Advances in multi-abstraction distributed vECU co-simulation technology for automotive software testing

Markus Wedler, Alexandru Fiodorov, Marcel Heistermann, Maximilian Fricke, Shubham Paul

Synopsys Inc.
Ritter Str. 23
52072 Aachen
wedler@synopsys.com

Abstract: This paper introduces the Synopsys electronics-digital-twin-fabric (eDT-fabric), a new backplane technology for distributed co-simulation for virtual electronic control-units (vECUs). This eDT-fabric technology addresses the need of OEMs to integrate vECUs and plant-models from various vendors into full-car simulations. These electronics digital twins are modeled at various abstraction levels with very different requirements regarding their execution platform. The eDT-fabric described in this paper therefore provides true multi-host, multi-host-platform, multi-host-OS and multi-abstraction co-simulation capabilities. Its advanced synchronization and data exchange mechanisms enable maximum performance while keeping the overall simulation deterministic.

1 Introduction

OEMs aim for virtualization of the entire car and its environment for full system validation of their embedded system software stacks prior to deployment on hardware-in-the loop testing platforms and subsequent field testing. Depending on the individual validation targets, Virtual ECUs modelled at different levels of abstraction are used in these co-simulation environments. It is thus desired to seamlessly switch between the abstraction levels and this needs to be supported by a distributed co-simulation platform.

The introduction of Adaptive AUTOSAR as a POSIX compliant embedded OS further increases the heterogeneity of the simulation models. Availability of certain virtual models and tools in the automotive software testing eco-system may be limited to certain OS-platforms. Specific hardware requirements may also be imposed by scenario simulators that often call for advanced GPUs for rendering of environment images.

In this context we introduce our next generation parallel co-simulation backplane technology - the so-called Synopsys eDT-fabric for distributed data-exchange and synchronization between virtual ECUs and environment simulations. This is the crucial component of our platform for full system test & development of embedded vehicle software. It offers true multi-abstraction, -host, -host-architecture, -host-OS support. Differences in simulator performance are embraced and additional communication overhead is minimized. Our synchronization methods ensure deterministic simulation and maximize parallel execution.

2 Motivation

The automotive industry is facing a lot of new challenges. The world-wide shortage of chips has compressed the traditional supply chain of OEMs, Tier 1s and Tier 2s etc. New entrants into the market come up with radical new approaches designing their own SoCs. In this manner they have the entire value chain from hardware to software in their own hand. This enables system-wide optimizations and innovations from silicon to software.

At the same time, we see a major overhaul of vehicle E/E architectures. The classical scheme of one ECU per function does not scale sufficiently to support new HAD/ADAS features. The electrification of the powertrain drastically changes the software needed in this domain as well. Moreover, car owners require infotainment systems that deliver similar or even better UI experiences as they are used to in their consumer electronics. All this leads to new zonal architectures that leverage powerful central compute clusters to do the heavy lifting in computing for several zonal controllers that coordinate specific regions in the car.

In total we are talking about millions of lines of code being executed. In the end the OEM is responsible for all parts of the software working together flawlessly. The fact that the code stems from various vendors today and often is only delivered in binary causes many integration issues. We envision that over time completely new software ecosystems will evolve. Standardized interfaces, e.g., AUTOSAR Classic for real time applications and AUTOSAR Adaptive for the POSIX based applications, are promising first step in that direction.

Software updates over-the-air will become widely available to roll out new Software features and enable fixing issues in the field that slipped despite all efforts.

However, this increases the attack surface for malicious access through the over-the-air update infrastructure. Combined with the vision for self-driving cars that are always connected through 5G networks. Thus, a strong focus on safety and security aspects of these interfaces is mandatory.

New entrants into the automotive markets challenge the traditional segregated development processes and thereby accelerate their time to market.

3 Electronics Digital Twins and their abstraction levels

Virtualization is a key technology to massively scale the software testing efficiency. The traditional physical testing strategy cannot keep pace with the above-mentioned trends and has become the bottleneck in development. The trend in the industry is clearly to transition towards virtualization, i.e., simulation-based testing [1].

