In Proceedings of *5th International Conference on ASIC* (ASICON 2003)
Beijing, China, October 21–24, 2003, © IEEE.

Accelerating Design Space Exploration*

Christian Haubelt and Jürgen Teich
Hardware-Software-Co-Design
University of Erlangen-Nuremberg
D-91058 Erlangen, Germany
{haubelt|teich}@cs.fau.de

Abstract: The size of search spaces in embedded system design is one of the most critical problems during design space exploration. *Pareto-Front Arithmetics* (*PFA*) has shown to be useful to overcome this problem by decomposing a hierarchical search space and just exploring each part of the system separately. Later, the exploration results are combined at higher levels of the hierarchy. In order to decrease the exploration time, this combination is performed in the objective space only. In general, this will lead to suboptimal and infeasible results. In this paper, we present new results regarding the trade-off between the quality of the results and the exploration time.

1 Introduction

Due to the large complexity of design spaces in embedded system design, heuristic techniques are mostly used to solve these in general multi-objective optimization problems. Different heuristic optimization techniques are discussed in the literature for non-hierarchical system synthesis (see [2, 3]). In this paper, we propose *Pareto-Front Arithmetics* (PFA) to deal with the increased complexity in finding optimal solutions. PFA [5] is based on a hierarchical model of embedded systems [4]. This model allows the specification of design alternatives of the application algorithm as well as alternatives of different hardware architectures. Not only that this hierarchical approach helps designers to cope with the complexity, but it also captures the knowledge of problem composition.

The idea of PFA is to start exploring the Pareto-fronts by mapping the leaves in a given hierarchical specification. Later, these Pareto-fronts are combined to generate the Pareto-front on higher hierarchical levels. This way, we reduce the exploration time, but the constructed front might not be the true Pareto-front. Nevertheless, while using only a small fraction of time, we are able to find a substantial number of Pareto-optimal solutions.

In this paper, we focus on the trade-off between the quality of the results and the time needed for exploration. The concept of PFA was already mentioned in [6] and formalized in [1]. The application of PFA on embedded system synthesis problems is shown in [5, 4].

The rest of the paper is organized as follows: The characteristics of hierarchical search spaces and hierarchical objective spaces are presented in section 2. The novel approach of Pareto-Front Arithmetics for fast design space exploration is outlined in section 3. Subsequently, we propose an algorithm based on Pareto-Front Arithmetics and uncertain objectives in order to improve the quality of the design points. Finally, section 4 proposes the application of PFA on embedded system synthesis. Furthermore, we will show by experiment that we are able to find a substantial number of Pareto-points and additional points near the true Pareto-front by our novel approach.

2 Hierarchical Optimization Problems

Before considering the problem of design space exploration in embedded systems design, we will have a closer look on the more general problem of hierarchical optimization. Starting from (non-hierarchical) multi-objective optimization problems, we introduce the notion of *hierarchical search spaces* and *hierarchical objective spaces*.

A *multi-objective optimization problem* is given by:

$$\begin{array}{l} \text{minimize } o(x), \\ \text{subject to } c(x) \leq 0 \end{array}$$

where $x = (x_1, x_2, \ldots, x_m) \in X$ is the decision vector and X is called the search space. Furthermore, the constraints $c(x) \leq 0$ determine the set of feasible solutions, where c is k-dimensional, i.e., $c(x) = (c_1(x), c_2(x), \ldots, c_k(x))$. In embedded system design, the decision vector encodes a particular implementation.

The *objective function* o is n-dimensional, i.e., we optimize n objectives simultaneously. There are k constraints

*Supported in part by the German Science Foundation (DFG), SPP 1040 "Eingebettete Systeme".

