ISTANBUL TECHNICAL UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT

BLG 242E
LOGIC CIRCUITS LABORATORY
HOMEWORK 1 REPORT

HOMEWORK NO : 1
GROUP NO : G08

GROUP MEMBERS:

150200081 : KAAN KARATAŞ
150210719 : NACİ TOYGUN GÖRMÜŞ

SPRING 2023

Contents

1. INTRODUCTION **1**

2. Preliminary **1**
2.1. Part 1 . . . 1
2.1.a. Question a . . . 1
2.1.b. Question b . . . 2
2.1.c. Question c . . . 2
2.1.d. Question d . . . 3
2.1.e. Question e . . . 4
2.1.f. Question f . . . 5
2.2. Part 2 . . . 6
2.2.a. Circuit of F1 . . . 6
2.2.b. Circuit of F2 . . . 7
2.3. Part 3 . . . 7

3. Experiments **8**
3.1. Part 1 . . . 8
3.1.a. AND Gate . . . 8
3.1.b. OR Gate . . . 8
3.1.c. NOT Gate . . . 9
3.1.d. XOR Gate . . . 10
3.1.e. NAND Gate . . . 10
3.1.f. 8:1 Multiplexer . . . 11
3.1.g. 3:8 Decoder . . . 13
3.2. Part 2 . . . 15
3.3. Part 3 . . . 17
3.4. Part 4 . . . 18
3.5. Part 5 . . . 19
3.5.a. Part5 - 1 . . . 19
3.5.b. Part5 - 2 . . . 19
3.6. Part 6 . . . 21
3.7. Part 7 . . . 22
3.8. Part 8 . . . 23
3.9. Part 9 . . . 24
3.10. Part 10 . . . 25
3.11. Part 11 . . . 27

4. RESULTS **27**

5. DISCUSSION **27**

6. CONCLUSION **27**

1. INTRODUCTION

In this task, the specified functions were studied in introductory section and circuits needed were created. After that in experiment section part 1, gates were implemented as requested for subsequent use. Eventually, from the second part to the fifth part requested circuits were implemented. The remaining sections involved designing adders with various bit configurations and applying simulations to specific tests.

2. Preliminary

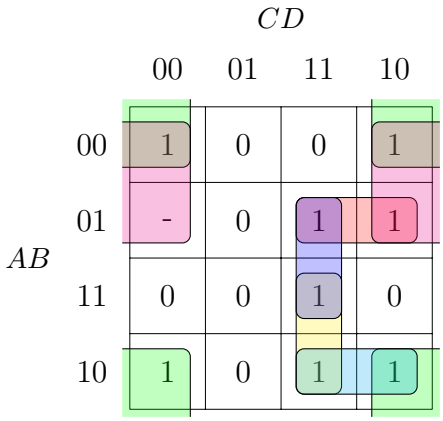
2.1. Part 1

In this part for the given in equation (F) applied the following operations

$$F(a,b,c,d) = \cup(0,2,6,7,8,10,11,15) + \cup_{\phi}(4)$$

2.1.a. Question a

Karnaugh diagram



Karnaugh Map of F

Prime implicants of the function

$$A'D', B'D', A'BC, AB'C, ACD, BCD$$

2.1.b. Question b

Quine-McCluskey method

Group	Decimal	Binary
0	0	0000 ✓
1	2	0010 ✓
1	8	1000 ✓
1	4	0100 ✓
2	6	0110 ✓
2	10	1010 ✓
3	7	0111 ✓
3	11	1011 ✓
4	15	1111 ✓

Group	Decimal	Binary
0	0, 8	-000 ✓
0	0, 4	0-00 ✓
0	0, 2	00-0 ✓
1	10, 2	-010 ✓
1	2, 6	0-10 ✓
1	10, 8	10-0 ✓
1	4, 6	01-0 ✓
2	6, 7	011- ×
2	10, 11	101- ×
3	15, 7	-111 ×
3	11, 15	1-11 ×

Group	Decimal	Binary
0	0, 10, 2, 8	-0-0 ×
0	0, 2, 4, 6	0–0 ×

Prime implicants of the function

$$A'D', B'D', A'BC, AB'C, ACD, BCD$$

2.1.c. Question c

Prime implicants chart

Minterms	0	10	11	15	2	6	7	8	Cost
A'BC						X	X		5
AB'C		X	X						5
BCD				X			X		4
ACD			X	X					4
B'D'	X	X			X			X	5
A'D'	X				X	X			5

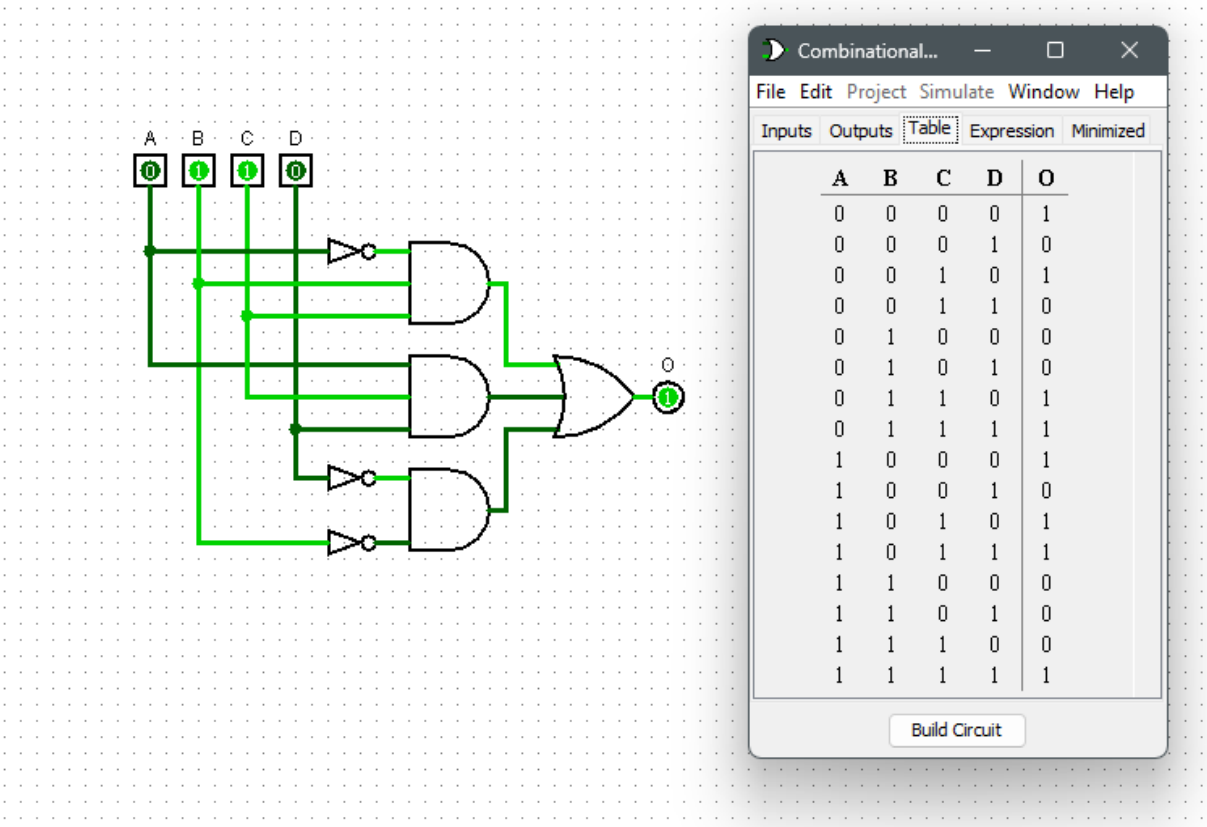
The Essential Prime Implicants Are

$$B'D', A'BC, ACD$$

$$Cost = 8 * 2 + 3 = 19$$

2.1.d. Question d

Circuit design with NOT, AND, and OR gates

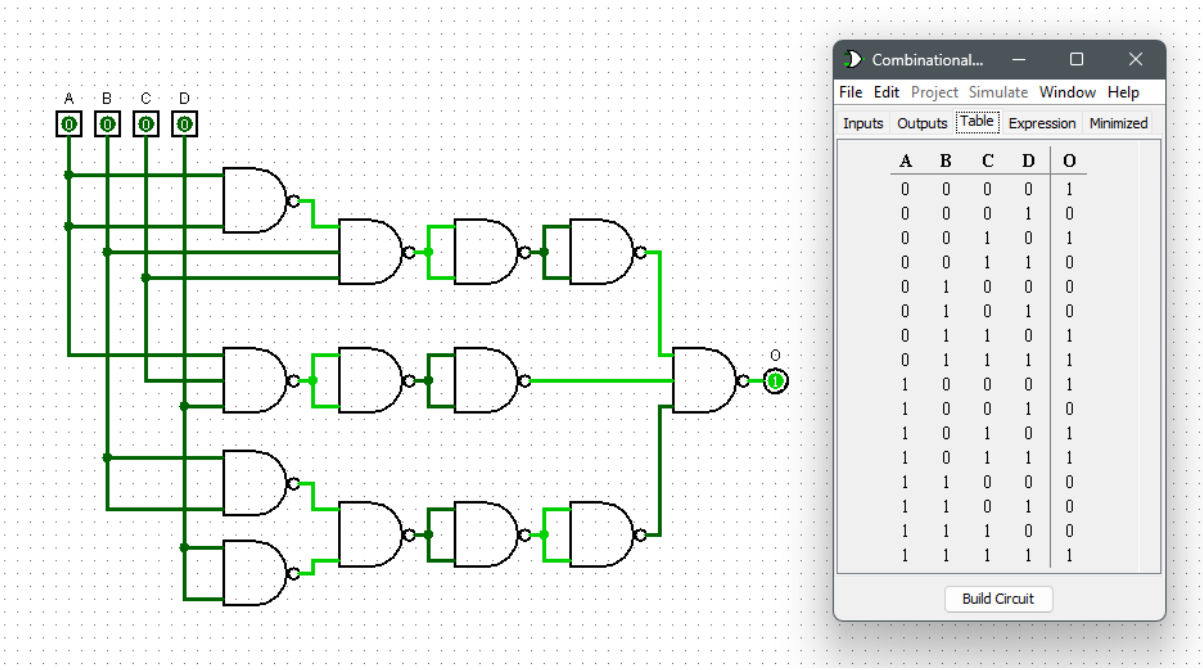


A	B	C	D	O
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Figure 1: Preliminary 1-d Circuit Design

