uGEMM: Unary Computing for GEMM Applications

Di Wu, Jingjie Li, Ruokai Yin, Younghyun Kim, and Joshua San Miguel, *University of Wisconsin–Madison, Madison, WI, 53706, USA*

Hsuan Hsiao, *University of Toronto, Toronto, ON, M5S 1A1, Canada*

General matrix multiplication (GEMM) is pervasive in various domains, such as signal processing, computer vision, and machine learning. Conventional binary architectures for GEMM exhibit poor scalability in area and energy efficiency, due to the spatial nature of number representation and computing. On the contrary, unary computing processes data in temporal domain with extremely simple logic. However, to date, there rarely exist efficient architectures for unary GEMM. In this work, we first present uGEMM, a hardware-efficient unary GEMM architecture enabled by universally compatible arithmetic units, which simultaneously achieves input-insensitivity and high output accuracy. Next, we demonstrate that the proposed uGEMM can reliably early terminate the computation and offers dynamic energy-accuracy scaling for real-world applications via an accuracy-aware metric. Finally, to propel the future research for unary computing, we open source our unary computing simulator, UnarySim.

General matrix multiplication (GEMM) is ubiquitous and essential in many applications, particularly emerging deep learning. Conventional binary GEMM implementations can leverage either hardware for efficiency or software for flexibility.[1] However, as the parallelism increases, binary GEMM implementations, which compute on multiple parallel bits, suffer from poor hardware area, power, and energy efficiency due to exponentially growing wire congestion.

To enable efficient GEMM processing on extremely area-, power-, or energy-constrained devices, unary computing has been leveraged in prior works,[2,3] which employ extremely simple logic that consumes unary data in the form of serial bit streams. Unary computing converts computations from the spatial domain (i.e., traditional parallel binary computing) to the temporal domain, enabling ultra-low-power hardware at the cost of longer latency. There are, however, a widely disparate set of unary computing schemes proposed in the literature (e.g., rate-coded stochastic computing,[4] temporal-coded race logic[3]), with each designed for application-specific use cases.[1] Stochastic computing supports both arithmetic and relational operations but can often be inaccurate,[5] while race logic, though accurate, supports only limited arithmetic operations and is not applicable to GEMM.[1]

Such disparities and limitations pose three fundamental challenges to an ideal unary GEMM architecture. 1) How can a designer figure out which unary computing units would work? 2) How can a designer ensure reliability, i.e., accuracy, while improving energy efficiency? 3) How can a designer simulate and explore the disparate design space? We propose the *unified unary GEMM architecture*, dubbed uGEMM, along with our *UnarySim* simulator to offer answers to these questions, striving to make unary computing a first-class citizen in resource-constrained systems.

To tackle the first challenge, we present new mechanisms for unary arithmetic units and design universally compatible microarchitectures. These units are highly accurate and support arbitrary input bit streams, i.e., *input-insensitivity*, as indicated in the input in Figure 1. Furthermore, they process bit streams in a fully streaming manner without unary–binary data interconversion, allowing for streaming consecutive uGEMM blocks in Figure 1. The resultant

0272-1732 © 2021 IEEE
Digital Object Identifier 10.1109/MM.2021.3065369
Date of publication 11 March 2021; date of current version 25 May 2021.

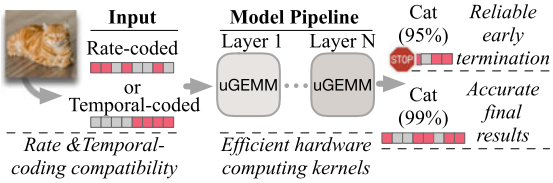

FIGURE 1. Illustrative example of uGEMM.

uGEMM, a multiply-accumulate (MAC) array of these novel units, naturally inherits all microarchitecture-level benefits and outperforms counterparts. Then, the second challenge is solved via a novel metric, named *stability*, which monitors the output accuracy in the temporal domain. Understanding the accuracy evolution through time, we can early terminate the computation, as shown in the output in Figure 1, to achieve a desired efficiency reliably, i.e., in a controllable manner. The final challenge on the design space exploration is also addressed in this work by open-sourcing our unary computing simulator, *UnarySim*, based on which the evaluations at microarchitecture, architecture, and application levels can be performed.[1,6,7]

BACKGROUND

According to the data encoding methods, we categorize unary computing into rate-coded stochastic computing[4] and temporal-coded race logic,[3] with examples of data representations and limitations shown in Figure 2.

