Zhihao Wang (zhihaow6), Laurenz Nava (lfnava2)

Introduction

The SLC-3 circuit

Written Description and Diagrams of SLC-3

Summary of Operations

The SLC-3 circuit is able to read instructions from memory (fetch), interpret what action it should perform (decode), and complete said action based on other parameters the instruction specifies (execute). There are 8 registers (0-7) to hold and manipulate data and one memory address set aside for memory mapped IO that supplement some of the instructions. The instructions the SLC-3 are able to perform include:

ADDing the value of two registers or a register and a sign extended 5-bit signed number and storing it in a register,

ANDing the value of two registers or a register and a sign extended 5-bit number and storing it in a register,

storing the NOT value of a register in to a register,

BRanching to another point of memory based on the last used value, the next position of the memory pointer, and a sign extended 9-bit number,

JuMPing to another point in memory based on the value in a register,

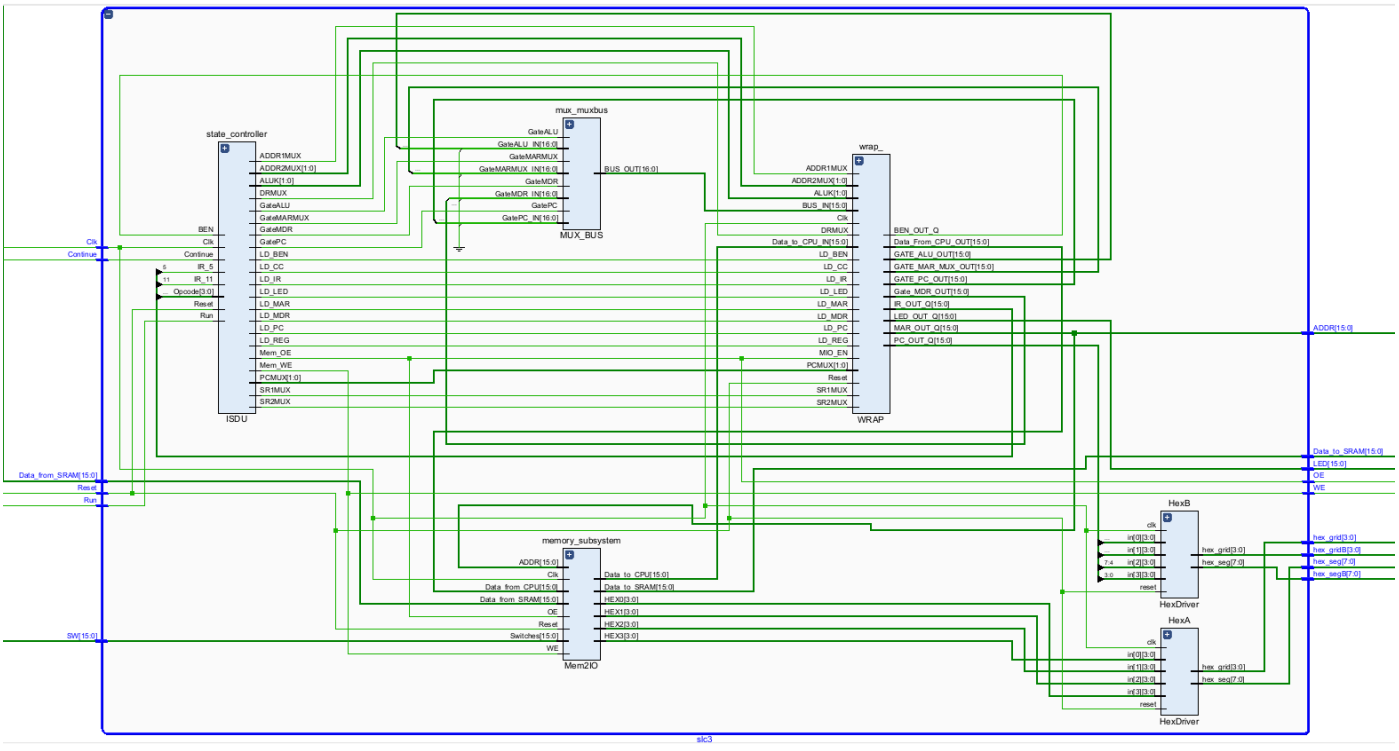
JSR which stores the next position of the memory pointer in register 7 and moves to a memory location sign extended 11-bits away from the next position,

