Digital Circuits Theory - Laboratory						
Academic year	Laboratory exercises on	Mode of studies	Field of studies	Supervisor	Group	Section
2020/2021	Wednesday 15:30 – 17:00	SSI	Informatics	US	1	3

Report from Exercise No 12

Performed on: 04.11.2020

Exercise Topic: Computer Aided Design for Circuit Development

Performed by:

Dawid Grobert

Purpose of the exercises

The aim of the exercises is to learn about the help computers can provide in the circuit development. Help – as the name suggests, it is not a substitute for human, but it can help him to do his job faster, and with fewer mistakes. These exercises will allow us to get to know the companion in the form of a computer that can help us in many situations.

1 Description of the first task

Task 1

Prepare your own definition of one logic function to be minimised, satisfying the given conditions:

a) number of inputs either 6 or 7,

b) number of specified on conditions (elementary implicants): at least 5,

c) number of specified off conditions (elementary implicates): at least 5,

d) when minimised to SoP form, obtained products included in the minimised outcome cannot be single-variable products, and after minimisation none of the input variables can be reduced

Minimise the function with the software implementation of Kazakov algorithm.

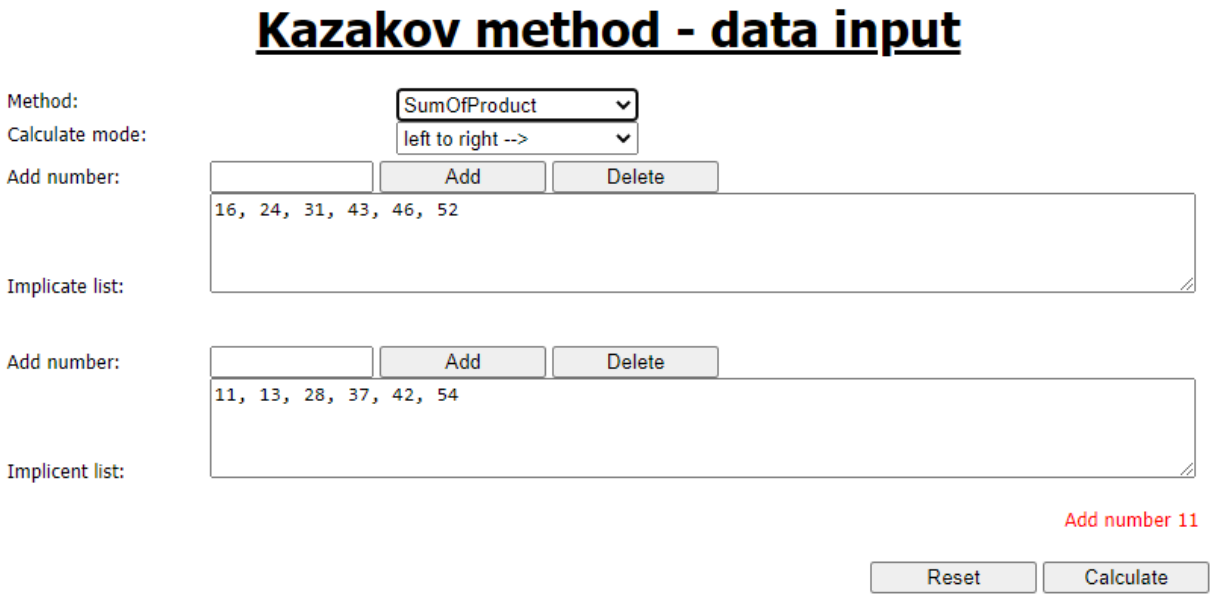
1.1 Tools used to perform the tasks

In order to accomplish this task, a website was used, which is located at the web address:

http://zmitacsim.zmitac.aei.polsl.pl/Kazakov/Page1.aspx

I entered the following function of six variables in the individual fields:

$$F = \Sigma(16, 24, 31, 43, 46, 52)_{x_6x_5x_4x_3x_2x_1x_0}$$
$$F = \Pi(11, 13, 28, 37, 42, 54)_{x_6x_5x_4x_3x_2x_1x_0}$$



Then the appropriate steps of the program were displayed:

F1:

Dec	x0	x1	x2	x3	x4	x5	Nr imp
16	0	1	0	0	0	0	1
24	0	1	1	0	0	0	
31	0	1	1	1	1	1	
43	1	0	1	0	1	1	
46	1	0	1	1	1	0	
52	1	1	0	1	0	0	

F0:

Dec	x0	x1	x2	x3	x4	x5
11	0	0	1	0	1	1
13	0	0	1	1	0	1
28	0	1	1	1	0	0
37	1	0	0	1	0	1
42	1	0	1	0	1	0
54	1	1	0	1	1	0

Cover of implicate

Nr Imp	Value
1	(~x0) * (~x2)

Number of implicate: 1
Cover: (~x0) * (~x2)

Step 1

F1:

Dec	x0	x1	x2	x3	x4	x5	Nr imp
16	0	1	0	0	0	0	1
24	0	1	1	0	0	0	2
31	0	1	1	1	1	1	
43	1	0	1	0	1	1	
46	1	0	1	1	1	0	
52	1	1	0	1	0	0	

F0:

Dec	x0	x1	x2	x3	x4	x5
11	0	0	1	0	1	1
13	0	0	1	1	0	1
28	0	1	1	1	0	0
37	1	0	0	1	0	1
42	1	0	1	0	1	0
54	1	1	0	1	1	0

Cover of implicate

Nr Imp	Value
1	(~x0) * (~x2)
2	x1 * (~x3)

Number of implicate: 2
Cover: x1 * (~x3)

Step 2

F1:

Dec	x0	x1	x2	x3	x4	x5	Nr imp
16	0	1	0	0	0	0	1, 2
24	0	1	1	0	0	0	2
31	0	1	1	1	1	1	
43	1	0	1	0	1	1	
46	1	0	1	1	1	0	
52	1	1	0	1	0	0	

F0:

Dec	x0	x1	x2	x3	x4	x5
11	0	0	1	0	1	1
13	0	0	1	1	0	1
28	0	1	1	1	0	0
37	1	0	0	1	0	1
42	1	0	1	0	1	0
54	1	1	0	1	1	0

Cover of implicate

Nr Imp	Value
1	(~x0) * (~x2)
2	x1 * (~x3)

Number of implicate: 2
Cover: x1 * (~x3)

Step 3

F1:

Dec	x0	x1	x2	x3	x4	x5	Nr imp
16	0	1	0	0	0	0	1, 2
24	0	1	1	0	0	0	2
31	0	1	1	1	1	1	3
43	1	0	1	0	1	1	
46	1	0	1	1	1	0	
52	1	1	0	1	0	0	

F0:

Dec	x0	x1	x2	x3	x4	x5
11	0	0	1	0	1	1
13	0	0	1	1	0	1
28	0	1	1	1	0	0
37	1	0	0	1	0	1
42	1	0	1	0	1	0
54	1	1	0	1	1	0

Cover of implicate

Nr Imp	Value
1	(~x0) * (~x2)
2	x1 * (~x3)
3	x1 * x5

Number of implicate: 3
Cover: x1 * x5

Step 4

F1:

Dec	x0	x1	x2	x3	x4	x5	Nr imp
16	0	1	0	0	0	0	1, 2
24	0	1	1	0	0	0	2
31	0	1	1	1	1	1	3
43	1	0	1	0	1	1	4
46	1	0	1	1	1	0	
52	1	1	0	1	0	0	

F0:

Dec	x0	x1	x2	x3	x4	x5
11	0	0	1	0	1	1
13	0	0	1	1	0	1
28	0	1	1	1	0	0
37	1	0	0	1	0	1
42	1	0	1	0	1	0
54	1	1	0	1	1	0

Cover of implicate

Nr Imp	Value
1	(~x0) * (~x2)
2	x1 * (~x3)
3	x1 * x5
4	x0 * x2 * x5

Number of implicate: 4
Cover: x0 * x2 * x5

Step 5

F1:

Dec	x0	x1	x2	x3	x4	x5	Nr imp
16	0	1	0	0	0	0	1, 2
24	0	1	1	0	0	0	2
31	0	1	1	1	1	1	3
43	1	0	1	0	1	1	4
46	1	0	1	1	1	0	5
52	1	1	0	1	0	0	

F0:

Dec	x0	x1	x2	x3	x4	x5
11	0	0	1	0	1	1
13	0	0	1	1	0	1
28	0	1	1	1	0	0
37	1	0	0	1	0	1
42	1	0	1	0	1	0
54	1	1	0	1	1	0

Cover of implicate

Nr Imp	Value
1	(~x0) * (~x2)
2	x1 * (~x3)
3	x1 * x5
4	x0 * x2 * x5
5	x0 * x2 * x3

Number of implicate: 5
Cover: x0 * x2 * x3

Step 6

F1:

Dec	x0	x1	x2	x3	x4	x5	Nr imp
16	0	1	0	0	0	0	1, 2
24	0	1	1	0	0	0	2
31	0	1	1	1	1	1	3
43	1	0	1	0	1	1	4
46	1	0	1	1	1	0	5
52	1	1	0	1	0	0	6

F0:

Dec	x0	x1	x2	x3	x4	x5
11	0	0	1	0	1	1
13	0	0	1	1	0	1
28	0	1	1	1	0	0
37	1	0	0	1	0	1
42	1	0	1	0	1	0
54	1	1	0	1	1	0

Cover of implicate

Nr Imp	Value
1	(~x0) * (~x2)
2	x1 * (~x3)
3	x1 * x5
4	x0 * x2 * x5
5	x0 * x2 * x3
6	x0 * x1 * (~x4)

Number of implicate: 6
Cover: x0 * x1 * (~x4)

Step 7

F1:

Dec	x0	x1	x2	x3	x4	x5	Nr imp
16	0	1	0	0	0	0	1, 2
24	0	1	1	0	0	0	2
31	0	1	1	1	1	1	3
43	1	0	1	0	1	1	4
46	1	0	1	1	1	0	5
52	1	1	0	1	0	0	6

F0:

Dec	x0	x1	x2	x3	x4	x5
11	0	0	1	0	1	1
13	0	0	1	1	0	1
28	0	1	1	1	0	0
37	1	0	0	1	0	1
42	1	0	1	0	1	0
54	1	1	0	1	1	0

Cover of implicate

Nr Imp	Value
1	(~x0) * (~x2)
2	x1 * (~x3)
3	x1 * x5
4	x0 * x2 * x5
5	x0 * x2 * x3
6	x0 * x1 * (~x4)

Number of implicate: 6
Cover: x0 * x1 * (~x4)

Step 8

The result of the program is the following minimized form of function:

$$F = x_1\,\overline{x_3} + x_1\,x_5 + x_0\,x_2\,x_5 + x_0\,x_2\,x_3 + x_0\,x_1\,\overline{x_4}$$

2 Description of the second task

Task 2

Copy the minimised logic expression obtained by Kazakov algorithm as the input definition of the function to be implemented with multiplexer and demultiplexer elements. With the help of MUX-DMUX software obtain the following solutions:

a) 16-bit MUX + gates

b) 8-bit MUX + gates

c) 4-bit MUX + gates

d) two different DMUX-MUX structures

e) a tree of 4-bit MUX

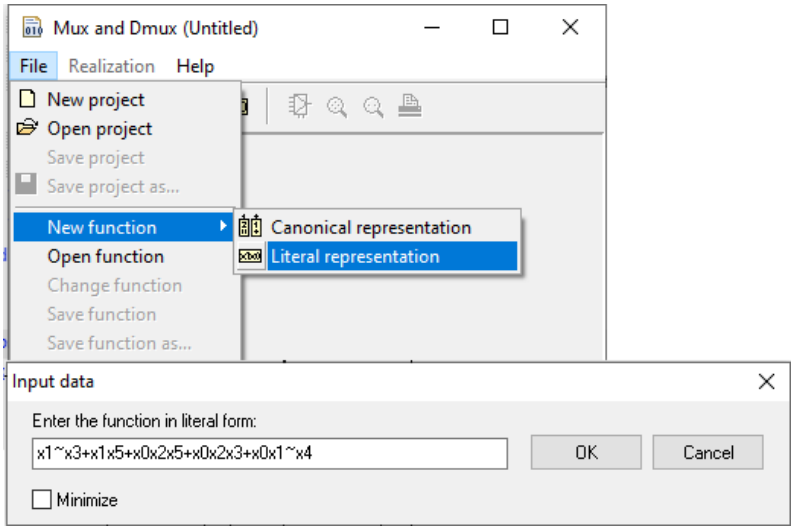
While using gates choose only the type available in the laboratory – either NANDs or NORs.

