NVALT: Non-Volatile Approximate Lookup Table for GPU Acceleration

Mohsen Imani, *Student Member, IEEE,* Daniel Peroni, *Student Member, IEEE,* and Tajana Rosing, *Senior Member, IEEE*

***Abstract*—In this paper, we design a non-volatile approximate lookup table, called NVALT, to significantly accelerate GPU computation. Our design stores high frequency input patterns within an approximate NVALT to model each application's functionality. NVALT searches for and returns the stored data best matching the input data to produce an approximate output. We define a similarity metric, appropriate for binary representation, by exploiting the analog characteristics of the non-volatile content addressable memory. Our design controls the ratio of the application running between the approximate NVALT and accurate GPU cores in order to tune the level of accuracy necessary for user requirements. Our evaluations on seven general GPU applications shows that, NVALT can improve energy computation by 4.5× and performance by 5.7× on average while providing less than 10% average relative error as compared to the baseline GPU.**

***Index Terms*—Non-volatile Memory, Content Addressable Memory, GPU, Approximate Computing.**

I. INTRODUCTION

INTERNET of Things (IoT) increases the number of smart devices and the rate of data generation around the world [1], creating a significant demand for high speed and efficient parallel processors such as GPU. However, GPUs cannot learn existing patterns in workload and adaptively process the data [2]. When repeatedly running a single application, e.g. Fast Fourier Transform, on a GPU for different input data, the GPU performs a set of computations repeatedly without learning the functionality. By performing non-conventional brain-like computation, e.g. Neural network and neuromorphic computing solutions perform, the cores avoid costly repeated computations by learning sets of functionality and approximately model them [3]. This technique is effective for applications which can accept a level of approximation. Many real world algorithms, such as machine learning, are statistic in nature and will accept some inaccuracy in their computation. Several image and video processing applications accept error in computation without losing the precision required by user [2]. We add an efficient brain-like processing unit beside the GPU cores to accelerate the computation with approximation techniques.

Prior work tried to improve efficiency and accelerate the GPU/CPU computation by enabling approximation such as voltage over-scaling [4], precision scaling [5], designing approximate circuits [6], [7], and memoization using resistive acceleration [8], [9], [10], [11], [12]. Work in [3] accelerates programs by utilizing a neural network placed beside the GPU cores to produce approximate results. Although in CPU this technique provides a large energy/performance improvement [14], the advantage of neurally-based approximation in GPU is minor. Several other designs use the high density and zero leakage power of non-volatile memories for memoization or enabling approximation in GPU and CPU cores [8], [10], [13]. Work in [8] enabled GPU approximation by using CAMs beside floating point units (FPUs) and approximately retrieving data at run-time. This is a promising technique to save the GPU energy, but it cannot improve the performance, and also does not have a large impact on the overall GPU energy consumption, considering data movement and other non-FPU units.

In contrast, we propose a hardware approximation technique that uses resistive content addressable memories to accelerate the GPU computation. We exploit the analog characteristics of the non-volatile approximate lookup table, called NVALT, with nearest distance search capability. Our design learns and models the functionality of different applications by storing high frequency patterns for each application on an NVALT. At run-time, instead of processing data on the inefficient GPU core, our design searches NVALT to find the data most similar to the input operands. Our similarity metric considers the impact of each bit index to provide the best match. Our experimental evaluation on AMD Southern Island GPU architecture, running seven different applications, shows the enhanced GPU, as compared to the baseline GPU, can achieve 5.7× energy and 4.5× performance improvement, while providing less than 10% quality loss.

All authors are with the Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA, 92093 USA
E-mail: {moimani, dperoni, tajana}@ucsd.edu

II. PROPOSED NVALT

A. NVALT *Program Acceleration*

GPU workloads exhibit significant data similarity and locality. This locality increases as we go through the IoT domain with larger datasets. Several basic programs, such as Fast Fourier Transform (FFT) or image processing applications, consist of repeated building blocks, where each has many addition and multiplication operations. Running these algorithms on the conventional processors (e.g. CPU or GPU) is slow and requires large energy consumption. These applications can be approximately processed on the NVALT block. We exploit data locality to model several basic functions on the GPU with a fast and efficient approximate computing unit.

