Area-saving technique for low-error redundant binary fixed-width multiplier implementation

Juang, Tso Bing; Wei, Chi Chung; Chang, Chip Hong

2009

https://hdl.handle.net/10356/80023

Area-Saving Technique for Low-Error Redundant Binary Fixed-Width Multiplier Implementation

Tso-Bing Juang and Chi-Chung Wei
Department of CSIE
National Pingtung Institute of Commerce
Pingtung City, Taiwan
tsobing@npic.edu.tw

Chip-Hong Chang
School of Electrical, Electronic Engineering
Nanyang Technological University,
Singapore
echhang@ntu.edu.sg

***Abstract*—A recently proposed architecture of redundant binary fixed-width multiplier was shown to outperform several normal binary fixed-width multipliers in terms of accuracy. However, its merit due to the carry-free addition property of the binary signed digit (BSD) partial products has been offset by the high area overhead of the redundant binary full adder tree. To achieve low–error fixed-width multiplication with smaller silicon area, we propose a hybrid structure which makes use of dual polarity high order column compressors and (3:2) counters to parallelly reduce the positive and negative BSD partial products. Our proposed technique has led to a fixed-width multiplier architecture with the same accuracy and up to 42% area saving for 10×10-bit multiplication over the conventional redundant binary fixed-width multiplier architecture in 0.18 m CMOS standard cell implementation under the same timing constraint.**

***Index Terms*—Fixed-width multipliers, modified Booth encoding, redundant binary signed digit representation.**

I. Introduction

Fixed-width multipliers [1-13] are widely used in application specific data paths in multimedia and wireless communication applications where some degree of saturation error within the dynamic range of interest is tolerable depending on the level of perceptual quality and signal-to-noise degradation it induced. For maximal area reduction, the least significant half of the partial products are truncated to produce a final product with reduced precision. To achieve a lower average error and narrower error spread for all legitimate operands within the dynamic range, error-compensation circuitry with area overhead much lower than the truncated part is introduced.

In [1-3], a constant error compensation technique is used to compensate for the truncation errors without considering the bit-width of the inputs. Although these methods are attractive in terms of their hardware simplicity, they introduce large average and mean square errors compared to those techniques that use adaptive error compensation [4-13]. Adaptive error compensation techniques estimate the error compensation values (ECV) from a few most significantly weighted bits of the truncated part of the partial products. Fixed-width multipliers can be categorized in the similar way as their full-width counterparts. The partial products can either be generated directly by wire-ANDing the corresponding bits of the multiplier with the multiplicand [1-5, 8, 11, 13], or via modified Booth encoders (MBE) [6-7, 9-10] to reduce the total number of partial products. The methods for summing the partial products can also be classified into array [1-6, 8, 9, 11] and tree structures [7, 10, 12]. Finally, fixed-width multipliers can also be distinguished by their partial product representation. Two commonly used representations are the two's complement binary [1-9, 11-13] and the binary signed digit (BSD) representations [10].

In [10], we proposed a redundant binary fixed-width multiplier by summing the truncated MBE partial products in BSD representation in a tree-structured architecture. Very low error fixed-width multiplication can be achieved with a very simple error compensation circuit. However, the merits of low error and carry-free addition are offset by the overhead of the summation network of BSD partial products. It incurs higher area cost comparing with the array structured summing network of two's complement partial products.

This paper proposes an area saving technique to overcome the above problem. Instead of using the conventional redundant binary adders, hybrid original and complementary high order column compressors are designed for summing the decomposed positive and negative binary partial products. The resulting fixed-width multiplier maintains the same advantage of low truncation error and structural regularity. A conspicuous saving of around a quarter of the area over the previously proposed fixed-width multiplier [10] is achieved when the design is mapped using TSMC 0.18μm standard cell library under the same delay constraint.

This paper is organized as follows. In Section II, the overhead of the redundant binary fixed-width multiplier architecture [10] is analyzed. Our proposed area-saving technique and the hybrid redundant binary partial product summing tree are presented in Section III. The synthesis results of the previous and proposed BSD fixed-width multipliers are compared in Section IV. Section V concludes the paper.

II. Overhead of Previous Redundant Binary Fixed-Width Multiplier

The statistical error of a fixed-width multiplier based on BSD representation is analyzed in [10]. This section provides the preliminaries of redundant binary fixed-width multiplier architecture and its overheads.

