ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY HOMEWORK REPORT

HOMEWORK NO	:	1
HOMEWORK DATE	:	10.03.2023
LAB SESSION	:	FRIDAY - 14.00
GROUP NO	:	G3

GROUP MEMBERS:

150020054 : ASLI YEL

150190028 : SEVİM EFTAL AKŞEHİRLİ

SPRING 2023

Contents

1 PRELIMINARY

1.1

The truth table for the function $F_1(a, b, c, d) = \cup_1(0, 2, 6, 7, 8, 10, 11, 15) + \cup_\phi(4)$ is given below and required operations will take place accordingly.

#	a	b	c	d	F1(a,b,c,d)
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	ϕ
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

Table 1: Truth table for $F_1(a, b, c, d) = \cup_1(0, 2, 6, 7, 8, 10, 11, 15) + \cup_\phi(4)$

1.1.a Karnaugh Map Method

In order to find the prime implicants of the function F_1, we can use the Karnaugh Map Method.

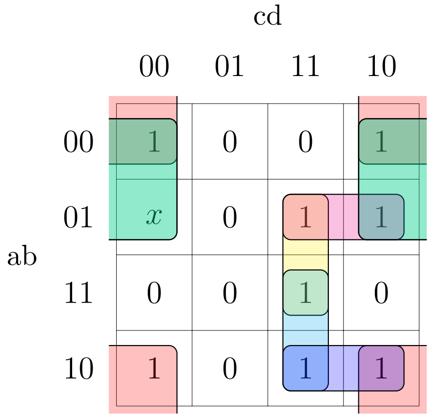


Figure 1: Karnaugh Map for F_1

All prime implicants created on the Karnaugh Map are : $\bar{a}\bar{d}$, $\bar{b}\bar{d}$, $a\bar{b}c$, $\bar{a}bc$, acd , bcd.

1.1.b Quine-McCluskey Method

Prime implicants of function F_1 can also be obtained by using Quine-McCluskey Method.

Firstly, all the minterms that generates 1 are placed in a minterm table. Don't-care term is also included in the table in order to enable the potential grouping of this term with the minterms.

Minterm	Binary Representation
m(0)	0000
m(2)	0010
m(4)	0100
m(6)	0110
m(7)	0111
m(8)	1000
m(10)	1010
m(11)	1011
m(15)	1111

Table 2: Minterms and their Binary Representations

Then, for the purpose of shortening the running time of the algorithm, this 1-generating points are clustered according to the number of 1s in the input combinations corresponding to each minterm.

Num. of 1s	Minterms	a	b	c	d	State
0	m(0)	0	0	0	0	✔
1	m(2)	0	0	1	0	✔
	m(4)	0	1	0	0	✔
	m(8)	1	0	0	0	✔
2	m(6)	0	1	1	0	✔
	m(10)	1	0	1	0	✔
3	m(7)	0	1	1	1	✔
	m(11)	1	0	1	1	✔
4	m(15)	1	1	1	1	✔

Table 3: Minterms and Corresponding Input Combinations

Then what is needed to be done is comparing the values of the all input variables for the minterms that are in neighboring clusters and grouping the minterms that differ by only a single value by pairs.

Marking the ones that are grouped with a ✔ symbol and the ones that are not grouped with a ✗ symbol during this process will indicate us the state of whether they are in the prime implicant set or not, at the end.

Num. of 1s	Size 2 Implicants	a	b	c	d	State
0	m(0), m(2)	0	0	-	0	✔
	m(0), m(4)	0	-	0	0	✔
	m(0), m(8)	-	0	0	0	✔
1	m(2), m(6)	0	-	1	0	✔
	m(2), m(10)	-	0	1	0	✔
	m(4), m(6)	0	1	-	0	✔
	m(8), m(10)	1	0	-	0	✔
2	m(6), m(7)	0	1	1	-	✗
	m(10), m(11)	1	0	1	-	✗
3	m(7), m(15)	-	1	1	1	✗
	m(11), m(15)	1	-	1	1	✗

Table 4: Size 2 Implicants and Corresponding Input Combinations

Grouping operation will continue until it is no longer possible to create any more groups.

Num. of 1s	Size 4 Implicants	a	b	c	d	State
0	m(0), m(2), m(4), m(6)	0	-	-	0	✗
	m(0), m(2), m(8), m(10)	-	0	-	0	✗

Table 5: Size 4 Implicants and Corresponding Input Combinations

Now the set of prime implicants consists of all the input combinations that are not grouped, the ones that are marked with ✗.

Therefore, all prime implicants of the given expression obtained by utilizing the Quine-McCluskey Method are : $\bar{a}\bar{d}$, $\bar{b}\bar{d}$, a$\bar{b}$c , $\bar{a}$bc , acd , bcd.

1.1.c Prime Implicant Chart

Expression of a function with the lowest cost according to a given rule of cost can be obtained with prime implicant chart.

First of all, the costs of each prime implicant are found with the given rule, the 1-generating points that are covered by each prime implicant are determined, and a symbol is assigned to each prime implicant.

Rule of Cost: 2 units of cost for each variable and 1 unit of cost for complement of a variable.

	$\bar{a}\bar{d}$	$\bar{b}\bar{d}$	a$\bar{b}$c	$\bar{a}$bc	acd	bcd
Symbol:	A	B	C	D	E	F
Cost:	6	6	7	7	6	6
Covered True Points:	0,2,6	0,2,8,10	10,11	6,7	11,15	7,15

Table 6: Costs of and True Points Covered by Prime Implicants

Now the prime implicant chart is set with this information.

	0	2	6	7	8	10	11	15	*Cost*
A	*X*	*X*	*X*						6
~~*B*~~	*X*	*X*			*X*	*X*			~~6~~
C						*X*	*X*		7
D			*X*	*X*					7
E							*X*	*X*	6
F				*X*				*X*	6

Table 7: Prime Implicant Chart - Step1

Since the column of the point 8 contains only one 'X', 8 is a distinguished point and B, the only prime implicant that covers 8, is an essential prime implicant.

Therefore, B is selected and the rows and coulmns that are covered by it are eliminated from the chart.

B($\bar{b}\bar{d}$) will be included in the lowest cost expression. $\Rightarrow$ Cost:6, Covered Points:0,2,8,10

	6	7	11	15	*Cost*
A	*X*				6
~~*C*~~			*X*		~~7~~
D	*X*	*X*			7
E			*X*	*X*	6
F		*X*		*X*	6

Table 8: Prime Implicant Chart - Step2

Since E covers C and the cost of E is less than C, C is eliminated from the chart.

C(a$\bar{b}$c) will not be included in the lowest cost expression.

	6	7	11	15	$Cost$
A	X				6
D	X	X			7
~~E~~			X	X	~~6~~
F		X		X	6

Table 9: Prime Implicant Chart - Step3

In this chart the column of the point 11 contains only one 'X', so 11 is a distinguished point. Then E is selected and the rows and coulmns that are covered by it are eliminated from the chart.

E(acd) will be included in the lowest cost expression. $\Rightarrow$ Cost:6, Covered Points:11,15

	6	7	$Cost$
A	X		6
D	X	X	7
F		X	6

Table 10: Prime Implicant Chart - Step4

Since D covers both A and F, and its cost is less than the total cost of A and F, D is selected.

