FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 15 Dec. 2002, 451-464

FPGA Implementation of Folded FIR Filter Architecture with Changeable Folding Factor

Ivan Milentijević, Vladimir Ćirić, Teufik Tokić and Oliver Vojinović

Abstract: The application of folding technique to the bit–plane systolic FIR filter architecture that enables the implementation of changeable folding factor on to the fixed size array is described in this paper. The bit-level transformation of the original data flow graph (DFG), for the bit–plane architecture, that provides the successful application of the folding technique with changeable folding is presented at transfer function level. The mathematical path that describes the transformation is given, and implications at the DFG level are discussed. Changeable folding sets are involved with aim to increase the throughput of the folded system reducing the folding factor according to the coefficient length. The folded FIR filter architecture is described in VHDL as a parameterized FIR filtering core and implemented in FPGA technology. The design "tradeoffs" relating on the occupation of the chip resources and achieved throughputs are presented.

Keywords: Systolic arrays, FIR filtering, folding technique

1 Introduction

Considerable attention has been placed on the implementation of signal processing algorithms in VLSI, ranging from full custom VLSI to general-purpose digital signal processors. A variety of approaches to custom implementation of Finite Impulse Response (FIR) filters have been pursued. In order to attain high performance, parallel implementation strategies such as systolic methods, have been applied [1] - [4]. Thus, due to their geometrical regularity, they are suitable for VLSI implementations, either as stand-alone

Manuscript received September 2, 2002
The authors are with Faculty of Electronic Engineering, University of Niš, Beogradska 14, PO Box 73, 18000 Niš, Yugoslavia (-mail: milentijevic@elfak.ni.ac.yu).

modules or as a part of complex digital data path. However, the design of such a processor inevitably faces the limitation of VLSI area available. Excessive use of VLSI area for such a processor would be prohibited by both cost and performance. Under a restricted VLSI area the design of such a processor often introduces a conflict between its versatility and computation speed [5].

Advances in Field Programmable Gate Array (FPGA) technology have enabled FPGAs to be used in a variety of applications. In particular, FPGAs prove particularly useful in data path designs, where the regular structure of the array can be utilized effectively. The programmability of FPGAs adds flexibility not available in custom approaches, while retaining relatively high system clock rates. Furthermore, the FPGA technology is ideal for rapid prototyping [6].

It is well known that performances and cost of any digital circuit depend on circuit design style. Therefore, creating a given architecture, to establish optimal area-time-power tradeoff, a careful choice of circuit design style to use is necessary. In synthesizing DSP architectures, it is important to minimize the silicon area of the integrated circuits, which is achieved by reducing the number of functional units (such as multipliers and adders), registers, multiplexers, and interconnection wires. The folding transformation is used to systematically determine the control circuits in DSP architectures where multiple algorithm operations are time multiplexed to a single functional unit [7]. By executing multiple algorithm operations on a single functional unit, the number of functional units in the implementation is reduced, resulting in integrated circuit with low silicon area [8].

As a starting architecture for the synthesis of the folded bit–plane FIR filter architecture with changeable folding sets we use well-known bit–plane architecture (BPA). The BPA is highly regular architecture, which allows extensive pipelining, regular layout, high computational throughput, truncation of Least Significant Bits (LSBs) of intermediate results without any loss of accuracy, and programmability of coefficients [9, 10].

The goal of this paper is to present the application of folding technique to the bit–plane systolic FIR filter architecture that enables the implementation of changeable folding sets onto the fixed size array. The involving of changeable folding sets and changing of the folding factor are aimed to the increasing of versatility of bit plane-arrays. We apply the folding technique with goal to find suitable area-time tradeoff for bit–plane architecture keeping all desirable features of the source architecture. An additional goal is the providing of the wider application area for folded bit–plane architec-

ture. With aim to illustrate the application of folding technique to the BPA and to highlight performances of the derived folded architecture we have implemented both the BPA and the folded BPA (FBPA) in FPGA technology. The design "tradeoffs" relating on occupation of the chip resources and achieved throughputs are presented.