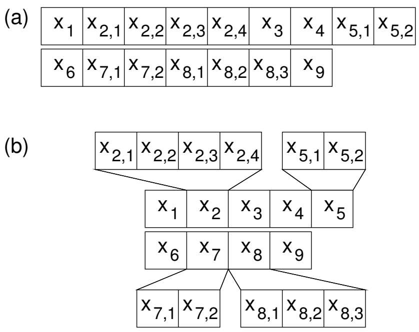

Figure 1. Example of a (a) non-hierarchical decision vector and (b) a hierarchical decision vector consisting of a non-hierarchical part $\mathrm{x}_1, \mathrm{x}_3, \mathrm{x}_4, \mathrm{x}_6, \mathrm{x}_9$ and a hierarchical part $\mathrm{x}_2, \mathrm{x}_5, \mathrm{x}_7, \mathrm{x}_8$.

c_i, $i = 1, \ldots, k$. Only those *decision vectors* $x \in X$ that satisfy all constraints c_i are in the set of feasible solutions, or for short in the *feasible set* called $X_\mathrm{f} \subseteq X$. The image of X is defined as $Y = o(X) \subset \mathbb{R}^n$, where the objective function o on the set X is given by $o(X) = \{o(x) \mid x \in X\}$. Analogously, the *objective space* is denoted by $Y_\mathrm{f} = o(X_\mathrm{f}) = \{o(x) \mid x \in X_\mathrm{f}\}$.

In general, there is not only one global optimum, but a set of so-called *Pareto-points* [7]. A Pareto-optimal solution x_p is a decision vector which is not worse than any other decision vector $x \in X$ in all objectives (without loss of generality, we assume that all objectives are to be minimized), i.e., $\nexists x \in X_\mathrm{f} : x \succ x_\mathrm{p}$, where

$$\begin{array}{rcl} x \succ \widetilde{x} \ (x \text{ dominates } \widetilde{x}) & \text{iff} & o(x) < o(\widetilde{x}) \\ x \succeq \widetilde{x} \ (x \text{ weakly dominates } \widetilde{x}) & \text{iff} & o(x) \leq o(\widetilde{x}) \\ x \sim \widetilde{x} \ (x \text{ is indifferent to } \widetilde{x}) & \text{iff} & o(x) \nleq o(\widetilde{x}) \wedge \\ & & o(x) \ngeq o(\widetilde{x}) \end{array}$$

and the relations $\circ \in \{=, \leq, <, \geq, >\}$ are defined as: $o(x) \circ o(\widetilde{x})$ iff $\forall j = 1, \ldots, n : o_j(x) \circ o_j(\widetilde{x})$.

The set of all Pareto-optimal solutions is called the *Pareto-set* X_p. An approximation of the Pareto-set X_p will be termed *quality set* X_q subsequently.

In the following, we assume that the search space has a hierarchical structure, i.e., each element x_i of a decision vector x itself may be a vector $(x_{i1}, x_{i2}, \ldots, x_{i\mathrm{k}_i})$. The decision vector x consists of a non-hierarchical and a hierarchical part, i.e., $x = \left(x^\mathrm{n}, x^\mathrm{h}\right)$. Each element $x^\mathrm{h}_j \in x^\mathrm{h}$ itself may be a decision vector consisting of a non-hierarchical and a hierarchical part. Thus, the structure is not limited to only a single level of hierarchy.

Example 1 *Figure 1 shows an example of (a) a non-hierarchical and (b) a hierarchical decision vector* x. *The non-hierarchical decision vector is a 1-dimensional vector. The hierarchical decision vector consists of a non-hierarchical part* x^n *given by the elements* $\mathrm{x}_1, \mathrm{x}_3, \mathrm{x}_4, \mathrm{x}_6$, *and* x_9 *and a hierarchical part* x^h *of the four elements* $\mathrm{x}_2, \mathrm{x}_5, \mathrm{x}_7$, *and* x_8.

By using hierarchical decision vectors, we must reconsider the objective functions, too. On the top-level, the objective function is given by: $o(x_1, x_2, \ldots, x_\mathrm{m})$. Since we do not assume monotonicity for the objective functions, we must introduce a decomposition operator $\otimes$ to construct the top-level objective function from deeper levels of hierarchy, i.e., $o(x) = o^\mathrm{n}(x^\mathrm{n}) \otimes o^\mathrm{h}(x^\mathrm{h})$ where o^n denotes the partial objective function for the non-hierarchical part of the decision vector, o^h denotes the partial objective function for the elements of the hierarchical part of the decision vector, and $o^\mathrm{h}(x^\mathrm{h}) = \bigotimes_{x_i^\mathrm{h}} o(x_i^\mathrm{h})$.