LoaDing from Register which reads a memory location based on the sum of a value of a register and a sign extended 6-bit number and puts result into a register,

SToring to a Register which stores the value in a register into a memory location based on the sum of a value of a register and a sign extended 6-bit number,

PAUSE halts the SLC-3 operation until the continue button is pressed, and lights the LEDs with a 12-bit number

SLC-3 Block Diagram



Written Description of SV files

Module: slc3_sramtop.sv
Inputs: [15:0] SW, Clk, Reset, Run, Continue
Outputs: [15:0] LED, [7:0] hex_seg, [3:0] hex_grid, [7:0] hex_segB, [3:0] hex_gridB
Description: Takes inputs from switches to act as the initial memory value and starts there to perform the instructions listed in the internal instantiated ram
Purpose: The FPGA board implementation of the SLC-3 circuit

Module: slc3.sv
Inputs: [15:0] SW, Clk, Reset, Run, Continue, [15:0] DATA_from_SRAM
Outputs: [15:0] LED, OE, WE, [7:0] hex_seg, [3:0] hex_grid, [7:0] hex_segB, [3:0] hex_gridB, [15:0] ADDR, [15:0] Data_to_SRAM
Description: Interprets
Purpose:

Module: Hex Driver, hex.sv
Inputs: clk, reset, [3:0] in[4]
Outputs: [7:0] hex_seg, [3:0] hex_grid
Description: Convert input data into displayable data, then display on 7 segment display
Purpose: Display input data on 7 segment display

Module: nibble to hex, hex.sv
Inputs: [3:0] nibble
Outputs: [7:0] hex
Description: Turns the 4-bit nibble input into an 8-bit output readable by the hex segments to display the hexadecimal value of the input
Purpose: Convert binary input data into displayable on 7 segment display output data

Module: Mem2IO.sv
Inputs: Clk, Reset, [15:0] ADDR, OE, WE, [15:0] Switches, [15:0] Data_from_CPU, [15:0] Data_from_SRAM
Outputs: [15:0] Data_to_CPU, [15:0] Data_to_SRAM, [3:0] HEX0, [3:0] HEX1, [3:0] HEX2, [3:0] HEX3
Description: When trying to write to ADDR = xFFFF, we display the MDR data to the hex display, when trying to read from ADDR = xFFFF, we read data from switch to MDR.
Purpose: Core part for the SLC3 interacts with the outside world under the MMIO mode.

Module: 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, GateMARMUX, [1:0] PCMUX, DRMUX, SR1MUX, SR2MUX, ADDR1MUX, [1:0] ADDR2MUX, ALUK, Mem_OE, Mem_WE

Description: There is a common fetch and decode path that sets the PC, reads from memory the instruction, and decodes the instruction for the SLC-3 to perform. Each instruction has its own path after the initial fetch and decode that performs the action based on the other specifications the instruction makes, such as the DR, SR1, SR2, and immediate sign extended values. In the case of instructions LDR and STR, and load MAR into MDR, they are all implemented with 3 states total in place of an actual R signal, in order for the read or write from memory to a register is able to be read from the register for the next states. After an instruction is completed, it returns to the beginning of the fetch cycle to begin the next instruction.
Purpose: The ISDU manages all the states in the finite state machine of the SLC-3.

