Learning Early-Stage Platform Dimensioning From Late-Stage Timing Verification

Kai Richter, Marek Jersak
Symtavision GmbH
Braunschweig, Germany
{richter,jersak}@symtavision.com

Rolf Ernst
Institut für Datentechnik und Kommunikationsnetze
Technische Universität Braunschweig
Braunschweig, Germany
ernst@ida.ing.tu-bs.de

Abstract— **Today's innovations in the automotive sector are, to a great extent, based on electronics. The increasing integration complexity and stringent cost reduction goals turn E/E platform design into a challenging task. Timing/performance is becoming a key aspect of architecture design, because the platform must be dimensioned to provide just the right amount of computing power and network bandwidth, including reserves for future extensions, in order to be cost efficient. In other words, it must be as powerful as needed but as cheap as possible. Finding this sweet spot is a key challenge. Therefore, OEMs and Tier-1 are in search of new methods, processes, and timing analysis techniques that assist in early platform design stages. In this paper, we demonstrate how some selected techniques that are established for verification (in late design stages) can also be used to guide the design (in early stages). We present examples in the areas ECU (OSEK), buses (CAN, FlexRay) and gated networks. Flow and applicability aspects are highlighted. As a key result, we show that and how we can learn from late-stage verification for early-stage design. Finally, we also outline future challenges in the area of multi-core ECUs.**

I. Introduction

Today's innovations in the automotive sector are, to a great extent, based on electronics. The increasing integration complexity of automotive systems, the many vehicle variants which a single E/E platform must support and stringent cost reduction goals turn E/E platform design into a highly challenging task. Electronics cost is determined by many factors, including topology and the processing and communication technologies used. Although timing/performance is just one aspect of architecture design, it is a key one. The platform must provide just the right amount of computing power and network bandwidth, including reserves for future extensions, in order to be cost efficient. Finding this sweet spot, where neither reliability nor extensibility are sacrificed while keeping cost to a minimum, is a key challenge. This means it has become clear that timing and performance issues must essentially be included in the early design stages. The question is: How?

OEMs and Tier-1 are in search of new methods, processes, and timing analysis techniques that assist in early platform design. A ***re***-volutionary early-stage solution might appear some day. More likely is an ***e***-volution of existing timing and performance analysis technologies that are established in later-stage verification already and can be transferred to earlier stages. In fact, we can learn a lot from the way later-stage timing verification has found its place, and finally transfer key concepts and techniques also to earlier design stages. In other words, we put the cart before the horse.

In this paper, we highlight the technology transfer from late stages to early stages using examples from ECU (OSEK), bus (CAN), and gated network design from real-world industrial applications of timing and scheduling analysis. We specifically look at the flow and process aspects as well as on the amount and detail of data needed for the analysis, which is typically critical in early stages. This procedure has proved to be effective in practice.

The paper is organized as follows. The next section outlines the envisioned design flow, separated into "learning from verification of existing platforms" and "applying to early design of new platforms". In Section 3, we summarize three projects in which this approach has been applied in industry for ECU hardware selection, bus design, and network dimensioning. Section 4 focuses on the integration and interfaces between the different players (OEMs and Tier-1s, function and network designers, etc.). In Section 5 we provide an outlook on future multi-core ECUs. Finally, in Section 6 we draw our conclusions.

II. From Late Verification to Early Design

Before learning from late-stage verification, we summarize the most prominent techniques and discuss their applicability. The most prominent late stage timing verification techniques are:

- testing, i.e. measurement and tracing (on target or prototype),
- timed simulations (on detailed target model),
- worst-case execution time analysis (on target object code), and
- scheduling analysis (on abstract target model).

This document is based on the TIMMO project in the framework of the ITEA 2, EUREKA cluster programme Σ! 3674, funded by the German Ministry for Education and Research (BMBF) under the funding IDs 01IS07002(B-I,K). The responsibility for the content rests with the authors.

Not all of them are directly suitable for analysis at an early platform design step, simply because their input data is not (yet) available. In this context, abstraction, accuracy, and information interfaces are important.

