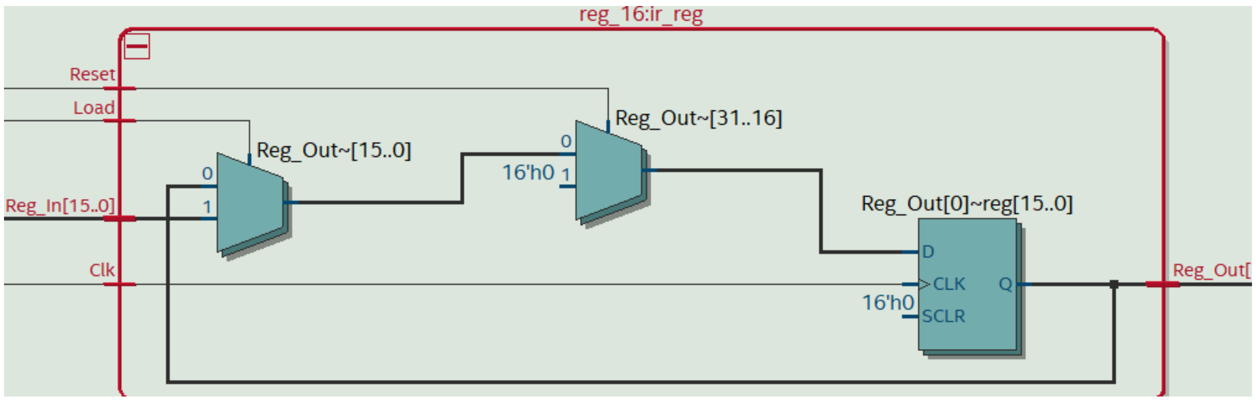


Figure 24: RTL Block diagram for reg_16.sv

Module: ISDU.sv

Inputs: Clk, Reset, Continue, Run, Opcode, IR_5, IR_11, BEN

Outputs: LD_MAR, LD_MDR, LD_BEN, LD_IR, LD_CC, LD_PC, LD_LED, GatePC, GateALU, GateMDR, GateMARMUX, PCMUX, DRMUX, SR1MUX, SR2MUX, ADDR1MUX, ADDR2MUX, ALUK, MEM_OE, MEM_WE.

Description: State Machine for the LC3, goes through fetch decode and execute states

Purpose: To allow proper function of the LC3 opcodes.

