Family of Folded Bit-Serial Multipliers

Vladimir M. Ciric [1], Ivan Z. Milentijevic [2], Oliver M. Vojinovic [3], Teufik I. Tokic [4]

Abstract – **The synthesis of new family of folded bit-serial multipliers for integer multiplication is presented in this paper. Folding technique is applied to serial-parallel serial multiplier architecture. The resulting architecture can operate with operands of arbitrary length. In order to illustrate functionality of proposed architecture the preliminary results of FPGA implementation are given.**

Keywords – **systolic arrays, array multipliers, folding technique.**

I. INTRODUCTION

In several cases, it is senseless to use a bit-parallel circuit: it has an important cost in area and runs faster than the throughput required by the application. In these cases, the bit-serial approach became an important alternative for efficient implementation of custom Digital Signal Processing (DSP) circuits [1].

Modern VLSI technology allows integrating massive parallel systems on a single chip. The area limitations for the processors of such systems require small but efficient computational units. Particularly, in public-key cryptography special features are required for multiplier units. In RSA encryption and decryption, large integers (typically 1024 bits or more) must be multiplied, and in elliptic curve cryptosystems, a multiplication in finite fields is required. In contrast to cryptography, operands in signal processing applications are usually short, e.g. 8-bit pixels [2]. A parameterized family of folded bit-serial multipliers that covers all these applications is presented in this paper.

As a source architecture, which provides a starting point in the synthesis procedure, standard bit-serial multiplier is chosen. The multiplier is characterized with serial input; parallel bit-level processing, and serial output - SPS multiplier [3].

This paper is organized as follows: Section 2 gives a brief review of folding technique. Basic SPS multiplier architecture is described in Section 3. In Section 4, the synthesis procedure of folded bit-serial multipliers family is given. Section 5 covers the functional description of folded architecture while in Section 6 we give preliminary results of FPGA implementation. Section 7 is concerned with discussion and conclusions.

[1] Vladimir M. Ciric is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia and Montenegro, E-mail: vciric@elfak.ni.ac.yu

[2] Ivan Z. Milentijevic is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia and Montenegro, E-mail: milentijevic@elfak.ni.ac.yu

[3] Oliver M. Vojinovic is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia and Montenegro, E-mail: oliver@elfak.ni.ac.yu

[4] Teufik I. Tokic is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia and Montenegro, E-mail: fika@elfak.ni.ac.yu

II. FOLDING TECHNIQUE

The folding technique is introduced by K.K. Parhi and described in [4, 5]. With aim to clarify the technique of applying of folding technique, 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 that 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 $N - 1 \geq u, v \geq 0$.

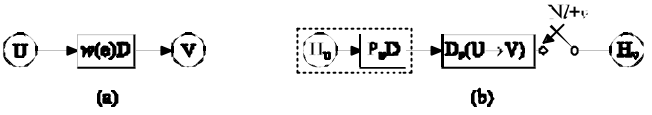

Fig. 1. The synthesis of folded data path: (a) An edge $U \rightarrow V$ with $w(e)$ delays; (b) The corresponding folded data path

A folding set, S, is defined as an ordered set of operations, which contains N entries, executed by the same functional unit. For a folded system to be realizable $D_F(U \rightarrow V) \geq 0$ must hold for all of the edges in the DFG. Once valid folding sets have been assigned, if this property in not satisfied, retiming technique has to be used.

Since the folding technique will be applied to SPS multiplier, the short description of SPSM architecture is given in the next section.

III. THE BASIC MULTIPLIER

The Data Flow Graph (DFG) that represents the SPS multiplier is given in Fig. 2. Processing elements (PEs) are denoted with dashed lines. SPS multiplier consists of a_{La} PEs,

while *La* represents the length of operand *a*. Each PE is marked with *i* (*0£i£La-1*) in bottom-left corner and consists one AND gate and one full adder (Fig. 2). Before the start of multiplication, operand (*a*) has to be available at parallel input a_{La-1}, a_{La-2},..., a_0, with the LSB at left first position, bearing in mind that the length of operand *a* is L_a. Second operand *b*, which has a length L_b, is entered into the multiplier in LSB-first manner. In each clock cycle *i* one bit b_i of operand *b* is processed. It is multiplied by *a* using AND operation simultaneously in all PEs. The result is added to the current sum, and new intermediate result is computed.

