FOLDED BIT-PLANE ARCHITECTURES

I. Milentijević[1], I. Nikolić[1], O. Vojinović[1], V. Đirić[2] and T. Tokić[1]

[1]Faculty of Electronic Engineering, University of Niš,
Beogradska 14, PO Box 73, 18000 Niš, Yugoslavia
{milentijevic, fika, oliver}@elfak.ni.ac.yu
[2]A.D. Tigar, Pirot, Yugoslavia
mciric@tigar.com

Abstract: This paper describes the application of folding technique to the bit-plane systolic FIR filter architecture. We present two additional transformations of original DFG (Data Flow Graph) that enable the application of folding technique and the synthesis of two different folded architectures. One without latches in carry and sum paths suitable for the filtering with small number of coefficients and the other with greater hardware complexity which is fully pipelined.

Keywords: systolic arrays, folding technique, FIR filtering

1. Introduction

Finite Impulse Response (FIR) filters have been widely used for video rate digital filtering. Regular structure of FIR filter algorithm is suitable for implementations on systolic arrays (Lin, 1991). Pipelined cellular arrays represent an appropriate implementation approach for arithmetic circuits where a high computational speed is required (Milentijevic}, 1996, 1998). They are designed in the form of regularly repeated patterns of identical circuits. Thus, due to their geometrical regularity, they are suitable for VLSI implementations, either as stand-alone modules or as a part of a complex digital data path (Corsonello, 2000).

The operations of a FIR filtering can be arranged to a sequential summation of the products using the transposed direct form. However, the approach is limited by the speed of the multipliers. To overcome this deficiency it is common to pipeline multipliers and to combine multiplications and the accumulation of products into an array. This is the basic principle of Bit-Plane Architecture (BPA) (Noll, 1986, Reuver, 1992). 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. All these features point why the BPA is our candidate for application of folding technique.

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 (Parhi, 2000). 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 (Denk, 1998).

The aim of this paper is the synthesis of folded FIR filter architecture based on the BPA. However, the folding transformation can not be applied in a straightforward manner, because the algorithm is based on resorting of partial products. Multiplication cannot be represented as node in the data flow graph (DFG) (Milentijevic}, 2001). Here, we propose the additional transformations of source DFG, based on retiming and reordering of partial products, in order to enable the application of folding transformation. After that, we describe the application of folding technique and present derived folded architectures.

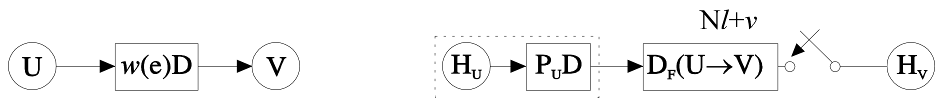
2. Folding technique

The folding technique is introduced by K.K. Parhi and described in (Parhi, 2000, Denk 1998). 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. 1 a) and Fig. 1 b). Fig. 1 a) shows an edge $U \rightarrow V$ with $w(e)$ delays, while Fig. 1 b) depicts the corresponding folded data path. The data begin at the functional unit H_u which has P_u pipelining stages, pass through

$$D_F(U \rightarrow V) = Nw(e) - P_u + v - u \qquad (1)$$

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 $N-1 \geq u, v \geq 0$ [8]. 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 (Parhi, 2000).



a) An edge *U→V* with *w(e)* delays b) The corresponding folded data path.

Figure 1: The synthesis of folded data path

After this short description of folding technique and involving of suitable notation, let us to introduce the source architecture (BPA).

3. Bit-plane architecture

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

$$y_i = c_0x_i + c_1x_{i-1} + ... + c_{k-1}x_{i-k+1}, \qquad (2)$$

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

Computation (2) can be realized in different manners. When high performances are required systolic arrays are frequently used. Semi-systolic array share with systolic arrays desirable simplicity and regularity properties, in addition to their pipelining and multiprocessing schemes of operation. The only difference is that the broadcasting of data to many PEs in one time step is allowed in semi-systolic arrays, while systolic arrays are restricted to temporal locality of communication. Also, the existence of some additional connections can be allowed for semi-systolic architectures (Milentijevic, 1996).

