Multiple-base Logarithmic Quantization and Application in Reduced Precision AI Computations

Vassil Dimitrov[1, 2], Richard Ford[1], Laurent Imbert[1, 3], Arjuna Madanayake[1], Nilan Udayanga[1], Will Wray[1]

[1] Lemurian Labs, Oakville, Canada
[2] University of Calgary, Calgary, Canada
[3] LIRMM, CNRS, University of Montpellier, Montpellier, France
vassil@lemurianlabs.com; richard@lemurianlabs.com;
laurent@lemurianlabs.com; arjuna@lemurianlabs.com;
nilan@lemurianlabs.com; will@lemurianlabs.com;

Abstract. The power of logarithmic quantizations and computations has been recognized as a useful tool in optimizing the performance of large ML models. There are plenty of applications of ML techniques in digital preservation. The accuracy of computations may play a crucial role in the corresponding algorithms. In this article, we provide results that demonstrate significantly better quantization signal-to-noise ratio performance thanks to multiple-base logarithmic number systems (MDLNS) in comparison with the floating point quantizations that use the same number of bits. On a hardware level, we present details about our Xilinx VCU-128 FPGA design for dot product and matrix vector computations. The MDLNS matrix-vector design significantly outperforms equivalent fixed-point binary designs in terms of area (A) and time (T) complexity and power consumption as evidenced by a $4\times$ scaling of AT^2 metric for VLSI performance, and 57% increase in computational throughput per watt compared to fixed-point arithmetic.

1 Introduction

Over the last few years, it has been recognized that commonly used numerical data formats, like fixed-point or floating-point representations, do not offer the best balance of efficiency and accuracy in large machine learning (ML) models (LeCun, 2019). Logarithmic number systems (LNS) and residue number systems (RNS) have become a popular tool for implementing large scale matrix-vector products with limited but sufficient precision. The work by Miyashita et al. (2016) should be noted as a critical milestone. The authors pointed out that one needs fairly low quantization resolution (e.g., 5 bits or less) to achieve good accuracy in some commonly used ML models. They show that logarithmic quantizations seem more suitable than fixed or floating-point quantizations in terms of matching the histograms of the weights and activations distributions. In addition, they claim that the use of LNS with base $\sqrt{2}$ is superior for those limited precision quantization and computation applications in comparison to LNS with base

Digital Presentation and Preservation of Cultural and Scientific Heritage. Conference Proceedings. Vol. 14, Sofia, Bulgaria: Institute of Mathematics and Informatics – BAS, 2024. ISSN: 1314-4006, eISSN: 2535-0366

2. Vogel et al. (2018) extended those results to bases like $2^{1/4}$ and even $2^{1/8}$. Arnold et al. (2019) and Alam et al. (2021) investigated in more detail LNS base selection. It is important to point out that perhaps the earliest research on the base selection of the LNS in digital signal processing (DSP) was done in the early 1980s by Sicuranza (1982).

Several publications generalized the concept of classical logarithmic representations. Since some of these representations use the same name but have completely different meanings it is appropriate to give brief descriptions. The term *multi-dimensional logarithmic number system* (MDLNS, see Definition 1) was introduced at ARITH-2001 by Dimitrov et al. (2001), with various applications in DSP published over the years. *Dual-logarithmic representation*, introduced by J. Johnson (META) in his ARITH-2020 paper (Johnson, 2020), uses representations of real numbers of the form $2^a e^b$ (with e the base of the natural logarithm, a an integer, and b a fixed-point real number). *Multi-base LNS* (Niu et al., 2024) is a hybrid version of the LNS considered by Miyashita et al. (2016), where base $\sqrt{2}$ is used whenever it better matches the histogram of the corresponding weights and activations in a particular layer of the DNN; otherwise, classical LNS with base 2 is used. In Zhao et al. (2022) a team from NVIDIA uses the same term, *Multi-base LNS*, for a system using separate LNSs for the integer and fractional parts of the numbers involved in the computations. It can be viewed as a MDLNS with bases $(2,1/2)$.

The aim of this paper is to introduce promising new results specifically tailored to machine learning applications with limited precision. We show that optimized MDLNS could lead to quantizations that outperform the best floating-point counterparts. We present FPGA simulations for matrix-vector products that achieve an almost $4\times$ improvement of the area-time complexity over the equivalent binary (fixed-point) designs and a 57% improvement in throughput per watt. We also outline research directions to explore towards further improvements in accuracy, speed and power consumption.

