EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods

Vojtech Mrazek, Radek Hrbacek, Zdenek Vasicek and Lukas Sekanina
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: {imrazek, ihrbacek, 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 propose a library of approximate adders and multipliers called EvoApprox8b. This library contains 430 non-dominated 8-bit approximate adders created from 13 conventional adders and 471 non-dominated 8-bit approximate multipliers created from 6 conventional multipliers. These implementations were evolved by a multi-objective Cartesian genetic programming. The EvoApprox8b library provides Verilog, Matlab and C models of all approximate circuits. In addition to standard circuit parameters, the error is given for seven different error metrics. The EvoApprox8b library is available at: www.fit.vutbr.cz/research/groups/ehw/approxlib**

I. Introduction

Approximate circuits are becoming a viable alternative to conventional accurately operating circuits if energy efficiency is crucial and target application is error resilient [1]. This is the case of many, for example, image and video processing circuits that are predominantly composed of adders and multipliers. In order to approximate circuits such as adders and multipliers for a particular application, a designer can either perform a single purpose (ad hoc) approximation or apply some of the circuit approximation methods available in the literature. We will only deal with functional approximation in which logic function implemented by the original circuits is simplified. Other approximation techniques enabling power reduction such as voltage over scaling are not considered in this paper.

Unfortunately, almost all papers dealing with circuit approximation show some of the following features that are undesirable from a practical point of view: (1) The approximation method is described, but a corresponding software implementation is not available. (2) An implementation of the original (accurate) circuit is not available. (3) The quality of approximation and other parameters of approximate circuits are expressed relatively to parameters of the original circuit. If the original circuit is not available, it is hard or even impossible to obtain real parameters of the approximate circuits and reproduce the results. (4) Implementations of the resulting approximate circuits are not available. (5) Only a few approximate versions created from the original circuit are reported, forming thus a sparsely occupied Pareto front. (6) It is unclear if a given number of test vectors used to evaluate approximate circuits is sufficient for obtaining a trustworthy error quantification if the error is determined using simulation. (7) A given approximation method is only rarely compared against competitive approximation methods.

To the best of our knowledge, paper [2] and related software is the only one which does not suffer from the aforementioned problems. However, the paper is solely devoted to 16 bit adders and still only a few approximate adders can be downloaded.

The undesirable properties (1) – (7) are resulting in a situation in which the user does not know which approximation method is the most suitable one for his/her problem because the quality of available approximation methods cannot fairly be compared. If the user does not want to apply any of the approximation methods, its intention could be to use just a resulting approximate circuit showing desired parameter values. However, the chance of direct downloading a circuit with desired parameters is very low, because only a few approximate versions of the original circuit exist in the corresponding repository. On the other hand, it has to be emphasized that a fair comparison of approximation methods is difficult as they should be compared under the same conditions (under the same error metrics, fabrication technology, synthesis tools, available run-time etc.) and start with the same original circuit, which is difficult to ensure in the progressively developing field of approximate computing.

In order to address at least some of the aforementioned challenges, the objective of this paper is to introduce a rich and well-focused library of approximate components (called EvoApprox8b) that can be downloaded and immediately used in various applications or for benchmarking of circuit approximation methods. The library consists of approximate 8-bit adders and 8-bit multipliers that are crucial components of, for example, approximate image and video processing applications. In addition to circuit parameters (error, area, delay, power etc.), the library contains Matlab, C and Verilog implementations for all components which allows the user not only to immediately use the components, but also to calculate the error for a new target error metric or obtain desired parameters by performing a synthesis for a fabrication technology different to the reported one. In addition to that these circuits can be utilized as building blocks of more complex approximate arithmetic circuits (as shown e.g. in [3]).

