A Low Error and High Performance Multiplexer-Based Truncated Multiplier

Chip-Hong Chang and Ravi Kumar Satzoda

***Abstract*—This paper proposes a novel adaptive pseudo-carry compensation truncation (PCT) scheme, which is derived for the multiplexer based array multiplier. The proposed method yields low average error among existing truncation methods. The new PCT based truncated array multiplier outperforms other existing truncated array multipliers by as much as 25% in terms of silicon area and delay, and consumes about 40% less dynamic power than the full-width multiplier for 32-bit operation. The proposed truncation scheme is applied to an image compression algorithm. Due to its low truncation error, the mean square errors (MSE) of various reconstructed images are found to be comparable to those obtained with full-precision multiplication.**

***Index Terms*—Computer arithmetic, digital multiplier, truncated multiplier, truncation scheme, VLSI design.**

I. Introduction

Multiplication is a fundamental arithmetic operation used pervasively in digital signal processing (DSP) applications like filtering, convolution, and compression [1]. From VLSI perspective, since a full-width digital multiplier receives two single precision operands to produce a double precision output, it greatly benefits from truncation for applications with a limited-precision datapath. Due to the noteworthy hardware reduction, the dynamic power dissipation can also be proportionally reduced without having to resort to sophisticated power reduction techniques.

Several truncation schemes have been proposed for fast digital multipliers [2]–[10]. The schemes proposed in [3] and [4] ignore the least significant n columns in the partial product (pp) bit matrix obtained from an $n \times n$-bit multiplication. A small amount of additional hardware is added to compensate for the truncation errors. These methods generally produce high error for the truncated product. The second category of truncation schemes preserves the hardware used to compute k additional pp bits beyond the ulp of the truncated product [5]–[9]. In [5], Schulte *et al.* introduces a constant-correction truncation (CCT) scheme where a non-zero dc component is added based on specific values of n and k, to Columns $(n-1)$ to $(n-k)$ of the pp matrix. In [6], a data-dependent variable correction truncation scheme (VCT) is proposed where the most significant pp bits from the $(n-k-1)^{\text{th}}$ column are stacked over the $(n-k)^{\text{th}}$ column and a constant bias of 1 is added in Columns $(n-2)$ to $(n-k)$ [7].

In this paper, we propose a novel truncation scheme for the multiplexer based array multiplier [11]. A brief antecedent of this work was presented in [10]. The proposed method achieves lower average and spread of errors by means of an adaptive pseudo carry compensation and a simple deterministic constant bias that is independent of k. By exploiting the symmetry of the multiplexer-based array multiplier, the pp bits generated by the multiplexers in our truncated multiplier can be accumulated in a carry-save format to further reduce the area and improve the speed over other truncated array multipliers.

Manuscript received December 23, 2008; revised April 24, 2009. First published September 22, 2009; current version published November 24, 2010.

The authors are with the Nanyang Technological University, School of Electrical and Electronic Engineering, 639798 Singapore (e-mail: echchang@ntu.edu.sg; rksatzoda@ntu.edu.sg).

Digital Object Identifier 10.1109/TVLSI.2009.2027327

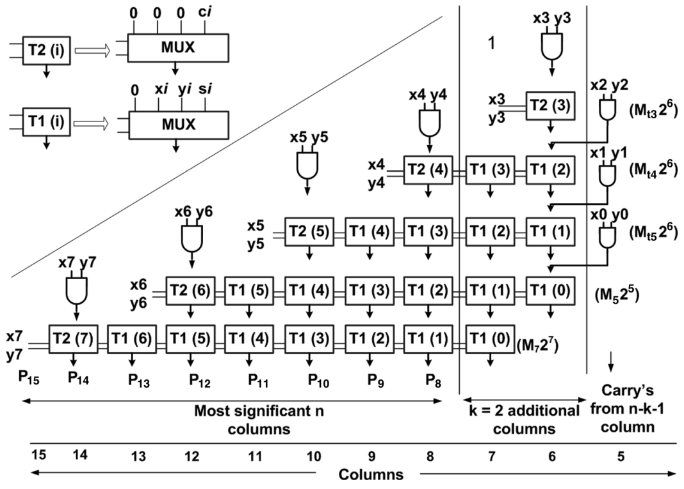

Fig. 1. Truncated multiplexer matrix for an 8 × 8-bit multiplier; $\boldsymbol{n(=8)}$ most significant columns and $\boldsymbol{k(=2)}$ additional columns of multiplexers are indicated; multiplexers in the $\boldsymbol{(n-k-1)^{\text{th}}\ \text{column}=5}$ are replaced with $\textbf{AND}$ cells corresponding to the carry signals.

