A New Architecture for 2's Complement Gray Encoded Array Multiplier

Eduardo Costa
UCPel, Pelotas, Brazil
ecosta@atlas.ucpel.tche.br

Sergio Bampi
UFRGS, P. Alegre, Brazil
bampi@inf.ufrgs.br

José Monteiro
IST/INESC, Lisboa, Portugal
jcm@inesc.pt

Abstract

We present a new architecture for signed multiplication. The proposed architecture maintains the pure form of an array multiplier, exhibiting a much lower overhead than the Booth architecture. We propose a Hybrid encoding for the architectures, which is a compromise between the minimal input dependency presented by the Binary encoding and the low switching characteristic of the Gray encoding. We have experimented using the Gray code for each group of m bits, thus potentially further reducing the switching activity both internally and at the inputs. The architecture is extended for radix-2^m encoding, which leads to a reduction of the number of partial lines, enabling a significant improvement in performance and power consumption. The flexibility of our architecture allows for the easy construction of multipliers for different values of m. The results we present show that the proposed architecture with radix-4 compares favorably in performance and power with the Modified Booth multiplier.

1 Introduction

Multiplier modules are common to many DSP applications. The fastest types of multipliers are parallel multipliers. Among these, the Wallace multiplier [1] are among the fastest. However, they suffer from a bad regularity. Hence, when regularity, high-performance and low power are primary concerns, Booth multipliers tend to be the primary choice [2], [3], [4], [5], [6].

Booth multipliers allow the operation on signed operands in 2's-complement. They derive from array multipliers where, for each bit in a partial product line, an encoding scheme is used to determine if this bit is positive, negative or zero. The Modified Booth algorithm achieves a major performance improvement through radix-4 encoding. In this algorithm each partial product line operates on two bits at a time, thereby reducing the total number of the partial products.

In this paper, we propose a new approach to handle operands in 2's-complement. We use exactly the same structure as an array multiplier, with the same unsigned bit products for all the bits except those that involve sign bits. We use an Hybrid operand encoding [7] to obtain power efficient arithmetic modules that operate directly on these codes. The aim is not only to reduce the switching activity in the input and output buses, but also the minimization of the complexity of the combinational logic in the modules. The proposed Hybrid architecture is more efficient than the Modified Booth architecture because no additional encoding is necessary. Additionally, there is no power overhead due to code converters since these are not required.

The regularity of the proposed architecture makes it naturally applicable for generic radix-2^m operations. By carefully designing these modules, significant improvement in delay and power are achieved as the number of product lines reduces.

We compare the Modified Booth architecture, which uses radix-4, to our Hybrid architecture with $m = 2$. The results show that the proposed architecture is significantly more efficient, with no delay penalties and 45% less power consumption. This power reduction is mainly due to the lower logic depth which has a big impact in the amount of glitching in the circuit.

This paper is organized as follows. The next section makes an overview of relevant work related to our own. In Section 3 we present operators built using the Hybrid encoding. The proposed modified array architecture to handle signed operands is presented in Section 4. This section also describes how this architecture can be directly extended for radix-2^m operation. Performance comparisons between the proposed array multiplier with m=2 and the Modified Booth, are presented in Section 5. Finally, in Section 6 we conclude this paper, discussing the main contributions and future work.

2 Related Work

A substantial amount of research work has been put into developing efficient architectures for multipliers given their widespread use and complexity. Schemes such as bisection, Baugh-Wooley and Hwang [8] propose the implementation of a 2's complement architecture, using repetitive modules with uniform interconnection patterns. However, it is not permitted an efficient VLSI realization due to the irregular tree-array form used. The same non-regularity aspect is observed in [9], where a scheme of a multiplexer-based multiplier is presented. In [6] an im-

provement of this technique is observed where the architecture has a more rectangular layout than [9].

The techniques described above have been applied to conventional array multipliers whose operation is performed bit by bit and sometimes the regularity of the multipliers is not preserved. In these work all the multiplying operations are performed in binary encoding representation.