Rate-Coding-Based Unary Computing

Stochastic computing adopts *rate coding*, whose data value relies on the *frequency* of ones and zeros in the bit stream, with each bit generated by comparing source data with a random number generator (RNG).[4] According to the polarity, bit streams can be in unipolar or bipolar (unsigned or signed) formats. Given a bit stream with the probability/frequency of ones as $P(S=1) \in [0,1]$, unipolar and bipolar values are $P(S=1)$ and $2 \times P(S=1) - 1$, respectively. Any real number can be scaled and mapped into a bit stream.

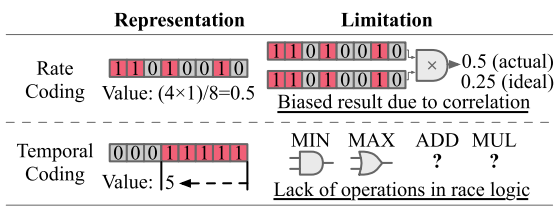

FIGURE 2. Unary computing data representations and limitations. .

Stochastic computing performs computation by manipulating input bit streams *statistically*. In stochastic computing, an adder with two inputs A and B can be implemented by a two-input multiplexer (MUX) whose select signal is a stochastic bit stream with $P(S=1)=0.5$. This computes $V_{out} = (V_A + V_B)/2$, where a scaling factor of 2 is introduced to prevent overflow. Besides the scaled addition using MUX, unipolar nonscaled addition can be done with an OR gate. Multiplication can be achieved using an AND gate for unipolar bit streams and an XNOR gate for bipolar bit streams.

Although such extreme simplicity offers benefits in hardware efficiency, stochastic computing suffers the correlation problem[5] (i.e., two bit streams are more correlated when they are more similar), leading to inaccuracy. For example, an AND gate only accurately computes the product of two input bit streams when they have zero correlation. Otherwise, if more paired ones appear than expected, it ends up computing the minimum function,[7] as shown in Figure 2. Existing solutions raise the demands for costly RNGs and increase the latency to achieve accurate results.[8,9]

Temporal-Coding-Based Unary Computing

Temporal coding is applied in race logic, which encodes data into the timing of a signal's transition (or edge), with the bit stream as a chain of ones followed by a chain of zeros, or *vice versa*, and each bit generated by comparing source data with a counter output.[3] The polarity categorization of temporal-coded bit streams can follow that in stochastic computing.

In race logic, an AND gate and an OR gate now perform minimum and maximum functions, respectively, unlike the unipolar multiplication and unipolar nonscaled addition in stochastic computing. As the signal edges are deterministic, race logic can compute the minimum and maximum accurately. However, prior to our work, multipliers and adders have never been proposed for temporal computing.

UGEMM MICROARCHITECTURE AND ARCHITECTURE

In this section, we present novel linear functional unit designs in Figure 3, including multiplication (uMUL), scaled, and nonscaled additions (uSADD and uNSADD), which are universally compatible to varying input codings, and the integrated uGEMM architecture in Figure 4.

