

Digital Circuits Theory - Laboratory						
Academic year	Laboratory exercises on	Mode of studies	Field of studies	Supervisor	Group	Section
2024/2025	Wednesday	SSI	Informatics	US	1	1
	11:45 – 13:15					

Report from Exercise No 12

Performed on: 07.01.2025

Exercise Topic: Computer Aided Design for Circuit Development

Performed by:

Piotr Copek

Introduction

This report explores the use of software tools to help with implementation of digital circuits. These tools assist in minimizing logic functions using the Kazakov algorithm, implementing expressions through MUX-DMUX structures, and creating solvable tables and functions for unsolvable SSTs.

Task 1

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

- number of inputs either 6 or 7
- number of specified on conditions (elementary implicants): at least 5,
- number of specified off conditions (elementary implicants): at least 5,
- 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 (they all must still be present in the obtained minimised expression)

Minimise the function with the software implementation of Kazakov algorithm

Solution


To obtain desired solution I used the following tool:



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

I entered the following function of six variables into the input:

$$F = \begin{cases} \sum (3, 7, 13, 19, 29, 37, 43)_{x_5 x_4 x_3 x_2 x_1 x_0} \\ \prod (2, 5, 11, 17, 23, 31, 41)_{x_5 x_4 x_3 x_2 x_1 x_0} \end{cases}$$

Documentation made through the working process with software:



Kazakov method - data input

Method: SumOfProduct

Calculate mode: left to right -->

Add number: Add Delete

3, 7, 13, 19, 29, 37, 43

Implicate list:

Add number: Add Delete

2, 5, 11, 17, 23, 31, 41

Implicant list:

Add number 41
Reset
Calculate

(c) 2010, 2013 Mirka Kuczera, Piotr Czekalski, v. 2.0.0.20151

Figure 1 - Step 1

F1:

Dec	x0	x1	x2	x3	x4	x5	Nr imp
3	0	0	0	0	1	1	1
7	0	0	0	1	1	1	
13	0	0	1	1	0	1	
19	0	1	0	0	1	1	
29	0	1	1	1	0	1	
37	1	0	0	1	0	1	
43	1	0	1	0	1	1	

F0:

Dec	x0	x1	x2	x3	x4	x5
2	0	0	0	0	1	0
5	0	0	0	1	0	1
11	0	0	1	0	1	1
17	0	1	0	0	0	1
23	0	1	0	1	1	1
31	0	1	1	1	1	1
41	1	0	1	0	0	1

