ECE 385

Spring 2020

Experiment 6

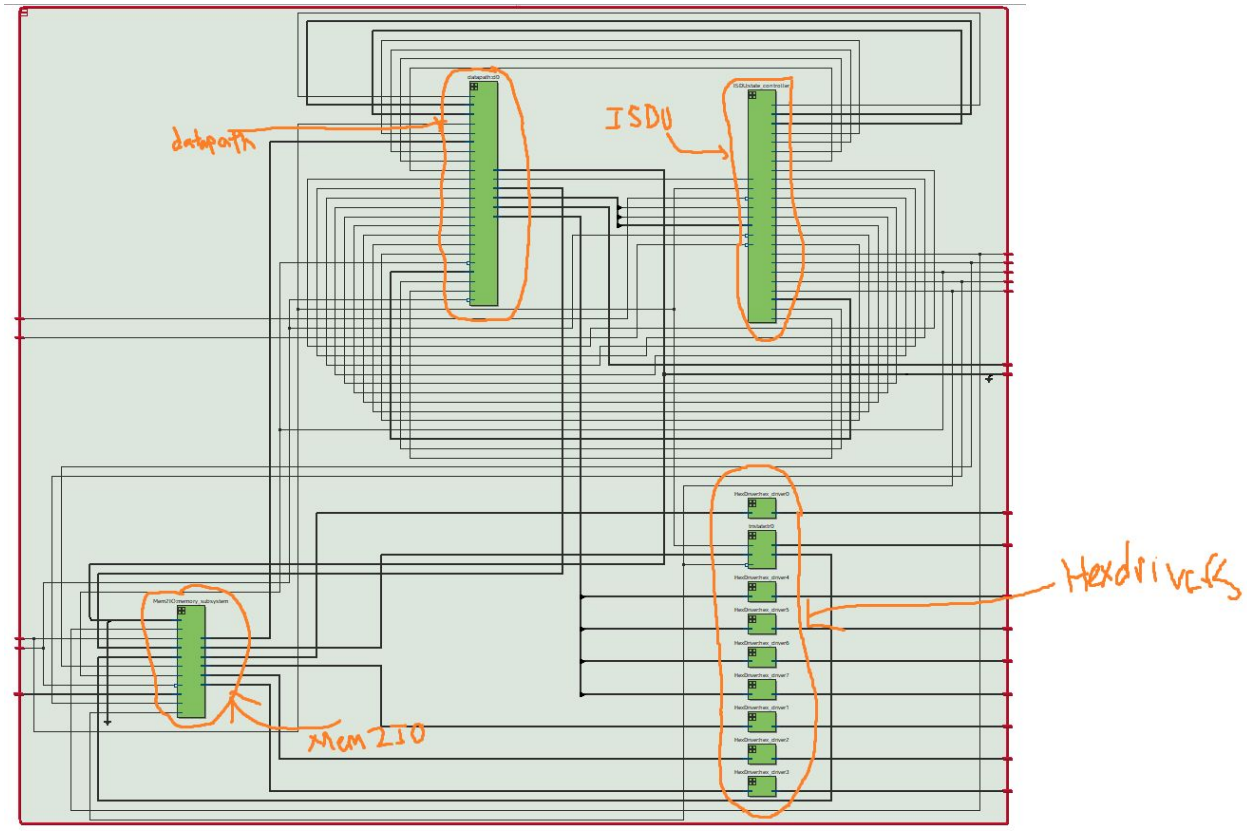
Simple Computer SLC-3.2 in SystemVerilog

Michael Faitz and Zohair Ahmed
Section ABJ: Friday 2:00-4:50
Yuming Wu and Lian Yu

1. **Introduction**

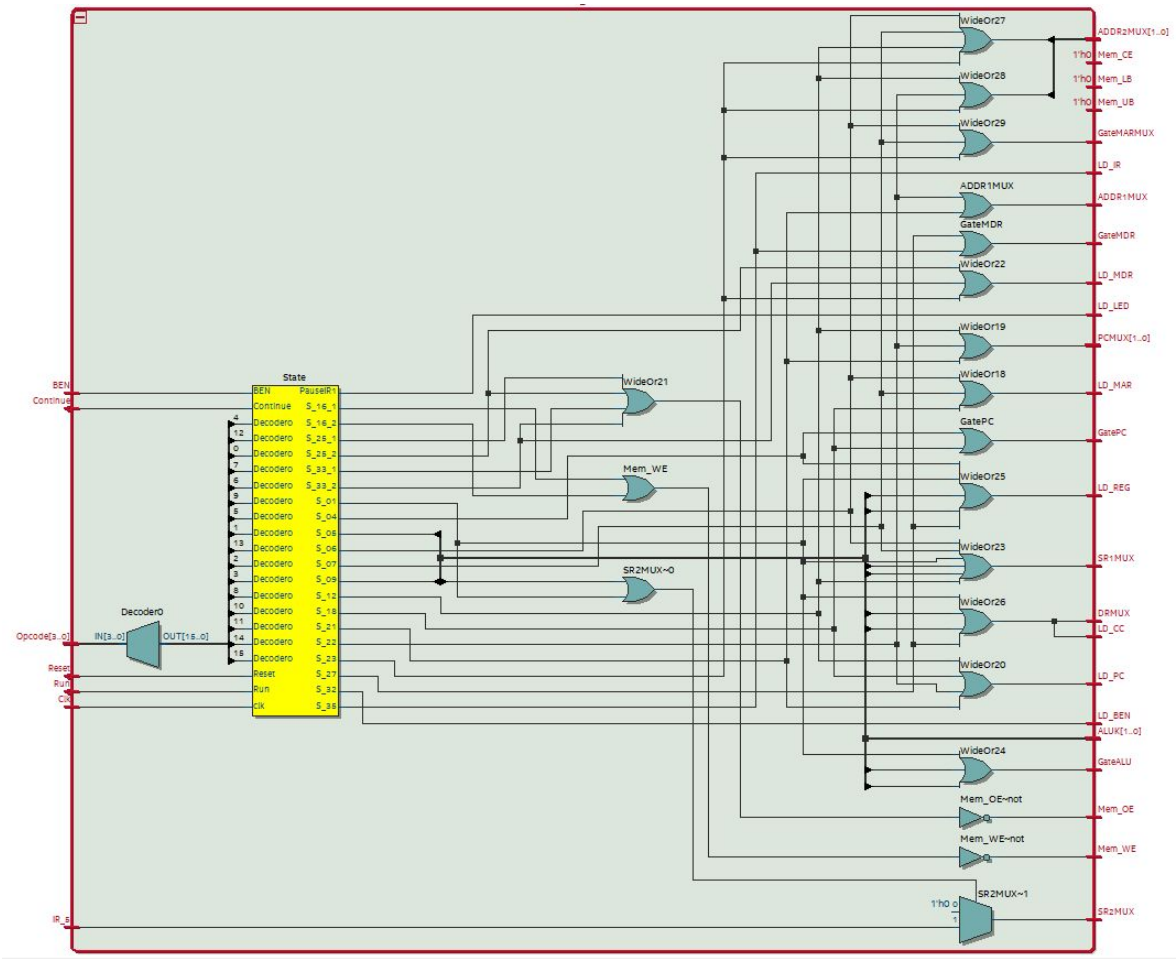
   The SLC-3 microprocessor functions similarly to the LC-3 architecture. It takes a list of instructions from a separate file (in this case, RAM) and executes them with the help of data registers. It is capable of arithmetic and logic functions, can load and store data, change the instruction the user wishes to execute next, and can be paused.

