Helix: Algorithm / Architecture Co-design for Accelerating Nanopore Genome Base-calling

Qian Lou
louqian@iu.edu
Indiana University Bloomington

Sarath Chandra Janga
scjanga@iupui.edu
Indiana University - Purdue University Indianapolis

Lei Jiang
jiang60@iu.edu
Indiana University Bloomington

ABSTRACT

Nanopore genome sequencing is the key to enabling personalized medicine, global food security, and virus surveillance. The state-of-the-art base-callers adopt deep neural networks (DNNs) to translate electrical signals generated by nanopore sequencers to digital DNA symbols. A DNN-based base-caller consumes 44.5% of total execution time of a nanopore sequencing pipeline. However, it is difficult to quantize a base-caller and build a power-efficient processing-in-memory (PIM) to run the quantized base-caller. Although conventional network quantization techniques reduce the computing overhead of a base-caller by replacing floating-point multiply-accumulations by cheaper fixed-point operations, it significantly increases the number of systematic errors that cannot be corrected by read votes. The power density of prior nonvolatile memory (NVM)-based PIMs has already exceeded memory thermal tolerance even with active heat sinks, because their power efficiency is severely limited by analog-to-digital converters (ADC). Finally, Connectionist Temporal Classification (CTC) decoding and read voting cost 53.7% of total execution time in a quantized base-caller, and thus became its new bottleneck.

In this paper, we propose a novel algorithm/architecture co-designed PIM, Helix, to power-efficiently and accurately accelerate nanopore base-calling. From algorithm perspective, we present systematic error aware training to minimize the number of systematic errors in a quantized base-caller. From architecture perspective, we propose a low-power SOT-MRAM-based ADC array to process analog-to-digital conversion operations and improve power efficiency of prior DNN PIMs. Moreover, we revised a traditional NVM-based dot-product engine to accelerate CTC decoding operations, and create a SOT-MRAM binary comparator array to process read voting. Compared to state-of-the-art PIMs, Helix improves base-calling throughput by 6×, throughput per Watt by 11.9× and per mm^2 by 7.5× without degrading base-calling accuracy.

CCS CONCEPTS

• **Hardware → Spintronics and magnetic technologies**; • **Applied computing → Computational genomics**.

KEYWORDS

nanopore sequencing; base-calling; processing-in-memory

ACM Reference Format:
Qian Lou, Sarath Chandra Janga, and Lei Jiang. 2020. Helix: Algorithm / Architecture Co-design for Accelerating Nanopore Genome Base-calling. In *Proceedings of the 2020 International Conference on Parallel Architectures and Compilation Techniques (PACT '20), October 3–7, 2020, Virtual Event, GA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3410463.3414626

1 INTRODUCTION

Genome sequencing [8, 21, 34, 35, 37] is a cornerstone for enabling personalized medicine, global food security, and virus surveillance. The emerging nanopore genome sequencing technology [15] is revolutionizing the genome research, industry and market due to its ability to generate ultra-long DNA fragments, aka **long reads**, as well as provide portability. Producing long reads [23] is the key to improving the quality of *de novo* assembly, spanning repetitive genomic regions, and identifying large structural variations. Moreover, portable real-time USB Flash drive size nanopore sequencers, MinION [15] and SmidgION [24], have demonstrated their power in tracking genomes of Ebola [12], Zika [6] and COVID-19 [20] viruses during disease outbreaks.

Compared to conventional short-read Illumina sequencing, nanopore sequencing suffers high error rate [15], e.g., 12%. A nanopore sequencer measures changes in electrical current as organic DNA fragments pass through its pore. Due to the tiny amplitude of currents triggered by DNA motions, a nanopore sequencer inevitably introduces noises into raw electrical signals, thus producing sequencing errors. A base-caller translates raw electrical signals to digital DNA symbols, i.e., $[A, C, G, T]$. In order to reduce sequencing errors, a sequencing machine generates multiple unique reads [15] that include a given DNA symbol. These reads are base-called individually, and then assembled to decide the correct value of each DNA symbol. The number of unique reads containing a given DNA symbol is called coverage. Typically, the coverage is between 30 ∼ 50 [29, 33, 36]. To further enhance base-calling accuracy, recent works [3, 7, 29, 33, 36] use deep neural networks (DNNs) for base-calling. A DNN-based base-caller, e.g., Guppy [36], Scrappie [29], and Chiron [33], consists of convolutional, recurrent, fully-connected layers, as well as a Connectionist Temporal Classification (CTC) decoder. Although achieving high base-calling accuracy, prior DNN-based base-callers are slow. For instance, Guppy with its high base-calling accuracy obtains only 1 million base pairs per second (bp/s) on a server-level GPU. *At such a speed, it takes 25 hours for Guppy to base-call a 3G-bp human genome with a* 30× *coverage.* During virus outbreaks, it is challenging for even a data center equipped with powerful GPUs to processing base-calling for a large

PACT '20, October 3–7, 2020, Virtual Event, GA, USA

ACM ISBN 978-1-4503-8075-1/20/10…$15.00
https://doi.org/10.1145/3410463.3414626

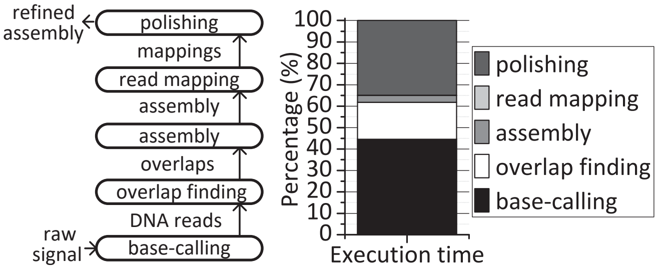


Figure 1: The pipeline of nanopore sequencing.

group of presumptive positive patients. As a result, base-calling becomes the most time-consuming step in a nanopore sequencing pipeline [30].

Recently, both industry [19] and academia [18, 38] proposed network quantization algorithms to power-efficiently accelerate DNN inferences without sacrificing inference accuracy by approximating inputs, weights and activations of a DNN to fixed-point representations with smaller bit-widths. In this way, computationally expensive floating-point multiply-accumulates (MACs) in a DNN can be replaced by fixed-point operations. Besides conventional CPUs and GPUs, FPGAs and ASICs are adopted to accelerate quantized DNN inferences in data centers. Moreover, to further overcome the *von Neumann* bottleneck in data centers, recent search efforts use various nonvolatile memory (NVM) technologies including ReRAM [31, 40], PCM [1] and STT-MRAM [39] to build processing-in-memory (PIM) accelerators to process quantized DNN inferences in memory arrays.

However, it is difficult to apply prior network quantization techniques on base-callers and accelerate quantized base-callers by state-of-the-art NVM PIM architectures. Naïvely quantizing a base-caller via prior network quantization algorithms substantially increases the number of *systematic* errors that cannot be corrected by voting operations among multiple reads containing the same DNA symbols. Furthermore, state-of-the-art PIM accelerators take advantage of analog computing to maximize inference throughput of quantized DNNs, but the functioning of their analog computing style heavily depends on a large number of CMOS analog-to-digital converters (ADCs) that significantly increase their power consumption and area overhead. For instance, CMOS ADCs cost 58% of power consumption and 30% of chip area in a typical PIM design [31]. Finally, state-of-the-art NVM PIM designs cannot process some essential operations of a base-caller such as CTC decoding and read voting that usually consume >50% of total execution time in a quantized base-caller.

In this paper, we propose a novel algorithm and architecture co-designed PIM accelerator, *Helix*, to efficiently and accurately process quantized nanopore base-calling. Our contributions are summarized as:

- **Systematic error aware training**. We present systematic error aware training (SEAT) to reduce the number of systematic errors that cannot corrected by read votes in a quantized base-caller. We introduce a new loss function to indirectly minimize the edit distance between a consensus read and its ground truth DNA sequence. SEAT enables 5-bit quantized base-callers to achieving their full-precision base-calling accuracy.

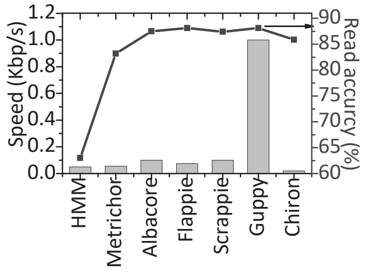


Figure 2: Base-caller comparison.

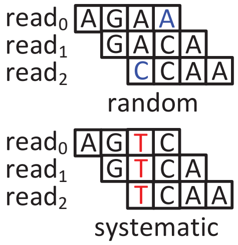


Figure 3: Errors.

- **An ADC-free PIM accelerator**. We propose a Spin Orbit Torque MRAM (SOT-MRAM)-based array architecture to accelerate analog-to-digital conversion operations without CMOS ADCs. We also show our SOT-MRAM ADC arrays are resilient to process variation. We modify a conventional NVM-based dot-product engine to accelerate CTC decoding operations, and then present a SOT-MRAM-based binary comparator array to process read voting operations in a quantized base-caller.
- **Base-calling accuracy and throughput**. We implemented all proposed techniques of Helix and compared Helix against state-of-the-art PIM designs that accelerate quantized DNN inferences. Experimental results show that, compared to state-of-the-art PIM accelerators, Helix improves base-calling throughput by 28×, throughput per Watt by 80×, and throughput per mm^2 by 27× without degrading accuracy.

