A New Architecture for Signed Radix-2^m Pure Array Multipliers

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 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. The flexibility of our architecture allows for the easy construction of multipliers for different values of m, as opposed to the Booth architecture for which implementations for $m > 2$ are complex. The results we present show that the proposed architecture with radix-4 compares favorably in performance and power with the Modified Booth multiplier. We have experimented our architecture with different values of m and concluded that $m = 4$ minimizes both delay and power.

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 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 present results that show that the delay and power decrease for $m = 2$ and $m = 4$. From $m = 1$ to $m = 2$ we obtain a 18% performance improvement and 57% power savings, with an area penalty of less than 10%. On the other hand, from $m = 1$ to $m = 4$ we obtain a 28% performance improvement and 72% power savings at cost of an expressive area penalty.

We compare the Modified Booth architecture, which uses radix-4, to our architecture with $m = 2$. The results show that the proposed 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 as 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 the proposed architecture to handle signed operands. Section 4 describes how this architecture can be directly extended for radix-2^m operation. Performance comparisons between different multiplier architectures, namely different values of m 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 [7] 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 [8], where a scheme of a multiplexer-based multiplier is presented. In [6] an improvement of this technique is observed where the architecture has a more rectangular layout than [8].

Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02)
1063-6404/02 $17.00 © 2002 IEEE

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. 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.

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 [2, 5] high performance multipliers based on higher radices are proposed. However, these circuits have little regularity and no power savings are reported. Research work that directly targets power reduction by using higher radices for the Booth algorithm is presented in [3, 4]. Area, delay and power improvements are reported with a highly optimized encoding scheme at the circuit level. At this level of abstraction some other works have applied complementary pass-transistor logic in their design in order to improve the Booth encoder and full adder circuits [9, 10, 11].

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. We show that our architecture can be more naturally extended for higher radices, using less logic levels and hence presenting much less spurious transitions. We have not applied yet any transistor-level techniques which can further improve the efficiency of the architecture.

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

Consider two operands W-bits wide, $A = \sum_{i=0}^{W-1} a_i 2^i$ and $B = \sum_{j=0}^{W-1} b_j 2^j$. We have that

$$A \times B = \sum_{j=0}^{W-1} A \cdot b_j 2^j \tag{1}$$

where in turn,

$$A \cdot b_j = \sum_{i=0}^{W-1} b_j \cdot a_i 2^i \tag{2}$$

A conventional array multiplier [7] translates this expression directly to hardware, where we have the W partial product rows from Equation 1, each made of W bit level products as in Equation 2, which can be arranged in a simple, very regular, array structure. Each bit product is simply an AND gate.

The conventional array architecture is only applicable to unsigned operands. We are able to show that exactly the same architecture can be used on signed operands in 2's complement with very little changes.

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:

$$A = \begin{cases} A & , \quad a_{W-1} = 0 \\ A - 2^W & , \quad a_{W-1} = 1 \end{cases} \tag{3}$$

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 3 becomes:

$$A = \begin{cases} A' & , \quad a_{W-1} = 0 \\ A' - 2^{W-1} & , \quad a_{W-1} = 1 \end{cases} \tag{4}$$

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

What Equation 4 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 $A \times B$:

$$\begin{array}{ll} A > 0, B > 0: & A' \times B' \\ A > 0, B < 0: & A' \times B' - A' 2^{W-1} \\ A < 0, B > 0: & A' \times B' - \sum_{j=0}^{W-1} b_j 2^{W-1+j} \\ A < 0, B < 0: & A' \times B' - A' 2^{W-1} - \sum_{j=0}^{W-1} b_j 2^{W-1+j} \end{array} \tag{5}$$

which can be reduced to

$$A \times B = A' \times B' - b_{W-1} A' 2^{W-1} - a_{W-1} \sum_{j=0}^{W-1} b_j 2^{W-1+j} \tag{6}$$

The form of Equation 6 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.

