On Search-Space Restriction for Design Space Exploration of Multi-/Many-Core Systems

Valentina Richthammer and Michael Glaß
Ulm University
Germany
{valentina.richthammer,michael.glass}@uni-ulm.de

Abstract.

Design Space Exploration (DSE) for embedded system design with its multi-objective nature and large search spaces typically prohibits exhaustive search and popularized the use of metaheuristic optimization techniques. Recent large-scale multi- and especially many-core architectures offering a multitude of application mapping possibilities create tremendously large search spaces which give rise to the question whether established metaheuristics are still efficient. In this work, we propose to employ a heuristic search-space restriction (SSR) approach based on the exploration of subsystems, which significantly reduces individual search-space size and, thus, exploration time. Knowing that this kind of restriction may miss global optima, we also investigate the use of high-quality solutions derived from subsystems as an *initial population* for the optimization of the complete system. Experimental results for tiled 8×8 to 24×24 many-core architectures and several benchmark applications show that the proposed SSR enables the metaheuristic to derive implementations of higher quality in a significantly reduced exploration time. Although not all global optima are exposed to the restricted problem, this work gives evidence that too complex search spaces may sacrifice the efficiency of metaheuristics drastically and, thus, serves as a motivation for future research into SSR techniques for DSE.

1. Introduction

The growing complexity of modern embedded applications combined with the increasing utilization of heterogeneous multi- and many-core architectures results in a multitude of application mapping possibilities that exhibit great variance in execution properties like energy consumption, latency, etc. Therefore, Design Space Exploration (DSE) techniques are typically employed to approximate a Pareto-optimal front of *operating points* for run-time embedding, optimized for often conflicting design objectives such as resource or energy requirements. Design- as well as run-time application mapping consists of the following steps: (a) determining an *allocation* of resources from the architecture, (b) *binding* each task of the application to a resource in the allocation so that communication constraints imposed by data dependencies between the tasks can be met, and (c) determining a feasible *schedule* for the execution of tasks. This in itself is an NP-complete problem (commonly known as *system synthesis*) [1] that has been the subject of a large body of research (for an overview see, e.g., [17]). During DSE, a multitude of feasible mappings needs to be constructed and evaluated with respect to the design objectives, constituting a multi-objective optimization problem (MOP). Evaluation is either done using simulation—which may require a considerable amount of execution time—or analytically, often trading off evaluation accuracy for speed. Either

This work was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" under Grant SFB/TR 89.

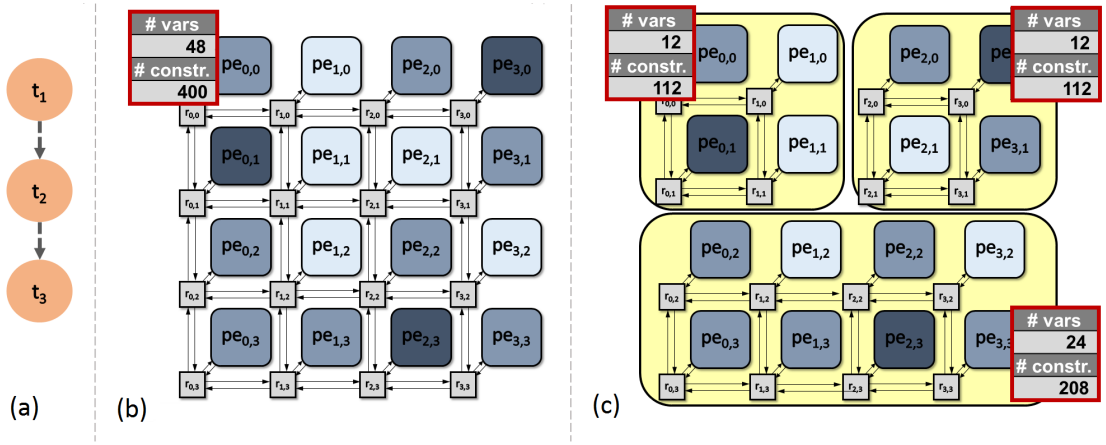


Figure 1: Application (a) and architecture graph (b) with number of variables/constraints in a SAT-formulation generating feasible mappings. (c) shows three subsystems with corresponding search-space sizes.

way, the evaluation of tens of thousands of mapping possibilities contained in many-core search spaces may become a bottleneck of DSE [14]. This, combined with ever-growing design spaces, limits the applicability of exact and exhaustive DSE techniques covering the complete search space. To illustrate typical search-space sizes many-core DSEs face nowadays, Fig. 1 shows an exemplary 3-task application (a) and the size of a Boolean Satisfiability (SAT)-problem formulation, based on [8, 19], used to generate feasible mappings on a heterogeneous 4×4 Network-on-Chip (NoC) architecture (b). This very small example already consists of $2^{48} \approx 2.81 \times 10^{14}$ mapping possibilities that have to fulfill 400 constraints to constitute a *feasible mapping*. The increasingly complex search-space sizes for three benchmark applications from the Embedded Systems Synthesis Benchmarks Suite (E3S) [2] and more realistic 8×8 and 24×24 NoCs, that will be used for evaluation later on, are shown in Table 1.[1] State-of-the art DSEs, therefore, commonly employ problem-specific heuristics or problem-independent metaheuristic optimization techniques like evolutionary algorithms (EAs) to obtain high-quality solutions from these huge search spaces within reasonable time [17]. [3] gives an overview of such techniques.