The paper is organized as follows: section 2. describes the BPA as a basic architecture; section 3. contains basic principles of folding technique; in section 4. we give the transformation of the original DFG for the BPA that enables the application of folding technique; section 5. describes the involving of changeable folding factor, section 6. is devoted to the FPGA implementation of the folded FIR filter architecture, while in section 7 concluding remarks are given.

2 Bit-plane FIR filter architecture

Output words $\{y_i\}$ FIR filter are computed as

$$y_i = c_0 x_i + c_1 x_{i-1} + ... + c_{k-1} x_{i-k+1}, \tag{1}$$

where c_0, c_1, ..., c_{k-1} are coefficients while $\{x_i\}$ are input words.

The bit–plane architecture (BPA) is semi-systolic architecture which provides regular connections with extensive pipelining and high computational throughput. The BPA is basic architecture for synthesis of folded architecture (FA), so we give a brief description of the BPA. In order to explain the BPA following notation is adopted:

m – coefficient word length,

k – number of coefficients (c_0, c_1, ..., c_{k-1}), and

c_i^j – bit of coefficient c_i (with weight 2^j).

The BPA is obtained by resorting of the partial products of different multipliers as it is shown in Fig. 1.

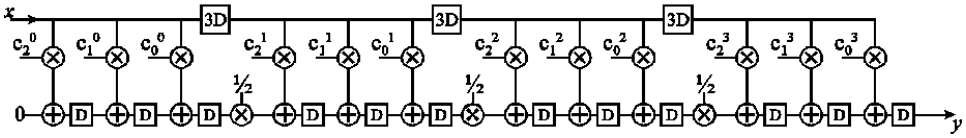

Fig. 1. The DFG (Data flow graph) for the BPA with $k = 3$ and $m = 4$

With fine-grained pipelining, the splitted parts of the multiplications become input word times $1-b$ coefficient multiplications, the partial products. These are just logical AND function between the input word and coefficient bit. In the first bit–plane the least significant partial products of all coefficients are computed and accumulated (Fig. 1). The output of the first bit–plane is shifted by one weight and then the second lowest significant partial products are processed in the second bit plane and so on [9, 10]. Starting bit–plane processing with the LSB's first, enables to truncate one LSB of the intermediate output signal after each bit–plane without any loss of accuracy in the more significant weights. We choose this architecture as a basis for the synthesis of the fully pipelined folded FIR filter architecture.

After this short description of the BPA as a source architecture, let us to introduce the basic elements of folding technique.

3 Basic elements of folding technique

The folding technique is introduced by K.K. Parhi and described in [7, 8]. With aim to clarify the applying of folding technique to the BPA we give a brief review of folding transformation.

The synthesis of folded data path is explained in Fig. 2(a) and Fig. 2(b). Fig. 2(a) shows an edge $U \rightarrow V$ with $w(e)$ delays, while Fig. 2(b) depicts the corresponding folded data path. The data begin at the functional unit H_u which has P_u pipelining stages, pass through

$$D_F(U \rightarrow V) = Nw(e) - P_u + v - u \tag{2}$$

delays, and are switched into the functional unit H_v at the time instances $Nl + v$, where N is the number of operations folded to a single functional unit (folding factor), while u and v are the folding orders of nodes U and V that satisfy $N - 1 \geq u, v \geq 0$.

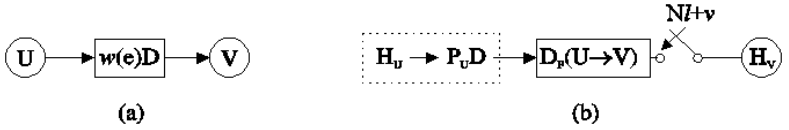

Fig. 2. The synthesis of folded data path. (a) An edge $U \rightarrow V$ with $w(e)$ delays; (b) The corresponding folded data path.

A folding set, S, is defined as an ordered set of operations, which contains N entries, executed by the same functional unit. For a folded system to be

realizable, $D_F(U \rightarrow V) \geq 0$ must hold for all of the edges in the DFG. Once valid folding sets have been assigned, retiming can be used to satisfy this property or determine that the folding sets are not feasible [7].