Example 2 *For the decision vector shown in Figure 1(b) we get* $o(\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_9) = o^\mathrm{n}(\mathrm{x}_1, \mathrm{x}_3, \mathrm{x}_4, \mathrm{x}_6, \mathrm{x}_9) \otimes o^\mathrm{h}(\mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4, \mathrm{x}_6, \mathrm{x}_9)$.

Abraham et al. name three advantages of hierarchical decomposition [1]. The most important advantage refers to the fact that a feasible decision vector must be composed of feasible decision vectors of the elements in the subsystem. This reduces the search space drastically. Unfortunately, we cannot assume that a Pareto-optimal top-level decision vector is composed of Pareto-optimal decision vectors of the elements in the hierarchical part.

Abraham et al. define necessary and sufficient conditions of the decomposition function of the objectives which guarantee Pareto-optimality for the top-level decision vector depending on the Pareto-optimality of elements in the hierarchical part of the decision vector [1]. Although these results are important and interesting, many optimization goals do not possess these monotonicity properties. In fact, they depend on the decomposition operator $\otimes$.

3 Pareto-Front Arithmetics

This section illustrates the novel approach of *Pareto-Front Arithmetics* (PFA). PFA is used to solve optimization problems by exploiting the hierarchical structure of the underlying optimization problems.

The inputs to Pareto-Front Arithmetics are the quality sets from mutually disjunctive parts of a decision vector x. With PFA, we construct quality sets at higher levels according the structure of the decision vector. Figure 2 shows this concept of Pareto-Front Arithmetics for the decision vector given in Figure 1. The main idea is now to do the necessary combinations in the objective space only. But how should this combination be established?

Figure 3 shows the two most important operations used during PFA. The first operation is to take the maximum of each objective of two (or more) points (Figure 3(b)). Here,

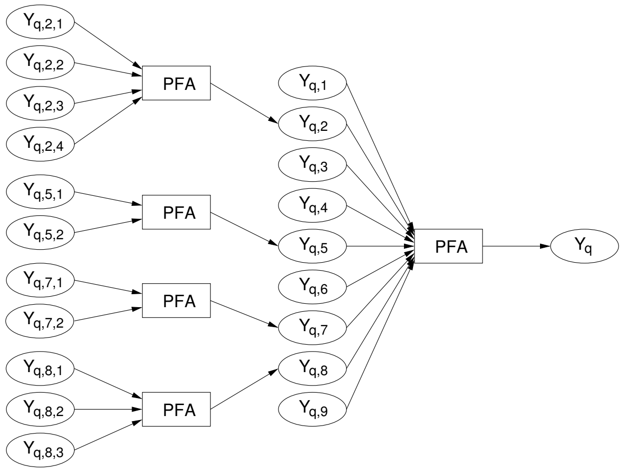

Figure 2. Concept of Pareto-Front Arithmetics illustrated by the example given in Figure 1(b). In a first step, the quality sets of the leaves are combined at a higher level of hierarchy. In a second step, these results together with other quality sets are combined at top-level.

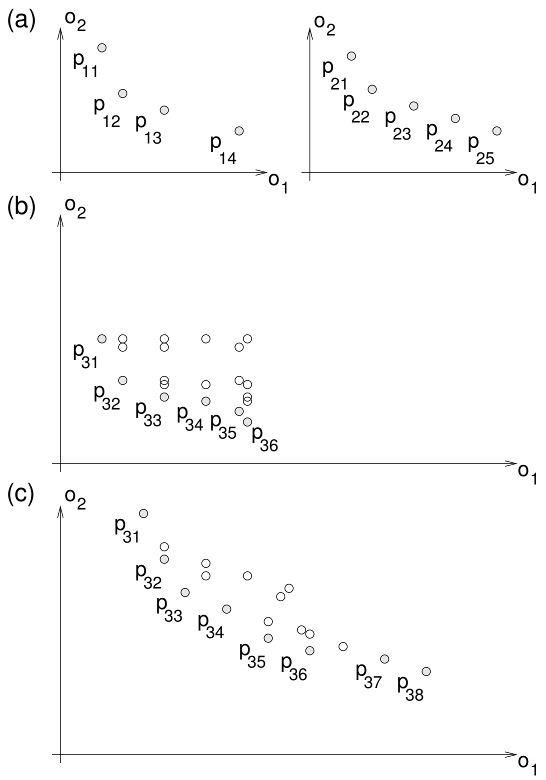

Figure 3. Example Pareto-Front Arithmetics operations. (a) Two quality sets. (b) The two quality sets are combined by using the maximum operator. (c) The two quality sets are combined using the addition.

