PAALM: Power Density Aware Approximate Logarithmic Multiplier Design

Shuyuan Yu
University of California, Riverside
Department of Electrical and Computer Engineering
Riverside, California, United States
syu070@ucr.edu

Sheldon X.-D. Tan
University of California, Riverside
Department of Electrical and Computer Engineering
Riverside, California, United States
stan@ece.ucr.edu

ABSTRACT

Approximate hardware designs can lead to significant power or energy reduction. However, a recent study showed that approximated designs might lead to unwanted higher temperature and related reliability issues due to the increased power density. In this work, we try to mitigate this important problem by proposing a novel power density aware approximate logarithmic multiplier (called *PAALM*) design for the first time. The new multiplier design is based on the approximate logarithmic multiplier (ALM) framework due to its rigorous mathematics based foundation. The idea is to re-design the high computing switch activities of existing ALM designs based on equivalent mathematical formula so that the power density can be reduced at no accuracy loss while at costs of some area overheads. Our results show that the proposed PAALM design can improve 11.5%/5.7% of power density and 31.6%/70.8% of area with 8/16-bit precision when compared with the fixed-point multiplier baseline, respectively. And also achieves extremely low error bias: -0.17/0.08 for 8/16-bit precision, respectively. On top of this, we further implement the PAALM design in a *Convolutional Neural Network* (CNN) and test it on CIFAR10 dataset. The results show that with error compensation, PAALM can achieve the same inference accuracy as the fixed-point multiplier baseline. We also evaluate the PAALM in a discrete cosine transformation (DCT) application. The results show that with error compensation, PAALM can improve the image quality of 8.6dB in average when compared to the ALM design.

ACM Reference Format:
Shuyuan Yu and Sheldon X.-D. Tan. 2023. PAALM: Power Density Aware Approximate Logarithmic Multiplier Design. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3567884

1 INTRODUCTION

One of the important paradigm changes for today's emerging computing workloads is that a large number of application domains, e.g., machine learning, imaging process, etc., exhibit an intrinsic error tolerance [1, 2]. It is further shown that in those applications 70% of their energy consumption on average is dissipated in computations that can be approximated for a given certain accuracy level [3]. As a result, accuracy can be traded off to improve hardware footprint, power/energy efficiencies and performance via so-called approximate computing [4, 5]. Approximate computing concept basically is to introduce error or quality loss as a new design metric to be traded off for area/power/energy reduction and/or improved performance. Recently, many research efforts have been spent to develop more performance and energy-efficient approximate computing hardware, especially for the machine learning and neural network applications [6]. Those workloads are heavily dominated by the multiplication operations and hence designs of hardware-efficient multiplier have been intensively investigated recently. The primary goal of the approximate multiplier design is to reduce the power and area for the least accuracy loss.

Table 1: 8-bit multipliers power density (synthesized using EDK 32nm standard cell library [7])

Approaches	Area (μm^2)	Power Density ($\mu W/\mu m^2$)	
		same frequency	same throughput
FxP (baseline)	**1754.10**	**0.0512**	**0.0512**
ALM [8]	820.63	0.0853	0.0693
LeAp [9]	1040.21	0.1023	0.0831
REALM [10]	1235.14	0.1010	0.0821
ILM-EA [11]	1221.92	0.0883	0.0718
ILM-AA [11]	887.47	0.0871	0.0708
HEALM-TA [12]	595.46	0.0863	0.0702
HEALM-SOA [12]	664.33	0.0896	0.0729

A number of approximate multiplier designs have been proposed recently [10, 13–21]. Those approximate multipliers employed some ad-hoc truncation or reduction methods or mathematically formulated approximation schemes. Most of the existing methods, however, lack the systematic (re)configurability for accuracy versus area/power/latency trade-off. On the other hand, a class of approximate multipliers that are mathematically formulated includes logarithmic multipliers, which convert multiplication into only shifting and addition operations, due to the inherent approximate nature of logarithmic operation and the easy accuracy manipulation of the resulting addition, the area, latency and power can be traded off at the cost of accuracy. The logarithmic multiplier was originally proposed by Mitchell [8]. Since then, many approximate logarithmic multipliers (ALMs) have been proposed to improve Mitchell's work [10, 11, 20, 22]. Most of those methods focused on how to reduce and compensate the errors introduced in the piece-wise approximation of the log function, which tends to cause negative error behavior.

Recently Ansari *et al.* [11] developed an approximate scheme to make the error distribution more balanced (double sided errors) for the ALM method. Saadat *et al.* [10] introduced a general error

This work is supported in part by NSF grants under No. OISE-1854276, in part by NSF grant under No. CCF-1816361, No. CCF-2007135 and No. CCF-2113928.