Cover of implicate

Nr Imp	Value
1	$(\sim x_1) * (\sim x_2) * (\sim x_3) * x_5$

Number of implicate: 1

Cover: $(\sim x_1) * (\sim x_2) * (\sim x_3) * x_5$

Figure 2 - Step 2

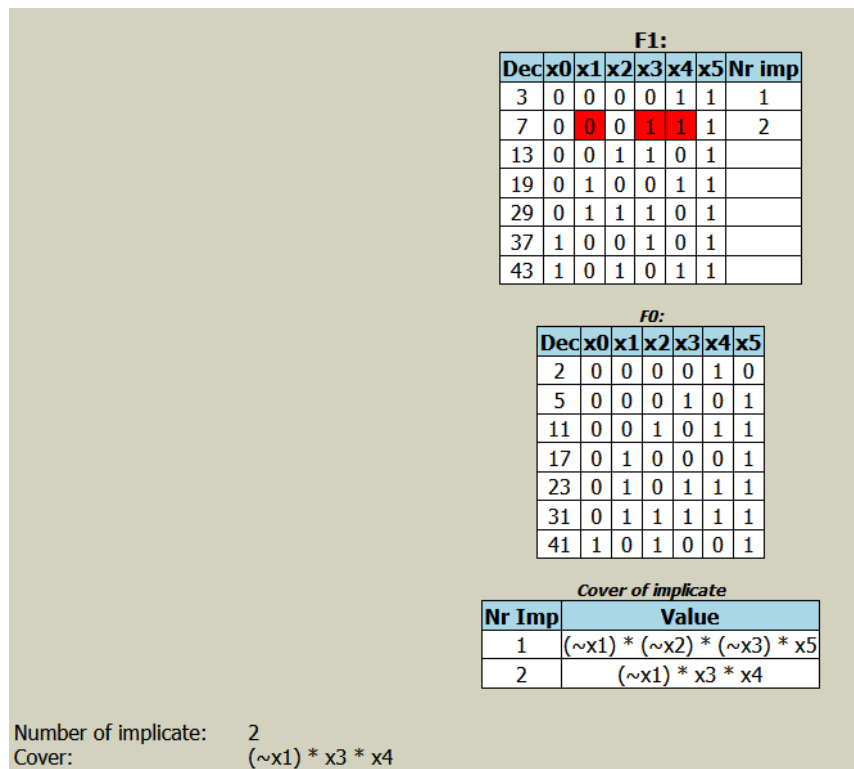


Figure 3 - Step 3

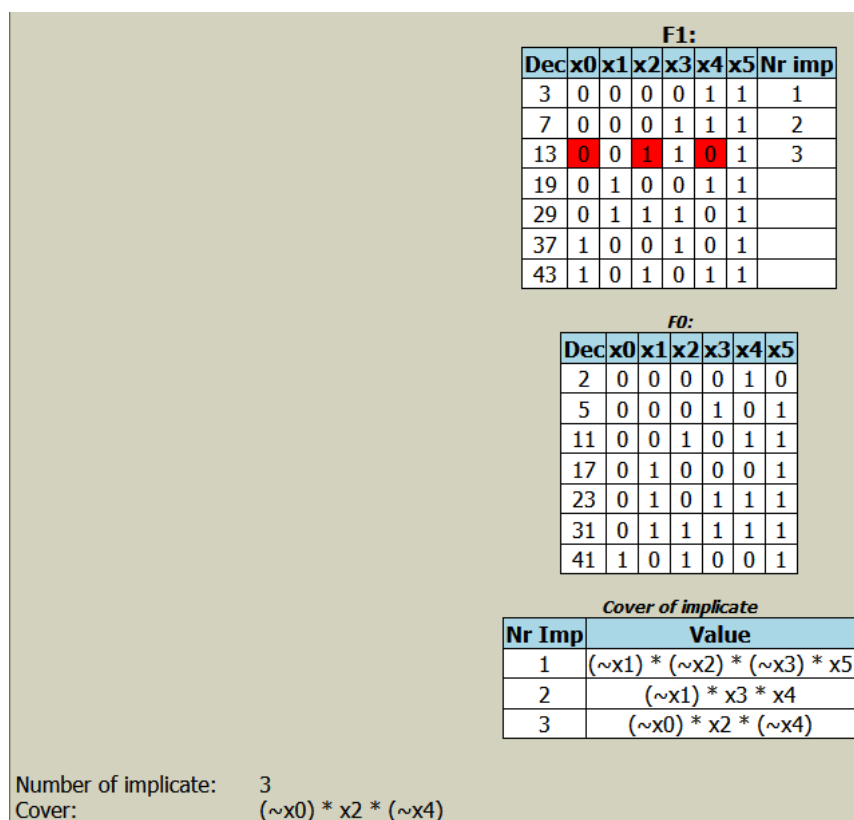


Figure 4 - Step 4

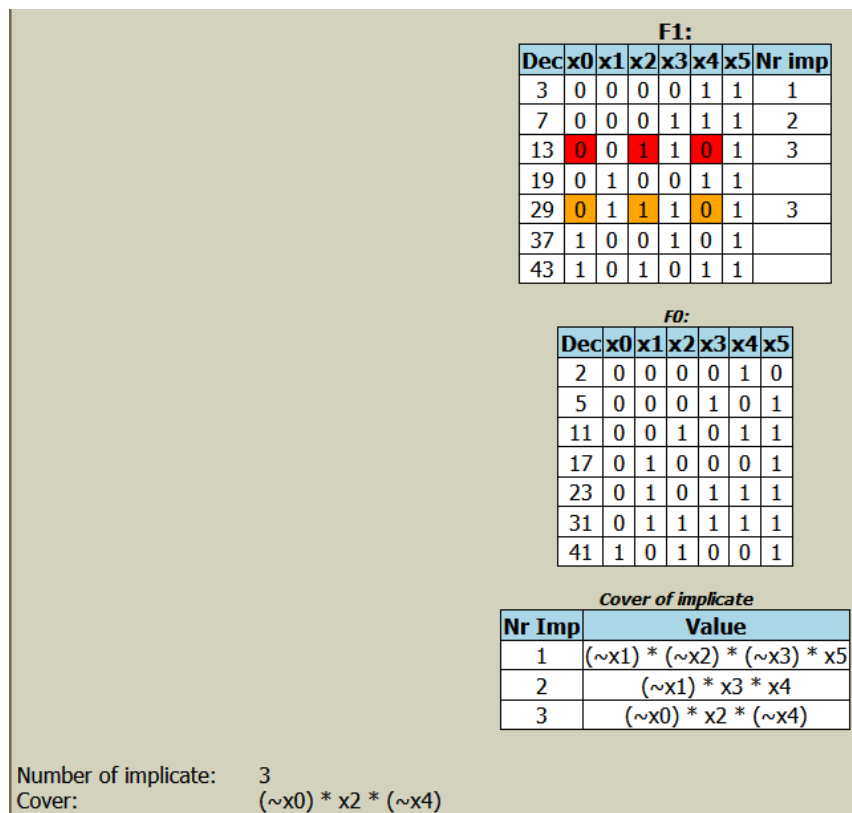


Figure 5 - Step 5

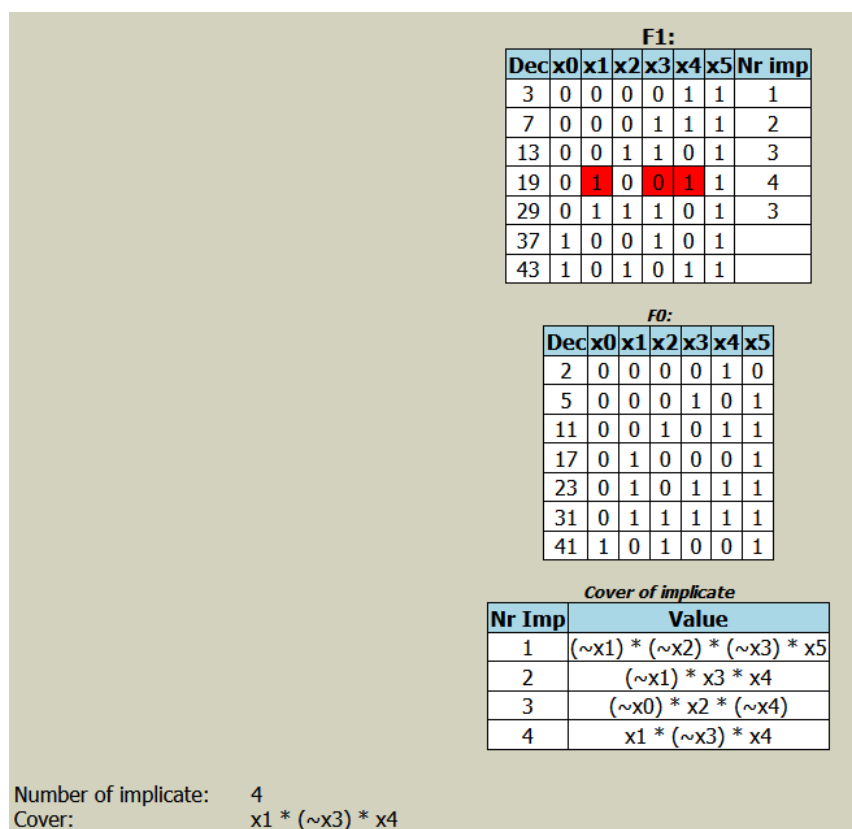


Figure 6 - Step 6

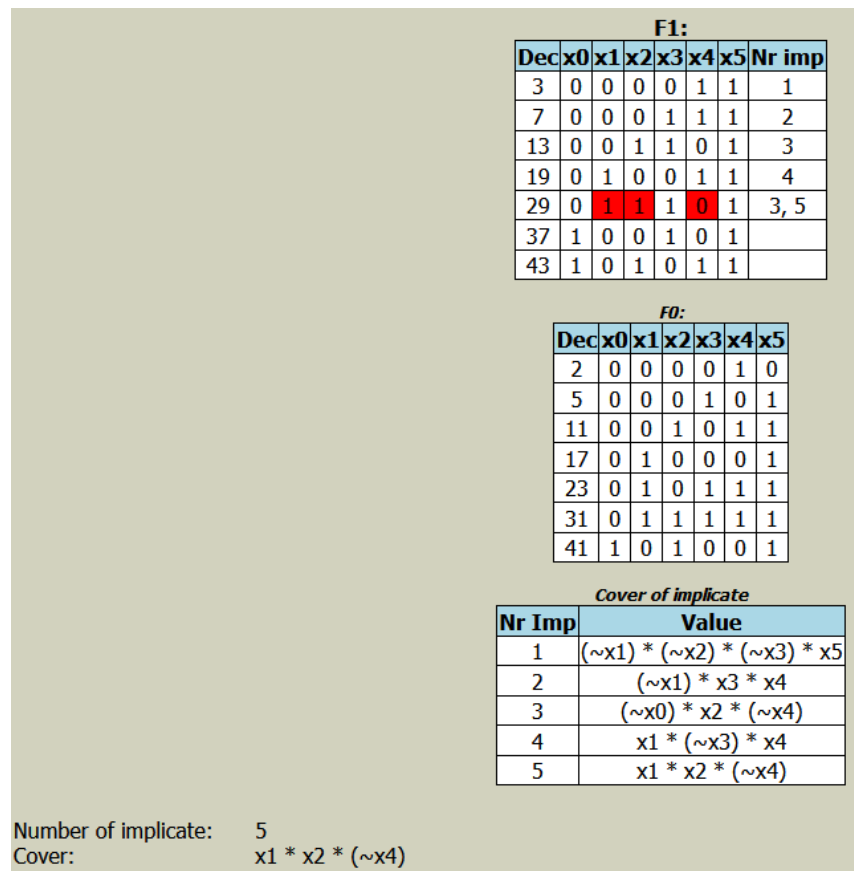


Figure 7 - Step 7

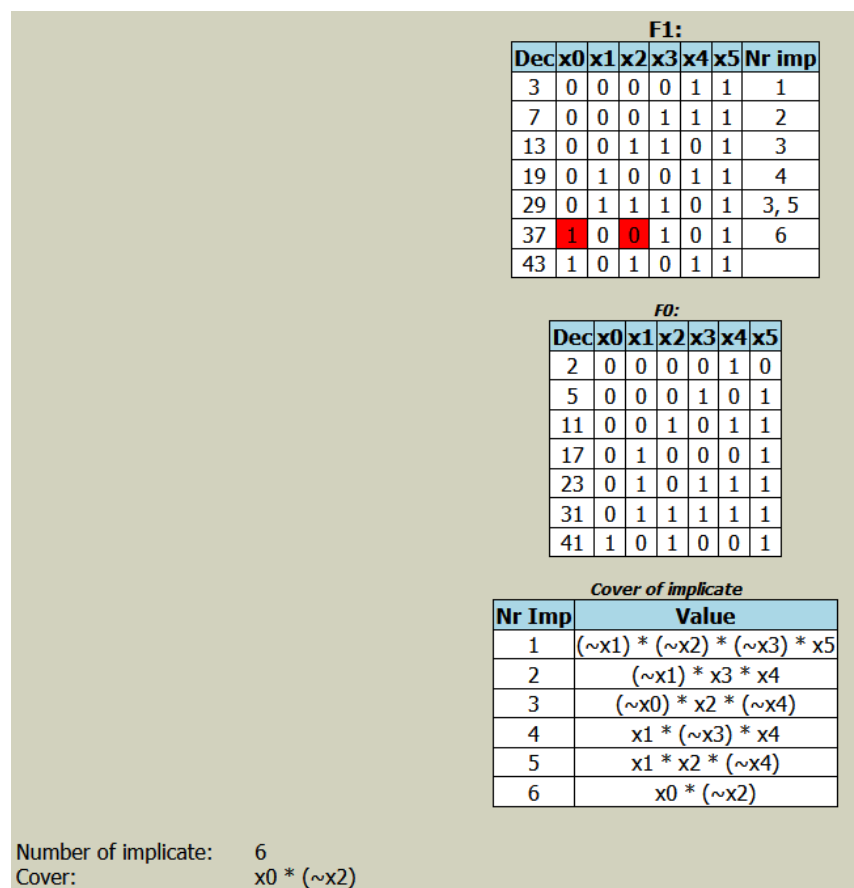


Figure 8 - Step 8

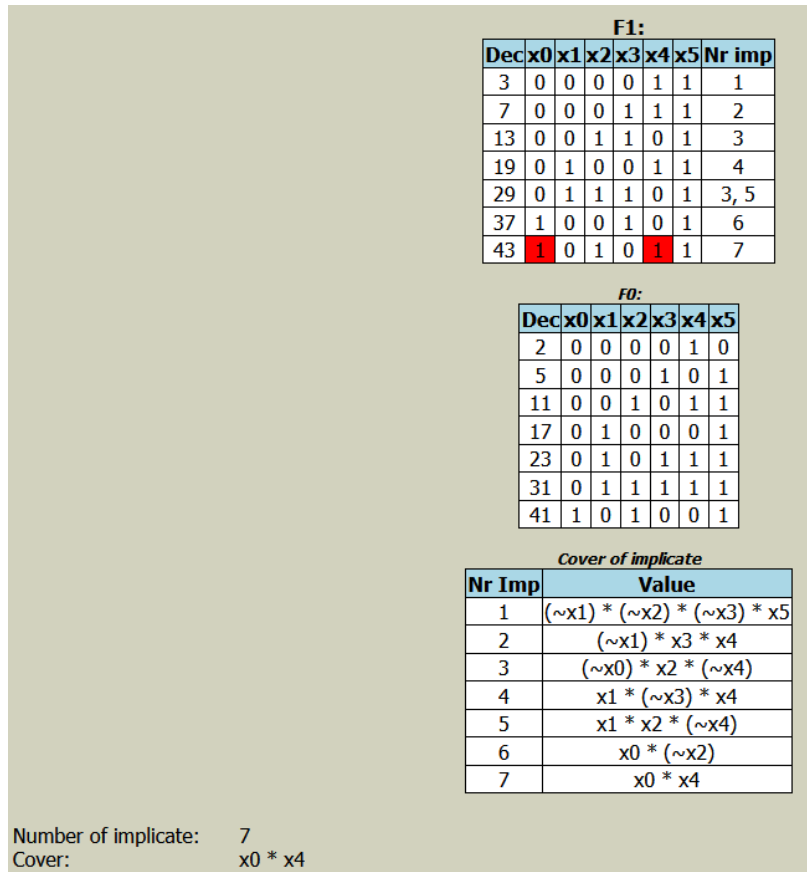


Figure 9 - Step 9

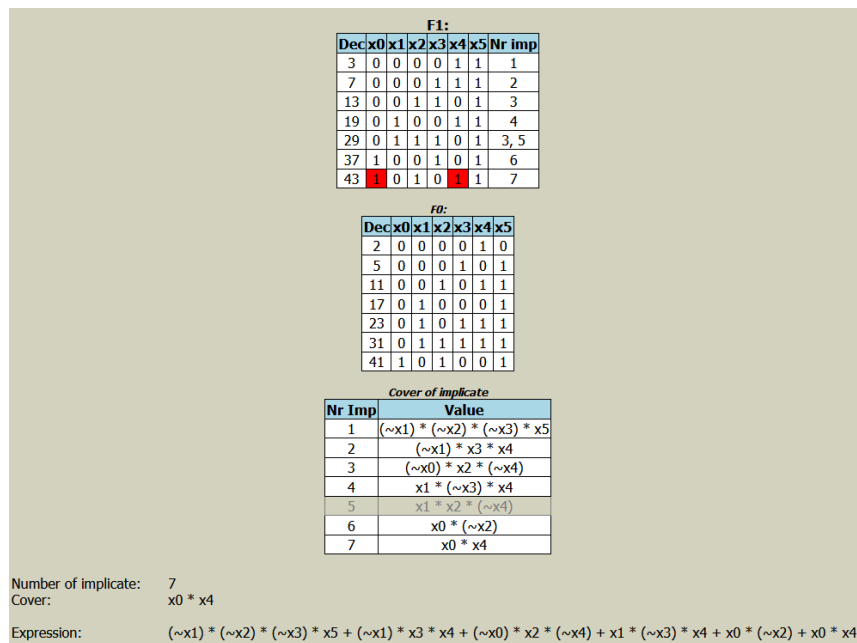


Figure 10 - Step 10

