Multi-ALM: Run-time Multi-Level Reconfigurable Approximate Logarithmic Multiplier

Maliha Tasnim
Electrical and Computer Engineering, University of California at Riverside
Riverside, CA, USA
mtasn004@ucr.edu

Chinmay Raje
Electrical and Computer Engineering, University of California at Riverside
Riverside, CA, USA
craje001@ucr.edu

Sheldon X.-D. Tan
Electrical and Computer Engineering, University of California at Riverside
Riverside, CA, USA
stan@ece.ucr.edu

2024 25th International Symposium on Quality Electronic Design (ISQED) | 979-8-3503-0927-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISQED60706.2024.10528751

***Abstract*—This paper presents a novel multi-level approximate logarithmic multiplier (ALM) named *multi-ALM* that offers dynamic run-time re-configurability for various precision, performance, and power requirements. While previous research on ALM designs has focused primarily on trade-offs between performance, energy, and accuracy during design time, our work addresses the critical need for dynamic re-configurability at run-time to enable efficient power, performance, and quality management at the architecture and algorithm levels. The proposed multi-ALM is based on an innovative iterative formulation of logarithmic multiplication, resulting in a Taylor series-like formula. This formulation facilitates straightforward trade-offs between multiplication accuracy and power consumption. We demonstrate that traditional ALMs can be considered level 1 ALMs within the multi-ALM framework. Furthermore, we introduce a new four-level ALM MAC array architecture design that enables run-time reconfigurable MAC (Multiply-Accumulate) computing with customizable accuracy and performance settings. This architecture empowers system designers to adapt the ALM functionality on-the-fly, tailoring it to the specific requirements of different applications and scenarios. Numerical results show that 8-bit two-level *multi-ALM* can achieve up to $17.22\times$, $2.78\times$, and $1.37\times$ improvement in mean error, peak error, and power consumption respectively over the baseline *ALM*, while the area increases by $1.40\times$. 16-bit two-level *multi-ALM* can achieve up to $17.5\times$ and $2.75\times$ improvement in mean error and peak error over *ALM*. Furthermore, we evaluate the proposed *multi-ALM* design in several multiplication and accumulation (MAC) applications and discrete cosine transformation (DCT) application. The result shows that, *multi-ALM* can effectively trade-off performance such as throughput per unit of resource consumption to provide better accuracy of MAC computation, and quality of reconstructed image upon the conventional fixed configuration approximate multipliers.**

***Index Terms*—Run time reconfiguration, approximate computing, parallel computing**

I. Introduction

Approximate computing has emerged as a powerful approach that allows efficient trade-offs between energy, performance, and accuracy or quality metrics. By introducing an additional knob for metric optimization, it offers valuable benefits during both design-time and run-time stages. The growing popularity of approximate computing is largely attributed to the increasing prevalence of error-tolerant applications, particularly in fields like machine learning and multimedia workloads [1]. These applications heavily rely on multiplication operations, prompting significant research efforts towards the design of hardware-efficient multipliers. The primary objective in the design of approximate multipliers is to minimize power consumption and area while minimizing accuracy loss. This pursuit of hardware efficiency is essential to ensure the optimal performance of error-tolerant applications while maintaining an acceptable level of accuracy.

Several approximate multiplier designs have been recently introduced [2]–[13]. These approximate multipliers either employ ad-hoc truncation or reduction methods or utilize mathematically formulated approximation schemes. However, most of the existing methods lack systematic configurability for achieving a balance between accuracy, area, power consumption, and latency.

In contrast, a category of approximate multipliers that are mathematically formulated includes logarithmic multipliers, which transform multiplication operations into a combination of shift and addition operations. The logarithmic multiplier leverages the inherent approximate nature of logarithmic operations and allows for easy manipulation of accuracy while trading off area, latency, and power consumption. The concept of the logarithmic multiplier was originally proposed by Mitchell [14]. Since then, numerous variations of approximate logarithmic multipliers (ALMs) have been introduced to enhance Mitchell's initial work [9], [10], [12], [15], [16]. Most of these approaches primarily focus on achieving trade-offs between accuracy and various sources of error compensation during the design phase, typically based on single-level approximation techniques (which we will elaborate on later).

In this work, we proposed a novel run-time re-configurable multi-level ALM design, called *Multi-ALM*, which allows the trade-off between performance/power and accuracy at the run-time. The key contributions of this work are listed as follows:

1. First, we present a novel iterative formulation of logarithmic multiplication, akin to the Taylor series formula, enabling easy trade-offs between multiplication accuracy and efficiency by selectively controlling the number of terms used. Here, we show that conventional ALMs essentially serve as level 1 ALMs within this new formulation. Furthermore, we introduce multi-level logarithmic multiplication through our newly developed *multi-ALM* approach, incorporating more than one level. The multi-ALM technique offers an improved trade-off between accuracy and performance/power, making it a valuable tool for optimizing hardware designs for various computing applications.
2. Second, we introduce a novel 4-level ALM MAC (Multiplier and Accumulator) array architecture design, enabling run-time reconfigurability of ALMs. This versatile architecture can be dynamically configured to support different ALM or ALM MAC arrays, offering varying levels of accuracy and performance based on the specific application requirements. By providing this level of flexibility and adaptability, our 4-level ALM MAC array represents a significant advancement in implementing reconfigurable ALMs for various computing tasks.

