A Systematic Method for Approximate Circuit Design Using Feature Selection

Ling Qiu and Yingjie Lao
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29634
Email: {lingq, ylao}@clemson.edu

***Abstract*—As the size of technology reaches deep nanometer realm, the improvements in area, power, and timing resulting from developments in scaling have started to see a decrease. Alternative approaches to explore design space to achieve energy-efficient digital systems are of great interest in recent years. Approximate computing in hardware design has emerged as a promising paradigm which seeks to trade off the requirement of accuracy for reduction in power consumption and hardware cost. This paper presents a systematic and scalable method for approximate circuit design by employing data-driven feature selection techniques rather than using statistical or theoretical analysis, which is extremely suitable for applications at a larger scale. A case study on approximate multiplier is presented to demonstrate the proposed design flow. Our experimental results show that the proposed approach could achieve better area/power saving and comparable error performance with other existing manual approximate multiplier designs, while greatly reducing the design workload and complexity.**

I. Introduction

Approximate computing is well suited for applications where an approximate result is sufficient or the errors can be tolerated internally and are not perceivable by the end user. Many modern computational tasks, such as applications in machine learning, recognition and data mining, exhibit algorithmic error resilience [1]. In addition, with rising performance demands confronting with plateauing resource budgets, approximate computing has become, not merely attractive, but even imperative.

The main research focuses of approximate computing at the logic level have been the basic arithmetic elements of a processor: addition and multiplication [2]–[9]. In most of these prior works, the approximate versions of arithmetic elements are manually designed to exploit the trade-off space to achieve the best performance. However, these manual design strategies become infeasible when it comes to a complex system. The straightforward implementation for constructing a large energy-efficient system by directly employing approximate arithmetic elements may be inefficient or erroneous. Therefore, it is crucial to develop general and scalable design techniques for approximate logic circuits.

In this work, we consider the design of error compensation block in approximate logic circuits. Error compensation is a commonly used scheme in approximate computing, which is aimed at correcting the errors generated from logic simplification by using a small number of logic gates. For example, in the context of a truncated approximate multiplier, the total area and power can be significantly reduced by completely eliminating lower bits in the multiplier and compensating the errors using a few logic gates as the error compensation block [7]. Similar concepts are also employed in precision over-scaling and voltage over-scaling [10] applications, where the compensation blocks (e.g., statistical error compensation [11]) are also usually designed either theoretically or empirically according to the logic of the algorithm and the error distribution.

With the advent of machine learning and data mining algorithms, it is also plausible to utilize these algorithms to train a compensation block. In this paper, we introduce the idea of designing approximate computing circuits by using data-driven techniques to reduce design workload and complexity. We propose a novel systematic and scalable method of using feature selections methods to *redesign* the compensation blocks in approximate circuits. Comparing to the previous works [4]–[9], [12], [13], the proposed approach skips all the complicated theoretical analysis steps and solely looks into the input and error patterns to design the error compensation circuit efficiently. Area/power saving and error are the two most important performance metrics to evaluate an approximate circuit design. In this paper, we use mean error and mean squared error as the error metrics. We illustrate the proposed design flow on a case study of a radix-4 modified Booth multiplier, which is one of the most popular schemes for signed multiplication [3].

The rest of the paper is organized as follows: Section II briefly reviews the prior related work. The general methodology for compensation circuit design using feature selection is presented in Section III. In Section IV, we present a case study to show the steps of data-driven approximate circuit design flow. Experimental results are presented in Section V to verify the effectiveness of the proposed method.

II. Related Work

A. Approximate Computing

In the literature, various designs for approximate arithmetic elements have been developed [3]–[9], [12]–[14]. The designs of approximate multipliers can be broadly classified into two categories: truncated and non-truncated schemes [3]. Note that a fixed-width multiplier can also be considered as a type of approximate multiplier, where the output bit-length is the same as the input bit-length. In a typical $n \times n$ truncated multiplier, an error compensation block is introduced to reduce the truncation errors that are generated by removing the logic cells to compute the least significant n product bits. One approach to compensate the error is to add a constant error-compensation bias [13]. This methods leads to very simple logic and small implementation cost; however, the compensated error is still considerably large as the bias cannot adjust according to the input signals. An improved approach is to add an adaptive error-compensation bias to the retained adder cells. In [4], the Booth encoded inputs are transformed into a new set of variables which are then used to design the compensation circuit. In [12], an adaptive conditional-probability estimator is proposed to compensate the fixed-width Booth multiplier error. In [9], the sign bit of the Booth encoded multiplier is applied to conditional probability to generate compensation values. For the non-truncated scheme, the approximate design is achieved by altering internal components [3]. For example, an approximate 2×2 multiplier is utilized as the building block to construct a larger multiplier to elevate computation efficiency in [14]. In [3], the Booth encoders are approximated to a simpler logic to reduce power consumption.

