ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION PROJECT 1 REPORT

PROJECT NO	:	1
PROJECT DATE	:	10.04.2023
GROUP NO	:	G24

GROUP MEMBERS:

150200029	:	HAVVA EDA KÖRPE
150200066	:	YUSUF EMİR SEZGİN
150200033	:	ÖZKAN GEZMİŞ

SPRING 2023

Contents

1 INTRODUCTION

Data operations that are done with registers are called microoperations. To obtain the results of these operations Arithmetic Logic Units (ALU) are used in the CPU. In this project, we have learned the working principle of this system. Then, we used Verilog to create an ALU system and examined its process. After the implementation, using test files, the simulation has been analyzed.

2 MATERIALS AND METHODS

We used our Digital Design knowledge and slides that were provided by lecturers. We have provided the essential screenshots and explanations in this section.

2.1 PART 1

In this section, the n-bit register has been designed to use for different purposes. Outputs of the register are controlled with FunSel and Enable (E) inputs.

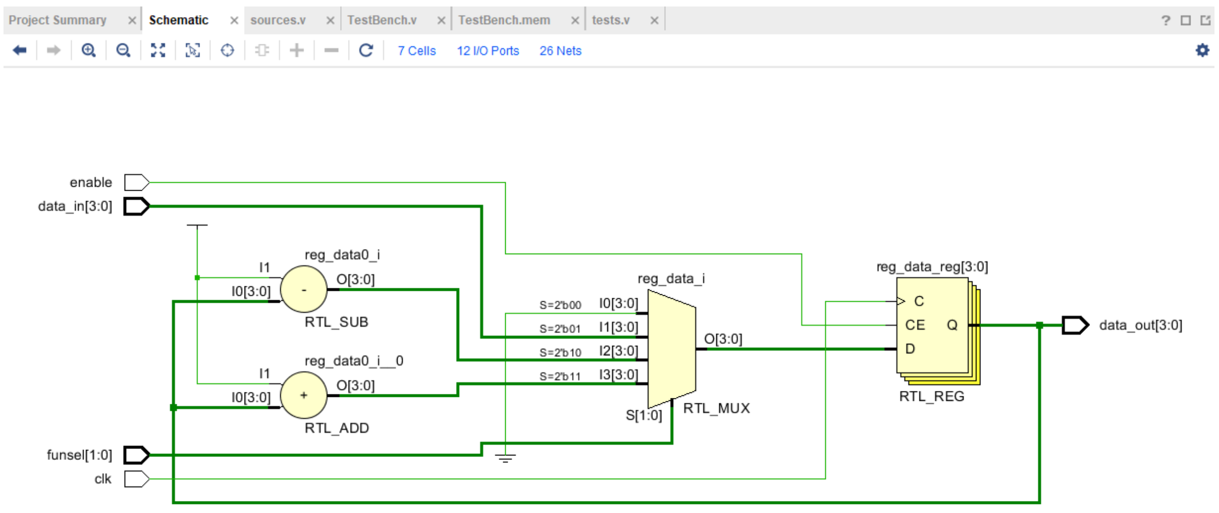


Figure 1: N-Bit Register RTL

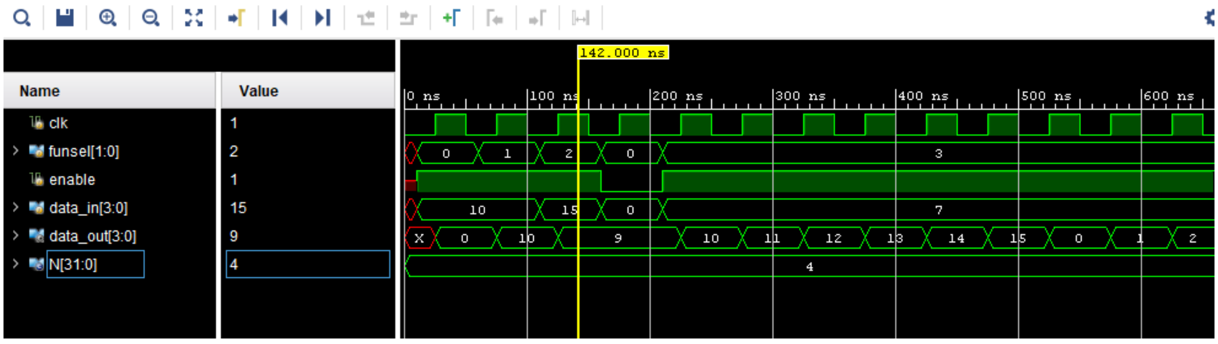


Figure 2: N-Bit Register Simulation

2.2 PART 2

2.2.1 PART 2.A

Instruction register (IR) using additional low/high signal that indicates the Most Significant and Least Significant bits have been created.

