In Proceedings of *Asia and South Pacific Design Automation Conference* (ASP-DAC'03)
Kitakyushu, Japan, January 21–24, 2003, © IEEE.

Accelerating Design Space Exploration Using Pareto-Front Arithmetics[*]

Christian Haubelt and Jürgen Teich
University of Paderborn
D-33098 Paderborn, Germany
{haubelt|teich}@date.upb.de

Abstract— In this paper, we propose an approach for the synthesis of heterogeneous (embedded) systems, while exploiting a hierarchical problem structure. Particular to our approach is that we explore the set of so-called *Pareto-optimal solutions*, i.e., optimizing multiple objectives simultaneously. Since system complexity grows steadily leading to giant search spaces which demand for new strategies in design space exploration, we propose *Pareto-Front Arithmetics* (PFA) using results of subsystems to construct implementations of the top-level system. This way, we are able to reduce the exploration time dramatically. An example of an MPEG4 coder is used to show the benefit of this approach in real-life applications.

I. Introduction

Simultaneously optimizing multiple conflicting objectives like power consumption, implementation cost, etc. becomes more and more important in SoC design. Since it is possible to implement different functions of the system on different hardware components, the design space is very complex. The question which function has to be performed by which hardware component on the discussed conditions is a multi-objective optimization problem which can be formalized and is often called *system synthesis* [2]. Moreover, the basic problems are to find an *allocation* of components and to find a *binding* of functions to components while regarding data dependencies. Our main goal is to formalize these tasks in order to be able to understand the complexity of finding optimal allocations and bindings.

Due to the large complexity of the design space, heuristic optimization techniques are mostly used to solve the optimization problem. Different heuristic optimization techniques are discussed in the literature for non-hierarchical system synthesis (see [2, 5]). In this paper, we propose *Pareto-Front Arithmetics* (PFA) to deal with the increased complexity in the problem of finding optimal solutions, the so-called *Pareto-set*. PFA is based on a hierarchical model of embedded systems [10]. This model allows the specification of design alternatives of the application algorithm as well as alternatives of different hardware architectures. In contrast to other hierarchical models (see [4, 3]), this model supports the specification of resource constraints. Not only that this hierarchical approach helps designers to cope with 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 we will show that 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.

The concept of PFA was already mentioned in [7] and formalized in [1]. While [7] only shows experimental results, Abraham et al. [1] discuss a very special kind of search space which possesses certain monotonicity properties that we show do not hold in SoC design.

The rest of the paper is organized as follows: We start with an introduction to system synthesis. The characteristics of hierarchical design spaces of embedded systems are shown in Section III. The novel approach of Pareto-Front Arithmetics for fast design space exploration is outlined in Section IV. Subsequently, we propose an algorithm based on PFA and uncertain objectives in order to improve the quality of the design points. We capture the results (Section V) of these approaches for the example of an MPEG4 coder which is the running example in this paper.

II. System Synthesis

The task of system synthesis is to find the set of optimal feasible implementations for a given specification. Here, an implementation consists of two parts ([10]):

1. the *allocation* α is the set of all used hardware resources like processors, IP cores, etc. as well as the set of implemented tasks.
2. the *binding* β determines the hardware resource used for the execution of each task in the allocation.

Example 1 *We use the model of a so-called* specification graph *(see also [10]) in order to specify embedded systems. A specification graph consists of three main components:*

- *The problem graph g_{p} describing the behavior of the system modeled by a process graph.*
- *The architecture graph g_{a} models the set of possible architectures.*

[*]Supported in part by the German Science Foundation (DFG), Project SFB 614 "Selbstoptimierende Systeme des Maschinenbaus".

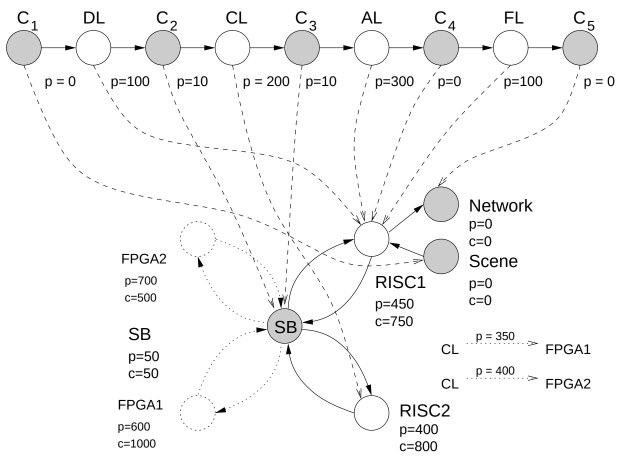

Fig. 1. Implementation of an MPEG4 coder. All vertices and edges drawn solid describe an allocation. The binding is given by the dashed edges.

