Interleaving Methods for Hybrid System-level MPSoC Design Space Exploration

Roberta Piscitelli and Andy D. Pimentel
Computer Systems Architecture group, Informatics Institute, University of Amsterdam, The Netherlands
Email: {r.piscitelli,a.d.pimentel}@uva.nl

***Abstract*—System-level design space exploration (DSE), which is performed early in the design process, is of eminent importance to the design of complex multi-processor embedded system architectures. During system-level DSE, system parameters like, e.g., the number and type of processors, the type and size of memories, or the mapping of application tasks to architectural resources, are considered. Simulation-based DSE, in which different design instances are evaluated using system-level simulations, typically are computationally costly. Even using high-level simulations and efficient exploration algorithms, the simulation time to evaluate design points forms a real bottleneck in such DSE. Therefore, the vast design space that needs to be searched requires effective design space pruning techniques. This paper presents and studies different strategies for interleaving fast but less accurate analytical performance estimations with slower but more accurate simulations during DSE. By interleaving these analytical estimations with simulations, our hybrid approach significantly reduces the number of simulations that are needed during the process of DSE. Experimental results have demonstrated that such hybrid DSE is a promising technique that can yield solutions of similar quality as compared to simulation-based DSE but only at a fraction of the execution time.**

I. Introduction

Platform based design of heterogeneous multi-processor system-on-chip (MPSoC) systems is becoming today's predominant design paradigm in the embedded systems domain [18]. In contrast to more traditional design paradigms, platform based design shortens design time by eliminating the effort of the low-level design and implementation of system components. A platform based design environment typically consists of a fixed, parameterizable platform or a set of (parameterizable) components that can be combined in specific ways to compose a platform. The parameters make it possible to adjust platforms and individual components to the required application domain and platform design requirements. However, as the number of possible system candidates increases exponentially with the number of parameters, traditional design space exploration (DSE) methods fall short. As a consequence, there has been a recent increase of research attention focusing on efficient and effective system-level exploration techniques to identify those parameters that result in an optimal system.

System parameters in system-level DSE typically include the number and type of processors in the system, the sizes of the different memories in the system, or the mapping of application tasks to the underlying architectural resources. A multi-dimensional design space – the so-called parameter space – can be constructed by using each parameter type as an axis of the design space. The design criteria for a system can usually be translated to one or multiple objectives, like power consumption, system performance or cost. The parameter space maps onto the objective space by associating objective values to each point in the parameter space. If the complete objective space were given, then a designer could easily select those system candidates that meet the design requirements or that are optimal in some pre-defined way. In practice, however, it is infeasible to obtain a representation of the objective space that is both accurate and complete.

State-of-the-art solutions for system-level DSE are essentially composed of two elements: the evaluation of a single design point in the design space in terms of objective values like performance and power consumption, and a mechanism for traversing the design space to search for an optimal (set of) design point(s). To evaluate a single design point, roughly three approaches are available: 1) performing measurements on a (prototype) implementation, 2) simulation-based measurements and 3) estimations based on some kind of analytical model. Each of these methods has different properties with respect to evaluation time and accuracy. Evaluation of prototype implementations provides the highest accuracy, but long development and/or synthesis times prohibit evaluation of many design options. Analytical estimations are considered the fastest, but accuracy is limited since they are typically unable to sufficiently capture particular intricate system behavior. Simulation-based evaluation fills up the range in between these two extremes: both highly accurate (but slower) and fast (but less accurate) simulation techniques are available. This trade-off between accuracy and speed is very important, especially for early system-level DSE in which the design space that needs to be explored is vast and some accuracy can often be traded for efficiency to cope with these large design spaces. Current DSE efforts typically use simulation or analytical models to evaluate single design points together with a heuristic search method [6] to search the design space. These DSE methods search the design space using only a finite number of design-point evaluations, not guaranteeing to find the absolute optimum in the design space, but they reduce the design space to a set of design candidates that meet certain requirements or are close to the optimum with respect to certain objectives.

Our focus is on *system-level mapping DSE*, where mapping involves two aspects: 1) allocation and 2) binding. Allocation deals with selecting the architectural components in the

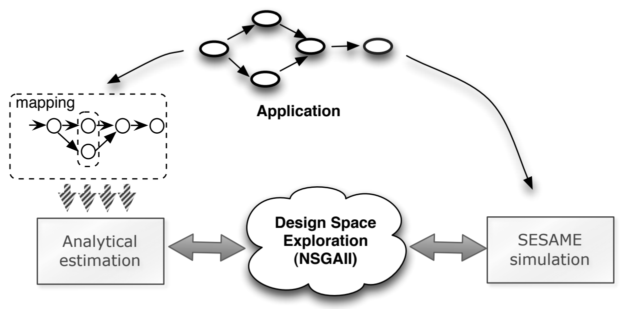

Fig. 1. Driving experiments with the expected throughput.

