REALM: Reduced-Error Approximate Log-based Integer Multiplier

Hassaan Saadat†, Haris Javaid*, Aleksandar Ignjatovic†, and Sri Parameswaran†

†The University of New South Wales, Sydney, Australia
*Xilinx Inc., Singapore
†{h.saadat, a.ignjatovic, sri.parameswaran}@unsw.edu.au, *harisj@xilinx.com

***Abstract*—We propose a new error-configurable approximate unsigned integer multiplier named REALM. It incorporates a novel error-reduction method into the classical approximate log-based multiplier. Each power-of-two-interval of the input operands is partitioned into $M \times M$ segments, and an error-reduction factor for each segment is analytically determined. These error-reduction factors can be used across any power-of-two-interval, so we quantize only M^2 factors and store them in the form of read-only hardwired lookup tables to keep the resource overhead to a minimum.**

Error characterization of REALM shows that it achieves very low error bias (mostly $\leq$0.05%), along with lower mean error (from 0.4% to 1.6%), and lower peak error (from 2.08% to 7.4%) than the classical approximate log-based multiplier and its state-of-the-art derivatives (mean errors $\geq$2.6% and peak errors $\geq$7.8%). Synthesis results using TSMC 45nm standard-cell library show that REALM enables significant power-efficiency (66% to 86% reduction) and area-efficiency (50% to 76% reduction) when compared with the accurate integer multiplier. We show that REALM produces Pareto optimal design trade-offs in the design space of state-of-the-art approximate multipliers. Application-level evaluation of REALM demonstrates that it has negligible effect on the output quality.

I. Introduction

Approximate computing enables area- and power-efficient implementations for error-resilient applications such as machine learning, artificial intelligence, and multimedia processing [1]. Such applications are heavily dominated by multiplication operations, and hence the design of hardware approximate multipliers is an active research area [2]. The primary goal when designing an approximate multiplier is to gain most resource-efficiency for least error (inaccuracy), which is typically characterized using various error metrics [2]–[4]. Other important design considerations are: (a) that the approximate multiplier should be *error-configurable* to provide various accuracy vs. resource-efficiency trade-offs for targeting applications with differing error resiliency [4]; (b) that it should have *low error bias* to facilitate cancellation of errors in successive computations [3], [4]; and, (c) that *systematic approaches for the introduced approximations* are preferred over ad-hoc approaches [5].

Most existing approximate multipliers do not take one or more of these design considerations into account, e.g., they either exhibit high error bias [4] and/or are based on ad-hoc approaches [6], [7]. A class of approximate multipliers that are mathematically formulated includes the classical log-based multiplier [8] and its state-of-the-art derivatives ALM-SOA [9], MBM [4], and ImpLM [10]. However, as depicted in Fig. 1(a-e), the errors of these multipliers are relatively high[1], i.e., mean error[2](ME) $\geq$2.6% and peak error[3](PE) $\geq$7.8%.

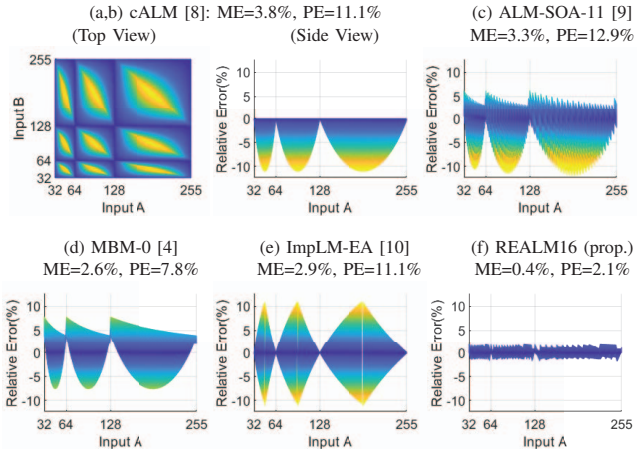

Fig. 1. Relative error profiles of (a-e) the classical and state-of-the-art log-based multipliers, and (f) the proposed multiplier. The profiles are shown for A,B={32,33,...,255} for better visualization.

In this paper, we propose a new Reduced-Error Approximate Log-based unsigned integer Multiplier (REALM), designed by integrating a novel mathematically formulated error-reduction method with the classical log-based multiplier. REALM is error-configurable as well as has low error bias[4]. Hence, it addresses all of the above design considerations. Fig. 1(f) shows the error profile of one of the REALM design-configurations; it outperforms the state-of-the-art approximate multipliers in error behavior with ME=0.4% and PE=2.08%. More extensive results in Section IV demonstrate that the error-configurable REALM can achieve 66%–86% power-reduction and 50%–76% area-reduction compared to an accurate multiplier, for mean error in the range 0.4%–1.6%, and peak error in the range 2.08%–7.4%. Moreover, it produces Pareto-optimal design points for accuracy vs. resource-efficiency trade-offs.

In particular, we make the following **contributions**.