The signed digit partial products for an *n*-bit fixed-width multiplication, $P = A \times B$ can be obtained from its two's complement binary multiplicand $A=-a_{n-1}2^{n-1}+\sum_{i=0}^{n-2}a_i2^i$ and multiplier $B=-b_{n-1}2^{n-1}+\sum_{i=0}^{n-2}b_i2^i$, where $a_i,b_i\in\{0,1\}$, using the MBE architecture proposed in [14]. $\lceil n/2\rceil$ rows of signed digit partial products, $PP[i]=\sum_{j=n-2i}^{n}p_{i,j}2^{2i+j}, i=0,..,\lceil n/2\rceil-1$ are generated. Each partial product corresponds to a shifted version of an MBE encoded multiplicand, $(b_{2i}+b_{2i-1}-2b_{2i+1})\times A$ with $b_{-1}=0$ and each partial product digit, $p_{i,j}\in\{\pm1,0\}$ can be encoded in redundant binary form as $p_{i,j}=p_{i,j}^{+}-p_{i,j}^{-}$, where $p_{i,j}^{+},p_{i,j}^{-}\in\{0,1\}$. For ease of exposition, we call $p_{i,j}^{+}$ the positive bit and $p_{i,j}^{-}$ the negative bit of the signed digit partial product, $p_{i,j}$.

Fig. 1 shows the redundant binary implementation of an 8×8-bit fixed-width multiplier designed with a very simple compensation circuit, *C*. This circuit computes the carry generated out of the most significant digit position of the truncated part of two partial products [10]. The block MBE represents the modified Booth encoder [14] used to produce the four BSD partial products, *PP*[0] to *PP*[3]. They are summed in two stages of redundant binary full adders (RFAs) [15]. The final truncated product is converted to two's complement form by a BSD-to-binary converter.

This work is inspired by the low statistical error of fixed-width multiplier designed based on the BSD representation. The analyses of [10] shows that the BSD partial products generated by [14] has lower average bias due to the bipartite weights of the truncated bits. The positive ($+2^i$), neutral (0) and negative (-2^i) weights of BSD make the estimation of ECV simpler and more accurate than using the partial product bits generated from the conventional two's complement representation.

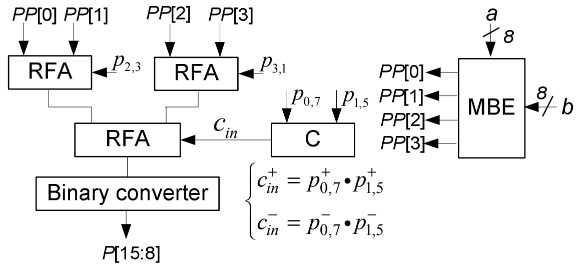

Fig. 1: Redundant binary fixed-width multiplier design [10] for *n* = 8 where only two AND gates are required in the compensation circuit *C*.

The penalty of achieving the lower mean error and lower error variance by the BSD partial products is the higher area cost. Table I shows the logic area in μm^2 obtained by synthesizing the structural VHDL code of various fixed-width multipliers by Synopsys Design Compiler using the TSMC 0.18 μm CMOS standard cell library with a consistent delay constraint of 4.2 ns. In Table I, the numbers in the parenthesis represent the area overheads over the least area fixed-width multiplier [3], which was designed based on a constant ECV. Notably, the area overhead of the fixed-width multiplier of [10] is more than 60% that of [3] although its absolute average error and absolute error variance are only respectively 0.55 and 0.18 that of [3].

The area overhead of [10] arises from two constituent components, namely the circuitries for the generation of the BSD partial products and their summation. The BSD partial products are summed using an array of one-digit RFAs designed according to the circuit proposed in [15]. Since the redundant binary fixed-width multiplier has better error performance than those that are based on the conventional binary representation, it makes sense to keep the original MBE hardware for the generation of BSD partial products while eradicating the use of the area consuming RFAs of [15] for the summation of these partial products. This area-saving technique will be discussed in the next section.