Module: wrap.sv
Inputs: Clk, Reset, [15:0] BUS_IN, [15:0] Data_to_CPU_IN, LD_MAR, LD_MDR, LD_IR, LD_BEN, LD_CC, LD_REG, LD_PC, LD_LED, SR2MUX, ADDR1MUX, MIO_EN, DRMUX, SR1MUX, [1:0] PCMUX, [1:0] ADDR2MUX, ALUK,
Outputs: [15:0] Gate_MDR_OUT, [15:0] Data_From_CPU_OUT, [15:0] MDR_OUT_Q, [15:0] MAR_OUT_Q, [15:0] GATE_ALU_OUT, [15:0] GATE_PC_OUT, PC_OUT_Q, [15:0] GATE_MAR_MUX_OUT, BEN_OUT_Q, [15:0] IR_OUT_Q, [15:0] LED_OUT_Q
Description: The major part of the SLC3 CPU, with reading from BUS, MEM2IO, and outputting to BUS and MEM2IO, with the control signal received from ISDU. It outputs the IR value which is the current decoded PC for ISDU, and outputs the LED value for the outside driver. The control unit is in ISDU.
Purpose: Connect the major component of the SLC3. From PC Reg, RegFile, SR1MUX, SR2MUX, ADDER, ALU.

Module: reg_16, register.sv
Inputs: Clk, Reset, Load_En, [15:0] Data_In_D
Outputs: [15:0] Data_Out_Q
Description: Synchronous reset 16-bit register with load enable, without load enable, output remains the same
Purpose: Stores 16-bit values used by SLC-3 such as MAR, MDR, IR, LED, and PC

Module: reg_nzp, register.sv
Inputs: Clk, Reset, Load_En, [15:0] Data_In_D
Outputs: [2:0] Data_Out_Q
Description: Synchronous reset 3-bit register with load enable, without load enable, output remains the same
remains the same. If Data_In_D interpreted as signed is negative, output 100, if Data_In_D is zero, output 010, and if Data_In_D is positive, output 001.
Purpose: Generate NZP value from input value

Module: reg_1, register.sv
Inputs: Clk, Reset, Load_En,
Outputs: Data_Out_Q

Description: Synchronous reset 1-bit register with load enable, without load enable, output remains the same
Purpose: Stores the BEN for BR

Module: regfile.sv
Inputs: Clk, Reset, LD_REG, [2:0] DRMUX_IN, [2:0] SR1MUX_IN, [2:0] SR2_IN, [15:0] BUS_IN
Outputs: [15:0] SR1_OUT_Q, [15:0] SR2_OUT_Q
Description: Active on LD_REG, selects the register for DR, SR1, and SR2 from respective MUX based on input
Purpose: Takes in input values to select appropriate path to specified registers

Module: MUX_2_16, mux.sv
Inputs: Select, [15:0] Z_IN, [15:0] O_IN
Outputs: [15:0] OUT
Description: Two to One mux, 16-bit inputs
Purpose: Used for MIO_EN for MDR, to choose between PC and ALU for ADDR1MUX, and to choose between SR2 or immediate value for SR2MUX

Module: MUX_4_16, mux.sv
Inputs: [1:0] Select, [15:0] ZZ_IN, [15:0] ZO_IN, [15:0] OZ_IN, [15:0] OO_IN
Outputs: [15:0] OUT
Description: Four to One mux, 16-bit inputs
Purpose: Used to choose between the various sign extended values for ADDR2

Module: MUX_2_3, mux.sv
Inputs: Select, [2:0] Z_IN, [2:0] O_IN
Outputs: [2:0] OUT
Description: Two to One mux, 3-bit inputs
Purpose: Used to choose the DR and SR1

Module: MUX_3_16, mux.sv
Inputs: [1:0] Select, [15:0] ZZ_IN, [15:0] ZO_IN, [15:0] OZ_IN
Outputs: [15:0] OUT
Description: Three to One mux, 16-bit inputs
Purpose: Used to choose if PC increments by 1, by the sum of some values, from the BUS

Module: MUX_BUS, mux.sv
Inputs: GatePC, GateMDR, GateALU, GateMARMUX, [16:0] GatePC_IN, [16:0] GateMDR_IN, [16:0] GateALU_IN, [16:0] GateMARMUX_IN,
Outputs: [16:0] BUS_OUT
Description: BUS_OUT xZZZZ by default, if any of the Gate signals is 1, connects BUS_OUT with corresponding Gate_IN