D($\bar{a}$bc) will be included in the lowest cost expression. $\Rightarrow$ Cost:7, Covered Points:6,7

With the last selection all true points of the function are covered. Selected prime implicants make up the expression of the function F_1 with the lowest cost.

Selected prime implicants: B($\bar{b}\bar{d}$), D($\bar{a}$bc), E(acd)

Total Cost: $6 + 7 + 6 = 19$

F_1(a, b, c, d) = $\bar{b}\bar{d}$ + $\bar{a}$bc + acd

1.1.d Design with AND, OR and NOT Gates

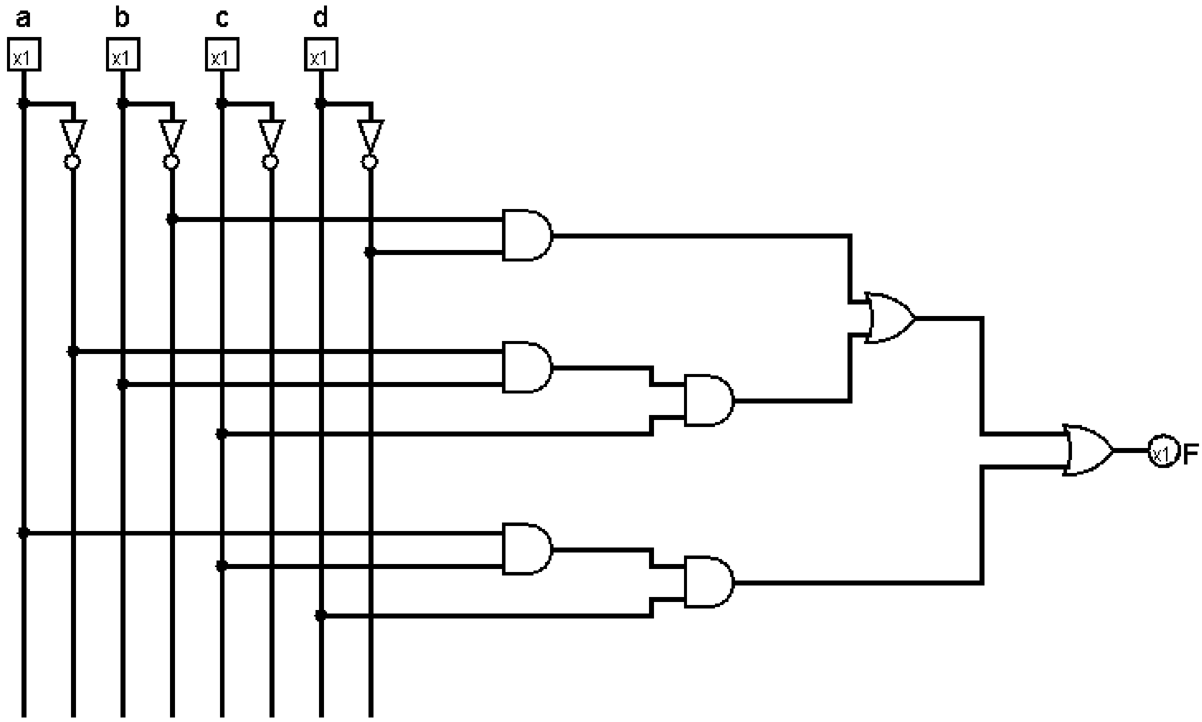


Figure 2: Design of F_1 with AND, OR and NOT Gates

