A Hardware/Software Co-Design Method for Approximate Semi-supervised K-Means Clustering

Pengfei Huang[1], Chenghua Wang[1], Ruizhe Ma[1], Weiqiang Liu[1*] and Fabrizio Lombardi[2]

[1]College of EIE, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2]Department of ECE, Northeastern University, Boston, USA
E-mail: {pfhuang, chwang, mraiser, liuweiqiang}@nuaa.edu.cn, lombardi@ece.neu.edu

***Abstract*—As one of the most promising energy-efficient emerging paradigms for designing digital systems, approximate computing has attracted a significant attention in recent years. Applications utilizing approximate computing can tolerate some loss of quality in the computed results for attaining high performance. Approximate arithmetic circuits have been extensively studied; however, their application at system level has not been extensively pursued. Furthermore, when approximate arithmetic circuits are applied at system level, error-accumulation effects and a convergence problem may occur in computation. Semi-supervised learning can improve accuracy and performance by using unlabeled examples. In this paper, a hardware/software co-design method for approximate semi-supervised k-means clustering is proposed. It makes use of feature constraints to guide the approximate computation at various accuracy levels in each iteration of the learning process. Compared with a baseline design, the proposed method reduces the power-delay product by over 67% while only a small loss of accuracy is introduced. A case study of image segmentation validates the effectiveness of the proposed method.**

***Index Terms*—Approximate computing, semi-supervised learning, approximate multiplier, K-means clustering**

I. Introduction

Speed is improving due to technology advancement; however energy efficiency still remains an hurdle. Power and energy consumption have become a major concern for chip design. Solutions such as dark silicon are of only a limited viability when considering extensive silicon resources [1].

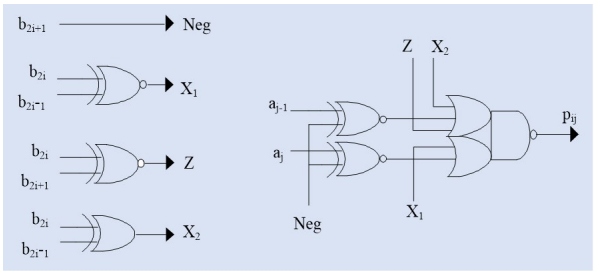
For hand-held systems and emerging smart devices, energy efficiency has become critical when dealing with computation intensive tasks, such as machine learning and multimedia signal processing. Significant efforts have already been devoted to improve energy efficiency at various levels, from software, to architecture all the way down to circuit and device levels. However, as computer systems become pervasive, computing workloads have significantly increased due to new applicative areas such as big data and IoT. So an improvement in the energy efficiency for these emerging workloads is urgently needed to keep pace with the growth of processed information. Applications such as signal processing, machine learning and pattern recognition are generally error-resilient in nature [2] so this feature can be used to alleviate this problem.

Approximate computing [3], trades computing accuracy for high performance and energy efficiency; it has attracted research and development efforts from both academia and industry. Approximate computing is based on the observation that the inputs and outputs of some algorithms are robust to an appropriate imprecision, so inexact operations in computation may have little or no effect on the final quality. Approximate computing techniques have been extensively studied at both hardware (such as circuit designs and computing architectures) and software levels [4]–[6]. As key components in arithmetic circuits, a number of approximate adders [7], [8] and multipliers [9] have been proposed; these circuits yield incorrect results for some input combinations.

