MULTIOBJECTIVE INTRINSIC HARDWARE EVOLUTION

Paul Kaufmann, Marco Platzner

University of Paderborn
Warburger Str. 100
33098 Paderborn, Germany
email: {paul.kaufmann,platzner}@upb.de

ABSTRACT

Evolutionary design of digital circuits reveals the potential to provide autonomous systems with the properties of self-adaptation, self-optimization and functional recovery. Especially for functions that are rated by a data-dependent and time-varying functional quality, rather than an absolute measure of correctness, evolutionary techniques are attractive.

In this paper, we present a novel approach to evolvable embedded systems that is able to adapt to both slow and radical changes in the environment and the system state, respectively. First, a multiobjective evolutionary search algorithm with a selection scheme based on Pareto dominance is used to compute a set of reasonable trade-offs. Then, the decision is made which solution to use for the present situation. During operation, the system adapts to slowly changing environmental conditions by the evolutionary search process. To handle radical changes, precomputed dominant solutions are stored in the system. When a radical change occurs, the system switches to a "good-enough" solution, and the online evolutionary process is restarted.

In our work, we treat evolutionary circuit design as a multiobjective optimization problem that tries to evolve functionally good or correct FPGA circuits with minimal area and maximal speed. Additionally, the optimization problem can be specified by constraints as, for example, usable area of a chip or a minimal functional quality of a circuit.

1. INTRODUCTION

In the last decades, *natural computing* methods which take problem solving principles from nature have gained popularity. Among others, natural computing includes evolutionary computing. Evolutionary computing covers population-based, stochastic search algorithms inspired by principles from evolution theory. An evolutionary algorithm tries to solve a problem by keeping a set (population) of candidate solutions (individuals) in parallel and improving the quality (fitness) of the individuals over a number of iterations (generations). To form a new generation, genetically-inspired operators such as crossover and mutation are applied to the individuals. A fitness-based selection process steers the population towards better candidates.

Evolvable hardware denotes the combination of evolutionary algorithms with reconfigurable hardware technology to construct self-adaptive and self-optimizing hardware systems. The term evolvable hardware was coined by de Garis [1] and Higuchi [2] in 1993. Evolvable hardware is still a rather young area of research, and many issues and problems have not been addressed yet. Two main results have been achieved so far: First, evolutionary techniques are able to generate astonishing circuits that are totally different from classically engineered circuits, and sometimes even superior [3]. Second, for applications with over the time varying specifications, very promising first results were achieved that indicate the potential of evolutionary techniques to construct self-adapting systems. Examples include evolved controllers for prosthetic hands and robot navigation [4].

In this paper, we argue that circuit design is a multiobjective optimization problem. Apparently, a circuit implementation reveals properties such as the required silicon area, the resulting delay (speed), and the power consumption. In classic circuit engineering, we seek to design correct functions (with regard to some specification) while either respecting constraints on area, speed and power or treating these parameters as optimization goals.

At this point, we can classify functions into two groups. The first group are functions that reveal a binary correctness property. The prime example are arithmetic functions, where we typically accept nothing less than 100% correctness. Actually, we view correctness here as a constraint. Much of the work in evolvable hardware has been concerned with design of arithmetic circuits at the gate level. While in a few cases the evolved circuits showed marginal area improvements over classic circuit designs, the input sizes were restricted to only a few bits and the design times were excessive. It seems unrealistic that larger designs can be successfully evolved that way.

The second group of functions does not have a binary correctness, just a quality metrics. A prominent example is the hashing function. The quality of a hashing function can be measured by its ability to distribute the input keys evenly. Since this depends on the input key distribution, we deal with a specification that varies over the time. Other examples include image compression, prosthetic hand control and robot navigation. These functions are ideal candidates for evolutionary design. The optimal solutions depends on the input data, which is the sole motivation for self-adaptation.

The main contributions of this paper is the combination of multiobjective optimization with evolvable hardware design. Up to now, little work has been presented that treats evolutionary hardware design as a multiobjective optimization problem. We propose TSPEA2 as a new algorithm variant that combines elements from previous multiobjective optimizers, and compare it to known approaches by means of test functions.