This has numerous benefits. It enables early software bring-up even before physical prototypes of the tested subsystems become available. State-of-the-art virtual ECU technology provides deterministic simulations and a very high degree of debug visibility into the systems under test. This shortens debugging cycles and boosts the overall testing productivity. Electronics digital twins (eDTs) support agile development methodologies based on continuous integration/continuous delivery of the embedded software under development. The software nature of these virtual environments also enables a tighter collaboration through the supply chain of the OEMs [2]. Exchanging software in binary or source can be done frequently in increments, whereas Hardware needs to be physically manufactured and shipped with the attached costs. Thus, a high number of iterations that characterizes true agile development processes is no longer a showstopper for the collaboration.

Standardized APIs like the Functional Mockup Interface (FMI) for co-simulation and model exchange [2] for host-compiled vECUs or the SystemC/TLM standard [3] for modeling virtual prototypes of the underlying hardware that can run unmodified binary code, provide another important foundation for such collaborative development approaches.

For a more precise categorization of the various abstraction levels at which virtual ECUs can be modeled, we refer to the whitepaper [4]. In this paper, the authors introduce the abstraction levels visualized in Figure 1.

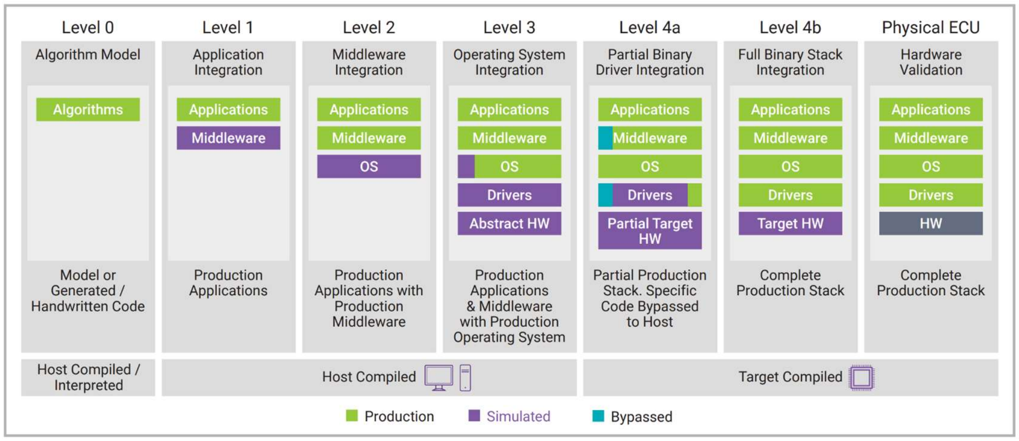


Figure 1: Virtual SW/HW abstractions for broadest use cases.

The categorization reaches from algorithmic models, created by model-based design tools like Matlab Simulink and tested in so called model-in-the-loop setups on the left side of the figure, to a complete physical prototype of an ECU running on a hardware prototyping system. The spectrum between these two cases is filled with vECU models that take into account ever more details of the underlying middleware, OS, hardware drivers and target hardware behavior. The figure shows which portions of the software stacks are taken from the real production code and which portions are simulated at the respective levels.

The Figure enhances the categorization of [5] that was mainly focused on AUTOSAR Classic use-cases to also reflect POSIX based developments around AUTOSAR Adaptive. For this purpose, the level 4 dealing with target compiled software is split into the levels 4a and 4b. At level 4a VirtIO driver substitution and host-extension technologies may be involved to simulate or bypass parts of the native driver behavior and thus reduce the scope of the necessary hardware models. By contrast, vECUs at level 4b can run the full binary production code like the target hardware.

