June 2002
VOLUME 10 NUMBER 2
ISSN 1061 5369

Prof. I. Milentijevic
Faculty of Electronic Engineering
University of Nis
P. O. Box 73,
18000 Nis, Serbia

NEURAL, PARALLEL & SCIENTIFIC COMPUTATIONS

DYNAMIC PUBLISHERS • ATLANTA

Neural, Parallel & Scientific Computations

The aim of this quarterly journal is to provide an international forum for the information in the theory and practice of Neural, Parallel & Scientific Computations. This journal publishes carefully selected original research papers on all aspects of neural, parallel and scientific computing from Computational Mathematics, Biomedical Engineering, Artificial Intelligence, Systolic Algorithms, Evaluation and Prediction of Computer Complexes, Cluster Computing, Computational Complexity, VLSI Design, Computer Architectures, Simulation, ODL (Open Distance Learning) Systems, Systems Security, Combinatorics, Graph Theory, Numerical Analysis, etc.

Articles published in **Neural, Parallel and Scientific Computations** are indexed or abstracted in: Computer Abstracts; Computer & Control Abstracts; Computer Information System Abstracts; Computer Literature Index; Computer Review; Ergonomics Abstracts; Abstract Journal, USSR Academy of Sciences; Zentralblat fur Mathematik/Mathematics Abstracts; Mathematical Reviews; Current Mathematical Publications; MathSci.

Neural, Parallel and Scientific Computations (ISSN 1061-5369) is published quarterly by Dynamic Publisher, Inc, Atlanta, GA 30362, USA, in the months of March, June, September, and December.

Subscription Rates: Volume 10 (2002)

Library/Institutional: **US$ 400.00**; Personal: **US$ 200.00**

(All countries except USA add **US$ 40.00** for surface mail, or **US$ 60.00** for Air Mail). Subscription and all other inquires should be addressed to **Neural, Parallel and Scientific Computations , Dynamic Publisher, Inc, P. O. BOX 48654, Atlanta, GA 30362-0654, USA**. Editorial inquires should be directed to the **EDITOR (M. Sambandham)**.

POSTMASTER: Send address changes to **Neural, Parallel and Scientific Computations,** Dynamic Publisher, Inc, P. O. BOX 48654, Atlanta, GA 30362-0654, USA.

Printed in U.S.A

Neural, Parallel & Scientific Computations 10 (2002) 235-248

FOLDED SEMI-SYSTOLIC FIR FILTER ARCHITECTURE WITH CHANGEABLE FOLDING FACTOR

I. Milentijević, V. Ćirić, O. Vojinović, T. Tokić
Faculty of Electronic Engineering
University of Niš
P.O. box 73, 18000 Niš, Serbia

ABSTRACT: The synthesis of new folded semi-systolic FIR filter architecture with changeable folding factor is presented in this paper. The transformation of the original data flow graph for the bit-plane architecture that enables the successful application of the folding technique with changeable folding sets is proposed. The application of folding technique at bit level that allows the implementation of changeable folding factor onto the fixed size array is described. 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 finding of suitable area-time tradeoffs for the folded semi-systolic FIR filter architecture is provided by the presented synthesis procedure.

1. INTRODUCTION

The Finite Impulse Response (FIR) filtering is one of the important special purpose arithmetic operations widely used for video rate digital filtering. It is very inefficient if the computation is done by software on the core central processing unit [Howley et al., 1996]. Regular structure of FIR filter algorithm is suitable for implementation on systolic arrays, which are attached to the host computer as hardware accelerators. Pipelined cellular arrays are designed in the form of regularly repeated patterns of identical circuits [Lin, 1991; Corsonello et al., 2000; Milentijević et al., 1998]. Thus, due to their geometrical regularity, they are suitable for VLSI implementations, either as stand-alone 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 [Milovanović, 1996]. Under a restricted VLSI area the design of such a processor often introduces a conflict between its versatility and computation speed [Lin, 2001].

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

Received June 18, 2002

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&Parhi, 1998].

