SYNTHESIS OF DIGITAL SYSTEMS IMPLEMENTED WITH PROGRAMMABLE LOGIC DEVICES, USING DECOMPOSITION ALGORITHMS

BY

ALEXANDRU VALACHI, MIHAI TIMIŞ*, ALEXANDRU BÂRLEANU and ANDREI STAN

“Gheorghe Asachi” Technical University of Iași,
Faculty of Automatic Control and Computer Engineering

Received: May 22, 2013
Accepted for publication: June 27, 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 matrices calculation. It is calculated the implementation costs emphasizing the most economical solutions. 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 \( f(x_1, \ldots, x_n) \) has a serial decomposition if it can be expressed as \( h(u_1, \ldots, u_r, g(v_1, \ldots, v_s)) \), where \( U = \{u_1, \ldots, u_r\} \) and \( V = \{v_1, \ldots, v_s\} \) are subsets of the set \( X = \{x_1, \ldots, x_n\} \) of input variables, and \( g \) and \( h \) have fewer input variables than \( f \).

Key words: decomposition algorithms, Boolean functions, programmable logic devices.

2010 Mathematics Subject Classification: 06E30, 49M27, 94C05.

*Corresponding author; e-mail: mtimis@tuiasi.ro
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 (Brzozowski & Łuba, 2003; Łuba & Selvaraj, 2012; Denouette et al., 1967). 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 \( f(x_1, \ldots, x_n) \) has a serial decomposition if it can be expressed as \( h(u_1, \ldots, u_r, g(v_1, \ldots, v_s)) \), where \( U = \{u_1, \ldots, u_r\} \) and \( V = \{v_1, \ldots, v_s\} \) are subsets of the set \( X = \{x_1, \ldots, x_n\} \) of input variables, and \( g \) and \( h \) have fewer input variables 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 requires 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 \( m < M \).

Initially, we will consider a decomposition algorithm of logical functions, as:

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

\[
X = Y \cup Z \quad \text{and} \quad Y \cap Z = \emptyset \quad \text{(the empty set)} \tag{1}
\]

We will call this proceeding, the type I problem.

b) Given a Boolean function \( f(x_{n-1}, \ldots, x_1, x_0) \) there are \( q \) functions denoted by \( \varphi_{q-1}(y_{n-1}, \ldots, y_1, y_0), \ldots, \varphi_0(y_{n-1}, \ldots, y_1, y_0) \) and a function \( F \) so that \( F(\varphi_{q-1}, \ldots, \varphi_0; z_{n-1}, \ldots, z_i) = f(x_{n-1}, \ldots, x_0) \), where \( Y = \{y_{i-1}, \ldots, y_0\} \) and \( Z = \{z_{n-1}, \ldots, z_i\} \) have the same meaning as in 1. We will call this proceeding, the type II problem.
2. Matrices Related to Boolean Functions. The Image of a Logical Function

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, \( f(x_2, x_1, x_0) = R_4(0,1,3,5,7) \) has the following truth table, table:

\[
\begin{array}{cccc}
\text{(dec.echiv.)} & x_2 & x_1 & x_0 & f \\
0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 \\
2 & 0 & 1 & 0 & 0 \\
3 & 0 & 1 & 1 & 1 \\
4 & 1 & 0 & 0 & 0 \\
5 & 1 & 0 & 1 & 1 \\
6 & 1 & 1 & 0 & 0 \\
7 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Considering the above, we can write:

\[
\div f = 11010101
\]

We can verify the following properties:

\[
\begin{align*}
\div (f_1 \cdot f_2) &= (\div f_1) \cdot (\div f_2) \\
\div (f_1 + f_2) &= (\div f_1) + (\div f_2)
\end{align*}
\]

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

\[
E = \begin{bmatrix} 1101 \\ 0101 \end{bmatrix}
\]

The Representation of a Boolean Function Using Subfunctions. RJI Matrix

Let’s consider a function \( G \) of two subfunctions \( f_i \) and \( f_0 \) that depend on the Boolean variables \( x_2, x_1, x_0 \) and on the two variables \( x_4, x_3 \):
After a simple calculation the image of function $G$ is deduced.

