DSD 2002
EUROMICRO Symposium on DIGITAL SYSTEM DESIGN

Architectures, Methods and Tools

Dortmund, Germany
4-6 September 2002

Edited by Martyn Edwards

UNIDO

Proceedings

Euromicro Symposium on Digital System Design

Architectures, Methods and Tools

September 4 – 6, 2002
Dortmund, Germany

Edited by: Martyn Edwards

IEEE
COMPUTER
SOCIETY
http://computer.org

Los Alamitos, California
Washington • Brussels • Tokyo

Folded Bit-Plane FIR Filter Architecture with Changeable Folding Factor

Ivan Milentijević, Vladimir Ćirić, Teufik Tokić and Oliver Vojinović
milentijevic@elfak.ni.ac.yu
Faculty of Electronic Engineering, University of Niš
Beogradska 14, PO Box 73, 18000 Niš, Yugoslavia

Abstract

The application of folding technique to the bit-plane systolic FIR filter architecture that enables the implementation of changeable folding factor onto the fixed size array is described in this paper. The transformation of original Data Flow Graph (DFG) for bit-plane architecture that provides the successful application of the folding technique with changeable folding sets is presented. 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.

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. 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 [1-4]. 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. Under a restricted VLSI area the design of such a processor often introduces a conflict between its versatility and computation speed [5].

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 is used. 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 [6]. 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 [7].

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 [8,9].

The straightforward application of folding technique to the BPA is not possible. The algorithm is based on the resorting of partial products while the multiplication is spread through the whole processing array. It implies that an additional bit-level transformation of the source DFG, that will enable the successful application of the folding technique should be found. We propose such transformation in a general form for k taps and the coefficient length m. Thus, the application of folding technique becomes possible as well as the involvment of changeable folding sets. The changing of folding factor, i. e. its reduction according to the coefficient length is aimed to the increasing of throughput of the folded system. Also, the proposed application of the folding technique should provide the increase of versatility and suitable area-time tradeoffs for the BPA. 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 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 involvement of changeable folding factor, while in section 6. the performances of derived folded architecture are discussed.

2. Bit-plane FIR filter architecture

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

$$y_i = c_0 x_i + c_1 x_{i-1} + \dots + c_{k-1} x_{i-k+1}, \qquad (1)$$

where $c_0, c_1, \dots, c_{k-1}$ 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 $(c_0, c_1, \dots, c_{k-1})$, 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 Figure 1.

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 (Figure 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 [8, 9]. 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 and involving of suitable notation, 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 [6, 7]. 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 2 a) and Figure 2 b). Figure 2 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

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

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 $N - 1 \geq u, v \geq 0$. 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 [6].

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 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 Figure 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 which corresponds to the DFG for the BPA ($k = 3$; $m = 4$) shown in Figure 1.:

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

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

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

The same transfer function without brackets can be rewritten as follows

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

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

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

The developed form of equation (4) is

$$G(z)=z^0(c_0{}^{m-1}2^{m-1}+c_0{}^{m-2}2^{m-2}+\ldots+c_0{}^0 2^0+$$
$$+z^{-1}(c_1{}^{m-1}2^{m-1}+c_1{}^{m-2}2^{m-2}+\ldots+c_1{}^0 2^0+$$
$$\ldots$$
$$+z^{-1}(c_{k-1}{}^{m-1}2^{m-1}+c_{k-1}{}^{m-2}2^{m-2}+\ldots+c_{k-1}{}^0 2^0)\ldots) \quad (5)$$

The corresponding DFG for equation (5) is shown in Figure 3. In other words, the transformation of the transfer function from (3) to (5), at the DFG level starting from the DFG from Figure 1, is performed according to the following scenario:

- 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.

The architecture with transformed DFG from Figure 3. is impractical for implementation because of 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 Figure 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, $S_0, S_1, S_2, \ldots, S_{k-1}$, 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 Figure 3. denoted with $1,2,\ldots,m,m+1,\ldots,km$ are

$$D_F(1\rightarrow 2)=m\cdot 0-0+1-0=1$$
$$D_F(2\rightarrow 3)=m\cdot 0-0+2-1=1$$
$$\ldots$$
$$D_F(m-1\rightarrow m)=m\cdot 0-0+(m-1)-(m-2)=1$$
$$D_F(m\rightarrow m+1)=m\cdot 1-0+0-(m-1)=1$$
$$D_F(m+1\rightarrow m+2)=m\cdot 0-0+1-0=1$$
$$\ldots$$
$$D_F(2m-1\rightarrow 2m)=m\cdot 0-0+(m-1)-(m-2)=1$$

