Exact Multi-Objective Design Space Exploration using ASPmT

Kai Neubauer*, Philipp Wanko†, Torsten Schaub†, and Christian Haubelt*
*University of Rostock, Applied Microelectronics and Computer Engineering, Germany
†University of Potsdam, Knowledge Processing and Information Systems, Germany
{kai.neubauer,christian.haubelt}@uni-rostock.de, {wanko, torsten}@cs.uni-potsdam.de

***Abstract*—An efficient *Design Space Exploration (DSE)* is imperative for the design of modern, highly complex embedded systems in order to steer the development towards optimal design points. The early evaluation of design decisions at system-level abstraction layer helps to find promising regions for subsequent development steps in lower abstraction levels by diminishing the complexity of the search problem. In recent works, symbolic techniques, especially Answer Set Programming (ASP) modulo Theories (ASPmT), have been shown to find feasible solutions of highly complex system-level synthesis problems with non-linear constraints very efficiently. In this paper, we present a novel approach to a holistic system-level DSE based on ASPmT. To this end, we include additional background theories that concurrently guarantee compliance with hard constraints and perform the simultaneous optimization of several design objectives. We implement and compare our approach with a state-of-the-art preference handling framework for ASP. Experimental results indicate that our proposed method produces better solutions with respect to both diversity and convergence to the true Pareto front.**

I. Introduction

In order to cope with the ever-increasing complexity of embedded systems, system-level descriptions are utilized to diminish the complexity of finding potentially good solutions which can then be used as initial starting points for further optimization in lower abstraction levels. At system level, applications are composed of communicating tasks while the hardware architecture contains heterogeneous processing elements (e.g. CPU, DSP, GPU) as well as a communication infrastructure like routers and links.

Depending on the decisions that have been made, the qualitative properties (e.g. latency, energy consumption, area requirements) of the resulting system implementation may vary considerably from solution to solution resulting into a multi-objective optimization problem (MOOP). Thus, a Design Space Exploration (DSE) is imperative to find solutions with optimal properties.

Essentially, DSE approaches can be characterized into two types [1]: First, (meta-)heuristics like evolutionary algorithms and ant colony optimization (e.g. [2], [3]) and second, exact methods such as Integer Linear Programming (ILP) and branch-and-bound algorithms (e.g. [4], [5]).

Most of the works presented in the field of meta-heuristics extend basic techniques in order to respect domain specific characteristics. For example, in [2], the authors extend genetic algorithms by utilizing domain knowledge. They state, that small differences in design decisions lead to similar system implementations and that symmetrical design points can be pruned.

Another approach (e.g. [6], [7]) of handling the infeasibility problem is to integrate dedicated constraint solvers into a multi-objective evolutionary algorithm (MOEA). The work of Schlichter et al. [7] integrates, for example, a Boolean Satisfiability (SAT) solver into a MOEA. Here, the decisions are not directly controlled by the randomized search algorithm of the MOEA but the heuristic of the decision variables is subject to exploration. This way, solutions are guaranteed to be feasible.

Finally, fully exact methods have been developed to explore the design space systematically. While meta-heuristics normally only cover a limited portion of the design space, exact methods are guaranteed to find the optimal solutions. Nevertheless, for a long time those methods were restricted to single-objective optimization problems only. As one of the few exceptions, Lukasiewycz et al. [4] present a complete multi-objective Pseudo-Boolean solver based on branch-and-bound algorithms. The results show that this technique is able to find the proven optimal solutions for small problems in a short time. However, exact methods are often replaced in favor of heuristic approaches as the complexity of large systems hinders reasonable employment of those techniques.

The disadvantage of using meta-heuristics, on the other hand, is that the initial population is created by a randomized process. Finding feasible regions becomes therefore a problem for stringently constraint environments. Moreover, because the search is generally not executed systematically but based on combining previously found solutions, MOEAs tend to run into saturation and stop finding novel solutions after a number of iterations.