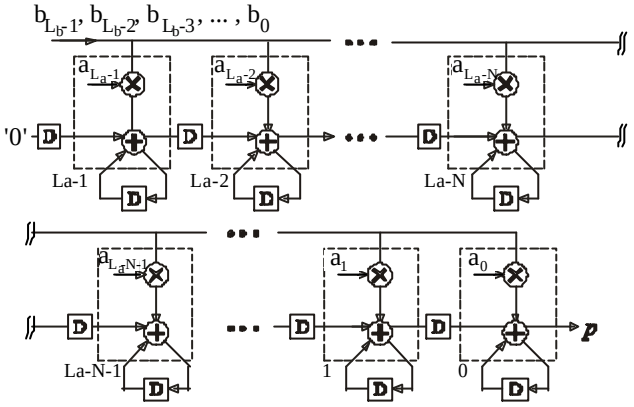

Fig. 2. DFG representation of Serial-Parallel Serial Multiplier

Since the carry save arithmetic is implemented intermediate result is available in two binary words: *s*-word and *c*-word.

The next bit b_{i+1} of *b* is processed in the next clock cycle. Since the intermediate result has to be multiplied by 2, the intermediate is shifted for one position to the right before the new product is added. Shift operation is preformed by propagation of the sum bits to the next right neighbor. Carry bit loops provide the processing of carries in the same processing element. The shift operation gives one resulting bit at the output of the rightmost full adder. Thus the *i-th* bit of the final product is obtained.

Introduced architecture is highly regular and contains broadcast line. It is a very suitable architecture for application of folding technique.

IV. SYNTHESIS OF FOLDED ARCHITECTURE

In order to provide the reducing of chip area and to enable the changing of operand's length in fixed folded multiplier architecture we propose a new mapping of operations onto the DFG nodes (processing elements). Let us note that *N* will stand for folding factor while *k* for the number of processing elements in the folded multiplier.

The multiplier described in the previous section multiplies two unsigned integers. With aim to keep the functionality of architecture and to involve features that folding technique brings, we start with application of folding technique, in general form, with assigning of folding sets on the basic architecture.

Folded architecture should consist of *k* PEs, so the number of involved folded sets has to be equal to *k*. Total number of multiplication operations per clock cycle in basic architecture is *La*, so the folding factor can be computed as:

$$N = La/k. \quad (2)$$

The assignment of folding sets on DFG from Fig 2 is done according to following rules:

1. First N operations starting from the leftmost PE belong to folding set S_{k-1};
2. Operation denoted with *La-1* in Fig. 2 is the first operation in the folding set S_{k-1} ($S_{k-1}|N-1$);
3. Operation denoted with *La-2* is the second operation in the folding set S_{k-1} ($S_{k-1}|N-2$);
4. N-th operation, counted from the left, is the last operation in folding set S_{k-1} ($S_{k-1}|0$);
5. (N+1)-st operation belongs to folding set S_{k-2} ($S_{k-2}|N-1$);
6. The rightmost operation belongs to folding set S_0 ($S_0|0$).

As the basic architecture (Fig. 2) is highly regular, for results of folding Eq. (1) we differ two cases. The first case occurs when folding Eq. (1) is computed for two neighboring nodes that are folded onto one node while the second case is for neighboring nodes that are folded onto different nodes. Thus, for neighboring nodes in basic architecture denoted with *i and i+1*, *0£i£La-2* (Fig. 2), we should differ two groups of equations. First is whenever *i* is equal to *N-1, 2N-1, ... , L_{a-N-1}* and the second is otherwise. In the first case, folded equation (1) can be computed as follows:

$$D_f(i \to i+1) = N \cdot 1 - 0 + 0 - 1 = N-1,\ i = N-1, 2N-1, \dots, L_{a-N-1}. \quad (3)$$

Folding equations for all other nodes are computed as:

$$D_f(i \to i+1) = N \cdot 1 - 0 + (N-1) - 0 = 2N-1,\ i \neq N-1, 2N-1, \dots, L_{a-N-1}. \quad (4)$$

Folding equation explains how many clock cycles the result should be stored before the next using. In order to save required number of latches, the following analysis has to be done.