One issue that research into heuristic DSE techniques faces as a consequence of this, is the fact that the *Pareto-optimal* solutions of the MOP are usually not known and can therefore not be used for the evaluation of novel approaches. New work in this area is, thus, typically only compared to existing DSEs *that are based on (meta)heuristics as well*. It therefore remains an unanswered—but all the more important—question, whether the high-quality solutions found by these techniques are a good approximation of the global optima for increasingly complex search spaces, as encountered in many-core DSE. To bypass this problem and provide a first insight into potential shortcomings of metaheuristic optimization techniques for DSE of large-scale systems, we present a novel perspective by evaluating the effect a reduction of the search-space size has on the performance of state-of-the-art metaheuristics. By decomposing the DSE into a set of subproblems, we will be able to compare optimization quality of the decomposed problems to the solutions obtained from an optimization of the complete system. The search-space restriction (SSR) is achieved by decomposing the target architecture and limiting the exploration and optimization of application mappings to small parts of the original system. Figure 1(c) exemplifies a possible decomposition into three yellow subsystems and the respective search-space sizes, that are already dramatically reduced. Furthermore, Table 1 shows search-space sizes for a 4×4 subsystem and the benchmark applications. Since the search space of subproblems is significantly smaller, we will be able to demonstrate that the performance of modern optimization techniques (both in terms of optimization quality and time) is affected by the complexity of current DSE search spaces and can easily be improved by employing a simple

[1] All reported SAT problems have already undergone advanced preprocessing, e.g. by unit clause propagation.

Table 1: Number of *variables* and *constraints* in a SAT-based problem formulation generating feasible application mappings for benchmark applications onto various NoCs.

	4×4 NoC		8×8 NoC		24×24 NoC	
application (# tasks)	# vars	# constr	# vars	# constr	# vars	# constr
consumer (11)	162	2,026	620	7,512	5,944	71,356
telecom (14)	224	3,575	896	13,895	8,064	123,975
automotive (18)	288	3,809	1,152	14,801	10,368	132,049

SSR based on a decomposition of the architecture. In short, even a very basic SSR strategy may already outperform state-of-the-art DSEs without SSR.

The remainder of the paper is organized as follows: Sections 2 and 3 present related work and the fundamentals of DSE. The proposed SSR is introduced in Sec. 4 and evaluated using a set of benchmark applications and architectures in Sec. 5. Finally, Sec. 6 concludes the paper.

2. Related Work

Traditionally, embedded system design tailors the platform to an application and optimizes it for execution objectives and requirements. Related research into simplification techniques for DSE of embedded systems is, therefore, based on a hierarchical view and decomposition of the *application* rather than the architecture [6, 11, 15]. Recent years have seen a shift to the development and utilization of multi- and many-core architectures that offer the possibility to dynamically embed a multitude of applications. These large-scale platforms typically consist of a variety of heterogeneous processing elements (PEs) connected by a scalable Network-on-Chip (NoC) communication infrastructure [17], supporting the execution of diverse application mixes. Current DSEs for many-core systems as in [19] explore and optimize concrete application mappings to the given target architecture. A problem with this type of approach for increasingly complex systems is the size of the search space of the optimization problem that strongly depends on the number of PEs and the resulting multitude of mapping possibilities [18]. Therefore, most existing run-time—and even design-time—techniques are not exact and do not consider and evaluate all possible mappings; instead, metaheuristic techniques like EAs are often employed to determine a set of high-quality solutions, while limiting exploration time to practicable magnitudes [13, 17]. Furthermore, large-scale systems are very likely to comprise redundancy in the search space due to *architectural symmetries*, i.e. recurring patterns in the topology or distribution of PEs. Recent *symmetry-eliminating* approaches circumvent this issue by identifying symmetric subsystems [4] or reformulating the optimization problem to avoid the repeated exploration of equivalent mappings [16]. We will study the effect of SSR for both types of DSE to take into account any inherent differences between both types of design spaces. Other works in this area include *search-space pruning* techniques [14] or application-specific *decomposition* of DSE problems [12]. Finally, decomposition based on a *classification* scheme that is envisaged to automatically detect suitable subproblems in any DSE is proposed in [22]. This may be an interesting aspect of future advanced SSR techniques, whose relevance the work at hand shall motivate.

3. Fundamentals

DSE in the context of Electronic System Level (ESL) design requires the exploration and optimization of *implementations* representing application-to-architecture mappings.

System Model For many-core systems, graph-based models of application and architecture as in [1] are particularly suitable, since they easily capture the regular 2D-mesh topology of NoC-based architectures and the data dependencies inherent in modern embedded applications (Fig. 1 (a), (b)).

