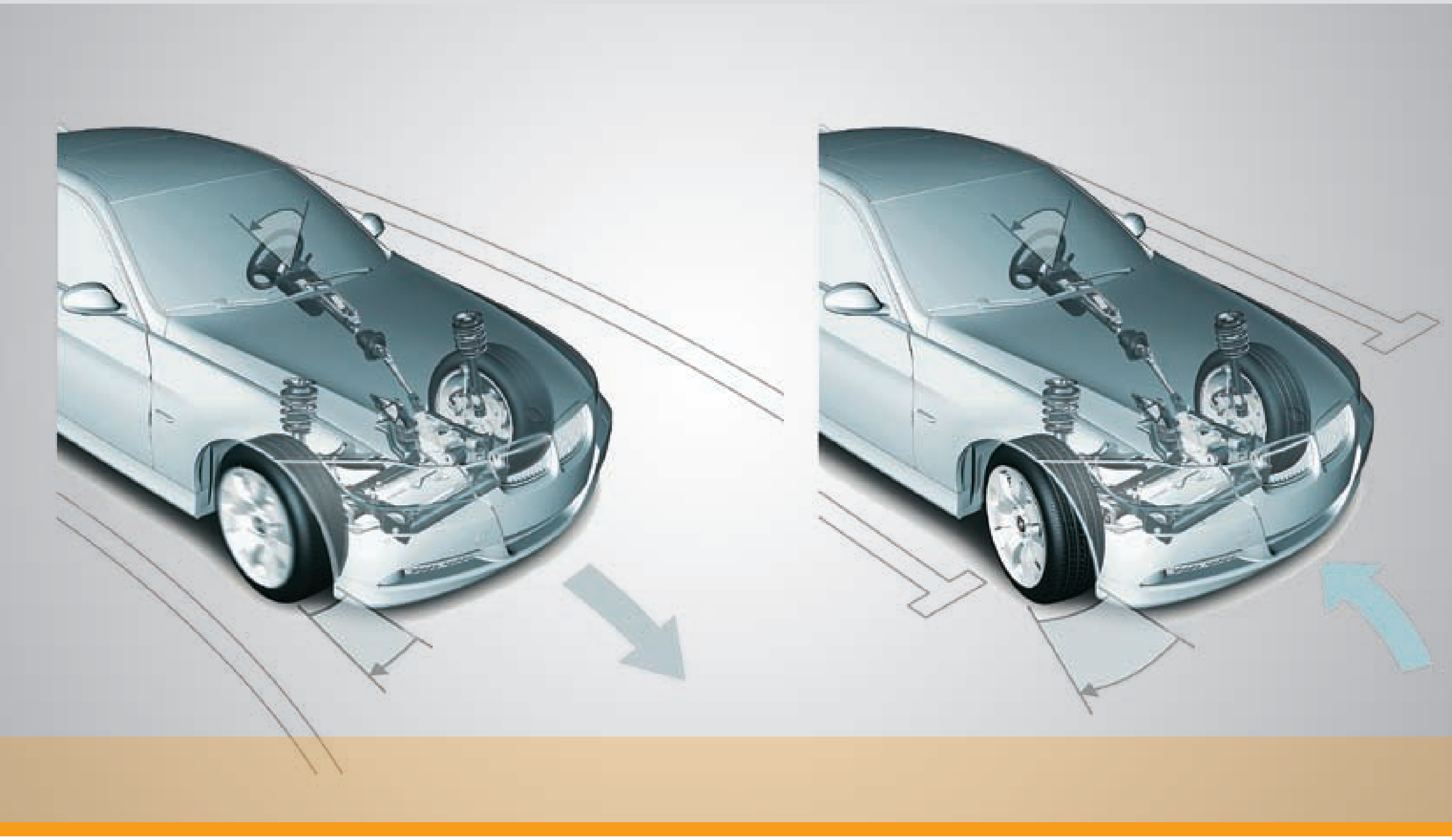

The Right Timing Analysis Tools Increase Safety and Productivity

It is surprising how many embedded systems are being developed without keeping track of the software's real-time behaviour. Problems caused by deficient timing are mostly hard to detect and even harder to solve. Such problems did not appear during the development of BMW´s active front steering, though. Right from the start of software design, the real-time behaviour was considered and was ensured continuously through measurement and analysis. The obtained results also offer a view of future requirements and solutions for systematic timing consideration during the development process.

