High-Speed Booth Encoded Parallel Multiplier Design

Wen-Chang Yeh and Chein-Wei Jen, *Member*, *IEEE*

Abstract—This paper presents a design methodology for high-speed Booth encoded parallel multiplier. For partial product generation, we propose a new modified Booth encoding (MBE) scheme to improve the performance of traditional MBE schemes. For final addition, a new algorithm is developed to construct multiple-level conditional-sum adder (MLCSMA). The proposed algorithm can optimize final adder according to the given cell properties and input delay profile. Compared with a binary tree-based conditional-sum adder, the speed performance improvement is up to 25 percent. On average, the design developed herein reduces the total delay by 8 percent for parallel multiplier. The whole design has been verified by gate level simulation.

Index Terms—Final adder, Booth encoding, multiple-level conditional-sum adder, and parallel multiplier.

1 Introduction

In various computing and signal processing applications, parallel multiplier has been a basic building block for many algorithms. Many high performance algorithms and architectures have been proposed to accelerate multiplication. Multiplication can be divided into three steps: generating partial products, summing up all partial products until only two rows remain, and adding the remaining two rows of partial products by using a carry propagation adder.

In the first step, two methods are commonly used to generate partial products. The first method generates partial product directly by using a 2-input AND gate. The second one uses radix-4 *modified Booth encoding* (MBE) to generate partial products [1], [2]. Radix-4 MBE has been widely used in parallel multipliers to reduce the number of partial products by a factor of two. In [3], [4], the speed performance of using radix-4 MBE was denied. However, it is found herein that these results depend on the implementation of MBE scheme.

After generating partial products, a *partial product reduction tree* (PPRT) is used to sum up all the partial products efficiently. The Wallace tree and Carry-save tree were developed to solve this problem [5], [6]. Both approaches employ 3:2 counter, i.e., full adder, as their basic element. Generally, a counter compresses (n-1) rows of partial products into $log_2(n)$ rows of partial products. However, the delay of an $(n-1):log_2(n)$ counter is still proportional to $(log_2(n)-1)$ times of a full adder (FA) as the inputs are assumed to arrive simultaneously. Therefore, using larger counters to build PPRT is not beneficial. The introduction of 4:2 compressor was a departure from the counter-based scheme [7], [8], [9]. As the delay paths are well balanced, the latency for a 4:2 compressor is only three XOR delays, rather than two full adder delays. Note that the difference between the compressor and the traditional balanced delay tree [10], [11] is that the compressor considers the fast path and the slow path of a full adder. To further speed up, a search algorithm, Three-Dimensional-reduction-Method (TDM) [12], [13], was proposed. The TDM algorithm finds optimal PPRT by carefully modeling the delay paths of a counter and constructing n:2 column compressor according to inputs arrival time. Owing to the effectiveness of the column compressor, the PPRT constructed by using TDM algorithm outperforms the conventional designs. However, few studies have been done on using TDM with MBE. This paper examines the performance of parallel multiplier constructed with TDM and MBE. According to our results, such a design can be faster and occupy a smaller area than a non-Booth design.

To generate the product in 2's complement format, a fast carry-propagation adder is required to add the final two rows of partial products from the PPRT. The problem of designing a final adder is that the input signals *do not* arrive simultaneously, unlike the ordinary carry-propagation adder design that assumes all the inputs arrive simultaneously. Several techniques have been developed to eliminate or reduce the final adder delay [14], [15], [16]. The Left-to-Right-Carry-Free algorithm proposed in [14] requires n-level conversions to generate n-bit MSB products. It was improved in [15] by reducing the levels required. However, this approach still cannot fully exploit the unequal delay profile because it applies to the MSB-part only. In [16], a hybrid adder structure, which consists of ripple-carry adder, carry-skip adder, and conditional-sum adder blocks, was proposed. However, their empirical methodology is not general enough and requires many trials to determine the final adder partition boundary for different sizes of multiplier, thus increasing design effort.

In this paper, we will propose a design methodology for high-speed Booth encoded parallel multiplier. The rest of this paper is organized as follows: Section 3 presents a new modified-Booth encoding scheme for efficiently generating partial products. When radix-4 MBE is used with TDM

• *The authors are with the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, ROC. E-mail: wcyeh@twins.ee.nctu.edu.tw.*

*Manuscript received 1 Sept. 1999; revised 1 Feb. 2000; accepted 10 Mar. 2000.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 111790.*

TABLE 1
Normalized Gate Delay

Gate Name	Normalized delay	Description
XOR_2	1.0	2-input Exclusive-OR
XNOR_2	1.0	2-input Exclusive-NOR
AND_2	0.7	2-input AND gate
NAND_2	0.5	2-input NAND gate
MUX_2	1.0	2-input Multiplexer
MUXI_2	1.0	2-input inverted Multiplexer
NAND_3	0.5	3-input NAND gate
NOR_2	0.5	2-input NOR gate
NOR_3	0.75	3-input NOR gate
OR_2	0.7	2-input OR gate

algorithm, we found that the traditional MBE schemes are either slow or area inefficient. We solve this problem by using a new MBE recoder and an improved partial product array. Compared with the fastest MBE design, the scheme proposed performs equally fast, but requires smaller area. The proposed modified Booth recoder has a latency of two XOR delays and the Booth decoder consists of 18 transistors. The derived new MBE partial product array improves the speed performance at the LSB part. By combining both approaches, it was found that radix-4 MBE design outperforms conventional non-Booth designs.

Section 4 proposes an algorithm which constructs final adder efficiently. The proposed algorithm can build multiple-level conditional-sum adder for any word-length in linear time. This algorithm is based on the observation that the delay profile of the inputs for final adder changes gradually. Previous methods divide the final adder into two parts [15] or three parts [16]. Instead of such a rough partition, we optimize the final adder incrementally from LSB to MSB. Conditional-carry adder [25] and conditional-sum adder are used as our basic building elements and the final adder is optimized bit by bit. Compared with tree-based conditional-sum adder, the developed final adder offers speed improvement up to 25 percent. On the average, the speed improvement is 8 percent for a parallel multiplier. To verify our design, Synopsys tools are employed which consider buffer delay, capacitance loading, and driving capability. Section 5 provides gate-level simulation of our design. The simulation results closely match the estimation from our algorithm.