TABLE I

SYNTHESIS RESULTS OF DIFFERENT 8×8-BIT FIXED-WIDTH MULTIPLIERS

Methods	Area (μm^2)
[3]	5129
[4]	7115 (+38.72%)
[5]	5967 (+16.34%)
[10]	8432 (+64.4%)

III. PROPOSED AREA SAVING TECHNIQUE FOR LOW-ERROR FIXED-WIDTH MULTIPLIER DESIGN

The BSD partial products are generated using MBE circuit of [14], where each partial product digit is encoded using two binary bits, $p_{i,j}^{+}$ and $p_{i,j}^{-}$ for i = $0,1,\cdots,\lceil n/2\rceil-1$ and j = $0,1,\cdots,n-1$. This MBE cell does not account for the sign bit of the multiplicand. For example, if the multiplicand is 10110011 = −77, the BSD partial product should be $\overline{1}01100110$ when the encoded multiplier digit is +2 and $010\overline{1}\overline{1}00\overline{1}\overline{1}$ when the encoded multiplier digit is −1.

Instead of summing the generated BSD partial products using the RFAs proposed in [15], the positive and negative partial product bits are separately added in two parallel binary adder trees. The BSD partial product accumulation can be performed by trees of $PP^{+}[i]=\sum_{j=8-2i}^{8}p^{+}_{i,j}2^{2i+j}$ and $PP^{-}[i]=\sum_{j=8-2i}^{8}p^{-}_{i,j}2^{2i+j}$ for $i=0,1,2,3$. The final product of the 8×8-bit fixed-width multiplier is obtained by subtracting PP^{-} from PP^{+} with a two's complement carry propagate adder.

Intuitively, the summation of PP^{+} and PP^{-} can be implemented using carry-saved adders followed by a low complexity ripple carry adder or a high-speed carry lookahead adder. This solution will require more carry propagate adders than the binary-based fixed-width multiplier with only one carry-saved adder tree, although the average error is lower. In order to further minimize the area without compromising the average error of the fixed-width multiplier, we propose to use a single hybrid adder tree with bipolar column compressors and counters of different reduction ratios for the addition of the BSD partial products.

Our carry-saved addition tree consists of three stages. The BSD partial product accumulation of an $n\times n$-bit redundant binary fixed-width multiplication is illustrated in Fig. 2. For ease of exposition, we assume n is even. The partial product matrix is divided into two equal halves, PP^+ and PP^-. Each half matrix contains $n/2$ rows and n columns. The elements in the jth column have a weight of 2^{n+j}. Let $h(j)$ be the number of partial product bits in the jth column. The ECV in column $j = -1$ of each half matrix consists of the partial product terms, $p_{n/2-i,2i-1}$ for $i = 1, 2, \ldots\cdots, n/4$ and a carry-in, c_{in} [10]. From Fig. 2, it is easy to see that:

$$h(-1) = \frac{n}{4}+1;\ h(0) = \frac{n}{2};$$
$$h(j) = h(j-1) = \frac{n-j}{2}\ \forall j = 2,4,\cdots,n-2 \tag{1}$$

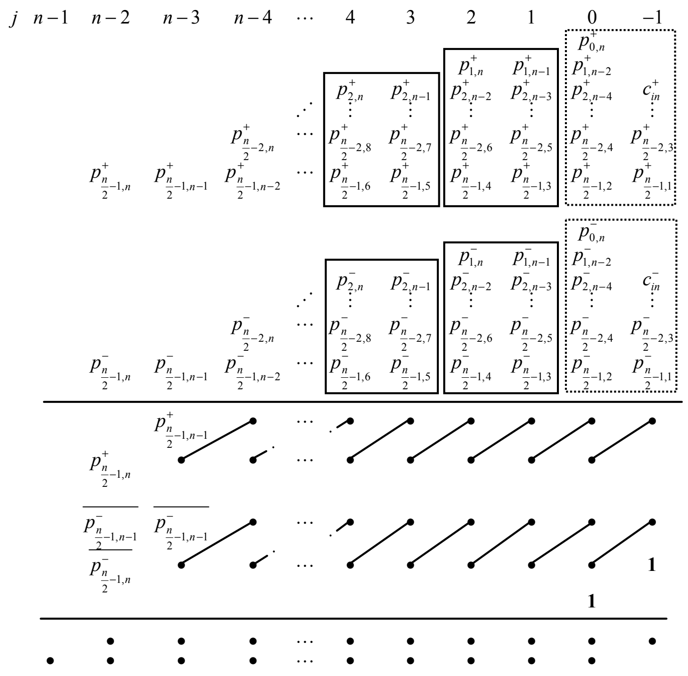
Fig. 2: Three-stage summation of partial product matrix of $n\times n$-bit redundant binary fixed-width multiplier

