Profile-based Output Error Compensation for Approximate Arithmetic Circuits

Ke Chen, Weiqiang Liu, *Senior Member, IEEE*, Jie Han, *Senior Member, IEEE*, and Fabrizio Lombardi, *Fellow, IEEE*

***Abstract*—Truncation is one of the most commonly used approaches for circuit-level approximate computing. This paper proposes a scheme for error compensation of arithmetic circuits in which a so-called padding is utilized to compensate at the output for the truncated bits of the input operands. Compensation relies on adjusting the output results of an arithmetic circuit; the padding takes a value determined by utilizing statistical information based on profiling an arithmetic circuit to reduce the average signed difference between the inexact and exact values and so the mean square error. An extensive analysis and simulation-based evaluation of error metrics are performed on signed truncated adders, multipliers and dividers; an excellent agreement is found. Additional design metrics such as power consumption and circuit complexity are also assessed. Different applications of approximate arithmetic circuits with the proposed output error compensation scheme are presented. Matrix multiplication and image processing (changing detection) are investigated to show the effectiveness of the scheme proposed in this paper.**

***Index Terms*—Approximate Computing, Arithmetic Errors, Computer Arithmetic, Error Compensation**

I. Introduction

THE need for high performance digital systems is often constrained by the requirement of low power/energy consumption. In many applications such as those requiring arithmetic functions for digital signal processing, low delay and energy consumption of the arithmetic units are of primary importance in the overall evaluation of the entire computing system [1]. Most computer arithmetic applications require the highest degree of accuracy; however, applications such as those found in multimedia and image processing, can tolerate errors and imprecision in computation and still produce meaningful and useful results [2]. This allows imprecise or approximate computation to redirect the existing design process to decrease computational complexity and hardware requirements with increases in performance and power efficiency [3].

K. Chen and F. Lombardi are with Department of Electrical and Computer Engineering, Northeastern University, Boston MA 02115, USA (email: chen.ke1@husky.neu.edu, lombardi@ece.neu.edu).

W. Liu is with College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China (email: liuweiqiang@nuaa.edu.cn).

J. Han is with Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, T6G 1H9, Canada (email: jhan8@ualberta.ca).

Approximate computing (AC) has been applied at circuit and algorithmic levels. At circuit level, several arithmetic circuits have been proposed; for example, in adders the most significant bits are exactly computed, while several less significant bits are computed inexactly using a simplified circuit. Approximate mirror adders [4][5], approximate XOR/XNOR-based adders [6] and lower-part-OR adders [7] are some examples of adder circuits used for AC. [8][9] have implemented approximation by employing speculative adders. Approximate multipliers have utilized speculative adders for generating the partial products [10]. [11] has proposed a compressor using only one majority gate by AND-OR and XOR logic. [12] has introduced the partial product perforation technique, in which the generation of some partial products based on the modified Booth encoding, is omitted. [13] has proposed three Approximate Booth Multiplier Models (ABMM) to archive low power operation, while preserving accuracy. Approximate logic level synthesis has also been investigated [14]. [15] has proposed a Systematic methodology for Automatic Logic Synthesis of Approximate circuits (SALSA) that synthesizes an approximate version of the circuit to meet pre-specified quality bounds. At algorithmic level, the feature of incremental refinement found in iterative algorithms, has been exploited to achieve results that gradually increase the desired metric [16][17]; for example, dynamic bit width adaptation has been used to increase energy efficiency [18].

AC employs designs that produce imprecise results [19]. The simplest scheme for AC is truncation, i.e. the removal of some parts of less importance from the inputs of a circuit, such as the least significant bits (LSBs) in an arithmetic circuit. Truncation is simple to implement and control; the so-called reduced width circuit has been extensively proposed for AC, but its implications on error are rather severe, as inaccuracy increases with the level of truncation. Statistical error compensation exploits hardware features of the underlying implementation to provide compensation, such that the output may be in error, but it still guarantees that application-level specifications on inaccuracy are met. Corrective schemes by which mitigation can be implemented, remedy an inexact output when the error (as measured by the relevant metrics) exceeds the desired level for an application. Therefore, there is a compelling need to provide adequate output compensation for approximate arithmetic circuits employing input truncation.

In this paper, circuit-level input truncation is complemented by utilizing a simple error compensation scheme at the output

to reduce power consumption, circuit complexity and error metrics. Using the proposed low complexity error compensation scheme, the objective of significantly reducing the average (signed) error introduced by truncation is met by analyzing and proposing an appropriate value for the so-called padding (which is generated based on the statistical information of the input pattern and profiling an arithmetic circuit). The proposed scheme is utilized for (signed) addition, multiplication and division; an exhaustive simulation assessment and error analysis are also provided to show the effectiveness of the proposed scheme. Different applications of approximate arithmetic circuits with the proposed error compensation scheme are also presented.

The rest of the paper is organized as follows: Section 2 provides a brief overview of AC. The proposed compensation scheme is outlined in Section 3. The detailed implementation and evaluation for circuits such as the adder, multiplier and divider using the proposed scheme are presented in Sections 4 to 6. The application of the proposed schemes to matrix multiplication and image processing are presented in Section 7. The paper ends with the conclusion in Section 8.

II. Review

AC spans a wide range of research activities from circuits to programming languages. It includes arithmetic circuit design at transistor and logic levels, approximate memory/storage (including SRAM, DRAM and non-volatile memories), and processor architectures (including neural networks, general-purpose and reconfigurable processors such as instruction set architectures (ISAs), graphic processing units (GPUs) and FPGAs) [4]. Applications of AC include image and signal processing, classification, recognition and machine learning, among the many possible.

This paper focuses on arithmetic circuits; arithmetic circuits are usually modular and homogeneous, often designed as arrays. For AC, two approaches are usually utilized in a hardware array for modular arithmetic design:

Cell Replacement. In this case, an approximate design consists of removing at least an exact cell and replacing it by an approximate cell; for example, the extent by which this replacement process is performed in an array (such as a divider) is quantified by the depth (d), i.e., the number of rows (and/or columns) in the array with approximate cells. The approximate cells have a smaller circuit complexity, so making the overall design less complex, consuming less power and often in many cases operating faster. These advantageous features are accomplished at the expenses of introducing errors at the outputs.

Cell Truncation. Truncation consists of fully removing at least a cell (so no replacement with approximate cell(s)) in the rows/columns in the array. Usually, cell truncation introduces more error compared with cell replacement, while consuming also less power. The tradeoff between error and power dissipation must be carefully considered in a truncation scheme.

For different truncation configurations, the inexact computation circuit must be configured in hardware to meet the specific requirements of an application; so, at least some parts of a computational module should be turned on/off according to the desired configuration and the reduced bit-width of the arithmetic operation. In this paper, power gating is used to accomplish a bit-width reduction. In this scheme, two transistors are added to a generalized CMOS gate: a PMOS in series with a pull-up network (PUN) and an NMOS in parallel with a pull-down network (PDN) (Fig. 1). Note that there is also a small increase in operational delay due to an additional PMOS transistor. The value of the control signal CON is determined by the target error; if the control signal CON is low, then the circuit operates as a normal gate. When the control signal is high, the output of the gate is forced to zero regardless of the input signal. If only the input gates of a functional block (or module) are modified, then a high value on CON turns off

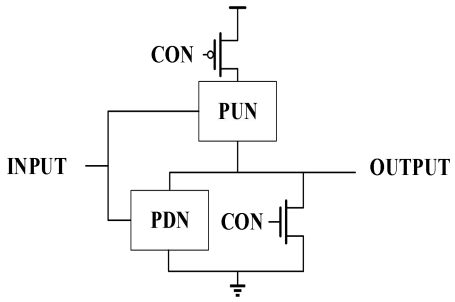


Fig. 1. Power gating scheme.

the functional block, because there is no switching activity in the connected gates. Power gating incurs in hardware and delay overheads (albeit, they are both very small, nearly negligible in most cases).

