www.czasopisma.pan.pl PAN www.journals.pan.pl
jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2011, VOL. 57, NO. 2, PP. 209–216
Manuscript received April 18, 2011; revised May 2011. DOI: 10.2478/v10177-011-0029-4

Application of Indexed Partition Calculus in Logic Synthesis of Boolean Functions for FPGAs

Mariusz Rawski

***Abstract*—Functional decomposition of Boolean functions specified by cubes proved to be very efficient. Most popular decomposition methods are based on blanket calculus. However computation complexity of blanket manipulations strongly depends on number of function's variables, which prevents them from being used for large functions of many input and output variables. In this paper a new concept of indexed partition is proposed and basic operations on indexed partitions are defined. Application of this concept to logic synthesis based on functional decomposition is also discussed. The experimental results show that algorithms based on new concept are able to deliver good quality solutions even for large functions and does it many times faster than the algorithms based on blanket calculus.**

***Keywords*—indexed partition, logic synthesis, FPGA.**

I. Introduction

FOR years now, *functional decomposition* is perceived as one of the best logic synthesis methods targeted FPGAs. Today's FPGAs are entire programmable systems on a chip (SoC) which are able to cover an extremely wide range of applications. They specific architecture based on the lookup table (LUT) as basic building block requires a specific logic synthesis methods. A logic network describing implemented system must be transformed into network that consists of nodes of limited number of inputs only, since an n-input LUT is capable of implementing any Boolean function of up to n variables.

Functional decomposition relies on breaking down a complex system into a network of smaller and relatively independent co-operating subsystems, in such a way that the original system's behavior is preserved. A system is decomposed into a set of smaller subsystems, such that each of them is easier to analyze, understand and synthesize. Decomposition allows synthesizing the Boolean function into multilevel structure that is built of components, each of which is in the form of LUT logic block specified by truth tables.

Since the Ashenhurst-Curtis decomposition have been proposed, many new decomposition techniques have been developed [1]–[5], but they are still based on Ashenhurst's ideas.

The classical methods for computing functional decomposition were based on representing the Boolean function by a *decomposition chart* [6], [7]. Shortly after their introduction, decomposition charts were abandoned in favor of cube representation [8], and computing column multiplicity on charts was replaced by computing compatible classes for a set of cubes. Cubes can efficiently represent many practical logic multiple-output, partially specified Boolean function. In [1] a comprehensive theory of serial functional decomposition for multiple-output, partially specified Boolean functions represented by cubes was proposed. This theory uses the concept of *blankets* that are generalized set systems [9]. The blanket calculus allowed construction of very efficient decomposition algorithms [10], [11]. However the computational complexity of blanket manipulations made it impossible to efficiently apply these algorithms to large Boolean functions.

In [12] the theory of information relationships and measures has been proposed that also uses cube representation of Boolean function. In this approach the concept of *information* and *abstraction sets* is used to model various "information streams" in discrete information systems. This concept allowed to investigate relationships between information in various information streams of a digital system represented by Boolean function. For instance, relationships between information delivered by certain inputs and information necessary for computing certain outputs could be measured, such as: similarity, dissimilarity, missing information and extra information. However the analysis of dependency between "information streams" in function on such "atomic" level makes it very difficult to construct efficient algorithms for synthesis of large multi-output Boolean functions. There are algorithms presented in literature that use this concept to construct efficient algorithms of functional decompositions limited only to small or single-output functions [13], [14] or to support decomposition methods based on blankets [11]. Moreover to express large functions the espresso format is often used, that can use cubes that have "no meaning" for some outputs. This in general does not specify "pure" truth table, so algorithms based on concept of information relationships and measures cannot be directly used in contrast to methods based on blanket calculus.

A Boolean function can be represented using binary decision diagrams. BDDs as a method of representation of single-output Boolean functions were introduced by Lee [15] and later Ackers [16]. In [17] Bryant presented algorithms that efficiently manipulated BDDs assuming ordering of the variables. He developed a method to reduce the size of BDDs by removing 'redundant' nodes and sub-graphs which occur more than once. Decomposition with use of BDDs involves the process of variable ordering that leads to good decomposition. Normally, the size of BDD varies for different variable orderings and, for some functions, it is highly sensitive to the ordering. Finding the best ordering that minimises the size of the BDD requires, in the worst case, a time exponential in the number of variables. Finding a good variable ordering for

This work was partly supported by the Ministry of Science and Higher Education of Poland – research grant no. N N516 418538 for 2010-2012.

M. Rawski is with Institute of Telecommunications, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland (e-mail: rawski@tele.pw.edu.pl).

evaluation of functional decomposition is time consuming and influences the efficiency whole process of decomposition.

Logic circuits usually have many outputs. The functional decomposition of multi-output circuit requires such representation of multiple-output function that allows efficient creation of sub-circuits common to all of the outputs. Cube representation of multiple-output Boolean function is very easy. To represent such functions by using BDDs there have been proposed several methods. One of the first methods is a multi-terminal binary decision diagram (MTBDD) [18]. Unfortunately, MTBDDs tend to be too large to construct. The second method is a binary decision diagram (BDD) for the characteristic function (CF) of the multiple-output function. The advantage of the CF is its small evaluation time. CFs are used in logic simulation and multi-level logic optimization [19]. The third method is a shared binary decision diagram (SBDD) [18]. In many cases, SBDDs are smaller than corresponding MTBDDs and BDDs for CFs. Recently an encoded characteristic function for non zero outputs has been proposed for compact representation of multi-output Boolean function [4]. Decomposition methods based on concept of BDDs allow synthesizing even large functions but methods based on blanket calculus offer better quality of obtained solutions (in terms of number of blocks decomposed network).

