ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E
COMPUTER ORGANIZATION
PROJECT 1 REPORT

CRN	:	21336
LECTURER	:	Prof. Dr. Mustafa Ersel KAMAŞAK

GROUP MEMBERS:

150210055	:	Taha TEMİZ
150210060	:	Mehmet Mert ERGİN

SPRING 2024

Contents

1 INTRODUCTION

In this project, we designed a 16-bit register, an instruction register, and an ALU. We designed a register file module and an address register file module by using the register module that we created. By using the instruction register (IR), address register file, register file, ALU, MUXs, and given memory block, we implemented an arithmetic logic unit system. After our designs were finished, we simulated each part according to the various input combinations given. We implemented all the modules by using the Verilog hardware description language.

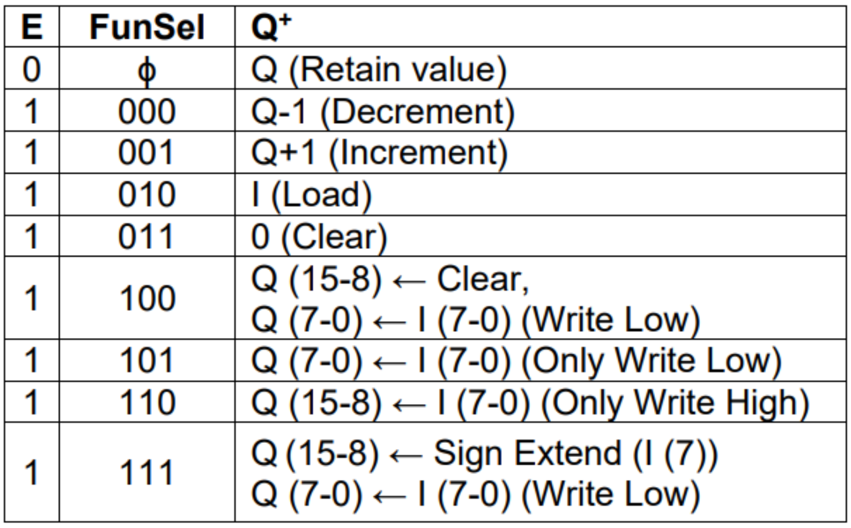
1.1 TASK DISTRIBUTION

Since we were roommates, we had the opportunity to complete all the parts of the project together. We did not distribute a specific task. We were together at all stages.

2 PROJECT PARTS

2.1 PART-1

We designed a 16-bit register. This register has 8 functionalities that are controlled by 3-bit control signals (FunSel), an enable input (E), and a clock signal. When the clock signal goes from low to high, the register applies the selected function according to the register's characteristic table, which can be seen in Figure 1.



E	FunSel	Q^+
0	ϕ	Q (Retain value)
1	000	Q-1 (Decrement)
1	001	Q+1 (Increment)
1	010	I (Load)
1	011	0 (Clear)
1	100	Q (15-8) ← Clear, Q (7-0) ← I (7-0) (Write Low)
1	101	Q (7-0) ← I (7-0) (Only Write Low)
1	110	Q (15-8) ← I (7-0) (Only Write High)
1	111	Q (15-8) ← Sign Extend (I (7)) Q (7-0) ← I (7-0) (Write Low)

Figure 1: Register's Characteristic Table

The elaborated design of the register can be seen in Figure 2.

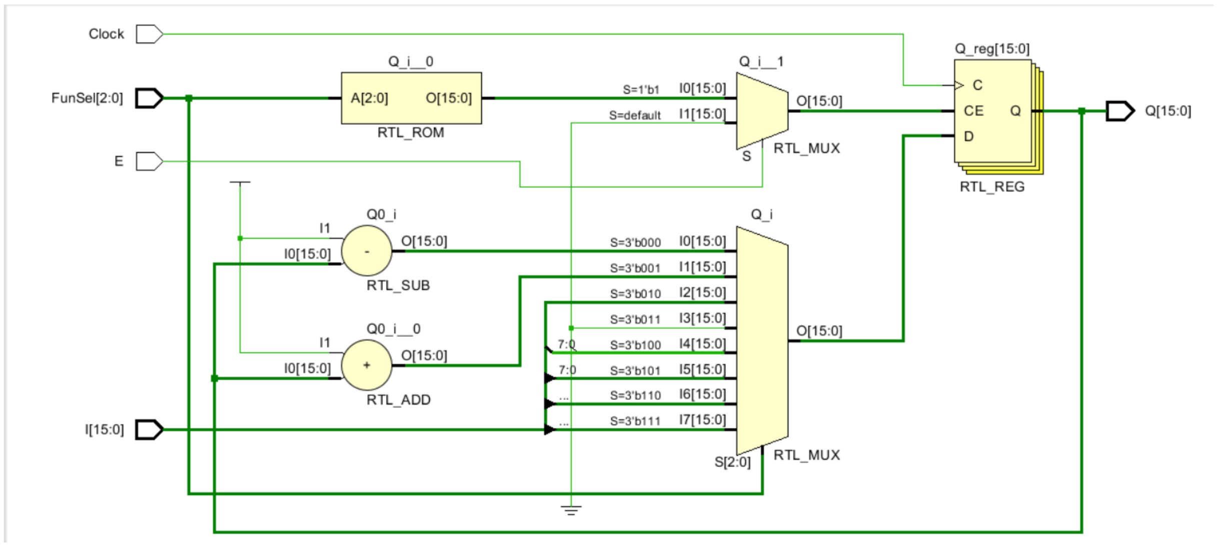


Figure 2: Register Schematic

2.2 PART-2

2.2.1 PART-2a

We designed a 16-bit instruction register (IR). It has an 8-bit input. We connected write, L'H, and the clock signal to this register. When the clock signal goes from low to high, if the write signal is 1, the register loads input to the left (bits 15–8) or right (bits 7–0) side of its value according to the L'H signal (left side for H, right side for L). If the write signal is 0, the register retains its value. The elaborated design of the IR can be seen in Figure 3.

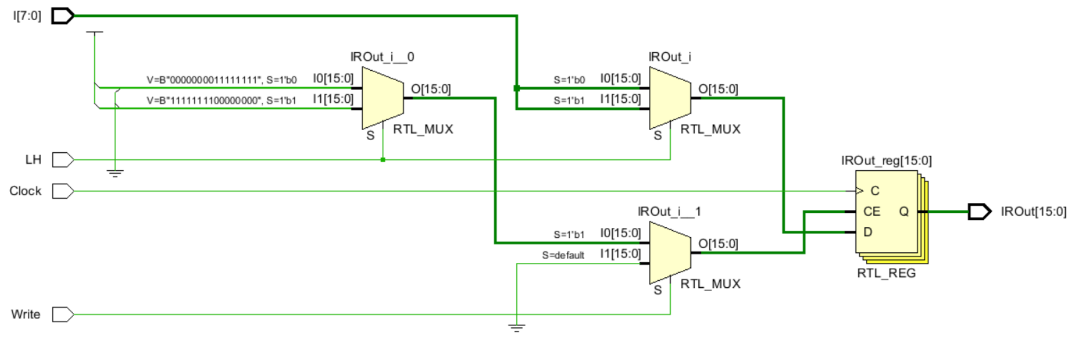


Figure 3: Instruction Register Schematic

2.2.2 PART-2b

We designed a register file containing four 16-bit general purpose registers and four 16-bit scratch registers. The given functions of the registers were the same as the register module that we designed in Part 1. So, we used the same register module. This system has 7 inputs. These inputs are I (16-bit), OutASel (3-bit), OutBSel (3-bit), FunSel (3-bit), RegSel (4-bit), ScrSel (4-bit), and Clock. It has two outputs: OutA (16-bit) and OutB (16-bit). With OutASel, we can choose which register's output will be OutA. With OutBSel, we choose which register's output will be OutB. With RegSel, we select which of the general purpose registers are enabled or not. With ScrSel, we select which of the scratch registers are enabled or not. When the clock signal goes from low to high, according to FunSel, enabled registers apply the selected function. The elaborated design of the register file can be seen in Figure 4.