2 BACKGROUND

2.1 Nanopore Sequencing Pipeline

As Figure 1 shows, a nanopore sequencing pipeline [30] consisting of *base-calling*, *overlap finding*, *assembly*, *read mapping*, and *polishing* is employed to generate a digital assembly. The input of a pipeline is raw electrical signals produced by nanopore sequencers, e.g., MinION [15] and SmidgION [24]. Base-calling translates raw signal data to digital DNA symbols, i.e., $[A, C, G, T]$. Overlap finding computes all suffix-prefix matches between each pair of reads, and then generates an overlap graph, where each node denotes a read and each edge indicates the suffix-prefix match between two nodes. The assembly step traverses an overlap graph to construct a draft assembly. Base-called reads are mapped to the generated draft assembly by read mapping. Lastly, the final assembly is polished.

2.2 Nanopore Base-calling

DNN-based base-caller. DNNs are adopted to filter noises and accurately translate raw electric signals to digital DNA symbols. A DNN-based base-caller typically consists of multiple convolutional (Conv), gated recurrent unit (GRU), and fully-connected (FC) layers. The convolutional layers recognize local patterns in input signals, whereas the GRU layers integrate these patterns into base-calling probabilities. A CTC decoder is used to compute digital DNA symbols according to the base probabilities. Compared to the Hidden Markov Model (HMM) [22], a series of DNN-based base-callers including Metrichor [27], Albacore [26], Flappie [7], Scrappie [29], Guppy [36], and Chiron [33], significantly improve base-calling accuracy, as shown in Figure 2. Among all base-callers, the Oxford Nanopore Technologies official GPU-based base-caller,

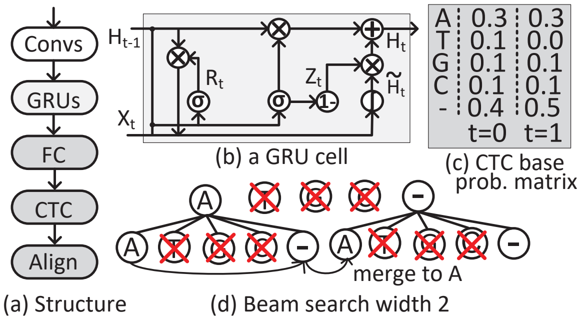


Figure 4: The DNN architecture of Guppy.

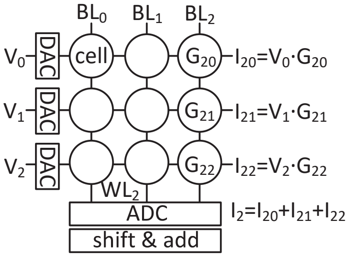


Figure 5: A dot-product engine.

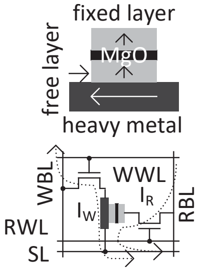


Figure 6: SOT-MRAM.

Guppy, achieves the best accuracy and the highest speed. We selected Guppy as our base-caller baseline, and also considered other DNN-based base-callers in §6. Due to complex DNN structures, base-callers are generally slow [36]. As a result, base-calling consumes 44.5% [30] of total execution time of a nanopore sequencing pipeline. The details of base-callers are introduced in §5.2.

Base-calling error. We define the number of base-calling errors as the edit distance between a read predicted by a base-caller and its ground truth. The edit distance quantifies how dissimilar two reads are to one another by counting the minimum number of insertions, deletions, and substitutions required to transform one into the other. To enhance base-calling accuracy, a base-caller translates each signal data multiple times and generates multiple reads containing the same signal data. At the end of base-calling, each DNA symbol value is decided by votes among all reads containing its corresponding signal data. As Figure 3 shows, for a DNA symbol, if base-calling errors randomly occur among reads, the voting result can still be correct, since most reads have the correct value. This is a *random* error. However, for a DNA symbol, if base-calling errors happen in a systematic way, i.e., all copies of a signal are translated to the same wrong value, it is impossible to produce the correct value by read voting. It is a *systematic* error.

Convolutional layer. As Figure 4a shows, a base-caller includes multiple convolutional layers to process raw electric signals. The first convolutional layer receives an $L \times N$ floating-point signal vector, where L is the input length; and N indicates the input channel number, e.g., $L = 5$ and $N = 1$. Then, it uses a $K \times N \times M$ weight filter to convolve with the input vector to generate an output vector for the next activation layer [33], where K is the weight kernel size; and M means the output channel number, e.g., $K = 2$, and $M = 256$. The $L \times N$ floating-point signal vector is generated by a fixed-size window sliding on the entire signal data array. After a base-calling operation, the sliding window moves forward by T elements [33], where T is the sliding offset, e.g., $T = 1$. The base-caller then works on a new signal vector. At the end of base-calling, $\lfloor L/T \rfloor$ reads containing the same signal element vote for its value.

GRU Layer. A base-caller uses a set of GRU layers to integrate patterns produced by convolutional layers into base-calling probabilities. As Figure 4b describes, a GRU layer receives an input X_t and its output of the last time step H_{t-1}. And then, it uses two memory cells, R_t and Z_t, to reset and update the gate state at the time step t. The output H_t of a GRU layer can be computed as

$$\begin{aligned} Z_t &= \sigma(W_z X_t + U_z H_{t-1}) + b_z \\ R_t &= \sigma(W_r X_t + U_r H_{t-1}) + b_r \\ \tilde{H_t} &= \int(W_h X_t + U_h(R_t \otimes H_{t-1})) + b_h \\ H_t &= Z_t \otimes H_{t-1} + (1 - Z_t) \otimes \tilde{H_t} \end{aligned} \tag{1}$$

where W_z, U_z, W_r, U_r, W_h and U_h are weights for Z_t, R_t and hidden state $\tilde{H_t}$ respectively; b_z, b_r and b_h are their biases; σ is the *sigmoid* activation; $\int$ indicates the *tanh* activation; and $\otimes$ means element-wise multiplications.

CTC decoder. Since it is difficult for a nanopore sequencer to precisely control DNA motions at uniform speed, multiple elements in the input signal vector may be generated by a single DNA nucleotide [33]. A base-caller adopts a CTC decoder [10, 11] to map an input signal vector $R = [I_0, I_1, \ldots, I_{L-1}]$ to a corresponding digital read $D = [H_0, H_1, \ldots, H_{Z-1}]$, where $L \neq Z$; and there is no alignment between R and D. More specifically, convolutional, GRU and FC layers provide all symbol probabilities $p_t(a_t|R)$ for each time step, where $a_t \in [A, C, G, T, -]$ ($-$ indicates blank). The probabilities $p_t(a_t|R)$ of a symbol of all time steps form a base probability matrix, as shown in Figure 4c. By looking up the base probability matrix, a CTC decoder can decide the probability of a read. The probability of D is calculated by

$$p(D|R) = \sum_{A \in \mathbb{A}_{D,R}} \prod_{t=0}^{L-1} p_t(a_t|R) \tag{2}$$

where $\mathbb{A}_{D,R}$ indicates all valid alignments between D and R. The CTC decoder infers the most likely read by a *beam* search on the matrix. As Figure 4d highlights, during a beam search with width 2, the CTC decoder keeps only the symbols with the top-2 largest probabilities at each time step. At $t = 0$, it keeps A and $-$. At $t = 1$, the decoder calculates the probabilities for various 2-symbol reads including $p(AA) = 0.3 * 0.3 = 0.09$, $p(A-) = 0.15$, $p(-A) = 0.12$, and $p(--) = 0.2$. Since AA, $A-$, $-A$ indicate A, they can be merged to A. So $p(A) = 0.09 + 0.15 + 0.12 = 0.36$. The beam search finds A as the most likely read.

2.3 Network Quantization

To reduce the computing overhead of DNNs, recent work proposes network quantization [18, 19, 38] that approximates 32-bit floating-point inputs, weights and activations to their fixed-point representations with smaller bit-widths. In this way, the quantized networks perform quantized inferences by low-cost fixed-point MACs.

2.4 NVM-based Dot-Product Engine

Various NVM-based dot-product engines (e.g., STT-MRAM [39], PCM [1], ReRAM [31]) are used to improve performance per Watt of vector-matrix multiplications by $\sim 10^3$ over conventional CMOS ASIC designs. One example of a NVM-based dot-product engine is shown in Figure 5, where the array consists of word-lines (WLs), bit-lines (BLs) and NVM cells. Each cell on a BL is programmed into a certain resistance (R), e.g., $cell_{2x}$ on BL_2 is written to R_{2x}, where $x = 0, 1, 2$. The cell conductance (G) is the inverse of the cell resistance ($\frac{1}{R}$), e.g., $cell_{2x}$ has a conductance of $G_{2x} = \frac{1}{R_{2x}}$. A voltage ($V_x$) can be applied to each WL, so that the current, e.g., I_{2x}, passing through a cell ($cell_{2x}$) to the BL is the product of the voltage and the cell conductance ($V_x \cdot G_{2x}$). Based on the Kirchhoff's law, the total current (e.g., I_2) on a BL (BL_2) is the sum of currents passing through each cell on the BL, so $I_2 = \sum_0^2 (V_x \cdot G_{2x})$. All BLs in the array produce the current sums simultaneously with the same voltage inputs along WLs. In this way, in each cycle, a vector-matrix multiplication between the input vector V and the conductance matrix G stored in the array is computed by the dot-product engine. The conversion between analog and digital signals is necessary for dot-product engines to communicate with other digital circuits. A digital-analog converter (DAC) converts digital inputs into corresponding voltages that are applied to each WL, while an ADC converts the outputs of a dot-product engine, i.e., the BL accumulated currents, to digital values.