- We partition each power-of-two-interval (see Fig. 2) of input operands in the classical log-based multiplier into $M \times M$ segments, and apply error-reduction to each seg-

[1]For each multiplier, we picked its configuration with the least mean error.
[2]Mean error is defined as *mean of absolute relative error*. It is also referred to as mean relative error distance (MRED) in the literature [2].
[3]Peak error is defined as *peak (or maximum) absolute relative error* [4].
[4]Error bias is defined as *mean of relative errors* [3], [4].

978-3-9819263-4-7/DATE20/©2020 EDAA

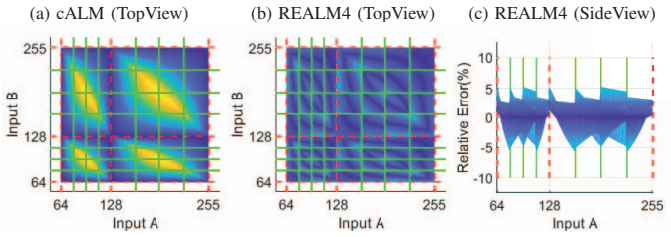

Fig. 2. 4×4 partitioning (green lines) of each power-of-two-interval (dashed red lines). Error is reduced for each segment separately, which results in an overall reduction in error. Demonstrated for M=4 and A,B=$\{64,...,255\}$ for better visualization. The term *power-of-two* implies a number 2^i, e.g., 64, 128. A *power-of-two-interval* refers to the values between two consecutive power-of-twos.

ment individually which leads to overall better error characteristics than the state-of-the-art approximate multipliers (Section III-B).

- For determining the error-reduction factors, a novel mathematical formulation is proposed involving minimization of the average relative error over each of the $M\times M$ segments. Thus, M^2 error-reduction factors are computed for a given power-of-two-interval. The error-reduction factors are independent of that interval and can be applied to any other interval. Hence, we need only M^2 factors for the whole multiplier (Section III-B).
- We also present a hardware design of the proposed multiplier. The error-reduction factors are quantized to q-bit precision and stored as read-only hardwired lookup tables. This eliminates the need for memory and its associated overheads, resulting in little overhead (Section III-C).
- We introduce two knobs for error-configurability through the configurable number of segments M, and application of bit truncation to the fractional parts of the log-values of the input operands. As a result, we provide a wide and dense design space to target different applications (Section III-C).

We conducted a thorough evaluation of REALM and compared it with many state-of-the-art approximate multipliers (such as [3], [4], [9], [10], and an integer version of [11]). Our results show that REALM produces Pareto-optimal accuracy vs. resource-efficiency trade-offs, and is able to achieve double-sided and symmetric error distribution. Moreover, evaluation with JPEG application shows negligible effect on application-level output quality (Section IV).

II. Related Work

Approximate Integer Multipliers. Several designs for approximate unsigned integer multipliers have been proposed. The earlier designs mostly involve ad-hoc based approximations. They involve approximating 2x2 multiplier blocks in recursive multipliers [7], simplification of Wallace tree [6], or approximation in 4:2 counters [12], [13]. Details of these earlier works can be found in [2], [4]. Among the recent unsigned integer multipliers, SSM [14], ESSM [14], and DRUM [3] multipliers extract m-bit (or k-bit) fragments from the N-bit inputs to use a smaller multiplier. They differ with respect to the starting position of the extracted fragment in the N-bit input. AM1 and AM2 use approximate adders that generate error vectors to be used for configurable error recovery [15].

The classical approximate log-based multiplier (cALM) was proposed by Mitchell using a mathematical formulation [8]. It is based on the log-multiplication property. It approximates log of the input operands by linear approximation between each power-of-two-interval of the operands, and then adds and scales them to get the approximate product. Over the last two years, new designs derived from cALM have been proposed. In [9], the authors use different types of approximate adders (LOA, MAA, and SOA) for the addition step. In ImpLM [10], the log approximation is improved by determining the power-of-two nearest to the operand, instead of the highest power-of-two smaller than the operand. In MBM [4], an error-correction mechanism is coupled with cALM. A single error-correction term is used for the whole multiplier, which is computed by averaging the actual error over a complete power-of-two-interval of inputs. While this approach in MBM achieves a low error bias, the mean error and peak error are still high (Fig. 1(d)), because the error profile over a power-of-two-interval varies significantly, as depicted in Fig. 1(a,b).

In contrast, in REALM, we partition each power-of-two-interval into $M\times M$ equispaced segments, and determine an error-reduction factor for each partitioned segment individually, which consequently leads to reduction in overall error, as mentioned in Section I (Fig. 2). Moreover, we propose a relative error based mathematical formulation to determine error-reduction factors instead of using the actual error (unlike MBM), because most of the common error metrics are based on relative error [2]–[4]. To the best of our knowledge, there is no work on approximate integer multiplication that partitions the power-of-two-interval into multiple segments and determines the error-reduction factors mathematically.