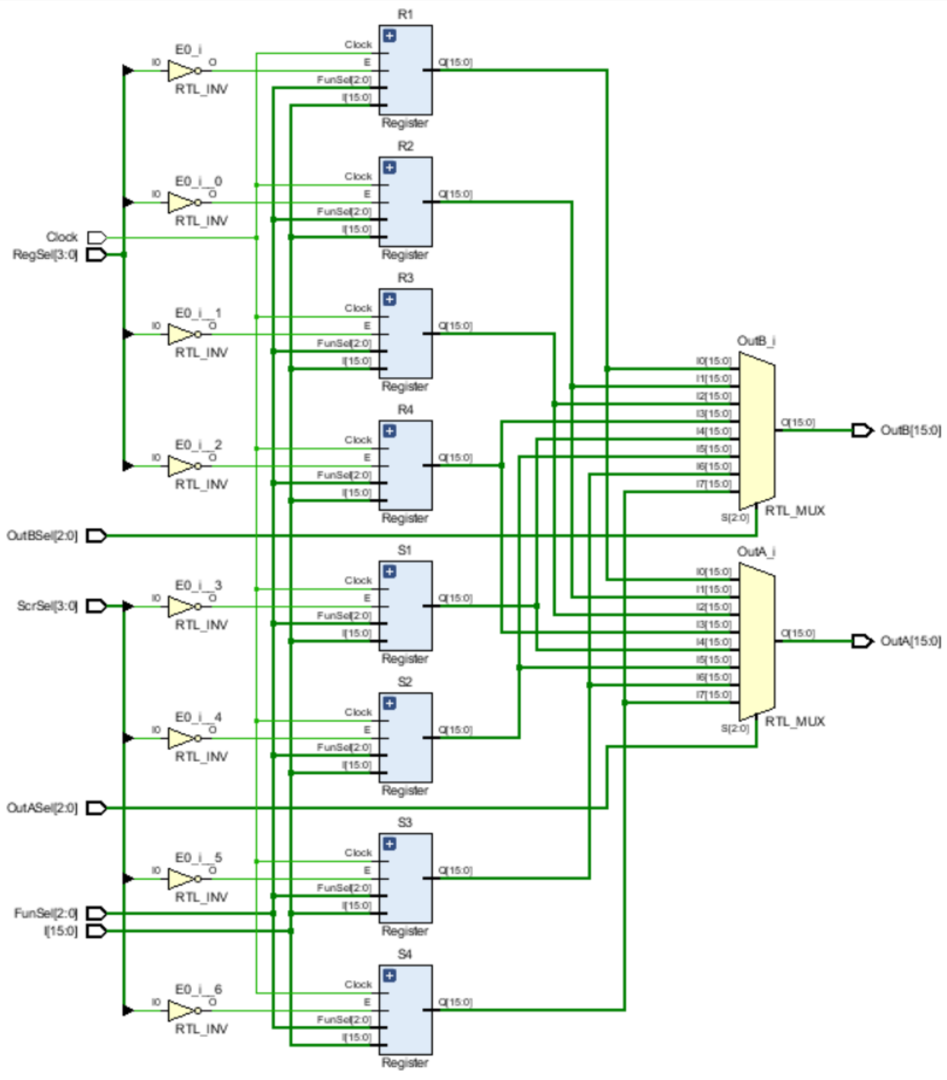


Figure 4: Register File Schematic

2.2.3 PART-2c

We designed an address register file containing three 16-bit address registers: program counter (PC), address register (AR), and stack pointer (SP). We used the register module that we created in Part 1. This system has six inputs, which are I (16-bit), OutCSel (2-bit), OutDSel (2-bit), FunSel (3-bit), RegSel (3-bit), and Clock. This system has two outputs: OutC (16-bit) and OutD (16-bit). FunSel and Regsel work as in Part 2b. OutC and OutD are the output lines that are fed by OutCSel and OutDSel, as seen in Figure 5. When the clock signal goes from low to high, enabled registers apply the selected function by FunSel.

OutCSel	OutC
00	PC
01	PC
10	AR
11	SP

OutDSel	OutD
00	PC
01	PC
10	AR
11	SP

Figure 5: OutC and OutD

The elaborated design of the address register file can be seen in Figure 6

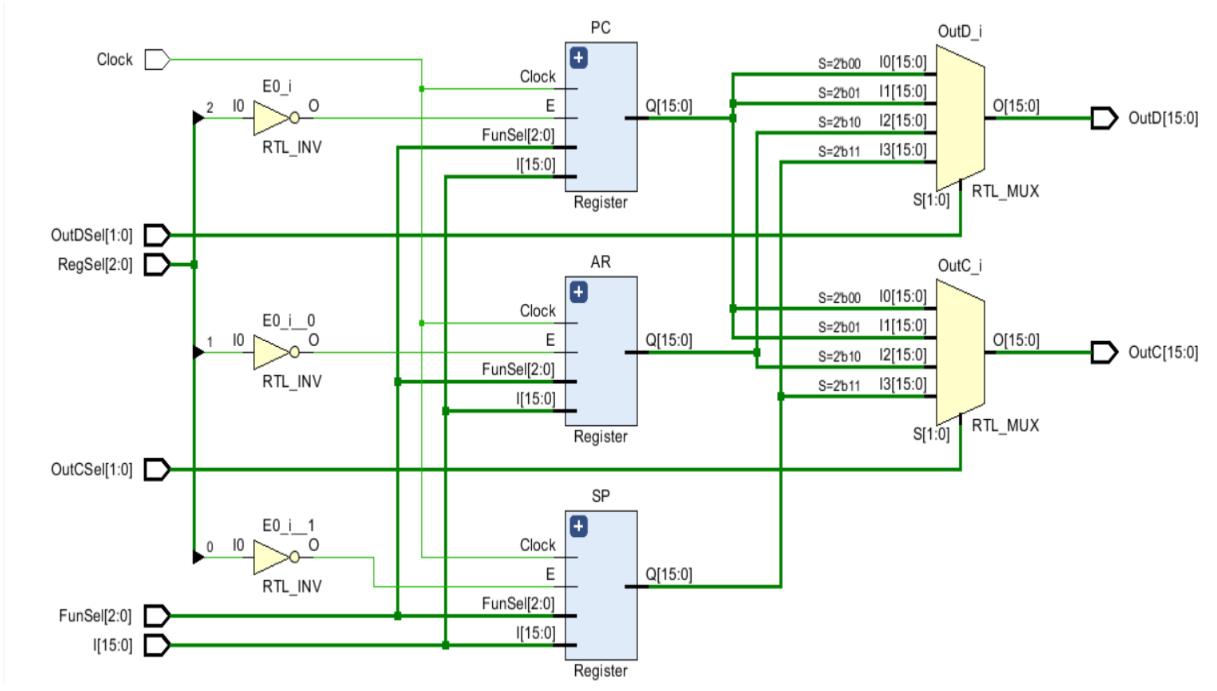


Figure 6: ARF Schematic

2.3 PART-3

We designed an Arithmetic Logic Unit (ALU) with two 16-bit inputs, producing a 16-bit output along with a 4-bit output indicating flags for zero (Z), negative (N), carry (C), and overflow (O) conditions. Flags are stored in a register named FlagsOut. The Z flag is the MSB of the register, and the O flag is the LSB of the register. The FlagsOut has a WF input and a clock input. According to FunSel input, ALU applies the selected function. 8-bit operations can also be applied with the ALU. When it needs carry, it gets carry input from FlagsOut. ALU always applies functions, but FlagsOut only operates when the clock signal goes from low to high. Additionally, if the WF signal of FlagsOut is 0, it retains its own value. The elaborated design of the ALU can be seen in Figure 7.