1.1.e Design with only NAND Gates

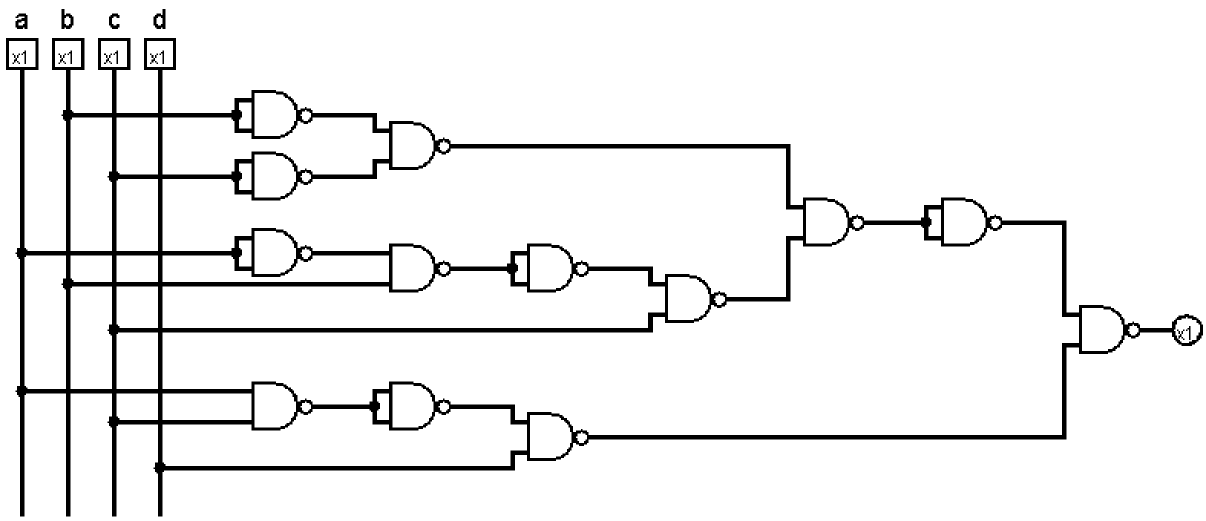


Figure 3: Design of F_1 with only NAND Gates

1.1.f Design with a single 8:1 Multiplexer, AND, OR and NOT Gates

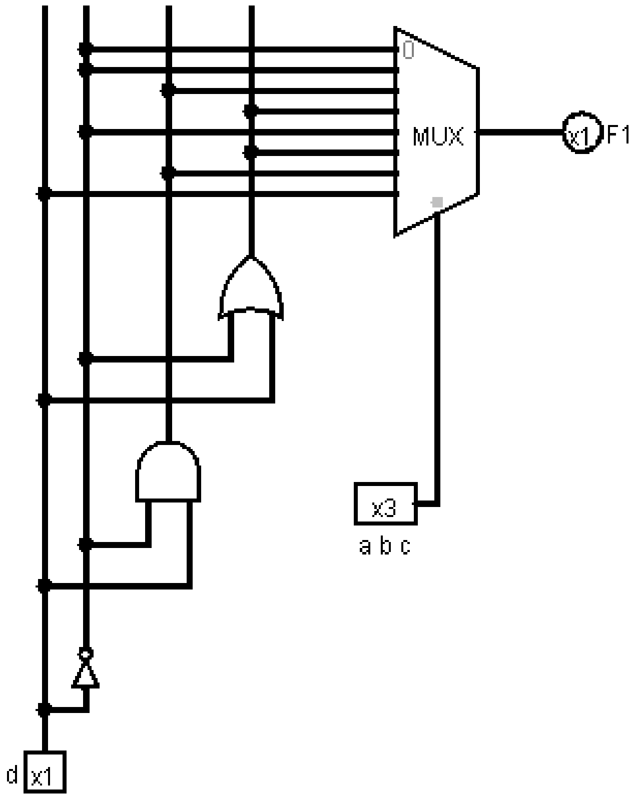


Figure 4: Design of F_1 with a single 8:1 Multiplexer, AND, OR and NOT Gates

1.2 Design of F_2 and F_3

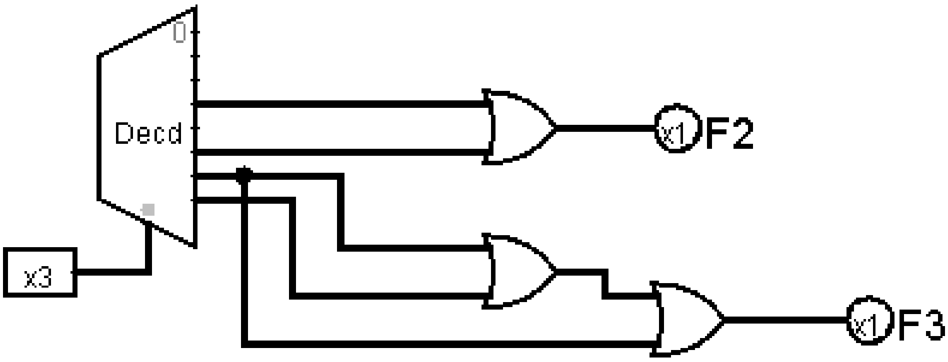


Figure 5: Design of F_2 and F_3

1.3

1.3.a Signed and Unsigned Addition of Binary Numbers

1.3.b Signed and Unsigned Subtraction of Binary Numbers

2 EXPERIMENT

2.1 Part1

Figure 6: Simulation of AND Gate

