# AUTOMATIC FUNCTIONALITY ASSIGNMENT TO AUTOSAR MULTICORE DISTRIBUTED ARCHITECTURES

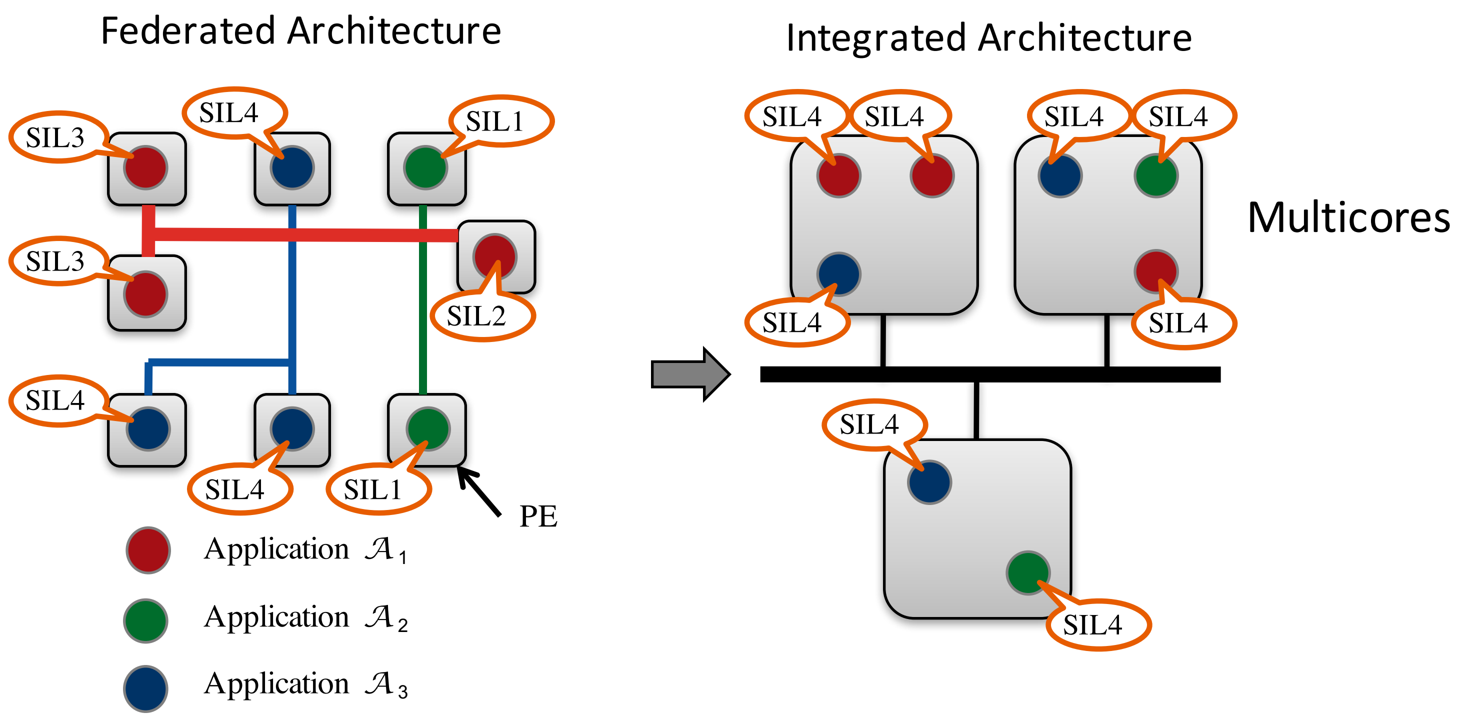
Florin Maticu, Paul Pop
Technical University of Denmark (DTU)
Axbrink Christian, Islam Mafijul
Volvo Group Trucks Technology, Sweden

Presenter: Sune Mølgaard Laursen, DTU

# Motivation

**From federated to integrated architectures, using multicores**

- Multicores have many advantages: SWaP
- Complexity of functionality is increasing
- Stringent timing and safety requirements (ISO 26262)



# Problem and proposed solution

**Business needs for the the next generation vehicles**

- Efficient utilization of multicores
- Compliance with functional safety standard ISO 26262

**Challenges**

- Large number of functions
- Distributed multicore architectures,
  resulting in a large total number of processing cores