Approximate Floating-Point (FP) Multipliers. Some recent works in the literature have proposed approximate FP multipliers which are not directly relevant, but are worth mentioning [4], [11]. In [4], optimized versions of MBM and DRUM are used as mantissa multipliers. The ApproxLP [11] multiplier segments the product curve of the two mantissas $((1+x)(1+y)$ where $x, y\in[0, 1.0))$ through the use of piece-wise linear-plane approximation. ApproxLP is fundamentally different from our work, because it targets FP arithmetic and segments product curve instead of the error profile. Moreover, their work does not report any mathematical formulation and involves much complex selection logic for the segments. Nonetheless, in this paper, we created an integer version of ApproxLP (IntALP) for comparison purposes (see Section IV).

III. The Proposed REALM

A. Preliminaries (mathematical formulation of cALM)

Suppose two N-bit unsigned integers A and B, having leading-ones at positions k_a and k_b, respectively. These can be represented as: $A = 2^{k_a}(1+x)$ and $B = 2^{k_b}(1+y)$, where $x = \sum_{p=0}^{k_a-1} 2^{p-k_a}a_p$ and $y = \sum_{p=0}^{k_b-1} 2^{p-k_b}b_p$, with a_p and b_p representing the p^{th} bits of the N-bit integers A and B, respectively. Note that $0 \le x < 1$ and $0 \le y < 1$. The linearly approximated binary log ($\widetilde{lg}()$) of A and B are,

$$\widetilde{lg}(A) = k_a + x, \qquad \widetilde{lg}(B) = k_b + y \tag{1}$$

Integers k_a and k_b represent the characteristic parts of the two log-values, and x and y represent the fractional parts. To use the log-multiplication property, the two log-values are added,

$$\widetilde{lg}(C) = k_a + x + k_b + y \tag{2}$$

where C denotes the product of A and B. To obtain the approx. product ($\widetilde{C}$), inverse of the approx. log (scaling) is applied,

$$\widetilde{C} = \begin{cases} 2^{k_a+k_b}(1+x+y), & x+y<1 \\ 2^{k_a+k_b+1}(x+y), & x+y\geq 1 \end{cases} \tag{3}$$

The equation is decomposed into two conditions because to apply the inverse of approximate log, the mantissa (fractional) part of the log-value should be in the interval [0,1).

B. Proposed error-reduction methodology for REALM

The accurate product C of the two integers is,

$$C = 2^{k_a+k_b}(1+x)(1+y) \tag{4}$$

The relative error is calculated as,

$$\widetilde{E}_{rel} = \frac{(\widetilde{C}-C)}{C} = \begin{cases} \frac{1+x+y}{(1+x)(1+y)} - 1, & x+y<1 \\ \frac{2(x+y)}{(1+x)(1+y)} - 1, & x+y\geq 1 \end{cases} \tag{5}$$

This is plotted in Fig. 1(a,b) and Fig. 2(a). We partition each power-of-two-interval into $M\times M$ equispaced segments and then apply an error-reduction factor to each segment (Fig. 2).

Specifically, to incorporate the error-reduction, we introduce a factor $r_{ij}^{k_a,k_b}$ in the approximate product $\widetilde{C}$ for a given interval k_a, k_b, where $i,j\in\{0,1,\ldots,(M-1)\}$ denote the indices for segments in a given power-of-two-interval. Mathematically, the error-reduced product $\widehat{C}$ can be written as,

$$\widehat{C} = \widetilde{C} + r_{ij}^{k_a,k_b} \tag{6}$$

Using Equations 5 and 6, the new relative error $\widehat{E}_{rel}$ is,

$$\widehat{E}_{rel} = \widetilde{E}_{rel} + \frac{r_{ij}^{k_a,k_b}}{2^{k_a+k_b}(1+x)(1+y)} \tag{7}$$

We determine $r_{ij}^{k_a,k_b}$ such that the error within each segment is minimized. Specifically, we solve for $r_{ij}^{k_a,k_b}$ such that average relative error over each segment is 0. Mathematically,

$$\frac{1}{(\frac{1}{M})(\frac{1}{M})} \int_{i/M}^{(i+1)/M}\int_{j/M}^{(j+1)/M} \widehat{E}_{rel}\, dxdy = 0 \tag{8}$$

which is equivalent to,

$$M^2 \int_{i/M}^{(i+1)/M}\int_{j/M}^{(j+1)/M} \left(\widetilde{E}_{rel} + \frac{r_{ij}^{k_a,k_b}}{2^{k_a+k_b}(1+x)(1+y)}\right) dxdy = 0 \tag{9}$$

This equation implies,

$$r_{ij}^{k_a,k_b} = -(2^{k_a+k_b}) \frac{\int_{i/M}^{(i+1)/M}\int_{j/M}^{(j+1)/M} \widetilde{E}_{rel}\, dxdy}{\int_{i/M}^{(i+1)/M}\int_{j/M}^{(j+1)/M} \frac{1}{(1+x)(1+y)}\, dxdy} \tag{10}$$

The limits of integration correspond to M equispaced segments of x and y. Since the fraction in the above expression is independent of k_a, k_b, let us denote it by s_{ij}.

$$s_{ij} = -\frac{\int_{i/M}^{(i+1)/M}\int_{j/M}^{(j+1)/M} \widetilde{E}_{rel}\, dxdy}{\int_{i/M}^{(i+1)/M}\int_{j/M}^{(j+1)/M} \frac{1}{(1+x)(1+y)}\, dxdy} \tag{11}$$

