On Area Minimization of Complex Combinational Circuits Using Cartesian Genetic Programming

Zdenek Vasicek and Lukas Sekanina
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: vasicek@fit.vutbr.cz, sekanina@fit.vutbr.cz

***Abstract*—The paper deals with the evolutionary post synthesis optimization of complex combinational circuits with the aim of reducing the area on a chip as much as possible. In order to optimize complex circuits, Cartesian Genetic Programming (CGP) is employed where the fitness function is based on a formal equivalence checking algorithm rather than evaluating all possible input assignments. The standard selection strategy of CGP is modified to be more explorative and so agile in very rugged fitness landscapes. It was shown on the LGSynth93 benchmark circuits that the modified selection strategy leads to more compact circuits in roughly 50% cases. The average area improvement is 24% with respect to the results of conventional synthesis. Delay of optimized circuits was also analyzed.**

I. Introduction

Cartesian Genetic Programming (CGP) can be considered as one of the most efficient methods for evolutionary design and optimization of digital combinational circuits [1], [2]. Miller and his collaborators showed more than 10 years ago that small combinational circuits such as 4-bit multipliers can be designed in a new way, often saving around 20% gates in comparison with the state-of-the art conventional synthesis tools. This seminal work has been extended in several ways. The most recent research in the area of digital circuits includes the evolution of novel standard cell libraries for future technology nodes [3] and a post synthesis optimization of complex combinational circuits using formal verification principles [4]. While the achievable quality of resulting design/optimization is usually very high, the computational time required to achieve that result is the main drawback.

The fitness computation is the most time consuming procedure. The basic method testing 2^n input vectors for n-input circuit is not scalable. Hence, good results were produced for combinational circuits with only 10–20 inputs (depending on a particular instance) [1], [2], [5], [6], [7], [8]. Most work in this area considers the number of gates as the only criterion for optimization. Delay, area on a chip, power consumption, testability and other important properties are not usually addressed. Multiobjective optimization has been applied for very small problem instances only [9].

In this work, we will follow the path of the single-criterion evolutionary synthesis with the aim of area minimization because it is still the most important criterion for many applications. Moreover, this is a classic subtask in the logic optimization research where delay and other criteria are not always considered [10]. The goal of this work is to obtain as small phenotypes (resulting circuits) as possible for complex combinational circuits. In order to reduce the fitness computation time, we will replace the standard approach, which applies all possible input combinations to determine the fitness value of a candidate circuit by a formal equivalence checking algorithm, which is able to relatively quickly check whether a candidate solution is functionally correct even for most large-scale circuit instances (hundreds of inputs) [11].

In this context, two selection strategies will be compared. We will show that the selection of a parent on basis of its functionality solely instead of compactness (the area on the chip) can lead to small phenotypes at the end of evolution for many of our benchmark problems. This behavior has already been observed during evolution of small combinational circuits [12]. In order to compare and evaluate the role of selection mechanisms, we propose to measure the number of nondestructive mutations during evolution. It is supposed that the modified selection will generate more functionally correct individuals than the standard selection of CGP.

In contrast to most works on this topic, the optimization criterion will be the estimated area on a chip instead of the number of gates. This approach will allow us to fairly compare our results with the state-of-the-art synthesis tools such as ABC [13] or SIS [14]. Another issue, which is not usually addressed by the evolvable hardware community, is to what extent the delay of a combinational circuit is modified when the circuit is optimized for the area only. The evaluation of proposed methods will be performed on a cluster of workstations and using the LGSynth93 benchmark circuits.

The rest of the paper is organized as follows. Section II presents CGP as a method for digital circuit evolution and optimization. The modified selection mechanism for CGP is introduced in Section III. We applied a fitness calculation based on formal equivalence checking. The method is described in Section IV. Our proposal for combining the modified selection strategy and equivalence checking-based fitness function is presented in Section V. The results of experiments are summarized in Section VI. Section VII deals with discussion of obtained results and analysis of results on the basis of the non-destructive mutations measurement. Finally, conclusions are given in Section VIII.

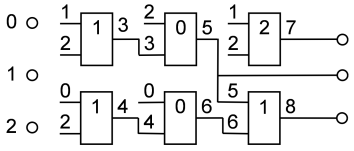


Fig. 1. Example of a candidate circuit in CGP with parameters: $n_c = 3$, $n_r = 2$, $n_i = 3$, $n_o = 3$, $l = 3$, $n_a = 2$, Γ = {NOR (0), NAND (1), XOR(2)}. Chromosome: 1, 2, **1**, 0, 2, **1**, 2, 3, **0**, 0, 4, **0**, 1, 2, **2**, 5, 6, **1**; 7, 5, 8. (Gate function is typed in bold.).

II. Circuit Evolution Using CGP

Cartesian Genetic Programming is a subarea of genetic programming where candidate individuals are represented as directed acyclic graphs, a mutation is considered as the main genetic operator and evolution is carried out using the $1 + \lambda$ evolution strategy [15], [1].

A. CGP

From the perspective of circuit design, a candidate circuit is represented as an array of n_c (columns) $\times$ n_r (rows) of programmable gates. The number of inputs, n_i, and outputs, n_o, is fixed. Each gate can be connected either to the output of a gate placed in previous l columns or to one of the circuit inputs. Setting of the l parameter allows to control the maximum circuit delay. Feedback is not allowed. Each gate is programmed to perform one of n_a-input functions defined in the set Γ (n_f denotes $|\Gamma|$). Each gate is encoded using $n_a + 1$ integers where values $1 \dots n_a$ are the indexes of the input connections and the last value is the function code. Every circuit is encoded using $n_c.n_r.(n_a+1)+n_o$ integers. Figure 1 shows an example of a candidate circuit and its chromosome.

