2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION JAPAN

Profiling and Optimization of Level 4 vECU Performance for faster ISO26262 Testing

A Gateway vECU Example based on QEMU and SystemC TLM

Lukas Jünger, MachineWare GmbH, Aachen, Germany (*lukas@mwa.re*)

Hitoshi Hamao, Renesas Electronics Corporation, Tokyo, Japan (*hitoshi.hamao.yw@renesas.com*)

Megumi Yoshinaga, Renesas Electronics Corporation, Tokyo, Japan (*megumi.yoshinaga.uf@renesas.com*)

Koichi Sato, Renesas Electronics Corporation, Tokyo, Japan (*koichi.sato.xv@renesas.com*)

***Abstract*—The complexity of HW/SW systems has surged in recent decades, with even small embedded systems now running huge software stacks comprising millions of lines of code. This complexity is particularly critical in safety- and security-critical domains like automotive and aerospace, where thorough testing and verification are essential, often prerequisites for certifications such as ISO26262. Moreover, software issues can lead to substantial consequences, including accidents, damage to brand image, delayed product launches, and costly product recalls, underscoring the paramount importance of software quality. However, the limited availability, scalability and flexibility of physical hardware prototypes pose significant challenges during software development and verification. To address these challenges, testing strategies leveraging full system simulations executing unmodified target software, commonly referred to as Virtual Platforms (VPs) or Level 4 virtual Electronic Control Units (L4 vECUs), have emerged as a recommended and established practice. Unfortunately, the execution performance of L4 vECUs is frequently subpar, introducing scalability limitations. Additionally, identifying performance bottlenecks within these virtual environments can be challenging. In this paper, we propose a novel approach to construct fast, scalable, and flexible L4 vECUs utilizing the popular open-source simulator QEMU and the SystemC TLM standard. Leveraging SystemC TLM allows for seamless integration of existing simulation models and facilitates profiling of simulation execution to pinpoint bottlenecks accurately. We demonstrate our approach through a representative case study involving the construction, profiling, and optimization of a L4 vECU for a gateway ECU. By identifying and addressing simulation performance bottlenecks, we achieve a speedup of 1000x for the test scenario. This underscores the effectiveness of our methodology in enhancing the performance and scalability of L4 vECUs, offering significant benefits for the development and testing of complex HW/SW systems in safety-critical domains.**

Keywords—vECU, ESL, Simulation, SystemC, TLM, QEMU, Profiling, Optimization

I. Introduction

In recent decades, the complexity of hardware/software (HW/SW) systems has surged, with contemporary devices consisting of multiple processor clusters of varied architectures and a diverse array of peripherals, including interface controllers, GPUs, and Artificial Intelligence (AI) accelerators [1]. These sophisticated systems execute extensive software stacks comprising millions of lines of code, enabling the complex functionalities they offer [2]. While this trend affects various industries, its impact is particularly profound in safety- and security-critical domains like automotive and aerospace. In these domains, software issues pose significant risks, ranging from potential accidents to substantial financial losses due to brand image damage, costly product recalls, and delayed product launches. Consequently, ensuring correct system functionality is paramount, necessitating extensive and rigorous software testing and verification processes.

Traditionally, software development and testing in the automotive domain follow a waterfall model, commencing with abstract Model-in-the-Loop (MiL) development, followed by more precise Software-in-the-Loop (SiL) development. However, in both scenarios, the target hardware is not yet available. The transition from SiL to Hardware-in-the-Loop (HiL) environments marks a significant step, as SiL environments are compiled for the simulation host rather than the actual target, resulting in challenges when transitioning to real hardware for HiL testing. This approach places significant restrictions on software testing, relying heavily on the availability of physical Electronic Control Unit (ECU) prototypes, which are typically limited, particularly in the early stages of

development. Additionally, issues often emerge late in the design cycle during HiL testing, necessitating costly HiL testing hardware.