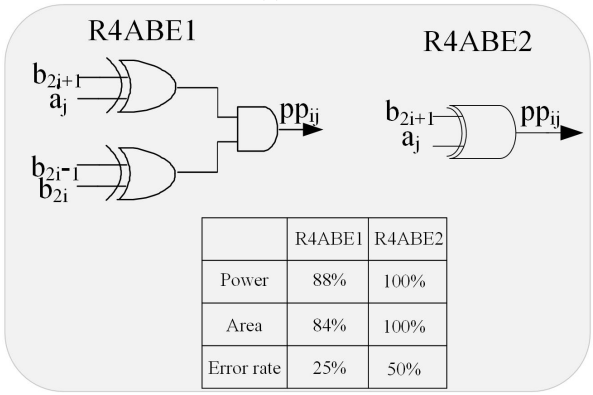
Multipliers require more hardware resources and incur in a higher energy consumption than adders; moreover, they are also slower than adders. Approximate multipliers are important due to the extensive use in error-tolerant applications. [10] considers a radix-8 Booth encoding of 3X by utilizing an error reduction scheme in the approximate adders. [11] reports power savings of up to 66% without affecting the accuracy of programs (manipulating low resolution data) by utilizing a bit width reduction in floating-point multipliers. In [12], approximate adders are leveraged to design an approximate multiplier with configurable accuracy. A novel design of an approximate Booth multiplier is proposed in [13]. A power-efficient multiplier designed using 2x2 approximate multiplier blocks is presented in [14]. However, most of these works only consider approximate designs at circuit level, so error effects at system or algorithm level are not fully addressed.

The advantages of approximate computing cannot be fully exploited by only considering hardware or software; therefore, an hardware/software (HW/SW) co-design [5], [15], [16] must be considered to change the abstractions and relationships between hardware and software for a trade-off between accuracy and efficiency [15] for approximate computing. [17] proposes a processor for high performance on-demand approximate computing with a complete open-source development framework; it consists of a hardware processor and an associated software tool set. Fine-grained [18] approximations are explored for individual instructions and individual words of memory. Coarse-grained approximations [19] can holistically transform entire algorithms.

In this paper, we propose a hardware/software co-design



(a)



	R4ABE1	R4ABE2
Power	88%	100%
Area	84%	100%
Error rate	25%	50%

(b)

Fig. 1: Gate-level circuit of: (a) exact MBE and (b) approximate MBE (with hardware evaluation).

method for approximate semi-supervised k-means clustering; in the proposed method, we utilize a semi-supervised learning method to smoothly calibrate the approximation level and allowing an algorithm to tolerate errors from the approximate hardware. During the computation process, approximation is adaptively tuned and in some cases, it can be even completely turned off if specific feature constraints are applicable.

The main contributions of this paper are summarized as follows:

- A novel hardware/software co-design method is proposed for semi-supervised k-means clustering.
- The proposed method explores error tolerance of semi-supervised k-means clustering.
- Supervised information, which are the minority of the data, improves not only the efficiency of clustering but it also contributes to the optimization of the so-called approximate factor as figure of merit.
- Adaptive approximate levels of hardware implementation can be obtained according to the result of each multiplication or accumulated results at each iteration with supervised information.
- The advantages of the proposed method are theoretically analyzed and experimentally demonstrated.

The remaining part of this paper is organized as follows: In Section 2, the preliminary of semi-supervised k-means and approximate multiplier is briefly reviewed. Section 3 presents the proposed HW/SW co-design method for the approximate semi-supervised k-means clustering algorithm. Section 4 provides the error analysis and the simulation results. The application of the proposed algorithm to image processing is given in Section 5. Section 6 concludes the paper.

II. Preliminary

A. Semi-supervised K-means

Semi-supervised learning is a class of supervised learning tasks and techniques that make use of unlabeled data for typically training a small amount of labeled data with a large amount of unlabeled data; k-means clustering partitions n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

Semi-supervised k-means leverages user-provided pairwise constraints either to learn an appropriate distance metric in the feature space, or to guide a clustering algorithm towards the correct clusters. The objective function $\mathcal{J}_{obj}$ of semi-supervised k-means utilizing feature constrains is as follows:

$$min\mathcal{J}_{obj} = D(\vec{X}, \vec{C}) + \mathcal{W}\varphi_D(\vec{X}_{pw}) \quad (1)$$