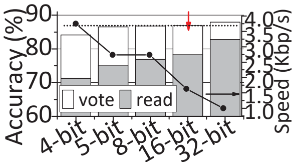


Figure 7: The accuracy & speed of quantized Guppy.

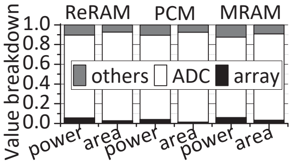


Figure 8: The area and power breakdown of NVM engines.

2.5 SOT-MRAM

Spin Orbit Torque MRAM (SOT-MRAM) [13] emerges as one of the most promising nonvolatile memory alternatives to power hungry SRAM. To record data, SOT-MRAM uses a heavy metal and a perpendicular Magnetic Tunnel Junction (MTJ) consisting of two ferromagnetic layers separated by a thin insulator (MgO), as shown in Figure 6. A reference layer has a fixed magnetic direction, while the magnetic direction of the free layer can be switched by an in-plane current flowing through the heavy metal. When two layers have parallel magnetic direction, the MTJ has low resistance state (LRS) and indicates "0". In contrast, if two layers are in anti-parallel direction, the MTJ has high resistance state (HRS) and represents "1". To write a cell, a write word-line (WWL) is first activated. When the write bit-line (WBL) voltage is larger than the source line (SL) voltage by a threshold, "1" is written to the cell. On the contrary, if the WBL voltage is smaller than the SL voltage by a threshold, "0" is written to the cell. To read a cell, a read word-line (RWL) is activated, read voltage is applied on the read bit-line (RBL) and the SL is grounded.

2.6 Integration of NVM Technologies

Most emerging NVM technologies, e.g., SOT-MRAM [13], PCM [1], ReRAM [31], are generally CMOS-compatible, so they can be integrated with each other and CMOS logic in the same chip. For instance, a MTJ, i.e., the core of a SOT-MRAM cell, is successfully fabricated with ReRAM cells in a single chip [41]. Furthermore, the monolithic 3D stacking technology [28] can also integrate various NVM technologies including ReRAM and STT-MRAM into a 3D vertical memory array to offer complementary tradeoffs among high density, low latency, and long endurance.

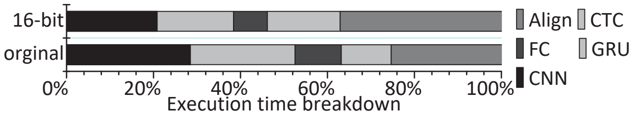


Figure 9: Execution time breakdown of Guppy.

3 MOTIVATION

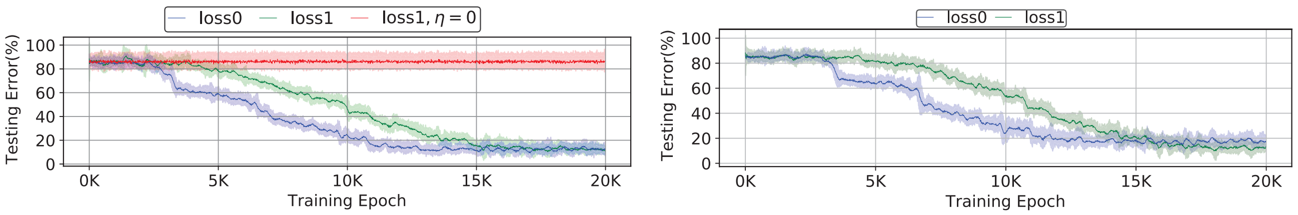
It is challenging to accelerate nanopore base-calling from both algorithm and architecture perspectives. If we naïvely accelerate a base-caller using prior network quantization techniques, the quantized base-caller greatly increases the number of systematic errors that cannot be corrected by read voting. State-of-the-art NVM-based PIMs suffer from huge power consumption and area overhead of CMOS ADCs, when executing a quantized base-caller. New bottlenecks, CTC decoding and read voting operations, emerge in a quantized base-caller, but no prior PIM supports these operations.

3.1 More Systematic Errors in a Quantized Base-caller

We applied the latest network quantization technique, FQN [18], on Guppy to improve its base-calling speed. As Figure 7 shows, the Conv, GRU, FC, and CTC layers of Guppy are quantized with various bit-widths from 4-bit to 32-bit. We executed the quantized Guppy on an NVIDIA Tesla T4 GPU. Although quantizing Guppy with a smaller bit-width, e.g., 4-bit, increases base-calling throughput by 2.75×, base-calling accuracy of the quantized Guppy after reads vote decreases by 4.3%, which dramatically jeopardizes the quality of final DNA mappings. The base-calling accuracy includes two parts: one is the *read accuracy* before reads vote; the other is the *vote accuracy* after reads vote. The base-calling accuracy after reads vote is more important, since read voting operations eliminate all random errors and leave only systematic errors. Even the 16-bit quantized Guppy suffers from significant systematic errors that cannot be corrected by read voting operations.

3.2 Large ADC Overhead in NVM-based Dot-product Engines

Although prior PIM designs process DNN inferences using ReRAM- [9, 31], PCM- [1], and STT-MRAM [39]-based dot-product engines, the power efficiency and scalability of these PIMs are limited by CMOS ADCs. The in-situ analog arithmetic computing fashion is the key for a NVM-based dot-product engine [1, 9, 31, 39] to substantially improving computing throughput of vector-matrix multiplications. However, as Figure 8 highlights, CMOS ADCs cost 82% $\sim$ 85% of power consumption and 87% $\sim$ 91% of area overhead in a ReRAM- [31], PCM- [1] and STT-MRAM [39]-based dot-product engine. Although ReRAM, PCM and STT-MRAM has the cell size of $4F^2$, $4F^2$, $60F^2$, respectively, the power and area of array in various NVM dot-product engines are similar, since peripheral

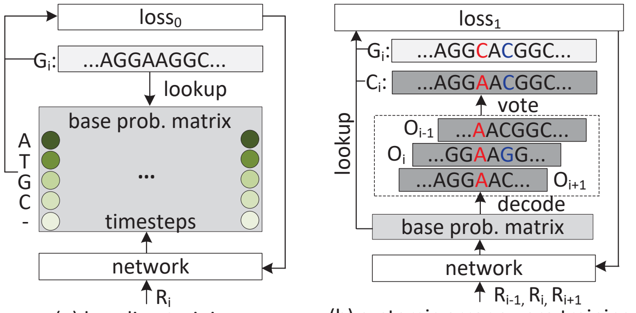


(a) Full-precision model training

(b) 8-bit quantized model training

Figure 10: The training of full-precision and quantized base-callers with different loss functions.

circuits including row decoders, column multiplexers and sense amplifiers dominate power consumption and area overhead of a dot-product engine. As a result, CMOS ADCs cost 58% of power consumption and 30% of chip area in a typical NVM-based PIM design [31]. The power density of recent NVM-based PIMs has already exceeded the memory thermal tolerance even with active heat sinks. Particularly, a $416W$ ReRAM-based PIM [9] has the power density of $842mW/mm^2$, much larger than the thermal tolerance of a ReRAM chip with active heat sinks [42]. CMOS ADCs seriously limit the scalability and power-efficiency of state-of-the-art NVM-based PIM accelerators.



(a) baseline training (b) systemic error aware training

Figure 11: Systematic error aware training.

3.3 New Bottlenecks in a Quantized Base-caller

Besides more systematic errors, new performance bottlenecks emerge in a 16-bit quantized Guppy. As Figure 9 shows, CTC decoding operations consume 16.7% of base-calling latency, while read voting operations cost 37% of base-calling latency in the 16-bit quantized Guppy. The Conv, GRU and FC layers in the quantized Guppy heavily rely on 16-bit fixed-point vector-matrix multiplications that can be efficiently executed by a state-of-the-art GPU. Therefore, we anticipate these Conv, GRU and FC layers can be completed by a NVM-based PIM with a shorter latency. In contrast, CTC decoding and read voting operations of a base-caller are not fully optimized on the GPU. Moreover, no prior PIM design supports CTC decoding or read voting.

4 HELIX

4.1 Systematic Error Aware Training

To reduce the systematic errors that cannot be corrected by read votes, we propose *Systematic Error Aware Training* (SEAT) that aims to minimize the edit distance between a consensus read and its ground truth DNA sequence by a novel loss function during the training of a quantized base-caller.