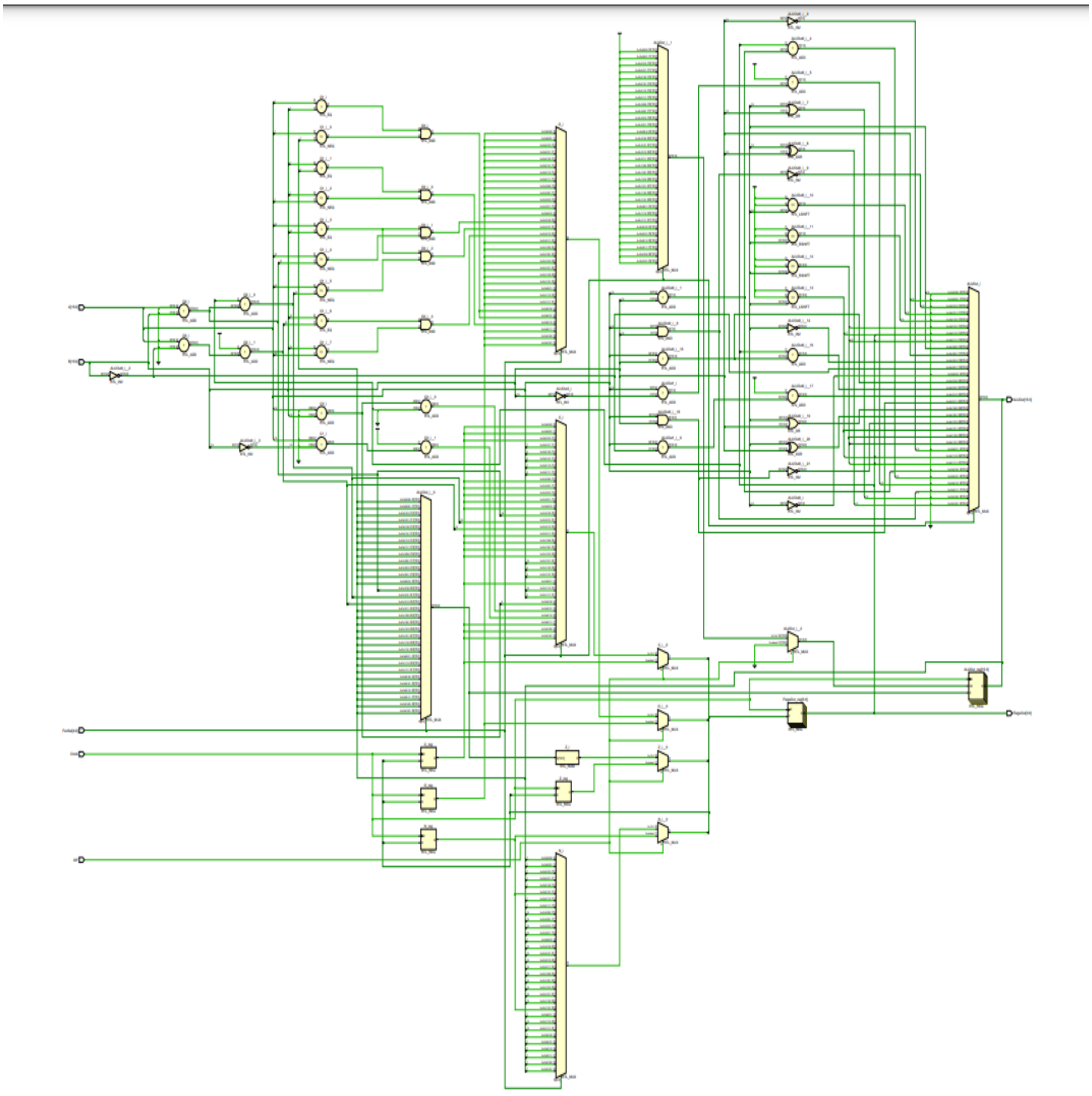


Figure 7: ALU Schematic

2.4 PART-4

In Part 4, we designed the Arithmetic Logic Unit System, which is a fundamental component of a computer's central processing unit (CPU) responsible for performing arithmetic and logic operations on binary numbers. This design includes a register file, an address register file, an instruction register, an ALU, two 4:1 MUXs, one 1:1 MUX, and a memory block that is given to us in homework files. We used the modules that we created in the previous parts. We also design the MUXs. MUX A is one of the 4:1 MUXs. Inputs of MUXA are: ALUOut, OutC of the ARF, memory output,and IROut (7:0). According to MUXASel, MUX A sends one of them to the register file. The register file works as in Part 2b. OutA and OutB of this register are the inputs of the ALU. ALU applies the selected function according to FunSel, and the ALU sends ALUOut to both MUX A and MUX C. According to MUXCSel, MUX C sends ALUOut (15:8) or ALUOut (7:0) to the memory block. MUX B is the other 4:1 MUX. Inputs of MUX B are: ALUOut, OutC of ARF, memory output,and IROut (7:0). According to MUXBSel, MUX B sends one of them to ARF. The address input of the memory block is fed by OutD of the ARF. The whole system uses the same single clock signal. The elaborated design of the ALU system can be seen in Figure 8.

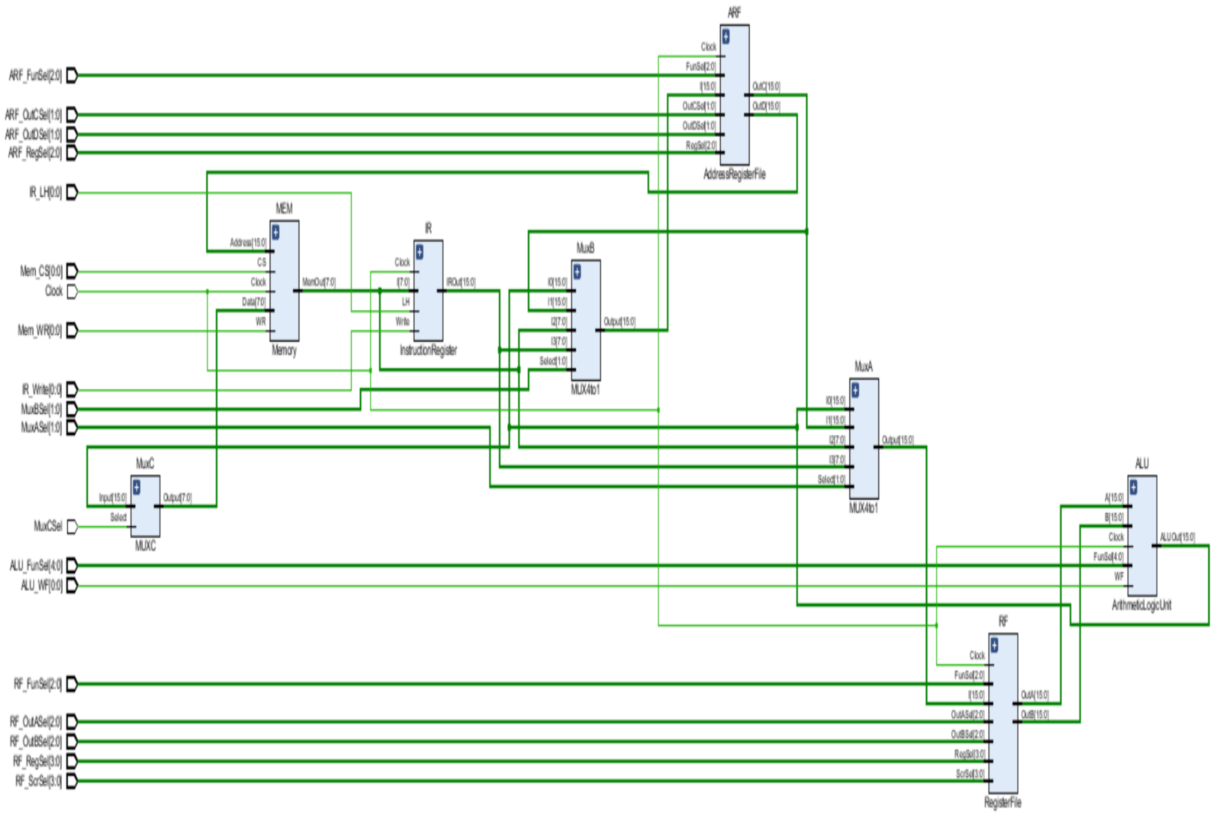


Figure 8: ALU System Schematic

3 RESULTS

3.1 SIMULATION OF PART-1

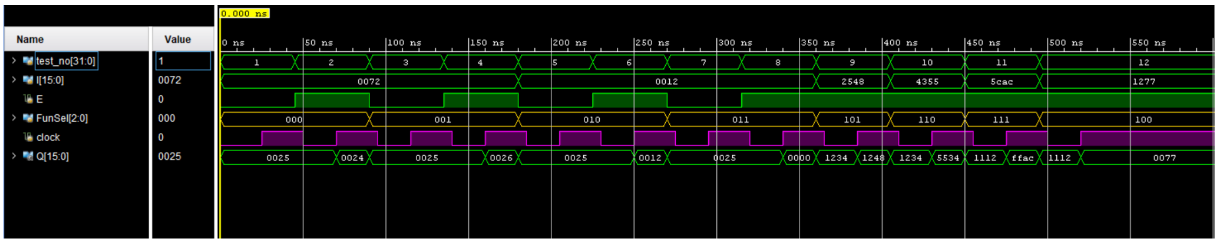
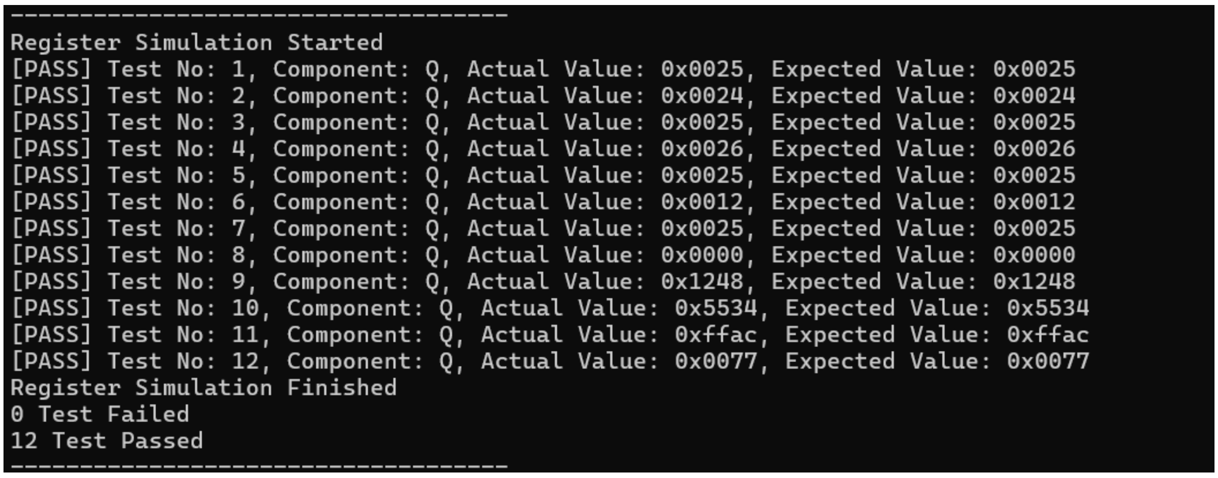


Figure 9: Register Simulation



```
-----------------------------------
Register Simulation Started
[PASS] Test No: 1, Component: Q, Actual Value: 0x0025, Expected Value: 0x0025
[PASS] Test No: 2, Component: Q, Actual Value: 0x0024, Expected Value: 0x0024
[PASS] Test No: 3, Component: Q, Actual Value: 0x0025, Expected Value: 0x0025
[PASS] Test No: 4, Component: Q, Actual Value: 0x0026, Expected Value: 0x0026
[PASS] Test No: 5, Component: Q, Actual Value: 0x0025, Expected Value: 0x0025
[PASS] Test No: 6, Component: Q, Actual Value: 0x0012, Expected Value: 0x0012
[PASS] Test No: 7, Component: Q, Actual Value: 0x0025, Expected Value: 0x0025
[PASS] Test No: 8, Component: Q, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 9, Component: Q, Actual Value: 0x1248, Expected Value: 0x1248
[PASS] Test No: 10, Component: Q, Actual Value: 0x5534, Expected Value: 0x5534
[PASS] Test No: 11, Component: Q, Actual Value: 0xffac, Expected Value: 0xffac
[PASS] Test No: 12, Component: Q, Actual Value: 0x0077, Expected Value: 0x0077
Register Simulation Finished
0 Test Failed
12 Test Passed
-----------------------------------
```

