A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery

Cong Liu
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, Alberta, Canada T6G 2V4
Email: cong4@ualberta.ca

Jie Han
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, Alberta, Canada T6G 2V4
Email: jhan8@ualberta.ca

Fabrizio Lombardi
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115
Email: lombardi@ece.neu.edu

***Abstract*—Approximate circuits have been considered for error-tolerant applications that can tolerate some loss of accuracy with improved performance and energy efficiency. Multipliers are key arithmetic circuits in many such applications such as digital signal processing (DSP). In this paper, a novel approximate multiplier with a lower power consumption and a shorter critical path than traditional multipliers is proposed for high-performance DSP applications. This multiplier leverages a newly-designed approximate adder that limits its carry propagation to the nearest neighbors for fast partial product accumulation. Different levels of accuracy can be achieved through a configurable error recovery by using different numbers of most significant bits (MSBs) for error reduction. The approximate multiplier has a low mean error distance, i.e., most of the errors are not significant in magnitude. Compared to the Wallace multiplier, a 16-bit approximate multiplier implemented in a 28nm CMOS process shows a reduction in delay and power of 20% and up to 69%, respectively. It is shown that by utilizing an appropriate error recovery, the proposed approximate multiplier achieves similar processing accuracy as traditional exact multipliers but with significant improvements in power and performance.**

I. Introduction

Approximate computing has emerged as a potential solution for the design of energy-efficient digital systems [1]. Applications such as multimedia, recognition and data mining are inherently error-tolerant and do not require a perfect accuracy in computation. For these applications, approximate circuits may play an important role as a promising alternative for reducing area, power and delay in digital systems that can tolerate some loss of accuracy, thereby achieving better performance in energy efficiency.

As one of the key components in arithmetic circuits, adders have been extensively studied for approximate implementation (see [1] for a review). New methodologies to model, analyze and evaluate the approximate adders have been discussed in [2]–[4]. However, there has been relatively less effort in the design of approximate multipliers. A multiplier usually consists of three stages: partial product generation, partial product accumulation and a carry propagation adder (CPA) at the final stage. [5] considers using approximate adders to generate the radix-8 Booth encoding 3x with error reduction. In [6], approximate partial products are computed using inaccurate 2×2 multiplier blocks, while accurate adders are used in an adder tree to accumulate the approximate partial products. [2] briefly discusses the use of approximate speculative adders for the final stage addition in a multiplier. The error tolerant multiplier (ETM) of [7] is based on the truncation of a multiplier into an accurate multiplication part for MSBs and a non-multiplication part for LSBs.

In this paper, a novel approximate multiplier design is proposed using a simple, yet fast approximate adder. This newly designed adder can process data in parallel by cutting the carry propagation chain (and thus, introducing an error). It has a critical path delay that is even shorter than a conventional one-bit full adder. Albeit having a high error rate, this adder simultaneously computes the sum and generates an error signal; this feature is employed to reduce the error in the final result of the multiplier. In the proposed approximate multiplier, a simple tree of the approximate adders is used for partial product accumulation and the error signals are used to compensate the error for obtaining a better accuracy. Compared to the traditional (exact) Wallace and Dadda trees, the proposed multiplier has a significantly shorter critical path as well as a reduced circuit complexity.

II. Proposed Approximate Adder

In this section, the design of a new approximate adder is presented. This adder operates on a set of pre-processed inputs. The input pre-processing (IPP) is based on the *interchangeability* of bits with the same weights in different addends. For example, consider two sets of inputs to a 4-bit adder: i) $A = 1010, B = 0101$ and ii) $A = 1111, B = 0000$. Clearly, the additions of i) and ii) produce the same result. In this process, the two input bits $A_iB_i = 01$ are equivalent to $A_iB_i = 10$ (with i being the bit index), because of the interchangeability of the corresponding bits in the two operands.

The basic rule for the IPP is to switch A_i and B_i if $A_i = 0$ and $B_i = 1$ (for any i), while keeping the other combinations (i.e., $A_iB_i = 00, 10$ and 11) unchanged. By doing so, more 1's are expected in A and more 0's are expected in B. If $\dot{A}_i\dot{B}_i$ are the ith bits in the pre-processed inputs, the IPP functions are given by:

$$\dot{A}_i = A_i + B_i, \tag{1}$$

$$\dot{B}_i = A_iB_i. \tag{2}$$

(1) and (2) compute the propagate and generate signals used in a parallel adder such as the carry look-ahead (CLA). The

978-3-9815370-2-4/DATE14/2014 EDAA

TABLE I. TRUTH TABLE OF AN APPROXIMATE ADDER CELL.

