Circuits and Systems, **2013, 4, 472-477**
Published Online November 2013 (http://www.scirp.org/journal/cs)
http://dx.doi.org/10.4236/cs.2013.47062

Scientific Research

Logical Function Decomposition Method for Synthesis of Digital Logical System Implemented with Programmable Logic Devices (PLD)

Mihai Grigore Timis, Alexandru Valachi, Alexandru Barleanu, Andrei Stan
Automatic Control and Computer Engineering Faculty, Technical University Gh.Asachi, Iasi, Romania
Email: mtimis@tuiasi.ro, avalachi@tuiasi.ro, abarleanu@tuiasi.ro, andreis@tuiasi.ro

Received August 18, 2013; revised September 18, 2013; accepted September 26, 2013

ABSTRACT

The paper consists in the use of some logical functions decomposition algorithms with application in the implementation of classical circuits like SSI, MSI and PLD. The decomposition methods use the Boolean matrix calculation. It is calculated the implementation costs emphasizing the most economical solutions. One important aspect of serial decomposition is the task of selecting "best candidate" variables for the G function. Decomposition is essentially a process of substituting two or more input variables with a lesser number of new variables. This substitutes results in the reduction of the number of rows in the truth table. Hence, we look for variables which are most likely to reduce the number of rows in the truth table as a result of decomposition. Let us consider an input variable purposely avoiding all inter-relationships among the input variables. The only available parameter to evaluate its activity is the number of "l"s or "O"s that it has in the truth table. If the variable has only "1" s or "0" s, it is the "best candidate" for decomposition, as it is practically redundant.

Keywords: Combinational Circuits; Static Hazard; Logic Design; Boolean Functions; Logical Decompositions

1. Introduction

In the implementation of logical functions we are looking to optimize some parameters such as the propagation time, cost, areas, power, etc. The decomposition problem is old, and well understood when the function to be decomposed is specified by a truth table, or has one output only. However, modern design tools handle functions with many outputs and represent them by cubes, for reasons of efficiency. We develop a comprehensive theory of serial decompositions for multiple-output, partially specified, Boolean functions. A function $f(x_1,\cdots,x_n)$ has a serial decomposition if it can be expressed as $h(u_1,\cdots,u_r,g(v_1,\cdots,v_s))$, where $U=\{u_1,\cdots,u_r\}$ and $V=\{v_1,\cdots,v_s\}$ are subsets of the set $X=\{x_1,\cdots,x_n\}$ of input variables, and g and h have fewer inputvariables than f.

It is sometimes the case that a set of Boolean functions cannot be made to fit into any single module intended for its implementation. The only solution is to decompose the problem in such a way that the requirement can be met by a network of two or more components each implementing a part of the functions. The general problem can be stated as follows. The set of functions to be implemented quires a logic block with N inputs and M outputs. The decomposition task is to design a network which will implement the function using blocks with a maximum of n inputs and m outputs, where $n < N$ or $m < M$.

(*A*) *Initially, we will consider a decomposition algorithm of logical functions* [1].

1.1. Given a Boolean function $f(x_{n-1},\cdots,x_1,x_0)$ and p Boolean functions denoted by $\varphi_{p-1}(y_{i-1},\cdots,y_1,y_0),\cdots,\varphi_0(y_{i-1},\cdots,y_1,y_0)$, it is possible to decompose the function f depending on $\varphi_{p-1},\cdots,\varphi_0$? In other words, there is a function F so that $F(\varphi_{p-1},\cdots,\varphi_0;z_{n-1},\cdots,z_i)=f(x_{n-1},\cdots,x_0)$, where $Y=\{y_{i-1},\cdots,y_0\}$ and $Z=\{z_{n-1},\cdots,z_i\}$ are disjoint subsets of the set $X=\{x_{n-1},\cdots,x_0\}$, that means

$$X=Y\bigcup Z \quad \text{and} \quad Y\bigcap Z=\phi \quad (1.1)$$ (the empty set).

We will call this proceeding, Type I problem.

