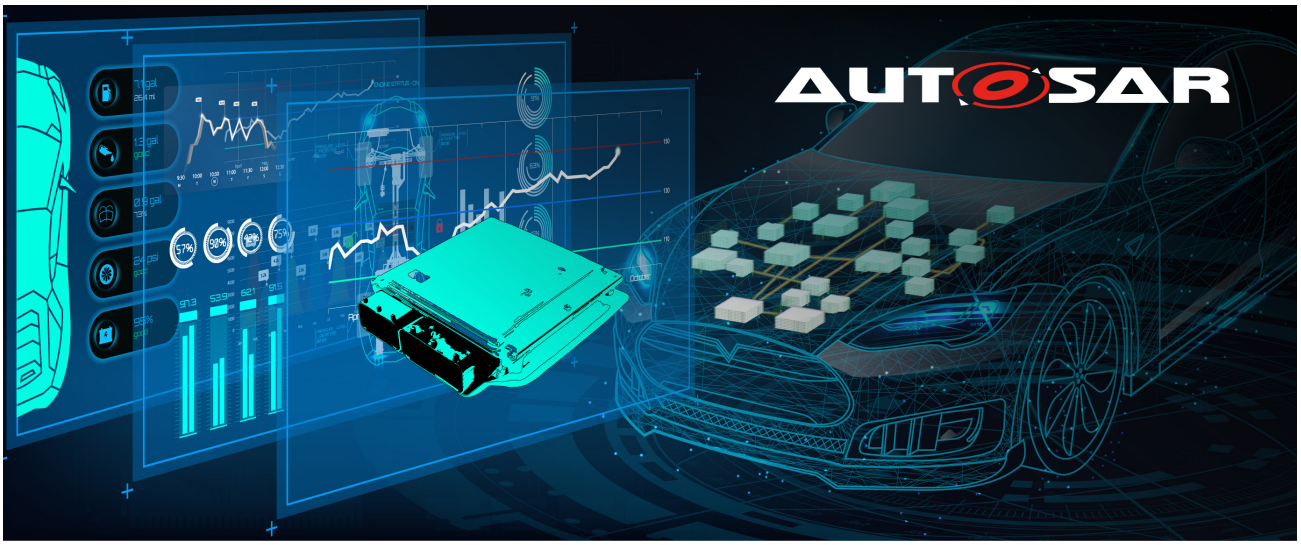
LAUTERBACH
DEVELOPMENT TOOLS



FROM THE REAL TO THE VIRTUAL TARGET WITH TRACE32®

Automotive Supplier ZF Implements Runtime Analyses on VLAB Virtual ECUs

Understanding the runtime behavior of an embedded application is crucial for optimizing resource utilization and performance, and ultimately for meeting all safety requirements. Together with Lauterbach, ZF's long-time partner for debug and trace tools, timing analyses for automotive ECUs could be moved from real hardware to virtual targets. The benefits are compelling.

By Frank Riemenschneider, Lauterbach GmbH

ZF, headquartered in Friedrichshafen, Germany, is one of the most important suppliers to the automotive industry with 160,000 employees worldwide. The company is shaping vehicles of the future at all relevant levels: from sustainability, electromobility, automated and autonomous driving to software and digitalization to vehicle motion control. The product range also includes automotive control units (ECUs), which are supplied to almost all well-known car manufacturers.

When developing an embedded application such as an ECU, not only is error-free software essential, but understanding the runtime behavior of the application is also crucial for optimizing resource utilization and performance, and ultimately for meeting all safety requirements. To this end, appropriate tools are used to measure execution times and verify timing requirements. To do this, the user must record hardware events by means of traces at the instruction level and process the trace data into system events at an abstracted level, which is known as profiling.