In this paper a concept of *indexed partition* is presented that allows manipulate Boolean function described by cubes in similar way as blanket calculus but requires much less computational and memory complexity. It combines the advantages of both blanket calculus and information relationships and measures. Presented results prove that application of indexed partition allows to efficiently apply synthesis method even for large Boolean functions.

II. Preliminary Information

A. Cube Representation of Boolean Functions

A Boolean function can be specified using the concept of cubes (input terms, patterns) representing some specific subsets of minterms. In a minterm, each input variable position has a well-specified value. In a cube, positions of some input variables can remain unspecified and they represent "any value" or "don't care" ($-$). A cube may be interpreted as a p-dimensional subspace of the n-dimensional Boolean space or as a product of $n-p$ variables in Boolean algebra (p denotes the number of components that are '$-$'). Boolean functions are typically represented by truth tables. Truth table description of function using minterms requires 2^n rows for function of n variables. For function from Table I truth table with $2^6 = 64$ rows would be required. Since cube represents a set of minterms, application of cubes allows for much more compact description in comparison with minterm representation. For example cube $0101-0$ from row 1 of truth table from Table I represents set of two minterms $\{010100, 010110\}$.

For pairs of cubes and for a certain input subset B, we define the *compatibility relation* COM as follows: each two cubes S and T are compatible (i.e. $S, T \in \text{COM}(B)$) if and only if $x(S) \sim x(T)$ for every $x \subseteq B$. The compatibility relation $\sim$ on $\{0,-,1\}$ is defined as follows [1]: $0 \sim 0, - \sim -, 1 \sim 1, 0 \sim -, 1 \sim -, - \sim 0, - \sim 1$, but the pairs (1, 0) and (0, 1) are not related by $\sim$. The compatibility relation on cubes is reflexive and symmetric, but not necessarily transitive. In general, it generates a "partition" with non-disjoint blocks on the set of cubes representing a certain Boolean function F. The cubes contained in a block of the "partition" are all compatible with each other.

TABLE I
Boolean Function $y_1 = f(x_1, x_2, x_3, x_4, x_5, x_6)$

	x_1	x_2	x_3	x_4	x_5	x_6	y_1
1	0	1	0	1	–	0	0
2	0	1	0	–	0	0	0
3	–	1	0	0	0	–	0
4	0	1	0	1	1	–	0
5	0	0	1	–	–	1	0
6	–	–	1	1	–	1	0
7	1	–	1	1	0	–	0
8	0	0	–	–	–	0	1
9	–	1	0	0	1	–	1
10	1	–	1	0	–	–	1

The compatibility relation COM can be represented as a *compatibility graph*. Each vertex in such a graph is associated with a cube. There is an edge between the two vertices of that graph if the related cubes are compatible.

Information on the input patterns of a certain function F is delivered by the function's inputs and used by its outputs with precision to the blocks of the input and output "partitions". Knowing the block of a certain "partition", one is able to distinguish the elements of this block from all other elements, but is unable to distinguish between elements of the given block. In this way, information in various points and streams of discrete information systems can be modeled.

"Partitions" with non-disjoint blocks are referred to as *blankets* [1]. The concept of blanket is a simple extension of ordinary partition and typical operations on blankets are strictly analogous to those used in the ordinary partition algebra.

The compatibility relation on Boolean function's cubes can also be modeled using the concept of information and abstraction sets [12].

B. Blanket Calculus

A *cover* on a set S is such a collection of (not necessary disjoint) subsets B_i of S, called blocks, that

$$\bigcup_i B_i = S \tag{1}$$

The product of two covers σ_1 and σ_2 is defined as follows:

$$\sigma_1 \bullet \sigma_2 = \{B_i \cap B_j | B_i \in \sigma_1 \text{ and } B_j \in \sigma_2\}. \tag{2}$$

A *blanket* on a set S is a cover $\beta = \{B_1, \ldots, B_k\}$ of nonempty and distinct subsets of S, called blocks.

Define "nonempty" operator ne as follows. For any set $\{S_i\}$ of subsets of set S, $ne\{S_i\}$ is $\{S_i\}$ with empty subset removed, if was originally present and only one instance of block if more similar block were originally present.

The product of two blankets β_1 and β_2 is defined as follows:

$$\beta_1 \bullet \beta_2 = ne\{B_i \cap B_j | B_i \in \beta_1 \text{ and } B_j \in \beta_2\}. \tag{3}$$

For two blankets we write $\beta_1 \leq \beta_2$ if and only if for each B_i in β_1 there exists a B_j in β_2 such that $B_i \subseteq B_j$. The relation $\leq$ is reflexive and transitive.

Each block B_i of cover (blanket) has its cube representative $r(B_1)$ that indicates the value of variables inducing this cover(blanket) corresponding to this block.

Example 1. (Blanket-based representation of Boolean functions).

