Timing Correctness and Model-Based Software Development for Safety-Critical Automotive Applications
- An Integrated, Tool-Supported Workflow

Kai Richter, Symtavision GmbH
Christian Ferdinand, Reinhold Heckmann, AbsInt GmbH
Patrick Frey, ETAS GmbH
Overview / Agenda

- Automotive ECU software & timing issues
- ECU software development with ASCET
- Comparison of different methods dealing with timing
- The Timing Analysis Tools from AbsInt & Symtavision
- Tool Integration: ASCET + aiT + SymTA/S
- Interfaces and Tool Flow
- Example: Engine Control System
- Conclusions
Automotive ECU software

- **Real-time critical functions**
  - Control applications: Timing as inherent quality characteristic
  - OS tasks and processes have to finish within deadlines
  - Minimize/Maintain end-to-end deadlines (sensor → actuator)

- **Safety functions**
  - Hard deadlines
  - Guaranteed timings simplify certification

- **Examples from different vehicle domains**
  - Power train: Engine Sync. Tasks
  - Chassis: Active Damping System
Market trend

Embedded market today
~132 B€

2015 Embedded market
~230 B€

~ 6% / year

Source: JD Powers/Frost & Sullivan

New Functions

New Architectures

New Technologies

Consequences

- More complexity, more load and more integration problems
- Integration problems are very often timing problems
Example 1: Engine-Control ECU

Problems

- Verifying Performance and Timing for all engine speeds (RPM)
- Avoiding Deadline Overruns (would lead to ECU reset)
- Optimizing ECU performance and cost for different markets

<table>
<thead>
<tr>
<th>Use-Case</th>
<th>Effort Comparison</th>
<th>Time saved</th>
<th>Cost saved*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tune performance for new software version</td>
<td>4-6 days</td>
<td>50%</td>
<td>250 k€ / year</td>
</tr>
<tr>
<td></td>
<td>2-3 days</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design ECU configuration to achieve performance goals</td>
<td>2 months</td>
<td>75%</td>
<td>120 k€ / year</td>
</tr>
<tr>
<td></td>
<td>2 weeks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Assumption: 1000 new SW versions / year, 10% requiring performance fine-tuning. 4 new ECU configurations / year. Cost for engineer 1k € / day.
Example 2: Electronic Power Steering ECU

Problems

- Verifying Performance and Timing for all critical cases
- Safeguarding against liability claims
- Optimizing ECU performance and cost (use of cheaper CPU)

<table>
<thead>
<tr>
<th>Use-Case</th>
<th>Main benefit using</th>
<th>Cost saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Availability &amp; Safety</td>
<td><strong>Verify Availability &amp; Safety</strong> (analytical proof for reaction times - IEC 61508)</td>
<td>helps secure contract</td>
</tr>
<tr>
<td></td>
<td><strong>Main benefit using</strong></td>
<td><strong>Cost saved</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Systematic Timing Analysis</strong></td>
<td><strong>Cost saved</strong></td>
</tr>
<tr>
<td>Select hardware (resource availability at any time without over-dimensioning)</td>
<td><strong>Meet critical OEM requirement</strong></td>
<td><strong>Cost saved</strong></td>
</tr>
<tr>
<td></td>
<td><strong>CPU-cost savings for each ECU</strong></td>
<td><strong>Cost saved</strong></td>
</tr>
</tbody>
</table>

*Over product life-cycle.*
Timing Issues with automotive ECU software

Why timing matters …
- Not being "on time" can cause
  - deteriorated control quality / instability
  - system failure
- Wasting ECU resources increases hardware (=production) costs

Technological challenge
- How to verify timing characteristics of embedded software?
- In principle, different methods exist to handle timing …
- … however not all can be used for verification

Methodological challenge
- How to integrate timing verification in established development processes?
- Main driver: Early problem detection saves development costs
Automotive ECU software development with ASCET

- Model-based specification & design
- Modular software development (modules, processes)
- Functional Validation phases:
  - Simulation ➔ Virtual & Rapid Prototyping ➔ Target Implementation
- Implementation through automatic code generation
- Employment of real-time OS (ERCOSek, RTA-OSEK)

Implemented ECU system
- Has function and timing properties
- Besides functional properties, has to satisfy timing requirements
- How can timing properties be verified?
## Methods to gain confidence in ECU system timing

<table>
<thead>
<tr>
<th>Method</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing Measurement / Testing</td>
<td>+ easy to employ</td>
<td>- applicable late in development</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- instrumentation overhead</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- not reliable due to <strong>incomplete coverage</strong></td>
</tr>
<tr>
<td>Timing Simulation</td>
<td>+ works without target hardware</td>
<td>- applicable late (needs entire code incl. BSW, OS, COM, etc.; like test)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- needs timing accurate simulators (expensive)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- not reliable due to <strong>incomplete coverage</strong></td>
</tr>
<tr>
<td>Static Timing Analysis</td>
<td>+ model-based approach</td>
<td>- perceived complex and not very accurate (- new technology)</td>
</tr>
<tr>
<td></td>
<td>+ reliable through <strong>full corner case coverage</strong></td>
<td><strong>But: Much recent progress and easy-to-use tools available!</strong></td>
</tr>
<tr>
<td></td>
<td>+ supports incomplete designs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+ abstract environment models</td>
<td></td>
</tr>
<tr>
<td></td>
<td>➔ Applicable from <strong>early</strong> planning to <strong>final</strong> verification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>➔ Supports <strong>what-if analysis</strong></td>
<td></td>
</tr>
</tbody>
</table>
Example: Traditional Measurement vs. Systematic Analysis

- Traditional Measured 10ms task: Response time 6.9ms
  - 4 CAN, 8 SPI interrupts, 7 preemptions by 1ms task

- Systematic Analysis of 10ms task: Worst-case response time 9ms
  - 10 CAN, 8 SPI interrupts, 9 preemptions by 1ms task, blocking

Coverage increased, Risk avoided
Structuring the Influences on Timing

What do we need?