4 Synthesis of folded architecture

The BPA can not be transformed into the folded bit-plane architecture by direct application of folding technique. It is obvious, from Fig. 1, that multiplications of coefficients and input words can not be recognized as operations, i.e. nodes in the DFG, because the algorithm is based on the resorting of partial products. It implies that it is necessary to apply the folding technique at bit-level. Each multiplication node in the DFG, shown in Fig. 1, represents one row of basic cells (full adder and AND gate). Thus, one multiplication is distributed through the whole array. Even, if we declare forming of all partial products in row as one "row" operation we can not apply folding technique successfully, because there are delays both in input data path and summation path which are obstacles for satisfying of conditions $D_F\,(U \rightarrow V) \geq 0$.

Therefore, we suggest the transformation of source architecture that will enable the successful application of folding technique and the involving of changeable folding sets. The successful application assumes that the hardware size is reduced approximately for the factor N at the cost of time, and that the derived architecture keeps all desirable features especially fine-grain pipelining.

Let us start from the transfer function that corresponds to the DFG for the BPA ($k = 3; \quad m = 4$) shown in Fig. 1:

$$\begin{aligned} G\,(z) = \tfrac{Y(z)}{X(z)} = {} & z^{-1}(c_0^3 2^3 z^{-9} + z^{-1}(c_1^3 2^3 z^{-9} + z^{-1}(c_2^3 2^3 z^{-9} + \\ & + z^{-1}(c_3^3 2^3 z^{-9} + z^{-1}(c_0^2 2^2 z^{-6} + z^{-1}(c_1^2 2^2 z^{-6} + \\ & + z^{-1}(c_2^2 2^2 z^{-6} + z^{-1}(c_3^2 2^2 z^{-6} + z^{-1}(c_0^1 2^1 z^{-3} + \\ & + z^{-1}(c_1^1 2^1 z^{-3} + z^{-1}(c_2^1 2^1 z^{-3} + z^{-1}(c_0^0 2^0 + \\ & + z^{-1}(c_1^0 2^0 + z^{-1} c_2^0 2^0)...) \end{aligned}$$

The general form of transfer function for k taps and m bit coefficient wordlength is

$$
\begin{aligned}
G(z) &= z^{-1}(c_0^{m-1}2^{m-1}z^{-(m-1)k} + z^{-1}(c_1^{m-1}2^{m-1}z^{-(m-1)k} + ... \\
&\quad + z^{-1}(c_{k-1}^{m-1}2^{m-1}z^{-(m-1)k} + \\
&+ z^{-1}(c_0^{m-2}2^{m-2}z^{-(m-2)k} + z^{-1}(c_1^{m-2}2^{m-2}z^{-(m-2)k} + ... \\
&\quad + z^{-1}(c_{k-1}^{m-2}2^{m-2}z^{-k} + \\
&\cdots \\
&\quad + z^{-1}(c_0^0 2^0 z^0 + z^{-1}(c_1^0 2^0 z^0 + ... + z^{-1}(c_{k-1}^0 2^0 z^0)...)
\end{aligned} \tag{3}
$$

The same transfer function without brackets can be rewritten as follows

$$
\begin{aligned}
G(z) &= z^{-1}c_0^{m-1}2^{m-1}z^{-(m-1)k} + z^{-2}c_1^{m-1}2^{m-1}z^{-(m-1)k} + ... \\
&\quad + z^{-k}c_{k-1}^{m-1}2^{m-1}z^{-(m-1)k} + \\
&+ z^{-(k+1)}c_0^{m-2}2^{m-2}z^{-(m-2)k} + z^{-(k+2)}c_1^{m-2}2^{m-2}z^{-(m-2)k} + ... \\
&\quad + z^{-2k}c_{k-1}^{m-2}2^{m-2}z^{-(m-2)k} + \\
&\cdots \\
&+ z^{-[(m-1)k+1]}c_0^0 2^0 z^0 + z^{-[(m-1)k+2]}c_1^0 2^0 z^0 + ... \\
&\quad + z^{-mk}c_{k-1}^0 2^0 z^0 = \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{k-1} z^{-[(m-1-i)k+j+1]}c_j^i 2^i z^{-ik} = \sum_{i=0}^{m-1}\sum_{j=0}^{k-1} c_j^i 2^i z^{-[(m-1)k+j+1]} \\
&= z^{-[(m-1)k+1]}\sum_{i=0}^{m-1}\sum_{j=0}^{k-1} c_j^i 2^i z^{-j}.
\end{aligned}
$$