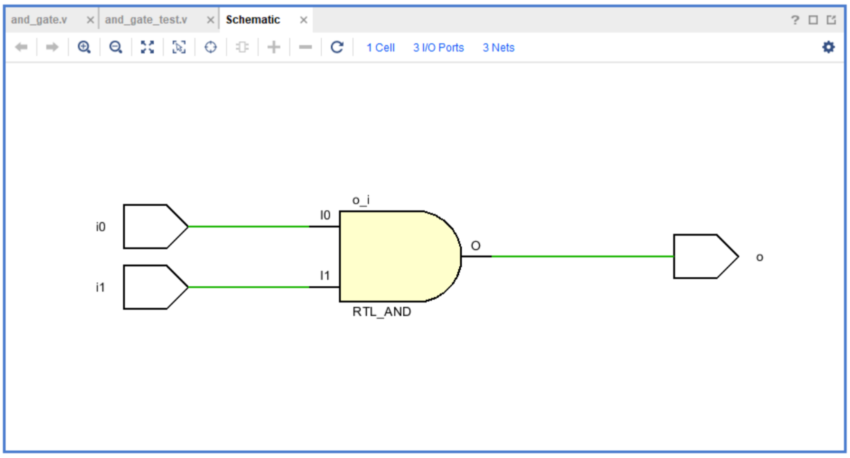


Figure 7: RTL Schematic of AND Gate

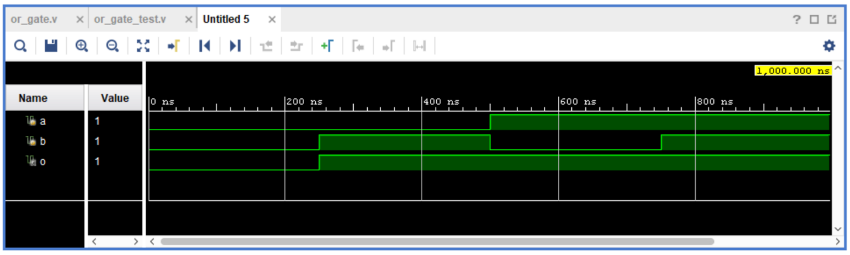


Figure 8: Simulation of OR Gate

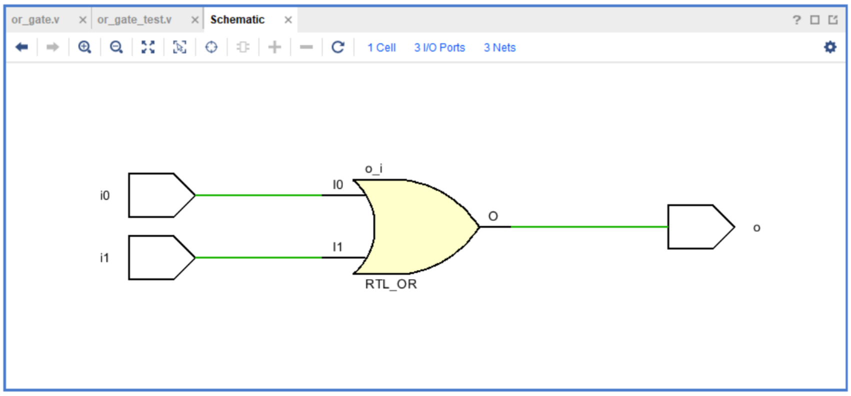


Figure 9: RTL Schematic of OR Gate

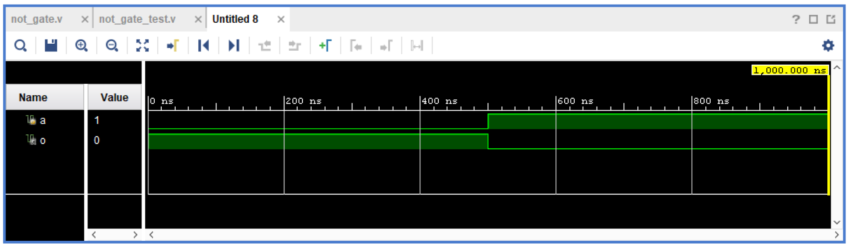


Figure 10: Simulation of NOT Gate

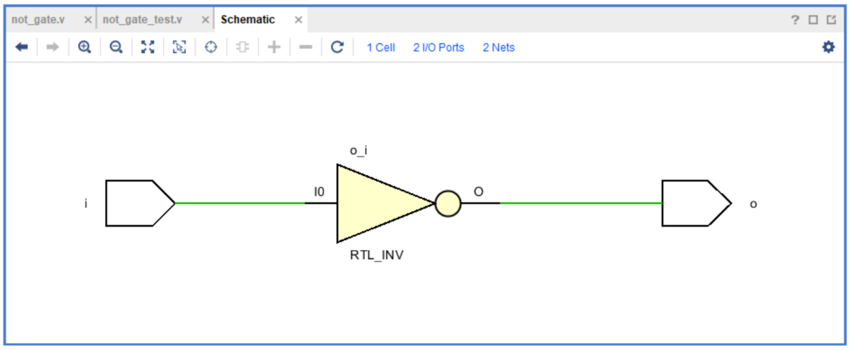


Figure 11: RTL Schematic of NOT Gate

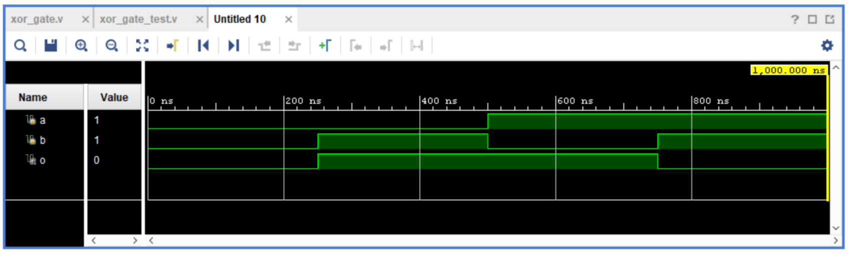


Figure 12: Simulation of XOR Gate

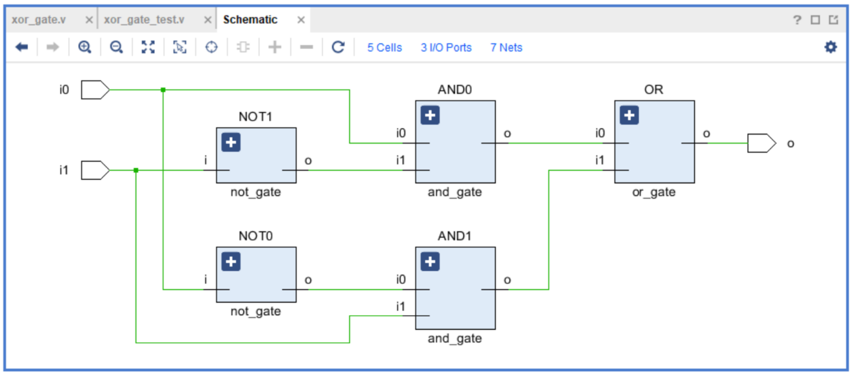


Figure 13: RTL Schematic of XOR Gate

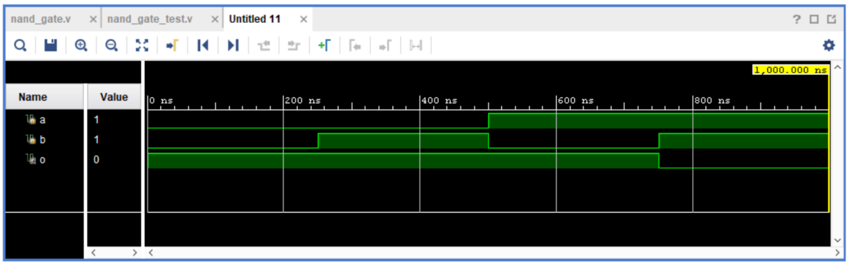