$$(\sim x1) * (\sim x2) * (\sim x3) * x5 + (\sim x1) * x3 * x4 + (\sim x0) * x2 * (\sim x4) + x1 * (\sim x3) * x4 + x0 * (\sim x2) + x0 * x4$$

Figure 11 - Final result

The function obtained with the use of the software looks following:

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

Task 2

Copy the minimized logic expression obtained from the Kazakov algorithm as the input definition of the function to be implemented using multiplexer and demultiplexer elements. Then, with the assistance of MUX-DMUX software, produce the following solutions:

- 16-bit MUX + gates
- 8-bit MUX + gates
- 4-bit MUX + gates
- Two different DMUX-MUX structures
- A tree of 4-bit MUX modules

Note: When adding gates, use only those available in the laboratory (NANDs or NORs).

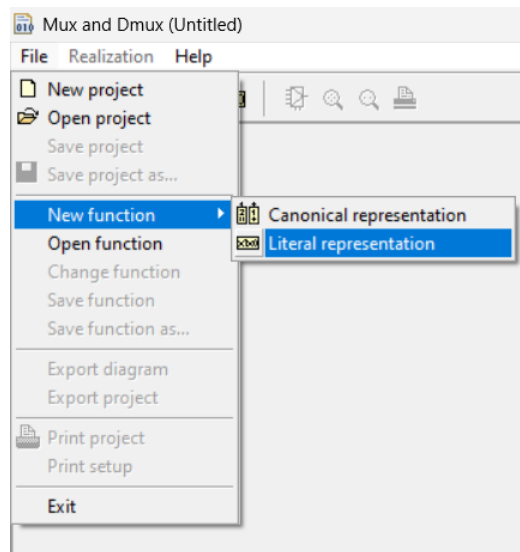
Solution

Minimised logic expression obtained from previous task:

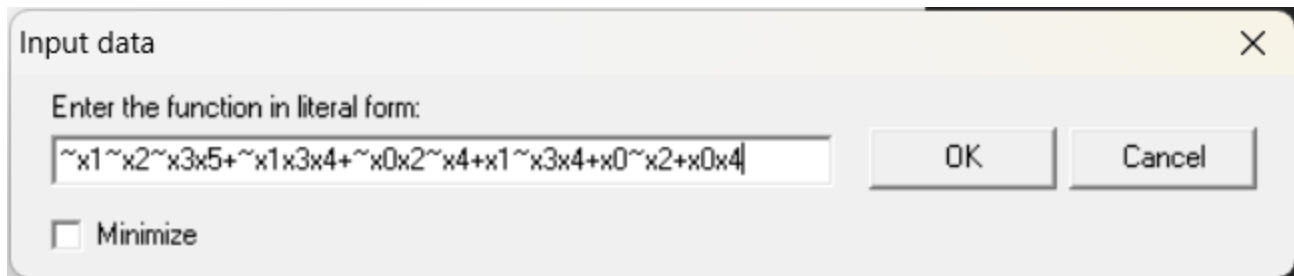
$$F = \overline{x_1} \overline{x_2} \overline{x_3} x_5 + \overline{x_1} x_3 x_4 + \overline{x_0} x_2 \overline{x_4} + x_1 \overline{x_3} x_4 + x_0 \overline{x_2} + x_0 x_4$$

I entered the minimised function to the program using following steps:

File → New function → Literal representation



Next I putted the expression to the dialog input using ~ as negation



a) 16-bit MUX + gates

Function:

$$x_5 \sim x_3 \sim x_2 \sim x_1 + x_4 x_3 \sim x_1 + \sim x_4 x_2 \sim x_0 + x_4 \sim x_3 x_1 + \sim x_2 x_0 + x_4 x_0$$

Realized in the structure:

Multiplexer and Gates

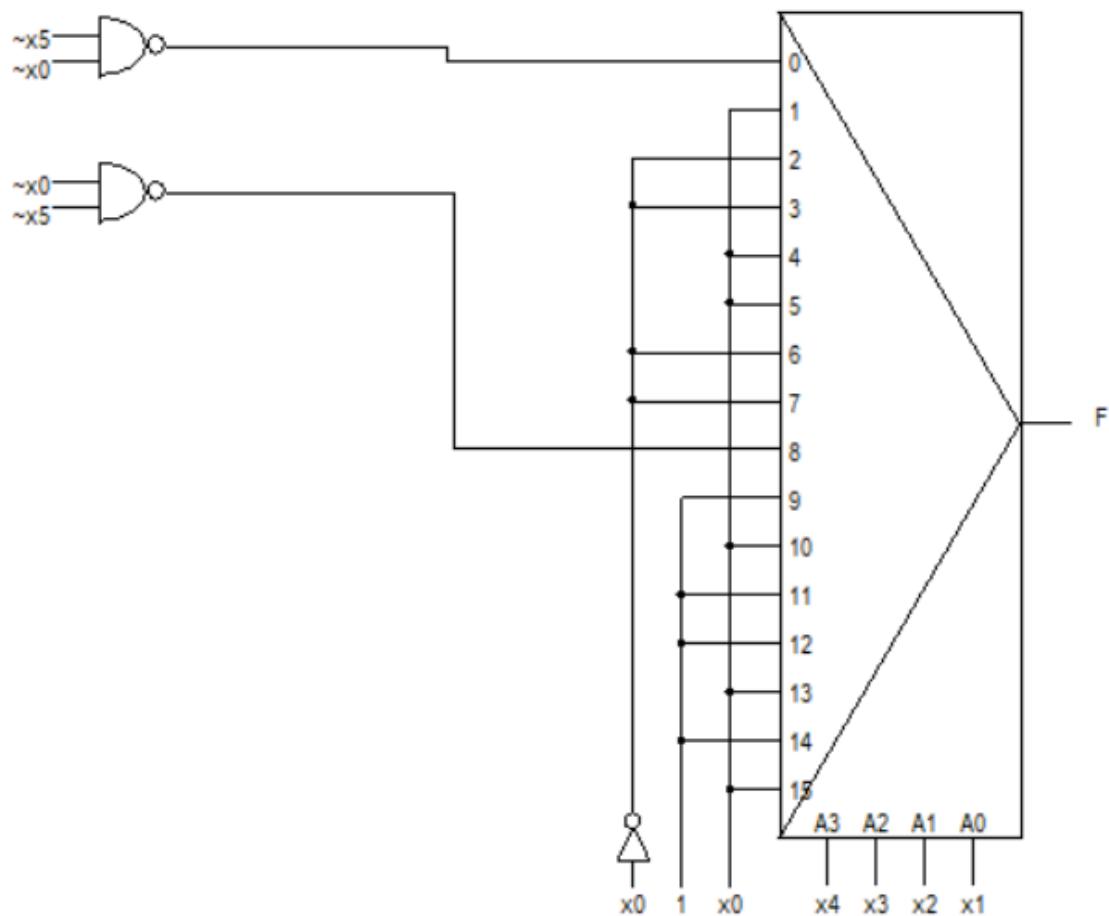


Figure 12 - Software output after selecting *Realization* → *Multiplexer* and selecting 16 for *Multiplexer's* size and checking *use gates* and selecting *NANDs only* in *Gates*.

b) 8-bit MUX + gates

Function:

$$x_5 \sim x_3 \sim x_2 \sim x_1 + x_4 x_3 \sim x_1 + \sim x_4 x_2 \sim x_0 + x_4 \sim x_3 x_1 + \sim x_2 x_0 + x_4 x_0$$

Realized in the structure:

Multiplexer and Gates

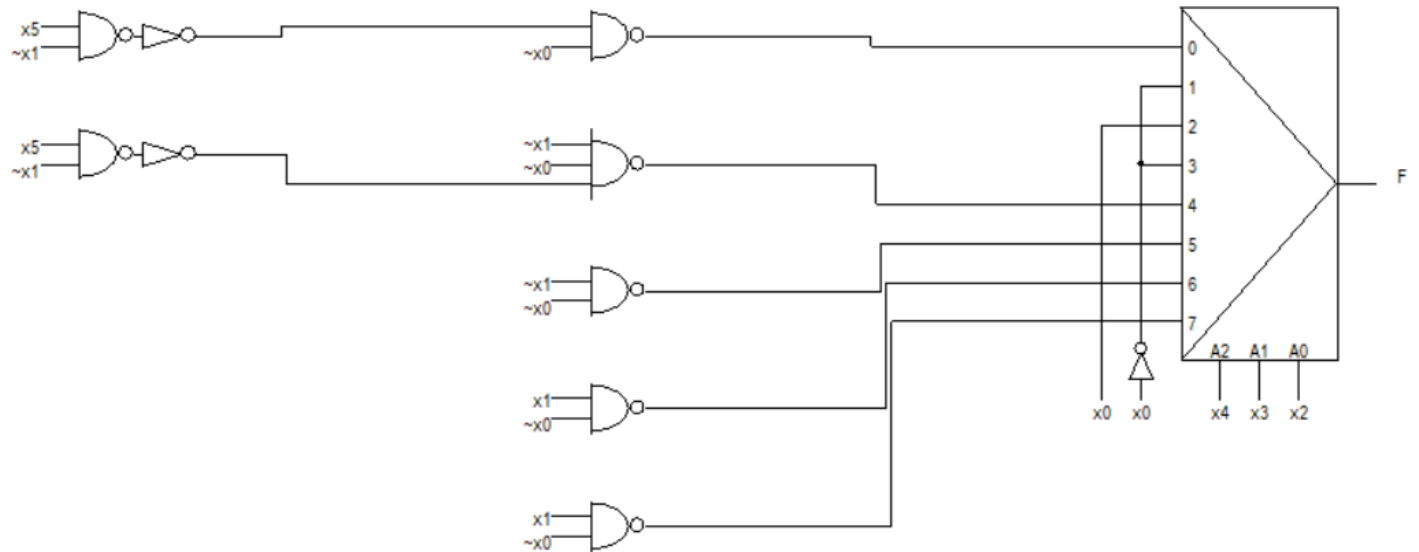


Figure 13 - Software output after selecting *Realization* \rightarrow *Multiplexer* and selecting 8 for *Multiplexer's size* and checking *use gates* and selecting *NANDs only* in *Gates*.

c) 4-bit MUX + gates

Function:

$$x_5 \sim x_3 \sim x_2 \sim x_1 + x_4 x_3 \sim x_1 + \sim x_4 x_2 \sim x_0 + x_4 \sim x_3 x_1 + \sim x_2 x_0 + x_4 x_0$$

Realized in the structure:

Multiplexer and Gates

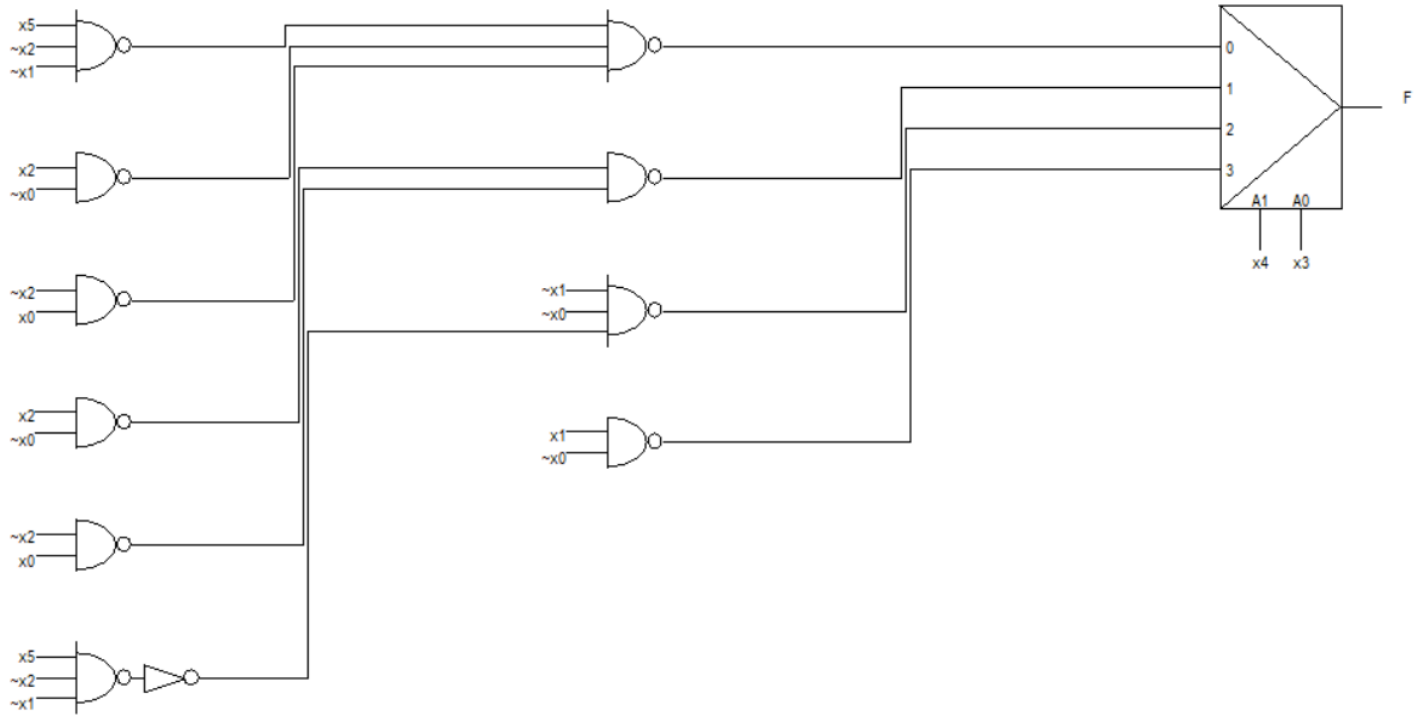


Figure 14 - Software output after selecting *Realization* → *Multiplexer* and selecting 4 for *Multiplexer's size* and checking *use gates* and selecting *NANDs only* in *Gates*.

d) two different DMUX-MUX structures

Function:

$$x5 \sim x3 \sim x2 \sim x1 + x4 x3 \sim x1 + \sim x4 x2 \sim x0 + x4 \sim x3 x1 + \sim x2 x0 + x4 x0$$

Realized in the structure:

Multiplexer and Demultiplexer

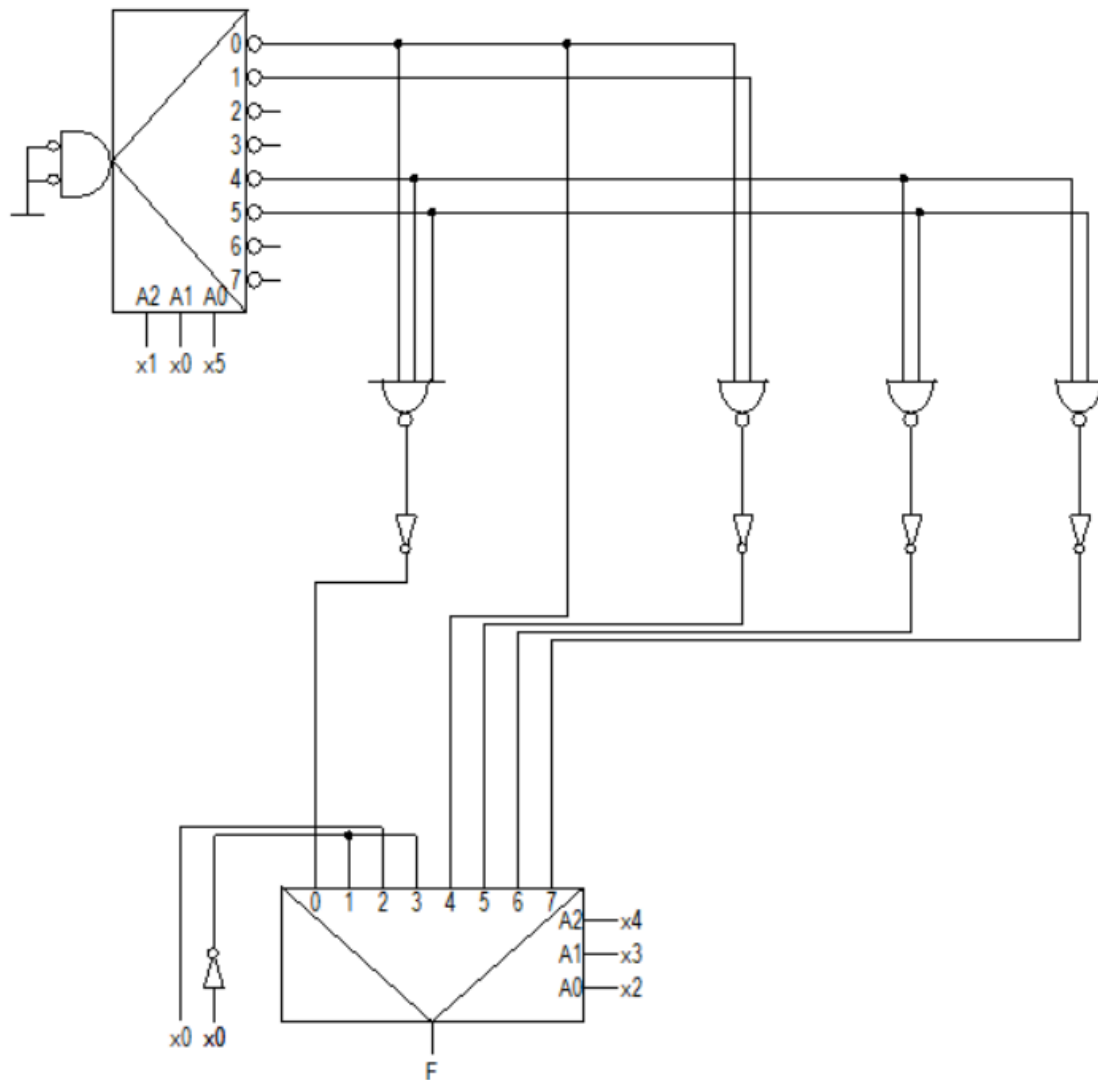


Figure 15 - Software output after selecting *Realization* → *Multiplexer and Demultiplexer* and selecting 8 for *Multiplexer's size* and *Demultiplexer's size*.

Function:

$$x^5 \sim x^3 \sim x^2 \sim x^1 + x^4 x^3 \sim x^1 + \sim x^4 x^2 \sim x^0 + x^4 \sim x^3 x^1 + \sim x^2 x^0 + x^4 x^0$$

Realized in the structure:

Multiplexer and Demultiplexer

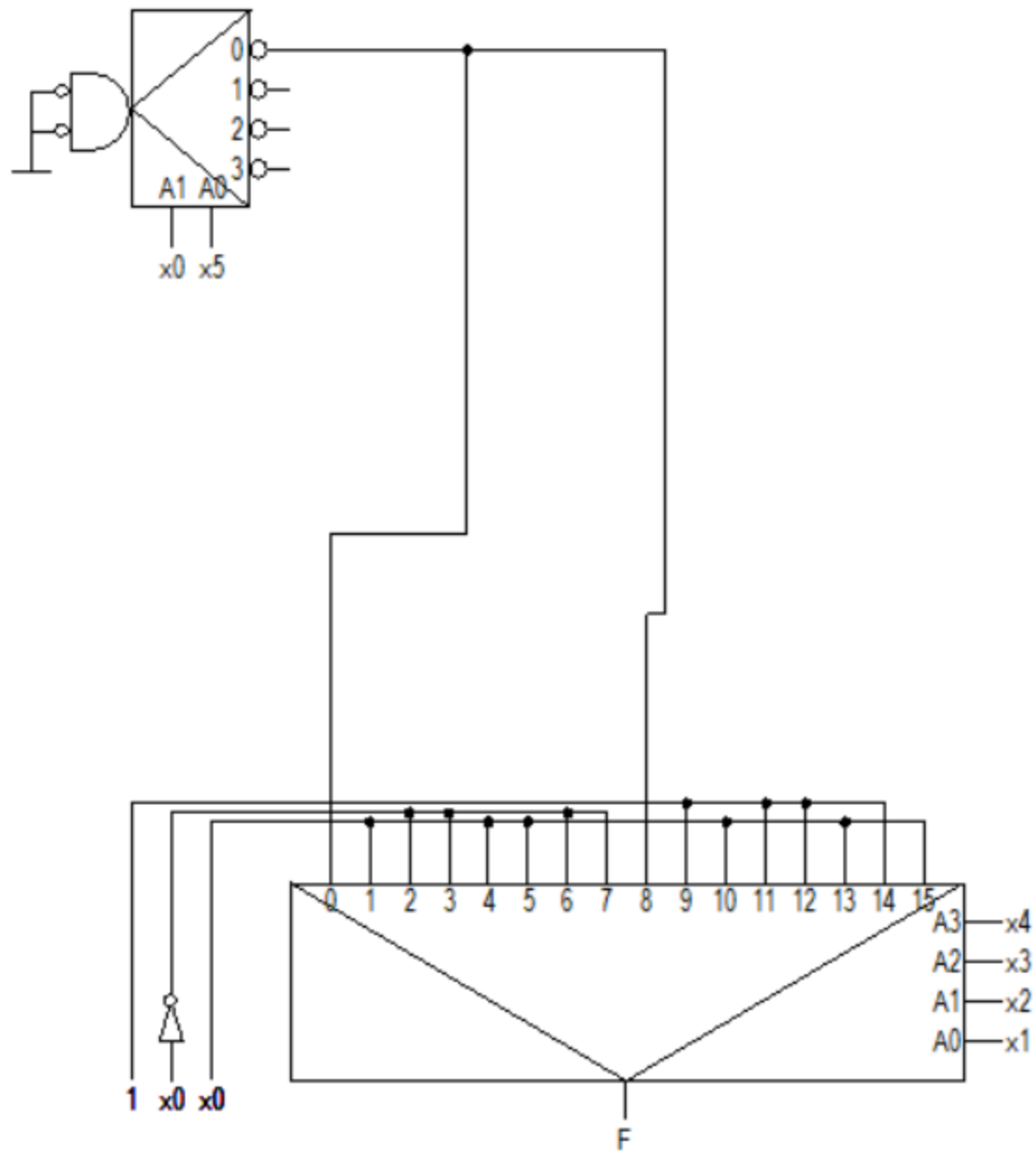


Figure 16 - Software output after selecting *Realization* \rightarrow *Multiplexer* and *Demultiplexer* and selecting 16 for *Multiplexer's size* and 4 for *Demultiplexer's size*.

e) a tree of 4-bit MUX

Function:

$$x_5 \sim x_3 \sim x_2 \sim x_1 + x_4 x_3 \sim x_1 + \sim x_4 x_2 \sim x_0 + x_4 \sim x_3 x_1 + \sim x_2 x_0 + x_4 x_0$$

Realized in the structure:

Multiplexer Tree

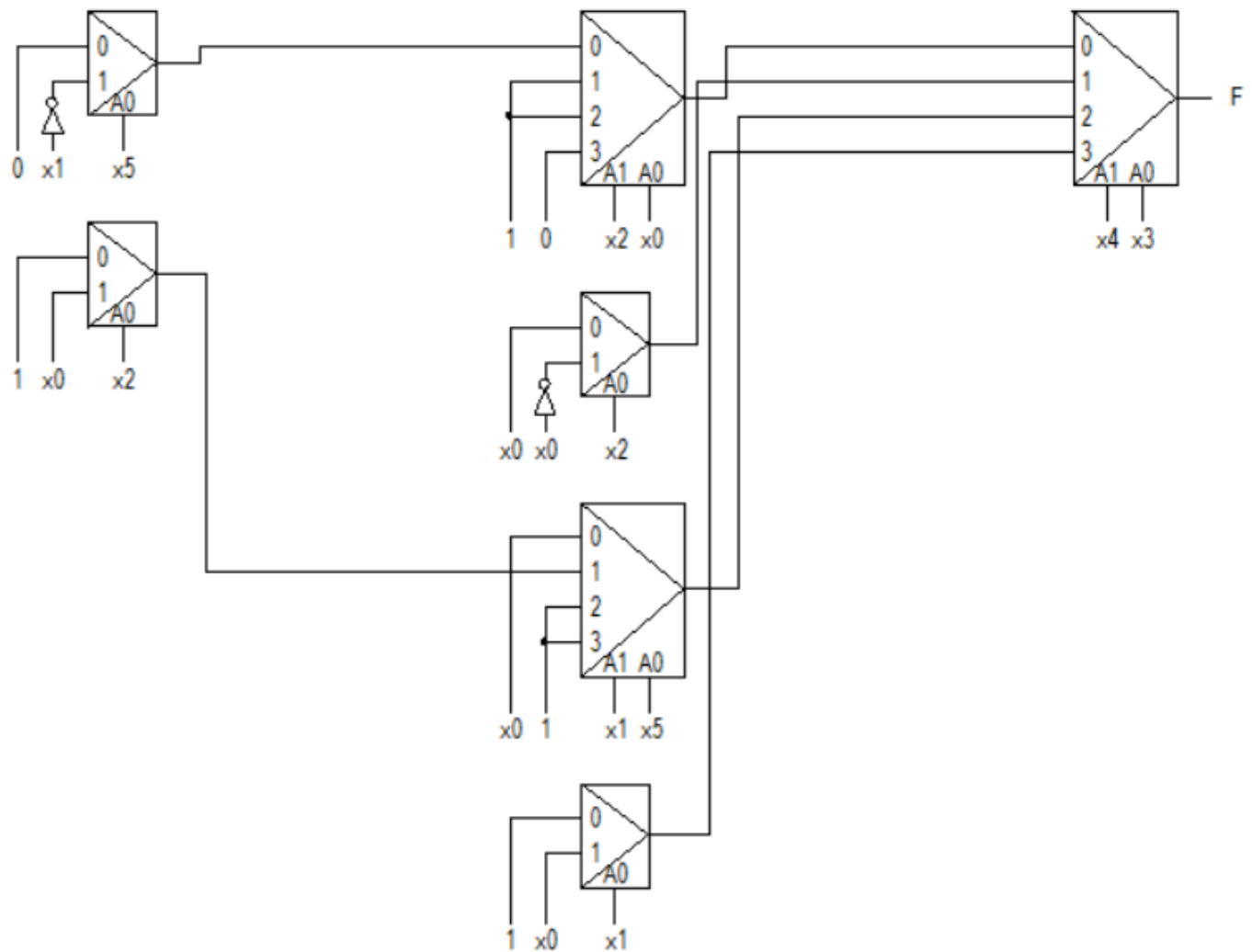


Figure 17 - Software output after selecting *Realization* \rightarrow *Multiplexer tree* and selecting 4 for *Multiplexer's size*. Worth notice that program replaced some 4-bit multiplexers with the 2-bit ones.

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.

Solution

To obtain desired solution I used the following tool:

- <http://zmitacsim.zmitac.aei.polsl.pl/SST/Input/Input>

I entered the following formula to the program:

$$x_0 + z + x_1 + z - x_1 - z + x_0 - x_0 + x_1 + x_0 - z - x_1 -$$

Documentation made through the working process with software:

Enter data

Input type: ☐ Diagram ☒ Formula

Formula:

Figure 18 - Step 1

Initial table

Change Id	0	1	2	3	4	5	6	7	8	9	10	11
x_0		+						-	+		-	
x_1	-			+		-				+		
z			+		-		+					-
NCS	0	1	5	7	3	1	5	4	5	7	6	2

Figure 19 - Step 2

Borders

Change Id	0	1	2	3	4	5	6	7	8	9	10	11
x_0		+						-	+		-	
x_1	-			+		-				+		
z			+		-		+					-
NCS	0	1	5	7	3	1	5	4	5	7	6	2

Figure 20 - Step 3

Merging

Change Id	0	1	2	3	4	5	6	7	8	9	10	11
x_0		+						-	+		-	
x_1	-			+		-				+		
z			+		-		+					-
NCS	0	1	5	7	3	1	5	4	5	7	6	2
Regions	← R1			R0					R1 →			

Figure 21 - Step 4

Encoding

Change Id	0	1	1'	2	3	4	5	6	7	7'	8	9	10	11
x_0		+							-		+		-	
x_1	-				+		-					+		
z				+		-		+						-
q_0			-							+				
NCS	8	9	1	5	7	3	1	5	4	12	13	15	14	10

Figure 22 - Step 5

Solvable SST

Change Id	0	1	1'	2	3	4	5	6	7	7'	8	9	10	11
x_0		+							-		+		-	
x_1	-				+		-					+		
z				+		-		+						-
q_0			-							+				
NCS	8	9	1	5	7	3	1	5	4	12	13	15	14	10

Signal	$\Sigma()_{q_0 z x_1 x_0}$	$\Pi()_{q_0 z x_1 x_0}$
z	1, 4, 5, 12, 13, 15	3, 7, 8, 9, 10, 14
q_0	4, 8, 10, 12, 13, 14, 15	1, 3, 5, 7, 9

Figure 23 - Step 6; Final result

Functions read from the result table:

$$z = \begin{cases} \Sigma(1, 4, 5, 8, 12, 13, 15)_{q_0 z x_1 x_0} \\ \Pi(3, 7, 9, 10, 14)_{q_0 z x_1 x_0} \end{cases}$$

$$q_0 = \begin{cases} \Sigma(4, 8, 10, 12, 13, 14, 15)_{q_0 z x_1 x_0} \\ \Pi(1, 3, 5, 7, 9)_{q_0 z x_1 x_0} \end{cases}$$

Summary

The software tools simplify handling complex digital circuit tasks, such as logic minimization, MUX-DMUX implementation, and addressing asynchronous circuit challenges. They help validate the correctness of solutions and are helpful in learning basics of digital circuit design and applications.