Energy Efficient SVM Classifier Using Approximate Computing

Yangcan Zhou, Jun Lin, and Zhongfeng Wang
School of Electronic Science and Engineering, Nanjing University, P.R. China
yczhou@smail.nju.edu.cn, {jlin, zfwang}@nju.edu.cn

***Abstract*—Approximate computing is a promising approach to reduce great power consumption in modern very large scale integration (VLSI) systems. In this paper, a novel approximate adder with a much simplified carry compensation scheme is proposed. Besides, an approximate fixed-width multiplier with a low-cost compensation unit is introduced. To evaluate the effectiveness of the proposed approximate computing units, the application of the approximate computing in support vector machine (SVM) is investigated. Simulation results show that the approximation errors have negligible impact on the classification accuracy. Under the TSMC 90nm CMOS technology, the synthesis results indicate that the SVM classifier using approximate computing can reduces the power-delay product (PDP), area and critical path delay by 32.4%, 18.7% and 16.0%, respectively, compared to the same classifier with accurate computation units.**

I. Introduction

Energy efficiency and processing speed have become the major concerns in the design of the modern very large scale integration (VLSI) systems. The requirement of low power and high speed has posed great challenges for circuit designers. Recently, a new design paradigm called approximate computing [1]–[3] has drawn much attention to achieve better energy efficiency and high speed at the same time . Approximate computing relies on the ability of many systems and applications to tolerate a certain degree of errors in computational results. By relaxing the fully precision, in one hand, approximate computing may not weaken the system capability significantly as long as the computed values are "good enough", on the other hand, it provides the opportunities to improve speed and energy efficiency with reduced complexity of hardware. Researchers have found various error-resilient applications in which approximate computing is applied successfully [4], such as neuromorphic systems [5] and multi-media systems [6]–[8].

Adder and multiplier are the most primary components in many digital signal processing (DSP) systems. The traditional ripple-carry adder (RCA) is too slow to meet the requirement of high speed. Many types of fast adders, such as carry-skip adder (CSA), carry-select adder (CSA), and carry-look-ahead adder (CLA) have been developed with great power consumption. In this regard, design of approximate adders for considerable energy saving while maintaining fast speed has attracted tremendous attention. In [9], a low power and area efficient approximate mirror adder in transistor level is proposed. In [5], an approximate adder with carry skip for neuromorphic system is proposed. A Lower-part-Or Adder (LOA) is proposed in [10], and a very similar one called Error-Tolerant Adder (ETA) is proposed in [6]. Approximate multipliers have also been extensively investigated by many researchers. In [11], a data-dependent truncation scheme is proposed. A low-error fixed-width approximate multiplier is proposed in [12]. An approximate multiplier with a compensation scheme based on statistical linear regression analysis is developed in [13].

In this paper, the application of approximate computing in SVM is investigated. The main contributions of this work are as follows:

- A novel approximate adder with a much simplified carry compensation scheme is proposed. Compared to the best accurate adder implemented by synthesis tool, the proposed one can reduce critical path delay, area and power-delay product (PDP) by 15.9%, 32.5% and 31.7%, respectively.
- An approximate fixed-width multiplier with a low-cost error compensation unit is proposed. The proposed multiplier can cut down critical path delay, area and PDP by 16.5%, 44.3% and 42.9%, respectively, compared to the accurate one generated by synthesis tool.
- To the best of our knowledge, the application of approximate adder and multiplier at the same time in SVM is discussed for the first time. The proposed approximate adders and multipliers are employed in an SVM classifier. Simulation results show that the approximation errors degrade the classification accuracy slightly based on the data set in [14]. Implemented under the TSMC 90nm CMOS technology, the approximate SVM classifier can reduce energy dissipation, area and critical path delay by 32.4%, 18.7% and 16.0%, respectively, compared to the original one.