There are 430 different approximate 8-bit adders and 471 different approximate 8-bit multipliers forming a Pareto front in a three-dimensional space defined using the error, delay, and power metrics. These implementations were obtained by a multi-objective Cartesian genetic programming (CGP) according to [4]. As a starting point for the CGP-based approximation process, 13 different adder and 6 different multiplier accurately-operating architectures were employed. The adders include Ripple-Carry Adder (RCA), Carry-Select Adder (CSA), Carry-Look-ahead Adder (CLA), multiple Tree Adder (TA) and Higher Valency Tree Adder (HVTA) architectures. The multipliers include Ripple-Carry Array, multiple Carry-Save Array and Wallace Tree architectures. These accurate implementations are also included in the library.

CGP has been adopted as our design method because papers [5], [6], [7] revealed that it can provide much better implementations of accurate as well as approximate circuits than common circuit design and optimization tools. The EvoApprox8b library is provided as an open source project, see the EvoApprox8b web site [8].

The rest of the paper is organized as follows. Section II surveys the methods developed for approximation of adders and multipliers. Section III describes the approximation method used to create the EvoApprox8b library. The library is presented in Section IV. Conclusions are given in Section V.

II. Approximate adders and multipliers

Software-oriented benchmark sets such as AxBench [9] were developed for evaluation of approximate software and corresponding approximation methods. Regarding circuit approximation, several approximate circuits can be downloaded from KIT CES web site, including Generic Accuracy Configurable Adders (GeAR) [2]. The remaining papers dealing with circuit approximation methods suffer from problems discussed in Section I and, hence, performing a fair comparison of approximation methods or resulting approximate circuits is difficult. In this section, we will briefly survey approaches developed for adders and multipliers approximation.

Adders and multipliers are approximated by either general-purpose approximation methods or problem-specific methods. In the case of general-purpose methods (e.g. SALSA [10] and SASIMI [11]), adders and multipliers serve as one of many circuit classes that can be approximated. Problem-specific methods exploit the structure of conventional adders and multipliers. Another class of circuits are quality configurable circuits (e.g. GeAR adders) which allow for a dynamic approximation according the expected quality of result [2].

Four types of approximate adders are considered in [12]: (1) Speculative adders in which it is presumed that the probability that the carry propagation chain is longer than k bits ($k << n$) is very low for n-bit adders. Hence, according to k bits the carry is speculated for each sum bit. (2) Segmented adders, where an n-bit adder is divided into k-bit sub-adders and the carry is then generated by using different methods. (3) Carry-select adders in which multiple sub-modules are used to compute the sum for different carry values, and the result is determined according to the carry of a sub-module. (4) Approximate full adders where the full adder (an elementary adder's component) is approximated.

Approximate multipliers are based on the following principles [13]: (1) Approximation in generating partial products utilizing a simpler structure to generate partial products [3]. (2) Approximation in the partial product tree by ignoring some partial products or dividing partial products into several modules and applying approximation in the less significant modules. (3) Using approximate counters or compressors in the partial product tree. (4) Using approximate Booth multipliers. (5) Composing complex approximate multipliers by means of simple approximate multipliers as shown in [3].

Recently, an evolutionary approach was applied in the task of approximate circuits design with respect to multiple objectives [6], [7], [4]. Conventional circuits were used as an initial population. The circuit approximation problem is formulated as a multi-objective search problem and solved using the state-of-the art CGP method [14] combined with the NSGA-II algorithm [15]. In many cases, CGP-based approach is capable of optimizing accurate circuits in such a way that they remain accurate, but they show better parameters (e.g. area) than approximate circuits [4].

There are many error metrics developed to evaluate the quality of approximate arithmetic circuits [16]. While most methods apply test vectors to estimate the error (e.g. 10^8 test vectors were used to evaluate 16 bit adders in [12]), the exact error calculation based on formal models such as binary decision diagrams is rarely used. The accuracy of simulation-based error calculation (which depends on the number and quality of test vectors) can significantly influence the whole approximation process.

III. Circuit Approximation Method

The method used to obtain the library follows the methodology introduced in [4]. It is a general-purpose approximation method for combinational circuits based on a multi-objective CGP. It represents candidate circuits as directed acyclic graphs and tries to simultaneously minimize delay, power consumption and error to discover a set of approximate circuits along a Pareto front. Moreover, various constraints can be formulated to reduce the search space.