To address these challenges, leveraging full system simulations that execute unmodified target software, commonly referred to as Virtual Platforms (VPs) or Level 4 virtual ECUs (L4 vECUs), offers a promising solution. Developed in parallel with hardware, these platforms can be available much earlier in the design cycle, facilitating a "shift-left" approach to software testing. This approach leads to more mature software when transitioning to physical prototypes for testing, thereby reducing time to market. Moreover, these platforms are easily scalable, either on-premise or in the cloud, facilitating parallelized software development and testing processes.
Ideally, L4 vECUs should execute target software at real-time speed or faster to expedite software testing tasks, such as code coverage or unit testing for ISO26262 compliance. However, slow L4 vECUs hinder the applicability of this approach. Generally, a L4 vECU comprises simulation models of individual ECU components, which may originate from different vendors and may only be available in binary form. SystemC TLM-2.0 [3] serves as the de-facto standard for interfacing these diverse models to construct a unified L4 vECU. However, identifying slow models within the vECU poses challenges.

This paper contributes to addressing these challenges through the following:

- Proposing a novel approach to constructing L4 vECUs based on the fast, open-source simulator QEMU [4] and SystemC TLM-2.0.
- Introducing a novel profiling method to identify simulation bottlenecks in SystemC TLM-2.0-based L4 vECUs.
- Presenting a representative case study to demonstrate the effectiveness of our approach.

II. Fast Level 4 vECUs with QEMU and SystemC TLM

Level 4 vECUs are able to execute the unmodified target software binary such as AUTOSAR, Linux, or Real-Time Operating Systems. This means that the same exact software stack can be run on the vECU as on the physical ECU without having to introduce any changes. To enable this, L4 vECUs contain simulation models of all ECU components such as CPUs, but also peripherals, so that the hardware driver code can be executed and tested. An overview of the L4 vECU architecture and integrations is provided in Figure 1. At the center is the L4 vECU comprised of different simulation models which are interfaced using the SystemC TLM2.0 standard. The simulation components in our architecture stem from QEMU, the open-source VCML [5] project and standard SystemC TLM2.0 models. QEMU is used for its fast CPU models and extended with SystemC TLM2.0 interfaces. VCML is used as it provides many useful modeling primitives that speed-up model creation. The L4 vECU is usually executed on a simulation host of a different architecture than the target, e.g. an ARM Cortex-A target is simulated on an Intel x86 host.

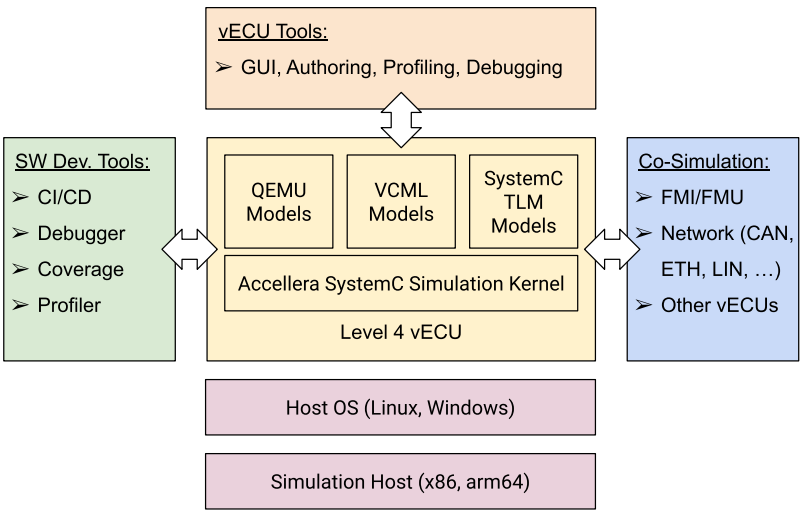

Figure 1: L4 vECU Overview.

The L4 vECU supports several integrations to enhance software development and testing. These integrations include tools for debugging, code coverage analysis, profiling, and Continuous Integration/Continuous Delivery (CI/CD). By utilizing these tools, the L4 vECU can replace a physical ECU in interactive scenarios such as software development and debugging. Additionally, the L4 vECU can be integrated with CI/CD platforms like Jenkins to automate test case execution and code coverage analysis for every commit.
Beyond these integrations, there are also specialized tools designed for the vECU itself, such as authoring tools for modifying the vECU, graphical user interfaces (GUIs) for examining vECU states, and profiling tools, which will be detailed in Section 3. The L4 vECU includes models of various ECU interface controllers, such as CAN, LIN, USB, and I2C, enabling it to send and receive traffic on these interconnects. This capability allows seamless integration with other simulation environments like FMI [6] or Vector SILKit [7]. It can also connect with other vECUs to form larger simulation environments for testing advanced middleware features. Furthermore, these interfaces facilitate fault-injection testing, enhancing the robustness and reliability of the software.

III. LEVEL 4 VECU PROFILING

As previously emphasized, achieving high performance in L4 vECUs is imperative. When developers interact with vECUs, they anticipate performance levels at least comparable to those of physical ECUs. In Continuous Integration (CI) scenarios, while performance may be less critical, it still significantly impacts test case execution time. Slow vECUs not only prolong developer wait times but also escalate compute costs. Overall vECU performance is invariably constrained by the slowest models within.

Navigating the complexity of vECUs, compounded by the absence of source code, presents a challenge in pinpointing performance bottlenecks and discerning which models contribute to diminished simulation performance. By leveraging SystemC TLM 2.0 as the foundation for our L4 vECUs, all component models register themselves with the SystemC kernel scheduler, responsible for orchestrating the simulation process. Leveraging standardized SystemC TLM2.0 interfaces facilitates seamless replacement of the SystemC kernel with an advanced counterpart equipped with profiling capabilities, as illustrated in Figure 2. The profiling kernel meticulously instruments simulation execution, capturing profiling data stored in a dedicated, low-overhead database throughout runtime. To mitigate storage requirements, profiling can be selectively enabled and disabled during simulation execution.

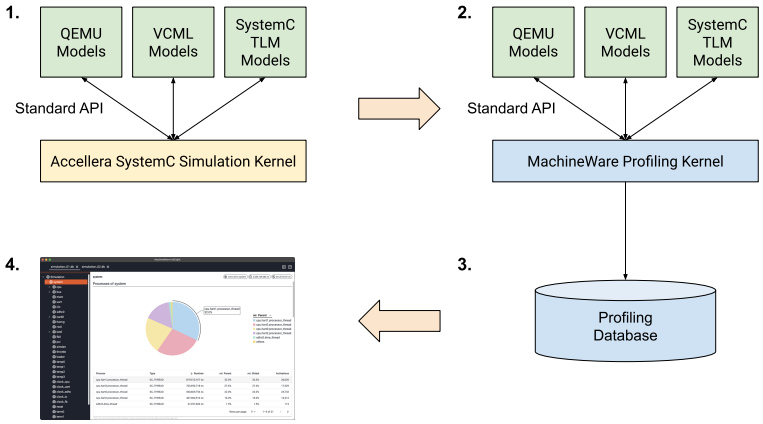

Figure 2: L4 vECU profiling process.

Upon completion of simulation execution, the profiling database is imported into the "InSCight" visualization tool for comprehensive post-processing and analysis. Within this tool, users can evaluate the simulation performance of each vECU component using various metrics collected by the profiling kernel. Once critical bottlenecks are identified, simulation models can be enhanced with performance optimizations. Subsequently, the execution and profiling steps can be iteratively repeated, allowing for ongoing comparison and refinement. This iterative process enables the systematic optimization of overall vECU performance, ensuring efficient and effective virtualized system development.

IV. Case Study: Server/Communication Gateway ECU

To demonstrate the effectiveness of our innovative L4 vECU architecture and profiling methodology, we apply it to a Server/Communication Gateway ECU scenario. Initially, an L4 vECU of the gateway is constructed, integrating models from QEMU, VCML, and custom SystemC TLM-2.0 models. This comprehensive assembly enables the execution of diverse operating systems, including Linux, real-time operating systems like AUTOSAR or Zephyr RTOS, and facilitates connectivity to network interfaces—an essential feature for a gateway ECU. Integration with host networks is achieved via Linux TAP device backends and Linux SocketCAN backends for Ethernet and CAN connectivity, further enabling co-simulation with environments like Vector SIL Kit.