II. Current Approximate Arithmetic Units

A. Approximate Adders

The truncation schemes have been proposed to reduce the critical path delay and power consumption of the accurate adders. Although there are many different designs, the basic idea is truncating carry propagation chain to prevent the carry signal rippling from less significant bits to more significant ones. As shown in Fig. 1, the simplest scheme is dividing an adder into two parts. In some complicated truncation schemes [6], an adder is divided into more than two parts, which leads to significant errors. Many carry compensation schemes have been developed to reduce error. A class of approximate adders using multiple less significant bits to speculate carry signal propagating to more significant bits are proposed in [5], [6], [8]. Lower-part-OR Adder (LOA) divides a normal adder into an accurate part for more significant bits and an inaccurate part for less significant bits [10]. OR

978-1-5090-6625-4/17/$31.00 ©2017 IEEE

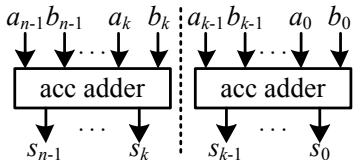

Fig. 1. The approximate adder truncated into two parts.

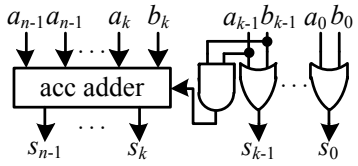

Fig. 2. The architecture of LOA.

gates are employed to generate the sum approximately in the inaccurate part as shown in Fig. 2. The approximate adder called error-tolerant adder (ETA) proposed in [6] is very similar to LOA.

B. Approximate Multipliers

Approximate fixed-width multipliers have been extensively investigated in the open literature. The basic idea of most of the approximate $n\times n$ fixed-width multipliers is removing n columns of the partial product array to reduce energy consumption and area overhead. Various compensation schemes [15], [16] are proposed to reduce the errors at the expense of extra hardware cost. A low-cost scheme called constant compensation were proposed in [16]. A very effective scheme keeping the most significant $n+w$ columns of partial product array was proposed in [17], where w is a nonnegative integer between 0 and $n-1$. As w increases, the errors become smaller and less hardware overhead is saved. A micromesh compensation scheme was developed in [15], which divides whole input space into several groups by statistical analysis, and different groups accompany with different compensation values.

III. Proposed Approximate Arithmetic Units

A. Proposed Approximate Adder

For an n-bit accurate adder with inputs A and B, we truncate it into two parts at k-th bit as shown in Fig. 1. Errors will occur if a carry is generated at truncated position. It is assumed that A and B are random numbers, The $Error\ Rate$ (ER) is a function of the parameter k which is given as follows:

$$\begin{aligned} ER(k) =& P_r(g_{k-1}=1)+P_r(p_{k-1}g_{k-2}=1) \\ &+\cdots+P_r((\prod_{j=1}^{k-1}p_j)g_0=1) \\ =& \frac{2^k-1}{2^{k+1}}, \end{aligned} \tag{1}$$

where $g_i=a_ib_i, p_i=a_i\oplus b_i$. As k increases, $ER(k)$ tends to approach 1/2. To reduce the magnitude of $ER(k)$, a_{k-1} is used as an approximate carry signal that propagates to k-th bit. Table I shows all combinations of approximate carry and accurate carry, where c_i is the carry out signal of i-th bit. According to Eq. (1), the probability of c_{k-2}=1 is $ER(k-1)$, so the probability of c_{k-2}=0 is $1-ER(k-1)$. Then the overall error rate of the proposed approximate adder is given by the following equation:

$$ER'(k)=\frac{1}{4}ER(k-1)+\frac{1}{4}[1-ER(k-1)]=\frac{1}{4}. \tag{2}$$

Compared to Eq. (1), the accuracy of the proposed approximate adder is significantly improved with almost no extra hardware cost.

TABLE I. Error analysis of the proposed approximate adder

a_{k-1}	0		0		1		1	
b_{k-1}	0		1		0		1	
c_{k-2}	0	1	0	1	0	1	0	1
accurate c_{k-1}	0	0	0	1	0	1	1	1
approximate c_{k-1}	0	0	0	0	1	1	1	1
error	0	0	0	-2^k	2^k	0	0	0

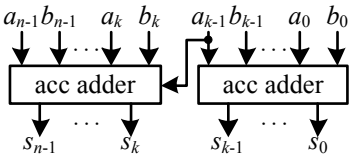

Fig. 3. The architecture of the proposed approximate adder.

The proposed approximate adder will generate two types of errors as shown in Table. I, called positive error and negative error, making the sum bigger and smaller, respectively. This kind of error pattern makes the proposed adder suitable for accumulation, during which positive and negative errors may cancel each other on large extent as long as the addends are close to uniform distribution.

