MULTI-LEVEL SYNTHESIS BASED ON LOGIC DECOMPOSITION

Tadeusz Łuba*

In this paper an effective decomposition algorithm for mapping of logic functions onto PLDs as well as FPGAs is proposed. The algorithm exploits our symbolic decomposition concept to find PLD (FPGA) based implementation with a minimal number of devices (CLBs). Experimental results of the presented method are provided and compared to other similar tools.

1. Introduction

In this paper we propose an effective logic synthesis method based on decomposition. Its distinctive feature is that the decomposition process is carried out as the very first step in logic synthesis. This general design strategy illustrated in Figure 1 can be applied for a variety of implementation styles, such as PLAs, gate arrays, or even programmable logic devices. In the paper we will focus on the serial and parallel decompositions aimed primarily at multi-level PLA and PLD structures and its potential capabilities to reduce the silicon area of the semi-custom implementations.

For a long time the idea of Boolean decomposition, formulated originally by Curtis (Curtis, 1962), was applied rather infrequently in practical chip designs because of the lack of effective algorithms. In recent years, however, it is gaining more and more attention. Spectral methods of logic decomposition have been presented by Varma (Varma and Trachtenberg, 1989). The complexity of the proposed algorithms precludes, however, their practical application. Devadas et al. have shown in (Devadas et al, 1988) that re-encoding of selected inputs reduces the number of product terms in logic function. The re-encoding network is serially connected with the PLA that implements the reduced function. The general disjunctive decomposition is the preferred technique of the GATEMAP logic synthesis system (Pitty, 1988). The GATEMAP synthesizer is used to partition complex logic equations. A similar approach, but based on algebraic division method, has been presented in (Saucier, Sicard and Bouchet, 1990). Others methods which target the technology mapping onto FPGAs include so called *groupability* (Dresig, Rettig and Baitinger, 1991), which uses a differential calculus approach (Hurst, Miller and Muzio, 1985). However, these algorithms require exponential time to find the minimum cost decomposition and are still limited to the single-output functions.

* Institute of Telecommunications, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw

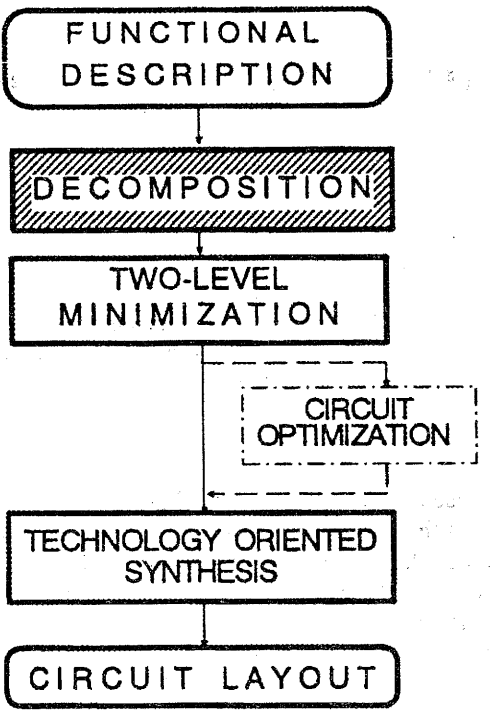

Fig. 1. Logic synthesis based on decomposition.

In this paper we propose new algorithms that attempt to resolve the general problem of functional decomposition of incompletely specified multiple-output Boolean functions. As shown in Figure 1, the results of the decomposition procedure can be directly employed to find an effective physical arrangement of the obtained components. An automated synthesis system, developed at the Warsaw University of Technology, implements the proposed design strategy based on logic decomposition (Łuba, Markowski and Zbierzchowski, 1992).

This paper is organized as follows. First, the symbolic decomposition idea is formulated. An effective algorithm that finds decomposition of the logic function is then presented. Finally, experimental results for PLA and PLD-based implementation are provided.

