Logarithm-Approximate Floating-Point Multiplier for Hardware-efficient Inference in Probabilistic Circuits

Lingyun Yao[1] **Martin Trapp**[2] **Karthekeyan Periasamy**[1] **Jelin Leslin**[1] **Gaurav Singh**[1] **Martin Andraud**[1]

[1]Electrical Engineering Dept., Aalto University, Espoo, Finland
[2]Computer Science Dept., Aalto University, Espoo, Finland

Abstract

Machine learning models are increasingly being deployed onto edge devices, for example, for smart sensing, reinforcing the need for *reliable* and *efficient* modeling families that can perform a variety of tasks in an uncertain world (*e.g.*, classification, outlier detection) without re-deploying the model. Probabilistic circuits (PCs) offer a promising avenue for such scenarios as they support efficient and exact computation of various probabilistic inference tasks by design, in addition to having a sparse structure. A critical challenge towards hardware acceleration of PCs on edge devices is the high computational cost associated with multiplications in the model. In this work, we propose the first approximate computing framework for energy-efficient PC computation. For this, we leverage addition-as-int approximate multipliers, which are significantly more energy-efficient than regular floating-point multipliers, while preserving computation accuracy. We analyze the expected approximation error and show through hardware simulation results that our approach leads to a significant reduction in energy consumption with low approximation error and provides a remedy for hardware acceleration of general-purpose probabilistic models.

1 INTRODUCTION

The development of smart sensing and Internet-of-Things applications based on embedded artificial intelligence (AI), such as smartphones, wearables, or other sensor networks, is pushing the computation of machine learning methods directly onto edge devices. Recent innovations (*e.g.*, [12, 26, 17]) have pushed up the computational efficiency of deep feedforward neural networks (NNs) and improved the energy efficiency of dedicated AI processors by $10\times$ – $100\times$ compared to Graphical Processing Units [17]. However, NNs that have been adopted into real-world use often raise concerns related to their reliability, fairness, and interpretability [9, 7] alongside their high inference costs [27, 23].

Consequently, to be suitable for the challenges associated with edge AI, there is an urgent need to develop *effective* hardware acceleration of machine learning models that are *probabilistic*, *i.e.*, they enable reasoning in an uncertain world [6], and *tractable*, *i.e.*, they can reliably answer many probabilistic queries without re-deployment. Recent work on tractable probabilistic models, specifically on probabilistic circuits (PCs) [2], poses a promising avenue as these models (i) exhibit high expressive efficiency (representational power), (ii) enable reliable [25, 13] and fair [1] reasoning, and (iii) allow many probabilistic queries to be computed tractably by design. Yet, while pioneering works have explored acceleration of PCs on Field Programmable Gate Arrays (FPGAs) [3, 21, 22] and Application-Specific Integrated Circuits (ASICs) [18, 19], the hardware acceleration of PCs poses many open challenges. In particular, their *irregularity* (*i.e.*, PCs are sparsely connected making parallelism more challenging [20]) and *high computation resolution* (*i.e.*, probabilistic inference with PCs typically requires 30 – 40 floating-point bits [22, 20] as arithmetics are performed on probabilities) hinders their deployment on edge devices where efficiency and reduced resolution are key due to the limited energy resources.

In this work, we propose to approximate floating-point multipliers through Addition-as-Int [10], suggesting high potential gains in computational efficiency (Addition-as-Int can reduce the hardware cost of multiplication by a factor of up to $112\times$) with little impact on the accuracy of the computations. In addition, we carry out a theoretical analysis of the expected error and show that our approach can result in accurate computations for maximum-a-posteriori (MAP) and marginal queries and enables to concisely trade-off accuracy and computational efficiency.

Accepted for the 6th Workshop on Tractable Probabilistic Modeling at UAI (TPM 2023).