For function F from Table I, the blankets induced by particular input and output variables on the set of function $F's$ input patterns (cubes) are as follows:

$$\beta_{x1} = \{B_1; B_2\} = \{\overline{1,2,3,4,5,6,8,9}; \overline{3,6,7,9,10}\}, \quad (4)$$

$$\beta_{x2} = \{\overline{5,6,7,8,10}; \overline{1,2,3,4,6,7,9,10}\}$$

$$\beta_{x3} = \{\overline{1,2,3,4,8,9}; \overline{5,6,7,8,10}\}$$

$$\beta_{x4} = \{\overline{2,3,5,8,9,10}; \overline{1,2,4,5,6,7,8}\}$$

$$\beta_{x5} = \{\overline{1,2,3,5,6,7,8,10}; \overline{1,4,5,6,8,9,10}\}$$

$$\beta_{x6} = \{\overline{1,2,3,4,7,8,9,10}; \overline{3,4,5,6,7,9,10}\}$$

$$\beta_{y_1} = \{\overline{1,2,3,4,5,6,7}; \overline{8,9,10}\}$$

The representative of block B_1 of blanket β_{x1} is $r(B_1) = 0$, since variable x_1 has value 0 for input patterns 1, 2, 4, 5, 8. Similarly for block B_2 of blanket β_{x1} $r(B_2) = 1$.

Blanket generated by variable x_1 models information delivered by this variable. Variable x_1 cannot be used to distinguish input pattern 1 and 2, since it has value 0 for both of them in Table I and this is reflected in blanket β_{x1} – these patterns are contained in the same block. Similarly input pattern 1 and 7 are distinguish by this variable, since in Table I it has value 0 for input pattern 1 and value 1 for input pattern 7. In blanket β_{x1} these patterns are contained in different blocks.

Product of blankets $\beta_{x2}, \beta_{x4}, \beta_{x5}$ presented as cover σ_{x2x4x5} may have empty and repetitive blocks.

$$\sigma_{x2x4x5} = \beta_{x2} \bullet \beta_{x4} \bullet \beta_{x5}$$

$$= \{B_1; B_2; B_3; B_4; B_5; B_6; B_7; B_8\}$$

$$= \{\overline{5,8,10}; \overline{5,8,10}; \overline{5,6,7,8}; \overline{5,6,8};$$

$$\overline{2,3,9,10}; \overline{9,10}; \overline{1,2,4,6,7}; \overline{1,4,6}\}$$

The cube representative of a block B_3 of cover σ_{x2x4x5} is $r(B_3) = 010$, since this block was obtained from blocks B_1 of β_{x2}, B_2 of β_{x4} and B_1 of β_x5. The representatives of these blocks are respectively 0, 1 and 0. Representatives of cover's blocks are always minterms.

Product of blankets $\beta_{x2}, \beta_{x4}, \beta_{x5}$ presented as blanket β_{x2x4x5} has empty and repetitive blocks removed.

$$\beta_{x2x4x5} = \beta_{x2} \bullet \beta_{x4} \bullet \beta_{x5} = \{B_1; B_2; B_3; B_4; B_5; B_6; B_7\}$$

$$= \{\overline{5,8,10}; \overline{5,6,7,8}; \overline{5,6,8}; \overline{2,3,9,10};$$

$$\overline{9,10}; \overline{1,2,4,6,7}; \overline{1,4,6}\}.$$

The relationship between blocks of β_{x2x4x5} and their cube representatives $r(B_i)$, relies on containment of block B_i in blocks of blankets used in product. Denoting blocks of β_{x2x4x5} as B_1 through B_7, we have $r(B_1) = 00-$. This is because $B_1 = \{5, 8, 10\}$ is included in the first blocks of β_{x2}, β_{x4} and in both blocks of β_{x5}. For $B_2 = \{5, 6, 7, 8\}$, we have: B_2 is included in the first block of β_{x2}, in the second block of β_{x4} and in first block of β_{x5}. Hence, $r(B_2) = 010$.

Blanket calculus proved to be very efficient in decomposition based logic synthesis of Boolean functions specified by cubes. Many ideas based on this concept can be found in literature that solve problems from field of general (functional) decomposition of combinational circuits [3], [10], [20]–[22] and sequential machines [23], pattern analysis, knowledge discovery, machine learning, decision systems, data bases, data mining etc. [24]–[26]. Unfortunately computational and memory complexity of basic manipulation on blankets grows exponentially with the number of variables of Boolean function. This makes it unable to use them for large Boolean functions.

C. Information Relationships and Measures

The theory of information relationships and measures is presented in paper [12]. Information on symbols from a certain set S means the ability to distinguish certain symbols from some other symbols. An *elementary information* describes the ability to distinguish a certain single symbol s_i from another single symbol s_j, where: $s_i, s_j \in S$ and $s_i \neq s_j$. Any information on elements from a certain set S can be represented as a composition of such elementary portions of information. In particular, one can represent any set of elementary portions of information by an *information (incompatibility) relation* I or an *information set* IS defined on $S \times S$ as follows:

$$I = \{(s_i, s_j) | s_i \text{ is distinguished from } s_j \text{ by the modelled information}\},$$

$$IS = \{\{s_i, sj\} | s_i \text{ is distinguished from } s_j \text{ by the modelled information}\}.$$

In similar way an *elementary abstraction* can be defined as inability to distinguish a certain single symbol s_i from another single symbol s_j, where: $s_i, s_j \in S$ and $s_i \neq s_j$.

Having defined elementary portions of information and abstraction it is possible to express the information modeled by blankets in terms of these elementary portions, i.e. in terms of the information relations and sets and it is possible to analyze the relationships between blankets by analyzing the relationships between their corresponding information relations and sets. In particular, the correspondence between blankets and IS is as follows: IS contains the pairs of symbols that are not contained in any single block of a corresponding blankets.