**Problem: how to assign the functions to the cores**

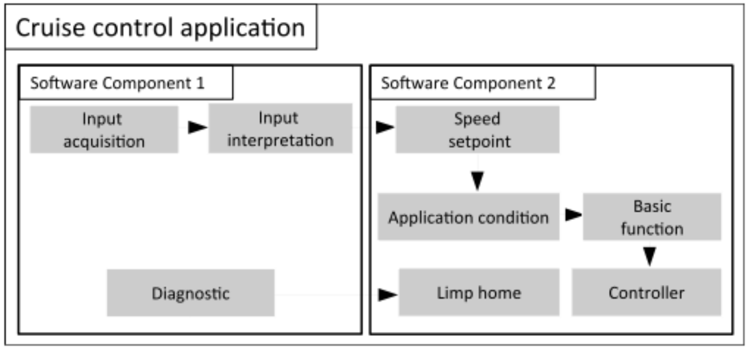
**Solution: automatic mapping tool**

- Reduce the costs (by using multicores, reducing ECUs)
- Maximize performance and resource utilization
- Handle the increased software complexity

## Application model

### Set of automotive applications

- Each *Application* is a set of *Software Components*
- Each Software Component is composed of a set of *Runnables*
  - We know for each runnable
    - ASIL (Automotive SIL) according to ISO 26262
    - Worst-Case Execution Time (WCET)
    - Period and Deadline
  - Runnables are communicating via *Signals*




| $Runnable_i$ | $WCET_i$ (ms) | $T_i$ (ms) | $D_i$ (ms) | $ASIL_i$ |
|---|---|---|---|---|
| Input acquisition | 0.5 | 10 | 10 | A |
| Input interpretation | 1 | 10 | 10 | A |
| Diagnostic | 1.5 | 10 | 10 | A |
| Speed Setpoint | 1 | 10 | 10 | QM |
| Limp home | 1.5 | 10 | 10 | QM |
| Basic function | 2.5 | 10 | 10 | QM |
| Controller | 3 | 10 | 10 | QM |

**AUTOSAR (AUTomotive Open System ARchitecture)**

- Standardized model of development
- Possible for software developers to create reusable software components that are hardware independent

