K-Nearest Neighbor Hardware Accelerator Using In-Memory Computing SRAM

Jyotishman Saikia*, Shihui Yin*, Zhewei Jiang†, Mingoo Seok†, Jae-sun Seo*
*School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA
†Department of Electrical Engineering, Columbia University, New York, NY, USA
Email: jsaikia@asu.edu

***Abstract*—The k-nearest neighbor (kNN) is one of the most popular algorithms in machine learning owing to its simplicity, versatility, and implementation viability without any assumptions about the data. However, for large-scale data, it incurs a large amount of memory access and computational complexity, resulting in long latency and high power consumption. In this paper, we present a kNN hardware accelerator in 65nm CMOS. This accelerator combines in-memory computing SRAM that is recently developed for binarized deep neural networks and digital hardware that performs top-k sorting. We designed and simulated the kNN accelerator, which performs up to 17.9 million query vectors per second while consuming 11.8 mW, demonstrating >4.8X energy improvement over prior works.**

***Index Terms*—k-nearest neighbor, content addressable memory, in-memory computing, hardware accelerator**

I. Introduction

A number of machine learning (ML) tasks such as pattern matching, data mining, and object recognition employ kNN algorithm to obtain the k nearest vectors from a large database upon an input vector query [1]. Compared to deep learning algorithms that require extensive parameter training, kNN algorithms do not require such parameters, allowing easy adoption in many ML applications. kNN algorithms are based on relatively simple arithmetic operations to calculate distance metrics (e.g. Euclidean, Manhattan, etc.). However, executing kNN tasks for large-scale databases consumes long latency and high energy, because it requires a large amount of memory accesses and arithmetic operations.

A kNN algorithm typically consists of two computation stages, namely distance calculation and a global top-k sort computation, where the distance calculation poses a major computational bottleneck. For distance calculation, prior works proposed using binary/ternary content addressable memory (BCAM/TCAM) [2], [3]. However, the challenge here is that BCAMs/TCAMs only produce a binary result of whether a search vector is matched with each storage vector or not, instead of providing a distance value between them. To address this limitation, prior works needed to use multiple TCAM searches [2] or multiple stages of data storage and computation [3]. But this inevitably increases latency/storage requirement. Ideally, we want to have CAM-like hardware that can quickly produce the distance value between a search vector and each of its storage vectors, instead of just informing whether there is a match or not.

This work is in part supported by NSF grant 1652866 and C-BRIC, one of six centers in JUMP, a SRC program sponsored by DARPA.

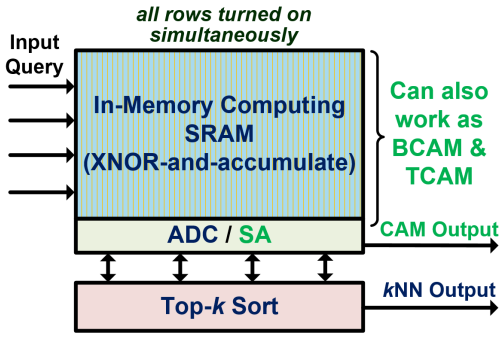

Fig. 1. Proposed kNN accelerator based on in-memory computing SRAM.

Such ideal CAM hardware can actually be built based on *in-memory computing SRAM hardware* [4]–[6] originally developed for accelerating binarized deep neural networks (DNNs) [7], [8]. Such in-memory computing SRAM can compute bitwise XNOR between a wordline vector and a storage vector of an SRAM column, and then accumulate the bitwise XNOR values of the two vectors. The XNOR-and-accumulate result is often represented as the analog bitline voltage, which subsequently goes through an analog-to-digital converter (ADC) at the column periphery.

For two binary vectors, since the bitwise XNOR-and-accumulate computation also represents the Hamming (and equivalently Manhattan) distance, we observe that the ADC output of XNOR-based in-memory computing SRAM (e.g. XNOR-SRAM [4]) can directly provide the distance information for kNN applications. The only additional hardware required is the module to perform the top-k sorting of the distance values from the in-memory computing SRAM. Fig. 1 shows the envisioned architecture of such an accelerator that combines the in-memory computing SRAM and the top-k sorter. We designed this accelerator in 65nm CMOS and simulated the performance and energy-efficiency. The results demonstrate >4.8X energy improvement over prior arts on kNN accelerators. Furthermore, by driving the wordlines of XNOR-SRAM arrays slightly differently, we show that the XNOR-SRAM array can be re-purposed to work as both BCAM and TCAM.