II. New Truncated Multiplication Scheme

The product of two n-bit positive integers $X = x_{n-1}x_{n-2}\ldots x_1x_0$ and $Y = y_{n-1}y_{n-2}\ldots y_1y_0$ is a $2n$-bit product $P = XY$. In [7], the numbers are assumed to be fractional in their error analysis and the inputs and output are scaled by a factor of 2^{-n} and 2^{-2n}, respectively. For multiplexer-based array multiplier [11], two new variables $X_{n-1} = x_{n-2}x_{n-3}\cdots x_0$ and $Y_{n-1} = y_{n-2}y_{n-3}\cdots y_0$ are defined such that

$$P = \{X_{n-1} + 2^{n-1}x_{n-1}\}\{Y_{n-1} + 2^{n-1}y_{n-1}\} = \sum_{i=0}^{n-1}(x_i \cdot y_i)2^{2i} + \sum_{i=1}^{n-1} M_i 2^i \quad (1)$$

where $M_i = x_i \cdot Y_i + X_i \cdot y_i$. M_i can be implemented as a multiplexer with x_i and y_i as select signals. $M_i = 0$, X_i, Y_i and $X_i + Y_i$ when x_iy_i = "00", "01", "10" and "11", respectively. (1) can be realized by a multiplexer array shown in [11, Fig. 3]. Without loss of generality, our proposed truncated scheme can be explained with the help of a truncated multiplexer matrix of Fig. 1 with $n = 8$ and $k = 2$, where k is the number of partial product (pp) columns to be kept beyond the width, n of the truncated product. Using VCT scheme, the multiplexers from Column 5 (i.e., $n-k-1$) will be stacked on Column 6 (i.e., $n-k$). However, the product bit in Column 6 is dependent on the carry's generated from Column 5 into Column 6 rather than the sum of its pp bits. Directly stacking the pp bits of Column 5 onto Column 6 can also create excessive error due to the carry propagated into Column 7 (i.e., $n-k+1$).

From the definition of M_i, if both x_i and y_i are 1's, then the sum of X_i and Y_i is selected. If the select signals of all the multiplexers in Column 5 are 1's, then the sums of the multiplexer inputs, i.e., s_0, s_1, and s_2, are selected. In the worst case, the carry signals, $c_0 = x_0 \cdot y_0$, $c_1 = x_1 \cdot y_1$, and $c_2 = x_2 \cdot y_2$ are obtained. Our idea is to add the `carry` signals generated from the inputs to the multiplexers in the $(n-k-1)^{\text{th}}$ column to the $(n-k)^{\text{th}}$ column. The error is reduced as only the necessary `carry's` from the $(n-k-1)^{\text{th}}$ column are added to the $(n-k)^{\text{th}}$ column. The error is still present, though, because of the assumption that all select signals of the multiplexers in the $(n-k-1)^{\text{th}}$ are "1". This approach is, however, closer to the carry propagation in full-width multiplier as opposed to VCT.

To minimize the truncation error for an unsigned integer multiplication, a new pseudo-carry compensated truncation (PCT) scheme consisting of an adaptive compensation circuit and a fixed bias is proposed. Hardware reduction is achieved by retaining only the $n+k$ most significant columns of the multiplexer matrix. The reduction error is adaptively compensated by adding the `carry's` that are generated from the summation of the multiplexer inputs from the $(n-k-1)^{\text{th}}$ column, i.e., $c_i = x_i \cdot y_i$ for $i \in \{0, 1, \ldots, \lceil (n-k)/2 \rceil - 1\}$ to the $(n-k)^{\text{th}}$ column. The rounding error is compensated by adding a fixed bias of 0.5 LSB, irrespective of the values of n and k. This rounding constant can be added by simply inserting a "one" in the $(n-1)^{\text{th}}$ column.