- *The user-defined mapping constraints, called* mapping edges E_{m}*, relate tasks and resources in the form: "can be implemented by".*

The example introduced here and used throughout the paper is an MPEG4 coder. The problem graph of the MPEG4 coder is shown at the top of Fig. 1. We start with a given scene which is decomposed (DL) into audio/visual objects (AVO). Each AVO is coded by an appropriate coding algorithm (coding layer, CL). In a next step (Access Unit Layer, AL), the data are provided with time stamps, data type (audio, video), etc. In a last operation (Flexible Multiplexer) it is possible, e.g., to group streams with the same quality of service requirements.

The target architecture for the problem graph is shown at the bottom of Fig. 1. The architecture consists of four functional resources, two programmable RISC processors, two field programmable gate arrays (FPGAs), and a single shared bus. Additionally, the processor RISC1 is equipped with two special ports. The dashed edges between the two graphs are the additional mapping edges. For example, operation DL can be executed only on RISC1.

The mapping edges are annotated with additional power consumptions which arise when this particular mapping edges is part of the binding. Furthermore, all resources in Figure 1 are annotated with allocation cost and power consumptions. These values have to be taken into account if the corresponding resource is used in an implementation.

Consider the case that the operation CL in Figure 1 is mapped onto the resource RISC2. All other operations are non-ambiguously bound onto resources. The dashed mapping edges shown in Figure 1 indicate a feasible binding if the allocation is given by the two RISC processors, the shared bus, and the two external interfaces.

Due to data dependencies, a binding could be infeasible. A binding is called feasible if it guarantees that data communications imposed by the problem graph could by established by the allocated resources. Furthermore, a *feasible allocation* is an allocation α that allows at least one feasible binding β. In order to restrict the combinatorial search space, it is useful to determine the set of feasible allocations and bindings, With this knowledge, we define an implementation as a pair (α, β).

The Task of System Synthesis With the model introduced previously, the task of system synthesis can be formulated as an optimization problem.

Definition 1 (System Synthesis) *The task of* system synthesis *is the following multi-objective optimization problem:*

$$\begin{array}{l} \text{minimize } o(\alpha, \beta), \\ \text{subject to:} \\ \qquad \alpha \text{ is a feasible allocation,} \\ \qquad \beta \text{ is a feasible binding,} \\ \qquad c_i(\alpha, \beta) \geq 0, \ \forall i \in \{1, \ldots, q\}. \end{array}$$

The constraints on α and β define the set of valid implementations.Additionally, there are functions $c_i, i = 1, \ldots, q$, that determine the set of feasible solutions.

Normally, the *objective function* o is n-dimensional, i.e., we optimize multiple objectives simultaneously. Furthermore, there are q constraints c_i, $i = 1, \ldots, q$. All possible allocations α and bindings β span the design space X. Only these *design points* $x = (\alpha, \beta) \in X$ that represent a feasible implementation and that satisfy all constraints c_i, are in the set of feasible solutions, or in 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 $Y_{\mathrm{f}} = o(X_{\mathrm{f}})$.

Since we are dealing with multi-objective optimization problems, there is generally not only one global optimum, but a set of so-called *Pareto-points* [8].

Definition 2 (Pareto-optimality) *A feasible implementation* $i = (\alpha, \beta) \in X_{\mathrm{f}}$ *is said to be Pareto-optimal, if there is no other design point* $\widetilde{i} = (\widetilde{\alpha}, \widetilde{\beta}) \in X_{\mathrm{f}}$ *which dominates it, i.e.,* $\nexists \widetilde{i} \in X_{\mathrm{f}} : \widetilde{i} \succ i$*, where*[1]

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

The set of all Pareto-optimal solutions is called the *Pareto-optimal set*, or for short the *Pareto-set* X_{p}. An approximation of the Pareto-set X_{p} will be termed *quality set* X_{q} subsequently.

Example 2 *An example of a two-dimensional objective space is given in Figure 4(a). Assume that the objectives* o_1 *and* o_2 *are both to be minimized. There are five Pareto-optimal points,* $\mathrm{p}_{31}, \mathrm{p}_{32}, \mathrm{p}_{33}, \mathrm{p}_{34}$*, and* p_{35}*. The Pareto-set is given by* $\{\mathrm{p}_{31}, \mathrm{p}_{32}, \mathrm{p}_{33}, \mathrm{p}_{34}, \mathrm{p}_{35}\}$*. The remaining points are all dominated by at least one Pareto-optimal solution.*

[1]The relations $\circ \in \{=, \leq, <, \geq, >\}$ are defined as: $o(i) \circ o(\widetilde{i})$ iff $\forall j = 1, \ldots, n : o_j(i) \circ o_j(\widetilde{i})$