In order to be able to trace at all, the user must first of all use chips that are trace-capable. For economic reasons, this is still not the case for all microcontrollers or processors in 2023. ZF has taken this requirement into account from the very beginning and, among other chips, relies on the AURIX™ TC397XE, a microcontroller from Infineon with six TriCore™ CPUs, each clocked at 300 MHz. The Multi-Core Debug Solution (MCDS) implemented in

CASE STUDY

the TC397XE provides parallel and timed trace of cores and buses, powerful trigger conditions, and on-chip logic analyzer functions. The TriCore™ CPU architecture is also particularly designed for high interrupt loads, such as those encountered in engine control systems. Thanks to this predictive chip selection, ZF is thus not forced to perform a purely software-based analysis, as would have been the case with non-trace-capable chips. The code instrumentation required in this case would have a considerable influence on the timing behavior and would make the measurement results obtained only of limited use. In addition to the chip itself, ZF also makes sure from the very beginning that the software developed in-house is prepared for corresponding timing analyses.

In the so-called „real-time flow trace" with Lauterbach's TRACE32® tools, the information of the address/data bus is transmitted by the MCDS as it occurs directly at the CPU core (Figure 1). The process itself is not „Rocket Science," but trace products from different manufacturers differ in terms of data rates and functional scope. Even though a flow trace is already very powerful through the use of Lauterbach's TRACE32®, the analysis capabilities could be significantly expanded again with the introduction of AUTOSAR ARTI (AUTOSAR Run Time Interface). This was implemented at ZF in November 2021 by a joint ZF/Lauterbach project group. Lauterbach has been a member of the relevant AUTOSAR committees from the very beginning and has contributed significantly to this new standard.

