Raising Abstraction in Timing Analysis for Vehicular Embedded Systems through Model-Driven Engineering

Alessio Bucaioni[1][2][⋆]

[1] Mälardalen University, Västerås, Sweden, alessio.bucaioni@mdh.se
[2] Arcticus Systems AB, Järfalla, Sweden, alessio.bucaioni@arcticus.se

Abstract. The complexity of vehicular embedded systems is continuously increasing and this can negatively affect their development cost and time to market. One way to alleviate these issues is to anticipate analysis of system properties at design time for early architectural refinements. In this paper, we present a licentiate work which aims at contributing to this effort. In particular, considering the importance of timing constraints typical of vehicular embedded systems, we leverage Model-Driven Engineering for realizing an automatic approach which allows the developer to perform timing analysis on design models, without having to manually specify timing elements. The proposed approach, starting from a high-level model of the vehicular embedded application, generates a set of candidate models enriched with timing elements in a semi-automatic manner. Timing analysis is run on the generated models and, based on its results, the approach supports the selection of the best candidate model for a specific, non-empty, set of timing constraints.

Keywords: Vehicular Embedded Systems, Model-Driven Engineering, Component-Based Software Engineering, EAST-ADL

1 Introduction

During the last decades, ever-growing complexity of vehicular embedded systems development negatively affected their development cost and time-to-market [17]. To mitigate these issues, a common practice is to anticipate analysis of systems properties at design time to drive early architectural refinements. In this paper, we present a licentiate work[3] which aims at contributing to this effort. More precisely, we propose an approach to allow the developer to perform end-to-end and delay timing analysis[4] [5] on design models without having to manually specify their timing elements. Starting from a high-level model of a vehicular embedded application, we provide a semi-automated mechanism for generating a set of

[⋆] Copyright held by the author.
[3] Within the Swedish Higher Education Systems, the Degree of Licentiate is a third-cycle qualification formally equivalent to half of the Degree of Doctor.
[4] In the remainder of the paper we will refer to end-to-end and delay timing analysis simply as *timing analysis*.
[5] We refer the reader to [13] [18] for further details on timing analysis.

candidate models enriched with timing elements with the aim of enabling early timing analysis. Leveraging the timing analysis results, we support the selection of the best candidate model for a specific, non-empty, set of timing constraints.

1.1 Context

One of the first attempts in mitigating the increasing complexity of vehicular embedded systems development was the establishment of different views [14], each of which often exploiting a specific language, in the software development. As a result, the vehicular software development was characterized by a plethora of heterogeneous languages each targeting specific aspects related to a particular view. Nevertheless, the usage of heterogeneous languages introduced new challenges towards interoperability, e.g., integration between general purpose languages, e.g., UML, and domain-specific languages, e.g., AUTOSAR. In trying to solve to these challenges, the vehicular embedded research community developed a layered architectural language, namely EAST-ADL [2].

EAST-ADL proposes a top-down development process composed by four different abstraction levels, i.e., vehicle, analysis, design and implementation level. Within EAST-ADL, interoperability is ensured by well-defined relationships among elements in the different abstraction levels. Nevertheless, these relationships are not leveraged in any mechanism supporting the automatic translation of different artifacts through the EAST-ADL abstraction levels. For this reason, automotive industry is currently pushing for having a closer linkage among the EAST-ADL abstraction levels for enabling a seamless development chain that takes into account relationships among different levels. Such a chain would improve the development of vehicular embedded software by providing automation of tedious and error-prone activities, e.g., transition from one level to another. Towards this goal, the vehicular embedded research community is considering the adoption of Model-Driven Engineering (MDE).

EAST-ADL. EAST-ADL [2] is an architecture description language for modeling product-lines of vehicular embedded systems. Currently it is managed by the EAST-ADL Association together with the European FP7 MAENAD project. EAST-ADL proposes a view over the development process composed by four different abstraction levels, which implicitly ensure separation of concerns through the different engineering phases. Each abstraction level is described by means of metamodeling constructs. Figure 1 shows the abstraction levels together with methodologies and languages used at each one of them. EAST-ADL does not provide modeling constructs for representing the software implementation architecture. Instead, for the last abstraction level, i.e., implementation level, EAST-ADL suggests the usage of existing modeling languages, e.g., AUTOSAR, the Rubus Component Model (RCM).