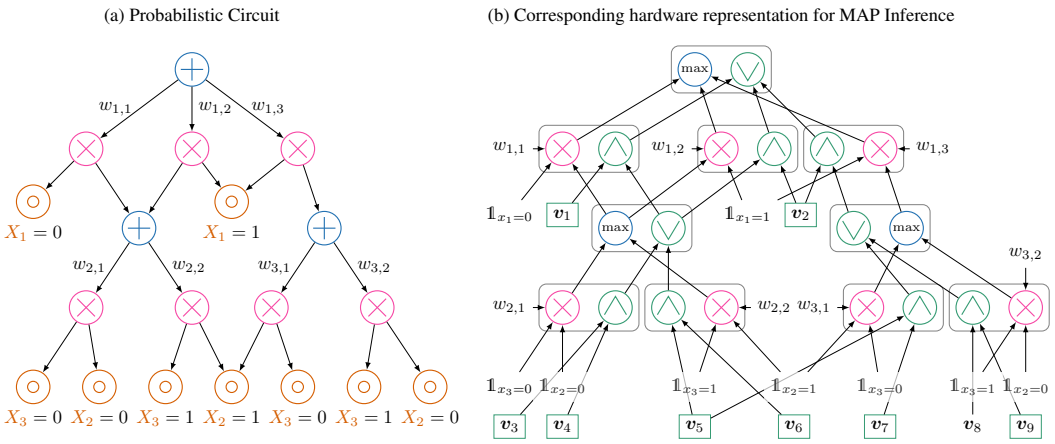

Figure 1: Illustration of a PC (a) over discrete RVs (X_1, X_2, X_3) and the corresponding hardware realization of MAP inference (b). For this, sum nodes are replaced by max operators, and an additional propagation path for information bits is added to back-track the most probable path (MAP result)

2 BACKGROUND: PROBABILISTIC CIRCUITS

Probabilistic circuits (PCs) have recently been introduced as an umbrella to unify a variety of existing tractable probabilistic models (*e.g.*, [4, 14, 15, 8]). They represent the (possibly unnormalized) distribution function (density or mass function) of a multivariate probability distribution over random variables (RVs) $\boldsymbol{X} = \{X_i\}_{i=1}^{d}$ through a directed acyclic graph $\mathcal{G}$. The computational graph ($\mathcal{G}$) constitutes weighted sums $\mathsf{S}(\boldsymbol{x}) = \sum_{\mathsf{C}\in\mathrm{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{C}}\mathsf{C}(\boldsymbol{x})$ with $\sum_{\mathsf{C}\in\mathrm{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{C}} = 1$, products $\mathsf{P}(\boldsymbol{x}) = \prod_{\mathsf{C}\in\mathrm{ch}(\mathsf{S})} \mathsf{C}(\boldsymbol{x})$, and leaf nodes associated with parametric functions, typically assumed to be density/mass functions of univariate probability distributions $\mathsf{L}(\boldsymbol{x}) = p(\boldsymbol{x} \mid \theta_{\mathsf{L}})$. We use $\mathrm{ch}(\mathsf{N})$ to denote the set of children of a node (N) and θ denotes parameters of the parametric leaves. In addition, each node $\mathsf{N} \in \mathcal{G}$ is associated with a scope $\psi(\mathsf{N}) \subseteq \boldsymbol{X}$ provided by a scope function $\psi\colon \mathbf{N} \to \mathcal{P}(\boldsymbol{X})$ [24], where $\mathcal{P}(\boldsymbol{X})$ denotes the power set of $\boldsymbol{X}$, specifying the set of RVs the node represents a joint distribution over. Fig. 1(a) illustrates a PC over three discrete RVs using indicator functions at the leaves, where we use $\oplus$ to illustrate sum nodes and $\otimes$ for product nodes. Fig. 1(b) illustrates our proposed hardware realization of MAP inference for a PC. A particularly relevant class of PCs are those that are *smooth* and *decomposable*, as both properties are requirements for many probabilistic queries to be computable exactly and in time linear in the number of nodes of $\mathcal{G}$. Henceforth, we will briefly review smoothness and decomposability.

Definition 2.1 (Smooth & Decomposability)**.** *A sum node* S *is* ***smooth*** *if all children have the same scope, i.e.,* $\psi(\mathsf{C}) = \psi(\mathsf{C}'), \forall \mathsf{C}, \mathsf{C}' \in \mathrm{ch}(\mathsf{S})$. *Further, a product node* P *is* ***decomposable*** *if all children have pairwise disjoint scopes, i.e.,* $\psi(\mathsf{C}) \cap \psi(\mathsf{C}') = \emptyset, \forall \mathsf{C}, \mathsf{C}' \in \mathrm{ch}(\mathsf{P})$. *A PC is smooth if all sum nodes are smooth and decomposable if all product nodes are decomposable.*

Definition 2.2 (Determinism)**.** *A sum node* S *is* ***deterministic*** *if for every complete evidence* $\boldsymbol{x}$ *at most one child has a positive value. Consequently, a PC is deterministic if all sum nodes are deterministic.*

We refer the reader to [2] for further details on the structural properties of PCs.