Thus,

$$r_{ij}^{k_a,k_b} = 2^{k_a+k_b} s_{ij} \tag{12}$$

These error-reduction factors are applied to the approximate product $\widetilde{C}$ to get an error-reduced product $\widehat{C}$ (using Equations 3 and 6) as follows,

$$\widehat{C} = \begin{cases} 2^{k_a+k_b}(1+x+y) + r_{ij}^{k_a,k_b}, & x+y<1 \\ 2^{k_a+k_b+1}(x+y) + r_{ij}^{k_a,k_b}, & x+y\geq 1 \end{cases}$$

$$= \begin{cases} 2^{k_a+k_b}(1+x+y+s_{ij}), & x+y<1 \\ 2^{k_a+k_b+1}(x+y+s_{ij}/2), & x+y\geq 1 \end{cases} \tag{13}$$

Few observations can be made from Equation 13. Firstly, the error-reduction factors s_{ij} depend only on the segment i,j (interval of x,y values), irrespective of k_a and k_b. Thus, for $M\times M$ partitions, we only need to compute and store M^2 values. Moreover, since the error-reduction factors s_{ij} are now moved inside the parenthesis, in hardware they can be applied before the final scaling (independent of k_a and k_b). The calculations for s_{ij} are done offline before building the multiplier, and then stored in a lookup table as explained in the following subsection. We compute the values of the error-reduction factors s_{ij} for $M=\{4,8,16\}$ using MATLAB Symbolic Math toolbox. Our MATLAB scripts (and the computed values) are available as open-source at *https://github.com/hassaansaadat/realm*. Note that the formulation in Equation 8 can also be modified for other error metrics, such as mean square error. However, we will extend this work to use other error metrics in the mathematical formulation in the future.

C. The proposed hardware design

Overview. The hardware design of the proposed REALM for combinational realization is shown in Fig. 3. At the inputs, the leading-one detectors (LOD) and barrel shifters compute the characteristic (k_a, k_b) and the fractional parts (x, y) of the log-values, which are appended together and added. Then, an error-reduction factor (s_{ij}) from the lookup table (LUT) is added to the fractional part to realize Equation 13 (proposed error-reduction method). A 2×1-mux is used to realize $s_{ij}/2$. The mux selects between s_{ij} or $s_{ij}>>1$ with c_{of} (carry out from sum of fractional parts, $x+y$) as the select line. At the end, a barrel shifter is used to apply the final scaling to get a $2N$-bit approximate integer product.

Lookup Table and Selection Logic. The s_{ij} values are calculated offline and stored in an M^2-sized lookup table, in a row-major fashion. We round the s_{ij} values to q-bit precision

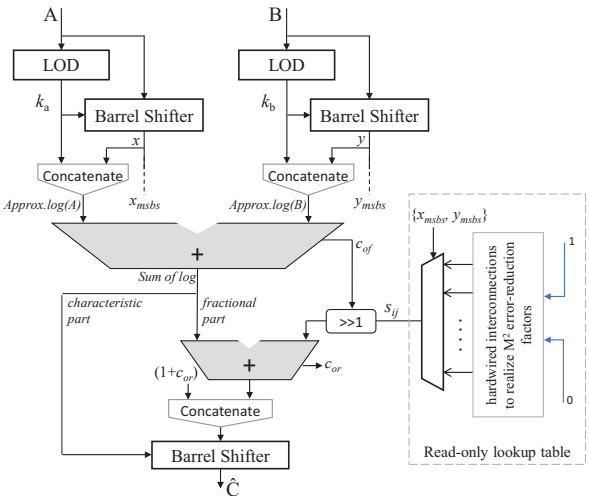

Fig. 3. The Proposed REALM design.

(weight of LSB is 2^{-q}) by applying round-to-nearest rounding. We noticed that for practical values of M=$\{4, 8, 16\}$, s_{ij} is always positive and $<$0.25. Thus, the first two bits of the fractional part of s_{ij} values (weight 2^{-1} and 2^{-2}) are always zero, and thus need not to be stored. Effectively, the width of the lookup table is $(q-2)$-bits. We store the values as read-only hardwired constants, hence no memory is required. Consequently, only a $(q-2)$-bit $M^2\times1$ multiplexer with constant inputs constitutes our proposed lookup table.

Recall that we partitioned the power-of-two-intervals into equispaced segments; hence, the $log_2(M)$-MSB bits of x, y (labelled as x_{msbs}, y_{msbs} in Fig. 3) for the current inputs identify the given segment and are used to retrieve the corresponding value from the lookup table. Essentially, the x_{msbs}, y_{msbs} bits become the select-lines of the LUT-multiplexer.

Error-configurability. The proposed REALM is error-configurable by parameter M. Higher M means greater reduction in error. At the same time, it means a larger multiplexer is required. Hence, design complexity increases, consequently providing various accuracy vs. efficiency trade-offs.