In the test scenario, data is exchanged with the vECU through these interfaces, driven by Linux target software executing on the vECU. To optimize vECU performance, we replace the standard Accellera SystemC kernel with a profiling kernel while maintaining the standardized SystemC API. Profiling data is collected during the execution of the test scenario, as outlined in the previous section. After simulation completion, the profiling database is loaded into the InSCight tool for analysis. Figure 3 presents a subset of the analysis results, showcasing the distribution of simulation workload among different components and models involved. Notably, the initial analysis reveals that a significant portion of simulation execution time —43.5% in part A and 19.6% in part B— is allocated to two specific subcomponents making them the focus point of our optimization efforts. Investigation reveals that this inefficiency stems from suboptimal implementations within these components, which can be readily optimized.

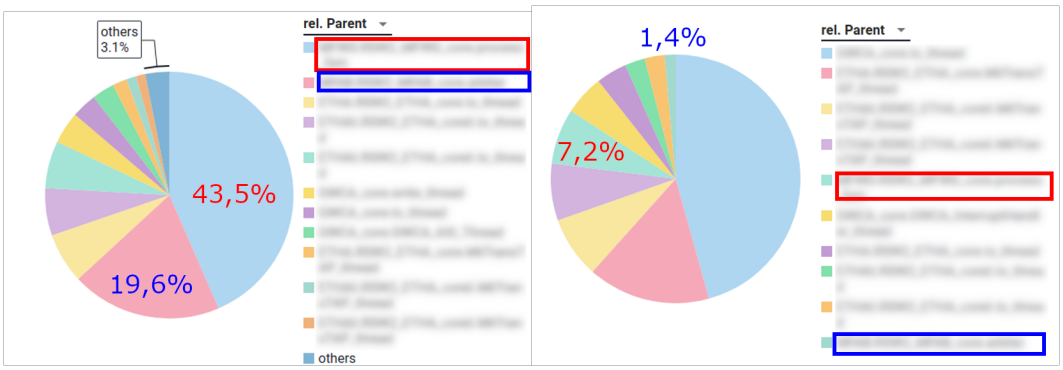

Figure 3: Profiling result comparison. Left side before, right side after optimization.

Upon optimization, subsequent execution of the test scenario demonstrates substantial improvements. In the optimized version, only 7.2% of execution time is spent in part A, and 1.4% in part B, freeing up simulation cycles for relevant components. This optimization translates into a remarkable enhancement in test scenario performance, as illustrated in Table I. Prior to optimization, the processing data rate was limited to 162kBit/s, whereas post-optimization, a performance of 110Mbit/s is achieved—an impressive speed-up of 679x over the baseline.

Table I. Data Processing Performance measurements.

Benchmark	***Base Performance***	***Optimized Performance***	***Speedup***
Data Transfer Scenario	162 kBit/s	110 Mbit/s	**679x**

These results underscore the effectiveness of our L4 vECU construction approach, offering users swift vECUs with advanced profiling capabilities. By easily identifying and optimizing simulation performance bottlenecks attributed to specific components, our methodology facilitates efficient virtualized system development.

V. Level 4 vECUs for ISO26262 Testing

Automotive functional safety requirements are defined by the ISO26262 standard, which outlines the requirements for the Automotive Safety Integrity Level (ASIL) in automotive applications. L4 vECUs play a crucial role in this process by streamlining the qualification process, ultimately leading to a shorter time to market. Many ISO26262 recommended validation and verification strategies can be executed using L4 vECUs, leveraging their early availability. This allows users to begin ISO26262 compliance testing even before physical hardware prototypes

are available, even if final testing is conducted using a physical prototype. Additionally, the early availability of L4 vECUs significantly reduces testing costs by minimizing the need for specialized hardware test setups.