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 $(c_0, c_1, ..., c_{k-1})$,

c_i^j – bit of coefficient c_i (with weight 2^j, and

n – input word length.

The BPA is obtained by resorting of the partial products of different multipliers as it is shown in Fig.2.

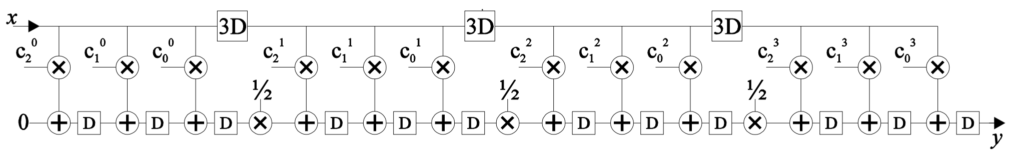


Figure 2: The DFG (Data Flow Graph) for the BPA with *k=3* and *m=4*

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. 2.). 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 (Noll, 1986, Reuver, 1992). 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 synthesis of folded FIR filter architectures.

4. Synthesis of folded architectures

The folding transformation cannot be applied to the BPA directly (Fig. 2.), because the algorithm is based on resorting of partial products, so that multiplications of coefficients and input words are not recognized as operations, i.e. nodes in DFG (Fig. 2.). Each multiplication node in the DFG, shown in Fig. 2., represents one raw of basic cells (full adder and *and* gate) in the array. Thus, one multiplication is distributed through the whole array. Also, additions are performed in rows of basic cells and accumulation is provided by carry–save arithmetic along the array.

Our idea is to declare all operations in one plane to one operation "plane" and to provide time multiplexing to a single functional unit. In order to do that, we have to transform DFG from Fig. 2. Transformed DFG is shown Fig.3. Delays between planes are removed, as well as delays in the addition path. It allows simultaneous operation of plane units, but requires additional latches inside the plane. Transformed DFG is not suitable for implementation because of broadcasting line for input data words, but it is well prepared for folding.

Planes in the transformed DFG are denoted with dashed lines (Fig. 3). There is only one folding set $S=\{1,2,3,4\}$ which contains 4 operations "plane". The folding factor is equal to the coefficient length, i.e. $N=m=4$. There is internal pipelining, P_{int}, in each plane $P_{int} = k-1 = 2$, but planes are connected via broadcast line for input words (without delays) and addition path (without delays, too). Thus, folding equations (1) for the determined folding set (Fig. 3.), where $w(e)=0$ and $P_v=0$, are

$D_F(1\to2) = 4\cdot0 - 0 + 1 - 0 = 1$

$D_F(2\to3) = 4\cdot0 - 0 + 2 - 1 = 1$

$D_F(3\to4) = 4\cdot0 - 0 + 3 - 2 = 1.$

The condition $D_F(U\to V) \geq 0$ is satisfied and additional retiming is not needed. Finally, transformed DFG from Fig. 3. is mapped to the folded architecture (Fig. 4.).