*Vehicle Level.*The highest abstraction level is represented by the vehicle level, which captures information regarding the system's functionality. Feature models can be used for showing what the system provides in terms of functionality. These models are decorated with requirements. The vehicle level is also known as end-to-end level as it serves to capture requirements and features on the end-to-end vehicle functionality.

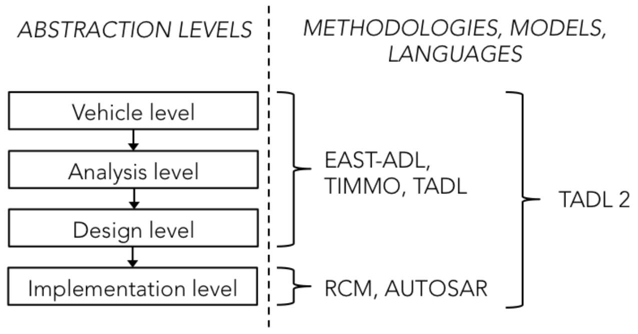


Fig. 1: EAST-ADL abstraction levels

Analysis level. At the analysis level, vehicle functions are expressed, using formal notations, in terms of behaviors and interfaces. Yet, design and implementation details are omitted. At this stage, high level analysis for functional verification can be performed.
Design level. At this abstraction level, the analysis-level artifacts are refined with design-oriented details: while the analysis level does not differentiate among software, middleware and hardware, the design level explicitly separates them. Allocation of software functions to hardware nodes is expressed at this level too.
Implementation level. At the implementation level, artifacts introduced at the design level are refined with implementation details. At this stage component models, e.g., RCM, AUTOSAR, can be used to model the system in terms of components and interactions among them. The output of this level is a complete software architecture which can be used for code generation.

Rubus Component Model Rubus Component Model (RCM) is a component model for the development of resource-constrained embedded real-time systems. It is developed by Arcticus Systems AB in collaboration with Mälardalen University and it is currently adopted by several companies as alternative to, e.g., AUTOSAR. In fact, in contrast with its competitors, RCM offers high-precision timing analysis together with a well-established supporting framework.

In the context of EAST-ADL abstraction levels, RCM is used at the implementation level (Figure 1). Its main goal is to express the software architecture in terms of software functions and interactions among them. In RCM, the basic entity is the so-called *software circuit* (SWC) which represents the lowest-level hierarchical element in RCM and it encapsulates basic software functions. Each SWC is defined by its *behavior* and *interface*. Interfaces manage the interactions among SWCs via ports. RCM distinguishes between data and control flow. Therefore, the interfaces have two types of ports: *data ports* for the data flow and *trigger ports* for the control flow. SWCs are characterized by run-to-completion semantics meaning that a SWC, upon triggering, reads data from the data input ports, executes its behavior and writes data on the data output ports. SWCs can be grouped and organized in *assemblies*, for decomposing the system in a hierarchical manner. *Modes* are used to distinguish among different states of the system. That is, each mode describes the architecture of the functions which are relevant for that mode. *Target* entities are used for grouping modes that are deployed on the same Electronic Control Unit (ECU). Moreover, they provide details regarding hardware and operating system. *Node* is a hardware and operating-system independent abstraction of a target entity. Finally, *System* is the top-level hierarchical entity, which describes software architecture for the complete vehicular system. RCM facilitates analysis and reuse of components in different contexts by separating functional code from the infrastructure that implements the execution environment.

The RCM metamodel definition is part of the intended research contributions of the Licentiate thesis.

1.2 Paper Outline

The rest of the paper is organized as follows. Section 2 describes the research problem. Section 3 presents the proposes solution and the related research contributions. Section 4 describes the work-to date and the current status. Finally, Section 5 and Section 6 discuss the validation methodologies and related work, respectively.

2 Problem Formulation

In this section, we discuss the research goal for the licentiate work, together with the research challenges to be tackled towards its achievement.

2.1 Research Goal

Given the timing constraints typical of vehicular embedded applications, anticipating timing analysis is a way for mitigating development issues, such as cost and time-to-market. Within the EAST-ADL development methodology, the way towards early timing analysis is hampered by the weak linkage between the modeling language used at the implementation level - where timing analysis is usually performed - and the design level. The goal of this licentiate research work is to provide a semi-automated support for allowing timing analysis on design models, without the need of manually adding their timing elements.

