Overview

• performance challenges in automotive embedded system
• timing models and compositional analysis
• embedding in design process
• some recent enhancements
• conclusion
Overview

- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion

Formal Performance Analysis Applications

- formal models for performance analysis optimization are in use for very different types of embedded system
  - distributed networks
  - MpSoC

55 ECUs & 7 Buses of 4 types with Gateways

source: Bekooj – this tutorial
Example: Automotive Embedded Systems

- Complexity challenge
  - Hundreds of functions
  - Networked control
  - Many suppliers
  - Heterogeneous
- Design challenges
  - Supply chains
  - Systems integration
  - Verification
- Software standardization: OSEK → AUTOSAR

Automotive Networked Systems

- Distributed networked system with complex end-to-end time constraints and numerous integrated functions on shared resources
Automotive Design Chain

OEM
- BMW, Daimler, GM, PSA, Toyota, ...
- global system, integration and network

specs ↓ ↑ ECU

ECU - Supplier
- Bosch, Delphi, Valeo, ...
- ECU responsibility

specs ↓ ↑ SoC

HW Component - Supplier
- Infineon, Freescale, ST, Toshiba, ...

© R. Ernst, TU Braunschweig, 2008

V Model

Requirements
System Design
Module Design
Function Design
Implementation

Requirements
Test
System Test
Module Test
Function Test
Integration

© R. Ernst, TU Braunschweig, 2008
V – Model and Timing

Requirements

System Design

System Timing Estimation

Network Timing Estimation

Module Design

ECU Timing Estimation

Function Design

ECU Timing Estimation

Function Test

System Test

Requirements Test

System Timing Verification

Network Timing Verification

Module Test

ECU Timing Verification

Function Test

AUTOSAR - Automotive Software Architecture

• SW-Components (SW-C)
  – encapsulate the applications

• Virtual Functional Bus (VFB)
  – communication mechanisms
  – interface to Basic SW

• Mapping
  – configuration and generation of RTE and Basic SW

• Runtime Environment (RTE)
  – VFB implementation on a specific ECU

• Basic Software (BSW)
  – infrastructural functionality on an ECU

Source: www.autosar.org
SW Component Execution

- Standardized RTE eases compiling & linking together several SW components from different teams/vendors

Vehicle Function

Timing Chains and Hand-Over Points

- Hidden timing chains and non-functional dependencies challenge predictability
- Timing verification required to support design process
Overview

- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion
Timing Model Hierarchy

- **System timing model**
  - performance of components integrated in a network

- **Component timing model**
  - activation function
  - component scheduling/arbitration

- **Task timing model**
  - execution load and timing
  - communication load and timing

---

Timing Model Hierarchy - Task

- **System timing model**
  - performance of components integrated in a network

- **Component timing model**
  - activation function
  - component scheduling/arbitration

- **Task timing model**
  - execution load and timing
  - communication load and timing

**Formal Modeling Fundamentals – Task Execution**

- **task core execution time** is the time needed to execute a given task when running alone on a processor

- task core execution time does *not* include
  - operating system overhead
  - the influence of other tasks
  - waiting and synchronization times for global resources
  - shared cache and memory access times (L1 cache often included)

- task core execution time is determined in different ways
  - estimated in early design phases
  - measured with a cycle accurate simulator (e.g. CoWare, Vast)
  - measured with instrumented code on a prototype (e.g. dspace)
  - formally analyzed using WCET analysis

**Formal Modeling Fundamentals – Communication**

- **core communication time** is the transmission time for a given message to be communicated over a link when no other communication is active

- core communication time does *not* include
  - arbitration (scheduling)
  - buffering
  - gateway, multi-hop or MIN timing overhead

- core communication time is determined in different ways
  - simulation or prototyping
  - **communication analysis** based on formal model of communication protocol
    - individually adapted to communication protocol
    - typically simpler than WCET analysis
Timing Model Hierarchy - Activation

- **system timing model**
  - performance of components integrated in a network

- **component timing model**
  - activation function
  - component scheduling/arbitration