3 APPROXIMATE COMPUTING FOR PROBABILISTIC CIRCUITS

Assuming positive numbers in floating-point representation, two operands x and y can be written as $x = 2^{E_x}(1 + M_x)$ and $y = 2^{E_y}(1 + M_y)$. Note that we can omit the sign bit and only have to consider their exponent (E) and mantissa (M) values. Therefore, the exact product $x \times y$ is given as:

$$x \times y = 2^{E_x+E_y}(1 + M_x)(1 + M_y) \tag{1}$$

This product can be conveniently expressed in log-space, *i.e.*,

$$\log_2(x \times y) = E_x + E_y + \log_2(1 + M_x) + \log_2(1 + M_y), \tag{2}$$

A popular approximate solution is based on Mitchell's method [10]. To approximate the logarithm, Mitchell's method uses $\log_2(1 + F) \approx F$, which is the first-order Taylor series expansion of $\log_2(1 + F)$. Using this approximation, Eq. (2) becomes:

$$\log_2(x \times y) \approx E_x + E_y + M_x + M_y. \tag{3}$$

Previous work pointed out that adding two IEEE 754 floating-point numbers with an integer addition instruction

results in Mitchell's approximate multiplication and called as Addition-As-Int (AAI) [11]. By doing so, we can directly obtain an approximation from Eq. (2) to Eq. (3). Denoting $\widetilde{\times}$ as the approximate multiplication, we obtain:

$$x \,\widetilde{\times}\, y = \mathrm{FLOAT}(\mathrm{INT}(x) + \mathrm{INT}(y)) \tag{4}$$

Where $\mathrm{INT}(\cdot)$ interprets the binary string of the IEEE 754 floating-point representations as integer strings and $\mathrm{FLOAT}(\cdot)$ interprets the resulting integer string back to the IEEE 754 floating-point representation. Therefore, performing AAI in hardware only requires integer addition operators.

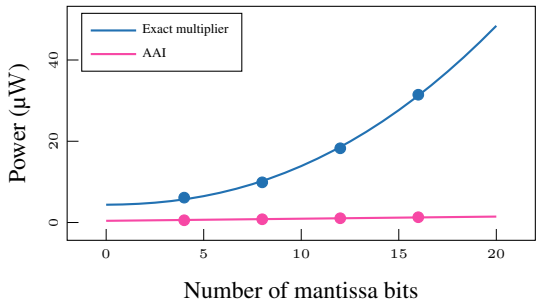

Figure 2: Power cost of multipliers on 65nm CMOS using 8 exponent bits.

4 EXPERIMENTS

We evaluated our approach on four benchmark data sets: NLTCS, Jetser, DNA, and Book, which are a subset of frequently used data sets in the community (*e.g.*, [16, 5, 24]). We generated PC structures and parameters using LearnSPN [5], a popular method for structure learning, resulting in smooth and decomposable PCs. All evaluations are performed on the test set.

4.1 POWER CONSUMPTION COMPARISON BETWEEN EXACT MULTIPLICATIONS AND AAI

Floating-point and AAI multipliers have been designed and simulated for various resolutions in a 65nm CMOS technology, and models have been fitted to the simulation results. Fig. 2 shows the resulting model for 8 exponent bits and varying number of mantissa bits. We see that the hardware cost is dominated by mantissa processing, and the hardware complexity grows significantly with the number of mantissa bits. As AAI uses much simpler addition hardware, the complexity and power grow linearly with the number of bits.

4.2 ENERGY SAVING WITH DIFFERENT NUMBER OF BITS

We replaced all multipliers with AAI to assess the error and the power savings for MAR and MAP queries under varying resolutions. For MAR queries, we computed the squared error according to a software baseline (64-bits), *i.e.*, $\sum_{\boldsymbol{x}}(p(\boldsymbol{x})-q(\boldsymbol{x}))^2$ where $q(\cdot)$ denotes the model with lower resolution multipliers and $p(\cdot)$ the PC in software. In addition, we calculated the maximum and minimum obtainable errors. For MAP queries, we calculated the MAP inference accuracy over the latent variables (assuming complete evidence) regarding the baseline. We collected the optimized bits in Table 1 where the N_b represents 32 bits, N_{be} and N_{ba} are the number of bits related to the smallest error in the exact multiplier and approximate multiplier respectively.

