A New Pipelined Array Architecture for Signed Multiplication

Eduardo Costa,
UCPel, Pelotas, Brazil
ecosta@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 which maintains the pure form of an array multiplier, exhibiting a much lower overhead than the Booth architecture. This 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. We have implemented a pipelined version of the radix-4 architecture in order to reduce both the critical path and useless signal transitions that are propagated through the array. The performance of our pipelined architecture is compared with the pipelined Modified Booth. The results we present show that the proposed architecture with radix-4 compares favorably in performance and power with the Modified Booth multiplier in the pipelined and non-pipelined approaches.

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], [7], [8]. 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. This is particularly true for operands using 16 bits or more [9]

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 a sign bit. The proposed architecture is more efficient than the original Booth architecture because only one bit is examined for each bit product and no encoding is necessary. The regularity of the proposed architecture makes it naturally applicable for generic radix-2^m operations. We simply replace each bit product by m-bit modules that compute the partial products between m bits.

We compare the radix-4 Modified Booth architecture to our architecture with $m = 2$. The results show that our architecture is significantly more efficient, with no delay penalties and 54% 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. In order to reduce the amount of glitching along the array we have implemented pipelined architectures for both Modified Booth and our architecture with $m = 2$. The results show that although the power reduces more for the Modified Booth, the proposed multiplier is still more efficient, with almost 23% less power and almost 16% lower delay.

This paper is organized as follows. The next section makes an overview of relevant work related to our own. In Section 3 we present the proposed architecture to handle signed operands. Section 4 describes how this architecture can be directly extended for radix-2^m operation and the architectures with the pipelined strategy. 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. Early multiplier schemes such as bi-section, Baugh-Wooley and Hwang [10] propose the implementation of a 2's complement architecture, using repetitive modules with uniform interconnection patterns. In these works, partial products with negative sign are moved to the last steps. Thus, two types of arrays are used for the sum of the partial products. While in the first type the array presents a regular structure, the second one does not permit an efficient VLSI realization due to the irregular tree-array form used. Furthermore, it is difficult to improve the multiplier's performance or to reduce its power consumption.

More regular and suitable multiplier designs based on the Booth recoding technique have been proposed [2], [3], [7]. 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 [11] and Dadda parallel counters [12] have been applied. However, it is ob-

served that when the multiplier operands are wide enough these techniques may not be adequate for effective reduction of the multiplier latency [4]. When arithmetic throughput is more important than latency, pipelined multipliers are useful [13] because the introduction of registers along the array reduces the unnecessary activity. In [14], it is observed that latches are inserted in a radix-4 Booth multiplier to reduce the switching activity. According to the authors 40% of power savings is achieved with a power supply of 3V. In [15], the influence of transition activity due to combinational logic, flip-flops and the clock line is analyzed in array multipliers and an optimal level of pipelining is hinted.

In our work, 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. The use of our proposed array multiplier in a pipelined implementation in order to verify the trade-off between higher performance and power reduction is also investigated.

3 Parallel 2's Complement Architecture

In this section we describe how we derive the proposed architecture for a signed array multiplier.

3.1 2's Complement Binary Multiplication

2's complement is the most used encoding for signed operands. The most significant bit, a_{W-1}, is the sign bit. If the number A is positive, its representation is the same as for an unsigned number, simply A. If the number is negative, it is represented as $2^W - A$.

Conversely, the value of the operand can be computed as follows:

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

We make the following observation that enables us to simplify our architecture. Let us define $A' = \sum_{i=0}^{W-2} a_i 2^i$, an unsigned value. For positive numbers, $a_{W-1} = 0$, hence the value represented by A is A'. For negative numbers, $a_{W-1} = 1$, hence this value is $A - 2^W = 2^{W-1} + A' - 2^W = A' - 2^{W-1}$. Then Equation 1 becomes:

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

or simply $A = A' - a_{W-1}2^{W-1}$.