Besides, error compensation block design for a number of other applications have also been studied [10], [11], [15], [16]. For example, algorithmic noise-tolerance (ANT) specifically targeting at DSP circuits was introduced to compensate the error due to voltage over-scaling in [10]. A method for generating approximate circuit with minimum area under a given rate-significance requirement was also proposed in [15].

Unlike these prior works, our proposed method is based on data-driven feature selection methods, which, to the best of our knowledge, has never been exploited before.

B. Feature Selection

Feature selection is the process of selecting a subset of relevant features to reduce the size of the structure without significantly decreasing the accuracy of the computation, which is widely-used in machine learning and statistics. In some of machine learning tasks, feature selection can even improve the prediction accuracy by eliminating irrelevant features. Feature selection can be mainly categorized into three general methods [17]: filter methods, wrapper methods and embedded methods.

In an approximate circuit, the error is dependent on the input combinations. In addition, the size of the compensation block is also proportional to the number of inputs to this block. As a result, the quantity and quality of the inputs used for error compensation is critical to the performance of approximate logic circuit design. In general, better performance can be achieved by using only a few but highly correlated inputs to design the error compensation block. Therefore, feature selection techniques, especially classifier-independent feature selection algorithms, are well suited in finding such correlation by treating the input bits as binary features and errors as corresponding class labels.

In this paper, we use the χ^2 feature selection, which is a commonly used univariate feature selection approach, to detect the most informative features. Univariate feature selection examines each feature separately to determine the degree of correlation of the feature with the given response labels. The χ^2 algorithm can automatically discretize the continuous attributes and removes irrelevant attributes based on the χ^2 statistic and the inconsistency found in the data [18]. Note that other feature selection methods can also be applied to this task.

III. Methodology

The overall design flow of the proposed methodology for designing approximate logic circuit using feature selection is shown in Fig. 1, which consists of five steps.

We use the ideal output, truncated output and compensated output to represent the outputs of the original circuit, the pre-compensation approximate circuit (PCAC), and the compensated approximate circuit (CAC), respectively. The final circuit, CAC, essentially consists of the PCAC and the designed compensation circuit.

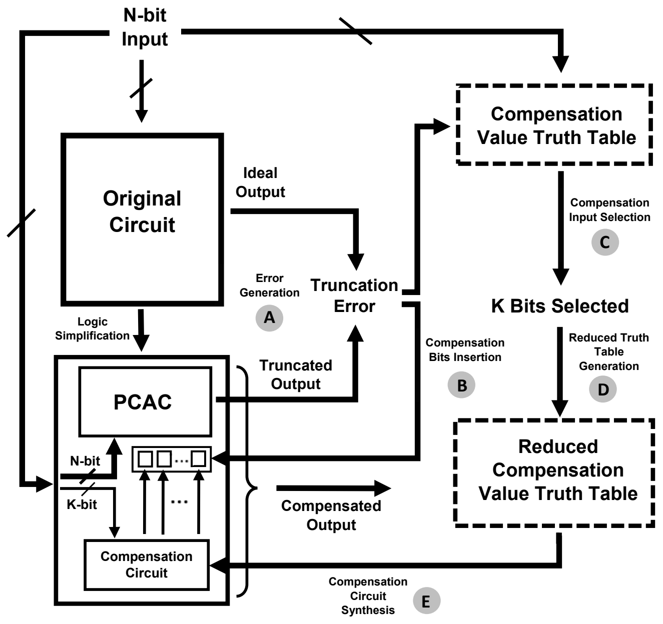
N-bit Input
Compensation Value Truth Table
Original Circuit
Ideal Output
Compensation Input Selection
C
Error Generation
A
Truncation Error
K Bits Selected
Logic Simplification
Compensation Bits Insertion
B
Reduced Truth Table Generation
D
Truncated Output
PCAC
Reduced Compensation Value Truth Table
N-bit
K-bit
Compensated Output
Compensation Circuit
Compensation Circuit Synthesis
E

Fig. 1. Compensation Circuit Design Flow.