For instance, let's consider the blanket β_1 from (4), the corresponding information set is as follows:

$$IS_{x1} = \{1|7, 1|10, 2|7, 2|10, 4|7, 4|10, 5|7, 5|10, 8|7, 8|10\}$$

Symbol "|" in pairs $s_i|s_j$ from IS_{x1} is used to stress that the elements s_i and s_j of a certain pair $\{s_i, s_j\}$ are distinguished from each other.

Performing analysis or design of discrete information systems, we often ask for relationships between information in various information streams of a considered system. For instance, one may be interested in relationships between information delivered by certain inputs and information necessary for computing certain outputs, such as: similarity, dissimilarity, missing information and extra information.

In [12], among others the following *relationships between information* of two blankets β_1 and β_2 are defined:

common information (i.e. information present in both β_1 and β_2): $CI(\beta_1, \beta_2) = IS(\beta_1) \cap IS(\beta_2)$,
extra information (i.e. information missing in β_1, but present in β_2): $EI(\beta_1, \beta_2) = IS(\beta_2) - IS(\beta_1)$.

The simplest quantitative measures for the amount of information and for the strength of information relationships are the *absolute measures* that are defined by the number of elements in the appropriate information set. In this way, for a single blanket β the following measure was defined in [12]:

information quantity: $IQ(\beta) = |IS(\beta)|$

For two blankets β_1 and β_2, the following *relationship measures* were defined:

information similarity measure:

$$ISIM(\beta_1, \beta_2) = |CI(\beta_1, \beta_2)|$$

and
information increase measure:

$$IINC(\beta_1, \beta_2) = |EI(\beta_1, \beta_2)|$$

Information relationships and measures concept operates on sets of "elementary information" and "elementary abstraction". Basic operations here are very simple: union, intersection, difference, symmetric difference of sets. Computational and memory complexity of these operations is polynomial and depends on the number of cubes representing Boolean function. However there are no functional decomposition methods that are based on this concept only. Information relationships and measures deliver very efficient mean for analyzing dependencies between variables of Boolean functions, but do not show how to construct decomposition without the concept of blanket. Still there are methods that use this concept to support some parts of decomposition process, such as input variable partitioning, symbolic sub-function selection and encoding [11], [13], [14], [27].

III. Indexed Partition Concept

Blankets, as well as information sets are the way of expressing the compatibility relation COM on Boolean function's cubes. This also can be done using the concept of compatibility or *incompatibility graph.*

Definition 1. For Boolean function F specified by set S of cubes and for a certain set B of function F's variables we define an incompatibility graph $\Gamma_B = (N, E)$, where the set N of vertices corresponds to the set S of cubes and set E of edges is formed by set of incompatible pairs of cubes.

Operations on blankets or information sets have direct analogue in operations on incompatibility graphs. For example the product of two blankets β_1 and β_2 can be modeled as computing incompatibility graph $\Gamma_{12} = (N, E = E_1 \cup E_2)$ using graphs $\Gamma_1 = (N, E_1)$ and $\Gamma_2 = (N, E_2)$ (corresponding to these blankets).

Here the concept of *indexed partitions* is introduced, that is used to model such operations on incompatibility graphs. Indexed partitions are compact way of representing incompatibility graphs.

Definition 2. An *indexed partition* is a set of ordered dichotomies $\{B_i, P_i\}$ on set S, such that B_i are disjoint subsets of S and

$$\bigcup_i B_i = S \tag{5}$$

Example 2. (Indexed partition-based representation of Boolean functions).

For function F from Table I, the indexed partitions induced by selected input and output variables on the set of function F's input patterns (cubes) are as follows:

$$\begin{gathered}\delta_{x1} = \{\{B_1, P_1\}; \{B_2, P_2\}; \{B_3, P_3\}\} = \\ \{\{\{1,2,4,5,8\}, \{7,10\}\}; \{\{3,6,9\}, \{\varnothing\}\}; \\ \{\{7,10\}, \{1,2,4,5,8\}\}\}, \\ \delta_{x2} = \{\{\{5,8\}, \{1,2,3,4,9\}\}; \{\{6,7,10\}, \{\varnothing\}\}; \\ \{\{1,2,3,4,9\}, \{5,8\}\}\}, \\ \delta_{y1} = \{\{\{1,2,3,4,5,6,7\}, \{8,9,10\}\}, \\ \{\{8,9,10\}, \{1,2,3,4,5,6,7\}\}\}.\end{gathered} \tag{6}$$

Equation (6) can be interpreted in such way that input variable x_1 can be used to distinguish every input pattern from set B_1 from every pattern from set P_1, (similarly $\{B_3, P_3\}$), however P_2 is an empty set so variable x_1 cannot distinguish patterns included in B_2 from any other pattern.

Definition 3. The product of two indexed partitions δ_1 and δ_2 is defined as follows:

$$\begin{gathered}\delta_1 \bullet \delta_2 = \{\{B_i \cap B_j, P_i \cup P_j\} | B_i \cap B_j \neq \varnothing, \\ \{B_i, P_i\} \in \delta_1 \text{ and } \{B_j, P_j\} \in \delta_2\}.\end{gathered} \tag{7}$$

Example 3. (Product of two indexed partitions).

For indexed partitions from example 2 we have:

$$\begin{gathered}\delta_{x1x2} = \delta_{x1} \bullet \delta_{x2} = \\ \{\{\{5,8\}, \{1,2,3,4,7,9,10\}\}; \\ \{\{1,2,4\}, \{5,7,8,10\}\}; \\ \{\{6\}, \{\varnothing\}\}; \\ \{\{3,9\}, \{5,8\}\}; \\ \{\{7,10\}, \{1,2,4,5,8\}\}\};\end{gathered}$$

Definition 4. For two indexed partitions we write $\delta_1 \leq \delta_2$ if and only if for each $\{B_i, P_i\}$ in δ_1 and each $\{B_j, P_j\}$ in δ_2 $P_i \supseteq P_j$ if $B_i \cap B_j \neq \varnothing$. The relation $\leq$ is reflexive and transitive.