B. Proposed Approximate Multiplier

Since full-width multipliers extend the data-width, fixed-width multipliers are more widely used in VLSI systems in order to reduce hardware overhead. A 6×6 fixed-width multiplier is shown in Fig. 4 as an example. A dash line divides the partial product array into an upper part and a lower part at column 6. The lower part should be reserved and participate in computation if accurate $P[11:6]$ is required. The lower part can be omitted in approximate computing to save hardware cost and improve energy efficiency with some errors. To improve the precision of the approximate multiplier, we propose a low-cost compensation unit as shown in Fig. 5, where $p_{i,j}$ stands for the j-th bit of i-th partial product. Every 3 bits of the compensation column in the rectangular box are packaged as the input of the compensation unit. In general, if there are 2 bits remainding, an extra AND gate is used to generate a carry as compensation. Otherwise, no extra gate is needed. Note that the proposed compensation scheme is an universal method, no matter how many bits the inputs are.

IV. SVM Classifier Using Approximate Computing

Developed from statistical learning theory, SVM is a well defined machine learning algorithm which has found many successful applications such as bio-informatics, text and image recognition, and so on. The major computation of the training and classification is the calculation of the kernel function. The

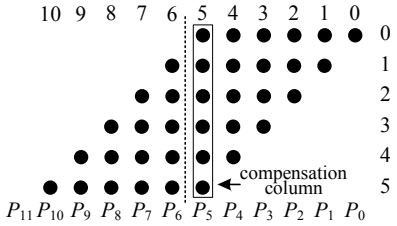

Fig. 4. The partial product array of 6×6 multiplier.

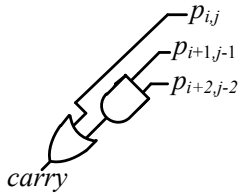

Fig. 5. Proposed compensation unit.

Gaussian kernel $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ in Eq. (3) is a very popular kernel function used in SVM especially when the dimension of the sample is not too large.

$$\begin{aligned} K(\boldsymbol{x}_i,\boldsymbol{x}_j) &= e^{-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{2\sigma^2}}, \\ \|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2 &= \sum_{k=0}^{n}(x_{i_k}-x_{j_k})^2, \end{aligned} \tag{3}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are two vectors, n is the dimension of the vectors, and σ is the width of Gaussian kernel. The hardware efficient CORDIC algorithm can be used to calculate the exponential function [18] with only addition and shift operation. The equation is given as follows:

$$\begin{aligned} x^{(i+1)} &= x^{(i)} + d_i y^{(i)} 2^{-i}, \\ y^{(i+1)} &= y^{(i)} + d_i x^{(i)} 2^{-i}, \\ z^{(i+1)} &= z^{(i)} - d_i \tanh^{-1}(2^{-i}), \\ d_i &= \text{sign}(z^{(i)}), \end{aligned} \tag{4}$$

where i is the iteration index. The support vector expansion used to classify the sample $\boldsymbol{x}$ is given as follows:

$$f(\boldsymbol{x}) = \sum_{i}^{m} \alpha_i y_i K(\boldsymbol{x},\boldsymbol{x}_i), \tag{5}$$

where $\boldsymbol{x}_i$ is the support vector, y_i is the label of $\boldsymbol{x}_i$, α_i is the Lagranian multiplier of $\boldsymbol{x}_i$, and m is the number of support vectors. The sign of the $f(\boldsymbol{x})$ indicates the category that $\boldsymbol{x}$ belongs to.

Based on the data set in [14], the coordinates of all samples are scaled to the range [-1,1], as shown in Fig. 6. For the fixed-point implementation, the classification accuracy under different width of the fractional part is summarized in Table II. Note that when the width of the fractional part is more than 6 bits, the classification accuracy increases slowly. So the coordinates of all samples are quantized to 6 bits for the fractional part with 1 bit for sign. The classification accuracy using the proposed approximate adder with different truncation schemes is shown in Table III. In this work, 6×6 fixed-width multipliers are used. The classification accuracy using the proposed approximate fixed-width multiplier is shown in Table IV, and another two approximate multipliers (APM1, APM2) [17] which omit the lower part directly are used as references for comparison. Based on the classification accuracy, A proposed approximate 6×6 fixed-width multiplier truncated at 6-th column and a proposed approximate 7-bit adder truncated at 3-th bit and used together to verify the overall classification accuracy, simulation result shows that the classifier can still identify the majority of samples as shown in Fig. 6.