A. Error Generation

The first step is to compute the pre-compensation error of the approximate circuit. The errors are generated due to the logic simplification of the original circuit or the timing errors in voltage over-scaling applications. If we consider a truncated scheme in approximate circuit design, errors are basically calculated by taking the differences between the ideal output and truncated output without adding error compensation, i.e., the output of PCAC, as shown in Fig. 1. Choosing the appropriate methods to provide the best simplification opportunity is another important research topic in approximate computing, which is peripheral to this paper. Intuitively, if the truncation in a truncated approximate circuit is too large, it would be difficult to implement a simple compensation circuit to offset the errors; on the other hand, if the truncation is too small, the area/power saving would be minimal. In this paper, we consider the error patterns are given and focus on the design of the error compensation block. Future work will be directed towards using machine learning algorithms to automatically determine the appropriate logic simplification without significantly compromising the accuracy of the computation.

B. Compensation Bits Insertion

We insert compensation bits into the approximate circuit at the second stage. After logic simplification, there are 2^N input-error pairs for an approximate circuit with an N-bit input, if every input bit is independent. We determine the number and the locations of compensation bits to insert based on the error distribution. Generally, at least $\lceil log_2 R_e \rceil$ compensation bits are required for an error dynamic range of R_e. However, since the errors are usually distributed unevenly across the dynamic range, it is possible to reduce the number of compensation bits to $\lfloor log_2 R_e \rfloor$ and approximate the $R_e - 2^{\lfloor log_2 R_e \rfloor}$ least frequent error values to the nearest value in the reduced dynamic range, which can be considered as another step of logic simplification. For example, most truncation-based logic simplification methods lead to all positive errors and tend to have less probability for larger magnitude errors. Therefore, we can assign compensation value cp_i to each input-error pair according to Equation (1), where the threshold T is equal to $2^{\lfloor log_2 R_e \rfloor} - 1$.

$$cp_i = \begin{cases} e_i & e_i \leq T \\ T & e_i > T \end{cases} \tag{1}$$

Consequently, the overall complexity of the compensation block is also reduced, while the errors would increase only slightly. In this paper, we construct the compensation bits in different bit positions, i.e., N compensation bits could generate 2^N values. However, it is important to note that it is not necessary to always put all compensation bits in distinct bit positions. For example, if the error only ranges from 0 to 2, we can put two compensation bits both in the LSB of the circuit, which may result in better compensation block design compared to inserting the bits into two consecutive positions.

C. Compensation Input Selection

We then apply feature selection algorithms to identify the inputs that are most correlated with the compensation values. We use the χ^2 univariate feature selection to select the inputs for eahich compensation bit. The total number of compensation input bits K should be picked based on the accuracy requirements or the error constraints. We can either select the inputs for all compensation bits simultaneously based on the overall feature selection rankings and scores or use logic synthesis tools to optimize the multiple-output combinational error compensation circuit automatically. The more features selected, the more accurate result the circuit will generate, however, the hardware cost and the power consumption of the error compensation block will be higher.

In this paper, we apply feature selection directly on the primary inputs of a logic circuit. Ongoing work includes the investigation of using feature extraction algorithms to generate higher-order features as the inputs to the compensation blocks, which might be more informative and non-redundant and hence facilitate the subsequent compensation block synthesis steps to produce better performance. In fact, the Booth encoded inputs can be considered as higher-order features for designing an approximate multiplier.

D. Reduced Truth Table Generation

After we determined the highest correlated K inputs by using feature selection, the truth table of the input and the compensation value will be reduced from 2^N rows to 2^K rows. Thus, each row in the reduced truth table corresponds to 2^{N-K} rows in the original truth table. The compensation value cp_j for each row in the reduced truth table should be selected such that the overall error for this row is minimized, as expressed in Equation (2):

$$cp_j = \arg\min_{cp_j}(\sum_{i \in R_e} ef(cp_i, cp_j) \times p_i), \tag{2}$$

where p_i is the percentage appearance of cp_i in the corresponding 2^{N-K} rows of the original truth table and $ef()$ represents the error metric. For example, if we use the mean squared error as the error metric, Equation (2) can be reduced to

$$cp_j = \arg\min_{cp_j}(\sum_{i \in R_e} (cp_i - cp_j)^2 \times p_i) \tag{3}$$

We repeat this for all 2^K reduced input combinations to generate the complete reduced truth table. When K is large (i.e., 2^{N-K} is small), it is possible to have multiple cp_j with the same minimum value of Equation (2) for certain inputs. In this case, we can assign don't care terms to these rows in the reduced truth table, which could help to minimize the logic in the final step.