A. Circuit representation

In CGP, candidate solutions are represented in a two-dimensional array of programmable nodes [14]. An n_i-input and n_o-output combinational circuit is modelled using an array of $n_c \cdot n_r$ programmable nodes forming a Cartesian grid. A set of available n_a-input/n_b-output node functions is denoted Γ. The primary inputs and programmable node outputs are uniquely numbered. For each node the chromosome contains (n_a+1) values that represent (i) node function and (ii) addresses specifying the input connections. The chromosome also contains n_o values specifying the gates (node outputs) connected to the primary outputs. The chromosome size is $n_c n_r(n_a+1)+n_o$ integers.

In our case, Γ contains functions implemented by standard components (such as gates and adders) of technology library. We used a subset of componets from a TSMC 180 nm technology library. A complete list of technology components including their area and leakage power can be found in [4]. There are multiple implementations of the same component differing in the implementation cost. During the circuit approximation, a proper size was selected for each gate depending on the output load of the gate.

B. *Search method*

New candidate circuits are created by means of a point mutation operator which modifies a given number of integers of the chromosome. The multi-objective search is conducted by the NSGA-II algorithm which is based on the idea of *Pareto dominance relation*. The individuals in each generation are sorted according to the dominance relation into multiple fronts. The solutions within the individual fronts are sorted according to the *crowding distance* metric, which helps to preserve a reasonable diversity along the fronts [15]. The crowding distance is the average distance of two solutions on either side along each of the objectives. A new population is then created by exploiting appropriate Pareto fronts as defined in [17]. The result of a single evolutionary run is not just one solution, but a set of non-dominated solutions occupying the Pareto front. The search method is implemented as a parallel evolutionary algorithm operating with multiple populations distributed on several islands and the best individuals are allowed to migrate among the islands.

C. *Fitness functions*

A selection of the error metric significantly influences obtained results [6]. As the error metric is usually an application-specific function there is no reason to prefer one over another in our library. We decided to optimize according to the mean relative error, but we also quantified the errors according to other commonly used error metrics for all evolved circuits (Section IV). The mean relative error is calculated accurately, i.e. for all possible 2^{n_i} input vectors, as:

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

where $O^{(i)}_{\mathrm{orig}}$ is the decimal representation of the i-th circuit correct output, $O^{(i)}_{\mathrm{approx}}$ is the individual's i-th output, and n_i is the number of primary inputs (i.e. the operand's width is $n_i/2$ bits). In addition to that, we constrained the worst absolute and relative errors to reduce the search space.

The circuit area is a sum of the areas of components used in the circuit. Power consumption is estimated according to the algorithm given in [4]. The delay of a candidate circuit is calculated using the parameters defined in the liberty timing file available for the utilized semiconductor technology. The delay t_d of a cell c_i is modeled as a function of its input

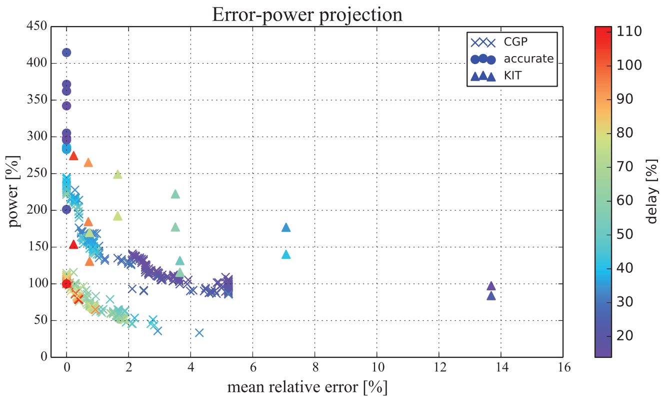

Fig. 1. Pareto fronts with parameters of evolved approximate 8-bit adders, conventional accurate adders and approximate adders according to KIT's paper [2].