2.1 The way the software was used

The minimised logic expression obtained by Kazakov algorithm in the previous task was:

$$F = x_1\,\overline{x_3} + x_1\,x_5 + x_0\,x_2\,x_5 + x_0\,x_2\,x_3 + x_0\,x_1\,\overline{x_4}$$

The above function has been entered to the program using: *File* $\longrightarrow$ *New function* $\longrightarrow$ *Literal representation*.

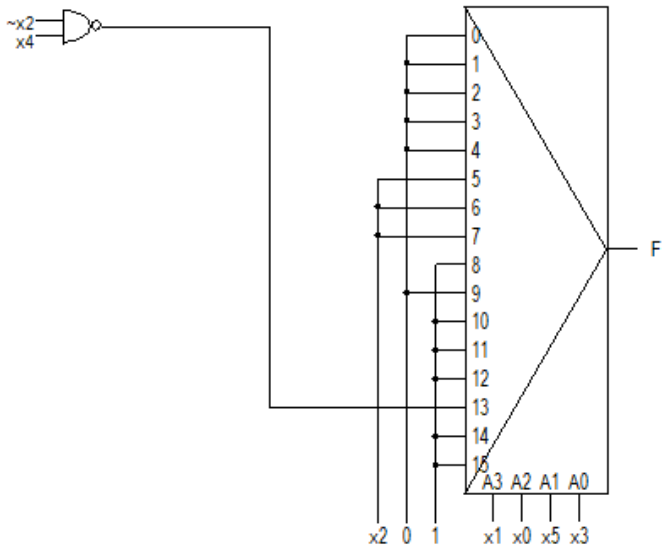


The circuit diagrams, and solutions are then generated by the corresponding wizard under the *Realisation* tab.

2.1.1 16-bit MUX + gates

Function:
~x3x1 + x5x1 + x5x2x0 + x3x2x0 + ~x4x1x0

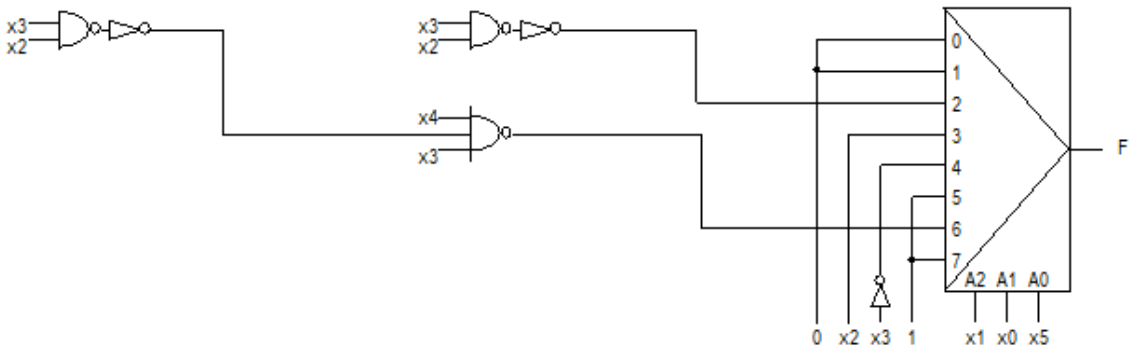
Realized in the structure:
Multiplexer and Gates



2.1.2 8-bit MUX + gates

Function:
~x3x1 + x5x1 + x5x2x0 + x3x2x0 + ~x4x1x0

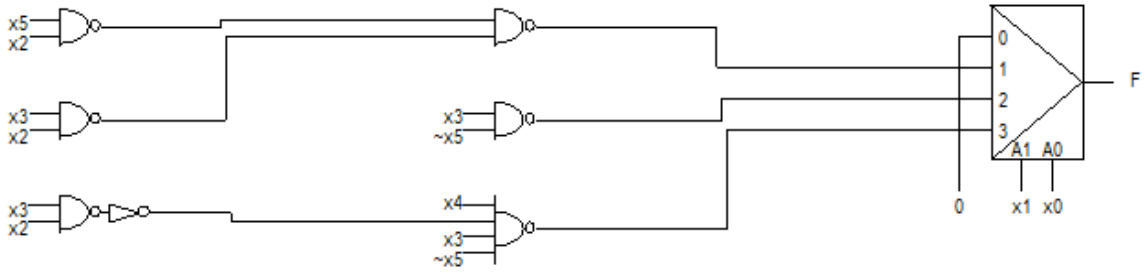
Realized in the structure:
Multiplexer and Gates



