Archives of
Transport System
Telematics

Volume 11

Issue 3

September 2018

Computer Aided Synthesis of Reversible Logic in Traffic Control Systems

R. PNIEWSKI
UNIVERSITY OF TECHNOLOGY AND HUMANITIES IN RADOM, Faculty of Transport and Electrical Engineering, Malczewskiego 29, 26-600 Radom, Poland
EMAIL: r.pniewski@uthrad.pl

ABSTRACT

Currently traffic control systems in place relay systems used microprocessor systems. A better solution may be the use of reversible logic in the synthesis of digital control systems. The main problem in the design of reversible logic is the transformation of the description of the system from the form of Boolean equations to the reversible form. The article presents error-proof reversible gates and software supporting the automatic synthesis process. The presented programs were developed and launched by the author of the article. The article presents the algorithms used to describe the description of circuits. The results of the programs' operation and results of simulation of systems with reversible logic are presented.

KEYWORDS: Traffic control system, revesible logic, synthesis

1. Introduction

The use of programmable logic circuits (PLD) in railway automation systems significantly increases the reliability of these systems in relation to systems implemented in discrete technology and computer systems [5, 6, 9]. Increasing reliability results in an increase in the level of security, provided that harmful phenomena from digital systems based on programmable systems are eliminated. The second important factor in the use of digital systems (especially asynchronous automata) is the ability to "directly map" in PLD systems of proven solutions used in relay systems. The use of these systems is also more beneficial due to the power dissipated in the system. Power balance for asynchronous circuits more advantageous in relation to synchronous systems [12]. The technique of asynchronous circuits has been neglected for many years (due to major implementation difficulties - presented in this publication) despite its unquestionable advantages in relation to synchronous machines.

The advantages of asynchronous automata in relation to the synchronous technique include:

- Smaller, about 50% power loss
- Higher frequency of work
- Quick response without waiting for the clock
- Greater robustness (Robust)
- Possibility to create modular systems
- Possibility to connect multiple systems with their own independent clocks

During the design of devices the biggest problem turned out to be the complexity of the synthesis process of reversible logic circuits [12, 13, 14]. The article presents three programs designed by the author to support the design of digital systems:

- program for coding automata without critical races
- program to convert from boolean equations to Reed-Müller codes
- models of reversible gates in the QUCS program

The basic disadvantages of asynchronous systems include:

- No tools (EDA tools) to support the design of asynchronous automata
- Lack of system design methodology
- Difficulties in project verification

2. The method of elementary conditions

The presented method allows full automation of the process of unconditional encryption of asynchronous automata [11]. In this method, elementary transitions between states are compared. The idea of the method will be presented in a simple example of the machine described in the table of transitions (Table 1)

Table 1. Sate table of the automata[own study]

	X1	X2	X3	X4
1	1	4	1	2
2	2	2	1	2
3	1	2	3	4
4	2	4	3	4

The method compares (separately for each input vector) transitions between states having different successors (different target states in the transition table). For example, for the input vector X1, the table has the following transitions:

1 -> 1 , 2 -> 2 , 3 -> 1 , 4 -> 2

As a result of the comparison, conditions (codes) are created in which the states contained in one pair of transitions are assigned the value "0" to the other pair "1", the remaining states are given the value "-". In the presented example, the following conditions will be created for particular transitions:

[1234]
10--
10-0
010-   (101-)
1010

For the transitions specified with the input vector X2, the following conditions will be created:

[1234]
10-1
1001
-1-0
-001

For the X3 input vector:

[1234]
1-0-
1-00
110-
1100

For the X4 input vector:

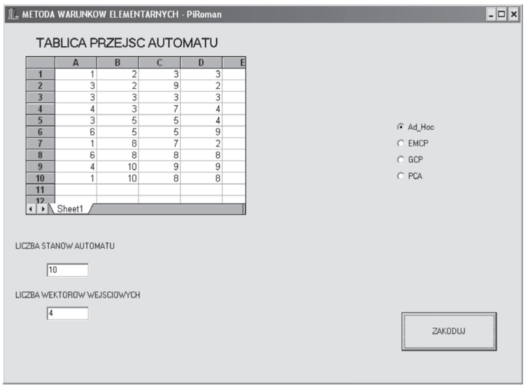
[1234]
1100
11-0
-100
-1-0

Minimization boils down to the combination of non-contradictory conditions, whereby the connection is made for conditions presented as in the above example or for their negation (condition 3 for input X1). The possibility of "sticking" negated conditions results from the freedom to accept "0" or "1" for states assigned to selected transitions.