Example 4. (The relation $\leq$).

It can be easily verified that $\delta_{x1x2} \leq \delta_{x1}$. However it is not true that $\delta_{x1x2} \leq \delta_{y1}$, since for $\{B_4, P_4\} = \{\{3,9\},\{5,8\}\}$ from δ_{x1x2} and $\{B_2, P_2\} = \{\{8,9,10\},\{1,2,3,4,5,6,7\}\}$from δ_{y1} we have that $B_4 \cap B_2 \neq \varnothing$ but $P_4, \supseteq P_2$.

Definition 5. The quotient of two indexed partitions δ_1 and δ_2 is defined as follows:

$$\delta_1|\delta_2 = \{\{B_i \cap B_j, P_i - P_j\}|B_i \cap B_j \neq \varnothing, \{B_i, P_i\} \in \delta_1 \text{ and} \{B_j, P_j\} \in \delta_2\}. \quad (8)$$

Example 3. (Quotient of two indexed partitions).

For indexed partitions from example 2 we have:

$$\delta = \delta_{y1}|\delta_{x1} =$$
$$\{\{\{1,2,4,5\},\{8,9\}\};$$
$$\{\{3,6\},\{8,9,10\}\};$$
$$\{\{7\},\{9,10\}\};$$
$$\{\{8\},\{1,2,3,4,5,6\}\};$$
$$\{\{9\},\{1,2,3,4,5,6,7\}\}$$
$$\{\{10\},\{3,6,8\}\}\}$$

For indexed partition we can also use quantitative measures similar to these defined for information sets:

information quantity: $IQ(\delta) = \frac{1}{2}\sum_i(|B_i| \times |P_i|)$.

For two indexed partition δ_1 and δ_2, for example the following relationship measure can be defined:

information similarity measure: $ISIM(\delta_1, \delta_2) = IQ(\delta_1) - IQ(\delta_1|\delta_2)$.

IV. Application of Indexed Partitions to Logic Synthesis

Indexed partitions are the method for expressing compatibility relation COM on cubes in different way than blankets and information sets. Some existing algorithms based on blankets and information sets can be adopted to work with indexed partitions.

In [12] an efficient heuristic method for computing minimal input support and parallel decomposition for Boolean function was presented that can be easily implemented with use of concept indexed partitions. Similarly, efficient heuristic method for input support selection proposed in [11] can be used with new concept to reduce the search space to a manageable size while keeping the high-quality solutions in the reduced space.

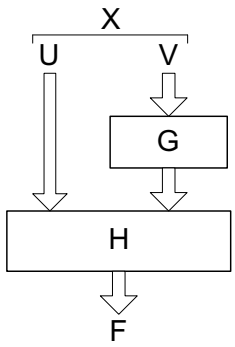

Fig. 1. Schematic representation of the serial decomposition.

It is also possible to propose algorithms based on new concept that have much better efficiency that similar algorithms constructed for blankets or information sets.

This section demonstrates the examples of application of introduced indexed partition concept in logic synthesis of Boolean functions specified by cubes.

A. Serial Functional Decomposition

The set X of function's input variable is partitioned into two subsets: *free variables* U and *bound variables* V, such that $U \cup V = X$. Assume that the input variables $x_1, \ldots, x_n$ have been relabeled in such way that:

$$U = \{x_1, \ldots, x_r\} \text{ and}$$
$$V = \{x_{n-s+1}, \ldots, x_n\}.$$

Consequently, for an n-tuple x, the first r components are denoted by x^U, and the last s components, by x^V.

Let F be a Boolean function, with $n > 0$ inputs and $m > 0$ outputs, and let (U, V) be as above. Assume that F is specified by a set F of the function's cubes. Let G be a function with s inputs and p outputs, and let H be a function with $r + p$ inputs and m outputs. The pair (G, H) represents a serial decomposition of F with respect to (U, V), if for every minterm b relevant to $F, G(b^V)$ is defined, $G(b^V) \in \{0,1\}^p$, and $F(b) = H(b^U, G(b^V))$. G and H are called blocks of the decomposition (Fig. 1).

In [1] a theorem based on concept of cubes can be found that describes the existence of the serial decomposition.

Theorem 1.

Let β_V, β_U, and β_F be blankets induced on the function's F input cubes by the input sub-sets V and U, and outputs of F, respectively.

If there exists a blanket β_G on the set of function F's input cubes such that $\beta_V \leq \beta_G$, and $\beta_U \bullet \beta_G \leq \beta_F$, then F has a serial decomposition with respect to (U, V).

As follows from Theorem 1 the main task in constructing a serial decomposition of a function F with given sets U and V is to find a blanket β_G which satisfies the condition of the theorem. The number of blocks of blanket grows exponentially with the number of variables used to induce this blanket. Usually set V is small (up to 7 variables). However, for large

functions (with many input and output variables) β_U and β_F may have large number of blocks, thus the test of condition from Theorem 1 can be computationally complex.

The Theorem 1 can be re-expressed using the concept of indexed partitions.

Theorem 2.

Let δ_V, δ_U, and δ_F be indexed partitions induced on the function's F input cubes by the input sub-sets V and U, and outputs of F, respectively.