MPSoC platform architecture that will be involved in the execution of the application workload (i.e., not all platform components need to be used). Subsequently, the binding specifies which application task or application communication is performed by which MPSoC component. As mentioned above, state-of-the-art DSE approaches typically use either simulation or an analytical model to evaluate mappings, where simulative approaches prohibit the evaluation of many design options due to the higher evaluation performance costs and analytical approaches suffer from accuracy issues. This paper deals with a new, hybrid form of DSE, which has recently been proposed in [17], combining simulations with analytical estimations to prune the design space in terms of application mappings that need to be evaluated using simulation. To this end, the DSE technique uses an analytical model that estimates the expected throughput of an application (which is a natural performance metric in our case given our target application domain which is multimedia and streaming applications) given a certain architectural configuration and application-to-architecture mapping. In the majority of the search iterations of the DSE process, the throughput estimation avoids the use of simulations to evaluate the design points. However, since the analytical estimations may in some case be less accurate, the analytical estimations still need to be interleaved with simulative evaluations in order to ensure that the DSE process is steered into the right direction. In this paper, we study different techniques for interleaving these analytical and simulative evaluations in our hybrid DSE. We will demonstrate that by properly interleaving the analytical and simulative estimations, significant efficiency improvements can be obtained while still producing similar solutions in terms of quality as compared to pure simulation-based DSE.

The remainder of the paper is organized as follows. The next section briefly describes how the throughput analysis is combined with simulation in the DSE process. Section 3 proposes a range of different interleaving methods techniques. In section 4, we present a number of experiments in which we compare the different interleaving methods for our hybrid DSE. Section 5 describes related work, after which Section 6 concludes the paper.

II. Hybrid Design Space Exploration

In Figure 1, the hybrid DSE framework is shown. The framework adopts an approach in which the DSE process interleaves simulations with analytical throughput analysis, where both types of evaluation deploy the same initial application model. The throughput analysis is used to quickly predict the performance consequences of different design points as represented by the application mapping on the underlying architecture. Evidently, the aim is to interleave these fast analytical evaluations with the slower simulative evaluations in a way such that most evaluations are performed analytically. As a consequence, such an approach could significantly improve the efficiency of the DSE process, allowing for searching a much larger design space.

Applications in our DSE framework are modeled using Kahn Process Networks (KPNs) [8], in which parallel processes communicate with each other via unbounded FIFO channels. As will described later on, before performing analytical throughput analysis for such a KPN, we first need to perform some transformations to the application graph of the KPN in order to take into account mapping decisions. The subsequent throughput analysis – performed on the transformed KPN – is fast and allows for capturing the throughput trend for different mappings in a reasonable accurate fashion [17]. The analysis requires the process workloads W_{P_i} as a parameter for the throughput modeling. The workload W_{P_i} of an application process P_i denotes the number of time units that are required to execute a single invocation of the process, i.e., the pure computational workload, excluding the communication. It should be provided by the designer who can obtain it, for example, by executing the process once on the target platform, or by using an instruction set simulator.

However, as will also be shown later on, the analytical throughput model may encounter accuracy problems when the (transformed) application graph is cyclic. Addressing these problems by improving the accuracy of the analytical model in case of cycles through e.g. using fixed-point iterative solutions would be possible, but this would defeat our purpose. It would significantly slow down our analytical estimations, possibly making them even slower than fast system-level simulations like those performed by our Sesame framework (as discussed below). Therefore, we have chosen for an alternative solution. To correct the accuracy errors during DSE, we interleave the throughput estimation with fast system-level simulations.

To evaluate design points by means of simulation, we deploy the Sesame simulation framework [16]. Sesame enables rapid performance evaluation of different MPSoC architecture designs, application to architecture mappings, and hardware/software partitionings with a typical accuracy of 5% compared to the real implementation [16], [13]. In Sesame, MPSoC system models are comprised of separate application and architecture models, which are linked together by a mapping model. This mapping model specifies what application tasks and communications are implemented by what architectural resources. This mapping step has been implemented using trace-driven co-simulation of the application and architecture models. In this approach, traces with computational and communication events generated by an application model and consumed by an architecture model are an abstract

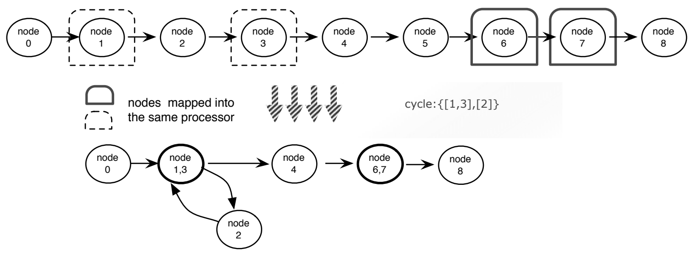

Fig. 2. Process merging in an example Kahn Process network.

representation of the workload imposed on the architecture.