2.1.e. Question e

Circuit design with only NAND gate



A	B	C	D	O
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Figure 2: Preliminary 1-e Circuit Design

2.1.f. Question f

Circuit design with 8:1 Multiplexer, AND, OR and NOT gates

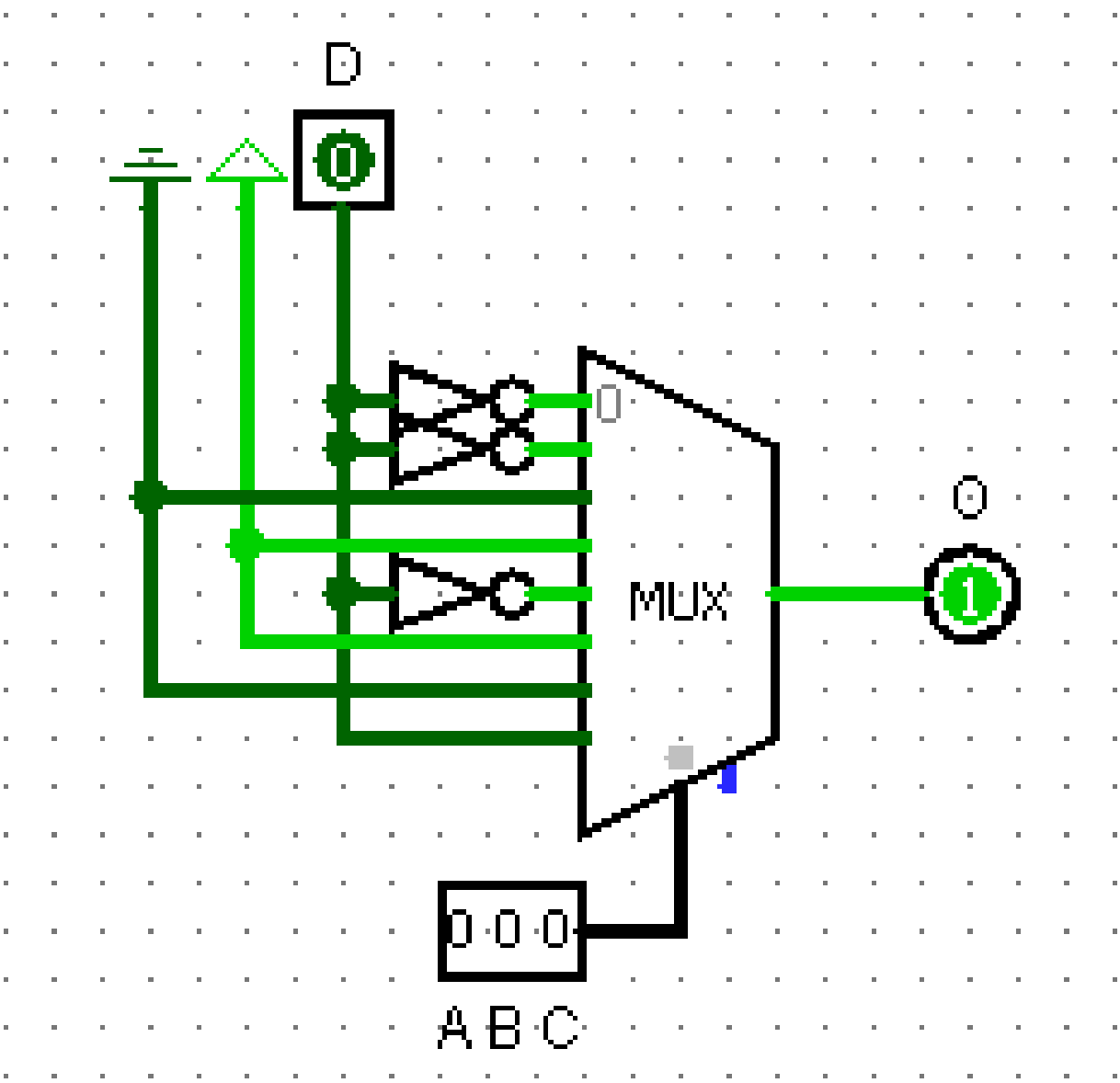


Figure 3: Preliminary 1-f Circuit Design

2.2. Part 2

In this part for the provided equations (F1, F2) designed circuit using a 3:8 decoder and 2-input OR gates