2. **Written Description and Diagrams of SLC-3**
   a. The SLC3 reads in data from a memory file, moves it to a register to handle instruction decoding, then executes said instruction. It can also write to memory, perform data loads, stores, and logic operations.
   b. The SLC-3 takes instructions from a separate RAM file. First, it fetches the instruction at the address given to it by the switches, before updating the program counter to move to the next instruction. Then, it sends the instruction to the instruction register. After that, the IR decodes the instruction. It checks the first 4 bits to determine what the instruction is, with the rest of the bits used for relevant parameters for the function. Then, it executes the instruction. It can execute addition and basic logic functions, memory loading and storing, instruction address changing, and pausing. At the end of all of these instructions, the microprocessor will gather the next instruction from the SRAM and repeat these actions.
   c. Block diagram of slc3.sv



d. Written Description of all .sv modules

Module: ISDU
Inputs: Clk, [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, [1:0] ALUK, Mem_CE, Mem_UB, Mem_LB, Mem_OE, Mem_WE
Description: This module handles our state machine, load, MUX, and gate variables.
Purpose: Updates control signals to allow the datapath and other modules to function and pass data as desired.



Module: lab6_toplevel
Inputs: [15:0] S, Clk, Reset, Run, Continue
Outputs: [11:0] LED, [6:0] HEX0, [6:0] HEX1, [6:0] HEX2, [6:0] HEX3, [6:0] HEX4, [6:0] HEX5, [6:0] HEX6, [6:0] HEX7, CE, UB, LB, OE, WE, [19:0] ADDR, [15:0] Data
Description: Instantiates an instance of slc3 as well as test_memory and allows for the linking of the two modules.

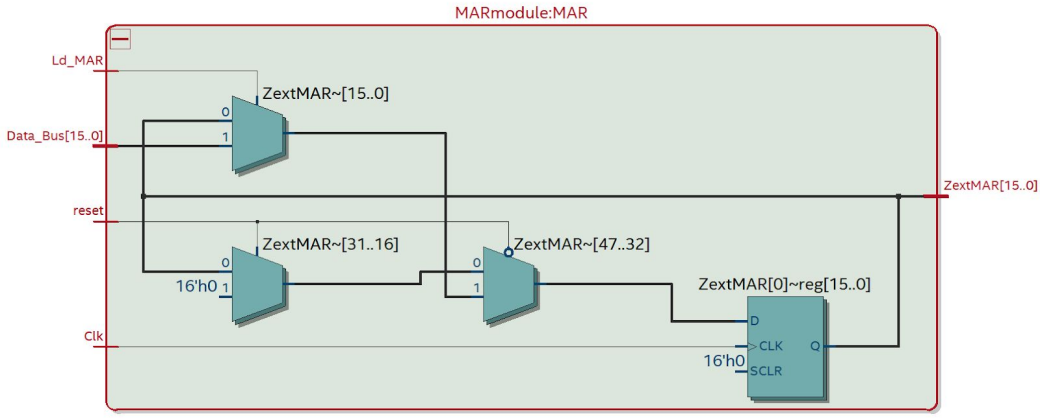
Purpose: Serves as the link between the slc3 top level and the available test_memory.

Module: Mem2IO
Inputs: Clk, Reset, [19:0] ADDR, CE, UB, LB, 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: Inputs the address from the MAR and can write and read memory based on the address and the WE and OE bits in the ISDU.
Purpose: Serves as the link between the datapath and the memory contents and allows for modification of memory outside of just registers.