Figure 10: Register Tests

3.2 SIMULATION OF PART-2a

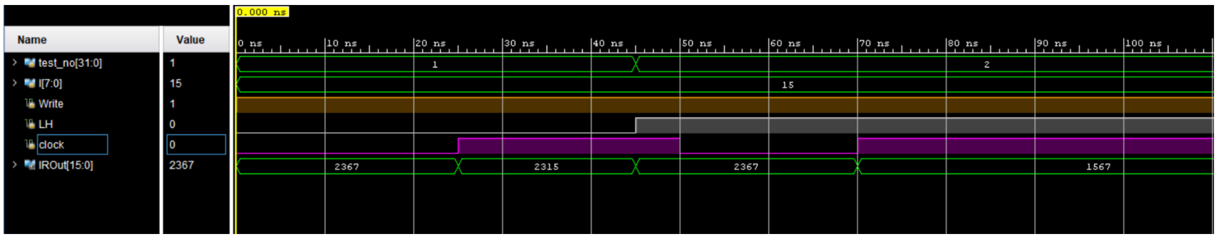
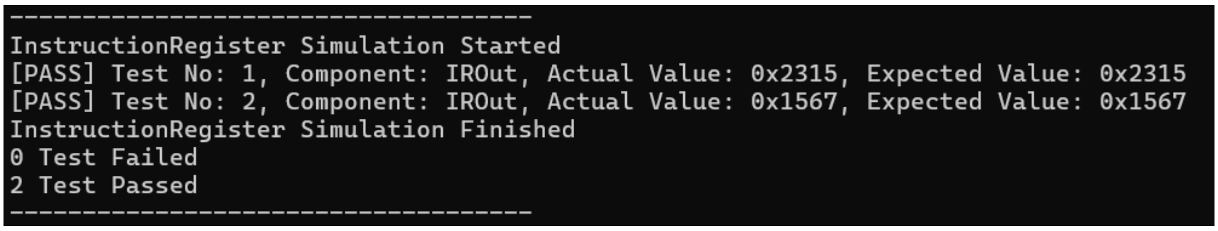


Figure 11: Instruction Register Simulation



```
------------------------------------
InstructionRegister Simulation Started
[PASS] Test No: 1, Component: IROut, Actual Value: 0x2315, Expected Value: 0x2315
[PASS] Test No: 2, Component: IROut, Actual Value: 0x1567, Expected Value: 0x1567
InstructionRegister Simulation Finished
0 Test Failed
2 Test Passed
------------------------------------
```

Figure 12: Instruction Register Tests

3.3 SIMULATION OF PART-2b

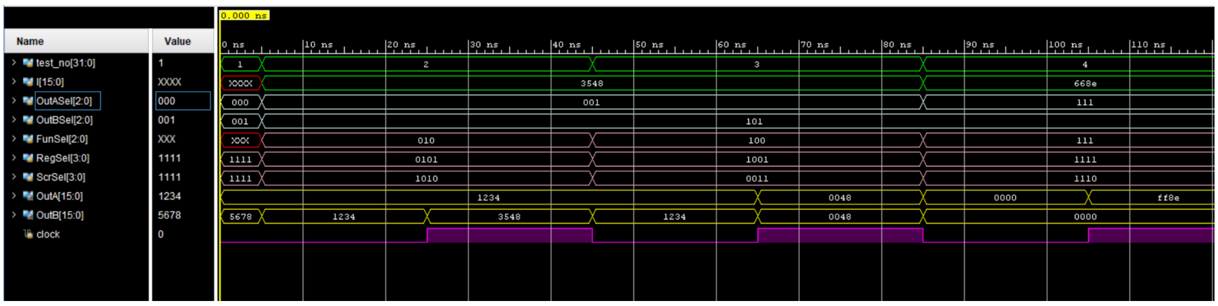


Figure 13: Part 1 of Register File Simulation

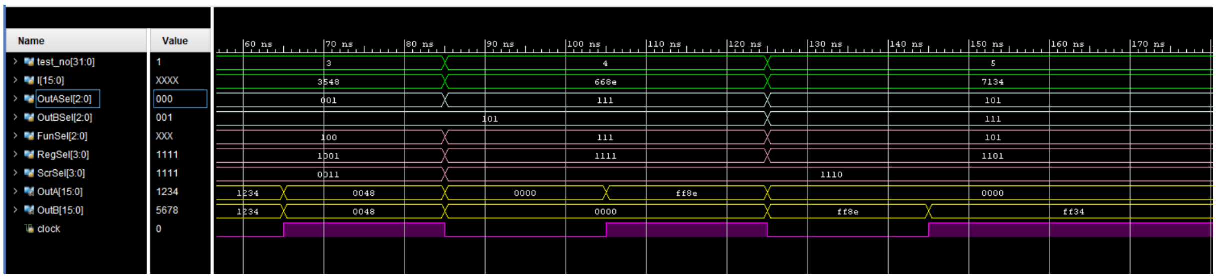
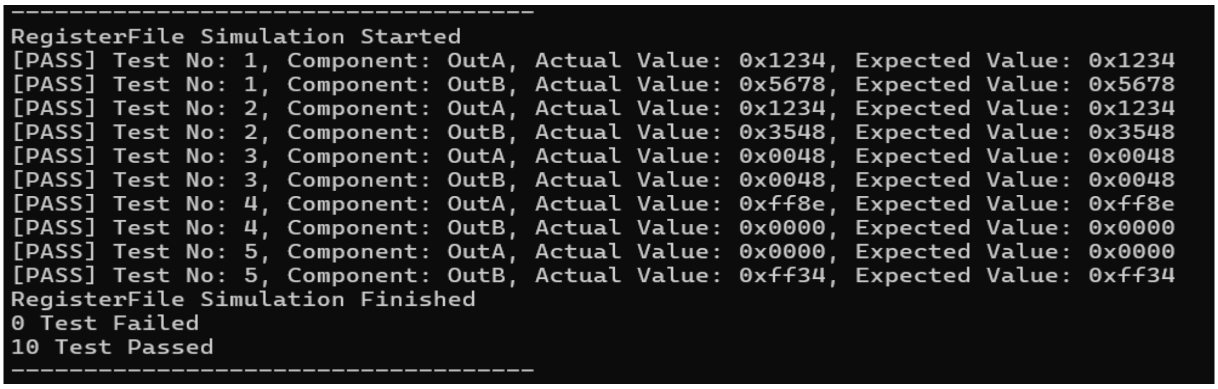


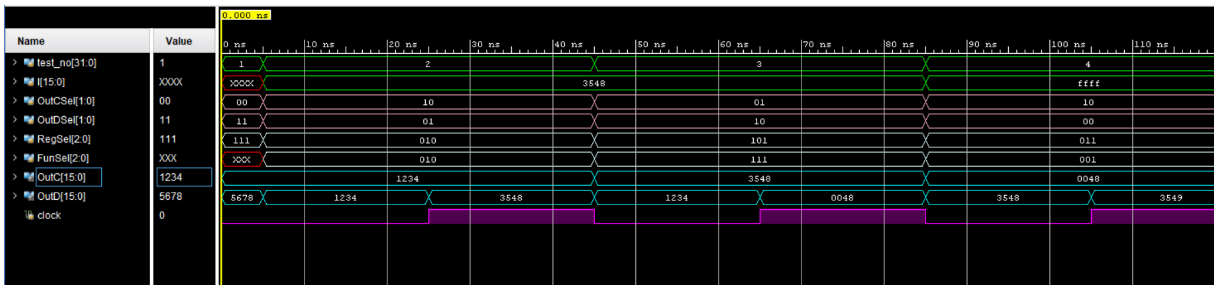
Figure 14: Part 2 of Register File Simulation



```
------------------------------------
RegisterFile Simulation Started
[PASS] Test No: 1, Component: OutA, Actual Value: 0x1234, Expected Value: 0x1234
[PASS] Test No: 1, Component: OutB, Actual Value: 0x5678, Expected Value: 0x5678
[PASS] Test No: 2, Component: OutA, Actual Value: 0x1234, Expected Value: 0x1234
[PASS] Test No: 2, Component: OutB, Actual Value: 0x3548, Expected Value: 0x3548
[PASS] Test No: 3, Component: OutA, Actual Value: 0x0048, Expected Value: 0x0048
[PASS] Test No: 3, Component: OutB, Actual Value: 0x0048, Expected Value: 0x0048
[PASS] Test No: 4, Component: OutA, Actual Value: 0xff8e, Expected Value: 0xff8e
[PASS] Test No: 4, Component: OutB, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 5, Component: OutA, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 5, Component: OutB, Actual Value: 0xff34, Expected Value: 0xff34
RegisterFile Simulation Finished
0 Test Failed
10 Test Passed
------------------------------------
```