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01…$15.00
https://doi.org/10.1145/3566097.3567884

compensation technique based on Ansari's work, called *REALM*, using an analytically generated error reduction factor lookup table for different regions of input operands. The benefits of this method are that it can generate more balanced errors by designing and providing configurable design trade-off between area and precision. However, this method uses one lookup table for all the truncation configuration in the approximate addition, which may lead to large errors especially for low precision cases. Based on Saadat's work, Yu *et al.* [12] further proposed different lookup tables for different truncation configuration in the approximate summation, achieving improvements in both hardware footprint and error metrics for the low precision cases.

However, recent study showed that approximate multiplier designs may lead to unwanted higher power density, thus higher temperature and related reliability issues in the final chip design [23]. The reason is that though approximation schemes reduce both power and area at the same time, the power density, which is power per unit area, can increase as reduction ratios for power and area can be different. It turns out that as shown in Table 1, the power densities of recently proposed approximate multipliers are all larger than the fixed point exact multiplier (the baseline) no matter with the same working frequency or with the same throughput. As a result, those approximate multiplier designs can't run on the same frequency and voltage ranges of the original exact multiplier designs under the same temperature constraint [23].

In this work, we would like to address this new issue by explicitly considering the power density impacts in the approximate multiplier design. The new multiplier design is still based on the approximate logarithmic multiplier (ALM) framework due to its rigorous mathematics based foundation. The key contributions of this work are summarized as below:

- We show for the first time that most of recently proposed approximate logarithmic multipliers (ALMs) can actually lead to higher power density than the baseline design with potential thermal issues. To mitigate this issue, we profile the switching activities of the existing ALM design and try to reduce the high computing switch activities based on equivalent mathematical formula at some costs of area so that the power density can be reduced with no accuracy loss. Our method can be viewed as a way to trade-off some area overhead for power density reduction, which is however necessary for reducing local hot spots and maintain the long-term reliability of chips.
- Experimental results show that the proposed 8/16-bit *PAALM* design can improve 11.5%/5.7% of power density and 31.6%/70.8% of area when compared to the fixed-point multiplier baseline, respectively. By introducing inexact summation and error compensation, PAALM can achieve extremely low error bias of -0.17/0.08 with 8/16-bit precision, respectively.
- Furthermore, we implement the PAALM design in a *Convolutional Neural Network* (CNN) and test it on CIFAR10 dataset, showing that with error compensation, PAALM can achieve the same inference accuracy as the fixed-point multiplier. Besides, we evaluate the PAALM in a discrete cosine transformation (DCT) application. The results further show that with error compensation, PAALM can improve the image quality of 8.6dB in average when compared to the ALM design.

2 REVIEW OF RELATED WORK

Various approximate multiplication designs, especially for the unsigned integer inputs, have been proposed in recent years. Some earlier works often involved ad-hoc based approximations, such as recursive multipliers [13] which consisted of 2×2 multiplication blocks, simplification of Wallace tree [14], simplifying partial product generation/summation [15–17]. Others used a smaller multiplier by extracting m-bit fragment from the N-bit precision inputs such as [18, 19].

Recently, the approximate logarithmic multiplier (*ALM*) [8], and its derivatives, which are more mathematically formulated have been proposed. Those ALMs can convert multiplication into only shifting and addition operations, which are more hardware friendly and efficient. The ALM design shows good overall performance and has flexibility for trade-offs among area, power and accuracy. Specifically, in the ALM method, the two inputs A and B are first represented by the following format: $2^{k_a} \cdot (1 + x)$ and $2^{k_b} \cdot (1 + y)$, respectively. Then the multiplication result can be approximated as (1).

$$C_{ALM} = \begin{cases} 2^{k_a+k_b} \cdot (1 + x + y), & x + y < 1, \\ 2^{k_a+k_b+1} \cdot (x + y), & x + y \geq 1 \end{cases} \tag{1}$$

Where C_{ALM} is the approximate multiplication product. The ALM method requires four steps to do the approximate multiplication: First it utilizes leading-one detectors (LODs) to find the leading bit '1' as the integer part; second, barrel shifters are used to re-align the rest of the bits as the fraction part; then it sums the two fraction and integer parts up as $k_a + k_b + x + y$; and finally it shifts back with the same bits.

Though ALM performs a good trade-off among accuracy, area and power, it suffers from high absolute MRED (mean relative error distance) and peak relative error of 3.76% and 11.11%, respectively. To improve the accuracy of the ALM method, several derivative works have been proposed by means of different error compensation mechanisms. For instance, the MBM design tried to add a fixed single error-correction term to the final result [20]. This was further improved by the LeAp multiplier, which added different error coefficients to the fraction parts based on the value ranges of the results [9]. The REALM multiplier design further improved the compensation scheme by using a lookup table to store $M \times M$ coefficients/factors for $M \times M$ partitions of input ranges with some hardware resource overheads [10]. These works indeed improved the error metrics of the approximate logarithmic multiplication without incurring too much resource overheads.

