Bitte decken Sie die schraffierte Fläche mit einem Bild ab.

Dagstuhl Seminar 18092
Implementation of Logical Execution Time in an AUTOSAR Based embedded automotive multi-core application

Ralph Mader, 26. February 2018
# LET in an embedded automotive multi-core application

<table>
<thead>
<tr>
<th></th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Powertrain Application</td>
</tr>
<tr>
<td>2</td>
<td>Motivation for LET</td>
</tr>
<tr>
<td>3</td>
<td>Implementation of LET</td>
</tr>
<tr>
<td>4</td>
<td>Outlook</td>
</tr>
</tbody>
</table>
Powertrain Application

Where is Multi-Core in Use?

Combustion Engine

Domain Controller

HV Inverter

DC/DC Converter

Transmission Control

Battery
Powertrain Application
The Problem

A highly cohesive system

How to find parallelization patterns in this kind of application?
LET in an embedded automotive multi-core application

1. Powertrain Application
2. Motivation for LET
3. Implementation of LET
4. Outlook
Motivation for LET
The Problem: Speedup of Partly Parallel Code

How to speedup a calculation sequence while not impacting the functional behavior?
Motivation for LET
LET as Enabler for Parallel Execution
Motivation for LET
LET to Ensure Determinism

Even with changed core allocation:
System behavior defined! “Correct by construction!”
→ Planned determinism!
# LET in an embedded automotive multi-core application

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Powertrain Application</td>
</tr>
<tr>
<td>2</td>
<td>Motivation for LET</td>
</tr>
<tr>
<td>3</td>
<td>Implementation of LET</td>
</tr>
<tr>
<td>4</td>
<td>Outlook</td>
</tr>
</tbody>
</table>
Implementation of LET
Impacts to Architecture

- **OEM-ASW AUTOSAR**
- **Continental-ASW AUTOSAR**
- **IF’s**
- **OEM-ASW Non AUTOSAR**
- **Continental-ASW Non AUTOSAR**
- **Legacy Engine Drivers (CDD)**

**PDA Layer LET Buffer**

**RTE**

**Legacy IF**

- **System Services**
- **Memory Stack**
- **Communication Stack**
- **IO HW abstraction**

**Microcontroller Abstraction - Drivers**

**Multicore Microcontroller**
Implementation of LET
T-Machine on top of AUTOSAR OS

Activate Task

Interrupt Event for T-Machine

Driver Task:
- Task Activations
- Buffer Management
- Synchronization

Period

Core A2

LET 01
LET 02
LET 03

Core A1

TASK A2 LET01
TASK A2 LET02
TASK A2 LET03

Period

TASK A1 LET01
TASK A1 LET02
TASK A1 LET03

LET in Powertrain Application
Dagstuhl 18092
Public

26 February 2018
Ralph Mader, © Continental AG
# Implementation of LET

Example LET Frame Design for demo purpose

<table>
<thead>
<tr>
<th>Proposal 3</th>
<th>System Event</th>
<th>Core</th>
<th>Relative Mode</th>
<th>0 (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LETDRV</td>
<td>A0</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>1msT0_LET</td>
<td>A0</td>
<td>CG2</td>
<td>1ms</td>
<td>01</td>
</tr>
<tr>
<td>5msT1_LET</td>
<td>A0</td>
<td>CG2</td>
<td>5ms</td>
<td>01</td>
</tr>
<tr>
<td>10msT3_LET</td>
<td>A0</td>
<td>CG2</td>
<td>10ms</td>
<td>01</td>
</tr>
<tr>
<td>100msT1_LET</td>
<td>A0</td>
<td>CG2</td>
<td>100ms</td>
<td>01</td>
</tr>
<tr>
<td>1000msT1_LET</td>
<td>A0</td>
<td>CG2</td>
<td>1000ms</td>
<td>02</td>
</tr>
<tr>
<td>SysTTran</td>
<td>A0</td>
<td>CG1</td>
<td>Not TDL</td>
<td></td>
</tr>
<tr>
<td>100msLLC</td>
<td>A0</td>
<td>CG1</td>
<td>Not TDL</td>
<td></td>
</tr>
<tr>
<td>LETDRV</td>
<td>A1</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5msT1_LET</td>
<td>A1</td>
<td>CG2</td>
<td>5ms</td>
<td>01</td>
</tr>
<tr>
<td>10msT3_LET</td>
<td>A1</td>
<td>CG2</td>
<td>10ms</td>
<td>01</td>
</tr>
<tr>
<td>20msT3_LET</td>
<td>A1</td>
<td>CG2</td>
<td>20ms</td>
<td>01</td>
</tr>
<tr>
<td>100msT1_LET</td>
<td>A1</td>
<td>CG2</td>
<td>100ms</td>
<td>01</td>
</tr>
<tr>
<td>1000msT1_LET</td>
<td>A1</td>
<td>CG2</td>
<td>1000ms</td>
<td>01</td>
</tr>
<tr>
<td>SysTTran</td>
<td>A1</td>
<td>CG1</td>
<td>Not TDL</td>
<td></td>
</tr>
<tr>
<td>100msLLC</td>
<td>A1</td>
<td>CG1</td>
<td>Not TDL</td>
<td></td>
</tr>
<tr>
<td>LETDRV</td>
<td>A2</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5msT1_LET</td>
<td>A2</td>
<td>CG2</td>
<td>5ms</td>
<td>01</td>
</tr>
<tr>
<td>10msT3_LET</td>
<td>A2</td>
<td>CG2</td>
<td>10ms</td>
<td>01</td>
</tr>
<tr>
<td>20msT3_LET</td>
<td>A2</td>
<td>CG2</td>
<td>20ms</td>
<td>01</td>
</tr>
<tr>
<td>100msT1_LET</td>
<td>A2</td>
<td>CG2</td>
<td>100ms</td>
<td>01</td>
</tr>
<tr>
<td>1000msT1_LET</td>
<td>A2</td>
<td>CG2</td>
<td>1000ms</td>
<td>01</td>
</tr>
<tr>
<td>SysTTran</td>
<td>A2</td>
<td>CG1</td>
<td>Not TDL</td>
<td></td>
</tr>
<tr>
<td>100msLLC</td>
<td>A2</td>
<td>CG1</td>
<td>Not TDL</td>
<td></td>
</tr>
</tbody>
</table>
Implementation of LET
Scheduler Overhead

Design of LET Frames influences significantly the scheduler overhead!

LET Frame design with gaps:
~ 2.3% Load*

Seamless LET Frame design:
~ 1.3% Load*

*3 Cores = 100%, AURIX 27x @200MHz
LET in an embedded automotive multi-core application

1. Powertrain Application
2. Motivation for LET
3. Implementation of LET
4. Outlook
Outlook

- Standardization in AUTOSAR
- Need for Abstract “Language” to describe integration needs (perhaps Phase Concept [3]). Embedding LET in a system event chain model
- Tool support for: Design of large scale systems and Synthesizing the implementation
- HW support for scheduler to become less dependent on delays caused by high prior interrupts
- Extension to POSIX based Systems
- Extension to GP GPUs or other HW accelerators
Discussion

Questions
References


Thank you for your attention!