Low-power techniques using data encoding methods are overviwed in [10], where it is shown that such techniques can decrease the power consumed for transmitting information over heavy load communicatiion paths (buses) by reducing the switching activity. Techniques such as Gray, Transition, Limited Weight and Bus-Invert are used to signal encoding in order to reduce switching activity on buses.

In [11], the process of implementing arithmetic operators that operate directly upon Gray code and Transition code inputs was investigated. However, it was observed that the combinational logic required by the arithmetic modules is large. Furthermore, it was not possible to develop a regular structure for the operators such that different word sizes could be accommodated.

More regular and suitable multiplier designs based on the Booth recoding technique have been proposed [2], [3], [5]. The main purpose of these designs is to increase the performance of the circuit by the reduction of the number of partial products. In the Modified Booth algorithm approximately half of the partial products that need to be added is used. To speed-up the summation of the partial products in Modified Booth multiplier, some other techniques such as Carry-save adder arrays, Wallace trees [12] and Dadda parallel counters [13] have been applied. However, it is observed that when the multiplier operands are wide enough these techniques may not be adequate for effective reduction of the multiplier latency [14].

In our work we propose an Hybrid operand encoding that is a compromise between the minimal iput logic dependency presented by the Binary encoding and the low switching characteristic of the Gray encoding. 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 [8]. We show that our architecture can be more naturally extended for higher radices, using less logic levels and hence presenting much less spurious transitions.

3 Unsigned Hybrid Arithmetic Operators

By using low-switching operand codes directly in the arithmetic modules, the process of encoding and decoding of the signals can be avoided.

We propose the use of a Hybrid encoding for the operators, which is a compromise between the minimal input dependency presented by the Binary encoding and the low switching characteristic of the Gray encoding [7]. In this section, we propose a methodology for the generation of regular structures for arithmetic operators, such as adders and multipliers, using Hybrid encoded operands.

3.1 Hybrid Code Definition

The idea of the Hybrid code is to split the operands in groups of m-bits, encode each group using the Gray code and use the Binary approach to propagate the carry between the groups. In this manner, the number of transitions inside each group is minimized and a regular structure can be easily built. Table 1 exemplifies the Hybrid encoding for 4-bit numbers and for m=2. In the remaining of this paper we will be using m=2.

Table 1. HYBRID CODE REPRESENTATION FOR m=2.

Dec	Hyb	Dec	Hyb	Dec	Hyb	Dec	Hyb
0	0000	4	0100	8	1100	12	1000
1	0001	5	0101	9	1101	13	1001
2	0011	6	0111	10	1111	14	1011
3	0010	7	0110	11	1110	15	1010

3.2 Radix-2^m Multiplication

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.

Consider two operands W-bits wide, $A = \sum_{i=0}^{\frac{W}{m}-1} a_i 2^{i \cdot m}$ and $B = \sum_{j=0}^{\frac{W}{m}-1} b_j 2^{j \cdot m}$, where each a_i, b_j are m-bit digits in radix-2^m representation. We have that

$$A \times B = \sum_{j=0}^{\frac{W}{m}-1} A \cdot b_j 2^{j \cdot m}, \qquad A \cdot b_j = \sum_{i=0}^{\frac{W}{m}-1} b_j \cdot a_i 2^{i \cdot m} \quad (1)$$

We illustrate this operation for operators with $W = 8$ bits using radix-4 ($m = 2$) in Figure 1. For the example shown, the partial product terms are obtained by multiplying each m-bit groups of the multiplier and multiplicand terms. Thus, each partial product line is computed by a $m \times W$ multiplication, as depicted in Figure 1(b).

