Michał STAWORKO, Mariusz RAWSKI

Politechnika Warszawska, Instytut Telekomunikacji

Application of Modified Distributed Arithmetic Concept in FIR Filter Implementations Targeted at Heterogeneous FPGAs

Abstract. *Distributed arithmetic is a very efficient method for implementing digital FIR filters in FPGA structures. In this approach general purpose multipliers of traditional MAC implementations are replaced by combinational LUT blocks. Since LUT blocks can be of considerable size thus, the quality of digital filter implementation highly depends on efficiency of logic synthesis algorithm that maps it into FPGA resources. Modern FPGAs have heterogeneous structure, there is a need for quality algorithms to target these structures and the need for flexible architecture exploration aiding in appropriate mapping. The paper presents an application of modified distributed arithmetic concept that allows for very efficient implementation of FIR filters in heterogeneous FPGA architectures.*

Streszczenie*. Arytmetyka rozproszona jest bardzo wydajną metodą implementacji filtrów SOI w układach FPGA. Pozwala na zastąpienie kosztowych układów mnożących tablicami prawdy (LUT). Dla filtrów wysokich rzędów tablice LUT osiągają wielkie rozmiary, dlatego jakość implementacji filtru zależy głównie od jakości dekompozycji tej tablicy. Artykuł przedstawia nową metodę dekompozycji tablic LUT filtrów SOI dedykowaną do heterogenicznych stukrur rekonfigurowalnych. (****Zastosowanie metody zmodyfikowanej arytmetyki rozproszonej do implementacji filtrów SOI w heterogenicznych układach FPGA****).*

Keywords: modified distributed arithmetic, FIR filters, heterogeneous programmable structures, logic synthesis.
Słowa kluczowe: zmodyfikowana arytmetyka rozproszona, filtry SOI, heterogeniczne struktury programowalna, synteza logiczna.

Introduction

Digital Signal Processing (DSP), thanks to explosive growth in wired and wireless networks and in multimedia, represents one of the hottest areas in electronics. The applications of DSP continue to expand, driven by trends such as the increased use of video and still images and the demand for increasingly reconfigurable systems such as Software Defined Radio (SDR). Many of these applications combine the need for significant DSP processing with cost sensitivity, creating demand for high-performance, low-cost DSP solutions.

In recent years digital filters have been recognized as primary digital signal processing operation. With advances in digital technology they are rapidly replacing analogue filters, which were implemented with RLC components. Digital filters are used to modify attributes of signal in the time or frequency domain trough the process called linear convolution. They are typically implemented as multiply-accumulate (MAC) algorithms with use of special DSP devices [1, 2, 3]. Such devices are based on the concept of RISC processors with an architecture consisting of fast array multipliers. General-purpose DSP chips combine efficient implementations of these functions with a general-purpose microprocessor. The number of multipliers is generally in the range of one to four, and the microprocessor will sequence data to pass it through the multiply and other functions, storing intermediate results in memory or accumulators. Performance is increased primarily by increasing the clock speed used for multiplication. By using pipeline architecture the speed of such implementation is limited by the speed of array multiplier. Typical clock speeds run from tens of MHz to 1GHz. Performance, as measured by millions of Multiply And Accumulate (MAC) operations per second, typically ranges from 10 to 4000.

Field Programmable Gate Arrays (FPGAs), with their newly acquired digital signal processing capabilities, are now expanding their roles to help offload computationally intensive digital signal processing functions from the processor. Progress in development of programmable architectures observed in recent years resulted in digital devices that allow building very complex digital circuits and systems at relatively low cost in a single programmable structure. Programmable technology, however, provides possibility to increase the performance of digital system by exploitation of parallelisms of implemented algorithms. This technology allows also application of special techniques such as distributed arithmetic (DA) [4, 5].

Distributed arithmetic is an important technique to implement digital signal processing (DSP) functions in FPGAs [1]. It provides an approach for multiplier-less implementation of DSP systems, since it is an algorithm that can perform multiplication with use of lookup table (LUT) that stores the precomputed values and can be read out easily, which makes DA-based computation well suited for FPGA realization, because the LUT is the basic component of FPGA. DA specifically targets the sum of products computation that is found in many of the important DSP filtering and frequency transforming functions.