each Pareto-optimal point p_{1i} is combined with each Pareto-optimal solution p_{2j}. The resulting objectives $(\mathrm{o}_1, \mathrm{o}_2) = (max(\mathrm{o}_1(p_{1i}), \mathrm{o}_1(p_{2j})), max(\mathrm{o}_2(p_{1i}), \mathrm{o}_2(p_{2j})))$ are filtered regarding Pareto-optimality.

Figure 3(c) outlines the addition of the objective of two or more Pareto-points: Each Pareto-optimal solution p_{1i} is combined with each point p_{2j}. Here, the resulting objectives are calculated as the sum of the objectives of the subsystems, i.e., $\mathrm{o}_k(p_{3x}) = \mathrm{o}_k(p_{1i}) + \mathrm{o}_k(p_{2j})$ for $k = 1, 2$.

More formally, PFA operations can be defined as: $\mathrm{o} = \mathrm{h}(y_1, y_2, \ldots, y_n)$, where $y_j = \mathrm{o}(x_j)\ \forall 1 \leq j \leq n$.

In embedded system design, these two operations may be used to construct a lower and an upper bound of the objectives, as will be shown later. For example, the implementation cost of a system that is composed of two subsystems can be restricted by the maximum implementation cost of each subsystem and the sum of those cost. The maximum of the cost of the two subsystems corresponds to the case where both subsystems share the same resource (e.g., IP core), while the sum of the cost model the fact that both subsystems are implemented using dedicated resources.

The objectives used during the optimization of embedded systems are non-monotonic due to resource sharing, power consumption being dependent on the binding, etc. Hence, we cannot claim Pareto-optimality for the implementations in the resulting optimality set when using PFA in general. But note: Even if we may not construct the best solutions, we are able to produce *good* implementations in less time by using PFA. This is due to the fact that we avoid the NP-complete computation of a feasible binding at higher hierarchical levels.

In order to improve the approach of Pareto-Front Arithmetics, we have to prevent the algorithm above from rejecting good points, i.e., we increase the quality of the results by still benefiting from short exploration times. This is done by considering a lower and upper bound of the objectives as described in the case of implementation cost.

Figure 4 shows the concept of PFA using so-called *uncertain objectives*. The two quality sets to be combined are given in Figure 4(a). The resulting quality set is given in Figure 4(b). The objectives $\mathrm{o}_1, \mathrm{o}_2$ are uncertain. Both objectives ($k = 1, 2$) are given by $\mathrm{o}_k(\mathrm{o}_k(\mathrm{p_{1i}}), \mathrm{o}_k(\mathrm{p_{2j}})) = [max(\mathrm{o}_k(\mathrm{p_{1i}}), \mathrm{o}_k(\mathrm{p_{2j}})), \mathrm{o}_k(\mathrm{p_{1i}}) + \mathrm{o}_k(\mathrm{p_{2j}})]$. Here, $[o_\mathrm{l}, o_\mathrm{u}]$ denotes a so-called *property interval* that is defined by its lower o_l and its upper o_u bound.

Generally, we can define an uncertain objective o by a property interval $[o_\mathrm{l}, o_\mathrm{u}]$. In this paper, we only consider discrete objectives represented by positive integers. Hence, we restrict our uncertain objective by $o \in [o_\mathrm{l}, o_\mathrm{u}] \cap \mathbb{Z}$. The lower and upper bounds are given by: $o_\mathrm{l}(x_1, x_2) = \max(o_\mathrm{l}(x_1), o_\mathrm{l}(x_2))$ and $o_\mathrm{u}(x_1, x_2) = o_\mathrm{u}(x_1) + o_\mathrm{u}(x_2)$.