Definition 1 (Architecture Graph) *An NoC with dimensions X and Y is a directed graph* $G_{NoC}(P \cup R, L)$. *PEs* $pe_{x,y}$ *form the set of nodes* P *with* x, y *indicating each PE's position within the NoC. Communication is modeled using an NoC router* $r_{x,y} \in R$ *for each PE and the respective communication links* L *with maximal bandwidth capacity* bw_{max}. *Heterogeneity among PEs is captured by annotating each PE node with a* resource type.

Definition 2 (Application Graph) *An application is a directed acyclic graph* $G_{app}(V, E)$. *Vertices* $V = T \cup M$ *represent tasks* T *and messages* M. *The messages and edges* $E \subseteq (T \times M) \cup (M \times T)$ *model data dependencies in the application, since each edge connects exactly one task with one message (or vice versa). DSE requires information about properties of tasks and messages to determine feasible mappings and evaluate their run-time performance in regards to, e.g., energy consumption, payload, or period. This can be annotated to the respective vertices.*

DSE for Many-Cores DSE for modern embedded systems is an MOP of typically conflicting design objectives such as resource costs or execution time.

Definition 3 (Multi-Objective Optimization (cf. [9])) *In general, an MOP is defined as:*

$$minimize\,\{C^T\mathbf{x} \mid A\mathbf{x} \leq b\}$$

$$with\ C \in \mathbb{Z}^{n,z}, A \in \mathbb{Z}^{m,n}, b \in \mathbb{Z}^m,\ and\ x \in \{0,1\}^n.$$

In system synthesis, $C = (c_1, \ldots, c_z)$ represents z objective functions $f_i(x) = c_i^T\mathbf{x}$ with $i \in \{1, \ldots, z\}$ that are used to evaluate any implementation described by the n-dimensional *decision vector* $\mathbf{x} \in \{0,1\}^n$.[2] During DSE, variables $x_i \in \mathbf{x}$ are varied to generate a variety of differing implementations in the search for high-quality mapping possibilities. Constraints $A\mathbf{x} \leq b$ define the set of *feasible implementations* X_f, i.e. application-to-architecture mappings that enable successful execution of the application. Such constraints may, e.g., ensure that data dependencies can be respected by sufficient communication bandwidth between PEs hosting communicating tasks [6].

Pareto-Optimality of MOP Solutions Since the objective function $f = (f_1, \ldots, f_z)$ for DSE contains multiple design objectives, the MOP does not have a single optimal solution. Instead, the optimization results in a *set* of *Pareto-optimal* solutions $X_p \subseteq X_f$. In the context of DSE, an implementation is said to be Pareto-optimal if it is not *dominated* by any other explored implementation. Mathematically, this is defined as:

Definition 4 (Pareto Dominance (cf. [21])) *An objective vector* $f(\mathbf{x}) = (a_1, \ldots, a_z)$ dominates *another objective vector* $f(\mathbf{x}') = (b_1, \ldots, b_z)$, *i.e.* $f(\mathbf{x}) \succ f(\mathbf{x}')$, *iff* $\forall i : a_i \leq b_i \wedge \exists j : a_j < b_j$.

Therefore, a successful DSE will yield an interesting mix of high-quality implementations with variance in the objective values: Some implementations may require few resources, but face high latencies. Other mapping possibilities may trade off poor energy efficiency for a fast performance. At run time, a suitable mapping can thus be dynamically selected depending on current requirements.

Due to the described extreme size of many-core DSE search spaces, exhaustive search resulting in the true Pareto-front of solutions is practically impossible. Instead, (meta)heuristic techniques are used to *approximate* the Pareto-optimal solutions by optimizing for a set of *non-dominated high-quality solutions* X_{p^*}. The quality of this approximation is, however, currently not known since only relative comparisons with other (meta)heuristics are feasible.

[2] It can w.l.o.g. be assumed that typical design objectives can be formulated as minimization problems (by, e.g., negating the objective function for objectives to be maximized) [9].

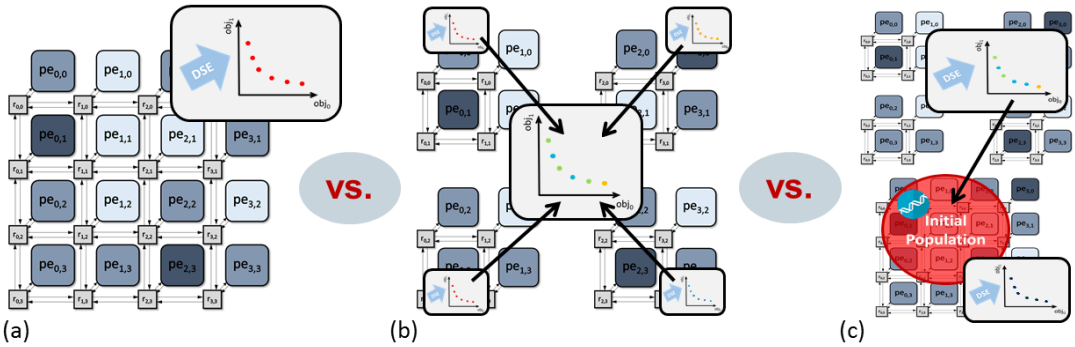


Figure 2: DSE of the complete system (a), basic subsystem DSE (b), and initial population construction (c).