The paper is organized as follows: Section 2 reviews related work in multiobjective optimization of evolvable hardware. Section 3 presents the basic model used in our work, a variant of the Cartesian Genetic Programming model, along with the main genetic algorithm. In Section 4, we discuss details of the multiobjective optimizers including our new technique called TSPEA2. Experiments and results are presented in Section 5. Finally, Section 6 summarizes and concludes the paper.

2. MULTIOBJECTIVE OPTIMIZATION OF DIGITAL CIRCUITS

Many optimization problems, especially in engineering disciplines, involve several objectives. Often, the objectives are conflicting and cannot be optimized simultaneously. Then, a trade-off has to be found between the different objectives. One way to deal with trade-offs is to make a decision on the relative importance of the single objectives and merge them into a scalar objective function. A different approach is to compute a set of reasonable trade-offs first, and then make the decision which solution to use for an implementation. The definition of a reasonable trade-off can be based on the concept of *Pareto dominance*. A solution x dominates another solution y when x is superior or equal to y in all objectives, and superior in at least one objective. A non-dominated solution is denoted as a Pareto point. Evolutionary algorithms are population-based methods and are thus well-suited to approximate the set of Pareto points (Pareto front) in a single optimization run.

Research in multiobjective evolutionary algorithms has identified two key issues [5]: minimizing the distance between the approximated and the real Pareto front, and maintaining a diverse population to avoid premature convergence to a single objective. Recently, an algorithm called Strength Pareto Evolutionary Algorithm (SPEA2) was introduced [5]. SPEA uses Pareto-based ranking and stores non-dominated solutions externally. Diversity is preserved by a Pareto-based niching method. SPEA has been applied to a number of design problems [6] [7].

Closest to our work is the work of Trefzer et al. [8] who introduced MO-Turtle GA for evolving analog circuits built of operational amplifiers on a field-programmable transistor array. The authors consider a large number of objectives, including ”pull-to-rails”, dc offsets, slew rates, etc. MO-Turtle GA is a multiobjective non-dominated sorting genetic algorithm based on NSGA-II, presented in [9]. At each generation, the population of individuals consists of two parts. The first part contains the non-dominated individuals from the previous population. The second part contains mutated and recombined individuals that have been additionally selected due to a random and a main objective. By this, MO-Turtle GA tries to guide the evolution process to solutions that excel in the main objective while maintaining a high diversity of the population.

Another approach to optimize for correctness and hardware area is to use a multi-stage fitness function. Such an approach was used by Kalganova and Miller [10]. They evolve arithmetic circuits based on a Cartesian Genetic Program model (see Section 3.1). In this model, the chromosome represents a two-dimensional array of simple gates and their interconnect, which is restricted to feed-forward wires. Kalganova and Miller defined the fitness F of a chromosome as:

$$F = \begin{cases} c & \text{if } c < \ 100\%, \\ c + \gamma & \text{else} \end{cases} \quad (1)$$

In Equation 1, c denotes the percentage of the correct output bits of the circuit and γ is the number of gates in the array that are not used by an individual. As long as an individual is not correct, the selection process bases solely on the functional quality. Once an individual is correct, the required area is taken into account.

In a similar way, Coello et al. [11] evolved fully functional circuits while minimizing the number of gates. The authors proposed a multiobjective search algorithm, redefining the initial single objective in a way that treats the function of each circuit output as a separate objective. All these objectives, actually being constraints, have to be met by the evolutionary search process. In the first stage the algorithm maximizes the number of complied constraints. If all constraints are satisfied, the algorithm increases the fitness of the circuit by adding the number of ”WIRE” gates (gates that do not contribute to the logic function) to the fitness value.

