Design Space Exploration for an industrial Lane-Keeping-Support Case Study

Raphael Weber, Eike Thaden, Stefan Henkler
OFFIS, Germany
{raphael.weber|eike.thaden|stefan.henkler}@offis.de

Jens Höfflinger
Robert Bosch GmbH, Germany
jens.hoefflinger@de.bosch.com

Steffen Prochnow
ETAS GmbH, Germany
steffen.prochnow@etas.com

Developing embedded safety-critical systems was and remains a challenging task. Embedded systems engineers are supposed to design solutions that satisfy company shareholders, customers, certification authorities, etc. With these many partly contradicting demands it is very hard to find an optimal or optimized solution. Under special circumstances it sometimes is even impossible to find a viable embedded system design.

Finding solutions i. e. exploring the vast design space, can be sped up by automating certain steps. This work presents our previously published semi-automated design space exploration (DSE) for safety-critical embedded real-time systems [1] applied to an industrial lane-keeping-support (LKS) case study. We thereby minimize communication, hardware costs, weight, and the number of processing units also satisfying hard real-time constraints for distributed embedded systems.

In our tool demonstration we present how we model a lane-keeping-support system consisting of multiple functions like “line detection” or “situation evaluation” using the SPES methodology from the SPES_XT project [2]. Using this methodology for the case study we started out with a functional structure (see Fig. 1) and refined our model. The functional *chain* starts at the video sensing unit which produces an uncompressed 2 mega-pixel 15 frames per second video stream. This video stream is used to extract line information which may indicate lanes of a street, so the data workload between video sensing and line detection is quite high. From there, rather small data amounts are passed down the chain, picking up additional sensor data along the way. After the trajectory planning there is a breakdown in different options, depending on which hardware is actually used in the system (either ESP or EPS; neither ESP nor EPS; both ESP and EPS). In our case study we considered the latter.

In subsequent steps we derived a logical component structure from the functional view. The logical structure included explicit distinctions between what is a logical *task* and a logical *signal*. In a last refinement step we created a hardware architecture and deduced software tasks and technical signals/messages from their logical counterparts. The mentioned hardware structure is represented by a typical generic bus-based hierarchy consisting of one global time division multiplex (TDMA) based backbone bus which is connected via gateways to local cluster buses (e. g. CAN, FlexRay, or MOST).

While the architectural design was done using the SPES methodology we still had to include certain aspects relevant for an automatic DSE in our model. Therefore, we used timing property extensions based on the timing augmented description language (TADL) version 2 from the TIMMO-2-USE project [3]. Further extensions include the annotation of execution times and memory consumption data of tasks for different ECU types. This way, we can allow (well defined) variations

This work was supported by the BMBF funded project SPES_XT.

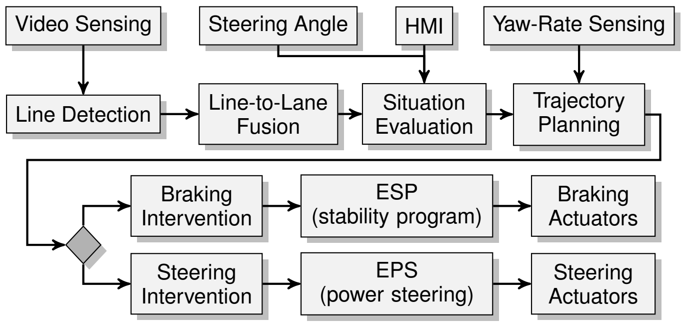

Fig. 1. Functional structure of the LKS case study.

in the underlying hardware architecture while ensuring that all timing requirements are met.

With the special hardware architecture in mind the DSE can be performed more efficiently compared to holistic approaches. We divide the NP-hard DSE problem in a global analysis minimizing communication on the backbone bus using a heuristic and multiple local analyses where a cost-minimal extension of the hardware architecture for a valid deployment of the task structure is computed. A valid deployment is characterized by a mapping of the tasks and their communication to the ECUs and buses such that all timing requirements and mapping constraints are met.

Satisfying constraint bounds and all given hard real-time requirements is one thing. Enhancing the design targeting one or even multiple optimization objectives is another. In our demonstrator we show how we automatically compute optimal solutions for minimizing costs, weight, and the number of ECUs.

For the embedded system design of the LKS we had to deploy 17 tasks along with 14 communication signals to the hardware architecture. Each optimization objective yielded one solution – an overview is shown in TABLE I. Future work on the DSE will allow for prioritizing or weighing the optimization objectives.

TABLE I
RESULTS OF OUR DSE FOR THE LKS.

Opt. objective ↓	**Costs (€)**	**Weight (g)**	**# of ECUs**
min costs	89	1030	5
min weight	90	930	4
min # of ECUs	111	1050	3

REFERENCES

[1] M. Büker, G. Ehmen, S. Henkler, A. Rettberg, I. Stierand, and E. Thaden, “From Matlab-Simulink to Distributed Embedded Applications: An Automotive Tool Demonstration,” in *Proceedings of DATE Conference – University Booth*, 19 - 22 Mar. 2013.

[2] M. Broy, W. Damm, S. Henkler, K. Pohl, A. Vogelsang, and T. Weyer, “Introduction to the SPES Modeling Framework,” in *Model-Based Engineering of Embedded Systems*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 31–49.

[3] TIMMO-2-USE Project, “TIMMO-2-USE D11,” 2012. [Online]. Available: http://www.timmo-2-use.org/deliverables/TIMMO-2-USE_D11.pdf