Finally, the n-bit truncated product, P_t, with its least significant bit weighted 2^0, is given by

$$P_t = \sum_{i=r}^{n-1} M_i 2^{i-n} + \sum_{i=\lfloor \frac{r+1}{2} \rfloor}^{r-1} M_{ti} 2^{r-n} + \sum_{i=\lfloor \frac{r+1}{2} \rfloor}^{n-1} (x_i \cdot y_i) 2^{2i-n} + \sum_{i=0}^{r-1} (x_i \cdot y_i) 2^{r-n} + 2^{-1} \tag{2}$$

where $r = n-k$, $M_{ti} = x_i \cdot Y_{ti} + X_{ti} \cdot y_i$, $Y_{ti} = y_{i-1} y_{i-2} \cdots y_{r-i}$, and $X_{ti} = x_{i-1} x_{i-2} \cdots x_{r-i}$. M_{ti} can be considered as a truncated expression for M_i with truncated X_{ti} and Y_{ti}. In Fig. 1, M_{t3}, M_{t4}, and M_{t5} are the truncated multiplexer rows corresponding to M_3, M_4, and M_5. It should be noted that P_{ti} of P_t corresponds to P_{n+i} of the full-width multiplier's product, P.

In order to derive the average error of the proposed PCT scheme, each input bit x_i and y_i is assumed to have equal probability. The probability that the output of a 2-input AND gate is a "one" is $P_a(x_i, y_i) = P_{x_i} P_{y_i} = 0.25$. For a 4-to-1 multiplexer, however, the output, f dependent on the select signals, s_1 and s_2 and the inputs, i_1, i_2, i_3, and i_4 is given by

$$f = \overline{s_1} \cdot \overline{s_2} \cdot i_1 + \overline{s_1} \cdot s_2 \cdot i_2 + s_1 \cdot \overline{s_2} \cdot i_3 + s_1 \cdot s_2 \cdot i_4. \tag{3}$$

Since s_1 and s_2 are mutually exclusive, if all the data and select signals are equally probable to be 1 or 0, the probability of f being a "one" is $P_f = 0.5$. However, this assumption does not hold for the multiplexers in Fig. 1. Two types of multiplexers with constant inputs are used. `Type-1` multiplexer consists of four data inputs, x_i, y_i, s_i, and 0, and two select signals, x_j, y_j in any permutable order. Since the constant input has a probability of 0, the probability that this multiplexer outputs a "one" reduces to $P_{f1} = 0.375$. `Type-2` multiplexer comprises four data inputs, 0, 0, 0, and c_i, and two select signals, x_j and y_j. The data input, c_i can be written as $c_i = x_i \cdot y_i + (x_i + y_i) \cdot c_{i-1}$ and its expected value is given by

$$\begin{aligned} P_{c_i} &= P_{x_i \cdot y_i} + P_{(x_i+y_i) \cdot c_{i-1}} - P_{(x_i \cdot y_i) \cdot (x_i+y_i) \cdot c_{i-1}} \\ &= 0.25 + 0.75 P_{c_{i-1}} - 0.25 \cdot 0.75 P_{c_{i-1}} \\ &= 0.25 + 0.5625 P_{c_{i-1}}. \end{aligned} \tag{4}$$

Unrolling the above carry probability equation generates a geometric progression series with a common ratio of 0.5625. This sum approaches 0.5714 rapidly. Therefore, P_{c_i} is assumed an asymptotic value of 0.5714 and the probability that a `type-2` multiplexer outputs a "one" is $P_{f2} = 0.143$.

Let π_{1_i} and π_{2_i} be the pp bits generated respectively by the `type-1` and `type-2` multiplexers in the i^{th} column of the multiplexer matrix. If we normalize the weights of the pp bits of the truncated multiplexer matrix to the ulp of the truncated product so that the LSB of the full-width product with column index $i = 0$ is weighted 2^{-n}, the expected values of π_{1_i} and π_{2_i} are given by

$$E[\pi_{1_i}] = \mu_{1_i} = P_{f1} 2^{i-n} \quad E[\pi_{2_i}] = \mu_{2_i} = P_{f2} 2^{i-n} \tag{5}$$

where $i \in \{0, 1, \ldots, n-k-1\}$. Similarly, the expected value of the output of an AND gate in the i^{th} column is given by

$$E[\pi_{and_i}] = \mu_{\text{and}_i} = P_a 2^{i-n} \forall i \in \{0, 1, \ldots, n-k-1\}. \tag{6}$$

These expected values can be used to compute the following.