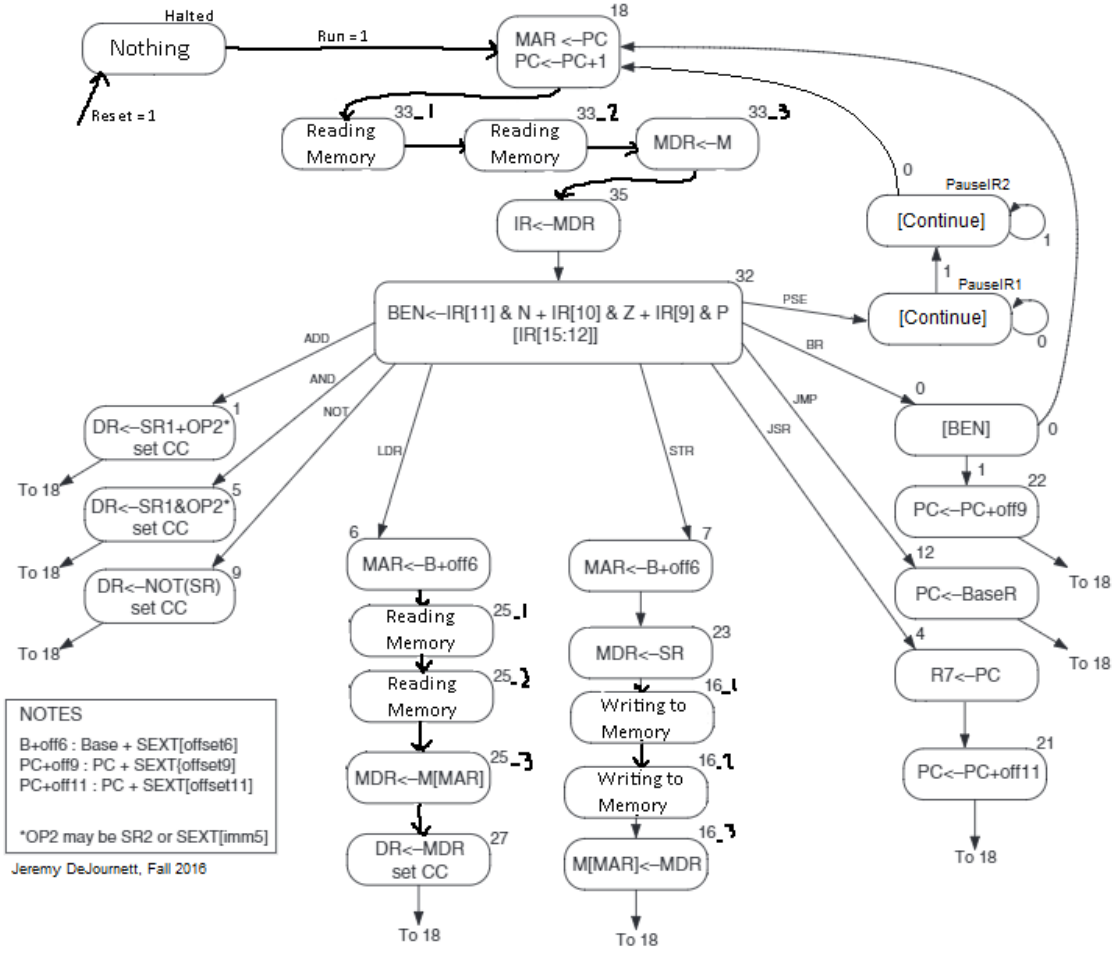
Purpose: Acts as the tristate buffers as there are none on the FPGA

Module: Instantiatteram.sv
Inputs: Reset, Clk
Outputs: [15:0] ADDR, wren, [15:0] data
Description: Instantiated ram to hold all the instructions
Purpose: Contains the digital information of the instructions for the tests

