Flexible Folded FIR Filter Architecture

I. Milentijevic, V. Ciric, O. Vojinovic

Abstract - Configurable folded bit-plane architecture for FIR filtering that allows programming of both number of taps and coefficient length is proposed in this paper. Proposed architecture allows designing of flexible folded FIR filter array with fixed size that enables efficient implementation of different wireless standards on single filter. This paper deals with the mapping of unfolded data flow graph onto the configurable folded system using a new folding set assignment. The obtained architecture as a folded system is described by data flow graph, functional block diagram and data flow diagram.

I. Introduction

Cellular-phone technology is changing rapidly. There is an increasing number of wireless-communications standards, including variants of the IEEE 802.11 wireless LAN specification, code-division multiple access, the global system for mobile communications, and emerging third-generation technologies. Traditionally, devices need a separate chip to work with each standard. However, as wireless technologies mature, service providers differentiate themselves by offering new features, such as multimedia capabilities. Providing each feature typically requires a separate chip, or essence, multiple circuitry systems physically joined on a peace of silicon [1]. The additional circuitry adds cost, takes up space, increases power usage in mobile devices, and increase product-design time. This problem can be solved using adaptive approach. With this approach, software can redraw a chip's physical circuitry on the fly, letting a single processor to perform multiple functions [1].

Factors such as regularity, scalability ande ability to trade hardware for speed within the context of an architecture become more important [2]-[4]. The application of folding technique onto array type architectures for FIR filtering gives designers greater flexibility in finding the best trade-off between hardware size and throughput rate [5]. 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, well-known bit-plane architecture (BPA) [8] was used in [9]-[10]. 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.

I. Milentijevic, V. Ciric, O. Vojinovic are with the Department of Computer Science, Faculty of Electronic Engineering, University of Nis, Beogradska 14, 18000 Nis, Serbia and Montenegro, E-mail: {milentijevic,vciric,oliver}@elfak.ni.ac.yu

In this paper we propose flexible folded bit-plane architecture for FIR filtering that allows programming of both number of taps and coefficient length. As a starting point we use the transformed Data Flow Graph (DFG) for the BPA proposed in [9] and involve new assignment of folding sets [10]. This paper deals with the mapping of the transformed DFG for the BPA onto the configurable folded system where new assignment of folding sets is implemented.

The obtained architecture as a flexible folded system, will be described by DFG, functional block diagram and the data flow diagram. The method of operation and operations mapping onto the processing units will be described in detail. The folded processor array is fed with data by hardware module for input data entering and coefficient supply module. The algorithms for data reordering in both modules will be presented, too. The folded FIR filter architecture is described in VHDL as a parameterized FIR filter core. For the sake of the illustration of the architecture functionality and programming capabilities, few examples are implemented in FPGA technology. The results of implementation, concerning throughput and chip occupation, will be presented.

II. Operations mapping

The following notation provides the basis for further explanation of mapping of DFG for BPA onto the configurable system: m_c – coefficient length, k_c –number of coefficients, c_i^j – bit of coefficient c_i (with weight 2^j), N – folding factor, k – number of folding sets, n – length of input words x_i, L – total number of "operations" in the DFG, where one operation assumes forming of partial product and the addition performed on one "row" of basic cells (basic cell contains AND gate and full adder), p-position of *operation* within the DFG ($0 \le p \le L-1$), S_s–*s-th* folding set ($0 \le s \le k-1$).

Starting DFG, which is well prepared for application of folding technique is shown in Fig. 1. The mathematical description of folding sets assignment ($S_s|r$) is done according to the following equations

$$s = p \ mod \ k$$
$$r = p \ mod \ N. \quad (1)$$

The idea of mapping different operations onto the different hardware units according to the chosen number of coefficients and coefficient length in fixed array structure is

introduced with (1). The proposed mapping of operations enables both changing the number of coefficients and coefficient length including constraint L= $k_c m_c$= kN.

The hardware size reducing for factor N is provided in a respect to the architecture from Fig. 1.

Let us note that, the number of folding sets is not obligatory equal to the number of coefficients.