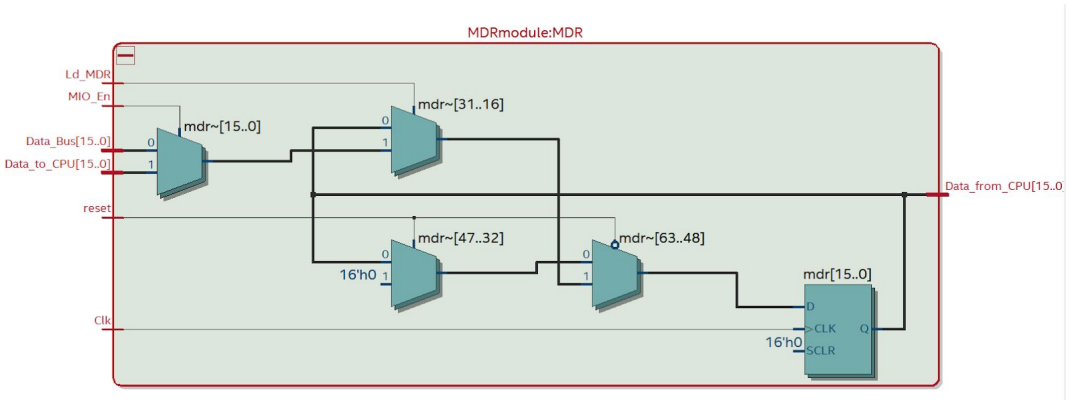
Module: slc3
Inputs: [15:0] S, Clk, Reset, Run, Continue
Outputs: [11:0] LED, [6:0] HEX0, [6:0] HEX1, [6:0] HEX2, [6:0] HEX3, [6:0] HEX4, [6:0] HEX5, [6:0] HEX6, [6:0] HEX7, CE, UB, LB, OE, WE, [19:0] ADDR, [15:0] Data
Description: Instantiates datapath, MEM2IO, the ISDU, and several other modules and serves as the central glue to hold them all together using the same variable names.
Purpose: Glues all the elements of the device together.

Module: tristate
Inputs: Clk, tristate_output_enable, [N-1:0] Data_write,
Outputs: [N-1:0] Data_read, [N-1:0] Data
Description: Models a tristate buffer to allow data reads and writes to the SRAM.
Purpose: Used to transmit data to and from MEM2IO and SRAM.

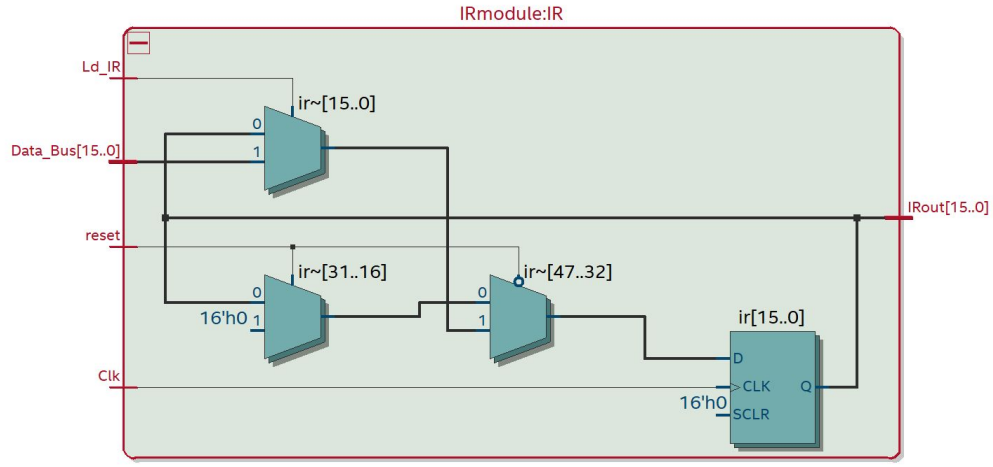
Module: MARmodule
Inputs: [15:0] Data_Bus, Clk, Ld_MAR, reset
Outputs: [15:0] ZextMAR
Description: Module that contains MAR.
Purpose: Used to hold the address of the SRAM when reading from or writing to it.



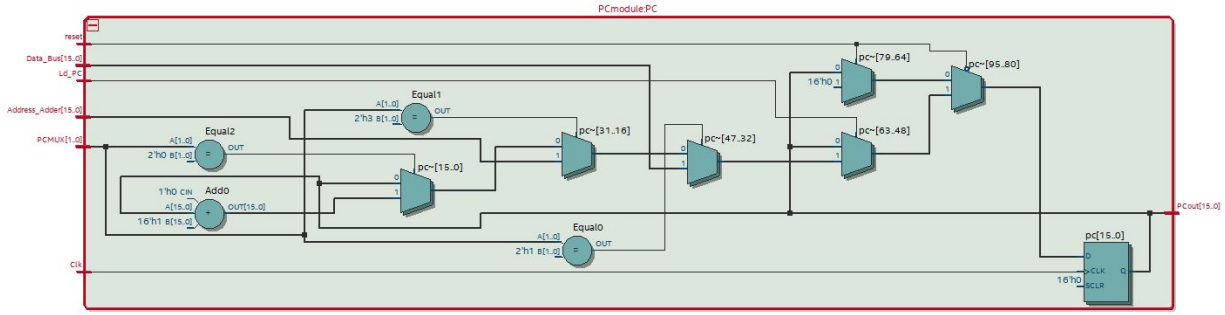
Module: MDRmodule
Inputs: [15:0] Data_to_CPU, Clk, Ld_MDR, [15:0] Data_Bus, MIO_En, reset
Outputs: [15:0] Data_from_CPU
Description: Module that controls the MDR.
Purpose: Used to hold the data that is read from or used to write to SRAM.