1 Introduction

When implementing the BMW 5 series active front steering, BMW entered new territory. For the first time, electronically controlled steering intervention was authorized in a large-scale project in order to enable higher agility and new dynamic safety functions. A stable and predictable timing behaviour of the software is one of the measures taken to fulfil these safety requirements. Like the software's functionality design ("what happens?"), also the timing behaviour needs to be defined exactly ("when does it happen?"). This may sound trivial, as it is possible to define when which part of the software is calculated through configuration of the operating system. However, the interesting question in this context is: Can it be guaranteed in all cases that there is always sufficient computing power to comply with the requirements?

2 Technical and Commercial Aspects

External experts were consulted to answer these and other questions regarding real-time. Since the first generation of active front steering, Gliwa GmbH has been providing measurement solutions for capturing, and their tool “traceGURU“ for the visualization and analysis of real-time behaviour. Both are key to find software errors and optimize execution time. Real-time analyses are carried out continuously in order to detect possible problems early and to document the software’s execution time during the course of the project. “traceGURU“ reports have become an inherent part of BMW´s integration level approval documents. The approach also allows predicting future computing power requirements. When the development of the third generation of active front steering started, for example, real-time analyses of the previous project and of the planned additional functions showed that one of both CPUs could be substituted by a considerably cheaper (flash-less) derivative. The prediction regarding the system processor load had an error margin of only 3 % and the decision for the cheaper processor proved to be the right choice.

Such predictions are enabled through the consequent use of measurement and analysis tools – the trend from describing and understanding to controlling is clearly recognizable. As a result, BMW established another safety level for the third generation of active front steering, the so-called scheduling analysis. It provides mathematical verification of real-time requirements. BMW uses Symtavision’s scheduling analysis tool “SymTA/S” which finds and examines timing corner cases, which do not appear frequently in reality but are critical. “SymTA/S” can do this even when such cases were not observed during measurements. This may sound paradox but is enabled through the mathematical analysis of the system’s timing features (see section 4). Automated reports document the results in tables and timing diagrams – developers are not confronted with the basic mathematics.

The open interfaces for various input data make “SymTA/S” easy to use. For example, “traceGURU“ measurements are carried out at a test bench or inside the car first. The results are exported as XML files and are imported and analyzed by SymTA/S. The tool systematically verifies all timing corner cases. In the best case, it simply gives “green light”. In case any problems occur, it detects the cause of error. This completely automated workflow is integrated in the test environment as an inherent part of system verification, **Figure 1**.

Besides the various technical and commercial aspects which militate in favour of keeping track of embedded systems timing, there is another reason that has been gaining importance over the last years: product liability makes it virtually indispensable not to save at the wrong end when it comes to the safety of embedded systems.

3 Real-time Measurement

The operation of “traceGURU“ can be compared with that of a storage oscilloscope in various aspects. However, OS data are recorded, namely the points in time for the activation, start, and end of tasks, interrupts, and processes/runnables. These data allow to reconstruct, to chart, and to analyze the real-time situation during the recorded time period. **Figure 2** visualizes such a situation: the state of tasks and interrupts are displayed on a timeline. The “ready” state is displayed in light grey, the “running” state in dark grey. Workflows, processor load, distribution of processor load, et cetera, can be identified qualitatively at a glance. For example, Figure 1 shows the timing of SPI interrupts relative to the tasks. Besides tasks and interrupts, the figure also displays user events (green lines) and a stopwatch (blue bar). They allow the user to record custom data including their relative timing, and to measure arbitrary code segments.

An exact “quantitative” analysis can be carried out through the generation of a report – among other things – which documents the values of the measured “worst case execution times” (WCETs) of the respective tasks, processes/runnables, and interrupts and many more timing parameters. Gross as well as net execution times are recorded. In the latter case, the duration of preemptions is deducted.

Future projects will use Gliwa GmbH´s completely new developed measuring software “T1”. It carries out the analysis directly on the ECU in real-time. If required, the results – e.g. the measured WCETs of tasks – can be retrieved over a hardware interface. For this target-side online analysis it is crucial that ALL interrupts and tasks are considered and that the required bandwidth to the outside world can still be chosen freely – as only the results are transmitted, this work step can be carried out slowly.