The final product for the radix-2^m multiplication is obtained by adding each m-bit groups of the partial product terms, as shown in Figure 1(a). Also exemplified in Fig-

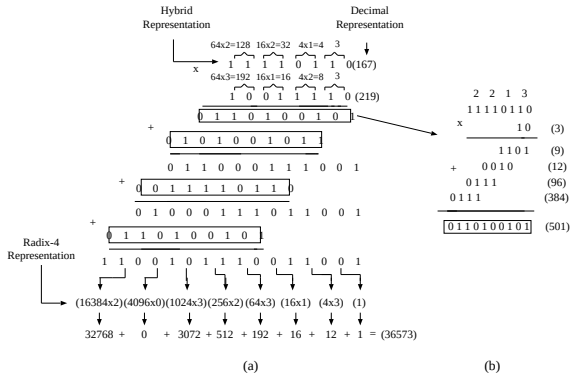

Figure 1. Example of a 8-bit wide radix-4 multiplication.

ure 1 is the conversion of radix-4 numbers to decimal values.

3.3 Arithmetic Operators Architectures

We have found that adders operating directly with the Hybrid encoded operands are more complex than Binary adders. Moreover, given the ease of conversion between Hybrid and Binary codes, the most efficient implementation for a Hybrid adder is in fact to use a Binary adder and make the conversion at the inputs and outputs. This is shown in Figure 2 for a 8-bit adder operator. Naturally, this implies that the Hybrid adders will consume more power than the Binary adders. However, depending on the data correlation and capacitance load of the data buses, an overall power reduction is still possible. Translating words of W bits at the inputs and outputs of an Hybrid adder with m=2, all that is required are $\frac{3}{2}W$ EXOR gates.

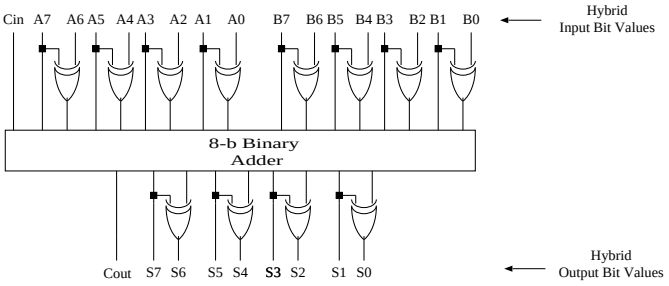

Figure 2. 8-bit Hybrid Adder Architecture.

In a similar manner as for the Binary array multiplier, the Hybrid array multiplying operation can be performed in a regular form. However, contrary to the Binary multiplier where the operation is performed bit by bit, the Hybrid operation is performed for a m-bit group at a time. The full Hybrid architecture can be seen in Figure 3 for a 8-bit Hybrid array multiplier with m=2.

As shown in Figure 3, the Hybrid array multiplier architecture presents a regular structure. The 2-least significant bits of the product are produced at the right hand side of

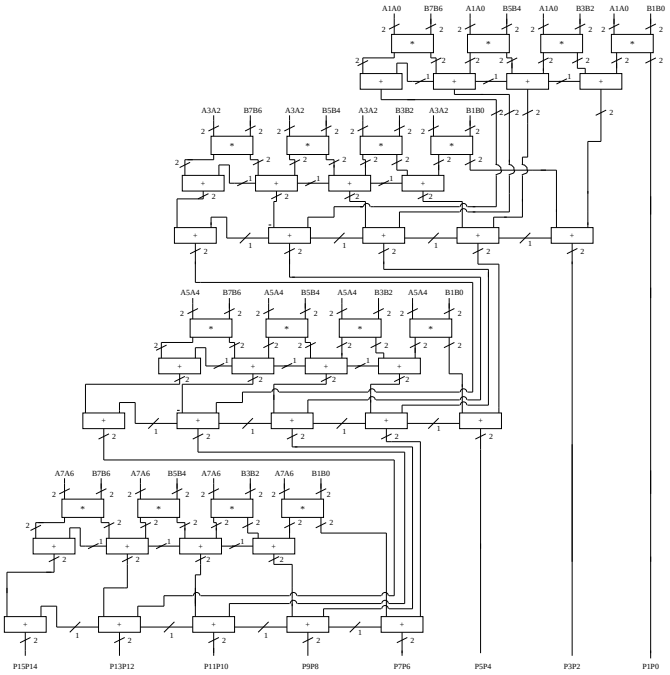

Figure 3. 8-bit Hybrid Array Multiplier Architecture.

the array. The other bits are produced by adding partial product terms. As can be observed for the 8-bit Hybrid architecture shown in Figure 3, it is necessary just one line composed by adders responsible for adding partial product terms. For a W-bit Hybrid architecture, (W/2)-1 of these adder lines will be used.