Baseline training. During the training of a base-caller [7, 36], the gradient is not computed through the edit distance between the predicted DNA sequence and its corresponding ground truth, since the computation of edit distance is non-differentiable. As Figure 11a shows, the Conv, GRU and FC layers generate the base probability matrix by an input signal vector R_i. Instead of edit distances, the CTC decoder [7, 33] computes the probability of the ground truth read G_i, $p(G_i|R_i)$, as the loss function by applying Equation 2 on the base probability matrix. For a training set $\mathbb{D}$, the weights of the base-caller are tuned to minimize:

$$loss_0 = \sum_{(G_i,R_i)\in\mathbb{D}} (-\ln p(G_i|R_i)) \tag{3}$$

where the more similar to G_i the predicted read is, the smaller $-\ln(p(G_i|R_i))$ is. By making each predicted read more similar to the ground truth, state-of-the-art base-callers indirectly minimizes the number of random and systematic errors. However, random errors can be corrected by read voting operations, whereas only systematic errors are the "real" errors that degrade the quality of final DNA mappings.

Systematic-error-aware training. The number of systematic errors significantly increases in a quantized base-caller. We created SEAT for the quantized base-caller to minimize the number of systematic errors. SEAT is shown in Figure 11b. The base-caller uses multiple input signal data vectors, i.e., R_{i-1}, R_i, and R_{i+1}, to generate multiple predicted reads, i.e., O_{i-1}, O_i, and O_{i+1}, that vote to create a consensus read C_i. Instead of minimizing the edit distance between C_i and the ground truth read G_i, we build a new loss function to make C_i more similar to G_i. For a training set $\mathbb{D}$, the parameters of the base-caller are tuned to minimize:

$$\begin{aligned} loss_1 = \sum_{(G_i,R_i)\in\mathbb{D}} & [-\eta \cdot \ln p(G_i|R_i)+ \\ & (\ln p(G_i|R_i) - \ln p(C_i|R_i))^2] \end{aligned} \tag{4}$$

where $-\ln p(G_i|R_i)$ makes each predicted read more similar to G_i; $(\ln p(G_i|R_i) - \ln p(C_i|R_i))^2$ minimizes the probability difference between the consensus read C_i voted by multiple predicted reads and G_i; and $\eta \in [0,1]$ is a floating-point constant regulating the impact of $-\ln p(G_i|R_i)$.

The effect of SEAT. As Figure 10(a) shows, we trained a full-precision Guppy by Equation 3 ($loss_0$) and Equation 4 ($loss_1$). If we set η in $loss_1$ to 0, the training cannot converge, since it has no motivation to improve the accuracy of each read. When we set η to 1, compared to $loss_0$, $loss_1$ slows down training convergence. When the read error rate is high, it is faster to improve the quality of each read independently. However, two loss functions achieve

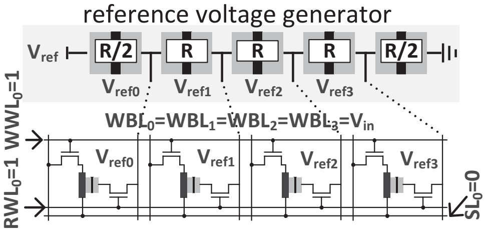
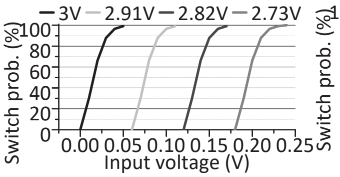


Figure 12: The ADC SOT-MRAM array.



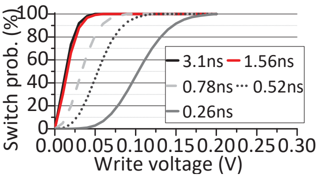


Figure 13: Input voltage vs. RBL voltage.

Figure 14: Write voltage vs. pulse duration.

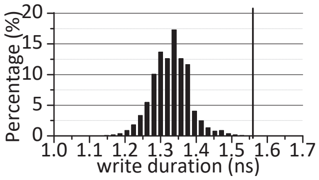
similar base-calling accuracy at the end of the training of Guppy. Full-precision Guppy is powerful enough to minimize the number of systematic errors even without read voting operations. In contrast, the training of 8-bit quantized Guppy with $loss_0$ and $loss_1$ is shown in Figure 10(b). For the 8-bit quantized Guppy, compared to $loss_0$, $loss_1$ increases base-calling accuracy by 6% and obtains the same base-calling accuracy as the full precision model. After the systematic error reduction capability of Guppy is damaged by network quantization, $loss_1$ can reduce the systematic errors for the quantized Guppy.

4.2 ADC-free PIM Accelerator

To reduce area overhead and power consumption of CMOS ADCs in prior NVM-based PIM accelerators, we propose a SOT-MRAM-based ADC array to reliably process analog-to-digital conversions.

ADC array. An example of a 2-bit ADC array is shown in Figure 12. To distinguish 2 bits, an ADC array produces four reference voltages ($[V_{ref0} - V_{ref3}] = [3V, 2.91V, 2.82V, 2.73V]$) by a MTJ-based reference voltage generator. In the ADC array, all write word-lines (WWLs) and read word-lines (RWLs) are set to 1, and source lines (SLs) are set to 0. Input voltages are applied to write bit-lines (WBLs), and reference voltages are assigned to read bit-lines (RBLs). As Figure 13 highlights, due to the spin hall effect and voltage-controlled magnetic anisotropy [17], the write voltages of SOT-MRAM are different under various RBL voltages. When a larger voltage is applied on the RBL, the SOT-MRAM write voltage reduces significantly. There are four cases, i.e., 1000, 1100, 1110 and 1111, when an input voltage writes four cells in the ADC Array. By a small encoder, these four cases are encoded to 0, 1, 2 and 3. In this way, the input voltage is converted to a 2-bit digital value. Although a recent work [4] leverages the MTJ stochasticity to build an 8-bit ADC by MTJ, the design relies on CMOS counters and registers that introduce large power consumption and area overhead.

Resolution and frequency. We need to precisely control write pulses in order to enable a higher resolution for the ADC array. There is a trade-off between the resolution and frequency of an ADC array. Figure 14 shows the switching probability of a SOT-MRAM cell under different voltages and pulse durations. The shorter the pulse duration is, the higher frequency an ADC array can be operated at. With a shorter write duration, a higher write voltage is required to reliably switch a cell. Under a fixed maximum input voltage, e.g., $3V$, we can distinguish fewer levels of the input voltage (fewer bits) in Figure 13. For a higher resolution under $3V$, a smaller write voltage is preferred. In this case, we have to use a longer write pulse duration resulting in lower ADC frequency. To balance the trade-off, we use a $1.56ns$ write pulse to switch a SOT-MRAM cell with $0.05V$. In this way, 32 levels of the input voltage, i.e., 5-bit, can be distinguished. The ADC array can be operated at $640MHz$.



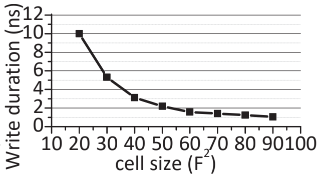


Figure 15: Write duration with $60F^2$ cell size.

Figure 16: Worst case duration with varying cell sizes.

Table 1: Process variation of SOT-MRAM

Parameter	μ	σ
WR/RD transistor width (W_{wt})	$384nm$	10%
WR/RD transistor length (L_{wt})	$192nm$	10%
Threshold voltage (V_{th})	$0.2V$	10%
MTJ resistance area product ($R \cdot A$)	$25\Omega \cdot \mu m^2$	8%
Cross section area of MTJ (A)	$64nm \times 128nm$	5%
Magnetization stability (Δ)	22	27%

Reliability. SOT-MRAM has no endurance issue, since on average a cell tolerates 10^{15} writes [16]. However, process variation makes a SOT-MRAM ADC array to generate wrong outputs. The relation between write current I and pulse duration t can be approximated as

$$t = \tau_0 e^{(1-\frac{I}{A \cdot J_{c0}})\Delta} \qquad (5)$$

where A is the cross sectional area of the MTJ free layer; J_{c0} is the critical current density at zero temperature; Δ is the magnetization stability energy height; and τ_0 is a fitting constant. Δ is decided by the MTJ volume. Due to process variation, different SOT-MRAM cells have different critical parameters including MTJ size, Δ, write transistor width, length and threshold voltage, thereby requiring different write pulse durations. We iteratively increase the write transistor size to guarantee that the worst case cell can be switched in $1.56ns$ by considering process variation. To model the process variation on SOT-MRAM, we adopted the parameters shown in Table 1 from [25]. In each iteration, we conducted 10 billion Monte-Carlo simulations with Cadence Spectre to generate a write duration distribution under a certain SOT-MRAM cell size, which is dominated by the write transistor size. At last, we show the relation between the worst case cell write duration and the cell size in Figure 16. We selected $60F^2$ to tolerate process variation and guarantee the worst case cell write duration is $1.56ns$.

Pipelined dot-product engine. SOT-MRAM ADC arrays can be easily integrated with prior NVM-based dot-product engines. As Figure 17 shows, the pipeline of a fixed-point vector-matrix multiplication includes fetching data, MAC, ADC, shift-&-add, and

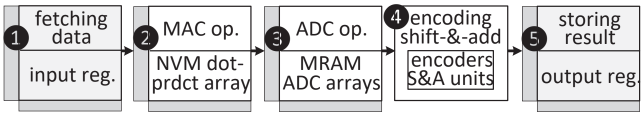


Figure 17: The pipeline of a NVM-based dot-product engine.

storing result. ❶ During the stage of fetching data, 128 1-bit fixed-point inputs are read from input registers. The 2-bit weights are stored in a 128×128 array of a NVM-based dot-product engine. ❷ A NVM-based dot-product engine converts 1-bit fixed-point inputs to analog voltages by DACs, and performs 1-bit×2-bit matrix-vector multiplications [31]. ❸ Multiple ADC arrays digitize a MAC result. The NVM-based dot-product engine generates 128 MAC results simultaneously. ❹ After encoding, digital values are sent to shift-&-add units to generate final dot-product results. ❺ At last, the final dot-product results are written into output registers.