Figure 14: Simulation of NAND Gate

2.2 Part2

2.3 Part3

2.4 Part4

2.5 Part5

2.6 Part6

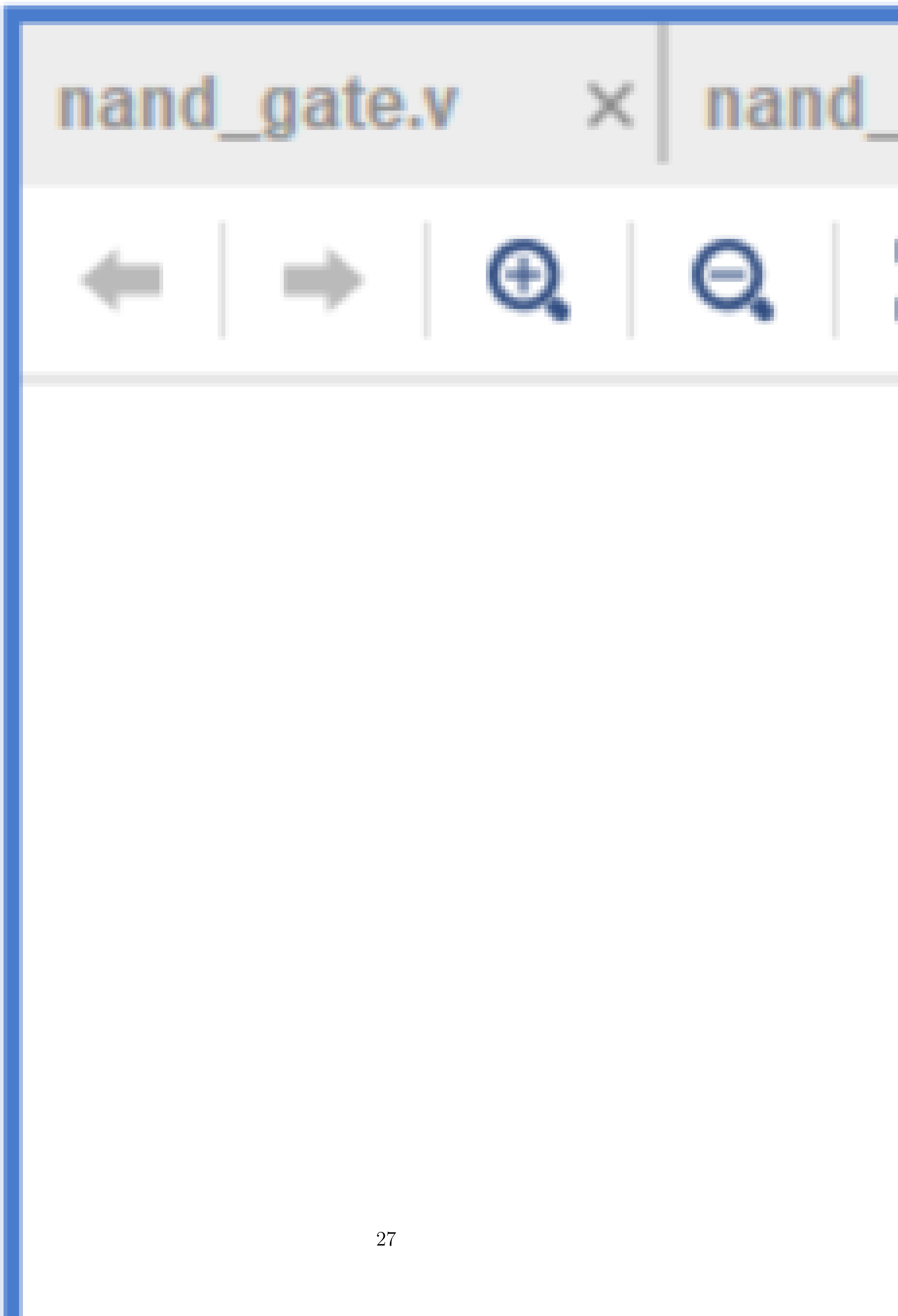
2.7 Part7

2.8 Part8

2.9 Part9

2.10 Part10

2.11 Part11


nand_gate.v

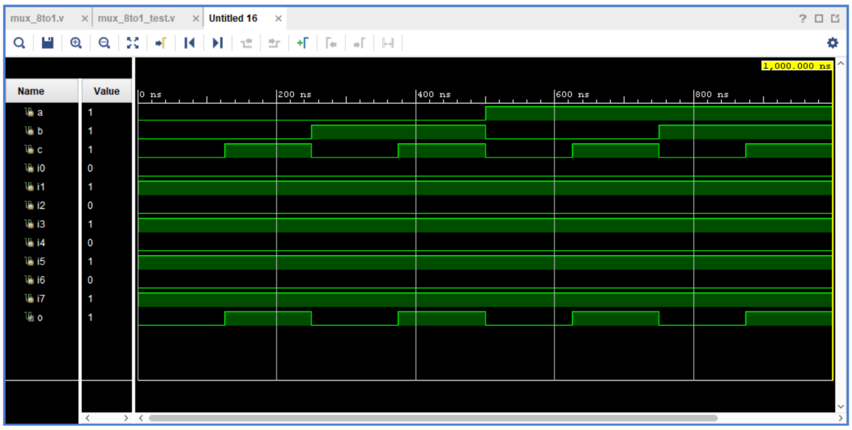


Figure 16: Simulation of 8:1 Multiplexer

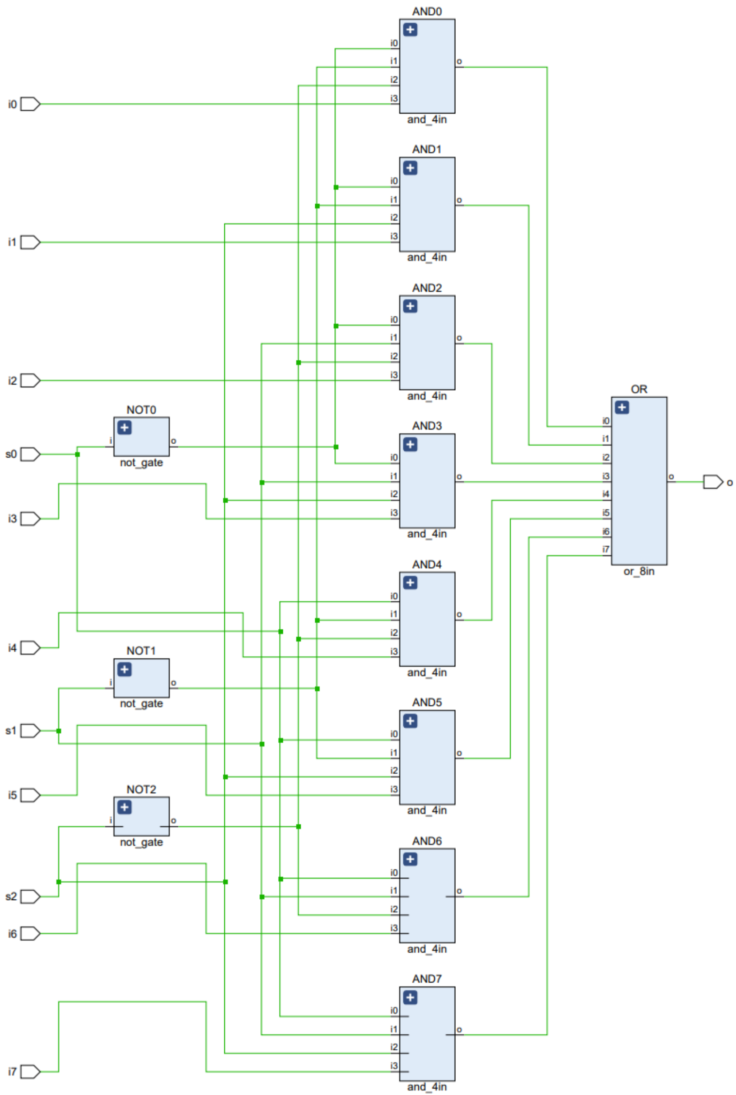


Figure 17: RTL Schematic of 8:1 MULTIPLEXER

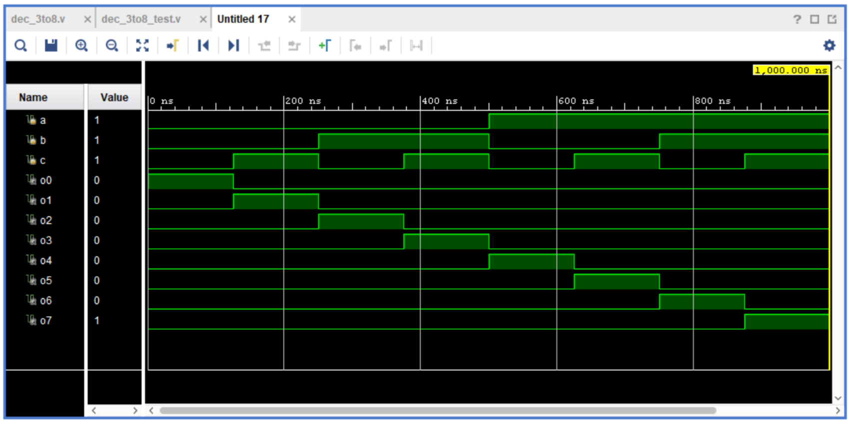


Figure 18: Simulation of 3:8 DECODER

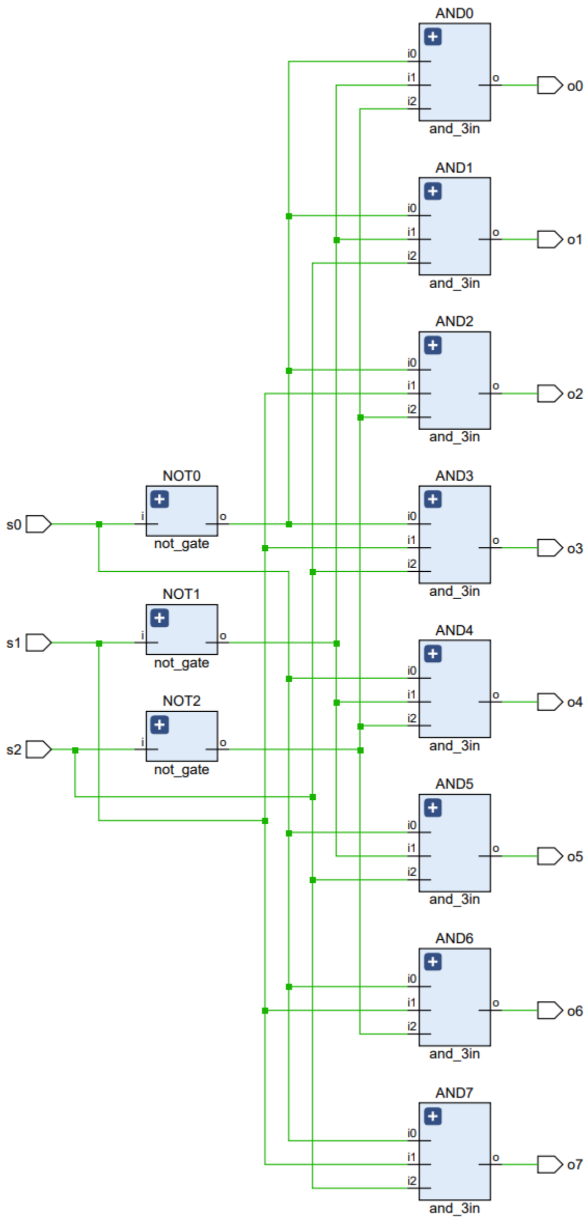


Figure 19: RTL Schematic of 3:8 DECODER