The remainder of the paper is organized as follows. Section II presents related prior works in the field. We discuss the

XNOR-SRAM and its application in Section III. The implementation of the kNN accelerator is elaborated in Section IV, and the experimental results are presented in Section V. The paper is concluded in Section VI.

II. Related works

A. Content Addressable Memory

Content addressable memories (CAMs) is a special type of memory hardware, whose contents are accessed by the contents itself rather than addresses. They can generate the results of a search operation (i.e., whether a content of interest is in the memory) in a single clock cycle. However, it requires a large amount of parallel hardware, embedded both in bitcells and peripheral circuits, which significantly increases dynamic power dissipation [9].

Over the years, several approaches have been proposed to improve the the performance and energy consumption of CAMs. In [10], a butterfly architecture has been designed to increase parallelism and reduce search time. A hierarchical match line (ML) scheme was also introduced to reduce the switching activity and consequently the power consumption. An automated background checking (ABC) scheme was presented in [11], which monitors the ML sensing using two dummy rows to track the optimal operating point. [12] implemented BCAM and TCAM based on off-the-shelf 6T SRAMs with new peripheral circuits in 28nm CMOS. A reconfigurable logic is used for the SRAM to enable operation of a single SRAM cell as a BCAM and two SRAM cells as a TCAM. This reduces CAM area and improves energy-efficiency.

B. kNN accelerators

Several prior works presented kNN accelerators. [13] present a 14nm kNN accelerator, which employed adaptive precision to reduce the hardware requirement and improve the latency. In this accelerator, after computing the current accuracy within pre-defined bounds, only vectors that could be possible winners are kept. This approach eliminates a considerable number of vectors. It can also support both Manhattan and Euclidean distance metrics.

[14] presented an approximate nearest-neighbor processor based on a spatio-temporal locality searching scheme. This architecture requires just four frames overlapping with the query vector, reducing the external memory bandwidth. It also incorporated a neuro-fuzzy module to alleviate the large error owing to a dynamically moving target object. The aforementioned CMOS ASIC based kNN accelerators [13], [14] used off-the-shelf SRAM and registers, which incur a large amount of memory accesses and data movements.

[15] investigated similarity search using Micron's Automata Processor (AP). AP's near-data processing architecture can minimize data movement and thus achieved superior kNN acceleration performance over CPUs and GPUs.

Other works have presented non-volatile memory (NVM) based CAMs for kNN acceleration. NNgine [3] presented similarity search using MTJ based TCAMs. [16] and [17] also investigated RRAM based kNN acceleration. On the other hand, [18] proposed a design that can morph CAMs into a binary neural network accelerator.

III. XNOR-SRAM: Custom SRAM Macro for In-Memory Computing

In [4], a mixed-signal in-memory computing SRAM macro titled "XNOR-SRAM" was presented, targeting energy-efficient DNN implementation. It performs XNOR-and-accumulate (XAC) operations in binarized neural networks (BNNs) [7], [8], which replaces multiply-and-accumulate (MAC) operations in non-binary DNNs, with high speed and energy-efficiency.

Fig. 2(a) presents the reported XNOR-SRAM array and peripheries, which can map convolution and fully-connected layers of convolutional neural networks (CNNs) and multi-layer perceptrons (MLPs). It consists of a 256-by-64 custom SRAM array, a row decoder, and a read periphery including a 3.46-bit (11-level) flash ADC.

Fig. 2(b) shows the 12T SRAM bitcell proposed in [4]. T1 to T6 form the conventional 6T SRAM bitcell; T7 to T10 form complimentary pull-up/-down circuits for the XNOR mode; T11 and T12 can power-gate the pull-up/-down circuits, if the corresponding column is not enabled, to save power. As shown in Fig. 2(c), the RWL driver translates each ternary/binary input of an input vector to four RWLs accordingly. Parallel pull-up and pull-down paths from all bitcells (controlled by bitwise XNOR outputs) in a column form a resistive voltage divider, where RBL is the output node. V_{RBL} is a monotonic function of XNOR bitcount (Fig. 2(d)); therefore, we can obtain the XAC results by digitizing V_{RBL} with the ADC. In one cycle, XNOR-SRAM supports computation with binary weights (+1/-1) and binary inputs (+1/-1 or +1/0) as well as ternary inputs (+1/0/-1). The embedded ADC plays a key role in speed and DNN accuracy. Employing 11 levels (3.46 bit) reportedly provides relatively high accuracy, and non-linear quantization based on statistical distribution of XAC values can further improve it. Fig. 2(d) shows the measured V_{RBL} for different XAC values. In the non-linear quantization scheme, the worst-case 3-σ deviation is equivalent to 1.78·LSB.