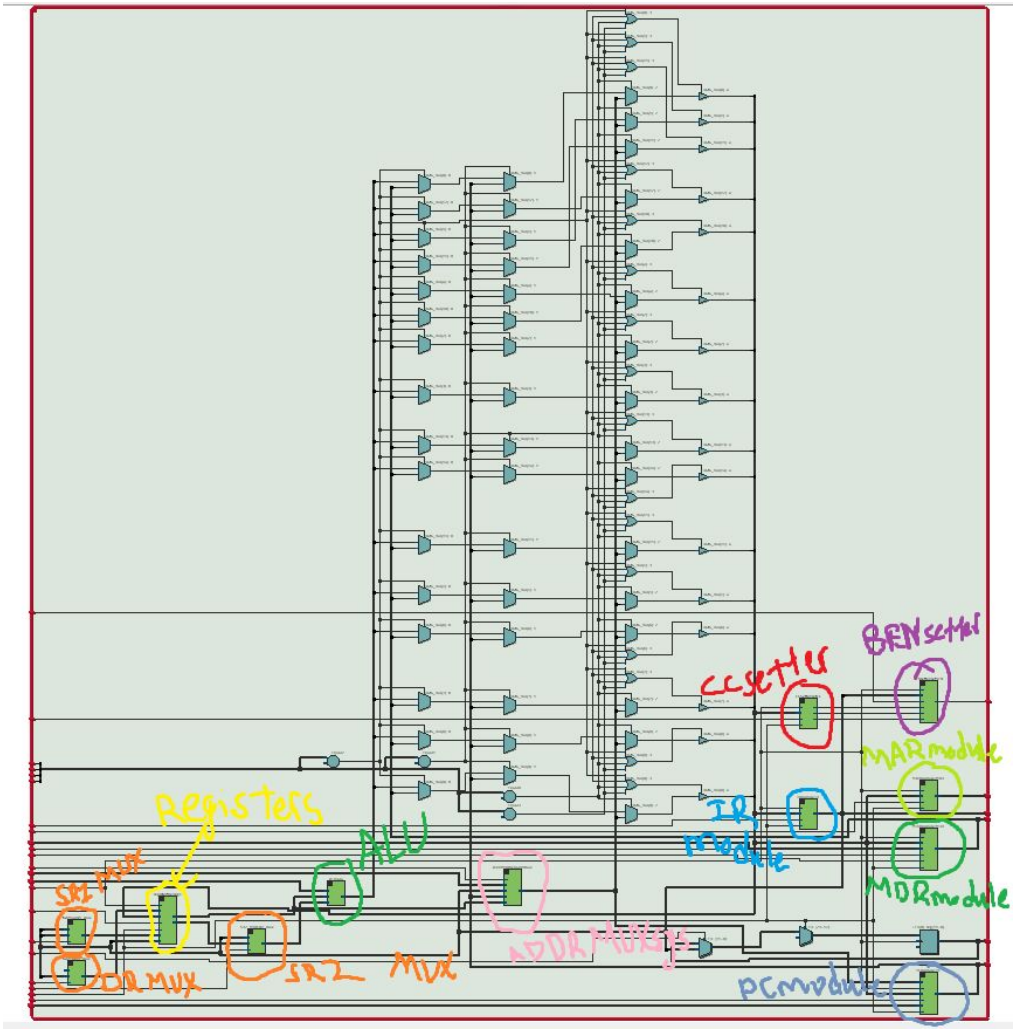
Module: IRmodule
Inputs: [15:0] Data_Bus, Clk, Ld_IR, reset,
Outputs: [15:0] IRout
Description: Module that controls the instruction register.
Purpose: Used in the decode state to determine which instruction to perform, along with other necessary bits as parameters.



Module: PCmodule
Inputs: [15:0] Data_Bus, Clk, [15:0] Address_Adder, Ld_PC, [1:0] PCMUX, reset
Outputs: [15:0] PCout
Description: Contains the PC register, along with all associated MUXes, select bits, LD signals, and the data bus input.
Purpose: Used to increment the PC or load a specific value into it, based on given signals.