[1234]
1100
1001
1010

In the asynchronous automaton, whose states will be coded according to minimized conditions, no critical races will occur. In the given example, no-negotiable transitions will be obtained when the following codes are adopted for each state:

State 1 – 111
State 2 – 100
State 3 – 001
State 4 – 010



TABLICA PRZEJSC AUTOMATU

	A	B	C	D	E
1	1	2	3	3	
2	3	2	9	2	
3	3	3	3	3	
4	4	3	7	4	
5	3	5	5	4	
6	6	5	5	9	
7	1	8	7	2	
8	6	8	8	8	
9	4	10	9	9	
10	1	10	8	8	
11					
12					

Sheet1

LICZBA STANOW AUTOMATU: 10

LICZBA WEKTOROW WEJSCIOWYCH: 4

● Ad_Hoc
○ EMCP
○ GCP
○ PCA

Fig. 1. Program for coding asynchronous automata [own study]

2. Reed- Müller polynomials

Polynomial forms can be divided into:

- Reed-Müller polynomials - these are polynomials represented only by AND XOR and NOT operations. With the help of these three functions, each logic function can be represented
- Arithmetic polynomials - these are normal polynomials (arithmetic operations + and -), which return the result 0 or 1 in the case when the variables are 0 or 1 eg f = x1 + x2 - 2 * x1 * x2 is the same as f = x1 XOR x2. Each logical function corresponds to many arithmetic polynomials, and one logical function can be the same for one arithmetic polynomial. Arithmetic polynomial is considered to be better when its coefficients are lower and when its products (components eg x1 * x2 * x3 - product of 3 variables) are smaller.

R-M polynomials consist only of simple variables (a, b, c, etc.) and constant "1" connected by the AND (logical quantity) and XOR (sum of modulo 2). It should be emphasized that there are no negations or brackets in polynomials of R-M. The Reed-Muller system (R-M) is called a full system, i.e. it can present every function. It consists of goals XOR, AND and "1", so the simplest (cheapest) technologically. Thanks to this, he has found a wide application in digital technology. The R-M function shows the following relationships

$$F = \sum_{i=1}^{n^2-1} r^{(i)} \left(x_1\right)^{i_1} \left(x_2\right)^{i_2} \ldots \left(x_n\right)^{t_n} \qquad (1)$$

where:

$$r^{(i)} = 1,0$$

$$\left(x_j\right)^{i_j} = \left| \begin{array}{l} x_j \text{ if } j = 1 \\ 1 \text{ if } j = 0 \end{array} \right.$$

Reed-Müller polynomials have found application in the design of diagnostic trees and synthesis of reversal logic systems. The conversion of functions from the canonical form to the R-M form is done as follows [3, 4]:

1. The given expression is brought to the form of CCF (normal conjunction form) or DCF (normal disjunctive form),
2. Using deMorgan's laws, all alternatives (OR) are changed into conjunctions (AND). The result should be an expression consisting only of conjunction and negation (possibly with parentheses),
3. All negations are transformed into XOR using the formula: ~f=1⊕f
4. The brackets are simplified using the law of separation:

$$x_1 \wedge \left(x_2 \oplus x_3\right) = x_1 x_2 \oplus x_1 x_3 \qquad \textbf{(2)}$$

The R-M polynomial can also be constructed directly from the truth vector using the appropriate transformation matrix f. The matrix designed to transform the truth vector of the function of one variable presents the formula:

$$f_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad \textbf{(3)}$$

To transform functions with more variables, you need larger transformation matrices [2]. The next matrices are constructed from the primary matrix, determined by the formula 3 with the Kronecker product:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} & b \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} \\ c \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} & d \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} \end{bmatrix} = \begin{bmatrix} ae & af & be & bf \\ ag & ah & bg & bh \\ ce & cf & de & df \\ cg & ch & dg & dh \end{bmatrix} \qquad \textbf{(4)}$$