2. Basic Notions

The fundamental difference between the traditional approaches to Boolean function decomposition and our method, lies on the type objects used and the calculation process itself. Instead of commonly used cubes, we introduce Symbolic Partition Description (SPD) that represents the relation among cubes in the cover of function F.

Consider function $F(x_1,...,x_n)$: $X^n \rightarrow Y^m$. This function can be uniquely represented by ON and OFF sets. The DC-set (DON'T CARE-set) can be easily calculated as a complement of the union ON and OFF sets. We introduce mapping from ON and OFF sets onto the subset of integers:

M: M $\rightarrow$ {1,...,k,k+1,...,p}, such that

ON $\rightarrow$ {1,...,k} = M^0, OFF $\rightarrow$ {k+1,...,p} = M^1

There is a one to one correspondence between cubes from ON and OFF sets and the integers 1,...,p.

Given $F(x_1,...,x_n)$: $X^n \rightarrow Y^m$, and a mapping M: M $\rightarrow$ {1,...,k,k+1,...,p},the generalized partition $P(x_i)$ on the set M of cubes is defined as follows:

$$P(x_i) = \{B^0, B^1\}, \text{ such that, } 1 \in B^j \text{ only if } x_i = j \text{ or } x_i = -,$$

where "–" denotes DON'T CARE entry.

The generalized partitions concept is simple extension of partition algebra, with which reader familiarity is assumed (Almaini, 1964; Hartmanis and Stearns, 1966).

Summarizing, we can say that using SPD, a function is represented as a set of two block input partitions, $P(x_i)$, i=1,..,n, and a multiple-block output partition, P_F. For an input partition, $P(x_i)$, one block contains input vectors for which $x_i = 0$, the other block - the input vectors for which $x_i = 1$. The input vectors that correspond to the same output vector are grouped in the same block of P_F. If the Boolean function is given as array of cubes, then the blocks of partitions may not be disjoint.

3. Logic Decomposition

It is sometimes the case that a set of Boolean functions cannot be made to fit into any single module destinated to 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 n inputs and m outputs, where $n < N$ or $m < M$. There are possible two basic logic decomposition strategies as illustrated in Figure 2. The first one, called parallel decomposition (Fig. 2b), is based on argument reduction concept (Bolton, 1990), whereas the second, called serial, finds serially connected components as shown in Figure 2c.

3.1. Parallel Decomposition

Consider a multiple-output Boolean function F given in a form of the truth table specification. The set Y of outputs can be partitioned into two disjoint subsets such that

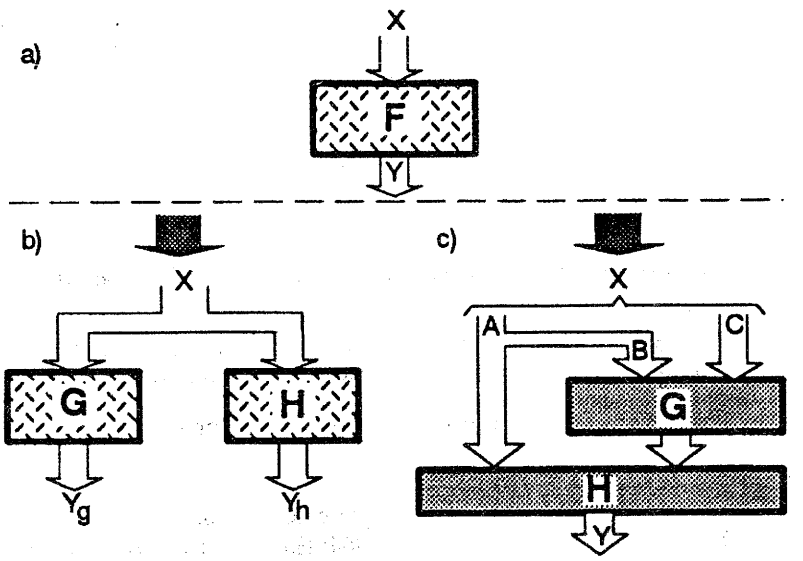

Fig. 2. Parallel and serial decomposition of function F.

the input supports of the obtained components are X_h and X_g, respectively. These may be easily obtained because each output usually depends on a different set of variables, which cardinality is smaller than for the primary X set. Thus, we can group the outputs into separate sets, to obtain the minimal input support sets X_h and X_g.

