Efficient Dedicated Multiplication Blocks for 2´s Complement Radix-16 and Radix-256 Array Multipliers

Leandro Zafalon Pieper [1]
leandrozaf@pop.com.br

Eduardo A. C. da Costa [1]
Sérgio J. M. de Almeida [1]
{ecosta,smelo}@ucpel.tche.br

Sergio Bampi [2]
bampi@inf.ufrgs.br

José C. Monteiro[3]
jcm@inesc-id.pt

Universidade Católica de Pelotas – UCPel [1]
Universidade Federal do Rio Grande do Sul – UFRGS [2]
TU Lisbon / Instituto Superior Técnico – IST / INESC-ID[3]

Abstract

In this paper, we introduce new dedicated blocks for radix-16 and radix-256 multiplication. These blocks are basic components of the structure of the 2´s complement radix-2^m array multiplier proposed previously in the literature. In the original array multiplier, the blocks that perform the radix-16 multiplication were automatically synthesized from a truth table. The dedicated radix-16 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 radix-256 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.

1. 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, dependent 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-16 multiplication block using less complex radix-4 block and Carry Save (CSA) adder, 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 a radix-256 block.

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], even the radix-16 array multiplier can save power when compared against the radix-4 Modified Booth multiplier. This is mainly due to the lower logic depth presented by the array multiplier, which has 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 radix-16 and radix-256 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 3 makes an overview of relevant work related to our own, namely an overview of the Booth algorithm and a summary of the 2´s complement radix-2^m binary array multiplier. The alternatives for the radix-16 and radix-256 dedicated blocks are presented in Section 4. Performance comparisons are presented in Section 5. Finally, in Section 6 we conclude this paper, discussing the main contributions and future work.

2. 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 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.

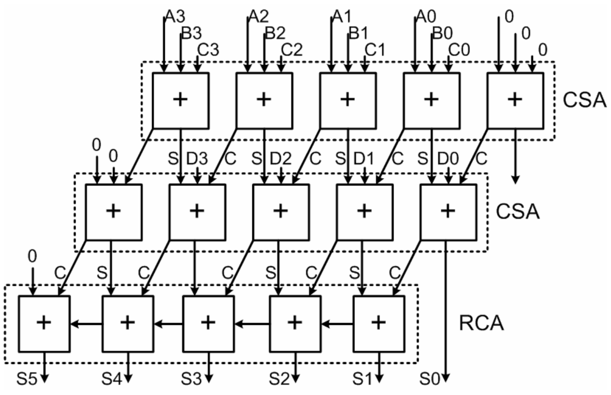

Fig. 1. Example of a carry save structure.

For the composition of the radix-16 dedicated block, we use the simpler radix-4 block, which are composed of only a few logic gates [5].

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

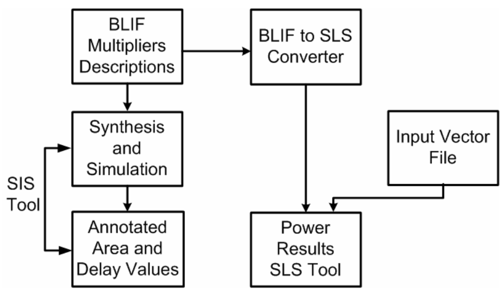

Fig. 2. Design flow for the multipliers implementation.

3. Related Work

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.

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-16 and radix-256 multiplication block in order to improve the efficiency of the array multiplier of [5].

3.1. 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, +X, +2X, -X and -2X) 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. 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.

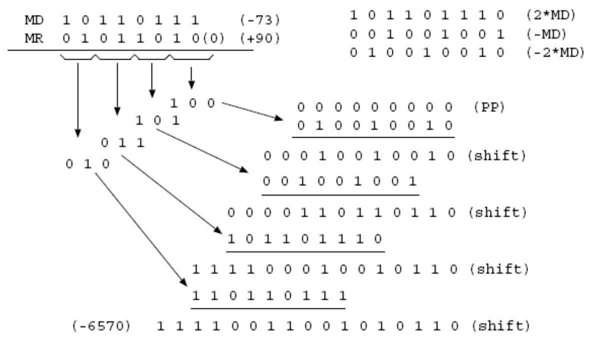
MD 1 0 1 1 0 1 1 1 (-73)
MR 0 1 0 1 1 0 1 0(0) (+90)
1 0 1 1 0 1 1 1 0 (2*MD)
0 0 1 0 0 1 0 0 1 (-MD)
0 1 0 0 1 0 0 1 0 (-2*MD)
1 0 0
1 0 1
0 1 1
0 1 0
0 0 0 0 0 0 0 0 0 (PP)
0 1 0 0 1 0 0 1 0
0 0 0 1 0 0 1 0 0 1 0 (shift)
0 0 1 0 0 1 0 0 1
0 0 0 0 1 1 0 1 1 0 1 1 0 (shift)
1 0 1 1 0 1 1 1 0
1 1 1 1 0 0 0 1 0 0 1 0 1 1 0 (shift)
1 1 0 1 1 0 1 1 1
(-6570) 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 1 0 (shift)

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

3.2. 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 Eq. (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.

