Efficient Design Space Exploration of Embedded Platforms

Martin Lukasiewycz, Florian Sagstetter, Sebastian Steinhorst
TUM CREATE, Singapore
martin.lukasiewycz@tum-create.edu.sg

ABSTRACT

Embedded systems are developed within a cost-driven process, such that many purpose-build systems share a common embedded platform. For instance, a shared hardware device might be applied in various use cases with different software. Here, the bare hardware is considered as *platform* while the final hardware/software system is the *product* that implements a certain set of *features*. In this paper, we propose an efficient design space exploration methodology that optimizes a set of embedded platforms. For this purpose, we separate the exploration into two stages: (1) A set of candidate platforms is determined within an optimization framework that can cope with different system models, considering the optimization objectives and feature coverage. (2) The final set of target platforms is determined as subset of the candidates by minimizing the expected objectives like cost, area, etc. The experimental results give evidence that our modular approach is more flexible and scalable than state-of-the-art methodologies.

Categories and Subject Descriptors

B.0 [**Hardware**]: General

General Terms

Algorithms, Design

Keywords

Design Space Exploration, Platforms, Embedded Systems

1. INTRODUCTION & RELATED WORK

Embedded platforms [1, 2] can be found across all domains in embedded system design, ranging from processors to complex networks that consist of dozens of devices. Due to economic considerations, it is often beneficial to rely on one or more shared platforms for a certain set of embedded products as illustrated in Fig. 1. Depending on the domain, a product (or system) is obtained by additional hardware or software, respectively, as well as a corresponding configuration. On the one hand, the platform constrains which final product is feasible and it should therefore be designed towards the highest possible flexibility. On the other hand, the platform defines the fundamental costs of the system which have to be minimized. These two conflicting objectives make the determination of a set of optimal embedded platforms very challenging.

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

DAC '15, June 07 - 11, 2015, San Francisco, CA, USA

http://dx.doi.org/10.1145/2744769.2744829.

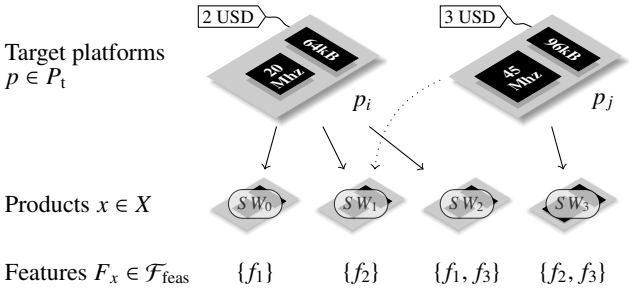

Figure 1: Illustration of two hardware platforms that are used to obtain four products with different features by implementing specific software (SW_x). Note that the second platform might implement a product with $\{f_2\}$ as well but it is more expensive (3 USD instead of 2 USD) and, therefore, the first platform is used.

Design space exploration approaches have been applied in many domains in embedded system design. These domains comprise automotive networks [3], System-on-Chip (SoC) design [4], streaming devices [5], and many more. The goal of these exploration approaches is the system optimization, considering a single or multiple objectives like cost, area, reliability, etc. Depending on the field of application, the optimization is carried out for task and communication mapping, parameter determination, or hardware allocation. While there exist more or less generalized system models for design space exploration [6], often domain-specific models are used that perfectly represent the respective problem.

In general, exploration approaches in literature focus on the optimization of a single system or platform for a specific purpose. By contrast, this paper provides a methodology for the determination of a set of optimal embedded platforms. In this scenario, common exploration approaches from literature reach their limits and, thus, cannot be applied to many real-world design problems in the embedded system domain. As a result, platforms are optimized manually, leading to suboptimal designs. An approach that optimizes multiple platforms was recently presented in [7, 8] where a specific system model is extended to cope with multiple platforms. While the approach shows to be effective for the considered domain with a few variants, we aim at providing a general framework that might be applied with minimal changes to various system and exploration models.

In the marketing domain, product line optimization answers the question how to optimize a set of products with the goal to satisfy customer demands and increase revenue. These optimization methods boil down a selection process of features for a fixed number of products such that costs are kept minimal while getting a high market share. As a result, the combinatorial problem can either be solved by enumeration techniques if it is small or using heuristics for larger problems. An overview of the approaches is given in [9]. In contrast to classic product line optimization techniques from literature, embedded system design beyond the consumer market often has different underlying conditions. The growing extensibility of hardware/software systems enables the implementation of a multitude of different systems. Thus, in contrast to classic product line optimization, it is possible to define embedded platforms that serve as a reusable basis for a set of products.

Contributions of the paper. This paper aims at closing the gap between classic product line optimization and design space explo-

ration for embedded systems. In Section 2, we propose our framework towards the efficient optimization of platforms in the embedded domain. Our approach separates the design space exploration into two stages:

1. For the first stage, detailed in Section 3, we introduce an optimization methodology that is capable of determining a set of candidate platforms, using Binary Decision Diagrams (BDDs) [10] for an efficient feature coverage representation.
2. In the second stage, detailed in Section 4, we present our selection methodology that determines the expected objectives (e.g., cost or area) for a set of target platforms, quantifying the feature coverage efficiently.

Given the various domains of design space exploration problems, this two-stage approach becomes very practical since it enables to implement suitable techniques in a modular fashion for each stage.