DA concept proves to be a powerful technique for implementing MAC unit as a multiplierless algorithm. The efficiency of implementations based on this concept and targeted FPGAs strongly depend on implementation of DA-LUT. These blocks have to be efficiently mapped onto FPGA's logic resources. The major disadvantage of DA technique is that the size of DA-LUT increases exponentially with the length of input. Several efforts have been made to reduce the DA-LUT size for efficient realization of DA-based designs. In [2] to use offset-binary coding is proposed to reduce the DA-LUT size by a factor of 2. Recently, a new DA-LUT architecture for high-speed high-order has been introduced in [6], where the major disadvantage of the FIR filters is vanished by using carry lookahead adder and the tri-state buffer. On the other side, some structures are introduced for efficient realization of FIR filter. Recently, novel one- and two-dimensional systolic structures are designed for computation of circular convolution using DA [7], where the structures involve significantly less area-delay complexity compared with the other existing DA-based structures for circular convolution. In [8] modified DA architecture is used to obtain an area-time-power-efficient implementation of FIR filter in FPGA.

With rapidly growing 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. 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 [9, 10]. 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. The method presented in [11] is designed specifically to implement FIR filters using the concept of distributed arithmetic. In [12] advanced synthesis method based on functional decomposition was proposed that utilizes embedded memory block as large LUTs.

In [13] the modified DA concept was presented that allows for very efficient implementation of DA-LUT blocks in heterogeneous programmable structures. The method introduced there may have great impact on performance of DSP modules based on DA and targeted at modern FPGA architectures.

This paper demonstrates the application of the modified distributed arithmetic concept in FIR filter implementation that is targeted at modern FPGA with heterogeneous structure. Presented results prove that this method allows utilizing heterogeneous resources of programmable structures very efficiently. Comparison of implementation results obtained with modified DA and results of specialized CAD tool demonstrates the superiority of new approach.

Preliminary information

Architectures of modern FPGA

The technological advancements in Field Programmable Gate Arrays in the past decade have opened new paths 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 advancements in microelectronics in the past decade have 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 enhancements 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 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. These 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.

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.

Modified distributed arithmetic

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

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

where: $c[n]$ – constants, $x[n]$ – variables

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

$$x[n]=\sum_{b=0}^{B-1}x_b[n]\times 2^b \tag{2}$$

where: $x_b[n]\in[0,1]$, the whole convolution sum can be described as shown in (3).

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

Since *c*[*n*] are constant the second sum in (3) can be implemented as a mapping $f(x_b)$, where $x_b=(x_b[0],x_b[1],...,x_b[N-1])$. 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 schematic representation of such implementation is shown in Fig. 1, where the mapping f is presented as a lookup table (DA-LUT) that includes all the possible linear combinations of the coefficients and the bits of the incoming data samples [1].

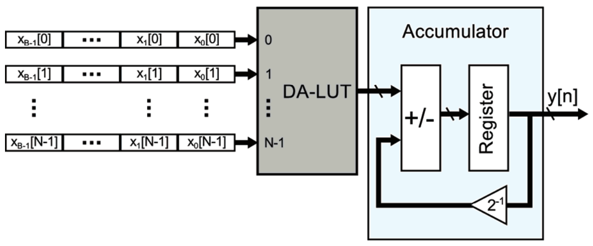

Fig.1. DA architecture with look-up table (LUT)

DA concept proves to be a powerful technique for implementing MAC unit as a multiplierless algorithm through the use of combinational DA-LUT to store the precomputed values of $f(x_b)$ (3). The efficiency of implementations based on this concept strongly depends on implementation of DA-LUT representing the function $f(x_b)$. These blocks have to be efficiently mapped onto FPGA's logic resources. Since heterogeneous logic blocks are often used in the modern FPGA architectures (Xillinx Virtex-5 and Altera Stratix III series) construction of efficient mapping algorithm is very challenging task. Modern heterogeneous structures are composed of programmable logic elements and embedded memory blocks. However all these resources can be referred to as LUTs of various sizes.