The interleaving of analytical and simulative evaluations in the DSE process is determined by the value of a function ϕ: $\phi = 1$ implies that simulative evaluation is used and otherwise analytical evaluation is performed. The setting of ϕ is done according to specific *switching criteria* chosen in the DSE process. Exactly this function ϕ and the switching criteria are the topic of study in this paper, as will be further explained explained in Section 3.

In our DSE framework, we use the widely-used NSGAII Genetic Algorithm (GA) [3] to actually search through the mapping design space. This results in a hybrid DSE method with the following steps:

1) Generate an initial population of unique mappings.
2) Transform the application KPN according to the mappings in the population and build the corresponding merged KPNs.
3) Perform the static throughput analysis for the merged KPN graphs and identify the best mappings based on the highest estimated throughput.
4) In case of $\phi = 1$, we interleave the throughput analysis with real simulation, in order to correct the ranking in the NSGAII GA.
5) Verify the stopping criterion. If the mapping population within the NSGAII algorithm remains unchanged or a maximum number of iterations has been performed, the algorithm stops. Otherwise, change the mapping population using NSGAII's genetic operators, and restart from the third step.

We perform the analytical throughput analysis at the KPN level [17]. As the performance of a KPN is mapping dependent, the mapping needs to be represented inside the KPN itself. To this end, *merging transformations* are applied to the KPN to reflect the mapping of the different processes. Consequently, if two processes are mapped onto the same architectural component, they are merged into a single process in the KPN, as is illustrated in Figure 2. Figure 2 shows the initial example KPN consisting of nine processes. Performing throughput analysis of this KPN assumes that each process is mapped onto a different processor and each KPN channel is mapped onto a unique communication memory in the MPSoC (i.e., all the connections are point-to-point connections). The transformed KPN subsequently reflects the decisions that, respectively, KPN processes 1,3 and 6,7 are mapped onto a single processor. Subsequently, to assess the performance

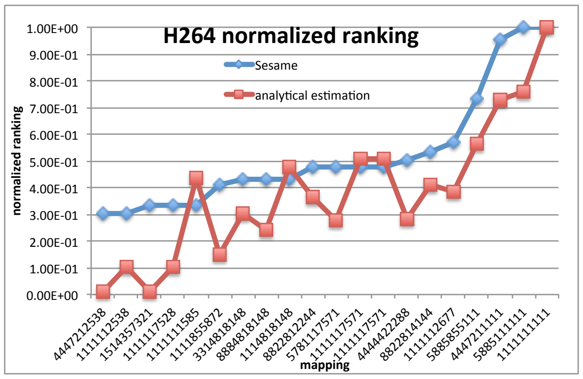

Fig. 3. Ranking a GA population using analytical estimation and Sesame simulation.

of such a mapping decision, we perform throughput analysis on the transformed KPN. This throughput analysis relies on calculating the throughput τ_{P_i} of a process (i.e., node) P_i for all KPN processes and propagation of the lowest process throughput to the sink process. Here, we use a depth first search to determine the order of the processes for propagating throughputs. For more details about the throughput analysis, the interested reader is referred to [17].

It is possible that the aforementioned merging transformation to account for mapping decisions might introduce cycles in the transformed KPN. As shown in Figure 2, if processes 1,3 are mapped onto the same processor, this results in a cycle containing process 2 and the merged process $1, 3$. For nodes 6, 7 the resulting merged node does not generate any cycle. Cycles in a KPN are responsible for sequential execution of some of the processes involved in the cycle. The sequential execution can vary from a single initial delay to a delay at each execution of some of the processes. For accurate throughput modeling, these cycles must be taken into account. Since we use a conservative approximation to estimate the performance in case of cycles, the analytical throughput analysis may present inaccuracies in case cycles are introduced in the transformed KPN, especially when there are many and/or complex cycles [17].