1.2. Given a Boolean function $f(x_{n-1},\cdots,x_1,x_0)$ there are q functions denoted by $\varphi_{q-1}(y_{i-1},\cdots,y_1,y_0),\cdots,\varphi_0(y_{i-1},\cdots,y_1,y_0)$ and a function F so that

$F\left(\varphi_{q-1},\cdots,\varphi_0;z_{n-1},\cdots,z_i\right)=f\left(x_{n-1},\cdots,x_0\right)$, where $Y=\left\{y_{i-1},\cdots,y_0\right\}$ and $Z=\left\{z_{n-1},\cdots,z_i\right\}$ have the same meaning as in 1.1. We will call this proceeding, the type II problem.

(*B*) *Matrices related to Boolean functions. The image of a logical function* [1]

It defines the image of a logical function the Boolean row array that represents the values of this function, ordered by truth table.

For example, $f\left(x_2,x_1,x_0\right)=R_1\left(0,1,3,5,7\right)$ has the following truth table:

(dec.echiv.)	x_2	x_1	x_0	f
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Considering the above, we can write

$$\div f=11010101 \tag{1.2}$$

We can verify the following properties:

$$\begin{aligned}\div\left(f_1\cdot f_2\right)&=\left(\div f_1\right)\cdot\left(\div f_2\right)\\ \div\left(f_1+f_2\right)&=\left(\div f_1\right)+\left(\div f_2\right)\end{aligned} \tag{1.3}$$

To a function can be attached a Veitch matrix, for the previous case being:

$$E=\begin{matrix}x_2\setminus x_1x_0\\ \begin{bmatrix}1101\\0101\end{bmatrix}\end{matrix} \tag{1.4}$$

2. The Representation of a Boolean Function Using Subfunctions. The R_{JI} Matrix

Let's consider a function G of two subfunctions f_1 and f_0 that depend on the Boolean variables x_2,x_1,x_0 and on the two variables x_4,x_3:

$$G\left(x_4,x_3,f_1,f_0\right)=f_0\cdot\overline{x_4}\cdot x_3+\overline{f_1}\cdot x_4\cdot\overline{x_3}+\overline{f_1}\cdot f_0\cdot x_4 \tag{2.1}$$

After a simple calculation is deduced the image of function G.

$$\div G=0000010111000100 \tag{2.2}$$

We suppose that the images of the two subfunctions are:

$$\begin{aligned}\div f_1&=01110100\\ \div f_0&=01010011\end{aligned} \tag{2.3}$$

that means:

$$\begin{aligned}f_1&=\overline{x_2}\cdot x_1+\overline{x_1}\cdot x_0\\ f_0&=\overline{x_2}\cdot x_0+x_2\cdot x_1\end{aligned} \tag{2.4}$$

Starting from the expressions of G, f_1 and f_0 can be calculated:

$$\begin{aligned}&F\left(x_4,x_3,x_2,x_1,x_0\right)\\ &=G\left(x_4,x_3,f_1\left(x_2,x_1,x_0\right),f_0\left(x_2,x_1,x_0\right)\right)\\ &=\overline{x_4}\cdot x_3\cdot\overline{x_2}\cdot x_0+\overline{x_4}\cdot x_3\cdot x_2\cdot x_1\\ &\quad+x_4\cdot\overline{x_3}\cdot x_2\cdot\overline{x_0}+x_4\cdot\overline{x_3}\cdot\overline{x_1}\cdot\overline{x_0}+x_4\cdot x_2\cdot x_1\end{aligned} \tag{2.5}$$

The image of function F is calculated below:

$$\div F=00000000101001110001011000000011 \tag{2.6}$$

The Veitch tables $E'_{x_4x_3:f_1f_0}$ and $E_{x_4x_3:x_2x_1x_0}$ relating to the G and F functions are:

$$E'=\begin{matrix}x_4x_3\setminus f_1f_0\\ \begin{bmatrix}0000\\0101\\1100\\0100\end{bmatrix}\end{matrix},\qquad E=\begin{matrix}x_4x_3\setminus x_2x_1x_0\\ \begin{bmatrix}00000000\\01010011\\10001011\\00000011\end{bmatrix}\end{matrix} \tag{2.7}$$

Note that the E' matrix has only four distinct columns that are found in E matrix.

