Low-Power Fixed-Width Array Multipliers

Jinn-Shyan Wang
Institute of Electrical Engineering
Chung-Cheng University
160, San-Hsing,Ming-Hsiung,
Chia-Yi,Taiwan
+886-05-2720411 ext. 33202
ieegsw@ccu.edu.tw

Chien-Nan Kuo
Institute of Electrical Engineering
Chung-Cheng University
160, San-Hsing,Ming-Hsiung,
Chia-Yi,Taiwan
+886-05-2720411 ext. 23280
joseph@vlsi.ee.ccu.edu.tw

Tsung-Han Yang
Institute of Electrical Engineering
Chung-Cheng University
160, San-Hsing,Ming-Hsiung,
Chia-Yi,Taiwan
+886-05-2720411 ext. 23280
u8943041@ccu.edu.tw

ABSTRACT

A fixed-width multiplier using the left-to-right algorithm for partial-product reduction is presented. The high-speed feature offered by this design is used to trade for low power. In one design, the proposed multiplier not only owns 8% speed improvement but also gains 14% power and 13% area reduction. When applying the voltage scaling to balance the speed, the power reduction is increased to 29%.

Categories and Subject Descriptors

B.2.1 [**Arithmetic and Logic Structure**]: Design Style – *parallel.*

General Terms

Design.

Keywords

Left-to-right multiplier, fixed-width multiplier, reduced-width multiplier, low power.

1. INTRODUCTION

The multiplier is an important kernel of digital signal processors (DSP) because it typically determines the performance of the chips. Furthermore, because of high circuit complexity, the power consumption and the layout area are another two design considerations of the multiplier. In some of the DSP applications, precision can be sacrificed to improve the speed and to reduce the area. Therefore, several fixed-width or reduced-width multipliers [1]-[3] have been proposed for this purpose.

ISLPED'04, August 9–11, 2004, Newport Beach, California, USA

By removing some unnecessary gates in a full-width multiplier, the fixed- and reduced-width multipliers also gain the advantage of power reduction because of the reduced gate count. In this circumstance, how to reduce the error caused by removing part of the operation in a full-width multiplier becomes the main focus in the previous designs for fixed- and reduced-width multipliers.

Conventional full-width multipliers often adopt the right-to-left algorithm for summing the partial product terms into the final product. Recently, the left-to-right algorithm without a final carry propagation step [4]-[5] had been proposed in an attempt to improve the performance of the right-to-left algorithm. More recently, the work [6][7] used several kinds of compressors in the LSB parts of the partial product reduction array to further increase the operating speed of a left-to-right multiplier. However, to the best of our knowledge, no one fixed- or reduced-width multiplier adopts the left-to-right algorithm.

In this work, the design of a fixed-width multiplier based on the left-to-right algorithm is studied. For convenience of comparison, all the evaluated multipliers are designed in the cell-based approach. Different multipliers are coded in the structure-level Verilog language, and the performance change can be observed by applying different speed constraints during synthesizing. Our study shows that the performance of the proposed design, without using any compressors and even under default synthesis, is still superior to conventional fixed-width multipliers utilizing a fast final adder. On the other hand, if tighter speed constraints are applied to the conventional right-to-left fixed-width multipliers, the proposed left-to-right fixed-width multiplier gains more power saving.

The rest of this paper is organized as follows. Section 2 briefly reviews the background information about conventional fixed-width right-to-left multipliers and conventional full-width left-to-right multipliers. Section 3 describes the architecture design and the error analysis of low-power fixed-width left-to-right multipliers. Evaluation results are discussed in section 4, and conclusions are given in the last section.