4. Search-Space Restriction in DSE

This work, therefore, investigates the suitability of metaheuristics for DSE of modern many-core systems. This is the current state-of-the-art in automatic system level design, since search-space size prevents the use of exact optimization techniques. To this end, we evaluate the performance of metaheuristic optimization techniques in regard to search-space coverage and execution time on restricted sub-parts of system-level DSE, and compare them to a DSE of the complete system.

The regular layout of NoC-based many-core architectures inspires a restriction of the search space by decomposing the architecture into smaller parts, or *sub-architectures*. Search-space size can be significantly reduced by limiting the DSE to the exploration of embeddings on such subsystems, since this drastically reduces the number of mapping possibilities (cf. Fig. 1 and Table 1).

4.1. Exploration of Subsystems

In principle, any architecture can be decomposed into subsystems to reduce search-space size—even though problem-specific knowledge, e.g. about resources that are critical for the execution of the application, may be required for complex architectural layouts with non-regular network topologies, etc. As an illustrating case-study demonstrating the potential of SSR for DSE, we use regular heterogeneous many-core architectures and decompose them into suitable sub-architectures automatically and without relying on any problem-specific information (see Fig. 1(c)).

A first possibility to implement *subsystem DSE* is the exploration of one (or more) sub-architectures *instead* of the complete system (see Fig. 2(b)). The resulting set of high-quality solutions is then the Pareto-optimal combination of solutions obtained from all s explored sub-architectures, i.e. $\bigcup_s X_{p_s^*} \setminus X_{dom}$. $X_{dom} = \{\mathbf{x} \mid \exists \mathbf{x}' : f(\mathbf{x}') \succ f(\mathbf{x})\}$ (with $\mathbf{x}, \mathbf{x}' \in \bigcup_s X_{p_s^*}$) is the set of dominated solutions within the set union, according to Def. 4. This allows for a straight-forward evaluation of SSR in regard to optimization time and quality, compared to a DSE of the complete system (Fig. 2(a)).

Alternatively, we propose a *pre-exploration* of sub-architectures to select a single subsystem for extended exploration. The pre-exploration phase consists of the rapid optimization of multiple subsystems for a limited period of time. By comparing optimization qualities after the pre-exploration phase, the most promising subsystem for further exploration can easily be determined. This approach offers the advantage of exploring a variety of subsystems in the beginning, while allowing an in-depth optimization of a single subsystem, to balance exploration time and search-space coverage.

Benefits of SSR The exploration of sub-architectures improves DSE of many-cores twofold: Since large-scale architectures are likely to contain recurring patterns in the topology and distribution of PEs [4, 16], any DSE exploring concrete mappings on the complete architecture will suffer

from the exploration of *redundant implementations* that are equivalent in terms of the optimized quality numbers. The repeated exploration and evaluation of these mappings does not add any new information to the DSE; instead, exploration time is wasted on the construction and especially the often time-consuming evaluation of solutions that will not improve optimization quality. Traditional DSE techniques that do not explicitly implement mechanisms to eliminate architectural symmetries will therefore explore large areas of the search space that contribute little to none to the final set of high-quality solutions. Furthermore, many-core search spaces consist in large part of low-quality implementations that are dominated in their objective values by other solutions. These solutions will, therefore, not be part of the resulting set of high-quality solutions either.

Both problems are easily avoided by restricting DSE to subparts of the original architecture: Redundant mappings are less likely to be contained in small subsystems and the number of dominated solutions will be reduced. Since the search space of any sub-problem is furthermore significantly smaller than that of the complete system, a faster and more thorough coverage is to be expected. A final benefit is the reduced effort required to generate and evaluate subsystem implementations [5].

Drawbacks An easily rectified drawback is the problem of generalizing subsystem mappings to the complete architecture, since a particular sub-architecture might already be occupied when the application has to be mapped at run time. One possibility, as introduced in [19], is the abstraction to *equivalence classes* of application mappings: From each resulting operating point, a set of constraints on required resource types, maximal distance between communicating PEs, etc. is extracted. Now, run-time mapping is equivalent to a constraint satisfaction problem that can be fulfilled by any constellation of PEs in the system, so that dynamic mapping is possible at run time.

A second and more severe issue is the risk of losing high-quality solutions—and even the global Pareto-optima—by no longer exploring the complete architecture and all mapping possibilities it offers. For instance, mappings requiring a great number or rare constellation of PEs, specialized components, etc. cannot all be captured in the restricted search spaces of subsystems. On the other hand, the value of such implementation for run-time mapping may be limited, since the probability of being able to dynamically allocate a suitable part of the architecture that is not occupied by concurrently executed applications may be low. More importantly, the sheer size of the complete search space may prohibit current techniques from ever reaching these solutions in the first place.

