ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E
COMPUTER ORGANIZATION
PROJECT REPORT

PROJECT NO	:	1
PROJECT DATE	:	11.05.2022
LECTURE SESSION	:	THURSDAY - 8.30

GROUP MEMBERS:

040190218	:	MEHMET EYMEN & ÜNAY
150190049	:	EMRE & ÇELİK
150210735	:	ENES SEFA & ÇETİN

SPRING 2021

Contents

1 INTRODUCTION [10 points]

Every basic computer has its foundations on a system based on data retention and calculation. Registers, ALU and memory were sufficient to gather a basic computer system. In this project we implemented given functions with Verilog Hardware Description Language which we are using in Computer Organization(BLG222E) lecture. There are 4 parts in Project1 and we need to implement design these register and write simulation code to simulate our designs.

We are using Vivado environment to simulate these designs and compile our code to see errors and warnings.

2 PROJECT [40 points]

2.1 PART 1

We need to design 2 different register that are 8-bit register and 16-bit register. These register are going to have 4 different functionalities and these functionalities will be controlled by 2-bit control signals(FunSel) and an enable input(E).

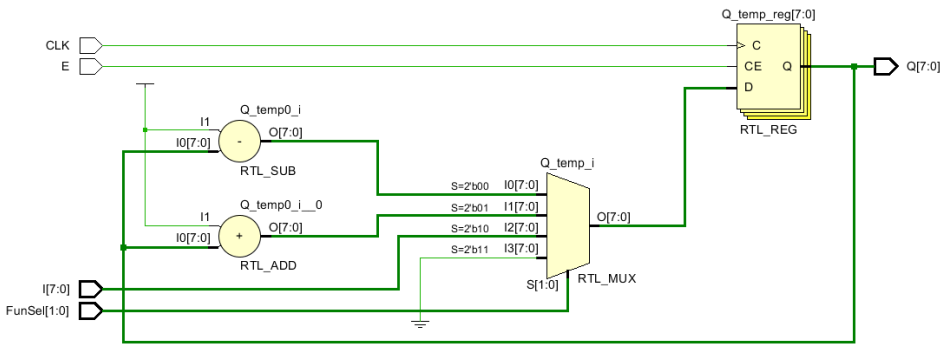


Figure 1: RTL Schematic of n-bit register

2.2 PART 2

2.2.1 PART 2.a

In this part there are 5 input registers and clock. FunSel is selecting function for output; RegSel is making enable R1, R2, R3 and R4 with its value. OutASel and OutBSel are selecting output register with their value. I is load part of this circuit.