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 [Noll, 1986; Reuver&Klar, 1992]. However, the folding transformation can not be applied to the BPA in a straight foreword manner, because the algorithm is based on resorting of partial products, so that multiplication of coefficient and input data word is not recognized as an operation i.e. a node in Data Flow Graph (DFG). Therefore, the additional transformation of the original DFG for the BPA that prepares DFG for the application of folding technique should be found. One solution is presented in [Milentijević, 2001] where proposed transformation removes latches in carry and sum paths, so the pipelining inside the plane is not employed. It leads to the successful application of the folding technique, but the obtained target architecture is suitable only for small number of coefficients (plane length is equal to the number of coefficients). All these facts have motivated us to propose a new transformation of source DFG for BPA that enables the application of folding technique. As a result we obtain folded fully pipelined semi-systolic architecture for FIR filtering. In this paper we describe the complete synthesis path from the source DFG to the target architecture. Also, the goal of this paper is to present the application of folding technique 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. The proposed application of folding technique should enable the finding of suitable area-time tradeoffs for bit-plane architecture keeping all desirable features of the source architecture and to provide wider application area for the synthetized folded semi-systolic architecture.

The paper is organized as follows: the section 2. contains basic elements of folding technique; the section 3. describes the BPA as a basic architecture; in the section 4. we give the transformation of the original DFG for the BPA that enables the application of folding technique; the section 5. describes the involving of changeable folding factor and contains the description of the synthetized folded semi-systolic architecture, while the section 6. gives the concluding remarks.

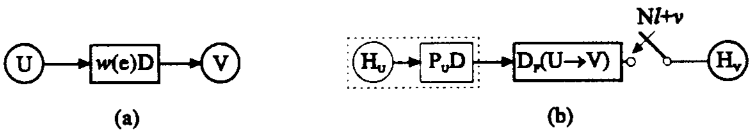
2. BASIC ELEMENTS OF FOLDING TECHNIQUE

The folding technique is introduced by K.K. Parhi and described in [Parhi, 2000; Denk&Parhi, 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 Figure 1 a) and Figure 1 b). Figure 1 a) shows an edge $U \rightarrow V$ with $w(e)$ delays, while Figure 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

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

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 $N-1 \geq u, v \geq 0$. 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 $U \rightarrow V$ 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, let us to introduce the BPA as a source architecture for synthesis of folded semi-systolic architecture with changeable folding factor.

3. FIR FILTERING ON BIT PLANE SEMI-SYSTOLIC ARCHITECTURE

Output words { y_i } of FIR filter are computed as

$$y_i = c_0 x_i + c_1 x_{i-1} + \ldots + c_{k-1} x_{i-k+1}, \tag{2}$$

where $c_0, c_1, \ldots, c_{k-1}$ 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 [Noll, 1986; Reuver&Klar, 1992; Milentijević, 1996].

The bit–plane architecture (BPA) is semi-systolic architecture that provides regular connections with extensive pipelining and high computational throughput. The BPA is a basis for synthesis of folded semi-systolic FIR filter architecture, 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 $(c_0, c_1, \dots, c_{k-1})$, and

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

n – input word length.