III. Proposed Error Compensation Scheme

Truncation is usually applied at the inputs and/or intermediate results. When truncation is introduced only at the inputs, the computation result resembles its exact counterpart in the MSBs, but at a reduced bit width for computing, often reducing power dissipation and circuit complexity. In this paper, truncated approximate arithmetic circuits are considered; truncation is introduced during the calculation (so reducing computation and circuit complexities) and a so-called padding is inserted at the output (Fig. 2). This approach is referred to as compensation. The principle of error compensation is based on partially adjusting/reducing the error introduced by input truncation; this arrangement is based on a-prior acquired statistical information of the approximate circuit. In general, approximate circuits are truncated in the k least significant bits. So rather than representing the truncated k bits with a predetermined simple pattern (such as all '0's), a k-bit padding is used; padding takes a value to reduce the total average distance as well as the mean square error in a specific application. The value of padding is calculated by statistical analysis and by considering the nature of the arithmetic computation of the circuit.

The following conditions are assumed throughout this work.

1. The (exact) arithmetic circuit has a modular design, i.e. the basic module (referred to as a cell) is employed in a homogeneous scheme.

2. The (exact) arithmetic circuit operates on a signed integer format (using 2's complement).

3. All designs for the proposed schemes have been implemented in HDL and mapped to the FreePDK 45nm library.

The general flow of the proposed approximate and error compensation schemes using padding is shown in Fig. 3

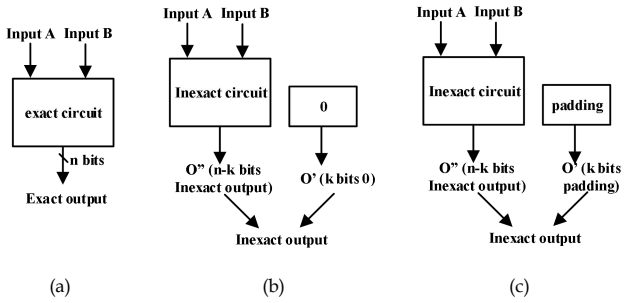


Fig. 2. Block diagrams of (a) exact circuit, (b) inexact circuit with conventional truncation, and (c): input truncation with output padding.

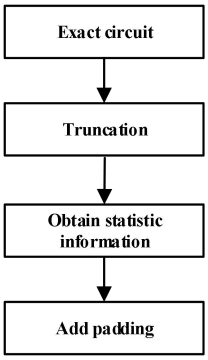


Fig. 3. Design flow for AC by utilizing the proposed padding scheme.

- **Truncation:** Starting from an exact arithmetic circuit, the first step employs input truncation for AC. By truncating some of the cells in a modular design implementation, circuit complexity and power consumption can be significantly decreased; however, for such inexact circuit, an error is also introduced due to truncation of the least significant bits.
- **Statistical profiling:** An assessment (likely if possible, involving exhaustive simulation) of the inexact circuit is pursued. Specifically, the metric of average signed error between the exact and the approximate outputs for all input combinations is recorded; the distribution of the signed error among all supplied input combinations is then plotted.
- **Adding padding based on the statistical profiling:** In 2's complement, once truncation is applied to an arithmetic circuit, the inexact result is normally smaller than the exact value. This occurs because in most of these schemes, the least significant bits of the inputs (or intermediate values) are truncated. After obtaining statistic information, padding is added at the output to the inexact result in place of the truncated bits to compensate for the error. So, the value of the output padding is based on the statistical distribution of the exact output value. Under a uniform input distribution, the signed average error is selected as the value of the padding.

As shown in the next section, the most common arithmetic operations are analyzed with respect to two operand signed arithmetic in integer format. Under exact conditions, the common feature of these operations is that the output distribution is symmetric in both the negative and positive ranges and centered by the largest number of occurrences on 0 as output value. This is a very important feature that can be used for AC when utilizing input truncation to reduce the error.

Therefore, the basic principles of the proposed approach for AC are based on the following features.

- Truncation with no padding results (such as by using the commonly used pattern of all 0's) in a shift of the output distribution, while still remaining symmetric, but not centered on 0 as the output value of largest occurrence.
- The value of padding shifts back to the center value of the distribution, while significantly reducing the average signed error.

The value of the padding is dependent on the arithmetic operation as well as the distribution of the two input values in the inexact circuit. In this paper, a normal distribution is assumed for the input values; this assumption is validated using three applications in a later section as well as analytically. It should be also noted that compensation by padding is analyzed in this paper at an arithmetic level; while specific circuits for these operations are considered, this technique can be utilized for other arithmetic circuits provided the truncated part is identified from the retained (not truncated) part. The proposed technique is therefore an output-based compensation that remedies on average the error of an arithmetic operation for AC.

IV. Addition

Consider the operation of addition and as an example, the case of adding two 8-bit operands in signed (2's complement) format, only those results with no under/overflow are considered. For this operation, the distribution of the sum using exhaustive simulation is shown in Fig. 4; the distribution is symmetric with respect to a sum value of 0. Consider next an n-bit ripple carry adder as implementation (Fig. 5). As a truncated adder, only n-k cells are used for calculating the sum (in Fig. 5, k=2); so, the least significant k cells are truncated and when employing the proposed scheme, a k-bit padding must be used for the sum (where the k-bit padding is considered as having an unsigned value).

The distribution of the (inexact) sum for n=8 and k=2 can be plotted for a specific padding value; the distributions for padding values of 00 and 11 are plotted in Fig. 6 under an exhaustive simulation. The shape of the error distribution is still the same as in Fig. 4; however, the center value of the distribution is a function of the value of the padding. It is shifted in the negative range for a padding value of 00 and

centered again for a padding value of 11 (3).

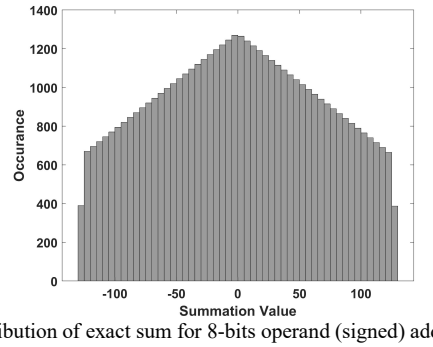


Fig. 4. Distribution of exact sum for 8-bits operand (signed) addition.

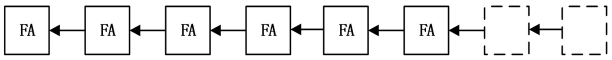
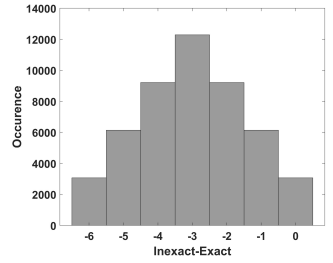
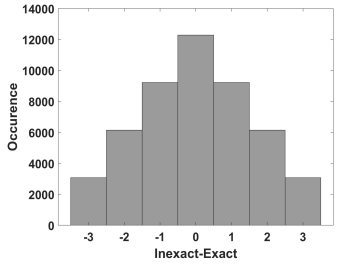


Fig. 5.Truncated inexact ripple carry adder (n=8, k=2).



(a)



(b)

Fig. 6. Distribution of inexact sum for 8-bits (signed) addition with truncation k=2 for values of a) padding=00, and b) padding=11.

