Efficient Dedicated Multiplication Blocks for 2´s Complement Radix-2m Array Multipliers

Leandro Z. Pieper, Eduardo A. C. da Costa, Sérgio J. M. de Almeida
Universidade Católica de Pelotas (UCPel), Pelotas, Brazil
leandrozaf@gmail.com, {ecosta,smelo}@ucpel.tche.br

Sergio Bampi
Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil
bampi@inf.ufrgs.br

José C. Monteiro
TU Lisbon / Instituto Superior Técnico – IST / INESC-ID, Lisboa, Portugal
jcm@inesc-id.pt

***Abstract* - In this paper, we introduce new dedicated blocks for radix-2^m multiplication. These blocks are basic components of the structure of the 2´s complement radix-2^m array multiplier previously proposed in the literature. In the original array multiplier, the blocks that perform the radix-2^m multiplication were automatically synthesized from a truth table. The dedicated multiplication blocks we propose are themselves composed of a structure of less complex multiplication blocks and resort to efficient Carry Save adders (CSA). This new scheme can be naturally extended for different radices multiplication. We present results of area, delay and power consumption for 16, 32 and 64 bit array multipliers using the new dedicated modules. The results show that by using the new dedicated modules, the array multipliers are more efficient in terms of delay and power consumption when compared both against the original array structure and the Modified Booth multiplier**.

***Index Terms*— array multiplier, radix-2^m multiplication, dedicated multiplication blocks, power consumption reduction.**

I. INTRODUCTION

Multiplier modules are common to many Digital Signal Processing (DSP) applications. The fastest types of multipliers are parallel multipliers. Among these, Wallace multipliers [1] are among the fastest. However, when regularity, high-performance and low power are primary concerns, Booth multipliers tend to be the primary choice [2], [3]. The Modified Booth algorithm achieves a major performance improvement through radix-4 encoding. This is particularly true for operands using 16-bit or more [4]. The radix-2^m binary array multiplier presented in [5] follows the same strategy as the Booth architecture, the reduction of the partial product terms, while keeping the regularity of an array multiplier. In this multiplier, the radix-2^m multiplication is performed by a set of dedicated m-bit multiplication blocks.

For the array multiplier of [5], the radix-4 blocks (m=2) are easily obtained. However, for radices higher than 4, the more complex dedicated blocks are obtained by the automatic synthesis in the SIS environment [6] from PLA (Programmable Logic Array) description. Since these blocks are replicated a number of times, that depends on the operand bit-width (W) and the group of bits (m), in the array multiplier circuit, it is important to reduce the complexity of these blocks.

We propose a new structure for the radix-2^m multiplication blocks. Namely, we present schemes for radices 8, 16, 32 ,64 and 256 multiplication blocks, that are composed by less complex radix-4 block and Carry Save (CSA) adder. The CSA is used in order to speed-up the carry propagation inside the multiplication modules. As will be presented, the radix-16 scheme can be naturally extended for different radices multiplication blocks [13].

We should stress further that, in contrast to the architectures presented in this work, raising the radix for the Booth architecture is a difficult task, thus not being able to leverage from the potential savings of higher radices. As shown previously in [5], radix-4 array multiplier can save power when compared against the radix-4 Modified Booth multiplier. Moreover, even radix-16 array multiplier can save more power than Modified Booth.. This is mainly due to the lower logic depth presented by the array multiplier, which can have a big impact on the amount of glitching in the circuit.

In this work we improve the multiplier architecture of [5] by using new dedicated multiplication blocks in the 16-bit wide architecture and then we extend the new schemes for 32 and 64-bit wide architectures. The results obtained show that delay and power consumption can be significantly reduced by using the new proposed blocks.

This paper is organized as follows. The next section presents the design methodology for the multipliers implementation. Section III makes an overview of relevant work related to our own, namely an overview of the Booth algorithm and a summary of the 2´s

doi:10.4304/jcp.5.10.1502-1509

complement radix-2^m binary array multiplier. The alternatives for the radices 8, 16, 32, 64 and 256 multiplication dedicated blocks are presented in Section IV. Performance comparisons are presented in Section V. Finally, in Section VI we conclude this paper, discussing the main contributions and future work.

.II. Design Methodology

We have applied carry save adders (CSA) in the dedicated multiplication blocks in order to speed-up the carry propagation in the addition operation. The basic idea of the CSA is that three numbers can be reduced to 2, in a 3:2 compressor, by doing the addition while keeping the carries and the sum separate [7], as shown in the example of the Fig. 1. As can be observed, there is no carry propagation within each CSA cell. It is only the recombination of the final carry and sum that requires a carry propagation addition. A carry save adder is very fast because it simply outputs the carry bits instead of propagating them to the left.

