EvoApproxLib: Extended Library of Approximate Arithmetic Circuits

Vojtech Mrazek, Zdenek Vasicek, Lukas Sekanina
Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno University of Technology
Brno, Czech Republic
{mrazek,vasicek,sekanina}@fit.vutbr.cz

***Abstract*—Approximate circuits and approximate circuit design methodologies attracted a significant attention of researchers as well as industry in recent years. In order to accelerate the approximate circuit and system design process and to support a fair benchmarking of circuit approximation methods, we proposed a library of 8-bit approximate adders and multipliers called EvoApprox8b. The library was extended by thousands of new designs with bit-width from 8 to 128 that form Pareto fronts with respect to several error metrics. The library provides Verilog, Python, Matlab and C models of all approximate circuits. A subset of circuits selected from EvoApproxLib library is available at: https://ehw.fit.vutbr.cz/evoapproxlib.**

***Index Terms*—Approximate computing, VLSI, Library, Benchmarking**

I. Introduction

Despite the rapid developments in the very-large-scale integration (VLSI) circuit technologies and in modern circuit design techniques, the overall energy consumption of integrated circuits is rapidly growing mainly due to their increasing complexity needed in current computing systems. At the same time, many computationally intensive applications, e.g. image recognition, signal processing and data mining, are widely implemented in these systems. Moreover, the expansion of modern battery-powered and smart devices such as mobile systems, IoT nodes and wearable electronics emphasizes the need for low-power implementations.

Fortunately, many of the computationally intensive applications feature an intrinsic error-resilience property [1]. Since they often process noisy or redundant data and their users are willing to accept certain errors in many cases, an emerging paradigm, the so-called *approximate computing*, is now employed in the design of the energy-efficient implementations. At the circuit level, approximations (i.e. circuit simplifications) are intentionally introduced to find a good trade-off between power consumption, performance and error.

The approximations can be introduced to the circuit in various steps of the standard circuit design flow. In this work, we primarily focus on the technology independent logic synthesis step. The approximations introduced in this step, the so-called *functional approximations*, modify the Boolean function of the circuit. It has one important advantage – the approximate circuit can be implemented in arbitrary ASIC as well as FPGA technology, because it is assumed that the technology dependent synthesis is performed by means of some well-optimized open source or commercial tools after the approximation is conducted.

The methods introduced for the functional approximations can be divided into two categories: (1) manual, and (2) automated. The manual (ad-hoc) methods are developed for a specific circuit component such as adders and multipliers [2], [3]. On the other hand, the automated methods use some general-purpose circuit simplification, resynthesis and approximation techniques and enable us to approximate arbitrary circuits. These methods start with an original (exact) circuit and, typically iteratively, modify its structure.

However, the functional approximation of complex circuits is a time-consuming process. As many of these circuits contain common arithmetic components (circuits) such as adders and multipliers, they can be approximated by replacing selected components by their approximate implementations available in a suitable library. Hence, we proposed a comprehensive library of approximate arithmetic circuits two years ago [4]. These circuits were designed by means an automated approximation algorithm described in Section II. In this paper, we report an extended version of the library which has been developed according to the needs of the design community and with the help of design techniques that were available after introducing the first version of the library. The extended version of the library is presented in Section III.

II. Automated construction of approximate arithmetic circuits

The method used to obtain the library follows the methodology introduced in [5]. It is a general-purpose approximation method for combinational circuits based on Cartesian Genetic Programming (CGP). CGP represents candidate circuits as directed acyclic graphs and iteratively modifies the circuit to reach design objectives while ensuring that various constraints (e.g. the error below a given threshold) are not violated.

A. Errors of approximate circuits

The quality of approximate combinational circuits is typically expressed using one or several error metrics. In addition to the error rate (ER), the average-case as well as the worst-case situation can be analyzed. Among others, the mean absolute error (MAE) and the mean relative error (MRE) are the most useful metrics that are based on the average-case

analysis. Moreover, the mean square error (MSE) is typically analyzed because it is important for PSNR calculation. Selection of the right metrics is a key step of the whole design.