If operations *i* and *i+1*, from the source architecture, are folded to the same node *j* in the folded architecture (*0£j£k-1*), data produced by operation *i* have to be stored for N-1 clock cycles (Eq. 3) before they are consumed by operation *i+1* in the folded architecture. If operations *i* and *i+1* from the source architecture are folded onto neighbor nodes, *j* and *j+1* in folded architecture, data produced by operation *i* have to be stored for 2N-1 clock cycles before they are used (Eq. 4). According to (Eq. 3) and (Eq. 4), folded architecture contains *max(N-1,2N-1)=2N-1* latches between nodes *j* and *j+1* that will be used for data buffering. Data from the latch N-1, counted left to right from node *j*, have to be recycled again to the node *j* (Eq. 4).

As the previous analysis is concerned with storing of sum bits in the folded architecture, we have to provide the corresponding analysis for the storing of carry bits.

In the basic architecture, carry bits within every PE are recycled in the same node. Thus, only one folding equation has to be computed for all carry data paths:

$$D_f(i \to i) = N \cdot 1 - 0 + 0 - 1 = N-1,\ i = 0, 1, \dots, L_{a-1} \quad (5)$$

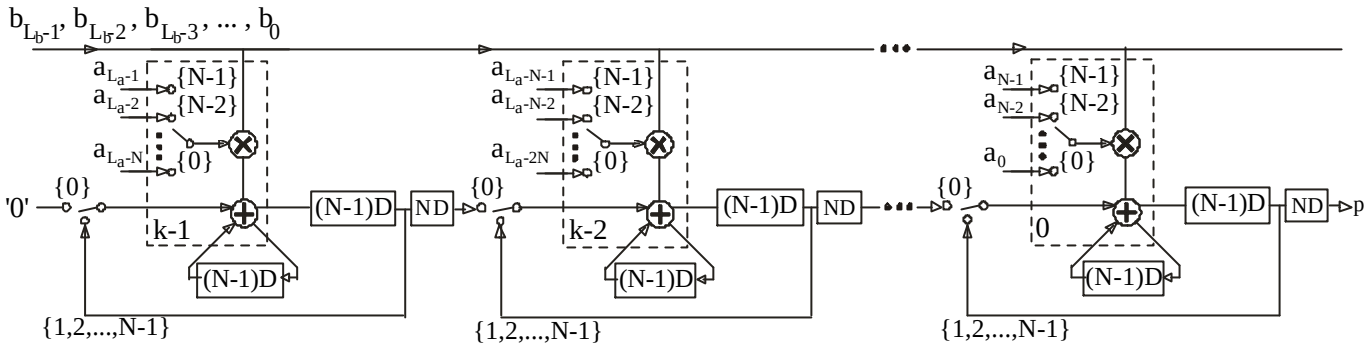

Fig 3. Family of Folded Bit-Serial Multipliers

Equation 6) gives the number of latches for carry bits storage in the carry-recycling data path of the folded architecture.

Analysis given in this section is done in general, so the final architecture is synthesized as parameterized family of Folded Bit-Serial Multipliers - FBSM (Fig. 3).

For the design of specific folded architecture the values for *k* and *N* can be choused in the parameterized family of FBSM. It provides the finding of optimal area-time solution for the given requirements.

Next section gives the description of data flow within the folded architecture.