CGP operates with population of $1+\lambda$ individuals (typically, λ is between 1 and 20). The initial population is constructed either randomly or by a heuristic procedure. Every new population consists of the best individual of the previous population and its λ offspring individuals. The offspring individuals are created using a point mutation operator which modifies h randomly selected genes of the chromosome, where h is a user-defined value.

An important rule for selection of the parent is utilized. In case when two or more individuals can serve as the parent, an individual which has not served as the parent in the previous generation will be selected as the new parent. This strategy is important because it ensures a diversity of population [16]. The algorithm is terminated when the maximum number of generations is exhausted or a sufficiently working solution is obtained.

B. Fitness Function

In case of digital circuit evolution, the fitness value of a candidate circuit is defined as [17]:

$$fit1 = \begin{cases} b & when \quad b < n_o 2^{n_i}, \\ b + (n_c n_r - z) & otherwise, \end{cases} \tag{1}$$

where b is the number of correct output bits obtained as response for all possible assignments to the inputs, z denotes the number of gates utilized in a particular candidate circuit and $n_c.n_r$ is the total number of available gates. It can be seen that the last term $n_c n_r - z$ is considered only if the circuit behavior is perfect, i.e. $b = b_{max} = n_o 2^{n_i}$. Alternatively, we can replace the number of utilized gates by the number of utilized transistors which is a more precise measure as implementation costs of gates are different [18]. We can observe that the evolution has to discover a perfectly working solution firstly while the size of circuit is not important. Then, the number of gates is optimized. Similarly, delay or power consumption may be optimized.

Although many new designs have been discovered using the standard CGP, the method is not applicable for design or optimization of large circuits because of the time consuming fitness evaluation. A multi-objective formulation of circuit evolution problem was also proposed, but evaluated using small benchmark problems only [9].

III. Selection Mechanisms in CGP

In the proposed modification of the selection mechanism, there is only one requirement for selection of the parent individual in case that a functionally correct circuit has already been discovered: the parent must be fully functional. Note that in the standard CGP, the parent is always the best circuit discovered so far, i.e. functionality as well as size are considered. The selection procedure can be formalized as follows.

Let p denote the highest-scored individual with the fitness value f_p. The new selection strategy is proposed *only* for situation when the number of gates is optimized, i.e. the fitness value of the best individual is higher than or equal to b_{max}. Otherwise, the algorithm works as the standard CGP. As the best individual found so far will not be copied to the new population automatically, it is necessary to store it in an auxiliary variable. Let β denote the best discovered solution and let f_β be its fitness value. In the first population, β is initialized using p.

Assume that $x_1 \dots x_\lambda$ are individuals (with fitness values $f_{x_1} \dots f_{x_\lambda}$) created from the parental solution p using the mutation operator and $f_\beta \geq b_{max}$. Because the best individual β and parental individual p are not always identical we have to determine their new instances β' and p' separately. The best-discovered solution is defined as:

$$\beta' = \begin{cases} \beta & when \quad f_\beta \geq f_{x_i}, i = 1 \dots \lambda, \\ x_j & otherwise, \end{cases} \tag{2}$$

where x_j is the highest-scored individual for which $f_{x_j} > f_\beta$ holds. If multiple individuals exist in $\{x_1 \dots x_\lambda\}$ that satisfy the previous condition then one of them is randomly chosen.

The new parental individual is defined as:

$$p' = \begin{cases} p & when \quad \forall i, i = 1 \dots \lambda : f_{x_i} < b_{max} \\ x_j & otherwise, \end{cases} \tag{3}$$

where x_j is a randomly selected individual from those in $\{x_1 \dots x_\lambda\}$ which obtained the fitness score higher than or equal to b_{max}.

In other words, the new parent must be a fully functional circuit; however, the number of gates is not important for its selection. Note that the result of evolution is no longer p but β. Paper [12] showed that the modified selection strategy leads to smaller circuits than the strategy used in the standard CGP for majority of tested circuits; however, only small problem instances have been studied so far.

IV. Fast Fitness Evaluation

Since there are many conventional circuit design methods, we can easily obtain a (typically non-optimal) implementation of target circuit and use it to seed the initial population for CGP. Despite the fact that the conventional solution might bias the search algorithm to some subareas of the search space there are some benefits, namely the design time reduction in comparison to a search from scratch. It is important for us that the conventional solution can also be utilized as a reference solution for an equivalence checking algorithm. Instead of applying all possible assignments to the inputs (see eq. 1) every new candidate circuit is compared with the reference circuit in order to determine its functionality. Functionally incorrect candidate circuits are discarded. In case that a candidate circuit is fully functional then its fitness value is given by the number of gates or transistors.

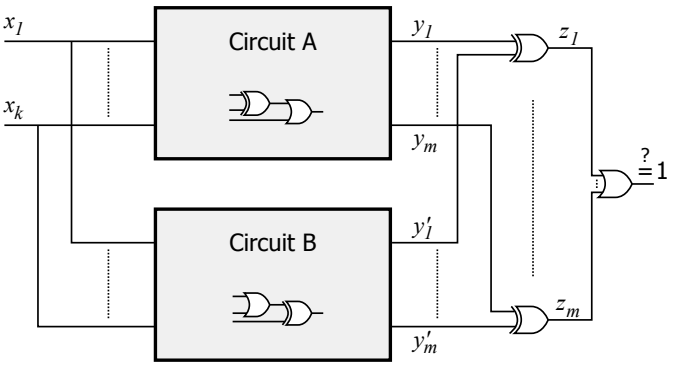


Fig. 2. Equivalence checking of two combinational circuits

A. Equivalence Checking

Determining whether two Boolean functions are functionally equivalent represents a fundamental problem in formal verification. Although the functional equivalence checking is an NP-complete problem, several approaches have been proposed so far to reduce the computational requirement for practical circuit instances.