3. BASIC CIRCUIT MODEL

3.1. Cartesian Genetic Programming

We use the Cartesian Genetic Programming (CGP) model in our work. Cartesian genetic programs have been intro-

Table 1. CGP model parameters

n_i	number of primary inputs
n_o	number of primary outputs
n_c	number of columns
n_r	number of rows
n_n	number of gate inputs
l	levels back parameter
n_f	number of gate functions

duced by Miller and Thomson in [12]. CGP is a structural hardware model, where a circuit is formed by combinational logic blocks arranged in a two-dimensional array and an interconnect (wires) between the blocks. The CGP model is highly parameterizable; the parameters we use in this work are summarized in Table 1.

The array consists of $n_c \times n_r$ combinational blocks, n_i primary inputs, and n_o primary outputs. The primary inputs can be connected to the inputs of any logic block in the array. A logic block in column c has n_n inputs that can be connected to the columns $c-l, \ldots, c-1$ of the array and to the primary inputs, respectively. This ensures that no combinational feedback loops are generated. A combinational block implements one out of n_f different logic functions of its inputs.

An individual is defined by its chromosome (genotype). The length of the chromosome is given by $n_c \cdot n_r(n_n+1)+n_o$. Each of the logic blocks in the array is defined by n_n+1 values, one for each input and one for the logic function. Additionally, an n_o-tuple of values selects the block outputs that are connected to the primary outputs of the array.

Figure 1 presents an example for a GCP model instance with the parameters $n_i = 4, n_o = 4, n_c = 5, n_r = 4, n_n = 2, n_f = 9$, and $l = 4$. In this example, the nine possible logic block functions have been chosen as `AND, ONE, XOR, NULL, NAND, NOT, NOR, OR,` and `XNOR`. Figure 1 illustrates a successfully evolved 2×2 bit-adder.

3.2. Reference Algorithm GA

As a reference, we have implemented a standard single-objective genetic algorithm (GA) on the GCP model. The parameters are set as follows: Top 5% of the individuals are selected and transferred without any modification to the next generation. The recombination probability is chosen to be 90%. The individuals are recombined uniformly. This means the two parents' logic blocks with according wires at position i have the same probability to become the child's logic block at the same position i. The mutation rate is a sensitive parameter with only a rather small interval inside which the evolutionary search process converges well. We choose the mutation rate such that only one combinational block or wire is mutated each time the mutation operator is applied. In our implementation each recombined child is mutated once. These parameter settings have been used for the experiments described in the following section.

3.3. Genotype-Phenotype Mapping, Fitness Evaluation

The main reason of the popularity of the CGP model is that this structural model resembles well the architectures of field-programmable reconfigurable hardware arrays. The block functions can be set from simple two-input gates, over n_n input lookup tables, to complex word-based arithmetic operators. The interconnect can model bit wires or busses. However, the routing resources of real FPGAs are much more versatile than the restricted interconnect of the CGP model. The CGP model uses such a restricted interconnect model mainly to keep the genotype (chromosome length) short and, thus, to reduce the complexity of the evolutionary operators.

Generally, the genotype has to be mapped to a corresponding phenotype for evaluating the fitness. The phenotype represents the actual circuit and is achieved from the genotype by removing all blocks of the array that do not contribute to the outputs. Note that there might still be redundancy in the phenotype. An important previous result with the CGP model [12] is that propagating redundant and currently unused structures inside the chromosomes through the search process of the evolutionary algorithm increases the speed of convergence.

In this paper, we are interested in evaluating the circuit's fitness with regard to three objectives: the functional quality, the required hardware area, and the speed of the circuit. Depending on the effort spent for the genotype-phenotype mapping, we have three variants for the fitness evaluation:

1. The functional quality of a circuit is determined by logic simulation. As we deal with combinational circuits, logic simulation actually reduces to computing the circuit's output values. The area of the circuit is estimated by counting the number of logic of blocks that are connected directly or indirectly to outputs. Routing is not taken into account. The delay of the circuit is estimated by the circuit's level, i.e., the number of logic blocks on the longest path from any primary input to any primary output. We currently use this simple genotype-phenotype mapping in our work.

2. To improve the accuracy of the area and speed estimates, FPGA backend tools (place & route) will be included in the fitness evaluation in a next step. Synthesis and technology-mapping is not necessary as the blocks of the CGP model can be mapped one-to-one to logic blocks of the target hardware.