As a remedy, by encoding the problem symbolically, recent advances of constraint solving technologies can be utilized to cope with the complexity of finding feasible solutions. Especially, *Answer Set Programming (ASP)* has been shown to deal with such stringently constrained design problems very efficiently (e.g. [8]). Opposed to other symbolic techniques like SAT, reachability can be expressed naturally in ASP which fastens the communication synthesis. However, one problem is that non-linear constraints cannot be easily expressed within ASP.

In the paper at hand, we therefore propose an approach that utilizes an exact symbolic encoding for both constraint solving and design space exploration. To address the shortcomings of ASP, we present specific background theory solvers to handle *non-linear objectives* as well as Pareto filtering of found solutions. By utilizing the state-of-the-art ASP solver clingo 5 [9], these background theories can be tightly integrated into the solving process (*ASP modulo Theories (ASPmT)*). This way, we are able to utilize conflict clauses on partial solutions to prune the search space from infeasible and dominated regions of design points early in the decision process.

Note that our methodology uses *exact* search strategies with "*any-time*" characteristic, i.e., canceling the search at any time returns an approximate Pareto set that strictly improves with increased solving time until the true Pareto front is reached.

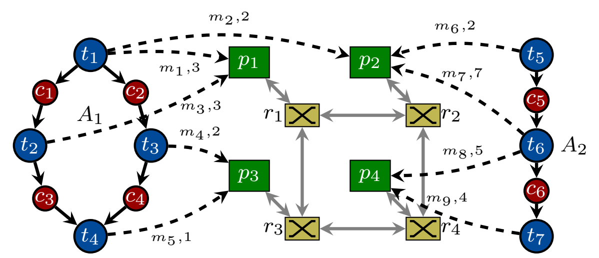

Figure 1. Specification example consisting of two applications A_1 and A_2, a 2×2 platform template with four processing elements p_{1-4} and routers r_{1-4}, and mapping options m_1 to m_9 annotated with WCET values.

II. Prerequisites

The focus of the work at hand is on how to efficiently explore the design space of embedded systems using ASPmT. To this end, the description of the specification model considered in the remainder of the paper is followed by the basics of Answer Set Programming (ASP) and the utilization of background theories.

A. Specification Model

As depicted in Fig. 1, we model the system specification as a graph separated into applications, an heterogeneous architecture template, and mapping options that connect the former two.

Application: An application is specified in task-level granularity and is modeled as a directed acyclic graph $A = (T, C, E)$. That is, it contains sets of computational tasks T and communication messages C. A set of edges $E \subseteq T \times C \cup C \times T$ specifies dependency relations between the elements. Each message $c \in C$ has exactly one predecessor task, i.e. inter process communication is characterized in a point-to-point fashion. Furthermore, each application is constrained by a maximum period by which it repeats its execution.

Architecture Template: The architecture, or *platform* template $P = (V_P, V_R, L)$ consists of processing elements V_P and the communication infrastructure split into routers V_R and links L. Both processing elements and routers are annotated by individual area and static power requirements that are used in the evaluation process to determine the quality of a solution. Additionally, the routing delay and energy determine the time and energy needed for each message to get send over one link. In this paper, we assume a circuit switching strategy for the routing of messages. That is, a message blocks the whole route from sender to receiver until it has been received completely. Note that bidirectional arrows represent two separate links.

Problem Instance: For each task, a set of mapping options $M \subseteq T \times V_P$ is specified. A mapping option $m = (t, p)$ indicates that task t may be executed on processing element p and is annotated with a worst case execution time (WCET) as well as the dynamic energy consumed by p when executing t. Specifying several mapping options per tasks with different WCETs and energy annotations corresponds to the modeling of heterogeneous systems. Together with the applications and the platform template, the mapping options complete the problem instance $I = (A, P, M)$.