Unfortunately, by using property intervals, our definition of dominance becomes meaningless. In Figure 5 five differ-

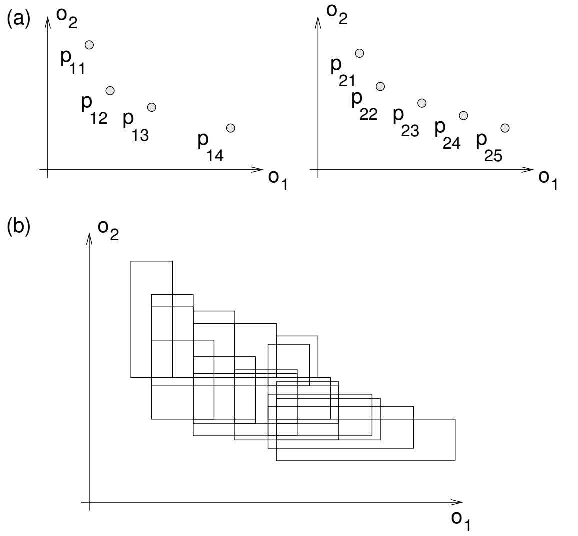

Figure 4. Example Pareto-Front Arithmetics operations using uncertain objectives. (a) shows two quality sets. (b) the quality sets are combined by using the maximum and addition operators in order to determine a lower and an upper bound for the resulting design point.

ent design points are represented by discrete property intervals. An actual design point is one of the points shown in each interval. Clearly, all (actual) points in e are dominated by any actual point in b. Thus, we say $\mathrm{b} \succ \mathrm{e}$. But we cannot assume that $\mathrm{c} \succ \mathrm{d}$ or $\mathrm{d} \succ \mathrm{c}$, since there are actual design points in c and d which are worse in o_2 or o_1, respectively.

These problems arise when two property intervals overlap. This is also shown in Figure 5. According to [8], we use the notion of *probabilistic dominance* for Pareto-optimality. Here, we consider the case of uniform distributed design points, i.e., each discrete point $y \in [o_\mathrm{l}, o_\mathrm{u}]$ in a given property interval $[o_\mathrm{l}, o_\mathrm{u}]$ is with the same probability the actual

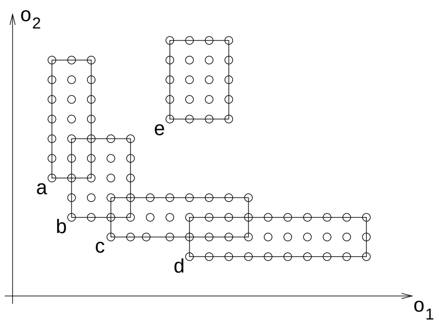

Figure 5. Dominance in case of property intervals. Here, b $\succ$ e but we do not know if c $\succ$ d or d $\succ$ c.

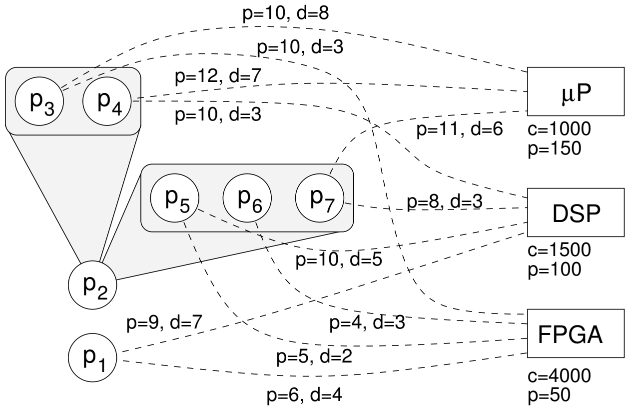

Figure 6. Specification of an embedded system. Beside the processes and the resources, the specification also shows the possible bindings by mapping edges.

design point. Furthermore, we assume that all objectives are statistically independent.

For any two design points x_1 and x_2, and m statistically independent objective functions $o_1, o_2, \ldots, o_\mathrm{m}$ the probability that x_1 dominates x_2 weakly is given by (see [8]):

$$P[x_1 \succeq x_2] = \prod_{i=1}^{m} P[o_i(x_1) \leq o_i(x_2)],$$

where $P[o_i(x_1) \leq o_i(x_2)]$ denotes the probability that the objective value $o_i(x_1)$ of design point x_1 is less or equal than the corresponding objective value $o_i(x_2)$ of design point x_2. [5] explains how to calculate this probability under the assumption of uniformly distributed design points.