Numerical results show that 8-bit two-level *multi-ALM* can achieve up to $17.22\times$, $2.78\times$ and $1.37\times$ improvement in mean error, peak error, and power consumption respectively over the baseline *ALM*, while the area increases by $1.40\times$. 16-bit two-level *multi-ALM* can achieve up to $17.5\times$ and $2.75\times$ improvement in mean error and peak error over *ALM*.

This work is supported in part by NSF grant under No.CCF-2113928.

Furthermore, we evaluate the proposed *multi-ALM* design in a multiplication-accumulation (MAC) applications, and a discrete cosine transformation (DCT) application. The result shows that, *multi-ALM* can effectively trade-off performance such as throughput per unit of resource consumption to provide better accuracy of MAC computation, and quality of reconstructed image upon the conventional fixed configuration approximate multipliers with small hardware overhead to introduce runtime re-configurability.

II. Preliminaries

Approximate unsigned integer multiplier has been inspected in several studies and various efficient designs have been proposed recently. Earlier ad-hoc based approximate designs, such as recursive multipliers [2] consisting of 2×2 multiplication blocks involved simplification of Wallace tree [3], and partial product generation/summation [4]–[6]. However, recently, many approximate multipliers have been developed based on the classic approximate logarithmic multiplier (also known as *ALM*) proposed by Mitchell. In a baseline *ALM* design, the two inputs X and Y are first represented by the following format: $2^{k_x} \cdot (1+x)$ and $2^{k_y} \cdot (1+y)$, respectively. Then the multiplication result, C_{ALM} can be approximated as (1).

$$C_{ALM} = \begin{cases} 2^{k_x+k_y} \cdot (1+x+y), & x+y < 1, \\ 2^{k_x+k_y+1} \cdot (x+y), & x+y \geq 1 \end{cases} \quad (1)$$

The hardware implementation of ALM design requires four modules:a leading-one detectors (LOD) to find the leading bit '1' as the integer part; two barrel shifters to re-align the rest of the bits as the fraction part; two adders to sum the two aligned fractions and integer parts up as $k_x + k_y + x + y$; and finally, a shifter to realign the bits of the fractions sum according to integer sum. *ALM* shows good overall performance and has flexibility for trade-offs among area, power, and accuracy [14].

III. The Proposed Multi-Level Approximate Multiplier

In this section, we present the new multi-level reconfigurable approximate multiplier, called *multi-ALM*. We first present the new iterative formula for the ALM algorithm and then introduce our reconfigurbale two-level ALM design.

A. New iterative definition of ALM accuracy

Let us first define $X_1 = (1+x_1) \times 2^{k_{x_1}}$ and $Y_1 = (1+y_1) \times 2^{k_{y_1}}$. The exact logarithmic multiplication, $X_1 \times Y_1$ of two N-bit number, X_1 and Y_1 can be written as

$$X_1 \times Y_1 = (1 + x_1 + y_1 + x_1 \times y_1) \times 2^{k_{x_1}+k_{y_1}} \quad (2)$$

Here, x and y are mantissa and k_x and k_y are exponent part of X and Y respectively. Now we define a new operation, call *unified-ALM*, or UALM(X, Y). This multiplication is approximated by removing $x_1 \times y_1$ term in (2). Then we have,

$$\text{UALM}(X_1, Y_1) = (1 + x_1 + y_1) \times 2^{k_{x_1}+k_{y_1}} \quad (3)$$

Similar to the existing ALM methods, the unified-ALM consists of total two leading one detectors, two $N-1$ bit adders, one half-adder and a shifter which results in significant reduction in area and energy compared to both exact multiplier and conventional ALM but at cost of certain accuracy loss. Note that such unified ALM definition is a bit different than existing ALM formula (1) as we do not need to consider the two scenarios in the new method. The accuracy loss in both conventional ALM in (1) and unified-ALM in (3) mainly comes from removing the mantissa multiplication, $x_1 \times y_1 \times 2^{k_{x_1}+k_{y_1}}$ in (2). Specifically, the exact log multiplication of two N-bit numbers, X_1 and Y_1 can be re-arranged as in the following (4a) as addition of $UALM(X_1, Y_1)$ with an error correcting term $E_{C1} = (x_1 \times 2^{k_{x_1}})(y_1 \times 2^{k_{y_1}})$, which essentially is the multiplication of two new number $X_2 = x_1 \times 2^{k_{x_1}}$ and $Y_2 = y_1 \times 2^{k_{y_1}}$ as shown in (4b).