TABLE I
ERROR STATISTICS FOR SIGNED ADDITION WITH K=2 AND N=8 WITH PADDING

Padding Value	0 Error Count	Largest error distance	Average distance	Average signed error
00	3072	-6	3	-3
01	6144	-5	2.125	-2
10	9216	-4	1.5	-1
11	12288	-3	1.25	0

TABLE II
AVERAGE SIGNED ERROR (INEXACT-EXACT) FOR ADDER TRUNCATION WITH/WITHOUT PADDING (N=8).

Average signed error	k=1	k=2	k=3	k=4
Padding	'1'	'11'	'111'	'1111'
No padding	-1	-3	-7	-15
After padding	0	0	0	0

TABLE I shows the results for different error metrics (the so-called 0-error count denotes the number of occurrences when the output shows no error with respect to the exact value, so at a 0 distance). The distribution is also a function of k; TABLE II shows the average signed error results with different truncation levels for n=8. For an input uniform distribution, output padding can fully compensate the average signed error introduced by truncation for an addition.

Compensation can be analyzed as follows; consider again as an example an adder for n=8 and k=4; the exact result can be expressed by:

$$\text{Exact} = A_7A_6A_5A_4A_3A_2A_1A_0 + B_7B_6B_5B_4B_3B_2B_1B_0$$
$$= A_7A_6A_5A_40000 + B_7B_6B_5B_40000 + A_3A_2A_1A_0 + B_3B_2B_1B_0 \quad (1)$$

Let

$$\text{truncated sum} = A_3A_2A_1A_0 + B_3B_2B_1B_0$$

Thus,

$$\text{exact} = A_7A_6A_5A_40000 + B_7B_6B_5B_40000 + \text{truncated sum}$$

For the inexact result,

$$\text{Inexact} = A_7A_6A_5A_40000 + B_7B_6B_5B_40000 + \text{padding}$$

Therefore,

$$\text{error} = \text{inexact} - \text{exact} = \text{padding} - \text{truncated sum} \quad (2)$$

The distribution is symmetric, in this case it is centered at 15 and the range is [0, 30]. Using the results found by exhaustive simulation, the distribution can be represented as a function given by

$$\text{Occurence} = 3072 - 192 * |15 - \text{Truncated sum}|$$
$$\text{Truncated sum} = \{0,1,\ldots 30\}$$

The average value of the truncated sum is given by

$$\text{Avg sum} = \frac{\sum_{i=0}^{30} \text{Truncated sum}_i \times \text{Occurence}_i}{\sum_{i=0}^{30} \text{Occurence}_i} = 15$$

The error is equal to a smaller truncated sum to which the padding value is added; so, the error distribution effectively flips the truncated sum over the y-axis and then shifts it to the right by M, where M denotes the value of the padding. If the padding value is '0000', the error distribution only flips the truncated sum with a center value of -15 and the range is now given by [-30, 0]. When the padding value is '1111' (i.e. 15), the error distribution flips the truncated sum and so it shifts it right by 15, i.e. now the center value is at 0 and the range is restored to [-15,15].

In the general case, a discrete signed error distribution with both positive and negative errors is applicable. Let the largest positive error be $+E_{p.max}$ while the largest negative error be $-E_{N.max}$. Each occurrence in the distribution is given by O_i with $i \in [-E_{N.max}, +E_{p.max}]$. The total signed error is given by :

$$\text{SE}_{\text{total}} = \sum_{i=1}^{+E_{p.max}} i \times O_i - \sum_{j=1}^{+E_{N.max}} j \times O_j \quad (3)$$

$$\text{SE}_{\text{avg}} = \text{SE}_{\text{total}} \Big/ \sum_{i=-E_{N.max}}^{+E_{p.max}} O_i \quad (4)$$

The design of the tunable truncated inexact (ripple carry) adder is shown in Fig. 7. It includes full adder cells, MUXes

and Pull UP Networks (PUNs). The selector of each MUX (C3 to C0) is connected with its PUN that allows the MUX to select the result from the full adder cell when the selector signal is 0 as we assume that the largest truncation level of the adder is the half bit-width for the operand. Padding is available to the final result provided the digit has a selector signal with a value of 1, i.e. to turn off the full adder cell for that digit. The simulation results at different values of n and k for the area, delay and power per the proposed scheme with padding are reported in TABLE III; by increasing k, the working power and delay metrics decrease as expected because of the power gated of the truncated FAs. However, the auxiliary MUXes and the PUN will introduce more area cost. If the truncation level k can be fixed in specific applications, the padding can be directly inserted to the LSB of the result without the FAs and MUXes, which can save the circuit cost further.

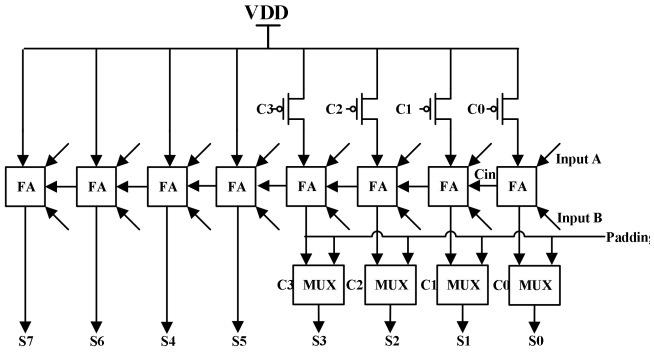


Fig. 7. Tunable truncated inexact adder with output padding as compensation.

TABLE III
AREA, POWER AND DELAY FOR ADDITION SCHEME

Truncation level	Area (μm^2)	Delay (ns)	Power (μw)
	n=8		
Exact	182.8	0.34	53.4
k=2	205.3	0.25	47.4
k=4	205.3	0.25	45.8
	n=16		
Exact	274.4	0.57	101.5
k=2	334.2	0.44	95.7
k=4	334.2	0.43	93.8
k=8	334.2	0.43	90.2

V. MULTIPLICATION

In this section, the more complex arithmetic operation of signed multiplication is analyzed for AC with output compensation.

$$
\begin{array}{cccccc}
 & & & a_2 & a_1 & a_0 \\
 & & \times & b_2 & b_1 & b_0 \\
\hline
 & & 1 & \overline{a_2b_0} & a_1b_0 & a_0b_0 \\
 & & \overline{a_2b_1} & \overline{a_1b_1} & a_0b_1 & \\
1 & a_2b_2 & \overline{a_1b_2} & \overline{a_0b_2} & & \\
\hline
m_5 & m_4 & m_3 & m_2 & m_1 & m_0
\end{array}
$$

Fig. 8. 3-bit operand binary multiplication.

Consider as an example the Baugh-Wooley multiplier as implementation of a signed multiplier for calculating the product of the two n-bit binary input operands $A_{ik} = (a_n-1, \ldots, a_1,a_0)$ and $B_{kj} = (b_{n-1}, \ldots, b_1,b_0)$) where an−1 and bn−1are the sign bits. The result is represented by a 2n-bit output value M $= A_{ik} \cdot B_{kj} = (m_{2n-1}, \ldots, m_1,m_0)$. The computation in a 3-bit Baugh-Wooley multiplier is shown in Fig. 8.

In Fig.8 , each bit of the product (m_i) corresponds to the cumulative sum of the partial products ($a_k b_{i-k}$, i.e.$m_{I,i=0,1,2,3,4} = \sum_{k=0}^{i} a_k b_{i-k}$).