In order to narrow our search space, we reject points which are dominated with a greater probability than a given, user specified probability bound p_max. With this approach, we narrow our search space but still regard a great number of *good* points. Nevertheless, if we use a probability bound p_max it still may happen that we reject a Pareto-optimal solution with a probability of $1 - p_\mathrm{max}$.

4 Results

In this section, we present our results regarding the application of PFA on the problem of embedded system design. We start from a specification of our system by using a graphical representation.

Figure 6 shows a specification of an embedded system. The behavior of the system is modeled by a so-called *process graph* g_p where processes P are represented by vertices. A process p may be refined by a set of associated process graphs. The set of available resources R of the system is given by a so-called *architecture graph* g_a. The relation between a process $p \in P$ and a resource $r \in R$ is modeled by a mapping edge $m \in M \subseteq P \times R$. A mapping edge $m = (p, r)$ indicates that process p can be implemented on resource r. Furthermore, there are some attributes associated with the mapping edges $m \in M$ and the resources $r \in R$ regarding the implementation cost c, the power consumption p, and the latency l.

Example 3 *The specification in Figure 6 consists of two processes* p_1 *and* p_2*. The process* p_2 *can be refined by any of the two associated process graphs consisting of the processes* $\mathrm{p}_3, \mathrm{p}_4$*, and* $\mathrm{p}_5, \mathrm{p}_6, \mathrm{p}_7$*. The architecture graph is given by the resources* $\mu\mathrm{P}, \mathrm{DSP}$ *and* FPGA*. Figure 6 shows 11 mapping edges.*

The goal of system synthesis is now to find an allocation α, i.e., a subset of resources, and a corresponding binding β, i.e., which process is implemented on which resource ($r \in \alpha$). After determining the allocation and binding, we are able to compute the parameters of the implementation. Before considering the task of design space exploration, we discuss the objective space used in our benchmark.

Objective Space We consider a three-dimensional objective space which is defined by the most important objectives in embedded design: the implementation cost, the overall power consumption and the latency of an implementation.

Implementation Cost: The *implementation cost* $\mathrm{c}(x)$ for a given implementation $x = (\alpha, \beta)$ is given by the sum of costs of all allocated resources. In case that only the resources $\mu\mathrm{P}$ and FPGA are allocated in the example of Figure 6, we have an implementation cost of $\mathrm{c} = 5000$.

Power Consumption: The overall *power consumption* $\mathrm{p}(x)$ of a given implementation $x = (\alpha, \beta)$ is calculated by the sum of power consumption of all allocated resources plus the additional power consumption annotated at the mapping edges. Again, we consider the example given in Figure 6. Furthermore, the allocation is given by $\alpha = \{\mu\mathrm{P}, \mathrm{FPGA}\}$ and the binding is $\beta = \{(\mathrm{p}_1, \mathrm{FPGA}), (\mathrm{p}_3, \mu\mathrm{P}), (\mathrm{p}_4, \mu\mathrm{P})\}$. Then the overall power consumption is given by $\mathrm{p}(x) = 200 + 28 = 228$.

Latency: In order to get the latency $\mathrm{l}(x)$ of an implementation, we have to compute a schedule for all operations in the system. With the allocation and binding given above we are able to execute p_1 in parallel to $\mathrm{p}_3, \mathrm{p}_4$. The processes p_3 and p_4 must be performed sequentially due to the resource sharing. The minimum latency for this implementation is given by $\mathrm{l}(x) = \max(8 + 7, 4) = 15$.

Note, that all three objectives are non-monotonous.

Benchmark Here, we present our results in using Pareto-Front Arithmetics in embedded system design. The hierarchical design space exploration starts from a specification as shown in Figure 6. In a first step, we compute the partial implementations for each leaf graph in the specification. In a next step, we combine these results at a higher level of hierarchy as described in section 3. For the example in Figure 6 the decision vector is given by $\mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2)$, $\mathrm{x}_2 = (\mathrm{x}_{2,1}, \mathrm{x}_{2,2})$, and $\mathrm{x}_{2,1} = (\mathrm{x}_3, \mathrm{x}_4)$, $\mathrm{x}_{2,2} = (\mathrm{x}_5, \mathrm{x}_6, \mathrm{x}_7)$. We start with calculating the (partial) implementations for $\mathrm{x}_{2,1}$ and $\mathrm{x}_{2,2}$ only. Next, we combine these results.