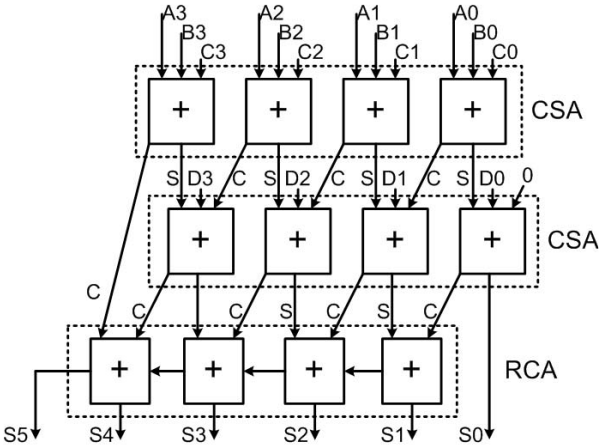

Figure 1. Example of a carry save structure.

For the composition of the dedicated multiplication block, we use the simpler radix-4 block, which is composed of only a few logic gates [5].

The design flow for the multipliers implementation is shown in Fig. 2.

.

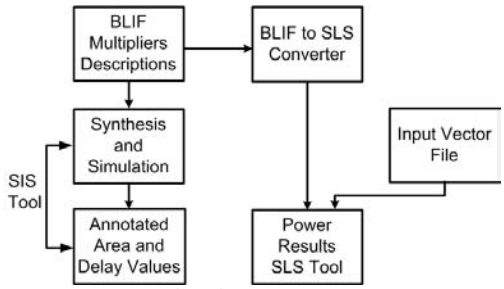

Figure 2. Design flow for the multipliers implementation.

The multipliers were all described in BLIF (Berkeley Logic Interchange Format). After verifying the functionality of the multipliers in SIS environment, area and delay values are annotated. Power values are obtained in SLS tool [8], a switch-level simulator, using the general delay model, after converting BLIF textual description to SLS format. A file containing a set of random vectors is used as input for the power consumption estimation.

Early multiplier schemes such as bi-section, Baugh-Wooley and Hwang [9] propose the implementation of a 2´s complement architecture, using repetitive modules with uniform interconnection patterns. However, some of these schemes do not permit an efficient VLSI realization due to the irregular tree-array form used.

III. Related Work

More regular and suitable multiplier designs based on the Booth recoding techniques have been proposed [10,11]. In the Modified Booth algorithm approximately half of the partial products that need to be added is used.

Although the Booth algorithm provides simplicity, it is sometimes difficult to design for higher radices due to the complexity to pre-compute an increasing number of multiples of the multiplicand within the multiplier unit. In [5] it is shown that the array multiplier can be more naturally extended for higher radices, using less logic levels and hence presenting much less spurious transitions. In this work it is proposed a new scheme for the dedicated radix-2^m blocks in order to improve the efficiency of the array multiplier of [5].

A. Booth Algorithm Overview

The radix-4 Booth's algorithm (also called Modified Booth) has been presented in [12]. In this architecture it is possible to reduce the number of partial products by encoding the 2´s complement multiplier. In the circuit the control signals (0, +MD, +2MD, -MD and -2MD) are generated from the multiplier operand for each group of 3-bit as shown in the example of Fig. 3 for a 8 bit wide operation. A multiplexer produces the partial product according to the encoded control signal. Note that the partial product terms are sign extended up.

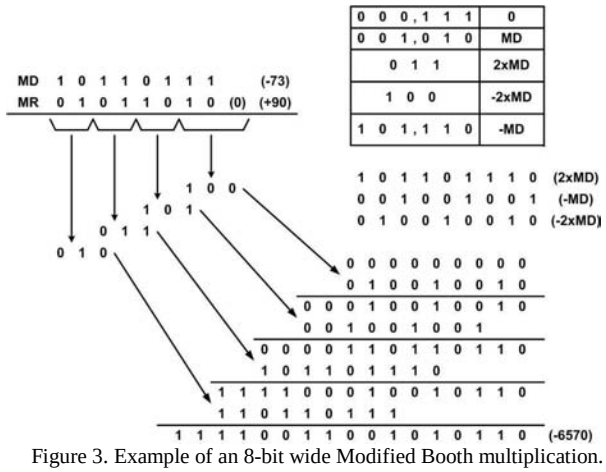

Figure 3. Example of an 8-bit wide Modified Booth multiplication.

To make the array more regular it is used a sign extension technique, where two extra bits in the partial product are used. It should be observed that the basic Booth cells are not simple adders as in our multipliers, but must perform addition-subtraction-no operation and

controlled left-shift of the bits on the multiplicand. The complexity presented by the cells makes it more difficult to increase its radix value in the Booth architecture.

B. Overview of 2's-Complement Radix-2^m Array Multiplier

In this section we summarize the methodology of [5] for the generation of regular structures for arithmetic operators using signed radix-2^m representation.