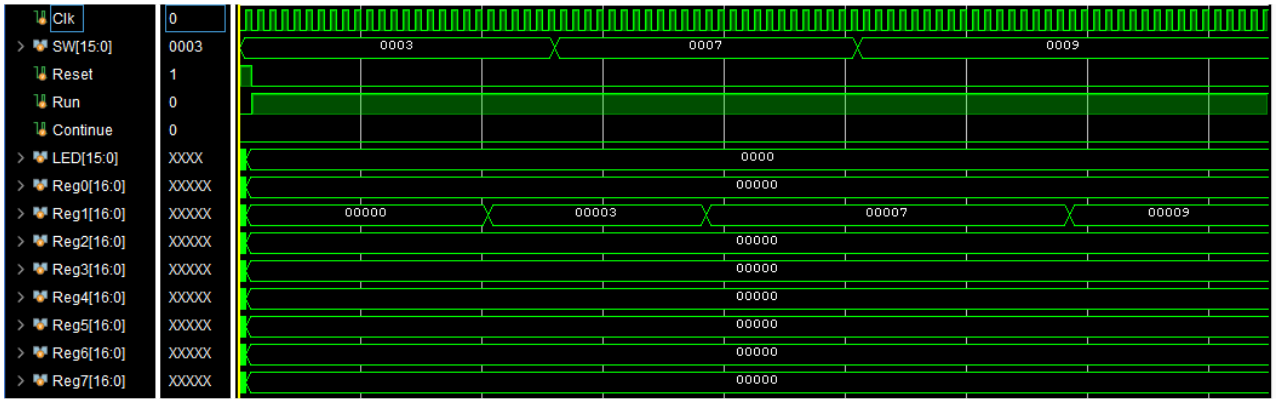
Module: blk_mem_gen_0.xci
Inputs: ADDR[9:0], Clk, [15:0] Data_to_SRAM, OE, WE
Outputs: [15:0] Data_from_SRAM
Description: IP for 16x1000 memory storage
Purpose: The physical implementation of memory for FPGA

Module: sync, Synchronizers.sv
Inputs: Clk, d
Outputs: q
Description: Synchronous update of q to d
Purpose: Synchronize inputs from physical devices, switches and buttons, to internal timings