If there exists a indexed partition δ_G on the set of function F's input cubes such that $\delta_V \leq \delta_G$, and $\delta_U \bullet \delta_G \leq \delta_F$, then F has a serial decomposition with respect to (U, V).

In practice, even large Boolean functions of many input variables are often specified by relatively small number of cubes. Since the size of indexed partition depends on number of cubes the test from Theorem 2 can generally be computationally much less complex than the test based on blankets.

B. R-admissibility Concept

The r-admissibility concept was introduced in [28]. The r-admissibility test allows to obtain the set U of free variables for which there may exist function G (generally with t outputs) such that $|U| + t < n$, where $n = |X|$.

More information on application of the r-admissibility in logic synthesis can be found in [29].

Let $\beta_1, \ldots, \beta_k$ be blankets induced by k variables from free set U on S the set of cubes of a function F. The set of blankets $\{\beta_1, \ldots, \beta_k\}$ is r-admissible in relation to blanket β_F (induced by output variables) if and only if there is a set $\{\beta_{k+1}, \ldots, \beta_r\}$ of two-block blankets such that the product β of blankets $\{\beta_1, \ldots; \beta_k\} \bullet \{\beta_{k+1}, \ldots, \beta_r\}$ satisfies the inequality $\beta \leq \beta_F$, and there does not exist any set of $r-k-1$ two-block blankets which meets this requirement.

The r-admissibility has the following interpretation. If a set of blankets $\{\beta_1, \ldots, \beta_k\}$ is r-admissible, then there might exist a serial decomposition of F (Fig. 1) in which component H has r inputs: k primary inputs corresponding to free input variables which induce $\{\beta_1, \ldots, \beta_k\}$ and $r - k$ inputs being outputs of G. Thus, to find a decomposition of F in which component H has r inputs, we must find a set of input variables which induces an r-admissible set of input blankets. The following corollary can be applied to check whether or not a given set of input blankets is r-admissible.

Corollary 1. For $\beta \leq \gamma$, let $\beta|\gamma$ denote the quotient blanket and $\epsilon(\beta|\gamma)$ be the number of elements in the largest block of $\beta|\gamma$. Let e$(\beta|\gamma)$ be the smallest integer equal to or larger than $\log_2(\epsilon(\beta|\gamma))$ (i.e., $e(\beta|\gamma) = \lceil \log_2(\epsilon(\beta|\gamma)) \rceil$).

Definition 6. A *r-admissibility* of the two-block blankets set $\{\beta_1, \ldots, \beta_k\}$ on S in relation to the blanket γ on S, is defined as $r = k + e(\beta_1, \bullet \ldots, \bullet\beta_k|\gamma)$.

The r-admissibility may also have the following interpretation. If a blanket β (being product of blankets $\beta_1, \ldots, \beta_k$) is r-admissible in relation to the blanket γ, then there is required blanket α with at least $\epsilon(\gamma|\beta)$ blocks to satisfy condition $\beta \bullet \alpha \leq \gamma$. There are required at least $e(\gamma|\beta)$ two-block blankets such that they product can give α.

R-admissibility can also be computed using concept of indexed partitions.

Corollary 2. For indexed partitions $\delta \leq \theta$, let $\chi(\theta|\delta)$ denote the chromatic number of incompatibility graph described by quotient indexed partitions $\theta|\delta$. Let $e(\theta|\delta)$ be the smallest integer equal to or larger than $\log_2(\chi(\theta|\delta))$.

Definition 7. A *r-admissibility* of indexed partitions δ (being product of k indexed partitions $\delta_1, \ldots, \delta_k$) in relation to the indexed partitions θ, is defined as $r = k + e(\theta|\delta)$.

Since there are efficient heuristics for computing graph coloring the optimal or sub-optimal r-admissibility of given set of variables can be efficiently evaluated even for large Boolean function.

V. RESULTS

In this section efficiency of synthesis algorithms based on concept of indexed partitions and on concept of blankets is compared. The set of C++ classes has been implemented to model operations on blankets as well as on indexed partitions. For comparison functions computing r-admissibility, as well as testing serial decomposition existence has been implemented for both: blanket calculus and indexed partition calculus. All experiments were performed on PC with Phenom II X6 110T @ 3.3 GHz and 6 GB of RAM.

For experiment several benchmark Boolean with functions have been selected. Table II presents the number of inputs ($|X|$), outputs ($|Y|$) and the number of cubes ($|S|$) used to specify Boolean function used in comparison.

In Table III there are presented results of comparison of functional decomposition evaluation time (in seconds) for several sizes of bound set V. The comparison was performed for V set size ranging from 3 to 6. In the experiment 10 randomly selected V sets for every size of bound set were generated and used to evaluate blanket based, as well as indexed partition based decompositions. For all benchmarks time necessary for evaluation one single functional decomposition was measured as average value of time needed to evaluate every 10 bound sets. Evaluation for blanket based decomposition was performed using algorithm implementing decomposition test given by Theorem 1, while for evaluation

TABLE II
BENCHMARK FUNCTIONS USED IN COMPARISON

Benchmark	$\|X\|$	$\|Y\|$	$\|S\|$
table3	14	14	175
alu4	14	8	1028
misex3	14	14	1848
b12	15	9	431
table5	17	15	158
t2	17	16	301
opa	17	69	342
shift	19	16	100
in2	19	10	137