4 2's Complement Architectures

Besides the high level of regularity presented by the architecture developed in the previous section, its flexibility allows us to easily extend it to operands using any radix we choose. Besides, as will be shown the same architecture presented in the previous section can be easily built to operates in a 2's complement representation.

4.1 2's Complement Hybrid Code

The idea of splitting the operands in groups of m-bits and encode each gropu using the Gray code can be also used for operands that operate in 2's complement representation. Table 2 shows the 2's complement Hybrid encoding for 4-bit numbers and m=2.

Table 2. 2'S COMPLEMENT HYBRID CODE REPRESENTATION FOR m=2.

Dec	Hyb	Dec	Hyb	Dec	Hyb	Dec	Hyb
0	0000	4	0100	-8	1100	-4	1000
1	0001	5	0101	-7	1101	-3	1001
2	0011	6	0111	-6	1111	-2	1011
3	0010	7	0110	-5	1110	-1	1010

As can be observed in Table 2, even for a signed representation the Hybrid encoding features the desirable property of having a single bit different for consecutive values. For highly correlated streams, this encoding can in theory significantly reduce the number of transitions, and thus power.

4.2 2's Complement Radix-2^m Hybrid Multiplier Architecture

We demonstrate that all the observations made in Section 3 apply to signed operation. Consider now $A' = \sum_{i=0}^{\frac{W}{m}-2} a_i 2^{i \cdot m}$, where a_i is a m-bit digit. For positive numbers, the value represented by A is A' as before. For negative numbers, this value is $A - 2^W = a_{\frac{W}{m}-1} 2^{W-m} + A' - 2^W = A' - a_{\frac{W}{m}-1} 2^{W-m}$, since $a_{\frac{W}{m}-1} 2^{W-m} - 2^W$ is the 2's complement of $a_{\frac{W}{m}-1} 2^{W-m}$. Then we have:

$$A = \begin{cases} A' & , \quad a_{W-1} = 0 \\ A' - a_{\frac{W}{m}-1} 2^{W-m} & , \quad a_{W-1} = 1 \end{cases} \tag{2}$$

or simply

$$A = A' - a_{W-1} a_{\frac{W}{m}-1} 2^{W-m} \tag{3}$$

Using analogous observations as made for the binary case, from Equation 3 we can write:

$$\begin{aligned} A \times B &= A' \times B' - A' b_{W-1} b_{\frac{W}{m}-1} 2^{W-m} \\ &\quad - a_{W-1} a_{\frac{W}{m}-1} \sum_{j=0}^{\frac{W}{m}-1} b_j 2^{W-m+j} \end{aligned} \tag{4}$$

We illustrate this operation through an example in Figure 4.

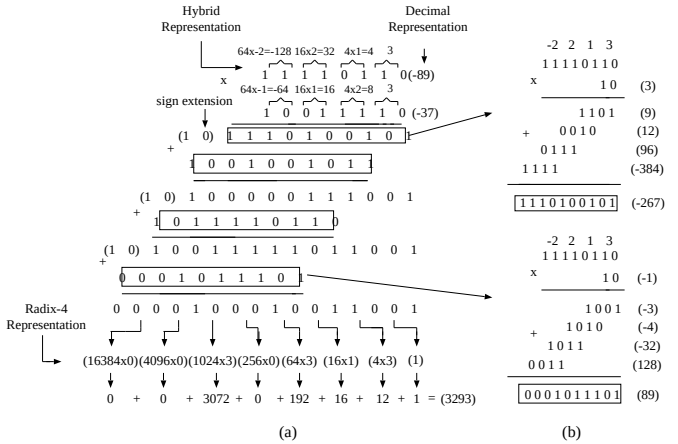

Figure 4. Example of a 2's complement 8-bit wide radix-4 multiplication.