The error metrics for an approximate circuit O_{approx} with respect to the accurate circuit O_{orig} are defined as follows. Note that as n_i is the number of primary inputs, the operand's width is $n_i/2$ bits and $\forall i$ is enumeration of all possible input vectors.

$$\text{ER} = \frac{\sum_{\forall i: O^{(i)}_{\text{approx}} \neq O^{(i)}_{\text{orig}}} 1}{2^{n_\text{i}}}, \tag{1}$$

$$\text{MAE} = \frac{\sum_{\forall i} \left| O^{(i)}_{\text{approx}} - O^{(i)}_{\text{orig}} \right|}{2^{n_\text{i}}}, \tag{2}$$

$$\text{MSE} = \frac{\sum_{\forall i} \left| O^{(i)}_{\text{approx}} - O^{(i)}_{\text{orig}} \right|^2}{2^{n_\text{i}}}, \tag{3}$$

$$\text{MRE} = \frac{\sum_{\forall i} \frac{\left| O^{(i)}_{\text{approx}} - O^{(i)}_{\text{orig}} \right|}{\max(1, O^{(i)}_{\text{orig}})}}{2^{n_\text{i}}}, \tag{4}$$

$$\text{WCE} = \max_{\forall i} \left| O^{(i)}_{\text{approx}} - O^{(i)}_{\text{orig}} \right|, \tag{5}$$

$$\text{WCRE} = \max_{\forall i} \frac{\left| O^{(i)}_{\text{approx}} - O^{(i)}_{\text{orig}} \right|}{\max(1, O^{(i)}_{\text{orig}})}. \tag{6}$$

B. Cartesian genetic programming

Cartesian genetic programming (CGP) has grown from a method of evolving digital circuits. CGP especially differs from other genetic programming branches in (i) the solution representation and (ii) the search mechanism.

1) Representation: The key part of CGP is the representation of candidate circuits. A candidate circuit is represented as an integer netlist describing a constant number of components (N). These components (nodes) are organized in a two-dimensional grid of n_c columns and n_r rows ($N = n_c \cdot n_r$). The number of primary inputs and outputs of the circuit is denoted n_i and n_o. The function of the nodes depends on the level of abstraction used in modeling, where logic gates and more complex components from the technology library can naturally be utilized. Every component has up to n_a inputs and n_b outputs. For example, if standard logic gates are used as components, $n_a = 2$ and $n_b = 1$. An example of a circuit represented in CGP is given in Fig. 1).

2) Search algorithm: Every candidate circuit represents one design point in the design space. In CGP, new designs are created by introducing small random modifications to the chromosome. This operation is called the mutation and it typically modifies h integers of the chromosome. Note that all modifications must lead to valid circuits, i.e. only valid function codes and connections can be created.

The search method is based on $(1+\lambda)$ evolutionary strategy which is usually used for a single-objective circuit approximation by means of CGP [6]. The search algorithm can start

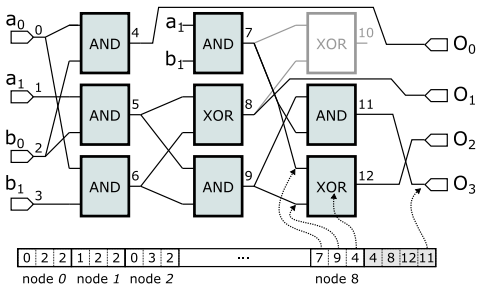

Fig. 1. A two-bit multiplier represented in CGP with parameters: $n_i = n_o = 4, n_c = n_r = 3, n_a = 2, n_b = 1, \Gamma = \{0^{identity}, 1^{not}, 2^{and}, 3^{or}, 4^{xor}, 5^{nand}, 6^{nor}, 7^{xnor}, 8^{cont0}, 9^{const1}\}$.

with either a randomly generated initial population or existing designs. The population size is $1 + \lambda$. After evaluating the initial population (i.e. measuring the circuit functionality and cost the following steps are repeated until the termination condition is not satisfied: (i) the best-scored circuit (called the parent) is selected; (ii) λ offspring circuits are created from the parent by means of mutation; (iii) the population is evaluated.

C. Circuit approximation using CGP

In the context of approximate computing, three evolutionary approximation strategies were developed (Fig. 2).

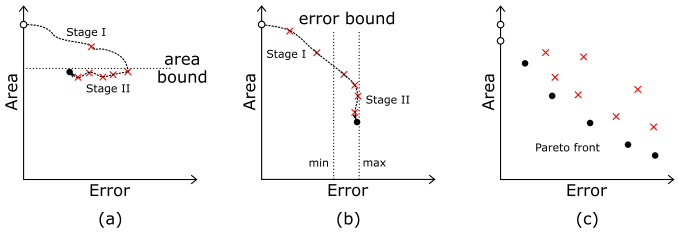

Fig. 2. Evolutionary approximation strategies (a simplified situation for the area-error optimization): (a) resource-oriented, (b) error-oriented and (c) multi-objective. The white points denote the initial (accurate) circuit, the black points are the desired approximate circuits and the crosses denote the valid candidate solutions (design points).