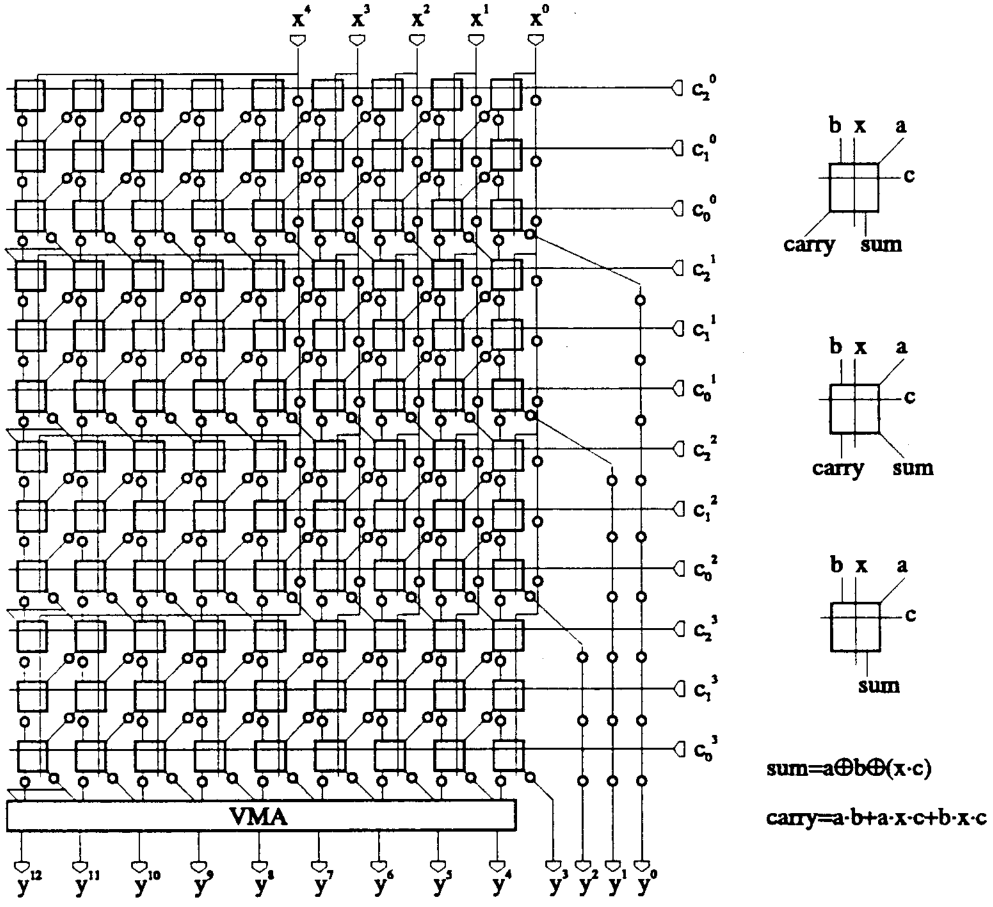
The BPA is obtained by resorting of the partial products of different multipliers as it is shown in Figure 2. With fine-grained pipelining, the splitted parts of the multiplications become input word times $1-b$ `coefficient multiplications, the partial products. These are just logical AND function between the input word and coefficient bit. In the first bit–plane the least significant partial products of all coefficients are computed and accumulated. 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&Klar, 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 the synthesis of the fully pipelined folded FIR filter architecture. The BPA with $k=3$, $m=4$ and $n=5$ is shown in Figure 2. The corresponding DFG for $k=3$ and $m=4$ is given in Figure 3. The transfer function for the BPA ($k=3$; $m=4$), which is shown in Figures 2 and 3, is

$$G(z)=\frac{Y(z)}{X(z)}=z^{-1}(c_0^{\,3}2^3z^{-9}+z^{-1}(c_1^{\,3}2^3z^{-9}+z^{-1}(c_2^{\,3}2^3z^{-9}+z^{-1}(c_0^{\,2}2^2z^{-6}+z^{-1}(c_1^{\,2}2^2z^{-6}+z^{-1}(c_2^{\,2}2^2z^{-6}+$$
$$+z^{-1}(c_0^{\,1}2^1z^{-3}+z^{-1}(c_1^{\,1}2^1z^{-3}+z^{-1}(c_2^{\,1}2^1z^{-3}+z^{-1}(c_0^{\,0}2^0+z^{-1}(c_1^{\,0}2^0+z^{-1}c_2^{\,0}2^0)\dots).$$

In order to enable the application of folding technique and to synthesize the fully pipelined semi-systolic FIR filter architecture with changeable folding factor we will concern the general form of the DFG for BPA with k-taps and m-bit coefficients (Figure 4). The general form of transfer function for k taps and m bit coefficient wordlength, which corresponds to the DFG from Figure 4, is

$$G(z)=z^{-1}(c_0^{\,m-1}2^{m-1}z^{-(m-1)k}+z^{-1}(c_1^{\,m-1}2^{m-1}z^{-(m-1)k}+\dots+z^{-1}(c_{k-1}^{\,m-1}2^{m-1}z^{-(m-1)k}+$$
$$+z^{-1}(c_0^{\,m-2}2^{m-2}z^{-(m-2)k}+z^{-1}(c_1^{\,m-2}2^{m-2}z^{-(m-2)k}+\dots+z^{-1}(c_{k-1}^{\,m-2}2^{m-2}z^{-(m-2)k}+$$
$$\dots$$
$$+z^{-1}(c_0^{\,0}2^0z^0+z^{-1}(c_1^{\,0}2^0z^0+\dots+z^{-1}(c_{k-1}^{\,0}2^0z^0)\dots). \qquad (3)$$