A. Abstraction

On the one hand, a technique must be able to cope with situations in which much less data is available as it is in a typical late-stage verification situation. On the other hand, the techniques must keep their key expressiveness and focus on the effects they consider. This means, the abstraction must not ignore the nature of the effects to be analyzed.

B. Accuracy

As a second requirement, the successful techniques must also provide valuable results with only little info on final implementation, i.e. without detailed task layout, detailed signal-to-frame mapping, and finalized gateway strategy. This includes reasoning about the accuracy of the results. Interestingly, the absolute accuracy is not so far an issue as the relative accuracy is when e.g. comparing two platform candidates.

The examples in this paper start at verification and gradually "reduce" the system models such that they are applicable earlier with valuable results.

C. Interfaces

A third requirement is that the techniques must be compatible with established industry supply chains. This means they must support interfaces that allow exchanging information about components without disclosing too much IP. These interfaces must also be applicable across company borders and let OEMs, Tier-1 and other suppliers to exchange key information about the model components.

Of similar importance are interfaces that support the transitions between early design, middle implementation, and late verification, because timing and performance are aspects that are best considered through the entire design process. Clearly, a seamless transition and refinement strategy is desirable.

D. The Learning Process

Let's briefly consider how well each technique fulfills these requirements.

Testing is not applicable in the early stage because no implementation is available and the target has not been chosen (the very definition of 'early stage').

Timed simulation partially avoids this problem because no target is necessary. However, to produce meaningful results the simulated target model has to be highly accurate, and simulation always requires executable code. Both conditions can be difficult to meet in the early stage. In fact, timed simulation is most successful in SW-development either for a platform which has been chosen but is not yet available in HW, or because the superior debugging capabilities of the simulator are needed.

Scheduling analysis, in contrast, is abstract by nature. It works at the level of the operating system or bus protocol and analyses timing effects resulting from integration of tasks and messages on controllers and buses, respectively. This focus is key, because there are rare cases in which a single task or message causes performance problems. Much more often, their integration creates bottlenecks.

Scheduling analysis considers integration without the need for an executable system model. This makes scheduling analysis a promising candidate for being applied also in the early design stage. Scheduling analysis is applicable to system models of very different level of detail. In addition, scheduling analysis allows for "what-if analysis", i.e. the analysis can be applied to a hypothetic data set without any coding, building, wiring, or other implementation work to be done. If put into the right context, this can tell about possible performance reserves, optimization possibilities, or upcoming bottlenecks *before* the entire system is build.

Obviously, such "what-if analysis" can tell, if or not a platform provides enough computation and/or communication performance to realize the envisioned application. This way, it directly supports the exploration of different possible design decisions and variants in platform design.

This is illustrated in the V-model view of Figure 1. In addition to the late-stage verification of a known platform (step 1: top-right of the V-model), there are two more application areas of earlier-stage "what-if scheduling analysis":

- when an existing platform is to be extended, modified, or optimized (step 2)
- when a new platform is being build (step 3)

The next section briefly outlines the techniques behind scheduling analysis. After that we give concrete examples from customer projects, namely from Daimler [1], Volkswagen [2] and EDAG [3]. The examples are from different domains (ECU, bus, and network) but have one key aspect in common. All projects started at the verification (top-right in the V-model) and then gradually transferred the experience to earlier steps in the process.