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

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

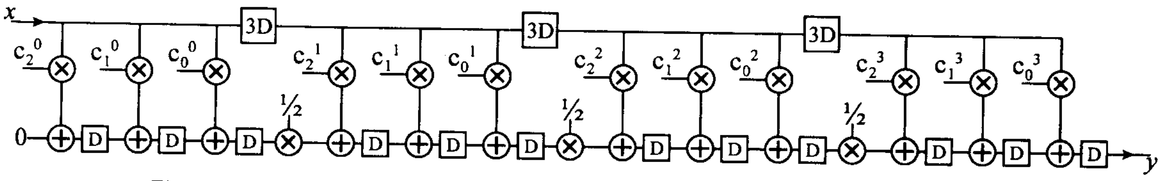
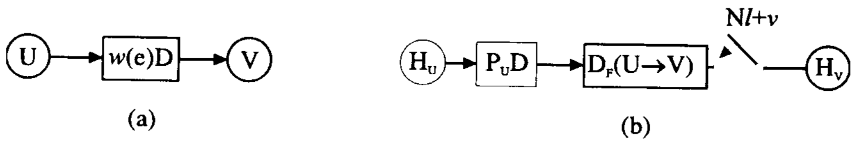


Figure 1. The DFG (Data flow graph) for the BPA with *k=3* and *m=4*



(a) An edge $U \to V$ with $w(e)$ delays;(b) The corresponding folded data path

Figure 2. The synthesis of folded data path.

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, Figure 4. allows the simultaneous bit-serial operation of all taps. All shift sections ($SS_0, SS_1, \dots, SS_{k-1}$) 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 (Figure 3. and Figure 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 Figure 4. for $k = 3, n = 5$ and $1 \leq m \leq m_1$ is given in Figure 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.

In order to clarify the operation of folded architecture from Figure 5 we give the data flow for the case when $k = 3$ and $m = 4$ in Figure 6.

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 5 and Figure 6 for

$k = 3$). Line x in Figure 6 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 ($j=0, \dots, m-1$) denotes the presence of coefficient bits with weight 2^j from all coefficients c_i ($i=0, \dots, k-1$) at the corresponding rows. Additional multiplexers in each basic cell, denoted with L and R in Figure 5, 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 4, 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 6.

The proposed architecture operates with the fixed number of folding sets, where the number of folding sets is equal to the number of coefficients, but can be reconfigured for operation with different number of operations inside folding sets. 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.

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 m. The number of basic cells is reduced to the number of basic cells in only 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 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 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 finding suitable area-time tradeoffs are provided for the bit-plane architecture through the application of folding technique with changeable folding factor.