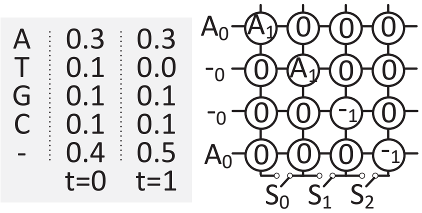


Figure 18: CTC decoding in a NVM dot-product engine.

4.3 CTC Decoding and Read Vote

CTC decoding. To process CTC beam searches, we rely on a NVM-based dot-product array. Figure 18 shows how to process a CTC beam search with width of 2. The top-2 largest probabilities of bases (i.e., A_1 and $-_1$) at the time step 1 ($t = 1$) in the CTC base probability matrix are written to the diagonal line cells of a NVM-based dot-product array. Since the search width is 2, each probability of a base at $t = 1$ is written twice in two different cells in the diagonal line of the dot-product array. All the other cells in the array are initialized to 0s. We can input the top-2 largest probabilities of bases (i.e., A_0 and $-_0$) at $t = 2$ to the corresponding WLs, so that $p(A_0A_1)$, $p(A_0-_1)$, $p(-_0A_1)$, and $p(-_0-_1)$ can be concurrently computed. To support the merges of probabilities of multiple-base sequences, we proposed to add a transistor to each BL to connect itself and its neighboring BL. By closing all transistors ($S_0 \sim S_2$), we merged the probabilities of four 2-base sequences. In this way, we have $p(A) = p(A_0A_1) + p(A_0-_1) + p(-_0A_1) + p(-_0-_1)$.

Reliability of NVM dot-product arrays. Since each BL has only one base's probability, the resistance of the transistor we add on each BL is too small to introduce errors in CTC decoding. Since a NVM dot-product array can operate at only 10MHz [31], the extra transistor does not slow down the dot-product array. However, our design increases writes to a NVM dot-product array. A ReRAM cell stands for 10^{11} writes. A recent ReRAM-based PIM [9] can reliably run back-propagation for 15.7 years. Compared to back-propagation, the Conv, GRU, FC layers and a CTC decoder of a base-caller have much less writes. Based on our estimation, the NVM dot-product arrays of Helix can reliably work for >20 years even when running Chiron having the most complex architecture and the largest number of parameters among all base-callers.

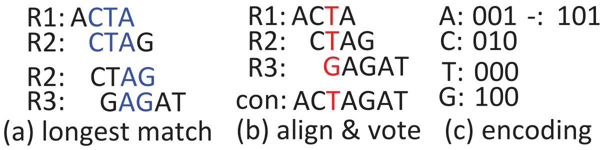


Figure 19: Read voting.

Read vote. After a base-caller generates multiple consecutively predicted reads, a read vote is required to produce a consensus read. A voting example is shown in Figure 19, where there are three reads, i.e., R_1="ACTA", R_2="CTAG", and R_3="GAGAT". A vote finds the longest matches between all reads (Figure 19a), aligns reads, and computes the consensus (Figure 19b). Finding the longest matches between all reads is the most important operation in a read vote. To find the longest match between R_1 and R_2, all of their sub-strings have to be compared. As Figure 19(c) describes, we encoded each DNA symbol by 3-bit. The string match problem is converted to comparing two binary vectors. We propose a SOT-MRAM-based binary comparator array to accelerate binary vector comparisons.

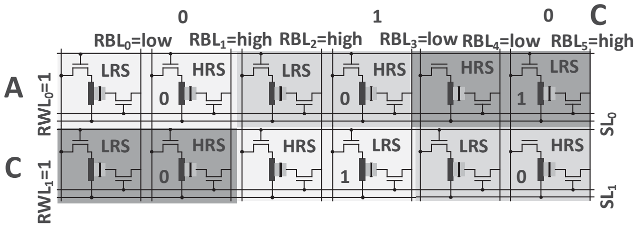


Figure 20: A binary comparator array.

Binary comparator array. We wrote all sub-strings of R_1, e.g. "ACTA" and "CTA", into a SOT-MRAM array shown in Figure 20. Each sub-string stays in a row of the array. For instance, "ACTA" is in the first row, while "CTA" is in the second row. We used a 2-cell pair in a row to record each bit in the encoding of a DNA symbol. 0 is represented by a low resistance state (LRS) cell and a high resistance state (HRS) cell, while 1 is indicated by a HRS cell and a LRS cell. Therefore, in Figure 20, 6 cells in the first row indicate the first "A" of "ACTA", while 6 cells in the second row represent the first "C" of "CTA". We applied the corresponding voltages representing a sub-string of R_2, e.g., "C", on the RBLs of the binary comparator array. Each bit in the encoding of "C" (010) is represented by two voltages applied on the two RBLs of a 2-cell pair respectively, i.e., 0 is represented by low and high voltages, while 1 is denoted by high and low voltages. If two DNA symbols are the same, there is no current accumulated on the SL, e.g., SL_1. The sense amplifier can sense a current on the SL, e.g., SL_0, if two DNA symbols are different. Unlike alignment and assembly, aligning reads during read voting is easy [33], because the order of these reads is already known and the length of each read is only 10 ∼ 30 bases.

Reliability of binary comparator arrays. To compare two 30-base reads, a binary comparator array requires > 180 cells on a RWL. We used the $60F^2$ cell size to build 256 × 256 arrays as binary comparators to study process variation. We also adopted the same process variation parameters in Table 1. We performed 10 billion Monte-Carlo simulations to profile the error rate with random 30-base read inputs. The error rate for reading a single

Table 2: The area and power of Helix

Component	Params	Spec	Power (mW)	Area (mm^2)
eDRAM	bank num	4	20.7	0.083
Buffer	capacity	64KB		
Bus	wire num	384	7	0.09
Router	flit size	32	10.5	0.0378
Activation	number	2	0.52	0.0006
S+A	number	1	0.05	0.00006
MaxPool	number	1	0.4	0.0024
OR	size	3KB	1.68	0.0032
Total			40.9	0.215
NVM	number	8		
Array	size	128×128	2.4	0.0002
	bits/cell	2		
S+H	number	8×128	0.001	0.00004
S+A	number	4	0.2	0.00024
IR	size	2KB	1.24	0.0021
OR	size	256B	0.23	0.00077
DAC	resolution	1 bit	4	0.00017
	number	8×128		
	resolution	8 bits		
ADC	frequency	1.28 GSps	16	0.0096
	number	8		
ISAAC Total	number	12	289	0.157
ISAAC Tile Total			330	0.372
ISAAC Total	tile num	168	**55.4W**	**62.5**
SOT-MRAM	size	32×32		
ADC array	frequency	640MHz	0.6	0.00005
	number	8×4		
voltage ref	number	1	0.02	0.00003
encoder	number	8×4	0.001	0.000002
Helix Total	number	12	122	0.0439
Helix Tile Total			163	0.259
SOT-MRAM	size	256×256	1.3W	0.11
binary cmp	number	1024		
Helix Total	tile num	168	**25.7W**	**43.83**

cell is low, i.e., 10^{-11}. After comparing 556 million 30-base reads, on average, our binary comparator array makes 1 mistakes. We believe this error rate is acceptable for Helix, since assembly, read mapping, and polishing in the nanopore sequencing pipeline may correct systematic errors.

4.4 Design Overhead

For the algorithm modification, our systematic error aware training increased the training time of quantized base-callers by 32% ∼ 52% (∼ 2 days). For the NVM PIM design, we developed Helix based on a well-known ReRAM PIM ISAAC [31], because we showed that PCM-, STT-, and ReRAM-based dot-product engines have similar power consumption and area overhead (Figure 8). Although recent search efforts on NVM PIMs propose compilation support [9], data flow optimization [2], and sparsity reduction [40], all their architectures are built upon ISAAC [31]. To estimate the hardware overhead of Helix, we modeled the leakage power, dynamic energy, latency and area of Helix by NVSim [5] with 32nm process technology. The power consumption and area overhead of Helix is described in Table 2. The NVM dot-product pipeline is operated at 10MHz [31]. 8-bit [31], 6-bit [40], and 5-bit [9] ADCs are adopted by prior PIMs. Although we selected 8-bit ADCs in our baseline, we perform a sensitivity study on the ADC resolution in §6. To support CTC decoding, we add a transistor to each BL of a NVM-based dot-product engine introducing insignificant power and area overhead. To accelerate read votes, we also integrated 1K 256×256 SOT-MRAM arrays that cost only 1.3W power and occupy 0.11mm^2.

