jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2012, VOL. 58, NO. 1, PP. 15–20
Manuscript received December 31, 2011; revised March 2012. DOI: 10.2478/v10177-012-0002-x

Input Variable Partitioning Method for Decomposition-Based Logic Synthesis targeted Heterogeneous FPGAs

Mariusz Rawski

***Abstract*—The functional decomposition has found an application in many fields of modern engineering and science, such as combinational and sequential logic synthesis for VLSI systems, pattern analysis, knowledge discovery, machine learning, decision systems, data bases, data mining etc. It is perceived as one of the best logic synthesis methods for FPGAs. However, its practical usefulness for very complex systems depends on efficiency of method used in decomposition calculation.**

One of the most important steps in functional decomposition construction is selection of the appropriate input variable partitioning. In case of modern heterogeneous programmable structures efficiency of methods used to solve this problem becomes especially important. Since the input variable partitioning problem is an NP-hard, heuristic methods have to be used to efficiently and effectively search for optimal or near-optimal solutions.

The paper presents a method for bound set selection in functional decomposition targeted FPGAs with heterogeneous structure. This heuristic algorithm delivers optimal or near optimal results and is much faster than other methods.

***Keywords*—Functional decomposition, heterogeneous FPGA, bound set selection.**

I. Introduction

With rapid growth of traditional FPGA industry, heterogeneous logic blocks are often used in the actual FPGA architectures such as Xillinx Virtex-5 and Altera Stratix III series. An FPGA structure can be described as an array of programmable logic elements (cells) interconnected by programmable connections. Early FPGAs used a logic cell consisting of a 4-input lookup table (LUT) and register. Although the basic architecture of FPGAs has not changed dramatically since their introduction in the 1980s, present devices employ larger numbers of inputs (6-input for Virtex-5 and 7-input for Stratix III) and have other associated circuitry. Another enhancement extensively used in modern FPGAs are specialized embedded blocks, serving to improve delay, power and area if utilized by the application, but waste area and power if unused.

How to handle this kind of heterogeneous design network to generate LUTs with different input sizes in the mapping is a very important and practical problem. The existing CAD tools are not well suited to utilize all possibilities that modern heterogeneous programmable structures offer due to the lack of appropriate synthesis methods. Typically, after the logic synthesis stage, technology-dependent mapping methods are used to map design into available resources [1], [2]. However, such an approach is inefficient due to the fact that the quality of postsynthesis mapping is highly dependent on the quality of technology independent optimization step [3]. Recently, efforts have been made to develop methods based on functional decomposition that would allow for efficient utilization of heterogeneous structure of FPGA [4]. This method is designed specifically to implement FIR filters using the concept of distributed arithmetic. In [5] there were proposed advanced synthesis methods based on functional decomposition that utilizes embedded memory block as large LUTs.

Functional decomposition is a logic synthesis method that has recently gained much recognition [6]. The main reason is the evolution of field programmable gate-arrays (FPGAs) as a new technology for digital system implementation. Architecture of FPGA is based on the lookup table as basic building block, where an n-input LUT is capable of implementing any Boolean function of up to n variables. Thus, logic synthesis for LUT-based FPGAs must transform a logic network into network that consists of nodes with up to n inputs only. Each node of such network can be then implemented by a single LUT. For this reason, for the case of implementation targeting FPGA structure, functional decomposition is perceived as one of the best logic synthesis methods. However modern FPGA devices have very complex structure. Today's FPGAs are entire programmable systems on a chip (SoC) which are able to cover an extremely wide range of applications. Their heterogeneous structure enforces development of new functional decomposition based synthesis methods that could take advantage of such architecture of nowadays FPGAs.

In the functional decomposition process the following three factors play an extremely important role: an appropriate input variable partitioning, decision which (multi-valued) function will be computed by a certain subsystem and encoding of the subsystem's function with binary output variables.

In the paper a heuristic method of input variable partitioning is described that is suitable for decomposition-based logic synthesis for FPGAs with heterogeneous structure. Application of this method for construction of the input variable partition allows reducing the search space to a manageable size while keeping the high-quality solutions in the reduced space.

