Functionality assignment to partitioned multi-core architectures

Florin Maticu

Technical University of Denmark (DTU)

DTU Compute
Institut for Matematik og Computer Science

Outline

Motivation

- federated to integrated architectures.
- multi-core ECUs.
- increase complexity of software functionalities.
- safety according to ISO 26262[1].
- schedulability of tasks running of different cores.
- bus bandwidth constraints.

Figure: ECUs interconnected inside a vehicle [2]

[1] http://www.iso.org/iso/catalogue_detail?csnumber=43464

[2] http://www.embedded.com/print/4011425

Mapping of functionalities

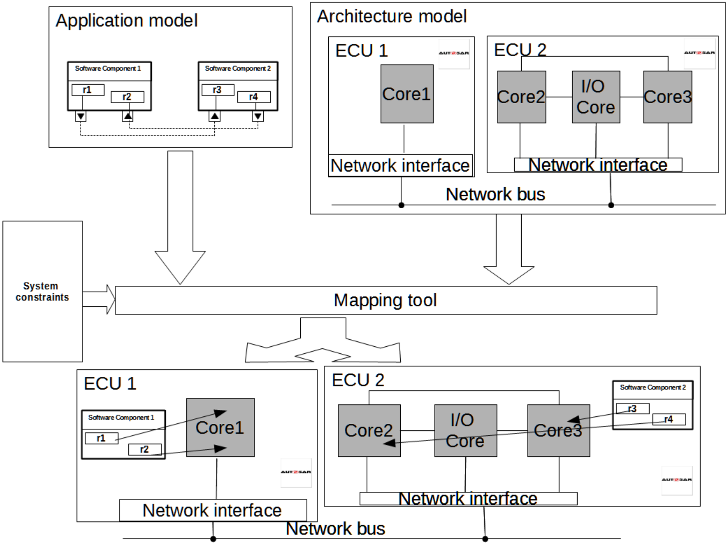

Figure: Mapping tool

Hardware

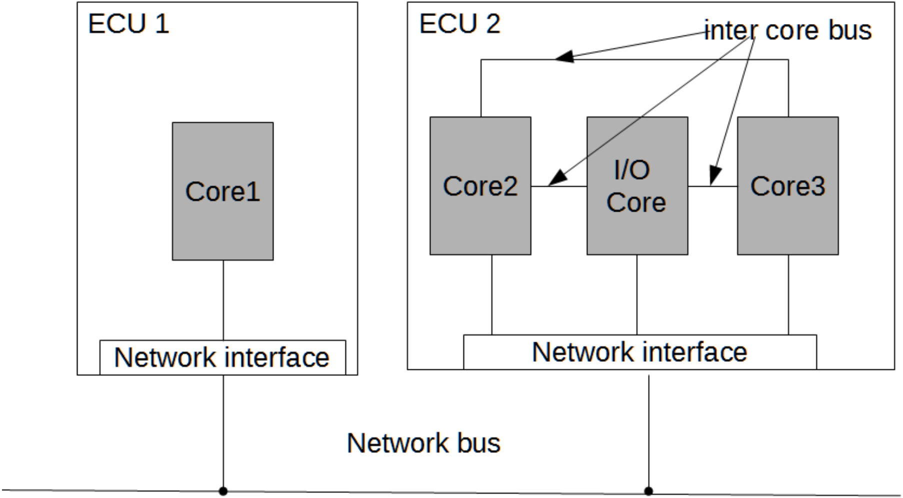

Figure: Hardware architecture model

AUTOSAR

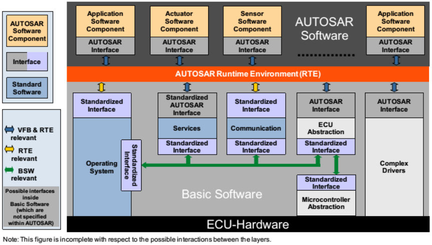

Figure: AUTOSAR layers, [AUT14]

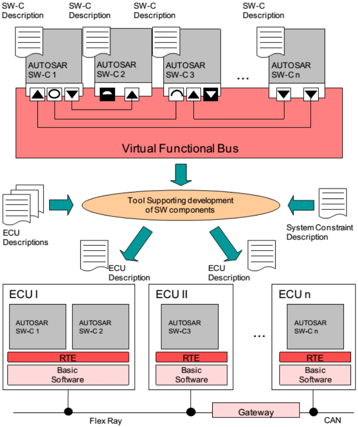

Figure: “Conf System” activity in AUTOSAR, [AUT14]

Communication

- sender-receiver mode with last is best semantics.