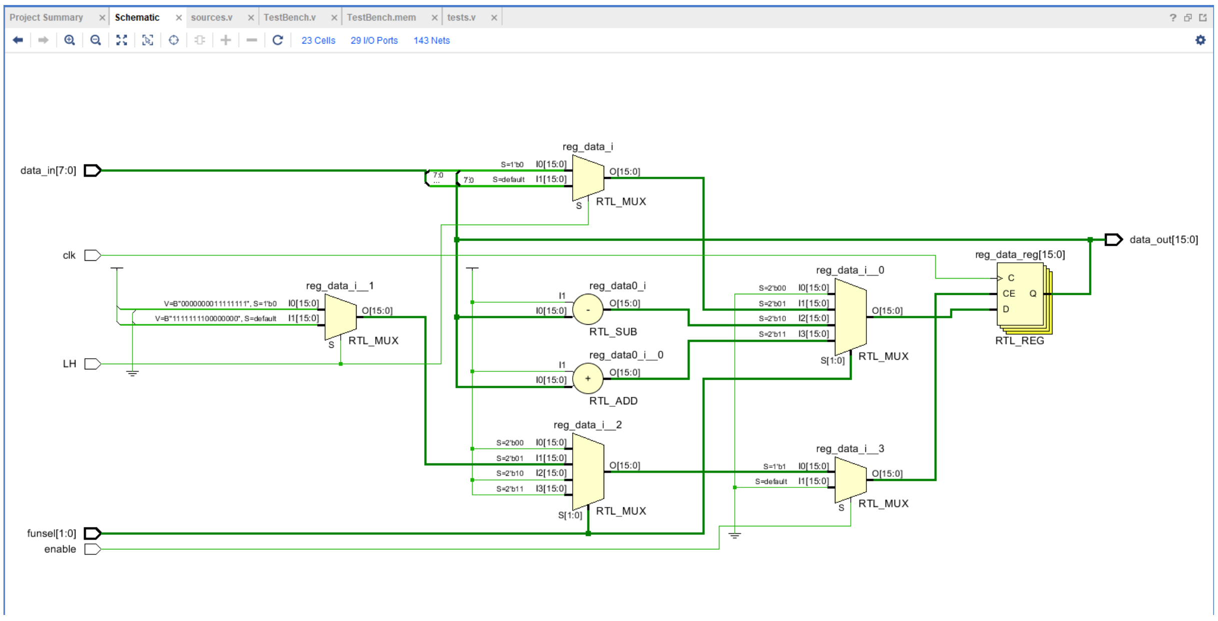


Figure 3: 16-Bit Register RTL

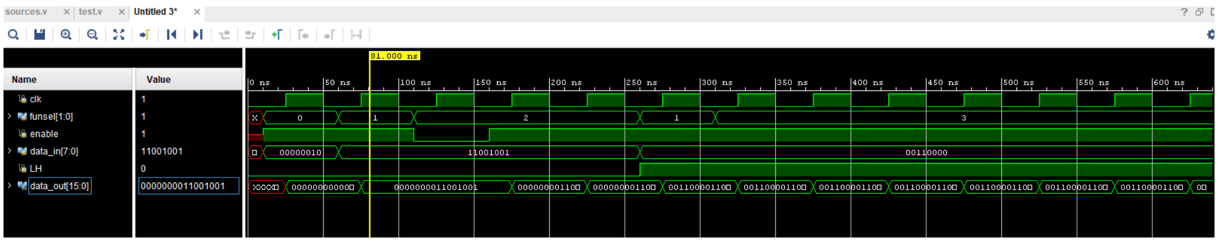
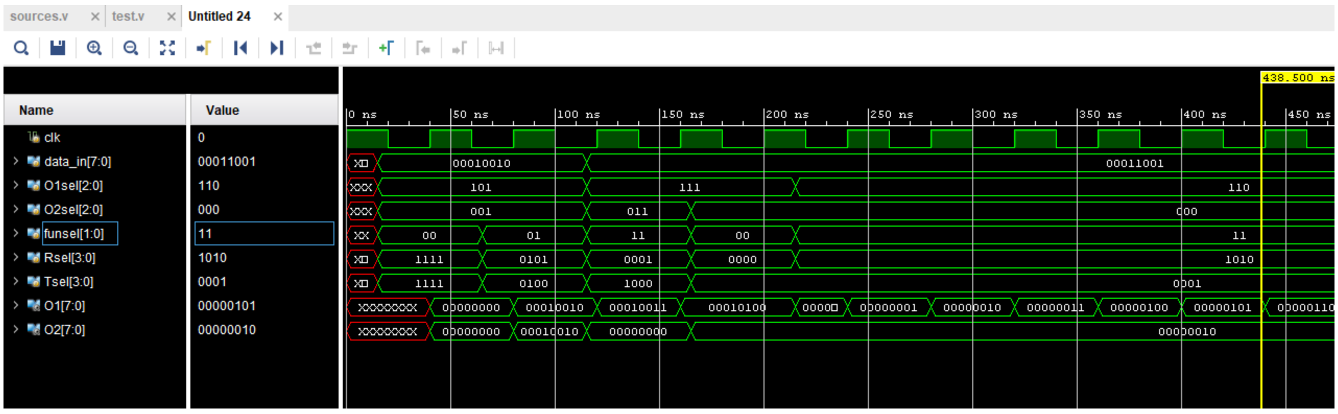