Table 3: The architecture of various base-callers

		Scrappie	Chiron	Guppy
Input		300 × 1		
Conv	layer #	1	3	1
	filter size	11 × 1	1 × 1/3	11 × 1
	filter #	96	256	96
	strides	5	1	2
	output	60 × 96	60 × 256	150 × 96
	MAC #	0.063M	570M	0.2736M
	Param #	1056	1.9M	0.0018M
RNN	type	GRU	LSTM	GRU
	layer #	5	6	5
	filter	96	100	256
	output	60 × 1025	300 × 100	150 × 40
	MAC #	8.1M	45M	36M
	Param #	0.14M	0.15M	0.23M
FC	layer #	1	1	1
	filter	1025 × 5	100 × 5	40 × 5
	output	60 × 5	300 × 5	60 × 5
	MAC #	0.31M	0.15M	0.012M
	Param #	0.31M	0.15M	0.012M
CTC		output 60 × 1 and then merge		
Align		align multiple reads		
Total MAC #		8.47M	615.2M	36.3M
Total Param #		0.45M	2.2M	0.244M

5 EXPERIMENTAL METHODOLOGY

5.1 Simulation and Evaluation

We adopted a NVM dot-product engine simulator from [40] and modified it to cycle-accurately study the performance, power and energy consumption of Helix and our baseline NVM-based PIM accelerator. According to a user-defined accelerator configuration and a DNN topology description, the simulator generates the performance and power details of the accelerator inferring the DNN. We integrated the ADC array and binary comparator arrays of Helix into the pipeline and data flow of the simulator. We implemented our systematic error aware training in base-callers [29, 33, 36] that are trained on either an NVIDIA Tesla T4 GPU or an Intel Xeon E5-4655 v4 CPU.

5.2 Base-callers and Datasets

Base-callers. Oxford nanopore technology had updated its pore type to R9.4. Among all base-callers, only Metrichor [27], Albacore [26], Flappie [7], Scrappie [29], Guppy [36], and Chiron [33] can base-call R9.4 reads. Metrichor is a cloud-based base-caller whose details are unknown, while Albacore is deprecated by Oxford nanopore technology. Albacore has been replaced by its GPU-version successor Guppy and CPU-version successor Flappie. Guppy and Flappie share the same DNN topology. In this paper, we include three base-callers: Guppy, Scrappie, and Chiron. Guppy and Chiron are GPU-based base-callers, while Scrappie can be executed on only a CPU. We redesigned Scrappie using TensorFlow, so that it can also be processed by a GPU. The base-caller architectures can be viewed in Table 3. All base-callers share a similar network architecture including convolutional, recurrent neural network (RNN), and fully-connected layers. The RNN can be a GRU or Long Short Term Memory (LSTM) layer. Chiron has the most complex DNN topology. Particularly, its convolutional layers have the largest number of weights, while its RNN is a LSTM layer having more recurrent gates. We assume the beam search width of the CTC decoder in each base-caller is 10.

Table 4: The dataset for various base-callers.

Sample	# of reads	Median read length
Phage Lambda	34,383	5,720 bases
E.coli	15,012	5,836 bases
M.tuberculosis	147,594	3,423 bases
Human	10,000	6,154 bases

Datasets. We used R9.4 training datasets [32] including *E. coli*, *Phage Lambda*, *M. tuberculosis* and *human* to train base-callers. The input signal is normalized by subtracting the mean of the entire read and dividing by the standard deviation. At the beginning of each training epoch, the dataset was shuffled first and then fed into the base-caller by batch. Training with this mixed dataset enabled each base-caller to have better performance both on generality and base-calling accuracy. The datasets for the evaluation of various base-callers are summarized in Table 4.

Table 5: The comparison between CPU, GPU and Helix.

Parameter	CPU	GPU	Helix
core #	8	2560	16128
Frequency	3.2GHz	1.5GHz	10MHz
Area	$450mm^2$	$515mm^2$	$43.83mm^2$
TPD	135W	70W	25.7W
Cache	30MB L3	6MB L2	-
Memory	32GB DDR4	16GB GDDR6	32GB NVDIMM

5.3 Schemes

We compared our Helix PIM against the state-of-the-art CPU, GPU and NVM PIM baselines summarized as:

- CPU. Our CPU baseline is a 3.2GHz Intel Xeon E5-4655 v4 CPU, which has 8 cores and 30MB last level cache. More details can be viewed in Table 5.
- GPU. We selected NVIDIA Tesla T4 GPU as our GPU baseline, since it can support INT8 and INT4 MAC operations. A 1.5GHz NVIDIA Tesla T4 GPU has 2560 cudaCores and a 16GB GDDR6 main memory.
- ISAAC. We also chose ISAAC [31] as our NVM PIM baseline. We assumed ISAAC has the same processing throughput of CTC decoding and read vote without introducing extra power consumption and area overhead. By studying the sensitivity of the ADC resolution, we compared Helix against two successors of ISAAC including IMP [9] and SRE [40].
- 16-bit. We quantized base-callers with 16-bit and without systematic error aware training (SEAT) to achieve no obvious accuracy degradation. The quantized base-callers are ran on ISAAC.
- SEAT. We quantized base-callers with 5-bit and SEAT to guarantee no accuracy loss. The quantized base-callers are ran on ISAAC.
- ADC. We replaced CMOS ADCs of SEAT by our proposed ADC arrays.
- CTC. We used NVM-based dot-product arrays to process CTC decoding operations for ADC.
- Helix. We used SOT-MRAM-based binary comparator arrays to accelerate read votes for CTC. All techniques we proposed in this paper are accumulated in this scheme.

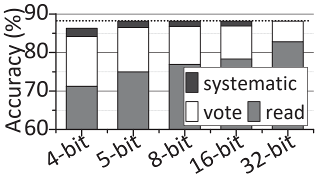


Figure 21: SEAT on Guppy.

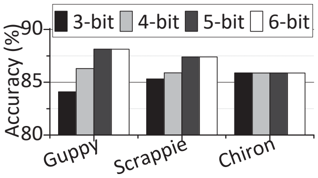


Figure 22: Quant. w. SEAT.

6 EVALUATION AND ANALYSIS

6.1 Systematic Error Aware Training

SEAT & quantization. Though naïvely applying the quantization scheme FQN [18] on base-callers improves base-calling throughput, the number of systematic errors that cannot be corrected by read votes greatly increases. After we trained Guppy with our systematic error aware training (SEAT), we can reduce the number of systematic errors. As Figure 21 shows, SEAT makes the quantized Guppy have no accuracy loss by reducing the number of systematic errors in its loss function, if it is quantized with ≥ 5-bit. In contrast, without SEAT, the 16-bit quantized Guppy starts to suffer from a significant number of systematic errors. In this way, SEAT enables more aggressive quantization with smaller bit-widths. We show base-calling accuracy of various quantized base-callers in Figure 22. We find that with 5-bit, no quantized base-caller suffers from accuracy degradation. However, with smaller bit-widths, e.g., 4-bit, Scrappie and Guppy suffer from obvious accuracy degradation, since they have compact architectures and less parameters. The parameter-rich Chiron does not decrease its base-calling accuracy, even when quantized with 3-bit.

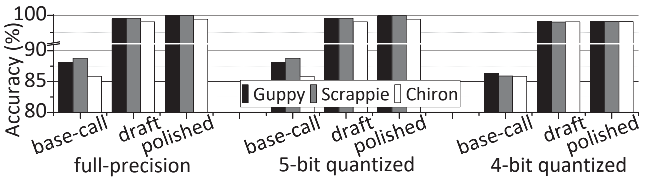
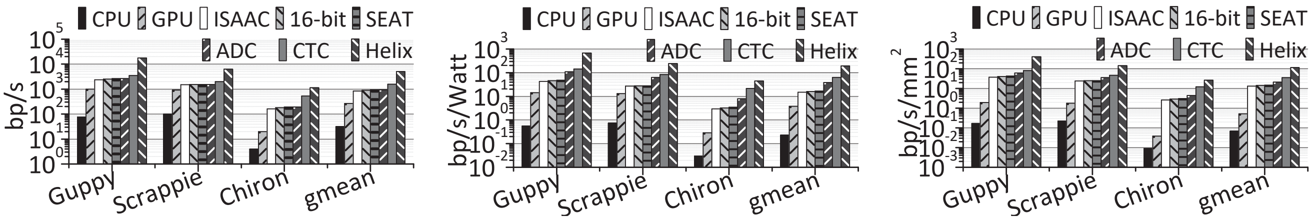


Figure 23: The comparison of base-callers with SEAT.



(a) Performance (b) Performance/Watt (c) Performance/mm^2

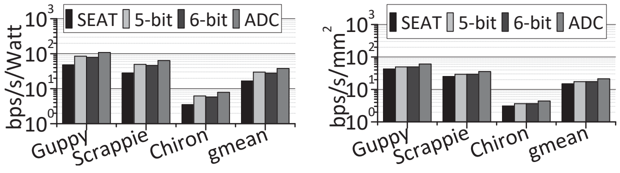
Figure 24: The performance, power and area comparison between various accelerators.

Quality of final genome mappings. We fed base-called DNA reads generated by quantized base-callers with SEAT into the nanopore sequencing pipeline to evaluate the quality of final DNA mappings. The accuracy comparison of various DNA mappings generated by both the full-precision, 4-bit, and 5-bit quantized base-callers with SEAT is shown in Figure 23, where "base-call" indicates the accuracy of reads generated by base-callers; "draft" represents the accuracy of alignment produced by read mapping; and "polished" means the accuracy of final read mapping after the polishing step. Compared to full-precision base-callers, the accuracy of reads, corresponding draft alignment, and final mapping generated by 5-bit quantized base-callers with SEAT has no accuracy loss. However, if we quantize the base-callers with 4-bit, the accuracy of base-called reads, their alignment and final mapping significantly degrades even with SEAT. Particularly, the 4-bit quantized `Scrappie` reduces the accuracy of the final mapping by 6%. Low quality genome mappings substantially increased the probability of misdiagnosis and false negative testings. Therefore, we used 5-bit to quantize these base-callers with SEAT.