State Diagram for ISDU

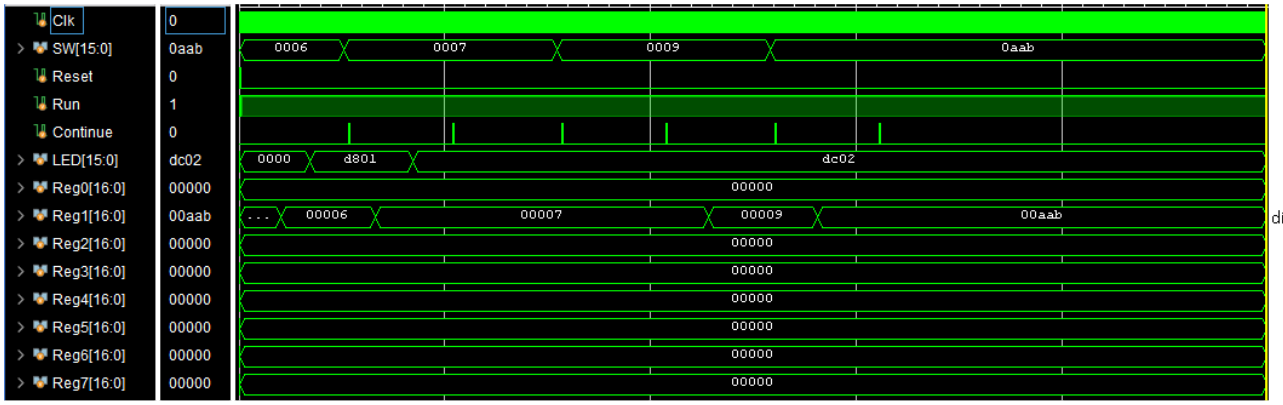


Simulations of SLC-3 Instructions



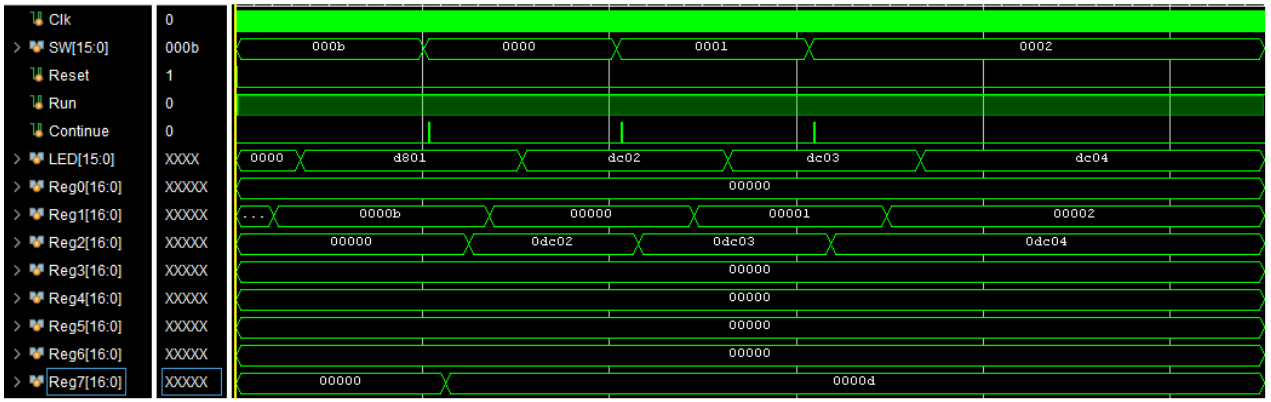
IO Test #1

displayed output



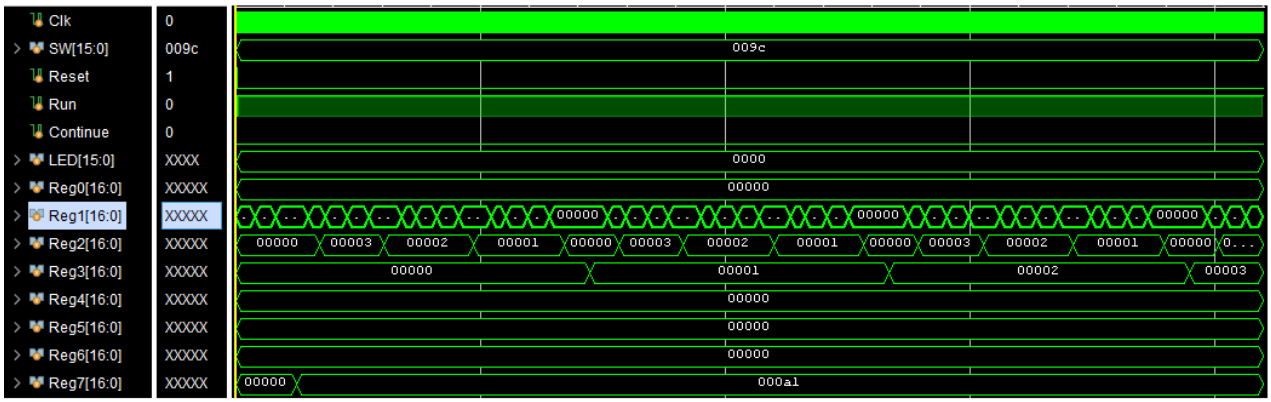
IO Test #2

displayed output



Self Modfying Code

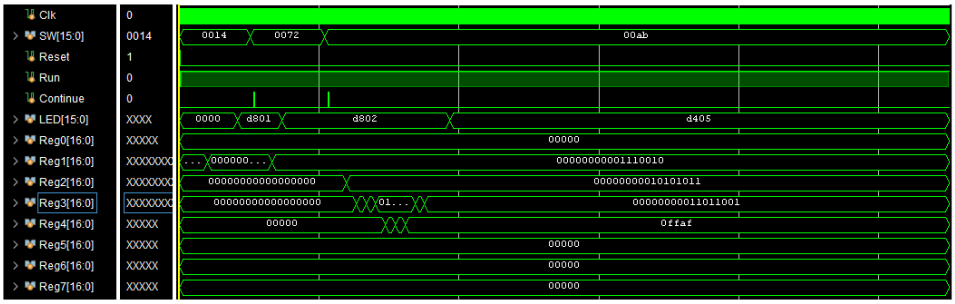
displayed output

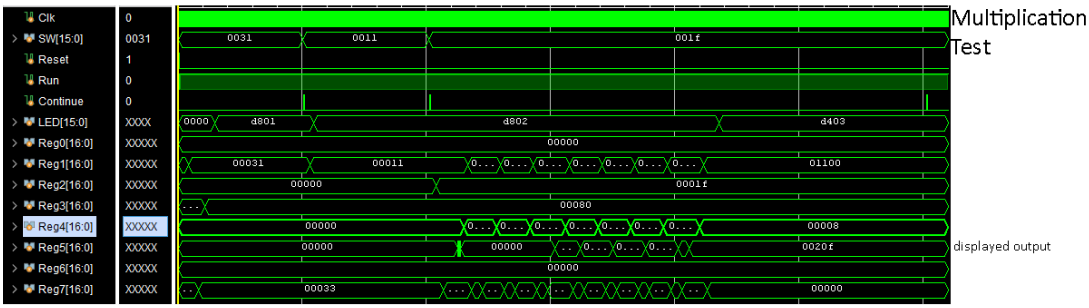


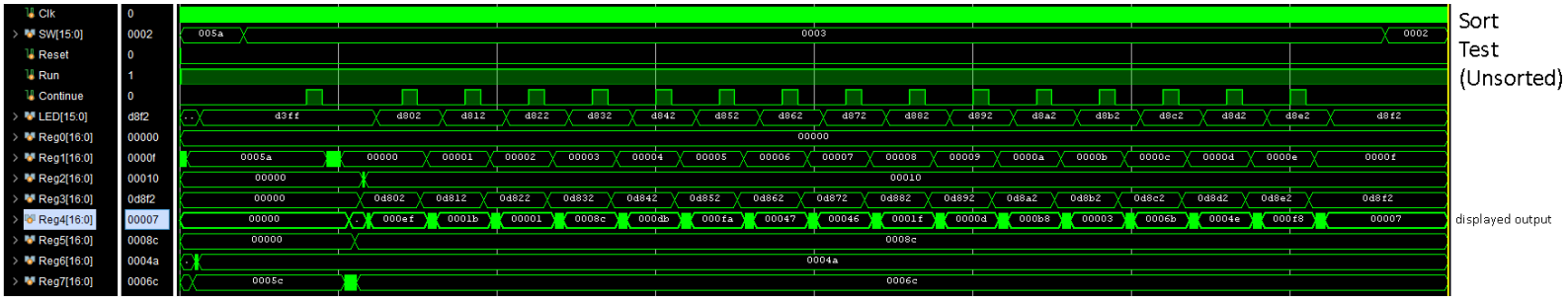
Auto Counting Test

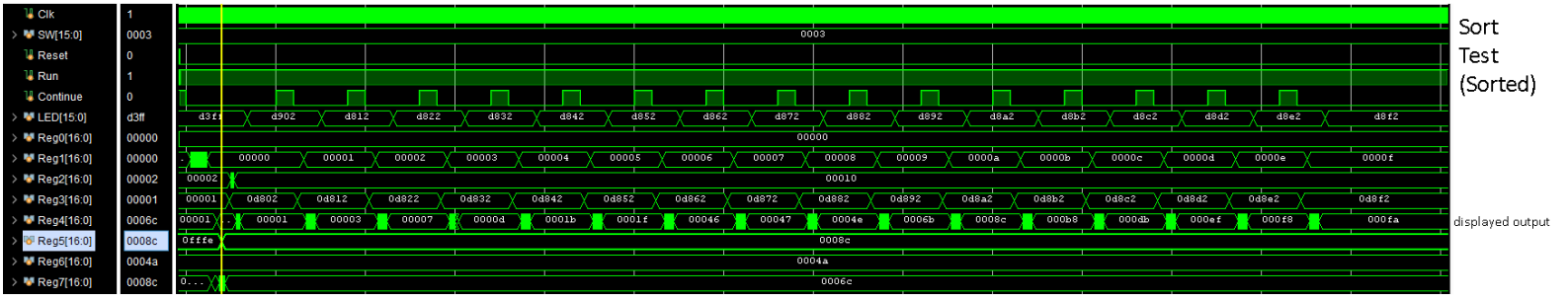
Loops 3->2->1->0->3

displayed output


Clk
SW[15:0]
Reset
Run
Continue
LED[15:0]
Reg0[16:0]
Reg1[16:0]
Reg2[16:0]
Reg3[16:0]
Reg4[16:0]
Reg5[16:0]
Reg6[16:0]
Reg7[16:0]
XOR
Test
displayed output
R3 = R1 XOR R2


Clk
SW[15:0]
Reset
Run
Continue
LED[15:0]
Reg0[16:0]
Reg1[16:0]
Reg2[16:0]
Reg3[16:0]
Reg4[16:0]
Reg5[16:0]
Reg6[16:0]
Reg7[16:0]
Multiplication
Test
displayed output


Clk
SW[15:0]
Reset
Run
Continue
LED[15:0]
Reg0[16:0]
Reg1[16:0]
Reg2[16:0]
Reg3[16:0]
Reg4[16:0]
Reg5[16:0]
Reg6[16:0]
Reg7[16:0]
Sort
Test
(Unsorted)
displayed output


Clk
SW[15:0]
Reset
Run
Continue
LED[15:0]
Reg0[16:0]
Reg1[16:0]
Reg2[16:0]
Reg3[16:0]
Reg4[16:0]
Reg5[16:0]
Reg6[16:0]
Reg7[16:0]
Sort
Test
(Sorted)
displayed output

