High-quality Sub-function Construction in Functional Decomposition Based on Information Relationship Measures

Lech Jóźwiak and Artur Chojnacki
Eindhoven University of Technology
P.O. Box 513, EH 10.16, 5600 MB Eindhoven, The Netherlands
{L.Jozwiak, A.Chojnacki}@tue.nl

Abstract

Functional decomposition seems to be the most effective circuit synthesis approach for look-up table (LUT) FPGAs, (C)PLDs and complex gates. In the functional decomposition that targets LUT FPGAs, the circuit is constructed by recursively decomposing a given function and its sub-functions until each of the resulting sub-functions can be directly implemented with a LUT. The choice of sub-functions constructed in this process decides the quality of the resulting multi-level circuit expressed in terms of the logic block count and speed. In this paper, we propose a new effective and efficient method for the sub-function construction, and we consider its application in our circuit synthesis tool that targets LUT-based FPGAs. The method is based on the information relationship measures. The experimental results demonstrate that the proposed approach leads to extremely fast and very small circuits.

1 Introduction

In the case of look-up table (LUT) FPGAs, (C)PLDs and complex gates, constraints are not imposed on the function type that a certain logic building block can implement, but rather on various structural parameters of logic blocks (e.g. the maximum number of inputs and outputs in a programmable block, or serial and parallel transistors in a gate) and on the interconnections between logic blocks. A logic block is able to implement any function with limited dimensions. However, the traditional logic synthesis methods do not consider hard structural constraints. Moreover, they only consider some very special cases of possible implementation structures involving some minimal functionally complete systems of logic operators (e.g. AND+OR+NOT). If the actual synthesis target strongly differs from the used minimal system (e.g. if it involves LUT FPGAs), no form of technology mapping can guarantee proper final results, because the initial synthesis is performed without a close relation to the actual target. Therefore, much research has recently taken place in the field of functional decomposition [2][3][6-8][10][12][15-17]. A sub-function in functional decomposition can be any function that satisfies certain specific structural constraints. This enables adequate synthesis for the targets mentioned above, and in particular, for LUT FPGAs.

2 Functional decomposition

The functional decomposition approach was considered by Shannon [14], Povarov [9] and Ashenhurst [1], and extended by Roth and Karp [11], Curtis [4] and Jóźwiak [6]. All known decomposition schemes for discrete functions are some special cases of the general decomposition scheme presented in [6]. The most promising recent approaches to functional decomposition are perhaps the BDD-based approach [2][12][17] and the information-based approach [3][6][8][10] being the subject of this paper.

The BDD-based approach can be sub-divided into the substitution [2] and cutting approach [2][12][17]. The substitution consists of replacing some BDD sub-graphs by new variables representing them. It is limited to completely specified functions and fully determined by the BDD structure for a certain variable ordering. The cutting approach is more general and it implements the Roth-Karp decomposition [11]. It consists of selecting a sub-function's input support that defines a BDD cut, properly encoding the sub-function, and expressing the original function in the new sub-function's output variables. If a single BDD is used to represent a Boolean function, the approach is limited to completely specified functions [17], but it can account for the incompletely specified functions by using two BDDs [12] (e.g. one for ON-set and another one for (ON∪DC)-set) or using a modified BDD (e.g. a three terminal diagram). The BDD-based methods can account for specific non-disjoint decompositions either explicitly (non-disjunctive cut sets [12]) or implicitly (specific encoding [17]). The sub-function input support selection is either exhaustive [12] (not efficient for large functions) or improves an initial support in a greedy trial and error procedure [17] (not effective for large functions). The encoding either minimizes the number of nodes in the resulting BDD [2] or the input supports of the binary sub-functions [12][17], or tries to produce sub-functions common to as many as possible outputs of a multiple-output function [12][17].

Our information-based approach considers a discrete multiple output function (relation) as a computation process specification of an information processing system, and a circuit that implements the function (relation) as a structure of the system. This structure supports the specified computation process, but at the same time, satisfies specific constraints and optimises certain objectives. Information, its processing, distribution and transmission play a central role in our approach. The circuit synthesis process proposed by us aims at structuring the circuit in such a way that the hard constraints imposed by the logic building blocks and their interconnections are satisfied, and the circuit is quick and compact. This is achieved by constructing explicitly the sub-functions that fit directly in the logic building blocks and structuring the resulting binary network in such a way that its sub-networks for particular outputs converge rapidly. Moreover, the information flows in the network are ordered according to the information production and consumption, appropriately combined, compressed, and kept as local as possible. The network is composed of relatively independent and coherent parts. In this way, the interconnections are minimized and the sub-functions have smaller number of inputs and outputs, because they process the combined and compressed compatible information. To facilitate the information flow and structure analysis that is necessary to enable the proposed synthesis approach, an adequate analysis apparatus is necessary, which ensures the analysis of where and how a particular information is produced/consumed, analysis of the relationships (similarity, difference etc.) between various information flows, and quantitative characterization of the information flows and their relationships. All this is ensured by the apparatus of information relationships and measures proposed by us in [7]. The analysis results from this apparatus are used to control the functional decomposition process that implements the proposed circuit synthesis approach.

