2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

Virtual ECUs with QEMU and SystemC TLM-2.0

The Best of Both Worlds

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

Jan Henrik Weinstock, MachineWare GmbH, Aachen, Germany (jan@mwa.re)

Munish Jassi, Renesas Electronics Europe GmbH, Düsseldorf, Germany (munish.jassi.wg@renesas.com)

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

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

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

***Abstract*—With the increasing complexity of overall vehicle architecture due to the emergence of autonomous driving and other ADAS technologies, virtual prototyping is gaining popularity in the automotive industry. Modern cars consist of numerous ECUs that run millions of lines of software code and are interconnected to form the vehicle's electronic system, which is responsible for critical functions. Ensuring the safety and security of such a complex machine is crucial for roadworthiness certification and customer satisfaction. To comprehensively verify these systems, expensive hardware testbenches are traditionally used, but they are difficult to scale. To address this scalability issue, virtual platforms or virtual ECUs are used as full system simulators. They enable automotive companies to verify the correct functionality of their unmodified target software even before the first hardware prototypes are available, following a "shift-left" approach. In this study, we present a novel framework for constructing and deploying virtual ECUs using the popular QEMU system simulator and standard SystemC TLM-2.0. Our approach combines existing models from the QEMU and SystemC TLM-2.0 domain into one fast virtual ECU. Besides offering high performance, our simulation environment also allows easy integration into full vehicle simulations through standard vehicle interconnects such as Ethernet and CAN. In our case study, we demonstrate the effectiveness of this approach on a virtual ECU model of a popular automotive gateway ECU. By adopting this framework, automotive companies can efficiently verify their electronic systems, reduce costs, and improve time-to-market.**

***Keywords*—ESL, SystemC TLM, QEMU, Virtual ECU, Verification**

I. Introduction

Moore's Law has led to an increase in the complexity of electronic systems over the last few decades. Not only has hardware complexity risen, but software complexity has also seen a similar, if not stronger, increase. Even small embedded devices now run software stacks consisting of millions of lines of code, driving the need for advanced verification techniques to ensure correct, safe, and secure operation of these systems.

The automotive market has been heavily impacted by these trends, with modern cars containing numerous Electronic Control Units (ECUs) connected in complex networks using technologies such as Controller Area Network (CAN) or Ethernet. The software stacks in these cars can surpass 100 million lines of code due to advancements in the ADAS field, which further increases the need for rigorous verification. Safety and security are also paramount in the automotive industry, as they are necessary for roadworthiness certification and the end user.

To address these challenges, this work proposes a novel framework for constructing virtual ECUs based on QEMU [1], a fast open-source system simulator and the open-source Virtual Components Modeling Library (VCML), which is based on SystemC TLM-2.0. QEMU is used because its Dynamic Binary Translation (DBT) based CPU models allow for rapid software execution. However, QEMU's monolithic architecture poses challenges,when it comes to modeling systems consisting of several CPU clusters of different architectures. To address this limitation VCML is integrated with QEMU enabling the modeling of these complex System-on-a-Chip (SoC) architectures, utilizing flexible combinations of models from both worlds without limitations. Additionally, backends for CAN and Ethernet networks are provided through VCML for full vehicle simulation.

Overall, this framework provides a solution to the increasing complexity of hardware and software in electronic systems and addresses the limitations of traditional SystemC simulations. It offers a flexible infrastructure for integrating different HW models, allowing for efficient and effective verification of software for modern automotive systems.

II. Related Work

QEMU's performance and broad support especially for the ARM CPU architecture has attracted the interest of the simulation community for quite a while. In the past, it has been integrated into SystemC environments for different use cases. Delbergue et al. present an integration used to simulate ARM systems in [2]. In their setup, SystemC serves as the source of simulation time. They analyze the overhead of SystemC QEMU synchronization and compare it to a version in which SystemC models were integrated into QEMU and conclude that the overhead is comparable. Overall the integration overhead is about 10%, when executing the Dhrystone benchmark. For Linux boot the overhead is negligible. A similar approach is shown in [5].

The authors of [3] discuss a QEMU SystemC integration focused on integrating QEMU CPU models. For this they use the Unicorn emulator as a starting point [4], which provides a mechanism for building QEMU as a library and interfaces for accessing QEMU CPU models. A benchmark VP is constructed and performance is measured for the CoreMark benchmark. A comparison is made to the QEMU SystemC integration from [2]. For small SystemC TLM quanta, a significant performance improvement is demonstrated. In [6] the author has presented simulation speed gain using QEMU models of an automotive device model.