Most of proposed techniques are based on representing a circuit by means of its canonical representation. Generally, two Boolean functions are equivalent if and only if canonical representations of their output functions are equivalent. The Reduced Ordered Binary Decision Diagrams (ROBDD) represent a widely used canonical representation in formal verification [19]. Some of methods developed to determine whether two ROBDDS are isomorphic are based on graph-based algorithms. Other methods are based on the combination of ROBDDs with the XOR operation and checking whether the resulting ROBDD is a constant node (zero). And-or-invert graphs represent another canonic representation with similar properties. All these graph-based approaches rely on the fact, that the number of nodes in the resulting graph will be relative small, otherwise, the time of the ROBDD construction as well as the time of comparison will be enormous. In practice, these methods are rarely implemented directly without any further circuit preprocessing. High consumption of memory resources has motivated researchers to look for alternative methods. Since the satisfiability (SAT) solvers were significantly improved during the last few years, the SAT-based equivalence checking becomes to be a promising alternative to the BDD-based checking.

B. SAT-based Equivalence Checking

A SAT-based equivalence checking was applied to CGP in [4], [11]. As Figure 2 shows the circuits to be checked are compared using a set of XOR gates followed by the OR detector (so-called miter). In order to disprove the equivalence, it is necessary to identify at least one XOR-gate which evaluates to 1 for an input assignment, i.e. it is necessary to find an input assignment for which the corresponding outputs y_i and y'_i provide different values and thus $z_i = 1$. The resulting circuit, which is composed of Circuit A (reference circuit), Circuit B (candidate circuit) and the miter, is transformed into one Boolean formula in conjunctive normal form (CNF) which is unsatisfiable if and only if circuits A and B are functionally equivalent [20]. The transformation to CNF is conducted gate by gate using the Tseitin's algorithm [21]. Table I contains the CNF representation for selected gates.

C. Optimized Approach for CGP

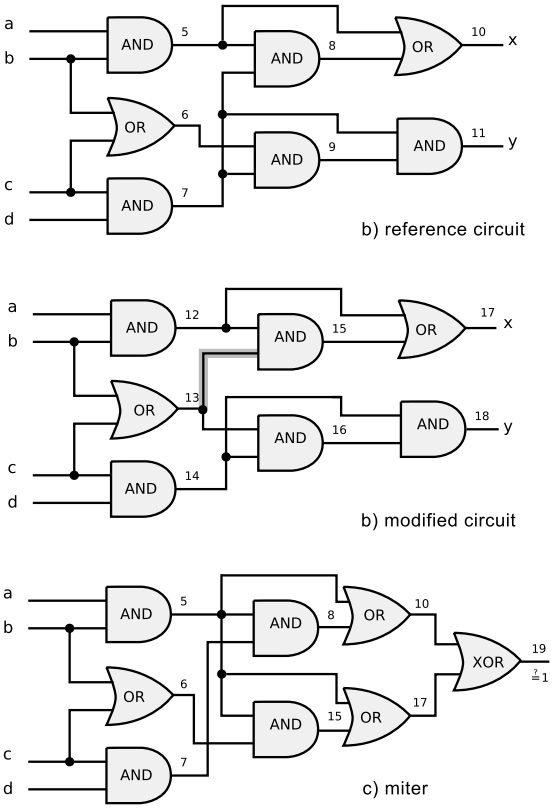
Although the SAT-based equivalence checking applied in the fitness function allows us to optimize large logic circuits using CGP, there exist circuits for which the runtime of the state-of-the-art SAT solvers grows exponentially with increasing the size of problem instance. In order to shorten the decision time, various methods can be applied to reduce the number of clauses for the SAT solver. We proposed to utilize knowledge of genes which have been modified by the mutation operator to calculate a 'difference' between the parent individual and its offspring [11]. Note that this 'difference' circuit is sufficient for checking the functional equivalence of parent circuit and its offspring and thus only the 'difference' is submitted to the SAT solver. An example of a reference (parent) circuit and modified circuit (offspring) is shown in Fig. 3a,b. The 'difference' circuit (Fig. 3c) consists of 8 gates (7 + 1 XOR). This is a significant reduction with respect to the standard all-output approach which led to 17 gates (14 + 2 XOR + 1 OR). Resulting CNF is shown in Fig. 3d.

V. Experiments

The modified selection strategy will be combined with the fast equivalence checking-based fitness function to optimize complex combinational circuits. In order to investigate whether

TABLE I
CNF REPRESENTATION OF SOME COMMON GATES

Gate	Corresponding CNF representation
$y = \mathrm{NOT}(x_1)$	$(\neg y \vee \neg x_1) \wedge (y \vee x_1)$
$y = \mathrm{AND}(x_1, x_2)$	$(y \vee \neg x_1 \vee \neg x_2) \wedge (\neg y \vee x_1) \wedge (\neg y \vee x_2)$
$y = \mathrm{OR}(x_1, x_2)$	$(\neg y \vee x_1 \vee x_2) \wedge (y \vee \neg x_1) \wedge (y \vee \neg x_2)$
$y = \mathrm{XOR}(x_1, x_2)$	$(\neg y \vee \neg x_1 \vee \neg x_2) \wedge (\neg y \vee x_1 \vee x_2) \wedge (y \vee \neg x_1 \vee x_2) \wedge (y \vee x_1 \vee \neg x_2)$
$y = \mathrm{NAND}(x_1, x_2)$	$(\neg y \vee \neg x_1 \vee \neg x_2) \wedge (y \vee x_1) \wedge (y \vee x_2)$
$y = \mathrm{NOR}(x_1, x_2)$	$(y \vee x_1 \vee x_2) \wedge (\neg y \vee \neg x_1) \wedge (\neg y \vee \neg x_2)$