Figure 4: 16-Bit Register Simulation

2.2.2 PART 2.B

Using general purpose and temporary registers, we built a system that takes inputs such as O1Sel, O2Sel, FunSel, RSel, and TSel; gives outputs as O1 and O2 with n-bit registers that we implemented in the first part.



Name	Value
clk	0
data_in[7:0]	00011001
O1sel[2:0]	110
O2sel[2:0]	000
funsel[1:0]	11
Rsel[3:0]	1010
Tsel[3:0]	0001
O1[7:0]	00000101
O2[7:0]	00000010

Figure 5: Register File Simulation

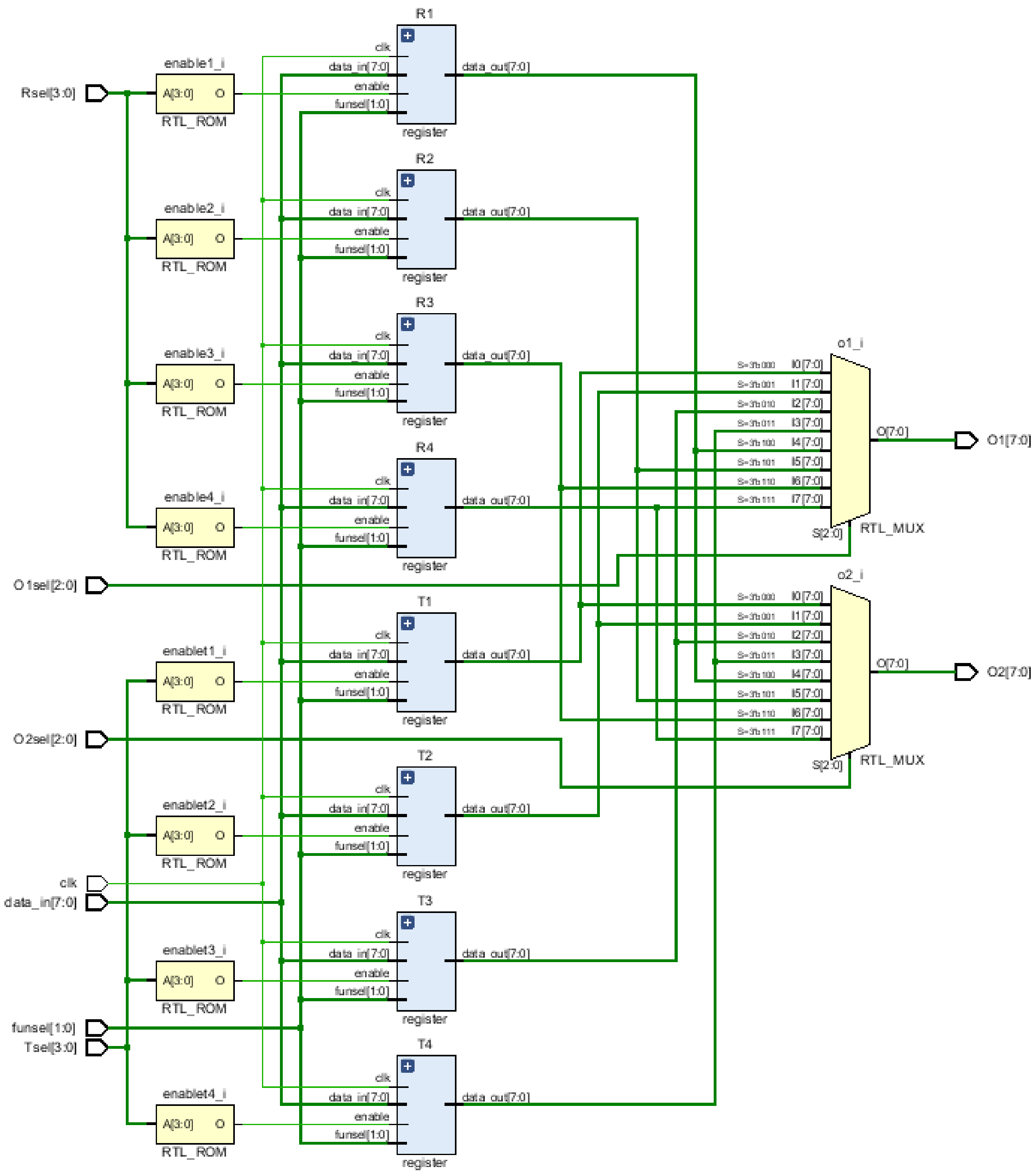


Figure 6: Register File RTL

2.2.3 PART 2.C

We have designed the address register file in this section which includes the program counter (PC), address register (AR), stack pointer (SP), and the past program counter (PCPast). This system takes inputs as OutASel, OutBSel, FunSel, and RSel; gives outputs as OutA and OutB.

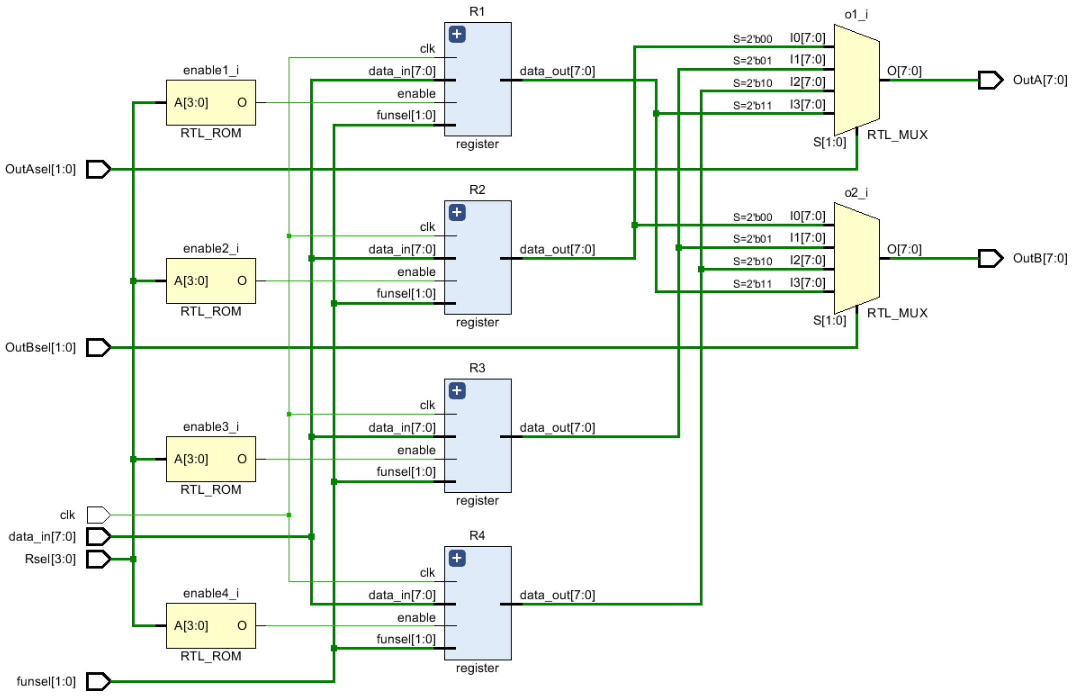
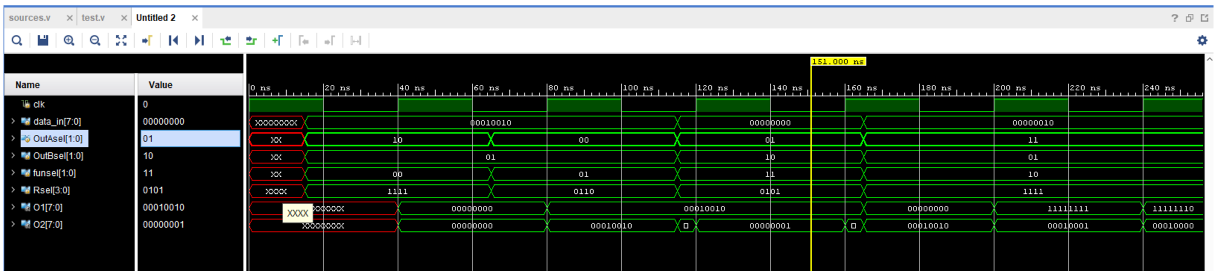


