INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2012, VOL. 58, NO. 4, PP. 335–344
Manuscript received November 12, 2012; revised Decemer, 2012. DOI: 10.2478/v10177-012-0046-y

# Modeling the Arithmetic Decomposition of DA-LUT Block for Heterogeneous FPGA Structures

Michał Staworko and Mariusz Rawski

***Abstract*—Distributed arithmetic is well known technique of designing FIR filters in FPGA devices. The quality of such filter implementation strongly depends on synthesis results of the DA-LUT block. Heterogeneity of modern FPGA structures introduces new possibilities into implementation process, that may lead to better results, but also makes it more complicated. This paper presents the simple mathematical model for estimating the necessary FPGA resources to implement DA-LUT using decomposition-based approach. The model takes into account the type of logic cells or memory blocks used for decomposition process. The proposed model is helpful to determinate the DA-LUT decomposition strategy for further automation of modified distributed arithmetic decomposition method.**

***Keywords*—Distributed arithmetic, FPGA, FIR filter, heterogeneous programmable structures.**

## I. Introduction

DISTRIBUTED ARITHMETIC (DA) provides multiplier-less implementation of Digital Signal Processing (DSP) functions. This technique gains popularity in Field Programmable Gate Arrays (FPGA) community since it is an algorithm that can perform multiplication with use of lookup tables (LUT), which are basic building blocks of FPGAs. DA is mainly used for implementation of FIR filters and may be used for time-frequency transformations. Longa et al. [1], [2] shown the promising application of DA-based filters in implementation of Discrete Wavelet Transform. Their solution reached three times better performance with slightly more occupied resources comparing to the implementation using lifting scheme. Meyer-Base et al.[3] presented comparison of several implementations of FIR filters designed using DA technique and RAGn algorithm [4]. DA-based filters reached greater frequency but occupied in average 70% more resources.

The main disadvantage of DA is the exponential growth of DA-LUT blocks size with the number of its inputs, which corresponds to the number of filter's coefficients. Several efforts have been made to reduce the DA-LUT size for efficient realization of DA-based designs. Martiez-Peiro et al. [5] proposed mapping the DA-LUT block into logic cells of FPGA structure by grouping filter's coefficients in small LUTs corresponding to the size of logic cells, and then adding up their outputs in a tree composed of two argument adders. Yoo and Anderson [6] showed a recursive method to reduce the block size introducing additional adders and $2\times1$ multiplexers. Using this technique, the DA-LUT with $N$ inputs can be implemented using $N$ $2\times1$ multiplexers and an adders tree. Meher et al. [7] presented the DA-based architecture in form of systolic array. This approach describes the architecture of serial and parallel DA-based filters as one dimensional and two dimensional systolic arrays. DA-LUT module was mapped as a set of $M$ input LUTs and adders connected in a cascade. It was experimentally verified that using 4 input LUTs gives the best results in terms of occupied resources (in experiments homogeneous structure of Xilinx Virtex-E containing 4 input logic cells was used). The forementioned methods that are based on the dividing DA-LUT block into equal-sized parts enable the rapid decomposition. Unfortunately those methods do not take into consideration the heterogeneous structure of programmable architectures and do not allow using the embedded memory blocks. With rapidly growth of traditional FPGA industry, heterogeneous logic blocks are often used in the nowadays FPGA architectures such as Xillinx Virtex-5 and Altera Stratix III series.

Recently, efforts have been made to develop methods based on functional decomposition that would allow for efficient utilization of heterogeneous structure of modern FPGAs. The method presented in [8], [9] is designed specifically to implement FIR filters using the concept of distributed arithmetic. In [10] advanced synthesis method based on functional decomposition was proposed that utilizes embedded memory block as large LUTs. Although those methods were reported to give good quality results, their practical application is limited only to filters containing small number of coefficients.

Arithmetic decomposition method presented by Rawski [11] seems to be not limited by the number of coefficients and particular size of the logic cell. Staworko and Rawski [12] showed that this decomposition method applied to heterogeneous FPGAs allows to synthesize filters operating on higher frequency and requiring fewer resources than typical implementations made with industry standard tool *Altera FIR Compiler* [13].

The presented paper introduces the simple mathematical model for estimation of FPGA resources necessary to implement the DA-LUT depending on the type of logic cells or memory blocks used for decomposition process. The proposed model is helpful to determinate the DA-LUT decomposition strategy for automation of modified distributed arithmetic decomposition method. This article is organized as follows. Section 2 introduces preliminary information concerning FPGA heterogeneity, structures of modern FPGA, modified distributed arithmetic concept including short revision of arithmetic decomposition concept and technology mapping. Section 3 presents the simple mathematical model for estimation of the arithmetic decomposition results, then in section 4 the evaluation of the model for different cases is presented and discussed. Finally section 5 presents short summary.

M. Staworko and M. Rawski are with Institute of Telecommunications, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland (e-mails: mstaworko@elka.pw.edu.pl; rawski@tele.pw.edu.pl).

## II. Preliminary Information

### A. Heterogeneous FPGAs

Heterogeneity in FPGA may be straightforwardly considered as the presence of a variety of general-purpose logic cells. It was shown, that the use of a different sized logic cells has a positive effect on the number of utilized resources and the performance of the system [14], [15]. The term heterogeneity is also used to describe the enhancement of the general purpose logic with dedicated circuits to gain FPGA performance. Extending logic cell functionality with registers, adders and fast carry lanes is known as *soft-fabric* heterogeneity, whereas introducing additional separate blocks not paired with general purpose logic, like embedded memory blocks, embedded multipliers is defined as *tile-based* heterogeneity [16]. It was also shown, that extending the general purpose logic may improve the performance of FPGA implementation and narrow the gap between FPGA and ASIC systems [17], [16]. Modern FPGAs offers both *soft-fabric* and *tile-based* heterogeneity and their general purpose logic structure is composed of various sized logic cells.

Logic cells are basic building blocks of FPGAs and may be used to implement combinational functions, sequential and arithmetic circuits. There is a tradeoff between the functionality and performance of FPGA circuit depending on the number of logic cell's inputs. The architecture of logic block evolved over time. One of the first commercial FPGA (Xilinx XC3000) had sophisticated structure, but existing synthesis tools were not able to use all of its potential [16], thus the next generations of FPGA were simplified to have homogeneous logic cells containing only 4-input LUTs and the register. This logic cell architecture was shown to give best functionality versus circuit area [18]. The newer considerations of the problem by Ahmed and Rose [19] showed, that in terms of propagation time and circuit area the best configuration is obtained by grouping 4 to 6-input LUTs in 3 to 10-element clusters. That results led to development of the new architecture of logic cells – the adaptive logic module (ALM) of Stratix II FPGA, capable of implementing logic function from 4 up to 7 input logic function; [20] shows the methodology and design decisions in details. The architecture of ALMs remains mostly unchanged in newer generations of high performance Altera devices form Straix III to Stratix V families. Heterogeneous Stratix II logic cells, comparing with homogeneous Virtex-4 structure, containing only 4-input LUT, both designed in 90 nm CMOS technology, were shown to give better performance with fever resources [21], [22]. Virtex-5 architecture is direct successor of Virtex-4, enhanced with heterogeneous logic cell structure. Solutions introduced to Virtex-5 logic cell remained mostly unchanged in modern generations of Xilinx FPGAs.