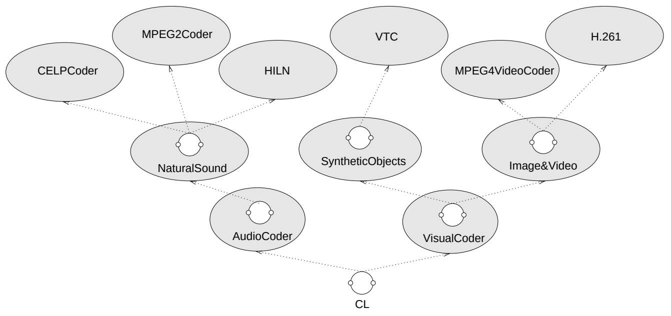

Fig. 2. Complete functional specification of the MPEG4 coding layer.

III. Hierarchical System Synthesis

By looking at the MPEG4 standard, one can see that the coding layer consists of several different coding schemes which cannot be expressed in the given specification model. In order to model these refinements, here, we introduce a hierarchical model. This *hierarchical specification graph* is based on the (non-hierarchical) specification graph described in the previous section and the concept of *hierarchical graphs* [10]. Mapping edges in the hierarchical specification graph map leaf vertices of the problem graph to leaf vertices of the architecture graph. An example of a hierarchical specification graph is shown in Figure 3.

Example 3 *Figure 2 shows possible refinements of the coding layer. There are two possible codings: audio and visual coding, i.e.,* $\mathrm{CL}.G = \{\mathrm{AudioCoder}, \mathrm{VideoCoder}\}$. *The audio coder subgraph consists of only one vertex* $\mathrm{AudioCoder}.V = \{\mathrm{v_{ac}}\}$ *and no edges* $\mathrm{AudioCoder}.E = \{\}$. *In the next level of the hierarchy, we can refine* $\mathrm{v_{ac}}$ *by a single subgraph* $\mathrm{NaturalSound}$. *Due to space limitations, we omit the details of the different coding schemes. The six leaf graphs can be refined with the non-hierarchical model.*

In the following we use the notation (g_{s}, g) to denote a *partial specification* where g is any subgraph in the problem graph of the given specification g_{s}. (g_{s}, g) is obtained by removing all vertices and subgraphs from the problem graph leaving only g and all its associated subgraphs. To guarantee a meaningful specification, all unconnected mapping edges are removed from the specification graph, too. Furthermore, vertices in the architecture graph which are not incident to any mapping edge, will also be deleted.

Obviously, the partial specification $(g_{\mathrm{s}}, g_{\mathrm{p}})$ corresponds to the original specification g_{s} with g_{p} being the original problem graph. In the following, we denote this particular partial specification as *top-level specification*.

With the notion of a partial specification, we can decompose our system: A *decomposition* $\Theta(g_{\mathrm{s}}, g_{\mathrm{p}})$ of a top-level specification g_{s} is a partition of g_{s} into disjunctive partial specifications.

Example 4 *The decomposition of our MPEG4 coder* $\mathrm{g_s}$ *given in Figure 2 into its leaf specifications is given by (Here, we use* $\otimes$ *as the decomposition operator, see also [10]):*

$$\begin{aligned}\Theta(\mathrm{g_s}, \mathrm{g_p}) \quad = \quad & (\mathrm{g_s}, \mathrm{CELP}) \otimes (\mathrm{g_s}, \mathrm{MPEG2}) \otimes (\mathrm{g_s}, \mathrm{HILN}) \otimes \\ & (\mathrm{g_s}, \mathrm{VTC}) \otimes (\mathrm{g_s}, \mathrm{MPEG4}) \otimes (\mathrm{g_s}, \mathrm{H.261})\end{aligned}$$

With the definitions given above, we use the notation $X(g_{\mathrm{s}}, g)$ to denote the partial design space regarding the partial specification (g_{s}, g). Furthermore, let $X_{\mathrm{p}}(g_{\mathrm{s}}, g_{\mathrm{p}})$ and $X_{\mathrm{f}}(g_{\mathrm{s}}, g_{\mathrm{p}})$ denote the Pareto-set and the feasible set of specification graph g_{s} while meeting the imposed constraints $c = (c_1, c_2, \ldots, c_{\mathrm{q}})$, respectively. $Y_{\mathrm{f}}(g_{\mathrm{s}}, g_{\mathrm{p}})$ is the objective space regarding $(g_{\mathrm{s}}, g_{\mathrm{p}})$.

Hierarchical Design Space A typical design space for an embedded system exploration consists very often of many infeasible solutions. The probability of finding a feasible implementation „ad hoc" is nearly zero. Here, we present a fast approach which relies strongly on the hierarchical decomposition of a system.