2. BACKGROUND

2.1 Conventional right-to-left fixed-width multiplier

This section briefly reviews the design of a conventional right-to-left fixed-width multiplier. For designing a high-speed multiplier, one usually utilizes the modified Booth algorithm [8] to reduce the number of the rows of partial products. The block diagram of an 8×8 modified-Booth full-width array multiplier, without showing the Booth encoder, is depicted in Fig. 1. The partial-product reduction array can be replaced by a Wallace tree to shorten the summation time if the input bit-width is larger than 16. However, the complexity of the intra-signal connection will become much higher. Note that the operands are summed from right to left in a most intuitive way. In this case, the input bit width is 8, and the output bit width is 16. Therefore, this is called an 8×8 right-to-left full-width array multiplier.

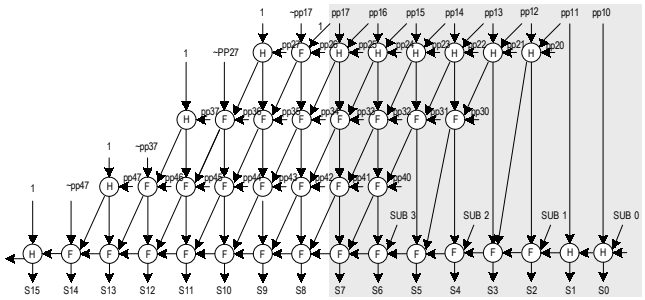


Figure 1. An 8×8 right-to-left full-width multiplier.

For designing a right-to-left fixed-width multiplier, those partial products, that enter the array from the right of the column generating S8, and the corresponding gates can be omitted. These terms are in the shadow region of Fig. 1. To reduce the accuracy loss induced from truncating part of the operands, one method is to add some compensation terms. Figure 2(a) shows the block diagram of such an 8×8 right-to-left fixed-width multiplier [3], where cells AO and the AND gate are the correction terms and the realized function of the AO cell is shown in Fig. 2(b).

A word worth mentioning is that in Fig. 2(a), a ripple-carry adder (RCA) is used to produce the final product. If the operating speed is expected to be higher, one easy way is to use a fast adder instead, such as the carry lookahead adder (CLA) or the carry select adder (CSA). In a cell-based design environment, we can use a tool such as the *DesignWare* to automatically choose a better architecture when setting a tighter constraint to squeeze the speed of the final adder.

2.2 Conventional left-to-right full-width multiplier

The basic idea of the left-to-right algorithm is to arrange the rows of the partial products up-side-down as opposed to a conventional right-to-left multiplier. Then, to utilize the advantage afforded by this arrangement, some on-the-fly converters should be added to the left of the original array. The block diagram of a left-to-right full-width multiplier is shown in Fig. 3(a) according to the design

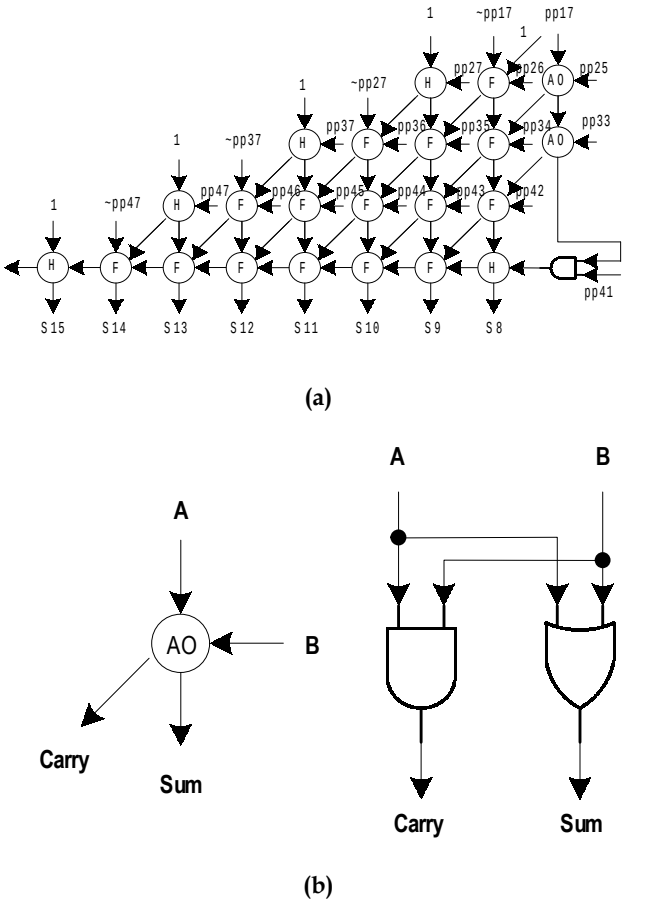
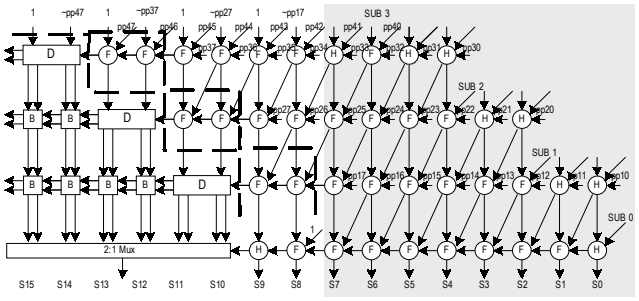