Optionally, these results can be stored in non-volatile memory. That way real-time recording can be performed as a side effect during testing without extra effort. An interaction with the measurement engineering is not necessary and

The Authors

Dr. Marek Jersak
is co-founder and CEO of Symtavision GmbH in Braunschweig (Germany), responsible for strategy and business development.

Dr. Kai Richter
is co-founder and CTO of Symtavision GmbH in Braunschweig (Germany), responsible for technology and products.

Hans Sarnowski
is software engineer for active front steering for BMW AG in Munich (Germany).

Peter Gliwa
is CEO of Gliwa GmbH and provides real-time expertise consulting with focus on automotive in Munich (Germany).

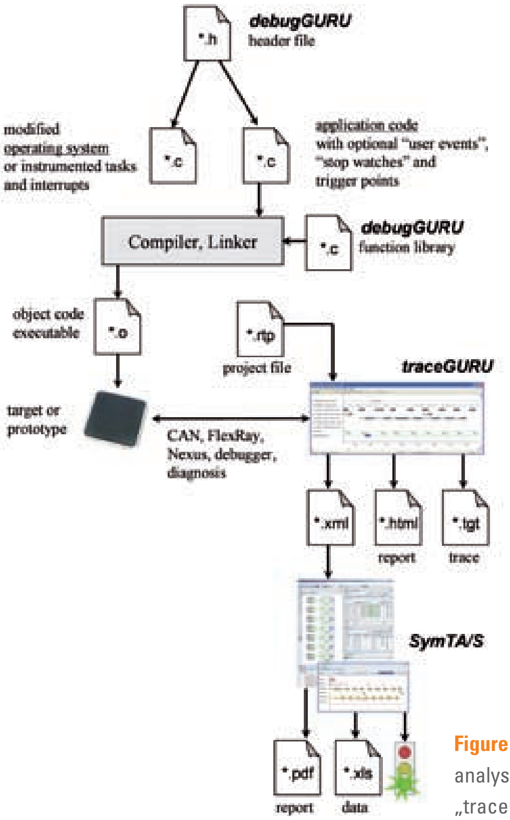

Figure 1: Integrated timing-analysis flow using the tools „traceGURU""and „SymTA/S"

the developer will not miss a single real-time error.

The online analysis affects the runtime of the target system only slightly as it is decoupled from the capturing of measurement data – in comparison to competing approaches.

Depending on the used OS, the recording of real-time data is carried out either through modification of the OS or the instrumentation of tasks and interrupts. Overloading the "TASK" (...) and "ISR" (...) macros is possible for "OSEK" and Autosar operating systems.

In principle, any OS that knows the states "suspended", "ready", and "running", or offers comparable states, can be measured. The hardware interfaces "Debugger", "Nexus", and "CAN" enable the transmission of the recorded real-time data. From spring 2009 on, this service will also be available via Diagnose and Flexray.

4 Scheduling Analysis

Scheduling analysis goes one crucial step further in comparison to measurement, by systematically detecting critical performance bottlenecks and error situations, even if they were not observed during measurement. This is possible because the tool SymTA/S covers the full range of dynamic variables that can occur during system operation, instead of considering only selected "frozen" situations observed during tracing. Specifically, SymTA/S considers dynamic interrupts and other events as well as varying code execution times. „SymTA/S"„ also considers data-dependencies between tasks, which are systematically captured as event models. The actual scheduling analysis is then done in two steps. „SymTA/S"„ first decomposes the imported traces into their individual parts (tasks, interrupts ...). In the second step, the parts are re-composed in a worst-case way which takes all dynamics into account. As a result, "SymTA/S"„ construct those situations when load and execution times are maximal.