4.2. Sub-System Solutions as Initial Population

To circumvent the problem of losing high-quality solutions that can only be found on the complete system, we investigate the use of subsystem DSE to construct an *initial population* as starting point for DSE of the complete system (Fig. 2(c)). A related approach bases initial population construction on subsystems of decomposed parts of the *application* [5]. In contrast, the approach proposed here does not require a hierarchical structure of the application, but is applicable to any system description that allows a decomposition of the target architecture.

First, a set of subproblems is explored for a limited period of time. These subsystem DSEs start out with *random* populations as is characteristic of metaheuristic optimization techniques [5]. The best implementations from all s subsystem DSEs are accumulated into a set of non-dominated solutions $\bigcup_s X_{p_s^*} \setminus X_{dom}$, where $X_{dom} = \{\mathbf{x} \mid \exists \mathbf{x}' : f(\mathbf{x}') \succ f(\mathbf{x})\}$ (with $\mathbf{x}, \mathbf{x}' \in \bigcup_s X_{p_s^*}$). Using these solutions as initial population for a DSE of the complete system may guide the DSE towards areas of the search space containing high-quality solutions. This offers the advantages of significantly less complex subsystem DSE—yielding high-quality results in a short exploration time, see Sec. 5—while exposing the complete search space to the optimization, so that all global Pareto-optima can theoretically be found by the procedure.

5. Experimental Evaluation

This section presents an experimental evaluation of SSR for two types of DSE: a classic exploration of concrete mappings [19] (CLASSIC) and a symmetry-eliminating DSE (SYM) that already reduces the search space by optimizing *classes* of mappings [16]. Both DSEs were implemented in the open-source optimization framework OPT4J [10]. We first present the experimental setup (Sec. 5.1) and discuss the effect of SSR on DSE time and optimization quality in Sec. 5.2.

5.1. Experimental Setup

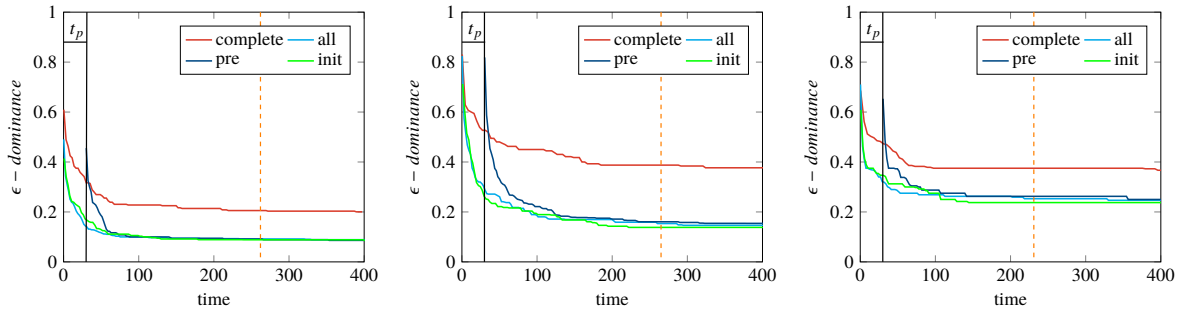
We show the effect of SSR for three applications from the E3S [2] (*consumer*, *telecom*, *automotive*) that differ in communication requirements and number of tasks. Two heterogeneous NoC-based architectures consisting of 8×8 and 24×24 PEs (each containing 3 different resource types) are compared. Since the aim of this work is to give a first indication of the potential of subsystem DSE, automatically generated 4×4 sub-architectures were used for all subsystem DSEs. Advanced decomposition methods may, of course, yield even better results and may also be required for more complex architectural topologies. The decomposition created 2 non-isomorphic 4×4 subsystems for the 8×8 NoC and 4 such subsystems for the 24×24 NoC. For the basic SSR approach, two methods have been implemented: We first discuss a simple strategy of exploring all automatically generated sub-architectures (**-all**) and combining the resulting approximated Pareto-fronts according to Def. 4. A second approach is the optimization of a single subsystem after a *pre-evaluation phase* (**-pre**). To this end, all subsystems are explored for a limited period of time (300 generations); subsequently, the subsystem offering the best optimization quality after the pre-exploration phase is selected for further exploration. For comparison, the optimization quality of the single best subsystem DSE is used; pre-exploration time is included in the reported execution times. Furthermore, we investigate a strategy to construct the initial population for DSE of the complete system (**-init**). Here, all subsystems are optimized for 500 generations; afterwards, the complete system is explored using the combined non-dominated subsystem solutions as initial population. Results are averaged over 10 DSE runs for each implemented approach with a population size of 100 individuals and minimized optimization objectives *resource cost* [number of PEs], *energy consumption* [mJ], and *latency* [ms].