Fig.. 1 shows the overview of the structure of AMD Southern Island GPU, enhanced with NVALT accelerators. NVALT blocks are placed along side of each SIMD lane. When the approximate application is launched on GPU, the scheduler block runs the application on an NVALT block instead of sending them to GPU cores to process. An NVALT block approximately processes these basic functions inside a lookup table instead of precisely processing them. NVALT uses offline profiling for each program to find and store common input

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2746742, IEEE Embedded Systems Letters

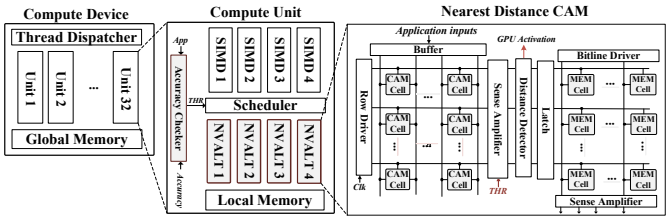
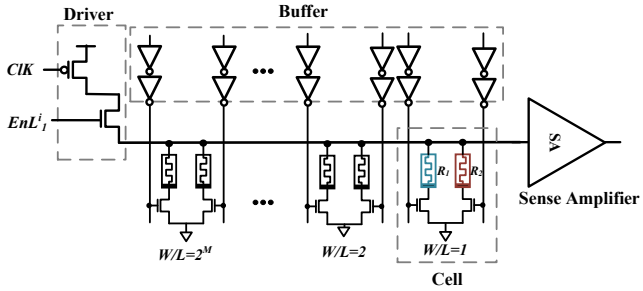


Fig. 1. The overview of the enhanced GPU with approximate NVALT block



Resistance value and select lines voltage during the search

Data	Conv. Cell		NVALT Cell		Search Operation	
	R1	*R2*	*R1*	*R2*	SL/SL'	ML
0	L	H	H	L	Vdd/0	Precharge
1	H	L	L	H	0/Vdd	

Fig. 2. The structure of CAM in NVALT block capable of searching for nearest distance row.

patterns and the corresponding outputs. At runtime, our design searches the input data stored in the NVALT and returns the most similar pattern as an approximate output. The accuracy of depends two main metrics:

- **Similarity metric:** In traditional cores the numbers represent by the fixed-point or floating point binary values, where the definition of similarity can be different based on representation. For instance, our prior work [10] considers Hamming distance as a metric to find the most similar row in a lookup table. This metric cannot provide high computation accuracy because it does not consider the impact each bit index has on the computation. Designing a lookup table/CAM which can find the exact nearest value requires large and inefficient peripheral circuitry. In this work we propose a simple technique which changes the weight of each bit index on the CAM structure and considers the binary weights on the search operation. We detail the implementation in Section II-B.
- **NVALT size:** The lookup table size directly impacts the computation accuracy. A large size NVALT can store a large number of patterns and increase the chance of a close match. Larger lookup tables result in the increase energy and degraded performance.

NVALT works for the applications which accept approximation in their computation. Applications must have limited number of input/output signals to efficiently fit within the NVALT table. We model the computation of these applications by storing their highly frequent input/output patterns inside a table. Profiling mode examines of the input data to find the most common occurrences.

Our evaluation shows that running all applications on the approximate NVALT may not result in high energy saving or speedup. We need to increase the size of the NVALT to store a sufficient number of patterns to ensure the acceptable computation accuracy. The increased size reduces the energy and performance efficiency that NVALT achieves during the search operation. The low energy and performance advantage of neural network-based GPU acceleration in [3] comes from using large and inefficient neural network blocks to provide enough computation accuracy. To address this issue, our paper provides configurable GPU approximation, where a fraction of data runs on the accurate GPU core, while the rest is run on the approximate NVALT. Our design calculates the distance of the input data with the stored value on the NVALT, and if the distance is larger than a threshold ($> THR$), it dynamically assigns this input data to an accurate core. The value of THR is determined based on the application type and user accuracy requirement.