MAR queries. With AAI, the error varies across benchmarks but generally requires higher exponent bits E , *c.f.* Fig. 3. In practice, exact multipliers produce a small error at the tested resolutions, as seen in Table 1. Indeed, E determines the minimum representable value, and M represents the quantization in every exponent range, which only depends on the representation error. Going from a 32-bit resolution to N_{be} enables saving around $2\times$ power. We find that using AAI can allow for $24\times$ to $40\times$ extra savings if the tolerated error is a few percent. The total power savings from 32-bit to the optimal AAI are between $56\times$ and $88\times$, *c.f.* Table 1 .

MAP queries. We find that the resolution of MAP computation can be drastically reduced while introducing no error since MAP stays correct as long as the argmax at sum nodes stays the same. Further, AAI multipliers can achieve higher accuracy for fewer bits, *c.f.* Fig. 4. In contrast to exact floating-point multiplication, where mantissa values are normalized (see Appendix A), and successive multiplications result in smaller mantissa values, AAI handles normalization by using a carry, hence, requiring fewer bits. Most power savings are obtained from N_b to N_{be}, *i.e.* $18.6\times$. Switching for AAI increases savings by up to $11\times$. Total power savings can be $206\times$, *c.f.* Table 1 .

5 CONCLUSION AND DISCUSSION

We introduced approximate computing in PCs to increase their energy efficiency for deployment on edge devices and provided a theoretical and empirical analysis of the introduced error. Specifically, we investigated the energy efficiency and approximation error of Addition-as-Int multipliers in PCs for different benchmarks and query types (marginals and MAP). Our results show that maximum power savings of $88\times$ and $206\times$ can be achieved for MAR and MAP queries, respectively.

Table 1: Overview of optimal configuration and performances over several data sets. N_{be} and N_{ba} correspond to the settings with the smallest error and the loss is the error relative to the max. error.

Data set	Query	Power $N_b = 32$ μW,@N_b	Exact $\otimes$ N_{be} (E,M)	Exact $\otimes$ Power μW,@N_{be}	AAI $\otimes$ N_{ba} (E,M)	AAI $\otimes$ Power μW,@N_{ba}	Loss Exact %	Loss AAI %
NLTCS	MAP	85482	5,3	4594	5,1	414	0	0
	MAR	85482	8,15	36699	8,7	1035	3e-7	0.8
Jester	MAP	660408	5,3	35492	5,1	3199	0	0
	MAR	660408	8,15	283530	11,11	11731	4e-7	5.9
DNA	MAP	674902	5,3	36271	5,1	3269	0	0
	MAR	674902	11,15	306942	11,3	7629	3e-6	3.3
Book	MAP	1272053	5,3	68364	5,1	6162	0	0
	MAR	1272053	8,15	546124	11,7	18488	7e-6	0.4

