Information Relationships and Measures in Application to Logic Design

Lech Jóźwiak
Eindhoven University of Technology
Faculty of Electrical Engineering
P.O. Box 513, EH 10.25
5600 MB Eindhoven, The Netherlands
e-mail:LECH@ics.ele.tue.nl

Abstract

In this paper, the theory of information relationships and relationship measures is considered and its application to logic design is discussed. This theory makes operational the famous theory of partitions and set systems of Hartmanis. The information relationships and measures enable us to analyze relationships between the modeled information streams and constitute an important analysis apparatus that can be used for analysis and synthesis of various information systems. It can be applied in logic design, pattern analysis, machine learning, and other fields. This paper shows how to apply the relationships and measures to logic design when using as examples three important problems: input support minimization, parallel decomposition and serial functional decomposition. The application examples and experimental results presented in the paper demonstrate the high potential of the information relationships and measures in solving logic synthesis problems.

1. Introduction

The opportunities created by modern microelectronic technology cannot be fully exploited, because of weaknesses of the traditional logic synthesis methods. Particularly in the case of (C)PLDs, look-up table FPGAs and complex CMOS gates, the constraints are imposed not on the function type a certain block can implement, but on various building block structural parameters (e.g. the number of inputs, outputs, product terms in a building block or the number of serial and parallel transistors in a gate) and on interconnections between the building blocks. A block is able to implement any function with limited dimensions. On the other hand, the traditional logic synthesis methods do not consider hard structural constraints. In principle, they are devoted to only some very special cases of possible implementation structures involving some minimal functionally complete systems of logic gates (e.g. AND+OR+NOT, AND+EXOR, MUX). They require a post synthesis technology mapping for another implementation structures. If the actual synthesis target strongly differs from these minimal systems, e.g. involves a lot of complex gates, look-up table FPGAs or (C)PLDs, any technology mapping cannot guarantee a good result if the initial synthesis was performed without close relation to the actual target.

Therefore, there is presently much research in the field of general (functional) decomposition of combinational circuits and sequential machines [3]-[10] [12] [14] [15] [17] [18] [20]-[22]. The most promising recent approaches in pattern analysis, knowledge discovery, machine learning, decision systems, data bases, data mining etc. are also based on general functional decomposition [11] [12] [13] [16] [19] [23]. In [5], I presented the theory of general decomposition and explained how the theory can be used for the synthesis of sequential and combinational circuits. For large circuits however, the number of possible decompositions is so great that it becomes necessary to construct only the most promising decompositions, using the general decomposition theorems presented in [5] together with the appropriate heuristic evaluation functions and selection mechanisms. These evaluation and selection mechanisms must limit the search space to a manageable size while keeping high-quality solutions in the limited space. In particular, the analysis and evaluation of relationships between information in various information streams of a considered system is here of primary importance. In [8] I proposed the fundamental analysis apparatus for this aim: the theory of information relationships and measures.

The information relationships and measures serve for analysis and estimation of information and information interrelationships in discrete systems, as modeled by any sort of discrete relations, functions and sequential machines. They can be applied for both the binary as well as the multiple-valued and symbolic systems, in many fields of modern engineering and science, including logic and architecture synthesis for VLSI systems [3] [10] [12] [14] [15] [17] [18] [20]-[22], pattern analysis, knowledge discovery, machine learning, decision systems, data bases, etc. [11][12][13][16][19][23]. Results of the relationship

analysis make it possible to discover the nature of the considered system or to decide its structure in order to satisfy given design constraints or optimize certain objectives.

The theory of information relationships and measures makes operational the famous theory of partitions and set systems of Hartmanis [2]. Partitions and set systems enable us to model information streams. The information relationships and measures enable us to analyze the relationships between the modeled information streams.

2. Representation of information in discrete information systems

Information is represented in discrete systems by values of some signals or variables. Let's consider a certain finite set of elements *S* called symbols. Information about symbols (elements of *S*) means the ability to distinguish certain symbols from some other symbols. Knowing a certain value of a certain attribute, signal or variable *x*, it is possible to distinguish a certain subset *B* of elements from *S* from all other elements of *S*, but it is impossible to distinguish between the elements from *B*. For example, if we consider an incompletely specified multiple-output Boolean function in Fig. 1, where symbols 1,…,5 represent terms (cubes) on input variables x_1,..., x_6, then different values of the input variable x_3 enable us to distinguish between the subset {1, 2, 3} and {4, 5}. For 1, 2 and 3: $x_3 = 0$ and for 4 and 5: $x_3 = 1$. Thus, knowing that $x_3 = 0$, we know that 1, 2 or 3 occurred, but we do not know which of the three symbols occurred. Knowing that $x_3 = 1$, we know that 4 or 5 occurred, but we do not know which of the two symbols occurred. In such a way information is modeled with partitions and set systems [1][2][5].