If we reorder partial products according to the z^{-j}, $G(z)$ is of the form:

$$
G(z) = z^{-[(m-1)k+1]}\sum_{j=0}^{k-1}\sum_{i=0}^{m-1} c_j^i 2^i z^{-j} = z^{-[(m-1)k+1]}\sum_{j=0}^{k-1} z^{-j}\cdot\sum_{i=0}^{m-1} c_j^i 2^i. \tag{4}
$$

The developed form of equation (4) is

$$
\begin{aligned}
G(z) &= z^0(c_0^{m-1}2^{m-1} + c_0^{m-2}2^{m-2} + ... + c_0^0 2^0 + \\
&+ z^{-1}(c_1^{m-1}2^{m-1} + c_1^{m-2}2^{m-2} + ... + c_1^0 2^0 + \\
&\cdots \\
&+ z^{-1}(c_{k-1}^{m-1}2^{m-1} + c_{k-1}^{m-2}2^{m-2} + ... + c_{k-1}^0 2^0)\ldots).
\end{aligned} \tag{5}
$$

The corresponding DFG for equation (5) is shown in Fig. 3. In other words the transformation of the transfer function from (3) to (5) is performed according to the following scenario. In order to illustrate the mathematical path that describes the transformations we follow the implications onto the DFG from Fig. 1:

- delays between planes are removed, as well as delays in the addition path; while the delays inside the plane are added;
- instead of multiplications by 1/2 in the addition path, multiplication by 2 are involved in the input data path;
- partial products are resorted, i.e. coefficient bits from each coefficient are collected separately and delays are involved in the input data path;
- removing of delays from input data path to the addition path followed by reverse ordering of coefficients.

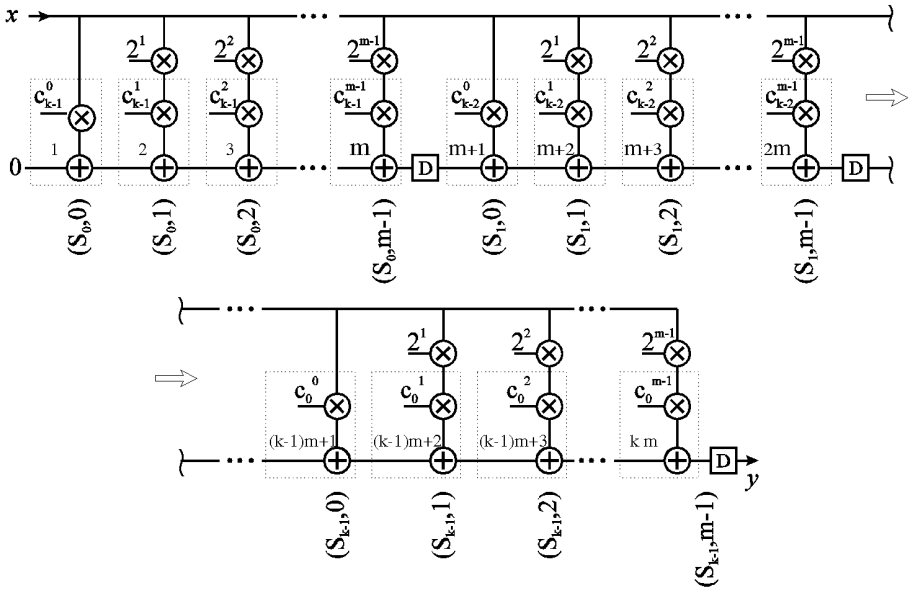

Fig. 3. Transformed DFG that enables the application of folding technique.

This paper gives the mathematical path that proves the correctness of the proposed transformation, while the generalized scenario, as 5-step synthesis procedure at the DFG level, can be found in [11].

The architecture with transformed DFG (TDFG) from Fig. 3. is impractical for implementation because of the broadcast line at input data path, but TDFG is well prepared for further application of folding technique. Besides the deriving of suitable DFG for folding the important issue is the setting of folding sets. The folding sets are formed as it is shown in Fig. 3. The operations which will be folded are denoted with dashed lines. One operation from TDFG assumes forming of partial products and the addition performed on one "row" of basic cells, (where basic cell contains AND gate and full adder).