XNOR-SRAM macro prototyped in 65nm CMOS achieves 81.28 pJ and 178 ns for 64 operations of 256-input XAC at 0.6V, which represents 2.48 fJ per operation or 403 GOPS/W. For MNIST, a 3-layer MLP (512 neurons per layer) was used. The CNN for CIFAR-10 had six convolution layers and three fully-connected layers [8]. Performing accumulation of the XNOR-SRAM outputs, max-pooling, and batch normalization in digital simulation, the DNNs with XNOR-SRAM (digital baseline) achieve 85.7% (90.7%) accuracy for CIFAR-10 and 98.3% (98.8%) for MNIST [4].

One thing to note is that XNOR-SRAM performs column-by-column operation, sharing the ADC among 64 columns. Although XNOR-SRAM as well as other in-memory computing SRAMs show promising energy-efficiency at the single-array-level, none have demonstrated kNN or CAM functionalities with the same hardware.

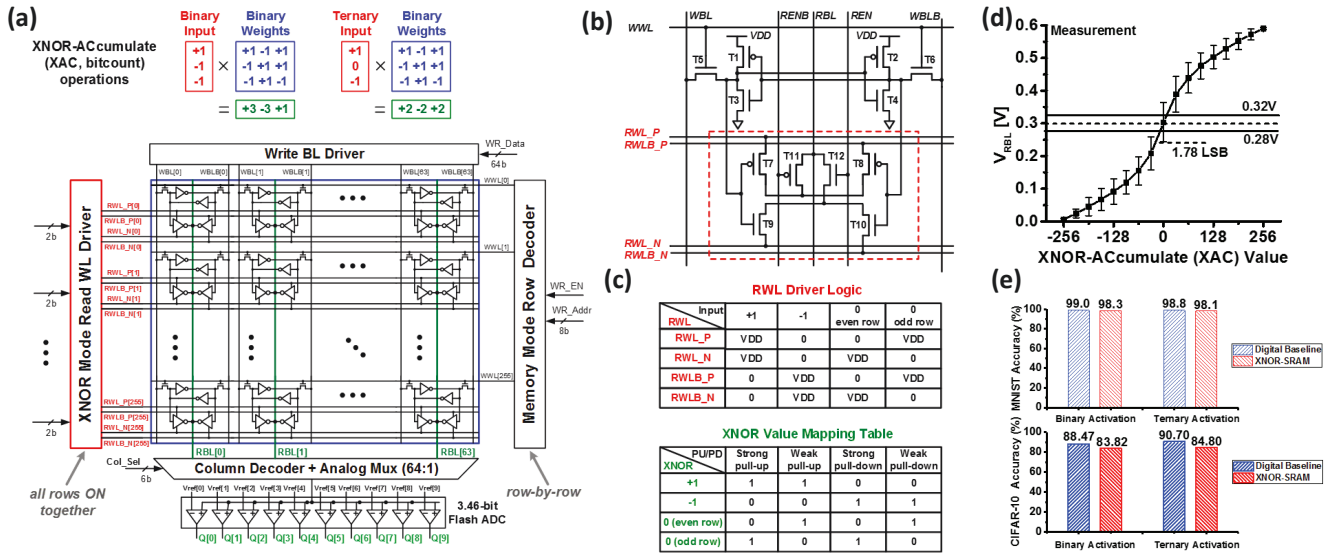

Fig. 2. (a) XNOR-SRAM [4] architecture can map XNOR-and-accumulate (XAC) operations. (b) Bitcell schematic. T7 to T12 are added to a 6T SRAM. (c) XAC operation with ternary activations and binary weights. (d) Measured V_{RBL} over corresponding logical results of 256-input XAC operations. (e) Classification accuracy for MNIST and CIFAR-10 datasets.

IV. Proposed BCAM/TCAM Design and kNN Accelerator Architecture

A. BCAM/TCAM Design

Using the same XNOR-SRAM array, BCAM and TCAM functionalities can be accomplished with small modifications. Out of 10 sense amplifiers (SAs) that comprised the flash ADC of XNOR-SRAM, we use only one SA for BCAM/TCAM functionalities, similar to conventional BCAM/TCAMs. If the input query vector and storage column vector are identical, then the resulting RBL voltage will be ∼Vdd. Therefore, we will use a high voltage value (e.g. 0.95×Vdd) for the reference voltage of the single SA.