V. FUNCTIONAL DESCRIPTION

The family of Folded Bit-Serial Multipliers (FBSM) is given in Fig. 3. FBSM consists of *k* PEs that are denoted with dashed boxes and marked as *k-1*, *k-2*, …, *0*. Each of PEs is connected to its right neighbor via buffer that has (*2N-1*) delays. Data from (N-1)-st letch are feedback to source PE. Carry output from each PE is recycled to the carry input of the same PE via buffer with N-1 delays. Data output, recycled from PE, is switched again to the same PE only within time instances *{1,2,…,N-1}*. Within time instance *{0}* each PE collects data from its left neighbor's buffer.

Before the start of multiplication, operand *a* has to be available at parallel input a_{La-1}, a_{La-2},…, a_0 as it is shown in Fig. 3. The second operand *b* is entered into the multiplier in LSB first manner. In the first clock cycle bit b_0 of operand *b* is processed. It is multiplied with *k* bits of operand $a - a_{La-N}$, a_{La-2N},…, a_0, using AND operation simultaneously and the result is moved to the right - into buffers. In the second clock cycle bit b_0 is processed again. It is bit-wise AND, multiplying b_0 with next *k* bits of operand a - a_{La-N+1}, $a_{La-2N+1}$,…, a_1. The result is moved to the right into the buffers again. The result produced in the first clock cycle will be recycled to the input of the source PE passing through buffer of N-1 delays. Intermediate result produced in the same clock cycle will be added to the intermediate result computed in (2N-1)-st clock cycle and shifted for one position right after the passing thought 2N-1 latches.

The corresponding data flow for FBSM with *k=2, N=2,L_a=4,L_b=4*, which is shown in Fig. 4, is given in Table I. Column *Dn.m* of Table I contains the data, stored in *m - th* letch (from left to right) starting from PE *n*, for first 6 clock cycles. Clock cycles in which the result bit is present on output (D0.3) are denoted with (*) – 2, 4 and 6.

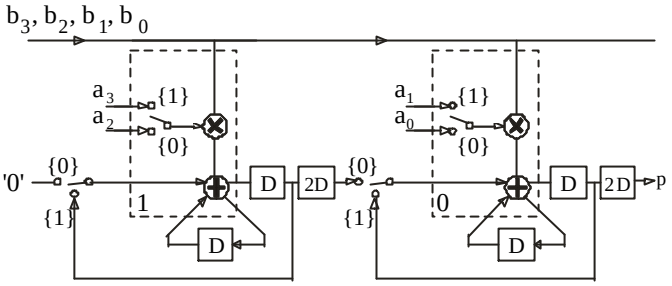

Fig 4. FBSM architecture (*k=2, N=2,L_a=4,L_b=4*)

TABLE I.
FBSM DATA FLOW FOR *k=2, N=2,L_a=3,L_b=3*

time inst.	PE1			PE0		
	D1.1	D1.2	D1.3	D0.1	D0.2	D0.3
0 {0}	a_2b_0	0	0	a_0b_0	0	0
1 {1}	a_3b_0	a_2b_0	0	a_1b_0	a_0b_0	0
2 {0}*	a_2b_1+ a_3b_0	a_3b_0	a_2b_0	a_0b_1+ a_1b_0	a_1b_0	a_0b_0
3 {1}	a_3b_1	a_2b_1+ a_3b	a_3b_0	a_1b_1+ a_2b_0	a_0b_1+ a_1b_0	a_1b_0
4 {0}*	a_2b_2+ a_3b_1	a_2b_2+ a_3b	a_2b_1+ a_3b_0	a_0b_2+ a_1b_1+ a_2b_0	a_1b_1+ a_2b_0	a_0b_1+ a_1b_0
5 {1}	a_3b_2	a_2b_2+ a_3b	a_2b_2+ a_3b_1	a_1b_2+ a_2b_1+ a_3b_0	a_0b_2+ a_1b_1+ a_2b_0	a_1b_1+ a_2b_0
6 {0}*	a_2b_3+ a_3b_2	a_3b_2	a_2b_2+ a_3b_1	a_0b_3+ a_1b_2+ a_2b_1+ a_3b_0	a_1b_2+ a_2b_1+ a_3b_0	a_0b_2+ a_1b_1+ a_2b_0

FBSM's initial latency is *2N-1* clock cycles. One bit of the final products is produced every *N* clock cycles.