3. The genotype will be placed, routed and configured onto the target device. This allows evaluating the func-

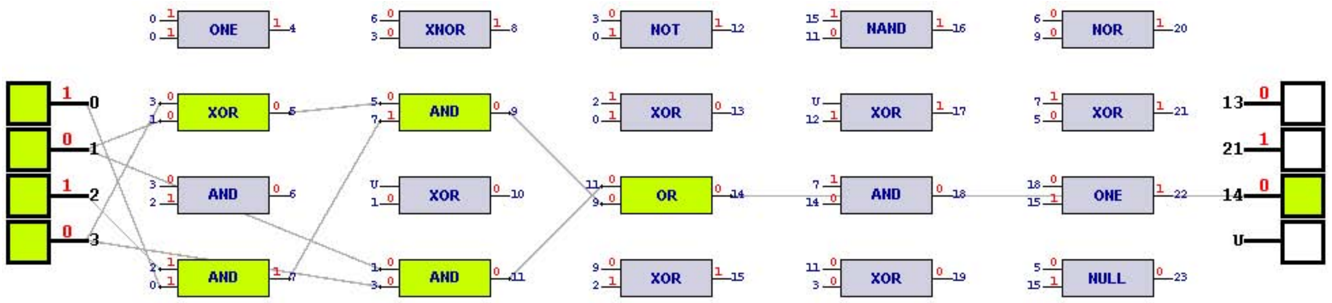

Fig. 1. 2×2 bit-adder: The data-path for the MSB is highlighted. {G, ni=4, no=4, nn=2, F, nf=9, nr=4, nc=5, l=4}, G={0 0 -1 3 1 6 3 2 8 2 0 8 6 3 9 5 7 8 x 1 6 1 3 8 3 0 5 2 0 6 11 9 14 9 2 6 15 11 7 x 12 6 7 14 8 11 3 6 6 9 1 7 5 6 18 15 -1 5 15 0 13 21 14 x}, F={AND=8, ONE=-1, XOR=6, NULL=0, NAND=7, NOT=5, NOR=1, OR=14, XNOR=9}

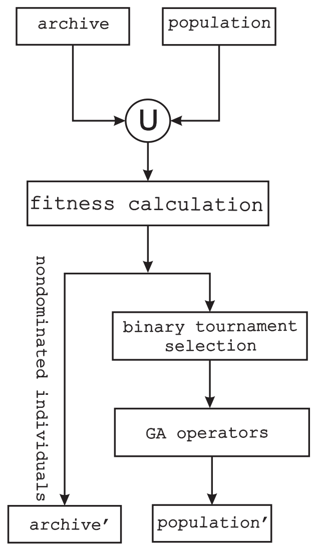

Fig. 2. Structure of SPEA2 and TSPEA2 optimizers.

tional quality by executing the circuit rather than simulating it. We intend to use this genotype-pheno-type mapping in the future to speed up the fitness evaluation and, thus, the evolutionary algorithm.

4. MULTIOBJECTIVE OPTIMIZATION ALGORITHMS

4.1. SPEA2

SPEA2 is a recent multiobjective evolutionary optimizer [13] with a structure shown in Figure 2. SPEA2 maintains two sets of individuals: an archive that contains non-dominated individuals and a breeding population. In each generation, the two sets are merged and the fitness of the individuals is evaluated. The non-dominated individuals are then copied to the new archive. If the archive exceeds a pre-defined maximal size, SPEA2 applies a nearest neighbor density estimation technique to thin out clusters on the Pareto front. The fitness assigned to an individual considers the number of individuals it dominates - the dominance count, the number of individuals that are dominators - the dominance rank, and a density estimate based on the k-th nearest neighbor method. All individuals undergo a binary tournament selection which selects parents for the recombination and mutation.

Therefore, SPEA2 is an elitism-based algorithm using non-dominated sorting and k-th nearest neighbor density estimation. The goal of SPEA2 is to approximate the Pareto front well while keeping diversity.