5.2. Experimental Results

We compare the quality of all SSR approaches to a DSE of the complete system. In particular, we provide an analysis of optimization quality depending on exploration time to analyze the performance of subsystem DSEs both in terms of search-space coverage and execution-time requirements. Multi-objective optimization does not result in a single optimal solution, but in a set of non-dominated points in the objective space. To compare different objective fronts with each other, the well-established quality measures ϵ-dominance [7] and hypervolume [20] are used in the following analysis. ϵ-dominance measures the distance between the solutions of one approach and a *reference set* accumulated from all approaches to be compared. Hypervolume analyses the volume in the

Table 2: Optimization quality, depending on DSE time, and coverage of non-dominated solutions (CLASSIC).

		8×8 NoC						24×24 NoC					
						coverage						coverage	
application	approach	ϵ-dom. (50 s)	ϵ-dom. (400 s)	hyperv. (50 s)	hyperv. (400 s)	sub/orig	orig/sub	ϵ-dom. (50 s)	ϵ-dom. (1000 s)	hyperv. (50 s)	hyperv. (1000 s)	sub/orig	orig/sub
consumer	CLASSIC	0.264	0.200	0.636	0.475	-	-	0.581	0.428	1.159	0.914	-	-
	CLASSIC-all	**0.117**	0.089	**0.218**	0.109	0.946	0.014	**0.117**	0.086	**0.209**	**0.078**	**0.978**	0.000
	CLASSIC-pre	0.217	**0.086**	0.467	**0.074**	**0.960**	0.005	0.225	**0.085**	0.468	**0.078**	0.976	0.000
telecom	CLASSIC	0.480	0.377	1.056	0.759	-	-	0.707	0.667	1.539	1.421	-	-
	CLASSIC-all	**0.230**	0.162	**0.512**	0.236	0.907	0.012	**0.331**	**0.153**	**0.652**	**0.274**	**0.990**	0.000
	CLASSIC-pre	0.327	**0.146**	0.700	**0.221**	**0.930**	0.011	0.409	0.163	0.955	0.334	0.739	0.000
automotive	CLASSIC	0.421	0.367	1.162	0.881	-	-	0.700	0.568	1.564	1.378	-	-
	CLASSIC-all	**0.284**	**0.247**	**0.640**	**0.476**	0.948	0.002	**0.363**	0.213	**0.768**	0.407	0.959	0.000
	CLASSIC-pre	0.375	0.250	0.953	0.488	**0.958**	0.000	0.375	**0.204**	0.973	**0.396**	**0.979**	0.000



(a) consumer **(b)** telecom **(c)** automotive

Figure 3: ϵ-dominance over time for all applications on an 8×8 NoC (DSE: CLASSIC).

objective space covered by each approach and the reference. For both metrics, smaller values ($\rightarrow 0$) correspond to a higher optimization quality.

Sub-System DSE for Initial Population Construction DSE of the complete system did *not* benefit from initial population construction in any experiment (at least for the basic architecture decomposition mechanism proposed in this work). In particular, no new non-dominated solutions where found after switching to the complete system (see approach **-init** in Fig. 3, where optimization quality after the switching point, indicated by the orange dotted line, does not improve), giving further evidence that the extreme size of the complete search spaces prohibits successful exploration. The following discussion of results, therefore, omits a detailed examination of **-init**.

DSE Time and Optimization Quality Table 2 shows optimization quality, quantified in ϵ-dominance and hypervolume, of each approach after 50 s and 400 s/1000 s of DSE CLASSIC. In all experiments, the SSR approaches **-all** and **-pre** converge to significantly higher optimization qualities very quickly, whereas DSE of the complete system is inferior. Especially for the 24×24 NoC, DSE quality is worse by a factor of up to ≈ 5.04 for ϵ-dominance and ≈ 11.72 for hypervolume. Fig. 3 shows a graphical representation of ϵ-dominance over DSE time for all applications on the 8×8 NoC, where it furthermore becomes clear that DSE quality stagnates after a certain exploration time. Therefore, metaheuristic techniques will not be able to further improve the set of non-dominated solutions for complex search spaces even if more DSE time is provided; this demonstrates the need for novel approaches in DSE of large-scale systems all the more. Results for the *symmetry-eliminating* DSE are shown in Table 3. Here, sub-system approaches converge to significantly superior optimization qualities even more quickly than for the complete DSE.

The choice of SSR approach does *not* significantly influence performance for the current basic decomposition mechanism. Approach **-pre** suffers from the pre-exploration time in the beginning (shown as t_p in Fig. 3); optimization quality of the selected subsystem, however, quickly converges to similar values and can even exceed **-all**'s quality in some cases—albeit by only very small fractions.

Fig. 4 illustrates the difference in optimization quality by showing the non-dominated solutions for

Table 3: Optimization quality, depending on DSE time, and coverage of non-dominated solutions (SYM).