2.1.3 4-bit MUX + gates

Function:
~x3x1 + x5x1 + x5x2x0 + x3x2x0 + ~x4x1x0

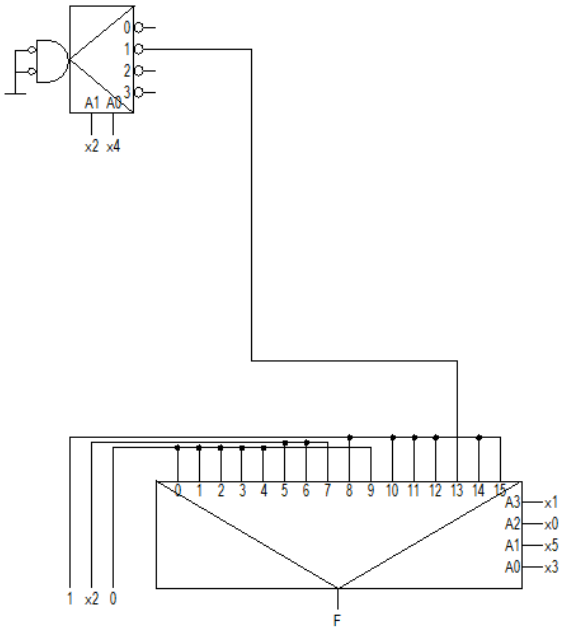
Realized in the structure:
Multiplexer and Gates



2.1.4 two different DMUX-MUX structures

Function:
~x3x1 + x5x1 + x5x2x0 + x3x2x0 + ~x4x1x0

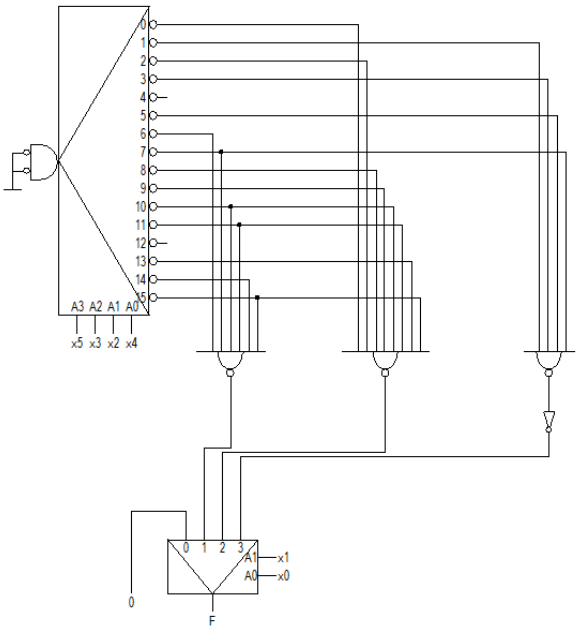
Realized in the structure:
Multiplexer and Demultiplexer