In [13] a modification of distributed arithmetic concept was proposed that allows decomposing DA-LUT into LUTs of sizes available in given FPGA architecture

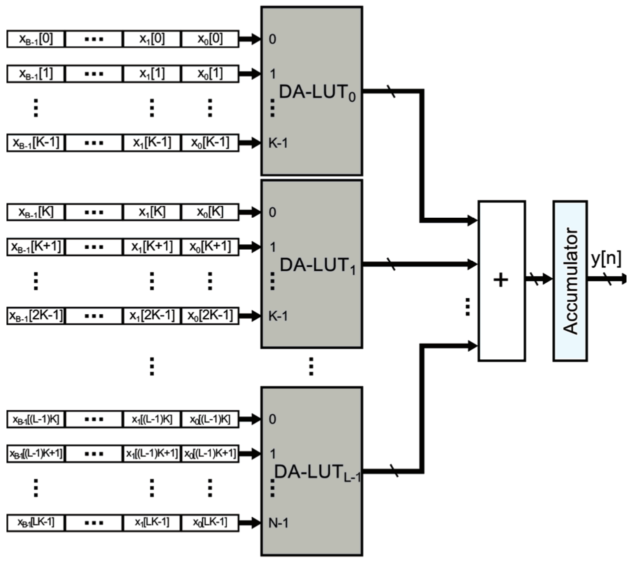

Fig.2. DA architecture with partitioned look-up table (LUT)

Let assume that available logic resources of specific FPGA are composed of LUT-like blocks grouped in *M* groups according to their sizes $Lin_i \times Lout_i$. Lin_i is the number of inputs and $Lout_i$ is the number of outputs of LUT blocks belonging to group i. To implement DA-LUT block using LUTs of specific size a modification of (4) can be used. Suppose the N coefficients have been grouped in L sets each containing K_i coefficients, 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. As can be noticed groups may have various sizes. Let denote the number of i-th coefficient form j-th set as n_i^j. Then computing the sum of products can be described as (5).

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

Function $f(x_b)$ has been decomposed into L functions $f_l(x^l_b)$. The sum is partitioned into L independent DA-LUTs. This allows implementing DA architecture similarly to this shown in Fig. 2. The number of inputs of l-th block ($DA\text{-}LUT_l$) is equal to K_l, while the number of outputs is equal to ($\lceil log_2K_l \rceil + q$), where q is the number of bits used to represent coefficients $c[n]$.

Corollary 1. To implement l-th block ($DA\text{-}LUT_l$) of modified distributed arithmetic described by (5) it is needed at most $\lceil(\lceil log_2K_l\rceil + q) / Lout_i\rceil$ LUTs from group i for which Lin_i is equal to K_l.

Modification (5) allows to adjust the number of inputs of DA-LUTs of structure presented on Fig. 2 to the size of available logic resources. Then the mapping of each DA-LUT into logic elements is straightforward (Corollary 1). However in some cases not all logic elements used for mapping are utilized in 100%. It happens, when the number of outputs of given DA-LUT is not the multiple of number of outputs of logic element used for mapping.

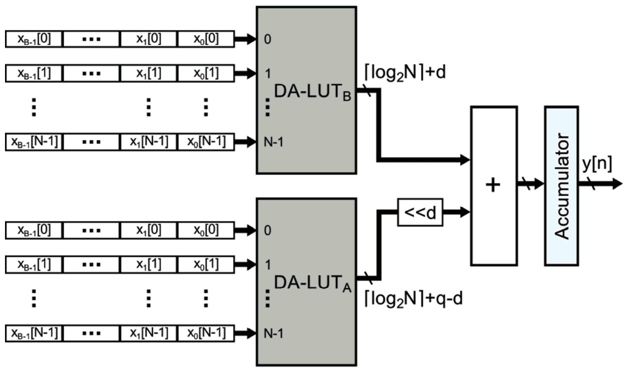

Fig.3. DA architecture with decomposed look-up table (LUT)

This issue can be addressed by application of another modification of DA concept. Let represent all coefficients of sum of product from (3) as follows:

$$c[n] = 2^d \times c_A[n] + c_B[n] \tag{6}$$

where $c_B[n]$ represents d least significant bits of coefficient $c[n]$ and $c_A[n]$ represents $q - d$ most significant bits of coefficient $c[n]$.

Then (3) can be expressed in following way:

$$\begin{aligned} y[n] &= \sum_{b=0}^{B-1} 2^b \times \sum_{k=0}^{N-1} x_b[k]\times(2^d\times c_A[k] + c_B[k]) = \\ &= \sum_{b=0}^{B-1} 2^b \times (2^d \times \sum_{k=0}^{N-1} x_b[k]\times c_A[k] + \sum_{k=0}^{N-1} x_b[k]\times c_B[k]) = \\ &= \sum_{b=0}^{B-1} 2^b \times (2^d \times f_A(x_b) + f_B(x_b)). \end{aligned} \tag{7}$$

Function $f(x_b)$ has been decomposed into two functions $f_A(x_b)$ and $f_B(x_b)$. The sum is partitioned into two independent DA-LUTs, one with N inputs and $\lceil log_2N\rceil + q - d$ outputs and the second with N inputs and $\lceil log_2N\rceil + d$ outputs. This allows implementing DA architecture as shown in Fig. 3. At the cost of additional adders the number of outputs of DA-LUTs can be reduced.