$$F1(a,b,c) = a'bc + ab'$$
$$F2(a,b,c) = abc' + ab$$

2.2.a. Circuit of F1

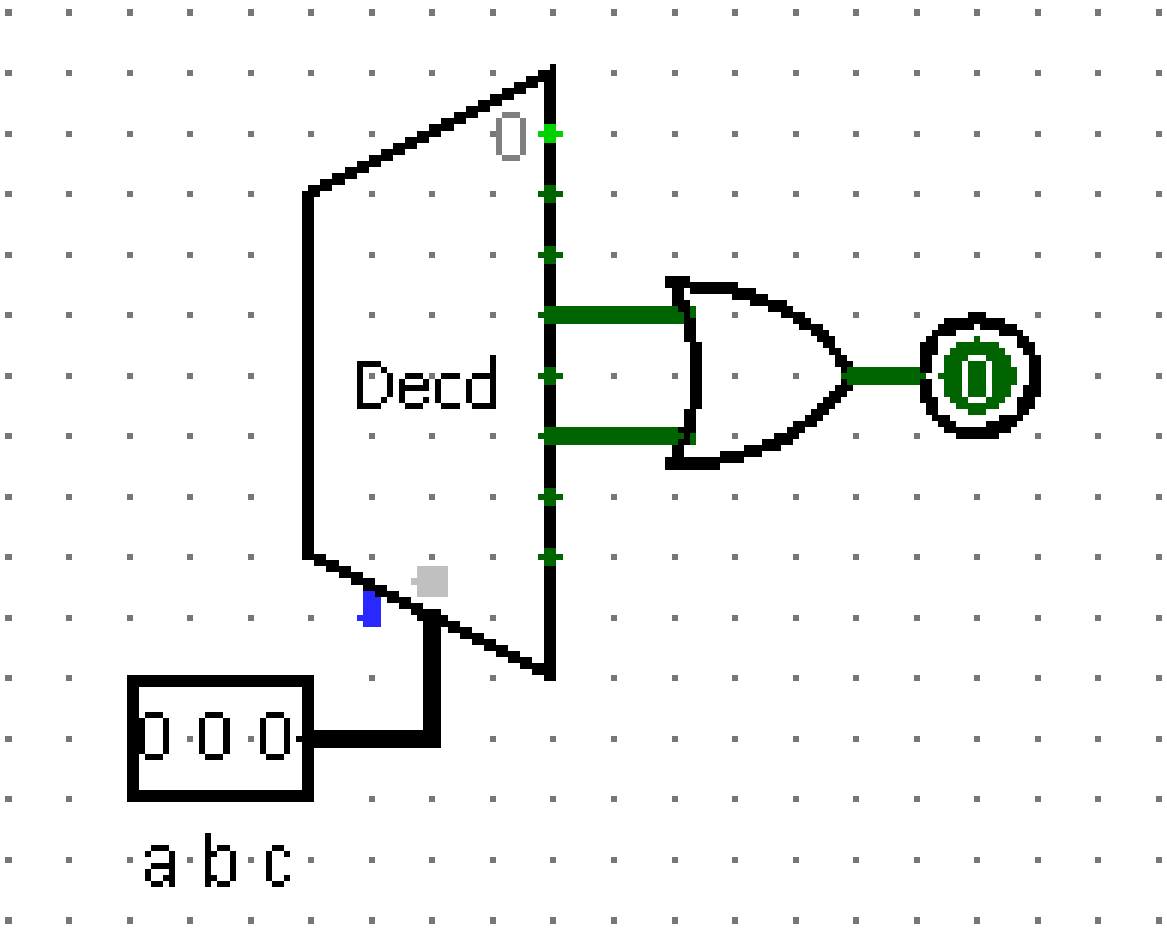


Figure 4: Preliminary 2-a Circuit Design

2.2.b. Circuit of F2

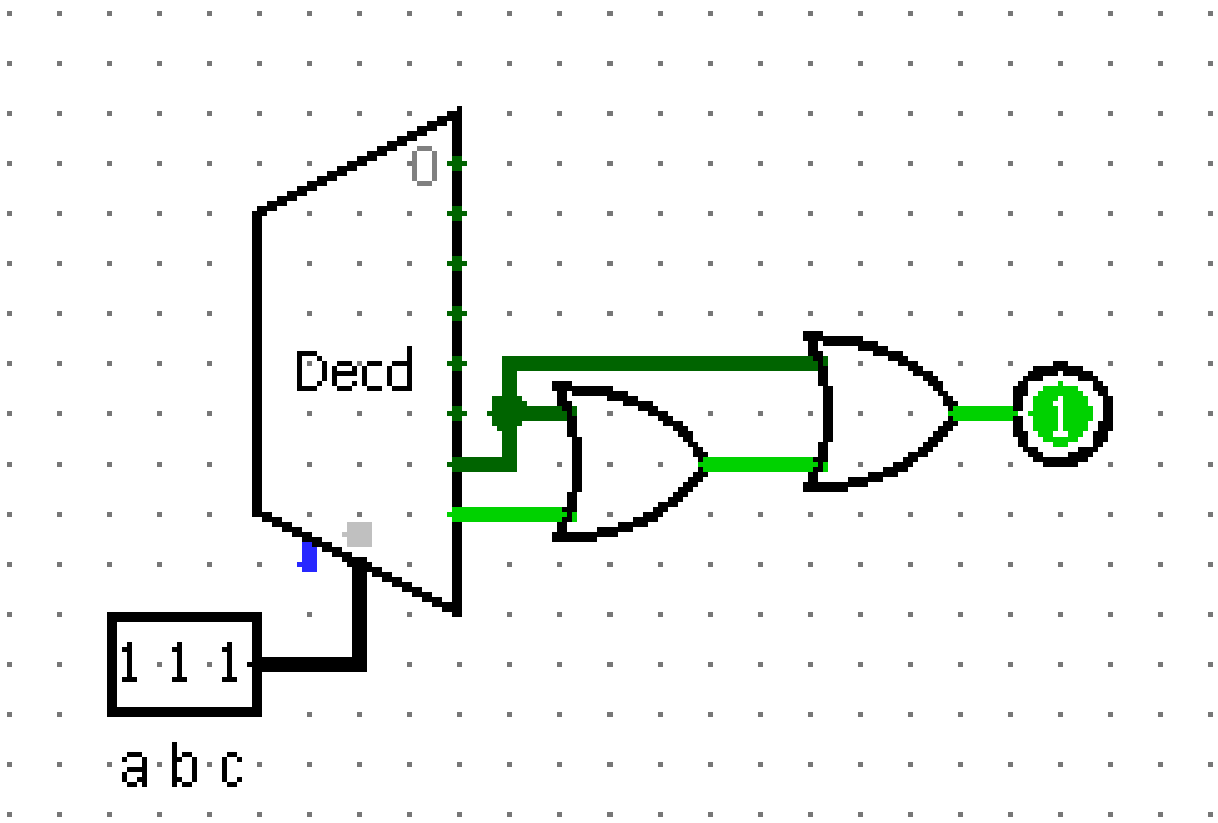


Figure 5: Preliminary 2-b Circuit Design

2.3. Part 3

Signed	Unsigned
Carry bit is important for addition Borrow can occur in subtraction	Carry bit is ignored in both operations Does not have borrow in subtraction

3. Experiments

3.1. Part 1

In this section, we implemented the following gates

3.1.a. AND Gate

```
module and_gate(input A, B, output Z);
    assign Z = A & B;
endmodule
```

Figure 6: AND Gate Code

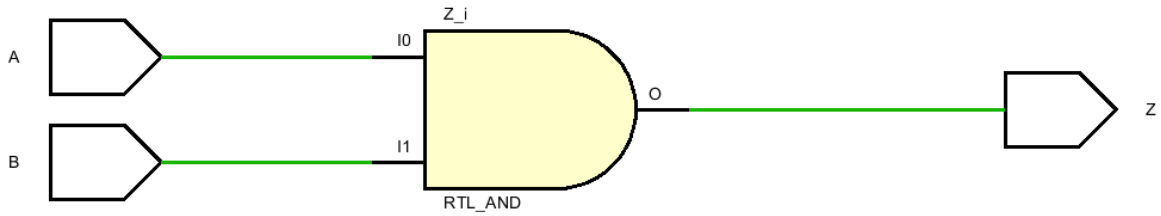


Figure 7: AND Gate Design

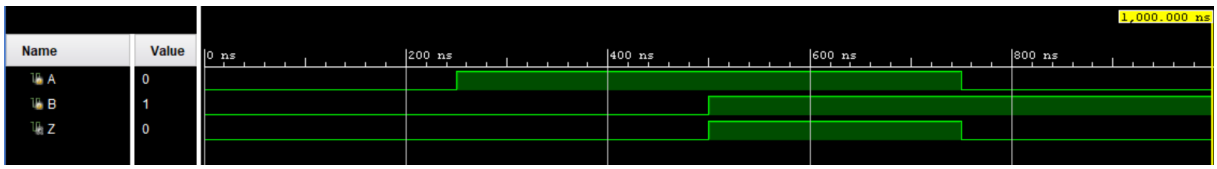


Figure 8: AND Gate Testbench

3.1.b. OR Gate

```
module or_gate(input A, B, output Z);
    assign Z = A | B;
endmodule
```

Figure 9: OR Gate Code

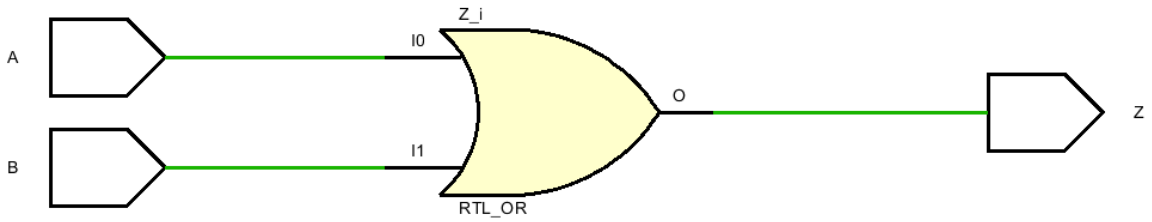


Figure 10: OR Gate Design

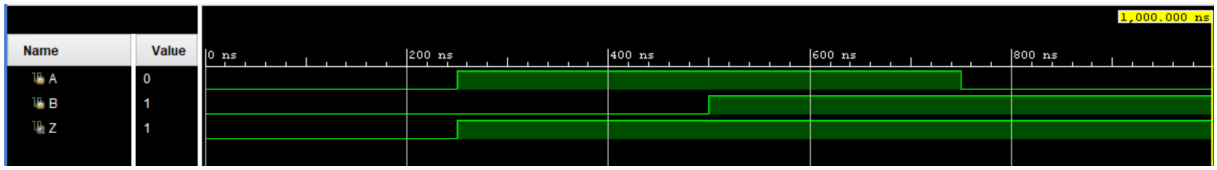


Figure 11: OR Gate Testbench

3.1.c. NOT Gate

```
module not_gate(input A, output Z);
    assign Z = ~ A;
endmodule
```

Figure 12: NOT Gate Code

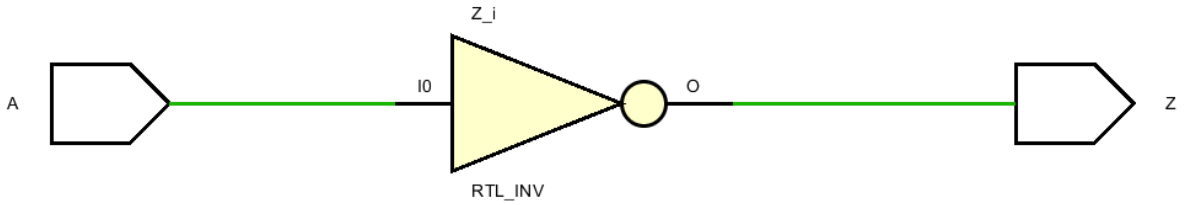


Figure 13: NOT Gate Design

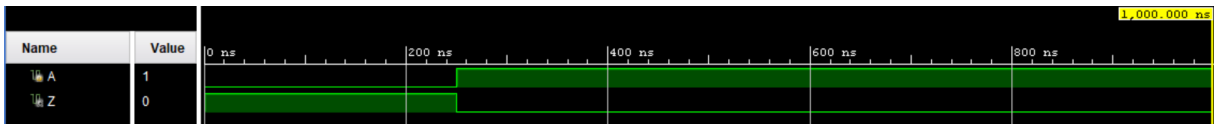


Figure 14: NOT Gate Testbench

3.1.d. XOR Gate

```
module xor_gate(input A, B, output Z);
    wire NA, NB, O0, O1;
    not_gate na(A,NA);
    not_gate nb(B,NB);
    and_gate a1(A, NB, O0);
    and_gate a2(NA, B, O1);
    or_gate o1(O0, O1, Z);
endmodule
```

Figure 15: XOR Gate Code

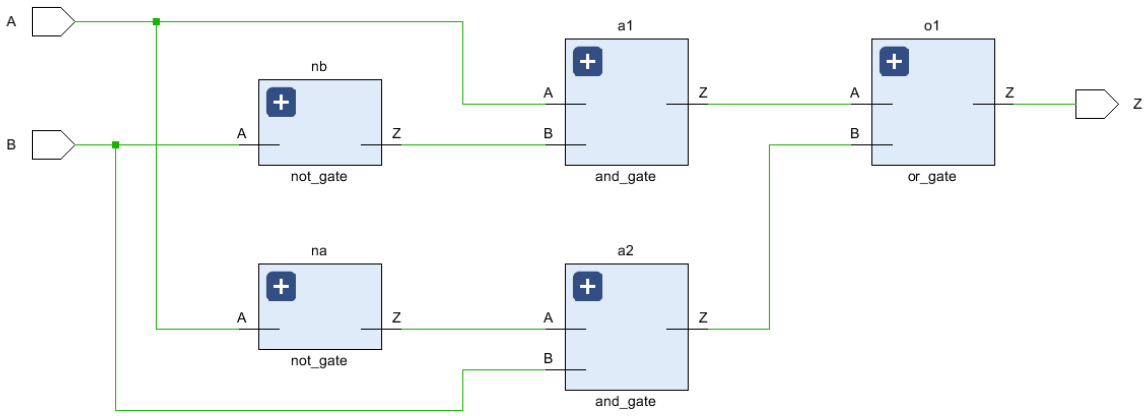


Figure 16: XOR Gate Design

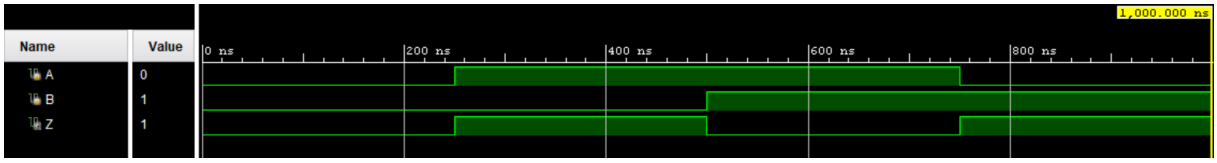


Figure 17: XOR Gate Testbench

3.1.e. NAND Gate

```
module nand_gate(input A, B, output Z);
    assign Z = ~(A & B);
endmodule
```