Exploration Model: Acquiring a *feasible* solution to the problem instance involves selecting a mapping for each task, routing the messages over the communication infrastructure, and determining a schedule while adhering to given timing constraints. Each solution is evaluated by three objective functions *latency*, *area*, and *energy*, that determine timing properties as well as area and energy requirements, respectively. These objective functions represent soft constraints that have to be optimized. Without loss of generality, the DSE is formulated as a minimization problem as follows:

$$\text{minimize } f(x) = (latency(x), area(x), energy(x)),$$
$$\text{subject to:}$$
$$x \text{ is a feasible system implementation.}$$

Here, a feasible system implementation is a solution that adheres to all given mapping, routing, and timing constraints.

In this work, we consider three different routing strategies. First, *dimension order routing* (DOR) only allows one route for each pair of sending and receiving processing elements but simultaneously guarantees the shortest path. Second, *shortest path routing* (SPR) also guarantees a shortest path between sender and receiver. However, the route is not fixed and various alternative routes can be selected. Finally, *arbitrary length routing* (ALR) allows every acyclic route over the communication infrastructure and may be able to find solutions that distribute communication traffic over less congested links. That is, the number of decision variables and thus the complexity increases from DOR to ALR.

B. Answer Set Programming

The specification model and the constraints described in the previous section are encoded as Answer Set Programming (ASP) facts and rules. ASP is a programming paradigm that is tailored towards NP-hard search problems and is based on the stable model (*answer set*) semantics. Problems are formulated in a first-order input language as a set of facts and rules that are used to represent and infer domain knowledge, respectively.

Determining answer sets of logic programs (i.e., the combination of facts and rules) in ASP is a two-step process. First, the logic program is translated (*grounded*) into a variable free representation before it can be solved by an answer set solver that determines stable models (*solutions*). The actual solving process is out of scope of this paper and we refer to [10] for further information.

Generally, ASP is capable of finding stable models efficiently if there are only linear constraints involved. However, in system design, non-linear constraints such as timing requirements must be taken into account. To this end, we use ASP modulo Theories (ASPmT) (see [9]) to incorporate custom background theories directly into the solving process. The details are displayed in the following section.

III. Optimization Framework

In this section, we present our novel system-level Design Space Exploration (DSE) framework for finding Pareto optimal solutions. It utilizes the ASPmT paradigm to evaluate non-linear objectives and performs dominance checks of feasible design points in individual background theories. In the following, an overview of our framework is given before various exact optimization strategies are presented.

A. Framework Overview

The general overview of our proposed DSE is depicted in Fig. 2 and essentially consists of three pillars - the ASP solver *clingo*, a theory and an optimization propagator. A problem instance I as described in Sec. II in conjunction with a uniform problem definition encoded as ASP program serve as input for

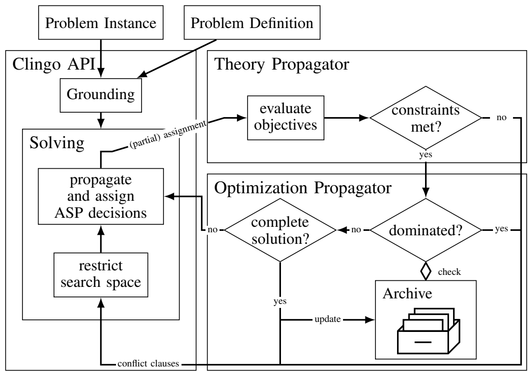

Figure 2. Architecture of the ASPmT optimization framework

our framework. In a first step, the program is grounded into a variable free representation that can be processed by the ASP solver module of *clingo*. During solving, *clingo* propagates and assigns ASP variables. In each solving step, routing and mapping decisions are made by *clingo* before the (possibly partial) assignment is relayed to the background theory propagator that evaluates the current decisions w.r.t. the objectives *latency*, *area*, as well as *energy* and checks if given timing constraints are met. Subsequently, the current solution is forwarded to the optimization propagator that checks if it is non-dominated w.r.t. already found solution stored in an archive.[1] Both constraint and dominance checking steps directly return conflict clauses to the ASP solver whenever the results are negative. Here, they are utilized to restrict the search space that results in backtracking of already performed decisions. Note that this procedure is applied to partial assignments, i.e., incomplete solutions where not all decisions have been made, to prune infeasible and dominated regions early from the search.