2 MDLNS Quantization

Quantization is defined as the process of mapping an infinite set of continuous values to a finite set of discrete values. A famous example of quantization is the mapping of the infinite set $\mathbb{R}$ to the set of floating points values for arithmetic operations involving real numbers. As such, quantization introduces limits on the precision and range of a value, as well as various sources of errors (e.g. rounding errors, underflow or overflow, computational noise, etc.). In the very active area of deep neural networks, quantization has become a very popular method for accelerating inference as well as for reducing memory and power consumption on resource-constrained devices (Gholami et al., 2022). In this context, post training quantization consists in reducing the precision at which neural network weights and activations are stored and manipulated. Several approaches based on fixed-point (integers), low-precision floating-point (FP), and variants of Logarithmic Number Systems (LNS) have been investigated.

Definition 1 (MDLNS). *Let* $R = (\beta_1, \dots, \beta_k) \in (\mathbb{R}_{>0})^k$ *a finite sequence of multiplicatively independent*[1] *positive real numbers,* $W = (w_1, \dots, w_k) \in \mathbb{N}^k$, *and* $B = (b_1, \dots, b_k) \in \mathbb{Z}^k$. *Let also* $n = 1 + \sum_{i=1}^{k} w_i$. *Then, MDLNS* $n(R, W, B)$ *denotes the finite set of real numbers of size* 2^n *given by (1). When there is no ambiguity, we omit* R, W *and* B *and simply write*

$$\text{MDLNS } n = \left\{ \pm \prod_{i=1}^{k} \beta_i^{e_i}; 0 \le e_i + b_i < 2^{w_i} \right\} \tag{1}$$

The elements of R *are called the MDLNS bases. The exponents* $e_i \in \mathbb{Z}$ *in (1) have bitlength* w_i *respectively and are biased with bias* b_i, *i.e. the unsigned binary encoded value* $\hat{e}_i$ *corresponds to the integer* $e_i = \hat{e}_i - b_i$. *A bias equal to* $2^{w_i - 1}$ *corresponds to the two's complement notation.*

An important metric that has been used to measure the numerical fidelity of a quantization scheme is the Quantization Signal to Noise Ratio (QSNR) introduced in Darvish Rouhani et al. (2023). QSNR is defined as the ratio of the power of the non-quantized signal (i.e., the original vector $X = (x_1, x_2, \dots, x_k) \in R^k$) to the power of the quantization noise expressed in decibels. It is calculated as:

$$\text{QSNR} = -10\log_{10} \frac{E(\| Q(X) - X \|^2)}{E(\| X \|^2)}$$

For example, assuming $\mathcal{N}(0,1)$ distribution of data the QSNR values for the small and widely used floating point formats FP6 (e3m2), FP8 (e4m3) and FP10 (e5m4) are 25.46 dB, 31.52 dB and 37.53 dB respectively.

A. MDLNS *bases with high QSNR*

In Figure 1, QSNR values for 6-bit, 8-bit and 10bit MDLNS formats with bases $(2, \beta)$ are plotted as a function of β. The three horizontal lines correspond to the QSNR values obtained for FP6, FP8 and FP10 formats. This plot clearly shows that finding MDLNS parameters that provide the highest QSNR, amounts to finding the maximum of a highly non-monotonic function. Yet, the interesting observation derived from Figure 1 is that there are many MDLNS parameters that yield better QSNR than floating point formats of the same size (the points above the horizontal lines).

In Table 1, we report parameters of four MDLNS6 formats whose bases contain the golden ratio $\phi = (1 + \sqrt{5})/2$ or its inverse, a famous irrational number known for its many interesting properties. Asymptotically, this base selection offers the most homogeneous distribution of the real numbers generated by such MDLNS. In the next section, we present implementation details and results for the core operation of matrix-vector product with bases $[2, 2^{\phi}]$.

[1] $\beta_1, \dots, \beta_k$ are multiplicatively dependent iff the equation $\prod_{i=1}^{k} \beta_i^{e_i} = 1$ has a non-trivial solution in integers.