Some previous works [11, 22] tried to do further area reduction by replacing the exact adder with an inexact one. Since the exact adder unit is the bottleneck of the ALM critical path and occupies large area, this idea does help in area saving. But the inexact adder also introduces extra error. A state of art work, called *HEALM*, did specific error compensation for different inexact adder with different number of truncation bits, improving the hardware footprint and the error metrics at the same time [12].

These existing logarithmic multiplication designs though achieved good trade-offs among accuracy and hardware metrics, they didn't also perform well in the thermal aspect. We list the power density of the *FxP* (fixed-point multiplier), which is selected as the baseline, the conventional ALM design and some state of art improved versions with 8-bit precision in Table 1. From the table, we notice that all the logarithmic multiplication designs have higher power density than the accurate baseline design, no matter with the same working frequency or with the same throughput (delay), which means a worse thermal behavior. And when compared

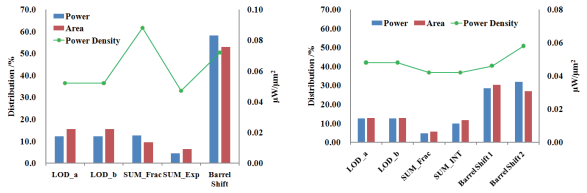
to the conventional ALM, the improved versions even further increase the power density. The logarithmic multiplication designs decrease the area and power at the same time. However, the area usually decreases faster than the power, which leads to a larger power density.

In this work, we will focus on the 8-bit and 16-bit power density aware approximate logarithmic multiplication design and demonstrate that the proposed work shows superior thermal performance not only against the ALM [8] and other state of art works like LeAp [9], REALM [10], ALM-SOA [22], ILM [11] and HEALM [12]; but also better than the accurate baseline in Sec. 4.

3 PROPOSED POWER DENSITY AWARE ALM

In this section, we present the details of our proposed power density aware logarithmic approximate multiplier (*PAALM*) design to mitigate the increase in the power density of most existing approximate multiplications, especially the log-based multiplication designs.

3.1 Power density aware logarithmic multiplication



(a) (b)

Figure 1: (a) The power, area distribution and power density of each component in the ALM design; (b) The power, area distribution and power density of each component in the PAALM design

The first thing we do is to profile the power consumption of the key components of the ALM design. The goal is to find the local power or power density hot spots so that we can re-design the multiplier to avoid those hot spots and the overall power density can be reduced in this way.

Specifically, we first show the power, area and the power density for each component in a conventional ALM design in Fig. 1(a). In this figure, the *LOD_a* and *LOD_b* modules represent the blocks detecting the leading one bit and performing the bit shifting operation to obtain the exponential part k_a, k_b and the fraction part x, y. The *Sum_Exp* module is used to sum the exponential parts k_a, k_b up. *Sum_Frac* module does the fraction part summation. The *Barrel Shift* module stands for the barrel shifter which delivers the output C_{ALM}. These ALM components are illustrated in Fig. 2.

From Fig. 1(a) we can see, the fraction summation part (*Sum_Frac* module) is the most power dense unit. However, it just takes up about 12.7% of the total power consumption and 9.5% of the area. On the other hand, the barrel shifter unit is not the most power dense component in the ALM structure, but dominates the power consumption of the ALM design, taking up 58.2% of the total power consumption and has higher power density ($0.072\mu W/\mu m^2$) compared to the fixed-point baseline ($0.051\mu W/\mu m^2$). Besides, the *Sum_Frac* module which has the highest power density, should be

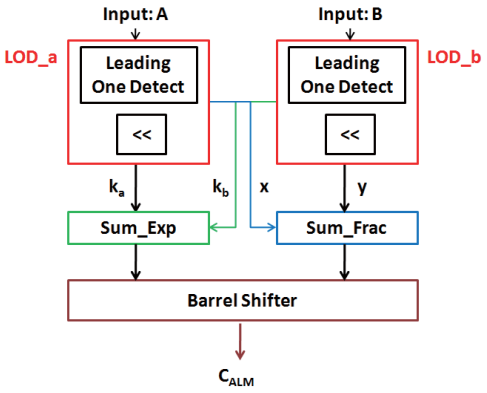


Figure 2: The conventional ALM design

re-designed and be replaced with modules which have less power density.