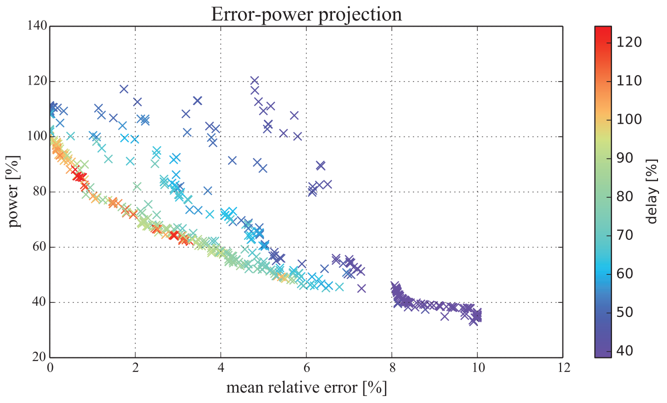

Fig. 2. Pareto fronts with parameters of evolved approximate 8-bit multipliers.

transition time t_s and capacitive load C_l on the output of the cell, i.e. $t_\mathrm{d}(c_i) = f(t_\mathrm{s}^{c_i}, C_\mathrm{l}^{c_i})$. The delay of the circuit C is determined as the delay along the longest path.

D. *Parameters setting*

The CGP/NSGA-II parameters were set as follows: 500 individuals in the population, 100,000 generations, 10 islands, mutation rate 5 %, number of rows $n_\mathrm{r} = 1$. The number of columns was $n_\mathrm{c} = 200$ in the case of the adders and $n_\mathrm{c} = 1000$ in the case of the multipliers.

The circuits were designed with respect to three objectives – the mean relative error (MRE), power consumption and delay. The MRE was constrained to be at most 10 %, the worst case error was constrained to be at most 5 % of the output range and the worst case relative error was limited to 1000 %. All candidate solutions violating these requirements were discarded.

IV. EvoApprox8b Library

The EvoApprox8b library contains 430 non-dominated 8-bit approximate adders evolved from the initial population of 13 conventional adders and 471 non-dominated 8-bit approximate multipliers that were evolved from 6 conventional implementations. In addition to the conventional implementations, the library also contains 43 exact adders and 28 exact multipliers that were obtained by CGP and that are not dominated by any accurate implementation. All Pareto fronts are shown in Figure 1 and 2. All parameters are related to the Ripple-Carry Adder and Ripple-Carry Array Multiplier architectures (considered as 100% in the figures). Figure 1 also provides parameters of 8-bit approximate adders created according to [2]. Evolved adders show quite competitive properties under the metrics used in the figure.

All approximate circuits and their various parameters can be found at the EvoApprox8b web site [8]. It contains circuit models for Verilog, Matlab and C. This enables the user to integrate the circuits to hardware as well as software projects and design tools. All circuits can thus be simulated in order to obtain their other parameters that are not listed on the web site (e.g. errors under different error metrics or power consumption for another fabrication technology). The circuits can be sorted according to a chosen parameter which is useful when the user is looking for a circuit satisfying particular constraints.

The following list gives parameters that are provided for all circuits in the library: Area, delay, power (all estimated according to [4] which was validated against Cadence Encounter RTL Compiler and TSMC 180 nm library), nodes (the number of nodes, where a gate, a half adder and a full adder are counted as one node), HD (Hamming distance), EP (error probability), MAE (mean absolute error), MSE (mean squared error), MRE (mean relative error), WCE (worst case error), and WCRE (worst case relative error). Note that as n_i is the number of primary inputs, the operand's width is $n_i/2$ bits.

$$\mathrm{HD} = \sum_{\forall i} OnesCount(O^{(i)}_{\mathrm{approx}} \oplus O^{(i)}_{\mathrm{orig}}), \tag{2}$$

$$\mathrm{EP} = \frac{\sum_{\forall i: O^{(i)}_{\mathrm{approx}} \neq O^{(i)}_{\mathrm{orig}}} 1}{2^{n_\mathrm{i}}}, \tag{3}$$