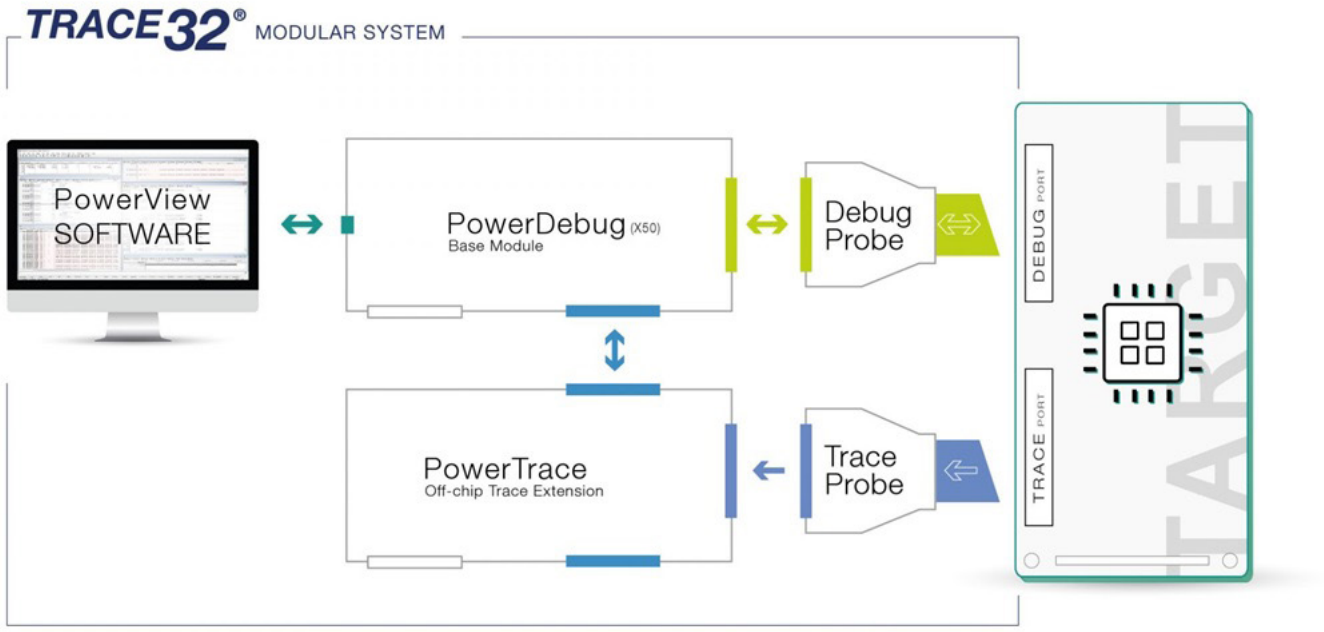


Figure 1: Diagram for real-time flow trace with Lauterbach TRACE32®. ©Lauterbach

AUTOSAR ARTI Extends Analysis Capabilities

AUTOSAR ARTI uses a similar approach to its predecessor ORTI (OSEK Run-Time Interface), but extends the trace capabilities and addresses some shortcomings of ORTI-based analysis. With ORTI, the developer is usually not able to distinguish between task completion, interruption and waiting in the trace, since only task changes can be recorded, and thus cannot perform a detailed timing analysis on activation delays or total response time, for example.

In addition, runnables are not covered by the ORTI file, nor the communication between software components. ORTI-based timing analysis focuses on the operating system and ignores other AUTOSAR modules such as software components (SWCs) and the runtime environment (RTE). The scheduling of AUTOSAR runnables (by means of OS tasks) is often the authoritative object of investigation in timing analysis.

AUTOSAR ARTI provides extended information, which is important for the automotive industry, e.g. about tasks, ISRs (interrupt service routines), runnables, RTE communication or spinlocks in an ARXML format. Thus, ARTI support enables an in-depth analysis of the runtime behavior of AUTOSAR-based systems. ARTI defines a comprehensive, standardized interface between the build tools and the debug and trace tools and, in addition to detailed debug information, also provides a model for detailed runtime measurement and analysis of operating system tasks and their runnables. This includes not only simple timing measurements, but also the possible states of tasks, runnables and ISRs. Because ARTI consistently records only this relevant information, it gets by with relatively low bandwidth.

The trace data generated for ARTI profiling (profiling the runtime behavior of an AUTOSAR-based application is often also referred to as AUTOSAR profiling) can be streamed to the host computer at program runtime, allowing very long recording times. The trace data can be used to cover CPU load analysis, event chain analysis, calculation of OS metrics, and much more, in addition to the use case validation of timing requirements, which is the focus here.

Lauterbach's TRACE32® trace tools are an established part of the AUTOSAR Classic timing tool chain and also support ARTI profiling for the AUTOSAR Adaptive Platform. Figure 2 shows an example of a decoded ARTI trace in Lauterbach's PowerView software. (Figure 2)