Based on the above observation, we propose to split the large barrel shifter into two small barrel shifters. In this way, we can resolve both of these aforementioned issues. Specifically, we rewrite the math expression of the ALM: (1) as follows:

$$C_{ALM} = \begin{cases} 2^{k_a+k_b} \cdot (1+x+y) \\ = 2^{k_a} \cdot (1+x) \cdot 2^{k_b} + 2^{k_b} \cdot y \cdot 2^{k_a} \\ = A \cdot 2^{k_b} + (B - 2^{k_b}) \cdot 2^{k_a}, & x+y<1, \\ 2^{k_a+k_b+1} \cdot (x+y) \\ = 2^{k_a} \cdot (1+x) \cdot 2^{k_b+1} + 2^{k_b} \cdot y \cdot 2^{k_a+1} \\ = (A - 2^{k_a}) \cdot 2^{k_b+1} + (B - 2^{k_b}) \cdot 2^{k_a+1}, & x+y \geq 1 \end{cases} \quad (2)$$

As we can see, compared to (1), one time $k_a + k_b$ bit shifting operation is separated to a k_a bit shifting and a k_b bit shifting operation.

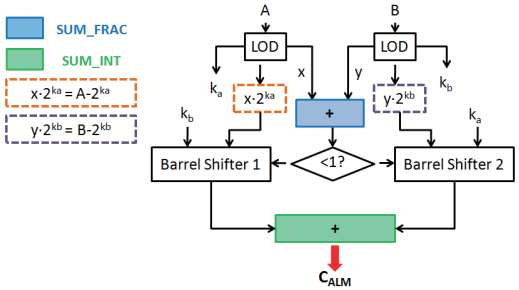


Figure 3: The power density aware ALM design

We show the PAALM design in Fig. 3. Unlike the conventional ALM, the LOD blocks in the PAALM structure not only detect the leading one bit and output the exponential and fractional parts, but also deliver the $A - 2^{k_a}$ and $B - 2^{k_b}$ according to (2). In this way, the barrel shifting operations are split into two shifting operations. But the real benefits is that the area increase from two barrel shifters will exceed their power increase, which leads to lower power density. After barrel shifting, the PAALM obtains $A \cdot 2^{k_b}$ or $(A - 2^{k_a}) \cdot 2^{k_b}$ depending on whether $x + y$ is smaller or larger than one, using a smaller barrel shifter to obtain two partial products. Then we add these two partial products up to achieve the final output C_{ALM} which is just the same as the output from the conventional ALM. For the most power dense unit *Sum_Frac*, we introduce the inexact summation to replace the exact adder to reduce power density. We will give more details in Sec. 3.2.

Fig. 1(b) shows the power, area and power density of the components in the new PAALM design. As we can see, the highest power density has been reduced from $0.088\mu W/\mu m^2$ to $0.058\mu W/\mu m^2$. Second, the new *Sum_INT* unit and the improved *SUM_Frac* unit have much lower power density ($0.042\mu W/\mu m^2$) compared to the initial *Sum_Frac* ($0.088\mu W/\mu m^2$) and *Sum_Exp* ($0.047\mu W/\mu m^2$) units. Third, the two small barrel shifters have lower power density than the previous large barrel shifter, which is beneficial to the overall power density. Fourth, the power density differences between different components are much smaller than the previous design, which is good for reducing some local hot spots (although this is less relevant as we assume that the typical thermal analysis granularity will be at the gate level). But the most important factor is that, we end up with reducing the power density of the conventional ALM from $0.069\mu W/\mu m^2$ to $0.051\mu W/\mu m^2$ of PAALM without hurting the accuracy.

3.2 The error compensation scheme

To split one large barrel shifter and replace it with two small barrel shifters, the PAALM actually introduces some area overheads, which will be demonstrated in Sec. 4. To mitigate this issue, also reduce the power density of the *SUM_Frac* unit which is mentioned in Sec. 3.1, we replace the exact summation of $x + y$ with simple inexact summation. Here, we select the simplest truncation adder (TA) to save resource. We use k to represent the number of bits truncated in the $x + y$ summation. As implementing the inexact summation will introduce extra accuracy loss, we need to do error compensation.

However, since the PAALM does the bit shifting operation separately instead of doing just once, we cannot do error compensation, like adding error coefficients directly to the fraction summation part $x + y$, which was usually tried in previous log-based multiplication works like [9, 10, 12]. To do special error compensation for the PAALM design, we replace one LOD unit with a *SO-LOD* (set one LOD) unit, which is shown in Fig. 4. Different from the conventional LOD unit, SO-LOD not only generates the exponential part and the fractional part, but also set all the bits of $A - 2^{k_a}$ part to bit '1'. As $A - 2^{k_a}$ is equivalent to $x \cdot 2^{k_a}$, we rewrite this part as $x' \cdot 2^{k_a}$ and show the structure of PAALM design with error compensation in Fig.5. Note that we just replace one of the two LOD units in PAALM with the SO-LOD unit to avoid "over compensation".