Additionally, we truncate t LSB bits from the $N-1$-bit fractional part of the log-values. We also set the LSB of the remaining bits to 1 to round the errors induced by truncation. This approach is similar to [3], [4]. Effectively, it means that $(t+1)$ bits from the output of the input barrel shifters are truncated. This reduces the size of the input barrel shifters, the main adder, and the final barrel shifter. Thus, our proposed design is configurable by two design-knobs. Note that both configuration parameters (M and t) are design-time, and hence they also enable logic-area reduction.

Special Cases. Application of error-reduction results in two special cases. First, the output may overflow to $(2N+1)$ bits when the input operands are close to 2^N-1. Secondly, when the output products are very small (the final scaling factor is low), some of the bits from the error-reduction factors remain in the fractional part after the final scaling. These bits need to be truncated in the final approximate integer product and hence lost, resulting in high error for those cases and affecting the overall peak error. Logic for detecting and handling such cases appropriately is also included in the design. To avoid clutter, this is not shown in Fig. 3.

Handling Signed Numbers. In this paper, we have presented and evaluated REALM multiplier as an *unsigned* integer multiplier. However, it is straightforward to extend any unsigned integer multiplier for handling *signed* numbers. For details, the readers are referred to [3].

IV. Experiments and Results

A. Comparison with the state-of-the-art

To evaluate the efficacy of the proposed REALM design, we evaluate its error behavior and compare its design metrics with an accurate unsigned integer multiplier. Additionally, we also compare it with several state-of-the-art approximate unsigned integer multipliers. These multipliers are: cALM [8], ALM-MAA [9], ALM-SOA [9], ImpLM [10], MBM [4], AM1 [15], AM2 [15], DRUM [3], SSM [14], and ESSM [14]. As discussed in Section II, we also implement an integer version (IntALP) of ApproxLP [11]. For this, we first compute the log of the integer inputs to get the characteristic and fractional parts. We then apply the linear-plane approximation to compute the product of the fractional parts (Equation 4) and then scale it with respect to the sum of the characteristic parts.

B. Experiment setup and implementation details

We implemented 16-bit versions of the above-mentioned multipliers in Verilog HDL as single-cycle designs. We implemented REALM multiplier (full design in Fig. 3, including the lookup table and selection logic) for q=6, M=$\{16, 8, 4\}$ with t=$0, 1, \ldots, 9$. The leading-one and nearest-one detectors, and the barrel shifters in the log-based multipliers were implemented using behavioral description (case statements and shift operator). The proposed multiplexer-based LUT with constant inputs was also described using case statement. The accurate multipliers were implemented using Wallace tree.

We synthesized all multipliers with Cadence Encounter(R) RTL Compiler RC14.2 using TSMC 45nm standard-cell library at the frequency of 1 GHz. We placed sequential elements at the inputs and outputs of the multipliers to apply timing constraints, however, we used the combinational area and power to report the results. For the power analysis, we annotated the inputs with the switching activity of 25% toggle rate and 50% probability.

For error characterization, we developed behavioral simulation models of the multipliers in MATLAB. Monte-Carlo simulations were performed using 2^{24} random inputs uniformly distributed over the set $\{0, \ldots, (2^{16}-1)\}$. The errors are reported with respect to the accurate product. The error metrics used to report error are: error bias [3] (mean of relative error); mean error [4] (mean of *absolute* relative error, also referred as MRED in the literature [2]); variance [3] (variance of relative error); maximum and minimum relative error; and, peak error [4] (maximum of absolute relative error). All error metrics are in percentages.

C. Results for 16-bit approximate multipliers

The error and design metric results for the implemented approximate multipliers are given in Table I. The area and power results are reported as percentage reductions, calculated with

TABLE I
RESULTS FOR 16-BIT MULTIPLIERS. THE AREA- AND POWER-REDUCTION ARE W.R.T ACCURATE MULTIPLIER (*Area=1898.1μm^2, Power=821.9μW*).