$\dot{B}_i\dot{B}_{i-1}$	00	01	10	11
$\dot{A}_i$	$\dot{A}_i$	$\dot{A}_i$	1	1
$C_{i-1}/\dot{B}_{i-1}$	0	1	0	1
S_i	$\dot{A}_i$	1	0	1
E_i	0	$\dot{A}_i$	0	0

proposed adder can process data in parallel by cutting the carry propagation chain. A carry propagation chain starts at the ith bit when $\dot{B}_i = 1, \dot{A}_{i+1} = 1, \dot{B}_{i+1} = 0$. In an accurate adder, S_{i+1} is 0 and the carry propagates to the higher bit. However, in the proposed approximate adder, S_{i+1} is set to be 1 and an error signal is generated as $E_{i+1} = 1$. This prevents the carry signal from propagating to higher bits. By doing so, a carry signal is produced only by the generate signal, i.e. $C_i = 1$ only when $\dot{B}_i = 1$, and it only propagates to the next higher bit, i.e. the (i+1)th position. Table I shows the truth table of the approximate adder, where $\dot{A}_i, \dot{B}_i$ and $\dot{B}_{i-1}$ are the inputs after IPP, C_{i-1} is the carry signal, S_i and E_i are the sum and error bits, respectively. The error signal is utilized for error compensation purposes as discussed in a later section. In this case, the approximate adder is similar to a redundant number system [8] and the logical functions of Table I are given by

$$S_i = \dot{B}_{i-1} + \overline{\dot{B}_i}\dot{A}_i, \tag{3}$$

$$E_i = \overline{\dot{B}_i}\dot{B}_{i-1}\dot{A}_i. \tag{4}$$

Replacing $\dot{A}, \dot{B}$ using (1) and (2), the logic functions with respect to the original inputs are given by

$$S_i = (A_i \oplus B_i) + A_{i-1}B_{i-1}, \tag{5}$$

$$E_i = (A_i \oplus B_i)A_{i-1}B_{i-1}. \tag{6}$$

Consider as an example a 6-bit adder with two inputs given by $A = 001111$ and $B = 000110$. The correct (exact) sum S is 010101; however, the approximate adder produces the sum $S^{'} = 001101$ and an error $E = 001000$. It is easy to show that

$$S = S^{'} + E. \tag{7}$$

Note that in (7) '+' means the addition of two binary numbers rather than the 'OR' function. The error E is always non-negative and the approximate sum is always equal to or smaller than the accurate sum. This is an important feature of this adder, because an additional adder can be used to add the error to the approximate sum as a compensation step.

III. PROPOSED APPROXIMATE MULTIPLIER

In the proposed approximate multiplier, an adder tree is utilized for partial product accumulation; the error signals in the tree are then used to compensate the error in the output to obtain a product with a better accuracy.

A. Partial Product Accumulation

A significant feature of the proposed approximate multiplier is the simplicity to use approximate adders in the partial product accumulation. [6] has shown that this may lead to poor performance, because errors may accumulate and it is difficult to correct errors using existing approximate adders. However, the use of the newly proposed approximate adder overcomes this problem by utilizing the error signal. The resulting design has a critical path delay that is shorter than a conventional one-bit full adder, because the new n-bit adder can process data in parallel.

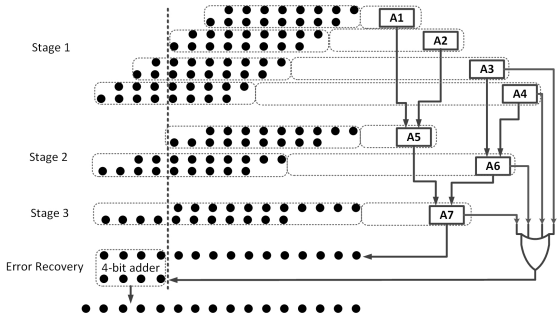

Fig. 1. An approximate multiplier with OR-gate based partial error recovery using 4 MSBs of the error vector.

B. Error Reduction

As (7) is applicable to the sum of every single approximate adder in the tree, an error reduction circuit is applied to the final multiplication result rather than to the output of each adder. Two steps are required to reduce errors: i) error accumulation and ii) error recovery by the addition of the accumulated errors to the adder tree output using an accurate adder (Fig. 1).