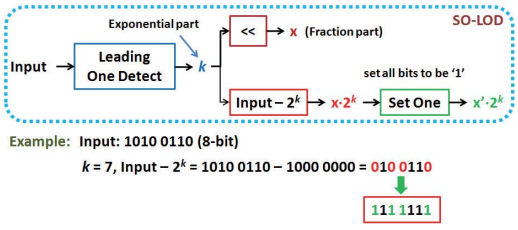


Figure 4: The set one LOD (SO-LOD) unit structure

4 EXPERIMENTAL RESULTS AND DISCUSSIONS

In this section, we evaluate the performance of the proposed power density aware approximate logarithmic multiplication approach, named *PAALM* under 8-bit and 16-bit precision. We also compare the proposed approach against the fixed-point exact multiplier

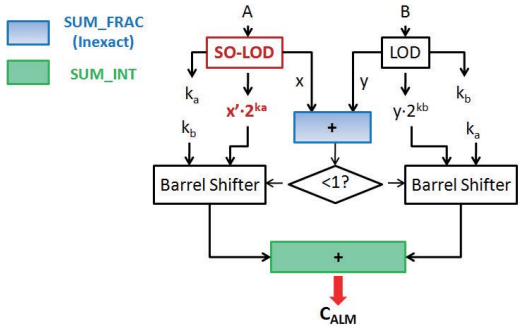


Figure 5: The power density aware ALM design with error compensation

baseline and the conventional ALM design under the same precision.

4.1 Experimental setup

To evaluate the performance of the proposed PAALM design, we demonstrate the hardware and error metrics of PAALM in Table 2 and Table 3. We also compare PAALM with other state of art improved ALMs. These improved ALMs include: LeAp [9], REALM [10], ILM-EA [11], ILM-AA [11], HEALM-TA [12], HEALM-SOA [12].

All the multipliers listed in Table 2 and Table 3 are implemented in Verilog HDL and synthesized with Synopsys Design Compiler using EDK 32nm standard cell library [7] as single-cycle designs. As different working frequency will lead to different power consumption of the circuit, and further introduce difficulties to compare the power density. We constrain the throughput (delay) of all the listed multipliers in each table to be the same. Note that different multiplication approaches require different computing latency (*Lat*) to give outputs, the working frequency (*Freq*) actually will be different for these listed approaches ($Freq = Lat/Delay$). Here, we select the delay/throughput of the fixed-point exact multiplier, which is the baseline, as standard. The critical paths for the 8-bit and 16-bit fixed-point multiplier are 2.07ns (5 cycles) and 4.51ns (7 cycles), respectively; so the delay are 10.35ns ($2.07ns \times 5$) and 31.57ns ($4.51ns \times 7$), respectively. Under these time constraints, we obtain the hardware performance for all the multipliers, including area and power density (*Power*/*Area*) aspects. As *Power* and *Energy* for all the multipliers can be easily calculated from the products of $Area \times PowerDensity$ and $Power \times Delay$, respectively. To save space, these two metrics will no be listed in the table.

To evaluate the error metrics, we develop behavioral simulation models for all the multipliers listed in Table 2 and Table 3 in MATLAB and measure the accuracy using 1 million (for 8-bit precision) and 10 million (for 16-bit precision) random inputs uniformly distributed over the set $\{0, 1, ..., (2^8-1)\}$ and the set $\{0, 1, ..., (2^{16}-1)\}$, respectively. The errors are reported with respect to the exact results. The error metrics used to report the error behavior include: mean error (mean of absolute relative error, also referred as MRED in some previous works [11]); and peak error (maximum value of the absolute relative error). All the error metrics are in percentages.

4.2 Performance evaluation and comparison

We show the hardware performance of the proposed PAALM design under 8-bit precision in Table 2. Note that as PAALM will introduce area overheads when compared to the conventional ALM design. We introduce inexact summation and error compensation scheme to save resource and improve error metrics as mentioned

Table 2: Design metrics for the 8-bit logarithmic multipliers with the same throughput