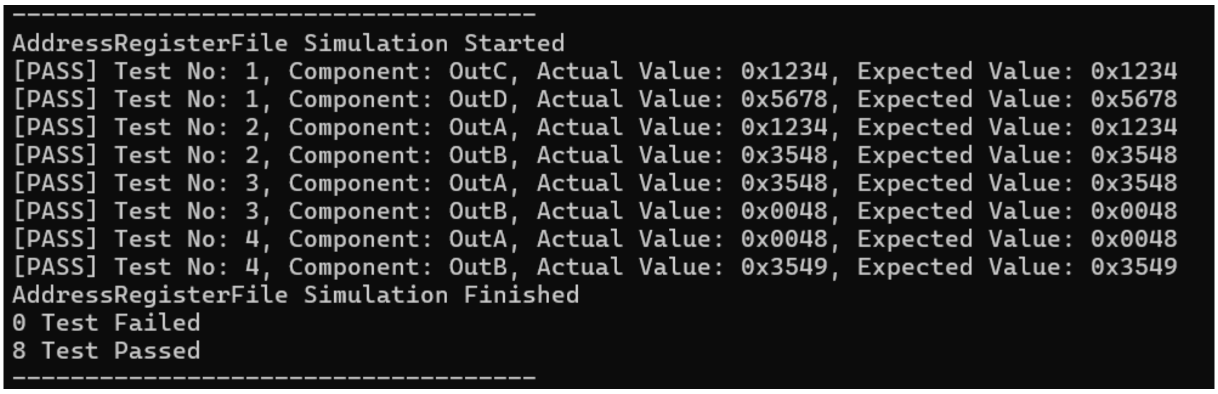
Figure 15: Register File Tests

3.4 SIMULATION OF PART-2c



```
Name              Value
test_no[31:0]     1
I[15:0]           XXXX
OutCSel[1:0]      00
OutDSel[1:0]      11
RegSel[2:0]       111
FunSel[2:0]       XXX
OutC[15:0]        1234
OutD[15:0]        5678
clock             0
```

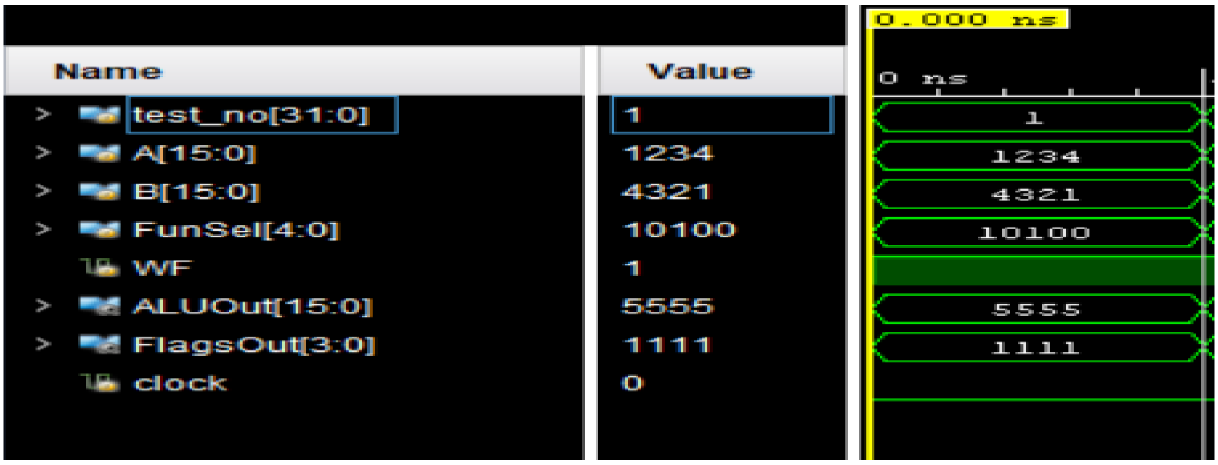
Figure 16: Address Register File Simulation



```
-----------------------------------
AddressRegisterFile Simulation Started
[PASS] Test No: 1, Component: OutC, Actual Value: 0x1234, Expected Value: 0x1234
[PASS] Test No: 1, Component: OutD, Actual Value: 0x5678, Expected Value: 0x5678
[PASS] Test No: 2, Component: OutA, Actual Value: 0x1234, Expected Value: 0x1234
[PASS] Test No: 2, Component: OutB, Actual Value: 0x3548, Expected Value: 0x3548
[PASS] Test No: 3, Component: OutA, Actual Value: 0x3548, Expected Value: 0x3548
[PASS] Test No: 3, Component: OutB, Actual Value: 0x0048, Expected Value: 0x0048
[PASS] Test No: 4, Component: OutA, Actual Value: 0x0048, Expected Value: 0x0048
[PASS] Test No: 4, Component: OutB, Actual Value: 0x3549, Expected Value: 0x3549
AddressRegisterFile Simulation Finished
0 Test Failed
8 Test Passed
-----------------------------------
```

Figure 17: Address Register File Tests

3.5 SIMULATION OF PART-3



```
Name              Value
test_no[31:0]     1
A[15:0]           1234
B[15:0]           4321
FunSel[4:0]       10100
WF                1
ALUOut[15:0]      5555
FlagsOut[3:0]     1111
clock             0
```

Figure 18: ALU Simulation Part1

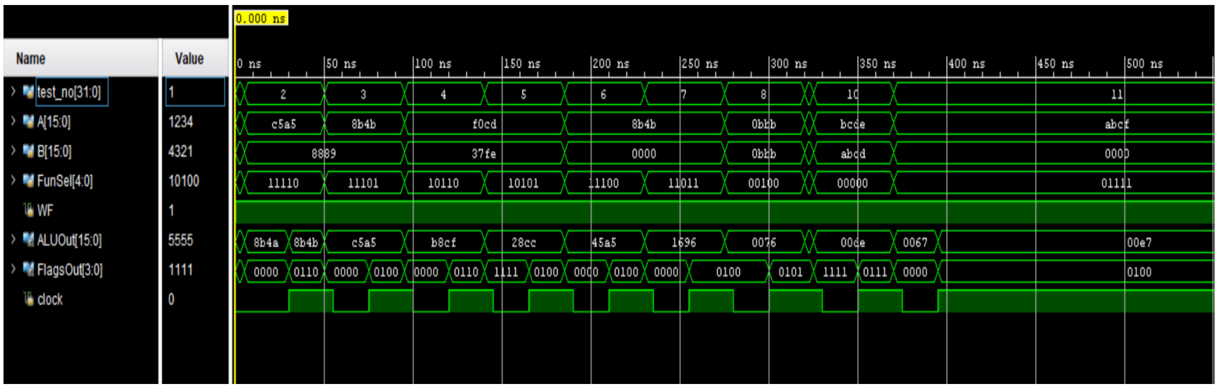
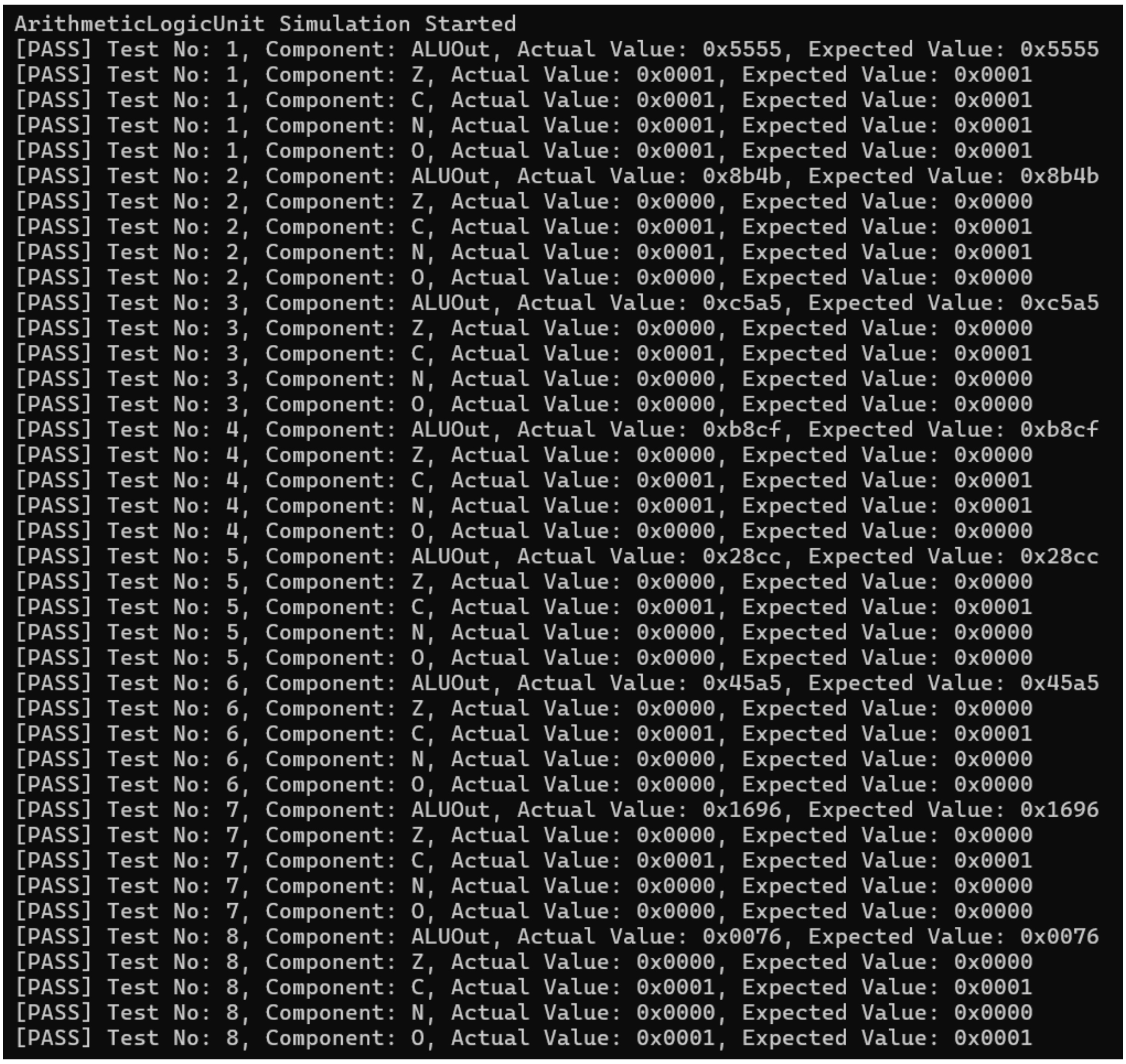
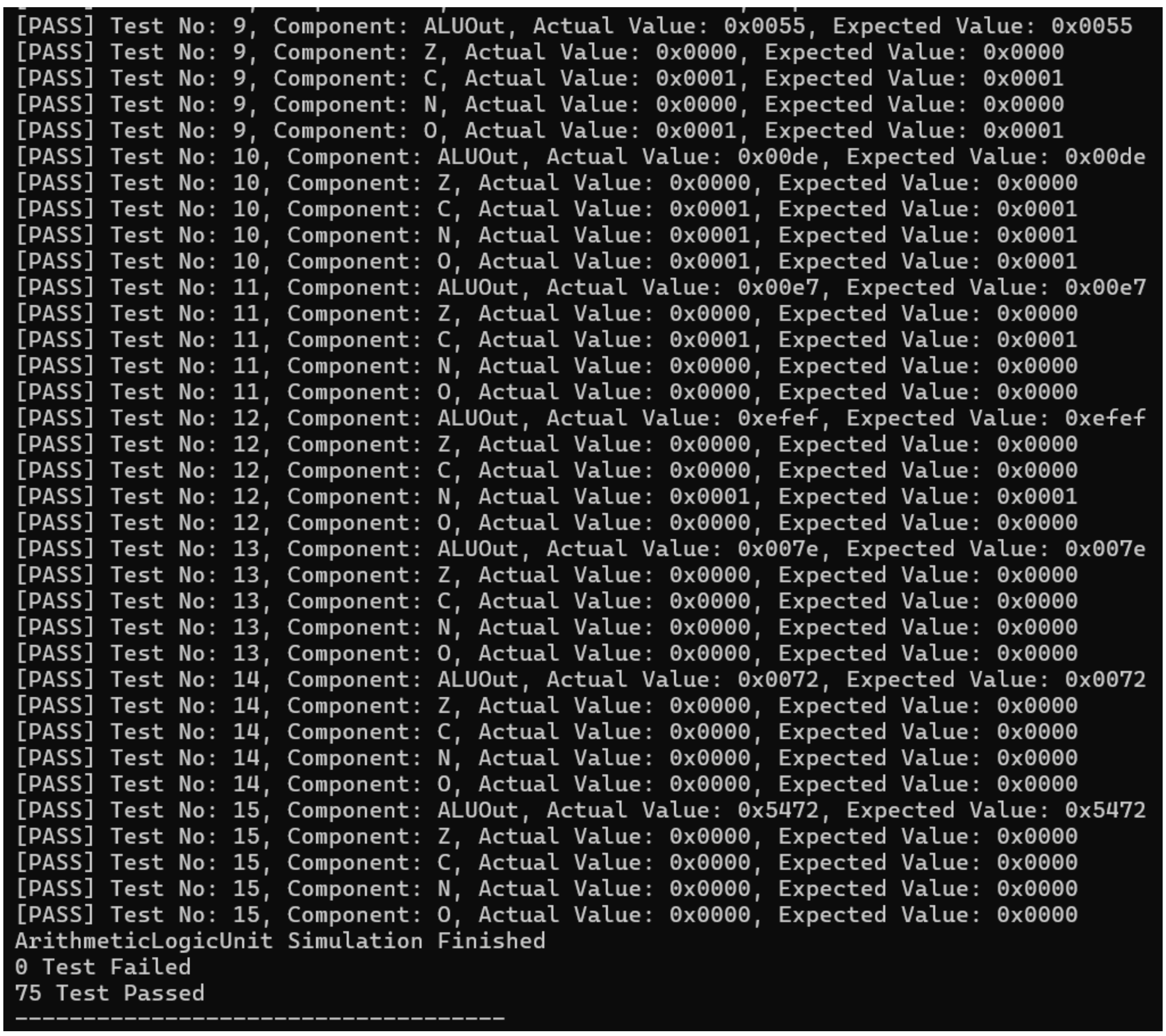