For the operation of a radix-2^m multiplication W-bit values, the operands are split into groups of m bits. Each of these groups can be seen as representing a digit in a radix-2^m. Hence, the radix-2^m multiplier architecture follows the basic multiplication operation of numbers represented in radix-2^m. The radix-2^m operation in 2's complement representation is given by (1) [5].

For the W-m least significant bits of the operands unsigned multiplication can be used. The partial product modules at the left and bottom of the array need to be different to handle the sign of the operands. A concrete diagram for W=8 bit wide operands using radix-16 (m=4) is presented in Fig. 4.

$$\begin{aligned} A \times B &= A' \times B' - A' b_{W-1} b_{\frac{W}{m}-1} 2^{W-m} \\ &\quad - a_{W-1} a_{\frac{W}{m}-1} \sum_{j=0}^{\frac{W}{m}-1} b_j 2^{W-m+j} \end{aligned} \quad (1)$$

For this architecture, three types of modules are needed. Type I are the unsigned modules. Type II modules handle the m-bit partial product of an unsigned value with a 2's complement value. Finally, we have Type III modules that operate on two signed values. Only one Type III module is required for any type of multiplier, whereas ($2\frac{W}{m}$-2) Type II modules and $(\frac{W}{m}-1)^2$ Type I modules are needed.

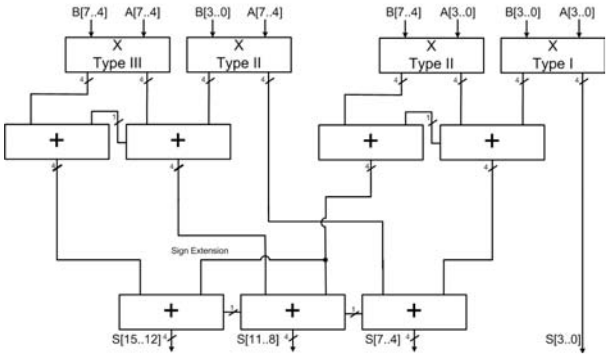

Figure 4. 8 bit wide binary array multiplier architecture (m=4).

In this work it is proposed new Type I, Type II and Type III dedicated blocks for radix-2^m multiplication. These dedicated blocks are applied to 16, 32 and 64 bit wide array multipliers. The dedicated blocks can be composed by regular or asymmetrical structures depending on the radix of the multiplication operation. The number of partial product lines for the regular architectures is given by: ($\frac{W}{m}$-1) [5]. Table I shows the impact on reducing the number of partial product lines in the multipliers presented in this work. As will be shown later, the reduction of partial product lines has a real impact on delay and power consumption improvements.

TABLE I. NUMBER OF PARTIAL LINES FOR DIFFERENT RADICES OF THE ARRAY MULTIPLIERS.

Number of bits (W)	Number of Partial Lines				
	radix-8 (m=3)	radix-16 (m=4)	radix-32 (m=5)	radix-64 (m=6)	radix-256 (m=8)
16	4	3	2	1	1
32	9	7	5	4	3
64	20	15	11	9	7

When the result of ($\frac{W}{m}$) is not an integer number the multiplier is called asymmetrical because its structure will be composed of some irregular 1:m multiplication blocks, as explained in Section IV (A). In the case of the multipliers mentioned in Table I for (m=3, m=5, m=6) the multipliers are asymmetrical and the number of partial product lines is 1 less than the integer part of the expression. As an example, for a 16-bit wide multiplier operating on radix-8, there will be four partial product lines present in the multiplier architecture, as can be seen in Table I. This occurs because in the radix-8 operation, three bits are multiplied at a time and in this case, three partial results from the partial lines can be added together in the CSA tree structure. The number of dedicated multiplication blocks that are needed in this operation is presented in the next section.

IV. DEDICATED BLOCKS FOR RADIX-2^M MULTIPLICATION

In the work of [5], the radix-16 dedicated blocks are all described in PLA format and automatically synthesized in the SIS environment. In the PLA description all the inputs and outputs of the radix-16 multiplications are taken into account. The PLA description is mapped onto a gate level library (*mcnc.genlib*). The radix-16 dedicated blocks are finally generated in Berkeley Logic Interchange Format (BLIF). It was observed that these dedicated blocks automatically generated by SIS are complex and thus, we propose a new scheme for the improvement of these blocks as well as the extension for asymmetrical structures that are proposed in this work.