1) Resources-oriented method: CGP is used to minimize the error criterion under the assumption that only m_i components (gates) are available and m_i is lower than the minimal number of components (gates) needed to implement the accurate function [7].

2) Error-oriented method: The target error range (e.g. the worst-case error), determined by e_{min} and e_{max}, is specified by the user. The goal is to optimize the number of components (or area or power consumption) while the error of the circuits is kept between the target valuese_{min} and e_{max} [8]. If various tradeoffs between the error and the number of components are requested, CGP is executed several times with e_{max} as the parameter.

3) Multi-objective CGP: Compared to previous methods that employ a single-objective optimization (one fitness function with constraints), the multi-objective method allows to optimize the error and other key circuit parameters (area, delay and power consumption) together in one CGP run [9].

We are primarily interested in approximate circuits belonging to the *Pareto set* which contains the so-called *nondominated solutions*. For example, consider two circuits C1 and C2. Circuit C1 dominates another circuit C2 if: (1) C1 is no worse than C2 in all objectives, and (2) C1 is strictly better than C2 in at least one objective.

The design flow for approximation of arithmetic circuits is given in Figure 3. The methodology typically starts with an accurate circuit. The candidate circuits are generated and consequently evaluated by means of some evaluation tool. The evaluation must be fast — for small circuits the simulation utilizing all possible input vectors can be used. However, for large circuits this approach is not tractable. A possible solution is to employ advanced verification methods, e.g. SAT-based [10] or BDD-based [11] formal verification. The methodology can easily handle arbitrary constraints such as the time-limit for formal verification (*promptly verifiable circuits* [10]) or special requirements such as accurate multiplication by zero [12].

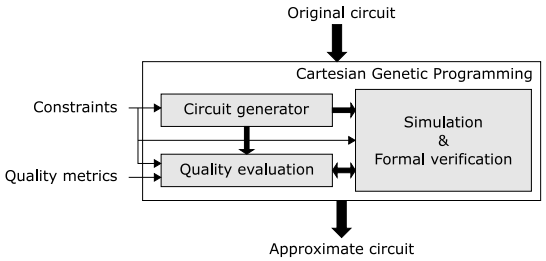

Fig. 3. Overall scheme of automated approximation using Cartesian Genetic Programming.

III. Library of Approximate Arithmetic Circuits

The library contains thousands of various arithmetic circuits as shown in Table I. Since the enormous number of components makes the selection of the most suitable component for a given application difficult, we identified a subset of components and stored them to EvoApproxLib$^{\text{LITE}}$ library. The selection follows the principles Pareto optimality with respect to five objectives in which power consumption is compared against EP, MAE, WCE, MSE and MRE errors. For each of the five subsets of components, ten circuits evenly distributed along the power axis are taken.

An example of one Pareto subset for 8-bit adders and multipliers is shown in Fig. 4. The components in the selected subset are distributed along the whole range of the error. Contrasted to the previous version of the library, no additional constraints on the WCRE error are introduced (the older version limits the WCRE to 100% [4]). It means that the selected subset of components represents better compromises (between exactly two objectives) that those components occupying the Pareto front in [4].

Selected approximate circuits and their various parameters can be downloaded from the EvoApprox8b website https://ehw.fit.vutbr.cz/evoapproxlib. It contains circuit models for Verilog, Matlab, Python and C. This enables the user to integrate the circuits to hardware as well as software projects

TABLE I
Number of components for various circuits in EvoApproxLib

Circuit	Bit-width	# components
adder	8	6,979
	9	332
	12	4,661
	16	1,437
	32	916
	64	176
	128	196
multiplier	8	29,911
	12	3,495
	16	35,406
	32	349