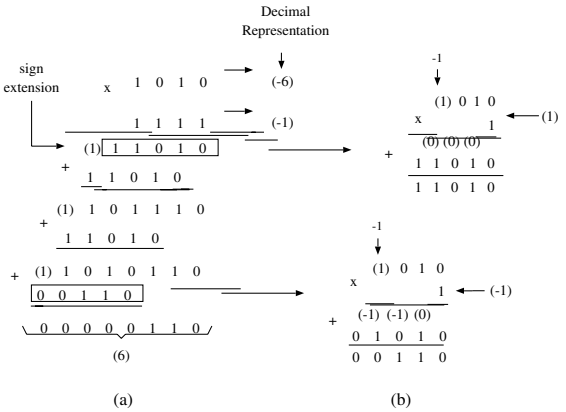

Figure 1. Example of a $W = 4$ bit wide signed multiplication.

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.

4 Higher Radices 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.

4.1 Unsigned 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

$$A \times B = \sum_{j=0}^{\frac{W}{m}-1} A \cdot b_j 2^{j \cdot m}, \qquad A \cdot b_j = \sum_{i=0}^{\frac{W}{m}-1} b_j \cdot a_i 2^{i \cdot m} \quad (7)$$

We illustrate this operation for operators with $W = 8$ bits using radix-16 ($m = 4$) in Figure 3. 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 3(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 3(a). Also exemplified in Figure 3 is the conversion of radix-16 numbers to decimal values.

4.2 Unsigned Radix-2^m Multiplier Architecture

The structure of the radix-2^m multiplier architecture is the same as the plain array multiplier. However, each partial product line operates on groups of m bits instead of a single bit. This reduces the number of product lines to $\frac{W}{m}$. Although the operation of each line is more complex, there is some room for the optimization of the partial product generation modules which enables the performance and power improvement shown in this paper.

Returning to our example, we present in Figure 4 the radix-16 (m=4) array multiplier architecture for $W = 8$ bits wide operators.

As can be observed in Figure 4, for a W-bit multiplier we require $\frac{W}{m}$ lines each with $\frac{W}{m}$ basic modules of m by m multipliers and the same number of m by m adders. An additional one line composed of $m+1$ of these basic adders

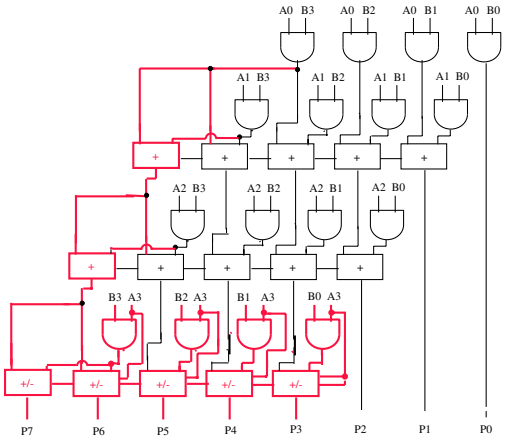

Figure 2. Example of a $W = 4$ bit wide signed array multiplier.

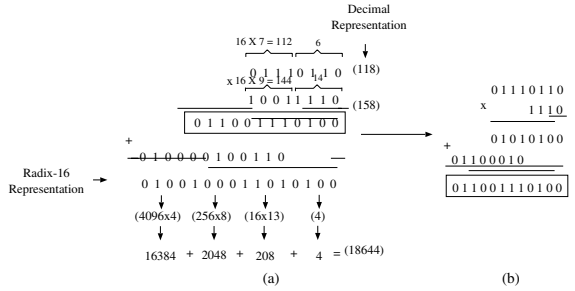

Figure 3. Example of a 8-bit wide radix-16 multiplication.