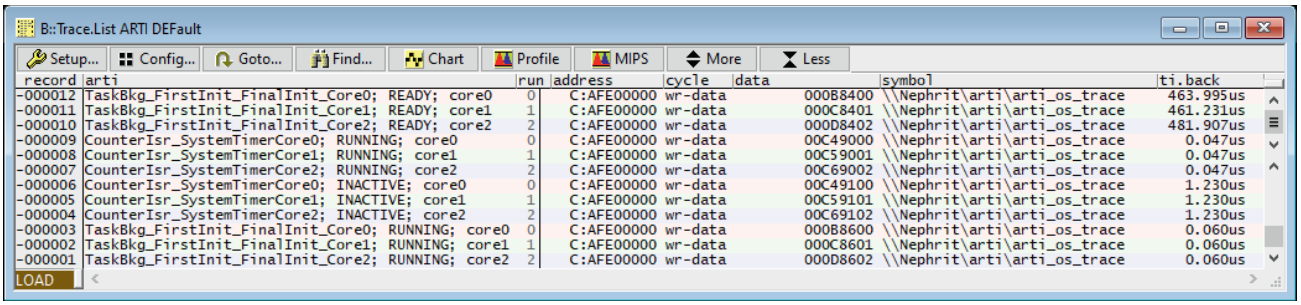


Figure 2: Example of decoded ARTI trace. ©Lauterbach/ZF

CASE STUDY

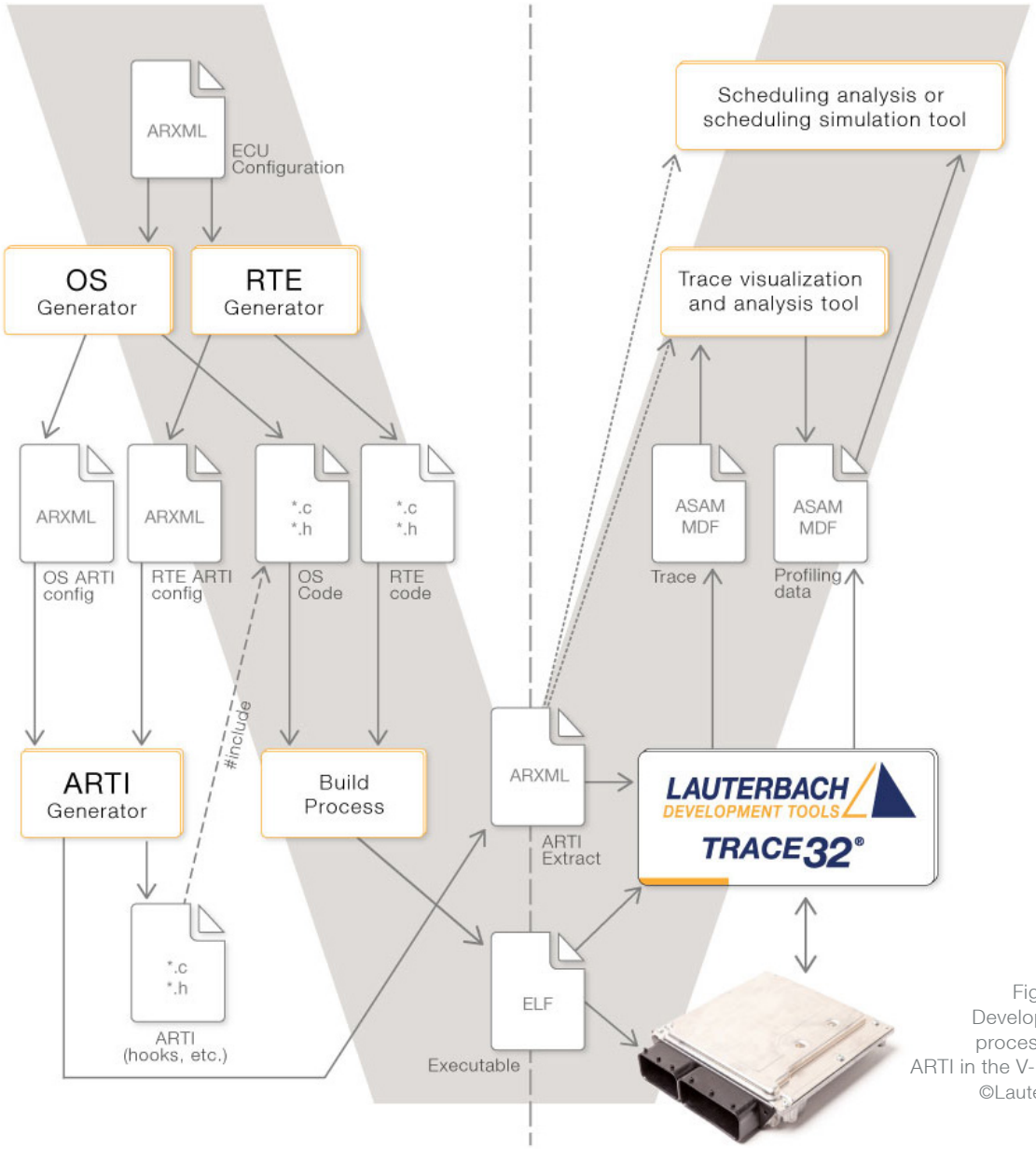


Figure 3: Development process with ARTI in the V-model ©Lauterbach

TRACE32® uses the ARTI file (arxml) generated during the build process to display all relevant debug information in a suitable manner. The recorded trace data can also be exported as ASAM MDF (Measurement Data Format) and then processed further using timing tools. Figure 3 shows the workflow.