$$\begin{aligned} & X_1 \times Y_1 \\ & = (1 + x_1 + y_1) \times 2^{k_{x_1}+k_{y_1}} + (x_1 \times 2^{k_{x_1}})(y_1 \times 2^{k_{y_1}}) && (4a) \\ & = \text{UALM}(X_1, Y_1) + X_2 Y_2 && (4b) \\ & = \text{UALM}(X_1, Y_1) + (1 + x_2 + y_2 + x_2 y_2) 2^{k_{x_2}+k_{y_2}} \\ & = \text{UALM}(X_1, Y_1) + \text{UALM}(X_2, Y_2) + X_3 Y_3 && (4c) \end{aligned}$$

Now, this multiplication for E_{C1} can be further rearranged again as a new UALM(X_2, Y_2) plus a new error correcting term $E_{C2} = X_3 Y_3$ as shown in (4). If we repeat this process, we will end up with a new iterative definition of the multiplication of $X \times Y$ in terms of ALM shown below:

$$X_1 \times Y_1 = \sum_{i=1}^{l+1} \text{UALM}(X_i, Y_i); 0 \leq l < l_{max} \quad (5)$$

where l represents the number of level of the $UALM$, and the maximum level we can have is $l_{max} + 1$. Now we introduce new concept of ALM, called, *multi-level ALM* design and (5) basically defines $(l+1)^{th}$ level ALM multiplication of X and Y. Here, $l_{max} = N - 1$, and N is the bit width of the input multiplicands of a UALM. We note that the iterative formula (5) works for both cases defined in (1) as we start with the exact solution defined in (4a). The traditional ALM is essentially a level-1 ALM in the new design. The multi-level ALM formula in (5) basically gives us more *opportunity to trade the power/area with the accuracy* as shown in the next subsection.

B. Bit segmentation in logarithmic multiplier

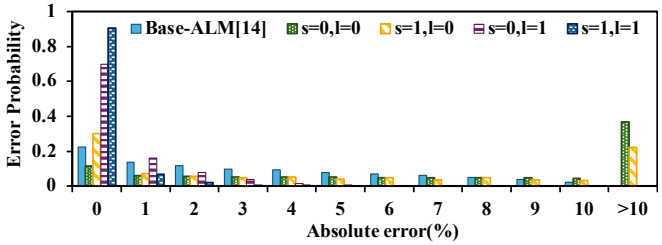


Fig. 1: Absolute error distribution of 8-bit log multiplier

Let us consider multiplication of two 8-bit numbers, X_1 and Y_1 with two level of segmentation ($s = 1$) as an example shown in (6a). Segmentation is to partition of data into many segments. The multiplication of two 8-bit numbers can be computed as shifted summation of four partial products as shown in (6b).

$$\begin{aligned} & X_1 = X_{1,1} + X_{1,2} \times 2^4; Y_1 = Y_{1,1} + Y_{1,2} \times 2^4 && (6a) \\ & X_1 \times Y_1 = X_{1,1} \times Y_{1,1} + X_{1,2} \times Y_{1,1} \times 2^4 \\ & \qquad + X_{1,1} \times Y_{1,2} \times 2^4 + X_{1,2} \times Y_{1,2} \times 2^8 && (6b) \end{aligned}$$

Each of the partial products in (6b) can be computed with one UALM and then properly shifted-added together to calculate the final approximate multiplication, of $X_1 \times Y_1$ as shown in (7a). To reduce the error of this approximate multiplication in (7a), another set of UALMs can be introduced to calculate 2-level *multi-ALM* as in (7b).

$$X_1 \times Y_1 = \sum_{k=1}^{2}\sum_{j=1}^{2}(X_{1,j} \times Y_{1,k}) \times 2^{4(j+k-2)} \tag{7a}$$

$$\approx \sum_{k=1}^{2}\sum_{j=1}^{2}\left(\text{UALM}(X_{1,j}, Y_{1,k}) + X_{2,j} \times Y_{2,k}\right) \times 2^{4(j+k-2)} \tag{7b}$$

$$\approx \sum_{k=1}^{2}\sum_{j=1}^{2}\left(\text{UALM}(X_{1,j}, Y_{1,k}) + \text{UALM}(X_{2,j}, Y_{2,k})\right) \times 2^{4(j+k-2)} \tag{7c}$$

where, we have

$$X_{1,j} = (1 + x_{1,j}) \times 2^{k_{x_{1,j}}};\ Y_{1,j} = (1 + y_{1,j}) \times 2^{k_{y_{1,j}}}$$
$$X_{2,j} = x_{1,j} \times 2^{k_{x_{1,j}}};\ Y_{2,j} = y_{1,j} \times 2^{k_{y_{1,j}}}$$

In general, given ALM level l and segmentation number s, and precision in each segment N, the precision of the given numbers are $N \times 2^s$. Then the multiplication of two numbers can be implemented with $(s+1)^2$ numbers of N-bit UALMs as shown in (8).