Approach	k	Lat.	Area (μm^2)	Power Density ($\mu W/\mu m^2$)	Err. Comp.	Mean Error /%	Peak Error /%
FxP (baseline)	/	5	1754.10	0.0512	/	/	/
ALM [8]	0	3	820.63	0.0693	w/o	3.76	11.11
PAALM (proposed)	0	2	1391.44	0.0507	w/o	3.76	11.11
	3	2	1326.89	0.0508	w/o	3.77	11.77
	3	2	1339.85	0.0505	w/t	2.96	11.77
	4	2	1274.79	0.0466	w/o	3.83	12.59
	4	2	1287.24	0.0455	w/t	3.27	12.59
	5	2	1212.52	0.0453	w/o	4.09	14.05
	5	2	1245.56	0.0503	w/t	3.52	14.05
	6	2	1199.81	0.0453	w/o	5.13	16.43
	6	2	1201.08	0.0467	w/t	4.42	16.43
Other state of art improved logarithmic multipliers							
LeAp [9]	0	3	1040.21	0.0831	w/t	1.38	4.71
REALM [10]	0	3	1235.14	0.0821	w/t	0.90	3.96
	4	3	709.57	0.0752	w/t	6.58	23.07
ILM-EA [11]	0	3	1221.92	0.0718	w/o	2.84	11.11
ILM-AA [11]	4	3	887.47	0.0708	w/o	5.47	23.47
HEALM-TA [12]	4	3	595.46	0.0702	w/t	3.66	13.77
HEALM-SOA [12]	4	3	664.33	0.0729	w/t	3.12	12.17

Table 3: Design metrics for the 16-bit logarithmic multipliers with the same throughput

Approach	k	Lat.	Area (μm^2)	Power Density ($\mu W/\mu m^2$)	Err. Comp.	Mean Error /%	Peak Error /%
FxP (baseline)	/	7	8753.48	0.0369	/	/	/
ALM [8]	0	3	1714.20	0.0407	w/o	3.85	11.11
PAALM (proposed)	0	2	3296.25	0.0360	w/o	3.85	11.11
	9	2	2939.18	0.0369	w/o	3.85	11.34
	9	2	2978.31	0.0365	w/t	3.43	11.34
	10	2	2836.50	0.0363	w/o	3.85	11.56
	10	2	2957.98	0.0361	w/t	3.15	11.56
	11	2	2733.06	0.0368	w/o	3.88	12.00
	11	2	2898.26	0.0368	w/t	2.92	12.00
	12	2	2678.93	0.0365	w/o	3.96	12.82
	12	2	2798.89	0.0361	w/t	3.43	12.82
	13	2	2642.34	0.0363	w/o	4.29	14.28
	13	2	2743.48	0.0357	w/t	3.70	14.28
	14	2	2553.89	0.0348	w/o	5.48	16.67
	14	2	2619.46	0.0359	w/t	4.62	16.67
Other state of art improved logarithmic multipliers							
LeAp [9]	0	3	1990.71	0.0455	w/t	0.98	4.76
REALM [10]	0	3	2383.36	0.0488	w/t	0.75	3.70
	9	3	1572.90	0.0423	w/t	1.06	5.27
HEALM-TA [12]	9	2	1511.39	0.0410	w/t	1.64	5.83
HEALM-SOA [12]	9	2	1577.47	0.0412	w/t	1.38	5.15

in Sec. 3.2. The parameter k indicates the number of bits truncated in the $x + y$ summation. In the *Err. Comp.* column: the "w/t", "w/o" mark represent "with" or "without" error compensation, respectively. By introducing inexact summation, from Table 2, we observe that when compared to the exact multiplier baseline, PAALM can improve the power density and area up to 11.5% and 31.6%, respectively. Also when compared to the conventional ALM design, PAALM will improve the power density up to 34.6% with area overheads and outperforms all the listed state of art works in power density.

For the 16-bit multipliers, the hardware performance and error metrics are demonstrated in Table 3. PAALM reduces 5.7% power density and 70.8% area when compared to the exact multiplier baseline; also reduces up to 14.5% of power density when compared with the ALM design with area overheads. Similar to the 8-bit case, 16-bit PAALM also outperforms all the listed state of art works in respect of the power density.

The error metrics of 8-bit and 16-bit PAALM design are also demonstrated in Table 2 and Table 3. As proved in Sec. 3.1, without error compensation, PAALM shows completely the same error behavior as the conventional ALM. However, different from previous works, the error compensation scheme (mentioned in Sec. 3.2) for PAALM will do little improvement to the mean and peak error, but focus on improving the error bias. We show the curves for the absolute value of the error bias of PAALM with error compensation under 8-bit and 16-bit precision with different k values in Fig. 6. By observing Fig. 6, we can find an obvious convex property of the curves, not monotonically decreasing when we decrease the k value. The reason is that when k is small, the "set one" error compensation scheme will introduce larger and larger positive error to balance the negative error of ALM when we increase k. However, when the error bias reaches the minimum value: -0.17 (when $k = 4$ with 8-bit precision) or 0.08 (when $k = 12$ with 16-bit precision), and we further increase k, the absolute value of the positive error will become larger than which of the negative error, thus leads to a "rebound" of the error bias curve.