Both Stratix and Virtex FPGAs contain embedded memory blocks. Memory blocks are separated from the general purpose logic cells. The natural application of memory blocks is to implement large storage elements like RAMs, data buffers, FIFOs etc. more effectively than general purpose logic cells. However in designs, that do not require such a modules the great part of FPGA is left unused (in the largest device from Stratix III family embedded memory blocks constitute 23% of total die area [23]). Several projects have explored converting unused memory blocks into large lookup tables which implement combinational functions [24], [25]. One step further was made by the team lead by Łuba to use embedded memories in FPGA to synthesize synchronous circuits [26], [27], [28], [29].

TABLE I
Configuration of ALMs and Embedded Memory Blocks in Altera Stratix III

| ALM | MLAB | Block M9K | Block M144K |
|---|---|---|---|
| 2× [4 × 1] | 6 × 8 | 13 × 1 | 14 × 8 |
| [5 × 1] [3 × 1] | 6 × 9 | 12 × 2 | 14 × 9 |
| [5 × 1] [4 × 1] | 6 × 10 | 11 × 4 | 13 × 16 |
| [5 × 1] [5 × 1] | 5 × 16 | 10 × 8 | 13 × 18 |
| [6 × 1] | 5 × 18 | 10 × 9 | 12 × 32 |
| [6 × 1] [6 × 1] | 5 × 20 | 9 × 16 | 12 × 36 |
| | | 9 × 18 | 11 × 64 |
| | | 8 × 32 | 11 × 72 |
| | | 8 × 36 | |

Each Stratix III [30] ALM contains a look-up table 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 extended mode). 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, like two dual adders in arithmetic mode or two ternary adders in shared arithmetic mode 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. Table I presents configurations of logic elements and embedded memory blocks of Stratix III as LUTs of various sizes (number of inputs$\times$number of outputs).

The elementary programmable logic blocks in Xilinx Virtex-5 and Virtex-6 FPGAs [31], [32], called slices are organized in Configurable Logic Blocks (CLBs). The CLBs are the main logic resources for implementing sequential, as well as combinatorial circuits. Each slice consists of 4 function generators capable of implementing one 6-input ($6\times1$) logic function or two 5-input ($5\times2$) logic functions with shared inputs. Function generators in a slice can be paired using dedicated multiplexers to obtain two $7\times1$ logic function and all four slice function generators can be organized to fit one $8\times1$ logic function. Additionally fast carry logic can be used to implement four dual adders, appropriate configuration of logic generators as $3\times2$ compressors allows to implement four ternary adders in one slice. In Virtex-6 additional registers were introduced and the performance of ternary adders was improved. CLBs of Virtex-5 and 6 also support distributed memory – each look-up table can be configured to operate as a 64-bit memory. Because of the LUT structure of the Virtex-5 and 6, each LUT can be configured as a $64\times1$ or $32\times2$ RAM. However, when the slice is configured as RAM, it can no longer perform logic functions. The block RAM in Virtex-5 and 6 FPGAs [31], [33] stores up to 36K bits of

TABLE II
CONFIGURATION OF SLICES AND EMBEDDED MEMORY BLOCKS IN XILINX VIRTEX-5 AND VIRTEX-6

| SLICE | Block RAM 18K | Block RAM 36K |
|---|---|---|
| 4 × [5 × 2] | 14 × 1 | 15 × 1 |
| 4 × [6 × 1] | 13 × 2 | 14 × 2 |
| 2 × [7 × 1] | 12 × 4 | 13 × 4 |
| [8 × 1] | 11 × 9 | 12 × 9 |
| | 10 × 18 | 11 × 18 |
| | 9 × 36 | 10 × 36 |
| | | 9 × 72 |

data and can be configured as either two independent 18 Kb RAMs, or one 36 Kb RAM. Table II presents configurations of logic elements and embedded memory blocks of Virtex-5 and Virtex-6 as LUTs of various sizes (number of inputs×number of outputs)

Such architecture of modern programmable FPGAs greatly extends the space of possible solution during the process of mapping the design into FPGA resources. Unfortunately this heterogeneous structure of available logic resources also 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 [11].

## B. *Modified Distributed Arithmetic Concept*

The distributed arithmetic is a method of computing the sum of products:

$$y = \sum_{n=0}^{N-1} c[n] \times x[n], \tag{1}$$

In many applications, a general purpose multiplication is not required. This is the case of filter implementation, if filter coefficients are constant in time. The partial product term $x[n]$ × $c[n]$ becomes multiplication with a constant. Then taking into account the fact that the input variable x is a binary number:

$$x[n] = \sum_{b=0}^{B-1} 2^b x_b[n], \text{ where } x_b[n] \in [0,1] \tag{2}$$

the whole convolution sum can be described as shown in

$$y = \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} c[n] \times x_b[n] = \sum_{b=0}^{B-1} 2^b f(x_b) \tag{3}$$

Since $c[n]$ are constant the second sum in (3) can be implemented as a mapping $f(x_b)$, where $x_b = (x_b[0], x_b[1], ..., x_b[N\text{–}1])$. The efficiency of implementations based on this concept strongly depends on implementation of the function $f(x_b)$. The preferred implementation method is to realize the mapping $f(x_b)$ as the combinational module with $N$ inputs. The mapping $f$ is a lookup table (DA-LUT) that includes all the possible linear combinations of the coefficients and the bits of the incoming data samples [34].

The concept of arithmetic decomposition of DA-LUT block used in synthesis of FIR filters was proposed by Rawski as modified distributed arithmetic method [11]. It allows to use both general purpose programmable logic and embedded memory blocks. It is based solely on arithmetic transformations and is dedicated to decompose functions of DA-LUT block. The proposed concept takes advantage of distinctive structure of DA-LUT block. The key operations are: organizing filter coefficients in groups and if necessary splitting the values of filter coefficients into groups containing more significant and less significant bits.