$$X_1 \times Y_1 \approx \sum_{k=1}^{s+1}\sum_{j=1}^{s+1}\left(\sum_{i=1}^{(l+1)<N} \text{UALM}(X_{i,j}, Y_{i,k}) \times 2^{N(j+k-2)}\right) \tag{8}$$

where

$$X_1 = \sum_{j=1}^{s+1} X_{1,j} \times 2^{N_s(j-1)}$$
$$Y_1 = \sum_{k=1}^{s+1} Y_{1,k} \times 2^{N_s(k-1)}$$
$$X_{1,j} = (1 + x_{1,j}) \times 2^{k_{x_{1,j}}};\ Y_{1,j} = (1 + y_{1,j}) \times 2^{k_{y_{1,j}}}$$
$$X_{i,j} = x_{i-1,j} \times 2^{k_{x_{i-1,j}}};\ Y_{i,j} = y_{i-1,j} \times 2^{k_{y_{i-1,j}}}$$

Fig. 1 shows the absolute error distribution of 8-bit multiplication for *base-ALM* [14] and the proposed error-reducing algorithm in (4)-(8), under different level l and different bit segmentation s. The probability of error (Y axis) within the range of $0\% - 1\%$ (which is more desirable) increases as we increase the number of bit-segmentation from $s = 0$ to $s = 1$. This is also the case with increasing level l as well. The best results indeed comes from $l = 1$ and $s = 1$ as the probability of absolute error that is larger than one, goes to almost zero.

On the other hand, the number of partial product will increase with more bit segmentation. However each partial product can be implemented with smaller log multiplier, which will reduce the size of leading one detectors (LODs) and adders used in each unit log multiplier (UALM).

C. The proposed re-configurable 4-level ALM MAC design

Based on the new definition of multi-level ALM in (8), we propose a new *multi-ALM* MAC array design by increasing the level of log multiplier (l) and bit segmentation value (s). The new design can be reconfigured from 16×4 parallel N-bit ALM array to a single $2N$-bit multiplier with different accuracy and precision.

Fig. 2 shows the architecture of our proposed 4 level ALM MAC array design, which consists of total 4×4 units of N-bit **UALM** with $l_{max} = 3$. The control module of *multi-ALM* is a finite state machine (FSM) with three control input- '$start$', 's'' and 'l' as well as three bit output signal 'done', 'ren' and 'wen' and an output address bus '**raddr**'. The multiplication starts at the positive edge of clock after '$start$' bit goes high. The single bit control input 's' can be configured during runtime to '0' for single precision (N-bit) multiplication, and '1' for double precision ($2N$-bit) multiplication. The two-bit control input 'l' can be configured during run-time to values ['00','01','11'] for single, two-level and four level iteration of ALM respectively, depending on requirement of accuracy of the application.

There are two input data register **RX** and **RY** to store the MAC vectors of multiplicands. Each register can store up to 4 vectors of length P and bit-width of N. During double precision computation, each of **RX** and **RY** stores a single vector of length P and bitwidth of $2N$. The **Controller FSM** output **'raddr'** selects the data (**DX** and **DY**) to be sent to **UALM array** at each clock cycle when 'ren' is set to '1'.

The **UALM array** has 16 ALM units labeled as **UALM**. Each **UALM** has two sets of multiplexers to select multiplicands 'X_1'('Y_1') from **'DX'('DY')** or **'LX'('LY')** depending on the value of 's' and 'l'. The output of multiplexers are sent to a leading-one detector (**LOD**) unit that calculates the exponent 'k_x'('k_y') and mantissa 'x'('y') and next level input 'X_2'('Y_2') according to (4). The exponent and mantissa from **LOD** is sent to log-multiplier unit (**LM**) for computing approximate log multiplication of X_1 and Y_1 according to (3). The other output of **LOD**s namely $X_2(Y_2)$ is sent as data inputs **LX(LY)** for next **UALM** unit. The multiplication result '**m**' from **LM** unit is sent to a **Row accumulator** unit. The **Row accumulator** unit has four accumulators, one for each column of **UALM** array. Each accumulator adds multiplication result, '**m**(t)' of current clock cycle 't' from UALM units in four rows of each column to the accumulation result of last clock cycle ($M_j(t-1)$). The result of accumulators are sent to a **column shifter+adder** unit that shifts and adds four partial sums (M_j; j=1,..,4) according to the rules of partial product. Finally, the last multiplexer stage selects between concatenated outputs (M_1, M_2, M_3, M_4) or cumulative sum of partial products **S** and sends to an output data register **RZ**. When entire MAC vector of length P is finished with computation, a '$done$' signal is set as high by the **controller FSM**.

IV. Numerical results and discussions

In this section, we evaluate the performance of the proposed *multi-ALM* and compare it against the conventional *ALM* (approximate logarithmic multiplier) baseline [14] and exact multiplier.