Reduction Error: Elimination of the multipliexers and AND gates in the least significant $n-k$ columns of the multiplexer matrix of full-width multiplier leads to reduction error. Each even column, except Column 0, has one `type-2` multiplexer, one AND gate and some λ_{even_i} number of `type-1` multiplexers. Column 0 consists of only an AND gate. On the other hand, each odd column possesses some λ_{odd_i} number of `type-1` multiplexers only. The values of λ_{even_i} and λ_{odd_i} depend on the column number, i, where $\lambda_{\text{even}_i} = \lambda_{\text{odd}_i} = \lceil i/2 \rceil$.

The expected value, $E(\text{mux1})_{\text{reduce}_i}$ of the reduction error contributed by neglecting `type-1` multiplexers in any even column "i", except $i = 0$ is given by

$$E(\text{mux1})_{\text{reduce}_i} = \lambda_{\text{even}_i} \mu_{1_i} = \left\lceil \frac{i}{2} \right\rceil P_{f1} 2^{i-n}. \tag{7}$$

Similarly, the expected value of the error due to `type-2` multiplexer and AND gate, $E(\text{mux2}\wedge)_{\text{reduce}_i}$ in any even column except column 0 is given by

$$E(\text{mux2}\wedge)_{\text{reduce}_i} = \mu_{2_i} + \mu_{\text{and}_i} = (P_{f2} + P_a) 2^{i-n}. \tag{8}$$

The expected error for the 0-th column is just equal to μ_{and_i}.

Since the i^{th} odd column has $\lambda_{\text{odd}_i} = \lceil i/2 \rceil$ of `type-1` multiplexers, the expected error, $E(\text{odd})_{\text{reduce}}$ of eliminating the i^{th} odd column is given by

$$E(\text{odd})_{\text{reduce}_i} = \lambda_{\text{odd}_i} \mu_{1_i} = \left\lceil \frac{i}{2} \right\rceil P_{f1} 2^{i-n} \quad \forall \, \text{odd } i. \tag{9}$$

From (7)–(9), the total reduction error, E_{reduce}, due to the elimination of Columns 0 to $n-k-1$ is given by

$$E_{\text{reduce}} = \begin{cases} 2^{-n} P_a, & i = 0 \\ \sum_{i=2}^{n-k-1} \left(\left\lceil \frac{i}{2} \right\rceil P_{f1} + P_{f2} + P_a \right) 2^{i-n}, & \forall \, \text{even } i \\ \sum_{i=1}^{n-k-1} \left\lceil \frac{i}{2} \right\rceil P_{f1} 2^{i-n}, & \forall \, \text{odd } i. \end{cases} \tag{10}$$

Rounding Error: The expected value of P_0 is 0.25 but the expected value of P_i quickly approaches 0.5 as i increases. Since k is small, a uniform probability of 0.5 can be assumed for P_{n-k} to P_{n-1}. Thus, the expected value of error, E_{round}, induced by ignoring the product bits in the additional k columns is given by

$$E_{\text{round}} = 0.5 \sum_{i=n-k}^{n-1} 2^{i-n} = 0.5 \sum_{j=1}^{k} 2^{-j}. \tag{11}$$

Total Error Before Compensation: From (10) and (11), the expected value of the total error is defined as

$$E_{\text{total}} = -(E_{\text{round}} + E_{\text{reduce}}). \tag{12}$$

The reduction and rounding errors are assumed to be negative. The correction bias is assumed to be positive to counteract the total error so that the mean error approaches zero.

Correction: The carry signals from the $(n-k-1)^{\text{th}}$ column are added to the $(n-k)^{\text{th}}$ column. The `carry` bit is obtained by a logical AND operation, i.e., $c = x \cdot y$. Hence the probability of c being a "one" is $P_c = P_x P_y = 0.25$. Besides, a correction constant of half-LSB is added. Thus, the entire compensation has an expected value of

$$C = 0.5 + \left\lceil \frac{n-k}{2} \right\rceil \times P_c \times 2^{-k}. \tag{13}$$

From (10)–(13), the mean error of the truncated multiplier, E_{avg}, is given by

$$E_{\text{avg}} = E_{\text{total}} + C = -(E_{\text{round}} + E_{\text{reduce}}) + C. \tag{14}$$