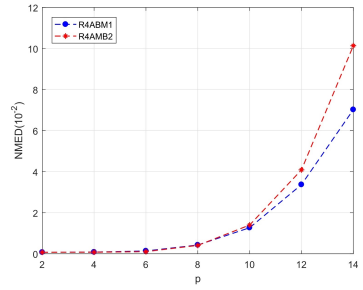
where, D is the distortion function, $\vec{X}$ and $\vec{C}$ are the samples and cluster centroids, $\mathcal{W}$ is the set of weights of the penalty for violating the feature constraints, φ_D is an increasing function of the distance between two samples and $\vec{X}_{pw}$ are samples with pair-wise constraints. For convenience, $\mathcal{W}$ is chosen as 1.

In the k-means clustering algorithm, a 5% loss in classification accuracy permits a 50 times energy saving compared to the fully accurate classification [20]; therefore, k-means is a typical application that trades accuracy for performance.

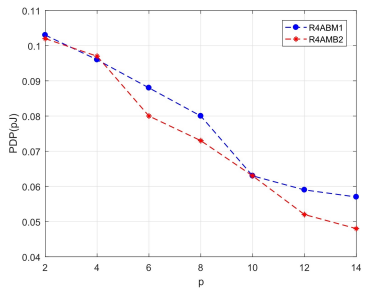
B. Radix-4 Approximate Booth Multiplier

In this paper, an 8-bit approximate Booth multiplier [13] is utilized. Booth encoding has been proposed for improving performance of multiplication of two complement binary numbers; this design has been further improved by using a radix-4 Booth encoding. In [13], complexity of the approximate Booth encoder has been reduced by at least an order of magnitude compared with an exact design as shown in Fig. 1. Furthermore, the so-called approximation factor p $(p = 1, 2, ..., 2N)$ is defined as the number of least significant partial product columns that are generated by the approximate Booth encoders as the approximate circuit ($i.e.$, approximate radix-4 Booth encoding (R4ABM)) can be used in all or only part of the partial product generation process. Fig. 2 presents a plot of the power-delay product (PDP) and the normalized mean error distance (NMED) under different approximation factors (p).

Fig. 2 shows that the NMED increases, while the PDP of R4ABM decreases with an increase of p. Therefore, the impact of p on the performance of an approximate multiplier is that



(a)



(b)

Fig. 2: (a) The error characteristics and (b) hardware resources consumption of the R4ABMs [13] at different approximate factors.

an increase in computing accuracy can be traded off for a higher energy efficiency by using an approximate multiplier at a higher value of p. A larger p leads to a simpler logic; this results in a lower hardware consumption but more error. For a given data path in an implementation, p establishes a design space in which every multiplication could utilize a better value for p such that this approximate multiplier would ensure a desirable computing accuracy while also achieving the highest energy efficiency.

III. Proposed HW/SW Co-Design Method of Approximate Semi-supervised K-means Clustering

Recently, few approximate designs of arithmetic circuits have been proposed to configure the approximation level [12], [13]. Most of these works consider a configuration at hardware level. [17] utilizes additional flags with instructions in the approximate behavior (for example enable/disable the cache look-up table); however, a gradual approximate computation is not allowed. To fully exploit the potential of approximate computing, the HW/SW co-design utilizing both R4ABM and an approximate semi-supervised k-means algorithm is studied in this section.