Figure 2. (a) An 8×8 right-to-left fixed-width multiplier and (b) the function of the AO cell.

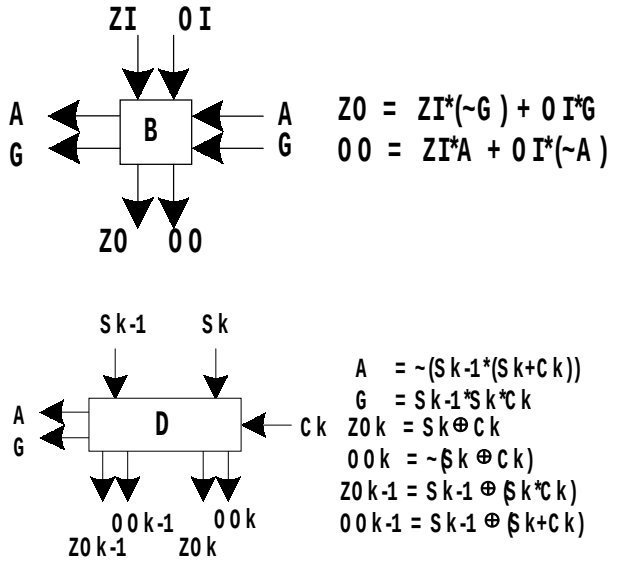
described in [5]. The circuit between the dashed lines is used to interface the added converters to the original array. The first diagonal row of type D cells is used to create the conditional forms of the product bits in that column. Type D cells also create the *Absorb* (A) and *Generate* (G) signals that are used to alter the conditional product bits in every higher bit position. The conditional products are updated as each product digit is reduced in carry-save form from the array composed of type B cells. Fig. 3(b) depicts the logic of both type D and type B cells. A 2:1 multiplexer is used to select between the two conditional forms of the product.

To see the advantage of the operating speed afforded by the left-to-right algorithm, we use a simplified gate delay model to estimate the critical path delay of the multiplication. The block diagrams of Fig. 1 and Fig. 3(a) are redrawn in Figs. 4(a) and 4(b), respectively, with the signal names removed for clarity. Instead,

the accumulated signal propagation delay is added beside each component. The propagation delay estimation of each building block is shown in Fig. 4(c). We find that the right-to-left array multiplier without a fast final adder needs 59 unit delays, while the left-to-right array multiplier with on-the-fly converters takes only 42 unit delays. Therefore, the 8×8 left-to-right full-width multiplier is about 28% faster than the right-to-left full-width multiplier, and this estimation is close to that reported in [5].



(a)



(b)

Figure 3. (a) The left-to-right full-width multiplier and (b) the functions of D cell and B cell.