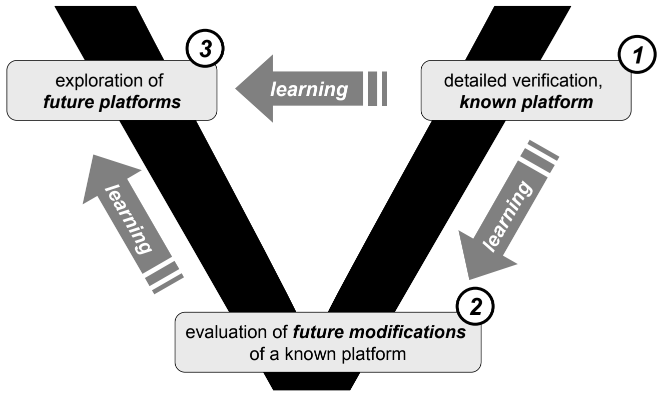

Figure 1. Learn from verification, exploit in design

III. Scheduling Analysis – Technology Sketch

Scheduling analysis has its roots in the early 1970's [4]. The basic objective is to map the resource sharing strategy (OS scheduling or bus arbitration) into a set of equations that can be solved to determine latency and response times [5], peak loads, buffer sizes, etc., thereby carefully considering the corner cases of system execution. Several popular sharing mechanisms for ECUs and buses have been extensively researched. OSEK [6], for instance, is based on static priorities with preemptive, deferred, and non-preemptive tasks [4][7]. Extensions exist to account for fine-grain OS timing influences such as offsets [8][9] and sporadic interrupts [10]. In a similar way, CAN [11] networks can be described by non-preemptive static-priority techniques [7]. FlexRay [12] Networks can be described by TDMA analysis [13]. Recently, new techniques for system-level end-to-end analysis were introduced [14][15][16]. This way, techniques for analyzing a large amount of real-world OS and arbitration mechanisms exists, most of them integrated into SymTA/S [17].

In all mentioned contributions (and many more that can simply not be mentioned all here), the basic idea is to create a formal (i.e. mathematical) model that a) covers the relevant load arrival into the system (tasks activation, interrupts, etc.) and b) captures the corresponding reaction into equations for certain properties (latency, buffer sizes, etc.). This clear focus on timing-relevant influences resulted in very small and efficient models with only few parameters. Examples are function execution times or frame cycle times. Neither details of the executed code, nor the content of the transmitted data, is of interest, as long as the "arrival of system load" can be described.

This powerful abstraction into few key parameters is the key enabler for both, the systematic and efficient analysis of timing and performance including full corner-case coverage, and the early application by means of "what-if" analyses for sensitivity analysis and design space exploration [18][2][19].

The SymTA/S analysis framework that has been used in the three commercial case studies is comprised of the key research results from three decades including the cited contributions.

IV. Case Studies

A. ECU Hardware Selection

Volkswagen Steering Systems has applied SymTA/S scheduling analysis during the development of its new electro-mechanical steering system for the Volkswagen Tiguan [2]. The required safety concept was developed according to the functional safety standard IEC 61508. One of the safety mechanisms is a watchdog on a second processor that supervises the internal timing of the software through a request-response mechanism. If the response signal does not arrive in a predefined timeframe, the system is assumed to have failed, and the fail-safe mode is activated. This means that every late delivery of control signals is interpreted as a system failure, even if it is just a consequence of a temporary peak load on the CPU and would not harm the overall function at all.

1) *Verification*

Volkswagen has verified the timing using the SymTA/S tool framework. This verification included scheduling analysis on the main controller and end-to-end timing analysis for the request-response mechanism. This way, Volkswagen eliminated false positives in failure detection and maximized the availability of the steering system. In a similar setting, SymTA/S was also used at BMW [20].

2) *Hardware Selection*

During the design of this system, Volkswagen could re-use an existing SymTA/S verification model from the previous generation of the steering system. Due to cost reasons, much of the system software was re-used unchanged, so also large parts of the SymTA/S model were already available at project start. And the design group had experience with timing and scheduling analysis already.

Based on this re-used SymTA/S model, Volkswagen "virtually verified" several software and hardware modifications with SymTA/S before key decisions were taken. Such sensitivity analysis, a special type of "what if analysis", was used to detect unused CPU performance and to select the slowest (cheapest) possible CPU hardware. At the same time, Volkswagen guaranteed the desired function availability according to IEC 61508.

The key parameters for this virtual verification were worst-case execution times (WCET) of functions and CPU speed values. It was shown that, after some initial learning, these could be used as a sufficient communication base between software developers, ECU integrators and hardware suppliers. As a result of the "what-if scheduling analysis", Volkswagen could safely decide for a much slower processor that initially envisioned, this way saving a significant mount of hardware cost.

It must be noted that full accuracy was not required in the early estimations, because the entire system was continuously re-analyzed as soon as more implementation details became available. In other words, the verification itself was seamlessly integrated into the design process, while the accuracy was constantly increasing, from first estimates to the final target.