The operation of organizing $N$ filter coefficients into $L$ groups, each containing $K_i$ elements where:

$$\sum_{l=0}^{L-1} K_l = N, \tag{4}$$

moreover $K_l$ can only be equal to one of available sizes $Lin_i$, is given by the expression:

$$\begin{aligned} y &= \sum_{l=0}^{L-1} \sum_{k=0}^{K_l-1} c[n_k^l] \times x[n_k^l] = \\ &= \sum_{b=0}^{B-1} 2^b \sum_{l=0}^{L-1} \sum_{k=0}^{K_l-1} c[n_k^l] \times x_b[n_k^l] = \\ &= \sum_{b=0}^{B-1} 2^b \sum_{l=0}^{L-1} f_l(x_b^l). \end{aligned} \tag{5}$$

In this case function $f(x_b)$ has been decomposed into $L$ functions $f_l(x_b^l)$. The sum is partitioned into $L$ independent DA-LUTs. This allows to implement DA architecture with smaller DA-LUTs and additional adders. Each DA-LUT has $K_l$ inputs and $(\lceil \log_2 K_l \rceil + q)$ outputs, where $q$ denotes the number of greatest coefficient's bits. The operation of grouping coefficients allows to significantly reduce the size of DA-LUT. In contrast the operation of splitting does not reduce the DA-LUT block, it only allows to match the size of DA-LUT block to the available LUTs. In fact the splitting operation makes the decomposed block even greater because the total size of obtained sub-blocks is greater than input block and the decomposition process introduces additional adder. The presented model does not include the splitting process, thus it is not presented in details.

Recursive and alternating use of these operations allows for decomposing the DA-LUT block into sub-blocks, that have the desired number of inputs and outputs. This allows the adjustment of the parameters of the sub-blocks to the size of LUTs corresponding to the available resources of FPGA.

The architecture of FIR filters designed for purposes of modified distributed arithmetic method was presented in details in [12]. The main modules of the parallel filter structure are tapped delay line, optional preadders, transposition, DA-LUTs and final adder tree. For the purpose of described model the most important part is DA-LUT structure, depicted in Fig. 1. The DA-LUT block contains the sub-blocks that are result of aforementioned arithmetic decomposition process and the adder tree. The advantage of such organization of the module is single layer of look-up tables which outputs are added up in a single adder tree.

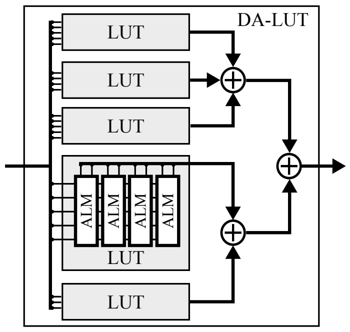

Fig. 1. Structure of DA-LUT module.

### C. Technology Mapping

Filters designed according to modified distributed arithmetic method contain three generic types of elements: registers, adders and LUTs. Delay tapped line registers and pipeline register are implemented directly in programmable fabric flip-flops. Adders are implemented with dedicated arithmetic circuits located in logic cells. Dual and ternary adders are implemented in Altera Stratix III FPGA soft-fabric using arithmetic and shared arithmetic logic-cell modes respectively. One ALM is able to fit two dual or ternary adders. Both types of adders operate at the same speed. In Xilinx Viretx-5 and 6 FPGAs the dual adders are made directly using carry select adder, to make ternary adder the function generators has to be configured as 3x2 compressors. One slice may fit up to four dual or ternary adders. In contrast to Altera solutions, Xilinx's dual adders operates on higher frequencies than ternary. The disproportion between operating speed of dual and ternary adder became smaller in Virtex-6 comparing to Virtex-5. In both Virtex and Stratix families the summing results may be stored in registers, that are placed directly after adder circuit in the same logic cell.

LUTs may be implemented in two ways: using the embedded block memories or logic cells. Block memories have output's bit-width up to 72 bits. It allows to directly map the contents of LUT or its part, after performing split operation. Memories have embedded address line registers, which does not allow to implement combinatorial LUTs. In Stratix III, as well as in Virtex-5 and 6 to achieve full operating frequency the output data register is required. This requirement introduces additional delay cycles. Thus using memory blocks as large LUTs with full operating frequency introduces two clock cycles delay.

Less intuitive is the mapping of LUT in logic cells, which has maximum two outputs. The problem of small number of outputs is solved by mapping the subsequent bit-planes of LUT into seperate logic cells. Thereby cells are organized in group and share the same inputs, but each cell implements different logic function. For this reason it is impossible in Stratix III architecture to use the configuration 6$\times$2, which requires the use of cells that have common truth table. The outputs from the logic cells may be directly connected to the register, which is placed in the same ALM or slice, in exception of Virtex-5 and configuration 5$\times$2, because this architecture is built from slices containing four logic function generators, and only four registers. This problem does not exist in Virtex-6 where each slice contains eight registers. Also the extended mode of logic cell in Stratix III supporting 7$\times$1 configuration is capable to fit only subset of 7-inputs 1-output logic functions.

TABLE III
HETEROGENEOUS STRATIX III COMPONTNES DIE AREA RATIOS [23]

| component | die area ratio |
|---|---|
| ALM | 1 |
| M9K | 28,7 |
| M144K | 267 |

The cost of mapping a DA-LUT into logic cells and embedded memory blocks may be considered in terms of different criteria. To achieve maximal operational frequency the embedded memory blocks introduce two delay cycles while logic cells introduce only one. Mapping DA-LUT into embedded memory blocks has bigger impact on routing, because memory blocks are located only in particular areas of the FPGA die. The comparison of ratios between ALM and particular embedded memory blocks die area in Stratix III is shown in Tab. III. Similar comparison for Xilinx devices is not available, because the vendor does not publish this information. The area estimation, based on the analysis of Virtex-II die picture may be found in [35], but it is useless for presented here considerations since Virtex-5 and Virtex-6 have significantly different logic cell structure. The use of embedded memory blocks as LUT instead of logic cells may become priceless, when implementation requires the use of logic cells for other purposes.

## III. PROPOSED MODEL OF DA-LUT DECOMPOSITION

Determination of technology mapping enables to develop simple model of uniform decomposition of DA-LUT block. The uniform decomposition means that only one type of logic cell is used. It is also assumed that the decomposed DA-LUT block coefficients has the same bit width $q$ and DA-LUT block architecture after decomposition process has structure shown schematically in Fig. 1. The proposed model allows to easily estimate the FPGA resources needed to fit DA-LUT after decomposition.