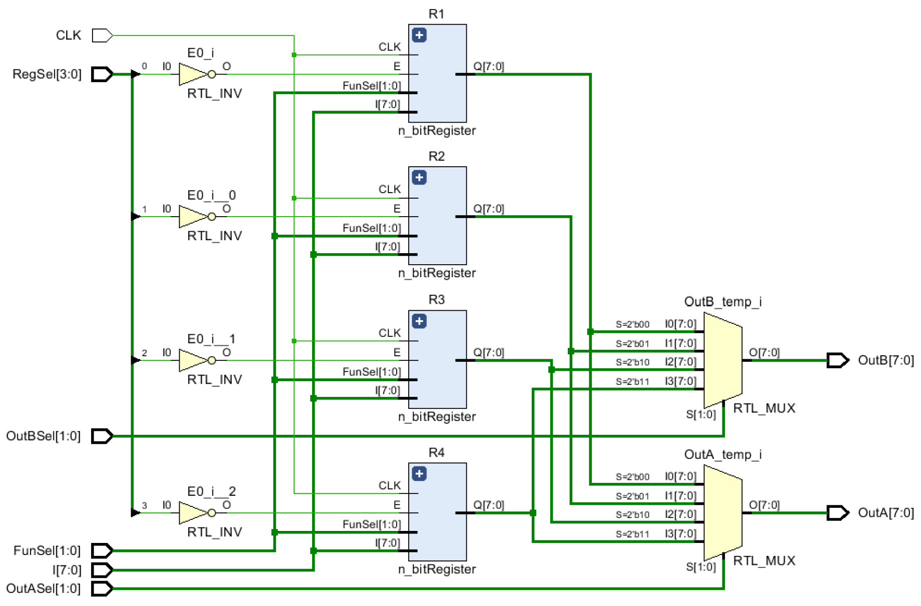


Figure 2: RTL Schematic of RegFile

2.2.2 PART 2.b

Adress Register File (ARF) system has three 8-bit registers which are program counter (PC), address register (AR), and stack pointer (SP). FunSel and RegSel works same as Part 2.a, OutCSEL and OutDSel are working like OutASel and OutBSel and I is load.

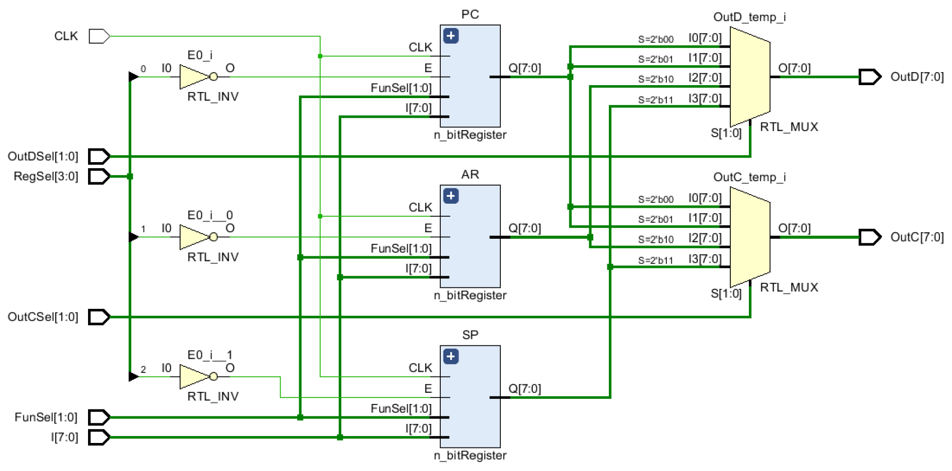


Figure 3: RTL Schematic of ARF

2.2.3 PART 2.c

We will design 16-bit IR register in this part and its' graphic symbol and characteristic table are given in our PDF file. Its' input register is 8-bit register and it should store 16-bit binary numbers. However out input is only 8-bit and we will make storage with

L/H signals; if signal is high we will store bits between 15-8, else if signal is lower we will store bits between 7-0.

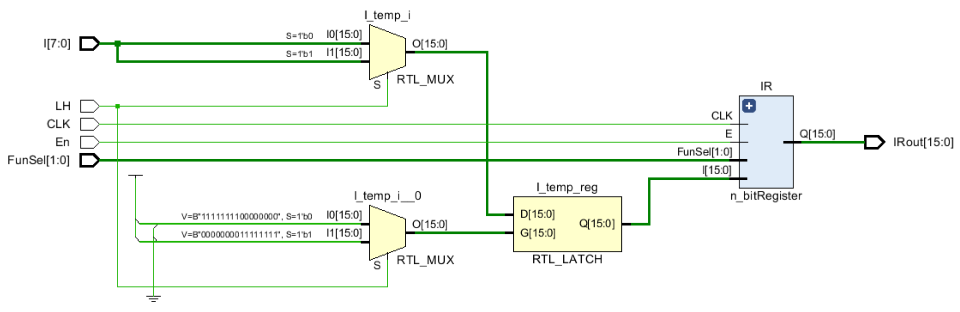


Figure 4: RTL Schematic of IR

2.3 PART 3

The ALU we designed has a combinational base and a 4-bit flag register which is sequential logic. Due to this hybrid structure delay in the flag registers is unavoidable. The ALU uses mostly standard Verilog operators except for the operations involving Carry and shifting.