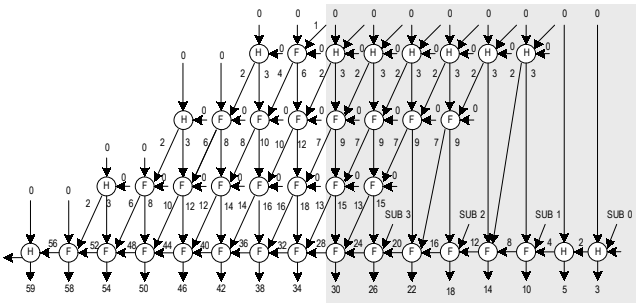
There still exist rooms for speed improvement for the basic left-to-right multiplier. To see why this is achievable, we observe that the outputs of the on-the-fly converter arrive at the multiplexer 11 unit delays earlier than the selection signal of the multiplexer. Based on this observation, the work in [6] proposed to use a strategic array of (3, 2), (5, 3), (7, 4) compressors to enhance the operation speed of the part of the least significant product (LSP) terms.

3. THE LEFT-TO-RIGHT FIXED-WIDTH MULTIPLIER

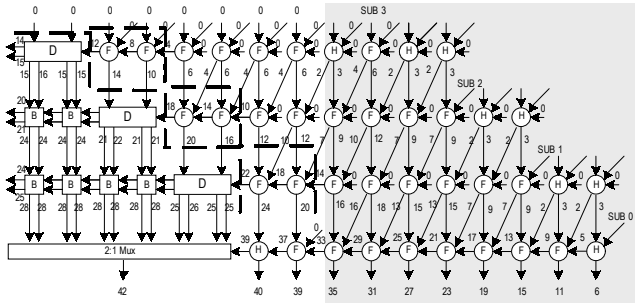
3.1 Selection of the architecture

Although there are several versions of left-to-right full-width multipliers proposed so far, we adopt the architecture of Fig. 3(a) to design a left-to-right fixed-width multiplier. The main reason is described as follows.

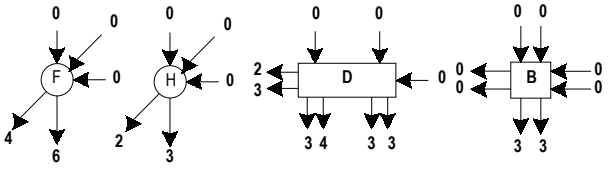
The speed bottleneck of the original left-to-right full-width multipliers comes from the LSP part, as described before. Therefore, the speed-up design [6] focused on reducing the signal propagation delay of the LSP part. However, in the design of fixed-width left-to-right multipliers, most gates lying in the LSP part will be deleted. Thus, the acceleration effect of the above design becomes insignificant for the fixed-width left-to-right multipliers.



(a)



(b)



(c)

Figure 4. (a) Right-to-left, (b) left-to-right full-width multipliers, and (c) propagation delay of each block.

For the design in Fig. 3(a), the gates to be removed to obtain a reduced design are in the shadow region. Similar to the design of a right-to-left fixed-width multiplier, several error compensation cells (AO and AND gates) should be added. The resultant block diagram of a left-to-right fixed-width multiplier is shown in Fig. 5.