II. Preliminary Information

A. Architectures of Modern FPGAs

The technological advancements in Field Programmable Gate Arrays in the past decade have opened new paths

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

for digital systems design engineers. The FPGA maintains the advantages of custom functionality like an ASIC, while avoiding the high development costs and the inability to make design modifications after production. An FPGA structure can be described as an array of LUT-based programmable logic elements (cells) interconnected by programmable connections. Each cell can implement a simple logic function (of a limited number of inputs) defined by a designer's CAD tool. A typical programmable device has a large number (64 to over 1 000 000) of such cells, that can be used to form complex digital circuits. The ability to manipulate the logic at the gate level means that the designer can construct a custom processor to efficiently implement the desired function. The technological progress in microelectronics in the past decade has changed this picture by introducing embedded specialized blocks into structure of FPGA chip.

Modern FPGA devices have very complex structure. Today's FPGAs are entire programmable systems on a chip (SoC) which are able to cover an extremely wide range of applications. The Altera Stratix III and Xilinx Virtex-5 families of devices, both using a 65 nm manufacture process, can be used as examples of contemporary FPGAs. The basic architecture of FPGAs has not changed dramatically since their introduction in the 1980s. Early FPGAs used a logic cell consisting of a 4-input lookup table (LUT) and register. Present devices employ larger numbers of inputs (6-input for Virtex-5 and 7-input for Stratix III) and have other associated circuitry. Another enhancement extensively used in modern FPGAs are specialized embedded blocks, serving to improve delay, power and area if utilized by the application, but waste area and power if unused. Early embedded blocks included fast carry chains, memories, phase locked loops, delay locked loops, boundary scan testing and multipliers. More recently, multipliers have been replaced by digital signal processing (DSP) blocks (which add support for logical operations, shifting, addition, multiply-add, complex multiplication etc.), allowing designers to use methodology known from DSP programming. Some architectures even contain hardware CPU cores.

The basic building block of logic in the Stratix III architecture is the adaptive logic module (ALM). Each ALM contains a LUT-based resources that can be divided between two combinational adaptive LUTs (ALUTs) and two registers. Combinational ALUTs may have up to eight inputs. An ALM can implement various combinations of two functions, any function of up to six inputs and certain seven-input functions. In addition to the adaptive LUT-based resources, each ALM contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. This dedicated resources allow efficiently implementing various arithmetic functions and shift registers. TriMatrix embedded memory blocks provide three different sizes of embedded SRAM: 640 bit (in ROM mode only) or 320 bit memory logic array blocks (MLABs), 9 Kbit M9K blocks, and 144 Kbit M144K blocks.

The elementary programmable logic blocks in Xilinx Virtex-5 FPGAs are called slices and are organized in Configurable Logic Blocks (CLBs). The CLBs are the main logic resources for implementing sequential as well as combinatorial circuits. Each CLB element is connected to a switch matrix for access to the general routing matrix. A CLB element contains a pair of slices. Each slice has four 6-input function generators (LUTs), embedded multiplexers, carry logic, and four registers. The function generators can implement any arbitrarily defined six-input Boolean function. Each function generator can also implement two arbitrarily defined five-input Boolean functions, as long as these two functions share common inputs.

In addition to the basic LUTs, slices contain three multiplexers. These multiplexers are used to combine up to four function generators to provide any function of seven or eight inputs in a slice. Virtex-5 CLBs also support distributed memory – each look-up table can be configured to operate as a 64-bit memory. The block RAM in Virtex-5 FPGAs stores up to 36K bits of data and can be configured as either two independent 18 Kb RAMs, or one 36 Kb RAM.

As was already mentioned specialized embedded memory blocks (EMB) make it possible to implement data storage modules, such as shift registers or RAM blocks. In many cases, though, the designer does not need such elements in design or not all such resources are utilized. This chip area need not be wasted, however, if the unused memory arrays are used to implement logic. Configuring the arrays as ROM results in large lookup-tables that might very efficiently implement some logic circuits. The memories act as very large logic cell, where the number of inputs is equal to the number of address lines and the number of output is equal to the size of memory word. Since the size of address and memory word of single EMB can be configured in several different ways it can act as logic cell of different sizes.