		8×8 NoC						24×24 NoC					
						coverage						coverage	
application	approach	ϵ-dom. (100 s)	ϵ-dom. (15,000 s)	hyperv. (100 s)	hyperv. (15,000 s)	sub/orig	orig/sub	ϵ-dom. (100 s)	ϵ-dom. (15,000 s)	hyperv. (100 s)	hyperv. (15,000 s)	sub/orig	orig/sub
consumer	SYM	0.294	0.004	0.633	0.003	-	-	0.294	0.004	0.633	0.003	-	-
	SYM-all	**0.003**	**0.003**	**0.002**	**0.002**	**0.952**	0.167	0.011	0.011	0.012	0.012	**0.536**	0.452
	SYM-pre	0.004	**0.003**	0.003	**0.002**	0.508	0.442	**0.004**	**0.003**	**0.003**	**0.002**	0.508	0.442
telecom	SYM	0.808	0.486	1.346	1.031	-	-	0.808	0.486	1.346	1.031	-	-
	SYM-all	**0.079**	0.055	**0.074**	0.047	0.503	0.097	0.097	0.095	**0.084**	0.071	**0.656**	0.004
	SYM-pre	0.082	**0.054**	0.081	**0.042**	**0.656**	0.019	**0.086**	**0.055**	0.085	**0.047**	0.503	0.097
automotive	SYM	0.250	0.114	0.433	0.113	-	-	0.513	0.321	1.108	0.685	-	-
	SYM-all	**0.100**	0.087	**0.095**	**0.071**	0.758	0.089	**0.107**	0.107	**0.100**	0.091	0.890	0.030
	SYM-pre	0.107	**0.079**	0.122	0.076	**0.772**	0.071	0.114	**0.092**	0.117	**0.078**	**0.908**	0.000

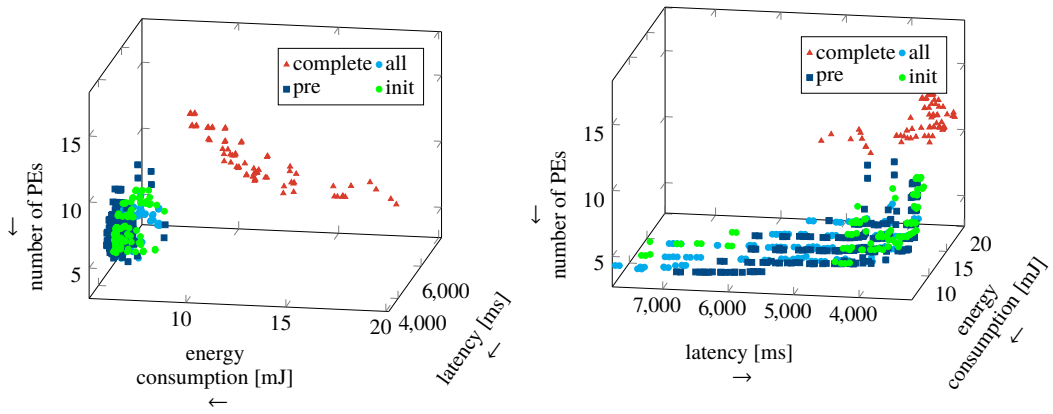


Figure 4: Non-dominated solutions in the 3-dimensional objective space for application *automotive* on a 24×24 architecture (DSE: CLASSIC).

each approach in the 3D-objective space (application *automotive* on the 24×24 NoC, DSE CLASSIC). Subsystem solutions accumulate in superior regions of the objective space that are not reached by a DSE of the complete system, explaining the significantly improved quality numbers.

Coverage Finally, we analyze the *coverage* between the resulting sets of non-dominated solutions. In Tables 2 and 3 columns *sub/orig* describe the percentage of solutions from the complete DSE that each subsystem approach was able to find. For DSE CLASSIC, subsystem approaches typically find ≈ 95% of solutions that were also discovered by a DSE of the complete system. On the other hand, the complete DSE only covers a fraction of results of each subsystem approach (at most ≈ 1.4%) (shown in columns *orig/sub*). This indicates again that too complex search spaces may prohibit successful exploration and optimization. For DSE SYM, the differences in coverage are not as extreme; the complexity reduction by the implemented symmetry-elimination, thus, successfully reduces search-space size to some extent. Nevertheless, subsystem DSE is superior for classic as well as symmetry-eliminating DSE, especially in terms of execution time.

6. Conclusion

This work investigates the suitability of established metaheuristics in the context of DSE for many-core systems and their increasingly complex search spaces. A simple SSR limiting the DSE to automatically generated subsystems significantly improves exploration time as well as optimization quality, clearly demonstrating the limitations of metaheuristic optimization techniques for large-scale problems. Initial population construction for a DSE of the complete system—using a basic decomposition—merely produced results comparable to those of subsystem DSEs; this emphasizes the need for advanced decomposition approaches and may give an indication for prospective research directions into more elaborate SSR techniques for DSE of large-scale systems.

References

[1] Blickle, Tobias, Jürgen Teich, and Lothar Thiele: *System-Level Synthesis Using Evolutionary Algorithms*. Design Automation for Embedded Systems, 3(1):23–58, 1998.

[2] Dick, Robert: *Embedded System Synthesis Benchmarks Suite (E3S)*. `http://ziyang.eecs.umich.edu/~dickrp/e3s/`.