In [1], it demonstrates that for the function F it can be attached a pseudo-unitary matrix denoted by R_{JI} in which in each column the logic digit 1 corresponds to the E' column's order number, therefore:

$$R_{JI}=\begin{matrix}\setminus x_2x_1x_0\\ \begin{bmatrix}10001000\\00000011\\00100100\\01010000\end{bmatrix}\end{matrix} \tag{2.8}$$

In [1] is also demonstrated the relation:

$$E=E'\otimes R_{JI}\quad(\otimes\text{—the matrix multiplication}) \tag{2.9}$$

Therefore, the decomposition of a function in subfunctions is reduced to solving the following Boolean equations:

$X\otimes A=B$ (2.10) (the type I problem, where the E and R_{JI} matrices are known) or $A\otimes X=B$ (2.11) (the type II problem, where only the B matrix is known, $B=E$).

Considering that the columns of matrix E' are found in matrix E it deduces the matrix $A=E'$, and then the matrix R_{JI}.

Next, we present the solutions of the equations (2.10)

and (2.11), demonstrated in [1].

A. The solution of the equation $A \otimes X = B$ [1]

a) Let's consider X a some matrix. It is valid the relation (2.12), [1].

$$X \le \overline{t_A \otimes \overline{B}} \quad (t_A\text{-the transpose of the matrix } A) \tag{2.12}$$

b) Let's consider X a pseudo-unitary matrix. It is valid the relation (2.13) [1].

$$X \le \overline{t_A \otimes \overline{B}} \cdot \overline{t_{\overline{A}} \otimes B} \tag{2.13}$$

B. The solution of the equation $X \otimes A = B$ [1]

a) Let's consider A a some matrix. It is assumed [1]:

$$X \le \overline{\overline{B} \otimes t_A} \tag{2.14}$$

b) Let's consider A a pseudo-unitary matrix. It is denoted by $X_c = B \otimes t_A$ and $X_a = \overline{\overline{B} \otimes t_A}$. The sufficient condition of existence of the solution [1] is:

$$X_c = B \otimes t_A \le X_a = \overline{\overline{B} \otimes t_A} \text{ and } X_c \le X \le X_a \tag{2.15}$$

If $X_c > X_a$ or $X_c \ne X_a$, there is no solution for the matrix X. In this case it is trying to solve the following equations:

$$\begin{aligned} &F(x_4, \cdots, x_0) \\ &= G_1(x_4, x_3, f_1, f_0) \\ &\quad + H_1(x_4, x_3, x_2, x_1, x_0) \end{aligned} \tag{2.16}$$

(the previous solution) or

$$\begin{aligned} &F(x_4, \cdots, x_0) \\ &= G_2(x_4, x_3, f_1, f_0) \\ &\quad \times H_2(x_4, x_3, x_2, x_1, x_0) \end{aligned} \tag{2.17}$$

(the consequence solution). We will return to these problems in a future paper.

3. Examples

(*A*) *Let's consider the function defined by*

$$\begin{aligned} &F(x_4, x_3, x_2, x_1, x_0) \\ &= R_1(0,3,5,6,9,10,12,14,15,16,17,18,19,21,22,30) \end{aligned} \tag{3.1}$$

Applying the Veitch-Karnaugh method [2], a minimal form is given by the expression:

$$\begin{aligned} F(x_4, x_3, x_2, x_1, x_0) &= \overline{x_4} \cdot x_3 \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0 \\ &+ \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0} + \overline{x_3} \cdot \overline{x_2} \cdot x_1 \cdot x_0 + \overline{x_4} \cdot x_3 \cdot x_1 \cdot \overline{x_0} \\ &+ \overline{x_4} \cdot x_3 \cdot x_2 \cdot \overline{x_0} + \overline{x_4} \cdot x_3 \cdot x_2 \cdot x_1 + x_4 \cdot \overline{x_3} \cdot \overline{x_2} \\ &+ x_2 \cdot x_1 \cdot \overline{x_0} + \overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot x_0 \end{aligned} \tag{3.2}$$

We define the cost of implementation as the number of the inputs in the basic circuits, components [3]. In the previous case, by implementing with AND-OR circuits, results: $C_1(F) = (5 + 6 \cdot 4 + 2 \cdot 3) + 9 = 44$. (It is considering that the input variables are provided inverted and non-inverted, *i.e.* $x_i, \overline{x_i}$.)