2.2 Research Challenges

RC 1 – Definition of a metamodeling characterization of RCM. Although EAST-ADL does not fully embrace the MDE paradigm, the languages defined for the abstraction levels are formalized by metamodeling. For this reason we want to leverage MDE for automating the development of the vehicle embedded software in EAST-ADL. To this end, all the involved languages have to be provided with a proper metamodel definition. In our case, the challenge is the definition of a metamodel for RCM, the language used at implementation level.

RC 2 – Definition of a mapping between EAST-ADL design level metamodel and RCM metamodel. Due to the lack of timing information, high-precision timing analysis can not be performed at the EAST-ADL design level. One way to solve this issue, is the automatic translation of design level models to implementation level models, i.e., RCM models, on which timing analysis can be performed. RCM models contain in fact timing elements, e.g. clocks, control-flow ports, to mention a few, that cannot be modeled at the design level. These elements represent a variability points in the transition from design to implementation level, meaning that more than one RCM model can be a valid translation of a given EAST-ADL design level model. The challenge is to define and implement a semi-automatic translation such that all the possible RCM models for a given EAST-ADL design level model are produced without any error-prone and time consuming manual activity. Semi-automatic in the sense that, while timing elements (e.g., clock, control flow port) can be fully automatically specified, their completion with actual timing properties (e.g., clock period, wcet) is still guided by the developer.

RC 3 – Definition of a mechanism for the selection of the best RCM model for a set of timing constraints. Starting from the generated RCM models, the challenge is to define a mechanism able to, based on the analysis results, select the RCM model which better meets the given set of timing constraints. In fact, at the end of the process, starting from a design level model, either an RCM model is chosen or refinements of the design level model are needed. This represents the last step for exploiting timing analysis results at design level for early enabling software architecture refinements.

3 Proposed Solution and Intended Contributions

The contribution of the licentiate work presented in this paper, is the definition of a process which, from a design level model, generates a set of candidate implementation level models enriched with timing elements whose properties are set at generation time by the developer. That is to say, the developer drives the automatic generation of all the relevant combinations of timing elements by *inserting timing properties only once per element* instead of having to manually edit all the generated models. At this point timing analysis can be run and from its results, the best candidate implementation model, for a specific timing property or a set of them, is selected. Figure 2 depicts the proposed approach and it also provides a breakdown of the overall contribution in specific research contributions (RCOs).

RCO 1 – RCM metamodel. This contribution, marked as 1 in Figure 2, provides a metamodel definition for RCM. The metamodel has been realized as an Ecore model, within the Eclipse Modeling Framework [6] (EMF). The definition of the metamodel comprised the addition of some modeling elements, e.g., connectors, as well as the refinements of already existing elements relations, e.g., ports and data element hierarchies.

RCO 2 – DL2RCM transformation. This contribution, marked with 2 in Figure 2, provides a model to model transformation between EAST-ADL design level metamodel and RCM metamodel (DL2RCM). The transformation has been implemented by means of a bidirectional model transformation language, namely Janus Transformation Language (JTL). To the best of our knowledge, JTL is

[6] http://www.eclipse.org

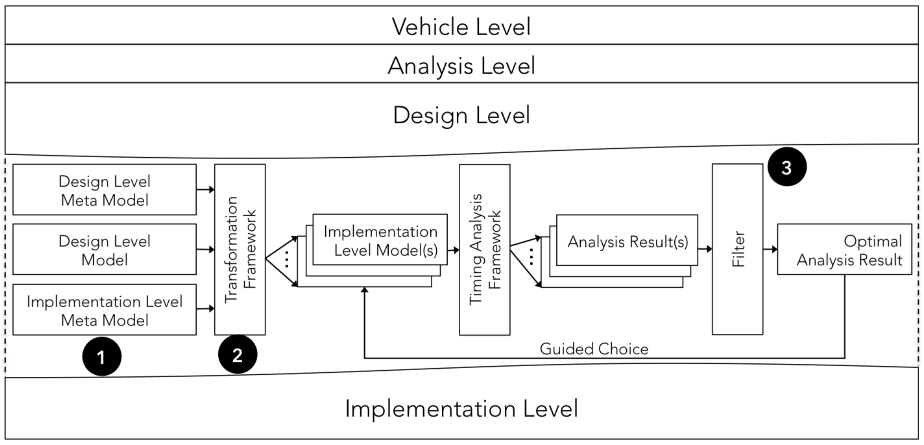
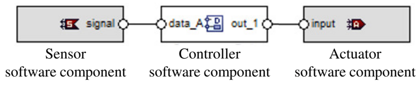