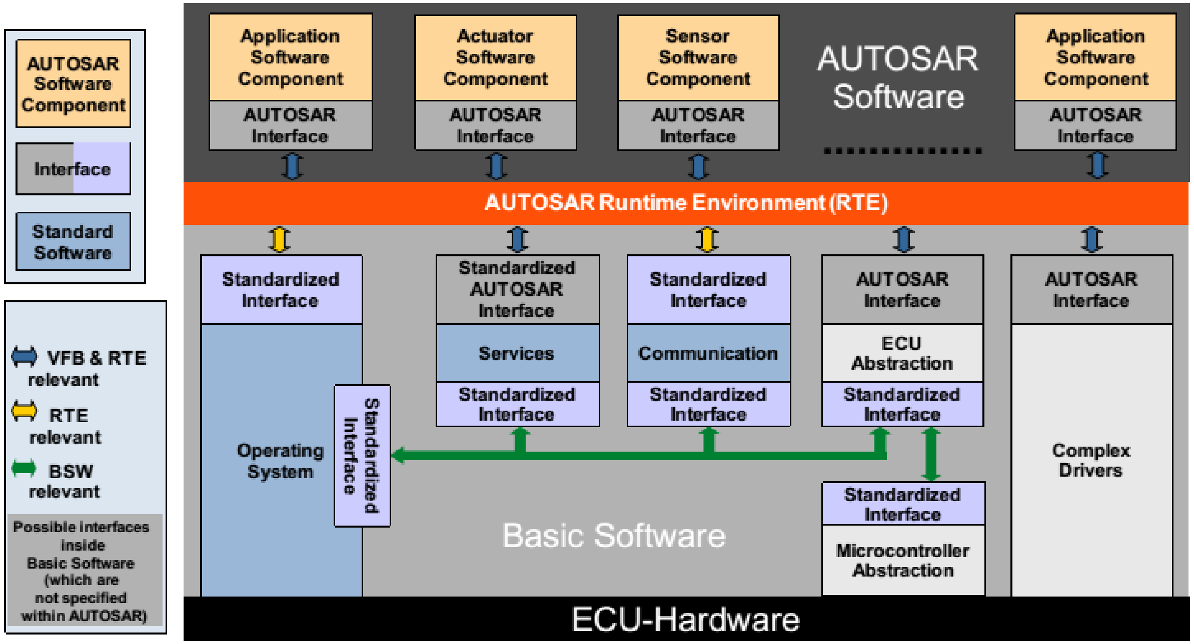
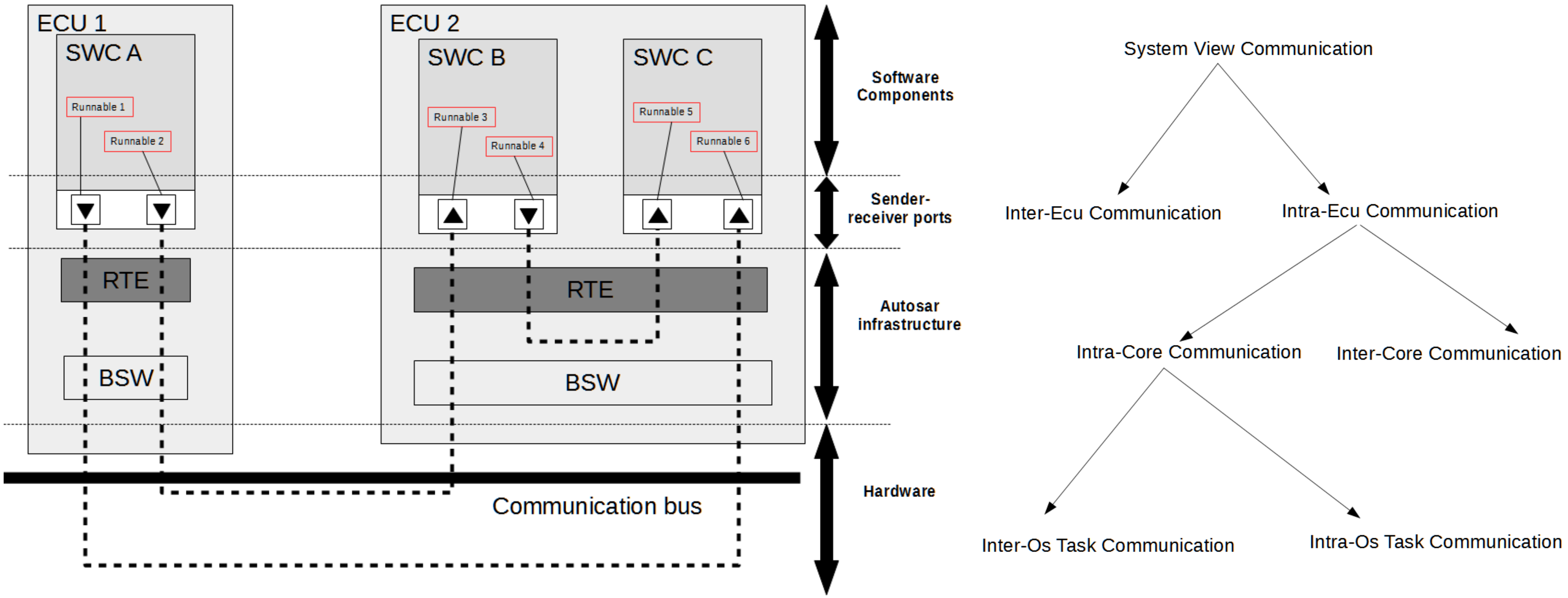



Figure source: AUTOSAR SW OS. Specification of Operating System. Tech. rep. AUTOSAR 4.2.1, 2014.

## Distributed architecture, using AUTOSAR

- Multicore ECUs interconnected using CAN (more protocols can be modeled)
- AUTOSAR software architecture
  - Detailed communication model, takes into account the type of comm.



# Software architecture model

**Scheduling policy**

- Fixed-priority preemptive scheduling, e.g., Rate Monotonic

**A software implementation consists of**

- A set of *OS-Applications*
  - The separation required for safety is ensured through OS-Applications

- Each OS-Applications consists of a set of *OS-Tasks*

- Each OS-Task is composed of a set of Runnables
  - An OS-Task is characterized by
    - WCET
    - Period and Deadline
    - ASIL