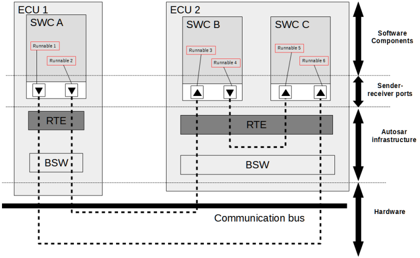

Figure: Runnable communication over AUTOSAR RTE

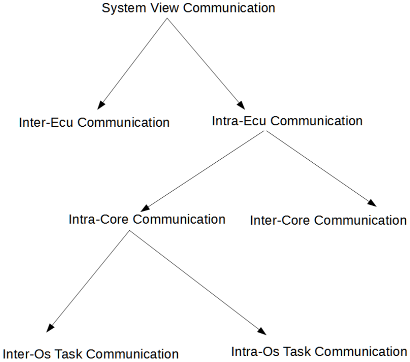

Figure: Types of communication

Os-tasks

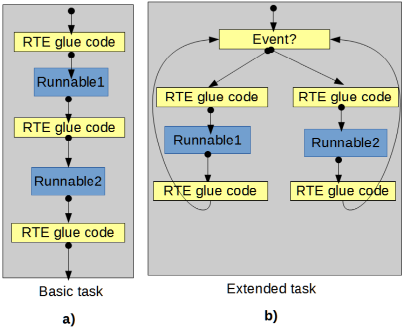

Figure: Runnables with implicit sender/receiver mapped into same Os-task

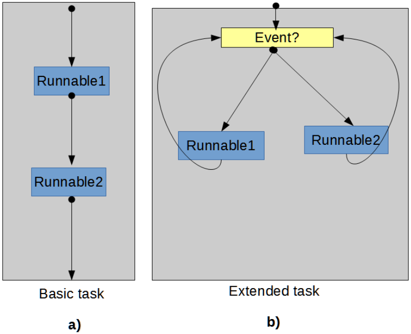

Figure: Runnables with explicit sender/receiver mapped into same Os-task

Scheduling

AUTOSAR Multicore scheduling

OS-tasks are scheduled independently on each core.

$$U_{core} = \sum_{i=1}^{m} \frac{R_i.WCET}{R_i.T} \quad (1)$$

$$U_{core} \leq 0.69, \text{[LL73]} \quad (2)$$

Spatial partitioning

- Spatial protection at the *Os-application* level.

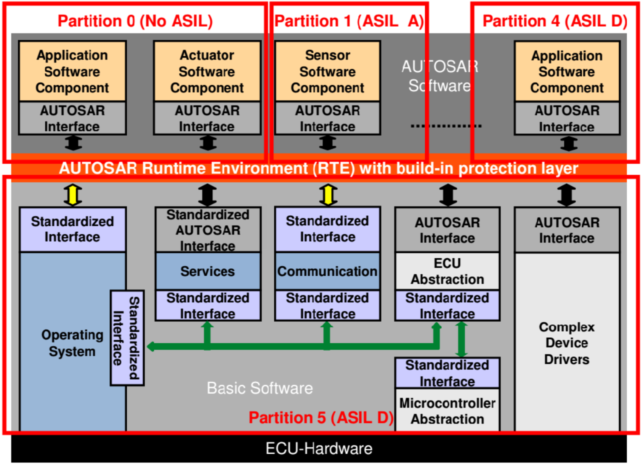

Figure: Memory partitioning example in AUTOSAR, source:[BFWS10]

Temporal partitioning

- Protection against timing faults at the *Os-Task* level.
  - *Execution time budget*.
  - *Resource lock time budget*.
  - *Inter-arrival time budget (Time Frame)*.