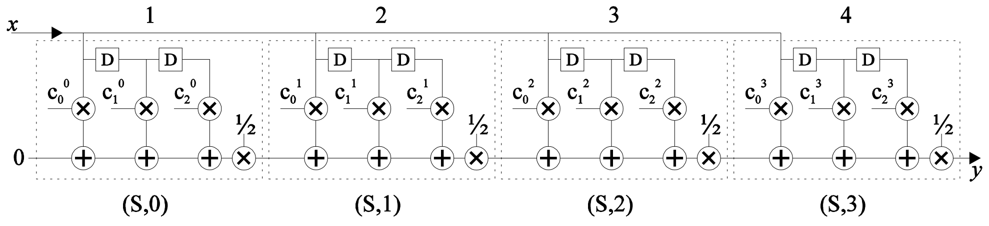


Figure 3: Transformed DFG - TDFG1 with $k=3$ and $n=4$

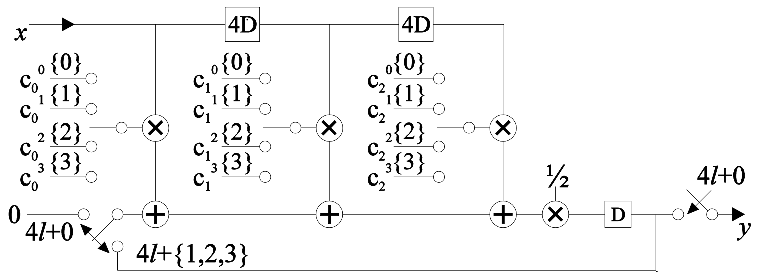
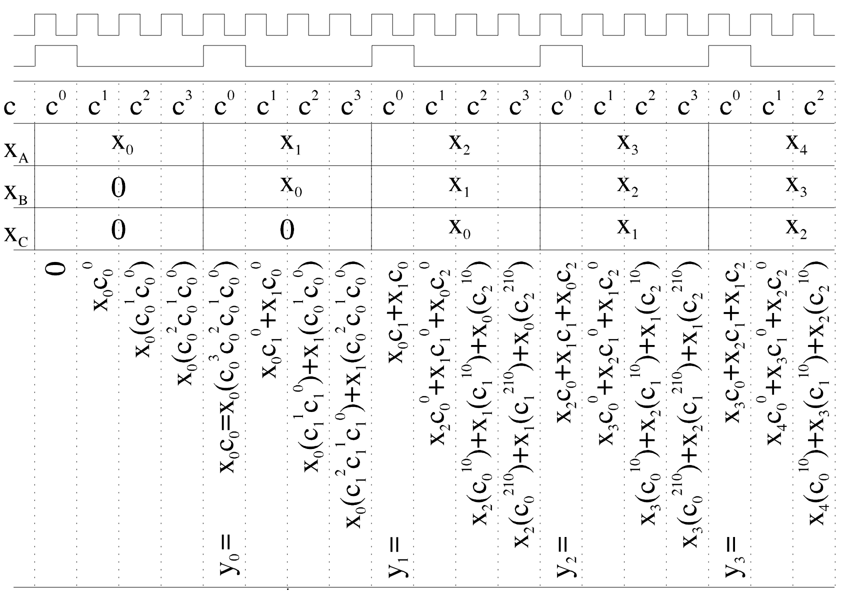


Figure 4: Folded architecture FBPA with folding set $S=\{1,2,3,4\}$

Data flow for this example is given in Fig. 5., where X_A, X_B and X_C describe the presence of input data words at first, second and third row of cells, respectively, while C shows which coefficient bits are used. After the entering of new input word at first clock period all coefficient bits with weight 2^0 are available, zero values are passed through the entering multiplexers and sums of partial products are computed. At second clock period those sums are shifted for one weight right and reentered to the array for summation with partial products of weight 2^1. At the m-th clock period coefficient bits with weight 2^{m-1} are involved. Partial products with corresponding input words are formed and summations are performed. During the $(m+1)$-st clock period the result is obtained and new input value is entered. Each m clock periods one resulting y is generated (in Fig. 4. and Fig. 5. m=4). The initial latency for this architecture is m clock periods.


c
c^0 c^1 c^2 c^3 c^0 c^1 c^2 c^3 c^0 c^1 c^2 c^3 c^0 c^1 c^2 c^3 c^0 c^1 c^2
x_A: x_0 x_1 x_2 x_3 x_4
x_B: 0 x_0 x_1 x_2 x_3
x_C: 0 0 x_0 x_1 x_2
0
$x_0c_0^0$
$x_0(c_0^1c_0^0)$
$x_0(c_0^2c_0^1c_0^0)$
$y_0=$ $x_0c_0=x_0(c_0^3c_0^2c_0^1c_0^0)$
$x_0c_1^0+x_1c_0^0$
$x_0(c_1^1c_1^0)+x_1(c_0^1c_0^0)$
$x_0(c_1^2c_1^1c_1^0)+x_1(c_0^2c_0^1c_0^0)$
$y_1=$ $x_0c_1+x_1c_0$
$x_2c_0^0+x_1c_1^0+x_0c_2^0$
$x_2(c_0^{10})+x_1(c_1^{10})+x_0(c_2^{10})$
$x_2(c_0^{210})+x_1(c_1^{210})+x_0(c_2^{210})$
$y_2=$ $x_2c_0+x_1c_1+x_0c_2$
$x_3c_0^0+x_2c_1^0+x_1c_2^0$
$x_3(c_0^{10})+x_2(c_1^{10})+x_1(c_2^{10})$
$x_3(c_0^{210})+x_2(c_1^{210})+x_1(c_2^{210})$
$y_3=$ $x_3c_0+x_2c_1+x_1c_2$
$x_4c_0^0+x_3c_1^0+x_2c_2^0$
$x_4(c_0^{10})+x_3(c_1^{10})+x_2(c_2^{10})$