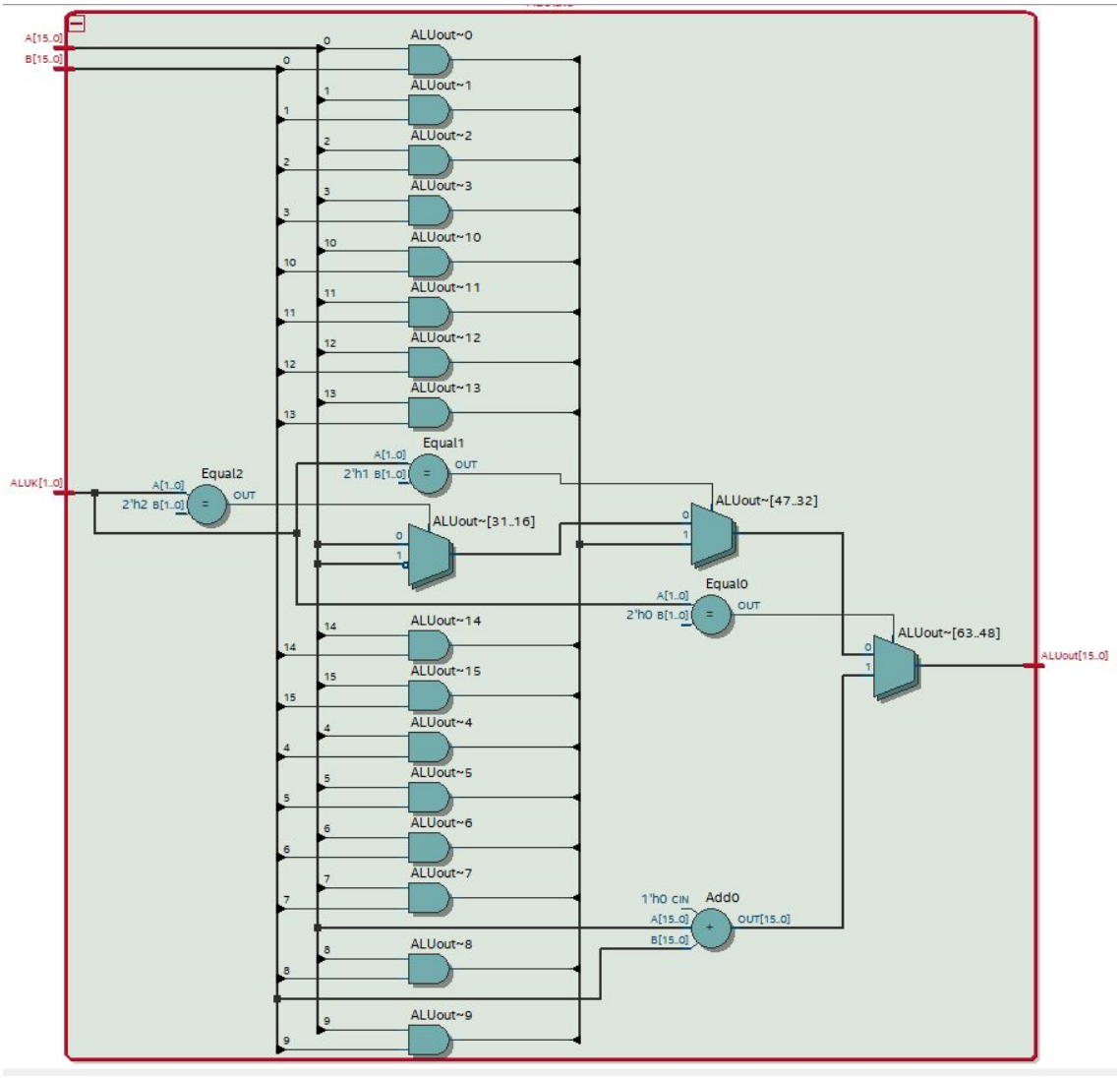


Module: datapath
Inputs: GateMARMUX, GatePC, GateMDR, GateALU, Clk, reset, LD_MAR, LD_MDR, LD_IR, LD_BEN, LD_CC, LD_REG, LD_PC, LD_LED, [1:0] PCMUX, [1:0] ADDR2MUX, [1:0] ALUK, DRMUX, SR1MUX, SR2MUX, ADDR1MUX, MIO_EN, [15:0] Data_to_CPU
Outputs: BEN, [11:0] LED, [15:0] Data_from_CPU, [15:0] IRout, [15:0] ADDR, [15:0] PCout
Description: Holds all registers, gates, MUXes, LD and MUX select bits, and the data bus.
Purpose: Given signals from ISDU, performs all operations associated with those signals and outputs the results.

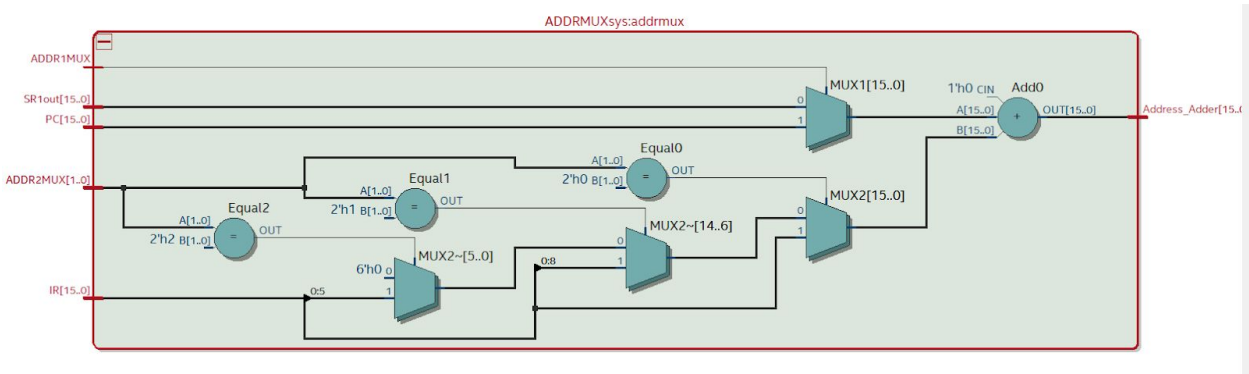


Module: HexDriver
Inputs: [3:0] In0
Outputs: [6:0] Out0
Description: Used to convert 4 bit data inputs into a 7 bit output.
Purpose: Used to light up hex displays to properly output a group of 4 bits.