Approximate Multiplier Design	Error Config. Param.	Design Metrics (%) Area Reduc.	Power Reduc.	Errors Metrics (%) Error Bias	Mean Error	Peak Errors Min	Max	Variance
REALM16	t=0	50.0	65.6	0.01	0.42	-2.08	1.79	0.28
	t=1	51.5	66.2	0.01	0.42	-2.07	1.79	0.28
	t=2	51.5	67.9	0.02	0.42	-2.08	1.80	0.28
	t=3	53.4	69.2	0.02	0.42	-2.10	1.81	0.28
	t=4	55.0	70.2	0.02	0.42	-2.12	1.84	0.28
	t=5	56.6	72.0	0.02	0.42	-2.16	1.90	0.28
	t=6	57.3	73.4	0.02	0.43	-2.26	2.01	0.29
	t=7	58.3	74.8	0.02	0.45	-2.47	2.23	0.33
	t=8	60.1	76.5	0.02	0.55	-2.84	2.68	0.47
	t=9	62.9	79.2	-0.13	0.86	-4.37	3.81	1.12
REALM8	t=0	59.5	70.8	-0.05	0.75	-3.70	2.88	0.92
	t=1	62.2	72.9	-0.05	0.75	-3.70	2.89	0.92
	t=2	62.6	74.1	-0.05	0.75	-3.70	2.90	0.92
	t=3	64.4	75.2	-0.04	0.75	-3.72	2.91	0.92
	t=4	65.9	76.8	-0.04	0.75	-3.74	2.94	0.92
	t=5	66.8	77.9	-0.04	0.76	-3.78	3.00	0.92
	t=6	68.3	79.4	-0.04	0.76	-3.88	3.13	0.93
	t=7	69.0	80.6	-0.04	0.77	-4.09	3.37	0.96
	t=8	70.9	82.5	-0.04	0.83	-4.48	3.85	1.11
	t=9	72.9	84.9	-0.18	1.06	-5.27	4.81	1.75
REALM4	t=0	62.8	73.2	-0.02	1.38	-5.71	5.21	3.07
	t=1	64.5	74.7	-0.02	1.38	-5.71	5.22	3.07
	t=2	64.2	74.9	-0.02	1.38	-5.71	5.22	3.07
	t=3	67.0	77.4	-0.02	1.38	-5.73	5.24	3.07
	t=4	66.1	77.3	-0.02	1.38	-5.75	5.27	3.07
	t=5	69.1	79.5	-0.01	1.38	-5.81	5.34	3.07
	t=6	68.5	80.1	-0.01	1.39	-5.90	5.47	3.08
	t=7	71.7	82.3	-0.01	1.39	-6.12	5.73	3.12
	t=8	74.0	84.2	-0.01	1.43	-6.53	6.25	3.26
	t=9	75.6	86.4	-0.22	1.58	-7.35	7.29	3.96
Approximate Log-based Multipliers from the Literature								
cALM [8]	-	69.8	77.3	-3.85	3.85	-11.11	0.00	8.63
ImpLM [10]	EA	11.9	54.2	-0.04	2.89	-11.11	11.11	14.70
MBM [4]	t=0	63.9	74.3	-0.09	2.58	-7.64	7.81	10.02
	t=2	66.0	76.8	-0.09	2.58	-7.65	7.84	10.02
	t=4	68.5	79.0	-0.09	2.58	-7.69	7.91	10.02
	t=6	70.4	81.3	-0.09	2.58	-7.87	8.20	10.03
	t=8	74.3	84.8	-0.08	2.60	-8.59	9.38	10.23
	t=9	76.2	86.8	-0.38	2.70	-10.19	10.94	11.33
ALM-MAA [9]	m=3	72.5	79.9	-3.85	3.85	-11.12	0.01	8.63
	m=6	74.1	82.0	-3.85	3.85	-11.16	0.10	8.63
	m=9	74.7	83.5	-3.84	3.86	-11.56	0.78	8.72
	m=11	76.8	85.7	-3.84	4.00	-12.92	3.03	10.08
	m=12	76.9	86.7	-3.81	4.37	-14.66	5.88	14.43
ALM-SOA [9]	m=3	72.9	79.9	-3.84	3.84	-11.12	0.02	8.63
	m=6	75.1	83.2	-3.81	3.81	-11.16	0.19	8.64
	m=9	76.8	86.3	-3.58	3.63	-11.56	1.56	8.80
	m=11	78.8	88.8	-2.80	3.34	-12.91	6.25	10.78
	m=12	80.2	90.3	-1.75	3.58	-14.66	12.50	17.03
IntALP* (inspired by [11])	L=2	17.8	21.5	0.03	0.99	-2.86	4.17	1.67
	L=1	56.9	66.0	3.91	3.91	0.00	12.50	9.79
Other Existing Approximate Multipliers								
AM1 [15]	nb=13	22.5	46.9	-0.44	0.44	-61.57	0.00	1.79
	nb=9	31.1	55.4	-1.41	1.41	-61.71	0.00	12.22
	nb=5	38.4	62.4	-6.27	6.27	-61.93	0.00	79.41
AM2 [15]	nb=13	12.8	40.3	-0.25	0.25	-61.57	0.00	1.20
	nb=9	26.1	52.6	-1.21	1.21	-61.71	0.00	11.74
	nb=5	37.1	61.8	-6.12	6.12	-61.93	0.00	79.59
DRUM [3]	k=8	49.4	59.6	0.01	0.37	-1.49	1.57	0.20
	k=7	54.9	67.8	0.02	0.73	-2.96	3.15	0.81
	k=6	60.3	75.1	0.04	1.47	-5.78	6.35	3.26
	k=5	76.8	85.3	0.14	2.94	-10.76	12.89	13.06
	k=4	80.4	88.6	0.53	5.89	-18.96	26.56	52.69
SSM [14]	m=10	56.8	61.0	-0.40	0.40	-10.26	0.00	0.30
	m=9	63.8	69.6	-0.93	0.93	-34.27	0.00	2.54
	m=8	71.4	77.3	-2.08	2.08	-72.70	0.00	17.61
ESSM8 [14]	m=8	68.4	74.5	-1.14	1.14	-11.26	0.00	0.92

respect to the accurate multiplier as $(d_{acc}-d_{appx})/d_{acc}\times 100$, where d_{acc} and d_{appx} denote the area/power of the accurate and approx. multiplier, respectively. Several error-configurations are used for the approx. multipliers, as depicted in the second column of the table. Details of these configurations can be found in the respective papers. For ImpLM, we used the exact adder to achieve the smallest possible error in the design. In REALM, the parameter M is used as a suffix, i.e., REALM4 means M=4.

