Model-Driven Performance Optimization Tool Platform for Multi-core Systems with Open Sources Technologies

Syed Aoun Raza
Robert Bosch GmbH
E-mail:Aoun.Raza@de.bosch.com

1 Abstract

With the advent of multi-core ECUs and hardware fusion in the automotive domain, the tooling environment to support multi-core software development has gained significance. Especially, tools that can provide an early indication about the architectural behavior before the existence of the code. Such domain specific tool platforms, which enable analyses (e.g., data-consistency) and optimizations (memory management, task-to-core mapping, timing simulations and distribution) are not easily available. The commercial solutions available on the market cannot be applied generally with off the shelf optimization options. Therefore, at Bosch we have developed a tool platform that bridges the open source and commercial solutions to support analysis and optimizations of multi-core system development. This paper provides an overview of the different optimization use cases and developed tool platform.

2 Introduction

The motivation behind this work was to establish a common tooling platform solution for Bosch productive system development, where several existing and future development use cases for multi-core analysis can be synergistically combined. We started with a vision and strategy that this tooling should focus mainly on automotive sector specific scenarios and support multi-core developers in all system development steps i.e., from different abstraction levels. After analyzing the requirements and development use cases, we identified several scenarios and divided them into following two categories

- Legacy system migration from single- to multi-core
- Green Field Approach (GFA) or development from scratch

In the first category, the main focus is to keep the system running with a gain in performance from the multi-core hardware architectures. Further, once migrated the legacy system should not become a victim of classical multi-core system problems such as data-inconsistency and deadlocks. On the other hand, there can be a tendency to be conservative and over-protect shared data during the migration to multi-core hardware architecture. This can lead to resource usage bottlenecks, because the legacy architecture design lacks the initial multi-core system design and development mindset.

Second category provides extensive opportunities to utilize the multi-core hardware architectures efficiently. As development from scratch has more freedom and flexibility for new concepts on system architecture level to develop a behavior that can utilize the capabilities of the new hardware architectures. However, if we can extract a higher-level of abstraction from legacy systems, the solution and opportunities becomes nearly same for architecture designers and analysts. Figure 1 depicts, how scenarios from both categories can be combine to achieve maximum benefit.

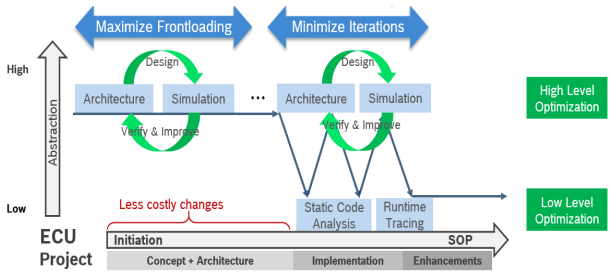

Figure 1: Identified use cases

In the next sections, we shall briefly discuss main uses cases and tool solution platform to support the software development in both categories.

3 Use Cases

Currently, the automotive system softwares already support parallel execution with the help of concurrency and multi-tasking concept on a single-core microcontroller for hard real-time. However, to exploit the multi-core hardware architecture these concurrency concepts are not fully optimized and needs to be restructured for extensions. Additionally, the first challenge is to achieve a performant allocation of the system to the cores in such a way that the existing program does not require any change. The solutions must be scalable to support a potential future hardware architectures with increased number of cores. This decision is complex and must take into consideration available knowledge on the computations to avoid any decrease in quality or reliability of the system. Therefore, this process need proper tool infrastructure to support extracting the requirements from the legacy

system and re-analyze the system architecture. Similarly, the systems which are designed from the scratch need to keep the growing nature to hardware architecture cores and introduce the flexibility for future migrations. Hence, we see two major activities for us to support such systems development where architects need support from the tools and methodologies, these are detailed in the next subsections.

3.1 Software distribution

The initial step toward a multi-core system design involves the software distribution according to the underlying hardware architecture. This means the architects should be able to design system component and get an early response of core-load distribution. This can be achieved with current model-based simulation tools such as Timing Architect (TA) or INCHRON ChronSIM. Therefore, this acts as a first major challenge in development of the flexible infrastructure to use simulation tools.

3.2 Core-load improvements

Secondly, after the system has been distributed on multi-core hardware and optimized for timing and data consistency requirements. There are still possibilities to improve the system in terms of memeory accesses. As every core spends a significant amount of time for data and code accesses. The goal of core-load optimization is to allocate data and code to the available memories in a way that the overall performance of the system is optimized. In other words, the cross core loads (CPU loads that stem from cross-communication of coreX to coreY-local memories) are reduced to a minimum. Hence, as a second major requirement we developed a generic solution Local Core Memory Optimization (LOCOMO) to reduce the time spent by the core for accessing data and code. Another, aspect for us was to make it possible for our user to be able to utilize this technique in different projects.

4 Methodology and Tools

As there are different sources of information which are available, we focused on developing a platform tool solution for all such requirements where user can benefit from it without spending much effort to make several format transformations. To keep this effort minimal and achieve global visibility we have decided to use AMALTHEA data model from eclipse open source APP4MC (`https://www.eclipse.org/app4mc/`) project. The aim of APP4MC was to develop a platform model specialized for multi-core system description. Therefore, we decided to base our tooling platform (PLAT4MC) on AMALTHEA model to achieve higher flexibility in the tool design and interface it with proprietary simulation tools.

PLAT4MC provides different tools to obtain an AMALTHEA model from either already existing software specification (AUTOSAR/MSR) or C sources and Executable and Linking Format (ELF). Later, the obtained AMALTHEA model can be refined, enriched or adapted according to the optimization use case. Therefore, PLAT4MC additionally provides interfaces and tools such as data consistency checks, local core memory optimizations and the merger. It also include the visualization architecture explorer based on eclipse technologies. Figure 2 show the architecture details of PLAT4MC which is based on open source technologies and mainly includes

- SCA2AMALTHEA: model export from C Source Code Analysis
- ELF2AMALTHEA: model export from ELF
- AUTOSAR/MSR2AMALTHEA: model export from MSR/AUTOSAR specification
- RB_LOCOMO: Local Core Memory Optimizations based on runtime statistics
- Data Consistency Checks (DCC): Static checks for the data access inconsistencies
- Software sharing infrastructure to exchange models with OEMs

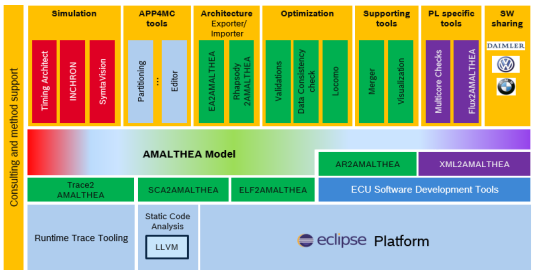

Figure 2: Architecture of PLAT4MC

Different project can use combinations of tools from PLAT4MC to achieve high- or low-level optimizations. For example, some projects obtain AMALTHEA model from C-sources, enrich memory allocation information from linker configurations and ELF sources. The obtained model depending on the core-load distribution requirement is mapped to HW architecture, this newly derived SW distribution is further optimized for data and code allocation using rb_locomo. The resultant system has lower load on communication network on the ECUs. Similarly, the AMALTHEA model can be used to verify ASIL specific component communications. It can be seen that strength of PLAT4MC lies in its shared exchange data format among several tools.

5 Summary

In this paper, we have introduced PLAT4MC an ecosystem that is based mainly on open source eclipse technologies, and discussed how its core, AMALTHEA model, can be used as an exchange format to achieve multi-core system related optimizations across several tools.