A. Fixed-Point Quantization

When data has large values, a higher precision in approximate computation usually incurs in a large penalty. Recently, [21]–[23] has exploited error-tolerance at data level

Algorithm 1 Approximate Semi-supervised k-means Clustering

Require:
The dataset $\mathcal{X}$;
The number of clusters k;
A set of must and cannot links;
A distance function D;
A set of weights for violating the feature constrains;
A default value of approximate factor p;
Ensure:
A partition of $\mathcal{X}$ in k groups
Assign initial centroids C
repeat
  Iteration i increase;
  Re-assign the labels of the examples using the centroids c_i to minimize $\mathcal{J}_{obj}$;
  Check the penalty φ of the violation of the feature constraints;
  if $\varphi_i > \varphi_{i-1}$ and $p > p_{min}$ **then**
    $p--$
  else
    if $p < p_{max}$ **then**
      $p++$
    end if
  end if
until Convergent or Pre-defined Max number of iterations reaches

by utilizing fixed-point quantization. Based on the assumption that the dynamic range of the precision of the floating points from a data set is bounded, an approach for fixed-point quantization is constructed in terms of the width of the arithmetic unit. Floating-point numbers can be extended or compressed as fixed-point numbers according to the desired function as follows:

$$\begin{cases} k * x_{max} + b = y_{max} \\ k * x_{min} + b = y_{min} \end{cases} \tag{2}$$

where, x_{max} and x_{min} are the max and min values of the floating-point inputs, y_{max} and y_{min} are the max and min of the available fixed-point values. Consider an 8-bit width as example; y_{max} and y_{min} are given by $127(2^7-1)$ and $-128(-2^7)$. For sake of efficiency, k will be modulated as k' commonly like the $m-th$ power of 2:

$$\begin{aligned} k' &= 2^m \\ s.t.\ 2^m &\leq k < 2^{m+1} \end{aligned} \tag{3}$$

After the modulation of k, x can be easily mapped into the range of (y_{min}, y_{max}) and get its fixed-point quantization with shifting bits of the fraction based on m above and its exponent part.

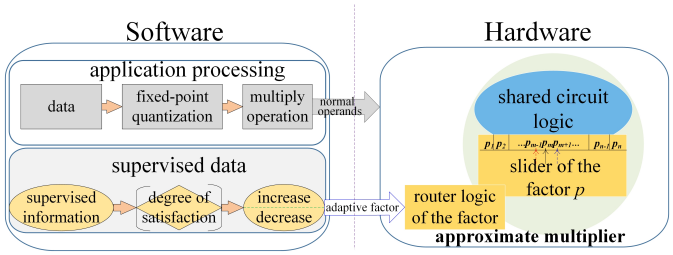


Fig. 3: Overview of proposed framework for semi-supervised approximate computing co-design.

B. Proposed Method for Semi-Supervised k-means Clustering

Usually, the approximate factor is established a-priori and its value is stored in the cache of the control circuit for p. Feature constrains of the supervising information are not as common as for the entire samples; so it is possible to control error-tolerance using an approximate circuit. Therefore, a co-design framework for semi-supervised approximate k-means clustering is proposed by using feature constraints. Its objective function can be obtained as follows:

$$min\mathcal{J}_{obj} = D(\vec{X},\vec{C}) + \mathcal{W}\varphi_D(\vec{X}_{pw})$$
$$s.t. \begin{cases} \varphi_{D_{m_p}}(\vec{X}) \le \varphi'_{D_{m'_p}}(\vec{X}) \;\&\&\; p < p_{max},\; p{+}{+} \\ \varphi_{D_{m_p}}(\vec{X}) > \varphi'_{D_{m'_p}}(\vec{X}) \;\&\&\; p > p_{min},\; p{-}{-} \end{cases} \quad (4)$$

where, m_p is the multiplier of the approximate factor p, $\varphi_{D_{m_p}}(\vec{X})$ is the penalty value of the violations of the constraints at the current stage based on the multiplier m_p and $\varphi'_{D_{m_p}}(\vec{X})$ is at the previous stage based on the previous multiplier m'_p with approximate factor p' . When the new penalty value $\varphi_{D_{m_p}}(\vec{X})$ is higher than the previous one, the approximate level should be reduced and the approximate factor p is decreased if p is greater than p_{min}. By contrast, the approximate factor p will increase if p is less than p_{max}.