Again, we have that 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.

We have constructed three types of modules. `Type I` are the unsigned modules used in the previous section. `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.

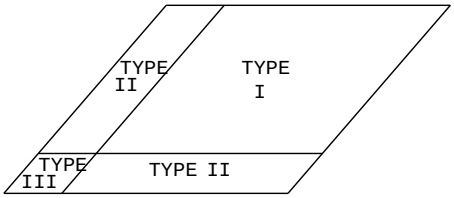

Figure 5. General structure for a 2's complement radix-2^m multiplier.

The general architecture for 2's complement radix-2^m multiplier is shown in Figure 5. We present a concrete example for $W = 8$ bit wide operands using radix-4 ($m = 2$) in Figure 6.

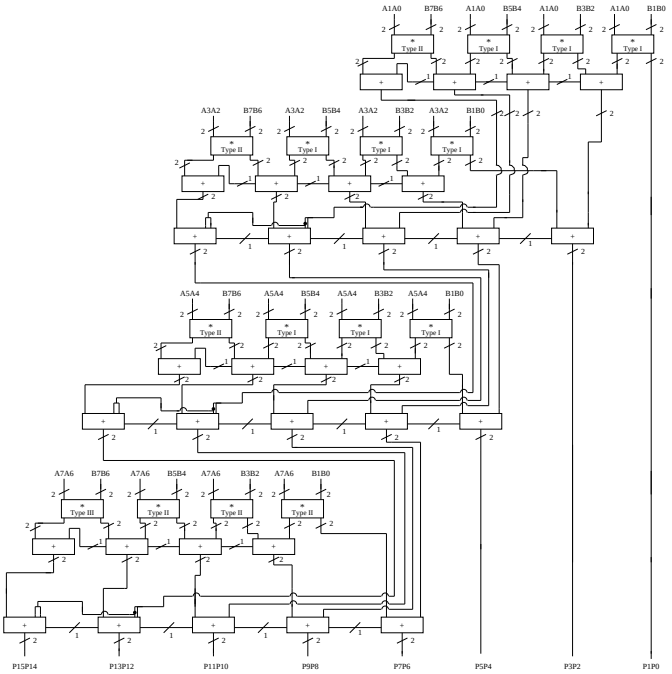

Figure 6. Example of a 8-bit wide 2's complement radix-4 array multiplier.

4.3 Modified Booth Multiplier

The radix-4 Booth's algorithm (also called Modified Booth) has been presented in [15]. In this architecture it is possible to reduce the number of partial products by encoding the two'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-b as shown in the example of Fig. 7 for a 8 bits wide operation. A multiplexer produces the partial product according to the encoded control signal.

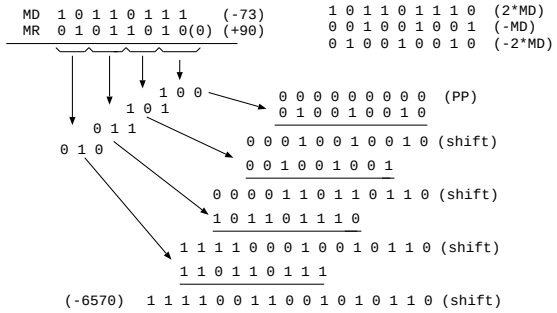

Figure 7. Example of a 8-bit wide Modified Booth multiplication.

Common to both architectures is that at each step of the algorithm two bits are processed. However, the basic Booth cells are not simple adders as in the proposed array multiplier, but must perform addition-subtraction-no operation and controlled left-shift of the bits on the multiplicand. Besides taking more area, this complexity also makes it more difficult to increase the radix value in the Booth architecture.

Note that the partial product terms are sign extended up. To make the array more regular a sign extension technique is used where it is used two extra bits in the partial product.

