Design and Implementation of High-Speed and Energy-Efficient Variable-Latency Speculating Booth Multiplier (VLSBM)

Shin-Kai Chen, Chih-Wei Liu, *Member, IEEE*, Tsung-Yi Wu, and An-Chi Tsai

***Abstract*—Data hazards cause severe pipeline performance degradation for data-intensive computing processes. To improve the performance under a pessimistic assumption on the pipeline efficiency, a high-speed and energy-efficient VLSBM is proposed that successively performs a speculating and correcting phase. To reduce the critical path, the VLSBM partial products are partitioned into the $(n-z)$-bit least significant part (LSP) and the self-reliant $(n+z)$-bit most significant part (MSP), and an estimation function stochastically predicts the carry to the MSP, thereby allowing independent calculation of the partial-product accumulation of parts. When a carry prediction is accurate, the data dependence is hidden and the correcting phase is bypassed, thereby ensuring the potential speed-up of the pipelined datapath. If a prediction is inaccurate, the speculation is flushed and the correcting phase is executed to obtain the exact multiplication. The simulation results verify the effectiveness of the proposed VLSBM. When applied to a DSP algorithm with a data hazard (or dependence) probability P_D, $0 \leq P_D \leq 1$, the results show that the proposed VLSBM outperforms the original Booth multiplier and the fastest conventional well-pipelined modified Booth multiplier when $P_D > 0.32$. For the case of high P_D with $P_D \approx 1$, the proposed VLSBM improves approximately 1.47 times speedup against the fastest conventional pipelined Booth multiplier (@UMC 90 nm CMOS) and, furthermore, approximately 25.4% of energy per multiplication and 7% of area are saved. By examining multiplications during three multimedia application processes (i.e., JPEG compression, object detection, and H.264/AVC decoding), the proposed VLSBM improves the speed-up ratio by approximately 1.0 to 1.4 times, and reduces the cycle count ratio by approximately 1.3 to 1.8 times in comparison to the fastest conventional two-stage pipelined Booth multiplier.**

***Index Terms*—Adaptive carry estimation, error compensation, speculating multiplier.**

I. Introduction

THE parallel multiplier is a fundamental element of various computing and signal processing applications [1]. The latency for multiplication is generally divided into three steps: 1) the generation of partial products; 2) a reduction-tree process that simplifies partial products into two rows; and 3) the final addition of the result by a carry-propagation adder (CPA) [2]–[6]. Because all partial product bits within each column are summed in parallel, the Wallace tree compression is superior during the second step. For the parallel multiplier CPA, a carry-lookahead adder (CLA) has been frequently applied because of its reduced critical path.

On synchronous VLSI designs, synthesis tools employ a significant volume of logic- and gate-level optimizations to minimize the critical path. For example, the circuit re-structuring method reduces the logic depth, thereby decreasing the circuit delay, whereas the gate-sizing approach replaces the critical path cells with those of high driving strength. Although these techniques effectively reduce the critical path, they require an additional overhead of silicon area that inherently increases the energy dissipation [1]. Fig. 1 shows an example of the energy-delay plot of an 8 × 8 carry-save-array (CSA) multiplier using a UMC 90 nm CMOS cell library. The x- and y-axes in Fig. 1 represent the synthesis timing constraint, and the energy per multiplication of the multiplier (evaluated by 10,000 random multiplications), respectively. As shown in Fig. 1, a period of 1.9 ns of the CSA multiplier consumes up to 1.1 pJ per multiplication. When operated for 1.6 ns using an identical architecture, the CSA multiplier speed-up increases by approximately 19%; however, more than approximately 82% of the energy dissipates.

Pipelining is an effective approach to alleviate the drawback from aggressive logic- and gate-level timing optimizations while maintaining sufficient throughput. Fig. 2 shows a multiplier-enhanced five-stage pipelined (IF:ID:EXE:M:WB) RISC. Because no operation in the M-stage exists for any arithmetic instruction, a fast two-stage pipelined multiplier can be integrated to enhance multiplicative performance. However, true data dependence causes a pipeline stall cycle to be inserted, despite a forwarding path being employed (Fig. 2(b)). Consequently, under the assumption of poor pipeline efficiency, the throughput benefit of the pipelined datapath is eventually undermined by data hazards [1].

It is possible to hide data dependence by employing a speculative functional unit (SFU), which is an arithmetic unit that employs a predictor for the carry signal [7]. Without waiting for the true carry propagation, the SFU predicts the carry of one or more cells in the design. Similar to a microprocessor branch predictor, a true carry prediction causes the data dependence to be hidden. Conversely, when inaccurate carry predictions occur,

Manuscript received September 04, 2012; revised December 05, 2012; accepted January 17, 2013. Date of current version September 25, 2013. This work was supported in part by the Nation Science Council, Taiwan, under Grant NSC101-2220-E-009-029- and Information and Communications Research Labs (ICL), Industrial Technology Research Institute (ITRI), Taiwan, under Grant 101-EC-17-A-05-01-1111, and by a grant of chip fabrication in the National Chip Implementation Center (CIC), Taiwan. This paper was recommended by Associate Editor C. H. Chang.

The authors are with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan (e-mail: cwliu@twins.ee.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2013.2248851

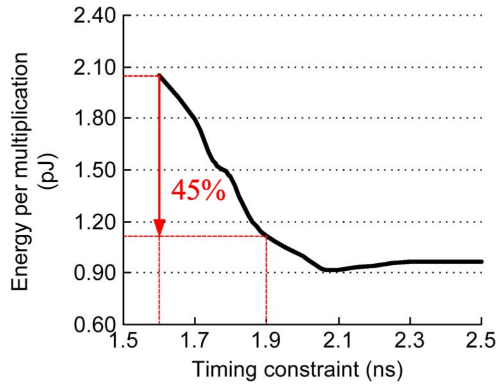


Fig. 1. Energy-delay plot of 8 × 8 CSA multiplier (@ UMC 90 nm CMOS).

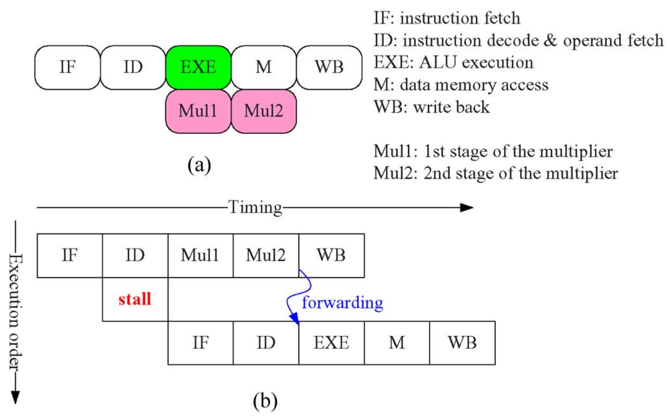


Fig. 2. Multiplier-enhanced 5-stage pipelined RISC: (a) datapath, (b) timing diagram of two dependent instructions.

error detection logic and the associated error recovery are necessary to generate (or reconfirm) the real result. However, the cumbrous overhead or complex control could have an adverse impact on the SFU performance [7].

By assuming that each output bit of the n-bit adder is dependent only on the preceding k bits (when k is significantly less than n), Verma *et al.* proposed a fast n-bit almost-correct adder (ACA) [8]. Compared with the n-bit CLA provided by DesignWare, the simulation results (@ UMC 0.18 μm CMOS) in [8] increased the speed-up of the n-bit ACA throughput by approximately 1.5 times when $n > 64$. Unfortunately, an error occurs when the carry chains equal or exceed k bits. To improve the addition reliability, previous research proposed a variable-latency speculating adder (VLSA) [8] by integrating error detection logic into the ACA, which achieved a critical path almost identical to that of a conventional CLA, and an error recovery process (i.e., the original CLA) that reproduced the true addition. Therefore, the proposed n-bit VLSA [8] required approximately 50% more silicon area overhead to achieve a speed-up similar to that of the DesignWare CLA. Previous research proposed a modified VLSA that improved the accuracy to 99.46% when applied to a fixed or specific application [9]. However, this required 25% more silicon area [9] than the original VLSA [8].

Although the ACA is faster than the VLSA, it is less reliable. However, they are both potentially unsuitable for the fast and reliable parallel multiplier CPA. Based on a probabilistic result-speculation technique, this study proposes a design for an efficient variable-latency speculating multiplier (VLSM). The VLSM design concept and block diagram in Fig. 3 comprise the following two operating phases: 1) the speculating phase and 2) the correcting phase. To reduce the critical path, the multiplier partial products are partitioned into the $(n+z)$-bit most significant parts (MSP) and the $(n-z)$-bit least significant parts (LSP). During the speculating phase, the partial-product accumulations of the LSP and the MSP are performed in parallel and independently. Without waiting the true carry from the LSP, the carry σ_z to the $(n+z)$-bit MSP is generated by the carry estimation function (Fig. 3(a)). Consequently, the speculating output of the VLSM is

$$\sum_{n+z}(MSP) + 2^{n-z}\sigma_z + \sum_{n-z}(LSP), \tag{1}$$

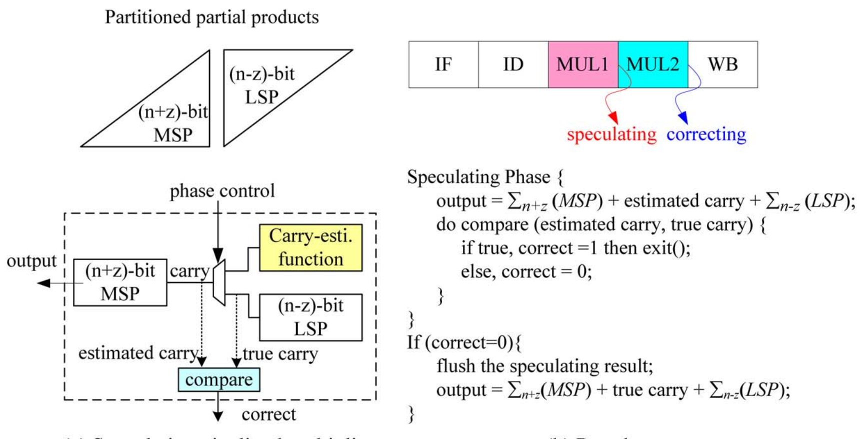
which can be forwarded to the successive instruction for further processing. Note that $\Sigma_{n+z}(MSP)$ in (1) is the partial-product summation of the $(n+z)$-bit MSP and $\Sigma_{n-z}(LSP)$ is the summation of the $(n-z)$-bit LSP minus the carry. If σ_z is true, the corresponding correcting phase is bypassed and the data dependence is hidden; thus, the multiplier stall cycle is saved. If an inaccurate prediction occurs, the speculation is flushed and the exact multiplication is obtained by summing $\Sigma_{n-z}(LSP)$, the true carry, and $\Sigma_{n+z}(MSP)$ during the correcting phase. Unlike the VLSA [8], the error recovery of the proposed VLSM is simple; thus, correcting the miss-carry-prediction does not diminish the overall performance of the VLSM.