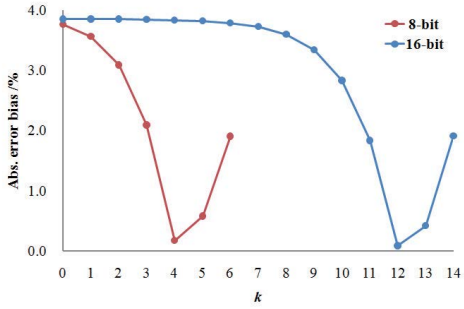


Figure 6: The absolute error bias values of 8-bit and 16-bit error compensated PAALM for different k values

4.3 Evaluation on CNN application

To further evaluate the performance of PAALM in applications which are multiplication-intensive, we first implement the multiplier into a convolutional neural network (CNN) and compare with the fixed-point multiplier baseline and the conventional ALM. The application is developed on a Python platform.

We use CIFAR10 [24] dataset to test the CNN application with approximate multipliers. The network consists of two convolution (CONV) layers and three fully connected (FC) layers. The network is trained with 3000 steps, using double-precision floating point numbers with a batch size of 128 in one step. To test the performance of the approximate multipliers, we replace the floating-point multipliers with PAALM, ALM and fixed-point multipliers in 8-bit precision to do the inference. Since the CONV layer is computation-intensive and FC layer is memory-extensive, we simply replace the multipliers in CONV layers. Fig. 7 shows the classification accuracy comparison of PAALM, conventional ALM and fixed-point multiplier on 1024 images. The accuracy is in percentage.

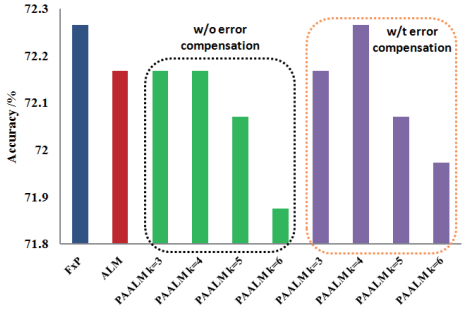


Figure 7: CIFAR10 dataset inference accuracy based on different approximate multipliers

From Fig. 7, we can observe that all the listed multipliers can achieve similar inference accuracy. Without error compensation,

Table 4: PSNR (dB) for images after DCT-iDCT using logarithmic multipliers

Approach	k	Err. Comp.	Lena	Boat	Barbara	House	Pepper	Avg.
FxP (baseline)	/	/	40.3	39.7	39.6	40.1	38.5	39.6
ALM	0	w/o	19.1	18.7	19.3	18.4	18.7	18.8
PAALM (proposed)	0	w/o	19.1	18.7	19.3	18.4	18.7	18.8
	3	w/o	19.0	18.6	19.3	18.4	18.6	18.8
	3	w/t	21.7	21.2	21.9	21.3	21.5	21.5
	4	w/o	19.1	18.7	19.3	18.4	18.7	18.8
	4	**w/t**	**27.5**	**26.5**	**27.0**	**28.8**	**27.0**	**27.4**
	5	w/o	18.4	18.1	18.6	17.1	17.9	18.0
	5	w/t	24.8	23.5	24.5	27.4	24.7	25.0
	6	w/o	16.3	16.1	16.4	14.9	15.9	15.9
	6	w/t	20.5	20.0	20.2	17.4	19.2	19.5

PAALM will lose some accuracy when inexact summation is introduced; while when $k < 5$, PAALM can perform as well as the conventional ALM. If error compensation is considered, we notice that when $k = 4$, PAALM can outperform the conventional ALM, and achieves the same inference accuracy as the fixed-point multiplier baseline. This is due to the extremely low error bias (almost zero) as demonstrated in Fig. 6.

4.4 Evaluation on image processing application

Now, we show how the proposed *PAALM* design compare to the conventional ALM and fixed-point multiplier baseline in a multimedia application. Discrete cosine transformation (DCT) is a commonly used lossy image compression method. The quality of the compressed images is usually evaluated using metrics such as PSNR (peak signal noise ratio) and higher PSNR value represents better image quality. We implement the proposed PAALM design with 8-bit precision in the DCT-iDCT (inverse DCT) workloads, and compare with the conventional ALM and fixed-point multiplier on five example images. As the error bias behavior of error compensated PAALM with different k value shows a convex trend, and performs the best when k equals to 4. We just list the experimental results from $k = 3 \sim 6$.