4 Integration of vECUs

One important benefit of electronics digital twins is that we can leverage vECUs and their early availability for early system integration. Instead of waiting for a big bang integration when all ECUs are available, partial systems needed for specific driving scenarios can be setup up and tested. As an example, let us consider an adaptive cruise control and navigation tracking scenario. The ECUs involved in this scenario are integrated into the setup illustrated in Figure 2. It consists of 4 virtual ECUs for the navigation system, the central compute cluster, a zonal domain controller, and a control ECU, modeled at different abstraction levels. In addition, a third-party environment simulator and a playback node representing the Human Machine Interface (HMI) are added to the setup. Visualization, debug, and tracing capabilities are enabled for validation and root cause analysis in case of unexpected system behavior.

The need for integration of simulators and simulation models from various tool vendors with their specific hardware and platform requirements further increases the heterogeneity of the co-simulation. For example, the environment simulator may require a dedicated graphics card to render images. Other models may need an NPU, e.g., in the sensor data processing. Some tools may only be available on either Windows or Linux. Hypervisor based simulations may take benefit of specific host-architectures like aarch64.

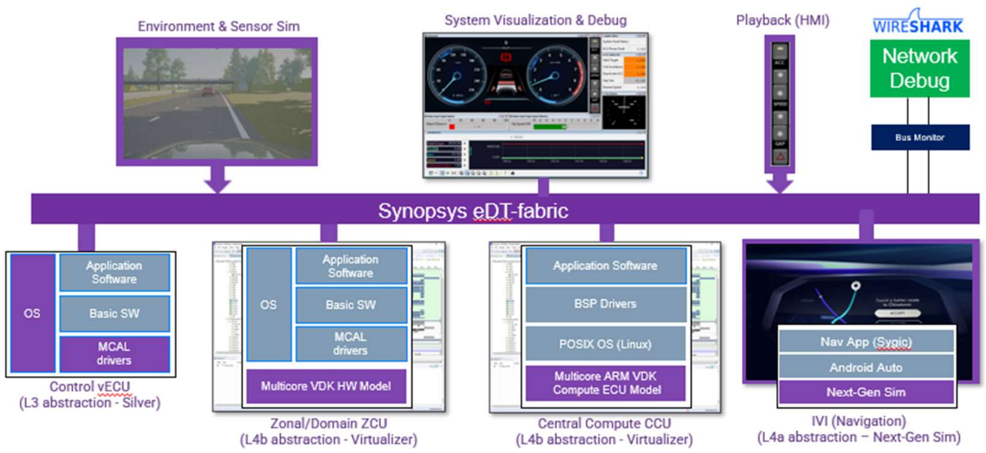


Figure 2: Early system integration for complex function validation.

5 Implications for the eDT-fabric and its features

In this section we will focus on our technology for the challenges described in the previous two sections. We introduce our new co-simulation backplane technology called electronics-digital-twin-fabric (eDT-fabric). This eDT-fabric has the following characteristics:

- **Multi-abstraction**: It allows for seamless integration of vECUs at level 1 to 4. It facilitates idealized Signal based communication as well as packet/frame-based communication for the CAN, Ethernet, and LIN. Further automotive field busses are on the roadmap.
- **Multi-host**: It provides a true distributed simulation on multiple execution hosts. It is cloud ready including container support. It does not rely on a central data-broker to avoid congestion and is scalable to the vehicle level.
- **Multi-Platform**: The eDT-fabric supports Linux and Windows OSes. Thus, it aids with restrictions regarding model availability, licensing, and dependence on platform specific APIs.
- **Multi-Architecture**: The core of the eDT-fabric is available on both Intel X86/X64 and Arm aarch64 host architectures. Simulators with dedicated hardware requirements, e.g., GPUs, NPUs or other hardware accelerators may run on dedicated machines that satisfy those requirements. Reserving the costly hardware for those participants in a co-simulation, that really need it may result in overall improved resource utilization.