A. New Scheme for Asymmetrical Multiplication Structure

In the structure of the radix-2^m array multiplier, when the number of dedicated blocks of multiplication is less than the number of bits of the operands, it is said that this structure is asymmetrical. Thus, the number of multiplication blocks in an asymmetrical structure is calculated by the following: in the structure of the multiplier, the composition of the array structure depends on the values of W and m and its calculation is given according to (2).

$$x = \frac{W}{m} \quad (2)$$

Equation (2) 2) is used to calculate the number of dedicated multiplication blocks into each multiplier line. As x is not an integer number for asymmetrical multipliers, a temporary value for W, named W_temp, and defined in (3), is calculated. This temporary value is needed to calculate the number of asymmetrical dedicated multiplication blocks. In (3), the term $int(x)$ represents the ceiling of x, the smallest integer greater than x.

$$W\_temp = m \times \text{int}(x) \quad (3)$$

This strategy is needed to calculate the size of the asymmetrical block (fb). This is made according to (4).

$$fb = m - (W\_temp - W) \quad (4)$$

The number of $m \times fb$ blocks is defined as fc and is calculated according to (5). The term $int(x)$ represents the floor of x, the largest integer smaller than x, which is multiplied by 2 because the two operands will be multiplied by the fb terms.

$$fc = 2 \times \text{int}(x) \quad (5)$$

As an example, if a multiplier with operands of W=16 bits operates on radix-8 (m=3), the values of W_temp= 18 and fb=1 are obtained. So, the architecture of the multiplier will be composed by ten 3x1 (fc) and one 1x1 (fb x fb) dedicated multiplication blocks. As should be observed, independently of the size of the operands, we will ever need only one fb x fb block, because this represents the multiplication of the last most significant bits.

An example of an asymmetrical structure of a multiplication in radix-8 operation is presented in Fig. 5. The radix-8 multiplication block can be optimized using one simple radix-4 (m=2) multiplication block, one CSA, one 2:1 adder block and two 1:2 asymmetrical multiplication blocks. As should be observed, the 1:m asymmetrical blocks are easily obtained by Karnaugh maps. However, in the example of Fig. 5, the 1:2 multiplication blocks are composed of simple AND gates. Moreover, one additional AND gate is used for the multiplication of the most significant bits. The critical path of the radix-8 optimized multiplication block is shown by the dotted lines of the Fig. 5.

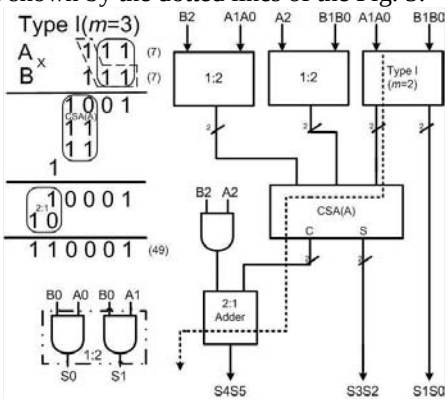

Figure 5 - Radix-8 dedicated multiplication block structure.

These new scheme presented in Fig. 5 can be naturally extended to radices 32 and 64 multiplication blocks using the same methodology used in the radix-8 optimized multiplication blocks.

B. New Scheme for Regular Multiplication Structure

In this scheme, as well as radix-8, more simple dedicated radix-4 (m=2) is used as basic block for the composition of more complex blocks. This radix-4 block is easily obtained using Boolean logic. For this block few logic gates are needed. Thus, the radix-16 block is arranged by using simple radix-4 blocks and CSA. The new scheme is applied to Type I, Type II and Type III blocks. A concrete example is presented in Fig. 6, for an 8-bit radix-16 multiplication.

As can be observed, groups of 4 bits are multiplied at a time and groups of 4 bits of the partial terms are added together in order to obtain the final result.

The structure of Type I, Type II and Type III blocks are quite the same, as can be observed in Fig. 6. However, Type II and Type III blocks use one more CSA block due to the sign extension strategy.

As can be observed in the dotted lines of the Fig. 6, the critical paths of the optimized Type I, Type II and Type III blocks are composed of only one multiplication block m=2, one CSA block, one half adder and one full adder.

The regular structure presented by the new radix-16 scheme enables it to be easily extended for radix-256, as can be observed in the examples of Fig. 7 for Type III radix-16 and radix-256 multiplications.

As can be seen in the examples, the m=8 structure is composed by the same number of CSA adders as for the m=4 example. However, the number of bits of the CSA is 2 times in the m=4 example. The number of bits for the last line of ripple carry addition for the m=8 multiplication is also higher than the m=4.

As should be observed in Fig. 7, the m=8 Type III block is composed by m=4 Type I, Type II and Type III optimized blocks. This occurs because the m=8 multiplication is broken into less complex m=4 multiplication as can observed in the example of Fig. 6. As for the radix-16 structure, one dedicated block, one CSA, one FA and one HA are present in the critical paths of the Type I, Type II and Type III radix-256 blocks.

The dotted lines of the Fig. 7 show the critical path of the Type III multiplication block.

The architecture for the m=8 example is shown in Fig. 8, for a Type III multiplication.

V. RESULTS

In this section we present results for W=16, 32 and 64-bit multiplier architectures on radix-2^m by using the original and the new alternative for the dedicated radix-2^m multiplication blocks presented before. Firstly, we present the results for the array multiplier by using the new scheme for the dedicated optimized radix-2^m blocks comparing with the radix-2^m multipliers using the original blocks based on PLA as proposed by [5] and also including the asymmetrical multipliers implemented in this work.