In order to obtain mapping dependencies between operations and DFG nodes, transformed DFG from Fig. 1 should be used. Each operation from DFG (Fig. 1) stands for multiplication of input data words by one coefficient bit. Assignment of position numbers to operations in DFG (p) is done as follows: the leftmost operation is denoted with 0, while the rightmost operation is denoted with *L-1* (Fig. 1).

Operation p ($0 \le p \le L-1$) from Fig. 1 performs multiplication of input data word by coefficient bit c_i^j. According to (1), folded architecture multiplies input data word (Fig. 1) by coefficient c_i^j on folding set s ($0 \le s \le k-1$) in time instance $\delta \cdot N+r$ ($0 \le r \le N-1$; $\delta=0,1,2,\ldots$). The operation that has position in DFG equal to p (Fig. 1), according to folding set assignment (1), can be described as

$$p = m_c\left(k_c - \left(j+1\right)\right) + i \,. \tag{2}$$

The dependency between folding set s and folding order r of coefficient bit c_i with weight 2^j, using (1) and (2), is obtained as:

$$\begin{aligned} s &= \left(m_c \cdot \left(k_c - \left(j+1\right)\right) + i\right) \bmod k \\ r &= \left(m_c \cdot \left(k_c - \left(j+1\right)\right) + i\right) \bmod N \end{aligned} \tag{3}$$

Expression (3) describes the folding set s that performs multiplication by coefficient c_i^j in time instances $\delta \cdot N+r$ ($0 \le r \le N-1$; $\delta=0,1,2,\ldots$).

III. Folded Bit-Plane Architecture

Using a new assignment of folding sets that is applied on the transformed DFG from Fig. 1, we obtain folded Bit-Plane architecture in general form (Fig. 2). Input Data Entering Module (IDEM), denoted with dashed lines in Fig. 2, provides input data for the folded architecture in accordance with folding set assignment (1).

Sections $S_0,S_1,\ldots,S_{k-1}$ in Fig. 2 are Processing Elements (PE) of the folded architecture. Each section is devoted to computations from the corresponding folding set. Sections are implemented as rows of basic cells in functional block diagram, where the basic cell is comprised of AND gate and full adder. Functional block diagram for obtained folded FIR filter architecture with changeable number of coefficients and coefficient length is shown in Fig. 3 for case k=3, N=4, k_c=2 and m_c=6.

Let us note that the ordering of coefficient depends on number of coefficients, k_c, and coefficient length m_c. The Coefficient Bit Supply Module (CBSM) from Fig. 3 provides the proper ordering of coefficient bits regardless to coefficient number and length. The internal structure of CBSM from Fig. 3 is given in Fig. 4.

In the respect to operations mapping (Eq. 3), CBRM has two operational modes. First, initialization mode, when coefficient bits are entered into the CBRM, and the second, run mode, when CBRM is feeding array with coefficient bits.

The CBSM is implemented as two-dimensional array of latches where each latch stores one bit of the coefficient (Fig. 4). The number of rows is equal to the number of folding sets in FBSM (k) while the number of columns is equal to the FBSM's folding factor (N). Output from each row is feeding one folding set of the folded array with coefficient bits. Rows are implemented as shift registers, so during the run mode coefficients rotate through the rows from right to left, feeding each folding set of folded array with coefficient bits in correct order (solid arrows in Fig. 4).

Problem of providing the correct bit order, during initialization mode, can be solved using the property of modulo dependence in (3). Due, the trace of the first coefficient bit $c_{k_c-1}^0$ can be described with mapping of time instances $t \in \{1,2,\ldots,k \cdot N\}$ onto the array position $[\alpha,\beta]$:

$$\alpha=((t-1) \bmod k)+1$$

$$\beta=((t-1) \bmod N)+1.$$

The number of clock cycles, required for initializing the structure, is $k \cdot N$.