- **task timing model**
  - execution load and timing
  - communication load and timing

---

Formal Modeling Fundamentals – Activation

- total task load, also called utilization of task i, \( U_i \), depends on activation function
  
  \[
  \text{total task load} = \frac{\text{load/task execution}}{\text{task activation frequency}}
  \]
  
  \[
  = \frac{\text{task core execution time}}{\text{task activation frequency}}
  \]
  
  - example: periodic task i with core execution time \( C_i \) and period \( T_i \)
    
    \[
    U_i = \frac{C_i}{T_i}
    \]

- what defines the task activation function?
  
  - application model (Simulink, SPW, LabView, ...)
  
  - environment model (reactive systems)
  
  - service contracts (max no of requests per time, ...)

  → typically application rather than platform dependent

  → platform can „modulate“ activation timing to avoid malfunction (e.g. traffic shaping, back pressure)

- two classes of activation – time activation, event activation
Activation Functions

- two classes of activation
  - time activation – tasks are periodically activated by clock
    - example: periodic sample in signal processing / control eng.
  - event activation – tasks are activated when event arrives
    - example: automata, flow graph

Events and Activation Dependency

- event activation requires event buffering
  - event queues
  - no event lost (under normal operation)

- time activation does not always require buffering
  - buffering requirements only depends on the application semantics
    - may allow to drop data
    - dropping data may be required to obtain latest sample
Modeling Events as Streams

- in system level formal performance models, events are modeled as *streams* rather than as sequences of individual events
- examples
  - a clock is given by its period rather than as a sequence of clock ticks
    → clock can be modeled as an event stream
  - a sampled sensor signal is modeled by the sample period and the sample jitter
- the event streams are defined as functions or as parameter tuples

Popular Event Stream Models – PJD

- standard event model used in classical schedulability analysis
  - event sequences modeled by three parameters, period $p$, jitter $j$, and minimum time interval $d$ between 2 events
  - important models that can easily be described
    - strictly periodic events (typically clock released)
    - periodic events with jitter
    - sporadic events
    - sporadically periodic events
  - covers a large class of applications
  - conservatively approximates more complex functions
Popular Event Stream Models - Arrival Curves

- arrival curves of the network calculus
  - captures the no. of event in a time interval $\Delta t$
  - $\alpha^l(\Delta t)$ is lower bound
  - $\alpha^u(\Delta t)$ is upper bound
- can be used to describe the standard event models
- reaches infinite values for $\Delta t \to \infty$
  - must be approximated or extended by periodic function for $\Delta t \to \infty$
- is approximated when event sequences become very complex, e.g. as a result of operations on event sequences

Arrival Curves - Example

Event Stream

number of events in
in $t=[0..2.5]$ ms

Arrival Curves

maximum / minimum
arriving events in any
interval of length 2.5 ms
Example 1: Periodic with Jitter

Example 2: Periodic with Jitter and Minimum Distance $d$

- **Arrival curves:**

\[
\alpha'(\Delta) = \left\lfloor \frac{\Delta - j}{p} \right\rfloor \\
\alpha''(\Delta) = \min \left\{ \left\lfloor \frac{\Delta + j}{p} \right\rfloor, \left\lfloor \frac{\Delta}{d} \right\rfloor \right\}
\]
Total Load of a Task

- with activation model and core execution time or (core communication time), we can now derive the total load of a task

\[ T_1 \quad T_2 \]

- the resource is not fully available to one task or communication, but is shared with others

Overview

- applications for formal performance analysis methods
- formal performance modeling and analysis principles
- modeling activation and event streams
- component analysis
- system analysis
- enhancements to the basic analysis
- summary and comparison
- conclusion
Timing Model Hierarchy - Component Timing

- System timing model
  - Performance of components integrated in a network

- Component timing model
  - Activation function
  - Component scheduling/arbitration

- Task timing model
  - Execution load and timing
  - Communication load and timing

Timing Effects of Scheduling/Arbitration

- Tasks execute longer than their core execution time
  - Time assigned to other tasks
  - Operating system overhead
  - Context switch, blocking, ...