Suppose that the accuracy probability of the carry estimator is $P_A(z)$, $0 \le P_A(z) \le 1$, and the VLSM critical stage delay is T_z. Different carry estimators with different z values produce various $P_A(z)$ and T_z values for the VLSM. This study defines the effective cycle time $T_E(z)$ of the VLSM as

$$T_E(z) = P_A(z)T_z + (1 - P_A(z)) \times 2T_z = 2T_z - P_A(z)T_z. \tag{2}$$

The design goal of the VLSM is to minimize $T_E(z)$. With a more accurate carry estimator (i.e., $P_A(z) \to 1$), the throughput obtained by the VLSM is more efficient. However, ensuring high-accuracy probability may inevitably increase the critical path T_z of the VLSM. Thus, the tradeoff between $P_A(z)$ and the T_z of the VLSM must be considered carefully.

To examine the cost function carefully, this study proposes a high-speed, area-/energy- efficient 16 × 16 variable-latency two-stage pipelined modified Booth multiplier (VLSBM). Applied a DSP algorithm with a data hazard (or dependence) probability P_D, $0 \le P_D \le 1$, the optimized VLSBM outperforms not only the original Booth multiplier but the fastest, well-pipelined modified Booth multiplier if $P_D > 0.32$. Under a pessimistic assumption on the pipeline efficiency, i.e. for the case of high P_D with $P_D \to 1$, the proposed VLSBM improves approximately 1.47 times speedup against the fastest conventional pipelined Booth multiplier (@UMC 90 nm CMOS) and approximately 25.4% of energy per multiplication and 7% of silicon area are saved. Compared to the fastest conventional two-stage pipelined Booth multiplier, an examination of the multiplicative performance of the proposed VLSBM for three multimedia application processes (JPEG compression, object detection, and H.264/AVC decoding) shows that the proposed



```
Speculating Phase {
    output = $\Sigma_{n+z}$ ($MSP$) + estimated carry + $\Sigma_{n-z}$ ($LSP$);
    do compare (estimated carry, true carry) {
        if true, correct =1 then exit();
        else, correct = 0;
    }
}
If (correct=0){
    flush the speculating result;
    output = $\Sigma_{n+z}$($MSP$) + true carry + $\Sigma_{n-z}$($LSP$);
}
```

Fig. 3. Proposed variable-length speculating multiplier (VLSM): (a) Block diagram and (b) pseudo-program.

VSLBM obtains a performance speed-up and cycle count reduction ratios of approximately 1.0 to 1.4 times, and 1.3 to 1.8 times, respectively.

The remainder of this paper is organized as follows. Based on conditional probability theory, Section II briefly introduces three carry estimation schemes (Type-I, Type-II, and Type-III) that were developed originally by Liao *et al.* [20] for the fixed-width Baugh-Wooley multiplier. With different truncation widths of the partial products, the three types of carry estimation functions are generalized and developed, not only for Baugh-Wooley multipliers but for modified Booth multipliers. Based on the generalized carry estimation schemes, Section III details the optimized VLSBM design and presents simulation and experimental results that verify the effectiveness of the proposed VLSBM. Finally, Section IV presents a conclusion.

II. Generalized Carry Estimations for z-PT Fixed-Width Multipliers

By handling sign bits of the 2's-complement integers efficiently, the Baugh-Wooley multiplication algorithm is applied frequently to design regular multipliers. Given integers A and B,

$$A = -a_{n-1}2^{n-1} + \sum_{j=0}^{n-2} a_j 2^j;$$
$$B = -b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i, \tag{3}$$

where $a_j, b_i \in \{0,1\}$, a parallelogram of the n-row, $2n$-column partial products of $n \times n$ Baugh-Wooley multiplier is shown in Fig. 4, where $P_{i,j} \equiv b_i a_j$ and $\overline{P_{i,j}}$ is the binary inversion of $P_{i,j}$. Clearly, the n-bit MSP and the n-bit LSP divide the partial products into two groups. β and λ (Fig. 4) are defined by the summations of the most significant column and the remaining n-bit LSP columns, respectively. For simplicity, this study defines $\beta_i = P_{i,n-1-i}$ and $\alpha_j = P_{n-1-j,j}$, then $\beta = \Sigma_i \beta_i$. To reduce the critical path by diminishing the number of rows of partial products, the Booth's multiplication algorithm is developed. Table I shows that the radix-4 (or modified) Booth encoder transforms n-bit $B : (b_{n-1}, b_{n-2}, \ldots, b_0)$ into $\lceil n/2 \rceil$-digit $Y : (y_{\lceil n/2 \rceil -1}, y_{\lceil n/2 \rceil -2}, \ldots, y_0)$. With similar representation of $P_{i,j} = y_i a_j$, Fig. 5 shows the reduced $\lceil n/2 \rceil$-row, $2n$-column partial products of the $n \times n$ modified Booth multiplier. Hence, the results of the direct-truncation (DT), post-truncation (PT), or full-precision (FP) multipliers are expressed as

$$\begin{aligned}(A \times B)_{\mathrm{DT}} &= \sum_n (MSP);\\ (A \times B)_{\mathrm{PT}} &= \sum_n (MSP) + [2^{n-1}\beta + \lambda]_r\\ &= \sum_n (MSP) + 2^n \left[\frac{\beta}{2} + 2^{-n}\lambda\right]_r\\ &= \sum_n (MSP) + 2^n \sigma;\\ (A \times B)_{\mathrm{FP}} &= \sum_n (MSP) + 2^n\sigma + \sum_n (LSP), \end{aligned}\tag{4}$$

where σ is the carry-in to $\Sigma_n(MSP)$ and operator $[x]_r$ rounds x to the nearest integer. Without propagating the true σ from LSP to MSP, the DT multiplier is fast and economical, although it causes a significant loss in precision.

To design an efficient VLSM, this study defines the n-bit fixed-width z-PT multiplier, $1 \le z \le n$ (Fig. 6). By definition, the z-PT multiplier summates the $(n+z)$ most significant columns of the partial products, and subsequently rounds or truncates to the n-bit result. γ, β, and λ (Fig. 6) are the summations of the $(z-1)$ most significant columns, the z-th most significant column, and the remaining $(n-z)$ columns of the n-bit LSP, respectively. Note that Fig. 4 (or Fig. 5) is a special case of the z-PT multiplier. With γ, β, and λ, the n-bit z-PT multiplication can be expressed as

$$\begin{aligned}(A \times B)_{z\text{-PT}} &= \sum_n (MSP) + [\gamma + 2^{n-z}\beta + \lambda]_r\\ &= \sum_n (MSP) + 2^n[2^{-n}\gamma + \sigma_z]_r, \end{aligned}\tag{5}$$

where σ_z denotes the carry to $\Sigma_n(MSP)$ for the z-PT multiplier.

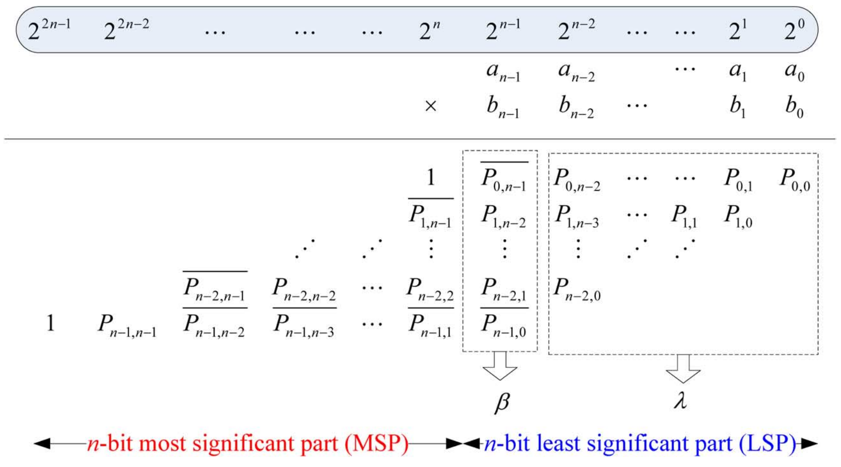

β
λ
n-bit most significant part (MSP)
n-bit least significant part (LSP)

Fig. 4. Partial products of n-bit A × B Baugh-Wolley multiplier.