B. NVALT Hardware Support

A conventional CAM consists of a CAM cell array, row driver, input buffer, and sense amplifier. Before each search operation, the match lines (ML) of all CAM rows are precharged to Vdd. During the search operation, input data is distributed to all CAM rows using an input buffer. This buffer strengths the input signals to ensure all rows receive the input signals at a similar time. During the search operation, the *ML*s in all CAM rows are discharged as long as at least one bit difference exists. We use a timing characteristic of *ML* discharging current to differentiate rows with different number of mismatches. In conventional CAM, any cell with mismatch discharges the *ML*. A larger number of mismatches results in a higher *ML* voltage drop, which can be detected by the sense amplifier. To detect a row with minimum mismatch, i.e., closest Hamming distance row, we need to find the row which discharges last. The sense amplifier tracks the *ML* voltages in all rows, until it find the slowest discharging one. The design is complicated and requires additional circuity, such as counters, while also taking a long time to determine the best matched row.
To address this issue, we use inverse CAM, proposed in [10], to design a CAM with nearest distance search capability. The CAM cells in our proposed NVALT work inversely, compare to the conventional CAM. Fig. 2 shows the functionality of NVALT cells storing inverse resistance values in match and mismatch cases. NVALT cell discharges the *ML* in case of matching data to the stored values, while a mismatch *ML* stays charged. A row which has more matched bits, i.e., a closest Hamming distance row, creates a faster *ML* discharging current than other rows. We detect the nearest distance row by finding the first row to discharge the *ML*.

Fig. 2 shows a CAM in single NVALT stage which is capable of searching for the row with the closest value to the provided inputs. The proposed NVALT consists of an input buffer, a row driver to selectively precharge *ML* of each row, and a sense amplifier to detect the first discharged row. Unlike other designs which use the hamming distance criteria, our design considers the impact of each bit index on the search operation of the NVALT block. We exploit different access transistor sizes for different bit indices. Based on the binary weight of an unsigned integer value, each cell in position i^{th} has access transistors which are $2\times$ larger than the cell in the $i-1^{th}$ adjacent bit. This results in $2\times$ higher *ML* discharging current in each match cell compared to its adjacent least significant

TABLE I
TUNING ACCURACY IN ROW-TUNABLE NVALT BLOCK

# of rows	128	256	512	1024	2048
BlackScholes	21%	13.1%	7.2%	4.0%	1.2%
FFT	8.3%	5.9%	3.1%	1.9%	0.7%
Sobel	13.4%	6.4%	4.2%	2.1%	1.4%
Inversek2j	17.3%	12.0%	8.3%	5.1%	2.9%

bits (LSB). The asymmetric access transistors weight each bit index on the search operation to find the closet row.

III. EXPERIMENTAL RESULTS

A. Experimental Setup

We evaluate the efficiency of the proposed NVALT on the AMD Southern Island GPU, Radeon HD 7970 device, which is a recent GP-GPU architectures. We modified the source code of Multi2sim, a cycle-accurate simulator [15], to integrate the NVALT within GPU. We used McPAT tool [16], to measure the energy consumption of GPGPU and other changes on the GPU architecture, including the registers and FIFO (4KB/SIMD). We also used CACTI to measure the energy consumption of the memory keeping the highly frequency patterns corresponding to each application. To measure the power consumption of the NVALT, we use HSPICE circuit level simulation in 45nm TSMC technology. We use VTEAM memristor model [17] for our memory design simulation with R_{ON} and R_{OFF} of $10k\Omega$ and $10M\Omega$ respectively. In terms of reliability, the endurance of resistive memory limits to 10^7 write operations, our design addresses this issue by limiting the number of writes to once for each application. We compare the efficiency of proposed design by running seven general OpenCL applications: *BlackScholes, FFT, Sobel, inversek2j, Laplacian, Convolution* and *Binarization*.