We consider the case of composing a top-level design by solutions of its subsystems. The exploration of the solutions of the subsystems and the composition of these solutions is called *hierarchical design space exploration*. The three main advantages for using hierarchical design space exploration are:

1. The size of each subsystem's design space is smaller than the top-level design space.
2. The evaluation effort for each subsystem design is low because of the smaller complexity of the subsystem.
3. The number of evaluated top-level design points is a small fraction of the original search space. This is due to the fact that a valid implementation is only composable of valid solutions of its subsystems.

Abraham et al. define necessary and sufficient conditions of the composition function of the objectives which guarantee Pareto-optimality for the top-level system depending on the Pareto-optimality of its subsystems [1]. For our problem we can formulate these conditions as: A given decomposition $\Theta(g_{\mathrm{s}}, g_{\mathrm{p}})$ of a top-level specification $(g_{\mathrm{s}}, g_{\mathrm{p}})$ is called (weak or strong) *monotonic* if the top-level Pareto-set $X_{\mathrm{p}}(g_{\mathrm{s}}, g_{\mathrm{p}})$ is given by the composition of the Pareto-sets $X_{\mathrm{p}}(g_{\mathrm{s}}, g_i)$ for all subsystems $(g_{\mathrm{s}}, g_i) \in \Theta(g_{\mathrm{s}}, g_{\mathrm{p}})$. This is true if the composition function of each objective is a monotonic function.

Although this result is important and interesting, the optimization goals in SoC design unfortunately do not possess these monotonicity properties as will be shown later. Hence, we cannot assume that a Pareto-optimal top-level design is composed of Pareto-optimal subsystem implementations.

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

Implementation Cost: The *implementation cost* $\mathrm{cost}(i)$ for a given implementation $i = (\alpha, \beta)$ is given by the sum of costs of all allocated resources.

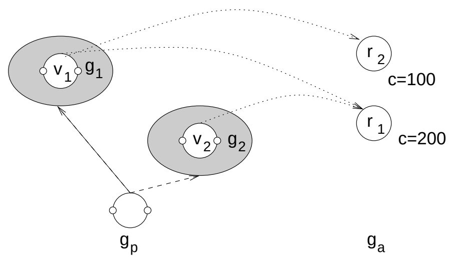

Fig. 3. Sample Specification.

Example 5 *Again, we look at Example 1. In order to calculate the implementation cost* $\mathrm{cost}(i)$*, we have to compute the set of allocated resources. With this set we calculate the implementation cost of this implementation* i*:*

$$\begin{aligned}\mathrm{cost}(i) &= \mathrm{cost(RISC1)} + \mathrm{cost(RISC2)} + \mathrm{cost(SB)} + \\ &\quad \mathrm{cost(Network)} + \mathrm{cost(Scene)} \\ &= 750 + 800 + 50 + 0 + 0 = 1600\end{aligned}$$

Now, we can show that the composition function of the implementation cost is indeed a non-monotonic function.

Theorem 1 *The composition function for the objective cost of an embedded system is non-monotonic.*

Proof 1 *We prove this theorem by contradiction: Given the specification graph* g_s *depicted in Figure 3. Let the allocation cost for* r_1 *and* r_2 *be 200 and 100, respectively. The cost objective for the Pareto-optimal design regarding the cost of allocated resources of the subsystems are given by* $\mathrm{cost}(X_\mathrm{p}(g_\mathrm{s}, g_1)) = \{(100)\}$ *and* $\mathrm{cost}(X_\mathrm{p}(g_\mathrm{s}, g_2)) = \{(200)\}$*. When considering subsystem* (g_s, g_1) *alone, we would allocate resource* r_1 *as a cost-minimal implementation. When considering subsystem* (g_s, g_2) *alone, the resource* r_2 *would determine its cost-optimal implementation. By combining both implementations, we obtain an implementation of the top-level design* $(g_\mathrm{s}, g_\mathrm{p})$*. Due to the allocation of both resources, we get implementation cost of* $\mathrm{cost}(X_\mathrm{p}(g_\mathrm{s}, g_1)) \otimes \mathrm{cost}(X_\mathrm{p}(g_\mathrm{s}, g_2)) = 300$ *for the combined implementation which is obviously suboptimal as* $\mathrm{cost}(X_\mathrm{p}(g_\mathrm{s}, g_\mathrm{p})) = 200$*.* ■

Power Consumption: The overall *power consumption* $\mathrm{pow}(i)$ of a given implementation $i = (\alpha, \beta)$ is approximated by the sum of power consumption of all allocated resources plus the additional power consumption annotated at the mapping edges.