A **set system** *SS* on a set *S* is defined as a collection of nonempty and distinct subsets $B_1, B_2, \ldots, B_k$ of *S* called blocks, such that:

$\bigcup_i B_i = S$ and $B_i \not\subset B_j$ for $i \neq j$

A **generalized set system** (blanket [23]) *SS* on a set *S* is defined as a collection of nonempty and distinct subsets $B_1, B_2, \ldots, B_k$ of *S* such that: $\bigcup_i B_i = S$.

The set system algebra is presented in [1]. The **product and sum operations** are defined **for set systems** as follows:

$SS_1 \bullet SS_2 = Max\ (\ \{B_{SS1} \cap B_{SS2} \mid B_{SS1} \in SS_1 \wedge B_{SS2} \in SS_2\}\ )$, and $SS_1 + SS_2 = Max\ (\ SS_1 \cup SS_2\ )$, where: $Max(\ \{B_i \mid B_i \subseteq S\ \}\ ) = \{B \mid B \in \{B_i\} \wedge ((B' \in \{B_i\} \wedge B \subseteq B') \Rightarrow B=B')\ \}$ (Max removes blocks contained in other blocks).

SS_1 is **smaller than or equal** to SS_2: $SS_1 \leq SS_2$ if and only if each block of SS_1 is included in a block of SS_2.

A set system SS on S can be interpreted as a compatibility relation defined on S, with the compatibility classes being the blocks of SS. Such a compatibility relation is reflexive and symmetric but is not required to be transitive. If it is transitive, i.e. the subsets B_i of the corresponding set system SS are disjoined, then it is an equivalence relation and the set system SS is a **partition**. A certain set system SS (partition P) gives information about the elements of S with limited precision to the compatibility (equivalence) class. With this information it is possible to distinguish the different classes although it is impossible to distinguish between the elements of the same class. The set system product can be interpreted as a product of the corresponding relations; it represents combined information about the elements of S that is provided by the relations together. The set system sum can be interpreted as the sum of the corresponding relations and it represents information about elements of S after applying the combined abstraction provided by the relations involved. The partial ordering relation denotes the fact that if $SS_1 \leq SS_2$, then SS_1 provides the same or more information about elements of S than SS_2. In this paper, we will use the term *set system* to designate a set system, generalized set system or partition, everywhere where this will not lead to misunderstanding.

term symbols	inputs x_1	x_2	x_3	x_4	x_5	x_6	outputs y_1	y_2	y_3	y_4
1	0	-	0	0	0	1	0	1	1	0
2	0	0	0	1	0	-	1	1	-	0
3	1	-	0	0	1	-	1	1	1	0
4	1	1	1	1	1	-	0	-	0	1
5	-	0	1	-	0	0	-	0	-	0

Figure 1. Incompletely specified multi-output Boolean function

When analyzing or designing information systems, we are interested in relationships between information streams which express such properties as similarity (common information), dissimilarity (different information), missing information, extra information, etc. If we want to consider the interesting relationships and to base the design decisions on results of such analysis, introduction of an adequate analysis apparatus is necessary. This is however impossible before answering the question: "**what information is modeled by a certain set system?**" To answer this question, it was necessary to discover what is an elementary (atomic) portion of information, i.e. such portion that any other information (for example as modeled by a set system) can be expressed as a composition of such atomic portions.

I proposed the following solution [8]: 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 set of atomic portions of information can be represented by an **information relation I**, **information set IS** and **information graph IG** defined on $S \times S$ as follows:

$I = \{(s_i, s_j) \mid$ s_i is distinguished from s_j by the modeled information$\}$,

IS = {{s_i, s_j} | s_i is distinguished from s_j by the modeled information}, and

IG = {S, {{s_i, s_j} | s_i is distinguished from s_j by the modeled information}}.

In a strictly analogous way, the notion of the elementary (atomic) abstraction, and the appropriate abstraction (compatibility) relation A, set AS and graph AG can be introduced [8].

Relationships between set systems can now be analyzed by considering the relationships between their corresponding information and abstraction relations, sets and graphs. In particular, the correspondence between SS, IS and AS is as follows: IS contains the pairs of symbols that are not contained in any single block of a corresponding SS, and AS contains the pairs of symbols that are contained in at least one single block of a corresponding SS.

Example 1. (information modeling with set systems and information sets)

For the function from Fig. 1 the appropriate set systems and information sets for particular input and output variables and for the whole 4-output function are listed below:

$SS(x_1)$={{1,2,5};{3,4,5}} $IS(x_1)$={1|3, 1|4, 2|3, 2|4},
$SS(x_2)$={{1,2,3,5};{1,3,4}} $IS(x_2)$={1|4; 2|4; 4|5},
$SS(x_3)$={{1,2,3};{4,5}} $IS(x_3)$={1|4,1|5,2|4,2|5,3|4,3|5},
$SS(x_4)$={{1,3,5};{2,4,5}} $IS(x_4)$={1|2, 1|4, 2|3, 3|4},
$SS(x_5)$={{1,2,5};{3,4}} $IS(x_5)$={1|3,1|4,2|3,2|4,3|5,4|5},
$SS(x_6)$={{1,2,3,4};{2,3,4,5}} $IS(x_6)$={1|5}.

