Tools and techniques for efficient system-level design space exploration
Thompson, M.

Citation for published version (APA):
Thompson, M. (2012). Tools and techniques for efficient system-level design space exploration

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 9

Conclusions

In the introduction of this thesis, we sketched some of the problems that currently play a major role in the design, development and implementation process of modern embedded systems. Such systems are used for more and more complex tasks, supported by increasing on-chip resource availability and implementation technologies. While the design of these systems becomes increasingly complex, there is also increasing demand and pressure to design such systems in shorter time frames. This in turn requires new design methodologies and tools to support the design process and offer automatic transitions from high level system specifications down to prototype system implementations. The research presented in this thesis was done in the context of the Daedalus design flow presented in Chapter 2. Daedalus is a good example of a next-generation design methodology supported by the necessary tools for each step in the design process. It takes the designer from a sequential application to a fully implemented FPGA prototype MPSoC system in a matter of hours. The three major tools in Daedalus take care of application parallelization, design space exploration and system implementation. The underlying methodology relies heavily on a few important methodology principles that favor the use of high-level system specifications, component-based design and component reuse, separation of concerns and a high degree of design process automation.

The contributions of this thesis focus on the design space exploration phase of the design process. This is a crucial phase in the overall design process, since the initial models that are created in the early stages of the design process can support the designer at the later stages, where the cost of recovering from wrong design decisions is typically higher due to the more detailed level of specification at that stage. In order to provide this kind of early feedback to the designer, the abstract system-level models have to meet a few criteria: in particular, they have to be sufficiently accurate, fast and relatively easy to construct. These criteria are often non-orthogonal and trade-offs have to be found in order to support an efficient and successful design process. In the introduction chapter (Section 1.6), we made a parallel between these trade-offs and the design criteria of the embedded MPSoC systems themselves. Indeed, we sketched the criteria for system-level modeling in a system of perpendicular axis (repeated here in Figure 9.1) as an analogy of the more common depiction of multi-dimensional parameter or objective spaces for the properties of the system under
study (an abstract representation of the latter is shown in Figure 1.2). As was discussed in the introduction, we identified specific criteria for evaluating single design points (indicated by accuracy, speed and effort) and for the traversal of the design space (indicated by convergence, confidence and effort). Although this list is perhaps not complete, we feel it captures the most pressing requirements to help to achieve the goal of successfully traversing the design process. In the following we will summarize the various topics addressed in this thesis according to these criteria and consider how they relate and contribute towards a design space exploration methodology that is ideally suited to support the process of modern embedded system design.

Figure 9.1: The two-part taxonomy of concerns for efficient DSE

9.1 Evaluation of a single design point

The ability to quickly and accurately model a design point are two of the basic requirements for any system-level modeling and simulation tool. Much work has already been done in this area in the past few decades. As we have shown, our Sesame tool (using non-refined model components) adheres to a higher level of modeling abstraction than most other tools. High abstraction levels can offer fast execution times for simulation, but at the same time makes it more challenging to maintain sufficient modeling accuracy. Sesame’s models also allow for quickly setting up models that are sufficiently flexible to model a wide range of systems. These topics were addressed in Chapter 3, 4, 5, and 6. Below, we summarize their specific contributions in a bit more detail.

Accuracy

Model accuracy refers to the ability of the simulation model to capture in a realistic way the properties of a system in one or more evaluation criteria in objective space: e.g., performance, power usage or cost. As was explained in detail in Chapter 3, a Sesame model associates latencies to application events using a pre-defined (application dependent) latency table. In Chapter 4 we proposed two techniques that allow a designer to calibrate the model
by deriving values for the latency table using an accurate, lower level Instruction Set Simulator (ISS). We proposed both an on-line and an off-line technique where the ISS is used to accurately measure latencies of the various application events. The off-line technique measures (average) latencies of application events at design time, which are then used in the model’s latency tables and can be reused for different architectural design options. Off-line calibrated models have fast execution times, but recalibration is needed when dynamic application behavior changes the number or type of trace instructions. On-line calibration combines Sesame’s high-level models with one or more instances of the ISS at runtime: measurements on event latencies are now done during model execution. The resulting co-simulation (Sesame application model + ISS) now measures a latency for each individual application event, instead of using an average value. This improves modeling accuracy in the architecture model and has the additional benefit that no recalibration is required for dynamic behavior in the application.