Let's consider the following possible decomposition:

$$G(x_4, x_3, f_1, f_0) = F(x_4, x_3, x_2, x_1, x_0)$$

where $f_1 = f_1(x_2, x_1, x_0)$, $f_0 = f_0(x_2, x_1, x_0)$.

For the function F corresponds the following Veitch matrix, denoted by E:

$$E = \begin{bmatrix} 10010110 \\ 01101011 \\ 11110110 \\ 00000010 \end{bmatrix} \tag{3.3}$$

Matrix E having four distinct columns, a solution for E' is:

$$E' = \begin{matrix} x_4x_3 \setminus f_1f_0 \\ \begin{bmatrix} 1001 \\ 0111 \\ 1101 \\ 0001 \end{bmatrix} \end{matrix} \tag{3.4}$$

Therefore, the matrix R_{JI}, solution of the equation $E' \otimes R_{JI} = E$, is:

$$R_{JI} \le \overline{t_{E'} \otimes \overline{E}} \cdot \overline{t_{\overline{E'}} \otimes E} = \begin{bmatrix} 10010100 \\ 01100000 \\ 00001001 \\ 00000010 \end{bmatrix} \tag{3.5}$$

From where we obtain:

$$\begin{aligned} \div f_1 &= 00001011 \\ \div f_0 &= 01100010 \end{aligned} \tag{3.6}$$

or after an elementary calculation:

$$\begin{aligned} f_1 &= x_2 \cdot (x_1 + \overline{x_0}) \\ f_0 &= x_1 \cdot \overline{x_0} + \overline{x_2} \cdot \overline{x_1} \cdot x_0 \end{aligned} \tag{3.7}$$

Using E' matrix we obtain:

$$\begin{aligned} G &= \overline{f_1} \cdot \overline{f_0} \cdot \overline{x_3} + f_1 \cdot f_0 + x_4 \cdot \overline{x_3} \cdot f_0 \\ &+ \overline{x_4} \cdot x_3 \cdot f_0 + f_1 \cdot \overline{x_4} \cdot x_3 \end{aligned} \tag{3.8}$$

with a possible implementation as in **Figure 1**.

So, we will have:

$$C(f_1) = 7 \quad C(f_0) = 8 \tag{3.9}$$

$C(G) = (4 \times 3 + 2) + 5 = 19$, so that $C(F) = 34$.

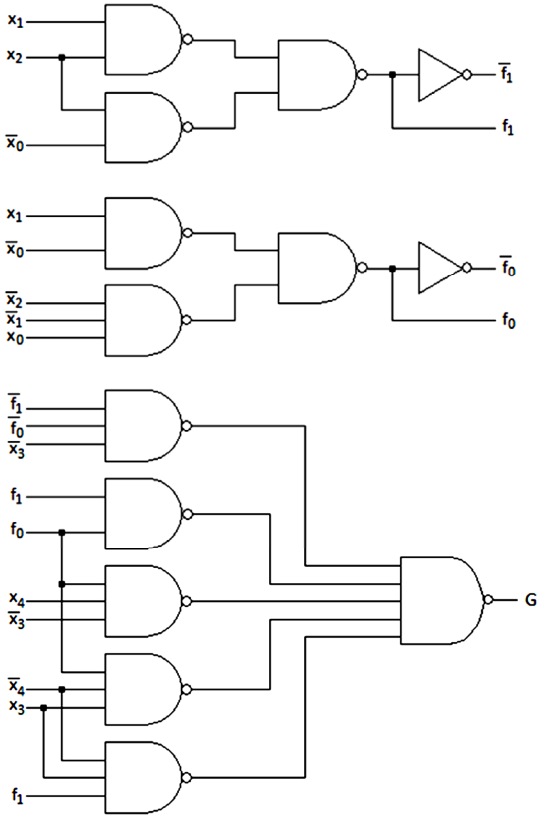

Figure 1. The implementation of the function F, using sub-functions.

(*B*) *Implementation using programmable logic devices* (*PLD*)

We will consider a circuit PAL10L8 [4], which has 10 inputs, 8 outputs and having an AND-OR configuration, each NOR having 2 inputs, with the structure illustrated in **Figure 2**.