A. Experimental setup

We have implemented our proposed multi-level ALM with Verilog HDL and synthesized this in Synopsys Design Compiler using 32nm technology. We have compared our design for accuracy, area, and power with three state-of-the-art approximate multipliers, and also an exact multiplier which is our baseline. Our proposed *multi-ALM* is defined with three parameters, s, l, and N. The parameter s and l are reconfigurable during runtime. These two parameters configure the segmentation of input data (partitioning of the data) and level of iteration (level of accuracy), respectively. The last parameter, N, denotes the bit-precision of each unit ALM multiplier. *Multi-ALM* dynamically sets the precision of each input multiplicand as $2^s \times N$.

We have measured the mean error and peak error of multiplication for each configuration in *multi-ALM* and using all possible combination of input multiplicands within the range of $[1, 2^N - 1]$ for both $N = 8$ and $N = 16$. We have also designed a finite state machine (FSM) using Verilog HDL that computes MAC of 2 input vectors (each of length 8) using different multipliers. The power and area of such MAC-FSM

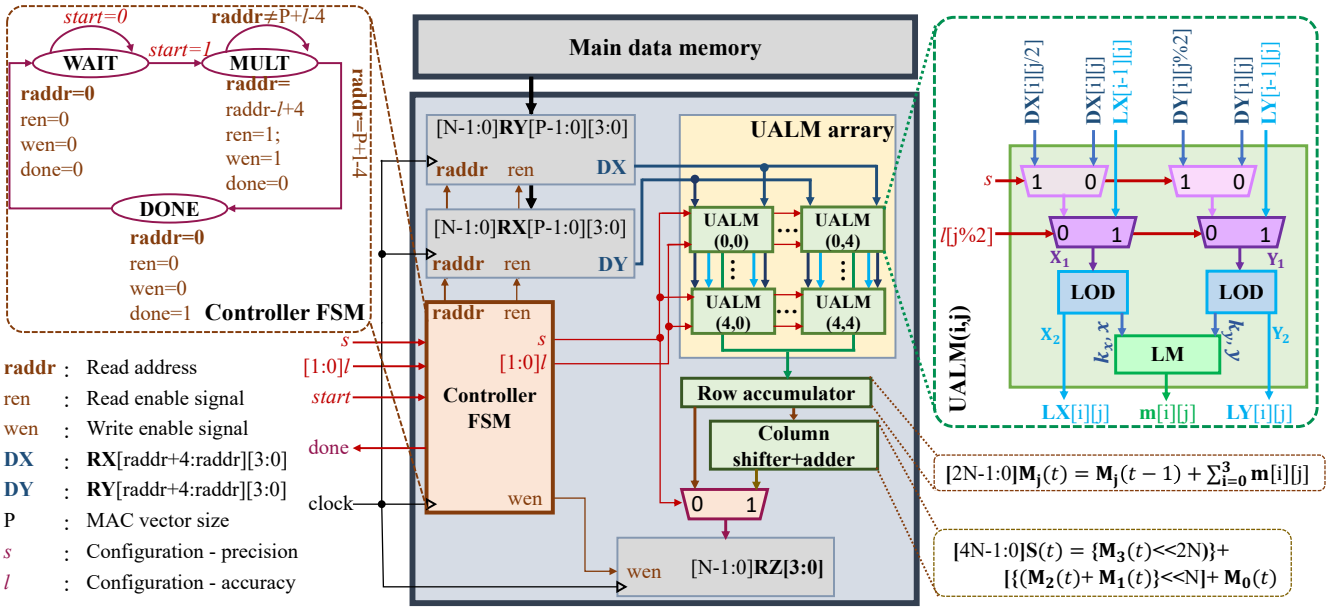


Fig. 2: The proposed 4 level multiple precision ALM MAC architecture

have been measured using Synopsys Design Compiler. We developed another behavioral simulation model to measure the accuracy of MAC application for such $100M$ pairs of randomly generated MAC of different data precision.

Finally, the proposed design has been evaluated for a very frequently used signal processing application of discrete cosine transform (DCT). We have designed a finite-state-machine (FSM) to transform a greyscale image of 512×512 pixels with DCT and compress it by 66.75% . We have computed the peak signal-to-noise ratio (PSNR) and mean square error (MSE) for all configurations of our proposed *multi-ALM* as well as other conventional approximate multipliers.

