PAALM: Power Density Aware Approximate Logarithmic Multiplier Design

<u>Shuyuan Yu</u> and Sheldon Tan

VSCLAB

Department of Electrical and Computer Engineering

University of California, Riverside

UC RIVERSIDE

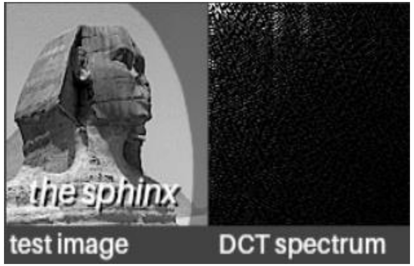
Outline

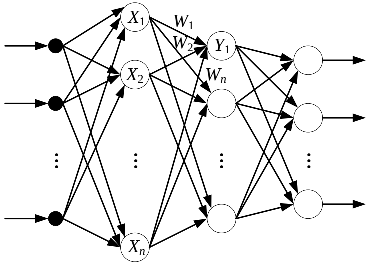
- Background
- Related Works & Motivation
- PAALM Design
- Results & Discussion
    - ➢ Hardware Performance
    - ➢ Error Evaluation
    - ➢ DCT & CNN application
- Conclusion

Background

Emerging workloads and application are error tolerant

- Image process and computer vision (DCT, edge detection, contrast stretch, etc.)
- Machine learning (DNN, CNN, Transformer, Language model, etc.)
- Applications interfacing with human being do not need exact values





Approximate computing, which allows the trade-off between area, delay, and power, is more efficient for error tolerant applications.

Background

Approximate Multipliers

- Ad-hoc based
- **Inexact adder based**
- Smaller multiplication with reduced precision
- **Log-based**

Advantages for ALM:

- Scalability (similar Rel. error for 8-bit, 16-bit, 32-bit)
- Area/Power/Energy efficient
- Acceptable accuracy: **3.76%** (Mean Rel.) & **11.11%** (Peak Rel.)

Approximate Log-based Multiplier (ALM [8])

Concept:

$A = 2^{ka} \cdot (1+x)$

$B = 2^{kb} \cdot (1+y)$

$$
\begin{aligned}
C_{Exact} &= 2^{k_a+k_b} \cdot (1+x) \cdot (1+y) \\
&= 2^{k_a+k_b} \cdot (1+x+y+\cancel{xy})
\end{aligned}
$$

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

UC RIVERSIDE

Related Work & Motivation

Problems

Approximate multipliers **reduce the area & power consumption** at the same time; however, the area usually **decreases faster** than the power, which leads to a **much higher power density** when compared with the FxP multiplier baseline.

8-bit multipliers power density of ALM and improved state of art works

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

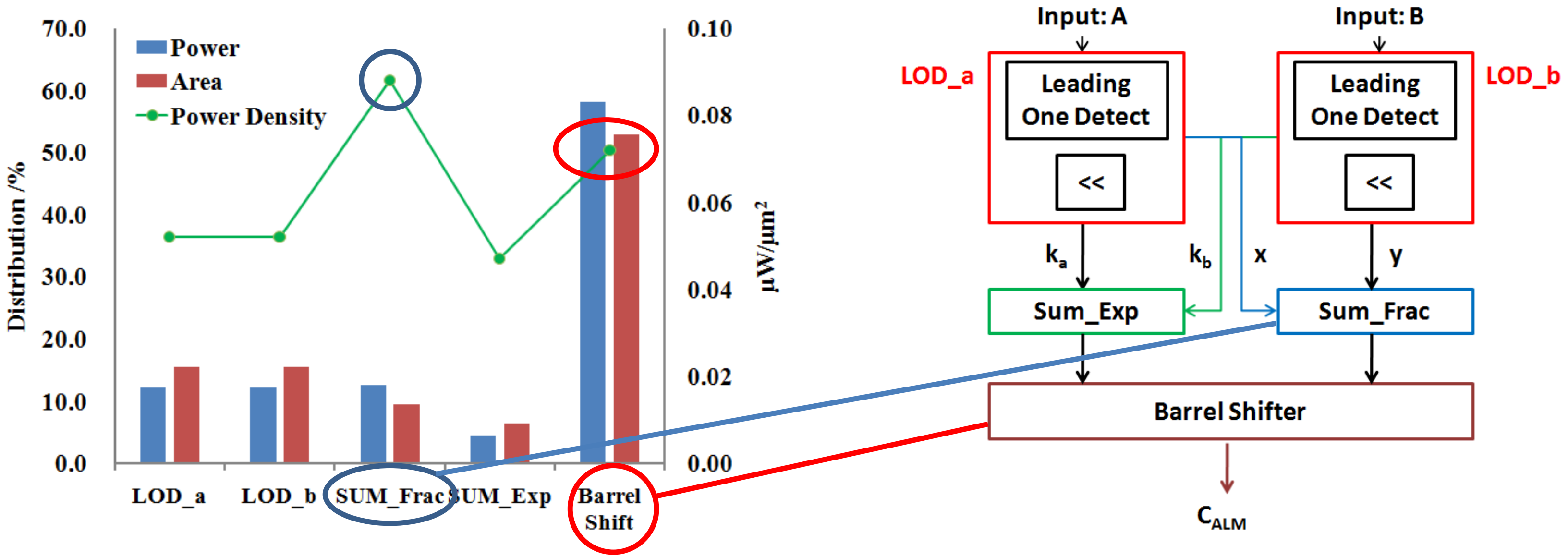
Motivation: propose approximate logarithmic multiplier design with reduced power density and improve the hardware performance at the same time.