$c_i^{10} \equiv c_i^1 c_i^0$

$c_i^{210} \equiv c_i^2 c_i^1 c_i^0$

Figure 5: Data flow for the FBPA (*k=3*, *m=4*)

However, the BPA is suitable for the filtering with small number of coefficients, because the proposed transformation, TDFG1 (Fig. 3), removes latches in carry and sum paths. Thus, the pipelining inside the plane is not employed and the critical path depends on the plane length, i.e. number of coefficients. It motivated us to find another solution for folding of the BPA which will enable the synthesis of fully pipelined folded BPA (FPFBPA).

Now, the crucial question is how to form the folding sets. The TDFG1 enables the existence of only one folding set *S={1,2,3,4}* with four "plane" operations. Let us back to the Fig. 3. Suppose that we have obtained TDFG1, but that we had not formed folding sets yet. If we remove multiplications by *½* from addition path we should involve multiplications by *2* in input data path. It is shown in Fig. 6. The next step is the resorting of partial products collecting all coefficient bits from each coefficient, separately. Now, we have different "plane" from the "plane" in the BPA. This resorting requires delays in input data path. Fig. 7 depicts this step. The DFG from Fig. 7 is not prepared for folding, yet. The obstacle is the existence of latches in input data path. The last step, before folding, shown in Fig. 8 assumes the removing of delays from input data path and their involving into the addition path. This is followed by reverse ordering of coefficients. Finally, we have well prepared DFG, TDFG2, for application of folding technique. The folding sets are formed as it is shown in Fig. 8. The operations which will be folded are denoted with dashed lines. For the BPA with *k=3* and *m=4* there are three folding sets S_1, S_2 and S_3 each of them containing four operations.