While partial assignments that pass constraint and dominance checks are handed back to *clingo*, complete solutions trigger an archive update, i.e., adding the current and removing dominated design points. Furthermore, a conflict clause is propagated to the ASP solver to avoid visiting the same design point multiple times.

Note that the system synthesis problem, i.e., acquiring a feasible binding, routing, and schedule, has to be solved for every design point considered in the DSE. To this end, we utilize an ASPmT approach as presented in [11] that makes mapping and routing decisions in ASP and leverages qunatifier-free integer difference logic (QF–IDL) as background theory to guarantee compliance to timing constraints.

B. Optimization Strategies

To the best of our knowledge, this work presents for the first time a system-level DSE executed in a background theory of a symbolic constraint solver. However, with *asprin* [12], a general framework for preference handling has been developed that supports multi-objective optimization of linear objectives. Therefore, our first DSE strategy is similar to *asprin* but with the extension of additional background theory propagators in order to support non-linear constraint solving. As it is based on the combination of *asprin* with background theories, we call it the *hybrid* approach in the following. Here, linear objectives are calculated and the dominance checks are performed directly by the solver while background theories are only utilized to evaluate design points w.r.t. non-linear objectives. This strategy adds constraints to the problem definition ensuring an improvement of a solution compared to the previous one or the incomparability to already obtained Pareto optimal solutions. As a consequence, only after a design point is proven to be located on the true Pareto front, incomparable design points can be found. Hence, the convergence of the approximation set towards the Pareto front contains a *depth-first* characteristic. As is common in ASP, previous stable models are partly saved via facts within the logic program rendering an explicit archive in a background theory obsolete.

In contrast to the first *hybrid* strategy, our second one utilizes background theory propagators for both evaluation and dominance checks. Similar to *hybrid*, it follows a *depth-first* characteristic. Consequently, this strategy is called $\textit{theory}_{\textit{depth}}$ in the following. The archive of the optimization propagator saves all found Pareto optima as well as the current design point that is guaranteed to be incomparable to them. Hence, the current approximation set can be obtained at any time.

Finally, $\textit{theory}_{\textit{breadth}}$ again makes use of the dominance check in the Pareto propagator and thus, contains the current approximation set. In contrast to $\textit{theory}_{\textit{depth}}$, design points are not strictly required to dominate the previously found solution but rather the archive is also updated whenever an incomparable design point w.r.t. the whole approximation set is found. The possible advantage of this strategy is two fold. First, diverse design points may be obtained more frequently as novel non-dominated solutions are added independent of the previously found design point. Second, as a consequence, the dominance check step has more information on dominated regions of the design space which allows for a more efficient pruning in early steps of the decision process. However, the convergence of the approximation set towards the true Pareto front may be slower as the solving is not primarily directed to prove Pareto optimality of found solutions.[2]

IV. Experiments

In this section, we evaluate our proposed framework with a number of test instances. The focus is on the different search strategies presented in the previous section, namely *hybrid*, $\textit{theory}_{\text{depth}}$, and $\textit{theory}_{\text{breadth}}$.

A. Experimental Setup

We randomly generated 30 test instances that are composed of series-parallel applications and a heterogeneous platform template organized in a regular grid. Each test instance consists of one to four applications A that are comprised of $|S|$ series and $|P|$ parallel patterns, resulting into a number of $|A| + 2 \cdot |S| + 3 \cdot |P|$ tasks and $2 \cdot |S| + 4 \cdot |P|$ messages per instance. Depending on the number of applications, the platform template size is accordingly adjusted. For all test instances with up to two applications, a grid size of $3 \times 3 \times 1$ is chosen while for instances with three and four applications grid sizes of $3 \times 3 \times 2$ and $3 \times 3 \times 3$ are chosen, respectively. In this work, we consider test instance with up to 166 task and 200 messages.