For example, a matrix for transformation of 2-variable functions is constructed by multiplying two basic matrices:

$$f_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad \textbf{(5)}$$

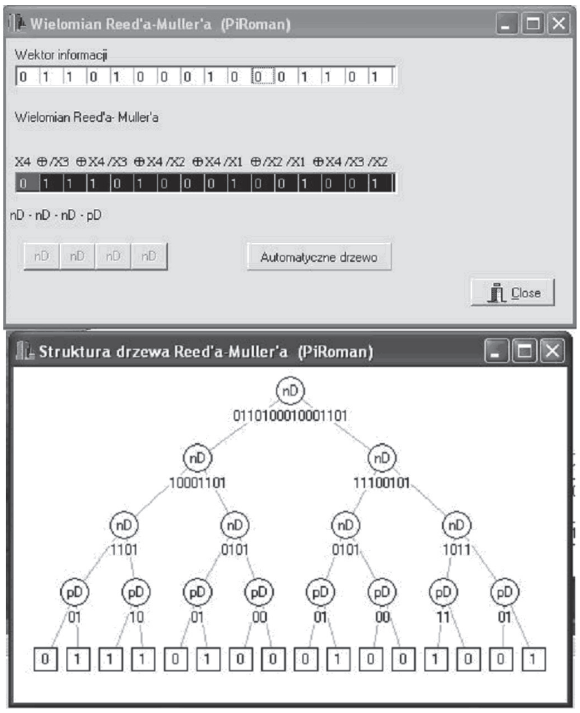
The Reed-Müller W_{RM} vector is obtained from the W_P truth vector by the following transformation:

$$W_{RM} = \left(f \times W_P\right) \bmod 2 \qquad \textbf{(6)}$$

Using the properties that matrices f in modulo 2 arithmetic are their own reversals, the same matrices serve the inverse transformation, i.e. from the form of the Reed-Müller W_{RM} vector to the truth vector W_P:

$$W_P = \left(f \times W_{RM}\right) \bmod 2 \qquad \textbf{(7)}$$

The method presented above was the basis for the author's development of a program for determining Reed-Müller polynomials based on canonical functions. Fig. 2 shows screenshots of the program developed by the author. The program was implemented using the Borland Builder 5.0 environment.



Wielomian Reed'a-Muller'a (PiRoman)

Wektor informacji

0 1 1 0 1 0 0 0 1 0 0 0 1 1 0 1

Wielomian Reed'a- Muller'a

X4 ⊕/X3 ⊕X4 /X3 ⊕X4 /X2 ⊕X4 /X1 ⊕/X2 /X1 ⊕X4 /X3 /X2

0 1 1 1 0 1 0 0 0 1 0 0 1 0 0 1

nD - nD - nD - pD

Struktura drzewa Reed'a-Muller'a (PiRoman)

Fig. 2. Program for generating the Reed-Müller tree [own study]

3. Modeling and simulation in the QUCS program

Software QUCS (Quite Universal Circuit Simulator) is a free electronic circuits simulator. QUCS can simulate both analog and digital circuits. In the first approach, reversing gates were modeled using analogue equivalents. Fig. 3 shows the Fredkin gate model built on the basis of switches implemented in the MOSFET technique[1]. Fig. 4 shows exemplary voltage courses obtained as a result of simulation.

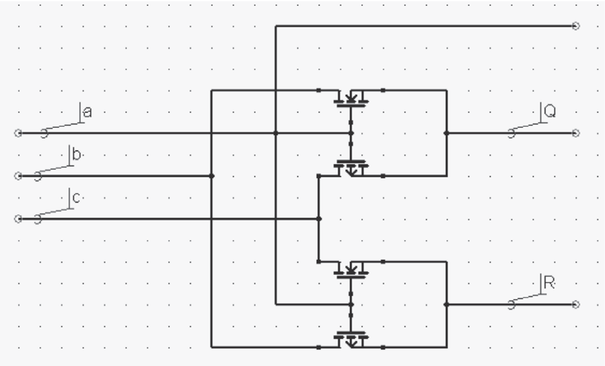


Fig. 3. Fredkin's gate built of MOS transistors [own study]

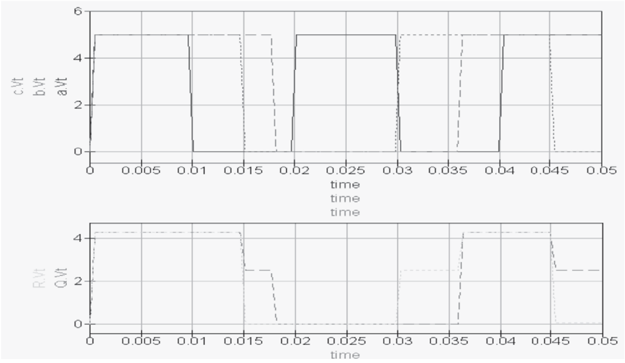


Fig. 4. Simulation results in the QUCS program [own study]

3.1. Reversible gates VHDL models

The Feynman's gate belongs to two-qubit gates. The gate symbol is shown in Fig. 1, Table 1 shows the gateway function. The following is a description of the gateway in the VHDL language [7].