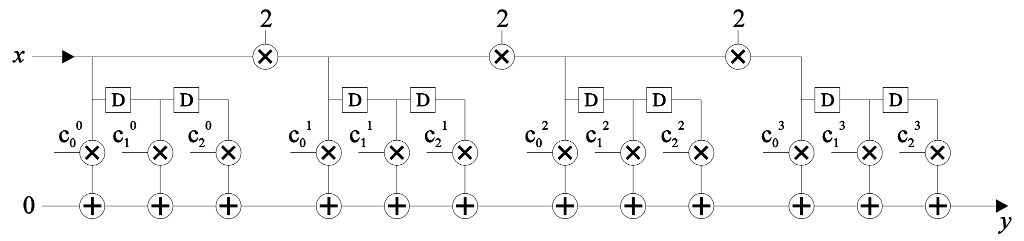


Figure 6: The involving of multiplications in the input data path

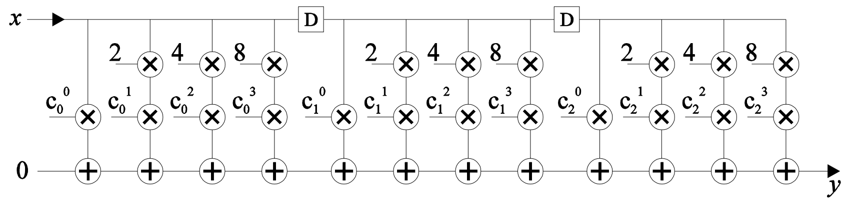


Figure 7: The resorting of partial products collecting all coefficient bits

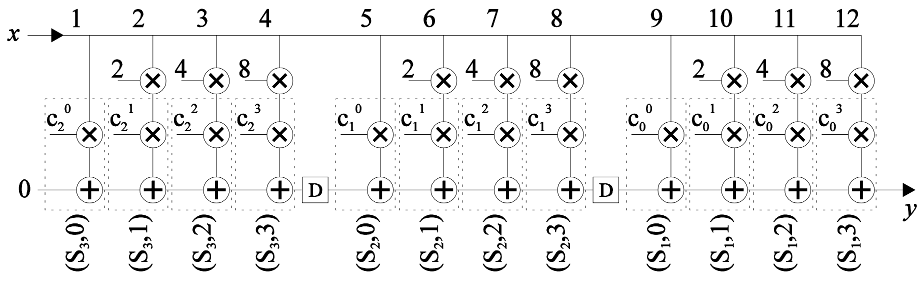


Figure 8: Transformed DFG – TDFG2 with k=3 and n=4

One operation from the TDFG2 (Fig. 8) includes the set of AND gates and full adders from the source BPA. The folding factor is equal to the coefficient length, *N=m=4*. Thus, folding equations (1) for determined folding sets are

$D_F(1\rightarrow 2) = 4\cdot 0 - 0 + 1 - 0 = 1,$

$D_F(2\rightarrow 3) = 4\cdot 0 - 0 + 2 - 1 = 1,$

$D_F(3\rightarrow 4) = 4\cdot 0 - 0 + 3 - 2 = 1,$

$D_F(4\rightarrow 5) = 4\cdot 1 - 0 + 0 - 3 = 1,$

$D_F(5\rightarrow 6) = 4\cdot 0 - 0 + 1 - 0 = 1,$

$D_F(6\rightarrow 7) = 4\cdot 0 - 0 + 2 - 1 = 1,$

$D_F(7\rightarrow 8) = 4\cdot 0 - 0 + 3 - 2 = 1,$

$D_F(8\rightarrow 9) = 4\cdot 1 - 0 + 0 - 0 = 1,$

$D_F(9\rightarrow 10) = 4\cdot 0 - 0 + 1 - 1 = 1,$

$$D_F(10 \rightarrow 11) = 4 \cdot 0 - 0 + 2 - 2 = 1,$$

$$D_F(11 \rightarrow 12) = 4 \cdot 0 - 0 + 3 - 3 = 1.$$

The condition $D_F(U \rightarrow V) \geq 0$ is satisfied and it proves that TDFG2 is well prepared for folding. Obtained fully pipelined folded bit-plane architecture, FPFBPA, is presented in Fig. 9, while Fig. 10 describes data flow through the FPFBPA.

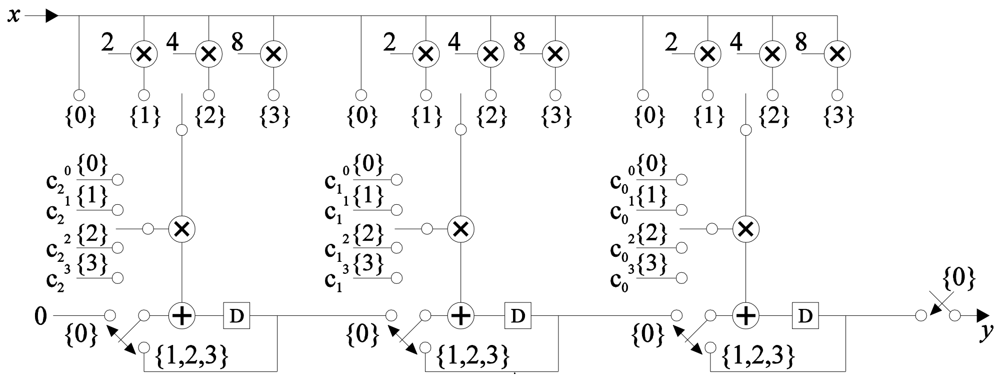
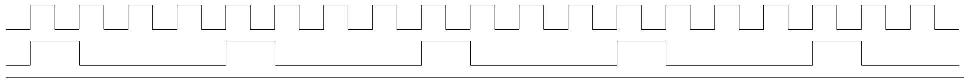