Performance, power and area. The performance, power and area comparison between our CPU, GPU, and NVM-based PIM baselines is shown in Figure 24. Besides the `CPU` and `GPU`, we ran the DNN part of full-precision base-callers with 32-bit weights on our PIM baseline `ISAAC`, but left the other parts of base-callers including CTC decoding and aligning on the GPU without introducing extra power consumption and area overhead. As Figure 24(a) shows, on average, `ISAAC` greatly improves base-calling throughput by 25× and 2.15× over the CPU and GPU, respectively. Among all base-callers, Chiron achieves the largest speedup by running its DNN part on `ISAAC`, since 95% of the base-calling time is consumed by its DNN part. `ISAAC` improves base-calling throughput of Chiron by 7.16× over `GPU`. `ISAAC` also increases base-calling throughput per Watt and per mm^2 by 127% and 25× over `GPU` respectively, as shown in Figure 24(b) and 24(c). If we quantize base-callers with 16-bit, `16-bit` improves base-calling speed by 6.25% over `ISAAC`. On the contrary, if we use SEAT to aggressively quantize base-callers with 5-bit, `SEAT` improves base-calling speed by 11.1% over `ISAAC` without accuracy loss. Although the base-calling throughput improvement achieved by SEAT is not dramatically significant, SEAT is the key to enabling our power-efficient SOT-MRAM ADC arrays with lower resolution.

6.2 ADC-free PIM Accelerator

Performance per Watt and per mm^2. Because of SEAT, base-callers can be quantized with 5-bit without accuracy loss. In this way, we can use our SOT-MRAM-based ADC arrays with lower resolution to reduce power consumption and area overhead of our PIM accelerator. After we replace the CMOS ADCs in `SEAT` by our SOT-MRAM-based ADC arrays (`ADC`), the PIM accelerator running 5-bit quantized base-callers can still achieve the same performance as `SEAT`, as shown in Figure 24(a). However, `ADC` significantly reduces power consumption and area overhead of the PIM accelerator. As Figure 24(b) shows, on average, `ADC` improves base-calling throughput per Watt by 127% over `SEAT`. Moreover, `ADC` increases base-calling throughput per mm^2 by 42.9%, as shown in Figure 24(c).



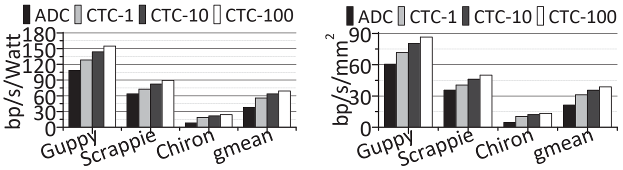
(a) Performance/Watt (b) Performance/mm^2

Figure 25: The comparison against various CMOS ADCs.

Comparison against ADCs with lower resolution. Recent works rely on CMOS ADCs with lower resolutions, e.g., 5-bit [9] and 6-bit [40], to reduce power consumption and area overhead of NVM-based dot-product engines. The lower resolution a CMOS ADC achieves, the smaller power consumption and area overhead it costs. We showed the comparison of performance per Watt and per mm^2 between NVM-based dot-product engines with our ADC arrays and with low-resolution CMOS ADCs in Figure 25. As Figure 25(a) shows, on average, our ADC arrays improve base-calling throughput per Watt by 27.9% and 37.3% over 5-bit and 6-bit CMOS ADCs respectively. Furthermore, on average, our ADC arrays increase base-calling throughput per mm^2 by 21.8% and 21.3% over 5-bit and 6-bit CMOS ADCs respectively, as shown in Figure 25(b). This is because a 5-bit CMOS ADC has similar area overhead to that of a 6-bit CMOS ADC.

6.3 CTC Decoding and Read Vote

CTC decoding. After we processed CTC decoding operations by NVM-based dot-product engines, as Figure 24(a) show, on average, `CTC` improves base-calling throughput by 67.8% over `ADC`. Particularly, `CTC` boosts base-calling throughput of Chiron to 2.74×. Moreover, `CTC` also reduces the data transfers between the GPU and our PIM accelerator. In `CTC`, CTC decoding operations and DNN inferences share the same NVM-based dot-product engines, so `CTC` does not increase power consumption or area overhead. As a result, `CTC` improves base-calling throughput per Watt and per mm^2 by 64% and 69% over `ADC` respectively, as shown in Figure 24(b) and 24(c).



(a) Performance/Watt (b) Performance/mm^2

Figure 26: The comparison w. varying beam search widths.

Sensitivity to beam search width. Figure 26 exhibits the sensitivity of base-calling throughput of CTC with varying beam search widths. With an enlarging width of beam search in the CTC decoder, CTC achieves larger improvement on base-calling throughput per Watt and per mm^2. This is because, with a larger width of beam search in the CTC decoder, the execution time of CTC decoding operations becomes more and more significant. A NVM-based dot-product engine requires more iterations to process a CTC decoding operation with larger beam search width.

Read voting. By enabling SOT-MRAM-based binary comparator arrays to process read votes, we have all proposed techniques for Helix. On average, Helix improves base-calling throughput by 2.22× over CTC, as shown in Figure 24(a). Helix can concurrently compare up to 256 reads by only one binary comparator array during each read voting without introducing significant power consumption or area overhead. As Figure 24(b) and 24(c) show, Helix boosts base-calling throughput per Watt and per mm^2 to 3.06× and 3.22× over CTC, respectively. Overall, on average, Helix achieves 6× base-calling throughput of ISAAC.

7 RELATED WORK

Nanopore sequencing. Nanopore sequencing [15] emerges as one of the most promising genome sequencing technologies to enabling personalized medicine, global food security, and virus surveillance, because its capability of generating long reads and good real-time mobility. In a nanopore sequencing pipeline, the step of base-calling costs 44.5% of total execution time, because of high computing overhead of state-of-the-art DNN-based base-callers. It takes more than one day for a server-level GPU to base-call a 3G-bp human genome with a 30× coverage by a DNN-based base-caller. This is unacceptably slow particularly during virus outbreaks.

Network quantization. Although prior works propose network quantization [18, 19, 38] to approximate floating-point network parameters by fixed-point representations with lower bit-widths, naïvely applying prior network quantization on base-callers greatly increased the number of systematic errors that cannot be corrected by read votes, thereby substantially degrading the quality of final genome mappings.

NVM dot-product engines. Although ReRAM- [9, 31, 40], PCM- [1] , and STT-MRAM [39]-based dot-product engines are proposed in order to accelerate DNN inferences, their power efficiency and scalability are limited by power-hungry CMOS ADCs. CMOS ADCs cost 58% of power consumption and 30% of chip area in a well-known ReRAM-based PIM [31]. Another recent ReRAM-based PIM [9] consumes $416W$ and has power density of $842mW/mm^2$, much larger than the thermal tolerance of a ReRAM chip with active heat sinks [42].

Hardware acceleration for genome sequencing. Hardware specialized acceleration is an effective way to overcome the big genomic data problem. However, most prior works focus on only accelerating genome alignment and assembly [34], particular short read alignment [8, 14, 21, 35, 37, 43]. However, long read alignment and assembly are not the most-time consuming steps in a nanopore sequencing pipeline.

8 CONCLUSION

In this paper, we proposed an algorithm/architecture co-designed PIM accelerator, Helix, to process nanopore base-calling. We presented systematic error aware training to decrease the bit-width of a quantized base-caller without increasing the number of systematic errors that cannot be corrected through read voting operations. We also create a SOT-MRAM ADC array to accelerate analog-to-digital conversion operations. Finally, we revised a traditional NVM-based dot-product engine to accelerate CTC decoding operations, and then introduced a SOT-MRAM binary comparator array to process read voting operations at the end of base-calling. Compared to state-of-the-art PIM accelerators, Helix improves base-calling throughput by 6×, throughput per Watt by 11.9×, and per mm^2 by 7.5× without degrading base-calling accuracy.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments and constructive suggestions. This work was supported in part by NSF CCF-1909509 and CCF-1908992.

REFERENCES

[1] S. Ambrogio, M. Gallot, et al. 2019. Reducing the Impact of Phase-Change Memory Conductance Drift on the Inference of large-scale Hardware Neural Networks. In *IEEE International Electron Devices Meeting*. 6.1.1–6.1.4.

[2] Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R Stanley Williams, Paolo Faraboschi, Wen-mei W Hwu, John Paul Strachan, Kaushik Roy, et al. 2019. PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 715–731.

[3] Vladimír Boža, Broňa Brejová, and Tomáš Vinař. 2017. DeepNano: Deep recurrent neural networks for base calling in MinION nanopore reads. *PloS one* 12, 6 (2017), e0178751.

[4] I. Chakraborty, A. Agrawal, and K. Roy. 2018. Design of a Low-Voltage Analog-to-Digital Converter Using Voltage-Controlled Stochastic Switching of Low Barrier Nanomagnets. *IEEE Magnetics Letters* 9 (2018), 1–5.