The overall BCAM and TCAM operations using the XNOR-SRAM bitcells are illustrated in Fig. 3. Compared to the requirement of four different RWL signals in XNOR-SRAM, it can be seen that we only need two RWL signals "RWL" and "RWLB" for BCAM/TCAM functionalities. For BCAM, we simply drive the RWL and RWLB in a differential manner, so that the two signals have opposite values. When the input bit and the storage bit matches, the corresponding bitcell will drive RBL towards Vdd, otherwise RBL will driven towards 0V. For TCAM, we also need to incorporate the 'X' input. As shown in Fig. 3, this functionality can be achieved by driving both RWL and RWLB to Vdd, so that T7-T10 transistors are all driven to Vdd. This way, no matter whether the input bit value matches with the bitcell value or not, the corresponding bitcell will drive RBL towards Vdd.

B. kNN Accelerator Design

1) Overall Operation: For the kNN accelerator, we employ the in-memory computing XNOR-SRAM for distance calculation. As described in Section III, the ADC output of each XNOR-SRAM column represents the XNOR-and-accumulate value of two 256-input vectors. Since XNOR-SRAM employed a flash ADC with 10 SAs, the ADC output is a 11-bit thermometer code. As XNOR-SRAM is accountable for distance calculation, we implemented a digital module that sorts the top-k results from the XNOR-SRAM columns.

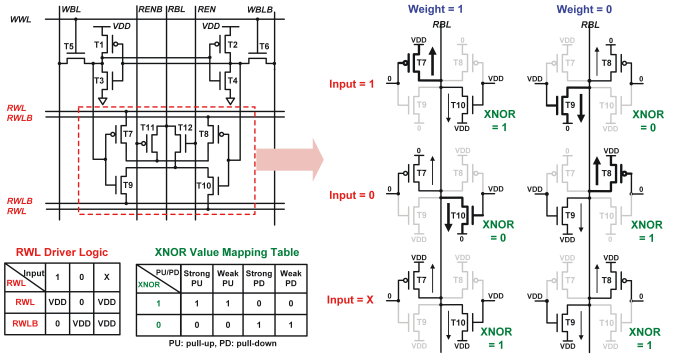

Fig. 3. XNOR-SRAM bitcell operation for BCAM (input: 1 or 0) and TCAM (input: 1, 0, or X) functionalities.

Considering that the XNOR-SRAM (one array consists of 64 columns) operates on a column-by-column basis [4], the digital sort module receives a new ADC output every cycle. Immediately after 64 cycles of receiving inputs, the digital sort module will output the top-k column addresses out of the sorted results.

The block diagram of the digital sort module is shown in Fig. 4. The digital sort module first converts the 11-bit thermometer code input to a 4-bit binary value. The remainder of the digital sort module consists of a comparison unit, data unit and address unit. The data unit holds the input vectors in descending order of their values, while the address unit maintains the input sequence number of the vectors in the same order. The data unit consists of shift registers to hold

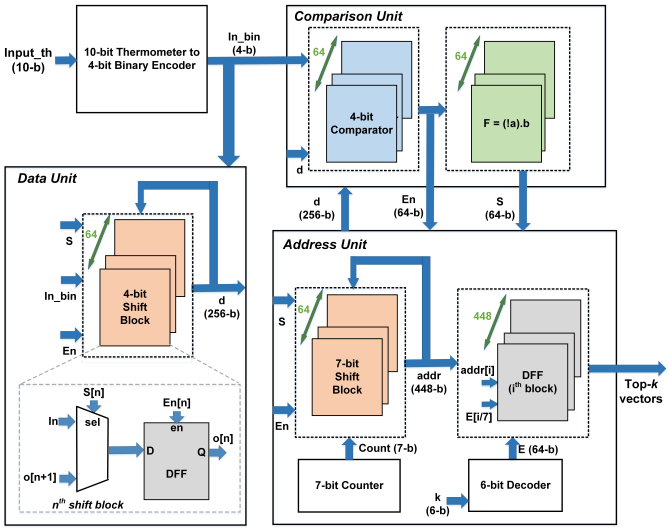

Fig. 4. Block diagram of digital sort computation module.

the input sequence numbers, one for each of the input vectors. Initially, all the shift registers in the data unit will have values of zero. As the new ADC outputs from the XNOR-SRAM arrive sequentially each cycle, the comparison unit compares the new 4-bit input value with the existing sorted results at the data block to determine its corresponding location for maintaining the updated sorted order. All the vectors with lower values are shifted by one slot to make space for the new input. In the address unit, the next sequence number is generated on the arrival of a new vector and it is placed at the slot corresponding to the new vector after shifting the sequence numbers of the shifted vectors in the data unit. After the 64^{th} column of XNOR-SRAM is processed, the digital sort module will output the sorted addresses of the columns that hold k-nearest neighbor vectors to the input query vector.