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

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.

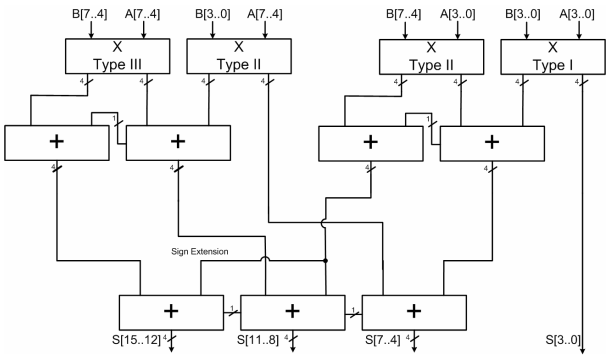

Fig. 4. 8 bit wide binary array multiplier architecture (m=4).

In this work it is proposed new Type I, Type II and Type III dedicated blocks for radix-16 and radix-256 multiplication. These dedicated blocks are applied to 16, 32 and 64 bit wide array multipliers. The number of partial product lines for the architectures is given by: $(\frac{W}{m}-1)$ [5]. Table I shows the impact of 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 1. Number of partial lines in the radix-16 and radix-256 array multipliers.

Number of bits (W)	Number of Partial Lines	
	radix-16 (m=4)	radix-256 (m=8)
16	3	1
32	7	3
64	15	7

4. Dedicated Blocks for Radix-16 and Radix-256 Multiplication

In the work of [5], the radix-16 dedicated blocks are all described in PLA format and automatically synthesized by 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.

In this scheme, dedicated radix-4 (m=2) blocks are used. This 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. 5, 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. 5. 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. 5, the critical paths of the optimized Type I, Type II and Type III blocks are composed by 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. 6 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 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.

The architecture for the m=8 example is shown in Fig. 7, for a Type III multiplication.

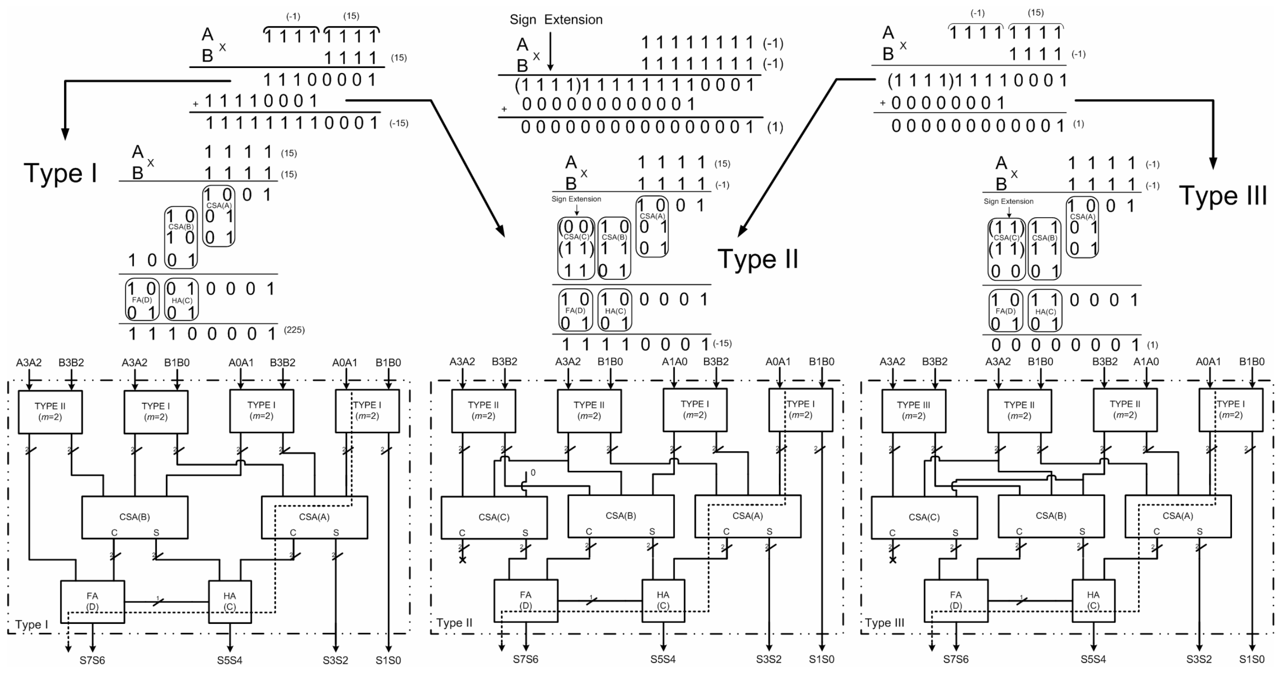

Fig. 5. A concrete example showing the new scheme for an 8-bit radix-16 array multiplication.