Our approach has two major benefits as shown in the experimental results in Section 5:

- The approach shows a very good *scalability*, handling large systems and more than 1000 feasible feature combinations. The scalability is investigated using three synthetic test cases and comparing the results to a reference approach [7].
- The approach is also highly *flexible* in terms of modularity due to the two separate stages as well as in terms of model independence since it can be applied to any kind of exploration model with minimal changes. A case study from the domain of battery management gives evidence of this flexibility by applying our approach to a different system model and considering two conflicting objectives.

Finally, concluding remarks are made in Section 6.

2. FRAMEWORK

In the following, we outline our framework for the design space exploration of embedded platforms. For this purpose, we first briefly introduce the problem before describing our approach.

2.1 Problem Definition

The goal of conventional design space exploration of embedded systems is to determine an optimal system under a set of specified constraints. A search space X is defined such that a product $x \in X$ with an optimal objective function $c(x)$ is determined. The search space is often constrained such that in practice, for instance, Integer Linear Program (ILP) solvers can solve the exploration problem. In case of multiple or non-linear objectives, heuristics like Evolutionary Algorithms (EAs) are used.

For design space exploration of embedded platforms, the goal is not to find a single optimal system but one or more target platforms $P_\mathrm{t} \subseteq P$ that can be extended to products. The set of all possible platforms is defined as P where each platform $p \in P$ can be extended to a set of products $X_p \subseteq X$. Each product x implements a specific set of features F_x. These features define in general the functionality of the product. Correspondingly, the feature coverage of a platform p is defined as $\mathcal{F}_p = \bigcup_{x \in X_p} F_x$. Exemplary, Fig. 1 shows two platforms that can be extended to four products. Note that multiple platforms can be used to obtain a product with identical features in which case the *cheaper* platform is preferred.

The determination of an optimal set of target platforms P_t has to consider many factors. First, the number of platforms $|P_\mathrm{t}|$ is constrained as each platform requires efforts in terms of design, production, and maintenance. A further optimization criterion is defined by the absolute feature coverage of all target platforms. In general, a large feature coverage is expected and some problems require a full feature coverage (this assumption is made in our experimental results) where all possible feasible combinations $\mathcal{F}_\mathrm{feas}$ can be obtained from the platforms. Last but not least, the expected product objectives such as cost, area, etc. determined by the function c have to be optimal.

2.2 Design Flow

The proposed design approach separates the optimization into two stages. First, a set of candidate platforms P_c is determined before a set of target platforms P_t is selected. This design flow is illustrated in Fig. 2.

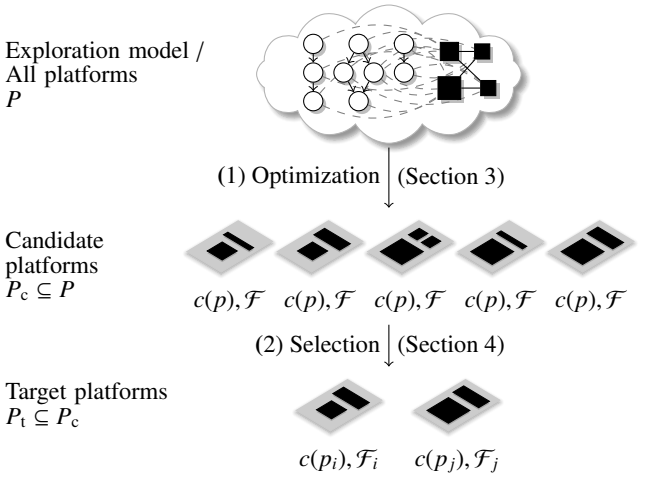

Figure 2: Illustration of the proposed design flow that separates the design space exploration into two stages: optimization and selection.

(1) Optimization. Within the optimization stage, we propose to extend existing design space exploration models such that a set of candidate embedded platforms P_c is determined. This is generally possible with a minor adaptation of the system model since the platform can be deduced from each product, assuming that there is exactly one platform per product.

To determine the candidate set P_c, we extend the domination criterion to both the optimization objectives and the feature coverage. In terms of the feature coverage, a platform dominates another platform if its feature coverage is a superset. We show how this domination criterion can be efficiently implemented with BDDs. As a result, any iterative optimization algorithm can be used to determine the set P_c.

(2) Selection. In the selection stage, the set of Pareto-front platforms P_c has to be reduced to the target platforms P_t. For this purpose, we propose methods to determine the expected objectives for any set of target platforms. Thus, the optimal platforms can be deduced either by complete enumeration or heuristics, using approaches from product line optimization [9].

The determination of the expected objectives of a set of target platforms is non-trivial. We show how this is achieved for both single and multi-objective optimization problems. Again, the compact representation of the feature coverage via BDDs is used to improve the efficiency significantly.

Flexibility and scalability. The design goals of our design space exploration approach are flexibility and scalability. With only minimal changes to existing exploration models, it is possible to use the proposed methodology and, thus, our approach is generally applicable to many domain-specific problems. Moreover, the optimization relies on two stages, where in each stage problem-dependent adaptations can be made. This modularity increases the flexibility of our approach significantly. Finally, our two-stage approach in combination with the usage of BDDs results in a very good scalability as shown in the experimental results. This scalability comprises both the system size as well as the number of features and products.