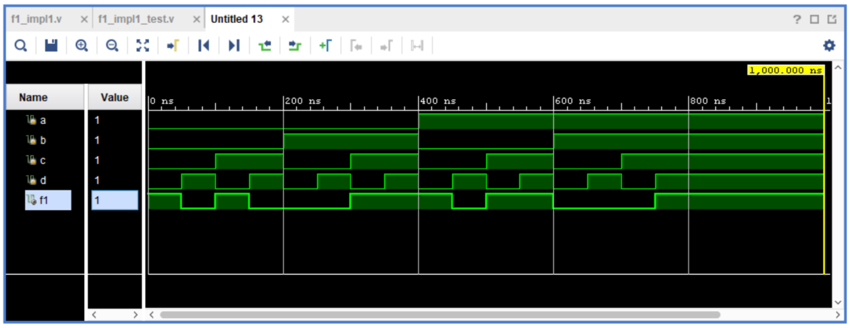


Figure 20: Simulation of F_1 Implemented with AND, OR and NOT Gates

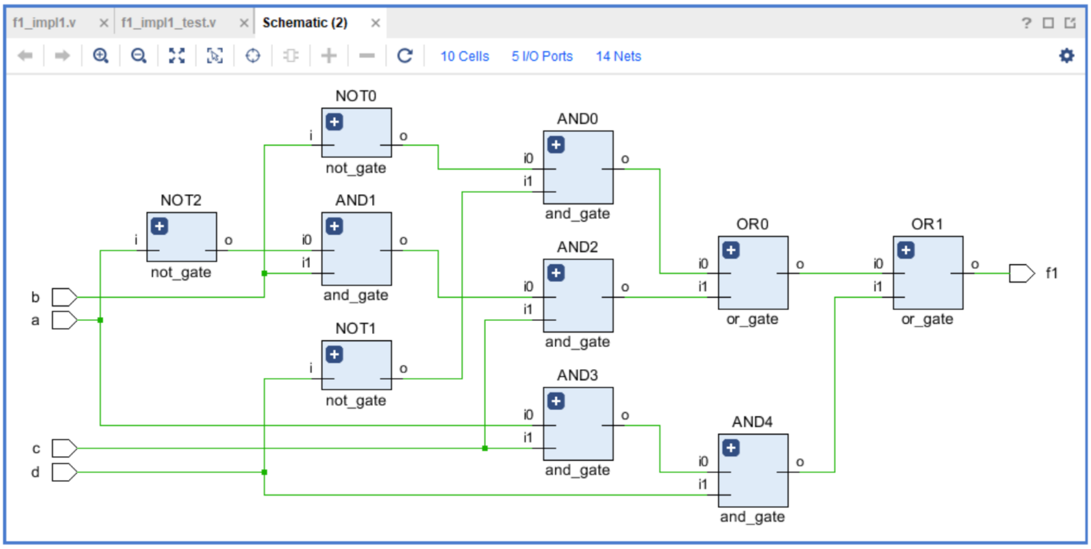


Figure 21: RTL Schematic of F_1 Implemented with AND, OR and NOT Gates

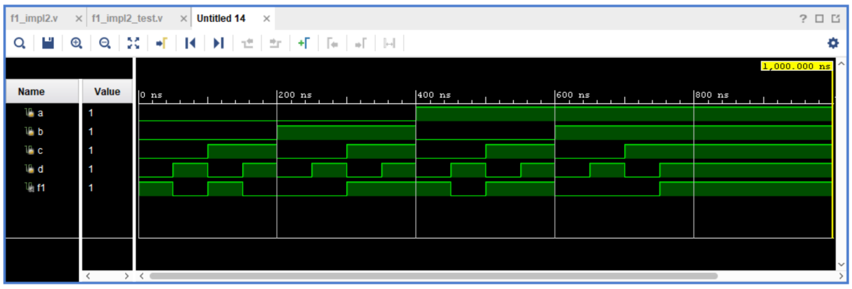


Figure 22: Simulation of F_1 Implemented with only NAND Gates

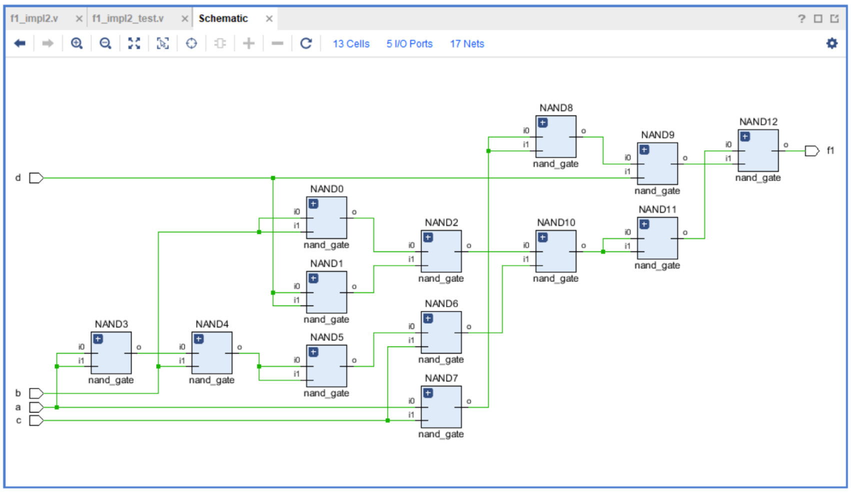


Figure 23: RTL Schematic of F_1 Implemented with only NAND Gates

Figure 24: Simulation of F_1 Implemented with 8:1 MUX, AND, OR and NOT Gates

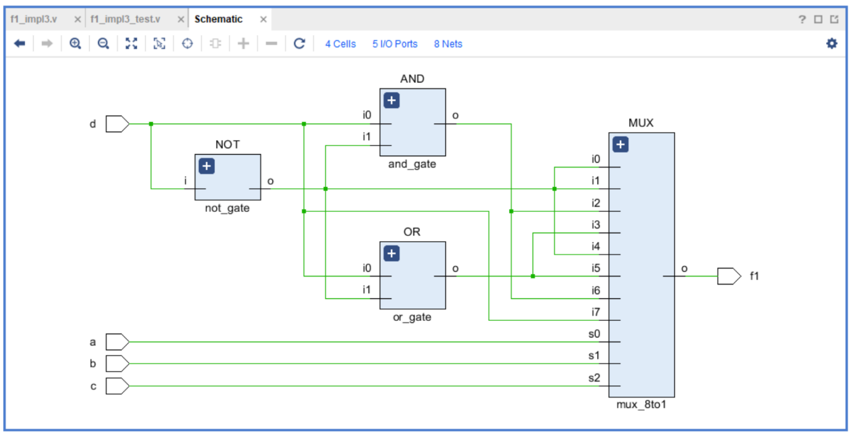