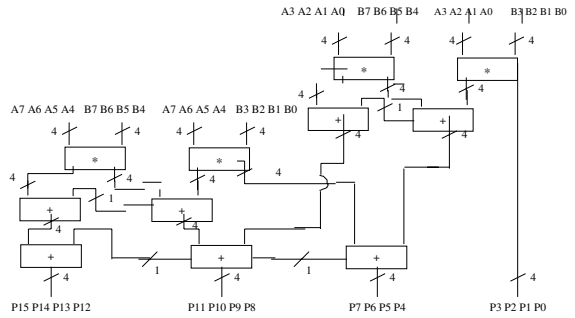

Figure 4. 8-bit wide radix-16 multiplier architecture.

is responsible for summing the partial product terms. This structure can accommodate any combination of values for W and m. The regularity of this structure has allowed us to implement a simple program that takes these two parameters and generates the corresponding radix-2^m array multiplier. To this end, highly optimized basic adder and multiplier modules have been designed for $m = 2$ and 4. In our architecture, the larger the value of m, the less the number of partial product lines, however the more complex the basic adder and multiplier modules will be.

4.3 2's Complement Radix-2^m Multiplier Architecture

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:

$$A = \begin{cases} A' & , \quad a_{W-1} = 0 \\ A' - a_{\frac{W}{m}-1} 2^{W-m} & , \quad a_{W-1} = 1 \end{cases} \tag{8}$$

or simply

$$A = A' - a_{W-1} a_{\frac{W}{m}-1} 2^{W-m} \tag{9}$$

Using analogous observations as made for the binary case, from Equation 9 we can write:

$$\begin{aligned} A \times B &= A' \times B' - A' b_{W-1} b_{\frac{W}{m}-1} 2^{W-m} \\ &\quad - a_{W-1} a_{\frac{W}{m}-1} \sum_{j=0}^{\frac{W}{m}-1} b_j 2^{W-m+j} \end{aligned} \tag{10}$$

We illustrate this operation through an example in Figure 5.

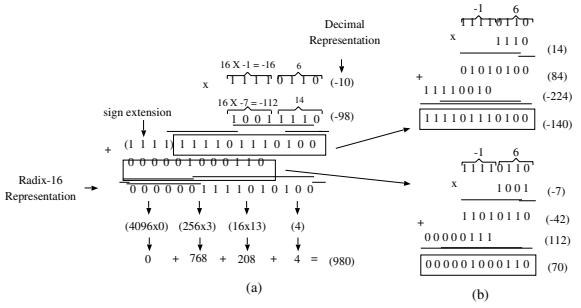

Figure 5. Example of a 2's complement 8-bit wide radix-16 multiplication.

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.

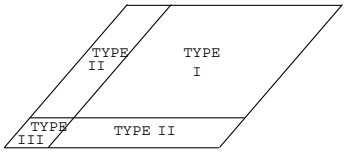

Figure 6. General structure for a 2's complement radix-2^m multiplier.

The general architecture for 2's complement radix-2^m multiplier is shown in Figure 6. We present a concrete example for $W = 8$ bit wide operands using radix-16 ($m = 4$) in Figure 7.

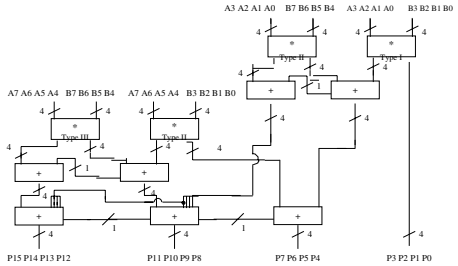

Figure 7. Example of a 8-bit wide 2's complement radix-16 array multiplier.

5 Performance Comparisons

In this section, we first compare area, delay and power of $W = 16$-bit array multipliers for groups of m=1, 2 and 4. Radix-4 Booth and the proposed architecture using radix-4 (m=2) are compared next. Area and delay results were obtained in the SIS environment [12]. 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 [13], a switch-level simulator, using the general delay model. For the power simulation we have applied both a *real trace* input signal and a random pattern signal, both with 10,000 input vectors. The *real trace* signal represent two sinusoidal signals with 90 degree phase difference.