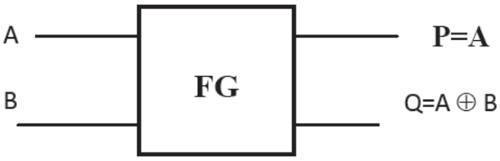


Fig. 5. Feynman Gate [8]

Table 2. Feynman Gate trutch table [own study]

INPUTS		OUTPUTS	
A	B	P	Q
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

```
Library ieee;
Use ieee std_logic.1164..all;
Entity feynmang is
Port(A, B: in std_logic;
   P, Q: out std_logic);
end feynmang;
architecture fmg of feynmang is
begin
P<= A;
Q<= A xor B;
End fmg;
```

The Toffoli gate is a three-qubit quantum gate called the double-controlled negation. The gate symbol is shown in Fig. 2. Table 2. presents the transition function of the gate.

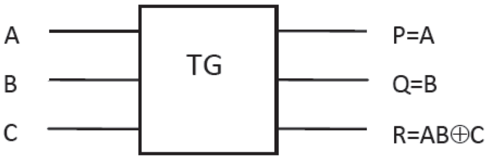


Fig. 6. Toffoli gate [15]

Table 3. Toffoli Gate trutch table [own study]

INPUTS			OUTPUTS		
A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

```
Library ieee;
Use ieee std_logic.1164..all;
Entity toffolig is
Port(A, B, C : in std_logic;
   P, Q, R : out std_logic);
end toffolig;
architecture tg of toffolig is
signal S : std_logic;
begin
P<= A;
Q<= B;
S1<=A and B;
R<= S xor C;
End tg;
```

Unlike the Torffoli gate, which has two control bits and one intentional bit, Fredkin's gate has one control qubit and two target bits. Intentional bits are exchanged if the control bit is equal to 1, otherwise they remain unchanged. In the Table 3 the transitional function of the gate is shown, and its graphical symbol in Fig. 3.

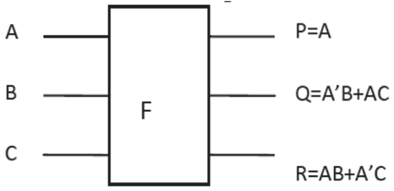


Fig. 7. Fredkin gate [8]

Table 4. Freadkin Gate trutch table [own study]

INPUTS			OUTPUTS		
A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

```
Library ieee;
Use ieee std_logic.1164..all;
Entity fredking is
Port(A, B, C : in std_logic;
   P, Q, R : out std_logic);
end fredking;
architecture fg of fredking is
signal Abar, S1, S2, S3, S4 : std_logic;
begin
P<= A;
```

```
Abar<= not A;
S1<=Abar and B;
S2<= A and C;
Q<= S1 xor S2;
S3<= Abar and C;
S4<= A and B;
R<= S3 xor S4;
End fg;
```

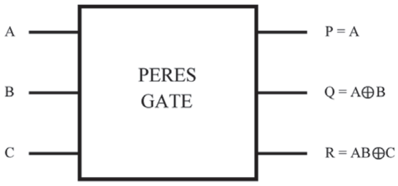


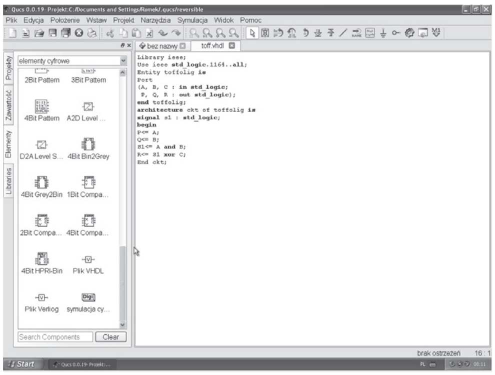
Fig. 8. Peres gate [10]

Table 5. Peres Gate trutch table [own study]

INPUTS			OUTPUTS		
A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

```
library ieee;
use ieee std_logic.1164.all;
entity peres is
port(A,B,C : in std_logic;
P,Q,R :out std_logic);
end peres;
architecture pg of peres is
signal S: std_logic
begin
P <= A;
S <= A and B;
Q <= A xor B;
R <= S xor C;
end pg;
```