$$\div G = 0000010111 \ 000100$$

We suppose that the images of the two subfunctions are:

$$\div f_1 = 01110100$$
$$\div f_0 = 01010011$$

that means:

$$f_1 = x_2 \cdot x_1 + x_1 \cdot x_0$$
$$f_0 = x_2 \cdot x_0 + x_2 \cdot x_1$$

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

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

The image of function $F$ is calculated below:

$$\div F = 0000000001 \ 0100111000 \ 1011000000 \ 11$$

The Veitch tables $E'_{x_4,x_3,f_1,f_0}$ and $E'_{x_4,x_3,f_2,f_0}$ relating to the $G$ and $F$ functions are:

$$E' = \begin{bmatrix}
0000 \\
0101 \\
1100 \\
0100
\end{bmatrix}, \quad E = \begin{bmatrix}
00000000 \\
01010011 \\
10001011 \\
00000011
\end{bmatrix}$$

Note that the $E'$ matrix has only four distinct columns that are found in $E$ matrix. It is demonstrated, (Rawski, 2007; Rawski et al., 2010; Rawski, 2012) 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:
It is also demonstrated the relation:

\[ E = E' \otimes R_f, \quad (\otimes - \text{the matrix multiplication}) \tag{13} \]

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

\[ X \otimes A = B \quad (\text{the type I problem, where the } E \text{ and } R_f \text{ matrices are known}) \tag{14} \]

or

\[ A \otimes X = B \quad (\text{the type II problem, where only the } B \text{ matrix is known, } B = E) \tag{15} \]

Considering that the columns of matrix \( E' \) are found in matrix \( E \) it deduces the matrix \( A = E' \), and then the matrix \( R_f \).

Next, we present the solutions of the eqs. (14) and (15).

A. The solution of the equation \( A \otimes X = B \)
   a) Let’s consider a matrix \( X \). It is assumed:

\[ X \leq t_A \otimes B \quad (t_A - \text{the transpose of the matrix } A) \tag{16} \]

b) Let’s consider \( X \) a pseudo-unitary matrix. It is valid the relation (17).

\[ X \leq t_A \otimes B \cdot t_A \otimes B \tag{17} \]

B. The solution of the equation \( X \otimes A = B \)
   a) Let’s consider a some matrix \( A \). It is assumed:

\[ X \leq B \otimes t_A \tag{18} \]

b) Let’s consider a pseudo-unitary matrix \( A \). It is denoted by \( X_e = B \otimes t_A \) and \( X_a = \overline{B} \otimes t_A \). The sufficient condition of existence of the solution is:
\[ X_c = B \otimes t_d \leq X_a = \overline{B} \otimes t_d \] and \[ X_c = X \leq X_a \] (19)

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

\[ F(x_4, ..., x_0) = G_1(x_4, x_3, f_1, f_0) + H_1(x_4, x_3, x_2, x_1, x_0) \] (20)

or

\[ F(x_4, ..., x_0) = G_2(x_4, x_3, f_1, f_0) \cdot H_2(x_4, x_3, x_2, x_1, x_0) \] (21)

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

3. Examples

3.1. Implementation Using Subfunctions

Let’s consider the function defined by

\[ F(x_4, x_3, x_2, x_1, x_0) = \]

\[ = R(0, 3, 5, 6, 9, 10, 12, 14, 15, 16, 17, 18, 19, 21, 22, 30) \] (22)

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

\[ F(x_4, x_3, x_2, x_1, x_0) = x_4 \cdot x_1 \cdot \overline{x_2} \cdot x_0 + \]

\[ + x_1 \cdot x_2 \cdot x_0 + x_3 \cdot x_2 \cdot x_1 \cdot x_0 + x_4 \cdot x_3 \cdot x_1 \cdot x_0 + \]

\[ + x_1 \cdot x_0 + x_2 \cdot x_1 \cdot x_0 \]

\[ (23) \]

We define the cost of implementation as the number of the inputs in the basic circuits, components, (Valachi et al., 2003). In the previous case, by implementing with AND - OR circuits, it results: \( C_i(F) = (5 + 6 \cdot 4 + 2 \cdot 3) + 9 = 44 \). (It is considered 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

\[ f_1 = f_1(x_2, x_1, x_0), \quad f_0 = f_0(x_2, x_1, x_0) \] (24)

For the function \( F \), the corresponding Veitch matrix is denoted by \( E \):
Matrix $E$ having four distinct columns, a solution for $E'$ is:

$$x_4 x_3 \setminus f_1 f_0$$

$$E' = \begin{bmatrix} 1001 \\ 0111 \\ 1101 \\ 0001 \end{bmatrix}$$

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

$$R_{ji} = t_{E'} \otimes E \cdot t_{E'} \otimes E = \begin{bmatrix} 10010100 \\ 01100000 \\ 00001001 \\ 00000010 \end{bmatrix}$$