- Response time of a task is maximum from time of activation to task termination

Example: Static priority preemptive scheduling
**Scheduling Analysis**

- different analysis algorithms
  - generalization of busy window algorithm (Lehoczky, Tindell) to fit general event model (Richter, Jersak, Henia, Racu, Ernst, Schliecker, et al.)
    - Tool SymTA/S
  - extension of Network Calculus to Real-time Calculus (Chakraborty, Wandeler, Künzli, Thiele, et al.)
    - Tool MPA

---

**Analysis uses “Busy Window” approach**

The analysis uses the busy window approach to determine the schedulability of a set of tasks. The busy window is the period during which the worst-case load is present, and it is used to calculate the maximum response time of a task.

- The busy window starts with the worst-case load.
- The busy window ends when the load drops below the worst-case load.
- The equations used in the analysis are:
  
  \[
  w_i(q) = q C_i + \sum_{j \in \text{hP}(i)} C_j \left[ \frac{w_i(q)}{T_j} \right]
  \]

  \[
  R_i(q) = w_i(q) - (q - 1) T_i
  \]

  increase \( w_i \) until fix point found where equations hold!
Busy Window Analysis

- very versatile approach
- has been extended to analyze even difficult scheduling strategies
  - round-robin, non preemptive, collaborative processes (e.g. OSEK), …
- can handle parameter dependent worst case (e.g. release offsets – time table)
- can handle stream queues and register communication
- window size increases with load (limited by deadline)
- this window „unrolling“ process can be considered as symbolic simulation

Importance of Context Switch Consideration

- context switch increases load → non load preserving
Time Table for Release Offset

- Release offset reduces peak load

Real-time Calculus

- Input Stream
- Task
- Service Model
- Load Model
Service Model (Resources)

Resource Availability

available service in $t = [0 .. 2.5]$ ms

Service Curves

maximum/minimum available service in *any interval* of length 2.5 ms

Greedy Processing Component (GPC)

Examples:
- computation (event – task instance, resource – computing resource [tasks/second])
- communication (event – data packet, resource – bandwidth [packets/second])
Greedy Processing Component

**Behavioral Description**

- Component is triggered by incoming events.
- A fully preemptable task is instantiated at every event arrival to process the incoming event.
- Active tasks are processed in a greedy fashion in FIFO order.
- Processing is restricted by the availability of resources.

---

**Greedy Processing Component (GPC)**

If the resource and event streams describe available and requested units of processing or communication, then

\[
C(t) = C'(t) + R'(t) \\
B(t) = R(t) - R'(t) \\
R'(t) = \inf_{0 \leq u \leq t} \{R(u) + C(t) - C(u)\}
\]

Conservation Laws
MPA-RTC – Scheduling - Examples

Fixed Priority Preemptive Scheduling

Time Division Multiple Access (TDMA)

\[ \beta \]

\[ \alpha_A \rightarrow \alpha'_A \]

\[ \alpha_B \rightarrow \alpha'_B \]

\[ \beta' \]

\[ \alpha_A \rightarrow \alpha'_A \]

\[ \alpha_B \rightarrow \alpha'_B \]

\[ \beta'_{s1} \beta'_{s2} \]

Delay and Backlog

\[ [\beta^l, \beta^u] \]

\[ [\alpha^l, \alpha^u] \]

\[ [\alpha^l, \alpha^u'] \]

\[ [\beta^l, \beta^u'] \]

maximum delay D

maximum backlog B

\[ \rightarrow \text{same solution for Busy Window analysis} \]
Timing Model Hierarchy – System Timing Model

- system timing model
  - performance of components integrated in a network
- component timing model
  - activation function
  - component scheduling/arbitration
- task timing model
  - execution load and timing
  - communication load and timing

