Efficient Dedicated Structures for the Radix-16 Multiplication

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]
Instituto de Engenharia e Sistemas de Computadores – INESC-ID[3]

Abstract

In this work, we present three new dedicated blocks for the radix-16 multiplication. The blocks compose the structure of the 16 bit binary array multiplier proposed by [1]. In the array multiplier of [1] the blocks that perform the radix-16 multiplication are automatically synthesized by using SIS environment. In our work we have tested three other new structures by using less complex multiplication blocks and more efficient adders such as Carry Save (CSA) in order to speed-up the carry propagation inside the multiplication modules. We have compared area, delay and power consumption of the 16 bit array multiplier using the new dedicated modules, by using SIS and SLS (Switch Level Simulator) tools. The results we present show that with the new radix-16 structures it is possible to save area, delay and power consumption when compared against the previous original structure.

1. Introduction

Multiplier modules are common to many Digital Signal Processing (DSP) applications. Thus, a substantial amount of research work has been put into developing efficient architectures for multipliers given their widespread use and complexity.

In this work we improve the radix-2^m binary array multiplier architecture that was proposed in [1], by using different schemes for the dedicated modules that perform the radix-16 multiplication. The multiplier of [1] uses radix-2^m encoding for the operands. This aspect enables the array multiplier to reduce the number of partial lines by keeping the same level of regularity presented by a conventional array multiplier [2]. In the binary array multiplier, the radix-2^m multiplication is performed by dedicated multiplication blocks. For the radix-4 operation (m=2) these blocks are easily obtained. However, for the operation performed by blocks with radices higher than 4, the construction of the dedicated multiplication blocks become more complex. In the multiplier of [1], the radix-16 blocks are synthesized automatically by using SIS environment [3]. As these blocks appear in the binary array multiplier circuit (W/m)-1 times, where W is the number of bits, it is important to reduce the complexity of these blocks. In this work we have proposed three new structures for the radix-16 dedicated blocks, using less complex radix-4 blocks, Ripple Carry (RCA) and Carry Save (CSA) adders and dedicated 4:2 multiplication blocks. The dedicated blocks were all tested in the 16-bit binary array multiplier of [1] and the results we have found show that we can reduce area, delay and power consumption of this multiplier by using the new proposed blocks.

This paper is organized as follows. The next section makes an overview of relevant work related to our own. In Section 3 we summarize the radix-2^m binary array multiplier. The alternatives for the radix-16 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. Related work

Early multiplier schemes such as bi-section, Baugh-Wooley and Hwang [2] 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 [4], [5]. In the Modified Booth algorithm approximately half of the partial products that need to be added is used.

In the work presented in [1], the improvement in delay and power has the same principal source as for the Booth architecture, the reduction of the partial product terms, while keeping the regularity of an array multiplier. As observed in [1], the array multiplier is more efficient than the Modified Booth due to the lower logic depth that reduces the amount of glitching along the circuit.

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 [1] 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 our work we have proposed three new schemes for the dedicated radix-16 multiplication block in order to improve the efficiency of the array multiplier of [1].

3. Overview of 2's Complement Array Multiplier

In this section we summarize the methodology of [1] for the generation of regular structures for arithmetic operators using signed radix-2^m representation.

For the operation of a radix-2^m multiplication, 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 Equation 1.

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

This operation is illustrated in Figure 1, for a radix-16 multiplication example.

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.

For this architecture, three types of modules are needed as shown in Figure 2. 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, 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. We present a concrete diagram for W=16 bit wide operands using radix-16 (m=4) in Figure 3.