In Stage 1, each PP^+ and PP^- matrix is independently and concurrently reduced to two binary partial product rows using a combination of (3:2) counters, (4:2) compressors and (5:2) compressors for $n \le 10$. For $n > 10$, $[h(j){:}2]$ adders [16] are required for some columns. A $[h{:}2]$ adder reduces h k-bit operands to two k-bit operands with the output carries independent of its input carries. The optimal k for different h to minimize the number of lateral carries has been derived in [16] based on a square dot matrix. Due to the unique staircase-like partial product matrix, the maximum value of the output carries of the $[h{:}2]$ adder can be increased and h can be extended to 8 for $k = 2$ following the same derivation. Thus for $10 < n \le 18$, $[h(j){:}2]$ adders (the solid rectangular blocks in Fig. 2) are used in the columns, $j = 1, 3, \ldots, n-7$ to reduce two adjacent columns (column numbers j and $j+1$) in each half matrix to produce a sum bit in the jth column and a carry bit in the $(j+1)$th column in the next stage (the sum and carry bits are connected by a slash). For the columns $j = -1$ and 0, a $[\frac{n}{2}-1{:}2]$ adder (the dotted rectangular block) can be used by transferring one bit from the column $j = 0$ to two equivalent bits in the column $j = -1$. The remaining columns can be reduced using either $(h(j){:}2)$ counters or compressors. To unify all binary vectors in the next stage in two's complement form, the sum and carry outputs generated from the negative partial product matrix are complemented and '1's are added to the two least significant bit columns, $j = -1$ and 0. In addition, the two most significant negative partial product bits (i.e., $p^-_{\frac{n}{2}-1,n-1}$ and $p^-_{\frac{n}{2}-1,n}$) that are directly brought forward to the next stage need to be complemented and $p^-_{\frac{n}{2}-1,n-1}$ needs to be signed extended to the leading bit position.

In Stage 2, since there is no distinction between the sum and carry bits generated from the positive and negative partial product bits, they can be further compressed using (4:2) compressors and (3:2) counters as in the conventional two's complement tree-structured multiplier. In Stage 3, the final carry and sum vectors are added using an n-bit carry propagate adder to produce the truncated product by discarding the last bit.

A numerical example is shown in Fig. 3 to illustrate our proposed method. The multiplicand and multiplier in two's complement representation are a = 10110110 and b = 10100011, respectively. The multiplier, b is Booth encoded to $1\bar{2}1\bar{1}$. The four full-width BSD partial products are respectively, $PP[0] = 010\bar{1}\bar{1}0\bar{1}\bar{1}0$, $PP[1] = 0\bar{1}011011000$, $PP[2] = 10\bar{1}\bar{1}0\bar{1}\bar{1}000000$ and $PP[3] = 010\bar{1}\bar{1}0\bar{1}\bar{1}0000000$. Therefore, $PP^+[0] = PP^-[0] = 0$, $PP^+[1] = 000$, $PP^-[1] = 010$, $PP^+[2] = 100000$, $PP^-[2] = 001101$, $PP^+[3] = 01000000$ and $PP^-[3] = 00011011$. From Fig. 1, $c^+_{in} = 1$ and $c^-_{in} = 0$.

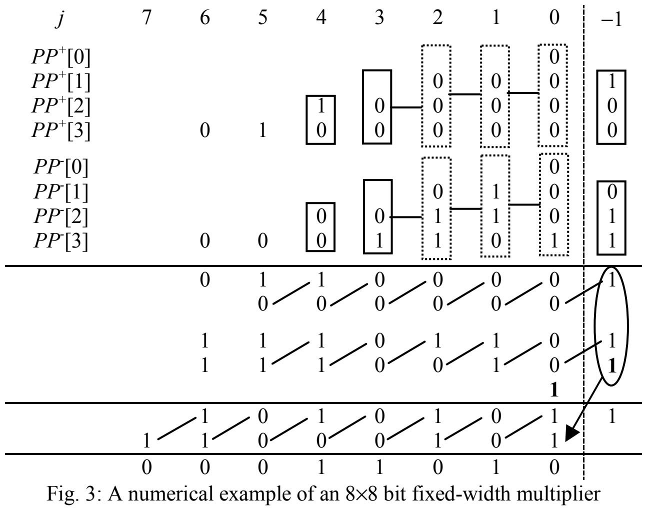
Fig. 3: A numerical example of an 8×8 bit fixed-width multiplier