There are k folding sets, S_0, S_1, S_2, ..., S_{k-1}, and the number of folding sets is equal to the number of taps. Each folding set contains m operations, i.e. the folding factor, N, is equal to the coefficient length, $N = m$. Thus, folded equations (2) for the determined folded sets, where $P_U = 0$ and U and V are nodes in TDFG from Fig. 3. denoted with 1, 2, ..., m, $m+1$, ..., km are

$$
\begin{aligned}
&D_F\,(1 \to 2) = m \cdot 0 - 0 + 1 - 0 = 1\\
&D_F\,(2 \to 3) = m \cdot 0 - 0 + 2 - 1 = 1\\
&\ldots\\
&D_F\,(m-1 \to m) = m \cdot 0 - 0 + (m-1) - (m-2) = 1\\
&D_F\,(m \to m+1) = m \cdot 1 - 0 + 0 - (m-1) = 1\\
&D_F\,(m+1 \to m+2) = m \cdot 0 - 0 + 1 - 0 = 1\\
&\ldots\\
&D_F\,(2m-1 \to 2m) = m \cdot 0 - 0 + (m-1) - (m-2) = 1\\
&D_F\,(2m \to 2m+1) = m \cdot 1 - 0 + 0 - (m-1) = 1\\
&D_F\,(2m+1 \to 2m+2) = m \cdot 0 - 0 + 1 - 0 = 1\\
&\ldots\\
&D_F\,(km-1 \to km) = m \cdot 0 - 0 + (m-1) - (m-2) = 1.
\end{aligned}
$$

The condition $D_F\,(U \to V) \geq 0$ is satisfied, for each pair of connected nodes (U, V), and it proves that TDFG from Fig. 3. is well prepared for folding. The obtained folded architecture (FA) with k taps is presented in Fig. 4.

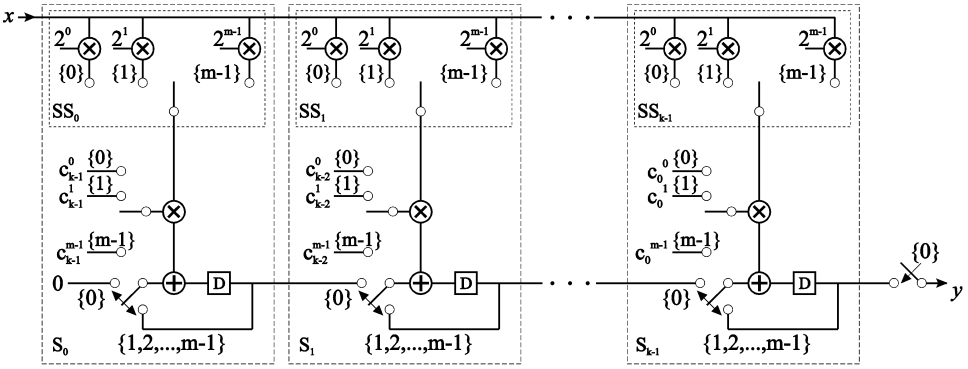

Fig. 4. Folded architecture with folding sets $S_0, S_1, ..., S_{k-1}$.

5 Changeable folding factor

In order to implement changeable folding sets, i.e. to enable the changing of folding factor we have derived folded architecture in the general form for k taps and coefficient length m, neglecting the input data width. Such application of folding technique, Fig. 4., allows the simultaneous bit-serial operation of all taps. All shift sections $(SS_0, SS_1, ..., SS_{k-1})$ can be implemented by one simple shift register. The duration of computation in each tap depends on the coefficient length m. So, the changing of coefficient length does not change the number of folding sets, but changes the number of folded operations in folding sets (Fig. 3. and Fig. 4.). The changeable folding set is the folding set where the number of folded operations can vary. Let us suppose that coefficient length can vary, $1 \leq m \leq m_1$, where m_1 denotes maximal coefficient length defined by implemented width of registers. The functional block diagram for folded architecture with changeable folding factor based on the DFG from Fig. 4. for $k = 3$, $n = 5$ and $1 \leq m \leq m_1$ is given in Fig. 5.