Fig. 3 gives the overview of the proposed framework. In general, this approach utilizes a small amount of data with supervised information to adaptively find the optimized approximate factor not only for the supervised data but also the great majority of the unlabeled data. Meanwhile, the approximate factor is optimized gradually to reduce the large accuracy loss introduced by approximation operations. The approximation level is controlled by the approximation factor p and can be adjusted according to the violations of the supervising information.

The procedures for semi-supervised approximate k-means clustering are summarized in Algorithm 1. The entire approximate clustering process is supervised using the semi-supervised feature constraints for different approximation levels, so reducing the accumulated errors at algorithm level with a small amount of supervised information.

From Fig. 4, after each iteration or each computation of supervised data in every iteration, the approximate factor can be adaptively tuned by decreasing (the red) or increasing (the blue). The approximate factor can be updated through the multiplication of the supervised data which are in the minority. Then the unlabeled data which are in the majority can also benefit from the better approximate multiplier for the current phase. Moreover, the approximate factor can be updated not only according to the penalty value after each iteration but also at each multiplication of the supervised process. In this work, we only update the approximate level through the factor after each iteration in the next evaluations.

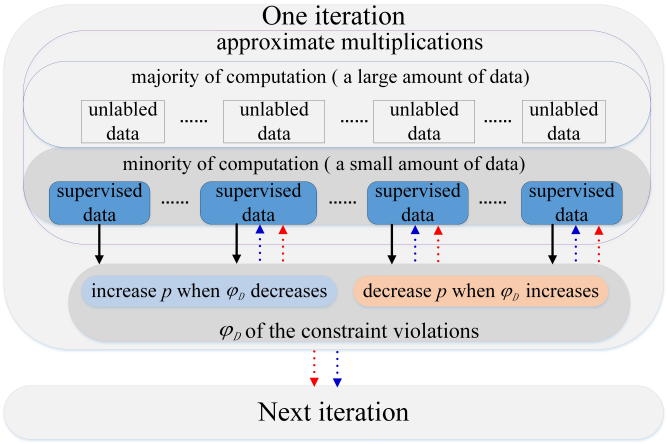


Fig. 4: The process of the proposed method with adaptive approximate factors.

IV. Evaluation and Analysis

To evaluate the effectiveness of the proposed method and the energy reduction for the multiplication, the classical k-means algorithm is selected as benchmark using the standard UCI datasets [24].

The clustering results using 32-bit full-precision, 8-bit fixed-point quantized, various (p=2,12) and adaptive p 8-bit approximate multipliers are presented in Table I.

The metric of the F-value is used and is defined as follows:

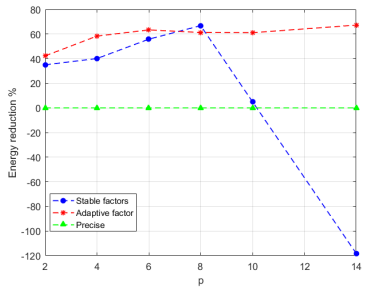
$$F = \frac{(1+\beta^2) * p * r}{\beta^2 * p + r} \quad (5)$$

where, p is precision, r is recall and β is used to balance the precision and the recall. It is usually set to be 1.

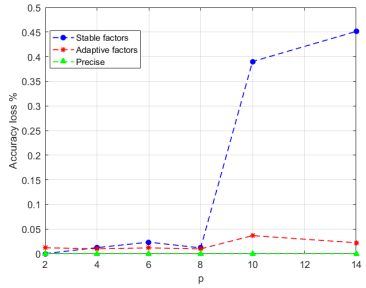
TABLE I
F-value of K-means Clustering

Data Sets	32b Float	8b Fixed	8b Fixed R4ABM		
			p=2	p=12	adaptive p
Iris	0.8067	0.8067	0.8201	0.7902	0.8103
Glass	0.4426	0.4426	0.4426	0.4021	0.4359

The results of Table I show that approximate computing can produce results with a small loss of accuracy; k-means clustering is sensitive to the initial centroids. Hence, an



(a)



(b)

Fig. 5: (a) Energy reduction and (b) accuracy loss at different values of approximate factor p.

adaptive approximate method can achieve a better accuracy (for example for the Iris data set with an adaptive p) than the accurate algorithm, because the error introduced by approximate computing avoids overfitting the initial centroids.