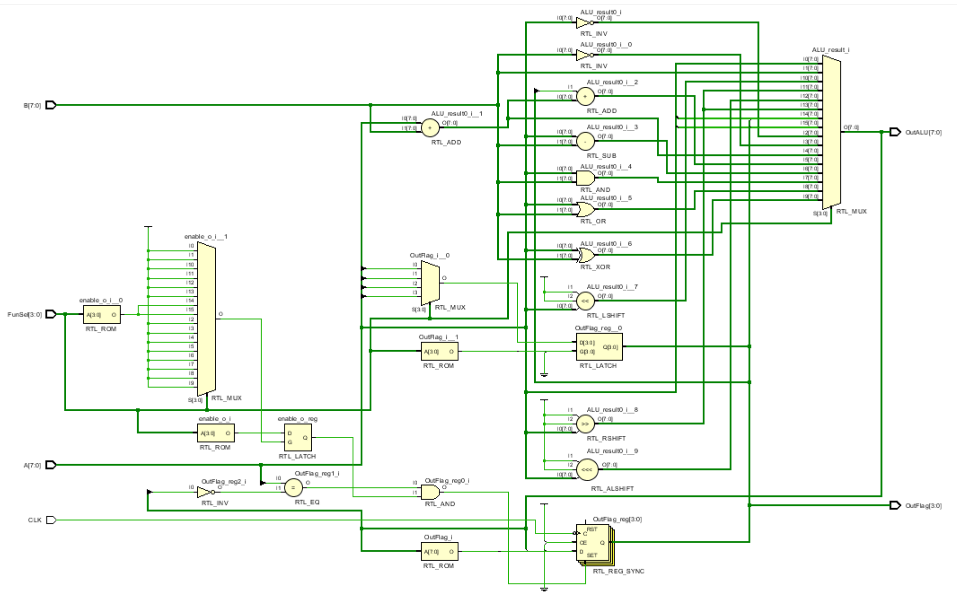


Figure 5: RTL Schematic of ALU

2.4 PART 4

The ALU system consists of the modules previously designed and multiplexers which were case or if statements in Verilog. The connection scheme was a bit hard to follow and explains the necessity of a common bus system even for a computer this basic.

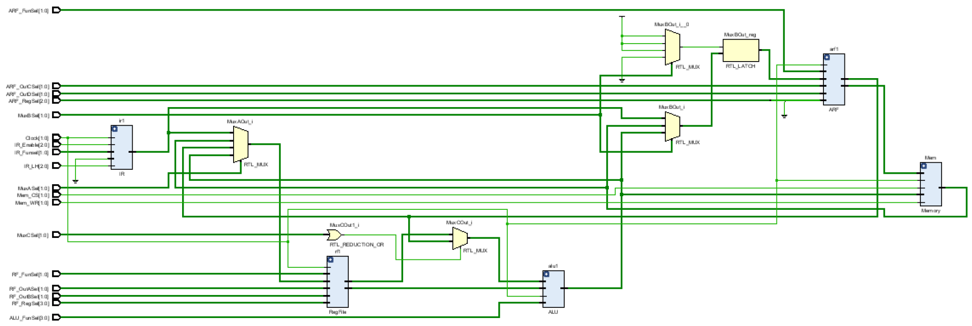


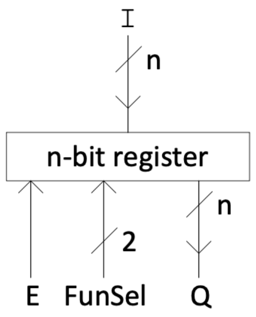
Figure 6: RTL Schematic of ALU system

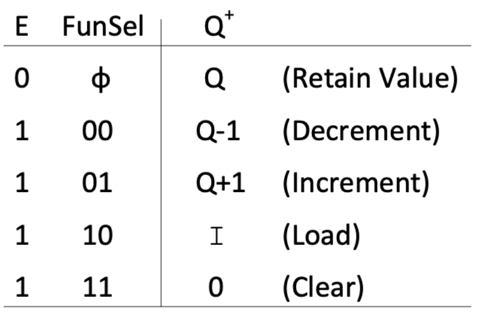
3 RESULTS [15 points]

End of this experiment we got different waveforms for each design that we made with Verilog Hardware Description Language. These waveforms are providing our function tables so our design codes and simulation codes are consistent.