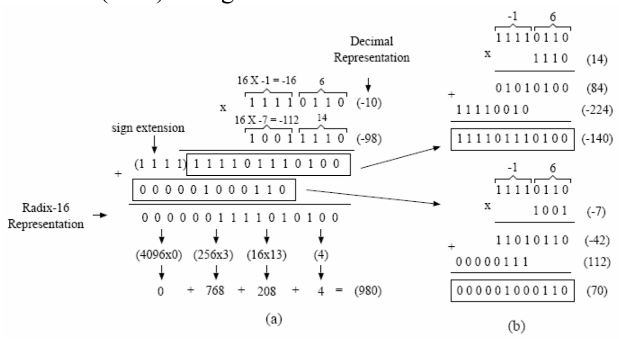

Fig. 1. Example of a 2's complement 8-bit wide radix-16 multiplication

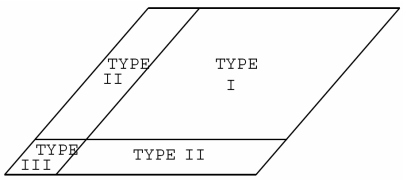

Fig. 2. General structure for a 2's complement radix-2^m multiplier

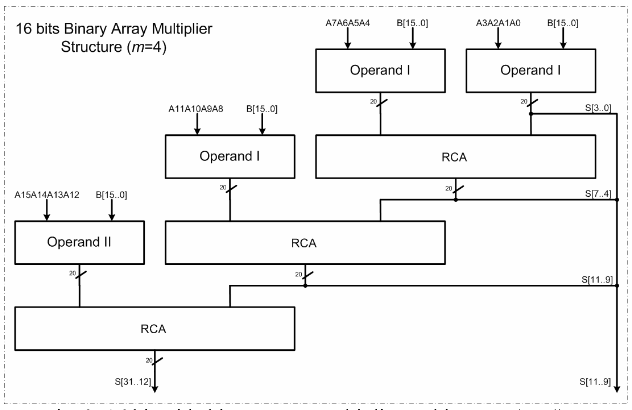

Fig. 3. 16 bit wide binary array multiplier architecture (m=4)

Figure 4 shows the internal structure of the Operand I (illustrated in Figure 3). As can be observed in Figure 4, the structure of this operand is composed by m=4 Type I multiplier blocks and adders to produce the partial line result. In this work we have proposed three new alternatives to optimize the m=4 Type I multiplier block.

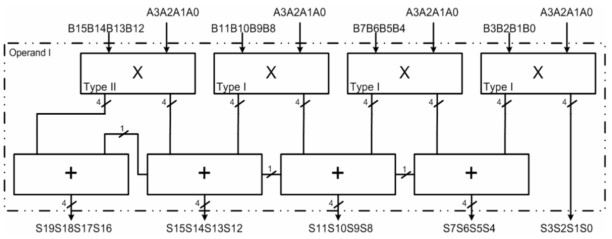

Fig. 4 Operand I structure

4. Dedicated blocks for the radix-16 multiplication

In the work of [1] the dedicated blocks for the radix-16 multiplication 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 multiplication are taken into account. The PLA description is mapped onto a gate level. The radix-16 dedicated blocks are finally generated in Berkeley Logic Interchange Format (BLIF). We have observed that these dedicated blocks automatically generated by SIS are complex and thus, we propose in this work three new alternatives for the improvement of this block. This is an important approach because in [1] it was shown that the radix-16 binary array multiplier, that uses this dedicated block, is more efficient than the radix-4 architecture. This due to the less amount of partial lines used in the radix-16 operation.

4.1 The use of radix-4 blocks and RCA Adders

In this first alternative we have used dedicated radix-4 (*m*=2) blocks. This block is easily obtained by Boolean logic. For this block, only 8 logic gates are needed. Thus, the radix-16 block is arranged by using simple radix-4 blocks and RCA, as can be observed in Figure 5(b). Figure 5(a) shows an example of the multiplication of two 4 bit operands. As can be observed, two bits are multiplied at a time and groups of two bits of the partial terms are added in order to obtain the final result.