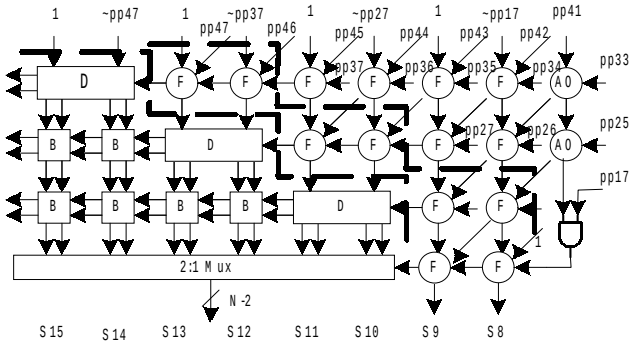
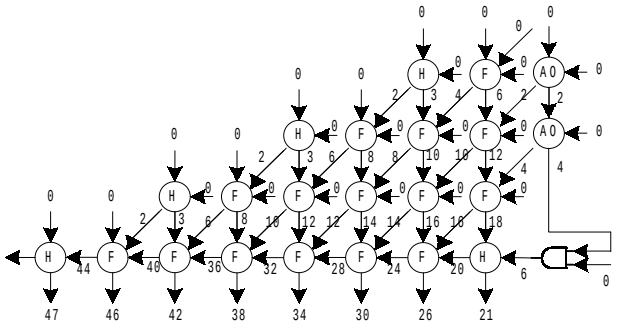
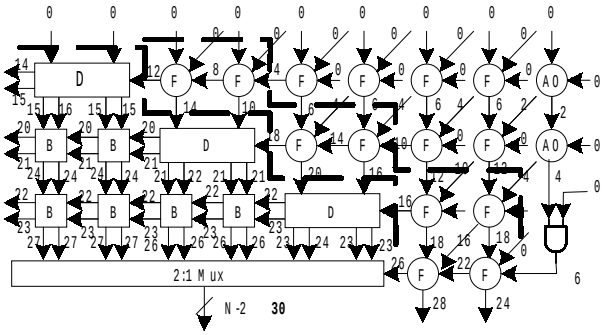


Figure 5. The proposed 8×8 left-to-right fixed-width multiplier.

To roughly compare the performance of the left-to-right and the right-to-left fixed-width multipliers, the block diagrams of these two multipliers with the propagation delay indicated are shown in Figs. 6(a) and 6(b), respectively.



(a)



(b)

Fig. 6. (a) Right-to-left and (b) left-to-right fixed-width multipliers with propagation delay indicated.

We have two observations from these two diagrams.

(a) The propagation delay of the right-to-left multiplier is 47 unit delays, but that of the left-to-right multiplier is only 30 unit delays. Thus, the 8×8 left-to-right fixed-width multiplier is about 36% faster than the right-to-left fixed-width multiplier with a ripple-carry final adder.

(b) In the left-to-right multiplier, the outputs of the on-the-fly converters arrive at the data input of the multiplexer just 1 unit delay *later* than the selection signal. Therefore, the previously described speed bottleneck from the LSP in the full-width multiplier disappears in this case. In other words, we can obtain a high-speed fixed-width multiplier just with an array structure, and we don't need any compressors in the LSP to accelerate the operating speed as proposed in [6].

3.2 Error analysis

When inspecting the designs in Fig. 2(a) and Fig. 5 carefully, we find that both designs keep the same partial products and having the same compensation terms. Therefore, both the left-to-right and the right-to-left fixed-width multipliers will have the same level of accuracy.

4. PERFORMANCE EVALUATION

In order to obtain a more realistic comparison, both kinds of 32×32 fixed-width multipliers are carried out to the physical design based on the cell-based design approach. All the designs use a 3.3-V 0.35-μm CMOS technology. The design procedures are described as follows.

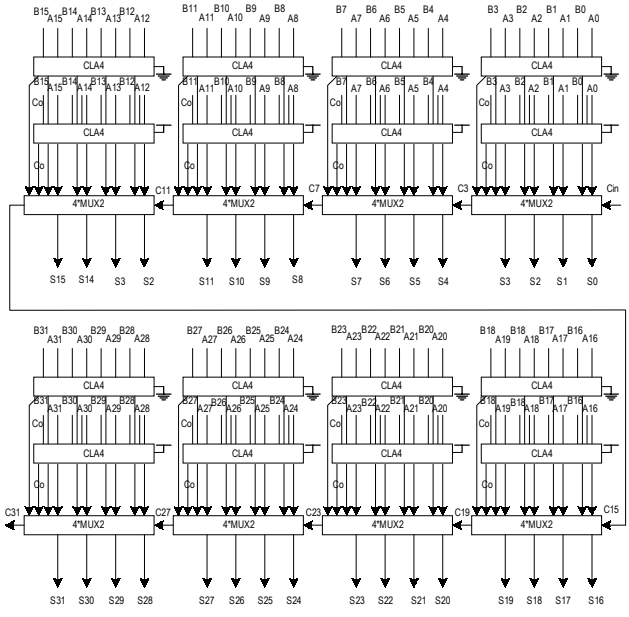


Fig. 7. The 32-b carry-select adder used in the right-to-left multiplier.