The energy consumption and the performance of an approximate multiplier have already been discussed in [13] using as metric the power-delay product (PDP). The total energy consumption can be estimated by adding all energies consumed by each multiplication along the data path. The use of an approximate multiplier is determined by the iterations and the number of samples; so, we can count the frequencies of occurrence of all 8-bit int weights in the entire path and add the PDPs of the corresponding approximate multipliers to find the total energy consumption. The results are presented in Table II.

TABLE II
ENERGY CONSUMPTION OF APPROXIMATE MULTIPLICATION.

Factor Type	Value of Factor	PDP (uJ)	Reduction (%)
Stable factors	2	0.0143	35.00%
	8	0.0073	66.82%
	14	0.048	-118.18%
Adaptive factors	2	0.0127	42.27%
	8	0.0085	61.36%
	14	0.0072	67.27%

Compared with the 8-bit precise design baseline of 0.022uJ, the proposed design reduces energy consumption by over 67% . However, the PDP could be very large if an inappropriate approximate factor (such as p=14) is utilized, thus contributing to the loss of convergence in the algorithm. Moreover, the proposed method achieves a small accuracy loss at the lowest energy consumption.

From Fig. 5, the performance of the stable approximate factor becomes worse when the factor is very large. However, the adaptive approximate factor can achieve a good energy reduction and a small accuracy loss in spite of the initial value of the approximate factor even when the initial approximate level is high.

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig. 6: Image segmentation: (a) original image; (b) original image with feature constrains; (c) accurate segmentation; approximate segmentation for (d) p=2; (e) p=8; (f) p=14; and approximate segmentation using the proposed method for (g) p=2; (h) p=8; (i) p=14.

V. CASE STUDY: IMAGE SEGMENTATION

In this section, the proposed method is applied to image segmentation. The semi-supervised feature constraints are chosen as more than 10 must-links and 10 cannot links. The semi-supervised k-means not only uses this information efficiently to segment the image, but it also dynamically calibrates the approximate levels for energy reduction and acceptable accuracy.

As shown in Fig. 6, the quality of the processed image deteriorates with an increase of p as factor during the segmentation process from (d) to (f). When this factor

is at the largest value of 14, the objective function loses convergence and the worst segmentation result is generated. The results from (g) to (h) are based on different initial factors; all achieve an acceptable segmentation result.

TABLE III
CHANGE OF P IN EACH ITERATION BASED ON INITIAL APPROXIMATE LEVEL DURING IMAGE SEGMENTATION

Initial Factors	Change of p in Each Iteration
p=2	2, 2, 2, 2, 4, 6, 8, 10, 8, 6, 4, 2, 2, 4
p=4	4, 4, 4, 6, 8, 10, 8, 6, 4, 2, 2, 2, 2, 2
p=6	6, 6, 8, 10, 8, 6, 4, 2, 2, 4, 2, 2, 4, 2, 2
p=8	8, 10, 12, 14, 14, 12, 10, 8, 6, 4, 2, 2, 2
p=10	10, 8, 10, 12, 14, 12, 10, 8, 6, 4, 2, 2, 4
p=12	12, 12, 10, 10, 8, 8, 10, 8, 6, 4, 2, 2, 4
p=14	14, 12, 10, 8, 6, 4, 2, 2, 2

Table III shows the change of p at each iteration based on different initial approximate levels during segmentation. The best initial factor is 14; also it is assumed that the error at the beginning is not as large as the errors accumulated along the data path. For a fast convergence, larger steps are always utilized and therefore, the early stage of computation of an algorithm should be robust to errors.