Execution Time Analysis

Unpreempted Execution of OS Tasks, processes, ISRs, etc...

Interrupt Occurrence

Priority Inversion due to Shared Resource Locking

Task Preemption

Priority Inversion due to Cooperative Scheduling
ETAS Partners for Timing Analysis

- Spin-Off from Saarland University
- Founded 1998
- Key Competence: Execution Time Analysis

- Spin-Off from TU Braunschweig
- Founded 2005
- Key Competence: Scheduling Analysis
aiT Worst Case Execution Time Analyzer

Benefits:
- reliable analysis due to full corner case coverage
- supports incomplete designs, code fragments
- delivers execution time profiles incl. call hierarchy
- comprehensive visualization
**SymTA/S Scheduling Analysis Tool Suite**

**Benefits:**
- Reliable task and end-to-end analysis due to full corner case coverage
- Supports incomplete designs, abstract systems
- Comprehensive visualization

**Optional:**
- Detects bottlenecks through sensitivity analysis
- Automatic optimizations (multiple objectives)

**System model:**
- task structure
- OS configuration
- execution times

**Perform System-level timing analysis**

---

Recall: Traditional Measurement vs. Systematic Analysis

- Measured 10ms task: Response time 6.9ms
  - 4 CAN, 8 SPI interrupts, 7 preemptions by 1ms task

- Systematic Analysis of 10ms task: Worst-case response time 9ms
  - 10 CAN, 8 SPI interrupts, 9 preemptions by 1ms task, blocking

Coverage increased,
Risk avoided
Tool Integration: ASCET + aiT + SymTA/S
Example: Engine Control System
INTEREST Automotive network validator (with AbsInt, EB, Esterel, ETAS, Symtavision)
INTEREST Automotive network validator (with AbsInt, EB, Esterel, ETAS, Symtavision)
Tool Coupling can be Demonstrated in Prototype Stage

- Tool coupling is shown on exhibition
- Feed-back is welcome

---

**ASCET-SymTA/S**
End-to-end Timing Analysis for Electronic Control Systems

- Timing verification of ECU software developed with ASCET
- Control functions and timing design in a high-level formalisms are setup in the design process
- Determination of OS schedule and model makes it accurate to vary the available resources
- Calculation of available resource for additional functions
- Tools' internal data structures and decision model allows focus on the critical design aspects

---

**ASCET-II/T**
Execution Time Analysis for Control Functions

- All worst case execution time analysis of ECU code generated by ASCET
- All ECU required
- Assumptions and safe bounds are set on execution times of functions based on data flow of the program
- All worst case execution time analysis of ECU system
- Streamlined and expert model for quick or in-depth analysis
- Support of ETAS/DE and RTA-DDBK operating system

---

In real-time control systems, the timely completion of each individual task is essential. Such a system is only effective in the specified time frame in order to ensure correct system behavior. To design a real-time control system, it is essential that the worst-case execution time (WCET) of each task is known.

**Worst Case Execution Time Determination**

In common practice, empirical WCET determination methods are used to validate the correctness of a real-time system by repeated execution time measurements of an instrumented test program. A test program is designed to inflame and modify the system under test to some extent. This approach is tedious and usually not feasible. In many cases it is impossible to prove that the worst case conditions really determine the maximum possible execution time.

The WCET tool chain allows for analysis based on full code and pipeline models. The static code and pipeline model is used to extract the critical path of the test program. Based on the critical path, the worst-case execution times are evaluated. If a path through the critical path is not critical, then an analysis of the pipeline is done to determine if it is possible to speed up the overall execution time.

**ASCET-II/T Tool Coupling**

ASCET-II/T is a coupling of the best-in-class WCET analysis and model-based software development tool. ASCET-II/T transparently integrates all Worst Case Execution Time Analysis. The tool can be used to evaluate the overall system performance and determine system utilization. The analysis is based on analysis of the critical path and the worst case execution time analysis of ECU code. The tool has been automatically generated (see Figure 1).
Conclusion

- The ASCET + aiT + SymTA/S tool integration provides ASCET users **comfortable access to systematic timing analysis**
  - Analysis covers all critical timing corner cases
  - Steers ASCET users to the timing critical parts of the project
  - Supports optimization and fine-tuning

- Benefits:
  - Increases safety and limits liability
  - Avoid timing problems early instead of fixing them late
  - Speed-up ECU development and system dimensioning
  - Reduction of component cost, avoidance of over-dimensioning

- Is becoming established practice / state-of-the-art

→ Experience tools, tool coupling, and process on the exhibition!
Thank you!