Application of this concept allows to adjust the number of outputs of DA-LUTs. Application of both described techniques makes it possible to map DA-LUT into heterogeneous architectures of modern FPGAs very efficiently.

FIR filter implementation tools based on modified distributed concept

We developed the application that supports decomposition process of FIR filter DA-LUT according to the methodology described in previous section. Software also generates decomposed FIR filter structure description in VHDL. On the current stage of development, tool supports only Altera Stratix FPGAs and our future goal is to extend support to Xilinx Virtex architecture.

Hierarchical decomposition

The application allows performing DA-LUT decomposition process in stages controlled manually by the designer. Each step is registered and visualized by the tool in form of tree-like structure. During the process software allows the designer to perform two operations that correspond to two types of DA-LUT decompositions described in previous section:
- filter coefficients grouping (5),
- splitting filter coefficients into groups of more and less significant bits (7).

Those operations create DA-LUT tables, which may be further recursively decomposed in similar way. Leafs of decomposition tree generated in this way may be interpreted as independent filters. Composition of the tree leafs gives the base filter. Filter coefficients are interpreted as values with variable bit widths. Depending on the size of other coefficients from the group it may be extended with zeros or with sign bit on the MSB position. All coefficients in the group are extended to the size of the largest one in the group. Grouping together negative and positive coefficients results in necessity to add extra sign bit to the positive ones. Coefficients with value equal zero are removed from DA-LUTs to minimize number of table inputs.

The basis of grouping operation is splitting the larger group of coefficients into smaller groups. As the result DA-LUT tables are created, that have less inputs, and that may fit better into target FPGA architecture. Appropriate grouping allows controlling table size (number of bits of the output value), because it is determined by the sum of grouped coefficient values. That provides a means to control the quantity of occupied FPGA resources.

Splitting coefficient's binary representation into groups of more significant and less significant bits enables elimination from DA-LUT series of bits equal to zero on the LSB positions as well as the areas of zero bits that are between MSB and LSB. Additionally this operation may be used to fit DA-LUT table, which number of outputs exceeds the number of outputs of FPGA building bock by cutting out MSBs or LSBs that may be put into other FPGA building blocks. The group of the MSB obtained by splitting operation gets additional shift value (Fig. 3). This attribute propagates to next stages of decomposition process and is used to properly construct final adder.

Aforementioned operations may be freely applied and mixed with each other to form a decomposition tree. The software supports design process decisions providing the designer necessary information about the DA-LUT tables size in terms of a FPGA logic cell utilization.

As a result of applied operations, the N-input DA-LUT table is decomposed into the structure of several k-inputs tables and the adder tree. The decomposition process significantly reduces DA-LUT table size.

Filter structure

The presented software automatically generates fully parallel DA FIR filter structure and its description in VHDL. The filter is parameterized with the input sample size and its binary format (signed, unsigned). The filter with parallel structure processes one input sample per cycle. It is achieved by multiple instantiation of DA-LUTs according to the number of bits of input samples. Each input bit is processed by DA-LUT table corresponding to its position in binary representation of the sample. FIR filter modules generated by the described software are shown at Fig. 4.

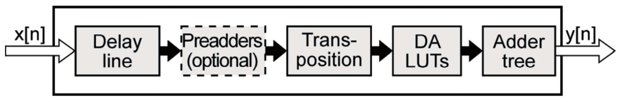

Fig.4. Parallel DA FIR filter structure

Input samples are registered and in every clock cycle are passed to next register in the delay line. If filter is symmetric, the pre-adder is implemented, that reduces by two number of processed data after delay line. The transposition module performs reorganization of data in words denoted as x_b, composed of bits that have similar weights, but come from different samples. Rearranged data constitute inputs to DA-LUTs. Outputs of DA-LUTs $y_b[n]$ are added up together including bit shifts in adder tree structure to form filter output sample $y[n]$.

Each module and every level of adder tree have pipeline registers on its outputs. It does not increase the number of utilized ALM modules, and reduces the critical path to one level of combinational logic. The delay form input to output ports introduced by the pipeline may be computed with formula (8):

$$delay = 2 + P + \lceil \log_3(D) \rceil + \lceil \log_3(B+P) \rceil \qquad (8)$$

where: P - 1 if filter is symmetric 0 otherwise, D - the number of DA-LUT elements (depending on decomposition process), B - the number of bits of input sample. Constant value 2 stands for input delay line and DA-LUT elements registers. The terms that include logarithm base 3 stands for the depth of ternary adder trees.

Input data format determinate the way of performing arithmetic operations in filter modules. Unsigned samples are extended with zero in pre-adder. In case of signed data each sample is extended with MSB in pre-adder and in final adder tree the outputs from DA-LUT corresponding to most significant bit is subtracted from the other.

Preadder for symmetric filters

If filter is symmetric, preadder may be used to sum up samples corresponding to similar filter coefficient values at delay stages. Application of preadder allows halving the number of DA-LUT inputs, but it requires instantiation of additional DA-LUT. The module utilizes *N*/2 adders, where *I* is the filter order. Figure 5 presents the scheme of data flow through the tapped delay line, preadder and transposition modules.
Assuming the N-th order filter, processing B bits samples, where $n \in [0, \ldots, N\text{-}1]$ denotes delay of processed sample, and $b \in [0, \ldots, B\text{-}1]$ the position of the bit in processing sample, with symmetric coefficients $c[n]{=}{=}c[n']$, where:

$$n' = N - 1 - n \qquad (9)$$

If the b-th bit of both $x[n]$ and $x[n']$ is 1, then b-th DA-LUT output includes values $c[n]$ and $c[n']$ which is equal to

2×$c[n]$. Similar effect is obtained when stimulating only n-th input of b+1-th DA-LUT which outputs has doubled weight. When adding samples $x[n]$ and $x[n']$ where $n \in [0, \ldots, N/2-1]$ the output of b-th bit adder is connected with input of b-th DA-LUT, the carry bit is responsible to pass information to b+1-th DA-LUT. The need of using additional DA-LUT may be explain performing analogical argumentation for B-th bit of input sample. Odd-order filters do not require preadder for central coefficient.

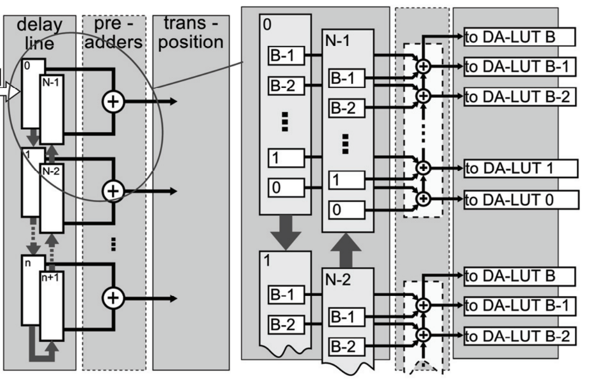

Fig.5. Data flow through delay line, preadders and transposition modules in details

DA-LUTs

Number of instantiated DA-LUT corresponds to bit width of input samples. Each module has the same structure. The decomposed DA-LUT contains several smaller LUTs, whose outputs are processed in ternary adder tree structure, as presented in Figure 6. Ternary adder tree are very efficiently implemented in for modern Stratix and Virtex FPGA architectures, and are more effective than binary trees. LUT tables are mapped to ALMs or embedded memories. To explicitly assign tables to specific FPGA atoms, the WYSWIG library component were used. Each row of LUT, which may be fit into ALUT is described as independent object instantiated from WYSWIG.

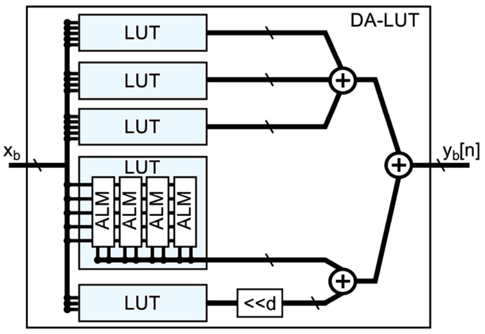

Fig.6. Internal structure of DA-LUT

The adder tree is composed of three and two operand adders and paths propagating signals between tree levels. Signals entering tree inputs are arranged in non-decreasing operands bit-widths order. If tree level is incomplete and it is not possible to carry out addition of all operands with ternary adder, in incomplete branch always the operand with largest number of bits is placed. In implementation described in this paper every tree level has pipeline registers, but in general the structure may be freely pipelined.

Output adder tree

The final adder tree sums up outputs from DA-LUTs with appropriate weights. The weights are bit shifts – the power of 2. The summed up values have similar bit-widths. In the current software version tree size depends only on the bit precision of input samples. Operands are grouped according to their weights. The tree forms regular structure. Every level implements additions with identical shifts. With increasing number of tree level, the shifts between operands grow three times e.g. 1st level 1 bit shift, 2nd level 3 bit shift, 3rd level 9 bit shift as is presented in Figure 8. If tree is incomplete, in incomplete branch are operands with the largest weight.

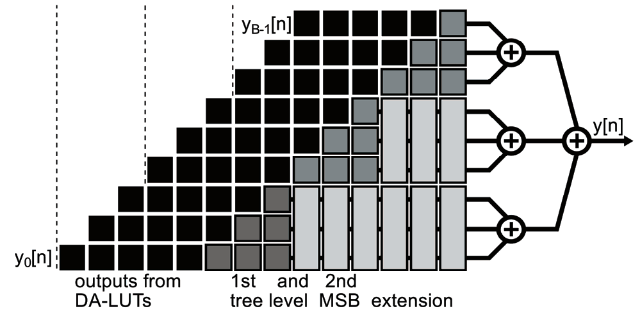

Fig.7. Data alignment in output ternary adder tree

The structure is implemented with three and two operand adders and paths propagating values between tree levels. In general structure may be freely pipelined, but in current implementation to gain operating frequency every tree level is pipeline step.

Results

To evaluate the efficiency of proposed tool a number of decompositions of different FIR filters was carried out. For test real filters were used, which coefficients are easily to be generated or found in literature. We used filters *l1*, *l2*, *l3* from [14] and *s1*, *s2* from [15], that are 121th, 63rd, 36th, 25th and 60th order respectively. Additionally we have generated number of filters: *db40h*, *db40g*, *sym30h*, *sym30g* and *lp100*. The abbreviation are low-pass and high-pass 80-taps analysis Daubechies wavelet filters [16], low and high-pass 60-taps analysis symlet filters [4] and 100th order low-pass filter with cutoff at 0.3 sampling frequency, designed using Parks-McClellan algorithm [17]. The coefficients were generated using MATLAB software[18] (Fig. 8). Generated coefficients were represented as double floating point numbers, thus had to be normalized in a way that sum of absolute values of filter coefficients could fit into 16 bits signed integers. For each of aforementioned coefficient sets we generated 4 filters able to process 8 bit signed or unsigned samples or 16 bit signed or unsigned samples (superscripts *_08s*, *_08u*, *_16s*, *_16u* are used to indicate the configuration).

```
>> lp100 = firpm(100,[0 0.2 0.3 1],[1 1 0 0]);
>> lp100 = floor((lp100/(sum(abs(lp100)))*2^16);
>> [db40_g,db40_h] = wfilters('db40');
>> db40_g = floor((db40_g/(sum(abs(db40_g)))*2^16);
>> db40_h = floor((db40_h/(sum(abs(db40_h)))*2^16);
>> [sym30_g,sym30_h] = wfilters('sym30');
>> sym30_g = floor((sym30_g/(sum(abs(sym30_g)))*2^16);
>> sym30_h = floor((sym30_h/(sum(abs(sym30_h)))*2^16);
```

Fig.8. MATLAB scrip generating *db40** *sym30** and *lp100* filter coefficients

The syntheses were performed in Quartus 10.1 SP1. The target FPGA platform was Altera Stratix III EP3SL50F484C2 [19]. The optimization method was set to *Balanced*, project parameters were default, except turned off automatic recognition of RAM, ROM memories and shift registers and automatic RAM balancing. Input and output ports were set as *Virtual* (implemented module is placed in the middle of the FPGA and behaves as an internal part of a larger system).

As a reference we used filters generated in commercial software Altera FIR Compiler 10.1 [19]. In reference project the pipeline level parameter was set to 2, (it gives the best ratio of utilized area vs. operational frequency that we consider as quality indicator). As top level module we took the entities describing only FIR filter structure (Avalon ST wrapper was not implemented). Similar to aforementioned implementations top level ports were set as *Virtual*. The optimization parameter was set to *Balanced*, and all project options were default.