E. Compensation Circuit Synthesis

The last stage is to use logic synthesis tool to optimize the combinational error compensation circuit that is specified by the reduced truth table from the previous step. State-of-the-art approximate logic synthesis (ALS) algorithms [19]–[21] can also be applied on top of the error compensation circuit to further improve the hardware implementation efficiency.

IV. Case Study and Experimental Results

Our proposed methodology will be extremely suitable for large and complex approximate circuit designs that are not feasible for manual analysis or simplifications, as only the input and error patterns are required for the design flow described in Section III. In this section, we apply our methodology to design a relatively small but well-studied circuit (i.e., an approximate multiplier) for the purposes of demonstration and comparison.

A. Experimental Setup

In our experiment, we use the Scikit-learn toolkit to perform the χ^2 univariate feature selection, which is an open source machine learning library in Python that includes a wide range of machine learning tools [22]. All the circuits are synthesized using Synopsys Design Compiler and mapped to a 32 nm standard cell library to evaluate the performance of their hardware implementations.

B. Approximate Multiplier Architectures

We employ the proposed methodology on a 10-bit fixed-width multiplier and use the radix-4 modified Booth encoding scheme to reduce the number of partial products. We consider a truncated scheme where the adder cells for calculating the 9 LSBs are deleted from the original circuit, as shown in Fig. 2(a). The corresponding truncation error distribution is shown in Fig. 2(b). It can be seen that the dominant errors are 1 and 2, and the maximum error is 3. Therefore, we insert two compensation bits, λ_1 and λ_0, to the 2 LSBs of the PCAC output product. The two compensation bits can be easily integrated into the Wallace tree or Dadda tree architecture. Next, we use feature selection to rank the inputs. We select 4, 6, 8, 10 features respectively in our experiment and then generate the reduced truth table based on the selected inputs accordingly.

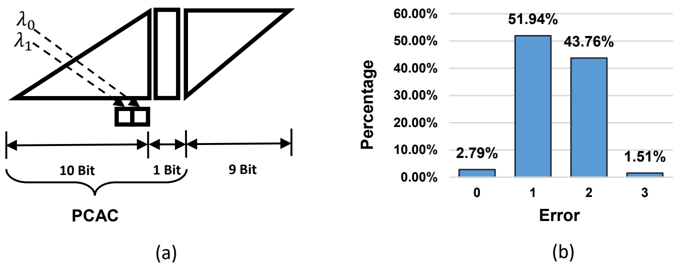

Fig. 2. 10-Bit Fixed-Width Approximate Multiplier: (a) Truncation and Bit Insertion Scheme, (b) Error Distribution after Truncation

C. Experimental Results

We use the mean error ε_{mean} and mean squared error ε_{mse} to evaluate the arithmetic performances of different approximate multiplier designs, which are calculated by:

$$\varepsilon_{mean} = mean(P_{oc} - P_{cac})/2^n, \tag{4}$$
$$\varepsilon_{mse} = mean(P_{oc} - P_{cac})^2/2^n, \tag{5}$$

where P_{oc} and P_{cac} represent the output products of the original booth modified multiplier and the compensated approximate booth modified multiplier, respectively. The mean error is an important metric for approximate multiplier design which captures the performance of systems that consist of a large number of multipliers, especially for multimedia and DSP applications where the final output results are usually accumulated by a series of multiplication products.

TABLE I. Comparison of the Error Performance of Different Fixed-Width Multipliers

multiplier	ε_{mean}	ε_{mse}
PCAC	1.4375×2^{-9}	2.3166×2^{-18}
[7]	-0.0039×2^{-9}	0.1542×2^{-18}
[8]	0.0689×2^{-9}	0.1544×2^{-18}
[9]	-0.0039×2^{-9}	0.1498×2^{-18}
proposed (K = 4)	0.0000×2^{-9}	0.2714×2^{-18}
proposed (K = 6)	0.0313×2^{-9}	0.2549×2^{-18}
proposed (K = 8)	0.0078×2^{-9}	0.2394×2^{-18}
proposed (K = 10)	0.0098×2^{-9}	0.2287×2^{-18}

TABLE II. Normalized Area and Power Consumptions of Different Fixed-width Multipliers

multiplier	$NormArea$	$NormPower$
Original Circuit	100%	100%
[7]	66.95%	64.91%
[8]	65.45%	64.31%
[9]	66.10%	63.66%
proposed (K = 4)	59.97%	60.14%
proposed (K = 6)	61.64%	60.79%
proposed (K = 8)	65.30%	62.69%
proposed (K = 10)	82.56%	68.46%