Moreover, the signature-based method for model calibration is presented (Section 4.6), which has the additional benefit of doing calibration in an application independent way. On the basis of a training set of benchmark applications, the performance of a specific type of processor is captured by a processor signature. On the basis of this processor signature, an application event (represented by an application signature) can then be assigned a latency value. Possible sources of inaccuracy for this method lie in the regression model for deriving the processor signature and the selection of training benchmarks. However, a design space exploration case study showed that the calibrated model performs good relative comparison of design points (although the absolute estimation was overestimated).

Finally, the case study in Chapter 8 showed that our simulation models are accurate and useful for design space exploration by validating our simulation results with measurements on actual prototype platforms as generated by the Daedalus design flow.

**Speed**

The *speed* concern refers to the performance of the evaluation mechanism of a design point. As various experiments have shown, Sesame’s abstract executable simulation models are fast: running a typical system-level simulation of a complete MPSoC system in a fraction of a second. We exploited this performance in Chapter 7 where we integrated the simulator directly as the fitness function of various automatic design space exploration methods based on genetic algorithms. Faster design point evaluation can benefit design space exploration, since a larger part of the design space can be explored in the same time. However, higher abstraction levels of modeling can reduce accuracy as was discussed before. What is interesting in this respect is that Sesame supports the designer in this trade-off by enabling gradual model refinement (Section 3.2.3). Moreover, the calibration techniques from Chapter 4 also support this trade-off. For example, compared to the on-line trace calibration technique, the off-line technique offers higher simulation speeds but possibly at the cost of lower accuracy. Performance of the on-line trace calibration method is constrained by the performance of multiple instances of the lower-level simulators, although we have shown that there are opportunities to mitigate this problem by parallelization. The signature-based
calibration technique resides in the middle of the on-line and off-line trace calibration methods trade-off and thus provides a viable alternative.

**Effort**

We consider design *effort* a highly important concern in the system design process as overall design time is sure to increase when large amounts of effort are needed to create a suitable model. Sesame’s high level of modeling abstraction is one of the main reasons why models are easy create: Sesame’s library of standard components is often sufficient to build an initial model in a few hours (which may subsequently be refined). Also contributing to effort reduction in Sesame (Chapter 3) is the separation of application and architecture models, the separation of model specification (in YML) and model behavior (component implementations), as well as its focus on modularity and component reuse. Moreover, effort is reduced by providing tool support as much as possible for routine modeling activities: an example of this is the automatically generated virtual layer, which also proved particularly useful for the stochastic multi-applications (Section 5.2) and reconfigurable models (Section 6). Additional tool support was provided for storing, querying and visualizing the results of design space explorations (6.2.3). The trace calibration and signature-based calibration methods can also be seen in light of effort-reduction, since they provide a standard way towards a more accurate or more refined (mixed-level) simulation model.