In [5] we compared Pareto-Front Arithmetics against an approach based on Evolutionary Algorithms. We have shown that PFA performs better in a sense that we find a substantial number of Pareto-optimal solutions in a very short time. But, since we are facing non-monotonic objectives, we are not able to find all Pareto-points, while the Evolutionary Algorithms have been shown to do so.

In this paper, we compare our results against the true Pareto-set (calculated by exhaustive search) by using the following quality measurements.

1. The coverage $\mathcal{C}(A, B)$ computes the relative number of points in a set B dominated by the points in the set A and is given by [10]: $\mathcal{C}(A, B) = \frac{|\{b \in B | \exists a \in A : a \succeq b\}|}{|B|}$.
2. The distance $\mathcal{D}(A, B)$ computes the average distance from a point in the set A to its nearest point in B and is given by [9]: $\mathcal{D}(A, B) = \frac{\left(\sum_{i=1}^{|A|} d_i^2\right)^{1/2}}{|A|}$ where d_i is the Euclidean distance between result i and the nearest member of the true Pareto-set B.

The results for the Pareto-Front Arithmetics, and PFA with three different probability bounds are shown in Figure 7 to Figure 9. The probability bounds are chosen to 20%, 25%, and 30%. Figure 7 shows the coverage of the Pareto-sets by the quality sets computed by the PFA approach in dependency of the problem size. As we can see, we are able to find a substantial number of Pareto-points using our novel approach. Beside the true Pareto-points, we are able to find additional points near the true Pareto-front (Figure 8 shows the distance of the quality sets computed by the PFA approach to the real Pareto-set). Figure 9 shows the number of combinations calculated by the PFA approach.

Obviously, we can improve our results (in coverage and distance) by varying the number of investigated points. This is done by using different probability bounds during the exploration. But, the most important point is that we can produce good results (for example $\mathcal{C} = 13.36\%$ and $\mathcal{D} = 0.81$ with our novel approach in a really short time (number of Combinations investigated: $< 0.78\%$). Hence, Pareto-Front Arithmetics should be used to construct initial solutions for iterative improving optimization techniques like Evolutionary Algorithms.

5 Conclusions

To handle the increasing complexity of embedded systems, we proposed Pareto-Front Arithmetics for fast design

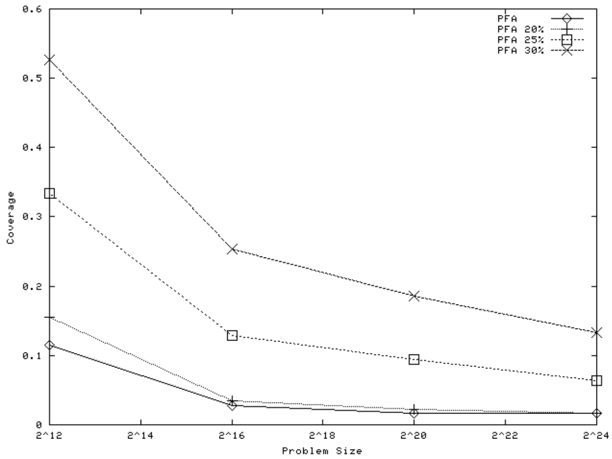

Figure 7. Coverage of the true Pareto-set in dependency of the problem size.

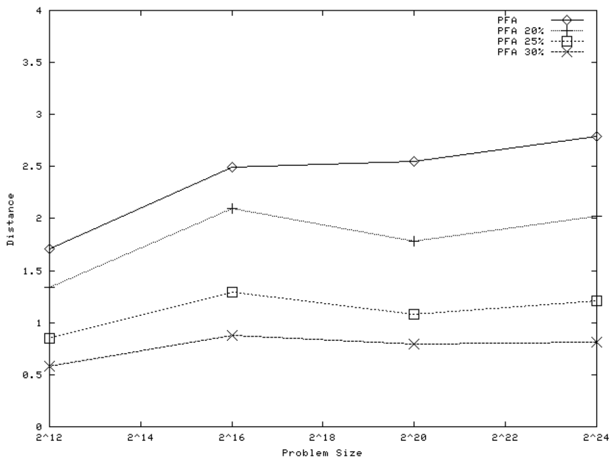

Figure 8. Distance to the true Pareto-set in dependency of the problem size.