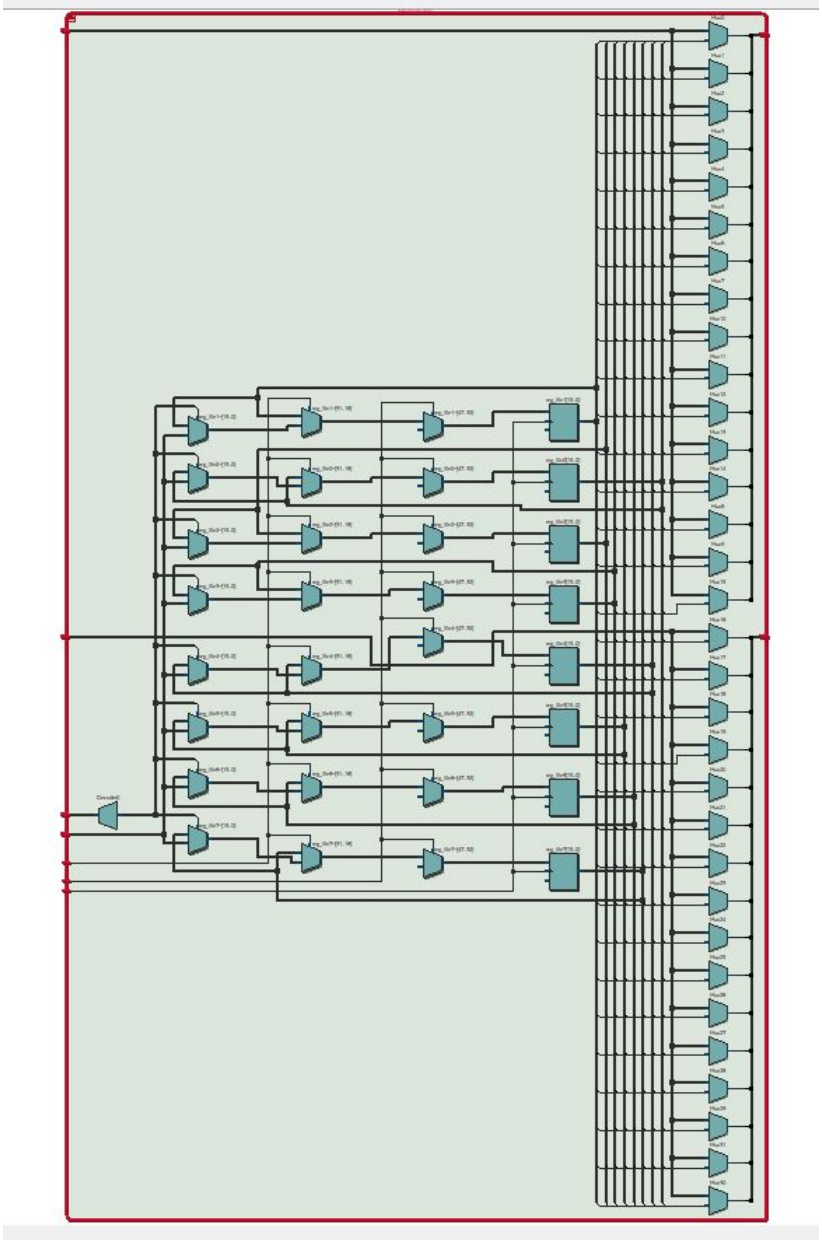
Module: ALU
Inputs: [1:0] ALUK, [15:0] A, B
Outputs: [15:0] ALUout
Description: An addition and logic unit that performs ADD, AND, NOT, and pass A through it, depending on the value in ALUK.
Purpose: Used for addition and logic instructions, as well as passing A through it for certain states.



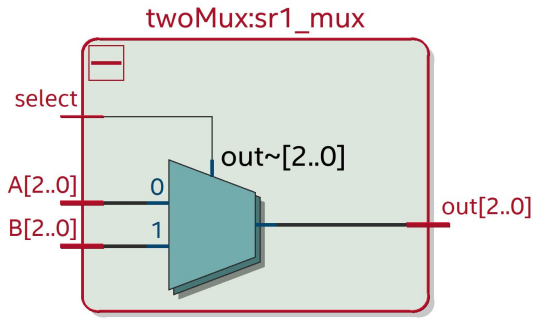
Module: ADDRMUXsys
Inputs: [1:0] ADDR2MUX, ADDR1MUX, [15:0] IR, [15:0] SR1out, [15:0] PC
Outputs: [15:0] Address_Adder
Description: A system of MUXes that take in data form the IR, SR1out, and PC and uses the results as operands in an adder. Outputs the result as Address_Adder.
Purpose: Used in instructions and operations that require adding the PC or SR1out to a sign extension in the IR.

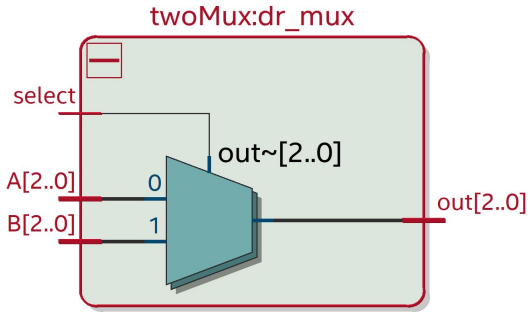


Module: registerfile
Inputs: Reset, Clk, [15:0] D_in, LD_REG, [2:0] DR_MUX_out, [2:0] SR1_MUX_out, [2:0] SR2_in
Outputs: [15:0] SR1_out, [15:0] SR2_out
Description: Module that holds 8 data registers.
Purpose: Used in jump, load, store, addition, logic, operations and storage of data.