4.2. TSPEA2

TSPEA2 is an algorithm we have devised to put an increased selection pressure on one objective while trying to keep diversity. This should be beneficial for evolving circuits with a correctness property. Compared to SPEA2, we expect degraded fitness values for the other objectives. Both SPEA2 and TSPEA2 use an archive and a breeding population and a selection scheme based on Pareto dominance ranking. TSPEA2, however, checks as a first selection rule in a binary tournament whether one of the two individuals dominates the other regarding the main objective. TSPEA2 has been motivated by an earlier algorithm MO-Turtle GA [8], that preferred a main and several random objectives during the evolution of analog circuits.

5. EXPERIMENTS AND RESULTS

We have evolved several test functions with GA, SPEA2, and TSPEA2. In this section, we report on typical results for a 6-parity function and a hashing function. The first function is an example for a function with a correctness property. We are only interested in circuits that are 100% correct. In contrast to that, the functional quality of the hashing function is the distribution of indices achieved when keys are mapped to indices. The functional quality of the hashing function does not need to be perfect. Moreover, the perfect hashing function cannot be computed a-priori as it depends on the distribution of the input keys.

The logic blocks in our CGP model have 2 inputs (simple gates) and the functional set available comprises the 9 functions shown in Figure 1. The parameters for crossover and mutation used in SPEA2 and TSPEA2 are set as described in Section 3.2. The tournament selection operator is configured to execute two tournaments before selecting one of the competitors as a parent. For all evolutionary algorithms, we conducted 10 optimization runs with a maximum of 100.000 generations.

The delay of a circuit is in the range $\{0, \dots, n_c + 1\}$. A delay of 0 means that the longest path of the circuit connects an input directly with an output. A delay of $n_c + 1$ indicates that none of the outputs is connected to an input. The fitness with respect to speed is determined as:

$$speed(c) = 1 - \frac{delay(c)}{n_c + 1} \tag{2}$$

The speed equals 1 for the fastest possible circuit (a circuit that maps primary inputs directly to primary outputs) and 0 for a circuit that has no connection at all from primary inputs to primary outputs.

The number of logic blocks used by a circuit, denoted as $used\_blocks(c)$, is in the range $\{0, \dots, n_c \cdot n_r\}$. Based on this number the fitness with respect to area is defined as:

$$area(c) = 1 - \frac{used\_blocks(c)}{n_c \cdot n_r} \tag{3}$$

A circuit with minimal area gets an area value of 1, a circuit that utilizes all available logic blocks has an area value of 0.

5.1. 6-parity

The used parameters for the CGP model are $n_c = n_r = n_i = 6$, $l = \frac{n_c}{2}$. For the parity function, a circuit's c fitness with respect to functional quality is calculated as follows:

$$f(c) = \frac{1}{1 + \sum_{i \in \mathbb{B}^6} (\text{parity}(i) - c(i))^2}. \tag{4}$$

Thus, a correct parity function has a functional quality of 1.

It is an easy task for a conventional GA to evolve a correct circuit for the 6-parity function. Using a population of size 100, only 69 generations are needed on average to evolve a fully functional circuit. In contrast to the GA, SPEA2 with an archive and population size of 100 evolved only four correct solutions overall and needed more than 30000 generations on average. When preferring the functional quality as a main objective in TSPEA2, the search process converges faster. On average, 903 generations are needed for TSPEA2 to find a correct 6-parity circuit. Fig. 3 compares the resulting fitness development for the algorithms.

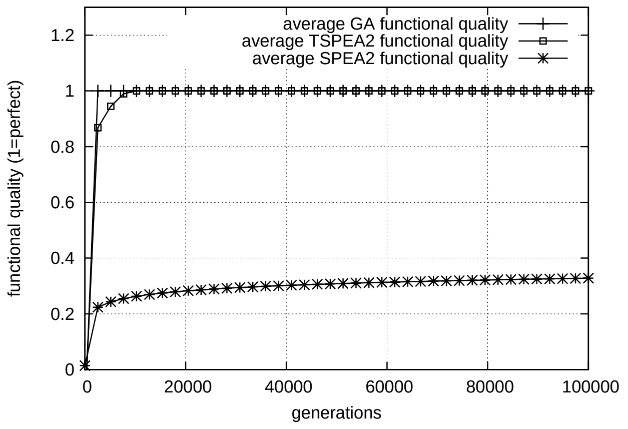

Fig. 3. Evolving the 6-parity function. Average fitness development of 10 experiments for GA, SPEA2 and TSPEA2. In each experiment SPEA2 and TSPEA2 are executed for 100.000 generations. GA is stopped after evolving a 6-parity function.