The performances are summarized in Table I and Table II, along with comparisons to existing approximate fixed-width multiplier designs [7]–[9]. The area and power consumptions are normalized to the original circuit It can be seen that the error can be significantly reduced from PCAC by using the proposed methodology, while achieving 30% to 40% saving in power consumption of the original circuit when we select 4, 6, 8, or 10 input bits for designing the compensation circuit. When we use more features, error is decreased, while the area/power consumption will be increased. For example, when we increase the number of input bits K from 4 to 10, the mean squared error is decreased by 15.7%; however, the area consumption is increased significantly, i.e., from 59.97% to 82.56% of the original circuit. In this particular case of approximate multiplier, $K = 4$ already achieves a very good performance, which only involves a very simple combinational circuit. The mean error is almost zero when $K = 4$.

Compared to existing approximate fixed-width multiplier designs, the proposed methodology generally leads to slightly worse mean squared error than the designs in [7]–[9] but comparable mean error, while achieving better performance in area/power reduction for K from 4 to 8. Further more, the main advantage of the proposed methodology is that it significantly reduces the design complexity and provides a large design space of approximate circuit architectures with different values K. The number of features used for designing the compensation circuit needs to be selected based on the specific application requirement. Besides, it is important to note that compared to other existing manual design techniques, it is much easier to apply the proposed methodology on large-scale systems.

References

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

[2] A. B. Kahng and S. Kang, “Accuracy-configurable adder for approximate arithmetic designs,” in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 820–825.

[3] 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 Transactions on Computers*, vol. 66, no. 8, pp. 1435–1441, Aug 2017.

[4] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, “Design of low-error fixed-width modified booth multiplier,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 5, pp. 522–531, May 2004.

[5] 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*, vol. 90, no. 6, pp. 1180–1187, 2007.

[6] 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: Fundamental Theory and Applications*, vol. 50, no. 11, pp. 1470–1474, Nov 2003.

[7] J. P. Wang, S. R. Kuang, and S. C. Liang, “High-accuracy fixed-width modified booth multipliers for lossy applications,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 1, pp. 52–60, Jan 2011.

[8] W.-Q. He, Y.-H. Chen, and S.-J. Jou, “High-accuracy fixed-width booth multipliers based on probability and simulation,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 8, pp. 2052–2061, 2015.

[9] Z. Zhang and Y. He, “A low error energy-efficient fixed-width booth multiplier with sign-digit-based conditional probability estimation,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2017.

[10] R. Hegde and N. R. Shanbhag, “Energy-efficient signal processing via algorithmic noise-tolerance,” in *Proceedings of 1999 International Symposium on Low Power Electronics and Design*, Aug 1999, pp. 30–35.

[11] E. P. Kim and N. R. Shanbhag, “Energy-efficient accelerator architecture for stereo image matching using approximate computing and statistical error compensation,” in *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec 2014, pp. 55–59.

[12] Y. H. Chen and T. Y. Chang, “A high-accuracy adaptive conditional-probability estimator for fixed-width booth multipliers,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 3, pp. 594–603, March 2012.

[13] S. S. Kidambi, F. El-Guibaly, and A. Antoniou, “Area-efficient multipliers for digital signal processing applications,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 2, pp. 90–95, Feb 1996.

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

[15] D. Shin and S. K. Gupta, “A new circuit simplification method for error tolerant applications,” in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[16] B. Shim, S. R. Sridhara, and N. R. Shanbhag, “Reliable low-power digital signal processing via reduced precision redundancy,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 5, pp. 497–510, May 2004.

[17] Z. M. Hira and D. F. Gillies, “A review of feature selection and feature extraction methods applied on microarray data,” *Advances in bioinformatics*, vol. 2015, 2015.

[18] H. Liu, R. Setiono *et al.*, “A probabilistic approach to feature selection-a filter solution,” in *International Conference on Machine Learning*, vol. 96, 1996, pp. 319–327.

[19] Y. Wu and W. Qian, “An efficient method for multi-level approximate logic synthesis under error rate constraint,” in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 128.

[20] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, “Salsa: systematic logic synthesis of approximate circuits,” in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 796–801.

[21] J. Miao, A. Gerstlauer, and M. Orshansky, “Multi-level approximate logic synthesis under general error constraints,” in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2014, pp. 504–510.

[22] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, “API design for machine learning software: experiences from the scikit-learn project,” in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.