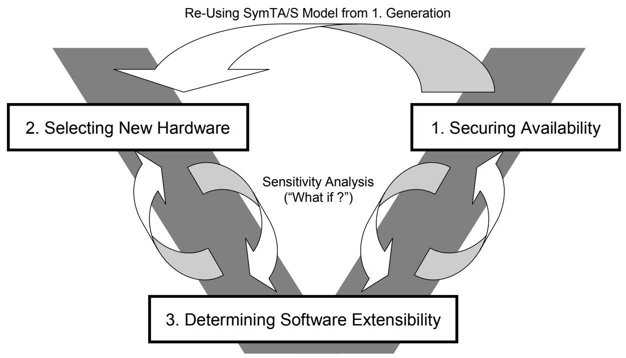
Re-Using SymTA/S Model from 1. Generation
2. Selecting New Hardware
1. Securing Availability
Sensitivity Analysis ("What if ?")
3. Determining Software Extensibility

Figure 2. SymTA/S „What-if" Analyses at Volkswagen Steering Systems – *Figure courtesy of Volkswagen*

3) *Future Extensibility*

Finally, the sensitivity analysis also helped in deliberately preserving some amount of CPU power for future software updates and extensions. Volkswagen considers having the abstract scheduling analysis models a key advantage for a systematic extension of existing systems, and as a solid argumentation base for the negotiation with suppliers [2].

4) *Bottom Line*

After collecting experience from verification, Volkswagen has used "what-if scheduling analysis" extensively. The communication with software developers and hardware suppliers could be efficiently done through few scheduling model parameters. Using estimates and a reduced amount of information, Volkswagen could systematically reserve timing budgets for future extensions. The systematic planning of the next-generation ECU platform secured significant cost savings. The three main steps are illustrated in Figure 2.

B. *CAN Bus Extension*

The second example is from EDAG, a leading German engineering house. In a project for EDAG's OEM customers, a new ECU "Lane Departure Warning" was to be added to an existing low-speed CAN bus in a mass production car. The key question was, if (and how) the existing bus could be safely extended to carry the additional messages of „Lane Departure Warning (LDW)" without critical influences on the existing communication [3].

1) *Verification.*

EDAG has set-up a SymTA/S scheduling model of the existing CAN bus and performed several analyses to understand better the peak load situations in which additional communication becomes most critical. This provided EDAG with new insights into the bus under design.

2) *Extension*

Then, the new LDW frames were virtually added to the SymTA/S model before the ECU was available for being integrated into a lab prototype. EDAG has performed more scheduling analyses in order to find out, in which configuration the new frames have the smallest influence on the existing bus. This "frame design" is a typical platform design task that involves signal-to-frame mapping, CAN Id (priority) selection, the assignment of a sufficient frame period, the selection of a reasonable offset, etc. All these decisions influence the timing in one or the other way.

The result of this assessment was that, from a timing and performance perspective, the new frames could be safely added to the bus. Figure 3. illustrates the enhancements in the overall performance of the communication.

3) *Robustness Against Dynamic Load*

Finally, EDAG has further performed a larger number of "what-if" scheduling analyses to determine the influence of different dynamic load situations on the timing of frames. In order to apply "realistic" dynamic load to the bus, several existing bus traces (from verification) were used and key abstract properties were extracted. These could then be re-used and applied to other "virtual" bus models, thereby simulating the external dynamic environment of a not jet build communication platform.

This has significantly increased EDAG's understanding of the bus performance in the presence of temporary overload and/or bus errors. The key parameters that were considered were signals and their timing requirements (such as rates and, if available, deadlines) and the allowed CAN Id (bus frame priority) ranges that were available for bus extensions. For the error robustness, EDAG considered different possible bus error models including periodic errors, sporadic errors and burst errors; all on a "virtual" SymTA/S bus scheduling model.

4) *Bottom line*

Like in the Volkswagen case, EDAG has learned from verification, how to use abstract analysis and how to perform "what-if" scheduling analysis with SymTA/S. EDAG explored different options for integration the new LDW ECU to an the existing low-speed CAN bus *before* the key decisions were taken, *how* the new ECU was added. Finally, EDAG determined the general extensibility of the existing CAN bus, and its timing robustness against dynamic overload, e.g. due to bus errors.