What Equation 2 tells us is that the multiplication of two operands in 2's complement can be performed as an unsigned multiplication for $(W-1)^2$ of the bit products. Let us consider the 4 possible scenarios for $A \times B$:

$$\begin{array}{ll} A > 0, B > 0: & A' \times B' \\ A > 0, B < 0: & A' \times B' - A'2^{W-1} \\ A < 0, B > 0: & A' \times B' - \sum_{j=0}^{W-1} b_j 2^{W-1+j} \\ A < 0, B < 0: & A' \times B' - A'2^{W-1} - \sum_{j=0}^{W-1} b_j 2^{W-1+j} \end{array} \tag{3}$$

which can be reduced to

$$A \times B = A' \times B' - b_{W-1}A'2^{W-1} - a_{W-1} \sum_{j=0}^{W-1} b_j 2^{W-1+j} \tag{4}$$

The form of Equation 4 highlights:

- from the first term, that the $W-1$ least significant bits of A and B can be treated exactly as an unsigned array multiplier;
- from the second term, that the last row of the multiplier is either non-existent ($B > 0$) or a subtracter of A' shifted by $W-1$ bits ($B < 0$);
- from the third term, that, at each partial product line, the most significant bit is either 0 ($A > 0$) or -1 ($A < 0$).

We illustrate the operation of an array multiplication of 4-bits wide operands in 2's complement in Figure 1.

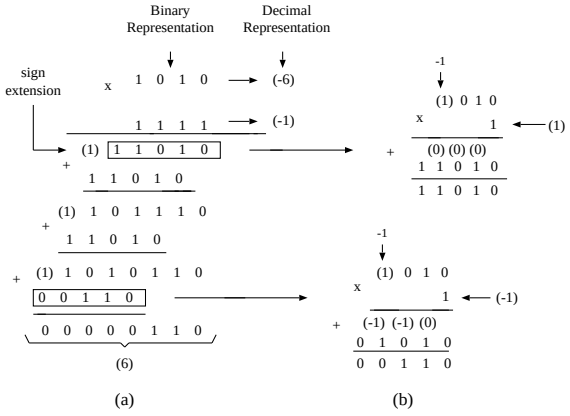

Figure 1. Example of a $W = 4$ bit wide signed multiplication.

Therefore, we can construct an array multiplier that handles signed operands simply by using slightly different elements at the left and bottom of the array. We present this architecture in Figure 2 for 4-bit operands. Note that to keep the figure simple we are using $W = 4$ and for such simple cases the signed elements make for a significant fraction of the array. This is because for $W = 4$ we have a total of 16 bit products of which 9 are unsigned and 7 signed. However, this ratio, $(W-1)^2$ vs. $2W-1$, increases with W. In the case of a 16-bit multiplier, we have 225 unsigned bit products and 31 signed.

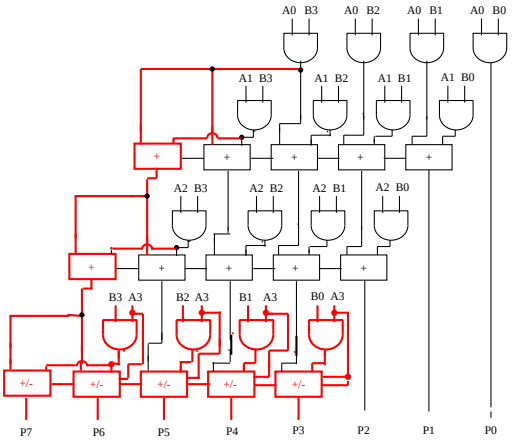

Figure 2. Example of a $W = 4$ bit wide signed array multiplier.

3.2 Comparison with the Booth Architecture

A 2's complement multiplication algorithm which verify the detection of the beginning, the interior and the ending of a string of ones was proposed in [16]. In the original Booth algorithm 2 bits are operated at a time in an overlapped scanning form. Thus, in a multiplier term the allowed digit set is (-1, 0, 1) which composes a radix-2 representation. This digit set is used to adjust the multiplicand terms. An encoding of the multiplicand is used to simplify this partial product. In our algorithm only the most significant bit from the multiplier and multiplicand terms is tested for the signed operation. Thus, most of the structure in the partial product terms are in the same form as in the conventional array multiplier and no encoding is necessary.

4 Higher Radices Architectures

In addition to 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.