Such architecture of modern programmable FPGAs greatly extends the space of possible solution during the process of mapping the design into FPGA structure. Unfortunately this heterogeneous structure of available logic resources greatly increases the complexity of mapping algorithms. The existing CAD tools are not well suited to utilize all possibilities that such modern programmable structures offer due to the lack of appropriate logic synthesis methods.

B. Serial Functional Decomposition

Functional decomposition relies on breaking down a complex system into a network of smaller and relatively independent co-operating sub-systems, in such a way that the original system's behavior is preserved [7], [8].

The set X of function's input variables 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: $U = x_1, \ldots, x_r$ 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

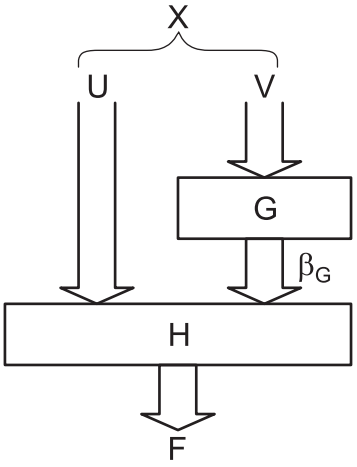

Fig. 1. Schematic representation of the serial decomposition.

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 [8] a theorem based on concept of blankets 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 $\beta_U \cdot \beta_G \leq \beta_F$, then F has a serial decomposition with respect to (U, V).

The process of functional decomposition consists of the following steps:

- the selection of an appropriate input support V for block G (input variable partitioning),
- the calculation of the blankets β_U, β_V and β_F,
- the construction of an appropriate multi-block blanket β_G (this corresponds to the construction of the multi-valued function of block G),
- the creation of the binary functions H and G by representing the multi-block blanket β_G as the product of a number of certain two-block blankets (this is equivalent to encoding the multi-valued function of block G defined by blanket β_G with a number of binary output variables).

In a multilevel decomposition, this process is applied to functions H and G repetitively, until each block of the obtained in this way net can be directly mapped in a logic block of a specific implementation structure [9].

The practical usefulness of functional decomposition for very complex systems is limited by lack of an efficient method for the construction of the high quality sub-systems. In the sub-system construction process one of the most important steps is to select an appropriate input support (bound set) for sub-systems, thus efficiency of input variable partitioning method has the great impact on quality the whole logic synthesis.

III. Input Variable Partitioning

The selection of an appropriate input variable partitioning is the main problem in functional decomposition. The choice of sets U and V from set X determines the construction of an appropriate blanket β_G which satisfies Theorem 1.

The input variable partitioning is NP-hard problem. For function F of n input variables and the size k of set V the number of possible solutions is described by binomial coefficient:

$$l = \binom{n}{k} = \frac{n!}{(n-k)!k!}.$$

Early methods ware based on enumeration of all partitions of fixed size [10]–[13]. However, when n and k is large evaluating all partition is too time consuming. There are two types of algorithms solving this problem. One of them are algorithms finding decompositions without using any search heuristics. The basic idea of these algorithms is to limit the search to some input variable partitions. This is done by using different functional methods to choose which partitions will be evaluated. These methods select partitions through Reed-Muller expansions, Fourier transforms, binary difference equations, and technology-based mappings [9], [14], [15].

When synthesis is performed for FPGA with heterogeneous structure an algorithm must take into account that such architectures deliver building blocks of different sizes, thus bound sets of different sizes have to be examined. For large functions the solution space is so huge that heuristic method for solving this problem has to be used.

In [16] the input variable partitioning method based on information relationship measures was presented, which produced optimal or sub-optimal results for factions of considerable size. However information relationship measures may be used for analyzing combinational functions described by truth tables only.