B. NVALT Accuracy Tuning

NVALT can be used next to GPU SIMD lanes to approximately process different applications. We use the approximate building block as a stand-alone computing unit to approximately process the desired applications. In this mode, the NVALT accuracy tunes using the size of the table that we are using. As we explained in section II, there is an efficiency and accuracy trade-off in choosing the best NVALT size to process the data. Small size table can provide significantly higher energy savings and performance improvement at the cost of decreased accuracy. Increasing the table size increases the computation accuracy by storing more common patterns, thus increasing the possibility of close hit. However, NVALT with many rows requires larger interconnects and column drivers to distribute the data simultaneously among all CAM rows. This energy overtakes the search energy (the energy consumed by the array), for CAM larger than 1024 rows.

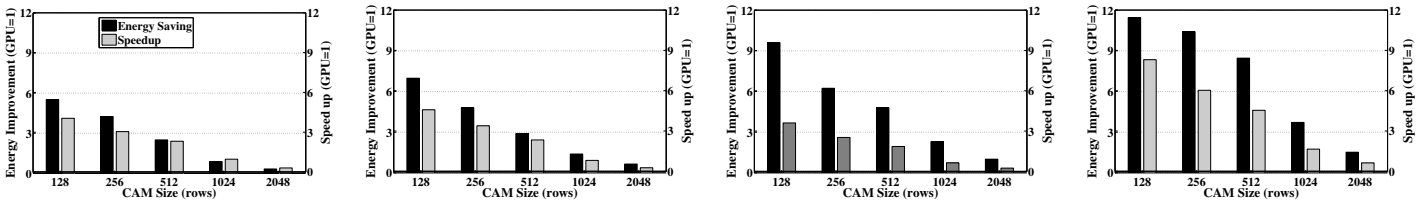
Our design is able to dynamically change the number of active rows on the NVALT (128-row granularity) in order to get the maximum accuracy for each application. We enable this functionality by chaining the structure of the row driver. Fig 3 shows the energy and performance improvement of four general applications on the proposed enhanced hardware. For each CAM size, the energy and performance have been normalized to the conventional GPU not using NVALT. Table I shows the computation accuracy of each application for different CAM sizes. Our result shows the applications have different accuracy sensitivity to the NVALT size, as each application requires a different number of rows to provide acceptable quality of service. To achieve less than 10% quality loss, our design needs to use an NVALT up to 512 rows. The result shows that our design achieves 5.3× energy improvement and 3.2× speedup, when selecting the optimal NVALT size for each application. The main issue of the row tuning technique is its weakness in providing energy/performance advantage for high quality computation. To provide less than 4% quality loss, our design does not save energy energy or performance.

C. GPU-NVALT computing

To address the quality issue in row tuning technique, we need to understand why very large NVALT blocks still result in sizable error. NVALT shows high error running inputs that have low similarity to the stored values. Although the percentage of data with low similarity makes up a small fraction of each workload, the impact of them on accuracy is high. In most cases, these infrequent patterns of low similarity data are the main information of the input. For example, in image processing, the image edges have the low frequency compared to the image background, but the edges contain the main information of image. To improve the NVALT accuracy, our design dynamically finds the far input values and assigning them to accurate SIMD cores to process. The ratio of running data on the approximate and precise core determines the level of accuracy that our design can achieve. Fig 4 shows the maximum energy improvement and speedup that our design can achieve in each NVALT size when the threshold value changes. The THR has been set in order to ensure that GPU quality loss is less than 4%. The top y-axis in the figure shows the percentage of time the application runs on CAM for a given THR value. Our evaluation shows that, in small NVALT size, we need to run larger portion of data on the precise GPU core in order to guarantee the computation accuracy satisfies the required level. By contrast, in large size, the NVALT can provide enough computation accuracy even using very large threshed value. We consider energy-delay product (EDP) to find the optimal CAM size resulting in the best efficiency. Our evaluation shows that 256-row CAM results in the best efficiency of 4.4× energy improvement and 3.4× speedup across the seven tested applications (normalized to conventional GPU).