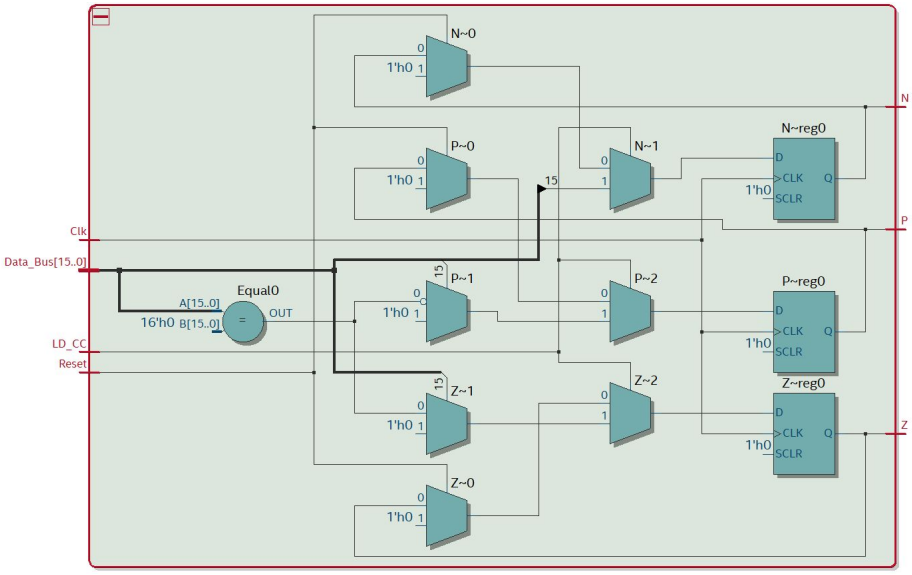


Module: twoMUX
Inputs: [2:0] A, [2:0] B, select
Outputs: [2:0] out
Description: A basic 2-1 MUX that handles 3 bit words.
Purpose: Used to determine address for DRMUX and SR1MUX.



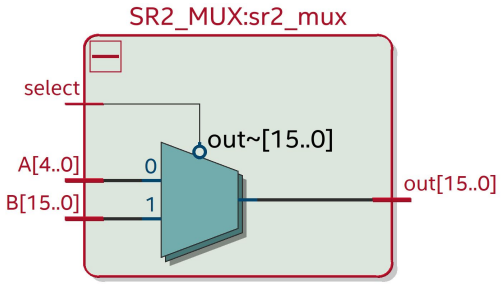


Module: CCsetter
Inputs: Clk, [15:0] Data_Bus, LD_CC, Reset
Outputs: N, Z, P
Description: Takes data from the bus and confirms whether the data is positive, negative or zero.
Purpose: Updates condition codes N, Z, P such that BEN can be correctly set for a BR.

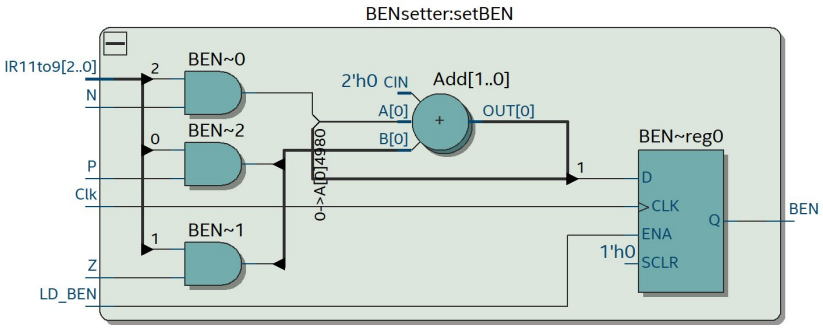


Module: SR2_MUX
Inputs: [4:0] A, [15:0] B, select
Outputs: [15:0] out

Description: Takes input from IR[4:0], SR2 from the register file, and determines which to use in an ALU operation.
Purpose: Used when using AND or ADD to use sign extended bits or the second register.



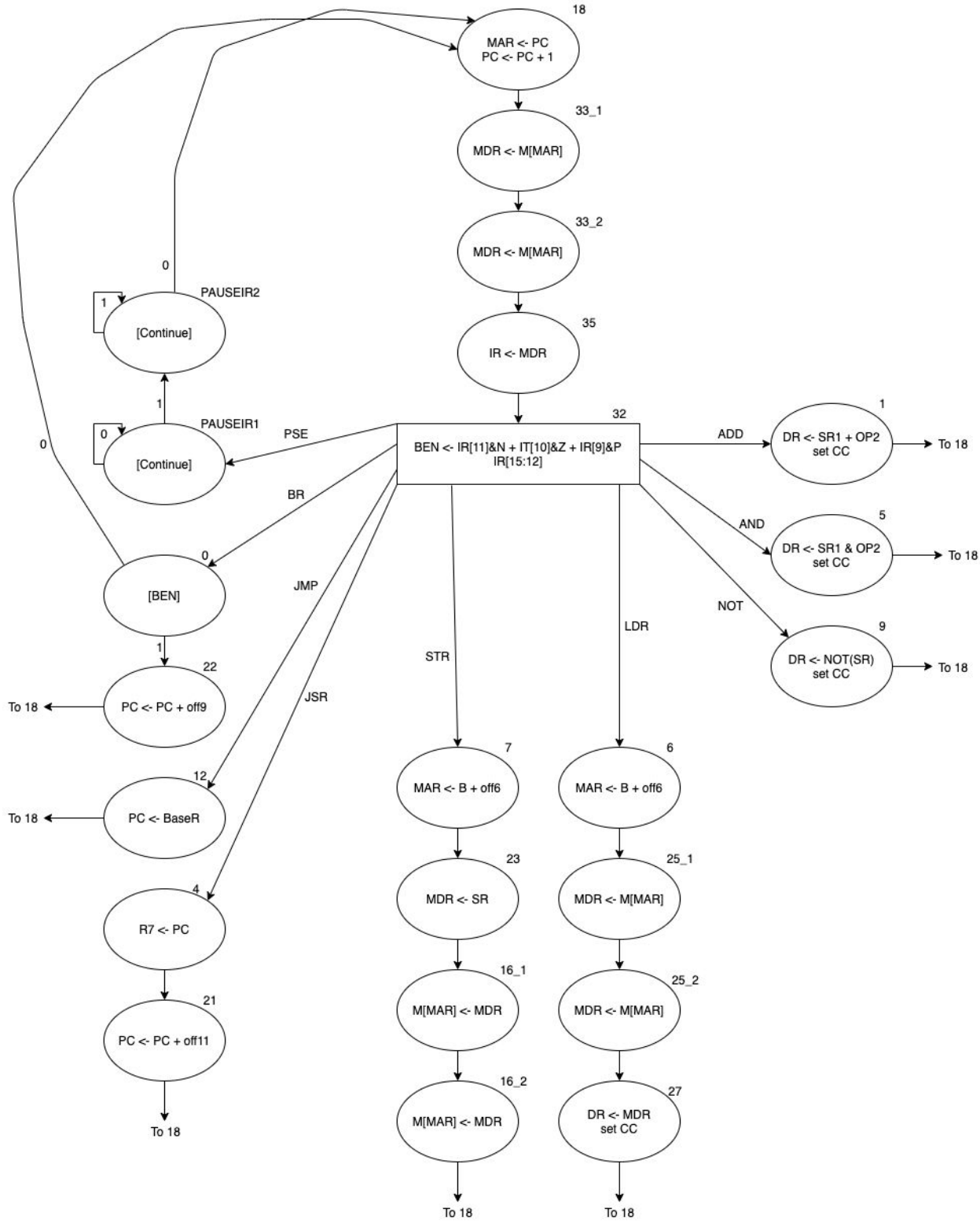
Module: BENsetter
Inputs: [2:0] IR11to9, N, Z, P, Clk, LD_BEN,
Outputs: BEN
Description: Uses bits IR[11:9] as well as N, Z, P bits. ands them, and sets BEN to the result.
Purpose: Used to determine whether or not to execute BR instruction.