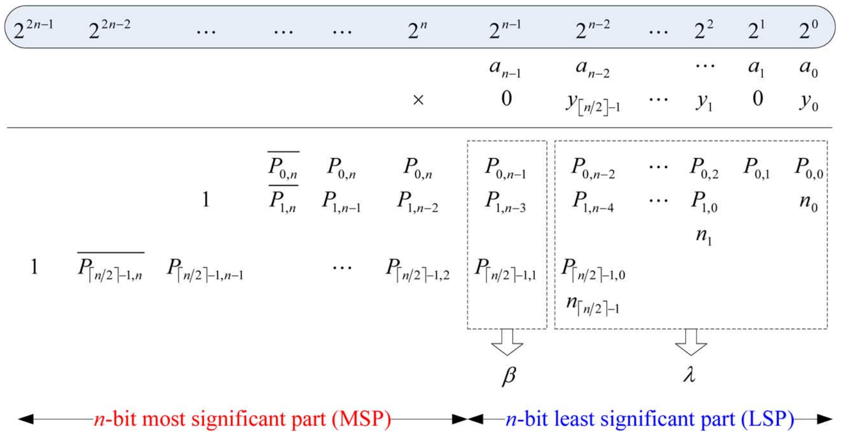

β
λ
n-bit most significant part (MSP)
n-bit least significant part (LSP)

Fig. 5. Partial products of n-bit A × B modified Booth multiplier.

TABLE I
MODIFIED BOOTH ENCODING. (NOTE THAT B_{-1} IS SET TO 0.)

b_{2i+1}	b_{2i}	b_{2i-1}	y_i	n_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	2	0
1	0	0	–2	1
1	0	1	–1	1
1	1	0	–1	1
1	1	1	0	0

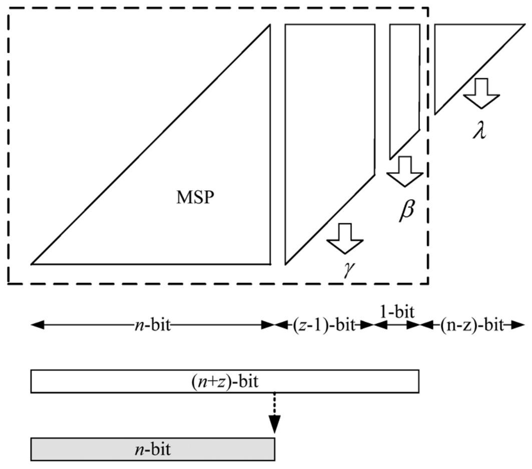


Fig. 6. The n-bit fixed-width z-PT multiplier.

If $z = 1$, previous studies [10]–[31] have proposed applying the β-values (to estimate λ) to compensate for the truncation error and obtain fast and accurate fixed-width multipliers. Previous studies first presented constant error compensations [10]–[12]; adaptive accurate error compensations, which relied on input bits, were subsequently proposed by [13]–[31]. By exhaustive simulations, Van *et al.* [14] generalized results obtained by Jou *et al.* [13] to develop the heuristic carry estimation for the Baugh-Wooley multiplier. By applying linear regression analysis, Jou *et al.* [15] proposed the simple carry estimation for the modified Booth multiplier. With exhaustive simulations for the modified Booth multiplier, Cho *et al.* [16] successfully proposed a closed form of the estimated carry, which is generated by non-zero Booth encoded digits. Except for 2's-complement multipliers, Juang and Hsiao [18] derived the statistical analysis for the binary-sign-digit (BSD) Booth multiplier. However, the conditional bits used in [18] were poorly selected. Based on conditional probability theory, Liao *et al.* [20] derived three adaptive carry estimation schemes for 2's-complement Baugh-Wooley multipliers (Type-I, Type-II, and Type-III). Subsequently, with a different approach from [20], Li *et al.* proposed the probabilistic carry estimation bias (PEB) for the modified Booth multiplier [28]. The PEB error performance was further improved in [31] by applying an adaptive conditional-probability estimator (ACPE). In brief, the conditional probability carry estimators do achieve the highest accuracy performance (not only for Baugh-Wooley multipliers [20] but also for modified Booth multipliers [31]); however, they are too complex for implementation [27].

To minimize hardware cost, the majority of the discussed error compensation approaches predicted σ_z by observing one column of partial products (i.e., $z = 1$). By examining further columns of partial products, the generalized carry estimator further minimizes the truncation error [23], [29]–[31]. Song *et al.* performed exhaustive simulations to adaptively compensate for the truncation error by establishing two types of binary threshold. [23] investigated z columns of the partial products ($z \leq 3$) to obtain accuracy carry estimations for the modified Booth multiplier. Wang *et al.* [29] performed time-consuming simulations that improved the results obtained by Cho [16], thereby resolving the truncation errors. Moreover, based on PEB, Chen *et al.* derived a generalized PEB, namely GPEB [30], for the design of a power-efficient modified Booth multiplier. Furthermore, they also showed that when $z = 3$, the ACPE Booth multiplier [31] improved speed-up by up to 25.2% speed (@ TSMC 0.18 μm CMOS) with only a 0.39 dB SNR loss compared to the results obtained by using the conventional Booth PT multiplier.

Without loss of generality, this study assumes that the bits of A and B are independently and identically distributed (i.e., i.i.d.) random variables with probabilities

$$P(a_j = 1) = P(a_j = 0) = \frac{1}{2}, \text{ and}$$
$$P(b_i = 1) = P(b_i = 0) = \frac{1}{2}.$$

Observing that the partial product $P_{i,j}$ is generated by input bits a_j and b_i, Liao *et al.* proposed the Type-I carry estimation scheme by conditioning a_j (or b_i) [20]. By further investigating dependencies among β, λ, A, and B (Fig. 4), they considered the row vector $\boldsymbol{P_{bi}}$ (i.e., $(P_{i,n-1-i}, P_{i,n-2-i}, \ldots, P_{i,0})$) and the anti-diagonal vector $\boldsymbol{P_{aj}}$ (i.e., $(P_{n-1-j,j}, P_{n-2-j,j}, \ldots, P_{0,j})$), respectively, where the entries are jointly dependent on β_I and α_j, respectively. It is evident that the n-bit LSP can be determined either by row vectors $\boldsymbol{P_{bi}}$ or by anti-diagonal vectors $\boldsymbol{P_{aj}}$, $0 \leq i, j \leq n-1$. Thus, the Type-II and the Type-III carry estimation schemes were subsequently proposed [20]. The three types of carry estimation schemes provide a systematic approach to discovering and examining the dependencies among σ, A, and B, and can be generalized to design various fixed-width z-PT multipliers.

A. Type I—Carry Estimation Conditioning on a_j

1) z-PT Baugh-Wooley Multiplier: Previous research [20] has shown that $E[P_{i,j}|a_j] = a_j/2$. Given by a_j, the value of $P_{i,j}$ can be estimated by its expectation (i.e., $a_j/2$). Therefore, the conditional expectation $\lambda_{BW-I}^{(a)}(z)$ of the z-PT Baugh-Wooley multiplier is expressed as

$$\begin{aligned} E\left[2^{-n}\lambda(z)|a_j\right] &= 2^{-n}\sum_{j=0}^{n-z-1}\sum_{i=0}^{n-z-1-j}\frac{a_j}{2}2^{i+j} \\ &= 2^{-z}\sum_{j=0}^{n-z-1}\frac{a_j}{2} - \sum_{j=0}^{n-z-1}\frac{a_j}{2}2^{j-n} \\ &\equiv \lambda_{BW-I}^{(a)}(z). \end{aligned} \tag{6}$$

Hence, by (5) and (6), conditioning on a_j, the generalized Type-I carry estimation function $\sigma_{BW-I}^{(a)}(z)$ of the z-PT Baugh-Wooley multipliers can be obtained using

$$\begin{aligned} E[\sigma_z|a_j] &= \left[2^{-z}\beta + \lambda_{BW-I}^{(a)}(z)\right]_r \\ &= \left[2^{-z}\beta + 2^{-z}\sum_{j=0}^{n-z-1}\frac{a_j}{2} - \sum_{j=0}^{n-z-1}\frac{a_j}{2}2^{j-n}\right]_r \\ &\cong \left[2^{-z}\left(\beta + \sum_{j=0}^{n-z-1}\frac{a_j}{2}\right)\right]_r \equiv \sigma_{BW-I}^{(a)}(z) \end{aligned} \tag{7}$$

Note that by (6), the generalized Type-I carry estimation scheme is hindered by high computation complexity, which is linearly proportional to the bit-width n.