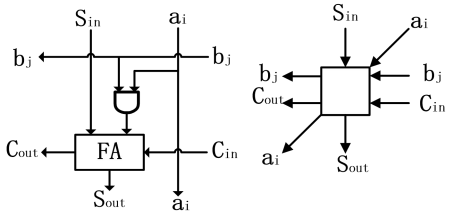


Fig. 9. Exact AND partial product cell (PPC): (a) Logic circuit and (b) symbol.

There are two basic operations in the partial product: AND and NAND; thus, two types of a partial product cell (PPC) are required. Fig. 9 shows the first type of PPC; it computes a single bit multiplication of a_i and b_j using an AND-gate. The result is then added using a full adder (FA) cell with the previous sum output Sin from a cell located above, and the previous carry output C_{in} from a cell to the right to generate the final S_{out} and C_{out}, the second cell (Fig. 10) differs only in the gate (now a NAND).

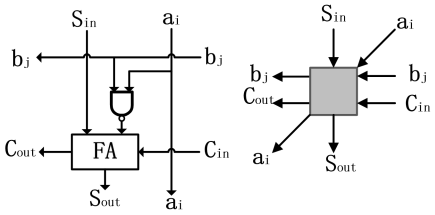


Fig. 10. Exact NAND partial product cell (PPC): (a) logic circuit, and (b) symbol.

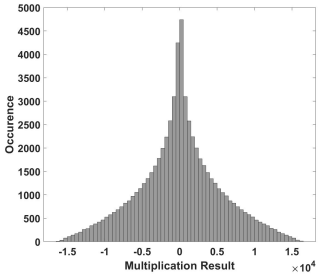


Fig. 11. Distribution of exact (product) output for 8-bit operand signed multiplication.

Consider next truncation. For an n-bit multiplier, truncation can be implemented in two ways: 1) Horizontal truncation; and 2) Vertical truncation.

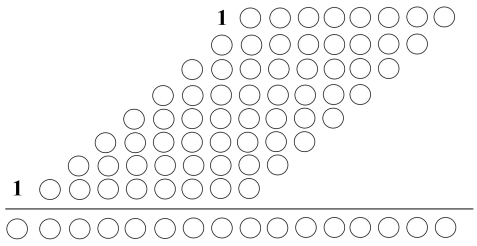


Fig. 12. 8-bit exact multiplier.

Fig. 12 shows an 8-bit exact multiplier. Horizontal truncation requires truncating the first k rows of partial products and then adding a k-bits padding to the result (i.e. the final product); therefore, a k×n partial product is truncated. An example of horizontal truncation (n=8, k=2) is shown in Fig. 13; the black dots identify those partial products that are truncated.

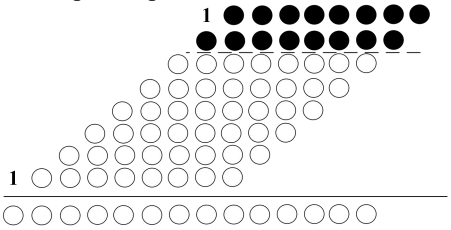


Fig. 13. Horizontal truncation for n=8, k=2.

Vertical truncation consists of truncating the least significant k columns of the partial products and then appending a k-bit padding to the result, i.e. the $\sum_{i=1}^{k} i$ partial products are truncated. An example of vertical truncation (n=8, k=4) is shown in Fig. 14.

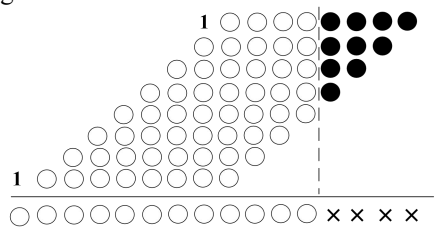


Fig. 14. Vertical truncation for n=8, k=4; the x denotes a padding bit.

Consider next the padding value and vertical truncation. In vertical truncation, padding is simple to implement. Consider the multiplication of two 8-bit operands as an example, in this case, initially an exhaustive simulation of the multiplication for two 8-bit operands is conducted to assess the statistical information on this arithmetic operation; this is a rather brute force approach (certainly not possible for large values of n), but it provides an excellent understanding of the entire multiplication process and its inexact counterpart.

The simulation results with and without padding are presented in TABLE IV for different values of k and n=8.

For each row of the partial products, the weight is different, i.e. the weight of the next row is twice the weight of the current row. Let w be the weight of the first row; the weights for an 8-bit multiplier are therefore given by 2^0×w, 2^1×w, … 2^7×w. If k is the number of bits for vertical truncation, then there are k truncated rows (from 0 to k-1); so for row i (i from 0 to k-1), the weight is given by 2^i×w and the number of truncated bits on row i is k-i. If every truncated digit is replaced by a 0, the average error introduced by row i is the product of the weight of row i and the average value generated by the k-i bits.

TABLE IV
AVERAGE SIGNED ERROR (INEXACT-EXACT) FOR VERTICAL TRUNCATION WITH/WITHOUT PADDING (N=8)

Padding	No Padding	With Padding
'0' (k=1)	-0.25	-0.25
'01' (k=2)	-1.25	-0.25
'100' (k=3)	-4.25	-0.25
'1100' (k=4)	-12.25	-0.5

The k-i bits have a value in a range from 0 to $2^{k-i}-1$. This value determines the corresponding bits of the multiplicand; the average value is in a range from 0 to $2^{k-i}-1$ is $(2^{k-i}-1)/2$ and the average value for the multiplicand bit is 0.5 (between 0 and 1). Thus, the average value of the k-i bits is $(2^{(k-i)}-1)/4$ (for example, k=2 and i=0, the average value of these 2-bits is $(00_2+01_2+10_2+11_2)/4\times 1/2=0.75$). The generalized expression for the error introduced by row i is:

$$\text{avg error}_i = 2^i w \times \frac{\sum_{j=0}^{2^{k-i}-1} j}{2^{k-i}} \times \frac{1}{2} = 2^i w \times \left(2^{k-i}-1\right) \times \frac{1}{4} = \frac{2^k - 2^i}{4} \quad (10)$$

The total error for k rows (vertical truncation) is given by:

$$\text{total error} = \sum \text{avg error}_i = \frac{(k-1)2^k + 1}{4} \quad (11)$$

Thus, to compensate this error, the padding should be an integer with a value given by $((k-1)\times 2^k + 1)/4$. Fig. 15 plots the average vertical truncation error varying the truncation level k.

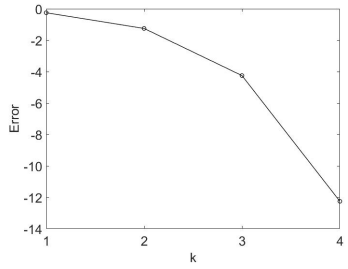


Fig. 15. Average vertical truncation error versus k for multiplication with padding.

Consider next horizontal truncation; this truncation scheme is more complicated than vertical truncation. As an example, again an exhaustive simulation of the multiplication for two 8-bit operands is conducted with a string of 0 used as initial padding. Next, to make the average signed error close to 0, a padding should be added like a partial sum of the truncated rows. As per the operation of a Baugh-Wooley multiplier, a 2^n padding is added to the exact result. Thus, different values of padding are shown in TABLE V. Same as in TABLE IV, the value of the average signed error using padding is negative.

TABLE V
AVERAGE SIGNED ERROR (INEXACT-EXACT) FOR HORIZONTAL TRUNCATION WITH/WITHOUT PADDING (N=8)

Padding	No Padding	With Padding
'11111111' (k=1)	-383.75	-0.75
'1001111111' (k=2)	-639.25	-0.25
'10001111110' (k=3)	-1150.25	-0.25
'100001111100' (k=4)	-2172.25	-0.25

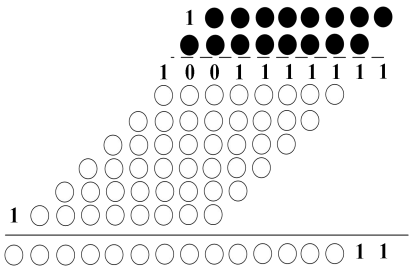


Fig. 16. Example of padding for horizontal truncation (n=8, k=2).