Figure 18: NAND Gate Code

Figure 19: NAND Gate Design

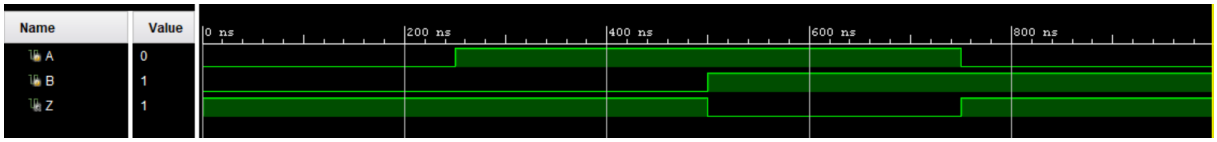


Figure 20: NAND Gate Testbench

3.1.f. 8:1 Multiplexer

```
module mux_2x1(input A, B, S, output Z);
    assign Z = S ? B : A;
endmodule

module mux_4x1(input A, B, C, D, S0, S1, output Z);
    wire O1, O2;
    mux_2x1 m1 (A, B, S0, O1);
    mux_2x1 m2 (C, D, S0, O2);
    mux_2x1 m3 (O1, O2, S1, Z);
endmodule

module mux_8x1(input A, B, C, D, E, F, G, H, S0, S1, S2, output Z);
    wire O1, O2;
    mux_4x1 m1(A, B, S0, S1, O1);
    mux_4x1 m2(C, D, S0, S1, O2);
    mux_2x1 m3(O1, O2, S2, Z);
endmodule
```

Figure 21: MUX8x1 Gate Code

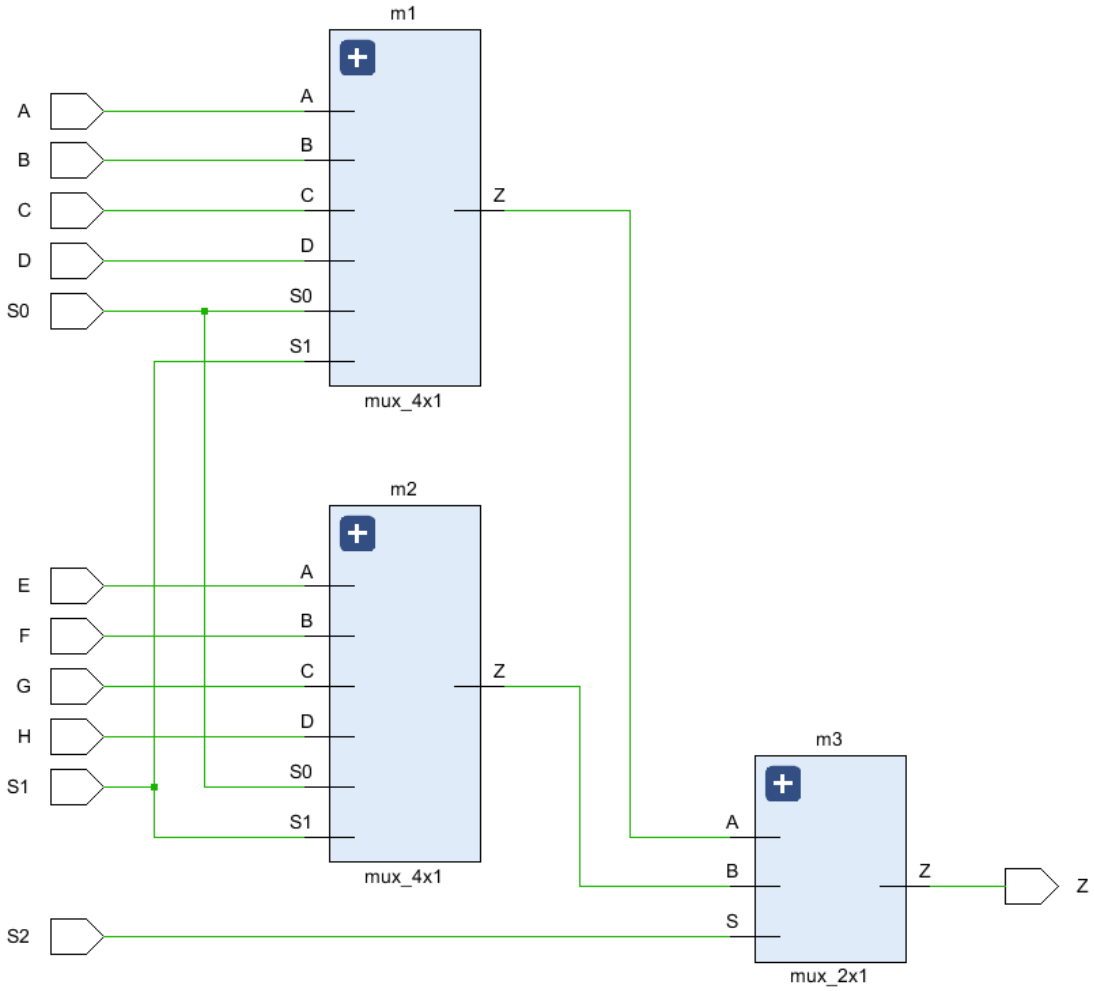


Figure 22: MUX8x1 Gate Design

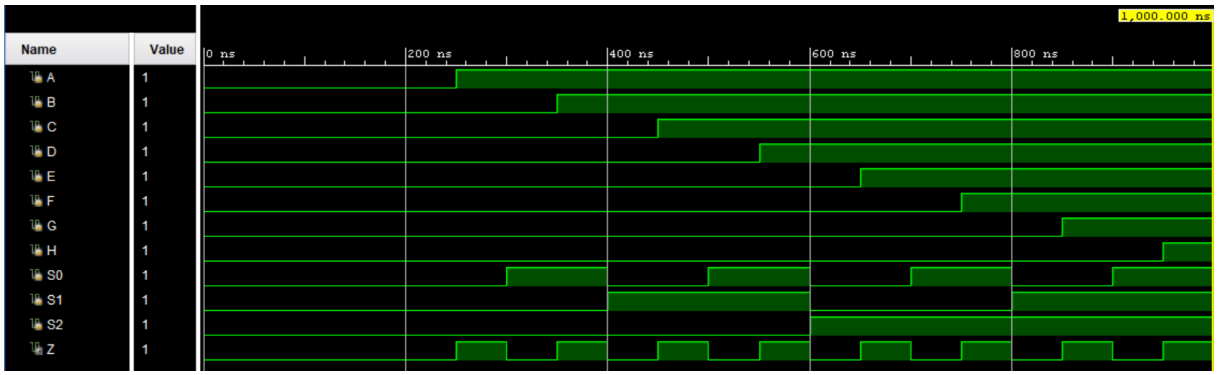
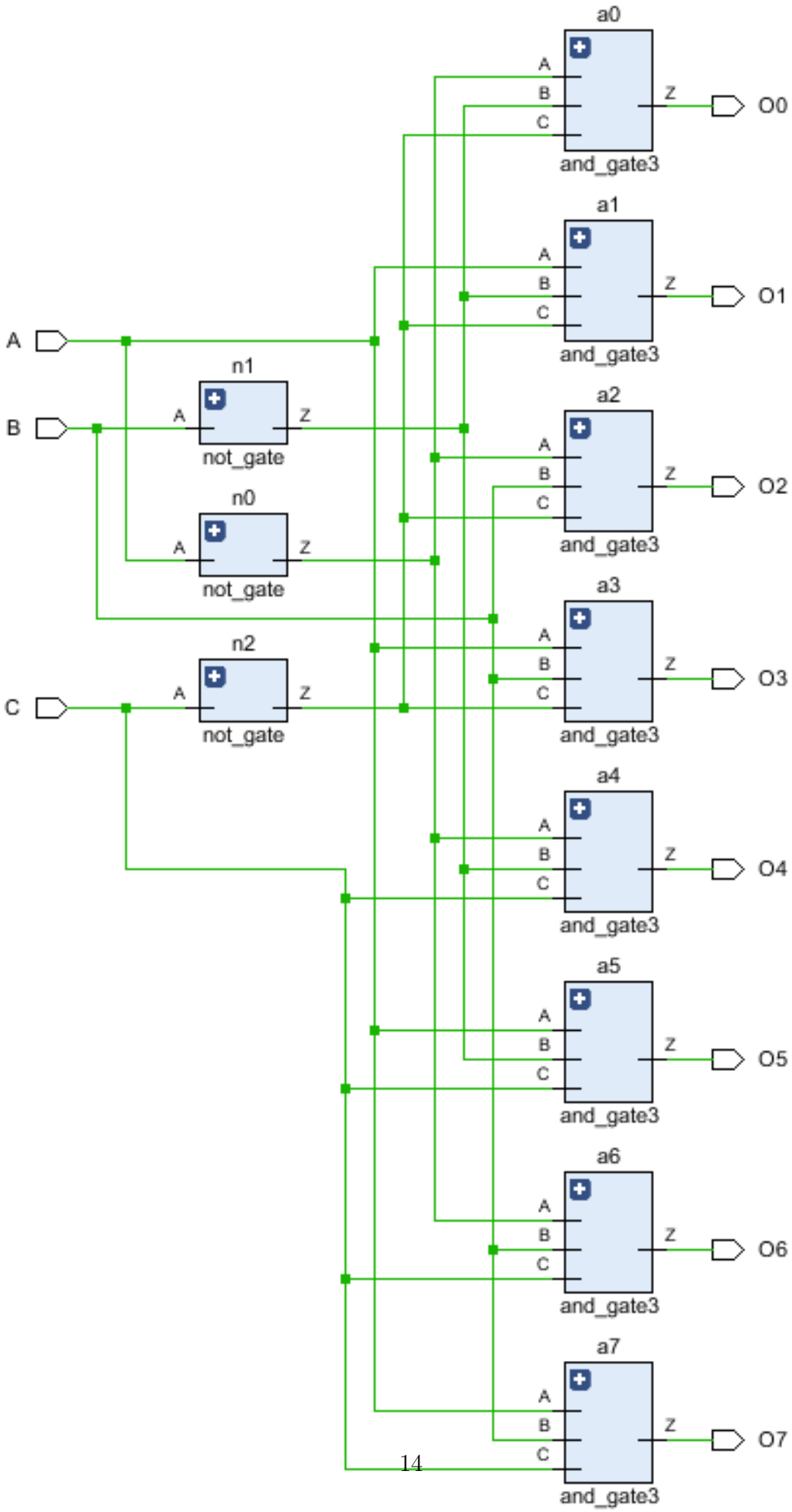