Figure 7: Address Register File RTL



Name	Value
clk	0
data_in[7:0]	00000000
OutAsel[1:0]	01
OutBsel[1:0]	10
funsel[1:0]	11
Rsel[3:0]	0101
O1[7:0]	00010010
O2[7:0]	00000001

Figure 8: Adress Register File Simulation

2.3 PART 3

We have implemented an ALU using FunSel as input which leads to complement, addition, subtraction, AND, OR, NAND, XOR, compare, and shifting operations. This module gives output as OutALU and there is a register that specifies Z (zero), C (carry), N (negative), and O (overflow) flags.

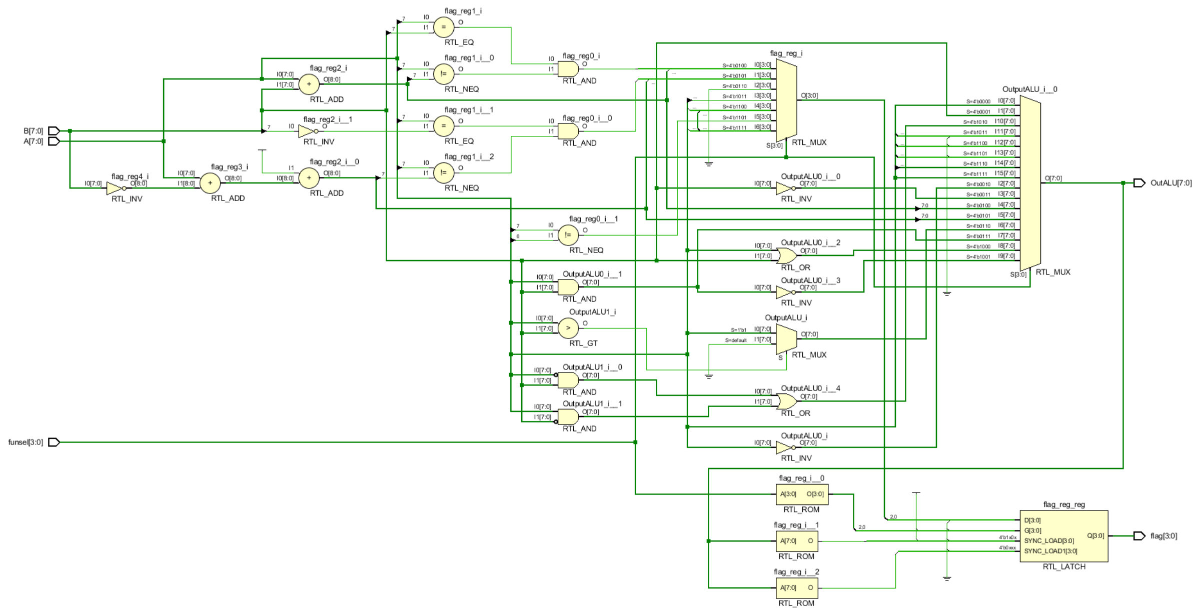


Figure 9: ALU RTL

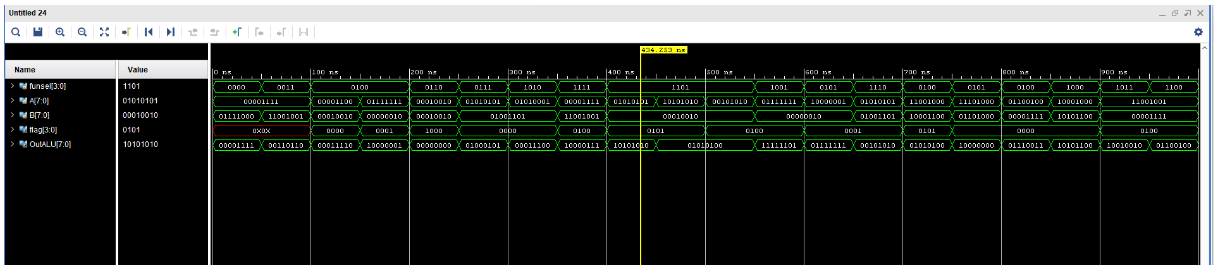


Figure 10: ALU Simulation

2.4 PART 4

In this part, previous parts have been integrated with three multiplexers to build a whole ALU system that uses a single clock signal. RAM module and memory data have been added to the project.