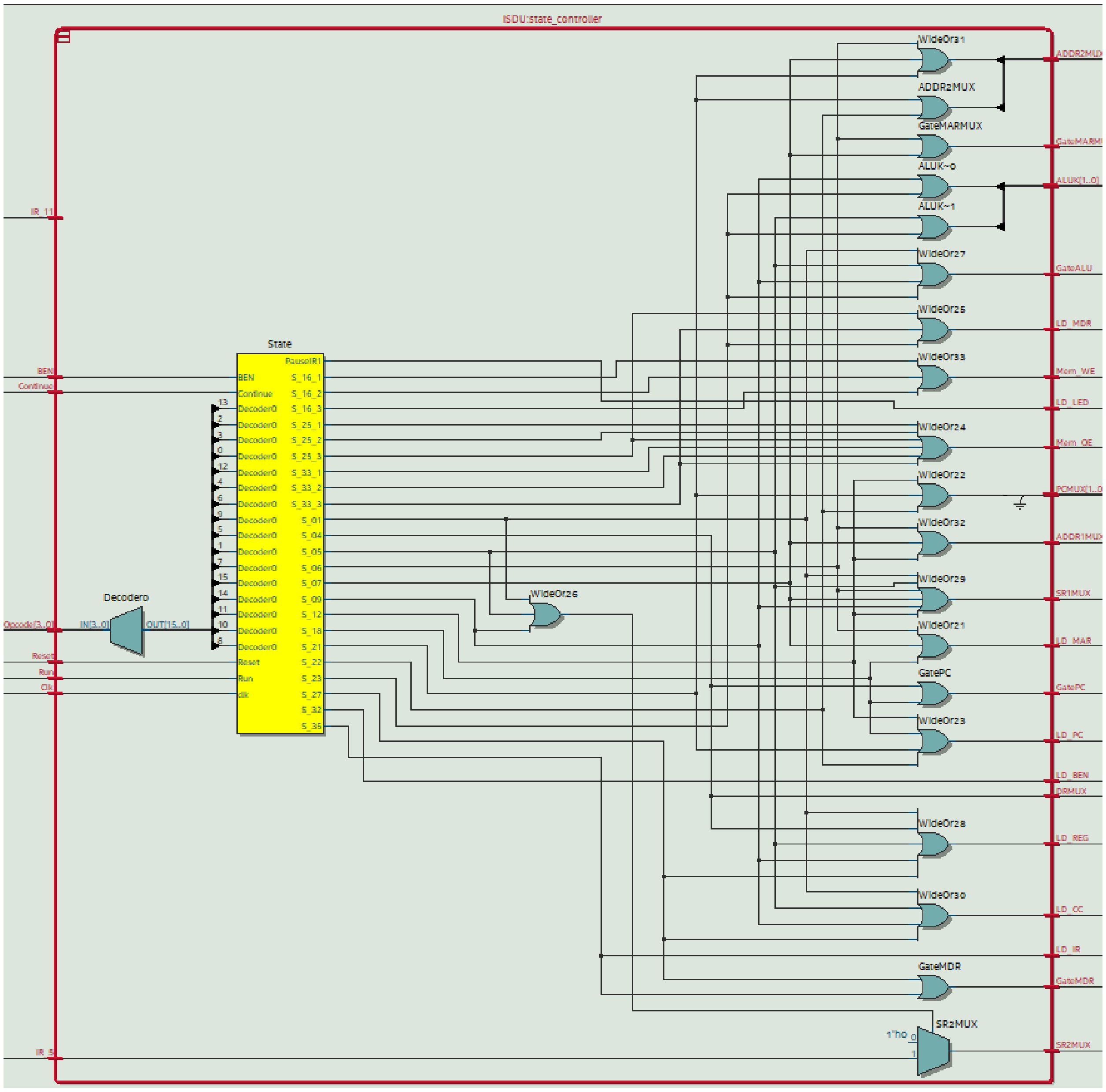


Figure 25: RTL Block diagram for ISDU.sv

Module: BEN.sv

Inputs: [15:0] IR, [15:0] BUS, LD_CC, LD_BEN, Reset, Clk

Outputs: BEN_out

Description: Branch logic to go to the control unit (ISDU). Given LD_CC and LD_BEN. Logic from IR [11:9] is loaded into NZP if LD_CC is on, the resulting logic is a SOP of all NZP and

Purpose: so that when a BR is hit, the next state branches accordingly

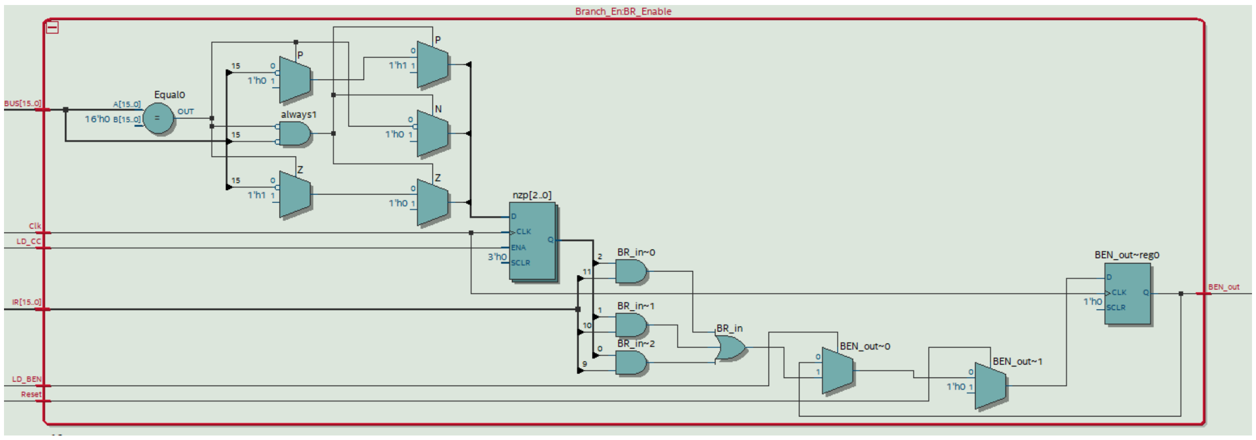


Figure 26: RTL Block diagram for BEN.sv

Module: ALU.sv

Inputs: [15:0] SR2MUXOUT, [15:0] SR1MUXOUT, [1:0] select,

Outputs: [15:0] ALUOUT

Description: Arithmetic Logic Unit to perform ADD, AND, NOT, and PASS operations on the register files outputs.