ZF has incorporated these timing measurements and analyses into the development process. The extended analysis options enable the developer to check the effects of functional changes on the time behavior in a simple way. This procedure is essential for automotive applications because automotive software is typically structured in time slots – e.g. task one gets 5 ms, task two 3 ms – which must be strictly adhered to.

In addition to TRACE32®, ZF also uses Vector's TA Tool Suite [1] for automatic requirement analysis, for which Lauterbach has developed an appropriate export format for its trace data. For example, if a task is allowed to take 10 ms, TRACE32® provides the actual runtime statistics (e.g., 11 ms), while the TA Tool Suite reports a violation of the requirements (11 ms > 10 ms) at a higher level.

ZF's Step from Real to VLAB Virtual Target

In the next step, ZF wanted to integrate the previously described procedures into a continuous integration process so that it could constantly monitor the extent to which changes to the software caused changes in runtime behavior. The goal was to be able to run the tests completely automatically on a server – 24 hours a day, 365 days a year, if necessary. So, when a developer at ZF commits code to the repository, a CI server not only initiates a build and documents the results of the build, but also automatically checks the impact on timing.

Making a test bench suitable for the project, including all the necessary debug and trace tools, permanently available for a CI service proved to be a major challenge. It therefore made sense to replace the real test bench with a virtual test bench including a virtual ECU with virtual AURIX™ MCU. A pleasing side effect: virtualizing the microcontroller eliminates hardware limitations with respect to trace, since a model naturally does not rely on hardware implementations such as MCDS to generate trace data. Thus, such a setup is even usable for projects that do not have the appropriate hardware.

The AURIX™ Virtual Development Machine (VDM) is supplied by the Australian Semiconductor Technology Company Pty Ltd (ASTC). Called VLAB AURIX™ Toolbox, ASTC's product[2] provides a set of simulation models for the AURIX™ TC2xx and AURIX™ TC3xx MCUs, covering not only the TriCore™ CPUs but also the peripherals and other programmable cores, and supporting e.g. advanced debugging for GTM with disassembly, tracing and breakpoints. VLAB runs on the CI server and ZF wanted to do the timing measurements on it as well. Even if the simulation is of course slower than a real target, this does not matter because the measurements can run at night.

For ZF, this presented two challenges. The first was that Lauterbach's TRACE32® had to be connected to VLAB in some way (Figure 4) so that it could simply debug the simulation from the PowerView software [3]. The second challenge, which turned out to be much greater, was to be able to trace the VLAB model at all.

Thanks to intensive cooperation between the development teams at Lauterbach and ASTC, the PowerView connection was implemented promptly. The MCD API, which ASTC had already implemented in the simulator by default, was used as the interface.

