Design Space Exploration at the Electronic System Level[⋆]

Luise Müller[0000−0002−3924−5852] and Christian Haubelt[0000−0002−1568−5423]

Applied Microelectronics and Computer Engineering, University of Rostock, Germany
{luise.mueller,christian.haubelt}@uni-rostock.de

1 Introduction

The continuous advance of application-specific computer systems has made technology more accessible and affordable to various domains, industries and consumers. Simultaneously, the demands for the system's performance, reliability and energy consumption have increased drastically. Regarding the complexity of individual systems, including the number of internal components, processes and their heterogeneity, a vast number of design decisions have to be made in the development process. Therefore, an efficient Design Space Exploration (DSE; [7]) is essential to enable a product designer to identify valid system implementations as well as the most favorable design alternatives. Our project aims at improving the DSE based on Answer Set Programming (ASP) and investigating ways to handle the enormous complexity of real-world examples.

2 Design Space Exploration

In the context of design space exploration for application-specific computer systems, we aim at finding feasible and preferably optimal solutions to the system synthesis problem, i.e., we look for the best transformation of a specification of a computer system into its implementation. The specification consists of three main aspects: the application, which contains a graph-based high-level description of the system's behavior; the architecture template, defined as a heterogeneous hardware platform organized as a mesh-based network on chip at the electronic system level; and mapping options indicating how the application graph could be mapped to the hardware template. The implementation represents the structure and characteristics of the final system, including decisions on the following synthesis steps: *allocation, binding, routing* and *scheduling*. The binding selects for each task of the application one processing unit from the hardware template for execution, while the routing ensures that messages from communicating tasks are correctly delivered through the network. Accordingly, the used hardware resources are allocated. Finally, the scheduling assigns start times to all tasks and communications so that there are no conflicts during the execution of the applications.

[⋆] Supported by the German Science Foundation (DFG) under grants HA 4463/4-2.

By assigning worst-case execution times to tasks and energy consumption and costs to resources, we can optimize three objectives, namely latency, energy consumption and hardware cost. The quality of each solution is evaluated via a Pareto preference, i.e., a design point is better if it is at least as good in all criteria and strictly better in at least one when compared to other design points.

A compact representation of the vast design space is achieved by a symbolic encoding of the system synthesis problem. That way, the design space exploration is reduced to computing stable models, i.e., *answer sets*. Within, each variable assignment, meeting all specified system requirements and hardware constraints, represents a feasible implementation.

Our framework is based on ASP. ASP supports expressing reachability, which allows the efficient encoding of multi-hop message routing in densely connected networks [2]. Further, ASP is tailored to knowledge representation and reasoning, but since it is inefficient for numerical problems, we rely on an extension to the ASP solver *clingo*, namely *clingo-dl* [4], which supports integer difference constraints. Validity checks regarding the schedulability as well as the Pareto optimization are tightly integrated into clingo's theory interface via background theories [6]. Hence, we can handle non-linear objectives and also check partial solutions to identify infeasible or dominated regions of the design space early.

3 Challenges

Despite enhancements in the solving process, deciding on a valid implementation of a system remains a NP-complete problem. Among other things, problem complexity scales with the size of the problem instances, i.e., with the number of tasks, communications and hardware resources. Especially real-world examples can be extremely complex, and thus, the vast design space cannot be exhaustively explored in a reasonable time. In this case, when we stop the search after a certain timeout, we cannot provide any information whether the optimal solution has been discovered nor how much of the design space remains unexplored. We see the need for a search space coverage tool for the ASP solver, e.g., following the approaches of Aloul et al. [1] or of Schuurmans and Southey [8].

An approach to handling the enormous complexity is to exploit the available ASP methods by, e.g., restricting the size of the feasible search space or guiding the search towards desired solutions. This usually requires domain-specific knowledge of the problem definition and the instances. Considering industrial product design strategies, we have investigated the use case of *evolutionary product design* [3, 5]. Given a launched product, the goal is to guide the development process of a derived product based on the knowledge of equal and unequal design decisions. Thus, we offer a formalized definition of the similarity information and employ strategies, domain-specific heuristics or preferences. Adapting the DSE, we can find good solutions, that are close to the previous product version, early.

Nevertheless, to be able to handle the complexity of real-world examples, we plan to further investigate efficient solution approaches based on iterative or evolutionary search organization, symmetry detection and task graph clustering.

References

1. Aloul, F., Sierawski, B., Sakallah, K.: Satometer: how much have we searched? In: Proceedings 2002 Design Automation Conference (IEEE Cat. No.02CH37324). pp. 737–742 (2002). https://doi.org/10.1109/DAC.2002.1012720
2. Andres, B., Gebser, M., Schaub, T., Haubelt, C., Reimann, F., Glaß, M.: Symbolic System Synthesis Using Answer Set Programming. In: Cabalar, P., Son, T.C. (eds.) Logic Programming and Nonmonotonic Reasoning. pp. 79–91. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
3. Haubelt, C., Müller, L., Neubauer, K., Schaub, T., Wanko, P.: Evolutionary system design with answer set programming. Algorithms **16**(4) (2023). https://doi.org/10.3390/a16040179
4. Janhunen, T., Kaminski, R., Ostrowski, M., Schellhorn, S., Wanko, P., Schaub, T.: Clingo goes linear constraints over reals and integers. Theory and Practice of Logic Programming **17**(5-6), 872–888 (2017). https://doi.org/10.1017/S1471068417000242
5. Müller, L., Neubauer, K., Haubelt, C.: Exploiting similarity in evolutionary product design for improved design space exploration. In: Orailoglu, A., Jung, M., Reichenbach, M. (eds.) Embedded Computer Systems: Architectures, Modeling, and Simulation. vol. 13227, pp. 33–49. Springer International Publishing, Cham (2022)
6. Neubauer, K., Wanko, P., Schaub, T., Haubelt, C.: Enhancing symbolic system synthesis through ASPmT with partial assignment evaluation. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017. pp. 306–309. IEEE, Lausanne, Switzerland (Mar 2017). https://doi.org/10.23919/DATE.2017.7927005
7. Pimentel, A.D.: Exploring exploration: A tutorial introduction to embedded systems design space exploration. IEEE Design & Test **34**(1), 77–90 (Feb 2017). https://doi.org/10.1109/MDAT.2016.2626445
8. Schuurmans, D., Southey, F.: Local search characteristics of incomplete sat procedures. Artificial Intelligence **132**(2), 121–150 (2001). https://doi.org/https://doi.org/10.1016/S0004-3702(01)00151-5