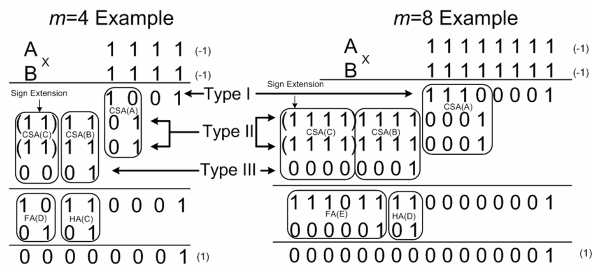

Fig 6. Type III radix-16 and radix-256 multiplication examples.

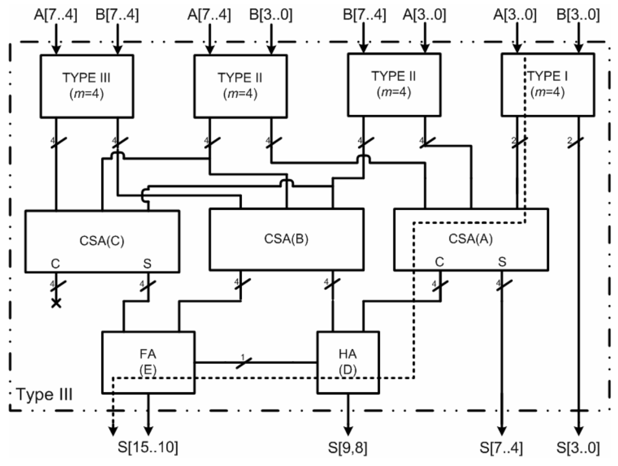

Fig.7. Type III radix-256 structure.

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. However, the dedicated block for the radix-256 is the optimized radix-16 block. Moreover, the FA, HA and CSA modules are larger in the radix-256 block. The dotted lines of the Fig. 7 show the critical path of the Type III multiplication block.

5. Results

In this section we present results for *W*=16, 32 and 64-bit multiplier architectures on radix-16 (*m*=4) and radix-256 (*m*=8) by using the original and the new alternative for the dedicated radix-16 multiplication blocks presented before.

Firstly, we present the results for the array multiplier by using the new scheme for the dedicated radix-16 and radix-256 blocks. After that, it will be compared the original radix-4 architecture of [5] and the new radix-16 and radix-256 architectures 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 10000 input vectors.

Table 2. Area, delay and power results for Radix-16 Original and Radix-16 Optimized Array multipliers comparisons

# bits	Area (literals)		Diff. Optim. vs. Original.	Delay (ns)		Diff. Optim. vs. Original.	Power (W)		Diff. Optim. vs. Original
	m=4 Orig.	*m*=4 Optim		*m*=4 Orig.	*m*=4 Optim.		*m*=4 Orig.	*m*=4 Optim.	
16	14983	5102	-65.95 (%)	206.2	205.7	-0.24 (%)	0.1416	0.0601	-57.55 (%)
32	61935	20166	-67.44 (%)	431.4	430.9	-0.12 (%)	0.7053	0.3552	-49.63 (%)
64	251551	79862	-68.25 (%)	881.8	881.3	-0.06 (%)	3.8364	1.8405	-52.02 (%)

Table 3. Area, delay and power results for Radix-16 and Radix-256 Optimized Array multipliers comparisons

# bits	Area (literals)		Diff. *m*=8 vs. *m*=4	Delay (ns)		Diff. *m*=8 vs. *m*=4	Power (W)		Diff. *m*=8 vs. *m*=4
	m=4	*m*=8		*m*=4	*m*=8		*m*=4	*m*=8	
16	5102	5772	+11.61 (%)	205.7	201.9	-1.88 (%)	0.0601	0.0689	12.77 (%)
32	20166	22552	+10.58 (%)	430.9	414.5	-3.95 (%)	0.3552	0.3226	-9.18 (%)
64	79862	88080	+9.33 (%)	881.3	839.7	-4.95 (%)	1.8405	1.7751	-3.55 (%)

Table 4. Area, delay and power results for Array (*m*=2) Original and Modified Booth comparisons