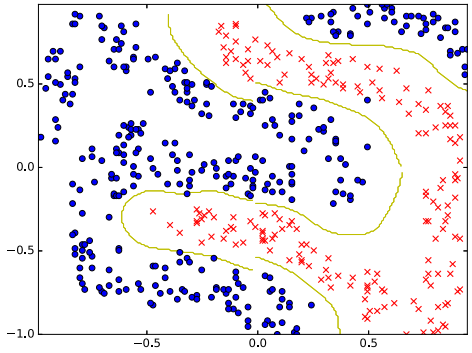

Fig. 6. Samples in data set [14].

TABLE II. CLASSIFICATION ACCURACY UNDER DIFFERENT WIDTH OF FRACTIONAL PART

width	4	5	6	7	float point
accuracy(%)	90.5700	93.7060	97.3913	98.2801	99.1304

TABLE III. CLASSIFICATION ACCURACY USING THE PROPOSED APPROXIMATE ADDER

k*	1	2	3	4
accuracy(%)	97.2862	97.1594	96.0869	86.9565

* k indicates the truncated position of the proposed adder

TABLE IV. CLASSIFICATION ACCURACY USING APPROXIMATE MULTIPLIER

	proposed	APM1	proposed	APM2
k*	6	5	5	4
accuracy(%)	96.6812	97.4014	97.5426	97.9790

* k indicates the truncated position of the partial product array.

V. HARDWARE IMPLEMENTATION AND COMPARISON

The computation units discussed below are all implemented under the TSMC 90nm CMOS technology. The proposed approximate 6×6 fixed-width multiplier truncated at the 6-th column and an accurate one are implemented. The synthesis results are summarized in Table V. A proposed approximate 7-bit adder truncated at 3-th bit and an accurate one are also implemented. The synthesis results are shown in Table VI.

The hardware architecture of the SVM classifier is shown in Fig. 7. The computation units in gray background are replaced by the proposed approximate ones. For the implementation of the SVM classifier, the proposed approximate adders with $k = 3$ and multipliers with $k = 6$ are used. As shown in Table VII, synthesis results demonstrate that the SVM classifier using approximate computing can reduce the PDP, area and

TABLE V. COMPARISON OF PROPOSED AND ACCURATE MULTIPLIER

	Power* (mw)	Delay (ns)	PDP (pJ)	Area (μm^2)
Accurate	0.6780	0.412	0.280	1408.4
Proposed	0.4653	0.344	0.160	784.6
gain (%)	-	16.5	42.9	44.3

* All the synthesis results of our designs in this paper are from synthesis report generated by the Synopsys Design Compiler.

TABLE VI. COMPARISON OF PROPOSED AND ACCURATE ADDER

	Power (mw)	Delay (ns)	PDP (pJ)	Area (μm^2)
Accurate	0.2828	0.145	0.041	297.8
Proposed	0.2317	0.122	0.028	201.1
gain (%)	-	15.9	31.7	32.5

critical path delay by 32.4%, 18.7% and 16.0%, respectively, with only 2.0942% accuracy loss in classification.

TABLE VII. COMPARISON OF ACCURATE AND APPROXIMATE SVM CLASSIFIER

	Power (mW)	Delay (ns)	PDP (pJ)	Area (um^2)	Classify Accuracy (%)
accurate	2.1583	2.715	5.86	23147	97.3913
approximate	1.7353	2.280	3.96	18826	95.2971
gain (%)	-	16.0	32.4	18.7	-2.0942

VI. CONCLUSION

In this paper, a novel approximate adder with a much simplified carry compensation scheme is proposed. Additionally, an approximate fixed-width multiplier with a low-cost compensation unit is proposed. To the best of our knowledge, the application of approximate computing in SVM is investigated for the first time. Under the TSMC 90nm CMOS technology, the synthesis results have shown that the SVM classifier utilizing approximate computing can reduce critical path delay, area and PDP by 16.0%, 18.7% and 32.4%, respectively, with negligible classification accuracy loss. This work has proved that the application of approximate computing in SVM is feasible. Our future work will focus on the construction of more efficient approximate computing units and the utilization of approximate computing in different applications.