Specifically, ISO26262-6:2018 [8] recommends or highly recommends (depending on the product's ASIL) the simulation of the dynamic behavior of the software architecture design for design verification, analysis of boundary values for deriving test cases, fault injection testing, resource usage evaluation, back-to-back comparison tests between model and code for software unit verification, and code coverage analysis. All these activities can be performed on L4 vECUs without the need for physical hardware prototypes.

Furthermore, these tests can be extended to cover complex electrical/electronic (E/E) architectures where applications are distributed among several ECUs. By combining multiple L4 vECUs into a larger simulation compound, the execution of the target application for ISO 26262 compliance testing can be achieved.

VI. Conclusion

In this paper, we present a novel approach for constructing fast and flexible Level 4 virtual ECUs (L4 vECUs) using Accellera SystemC TLM-2.0, QEMU, and VCML. Leveraging the SystemC TLM-2.0 standard allows for the seamless assembly of virtual platforms (VPs), incorporating preexisting simulation models, while also facilitating optimization for maximum performance through compatibility with SystemC TLM-2.0-compatible profiling kernels. This not only enhances productivity by reducing wait times for engineers and other vECU users but also increases test throughput and decreases costs and energy consumption due to shorter compute times.

L4 vECUs typically comprise simulation models from various vendors, often available only in binary form. However, the performance of the entire vECU can be adversely affected by just one slow simulation model. By utilizing standard SystemC TLM-2.0 interfaces to connect these simulation models, the SystemC TLM-2.0 simulation kernel can be easily replaced with an advanced profiling kernel. This enables the gathering of profiling data during simulation execution, which can then be analyzed using GUI tools such as InSCight to identify performance bottlenecks within simulation models.

We illustrate our approach through a representative case study focused on a Server/Gateway ECU scenario. Initially, we construct a fast L4 vECU using our methodology, which was then further optimized for a specific target scenario. Using MachineWare's advanced profiling kernel and the InSCight tool, bottlenecks in the simulation where uncovered. Using InSCight before and after the optimization enables the user to quickly understand the impact of their optimization on the overall system simulation performance. In the case study scenario, our optimizations reduced the overhead of redundant behavior in the targeted models by over 50%, resulting in a remarkable 679x speed-up in simulation workload.

Looking ahead, we aim to enhance InSCight to provide even more comprehensive visualization of simulation profiling data, facilitating easier analysis. Incorporating Artificial Intelligence (AI) techniques may enable automatic identification of simulation performance bottlenecks from profiling data, or at the very least, assist users in their analysis of simulation profiling data

References

[1] C. Scordino, A. G. Mariño and F. Fons, "Hardware Acceleration of Data Distribution Service (DDS) for Automotive Communication and Computing," in IEEE Access, vol. 10, pp. 109626-109651, 2022, doi: 10.1109/ACCESS.2022.3213664.

[2] D. F. Blanco, F. Le Mouël, T. Lin and M. -P. Escudié, "A Comprehensive Survey on Software as a Service (SaaS) Transformation for the Automotive Systems," in IEEE Access, vol. 11, pp. 73688-73753, 2023, doi: 10.1109/ACCESS.2023.3294256.

[3] "IEEE Standard for Standard SystemC® Language Reference Manual," in IEEE Std 1666-2023 (Revision of IEEE Std 1666-2011) , vol., no., pp.1-618, 8 Sept. 2023, doi: 10.1109/IEEESTD.2023.10246125.

[4] Bellard, F., 2005, April. QEMU, a fast and portable dynamic translator. In USENIX annual technical conference, FREENIX Track (Vol. 41, No. 46, pp. 10-5555).

[5] https://github.com/machineware-gmbh/vcml, accessed April 30th 2024

[6] https://fmi-standard.org/docs/3.0/, accessed April 30th 2024

[7] https://vectorgrp.github.io/sil-kit-docs/, accessed April 30th 2024

[8] ISO 26262-6:2018 Road vehicles — Functional safety, Part 6: Product development at the software level