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

$$\mathrm{MSE} = \frac{\sum_{\forall i} \left| O^{(i)}_{\mathrm{approx}} - O^{(i)}_{\mathrm{orig}} \right|^2}{2^{n_\mathrm{i}}}, \tag{5}$$

$$\mathrm{MRE} = \frac{\sum_{\forall i} \frac{\left| O^{(i)}_{\mathrm{approx}} - O^{(i)}_{\mathrm{orig}} \right|}{\max(1, O^{(i)}_{\mathrm{orig}})}}{2^{n_\mathrm{i}}}, \tag{6}$$

$$\mathrm{WCE} = \max_{\forall i} \left| O^{(i)}_{\mathrm{approx}} - O^{(i)}_{\mathrm{orig}} \right|, \tag{7}$$

$$\mathrm{WCRE} = \max_{\forall i} \frac{\left| O^{(i)}_{\mathrm{approx}} - O^{(i)}_{\mathrm{orig}} \right|}{\max(1, O^{(i)}_{\mathrm{orig}})}. \tag{8}$$

V. Conclusions

In this paper we presented a rich library of approximate 8 bit adders and multipliers which is primarily intended for creating approximate blocks of image and video processing circuits. The Matlab, C and Verilog models that are available from [8] can allow software as well as hardware developers to integrate the approximate adders and multipliers to their designs. As we are aware of, this is the first open-source library containing hundreds of approximate components that allows for reproducible comparisons across various layers of design abstraction. These components can be utilized as building blocks of more complex approximate adders and multipliers as demonstrated for multipliers in [3].

Acknowledgments

This work was supported by Czech science foundation project GA16-17538S and the Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science – LQ1602.

References

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, p. 62:162:33, 2016.

[2] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, pp. 86:1–86:6.

[3] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," *J. Low Power Electronics*, vol. 7, no. 4, pp. 490–501, 2011.

[4] R. Hrbacek, V. Mrazek, and Z. Vasicek, "Automatic design of approximate circuits by means of multi-objective evolutionary algorithms," in *Proc.of the 11th Int. Conf. on Design and Technology of Integrated Systems in Nanoscale Era*. IEEE, 2016, pp. 239–244.

[5] Z. Vasicek and L. Sekanina, "A global postsynthesis optimization method for combinational circuits," in *Proc. of the Design, Automation and Test in Europe, DATE*. EDA Consortium, 2011, pp. 1525–1528.

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

[7] ——, "Evolutionary approach to approximate digital circuits design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 432–444, 2015.

[8] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina. (2016) Approximate 8-bit Adders and Multipliers. [Online]. Available: www.fit.vutbr.cz/research/groups/ehw/approxlib

[9] A. Yazdanbakhsh, D. Mahajan, P. Lotfi-Kamran, and H. Esmaeilzadeh, "Axbench: A multi-platform benchmark suite for approximate computing," *IEEE Design and Test*, 2016.

[10] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: systematic logic synthesis of approximate circuits," in *The 49th Annual Design Automation Conference 2012, DAC'12*. ACM, 2012, pp. 796–801.

[11] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits," in *Design, Automation and Test in Europe, DATE'13*. EDA Consortium, 2013, pp. 1367–1372.

[12] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*. ACM, 2015, pp. 343–348.

[13] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, "A comparative evaluation of approximate multipliers," in *IEEE/ACM International Symposium on Nanoscale Architectures*. IEEE, 2016, pp. 191–196.

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

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[16] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Trans. Computers*, vol. 62, no. 9, pp. 1760–1771, 2013.

[17] R. Hrbacek, "Parallel multi-objective evolutionary design of approximate circuits," in *GECCO '15 Proceedings of the 2015 conference on Genetic and evolutionary computation*. ACM, 2015, pp. 687–694.