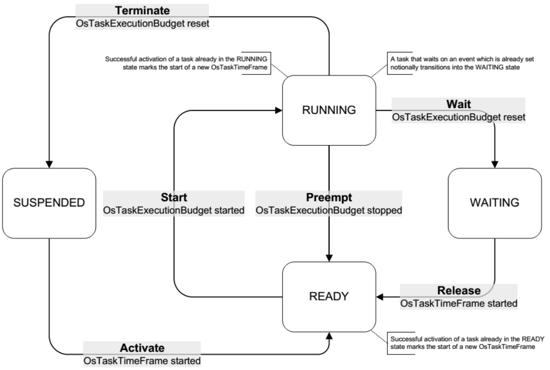

Figure: AUTOSAR OS timing monitoring, source:[AUT14]

End-to-end protection

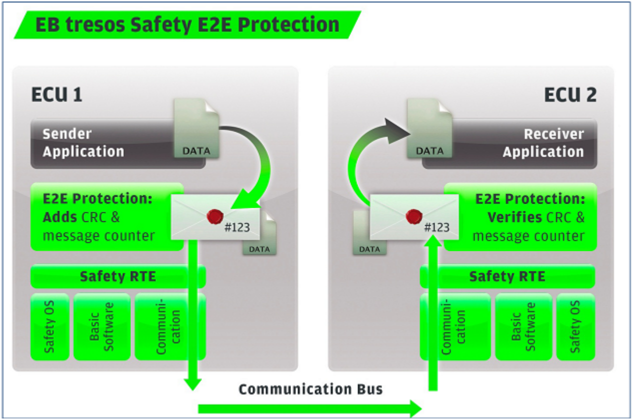

Figure: End-to-end protection example in AUTOSAR EB tresos product, source: [Mat14]

Example

- AUTOSAR application composed of a set of *software components*.
- Each *software component* contains a number of *runnables* (functions).

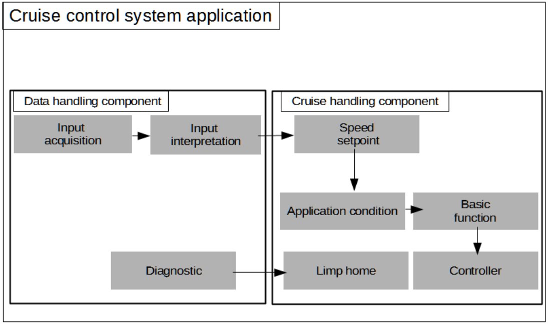

Figure: Control cruise application

WCET of the runnable

- $WCET_{runnable} = WCET_{computational} + WCET_{communication}$
- $WCET_{communication}$ overhead :
  - $\alpha =$ if *runnables* are mapped into the same *OS-Task*.
  - $\beta_0 =$ if *runnables* have the same ASIL levels and are mapped into different *OS-Tasks*.
  - $\beta_1 =$ if *runnables* have different ASIL levels and are mapped into different *OS-Tasks*.
  - $\gamma =$ if the *runnables* are mapped into *OS-Tasks* on different cores on the same ECU.
  - $\theta =$ if the *runnables* are mapped into *OS-Tasks* on different ECUs.
- $\theta > \gamma > \beta_1 > \beta_0 > \alpha$

Input

- **Given**
  - **Architecture model**
    - Each ECU is running on AUTOSAR framework.

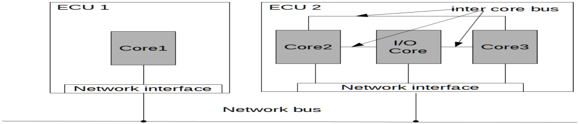

Figure: Architecture model example

  - **Application model**

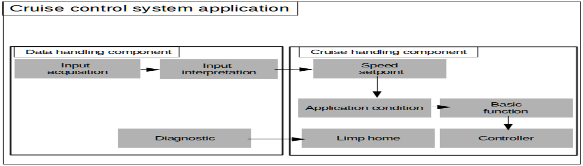

Figure: Application model example

Ouptut

- **Determine**
  - A mapping of *software components* to ECUs.
  - A mapping of *runnables* to *OS-tasks*.
  - A mapping of *OS-tasks* to cores.
  - A mapping of *OS-tasks* to *OS-applications*.