2 Normalized Gate Delay Model

Throughout this paper, we will use normalized gate delay model to analyze circuit performance. As in [12], [13], the delay of a 2-input XOR gate is considered as one unit delay to simplify the analysis. The delays of the other CMOS logic gates in [17] are normalized with respect to the unit delay. Table 1 summarizes the information for the logic gates used in this work. By using the cells from [17], three different full adders: Type_I, Type_II, and Type_III are available. Table 2 summarizes their characteristics. For the first row, the symbols "A," "B," and "C" denote the inputs of a full adder. The symbol "S" denotes the sum signal and "CO" denotes the carry out signal. The propagation delays from inputs to outputs are summarized as shown in Table 2.

3 New Radix-4 Modified Booth Encoding Scheme

3.1 New MBE Recoder

Three different MBE recoders, based on CMOS logic, are evaluated here for their distinct unique features. Note that these recoders can be realized by using pass-transistor logic or other logic styles.

The first one, denoted as MBE_I in Table 3, was presented in [18]. It uses only 10 transistors, making it the smallest one of all the static logic designs. However, the latency of this design is approximately three unit delays and the recoder may suffer similar power efficiency problem as the standard Booth recoder[20].

Fig. 1 shows the decoders of the next two MBE recoders. The decoder illustrated in Fig. 1a was used in [19]. This scheme is denoted as MBE_II in Table 3. Since the "X2" signal arrives at t = 1.5 unit delays, the total latency of this design is 3.5 unit delays. If the 2-input XNOR gate can be implemented using two transmission gates and two inverters, and if the two 2-input AND gates and the 2-input NOR gate are combined into one complex gate, then the transistor count of this decoder is 16. Fig. 1b shows the third design from [20] and this scheme is denoted as MBE_III in Table 3. Because "X1," "X2P," and "ZP" arrive at t = 1.0 unit delay, the total latency is reduced to two unit delays. However, the area of the decoder increases to 24 transistors for full CMOS implementation. As in the discussion of MBE_II, it is possible to implement the decoder of MBE_III scheme with 20 transistors.

The new MBE recoder was designed according to the above analysis. Table 4 presents the truth table of the new encoding scheme. The "Z" signal makes the output zero to compensate the incorrect "X2_b" and "Neg" signals. Fig. 2

TABLE 2
Characteristics of the Full Adders (Unit: Unit Delay)

Full Adder	$A \to S$	$B \to S$	$C \to S$	$A \to CO$	$B \to CO$	$C \to CO$	Description
Type_I	1.5	1.5	1.5	1.0	1.0	1.0	Library cell, smaller than Type_II.
Type_II	1.0	1.5	1.5	1.0	1.0	1.0	Composed of one 3-input XOR gate and one carry generator.
Type_III	1.0	2.0	2.0	1.0	1.0	1.0	The same configuration as in [13]. Composed of two 2-input XOR gates and one carry generator.

TABLE 3
Comparison of Partial Product Generation

	Unit delay	Decoder (trans. count)	Comments
MBE_I	3.0	10	Slow, small area.
MBE_II	3.5	16	Slow, moderate area.
Our Design	2.0	18	Fast, moderate area.
MBE_III	2.0	20	Fast, large area.

presents the new MBE scheme. In Fig. 2, the encoder and the decoder receive 3-bit x inputs and n-bit y inputs, respectively. Fig. 3 presents the circuit diagram of the encoder and decoder. The encoder generates "X1_b," "X2_b," and "Z" signals by encoding the three x-signals. The y_{LSB} signal is the LSB of the y signal and is combined with x-signals to determine the Row_LSB and the Neg_cin signals. Similarly, y_{MSB} is combined with x-signals to determine the sign extension signals. For simplicity, we omitted the Row_LSB, Neg_cin, and sign extension circuitry in Fig. 3. Fig. 4 shows an overview of the partial product array. From Fig. 2 and Fig. 3, it is easy to verify that the delay for any path is equivalent to two unit delays only. The new recoder has the same speed performance as MBE_III, but uses only 18 transistors for the decoder. Furthermore, if we can provide $\overline{Neg}$ and $\overline{y_j}$, then the XOR gate can be replaced with only two transmission gates. Thus, the decoder would use only 14 transistors. However, since the cell library used here does not provide transmission gates, we still use the decoder shown in Fig. 3a.

3.2 Area Analysis

The transistor count of the decoder is very important for the overall area of a parallel multiplier. To analyze the area cost, we can compare the transistor count for each scheme. At first, a non-Booth $n \times n$ multiplier will require approximately n^2 FAs and n^2 2-input AND gates (for partial product generation), while a radix-4 MBE will require

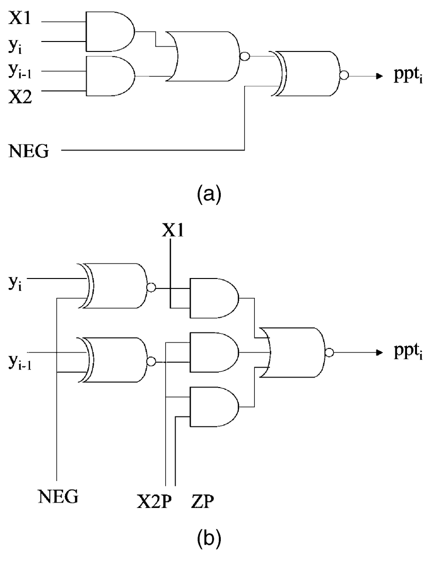


Fig. 1. The decoder from (a) MBE_II; (b) MBE_III.

TABLE 4
Truth Table of New MBE Scheme

x_{2i+1}	x_{2i}	x_{2i-1}	value	X1_b	X2_b	Neg	Z
0	0	0	0	1	0	0	1
0	0	1	1	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	2	1	0	0	0
1	0	0	-2	1	0	1	0
1	0	1	-1	0	1	1	0
1	1	0	-1	0	1	1	1
1	1	1	0	1	0	1	1

approximately $\frac{n^2}{2}$ FAs and $\frac{n^2}{2}$ decoders. An FA with fast carry output property can be realized by 24 transistors [21]. Because the area complexity of encoder and sign-extension circuitry is only of order n, they are neglected here for simplicity. Hence, the PPRT area ratio between non-Booth and radix-4 MBE can be written as:

$$Area\ ratio = \frac{A_{FA} \times n^2 + A_{AND} \times n^2}{A_{FA} \times \frac{n^2}{2} + A_{DECODER} \times \frac{n^2}{2}} = \frac{30}{12 + \frac{A_{DECODER}}{2}}. \quad (1)$$

The area ratio is calculated by using the transistor count for each decoder listed in Table 3. The area ratio ranges from 1.36 to 1.76. MBE_I is the most area efficient design because the decoder is shared for successive two bits and the decoder uses transmission-gate logic. Compared with non-Booth design, MBE scheme saves area significantly in terms of transistor count. The new scheme presented here is the fastest design with good area performance. If other logic families are available, the area performance will be better, as done in [18]. Note that (1) should not be applied without including the O(n) terms for small word length multipliers. Also, it is possible to generate partial product using 2-input NAND gate instead of using 2-input AND gate for the non-Booth design.