For the computation of variance, we assume that the reduction and rounding errors are independent [5]. This assumption gives an underestimate of the actual variance. For the i^{th} column, where $0 < i < n-k$, the variances $\sigma^2_{1_i}$, $\sigma^2_{2_i}$, and $\sigma^2_{\text{and}_i}$ respectively of the output of `type-1` multiplexer, `type-2` multiplexer and AND gate are given by

$$\sigma^2_{1_i} = \frac{15}{64} 2^{2(i-n)}; \quad \sigma^2_{1_i} = \frac{6}{49} 2^{2(i-n)}; \quad \sigma^2_{and_i} = \frac{3}{16} 2^{2(i-n)}. \tag{15}$$

In an even column i, except $i = 0$, the variance is due to $\lceil i/2 \rceil$ number of `type-1` multiplexers, one `type-2` multiplexer and one AND gate, i.e., $\sigma^2_{\text{even}_i} = \lceil i/2 \rceil \sigma^2_{1_i} + \sigma^2_{2_i} + \sigma^2_{and_i}$. For $i = 0$, the variance is due to an AND gate, i.e., $\sigma^2_{and_i}$. The variance of the reduction error by eliminating an ith odd columns is contributed by $\lceil i/2 \rceil$ number of `type-1` multiplexers. So the variance of the total reduction error by eliminating Columns 0 to $n-k-1$ is

$$\sigma^2_{reduce} = \begin{cases} \frac{3}{16} 2^{-2n}, & i = 0 \\ \sum_{i=2}^{n-k-1} \left(\lceil \frac{i}{2} \rceil \frac{15}{64} + \frac{243}{784} \right) 2^{2(i-n)}, & \forall \text{ even } i \\ \sum_{i=1}^{n-k-1} \lceil \frac{i}{2} \rceil \frac{15}{64} 2^{2(i-n)}, & \forall \text{ odd } i. \end{cases} \tag{16}$$

The variance of the rounding error is defined as the mean square difference between the rounding correction constant and the value of the truncated bits [5]. Rounding correction constant C_{round} is added to compensate for the total error and is equal to the difference between the correction, C and the total reduction error E_{reduce}

$$\sigma^2_{\text{round}} = 2^{-k} \sum_{q=0}^{2^k - 1} (C_{\text{round}} - q \cdot 2^{-k})^2. \tag{17}$$

Hence the variance of the total error is given by

$$\sigma^2_{\text{total}} = \sigma^2_{\text{round}} + \sigma^2_{\text{reduce}}. \tag{18}$$

The error histogram obtained by an exhaustive simulation with all input combinations of an 8-bit proposed truncated multiplier for $k = 2$ is shown Fig. 2. The Gaussian curve with the theoretical mean and variance follows the best-fit Gaussian to the error histogram, which validate the accuracy of our error analysis.

In Table I, we compare the average errors E_{avg} and variances σ^2 of PCT against CCT and VCT for $n = 8$, 12, and 16 and $k = 2$, 3, and 4. The architectures for $n = 8$ were simulated exhaustively for all possible input vectors. However, for $n = 12$ and $n = 16$, an exhaustive simulation is not feasible and two million pseudo-random input vectors were generated to simulate these architectures. Besides, we also compare PCT against the average errors and variances of the truncation schemes reported in [12] and [13]. The average errors of [13] are available for $n = 8$ with $k = 1$ and $k = 2$ only. For $n = 12$ and 16, the average errors were provided in [13] without indicating their k values. From Table I, the proposed PCT scheme has lower mean and variance for most values of n and k and the merit is more significant for $k > 2$. In cases where it is inferior, the error magnitudes are still comparable. Using the same percentage relative average error measure as in [8] for $k = 2$, our proposed PCT scheme produces a relative average error of 2.55% and 1.4% over the direct-truncated structure for $n = 8$ and 16, respectively. These values are nearly 4.4 times lower than the corresponding relative average errors of [8].

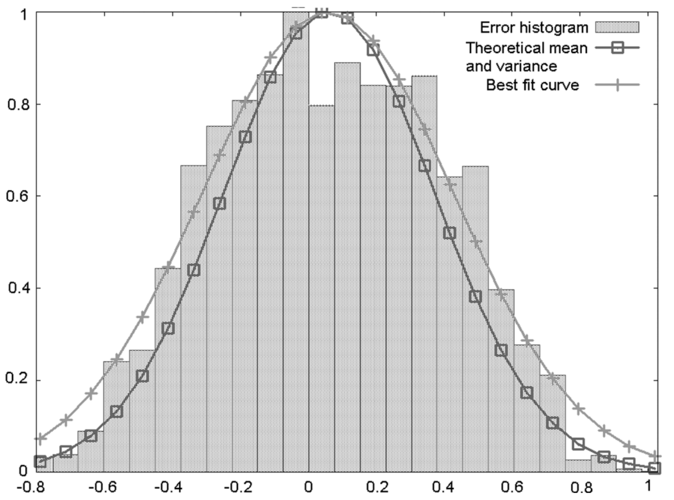

Fig. 2. Comparison of experimental result histogram for $\boldsymbol{n = 8}$ and $\boldsymbol{k = 2}$ and the Gaussian curve generated from the theoretical mean and variance.