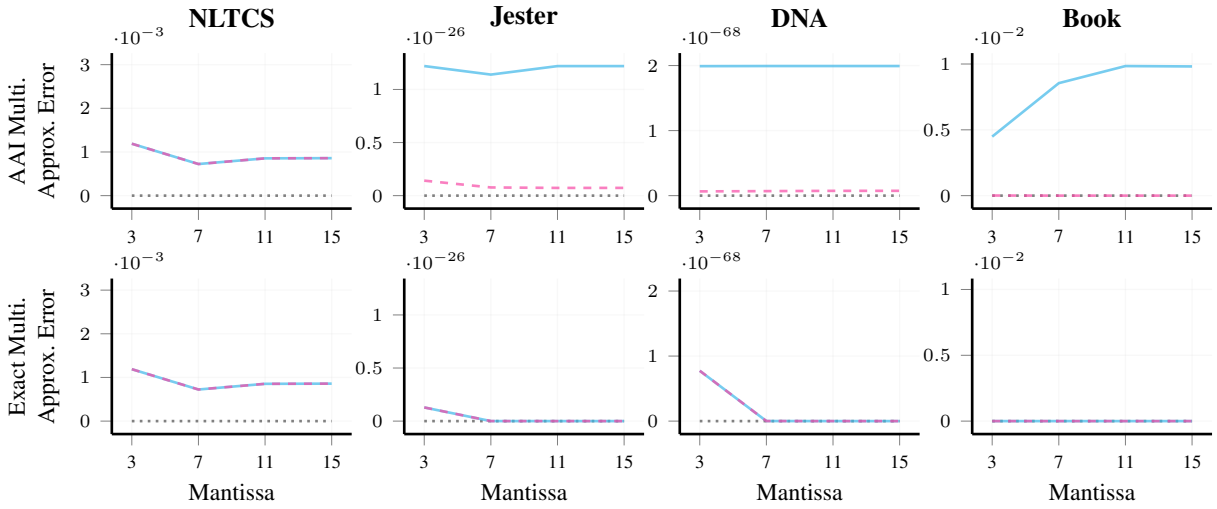

Figure 3: Results for AAI (first row) and exact (second row) multipliers using varying number of exponent (E=8, E=11) and mantissa bits. Maximum possible error (⋯) is shown for reference.

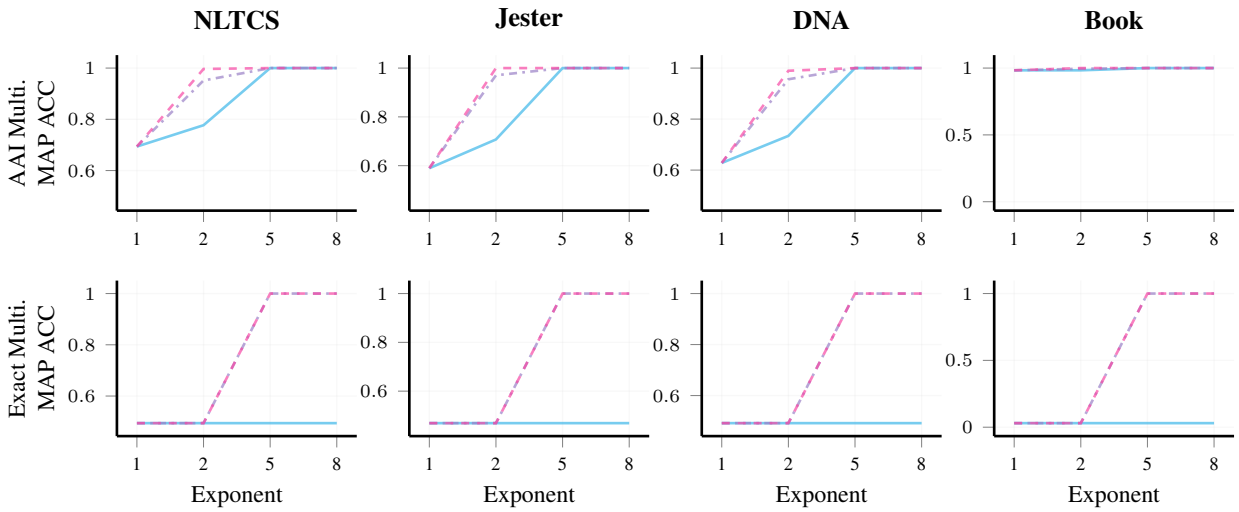

Figure 4: MAP accuracy (ACC) results for AAI (first row) and exact (second row) multipliers using varying the number of exponent and mantissa bits (m=1, m=3, m=5).

Acknowledgements

MT acknowledges funding from the Academy of Finland (grant number 347279).
MA acknowledges partial funding from the Academy of Finland through the project WHISTLE (grant number 332218). This work has also been partially funded by the European Union through the SUSTAIN project. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or EISMEA. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] YooJung Choi. *Probabilistic Reasoning for Fair and Robust Decision Making*. PhD thesis, 2022.

[2] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. oct 2020.

[3] Young-kyu Choi, Carlos Santillana, Yujia Shen, Adnan Darwiche, and Jason Cong. Fpga acceleration of probabilistic sentential decision diagrams with high-level synthesis. *ACM Trans. Reconfigurable Technol. Syst.*, sep 2022. ISSN 1936-7406. doi: 10.1145/3561514.

[4] Adnan Darwiche. A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280–305, 2003. doi: 10.1145/765568.765570.

[5] Robert Gens and Domingos Pedro. Learning the structure of sum-product networks. In *International conference on machine learning*, pages 873–880. PMLR, 2013.

[6] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, May 2015. ISSN 1476-4687. doi: 10.1038/nature14541.

[7] Ari Heljakka, Martin Trapp, Juho Kannala, and Arno Solin. Disentangling model multiplicity in deep learning. *arXiv preprint arXiv: 2206.08890*, 2023.

[8] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *14th International Conference on Principles of Knowledge Representation and Reasoning KR*. AAAI Press, 2014.