When applying grouping operation the number of coefficients groups is:

$$L = \left\lceil \frac{N}{K} \right\rceil , \tag{6}$$

where $N$ stands for the number of input coefficients (DA-LUT block inputs) and $K$ is the number of group's elements (number of destination logic cell inputs). The number of guard bits $q_g$ to ensure that $K$ coefficients may be stored in $K$- inputs LUT is:

$$q_g = \lceil \log_2 K \rceil . \tag{7}$$

Each group of coefficients is summed in a tree composed from $s$-operands adders (in Stratix III, Virtex-5 and 6 $s \in \{2, 3\}$). The number of adder tree level is:

$$T_L = \lceil \log_s L \rceil , \tag{8}$$

and the number of adder tree elements is

$$T_s = \sum_{l=1}^{T_l} \left\lceil \frac{L}{s^l} \right\rceil . \tag{9}$$

The total number of guard-bits in adder tree is given by the expression:

$$q_t = \sum_{l=1}^{T_l} \left( \lceil \log_2(s^l) \rceil \left\lceil \frac{L}{s^l} \right\rceil \right) \tag{10}$$

where the first term is the number of guard bits introduced on each tree level and the second term is the number of elements on each tree level. The total number of used logic cell in grouping operation is given by:

$$LC = L \left\lceil \frac{q + q_g}{Lout} \right\rceil + \left\lceil \frac{T_s\,(q + q_g) + q_t}{Lout_s} \right\rceil, \tag{11}$$

where the first term is the number of logic cells that is used for mapping the DA-LUT sub-block, while the second term shows the number of logic cells used for adder tree, $Lout$ is the number of outputs of logic cells used to fit DA-LUT sub-blocks and $Lout_s$ stands for the number of adder outputs mapped into single logic cell.

## IV. Simulation Results

The results presented in this section are the evaluation of the model (11) for parameters corresponding to cell configuration of Startix III and Virtex family FPGA devices presented in Tab. I and Tab. II and technology mapping described in section II-C.

Figure 2 presents the results of estimation of the resources required to fit DA-LUT block after arithmetic decomposition into various logic cells available in Stratix III FPGA, depending on the number of coefficients, coefficients bit width and using ternary adder tree. It may be noticed that for small coefficients bit widths the best choice is to fit them into 4×2 cells. This is due to the rounding toward positive infinity while computing the value of $q_g$ (7). For LUTs with four inputs, only two guard bits are needed, while using five inputs LUTs three guard bits are necessary, hence the mapping into 4×2 cells requires smaller number of FPGA resources. This is a special case, which is not valid for cases, when coefficients bit width becomes greater than four. For certain ranges of the number of decomposed DA-LUT block inputs, due to the smaller number of adders in the adder tree, the mapping using 7×1 cells gives minor saving comparing to 5×2. The obtained results from presented model show that in general the best results of decomposition process in Stratix III architecture is obtained while mapping LUTs into 5×2 logic cells.

Figure 3 shows the estimation of utilized resources in Stratix III device after decomposition of DA-LUT block using 5×2 logic cells and embedded memory blocks. The ratio of die area of M9K and M144K to the area of ALM was computed according to values presented in Tab. III. It can be clearly seen, that decomposition of DA-LUT block using M144K memory gives poor results comparing to M9K and logic cells. The only memory configuration, which usage may give better decomposition results than logic cells is 8×36 M9K. It allows to save up to 15% of device resources comparing to logic cells while mapping the coefficients which bit-width is smaller than the memory depth. Figure 3c presents the case where the coefficients bit width is greater than 8×36 M9K memory depth. The presented model does not consider the application of splitting operation in decomposition process, while the use of this operation would allow for more efficient utilization of memory blocks. Wider configurations of memory do not allow more efficient mapping of the sub-blocks of DA-LUT because the size of the sub-block exponentially grows with the number of inputs. The model analysis allows to rank the memory configurations in terms of area utilized by the DA-LUT block after the decomposition process. The order of memory configuration is presented in Tab. IV

TABLE IV
The Order of Stratix III Embedded Memory Configurations Considering the Quality of Mapping DA-LUT Block in Arithmetic Decomposition Process

| | type | configuration |
|---|---|---|
| 1 | M9K | 8 × 36 |
| 2 | M9K | 8 × 32 |
| 3 | M9K | 9 × 18 |
| 4 | M9K | 9 × 16 |
| 5 | M9K | 10 × 9 |
| 6 | M9K | 10 × 8 |
| 7 | M144K | 11 × 72 |
| 8 | M144K | 11 × 64 |
| 9 | M9K | 11 × 4 |
| 10 | M144K | 12 × 36 |
| 11 | M144K | 12 × 32 |
| 12 | M9K | 12 × 2 |
| 13 | M144K | 13 × 18 |
| 14 | M144K | 13 × 16 |
| 15 | M9K | 13 × 1 |
| 16 | M144K | 14 × 9 |
| 17 | M144K | 14 × 8 |

Figure 4 presents the analysis results of the arithmetic decomposition of DA-LUT block using dual adder tree and logic cells in Viretx-5 or Virtex-6 FPGA. Under the assumption that the length of the critical path is limited to a single level of combinatorial logic, the natural choice for mapping the DA-LUT block in Virtex-5 is 6×1 configuration of logic cell, because the Virtex-5 slice architecture has only one register available per function generator output. In Virtex-6 devices, where slice includes additional four flip-flops and every function generator output may be registered, the choice that allows to decompose the DA-LUT to the smallest number of resources is using 5×2 LUT configuration. Additionally using 5×2 gives better results, when using ternary adder tree. Straightforward analysis of DA-LUT decomposition results utilizing embedded memory in Xilinx devices is impossible, because there is no information about the area ratio of building blocks for these FPGAs, like it was presented for Altera devices. The analogy to results of decomposition process obtained for Stratix III device suggests, that using embedded memory blocks in Virtex family devices may be similarly inefficient. Additionally the smallest embedded memory blocks in Virtex devices are twice as large as Stratix III M9K, which also suggests, that this type of mapping may give worse results than using only logic cells. Table V shows the arrangement of embedded memory blocks in Virtex FPGA family due to expected quality of decomposition results of DA-LUT block.