A Boolean function with at most k inputs is called ***k*-feasible**. If all sub-functions in a certain logic network are

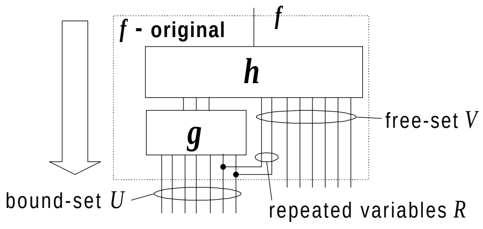


Figure 1. Single step of the functional decomposition

k-feasible, the network is k-feasible and it can be directly mapped into LUT FPGAs, where each logic building block is a k-input LUT (or CLB) that can implement any function up to k inputs (typically k = 4, 5, or 6).

In a single step of functional decomposition, function ***f*** being decomposed is divided into two sub-functions (see Fig.1): **predecessor sub-function *g*** (bound-set function) and **successor sub-function *h*** (image or composition function). To construct the sub-functions g and h, the input support X of f is divided into two (but not necessarily disjoint) subsets: **bound-set *U***, being the g's input support, and **free-set *V***, being a partial input support of h (see Fig. 1). Outputs of g constitute the remaining part of the h's support. This single decomposition step is recursively applied to both predecessor and successor sub-functions until a k-feasible network is constructed.

The choice of the predecessor sub-function g has a strong impact on the network structure that implements a given function f. This choice directly determines properties of the sub-network that implements g, and indirectly, of the sub-network that implements h. The g's outputs constitute a part of the h's input support. Moreover, the selection of variables to the g's input support ascertains the h's input support. In this way, the choice of the sub-functions in the multi-step decomposition process determines the quality of the resulting from this process multi-level logic network. It decides both the complexity of the resulting network (the logic block count and interconnection structure) and its speed (the number of the network's logic levels and interconnect length). In consequence, construction of the adequate sub-functions is a crucial problem in functional decomposition.

In this paper, we propose a new effective and efficient approach to solve this problem that is based on information relationship measures [7][8]. Experiments performed with our FPGA-targeted circuit synthesis tool that uses this approach clearly demonstrate that the general functional decomposition with sub-function construction based on information relationship measures results in high quality FPGA circuits.

3 Information and information relationships

Let us consider a finite set of elements S called symbols. Information about symbols pertains to the ability to distinguish certain symbols from other symbols. Table 1 shows the truth table of a multi-output Boolean function. Each row of the truth table (function's product term) is represented by a unique symbol from S. Through its two values 0 and 1, variable x_1 induces two compatibility classes on the symbols (terms): B^0={0,2,3,4} and B^1={1,2,3,5}. x_1 has value 0 (1) for each symbol in class B^0 (B^1) (don't care '-' means: 0 and 1). Variable x_1 is not able to distinguish between symbols 0, 2, 3, and 4, because they belong to the same compatibility class. x_1 is able to distinguish between 4 and 5, because they are not placed together in any compatibility class. In this way information is modeled with set systems [7][8].

Table 1. 3-input, 2-output Boolean function *f*

S	$x_1x_2x_3$	f_1f_2
0	000	00
1	111	00
2	-01	01
3	-10	01
4	011	11
5	100	11

A **set system** π on S is a collection of subsets $B_1, B_2, \dots, B_k$ of S (called blocks of π) such that: $\bigcup_i B_i = S$ and $B_i \not\subset B_j$ for $i \neq j$. The **product of two set systems** π_1 and π_2 represents combined information from both set systems and is defined as follows: $\pi_1 \bullet \pi_2 = \{B \mid \underset{B_1\in\pi_1}{\exists}\ \underset{B_2\in\pi_2}{\exists} B = B_1 \cap B_2 \wedge \underset{B_1'\in\pi_1}{\forall}\ \underset{B_2'\in\pi_2}{\forall} B \subseteq B_1' \cap B_2' \Rightarrow B = B_1' \cap B_2'\}$.

An **elementary information** describes the ability to distinguish a certain single symbol s_i from another single symbol s_j ($s_i,s_j \in S$ and $s_i \neq s_j$). Any set of such atomic portions of information can be represented by an **information set** IS [7][8] defined on $S \times S$ as follows: IS = {{s_i, s_j} | s_i is distinguished from s_j by the modeled information}. For instance, information given by set system $\pi_{x1}=\{\overline{0,2,3,4};\overline{1,2,3,5}\}$ induced by x_1, can be represented by information set IS(π_{x1})={0|1 0|5 1|4 4|5}.

Information relationships between variables or set systems representing various information streams can be analyzed by considering relationships between their corresponding information sets. In [7][8], an appropriate analysis apparatus is proposed for this aim: the theory of information relationships and measures. In particular, the relationship and measure expressing information similarity of two set systems π_1 and π_2 are defined in [7][8] as follows:

- **common information** CI (i.e. information that is present in both π_1 and π_2): CI(π_1,π_2) = IS(π_1) ∩ IS(π_2)
- **information similarity (affinity) measure ISIM**: ISIM(π_1, π_2) = |CI(π_1, π_2)|.