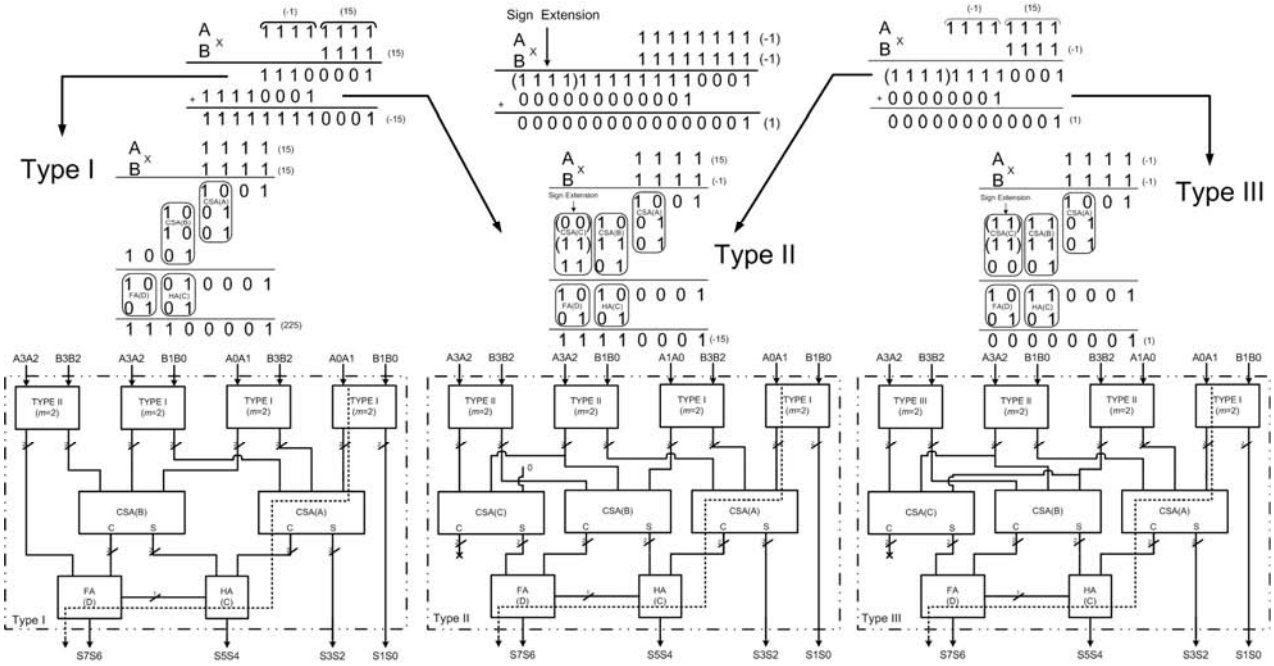

Figure 6. A concrete example showing the new scheme for an 8-bit radix-16 array multiplication.

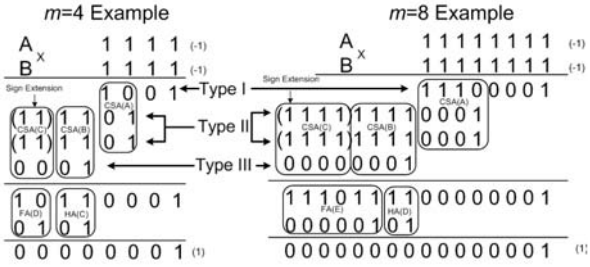

Figure 7. Type III radix-16 and radix-256 multiplication examples

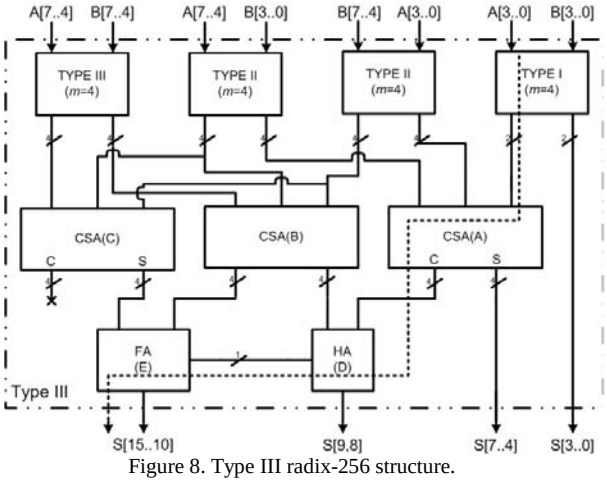

Figure 8. Type III radix-256 structure.