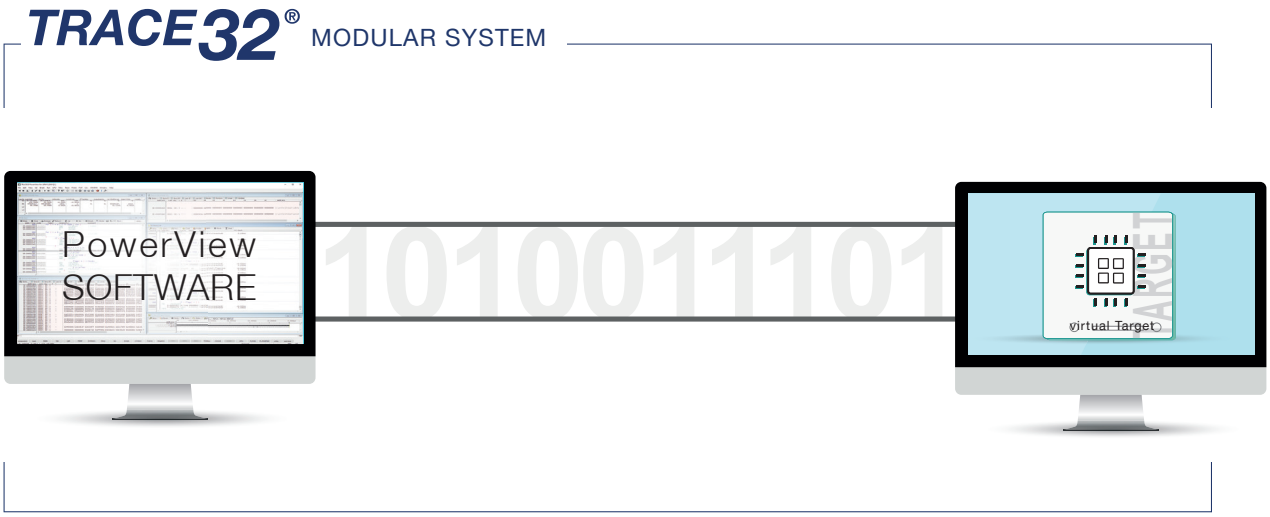


Figure 4: Debugging virtual targets with TRACE32. ©Lauterbach

The MCD API [4] was developed as part of a joint project between ARM, Infineon Technologies, Lauterbach, NXP Semiconductors, STMicroelectronics and TIMA Laboratory. It was defined to provide debugging tools with a unified debugging interface for both real hardware and software simulations. This allows application development to begin early in the design flow of an SoC platform without having to switch to different debug tools when moving from virtual prototypes to real hardware.

In addition, the MCD API supports multi-core debugging, which is a 'must have' due to the complexity of today's SoC designs. The MCD API is a powerful but simple C interface. It has a number of sub-APIs that provide functions for target connection, retrieving information from the target system, trigger support e.g. for breakpoints, memory and register accesses, and communication channels, among others.

Tracing via MCD API as a Challenge

In contrast to debugging, tracing the VLAB model via the MCD API did not work, although one of the sub-APIs provides a generic trace interface with predefined trace sources. The prerequisite for this to work is that the simulation first generates and provides the trace data and second provides the ability to retrieve this trace data via the API. The VLAB model provided inconsistent trace data at the beginning of the project due to an MCD API not being 100% implemented. However, what was even more serious was the unimplemented data trace, which is essential for ARTI. The simulation recorded the program run, but no data. However, it was imperative that TRACE32® be able to tell the VLAB simulation to selectively record certain data, such as when writing to a specific memory location.

As a result, in the interest of and at the request of ZF, Lauterbach worked with ASTC beginning in March 2022 to implement the missing data trace as well as to eliminate the inconsistencies in the MCD API. Lauterbach provided this support at no cost to ZF as part of customer support and did not charge ASTC.

By August 2022, ASTC's revised model, including data trace, was ready. Figure 5 shows an example of the configuration for recording variables with Lauterbach's PowerView software.

However, another challenge unexpectedly arose. The TC397XE is known to contain six cores, all of which should be recorded during the trace. In the scenario at ZF with the selective ARTI trace, core 0 delivered thousands of relevant messages, while the other cores delivered almost none, even though they were not idle but also processing workloads. They just did not generate as many relevant trace messages as core 0.

Figure 6 shows such a scenario reduced to three cores for simplification reasons. At time t0, all three cores deliver a message. At times t1, t2 and t3, however, only core 0 delivers a message, while cores 1 and 2 deliver their first further message after the one transmitted at time t0 only later, namely at time t4.