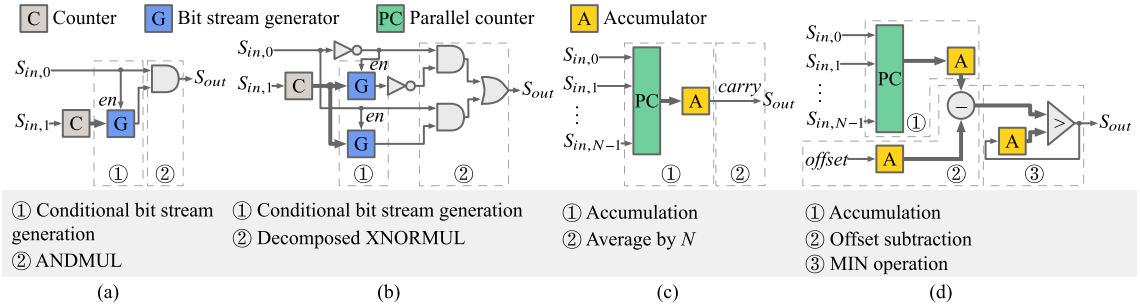

FIGURE 3. uGEMM functional units. Thick line: binary signal; and thin line: unary signal.

uMUL

The proposed unipolar and bipolar uMUL are shown in Figure 3(a) and (b). We recognize that unary multiplication using a naive AND gate [② in Figure 3(a)] conditionally produces a valid output bit when input 0 is logic one, implying a conditional bit stream generation for input 1 [① in Figure 3(a)]. More specifically, only when input 0 is logic one, the RNG inside the bit stream generator for input 1 will update, and the generator will eventually generate a new bit, i.e., input 0 is an enable signal to the bit stream generator. As such, we thoroughly eliminate the correlation problem and achieve high accuracy with merely an extra enable signal. Then, for bipolar multiplication using an XNOR gate, we decompose the XNOR gate to two AND gates, with each leveraging the conditional bit stream generation for high accuracy.

uSADD

Scaled addition calculates the average of all inputs, where the average comes via scaling the output. For scaled addition, the unipolar and bipolar microarchitectures are identical, as in Figure 3 (c). The takeaway here is that the input average is the mean of input-one counts. Therefore, the proposed uSADD first collects all input bits with the parallel counter and compute the sum in the accumulator as in block ①. Then, the output is set to the carry bit of the accumulator in block ②. This means that only when there are N ones in the input, a logic one at the output will be generated, exactly the output scaling. Such a carry bit overflow mechanism no longer considers the correlation between the input bit streams, and obtains accurate results.

uNSADD

Nonscaled addition calculates the clipped sum of all inputs, and the clipping happens when the sum overflows/underflows the legal unary data range. The parallel counter and accumulator in block ① are the same as in uSADD, except that now the accumulation output is entirely utilized, rather than only the carry bit. Next, a subtraction between the offset and the accumulation result is performed in block ② to retrieve the anticipated count of output ones. The offset is of value 0 and $(N-1)/2$ for unipolar and bipolar bit streams, respectively. Finally, block ③ generates the output based on the anticipated one count and the

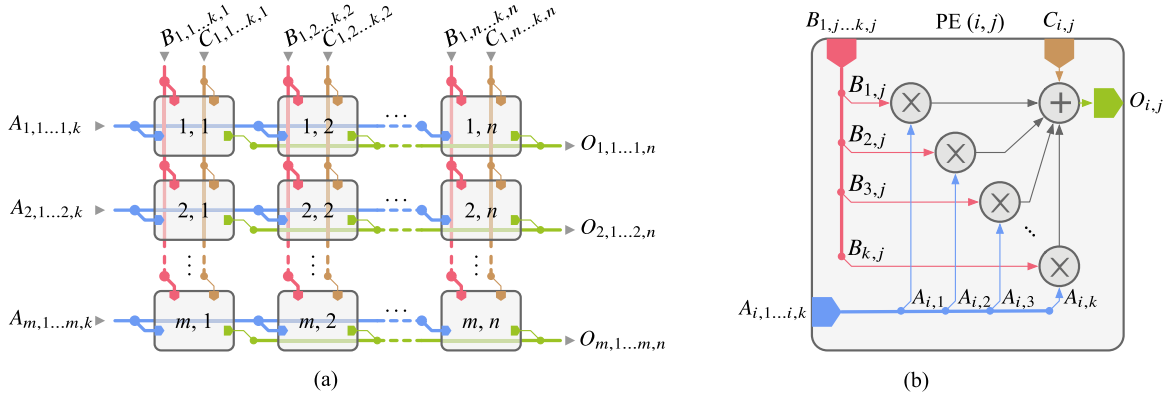

FIGURE 4. uGEMM architecture and its PE. Thick line: multibit stream; and thin line: single-bit stream.

actual one count in the accumulator via comparison. When there are more anticipated ones than actual, logic ones will be output every cycle until they are equal. Similar to uSADD, uNSADD computes based on one counts and solves the correlation issue. Note that we are the first to support bipolar nonscaled addition in unary computing.