Let's consider the previous function:

$$F = \sum_{i=0}^{8} m_i ,$$

where

$$\begin{aligned}
m_0 &= \overline{x_4} \cdot x_3 \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0 \\
m_1 &= \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0} \\
m_2 &= x_3 \cdot \overline{x_2} \cdot x_1 \cdot x_0 \\
m_3 &= \overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot x_0 \\
m_4 &= \overline{x_4} \cdot x_3 \cdot x_1 \cdot \overline{x_0} \\
m_5 &= \overline{x_4} \cdot x_3 \cdot x_2 \cdot \overline{x_0} \\
m_6 &= \overline{x_4} \cdot x_3 \cdot x_2 \cdot x_1 \\
m_7 &= x_4 \cdot \overline{x_3} \cdot \overline{x_2} \\
m_8 &= x_2 \cdot x_1 \cdot \overline{x_0}
\end{aligned} \qquad (3.10)$$

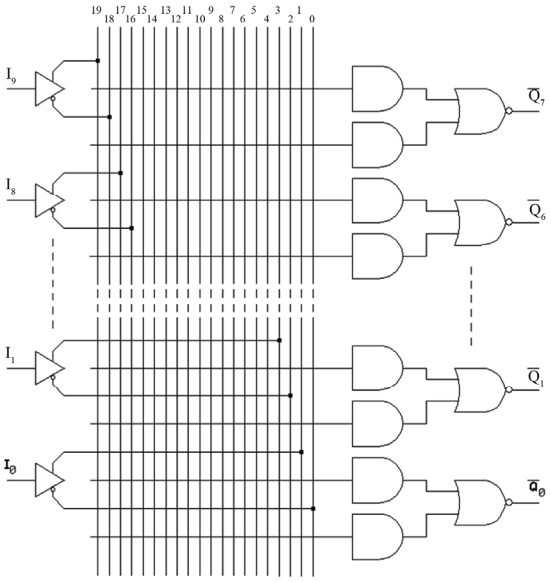

Figure 2. The circuit PAL10L8.

We will use the following algorithm:

$$\begin{aligned}
Q_0 &= m_0 + m_1 \\
Q_1 &= Q_0 + m_2 \\
&\vdots \\
Q_7 &= Q_6 + m_8 = F
\end{aligned} \qquad (3.11)$$

Therefore, $Q_i = Q_{i-1} + m_{i+1}$ with $Q_{-1} = m_0$, $0 \le i \le 7$.

Will be needed: 9—product terms $(m_0 \div m_8)$, $7 - Q_i$ terms $(0 \le i \le 6)$, so it will be used 16 product terms from maximum 20. But the number of inputs is insufficient (see **Figure 3**).

Classic, we should also use two circuits (PAL10L8), or a single circuit with greater capacity.

Let go back to the same function that uses the subfunctions f_1, f_0, which have the expressions:

$$\begin{aligned}
f_1 &= x_2 \cdot x_1 + x_2 \cdot \overline{x_0} \\
f_0 &= x_1 \cdot \overline{x_0} + \overline{x_2} \cdot \overline{x_1} \cdot x_0
\end{aligned}$$

and

$$\begin{aligned}
&F(x_4, x_3, x_2, x_1, x_0) \\
&= G(x_4, x_3, f_1, f_0) \\
&= \overline{f_1} \cdot \overline{f_0} \cdot x_3 + f_1 \cdot f_0 + x_1 \cdot \overline{x_3} \cdot f_0 \\
&\quad + \overline{x_4} \cdot x_3 \cdot f_0 + f_1 \cdot \overline{x_4} \cdot x_3
\end{aligned} \qquad (3.12)$$

Therefore, after a preliminary evaluation we have: 4 product terms (f_1, f_0) and 5 product terms for function G.

Let's consider $G = (a_0 + a_1) + (a_2 + a_3) + a_4$, where a_i are the terms of the decomposed function. A PAL implementation is like in **Figures 3** and **4**.