In [2][5][6] some normalized and weighted measures are also defined, by associating an appropriate importance weight $w(s_i|s_j)$ with each elementary information. The **weighted information similarity measure** is defined as follows:

$$\mathrm{WISIM}(\pi_1,\pi_2) = \sum_{s_i|s_j \in IS(\pi_1)\cap IS(\pi_2)} w(s_i \mid s_j),$$

where $w(s_i|s_j)$ is a **weighting function**.

The importance of information is related to its availability, i.e. the number of variables at which this information is present. Let f be a certain discrete function, X be a set of some input variables of f and ISS(X) be the set of information sets induced on the function's terms by particular variables from X. **Occurrence multiplicity *m*** of an elementary information $s_i|s_j$ from IS(f) in ISS(X) is defined as follows:

$$m(s_i \mid s_j)\Big|_{\mathrm{ISS}(X)}^{\mathrm{IS}(f)} = \sum_{x\in X} o(s_i \mid s_j)\Big|_{\mathrm{IS}(x)}^{\mathrm{IS}(f)}$$

where:

$$o(s_i \mid s_j)\Big|_{\mathrm{IS}(x)}^{\mathrm{IS}(f)} = \begin{cases} 1: \text{if } (s_i \mid s_j) \in (\mathrm{IS}(f) \cap \mathrm{IS}(x)) \\ 0: \text{ otherwise} \end{cases}$$

If $m(s_i \mid s_j)\Big|_{\mathrm{ISS}(X)}^{\mathrm{IS}(f)} = 1$, $s_i|s_j$ required by f is provided by only a single variable from X, then $s_i|s_j$ is called a **unique information** with respect to X. Unique information is of primary importance.

The formula:

$$km((s_i \mid s_j),k)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} = \begin{cases} 1: \text{if } m(s_i \mid s_j)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} = k \\ 0: \text{if } m(s_i \mid s_j)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} \neq k \end{cases}$$

divides the elementary information items into classes of equal multiplicity (**k-multiplicity**).

To ensure that the sum of weights of the less important information will not dominate the weight of the more important information, we use the following **normalization function *h***:

$$h(0)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} = 0,\ h(1)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} = 0\ ,$$
$$h(k)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} = h(k-1)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} + \sum_{(s_i|s_j)\in \mathrm{IS}(f)} km((s_i|s_j),k)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)}.$$

The **weighting function *w*** is defined as follows:

$$w(s_i \mid s_j)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} = \begin{cases} 0: \text{if } m(s_i \mid s_j)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} = 0 \\ 1: \text{if } m(s_i \mid s_j)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)} = 1 \\ \dfrac{2}{2^{m(s_i|s_j)\big|_{\mathrm{ISS}}^{\mathrm{IS}(f)}}\, h(m(s_i \mid s_j)\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)})\Big|_{\mathrm{ISS}}^{\mathrm{IS}(f)}} : \text{otherwise} \end{cases}$$

Example 1. The corresponding set systems and information sets for all inputs and outputs of the Boolean function shown in Table 1 are as follows:

$\pi_{f1}=\{\overline{0,1,2,3};\overline{4,5}\}$, $\pi_{f2}=\{\overline{0,1};\overline{2,3,4,5}\}$, $\pi_{x1}=\{\overline{0,2,3,4};\overline{1,2,3,5}\}$,

$\pi_{x2}=\{\overline{0,2,5};\overline{1,3,4}\}$, $\pi_{x3}=\{\overline{0,3,5};\overline{1,2,4}\}$,

$\mathrm{IS}(\pi_{f1}) = \{0|4\ 0|5\ 1|4\ 1|5\ 2|4\ 2|5\ 3|4\ 3|5\}$,
$\mathrm{IS}(\pi_{f2}) = \{0|2\ 0|3\ 0|4\ 0|5\ 1|2\ 1|3\ 1|4\ 1|5\}$,
$\mathrm{IS}(\pi_{x1}) = \{0|1\ 0|5\ 1|4\ 4|5\}$,
$\mathrm{IS}(\pi_{x2}) = \{0|1\ 0|3\ 0|4\ 1|2\ 1|5\ 2|3\ 2|4\ 3|5\ 4|5\}$,
$\mathrm{IS}(\pi_{x3}) = \{0|1\ 0|2\ 0|4\ 1|3\ 1|5\ 2|3\ 2|5\ 3|4\ 4|5\}$,
$\mathrm{CI}(\pi_{f1},\pi_{x1}) = \{0|5\ 1|4\}$, $m(0\,|\,5)\Big|^{\mathrm{IS}(f_1)}_{\mathrm{ISS}(x_1,x_2,x_3)}=1$

$m(1\,|\,4)\Big|^{\mathrm{IS}(f_1)}_{\mathrm{ISS}(x_1,x_2,x_3)}=1$ and $\mathrm{WISIM}(f_1,x_1) = 2$**.**

4 Overview of the method

Let $Y = \{y_i \mid i = 1\ldots n\}$ be the set of binary output variables of an incompletely specified multiple-output Boolean function f and $\pi_Y=\bullet\pi_{yi}$ be the product set system induced by these variables on the set of the function's terms (cubes). Let $\pi_U=\bullet\pi_{xi}$, where $x_i \in U$, be the product set system induced by the variables from U. Let $\pi_V=\bullet\pi_{xj}$, $x_j \in V$, be the product set system induced by the variables from V. Let π_g be the output set system of a sub-function g. The theorem that describes sufficient conditions for a single decomposition step [10] can be expressed using information sets as follows:

***Theorem 1*. Existence of serial decomposition**
If there is a set system π_g on f, such that $\mathrm{IS}(\pi_U)\supseteq\mathrm{IS}(\pi_g)$ and $\mathrm{IS}(\pi_g)\cup\mathrm{IS}(\pi_V)\supseteq\mathrm{IS}(\pi_Y)$, where: $\mathrm{IS}(\pi_U) = \bigcup_{xi\in U}\mathrm{IS}(\pi_{xi})$, $\mathrm{IS}(\pi_Y) = \bigcup_{yi\in Y}\mathrm{IS}(\pi_{yi})$, and $\mathrm{IS}(\pi_V) = \bigcup_{xi\in V}\mathrm{IS}(\pi_{xi})$, then the function f has a serial functional decomposition with respect to (U, V) in the form $f=h(g(U),V)$. □

Our circuit synthesis method constructs the circuit level by level from its primary inputs to primary outputs (bottom-up), by repeating the single decomposition step [3][8]. In the support of a certain level, only the variables from any lower level (primary inputs and/or logic blocks' outputs) can be used. The output variables of the logic blocks already built at the current level – that constitute the **cover-set *C*** - cannot be used in any bound-set of this level (see Fig. 2). At each level, the input support (primary inputs and/or intermediate variables) of the not yet synthesized part of a function being decomposed has to provide all information necessary to compute the function's output values (Theorem 1). Information necessary for computing the function's values is distributed across the current support variables. These variables also contain some redundant information. To implement the function, the decomposition network has to eliminate the redundant information, and preserve and restructure the required information. Therefore, each sub-function g should eliminate some redundant information, combine the required information delivered by its inputs, transfer the required information to its output and represent it in an appropriate manner. The *bound-set U* determines *what information is delivered* to a certain sub-function g. The g's *output set system* π_g determines *what information is transferred* to the g's outputs. U and π_g together define the multi-valued function of g. In order to implement this function in binary hardware, it has to be transformed into a set of binary functions, by assigning a binary code to each block of π_g. The *g's binary functions* determine *how the transferred information is represented* at the g's binary outputs.

The sub-function construction procedure is composed of the following steps:

1. Construct a limited set of the most promising bound-sets U and corresponding output set systems π_g,
2. Select the best π_g and corresponding U from the set constructed in step 2,
3. Construct an appropriate cluster of the binary functions that implement the multi-valued function g by encoding the selected π_g.

Each time a successive sub-function g is constructed, a new function h is computed by expressing f in new variables.

5 Construction and selection of the most promising bound-sets

Let X be the support of a function f at a certain decomposition step, C the cover set at this step, and $Z=\{x_i|x_i\in X\backslash C\}$ the set of variables that can be used at this step to build an input support U for g with maximum size k, $U \subseteq Z$, $1<|U|\leq k$. $\mathrm{NCIS}(f, C) = \mathrm{IS}(f)\backslash\mathrm{ISS}(C)$ represents the information required by f that is not covered by C. The g's output set system π_g is created by merging some blocks of the g's input set system π_U. The information that should be preserved during the merging is given by the preserved information set $\mathrm{PIS}(\pi_u, f, C, Z, k)$ =

$$\mathrm{IS}(\pi_U)\cap\mathrm{NCIS}(f,C)\cap\{s_i\mid s_j : m(s_i\mid s_j)\Big|^{IS(f)}_{ISS(Z)}\leq k\}.$$

In [10], we showed a strong positive correlation between the number of blocks in the set systems π_g used in the decomposition process and the number of LUTs

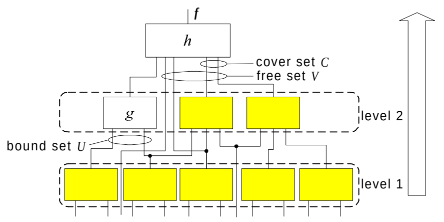


Figure 2. Bottom-up functional decomposition

and LUT-levels in the resulting multi-level network. Therefore, using an appropriate information relationship measure - the block merge-ability measure *bmergability* defined in [3] - the input support construction procedure tries to construct a limited set of the bound-sets U that result in π_g's that preserve as much as possible information from PIS, and have as few blocks as possible. If *bmergability* is extremely high or extremely low, many blocks of π_U can be merged without or with small loss of information that should be preserved [3]. The most promising input supports are constructed by clustering the variables from Z that are best correlated with each other according to the lowest and highest values of *bmergability*. For each obtained this way support U, an appropriate output set system π_g is constructed from π_U, using a heuristic minimal coloring of the block incompatibility graph IG(π_U). A node of IG(π_U) represents a block of π_U. There is an edge between two nodes of IG(π_U) iff the two corresponding blocks of π_U are incompatible, i.e. their merging will remove some information required by f, not covered by C and only provided by variables from U.