Stage 1 uses four FAs, two HAs and six simplified (4:2) compressors. Each FA and HA is indicated by a solid rectangle and each (4:2) compressor is indicated by a dotted rectangle with a lateral carry-out connection to its adjacent unit. A standard (4:2) compressor is actually a (5:3) counter with 5 input bits and three output bits [17]. In this case, the unused input is used to simplify the circuit of the (4:2) compressor. Besides, for the summation of the negative partial product bits, the output complement is absorbed into the logic of the (4:2) compressor, FA and HA cells. The positive and negative simplified (4:2) compressors are shown in Fig. 4. In Stage 2, six standard (4:2) compressors are used in the columns $j = 0$ to 5 and one simplified (4:2) compressor is used in the column, $j = 6$. Only the carry generator circuit of an FA (indicated by a circle) is needed for the column $j = -1$ as the sum bit will be discarded. An 8-bit carry lookahead adder is used in the third stage to produce the final truncated product of 00011010×2^8, which is different from the exact product by only 3.28%. The delay for the low-error redundant binary fixed-width multiplier is approximately $T_{\mathrm{MBE}} + 2T_{4:2\ \mathrm{compressor}} + T_{\mathrm{CLA}}$, where T_{MBE} is the delay of the MBE circuit, which is equivalent to seven two-input unit gate delay according to our simulations; $T_{4:2\ \mathrm{compressor}}$ is the delay of one 4:2 compressor, which can be approximated to twice the delay of a FA and T_{CLA} is the delay of an 8-bit carry-lookahead adder.

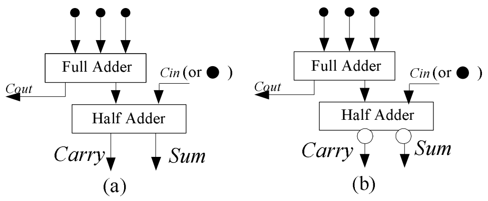

Fig. 4: Simplified (4:2) compressors for the reduction of (a) positive and (b) negative partial product bits. A dot represents an input bit and a circle denotes an output complement.

IV. EXPERIMATAL RESULTS

The proposed low-error redundant binary $n\times n$-bit fixed-width multipliers are implemented for $n = 8$, 10 and 12 and synthesized by Synopsys Design Compiler using the TSMC 0.18 μm CMOS standard cell library. To demonstrate the effectiveness of the proposed area-saving technique, the designs are compared with the most recent low cost carry-free fixed-width multipliers [10] of the same statistical error performance in Table II. Both designs are area optimized by the same synthesis tool with the same timing constraints of 3.6 ns to 4.0 ns.

TABLE II

COMPARISON OF AREA-OPTIMIZED SYNTHESIS RESULTS OF LOW-ERROR FIXED-WIDTH MULTIPLIERS FOR n = 8, 10 and 12.

Multipliers	Delay(ns)	Area (μm^2)		%Area Saving
		[10]	Proposed	
8×8	4.0	8372	7058	15.7%
	3.8	8395	7208	14.1%
	3.6	8478	7592	10.4%
10×10	4.0	17576	10897	38%
	3.8	19575	13175	33%
	3.6	23617	13748	42%
12×12	4.0	27765	19758	29%
	3.8	33985	21119	38%
	3.6	N/A	25270	N/A

Both methods are error compensated based on the BSD partial products generated by the baseline MBE of [14]. Using the hybrid and bipolar 4:2 compressors and 3:2 counters, our method achieves a significant 15.7% reduction of area for an 8×8 fixed-width multiplication at 250 MHz. The area savings are more prominent for higher operand length and more stringent timing constraint. It is also noted that the design of [10] can not be synthesized as the timing constraint becomes too tight. These cases are marked as 'N/A' in Table II.

V. CONCLUSIONS

This paper proposes an area-saving technique to reduce its high area cost in the summation of partial products. The proposed carry-saved addition of the BSD partial products consists of three stages and involves a mixture of high-order column compressors and (3:2) counters. While maintianing the low average error and error variance of the original fixed-width multiplier, around 40% of precious silicon premium can be saved from a 10×10-bit fixed-width multiplication.