TABLE I
COMPARISON OF $|\boldsymbol{E}_{\text{avg}}|$ AND $\boldsymbol{\sigma}^2$

	$\lvert E_{avg} \rvert$					
	$n = 8$		$n = 12$		$n = 16$	
	$k = 2$	$k = 3$	$k = 2$	$k = 3$	$k = 2$	$k = 3$
PCT	0.0576	0.0476	0.0180	0.0157	0.0623	0.0155
VCT	0.0615	0.0313	0.0625	0.0313	0.0621	0.0312
CCT	0.0595	0.0605	0.0625	0.0621	0.0631	0.0623
[13]	0.5264	NA	0.666		0.703	
[12]	0.4062		0.469		NA	
	σ^2					
PCT	0.1372	0.0675	0.1784	0.0832	0.2259	0.1038
VCT	0.0918	0.0848	0.0973	0.0864	0.1024	0.0877
CCT	0.1221	0.0933	0.1597	0.1031	0.1964	0.1121
[13]	NA		0.223		0.274	
[12]	0.0654		0.0763		NA	

NA: Not available

III. CS-MUX Implementation of Pseudo-Carry Compensation Truncated Multiplier

Fig. 3 shows the carry-saved multiplexer (CS-MUX) array implementation of the PCT scheme for $n = 8$ and $k = 2$. The outputs from each row of multiplexers in Fig. 1 are added in a carry-save format using full adders (FAs) and half adders (HAs). These FAs and HAs are combined with 4-to-1 multiplexers to give FMUX and HMUX. Except the multiplexer cells in the boundary, cells in the interior of the carry-save array architecture receive an additional `sum` signal from a preceding multiplexer cell. Therefore, HMUX cells are placed at the boundary and FMUX cells are placed in the interior of the array. The outputs from the topmost multiplexer and the AND gate in every even column of the multiplexer matrix in Fig. 1 are summed in an SA cell. These basic cells are similar to those shown in [11] and are illustrated in Fig. 3. ripple carry adders (RCAs) are used to add the final sum and

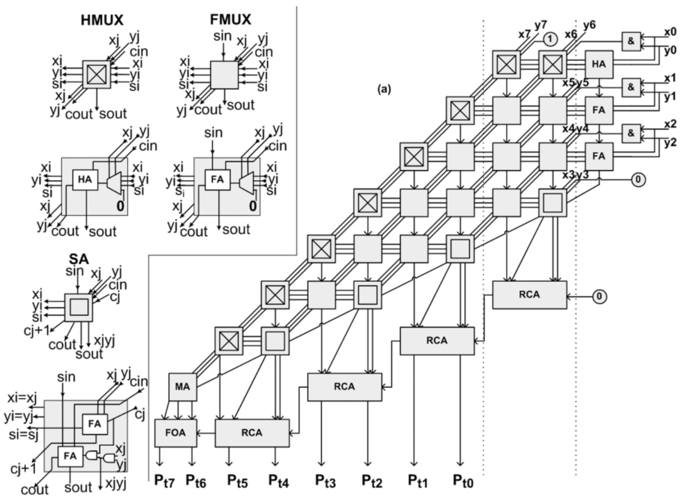

Fig. 3. Unsigned signed PCT multiplier architecture for $n = 8$ and $k = 2$ with its components HMUX, FMUX, SA.

carry. In addition, the MA cell at the lower left corner of the CS-MUX architecture is equivalent to a simplified SA cell with one FA and one HA. The RCA on the left end is substituted by a FOA cell to logically OR the two `carries` from the FA and MA cells to produce the MSB, $p_{t(n-1)}$. The adaptive error compensation is realized by the AND cells on the top right, and the fixed correction bias is realized with an input "1" placed on the leftmost HMUX in the first row.