4.3.1 Modified Booth Architecture

One of the main problems in the Modified Booth multiplier is the generation of 2's complement for the multiplicand term. This is calculated by each operand circuit shown in the example of Fig. 8 for a 8-bit wide. As can be observed in Figure 8, for a 8-bit architecture, 4 operand circuits are necessary in order to calculate the partial product terms. These circuits are composed by an encoder and a multiplexer which produces the multiplicand term according to the 3-bit in the multiplier term (MR). The partial product terms are shifted by the adders which are also used to produce the final result.

5 Performance Comparisons

In this section, we present results for $W = 16$-bit array architectures. Radix-4 Booth and our version using radix-4 (m=2) are compared. Area and delay results were obtained in the SIS environment [16]. Area results are presented in terms of the number of literals. Delay results were obtained using the general delay model from the `mcnc` library. Power results were obtained with the SLS tool [17] using the general delay model. For the power simulation

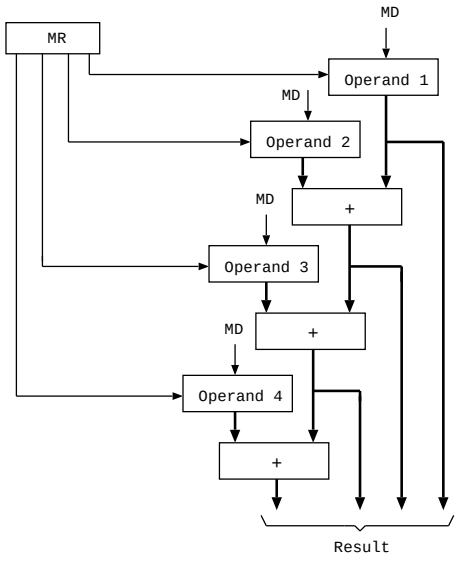

Figure 8. Example of a 8-bit wide pipelined Modified Booth architecture.

we have applied a random pattern signal, with 10,000 input vectors.

5.1 2's Complement Array Multipliers

In the Booth multiplier, an encoding scheme is used in order to reduce the number of stages in multiplication. Two bits of multiplication are performed at once and thus the multiplier requires half the stages. In our proposed 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. Table 3 presents area, delay and power results for radix-4 Booth multiplier and the proposed m=2 Hybrid array multiplier.

Table 3. AREA, DELAY AND POWER FOR 16-BIT 2'S COMPLEMENT PARALLEL MULTIPLIERS (m=2).

	Area	%	Delay (ns)	%	Power (mW)	%
Array	5316	–	231.6	–	94.1	–
Booth	3912	-26.4	233.7	+0.9	137	+45.6

As can be observed in Table 3, the m=2 array multiplier presents the highest area value. This occurs due to the fact that the partial product lines operate on group of m bits and the basic multiplier elements which composes the modules for the product terms are slightly more complex.

The reduction of partial product lines is an important issue for performance improvements. As can be observed in Table 3 m=2 Hybrid and Booth architectures present almost the same delay values. However, the m=2 Hybrid architecture presents a slightly improvement. This is due to the simpler structure presented by the proposed architec-

ture. The recoding and partial product selector logic used at each level in the Booth algorithm produces a larger number of interconnections and longer delay per row. Moreover, it should be observed that in the Modified Booth the first line to be added can be composed by two negative numbers, so that one of them needs to be complemented by a specific circuit. This circuit is composed by an additional line of half adders that is localized in the circuit for the generation of operand 1 shown in the Figure 8. Thus, this additional line also contributes for the higher delay value in the Modified Booth.

As observed in [18], the major sources of power dissipation in multipliers are spurious transitions and logic races that flow through the array. Thus, the larger number of interconnections present in the Booth multiplier is responsible for the generation of a significant amount of glitching in the circuit which justifies such a large gain in power for our approach as observed in Table 3. To confirm this, we have performed a power estimation of these two architectures using a zero-delay model and the values obtained were about the same.

Besides, although the radix-4 Booth multiplier presents a quite rectangular architecture, the regular structure presented by the m=2 Hybrid array multiplier makes him suitable for power reduction. Thus, the regularity characteristic presented by the m=2 Hybrid array multiplier makes this architecture consume significantly less power than Booth multiplier for random pattern signals as can be observed in Table 3.