Example 1.

For the Boolean function given in Table 1, the dependence sets of input variables for every single-output function are as follows:

$y_1 : \{x_1,x_2,x_6\}$,
$y_2 : \{x_3,x_4\}$,
$y_3 : \{x_1,x_2,x_4,x_5,x_9\}, \{x_1,x_2,x_4,x_6,x_9\}$
$y_4 : \{x_1,x_2,x_3,x_4,x_7\}$,
$y_5 : \{x_1,x_2,x_4\}$,
$y_6 : \{x_1,x_2,x_6,x_9\}$.

Therefore, we can determine an optimal two-block decomposition $G = \{y_2, y_4, y_5\}$ and $H = \{y_1, y_3, y_6\}$ with the input support sets $X_g = \{x_1, x_2, x_3, x_4, x_7\}$, and $X_h = \{x_1, x_2, x_4, x_6, x_9\}$. The truth tables of these components are shown in Table 2, respectively.

Table 1.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	y_1	y_2	y_3	y_4	y_5	y_6
1	0	0	0	1	1	1	0	0	0	0	0	0	0	–	0
2	1	0	1	0	0	0	0	0	0	0	0	-	1	0	1
3	1	0	1	1	1	0	0	0	0	0	1	1	0	1	1
4	1	1	1	1	0	1	0	0	0	0	1	1	1	1	0
5	1	0	1	0	1	0	0	0	0	0	0	0	-	0	1
6	0	0	1	1	1	0	0	0	0	1	1	0	1	0	0
7	1	1	1	0	0	0	0	0	0	1	0	-	0	1	0
8	1	0	1	1	0	1	0	0	0	1	1	0	0	-	1
9	1	0	1	1	0	1	1	0	0	-	1	0	1	-	1
10	1	1	1	0	0	0	0	1	0	1	0	1	0	1	-
11	0	0	0	1	1	1	0	0	1	0	0	1	0	-	1
12	0	0	0	1	1	0	0	0	1	-	-	1	0	0	0

Table 2.

a)

x_1	x_2	x_3	x_4	x_7	y_2	y_4	y_5
0	0	0	1	0	0	0	0
1	0	1	0	0	0	1	0
1	0	1	1	0	1	0	1
1	1	1	1	0	1	1	1
1	0	1	0	0	0	–	0
0	0	1	1	0	1	1	0
1	1	1	0	0	0	0	1
1	0	1	1	0	1	0	–
1	0	1	1	1	–	1	–

b)

x_1	x_2	x_4	x_6	x_9	y_1	y_3	y_6
0	0	1	1	0	0	0	0
1	0	0	0	0	0	0	1
1	0	1	0	0	0	1	1
1	1	1	1	0	0	1	0
1	0	1	0	0	1	0	0
0	0	0	0	0	1	1	0
1	1	1	1	0	1	0	1
1	0	1	1	1	0	1	1
1	0	1	0	1	–	1	0

3.2. Serial Decomposition

Let F be a Boolean function representing functional dependency $Y = F(X)$, where X is the set of input variables and Y is the set of binary output variables. Let $X = A \cup B$ and $C \subseteq A$. We say there is a serial decomposition of F iff

$$F = H(A,G(B,C)) = H(A,Z)$$

where G and H denote functional dependencies: $G(B,C) = Z$ and $H(A,Z) = Y$ and Z is the set of two-valued variables. If in addition, $C = \Phi$, then H is called a simple disjoint decomposition of F.

Figure 2c illustrates the problem which has to be solved. We try to find a function H depending on the variables of the set A as well on the outputs of a function G depending on the set $B \cup C$. The outputs of the function H are identical with the function values of F.