Here are waveforms and function tables for each design:

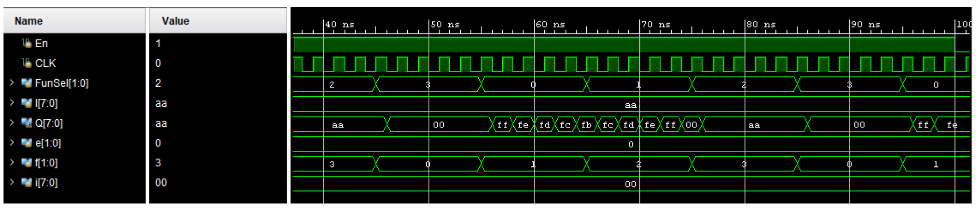
3.1 PART 1





E	FunSel	Q^+	
0	ϕ	Q	(Retain Value)
1	00	Q-1	(Decrement)
1	01	Q+1	(Increment)
1	10	I	(Load)
1	11	0	(Clear)

Figure 7: Graphic symbol of the registers (Left) and the characteristic table (Right)



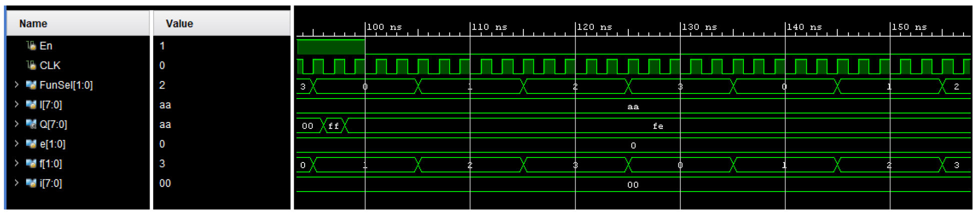


Figure 8: Waveform of n-bit register

3.2 PART 2

3.2.1 PART 2.a

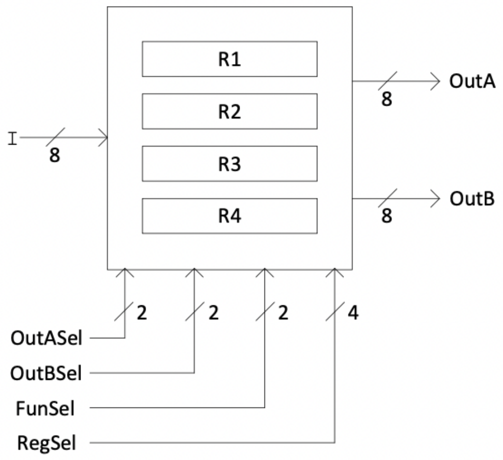


Figure 9: 8-bit general purpose registers, inputs, and outputs