3. OPTIMIZATION METHODOLOGY

The goal of the optimization methodology is the determination of the candidate set P_c from all feasible platforms P. Here, existing models for conventional design space exploration can be adapted as presented exemplary. At the same time, it is necessary to specify the dominance in terms of both the objective function and the feature coverage. For this purpose, we introduce an efficient encoding of the feature coverage as BDD.

3.1 Optimization Process

In order to determine the set P_c, we propose a general procedure as given in Algorithm 1 which might be used within any iterative optimization approach.

Algorithm 1: Iterative procedure to determine the set of non-dominated platforms P_{nd}.

```
1  P_c ← {}
2  while true do
3  |   determine x ∈ X
4  |   deduce p with x ∈ X_p
5  |   determine c(p)
6  |   F_p = {}
7  |   while X_p ≠ {} do
8  |   |   determine x ∈ X_p with F_x
9  |   |   X_p ← X_p \ {x̃ | F_x̃ ⊆ F_x}
10 |   |   F_p ← F_p ∪ (⋃_{F⊆F_x} {F} ∩ F_feas)
11 |   end
12 |   P_c ← P_c ∪ {p}
13 |   P_c ← {p̃ | p̃ ∈ P_c, ∀p̄ ∈ P_c \ {p̃} : ¬(p̄ ⪯ p̃)}
14 end
```

The procedure starts with a set P_c that is iteratively filled (line 1). It is required to define a search space that covers all feasible products X and in each iteration one of these products is determined (line 3). The products X are often defined by a set of constraints as shown exemplary for a simple model in the following Subsection 3.2. From the obtained product, platform p is deduced (line 4). In a next step, objectives such as cost, area, etc. of the platform are determined (line 5).

The feature coverage of the platform $\mathcal{F}_p$ is determined iteratively (lines 6-10). Here, the search space X_p is deduced from X by adding constraints that restrict it to a single platform (by setting of resource variables to either 0 or 1). The set of all products for the current platform is then iterated, determining the next product x with its specific feature coverage F_x. All products with these features or any subset are removed from the search space X_p (line 9) while the feature coverage is extended with exactly these feature sets that are at the same time feasible (line 10). The efficient encoding and, thus, determination of the feature coverage using BDDs is proposed in Subsection 3.3.

The set P_c has to be updated with the currently obtained platform (line 12) and dominated platforms have to be removed (line 13). The domination criterion ($\preceq$) is satisfied if a platform is not worse in any of its objectives determined by c as well as the domination for the feature coverage is fulfilled. The domination for feature coverage for two platforms is defined as follows:

$$p \overset{\mathcal{F}}{\preceq} \tilde{p} \text{ iff } \mathcal{F}_p \supseteq \mathcal{F}_{\tilde{p}} \tag{1}$$

That means, each feature set that is covered by platform $\tilde{p}$ is also covered by platform p. Thus, a general domination is defined via the domination of feature coverage and the domination of optimization objectives:

$$p \preceq \tilde{p} \text{ iff } p \overset{\mathcal{F}}{\preceq} \tilde{p} \wedge p \overset{c}{\preceq} \tilde{p} \tag{2}$$

With this domination, it is possible to construct the set P_c, considering all relevant platforms that might be necessary to determine the optimal target platforms within the selection framework.

3.2 Platform-aware Model Encoding

In the following, we present how the model from [6] might be extended to allow the determination of platforms (deduced from a determined product). The model consists of a task graph, resource graph, and the mappings between both. The task graph G_T consists of vertices that represent tasks T and directed edges E_T that represent data-dependencies. The resource graph G_R consists of vertices that represent resources R and edges E_R that represent a possible communication between two resources. The mappings $M \subseteq T \times R$ contain edges between tasks and resources, indicating possible implementations. An example of this model is given in Fig. 3.

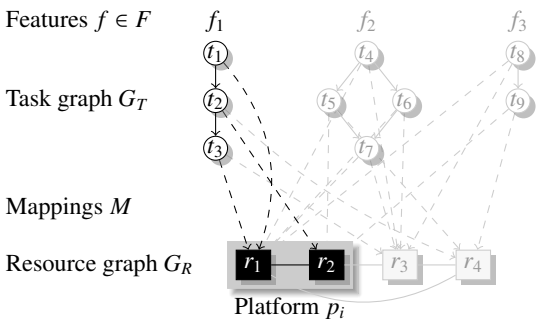

Figure 3: Illustration of the used exploration model. A set of data-dependent tasks are mapped to interconnected resources. Exemplary, a mapping of a set of tasks is illustrated, resulting in platform p_i from Fig. 1 and the product that implements the feature set $\{f_1\}$.

The model requires that an allocation $\alpha \subseteq R$ is determined. Additionally, for each task $t \in T$, exactly one mapping edge is selected such that the binding $\beta \subseteq M$ ensures that data-dependent tasks are either mapped on the same resource or adjacent resources which are allocated. By introducing binary variables (indicated by bold characters) for all mappings and resources that represent whether a certain mapping is bound or resource is allocated, respectively, linear constraints can be defined as follows:

$$\forall t \in T: \sum_{m=(t,r)\in M} \mathbf{m} = 1 \tag{3}$$

$$\forall m = (t,r) \in M: \mathbf{r} - \mathbf{m} \geq 0 \tag{4}$$

$$\forall t, \tilde{t} \in E_T, m = (t,r), \tilde{m} = (\tilde{t}, \tilde{r}) \in M, r \neq \tilde{r} \wedge (r, \tilde{r}) \notin E_R: \mathbf{m} + \tilde{\mathbf{m}} \leq 1 \tag{5}$$

Constraint (3) specifies that for each task exactly one mapping has to be used. Constraint (4) ensures that tasks are only bound to allocated resources. Finally, Constraint (5) guarantees that data-dependent tasks are not bound to resources where no communication is possible.