Figure 23: MUX8x1 Gate Testbench

3.1.g. 3:8 Decoder

```
module dec_3x8(input A, B, C, output O0, O1, O2, O3, O4, O5, O6, O7);
    wire NA, NB, NC;
    not_gate n0(A, NA); not_gate n1(B, NB); not_gate n2(C, NC);
    and_gate3 a0(NA, NB, NC, O0);
    and_gate3 a1(A, NB, NC, O1);
    and_gate3 a2(NA, B, NC, O2);
    and_gate3 a3(A, B, NC, O3);
    and_gate3 a4(NA, NB, C, O4);
    and_gate3 a5(A, NB, C, O5);
    and_gate3 a6(NA, B, C, O6);
    and_gate3 a7(A, B, C, O7);
endmodule
```

Figure 24: DEC3x8 Gate Code


a0
A
B
C
Z
O0
and_gate3
a1
O1
a2
O2
a3
O3
a4
O4
a5
O5
a6
O6
a7
O7
A
B
C
n1
not_gate
n0
n2

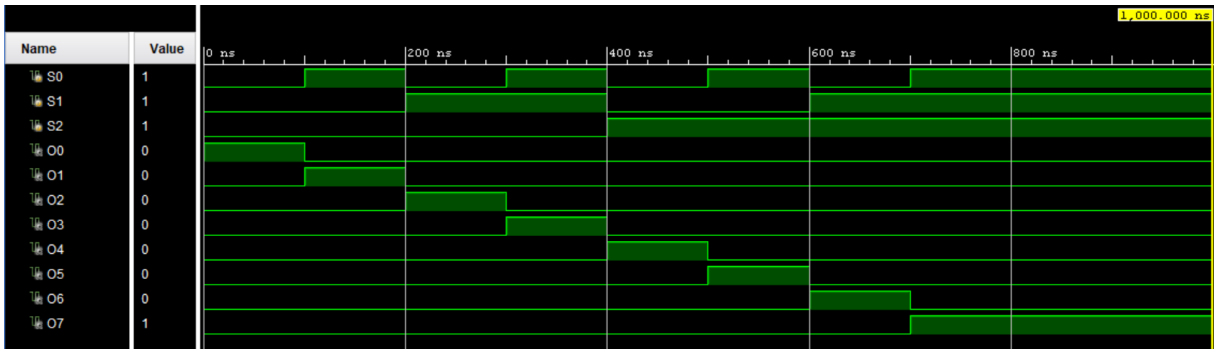


Figure 26: DEC3x8 Gate Testbench

3.2. Part 2

```
module part2(input A, B, C, D, output Z);
    wire NA, ND, AG1, AG2, AG3, OG1;
    not_gate na(A, NA); not_gate nd(D, ND);
    and_gate3 a0(NA, B, C, AG1);
    and_gate3 a1(A, C, D, AG2);
    and_gate a2(NA, ND, AG3);
    or_gate3 a3(AG1, AG2, AG3, Z);
endmodule
```

Figure 27: part2 Code

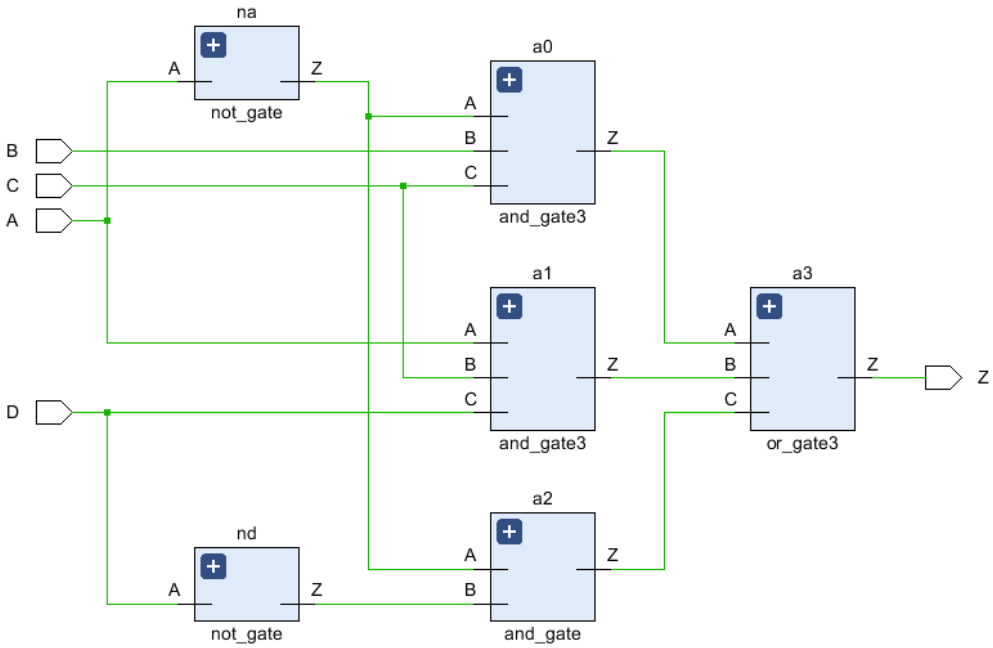


Figure 28: part2 Schematic

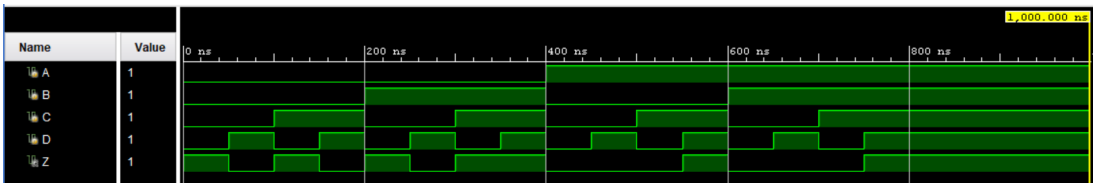


Figure 29: part2 Test Result

3.3. Part 3

```
module part3(input A, B, C, D, output Z);
    wire NA, NB, ND, nabc, nabc1, nabc2, nacd, nacd1, nacd2, nbd, nbd1, nbd2;
    nand_gate na(A, A, NA);
    nand_gate nb(B, B, NB);
    nand_gate nd(D, D, ND);

    nand_gate3 ngabc(NA, B, C, nabc);
    nand_gate ngabc1(nabc, nabc, nabc1);
    nand_gate ngabc2(nabc1, nabc1, nabc2);

    nand_gate3 ngacd(A, C, D, nacd);
    nand_gate ngacd1(nacd, nacd, nacd1);
    nand_gate ngacd2(nacd1, nacd1 ,nacd2);

    nand_gate ngbd(NB, ND, nbd);
    nand_gate ngbd1(nbd, nbd, nbd1);
    nand_gate ngbd2(nbd1, nbd1, nbd2);

    nand_gate3 nf(nabc2, nacd2, nbd2, Z);
endmodule
```