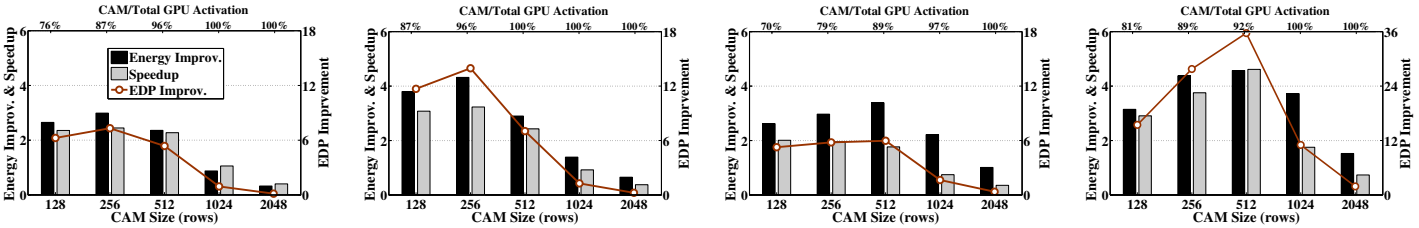
In all applications, 4% quality loss does not show the acceptable quality of service. Therefore, we change the *THR* value for each application to see the efficiency of the NVALT in different level of accuracy. Table II shows the energy-delay product improvement (Normalized to GPU) that our design can achieve for different quality loss from 2% to 10%, as the NVALT size changes. The result shows that our design can tune the level of approximation by partially running the application on NVALT and accurate GPU core. Our result shows that NVALT works much more efficiently for applications which can accept large approximation. For instance, by accepting 6% error, the NVALT can achieve up to 3.7× speedup, while this number decreases to 1.4× to achieve 1% quality loss. Fig 5 shows an example of tuning when running

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2746742, IEEE Embedded Systems Letters



(a) BlackScholes (b) FFT (c) Sobel (d) inversek2j

Fig. 3. Energy improvement and speedup of the NVALT in different sizes.

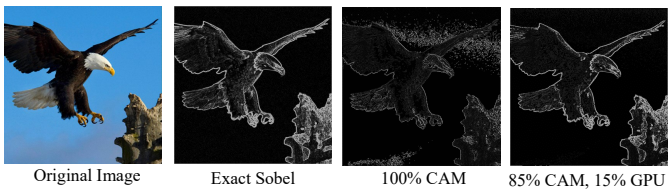


(a) Black Scholes (b) FFT (c) Sobel (d) inversek2j

Fig. 4. Energy improvement, speedup and EDP of enhanced-GPU with NVALT in different CAM sizes while ensuring less than 4% quality loss.

TABLE II
ENERGY-DELAY PRODUCT IMPROVEMENT OF THE ENHANCED-GPU WITH NVALT IN DIFFERENT QUALITY LOSS

Quality loss	**1%**	**2%**	**4%**	**6%**	**8%**	**10%**
Best CAM size	1024-row	512-row	256-row	256-row	128-row	128-row
BlackScholes	1.1×	3.2×	7.3×	10.2×	12.3×	14.7×
FFT	2.7×	6.3×	13.9×	20.5×	23.9×	30.1×
Sobel	2.9×	9.9×	27.7×	38.6×	45.5×	54.0×
inversek2j	1.0×	2.8×	5.7×	8.4×	10.3×	13.3×
Laplacian	3.1×	5.2×	8.4×	12.3×	14.9×	19.3×
Convolution	8.6×	21.2×	32.8×	45.8×	61.4×	79.3×
Binarization	2.9×	5.8×	9.3×	13.5×	17.0×	22.1×
Average	**3.2**×	**7.6**×	**15.0**×	**21.3**×	**26.5**×	**33.2**×



Original Image Exact Sobel 100% CAM 85% CAM, 15% GPU

Fig. 5. Output quality of Sobel application when the application completely or partially runs on NVALT.

an image through the Sobel application. When the image runs completely on NVALT it shows noticable degredation, but when it runs partially on GPU, the result shows little observable difference between the baseline GPU output image and the tuned approximate output image. The majority of an application's data can be accelerated on approximate NVALT and only small portion needs to run on precise cores.