The overall timing diagram of the sorting operation is shown in Fig. 5. At the start of every cycle, a new data input comes into the sorting module from the ADC output of XNOR-SRAM. The shift registers in the data and address units hold the sorted list of data, which will be updated every cycle with respect to new input data. After the 67^{th} cycle, all data are sorted at the data unit, and the address unit outputs the addresses of the top-k vectors.

2) Sub-module Functionality: The comparison unit generates two signals, 'En' and 'S', which control the data and address units. The 'S' signal handles shifting of the values smaller than the input 'In_bin' towards LSB, and the 'En' signal is responsible for appropriate placement of 'In_bin' in the sorted list. The binary input is compared with the existing vectors in the shift registers of the data block. For all values greater than d (sorted data), we obtain 1, or otherwise 0. As there are 64 such computations, the comparison unit outputs a 64-bit 'En' result, which will have a distinct edge. This edge corresponds to the address in the sorted data at which the new input fits. By comparing two consecutive 'En' bit values, this edge is determined as 'S', which is used to target the specific address in the data unit where the new vector will be placed.

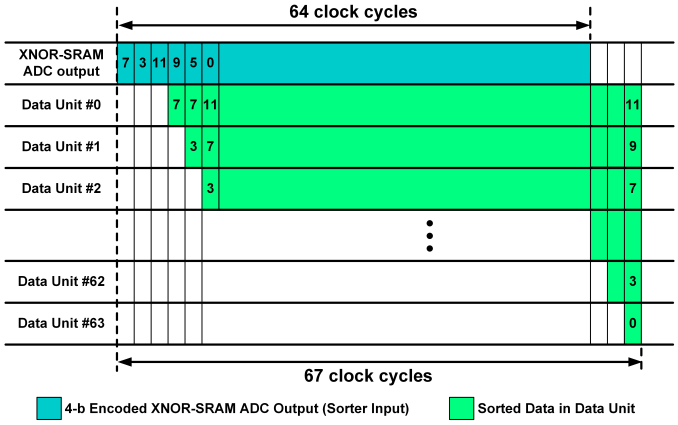

Fig. 5. Timing diagram and operation of the kNN accelerator.

In the data unit, we have 64 shift registers (shift blocks) that maintain the sorted data. Each shift register consists of a multiplexer and a D-flip-flop (DFF). The 'En' signal helps the multiplexer toggle between either the binary input or the output of the previous shift register. The 'S' signal is active when the input is greater than the existing data values, such that the data at these values will be shifted towards the LSB.

The address unit shares the same 'En' and 'S' signals with the data unit, and continually updates the sorted address values of XNOR-SRAM columns. The k parameter is an input to the address unit, which ensures that the k highest values are sent out as the final output of the proposed kNN accelerator.

3) Scalability to Larger Vector Sizes: The aforementioned digital sort module performs sorting of 64 distance values, since one XNOR-SRAM array has 64 columns. To scale up the number of vectors in our kNN accelerator from 64 to 128 and 256, we employ two and four XNOR-SRAM arrays, respectively, and we also design corresponding digital sort modules for 128 and 256 vector sizes. Compared to the 64-vector digital sort module, the main difference in the 128-vector and 256-vector digital sort modules is that the number of comparators in the comparison unit, the number of shift registers in the data/address units increased from 64 to 128 and 256, respectively.

V. Experimental Results

The two main modules that comprise the proposed kNN accelerator design are the XNOR-SRAM arrays and the digital top-k sort module. We implemented both in a 65nm CMOS technology. The XNOR-SRAM has been fabricated in a prototype chip and the measurement results have been reported in [4]. The digital sort module has been synthesized with Synopsys Design Compiler and place-and-routed using Cadence Innovus in the standard cell design flow. We have used Synopsys PrimeTime to perform post-layout static timing and power analysis simulation. We used RC parasitic annotated netlists and actual node toggling activity to obtain the latency and power results. Fig. 6 shows the conceptual physical design of the kNN accelerator based on the die photo of the XNOR-SRAM prototype and the digital sorter module layout view.