The following theorem states the sufficient condition for serial decomposition.

Theorem 1: Functions G and H represent a serial decomposition of function F i.e. $F = H(A,G(B,C))$ iff there exists a partition $\Pi_G \geq P(B \cup C)$ such that

$$P(A) \cdot \Pi_G \leq P_F \tag{1}$$

where all the partitions are over the set of minterms and the number of two-valued output variables of component G is equal to $\lceil \log_2 L(\Pi_G) \rceil$, where $L(\Pi)$ denotes the number of blocks of partition Π.

In the theorem partition Π_G represents component G, and the product of partitions P(A) and Π_G corresponds to H. The truth tables of the resulting components can be easily obtained from these partitions.

Example 2.

Let us decompose a function F of 5 input variables and 3 binary output variables as it is shown in Table 3.

For $A = \{x_1,x_3,x_4\}$, $B = \{x_2,x_5\}$, $C = \Phi$, we obtain

$$P(A) = (1,7\ ;\ 8,13\ ;\ 2,3\ ;\ 9,14,15\ ;\ 4,5\ ;\ 10\ ;\ 6\ ;\ 11,12)$$

Table 3.

	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3
1	0	0	0	0	0	0	0	0
2	0	0	0	1	1	0	1	0
3	0	0	0	1	0	1	0	0
4	0	1	1	0	0	0	1	1
5	0	1	1	0	1	0	0	1
6	0	1	1	1	0	0	1	0
7	0	1	0	0	0	0	0	1
8	1	1	0	0	0	0	0	1
9	1	1	0	1	0	0	0	0
10	1	1	1	0	0	1	0	0
11	1	1	1	1	1	0	1	1
12	1	1	1	1	0	0	1	0
13	1	0	0	0	1	0	0	1
14	1	0	0	1	1	0	0	0
15	1	0	0	1	0	1	0	0

$$P(B) = (1,3,15\ ;\ 2,13,14\ ;\ 4,6,7,8,9,10,12\ ;\ 5,11)$$

$$\Pi_G = (1,3,5,11,15\ ;\ 2,4,6,7,8,9,10,12,13,14)$$

It can be easily verified that since $P(A) \bullet \Pi_G \leq P_F$, where

$$P_F = (1,9,14\ ;\ 5,7,8,13\ ;\ 2,6,12\ ;\ 4,11\ ;\ 3,10,15),$$

function F is decomposable as $F = H(x_1,x_3,x_4,G(x_2,x_5))$, where G is one-output function of two variables.

A key concept in the decomposition method is partitioning of the input support set, X, into two subsets, A and B, that include inputs for components H and G, respectively. The input support set for component H includes also the outputs of G, i.e. $X_H = A \cup Z$. For component G there can exist an additional set of inputs, C, that includes the inputs which are shared between G and H, i.e., $C \subseteq A$.

The r-admissability test presented below allows to obtain the set A of variables which should be connected directly to the circuit H and for which there exists function G (generally with t outputs) such that $|A| + t < n$, where $n = |X|$.

Let $P_1,...,P_k$ be partitions on M, the set of minterms of a function F. The set of partitions $\{P_1,...,P_k\}$ is r-admissible in relation to partition if and only if there is a set $\{P_{k+1},...,P_r\}$ of two-block partitions such that the product Π of partitions $P_1,...,P_k,P_{k+1},...,P_r$ satisfies the inequality $\Pi \leq \theta$, and there does not exist any set of $r - k - 1$ two-block partitions which meets this requirement.

Theorem 2: Let for $\sigma \leq \tau$, $\tau|\sigma$ denotes the quotient partition and $\eta(\tau|\sigma)$ be the number of elements in the largest block of $\tau|\sigma$. Let $e(\tau|\sigma)$ denote the smallest integer equal to or larger than $\log_2(\tau|\sigma)$ i.e.

$$e(\tau|\sigma) = \lceil \log_2 \eta(\tau|\sigma) \rceil.$$

Then $\{P_1,...,P_k\}$ is r-admissible, where

$$r = k + e(\Pi|\Pi_F)$$

moreover Π is the product of partitions $P_1,...,P_k$ and $\Pi_F = \Pi \bullet P_F$.