$SS(y_1)$={{1,4,5};{2,3,5}} $IS(y_1)$={1|2, 1|3, 2|4, 3|4},
$SS(y_2)$={{1,2,3,4};{4,5}} $IS(y_2)$={1|5; 2|5; 3|5},
$SS(y_3)$={{1,2,3,5};{2,4,5}} $IS(y_3)$={1|4,3|4},
$SS(y_4)$={{1,2,3,5};{4}} $IS(y_4)$={1|4, 2|4, 3|4, 4|5}.

$SS(y_1,y_2,y_3,y_4)$={{1};{2,3};{4};{5}}
$IS(y_1,y_2,y_3,y_4)$={1|2,1|3,1|4,1|5,2|4,2|5,3|4,4|5,3|5,4|5} ▮

To stress that the symbols from a pair of incompatible symbols are distinguished from each other and to shorten the notation of information sets, we will further denote each pair of incompatible symbols {s_i, s_j} by $s_i|s_j$.

In the same way information can be modeled with set systems and information sets for any discrete system, including multiple-valued systems. For instance in the case of a multiple-valued function, the only difference compared to the binary function is the fact, that the set systems induced by particular input and output variables are the multiple-block set systems instead of two-block set systems. Each of them has as many blocks as many values can take a certain input/output variable.

3. Information relationships and measures

When performing analysis or design of information systems, we often ask for relationships between information in various information streams, places or parts of a considered system. In general, we are interested in information relationships that express similarity, dissimilarity, missing information, extra information etc. Below, some basic information relationships are recalled which express these elementary properties:

- **common information** CI (i.e. information that is present in both SS_1 and SS_2): $CI(SS_1, SS_2) = IS(SS_1) \cap IS(SS_2)$
- **total** (combined) **information** TI (i.e. information that is present either in SS_1 or in SS_2): $TI(SS_1, SS_2) = IS(SS_1) \cup IS(SS_2)$
- **missing information** MI (i.e. information that is present in SS_1, but missing in SS_2): $MI(SS_1, SS_2) = IS(SS_1) - IS(SS_2)$
- **extra information** EI (i.e. information that is missing in SS_1, but present in SS_2): $EI(SS_1, SS_2) = IS(SS_2) - IS(SS_1)$
- **different information** DI (i.e. information that is present in one of the set systems and missing in the other): $DI(SS_1, SS_2) = MI(SS_1, SS_2) \cup EI(SS_1, SS_2)$.

Analogous relationships can be defined for abstraction [8].

Example 2. (some information relationships for the function from Fig. 1)

$CI(SS(y_1), SS(x_4))$ = {1|2, 3|4}
$TI(SS(y_1), SS(x_4))$ = {1|2, 1|3, 1|4, 2|3, 2|4, 3|4}
$MI(SS(y_1), SS(x_4))$ = {1|3, 2|4}
$EI(SS(y_1), SS(x_4))$ = {1|4, 2|3}
$DI(SS(y_1), SS(x_4))$ = {1|3, 1|4, 2|3, 2|4} ▮

With the relationship apparatus defined above, we can answer such questions as: what information required to compute values of a certain variable is present in another variable, what information is missing, what extra information is present etc.

However, knowledge of "what" is often not sufficient to take appropriate design decisions, because the design decisions are based in most cases not only on some qualitative, but also on some quantitative comparisons. Along the knowledge of "what", some knowledge of "how much" and "how important" is required. For instance: how much of the required information provides a certain variable and how much another one, is it much more, etc. Moreover, information that is provided by just a single input may be considered as more important than information provided by all inputs, an input that delivers a unique information may be considered as more important than another input that delivers only information provided also by some other inputs etc.

Therefore, some quantitative relationship measures are often necessary.

For a single set system *SS* the **information quantity IQ** is defined: IQ(SS) = |IS(SS)|.

For two set systems SS1 and SS2, the following **relationship measures** are defined:

- **information similarity (affinity) measure ISIM**: $ISIM(SS_1, SS_2) = |CI(SS_1, SS_2)|$
- **information dissimilarity (difference) measure IDIS**: $IDIS(SS_1, SS_2) = |DI(SS_1, SS_2)|$
- **information decrease (loss) measure IDEC**: $IDEC(SS_1, SS_2) = |MI(SS_1, SS_2)|$
- **information increase (growth) measure IINC**: $IINC(SS_1, SS_2) = |EI(SS_1, SS_2)|$
- **total information quantity TIQ**: $TIQ(SS_1, SS_2) = |TI(SS_1, SS_2)|$.

It is also possible to define some relative measures, by normalizing the above absolute measures, and weighted measure, for example, by associating an appropriate importance weight $w(s_i|s_j)$ with each elementary information [8]. In a strictly analogous way, the abstraction relationship measures can be defined [8].

4. Application of the relationships and measures to logic synthesis

The information and abstraction relationships as well as the measures for the strength and importance of the relationships enable designers and tools to analyze relationships between the information streams of the considered systems and provide them with data necessary for effective and efficient decision-making. Results of the relationship analysis make it possible to discover the nature of the considered system or to decide its structure in order to satisfy given design constraints or optimize certain objectives.