Figures 5a and 5b present the qualitative comparison of DA-LUT block decomposition results depending on the number of filter coefficients and the coefficients bit width in Altera Stratix III and Virtex family FPGA. Both for Stratix and Virtex architecture 5×2 logic cell configuration provides the

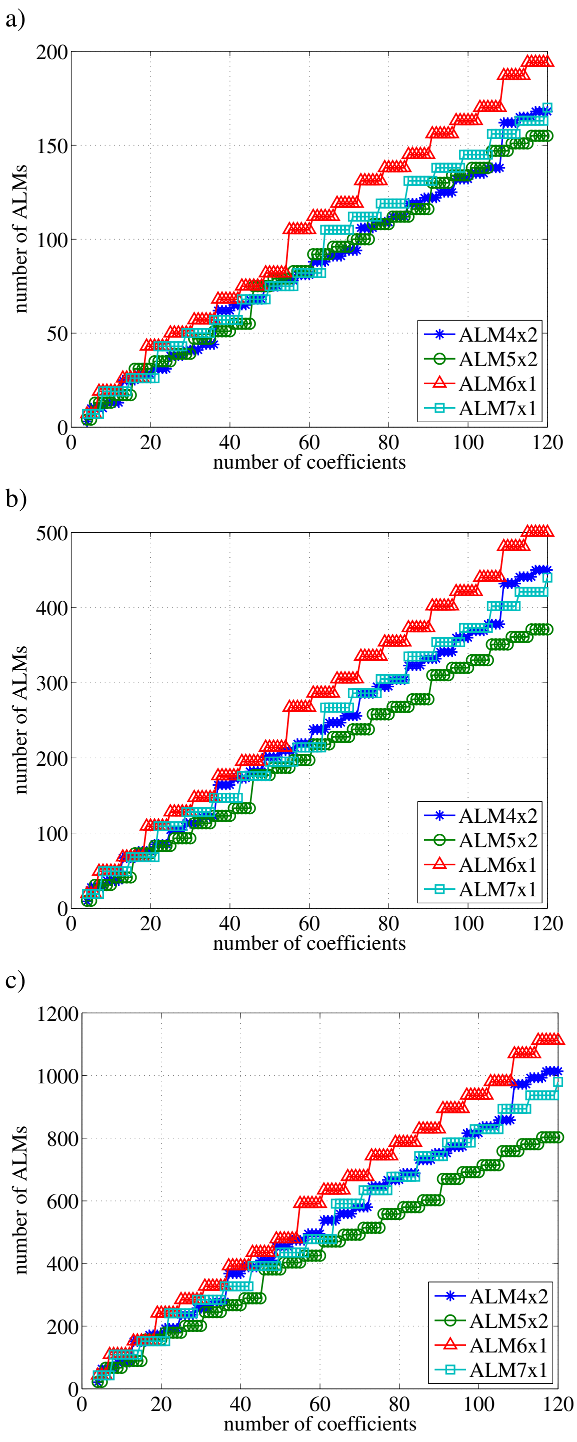

Fig. 2. The estimation of utilized Stratix III FPGA resources after decomposition of DA-LUT block using ternary adders, depending on number of coefficients, and type of logic cells, for various coefficients bit width a) 4 bits b) 16 bits c) 40 bits.

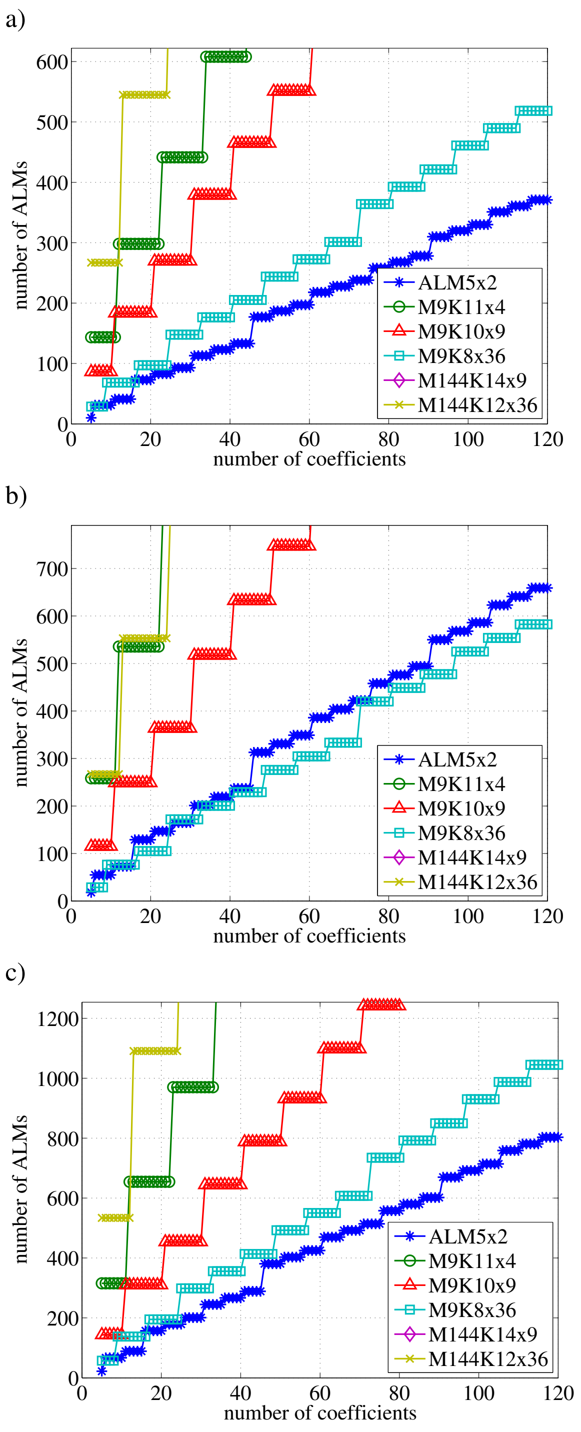

Fig. 3. The estimation of utilized Stratix III FPGA resources after decomposition of DA-LUT block using ternary adders, including embedded memory blocks, depending on number of coefficients, and type of logic cells, for various coefficients bit width a) 4 bits b) 16 bits c) 40 bits.