UC RIVERSIDE

Power Dense Unit in the Conventional ALM Design

Unit power density in an ALM design

ALM architecture



Power density are different for different units in ALM!

PAALM Design

Rewrite the math expression of ALM

Math equation of ALM:

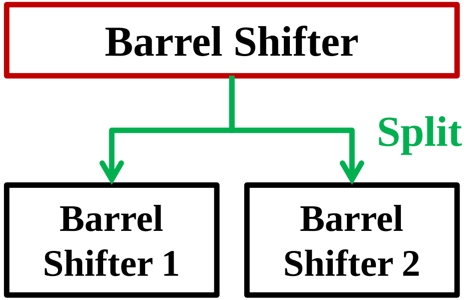
$$C_{ALM} = \begin{cases} 2^{k_a+k_b} \cdot (1+x+y), & x+y<1, \\ 2^{k_a+k_b+1} \cdot (x+y), & x+y \geq 1 \end{cases}$$

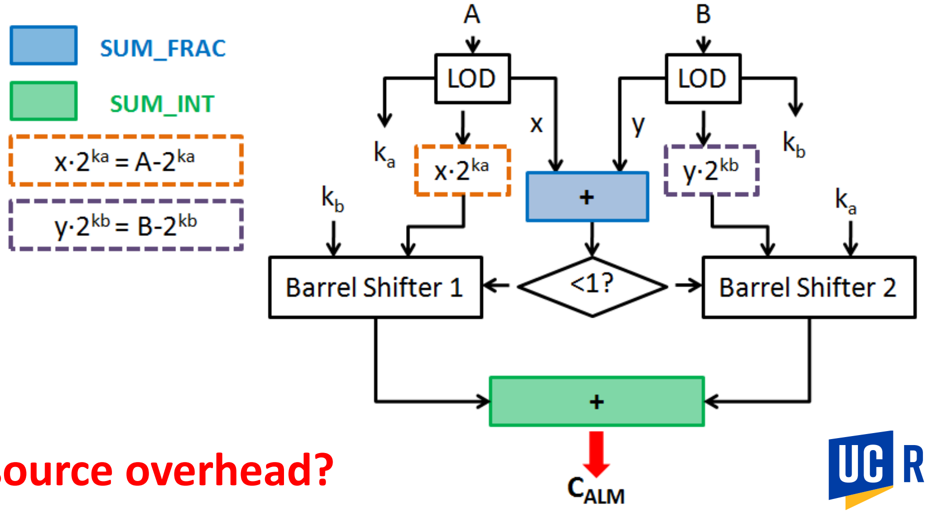
Where inputs:
$A = 2^{ka} \cdot (1+x)$
$B = 2^{kb} \cdot (1+y)$

When $x+y<1$: $2^{ka+kb} \cdot (1+x+y) = 2^{ka} \cdot (1+x) \cdot 2^{kb} + 2^{kb} \cdot y \cdot 2^{ka} = A \cdot 2^{kb} + (B-2^{kb}) \cdot 2^{ka}$

When $x+y \geq 1$: $2^{ka+kb+1} \cdot (x+y) = 2^{ka} \cdot (1+x) \cdot 2^{kb+1} + 2^{kb} \cdot y \cdot 2^{ka+1} = (A-2^{ka}) \cdot 2^{kb+1} + (B-2^{kb}) \cdot 2^{ka+1}$