System Analysis using Composition

- independently scheduled subsystems are coupled by data flow

⇒ subsystems coupled by streams of data
⇒ interpreted as activating events
⇒ coupling corresponds to event propagation
Compositional Analysis Principle

- environment model
- local analysis
- derive output event model
- map to input event model
- until convergence or non-schedulability

Symbolic Simulation or RTC

find fix point where input and output models converge

System-level Analysis Results

- end-to-end latencies
- buffer sizes
- system load
- ....

example: complex end-to-end latency analysis w. SymTA/S

source: Symtavision
Compositional Analysis Properties

- compatible event stream models allow to couple any number of blocks for local analysis
  → scalable

- fixpoint iteration automatically adapts to platform topology
  → easy integration and extension
  → RTC and SymTA/S analysis blocks have been shown to easily work together [KHT07]

- very short analysis time (few seconds) opens new opportunities in design space and robustness optimization

The Compositional Analysis „Landscape“

- Event model
  - PJD
  - Arrival curve
  - Symbolic simulation
  - System analysis
  - SymTA/S
  - MPA
  - RTC
  - Approximation

- Fix point iteration for composition
Further Performance Models

- timed automata have been used to explicitly model the task scheduling algorithm and OS interactions and then apply model checking to identify deadline violations
- can be more accurate in the individual component model but is computationally far more expensive
- work e.g. Madsen or Johnson
- can potentially be linked via common event stream models

source: Jan Madsen, MpSoC 2007

Overview

- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion
Performance Verification Flow- ECU

- application modeling tool
- code generator/manual design
- SW processes
- RTE configuration
- ECU integration
- composition performance analysis
- system performance parameters
- executables
- stimulation
- simulation tool
- tracing tool
- process performance parameters
  - timing
  - communication
- WCET analysis

Example: Safety-Critical ECU

- Chassis domain: Active Front Steering (SIL 3)
  - Verifying Performance and Timing for all critical cases
  - Optimizing ECU performance and cost (use of cheaper CPU)
  - Safeguarding against liability claims
- late design phase
- formal analysis used as complement to prototyping to reach higher confidence

Source: BMW
Integration: Tracing + SymTA/S

- Single function execution times

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task Name</th>
<th>Execution Time (ms)</th>
<th>ECET (ms)</th>
<th>WCET (ms)</th>
<th>Preempted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Main, Task 1</td>
<td>0.5</td>
<td>0.8</td>
<td>1.0</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Main, Task 2</td>
<td>0.7</td>
<td>1.1</td>
<td>1.3</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Main, Task 3</td>
<td>0.9</td>
<td>1.2</td>
<td>1.4</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Main, Task 4</td>
<td>1.0</td>
<td>1.3</td>
<td>1.5</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Interrupt Frequency

Tracing plus SymTA/S Analysis

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

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

Risk avoided
Overview

- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion
Enhancements to the Basic Analysis

- shared memory modeling on multi-core systems
  - “secondary“ traffic from shared memory and coprocessor access in conflict with other traffic
  - many schedulability hazards
  - requires model and analysis extension
  - active research area (see literature and tutorial link)

- robustness analysis
  - identification of “system reserves“ for potential changes
  - can be used for optimization (see literature and tutorial link)

- scenario analysis

Scenario Analysis

- Identify different sets of tasks or deviating core execution times of tasks for different application contexts → scenarios

- Example: engine acceleration/idle, ..

- Interesting is transition between scenarios
  - possibly leading to overloads, lost data, ...

- Analysis: analyse scenarios individually + analyse transition

Accelerating on a hill

Holding the speed
Overview

- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion

Conclusion

- several performance analysis and optimization approaches have brought system analysis far beyond the stage of toy examples
- the cost of a predictable design has been reduced by higher modeling and analysis precision
- WCET analysis is part of the analysis chain and currently often “substituted” by extensive simulation
- challenges for WCET analysis arise in multi-core systems, scenario analysis and system optimization
References Symta/S 1/4


References Symta/S 2/4


References MPA 3/4


References other 4/4