The final support selection is based on the support quality function for implementation with k-input LUTs, $Q(U,\pi_g,\pi_f,k)$, which is expressed by the following formula:

$$Q(U,\pi_g,\pi_f,k) = \frac{\mathrm{ISIM}(\pi_g,\pi_f)\,conv(U,\pi_g)\,ic\_use(\pi_g)}{ic\_\cos t(U,\pi_g,k)},$$

where ISIM(π_g,π_f) is the information similarity measure defined in Section 2, $conv(U, \pi_g)=|U|-l$ is the convergence factor that denotes the difference between the numbers of input and output variables of g ($l=\lceil log_2|\pi_g|\rceil$), $ic\_use(\pi_g) = |\pi_g|/2^l$ is the usage of the "information channel" induced by the output variables of g, and

$$ic\_\cos t(U,\pi_g,k) = \begin{cases} l^2\,|U| & : \text{for } 2 \le |U| \le k \le 4 \\ l^2\,|U|\,2^{|U|-4} & : \text{for } 5 \le |U| \le k \le 6 \\ l^2\,|U|\,2^{|U|-k} & : \text{for } |U| > k \end{cases}$$

represents the cost of the "information channel" (k-input LUTs, k > 4, are composed of 2^{k-4} 4-LUTs; coefficient $l^2|U|$ is found experimentally).

The support U with the highest value of Q(U,π_g,π_f,k) is selected to become the actual support for g. A more precise description of the input support construction and selection procedures can be found in [3].

6 Encoding of the multi-valued sub-functions

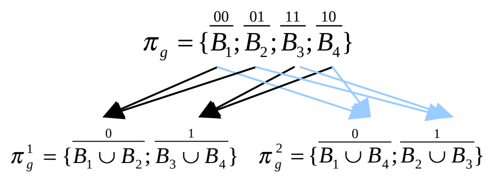
The selected support U and its corresponding set system π_g together define the multi-valued function of g, G:$\pi_u \rightarrow \pi_g$, where each particular value B_g of this function corresponds to a block of the set system π_g. The number of values of the function is equal to the number of blocks of π_g. In order to implement the multi-valued function in binary hardware, it has to be transformed into a set of binary functions by assigning a binary code to each block of π_g.

The binary code assignment implicitly defines a set of two-block set systems $\{\pi_g^i\}$ (i =1…l) - one two-block set system π_g^i for each binary output variable of g (see Fig. 3). For a minimum-length encoding, this set involves $l = \lceil log_2|\pi_g|\rceil$ two-block set systems. Block B^0 of a particular π_g^i is the union of the π_g's blocks that have value 0 at the i-th position of the assigned code. Block B^1 is the union of the π_g's blocks that have value 1 at the i-th code position. Usually the codes with minimum length are used, because they maximally reduce the number of binary functions that implement g and the h's input support. The resulting network is usually more compact and easier to decompose, when the number of the g's outputs is smaller. In the work reported in this paper, we also use the minimum length encoding.

It is possible to build a set of l two-block set systems $\{\pi_g^i\}$ (a set of l binary functions) with less items of unique or almost unique information than in the original π_g (multi-valued function). This is achieved by repeating the unique or almost unique information items in many different set systems π_g^i and results in a higher occurrence multiplicity m of the repeated items. The originally unique information items become non-unique. With growing repetition of the originally unique or almost unique information items at different binary outputs of g, function h tends to be easier to decompose. The information originally most difficult to transfer - the unique or almost unique information – is made easier to transfer, because it is present at more outputs of g being inputs to h. Moreover, information repetition causes growth of common information computed by different binary functions of g, and thus increases the chance for good common sub-functions for various binary functions of g. Therefore, our encoding procedure solves the following encoding problem:

Find such minimum length assignment of binary codes to blocks of π_g that the number of unique or almost unique elementary information items in $\{\pi_g^i\}$ is minimal.

The g's output set system π_g is created by merging some blocks of the g's input set system π_U that is induced by the selected support U. Even if particular information is originally not unique, i.e. it is provided by several input variables from $Z=\{x_i|x_i\in X\backslash C\}$, it may become unique, if it


$\pi_g = \{\overset{00}{B_1};\overset{01}{B_2};\overset{11}{B_3};\overset{10}{B_4}\}$
$\pi_g^1 = \{\overset{0}{B_1 \cup B_2};\overset{1}{B_3 \cup B_4}\}$
$\pi_g^2 = \{\overset{0}{B_1 \cup B_4};\overset{1}{B_2 \cup B_3}\}$
Figure 3. Binary code assignment

is delivered only by the variables from U, 9and in consequence, only by the multi-valued variable corresponding to π_g.

In general, merging some blocks of a set system reduces the amount of information provided by this set system. Let us define the **block merging cost *bmc*** for any two blocks of π_g as the sum of weights of the elementary information items removed by the merging:

$$bmc(B_k, B_l) = \sum_{s_i \in B_k \wedge s_j \in B_l \wedge (s_i | s_j) \in \mathrm{IS}(\pi_U)} w(s_i \mid s_j) \cdot$$

bmc describes how many and how important (unique, almost unique, etc.) elementary information items will be lost if we merge blocks B_k and B_l together.

Hamming distance *hd* of two binary vectors is the number of the corresponding positions at which these vectors differ (e.g. for c_1=00111 and c_2=10110, $hd(c_1, c_2)$=2.