The operation with different folding factors is provided by:

- simple changing of control signal $ck1$ which period should be equal to m periods of the basic clock signal $ck0$;
- shift / rotate registers for coefficients joined with simple control logic which provides m -bit rotation $(1 \leq m \leq m_1)$, i.e. cyclic repetition of coefficient bits for supplying of basic cell rows.

Bearing in mind that m_1 is maximal allowable coefficient length the size of the processing array of basic cells should be $k \cdot (m_1 + n + \lceil \log_2 k \rceil)$. The corresponding vector merging adder (VMA) is attached to the k -th row of array for calculating of final result.

During the configuration the number of folding sets is fixed and equal to the number of coefficients, while the number of operations which comprise folding sets is equal to the coefficient length, m, and can vary in the range of $1 \leq m \leq m_1$, where m_1 denotes maximal coefficient length defined by the implemented width of registers. The involving of changeable folding sets allows the increasing of throughput when the filtering with smaller coefficient length is configured. The computation time linearly depends on coefficient length. Instead of extending the coefficient to the full coefficient length when the operation with coefficients with smaller length is required, the proposed architecture with changeable folding sets reduces the folding factor according to the coefficient length and increases the throughput $\frac{m_1}{m}$ times.

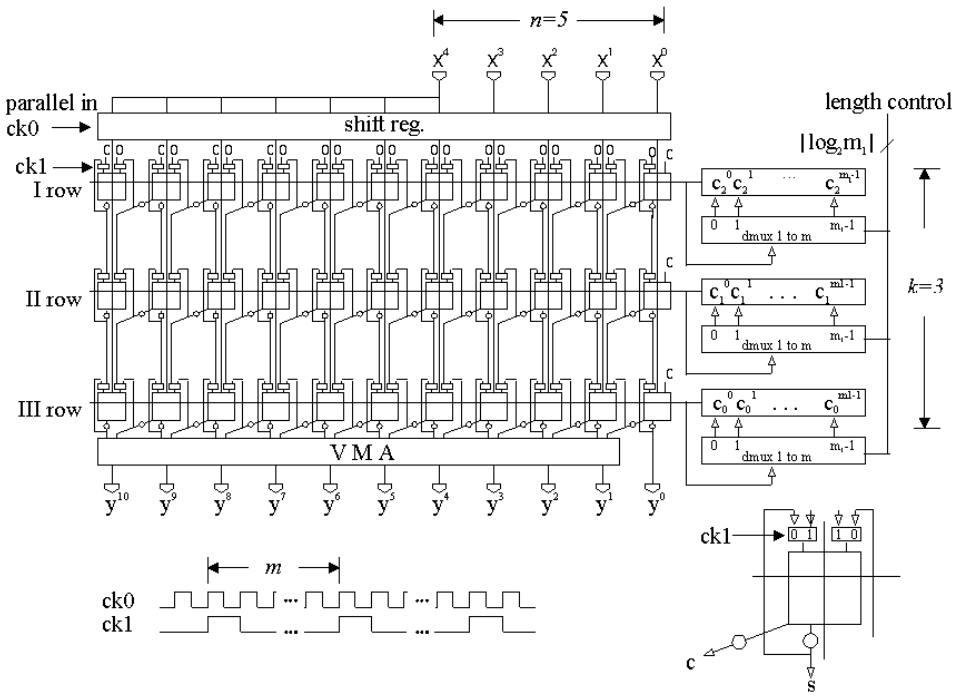

Fig. 5. Functional block diagram for folded architecture with changeable folding factor ($1 \leq N = m \leq m_1, k = 3$).

6 Implementation

In order to illustrate the method and possible design "trade offs" relating on synthesized folded architecture we have described FBPA in VHDL as parameterized FIR filtering core. The functionality of the design is proved through the logic simulation. For the sake of comparison both FBPA (synthesized architecture) and BPA (source architecture) are implemented in FPGA technology.

