Mapping and Scheduling Automotive Applications on ADAS Platforms using Metaheuristics

Shane D. McLean¹, Silviu S. Craciunas², Emil Alexander Juul Hansen¹, Paul Pop¹

¹Technical University of Denmark, Kongens Lyngby, Denmark
²TTTech Computertechnik AG, Vienna, Austria
Autonomous vehicles/Assisted driving

- Safety – less accidents
- Reduced Emissions due to lower fuel consumptions
- Better Health due to less stress through traffic jams
- Increased highway capacity
An autonomous vehicle can be understood as a robot

Robot = Sense = See Sensors + Think = Compute + Act

- Brake Actuators
- Steering Actuators
- Electric Drivetrain
Modern autonomous vehicles

Traditional approach based on distributed ECUs and separated domains, interconnected through different technologies (ETH, CAN, FlexRay)

Source: Ernst et al. - Ethernet as Future Automotive Communication Backbone
Modern autonomous vehicles – issues

• Rapid growth of software functionality and the necessary compute performance cannot be addressed with current electronics architecture and ECUs
• Too many ECU’s with too little processing power and memory
• Limitation of the domain concept (development cost, replication of basic software functions, sources of failure, maintenance cost)
• Fail-operational requirement for level 4 autonomous driving:
  • The domain concept is not sustainable for L4/L5 autonomous driving.
  • Autonomous driving functions require the integration of cross-domain information and functions.

From distributed, separate ECUs ➔ In-car Computing Platform
ICCP - Integrated platform

- from hardware to software
- from distributed to centralized
Processing Resources:
1x Renesas RH850P/1H-C (ASIL D MCU with lockstep cores @ 240MHz)
2x Renesas R-Car H3 (ASIL B SoC with 4x Cortex A57, 4x Cortex A53, 1x Cortex R7, 1x IMP-X5, 1x IMG GX6650 GPU)

Video Interfaces:
12 x camera inputs (GMSL) incl. remote supply (PoC) 2 x display outputs (FPD-Link III)

Communication Interfaces:
4x OABR 100BASE-T1 2 x FlexRay (A/B channel) – wakeup capable
2 x HS-CAN – wakeup capable
4 x CAN-FD 2 x LIN
I/O Interfaces 2 x analog/digital inputs 2 x high side outputs 1 x sensor supply output (5V)
The assignment of tasks to cores is captured by the mapping function (which may be tested independently) can be integrated with other software functions compositionally. The system is composed of several components, each executing different tasks and requiring different resources.

### Task Model

Tasks are modeled as directed acyclic graphs (DAGs) where each vertex represents a task and each edge represents a dependency arising from task chains (defined below). We model the dependencies as a set of tasks that are pre-assigned to cores or must be assigned by the scheduling algorithm. Tasks have real-time requirements, both in terms of period and deadline.

### Abstraction Layer / Middleware

The abstraction layer provides the capability to execute tasks with a combination of deterministic timing and isolation properties. This layer is running on top of the operating system (OS) and is responsible for mapping tasks to cores.

### Operating Systems

Each host can run a different operating system depending on its requirements. For example, the MotionWise [18] layer is running on top of the safety and performance requirements. Each such OS can be integrated with the abstraction layer and middleware.

### Integrated Platform

- **RTOS**: CPU
- **Autosar OS**: CPU
- **Linux / Android**: CPU
- **Safety / Performance**: CPU, GPU

The platform is interconnected through either a deterministic Ethernet backbone (TSN) or through PCIe.

### Different multi-core CPUs:

- process the information arriving from a variety of sensors (radar, ultrasonic sensors, cameras, LIDAR, etc.)
- run control loops
- run other utility functions (lane keeping/changing assistant, emergency braking, logging, etc.)

