Circuit Approximation Using Single- and Multi-Objective Cartesian GP

Zdenek Vasicek and Lukas Sekanina

Brno University of Technology, Faculty of Information Technology,
IT4Innovations Centre of Excellence
Božetěchova 2, 612 66 Brno, Czech Republic
vasicek@fit.vutbr.cz, sekanina@fit.vutbr.cz

Abstract. In this paper, the approximate circuit design problem is formulated as a multi-objective optimization problem in which the accuracy and power consumption are conflicting design objectives. We compare multi-objective and single-objective Cartesian genetic programming in the task of parallel adder and multiplier approximation. It is analyzed how the setting of the methods, formulating the problem as multi-objective or single-objective, and constraining the execution time can influence the quality of results. One of the conclusions is that the multi-objective approach is useful if the number of allowed evaluations is low. When more time is available, the single-objective approach becomes more efficient.

1 Introduction

Approximate computing is a promising approach for the design of energy efficient computer-based systems (see detailed motivation and survey in [1, 2]). It exploits the fact that many applications are error resilient which means that their users are willing to accept less than perfect solutions, simply because the inaccuracies in the output are not recognizable, or they are well justified under some circumstances. Multimedia applications, search, classification, prediction and recognition tasks are typical domains for approximate computing. Approximations can be introduced at the circuit, component, architecture, software, operating system or system's level. In some cases, the degree of approximation of the accurate solution can be adapted during system's deployment [1, 2]. Because of the nature of evolutionary design and optimization, in which target systems are evolved by introducing small changes into existing structures, it seems that evolutionary computing could be an efficient method to approximate (i.e. purposely modify) existing circuit designs [3, 4].

From the designer's perspective, a reasonable trade-off is sought between the accuracy and power consumption. (Alternatively, the accuracy can be traded for the speed of operation in some applications.) The approximate circuit design problem can be formulated as a multi-objective design problem in which the accuracy and power consumption are conflicting design objectives. A good approximate circuit design tool should provide a set of solutions which exhibit

various trade-offs among key circuit parameters, in particular, the accuracy and power consumption. These solutions should, in an idealized case, perfectly match the so-called Pareto optimal front [5]. Current tools (such as [6–8]) typically solve this problem by multiple executions of approximation engines in order to obtain a set of various solutions. With respect to given constraints and specification, the designer finally selects one of the compromises to be implemented on a chip.

The approximate circuit design is a computationally demanding process which involves generating and comparing many circuit designs. In order to justify this computation time, the resulting circuit should really represent a good compromise between the target objectives. The maximum number of circuits that are allowed to be generated and evaluated thus becomes the main constraint for approximation engines.

The goal of this paper is to compare multi-objective and single-objective versions of Cartesian genetic programming (CGP) [9] in the task of combinational circuit approximation. The reasons for using an advanced evolutionary approach (contrasted to a greedy search used in the state of the art tools [8]) are that the population-based approach suits well in finding multiple solutions and its niche-preservation methods can be exploited to discover diverse solutions [5].

The methodology presented in this paper uses the following principles: (1) the single- and multi-objective search methods are compared under various constraints on the execution time because design time is one of the key factors determining applicability of a design method; (2) the key circuit parameters (area and delay) are estimated during the optimization process while the resulting approximate circuits are implemented using a standard design flow and compared with their accurate counterparts. It has to be noted that performing a fair comparison of various approximation algorithms is not trivial in practice because different teams have the access to different test circuits (some of them are proprietary) and fabrication technologies (correct power estimation depends on a particular fabrication process).

Section 2.1 surveys relevant methods developed to approximate circuit designs. Section 2.2 is devoted to the principles of multi-objective optimization. The proposed single-objective and multi-objective approximate circuit design methods are introduced in Section 3. Experimental results are presented in Section 4. Conclusions are given in Section 5.

2 Related Work

2.1 Approximate Computing

In approximate computing systems, the accuracy (or quality) of the output is traded for improvements in power consumption or performance. This is possible because many applications are intrinsically error resilient and users are willing in many cases to accept less than perfect performance or quality. Approximations are currently applied at all system's levels [2]. We will solely focus on approximate circuits in this paper.

Initial approaches to the functional approximation have been based on a manual identification of subcircuits that should be approximated, for example, in adders and multipliers [10]. However, the manual approach is not efficient and scalable. Later, several systematic automated methodologies have been proposed [6–8, 3].

For example, ABACUS creates an abstract synthesis tree (AST) from the input behavioral description and then applies various operators to the AST using an iterative stochastic greedy approach [8]. Candidate designs are evaluated in terms of accuracy, power consumption and area using a single objective optimization algorithm. The objectives are combined together using a weight function. The Pareto front is obtained from multiple runs of the search algorithm (only about 50 candidate circuits are generated in each run [8]).

The aforementioned methods try to approximate the Pareto optimal front by either combining more design objectives in a single objective search (ABACUS) or executing the approximation algorithm with one fixed criterion (e.g. the error is constant) and optimizing for another one (minimizing power consumption). However, in many cases, the resulting solutions do not cover the whole Pareto front and the design alternatives are centered around a few dominant design alternatives. These methods use the standard design flow to construct and evaluate every candidate circuit, which is very time consuming. On the other hand, the circuit parameters obtained are very close to real ones.

Systematic methods based on the evolutionary design paradigm consider the approximate circuit design problem as a search problem. It was exploited in [3, 4] that power consumption is often highly correlated with occupied resources and the evolutionary design is capable of constructing partly working solutions even if sufficient resources (required for finding a fully functional solution) are not available. The user then obtains, in multiple runs of CGP, a set of approximate combinational circuits, each of which typically exhibits different trade-off between the accuracy and the number of gates. Delay was not addressed in [3].

2.2 Multi-objective Optimization

In general, the multi-objective optimization problem can be defined in the following form:

$$\begin{aligned} \text{optimize: } & f_m(\boldsymbol{x}), && m = 1, 2, ..., M \\ \text{subject to: } & g_j(\boldsymbol{x}) \geq 0 && j = 1, 2, ..., J \\ & h_k(\boldsymbol{x}) = 0 && k = 1, 2, ..., K \end{aligned} \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is a vector representing the solution consisting of n decision variables. The objective functions are denoted $f_1, \ldots, f_M$. Some of these functions have to be minimized, others have to be maximized. Functions $g_j(\boldsymbol{x})$ and $h_k(\boldsymbol{x})$ define the optimization constrains and thus determine the space of feasible solutions.

In order to compare two solutions, Pareto-dominance relations are employed [5]: Solution $\boldsymbol{x}^{(1)}$ *dominates* another solution $\boldsymbol{x}^{(2)}$ if the following conditions are sat-

isfied: (i.) The solution $\boldsymbol{x}^{(1)}$ is no worse than $\boldsymbol{x}^{(2)}$ in all objectives. (ii.) The solution $\boldsymbol{x}^{(1)}$ is strictly better than $\boldsymbol{x}^{(2)}$ in at least one objective.

The result of the multi-objective optimization is no longer a single solution, but a set of solutions. In a set of solutions P, a non-dominated subset of solutions P' contains those solutions that are not dominated by any member of P. The non-dominated subset of all possible solutions is called Pareto-optimal set (front). The members of this subset are optimal solutions to the multi-objective optimization problem. The ultimate goal of any multi-objective optimization algorithm is to find all solutions which belong to the Pareto-optimal front. In practice, the goal is to find a set of solutions as close as possible and as diverse as possible with respect to the Pareto-optimal front.

A straightforward approach to the multi-objective optimization is converting the multi-objective problem to a single objective one using a weight function $\sum w_i f_i$, where w_i is the weight of the i-th objective. Because a single run of the optimizer which uses the sum yields only one solution, multiple runs are needed for obtaining various trade-offs. The proper setting of weights w_i is not an easy task and is usually based on the user intuition. Another limitation of the weight function lies in the fact that certain Pareto-optimal solutions are not reachable in the case of nonconvex objective space [5]. Since it is difficult to detect whether the resulting objective space is nonconvex, the weight function has to be applied with caution.

In order to precisely approximate the whole Pareto-optimal front and obtain various diverse non-dominate solutions in a single run of an optimizer, truly multi-objective evolutionary algorithms have been introduced, for example, non-dominated sorting genetic algorithm (NSGA-II). Contrasted to the single-objective optimization algorithms, they internally sort individuals according to the dominance relation, build archives of non-dominating solutions, and ensure population diversity to avoid converging to a single solution. In the context of evolutionary design of (exact) circuits, multi-objective CGP has been applied in [11, 9].

3 The Proposed Search Methods

The proposed approach is based on Cartesian genetic programming [9] and its multi-objective extension utilizing the NSGA-II [12].

3.1 Circuit Representation

A candidate circuit is modeled by means of a directed acyclic graph whose nodes (gates) are organized in n_c columns and n_r rows. The circuit has n_i primary inputs and n_o primary outputs. Each node input can be connected either to the output of a node placed in previous l columns or to one of the primary circuit inputs, where l is one of CGP parameters.

A candidate solution consisting of two-input nodes is represented in the chromosome by $n_r \cdot n_c$ triplets (x_1, x_2, ψ) determining for each processing node its

function ψ ($\psi \in \Gamma$), and addresses of nodes x_1 and x_2 which its inputs are connected to. The last part of the chromosome contains n_o integers specifying either the nodes where the primary outputs are connected to or logic constants ('0' and '1') which can directly be connected to the primary output. While the chromosome size is constant for a given product $n_r \cdot n_c$, the phenotype size is variable and measured as the number of used nodes (gates).

3.2 Single-objective Search

The initial population of CGP is created either randomly or by means of existing circuits. Candidate circuits are evaluated using the fitness function. If a multi-objective optimization is conducted, there are several fitness functions formulated, each of them reflecting to what extent a given circuit parameter (accuracy, area, delay etc.) satisfies the specification.

When multiple-objectives are aggregated to a single fitness value (e.g. using the weight function), we speak about a single-objective optimization. Each member of the population then receives one fitness value and the highest-scored idividual becomes a new parent of the next population.

New circuits are created from the parent using mutation, which is the only operator used in CGP. The mutation modifies h randomly selected genes (integers) of the parent circuit. CGP usually employs a $1 + \lambda$ search strategy. The evolution is terminated when a predefined number of generations is exhausted or a suitable solution is discovered.

3.3 Multi-objective Search

In the multi-objective algorithm, the $1 + \lambda$ search strategy is replaced by procedures of NSGA-II which implement non-dominated sorting of the population (non-dominated solutions are emphasized) and diversity preservation mechanisms (less crowded points of the search space are promoted) – details can be found in [12]. Here, the population consists of λ_{MO} individuals. The non-dominated sorting algorithm of NSGA-II was modified in such a way that when all components of the fitness score of a parent and its offspring remain unchanged, the offspring is classed as dominating the parent, and is therefore ranked higher than the parent. Moreover, the maximum allowed error E_{max} (which the designer is going to observe and accept in the resulting Pareto fronts) is defined as a constraint in our algorithm. In order to optimize the error (inaccuracy), area and delay, three fitness functions (all to be minimized) will be constructed. If fitness $f_{error} > E_{max}$, the solution is considered as unacceptable.

3.4 Methodology

As many candidate circuits will be generated and evaluated in the course of evolution, it is intractable to precisely calculate power consumption and other circuit parameters for each of them. Hence we will only calculate the error and

estimate the area and delay. As power consumption is highly correlated with the area (which can be assumed for certain technology nodes), it is particularly important to find good compromises between the area and error. At the end of the evolutionary optimization, selected approximate circuits will be implemented using a standard design flow and compared with their fully functional versions.

In order to estimate parameters of a given circuit, the area and delay are calculated using the parameters defined in the liberty timing file available for a given semiconductor technology. This file gives the area, timing and power-relevant parameters of each cell (gate).

Delay t_d of a cell c_i is modeled as a function of its input transition time t_s and capacitive load c_l on the output of the cell, i.e. $t_d(c_i) = f(t_s^{c_i}, c_l^{c_i})$. Delay of circuit C is determined as delay of the longest path:

$$Delay(C) = \max_{\forall p \in \text{path}} \sum_{c_i \in p} t_d(c_i).$$

The capacitive load on the circuit outputs is chosen to be equal to the input capacitance of an inverter cell. The transition time on circuit inputs corresponds to the transition time on the output of an inverter cell.

The area of circuit C is calculated as the sum of areas of all cells c_i involved in the circuit:

$$Area(C) = \sum_{c_i \in C} area(c_i).$$

Various error criteria can be utilized to evaluate the quality of an approximate arithmetic circuit. The average error magnitude $E_{avg}(C)$ is employed in our case. This metric is defined as the sum of absolute differences in magnitude between the original and approximate circuits averaged over all inputs:

$$E_{avg}(C) = \frac{\sum_{\forall i} |Y(C_{orig}, i) - Y(C, i)|}{2^{2w}} \tag{2}$$

where $Y(C_{orig}, i)$ denotes the output value of the fully functional circuit for the input vector i, $Y(C, i)$ denotes the output value of approximate circuit C and w specifies the bit-width.

Multi- as well as single-objective algorithm is seeded with a fully functional version of an arithmetic circuit. In both cases the user is supposed to define E_{max}. However, the interpretation of E_{max} is different. In the case of multi-objective optimization, solutions with the error greater than E_{max} are unacceptable; the remaining solutions are considered during the Pareto front construction. In the case of single-objective optimization, the evolutionary algorithm is used to find a solution showing the error as close as possible to E_i. In order to construct Pareto front, the single-objective algorithm has to be executed multiple times with E_i increasing from a small error to E_{max} in several steps.

In order to obtain a solution with the required error level, the single-objective algoritm works in two stages. The goal of the first stage is to produce a circuit with error as close as possible to the required level E_i regardless the other optimization criteria. To achieve this objective, the fitness value $fitness_{L1}$ is calculated as the relative absolute difference from the required error level and the

goal is to minimize this difference, i.e.

$$fitness_{L1}(C) = \frac{|E_{avg}(C) - E_i|}{E_i}$$

If the required error is obtained ($fitness_{L1} < 0.01$), the algorithm continues by the second stage. In this stage, additional optimization objectives are considered and the error serves as a constraint which guarantees that the error value is kept as close as possible to the required error level. Each objective is normalized to the interval $< 0, 1 >$ and weighted with a weight w_e, w_a or w_d ($w_e + w_a + w_d = 1$). Then,

$$fitness_{L2}(C) = \begin{cases} w_e E'_{avg}(C) + \\ w_a Area'(C) + \\ w_d Delay'(C), & \text{if } fitness_{L1} < 0.01 \\ \infty, & \text{otherwise,} \end{cases}$$

where the apostrophe denotes a normalized value with respect to the original, fully functional circuit.

4 Results

4.1 Experimental Setup

The single-objective (SO) and multiple-objective (MO) algorithms based on CGP are evaluated in the task of 4-bit and 8-bit adder and multiplier approximation. E_{max} is chosen to be 2.5% of the maximum average error, where the maximum average error is $(2^w - 1)^2$ for the multiplier and $2(2^w - 1)$ for the adder. While the multi-objective algorithm is executed with E_{max}, the single-objective algorithm is executed SO_{run} times; one run for one error level from 0% to 2.5%. In both cases, CGP was initialized by fully functional circuits. We compared three sets of weights for the single-optimization algorithm $w_e/w_a/w_d = \{(0.8, 0.12, 0.08), (0.5, 0.38, 0.12), (0.12, 0.5, 0.38)\}$ (inspired in [8]).

In both approaches we used the following CGP parameters: $h = 5$, $l = n_c = N_g$, $n_r = 1$, $n_i = 2w$, $n_o = 2w$ for w-bit multiplier and $n_o = w + 1$ for w-bit adder, where N_g is the number of gates of the original fully functional circuit. In both cases, $5 \cdot 10^3$ evaluations (fitness calls) were allowed, corresponding to 100 generations of the multi-objective algorithm ($\lambda_{MO} = 50$). In the case of the single-objective algorithm, 50 generations are produced for each of 20 error levels ($\lambda_{SO} = 5$, $SO_{run} = 20$). This number of evaluations is very low from the perspective of evolutionary circuit design; however, this number is still much higher than in conventional methods [8].

The experiments were conducted for I3T25 technology (0.35 μm digital process). The following cells (and thus functions in Γ) are considered: *and, or, xor, nand, nor, xnor, buf, inv*, with corresponding relative areas 1.333, 1.333, 2, 1, 1, 2, 1.333, and 0.667.

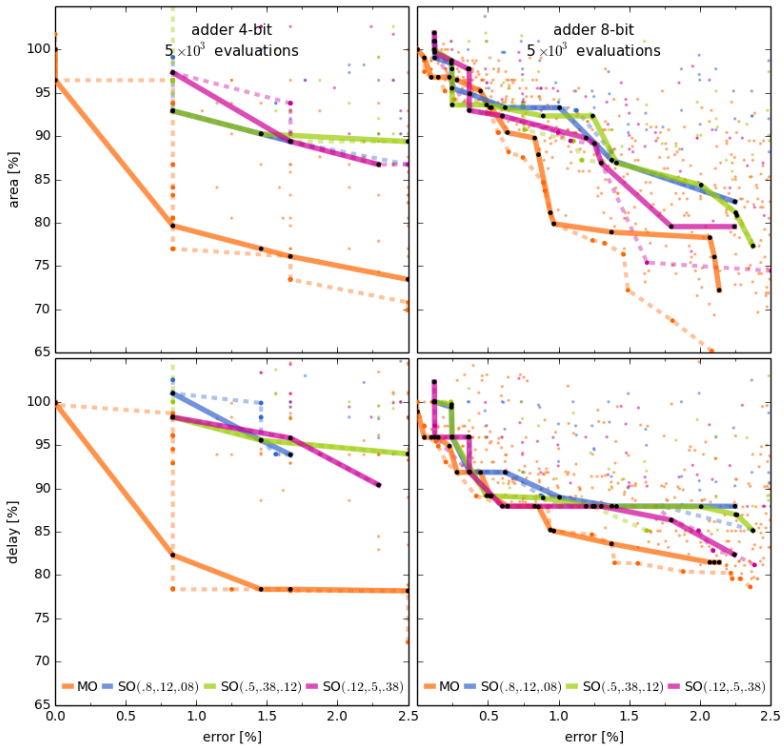

Fig. 1. Adders: Performance of single- and multi-objective methods with respect to original fully functional circuits (0% error).

4.2 Comparison of Single- and Multi-objective Search

Fig. 1 and 2 show the resulting parameters of all circuits as dots in two 2D plots (area vs. error and delay vs. error). These figures contain results from 25 independent runs of the algorithms for each scenario. The 3D Pareto front (projected to two 2D graphs) is interpolated using solid lines for each investigated scenario. Other Pareto fronts (dashed lines) are constructed in such a way that one objective (either area or delay) is ignored. These (dashed) Pareto fronts represent better compromises because the problem is simplified. One can observe that MO performs better than SO and the weights' setting in SO is negligible.

In another experiment, we investigated whether increasing the number evaluations to $500 \cdot 10^3$ can improve the quality of results. Fig. 3 shows that with decreasing the acceptable error, the results produced by SO are better than those from MO. The weight of the area in the fitness function (SO) becomes more important and its unsuitable setting can influence the result by 20%. The SO approach exploits the fact that the error is fixed and the overall effort can be put into minimizing the area and delay. On the other hand, MO has to cover the

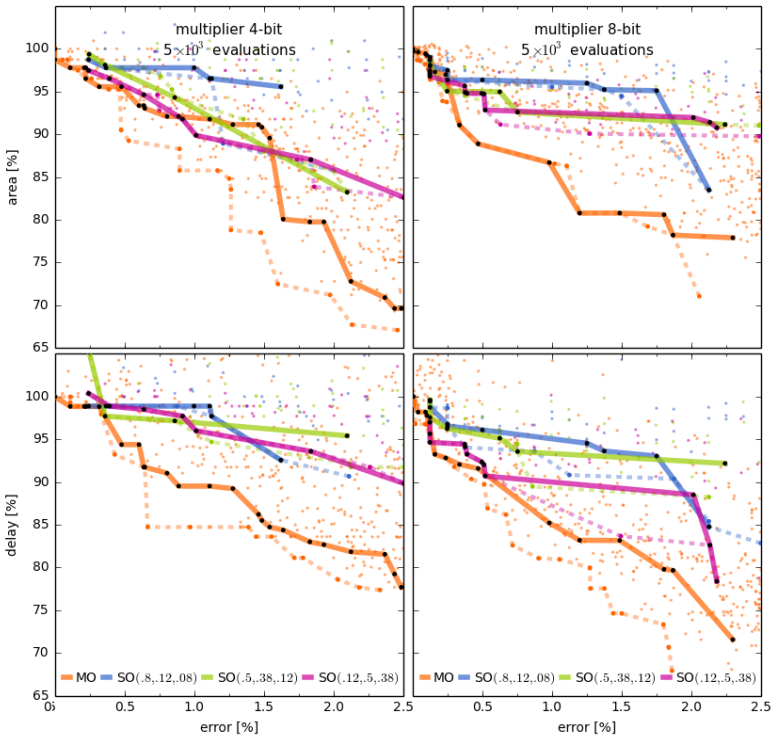

Fig. 2. Multipliers: Performance of single- and multi-objective methods with respect to original fully functional circuits (0% error).

whole Pareto front and the available time seems to be insufficient to compete with SO.

In order to further investigate the computation requirements, we analyzed the quality of resulting solutions with respect to the number of allowed evaluations (generations) in Fig. 4. In the case of MO, the progress of evolution is negligible after $50 \cdot 10^3$ evaluations. SO is capable of improving the quality until $250 \cdot 10^3$ evaluations are spent on average, when solutions probably very close to the Pareto optimal front are obtained.

The computational requirements of the multi-objective algorithm are slightly higher than for the single-objective method (457 vs. 491 evaluations per second for the 8-bit adder and 240 vs. 284 evaluations per second for the 8-bit multiplier).

4.3 Results of Synthesis

In order to validate the presented results, we implemented selected circuits using a standard design flow. The original circuits and selected circuits obtained by

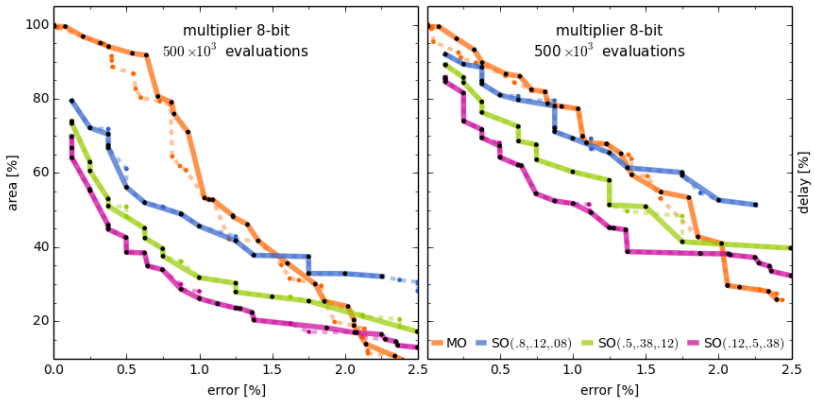

Fig. 3. Resulting Pareto fronts when $500 \cdot 10^3$ evaluations are allowed

CGP were converted into a netlist, and after synthesis, placement and routing (Cadence Encounter RTL Compiler), we compared parameters of resulting circuits with the estimated values used during the evolution. Table 1 shows that our estimated values are almost perfectly matched with the results of the professional design tool (see Impr. columns). The area is correlated with power consumption under the investigated scenario.

A direct and fair comparison with some results from the literature is not possible for several reasons: implementations of methods such as SASIMI and SALSA are not available; only some parameters of benchmark circuits reported in the literature are known (i.e. their implementation is not available); and results are given for different fabrication technology. The proposed method led to 71% power reduction with 0.6% average error, which seems to be a good result for 8-bit multiplier in comparison with SASIMI [7] (45% power reduction with 0.5% average error) and Gupta et al. [13] (35% power reduction with 2.5% average error), despite the fact that different technology was used.

5 Conclusions

In this paper, we proposed and compared two evolutionary approximation circuit design methods based on single- and multiple-objective CGP. Contrasted to current approaches, in which every candidate circuit is implemented and evaluated by means of a profession design tool, candidate circuits' parameters are only quickly estimated in the optimization process. It allowed us to generate many more candidate designs than state-of-the art methods. It was shown that the multi-objective method is useful if the number of allowed evaluations is low. On the other hand, when more time is available, the single-objective method outperforms the multi-objective one. We validated key circuit parameters of selected approximate circuits by means of a commercial design flow. By employing

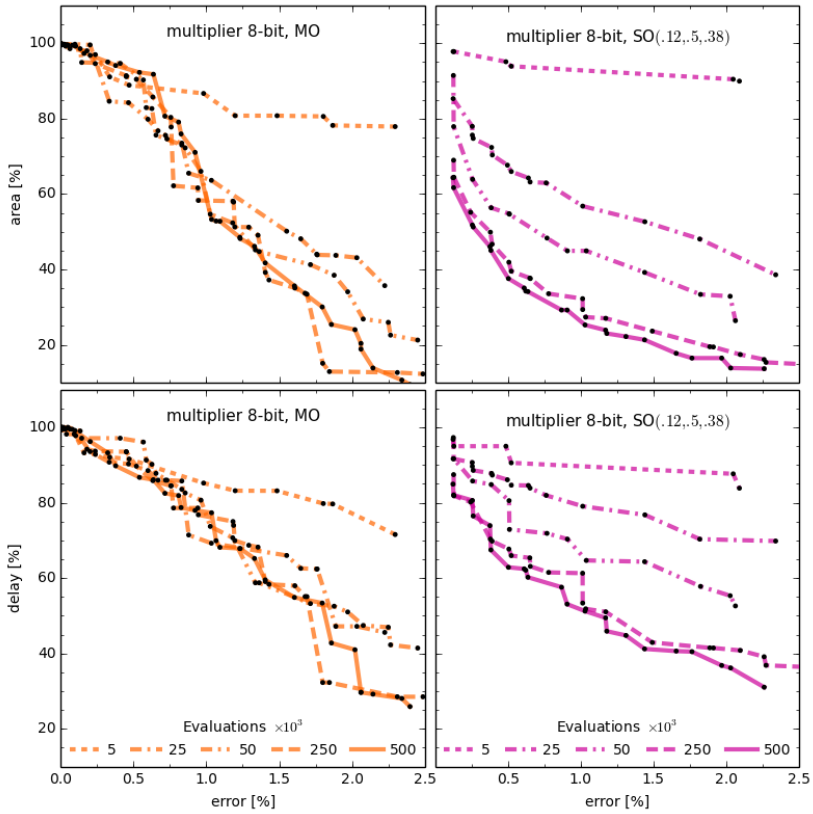

Fig. 4. Resulting Pareto fronts with respect to the number of evaluations

the advanced optimization algorithms and allowing more computation time, we obtained very good approximations of the Pareto optimal fronts for adders and multipliers.

Acknowledgments. This work was supported by the Czech science foundation project Advanced Methods for Evolutionary Design of Complex Digital Circuits 14-04197S. The authors would like to thank Jiri Petrlik for useful discussions on multiobjective evolutionary optimization.

References

1. Chakradhar, S.T., Raghunathan, A.: Best-effort computing: Re-thinking parallel software and hardware. In: Proceedings of the 47th Design Automation Conference – DAC, ACM (2010) 865–870

Table 1. Parameters of the approximate 8-bit adders and multipliers. The Impr. columns give the improvement w.r.t. the original circuits.

		estimated				professional tool					
	Error	Delay	Impr.	Rel.Area	Impr.	Delay	Impr.	Area	Impr.	Power	Impr.
	[%]	[ns]	[%]	[–]	[%]	[ns]	[%]	[μm^2]	[%]	[μW]	[%]
adder	0.0	2.8	–	99	–	2.7	–	4759	–	208	–
	0.6	2.5	12	51	49	2.2	19	2460	49	104	50
	1.2	2.1	27	32	68	1.9	29	1622	66	71	66
	1.9	1.7	38	28	72	1.6	40	1374	72	55	74
	2.5	1.8	36	20	80	1.6	39	978	80	42	80
multiplier	0.0	13.1	–	495	–	12.1	–	24245	–	1367	–
	0.6	8.7	34	175	65	8.0	35	8480	66	409	71
	1.3	6.3	52	118	77	5.6	54	5424	78	233	83
	1.9	5.4	59	92	82	5.1	58	4513	82	164	88
	2.5	4.5	66	64	87	4.3	65	3118	88	106	93

2. Han, J., Orshansky, M.: Approximate computing: An emerging paradigm for energy-efficient design. In: Proc. of the 18th IEEE European Test Symposium, IEEE (2013) 1–6
3. Sekanina, L., Vasicek, Z.: Approximate circuits by means of evolvable hardware. In: IEEE Int. Conf. on Evolvable Systems, SSCI-ICES, IEEE CIS (2013) 21–28
4. Vasicek, Z., Sekanina, L.: Evolutionary approach to approximate digital circuits design. IEEE Tran. on Evolutionary Computation – to appear (2015) 1–13
5. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley (2001)
6. Venkataramani, S., Sabne, A., Kozhikkottu, V.J., Roy, K., Raghunathan, A.: Salsa: systematic logic synthesis of approximate circuits. In: The 49th Annual Design Automation Conference 2012, DAC '12, ACM (2012) 796–801
7. Venkataramani, S., Roy, K., Raghunathan, A.: Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits. In: Design, Automation and Test in Europe, DATE'13, EDA Consortium San Jose, CA, USA (2013) 1367–1372
8. Nepal, K., Li, Y., Bahar, R.I., Reda, S.: Abacus: A technique for automated behavioral synthesis of approximate computing circuits. In: Proc. of the Design, Automation and Test in Europe. DATE '14, EDA Consortium (2014) 1–6
9. Miller, J.F.: Cartesian Genetic Programming. Springer-Verlag (2011)
10. Kulkarni, P., Gupta, P., Ercegovac, M.D.: Trading accuracy for power in a multiplier architecture. J. Low Power Electronics **7**(4) (2011) 490–501
11. Hilder, J., Walker, J., Tyrrell, A.: Use of a multi-objective fitness function to improve cartesian genetic programming circuits. In: NASA/ESA Conference on Adaptive Hardware and Systems, IEEE (2010) 179–185
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation **6**(2) (2002) 182–197
13. Gupta, V., Mohapatra, D., Raghunathan, A., Roy, K.: Low-power digital signal processing using approximate adders. IEEE Trans. on CAD of Integrated Circuits and Systems **32**(1) (2013) 124–137