We explicitly include modeling scope of the tools in the *effort* concern, since a modeling tool that can not be used for wide range of different systems, will always result in an increased effort. This effort could consist of, for example, structural or superficial modification of the tool to make it fit the design problem, or an interoperability effort needed to make a combination of tools solve the single problem. To avoid such effort to become part of the design process, a modeling and simulation tool should ideally be capable to model a large variety of different types of systems. Extending the scope of Sesame’s modeling capabilities is one of the main contributions of this thesis. In particular, Chapter 5 extended Sesame with components and mechanisms for multi-application modeling and Chapter 6 for modeling partially dynamic reconfigurable systems. Together with multi-application modeling, we also introduced stochastic workload modeling (Section 5.2), which can provide feedback and testing capabilities even before the final target application workload has been fully defined (thus possibly reducing design times). Moreover, Sections 5.4.1 and 5.4.2 discussed some possibilities to capture the effects of inter and intra-task level dynamic behavior in a Sesame model. Chapter 6, provided modeling components and techniques to allow Sesame to model partially dynamic reconfigurable systems. A clear separation was made between the components and techniques that are generic and thus reusable in other types of reconfigurable systems, and those more geared towards the Molen platform used for the case-study.

Finally, we would like to emphasize that many of the modeling methods and techniques in Chapters 4,5 and 6 are generically applicable, even outside the context of Sesame.
9.2 Traversing the design space

In this section we look at the different criteria (according to the right part of the taxonomy represented in Figure 9.1): the efficient traversal of the design space. During the traversal, of course, we benefit from an optimal trade-off between speed, accuracy and effort for evaluating a single design point (the left part of the taxonomy). For example, the ability to quickly, easily and accurately evaluate a single design point helps to accurately evaluate a larger set of design candidates, thus increasing our chance to find optimal results. For smaller design problems this may indeed be sufficient, but for large design spaces, the total number of design points that can be feasibly evaluated may still be a very small fraction of the entire design space. Therefore, even if the speed, accuracy and effort trade-off is ideal, we still have to make sure that each evaluation of a design point contributes as much as possible to the an optimal traversal of the design space. A crucial component towards this goal is the search algorithm that navigates the design space towards areas of interest by proposing which design points to evaluate next. Regardless of the specific type of search method that is used for traversal, we proposed in Chapter 1 that its success (measured as its contribution towards the ideal DSE scenario) depends on three major concerns: confidence, convergence and effort. We note that once again these concerns typically can not be considered in isolation, since they are highly interdependent, contradictory and sometimes overlapping. The state-of-the-art in design space exploration research can be summarized as finding a good trade-off between these concerns. We will shortly discuss each of these concerns below and show how they relate to this thesis (mainly Chapter 7).

Confidence

The confidence concern refers to a measure of how certain we are that the results from the design space traversal are indeed the optimal results that we are looking for. For example, if the traversal consists of evaluating every design point (exhaustive search), then we can claim with 100% confidence that we can find the optimal result. However, when exhaustive search is not possible, then we typically have to resort to heuristic search algorithms that can not guarantee finding an optimum (but return a best-effort result instead). In some cases we can relate confidence to coverage: the fraction of the design space that was evaluated. This was for example the case with the uniform random search that was used as a baseline comparison in the PQ-plots introduced in Section 7.4. However, more advanced search algorithms (such as the various GA variants in Chapter 7), not only showed much better performance than random search, but also clearly showed that higher coverage did not necessarily improve the results (Section 7.7).

There are no generic and absolute metrics available to quantify confidence, rather we postulate that it will remain (for the foreseeable future) one of the intangible factors that constitute the “the art of system design”. In practice, a designer will typically be satisfied when he can use (with minimal effort, see below) a search algorithm that has previously shown good results for similar design problems. Confidence is then replaced by an approximate, experience-based assumption that the algorithm will perform well in general.
However, complete surveys that match search algorithms to specific categories of design problems have not been established in the system-design field. Moreover, literature shows that improvement of search algorithms is always possible for specific problems, thus reducing confidence that a general solution can provide the best results. For example, it turns out that simple applications of domain-specific knowledge can quite easily improve search performance. An example in this thesis were the distance-based GA extensions which improved upon the standard GA (Section 7.8).