4.1 2's Complement Radix-2^m Multiplication

We demonstrate that all the observations made in Section 3.1 apply to any radix we choose. 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:

$$\begin{array}{ll} A' & , \quad a_{W-1} = 0 \\ A' - a_{\frac{W}{m}-1} 2^{W-m} & , \quad a_{W-1} = 1 \end{array} \tag{5}$$

or simply

$$A' - a_{W-1} a_{\frac{W}{m}-1} 2^{W-m} \tag{6}$$

Using analogous observations as made for the binary case, from Equation 6 we can write:

$$\begin{aligned} A \times B \quad = \quad & A' \times B' - A' b_{W-1} b_{\frac{W}{m}-1} 2^{W-m} \\ & - a_{W-1} a_{\frac{W}{m}-1} \sum_{j=0}^{\frac{W}{m}-1} b_j 2^{W-m+j} \end{aligned} \tag{7}$$

We illustrate this operation through an example in Figure 3.

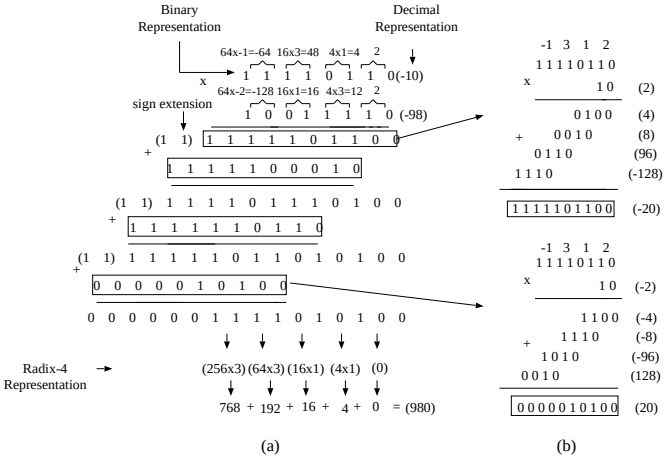

Figure 3. Example of a 2's complement 8-bit wide radix-4 multiplication.

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

Similarly to the binary, 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.

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

The regularity of our architecture makes it suitable for the application of other power reducing techniques. We have constructed a pipelined version in order to reduce the

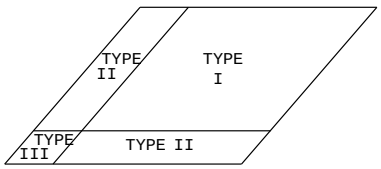

Figure 4. General structure for a 2's complement radix-2^m multiplier.

critical path and useless signal transitions that are propagated through the array. The dotted lines in Figure 5 shows the pipelined version of the radix-4 array multiplier for 8-bit operands. As can be observed, we have taken advantage of the layered structure of the array and introduced on the circuit two layers of registers along the array. Thus, 3 clock cycles are necessary to perform the computation. As presented in next section, the introduction of the registers reduces the delay and the power consumption, however the area is significantly increased.

Note that the results obtained with this pipelined implementation can be extrapolated to a serial implementation, where almost no glitching is produced.

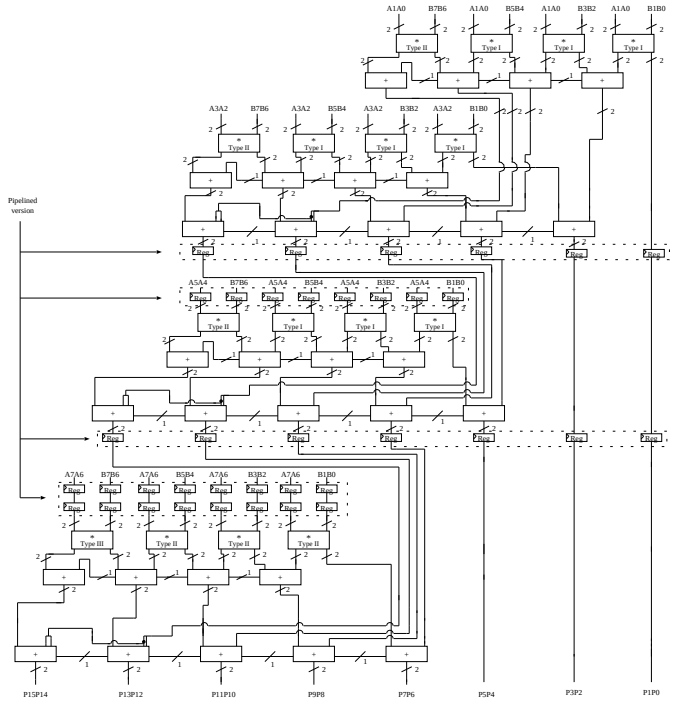

Figure 5. Example of a 8-bit wide 2's complement radix-4 array multiplier.