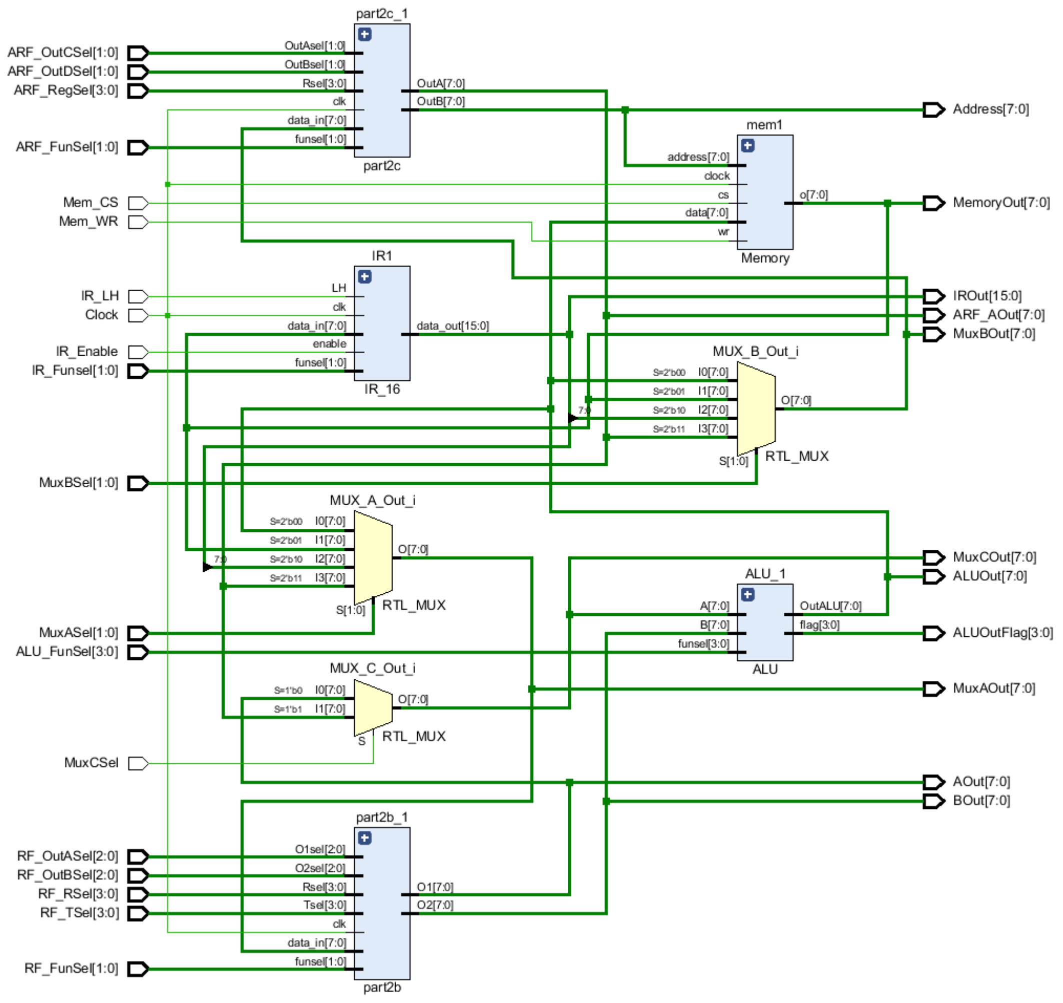