IV. Functional Description

Folding sets $S_0,S_1,\ldots,S_{k-1}$ are shown in dashed boxes (Fig. 2). Each folding set contains N operations. In order to clarify the folded FIR filter method of operation, the hardware section that performs the operation from set S_S ($s=0,1,\ldots,k-1$) is also denoted with S_S. Initially, the computation starts in folding set S_0 where the product $2^0 \cdot c_{k_c-1}{}^0 \cdot x_0$ is obtained in the first clock cycle. In the next clock cycle folding set S_1 generates the partial product $2^1 \cdot c_{k_c-1}{}^1 \cdot x_0$ adding previously computed partial product from S_0. Thus, the value $(2^0 \cdot c_{k_c-1}{}^0 \cdot x_0)+(2^1 \cdot c_{k_c-1}{}^1 \cdot x_0)$ is

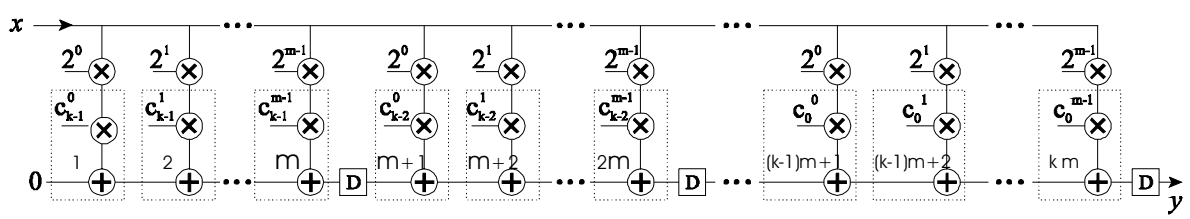

Fig. 1. Transformed DFG

entered into the next section, which performs the operations from S_2, in the third clock cycle. The next important time instance is (k+1)st clock cycle. In that clock cycle both input data path and summation path are folded from section S_{k-1} to S_0. In input data path $2^k \cdot x_0$ is present at input of the section S_0, while in the summation path $(2^0 \cdot c_{k_c-1}{}^0 \cdot x_0)+(2^1 \cdot c_{k_c-1}{}^1 \cdot x_0)+...+(2^{k-1} \cdot c_{k_c-1}{}^{k-1} \cdot x_0)$ enters the same section. S_0 adds $2^k \cdot c_{k_c-1}{}^k \cdot x_0$ to the entered sum. But the computation for the coefficient c_{k_c-1} is not finished yet.

The complete product $c_{k_c-1}x_0$ is obtained in the section $S_{(m_c-1) \bmod k}$ during clock cycle m_c. The computation of $c_{k_c-2}x_1$ starts in (m_c+1)st clock cycle. The section $S_{m_c \bmod k}$ computes

$$\{(2^0 \cdot c_{k_c-1}{}^0 x_0)+(2^1 \cdot c_{k_c-1}{}^1 \cdot x_0)+...+(2^{m_c-1} \cdot c_{k_c-1}{}^{m_c-1} \cdot x_0)\}+ +2^0 \cdot c_{k_c-2}{}^0 \cdot x_1 =$$

$$= (c_{k_c-1}x_0)+(2^{m_c-1} \cdot c_{k_c-2}{}^0 \cdot x_1).$$

The first completely generated result at output is y_0 with latency $2m_c$-N clock cycles. New result y is generated every N clock cycles. The data flow example that illustrates described process of computation is shown in Fig. 5.

The proposed architecture supports the operation with changeable number of coefficients and coefficient length.

V. Implementation

The implementation is done onto the FPGA Spartan II s200-5pq208 with aim to illustrate what filtering can be carried out onto the one folded programmable architecture. The Table I illustrates the abilities of one implemented architecture with n=8, k=8, N_{max}=16 and y=27 (y stands for implemented length of output word).

Table I gives the implementation results for clock period, throughput and initial latency for possible programmed values of k_c and m_c taking into account chosen folding factor N ($N_{max} \geq N \geq 1$). Using the data from Table I, the graphical representation in Fig. 6 that describes the throughput as a function of chosen folding factor, is generated. Increasing of throughput is achieved by decreasing of folding factor. Table I also contains values for initial latency as it is given by $2m_c$-N+$Adder_{Latency}$.

The initial latency depends also on adder's latency. In the implemented example adder's pipeline stages is equal to adder's length (y=27*)*. The number of adder's pipeline stages remains the same regardless to the programmed values for k_c and m_c.