If the values of two codes assigned to certain two blocks of π_g differ at a certain position i, then these two blocks of π_g are placed in two different blocks of $\pi_g^{\,i}$ and the information induced by the two blocks of π_g is also available in $\pi_g^{\,i}$. When the codes differ at several positions, the information is available at each binary variable corresponding to the code position at which they differ. The more different positions in the codes assigned to certain two blocks of π_g (higher Hamming distance), the more two-block set systems $\pi_g^{\,i}$ (binary output variables) provide information induced by these two blocks. In this way, we introduce multiplication of some information present in π_g in the set $\{\pi_g^{\,i}\}$. To decrease the number of unique and almost unique elementary

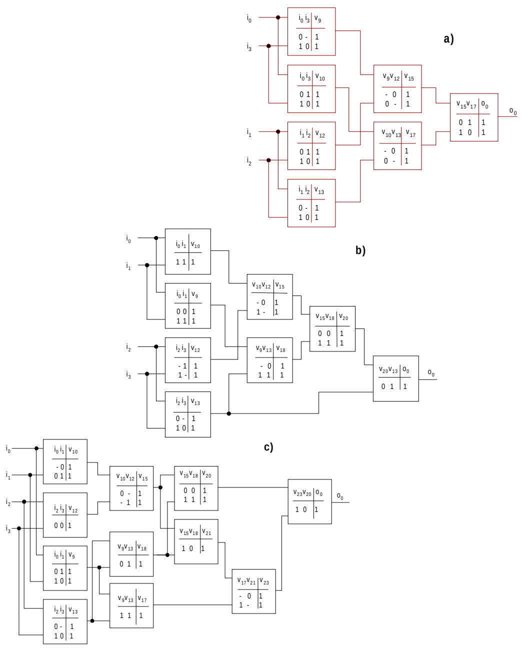


Figure 2 Different realizations of the symmetric Boolean function 1 of 4

Table 2 Symmetric Boolean function 1 of 4

S	$x_1x_2x_3x_4$	y
0	1000	1
1	0100	1
2	0010	1
3	0001	1
4	0000	0
5	--11	0
6	-11-	0
7	11--	0
8	-1-1	0
9	1--1	0
10	1-1-	0

information items in the set of information sets IS($\pi_g^{\,i}$) induced by the binary variables corresponding to $\pi_g^{\,i}$, the Hamming distance should be maximized for the pairs of codes assigned to the pairs of the π_g's blocks with the high merging cost *bmc*.

We implemented this encoding strategy by developing a fast greedy encoding algorithm executed inside a beam search. *beam* parameter limits the search space to a manageable size. The beam search selects *beam* most promising encoding directions, and the encoding algorithm constructs a set of encodings in these directions. Finally, the set of two-block set systems $\{\pi_g^{\,i}\}$ is selected that results in the lowest number of unique or almost unique information items.

First, the initial *beam* pairs of the π_g's blocks are selected. These are the pairs with the highest merging costs according to *bmc*. The encoding algorithm assigns some codes with maximum Hamming distance in between to each initial pair of blocks. The assigned codes are removed from the pool of the available codes. Then, the algorithm looks for the next pairs (B_k, B_l) of blocks with the highest merging costs until all blocks are encoded. If B_1 (B_2) from a certain selected pair is already encoded, the available code with the maximum Hamming distance to the code of B_1 (B_2) is selected and assigned to B_2 (B_1).

Example 2. Table 2 presents a 4-input symmetric Boolean function. The bound-set U={x_1, x_4} is selected and the g's output set system $\pi_g = \{\overline{1,2,4,6};\overline{5,6,7,8,9,10};\overline{0,3,5,6,7,8,10}\}$ is constructed from the input set system π_U. The following three different unique assignments are possible (other assignments are some permutations of $\pi_g^{\,i}$ and/or inversions one of $\pi_g^{\,i}$).

	$\pi_g^{\,1}\pi_g^{\,2}$	$\pi_g^{\,1}\pi_g^{\,2}$	$\pi_g^{\,1}\pi_g^{\,2}$
***B*₁**	00	00	01
***B*₂**	11	01	00
***B*₃**	01	11	11

The block merging costs are as follows:

B_2	6.1	
B_3	6.1	2
$Bmc(B_i,B_j)$	B_1	B_2

The cost of each assignment in terms of the unique elementary information items is shown in Table 3.

Table 3 Costs of the code assignments

	$\pi_g^1\pi_g^2$	$\pi_g^1\pi_g^2$	$\pi_g^1\pi_g^2$
B_1	00	00	01
B_2	11	01	00
B_3	01	11	11
cost	**8**	**8**	**4**

The assignment from column 3 is selected, because its cost is minimal. Figure 2 shows three different circuit realizations of the considered function corresponding to these three assignments. The selected assignment results in the smallest circuit from Figure 2a.

7 Experimental results

The method of the sub-function construction discussed in the previous sections was implemented in our CAD tool for FPGA-targeted circuit synthesis called IRMA2FPGAS (Information Relationship Measures Applied to FPGA Synthesis). In order to evaluate the quality of the proposed approach, we performed a number of experiments.