A very efficient method was presented in [17] where an application of evolutionary algorithms for solving input partitioning problem was proposed for decomposition based on cubes. This heuristic method of bound set selection turns out to be very efficient when applied for decomposition method based on ROBDDs as on [18]. In [19] a method based on evolutionary algorithms was proposed that was adopted for heterogeneous FPGAs.

IV. Iterative Input Variable Partitioning Algorithm

The method presented in this paper is based on observations made while examining evolutionary algorithms. The analysis of best possible solutions for given Boolean function results in interesting observations [17]. If partitioning of input variable set X into variables belonging to set U and belonging to set V that leads to optimal decomposition are examined, it can be noticed that certain variables appear in bound set often than others. This suggests that some variables are more predestined to be included in bound set and other to be included in free set when constructing good input variable partitions.

In [20] has been shown that there is a strong correlation of number of values in the sub-functions of the serial functional decomposition represented by the number of blocks in β_G with the decomposition's quality. Thus the number of blocks in β_G can be used to assess the quality of decomposition. Table I

presents the best solutions of input variable partitioning for *plan* example Boolean function from standard Microelectronics Center of North Carolina benchmark set [21]. This function has 13 inputs and 25 outputs.

Each row of Table I describes one partitioning of input variable set $X = x_1, \ldots, x_{13}$ into variables belonging to set U (marked by digit '1') and belonging to set V that leads to optimal decomposition (according to of the number of blanket β_G's blocks). It presents the best solutions for different sizes of sets V and U, as well as the frequency of appearance of given input variable in V set. It can be easily noticed that certain variables appear in bound set often than others. For example variable x_1 appears in V set for 16 solutions listed in Table I, while x_2 does not belong to V set for any of the best solutions. This suggests that some variables are more predestined to be included in V and other to be included in U set when constructing good input variable partitions.

There is another interesting observation that can be made. Let us assume that the V set is created in a way that it consists of variables that according to analysis of optimal U and V sets are least appropriate to be in bound set. As we could expect, the quality of decomposition for such selected V set will be very poor. However, if we move "good" variable from set U to set V and "bad" variable from set V to set U, the quality of decomposition most probably will be improved. If we further swap more "bad" and "good" variables further improvement will been obtained.

Let us create a set $V = x_2, x_3, x_4, x_9$ with variables that according to Table I are least appropriate to be in this set. Thus free set will be $U = x_1, x_5, x_6, x_7, x_8, x_{10}, x_{11}, x_{12}, x_{13}$. The quality of decomposition measured with the number of blanket β_G's blocks is 16 – the worst possible for this size of V. However, if "good" variable x_1 is moved from set U to set V and "bad" variable x_2 from set V to set U, the quality of decomposition is now 15, so it has improved. If we now swap variables x_3 and x_{12}, the decomposition will have quality 11, so further improvement has been obtained.

The method presented in this paper is based on identification of "good" variables by iteratively placing every input variable in test set of arbitrary selected variables. If improvement is obtained variable is marked as "promising". Appropriately performed selection process allows identifying V sets of very good quality. This algorithm can be described as "simplified" evolutionary algorithm, where individuals representing good solutions are composed of "good" genes identified in set of test individuals.