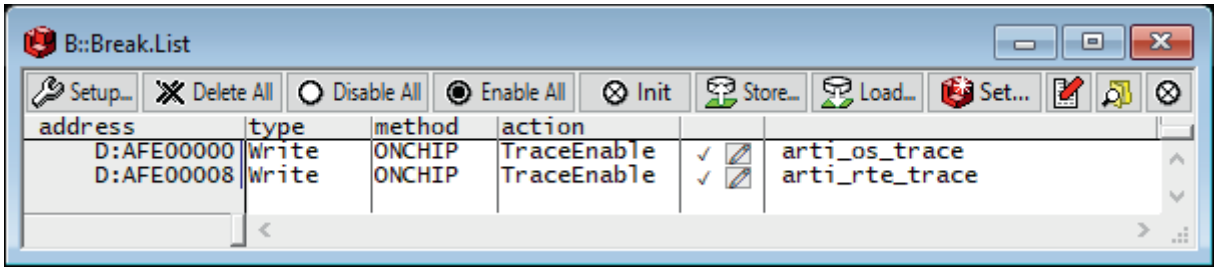


Figure 5: Variables recorded in the trace. ©Lauterbach

Now, if all cores were queried one after the other at time points t0, t1, t2, t3 and t4 respectively, the messages came in a wrong temporal order (Figure 6 above). The correct order would be, after querying Core 0, Core 1 and Core 2 at time t0, four more messages from each Core 0 at t1, t2, t3 and t4, and only then Core 1 and Core 2 again, also at t4 (Figure 6 bottom).

Lauterbach had subsequently worked on putting the timestamps in the correct order before exporting, which is a key requirement of Vector's TA Tool Suite for the timing analyses. By January 2023, everything was running to ZF's complete satisfaction.