To demonstrate these inaccuracies, please consider Figure 3. For a DSE experiment with a H264 decoder application, this graph shows a snapshot of a single GA search iteration. More specifically, it shows the performance ranking of the design points (i.e. mappings) in the population of the search iteration when evaluating them either using Sesame or analytical estimation. The y-axis shows the normalized performance and the x-axis shows the different design points in the GA population, where the integer strings refer to the processor identifiers the application processes are mapped on. E.g., a string "1111112677" means that tasks 1 to 6 are mapped on processor 1, task 7 is mapped on processor 2, etc. The design points on the x-axis have been ordered according to the performance estimation as obtained by Sesame. This implies that the Sesame-based ranking of the population shows a monotonically increasing curve. However, as this is not true

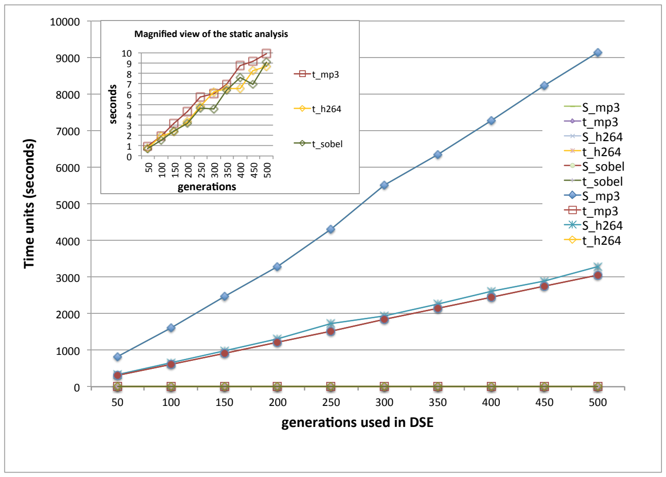

Fig. 4. DSE times using different methods.

for the curve of the analytical estimations, this ranking clearly shows prediction errors.

III. Interleaving methods

Using analytical throughput estimation as fitness function during DSE can yield significant efficiency improvements. To demonstrate this, Figure 4 shows the wall-clock times for a DSE experiment, using a NSGAII GA, for a heterogeneous 8-processor MPSoC and three multimedia applications: an Mp3 decoder, a H264 decoder, and a Sobel filter for edge detection in images. The curves labeled with an "S_" prefix show the DSE times when only using Sesame simulations, as a function of the number of generations used in NSGAII. The curves with a "t_" prefix show the results of exclusively using static throughput estimation during DSE. Clearly, the DSE based on analytical throughput analysis can be three orders of magnitude faster than simulation-based DSE. Avoiding simulation-based evaluations by replacing them with analytical evaluations therefore appears to be a promising technique for optimizing the DSE process.

The question that remains open is how to exactly perform the interleaving between analytical and simulative evaluations. Here, the ratio between the number of analytical and simulative evaluations plays an important role as this provides a valuable accuracy-performance trade-off. In addition, the decision of *when* (in time) to perform a simulative evaluation instead of an analytical estimation is an important factor in the interleaving strategy. In the remainder of this section, we will propose different strategies for interleaving analytical and simulative evaluations, which subsequently will be assessed in the next section.

A. Fixed-frequency interleaving

This is the simplest form of interleaving in which a fixed frequency k is chosen such that every k-th search iteration is performed using simulation-based evaluation instead of analytical estimation. For example, in case of 100 search iterations and $k = 10$, every 10th search iteration is performed using Sesame simulation, thereby reducing the number of simulations by 90% (9 out of 10 search iterations are performed using analyticcal estimation).

B. Switching method based on the bisection of the generation space

In this method, we divide the iteration space of the DSE according to the number of generations of the genetic algorithm used (NSGAII in our case). We switch from one method to the other (from simulation to analytical estimation, or vice versa) according to the number of generations executed by the genetic algorithm, as illustrated in Figure 5. More specifically, $\phi = 1$ if $c_{gen} == K$, where c_{gen} is the current search iteration and n_{gen} is the total number of DSE generations.

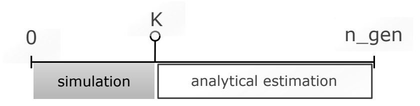

Fig. 5. NSGAII generation space using the bisection method.

C. Temperature-based interleaving

This method is inspired by the simulated annealing technique. That is, the probability of performing a Sesame simulation increases with the number of generations examined in the genetic algorithm. More specifically,

$$\phi = 1 \text{ if } U([0,1]) >= T$$

$$\text{where } T = \frac{c_{gen}}{n_{gen}}$$

where $U([0,1])$ is a uniform random distribution, and the temperature T is given by the ratio between the current generation c_{gen} and the total number of generations n_{gen} used for the design space exploration.

D. Population-property based interleaving

The last method we propose is based on the properties of the population in each generation of the genetic algorithm. It bases the decision on whether to use simulation or analytical estimation on the percentage of design points in the genetic algorithm's population that contains a cycle in the generated mapping. More specifically, the decision is based on the following algorithm:

```
**for** $d_i \in population$ **do**
    verify if $d_i$ generates a cycle
    **if** $d_i$ contains a cycle **then**
        $n_{cycles}$++;
    **end if**
**end for**
**if** $\frac{n_{cycles}}{n_{pop}} \times 100 >= K$ **then**
    $\phi = 1$;
**end if**
```

where n_{pop} is the number of mappings in the population and the threshold value K is a chosen proportion of the population.

