Parallel Multi-Objective Evolutionary Design of Approximate Circuits

Radek Hrbacek
Brno University of Technology, Czech republic
Faculty of Information Technology
ihrbacek@fit.vutbr.cz

ABSTRACT

Evolutionary design of digital circuits has been well established in recent years. Besides correct functionality, the demands placed on current circuits include the area of the circuit and its power consumption. By relaxing the functionality requirement, one can obtain more efficient circuits in terms of the area or power consumption at the cost of an error introduced to the output of the circuit. As a result, a variety of trade-offs between error and efficiency can be found. In this paper, a multi-objective evolutionary algorithm for the design of approximate digital circuits is proposed. The scalability of the evolutionary design has been recently improved using parallel implementation of the fitness function and by employing spatially structured evolutionary algorithms. The proposed multi-objective approach uses Cartesian Genetic Programming for the circuit representation and a modified NSGA-II algorithm. Multiple isolated islands are evolving in parallel and the populations are periodically merged and new populations are distributed across the islands. The method is evaluated in the task of approximate arithmetical circuits design.

Categories and Subject Descriptors

B.6.0 [**Hardware**]: Logic Design—*General*; I.2.8 [**Computing methodologies**]: Artificial intelligence—*Problem Solving, Control Methods, and Search*

Keywords

Cartesian Genetic Programming; Parallel Evolutionary Algorithms; Multi-objective Optimization; Cluster; Combinational Circuit Design; Approximate Circuits

1. INTRODUCTION

While evolutionary design of digital circuits has been well established in the past, the correct functionality has always been an essential requirement put on the circuits. The other parameters, like the area, delay or power consumption, have been considered as secondary and have not been optimized as long as a fully working solution has been found. Recently, power efficiency has become the most important parameter of many real circuits. At the same time, a wide range of applications capable of tolerating imperfections (e.g. multimedia) has spread out. As a consequence, a new research field has been brought into being – the *approximate computing* [3].

The approximate digital circuits are designed in such a way that the functionality specification is not fully met in exchange for savings in terms of area, delay, power consumption etc. Although the circuit is not working properly, it can still be suitable for applications in which certain level of error is not recognizable (e.g. human perception and multimedia applications). Moreover, in some cases (e.g. low battery), the users could knowingly tolerate even more inaccuracy in order to extend the battery life.

After the first manual attempts to circuit approximation suffering from low scalability and efficiency [2, 7], a new class of systematic methods has been developed. The Systematic methodology for Automatic Logic Synthesis (SALSA) uses a quality function which decides whether a predefined quality constraint is met. The algorithm is allowed to modify the circuit as long as the quality constraint is not exceeded [19]. Another approach, Substitute-and-Simplify (SASIMI), looks for signal pairs having similar values with a high probability. By substituting one signal for the other, a part of the circuit can be removed resulting in area and power savings at the cost of an error introduced to the output [18].

The aforementioned methods have to be applied repeatedly with different error constraints if a set of trade-offs is demanded. In this paper, we propose a multi-objective evolutionary design approach which is capable of providing a whole set of trade-offs between a set of conflicting objectives. The proposed method is based on Cartesian Genetic Programming (CGP), widely used for the design of digital circuits, and a modified NSGA-II algorithm providing the multi-objective approach.

Since the evolutionary design is very computationally demanding [6], much emphasis has been put to the parallel implementation of the method. In order to make full use of a computer cluster, a spatially structured evolutionary algorithm has been introduced to the design process.

The proposed method has been evaluated in the task of approximate arithmetical circuits design with respect to three objectives – error, area and latency.

GECCO'15, July 11-15, 2015, Madrid, Spain.
Copyright 2015 ACM 978-1-4503-3472-3/15/07 ...$15.00.
http://dx.doi.org/10.1145/2739480.2754785

2. EVOLUTIONARY DESIGN OF DIGITAL CIRCUITS

In our previous work, we used Cartesian genetic programming to either design digital circuits from scratch [6] or to optimize existing circuits [14]. CGP, a branch of genetic programming, has been introduced by Miller [9]. While GP uses tree representation, an individual in CGP is represented by a directed acyclic graph of a fixed size. The candidate solution can have multiple outputs and intermediate results can be reused, which makes CGP very suitable for the design of digital circuits, e.g. arithmetic and logic circuits, digital filters, cryptography related Boolean functions, etc. [10, 5].

CGP uses a fixed-sized cartesian grid of $n_\mathrm{r} \times n_\mathrm{c}$ nodes interconnected by a feed-forward network (see Figure 1). Node inputs can be connected either to one of n_i primary inputs or to an output of a node in preceding l columns. Each node has a fixed number of inputs n_ni (usually $n_\mathrm{ni} = 2$) and can perform one of the functions from the set Γ. Each of n_o primary circuit outputs can be connected either to a primary input or to a node's output. The area and delay of the circuit can be constrained by changing the grid size and the l-back parameter.

The genotype is of fixed length, whereas the phenotype is of variable length depending on the number of inactive nodes, i.e. nodes whose output is not used by any other node or primary output. This implies the existence of individuals with different genotypes but the same phenotypes, which is usually referred to as neutrality [20]. It was shown that for certain problems the neutrality significantly reduces the computational effort and helps to find more innovative solutions [8].

CGP uses a simple mutation based $(1 + \lambda)$ evolutionary strategy as a search mechanism. The population size $1 + \lambda$ is mostly very small, typically, $\lambda = 4$. The initial population is constructed either randomly (evolutionary design) or by mapping of a known solution to the CGP chromosome (evolutionary optimization). In each generation, the best individual is passed to the next generation unmodified along with its λ offspring individuals created by means of point mutation operator. In case more individuals with the best fitness exist, a randomly selected one is chosen. The mutation rate m is usually set to modify up to 5 % randomly selected genes.

In the case of digital circuit evolution, the fitness function usually corresponds to the quality of the candidate circuit measured as the number of correct output bits compared to a specified truth table (i.e. the Hamming distance). In order to obtain a fully working circuit, all combinations of input values have to be evaluated. For a circuit with n_i inputs and n_o outputs, 2^{n_i} test vectors need to be fetched to the primary inputs and $n_\mathrm{o} \cdot 2^{n_\mathrm{i}}$ output bits have to be verified so as to compute the fitness value.

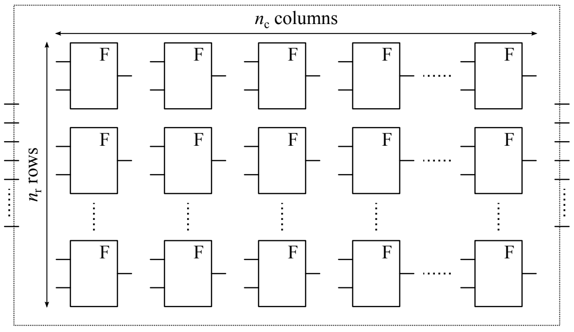


Figure 1: Cartesian genetic programming scheme.

The fitness calculation is computationally very intensive, since the number of test vectors grows exponentially with the number of primary inputs. Recently, it has been sped up by applying parallelism at various levels (data, thread, process) [6, 5] or by introducing formal methods, e.g. SAT solvers [14] or Binary Decision Diagrams (BDD) [16].

Besides the correct functionality, the demands placed on current digital circuits include the area of the circuit and its power consumption. The power consumption of digital circuits consists of two major components – the static and the dynamic power dissipation. The dynamic dissipation occurs while changing the state of the gates and thus it is highly dependent on the character of the gate's input signals. On the contrary, the static power consumption is rather constant and depends on the area of the circuit. The static dissipation has been substantially reduced by introducing the CMOS technology, which uses complementary connected transistors. However, with the decreasing size of the semiconductor technology process, the static dissipation is increasing due to rising leakage currents and is becoming the major component of the power consumption. Therefore, when evolving digital circuits with respect to the power consumption, the area of the circuit can be used to estimate the power consumption [13].

Another important characteristic of a digital circuit is the latency, i.e. the interval between the stimulation of the inputs and the response on the outputs. The latency can be determined by finding the longest path from the inputs to the outputs with respect to particular latencies of the gates along the path. Since the propagation delay of a gate differs for different transitions on the inputs, computing the total circuit latency would require simulating all possible input transitions on the whole circuit. The number of different transitions N_t grows rapidly with the number of primary inputs n_i: $N_\mathrm{t} = 2^{n_\mathrm{i}} \cdot (2^{n_\mathrm{i}} - 1)$. Applying all these combinations is computationally intensive, therefore, an estimation must be used instead. In this paper, we assign each node function a fixed latency. When computing the overall circuit latency, the longest path from an input to an output considering the latencies of all gates along the path is considered.

2.1 Approximate circuits

Many computer systems or programs have the ability to tolerate some loss of accuracy or quality in the computational process and still produce meaningful and useful results. Significant area or energy-efficiency improvements can be achieved by relaxing the functionality requirement. For example, the growing popularity of portable multimedia devices offers a great scope for approximate computation, since human perception is limited and the users are ready to tolerate degraded quality of the multimedia content (e.g. video playback) in exchange for longer battery life. Automatic approximate computing techniques are being developed to speed up the design process and to find the trade-offs between the resources being shrunk (e.g. energy, time, area) and the inaccuracy of the computation.

Recently, several single-objective evolutionary approaches to design approximate circuits have been introduced [12]. Different error metrics have been utilized, starting with the

Hamming distance and introducing new metrics more suitable for arithmetical circuits, e.g. the worst case error, mean absolute error, relative error etc. [15]. In this paper, we use a compromise error metric which penalizes both the mean error and the isolated deviations – the mean squared error:

$$f_{\text{mse}} := \frac{\sum_{\forall i} \left(O_{\text{orig}}^{(i)} - O_{\text{approx}}^{(i)} \right)^2}{2^{n_{\text{i}}}}, \qquad (1)$$

where $O_{\text{orig}}^{(i)}$ is the decimal representation of the i-th circuit correct output and $O_{\text{approx}}^{(i)}$ is the individual's i-th output. The choice, which error metric to use, always depends on a concrete application.

Although several attempts to use multi-objective evolutionary algorithms exist [11], recent techniques have been based mainly on multi-phase single-objective approaches [15] or have used constrained resources in order to find approximate circuits with smaller area or power consumption [12, 17].

3. MULTI-OBJECTIVE CGP

Unlike the single-objective optimization, which enables to compare any two candidate solutions and decide which one is better, the multi-objective optimization leads to the existence of a whole range of trade-off solutions, if the objectives are conflicting. In the case of digital circuits design, the better the circuit works, the larger area and power consumption it has.

Many multi-objective evolutionary algorithms have been proposed, most of them are based on the idea of *Pareto dominance*. The solution p *dominates* the solution q if p is no worse than q in all objectives and p is strictly better than q in at least one objective. The principle can be seen in Figure 2, the Pareto optimal solutions are not dominated by any other solutions and form the so called *Pareto front.*

3.1 NSGA-II and its modifications

One of the most popular multi-objective evolutionary algorithms is the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [1]. It is based on sorting individuals according to the dominance relation into multiple fronts. The first front F_0 contains all Pareto optimal solutions. Each subsequent front F_i is constructed by removing all the preceding fronts from the population and finding a new Pareto front. Each solution is assigned a *rank* according to the front it belongs to, the solutions from the front F_i have the rank equal to i. The NSGA-II fast non-dominated sort (see Algorithm 1) is

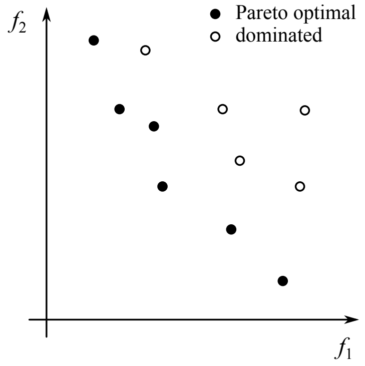


Figure 2: Pareto optimal and dominated solutions.

very efficient, the overall complexity is $\mathcal{O}(MN^2)$, where N is the population size and M is the number of objectives.

```
fast-non-dominated-sort($P$)
foreach $p \in P$ do
    $F_0 = \emptyset$
    $S_p = \emptyset$
    $n_p = 0$
    foreach $q \in P$ do
        if $p \prec q$ then
            $S_p = S_p \cup \{q\}$
        end
        else if $q \prec p$ then
            $n_p = n_p + 1$
        end
    end
    if $n_p = 0$ then
        $p_{\text{rank}} = 0$
        $F_0 = F_0 \cup \{p\}$
    end
end
$i = 0$
while $F_i \neq \emptyset$ do
    $Q = \emptyset$
    foreach $p \in F_i$ do
        foreach $q \in S_p$ do
            $n_q = n_q - 1$
            if $n_q = 0$ then
                $q_{\text{rank}} = i + 1$
                $Q = Q \cup \{q\}$
            end
        end
    end
    $i = i + 1$
    $F_i = Q$
end
$F = (F_0, F_1, \dots)$
return $F$
```

Algorithm 1: Fast non-dominated sort.

The solutions within the individual fronts are sorted according to the *crowding distance* metric, which helps to preserve a reasonable diversity along the fronts [1]. The crowding distance is the average distance of two solutions on either side along each of the objectives. The boundary solutions are assigned an infinite crowding distance, which ensures that these solutions will dominate the inner solutions (see Algorithm 2).

```
crowding-distance-assignment($P$)
$l = |I|$
foreach $p \in P$ do
    $p_{\text{dist}} = 0$
end
foreach objective $m$ do
    $P = \text{sort}(P, m)$
    $P[0]_{\text{dist}} = \infty$
    $P[l-1]_{\text{dist}} = \infty$ for $i$ in $1$ to $l-2$ do
        $P[i]_{\text{dist}} = P[i]_{\text{dist}} +$
            $(P[i+1]_m - P[i-1]_m)/(f_m^{\max} - f_m^{\min})$
    end
end
```

Algorithm 2: Crowding distance assignment.

Any solution from the front F_i always dominate any solution from F_j, $j > i$. Within the fronts, solutions with higher crowding distance are preferred.

Most real applications require to be able to constraint the solutions on particular objectives. NSGA-II offers a simple way to handle the constraints and keep the algorithm complexity low. Each solution can be either *feasible* or *infeasible*, the infeasible solutions are assigned a *constraint violation* according to the Algorithm 3. The constraints on the objective m are denoted by $\langle c_m^{\min}, c_m^{\max} \rangle$.

```
constraint-violation-assignment($P$)
foreach $p \in P$ do
    $p_{\text{constr\_viol}} = 0$
    foreach objective $m$ do
        if $p_m < c_m^{\min}$ then
            $p_{\text{constr\_viol}} = p_{\text{constr\_viol}} + (c_m^{\min} - p_m)/f_m^{\max}$
        end
        if $p_m > c_m^{\max}$ then
            $p_{\text{constr\_viol}} = p_{\text{constr\_viol}} + (p_m - c_m^{\max})/f_m^{\max}$
        end
    end
end
```

Algorithm 3: Constraint violation assignment.

When comparing two solutions, a feasible solution is always preferred. If both solutions are infeasible, the solution with smaller constraint violation is better. In the opposite case, when both solutions are feasible, the dominance depends on the rank and the crowding distance.

Since the original NSGA-II algorithm was based on a genetic algorithm, there must have been changes to use it with CGP [4, 11]. Firstly, due to the absence of the crossover operator in CGP, the offspring population is constructed only using mutation. Secondly, the crowding distance is often not sufficient for CGP to maintain the diversity of the population. The neutrality present in CGP causes a premature convergence, the Pareto fronts are flooded by individuals that are genotypically distinct but phenotypically identical. We propose to introduce a new *equivalence rank*, which enables to put the equivalent solutions in an order and preserve the neutrality character of the CGP. The principle can be seen from Algorithm 4. At the beginning, the population is randomly shuffled. Then, for each individual, the equivalence rank of all individuals (except for already processed ones) with the same fitnesses is incremented.

```
equivalence-rank-assignment($P$)
foreach $p \in P$ do
    $p_{\text{eq\_rank}} = 0$
end
random_shuffle($P$)
$Q = P$
foreach $p \in P$ do
    $Q = Q \setminus \{p\}$
    foreach $q \in Q$ do
        if $p \equiv q$ then
            $q_{\text{eq\_rank}} = q_{\text{eq\_rank}} + 1$
        end
    end
end
```

Algorithm 4: Equivalence rank assignment.

When comparing two individuals, the individual with a lower equivalence rank always dominates the other one. Two individuals with the same equivalence rank are compared using the standard constrained-domination rules. As a consequence, none of the fronts contains individuals with the same fitness and the dominance relation among the individuals with the same fitness is random.

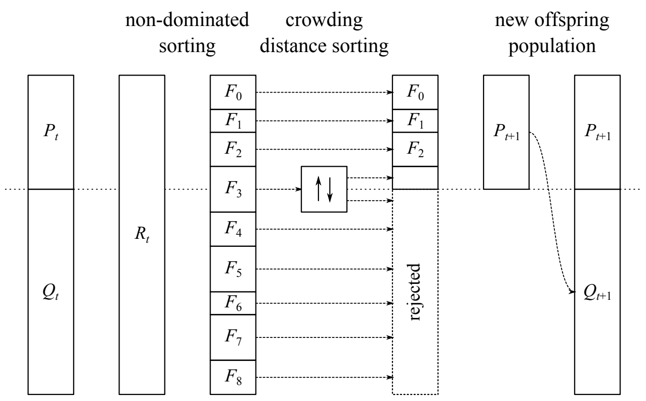


Figure 3: NSGA-II algorithm scheme.

Unlike the original NSGA-II algorithm, which uses a population of parents P and an offspring population Q, both of size N, our modification enables us to set the offspring size independently. In each generation, N_{p} individuals are selected as parents and N_{q} offspring individuals are created by means of mutation. Besides the tournament selection, we use a new deterministic selection mechanism, which cyclically takes the individuals from the parent population P and creates mutants.

The aforementioned principles make the multi-objective approach even more similar to the standard CGP. The overall algorithm works as follows:

```
nsga-ii($P_t$, $Q_t$)
$R_t = P_t \cup Q_t$
equivalence-rank-assignment($R_t$)
constraint-violation-assignment($R_t$)
$F$ = fast-non-dominated-sort($R_t$)
$P_{t+1} = \emptyset$
$i = 0$
while $|P_{t+1}| + |F_i| \leq N_{\text{p}}$ do
    crowding-distance-assignment($F_i$)
    $P_{t+1} = P_{t+1} \cup F_i$
    $i = i + 1$
end
crowding-distance-assignment($F_i$)
sort($F_i$, $\prec_{\text{n}}$)
$P_{t+1} = P_{t+1} \cup F_i\ [0 : (N_{\text{p}} - |P_{t+1}| - 1)]$
$Q_{t+1}$ = create-offspring($P_{t+1}$)
$t = t + 1$
```

Algorithm 5: Modified NSGA-II.

In each generation t, the populations P_t and Q_t form an unified population R_t. The individuals in R_t are assigned the equivalence rank and the crowding distance. Then, the Pareto fronts are identified and the new parental population P_{t+1} is filled with the individuals from the first fronts until P_{t+1} is not overcrowded. The individuals from the last used Pareto front are sorted using the crowding distance and a fraction of them is selected just to fill the population P_{t+1} (Figure 3).

4. PARALLEL MULTI-OBJECTIVE CGP

The evolutionary design is a very computationally demanding approach. In order to reduce the design time, one has to deal with a parallel implementation of the fitness function or search algorithm modifications. In our previous work, we have introduced parallelism at various levels (instruction, data, thread and process) to the CGP and sped up the design process significantly [6, 5]. However, the work was focused on single-objective design of (non-approximate) digital circuits having several specifics, e.g. small population size or the Hamming distance as the fitness function.

In the case of approximate arithmetical circuits, the fitness function (Equation 1) is much more computationally demanding and the implementation much less efficient. On the other hand, the population size of the multi-objective approach is much bigger, which makes the parallel processing of the individuals more efficient.

Besides the parallel implementation of the fitness evaluation, additional speed-up can be achieved by employing spatially structured evolutionary algorithms. Since CGP does not use any crossover operator, there is not a large scope of methods. However, a simple isolated islands model with a periodical exchange of the best individuals across the islands has been confirmed to be beneficial [6].

We propose to extend the multi-objective approach by introducing the isolated islands model. Unlike the single-objective case, the multi-objective algorithm requires to exchange the whole population. The Algorithm 6 is very similar to the single-population case, the only difference is that each G_r generations the populations are unified across the islands and a common Pareto front is identified on each island. Since the equivalence rank is assigned randomly to the individuals with the same fitness, the Pareto fronts on individual islands are phenotypically identical, but genotypically distinct. This principle should avoid the algorithm to converge prematurely and help to preserve the diversity across the populations.

```
nsga-ii-islands($P_t$, $Q_t$)
$R_t = P_t \cup Q_t$
if $t$ mod $G_r = 0$ then
    $R_t$ = MPI_Allgather($R_t$)
end
equivalence-rank-assignment($R_t$)
constraint-violation-assignment($R_t$)
$F$ = fast-non-dominated-sort($R_t$)
$P_{t+1} = \emptyset$
$i = 0$
while $|P_{t+1}| + |F_i| \le N_p$ do
    crowding-distance-assignment($F_i$)
    $P_{t+1} = P_{t+1} \cup F_i$
    $i = i + 1$
end
crowding-distance-assignment($F_i$)
sort($F_i$, $\prec_n$)
$P_{t+1} = P_{t+1} \cup F_i\,[0 : (N_p - |P_{t+1}| - 1)]$
$Q_{t+1}$ = create-offspring($P_{t+1}$)
$t = t + 1$
```

Algorithm 6: NSGA-II with the isolated islands model.

5. EXPERIMENTAL RESULTS

In this section, experiments regarding the multi-objective design of arithmetical circuits are presented and the proposed modifications to the NSGA-II algorithm are examined. All experiments were performed on a computer cluster of 180 nodes with the following hardware configuration: 2× 8-core Intel E5-2665, 64 GB RAM, connected by Infiniband links. Each node was fully loaded with 16 threads, the evaluation of the population was parallelized using OpenMP.

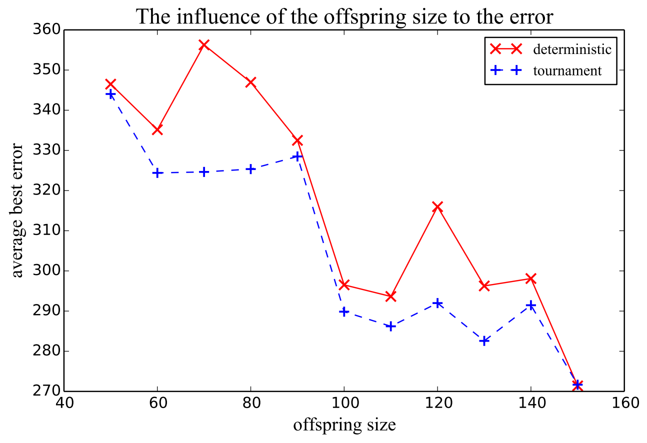
The circuits were design with respect to 3 objectives – the mean squared error (as defined in Equation 1), the area of the circuit (approximate number of transistors considering common CMOS gates) and the latency (see Section 2). The CGP parameters were set similarly to the single-objective case [6], i.e. $\Gamma = \{\texttt{BUF}, \texttt{NOT}, \texttt{AND}, \texttt{OR}, \texttt{XOR}, \texttt{NAND}, \texttt{NOR}, \texttt{XNOR}\}$ and the mutation rate was set to 5 %, we used a linear CGP ($n_r = 1$). The number of columns was $n_c = 800$ in the case of the 4-bit multiplier and $n_c = 100$ in the case of the adder. All experimental results were obtained by running 100 independent evolutionary runs.

5.1 Selection type

In Section 3.1, we have proposed a modification to the NSGA-II selection mechanism – a new deterministic selection. Furthermore, the offspring size N_q is not necessarily equal to the parental population size N_p. In order to draw a comparison between the original tournament selection and the new deterministic selection, a number of experiments were carried out, the task was to design a combinational 4-bit multiplier.

The parental population size was set to $N_p = 50$ and the offspring size was $N_q \in \{50, 60, \ldots, 150\}$, the generation count was $G = 5000$. The same experiments were run for both tournament and deterministic selection.

The results can be seen in Figure 4. The quality of the resulting Pareto front was measured in terms of the Pareto front size (number of individuals) and the best error achieved. No matter how imperfect such comparison is, some trends can be inferred. The tournament selection is beneficial in the



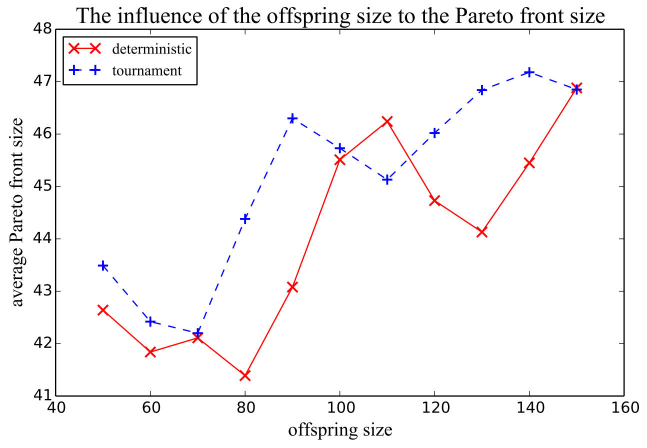
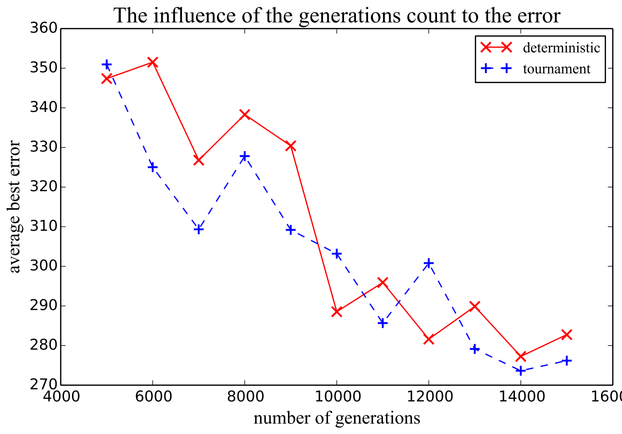


Figure 4: The influence of the offspring size.



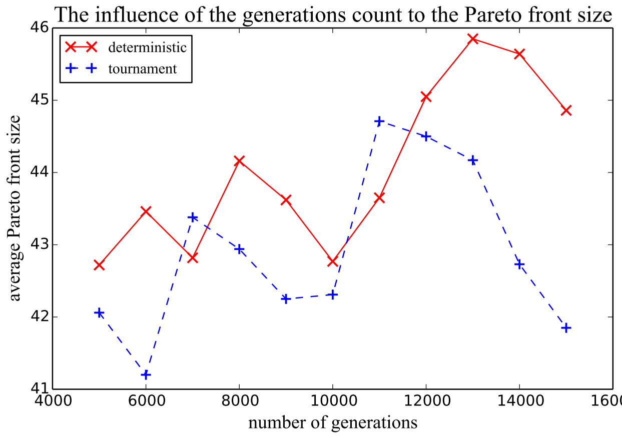


Figure 5: The influence of the number of generations.

cases, when N_q is not an integer multiple of N_p. Otherwise, both selection types evince about the same performance.

Similarly to the single-objective case, one can adjust the number of generation and the population size so as the computational effort is the same. In our second experiment, the population size was fixed ($N_p = 50$, $N_q = 50$), but the number of generations was $G \in \{5000, 6000, \ldots, 15000\}$.

As can be seen in Figure 5, there is no conclusive difference between the two selection types. However, when comparing with the previous experiment (Figure 4), increasing the offspring size seems to be slightly more advantageous than increasing the number of generations.

5.2 Number of Islands

Recently, we have shown that a spatially structured evolutionary algorithm can speed up the evolutionary design of combinational circuits in comparison with a single-population process [6]. The Figure 6 shows the influence of the number of islands on the mean error achieved during the evolutionary design of a 4-bit multiplier. The single-population approach was compared to the multiple islands model with 2, 4 and 8 islands. The populations (each of $N_p = 100$ parental and $N_q = 100$ offspring individuals) were exchanged every $G_r = 500$ generations across the islands. It can be seen that increasing the number of islands significantly reduces the mean error during the whole evolutionary process. The more islands, the less generations is needed to achieve comparable results.

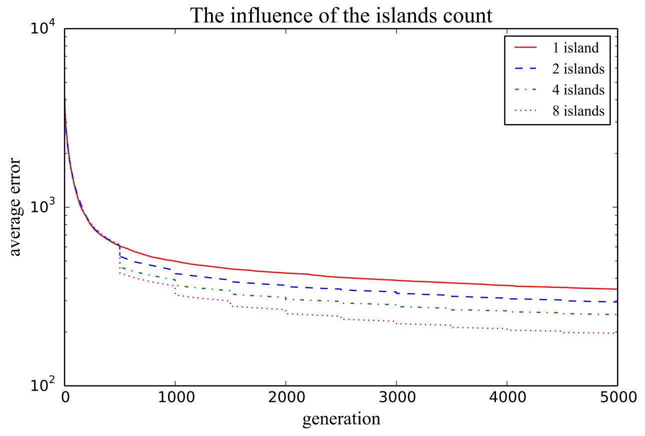


Figure 6: The influence of the number of islands.

5.3 Examples of Evolved Circuits

In this section, the proposed multi-objective approach is demonstrated on two examples of arithmetical circuits. Figure 8 shows the Pareto front of approximate 4-bit multipliers obtained after 1000000 generations. The population size was set to $N_p = 100$ parental and $N_q = 300$ offspring individuals and tournament selection was used to create the offspring population. Eight islands were exchanging the populations every $G_r = 1000$ generations. At the end of the evolution, 66 trade-off solutions were found having the mean squared error from 46.16 to 153.75, the area from 0 to 512 transistors and the latency from 0 to 19 gates. The extremely erroneous solutions are not shown in the Pareto front in Figure 8. Table 1 shows the output errors for all input combinations.

The same experimental setup was used for the design of combinational 4-bit adders. Since combinational adders are much less complex circuits than the multipliers, the number of generations was set to 100000. The resulting Pareto front can be seen in Figure 9, Table 2 shows the output errors for the individual with the lowest error. In comparison with the multiplier, the error is significantly smaller and the circuit has much smaller area (70 transistors) and latency (3 gates). The wiring diagram is depicted in Figure 7.

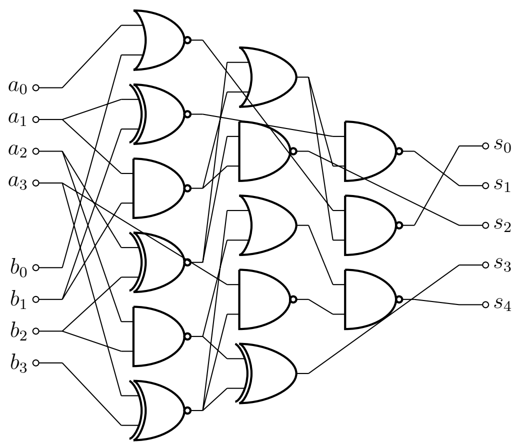


Figure 7: The best 4-bit adder diagram.

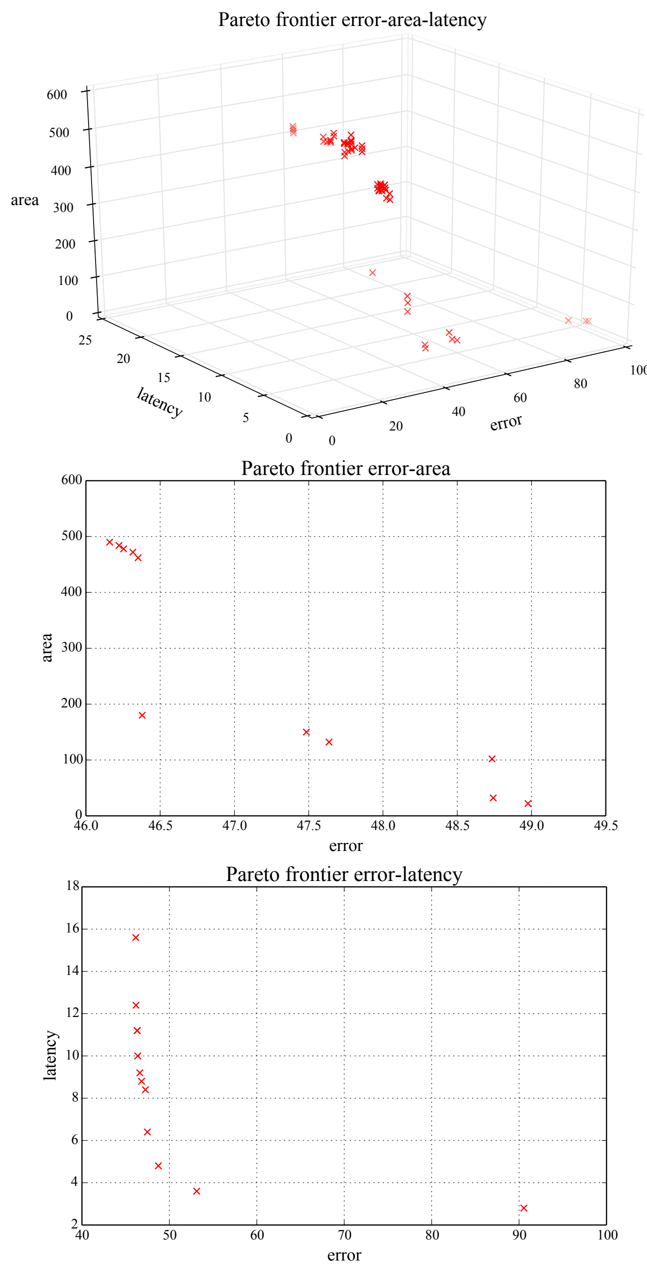


Figure 8: Combinational 4-bit multiplier.

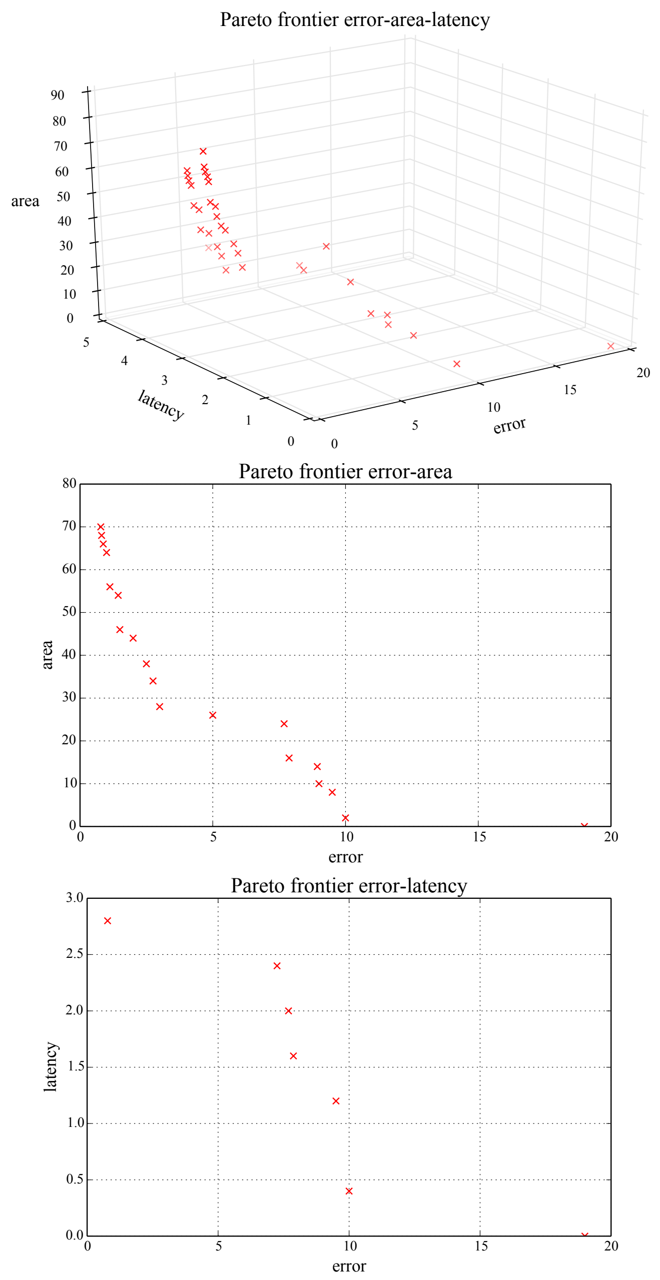


Figure 9: Combinational 4-bit adder.

·	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1
2	0	2	0	1	0	2	0	1	0	2	0	1	0	2	0	1
3	0	1	1	-2	0	0	-3	-6	0	1	1	-2	-5	-8	-11	-14
4	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	0
5	0	-1	0	0	0	-2	0	-4	-1	2	-3	-8	-5	-2	-7	-12
6	0	-1	0	-3	0	1	-5	-11	0	-1	0	-3	8	2	4	1
7	0	0	1	-6	0	-4	-11	-18	-1	0	-7	-14	0	1	-3	-10
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	1	0	1	0	-1	2	0	0	-3	0	-4	3	-6	1	-8
10	0	2	0	1	0	-3	3	-7	0	3	-5	-15	0	-3	-13	-23
11	0	3	0	-2	0	-8	-3	-14	0	-4	-15	-26	-5	-16	-27	-38
12	0	0	0	-5	0	0	8	4	0	0	0	-5	16	4	0	1
13	0	1	0	-8	0	-2	2	1	-1	-6	-3	-16	4	-1	2	-4
14	0	-1	3	-11	0	-7	0	-3	0	-1	-13	-27	8	1	-5	-19
15	0	0	1	-14	0	-12	-3	-10	-1	-8	-23	-38	0	-4	14	-1

Table 1: Error table for the best multiplier.

+	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1
2	0	0	0	0	0	0	-1	-2	0	0	0	0	0	0	-1	-2
3	0	-1	0	-1	0	-1	-2	-3	0	-1	0	-1	0	-1	-2	-3
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1
6	0	0	-1	-2	0	0	0	0	0	0	-1	-2	0	0	0	0
7	0	-1	-2	-3	0	-1	0	-1	0	-1	-2	-3	0	-1	0	-1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1
10	0	0	0	0	0	0	-1	-2	0	0	0	0	0	0	-1	-2
11	0	-1	0	-1	0	-1	-2	-3	0	-1	0	-1	0	-1	-2	-3
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1	0	-1
14	0	0	-1	-2	0	0	0	0	0	0	-1	-2	0	0	0	0
15	0	-1	-2	-3	0	-1	0	-1	0	-1	-2	-3	0	-1	0	-1

Table 2: Error table for the best adder.

6. CONCLUSIONS

Recently, a new application area for evolutionary algorithms has emerged, EAs have been confirmed to be competitive in the task of approximate circuits design [17]. The evolutionary design is a computationally demanding task, therefore, several approaches to speed up the entire process have been proposed [6].

In this paper, a new multi-objective evolutionary method for designing approximate digital circuits has been presented. The method is based on the well-known NSGA-II algorithm modified in order to be more suitable for the use with CGP. Besides a parallel implementation of the population fitness evaluation, a simple spatially structured algorithm is introduced for the purpose of speeding up the evolutionary process.

The proposed method has been evaluated in the task of approximate combinational multiplier and adder design. In comparison with existing methods, the multi-objective approach enables to obtain a set of Pareto optimal solutions in a single run.

In our future research, we will focus on increasing the scalability of the method in order to be able to design more complex circuits. For that purpose, the fitness function needs to be accelerated.

7. ACKNOWLEDGMENTS

This work was supported by the Czech Science Foundation project 14-04197S. The author thanks the IT4Innovations Centre of Excellence for enabling these experiments.

8. REFERENCES

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii, Apr. 2002.

[2] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. Low-power digital signal processing using approximate adders. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(1):124–137, Jan 2013.

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

[4] J. Hilder, J. Walker, and A. Tyrrell. Use of a multi-objective fitness function to improve cartesian genetic programming circuits. In *Adaptive Hardware and Systems (AHS), 2010 NASA/ESA Conference on*, pages 179–185, June 2010.

[5] R. Hrbacek. Bent functions synthesis on xeon phi coprocessor. In *Mathematical and Engineering Methods in Computer Science*, LNCS 8934, pages 88–99. Springer Verlag, 2014.

[6] R. Hrbacek and L. Sekanina. Towards highly optimized cartesian genetic programming: From sequential via simd and thread to massive parallel implementation. In *GECCO '14 Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 1015–1022. Association for Computing Machinery, 2014.

[7] P. Kulkarni, P. Gupta, and M. Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, pages 346–351, Jan 2011.

[8] J. Miller and S. Smith. Redundancy and computational efficiency in cartesian genetic programming. *Evolutionary Computation, IEEE Transactions on*, 10(2):167–174, 2006.

[9] J. Miller and P. Thomson. Cartesian genetic programming. In *Genetic Programming*, volume 1802 of *Lecture Notes in Computer Science*, pages 121–132. Springer Berlin Heidelberg, 2000.

[10] J. F. Miller, editor. *Cartesian Genetic Programming.* Natural Computing Series. Springer Verlag, 2011.

[11] J. Petrlík and L. Sekanina. Multiobjective evolution of approximate multiple constant multipliers. In *IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems 2013*, pages 116–119. IEEE Computer Society, 2013.

[12] L. Sekanina and Z. Vasicek. Approximate circuits by means of evolvable hardware. In *2013 IEEE International Conference on Evolvable Systems (ICES)*, Proceedings of the 2013 IEEE Symposium Series on Computational Intelligence (SSCI), pages 21–28. IEEE Computer Society, 2013.

[13] L. Sekanina and Z. Vasicek. Evolutionary computing in approximate circuit design and optimization. In *1st Workshop on Approximate Computing (WAPCO 2015)*, pages 1–6, 2015.

[14] Z. Vasicek and L. Sekanina. Formal verification of candidate solutions for post-synthesis evolutionary optimization in evolvable hardware. *Genetic Programming and Evolvable Machines*, 12(3):305–327, 2011.

[15] Z. Vasicek and L. Sekanina. Evolutionary design of approximate multipliers under different error metrics. In *17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 135–140. IEEE Computer Society, 2014.

[16] Z. Vasicek and L. Sekanina. How to evolve complex combinational circuits from scratch? In *2014 IEEE International Conference on Evolvable Systems Proceedings*, pages 133–140. Institute of Electrical and Electronics Engineers, 2014.

[17] Z. Vasicek and L. Sekanina. Evolutionary approach to approximate digital circuits design. *IEEE Transactions on Evolutionary Computation*, 99(99):1–13, 2015.

[18] S. Venkataramani, K. Roy, and A. Raghunathan. Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pages 1367–1372, March 2013.

[19] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan. Salsa: Systematic logic synthesis of approximate circuits. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 796–801, June 2012.

[20] T. Yu and J. Miller. Finding needles in haystacks is not hard with neutrality. In *Genetic Programming*, volume 2278 of *Lecture Notes in Computer Science*, pages 13–25. Springer Berlin Heidelberg, 2002.