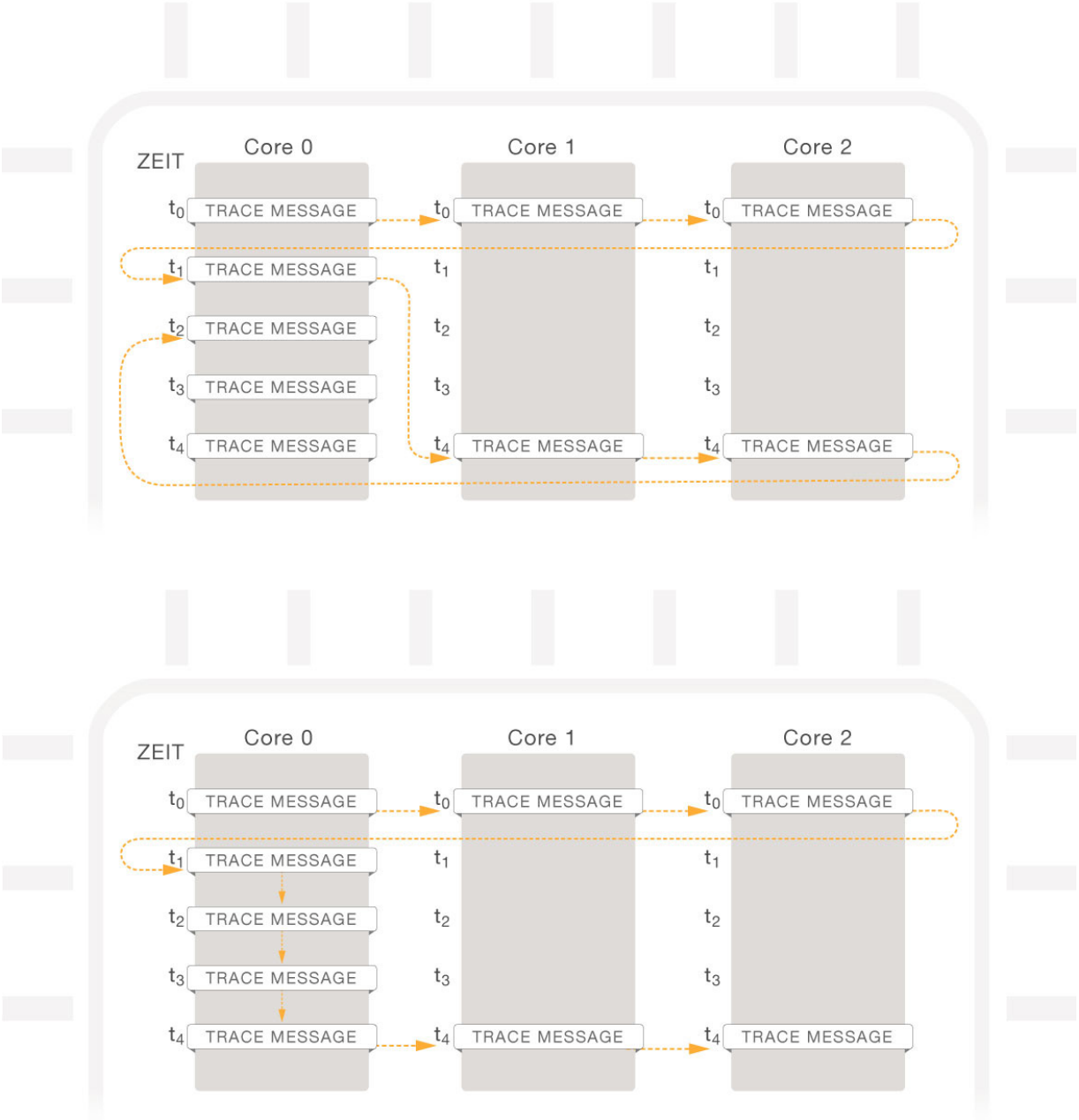


Figure 6: Asymmetrically distributed trace messages across three cores. ©Lauterbach

Conclusion and Outlook

ZF has been successfully using ASTC's VLAB Virtual Development Machines, on which its own application runs, since January 2023. Lauterbach's PowerView is used as the front end for virtualization, and program and data traces are used to record timing behavior and export it to Vector's TA Tool Suite.

Of course, for certification reasons, everything must also be traced on a real target before delivery. For this purpose, ZF uses the on-chip trace on the one hand, since the trace buffer at ARTI is sufficient up to a recording duration of 1 second or even longer, and on the other hand Lauterbach's CombiProbe [5] hardware module for even longer recordings.

The added value for ZF is obvious: on the one hand, a real target is no longer required at the beginning of a development project; and on the other hand, nightly automated tests with equally automated requirement analysis can be performed on the CI server. This means that every software change is automatically checked immediately for timing behavior.

Currently, the trend in the automotive industry is clearly moving in the direction of consolidating multiple applications onto one ECU. In light of this, it is all the more important to constantly check during the development process whether the ECU is still able to handle the load at all or whether the timing requirements can no longer be met. In the past, in the age of „simple" single-core CPUs with in-order instruction execution, there were mathematical methods that made it easy to calculate the runtime of individual functions.

Today, in the age of multi-core SoCs, multi-level cache architectures and out-of-order instruction execution, this no longer works. The only thing that really helps here is concrete measurement. This successful project at ZF is therefore likely to set a precedent in a similar form throughout the automotive industry.

REFERENCES:

[1] The TA Tool Suite from Vector: https://www.vector.com/ta-tool-suite

[2] The virtual AURIX™ from ASTC: https://vlabworks.com/vlab-vdms/

[3] Lauterbach's PowerView debug and trace software: https://www.lauterbach.com/products/software/trace32-powerview

[4] MCD API download: https://www.lauterbach.com/products/software/debugger-for-simulators/mcd-api

[5] Lauterbach's TRACE32® CombiProbe: https://www.lauterbach.com/products/trace-extensions/combiprobe