Figure 30: part3 Code

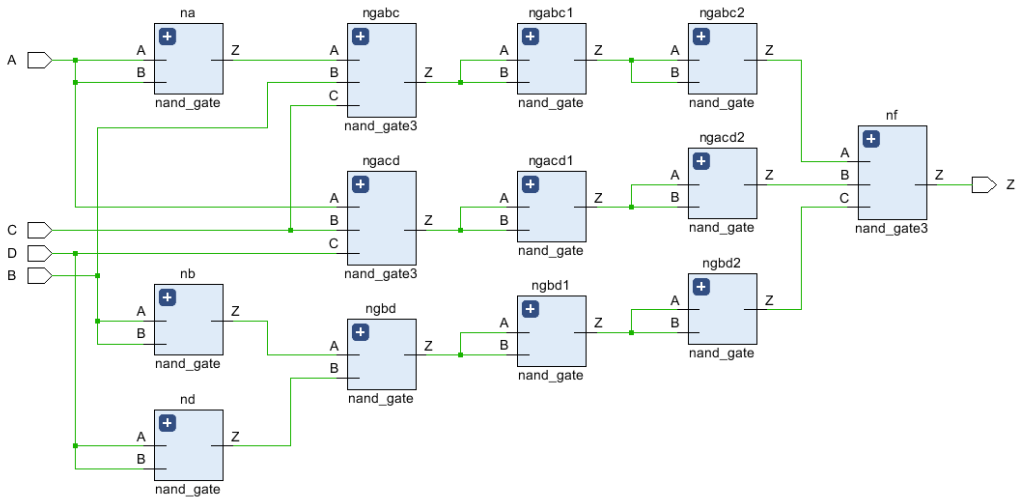


Figure 31: part3 Schematic

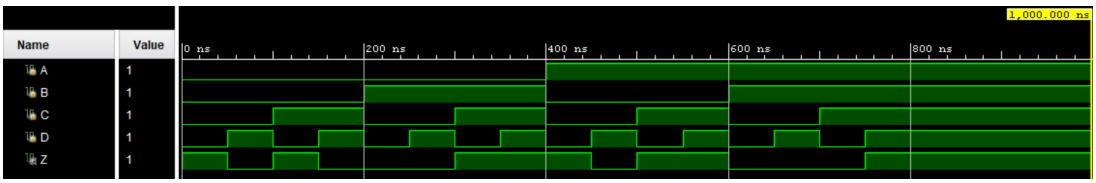


Figure 32: part3 Test Result

3.4. Part 4

```
module part4(input A, B, C, D, output Output);
    wire ND;
    not_gate n(D, ND);
    mux_8x1 mux(ND, ND, 0, 1, ND, 1, 0, D, C, B, A, Output);
endmodule
```

Figure 33: part4 Code

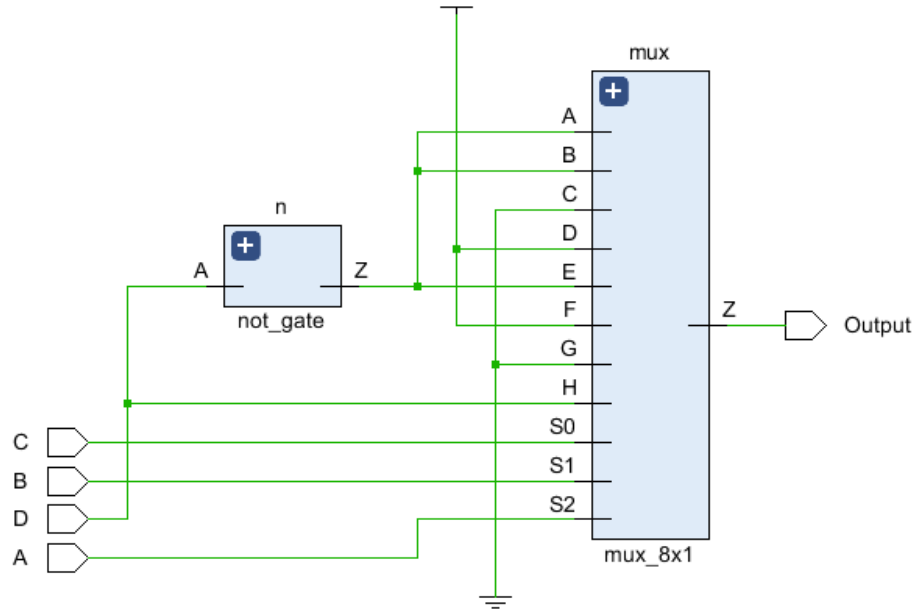


Figure 34: part4 Schematic

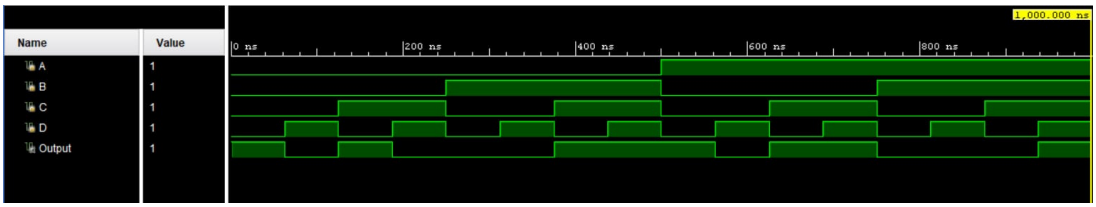


Figure 35: part4 Test Result

3.5. Part 5

3.5.a. Part5 - 1

```
module part5_1(input A, B, C, output Z);
    wire d0, d1, d2, d3, d4, d5, d6, d7;
    dec_3x8 dec(C, B, A, d0 , d1, d2, d3, d4, d5, d6, d7);
    or_gate og1(d3, d5, Z);
endmodule
```

Figure 36: part5-1 Code

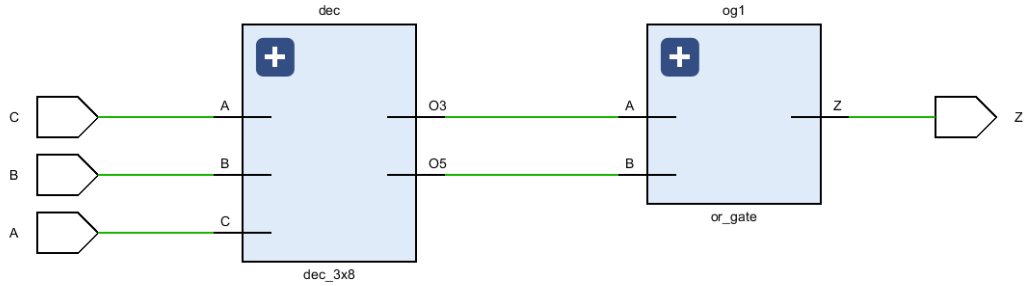


Figure 37: part5-1 Schematic

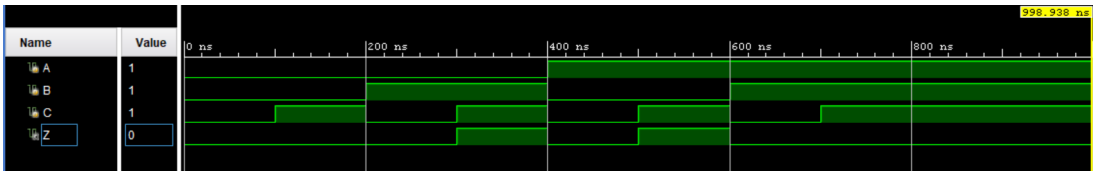


Figure 38: part5-1 Test Result

3.5.b. Part5 - 2

```
module part5_2(input A, B, C, output Z);
    wire d0, d1, d2, d3, d4, d5, d6, d7;
    dec_3x8 dec(C, B, A, d0 , d1, d2, d3, d4, d5, d6, d7);
    or_gate og1(d6 , d7, Z);
endmodule
```

Figure 39: part5-2 Code

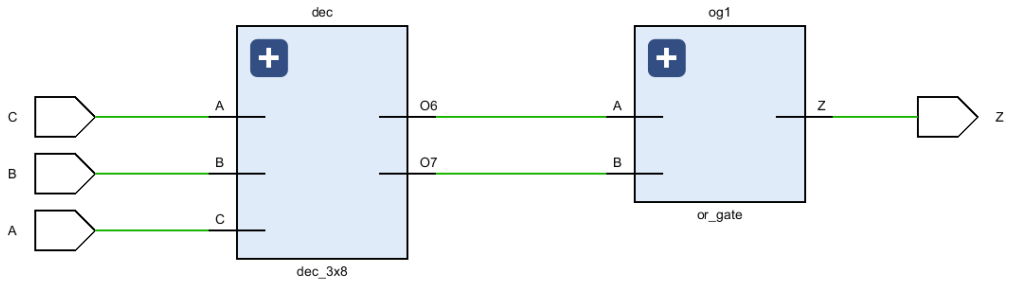


Figure 40: part5-2 Schematic

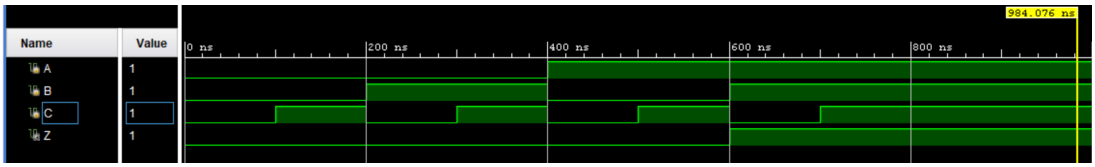


Figure 41: part5-2 Test Result

3.6. Part 6

```
module half_adder1(input A, B, output S, C);
    xor_gate xg1(A, B, S);
    and_gate ag1(A, B, C);
endmodule
```