Figure 19: ALU Simulation Part2



```
ArithmeticLogicUnit Simulation Started
[PASS] Test No: 1, Component: ALUOut, Actual Value: 0x5555, Expected Value: 0x5555
[PASS] Test No: 1, Component: Z, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 1, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 1, Component: N, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 1, Component: O, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 2, Component: ALUOut, Actual Value: 0x8b4b, Expected Value: 0x8b4b
[PASS] Test No: 2, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 2, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 2, Component: N, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 2, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 3, Component: ALUOut, Actual Value: 0xc5a5, Expected Value: 0xc5a5
[PASS] Test No: 3, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 3, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 3, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 3, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 4, Component: ALUOut, Actual Value: 0xb8cf, Expected Value: 0xb8cf
[PASS] Test No: 4, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 4, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 4, Component: N, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 4, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 5, Component: ALUOut, Actual Value: 0x28cc, Expected Value: 0x28cc
[PASS] Test No: 5, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 5, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 5, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 5, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 6, Component: ALUOut, Actual Value: 0x45a5, Expected Value: 0x45a5
[PASS] Test No: 6, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 6, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 6, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 6, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 7, Component: ALUOut, Actual Value: 0x1696, Expected Value: 0x1696
[PASS] Test No: 7, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 7, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 7, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 7, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 8, Component: ALUOut, Actual Value: 0x0076, Expected Value: 0x0076
[PASS] Test No: 8, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 8, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 8, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 8, Component: O, Actual Value: 0x0001, Expected Value: 0x0001
```

Figure 20: ALU Tests Part1



```
[PASS] Test No: 9, Component: ALUOut, Actual Value: 0x0055, Expected Value: 0x0055
[PASS] Test No: 9, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 9, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 9, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 9, Component: O, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 10, Component: ALUOut, Actual Value: 0x00de, Expected Value: 0x00de
[PASS] Test No: 10, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 10, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 10, Component: N, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 10, Component: O, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 11, Component: ALUOut, Actual Value: 0x00e7, Expected Value: 0x00e7
[PASS] Test No: 11, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 11, Component: C, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 11, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 11, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 12, Component: ALUOut, Actual Value: 0xefef, Expected Value: 0xefef
[PASS] Test No: 12, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 12, Component: C, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 12, Component: N, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 12, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 13, Component: ALUOut, Actual Value: 0x007e, Expected Value: 0x007e
[PASS] Test No: 13, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 13, Component: C, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 13, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 13, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 14, Component: ALUOut, Actual Value: 0x0072, Expected Value: 0x0072
[PASS] Test No: 14, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 14, Component: C, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 14, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 14, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 15, Component: ALUOut, Actual Value: 0x5472, Expected Value: 0x5472
[PASS] Test No: 15, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 15, Component: C, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 15, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 15, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
ArithmeticLogicUnit Simulation Finished
0 Test Failed
75 Test Passed
------------------------------------
```