Integrated uGEMM Architecture

Our uGEMM can simply be built on above units, as they are universally compatible to varying inputs (i.e., input-insensitive) and directly take in/produce bit streams as inputs and outputs (i.e., fully streaming process). We use $O = A \times B + C$ as an example to illustrate the uGEMM architecture, where O, A, B and C are of size $(m \times n)$, $(m \times k)$, $(k \times n)$, and $(m \times n)$, respectively. Figure 4(a) shows the uGEMM architecture. Input A, B, and C go through the m-by-n processing element (PE) array in the center of the figure. As shown in Figure 4(b), the (i,j)th PE performs the MAC of the ith row from A and the jth column from B, with k elements for each row and column, then adds the (i,j)th element from C as the result. Inheriting from the features of uMUL, uSADD, and uNSADD, uGEMM achieves 1) high parallelism due to simple unary logic, 2) input-insensitivity and high output accuracy by diminishing the correlation problem, and 3) fully streaming process to enable reliable early termination for controllable energy efficiency boost. More details on related mathematical theories and walkthrough examples can be found in Wu *et al.*'s work.[1]

ENERGY-ACCURACY SCALING EVALUATION

We list the hardware implementation result of uGEMM in Table 1. Here, we fix the GEMM shape to $m = k = n = 16$, and evaluate uGEMM against Gaines',[4] Sim's,[10] Jenson's[8] and Najafi's[9] unary schemes. Note that uGEMM is the only one supporting bipolar nonscaled addition and that Najafi's design does not support temporal coding. We find that uGEMM consumes small area and low power, outperforming Gaines,' Sim's and Najafi's designs. Jenson's unary GEMM is significantly more lightweight than the others due to its resource sharing scheme. However, Jenson's design introduces long latency, resulting in higher energy consumption.

Early terminating a bit stream in the temporal domain leads to a partial bit stream, which represents a less accurate value, but reduces the latency to offer higher energy efficiency. As a newly proposed metric for the capability of early termination in this work,

TABLE 1. Hardware comparison of unary GEMM implementations.

GEMM	Area (mm^2)	Power (W)	Latency (μs)	Energy (μJ)
Unipolar scaled				
uGEMM	0.43	0.15	0.64	0.07
Gaines	1.57	0.50	0.64	0.32
Sim	0.52	0.18	0.64	0.11
Jenson	0.08	0.02	163.84	3.91
Najafi	1.26	0.46	0.64	0.29
Unipolar nonscaled				
uGEMM	0.44	0.12	0.64	0.07
Gaines	1.55	0.49	0.64	0.31
Sim	0.50	0.15	0.64	0.09
Bipolar scaled				
uGEMM	0.76	0.21	0.64	0.13
Gaines	1.56	0.50	0.64	0.32
Sim	0.53	0.18	0.64	0.12
Jenson	0.08	0.02	163.84	3.90
Najafi	1.25	0.45	0.64	0.29
Bipolar nonscaled				
uGEMM	0.77	0.20	0.64	0.13

stability measures how early a bit stream's value stabilizes/converges, given a specific error budget. For a bit stream of length L, V_L represents its expected value and V_l represents the value of the partial bit stream based on the first l bits ($l \leq L$). If starting from the lth bit, the bias $\Delta V_l = |V_l - V_L|$ is consistently smaller than a user-defined threshold V_{THD}, then stability is calculated as follows:

$$\text{Stability} = 1 - \frac{\max\{l | \Delta V_l > V_{\text{THD}}\}}{L}. \quad (1)$$