Figure 25: RTL Schematic of F_1 Implemented with 8:1 MUX, AND, OR and NOT Gates

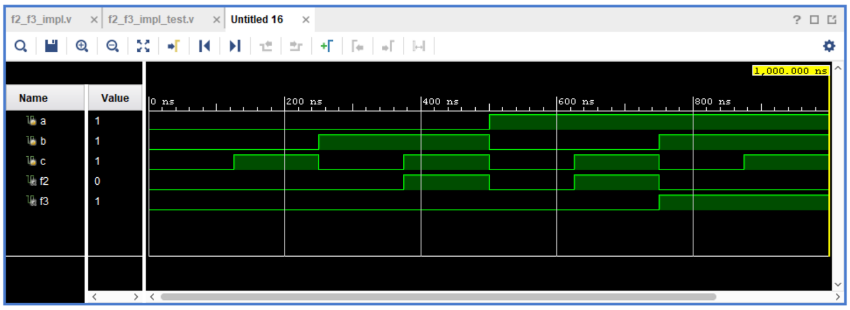


Figure 26: Simulation of F_2 and F_3

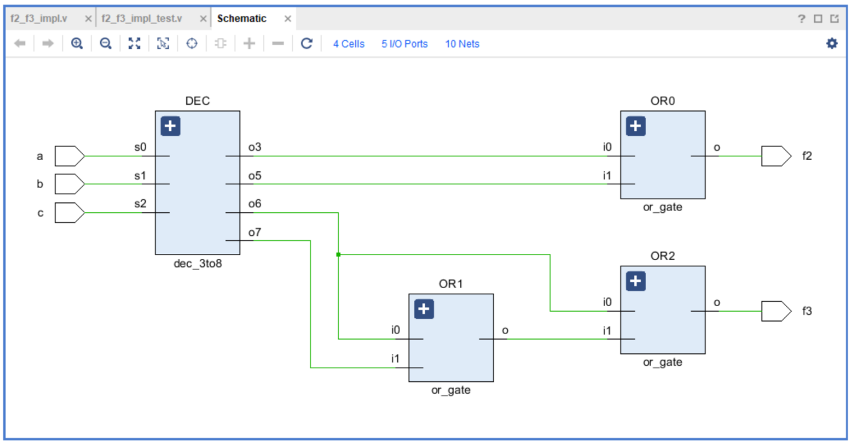


Figure 27: RTL Schematic of F_2 and F_3

Figure 28: Simulation of 1-BIT HALF ADDER

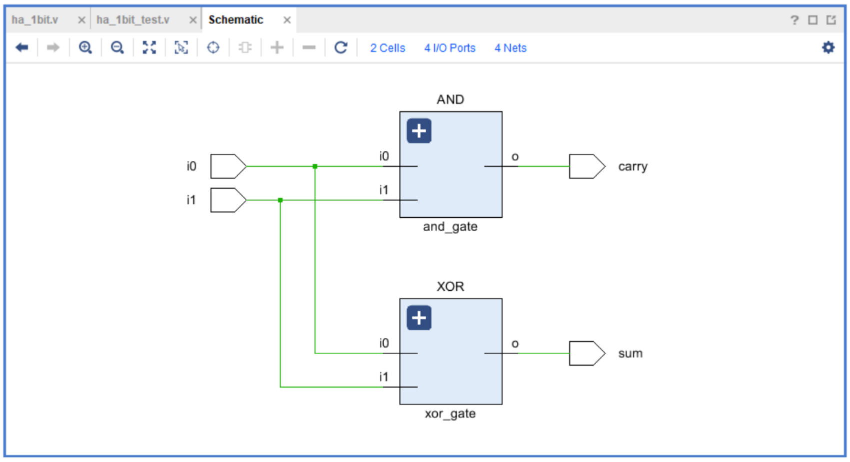


Figure 29: RTL Schematic of 1-BIT HALF ADDER

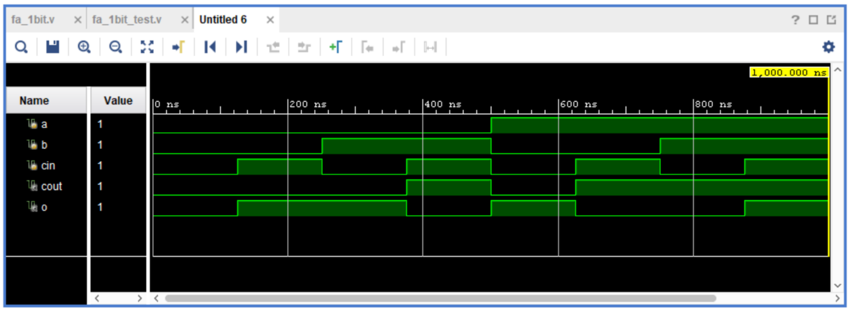


Figure 30: Simulation of 1-BIT FULL ADDER

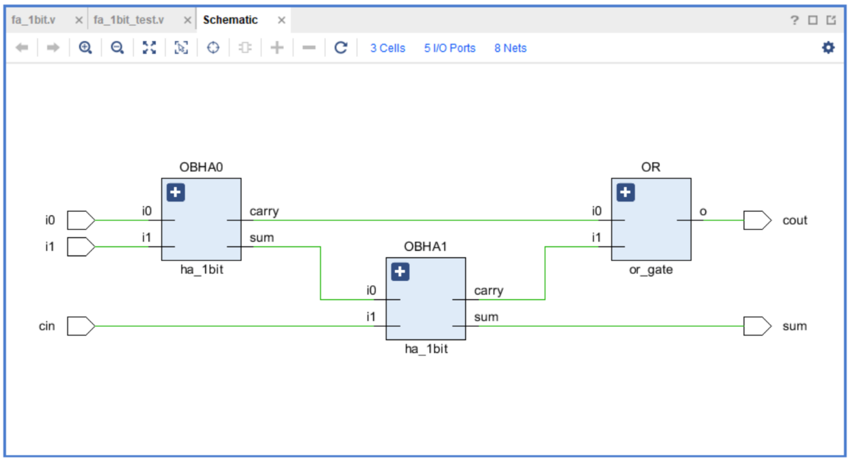


Figure 31: RTL Schematic of 1-BIT FULL ADDER

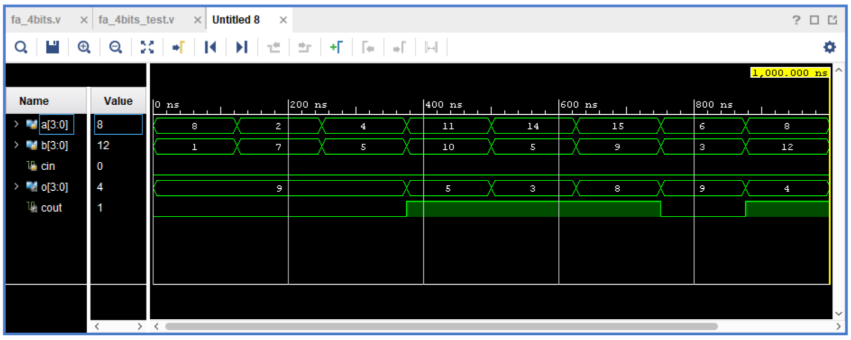