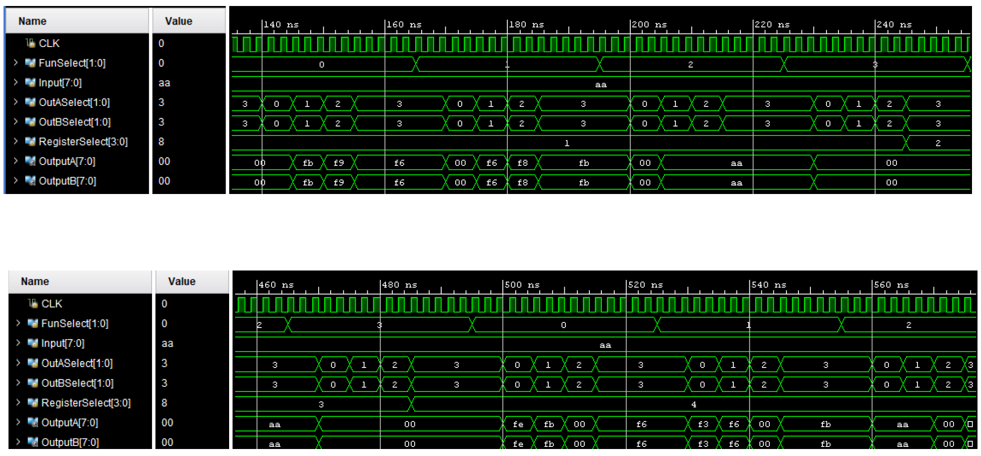


Figure 10: Waveform of RegFile

3.2.2 PART 2.b

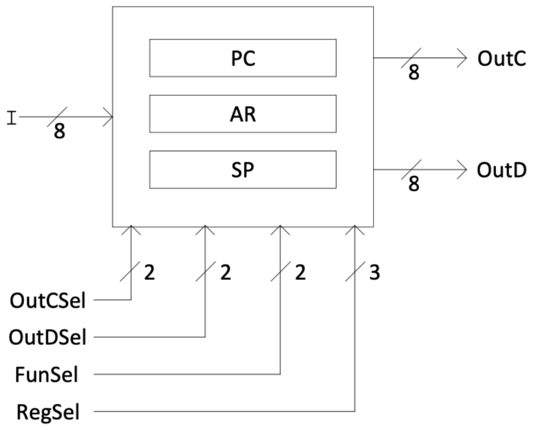


Figure 11: 8-bit address registers, inputs, and outputs

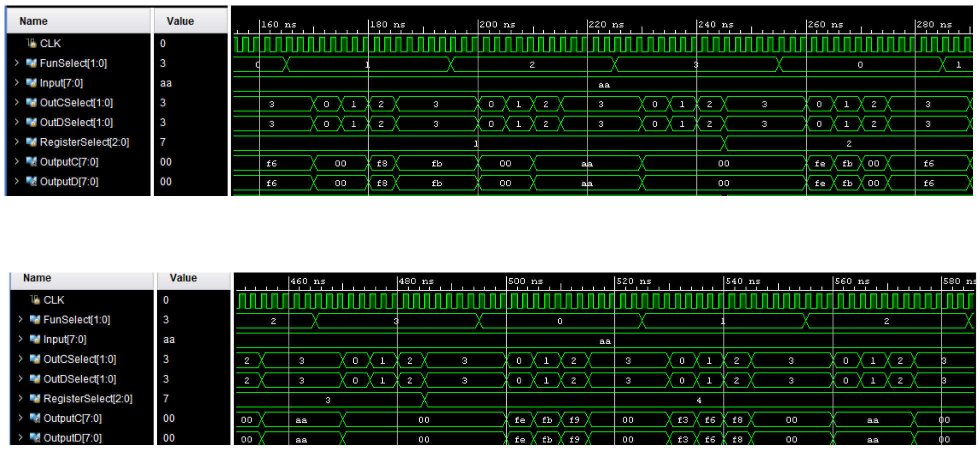
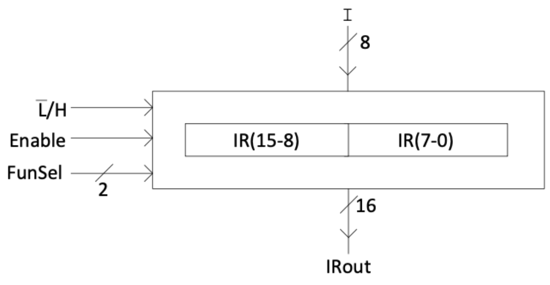
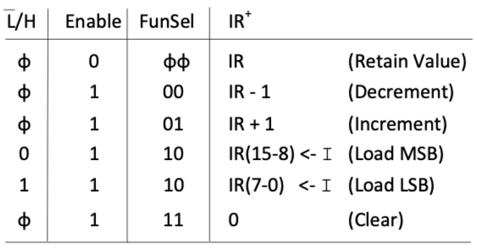


Figure 12: Waveform of ARF

3.2.3 PART 2.c





L̄/H	Enable	FunSel	IR^+	
ϕ	0	ϕϕ	IR	(Retain Value)
ϕ	1	00	IR - 1	(Decrement)
ϕ	1	01	IR + 1	(Increment)
0	1	10	IR(15-8) <- I	(Load MSB)
1	1	10	IR(7-0) <- I	(Load LSB)
ϕ	1	11	0	(Clear)