Post-lab Questions

LUT	492
DSP	0
Memory (BRAM)	0.5
Flip-Flop	278
Latches	0
Frequency	100.614 MHz (0.061 ns WNS)
Static Power	0.071 W
Dynamic Power	0.013 W
Total Power	0.083 W*

*I know the total power doesn't add up, it's just what was shown on the power report, must be something with the rounding

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

The main function for MEM2IO is, when the designated address is xFFFF, the FPGA utilizes memory mapped input/output to interpret this as either to display on the data on the hex displays for a write to(OE and WE are high) Hex_Driver, or to receive input(OE is high) from the Switches for a read from. In conclusion, we use MEM2IO to realize a simple MMIO struct.

What is the difference between BR and JMP instructions?

BR is different from JMP in that BR can be conditional with its reliance on nzp for BEN, as well as changing PC with a signed 9-bit where as JMP is unconditional and changes PC based on a value from a register. This makes BR easier to use to just move around, since it only needs an immediate value instead of allocating a register for JMP, but for something to return to or longer distances, JMP would be better since it has access to all 16-bits for any address and does not need to be changed depending on the current PC.

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 in Patt and Patel is to notify the computer that the read or write to memory has been completed and the next state can begin. Since we do not have such a signal, it is compensated by having 2 additional, for 3 total, states for any read or write, in the cases of LDR, STR, and MDR from memory. This allows time for the the memory to be accessed (1 state), the memory to be input into the register (1 state), and the input data to reach the output (1 state). This means the system stays synchronized so long as memory access does not take longer than a clock cycle, though it can get out of sync if it does.

Conclusions

The design worked to specifications, though in implementation on FPGA, there were some issues that occurred with bouncing switches for the Sort test. This is a known issue and is acceptable. The lab manual and other assorted materials were sufficient for completing the lab, and the state and block diagrams were particularly helpful in designing and understanding the system as a physical object.