As an example, consider the "10 ms task". Figure 1 shows a trace where the 10 ms task is interrupted by 4 CAN interrupts. In other traces, a larger concentration of CAN interrupts was observed, but many of these occurred in-between two executions of the 10 ms task. „SymTA/S"„ now automatically re-composes this information from different traces into a situation, where the number of CAN interrupts preempting the 10 ms task is maximized. The timing diagram in **Figure 3** shows, that

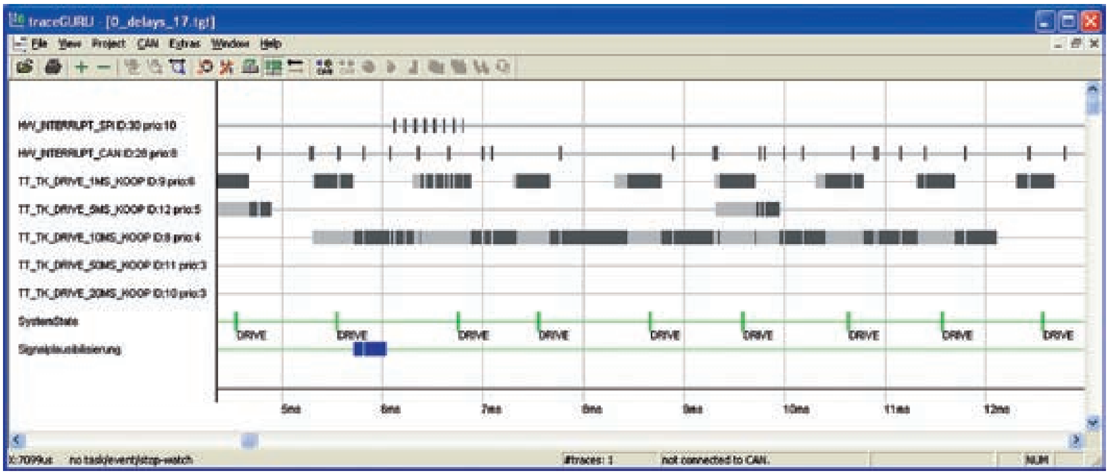
traceGURU - [0_delays_17.tgt]
File View Project CAN Extras Window Help
HW_INTERRUPT_SPI ID:30 prio:10
HW_INTERRUPT_CAN ID:26 prio:8
TT_TK_DRIVE_1MS_KOOP ID:9 prio:6
TT_TK_DRIVE_5MS_KOOP ID:12 prio:5
TT_TK_DRIVE_10MS_KOOP ID:8 prio:4
TT_TK_DRIVE_50MS_KOOP ID:11 prio:3
TT_TK_DRIVE_20MS_KOOP ID:10 prio:3
SystemState
Signalplausibilisierung
DRIVE
DRIVE
DRIVE
DRIVE
DRIVE
DRIVE
DRIVE
DRIVE
DRIVE
5ms
6ms
7ms
8ms
9ms
10ms
11ms
12ms
X:7099us
no task/event/stop-watch
#traces: 1
not connected to CAN.
NUM

Figure 2: „traceGURU" as an „oscilloscope of the operating system "

the 10 ms task can be preempted by up to ten CAN interrupts. Even though this situation was not observed during tracing, it is not impossible, and thus needs to be considered for reliable timing analysis.

In conclusion, the systematic construction and visualization of such critical corner-cases significantly improves test-coverage of ECUs and systems. Not until it is shown that time budgets and deadlines are met even in these critical cases, can reliable system operation in all situations be assumed, Figure 4.

“SymTA/S“„ achieves the necessary accuracy through analysis libraries specially adapted to the properties of the system under design. “SymTA/S“„ supports different OSEK variants, Autosar-OS as well as CAN and Flexray for networking. Each library is self-contained, but all libraries can be freely combined to model and analyze larger, distributed systems.