Figure 13: Graphic symbol of the IR register (Left) and its characteristic table (Right)

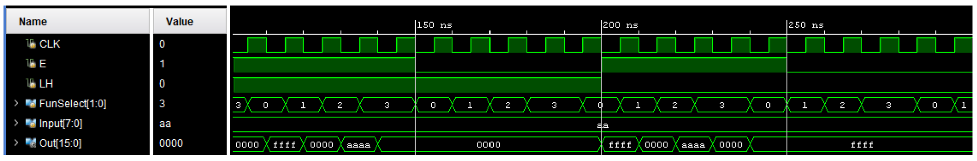
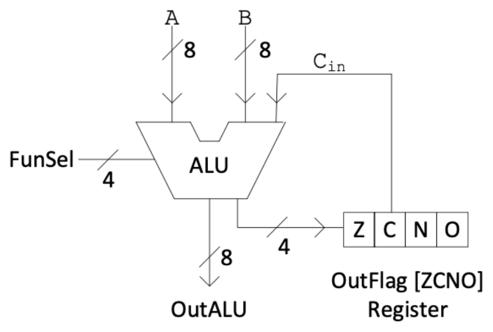
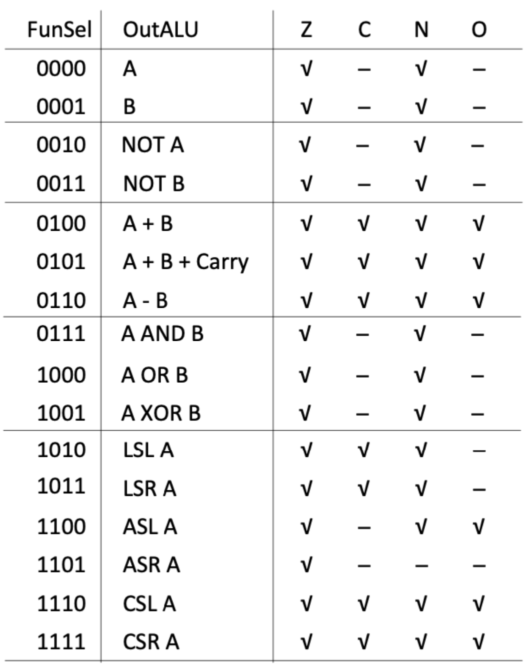


Figure 14: Waveform of IR

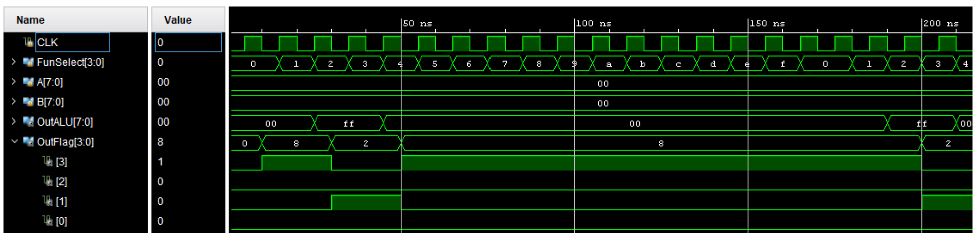
3.3 PART 3





FunSel	OutALU	Z	C	N	O
0000	A	√	–	√	–
0001	B	√	–	√	–
0010	NOT A	√	–	√	–
0011	NOT B	√	–	√	–
0100	A + B	√	√	√	√
0101	A + B + Carry	√	√	√	√
0110	A - B	√	√	√	√
0111	A AND B	√	–	√	–
1000	A OR B	√	–	√	–
1001	A XOR B	√	–	√	–
1010	LSL A	√	√	√	–
1011	LSR A	√	√	√	–
1100	ASL A	√	–	√	√
1101	ASR A	√	–	–	–
1110	CSL A	√	√	√	√
1111	CSR A	√	√	√	√

Figure 15: The ALU (Left) and its characteristic table (Right)