1) Error Accumulation: The error signals can be summed up using accurate adders and thus, the accumulated error can fully compensate the inaccurate product; however to reduce complexity, an approximate error accumulation is introduced. Consider the observation that the error vector of each approximate adder tends to have more 0's than 1's. Therefore, the probability that the error vectors have an error bit '1' at the same position, is quite small. Hence, an OR gate is used to approximately compute the sum of the errors for a single bit. If m error vectors (denoted by $E1, E2, ..., Em$) have to be accumulated, the sum of these vectors is obtained as

$$E_i = E1_i \; OR \; E2_i \; OR \; ... \; OR \; Em_i. \tag{8}$$

2) Error Recovery: To reduce the error, an accumulated error vector is added to the adder tree output using a conventional adder (e.g. a carry look-ahead adder). However, only several (e.g. k) MSBs of the error signals are used to compensate the outputs for further reducing the overall complexity. The number of MSBs is selected according to the extent that errors must be compensated. For example in an 8×8 adder tree, there are a total of 7 error vectors, generated by the 7 approximate adders in the tree. However, not all the bits in the 7 vectors need to be added, because the MSBs of some vectors are less significant than the least significant bits of the k MSBs. In the example of Fig. 1, 4 MSBs (i.e. the 11-14th bits) are considered for error recovery and as a result, 4 error vectors are considered (i.e. the error vectors of adders A3, A4, A6 and A7). Note that the error vectors of the other three adders are less significant than the 11th bit, so they are not considered. The accumulated error E is obtained using (8); then, the final result is found by adding E to S using a fast accurate adder. The adder tree and the error reduction scheme are shown in Fig. 1.

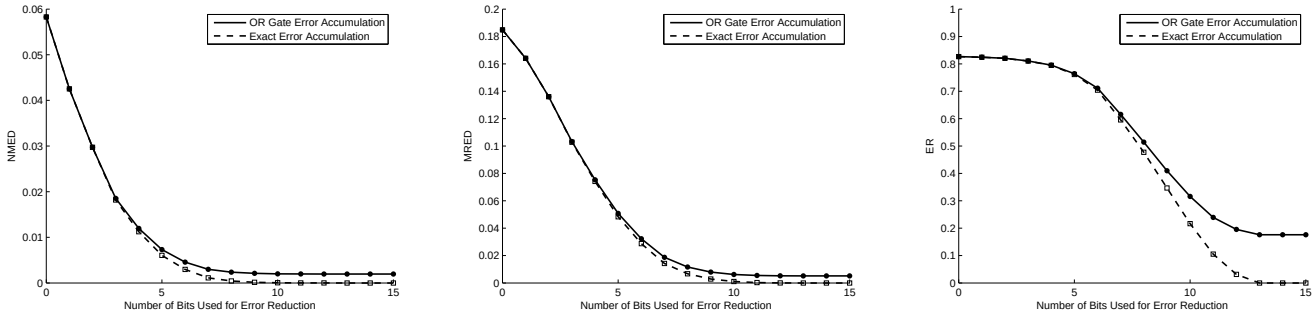

Fig. 2. Accuracy comparison of the approximate multiplier using OR-gate and exact error accumulation: (a) NMED (b) MRED (c) ER vs. different number of bits for error reduction.

IV. Accuracy Evaluation

In [4], the error distance (ED) and mean error distance (MED) are proposed to evaluate the performance of approximate arithmetic circuits. For multipliers, ED is defined to be the arithmetic difference between the accurate product (M) and the approximate product ($M^{'}$), i.e., $ED = |M^{'} - M|$. MED is the average of EDs for a set of outputs (obtained by applying a set of inputs). A metric for comparing multipliers of different sizes is the normalized MED (NMED): $NMED = \frac{MED}{M_{max}}$, where M_{max} is the maximum magnitude of the output of an (accurate) multiplier, i.e. $(2^n - 1)^2$ for an $n \times n$ multiplier. The relative error distance (RED) is defined as: $RED = \frac{|M^{'} - M|}{M} = \frac{ED}{M}$. Similarly, the mean relative error distance (MRED) can be obtained. The error rate (ER) is defined as the percentage of erroneous outputs among all outputs. These three metrics (NMED, MRED and ER) are used to evaluate the proposed multiplier. A functional model of the proposed multiplier is implemented using Matlab and an exhaustive simulation is performed for an 8×8 approximate multiplier.