3.3 Modified Partial Product Array

Fig. 4a illustrates the conventional MBE partial product array for an 8×8 multiplier. We use the sign extension circuitry developed in [22] and [23]. The conventional MBE partial product array has two drawbacks: 1) an additional partial product term at the (n-2)th bit position; 2) poor performance at the LSB-part compared with the non-Booth design when using the TDM algorithm. To remedy the two drawbacks, the LSB part of the partial product array is modified. Referring to Fig. 4a, the Row_LSB (gray circle) and the Neg_cin terms are combined and further simplified using Boolean minimization. The new equations for the Row_LSB and Neg_cin can be written as (2) and (3), respectively.

$$Row\_LSB_i = y_{LSB} \cdot (x_{2i-1} \oplus x_{2i}) \quad (2)$$

$$Neg\_cin_i = x_{2i+1} \cdot \overline{\overline{x_{2i-1} + x_{2i}} \cdot \overline{x_{2i-1} + y_{LSB}} \cdot \overline{x_{2i} + y_{LSB}}}. \quad (3)$$

Referring to Table 1, we can see that the delay for either equation is within two unit delays. Note that x_{2i-1} is zero

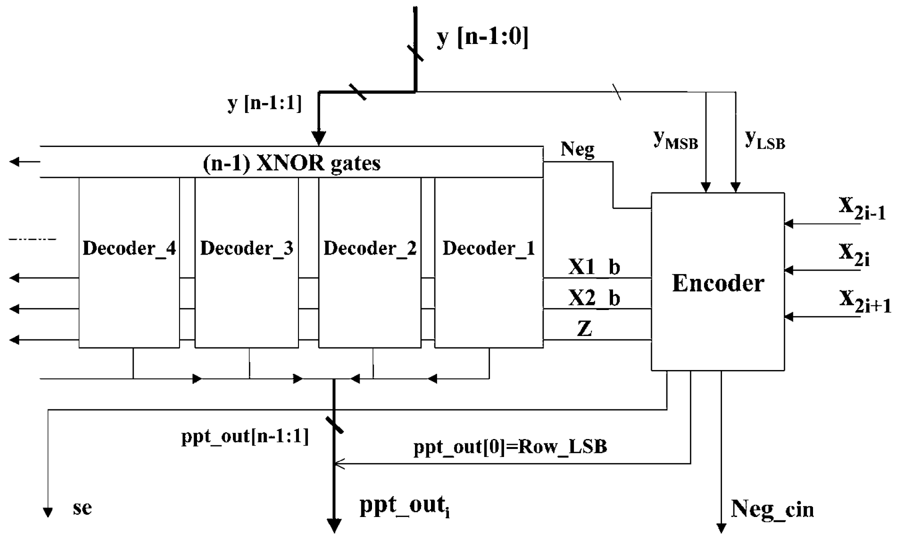


Fig. 2. The structure of the new MBE scheme.

for the first row. Fig. 4b shows the new partial product array. The additional partial product term is now moved to (n-1)th bit position, and the LSB-part array becomes more regular. Note that this optimization does not incur any overheads.

3.4 Effects of Using the New MBE Scheme

To examine the effects of applying the new MBE scheme, the delay profiles for each design are compared. The non-Booth delay profile used here is from [13]. The MBE delay profiles are generated by applying the TDM algorithm. For a fair comparison, only Type_III FA is used in the TDM program. However, similar results can be obtained for the other types of FA. Note that the partial product generation delay is included in the figures. The latency of partial product generation is 0.5 unit delays for non-Booth design and two unit delays for MBE design. The comparison between the conventional array and our modified array is shown in Fig. 5. For 32×32 and 48×48 parallel multipliers, most of the outputs arrive at the same time. However, at the bit positions: 0, 2, 5, 14, 25, 42, 43, 54, and 55, the modified array performs better than the conventional array. In Fig. 6, the new MBE scheme is 0.5 unit delays faster than the non-Booth scheme at almost every bit position. At bit positions from 0 to 5, our scheme is slower, but this will not reduce the overall performance at all. Obviously, the proposed MBE scheme is superior to the non-Booth design and conventional MBE schemes.

4 MLCSMA Algorithm

This section introduces the properties of the conditional-sum adder (CSMA) and the conditional-carry adder (CCA). CSMA and CCA are then used to construct the novel multiple-level conditional-sum adder (MLCSMA). The performance evaluation is given in the last subsection.

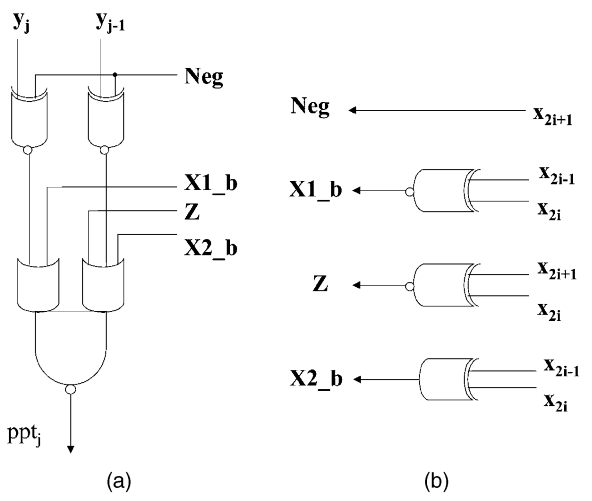


Fig. 3. The Encoder and Decoder for the new MBE scheme. (a) 18-transistor decoder. (b) Simple encoder.