In order to demonstrate the importance of the information relationship analysis and to show how to apply it in the logic synthesis process, let's consider the problems of the input support minimization, parallel (output) decomposition and serial functional decomposition. These problems are highly relevant and common for both the binary logic applications (e.g. VLSI circuit design [3]-[10][12][14][15][17][18][20]-[22]) and multiple-valued logic applications (e.g. pattern analysis, knowledge discovery, machine learning, decision systems, data bases, VLSI circuit design etc. [11][12][13][16][19][23]). They are considered below for the case of incompletely specified Boolean functions. However application of the information relationships and measures to the multiple-valued logic is strictly analogous, because in the multiple-valued case the information is modeled with information sets the same way as in the binary case, and the relationships and measures operate on the information sets.

4.1. Application to input support minimization.

Input support minimization consists of finding a minimal sub-set of inputs that contains all information that was required to compute by the originally specified function, or sequential machine [7][9].

Let's find the minimum support for the function from Fig. 1, by analyzing the relationships between the information required for computing the output values (modeled by set system $SS(y_1,y_2,y_3,y_4)$= {{1};{2,3};{4};{5}}) and information provided by particular input variables. Some of the interesting relationships and the corresponding measures are given below:

$CI(y, x_1) = \{1|3, 1|4, 2|4\}$ $|CI(y, x_1)| = 3$
$CI(y, x_2) = \{1|4, 2|4, 4|5\}$ $|CI(y, x_2)| = 3$
$CI(y, x_3) = \{1|4, 1|5, 2|4, 2|5, 3|4, 3|5\}$ $|CI(y, x_3)| = 6$
$CI(y, x_4) = \{1|2, 1|4, 3|4\}$ $|CI(y, x_4)| = 3$
$CI(y, x_5) = \{1|3, 1|4, 2|4, 3|5, 4|5\}$ $|CI(y, x_5)| = 5$
$CI(y, x_6) = \{1|5\}$ $|CI(y, x_6)| = 1$

Observe that:
$|CI(y, x_3)| > |CI(y, x_5)| > |CI(y, x_1)| = |CI(y, x_2)| = |CI(y, x_4)| > |CI(y, x_6)|$.

The information similarity measure tells us, that x_3 delivers more information that is required for computing y than x_5, x_5 more than x_1, etc. Furthermore, x_3 delivers a unique information that is necessary for computing the output, but is not delivered by any other input variable, namely: 2|5. Therefore, x_3 must be in any support, thus also in any minimal support. Since $CI(y, x_3)$ does not cover IS(y) completely, the symbols from the pairs given by $IS(y) - CI(y, x_3) = \{1|2, 1|3, 4|5\}$ are not distinguished from each other by x_3, and therefore at least one extra input variable is necessary for the minimum support. Because $CI((IS(y) - CI(y, x_3)), x_1) = \emptyset$, $CI((IS(y) - CI(y, x_3)), x_2) = \{4|5\}$, $CI((IS(y) - CI(y, x_3)), x_4) = \{1|2\}$, $CI((IS(y) - CI(y, x_3)), x_5) = \{1|3, 4|5\}$, $CI(y, x_6) = \emptyset$, no single variable delivers the lacking information, but x_4 and x_5 together provide it. This means that x_3, x_4 and x_5 constitute the minimum input support.

In a similar way the information relationships and weighted measures are used in our genetic algorithm (GA) for finding the minimal input support [7]. In particular, the weighted information similarity measure is used in GA for the semi-random construction of the initial support population. The probability of a certain input variable to be selected for an initial support is proportional to the value of this measure. The same measure is also used for the selection of input variables in the local search performed by the deterministic *merge* and *repair* operators of the genetic algorithm [7]. Table 1 presents the test results of GA compared to the strictly minimal results and results from *QuickScan* algorithm [7][9] computed at the HP9000/735 processor under HP-UX 9.05. The results from GA are computed quickly and are in almost all cases strictly optimal and sometimes on distance one from optimum. The test files were specially generated, because the standard logic synthesis benchmark set contained functions with minimal input supports. The benchmarks do not contain either essential or dominated inputs and are "difficult", because they cannot be solved or reduced by the preprocessing

Table 1. Test results of the genetic algorithm (GA) and *QuickScan* (QS).