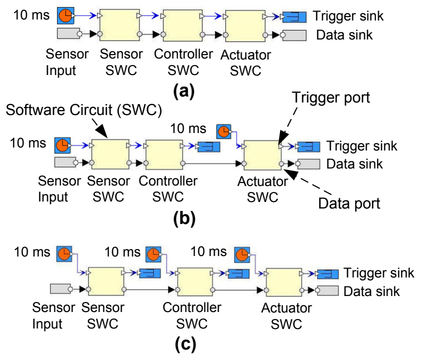
Fig. 2: Research Contributions

the only transformation language able to deal with non-bijective transformations by possibly producing multiple results. The contribution of the DL2RCM transformation is two-fold. On the one hand, it allows the automatic translation of EAST-ADL design level models to RCM models. On the other hand, it does not impose restrictions on the generation of the RCM models, i.e., it is able to generate all the possible RCM models for a given EAST-ADL design level model. For instance, given the EAST-ADL design level model depicted in Figure 3a and considering the generation of clocks in the RCM models, the DL2RCM transformation will produce the RCM models depicted in Figure 3b.



signal
data_A
out_1
input
Sensor software component
Controller software component
Actuator software component

(a) EAST-ADL Design level model of a component chain



(b) Possible RCM models of the EAST-ADL design level model of the component chain in Figure 3a

Fig. 3: Source and target model examples for DL2RCM

RCO 3 – Mechanism for the best RCM candidate selection. This contribution, marked as 3 in Figure 2, provides a conceptual mechanism supporting the selection of the best RCM model for a specific, non-empty, set of timing constraints as the last step in the process of anticipating timing analysis at design level for enabling early architecture refinements. For each generated RCM model, timing analysis is applied. Analysis results together with a non-empty set of timing constraints, are the inputs of the mechanism that checks analysis results versus timing constraints to identify, possibly the best RCM candidate implementation model. If the mechanism fails in identifying a candidate, early architecture refinements at the EAST-ADL design level model may be needed.

4 Preliminary Work and Current Status

In [7], we present the metamodel definition for RCM which focuses on the definition of metamodeling elements representing the software architecture. A work describing an extension of the RCM metamodel including the definition of new structural elements and elements used for describing timing information is currently under review at the Journal of Systems and Software[7]. In the same work, we also demonstrate the applicability of the RCM metamodel by conducting an automotive application case study. In [8] we propose a two-phase methodology which supports the extraction of timing models from EAST-ADL design-level models with the aim of anticipate timing analysis at design level. Within the proposed methodology, the software architecture at design level is automatically transformed to all the meaningful implementation-level models. The end-to-end timing analysis is performed on each generated implementation-level model and the analysis results are fed back to the design-level model. The aforesaid methodology is based on a model-to-model transformation between models conforming to the EAST-ADL design-level metamodel and models conforming to RCM metamodel (DL2RCM). The DL2RCM transformation has been implemented within the Eclipse Modeling Framework[8] using JTL [9]. The proposed methodology is concretely integrated in an industrial tool used by automotive companies. We are planning to validate the methodology by means of an industrial case study and submit the results of this validation to the International Conference on Model Driven Engineering Languages and Systems[9].

5 Validation

The works presented in [7], has been evaluated upon an industrial case-study. In [7], we demonstrate the applicability of the RCM metamodel by defining a model-to-model transformation between models conforming to the RCM metamodel and models conforming to AUTOSAR [1]. The models describe a single-node real time system composed of three components. Similarly, in [8], in order to prove the applicability of the proposed methodology, we instantiate the methodology within an industrial tool-suite, namely Rubus-ICE[10] used for the development of vehicular embedded systems. Also we are planning to demonstrate the validity of the methodology with an industrial case-study mimicking the TIMMO2USE break-by-wire validator [4].