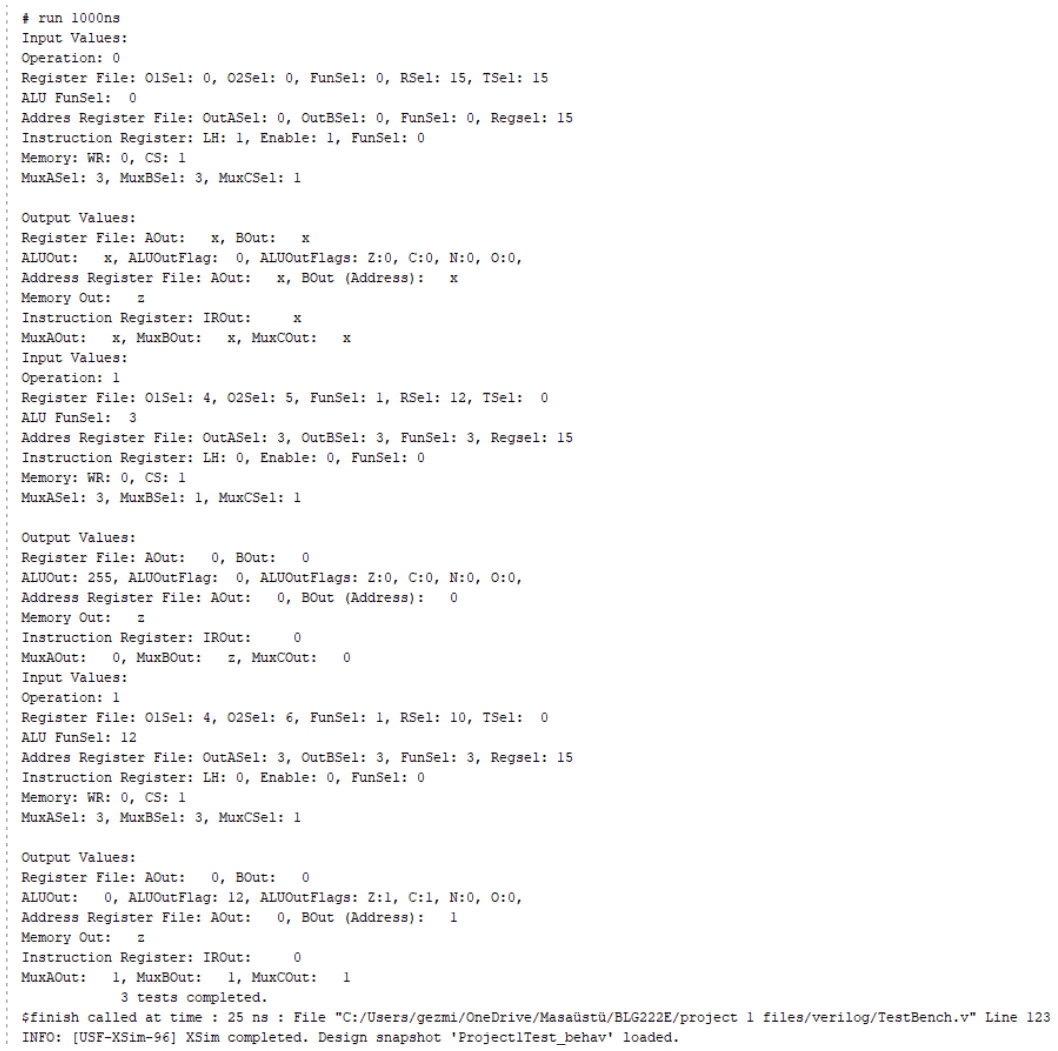