[9] Gary Marcus. The next decade in AI: Four steps towards robust artificial intelligence. *arXiv preprint arXiv: 2002.06177*, 2020.

[10] John N Mitchell. Computer multiplication and division using binary logarithms. *IRE Transactions on Electronic Computers*, (4):512–517, 1962.

[11] Tsuguo Mogami. Deep neural network training without multiplications. *arXiv preprint arXiv:2012.03458*, 2020.

[12] B. Moons and M. Verhelst. Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 4(4):475 – 486, 2014. ISSN 2156-3357. doi: 10.1109/JETCAS.2014.2361070.

[13] Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Xiaoting Shao, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In Amir Globerson and Ricardo Silva, editors, *35th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 115 of *Proceedings of Machine Learning Research*, pages 334–344. AUAI Press, 2019.

[14] Hoifung Poon and Pedro M. Domingos. Sum-product networks: A new deep architecture. In Fábio Gagliardi Cozman and Avi Pfeffer, editors, *27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 337–346. AUAI Press, 2011.

[15] Tahrima Rahman, Prasanna V. Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *European Conference in Machine Learning and Knowledge Discovery in Databases ECML*, volume 8725 of *Lecture Notes in Computer Science*, pages 630–645. Springer, 2014.

[16] Amirmohammad Rooshenas and Daniel Lowd. Learning sum-product networks with direct and indirect variable interactions. In *International Conference on Machine Learning*, pages 710–718. PMLR, 2014.

[17] Jae-sun Seo, Jyotishman Saikia, Jian Meng, Wangxin He, Han-sok Suh, Anupreetham, Yuan Liao, Ahmed Hasssan, and Injune Yeo. Digital versus analog artificial intelligence accelerators: Advances, trends, and emerging designs. *IEEE Solid-State Circuits Magazine*, 14(3):65–79, 2022. doi: 10.1109/MSSC.2022.3182935.

[18] N. Shah, L. I. G. Olascoaga, S. Zhao, W. Meert, and M. Verhelst. 9.4 piu: A 248gops/w stream-based processor for irregular probabilistic inference networks using precision-scalable posit arithmetic in 28nm. In

2021 IEEE International Solid- State Circuits Conference (ISSCC), volume 64, pages 150–152, 2021. doi: 10.1109/ISSCC42613.2021.9366061.

[19] N. Shah, W. Meert, and M. Verhelst. Dpu-v2: Energy-efficient execution of irregular directed acyclic graphs. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1288–1307, Los Alamitos, CA, USA, oct 2022. IEEE Computer Society.

[20] Nimish Shah, Laura I Galindez Olascoaga, Wannes Meert, and Marian Verhelst. Problp: A framework for low-precision probabilistic inference. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[21] L. Sommer, J. Oppermann, A. Molina, C. Binnig, K. Kersting, and A. Koch. Automatic mapping of the sum-product network inference problem to fpga-based accelerators. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pages 350 – 357, 2018. doi: 10.1109/ICCD.2018.00060.

[22] Lukas Sommer, Lukas Weber, Martin Kumm, and Andreas Koch. Comparison of arithmetic number formats for inference in sum-product networks on fpgas. In *2020 IEEE 28th Annual international symposium on field-programmable custom computing machines (FCCM)*, pages 75–83. IEEE, 2020.

[23] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL)*, pages 3645–3650. Association for Computational Linguistics, 2019.

[24] Martin Trapp, Robert Peharz, Hong Ge, Franz Pernkopf, and Zoubin Ghahramani. Bayesian learning of sum-product networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *32nd Conference on Neural Information Processing Systems (NeurIPS)*, pages 6344–6355, 2019.

[25] Fabrizio Ventola, Steven Braun, Zhongjie Yu, Martin Mundt, and Kristian Kersting. Probabilistic circuits that know what they don't know. *arXiv preprint arXiv: 2302.06544*, 2023.

[26] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L. Chen, B. Zhang, and P. Deaville. In-memory computing: Advances and prospects. *IEEE Solid-State Circuits Magazine*, 11(3):43–55, Summer 2019. ISSN 1943-0590. doi: 10.1109/MSSC.2019.2922889.

[27] Xiaowei Xu, Yukun Ding, Sharon Xiaobo Hu, Michael Niemier, Jason Cong, Yu Hu, and Yiyu Shi. Scaling for edge inference of deep neural networks. *Nature Electronics*, 1(4):216–222, 2018.