Table 1 presents summary of implementation results for full set of analyzed configurations for selected filters (*db40* – asymmetric filter, *lp100* – high order symmetric filter, *s1* – low order symmetric filters). The intention was to show the trend of improvement in the implementation quality with growth of input samples bit-width. For other filters Table 1 presents only implementation results for signed 16 bit width data, to show the difference in quality of results between proposed solution and reference and still be legible.

Table 1. The results of DA FIR filters implementation using proposed method and FIR Compiler Software

Filter name	Proposed Method (PM)			FIR Compiler (FC)			PM/FC QI RATE
	ALM	FREQ	QI	ALM	FREQ	QI	
db40g_08s	1312	473	2.77	1564	463	3.38	0.82
db40g_08u	1314	495	2.65	1565	470	3.33	0.80
db40g_16s	2638	493	5.35	3262	399	8.18	0.65
db40g_16u	2640	481	5.49	3297	398	8.28	0.66
lp100_08s	1533	415	3.69	1698	465	3.65	1.01
lp100_08u	1524	507	3.01	1700	477	3.56	0.84
lp100_16s	2931	465	6.30	3561	416	8.56	0.74
lp100_16u	2931	472	6.21	3587	378	9.49	0.65
s1_08s	333	467	0.71	384	530	0.72	0.98
s1_08u	325	538	0.60	391	576	0.68	0.89
s1_16s	644	518	1.24	808	498	1.62	0.77
s1_16u	644	533	1.21	816	479	1.70	0.71
l1_16u	3732	437	8.54	4609	373	12.36	0.69
l2_16u	1745	494	3.53	2102	409	5.14	0.69
l3_16u	1006	500	2.01	1127	484	2.33	0.86
s2_16u	1851	464	3.99	2311	375	6.16	0.65
sym30g_16u	2151	472	4.56	2579	412	6.26	0.73
sym30h_16u	2137	469	4.56	2583	388	6.66	0.68
db40h_16u	2638	461	5.72	3284	407	8.07	0.71
Average rate of implementation quality indicator of proposed method vs. reference.							0.76

In Table 1 the number of utilized adaptive logic modules (*ALM*), operating frequency (*FREQ*) are presented. As Quality Indicator (*QI*), after Mayer-Base et al. [20], the ratio of utilized logic to operating frequency was used. The smaller *QI* is the implementation is better. The last column presents the ratio of quality indicator of proposed method to reference software. The results of implementation of FIR filter based on distributed arithmetic obtained using only Quartus II system was not presented in the table, because even for low order filters synthesis algorithms used by this software are inefficient [13]. Additionally for most filters used in comparison, direct generation of DA-LUT description in HDL is impossible because of large number of coefficients, which implies large number of DA-LUT inputs. For presented set of FIR filters, the proposed method gives average quality indicator about 25% better than Altera FIR Compiler. Modified distributed arithmetic method in every case allows less FPGA area utilization and in most of the cases offers higher operating frequencies. The best case was observed for example *lp100_16u* where quality indicator was 35% smaller than the reference

Mayer-Base et al. [20] presented comparison of filters *l2-l3* and *s1-s2* implementations on Cyclone II FPGA with DA method using FIR Compiler software (DA-FC) vs. RAG-n algorithm [21]. These results cannot be directly compared with ones from Table 1, because of very different target FPGA architecture. However the ratios between RAG-n and DA-FC implementations quality indicator (0.60, 0.70, 0.62, 0.49 for *s1*, *s2*, *l2*, *l3* respectively) are quite similar to presented PM to FC ratios. Additionally, compared filter coefficients are relatively simple combination of powers of 2, which is advantage for RAG-n algorithm. The difference between quality indicator of DA-FC and RAG-n is decreasing when the filter order grows, while for proposed solution an inverse relation may be observed. Thus we may assume that the proposed method may be alternative for RAG-n algorithm and it will be considered in our future research.

Summary

Distributed Arithmetic is important technique used for implementing digital filters in FPGAs. The vital issue for quality of filter implementation using DA concept is proper mapping of DA-LUT into logic elements of FPGA device. In the article we have discussed details of the modified distributed arithmetic concept that is novel technique for efficient decomposition of DA-LUT targeted for heterogeneous FPGA architectures. We have discussed the basic features of software application designed to support this type of decomposition and presented the details of parallel pipelined DA FIR filter architecture automatically generated by this application. The quality of MDA concept was proven by the series of implementation results obtained for both symmetric and asymmetric FIR filters with different number of taps, ranging from 25 to 121. Presented decomposition method and parallel DA FIR filter architecture compared to typical DA FIR implementations gives in average 25% better results. For most tested filters proposed method produces circuits requiring less FPGA resources and operating at higher frequencies.