TABLE I
POSSIBILITIES FOR CONFIGURATION OF THE ARRAY

k	N	k_c	m_c	Used slices	Clock [MHz]	Throughput [clk]	Throughput [MHz]	Initial latency [clk]	Initial latency [ns]
8	16	2	64	642 [25%]	86.69	16	12.28	139	707.51
		4	32	642[25%]	86.69	16	12.28	75	381.75
8	8	2	32	642[25%]	86.69	8	24.56	83	422.47
		4	16	642[25%]	86.69	8	24.56	51	259.59
8	4	2	16	642[25%]	86.69	4	49.12	55	279.95
		4	8	642[25%]	86.69	4	49.12	39	198.51

VI. Discussion and Conclusions

The transformation of source DFG for the bit-plane architecture and proposed assignment of folding sets enable the synthesis of fully pipelined folded FIR filter architecture with changeable number of coefficients, changeable coefficient length, and adjustable 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. FPGA implementation of proposed architecture proved the functionality of the architecture and showed liner dependency of throughput as a function of folding factor in fixed size arrays. tradeoffs between throughput and occupation of on-chip resources as well as to illustrate configuration capabilities of the folded architecture. Proposed architecture allows designing of flexible folded FIR filter array with fixed size that enables efficient implementation of different wireless standards on single filter.

References

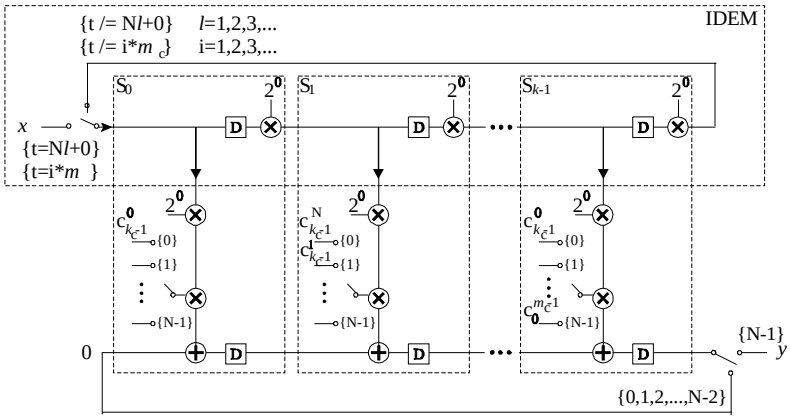

Fig. 2. Folded FIR filter architecture with changeable number of coefficients and coefficient length

[1] L. Paulson, L. Garber, “Reconfiguring Wireless Phones with Adaptive Chips”, IEEE Computer, Vol. 36, Number 9, September 2003, pp. 9-11.

[2] 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.

[3] 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

[4] Robert Hawley, Bennett Wong, Thu-ji Lin, Joe Laskowski, Henry Samueli, “Design Techniques for Silicon Compiler Implementations of High-Speed FIR Digital Filters”, *IEEE Journal of Solid-State Circuits*, Vol 31, No. 5, May 1996.

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

[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] 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.

[9] I. Milentijevic, V. Ciric, O. Vojinovic, T. Tokic, “Folded Semi-Systolic FIR Filter Architecture With Changeable Folding Factor”, *Neural, Parallel & Scientific Computations, Dynamic Publishers*, Atlanta, Vol.10, No2,2002, pp.235-247.

[10] I. Milentijevic, V. Ciric, “Assignment of Folding Sets for Adaptive FIR Filtering on Folded Array”, Proceedings of the WPS-DSD 2003, 29th EUROMICRO Conference, Belek,