2) z-PT Modified Booth Multiplier: By assuming the bits of B are i.i.d. random variables, the probabilities of the Booth encoded digit y_i, $0 \leq i \leq \lceil n/2 \rceil$, can be determined readily (Table II). From Table II, it concludes that $P(y_i \neq 0) = 1 - P(y_i = 0) = 3/4$. Hence, the following is obtained:

$$\begin{aligned} P(P_{i,j} \neq 0|y_i \neq 0) &= \frac{P(y_i a_j \neq 0 \cap y_i \neq 0)}{P(y_i \neq 0)} = \frac{\left(\frac{1}{2}\right)\left(\frac{3}{4}\right)}{\frac{3}{4}} \\ &= \frac{1}{2}. \end{aligned}$$

Consequently, the following conditional expectation of $P_{i,j}$ is

$$E[P_{i,j}|y_i \neq 0] = 1 \times 1/2 + 0 \times 1/2 = \frac{1}{2}.$$

With similar derivations, the following are obtained:

$$P(n_i = 1|y_i \neq 0) = \frac{1}{2} \text{ and } E[n_i|y_i \neq 0] = \frac{1}{2}.$$

Therefore,

$$E\left[P_{i,j}|y_i''\right] = \frac{y_i''}{2} \text{ and } E\left[n_i|y_i''\right] = \frac{y''}{2}, \tag{8}$$

where y_i'' stands for the non-zero Booth encoded digit, (i.e., $y_i \neq 0$). Note that considering only y_i'' when estimating σ is reasonable. Consequently, the generalized Type-I estimation of λ for the z-PT modified Booth multiplier (i.e., $\lambda_{Booth-I}(z)$) is

$$\begin{aligned} &E\left[2^{-n}\lambda(z)|a_j\right] \\ &\quad = 2^{-n}\sum_{i=0}^{\left\lceil \frac{n}{2}\right\rceil - \left\lfloor \frac{z}{2}\right\rfloor - 1}\left(\sum_{j=0}^{n-z-1-2i} E[P_{i,j}|y'']2^{2i+j}\right. \\ &\qquad\qquad \left. + E[P_{i,j}|y'']2^{2i}\right) \end{aligned}$$

TABLE II
PROBABILITY OF BOOTH ENCODED DIGITS

	$P(y_i=-2)$	$P(y_i=-1)$	$P(y_i=0)$	$P(y_i=1)$	$P(y_i=2)$
$i=0$	1/4	1/4	1/4	1/4	0
$i\neq 0$	1/8	1/4	1/4	1/4	1/8

$$= 2^{-n}\sum_{i=0}^{\lceil\frac{n}{2}\rceil-\lfloor\frac{z}{2}\rfloor-1}\left(\sum_{j=0}^{n-z-1-2i}\frac{y_i''}{2}2^{2i+j}+\frac{y_i''}{2}2^{2i}\right)$$
$$= 2^{-(z+1)}\sum_{i=0}^{\lceil\frac{n}{2}\rceil-\lfloor\frac{z}{2}\rfloor-1} y'' \equiv \lambda_{Booth-I}(z). \quad (9)$$

Hence, using (5), (9), the Type-I carry estimation $\sigma_{Booth-I}(z)$ of the z-PT modified Booth multiplier can be calculated by

$$\begin{aligned}E[\sigma_z|a_j] &= \left[2^{-z}\beta+\lambda_{Booth-I}(z)\right]_r\\ &= \left[2^{-z}\left(\beta+\frac{1}{2}\sum_{i=0}^{\lceil\frac{n}{2}\rceil-\lfloor\frac{z}{2}\rfloor-1} y''\right)\right]_r\\ &\equiv \sigma_{Booth-I}(z).\end{aligned} \quad (10)$$

Equation (10) not only coincides with the ACPE proposed in [31], but also is analogous to the results obtained by Wang *et al.* [29] and Cho *et al.* [16] if $z=1$. Furthermore, from (9), the computational complexity of the generalized Type-I carry estimation scheme for the z-PT modified Booth multiplier is linearly proportional to bit-width n.

B. Type II—Carry Estimation Conditioning on β

1) z-PT Baugh-Wooley Multiplier: The Type-II carry estimation considers row vectors $\boldsymbol{P_{bi}}$, $0\le i\le n-1$, in which the elements within $\boldsymbol{P_{bi}}$ are jointly dependent on β_i. Thus, given by β_i, [20] shows that

$$E[P_{i,j}|\beta_i]=\frac{2\beta_i+1}{6}. \quad (11)$$

It can be proven that the following equation is true when calculating λ in Fig. 4:

$$\sum_{j=0}^{n-2}\sum_{i=0}^{n-2-j}P_{i,j}2^{i+j}=\lambda=\sum_{i=0}^{n-2}\sum_{j=0}^{n-2-i}P_{i,j}2^{i+j}. \quad (12)$$

The right-hand side of (12) calculates λ based on the row-major order, whereas the left-hand side is ordered by the anti-diagonal order. Hence, provided by β_i, conditional estimation $\lambda_{BW-II}(z)$ of the z-PT Baugh-Wooley multiplier is

$$\begin{aligned}&2^{-n}\sum_{i=0}^{n-z-1}\sum_{j=0}^{n-z-1-i}\frac{(2\beta_i+1)}{6}2^{i+j}\\ &=\sum_{i=0}^{n-z-1}\frac{(2\beta_i+1)}{6}\sum_{j=0}^{n-z-1-i}2^{i+j-n}\\ &=2^{-z}\sum_{i=0}^{n-z-1}\frac{(2\beta_i+1)}{6}-\sum_{i=0}^{n-z-1}\frac{(2\beta_i+1)}{6}2^{i-n}\\ &\equiv\lambda_{BW-II}(z).\end{aligned} \quad (13)$$

Hence, by (5) and (13), the generalized Type-II carry estimation of the z-PT Baugh-Wooley multiplier ((i.e., $\sigma_{BW-II}(z)$), is

$$\begin{aligned}E[\sigma_z|\beta_i]&=\left[2^{-z}\beta+\lambda_{BW-II}(z)\right]_r\\ &=\left[2^{-z}\beta+2^{-z}\sum_{i=0}^{n-z-1}\frac{2\beta_i+1}{6}-\sum_{i=0}^{n-z-1}\frac{2\beta_i+1}{6}2^{i-n}\right]_r\\ &\cong\left[2^{-z}\left(\beta+\sum_{i=0}^{n-z-1}\frac{2\beta_i+1}{6}\right)\right]_r\\ &\equiv\sigma_{BW-II}(z).\end{aligned} \quad (14)$$

Liao *et al.* [20] verified that if $z=1$, (14) coincides with the results obtained by Jou [13] and Van [14], [17].

2) z-PT Modified Booth Multiplier: For some i, $0\le i\le\lceil n/2\rceil$, and $k=n-1,n-3,\ldots,1$, the probability $P(\beta_i=y_ia_k=1)$ can be derived readily from the information obtained from Table II as

$$\begin{aligned}P(\beta_i=1)&=P(a_{k-1}=0,y_i=-2)+P(a_k=0,y_i=-1)\\ &\quad+P(a_k=1,y_i=1)+P(a_{k-1}=1,y_i=2)\\ &=\frac{1}{2}\times\frac{1}{8}+\frac{1}{2}\times\frac{1}{4}+\frac{1}{2}\times\frac{1}{4}+\frac{1}{2}\times\frac{1}{8}=\frac{3}{8},\end{aligned}$$

Hence, the following equation is obtained:

$$P(\beta_i=0)=1-P(\beta_i=1)=\frac{5}{8}.$$

Consequently, given that i with $y_i\neq 0$ and for certain $j=n,n-1,\ldots,1$ and $k=n-1,n-3,\ldots,1$ when $j\neq k$, such that $P_{i,j}=y_ia_j$ and $\beta_i=y_ia_k$, the joint probability $P(P_{i,j}=1,\beta_i=1)$ can be derived by

$$\begin{aligned}&P(P_{i,j}=1,\beta_i=1)\\ &=P(a_{j-1}=0,a_{k-1}=0,y_i=-2)\\ &\quad+P(a_j=0,a_k=0,y_i=-1)\\ &\quad+P(a_j=1,a_k=1,y_i=1)\\ &\quad+P(a_{j-1}=1,a_{k-1}=1,y_i=2)\\ &=\frac{1}{2}\times\frac{1}{2}\times\frac{1}{8}+\frac{1}{2}\times\frac{1}{2}\times\frac{1}{4}+\frac{1}{2}\times\frac{1}{2}\times\frac{1}{4}+\frac{1}{2}\times\frac{1}{2}\times\frac{1}{8}\\ &=\frac{3}{16}.\end{aligned}$$

With similar derivations, $P(P_{i,j}=1,\beta_i=0)=3/16$ is obtained. The following conditional probabilities are then obtained by

$$\begin{aligned}P(P_{i.j}=1|\beta_i=1)&=\frac{P(P_{i,j}=1,\beta_i=1)}{P(\beta_i=1)}=\frac{\frac{3}{16}}{\frac{3}{8}}=\frac{1}{2};\\ P(P_{i.j}=1|\beta_i=0)&=\frac{P(P_{i,j}=1,\beta_i=0)}{P(\beta_i=0)}=\frac{\frac{3}{16}}{\frac{5}{8}}=\frac{3}{10}.\end{aligned}$$

Thus,

$$\begin{aligned}E[P_{i.j}|\beta_i=1]&=P(P_{i.j}=1|\beta_i=1)=\frac{1}{2};\\ E[P_{i.j}|\beta_i=0]&=P(P_{i.j}=1|\beta_i=0)=\frac{3}{10}.\end{aligned}$$

Similarly, the following conditional expectations are obtained:

$$E[n_i|\beta_i=1]=\frac{1}{2}\text{ and }E[n_i|\beta_i=0]=\frac{3}{10}.$$

These mathematical expressions imply that

$$E[P_{i,j}|\beta_i] = \frac{1}{2}\beta_i + \frac{3}{10}(1-\beta_i) = E[n_i|\beta_i]. \tag{15}$$

Therefore, the generalized Type-II conditional expectation $\lambda_{Booth-II}(z)$ of the z-PT modified Booth multiplier is

$$\begin{aligned} E[2^{-n}\lambda|\beta_i] &= 2^{-z} \sum_{i=0}^{\lceil \frac{n}{2}\rceil - \lfloor \frac{z}{2}\rfloor - 1} \left(\frac{\beta_i}{2} + \frac{3(1-\beta_i)}{10}\right) \\ &= 2^{-z}\left(\frac{\beta}{5} + \frac{3}{10}\left(\left\lceil\frac{n}{2}\right\rceil - \left\lfloor\frac{z}{2}\right\rfloor\right)\right) \\ &\equiv \lambda_{Booth-II}(z). \end{aligned} \tag{16}$$

Using (5) and (16), the Type-II carry estimation $\sigma_{Booth-II}(z)$ of the z-PT modified Booth multipliers can be calculated by

$$\begin{aligned} E[\sigma_z|\beta_i] &= \left[2^{-z}\beta + \lambda_{Booth-II}(z)\right]_r \\ &= \left[2^{-z}\left(\frac{6\beta}{5} + \frac{3}{10}\left(\left\lceil\frac{n}{2}\right\rceil - \left\lfloor\frac{z}{2}\right\rfloor\right)\right)\right]_r \\ &\equiv \sigma_{Booth-II}(z). \end{aligned} \tag{17}$$

Using (16) or (13), the computation complexity for the generalized Type-II carry estimation remains high, although it is less than that of the Type-I scheme.