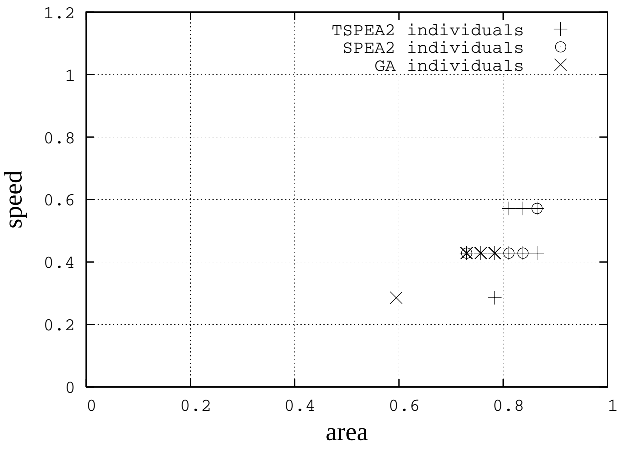

Fig. 4. Evolving the 6-parity function. All pareto-dominant individuals after 100.000 generations each 10 experiments of SPEA2 and TSPEA2 are drawn. Further the best individuals found by GA are plotted. The dominant solution is found by SPEA2 and TSPEA2. Classical GA discovers subdominant solutions.

Fig. 4 shows the speed and area parameters for all correctly evolved circuits. Both SPEA2 and TSPEA2 found the same dominant solution. Moreover, TSPEA2 managed to discover a more diverse solution set compared to SPEA2. The conventional single-objective GA evolved correct circuits with inferior area and speed.

5.2. Hashing function

We have chosen the hashing function as a test case because this function has been evolved previously by Tettamanzi et

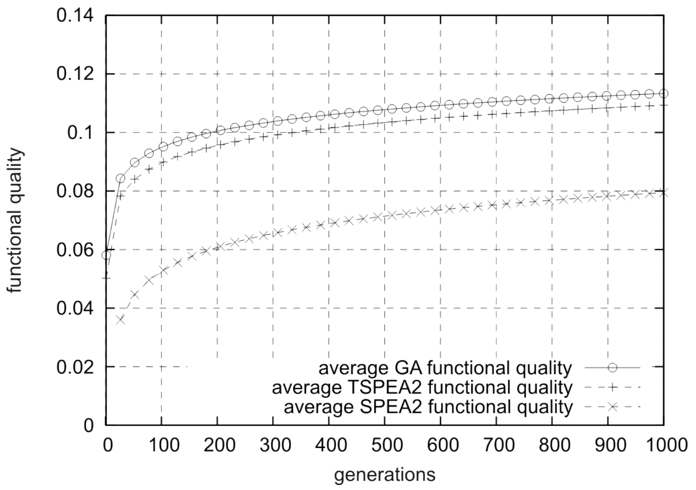

Fig. 5. Evolving the hashing function. Average fitness development of 10 experiments for GA, SPEA2, and TSPEA2.