The outline of the algorithm is presented on List. 1. First, test set T is constructed, that will be used to evaluate the quality of input variables. For every variable that is not already in set T a bound set V is created that holds it along with variables from T. Then the quality of decomposition with such bound set is evaluated and stored in table Q. Then partial bound set V_p is constructed by selecting variables of best quality according to table Q. Since variables from set T were not evaluated none of them can be selected for partial bound set V_p. In the last step set V_p is extended with variable that delivers the best quality. In this step variables from set T are also evaluated and thus have a chance to appear in final bound set V_R. There are parameters (size of test set, size of partial bound set V_p) that can be used to control the algorithm.

```
1  //Input:  X - set of input variables
2  //  n - number of input variables
3  //  t - size of test set
4  //  p - size of partial bound set V_p
5  //  k - size of bound set V_R
6  //Result: V_R - bound set

8  // Create test set T
9  T = ∅;
10 for i = 1 to t begin
11   T = T ∪ x_i;
12 end
13 // Evaluate quality of variables
14 for i = t+1 to n begin
15   T_i = T ∪ x_i;
16   V = T_i;
17   U = X - V;
18   Q[i] = evaluate(U,V);
19 end
20 // Construct partial bound set
21 for i = 1 to k-p begin
22   for j = t+1 to n begin
23     if Q[j] is max begin
24       V_p = V_p ∪ x_j;
25       Q[j] = 0;
26     end
27   end
28 end
29 // Find best extension of partial bound
       set
30 for i = 1 to n begin
31   if x_i ∉ V_p begin
32     V = V_p ∪ x_i;
33     U = X - V;
34     Q[i] = evaluate(U,V);
35   end
36   else begin
37     Q[i] = 0;
38   end
39 end
40 // Construct final bound set
41 V_R = V_p;
42 for i = 1 to p begin
43   for j = 1 to n begin
44     if Q[j] is max begin
45       V_R = V_R ∪ x_j;
46     end
47     else begin
48       Q[i] = 0;
49     end
50   end
51 end
52 return V_R;
```

Listing 1. Outline of the bound set selection algorithm

TABLE I
BEST INPUT VARIABLE PARTITIONING PROBLEM SOLUTIONS OF PLAN EXAMPLE

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}
$\|U\| = 10, \|V\| = 3, \|\beta_G\| = 5$												
1	1	1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	1	1	1	0	1	0
1	1	1	1	1	1	0	1	1	1	0	0	1
1	1	1	1	1	0	0	1	1	1	1	0	1
1	1	1	1	0	1	0	1	1	1	1	0	1
1	1	1	1	0	0	1	1	1	1	1	0	1
1	1	1	0	1	1	0	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	0	1	1	0
0	1	1	1	1	1	1	1	1	0	1	0	1
0	1	1	1	1	1	1	0	1	1	1	1	0
0	1	1	1	1	1	1	0	1	1	1	0	1
0	1	1	1	1	1	1	0	1	0	1	1	1
0	1	1	1	1	1	0	1	1	1	1	1	0
0	1	1	1	1	1	0	1	1	1	1	0	1
0	1	1	1	1	0	1	1	1	1	1	0	1
0	1	1	1	0	1	1	1	1	1	1	0	1
0	1	1	0	1	1	1	1	1	1	1	1	0
$\|U\| = 9, \|V\| = 4, \|\beta_G\| = 7$												
0	1	1	1	1	1	0	1	1	1	1	0	0
$\|U\| = 8, \|V\| = 5, \|\beta_G\| = 11$												
0	1	1	1	1	1	0	1	1	1	0	0	0
0	1	1	1	1	1	0	1	1	0	1	0	0
0	1	1	1	1	1	0	0	1	1	1	0	0
$\|U\| = 7, \|V\| = 6, \|\beta_G\| = 17$												
0	1	1	0	1	1	0	1	1	0	1	0	0
Frequency of appearance in V set												
16	0	0	3	3	3	13	4	0	5	3	16	13

V. RESULTS

In Table II there is presented the comparison of results obtained with the new selection method and results obtained by systematic search and evolutionary algorithm described in [19]. For each method search time is presented, as well quality of decomposition for bound set of size from 5 up to 8. Systematic search delivers optimal results, however for large function the search requires unacceptably long time to find solution. Since evolutionary algorithm is a probabilistic method average quality for 10 searches is presented. Results obtained with this method are optimal or near-optimal, while the search time is much shorter than in case of systematic search.

Algorithm based on new method presented in this paper requires several times shorter search time, while delivering solutions of comparable quality. This method performs very well even for large Boolean functions. It can be further improved by applying beam search, that would allow identifying many "promising" V sets.

VI. CONCLUSIONS

The usefulness of functional decomposition for very complex systems depends on efficiency of an algorithm used for the selection of bound and free variables. There are efficient methods based on evolutionary algorithm. The method of bound set selection presented in this paper delivers results of similar or comparable quality to results obtained from the evolutionary algorithm, but does it several times faster. These features make the proposed heuristic method very useful for decomposition-based synthesis of large systems that will be implemented in heterogeneous programmable structures.