$$(x_1 \vee \neg x_5)(x_2 \vee \neg x_5)(\neg x_1 \vee \neg x_2 \vee x_5)$$
$$(\neg x_2 \vee x_6)(\neg x_3 \vee x_6)(x_2 \vee x_3 \vee \neg x_6)$$
$$(x_3 \vee \neg x_7)(x_4 \vee \neg x_7)(\neg x_3 \vee \neg x_4 \vee x_7)$$
$$(x_5 \vee \neg x_8)(x_7 \vee \neg x_8)(\neg x_5 \vee \neg x_7 \vee x_8)$$
$$(x_5 \vee \neg x_{15})(x_6 \vee \neg x_{15})(\neg x_5 \vee \neg x_6 \vee x_{15})$$
$$(\neg x_5 \vee x_{10})(\neg x_8 \vee x_{10})(x_5 \vee x_8 \vee \neg x_{10})$$
$$(\neg x_5 \vee x_{17})(\neg x_{15} \vee x_{17})(x_5 \vee x_{15} \vee \neg x_{17})$$
$$(\neg x_{10} \vee \neg x_{17})(x_{10} \vee x_{17})$$

d) CNF

Fig. 3. Example of reference circuit (a) and its offspring (b) before a conversion to CNF is performed. A 'difference' circuit (c) is constructed and transformed to CNF (d)

the approach is useful, we will compare two CGP-based optimization methods.

- *Method A* utilizes standard CGP and a fast SAT-based fitness function.
- *Method B* utilizes CGP with modified selection strategy and a fast SAT-based fitness function.

The CGP parameters are identical for both methods: $\Gamma = \{\mathrm{BUFF}, \mathrm{NOT}, \mathrm{AND}, \mathrm{OR}, \mathrm{XOR}, \mathrm{NAND}, \mathrm{NOR}, \mathrm{XNOR}\}$, $\lambda = 2$, $l = N_g$, 2 mutations/chromosome, $n_c = N_g$ and $n_r = 1$ where N_g is the number of gates of the reference (seed) circuit, i.e. the circuit created using conventional synthesis. The effect of setting of the CGP parameters on the quality of optimization will be further investigated in Section VI-A.

The circuits were mapped to the 2-input gates using SIS. Note that ABC and SIS tools are deterministic. We have used them with the standard setting and the aim to minimize the area. In order to improve their results we applied them on their own results iteratively. Implementation cost (i.e. area) of a circuit is calculated as a weighted sum of gates that are utilized in a candidate circuit. The weights which are given in Table II reflect real sizes of corresponding transistor-level implementations.

TABLE II
RELATIVE IMPLEMENTATION COST OF GATES (RELATIVE AREA)

Weight	Gate
0.67	NOT
1.00	NAND, NOR
1.33	AND, OR
1.66	XNOR
2.00	XOR

The MiniSAT 2 (version 070721) has been used as a SAT solver [22] because it can easily be embedded into a custom application.

The experiments were carried out on a cluster consisting of Intel Xeon X5670 2.4 GHz processors using the Sun Grid Engine (SGE) that enables to run the experiments in parallel. All statistical results were calculated from 50 independent 12-hour runs.

VI. RESULTS

A. Setting of CGP parameters

In order to find suitable values for CGP parameters, we have performed initial experiments for selected benchmark circuits. We evaluated the effect of various combinations of

the population size (λ), mutation intensity (h) and l-back parameter. Table III gives relative improvement (in %) of the initial circuit that was achieved using Method A and B. We can observe that increasing of mutation intensity leads to increasing of circuit area which is not desired. Reducing the population size and modifying the l-back parameter influences the quality of optimization insignificantly. This observation holds for Method A as well as Method B. Therefore, the following setting seems to be the most suitable: $\lambda = 2$, $l = n_c = N_g$, $h = 2$, and $n_r = 1$.

TABLE III
RELATIVE IMPROVEMENT (%) OF THE INITIAL CIRCUIT EVALUATED FOR DIFFERENT CGP SETTINGS.

Method A

l	λ	h	apex1	apex2	apex3	apex5	Mean
max	3	2	21.0	33.6	8.5	6.1	17.3
max	3	10	10.1	32.9	3.7	4.3	12.8
max	3	15	7.2	32.4	2.5	3.8	11.5
20	3	2	21.9	33.6	8.8	6.2	17.6
10	3	2	21.7	32.8	8.9	6.0	17.3
max	1	2	22.4	33.9	9.1	6.0	17.9
10	1	2	22.7	33.3	9.5	6.1	17.9

Method B

l	λ	h	apex1	apex2	apex3	apex5	Mean
max	3	2	14.4	58.9	1.6	6.0	20.2
max	3	10	3.3	56.9	0.0	2.4	15.6
max	3	15	0.8	54.8	0.0	1.5	14.3
20	3	2	13.7	58.4	1.2	6.1	19.9
10	3	2	14.7	58.9	1.3	6.2	20.3
max	1	2	15.5	58.4	1.5	6.2	20.4
10	1	2	15.0	58.6	2.0	6.3	20.5

B. Optimization of LGSynth93 Benchmarks

Table IV gives basic parameters of 21 LGSynth93 benchmark circuits. It should be noticed that only circuits with 10 and more inputs were considered for our experiments. For each circuit the number of inputs and outputs is given. The 'Area' column shows the weighted number of gates (relative area) of an initial solution (a CGP seed), i.e. the best result of 100 iterative applications of a conventional synthesis conducted using ABC. Delay is also reported. The number of evaluations (the Eval column) allowed for CGP has resulted from a particular circuit size and a total optimization time available.

Table V and Table VI summarize the results for Method A and Method B in terms of:

- BstF – the best fitness (i.e. the smallest relative area).
- BstF$_D$ – delay of the best obtained circuit.
- BstD – the shortest delay obtained.
- BstD$_F$ – the area for the circuit with the shortest delay
- AvgF – the average fitness (with a standard deviation) calculated out of 50 independent 12-hour runs.
- AvgDly - the average delay
- NDM – the percentage of non-destructive mutations.

Figure 4 shows 12-hour convergence curves for 8 selected circuits that were optimized by Method A and B. The starting point is always the best result of ABC.

TABLE IV
THE SET OF BENCHMARK CIRCUITS. 'AREA' AND 'DELAY' OBTAINED USING ABC. 'EVAL' IS THE NUMBER OF EVALUATIONS ALLOWED FOR CGP.

circuit	N_i	N_o	Area	Delay	Eval$\times 10^6$
alu4	14	8	819.7	16	270
apex1	45	45	1761.3	14	15
apex2	39	3	206.3	13	195
apex3	54	50	1607.0	13	15
apex5	117	88	772.7	11	90
b12	15	9	61.0	6	600
cordic	23	2	55.0	8	405
cps	24	109	1021.7	12	15
duke2	22	29	330.7	11	75
e64	65	65	382.3	8	135
ex1010	10	10	2487.3	14	15
misex2	25	18	110.7	7	420
misex3	14	14	842.6	14	30
misex3c	14	14	499.3	13	60
pdc	16	40	503.0	13	60
sao2	10	4	132.3	9	195
spla	16	46	571.0	14	75
t481	16	1	25.0	5	645
table3	14	14	1382.3	15	15
table5	17	15	1068.0	14	15
vg2	25	8	92.3	9	240

VII. DISCUSSION

A. Comparison of Methods

We can see from Tables V and VI that Method B gives more compact circuits in 11 out of 21 optimized cases. Both methods significantly outperform the conventional ABC tool; however, computational requirements are very different. While iterative application of deterministic ABC quickly leads (in a few seconds/minutes) to a small reduction of circuit size, no further improvements have been observed in next 1 hour. The progress of CGP optimization continues for a longer time. For the cost of a runtime, the results of conventional synthesis were significantly improved for the LGSynth93 benchmarks at the end of optimization (area reduced by 24% on average). We can also observe that all runs produced a very similar result (standard deviation is relatively low) for a particular circuit which is good for practice. Method B could be recommended for optimization of smaller circuits (in terms of area; the number of inputs is not important).

The most remarkable result has been obtained for the alu4 circuit where the relative area was reduced to 106.7 by Method A and 70.8 by Method B from the original value of 819.7. This result confirms that the conventional optimization generates far from optimum solutions for some types of circuits (see the same point in [23], [24], [25]). Hence it is worth to perform this time consuming CGP-based optimization. Note that the results are not directly comparable to our previous work since we measured a relative area in this paper while papers [25], [11] give the number of gates.

In contrast to our assumptions, the significant reduction in the area is not automatically accompanied by a reduction of delay. It can be calculated that delay of the most area-efficient circuits (see the BstF columns) increased by 3.00 (5.19, respectively) for Method A (Method B, respectively)

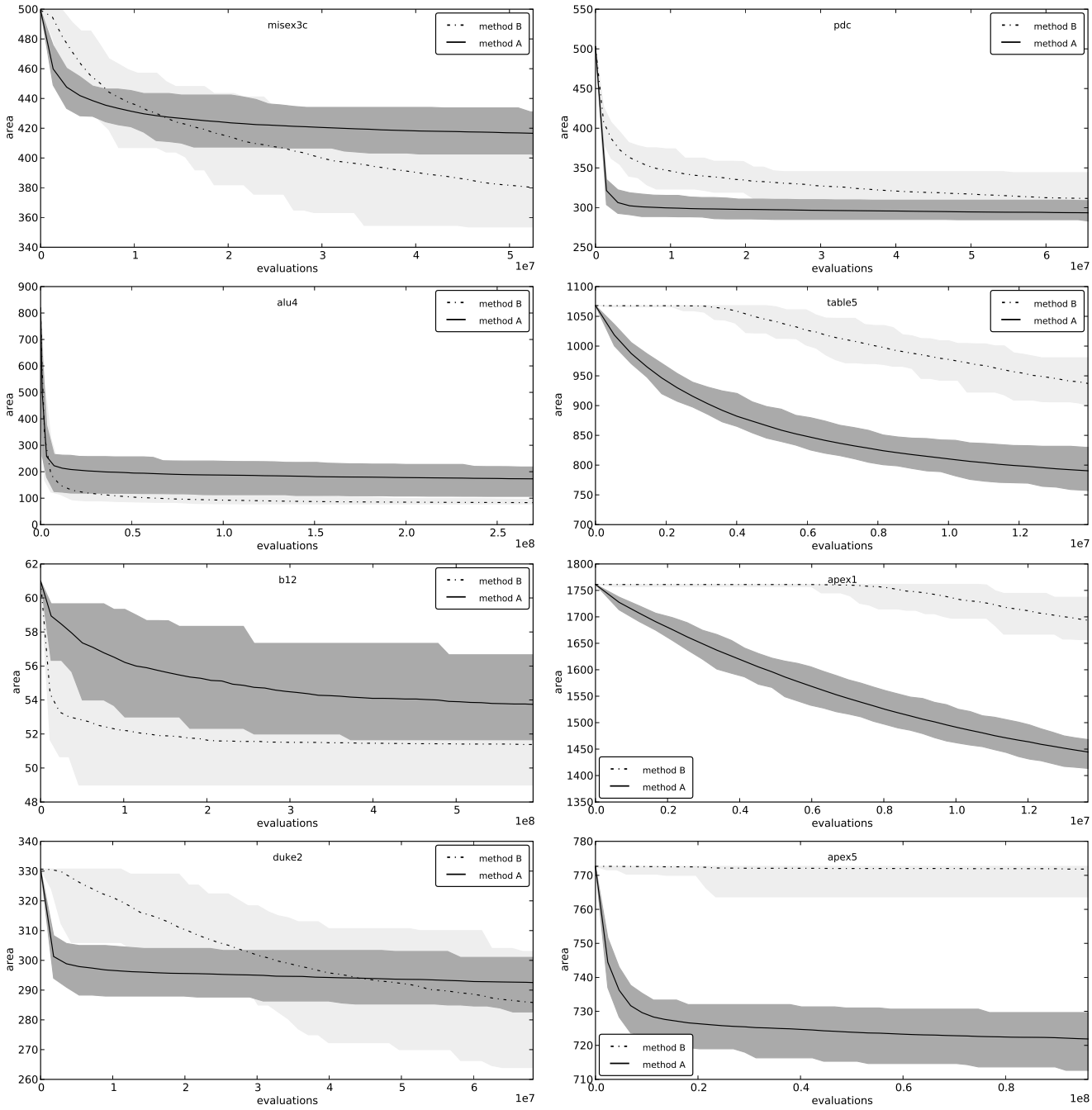


Fig. 4. Convergence curves for selected benchmark problems. The mean, minimum and maximum number of gates from 50 independent runs of Method A and B when seeded using the best circuit obtained from the ABC tool.

in average in comparison with the CGP seeds. Moreover, the circuits showing the shortest delay from all the independent runs exhibit higher delays than the CGP seeds. The average increase is 1.38 for Method A and 2.48 for Method B. Method A provides more stable results in terms of delay than Method B. The reason is that Method B samples more new candidate (however, fully functional) circuits and hence the probability that delay of the offspring is different w.r.t. its parent is higher.

B. Analysis of Selection

Although Method B does not consider the circuit size for selection of a parent individual it still provides better results than Method A for many problem instances. Recall that a fitness landscape is rugged and neutral in case of digital circuit evolution using CGP [26]. In the standard CGP, generating of offspring individuals is biased to the best individual that has been discovered so far. The best individual is changed only if a better or equally-scored solution is discovered. In Method

B, the changes of the parent individual are more frequent because the only requirement for a candidate individual to qualify as a parent is its functional correctness. Hence, we consider Method B as more *explorative* than Method A.

Paper [12] suggests that if a high degree of redundancy is present in the genotype, the modified selection strategy of Method B would generate more functionally correct individuals than Method A. And because the fitness landscape is rugged and neutral, Method B would be more efficient in finding compact circuit implementations than Method A. In order to verify this hypothesis for our case (where large circuits are optimized in contrast to small circuits in paper [12]), we measured the number of non-destructive mutations (NDMs), i.e. neutral-to-fitness and fitness-improving mutations.

Tables V and VI show the average percentage which is taken by NDMs (calculated from all runs). Method B produces 6.2% more NDMs than Method A. If Method B gives a smaller circuit than Method A it holds in 10 out of 11 cases that percentage of NDMs is higher for Method B. On the other hand, if Method A gives smaller circuits than Method B then percentage of NDMs is higher for Method A only in 5 out of 10 cases.

VIII. Conclusions

We evaluated several aspects of the CGP based evolutionary post-synthesis optimization of complex combinational circuits. In particular, we utilized an equivalence checking algorithm in the fitness function to reduce the fitness computation time. In addition, two selection strategies were compared with the aim of reducing the area (the weighted number of gates) in the LGSynth93 benchmark circuits. It was shown that the modified selection strategy leads to more compact circuits in roughly 50% cases in comparison with the original selection strategy of CGP. The average area improvement is 24% with respect to the results of conventional synthesis. The main drawback of the proposed methods is that delay of optimized circuits has increased in many cases. Our plan for future work is to integrate a truly multiobjective optimization engine to the proposed methods.

IX. Acknowledgments

This work was supported by the Czech science foundation project P103/10/1517, the research programme MSM 0021630528, the BUT project FIT-S-11-1 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

References

[1] J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the Evolutionary Design of Digital Circuits – Part I," *Genetic Programming and Evolvable Machines*, vol. 1, no. 1, pp. 8–35, 2000.

[2] V. Vassilev, D. Job, and J. F. Miller, "Towards the Automatic Design of More Efficient Digital Circuits," in *Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware*, J. Lohn, A. Stoica, D. Keymeulen, and S. Colombano, Eds. Los Alamitos, CA, USA: IEEE Computer Society, 2000, pp. 151–160.

[3] J. A. Walker, J. A. Hilder, D. Reid, A. Asenov, S. Roy, C. Millar, and A. M. Tyrrell, "The evolution of standard cell libraries for future technology nodes," *Genetic Programming and Evolvable Machines*, vol. 12, no. 3, pp. 235–256, 2011.

[4] Z. Vasicek and L. Sekanina, "Formal verification of candidate solutions for post-synthesis evolutionary optimization in evolvable hardware," *Genetic Programming and Evolvable Machines*, vol. 12, no. 3, pp. 305–327, 2011.

[5] L. Sekanina, "Evolutionary design of digital circuits: Where are current limits?" in *Proc. of the 1st NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE Computer Society, 2006, pp. 171–178.

[6] E. Stomeo, T. Kalganova, and C. Lambert, "Generalized disjunction decomposition for evolvable hardware," *IEEE Transaction Systems, Man and Cybernetics, Part B*, vol. 36, no. 5, pp. 1024–1043, 2006.

[7] A. P. Shanthi and R. Parthasarathi, "Practical and scalable evolution of digital circuits," *Applied Soft Computing*, vol. 9, no. 2, pp. 618–624, 2009.

[8] X. Yao and T. Higuchi, "Promises and Challenges of Evolvable Hardware," *IEEE Transactions on Systems, Man, and Cybernetics – Part C*, vol. 29, no. 1, pp. 87–97, 1999.

[9] J. Hilder, J. Walker, and A. Tyrrell, "Use of a multi-objective fitness function to improve cartesian genetic programming circuits," in *NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE, 2010, pp. 179–185.

[10] P. Fiser and J. Schmidt, "It is better to run iterative resynthesis on parts of the circuit," in *Proc. of the 19th International Workshop on Logic and Synthesis*. University of California Irvine, 2010, pp. 17–24.

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

[12] Z. Gajda and L. Sekanina, "An efficient selection strategy for digital circuit evolution," in *Evolvable Systems: From Biology to Hardware*, ser. LNCS 6274. Springer Verlag, 2010, pp. 13–24.

[13] Berkley Logic Synthesis and Verification Group, *ABC: A System for Sequential Synthesis and verification*, 2010. [Online]. Available: http://www.eecs.berkeley.edu/˜ alanmi/abc/

[14] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-vincentelli, "Sis: A system for sequential circuit synthesis," University California, Berkeley, Tech. Rep., 1992.

[15] J. F. Miller and P. Thomson, "Cartesian Genetic Programming," in *Proc. of the 3rd European Conference on Genetic Programming EuroGP2000*, ser. LNCS, vol. 1802. Springer, 2000, pp. 121–132.

[16] J. F. Miller and S. L. Smith, "Redundancy and Computational Efficiency in Cartesian Genetic Programming," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 2, pp. 167–174, 2006.

[17] T. Kalganova and J. F. Miller, "Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness," in *The First NASA/DoD Workshop on Evolvable Hardware*. IEEE Computer Society, 1999, pp. 54–63.

[18] Z. Gajda and L. Sekanina, "Reducing the number of transistors in digital circuits using gate-level evolutionary design," in *Genetic and Evolutionary Computation Conference*. ACM, 2007, pp. 245–252.

[19] S. Yanushkevich, D. M. Miller, V. P. Shmerko, and R. S. Stankovic, *Decision Diagram Techniques for Micro- and Nanoelectronic Design Handbook*. CRC, 2006.

[20] E. Goldberg, M. Prasad, and R. Brayton, "Using SAT for combinational equivalence checking," in *DATE '01: Proceedings of the conference on Design, automation and test in Europe*. Piscataway, NJ, USA: IEEE Press, 2001, pp. 114–121.

[21] G. S. Tseitin, "On the complexity of derivation in propositional calculus," in *Studies in Constructive Mathematics and Mathematical Logic, Part II*, 1968, pp. 115–125.

[22] N. Een and N. Sorensson, "MiniSAT." [Online]. Available: http://minisat.se

[23] P. Fiser and J. Schmidt, "Small but nasty logic synthesis examples," in *Proc. 8th Int. Workshop on Boolean Problems*, 2008, pp. 183–190.

[24] J. Cong and K. Minkovich, "Optimality Study of Logic Synthesis for LUT-Based FPGAs," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 230–239, 2007.

[25] P. Fiser, J. Schmidt, Z. Vasicek, and L. Sekanina, "On logic synthesis of conventionally hard to synthesize circuits using genetic programming," in *Proc. of the 13th Int. IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*. IEEE, 2010, pp. 346–351.

[26] J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the Evolutionary Design of Digital Circuits – Part II," *Genetic Programming and Evolvable Machines*, vol. 1, no. 3, pp. 259–288, 2000.

TABLE V

METHOD A: THE BEST AND AVERAGE FITNESS (BSTF, AVGF), THE BEST AND AVERAGE DELAY (BSTD, AVGDLY), DELAY FOR BSTF (BSTF$_D$), AREA FOR BSTD (BSTD$_F$) . THE NDM GIVES % OF NON-DESTRUCTIVE MUTATIONS.

circuit	BstF	BstF$_D$	BstD	BstD$_F$	AvgF	AvgDly	NDM(%)
alu4	106.7	17	13	113.7	173.1 $\pm$ 32.4	17.12 $\pm$ 1.76	68.22 $\pm$ 4.80
apex1	1410.7	19	18	1415.0	1441.5 $\pm$ 14.8	19.22 $\pm$ 0.64	9.89 $\pm$ 0.34
apex2	132.0	15	13	139.0	140.6 $\pm$ 4.3	14.30 $\pm$ 0.75	24.15 $\pm$ 1.43
apex3	1350.0	18	17	1356.3	1369.8 $\pm$ 11.6	18.40 $\pm$ 0.87	9.10 $\pm$ 0.28
apex5	712.7	13	12	715.0	721.9 $\pm$ 3.7	13.14 $\pm$ 0.77	6.78 $\pm$ 0.21
b12	51.7	8	6	53.7	53.7 $\pm$ 1.2	7.88 $\pm$ 1.19	10.58 $\pm$ 0.77
cordic	49.0	8	8	49.0	50.1 $\pm$ 0.4	8.38 $\pm$ 0.56	10.80 $\pm$ 0.53
cps	787.3	17	15	790.0	807.2 $\pm$ 9.3	16.60 $\pm$ 1.18	12.73 $\pm$ 0.38
duke2	282.7	13	12	287.0	292.5 $\pm$ 4.4	13.16 $\pm$ 0.70	10.46 $\pm$ 0.69
e64	262.7	14	11	274.3	278.4 $\pm$ 6.8	12.78 $\pm$ 1.10	18.75 $\pm$ 0.97
ex1010	2382.7	18	17	2390.7	2404.5 $\pm$ 7.7	17.64 $\pm$ 0.71	4.50 $\pm$ 0.09
misex2	85.3	10	8	92.7	95.1 $\pm$ 2.7	9.52 $\pm$ 0.88	10.68 $\pm$ 0.91
misex3	398.0	20	17	428.7	440.8 $\pm$ 16.5	19.14 $\pm$ 1.48	31.60 $\pm$ 1.44
misex3c	400.7	16	14	412.0	416.5 $\pm$ 7.0	16.46 $\pm$ 1.17	11.33 $\pm$ 0.65
pdc	282.7	14	13	295.3	293.4 $\pm$ 6.1	14.68 $\pm$ 0.84	29.82 $\pm$ 0.94
sao2	95.0	12	11	102.3	107.0 $\pm$ 3.9	12.72 $\pm$ 1.00	14.05 $\pm$ 1.46
spla	280.7	18	14	295.0	296.9 $\pm$ 7.1	16.12 $\pm$ 1.07	34.07 $\pm$ 1.03
t481	24.3	5	5	24.3	24.3 $\pm$ 0.0	5.00 $\pm$ 0.00	10.36 $\pm$ 0.00
table3	1055.3	20	18	1066.0	1081.0 $\pm$ 15.2	20.04 $\pm$ 1.04	11.76 $\pm$ 0.49
table5	756.3	18	17	799.0	789.6 $\pm$ 11.6	19.00 $\pm$ 1.02	15.17 $\pm$ 0.49
vg2	83.3	9	9	83.3	85.6 $\pm$ 1.0	10.18 $\pm$ 0.93	8.30 $\pm$ 0.37

TABLE VI

METHOD B: THE BEST AND AVERAGE FITNESS (BSTF, AVGF), THE BEST AND AVERAGE DELAY (BSTD, AVGDLY), DELAY FOR BSTF (BSTF$_D$), AREA FOR BSTD (BSTD$_F$) . THE NDM GIVES % OF NON-DESTRUCTIVE MUTATIONS.

circuit	BstF	BstF$_D$	BstD	BstD$_F$	AvgF	AvgDly	NDM(%)
alu4	78.0	14	12	83.0	83.0 $\pm$ 4.1	14.52 $\pm$ 1.32	83.41 $\pm$ 1.06
apex1	1657.0	21	19	1683.3	1693.7 $\pm$ 16.5	20.64 $\pm$ 0.95	9.18 $\pm$ 0.31
apex2	86.7	16	14	89.3	89.6 $\pm$ 1.5	16.40 $\pm$ 1.04	46.36 $\pm$ 0.61
apex3	1589.7	20	13	1603.3	1605.9 $\pm$ 3.0	13.88 $\pm$ 2.39	8.28 $\pm$ 0.22
apex5	763.7	14	11	772.7	771.8 $\pm$ 1.9	12.02 $\pm$ 1.79	9.00 $\pm$ 0.27
b12	49.0	7	6	51.3	51.4 $\pm$ 0.8	7.80 $\pm$ 1.08	13.54 $\pm$ 0.40
cordic	41.3	8	7	41.7	42.6 $\pm$ 0.6	8.72 $\pm$ 0.83	24.85 $\pm$ 0.51
cps	928.3	19	17	928.7	947.3 $\pm$ 11.2	19.78 $\pm$ 1.79	11.74 $\pm$ 0.36
duke2	264.0	18	15	292.3	285.8 $\pm$ 9.1	17.86 $\pm$ 1.70	16.72 $\pm$ 1.06
e64	237.3	41	26	240.7	245.6 $\pm$ 4.0	34.06 $\pm$ 4.05	27.82 $\pm$ 0.49
ex1010	2486.0	14	14	2486.0	2487.3 $\pm$ 0.2	14.00 $\pm$ 0.00	4.72 $\pm$ 0.08
misex2	78.3	9	8	80.7	81.5 $\pm$ 1.4	10.48 $\pm$ 1.42	21.80 $\pm$ 0.38
misex3	369.3	24	19	407.0	433.9 $\pm$ 32.6	22.58 $\pm$ 1.67	35.79 $\pm$ 2.55
misex3c	353.7	17	16	363.0	380.1 $\pm$ 11.7	19.68 $\pm$ 2.20	20.32 $\pm$ 1.40
pdc	292.3	22	17	309.0	311.5 $\pm$ 10.2	19.96 $\pm$ 1.87	32.52 $\pm$ 0.94
sao2	54.3	9	9	54.3	60.5 $\pm$ 3.4	11.60 $\pm$ 1.33	43.44 $\pm$ 2.81
spla	305.0	17	17	305.0	321.9 $\pm$ 7.3	20.62 $\pm$ 1.82	35.46 $\pm$ 0.83
t481	23.7	5	5	23.7	24.0 $\pm$ 0.2	5.00 $\pm$ 0.00	7.65 $\pm$ 0.08
table3	1290.0	21	20	1309.0	1331.3 $\pm$ 17.7	22.10 $\pm$ 1.17	10.77 $\pm$ 0.37
table5	900.3	21	19	917.0	937.2 $\pm$ 17.8	22.54 $\pm$ 1.42	14.57 $\pm$ 0.61
vg2	75.0	11	10	77.7	78.1 $\pm$ 1.7	12.52 $\pm$ 1.04	16.13 $\pm$ 1.36