Hard Real-time Guarantee of Automotive Applications during Mode Changes

P. Dziurzanski, A.K. Singh, L.S. Indrusiak
University of York

B. Saballus
Robert Bosch GmbH
Overview

- Automotive technology: an overview
- Modes in Engine Control Units (ECU)
- DemoCar use case
- Design flow
- Conclusion
Automotive technology: an overview

Engine control system

- 800 – 1600 SW components
- 1000 – 1500 runnables (basic executable entities) in about 10 tasks
- 3000 – 6000 sub functions
- 7000 – 15000 messages
- hard real-time and safety requirements

Source: Robert Bosch GmbH
### Current vs future trends in cars

<table>
<thead>
<tr>
<th>2015</th>
<th>Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>different HW units for different functionality</td>
<td>multiple applications per HW unit</td>
</tr>
<tr>
<td>more than 100 ECUs in a premium car</td>
<td>dynamic distribution</td>
</tr>
<tr>
<td>static distribution</td>
<td>Network on Chips</td>
</tr>
<tr>
<td>bus-based</td>
<td></td>
</tr>
</tbody>
</table>
Limited number of operating modes

- **Ignition key-On**: engine off state
- **Cranking**: engine starting state
- **Idle**: a throttle valve is not opened
- **Part load**: a throttle valve is partially opened
- **Wide open throttle**: a throttle valve is wide open

DemoCar- Minimal functionality gasoline Engine Control Unit

- 18 runnables
- 6 tasks
- 61 labels (data elements that can be read or written by runnables)
Problem formulation

Goals:
- no deadline violation
- minimal #used resources
- minimal energy dissipation
The current state is stored in a certain variable.
Runnables execution time in different modes

<table>
<thead>
<tr>
<th>Runnable</th>
<th>Min op.</th>
<th>Max op.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CylNumObserver</td>
<td>134</td>
<td>345</td>
</tr>
<tr>
<td>InstructionsDeviation</td>
<td>3728</td>
<td>5921</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Runnable</th>
<th>Min op.</th>
<th>Max op.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CylNumObserver</td>
<td>245</td>
<td>543</td>
</tr>
<tr>
<td>InstructionsDeviation</td>
<td>728</td>
<td>921</td>
</tr>
</tbody>
</table>
Steps of the proposed dynamic resource allocation method

1. **Design-time optimisation**
   - Taskset
   - Platform
   - Modes

2. **Allocations**

3. **Run-time Platform Resource Manager**
   - Current Mode

4. **Allocation Result**
Reasons:
• Runnables have similar runtime and resource consumption in neighbouring modes.
• Transition is required to be done immediately.
Spanning tree construction
Static mapping for initial mode

Using energy dissipation as a criterion, huge penalties for deadline violations
Genes in chromosomes

Runnable mapping (n genes)

Core P-mode (x·y genes)

\[ \tau_1, \tau_2, \tau_3, \ldots, \tau_n \]
Optimization result
Static mapping for non-initial modes minimizing amount of migrated data

- Multicriteria optimization:
  - migration cost from the previous mode
  - energy dissipation

- Huge penalties for deadline violations
Pareto frontier - Cluster_1

Data to be migrated [bytes]

Energy used in mode Cluster_1 [$\mu$J]

No deadline misses.
Idle hardware in states - 2x2 NoC

- 2 idle cores
- 10 idle links

5909.37 μJ per hyperperiod

- 3 idle cores
- 16 idle links

3093.01 μJ per hyperperiod

- 0 idle cores
- 10 idle links

9719.45 μJ per hyperperiod
Deadlines for runnables

Runnable execution

data written to labels
Schedulability analysis for taskset in any mode

Schedulability analysis for all packets that can be transmitted in this time slot

Schedulability analysis for all jobs that can be executed in this time slot
Schedulability analysis for taskset during mode changes

Schedulability analysis for all packets that can be transmitted in this time slot *together with migrations* (done with periodic servers)
Network bandwidth selection

How many hyperperiods are required to migrate all the necessary data using periodic server?

<table>
<thead>
<tr>
<th>Router latency [ns]</th>
<th>Link latency [ns]</th>
<th>#Hyperperiods (100ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>400</td>
<td>1</td>
</tr>
<tr>
<td>200</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td>400</td>
<td>800</td>
<td>2</td>
</tr>
<tr>
<td>500</td>
<td>1000</td>
<td>2</td>
</tr>
</tbody>
</table>

Data to be migrated [bytes]

Energy used in mode Cluster_1 [μJ]

<table>
<thead>
<tr>
<th>Router latency [ns]</th>
<th>Link latency [ns]</th>
<th>#Hyperperiods (100ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>400</td>
<td>2</td>
</tr>
<tr>
<td>200</td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td>400</td>
<td>800</td>
<td>3</td>
</tr>
<tr>
<td>500</td>
<td>1000</td>
<td>3</td>
</tr>
</tbody>
</table>
Conclusion

- Static mapping of modes and dynamic migration at runtime on mode switch has been proposed to decrease the number of cores in ECUs.
- The technique benefits from the modal nature of the ECU.
- GA has been used to determine the allocation for all the ECU's modes, minimizing the energy dissipation and migration.
- NoC bandwidth for full schedulability has been determined.
Thank you!
Estimation of dissipated energy

\[ E_0 = E_{0,\text{init}} + f_{d_0}(t_{0,0}) + f_{d_0}(t_{0,1}) + f_{s_0}(t_s) \]

\[ E_1 = E_{1,\text{init}} + f_{d_1}(t_{1,0}) + f_{s_1}(t_s) \]

Dynamic energy:
\[ f_{d_0}(t) = \text{const}_0 \cdot t \]
\[ f_{d_1}(t) = \text{const}_1 \cdot t \]

Static energy:
\[ f_{s_0}(t) = \text{const}_2 \cdot t \]
\[ f_{s_1}(t) = \text{const}_3 \cdot t \]