After that, we compare the optimized radix-2^m architecture against the Modified Booth multiplier. Area and delay were obtained using the SIS tool and power results were obtained with the SLS tool. Area results are presented in terms of number of literals. Delay results were obtained using the worst-case delay propagation between the input and output signals. Power results were obtained using the average power value of the SLS tool [8], under a general delay model. For the power simulation we have applied a random pattern signal with 10,000 input vectors

A. Radix-2^m Array Multiplier Comparisons

The use of the new proposed scheme contributes for a considerable area reduction in the 16, 32 and 64 bit radix-2^m array multiplier, as can be observed in Table II. This occurs because the new alternative uses more simple radix-4 dedicated multiplication block (m=2) in its structure. Since the optimized radix-2^m dedicated blocks are composed by radix-4 structures, the 16, 32 and 64 bit multipliers dedicated blocks present slightly less area than the multipliers that use radix-2^m original blocks, as can be observed in Table II.

The aspect of using simpler radix-4 dedicated blocks also contributes for the reduction of delay and power consumption in the multipliers.

This feature is more relevant for the power consumption, where the more regular and less complex structure presented by the optimized block contributes for a significantly power reduction in the 16, 32 and 64 bit multipliers, as presented in Table II, when compared against the original architectures. The best results obtained in the simulations for the 32 and 64-bit multipliers are those which uses radix-256 optimized block that enables a large reduction in the number of partial product lines in these multipliers. In terms of power consumption there is a relationship between the higher complexity of the dedicated multiplication blocks and the less number of partial product lines presented in the array structure.

As can be observed in Table II, for the operands of 32 and 64-bit wide, the radix-256 array multipliers present slightly less power consumption.

TABLE II. AREA, DELAY AND POWER RESULTS FOR RADIX-2^M ORIGINAL AND OPTIMIZED ARRAY MULTIPLIERS

		16 bits			32 bits			64 bits		
		Area	Delay	Power	Area	Delay	Power	Area	Delay	Power
m=3	Original	8382	218.5	0.09524	31849	448.5	0.70459	130622	919.3	5.5182
	Optimized	5117	227.3	0.08355	18951	457.3	0.66352	75581	928.1	5.3533
m=4	Original	14983	206.2	0.08718	61935	431.4	0.46172	251551	881.3	2.1491
	Optimized	5102	205.7	0.06013	20166	430.9	0.35525	79862	881.3	1.8405
m=5	Original	33452	234.5	0.09504	131061	458.9	0.70459	542804	893.7	5.5182
	Optimized	6821	218.7	0.04799	23265	443.1	0.46144	89204	887.9	3.8909
m=6	Original	61758	322.7	0.17101	325885	536.1	1.02614	1505800	973.1	5.5622
	Optimized	5260	210.7	0.06449	21469	424.1	0.38696	80236	861.1	2.2249
m=8	Optimized	5772	201.9	0.06899	22552	414.5	0.32264	88080	839.7	1.7751

TABLE III. AREA, DELAY AND POWER RESULTS FOR RADIX-2^M OPTIMIZED ARRAY AND MODIFIED BOOTH MULTIPLIERS

		Area (literals)		Delay (ns)		Power (W)	
16 bits	Booth	3708		233.7		0.09525	
	m=3 Optimized	5117	+27.53(%)	227.3	-2.73(%)	0.08355	-12.28(%)
	m=4 Optimized	5102	+27.32(%)	205.7	-11.98(%)	0.06013	-36.87(%)
	m=5 Optimized	6821	+45.63(%)	218.7	-6.42(%)	0.04799	-49.61(%)
	m=6 Optimized	5260	+29.50(%)	210.7	-9.84(%)	0.06449	-32.29(%)
	m=8 Optimized	5772	+35.76(%)	201.9	-13.61(%)	0.06899	-27.57(%)
32 bits	Booth	14596		474.7		0.7046	
	m=3 Optimized	18951	+22.98(%)	457.3	-3.60(%)	0.66352	-5.83(%)
	m=4 Optimized	20166	+27.62(%)	430.9	-9.23(%)	0.35525	-49.58(%)
	m=5 Optimized	23265	+37.26(%)	443.1	-6.59(%)	0.46144	-34.51(%)
	m=6 Optimized	21469	+32.01(%)	424.1	-10.60(%)	0.38696	-45.08(%)
	m=8 Optimized	22552	+35.28(%)	414.5	-12.68(%)	0.32264	-54.21(%)
64 bits	Booth	57876		959.2		5.5183	
	m=3 Optimized	75581	+23.45(%)	928.1	-3.42(%)	5.3533	-2.99(%)
	m=4 Optimized	79862	+27.53(%)	881.3	-8.12(%)	1.8405	-66.65(%)
	m=5 Optimized	89204	+35.12(%)	887.9	-7.43(%)	3.8909	-29.49(%)
	m=6 Optimized	80236	+27.86(%)	861.1	-10.22(%)	2.2249	-59.68(%)
	m=8 Optimized	88080	+34.29(%)	839.7	-12.46(%)	1.7751	-67.83(%)

This occurs because the impact of the higher complexity presented by the radix-256 dedicated multiplication blocks is minimized by the large reduction of the partial product lines in the 32 and 64-bit wide array multipliers.