Finally, a note on some other facets of confidence that are more concrete, but on a smaller scale. Firstly, it is generally preferable that a particular search algorithm is stable: repetitions of the algorithm give results within a small, consistent interval (we checked this using confidence intervals in Chapter 7). In this way, the designer can be assured that even a single design space traversal with the algorithm gives the best possible result, thus shortening exploration time. Secondly, confidence can be improved if a search algorithm has few parameters that influence its performance, since these would have to be tried by the designer. In our case studies we saw that this was quite a problem with GAs, but we also saw that parameter sensitivity seemed to be reduced with the proposed GA extensions. More research in this area is required. Finally, confidence improves by reducing as much as possible any discrepancy between the evaluation mechanism (e.g. simulator) and the real system. Within the Daedalus design flow, there are sufficient calibration and validation opportunities, as we have shown in Chapter 8. Equally relevant is the model-to-model validation in Chapter 4 where we showed that signature-based models show largely the same relative performance trend (Figure 4.12) required for design space exploration.

Convergence

Convergence refers to the desirable property of a design space traversal method to locate the optimum with as few iterations or evaluations as possible. It is an analogous concern to speed in the left side of the taxonomy, but now referring to the performance of the entire design space traversal. We note that convergence needs to be optimized as a trade-off between two extremes. Convergence that is too low indicates that the search algorithm is unable to extract the relevant information from the evaluated design points that could guide the search to an optimum. Too high convergence typically occurs when the search algorithm finds such information, but it leads to a local, instead of a global optimum. In both cases no good quality results can be found and therefore, finding the right balance for convergence is critical. Depending on the specific search algorithm, there may be parameters that stimulate a particular convergent behavior. For example, we experimented with the selective pressure parameter in Chapter 2 and attempted to control convergence by means of population diversity in Section 7.7. Convergence captures one of the essential components of design space exploration mechanisms and more research in the DSE area will be needed to understand it better.
9.2. TRAVERSING THE DESIGN SPACE

Effort

This concern refers to the engineering effort that goes into a design space exploration case study. We note that it is similar to the effort concern as was described previously for the evaluation of a single design point. Indeed, the motivation for this concern is the same: it is one of the main time consuming activities in the entire DSE process. Yet, even if we consider the effort concern solved for a single design point, then there are other, related concerns specific to the process of design space traversal. Although this was not one of the main topics in this thesis, we did frequently touch upon it, so we provide some conclusions here.

Steps in the design space exploration phase that typically require (manual) effort include: 1) the definition of a design space exploration experiment (e.g., parameter space boundaries, and search algorithm parameters), 2) automatic transformation of design points to simulator input specifications, and 3) methods to run and evaluate the results of the DSE. Points (1) and (2) were shortly discussed in Section 3.4 and (3) in 3.5. We identified two methods that are frequently used: the meta-platform specification and the generator-approach. We use the former extensively in Chapter x where we also propose a solution to the negative effects of an evolutionary algorithm that is inherent in the the symmetry of the mapping-based design point specification.

Most importantly, we emphasize that much of the engineering effort could (theoretically) be non-recurring. Instead, we see that creation of the tools required by (1), (2) and (3) is typically repeated for specific tools or even users, although this burden could be shared by the whole research community (thus shortening design cycles even further). This would, however, require a large amount of coordination work to firstly standardize specifications and methods, for example to define interchangeable, high-level specifications of a design space (or design problem) and secondly, provide a framework in which exploration methods and tools can be compared on a set of standardized benchmarks. Such a framework could be based on a software design where tool-specific back-ends are provided as plug-ins, while the core functionality is shared. Research and engineering in this particular subdomain of (embedded) system design is however still in its infancy and the required coordination for shared frameworks and benchmarks will remain future work.

On the topic of (3) we have seen in Chapters 2 and 3 that within the context of Daedalus, the results of a design space exploration can be stored in an interchangeable format (XML), as well as in a database. The database allows for evaluation of a previously evaluated design space by means of queries expressed in a standardized query language, while the XML format promotes tool interoperability. The latter has been exploited by the various design space visualization tools that have been developed for Sesame (see Section 3.5).