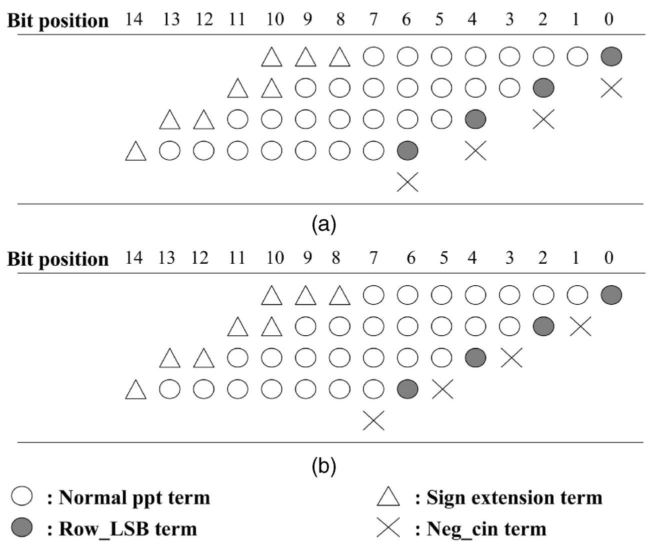

Bit position 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
(a)
Bit position 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
(b)
○ : Normal ppt term
△ : Sign extension term
● : Row_LSB term
✕ : Neg_cin term

Fig. 4. 8×8 MBE partial product array. (a) Traditional MBE partial product array. (b) New MBE partial product array.

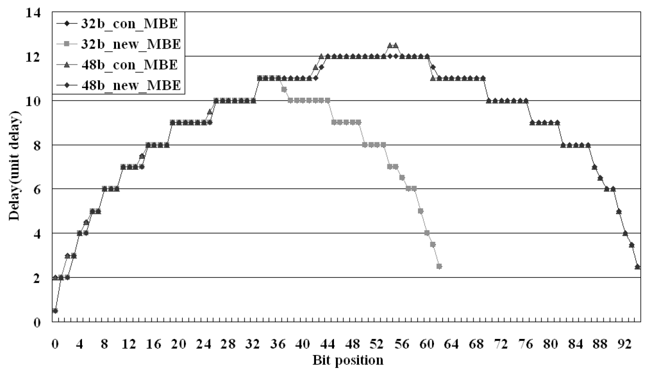


Fig. 5. PPRT delay profiles of the conventional array and the modified array.

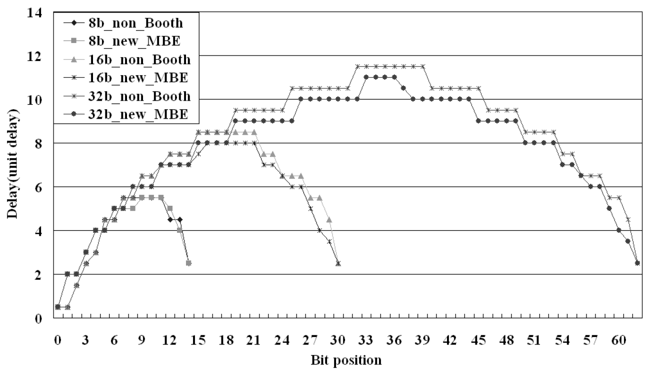


Fig. 6. PPRT delay profiles of non-Booth scheme and new MBE scheme.

4.1 Hybrid Structure Using CCA and CSMA

The previous designs in [14], [15], [16] partition the final adder into two or three blocks. Since the delay profile generated by the TDM algorithm changes gradually, these partition methods do not fully exploit the delay profile. The method presented herein is developed to optimize the final adder bit by bit. CSMA is chosen as the basic building block because the performance of a tree-based CSMA is comparable to that of a carry-lookahead adder. Moreover, the structure of a tree-based CSMA is very regular. The conventional CSMA [24] completes n-bit addition within $(\lceil log_2(n)\rceil + 1) \times t_{MUX}$. Conditional-carry adder (CCA) was proposed to save area [25] and its delay is about $(\lceil log_2(n)\rceil + 1.5) \times t_{MUX}$. Therefore, we use a hybrid structure as shown in Fig. 7. Note that in Figs. 7, 8, and 9, the blocks with gray are actually composed of two gates. For example, there are two 2-input multiplexers in a "mux" block with gray. In Fig. 7, CCA structure is used to save area for the first four bits. For the last block, from bits 4 to 7, CSMA structure is used to retain speed. The hybrid structure has the same latency as CSMA and saves area for those blocks using CCA. The hybrid adder shown in Fig. 7 was designed for equal delay input profile. However, as we will show in the following subsections, the new algorithm can be applied to any input delay profile.

4.2 Fundamentals of the MLCSMA Algorithm

The algorithm initially receives the outputs with the arrival timing information from PPRT and then processes the inputs from LSB to MSB. Let n denote the bit position index and c_n denote the carry signal for the nth bit. To construct CCA or CSMA, all the r terms, g terms, and p terms must first be generated from the inputs. The equations for the r, g, and p are:

$$r_n = a_n + b_n \tag{4}$$

$$g_n = a_n \cdot b_n \tag{5}$$

$$p_n = a_n \oplus b_n. \tag{6}$$

The symbols "+," "$\cdot$," and "$\oplus$" are "or," "and," and "exclusive-or" Boolean operations, respectively. The design presented here uses 2-input NOR gate and 2-input NAND gate to generate $\overline{r_n}$ and $\overline{g_n}$, respectively, to reduce delay. They are then corrected by using inverted MUX (MUXI_2 in Table 1) gates. The higher levels of r term and g term are generated by using the following equations:

$$r_{i,n} = r_{i-1,n} \cdot r_{j,m} + \overline{r_{i-1,n}} \cdot g_{j,m} \tag{7}$$

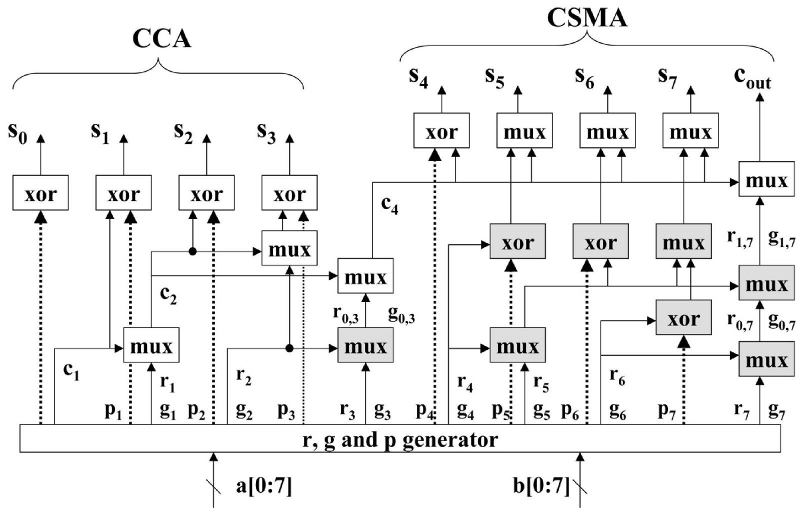


Fig. 7. 8×8 hybrid structure adder.