Figure 21: ALU Tests Part2

3.6 SIMULATION OF PART-4

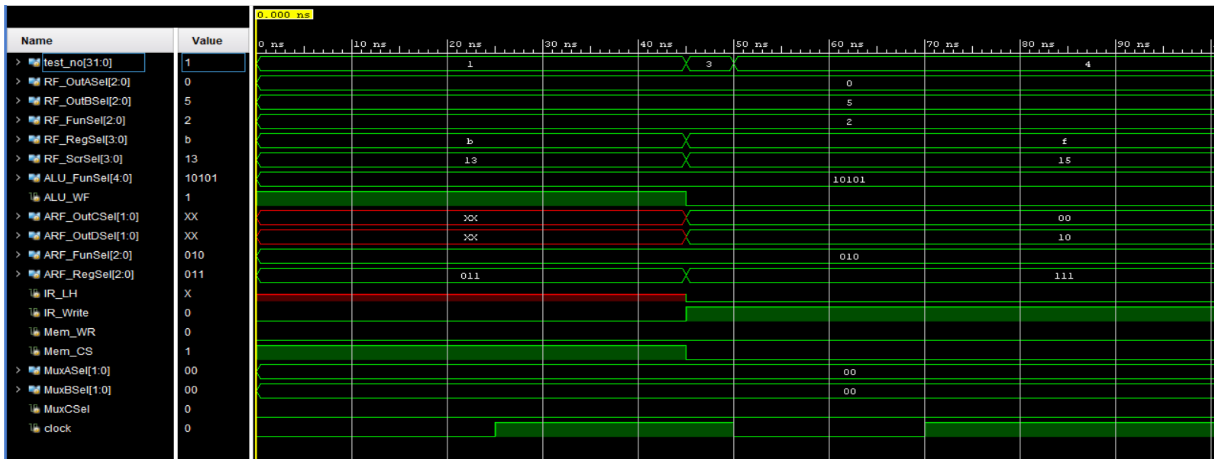


Figure 22: ALU System Simulation

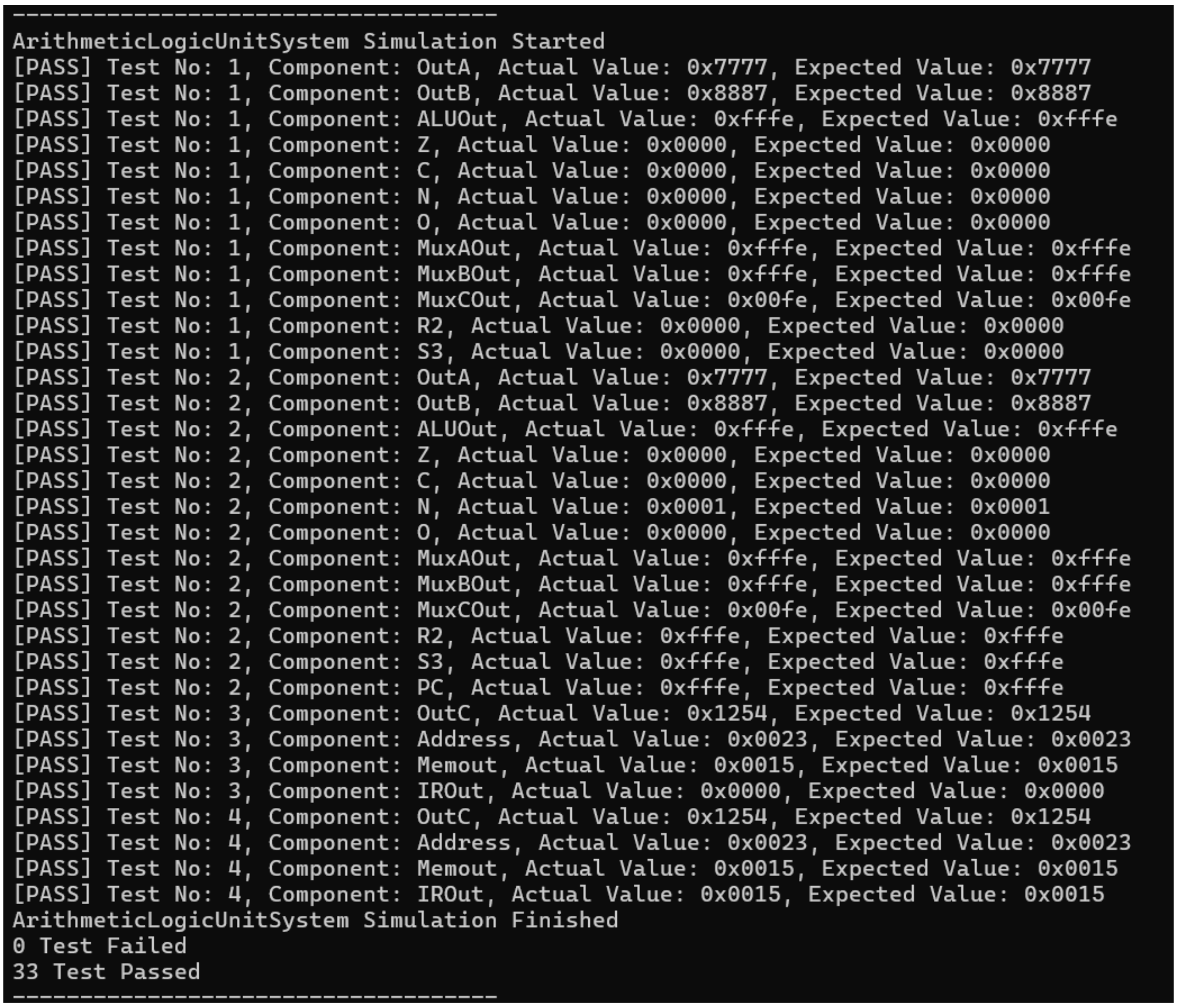
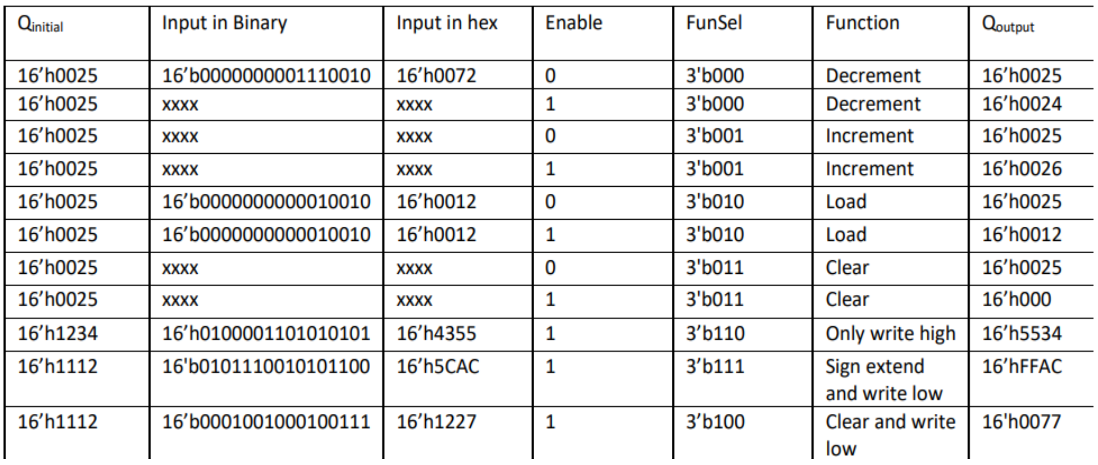

```
------------------------------------
ArithmeticLogicUnitSystem Simulation Started
[PASS] Test No: 1, Component: OutA, Actual Value: 0x7777, Expected Value: 0x7777
[PASS] Test No: 1, Component: OutB, Actual Value: 0x8887, Expected Value: 0x8887
[PASS] Test No: 1, Component: ALUOut, Actual Value: 0xfffe, Expected Value: 0xfffe
[PASS] Test No: 1, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 1, Component: C, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 1, Component: N, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 1, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 1, Component: MuxAOut, Actual Value: 0xfffe, Expected Value: 0xfffe
[PASS] Test No: 1, Component: MuxBOut, Actual Value: 0xfffe, Expected Value: 0xfffe
[PASS] Test No: 1, Component: MuxCOut, Actual Value: 0x00fe, Expected Value: 0x00fe
[PASS] Test No: 1, Component: R2, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 1, Component: S3, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 2, Component: OutA, Actual Value: 0x7777, Expected Value: 0x7777
[PASS] Test No: 2, Component: OutB, Actual Value: 0x8887, Expected Value: 0x8887
[PASS] Test No: 2, Component: ALUOut, Actual Value: 0xfffe, Expected Value: 0xfffe
[PASS] Test No: 2, Component: Z, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 2, Component: C, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 2, Component: N, Actual Value: 0x0001, Expected Value: 0x0001
[PASS] Test No: 2, Component: O, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 2, Component: MuxAOut, Actual Value: 0xfffe, Expected Value: 0xfffe
[PASS] Test No: 2, Component: MuxBOut, Actual Value: 0xfffe, Expected Value: 0xfffe
[PASS] Test No: 2, Component: MuxCOut, Actual Value: 0x00fe, Expected Value: 0x00fe
[PASS] Test No: 2, Component: R2, Actual Value: 0xfffe, Expected Value: 0xfffe
[PASS] Test No: 2, Component: S3, Actual Value: 0xfffe, Expected Value: 0xfffe
[PASS] Test No: 2, Component: PC, Actual Value: 0xfffe, Expected Value: 0xfffe
[PASS] Test No: 3, Component: OutC, Actual Value: 0x1254, Expected Value: 0x1254
[PASS] Test No: 3, Component: Address, Actual Value: 0x0023, Expected Value: 0x0023
[PASS] Test No: 3, Component: Memout, Actual Value: 0x0015, Expected Value: 0x0015
[PASS] Test No: 3, Component: IROut, Actual Value: 0x0000, Expected Value: 0x0000
[PASS] Test No: 4, Component: OutC, Actual Value: 0x1254, Expected Value: 0x1254
[PASS] Test No: 4, Component: Address, Actual Value: 0x0023, Expected Value: 0x0023
[PASS] Test No: 4, Component: Memout, Actual Value: 0x0015, Expected Value: 0x0015
[PASS] Test No: 4, Component: IROut, Actual Value: 0x0015, Expected Value: 0x0015
ArithmeticLogicUnitSystem Simulation Finished
0 Test Failed
33 Test Passed
------------------------------------
```

Figure 23: ALU System Tests

4 DISCUSSION

4.1 PART 1

The cases that we test for the register part can be seen in Figure 24.

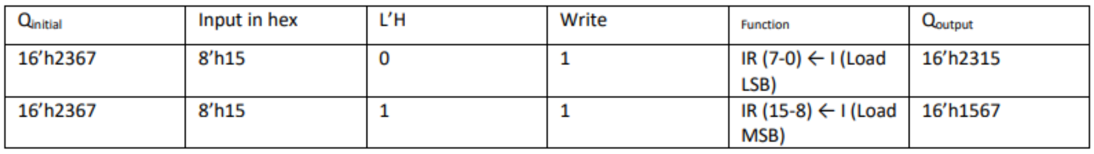


$Q_{initial}$	Input in Binary	Input in hex	Enable	FunSel	Function	Q_{output}
16’h0025	16’b0000000001110010	16’h0072	0	3'b000	Decrement	16’h0025
16’h0025	xxxx	xxxx	1	3'b000	Decrement	16’h0024
16’h0025	xxxx	xxxx	0	3'b001	Increment	16’h0025
16’h0025	xxxx	xxxx	1	3'b001	Increment	16’h0026
16’h0025	16’b0000000000010010	16’h0012	0	3'b010	Load	16’h0025
16’h0025	16’b0000000000010010	16’h0012	1	3'b010	Load	16’h0012
16’h0025	xxxx	xxxx	0	3'b011	Clear	16’h0025
16’h0025	xxxx	xxxx	1	3'b011	Clear	16’h000
16’h1234	16’h0100001101010101	16’h4355	1	3’b110	Only write high	16’h5534
16’h1112	16'b0101110010101100	16’h5CAC	1	3’b111	Sign extend and write low	16’hFFAC
16’h1112	16’b0001001000100111	16’h1227	1	3’b100	Clear and write low	16’h0077

Figure 24: Register Table

As can be seen from the table, when the enable input is 0, it retains its own value. In some tests, we assigned some values to the register’s Q without using the load function to test it. When the clock signal goes from low to high, our register applies the selected function with the enable signal. The results were as expected.