3 Multichannel MDLNS Matrix-Vector MULTIPLIERS

Computation of matrix-vector products is a fundamental operation in ML algorithms where efficient multichannel parallel realizations of products are building blocks for deep convolutional networks (CNNs) for image classification. A multichannel matrix-vector multiplier (MVM) is the building block. A large number of MVM are summed and applied to a non-linear block (e.g., ReLu). Taking as inputs a matrix $\mathbf{A} = \left(a_{i,j}\right)$ for $1 \leq i,j \leq N$ and a vector $\mathbf{x} = (x_1, x_2, \dots, x_N)$, an MVM computes the vector $\mathbf{y} = (y_1, y_2, \dots, y_N)$ where the k-th entry is given by $y_k = a_{k,1}x_1 + a_{k,2}x_2 + \cdots + a_{k,N}x_N$. MVM thus consists of N parallel vector dot products. An M-channel MVM multiplexes M distinct matrices $A_\ell, \ell = 1,2, \dots, M$ having elements $a_{i,j,\ell}$ over M consecutive input vectors denoted $\mathbf{x}_\ell = \left(x_{1,\ell}, x_{2,\ell}, \dots, x_{N,\ell}\right)$ to produce the output vector $\mathbf{y} = \sum_{\ell=1}^{M} \mathbf{A}_\ell \mathbf{x}_\ell$.

A. Overview of 8×8 *MVM Tile*

Our design uses a MDLNS7 with bases $(2, \beta)$, with $\beta = 3.06 \approx 2^{\phi}$ with $\phi = (1 + \sqrt{5})/2$ the golden ratio. In this MDLNS, numbers are represented as $(-1)^s 2^a \beta^b$ with $s \in \{0,1\}, |a| = 4$ bits and $|b| = 2$ bits. Each MDLNS partial-product inside a vector-vector multiplication is easily computed by adding exponents; the partial result $2^{(a+c)}\beta^{(b+d)}$ is then approximated back to two's complement format at 16 -bits of precision before accumulation. The number of channels is chosen as a power of 2 to efficiently map to circular buffers. The output rate of the multichannel MVM is F/M vectors, where F is the master system clock rate, allowing up to M different streams to be applied to a particular multi-channel MVM in the next stage of the computation.

B. Microarchitecture of MVM Tile

The MVM design supports up to 64 channels. It is prototyped on Xilinx VCU-128 FPGA. The number of channels is selected as 48 to satisfy timing requirements of an Ethernet adapter on the FPGA. In an ASIC design the number of channels can be any number up to 64 for $M = 64$. The MVM design has four main sub-systems (A-D) described below:

A: Fixed-point to MDLNS Conversion: an 8-bit two's complement fixed-point to MDLNS mapper. This unit is based on look up tables (LUTs) realized as SRAM; it produces two output channels corresponding to the 4-bit and 2-bit exponents of the MDLNS that update at the master clock rate F. Including sign bit, the total is 7 bits.

B: Commutating Coefficient Memory: a circular buffer clocked at the system rate F. Weights repeat every M cycles. Given $M = 64$, the MVM supports up to 64 different 8×8 coefficient matrices stored within the computational tile using N^2 number of B blocks. The adoption of larger values of M increases memory usage but reduces power consumption incurred in repeated movement of coefficient values from far memory located off-chip in external RAMs. Modern AI algorithms may require thousands of channels, which implies MVM unit has to support efficient memory architectures on ASIC.

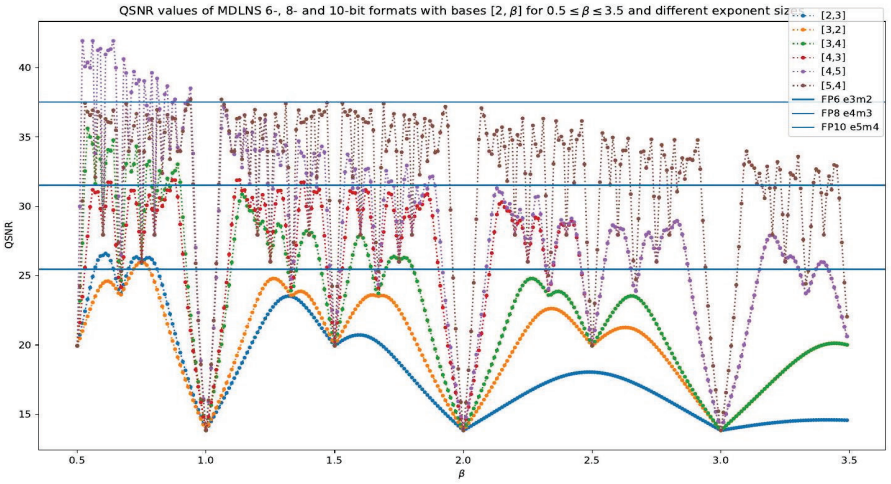

Fig. 1. QSNR values for MDLNS6, MDLNS8 and MDLNS10 formats with bases $[2, \beta]$ for $\beta \in [0.5,3.5]$ and various exponent sizes.

Table 1. Parameters for various MDLNS bases that include the golden ratio.