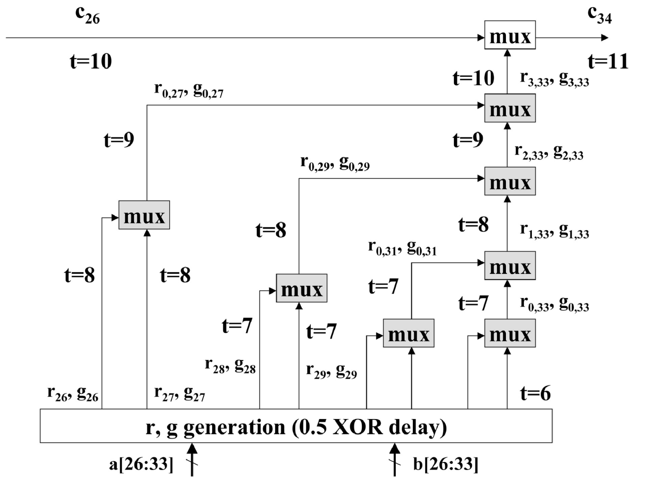


Fig. 8. An example of carry generation for MSB part. An 8-bit block is completed in one unit delay.

$$g_{i,n} = g_{i-1,n} \cdot r_{j,m} + \overline{g_{i-1,n}} \cdot g_{j,m}. \qquad (8)$$

For the subscript index of r and g, the first index denotes the level and the second index denotes the bit position. $r_{-1,n}$ and $g_{-1,n}$ are defined as r_n and g_n, respectively. Note that the values of "j" and "m" are determined by the previous level. To generate carry and sum, the following equations are used:

$$c_{n+1} = c_n \cdot r_n + \overline{c_n} \cdot g_n \qquad (9)$$

$$s_n = c_n \oplus p_n. \qquad (10)$$

Observation 1. By recursively decomposing (9) and representing carry signal by r, g, and c_k, where the $k = 2^i$ and $i = 0, 1, \ldots, \lfloor log_2(n) \rfloor$, we can find the same equations that generate carry signals for CSMA or CCA. However, k does not have to be 2^i. In other words, we can generate arbitrary carry or sum signal by the selected carry signals and intermediate r and g terms.

Observation 2. For CSMA, p_n is combined with r_{n-1} and g_{n-1} using two XOR gates at the beginning of addition. For CCA, p_n is combined with c_n using one XOR gate to generate the sum at the last step.

The two observations can be deduced from the two basic properties, associative and noncommute properties, of the multiplexing operation used in (7)-(9). For example, to generate c_{10}, c_8 (k = 8) is combined with the intermediate r and g terms, r_8, g_8, r_9, and g_9, as in the tree-based CSMA or CCA. Alternatively, c_{10} can be generated by combining c_7 with $r_{0,8}$, $g_{0,8}$, r_9, and g_9, where the $r_{0,8}$ and $g_{0,8}$ are generated by combining the r_7, g_7, r_8, and g_8 first. Note that, when applying associative property to combine the multiplexing operations, one should not violate the noncommute property. For example, $r_{i,n}$ and $g_{i,n}$ will be incorrect if ($r_{i-1,n}$, $g_{i-1,n}$) and ($r_{j,m}$, $g_{j,m}$) are exchanged in (7) and (8). Based on the above two observations and using the equations from (4) to (10), we developed our MLCSMA algorithm. Its pseudocode is listed in the next subsection. For the last block, CSMA block is used to obtain the optimal performance. For the other blocks, the sum signals and carry signals are generated by using CCA structure to save area. As we will show in this subsection, the partition of final adder is determined by the delay profile and cell properties.

To illustrate the MLCSMA algorithm, we can examine a typical example, shown in Fig. 8, first. The carry signal c_{26} arrives at t = 10 and the inputs a_{26}, a_{27}, b_{26}, and b_{27} arrive at t = 7.5. Because the timing difference is sufficiently large to span another level of CCA, the program constructs a 2-bit CCA section and generates $r_{0,27}$ and $g_{0,27}$ by using (7) and (8). Similarly, 2-bit CCA sections are constructed for the successive 6 bits. However, since the timing difference of $r_{0,31}$ and $r_{0,33}$ is smaller than one unit delay, both the level spanned at n = 31 and the level spanned at n = 33 are terminated and combined to form $r_{1,33}$ and $g_{1,33}$ by using (7) and (8). By repeating these steps, c_{33} is produced at t = 11. The sum signals from n = 26 to n = 33 are generated by using the method described in [25]. In this example, the algorithm spans four levels and completes an 8-bit block with one unit delay.

The algorithm can also handle LSB part easily. Two examples are shown in Fig. 9. In Fig. 9a, because a_n and c_n arrive at the same time, an FA is fast enough to generate c_{n+1}. In Fig. 9b, a typical case for the LSB part is shown. It takes two unit delays to complete 8-bit addition.

From the above examples and discussion, we can see that the algorithm can handle any delay profile easily. Because CSMA is used at the last block, the area of a MLCSMA is

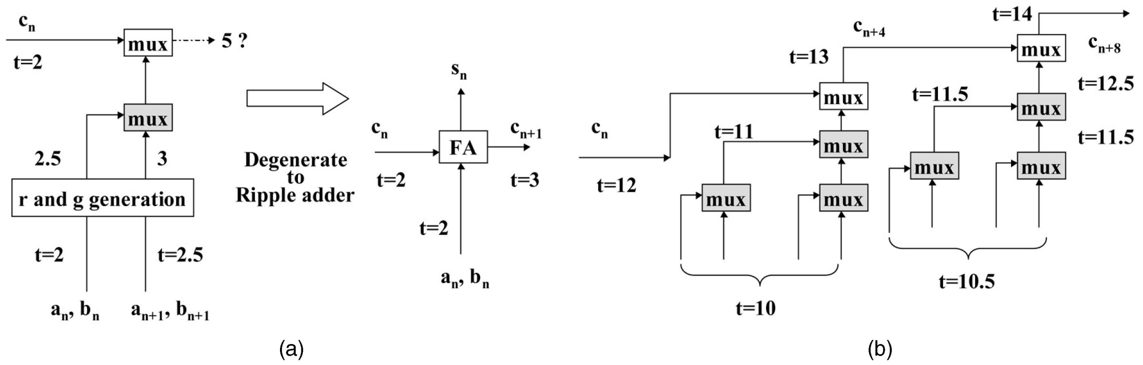


Fig. 9. Examples for LSB part: (a) Ripple carry adder is fast enough; (b) typical case for the LSB-part. Two 4-bit blocks are completed in two unit delays.

always between CSMA and CCA. Moreover, the delay will be smaller than the delay of a CSMA due to the multiple-level configuration and less capacitance loading along the carry propagation path. Most importantly, this algorithm is technology independent and solves the final adder problem in linear time. Thus, the algorithm can be applied to any delay profile.