Fig. 9 shows how to enter the model into the QUCS program.



```
Library ieee;
Use ieee std_logic.1164..all;
Entity toffolig is
Port
(A, B, C : in std_logic;
 P, Q, R : out std_logic);
end toffolig;
architecture ckt of toffolig is
signal s1 : std_logic;
begin
P<= A;
Q<= B;
S1<= A and B;
R<= S1 xor C;
End ckt;
```

Fig. 9. Toffoli gate in QUCS [own study]

4. Conclusion

Software presented in the article allows shortening the process of designing digital circuits. At the same time, the presented methods enable creating reliable railway automation systems. On the internet you can find programs for verifying reversible logic [16], however the presented method of simulation allows verification of hybrid systems (combining reversible logic with classic logic).

Acknowledgment

This material is based upon work supported by Polish National Center for Research and Development under Grant No. PBS3/A6/29/2015.

Bibliography

[1] DE VOS A.: Reversible Computing in c-MOS, Proc. Advanced Training Course on Mixed Design of VLSI Circuits, 1994, pp. 36-41

[2] DILL K., PERKOWSKI M.: Minimization of Generalized Reed-Muller Forms with a Genetic Algorithm, Proc. Genetic Programming'97 Conf., Stanford University,CA, July 1997, p. 362

[3] DRECHSLER R., et al.: Efficient Representation and Manipulation of Switching Functions Based on Ordered Kronecker Functional Decision Diagrams, Proc. DAC'94, 1994, pp. 415-419

[4] KERNTOPF P., PERKOWSKI M., PODLASKI K.: Synthesis of reversible circuits: A view on the state-of-the-art, Proceedings of the 12th IEEE Conference on Nanotechnology (IEEE-NANO), 2012

[5] KORNASZEWSKI M., CHRZAN M., OLCZYKOWSKI Z.: Implementation of new solutions of intellignet transport systems in railway transport in Poland, in Mikulski J. (ed) Smart Solutions in Today's Transport, Springer Verlag, Berlin Heidelberg, CCIS 715 (2017), pp. 282–292

[6] ŁUKASIK Z., et al.: Assessment of the safety of microprocessor-based semi-automatic block signalling system, Springer-Verlag, Series: Advances in Intelligent Systems and Computing, 2017, pp. 137-144

[7] MAHAMUD A.AL, et al.: Synthesis of Fault Tolerant Reversible Logic Circuits. Proceedings of IEEE International Conference on Testing and Diagnosis, Chengdu, China, 2009, pp. 1-4

[8] MASLOV D., DUECK G. W., MILLER D. M.: Techniques for the synthesis of reversible Toffoli networks, ACM Transactions on Design Automation of Electronic Systems, 12(4)

[9] NOWAKOWSKI W., T. CISZEWSKI T., ŁUKASIK Z.: The concept of railway traffic control systems remote diagnostic, in Mikulski J. (ed) Smart Solutions in Today's Transport, Springer Verlag, Berlin Heidelberg, CCIS 715 (2017) pp. 471-481

[10] PERES A.: Reversible logic and quantum computers. Physical Review A, 1985, pp. 3266-3276

[11] PNIEWSKI R.: Metoda oceny bezpieczeństwa cyfrowych systemów automatyki kolejowej. Monografie Wydawnictwo UTH Radom

[12] PNIEWSKI R., BOJARCZAK P., KORNASZEWSKI M.: Application of Reversible Logic in Synthesis of Traffic Control Systems, in Mikulski J. (ed) Smart Solutions in Today's Transport, Springer Verlag, Berlin Heidelberg, CCIS 715 (2017), pp 447-460

[13] PNIEWSKI R., KORNASZEWSKI M., CHRZAN M.: Safety of electronic ATC systems in the aspect of technical and operational. 16th International Scientific Conference Globalization and Its Socio-Economic Consequences. Proceedings, Part IV. University of Zilina, The Faculty of Operation and Economics of Transport and Communications, Department of Economics, Rajecke Teplice, Slovak Republic, October 2016, pp. 1729-1735

[14] TAHA S.M.R.: Reversible Logic Synthesis Methodologies with Application to Quantum Computing. Springer International Publishing Switzerland 2016

[15] TOFFOLI T.: Reversible computing, Tech Memo MIT/LCS/TM-151, MIT Lab for Computer Science, 1980

[16] http://www.revkit.org/ (the revkit program website) [date of access: 20.01.2018]