Power density aware ALM architecture



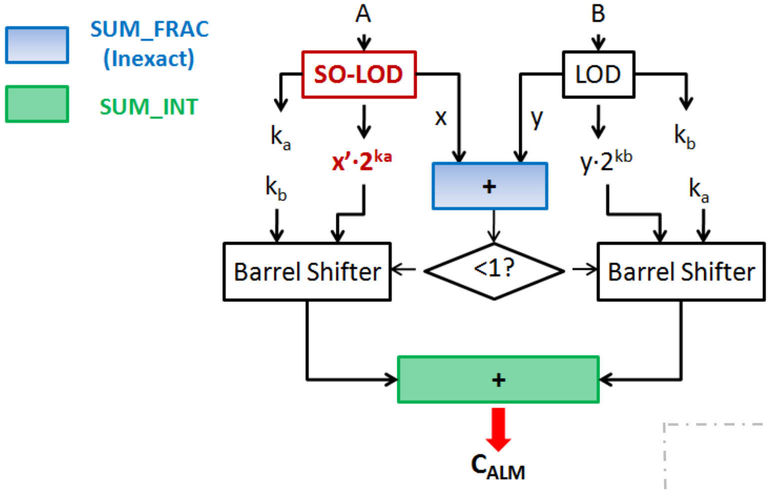


Resource overhead?

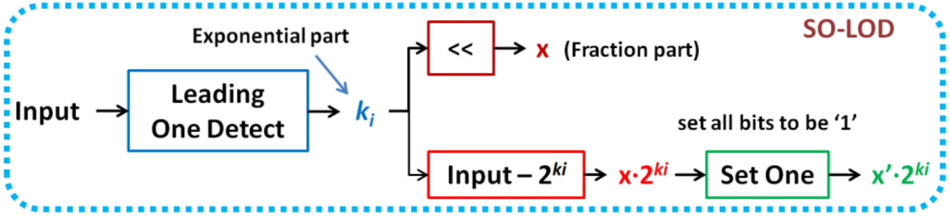
UC RIVERSIDE

Introducing Inexact Summation & Error Compensation Scheme

Improved PAALM architecture



Set one leading one detector (SO-LOD)



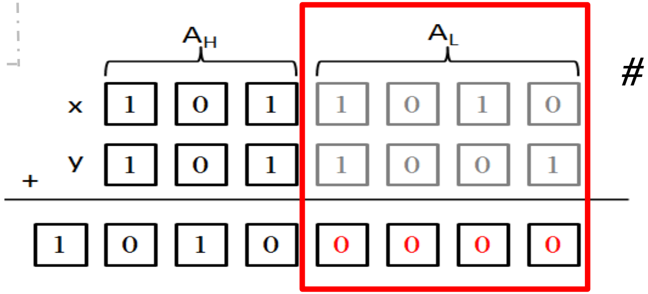
Example: Input: 1010 0110 (8-bit)

$k_i = 7$, Input – 2^{ki} = 1010 0110 – 1000 0000 = 010 0110

111 1111

Inexact summation unit:

simple truncation adder

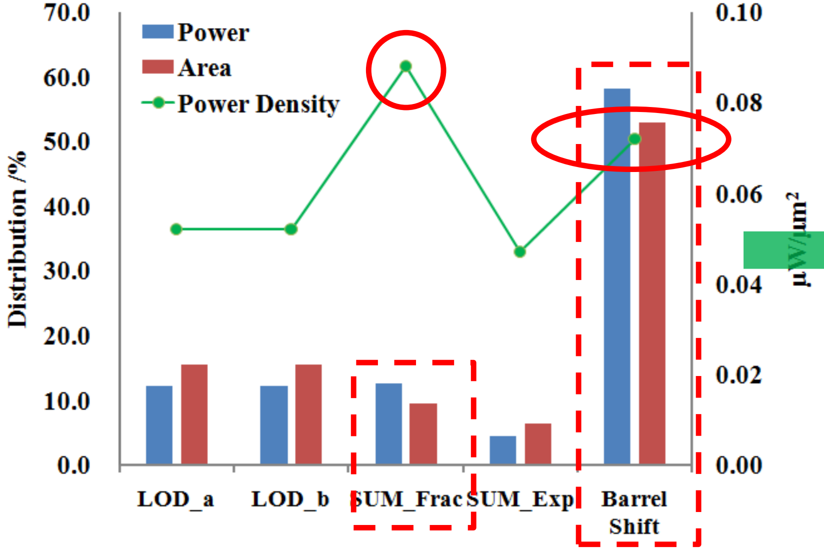


# of bits: k

UC RIVERSIDE

More Balanced Component Power Density

Unit power density in an ALM design



Unit power density in a PAALM design

The unit with the highest power density now is reduced from 88nW/μm^2 to 58nW/μm^2.