The design space exploration is conducted by our proposed framework with every possible combination of the presented search (i.e. *hybrid*, $\textit{theory}_{\text{depth}}$, and $\textit{theory}_{\text{breadth}}$) and routing (i.e. *arbitrary length routing* (ALR), *shortest path routing* (SPR), and *dimension order routing* (DOR)) strategies. That is, for each test instance, nine individual optimization runs have been

[1] Partial solutions can be used for dominance checks iff the problem is assignment monotonous, i.e., an additional decision must not improve the evaluation.

[2] As all the presented methods are exact, the Pareto optimality is proven inherently after all possible decisions have been made.

Table I. QUALITY FOR SOME TEST INSTANCES ACHIEVED BY THE DIFFERENT CONFIGURATIONS

$\lvert A \rvert$	$\lvert S \rvert$	$\lvert P \rvert$	Platform	Indicator	*hybrid*			*theory*$_{\text{breadth}}$			*theory*$_{\text{depth}}$		
					ALR	SPR	DOR	ALR	SPR	DOR	ALR	SPR	DOR
1	10	10	$3\times3\times1$	ε-Dominance	inf	1.248	1.288	inf	1.153	**1.085**	inf	1.437	1.391
				Entropy	0.000	2.472	2.473	0.000	**3.528**	3.514	0.000	2.473	2.471
2	13	16	$3\times3\times1$	ε-Dominance	1.602	1.578	1.356	1.271	**1.153**	1.196	1.662	1.425	1.779
				Entropy	2.467	2.473	2.473	3.130	**3.586**	3.415	2.473	2.473	2.466
3	22	27	$3\times3\times2$	ε-Dominance	1.855	1.650	1.477	1.835	**1.279**	1.309	1.855	1.637	1.594
				Entropy	2.473	2.463	2.473	2.501	**3.166**	2.572	2.473	2.473	2.473
4	21	20	$3\times3\times3$	ε-Dominance	1.610	1.614	**1.000**	1.510	1.301	1.151	1.615	1.579	1.435
				Entropy	2.473	2.471	2.473	2.743	3.043	**3.110**	2.473	2.472	2.473
4	24	38	$3\times3\times3$	ε-Dominance	inf	1.892	1.533	inf	1.515	**1.103**	1.895	1.894	1.837
				Entropy	0.000	2.472	2.472	0.000	2.915	**2.976**	2.473	2.473	2.473

conducted. To evaluate the performance of the variant strategies, we calculate the quality of found non-dominated solutions w.r.t. convergence and diversity. To this end, we utilize the *ε-Dominance* indicator from [13] and the *Entropy method* as presented in [14]. The ε-Dominance indicator determines how close a solution is situated to the Pareto front. A lower ε-Dominance value corresponds to a better convergence towards the reference front with a minimum value of one. The latter quality indicator evaluates the diversity of a Pareto front approximation. An approximation set with a high Entropy has a high diversity as well.

All optimization runs were configured to use eight threads and have been executed on an Intel Core i7-4770 with 32 GiB RAM running Ubuntu 14.04. The timeout has been set to 1800 s.

B. Results

For sake of brevity, the results of the optimization runs presented in Tab. I only constitute a representative part of all optimization runs. In each line, the value in the upper row represents the ε-Dominance while the value in the lower row indicates the entropy.