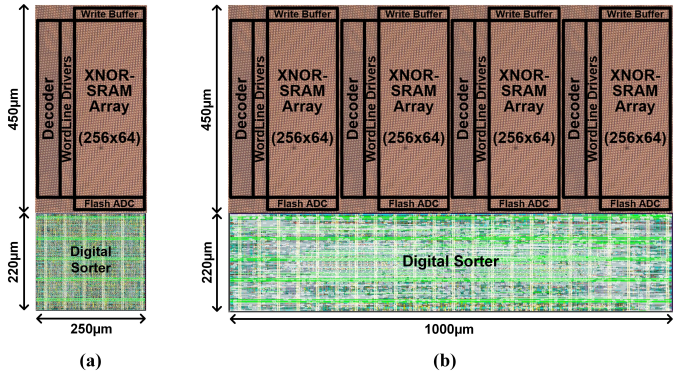

Fig. 6. Physical design diagram of the proposed kNN accelerator. The accelerators supporting (a) 64-vector and (b) 256-vector operations.

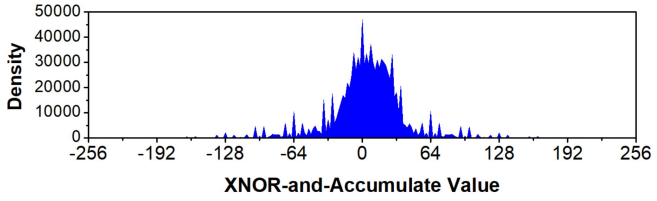

Fig. 7. Random data distribution of XNOR-and-accumulate values for each column.

For the kNN workload, we used random data for both the input query vector and the storage vectors. For one million pairs of 256-bit random data, we found the distribution of XNOR-and-accumulate values, which is shown in Fig. 7. Based on this distribution information, we found the average power using the power consumption for each XNOR-and-accumulate value reported from [4]. Since the distribution is highly centered around zero, the average power is similar to the power consumed by XNOR-SRAM array when the XNOR-and-accumulate value is zero, which is 4.4 mW at 1.2 GHz with 1.0V supply.

Fig. 8 shows the power breakdown of the kNN accelerator. Thanks to the high power-efficiency of XNOR-SRAM, it shows balanced power dissipation among the major building blocks. Specifically, the comparison unit responsible for generating the sorting signals takes less than $(1/5)^{th}$ of the total power. The data unit responsible for sorting the data consumes just over $(1/5)^{th}$ of the total power. The address unit handles a higher values requiring more architecture and as a result consumes about $(2/5)^{th}$ of the total power.

As we scale up the kNN vector size, we need to include

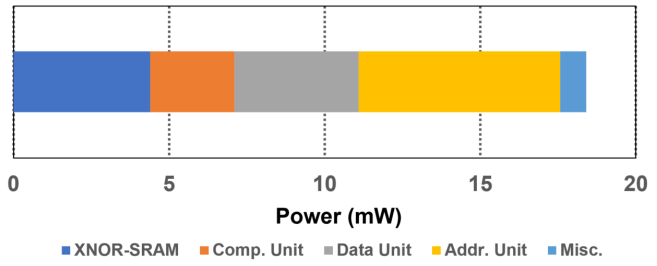

Fig. 8. Power breakdown of the kNN accelerator for vector size of 64.

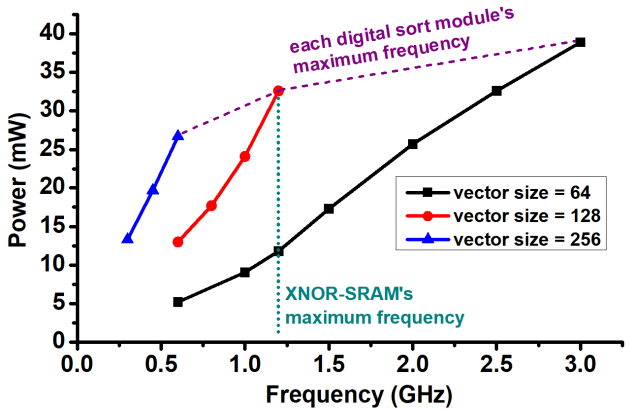

Fig. 9. Frequency and power consumption of the digital sort module for 64, 128 and 256 vector sizes.