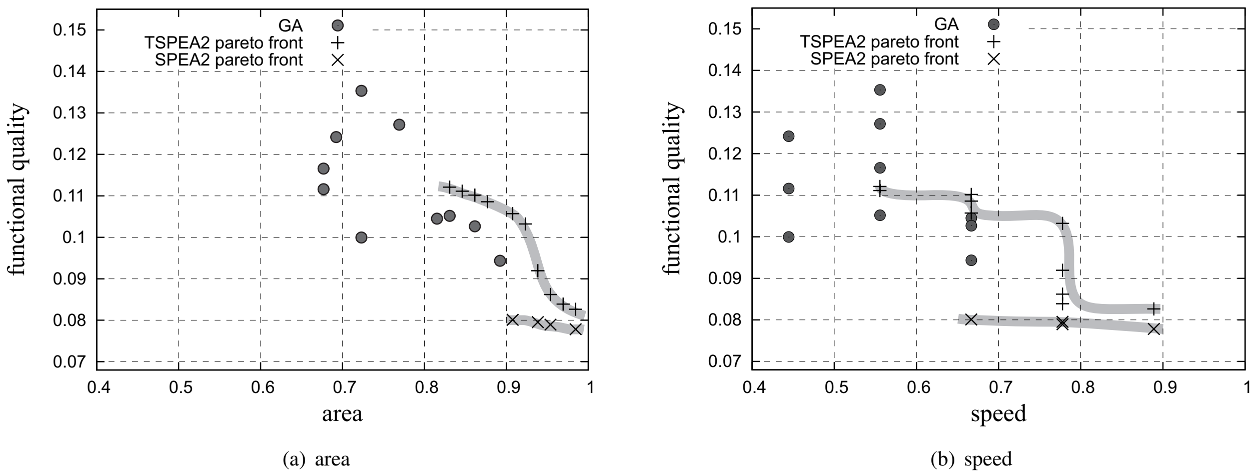

Fig. 6. Evolving the hashing function. 2D projections of the Pareto front for two typical populations. Also the objectives of the best individuals found by the GA during the 10 experiments are plotted.

Table 2. Evolving the hashing function. Reached functional qualities after 1000 generations.

	GA	SPEA2	TSPEA2
best	0.135	0.084	0.125
worst	0.094	0.075	0.092
average	0.114	0.079	0.110

al. [14]. These authors use a CGP model similar to that of Miller and Thomson [12]. The main difference is that wires can connect only to gates in the same row. We have relaxed this constraint which leads to an improvement using a conventional GA. To be able to compare the experiments, we stay consistent with CGP-parameters used in [14]: $n_c = 8$, $n_r = 8$, $l = 8$ and $n_n = 4$. The differences to the CGP-model parameters of Tettamanzi et al. in [14] are the relaxation of the constraint that wires could only connect functional blocks in the same row and in contrast to the unrestricted functional set of Tettamanzi et al. [14], a small functional set showed in Figure 1.

The problem statement formulated in [14] is as follows: Find a function $h : \mathbb{B}^{16} \rightarrow \mathbb{B}^8$ which maps a set M of 2^{12} keys to a set N of 2^8 indices in the most uniform way possible. The fitness function is defined as:

$$f(c) = \frac{1}{1 + \frac{1}{|N|}\sum_{i=1}^{|N|}(|\{j|j \in M,\ c(j) = i\}| - \frac{|M|}{|N|})^2} \tag{5}$$

Tettamanzi et al. [14] evolved the best individual with a fitness value of 0.097785 after 257 generations. On our, regarding the interconnect less constrained CGP model, the single-objective GA reaches easily an average fitness beyond 0.1, as is shown in Figure 5. After 257 generations, the best individual shows a fitness of 0.116469.

Table 2 lists the resulting functional qualities (best, worst and average case) after iterating for 1000 generations. As expected, TSPEA2 perform close to GA while SPEA2 lags behind. The slower speed of convergence of SPEA2 is also clearly illustrated by Figure 5.

Figure 6 displays functional quality vs. area and functional quality vs. speed – two-dimensional projections of the Pareto front – after evolving 1000 generations. As expected, our experiments confirmed that a conventional GA optimizes the functional quality faster than SPEA2 and TSPEA2. SPEA2 and TSPEA2 excel, however, in evolving solutions with improved area and speed.

Comparing SPEA2 with TSPEA2, we note that SPEA2 did not evolve individuals with better area or speed. In fact, all individuals found by SPEA2 are dominated by individuals generated by TSPEA2. This is an interesting observation, as one would expect that TSPEA2 which prefers the functional quality over the other objectives, leads to a somewhat deteriorated Pareto front. This result has been consistent over all simulation runs with the hashing function. A possible explanation is that in our experiments the objectives are not necessarily conflicting. Driving the evolution towards functional quality will then also improve area and/or speed. However, this may not be generalized as design experience shows that for many circuits the functional quality, speed and area are indeed conflicting.