4-bit DMUX and 16-bit MUX

Function:
~x3x1 + x5x1 + x5x2x0 + x3x2x0 + ~x4x1x0

Realized in the structure:
Multiplexer and Demultiplexer

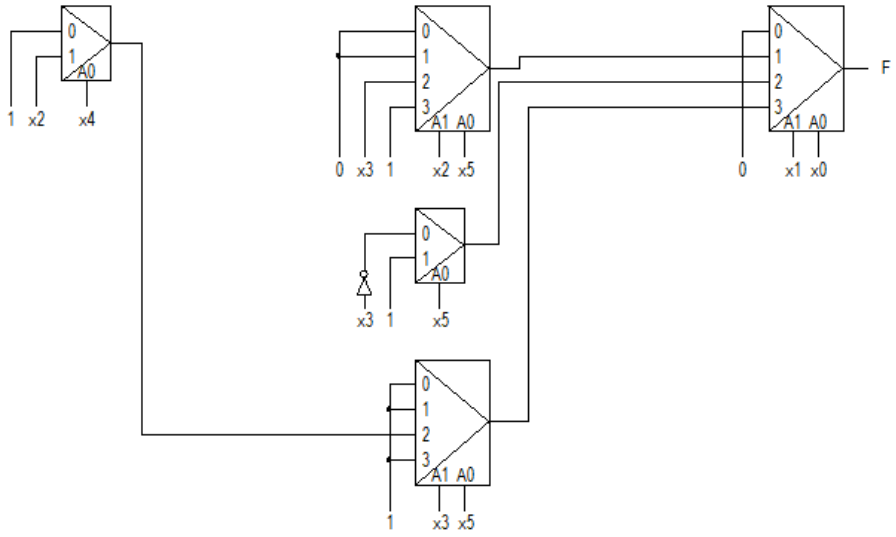


16-bit DMUX and 4-bit MUX

2.1.5 a tree of 4-bit MUX

Function:
~x3x1 + x5x1 + x5x2x0 + x3x2x0 + ~x4x1x0

Realized in the structure:
Multiplexer Tree



3 Description of the third task

Task 3

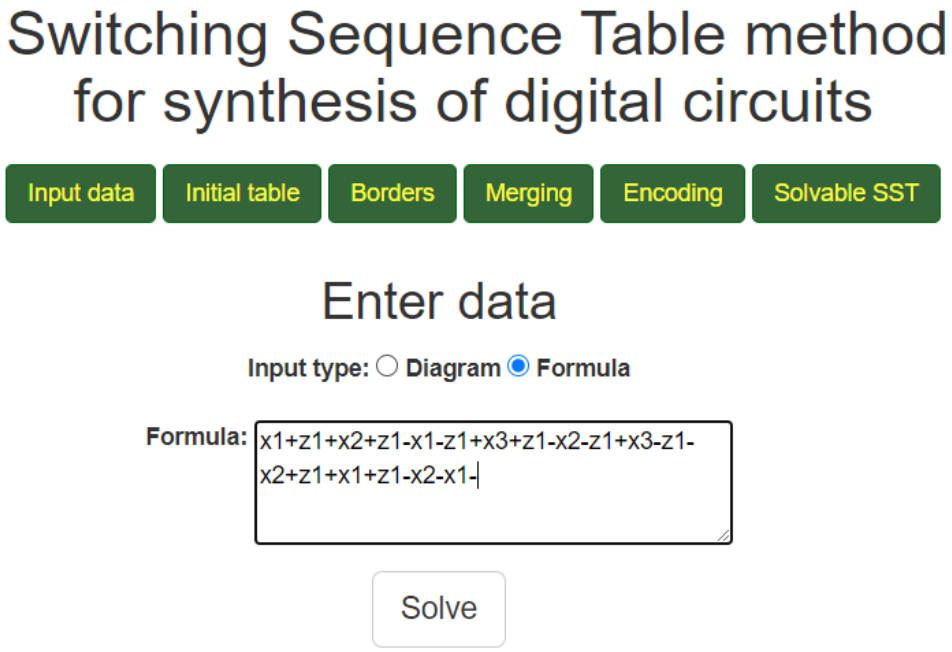
Prepare your own definition of a program for an asynchronous sequential circuit for which SST is unsolvable without auxiliary state variables. List it as a switching sequence and provide this definition to the software, then proceed through all steps of the design.

3.1 Using software to solve the task.

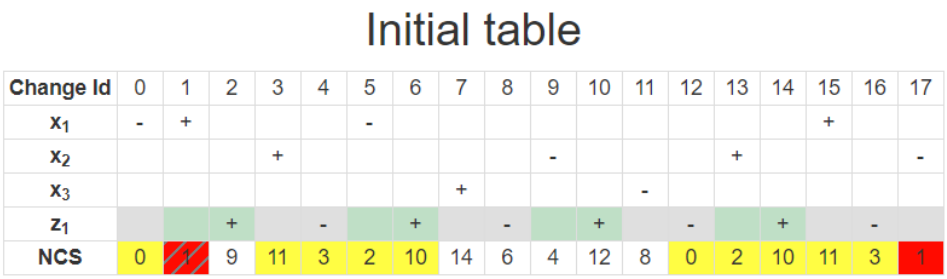
In order to accomplish this task, a website is used, which is located at the web address:

http://zmitacsim.zmitac.aei.polsl.pl/SST/Input/Input

In the first step, I enter the definition of the programme.