C. *Gated Network Dimensioning*

Daimler Research goes one step further. They are using SymTA/S not only for analyzing and dimensioning individual buses but for their entire network design covering different topologies of CAN and FlexRay buses and gateways [1]. In contrast to the two aforementioned examples, this example does not have a specific application in mind. It rather aims at methodological progress in the context of network topology design for next-generation car platform of Mercedes-Benz.

1) *Detailed Analysis of Existing Networks*

Daimler started by setting up a detailed SymTA/S scheduling analysis model of an existing network architecture. In the common project, Symtavision and Daimler developed an interface between SymTA/S and the central Daimler architecture database. The database is realized by the PREEvision tool from Aquintos (http://www.aquintos.com). This way, most of the bus configuration information can be efficiently imported into SymTA/S without manual modeling.

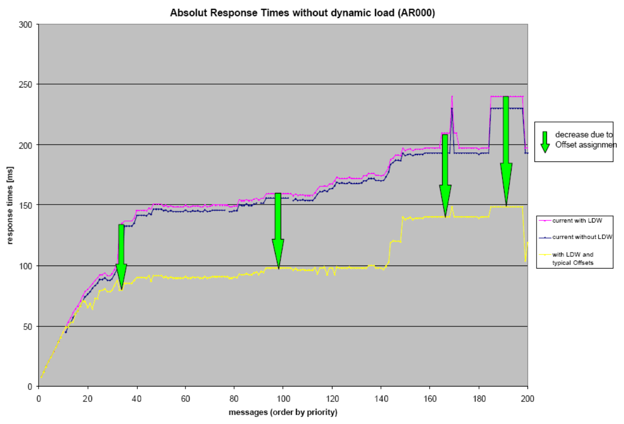

Figure 3. Bus Latency Times Before and After Optimization – *Figure courtesy of EDAG*

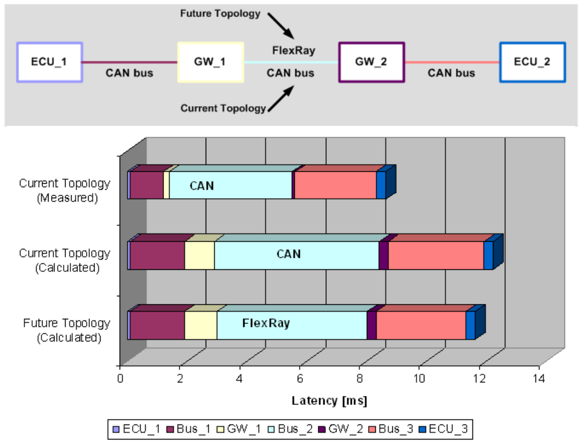

Figure 4. Example for End-to-End latencies (for one particular frame) – *Figure courtesy of Daimler*

The SymTA/S analysis results were then compared to measurements that were taken on the corresponding real buses. This provided good knowledge on the comparability of measured data and the "calculated" scheduling analysis results. Of course, the calculated values were higher than the measured ones for several reasons. Most importantly, scheduling analysis tends to be a bit pessimistic (to reduce modeling complexity) while measurements are always too optimistic (because they never reach all corner cases). A difference of 30% is not uncommon for network latencies. The detailed timing diagrams of SymTA/S illustrate which situation leads to one or the other latency, so the designer can distinguish insufficient modeling details from a clear measurement coverage problem. This way, Daimler derived simple but expressive metrics on how to use SymTA/S results when comparing network architectures.

2) Analysis of Future Networks

Next, Daimler set up a SymTA/S model of a network architecture that was already envisioned for future cars but not yet build. The configuration data was imported from PREEvision and SymTA/S was used to determine the timing of this not-yet-build network. Daimler was able to set-up a SymTA/S model that based on a small set of configuration parameters that a) are available early (or can be made available with minor invasion of established process) and b) are sufficiently expressive to reach the desired level of result confidence for taking architecture and topology decisions.

In fact, the result accuracy is not sufficient for a final verification (as it was in the Volkswagen ECU case above), but it was sufficiently good for platform decisions. The particular experience made during the detailed analysis of existing networks (late-stage analysis), and the derived comparison metrics mentioned above, are key enablers for applying timing analysis techniques earlier in the process with high result confidence.

3) Comparisons

Figure 4. shows a comparison of the measured and calculated (SymTA/S) end-to-end network latency of one particular frame on an existing CAN bus and the projected latency of that same frame when one bus in the topology is exchanged by a FlexRay bus segment.

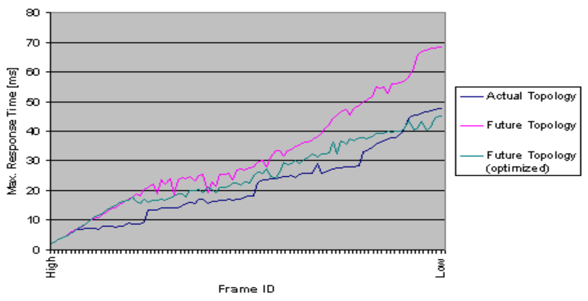

Figure 5. Comparison of latencies in a CAN cluster – *Figure courtesy of Daimler*

In a second comparison, Daimler focused on the latency increase when more frames are added to the CAN bus (without exchanging it by FlexRay), and also explored (with the SymTA/S exploration framework) several options for optimizations. Figure 5. shows the response time profiles of three bus configurations: "Actual topology" is the current latency profile of all frames, "future topology" indicates a huge increase in bus latencies as a consequence of future bus extensions (more frames), and "future topology optimized" shows that this future load can be optimized significantly.

Such comparisons help deciding if and when to transition from Can to FlexRay because of the increasing bus load, or when to postpone this costly technology step.

4) Bottom Line

From detailed scheduling analysis and from comparisons with measurements, Daimler has created competence in network timing covering multi-bus topologies with gateways and determined the latencies of future car networks, allowing them to take performance- and timing-optimized platform decisions.

This example shows once more that the proposed approach "exploit verification experience during early design" is effective in practice. And it shows which technology can be transferred from late to early process steps (here: SymTA/S scheduling analysis), and which can't (here tracing and measurements). As another side effect, one exploits many late-stage fine-tuning and optimization possibilities.

Daimler concludes that "the project has brought great value for the knowledge about timing of network architectures." [1].

V. System Integration

So far, we have considered individual views on early system design. Tier-1s look at a single ECUs (as in the Volkswagen example), individual OEM domain departments look at one bus (as in the EDAG example), and the OEM system department looks at the entire E/E network (Daimler example).

Another challenge is that these players will have to exchange information, especially requirements and guarantees about their sub-systems or components. This was indicated in all three projects:

- ECU extension needs estimates of software execution times (WCET) of own and supplied software
- Bus extension needs info on signal rates and maximum allowed jitters (typically come from the control engineers and can be in-house or at Tier-1s)
- E/E Platform design needs, in addition to information on the combined communication requirements, also info and control over the gateway strategy.

In the mentioned projects, we have used and extended established standards such as DBC, FIBEX, OIL to the needs in the project; and we developed interfaces to other configuration tools such as PREEvision. On the long term, standardized ways to exchange timing information are required. There are two relevant ongoing activities.

First, the EU-funded industry-driven TIMMO project [21] aims at defining a Timing Augmented Description Language (TADL), along with a detailed methodology when and how to exchange such TADL models. Several OEMs and Tier-1 suppliers develop TIMMO together with key software and tool vendors (incl. Symtavision) and two universities.

Secondly, timing has also become a hot topic in AUTOSAR [22]. The existing AUTOSAR templates are being enriched by a timing view that supports predicting, analyzing and verifying ECU and system timing along the entire process. Symtavision has recently become an AUTOSAR development member and contributes its expertise to the development of these necessary AUTOSAR extensions in work package In WPII-1.2.

The SymTA/S analysis techniques do support such interfaces in terms of appropriate model parameters and the possibility to include black-box models of parts of the system with very small model footprint [23]. Focusing on the right data is crucial for an efficient and effective cooperation between OEMs and Tier-1s [24].