III. Virtual ECU Architecture

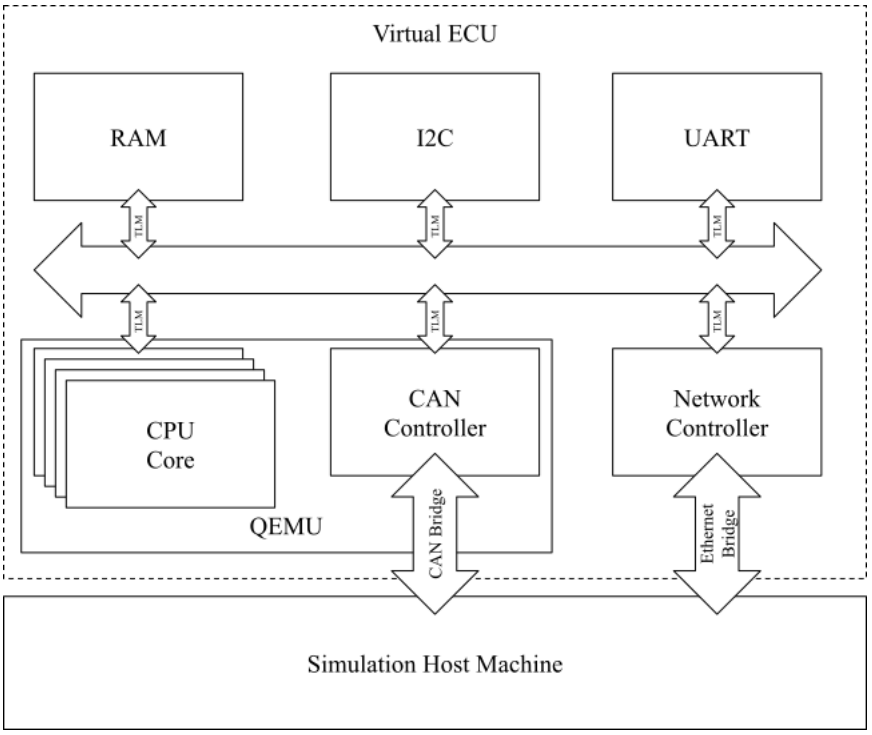

Figure 1: Virtual ECU Architecture Overview

The architecture of the virtual ECU framework is presented with a simplified example in Figure 1. It can be observed that the virtual ECU consists of two domains: the QEMU domain and the TLM domain. Models from

the QEMU domain are integrated using standard SystemC TLM-2.0 interfaces, like other models from the TLM domain. The framework allows use of both QEMU CPU as well as QEMU peripheral models. No restrictions are placed on the models in the TLM domain either, except that the use of standard SystemC TLM-2.0 interfaces is required. TLM models may be based on VCML or any other compatible library.

To enable communication with other simulators through the simulation host, the framework supports network bridges for transferring data bidirectionally between host and virtual ECU. As of now, CAN and Ethernet networks are supported. CAN traffic is handled by the Linux SocketCAN stack on the simulation host, allowing to connect the virtual ECU to other CAN simulators or even physical CAN hardware. Ethernet traffic is handled by Linux TAP interfaces for maximum flexibility, or by the SLIRP user space networking stack for easier use without requiring elevated privileges for interface creation.

The architecture of the framework combines advantages from the QEMU domain with the advantages of SystemC TLM-2.0. The main advantage of using SystemC TLM-2.0 standard interfaces for all models is that integration into existing simulation environments is simplified and the reuse of existing models from either the QEMU or the TLM world is supported. The advantage of using the fast processor models from QEMU is that high simulation performance can be achieved. Target software debugging is supported through QEMU's gdb implementation allowing the user to debug the virtual ECU just like its physical counterpart. Furthermore, the framework leaves the user with maximum flexibility in the SystemC/QEMU partitioning as both domains are supported equally well.

IV. Results

This chapter presents the experimental evaluation of a virtual platform simulator (VP) developed for a specific Renesas device. The VP was tested in common scenarios, including executing bare metal applications and Linux boot. These use cases align with the customer's needs, and the evaluation aimed to ensure the VP's reliability and effectiveness. The results provide insights into its performance and suitability as a virtualization solution.

The performance evaluation began by testing the VP's ability to simulate Linux boot, which is the most common use case. The performance measurement involved timing the simulator's simulation process from power-off until reaching the Linux command line. It is important to note that the VP simulated a custom Linux kernel specifically built for the Rensas device, ensuring a realistic emulation of the target system.

In this evaluation, two modes of operation were tested: icount and mttcg, which correspond to the operation modes of the underlying QEMU CPU model. The icount mode relies on instruction counting to determine the execution progress, while the mttcg mode employs a multi-threaded approach for faster simulation. Both modes were evaluated to assess their impact on the VP's performance and determine the most suitable mode for achieving optimal simulation results.