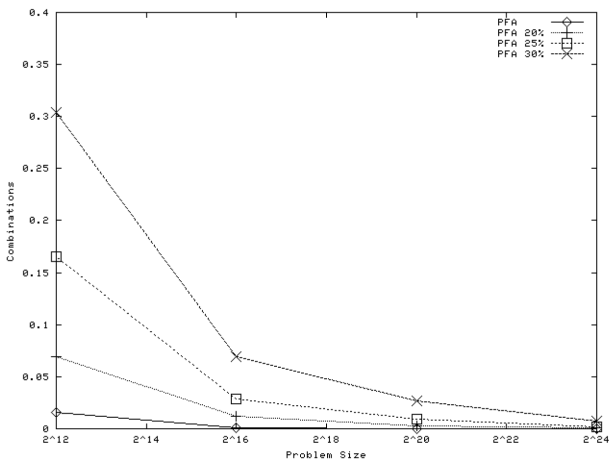

Figure 9. Combinations explored in dependency of the problem size.

space exploration using results of subsystems to approximate the set of Pareto-optimal implementations. By using this novel approach, we can reduce the exploration time dramatically. In general, this leads to suboptimal results. In this paper, we analyzed the trade-off between the quality of the results and the exploration time. As a result, we have shown that we can find a substantial number of Pareto-optimal points and a lot of additional points near the true Pareto-set. Thus, our approach seems to be good in creating initial solutions for iterative improving optimization techniques like Evolutionary Algorithms.

References

[1] S. G. Abraham, B. R. Rau, and R. Schreiber. Fast Design Space Exploration Through Validity and Quality Filtering of Subsystem Designs. Technical report, Hewlett Packard, Compiler and Architecture Research, HP Laboratories Palo Alto, July 2000.

[2] T. Blickle, J. Teich, and L. Thiele. System-Level Synthesis Using Evolutionary Algorithms. In R. Gupta, editor, *Design Automation for Embedded Systems*, 3, pages 23–62. Kluwer Academic Publishers, Boston, Jan. 1998.

[3] R. Dick and N. Jha. MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Cosynthesis of Distributed Embedded Systems. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 17(10)*, pages 920–935, 1998.

[4] C. Haubelt, S. Mostaghim, F. Slomka, J. Teich, and A. Tyagi. Hierarchical Synthesis of Embedded Systems Using Evolutionary Algorithms. In R. Drechsler and N. Drechsler, editors, *Evolutionary Algorithms for Embedded System Design*, Genetic Algorithms and Evolutionary Computation (GENA), pages 63–104, Boston, Dordrecht, London, 2003. Kluwer Academic Publishers.

[5] C. Haubelt and J. Teich. Accelerating Design Space Exploration Using Pareto-Front Arithmetics. In *Proceedings of Asia and South Pacific Design, Automation and Conference*, pages 525–531, Kitakyushu, Japan, Jan. 2003.

[6] J. R. Josephson, B. Chandrasekaran, M. Carroll, N. Iyer, B. Wasacz, G. Rizzoni, Q. Li, and D. A. Erb. An Architecture for Exploring Large Design Spaces. In *Proceedings of the National Conference of AI (AAAI-98)*, pages 143–150, Madison, Wisconsin, July 1998.

[7] V. Pareto. *Cours d'Économie Politique*, volume 1. F. Rouge & Cie., Lausanne, Switzerland, 1896.

[8] J. Teich. Pareto-Front Exploration with Uncertain Objectives. In *Proc. First International Conference on Evolutionary Multi-Criterion Optimization*, Zurich, Switzerland, Mar. 2001. In Lecture Notes in Computer Science (LNCS), Vol. 1993, pp. 314-328, Springer, 2001.

[9] D. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classification, Analyses, and New Implementations*. PhD thesis, Faculty of the Graduate School of Engineering of the Air Force Institute of Technology, Air Univesity, June 1999.

[10] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH) Zurich, Dec. 1999.