File	Q (min)	Q QS	T QS	Q GA-[1]	T GA-[1]	Q mg	T mg	Q GA[2]	T GA[2]
i10	7	7	<1	7	<1	7	<1	7	<1
i20s04a	4	4	<1	4	1	4	<1	4	<1
i20s05a	5	5	<1	5	1	5	<1	5	2
i20s06a	6	6	<1	6.17	2	6	<1	6	2
kaz	5	6	<1	5.17	2	5	<1	5	1
i30s08a	8	9	2	9.17	30	8	6	8	40
i30s08b	8	9	2	9.17	22	9	6	8	38
i30s08c	8	9	2	9.17	21	8	6	8.13	34
i30s10a	10	11	6	11.17	1:37	11	58	10.88	2:39
i40s09a	9	10	5	9.83	56	9	19	9	1:52
i40s09b	9	10	5	9.83	55	9	19	9	1:53
i40s10a	10	11	9	11.17	2:53	11	1:47	10	6:47
i40	10	11	9	11	3:09	11	1:47	10	6:32
i50	7	9	2	8.67	20	8	4	8	48
i100	9	11	30	10.83	3:31	9	1:02	9	9:58

Q-quality (number of variables in the support) of the best support found, Q(min) – quality of the minimum support, T – computation time (min:sec), QS – QuickScan, GA – genetic algorithm, GA- - GA without merge, mg – merge performed on the full original support, [1] Average over 6 runs, [2] Average over 8 runs.

techniques (i.e. recursively removing the rows covered by essential columns, and the dominated columns and rows). The number by "i" in the benchmark's name is equal to the number of input variables of the benchmark. More complete characteristics of the benchmarks and of the way they were generated as well as benchmark results of another input support minimization programs can be found in [7] and [9].

4.2. Application to parallel (output) decomposition.

Parallel decomposition is a very important decomposition technique used for multiple output functions, relations and sequential machines [4][6][12][17]. In parallel decomposition, computations of values for certain subsets of outputs are performed in separate blocks. Thus, parallel decomposition consists of partitioning of the set of output variables in disjoint subsets, each satisfying the building block constraints, so that some objectives are optimized and some other possible constraints satisfied.

Assume that each logic building block for implementing the considered example function can implement any function, with maximally 2 inputs and 2 outputs (as in look-up table FPGAs). Since the example function has 4 outputs and requires minimum 3 inputs, it is impossible to implement the function with a single building block. The function must be decomposed into at least two blocks. Find a parallel decomposition with minimal number of blocks and minimum interconnections.

Let's compute the affinity relationships between the information required by particular output variables and information delivered by particular input variables and the corresponding relative similarity measures. For simplicity sake, let's consider only the minimal input support $\{x_3, x_4, x_5\}$ (the function with the original support can be considered precisely the same way).

$CI(y_1, x_3) = \{2|4, 3|4\}$ $ISIM_r(y_1, x_3) = 0,5$
$CI(y_1, x_4) = \{1|2, 3|4\}$ $ISIM_r(y_1, x_4) = 0,5$
$CI(y_1, x_5) = \{1|3, 2|4\}$ $ISIM_r(y_1, x_5) = 0,5$
$CI(y_2, x_3) = \{1|5, 2|5, 3|5\} = IS(y_2)$ $ISIM_r(y_2, x_3) = 1$
$CI(y_2, x_4) = \varnothing$ $ISIM_r(y_2, x_4) = 0$
$CI(y_2, x_5) = \varnothing$ $ISIM_r(y_2, x_5) = 0$
$CI(y_3, x_3) = \{1|4, 3|4\} = IS(y_3)$ $ISIM_r(y_3, x_3) = 1$
$CI(y_3, x_4) = \{1|4, 3|4\} = IS(y_3)$ $ISIM_r(y_3, x_4) = 1$
$CI(y_3, x_5) = \{1|4\}$ $ISIM_r(y_3, x_5) = 0,5$
$CI(y_4, x_3) = \{1|4, 2|4, 3|4\}$ $ISIM_r(y_4, x_3) = 0,75$
$CI(y_4, x_4) = \{1|4, 3|4\}$ $ISIM_r(y_4, x_4) = 0,5$
$CI(y_4, x_5) = \{1|4, 2|4, 4|5\}$ $ISIM_r(y_4, x_5) = 0,75$

Since $ISIM_r(y_2, x_3) = 1$ and $ISIM_r(y_3, x_3) = 1$, x_3 delivers complete information required for computing y_2 and y_3. Thus, y_2 and y_3 can be implemented with just a single logic block with one input x_3 and two outputs y_2 and y_3. y_1 and y_4 require each at least two input variables for their computation, because no single variable delivers all required information ($ISIM_r(y_1, x_i)<1$ and $ISIM_r(y_4, x_i)<1$, for all i=3,4,5). Furthermore, x_5 provides a unique information for y_1 (1|3), and for y_4 (4|5), not provided by any other input. Importance of x_5 is thus highest, and therefore, x_5 must be used both for computing y_1 and y_4. Similarly, x_4 provides a unique information for y_1 (1|2), importance of x_4 is also highest and x_4 must be used for computing y_1. Since $CI(y_1, \{x_4, x_5\}) = \{1|2, 1|3, 2|4, 3|4\} = IS(y_1)$, $\{x_4, x_5\}$ is a complete minimum support for y_1. $\{x_4, x_5\}$ is also a complete minimal support for y_4, because:

$CI(y_4, \{x_4, x_5\}) = \{1|4, 2|4, 3|4, 4|5\} = IS(y_4)$.