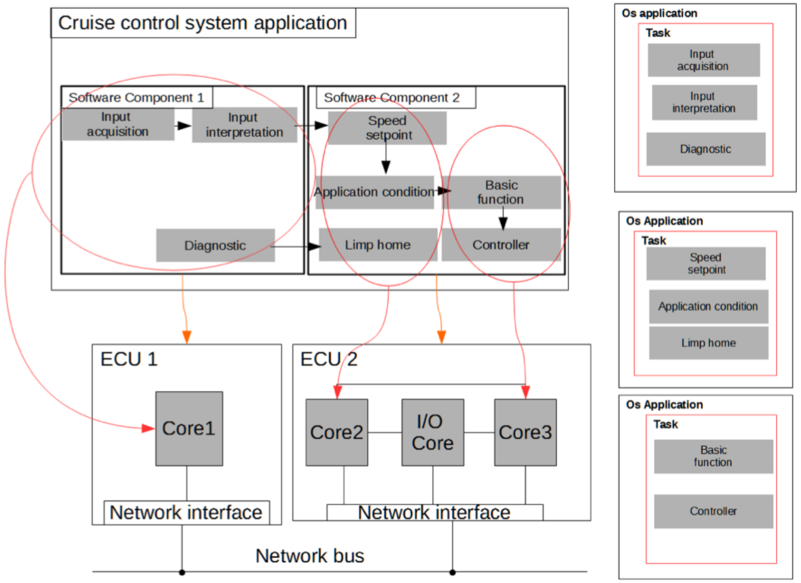
# Problem formulation

**Given**

- Application model and architecture model

**Determine the following mappings:**

- Software components to ECUs
- Runnables to cores
- Runnables to OS-Tasks
- OS-Task to OS-Applications




**Such that we minimize**

- The overall communication bandwidth
- The variance of core utilization of the system (balanced utilization)
- Under the following constraints:
  - Mapping constraints
  - Runnables are schedulable
  - Runnables with different safety integrity levels are spatially and temporally isolated

# Optimization strategy: Simulated Annealing meta-heuristic

**Problem: NP-Hard**

**Optimization strategy: Simulated Annealing**

- Meta-heuristic search method for combinatorial problems
- Uses *design transformations* to randomly explore the search space
- Minimizes an *Cost Function*
- Occasionally allows jumps from a current solution to an inferior one to avoid getting stuck in a local minimum

**Cost Function** $$cost = W_1 \times \sigma + W_2 \times U_b + P_1 \times \alpha + P_2 \times \beta$$

$\sigma$ the total variance in core utilization

$U_b$ the aggregated bus utilization

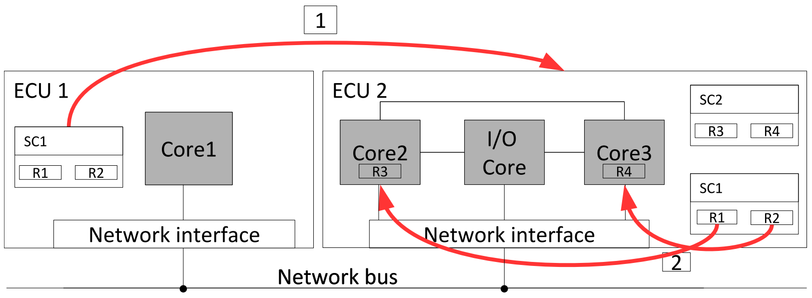
$\alpha$ the amount of cores which utilization has been exceeded

$\beta$ the amount of busses which utilization has been exceeded

*W* and *P* are weights and penalty values

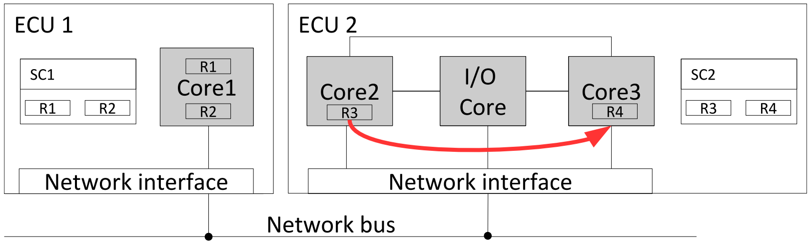
# Simulated Annealing: Design Transformations

**(a)** Randomly choose a software component and map it to a new, randomly chosen, ECU. Then Randomly map the runnables inside the software component to the cores of the new ECU.