Acknowledgments

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

REFERENCES

[1] Meyer-Baese U., Digital Signal Processing with Field Programmable Gate Arrays, Second Edition, Springer Verlag, Berlin (2004)

[2] Parhi K.K., VLSI Digital Signal Processing Systems: Design and Implementation, Wiley, New York, (1999)

[3] Rawski M., Łuba T., Jachna T., Tomaszewicz P., The influence of functional decomposition on modern digital design process, *Design of Embedded Control Systems*,(2005),193–203.

[4] Croisier A., Esteban D., Levilion M., Rizo V., Digital Filter for PCM Encoded Signals, US Patent No. 3777130, (1973)

[5] Peled A., Liu B., A new hardware realization of digital filters, *Acoustics, Speech and Signal Processing, IEEE Transactions on,* 22, (1974), nr 6, 456–462.

[6] Eshtawie M. A. M., Othman M., On-line DA-LUT architecture for high- speed high-order digital FIR filters, *Communication systems, 2006. ICCS 2006. 10th IEEE Singapore International Conference on*,(2006), 1-5

[7] Meher, P.K., Hardware-Efficient Systolization of DA-Based Calculation of Finite Digital Convolution, *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 53, (2006) nr 8, 707-711

[8] Xie J., Heand J., Tan G., FPGA realization of FIR filters for high-speed and medium-speed by using modified distributed arithmetic architectures, *Microelectronics Journal*, 41, (2010), nr 6, 365-370

[9] Cong J.,Yan K., Synthesis for FPGAs with embedded memory blocks, *FPGA '00 Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate array*, (2000), 75-82.

[10] Krishnamoorthy S., Tessier R., Technology mapping algorithms for hybrid FPGAs containing lookup tables and PLAs, *CAD of Integrated Circuits and Systems IEEE Trans. on*, 22, (2003), nr 5, 545-559

[11] Sasao T., Iguchi Y., Suzuki T., On LUT cascade realizations of FIR filters, *Digital System Design, 2005. Proceedings. 8th Euromicro Conference on*, (2005), 467-475

[12] Rawski M., Tomaszewicz P., Selvaraj H., Łuba T., Efficient Implementation of digital filters with use of advanced synthesis methods targeted FPGA architectures*, Digital System Design, 2005. Proceedings. 8th Euromicro Conference on*, (2005) 460-466

[13] Rawski M., Modified Distributed Arithmetic Concept for Implementations Targeted at Heterogeneous FPGAs, *International Journal of Electronics and Telecommunications*, 56, (2010) nr 4, 345-350

[14] Yong L., Parker S., Discrete coefficient FIR digital filter design based upon an LMS criteria, *Circuits and Systems, IEEE Transactions on*, 30, (1983), nr 10, 723-739

[15] Samueli H., An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients, *Circuits and Systems, IEEE Transactions on*, 36, (1989) nr 7, 1044-1047

[16] Daubechies I., Ten Lectures on Wavelets, SIAM, Philadelphia, (1992)

[17] Programs for Digital Signal Processing, IEEE Press, New York, (1979), Algorithm 5.1.

[18] Mathworks website http://www.mathworks.com, (2012)

[19] Altera website, http://www.altera.com (2012)

[20] Meyer-Baese U., Chen J., Hong Hang C., Dempster A.G., A Comparison of Pipelined RAG-n and DA FPGA-based Multiplierless Filters, *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on,* 2005, 1555-1558

[21] Dempster A.G., Macleod, M.D., Use of minimum-adder multiplier blocks in FIR digital filters, *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 42, (1995), nr 9, 569-577

***Authors**: mgr inż. Michał Staworko, Politechnika Warszawska, Instytut Telekomunikacji, ul. Nowowiejska 15/19, 00-665 Warszawa, E-mail: M.Staworko@elka.pw.edu.pl; dr inż. Mariusz Rawski, Politechnika Warszawska, Instytut Telekomuniakcji, ul. Nowowiejska 15/19, 00-665 Warszawa, E-mail: Rawski@tele.pw.edu.pl;*