Defining the exploration model using a formulation like Constraints (3)-(5) is a widely applied practice to allow, for instance, the usage of ILP solvers. To extend the exploration model such that platforms can be deduced, it is necessary to model specific products. For this purpose, we first need to model which feature $f \in F$ is implemented by introducing a binary variable for each feature. Consequently, we replace Constraint (3) as follows where $f \rightarrow t$ indicates that a task t (which might be applicable to multiple tasks) needs to be implemented to enable a certain feature f:

$$\forall t \in T, f \rightarrow t: \Big(\sum_{m=(t,r)\in M} \mathbf{m} \Big) - \mathbf{f} = 0 \tag{6}$$

That means a task is only bound to a resource if and only if the respective feature is implemented.

Additional feature constraints can be added to obtain only products with a feasible feature set. For our model, we propose an additional constraint over features where exactly one has to be selected (in $\mathcal{F}_{=1}$):

$$\forall F_{=1} \in \mathcal{F}_{=1}: \sum_{f \in F_{=1}} \mathbf{f} = 1 \tag{7}$$

Constraint (7) is for instance resulting in $\mathbf{f}_1 + \mathbf{f}_2 = 1$ for the example from Fig. 1. This constraint enables to model realistic feature constraints which might be extended with additional domain-specific constraints. All other features that are not in $\mathcal{F}_{=1}$ are considered to be optional (see f_3 in Fig. 1).

In summary, Constraints (6) and (7) extend the existing model towards a platform-aware exploration. While the discussed model from [6] is relatively simple, the extended model from [4] that also allows multi-hop and multi-cast communication tasks can be modified with exactly the same two Constraints (6) and (7). Thus, the modifications that are required to make the model platform-aware remain minimal.

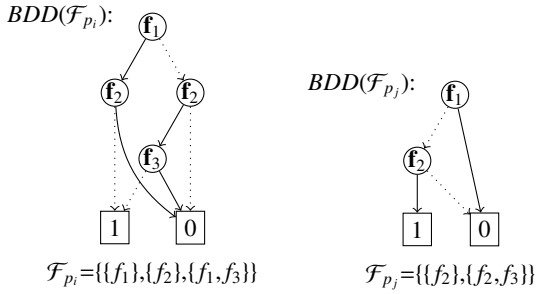

Figure 4: Illustration of BDD representation of the feature coverage for the platforms from Fig. 1. Each path from the root in the BDD that results in 1 *means that the respective feature set is covered. An outgoing solid edge (⟶) means that the feature is contained, an outgoing dotted edge (⋯›) means that the feature is absent. If a certain feature is not in the path, it may be both contained and absent.*

3.3 Feature Encoding

The determination of the dominance in Eq. (1) can become expensive, particularly because the number of elements in $\mathcal{F}$ might grow exponentially in the number of features. In order to efficiently implement the dominance relation for the feature coverage, we propose to use BDDs for encoding a feature set $\mathcal{F}_p$ of a platform p.

A BDD is an acyclic graph that efficiently encodes Boolean functions [10] with a single root node and two terminal nodes that represent the function results. Starting in the root node, each path in the BDD represents a certain binary assignment of input variables, leading to the result 0 or 1, respectively. The assignment of a variable is determined by choosing the respective outgoing edge for a specific node that represents a 0 or 1 assignment, respectively. If a variable is not in the path, it may have any assignment.

In this paper, we use the fact that BDDs can also be used to represent sets [11]. An illustration of BDDs representing the feature coverage of the platforms from Fig. 1 is given in Fig. 4. In the following, we show how to efficiently construct the BDD for the feature coverage and how to define the domination criterion from Eq. (1) using BDDs.

For a given feature coverage $\mathcal{F}_p$, the BDD is defined as Boolean function in disjunctive normal form as follows (bold characters are used to indicate binary variables):

$$BDD(\mathcal{F}_p) = \bigvee_{\tilde{F}\in\mathcal{F}_p} \Big(\bigwedge_{\tilde{f}\in\tilde{F}} \tilde{\mathbf{f}} \bigwedge_{f\in F\setminus\tilde{F}} \neg\mathbf{f} \Big) \tag{8}$$

This Boolean function encodes explicitly every element in $\mathcal{F}_p$ while the corresponding BDD generally encodes the set in a very efficient way, see Fig. 4. In order to benefit from the BDD representation, it becomes necessary to provide an efficient method to determine $\mathcal{F}_p$ in Alg. 1 (lines 6-10). While the initial feature coverage $BDD(\mathcal{F}_p) = 0$ is empty (line 6), the operation that incrementally extends the feature coverage (line 10) in each iteration directly translates to:

$$BDD(\mathcal{F}_p) \leftarrow BDD(\mathcal{F}_p) \vee \Big(\bigwedge_{f\in F\setminus F_x} \neg\mathbf{f} \wedge BDD(\mathcal{F}_{\text{feas}}) \Big) \tag{9}$$

Here, the union translates to a logical *OR* and the intersection to a local *AND*, making the BDD operation very efficient, particularly with a growing number of features. Note that within the iterative process, it is possible to restrict X_p (line 9) with the following constraint:

$$\sum_{f\in F\setminus F_x} \mathbf{f} \geq 0 \tag{10}$$

Thus, each solution with the feature set F_x and all subsets are excluded. This ensures that in the next iteration a product with a different feature set is obtained.