XC2s200 – FPGA chip that belongs to the SPARTAN II family is used. The chip contains 2352 slices where each slice is comprised of 2 Complex Logic Blocks (CLBs). As a software support for implementation we have used Xilinx Web Pack 4.2. Postimplementational report is presented in Table 1. The Table contains number of used slices over number of available slices, maximal values for pin to pin delay, MPPD, and maximal values for internal delay, MID. Both FBPA and BPA are implemented for four different

cases (k = 4, 8, 12 and 16), i.e. with different number of taps. In all cases the width of the input data word is 8, $n_x = 8$, while the maximal coefficient length is 8, too ($m_1 = 8$).

Table 1. Syntisized vs. source architecture (FBPA vs. BPA).

k		4	8	12	16
Used/Available [%]	BPA	28.6	59.5	98.9	99.9
	FBPA	9.4	18.0	27.7	36.6
MPPD [ns]	BPA	6.5	4.9	6.0	6.5
	FBPA	4.9	5.3	5.6	7.3
MID [ns]	BPA	3.1	3.6	4.6	5.8
	FBPA	4.0	4.5	5.0	6.0

The diagram that shows number of used slices over number of available slices as a fuction of implemented number of taps, k, is given in Fig. 6. The number of taps (coefficients) is equal to the number of processing elements, where one processing element is one "row" of basic cells (Fig. 5). It is possible to implement on to XC2s200 up to 16 taps, i.e. 8 planes with 16 rows of the source architecture (99.9%) of used slices while the folded architecture requires 36.6% of slices for the problem of the same size. The reduction of the processing array is performed at the cost of time.

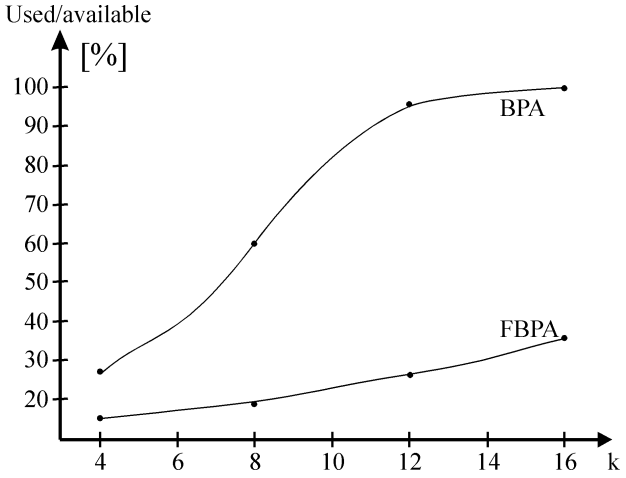

Fig. 6. Number of used slices over number of available slices as a fuction of implemented number of taps k.

For the clear picture of performaces of the folded system we present the comparison of throughtputs in Table 2. contains values for basic clock frequencies, f_{ck}, and throughputs, f_{th}, are given, for both BPA and FBPA

for cases k = 4, 8, 12 and 16 when m=8.

Maximal value for folding factor is equal to the maximal coefficient length. Thus, for implemnted architectures whith $m_1 = 8$ the throughtput is decreased 8 times.

Table 2. Basic clock frequences and throughtputs for BPA and FBPA with k = 4, 8, 12 and 16.

k		4	8	12	16
f_{ck} [MHz]	BPA	153.8	204.1	166.7	153.8
	FBPA	204.1	188.7	178.6	137.0
f_{th} [MHz]	BPA	153.8	204.1	166.7	153.8
	FBPA	25.5	23.6	22.3	17.1

Bearing in mind that the FBPA supports the changing of number of folded operations in folding sets we illustrate that feature on the implemented FBPA with 16 taps, $n_x = 8$ and $m_1 = 8$. The results are presented in Fig. 7.

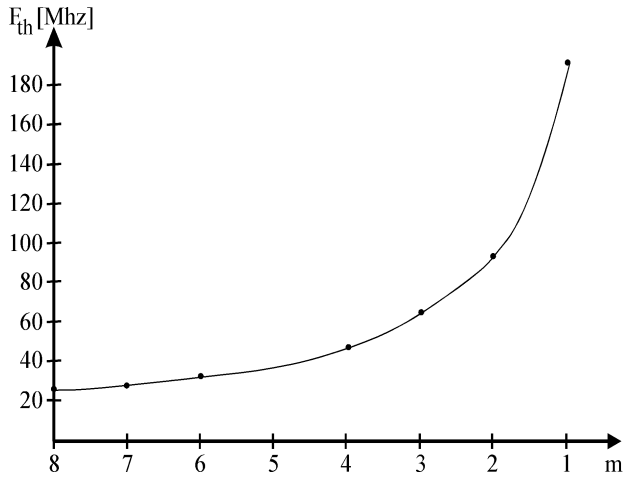

Fig. 7. The increasing of the FBPA throughput by reducing of folding factor according to the coefficient length.