Figure 9: Folded architecture FPFBPA with folding sets S1, S2 and S3

Both FBPA and FPFBPA are described as parameterized FIR filtering cores in VHDL and functionality is proved thorough the logic simulation.



c	x	I row ff (xc_2)	II row ff (xc_1)	III row ff (xc_0)
c^0	x_0	0	0	0
c^1	$2x_0$	$x_0c_2^0$	$x_0c_1^0$	$x_0c_0^0$
c^2	2^2x_0	$x_0(2c_2^1+c_2^0)$	$x_0(2c_1^1+c_1^0)$	$x_0(2c_0^1+c_0^0)$
c^3	2^3x_0	$x_0(2^2c_2^2+2^1c_2^1+c_2^0)$	$x_0(2^2c_1^2+2^1c_1^1+c_1^0)$	$x_0(2^2c_0^2+2^1c_0^1+c_0^0)$
c^0	x_1	$x_0(2^3c_2^3+2^2c_2^2+2^1c_2^1+c_2^0)$	$x_0(2^3c_1^3+2^2c_1^2+2^1c_1^1+c_1^0)$	$y_0 = x_0(2^3c_0^3+2^2c_0^2+2^1c_0^1+c_0^0)$
c^1	$2x_1$	$x_1c_2^0$	$x_1c_1^0+(x_0c_2)$	$x_1c_0^0+(x_0c_1)$
c^2	2^2x_1	$x_1(c_2^1c_2^0)$	$x_1(c_1^1c_1^0)+(x_0c_2)$	$x_1(c_0^1c_0^0)+(x_0c_1)$
c^3	2^3x_1	$x_1(c_2^2c_2^1c_2^0)$	$x_1(c_1^2c_1^1c_1^0)+(x_0c_2)$	$x_1(c_0^2c_0^1c_0^0)+(x_0c_1)$
c^0	x_2	$x_1c_2=x_1(c_2^3c_2^2c_2^1c_2^0)$	$x_1(c_1^3c_1^2c_1^1c_1^0)+(x_0c_2)$	$y_1 = x_1(c_0^3c_0^2c_0^1c_0^0)+(x_0c_1)$
c^1	$2x_2$	$x_2c_2^0$	$x_2c_1^0+(x_1c_2)$	$x_2c_0^0+(x_1c_1+x_0c_0)$
c^2	2^2x_2	$x_2(c_2^1c_2^0)$	$x_2(c_1^1c_1^0)+(x_1c_2)$	$x_2(c_0^1c_0^0)+(x_1c_1+x_0c_0)$
c^3	2^3x_2	$x_2(c_2^2c_2^1c_2^0)$	$x_2(c_1^2c_1^1c_1^0)+(x_1c_2)$	$x_2(c_0^2c_0^1c_0^0)+(x_1c_1+x_0c_0)$
c^0	x_3	$x_2c_2=x_2(c_2^3c_2^2c_2^1c_2^0)$	$x_2(c_1^3c_1^2c_1^1c_1^0)+(x_1c_2)$	$y_2 = x_2(c_0^3c_0^2c_0^1c_0^0)+(x_1c_1+x_0c_0)$
c^1	$2x_3$	$x_3c_2^0$	$x_3c_1^0+(x_2c_2)$	$x_3c_0^0+(x_2c_1+x_1c_0)$
c^2	2^2x_3	$x_3(c_2^1c_2^0)$	$x_3(c_1^1c_1^0)+(x_2c_2)$	$x_3(c_0^1c_0^0)+(x_2c_1+x_1c_0)$
c^3	2^3x_3	$x_3(c_2^2c_2^1c_2^0)$	$x_3(c_1^2c_1^1c_1^0)+(x_2c_2)$	$x_3(c_0^2c_0^1c_0^0)+(x_2c_1+x_1c_0)$
c^0	x_4	$x_3c_2=x_3(c_2^3c_2^2c_2^1c_2^0)$	$x_3(c_1^3c_1^2c_1^1c_1^0)+(x_2c_2)$	$y_3 = x_3(c_0^3c_0^2c_0^1c_0^0)+(x_2c_1+x_1c_0)$
c^1	$2x_4$	$x_4c_2^0$	$x_4c_1^0+(x_3c_2)$	$x_4c_0^0+(x_3c_1+x_2c_0)$
c^2	2^2x_4	$x_4(c_2^1c_2^0)$	$x_4(c_1^1c_1^0)+(x_3c_2)$	$x_4(c_0^1c_0^0)+(x_3c_1+x_2c_0)$