Theorem 2 provides an approach to evaluate the admissability of a set of input variable partitions $P(x_i)$. In searching for a maximum set of variables, which can be connected to the circuit H (Fig. 2) directly, we compute sets of t-admissible partitions $P(x_i)$ only, where t is the given number of inputs of the circuit H.

Example 3.

Consider the following set of partitions on $M = \{1,...,15\}$.

$$P_1 = (1,...,7\ ;\ 8,...,15)$$
$$P_2 = (1,2,3,13,14,15\ ;\ 4,...,12)$$
$$P_3 = (1,2,3,7,8,9,13,14,15\ ;\ 4,5,6,10,11,12)$$

P_4 = (1,4,5,7,8,10,13}, 2,3,6,9,11,12,14,15)
P_5 = (1,3,4,6,...10,12.15 ; 2,5,11,13,14)
P_F = (1,9,14 ; 5,7,8,13 ; 2,6,12 ; 4,11 ; 3,10,15)

representing function F of Example 2,where P_i denotes $P(x_i)$.

By examining admissability of P_1 we obtain

$P_1 \bullet P_F$ = (1 ; 9,14 ; 5,7 ; 8,13 ; 2,6 ; 12 ; 4 ; 11 ; 3 ; 10,15)
$P_1 | P_1 \bullet P_F$ = ((1)(2,6)(3)(4)(5,7) ; (8,13)(9,14)(10,15)(11)(12))

Hence, r-admissability, $r = 1 + \lceil \log_2 5 \rceil = 4$, i.e. $r(P_1) = 4$. Also, as the quotient partition $\Pi | \Pi \bullet P_F$, where $\Pi = P_1 \bullet P_3 \bullet P_4$ is equal

((1)(7) ; (8,13) ; (2)(3) ; (9,14)(15) ; (4)(5) ; (10) ; (6) ; (11)(12)).

the set $\{P_1, P_3, P_4\}$ is 4-admissible. Similarly, r = 4 for set $\{P_3, P_4, P_5\}$. Therefore, F can be realized in the circuit H with inputs x_1, x_3, x_4, g_1 or x_3, x_4, x_5, g_2, where $F = H(x_1, x_3, x_4, G_1(x_2, x_5, C_1))$ or $F = H(x_3, x_4, x_5, G_2(x_1, x_2, C_2))$, respectively.

The next task is to find a subset of inputs for component G that serially connected with component H will implement function F (Fig. 2), i.e. to find $P_G = P(B \cup C)$, such that there exists $\Pi_G \geq P_G$ that satisfies condition (1) in Theorem 1. To solve this problem, consider a subset of primary inputs, $D = B \cup C$, and an m-block partition $P(D) = (B_1; B_2; ...; B_m)$ generated by this subset.

A relation of compatibility of partition blocks will be used to verify whether or not partition P(D) is suitable for serial decomposition.

Two blocks $B_i, B_j \in P(D)$ are compatible iff partition Π_G' obtained from partition P(D) by merging blocks B_i and B_j into a single block B_j satisfies condition (1) in Theorem 1, i.e., iff

$$P(A) \bullet \Pi_G' \leq P_F$$

A subset of n partition blocks, B = {Bi1,Bi2,...,Bin}, where Bij $\in$ P(D), is a class of compatible blocks for partition P(D) iff all blocks in B are pairwise compatible.

A compatible class is called Maximal Compatible Class (MCC) iff it cannot be properly covered by any other compatible class.

If we can form suitable MCCs of blocks, we can merge them into a singular block to obtain partition Π_G such that

$$P(A) \bullet \Pi_G \leq P_F, \qquad (2)$$

where partition Π_G represents function G, and the product of partitions P(A) and P corresponds to function H. The truth table description of these functions can be easily obtained from the partitions.