REALM Multiplier Results. In Table I, we observe that the REALM achieves very low error bias for all values of M ($\leq$0.05% for $t\leq 8$). When no bit truncation is applied, REALM can achieve mean error of 0.42%, 0.75%, and 1.38% and peak error of 2.08%, 3.70%, and 5.71%, for $M=\{16, 8, 4\}$, respectively. For the same values of M, power-reductions

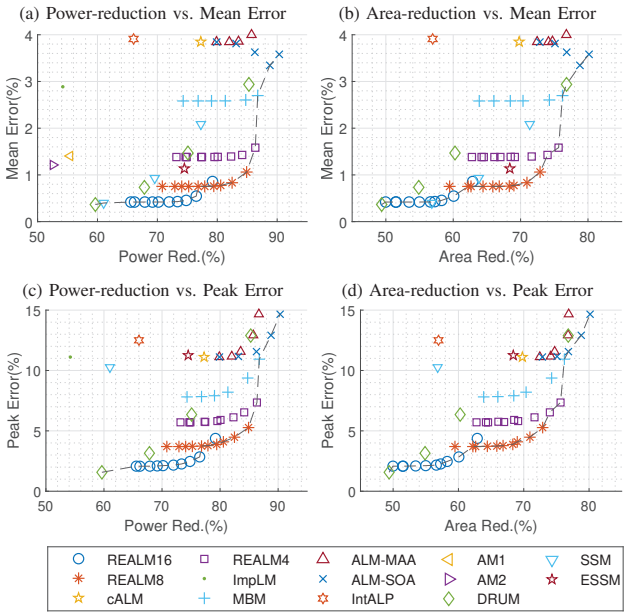

Fig. 4. Design space: comparison of REALM with the state-of-the-art.

are 65.6%, 70.8%, and 73.2%, and area-reductions are 50%, 59.5%, and 62.8%. As expected, the error improves with more partitions (increasing M) while the overhead also increases.

When bit truncation is applied in REALM, the error bias remains largely unaffected (except t=9), whereas other error metrics increase gradually. The effect of bit truncation on error becomes more prominent when $t\geq 7$. On the other hand, area-reduction and power-reduction show significant positive impact of bit truncation. In effect, the two error-configuration knobs t and M, enable area-reduction from 50.0% to 75.6% and power-reduction from 65.6% to 86.4%, with respect to the accurate multiplier. At the same time, the mean error varies from 0.42% to 1.58%, variance varies from 0.28% to 3.96%, and peak error varies from 2.08% to 7.35%.

Comparison with the State-of-the-art: From the table, we observe that the proposed REALM has much-improved error characteristics compared to other log-based multipliers. All of these multipliers have higher error and variance (e.g., mean error >2.5%, except IntALP-L2, but it gives power-reduction of only 21.5%). In terms of design metrics, the area- and power-reductions of REALM are comparable to that of cALM and other log-based multipliers, thus demonstrating that the proposed error-reduction method with the hardwired-lookup tables has little overhead.

To evaluate the efficacy of the proposed REALM in terms of design trade-offs, we present the results in the form of design space in Fig. 4. Area- and power-reductions are plotted on the x-axis, while mean error and peak error are plotted on the y-axis. We constrain the plots to mean error $\leq$4% and peak error $\leq$15%. The Pareto optimal points are outlined by a dashed grey line. We observe that the Pareto front is primarily achieved by our proposed REALM. At the extreme ends, DRUM8 (lower end) and MBM, DRUM5, ALM-SOA (upper end) also lie at the Pareto front. However, area-reduction for DRUM8 is less than 50%, while these MBM, DRUM5, and

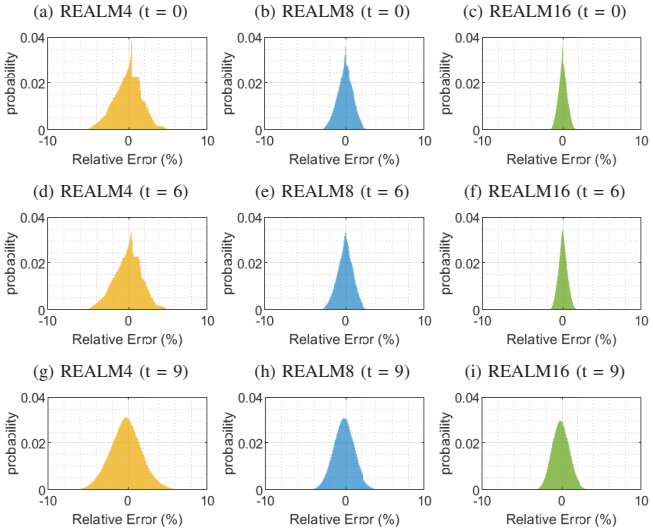

Fig. 5. Distributions of relative errors of the proposed REALM for different configurations of M and t.