Figure 11: ALU System RTL



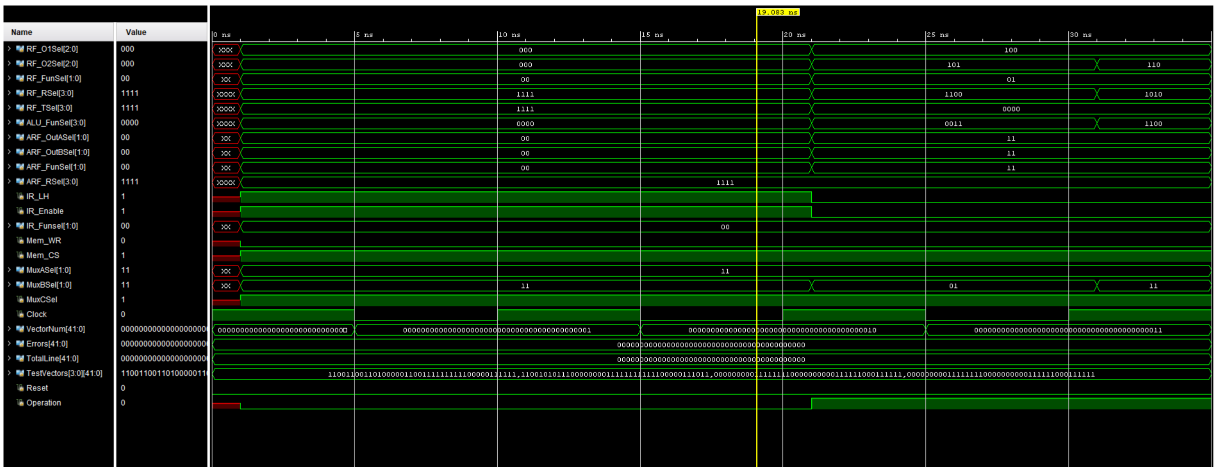
```
# run 1000ns
Input Values:
Operation: 0
Register File: O1Sel: 0, O2Sel: 0, FunSel: 0, RSel: 15, TSel: 15
ALU FunSel:  0
Addres Register File: OutASel: 0, OutBSel: 0, FunSel: 0, Regsel: 15
Instruction Register: LH: 1, Enable: 1, FunSel: 0
Memory: WR: 0, CS: 1
MuxASel: 3, MuxBSel: 3, MuxCSel: 1

Output Values:
Register File: AOut:   x, BOut:   x
ALUOut:   x, ALUOutFlag:  0, ALUOutFlags: Z:0, C:0, N:0, O:0,
Address Register File: AOut:   x, BOut (Address):   x
Memory Out:   z
Instruction Register: IROut:     x
MuxAOut:   x, MuxBOut:   x, MuxCOut:   x
Input Values:
Operation: 1
Register File: O1Sel: 4, O2Sel: 5, FunSel: 1, RSel: 12, TSel:  0
ALU FunSel:  3
Addres Register File: OutASel: 3, OutBSel: 3, FunSel: 3, Regsel: 15
Instruction Register: LH: 0, Enable: 0, FunSel: 0
Memory: WR: 0, CS: 1
MuxASel: 3, MuxBSel: 1, MuxCSel: 1

Output Values:
Register File: AOut:   0, BOut:   0
ALUOut: 255, ALUOutFlag:  0, ALUOutFlags: Z:0, C:0, N:0, O:0,
Address Register File: AOut:   0, BOut (Address):   0
Memory Out:   z
Instruction Register: IROut:     0
MuxAOut:   0, MuxBOut:   z, MuxCOut:   0
Input Values:
Operation: 1
Register File: O1Sel: 4, O2Sel: 6, FunSel: 1, RSel: 10, TSel:  0
ALU FunSel: 12
Addres Register File: OutASel: 3, OutBSel: 3, FunSel: 3, Regsel: 15
Instruction Register: LH: 0, Enable: 0, FunSel: 0
Memory: WR: 0, CS: 1
MuxASel: 3, MuxBSel: 3, MuxCSel: 1

Output Values:
Register File: AOut:   0, BOut:   0
ALUOut:   0, ALUOutFlag: 12, ALUOutFlags: Z:1, C:1, N:0, O:0,
Address Register File: AOut:   0, BOut (Address):   1
Memory Out:   z
Instruction Register: IROut:     0
MuxAOut:   1, MuxBOut:   1, MuxCOut:   1
            3 tests completed.
$finish called at time : 25 ns : File "C:/Users/gezmi/OneDrive/Masaüstü/BLG222E/project 1 files/verilog/TestBench.v" Line 123
INFO: [USF-XSim-96] XSim completed. Design snapshot 'Project1Test_behav' loaded.
```

Figure 12: ALU System Terminal Messages



Name	Value
RF_O1Sel[2:0]	000
RF_O2Sel[2:0]	000
RF_FunSel[1:0]	00
RF_RSel[3:0]	1111
RF_TSel[3:0]	1111
ALU_FunSel[3:0]	0000
ARF_OutASel[1:0]	00
ARF_OutBSel[1:0]	00
ARF_FunSel[1:0]	00
ARF_RSel[3:0]	1111
IR_LH	1
IR_Enable	1
IR_Funsel[1:0]	00
Mem_WR	0
Mem_CS	1
MuxASel[1:0]	11
MuxBSel[1:0]	11
MuxCSel	1
Clock	0
VectorNum[41:0]	0000000000000000000
Errors[41:0]	0000000000000000000
TotalLine[41:0]	0000000000000000000
TestVectors[3:0][41:0]	1100110011010000011
Reset	0
Operation	0

Figure 13: ALU System Simulation

3 RESULTS

We have experienced that by using registers, we can obtain the ALU system, thus; the working principle of the logic and arithmetic operations performed by digital circuit components has been understood clearly.

4 DISCUSSION

Firstly, we analyzed the project as a group and decided what our steps needs to be to create our ALU project. We divided our work into parts. However, after a while, we figured out that working on parts separately will be more stressful and difficult due to preparing the parts and include to each other, etc. After that, we worked together on the parts. We created the logic designs and write Verilog codes cooperatively for all the parts. We had difficulties from time to time while we trying to understand registers and the systems, but eventually, we analyzed correctly. We finished our project early before the due time. However, we waited for the test-bench typos to be resolved.

5 CONCLUSION

In this project, firstly; we have built an n-bit register and used it to create an instruction register (IR), general purpose, and temporary registers. Then, we used it for implementing an address register file including different types of registers such as program counter (PC), address register (AR), stack pointer (SP), and the past program counter (PCPast). Finally, using these parts, Arithmetic Logic Unit has been implemented and after the organization of the system, the performance of the whole system has been observed with the same clock signal.