Thus, y_1 and y_4 can be implemented with just a single building block with two inputs x_4 and x_5, and two outputs y_1 and y_4. In this way, we constructed a parallel decomposition of the considered function (Fig. 2) that satisfies the input and output constraints of the used building blocks, uses minimum number of building blocks (just two blocks) and minimum interconnections (no common inputs to different blocks).

More information on parallel decomposition, prototype tools we developed for this aim and some benchmark results of these tools can be found in [4] and [6].

4.3. Application to serial functional decomposition.

Fig. 3 shows the schematic representation of the serial functional decomposition.

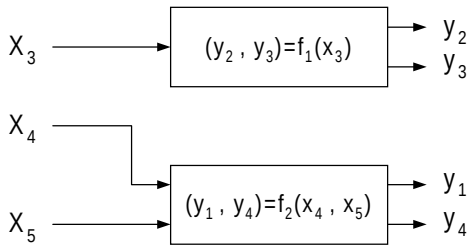

Figure 2. Optimal parallel decomposition of the example function from Fig. 1.

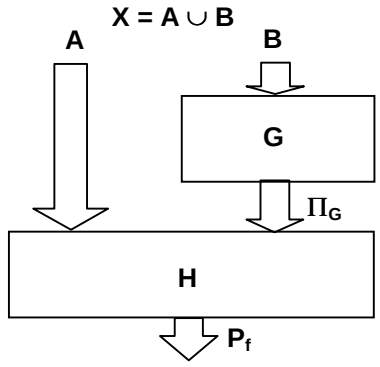

Figure 3. Serial decomposition scheme

Let *X* be the set of input variables of a given function F. Let *A* and *B* be two subsets of *X* such that $A \cup B = X$. Let P(*A*) be a set system induced by the input variables from the set *A* on the set of the function terms (cubes) F, let P(*B*) be a set system induced on F by the variables from *B* and P_F be a set system induced on F by the output variables of F.

Theorem 1 If there exists a set system Π_G on F such that $P(B) \leq \Pi_G$, and $P(A) \bullet \Pi_G \leq P_F$, then F has a serial decomposition with respect to (*A*,*B*) [17]. ∎

The conditions for serial decomposition, can be re-expressed in terms of the information sets in the following way:

Theorem 2. If there exists a set system Π_G on F such that $IS(P(B)) \subseteq IS(\Pi_G)$, and $IS(P(A) \bullet \Pi_G) \subseteq IS(P_F)$, then F has a serial decomposition with respect to (A, B) [20]. ∎

In general, information that is necessary for computing values of a certain output of a considered multiple-output function is distributed on a number of its inputs. The inputs also deliver some information, that is not needed for the output, and information on the inputs is represented otherwise than on the considered output. Block G can be therefore considered as a block where an intermediate information transformation is performed which involves an appropriate combination, abstraction and re-structuring of the input information. This transformation consists of construction of an appropriate set system Π_G from a selected set system P(B). The set system Π_G should carry a part of the information delivered by the set system P(*B*), which in combination with information delivered by the set system P(*A*), is essential to compute the required output information (Theorem 2). Since the number of block's G outputs should be in practical cases much smaller than the number of its inputs, Π_G must have much less blocks than P(*B*). Π_G is created by merging the blocks of P(*B*) and a lot of abstraction is made during this merging. To avoid too big loss of information during the merging process or creation of set systems Π_G with too large number of blocks (which would require too many physical logic blocks in implementation), set *B* should contain input variables that carry relatively much information which is not important for computing values of the output variables. Moreover, a part of the important information delivered by variables from set *B* is in most cases also delivered by some transfer this information to the output of G. It can be abstracted during the merging process. Also, the set systems induced by each of the input variables from set *B* should be similar each to another, because this results in less blocks of P(*B*) itself and less merging to perform.

Based on these observations the following rules of input variable selection for set *B* have been developed. Variable x_i should be included into set B if:

IINC(P_F, P(x_i)) is relatively high,

ISIM(P(A), P(x_i)) is relatively high,

ISIM(P(B'), P(x_i)) is relatively high,

where: x_i - candidate to set of indirect variables B, B' – partially created set B.

Using these rules, we developed and implemented a heuristic method for the (near-) optimal input support selection in serial decomposition. Experimental results from our method are presented below and compared to the results from the systematic search. In the systematic search the supports are implicitly enumerated. For each support, the corresponding set system Π_G is computed and the first found support with the minimum number of blocks in Π_G is used for decomposition. The number of blocks of partition Π_G shows strong positive correlation with the number of logic blocks and logic levels in the resulting decomposition, and therefore, it is a good measure for the support quality. In the case of our heuristic selection also the first support is used for decomposition, but the first one found based on information relationships measures. We used for experiments a number of functions from the international logic synthesis benchmark set. Table 2 and 3 report results of the input support selection for set B in a single serial decomposition step, as illustrated in Fig. 2. Table 4 shows the results of decomposition of some benchmark functions into a network of 4-input, 1-output logic cells, obtained by repeating the serial decomposition step from Fig. 3 a number of times. In Table 2 the comparison of the number of blocks of partition Π_G is presented for heuristic and systematic method.