We show results of image compression in Table 4. The results show that without error compensation, the image quality will worsen when the value of k increases. While if error compensation is introduced, when k equals to 4, thanks to a more balanced error behavior, we can achieve the best image quality of 27.4dB in average, improving 8.6dB when compared to the conventional ALM design. Note that the error compensation for PAALM introduces almost no area and power overheads, thus we obtain an approximate multiplier which achieves a balance of resource efficiency and low power density.

5 CONCLUSION

In this work, we have proposed a novel power density aware approximate logarithmic multiplier (*PAALM*) design to mitigate the high power density issue occurred in the existing ALM designs. By re-designing the high computing switch activities of existing ALM designs based on equivalent mathematical formula, the power density could be reduced with no accuracy loss but with some area overheads. Experimental results showed that the proposed PAALM design could improve the power density and the area at the same time when compared with the fixed-point multiplier baseline. The PAALM design could also achieve extremely low error bias with error compensation. Furthermore, the PAALM showed good performance when implemented in CNN and image processing applications.

REFERENCES

[1] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems*, pp. 1269–1277, 2014.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[3] G. Zervakis, H. Amrouch, and J. Henkel, "Design automation of approximate circuits with runtime reconfigurable accuracy," *IEEE Access*, vol. 8, pp. 53522–53538, 2020.

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

[5] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2015.

[6] G. Zervakis, H. Saadat, H. Amrouch, A. Gerstlauer, S. Parameswaran, and J. Henkel, "Approximate computing for ml: State-of-the-art, challenges and visions," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, ASPDAC '21, (New York, NY, USA), p. 189ÍC196, Association for Computing Machinery, 2021.

[7] R. Goldman, K. Bartleson, T. Wood, K. Kranen, V. Melikyan, and E. Babayan, "32/28nm educational design kit: Capabilities, deployment and future," in *2013 IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, pp. 284–288, IEEE, 2013.

[8] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, no. 4, pp. 512–517, 1962.

[9] Z. Ebrahimi, S. Ullah, and A. Kumar, "Leap: Leading-one detection-based soft-core approximate multipliers with tunable accuracy," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 605–610, IEEE, 2020.

[10] H. Saadat, H. Javaid, A. Ignjatovic, and S. Parameswaran, "Realm: reduced-error approximate log-based integer multiplier," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1366–1371, IEEE, 2020.

[11] M. S. Ansari, B. F. Cockburn, and J. Han, "A hardware-efficient logarithmic multiplier with improved accuracy," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 928–931, IEEE, 2019.

[12] S. Yu, M. Tasnim, and S. X.-D. Tan, "HEALM: Hardware-efficient approximate logarithmic multiplier with reduced error," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 37–42, IEEE, 2022.

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

[14] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power-and area-efficient approximate wallace tree multiplier for error-resilient systems," in *Fifteenth International Symposium on Quality Electronic Design*, pp. 263–269, IEEE, 2014.

[15] B. S. Prabakaran, S. Rehman, M. A. Hanif, S. Ullah, G. Mazaheri, A. Kumar, and M. Shafique, "Demas: An efficient design methodology for building approximate adders for fpga-based systems," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 917–920, IEEE, 2018.

[16] S. Ullah, S. Rehman, B. S. Prabakaran, F. Kriebel, M. A. Hanif, M. Shafique, and A. Kumar, "Area-optimized low-latency approximate multipliers for fpga-based hardware accelerators," in *Proceedings of the 55th Annual Design Automation Conference*, pp. 1–6, 2018.

[17] S. Ullah, S. S. Murthy, and A. Kumar, "Smapproxlib: library of fpga-based approximate multipliers," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2018.

[18] S. Hashemi, R. I. Bahar, and S. Reda, "Drum: A dynamic range unbiased multiplier for approximate applications," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 418–425, IEEE, 2015.

[19] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 23, no. 6, pp. 1180–1184, 2014.

[20] H. Saadat, H. Bokhari, and S. Parameswaran, "Minimally biased multipliers for approximate integer and floating-point multiplication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2623–2635, 2018.

[21] S. Yu, Y. Liu, and S. X.-D. Tan, "COSAIM: Counter-based stochastic-behaving approximate integer multiplier for deep neural networks," in *Proc. Design Automation Conf. (DAC)*, pp. 1–6, Dec. 2021.

[22] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, 2018.

[23] G. Zervakis, I. Anagnostopoulos, S. Alsalamin, O. Spantidi, I. Roman-Ballesteros, J. Henkel, and H. Amrouch, "Thermal-aware design for approximate dnn accelerators," *IEEE Transactions on Computers*, pp. 1–1, 2022.

[24] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.