Figure 42: part6 Code

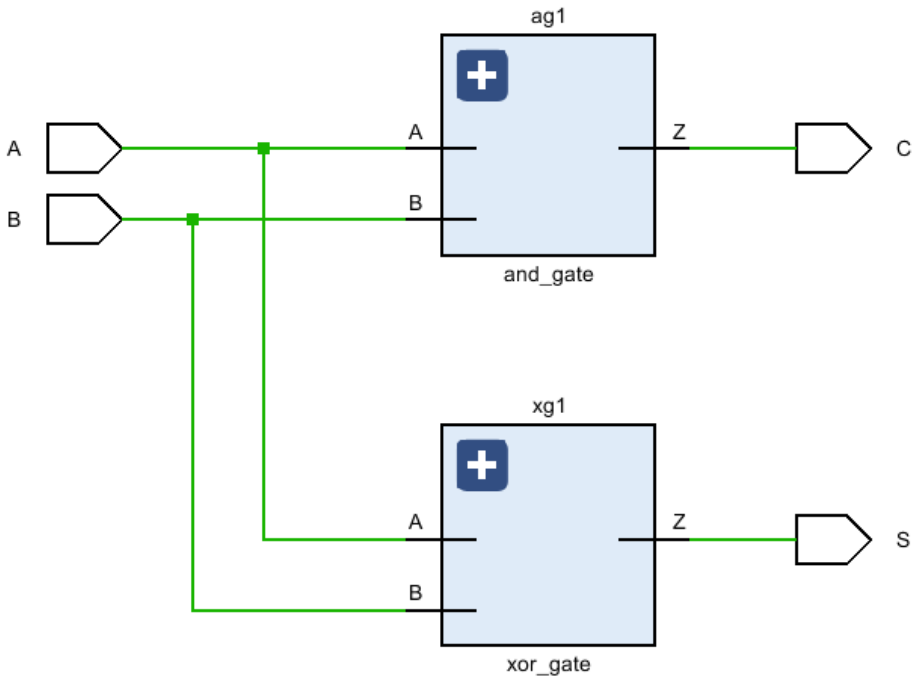


Figure 43: part6 Schematic

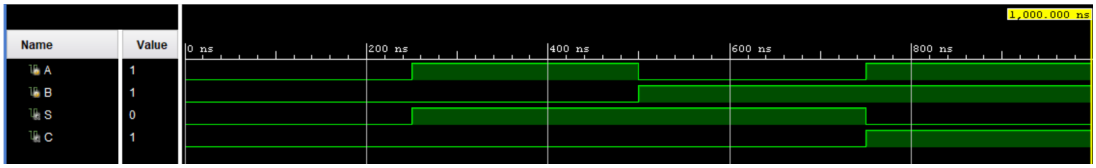


Figure 44: part6 Test Result

3.7. Part 7

```
module full_adder1(input A, B, Cin, output S, Cout);
    wire S0, C0, C1;
    half_adder1 ha(A, B, S0, C0);
    half_adder1 ha1(S0, Cin, S, C1);
    or_gate og(C0, C1, Cout);
endmodule
```

Figure 45: part7 Code

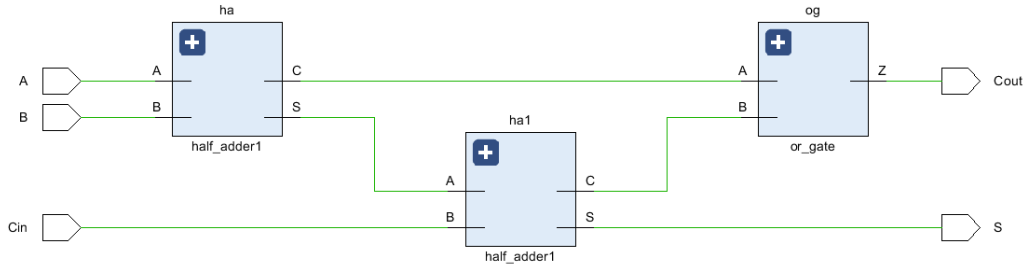


Figure 46: part7 Schematic

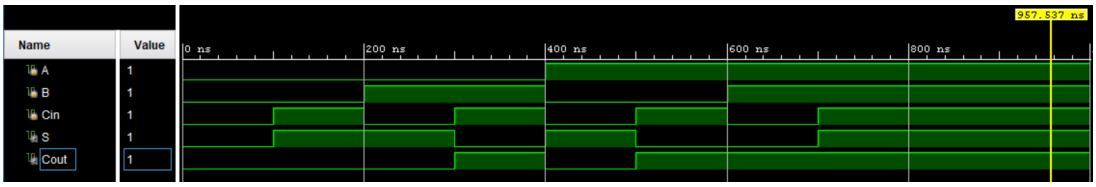


Figure 47: part7 Test Result

3.8. Part 8

```
module full_adder4(input [3:0] A, [3:0] B, Cin, output [3:0] S, Cout);
    wire C0, C1, C2;
    full_adder1 fa0(A[0], B[0], Cin, S[0], C0);
    full_adder1 fa1(A[1], B[1], C0, S[1], C1);
    full_adder1 fa2(A[2], B[2], C1, S[2], C2);
    full_adder1 fa3(A[3], B[3], C2, S[3], Cout);
endmodule
```

Figure 48: part8 Code

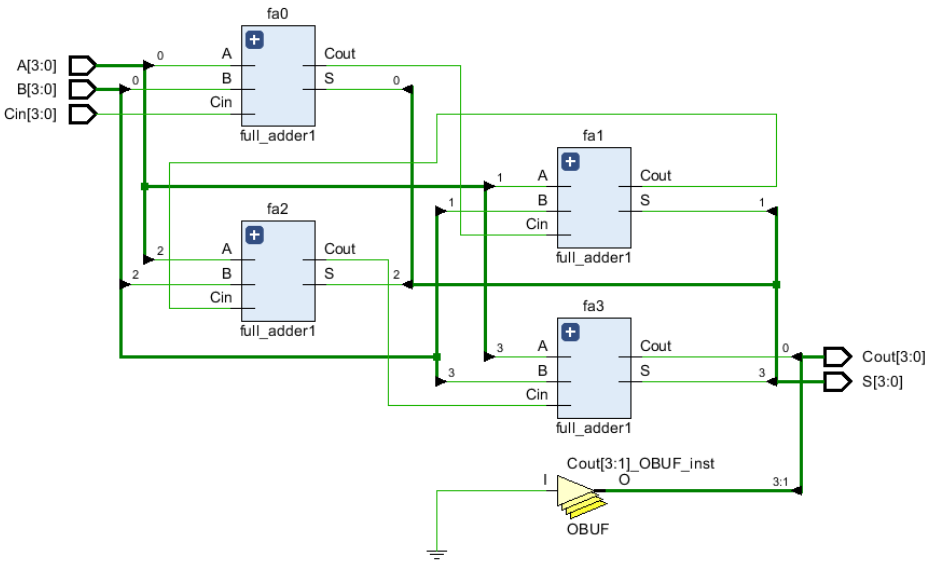


Figure 49: part8 Schematic

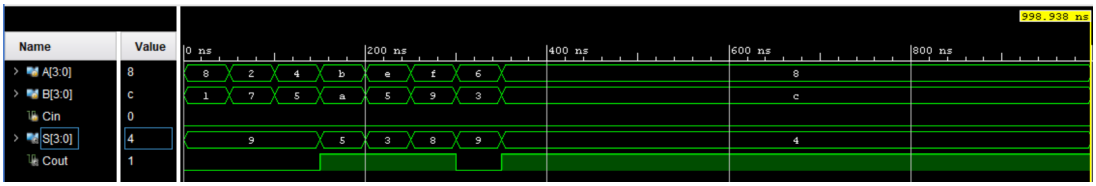


Figure 50: part8 Test Result

3.9. Part 9

```
module full_adder8(input [7:0] A, [7:0] B, input Cin, output [7:0] S, Co);
    wire [7:0] Cout;
    full_adder1 fa0(A[0], B[0], Cin, S[0], Cout[0]);
    full_adder1 fa1(A[1], B[1], Cout[0],S[1], Cout[1]);
    full_adder1 fa2(A[2], B[2], Cout[1], S[2],Cout[2]);
    full_adder1 fa3(A[3], B[3], Cout[2], S[3], Cout[3]);
    full_adder1 fa4(A[4], B[4], Cout[3], S[4], Cout[4]);
    full_adder1 fa5(A[5], B[5], Cout[4], S[5], Cout[5]);
    full_adder1 fa6(A[6], B[6], Cout[5], S[6], Cout[6]);
    full_adder1 fa7(A[7], B[7], Cout[6], S[7], Co);
endmodule
```