The CS-MUX array implementation can be extended to the signed PCT multiplication. (1) can be modified to (19) when two n-bit signed integers X and Y in 2's complement form are considered, as in [11]

$$P = XY = \sum_{i=0}^{n-1}(x_i \cdot y_i)2^{2i} + \sum_{i=1}^{n-2} M_i 2^i - M_{n-1}2^{n-1}. \tag{19}$$

Equations (1) and (19) differ only by the negative $M_{n-1}2^{n-1}$. Thus, the outputs from the last row corresponding to the term, M_{n-1} are to be subtracted. To account for the negative $M_{n-1}2^{n-1}$, the outputs of the HMUX cells that lie on the left boundary of Fig. 3 are inverted. The two "1"s to be added to the $(n-2)^{\mathrm{th}}$ column of the multiplexer matrix of Fig. 1 for the subtraction are equivalent to two 0.25 LSBs for the n-bit normalized truncated product. The correction bias for the rounding error to be added in the $(n-1)^{\mathrm{th}}$ column is equivalent to 0.5 LSB of the truncated product. They total up to 1 LSB of the truncated product. A simple chain of HAs can be used to add this constant "1" to the LSB of the final truncated product obtained from the row of RCAs in Fig. 1. The error analysis described in Section II is still valid for the signed PCT multiplication.

IV. Results and Discussion

Since the signed PCT multiplier has comparable hardware complexity as its unsigned version, it is sufficient to compare only the unsigned variants of the truncated multipliers. The architectures were described using structural VHDL and synthesized and optimized by Synopsys Design Compiler using the same TSMC 0.18-μm CMOS standard cell library. The input and output loads were set to 0.8 pF for all designs. The synthesis results of various truncated multipliers for $n = 16$ and 32, and $k = 2$ to 5 are shown in Table II. Our proposed truncated multiplier consumes less silicon area than VCT array multipliers consistently for all values of n and k. The area savings of our proposed truncated multiplier for $n = 16$ varies between 14% to 20% when it is collated with VCT array multiplier. When the input

TABLE II
Delay-Area-Power Comparison of PCT and VCT

Delay and Area Results								
	$m = 16$				$m = 32$			
	Delay (ns)		Area (μm^2)		Delay (ns)		Area (μm^2)	
k	PCT	VCT	PCT	VCT	PCT	VCT	PCT	VCT
2	4.04	4.94	21641.54	27249.76	7.21	9.69	71887.05	96618.18
3	4.03	5.11	23584.12	27971.57	7.08	9.67	74468.25	103204.31
4	4	5.11	25719.68	30682.57	7.06	10.12	79597.45	101009.03
5	4.05	5.22	26777.46	31128.29	7.21	10.14	82301.81	109910.1
Power Dissipation Results								
	Dynamic Power (μW)				Leakage Power (nW)			
	$m = 16$		$m = 32$		$m = 16$		$m = 32$	
k	PCT	VCT	PCT	VCT	PCT	VCT	PCT	VCT
2	161.8	157.95	545.41	540.09	102.32	125.79	324.73	448.02
4	180.11	176.36	592.32	586.44	113.42	141.21	351.02	486.85
6	194.32	191.53	635.64	629.53	122.68	153.63	375.41	522.87
8	206.05	202.51	671.61	666.07	130.09	163.05	398.03	555.98

TABLE III
Image Compression Metrics

	Lena		Baboon		Peppers	
	CR=5	CR=15	CR=5	CR=15	CR=5	CR=15
$k = 2$	23.55	219.72	102.31	224.36	54.17	247.74
$k = 6$	51.54	218.38	100.12	223.77	51.98	247.43
FW	50.19	218.09	99.27	223.41	51.23	246.82

wordlength increases to $n = 32$, the proposed truncated multiplier occupies 21% to 28% less area than VCT multiplier. It is worth noting that PCT multiplier with $k = 2$ saves 33% and 41.5% of silicon area over its full-width counterpart for $n = 16$ and $n = 32$, respectively, although their speeds are comparable. In terms of critical path delay, the proposed PCT multiplier is faster than the VCT array multiplier by 21% and 27% for $n = 16$ and 32, respectively. We have also technology mapped, optimized and simulated the redundant binary fixed-width multiplier codes for $n = 10$ provided by the author of [12] using the same standard cell library and simulation setup. The architecture of [12] is found to be 13% faster than our PCT multiplier but consume 65% more area than our design.