VI. Multi-Core Architectures

Currently, most ECUs contain a single core CPU possibly enhanced with specialized peripheral processors to offload some of the time critical and control functions. In the future, multi-core ECUs with homogeneous cores, with shared memory and uniform or non-uniform memory access will emerge. An obvious application is the merging of several ECUs into a single unit. This move, enabled by AUTOSAR, will improve function integration as in body electronics or sensor fusion, and it will save cost and improve maintainability. Here, virtualization is important, i.e. keeping task execution isolated, yet profiting from on-chip buses and memories as an efficient means to implement AUTOSAR SW component communication.

In this type of application, multi-core CPUs turn an ECU into a highly integrated "networked system" micro cosmos that (at first glance) appears similar to networked systems as of today. However, while virtualization works for functionality, sharing chip resources leads to further timing dependencies that go beyond networked systems, since program and data memory accesses use the same chip infrastructure. Memory accesses are often less regular and predetermined and require new analysis algorithms as presented in [25]. The algorithm is compatible to SymTA/S.

A second type of application will be high-performance domains such as engine control or advanced driver assistance. In this case, the multi-core architecture will be used to distribute the load of a single application over several cores. This also turns the shared memory communication of today into a distributed communication between several cores, which might share function units, such as coprocessors.

Traditionally, save sharing and communication in automotive task systems uses OS supported locking and synchronization mechanisms. When mapping such task systems to multi-core architectures, new types of arbitration conflicts emerge that require enhanced scheduling and analysis algorithms. Examples are shared buses and caches, introducing critical side effects leading to deviating results [26] that are hard to predict. Therefore, we expect that automotive multi-core scheduling will take an evolutionary approach extending the current fixed priority scheduling towards partitioned fixed priority scheduling, such as MPCP [27] or the algorithm by Chen [28]. In [29], a first algorithm is presented that is compatible to SymTA/S and can be combined with memory access analysis as described above. It can be used to analyze such evolutionary multi-core scheduling extensions.

Some more research is needed to also cover later multi-core generation issues, such as cache coherence, multithreading, and migration, as well as global scheduling algorithms that are discussed in the research community. A main issue for such architectures will be predictability.

VII. Conclusion

In this paper, we have shown that and how timing analysis techniques can assist in developing cost efficient automotive E/E platforms. In particular, we have demonstrated how scheduling analysis that is established for verification (in late design stages) can also be used to guide the design (in early stages), because it meets some particular requirements in terms of abstraction, accuracy, and interfaces. This allows finding the sweet spots with respect to performance and cost.

We outlined three industrial projects, in which our customers use SymTA/S scheduling analysis, in particular "what-if" analysis, for early-stage platform performance assessment as one important step in platform design. In all three projects, however, this early application of scheduling analysis was preceded by using scheduling analysis for detailed platform verification in late stages. The experience made in verification enabled our customers to exploit the abstraction possibilities of the technology and adopt and reduce the models such they could be used for "what-if analyses" in several ways, from software extensibility analysis, to CPU selection, to bus configuration, to topology decisions.

We ultimately believe that gathering late-stage experience is the key enabler for adopting new technology also to early stages. This is not an automotive-only observation. Clearly, the same flow can be expected in aerospace, multi-media, etc.

REFERENCES

[1] M. Traub, V. Lauer , J. Becker , M. Jersak, K. Richter , M. Kühl. Using timing analysis for evaluating communication behavior and network topologies in an early design phase of automotive electric/electronic architectures. SAE World Congress, Detroit, April 2009

[2] T. Jablonski, C. Busse, D. Brinkema, M. Jersak, K. Richter. Timing Analysis as a Safety Case. Hanser Automotive Magazin, 11.2008

[3] M. Girlach, Usage of SymTA/S for investigation of the K-Matrix in a vehicle development project. 1st SymTA/S News Conference, Braunschweig, Sept. 2007

[4] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 20(1):46–61, 1973.

[5] M. Joseph and P. Pandya. Finding response times in a real-time system. The Computer Journal, 29(5):390–395, 1986.

[6] OSEK Steering Committee. OSEK Open systems and the corresponding interfaces for automotive electronics. http://www.osek-vdx.org/.