TABLE III
COMPARISON OF FUNCTIONAL DECOMPOSITION EVALUATION

	Blanket based decomposition				Indexed partition based decomposition			
Benchmark	$\|V\|=3$	$\|V\|=4$	$\|V\|=5$	$\|V\|=6$	$\|V\|=3$	$\|V\|=4$	$\|V\|=5$	$\|V\|=6$
table3	0.0020	0.0062	0.0268	0.1108	0.0005	0.0005	0.0005	0.0011
alu4	0.0195	0.0058	0.0028	0.0048	0.0031	0.0034	0.0044	0.0056
misex3	0.0371	0.0119	0.0133	0.0368	0.0061	0.0067	0.0075	0.0091
b12	0.0173	0.0066	0.0037	0.0059	0.0013	0.0014	0.0022	0.0072
table5	0.0027	0.0066	0.0304	0.1388	0.0003	0.0005	0.0006	0.0009
t2	0.0009	0.0013	0.0036	0.0129	0.0008	0.0031	0.0013	0.0019
opa	0.0017	0.0023	0.0058	0.0226	0.0008	0.0011	0.0016	0.0023
shift	5.5664	1.5586	0.4719	0.1401	0.0003	0.0003	0.0006	0.0012
in2	0.0323	0.0298	0.0796	0.4625	0.0005	0.0005	0.0006	0.0011
Total time	5.6800	1.6290	0.6379	0.9354	0.0136	0.0175	0.0192	0.0304
Speedup factor	1	1	1	1	419	93	33	31

TABLE IV
COMPARISON OF R-ADMISSIBILITY COMPUTATION

	Blanket based r-admissibility evaluation								Indexed partition based r-admissibility evaluation							
	$\|V\|=3$				$\|V\|=4$				$\|V\|=3$				$\|V\|=4$			
Benchmark	$\|U\|$	R	T	$\|\beta_U\|$	$\|U\|$	R	T	$\|\beta_U\|$	$\|U\|$	R	T		$\|U\|$	R	T	
table3	11	14.2	0.0031	177	10	13.1	0.0031	158	11	14.2	0.0015		10	13.1	0.0016	
alu4	11	15.0	0.0265	2029	10	14.0	0.0078	1018	11	15.0	0.0234		10	14.0	0.0218	
misex3	11	15.0	0.0484	1421	10	14.0	0.0125	646	11	15.0	0.0655		10	14.0	0.0655	
b12	12	16.0	0.0250	1544	11	15.0	0.0093	885	12	16.0	0.0047		11	15.0	0.0063	
table5	14	17.1	0.0031	195	13	16.0	0.0031	211	14	17.1	0		13	16.2	0.0016	
t2	14	17.6	0.0016	114	13	16.7	0.0016	102	14	17.6	0.0031		13	16.7	0.0016	
opa	14	19.1	0.0047	171	13	18.3	0.0031	150	14	19.1	0.0032		13	18.4	0.0031	
shift	16	21.0	5.3555	24159	15	20.0	1.5647	13702	16	21.0	0		15	20.0	0.0016	
in2	16	19.7	0.0312	510	15	18.9	0.0188	472	16	19.7	0.0016		15	18.9	0	

of indexed partition based decomposition test from Theorem 2 was used. As can be noticed, with the growth of function size the superiority of indexed partition based algorithm over blanket based implementation also increases. In average, for benchmarks used in the comparison, the new method was 419 times faster. But in case of *shift* benchmark evaluation of decomposition for 3 variables in V set using indexed partitions was over 17 956 times faster than method based on blankets.

Table IV presents the result of comparison of both methods for r-admissibility of U set computation. Comparison was performed for two sizes of set V: 3 and 4 variables. Set U was computed as $U = X - V$. Table presents the size of set U (column labeled $|U|$) for every benchmark. Again, the same 10 randomly selected V sets for every size of bound set were used as in previous experiment. Column labeled T presents the time required for evaluation of r-admissibility of single U set measured as average value of time necessary to evaluate all 10 free sets. Column R holds average r-admissibility for these randomly selected free sets. Additionally for method based on blanket calculus average number of blocks in blanket β_U was shown in column labeled $|\beta_U|$.

It can be noticed that execution time of algorithm based on blankets strongly depends on number of blocks in blankets used in evaluation. Since for function of many input variables it is more likely that blankets used in computation will have large number of blocks this method is practically useless for function of more than 20 inputs. Algorithm based on indexed partitions is sensitive on number of cubes used to describe Boolean function. However, even for large functions (in terms of input variables, as well as number of cubes) this methods performs very well.

The quality of solutions delivered by both methods (column R) is very similar. Method based on indexed partitions uses heuristic algorithm for incompatibility graph coloring, hence in two cases the average r-admissibility R is slightly worse.

VI. CONCLUSIONS

Many practical logic multiple-output, partially specified Boolean function can be efficiently represented by cubes. There are methods that use blanket calculus to compute functional decomposition of such functions. Unfortunately computational complexity of operations on blankets highly depends on the size of Boolean function. This makes algorithms based on this concept practically useless for functions of many input and output variables.

The concept of indexed partitions introduced in this paper allows constructing algorithms that perform very well even for large Boolean functions. Experimental results presented here prove that indexed partition calculus can be efficiently used for functions of many input variables. Decomposition, as well as r-admissibility computation using indexed partitions delivers solution of similar quality as algorithms based on blankets, but does it many times faster.