Table 4 presents the comparison of the results obtained from IRMA2FPGAS (column IRMA) to the results from SIS 1.3 [13] and three state-of-the-art functional decomposers (IMODEC (column IMO) [17], Sawada's [12] and Shen's [15] decomposers), for the MCNC benchmarks [18] (LUT count and number of LUT levels in the 5-feasible networks). In the case of SIS, we used the script dedicated to the LUT-based architectures proposed in [13]. All results are for single output functions. In almost all cases, our tool constructs better or equally good circuits than SIS, IMODEC and Shen's decomposer in terms of LUT count. In most cases, networks from our tool have far fewer logic levels than networks constructed by SIS or Sawada's decomposer, and never have more levels. The number of levels is not given in [15] and [17] for the other functional decomposers. Our tool constructs always the fastest circuits that have similar LUT counts as the slower circuits from Sawada's decomposer.

In the next experiment, we compared our IRMA2FPGAS to SIS using a wide spectrum of completely and incompletely specified Boolean functions, ranging from symmetric to strongly asymmetric functions. We generated a set of 10-input and 20-input completely specified functions with various characteristics, each having few hundreds terms. Then, we mutated the basic functions, by replacing 20%, 50% and 70% of 1 or 0 entries with “don't cares” in each completely specified function. More than 100 benchmarks were generated this way. In Tables 4 and 5, rows Symmetric represent results for symmetric functions or obtained from symmetric functions by “don't care “ insertion, rows Asymmetric - results for asymmetric functions or obtained from asymmetric by “don't care“ insertion, and rows All - total results for all functions. Unfortunately, SIS was unable to synthesize circuits for most of the 33 20-input benchmarks being symmetric functions or obtained from the symmetric functions by “don't care“ insertion (200 MB memory overflow in 24 cases). The global results of this experiment for all benchmarks synthesized by SIS are presented in Table 5. The networks from both tools were mapped onto CLBs of the Xilinx XC4000 FPGA family. Results of this experiment demonstrate that our IRMA2FPGAS constructs much better circuits than SIS. The circuits produced by IRMA2FPGAS are on average over 2 times faster and have 3 times less CLBs than the circuits synthesized by SIS. IRMA2FPGAS is especially effective for symmetric functions or obtained from symmetric functions by “don't care“ insertion. For these functions, the circuits produced by IRMA2FPGAS are on average 2.7 times faster and have almost 5 times less CLBs than the circuits synthesized by SIS.

We also compared IRMA2FPGAS to the three state-of-the-art FPGA-targeted commercial tools, using the same wide spectrum set of more than 100 generated functions

Table 4. Comparison of IRMA2FPGAS to other research tools on MCNC benchmarks
(5-LUT and LUT-level counts)

Circuit	#i	#o	Sawada		IMO	Shen	SIS 1.3		IRMA	
			Σ lut	D	Σ lut	Σ lut	Σ lut	D	Σ lut	D
5xp1	7	10	15	2	19	19	19	3	16	2
9sym	9	1	7	3	7	6	7	3	7	3
alu2	10	6	48	6	55	77	86	9	47	4
apex4	9	19	374	5	364	426	456	6	355	5
apex6	135	99	192	6	-	-	223	8	216	4
apex7	49	37	120	5	-	-	124	6	122	3
b9	41	21	53	4	57	92	47	4	46	3
clip	9	5	18	3	24	36	42	6	20	2
cordic	23	2	15	5	-	-	16	6	17	3
count	35	16	52	4	40	52	52	4	51	2
duke2	22	29	175	7	256	722	164	7	213	5
e64	65	65	-	-	389	544	544	4	305	3
f51m	8	8	12	3	16	16	20	4	15	2
misex1	8	7	12	2	17	16	14	3	13	2
misex2	25	18	40	3	40	43	40	4	39	2
misex3	14	14	195	9	-	-	534	10	276	7
misex3c	14	14	107	9	-	-	143	8	112	6
rd73	7	3	8	2	8	8	9	2	8	2
rd84	8	4	12	3	13	8	13	3	12	2
sao2	10	4	23	4	25	37	37	6	28	3
t481	16	1	5	3	-	-	8	4	5	2
vg2	25	8	44	5	-	-	51	6	44	3
z4ml	7	4	6	2	7	6	7	2	6	2

Table 5. Comparison of IRMA2FPGAS to SIS
(total number of CLBs and total delay)

Circuits	SIS 1.3				*IRMA2FPGAS*			
	Σ CLBs	(%)	Σ delay*	(%)	Σ CLBs	(%)	Σ delay*	(%)
Symmetric	1511	477%	1481	268%	317	100%	553	100%
Asymmetric	1382	212%	1098	157%	652	100%	698	100%
All	2893	299%	2578	206%	969	100%	1250	100%

* - [ns] Mapped onto device 4013xlbg256-09

as for the experiment with SIS. Results of this experiment can be found in [3]. The results demonstrate that IRMA2FPGAS constructs much better circuits than the commercial tools. The circuits produced by IRMA2FPGAS are on average over 1.5 times faster and have over 2 times less CLBs than the circuits produced by the best state-of-the-art commercial tool used for the experiment.