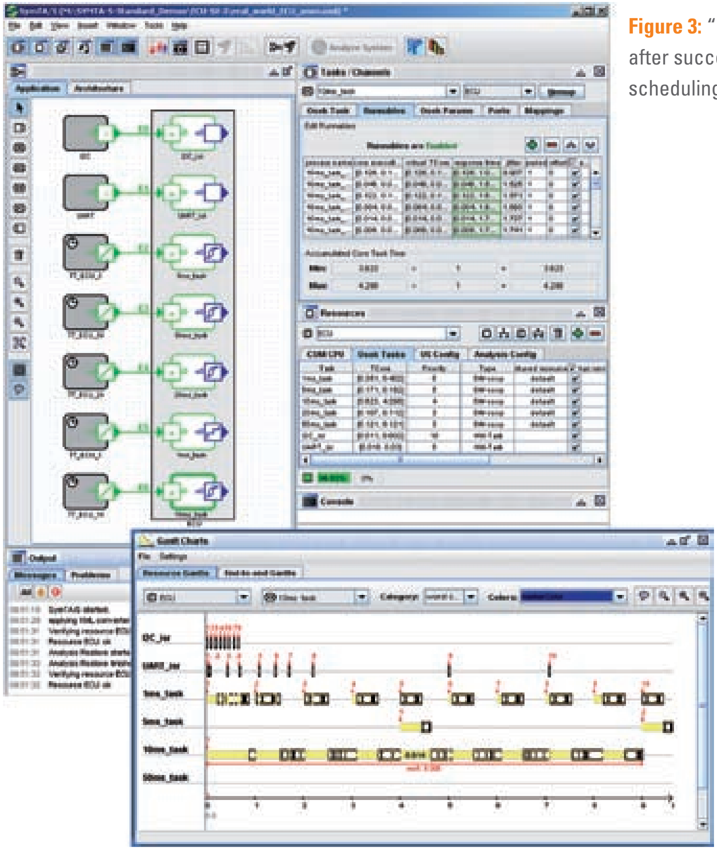

Figure 3: “SymTA/S“ after successful scheduling analysis

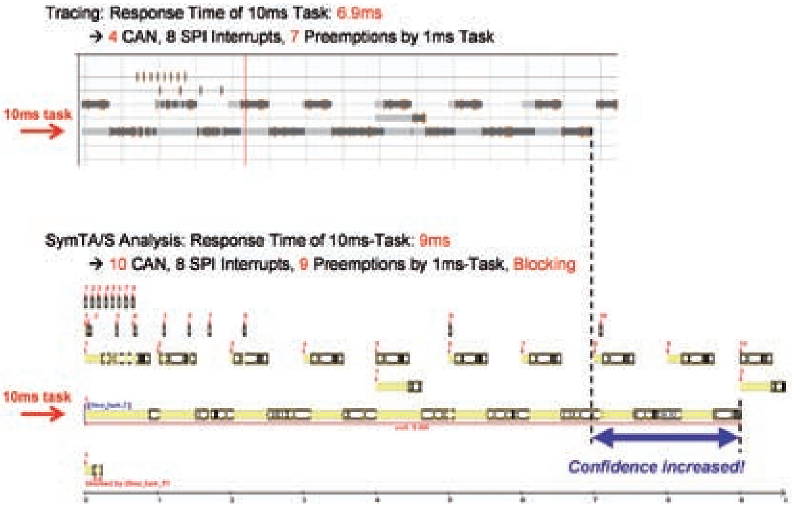

Figure 4: Comparison of measurement and “SymTA/S“ scheduling analysis shows improved test coverage

More and more, designers want to analyze system timing and performance during or before software development using virtual prototypes – at a time when tracing is not feasible. Here, “SymTA/S“„ shows its second strength. “SymTA/S“„ does not require executable code. Therefore, “SymTA/S“„ can perform integration analysis early based on both estimated code execution times (for new functions) and available times (legacy functions). In particular, “what-if analysis” is possible to determine the influence of longer or shorter execution times or of different processor clock speeds on overall performance and timing.

Furthermore, “SymTA/S“„ makes automatic design space exploration and optimization easy at very early design stages (platform- and architecture-design, function partitioning and integration). This allows to define key design constraints, such as software budgets or dimensioning of hardware.

5 Conclusion

The BMW active front steering example clearly shows the advantages for reliability and cost-optimization gained through systematic use of timing analysis tools during ECU development. Today, the topic of timing is still young in the automotive industry, but is rapidly gaining in importance. This view is supported by the current work of the Autosar timing team (www.autosar.org) with significant contribution from Symtavision, as well as the European projects “TIMMO” (www.timmo.org) and “ALL-TIMES” (www.all-times.org). The automotive industry needs to manage timing as a necessity for reliability and safety under strict cost-constraints for ever more complex systems. Timing influences decisions in E/E-architecture, software-architecture and function integration. Timing analysis helps to manage the associated risks. Systematic integration of timing analysis into the design process will thus be one key to future automotive success. ■