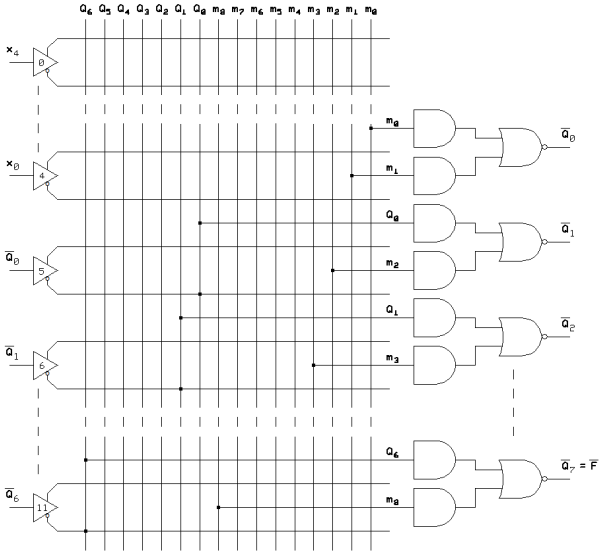

Figure 3. PAL implementation.

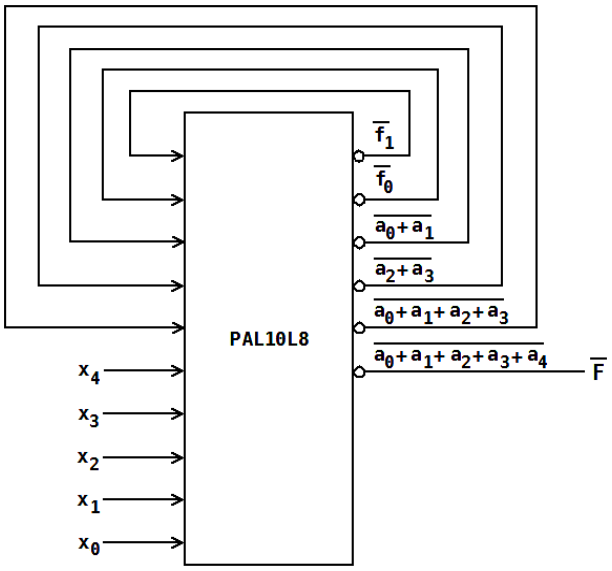

Figure 4. A possible implementation of PAL10L8 circuit.

A possible implementation would be (see **Figure 4**):

4. Decomposition into EMB Blocks

The single step of the functional decomposition replaces function F with two subfunctions [5]. This process is recursively applied to both the G and H blocks until a network is constructed where each block can be directly implemented in single logic cell of target FPGA architecture.

Logic cell can implement any function of limited input variables (typically 4 or 5). Thus the main effort of logic synthesis methods based on decomposition is to find such partition of input variables into free set and bound set that allows creating decomposition with block G not exceeding the size of logic cell. Various methods are used, including exhaustive search since the size of logic cell is small. It should be noted that the main constraint is the number of inputs to block G and not the number of outputs. This is because block G with more outputs than in logic cell can be implemented with use of few logic cells used in parallel. Since EMB blocks can be configured to work as logic cell of many different sizes [6], approach known from methods targeted for logic cells is not efficient. The main reason is that the method must check decomposition for many different sizes of block G. The second factor is that in case of EMB the efficiency of utilization of these blocks depends on carefully selected size of block G. For example M512 RAM block of Stratix device can be configured among others as 8 input and 2 output logic cell or 7 input and 4 output logic cell. Let assume that in decomposition search following solutions are possible: block G with 8 inputs and 1 output or block G with 7 inputs and 3 outputs. From the EMB utilization point of view the second solution is better, since it utilizes 384 bits of total 512 bits available, while the first solution utilizes only 256 bits. R-admissibility is used to evaluate serial decomposition possibilities for different sizes of G block according to possible configuration of EMB blocks. Since EMB can be configured as a block of many different sizes the possible solution space is large. Using Property 1 the search can be drastically reduced. This will be explained in the following example.

Example.

R-admissibility application to serial decomposition evaluation. For function from Example 1 we have that the admissibility of single input variables $x_1,\cdots,x_6$ is accordingly 4, 4, 4, 3, 3 and 4. This means that only for $U=\{x_4\}$, $V=\{x_1,x_2,x_3,x_5,x_6\}$ and $U=\{x_5\}$, $V=\{x_1,x_2,x_3,x_4,x_6\}$ decomposition with 2 outputs from block G may exist.