6 Conclusions

We have presented an array architecture multiplier that operates on 2's complement numbers using radix-2^m encoding. The structure of this array maintains the same level of regularity as the normal array multiplier, yet since each partial product handles m bits at a time, the number of levels in the array is reduced by a factor of m. In the radix-2^m architecture, we have used an Hybrid code for the m-bit digit. As was observed, this code is an alternative way to implement regular multiplier architectures operating in 2's complement representation.

The radix-2^m array multiplier has been used before in a similar manner in the well-known Booth architecture. However, the Booth multiplier implies some overhead in terms of coding to handle the sign bit. The results demonstrate that because of our simpler architecture, 2's complement multiplication can be performed with almost half the power of a radix-4 Booth multiplier.

The regularity of our architectures make them suitable for applying other reducing power techniques. Thus, as future work we hope test the use of more efficient full adders in our architectures, in order to reduce the critical path. We also hope to investigate the use of our approach in a serial implementation in order to verify the trade-off between higher performance and power reduction.

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 Conference on Signals, Systems and Computers*, volume 1, pages 545–549, 1994.

[3] B. Cherkauer and E. Friedman. A Hybrid Radix-4/Radix-8 Low Power, High Speed Multiplier Architecture for Wide Bit Widths. In *IEEE International Symposium on Circuits and Systems*, volume 4, pages 53–56, 1996.

[4] A. Goldovsky and et al. Design and Implementation of a 16 by 16 Low-Power Two's Complement Multiplier. In *IEEE International Symp. on Circuits and Systems*, volume 5, pages 345–348, 2000.

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

[6] Y. Wang, Y. Jiang, and E. Sha. On Area-Effi cient Low Power Array Multipliers. In *The 8th IEEE International Conference on Electronics, Circuits and Systems*, pages 1429–1432, 2001.

[7] E. da Costa, J. Monteiro, and S. Bampi. Power Optimization Using Coding Methods on Arithmetic Operators. In *IEEE Int. Symp. on Signals Circuits and Systems*, pages 505–508, 2001.

[8] K. Hwang. *Computer Arithmetic - Principles, Architecture and Design*. School of Electrical Engineering, 1979.

[9] K. Pekmestzi. Multiplexer-Based Array Multipliers. *IEEE Transactions on Computers*, 48:15–23, 1999.

[10] M. Stan and W. Burleson. Low-Power Encodings for Global Communication in CMOS VLSI. *IEEE Trans. on VLSI Systems*, March 1997.

[11] E. Costa, J. Monteiro, and S. Bampi. Power Effi cient Arithmetic Operand Encoding. In *Proceedings of the XIV Symp. on Integrated Circuits and Systems Design*, September 2001.

[12] P. Meier, R. Rutenbar, and L. Carley. Inverse Polarity Techniques for High-Speed/Low-Power Multipliers. In *IEEE International Symposium on Low Power Electronics and Design*, pages 264–266, 1999.

[13] P. Capello and K. Steiglitz. A VLSI Layout for a Pipelined Dadda Multiplier. *ACM Transactions on Computers Systems*, pages 157–174, 1983.

[14] C. Efstathiou and H. Vergos. Modifi ed Booth 1's Complement and Modulo 2^n-1 Multipliers. In *The 7th IEEE International Conference on Electronics, Circuits and Systems*, volume 2, pages 637–640, 2000.

[15] 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.

[16] E. Sentovich and et al. SIS: A System for Sequential Circuit Synthesis. Technical report, University of California at Berkeley, UCB/ERL - Memorandum No. M92/41, 1992.

[17] A.J. Genderen. SLS: An Effi cient Switch-Level Timing Simulator Using Min-Max Voltage Waveforms. In *VLSI Conference*, pages 79–88, 1989.

[18] T. Callaway and E. Swartzlander. Optimizing multipliers for WSI. In *Fifth Annual IEEE International Conference on Wafer Scale Integration*, pages 85–94, 1993.