Other aspect to be emphasized is the best results obtained by the radix-16 and radix-256 multipliers, in terms of performance and power consumption, when compared against radix-8, radix32 and radix-64 multipliers. This is explained by the regularity presented by these multipliers that enables the reduction of glitching activity and critical path in those architectures.

B. Radix-2^m Array Versus Modified Booth Multiplier Comparisons

In this subsection the radix-2^m Optimized Array multipliers are compared against the Modified Booth multiplier. In previous work [13] these comparison were limited to radices 16 and 256 for operands of W=16 bits. In this work we extend the comparisons to multipliers with W=32 and 64 bits operating on radices 8, 32 and 64 that make the multiplier asymmetrical, as explained in Section III (B). The main results are presented in Table III.

As can be observed in this table, radix-2^m optimized array multiplier presents more area than the Modified Booth. This occurs due to a more complexity presented by the basic multiplier elements that compose the modules for the radix-2^m product terms. However, the optimized array multipliers present reduction in terms of delay and power consumption for most of the cases. These results are quite the same presented before by [5], when radix-2^m array and Modified Booth multipliers were only compared to 16-bit architectures. Thus, we have observed that the strategy proposed in this work is more efficient than the strategy of the Modified Booth multiplier, even for the array multipliers that present asymmetrical structure. As can be observed in the results of Table III, all the multipliers operating on radices 8, 16, 32, 64 and 256 are more efficient in terms of delay and power consumption when compared against Modified Booth. In fact, while in the Booth multiplier, 2 bits of multiplication are performed at once and the multiplier requires half of the stage, in the radix-2^m array multiplier the number of stages can be reduced for more than half, while the regularity can be kept as in the pure array multiplier circuit, as presented before. Even in the case where the architectures are asymmetrical, the aspect of the regularity is almost the same.

In the same manner as in [5], it was observed that the regularity and the higher reduction of the number of partial product lines contributes for the reduction of the glitching activity in the array multiplier, which leads to a considerable less power consumption in this multiplier, when compared against Modified Booth, and this aspect can be observed in the results of Table III. To confirm this, we have performed power estimation of these multipliers for 32 and 64 bit width. In this case, it was

observed that the regularity characteristic presented by the array multiplier has a big impact on performance improvement and power reduction, mainly for the radix-256 architectures, where there are higher reductions of the partial product lines.

VI. CONCLUSIONS

In this work we presented a new scheme for the radix-2^m multiplication block of the array multiplier of [5]. We showed that the new scheme is naturally extended for different radices blocks. We have applied these dedicated blocks to 16, 32 and 64-bit multipliers and the results obtained showed that the multipliers with the new schemes are significantly more efficient than the original architectures. It was also observed that the multipliers that use optimized radix-256 block can be the most efficient among the multipliers analyzed in this work, mainly for a higher number of bits (32 and 64-bit wide). This is due to the higher reduction of the partial product lines presented by the multipliers with radix-256 dedicated multiplication blocks. We have performed a comparison between the optimized radix-2^m array multipliers against Modified Booth for 16, 32 and 64-bit wide operands and the results showed the considerable higher performance and less power consumption presented by the array multiplier architectures. As future work we intend to observe the impact on applying the proposed multipliers to architectures of Digital Signal Processing such as Finite Impulse Response (FIR) filters and Fast Fourier Transform (FFT).

REFERENCES

[1] C. Wallace. A Suggestion for a Fast Multiplier. IEEE Transactions on Electronic Computers, 13:14-17, 1964.

[2] W. Gallagher and E. Swartzlander. High Radix Booth Multipliers Using Reduced Area Adder Trees. In: Twenty-Eighth Asilomar Conf. on Signals, Systems and Computers, vol. 1, pages 545-549, 1994.

[3] A. Goldovsky and et al. Design and Implementation of a 16 by 16 Low-Power 2´s Complement Multiplier. In: IEEE ISCAS, pages 345-348, 2000.

[4] A. Bellaouar and M. Elmasry. Low-Power Digital VLSI Design Circuits and Systems. Kluwer Academic Publishers, 1995.

[5] E.Costa, J. Monteiro, and S. Bampi. A New Architecture for Signed Radix 2^m Pure Array Multipliers. In *IEEE International Conference on Computer Design*, pages 112-117, 2002.

[6] E. Sentovich. SIS: a System for Sequential Circuit Synthesis. *Berkeley: University of California*, 1992.

[7] A. Sohn. Computer Architecture – Introduction and Integer Addition. Computer Science Department – New Jersey Institute of Technology, 2004.