The average power dissipations of different architectures were simulated by Synopsys Power Compiler [14] using a Monte Carlo model [15]. This statistical method provides a good confidence that the estimated average power is bound within a given error [14] without the need for an exhaustive simulation of all input vectors. Table II shows the mean dynamic power dissipations with a confidence level of 90% and an error bound of 3% for 16-bit truncated multipliers and 5% for 32-bit truncated multipliers for $2 \leq k \leq 8$. The proposed truncated multiplier dissipates slightly more dynamic power but no more than 5% above the VCT truncated multipliers for all n and k. It still consumes a noteworthy 35% and 42% lower power than its full-width counterpart for $n = 16$ and 32, respectively. In terms of leakage power, the proposed PCT multiplier is lower than VCT multiplier by as much as 28%.

The proposed truncated multiplication scheme is applied to a DCT-based image compression algorithm, JPEG [16]. 12-bit multiplications are used in the compression and decompression operations. Table III shows the mean square errors (MSE) of three 128 $\times$ 128 images with various compression ratios (CR) obtained by using the full-width (FW) multiplier and the proposed PCT multiplier with $k = 2$ and 6. The deviations in peak signal-to-noise ratio (PSNR) are less than 1 dB even for $k = 2$. The 12-bit 2's complement truncated multiplier with $k = 2$ also saves about 30% area and 14% power over its full-width counterpart at the same speed of operation.

V. Conclusion

The efficiency of digital multiplication can be improved tremendously by truncation methods provided precise outputs are not required for the operation. This paper proposed a new multiplexer based truncation scheme with lower average and mean square errors than existing truncation methods. Its VLSI implementation on TSMC 0.18 μm technology shows an area reduction over existing truncated array multipliers by at least 14% and 21% in 16- and 32-bit multiplications, respectively. It also runs faster than the VCT multiplier by more than 21% with comparable dynamic power dissipation. When the new truncated multiplication was incorporated into JPEG image compression, the differences in PSNR of the reconstructed images were within 1 dB from those obtained with full-precision multiplication.

References

[1] U. M. Baese, *Digital Signal Processing With Field Programmable Gate Arrays*. Berlin, Germany: Springer, 2004.

[2] V. Mahalingam and N. Ranganathan, "An efficient and accurate logarithmic multiplier based on operand decomposition," in *Proc. VLSI Des.*, Jan. 2006, p. 6.

[3] N. Yoshida, E. Goto, and S. Ichikawa, "Pseudorandom rounding for truncated multipliers," *IEEE Trans. Computers*, vol. 40, no. 9, pp. 1065–1067, Sep. 1991.

[4] 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 Digit. Signal Process.*, vol. 46, no. 6, pp. 836–842, Jun. 99.

[5] M. J. Schulte and E. E. Swartzlander, Jr., "Truncated multiplication with correction constant," in *Proc. IEEE Workshop VLSI Signal Process. VI*, Oct. 1993, pp. 388–396.

[6] E. J. King and E. E. Swartzlander, Jr., "Data-dependent truncation scheme for parallel multipliers," in *Proc. 31st Asilomar Conf. Signals, Syst. Comput.*, Nov. 1997, vol. 2, pp. 1178–1182.

[7] M. J. Schulte, J. E. Stine, and J. G. Jansen, "Reduced power dissipation through truncated multiplication," in *Proc. IEEE Alessandro Volta Memorial Workshop Low-Power Des.*, Mar. 1999, pp. 61–69.

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

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

[10] C. H. Chang, R. K. Satzoda, and S. Sekar, "A novel multiplexer based truncated array multiplier," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Kobe, Japan, May 2005, pp. 85–88.

[11] K. Z. Pekmestzi, "Multiplexer-based array multipliers," *IEEE Trans. Comput.*, vol. 48, no. 1, pp. 15–23, Jan. 1999.

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

[13] 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.

[14] R. K. Satzoda, C. H. Chang, and T. Srikanthan, "Monte Carlo statistical analysis for power simulation in Synopsys design compiler," presented at the Synopsys Users' Group Conf., Singapore, Jun. 2006 [Online]. Available: http://www.snug-universal.org/papers/papers.htm

[15] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo approach for power estimation," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 1, no. 1, pp. 63–71, Mar. 1993.

[16] D. Salomon, *Data Compression—The Complete Reference*, 2nd ed. New York: Springer-Verlag, 2000.