Finally, for the corresponding BDDs, the dominance criterion from Eq. (1) translates to the following operation:

$$p \overset{\mathcal{F}}{\preceq} \tilde{p} \text{ iff } BDD(\mathcal{F}_p) \vee BDD(\mathcal{F}_{\tilde{p}}) = BDD(\mathcal{F}_p) \tag{11}$$

∧, ∨, ¬ correspond to *AND*, *OR*, and *NOT*, respectively, in Boolean algebra.

As Eq. (1) equals the condition $\mathcal{F}_p \cup \mathcal{F}_{\tilde{p}} = \mathcal{F}_p$, the BDD operation is identical and efficiently determines the dominance.

4. SELECTION METHODOLOGY

Given the candidate platforms P_c, the second stage of our framework determines a subset of final target platforms P_t. To determine the target platforms, it is necessary to be able to quantify the feature coverage of a certain platform as described in the following. Finally, we will show how the expected objectives (or costs) of a set of target platforms are determined.

4.1 Quantitative Feature Coverage

For the evaluation of target platforms, it is necessary to determine the quantitative value of feature coverage that can be covered by a certain platform. In general, it cannot be assumed that each feature set is equally distributed in terms of demand. Each feature is associated with a certain probability $Pr(f)$ that determines what is the likelihood that a certain feature is required or requested, respectively, in a final product. This is in fact a common approach in the marketing domain [9], resulting in sufficiently precise estimations of product demands. For features where exactly one of them has to be selected, the requirement is further (see f_1 and f_2 in Fig. 1):

$$\forall F_{=1} \in \mathcal{F}_{=1} : \sum_{f\in F_{=1}} Pr(f) = 1 \tag{12}$$

To determine the probability for a certain feature set $Pr(\mathcal{F})$, we can again make use of the efficient data representation as BDD. For this purpose, the Shannon-decomposition as proposed in [12] is used:

$$\begin{aligned} Pr(BDD(\mathcal{F})) = {} & Pr(f) \cdot Pr(BDD(\mathcal{F})|_{\mathbf{f}=1}) \\ & + \overline{Pr}(f) \cdot Pr(BDD(\mathcal{F})|_{\mathbf{f}=0}) \end{aligned} \tag{13}$$

where $Pr(BDD(\mathcal{F})|_{\mathbf{f}=1})$ and $Pr(BDD(\mathcal{F})|_{\mathbf{f}=0})$ define the succeeding nodes in the BDD following the outgoing 1-edge or 0-edge, respectively. Applying the Shannon-decomposition from Eq. (13) requires to iterate the BDD in its reverse order from the terminals to the root. This is illustrated in Fig. 5(a) for the platforms from Fig. 1. Thus, this approach is linear in the number of BDD nodes and usually very efficient in contrast to explicitly determining the probability of each feature set in $\mathcal{F}$.

For the complementary probability $\overline{Pr}(f)$, i.e., that a certain feature is not requested, we apply:

$$\overline{Pr}(f) = \begin{cases} 1 & \text{if } \exists F_{=1} \in \mathcal{F}_{=1} : f \in F_{=1} \\ 1 - Pr(f) & \text{otherwise} \end{cases} \tag{14}$$

Here, we extend [12] by introducing the special case for determining $\overline{Pr}(f)$ in case of selective features as defined in Eq. (12).

4.2 Expected Objectives Determination

To determine the quality of a set of target platforms P_t, it is necessary to evaluate the expected objectives of a product, taking into account the demand probabilities for certain features. This evaluation has also to take into account that one product x might be obtained from multiple different target platforms $p, \tilde{p} \in P_t$ with $x \in (X_p \cap X_{\tilde{p}})$. In this case, the platform with *better* objective values has to be chosen to ensure the most optimal expected objectives with reference to the feature distribution. Besides the expected objectives, it is important to ensure that the target platforms P_t satisfy a certain threshold in terms of their absolute feature coverage. The absolute feature coverage is determined as follows:

$$Pr\Big(\bigcup_{p\in P_t} \mathcal{F}_p\Big) = Pr\Big(\bigvee_{p\in P_t} BDD(\mathcal{F}_p)\Big) \tag{15}$$

In general, this threshold should be relatively high. In our experimental results we assume it to be 1.0, i.e., full coverage.