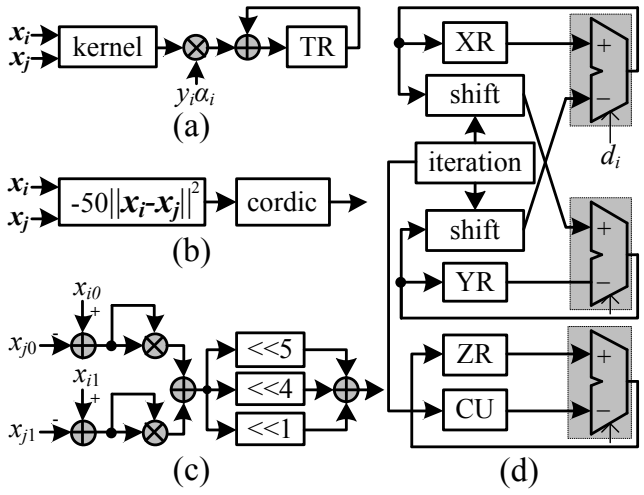

Fig. 7. The architecture of the SVM classifier. (a) Top architecture. (b) The architecture of the Gaussian Kernel. (c) The architecture of the norm square of two vectors. (d) The architecture of The CORDIC.

REFERENCES

[1] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar, "Scalable effort hardware design: exploiting algorithmic resilience for energy efficiency," in *47th ACM/IEEE Design Automation Conference (DAC)*. June 2010, pp. 555–560, IEEE.

[2] Q. Xu, T.K. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design & Test*, 2016.

[3] R. Venkatesan, A. Agarwal, and K. Roy, "Macaco: Modeling and analysis of circuits for approximate computing," in *Computer-Aided Design (ICCAD)*, Nov. 2011.

[4] H. Kakavand and A. El. Gamal, "On energy-reliability tradeoff in analog-to-digital converters with imperfect comparators," in *IEEE 40th Annual Conference on Information Sciences and Systems*, 2006, pp. 1366 – 1371.

[5] Y Kim, Y Zhang, and P Li, "Energy efficient approximate arithmetic for error resilient neuromorphic computing," *IEEE Transactions on VLSI Systems*, pp. 2733 – 2737, November 2014.

[6] Z. Ning, L. G. Wang, W. Gang, and K. S. Yeo, "Enhanced low-power high-speed adder for error-tolerant application," in *SoC Design Conference (ISOCC)*, November 2010, pp. 323 – 327.

[7] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, pp. 124–137, Jan. 2013.

[8] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Design Automation Conference (DAC)*, Jan. 2012.

[9] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: imprecise adders for low-power approximate computing," in *International Symposium on Low Power Electronics and Design (ISLPED)*, 2011, pp. 409 – 414.

[10] H. R. Mahdian, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, pp. 850 – 862, April 2010.

[11] E. J. King and E. E. Swartzlander, "Data-dependent truncation scheme for parallel multipliers," in *Thirty-First Asilomar Conference on Signals, Systems & Computers*, November 1997, pp. 1178 – 1182.

[12] J. M. Jou and S. R. Kuang, "Design of low-error fixed-width multiplier for dsp applications," *Electronics Letters*, vol. 33, pp. 1597 – 1598, September 1997.

[13] S. J. Jou, M. H. Tsai, and Y. L. Tsao, "Low-error reduced-width booth multipliers for dsp applications," *IEEE Transactions on Circuits and Systems I*, vol. 50, pp. 1470 – 1474, November 2003.

[14] C. hang, C. Chung, and C. J. Lin, "Libsvm: a library for support vector machines," in *ACM*, http://www.csie.ntu.edu.tw/ cjlin/cjlin/libsvmtools/datasets/binary/fourclasss-scale, 2011.

[15] B. Shao and P. Li, "Array-based approximate arithmetic computing: a general model and applications to multiplier and squarer design," vol. 62, pp. 1081 – 1090, April 2015.

[16] M. J. Schulte and E. E. Swartzlander, "Truncated multiplication with correction constant," *VLSI Signal Processing*, pp. 388 – 396, October 1993.

[17] M. A. Song, L. D. Van, and S. Y. Kuo, "Adaptive low-error fixed-width booth multipliers," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, pp. 1180–1187, June 2007.

[18] J. S. Walther, "The story of unified cordic," *Journal of VLSI Signal Processing Systems - special issue on CORDIC*, vol. 25, pp. 107–112, June 2000.