6. SUMMARY AND FURTHER WORK

In this paper, we have presented an multiobjective evolutionary optimizers and compared the known algorithm SPEA2 with the newly devised technique TSPEA2 and a baseline GA. We have presented comparisons of these algorithms for two test functions. An intrinsic multiobjective evolutionary algorithm designs hardware circuits and maintains a set of Pareto-dominant solutions with respect to functional quality, area and speed. The system uses evolutionary adaptation to slow changes in the environment and switches to preevolved alternatives as reaction to drastic changes in the available resources.

In further work will focus on the scalability problem and investigate variants of the CGP model with more coarse-granular building blocks. Moreover, we will validate our observations on larger test functions.

7. ACKNOWLEDGEMENT

This work was supported by the German Research Foundation under project number PL 471/1-1 within the priority program *Organic Computing*.

8. REFERENCES

[1] H. de Garis, "Evolvable Hardware – Genetic Programming of a Darwin Machine," in *Proceedings International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA)*. Springer, 1993.

[2] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya, "Evolving Hardware with Genetic Learning: A First Step Towards Building a Darwin Machine," in *Proceedings 2nd International Conference on Simulation of Adaptive Behavior (SAB)*. MIT Press, 1993, pp. 417–424.

[3] A. Thompson and P. Layzell, "Analysis of Unconventional Evolved Electronics," *Communications of the ACM*, vol. 42, no. 4, pp. 71–79, 1999, ACM Press.

[4] T. Higuchi and N. Kajihara, "Evolvable Hardware Chips for Industrial Applications," *Communications of the ACM*, vol. 42, no. 4, pp. 60–66, April 1999, ACM Press.

[5] E. Ziztler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization." in *Evolutionary Methods for Design, Optimisation, and Control*, 2002, pp. 95–100.

[6] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli, "Design Space Exploration of Network Processor Architectures," in *Proceedings 1st Workshop on Network Processors at the 8th International Symposium on High-Performance Computer Architecture (HPCA)*, 2002.

[7] M. Eisenring, L. Thiele, and E. Zitzler, "Conflicting Criteria in Embedded System Design," in *IEEE Design & Test of Computers*, vol. 17, no. 2. Los Alamitos, CA, USA: IEEE Computer Society Press, April-June 2000, pp. 51–59.

[8] M. Trefzer, J. Langeheine, K. Meier, and J. Schemmel, "Operational Amplifiers: An Example for Multi-objective Optimization on an Analog Evolvable Hardware Platform." in *ICES*. Springer, 2005, pp. 86–97.

[9] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*. Paris, France: Springer, 2000, pp. 849–858.

[10] T. Kalganova and J. Miller, "Evolving More Efficient Digital Circuits by Allowing Circuit Layout Evolution and Multi-Objective Fitness," in *The First NASA/DoD Workshop on Evolvable Hardware*. Pasadena, California: IEEE Computer Society, 19-21 July 1999, pp. 54–63.

[11] C. A. Coello Coello, A. Hernández Aguirre, and B. P. Buckles, "Evolutionary Multiobjective Design of Combinational Logic Circuits," in *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware*. Los Alamitos, California: IEEE Computer Society, 2000, pp. 161–170.

[12] J. F. Miller and P. Thomson, "Cartesian Genetic Programming," in *Proceedings of the European Conference on Genetic Programming*. London, UK: Springer-Verlag, 2000, pp. 121–132.

[13] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Swiss Federal Institute of Technology, Gloriastrasse 35, CH-8092 Zurich, Switzerland, Tech. Rep. 103, 2001.

[14] E. Damiani, V. Liberali, and A. Tettamanzi, "Evolutionary Design of Hashing Function Circuits Using an FPGA." in *ICES*. London, UK: Springer-Verlag, 1998, pp. 36–46.