The eDT-fabric technology complements the Synopsys co-simulation portfolio which consists of

- the Virtualizer Systems Interface (VSI 2.0) [6] - for co-simulation of level-4 ECUs with each other and with plant models, and
- Silver with its Synopsys Virtual Bus (SVB) - the co-simulation engine for vECUs at levels 3 and below.

Both VSI 2.0 and SVB rely on shared-memory-based communication and thus are very well suited for fast single host co-simulation of tightly coupled systems on a single host.

By contrast, the eDT-fabric has been designed with the distribution of the co-simulation on multiple hosts in mind from day 1. In general, we are targeting larger step sizes and assume that the vECUs communicate with each other through automotive networks or their abstraction in form of idealized communication at the software-signal level.

For more tightly coupled subsystems VSI 2.0 with its frontend Multi-Node and/or Silver can be used to co-simulate as part of an overall eDT-fabric co-simulation. The above example also illustrates that the eDT-fabric supports existing standards like the FMI standard for co-simulation. We provide a generic FMU that can be customized and imported into existing FMI importers, e.g., environment simulators or plant-model simulators to connect them to the eDT-fabric.

In the remainder of this Section, we will describe some of the key-features of the eDT-fabric.

5.1 High performance

When shifting to distributed co-simulation network latencies and bandwidth between simulation hosts need to be considered. Data exchange strategies that may be optimal for single-process and/or single-host co-simulations based on shared memory turn out to be suboptimal in the distributed case, as they assume that data can be exchanged between participants infinitely fast by swapping a bit in the shared memory.

The eDT-fabric therefore comes with a new loosely coupled synchronization and data exchange mechanism specifically tailored to support mixed abstraction. It exploits the fact that simulation speeds greatly vary between less abstract simulations at levels < 3 and detailed level-4 simulations. Even between different level-4 simulations performance may vary a lot from step to step. The above-mentioned algorithm exploits such load imbalances to reduce the overhead of the network communication for synchronization and data exchange.

Our full parallel launch mechanism that enables the user to kick-off an entire co-simulation from a central command line interface (CLI) cockpit substantially leads to the fastest possible startup time. Through respective configurations the authors of a co-simulation setup have full control over whether they want to run certain participants in headless mode, or whether they want their UI popup on the respective host to do some debugging.

5.2 Microservice-based architecture

The microservice-based architecture of the eDT-fabric includes a distributed life-cycle management for simulation and tool processes. Execution hosts for the eDT-fabric will run this service. Through the life-cycle management service an instance of the CLI cockpit may start, stop, and monitor preconfigured simulators which take part in a co-simulation.

To establish the communication matrix between the co-simulation participants, e.g., the vECUs and Plant models, a dedicated orchestration service is spawned by one of the involved lifecycle managers. The orchestration service is only involved in the co-simulation at the start and at the end, and whenever participants join or leave the co-simulation. For example, a user may be interested in tracing data in a specific interval of time. The tool used for the tracing would register itself through the orchestration service.

Note, that synchronization and data exchange between the simulators is not routed through the orchestration service, to avoid that its network interface becomes a central bottleneck in larger setups.

Simulations taking part in an eDT-fabric co-simulation also provide services to their peers, to the orchestrator, and to their lifetime manager. The participant service receives incoming synchronization and simulation data from other simulators. The orchestrator service enables the orchestrator to inform the participant about changes in the communication matrix. The lifetime manager can send commands, e.g., to gracefully stop the simulation.

The CLI-cockpit interacts with these services, e.g., for status and error retrieval, and to send commands to the participants.

5.3 Deployment

The eDT-fabric comes with a documented open API. This API is used by all integrations for Synopsys vECUs and external plant-models and tools into the eDT-fabric. We also provide a generic functional mockup unit FMU that supports the FMI standard for co-simulation. This FMU can be used to integrate third-party FMI importers into the eDT-fabric.

We support virtual networks for automatic protocols and idealized signal communication between vECUs. The API also provides a custom message type that allows to exchange binary raw data between co-simulation participants.