Table 2. Comparison of the number of blocks in partition Π_G obtained by the systematic and heuristic method.

Bench-mark	Size			Systematic method (|B|)				Heuristic method (|B|)			
	in-puts	out-puts	cubes	3	4	5	6	3	4	5	6
z4	7	4	128	4	6	8	12	4	6	8	12
5xp1	7	10	126	8	16	32	64	8	16	32	64
misex1	8	7	18	4	6	7	9	4	6	7	9
root	8	5	71	5	9	15	17	5	9	15	17
opus	9	10	23	4	6	8	10	5	6	8	10
9sym	9	1	191	4	5	6	7	4	5	6	7
alu2	10	3	391	6	12	24	43	6	12	24	43
sao2	10	4	60	4	6	9	11	4	6	9	11
sse	11	11	39	4	6	8	11	4	6	9	11
keyb	12	7	147	6	9	13	19	6	10	14	19
s1	13	11	110	5	8	13	19	6	10	15	19
plan	13	25	115	5	7	11	17	5	9	14	19
styr	14	15	140	4	6	9	13	5	7	10	13
ex1	14	24	127	4	6	8	11	4	6	8	11
kirk	16	10	304	4	4	5	6	4	5	5	6

The systematic search always finds a support, which results in partition Π_G with the strictly minimum number of blocks. However, the method based on information measures produces also the minimal or near-minimal results and does it much faster. Table 3 shows comparison of the computation time for both methods for a single support selection step. In Table 4, the number of logic cells (4-input, 1-output FPGA cells) in decompositions is presented for several benchmarks, for decompositions found by application of the heuristic or systematic support selection a number of times. The method based on information relationship measures produces decompositions with the same or almost the same number of blocks as the systematic method. In some cases the results from the heuristic support selection are even better than from the systematic. More information on the application of the information relationships and measures to the serial functional decomposition is presented in [18].

5. Conclusion.

The famous theory of partitions and set systems of Hartmanis [2] and the theory of information relationships and measures discussed here form just together an information modeling and analysis apparatus that is of primary importance for analysis and synthesis of discrete information systems, and in particular for logic design.

We applied the apparatus of information relationships and measures for design decision making in a number of prototype CAD tools for solving various logic design problems. Among others, we applied this apparatus for solving some logic decomposition problems and finding the minimal input support. The experimental results obtained with our prototype tools are very encouraging. For example, our input support minimization tool is able

Table 3. Comparison of the computation time (in seconds) for the systematic and heuristic method.

Bench-mark	Size			Systematic method (|B|)				Heuristic method (|B|)			
	in puts	out puts	cubes	3	4	5	6	3	4	5	6
z4	7	4	128	1	1	2	2	1	2	5	8
5xp1	7	10	126	0	2	1	2	2	2	5	7
misex1	8	7	18	0	1	1	3	1	1	2	5
root	8	5	71	1	2	2	3	1	2	5	12
opus	9	10	23	1	2	3	8	0	1	3	7
9sym	9	1	191	10	20	41	79	5	8	16	34
alu2	10	3	391	33	51	64	63	32	40	60	98
sao2	10	4	60	3	6	14	36	1	2	6	21
sse	11	11	39	2	7	20	63	1	2	8	25
keyb	12	7	147	17	52	156	503	7	11	24	51
s1	13	11	110	13	46	137	552	7	10	17	33
plan	13	25	115	16	51	184	595	6	9	19	42
styr	14	15	140	31	109	404	1453	12	17	32	58
ex1	14	24	127	24	91	333	1377	8	12	26	58
kirk	16	10	304	108	528	2125	11234	55	69	119	230

Table 4. Comparison of the number of logic cells in decomposition.

	Size			Systematic	Heuristic
Benchmark	inputs	outputs	cubes	method	Method
z4	7	4	128	7	7
5xp1	7	10	126	20	21
misex1	8	7	18	19	18
root	8	5	71	46	47
alu2	10	3	391	116	114

to compute the minimal supports for difficult and large (100 inputs) problem instances within few minutes. Our prototype multi-level decomposition tool for FPGAs is able to perform a single step of the functional decomposition more than 50 times faster than the tool based on systematic search, on benchmarks with 14-16 inputs, 10-20 outputs and 100-300 cubes, when producing the strictly optimal or near optimal results. By repetitively applying the decomposition step, our tool is able to completely decompose such a benchmark into 4-input CLBs thousands times faster then the systematic algorithm, when producing the minimum number of CLBs or at most 1 CLB more than the minimum on some of the checked benchmarks. The difference in processing time between these two tools grows very fast with the size of the decomposed function.

The application examples and experimental results from our prototype tools demonstrate the high potential of the information relationships and measures in solving various logic analysis and synthesis problems

6. Acknowledgements

The author is indebted to Prof. T. Luba for his cooperation in the part of the research concerning

the serial functional decomposition and to M. Rawski and N. Ederveen for their help in implementation of programs and performing experiments.

7. References

[1] J.A. Brzozowski and T. Luba: Decomposition of Boolean Functions Specified by Cubes, University of Waterloo Research Report, CS-97-01, Waterloo, Canada, January 1997.