C. Type III—Carry Estimation Conditioning on α

1) z-PT Baugh-Wooley Multiplier: The original Type-III carry estimation in [20] employed a 2D condition of a_j and b_i, which is too complex for implementation. For simplicity, this study considers the anti-diagonal vector $\boldsymbol{P_{aj}}$ for developing the generalized Type-III carry estimation scheme. Because A and B are symmetrical and i.i.d. random variables, $\lambda_{BW-III}(z) = \lambda_{BW-II}(z)$ and $\sigma_{BW-III}(z) = \sigma_{BW-II}(z)$ are concluded.

2) z-PT Modified Booth Multiplier: Any two distinct elements (e.g., $P_{i,j}$ and α_j) within the anti-diagonal vector $\boldsymbol{P_{aj}}$, $0 \le j \le n-1$, are determined by two different encoded digits (e.g., y_i and $y_{i'}$, $i \ne i'$). Because the bits of B are i.i.d. random variables, the encoded digits y_i and $y_{i'}$ can be considered to be uncorrelated (i.e., $E[P_{i,j}|\alpha_j] = E[P_{i,j}]$). Hence, the following is obtained:

$$E[P_{i,j}|\alpha_j] = E[P_{i,j}] = P(P_{i,j}=1) = \frac{3}{8}.$$

Similarly,

$$E[n_i|\alpha_j] = E[n_i] = P(n_i = 1) = \frac{3}{8}.$$

Consequently, the generalized Type-III carry estimation for the z-PT modified Booth multiplier is similar to that for the Type-II scheme, except for the different expectation from (15). Hence, the generalized Type-III expectation $\lambda_{Booth-III}(z)$ of the z-PT modified Booth multiplier can be obtained by rewriting (16) as

$$\begin{aligned} E[2^{-n}\lambda|\alpha_j] &= 2^{-z} \sum_{i=0}^{\lceil \frac{n}{2}\rceil - \lfloor \frac{z}{2}\rfloor - 1} \frac{3}{8} \\ &= 2^{-z}\frac{3}{8}\left(\left\lceil\frac{n}{2}\right\rceil - \left\lfloor\frac{z}{2}\right\rfloor\right) \equiv \lambda_{Booth-III}(z). \end{aligned} \tag{18}$$

By (5) and (18), the generalized Type-III carry estimation $\sigma_{Booth-III}(z)$ of the z-PT modified Booth multiplier is

$$\begin{aligned} E[\sigma_z|\alpha_j] &= \left[2^{-z}\beta + \lambda_{Booth-III}(z)\right]_r \\ &= \left[2^{-z}\left(\beta + \frac{3}{8}\left(\left\lceil\frac{n}{2}\right\rceil - \left\lfloor\frac{z}{2}\right\rfloor\right)\right)\right]_r \\ &\equiv \sigma_{Booth-III}(z). \end{aligned} \tag{19}$$

Based on (18) and the given n and z, the $\lambda_{Booth-III}(z)$ is just a constant. If $z = 1$, (19) coincides with the PEB [28]; it is anticipated that the generalized PEG [30] is analogous to (19) for the general case.

Before concluding this subsection, although some carry estimation functions presented here are not new to the literature, the generalized three types of carry estimation schemes provide systematic and complete approaches to designing each type of fixed-width multipliers. The mathematical formulations unified by conditional probability theory are more readable than those presented in [28]–[30]. Moreover, the error performance of the Type-III carry estimation scheme improves gradually when z is high. This indicates that the Type-III carry estimation scheme, given a constant bias by (18), employs a simpler circuit to achieve higher performance accuracy.

III. Variable-Latency Speculating Booth Multiplier

Based on generalized conditional-probability carry estimations for the z-PT modified Booth multiplier, this study proposes an optimized 16 × 16 variable-latency speculating two-stage pipelined modified Booth multiplier (VLSBM).

A. The Proposed Architecture

Fig. 7 shows the comprehensive microarchitecture of the proposed VLSBM, which comprises the following two operating stages: 1) Stage-1 performs the speculating phase of the VLSBM and 2) Stage-2 is the error recovery. To reduce the critical path, the VLSBM partial products are partitioned independently into the $(n+z)$-bit MSP and the $(n-z)$-bit LSP. For the differentiation, the "$\mathrm{product}[2n-1:0]$" shown in Fig. 7 is employed to signify the speculation, and the "$\mathrm{true}[2n-1:0]$" represents the $2n$-bit exact multiplication of the VLSBM. Without waiting the $\mathrm{true\_}\lambda(z)$ from the $(n-z)$-bit LSP, $\Sigma_{n-z}(LSP)$ and $\Sigma_{n+z}(MSP)$ can be calculated in parallel during the speculating phase with the predicted $\lambda(z)$.

1) Error Detection and Error Recovery: The (possible) error recovery of $\mathrm{product}[2n-1:n-z-2]$ is separated by the following two parts in the proposed VLSBM: 1) the primary z-bit compensation for $\mathrm{product}[n-1:n-z-2]$ and 2) the successive n-bit compensation for $\mathrm{product}[2n-1:n]$. As shown in Fig. 7, the error correction of each part is performed in stages to reduce the critical path.

Because $\Sigma_{n-z}(LSP)$ is always true, at the end of the Stage-1 we obtain $\mathrm{true}[n-z-1:0] := \mathrm{product}[n-z-1:0]$. It is readily proven that $\mathrm{true}[n-1:n-z-2]$ can be obtained by subtracting the predicted $\lambda(z)$ from $\mathrm{true\_}\lambda(z)$, and then adding it to the $\mathrm{product}[n-1:n-z-2]$. Consequently, to verify the prediction, an adder and subsequent subtractor are applied (Fig. 7). The sign of the difference $(\mathrm{true\_}\lambda(z) - \lambda(z))$ is recorded by the

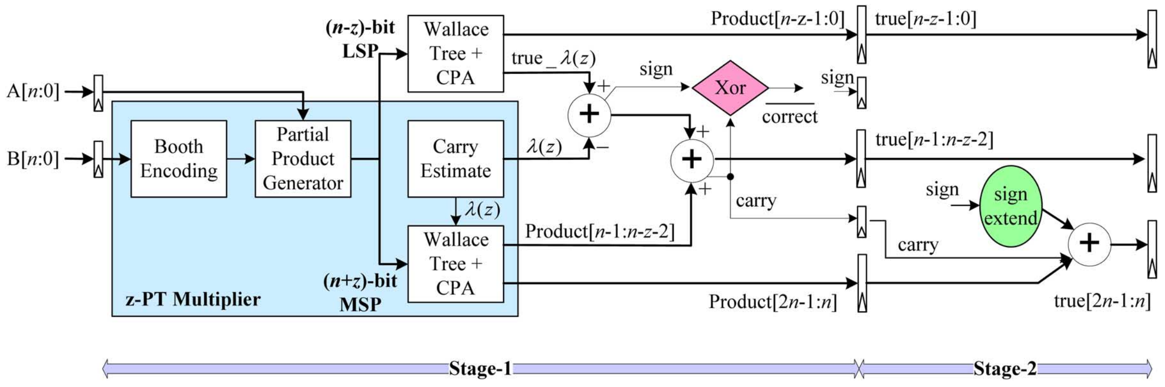


Fig. 7. The proposed speculating 2-stage pipelined modified Booth multiplier.

"sign" bit for further use. The following procedure is based on the true value of the sign bit.

(a) *If* $sign = 0$:
This means that an under-estimated (or correct) $\lambda(z)$ has been predicted. If a prediction is inaccurate, $(\text{true\_}\lambda(z) - \lambda(z))$ is the compensation that should be added to speculative $\text{product}[2n-1:n-z-2]$ for error correction. After the first z-bit compensation for $\text{product}[n-1:n-z-2]$ in the speculating phase, the carry of the error-compensation adder is recoded (Fig. 7). The following procedure is called based on the truth value of the recorded carry bit.
  (i) *If* $carry = 0$:
  This means that no remainder is required for further compensation of $\text{product}[2n-1:n]$ during the correcting phase. Hence, the "correct" bit is set to 1 and $\text{true}[2n-1:n] := \text{product}[2n-1:n]$.
  (ii) *If* $carry = 1$:
  This means that a non-zero carry must be added to $\text{product}[2n-1:n]$ during the correcting phase. Hence, the "correct" bit is set to 0.

(b) *If* $sign = 1$:
This means that, for $\lambda(z)$, an over-estimated prediction has been encountered, which leads to $(\text{true\_}\lambda(z) - \lambda(z)) < 0$. Hence, the sign-extend unit is necessary for error compensation. Similarly, the difference $(\text{true\_}\lambda(z) - \lambda(z))$ should be calculated, and then subsequently added to $\text{product}[2n-1:n-z-2]$ during the compensation stage error correction. At the end of Stage-1, the carry of the error-correction adder is recoded to supervise the following procedure.
  (i) *If* $carry = 0$:
  This means that, except for sign-extended bits, no remainder requires compensation for $\text{product}[2n-1:n]$ during the correcting phase. Hence, the "correct" bit should be set to 0.
  (ii) *If* $carry = 1$:
  This means that, except for sign-extended bits, a non-zero carry must be added to $\text{product}[2n-1:n]$ during the correcting phase. Adding the all-ones vector and the non-zero carry-bit obtains the all-zeros vector. Therefore, the "correct" bit should be set to 1 because $\text{true}[2n-1:n]$ is identical to $\text{product}[2n-1:n]$.