Example 6 *The implementation described in Example 1 possesses the following overall power consumption:*

$$\begin{aligned}\mathrm{pow}(i) &= \mathrm{pow(RISC1)} + \mathrm{pow(RISC2)} + \mathrm{pow(SB)} + \\ &\quad \mathrm{pow(Network)} + \ldots + \mathrm{pow}((\mathrm{C}_5, \mathrm{Network})) \\ &= 450 + 400 + 50 + 0 + 0 + 0 + 100 + 10 + \\ &\quad 200 + 10 + 300 + 0 + 100 + 0 = 1520\end{aligned}$$

The third objective is the reciprocal of the flexibility of an implementation [10]:

Flexibility: Without loss of generality, we only treat systems throughout this paper where the flexibility $f(g_\mathrm{p})$ equals the number of used leaf graphs of the problem graph g_p. For the problem graph in Figure 2 we obtain a maximum flexibility of $g(\mathrm{CL}) = 6$ by using the CELP, MPEG2, HILN, VTC, MPEG4, and H.261 coder in our implementation. For a comprehensive introduction to the flexibility of an embedded system see [6].

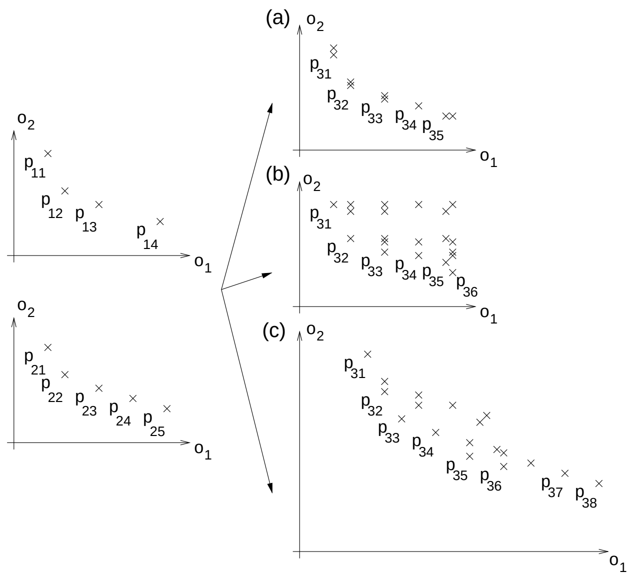

Fig. 4. Example Pareto-Front Arithmetics operations (a) union, (b) maximum, and (c) addition of objectives of Pareto-points.

IV. Hierarchical Design Space Exploration

This section proposes a novel approach, namely *Pareto-Front Arithmetics* (PFA) in design space exploration exploiting the hierarchical structure of the underlying specifications.

A. Pareto-Front Arithmetics

The inputs to Pareto-Front Arithmetics are the quality sets from the mutually disjunctive partial specifications $(g_\mathrm{s}, g_1), (g_\mathrm{s}, g_2), \ldots, (g_\mathrm{s}, g_n)$. Two partial specifications $(g_\mathrm{s}, g_1), (g_\mathrm{s}, g_2)$ are mutually disjunctive iff $g_1 \cap g_2 = \emptyset$. Here, we consider each partial specification as a non-hierarchical specification associated with a leaf graph of the problem graph. In order to optimize such a partial specification, we use evolutionary algorithms (EA) as described in [2]. The resulting quality sets are then used by the PFA to construct a quality set for the top-level specification $(g_\mathrm{s}, g_\mathrm{s}.g_\mathrm{p})$.

Figure 4 shows three possible operations used by PFA. The first operation (Figure 4(a)) is the union of two or more Pareto-fronts, i.e., each Pareto-optimal solution is added to the resulting set. All points not dominated in the resulting set are Pareto-optimal. The second operation is to take the maximum of each objective of two (or more) points (Figure 4(b)). Here, 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..

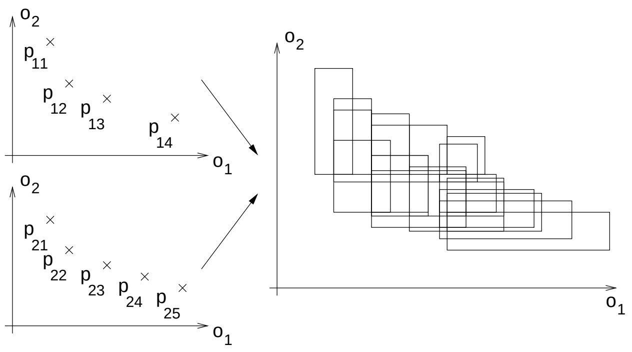

Fig. 5. Example Pareto-Front Arithmetics operations using uncertain objectives.