Stability ranges in [0,1], with a higher value indicating earlier termination. Usually, rate coding yields higher stability than temporal coding. Given a V_{THD}, the resultant maximum l is defined as the *stable point*, after which the output accuracy, statistically, will never drop below the threshold. More details on the usage of this metric at the microarchitecture, architecture, and application levels can be found in Wu *et al.*'s works.[1,6]

In addition to high accuracy, our unary multiplication and addition designs have consistently higher

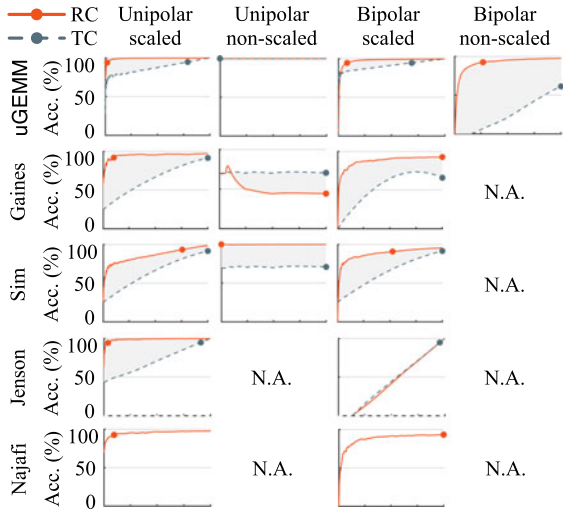

FIGURE 5. Progressive accuracy (curves) and stability (dots) comparison of GEMMs. Cycle ranges from 1 to 256 in uGEMM, Gaines,' Sim's and Najafi's; and from 1 to 256^2 in Jenson's. RC: rate-coded input; TC: temporal-coded input.

stability than that of prior counterparts. Such high stability indicates that the maximum I is smaller for our designs, implying that under the same error threshold, our design can early terminate the computation earlier than others to boost energy efficiency. By setting $V_{THD} = 0.05$, we present the accuracy evolution through time (i.e., progressive accuracy) and the stable point for various unary GEMM implementations in Figure 5. We observe that uGEMM outperforms all others in the final accuracy for both inputs, as well as their difference, suggesting input-insensitivity. Then, the stable points of uGEMM are also closer to the y-axis in this plot, demonstrating that with the same error budget, early termination in uGEMM happens earlier than others, providing even higher energy efficiency benefits on top of the results in Table 1. Therefore, for each configuration and input type, uGEMM outperforms all other approaches in terms of how early its accuracy can stabilize, i.e., high output stability.

Above results demonstrate that uGEMM is more suitable than its counterparts for ultra-low-power architectures, especially considering its reliable energy efficiency boost via the accuracy-aware metric.

UNARYSIM: A UNARY COMPUTING SIMULATOR

Although there exist tools for synthesizing unary circuits,[11] there are still little to no publicly available toolchains for rapid design space exploration and simulation of general unary architectures and applications. Given how diverse and specialized prior unary works are, the gap between the demand for ultra-low-power architecture exploration and the lack of off-the-shelf toolchains motivated us to provide a standard means for characterizing different unary designs and fairly quantifying their tradeoffs by open-sourcing our unary computing simulator: *UnarySim*.[12]

IN ADDITION TO HIGH ACCURACY, OUR UNARY MULTIPLICATION AND ADDITION DESIGNS HAVE CONSISTENTLY HIGHER STABILITY THAN THAT OF PRIOR COUNTERPARTS.

UnarySim is a cycle-accurate simulator to capture the precise behavior in a unary computing architecture, as shown in Figure 6. The entire simulator consists of two parts, including the software simulation and the hardware implementation. The backbone of the software simulation is the PyTorch deep learning framework from Facebook; as such, *UnarySim* naturally supports deep learning applications and is highly scalable and extensible. *UnarySim* inherits the high scalability of PyTorch by decoupling the high-level architecture design and the low-level performance simulation. The key simulated components are categorized into stream, kernel, and metric as follows.

1) The stream components are used to generate the bit streams using either rate or temporal coding and manipulate a pair of bit streams to exhibit a specific correlation, so as to cover major use cases in existing literature.[1,4,5,7]
2) The kernel components refer to the unary computing units for both arithmetic (linear and nonlinear) and logical operations. The proposed uGEMM microarchitectures and architecture are all covered in this category, as well as some

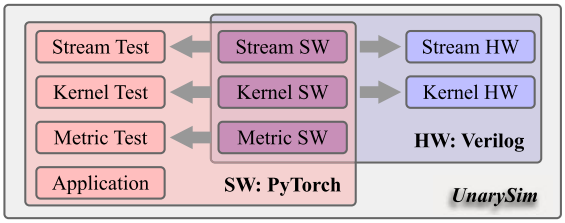

FIGURE 6. *UnarySim* diagram.

other general operations for deep learning and computer vision.

3) The metric components include key unary computing metrics in existing literature, like correlation,[5] accuracy, and our proposed stability metric.[1,6] Those metrics can be seamlessly pinned to any node in the system during simulation and monitor the system status over time, which can be used to tune the system accuracy and latency.

WE INTRODUCE OUR DEDICATED SIMULATOR FOR UNARY COMPUTING, UNARYSIM, WHICH CAN SIGNIFICANTLY LOWER THE LEARNING CURVE TOWARD RAPID AND RELIABLE UNARY COMPUTING RESEARCH.

After the system is fine-tuned, these metric components can be removed to speed up simulation. All of these components in software simulation are callable PyTorch modules, which follows the PyTorch programming rules, and are provided with test examples for clarity. With the growing popularity of PyTorch in both the academia and industry, we believe that *UnarySim* offers a low barrier-to-entry for researchers: further extending *UnarySim* to support more stream, kernel and metric components simply requires constructing additional PyTorch modules. In terms of the hardware implementation, we provide component-level implementation examples for the proposed and evaluated designs in this work, as well as those in prior works.[7] Note that current metric components are merely simulated in software for performance evaluation and do not have correspondent hardware implementations. Future works for this simulator will be the extension of the components to better support broader applications and the automation of executing arbitrary algorithms on unary hardware.

CONCLUSION

Unary computing has gained growing research attention in the past decade in the fields of signal processing, computer vision, and machine learning, etc., due to its ultra-low power consumption. In this work, to further promote the research of unary computing into broader disciplines, we focus on architecting GEMM, which is ubiquitous in applications. We first present our uGEMM microarchitecture and architecture with benefits in input-insensitivity, output-stability, and hardware efficiency compared to prior art. Then, we demonstrate the accuracy-aware energy-accuracy scaling for uGEMM using our stability metric, which offers further opportunities for energy savings via early termination. Finally, we introduce our dedicated simulator for unary computing, *UnarySim*, which can significantly lower the learning curve toward rapid and reliable unary computing research.