4.3 Modified Booth Multiplier

The radix-4 Booth's algorithm (also called Modified Booth) has been presented in [17]. 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. 6 for 8-bit operands. A multiplexer produces the partial product according to the encoded control signal.

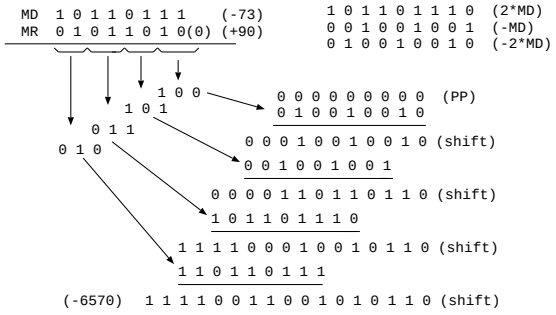

Figure 6. Example of a 8-bit wide Modifi ed Booth multiplication.

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 the 2's complement for the multiplicand term. This is calculated for each operand as shown in the example of Figure 7. For 8-bit operands, 4 operand circuits are necessary in order to calculate the partial product terms. These circuits are composed of 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.

We have implemented a pipelined Modified Booth by introducing registers along the layers of the array as shown in the dotted lines in Figure 7. As can be observed, we have added two layers of registers along the array as in the binary array multplier with m=2. Again, 3 clock cycles are required to compute the final result. Moreover, common to both architectures is that the registers are inserted at the output of the adders which are responsible for adding the partial product terms. However, as can be observed in Figure 7, in the Booth multiplier it is also necessary to introduce registers in the output of the encoders in order to perform the correct operation at each clock cycle.

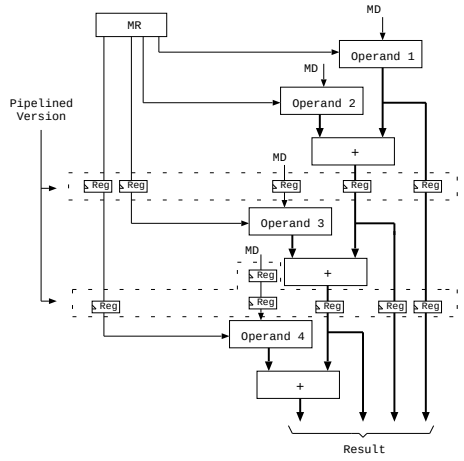

Figure 7. Example of a 8-bit wide Modifi ed Booth architecture.

5 Performance Comparisons

In this section, we present results for $W = 16$-bit array architectures. Radix-4 Booth and the proposed architecture using radix-4 (m=2) are compared. Area and delay results were obtained in the SIS environment [18]. 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 [19] using the general delay model. For the power simulation we have applied a random pattern signal and a *real trace* input signal which represent two sinusoidal signals with 90 degree phase difference, both with 10,000 input vectors.

5.1 2's Complement Array Multipliers

In the Booth multiplier, 2 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 by more than half while keeping the same regularity as in the pure array multiplier circuit. Table 1 presents area, delay and power results for radix-4 Booth multiplier and the proposed m=2 binary array multiplier.

Table 1. Area, delay and power for 16-bit 2's complement parallel multipliers.

	Area	%	Delay	%	Power	%
Array	4912	–	231.1ns	–	89mW	–
Booth	3912	-20.2	233.7ns	+1.1	137mW	+54

As can be shown in Table 1, the m=2 array multiplier presents larger area. This due to the fact that the partial product lines operate on groups of $m = 2$ bits and the basic multiplier elements are slightly more complex. From the same table, we can also see that the m=2 binary and Booth architectures present almost the same delay values. This is justified by the simpler structure presented by the proposed architecture, which leads to a lower logic depth. The recoding and partial product selector logic used at each level of the Booth algorithm produces a larger number of interconnections and longer delay per row.

As observed in [20], 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 1. To confirm this observation, we have performed a power estimation of these two architectures using a zero-delay model and the values obtained were about the same.

Additionally, although the radix-4 Booth multiplier presents quite a rectangular architecture, the regular structure presented by the m=2 multiplier makes it consume significantly less power for sinusoidal pattern signals as can be observed in Table 1.