4.2 PART-2a



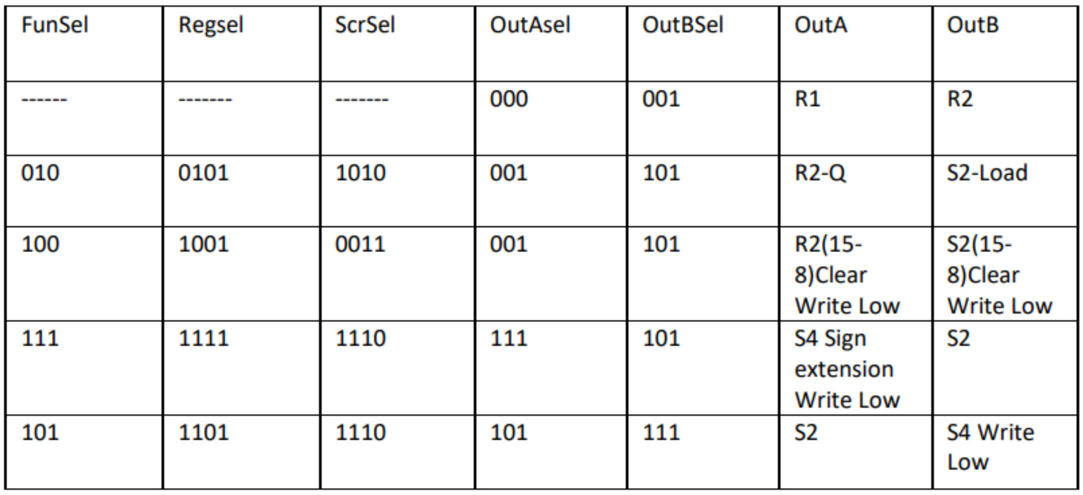
$Q_{initial}$	Input in hex	L’H	Write	Function	Q_{output}
16’h2367	8’h15	0	1	IR (7-0) ← I (Load LSB)	16’h2315
16’h2367	8’h15	1	1	IR (15-8) ← I (Load MSB)	16’h1567

Figure 25: Instruction Register Table

Since the Instruction Register is not a complex module, we used the given simulation tests. The Instruction Register successfully writes the input to the selected 8-bit part of its Q, according to the L’H signal and with the write and the clock signal.

4.3 PART-2b

Register File tests can be seen as in Figure 26. With the register file, we can select enabled general purpose registers and enabled scratch registers. With OutASel, we select one register to take input from. OutBSel works the same as OutASel. In some tests, we initialize the Q values of the registers. So, we represented output without loading registers. The results were as expected in the table.

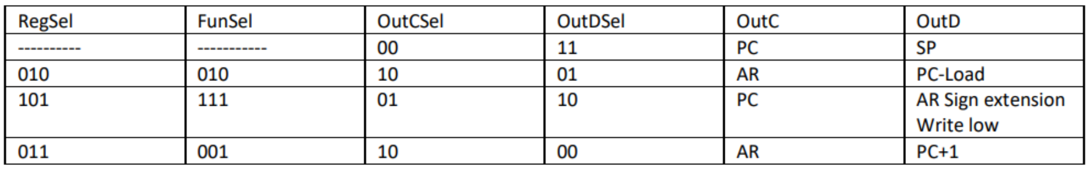


FunSel	Regsel	ScrSel	OutAsel	OutBSel	OutA	OutB
------	-------	-------	000	001	R1	R2
010	0101	1010	001	101	R2-Q	S2-Load
100	1001	0011	001	101	R2(15-8)Clear Write Low	S2(15-8)Clear Write Low
111	1111	1110	111	101	S4 Sign extension Write Low	S2
101	1101	1110	101	111	S2	S4 Write Low

Figure 26: Register File Table

4.4 PART-2c

Address Register File tests can be seen as in 27.



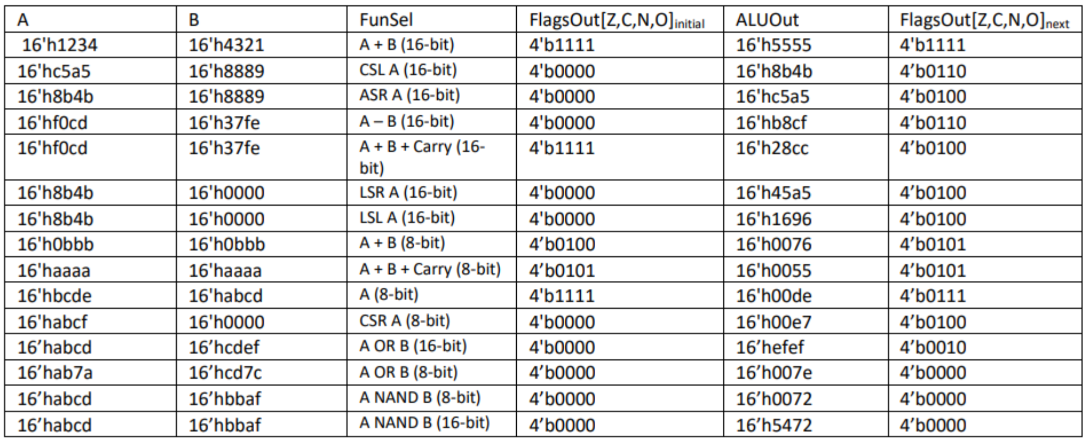
RegSel	FunSel	OutCSel	OutDSel	OutC	OutD
----------	-----------	00	11	PC	SP
010	010	10	01	AR	PC-Load
101	111	01	10	PC	AR Sign extension Write low
011	001	10	00	AR	PC+1

Figure 27: Address Register File Table

The address register file has three registers. We can select the enabled ones with RegSel. When the clock signal goes from low to high, enabled registers operate the selected function. In test 1, we initialized some values to registers, so we did not send RegSel and FunSel. The results were as expected.

4.5 PART-3

The given ALU tests were not sufficient. We added more of them. The ALU module that we designed can work with 8-bit and 16-bit numbers. At the start of some tests, we assigned ZCNO as 0000. ALU operated the function instantly. But the FlagsOut value changed only when the clock signal went from low to high. We also did not reset ZCNO in some tests. So, the FlagsOut value is retained until the clock signal. We observed the FlagsOut value before the clock signal and after the clock signal. The results were as expected. Nevertheless, the calculation of some test results was a bit confusing. So, we may have tested ALU incorrectly. ALU tests can be seen in Figure 28.



A	B	FunSel	FlagsOut[Z,C,N,O]$_{initial}$	ALUOut	FlagsOut[Z,C,N,O]$_{next}$
16'h1234	16'h4321	A + B (16-bit)	4'b1111	16'h5555	4'b1111
16'hc5a5	16'h8889	CSL A (16-bit)	4'b0000	16'h8b4b	4'b0110
16'h8b4b	16'h8889	ASR A (16-bit)	4'b0000	16'hc5a5	4'b0100
16'hf0cd	16'h37fe	A – B (16-bit)	4'b0000	16'hb8cf	4'b0110
16'hf0cd	16'h37fe	A + B + Carry (16-bit)	4'b1111	16'h28cc	4'b0100
16'h8b4b	16'h0000	LSR A (16-bit)	4'b0000	16'h45a5	4'b0100
16'h8b4b	16'h0000	LSL A (16-bit)	4'b0000	16'h1696	4'b0100
16'h0bbb	16'h0bbb	A + B (8-bit)	4'b0100	16'h0076	4'b0101
16'haaaa	16'haaaa	A + B + Carry (8-bit)	4'b0101	16'h0055	4'b0101
16'hbcde	16'habcd	A (8-bit)	4'b1111	16'h00de	4'b0111
16'habcf	16'h0000	CSR A (8-bit)	4'b0000	16'h00e7	4'b0100
16'habcd	16'hcdef	A OR B (16-bit)	4'b0000	16'hefef	4'b0010
16'hab7a	16'hcd7c	A OR B (8-bit)	4'b0000	16'h007e	4'b0000
16'habcd	16'hbbaf	A NAND B (8-bit)	4'b0000	16'h0072	4'b0000
16'habcd	16'hbbaf	A NAND B (16-bit)	4'b0000	16'h5472	4'b0000

Figure 28: ALU Table

4.6 PART-4

In the last part, we have combined all the modules we have designed in the process so far by using several MUXs. In the last part of the project, we can say that the big picture emerged. We didn't add any extra tests for the last part, but our design passed the given tests. We hope it works as expected.

5 CONCLUSION

Throughout the project, we learned how the computer does basic arithmetic operations, how to work with Verilog, and how to develop modules such as register files, address register files, instruction registers, ALUs, and MUXs. We tried to understand the difference between the modules that operate with a clock signal and the modules that always operate. The biggest problem we encountered while doing the project was deciding how to express the 8-bit outputs of ALU in Part 3. We had two options: either we would show the outputs by sign extension or we would fill the MSB (15–8) of the outputs with 0, like 00xx. According to the questions we asked the instructor and the feedback we received, we decided to represent the MSB (15–8) with zeros. Another major problem in the project was the inadequacy of the number of tests in the ALU section. We added tests to the given simulation files ourselves and made the necessary debugs.