IV. Experimental results

To evaluate the different interleaving methods, we have experimented with a DSE case study for a heterogeneous MPSoC platform consisting of up to 8 processors (interconnected by a crossbar) of possibly different types: MIPS, ARM or StrongARM. The DSE experiment is performed for three multimedia applications: an Mp3 decoder, a H264 decoder, and a Sobel filter for edge detection in images. The exploration considers two optimization objectives, namely performance (execution time) and system cost, and has been implemented using a NSGAII genetic algorithm performing 200 search iterations. Since genetic algorithms are stochastic processes, all results are averages from 10 execution runs.

To quantify the quality of the obtained Pareto fronts for the different interleaving methods, we consider two aspects: how close the found solutions are to a reference Pareto front and the spread of the solutions along the Pareto front. To this end, we use the hypervolume (HV) and ∇ metrics. The HV metric [20] measures the hypervolume of the objective space covered by members of a Pareto optimal set and a reference point. It represents the size of the region dominated by the solutions in the Pareto optimal set. The reference point can simply be found by constructing a vector of worst objective values. The hypervolume metric is interesting because it is sensitive to the closeness of solutions to the true Pareto optimal set as well as the distribution of solutions across the objective space. The hypervolume value is calculated by summing the volume of hyper-rectangles constructing the hypervolume. A Pareto optimal set with a large value for the hypervolume is desirable [19].

The normalized ∇ metric [5] measures the spread of solutions. It refers to the area of a rectangle formed by the two extreme solutions in the objective space, thus a bigger value spans a larger portion and therefore is better. The ∇ metric calculates the volume of a hyperbox formed by the extreme objective values observed in the Pareto optimal set:

$$\nabla = \prod_{m=1}^{M} (f_m^{max} - f_m^{min}) \qquad (1)$$

Where M is the number of objectives, (f_m^{max} and f_m^{min}) the maximum and respectively minimum values of the m^{th} objective in the Pareto optimal set. A bigger value spans a larger portion and therefore is better.

For the HV and ∇ metrics, we use *relative values*. That is, we relate the HV and ∇ values for our hybrid DSE experiments against those from a reference Pareto front. The reference Pareto fronts – one for each application – were obtained by combining the Pareto optimal solutions from 10 runs of Sesame-based DSE. This implies that, e.g., a HV (∇) value of 1.0 means that the experiment in question yields the same HV (∇) value as the reference Pareto front.

In Figure 6, the average hypervolume (HV) and relative spread (∇) values (averaged over 10 runs) are shown for the different DSE methods applied to the Mp3 decoder, H264 decoder, and Sobel filter applications. The *Sesame-only* results (left-most bars) form the baseline for our hybrid DSE experiments. These results are averages for a single run (averaged over 10 separate runs) of Sesame-based DSE. So, no hybrid DSE and interleaving are performed in this case, and it solely compares a single simulation-only DSE run to the reference Pareto front. The remaining bars show the results for the various hybrid DSE approaches. The label *Bisection12-K* refers to the bisection-based interleaving method in which the DSE start with simulation and switches to analytical estimation after K generations. Here, we have experimented with K values that equal to 25, 50, $\cdots$, 175 and $n_{gen} = 200$. Similarly, the label *Bisection21-K* refers to the same bisection-based interleaving but then starting with analytical estimations followed by simulations. The label *Pop-based-K* subsequently refers to the population based interleaving method with a certain K value. In our case, we have varied K from 10% up to 90%. The fixed-frequency based interleaving method has been applied with K values of 1%, 2%, 3%, 5%, and 10%, as indicated by the labels in Figure 6.

A number of observations can be made from Figure 6. Looking at the hypervolume, it is clear that hybrid DSE methods are capable of obtaining Pareto fronts with similar HV values as Sesame-only DSE, but at a fraction of the execution time (which will be shown later on). The spread of solutions (∇) in the obtained Pareto fronts, however, is highly dependent on the interleaving method as well as on the application under study. The population-based method and fixed-frequency approach with very low simulation frequencies clearly exhibit poorer ∇ values. To a lesser extent this is also true for the bisection-based method where the DSE starts with simulations and ends with analytical estimations.

There is no clear winner among the hybrid DSE methods, but looking at the HV/∇ combination, the fixed-frequency interleaving with $K = 10\%$ seems to perform slightly better than the other methods. Overall, the fixed-frequency interleaving with $K \geq 3\%$ and the bisection-based approach where the DSE starts with analytical estimations and ends with simulations (i.e., *Bisection21*) appear to outperform the other hybrid DSE methods.