[3] Glaß, Michael, Jürgen Teich, Martin Lukasiewycz, and Felix Reimann: *Hybrid Optimization Techniques for System-Level Design Space Exploration*. In Ha, Soonhoi and Jürgen Teich (editors): *Handbook of Hardware/Software Codesign*, pages 1–31, Dordrecht, 2017. Springer Netherlands.

[4] Goens, Andrés, Sergio Siccha, and Jerónimo Castrillón: *Symmetry in Software Synthesis*. CoRR, abs/1704.06623, 2017. http://arxiv.org/abs/1704.06623.

[5] Haubelt, Christian, Jürgen Gamenik, and Jürgen Teich: *Initial Population Construction for Convergence Improvement of MOEAs*. In Coello Coello, Carlos A., Arturo Hernández Aguirre, and Eckart Zitzler (editors): *Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005. Proceedings*, pages 191–205, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[6] Haubelt, Christian and Jürgen Teich: *Accelerating Design Space Exploration using Pareto-Front Arithmetics*. In *Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference 2003*, pages 525–531, 2003.

[7] Laumanns, Marco, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler: *Combining Convergence and Diversity in Evolutionary Multiobjective Optimization*. Evol. Comput., 10(3):263–282, September 2002.

[8] Lukasiewycz, Martin, Michael Glaß, Christian Haubelt, and Jürgen Teich: *SAT-decoding in Evolutionary Algorithms for Discrete Constrained Optimization Problems*. In *IEEE Congress on Evol. Comp.*, 2007.

[9] Lukasiewycz, Martin, Michael Glaß, Christian Haubelt, and Jürgen Teich: *Solving Multi-objective Pseudo-Boolean Problems*. In Marques-Silva, João and Karem A. Sakallah (editors): *Theory and Applications of Satisfiability Testing – SAT 2007: 10th International Conference, Lisbon, Portugal, May 28-31, 2007. Proceedings*, pages 56–69, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[10] Lukasiewycz, Martin, Michael Glaß, Felix Reimann, and Jürgen Teich: *Opt4J: A Modular Framework for Meta-heuristic Optimization*. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 1723–1730, New York, NY, USA, 2011. ACM, ISBN 978-1-4503-0557-0.

[11] Neubauer, Kai, Christian Haubelt, and Michael Glaß: *Supporting Composition in Symbolic System Synthesis*. In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*.

[12] Padmanabhan, Shobana, Yixin Chen, and Roger D. Chamberlain: *Decomposition Techniques for Optimal Design-space Exploration of Streaming Applications*. In *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '13, pages 285–286, New York, NY, USA, 2013. ACM.

[13] Panerati, Jacopo, Donatella Sciuto, and Giovanni Beltrame: *Optimization Strategies in Design Space Exploration*. In Ha, Soonhoi and Jürgen Teich (editors): *Handbook of Hardware/Software Codesign*, pages 189–216, Dordrecht, 2017. Springer Netherlands.

[14] Piscitelli, Roberta and Andy D. Pimentel: *Design Space Pruning through Hybrid Analysis in System-level Design Space Exploration*. In *Design, Automation, Test in Europe Conference Exhibition (DATE)*, pages 781–786, 2012.

[15] Rao, D. Sreenivasa and Fadi J. Kurdahi: *Hierarchical Design Space Exploration for a Class of Digital Systems*. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1(3):282–295, Sept 1993, ISSN 1063-8210.

[16] Schwarzer, Tobias, Andreas Weichslgartner, Michael Glaß, Stefan Wildermann, Peter Brand, and Jürgen Teich: *Symmetry-Eliminating Design Space Exploration for Hybrid Application Mapping on Many-Core Architectures*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(2):297–310, Feb 2018.

[17] Singh, Amit K., Muhammad Shafique, Akash Kumar, and Jörg Henkel: *Mapping on Multi/Many-core Systems: Survey of Current and Emerging Trends*. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–10, May 2013.

[18] Srinivasan, V. Prasanna and A. P. Shanthi: *A Survey Of Research And Practices In Multiprocessor System On Chip Design Space Exploration*. Journal of Theoretical and Applied Information Technology, 64:1, 2014.

[19] Weichslgartner, Andreas, Deepak Gangadharan, Stefan Wildermann, Michael Glaß, and Jürgen Teich: *DAARM: Design-Time Application Analysis and Run-Time Mapping for Predictable Execution in Many-Core Systems*. In *Proc. of CODES+ISSS 2014*, pages 34:1–34:10, 2014.

[20] Zitzler, Eckart and Lothar Thiele: *Multiobjective Optimization Using Evolutionary Algorithms — A Comparative Case Study*. In *In Proc. of PPSN*, pages 292–301, 1998.

[21] Zitzler, Eckart, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane G. da Fonseca: *Performance Assessment of Multiobjective Optimizers: An Analysis and Review*. IEEE Transactions on Evolutionary Computation, 7(2):117–132, April 2003.

[22] Zverlov, Sergey: *Decomposition of Design Space Exploration Problems in the Context of Model-Based Development*. In *Proc. of the Doctoral Symposium at the 19th ACM/IEEE International Conference of Model-Driven Engineering Languages and Systems 2016 (MoDELS 2016)*, 2016.