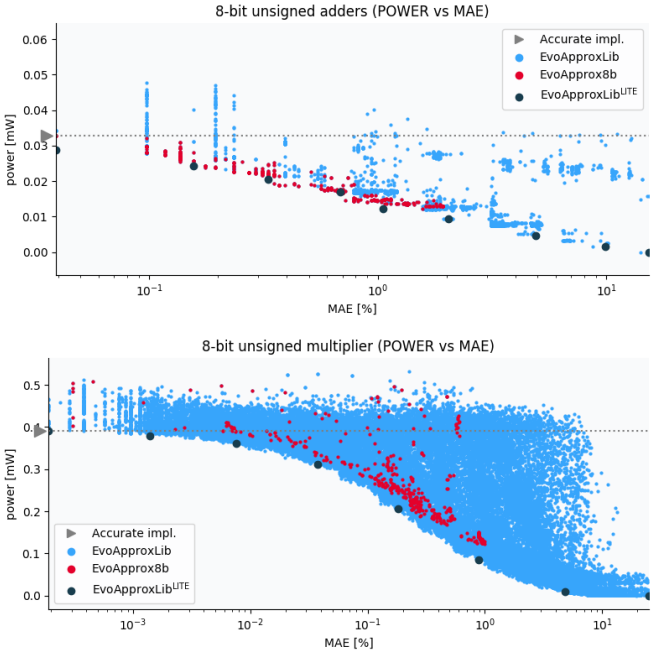

Fig. 4. Selected 8-bit unsigned adders (**top**) and 8-bit multipliers (**bottom**) from Power-MAE Pareto subset compared to the former version of EvoApprox8b library.

and design tools (Fig. 5). All circuits can thus be simulated in order to obtain their other parameters that are not listed on the web site (e.g. the errors under different error metrics or power consumption for another fabrication technology).

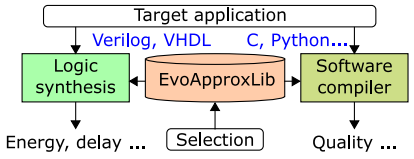

Fig. 5. Integration of the proposed library to the design process.

IV. Conclusions

In this paper we presented a rich library of approximate adders and multipliers which is primarily intended for creating

approximate circuits needed in approximate implementations of complex applications such as energy-efficient image and video processing, deep learning and data mining. For each components, Matlab, C, Python and Verilog models are available for the end users. The users can select the components from the library manually or automated methods can be used [13]. Our future work will be devoted to extending the library with other elementary components and their basic compositions (such as MAC blocks) and developing user-friendly interface.

ACKNOWLEDGMENT

This work was supported by Czech Science Foundation project 19-10137S.

REFERENCES

[1] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *The 50th Annual Design Automation Conference 2013, DAC'13*. ACM, 2013, pp. 1–9.

[2] H. Jiang, C. Liu *et al.*, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, Aug. 2017.

[3] H. R. Mahdiani, 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, no. 4, pp. 850–862, April 2010.

[4] V. Mrazek, R. Hrbacek *et al.*, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Proc. of DATE'17*, 2017, pp. 258–261.

[5] L. Sekanina, Z. Vasicek, and V. Mrazek, *Automated Search-Based Functional Approximation for Digital Circuits*. Springer International Publishing, 2019, pp. 175–203.

[6] J. F. Miller, *Cartesian Genetic Programming*. Springer-Verlag, 2011.

[7] Z. Vasicek and L. Sekanina, "Evolutionary approach to approximate digital circuits design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 432–444, June 2015.

[8] ——, "Evolutionary design of approximate multipliers under different error metrics," in *17th International Symposium on Design and Diagnostics of Electronic Circuits Systems*, April 2014, pp. 135–140.

[9] R. Hrbacek, V. Mrazek, and Z. Vasicek, "Automatic design of approximate circuits by means of multi-objective evolutionary algorithms," in *Proc. of DTIS'16*, 2016, pp. 239–244.

[10] M. Ceska, J. Matyas, V. Mrazek, L. Sekanina, Z. Vasicek, and T. Vojnar, "Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished," in *Proc. of 36th IEEE/ACM Int. Conf. On Computer Aided Design*. IEEE, 2017, pp. 416–423.

[11] Z. Vasicek, V. Mrazek, and L. Sekanina, "Towards low power approximate dct architecture for hevc standard," in *Proc. DATE'17*, 2017, pp. 1576–1581.

[12] V. Mrazek, S. S. Sarwar *et al.*, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *Proc. of ICCAD'16*. ACM, 2016, pp. 81:1–81:7.

[13] V. Mrazek, M. A. Hanif, Z. Vasicek, L. Sekanina, and M. Shafique, "autoax: An automatic design space exploration and circuit building methodology utilizing libraries of approximate components," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. New York, NY, USA: ACM, 2019, pp. 123:1–123:6.