[5] Xiangyu Dong, Cong Xu, Yuan Xie, and Norman P Jouppi. 2012. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 7 (2012), 994–1007.

[6] Nuno Rodrigues Faria, Ester C Sabino, Marcio RT Nunes, Luiz Carlos Junior Alcantara, Nicholas J Loman, and Oliver G Pybus. 2016. Mobile real-time surveillance of Zika virus in Brazil. *Genome medicine* 8, 1 (2016), 97.

[7] Flappie. 2019. Oxford Nanopore Technologies. https://github.com/nanoporetech/flappie

[8] Daichi Fuijiki, Arun Subramaniyan, Tianjun Zhang, Yu Zheng, Reetuparna Das, David Blaauw, and Satish Narayanasamy. 2018. GenAx: A Genome Sequencing Accelerator. In *IEEE/ACM International Symposium on Computer Architecture*.

[9] Daichi Fujiki, Scott Mahlke, and Reetuparna Das. 2018. In-Memory Data Parallel Processor. In *IEEE/ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 1–14.

[10] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ACM International Conference on Machine Learning*. 369–376.

[11] Awni Hannun. 2017. Sequence Modeling with CTC. *Distill* (2017). https://doi.org/10.23915/distill.00008 https://distill.pub/2017/ctc.

[12] Thomas Hoenen, Allison Groseth, Kyle Rosenke, Robert J Fischer, Andreas Hoenen, Seth D Judson, Cynthia Martellaro, Darryl Falzarano, Andrea Marzi, and R Burke Squires. 2016. Nanopore sequencing as a rapidly deployable Ebola outbreak tool. *Emerging infectious diseases* 22, 2 (2016), 331.

[13] H. Honjo, T. V. A. Nguyen, et al. 2019. First demonstration of field-free SOT-MRAM with 0.35 ns write speed and 70 thermal stability under 400°C thermal tolerance by canted SOT structure and its advanced patterning/SOT channel technology. In *2019 IEEE International Electron Devices Meeting*. 28.5.1–28.5.4.

[14] Wenqin Huangfu, Xueqi Li, Shuangchen Li, Xing Hu, Peng Gu, and Yuan Xie. 2019. MEDAL: Scalable DIMM Based Near Data Processing Accelerator for DNA Seeding Algorithm. In *IEEE/ACM International Symposium on Microarchitecture*. 587–599.

[15] Miten Jain, Sergey Koren, Karen H Miga, Josh Quick, Arthur C Rand, Thomas A Sasani, John R Tyson, Andrew D Beggs, Alexander T Dilthey, and Ian T Fiddes. 2018. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature biotechnology* 36, 4 (2018), 338.

[16] Jimmy J Kan, Chando Park, Chi Ching, Jaesoo Ahn, Yuan Xie, Mahendra Pakala, and Seung H Kang. 2017. A study on practically unlimited endurance of STT-MRAM. *IEEE Transactions on Electron Devices* 64, 9 (2017), 3639–3646.

[17] H. Lee, F. Ebrahimi, P. K. Amiri, and K. L. Wang. 2016. Low-Power, High-Density Spintronic Programmable Logic With Voltage-Gated Spin Hall Effect in Magnetic Tunnel Junctions. *IEEE Magnetics Letters* (2016).

[18] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. 2019. Fully Quantized Network for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2810–2819.

[19] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. 2016. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*.

[20] Roujian Lu, Xiang Zhao, Juan Li, Peihua Niu, Bo Yang, Honglong Wu, Wenling Wang, Hao Song, Baoying Huang, Na Zhu, et al. 2020. Genomic characterisation and epidemiology of 2019 novel coronavirus: implications for virus origins and receptor binding. *The Lancet* 395, 10224 (2020), 565–574.

[21] Advait Madhavan, Timothy Sherwood, and Dmitri Strukov. 2014. Race Logic: A hardware acceleration for dynamic programming algorithms. In *IEEE/ACM International Symposium on Computer Architecture*.

[22] Metrichor. 2017. Oxford Nanopore Technologies. https://metrichor.com

[23] Kazuma Nakano, Akino Shiroma, Makiko Shimoji, Hinako Tamotsu, Noriko Ashimine, Shun Ohki, Misuzu Shinzato, Maiko Minami, Tetsuhiro Nakanishi, and Kuniko Teruya. 2017. Advantages of genome sequencing by long-read sequencer using SMRT technology in medical area. *Human cell* 30, 3 (2017), 149–161.

[24] Nanopore. 2020. SmidgION Nanopore Sequencer. https://nanoporetech.com/products/smidgion.

[25] Janusz J Nowak, Ray P Robertazzi, Jonathan Z Sun, Guohan Hu, Jeong-Heon Park, JungHyuk Lee, Anthony J Annunziata, Gen P Lauer, Raman Kothandaraman, Eugene J O'Sullivan, et al. 2016. Dependence of voltage and size on write error rates in spin-transfer torque magnetic random-access memory. *IEEE Magnetics Letters* 7 (2016), 1–4.

[26] Oxford. 2018. Albacore. https://nanoporetech.com/about-us/news/new-basecaller-now-performs-raw-basecalling-improved-sequencing-accuracy.

[27] Oxford. 2018. Metrichor. https://nanoporetech.com/products/metrichor.

[28] M. M. Sabry Aly, T. F. Wu, A. Bartolo, Y. H. Malviya, W. Hwang, G. Hills, I. Markov, M. Wootters, M. M. Shulaker, H. . Philip Wong, and S. Mitra. 2019. The N3XT Approach to Energy-Efficient Abundant-Data Computing. *Proc. IEEE* 107, 1 (2019), 19–48.

[29] Scrappie. 2019. Oxford Nanopore Technologies. https://github.com/nanoporetech/scrappie

[30] Damla Senol Cali, Jeremie S Kim, Saugata Ghose, Can Alkan, and Onur Mutlu. 2018. Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions. *Briefings in bioinformatics* (04 2018).

[31] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. 2016. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. In *ACM/IEEE International Symposium on Computer Architecture*. 14–26.

[32] Haotian Teng. 2018. Chiron: A basecaller for Oxford Nanopore Technologies' sequencers. https://github.com/haotianteng/Chiron.

[33] Haotian Teng, Minh Duc Cao, Michael B Hall, Tania Duarte, Sheng Wang, and Lachlan JM Coin. 2018. Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning. *GigaScience* 7, 5 (2018).

[34] Yatish Turakhia, Gill Bejerano, and William J. Dally. 2018. Darwin: A Genomics Co-processor Provides Up to 15,000X Acceleration on Long Read Assembly. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.

[35] Y. Turakhia, S. D. Goenka, G. Bejerano, and W. J. Dally. 2019. Darwin-WGA: A Co-processor Provides Increased Sensitivity in Whole Genome Alignments with High Speedup. In *IEEE International Symposium on High Performance Computer Architecture*. 359–372.

[36] Ryan R. Wick, Louise M. Judd, and Kathryn E. Holt. 2019. Performance of neural network basecalling tools for Oxford Nanopore sequencing. *Genome Biology* 20, 1 (24 Jun 2019), 129.

[37] L. Wu, D. Bruns-Smith, F. A. Nothaft, Q. Huang, S. Karandikar, J. Le, A. Lin, H. Mao, B. Sweeney, K. Asanović, D. A. Patterson, and A. D. Joseph. 2019. FPGA Accelerated INDEL Realignment in the Cloud. In *IEEE International Symposium on High Performance Computer Architecture*. 277–290.

[38] Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. 2018. Alternating Multi-bit Quantization for Recurrent Neural Networks. In *International Conference on Learning Representations*.

[39] Hao Yan, Hebin R. Cherian, Ethan C. Ahn, and Lide Duan. 2018. CELIA: A Device and Architecture Co-Design Framework for STT-MRAM-Based Deep Learning Acceleration. In *ACM International Conference on Supercomputing*. 149–159.

[40] Tzu-Hsien Yang, Hsiang-Yun Cheng, Chia-Lin Yang, I-Ching Tseng, Han-Wen Hu, Hung-Sheng Chang, and Hsiang-Pang Li. 2019. Sparse ReRAM Engine: Joint Exploration of Activation and Weight Sparsity in Compressed Neural Networks. In *ACM/IEEE International Symposium on Computer Architecture*. 236–249.

[41] Yu Zhang, Xiaoyang Lin, Jean-Paul Adam, Guillaume Agnus, Wang Kang, Wenlong Cai, Jean-Rene Coudevylle, Nathalie Isac, Jianlei Yang, Huaiwen Yang, et al. 2018. Heterogeneous memristive devices enabled by magnetic tunnel junction nanopillars surrounded by resistive silicon switches. *Advanced Electronic Materials* 4, 3 (2018), 1700461.

[42] Yuxiong Zhu, Borui Wang, Dong Li, and Jishen Zhao. 2016. Integrated Thermal Analysis for Processing In Die-Stacking Memory. In *IEEE International Symposium on Memory Systems*. 402–414.

[43] F. Zokaee, M. Zhang, and L. Jiang. 2019. FindeR: Accelerating FM-Index-Based Exact Pattern Matching in Genomic Sequences through ReRAM Technology. In *International Conference on Parallel Architectures and Compilation Techniques*. 284–295.