We have developed an algorithm that finds serial decomposition of a given logic function. The algorithm includes several steps. At first it determines the input variable sets for both components, using an r-admissability criterion. In general the r-

admissability test allows to select the input variables for each component. Next we calculate so called compatibility classes for blocks of partition P(B ∪ C). These classes represent relation between blocks of partition induced by input support set for component G. Then the minimal cover of compatibility classes is found to minimize the number of outputs of component G. In the first run of decomposer we check an existence of disjoint decomposition. If such a decomposition does not exist we include in the input support of component G, an additional, minimal set of inputs, to satisfy the non-disjoint decomposition.

4. Application of The Decomposition for PLA, PLD and PGA Implementations

The decomposition procedure can be easily applied to decompose a PLA into a set of smaller interconnected PLAs such that the overall area of the resulting logic network, deemed to be the sum of the areas of the constituent PLAs, is minimized. The silicon area of the PLA can be estimated as S = (2n + m)P, where n and m denote the number of inputs and outputs, respectively, and P is the number of product terms in a minimized form. An interesting example is one of the MCNC benchmarks, called RD84. After Espresso minimization the truth table of RD84 has 8 inputs, 4 outputs and 255 cubes thus, the estimated area of the chip is: S = (2·8 + 4)·255 = 5100.

When the RD84 function is decomposed into 3 components connected as shown in Figure 3, the number of product terms for components G_0, G_1, and H are 16, 16, and 25, respectively. But minimizing them we gain 10, 10 and 19, respectively. Therefore the total PLA area is only 10% of the original circuit after minimization procedure. Advantages of the decomposition can be also demonstrated by comparing various PLD-based implementations of the RD84 function. Using the design method presented in (Saucier, Sicard and Bouchet, 1990), this function was reported to require 4 PALs from

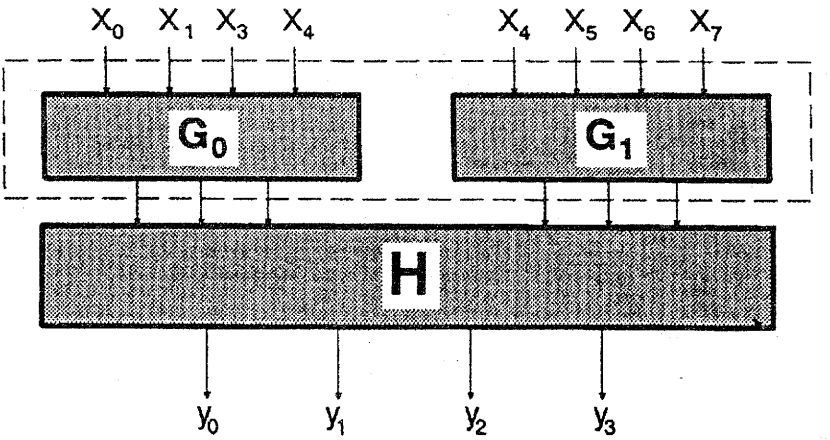

Fig. 3. Serial decomposition of RD84 benchmark circuit.