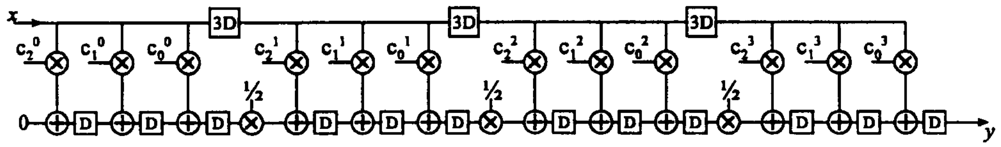
The transfer function (3) and the DFG from Figure 4 will be transformed into the forms that enable the application of folding technique.



$$\mathrm{sum}=a\oplus b\oplus(x\cdot c)$$

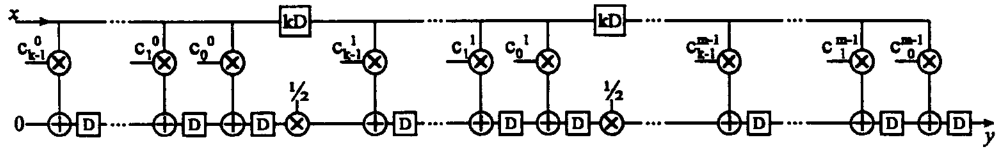
$$\mathrm{carry}=a\cdot b+a\cdot x\cdot c+b\cdot x\cdot c$$

Figure 2: The BPA for *k=3* and *m=4*



x
3D
3D
3D
c_2^0 c_1^0 c_0^0 c_2^1 c_1^1 c_0^1 c_2^2 c_1^2 c_0^2 c_2^3 c_1^3 c_0^3
½
½
½
0
D
y

Figure 3: The DFG for the BPA with *k=3* and *m=4*



x
kD
kD
c_{k-1}^0 c_1^0 c_0^0 c_{k-1}^1 c_1^1 c_0^1 c_{k-1}^{m-1} c_1^{m-1} c_0^{m-1}
½
½
0
D
y

Figure 4: The DFG for the BPA with *k* taps and *m* -bit coefficients

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 Figure 2 and Figure 3, 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 Figure 3, represents one row of basic cells (full adder and AND gate) in Figure 2. Thus, one multiplication is distributed through the whole array. Even, if we declare the 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 $D_F(U \rightarrow V) \geq 0$ for all of the edges in the DFG. The solution proposed in [Milentijević, 2001] declares all computations in one plane as one "plane" operation and provides time multiplexing to a single functional unit. Transformed DFG, TDFG1, proposed in [Milentijević, 2001] is shown Figure 5. 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. The TDFG1 is well prepared for folding, i.e. it enables the application of folding technique. However, TDFG1 leads to the folded BPA which is suitable for the filtering with small number of coefficients. The proposed transformation (Figure 5) 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.

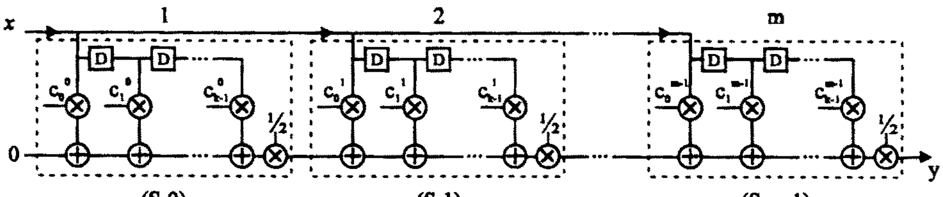


Figure 5: Transformed DFG-TDFG1 with one folding set S that contains *m* "plane" operations

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 folding factor *N*, at the cost of time, and that the derived architecture keeps all desirable features, especially fine-grain pipelining. A new transformation will be presented in two ways. Firstly, we introduce the transformation on DFGs as a 4-step procedure, and than we prove it at the transfer function level.

Step_1 The starting DFG is the DFG for the BPA with *k* taps and *m* -bit coefficients shown in Figure 4. Removing the delays between planes as well as the delays in the addition path we obtain the TDFG1 where delays are included inside the planes (Figure 5). The TDFG1 enables the existence of only one folding set *S={0,1,...,m-1}* with *m* "plane" operations. However, it leads to the solution proposed in [Milentijević, 2001], which suffers from long critical paths, i.e. there is no fine-grain pipelining. So, suppose that we have obtained TDFG1, but that we had not formed folding sets yet.

Step_2 Multiplications by ½ are removed from the addition path while multiplications by *2* are involved in the input data path. It is shown in Figure 6.