Figure 32: Simulation of 4-BIT FULL ADDER

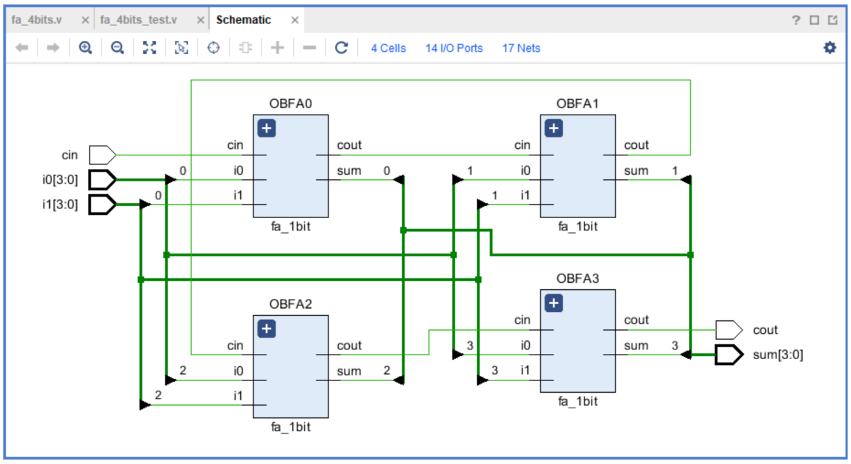


Figure 33: RTL Schematic of 4-BIT FULL ADDER

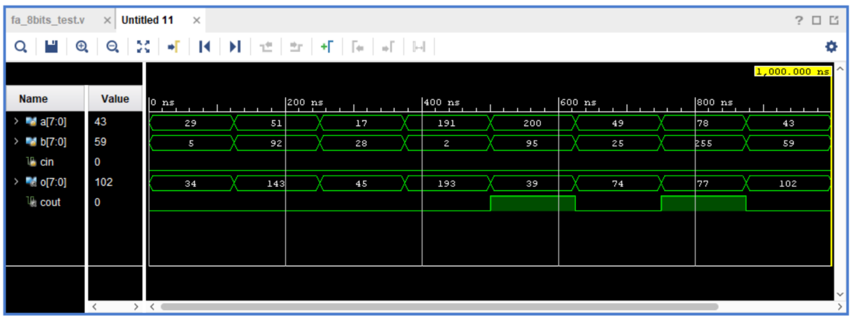


Figure 34: Simulation of 8-BIT FULL ADDER

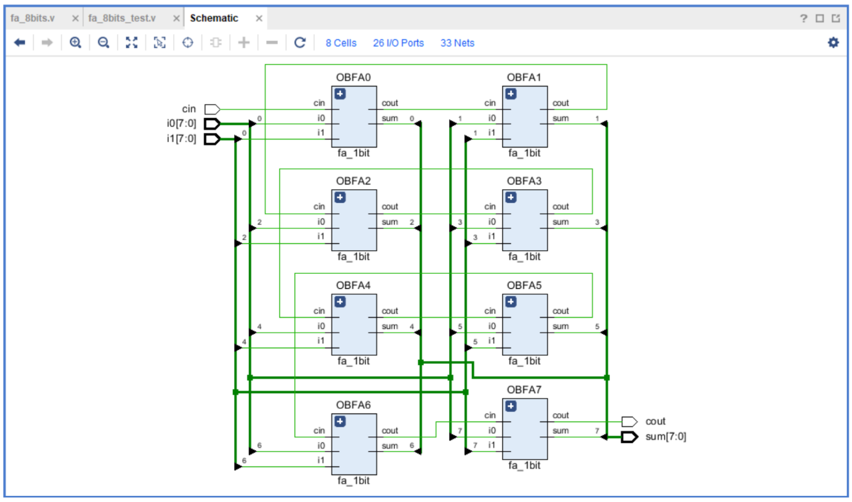


Figure 35: RTL Schematic of 8-BIT FULL ADDER

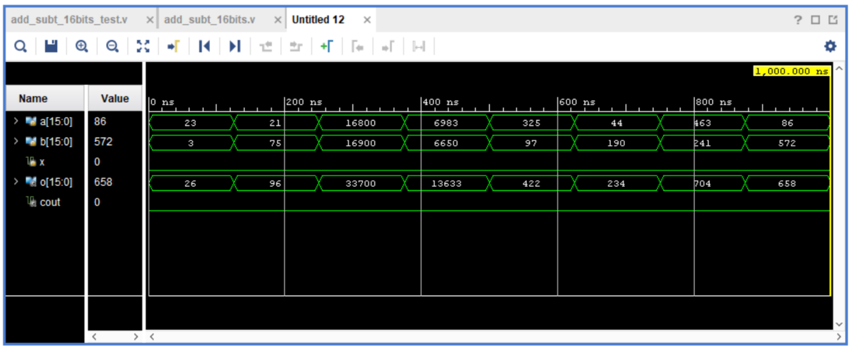


Figure 36: Simulation of 16-BIT FULL ADDER-SUBTRACTER

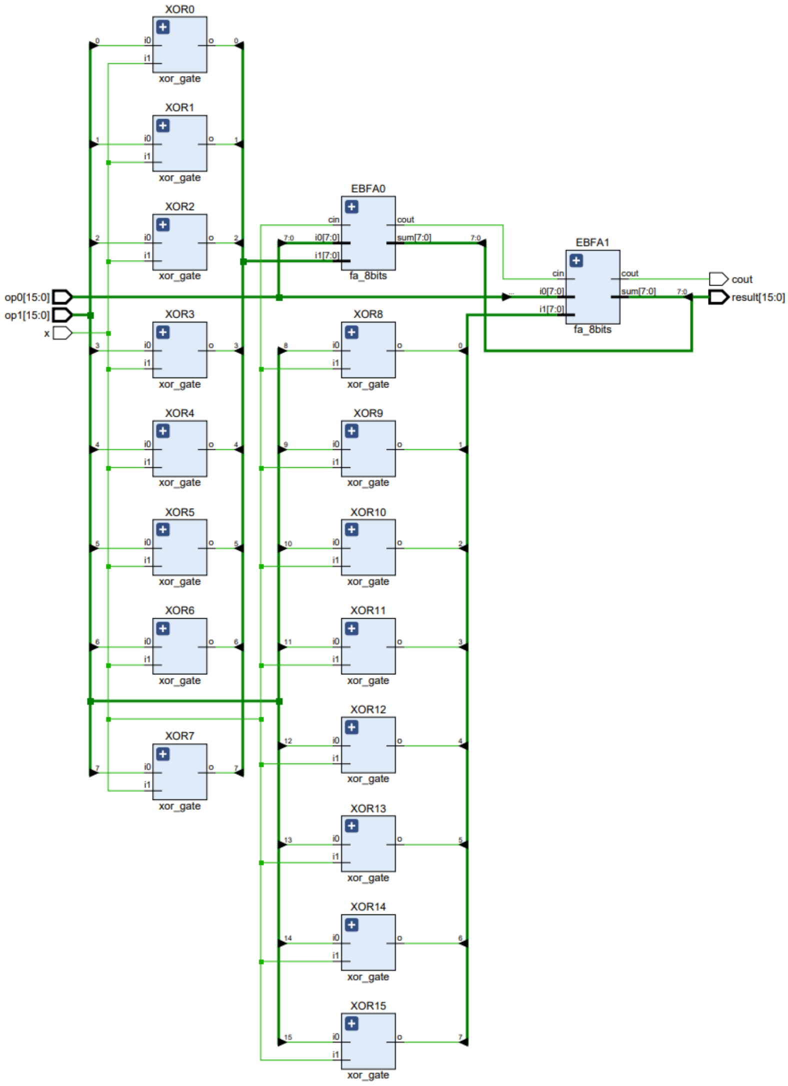


Figure 37: RTL Schematic of 16-BIT ADDER-SUBTRACTER

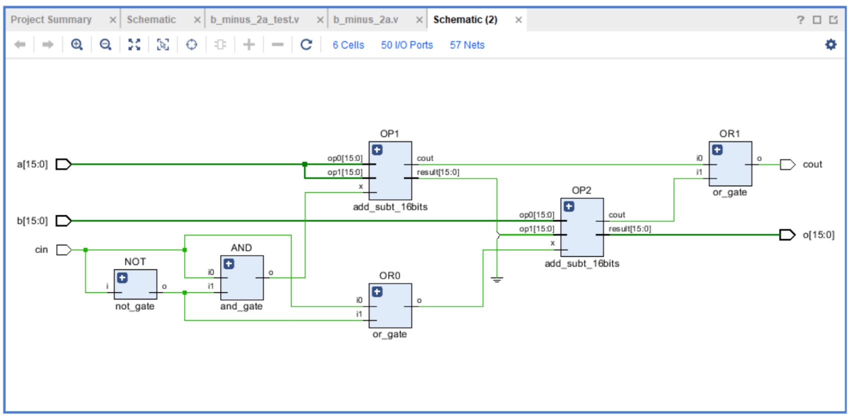


Figure 38: RTL Schematic of the Function B-2A