UC RIVERSIDE

Results & Discussion

Experimental setup

- Baseline: FxP
- EDK 32nm standard cell library (SNPS DC)
- Same throughput (different latency will lead to different working frequency)
- 1 million random input pairs (Matlab) for 8-bit; 10 million random input pairs for 16-bit

8-bit Hardware & Error Metrics

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

31.6%

34.5%

Results & Discussion

Experimental setup

- Baseline: FxP
- EDK 32nm standard cell library (SNPS DC)
- Same throughput (different latency will lead to different working frequency)
- 1 million random input pairs (Matlab) for 8-bit; 10 million random input pairs for 16-bit

16-bit Hardware & Error Metrics

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

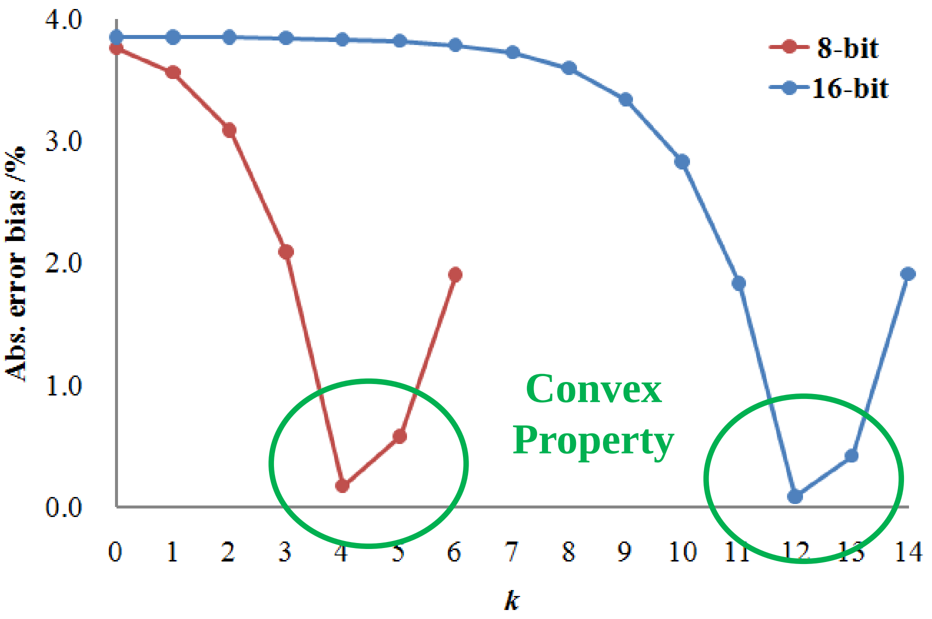
70.8%

14.5%

5.7%

Results & Discussion

Error Bias Improvement



A "rebound" in both 8-bit and 16-bit PAALM error bias curves.

The "set one" error compensation tends to introduce larger and larger positive error to balance the negative error due to the log-based approximation.

Application: Convolutional Neural Network

Dataset: CIFAR10

- 2 CONV (convolution) layers
+ 3 FC (fully connected) layers

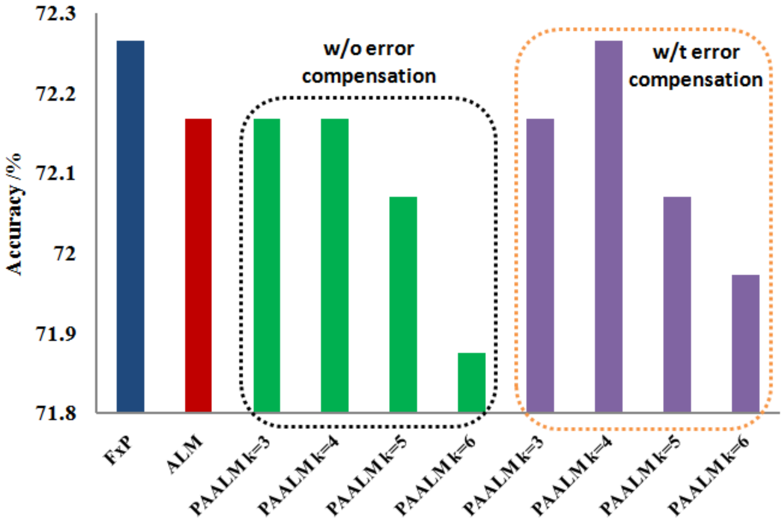
- Trained with 3000 steps using double-precision floating point

- Batch size: 128

- Inference: 8-bit

- Accuracy verify: 1024 images

CIFAR10 dataset inference accuracy based on different approximate multipliers



Application: Discrete Cosine Transform

Image quality: PSNR (dB)

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

8.6dB

Conclusion

- Propose a power density aware approximate logarithmic multiplier design (PAALM) by redesigning the components with high power density in the conventional ALM design.

- 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 CNN workload 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 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.

Questions???