Single-objective. In case of a single objective function, determining the expected objectives is done as follows:

$$c(P_t) = \sum_{p\in P_t} \widehat{Pr}(p) \cdot c(p) \tag{16}$$

Fig. 5(b) shows how the expected costs are determined for the platforms in Fig. 1. Here, $\widehat{Pr}(p)$ is the quantitative residual feature

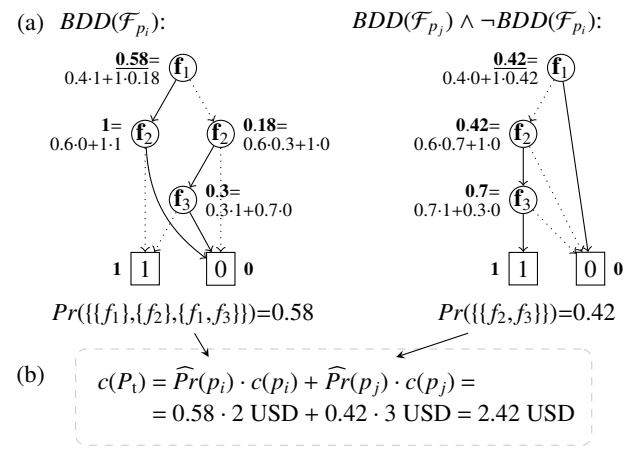

Figure 5: In (a), the feature coverage is quantified using the Shannon-decomposition. The individual feature probabilities are $Pr(f_1)$=0.4, $Pr(f_2)$=0.6, $Pr(f_3)$=0.7, $\overline{Pr}(f_3)$=0.3 while either f_1 or f_2 has to be implemented and f_3 is optional. In (b), the expected costs (2.42 USD) of the platforms from Fig. 1 are determined as illustrated.

coverage, i.e., only considering actual features that will finally be covered by the platform (see Fig. 1 where p_j does not cover $\{f_2\}$ because this can be obtained as well by the cheaper platform p_i). Using the BDD representation for feature coverage, the quantitative residual feature coverage is determined as follows:

$$\widehat{Pr}(p) = Pr\Big(\mathcal{F}_p \setminus \bigcup_{\overline{p}\in P_t, c(\overline{p})<c(p)} \mathcal{F}_{\overline{p}}\Big) =$$

$$= Pr\Big(BDD(\mathcal{F}_p) \wedge \neg \bigvee_{\overline{p}\in P_t, c(\overline{p})<c(p)} BDD(\mathcal{F}_{\overline{p}})\Big) \quad (17)$$

Here, all features that can be obtained by a better optimized platform $\overline{p}$ with $c(\overline{p}) < c(p)$ are removed. Fig. 5(a) illustrates the BDDs that represent the residual feature coverage for the platforms in Fig. 1. Note that here the residual feature coverage for p_j is determined by removing the feature set $\{f_2\}$.

Multi-objective. In case of a multi-objective design space exploration problem, it is generally not possible to obtain an order by the objective function. Therefore, the function in Eq. (16) is extended to:

$$c(P_t) = \sum_{\tilde{P}_t \subseteq P_t} \widehat{Pr}(\tilde{P}_t) \cdot \overline{c}(\tilde{P}_t) \quad (18)$$

The first part determines the feature coverage of a subset $\tilde{P}_t$ of target platforms while the second part determines the average objectives of this subset. Determining the feature coverage of the subset $\tilde{P}_t$ translates to BDD operations as follows:

$$\widehat{Pr}(\tilde{P}_t) = Pr\Big(\bigcap_{p\in\tilde{P}_t} \mathcal{F}_p \setminus \Big(\bigcup_{\substack{p\in\tilde{P}_t, \overline{p}\in P_t \\ c(\overline{p})<c(p)}} \mathcal{F}_{\overline{p}} \bigcup_{\overline{P}_t \supset \tilde{P}_t} \bigcap_{\overline{p}\in\overline{P}_t} \mathcal{F}_{\overline{p}}\Big)\Big) =$$

$$= Pr\Big(\bigwedge_{p\in\tilde{P}_t} BDD(\mathcal{F}_p) \wedge \neg \bigvee_{\substack{p\in\tilde{P}_t, \overline{p}\in P_t \\ c(\overline{p})<c(p)}} BDD(\mathcal{F}_{\overline{p}}) \wedge \neg \bigvee_{\overline{P}_t \supset \tilde{P}_t} \bigwedge_{\overline{p}\in\overline{P}_t} BDD(\mathcal{F}_{\overline{p}})\Big) \quad (19)$$

This corresponds to Eq. (17) for the multi-objective case by reducing the feature coverage by dominating platforms ($\prec$) and supersets of platforms ($\overline{P_t}$). The average objectives in Eq. (18) are determined assuming a uniform distribution:

$$\overline{c}(\tilde{P}_t) = \frac{\sum_{p\in\tilde{P}_t} c(p)}{|\tilde{P}_t|} \quad (20)$$

5. EXPERIMENTAL RESULTS

In the following, we present experimental results that give evidence of the flexibility and scalability of our approach. First, we use three synthetic test cases and compare our framework with the approach from [7]. Finally, a realistic case study from the domain of battery management is optimized with our approach, using a different exploration model. All experiments were carried out on an Intel Core i5 at 2.6 GHz with 8 GB RAM.