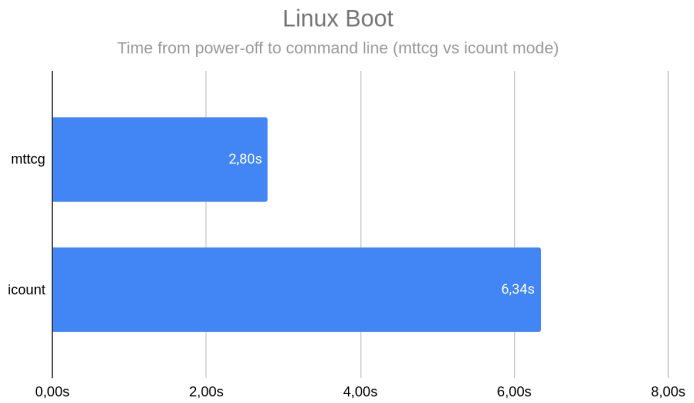

Figure 1: Runtime comparison of Linux boot for mttcg and icount mode

In the icount mode, the average boot time recorded was 6.3 seconds, while in the mttcg mode, it reduced significantly to only 2.8 seconds. The improved speed of the mttcg mode can be attributed to its ability to parallelize the workload across the four ARM processors in the system. However, it is important to note that the mttcg mode is nondeterministic, meaning that the simulation results may vary across different runs. On the other hand, the icount mode may be slower, but it consistently produces reproducible results in terms of timing simulation. The choice between the two modes depends on the specific customer requirements of the simulation, taking into consideration the need for speed or deterministic timing.

Next, we conducted measurements using bare metal benchmark software to assess single-core performance, utilizing only one core. The Dhrystone benchmark was executed with 10 million iterations, and Coremark was executed with 100,000 iterations. The results of these benchmark tests are visually represented in the bar diagram below:

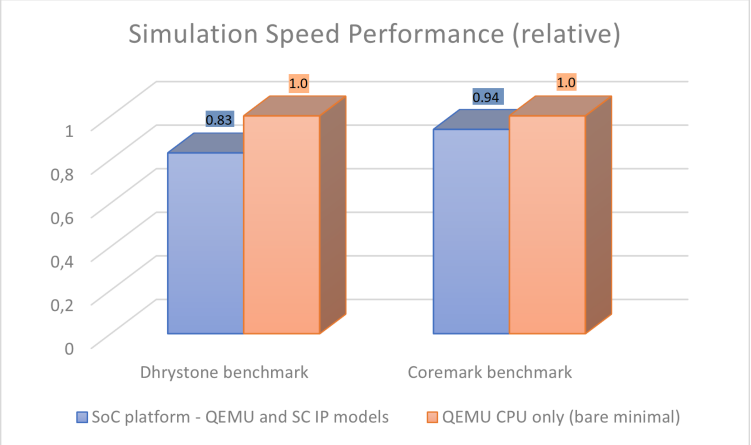

Figure 2: Comparison results of simulation speed of CPU benchmarks on QEMU and QEMU+SystemC platform

The bar diagram provides a clear comparison of the performance of the VP in terms of the Dhrystone and Coremark benchmarks, highlighting the capabilities and efficiency of the simulated system under different workloads. The results show simulation speed factors on the QEMU_SC platform and QEMU_CPU_only of 0.83x and 0.94x for Dhyrstone and Coremark benchmarks respectively. These results show that the SoC platform, including QEMU CPU and other SystemC IP components, is maintaining the advantage of simulation speed of QEMU models.

These results provide insights into the processing capabilities and efficiency of the simulated system when subjected to these specific benchmark workloads.

V. Conclusion

In this work a novel framework for virtual ECU construction was introduced based on QEMU, VCML and SystemC TLM-2.0. The framework was applied to create a model of a state-of-the art gateway ECU. In the presented case study performance was analyzed for different model execution modes. Overall, the virtual ECU performs well and is even usable interactively. Furthermore, integrations for full vehicle simulation via CAN and Ethernet interfaces are made available.

In the future, the virtual ECU will be extended with additional peripheral models to enable further interfacing via other industry-standard interfaces. In addition, it will be explored how the debug capabilities of the virtual ECU can be improved beyond standard GDB integration, e.g. by making an MCD interface available.

References

1. Bellard, Fabrice. "QEMU, a fast and portable dynamic translator." USENIX annual technical conference, FREENIX Track. Vol. 41. 2005.
2. Delbergue, Guillaume, et al. "QBox: an industrial solution for virtual platform simulation using QEMU and SystemC TLM-2.0." 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016). 2016.
3. Jünger, Lukas, et al. "Fast SystemC processor models with unicorn." Proceedings of the Rapid Simulation and Performance Evaluation: Methods and Tools. 2019. 1-6.
4. Quynh, NGUYEN Anh, and DANG Hoang Vu. "Unicorn: Next generation cpu emulator framework." BlackHat USA 476 (2015).
5. Weinstock, Jan Henrik, et al. "AMVP-a high performance virtual platform using parallel SystemC for multicore ARM architectures: work-in-progress." Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis. 2018.
6. Kota, Shima, Renesas QEMU environment for R-Car S4 for high-speed simulation, https://www.renesas.com/eu/en/blogs/r-car-s4-renesas-qemu-environment