Bases	$[2, 2^{\phi}]$	$[2, 2^{\phi}]$	$[2, 2^{\phi-1}]$	$[2, 2^{\phi-1}]$	$[2, 2^{2-\phi}]$	$[2, 2^{2-\phi}]$
Exponent sizes	[2,3]	[3,2]	[2,3]	[3,2]	[2,3]	[3,2]
Exponent biases	[2,4]	[4,2]	[2,4]	[4,2]	[2,4]	[4,2]
Min. pos. value	0.003	0.007	0.045	0.027	0.087	0.037
Max. pos. value	57.844	24.557	7.231	12.278	4.426	10.425
QSNR	20.672	23.407	26.519	24.611	27.234	24.646

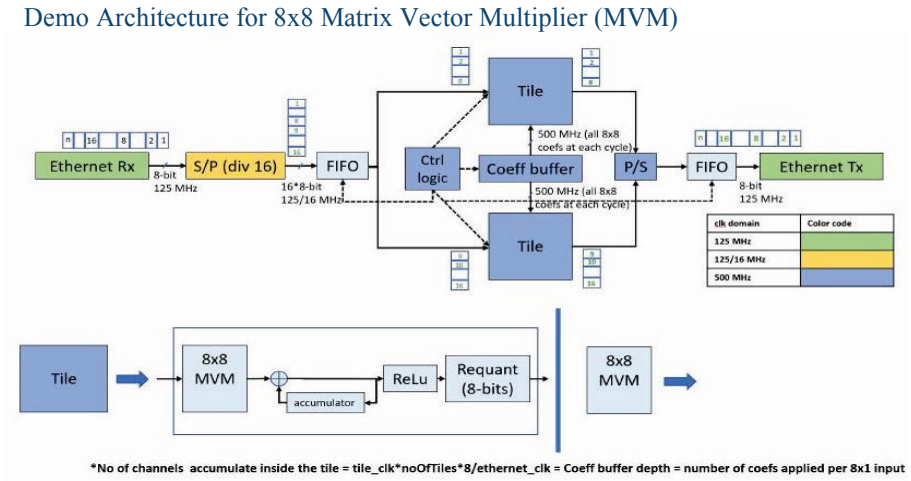

Fig. 2. VCU-128 based FPGA realization of two parallel datapaths (8-bit) having 8 × 8 MVM units. Coefficients are of 7 -bit precision for both fixed-point and MDLNS. High-speed connectivity to Linux host achieved via Gigabit ethernet port created within the FPGA and running UDP protocol (packet size 512 bits/frame).

C: MDLNS Based Multiplier: Let us denote the data inputs at row k as $a_k(n), b_k(n)$ and coefficients as $c_{k,\ell}(n), d_{k,\ell}(n)$ for $\ell = 1,2,\dots,N$ columns, and $k = 1,2,\dots,N$ rows. Since MDLNS multiplication is achieved by exponent additions, the N^2 different multiplicative components of a MVM operate at clock cycle n computed as $2N^2$ unsigned additions, of the form $Pa_{k,\ell}(n) = a_k(n) + c_{k,\ell}(n)$ and $Pb_{k,\ell}(n) = b_k(n) + d_{k,\ell}(n)$ where $k = 1,2,\dots,8$. These partial components are in MDLNS format and cannot be directly added to produce the row-output of the MVM. We convert $Pa_{k,\ell}(n)$ and $Pb_{k,\ell}(n)$ corresponding to matrix elements at (k,ℓ) to two's complement format before summation.

D: MDLNS to Fixed-point Converter: Converts a MDLNS quantity to fixed-point format. Each **D** block produces a fixed point output $Pc_{k,\ell}(n)$ that is applied to a $\log_2 N$ input adder tree that in turn produces each row of the MVM for a given channel at clock period n.

C. Implementation Results on Xilinx VCU-128

The VCU-128 was used for evaluating the resource and power consumption, clock, and throughput of a 7-bit MDLNS realization. For our simulations, FPGA was operated with MDLNS implementation of two parallel 8 × 8 MVM units, with data connectivity to a Linux host using 1 Gbps Ethernet via user datagram protocol (UDP). A comparison with two's complement MVM implementation of similar accuracy is provided in Table 2. The area is given in number of CLBs. No DSP blocks were used. As can be seen, MDLNS enabled 57% increase in throughput using much lower FPGA resources. VLSI circuits are often compared using AT^2 metric, where A is area and T is the critical path delay. Using this AT^2 metric, our fixedpoint and MDLNS designs lead to AT^2 values

of 0.37 and 0.09 respectively, that is a 4 × improvement by adopting MDLNS over fixed-point.