Objectives

- **Such that**
  - Minimize the overall communication bandwidths.
  - Minimize the variance of the core utilizations on the system.
  - Functions with different safety integrity levels are spatial and temporal isolated.
  - All the constraints regarding schedulability or provided by the software/system developer are met.

Problem formulation
Example

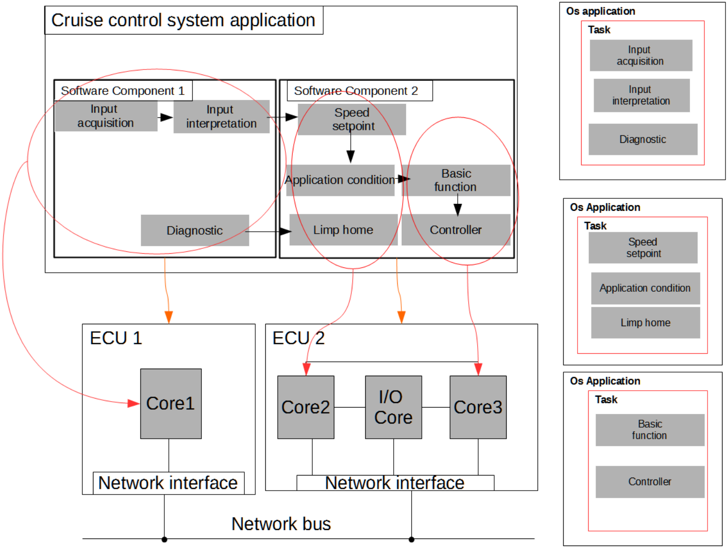

Figure: Mapping solution to ECUs

Cost function

$$cost\ \ function = W_1 \times (\sigma)$$
$$+ W_2 \times \left(\sum_{comm \in \{\cup\{ECU.comm\} \cup \{\cup\{ECU.\{Core.comm\}\}\}} comm \right)$$
$$+ penalty\ \ factor \times \left(\sum_{core \in \{\cup\{ECU.\{core\}\}\}} \max\left(0, U_{core} - U_{core\ \ max}\right) \right)$$
$$+ penalty\ \ factor \times \left(\sum_{comm \in \{\cup\{ECU.comm\} \cup \{\cup\{ECU.\{Core.comm\}\}\}} \right)$$
$$max\left(0, U_{comm} - 1\right).$$

$$\sigma = \frac{1}{N-1} \times \left(\sum_{core \in \{\cup\{ECU.\{core\}\}\}} \left(U_{core} - \mu\right)^2 \right)$$

$$\mu = \frac{1}{N} \times \left(\sum_{core \in \{\cup\{ECU.\{core\}\}\}} U_{core} \right)$$

Simulated annealing

- Heuristic search method for combinatorial problems.
- Finds a solution closed to the optimal one.
- Occasionally allows jumps from a current solution to an inferior one to avoid getting stuck in a local minimum.

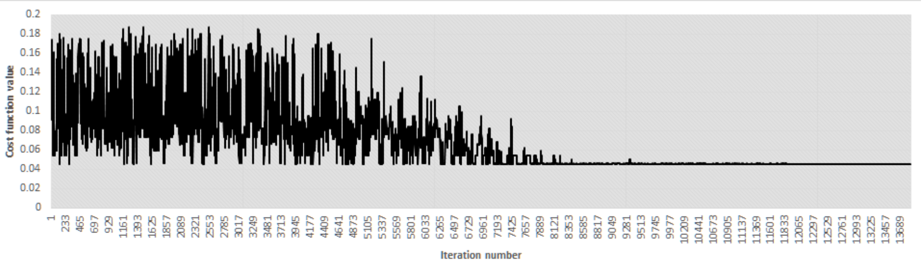

Figure: Cost function values

Algorithm

Input:
application model, architecture model, system mapping constraints
current temperature,minimum temperature,max steps per temperature
Output:
A mapping of software components to ECUs. A mapping of runnables to OS tasks.
A mapping of OS tasks to cores. A mapping of OS tasks to OS applications.

```
foreach software component in the application model do
    randomly assigned it to an ECU
    foreach runnable in the sofware component do
        randomly assigned it to an Core on the ECU
    end
end
Compute current cost;
while current temperature > minimum temperature do
    for step := 1..max steps per temperature do
        Randomly choose a transformation strategy;
        Generate new solution by applying the transformation to the current solution;
        Compute new cost;
        if new cost < curent cost then
            current solution = new solution
        else
            Choose a random number r ∈ [0, 1);
            if e^((old cost − new cost)/current temperature) > r then
                current solution = new solution;
            else
            end
        end
    end
    current temperature = current temperature * cooling factor
end
```