For horizontal truncation of k rows, the value of row i is equal to the multiplier (1st input) AND with the i_{th} LSB of the multiplicand (2_{nd} input). As per the operations of a Baugh-Wooley multiplier, a term of 2^n must be added to obtain the exact result.

For row 0, there are two scenarios: the LSB of the multiplicand is either 1 or 0; the occurrence of either scenario requires to take into account the following cases.

- If it is 0, in a Baugh-Wooley multiplier the $(n-1)_{th}$ bit of that row will be 1, while the other bits will be 0.
- If it is 1, the n-2 to i bits will have a copy of the corresponding bits of the 1st input. However, the $(n-1)_{th}$ bit will flip. The row can take a value from 0 to $2^n - 1$.

Thus, the average error due to truncation of row 0 is:

$$\text{avg error}_{\text{row 0}} = 2^{n-1} \times 0.5 + \sum_{j=0}^{2^n-1} j \times 0.5/2^n = (2^{n+1} - 1)/4 \quad (12)$$

For row I, the weight is 2^I of row 0, thus

$$\text{avg error}_{\text{row i}} = \text{avg error}_{\text{row 0}} \times 2^i = (2^{n+1} - 1) \times 2^{i-2}$$

The total average error is given by:

$$\text{total error} = \sum \text{avg error}_i + 2^n \text{truncation} = \frac{2^{n+1} - 1}{4} \times (2^k - 1) + 2^n \quad (13)$$

For example, when n=8 and k=3, the total average error is:

$$(2^{(8+1)} - 1)/4 \times (2^3 - 1) + 2^8 = 1150.25$$

This is the same as found from simulation for the considered values of n as shown in Fig. 17. By comparing the results of TABLE IV with TABLE V, horizontal truncation has an average signed error larger than vertical truncation when no output padding is applied; after padding, these two truncation methods are similar as far as the average signed error after compensation.

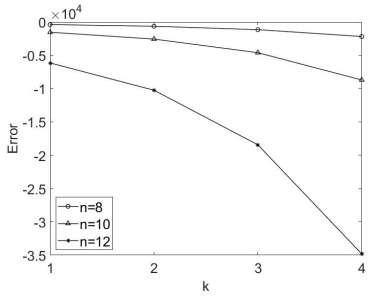


Fig. 17. Average horizontal truncation error versus k for different values of n for multiplication with padding.

VI. NON-RESTORING DIVIDER

Two of the most commonly used division algorithms are restoring and non-restoring; they are based on shift, subtraction and addition operations in array dividers[20][21]. In these array dividers, replicated units are used for the parallel divider; similar units are used for comparison of the partial remainder and divisor, while the shift operation is mostly implemented by wiring.

Consider integer division, the operands are given by the dividend A and the non-zero divisor D; the results of the operation are the quotient Q and the remainder R, i.e.,

$$A = D \times Q + R \quad (14)$$

where the sign of the remainder R is the same as the dividend A and |R|< |D|. A widely used algorithm for division is the so-called non-restoring division.

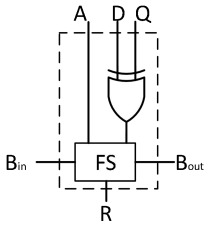


Fig. 18. Exact non-restoring array divider cell (ExDCnr).

In this paper, non-restoring division is considered, because it offers higher performance than the restoring one, i.e. in non-restoring division, restoration is not required. In this algorithm, the only operations are either addition or subtraction. The non-restoring array division algorithm (like the restoring one) is based on a modular architecture. The exact non-restoring array divider cell (ExDCnr) is shown in Fig. 18; ExDCnr is made up of a full subtractor and a XOR gate, it performs addition and subtraction according to the sign of the partial remainder in the previous row. Each row consists of several cells and by connecting these rows and back each of them to the next rows. The 8-by-4 non-restoring array divider is shown in Fig. 19[22]. It computes the integer division for A[7:0], D[3:0], Q[3:0] and R[3:0]. Truncation consists of fully removing at least a cell, this process is shown by the shaded cells in Fig. 20. The input A of the removed cell is left unchanged and moved downwards in the remainder output

direction, while the input D of the removed cell is discarded.

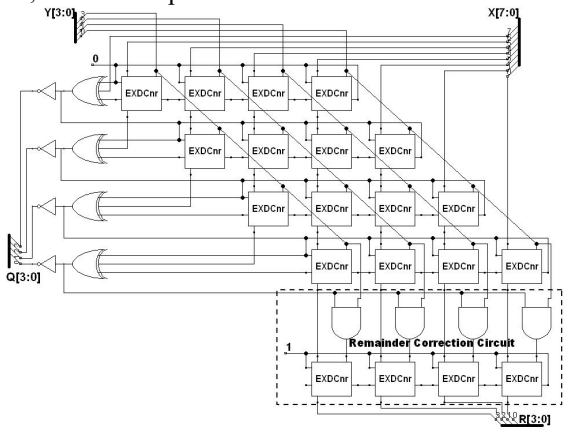


Fig. 19. Exact non-restoring array divider (8-by-4) [22].

Error generation and propagation can be qualitatively analyzed by the location of the replaced or truncated cells in the divider. An approximate scheme consists of replacing or truncating only a portion of the divider; this leads to different approximate configurations with different accuracy for the output values (Q and R). In a divider, the dividend A and the divisor D are provided as inputs at the north side, while the quotient Q and the reminder R are generated at the west and the south sides respectively (Fig. 19). Therefore, each cell (located at a unique position in the divider) plays a different role in generating the error.

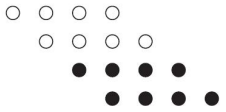


Fig. 20. Example of truncation for 8-to-4 bit divider.

So, for a general analysis, consider a 2N-N divider under the assumption that the truncation level is K,

$$\begin{aligned} A &= D \times Q + R \\ &= D \times Q^{'} + D \times Q^{''} + R \end{aligned} \qquad (15)$$

where Q' represents the N-k bits (MSB) of Q (as output of the array divider), Q'' represents the K bits (LSB) of Q. In the exact case, $A - R_{ex} = D \times Q^{'} + D \times Q^{''}$. In the truncated case, Q'' has a value of 0 (without adding the padding value), so $A - R_{ex} = D \times Q^{'} + \text{Error (from (15))}$.

For signed division, only the quotient is considered, because in most applications only the quotient is needed. Thus,

$$\text{Error} = D \times Q^{''} \qquad (16)$$

Error is a function of k, i.e. it can be $\{D \times 0, D \times 1, D \times 2,...,D \times (2^K - 1)\}$. If A and D follow a uniform distribution, then Error will also follow a uniform distribution; so, the average value of Error is $(2^k - 1)/2 \times D$ and the error distance for the quotient is given by $(2^k - 1)/2$. Under a uniform distribution, a 2^{k-1} padding value is then added to the quotient to reduce the average signed error.

TABLE VI shows that for division, the average signed error is reduced using padding. It is not zero as for addition; such error is now constant at a value of -0.5.