Compositions for distributed co-simulation are human readable and easy to manipulate through scripts or UIs. The eDT-fabric provides scripting support for their generation.

5.4 Advanced features

This section completes the feature overview of the eDT-fabric and outlines three of the advanced features that enhance performance and/or support fast debug turnaround times.

5.4.1 Record and Replay

The infrastructure of the eDT-fabric comes with a built-in option for recording the traffic from and towards each individual integrated simulator. This enables two different use-models for replay of traffic in debugging scenarios outlined in the sequel.

Fully isolated mode: For in-depth debugging, the user of the eDT-fabric can run a specific simulator standalone and use the input stimuli to that simulator from a recording taken in an earlier run of the complete setup. This enables extremely fast turnaround times for determining the root cause of an issue. Obviously, this requires narrowing down the root cause of a functional issue to a specific participant in the co-simulation. Here is where the second use-model for replay comes in handy.

Replace specific nodes by recording: Individual co-simulation participants (e.g., vECUs or plant-models) may be replaced by a dedicated replay node. This executable, that comes as part of the eDT-fabric installation, reads in a recording of the API calls of the participant, and exactly reproduces its behavior for the rest of the co-simulation. This can be used during debugging to remove, e.g., certain slow participants after they have been ruled as the source of an unintended behavior. As debugging typically requires multiple debug runs this saves time for the developer and lets him focus on those parts of the system, that are relevant for the current problem.

A second motivation for introducing replay nodes deals with the availability of models. At early stages of the integration the software for some of the ECUs may not be available yet. However, some of their behaviors may be crucial to be able to integrate and test already some of the other components. The eDT-fabric supports this by a recording format that is easy to generate and human readable and thus allows for generating a recording file without running the eDT-fabric. In this manner the replay component becomes a mock for the missing vECUs in early stages of the overall integration project.

5.4.2 “Zero”-overhead Tracing

Co-simulation participants in the eDT-fabric can be configured as monitors. In this mode, a participant will only read data from the eDT-fabric. With a good caching strategy and with vECUs in the picture that do real work, the rest of the system will hardly ever have to wait for these tracing nodes that do nothing but storing away traced data. Thus, the overhead for tracing participants is usually negligible.

5.4.3 Performance Profiling

Optimizing complex systems for performance, requires insight into where the bottlenecks of a system are. The eDT-fabric has a feature to collect profiling data for the participants of a co-simulation. It comes with a mechanism for time synchronization across machines and with visualizations for the following characteristics:

- Work and wait time diagrams to analyze bottlenecks is the co-simulation. On a per-step basis these diagrams show the duration that each simulator spent simulating and the duration that the simulators spent waiting for inputs from other simulators.
- Cumulative work and wait time diagrams.
- Real-time factor including and excluding wait times for comparison of achieved performance with theoretical limit.
- Concurrency visualization including cross-step concurrency for participants that only communicate indirectly or unidirectional.
- Step-Size variations for each of the simulators.

6 Case Study

As a demonstrator, we applied the eDT-fabric to simulate an adaptive cruise-control application with collision avoidance in a mixed abstraction setup. The overall co-simulation setup is illustrated by Figure 3. It consists of level 4b vECUs for the central compute cluster und the zonal/domain controller and a control vECU at level 3. The environment simulator IPG CarMaker provides external stimuli and the Silver cockpit is used for visualization and tracing.

The overall setup is mapped onto three execution hosts. A developer laptop with dedicated GPU runs CarMaker, a windows terminal server runs the Silver control ECU and the Virtualizer VDK for the zonal controller. Finally, the Arm based central compute cluster is mapped onto a hypervisor-based simulation running on an Arm aarch64 host.