Transform strategies

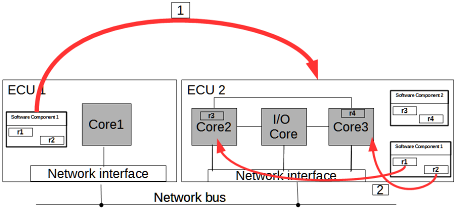

Figure: Move software component transformation

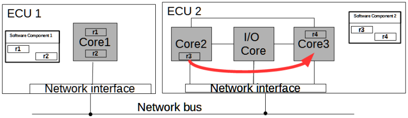

Figure: Move runnables between cores

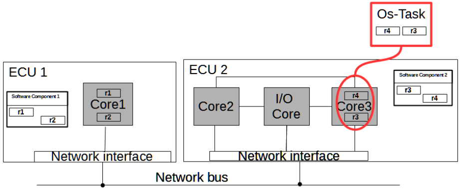

Figure: Move runnables into same Os-Task

Volvo use case

- 50 *software components* with 75 *runnables* in total.

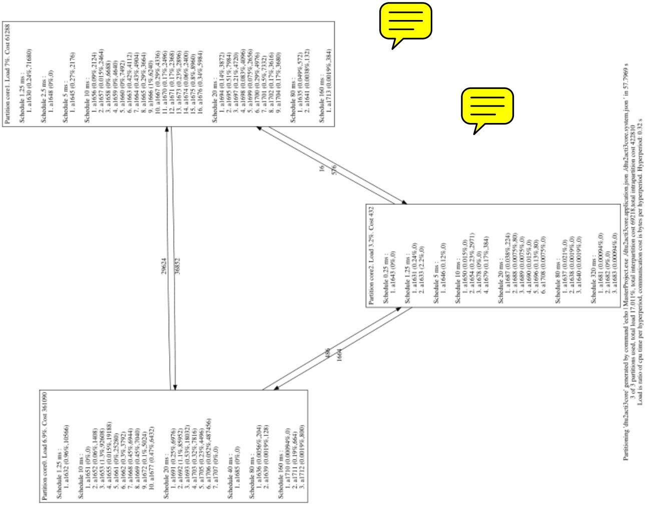

Figure: Volvo mapping result

Conclusions

- Method and a tool has been proposed for the problem of mapping AUTOSAR functionalities (runnables) with different ASIL levels on a distributed network of multi-core ECUs.
- Simulated annealing has been chosen together with a cost function for the mapping of functionalities.
- Three use cases, each composed of an application and architecture model were implemented and tested.
- The tool has been also tested by Volvo Advanced Technology & Research in Götheborg.

Future work

- Implementing new rules such that the tool provides a mapping solution were all the end-to-end timing constraints are met will be an important addition to the current implementation.
- The authors in [LLP^{+}09] have proposed new rules for mapping *runnables* to *Os-tasks* in AUTOSAR such that to minimize the intra-ECU communication. The tool can be improved by adding them and check if we can obtain better mapping solutions given the constraints.

Thank you for your attention!

References

AUTOSAR_SW_OS.
Specification of operating system.
Technical report, AUTOSAR 4.2.1, 2014.

S. Bunzel, S. Furst, J. Wagenhuber, and F. Stappert.
Safety and security related features in autosar, 2010.
http://www.automotive2010.de/programm/contentdata/Bunzel-AUTOSAR.pdf.

C. L. Liu and James W. Layland.
Scheduling algorithms for multiprogramming in a hard-real-time environment.
J. ACM, 20(1):46–61, January 1973.

Rongshen Long, Hong Li, Wei Peng, Yi Zhang, and Minde Zhao.
An approach to optimize intra-ecu communication based on mapping of autosar runnable entities.
In *Embedded Software and Systems, 2009. ICESS '09. International Conference on*, pages 138–143, May 2009.