Figure 51: part9 Code

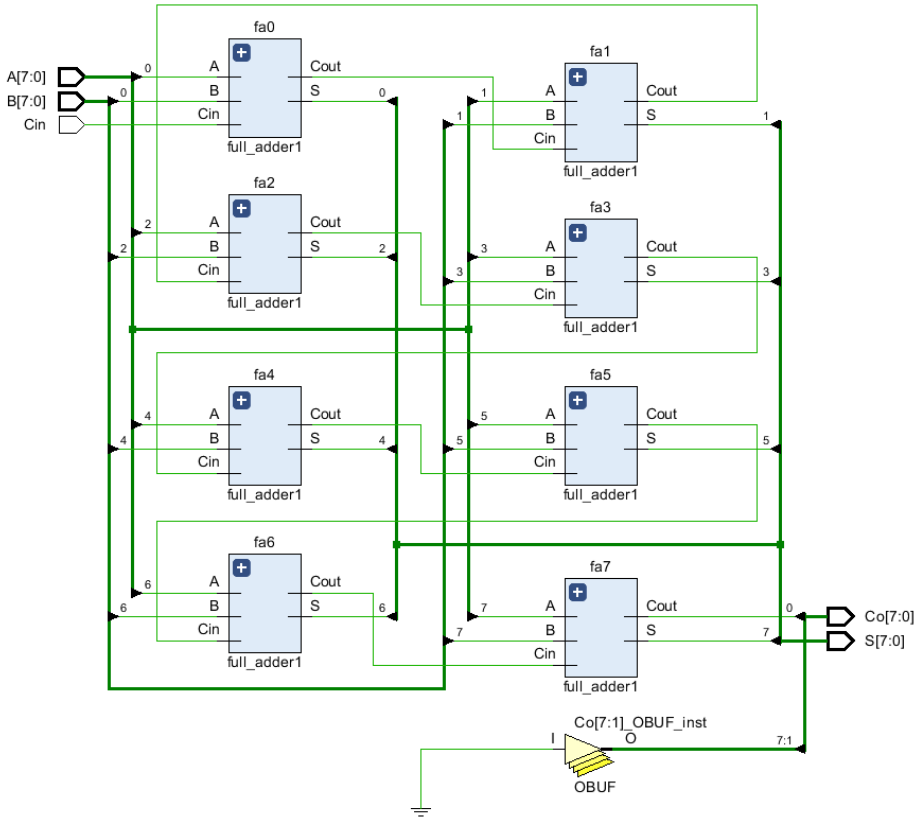


Figure 52: part9 Schematic

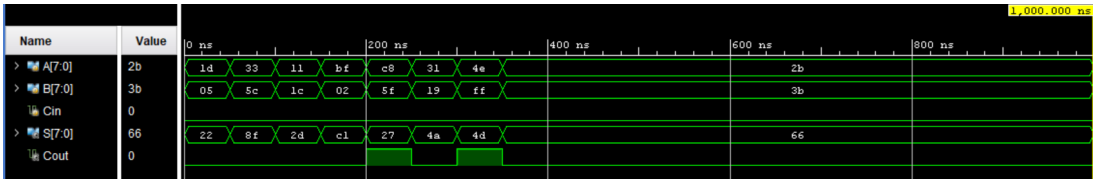


Figure 53: part9 Test Result

3.10. Part 10

```
module full_adder16(input [15:0] A, [15:0] B, M, output [15:0] S, Co);
    wire C1, C2;
    wire[15:0] Bo;
    xor_gate xg0(B[0], M, Bo[0]);
    xor_gate xg1(B[1], M, Bo[1]);
    xor_gate xg2(B[2], M, Bo[2]);
    xor_gate xg3(B[3], M, Bo[3]);
    xor_gate xg4(B[4], M, Bo[4]);
    xor_gate xg5(B[5], M, Bo[5]);
    xor_gate xg6(B[6], M, Bo[6]);
    xor_gate xg7(B[7], M, Bo[7]);
    xor_gate xg8(B[8], M, Bo[8]);
    xor_gate xg9(B[9], M, Bo[9]);
    xor_gate xg10(B[10], M, Bo[10]);
    xor_gate xg11(B[11], M, Bo[11]);
    xor_gate xg12(B[12], M, Bo[12]);
    xor_gate xg13(B[13], M, Bo[13]);
    xor_gate xg14(B[14], M, Bo[14]);
    xor_gate xg15(B[15], M, Bo[15]);
    full_adder8 fa0(A[7:0], Bo[7:0], M, S[7:0], C1);
    full_adder8 fa1(A[15:8], Bo[15:8], C1, S[15:8], Co);
endmodule
```

Figure 54: part10 Code

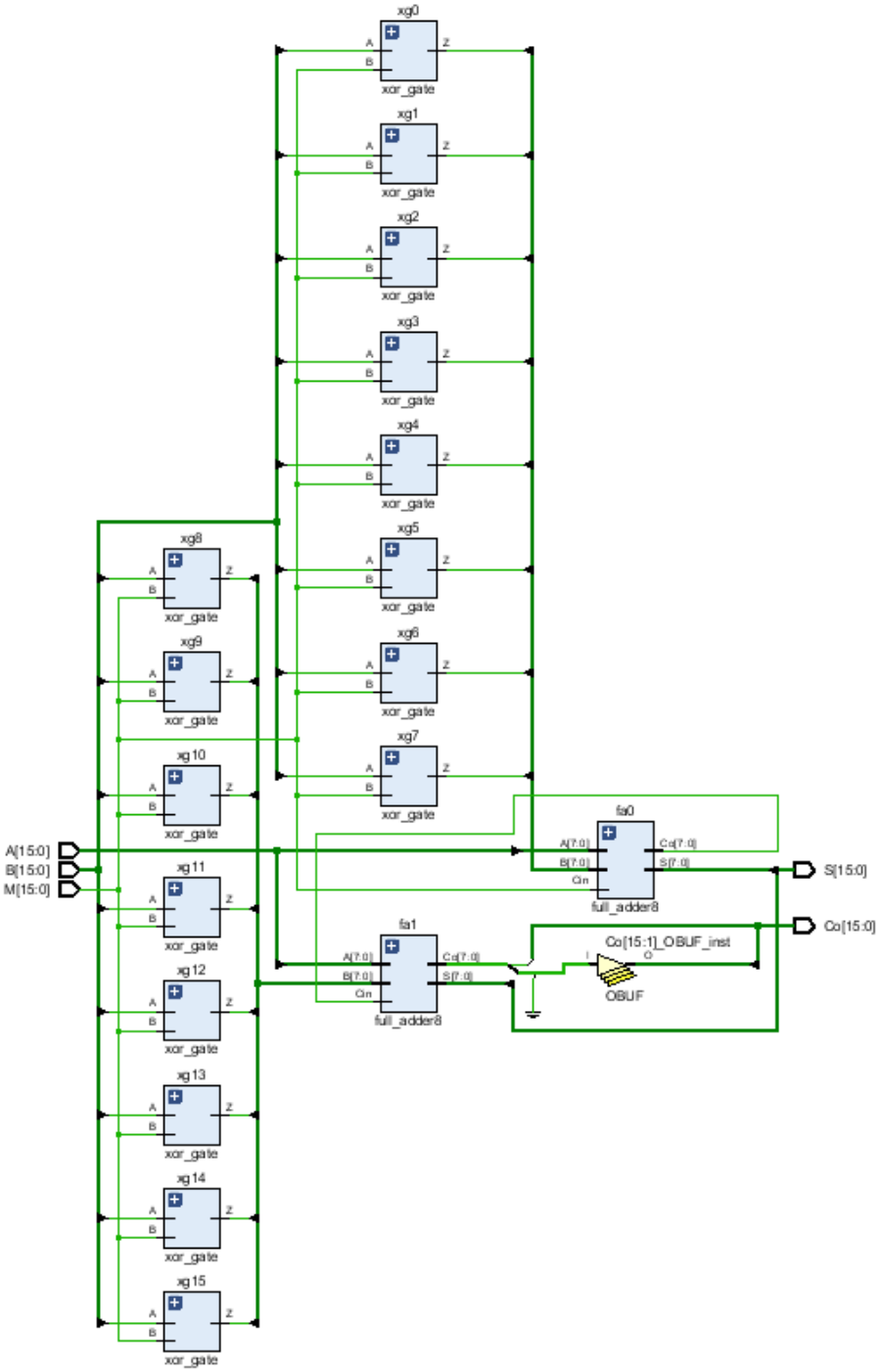


Figure 55: part10 Schematic

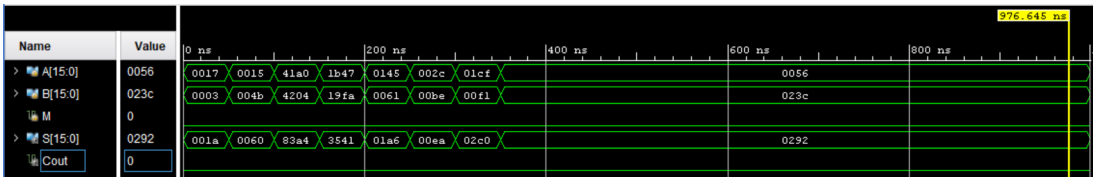


Figure 56: part10 Test Result

3.11. Part 11

```
module part11(input [15:0] A, [15:0] B, output [15:0]S, Cout);
    wire [15:0] A2X; wire c;
    full_adder16 add(A,A,0,A2X,c);
    full_adder16 sub(B, A2X, 1, S, Cout);
endmodule
```

Figure 57: part11 Code

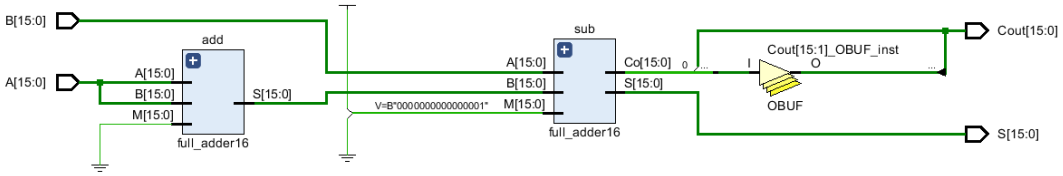


Figure 58: part11 Schematic

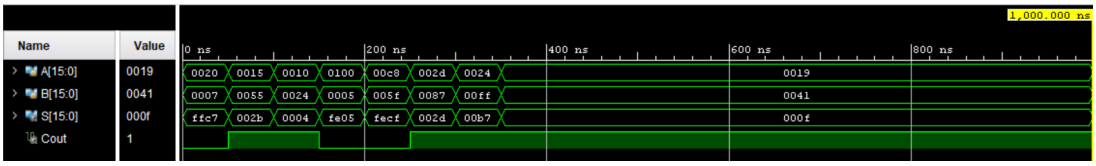


Figure 59: part11 Test Result

4. RESULTS

We used Verilog and the Vivado simulator to implement each circuit. The test outcomes confirmed our expectations based on our computations on paper. On the relevant part, all test results are supplied.

5. DISCUSSION

With this homework assignment, we learned how to use and implement basic verilog gates. Using the circuits we previously constructed on the previous parts, we also implemented several arithmetic circuits. Our outcomes were largely in line with what we had anticipated.

6. CONCLUSION

We have had a great opportunity to learn how to utilize Vivado and the Verilog programming language with this homework. We built some straightforward circuits using gates as well as some rather more challenging arithmetic circuits using the circuits we had already built. To sum up, this was a really lengthy and challenging homework assignment for us, but we were able to successfully complete almost all of the portions.