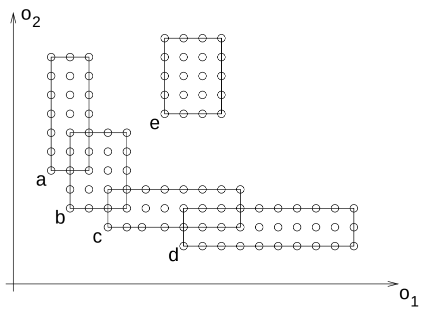

Fig. 6. Dominance in Case of Property Intervals. Here, $b \succ e$ but we do not know if $c \succ d$ or $d \succ c$.

Figure 4(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:

$$\mathrm{o} = \mathrm{h}(y_1, y_2, \ldots, y_n), \text{ where } y_j = \mathrm{o}(x_j)\ \forall 1 \leq j \leq n$$

The objectives used in optimization of embedded systems are more complex due to resource sharing, power consumption being dependent on the binding, etc. Consequently, most of these operations are non-monotonic. Hence, we cannot claim Pareto-optimality for the implementations in the resulting optimality set when using PFA in general. But note: Even if we do 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.

Section V shows a case study using PFA.

B. PFA with Uncertain Objectives

To get a better approximation of the Pareto-front, we have to prevent the algorithm described above from rejecting good points. We propose an improvement of the PFA as described above in a sense that the quality of the results increases by still benefiting from short exploration times. This is done by considering a lower and upper bound of the objectives, e.g., the implementation cost of a system that is composed of two subsystems can be restricted by the maximum implementation cost of the subsystems and the sum of those cost. The maximum of the cost of the two subsystems corresponds to the case were 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.

Figure 5 shows the concept of PFA using so-called *uncertain objectives*. 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 $o \in [o_\mathrm{l}, o_\mathrm{u}] \cap \mathbb{Z}$. We restrict ourselves to the case that the lower and upper bounds are defined as follows:

$$\mathrm{o_l}((g_\mathrm{s}, g_i), (g_\mathrm{s}, g_j)) = \max(\mathrm{o_l}((g_\mathrm{s}, g_i)), \mathrm{o_l}((g_\mathrm{s}, g_j)))$$
$$\mathrm{o_u}((g_\mathrm{s}, g_i), (g_\mathrm{s}, g_j)) = \mathrm{o_u}((g_\mathrm{s}, g_i)) + \mathrm{o_u}((g_\mathrm{s}, g_j))$$

Unfortunately, by using property intervals, our definitions for domination become meaningless (Figure 6). In Figure 6 five different 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 $\mathrm{b} \succ \mathrm{e}$. Unfortunately, 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 6. According to [9], 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 $p \in P$ in a given property interval P is with the same probability the actual design point. Furthermore, we assume that all objectives are statistically independent.

For any two design points a and b, and m statistically independent objective functions $o_1, o_2, \ldots, o_\mathrm{m}$ the probability that a dominates b weakly is given by (see [9]):

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

where $P[o_i(a) \leq o_i(b)]$ denotes the probability that the objective value $o_i(a)$ of design point a is less or equal than the corresponding objective value $o_i(b)$ of design point b. Here, we assume a uniform distribution of the objectives within the property intervals. Hence, the probability $P[o_i(a) \leq o_i(b)]$ is given as:

$$\begin{array}{ll} 0 & \text{if} \quad o_{i\mathrm{u}}(b) < o_{i\mathrm{l}}(a) \\ 1 & \text{if} \quad o_{i\mathrm{u}}(a) < o_{i\mathrm{l}}(b) \\ \sum_{j \in [o_{i_\mathrm{l}}(a), o_{i\mathrm{u}}(a)]} \frac{P[j \leq o_i(b)]}{o_{i\mathrm{u}}(a) - o_{i\mathrm{l}}(a) + 1} & \text{else} \end{array}$$

with $P[j \leq o_i(b)] = \sum_{o_i(b) \geq j} \frac{1}{o_{i\mathrm{u}}(b) - o_{i\mathrm{l}}(b) + 1}$. The first two cases are obvious and correspond to the case when both property intervals do not overlap. The third case, is simply the probability that $o_i(a)$ has a certain value multiplied with the probability that $o_i(b)$ takes a greater or equal value.

Example 7 *Considering the property intervals given in Figure 6, we get probabilistic dominance values like:*

$$\begin{array}{lcl} P[a \succeq b] & = & \frac{1}{3} \cdot \left(2 \cdot \frac{4}{4} + \frac{3}{4}\right) \cdot \frac{1}{7} \cdot \left(\frac{3}{5} + \frac{2}{5} + \frac{1}{5}\right) \approx 15.71\% \\ P[b \succeq a] & = & \frac{1}{4} \cdot \left(\frac{2}{3} + \frac{1}{3}\right) \cdot \frac{1}{5} \cdot \left(3 \cdot \frac{7}{7} + \frac{6}{7} + \frac{5}{7}\right) \approx 22.86\% \end{array}$$

In order to narrow our search space, we reject points which are dominated with a greater probability than a given, user specified probability bound $p_{\max}$.