### Heterogeneous multi-core multi-SoC platform

Heterogeneous multi-core multi-SoC platform featuring a variety of CPUs and GPUs running at different speeds, which are interconnected through either a deterministic Ethernet backbone (TSN) or through PCIe.
Periodic hard real-time tasks with \((\text{WCET}, \text{Period})\) definition
- Are pre-assigned to CPUs (WCET is already scaled to speed)
- Can be pre-assigned to core, if not assigned, assignment will be part of the allocation problem
- Can have deadline, activation, jitter constraints
- Preemption is allowed, migration is not allowed

Result of scheduling is a static table which determines the exact timely behavior of tasks

Different dimensions to the allocation problem:
- Assignment of tasks to cores/CPUs
- Scheduling of tasks
- Real-time requirements are met end-to-end
Real-time requirements – activation, deadline, period

0  period

activation  WCET  deadline
Real-time requirements – jitter

\[ \text{Jitter} = \text{Jitter}^* - \text{WCET} \]
Real-time requirements – dependency chains

Characteristic of automotive software – **Cause-effect chains:**
- provide additional timing and dependency requirements on the execution of tasks
- can span across multiple activation patterns
- include multiple tasks, even the same task multiple times
- have priorities and end-to-end latencies
- include communication latencies
Simulated Annealing (SA)-based metaheuristic approach which uses an EDF-based heuristic to solve the task scheduling problem.

The scheduling heuristic allows task preemption by simulating an Earliest Deadline First (EDF) scheduling policy parameterized by task offsets and local deadlines decided by SA.

Algorithm 1: SimulatedAnnealing($\mathcal{A}, \Gamma, s_0, t_s, cr, i$)

1: $t \leftarrow t_s$
2: $s \leftarrow$ ScheduleSynthesis($\mathcal{A}, \Gamma, s_0$)
3: $s^* \leftarrow s$
4: while $timeleft$ do
5:     while $t > 1.0$ do
6:         for $k \leftarrow 1$ to $i$ do
7:             $s' \leftarrow$ GenerateNeighbor($\mathcal{A}, \Gamma, s$)
8:             if $\text{Cost}(s') < \text{Cost}(s)$ then
9:                 $s \leftarrow s'$
10:            if $\text{Cost}(s') < \text{Cost}(s^*)$ then
11:                $s^* \leftarrow s'$
12:        end if
13:    else if $\exp\left(\frac{\text{Cost}(s) - \text{Cost}(s')}{t}\right) > \text{random}[0, 1]$ then
14:        $s \leftarrow s'$
15:    end if
16:     $t \leftarrow t \cdot (1 - cr)$
17: end for
18: end while
19: end while
20: return $s^*$
EDF simulation

EDF is an optimal online scheduling algorithm which at each time instant prioritizes the task with the earliest deadline.

We can use it to generate a static schedule table – simulate EDF until 2*Hyperperiod + max_offset.

Schedulability test: \( \forall t_1 \in \Phi^\sigma, \forall t_2 \in \Delta^\sigma, t_1 < t_2 : \sum_{\tau_i \in I^\sigma} C_i \times \left( \left\lfloor \frac{t_2 - \phi_i - D_i}{T_i} \right\rfloor - \left\lfloor \frac{t_1 - \phi_i}{T_i} \right\rfloor + 1 \right)_0 \leq t_2 - t_1, \)

where

\( \Phi^\sigma \overset{\text{def}}{=} \{ a_{i,j} = \phi_i + j \times T_i | \tau_i \in \Gamma^\sigma, j \geq 0, a_{i,j} \leq \lambda^\sigma \}, \)

\( \Delta^\sigma \overset{\text{def}}{=} \{ d_{i,j} = a_{i,j} + D_i | \tau_i \in \Gamma^\sigma, j \geq 0, d_{i,j} \leq \lambda^\sigma \}, \)

\( \lambda^\sigma = \max(\{ \phi_i | \tau_i \in \Gamma^\sigma \}) + 2 \times \text{lcm}(\{ T_i | \tau_i \in \Gamma^\sigma \}). \)

Two knobs to play around with: offset and deadline of each task.
Simulated Annealing + EDF simulation

Simple Algorithm:
Generate initial candidate:
- task offsets = 0
- task deadlines = Period
- task to core assignment based on best-fit/first-fit (load balancing)

Simulated Annealing Loop:
- EDF schedulability test/EDF simulator
- Generate new candidate through performing design transformations
  - SwapTask, AdjustOffset and AdjustDeadline
- Evaluate a solution based on the cost metric

\[
\text{Cost}(s) = \begin{cases} 
\sum_{i \in L} \frac{l_i}{\tau_i} \cdot p_i \cdot w_1 & \text{if } \chi(s) = \text{true} \\
 w_1 + \rho_R + \rho_D + \rho_J & \text{if } \chi(s) = \text{false}
\end{cases}
\]
Five test cases, ranging from 100% to 500% in scale, i.e., for ADAS1x100% the application contains 151 tasks and 31 chains using a model of the architecture. A test case is a scenario consisting of 30 synthetically generated task sets, with each undergoing 30 trials (900 trials for each algorithm).