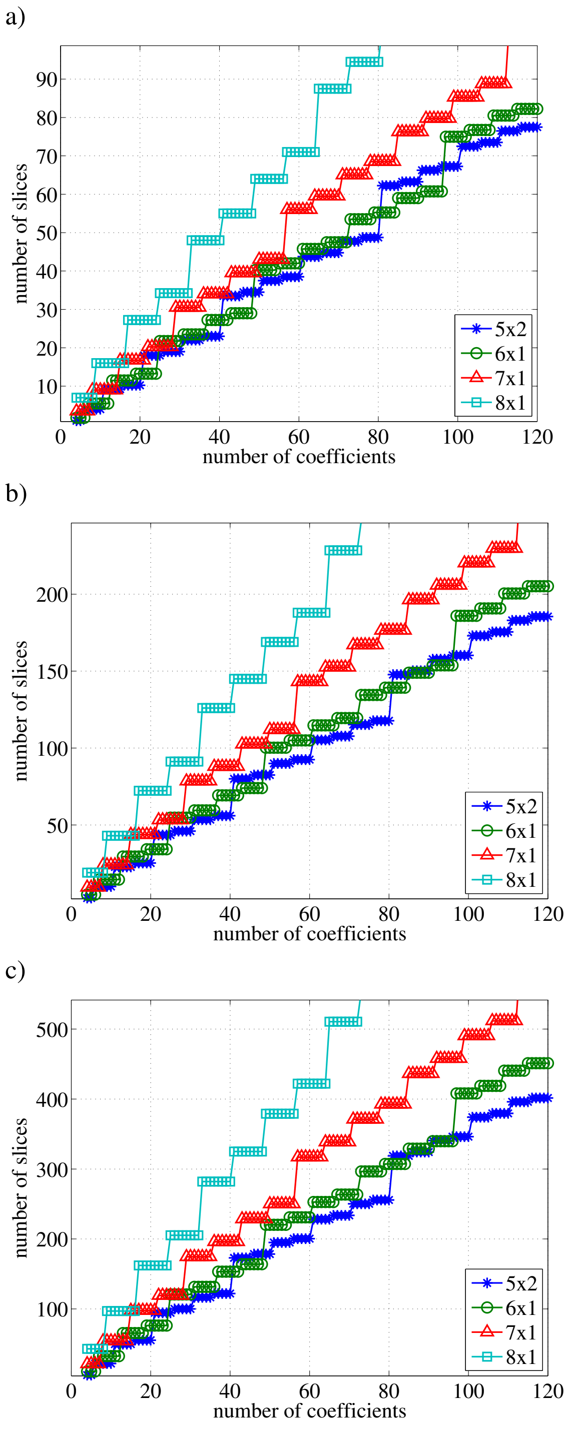

Fig. 4. The estimation of utilized Xilinx Virtex-5 and Virtex-6 FPGA resources after decomposition of DA-LUT block using dual adders, depending on number of coefficients, and type of logic cells, for various coefficients bit width a) 4 bits b) 16 bits c) 40 bits .

TABLE V
THE ORDER OF VIRTEX-5 AND VIRTEX-6 EMBEDDED MEMORY CONFIGURATIONS CONSIDERING THE QUALITY OF MAPPING DA-LUT BLOCK IN ARITHMETIC DECOMPOSITION PROCESS

| | type | configuration |
|---|---|---|
| 1 | 18K | 9 × 36 |
| 2 | 36K | 9 × 72 |
| 3 | 18K | 10 × 18 |
| 4 | 36K | 10 × 36 |
| 5 | 36K | 11 × 9 |
| 6 | 18K | 11 × 18 |
| 7 | 18K | 12 × 9 |
| 8 | 36K | 12 × 4 |
| 9 | 18K | 13 × 2 |
| 10 | 36K | 13 × 4 |
| 11 | 18K | 14 × 1 |
| 12 | 36K | 14 × 2 |
| 13 | 36K | 15 × 1 |

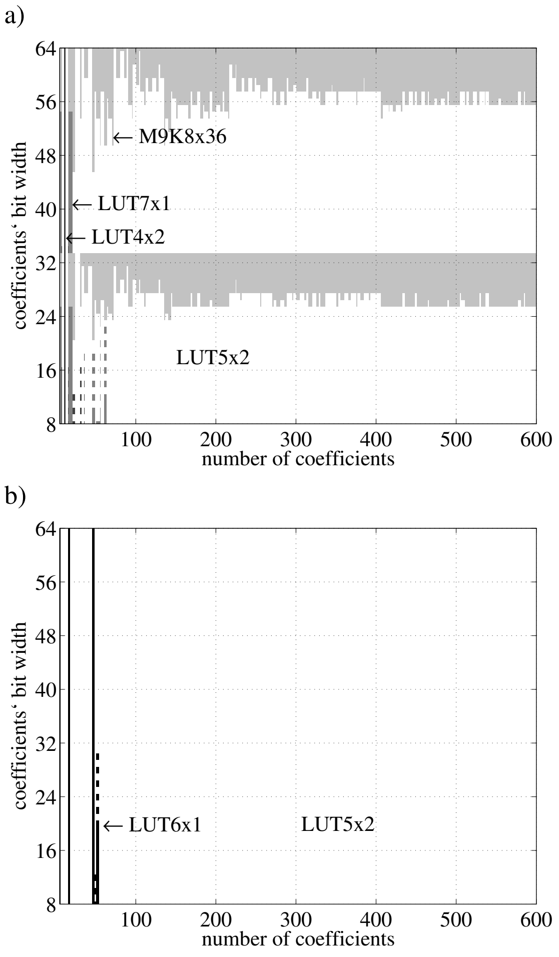

Fig. 5. Qualitative comparison of decomposition using ternary adders, depending on the number of coefficients and coefficients bit width a) Altera Stratix III b) Xilinx Virtex-5 and -6.

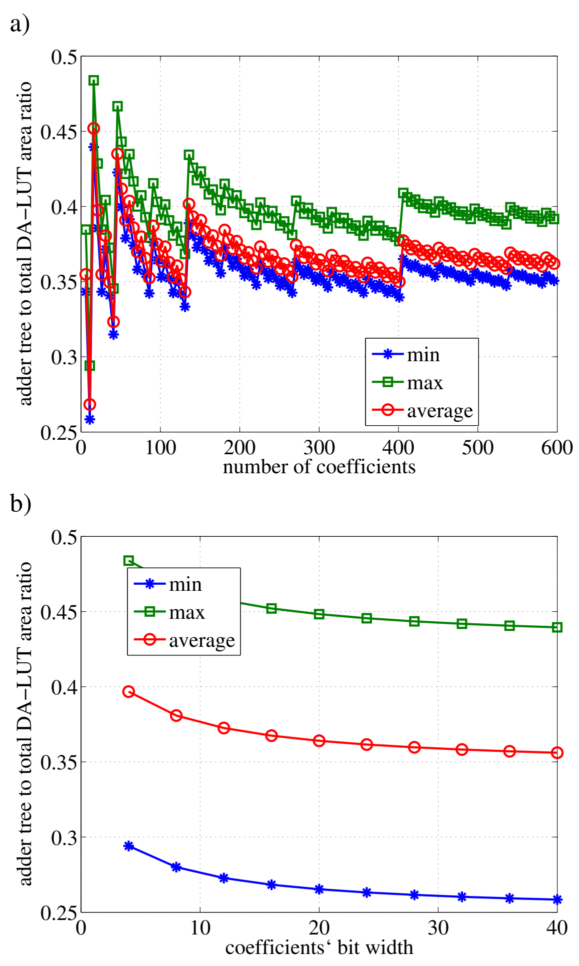

Fig. 6. The ratio of FPGA resources used for adder tree to DA-LUT after arithmetic decomposition in Startix III using 5×2 logic cells configuration and ternary adder tree.