B. Accuracy, energy and power of single multiplication

TABLE I
ACCURACY, AREA AND POWER OF VARIABLE PRECISION RE-CONFIGURABLE ALM. S= $\#\ segmentation - 1$; L= $\#iteration - 1$ IN LOG MULTIPLIER

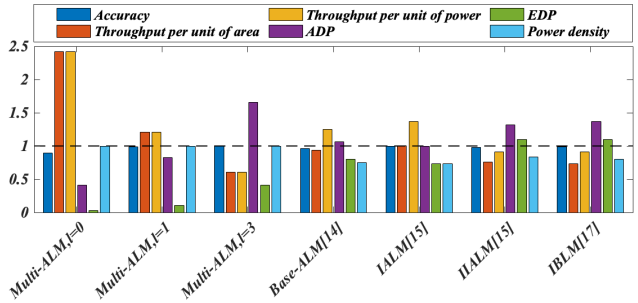
multipli-cand bit-width	s	l	mean error(%)	peak error(%)	area(μm^2)	power(μW)
8	Exact		0	0	2067.86	53.32
	Base-ALM [14]		3.79	11.11	820.63	72.83
	0	0	8.91	24.80	584.67	26.60
		1	0.83	6.10	1144.79	59.328
		2	0.071	1.48	1522.32	84.496
		3	0.0048	0.35	1877.36	104.061
	1	0	5.90	21.71	786	39.84
		1	0.22	4.00	1299	65.70
		2	0.0024	0.44	1552.59	77.07
		3	0.00	0.00	1692.15	88.00
16	Exact		0	0	8753.48	323.52
	Base-ALM [14]		3.85	11.11	1825.92	110.91
	0	0	9.41	25	1521.80	63.54
		1	0.99	6.25	2908.06	139.268
		2	0.11	1.56	4320.63	221.94
		3	0.012	0.39	5530.67	298.48
	1	0	8.99	24.8	2017.22	109.49
		1	0.85	6.10	3793.6	230.09
		2	0.072	1.48	4781.93	299.65
		3	0.005	0.35	5785.91	346.50

Table I shows the mean error and peak error percentage of exact multiplier, *base-ALM* and *multi-ALM* for 8-bit and 16-bit multiplication. The 8-bit single segment ($s = 0$) one-level ($l = 0$) *multi-ALM* can provide area-reduction and power-reduction of around 28.75% and 62.62%, respectively, over *base-ALM*, while the mean and peak error increases up to $2.4\times$ and $2.23\times$, respectively. However, Table I also shows that, the mean and peak error can be further improved for any value of s with multiple level of iteration ($l \geq 2$) in *multi-ALM*. The two-level *multi-ALM* ($l = 1$) can significantly reduce peak error by 78.10% and mean error by $94.03\%\times$ with an increase in area of $1.39\times$ over *base-ALM*. The power consumption in single-segment two-level *multi-ALM* is also 18.54% lower than *base-ALM*. The error metric can be reduced further by increasing segmentation (s) and/or increasing level of iteration (l) while simultaneously at increased area and power as shown in Table I.

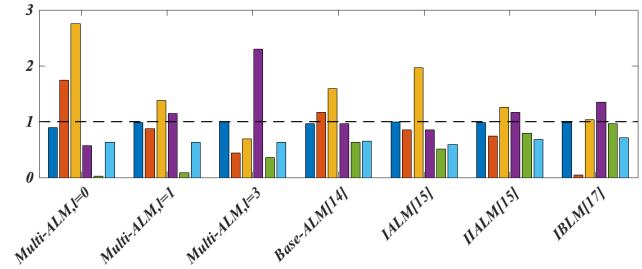
The 16-bit two-level *multi-ALM* can provide a similar improved error metric over the 16-bit *base-ALM*. While the mean and peak errors are higher in one-level ($l = 0$) *multi-ALM*, the area and power consumption are reduced by 16.66% and 42.71% over *base-ALM*. The mean and peak error improves significantly in two-level ($l = 1$) *multi-ALM* with 74.29% and 64.10% reduction over *ALM*, respectively. However, the area and the power consumption in two-level *multi-ALM* are also increased up to $1.91\times$ and $1.29\times$, respectively over *base-ALM*. The accuracy of 16-bit *multi-ALM* improves with a higher number of segmentation and/or iteration levels at the expense of increased area and power consumption.

C. Application-1: multiplication-accumulation (MAC)

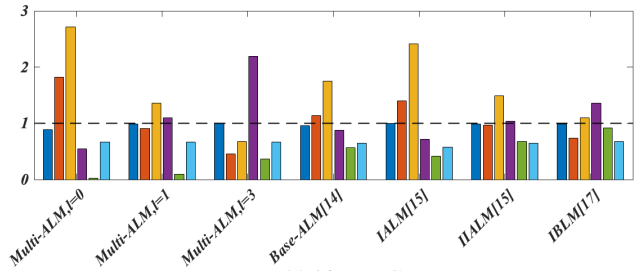
We have further compared *multi-ALM* with the exact multiplier and existing ALM approximate multipliers: *base-ALM* [14], *IALM* [15], *IIALM* [15], *IBLM* [17] for multiplication-accumulation (MAC) application, which is one of the most frequently used block in any digital signal processing application. We have designed a finite-state-machine (FSM) in Verilog HDL that takes two vectors, each of length eight as inputs. The elements of vectors can be integers of different precision (8-bit, 16-bit, or 32-bit). We have implemented the FSM with three different Multi-ALM designs- 8-bit Multi-ALM, 16-bit Multi-ALM and 32-bit Multi-ALM as well as with two baseline multipliers- exact and ALM. For each input data precision, we have generated $100M$ pairs of random row vectors, each of length 8 (**RX** and