When the operation with coefficients with lenght smaler then m_1 is required, insted of extending the coefficients to the full coefficient length, the implemented architecture with changable folding sets reduces the folding factor according to the required coefficient length and increases the folded system throughput $\frac{m_1}{m}$ times.

7 Conclusion

The proposed transformation of source DFG for the bit-plane architecture enables the synthesis of fully pipelined folded FIR filter architecture with changeable folding factor. The derived architecture has kept desirable features of source architecture such as extensive pipelining, high regularity, truncation of LSBs of intermediate results without any loss of accuracy.

The array is restricted for the factor m. The number of basic cells is reduced to the number of basic cells in one plane of source architecture. Also, the total number of latches corresponds to the number of latches in one plane of the BPA. The extensive pipelining in the synthesized architecture is paid by involving of two multiplexers per each basic cell. The critical path is extended for one additional multiplexer, so the basic clock frequency is slightly decreased.

The involving of changeable folding sets in the synthesized folded architecture allows the reducing of folding factor according to the coefficient length increasing the throughput of the folded system. The wider application area and the finding of suitable area-time tradeoffs are provided for the bit-plane architecture through the application of folding technique with changeable folding factor.

Both the BPA and the FBPA are implemented in FPGA technology. The results, relating on occupied chip resources and achieved throughputs, are presented and discussed.

References

[1] Y-C. Lin, F-C. Lin, "Classes of Systolic Arrays for Digital Filtering", *Int. J. Electronics*, Vol. 70, No. 4, 1991, pp. 729-737.

[2] I. Milentijević, M. S. Stojčev, D. Maksimović, "Configurable Digit - Serial Convolver of Type F", *Microelectronics Journal*, Vol. 27. No. 6, Sep. 1996, pp. 559-566.

[3] I. Milentijević, I. Milovanović, E. Milovanović, M. Tošić, M. Stojčev, "Two - Level Pipelined Systolic Arrays for Matrix - Vector Multiplication", *Journal of Systems Architecture*, The EUROMICRO Journal, Vol. 44, No. 5, Feb. 1998, pp. 383 -387.

[4] P. Corsonello, S. Perri, and G. Cocorullo, "Area-Time-Power Tradeoff in Cellular Arrays VLSI Implementations", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 8, No. 5, Oct. 2000, pp. 614-624.

[5] R. Lin, "Reconfigurable Parallel Inner Product Processor Architectures", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol.9, No.2, Apr. 2001, pp. 261-272.

[6] M. Gschwind, V. Salapura, "FPGA Prototyping of a RISC Processor Core for Embedded Applications", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9, No 2, April 2001, pp. 241-250.
[7] K. K. Parhi, *VLSI Digital Signal Processing Systems (Design and Implementation)*, John Wiley & Sons, In., New York, 2000.
[8] T. C. Denk, K. K. Parhi, Synthesis of Folded Pipelined Architectures for Multirate DSP Algorithms, *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol. 6, No. 4, Dec. 1998, pp. 595-607.
[9] T. Noll, "Semi-systolic Maximum Rate Transversal Filters with Programmable coefficients", *Workshop of Systolic Architectures*, Oxford, 1986, pp. 103-112.
[10] D. Reuver, H. Klar, "A Configurable Convolution Chip with Programmable Coefficients", *IEEE Journal of Solid State Circuits*, Vol. 27, No. 7, July 1992, pp. 1121 -1123.
[11] I. Milentijević, V. Ćirić, O. Vojinović, T. Tokić, "Folded Semi-Systolic FIR Filter Architecture With Changeable Folding Factor", *Neural, Parallel & Scientific Computations, Dynamic Publishers*, Atlanta, Vol. 10, No 2, 2002, pp. 235-247.