REFERENCES

1. D. Wu *et al.*, "UGEMM: Unary computing architecture for GEMM applications," in *Proc. 47th Annu. Int. Symp. Comput. Archit.*, 2020, pp. 377–390, doi: 10.1109/ISCA45697.2020.00040.
2. V. T. Lee, A. Alaghi, R. Pamula, V. S. Sathe, L. Ceze, and M. Oskin, "Architecture considerations for stochastic computing accelerators," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2277–2289, Nov. 2018, doi: 10.1109/TCAD.2018.2858338.
3. A. Madhavan, T. Sherwood, and D. Strukov, "Race logic: A hardware acceleration for dynamic programming algorithms," in *Proc. 41st Annu. Int. Symp. Comput. Archit.*, 2014, pp. 517–528, doi: 10.1109/ISCA.2014.6853226.
4. B. R. Gaines, "Stochastic computing systems," in *Proc. Adv. Inf. Syst. Sci.*, 1969, pp. 37–172, doi: 10.1007/978-1-4899-5841-9_2.
5. A. Alaghi and John P. Hayes, "Exploiting correlation in stochastic circuit design," in *Proc. 31st Int. Conf. Comput. Des.*, 2013, pp. 39–46, doi: 10.1109/ICCD.2013.6657023.
6. D. Wu, R. Yin, and J. SanMiguel, "Normalized stability: A cross-level design metric for early termination in stochastic computing," in *Proc. 26th Asia South Pacific Des. Autom. Conf.*, 2021, pp. 254–259, doi: 10.1145/3394885.3431549.
7. D. Wu and J. SanMiguel, "In-stream stochastic division and square root via correlation," in *Proc. IEEE/ACM 56th Des. Autom. Conf.*, 2019, pp. 1–6, doi: 10.1145/3316781.3317844.
8. D. Jenson and M. Riedel, "A deterministic approach to stochastic computation," in *Proc. 35th Int. Conf. Comput.-Aided Des.*, 2016, pp. 1–8, doi: 10.1145/2966986.2966988.
9. M. H. Najafi, D. Jenson, D. J. Lilja, and M. D. Riedel, "Performing stochastic computation deterministically," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 27, no. 12, pp. 2925–2938, Dec. 2019, doi: 10.1109/TVLSI.2019.2929354.
10. H. Sim and J. Lee, "A new stochastic computing multiplier with application to deep convolutional neural networks," in *Proc. 54th Des. Autom. Conf.*, 2017, pp. 1–6, doi: 10.1145/3061639.3062290.

11. K. Daruwalla, H. Zhuo, R. Shukla, and M. Lipasti, "BitSAD v2: Compiler optimization and analysis for bitstream computing," *ACM Trans. Archit. Code Optim.*, vol. 16, no. 4, pp. 1–25, 2019, doi: 10.1145/3364999.
12. D. Wu and R. Yin, "UnarySim." Accessed: Jan. 31, 2021. [Online]. Available: https://github.com/diwu1990/UnarySim

DI WU is currently a Ph.D. candidate with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI, USA. His research interests include stochastic and approximate computing, as well as numerical optimization for deep neural networks. Wu received B.S. and M.Eng. degrees from Fudan University in 2012 and 2015, respectively. He is a member of ACM. He is the corresponding author of this article. Contact him at di.wu@ece.wisc.edu.

JINGJIE LI is currently a Ph.D. candidate with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI, USA. His research interests include human-centered computing, Internet of Things, and embedded systems. Li received a B.S. degree from Beijing Institute of Technology and a B.Eng. (R&D) degree (Hons.) from The Australian National University in 2017. He is a Student Member of IEEE and ACM. Contact him at jingjie.li@wisc.edu.

RUOKAI YIN is currently a senior undergraduate at the University of Wisconsin-Madison, Madison, WI, USA, majored in electrical engineering and computer science. His research interests include design high-performance computer architecture for machine learning. Currently, he is working with Prof. Joshua San Miguel on unary computing and applying this paradigm to multiple real-world applications. Contact him at ryin25@wisc.edu.

YOUNGHYUN KIM is currently an Assistant Professor of Electrical and Computer Engineering at the University of Wisconsin-Madison, Madison, WI, USA. His research interests include energy-efficient computing, machine learning at the edge, and cyber-physical systems. Kim received a Ph.D. degree in electrical engineering and computer science from Seoul National University in 2013. Before joining University of Wisconsin-Madison in 2016, he was a postdoc at Purdue University, West Lafayette, IN, USA. He is a member of IEEE and ACM. Contact him at younghyun.kim@wisc.edu.

JOSHUA SAN MIGUEL is currently an Assistant Professor at the University of Wisconsin-Madison, Madison, WI, USA. His research interests include stochastic and approximate computing, intermittent computing, and interconnection networks. San Miguel received a Ph.D. degree in electrical and computer engineering from the University of Toronto. He is a member of IEEE and ACM. Contact him at jsanmiguel@wisc.edu.

HSUAN (JULIE) HSIAO is currently a Ph.D. candidate with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. Her research interests include stochastic computing and computer-aided design. Hsiao received an M.A.Sc. degree in electrical and computer engineering from the University of Toronto. Contact her at julie.hsiao@mail.utoronto.ca.