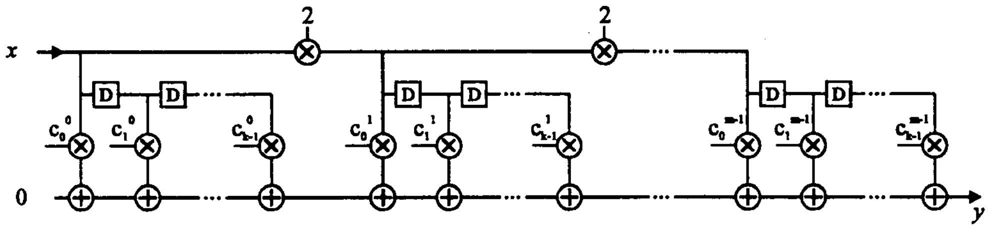


Figure 6: The involving of multiplications in the input data path

Step_3 The resorting of partial products collecting all coefficient bits from each coefficient, separately, is performed (Figure 7). Now, we have different "plane" from the "plane" in the BPA. This resorting requires delays in the input data path. Figure 7 depicts this step. The DFG from Figure 7 is not prepared for folding, yet. The obstacle is the existence of latches in the input data path.

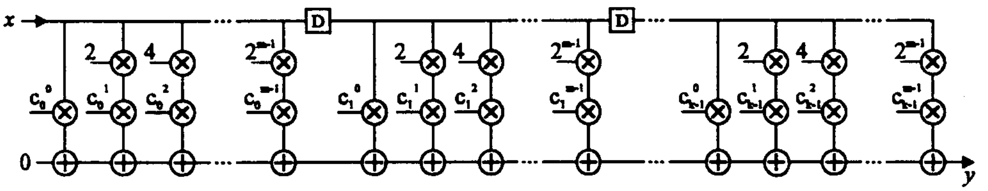


Figure 7: The resorting of partial products collecting of coefficient bits from each coefficient separately

Step_4 The last step before folding, shown in Figure 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 the DFG, TDFG2, for the application of folding technique.

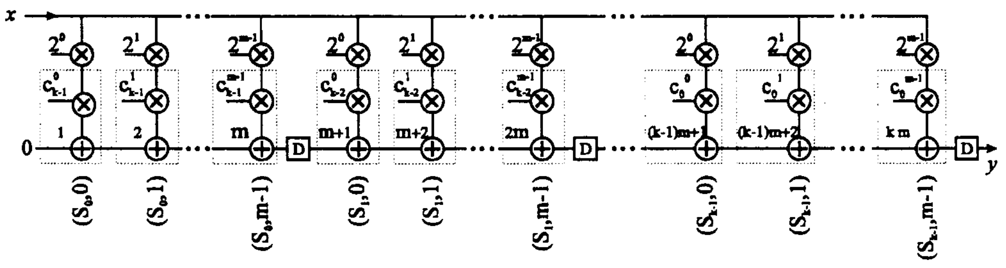


Figure 8: Transformed - TDFG2 with k folding sets each of them containing m "row" operations

The correctness of activities performed in previous steps (Step_1 to Step_4) can be proved at transfer function level. The starting point in Step_1 is the DFG shown in Figure 4, and the corresponding transfer function is given by equation (3). The same transfer function without brackets can be rewritten as follows

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

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

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

The developed form of equation (4) is

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

The corresponding DFG for equation (5) is actually the TDFG2, shown in Figure 8, derived through Steps_1 to 4. In other words the transformations on the transfer function, from (3) to (5), follow the previously described scenario.

The architecture with transformed DFG from Figure 8 is impractical for implementation because of broadcast line at input data path, but TDFG2 is well prepared for further application of folding technique at bit level.

Step_5 is an important additional step. 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 Figure 8. The operations that will be folded are denoted with dashed lines. One operation from TDFG2 assumes forming of partial products and the addition performed on one "row" of basic cells, (where basic cell contains AND gate and full adder – Figure 2). 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 (1) for the determined folded sets, where $P_U = 0$ and U and V are nodes in TDFG from Figure 8 denoted with $1, 2, ..., m, m+1, ..., km$ are