(1) Both designs are coded with the Verilog language in the structure level. The left-to-right multiplier (LR_MPY) is designed according to the expanded architecture of Fig. 5, and the right-to-left multiplier is designed according to the expanded architecture of Fig. 2(a). However, there are three versions of the right-to-left multiplier, and the difference lies in the implementation method of the final adder. The first version (RL_MPY_RCA) uses a 32-b ripple-carry adder, and the second version (RL_MPY_CSA) uses a simple 32-b carry-select adder as shown in Fig. 7. The third version (RL_MPY_DW) calls a 32-b adder cell from the *DesignWare* library.

(2) All the designs are mapped into the gate-level design with a default synthesis.

(3) All the designs are attached with two high speed ring generators [9] to generate random test patterns for X and Y inputs. The two random number generators for X and Y inputs are assigned with different seeds, and therefore the input patterns for X and Y inputs are different.

(4) The average power consumption is obtained by *NanoSim* through applying thousands of test patterns from random number generators.

(5) To obtain the speed information, we first use *NanoSim* to find the most critical pair of the test patterns, and then use *HSPICE* to extract the value of the worst propagation delay.

(6) The right-to-left multipliers are re-synthesized with tighter speed constraints to obtain several high-speed designs. These designs are analyzed to obtain the information of the operating speed and power consumption according to steps (4) and (5).

(7) All the designs are carried out all the way to the physical design, and the layout area and the speed and power from post-layout simulations are obtained for the final comparison.

Table 1 shows the obtained performance data. There are several observations from this table.

(1) With default synthesis(the 2nd row of the table), the left-to-right multiplier (LR_MPY, 3.3V) shows the highest speed with slightly smaller power consumption as compared to all the right-to-left multipliers.

(2) The operating speed of the right-to-left multipliers can be improved by applying tighter speed constraints. However, the speed improvement quickly reaches saturation but the power consumption grows at a much higher rate.

(3) By calling the *DesignWare* library, the conventional right-to-left multiplier (RL_MPY_DW) can have a better performance than just using a simple carry-select adder (RL_MPY_CSA). However, the speed improvement is still limited and the power consumption grows quickly to an unacceptable level by applying a more tight constraint.

(4) Because there is a tradeoff between speed and power consumption, we regard the design, among all the right-to-left multipliers, that has a smallest power-delay product as the best conventional design. If we take it as the reference, the proposed multiplier after default synthesis achieves 8%, 14%, and 21% improvement in delay, power, and power-delay product, respectively. See the last three columns of the table.

(5) Because the proposed multiplier owns a higher operating speed, we can trade speed for lower power consumption. When the supply voltage of the proposed design is reduced from 3.3-V to 3.0-V (LR_MPY, 3.0V), its speed is nearly equal to that of the best right-to-left multiplier. In this case, the proposed multiplier has 29% and 30% improvement in power and power-delay product, respectively.

We use Silicon Ensemble to finish the layout design, and the layout areas are reported in Table 2. As the operating speed of the right-to-left design gets higher by applying a tighter speed constraint, the proposed multiplier exhibits more area advantage. The study results show that the proposed multiplier requires a 13% smaller layout area as compared to the best conventional design as defined previously.

Table 1. Comparison of performance data.

Multiplier	Delay (τ, ns)	Power (p, mW)	p·τ (pJ)	Norm. τ	Norm. p	Norm. p·τ
LR_MPY (3.3V)	14.01	79.8	1119	0.92	0.86	0.79
LR_MPY (3.0V)	15.09	65.9	995	0.99	0.71	0.70
RL_MPY_RCA	19.83	80.95	1605	1.30	0.87	1.13
RL_MPY_CSA	17.29	89.36	1545	1.14	0.96	1.09
	16.75	98.01	1642	-	-	-
	16.09	163.78	2635	-	-	-
RL_MPY_DW	16.06	93.13	1496	-	-	-
	15.23	93.3	1422	1.0	1.0	1.0
	14.37	161.37	2319	-	-	-

Table 2 Layout areas and comparison.