Example 8 *With Example 7 and a probability bound of* $p_{\max} = 20\%$*, we reject the point* a *since it is dominated with a probability of* 22.86%.

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 $1 - p_{\max}$.

V. Case Study

This section presents first results obtained by using our new techniques in hierarchical design space exploration presented in this paper. Figure 2 shows the complete functional specification for our MPEG4 coder. There are six leaf graphs, each representing a different coding algorithm. Our goal is to implement at least one of these algorithms with the objective to minimize both cost and power.

As described in section II, we also need the architecture on which we can execute the tasks given by the problem graph. Here we use the same architecture template for all subgraphs. Due to space limitations, we omitted the detailed specification of the leaf graphs, the architecture graph as well as the mapping edges. For a full description of the case study see [10]. The search space for the example used in this paper consists of more than 2^{200} points.

As due to complexity reasons, we do not know the true Pareto-set, we compare the quality sets obtained by each approach against the quality set obtained by combining all these results and taking the Pareto-set of this union of optimal points. This set possesses 38 Pareto-optimal design points. A good measure of comparing two quality sets A and B is then to compute the so-called *coverage* $\mathcal{C}(A,B) = \frac{|\{b \in B | \exists a \in A : a \succeq b\}|}{|B|}$. Obviously, a coverage of $\mathcal{C}(A,B) = 1$ corresponds to the fact that all elements in B are weakly dominated by at least one element of A. On the other hand, a coverage of $\mathcal{C}(A,B) = 0$ means that none of the elements in B is weakly dominated by the elements of A.

When using PFA to construct a quality set, we have to

1. Generate leaf graph Pareto-fronts for each leaf graph in Figure 2 using one non-hierarchical evolutionary algorithm like [2] each time.

2. Apply Pareto-Front Arithmetics to these fronts.

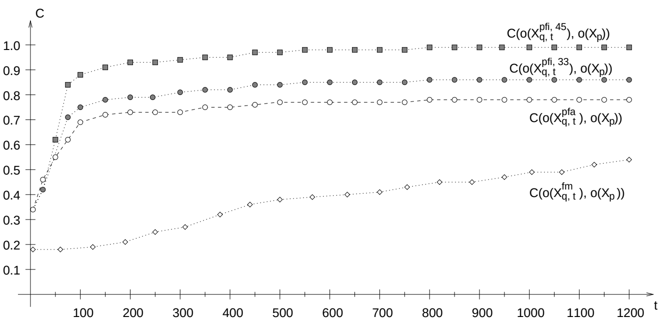

Fig. 7. Coverage of the Pareto-optimal implementations found after a given number of generations compared to the Pareto-set. While PFA produces better results as a non-hierarchical approach, we see that Pareto-Front Arithmetics with uncertain objectives (PFI 33% and 45%) outperforms the original PFA if a reasonable probability bound ($p_{\max} \geq$ 33%) is chosen. Furthermore, by setting the probablility bound to $p_{\max}$ = 45%, we reach full coverage of the Pareto-set.

Using the multi-objective evolutionary algorithm SPEA2 (see [11]) (with a crossover probability of $p_c = 0.25$, a mutation probability of $p_m = 0.2$, an archive size of $|\overline{P}_t| = 70$, and a population size $|P| = 300$) the Pareto-set for the partial specification of the leaf graphs of the problem graph of Figure 2 were constructed. The calculation time τ and number of generations t are given below:

	CELP	AAC	HILN	VTC	Image	H.261
τ	2 s	19.6 s	600.4 s	2 s	450.4 s	116.6 s
t	1	3	99.4	1	62.4	18.4

A. Pareto-Front Arithmetics

The times given above show that the exploration of the design space of small leaf graphs may already be very time consuming. Thus, we cannot expect to explore the full flat design space within a reasonable amount of computation time. As described in Section IV, we propose Pareto-Front Arithmetics for fast design space exploration as follows:

With the results for each leaf graph, we can start a quick construction ($< 1s$ for the given example) of our quality set $X_{q,t}^{\text{pfa}}$ after each generation t. Since this computation time is an order of magnitude smaller than the time needed for exploring the top-level design space (as experiments have shown), this approach seems to be a fast method of approximating the Pareto-set X_{p}.

For our example, we obtained a coverage of $\approx 78\%$ of the Pareto-set (see Figure 7). This is also the maximum coverage we can expect, since this quality set was constructed from the Pareto-sets of the leaf graphs. Furthermore, using Pareto-Front Arithmetics, our results converged fast ($<$ 350 generations).