$$D_F(1 \to 2) = m \cdot 0 - 0 + 1 - 0 = 1$$
$$D_F(2 \to 3) = m \cdot 0 - 0 + 2 - 1 = 1$$
$$\ldots$$
$$D_F(m-1 \to m) = m \cdot 0 - 0 + (m-1) - (m-2) = 1$$
$$D_F(m \to m+1) = m \cdot 1 - 0 + 0 - (m-1) = 1$$
$$D_F(m+1 \to m+2) = m \cdot 0 - 0 + 1 - 0 = 1$$
$$\ldots$$
$$D_F(2m-1 \to 2m) = m \cdot 0 - 0 + (m-1) - (m-2) = 1$$
$$D_F(2m \to 2m+1) = m \cdot 1 - 0 + 0 - (m-1) = 1$$
$$D_F(2m+1 \to 2m+2) = m \cdot 0 - 0 + 1 - 0 = 1$$
$$\ldots$$
$$D_F(km-1 \to km) = m \cdot 0 - 0 + (m-1) - (m-2) = 1.$$

The condition $D_F(U \to V) \geq 0$ is satisfied, for each pair of connected nodes (U, V), and it proves that TDFG2 from Figure 8 is well prepared for folding. The obtained folded architecture with k taps is presented in Figure 9.

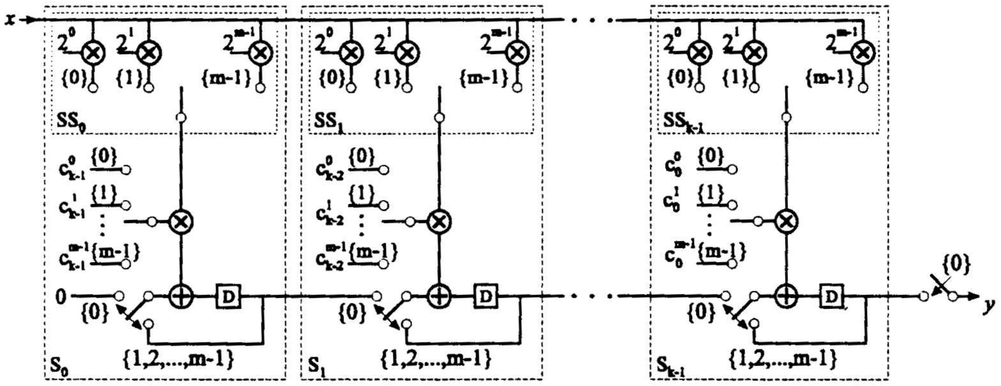


Figure 9: Folded architecture with folding sets S_0, S_1, ... , S_{k-1}

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. The proposed application of folding technique, Figure 8, allows the simultaneous bit-serial operation of all taps. All shift sections ($SS_0, SS_1, ..., SS_{k-1}$ from Figure 9) can be implemented by one simple shift register (Figure 10). 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 (Figure 8 and Figure 9). 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 from Figure 9 with changeable folding factor, based on the TDFG2 from Figure 8 for $k = 3$, $n = 5$ and $1 \leq m \leq m_1$, is given in Figure 10.