4.3 Pseudocode for MLCSMA Algorithm

```
    n: current bit position.
    M: current level.
    L: current level length.
    next_L: next level length.
    $t_n$: max arrival time of the nth bit.
    $t\_cur\_c$: current level carry delay(same for r, g delay).
    $t\_pre\_c$: previous level carry delay.
    $t\_next\_c$: next level carry delay.
    last_block: block reaching the last bit or not.
    $t_{MUX}$: delay of 2-input multiplexer.
    $t_{NOR\_2}$: delay of 2-input NOR gate (generate $\overline{r_n}$).
Begin function
  Use a half-adder for the first bit;
  n=n+1; begin_pos=0;
While $(n < adder\_width)$
  Check n and adder_width, set last_block.
  $t\_cur\_c$ = max($t\_cur\_c$, $t\_next\_c$)+$t_{MUX}$;
  if $(t\_cur\_c - t_n > 0.5 \times (t_{NOR\_2} + t_{MUX}))$
    Build_next_level(t_cur_c, n, m);
// Update current level length
    L=L+next_L;
    if last_block is true
      Use CSMA architecture:
      Construct sum signals for current level
        from n to begin_pos+L;
    else
      Use CCA architecture:
      Generate all carries of this level;
      Construct sum signal for previous level;
      $t\_cur\_c$=max($t\_cur\_c$, $t\_next\_c$)+$t_{MUX}$;
// Timing difference is small, ripple adder is fast enough.
  else
    Connect a FA for current level, current bit;
    n=n+1; L=L+1;
    Update timing information.
End While
End function
```

```
// Recursive function
Begin function
Build_next_level ($t\_pre\_c$, $pre\_m$, n)
  begin_pos=n; M=pre_m+1;
  Connect 2-bit CCA block;
    generate r and g of current position.
  Current level carry delay:
    $t\_cur\_c$=max($t_n$, $t_{n+1}$)+$t_{MUX}$;
  n=n+2; L=2;
  Check n and adder_width, set last_block.
```

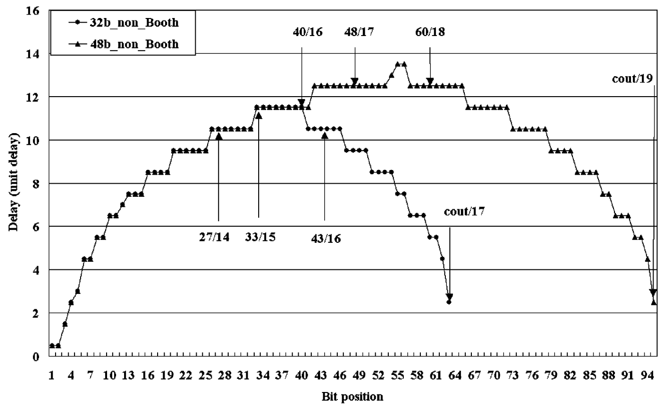


Fig. 10. Delay profiles from [13] and [16] and partition boundaries for final adder.

```
While $(n < adder\_width)$
  Find t_max=max($t_n$, $t_{n+1}$);
  Check t_max and previous level carry info., t_pre_c:
  if $((t\_cur\_c - t\_max) > 0.5 \times (t_{NOR\_2} + t_{MUX}))$ and
      $(t\_pre\_c - t\_cur\_c) > 0.5 \times t_{MUX})$
// Timing difference is large enough,
  span next level recursively.
  Build_next_level(t_cur_c, M, n);
// Update current level length
      L=L+next_L;
      n=begin_pos+L;
      $t\_cur\_c$= max($t\_cur\_c$, $t\_next\_c$)+$t_{MUX}$;
      Generate all carries required for next level.
  else
    if $((t\_pre\_c - t\_cur\_c) > 0.5 \times t_{MUX})$ and
        $(t\_pre\_c - t_n) > 0.5 \times (t_{NOR\_2} + t_{MUX}))$
        Connect a FA for bit n.
        n=n+1; L=L+1;
  else
    Finish this level;
    Return to previous level with L, n, t_cur_c info.
End While
End function
```

4.4 Performance Evaluation

To evaluate the performance of the algorithm, the new design generated from the algorithm is compared with the design from [16] first. Fig. 10 illustrates the delay profiles of a 32×32 PPRT from [16] and a 48×48 PPRT from [13]. Note that the figure includes the delay of generating partial products (0.5 unit delays). The design in [16] partitions the final adder into three blocks. The central block is an 8b block from n = 33 to n = 40. An 8b-CSMA, which takes 3.5 unit delays to generate carry_out signal, is used to process this block. The result is then combined with the LSB adder block and with the MSB adder block. Therefore, this design takes 5.5 unit delays (8b-CSMA + two MUX selections) to complete. Thus, the design completes the addition at t = 11.5 + 5.5 = 17. The new design uses MLCSMA from n = 1 to the last bit. Fig. 10 shows several block boundaries. For example, the carry signal c_{33} generated at t = 15 is denoted by 33/15 in the figure. For the 32×32 PPRT, the

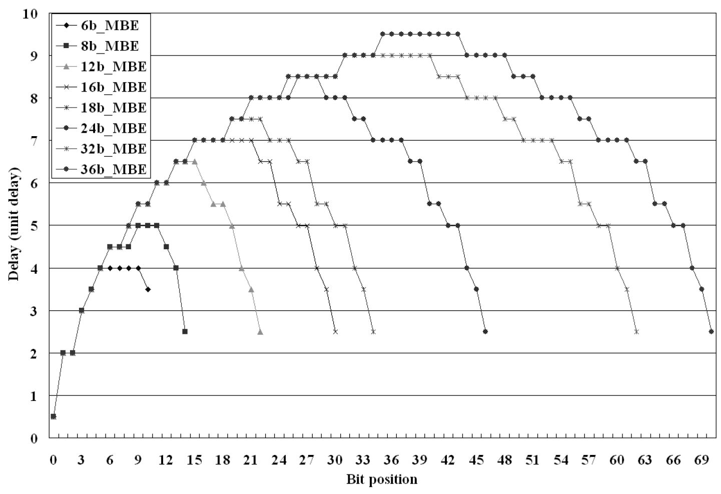


Fig. 11. Delay profiles for several sizes of new MBE scheme.