Consequently, based on this discussion, the correct (or valid) bit is the reverse of the logical disjunction (or exclusive) of the two-condition bits: the sign bit and the carry bit, which indicates that the speculation is valid (i.e., $\text{correct} = 1$) if the two-condition bits have identical truth values.

B. Optimized VLSBM by Design Space Exploration

A crucial issue during the early stage of system design is the selection of the appropriate parameters among the possible design spaces or alternatives. The design alternatives typically involve multiple metrics of interest, such as speed, area, and power (or energy). To provide an example, the 16 × 16 VLSBM is discussed in this subsection.

As shown in Fig. 7, the delay in Stage-1 (i.e., T_z) dominates the VLSBM critical path, which ends at "$\text{product}[2n-1:0]$." To achieve the highest throughput of the proposed VLSBM, this study explores the cost function by (2). Theoretically, the accuracy probability P_A(z) of the 16 × 16 VLSBM can be evaluated using

$$P_A(z) = \frac{\text{number of inputs with correct speculation}}{2^{16} \times 2^{16}\ \text{inputs}}. \quad (20)$$

Table III shows the simulation results for $T_z, P_A(z)$, and the estimated cost of the proposed VLSBM with certain z and different carry estimation schemes. Note that T_z in Table III is denoted by the tightest timing constraint (or the minimum clock period) of the VLSBM, as simulated by Synopsis Design Compiler (@ UMC 90 nm CMOS cell library).

Based on parallel prefix computation, previous studies have frequently applied the CLA to the final adder of the multiplier to obtain improve system speed. Although the CLA is the fastest, it requires a large area overhead. Because the multi-level Wallace tree compression causes the input signals to arrive separately, a full adder of partitioned carry select adders [2] and a full adder comprising a conditional sum adder (CSMA) and conditional carry adder (CCA) [3], [4] were proposed to reduce the surface area. Therefore, the area-efficient CPAs presented in [2]–[4] were implemented. However, their speeds are slightly lower than those obtained by the CLA (DesignWare). To provide a unified and fair evaluation of T_z for design space exploration, this study forces Synopsis Design Compiler to apply

TABLE III
EXPLORATION RESULTS OF THE 16 × 16 VLSBM WITH DIFFERENT z @ UMC 90 nm CMOS CELL LIBRARY

z	*Carry-Est.*	$P_A(z)$	T_z^* (ns)	Cost
2	type-I (cla)	86.4%	1.59	1.806
	type-III (cla)	83.5%	1.51	1.760
3	type-I (cla)	92.9%	1.59	1.703
	type-III (cla)	91.5%	1.50	1.628
4	type-I (cla)	96.8%	1.59	1.641
	type-III (cla)	**96.0%**	**1.50**	**1.560**
5	type-I (cla)	98.3%	1.61	1.637
	type-III (cla)	**97.9%**	**1.53**	**1.562**
6	type-I (cla)	99.3%	1.68	1.691
	type-III (cla)	99.0%	1.62	1.636

*: T_z is the tightest timing constraint

TABLE IV
VLSI IMPLEMENTATION RESULTS OF DIFFERENT 16 × 16 BOOTH MULTIPLIER @ UMC 90 nm CMOS CELL LIBRARY

Design	cycle (ns)	Area (μm^2)	P_A	P_D	T_E (ns)	Power (mW)	Energy (pJ)
Direct Implementation	1.66	14712			1.66	6.06	10.06
VLSA CPA [8]	1.62	18051	98.8%	100%	1.64	7.66	12.56
				0%	1.62	7.73	12.52
Cho [16] (or [29]) with z=4	1.60	15967	95.4%	100%	1.67	6.66	11.12
				0%	1.60	6.89	11.02
type-I (or [31]) with z=4	1.59	16288	96.8%	100%	1.64	7.08	11.61
				0%	1.59	7.25	11.53
Type-III (or [30]) with z=4	**1.50**	**15652**	**96.0%**	**100%**	**1.56**	**6.90**	**10.76**
				0%	**1.50**	**7.11**	**10.65**
Conventional 2-stage pipeline	1.15	16780	--	100%	2.30	6.27	14.42
				0%	1.15	10.45	12.02

the fastest DesignWare binary adders, i.e., the *Carry look-ahead synthesis model* (cla), for the VLSBM CPA, which is a similar approach to that applied in [8].

As shown in Table III, for a given z, the simulated T_z of the VLSBM with the Type-I scheme is greater than that of the Type-III scheme; however, the accuracy performance $P_A(\mathrm{z})$ of the Type-I scheme is superior to that of the Type-III scheme. By examining the associated cost shown in Table III, the proposed 16 × 16 VLSBM may achieve the optimized performance by applying the Type-III carry estimation scheme with $z = 4$ or $\mathrm{z} = 5$. Because of the smaller T_z (i.e., 1.50 versus 1.53 ns), the case with $z = 4$ is preferable.

C. Implementation Results

The silicon implementation results of numerous reliable 16 × 16 two-stage pipelined modified Booth multipliers are summarized and compared with the direct implementation of Booth PT multiplier, as shown in Table IV. To perform a fair comparison, the synthesis conditions considered in each multiplier in Table IV are identical.

By direct implementation, the simulated minimum period (i.e., tightest timing constraint) and the area of the 16 × 16 Booth PT multiplier are approximately 1.66 ns and 14,712 $\mu\mathrm{m}^2$, respectively. Based on an evaluation of 100,000 random multiplications, the average energy per multiplication is approximately 10.06 pJ.

Inserting pipeline registers to design the fastest, well-balanced two-stage pipelined modified Booth multiplier (shown in the last row of Table IV), the minimum period is reduced to 1.15 ns, which is approximately 1.44 times faster than that of the original one if no data-dependent hazard occurs, (i.e., if $P_D = 0$). Nevertheless, an increase of up to 14.1% and 19.5% occurs for area overhead and of energy dissipation, respectively. Performance degradation occurs if $P_D \neq 0$. Because the latency of a multiplication is two cycles, the multiplier instructions within the DSP algorithm may cause a significant portion of stall cycles by data dependencies. Let T_p denote the clock period of the fastest two-stage pipelined modified Booth multiplier. For a given DSP application with multiplier-stall dependence probability P_D, the effective cycle time T_E of the fastest two-stage pipelined modified Booth multiplier is

$$T_E = (1 - P_D) \times T_p + P_D \times 2T_p = T_p + P_D T_p. \tag{21}$$

Compared to the original non-pipelined Booth multiplier, if $P_D > 0.44$, then $T_E \geq 1.66$ ns; that is, the throughput benefit of the two-stage pipelined multiplier is overcome definitely. Hence, to minimize P_D to improve performance, data-intensive computing processors require a significant amount of time-consuming and error-prone hand-optimized assembly language programming efforts. Moreover, as shown in Table IV, the additional energy dissipation is wasted (i.e., 14.42 pJ versus 12.02 pJ) because of the pipeline hazards for the case of high P_D.

In contrast, if the pipelined multiplier is capable of result-speculation, the data dependence might be hidden, thereby the stall cycles due to data dependence are saved. Suppose that the cycle time of the proposed VLSBM is T_s, where T_s differs from T_p and T_s is generally greater than T_p. When data dependence occurs and the speculation result is true, the VLSBM requires only one cycle (i.e., T_s) to complete the multiplication. Conversely, an incorrect speculation invokes the correcting phase to obtain the exact multiplication; hence, $2T_s$ is necessary. Table V details the latencies required to complete a reliable multiplication for the VLSBM. Given by P_D and P_A, the effective performance T_E of the VLSBM then becomes

$$\begin{aligned} T_E &= P_A T_s + P_D(1 - P_A) \times 2T_s + (1 - P_D)(1 - P_A) \times T_s \\ &= T_s + P_D T_s - P_A P_D T_s. \end{aligned} \tag{22}$$

If P_A approaches 1, then the most effective performance of the VLSBM is clearly achieved (i.e., $T_E \cong T_s$), regardless of the application. Consequently, the potential speed-up benefit of the pipelined data path is ensured. This assists software porting on

TABLE V
LATENCY FOR COMPLETING THE MULTIPLICATION FOR VLSBM

Case	Accurate carry prediction	Inaccurate carry prediction
With data-dependence	T_s	$2T_s$
Without data-dependence	T_s	T_s

data-intensive computing processors because no assembly language programming is necessary during compiling.

Under identical synthesis conditions, four reliable 16 × 16 VLSBMs are tested (Table IV) to conduct a fair comparison. The first VLSBM directly applies the VLSA proposed in [8] to design the CPA of the modified Booth multiplier. With the additional circuits for error detection and error recovery, the simulated minimum period of the VLSA-based VLSBM is 1.62 ns, which is slightly faster than that of the original modified Booth PT multiplier by direct implementation. Further investigation using the mentioned 100,000 random multiplications, the P_A of the VLSA-based VLSBM is approximately 98.8%, which is the highest accuracy. However, the silicon area increases by approximately 22.7%. Consequently, directly applying the VLSA to design the CPA of the modified Booth multiplier appears impractical.