ALM-SOA configurations have mean errors $>2.5\%$ and peak errors $>10\%$. In contrast, *REALM offers a large number of Pareto optimal points in the range of 65.6%–86.4% power-reduction and 50.0%–75.6% area-reduction with less than* 1.6% *mean error and less than* 7.4% *peak error.*

REALM Error Distribution. The error distributions of the proposed REALM for different error-configurations are shown in Fig. 5. We observe that the distributions are double-sided and nearly centred on zero (low bias). Moreover, as M increases, the distributions become narrower and more symmetric. Specifically, the narrow distribution of REALM16 justifies the low variance and low error bias values reported in Table I. The effect of bit truncation is also shown in Fig. 5(d-i). For t=6, the error distribution is similar to that for t=0, thus showing that lower values of t have negligible effect on error characteristics (when $t\leq 6$). For t=9, the shape of the error distributions gets comparatively wider and displaced from zero, because the error characteristics get slightly worse (Table I).

D. Application Evaluation (JPEG)

JPEG is a lossy image compression method. The quality of the compressed image is typically evaluated by comparing it with the uncompressed image using metrics such as PSNR (higher is better). We implemented JPEG (quality level = 50) in 16-bit fixed-point arithmetic, using accurate and approximate multipliers and performed compression of three standard images from the image processing literature. For the other log-based multipliers, we selected the configuration that gives the least mean error for the design (IntALP-L2 is not included because its resource-efficiency is significantly low). Table II shows the PSNR values of the compressed images against uncompressed images. We observe that the PSNR values for our proposed REALM are almost similar to the accurate multiplier (difference ≤ 0.4dB), whereas, for all other log-based approximate multipliers, the drop in PSNR is >2dB. Thus, REALM has a negligible effect (performs better than state-of-the-art) on the JPEG output quality.

TABLE II
PSNR (dB) FOR JPEG COMPRESSION USING MULTIPLIERS

Image	**Accurate Multiplier**	*REALM16 (t=8)*	*REALM8 (t=8)*	*REALM4 (t=8)*	MBM (t=0)	cALM	ImpLM (EA)	IntALP (L=1)	ALM-SOA (m=11)
cameraman	**31.8**	*32.0*	*31.7*	*31.4*	28.4	22.1	28.0	21.5	23.8
lena	**32.1**	*32.2*	*32.1*	*31.7*	28.8	23.0	28.8	21.6	24.7
livingroom	**30.4**	*30.5*	*30.5*	*30.1*	28.1	23.3	27.7	22.5	24.8

V. Conclusions

In this paper, we propose a new Reduced-Error Approximate Log-based unsigned integer Multiplier (REALM). REALM is designed by integrating a novel error-reduction method into the classical approximate log-based multiplier. We partition each power-of-two-interval into $M\times M$ equispaced segments and determine an error-reduction factor for each segment using a proposed relative error based mathematical formulation. For hardware realization of the proposed method, the determined error-reduction factors are stored as read-only hardwired lookup tables that result in little overhead. REALM exhibits low error bias, and it has two error-configuration knobs that enable it to produce a wide and dense design space. We performed a thorough evaluation and comparison with state-of-the-art using several error metrics. Results show that REALM produces Pareto optimal accuracy vs. area- and power-efficiency trade-offs. Moreover, the error distribution of REALM is double-sided and symmetric with low bias. Application-level evaluation, performed using JPEG compression, demonstrated that the proposed REALM has negligible effect on output quality.

References

[1] S. Venkataramani *et al.*, "Approximate computing and the quest for computing efficiency," in *Proc. DAC*, 2015.
[2] H. Jiang *et al.*, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, pp. 60:1–60:34, 2017.
[3] S. Hashemi *et al.*, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. ICCAD*, 2015.
[4] H. Saadat *et al.*, "Minimally biased multipliers for approximate integer and floating-point multiplication," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2623–2635, 2018.
[5] S. Rehman *et al.*, "Architectural-space exploration of approximate multipliers," in *Proc. ICCAD*, 2016.
[6] K. Bhardwaj *et al.*, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in *Proc. ISQED*, 2014.
[7] P. Kulkarni *et al.*, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. VLSID*, 2011.
[8] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electronic Computers*, vol. EC-11, no. 4, pp. 512–517, 1962.
[9] W. Liu *et al.*, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, 2018.
[10] M. S. Ansari *et al.*, "A hardware-efficient logarithmic multiplier with improved accuracy," in *Proc. DATE*, 2019.
[11] M. Imani *et al.*, "ApproxLP: Approximate multiplication with linearization and iterative error control," in *Proc. DAC*, 2019.
[12] A. Momeni *et al.*, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Computers*, vol. 64, no. 4, pp. 984–994, 2015.
[13] N. Maheshwari *et al.*, "A design approach for compressor based approximate multipliers," in *Proc. VLSID*, 2015.
[14] S. Narayanamoorthy *et al.*, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 6, pp. 1180–1184, 2015.
[15] H. Jiang *et al.*, "Low-power approximate unsigned multipliers with configurable error recovery," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 189–202, 2019.