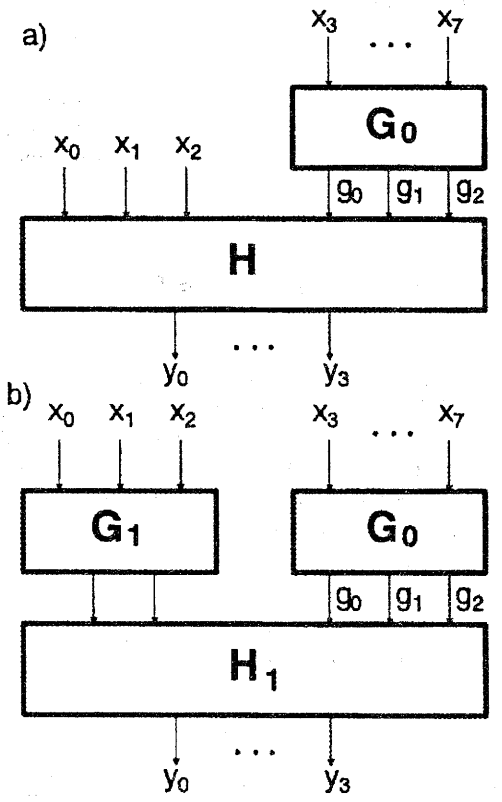

Fig. 4. Decomposition of RD84 for PGA implementation.

the MMI PAL library. Decomposition procedure yields the spectacular result: just one PAL device is sufficient.

Decomposition concept can be also effectively used for the logic synthesis based on PGAs. As a basic logic unit in PGA is a CLB, with the number of inputs limited to m (typically 4 or 5), so implementation of any logic function of more inputs, needs the function to be decomposed. However, it should be noticed that for PGA implementation a function does not need to be minimized, because the CLB block can realize any logic function of its input variables. Therefore serial decomposer that runs before the minimization procedure is applied would be an ideal tool for this purpose.

The example design produced by the serial decomposition procedure for RD84 benchmark example is given in Figure 4. In the first run decomposer finds the structure of 2 blocks: G_0 and H as shown in Figure 4a. The decomposition process is applied iteratively so component H can be decomposed in the next step with respect to the set $A = \{g_0, g_1, g_2\}$. Thus, complete decomposition process of the RD84 function, yields a

set of 3 hierarchically interconnected blocks shown in Figure 4b. Because the number of inputs to each block is less than 5, this circuit can be directly implemented by the network of CLBs in XILINX family 3000 (Xilinx Inc., 1991). In this serie a CLB can implement any single-output function of up to 5 input variables or any two-output function of up to 5 variables, with each output depending on at most 4 input variables. We can easily verify that this implementation occupies only 8 CLBs configured in 2 CLB levels (critical path), which is a much better result than results reported so far in the available literature. This function was reported to require 27 (Filo et al, 1991), 33 (Sicard et al, 1991), and 12 (Dresig, Rettig and Baitinger, 1991) CLBs with ?, 7, and 4 levels of CLBs in a critical path, respectively). However increasing the critical path up to 3 this result can be improved up to only 6 (!) CLBs (Łuba, Markowski and Zbierzchowski, 1992).

5. Summary

In this paper we have proposed a general method for decomposition of incompletely specified multiple-output Boolean functions. The symbolic decomposition algorithm is simple and, even for complex circuits, does not involve significant computational effort.

Our experiments have shown that, when a Boolean function is decomposed before the two-level minimization is applied, then the minimization process is more effective.

We have proved that combining logic decomposition with topological partitioning we can decrease product term count for PLA implementations. The decomposition concept can also be effectively used as the collapse algorithm that manipulates the network into a form suitable for PAL and PGA implementations. This becomes especially important with respect to the increasing market of Programmable Logic Devices and Programmable Gate Arrays.

It is our hope that the ideas described in this paper represent a foundation for the development multi-level logic decomposition algorithms. Many of the difficulties existing in previous decomposition methods now have practical solution. The starting point to all practical decomposition solutions can be, developed in this paper, general theorem of the decomposition, which allows to decompose any Boolean function.

References

Almaini A.E.A. (1964): *Electronic Logic Systems.*- Englewood Cliffs NJ, Prentice-Hall International.

AMD and MMI (1988): *PAL Device Handbook.*

Bolton M. (1990): *Digital Systems Design with Programmable Logic.*- Addison-Wesley Publishing Company.

Brayton R.K., Hachtel G.D., McMullen C.T. and Sangiovanni-Vincentelli A. (1984): *Logic Minimization Algorithms for VLSI Synthesis.*- Kluwer Academic Publ.