ACKNOWLEDGMENT

This work was partially sponsored by National Science Council in Taiwan under the contract number NSC 98-2221-E-251-005.

REFERENCES

[1] Y. C. Lim, "Single precision multiplier with reduced circuit complexity for signal processing applications," *IEEE Trans. on Computers*, Vol. 41, No. 10, pp. 1333-1336, Oct. 1992.

[2] M. J. Schutle and E. E. Swartzlander Jr., "Truncated multiplication with correction constant," *VLSI Signal Processing* VI, pp. 388-396, 1993.

[3] S. S. Kidambi, F. El-Guibaly, and A. Antoniou, "Area-efficient multipliers for digital signal processing applications," *IEEE Trans. on Circuits and Syst. II*, vol. 43, no. 2, pp. 90-95, Feb. 1996.

[4] J. M. Jou, S. R. Kuang and R. D. Chen. "Design of low-error fixed-width multipliers for DSP applications," *IEEE Trans. on Circuits and Syst. II*, vol. 46, no. 6, pp. 836-842, June 1999.

[5] L. D. Van, S. S. Wang and W. S. Feng. "Design of the lower error fixed-width multiplier and its application," *IEEE Trans. on Circuits and Systems II*, vol. 47, no. 10, pp. 1112-1118, Oct. 2000.

[6] S. J. Jou and H. H. Wang, "Fixed-width multiplier for DSP application," in *Proc. 2000 IEEE Int. Conf. on Computer Design*, pp. 318-322, Sept. 2000.

[7] K. J. Cho, K. C. Lee, J. G. Chung and K. K. Parhi, "Design of low-error fixed-width modified Booth multiplier," *IEEE Trans. on VLSI Syst.*, vol. 12, no. 5, pp. 522-531, May 2004.

[8] J. S. Wang, C. N. Kuo and T. H. Yang, "Low-power fixed-width array multipliers," in *Proc. 2004 Int. Symp. on Low Power and Electronics and Design*, pp. 307-312, Aug. 2004.

[9] K. J. Cho, S. M. Lee. S. H. Park and J. G. Chung, "Error bound reduction for fixed-width modified Booth multiplier," in *Proc. 38th Asilomar Conf. on* Signals, Syst. and Computers, vol. 1, pp. 508-512, Nov. 2004.

[10] T. B. Juang and S. F. Hsiao, "Low-error carry-free fixed-width multipliers with low-cost compensation circuits," *IEEE Trans. on Circuits and Syst. II*, vol. 52, no. 6, pp. 299-303, June 2005.

[11] L. D. Van and C. C. Yang, "Generalized low-error area-efficient fixed-width multipliers," *IEEE Trans. on Circuits and Syst. I*, vol. 52, no. 8, pp.1608-1619, Aug. 2005.

[12] A. G. M. Strollo, N. Petra and D. de Caro, "Dual-tree error compensation for high performance fixed-width multipliers," *IEEE Trans. on Circuits and Syst. II*, vol. 52, no. 8, pp. 501-507, Aug. 2005.

[13] Y. C. Liao, H. C. Chang and C. W. Liu, "Carry estimation for two's complement fixed-width multipliers," in *Proc. 2006 IEEE Workshop on Signal Processing Syst. Design and Implementation*, pp. 345-350, Oct. 2006.

[14] N. Besli and R. G. Deshmukh, "A novel redundant binary signed-digit (RBSD) Booth's encoding," in *Proc. 2002 IEEE Southeast Conf.*, South Carolina, USA, pp. 426-431, Apr. 2002.

[15] H. Makino, Y. Nakase, H. Suzuki, H. Morokina, H. Shinohara and K. Mashiko, "An 8.8-ns 54×54-bit multiplier with high speed redundant binary architecture," *IEEE J. of Solid-State Circuits*, vol. 31, no. 6, June 1996.

[16] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann Publishers, New York, 2004, pp. 140-148.

[17] C. H. Chang, J. Gu and M. Zhang, "Ultra low voltage, low power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," *IEEE Trans. on Circuits and Syst.-I*, vol. 51, no. 10, pp. 1985-1997, Oct. 2004.