In Figure 7, the execution times for the different DSE experiments are shown. As can be seen, a Sesame-only DSE experiment of 200 search iterations can take up to several thousands of seconds (like for Mp3). However, by interleaving simulations with analytical estimations most hybrid DSE techniques can significantly reduce the execution time of the DSE experiments. Only temperature-based interleaving fails to substantially improve the DSE execution times as it still uses a relative high number of simulations. The population-based interleaving yields the highest performance improvements. But, as shown in Figure 6, the quality of the Pareto fronts of this interleaving technique is fairly poor in terms of spread of solutions. The fixed-frequency interleaving with $K = 10\%$ reduces the execution time of the DSE by a factor 4, while a Bisection21 interleaving with $K = 100$ yields a 6 to 8 times performance improvement. We note that these time

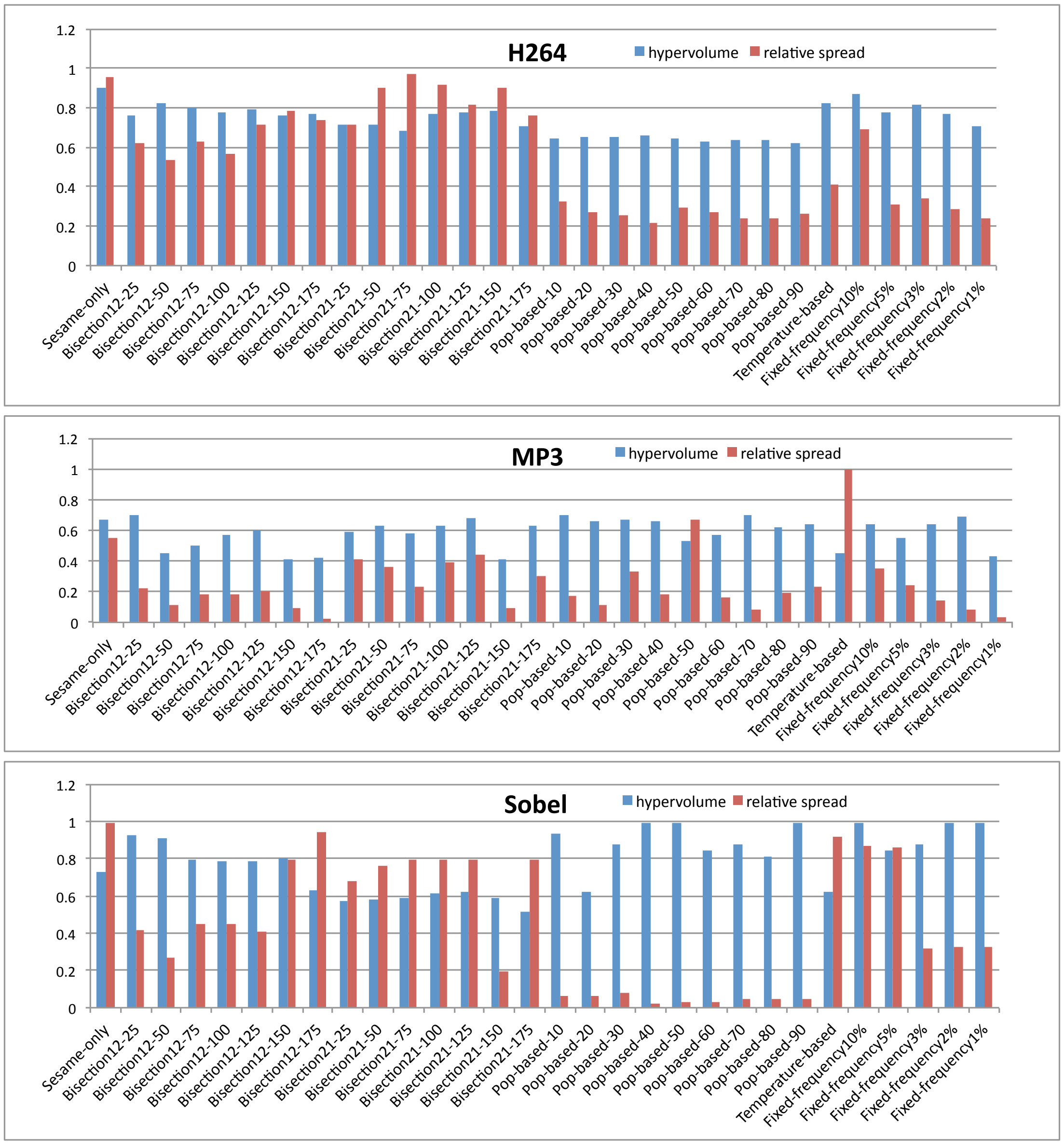

Fig. 6. Average hypervolume and ∇ values for the different DSE methods applied to the Mp3 decoder, H264 decoder, and Sobel filter applications.

savings could have also been used for performing more search iterations, thereby possibly improving the search results. We have not done this in this paper (we considered the number of search iterations to be fixed), but this is considered as future work.