(a) 8-bit MAC



(b) 16-bit MAC



(c) 32-bit MAC

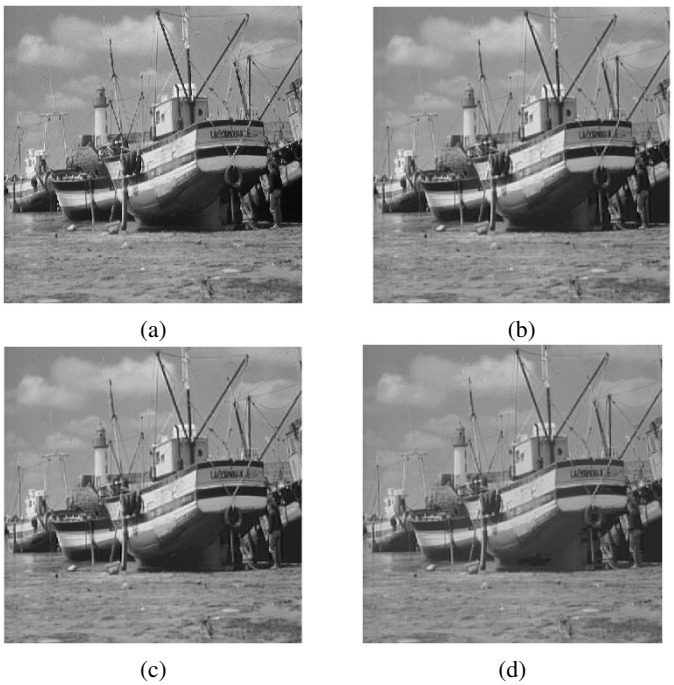
Fig. 3: Performance comparison of MAC application of vector length 8, and of three different bit-precision implemented with *multi-ALM* (three different configuration: $l = 0$, $l = 1$ and $l = 3$), *base-ALM* [14], *IALM* [15], *IIALM* [15], *IBLM [17]*

RY in Fig. 2). The comparison of accuracy and hardware resource consumption in different MAC designs normalized to that of the baseline exact multiplier is shown in Fig. 3. We have computed five different hardware resource consumption and performance metrics, namely throughput per unit of area, throughput per unit of energy, area delay product (ADP=$area \times \#of\ clock\ cycle \times clock\ period$), energy delay product (EDP=$energy \times \#\,of\ clock\ \ cycle \times \ clock\ \ period$) and power density ($\frac{power}{area}$). From our comparison of estimated average hardware resource consumption, we can see that, *multi-ALM* can be configured at $l = 0$ to provide a high throughput per unit of area, which is $1.98\times$ higher on average than exact multiplier and $2.09\times$ higher on average than the existing ALM approximate multipliers.

The same configuration of *multi-ALM* at $l = 0$ can also reduce ADP by 48.72% on average and reduce EDP by 97.52% from those of the exact multiplier based MAC. When compared to conventional approximate multipliers, the average reduction in ADP and EDP are 50.5% and 96.11% respectively. However, the accuracy in this configuration also reduces by 10.75% from that in the exact multiplier and 9.03% from other approximate ALM multipliers. The accuracy of *multi-ALM* can be configured to improve to be around 98.82% by changing the configuration parameter to $l = 1$, which is 2.58% higher than the *base-ALM*. However, the throughput per unit of hardware resource in this configuration reduces significantly and becomes close to that in exact multipliers. On average, the throughput per unit area and per unit energy in *multi-ALM* based MAC are $1.21\times$ and $1.31\times$ of the exact multiplier based MAC respectively. The same metrics in *multi-ALM* based MAC are $1.09\times$ and $0.99\times$ on average, respectively, when compared to that in the other approximate multiplier based MAC. The ADP and EDP in *multi-ALM* based MAC in this configuration reduce by -2.55% and 89.14%, respectively when compared to the exact multiplier based MAC and they reduce by 1.13% and 86.58% on average when compared to other approximate ALM multiplier based MACs. The accuracy of *multi-ALM* can be further improved by making $l = 3$. From Fig. 3, it is evident that the run-time reconfigurability of *multi-ALM* can be achieved by introducing hardware overhead that is very close to the exact multiplier.

The last performance metric we have computed is power-density which reduces by 32.95% on average in *multi-ALM* based MAC from that in exact multiplier based MAC and it only increases by 5.29% on average compared with the approximate ALM multiplier based MAC.