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

[3] S.C. Chang, M. Marek-Sadowska, T.T. Hwang: Technology Mapping for TLU FPGAs Based on Decomposition of Binary Decision Diagrams, Ieee Tr. On CAD, vol. 15, No 10, Oct. 1996, pp.1226- 1236.

[4] L. Jóźwiak, F. Volf: An Efficient Method for Decomposition of Multiple Output Boolean Functions and Assigned Sequential Machines, EDAC - The European Conference on Design Automation, Brussels, Belgium, March 16-19, 1992, pp. 114-122.

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

[6] L. Jóźwiak, F.A.M. Volf: Efficient Decomposition of Assigned Sequential Machines and Boolean Functions for PLD Implementations, IEEE International Conference on Electronic Technology Directions, Adelaide, Australia, 23-25 May, 1995, pp. 258-266.

[7] L. Jóźwiak, N.Ederveen: Genetic Algorithm for Input Support Minimization, International Conference on Computational Intelligence and Multimedia Applications (ICCIMA?98), Melbourne, Australia, Febr. 9-11, 1998.

[8] L. Jóźwiak: Information Relationships and Measures - An Analysis Apparatus for Efficient Information System Synthesis, Proceedings of the 23rd EUROMICRO Conference (EUROMICRO'97), Budapest, Hungary, September 1-4, 1997, pp. 13-23., IEEE Computer Society Press.

[9] P.A. Konieczny and L. Jóźwiak: Minimal Input Support Problem and Algorithms to Solve It, Eindhoven University of Technology Research Reports, EUT Report 95-E-289, Eindhoven University of Technology, The Netherlands, Apr. 1995.

[10] Y.T. Lai, K.R.R. Pan, M. Pedram: OBDD –Based Function Decomposition: Algorithms and Implementation, IEEE Tr. on CAD, vol.15. No 8, Aug.1996, pp. 977-990.

[11] T.Y.Lin and N. Cercone(Eds.): Rough Sets and Data Mining - Analysis of Impresise Data, Kluwer, 1997.

[12] T. Luba: Decomposition of Multiple-Valued Functions, Proc. IEEE ISMVL'95, Bloomington, Indiana, USA, May 23-25, 1995.

[13] T. Luba and J. Rybnik: Rough Sets and Some Aspects of Logic Synthesis, in R. Slowinski (Ed.): Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory, pp. 181-199, Kluwer, 1993.

[14] R. Murgai, N. Shenoy, R.K. Brayton, A. Sangiovanni Vincentelli: Performance Directed Synthesis for Table Look Up Programmable Gate Arrays, IEEE ICCAD, pp. 572-575, 1991.

[15] R. Murgai, R.K. Brayton, A. Sangiovanni Vincentelli: Optimal Functional Decomposition Using Encoding, 31st ACM/IEEE DAC, pp. 408-414, June 1994.

[16] M. Perkowski, T. Ross, D. Gadd, J.A. Goldman, N. Song: Application of ESOP Minimization in Machine Learning and Knowledge Discover, Proc. Reed Muller'95 Workshop, Chiba, Japan, 1995.

[17] M. Rawski, L. Jozwiak, M. Nowicka, T. Luba: Non-Disjoint Decomposition of Boolean Functions and Its Application in FPGA-oriented Technology Mapping, Proc. of the EUROMICRO'97 Conference, Budapest, Hungary, Sept. 1-4, 1997, pp.24-30, IEEE Computer Society Press.

[18] M.Rawski, L. Jóźwiak, T. Luba, A. Chojnacki: Efficient Logic Synthesis for FPGAs with Functional Decomposition Based on Information Relationship Measures, EUROMICRO'98, Västerås, Sweden, August 25-27, 1998, pp 8-15.

[19] T. Ross, M. Noviskey, T. Taylor, D. Gadd: Pattern Theory: An Engineering Paradigm for Algorithm Design, Final Technical Report, Wright Laboratories, WL/AART/WPAFB, 1991.

[20] F.A.M. Volf, L. Jóźwiak, M.P.J. Stevens: Division-Based versus General Decomposition-Based Multiple-Level Logic Synthesis, VLSI Design, vol.3, No 3, pp. 267 - 287, 1995.

[21] F.A.M. Volf, L. Józwiak: Decompositional Logic Synthesis Approach for Look Up Table Based FPGAs, 8th IEEE International ASIC Conference, Austin, Texas, 18-22 September, 1995.

[22] W. Wan, M.A. Perkowski: A New Approach to the Decomposition of Incompletely specified Multi-Output Functions Based on Graph Coloring and Local Transformation and Its Application to FPGA Mapping, European Design Automation Conference, Hamburg, Germany, September 7-10, 1992, pp. 230-237.

[23] B, Zupan and M. Bohanec: Learning Concept Hierarchies from Examples by Functional Decomposition, Research Report, Department of Inteligent Systems, Josef Stefan Institute, Ljubljana, Slovenia, Sept. 1966.