Figure10: Data flow for the FPFBPA (k=3, m=4)

5. Discussion

The first transformation (Fig. 3.) leads to the folded architecture FBPA where all operations are folded to the one "bit-plane" operation. The array is restricted for the factor *m*, where *m* is the length of coefficient. There is no internal pipelining and the critical path depends on number of coefficients. The number of basic cells (basic cell contains full adder and *and* gate) is reduced to the number of basic cells in one plane of source architecture, and the number of latches for carry and sum paths is reduced only to the latches used in one row of source array. The throughput of the FBPA should be decreased approximately $(k-1) \times m$ times in respect to the BPA. The FBPA is suitable only for the FIR filtering with small number of coefficients. The second transformation (Fig. 6, Fig. 7 and Fig. 8), which is more complex then first one, enables the synthesis of fully pipelined folded FIR filter architecture, FPFBPA. There are *k* folding sets, i.e. the number of folding sets is equal to the number of coefficients. Each folding set contains *m* operations. 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 FPFBPA is paid by involving two multiplexers per each basic cell. The critical path is extended for one additional multiplexer, so the basic clock frequency is slightly decreased. Thus, the execution time is increased for slightly more than *m* times in respect to the BPA.

6. Conclusion

The synthesis of two folded FIR filter architectures based on the bit-plane architecture is presented in this paper. The transformations of the source DFG (for BPA) that enable the application of folding technique are proposed. The derived architectures are described and discussed. The application of folding technique to the bit-plane architecture allows achieving of throughput requirements for FIR filtering on integrated circuits with low silicon area.

7. References

1. Corsonello P., Perri S., and Cocorullo G., "Area-Time-Power Tradeoff in Cellular Arrays VLSI Implementations", *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol. 8, No. 5, Oct. 2000, pp. 614-624.
2. Denk T. C., Parhi K. K., "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.
3. Lin Y-C., F-C, "Classes of Systolic Arrays for Digital Filtering", *Int. J. Electronics*, Vol. 70, No. 4, 1991, pp. 729-737.

4. Milentijevi} I., Milovanovi} I., Milovanovi} E., To{i} M., Stojčev M., "Two - Level Pipelined Systolic Arrays for Matrix - Vector Multiplication", *Journal of Systems Architecture*, *The EROMICRO Journal*, Vol. 44, No. 5, Feb. 1998, pp. 383 -387.
5. Milentijevi} I., Stojčev M. S., Maksimovi} D., "Configurable Digit - Serial Convolver of Type F", *Microelectronics Journal*, Vol. 27. No. 6, Sep. 1996, pp. 559-566.
6. Milentijević I., Tokić T., Nikolić I., Vojinović O., and Širić V., "Synthesis of Folded FIR Filter Architecture With Reordered Partial Products", *Proceedings of a Workshop on Computational Intelligence and Informational Technologies*, June 20-21, Ni{, Yugoslavia, 2001, pp. 155-160.
7. Noll T.,"Semi-systolic Maximum Rate Transversal Filters with Programmable coefficients", *Workshop of Systolic Architectures*, Oxford, 1986, pp. 103-112.
8. Parhi K. K., *VLSI Digital Signal Processing Systems (Design and Implementation)*, John Wiley & Sons, In., New York, 2000.
9. Reuver D., Klar H., "A Configurable Convolution Chip with Programmable Coefficients", *IEEE Journal of Solid State Circuits*, Vol. 27, No. 7, July 1992, pp. 1121 -1123.