e. Description of the operation of the ISDU

Q: 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.
A: The ISDU controls the various components of the SLC-3 through various enable bits that are set in each of the states. Below is the state diagram of the ISDU which shows the overarching goals of each state which is accomplished via the various enabled bit toggles. Examples of these enable bits are for selecting MUX values, loading the datapath, or adjusting memory values.

f. State Diagram of ISDU

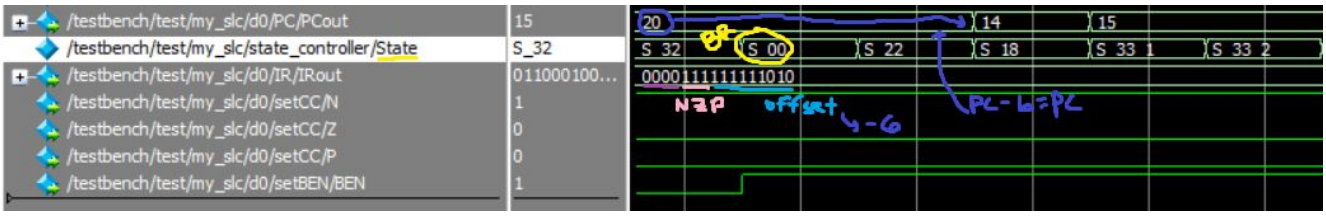


g.

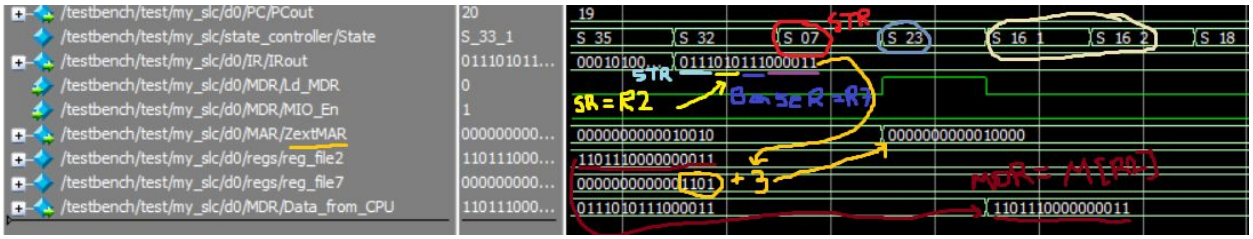
3. Simulations of SLC-3 Instructions

Example of Add

Example of BR



Example of STR



4. Post-Lab Questions

Design Resource Table

LUT	549
DSP	0
Memory (BRAM)	0
Flip-Flop	280
Frequency	65.48 MHz
Static Power	98.65 mW
Dynamic Power	8.04 mW
Total Power	178.67 mW

a. Answer the post-lab questions

i. MEM2IO handles the reading and writing between switches, the SRAM, MAR and MDR as well as the LED display.

ii. The difference between BR and JMP instructions is that JMP is unconditional, while BR is not. BR relies on the condition codes matching the required branch nzp bits, while JMP will alter the PC every time, regardless of condition codes.

iii. The R signal in Patt and Patel is there to confirm that a full data read or write has occurred. We compensate for the lack of this signal by adding additional states to instructions that require reading or writing, to account for the longest possible wait time. This should have minimal impact on synchronization, as no other actions need to happen as we wait, and we do not change the clock cycle to accomplish this.

5. Conclusion

a. For our design, we had most of our modules created in the datapath. This included MAR, MDR, IR, PC, various MUXes and adders, and our registers. It was helpful to have everything in one place, and we were able to leave the control signals to the ISDU. This approach worked well for us. We passed every test case, so there isn't much we would change.

b. The only aspects of this lab that we would suggest be changed are the explanations of the provided code in the beginning as there is a significant learning curve initially when starting the lab. Another thing that needs to be fixed is the reliability of the Control Panel for flashing the board. The remainder of the lab was well conducted and clear.