V. Related Work

Current state-of-the-art in system-level DSE often deploys population-based Monte Carlo-like optimization algorithms like hill climbing, simulated annealing, ant colony optimization, or genetic algorithms. By adjusting the parameters, or by modifying the algorithm to include domain-specific knowledge, these algorithms can be customized for different DSE problems to increase the effectivity of the search [14], [2]. Another promising approach is based on meta-model assisted optimizations, which combines simple and approximate models with more expensive simulation techniques [11], [15], [4], [1], [10]. In [4], the authors use meta-models as a pre-selection criterion to exclude the less promising configurations from the exploration. In [10], meta-models are used to identify the best set of experiments to be performed to improve the accuracy of the model itself. In [11], an iterative DSE methodology is proposed exploiting the statistical properties of the design space to infer, by means of a correlation-based analytic model, the design points to be analyzed with low-level simulations. The knowledge of a few design points is used to predict the expected improvement of unknown configurations. However, these meta-models usually have design space parameters relative to the micro-architecture of design instances, while they

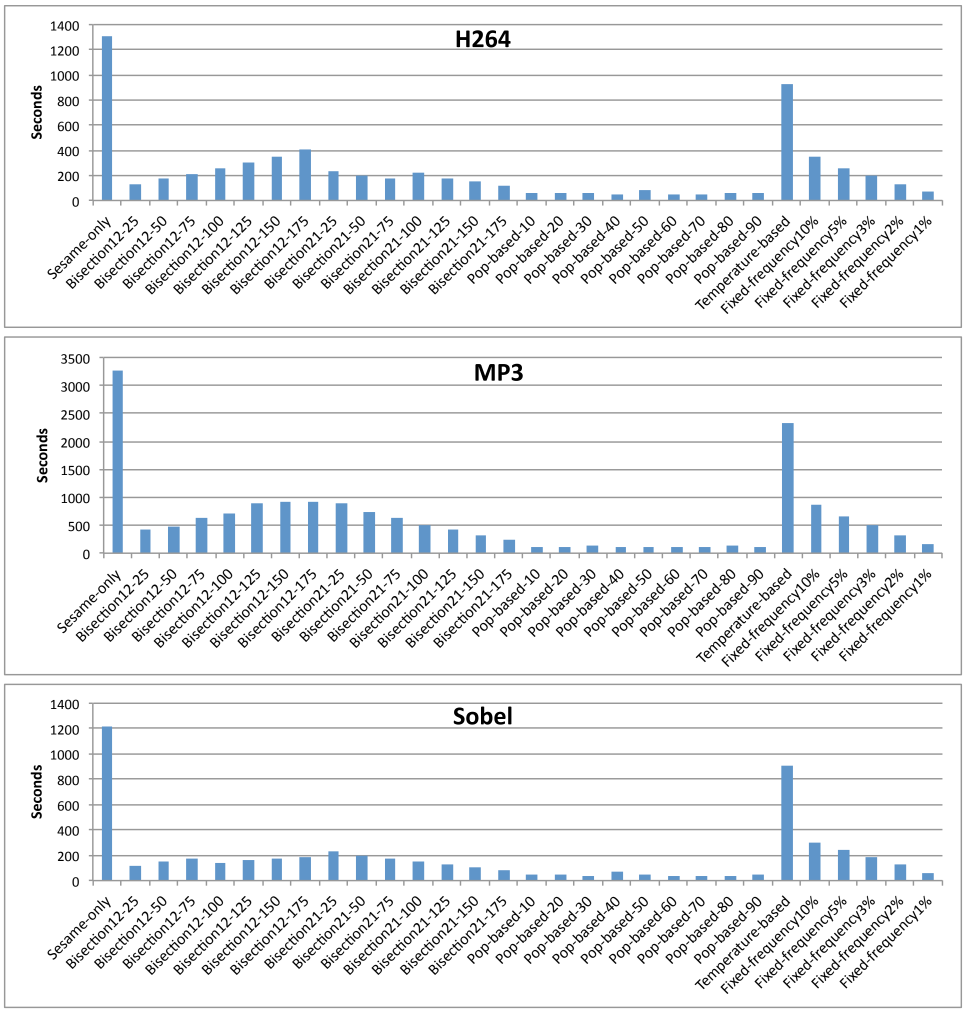

Fig. 7. Execution times for the different DSE methods.

do not address the problem of e.g. topological mapping of an application on the underlying MPSoC architecture. While micro-architecture parameters like cache size typically affect the system performance in a predictable, often linear, fashion, the resource binding of the application graph to the architectural platform presents a much less predictable performance.