B. Pareto-Front Arithmetics with Uncertain Objectives

Even better results are obtained by using Pareto-Front Arithmetics with uncertain objectives. Here, we assume a discrete uniform distribution of all design points in the property intervals. Figure 7 shows that the Pareto-Front Arithmetics using uncertain objectives produces better results if points dominated at least by 33% are rejected. The Pareto-Front Arithmetics with uncertain objectives yields slightly better results ($\approx$ 85%) than the original Pareto-Front Arithmetics. Nevertheless, by using uncertain objectives the computation times after each generation increases by a factor of approximately 10.

On the other hand, if we use a threshold of 20% our results are really poor ($<$ 20%, not shown in Figure 7). But also the computation time goes down to 0.2 s. Finally, with a threshold of 45% we compute the true Pareto-set ($\mathcal{C}(o(X^{\mathrm{pfi},45\%}_{\mathrm{q},1200}), o(X_{\mathrm{p}})) = 100\%$). Since the number of investigated points increases disproportionately with the threshold, we compare a remarkable fraction of the possible combinations of the subsystems. This fact is reflected in the increased computation time ($\approx$ 10 min.) after each generation.

C. Non-Hierarchical EAs

In a last step, we compare our two new approaches against a non-hierarchical approach. In this non-hierarchical exploration algorithm, we explore the design spaces individually for all $2^k - 1$ possible combinations where k is the total number of leaf subgraphs in the problem graph. Therefore, we perform six different exploration runs for each individual leaf subgraph, $\binom{6}{2}$ = 15 runs for combinations that select exactly two leaf subgraphs, etc. All in all, there are $2^6 - 1 = 63$ combinations of leaf subgraphs, where at least one leaf subgraph is chosen.[2]

For each of these 63 cases we apply the EA for a certain number of generations for each combination k to obtain the quality set of the different leaf graph selections. Since we use the same number of generations for each combination, we simulate the case were each combination is selected with the same probability. With the given archives $\overline{P}_{t,k}$, we are able to construct the quality set of the top-level design, denoted by $X^{\mathrm{fm}}_{\mathrm{q},t}$, by simply taking the union of all archives $\overline{P}_{t,k}$ of the combinations and calculating the Pareto-optimal points in the union. Figure 7 shows the result compared with the Pareto-set of our particular problem. The exploration time by using this non-hierarchical EA is nearly 1 day. Thus, we waste most of the computation time by trying to improve suboptimal implementations.

For our particular problem, we see that both Pareto-Front Arithmetics with or without uncertain objectives are superior to the non-hierarchical exploration. By using the flattened model, the coverage of the Pareto-front is $\mathcal{C}(o(X^{\mathrm{fm}}_{\mathrm{q},1200}), o(X_{\mathrm{p}})) \approx 52\%$. However, as in the case of Pareto-Front Arithmetics with uncertain objectives, it should be possible to find all Pareto-optimal solutions by using this non-hierarchical EA.

VI. Conclusions

To handle the increasing complexity of embedded systems, we proposed Pareto-Front Arithmetics for fast design space exploration using results of subsystems to approximate the set of Pareto-optimal implementations of the top-level system. This approach has proven to find a substantial number of Pareto-optimal and additional feasible implementations while reducing the exploration time dramatically. A second approach based on PFA and uncertain objectives improves the quality of solutions once more. The penalty for this improvement lies in the slightly increased exploration time.

[2]Note that this method is in general not a feasible way to go as the number of EA runs grows exponentially with the number of leaf graphs.

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] K. S. Chatha and R. Vemuri. MAGELLAN: Multiway Hardware-Software Partitioning and Scheduling for Latency Minimization of Hierarchical Control-Dataflow Task Graphs. In *Proc. CODES'01, Ninth International Symposium on Hardware/Software Codesign*, Copenhagen, Denmark, Apr. 2001.

[4] L. A. Cortés, P. Eles, and Z. Peng. Hierarchical Modeling and Verification of Embedded Systems. In *Proc. Euromicro Symposium on Digital Systems Design*, Warsaw, Poland, Sept. 2001.

[5] 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.

[6] C. Haubelt, J. Teich, K. Richter, and R. Ernst. System Design for Flexibility. In C. D. Kloos and J. da Franca, editors, *Proceedings of Design, Automation and Test in Europe*, pages 854–861, Paris, France, Mar. 2002. IEEE Computer Society.

[7] 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.

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

[9] 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.

[10] J. Teich, C. Haubelt, S. Mostaghim, F. Slomka, and A. Tyagi. Techniques for Hierarchical Design Space Exploration and their Application on System Synthesis. Technical Report 1/2002, Institute Date, Department of EE and IT, University of Paderborn, Paderborn, Germany, 2002.

[11] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, Swiss Federal Institute of Technology (ETH) Zurich, 2001. TIK-Report 103. Department of Electrical Engineering.