**Evaluation results on synthetic test cases**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min Avg Max</td>
<td>Min Avg Max</td>
<td></td>
<td>Min Avg Max</td>
<td></td>
<td>Min Avg Max</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADAS1x100%</td>
<td>1 hour</td>
<td>0.97 0.98 1.00</td>
<td>0.58 0.61 0.68</td>
<td>1.00</td>
<td>1.00 1.00 1.00</td>
<td>1.00</td>
<td>1.00 1.00 1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADAS1x200%</td>
<td>2 hours</td>
<td>0.97 0.99 1.00</td>
<td>0.55 0.67 0.75</td>
<td>1.00</td>
<td>0.98 1.00 1.00</td>
<td>0.94 1.00 1.00</td>
<td>1.00 1.00 1.00</td>
<td>0.98 1.00 1.00</td>
<td>0.71 0.95 1.00</td>
<td>1.00 1.00 1.00</td>
</tr>
<tr>
<td>ADAS1x300%</td>
<td>3 hours</td>
<td>0.97 0.99 1.00</td>
<td>0.52 0.64 0.72</td>
<td>1.00</td>
<td>0.97 0.99 1.00</td>
<td>0.70 0.87 1.00</td>
<td>1.00 1.00 1.00</td>
<td>0.97 0.99 1.00</td>
<td>0.70 0.88 1.00</td>
<td>1.00 1.00 1.00</td>
</tr>
<tr>
<td>ADAS1x400%</td>
<td>4 hours</td>
<td>0.97 0.97 0.98</td>
<td>0.52 0.64 0.73</td>
<td>1.00</td>
<td>0.97 0.99 1.00</td>
<td>0.69 0.80 0.88</td>
<td>1.00 1.00 1.00</td>
<td>0.94 0.99 1.00</td>
<td>0.70 0.81 0.92</td>
<td>1.00 1.00 1.00</td>
</tr>
<tr>
<td>ADAS1x500%</td>
<td>5 hours</td>
<td>0.97 0.98 0.98</td>
<td>0.51 0.62 0.70</td>
<td>1.00</td>
<td>0.95 0.98 0.99</td>
<td>0.63 0.78 0.86</td>
<td>1.00 1.00 1.00</td>
<td>0.95 0.98 1.00</td>
<td>0.64 0.79 0.87</td>
<td>1.00 1.00 1.00</td>
</tr>
</tbody>
</table>

**Evaluation results on realistic test cases**

<table>
<thead>
<tr>
<th>Test case</th>
<th>Time</th>
<th>Greedy Chains</th>
<th>Jitter</th>
<th>Sched.</th>
<th>SA Chains</th>
<th>Jitter</th>
<th>Sched.</th>
<th>ADAS1 3.20 0.81 0.37 1.00</th>
<th>0.97 0.99 1.00 0.95 0.99 1.00 1.00</th>
<th>0.94 0.99 1.00 0.84 0.99 1.0 1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAS2 6.40 0.65 0.21 1.00</td>
<td>0.97 0.99 1.00 0.95 0.99 1.00 1.00</td>
<td>0.94 0.99 1.00 0.84 0.99 1.0 1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADAS3 13.20 0.48 0.21 1.00</td>
<td>0.84 0.99 1.00 0.74 0.97 1.0 1.00</td>
<td>0.94 0.99 1.00 0.84 0.99 1.0 1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Real-world test-cases with 151 tasks and 31 chains
Vienna, Austria (Headquarters) | USA | Japan | China
---|---|---|---
Phone +43 1 585 34 34-0 office@tttech.com | Phone +1 978 933 7979 usa@tttech.com | Phone +81 52 485 5898 office@tttech.jp | Phone +86 21 501529250 china@tttech.com

www.tttech.com