The throughput of synthesized architecture is reduced for *N* times. For the sake of illustration of operating speed as well as chip occupation, comparative results of FPGA implementation for basic and folded architecture, are presented in the next section.

VI. IMPLEMENTATION

Each PE of basic multiplier requires two associated latches (Fig. 2), and for the folded architecture, each PE requires (N-1)+(N-1)+N=3N-2 associated latches. It the folded architecture 3N-2 latches per PE are grouped in three separated entities that form a simple shift register.

The Xilinx's Spartan-II function generators are implemented as 4-input look-up tables (LUTs). In addition to operating as a function generator, each LUT can provide a 16 x 1-bit synchronous RAM. The Spartan-II LUT can also provide a 16-bit shift register. Basic building block of the Spartan-II CLB is the logic cell (LC). Each Spartan-II CLB contains four LCs, organized in two similar slices. Thus, Spartan-II can provide two 16-bit shift-registers (32-bit shift register) per slice for implementation of proposed folded architecture. The equation that gives the number of required slices for folded architecture per PE, keeping in mind that number of available latches in Spartan II per slice is 32, is:

$$PE = \left\lceil \frac{3 \cdot N - 2}{32} \right\rceil \quad (5)$$

Concerning routing of wires between PEs it is hardly possible to force implementation of more than one PE of basic architecture per slice. However, involving optimization efforts as well as using of mentioned shifting property of Spartan II, one PE of folded architecture can be fineted at one slice of Spartan II. It is possible for folding factor less then 11 (Eq. 5). Chip occupation for both basic BSM and folded BSM according to (Eq. 5) are given in Table II.

TABLE II
CHIP OCCUPATION AND CLOCK PERIODS - SPARTAN II XC2S2000-5PQ208

Op. length	Basic BSM			Folded BSM			
	No. of PEs	Slices used	Clock period [ns]	Folding factor	No. of PEs	Slices used	Clock period [ns]
88	8	8	3.957	1	8	8	4.088
				2	4	4	4.105
				4	2	2	3.803
116	16	16	4.706	2	8	8	4.898
				4	4	4	4.785
				8	2	2	4.502
332	32	32	4.580	4	8	8	4.590
				8	4	4	4.214
				16	2	4	4.367
64	64	64	6.682	8	8	8	7.211
				16	4	8	7.202
				32	2	6	7.003
128	128	128	8.129	16	8	16	8.056
				32	4	12	7.855
				64	2	12	7.841

Table II also contains clock periods for both architectures. Clock periods are obtained by implementing these architectures on Xilinx's Spartan II xc2s2000-5pq208.

VII. DISCUSSION AND CONCLUSIONS

The synthesis of family of folded bit-serial multipliers is presented in this paper. For the design of specific folded architecture the number of processing elements and folding factor can be choused in the parameterized family of FBSM. It provides the finding of optimal area-time solution for the given requirements.

Saprtan II "shift register" property was used to relax the constraints caused by relatively large number of lathes in folded architecture.

Generated architecture has kept almost all desirable features of source SPSM architecture. The hardware reduction of active arithmetic elements for the factor N is done at the cost of execution time.

VIII. REFERENCES

[1] J. Valls, M. Martinez-Peiro, T. Sansaloni, and E. Boemo, *"Fast FPGA-Based Pipelined Digit-Serial/Parallel Multipliers"*, Proc. 1999 IEEE ISCAS (Int. Symp. on Circuits and Systems), Volumen I, pp.482-485, Orlando, Florida, May 1999.

[2] M. Schimmler, B. Schmidt, H.W. Lang, S. Heithecker, *"An Area-Efficient Bit-Serial Integer Multiplier"*, H.R. Arabnia, L.T. Yang (Eds.), Proceedings of the International Conference on VLSI, Las Vegas, Nevada, USA, CSREA Press, pp. 131-137 (2003)

[3] P. Denyer and D. Renshaw, *"VLSI SIGNAL PROCESSING: A Bit-Serial Approach"*, Addison-Wesley, 1985.

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

[5] 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.