From where we obtain:

$$\div f_1 = 00001011$$
$$\div f_0 = 01100010$$

(26)

or after an elementary calculation:

$$f_1 = x_2 \cdot (x_1 \div x_0)$$
$$f_0 = x_1 \div x_0 + x_2 \cdot x_1 \cdot x_0$$

(27)

Using $E'$ matrix we obtain:

$$G = f_1 \cdot f_0 \cdot x_5 + f_1 \cdot f_0 + x_4 \cdot \overline{x_3} \cdot f_0 + x_4 \cdot x_3 \cdot f_0 + f_1 \cdot \overline{x_0} \cdot x_3$$

(28)

with a possible implementation as it is presented in Fig. 1.
So, we will have:

\[
C(f_i) = 7 \\
C(f_o) = 8 \\
C(G) = (4 \cdot 3 + 2) + 5 = 19, \text{ so that } C(F) = 34.
\]  

3.2. Implementation Using Programmable Logic Devices (PLD)

We will consider a circuit PAL10L8, (Roth, 1999) which has 10 inputs, 8 outputs and having an AND-OR configuration, each NOR having 2 inputs, with the structure illustrated in Fig. 2.

Let’s consider the previous function:
\[ F = \sum_{i=0}^{d} m_i, \text{ where} \]
\[
\begin{align*}
m_0 &= x_4 \cdot x_3 \cdot x_2 \cdot x_1 \cdot x_0 \\
m_1 &= x_3 \cdot x_2 \cdot x_1 \cdot x_0 \\
m_2 &= x_3 \cdot x_2 \cdot x_1 \cdot x_0 \\
m_3 &= x_3 \cdot x_2 \cdot x_1 \cdot x_0 \\
m_4 &= x_4 \cdot x_3 \cdot x_1 \cdot x_0 \\
m_5 &= x_4 \cdot x_3 \cdot x_2 \cdot x_1 \cdot x_0 \\
m_6 &= x_2 \cdot x_1 \cdot x_0 \\
m_7 &= x_2 \cdot x_1 \cdot x_0 \\
m_8 &= x_2 \cdot x_1 \cdot x_0
\end{align*}
\]

We will use the following algorithm:

\[
\begin{align*}
Q_0 &= m_0 + m_1 \\
Q_1 &= Q_0 + m_2 \\
\vdots \\
Q_7 &= Q_6 + m_8 = F
\end{align*}
\]
Therefore, $Q_i = Q_{i-1} + m_{i+1}$ with $Q_{-1} = m_0$, $0 \leq i \leq 7$.

Will be needed: 9 – product terms ($m_0 + m_5$),

7 – $Q_i$ terms ($0 \leq i \leq 6$), so it will be used 16 product terms from maximum 20. But the number of inputs is insufficient (Fig. 3).

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

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

\[
\begin{align*}
    f_1 &= x_2 \cdot x_1 + x_2 \cdot x_0 \\
    f_0 &= x_1 \cdot x_0 + x_2 \cdot x_1 \cdot x_0 
\end{align*}
\]

and

\[
\begin{align*}
    F(x_4, x_3, x_2, x_1, x_0) &= G(x_4, x_3, f_1, f_0) = f_1 \cdot \overline{f_0} \cdot x_3 + \\
    &f_1 \cdot f_0 + x_1 \cdot x_3 \cdot f_0 + x_4 \cdot x_3 \cdot f_0 + f_1 \cdot x_4 \cdot x_3 
\end{align*}
\]

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.

Fig. 3 – PAL implementation.
4. Conclusions

The paper represents the “rediscovery” of some decomposition algorithms of Boolean logic functions, using subfunctions. 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, (Wilton, 1998). Application of r-admissibility concept makes possible fast evaluation of decompositions for different sizes of block G. This allows selecting best possible decomposition strategy.

REFERENCES


În această lucrare este prezentată o dezvoltare a algoritmului de descompunere a sistemelor digitale, prezentat de Łuba și colaboratorii în 2012 și care constă în studiul comparativ a două metode de detectie a hazardului static (static hazard 0), prin aplicării asupra formei POS – Product of Sums.

Ecuațiile Booleene, prezintă unele avantaje față de metoda consensului, dintre care se evidențiază determinarea tranzacțiilor, comutarea unei variabile de intrare ce provoacă hazard static.