A second class of design space pruning is based on hierarchical DSE (e.g., [7], [12], [9], [5]). In this approach, DSE is first performed using analytical or symbolic models to quickly find the interesting parts in the design space, after which simulation-based DSE is performed to more accurately search for the optimal design points. The main drawback of this method is that if the *first step* is not accurate enough, it may not produce the best set of design points to simulate. In our approach, the pruning and simulation phases are integrated to avoid this problem.

VI. Conclusion

In this paper, we have studied a pruning technique to reduce the simulation overhead in system-level MPSoC design space exploration (DSE). More specifically, we have proposed and examined different strategies for interleaving fast but less accurate analytical performance estimations with slower but more accurate simulations. Experimental results have demonstrated that such hybrid DSE is a promising technique that can yield solutions of similar quality as compared to simulation-based DSE but only at a fraction of the execution time.

Acknowledgment

This research has been supported by the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 248424, MADNESS Project.

References

[1] G. Ascia, V. Catania, A. G. Di Nuovo, M. Palesi, and D. Patti. Efficient design space exploration for application specific systems-on-a-chip. *J. Syst. Archit.*, 53:733–750, October 2007.

[2] V. Catania and M. Palesi. A multi-objective genetic approach to mapping problem on network-on-chip. *Journal of Universal Computer Science*, 12(4), 2006.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

[4] M. T. M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10:421–439, 2006.

[5] C. Erbas, S. Cerav-erbas, and A. D. Pimentel. Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design. *IEEE Transactions on Evolutionary Computation,vol.10,no.3*, 10:358–374, 2006.

[6] M. Gries. Methods for evaluating and covering the design space during early design development. *Integr. VLSI J.*, 38:131–183, December 2004.

[7] Z. J. Jia, A.D. Pimentel, M. Thompson, T. Bautista, and A. Núñez. Nasa: A generic infrastructure for system-level mp-soc design space exploration, Proceedings of the IEEE Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia) 2010.

[8] G. Kahn. The semantics of a simple language for parallel programming. In *Proc. of the IFIP Congress 74*, 1974.

[9] J. Kim and M. Orshansky. Towards formal probabilistic power-performance design space exploration. In *Proceedings of the 16th ACM Great Lakes symposium on VLSI*. ACM, 2006.

[10] J. Knowles. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1), 2006.

[11] G. Mariani, A. Brankovic, G. Palermo, J. Jovic, V. Zaccaria, and C. Silvano. A correlation-based design space exploration methodology for multi-processor systems-on-chip. In *Proceedings of the 47th Design Automation Conference (DAC)*. ACM, 2010.

[12] S. Mohanty, V. K. Prasanna, S. Neema, and J. Davis. Rapid design space exploration of heterogeneous embedded systems using symbolic search and multi-granular simulation. *SIGPLAN Not.*, 2002.

[13] H. Nikolov, M. Thompson, T. Stefanov, A. Pimentel, S. Polstra, R. Bose, C. Zissulescu, and E. Deprettere. Daedalus: Toward composable multimedia MPSoC design. In *Proc. of the 45th ACM/IEEE Int. Design Automation Conference (DAC '08)*, 2008.

[14] H. Orsila, E. Salminen, and T. D. Hämäläinen. Parameterizing simulated annealing for distributing kahn process networks on multiprocessor socs. In *Proc. of the Int. Conference on System-on-chip*, pages 19–26, 2009.

[15] G. Palermo, C. Silvano, and V. Zaccaria. Respir: a response surface-based pareto iterative refinement for application-specific design space exploration. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 28, 2009.

[16] A. D. Pimentel, C. Erbas, and C. Polstra. A systematic approach to exploring embedded system architectures at multiple abstraction levels. *IEEE Trans. Comput.*, 55(2):99–112, 2006.

[17] R. Piscitelli and A.D. Pimentel. Design space pruning through hybrid analysis in system-level design space exploration. In *Proceedings of the Int. Conference on Design, Automation, and Test in Europe (DATE'12)*, 2012.

[18] A. Sangiovanni-Vincentelli and G. Martin. Platform-based design and software design methodology for embedded systems. *IEEE Des. Test*, 18, 2001.

[19] T. Taghavi and A.D. Pimentel. Design metrics and visualization techniques for analyzing the performance of moeas in dse. In *Proc. of the 11th Int. Conference on Embedded Computer Systems: Architectures, MOdeling and Simulation (SAMOS '11)*, 2011.

[20] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In *Proc. of the 5th International Conference on Parallel Problem Solving from Nature*, 1998.