Curtis H.A. (1962): *A New Approach to the Design of Switching Circuits.*- Van Nostrand Company, Princeton, N.J.

Kuo Y.H., Wang R.R. and Kung L.Y. (1988): *Logic design using the PLA's with limited I/O pins and product terms.*- Microprocessing and Microprogramming. No.23.

Devadas S., Wang A.R., Newton A.R. and Sangiovanni-Vincentelli A. (1988): *Boolean decomposition in multi-level logic optimization.*- Proc. Int. Conf. on Computer-Aided Design, Nov., pp. 290-293.

Dresig F., Rettig O. and Baitinger U.G. (1991): *Logic synthesis for universal logic cells.*- Int. Workshop on Field Programmable Logic and Applications, Oxford, UK.

Filo D., Yang J.C., Mailhot F. and De Micheli G. (1991): *Technology mapping for a two-output RAM-based field programmable gate array.*- Proc. of European Conf. on Design Automation, Feb., pp. 534-538.

Hartmanis J. and Stearns R.E. (1966): *Algebraic Structure Theory of Sequential Machines.*- Prentice-Hall, Inc.

Hurst S.L., Miller D.M. and Muzio J.C. (1985): *Spectral Techniques in Digital Logic.*- Academic Press.

Jasinski K., Łuba T. and Kalinowski J. (1989): *Parallel decomposition in logic synthesis.*- Proc. 15th European Solid-State Circuits Conf., Sept., pp. 113-116.

Jasinski K., Łuba T. and Kalinowski J. (1991): *CAD tools for PLD implementation of ASICs.*- Proc. of Second Eurochip Workshop on VLSI Design Training, Grenoble, pp225-230.

Łuba T. (1986): *A Uniform method of boolean function decomposition.*- Rozprawy Elektrotechniczne, Journal of the Polish Academy of Science, No.4, pp. 1041-1054.

Łuba T., Kalinowski J., Jasinski K. and Krasniewski A. (1991): *Combining serial decomposition with topological partitioning for effective multi-level PLA implementations.*- P.Michel and G.Saucier (Eds.), Logic and Architecture Synthesis, Elsevier Science Publishers B.V. (North-Holland).

Łuba T., Kalinowski J. and Jasinski K. (1991): *PLATO: A CAD tool for logic synthesis based on decomposition.* Proc. of European Conference on Design Automation, Feb., pp. 65-69.

Łuba T., Markowski M. and Zbierzchowski B. (1992): *Logic decomposition for programmable gate arrays.*- Proc. EURO-ASIC'92, June, Paris, France.

McCluskey E.J. (1986): *Logic Design Principles With Emphasis on Testable Semicustom Circuits.*- Prentice-Hall.

Murgai R., Nishizaki Y., Shenoy N., BraytonR.K. and Sangiovanni-Vincentelli A.: *Logic synthesis for programmable gate arrays.*- Proc. of the 27th ACM/IEEE.

Pawlak Z. (1991): *Rough Sets. Theoretical Aspects of Reasoning about Data.*- Kluwer Academic Publishers.

Pitty E.B. (1988): *A Critique of the GATEMAP Logic Synthesis System.*- Proc. Int. Workshop on Logic and Architecture Synthesis for Silicon Compilers, May, Grenoble, France, pp.65-84.

Sicard P., Crastes M., Sakouti K. and Saucier G. (1991): *Automatic synthesis of Boolean functions on Xilinx and Actel programmable devices.*- Proc. Euro ASIC '91, May, pp.142-145.

Saucier G., Sicard P. and Bouchet L. (1990): *Multi-level synthesis on PALs.*- Proc. European Design Automation Conf., March, Glasgow, UK, pp.542-546.

Varma D. and Trachtenberg E.A. (1989): *Design automation tools for efficient implementation of logic functions by decomposition.*- IEEE Trans. on CAD, v.8, No.8, Aug., pp.901-916.

Xilinx Inc. (1991): *Xilinx Programmable Gate Array User's Guide.*