TABLE VI
AVERAGE SIGNED ERROR (EXACT-INEXACT) FOR TRUNCATION WITH/WITHOUT PADDING FOR QUOTIENT

Padding	No Padding	With Padding
'1' (k=1)	0.5	-0.5
'10' (k=2)	1.5	-0.5
'100' (k=3)	3.5	-0.5
'1000' (k=4)	7.5	-0.5

VII. COMPARISON

A. Multiplier

In this section, 8-bit and 16-bit approximate multipliers are compared. [23] has proposed radix-4 approximate Booth multipliers with accuracy that can be adjusted by utilizing a so-called approximate factor. [24] has proposed fixed-width truncated Booth multipliers with a rather simple error compensation. Also, for these designs, they are implemented in Verilog and synthesized by the Synopsys Design Compiler using the FreePDK 45 nm Library. The two input values to the multiplier are randomly generated by a uniform distribution; simulations consist of a sample of 1 million.

TABLE VII
AREA, POWER, DELAY, ASE AND APE PRODUCT FOR DIFFERENT MULTIPLIER TRUNCATION SCHEMES

Multiplier Type	Area (μm^2)	Delay (ns)	Power (μw)	ASE	NMED ($\times 10^{-3}$)	APE Product
N=8						
Exact Multiplier	757.8	0.75	163.4	-	-	-
Proposed k=2 (V)	624.0	0.63	144.9	-0.29	1.1	26221
Proposed k=2 (H)	601.4	0.62	142.3	-0.74	12.6	63329
Proposed k=4 (V)	500.5	0.59	108.9	-0.38	2.1	20712
Proposed k=4 (H)	451.7	0.61	96.2	-0.41	13.5	17816
Radix-4 Multiplier k=2 [23]	639.9	0.64	151.8	-0.27	0.9	26227
Radix-4 Multiplier k=4 [23]	592.5	0.64	140.1	-0.39	1.5	32374
Fix-width Multiplier [24]	593.2	0.73	137.9	-1.24	6.7	101435
N=16						
Exact Multiplier	2607.4	1.23	618.6	-	-	-
Proposed k=4 (V)	2441.6	1.14	594.6	-1.24	1.4	1800201
Proposed k=4 (H)	2406.8	1.13	578.1	-1.47	11.5	2045315
Proposed k=8 (V)	1999.0	1.09	484.7	-2.85	3.8	2761409
Proposed k=8 (H)	1871.7	1.09	464.2	-2.93	13.3	2545710
Radix-4 Multiplier k=4 [23]	2585.0	1.14	594.8	-0.96	1.2	1476056
Radix-4 Multiplier k=8 [23]	2462.2	1.12	539.5	-2.11	2.9	2802833
Fix-width Multiplier [24]	2304.4	1.18	529.9	-3.25	7.8	3968580

V for Vertical truncation; H for Horizontal truncation.

Power consumption, delay, area, NMED[25] and Average Signed Error (ASE) are evaluated and reported in TABLE VII. In order to compare the different approximation schemes, a metric called APE (Area, Power and Error) Product is used. It is the absolute value of the product of the area, power consumption and Average Signed Error. The smaller product value indicates the better performance in the combinational dimensions.

As shown in TABLE VII, the proposed multiplier incurs in an area and power consumption smaller than [23]; however, its accuracy is marginally worse than [23]. The fix-width multiplier in [24] has the least area and power consumption, but it is also the least accurate multiplier among the three schemes. In the term of the APE Product, generally, the proposed scheme has the least APE product among the multiplier in [23] and [24], one exception is configuration of horizontal truncation N=8 and k=2. The compensation padding added to this configuration results a large ASE which leads the worse APE Product value. In the term of the NMED metric, the horizontal truncation padding scheme has the largest NMED. This is because the padding added to the horizontal truncation has larger absolute value which lead a larger error distance compared to the vertical truncation padding.

B. Divider

In [22], several approximate dividers have been proposed; at array level, exact cells are either replaced by inexact cells, or truncated in the approximate divider designs. For all designs, all metrics are generated using the Synopsys Design Compiler with the FreePDK 45 nm Library. The input dividend and divisor are randomly generated using simulation with a uniform distribution and 1 million iterations. The results of area, power, delay and APE Product are shown in TABLE VIII.

TABLE VIII
AREA, POWER, DELAY, ASE AND APE PRODUCT FOR DIFFERENT DIVIDER TRUNCATION SCHEMES

Multiplier Type	Area (μm^2)	Delay (ns)	Power (μw)	ASE	NMED ($\times 10^{-2}$)	APE Product
N=8						
Exact Divider	757.8	0.75	163.4	-	-	-
Proposed k=2	697.4	0.70	184.9	-0.54	3.4	69633
Proposed k=4	514.1	0.69	145.7	-0.58	6.1	43445
Divider in [22] Truncation k=2	663.5	0.67	179.4	0.64	8.6	76180
Divider in [22] Replacement k=2	812.4	0.79	197.6	0.47	2.5	75449
Divider in [22] Truncation k=4	498.1	0.66	138.5	7.35	11.6	507053
Divider in [22] Replacement k=4	793.5	0.78	189.7	0.51	5.3	76769
N=16						
Exact Divider				-	-	-
Proposed k=4	2502.3	1.32	618.6	-0.80	4.2	1238338
Proposed k=8	1685.4	1.20	503.7	-1.28	6.8	1086638
Divider in [22] Truncation k=4	2397.2	1.30	601.9	3.51	9.7	5064490
Divider in [22] Replacement k=4	3195.1	1.44	706.5	0.61	2.9	1376976
Divider in [22] Truncation k=8	1504.4	1.20	491.1	10.4	13.8	7683633
Divider in [22] Replacement k=8	2962.7	1.43	674.2	0.88	3.3	1757758

In TABLE VIII, the proposed scheme incurs in less power consumption, area and delay [22]; however, the accuracy of the proposed scheme is marginal worse than the replacement policy of [22]. For the truncated policy of [22], power consumption, area and delay are better than the proposed scheme; however, the accuracy of proposed scheme is significantly improved compared to [22]. For the APE Product term, the proposed scheme has the least APE Product value compared with the dividers of replacement and truncation policies in [22]. The truncation policy in [22] produces significant ASE value which lead a large APE Product.

VIII. APPLICATIONS

In this section, three applications of the proposed schemes are presented; they utilize compensation with the different arithmetic operations analyzed in the previous section.

A. Matrix Multiplication

Matrix multiplication (MM) is one of the most important operations in different fields of science, engineering and technology, as utilized in signal and image processing, system theory, statistical and numerical analysis. [26][27] Consider the multiplication of two N×N matrices A and B, so

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1N} \\ P_{21} & P_{22} & \dots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \dots & P_{NN} \end{bmatrix} = A \times B$$
$$= \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1N} \\ B_{21} & B_{22} & \dots & B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N1} & B_{N2} & \dots & B_{NN} \end{bmatrix} \quad (17)$$

where

$$P_{ij} = \sum_{k=1}^{N} A_{ik} \cdot B_{kj}$$

The cumulative sum(i.e.$\sum_{k=1}^{N} A_{ik} \cdot B_{kj}$) must be calculated; so, a processing unit needs to have two functions: (1) calculate the product of the two binary input operands ($A_{ik}$and B_{kj}); (2) calculate the cumulative sum of the products. Therefore, in the execution of this algorithm, a processing unit consists of an adder and a multiplier with signed operands.

Thus, for the processing unit of matrix multiplication, its functionality can be expressed by the following equation:

$$S_{out} = A \times B + S_{in} \quad (18)$$

To implement an approximate processing unit of the matrix multiplication, an inexact adder and inexact multiplier are employed. For simulation, the inputs A, B and S_{in} are 16-bit values; they are randomly generated, and 1 million simulation runs are used.