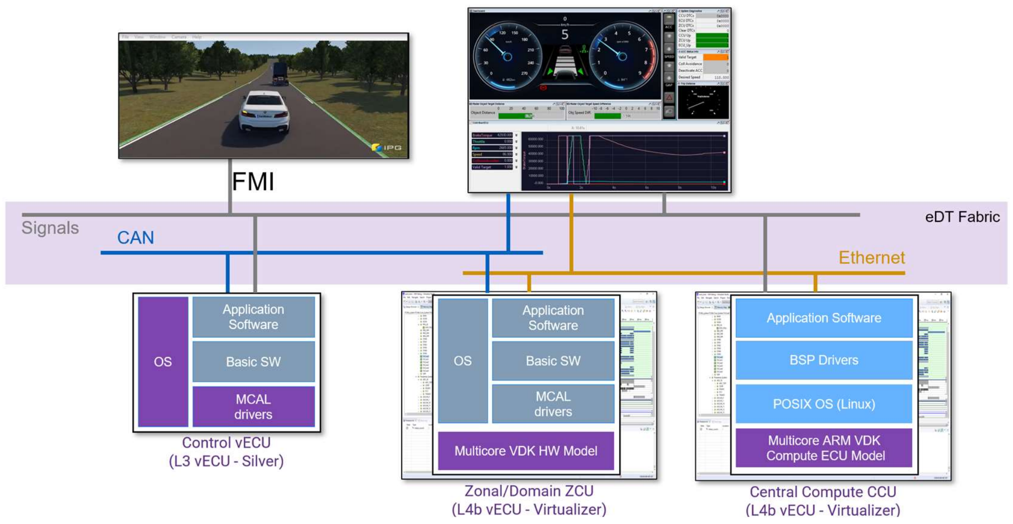
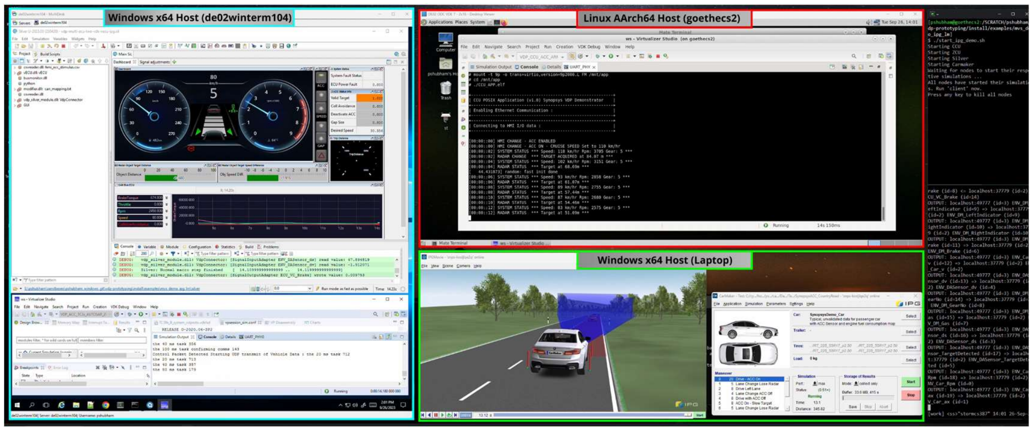


Figure 3: eDT ACC + collision avoidance mixed abstraction.

The central compute and the zonal controller are connected through Ethernet, the control ECU and the zonal controller through a CAN bus, and the environment stimuli are exchanged as software Signals.

The visualization and tracing cockpit visualizes the data and can be used to connect tools like Wireshark to inspect the network traffic.