# bits	Area (literals)		Difference Array vs. Booth	Delay (ns)		Difference Array vs. Booth	Power (W)		Difference Array vs. Booth
	Booth	Array *m*=2		Booth	Array *m*=2		Booth	Array *m*=2	
16	3708	4602	+19.43 (%)	233.7	227.8	-2.52 (%)	0.0953	0.0666	-30.04 (%)
32	14596	18586	+21.47 (%)	474.7	478.2	+0.73 (%)	0.7046	0.4337	-38.45 (%)
64	57876	74586	+22.40 (%)	959.2	979.0	+2.02 (%)	5.5183	3.1296	-43.29 (%)

Table 5. Area, delay and power results for Array (*m*=4) Optimized and Modified Booth comparisons

# bits	Area (literals)		Difference Array vs. Booth	Delay (ns)		Difference Array vs. Booth	Power (W)		Difference Array vs. Booth
	Booth	Array *m*=4		Booth	Array *m*=4		Booth	Array *m*=4	
16	3708	5102	+27.32 (%)	233.7	205.7	-11.98 (%)	0.0953	0.06013	-36.87 (%)
32	14596	20166	+27.62 (%)	474.7	430.9	-9.23 (%)	0.7046	0.35525	-49.58 (%)
64	57876	79862	+27.53 (%)	959.2	881.3	-8.12 (%)	5.5183	1.8405	-66.65 (%)

Table 6. Area, delay and power results for Array (*m*=8) Optimized and Modified Booth comparisons

# bits	Area (literals)		Difference Array vs. Booth	Delay (ns)		Difference Array vs. Booth	Power (W)		Difference Array vs. Booth
	Booth	Array *m*=8		Booth	Array *m*=8		Booth	Array *m*=8	
16	3708	5772	+35.76(%)	233.7	201.9	-13.61(%)	0.0953	0.06899	-27.57(%)
32	14596	22552	+35.28(%)	474.7	414.5	-12.68(%)	0.7046	0.32264	-54.21(%)
64	57876	88080	+34.29(%)	959.2	839.7	-12.46(%)	5.5183	1.7751	-67.83(%)

5.1 Radix-16 and Radix-256 Array multipliers comparisons

The use of the new proposed scheme contributes for a considerable area reduction in the 16, 32 and 64 bit radix-16 array multiplier, as can be observed in Table 2. This occurs because the new alternative uses more simple radix-4 dedicated multiplication block (*m*=2) in its structure. Since the radix-256 dedicated block is composed by two radix-16 structures, the 16, 32 and 64 bit multipliers with *m*=8 dedicated blocks present slightly more area than the multipliers that use radix-16 blocks, as can be observed in Table 3.

The aspect of using simpler radix-16 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 3, when compared against the original architectures.

The fact that the 16, 32 and 64-bit multipliers can use radix-256 optimized block enables a large reduction in the number of partial product lines in these multipliers. However, the radix-256 dedicated block is more complex than the simple radix-16 block. Thus, the 16, 32 and 64 bit multipliers present a slightly higher performance, when compared against the multipliers with radix-16 blocks, as can be seen in Table 3. 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 3, the 16-bit radix-256 array multiplier presents more power consumption, when compared against the radix-16 array multiplier. However, for the operands of 32 and 64-bit wide, the radix-256 array multipliers present slightly less power consumption. 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.

5.2 Radix-4, Radix-16 and Radix-256 Array versus Modified Booth multipliers comparisons

In the work of [5] the radix-4 array multiplier is compared against Modified Booth multiplier for only 16-bit wide architectures. In this section this comparison is extended for 32 and 64-bit, and the main results are presented in Table 4.

As can be observed in this table, radix-4 array multiplier presents more area than the Modified Booth. This due to a more complexity presented by the basic multiplier elements that compose the modules for the product terms. However, this multiplier presents quite the same delay and significant less power results. These results are quite the same presented before by [5] for 16-bit architectures. 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 in Fig. 4. 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, as can be observed in the results of Tables 4, 5 and 6. 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-16 and radix-256 architectures, where there are higher reductions of the partial product lines.

6. Conclusions

In this work we presented a new scheme for the radix-16 dedicated multiplication block of the array multiplier of [5]. We showed that the new scheme is naturally extended for radix-256 block. 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 also was observed that the multipliers that use optimized radix-256 block can be more efficient than that use the optimized radix-16 block, for a higher number of bits (32 and 64-bit wide). This due to the higher reduction of the partial product lines presented by the multipliers with radix-256 dedicated structure. We have performed a comparison between the optimized radix-4, radix-16 and radix-256 array multipliers against Modified Booth for 16, 32 and 64-bit wide operands and the results showed the considerable higher performance and less power presented by the array multiplier architectures, mainly for radix-16 and radix-256 operation. As future work we intend to extend our new scheme for other dedicated blocks with different radices and verify the impact on area, delay and power consumption by applying these strategies in the binary array multipliers.

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, pages545-549, 1994.

[3] A. Goldovsky and et al. Design and Implementation of a 16 by 16 Low-Power 2´s Complekment 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 *Conference 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.