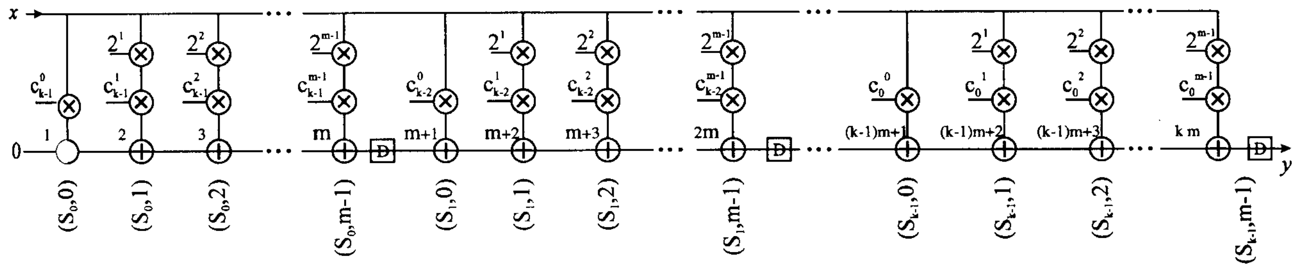


Figure 3. Transformed DFG that enables the application of folding technique

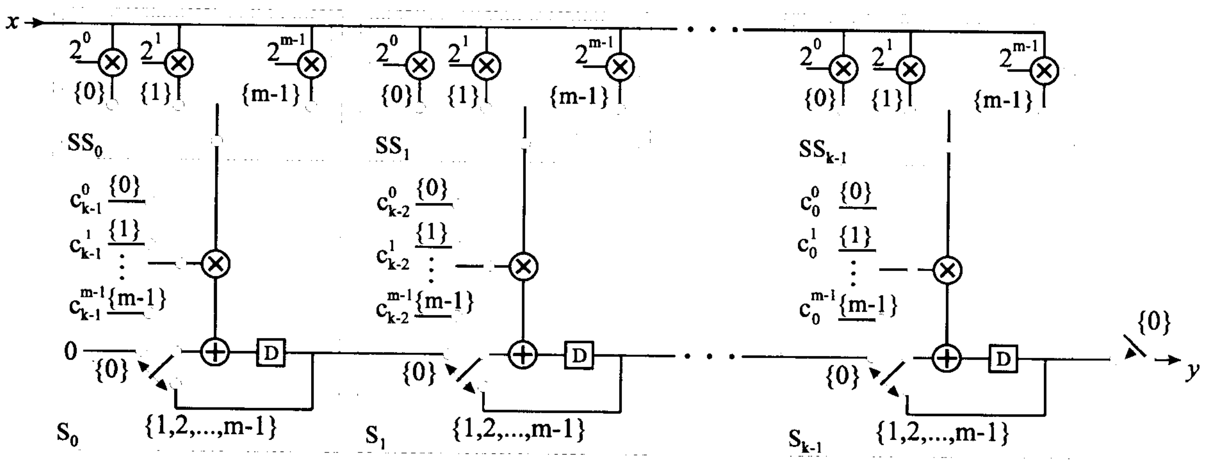


Figure 4. Folded architecture with folding sets S_0, S_1, ..., S_{k-1}

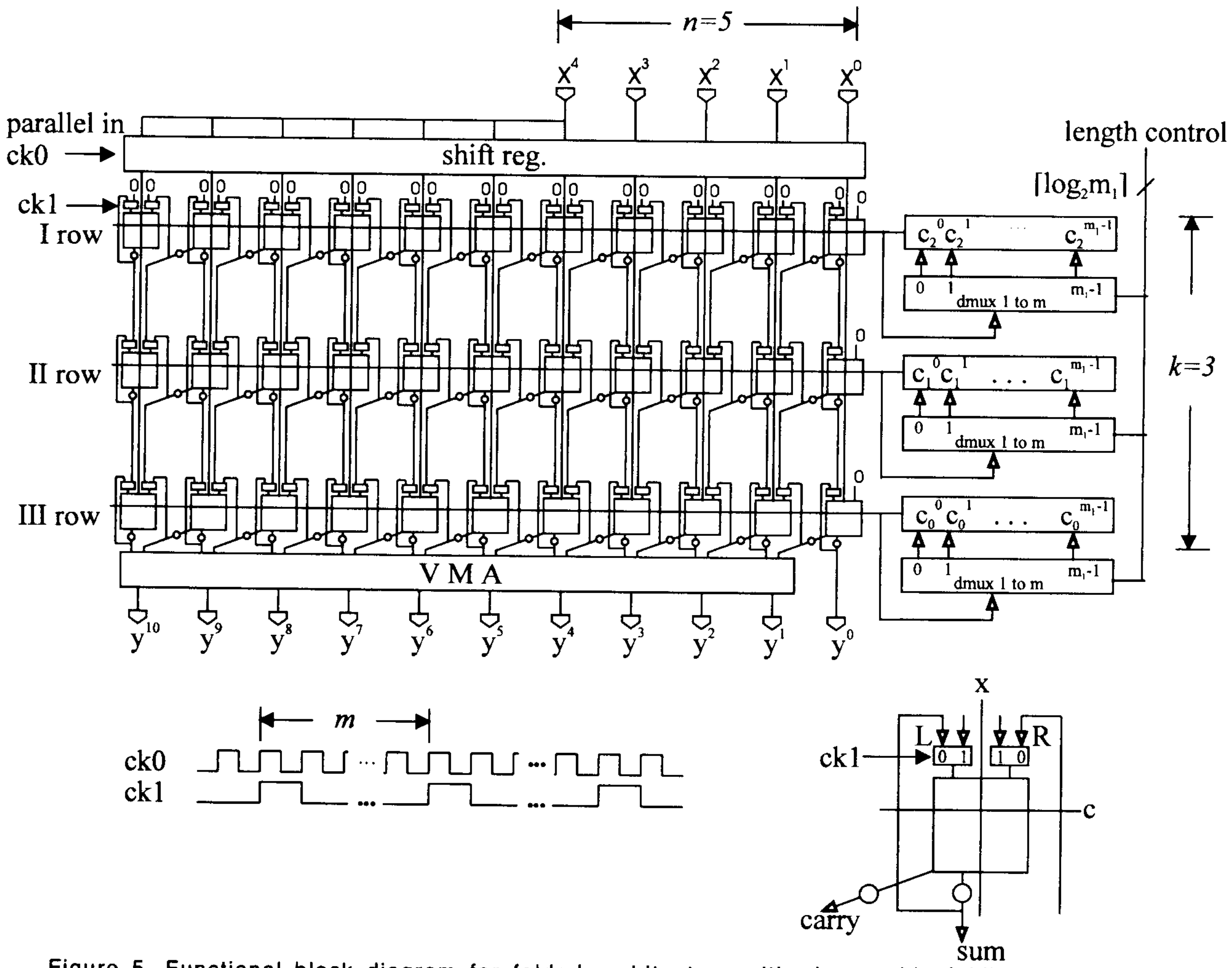

n=5
X^4 X^3 X^2 X^1 X^0
parallel in
ck0
shift reg.
ck1
I row
II row
III row
length control
$\lceil \log_2 m_1 \rceil$
k=3
dmux 1 to m
VMA
y^{10} y^9 y^8 y^7 y^6 y^5 y^4 y^3 y^2 y^1 y^0
m
ck0
ck1
x
L
R
c
carry
sum