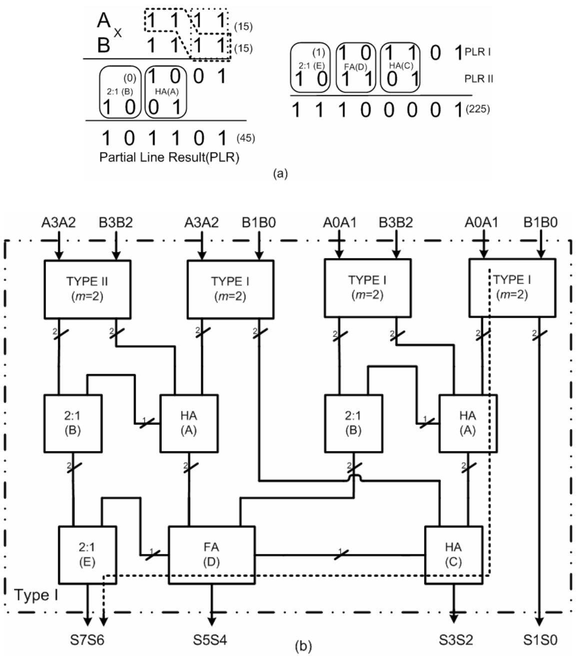

Fig. 5. Radix-16 block composed by *m*=2 multiplier blocks and RCA

As can be observed in Figure 5(b), only one full adder, three half adders and three adders 2:1 (addition of 2 bits and the carry-in) are needed to compose the radix-16 block. In this block, the critical path is composed by one multiplication block (*m*=2), two half adders and one 2:1 adder, as can be observed in the dotted lines of Figure 5(b).

4.2 The use of radix-4 blocks and CSA Adders

In this alternative we have used the same radix-4 block. But now we have used more efficient CSA [6] for the addition of the partial lines, as can be observed in the example of Figure 6(a). This example shows that by using CSA three numbers are partially added together. The basic idea 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. In a previous work [7], the CSA was applied to the partial product lines of the radix-4 multiplier and reductions on area, delay and power consumption were reported.

Figure 6(b) shows the use of CSA in the internal structure of the radix-16 block. As can be observed in the dotted lines of this figure, the critical path is composed by the sum of the delays of one multiplication block *m*=2, one CSA block, 1 half adder and one full adder.

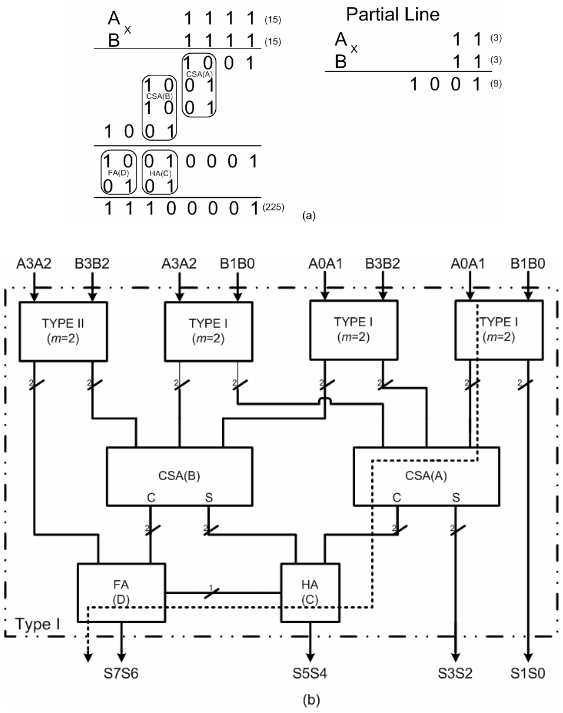

Fig. 6. Radix-16 block composed by *m*=2 multiplier blocks and CSA

4.3 The use of 4:2 multiplications blocks

In this alternative we have tried a similar methodology applied to the original radix-16 block. However, we have used a PLA description of a less complex 4:2 multiplication block rather than a more complex 4:4 multiplication block (as in the original methodology). The 4:2 block is automatically synthesized by SIS environment. An example of this operation is presented in Figure 7(a).