Both the OR gate error accumulation and the exact error accumulation are considered for the proposed multiplier; Fig. 2 shows the three metrics (NMED, MRED and ER) for using different numbers of MSBs for error reduction. Let m denote the number of MSBs used for error reduction. It can be seen that the NMED and MRED drop drastically as m is increased from 0 to 6 and continue to drop as m increases, even though at a slower rate. The ER also decreases as m is increased. For the approximate multiplier, there is no error in the most significant bit of the output, so the largest number of MSBs used is 15. It is also shown that the OR gate error accumulation produces a good approximation to the exact error accumulation. Therefore, m=6 or m=7 may be appropriate for a good trade-off in terms of the NMED and MRED. For m=7, the NMED is below 0.3% and the MRED is approximately 1.8%. However, the error rate is reduced significantly as m increases; it decreases to 20% when m=12 for OR gate error accumulation. These three figures indicate that the proposed approximate multiplier has a rather high error rate, but the error is usually very small compared to both the accurate and the largest possible output of the approximate multiplier. For example, for m=7, the error rate can be as high as 62%, but the MRED is only 1.8%, i.e., most of the errors are not significant.

TABLE II. Arithmetic accuracy comparison between three approximate multipliers.

	Proposed multiplier	ETM [7]	2×2 approximate multiplier [6]
NMED (%)	0.20	2.85	1.39
MRED (%)	0.62	25.21	3.25
ER (%)	31.59	98.88	46.73

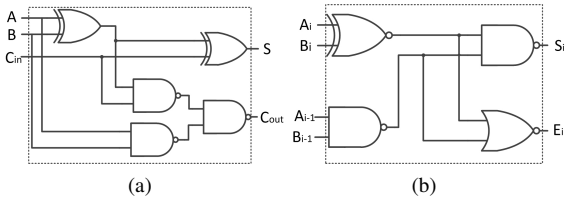

Fig. 3. (a) An exact full adder and (b) the approximate adder cell.

The proposed multiplier is compared with two other approximate multipliers: the ETM in [7] and the 2×2 approximate multiplier in [6]. In this comparison, the ETM is divided equally into multiplication and non-multiplication sections, while the proposed multiplier uses 10 MSBs for error reduction. As shown in the results in Table II, the proposed multiplier has the lowest NMED, MRED and ER among the three approximate multipliers. In particular, it has very low NMED and MRED compared to the other two designs.

V. Delay and Power Evaluation

A. Delay Estimation

Based on the linear model of [9], the delays of a full adder (Fig. 3(a)) and the approximate adder cell (Fig. 3(b)) are derived to be approximately $3\tau_g$ and $2\tau_g$, respectively, where τ_g is an approximate "gate delay". For an n-bit approximate multiplier, there are $\lceil log_2 n \rceil$ layers in the adder tree. Taking into consideration the delay of the error accumulation using OR gates, the delay of the proposed multiplier is given by

$$D_{Ap} = (2\lceil log_2 n \rceil + 1)\tau_g. \quad (9)$$

There are $\lfloor \log_{1.5} n \rfloor$ layers in the Wallace or Dadda tree and their delays are given by [10]

$$D_{W,D} = 3\lfloor \log_{1.5} n \rfloor \tau_g. \quad (10)$$

Table III shows the delay of the partial product accumulation tree in both the proposed and Wallace/Dadda multipliers. For a

TABLE III. DELAY OF PARTIAL PRODUCT ACCUMULATION TREE OF THE PROPOSED AND CONVENTIONAL MULTIPLIERS OF DIFFERENT SIZES.

n	8	16	32	64	2^k
$D_{Ap}(\tau_g)$	7	9	11	13	$2k+1$
$D_{W,D}(\tau_g)$	12	18	24	30	$\approx 5k$

TABLE IV. POWER CONSUMPTIONS OF FPGA IMPLEMENTATIONS OF THE 16-BIT APPROXIMATE AND WALLACE MULTIPLIERS.

	Dynamic	Quiescent	Total
Wallace (W)	0.122	0.083	0.205
Approximate (W)	0.068	0.082	0.150

16-bit multiplier, the delay of an exact multiplier tree is twice as large as the delay of the proposed multiplier tree; as the size of the multiplier increases, this factor is approximately 2.5. Since the approximate adder cell is simpler than a full adder, the approximate multiplier has no additional area overhead to achieve the shorter delay. For the 2×2 approximate multiplier in [6], only the partial product generation layer is simplified and the height of the partial product tree is only decreased by 1, so the delay reduction is quite limited. The ETM in [7] can reduce the $n \times n$ partial product tree to $\frac{n}{2} \times \frac{n}{2}$. By (10), the difference between the delays of $n \times n$ and $\frac{n}{2} \times \frac{n}{2}$ trees is approximately $3log_{1.5}2\tau_g \approx 5.13\tau_g$. In summary, the other two multipliers reduce the critical path delay by a limited value. In contrast, the proposed multiplier can reduce the delay of the partial product accumulation tree by nearly 60%, which scales with the size of the multiplier.