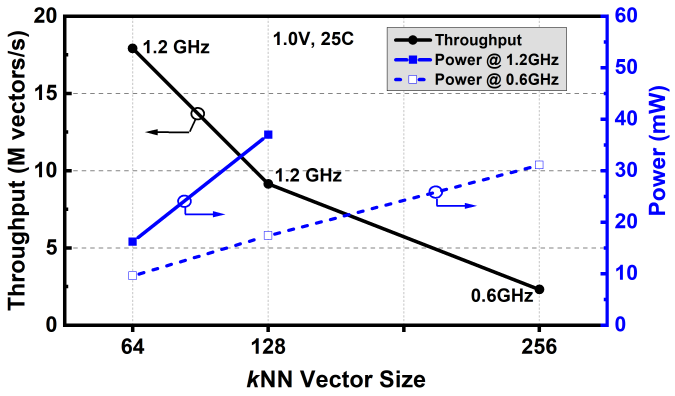

Fig. 10. Throughput and power consumption of the proposed kNN accelerator, including both XNOR-SRAM and digital top-k sorter.

additional hardware in the kNN accelerator. In particular, the sort module becomes more complex and also has longer interconnects, which adversely affects the maximum clock frequency of the accelerator. As shown in Fig. 9, the digital sort module of the kNN accelerator for the vector size of 64, 128 and 256 can operate up to 3 GHz, 1.2 GHz, and 0.6 GHz, respectively. The vector size increase also leads to higher power dissipation (Fig. 9). Also, the number of cycles required per query increases with larger vector size due to the increased waiting time for inputs. For vector size of 64, our kNN accelerator can perform 17.9 million query vectors per

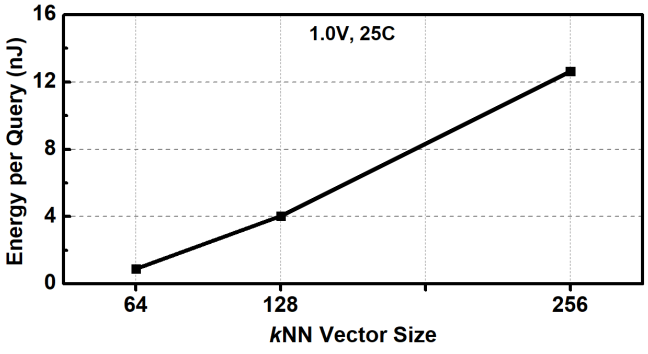

Fig. 11. Energy consumption of the proposed kNN accelerator across different vector sizes.

TABLE I
kNN ACCELERATOR COMPARISON

	This work	Kaul [13]	Hong [14]	Kim [19]
CMOS Tech.	65nm	14nm	130nm	130nm
Vector Dimension	256x64 / 256x128	128x128	128x128	272x128
Area (mm^2)	0.17 / 0.34	0.33	2.26	1.44
Supply (V)	1.0	0.85	1.2	1.2
Frequency (GHz)	1.2	0.42	0.2	0.2
Throughput (vectors/s)	17.9M / 9.15M	21.5M	0.13M	0.06M
Power (mW)	18.4 / 37.0	73	65	44
Normalized Energy/Query[1]	0.89 / 4.03	19.6	170	250

[1]Energy is normalized to 65nm, assuming capacitance scales by 0.7 in each technology generation, and used supply voltage values in the table.

second, consuming 16.2 mW when both XNOR-SRAM and the digital sort module operates at 1.2 GHz. When we increase the vector size to 128 and 256 for the kNN accelerator design, the throughput decreases (Fig. 10) and the energy per query linearly increases (Fig. 11).

In Table I, we compare our kNN accelerator results with prior hardware implementations. The design area for our work is at least 1.94X smaller than the next closest design [13]. Our kNN accelerator runs at a comparatively higher frequency, which helps to achieve higher throughput. With normalized energy values to 65nm CMOS using Dennard scaling, our work achieves >4.8X improvement in energy per query vector, compared to prior works for the case of 128 storage vectors.

In Table II, we compare our design in the CAM mode with prior BCAM/TCAM works. Due to the resistive divider nature of the XNOR-and-accumulate operation in XNOR-SRAM [4], the energy/search/bit is higher than other CAM-only works. Capacitive in-memory computing SRAM designs such as [5] do not exhibit crowbar current, and could help to reduce the energy/search/bit in the CAM mode. In our work, note that the same SRAM is capable of not only working as CAM, but also performing distance calculation for kNN applications, as well as for binary MAC computations for DNN applications.