The remaining reliable 16 × 16 VLSBMs shown in Table IV apply the proposed architecture shown in Fig. 7, although they employ the following carry estimation schemes: 1) the (Type-I-like) Cho's result [16]; 2) the Type-I scheme (or APCE [31]); and 3) the Type-III scheme (or GPEB [30]). Cho *et al.* [16] conducted exhaustive simulations and proposed the accurate carry estimation function (σ_{Cho}) as follows:

$$\sigma_{Cho} = \left\lfloor \frac{1}{2}\left(\beta + \left[\frac{1}{2}\sum_{i=0}^{\lceil \frac{n}{2}\rceil -2} y_i''\right]_r\right)\right\rfloor, \tag{23}$$

which differs slightly from (10). Hence, P_A and the corresponding T_z are different from those of the Type-I carry estimation scheme. The implementation results shown in Table IV verify the effectiveness of the proposed optimized 16 × 16 VLSBM by using the Type-III carry estimation scheme with $z = 4$, which employs the minimum area overhead to achieve the highest effective throughput, as well as dissipating the least energy per multiplication. Unlike the conventional pipelined Booth multiplier, the effective throughput of the optimized VLSBM always outperforms the non-pipelined Booth multiplier, regardless of the P_D value. The speed-up ratio, area overhead, and energy consumption are increased by 6.4% to 10.6%, 6.4%, and 5.9% to 7.0%, respectively. Compared to the fastest pipelined Booth multiplier, the proposed VLSBM is area-/energy-efficient, and reduces the area overhead and energy dissipation by approximately 7% and 11.4% to 25.4%, respectively. Furthermore, if $P_D > 0.32$, the throughput of the optimized VLSBM always outperforms the conventional pipelined Booth multiplier. As anticipated, the optimized VLSBM alleviates the performance degradation for the pipelined data path caused by high P_D values.

TABLE VI
IMPLEMENTATION RESULT COMPARISON OF THE WITH/WITHOUT AN EXTRA MULTIPLEXER IN THE STAGE-2 OF THE PROPOSED VLSBM

Proposed VLSBM (type-III with z=4)	Cycle (ns)	Area (μm^2)	P_D	Energy (pJ)	% of energy in stage-2
Without Mux/control	**1.50**	**15652**	**0%**	**10.65**	**2.7%**
With Mux/control	1.50	16156	0%	11.04	1.6%

Before concluding this subsection, by not equipping a multiplexer to select the correct result from the multiplier, Stage-2 of the proposed VLSBM (Fig. 7) is performed continually, regardless of the speculation result, which might consume more (dynamic) power. However, an additional multiplexer and control mechanism requires additional area overhead, which causes more power leakage. Table VI shows the implementation results of the proposed VLSBM with and without an additional Stage-2 multiplexer when $P_D = 0\%$. If the multiplexer is used, the energy consumed by the simple addition for Stage-2 error recovery can be conserved (i.e., 1.6% versus 2.7%). However, the total energy and the area of the VLSBM increase when the multiplexer is equipped. Consequently, we assert that the design of the proposed VLSBM without an additional Stage-2 multiplexer is preferable (Fig. 7). The silicon area of the proposed optimized VLSBM can be reduced further by applying a full adder combining both CSMA and CCA [3], [4]. The area of the modified VLSBM is decreased to 14 458 $\mu\mathrm{m}^2$; however, the estimated minimum period is increased slightly to 1.51 ns.

D. Experimental Results for Real Workloads

In this subsection, three real multimedia applications (JPEG compression, the human face object detection, and H.264/AVC decoding) are examined to verify the multiplicative performance of the proposed VLSBM integrated by classic five-stage pipelined 32-bit MIPS processors.

The open-source C-language reference codes of the applications tested in this study are available on the Internet (see Independent JPEG Group (IJG) [33], Open Source Computer Vision [34], and H.264/AVC JM Reference Software [35]). The open-source programs were compiled using a GNU-based C compiler for MIPS processors with an optimization option of "-o1," which attempts to reduces the code size without performing additional optimizations. To emphasize the effectiveness of the proposed VLSBM with the discussed applications, an ad-hoc framework is developed for the dynamic profiling of the software applications to trace the actual operand values driven to the processor multiplier, and to check the RAW (read-after-write) hazard of each multiplier instruction when produced during execution.

For the application of a 512 × 512 JPEG image compression [33], 2,064,384 multiplications were recorded with a P_D of approximately 0.76. Two standard test images (*Lena* and *Baboon*) were used to test the performance of the proposed VLSBM. The results show that the P_A values are approximately 93.9% and 94.1%, respectively. Furthermore, the required cycle count C_s values of the proposed VLSBM-enhanced MIPS for these images are 2,162,201 and 2,157,110, respectively. Compared

TABLE VII
MULTIPLICATIVE PERFORMANCE EVALUATIONS FOR REAL APPLICATIONS

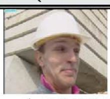
512×512 JPEG Image Compression

	# of mul.: 2064384 P_D=0.76		# of mul.: 2064384 P_D=0.76		
Design	P_A	cycle count	P_A	cycle count	Speedup
Conventional 2-stage pipeline	--	3637248	--	3637248	1×
Proposed VLSBM	**93.9%**	**2162201**	**94.1%**	**2157110**	**1.3×**

5-Photo Human Face Object Detection

	Photo 1 # of mul.: 16246820 P_D=0.36		*Photo 2* # of mul.: 17323768 P_D=0.36		
Design	P_A	cycle count	P_A	cycle count	Speedup
Conventional 2-stage pipeline	--	22075858	--	23543158	
Proposed VLSBM	**89%**	**17035379**	**89%**	**18145946**	
	Photo 3 # of mul.: 20521624 P_D=0.36		*Photo 4* # of mul.: 19835909 P_D=0.36		
Conventional 2-stage pipeline	--	27896398	--	26965190	
Proposed VLSBM	**90%**	**21196649**	**90%**	**20784426**	
	Photo 5 # of mul.: 19723580 P_D=0.36				
Conventional 2-stage pipeline	--	26816490			1×
Proposed VLSBM	**90%**	**20274945**			**1.0×**

H.264/AVC Decoder (Baseline Profile)

	 # of mul.: 106716355 P_D=0.85		
Design	P_A	cycle count	Speedup
Conventional 2-stage pipeline	--	197678607	1×
Proposed VLSM	**95%**	**111953450**	**1.4×**

to the conventional 2-stage pipelined Booth multiplier, the ratio of the average cycle count reduction is approximately 1.7. Table VII (top) shows the experimental results for the 512 × 512 JPEG image compression application. Compared with the fastest pipelined Booth multiplier with $T_p = 1.15$ ns, the multiplicative speed-up ratio of the proposed VLSBM for the 512 × 512 JPEG compression can be estimated by

$$Speedup = \frac{\overline{C_p} \times T_P}{\overline{C_s} \times T_s} = \frac{3637248 \times 1.15}{2159656 \times 1.5} \approx 1.3. \qquad (24)$$

where $\overline{C_p}$ and $\overline{C_s}$ are the average cycle counts required for the conventional pipelined Booth multiplier and for the proposed VLSBM, respectively.

Fig. 8 shows that five typical 320 × 240 photos are applied for the facial object detection application examination (OpenCV v2.0.0) [34]. Depending on the selected photo image, although

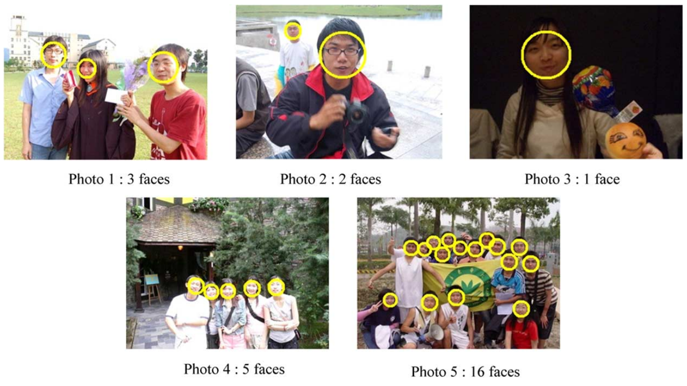


Fig. 8. Five selected photos for object detection application.

the number of multiplications for facial object detection is not constant, the estimated P_D for each image test is approximately 0.36, as shown in Table VII (center), and the simulated P_A of the proposed VLSBM is approximately 89%–90%. Compared to the conventional 2-stage pipelined Booth multiplier, the average cycle count reduction ratio is approximately 1.3. Similar to (24), compared to the fastest conventional two-stage pipelined architecture, the multiplicative speed-up ratio of the proposed optimized VLSBM for object detection is obtained by

$$Speedup = \frac{\overline{C_p T_P}}{\overline{C_s} T_s} = \frac{25457619 \times 1.15}{19487469 \times 1.5} \approx 1.$$

Although the effective throughput is almost identical, approximately 16.5% of the total energy is conserved by employing the proposed optimized VLSBM.

For the high-definition H.264/AVC decoder application [35], the CIF *Foreman* sequence was examined. The H.264/AVC decoder displays CIF I- and P-frames, using all inter- and intra-prediction modes. The motion compensation employs a single reference that applies all block sizes, with a search range of ±16 pixels $(\text{accuracy} = 1/4 - \text{pixel})$. In the study of 106,716,335 multiplications, a P_D value was obtained of approximately 0.85. The experimental results show that the P_A of the *Foreman* sequence is approximately 95%. Compared to the conventional 2-stage pipelined Booth multiplier, the average cycle count reduction ratio is approximately 1.8. Similar to (25), compared with the fastest conventional two-stage pipelined architecture, the multiplicative speed-up ratio of the optimized VLSBM for the high definition H.264/AVC decoder application is approximately

$$Speedup = \frac{C_p T_P}{C_s T_s} = \frac{197678607 \times 1.15}{111953450 \times 1.5} \approx 1.4.$$

Thus, the experimental results verify the effectiveness of the proposed VLSBM.