B. Experimental Results

1) FPGA Implementation: 16×16 approximate and Wallace multipliers are implemented in VHDL using the Xilinx Spantan3E XC3S500E FPGA. The critical path delays of the proposed approximate multiplier and the exact Wallace multiplier are 13.990ns and 21.999ns, respectively, thus achieving a reduction of 36.4%. The input data for simulating power consumption are given by the multiplication of two images. The node activity rates are extracted by performing post-place and route simulation running at the maximum frequency of the Wallace multiplier. Based on the activity rates, the Xilinx XPower Analyzer is used to obtain the power consumption, as shown in Table IV. The quiescent power of the approximate multiplier is slightly smaller than the Wallace multiplier, however the approximate multiplier saves 44.3% of the dynamic power compared to the Wallace multiplier. Overall, the proposed multiplier achieves a reduction of 26.8% in total power consumption.

2) ASIC Implementation: ASIC designs for $n \times n$ $(n = 8, 16)$ approximate multipliers with n-bit error reduction and

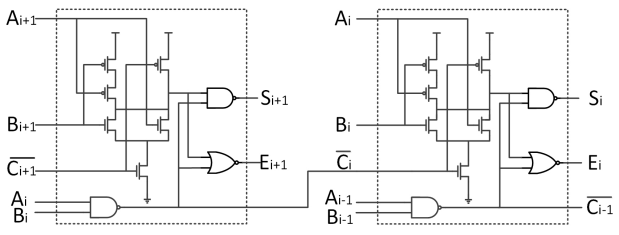

Fig. 4. Two neighboring approximate adder cells for ASIC implementation.

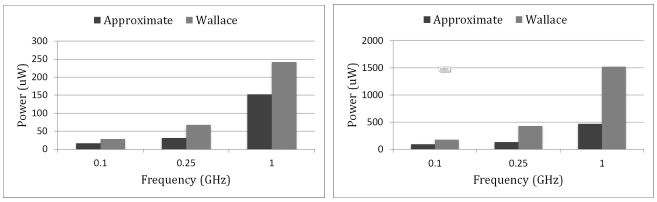

Fig. 5. Power vs. frequency for (a) 8-bit and (b) 16-bit approximate and Wallace multipliers.

Wallace multipliers of the same size have been implemented in STM 28nm CMOS process. The approximate adder cell in Fig. 3(b) is implemented using shared logic between two neighboring approximate adder cells, as shown in Fig. 4, thereby saving additional area. In Fig. 4, the signal $\overline{C_i}$ is given by $\overline{C_i} = \overline{A_i B_i}$ and shared between two cells. The critical path delays of 16×16 approximate and Wallace multipliers are 0.48ns and 0.6ns, respectively, resulting in a delay reduction of 20%. The power consumption for image multiplication is obtained by applying three frequencies (0.1 GHz, 0.25 GHz and 1GHz) to all these multiplier circuits. As shown in Fig. 5, the 8×8 and 16×16 approximate multipliers achieve power savings in the ranges of 37%-53% and 48%-69%, respectively, compared to the accurate Wallace multipliers.

VI. CONCLUSION

In this paper, a novel approximate multiplier design is proposed using a newly designed approximate adder. On a statistical basis the proposed multiplier has a very small error distance and thus a high accuracy. Simulations have shown that the proposed design has a shorter critical path delay and a significantly lower power consumption compared to an exact Wallace multiplier. It also uses a configurable error recovery that can produce more accurate results than other state-of-the-art approximate multipliers.

REFERENCES

[1] J. Han and M. Orshansky, "Approximate Computing: An Emerging Paradigm For Energy-Efficient Design," in *IEEE ETS*, 2013.

[2] J. Huang, J. Lach, and G. Robins, "A methodology for energy-quality tradeoff using imprecise hardware," in *DAC 2012*, pp. 504–509.

[3] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthesis of quality-energy optimal approximate adders," in *ICCAD 2012*, pp. 728–735.

[4] 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, 2013.

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

[6] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *24th IEEE Intl. Conf. on VLSI Design*, 2011, pp. 346–351.

[7] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in *IEEE Intl. Conf. Electron Devices and Solid-State Circuits (EDSSC)*, 2010, pp. 1–4.

[8] B. Parhami, *Computer arithmetic*. Oxford university press, 2000.

[9] N. H. Weste and H. David, *CMOS VLSI Design-A Circuit and Systems Perspective*, 3rd ed. Pearson Addison Wesley, 2005.

[10] K. Bickerstaff, E. Swartzlander, and M. Schulte, "Analysis of column compression multipliers," in *15th IEEE Symp. on Computer Arithmetic*, 2001, pp. 33–39.