[7] http://www.journals.elsevier.com/journal-of-systems-and-software/
[8] http://www.eclipse.org
[9] http://www.modelsconference.org
[10] https://www.arcticus-systems.com/products/rubus-ice/

6 Related Work

This section discusses some literature related to our problem and contributions.

6.1 Modeling Languages for Vehicular Embedded Systems

In this section we discuss modeling languages and methodologies specifically tailored for the development of vehicular embedded systems.

AUTOSAR. AUTOSAR [1] is an industrial initiative to provide standardized software architecture for the development of vehicular embedded systems. Within AUTOSAR, the software architecture is defined in terms of *software components* (SWCs) and *Virtual Function Bus* (VFB). VFB handles the virtual integration and communication among SWCs, hiding low-level implementation details. Unlike RCM, no particular focus was directed to specification and handling of timing-related details.

AUTOSAR metamodel describes the software development at a higher level of abstraction compared to RCM metamodel. Unlike RCM, it does not separate control and data flows among software components nor differentiate between the modeling of intra- and inter-node communication. Despite these differences, there are some similarities between AUTOSAR and RCM, e.g., the sender receiver communication mechanism in AUTOSAR is very similar to the pipe-and-filter communication mechanism in RCM. AUTOSAR is more focused on the functional and structural abstractions, hiding the implementation details about execution and communication. Whereas, RCM supports the modeling, analysis and synthesis of the execution environment of software functions. Essentially, AUTOSAR hides the details that RCM highlights.

TIMMO/TIMMO-2-USE. TIMMO [5], a large EU research project, is an initiative to provide AUTOSAR with a timing model. To this end, it provides a predictable methodology and language, called TADL [3] for expressing timing requirements and constraints. TADL is inspired by MARTE [6] which is the UML profile for the model-driven development of real-time and embedded systems. TIMMO methodology makes use of the EAST-ADL and AUTOSAR interplay. Although the TIMMO project has been evaluated upon prototypes, to the best of our knowledge, there is no concrete industrial implementation of it.

TIMMO-2-USE [4], a follow up project, presents a major redefinition of TADL and new functionality for supporting the AUTOSAR extensions regarding timing model. Although both TIMMO and TIMMO-2-USE attempt to annotate AUTOSAR with a timing model, this may be hard to accomplish as AUTOSAR aims at hiding implementation details of execution environment and communication using the VFB.

TIMMO-2-USE proposes a methodology for the software development of vehicular embedded systems solving particular issues related to the timing modeling. The methodology is based on the EAST-ADL abstraction levels and it introduces, for each level, a set of activities to perform in order to ensure the software timing requirements. Compared to the proposed solution, the TIMMO-2-USE methodology does not provide of automation nor supports early timing analysis.

6.2 Model-Driven Engineering for Vehicular Embedded Systems

In the last decades, several works investigated how to automatize the interplay between EAST-ADL and AUTOSAR through MDE. In this respect, in [19], the authors investigate the relationship between concepts of EAST-ADL and AUTOSAR. The authors use three case studies as drivers for the empirical establishment of these relationships. Nevertheless, the work only considers behavioral aspects and timing constraints from the control systems development view. Thus, it might be considered as a first step towards the automatic synthesis of an AUTOSAR architecture from EAST-ADL.

Similarly, in [10] the authors present a mapping between EAST-ADL and AUTOSAR artifacts. Nevertheless the mapping is limited only on few EAST-ADL elements and events. Furthermore, as EAST-ADL evolved, the mapping is no longer valid.

The work in [15] aims at achieving model synchronization between SysML and AUTOSAR using Triple Graph Grammars (TGG) [16]. The proposed approach is developed in an industrial project. The work represents one of the first attempts in using MDE for achieving model synchronization within the vehicular embedded domain. Nevertheless, compared to EAST-ADL, the adopted SysML profile is very generic and it does not cover most of the vehicular aspects of the software systems. On the other hand, having a more tailored language, such as EAST-ADL, would make the synchronization hard to achieve due to the underneath non-injective relations [12].