TABLE 5
Performance Improvement for $n \times n$ Multiplier

Multiplier size (=n)	6	8	12	16	18	24	32	36
Maximum PPRT delay (unit delay)	4	5	6.5	7	7.5	8.5	9	9.5
CSMA delay (unit delay)	5	5	6	6	7	7	7	8
MLCSMA delay (unit delay)	4	4.5	4.5	5.5	5.5	5.5	6.5	6.5
Improvement of Final Adder	20.0%	10.0%	25.0%	8.3%	21.4%	21.4%	7.1%	18.8%
Improvement of Multiplier	11.1%	5.0%	12.0%	3.8%	10.3%	9.7%	3.1%	8.6%

addition is completed at t = 17. The performance is the same in terms of gate delay in this case. For the 48×48 PPRT delay profile, our design completes addition at t = 19. However, it is hard to apply the method in [16] to the 48×48 PPRT delay profile because no adder partition methodology was provided. The 8-bit block in 32×32 PPRT delay profile is a special case.

To examine the performance of the MLCSMA algorithm for the optimized MBE scheme, the delay profiles are generated by using the new MBE scheme with TDM algorithm. Fig. 11 shows the delay profiles. The MLCSMA algorithm is then applied to these delay profiles to measure their performance. Note that, since Type_I FA and Type_II FA are used in our TDM program, the delay profiles have shorter latency than the delay profiles in [12], [13]. Depending on the heuristics and the basic cells used in the TDM program, various different delay profiles may be obtained. Table 5 lists the delays for MLCSMA and CSMA. Note that, for an $n \times n$ multiplier, the delay of the final adder based on tree-based CSMA is $(\lceil log_2(2 \times n - 2)\rceil + 1) \times t_{MUX}$. The performance improvement of the final adder in the table is calculated as:

$$Improvement = \frac{(CSMA\_delay - MLCSMA\_delay)}{CSMA\_delay} \times 100\%. \qquad (11)$$

For the 12×12 multiplier, the performance improvement of the final adder is 25 percent. The average performance improvement of multiplier is around 8 percent.

5 Simulation

Cells are selected from the standard cell library [17] with the properties described in Table 1 and Table 2. The PPRT is optimized by TDM algorithm according to the timing information given in Table 2. Then, the MBE scheme and the optimized PPRT are implemented by using the selected cells. We use Verilog code to describe the design. The code is fed into Synopsys Design_Analyzer to insert appropriate buffers. In this step, the only work done by Design_Analyzer is buffer insertion. No cell is changed or removed from our design. The PPRT delay profile, which considers buffer delay, capacitance loading, and driving capability, is obtained from Design_Analyzer. The delay profile is then used as the inputs to the MLCSMA algorithm. In this step, the maximum path delay of multiplexer is used in our MLCSMA algorithm. The maximum path delay of the multiplexer is 0.41ns. After the final adder is generated, the PPRT and the final adder are merged into a complete multiplier. Again, the Verilog code of the multiplier is fed into Design_Analyzer to do buffer insertion.

We estimate the delay of PPRT by multiplying the delay profile in Fig. 11 with unit delay (0.46ns). Fig. 12 shows the result and the delay profiles from Design_Analyzer. Clearly, the estimated delay profiles of PPRT are almost the same as those from Design_Analyzer. The delay of final adder can also be found in Fig. 12. For example, the delay of final adder is 2.3ns (= 6.46ns-4.16ns) for 32×32 multiplier. Because the delay of the 2-input multiplexer used to construct MLCSMA is smaller than one unit delay, the delay of final adder should be normalized with respect to one MUX delay. Therefore, the final adder delay is

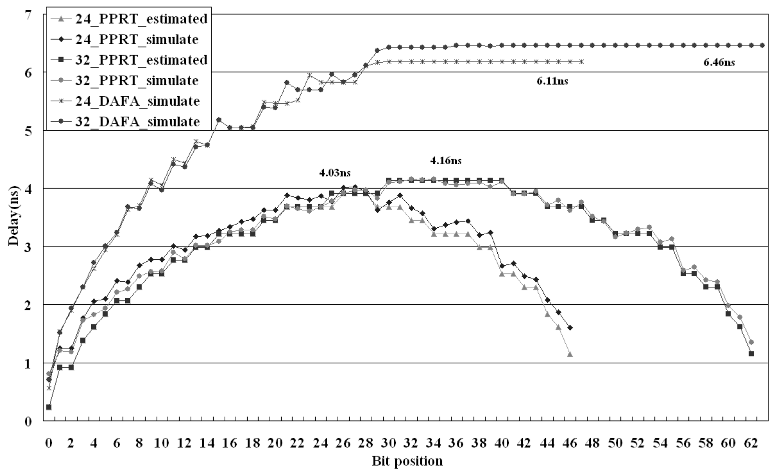


Fig. 12. Gate-level simulation and the estimated delay from normalized delay model.

2.3/0.41=5.6 MUX delays. The error for the estimated delay is 6.5-5.6 = 0.9 MUX delays. Similarly, the error of the estimated final adder delay for 24×24 multiplier is 0.43 MUX delays. In both cases, the error is smaller than one MUX delay.

6 Conclusion

In this paper, we have shown how to build a high-speed Booth encoded parallel multiplier. By combining the proposed new MBE recoder and the modified partial product array, the MBE-based multiplier can perform better than the non-Booth based design. For the final adder, a new algorithm that optimizes final adder incrementally is proposed. The proposed algorithm solves final adder problem efficiently for any size and shows performance improvement up to 25 percent for the final adder. This work has been verified by gate level simulation that considers the effects of buffer delay, capacitance loading, and driving capability. The simulation results meet the estimated delay closely.