When considering solutions with 4 inputs to block G, according to Property 1, [7,8] only solution with $U=\{x_4,x_5\}$, $V=\{x_1,x_2,x_3,x_6\}$ should be evaluated. We have:

$$\left(\beta_{x_4}\cdot\beta_{x_5}\right) \text{ or } \beta_F=\left\{\overline{(4)(8)};\overline{(1)(6)};\overline{(2)(3)};\overline{(5)(7)}\right\}$$

$$r\left(\beta_{x_4}\cdot\beta_{x_5}\right)=2+e\left(\left(\beta_{x_4}\cdot\beta_{x_5}\right) \text{ or } \beta_F\right)=2+\left[\log_2 2\right]=3 \tag{4.1}$$

This means that for such variable partitioning decomposition may exist with block G having 1 output. With this approach to serial decomposition, there is no difference between disjoint and non-disjoint decomposition in their calculation. Particularly, it can be concluded that for finding blanket G we can simply apply the method of calculating compatible classes of βV blocks [7] which was recently improved in [8].

5. Conclusions

The paper represents the "rediscovery" of some decomposition algorithms of Boolean logic functions, using subfunctions [1].

After a brief exposure of the decomposition methods of Boolean logical functions, the authors, through the proposed example, shows the reduction of the implementation cost using standard logical circuits.

The authors show that when using PLD circuits, the use of Boolean functions decomposition method reduces the number of circuits necessary for the implementation (see PAL10L8).

Balanced decomposition proved to be very useful in implementation of combinational functions using logic cell resources of FPGA architectures. However, results presented in this paper show that functional decomposition can be efficiently and effectively applied also to implement digital systems in embedded memory blocks. Application of r-admissibility concept makes possible fast evaluation of decompositions for different sizes of block G. This allows selecting best possible decomposition strategy.

The paper showed that the use of Boolean functions decomposition method reduces the number of circuits necessary for the implementation. However, this substitution process reduces the circuits cost by increasing the circuit complexity, which also enhances the likelihood of errors in the circuit design.

Balanced decomposition proved to be very useful in implementation of combinational functions using logic cell resources of FPGA architectures. However, results presented in this paper show that functional decomposition can be efficiently and effectively applied also to implement digital systems in embedded memory blocks.

REFERENCES

[1] M. Denouette, J. P. Perrin and E. Daclin, "Systemès Logiques," Tome I, Dunod, Paris, 1967, pp. 4-56.

[2] C. H. Roth, "Fundamentals of Logic Design," West Publishing Company, Eagan, 1999, pp. 148-172.

[3] Al. Valachi, Fl. Hoza, V. Onofrei and R. Silion, "Analiza, Sinteza şi Testarea Dispozitivelor Numerice," Nord-Est, 1993, pp. 31-32; 45-53.

[4] J. A. Brzozowski and T. Łuba, "Decomposition of Boolean Functions Specified by Cubes," *Journal of Multiple-Valued Logic and Soft Computing*, Vol. 9, 2003.

[5] M. Rawski, "Decomposition of Boolean Function Sets," *Electronics and Telecommunications Quarterly*, Vol. 53, No. 3, 2007, pp. 231-249.

[6] J. A. Brzozowski and T. Łuba, "Logic Decomposition Aimed at Programmable Cell Arrays," *International Conference of Microelectronics*: *Microelectronics*, Vol. 1783, 1992, pp. 77-88. http://dx.doi.org/10.1117/12.130993

[7] S. J. E. Wilton, "SMAP: Heterogeneous Technology Mapping for Area Reduction in FPGAs with Embedded Memory Arrays," *FPGA*, 1998, pp. 171-178.

[8] "Logic Synthesis Strategy for FPGAs with Embedded Memory Blocks," Mariusz Rawski, Grzegorz Borowik, Tadeusz Łuba, Paweł Tomaszewicz, Bogdan j. Falkow- ski. Przegląd Elektrotechnic Znyelectrical Review), R. 86 NR 11a/2010.