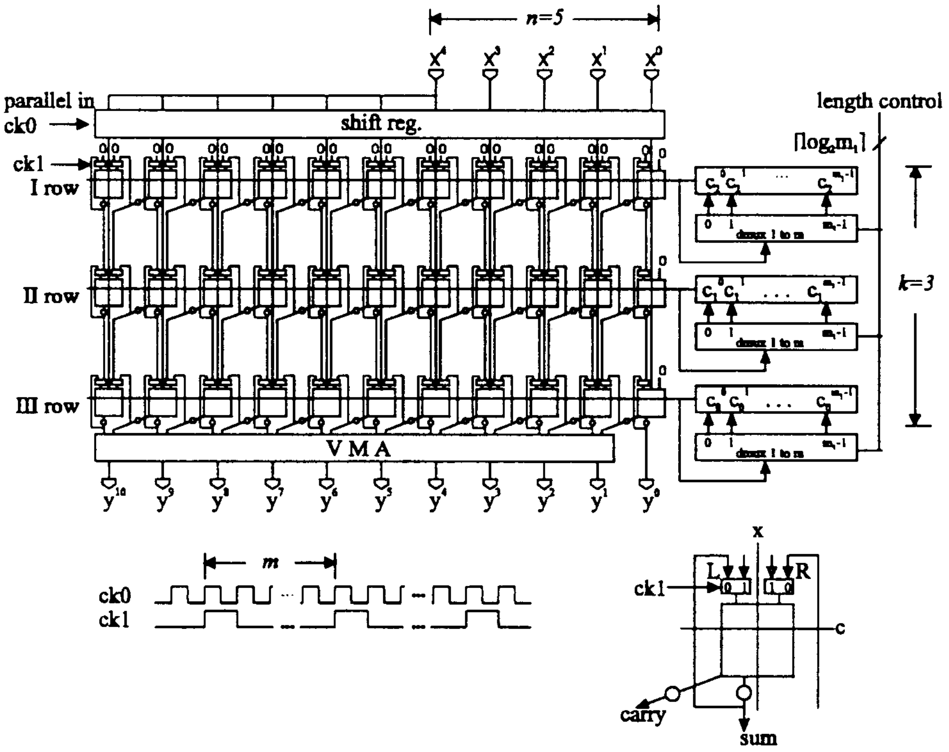


Figure 10: The functional block diagram for folded architecture with changeable folding factor (k=3)

The operation with different folding factors is provided by:

- simple changing of control signal *ck1* which period should be equal to the duration of m periods of the basic clock signal $ck0$;

- shift / rotate registers for coefficients joined with simple control logic which provides m-bit rotation ($1 \le m \le m_1$), 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 results.

In order to clarify the operation of folded architecture from Figure 10 we give the data flow for the case when $k=3$ and $m=4$ in Figure 11. New input data word x_i at inputs x^0, x^1, ... , x^{n-1} is entered into the shift register every m-clock cycles. Bits of input words are broadcasted to all bit-serial taps, i.e. rows (row I, row II and row III in Figure 10 and Figure 11 for $k=3$). Line x in Figure 11 shows the shifted values of input data words that are available to the processing array, while line c describes the presence of coefficient bits, i.e. c^j ($j=0$, ... , $m-1$) denotes the presence of coefficient bits with weight 2^j from all coefficients c_i ($i=0$, ... , $k-1$) at the corresponding rows. Additional multiplexers in each basic cell, denoted with L and R in Figure 10, accepts internal folded paths for carries and sums or carries and sums from the previous row (i.e. zeros for the first row). Thus, the internal folding in each tap, Figure 9, is implemented. It allows simultaneous operation of all rows using the same input data word. One row of basic cells provides 1-bit multiplication, i.e. forming of one partial product and summation with previous partial products. The row performs the multiplication by m-bit coefficient for m-clock cycles and provides accumulated intermediate result for the next row. Final summation performs the Vector Merging Adder (VMA) attached to the outputs of the last row. During the (m+1)-st clock period the result is obtained and a new input value is entered. The initial latency for the folded BPA is m clock periods, while one resulting y is generated each m clock cycles. The data flow for folded BPA when $k=3$ and $m=4$ is shown in Figure 11.

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 $1 \le m \le m_1$, 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 m_1/m times.