Fig. 21 shows the average signed error (ASE) ratio of the MM unit at different truncation levels (on the x axis the number

of truncated bits). The ratio is defined as:

$$\text{ASE ratio} = \frac{|\text{ASE with padding}|}{|\text{ASE without padding}|} \qquad (19)$$

The ratio is the absolute value of the average signed error with padding versus the average signed error without padding, i.e. when the ratio is smaller, compensation is more effective.

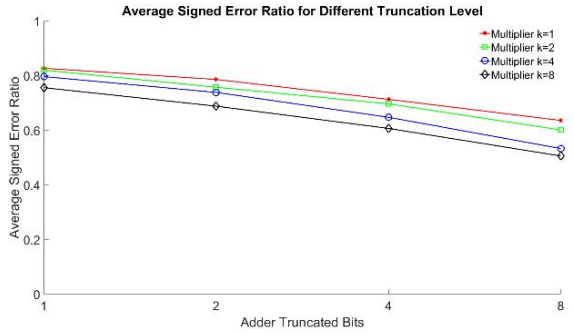


Fig. 21. Average signed error (ASE) ratio for different truncation level of MM.

The following conclusions can be drawn from Fig. 20.

1) The ASE ratio decreases at a larger truncated adder (so improving the effectiveness of the proposed scheme);

2) At the same adder truncation level, the ratio is inversely correlated with the truncation level of the multiplier.

When the truncation level is high and compensation is used, the average error is significantly reduced compared to the case when no output compensation is used.

TABLE IX lists the evaluation results for matrix multiplication using three approximate multipliers. The power, area, delay and ASE are measured.

TABLE IX
AREA, POWER, DELAY AND ASE FOR DIFFERENT MULTIPLIER TRUNCATION SCHEMES (N=16, ADDER TRUNCATION LEVEL=8)

Multiplier Type	Area (μm^2)	Delay (ns)	Power (μw)	ASE	NMED ($\times 10^{-3}$)
Proposed k=4 (V)	2770.8	1.59	661.2	-6.81	6.8
Proposed k=4 (H)	2735.0	1.58	644.5	-10.68	7.2
Proposed k=8 (V)	2315.0	1.54	550.3	-15.35	12.8
Proposed k=8 (H)	2183.9	1.54	529.6	-15.77	13.6
Radix-4 multiplier k=4 [23]	2918.6	1.59	661.4	-4.80	5.4
Radix-4 multiplier k=8 [23]	2792.1	1.57	605.6	-8.99	10.7
Fix-width multiplier [24]	2629.5	1.63	595.9	-17.47	19.1

V for Vertical truncation; H for Horizontal truncation.

In this evaluation, the truncation level of the approximate adder is fixed to 8 bits. Three multiplier implementations are used for comparison; for the proposed multiplier, the power consumption and area are the smallest among the three implementations. The Radix-4 multiplier is the most accurate implementation, but it has the largest values for power dissipation and area. The fix-width multiplier's accuracy is the worst among the three considered multipliers. The cascading arrangement of an approximate multiplier followed by an adder results in an increase of the error compared to a scheme with only a multiplier (as shown in TABLE VII). However, when the exact adder is utilized in this evaluation, the error has a at similar value as reported previously in TABLE VII.

B. Image Processing: Changing Detection

In this section, the proposed approximate scheme for division is evaluated for the changing detection application in image processing. The approximate divider with padding is assessed for pixel division. For image analysis, if only integer division is performed, then the results are typically rounded at the output to the next lowest integer. A 16-to-8 approximate divider is used to compute the inputs X and Y of a 8-bit grayscale images. The fractional change or ratio between corresponding pixel values is then calculated. If there is no movement between two frames, then the output image mostly consists of single-value pixels. If there is movement, then the pixels in those regions of the image in which the intensity spatially changes, will exhibit significant differences between the two frames. To evaluate changing detection by division, a series of 8-bit images are used; these eight images are from the publicly available database of [28]. Then, the average Peak Signal Noise Ratio (PSNR) and Structural Similarity Index (SSIM) of approximate result with exact result are measured for all permutations among the 8 images at different values of k (56 pairs in total). Based on the circuit metric reported in section VII, the figure of merit including power and image quality are also calculated. It is defined as PPS Product (PSNR × SSIM × Power Consumption^{-1}). The structure with larger PPS product value indicates that it has the better combination FOM. The results are reported in TABLE X and example images are shown in Fig. 22.

Fig. 22. An example of changing detection (N=16, k=1).

TABLE X
AVERAGE PSNR FOR 2N-N DIVIDER AT DIFFERENT TRUNCATION LEVEL FOR CHANGING DIRECTION APPLICATION, N=8

Divider Type		k=1	k=2	k=4
Proposed scheme	PSNR	48.18dB	45.87dB	43.39dB
	SSIM	0.9921	0.9906	0.9819
	PPS	**0.237**	**0.246**	**0.292**
Approximate divider truncation k=8 [22]	PSNR	44.93dB	41.87dB	38.78dB
	SSIM	0.9875	0.9830	0.9714
	PPS	0.226	0.229	0.272
Approximate divider replacement k=8 [22]	PSNR	**49.26dB**	**46.25dB**	**44.58dB**
	SSIM	**0.9925**	**0.9911**	**0.9825**
	PPS	0.232	0.232	0.231

Compared to the approximate divider with truncation of [22], the accuracy of the proposed divider is significantly improved; however, it is slightly worse than a replacement policy. However, TABLE VIII reveals that the power consumption of the proposed scheme is significantly reduced compared to the replacement policy of [22]. Moreover, the PPS product shows that the proposed scheme is the best among the three considered approximate dividers.

IX. Conclusion

This paper has presented a general scheme for error compensation for approximate computing operations using arithmetic circuits. The proposed schemes utilize output compensation in adders, multipliers and dividers when signed integer operations are executed. Following input truncation of the LSBs, padding is added to reduce the average (signed) error based on statistical information of these arithmetic operations; under a normal distribution, it has been shown that the distributions of these operations in the exact case are symmetric.

The reduction in error by compensation for these 3 arithmetic operations is proved analytically under the uniform distribution case. Then, application of this scheme to matrix multiplication and image procession has shown that the proposed scheme is very efficient and versatile. In all considered cases, the improvement in error metrics is substantial, for example for image processing (changing direction), the results show that a reduction of 20 to 40 to the Mean Square Error (MSE) can be achieved with only a marginal impact on the division implementation in terms of delay and hardware overhead.

References

[1] W. Liu, A. Nannarelli, "Power Efficient Division and Square Root Unit", IEEE Transactions on Computers, vol. 61, no. 8, pp. 1059-1070, 2012.

[2] L. Leem, H. Cho, J. Bau, Q.A. Jacobson, S. Mitra, "ERSA: Error resilient system architecture for probabilistic applications", Proc.Design Automation and Test in Europe (DATE), pp. 1560-1565, 2010.

[3] W. Liu, F. Lombardi, and M. Schulte, "A Retrospective and Prospective View of Approximate Computing", Proceedings of the IEEE, vol. 108, no. 3, pp. 394-399, Mar. 2020.

[4] V. Gupta, D. Mohapatra, P. Sang Phill, A. Raghunathan, K. Roy, "IPACT: IMPrecise adders for low-power approximate computing", Proc. International Symposium on Low Power Electronics and Design (ISLPED), pp. 409-414, 2011.

[5] V. Gupta, D. Mohapatra, A. Raghunathan, K. Roy, "Low-power digital signal processing using approximate adders", IEEE Trans. Comput-Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124-137, 2013.

[6] Z. Yang, A. Jain, J. Liang, J. Han and F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," Proc. 13th IEEE International Conference on Nanotechnology (IEEE-NANO 2013), Beijing, 2013, pp. 690-693.