Figure 5. Functional block diagram for folded architecture with changeable folding factor (*k=3*)

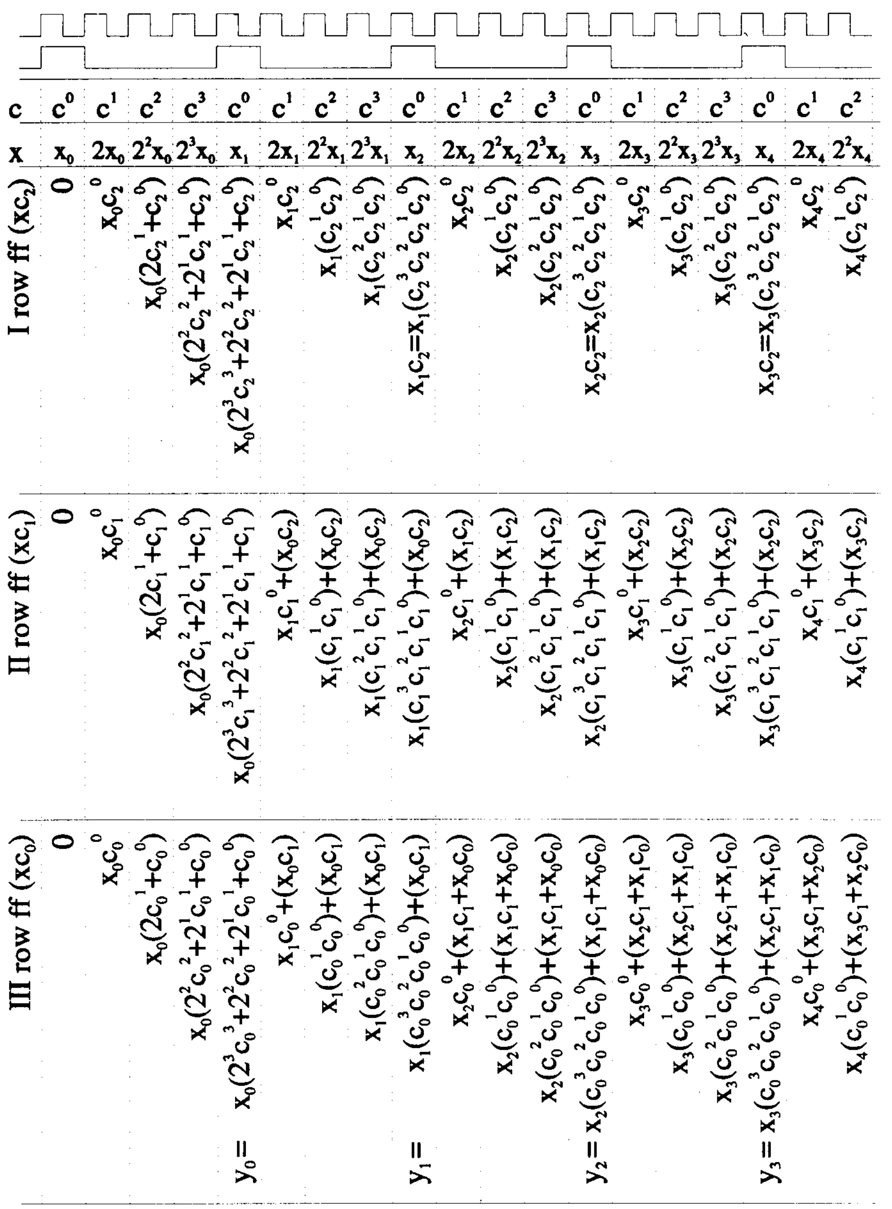


Figure 6. Data flow for folded architecture (*k=3, m=4*)

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 Transaction 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] K. K. Parhi, *VLSI Digital Signal Processing Systems (Design and Implementation)*, John Wiley & Sons, In., New York, 2000.

[7] 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.

[8] T. Noll, "Semi-systolic Maximum Rate Transversal Filters with Programmable coefficients", *Workshop of Systolic Architectures*, Oxford, 1986, pp. 103-112.

[9] 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.