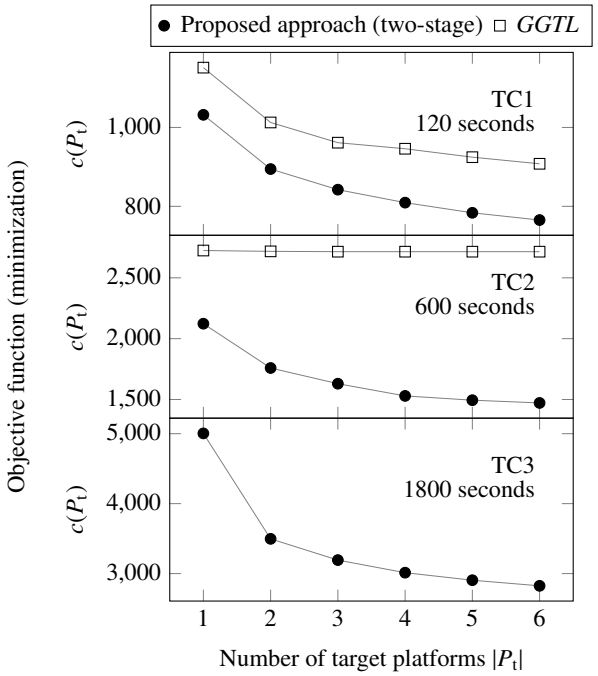

*Figure 6: Results of the three test cases to give evidence of the scalability of the proposed approach in comparison with an approach from literature (*GGTL*). Annotated is the maximum runtime after which the optimization process is stopped.*

5.1 Synthetic Test Cases

In order to show the scalability of our approach, we used three synthetic test cases as outlined in Table 1. These test cases rely on the model from [4] that allows multi-hop and multi-cast communication. Note that the first test case (TC1) resembles the case study from [7] with only eight feasible feature sets. Since these eight feasible sets cannot be described with the proposed constraints on features (see Constraint (7)), we explicitly modeled the feasible feature sets $\mathcal{F}_{\text{feas}}$ as one BDD that is then translated into linear constraints using [13].

We also re-implemented the method from [7] (*GGTL*) to enable a comparison with a reference approach. For the sake of simplicity, we consider a single objective function. The optimization process follows the heuristic EA approach from [4] to obtain feasible products in each iteration while each test case has a maximal runtime as indicated in Fig. 6.

The results of the test cases are illustrated in Fig. 6. It can be observed that *GGTL* does not scale well in the number of possible feature sets. While *GGTL* still delivers competitive results for TC1, it fails to scale for TC2 and the model becomes too large for TC3 (exceeding the 8 GB of memory) because it models each product separately. In comparison, our proposed approach is still capable of handling more than 1000 different products or feature sets, respectively, and larger specifications without any problem. Note that for the selection process in the proposed approach, enumerating all

Table 1: Overview of test case parameters.

Test case	features $\lvert F\rvert$	features size $\lvert\mathcal{F}_{\text{feas}}\rvert$	tasks $\lvert T\rvert$	resources $\lvert R\rvert$	mappings $\lvert M\rvert$
TC1	8	8	42	13	90
TC2	10	96	54	25	105
TC3	15	1152	81	41	238

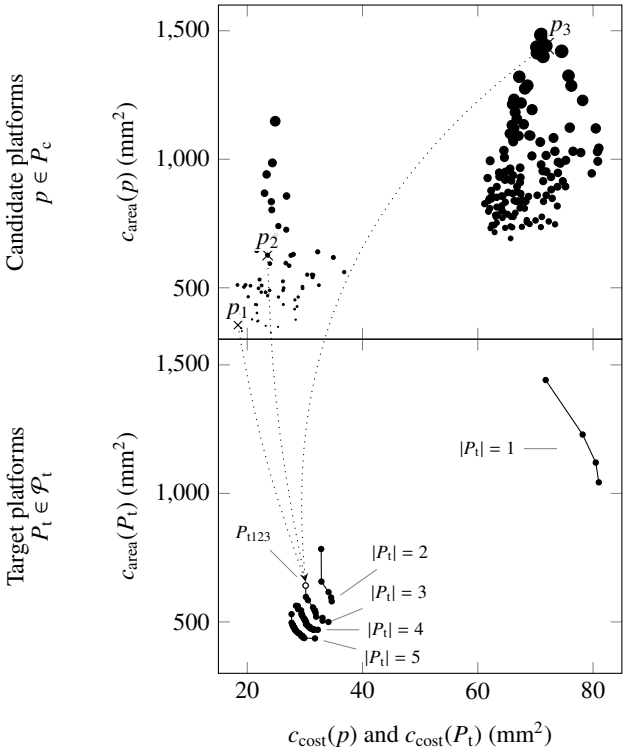

Figure 7: Illustration of the case study results. The first plot illustrates all 200 platforms in P_c where the size of the dots indicates the platform's quantitative feature coverage. The second plot illustrates all optimal target platforms P_t, resulting in fronts for each number of platforms due to the multi-objective nature of the problem. Exemplary, the target platforms $P_{t123} = \{p_1, p_2, p_3\}$ (○) are highlighted and detailed in Table 2.

possible combinations was used for up to three platforms and an EA was used if the number of target platforms was higher. In both cases, the runtime of the selection process was in the range of a few seconds.

5.2 Case Study - BMS Slave Board

Finally, a case study from the domain of battery management was used to evaluate the proposed approach. The goal was the design of a slave Printed Circuit Board (PCB) within a Battery Management System (BMS) for a large battery pack for an electric vehicle. This slave board monitors and controls 12 battery cells. A slave board has one micro-controller, a monitoring IC, and a set of sensors (temperature, humidity, etc.) and communication controllers (CAN, Ethernet). For each resource, we explore a set of components with their cost (in USD) and overall IC area (in mm^2), e.g., the controller can be chosen from 20 different components from different vendors. The exploration model that is capable of handling different components is available at [14] and is a superset of [4]. We explored features like different sensing functions, state-of-charge estimation algorithms, and charge balancing capabilities. In summary, the exploration problem contains 640 feasible feature combinations, 71 tasks, and 14 resources (for which there exist many different components). The large size and the different exploration model made it infeasible to use the reference approach *GGTL*.