TABLE II
CAM COMPARISON

	This work	Do [11]	Huang [10]	Jeloka [12]
CMOS Tech.	65nm	32nm	65nm	28nm
Transistors /cell	12T	10T	14T	6T/12T
Area/cell (μm^2)	3.92	3.3	7.05	0.15
Energy/search /bit (fJ)	4.62 (0.6V)	0.77 (1.2V)	0.165 (1V)	0.6 (1V)
Frequency (MHz)	360	500	400	370
Array Size	256x64	128x128	256x144	64x64
CAM Modes	BCAM /TCAM	BCAM	TCAM	BCAM /TCAM
Match-line Technique	XNOR-acc. + one SA	NOR	Butterfly match-line	Two single-ended SAs

VI. CONCLUSION

In this work, we have presented the design of a kNN accelerator based on XNOR-based in-memory computing SRAM. It supports variable vector sizes from 64 to 256, and it can also be configured to work as a binary/ternary CAM. The proposed kNN accelerator designed in 65nm CMOS achieves up to 17.9 million query vectors per second at 1.2 GHz frequency. Our work demonstrates >4.8X improvement in energy per query vector compared to prior works on kNN accelerators.

REFERENCES

[1] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: a lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[2] A. Bremler-Barr *et al.*, "Ultra-fast similarity search using ternary content addressable memory," in *International Workshop on Data Management on New Hardware*, pp. 12:1–12:10, 2015.

[3] M. Imani, Y. Kim, and T. Rosing, "NNgine: Ultra-efficient nearest neighbor accelerator based on in-memory computing," in *IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, Nov 2017.

[4] Z. Jiang, S. Yin, M. Seok, and J. Seo, "XNOR-SRAM: in-memory computing SRAM macro for binary/ternary deep neural networks," in *IEEE Symposium on VLSI Technology*, 2018.

[5] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *IEEE Symposium on VLSI Circuits*, 2018.

[6] R. Liu *et al.*, "Parallelizing SRAM arrays with customized bit-cell for binary neural networks," in *ACM/IEEE Design Automation Conference (DAC)*, 2018.

[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *European Conference on Computer Vision (ECCV)*, 2016.

[8] I. Hubara *et al.*, "Binarized Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 4107–4115, 2016.

[9] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, 2006.

[10] P.-T. Huang and W. Hwang, "A 65 nm 0.165 fJ/bit/search 256×144 TCAM macro design for IPv6 lookup tables," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 2, pp. 507–519, 2011.

[11] A. T. Do, C. Yin, K. S. Yeo, and T. T.-H. Kim, "Design of a power-efficient cam using automated background checking scheme for small match line swing," in *IEEE European Solid-State Circuits Conference (ESSCIRC)*, pp. 209–212, 2013.

[12] S. Jeloka, N. B. Akesh, D. Sylvester, and D. Blaauw, "A 28 nm configurable memory (TCAM/BCAM/SRAM) using push-rule 6T bit cell enabling logic-in-memory," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 4, pp. 1009–1021, 2016.

[13] H. Kaul *et al.*, "A 21.5 M-query-vectors/s 3.37 nJ/vector reconfigurable k-nearest-neighbor accelerator with adaptive precision in 14nm tri-gate CMOS," in *IEEE International Solid-State Circuits Conference*, 2016.

[14] I. Hong *et al.*, "A 125,582 vector/s throughput and 95.1% accuracy ANN searching processor with neuro-fuzzy vision cache for real-time object recognition," in *IEEE Symposium on VLSI Circuits*, 2013.

[15] V. T. Lee *et al.*, "Similarity search on automata processors," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 523–534, 2017.

[16] Y. Jiang, J. Kang, and X. Wang, "RRAM-based parallel computing architecture using k-nearest neighbor classification for pattern recognition," *Scientific Reports*, vol. 7, no. 45233, 2017.

[17] R. Kaplan, L. Yavits, and R. Ginosar, "PRINS: Processing-in-storage acceleration of machine learning," *IEEE Transactions on Nanotechnology*, vol. 17, pp. 889–896, Sep. 2018.

[18] W. Choi, K. Jeong, K. Choi, K. Lee, and J. Park, "Content addressable memory based binarized neural network accelerator using time-domain signal processing," in *ACM/IEEE Design Automation Conference (DAC)*, 2018.

[19] G. Kim, J. Oh, S. Lee, and H.-J. Yoo, "An 86 mW 98 GOPS ANN-searching processor for full-HD 30 fps video object recognition with zero-loss locality-sensitive hashing," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 7, pp. 1615–1624, 2013.