Regarding the diversity of the found non-dominated solutions, the search strategy *theory*$_{\text{breadth}}$ that explores the search space on a wider range outperforms both strategies *hybrid* and *theory*$_{\text{depth}}$. Due to the search strategy of the latter two, design points that are incomparable to previously found solutions are not considered until an optimal one on the front is found. In combination with the vast decision space, those two strategies were unable to find even one proven Pareto optimal solution (except for the first test instance, where *hybrid* in combination with DOR found two Pareto optimal solutions). Hence, *hybrid* and *theory*$_{\text{depth}}$ stick to an Entropy value of around 2.47.

With regard to the routing strategy, DOR and SPR achieve more diverse solutions than ALR. As expected, the strictly bounded search space of the former two enables faster exploration of larger regions in the design space.

As the *hybrid* strategy is adjusted towards good convergence, it performs better in this category. Here, it even finds solutions for two test instances that are not dominated by any other solution of another optimization run. However, *theory*$_{\text{breadth}}$ still outperforms the other approaches in most of the test instances.

The overall results are two fold. First, the achieved quality indicates that a search strategy which does not discard found incomparable design points explores the design space more efficiently. Second, a complex communication model hinders the exploration performance significantly. That is, ALR did not find any solution for 11 test instances and could exploit the advantages of more routing options only once.

V. CONCLUSION

In this paper, we have presented a design space exploration framework that extends the preference handling capabilities of the Answer Set Programming (ASP) solver *clingo* towards multi-objective optimization of non-linear constraints. To this end, we utilized background theory solving to execute the search for novel solutions as well as the constraint solving of non-linear constraints. We presented and implemented three different search strategies to perform the design space exploration. The experimental results indicate that a combination of a simple exploration model and a concurrent search of multiple regions in the objective space delivers the best results with respect to convergence and diversity of the found non-dominated solutions.

ACKNOWLEDGMENT

This work was funded by the German Science Foundation (DFG) under grants HA 4463/4-1 and SCHA 550/11-1.

REFERENCES

[1] A. D. Pimentel. Exploring exploration: A tutorial introduction to embedded systems design space exploration. *IEEE Design & Test*, 34(1):77–90, 2017.

[2] M. Thompson and A. D. Pimentel. Exploiting domain knowledge in system-level mpsoc design space exploration. *Journal of Systems Architecture*, 59(7):351 – 360, 2013.

[3] F. Ferrandi et al. Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(6):911–924, 2010.

[4] M. Lukasiewycz et al. Efficient symbolic multi-objective design space exploration. In *Proc. of ASP-DAC*, pages 691–696, 2008.

[5] N. Khalilzad, K. Rosvall, and I. Sander. A modular design space exploration framework for multiprocessor real-time systems. In *In Proc. of FDL*, pages 1–7, 2016.

[6] K. Neubauer, C. Haubelt, and M. Glaß. Supporting composition in symbolic system synthesis. In *Proc. of SAMOS*, pages 132–139, 2016.

[7] T. Schlichter et al. Improving system level design space exploration by incorporating SAT-solvers into multi-objective evolutionary algorithms. In *Proc. of ISVLSI*, 2006.

[8] B. Andres et al. Symbolic system synthesis using answer set programming. In *Proc. of LPNMR*, pages 79–91, 2013.

[9] M. Gebser et al. Theory solving made easy with clingo 5. In *Technical Communications of ICLP*, 2016.

[10] M. Gebser et al. Progress in clasp series 3. In *Proc. of LPNMR*, pages 368–383, 2015.

[11] K. Neubauer, P. Wanko, T. Schaub, and C. Haubelt. Enhancing symbolic system synthesis through ASPmT and partial assignment evaluation. In *Proc. of DATE*, pages 306–309, 2017.

[12] G. Brewka, J. Delgrande, J. Romero, and T. Schaub. asprin: Customizing answer set preferences without a headache. In *Proc. of AAAI*, 2015.

[13] E. Zitzler et al. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[14] A. Farhang-Mehr and S. Azarm. An information-theoretic entropy metric for assessing multi-objective optimization solution set quality. *Journal of Mechanical Design*, 125(4):655–663, 2003.