Windows x64 Host (de02winterm104)
Linux AArch64 Host (goethecs2)
Windows x64 Host (Laptop)
ws - Virtualizer Studio (on goethecs2)
Simulation Output
Console
Details
UART_PHY
Running
14s 150ms

```
# mount -t 9p -o trans=virtio,version=9p2000.L FM /mnt/app
# cd /mnt/app
# ./CCU_APP.elf

+------------------------------------------------------+
| CCU POSIX Application (v1.0) Synopsys VDP Demonstrator  |
+------------------------------------------------------+
| Enabling Ethernet Communication :                     |
+------------------------------------------------------+
+------------------------------------------------------+
| Connecting to HMI I/O data :                          |
+------------------------------------------------------+

[00:00::00] HMI CHANGE - ACC ENABLED
[00:00::00] HMI CHANGE - ACC ON - CRUISE SPEED Set to 110 km/hr
[00:00::02] SYSTEM STATUS *** Speed: 118 km/hr Rpm: 3705 Gear: 5 ***
[00:00::02] RADAR CHANGE  *** TARGET ACQUIRED at 84.07 m ***
[00:00::04] SYSTEM STATUS *** Speed: 102 km/hr Rpm: 3151 Gear: 5 ***
[00:00::04] RADAR STATUS  *** Target at 68.69m ***
[   44.431873] random: fast init done
[00:00::06] SYSTEM STATUS *** Speed: 93 km/hr Rpm: 2858 Gear: 5 ***
[00:00::06] RADAR STATUS  *** Target at 61.07m ***
[00:00::08] SYSTEM STATUS *** Speed: 89 km/hr Rpm: 2755 Gear: 5 ***
[00:00::08] RADAR STATUS  *** Target at 57.44m ***
[00:00::10] SYSTEM STATUS *** Speed: 87 km/hr Rpm: 2680 Gear: 5 ***
[00:00::10] RADAR STATUS  *** Target at 54.46m ***
[00:00::12] SYSTEM STATUS *** Speed: 83 km/hr Rpm: 2575 Gear: 5 ***
[00:00::12] RADAR STATUS  *** Target at 51.09m ***
```

```
[pshubham@goethecs2:/SCRATCH/pshubham
dp-prototyping/install/examples/mvs_d
o_ipg_lm]
$ ./start_ipg_demo.sh
Starting CCU
Starting ZCU
Starting Silver
Starting Carmaker
Waiting for nodes to start their resp
tive simulations ...
All nodes have started their simulati
s. Run 'client' now.
Press any key to kill all nodes
```

Figure 4: Screenshot of the running co-simulation

7 Conclusion

In this paper we introduced our new eDT-fabric for distributed multi-host, multi-abstraction, multi-host-platform, and multi-host-architecture co-simulation. It provides a rich feature set backed by a novel and performant distributed data exchange and synchronization algorithm. Its micro-service-based architecture makes it cloud ready and enables easy integration in existing customer cloud distribution solutions. Advanced features for analysis, profiling and debugging let our customers focus on their goal to develop innovative high quality embedded software that runs tomorrow's cars.

8 References

[1] T. DeSchutter, Ed., Better software. Faster! best practices in virtual prototyping, Synopsys,, 2014.

[2] I. Feldner et al., "The automotive virtual prototyping platform," Virtual Platform Working Group of the Arbeitskreis Automotive of the EDACentrum., 2019. [Online]. Available: https://www.edacentrum.de/en/whitepaper-automotive-virtual-prototyping-platform.

[3] FMI, MODELISAR Consortium, "Functional Mock-up Interface (FMI) specification," 10 05 2022. [Online]. Available: https://fmi-standard.org/.

[4] "IEEE Standard for Standard SystemC Language Reference Manual," *IEEE Std 1666-2011 (Revision of IEEE Std 1666-2005),* pp. 1-638, 2012.

[5] F. Thoen et al., "Whitepaper: Accelerating Development of Software-Defined Vehicles with Virtual ECUs," 2024. [Online]. Available: https://www.synopsys.com/verification/resources/whitepapers/virtual-ecu-wp.html.

[6] Prostep IVIP, "Whitepaper Virtual Electronic Control Units," 03 2020. [Online]. Available: https://www.prostep.org/fileadmin/downloads/WhitePaper_V-ECU_2020_05_04-EN.pdf. [Accessed 22 05 2023].

[7] Synopsys Inc., *Synopsys Virtualizer Co-Simulation with Third-Party Simulator manual,* Moutain View: Synopsys Inc., 2023.