VI. CONCLUSION

A HW/SW co-design method for approximate k-means clustering has been investigated using multi-precision approximate multipliers with various approximate factors. A fixed-point quantization for floating-point data has been proposed to extend or compress the number range, thus resulting in resilience to the so-called approximate factor.

The energy consumed by the multiplier logic decreases rapidly at larger values of the factor p, while introducing a small loss in accuracy for the clustering results. Compared with an 8-bit precise baseline, the proposed method reduces by over 67% the energy consumption.

Adaptive approximate levels can contribute to both the application and the hardware utilization with supervised information that is a very small portion of the entire data sets. Some additional operations for calibrating the optimized approximate levels from only supervised information can be ignored in comparison with the large amount of unsupervised data that can too benefit from the optimized approximate levels. The proposed HW/SW co-design method for semi-supervised learning can be used for a number of machine learning and pattern recognition applications as will be investigated in future works.

REFERENCES

[1] H. Esmaeilzadeh, E. R. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *IEEE Micro*, vol. 32, no. 3, pp. 122–134, 2012.

[2] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. 50th Annual Design Automation Conference (DAC)*, 2013, pp. 113:1–113:9.

[3] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2016.

[4] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Computing approximately, and efficiently," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 748–751.

[5] A. Sampson, "Hardware and software for approximate computing," 2015.

[6] S. He, S. K. Lahiri, and Z. Rakamaric, "Verifying relative safety, accuracy, and termination for program approximations," *J. Autom. Reasoning*, vol. 60, no. 1, pp. 23–42, 2018. [Online]. Available: https://doi.org/10.1007/s10817-017-9421-9

[7] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2013.

[8] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. 49th Annual Design Automation Conference (DAC)*, 2012, pp. 820–825.

[9] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Computers*, vol. 64, no. 4, pp. 984–994, 2015.

[10] S. Lu, "Speeding up processing with approximation circuits," *IEEE Computer*, vol. 37, no. 3, pp. 67–73, 2004.

[11] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *IEEE Trans. VLSI Syst.*, vol. 8, no. 3, pp. 273–286, 2000.

[12] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–4.

[13] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 Booth multipliers for error-tolerant computing," *IEEE Trans. Computers*, vol. 66, no. 8, pp. 1435–1441, 2017.

[14] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. 24th International Conference on VLSI Design*, 2011, pp. 346–351.

[15] A. Sampson, J. Bornholt, and L. Ceze, "Hardware-software co-design: Not just a cliché," in *Proc. 1st Summit on Advances in Programming Languages (SNAPL)*, 2015, pp. 262–273.

[16] K. Guo, S. Han, S. Yao, Y. Wang, Y. Xie, and H. Yang, "Software-hardware codesign for efficient neural network acceleration," *IEEE Micro*, vol. 37, no. 2, pp. 18–25, 2017.

[17] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. 18th IEEE European Test Symposium (ETS)*, 2013, pp. 1–6.

[18] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," pp. 301–312, 2012.

[19] ——, "Neural acceleration for general-purpose approximate programs," *Commun. ACM*, vol. 58, no. 1, pp. 105–115, 2015.

[20] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, 2016.

[21] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *CoRR*, vol. abs/1702.03044, 2017. [Online]. Available: http://arxiv.org/abs/1702.03044

[22] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, "Dorefa-net: training low bitwidth convolutional neural networks with low bitwidth gradients," *CoRR*, vol. abs/1606.06160, 2016. [Online]. Available: http://arxiv.org/abs/1606.06160

[23] P. Merolla, R. Appuswamy, J. V. Arthur, S. K. Esser, and D. S. Modha, "Deep neural networks are robust to weight binarization and other non-linear distortions," *CoRR*, vol. abs/1606.01981, 2016. [Online]. Available: http://arxiv.org/abs/1606.01981

[24] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml