Outlook

Context:
New tools and methodologies are needed as multicore ECUs are being introduced in the automotive EE architecture.

Problem:
How to address the scheduling of numerous runnables on a multicore ECUs in the context of the automotive domain?

Method:
Deployment of load balancing algorithms in ECU configuration tools.
The number of ECUs has more than doubled in 10 years

Typical number of ECUs in a car in 2000: 20
Typical number of ECUs in a car in 2010: over 40

Other examples
Between 60 and 80 ECUs in the Audi A8
Over 100 ECUs in some Lexus!

Moving towards multicore architecture

Decreasing the complexity of in-vehicle architecture:
- reduces EE design and verification efforts
- decreases number of network interfaces
- decreases traffic on CAN network
- reduces costs
Moving towards multicore architecture

Other use cases for the automotive domain
- Dealing with resource demanding applications
  - engine control, image processing...
- Improving the safety
  - segregation of multi-source software, ISO26262...
- Dedicated use of core
  - monitoring, event-triggered tasks

General benefits of multi-core
- reduced power consumption
- reduced heat
- reduced EMC

AutoSAR requirements
- Static partitioning
- Static cyclic scheduling using schedule tables
- BSW are all allocated on the same core
Problem

Goal: schedule numerous runnables on a multicore ECU

Two sub-problems
- Partitioning
  - 600 runnables on 2 cores
- Build schedule table
  - 300 runnables in 200 slots

Sub-objectives and criteria
- Avoid load peaks
  - Max
- Balance load over time
  - Standard Deviation

Model

Runnables
- Period
- WCET
- Initial Offset
- Core allocation constraint
- Colocation constraint

Sequencer task

\[ \text{T}_{\text{tic}} \quad \text{Slots} \quad \text{T}_{\text{cycle}} \]
Solution

**Partitioning** is dealt with as a bin packing problem
- **Worst fit decreasing algorithm** with fixed number of bins

**Load Balancing** is done with the Least Loaded algorithm (LL)
- Inspired from CAN domain [Grenier and Navet ERTSS2008]
- Extended to handle non harmonic runnable sets (G-LL)
- Improved so as to reduce further load peaks (G-LLσ)

**Implemented in a tool**
- Freely available soon at http://www.realtimeatwork.com
Harmonic task sets

**LL**
- Max: 4.79
- Min: 4.52
- StdDvt: 0.038

**G-LLσ**
- Max: 4.75
- Min: 4.65
- StdDvt: 0.018

Generated load: 94%, $T_{\text{tic}}=5\text{ms}$, $T_{\text{cycle}} = 1\text{s}$

Non harmonic task sets

Schedulability bound in the harmonic case

<table>
<thead>
<tr>
<th>Max WCET (μs)</th>
<th>150</th>
<th>300</th>
<th>900</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated CPU load</td>
<td>95%</td>
<td>97%</td>
<td>95%</td>
</tr>
<tr>
<td>Schedulability bound in the harmonic case</td>
<td>94% Max WCET = 300μs</td>
<td>82% Max WCET = 900μs</td>
<td></td>
</tr>
<tr>
<td>Success % of LL</td>
<td>96%</td>
<td>96%</td>
<td>92%</td>
</tr>
<tr>
<td>Success % of G-LL</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Success % of G-LLσ</td>
<td>100%</td>
<td>100%</td>
<td>97%</td>
</tr>
</tbody>
</table>

Statistics collected over 1000 generated runnable sets
Multiple synchronized sequencer tasks per core

Incremental scheduling of three synchronized sequencer tasks with respective load of 45%, 35% and 15% resulting in 95% of the core capacity.

\[ T_{cycle} = 1000 \text{ms} \quad \text{and} \quad T_{tic} = 5 \text{ms} \]

Multiple non synchronized sequencer tasks per core

Case arises for sequencer tasks using different tic counters

- Engine control applications (standard time vs RPM)

Any offset between the sequencer tasks and all clock rates are possible during runtime

- each sequencer task needs to be balanced independently

Verification is possible considering maximum clock rates

- Multi-frame scheduling results can be used
Conclusion

Adoption of multicore ECU raises new challenges
- Evolution of software architecture design
- Scheduling of software components

We propose runnable scheduling heuristics for ECUs
- Fast and performant
- Easily adaptable for more advanced applications
- Compatible with AutoSAR R4.0 and its multicore extensions

Future work
- Precedence constraints
- Lockstep synchronization
- Distributed timing chains

Thank you for your attention