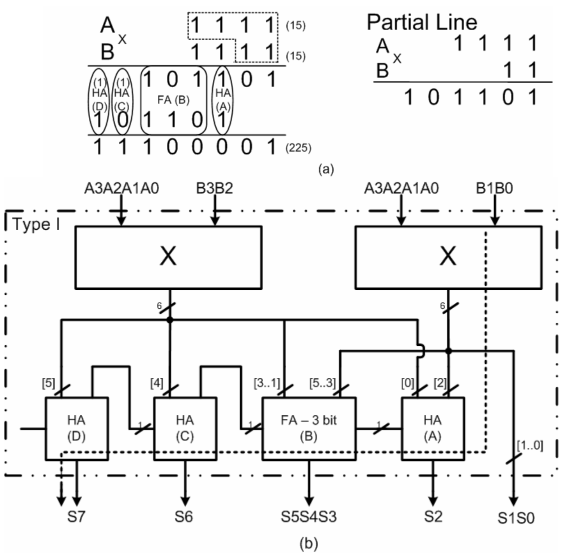

Fig. 7. Radix-16 block composed by 4:2 multiplication blocks

As can be observed Figure 7(b), by using dedicated 4:2 multiplier blocks, only one line of RCA is needed. The critical path is composed by one 4:2 multiplication block and the line of RCA as shown in the dotted lines of Figure 7(b).

5. Performance Results

In this section, we present results for W=16-bit multiplier architectures on radix 16 (m=4) by using the original and the new alternatives for the dedicated radix-16 multiplication blocks presented before. Area and delay were obtained by using SIS tool and power results were obtained with SLS tool. Area results are presented in terms of number of literals. Delay results were obtained using the worst delay propagation between the input and output signals. Power results were obtained by using the average power value of the SLS tool [8]. For the power simulation we have applied a random pattern signal with 10000 input vectors. The multipliers used in this work were all described in BLIF language.

5.1 Area results

As can be observed in Table 1, the three new alternatives present less area than the original one. This occurs because the new alternatives use more simple dedicated multiplication block. As can be also observed in Table 1, the first new alternative (radix-4 and RCA) presents less area value between all the alternatives. In fact, since this alternative uses more simple radix-4 multiplication block and more simple half adders and 2:1 adders, this alternative become more simple than the others.

Table 1. Area results

Alternatives	Area	%
original	15932	---
radix-4 and RCA	7544	-111.2
radix-4 and CSA	8612	-85.0
dedicated 4:2 block	7664	-107.8

5.2 Delay results

Although the second new alternative (radix-4 and CSA) presents the higher area, this alternative presents the less delay value, as can be seen in Table 2.

Table 2. Delay results

Alternatives	Delay (ns)	%
original	234.0	---
m=2 and RCA	204.8	-14.2
m=2 and CSA	199.3	-17.4
dedicated 4:2 block	200.3	-16.8

This occurs because this alternative uses more simple radix-4 multiplication block and more efficient tree of CSA. The use of CSA in the radix-16 block contributes for the reduction of carry propagation along the structure and thus, the critical path is reduced. As can be also observed in Table 2, all the new alternatives are more efficient than the original one. In fact, we have observed that the complexity of the original block contributes for the higher critical path presented by its internal structure.

5.3 Power consumption results

As can be observed in Table 3, the binary array multipliers using the three new alternatives are more efficient in terms of power consumption when compared against the original one. This occurs because the new alternatives present more simple and more regular structures, which contributes for the less amount of glitching inside the multiplication blocks.

Table 3. Power consumption results

Alternatives	Power (mW) random input	%
original	214.09	---
m=2 and RCA	189.94	-12.7
m=2 and CSA	190.42	-12.4
4:2 block	200.37	-6.8

6. Conclusions

We have presented 3 new alternatives for the dedicated radix-16 multiplication block which composes the radix-2^m binary array multiplier of [1]. As was observed, the binary array multiplier using the new alternatives become more efficient, because the new radix-16 blocks use more simple blocks of multiplication (radix-4). Besides, the inclusion of CSA into the dedicated radix-16 multiplication block, improves the performance and reduces the power consumption of the binary array multiplier. As future work we intend to test our new alternatives into higher radices dedicated multiplication blocks.

7. References

[1] 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

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

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

[4] 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.

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

[6] T. Kim, W. Jao, and S. Tjiang. Arithmetic Optimization using Carry-Save-Adders. *In 35th Design Automation Conference*, pages 433-438, 1998.

[7] M. R. Fonseca, E. Costa, S. Bampi, J. Monteiro. Performance Optimization of Radix-2^m Multipliers using Carry Save Adders. In XI Iberchip Workshop-IWS, 2005.

[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.