The area and timing properties for several MBE schemes investigated in Section 3 suggest that the effects of using different modified Booth recoders should be considered more carefully, especially when different logic styles are available. The proposed MLCSMA algorithm may also be applied to other unequal delay profile problems. For example, the signal delay profile from the output of a direct form filter or a multiple-constant multiplication module also has unequal delay profile characteristics. The final addition can be accelerated in these designs by using the novel MLCSMA algorithm. The various effects of using MLCSMA algorithm will be examined in the future studies.

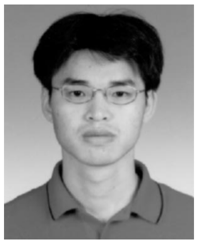
Acknowledgments

The authors would like to acknowledge the valuable suggestions from the reviewers and the associative editor. The work was supported by the National Science Council, Taiwan, Republic of China, under Grant NSC88-2215-E009-066.

References

[1] A.D. Booth, "A Signed Binary Multiplication Technique," *Quarterly J. Mechanical and Applied Math.*, vol. 4, pp. 236-240, 1951.
[2] O.L. MacSorley, "High Speed Arithmetic in Binary Computers," *Proc. IRE*, vol. 49, pp. 67-91, 1961.
[3] D. Villeger and V.G. Oklobdzija, "Analysis of Booth Encoding Efficiency in Parallel Multipliers Using Compressors for Reduction of Partial Products," *Proc. IEEE 27th Asilomar Conf. Signals, Systems, and Computer*, vol. 1, pp. 781-784, 1993.
[4] D. Villeger and V.G. Oklobdzija, "Evaluation of Booth Encoding Techniques for Parallel Multiplier Implementation," *Electronics Letters*, vol. 29, no. 23, pp. 2,016-2,017, Nov. 1993.
[5] C.S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Trans. Computers*, vol. 13, no. 2, pp. 14-17, Feb. 1964.
[6] L. Dadda, "Some Schemes for Parallel Multiplier," *Alta Frequenza*, vol. 34, pp. 349-356, Mar. 1965.
[7] A. Weinberger, "4:2 Carry-Save Adder Module," *IBM Technical Disclosure Bulletin*, vol. 23, Jan. 1981.
[8] M. Nagamatsu et al., "A 15 ns 32×32-bit CMOS Multiplier with an Improved Parallel Structure," *Digest of Technical Papers, IEEE Custom Integrated Circuits Conf.*, 1989.
[9] P. Song and G. De Micheli, "Circuit and Architecture Trade-Offs for High-Speed Multiplication," *IEEE J. Solid State Circuits*, vol. 26, no. 9, Sept. 1991.
[10] D. Zuras and W.H. McAllister, "Balanced Delay Trees and Combinatorial Division in VLSI," *IEEE J. Solid-State Circuits*, vol. 21, pp. 814-819, Oct. 1986.
[11] I. Koren, *Computer Arithmetic Algorithms*, chapter 6, pp. 112-114. Prentice Hall Int'l, 1993.
[12] V.G. Oklobdzija, D. Villeger, and S.S. Liu, "A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach," *IEEE Trans. Computers*, vol. 45, no. 3, pp. 294-306, Mar. 1996.
[13] P.F. Stelling, C.U. Martel, V.G. Oklobdzija, and R. Ravi, "Optimal Circuits for Parallel Multipliers," *IEEE Trans. Computers*, vol. 47, no. 3, pp. 273-285, Mar. 1998.
[14] M.D. Ercegovac et al., "Fast Multiplication without Carry-Propagate Addition," *IEEE Trans. Computers*, vol. 39, no. 11, Nov. 1990.
[15] R.K. Kolagotla et al., "VLSI Implementation of a 200-Mhz 16×16 Left-to-Right Carry-Free Multiplier in 0.35μm CMOS Technology for Next-Generation DSPs," *Proc. IEEE 1997 Custom Integrated Circuits Conf.*, pp. 469-472, 1997.
[16] P.F. Stelling and V.G. Oklobdzija, "Optimal Designs for Multipliers and Multiply-Accumulators," *Proc. 15th IMACS World Congress Scientific Computation, Modeling, and Applied Math.*, A. Sydow, ed., pp. 739-744, Aug. 1997.
[17] Passport 0.35 micron, 3.3 volt, Optimum Silicon SC Library, CB35OS142, Avant! Corporation, Mar. 1998.
[18] G. Goto et al., "A 4.1ns compact 54×54-b Multiplier Utilizing Sign-Select Booth Encoders," *IEEE J. Solid-State Circuits*, vol. 32, no. 11, pp. 1,676-1,682, Nov. 1997.

[19] G. Goto et al., “A 54×54-b Regularly Structured Tree Multiplier,” *IEEE J. Solid-State Circuits,* vol. 27, no. 9, Sept. 1992.
[20] R. Fried, “Minimizing Energy Dissipation in High-Speed Multipliers,” *Proc. 1997 Int'l Symp. Low Power Electronics and Design,* pp. 214-219, 1997.
[21] N.H.E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective,* second ed., chapter 8, p. 520. Addison Wesley, 1993.
[22] J. Fadavi-Ardekani, “M×N Booth Encoded Multiplier Generator Using Optimized Wallace Trees,” *IEEE Trans. VLSI Systems,* vol. 1, no. 2, June 1993.
[23] A.A. Farooqui et al., “General Data-Path Organization of a MAC Unit for VLSI Implementation of DSP Processors,” *Proc. 1998 IEEE Int'l Symp. Circuits and Systems,* vol. 2, pp. 260-263, 1998.
[24] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design,* chapter 3, p. 81. John Wiley & Sons, 1976.
[25] K.H. Cheng et al., “The Improvement of Conditional Sum Adder for Low Power Applications,” *Proc. 11th Ann. IEEE Int'l ASIC Conf.,* pp. 131-134, 1998.



Wen-Chang Yeh received the BS degree in electronics engineering from National Chiao Tung University in 1997. He is currently pursuing the PhD degree at the National Chiao Tung University. His research interests include computer arithmetic, 3D graphics, signal processing, and computer architecture.

Chein-Wei Jen (S'78-M'84) received the BS degree from National Chiao Tung University in 1970, the MS degree from Stanford University, Stanford, California, in 1977, and the PhD degree from National Chaio Tung University in 1983. He is currently with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, as a professor. During 1985-1986, he was with the University of Southern California, Los Angeles, as a visiting researcher. His current research interests include VLSI design, digital signal processing, processor architecture, and design automation. Dr. Jen is a member of the IEEE and of Phi Tau Phi.