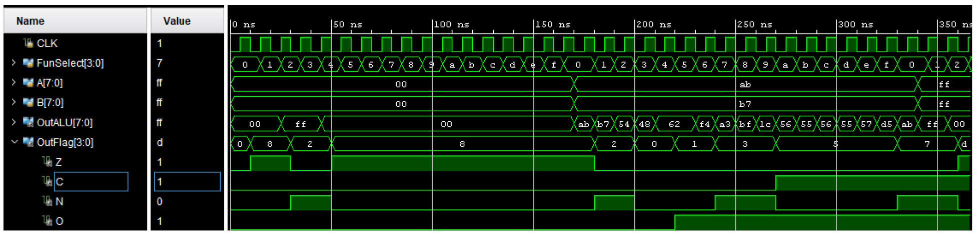


Figure 16: Waveform of ALU

3.4 PART 4

Result of the example automated system is as follows:

```
    Input Values:
Operation: 1
Register File: OutASel: 2, OutBSel: 3, FunSel: 3, Regsel:  0
ALU FunSel:  3
Addres Register File: OutCSel: 3, OutDSel: 1, FunSel: 3, Regsel: 0
Instruction Register: LH: 1, Enable: 1, FunSel: 3
Memory: WR: 1, CS: 1
MuxASel: 2, MuxBSel: 1, MuxCSel: 1

Ouput Values:
Register File: AOut:   x, BOut:   x
ALUOut:   x, ALUOutFlag:  0, ALUOutFlags: Z:0, C:0, N:0, O:0,
Address Register File: COut:   x, DOut (Address):   x
Memory Out:   z
Instruction Register: IROut:     x
MuxAOut:   x, MuxBOut:   x, MuxCOut:   x

Input Values:
Operation: 0
Register File: OutASel: 1, OutBSel: 2, FunSel: 1, Regsel:  2
ALU FunSel:  5
Addres Register File: OutCSel: 2, OutDSel: 2, FunSel: 0, Regsel: 2
Instruction Register: LH: 0, Enable: 0, FunSel: 1
Memory: WR: 0, CS: 0
MuxASel: 1, MuxBSel: 0, MuxCSel: 0

Ouput Values:
Register File: AOut:   0, BOut:   0
ALUOut:   0, ALUOutFlag:  0, ALUOutFlags: Z:0, C:0, N:0, O:0,
Address Register File: COut:   0, DOut (Address):   0
Memory Out:   0
Instruction Register: IROut:     0
MuxAOut:   z, MuxBOut:   x, MuxCOut:   0

Input Values:
```

```
Operation: 0
Register File: OutASel: 1, OutBSel: 2, FunSel: 1, Regsel:  2
ALU FunSel:  5
Addres Register File: OutCSel: 2, OutDSel: 2, FunSel: 1, Regsel: 2
Instruction Register: LH: 0, Enable: 0, FunSel: 1
Memory: WR: 0, CS: 0
MuxASel: 1, MuxBSel: 0, MuxCSel: 0

Ouput Values:
Register File: AOut:   0, BOut:   0
ALUOut:   0, ALUOutFlag:  8, ALUOutFlags: Z:1, C:0, N:0, O:0,
Address Register File: COut:   0, DOut (Address):   0
Memory Out:   0
Instruction Register: IROut:     0
MuxAOut:   z, MuxBOut:   x, MuxCOut:   0

         3 tests completed.
```

4 DISCUSSION [25 points]

We learned that sets of registers can have different purposes based on their structure and interfaces. As much as their connections matter, in a system with not a prominent bus multiplexers were used to effectively distribute the traffic. ALU is the main inducer of computing in the system. The process of working on the project was as follows. Firstly we tried to understand given PDF file to write these code for our registers. We discussed what should we do while coding and implementing these structures. After understanding of given PDF file, we begun to code our modules. While we were coding we faced with some errors and warnings, we tried to get over these problems. When we finished our modelling of registers we started to write simulation code. When our coding issues finished we started to write our report.

5 CONCLUSION [10 points]

We understand how registers and modules implementing in Verilog Hardware Description Language with our teammates. We understand how we can design modules in Verilog Hardware Description Language with team. Also we consolidate our knowledge with design what we learned.