D. Application-2: 8×8*-point two dimensional DCT*



(a) (b) (c) (d)

Fig. 4: Image compression using 8×8 DCT: (a) Original image; (b-d) DCT using *multi-ALM* multiplier, $l = 3, 1, 0$ respectively

We have further implemented discrete cosine transform (DCT) which is a commonly used lossy image compression method with our re-configurable MAC and existing other state-of-the-art approximate ALM multipliers [14], [15], [17]. We perform the transformation of a grey-scale image of resolution 512×512 pixels. The quality of the compressed images has been evaluated using metrics such as PSNR (peak signal-to-noise ratio) and MSE (mean square root error). We have implemented a DCT compression module in Verilog HDL using our proposed re-configurable *multi-ALM* based MAC-FSM with different configurations as well as other existing fixed-configuration approximate ALM multiplier based MAC-FSM. The module computes 8×8 2-DCT on each 8×8 pixel blocks of the original image and outputs compressed

TABLE II
8X8 DCT COMPUTATION, COMPRESSION 68.75%

PE multiplier	Configuration (l)	PSNR	MSE	Throughput/Area	Throughput/Power	Power density
Multi-ALM	0	26.0654	160.9269	19.6045	0.0223	0.0497
	1	28.5176	91.4783	10.3789	0.0118	0.0497
	3	28.6361	89.0173	5.3467	0.0061	0.0497
Base-ALM [14]	-	27.57	118.3	10.0335	0.0225	0.0473
IALM [15]	-	28.49	92.05	11.1473	0.0282	0.0465
IIALM [15]	-	28.51	98.62	8.3839	0.0154	0.0482
IBLM [17]	-	28.03	102.45	7.5168	0.0123	0.0487

DCT transformed data. We then compared the quality of the compression by reconstructing the compressed image and comparing the PSNR (peak signal-to-noise ratio) and MSE (mean squared errors) with respect to the original image. Fig. 4 shows the reconstructed image from the compressed data, which were generated by using different configurations in our proposed *multi-ALM* based DCT module.

Table II shows the performance comparison (throughput per unit of power and area, power density) with respect to the reconstructed image quality (PSNR and MSE). On average, *multi-ALM* in $l = 0$ configuration can provide 116.48% higher throughput per unit of area and 26.04% higher throughput per unit of power than other existing approximate multipliers. However, in this configuration, the PSNR of the reconstructed image from the *multi-ALM* based DCT module is on average 4.38% lower and MSE is 62.24% higher than other approximate ALM multiplier. The quality of the reconstructed image can be further improved by increasing the configuration parameter value to $l = 1$ in *multi-ALM*. The PSNR and MSE in this configuration improve by 9.41% and 43.15%, respectively than that in the base configuration of $l = 0$ in *multi-ALM*. However, such improvement in signal quality comes from trading off 47.06% lower throughput per resource consumption. Also, the throughput in this configuration is still 1.16% and 2.61% higher in terms of per unit area and per unit power consumption, respectively when compared to those in other approximate ALM multipliers. The accuracy of reconstructed image can be further improved by trading off more throughput in $l = 3$ configuration of *multi-ALM*. However, the power density remains almost constant in all accuracy levels. On average, the power density in our proposed runtime re-configurable *multi-ALM* based DCT module is 2.12% lower than in other fixed configuration approximate multiplier based DCT modules.

V. CONCLUSION

In this article, we have proposed a new multiple-level reconfigurable approximate logarithmic multiplication design, called *multi-ALM*. The *multi-ALM* is based on a novel iterative formulation of the approximate logarithmic multiplication, which is mathematically proven, and a Taylor series-like formula to trade-off between accuracy and performance/power. *Multi-ALM* gives a new opportunity to trade off the accuracy with the power/performance in a mathematical and progressive way. We then present the new multi-level ALM MAC array architecture design to implement run-time reconfigurable MAC computing. Numerical results show that 8-bit two-level *multi-ALM* can achieve up to $17.22\times$, $2.78\times$ and $1.37\times$ improvement in mean error, peak error, and power consumption respectively over the baseline *ALM*, while the area increases by $1.40\times$. 16-bit two-level *multi-ALM* can achieve up to $17.5\times$ and $2.75\times$ improvement in mean error and peak error over *ALM*. Furthermore, we evaluate the proposed *multi-ALM* design in a random 8-point MAC application and a discrete cosine transformation (DCT) application. The result shows that with the increase in the segmentation and levels of the approximate multiplier, *multi-ALM* can improve the accuracy of MAC and quality of reconstructed image upon the conventional fixed configuration approximate multipliers with small hardware overhead to introduce re-configurability.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[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] S. Yu, Y. Liu, and S. X.-D. Tan, "Cosaim: Counter-based stochastic-behaving approximate integer multiplier for deep neural networks," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 499–504, IEEE, 2021.

[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] S. Yu and S. X.-D. Tan, "PAALM: Power density aware approximate logarithmic multiplier design," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ASPDAC '23, (New York, NY, USA), p. 128–133, Association for Computing Machinery, 2023.

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

[15] 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.

[16] 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.

[17] Z. Babič, A. Avramovič, and P. Bulič, "An iterative mitchell's algorithm based multiplier," in *2008 IEEE International Symposium on Signal Processing and Information Technology*, pp. 303–308, IEEE, 2008.