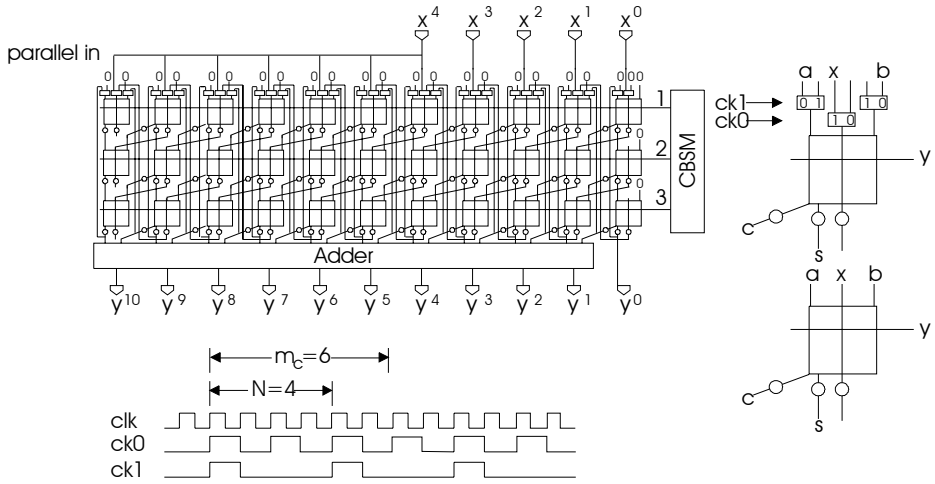

Fig. 3. Functional block diagram

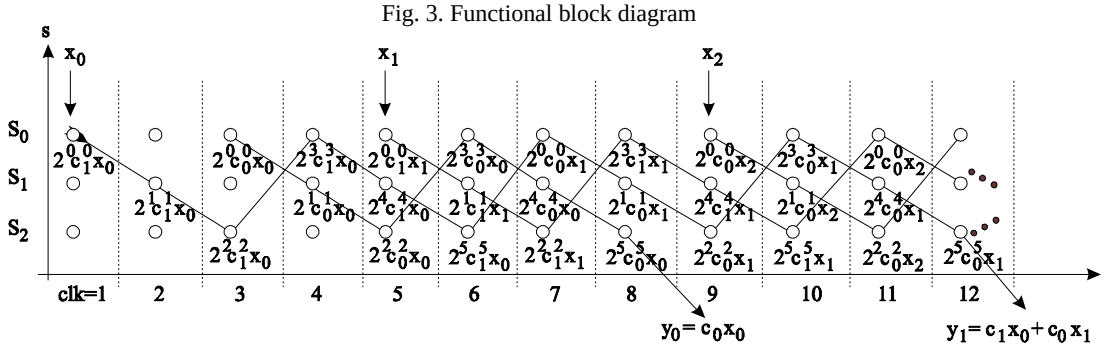

Fig. 5. Data flow for folded architecture (k=3, N=4, k_c =2 and m_c=6)

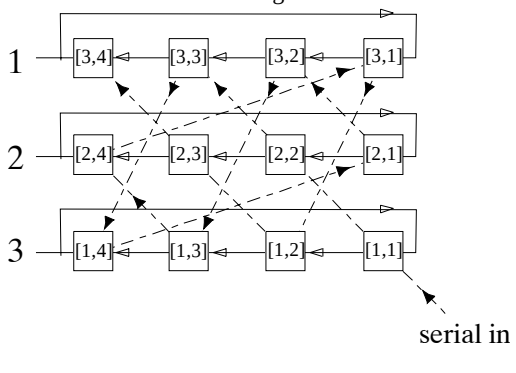

Fig. 4. Coefficient Bit Supply Module – CBSM

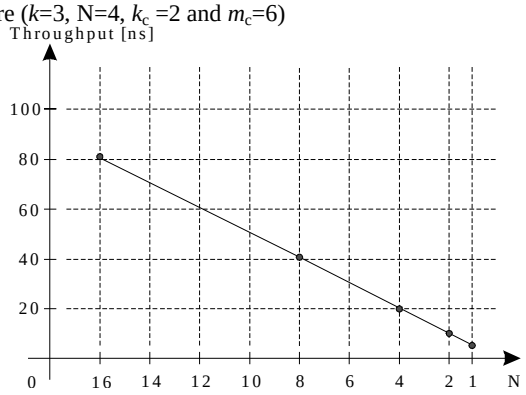

Fig. 6. Throughput as a function of chosen folding

Turkey, September 2003. pp. 21-22.