The computation time of our tool shows a slow quadratic growth with the number of the function's inputs and product terms, and a linear growth with the number of outputs. For functions having hundreds terms and up to 20 inputs, the computation time is in the order of single seconds, and up to 100 inputs, in the order of minutes (Pentium 3, 733 MHz, 128 MB). For functions having thousands terms and more than 20 inputs, the computation time is in the order of tenths of minutes.

8 Conclusion

In this paper, we proposed and discussed a new effective and efficient method for sub-function construction in functional decomposition. The method differs considerably from all other known methods. It implements our information-based approach to circuit synthesis, is based on the theory of information relationship measures [7][8] and uses novel evaluation functions to control the decomposition process. We implemented the method in an FPGA-targeted multi-level logic synthesis tool IRMA2FPGAS that is based on the bottom-up general functional decomposition [6]. The experimental results from our tool demonstrate the high quality of the proposed method. In almost all cases, our tool constructs better or equally good circuits than the other tools in terms of LUT count. Our tool constructs the fastest circuits. In most cases, networks from our tool have far fewer logic levels than networks constructed by SIS or Sawada's decomposer, and never have more levels. The circuits produced by IRMA2FPGAS are on average over 2 times faster and have 3 times less CLBs than the circuits synthesized by SIS, and they are on average over 1.5 times faster and consume over 2 times less CLBs than the circuits produced by the best state-of-the-art commercial tool.

9 References

[1] Ashenhurst, R.L.: The decomposition of switching functions, Proceedings of International Symposium on the Theory of Switching Functions, p. 74-116, April 1959.

[2] Chang S.-C. and Marek-Sadowska M.: Technology Mapping via Transformations of Function Graphs, IEEE ICCD'92, Cambridge, MA, October 92.

[3] Chojnacki, A., Jóźwiak, L.: High-quality FPGA Designs through Functional Decomposition with Sub-function Input Support Selection Based on Information Relationship Measures, IEEE International Symposium on Quality Electronic Design, ISQED'2001, San Jose, California, USA, March 26-28, 2001.

[4] Curtis, H.A.: A Generalized Tree Circuits, Journal of the Association for Computing Machinery, 8:484-496, 1961.

[5] Hartmanis, J., Stearns, R.E.: Algebraic Structure Theory of Sequential Machines, Englewood Cliffs, N.J.: Prentice-Hall, 1966.

[6] Jóźwiak, L.: General Decomposition and Its Use in Digital Circuit Synthesis, VLSI Design, vol.3, No 3, pp. 225 - 248, 1995.

[7] Jóźwiak, L.: Information Relationships and Measures - An Analysis Apparatus for Efficient Information System Synthesis, Proceedings of the 23rd EUROMICRO Conference, Budapest, Hungary, September 1-4, 1997, pp. 13-23, IEEE Computer Society Press.

[8] Jóźwiak, L.: Information Relationship Measures in Application to Logic Design, IEEE International Symposium on Multiple-Valued Logic, Freiburg Im Breisgan, Germany, May 20-22, 1998.

[9] Povarov G. N.: On Functional Separability of Boolean Functions, Lectures of the USSR Academy of Sciencies, Vol. 44, No 5, 1954.

[10] Rawski, M., Jóźwiak, L., Łuba, T.: The Influence of the Number of Values in Sub-functions on the Effectiveness and Efficiency of the Functional Decomposition, Proc. of the 25th EUROMICRO Conference, Milan, Italy, September 8-10, 1999.

[11] Roth, J.P. - Karp, R.M.: Minimization over Boolean Graphs, IBM Journal of Research and Development, April 1962.

[12] Sawada, H., Suyama, T., Nagoya A.: Logic Synthesis for Look-Up Table Based FPGAs using Functional Decomposition and Support Minimization, ICCAD'95.

[13] Sentovich, E. M., Singth, K., J., Lavagno, L., Moon, C., Murgai, R., Saldanha, A., Savoj, H., Stephan, P. R., Brayton, R. K., Sangiovanni-Vincentelli, A.: SIS: A System for Sequential Circuit Synthesis, Electronic Research Laboratory, University of California, Berkeley, Memorandum No. UCB/ERL M92/41, May 1992.

[14] Shannon, C. E.: The Synthesis of Two-Terminal Switching Circuits, The Bell Syst. Techn. Journal, Vol. 28, No 1, p. 59, 1949.

[15] Shen, W.-Z., Huang, J.-D., Chao, S.-M.: Lambda Set Selection in Roth-Karp Decomposition for LUT-based FPGA Technology Mapping, 32nd ACM/IEEE Design Automation Conference, 1995.

[16] Wan, W., Perkowski, M.: A New Approach to the Decomposition of Incompletely Specified Multi-Output Functions Based on Graph Coloring and Local Transformations and Its Application to FPGA Mapping, European Design Automation Conference, EURO-DAC ' 92. pp. 230-5.

[17] Wurth, B., Schlichtmann, U., Eckl, K., Antreich, K.J.: Functional Multiple-Output Decomposition with Application to Technology Mapping for Look-up Table Based, FPGAs, ACM Transactions on Design Automation of Electronic Systems, Vol. 4, No. 3, July, 1999.

[18] Collaborative Benchmarking Laboratory, Department of Computer Science at North Carolina State University, http://www.cbl.ncsu.edu/