NVALT uses a single CAM block to accelerate all applications. Our design reconfigures the NVALT to accelerate applications based on the current application running on GPU. The area efficiency is one of the main advantages of using resistive memory. This technology provides the cell density of $8F^2$ (F is feature size). Using CAM with 256 rows required $530\mu m^2$.

IV. CONCLUSION

This paper proposes a hardware approximation technique which accelerates programs in the GPU architecture. We design a non-volatile approximate lookup table (NVALT) with the capability of searching for the nearest distance row. NVALT approximately models computation by storing high frequency patterns corresponding to each application. Our NVALT searches and returns a stored value with the most similarity to the input data. Our design can tune the level of approximation, by dynamically splitting input data between the approximate NVALT or accurate GPU cores. Our experimental evaluation running seven general applications on AMD GPU shows that NVALT can improve the GPU energy computation by 4.5× and performance by 5.7× on average while providing less than 10% average relative error, as compared to the unmodified GPU.

V. ACKNOWLEDGMENT

This work was supported by NSF grants 1730158 and 1527034.

REFERENCES

[1] J. Gantz, et al., "Extracting value from chaos," *IDC iview*, Vol.1142, pp.1-12, 2011.
[2] M. Imani, et al., "ACAM: Approximate computing based on adaptive associative memory with online learning," *IEEE/ACM ISLPED*, pp. 162-167, 2016.
[3] A. Yazdanbakhsh, et al., "Neural acceleration for gpu throughput processors," *IEEE/ACM Micro*, pp. 482–493, 2015.
[4] L. Leem, et al., "ERSA: Error resilient system architecture for probabilistic applications," *IEE/ACM DATE*, pp. 1560–1565, 2011.
[5] S. Venkataramani, et al., "Quality programmable vector processors for approximate computing," *IEEE/ACM Micro*, pp.1-12, 2013.
[6] K. Nepal, et al., "ABACUS: A technique for automated behavioral synthesis of approximate computing circuits," *IEEE/ACM DATE*, pp. 361, 2014.
[7] M. Imani, et al., "CFPU: Configurable Floating Point Multiplier for Energy-Efficient Computing," *IEEE/ACM DAC*, pp. 76, 2017.
[8] M. Imani, et al., "Resistive configurable associative memory for approximate computing," *IEEE/ACM DATE*, pp. 1327–1332, 2016.
[9] X. Yin, "Design and benchmarking of ferroelectric FET based TCAM," *IEEE/ACM DATE*, pp. 1444–1449, 2017.
[10] M. Imani, et al., "Resistive CAM Acceleration for Tunable Approximate Computing," *IEEE TETC*, pp. 1, 2016.
[11] V. Akhlaghi, "Resistive bloom filters: from approximate membership to approximate computing with bounded errors," *IEEE/ACM DATE*, pp. 1441–1444, 2016.
[12] M. Imani, et al., "Masc: Ultra-low energy multiple-access single-charge tcam for approximate computing," *IEEE/ACM DATE*, pp. 373–378 2016.
[13] M. Imani, et al., "Approximate computing using multiple-access single-charge associative memory," *IEEE TETC*, pp. 1, 2016.
[14] H. Esmaeilzadeh, et al., "Neural acceleration for general-purpose approximate programs," *IEEE/ACM Micro*, pp. 449–460, 2012.
[15] R. Ubal, et al., "Multi2Sim: a simulation framework for CPU-GPU computing," *ACM PACT*, pp. 335–344, 2012.
[16] S. Li, et al., "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," *IEEE/ACM Micro*, pp. 469–480, 2009.
[17] S. Kvatinsky, et al., "VTEAM: a general model for voltage-controlled memristors," *IEEE TCAS II*, pp. 786–790, 2015.