5.2 Pipelined Array Multipliers

Table 2 presents results for the Booth multiplier and the proposed m=2 binary array multiplier in the pipelined version.

Table 2. Area, delay and power for 16-bit pipelined parallel multipliers.

	Area	%	Delay	%	Power	%
Array	6631	–	111.5ns	–	58.6mW	–
Booth	5676	-14.4	128.5ns	+15.2	69.6mW	+18.7

The introduction of registers along the layers of the arrays increase the area of both architectures when compared to the non-pipelined architectures. Moreover, the pipelined m=2 array multiplier presents the highest area value due to the higher complexity presented by the modules which process the product terms.

Figures 5 and 7 show that while in the binary array multiplier the critical path is given by a m=2 multiplier module and 2 full adders, in the Modified Booth the encoder, an operand circuit composed by a multiplexer and a full adder are present in the critical path. These circuits produce a large number of interconnections and longer delay per row. Thus, although the pipelined radix-2^m binary multiplier is larger, this architecture presents less delay values than the Modified Booth. Additionally, it should be observed that in the Modified Booth the first line to be added can be composed of two negative numbers, so that one of them needs to be complemented by a specific circuit. This circuit is composed of an additional line of half adders that is localized in the circuit for the generation of operand 1 shown

in the Figure 7. This additional line also contributes for the higher delay value in the Modified Booth. This is on top of the observation made before that our architecture presents less logic depth due to the more balanced paths to the basic blocks in the critical path. This fact contributes for improvement in power reduction because of the less generation of useless transitions as shown in Table 2 for a sinusoidal input pattern.

Table 2 shows that although our architecture consumes less power, it should be observed that the Modified Booth reduces more power when compared to non-pipelined version. This occurs because in the pipelined approach glitching is reduced significantly, and this reduction will have a greater impact in the case where the glitching was higher. However, the less logic depth and delay values presented by our architecture still makes it significantly more efficient, for a sinusoidal signal as shown in Table 2. For a random pattern at the inputs of the multipliers, where signal correlation is not present, power savings up to 20% are achievable in the m=2 multiplier as shown in Table 3.

Table 3. Power for random pattern signal.

Architectures	Power	%
Array	100.4mW	–
Booth	123.2mW	+22.7

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. We have presented results that show significant improvement in delay and power. 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 just half the power of a radix-4 Booth multiplier.

The regularity of our architecture make it suitable for applying other reducing power techniques. In this work we were able to test the use of pipelining approach in order to reduce the critical path and useless signal transitions that are propagated through the array. As we observed, our multiplier is more efficient than the Modified Booth due to the lower logic depth that reduces the amount of glitching along the circuit.

7 Acknowledgment

This research was supported in part by CAPES-Brazil and in part by FCT-Portugal under project POCTI.

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 ISCAS*, volume 4, pages 53–56, 1996.

[4] 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.

[5] 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.

[6] Z. Yu, L. Wasserman, and A. Willson. A Painless Way to Reduce Power by Over 18% in Booth-Encoded Carry-Save Array Multipliers for DSP. In *Workshop on Signal Processing Systems*, pages 571–580, 2000.

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

[8] 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.

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

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

[11] P. Meier, R. Rutenbar, and L. Carley. Inverse Polarity Techniques for High-Speed/Low-Power Multipliers. In *IEEE ISLPED*, pages 264–266, 1999.

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

[13] C. Asato, C. Ditzen, and S. Dholakia. A Data-Path Multiplier with Automatic Pipeline Stages. *IEEE JSSC*, 25:383–387, 1990.

[14] C. Lemmonds and S. Shetti. A low power 16 by 16 multiplier using transition reduction circuitry. In *International Worshop on Low Power Design*, pages 139–142, 1994.

[15] J. Leijten, J. van Meerbergen, and J. Jess. Analysis and reduction of glitches in synchronous networks. In *EDAC*, 1995.

[16] A. Booth. A Signed Binary Multiplication Technique. *J. Mech. and Applied Mathematics*, (4):236–240, 1951.

[17] I. Khater, A. Bellaouar, and M. Elmasry. Circuit Techniques for Low-Power High-Performance Multipliers. *IEEE JSSC*, 31:1535–1546, 1996.

[18] 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.

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

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