REFERENCES

[1] J. Cong and K. Yan, "Synthesis for fpgas with embedded memory blocks," in *Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays*, ser. FPGA '00. New York, NY, USA: ACM, 2000, pp. 75–82. [Online]. Available: http://doi.acm.org/10.1145/329166.329183

[2] S. Krishnamoorthy and R. Tessier, "Technology mapping algorithms for hybrid fpgas containing lookup tables and plas," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 5, pp. 545–559, 2003.

[3] M. Rawski, T. Łuba, Z. Jachna, and P. Tomaszewicz, *The Influence of Functional Decomposition on Modern Digital Design Process*. Springer, 2005, inbook 17. [Online]. Available: http://www.springerlink.com/content/p775582342t64847/

[4] T. Sasao, Y. Iguchi, and T. Suzuki, "On lut cascade realizations of fir filters," in *DSD*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 467–475.

TABLE II
COMPARISON OF THE NUMBER OF BLOCKS IN BLANKET β_G OBTAINED BY THE SYSTEMATIC METHOD, HEURISTIC METHOD BASED ON EVOLUTIONARY ALGORITHM AND NEW METHOD FOR DIFFERENT SIZE OF SET V

	Systematic search					Evolutionary algorithm					New method				
	Time [s]	5	6	7	8	Time [s]	5	6	7	8	Time [s]	5	6	7	8
9sym	0.6	6	7	6	4	35.4	6.0	7.0	6.0	4.0	0.6	6	7	6	4
Clip	0.927	14	18	22	30	25.0	14.0	18.0	22.0	30.0	1.0	14	18	22	30
TEST	1.185	1	1	1	1	27.2	1.0	1.0	1.0	1.0	1.3	1	1	1	1
Sao2	1.992	9	11	13	10	18.4	11.0	13.0	15.0	10.0	2.0	9	11	13	10
APEX4	3.804	29	57	113	208	112.6	29.0	57.0	113.0	208.0	3.8	29	57	113	208
f_12_10	61.9047	6	7	6	5	87.2	7.4	7.0	6.0	5.0	62.2	6	7	6	5
f_12_100	62.6307	16	28	30	15	63.7	31.0	45.0	30.0	15.0	62.7	16	28	30	15
f_12_1000	135.373	32	64	127	229	207.8	32.0	64.0	128.0	232.0	132.4	32	64	127	229
dek05_h	287.573	11	17	26	41	62.3	11.0	17.0	26.0	41.0	3.0	11	17	26	41
Alu4	290.576	25	39	71	88	58.8	25.8	42.2	77.8	96.7	3.0	25	39	78	125
Misex3	386.529	13	23	40	66	89.8	13.7	23.3	40.6	70.4	4.1	14	26	45	85
f9r	10872.2	–	–	–	–	325.9	1.0	1.0	1.0	1.0	17.3	1	1	1	1
Duke2	12240.9	–	–	–	–	40.9	7.0	8.0	9.0	11.0	2.0	7	9	11	13
Misex2	24018.6	–	–	–	–	23.2	2.9	2.9	3.5	4.0	1.0	2	2	3	5
Vg2	31756.6	–	–	–	–	31.1	5.0	6.3	6.8	7.5	1.5	5	6	8	14
Seq	1.77E+07	–	–	–	–	210.7	5.0	6.4	7.4	8.6	12.5	5	7	8	9
Apex1	2.23E+07	–	–	–	–	125.5	5.2	6.3	7.7	8.9	7.5	6	7	9	10
Apex3	7.94E+07	–	–	–	–	102.7	7.0	8.0	9.7	11.6	6.2	7	8	10	12
E64	1.67E+08	–	–	–	–	76.4	6.0	7.0	8.0	9.0	3.4	6	7	8	9
100X100	2.30E+09	–	–	–	–	30.1	1.0	1.0	1.0	1.0	1.4	1	1	1	1
APEX5	2.05E+11	–	–	–	–	335.5	2.1	2.2	2.5	3.6	44.4	2	2	2	2