Multiplier	Gate Count	Area (μm²)	Normalized Area
LR_MPY	19743	1030225	0.87
RL_MPY_RCA	19969	1041624	0.88
RL_MPY_CSA	22306	1135077	0.96
	23770	1195523	-
	28995	1412769	-
RL_MPY_DW	23343	1177225	-
	23462	1183308	1.0
	29303	1419433	-

5. CONCLUSIONS

The design and evaluation results of low-power fixed-width multipliers are presented in this paper. The proposed architecture is based on the left-to-right algorithm for partial-product reduction. For convenience of comparison, all the evaluated multipliers are coded in the structure-level Verilog, and the cell-based approach is used to obtain the physical layout. The cell-based approach is adopted to observe the performance change by applying different speed constraints. Default synthesis shows that the proposed multiplier has a highest operating speed, which can be used to trade for lower power consumption. The proposed multiplier achieves 8%, 14%, and 13% reduction in delay, power, and layout area, respectively, as compared to the best of right-to-left fixed-width multiplier. When applying the voltage scaling to balance the speed between two kinds of design, the power reduction is increased to 29%.

6. ACKNOWLEDGMENTS

This work was supported by the National Science Council under Research Grants NSC 91-2215-E-194-010 and NSC 92-2220-E-194-008.

7. REFERENCES

[1] S. S. Kidambi, F. El-Guibaly, and A. Antoniou, "Area-efficient multipliers for digital signal processing applications," *IEEE Trans. Circuits and Systems II*, vol. 43, pp. 90-95, Feb. 1996.

[2] Lan-Da Van, Shuenn-Shyang Wang, Tenqchen Shing, Wu-Shiung Feng, and Bor-Shenn Jeng, **"**Design of a lower-error fixed-width multiplier for speech processing application," in Proceedings of *the 1999 IEEE International Symposium Circuits and Systems*, vol. 3 , pp. 130–133.

[3] J.-M. Jou, S.-R. Kuang, and R.-D. Chen, "Design of low-error fixed-width multiplier for DSP applications," *IEEE Trans. Circuits and System II*, vol. 46, pp. 836-842, June 1999.

[4] M.D. Ercegovac and T. Lang, "Fast Multiplication Without Carry-Propagate Addition," *IEEE Transactions on Computer*, vol. C-39, pp. 1385-1390, November 1990.

[5] R. K. Kolagotla, H. R. Srinivas, and G. F. Burns, "VLSI Implementation of a 200-MHZ 16x16 Left-to-Right Carry-Free Multiplier in 0.35μm CMOS Technology for Next-Generation DSPs," in Proceedings of *the IEEE 1997 Custom Integrated Circuits Conference*, pp. 469-472.

[6] A. Goldovsky, B. Patel, M. Schulte, R. Kolagotla, H. Srinivas, and G. Burns, **"**Design and implementation of a 16 by 16 low-power two's complement multiplier," in Proceedings of *the 2000 IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 345-348.

[7] Zhijun Huang and M. D. Ercegovac, " High-performance left-to-right array multiplier design," in Proceedings of *the 16th IEEE Symposium on Computer Arithmetic*, pp. 4-11, 2003.

[8] Wen-Chang Yeh and Chein-Wei Jen, "High-speed Booth encoded parallel multiplier design,**"** *IEEE Transactions on Computers*, vol. 49, pp. 692-701, July 2000.

[9] G. Mrugalski, J. Rajski, and J. Tyszer, "High speed ring generators and compactors of test data," in Proceedings of *the 21st IEEE VLSI Test Symposium*, pp. 57-62, May 2003.