[7] K. Jeffay, D. F. Stanat, C. U. Martel. On Non-Preemptive Scheduling of Periodic and Sporadic Tasks. IEEE Real-Time Systems Symposium, San Antonio, Texas, December 1991

[8] K. Tindell. Adding time-offsets to schedulability analysis. Technical Report YCS 221, University of York, 1994.

[9] J. C. Palencia and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. In Proceedings of the IEEE Real-Time Systems Symposium, page 26. IEEE Computer Society, 1998.

[10] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard real-time systems. Journal of Real-Time Systems, 1(1):27–60, 1989.

[11] CAN in Automation Website. http://www.can-cia.org

[12] FlexRay Consortium Website. http://www.flexray.com/

[13] H. Kopetz and G. Gruensteidl. TTP - a time-triggered protocol for fault-tolerant computing. In Proceedings 23rd International Symposium on Fault-Tolerant Computing, pages 524–532, 1993.

[14] L. Mangeruca, M. Baleani, A. Ferrari, and A. Sangiovanni-Vincentelli. Semantics-preserving design of embedded control software from synchronous models. IEEE Trans. Software Eng., 33(8):497–509, 2007.

[15] S. Matic and T. A. Henzinger. Trading end-to-end latency for composability. In RTSS '05: Proceedings of the 26th IEEE International Real-Time Systems Symposium, pages 99–110, Washington, DC, USA, 2005. IEEE Computer Society.

[16] N. Feiertag, K. Richter, J. Nordlander, J. Jonsson. A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics, IEEE RTSS 08, Workshop Compositional Theory and Technology for Real-Time Embedded Systems, Barcelona, December 2008.

[17] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter and R. Ernst. System Level Performance Analysis – the SymTA/S Approach. IEE Proceedings on Computers and Digital Techniques, Vol 152, issue 2, March 2005.

[18] A. Hamann, M. Jersak, K. Richter, R. Ernst. A framework for modular analysis and exploration of heterogeneous embedded systems. Real-Time Systems, volume 33, pages 101-137, July 2006.

[19] R. Racu, A. Hamann, R. Ernst. A Formal Approach to Multi-Dimensional Sensitivity Analysis of Embedded Real-Time Systems. Euromicro Conference on Real-Time Systems, Dresden, Germany, July 2006

[20] M. Jersak, K. Richter, H. Sarnowski, P. Gliwa. The Right Timing Analysis Tools Increase Safety and Productivity. ATZe worldwide Edition: 2009-01

[21] ITEA-2 Project TIMMO. http://www.timmo.org

[22] AUTOSAR Partnership. http://www.autosar.org/

[23] K. Richter and R. Ernst, Event Model Interfaces for Heterogeneous System Analysis, In Proceedings of Design, Automation, and Test in Europe Conference, Paris, France, 2002.

[24] K. Richter, How OEMs can get Suppliers On Board for Designing Extensible Networks. In Proceedings Embedded World Conference, Nuremberg, 2007.

[25] S. Schliecker, M. Negrean, G. Nicolescu, P. Paulin, R. Ernst. Reliable Performance Analysis of a Multicore Multithreaded System-On-Chip. In Proc. 6th Int. Conf. on Hardware Software Codesign and System Synthesis (CODES+ISSS), Atlanta, USA, Oct. 2008.

[26] B.B. Brandenburg, J.M. Calandrino, J.H. Anderson. On the Scalability of Real-Time Scheduling Algorithms on Multicore Platforms: A Case Study. Real-Time Systems Symposium, Barcelona, Dec. 2008.

[27] R. Rajkumar. Synchronization in Real-Time Systems: A Priority Inheritance Approach. Kluwer Academic Publishers Norwell, MA, USA, 1991.

[28] C.M. Chen and S.K. Tripathi. Multiprocessor priority ceiling based protocols. Tech. Rep. Univ. of Maryland, 1994.

[29] M. Negrean, S. Schliecker, R. Ernst. Response-Time Analysis of Arbitrarily Activated Tasks in Multiprocessor Systems with Shared Resources. In Proc. Design, Automation, and Test in Europe (DATE), Nice, Apr. 2009.