Although the work presented in this paper does not address design space exploration, the generation of candidate models and their evaluation are two of the four common steps for design space exploration. The work in [20] describes a MDE approach which allows the automatic selection of the most adequate modeling solution for application, platform, and mapping between application and platform. The approach is based on an iterative algorithm which evaluates a candidate model per iteration, until the optimum is found. With respect our solution, such an approach does not generate all the candidate models in a single execution; rather, by means of heuristics, it considers a model per iteration. Similarly to the work proposed in this paper, in [11], the authors exploit JTL, for an automatic deployment exploration technique based on refinement transformations and platform-based design. More precisely, JTL is used for realizing the so-called refinement transformations responsible for the design space exploration. Differently from our approach, the refinements transformations are endogenous, i.e., the involved source and target metamodels are the same, as the source and target models both conform to the AUTOSAR metamodel.

References

1. AUTOSAR Techincal Overview, Version 2.2.2. AUTOSAR – AUTomotive Open System ARchitecture, Release 3.1, The AUTOSAR Consortium, Aug., 2008. http://autosar.org
2. EAST-ADL Domain Model Specification, Deliverable D4.1.1, 2010. http://www.atesst.org/home/liblocal/docs/ATESST2_D4.1.1_EAST-ADL2-Specification_2010-06-02.pdf

3. TADL: Timing Augmented Description Language, Version 2, Deliverable 6, October 2009. The TIMMO Consortium
4. TIMMO-2-USE. https://itea3.org/project/timmo-2-use.html
5. TIMMO Methodology, Version 2, Deliverable 7, Oct. 2009
6. The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems, 2010. OMG Group (January 2010), http://www.omgmarte.org/
7. Bucaioni, A., Cicchetti, A., Sjödin, M.: Towards a metamodel for the rubus component model. In: 1st International Workshop on Model-Driven Engineering for Component-Based Software Systems, ModComp 2014, 29 September 2014. pp. 46–56 (2014)
8. Bucaioni, A., Mubeen, S., Cicchetti, A., Sjödin, M.: Exploring timing model extractions at east-adl design-level using model transformations. In: 12th International Conference on Information Technology : New Generations (April 2015)
9. Cicchetti, A., Di Ruscio, D., Eramo, R., Pierantonio, A.: Jtl: a bidirectional and change propagating transformation language. In: Software Language Engineering, pp. 183–202. Springer (2011)
10. Cuenot, P., Frey, P., Johansson, R., Lönn, H., Reiser, M.O., Servat, D., Koligari, R.T., Chen, D.: Developing automotive products using the east-adl2, an autosar compliant architecture description language. In: Embedded Real-Time Software Conference. Citeseer (2008)
11. Denil, J., Cicchetti, A., Biehl, M., Meulenaere, P.D., Eramo, R., Demeyer, S., Vangheluwe, H.: Automatic deployment space exploration using refinement transformations. Electronic Communications of the EASST Recent Advances in MPM(50) (Jun 2012)
12. Eramo, R., Bucaioni, A.: Understanding bidirectional transformations with tggs and jtl. Electronic Communications of the EASST 57 (2013)
13. Feiertag, N., Richter, K., Nordlander, J., Jonsson, J.: A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics. In: Procs of CRTS (2008)
14. Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke, M.: Viewpoints: A framework for integrating multiple perspectives in system development. International Journal of Software Engineering and Knowledge Engineering 2(01), 31–57 (1992)
15. Giese, H., Hildebrandt, S., Neumann, S.: Towards integrating sysml and autosar modeling via bidirectional model synchronization. In: MBEES. pp. 155–164 (2009)
16. Giese, H., Wagner, R.: From model transformation to incremental bidirectional model synchronization. Software & Systems Modeling 8(1), 21–43 (2009)
17. Graaf, B., Lormans, M., Toetenel, H.: Embedded software engineering: the state of the practice. Software, IEEE 20(6), 61–69 (2003)
18. Mubeen, S., Mäki-Turja, J., Sjödin, M.: Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study. Computer Science and Information Systems 10(1) (2013)
19. Qureshi, T.N., Chen, D., Lönn, H., Törngren, M.: From east-adl to autosar software architecture: a mapping scheme. In: Software Architecture, pp. 328–335. Springer (2011)
20. da S. Oliveira, M.F., ao, E.W.B., Nascimento, F.A., Wagner, F.R.: Model driven engineering for mpsoc design space exploration. In: Proceedings of the 20th Annual Conference on Integrated Circuits and Systems Design. pp. 81–86. SBCCI '07, ACM (2007)