[7] H. R, Mahdiani, A. Ahmadi, S. M. Fakhraie, C. Lucas, "Bio-inspircd imprecise computational blocks for efficient VLSI implementation of soft-computing applications", IEEE Trans. Circuits Syst. I, vol. 57, no. 4, pp. 850-862, 2010.

[8] S.-L. Lu, "Speeding up processing with approximation circuits," Computer, vol. 37, no. 3, pp. 67–73, Mar. 2004.

[9] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

[10] K. Khaing Yin, G. Wang-Ling, Y. Kiat-Seng, "Low-power high-speed multiplier for error-tolerant application", Proc. IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC), pp. 1-4, 2010.

[11] F. Sabetzadeh, M. H. Moaiyeri and M. Ahmadinejad, "A Majority-Based Imprecise Multiplier for Ultra-Efficient Approximate Image Multiplication," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 11, pp. 4200-4208, Nov. 2019.

[12] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 10, pp. 3105–3117, Oct. 2016.

[13] S. Venkatachalam, H. J. Lee and S. Ko, "Power Efficient Approximate Booth Multiplier," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, 2018, pp. 1-4.

[14] S. Doochul, S. K. Gupta, "A new circuit simplification method for error tolerant applications", Proc. Design Automation and Test in Europe (DATE), pp. 1-6, 2011.

[15] S. Venkataramani, V. Kozhikkottu, A. Sabne, K. Roy and A. Raghunathan, "Logic Synthesis of Approximate Circuits," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[16] A. Sinha, A. Wang, A. P. Chandrakasan, "Algorithmic transforms for efficient energy scalable computation", Proc. International Symposium on Low Power Electronics and Design (ISLPED), pp. 31-36, 2000.

[17] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, S. T. Chakradhar, "Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency", Proc. 47th ACM/IEEE: Design Automation Conference (DAC), pp. 555-560, 2010.

[18] P. Jongsun, C. Jung-Hwan, K. Roy, "Dynamic bit-width adaptation in DCT: An approach to trade off image quality and computation energy", IEEE Trans. VLSI Syst., vol. 18, no. 5, pp. 787-793, 2010.

[19] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," Proc. 18th IEEE European Test Symposium (ETS), Avignon, 2013, pp. 1-6.

[20] S. R. Chowdhury, A. Banerjee, A. Roy, and H. Saha, "A high speed 8 transistor full adder design using novel 3 transistor XOR gates," Int. J. Electron., Circuits Syst., vol. 2, pp. 217–223, 2008.

[21] S. Vahdat, M. Kamal, A. Afzali-Kusha, M. Pedram and Z. Navabi, "TruncApp: A truncation-based approximate divider for energy efficient DSP applications," Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), , 2017, pp. 1635-1638.

[22] L. Chen, J. Han, W. Liu and F. Lombardi, "On the Design of Approximate Restoring Dividers for Error-Tolerant Applications," IEEE Transactions on Computers, vol. 65, no. 8, pp. 2522-2533, Aug. 1, 2016.

[23] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate Radix-4 Booth multipliers for error-tolerant computing," IEEE Trans. Comput., vol. 66, no. 8, pp. 1435–1441, Aug. 2017.

[24] 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. Syst., vol. 12, no. 5, pp. 522–531, May 2004.

[25] J. Liang, J. Han and F. Lombardi, "New Metrics for the Reliability of Approximate and Probabilistic Adders," IEEE Transactions on Computers, vol. 62, no. 9, pp. 1760-1771, Sept. 2014

[26] M. Stojčev, E. Milovanović, S. Marković, and I. Milovanović, "Synthesis of orthogonal systolic arrays for fault-tolerant matrix multiplication," in Proc. 27th International Conference on Microelectronics Proceedings (MIEL), pp. 327-334, 2010

[27] P. Saha, A. Banerjee, P. Bhattacharyya, and A. Dandapat, "Improved matrix multiplier design for high-speed digital signal processing applications," IET Circuits, Devices & Systems, vol. 8, no.1, pp. 27-37, Jan. 2014

[28] The USC-SIPI Image Database — A large collection of standard test images: http://sipi.usc.edu/database/.

Ke Chen received the B.Sc. degree in Engineering from Huazhong University of Science & Technology (HUST), Wuhan, China, in 2010, and the MS degree from Northeastern University, Boston, in 2012. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering, Northeastern University, Boston. His research interests include low-power and high-performance VLSI design, emerging logic and memory devices and circuits, inexact and fault tolerant computing. He is a student member of the IEEE.

Weiqiang Liu is currently an Associate Professor and the Vice Dean of College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China. He received the B.Sc. degree in Information Engineering from NUAA and the Ph.D. degree in Electronic Engineering from Queen's University Belfast (QUB), Belfast, United Kingdom, in 2006 and 2012, respectively. He has served as Associate Editors of IEEE Transactions on Circuits and Systems I: Regular Papers, IEEE Transactions on Emerging Topic in Computing and Computers, IEEE Transactions on Computers, a Guest Editor of Proceedings of the IEEE, and a Steering Committee Member of IEEE Transactions on Multi-Scale Computing Systems. He is the Program Co-Chair of IEEE Symposium on Computer Arithmetic (ARITH 2020), and program members for a number of international conferences including ARITH, DATE, ASP-DAC, ISCAS, ASAP, ISVLSI, GLSVLSI, AsiaHOST, NANORACH, AICAS, SiPS and ICONIP. He is a member of both Circuits & Systems for Communications (CASCOM) Technical Committee and VLSI Systems and Applications (VSA) Technical Committee, IEEE Circuits and Systems Society. He has published one research book by Artech House and over 90 leading journal and conference papers. His research interests include approximate computing, computer arithmetic, hardware security and VLSI design for digital signal processing and cryptography.

Jie Han received the B.Sc. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree from the Delft University of Technology, The Netherlands, in 2004. He is currently an Associate Professor in the Department of Electrical and Computer Engineering at the University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computing, reliability and fault tolerance, nanoelectronic circuits and systems, novel computational models for nanoscale and biological applications. Dr. Han was a recipient of the Best Paper Award at the International Symposium on Nanoscale Architectures (NanoArch) 2015 and Best Paper Nominations at the 25th Great Lakes Symposium on VLSI (GLSVLSI) 2015, NanoArch 2016 and the 19th International Symposium on Quality Electronic Design (ISQED) 2018. He was nominated for the 2006 Christiaan Huygens Prize of Science by the Royal Dutch Academy of Science. His work was recognized by Science, for developing a theory of fault-tolerant nanocircuits (2005). He is currently an Associate Editor for the IEEE Transactions on Emerging Topics in Computing (TETC), the IEEE Transactions on Nanotechnology, the IEEE Circuits and Systems Magazine and Microelectronics Reliability (Elsevier Journal). He served as a General Chair for GLSVLSI 2017 and the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT) 2013, and a Technical Program Committee Chair for GLSVLSI 2016, DFT 2012 and the Symposium on Stochastic & Approximate Computing for Signal Processing and Machine Learning, 2017.

Fabrizio Lombardi (M'81-SM'02-F'09) received the B.Sc. degree (Hons.) in electronic engineering from the University of Essex, U.K., in 1977, the master's degree in microwaves and modern optics and the Diploma degree in microwave engineering from the Microwave Research Unit, University College London, in 1978, and the Ph.D. degree from the University of London in 1982. He is currently the International Test Conference (ITC) Endowed Chair Professorship with Northeastern University, Boston, USA. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems. He has extensively published in these areas and coauthored/edited seven books.