5.1 Signed Array Multipliers Using Different Radices

Although the higher radices architectures require less basic multiplier elements than those used by the conventional array architecture (m=1), each basic multiplier element is composed of more logic gates. Therefore, the new array multipliers (m=2 and m=4) present higher area than the conventional m=1 binary array multipliers as shown in Table 1. This table also shows that the complexity of these basic modules increases very rapidly with m.

Table 1. Area and delay for 16-bit radix-2^m binary multipliers.

Group of bits	Binary			
	Literals	%	Delay	%
m=1	4266	–	284ns	–
m=2	4674	+9.5	231ns	-18.6
m=4	10544	+147.1	202ns	-28.8

Though the radix-2^m array multipliers are larger, these architectures present less delay values than the m=1 binary multiplier, as shown in Table 1. As can be seen in Figure 7, the radix-2^m architectures need $\frac{W}{m}$ lines of adders responsible for adding the partial product terms. The last line adds or subtracts the product terms depending on the last bit of the multiplier term. In the m=1 binary architecture is required a total of $W - 1$ of these lines. Thus, although the basic m by m modules are more complex, by a careful design that minimizes logic depth we were able to reduce the critical path delay for the cases of $m = 2$ and 4.

Despite the higher area presented by the radix-2^m multipliers for different bit group sizes, these architectures consume significantly less power than the conventional m=1 binary architecture. Table 2 shows the power results comparison between the conventional binary array multiplier and the new architectures using a real trace signal and a random pattern. As can be observed in this table, the m=2 architecture can save almost 60% of power. For the m=4 architecture, power can be reduced above 70%. As can be also observed in Table 2, even for a random pattern at the inputs of the multipliers, where the correlation aspect of the signal is not present, similar results are obtained. Power savings above 50% and 60% are achievable in the m=2 and m=4 multipliers.

Table 2. Power for 16-bit radix-2^m binary multipliers.

Group of bits	Binary - Power			
	sine	%	random	%
m=1	134.0mW	–	201.0mW	–
m=2	56.5mW	-57.8	89.0mW	-55.7
m=4	36.4mW	-72.8	76.4mW	-61.9

5.2 Comparison with the Booth Multiplier

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 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 binary array multiplier.

Table 3. Area, delay and power for 16-bit 2's complement parallel multipliers.

	Area	%	Delay	%	Power	%
Array	4674	–	231ns	–	89mW	–
Booth	3912	-20.2	234ns	+1.3	137mW	+54.0

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

As observed in [14], 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 binary array multiplier makes him suitable for power reduction. Thus, the regularity characteristic presented by the m=2 binary 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. 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 two thirds the power of a radix-4 Booth multiplier.

According to our results, increasing the radix can improve the efficiency up to a certain point. Although the good results we have found were for $m = 2$ compared to Modified Booth, we could show that the modules for $m = 4$ present better results with delay and power reduction improvements. Such higher order radices are more difficult to implement with the Booth architecture.

The regularity of our architectures make them suitable for applying other reducing power techniques. As future work we hope test the use of pipelining and more efficient full adders in our architectures, in order to reduce useless signal transitions that are propagated into the array and 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 Trans. 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 Intl. Symp. 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 Symposium 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-Efficient Low Power Array Multipliers. In *The 8th IEEE International Conference on Electronics, Circuits and Systems*, pages 1429–1432, 2001.

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

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

[9] K. Yano and et al. A 3.8-ns CMOS 16×16-b Multiplier Using Complementary Pass-Transistor Logic. *Journal of Solid-State Circuits*, 25:388–395, 1990.

[10] G. Goto and et al. A 4.1-ns Compact 54×54-b Multiplier Utilizing Sign-Select Booth Encoders. *IEEE Journal of Solid-State Circuits*, 32:1676–1682, 1997.

[11] 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.

[12] 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.

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

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