Purpose: to put register file outputs on the bus with the ALU operation applied if GateALU is high.

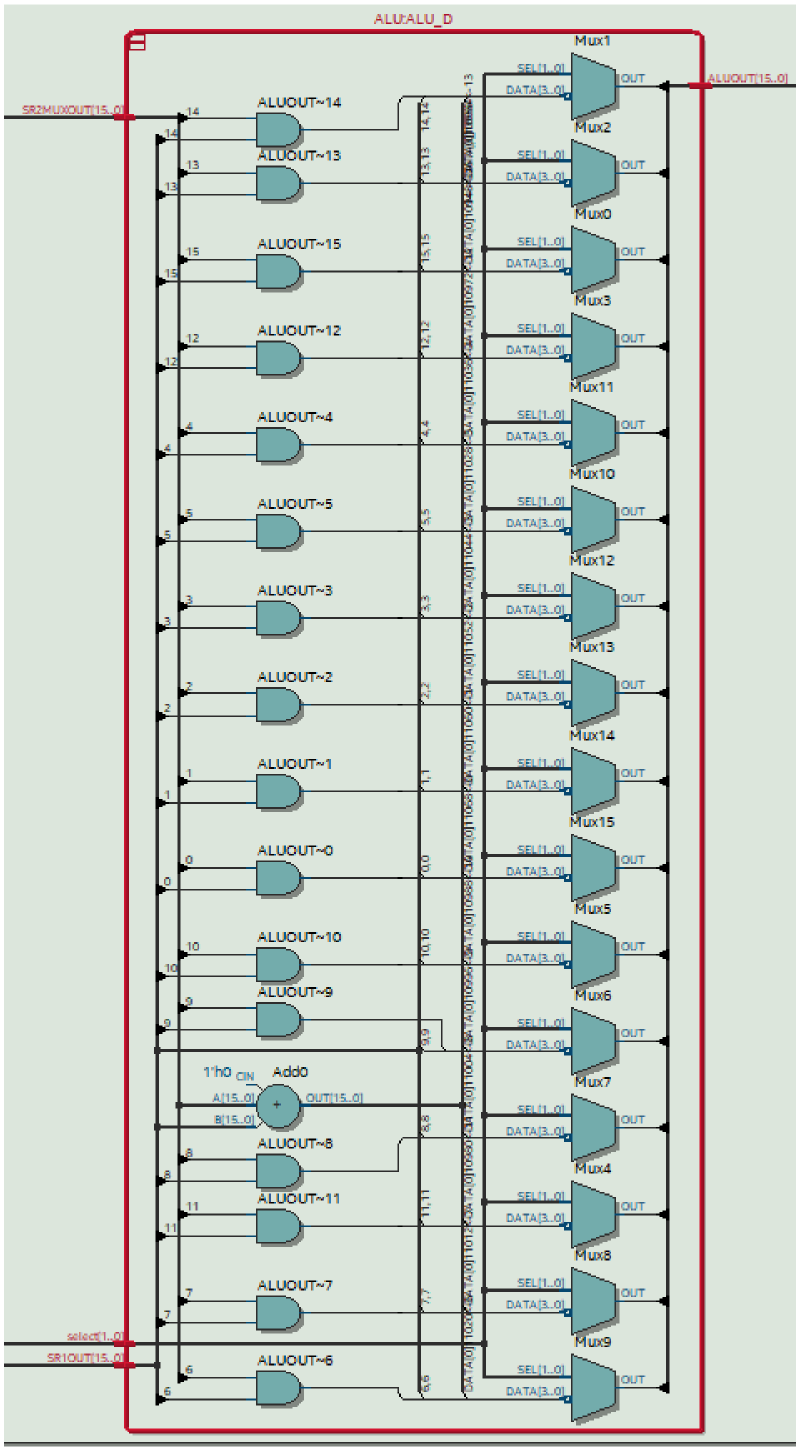


Figure 27: RTL Block diagram for ALU.sv