Step 1



Change Id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x_1	-	+				-										+		
x_2				+						-				+				-
x_3								+				-						
z_1			+		-		+		-		+		-		+		-	
NCS	0	1	9	11	3	2	10	14	6	4	12	8	0	2	10	11	3	1

Step 2

Borders

Change Id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x_1	-	+				-										+		
x_2				+						-				+				-
x_3								+				-						
z_1			+		-		+		-		+		-		+		-	
NCS	0	1	9	11	3	2	10	14	6	4	12	8	0	2	10	11	3	1

Step 3

Merging

Change Id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x_1	-	+				-										+		
x_2				+						-				+				-
x_3								+				-						
z_1			+		-		+		-		+		-		+		-	
NCS	0	1	9	11	3	2	10	14	6	4	12	8	0	2	10	11	3	1
Regions	← R3	R0											R1	R2				R3 →

Step 4

Encoding

Change Id	0	0'	1	2	3	4	5	6	7	8	9	10	11	11'	12	12'	13	14	15	16	16'	17
x_1	-		+				-												+			
x_2					+						-						+					-
x_3									+				-									
z_1				+		-		+		-		+			-			+		-		
q_0		-														+						
q_1														+							-	
NCS	16	0	1	9	11	3	2	10	14	6	4	12	8	40	32	48	50	58	59	51	19	17

Step 5

Solvable SST

Change Id	0	0'	1	2	3	4	5	6	7	8	9	10	11	11'	12	12'	13	14	15	16	16'	17
x_1	-		+				-												+			
x_2					+						-						+					-
x_3									+				-									
z_1				+		-		+		-		+			-			+		-		
q_0		-														+						
q_1														+							-	
NCS	16	0	1	9	11	3	2	10	14	6	4	12	8	40	32	48	50	58	59	51	19	17

Signal	$\Sigma()_{q_1q_0z_1x_3x_2x_1}$	$\Pi()_{q_1q_0z_1x_3x_2x_1}$
z_1	1, 2, 4, 8, 9, 10, 12, 50, 58	0, 3, 6, 11, 14, 16, 17, 19, 32, 40, 48, 51, 59
q_0	17, 19, 32, 48, 50, 51, 58, 59	0, 1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 14, 16, 40
q_1	8, 32, 40, 48, 50, 58, 59	0, 1, 2, 3, 4, 6, 9, 10, 11, 12, 14, 16, 17, 19, 51

Step 6

From the table above we read the following results:

$$z_1 = \Sigma(1,2,4,8,9,10,12,50,58)_{q_1 q_0 z_1 x_3 x_2 x_1}$$
$$z_1 = \Pi(0,3,6,11,14,16,17,19,32,40,48,51,59)_{q_1 q_0 z_1 x_3 x_2 x_1}$$

$$q_0 = \Sigma(17,19,32,48,50,51,58,59)_{q_1 q_0 z_1 x_3 x_2 x_1}$$
$$q_0 = \Pi(0,1,2,3,4,6,8,9,10,11,12,14,16,40)_{q_1 q_0 z_1 x_3 x_2 x_1}$$

$$q_1 = \Sigma(8,32,40,48,50,58,59)_{q_1 q_0 z_1 x_3 x_2 x_1}$$
$$q_1 = \Pi(0,1,2,3,4,6,9,10,11,12,14,16,17,19,51)_{q_1 q_0 z_1 x_3 x_2 x_1}$$

4 Conclusions

The use of a computer, which is able to give the results of many useful algorithms very quickly, significantly accelerates the process of creating digital circuits. We ourselves use various simulators during classes, which allow us to check the operation and correctness of systems. In these matters, the computer can be of real help to both inexperienced circuit developers (as way to learn) and experienced circuit developers (as a quick way of checking the correctness of advanced systems).