REFERENCES

[1] J. A. Brzozowski and T. Łuba, "Decomposition of Boolean Functions Specified by Cubes," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 9, pp. 377–417, 2003.
[2] T. Łuba and H. Selvaraj, "A General Approach to Boolean Function Decomposition and its Applications in FPGA-Based Synthesis," *VLSI Design*, vol. 3, no. 3-4, pp. 289–300, 1995.
[3] M. Rawski, "Decomposition of Boolean Function Sets," *Electronics and Telecommunications Quarterly*, vol. 53, no. 3, pp. 231–249, 2007.

[4] T. Sasao, M. Matsuura, Y. Iguchi, and S. Nagayama, "Compact BDD Representations for Multiple-Output Functions and Their Application," in *Proceedings of ISMVL*, 2001, pp. 207–212.
[5] C. Scholl, *Functional Decomposition with Application to FPGA Synthesis*. Kluwer Academic Publisher, 2001.
[6] R. Ashenhurst, "The Decomposition of Switching Functions," in *Proceedings International Symposium Theory of Switching*, vol. 29, 1959, pp. 74–116.
[7] H. A. Curtis, *A New Approach to the Design of Switching Circuits*. New Jersey: D. van Nostrand company, Princeton, 1962.
[8] R. M. Karp, "Functional Decomposition and Switching Circuit Design," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 291–335, June 1963.
[9] J. Hartmanis and R. E. Stearns, *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall, 1966.
[10] H. Selvaraj, M. Nowicka, and T. Łuba, "Performance Oriented Decomposition Strategies for FPGA Based Technology Mapping," in *International Conference VLSI for Signal Processing*, Chennai, India, 1998.
[11] M. Rawski, L. Jóźwiak, and T. Łuba, "Functional Decomposition with an Efficient Input Support Selection for Sub-Functions Based on Information Relationship Measures," *Journal of Systems Architecture*, vol. 47, no. 2, pp. 137–155, February 2001.
[12] L. Jóźwiak, "Information Relationships and Measures: An Analysis Apparatus for Efficient Information System Synthesis," *23rd EUROMICRO Conference*, pp. 13–23, 1-4 September 1997.
[13] A. Chojnacki and L. Jóźwiak, "High-Quality FPGA Designs Through Functional Decomposition with Sub-Function Input Support Selection Based on Information Relationship Measures," in *IEEE International Symposium on Quality of Electronic Design*. San Jose, USA: IEEE Computer Society Press, Los Alamitos, CA, USA, 26-28 March 2011, pp. 409–414.
[14] ——, "An Effective and Efficient Method for Functional Decomposition of Boolean Functions Based on Information Relationships Measures," in *Design and Diagnostics of Electronic Circuits and Systems DDECS'2000*, Smolenice, Slovakia, April 2000.
[15] C. Y. Lee, "Representation of Switching Circuits by Binary-Decision Diagrams," *The Bell System Technical Journal*, pp. 985–999, 1959.
[16] S. B. Akers, "Binary Decision Diagrams," *IEEE Transactions on Computers*, vol. 27, no. 6, pp. 509–516, 1978.
[17] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, vol. 35, no. 6, pp. 677–691, 1986.
[18] T. Sasao and M. Fujita, *Representations of Discrete Functions*. Kluwer Academic Publishers, 1996.
[19] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.
[20] M. Nowicka, T. Łuba, and M. Rawski, "FPGA-Based Decomposition of Boolean Functions, Algorithms and Implementation," *Advanced Computer Systems*, pp. 502–509, 1999.
[21] M. Rawski, "Efficient Variable Partitioning Method for Functional Decomposition," *Electronics and Telecommunications Quarterly*, vol. 53, no. 1, pp. 63–81, 2007.
[22] ——, *Evolutionary Algorithms*. Intech, 2011, ch. Evolutionary Algorithms in Decomposition-Based Logic Synthesis.
[23] M. Rawski, H. Selvara, T. Łuba, and P. Szotkowski, "Multilevel Synthesis of Finite State Machines Based on Symbolic Functional Decomposition," *International Journal of Computational Intelligence and Applications*, vol. 6, no. 2, pp. 257–271, June 2006, imperial College Press.
[24] T. Łuba, "Decomposition of Multiple-Valued Functions," in *25th International Symposium on Multiple-Valued Logic*, Bloomington, Indiana, 1995, pp. 256–261.
[25] J. Lewandowski, M. Rawski, and H. Rybiński, "Application of Parallel Decomposition for Creation of Reduced Feed-Forward Neural Networks," in *Proceedings of the International Conference, Rough Sets and Intelligent Systems Paradigms*. Warsaw, Poland: Springer, 28-30 June 2007, pp. 564–573, lecture Notes in Artificial Intelligence, Subseries of Lecture Notes in Computer Science.
[26] H. Selvaraj, P. Sapiecha, and T. Łuba, "Functional Decomposition and Its Applications in Machine Learning and Neural Networks," *International Journal of Computational Intelligence and Applications*, vol. 1, no. 3, pp. 259–271, 2001.
[27] A. Chojnacki and L. Jóźwiak, "Multi-Valued Sub-Function Encoding in Functional Decomposition Based on Information Relationship Measures," in *30th IEEE International Symposium on Multiple-valued Logic*, Portland, OR, May 2000.
[28] T. Łuba, "Multi-Level Logic Synthesis Based on Decomposition," *Microprocessors and Microsystems*, vol. 18, no. 8, pp. 429–437, 1994.
[29] G. Borowik, T. Łuba, and P. Tomaszewicz, "A Notion of R-Admissibility and its Application in Logic Synthesis," in *DESDes'09, Preprints of the 4th IFAC Workshop Discrete-Event System Design 2009*, Spain, 6–8 October 2009, pp. 207–212.