IV. CONCLUSION

Data hazards cause severe performance degradation in the pipelined datapath of data-intensive computing processes. Under a pessimistic assumption on the pipeline efficiency, this study implemented an optimized VLSBM. Without waiting for the true carry propagation, the VLSBM applies a conditional-probability carry estimator for the carry signal. Similar to the branch predictor used in microprocessors, if a true carry prediction occurs, the multiplier-stall cycles caused by data dependence can be hidden. Examined by the instruction streams in the real workload: JPEG compression, object detection, and H.264/AVC decoder, experimental results reveal that the proposed VLSBM obtains a speculation accuracy between approximately 89% to 95%, thereby achieving a cycle count ratio reduction of approximately 1.3 to 1.8×. Thus, the potential speed-up benefit of the pipelined data path is ensured. The proposed VLSBM is useful for software porting on data-intensive computing processors because no time-consuming efforts for assembly language programming are necessary when compiling. If $P_D \geq 0.32$, the speed of the proposed VLSBM is greater than that of the fastest conventional well-pipelined modified Booth multiplier (@ UMC 90 nm CMOS). Moreover, area overhead and energy dissipation are reduced by approximately 7% and 11.4% to 25.4%, respectively. The contributions of this study are detailed as follows: 1) the first high-speed and area-/energy-efficient architecture is proposed for the variable-latency speculating modified Booth multiplier, which hides approximately 90% of the multiplier-stall cycles, thereby ensuring the pipelined data path of data-intensive computing processors; 2) based on conditional probability theory, a complete and systematic framework of three carry estimation schemes for z-PT fixed-width multipliers (for the modified Booth multiplier and the Baugh-Wooley multiplier) have been developed; and (3) thorough design-space considerations are provided for constructing the optimized VLSI architecture of the proposed 16 × 16 VLSBM. Numerous simulations were performed and experimental results obtained that verify the overall effectiveness of the proposed VLSBM.

The future direction of this research will consider additional pipeline stages. Devising an efficient control and error recovery mechanism for compensating for inaccurate predictions without adversely impacting the overall performance are the most critical challenges [7].

REFERENCES

[1] K. K. Parhi, *VLSI Digital Signal Processing Systems*. New York, NY, USA: Wiley, 1999.

[2] J. Fadavi-Ardekani, " $M \times N$ Booth encoded multiplier generator using optimized Wallace trees," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 2, pp. 120–125, Jun. 1993.

[3] W.-C. Yeh and C.-W. Jen, "High-speed Booth encoded parallel multiplier design," *IEEE Trans. Computers*, vol. 49, no. 7, pp. 692–701, Jul. 2000.

[4] Q. Li, G. Liang, and A. Bermak, "A high-speed 32-bit signed/unsigned pipelined multiplier," in *IEEE Int. Symp. Electronic Design, Test & Applications, DELTA*, 2010, pp. 207–211.

[5] R. K. Yu and G. B. Zyner, "167 MHz radix-4 floating point multiplier," in *Proc. 12th Symp. Computer Arithmetic*, 1995, pp. 149–154.

[6] G. Even and P.-M. Seidel, "A comparison of three rounding algorithms for IEEE floating-point multiplication," *IEEE Trans. Computers*, vol. 49, no. 7, pp. 638–650, Jul. 2000.

[7] A. A. Del Barrio, S. O. Memik, M. C. Molina, J. M. Mendias, and R. Hermida, "A distributed controller for managing speculative functional units in high level synthesis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 3, pp. 350–363, March 2011.

[8] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Des., Automat. Test Eur.*, 2008, pp. 1250–1255.

[9] A. Cilardo, "A new speculative addition architecture suitable for two's complement operation," in *Proc. Des., Automat. Test Eur.*, 2009, pp. 664–669.

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

[11] M. J. Shulte and E. E. Swartzlander, Jr., "Truncated multiplication with correction constant," *VLSI Signal Processing*, vol. VI, pp. 388–396, 1993.

[12] S. S. Kidambi, F. El-Guibaly, and A. Antonious, "Area-efficient multipliers for digital signal processing applications," *IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing*, vol. 43, no. 2, pp. 90–95, Feb. 1996.

[13] J.-M. Jou, S.-R. Kuang, and R.-D. Chen, "Design of low-error fixed-width multipliers for DSP applications," *IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing*, vol. 46, no. 6, pp. 836–842, Jun. 1999.

[14] L.-D. Van, S.-S. Wang, and W.-S. Feng, "Design of the lower error fixed-width multiplier and its application," *IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing*, vol. 47, no. 10, pp. 1112–1118, Oct. 2000.

[15] S.-J. Jou, M.-H. Tsai, and Y.-L. Tsao, "Low-error reduced-width Booth multipliers for DSP applications," *IEEE Trans. Circuits Syst. I*, vol. 50, no. 11, pp. 1470–1474, Nov. 2003.

[16] K. J. Cho, K. C. Lee, J. G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified Booth multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 5, pp. 522–531, May 2004.

[17] L.-D. Van and C.-C. Yang, "Generalized low-error area-efficient fixed-width multiplies," *IEEE Trans. Circuits Systems I: Regular Papers*, vol. 52, no. 8, pp. 1608–1619, Aug. 2005.

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

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

[20] Y.-C. Liao, H.-C. Chang, and C.-W. Liu, "Carry estimation for two's complement fixed-width multipliers," in *Workshop on Signal Processing Systems (SiPS)*, Banff, Canada, Oct. 2006, pp. 345–350.

[21] S.-R. Kuang and J.-P. Wang, "Low-error configurable truncated multipliers for multiply-accumulate applications," *Electron. Lett.*, vol. 42, no. 16, pp. 904–905, Aug. 3, 2006.

[22] H.-A. Huang, Y.-C. Liao, and H.-C. Chang, "A self-compensation fixedwidth Booth multiplier and its 128-point FFT applications," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, pp. 3538–3541.

[23] M. A. Song, L. D. Van, and S. Y. Kuo, "Adaptive low-error fixed-width Booth multipliers," *IEICE Trans. Fundam.*, vol. E90-A, no. 6, pp. 1180–1187, Jun. 2007.

[24] J.-H. Tu and L.-D. Van, "Power-efficient pipelined reconfigurable fixed-width Baugh-Wooley multipliers," *IEEE Trans. Comput.*, vol. 58, no. 10, pp. 1346–1355, Oct. 2009.

[25] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 6, pp. 1312–1325, Jun. 2010.

[26] C.-H. Chang and R. K. Satzoda, "A low error and high performance multiplexer-based truncated multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 12, pp. 1767–1771, Dec. 2010.

[27] N. Petra, D. D. Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Design of fixed-width multipliers with linear compensation function," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 5, pp. 947–960, May 2011.

[28] C.-Y. Li, Y.-H. Chen, T.-Y. Chang, and J.-N. Chen, "A probabilistic estimation bias circuit for fixed-width Booth multiplier and its DCT applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 4, pp. 215–219, Apr. 2011.

[29] J.-P. Wang, S.-R. Kuang, and S.-C. Liang, "High-accuracy fixed-width modified Booth multipliers for lossy applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 1, pp. 52–60, Jan. 2011.

[30] Y.-H. Chen, T.-Y. Chang, and C.-Y. Li, "Area-effective and power-efficient fixed-width Booth multipliers using generalized probabilistic estimation bias," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 1, no. 3, pp. 277–288, Sep. 2011.

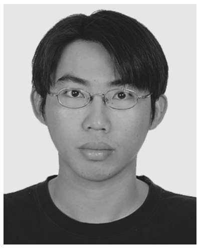
[31] Y.-H. Chen and T.-Y. Chang, "A high-accuracy adaptive conditional-probability estimator for fixed-width Booth multipliers," *IEEE Trans. Circuits Syst. I*, vol. 59, no. 3, pp. 594–603, March 2012.

[32] S.-R. Kuang, J.-P. Wang, and C.-Y. Guo, "Modified Booth multipliers with a regular partial product array," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 5, pp. 404–408, May 2009.

[33] Independent JPEG Group [Online]. Available: http://www.ijg.org/

[34] Open Source Computer Vision v 2.0.0 [Online]. Available: http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.0/OpenCV-2.0.0a-win32.exe/download

[35] H.264/AVC JM Reference Software [Online]. Available: http://iphome.hhi.de/suehring/tml/

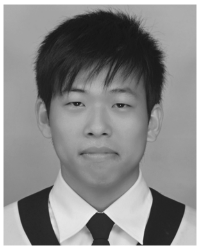


Shin-Kai Chen was born in Taiwan. He received the B.S. degree in electronics engineering from National Chiao Tung University, Taiwan, in 2004, where he is working toward the Ph.D. degree.

His research includes processors for embedded computing system, digital signal processing, and system software designs.

Chih-Wei Liu (M'03) was born in Taiwan. He received the B.S. and Ph.D. degrees, both in electrical engineering, from National Tsing Hua University, Hsinchu, Taiwan, in 1991 and 1999, respectively.

From 1999 to 2000, he was an integrated circuits design engineer at the Electronics Research and Service Organization (ERSO) of Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan. Then, near the end of 2000, he joined SoC Technology Center (STC) of ITRI as a project leader and eventually left ITRI at the end of Sepember 2003. He is currently with the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, as an Associate Professor. His current research interests are SoC and VLSI system design, processors for embedded computing system, digital signal processing, digital communications, and coding theory.



Tsung-Yi Wu was born in Taiwan. He received the B.S. degree in electrical engineering from Chang Gung University, Taoyuan, Taiwan, in 2010. He is currently pursuing the M.S. degree in the Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan.

His research interests include low-power design, digital signal processing, and computer architecture.

An-Chi Tsai was born in Taiwan. She received the B.S. degree in electrical engineering from National Dong Hwa University, Hualien, Taiwan, in 2007, and the M.S. degree in electronics engineering from National Chiao Tung University, Taiwan, in 2012.

She is currently a senior engineer at the ITE Tech. Inc., Hsinchu, Taiwan. Her research interests include SoC and VLSI system design, IC design, and digital signal processing.