[8] A. Genderen. SLS: an efficient switch-level timing simulator using min-max voltage waveforms. International c*onference on Very Large Scale Integration*, VLSI, 1989, Munich, pages 79-88, 1978.

[9] K. Hwang. Computer Arithmetic – Principles, Architecture and Design. School of Electrical Engineering, 1979.

[10] B. Cherkauer and E. Friedman. A Hybrid Radix-4/Radix-8 Low Power, High Speed Multiplier Architecture for Wide Bit Widths. In IEEE ISCAS, vol. 4, pages 53-56, 1996.

[11] P. Seidel, L. McFearing, and D. Matula. Binary Multiplication Radix-32 and Radix-256. In 15th Symp. On Computer Arithmetic, pages 23-32, 2001.

[12] I. Khater, A. Bellaouar, and M. Elmasry. Circuit Techniques for CMOS Low-Power High-Performance Multipliers. IEEE Journal of Solid-State Circuits, 31:1535-1546, 1996.

[13] L. Pieper, E. Costa, S. M. Almeida, S. Bampi and J. C. Monteiro. Efficient Dedicated Multiplication Blocks for 2´s Complement Radix-16 and Radix-256 Array Multipliers. In: 2nd International Conference on Signals, Circuits and Systems, 2008, Hammamet. Proceedings of the 2nd SCS 2008.

Leandro Z. Pieper received the B.E.E. degree from the Catholic University of Pelotas (UCPel), Pelotas, Brazil, in 2004 and the M.Sc. degree in computer science from Catholic University of Pelotas (UCPel), Pelotas, Brazil, in 2008. Currently, he is a Professor of electrical engineering and computer science of the Catholic University of Pelotas, Pelotas, Brazil. His research interests are in microelectronics including low power arithmetical operands and digital signal processing.

Eduardo da Costa received the engineering and master´s degrees in electrical engineering from the University of Pernambuco and University of Paraiba, Brazil, in 1988 and 1991, respectively, and the Ph.D. degree in computer science from the University of Rio Grande do Sul, Porto Alegre, Brazil, in 2002. Part of his doctoral work was developed at the Instituto de Engenharia de Sistemas e Computadores (INESC-ID), Lisbon, Portugal. He is currently a Professor with the Departments of Electrical Engineering and Informatics, Catholic University of Pelotas (UCPel),Pelotas, Brazil. He is also with the Master Degree Program in Computer Science, UCPel, as a Professor and a Researcher. His research interests are VLSI architectures and low-power design.

Sérgio J. M. de Almeida received the B.E.E. degree from the University of Pernambuco (UPE), Pernambuco, Brazil, in 1988, the M.Sc. degree in electrical engineering from the Federal University of Paraíba (UFPB), Paraíba, Brazil, in 1991, and the Ph.D. degree in electrical engineering from the Federal University of Santa Catarina (UFSC), Florianópolis, Brazil, in 2004. Currently, he is a Professor of electrical engineering and computer science of the Catholic University of Pelotas, Pelotas, Brazil. His research interests are in digital signal processing, including stochastic analysis of adaptive filtering, echo cancellation, and dedicated hardware for signal processing.

Sergio Bampi is an Electronics Engineer and B.Sc. in Physics from the Federal Univ. of Rio Grande do Sul (UFRGS, 1979), M.Sc. and Ph.D. degrees in Electrical Engineering from Stanford University in 1986. He is a professor at the Informatics Institute at UFRGS University since 1981 and Distinguished Lecturer of IEEE CAS Society (2009-10). Sergio Bampi is a Ph.D advisor and project leader on the Microelectronics and Computer Science Programs, and his research interests are in the area of IC design and modeling, mixed signal and RF CMOS design, low power digital design, dedicated complex architectures and ASICs for image and video processing. Sergio Bampi has co-authored more than 130 papers in these fields and in MOS devices, circuits, technology and CAD. He was the President of the Research Funding agency FAPERGS in Brazil, the Technical Director of the Microelectronics Center CEITEC at Porto Alegre, and Coordinator of the Graduate Program on Microelectronics at Federal University UFRGS.

José Monteiro received a Licenciatura degree (a 5-year engineering degree) and an M.Sc. degree in Electrical and Computer Engineering, in 1989 and 1992 respectively, both from the Technical University of Lisbon (IST), Portugal, and a PhD degree in Electrical Engineering and Computer Science in 1996 from the Massachusetts Institute of Technology, Cambridge, MA, USA. He pioneered the area of data-dependent clock gating and has had important contributions in the power estimation of sequential circuits. He is currently an Associate Professor at IST and Director of INESC-ID, a research institute in Lisbon, Portugal. Prof. Monteiro has published 2 books and over 80 papers in international journals and conferences, having received a best paper award at IEEE Transactions on VLSI Systems. His current research interests include power analysis methods and synthesis algorithms for both hardware and software.