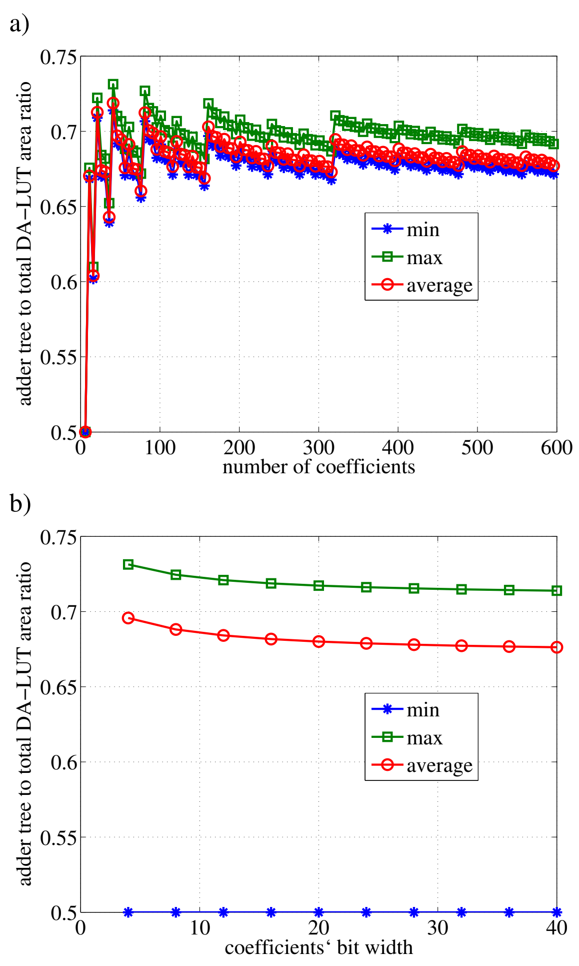

Fig. 7. The ratio of FPGA resources used for adder tree to DA-LUT after arithmetic decomposition in Virtex-5 using 6×1 logic cells configuration and dual adder tree.

best results in the decomposition process. For some ranges of coefficient number, grouping in larger clusters, which gives smaller number of adders in adder tree, may give minor advances. Embedded memory blocks M9K in Stratix III allow to map efficiently coefficients with large bit widths. For Xilinx devices, due to lack of data it was not possible to compare the decomposition results that include embedded memory mapping.

Figure 6 shows the ratio of FPGA resources utilized by the adder tree to the whole DA-LUT block. The presented results were obtained with the decomposition process into 5×2 logic cells and using ternary adder tree in Stratix III device. Figure 7 shows similar data for Virtex-5 and decomposition using 6×1 logic cells and dual adder tree. In Altera Stratix III devices the adder tree is in average 40% of the DA-LUT block. When using dual adders as presented on example of Virtex-5 the adder tree stands for up to 74% of the DA-LUT block resources. In Virtex family, when using ternary adder tree and 5×2 logic cells the adder tree consumes 50% of the DA-LUT block. The different ratios of ternary tree to whole DA-LUT block between Altera and Virtex FPGAs comes from the different architecture of logic cells. Stratix III logic cell implements two ternary adders, and Virtex-5 or Virtex-6 function generator paired with carry chain implements only one ternary adder, one slice may implement four ternary adders, hence Stratix III implements two times more adders per logic cell than Virtex family FPGAs.

Figures 6a and 7a present contribution of adder tree in whole DA-LUT resources depending on the number of coefficients. Spikes and dips of adder tree structure contribution in DA-LUT block resources depending on the number of coefficients are caused by including new tree levels and introducing more than one adder to the structure. The best adder tree utilization, thus less DA-LUT block resources part utilized by adders, is when all inputs of all bottom layer adders are used. The lowest adder tree to DA-LUT resources ratio is, when the number of lowest level nodes inputs are $3^n$ and $2^n$ for ternary and dual adder tree respectively, where $n$ denotes number of inputs to the adder tree.

The coefficients bit width does not have significant impact on the size of contribution of adder tree to the DA-LUT block utilization as shown in Figs. 6b and 7b. For small bit widths the contribution of adder tree in DA-LUT block resources is greater because for large number of tree level the number guard bits may become larger than the representation of coefficient.

Presented simulation results show, that the adder tree is very important part of decomposed DA-LUT block and the adder tree generation process should be done very precisely. Especially, when FPGA architecture allows only for implementation of dual adders the proper generation of adder tree should be the priority.

## V. Summary

The article presented a simple mathematical model, that allows to estimate the utilization of resources in arithmetic decomposition process of DA-LUT block. Also the evaluation of this model is discussed for configurations of FPGA resources native to Startix III and Virtex-5 and Virtex-6 FPGA. The basic conclusion of model evaluation for both, Stratix III and Virtex-5 family is that there exists one particular logic cell configuration, that in most cases guaranties utilizing minimal number of resources considering various number of coefficients and various coefficient bit width. For Stratix III and Virtex-6 the dominant configuration is 5×2, and for Virtex-5 is 6×1. It was explicitly shown in Stratix III that in terms of FPGA die area the only memory configuration that may give better decomposition performance is M9K 8×36. This is only configuration able to outperform the 5×2 configuration. The evaluation of proposed model for different memory types allows to arrange the embedded memory blocks according to their usefulness in decomposition process. It was also shown that appropriate implementation of an adder tree plays important role in structure of decomposed DA-LUT. When adder tree is composed of dual adders it may utilize up to 75% resources of DA-LUT block. Thus proper synthesis process of adder tree is very important for efficient implementation of DA-LUT block.

## References

[1] P. Longa, A. Miri, and M. Bolic, "A Flexible Design of Filterbank Architectures for Discrete Wavelet Transforms," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 3, Apr. 2007, pp. III–1441–III–1444, DOI: 10.1109/ICASSP.2007.367118.

[2] ——, "Modified distributed arithmetic based architecture for discrete wavelet transforms," *Electronics Letters*, vol. 44, no. 4, pp. 270–271, 2008, DOI: 10.1049/el:20082418.

[3] U. Meyer-Baese, J. Chen, C. H. Chang, and A. Dempster, "A Comparison of Pipelined RAG-n and DA FPGA-based Multiplierless Filters," in *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, Dec. 2006, pp. 1555–1558, DOI: 10.1109/APCCAS.2006.342540.

[4] A. Dempster and M. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 42, no. 9, pp. 569–577, Sep. 1995, DOI: 10.1109/82.466647.

[5] M. Martinez-Peiro, J. Valls, T. Sansaloni, A. Pascual, and E. Boemo, "A comparison between lattice, cascade and direct form FIR filter structures by using a FPGA bit-serial distributed arithmetic implementation," in *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on*, vol. 1, 1999, pp. 241–244, DOI: 10.1109/ICECS.1999.812268.