The results of the case study are illustrated in Fig. 7. After the optimization stage, 200 candidate platforms P_c were obtained within 1200 seconds. Note that these 200 platforms are candidates that have to be reduced to a much smaller number in the second stage. It can be seen that two clusters of candidate platforms exist: The solutions with passive battery cell balancing and the ones with active cell balancing which require significantly more expensive LTC8584 ICs (one for each cell).

After the selection stage that required 630 seconds, 80 Pareto-optimal target platform sets $\mathcal{P}_t$ were obtained. Due to the two concurrent objectives, there exist multiple optimal solutions for a certain number of target platforms $|P_t|$. It is finally the task of a decision maker to decide for a single set of target platforms in $\mathcal{P}_t$. Exemplary for a solution with three platforms, we choose P_{t123} (see Fig. 7) which is detailed in Table 2. Increasing the number of target platforms from one to two already significantly reduces the expected cost and area. This is due to the fact that one expensive board is required to allow active cell balancing while the cheaper board with passive cell balancing has a relatively high feature coverage. For P_{t123}, the expensive board is only required in about 19% of all products and, thus, it contributes less to the expected cost and area.

6. CONCLUDING REMARKS

In this paper, we proposed a very efficient approach for the exploration of embedded platforms. Our methodology does not require a specific model and, thus, is generally applicable to any domain-specific problem. The experimental results clearly give evidence of the good scalability of the proposed approach.

The high modularity due to two stages and the independence from the exploration model make the proposed methodology very flexible. Therefore, it can subsequently be extended to handle variable component costs as well as to incorporate profit margin calculations. Finally, the proposed work can serve as basis for further research in the domain of embedded platform exploration.

7. REFERENCES

[1] G. Beltrame, L. Fossati, and D. Sciuto, "Decision-theoretic design space exploration of multiprocessor platforms," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 7, pp. 1083–1095, 2010.

[2] A. Sangiovanni-Vincentelli, L. Carloni, F. De Bernardinis, and M. Sgroi, "Benefits and challenges for platform-based design," in *Proc. of DAC*, 2004, pp. 409–414.

[3] M. Zeller and C. Prehofer, "Modeling and efficient solving of extra-functional properties for adaptation in networked embedded real-time systems," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 1067–1082, 2013.

[4] M. Lukasiewycz, M. Streubühr, M. Glaß, C. Haubelt, and J. Teich, "Combined system synthesis and communication architecture exploration for mpsocs," in *Proc. of DATE*, 2009, pp. 472–477.

[5] J. Keinert, T. Schlichter, J. Falk, J. Gladigau, C. Haubelt, J. Teich, M. Meredith *et al.*, "SystemCoDesigner - an automatic esl synthesis approach by design space exploration and behavioral synthesis for streaming applications," *ACM Trans. on Design Automation of Electronic Systems*, vol. 14, no. 1, p. 1, 2009.

[6] T. Blickle, J. Teich, and L. Thiele, "System-level synthesis using evolutionary algorithms," *Design Automation for Embedded Systems*, vol. 3, no. 1, pp. 23–58, 1998.

[7] S. Graf, M. Glaß, J. Teich, and C. Lauer, "Multi-variant-based design space exploration for automotive embedded systems," in *Proc. of DATE*, 2014, pp. 1–6.

[8] ——, "Design space exploration for automotive E/E architecture component platforms," in *Proc. of DSD*, 2014.

[9] A. Belloni, R. Freund, M. Selove, and D. Simester, "Optimizing product line designs: Efficient methods and comparisons," *Management Science*, vol. 54, no. 9, pp. 1544–1552, 2008.

[10] R. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. on Computers*, vol. 100, no. 8, pp. 677–691, 1986.

[11] J. Burch, E. Clarke, K. McMillan, D. Dill, and L.-J. Hwang, "Symbolic model checking: 10^{20} states and beyond," in *Proc. of LICS*, 1990, pp. 428–439.

[12] A. Rauzy, "New algorithms for fault trees analysis," *Reliability Engineering & System Safety*, vol. 40, no. 3, pp. 203–211, 1993.

[13] N. Eén and N. Sörensson, "Translating pseudo-boolean constraints into sat," *JSAT*, vol. 2, no. 1-4, pp. 1–26, 2006.

[14] OpenDSE, "Open design space exploration framework," http://opendse.sf.net/.

Table 2: Components, cost, and area of three selected target platforms P_{t123}.

	Controller	Monitoring IC	c_{cost} (USD)	c_{area} (mm^2)	$Pr(p)$	$\widehat{Pr}(p)$
p_1	MK20DN32 50Mhz/32kB	LTC6803	**18.39**	**355**	0.5233	**0.5233**
p_2	STM32F373 72Mhz/32kB	LTC6803	**23.54**	**627**	0.8075	**0.2842**
p_3	STM32F407 168Mhz/192kB	LTC6804 +LTC8584(x12)	**71.78**	**1441**	1.0	**0.1925**
$P_{t123} = \{p_1, p_2, p_3\}$			**30.13**	**641**		**1.0**