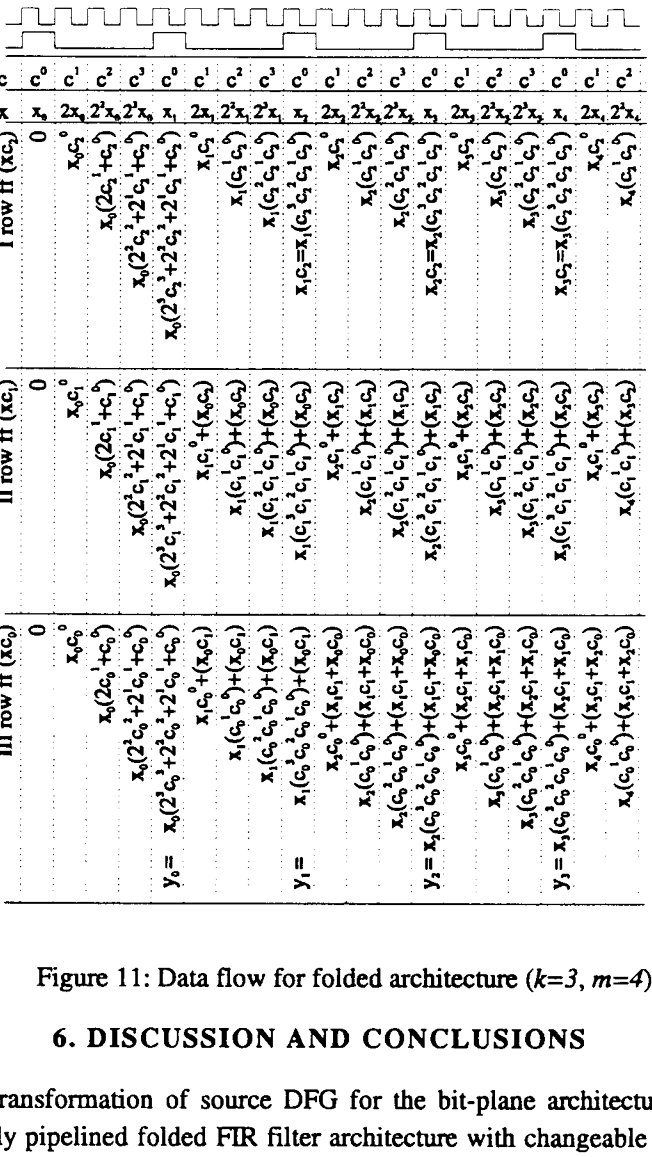


Figure 11: Data flow for folded architecture (*k=3*, *m=4*)

6. DISCUSSION AND CONCLUSIONS

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 $N = m$, i.e. the folding factor is equal to the coefficient length. 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 bit-plane architecture. 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. Thus, the throughput is decreased for slightly more than m times in respect to the BPA.

The obtained folded semi-systolic architecture is presented by the DFG, the functional block diagram and the data flow diagram.

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. Wider application area and the finding of suitable area-time tradeoffs are provided for the bit-plane architecture through the application of folding technique while the additional increasing of throughput for folded semi-systolic architecture is achieved by the implementation of the changeable folding factor.

REFERENCES

1. Corsonello, P., Perri S., & Cocorullo, G., (2000). Area-Time-Power Tradeoff in Cellular Arrays VLSI Implementations. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems,* v. 8, No. 5, 614-624.
2. Denk, T. C., & Parhi, K. K., (1998). Synthesis of Folded Pipelined Architectures for Multirate DSP Algorithms. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems,* v. 6, No. 4, 595-607.
3. Hawley, R., Wong, B., Lin, T-J., Laskowski, J., & Samueli, H., (1996). Design Techniques for Silicon Compiler Implementations of High-Speed FIR Digital Filters. *IEEE Journal of Solid State Circuits,* v. 31, No 5, 656-667.
4. Lin, R., (2001). Reconfigurable Parallel Inner Product Processor Architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* v. 9, No. 2, 261-272.
5. Lin, Y-C., Lin, F-C., (1991). Classes of Systolic Arrays for Digital Filtering. *Int. J. Electronics,* v. 70, No. 4, 729-737.
6. Milentijević, I., Milovanović, I., Milovanović, E., Tošić, M., & Stojčev, M., (1998). Two - Level Pipelined Systolic Arrays for Matrix - Vector Multiplication. *Journal of Systems Architecture, The EROMICRO Journal,* v. 44, No. 5, 383-387.
7. Milentijević, I., Stojčev, M. S., & Maksimović, D., (1996). Configurable Digit - Serial Convolver of Type F. *Microelectronics Journal,* v. 27. No. 6, 559-566.
8. Milentijević, I., Tokić, T., Nikolić, I., Vojinović, O., & Ćirić, V., (2001). Synthesis of Folded FIR Filter Architecture With Reordered Partial Products. Proceedings of a Workshop on Computational Intelligence and Informational Technologies, pp. 155-160, Niš, Yugoslavia, June.
9. Milovanović, I., Milovanović, E., Milentijević, I., Stojčev, M., (1996). Designing of Processor-Time Optimal Systolic Arrays for Band Matrix-Vector Multiplication. *Computers Math. Applic.,* v. 32, No. 2, 21-31
10. Noll, T., (1986). Semi-systolic Maximum Rate Transversal Filters with Programmable coefficients, Workshop of Systolic Architectures, pp. 103-112, Oxford, U.K.
11. Parhi, K. K., (2000). *VLSI Digital Signal Processing Systems (Design and Implementation).* New York: John Wiley & Sons, In..
12. Reuver, D., & Klar, H., (1992). A Configurable Convolution Chip with Programmable Coefficients. *IEEE Journal of Solid State Circuits,* v. 27, No. 7, 1121 -1123.

Neural, Parallel & Scientific Computations

Volume 10, Number 2, June 2002

CONTENTS