[5] M. Rawski, P. Tomaszewicz, H. Selvaraj, and T. Łuba, "Efficient implementation of digital filters with use of advanced synthesis methods targeted fpga architectures," in *8th Euromicro Conference on DIGITAL SYSTEM DESIGN, Architectures, Methods and Tools DSD'05*, C. Wolinski, Ed., IEEE Computer Society. Portugal: IEEE Computer Society, 2005, inproceedings, pp. 460–466. [Online]. Available: http://portal.acm.org/citation.cfm?id=1100234

[6] C. Scholl, *Functional Decomposition with Application to FPGA Synthesis*. Kluwer Academic Publisher, 2001.

[7] M. Rawski, "Decomposition of boolean function sets," *Electronics and Telecommunications Quarterly*, vol. 53, no. 3, pp. 231–249, 2007.

[8] 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.

[9] T. Łuba, H. Selvaraj, M. Nowicka, and A. Kraśniewski, "Balanced multilevel decomposition and its applications in fpga-based synthesis," in *Novel Approaches in Logic and Architecture Synthesis*, A. M. Gabriele Saucier, Ed. Chapman and Hall, 1995, pp. 109–115.

[10] R. Murgai, N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Improved logic synthesis algorithms for table look up architectures," in *ICCAD*, 1991, pp. 564–567.

[11] Y.-T. Lai, M. Pedram, and S. B. K. Vrudhula, "Bdd based decomposition of logic functions with application to fpga synthesis," in *Proceedings of the 30th international Design Automation Conference*, ser. DAC '93. New York, NY, USA: ACM, 1993, pp. 642–647. [Online]. Available: http://doi.acm.org/10.1145/157485.165078

[12] T. Sasao, *Logic synthesis and optimization*, ser. Kluwer international series in engineering and computer science. Kluwer Academic Publishers, 1993. [Online]. Available: http://books.google.pl/books?id=GuaV_re0DF8C

[13] H. Sawada, T. Suyama, and A. Nagoya, "Logic synthesis for look-up table based fpgas using functional decomposition and support minimization," in *Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design*, ser. ICCAD '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 353–358. [Online]. Available: http://dl.acm.org/citation.cfm?id=224841.225063

[14] M. Perkowski, "A survey of literature on function decomposition. final report for summer faculty research program," Wright Laboratory, Sponsored by Air Force Office of Scientific Research, Bolling Air Force Base, DC and Wright Laboratory, Tech. Rep., September 1994.

[15] W. Wan and M. A. Perkowski, "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," in *Proceedings of the conference on European design automation*, ser. EURO-DAC '92. Los Alamitos, CA, USA: IEEE Computer Society Press, 1992, pp. 230–235. [Online]. Available: http://dl.acm.org/citation.cfm?id=159754.161757

[16] M. Rawski, L. Jóźwiak, and T. Łuba, "Efficient input support selection for sub-functions in functional decomposition based on information relationship measures," *Journal of Systems Architecture*, vol. 47, no. 2, pp. 137–155, 2001.

[17] M. Rawski, "Efficient variable partitioning method for functional decomposition," *Electronics and Telecommunications Quarterly*, vol. 53, no. 1, pp. 63–81, 2007.

[18] P. Morawiecki and M. Rawski, "Method of input variable partitioning in functional decomposition based on evolutionary algorithm and binary decision diagrams," *Proc. of IEEE 2008 Conference Human Systems Interaction*, 2008.

[19] M. Rawski, "Evolutionary algorithms in decomposition-based logic synthesis," in *Evolutionary algorithms*, E. Kita, Ed. Intech, 2011.

[20] M. Rawski, L. Jóźwiak, and T. Łuba, "The influence of the number of values in sub-functions on the effectiveness and efficiency of functional decomposition," *Proceedings of the 25th EUROMICRO Conference, IEEE Computer Society*, pp. 86–93, 1999.

[21] S. Yang, "Logic synthesis and optimization benchmarks user guide version 3.0," 1991.