[6] H. Yoo and D. Anderson, "Hardware-efficient distributed arithmetic architecture for high-order digital filters," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, vol. 5, Mar. 2005, pp. v/125–v/128, DOI: 10.1109/ICASSP.2005.1416256.

[7] P. Meher, S. Chandrasekaran, and A. Amira, "FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic," *Signal Processing, IEEE Transactions on*, vol. 56, no. 7, pp. 3009–3017, Jul. 2008, DOI: 10.1109/TSP.2007.914926.

[8] T. Sasao, Y. Iguchi, and T. Suzuki, "On LUT cascade realizations of FIR filters," in *Digital System Design, 2005. Proceedings. 8th Euromicro Conference on*, Sep. 2005, pp. 467–474, DOI: 10.1109/DSD.2005.82.

[9] T. Sasao, "Analysis and synthesis of weighted-sum functions," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 5, pp. 789–796, May 2006, DOI: 10.1109/TCAD.2006.870407.

[10] M. Rawski, P. Tomaszewicz, H. Selvaraj, and T. Łuba, "Efficient Implementation of Digital Filters with Use of Advanced Synthesis Methods Targeted FPGA Architectures," in *Proceedings of the 8th Euromicro Conference on Digital System Design*, ser. DSD '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 460–466, DOI: 10.1109/DSD.2005.81.

[11] M. Rawski, "Modified Distributed Arithmetic Concept for Implementations Targeted at Heterogeneous FPGAs," *International Journal of Electronics and Telecommunications*, vol. 56, no. 4, pp. 345–350, Nov. 2010, DOI: 10.2478/v10177-010-0045-. [Online]. Available: http://versita.metapress.com/content/048788042483024H

[12] M. Staworko and M. Rawski, "Application of Modified Distributed Arithmetic Concept in FIR Filter Implementations Targeted at Heterogeneous FPGAs," *Przegląd Elektrotechniczny (Electrical Review)*, vol. 88, no. 6, pp. 240–246, Jun. 2012.

[13] Altera. (2011, May) FIR Compiler User Guide. Altera Corporation.

[14] J. He and J. Rose, "Advantages of heterogeneous logic block architecture for FPGAs," in *Custom Integrated Circuits Conference, 1993., Proceedings of the IEEE 1993*, May 1993, pp. 7.4.1–7.4.5, DOI: 10.1109/CICC.1993.590578.

[15] J. Cong and S. Xu, "Delay-optimal technology mapping for FPGAs with heterogeneous LUTs," in *Design Automation Conference, 1998. Proceedings*, Jun. 1998, pp. 704–707.

[16] I. Kuon, R. Tessier, and J. Rose, "FPGA Architecture: Survey and Challenges," *Found. Trends Electron. Des. Autom.*, vol. 2, no. 2, pp. 135–253, Feb. 2008, DOI: 10.1561/1000000005.

[17] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 203–215, Feb. 2007, DOI: 10.1109/TCAD.2006.884574.

[18] J. Rose, R. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate arrays: the effect of logic block functionality on area efficiency," *Solid-State Circuits, IEEE Journal of*, vol. 25, no. 5, pp. 1217–1225, Oct. 1990, DOI: 10.1109/4.62145.

[19] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 3, pp. 288–298, Mar. 2004, DOI: 10.1109/TVLSI.2004.824300.

[20] D. Lewis, E. Ahmed, G. Baeckler, V. Betz, M. Bourgeault, D. Cashman, D. Galloway, M. Hutton, C. Lane, A. Lee, P. Leventis, S. Marquardt, C. McClintock, K. Padalia, B. Pedersen, G. Powell, B. Ratchev, S. Reddy, J. Schleicher, K. Stevens, R. Yuan, R. Cliff, and J. Rose, "The Stratix II logic and routing architecture," in *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, ser. FPGA '05. New York, NY, USA: ACM, 2005, pp. 14–20, DOI: 10.1145/1046192.1046195.

[21] Altera. (2005, Aug.) Stratix II vs. Virtex-4 Density Comparison. Altera Corporation.

[22] ——. (2005, Aug.) Stratix II vs. Virtex-4 Performance Comparison. Altera Corporation.

[23] H. Wong, V. Betz, and J. Rose, "Comparing FPGA vs. custom cmos and the impact on processor microarchitecture," in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, ser. FPGA '11. New York, NY, USA: ACM, 2011, pp. 5–14, DOI: 10.1145/1950413.1950419.

[24] J. Cong and S. Xu, "Technology mapping for FPGAs with embedded memory blocks," in *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, ser. FPGA '98. New York, NY, USA: ACM, 1998, pp. 179–188, DOI: 10.1145/275107.275138.

[25] S. Wilton, "Heterogeneous technology mapping for area reduction in FPGAs with embedded memory arrays," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 19, no. 1, pp. 56–68, Jan. 2000, DOI: 10.1109/43.822620.

[26] G. Borowik, T. Łuba, and B. J. Falkowski, "Logic synthesis method for pattern matching circuits implementation in FPGA with embedded memories," in *DDECS*, 2009, pp. 230–233, DOI: 10.1109/DDECS.2009.5012135.

[27] G. Borowik, "Improved State Encoding for FSM Implementation in FPGA Structures with Embedded Memory Blocks," *Electronics and Telecommunications Quarterly*, vol. 54, pp. 9–28, Mar. 2008.

[28] M. Rawski, T. Łuba, and B. J. Falkowski, "Logic synthesis method for FPGAs with embedded memory blocks," in *ISCAS*. IEEE, May 2008, pp. 2014–2017, DOI: 10.1109/ISCAS.2008.4541842.

[29] M. Rawski, G. Borowik, T. Łuba, P. Tomaszewicz, and B. Falkowski, "Logic synthesis strategy for fpgas with embedded memory blocks," in *Mixed Design of Integrated Circuits Systems, 2009. MIXDES '09. MIXDES-16th International Conference*, Jun. 2009, pp. 296–301.

[30] Altera, "Stratix III Device Handbook," Altera Corporation, Mar. 2011.

[31] Xilinx, "Virtex-5 FPGA User Guide," Xilinx Inc., p. 385, Mar. 2012.

[32] ——, "Virtex-6 FPGA Configurable Logic Block User Guide," Xilinx Inc., p. 385, Mar. 2012.

[33] ——, "Virtex-6 FPGA Memory Resources User Guide," Xilinx Inc., p. 385, Apr. 2011.

[34] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 3rd ed. Springer Publishing Company, Incorporated, 2007.

[35] M. J. Beauchamp, S. Hauck, K. D. Underwood, and K. S. Hemmert, "Embedded floating-point units in FPGAs," in *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, ser. FPGA '06. New York, NY, USA: ACM, 2006, pp. 12–20, DOI: 10.1145/1117201.1117204.