Table 2. Comparisons between fixed-point and MDLNS implementation results.

	Fixed-point	MDLNS
Configurable logic blocks	35659CLBs	28813CLBs
Static power	3.93 W	3.22 W
Dynamic power	3.2 W	4.41 W
Maximum clock rate	312MHz	555MHz
Throughput	47.4Gops/W	74.4Gops/W

4 Conclusions

The article showcases the applicability of MDLNS representations. One advantage of MDLNS is the freedom to choose the bases in order to optimize the quality of the data quantizations in a way that is unachievable with other number systems like fixed-point, floating-point or classical logarithmic representations. Further improvements can be obtained, for example by extending the number of bases in the MDLNS. On a hardware level, the FPGA results have to be extended to a VLSI level. Some of the main findings from this paper and additional improvements are subject to IP protection. We hope that these results will encourage the computer arithmetic community to explore the great potential of multidimensional logarithmic representation in other application domains.

References

Alam, S. A., Garland, J., & Gregg, D. (2021). Low-precision logarithmic number systems: beyond base-2. *ACM Transactions on Architecture and Code Optimization (TACO)*, *18*(4), Article 47, 1-25. https://doi.org/10.1145/3461699

Arnold, M., Chester, E., Cowles, J., & Johnson, C. (2019, October). Optimizing Mitchell's Method for Approximate Logarithmic Addition via Base Selection with Application to Back-Propagation. In *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC), Helsinki, Finland* (pp. 1-6). IEEE. https://doi.org/10.1109/NORCHIP.2019.8906904

Darvish Rouhani, B., Zhao, R., Elango, V., Shafipour, R., Hall, M., Mesmakhosroshahi, M., More A, Melnick L, Golub M, Varatkar G, Shao L., Kolhe, G., Melts, D., Klar, J., L'Heureux, R., Perry, M., Burger, D., Chung, E., Deng, Z., Naghshineh, S., Park, J., & Naumov, M. (2023, June). With shared microexponents, a little shifting goes a

long way. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23),* (Article 83, pp. 1-13). Association for Computing Machinery. https://doi.org/10.1145/3579371.3589351

Dimitrov, V. S., Eskritt, J., Imbert, L., Jullien, G. A., & Miller, W. C. (2001, June). The use of the multi-dimensional logarithmic number system in DSP applications. In *Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001* (pp. 247-254). IEEE. https://doi.org/10.1109/ARITH.2001.930126

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2022). A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision* (1st ed.) (pp. 291-326). Chapman and Hall/CRC.

Johnson, J. (2020, June). Efficient, arbitrarily high precision hardware logarithmic arithmetic for linear algebra. In *2020 IEEE 27th Symposium on Computer Arithmetic (ARITH)* (pp. 25-32). IEEE. https://doi.org/10.1109/ARITH48897.2020.00013

LeCun, Y. (2019). 1.1 Deep Learning Hardware: Past, Present, and Future. In *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*, San Francisco, CA, USA (pp. 12-19). IEEE. https://doi.org/10.1109/ISSCC.2019.8662396.

Miyashita, D., Lee, E. H., & Murmann, B. (2016). *Convolutional neural networks using logarithmic data representation*, arXiv:1603.01025. https://doi.org/10.48550/arXiv.1603.01025

Niu, Z., Zhang, T., Jiang, H., Cockburn, B. F., Liu, L., & Han, J. (2024, January). Hardware-Efficient Logarithmic Floating-Point Multipliers for Error-Tolerant Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *71*(1), 209-222. https://doi.org/10.1109/TCSI.2023.3326329

Sicuranza, G. (1982, May). On the accuracy of 2-D digital filter realizations using logarithmic number systems. In *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing* (Vol. 7, pp. 48-51). IEEE. https://doi.org/10.1109/ICASSP.1982.1171386

Vogel, S., Liang, M., Guntoro, A., Stechele, W., & Ascheid. G. (2018, November). Efficient Hardware Acceleration of CNNs using Logarithmic Data Representation with Arbitrary log-base. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (pp. 1–8). https://doi.org/10.1145/3240765.3240803

Zhao, J., Dai, S., Venkatesan, R., Zimmer, B., Ali, M., Liu, M. Y., Khailany, B., Dally, W.J. & Anandkumar, A. (2022, December). Lns-madam: Low-precision training in logarithmic number system using multiplicative weight update. *IEEE Transactions on Computers*, *71*(12), 3179-3190. https://doi.org/10.1109/TC.2022.3202747

Received: March 15, 2024
Reviewed: April 05, 2024
Finally Accepted: May 10, 2024