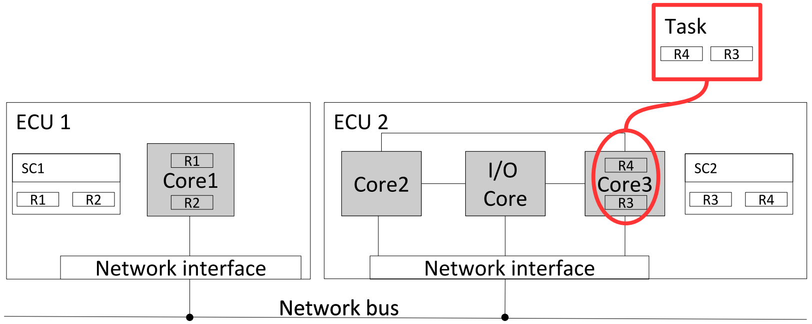

(a) Move a software component

**(b)** Randomly choose a runnable and map it to a new, randomly selected, core within the same ECU.



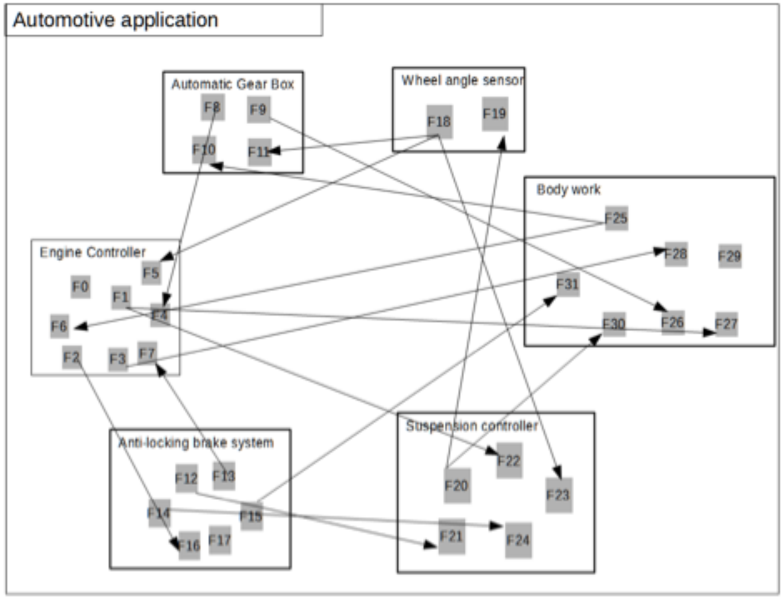

(b) Move runnables between cores

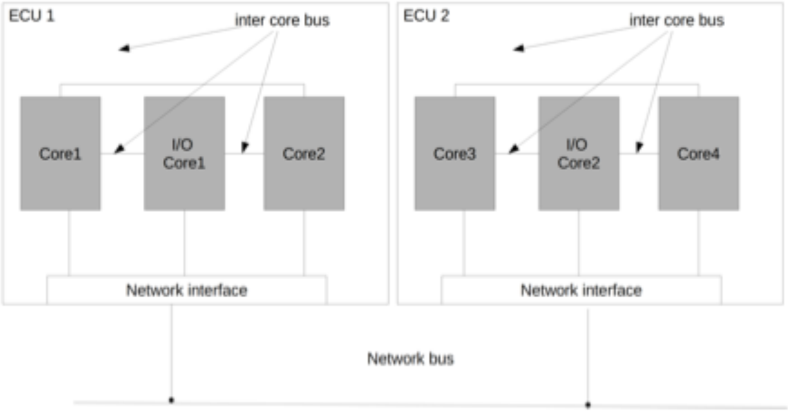
**(c)** Randomly choose two runnables of the same ASIL level assigned to the same core and group them together into an OS-Task.




(c) Move runnables into the same Task

# Example input model (left) and solution (right)




| Software component | ECU |
|---|---|
| Automatic Gear Box ID | ECU1 |
| Suspension controller ID | ECU1 |
| Body work ID | ECU1 |
| Engine Controller | ECU2 |
| Anti-locking brake | ECU2 |
| Wheel angle sensor | ECU2 |

| Runnables | ECU;Core |
|---|---|
| F10, F21, F23, F24, F25, F28, F29 | ECU1;Core1 |
| F11, F22, F27, F31 | ECU1;Core2 |
| F8, F9, F20, F26, F30 | ECU1;I/O Core1 |
| F10, F21, F23, F24, F25, F28, F29 | ECU2;Core3 |
| F3, F4, F6, F7, F12, F13, F17 | ECU2;Core4 |
| F5, F14, F15, F18 | ECU2;I/O Core2 |

# Experimental evaluation

## Experimental setup

- Test Cases

| ID | Name | Software components | Runnables | Signals |
|---|---|---|---|---|
| CS1 | Cruise control | 2 | 8 | 6 |
| CS2 | PSA case study | 6 | 31 | 17 |
| CS3 | Volvo case study | 50 | 75 | 300 |

- Architectures

| ID | ECUs | Cores | ECU bandwidth (bytes/s) | Core bandwidth (bytes/s) |
|---|---|---|---|---|
| Arch1 | 2 | 4 | 50,000 | 10,000 |
| Arch2 | 2 | 6 | 500,000 | 100,000 |
| Arch3 | 1 | 3 | N/A | 500,000 |

# Experimental evaluation, continued

## Results

| Case study | Arch. | Tasks | OS-Apps. | Sched. | Runtime |
|---|---|---|---|---|---|
| CS1 | Arch1 | 7 | 4 | yes | 0.5 sec. |
| CS2 | Arch2 | 19 | 16 | yes | 8 sec. |
| CS3 | Arch3 | 33 | 3 | yes | 45 sec. |

## Additional results

- Volvo use case
  - 50 Software Components with 75 runnables in total.
  - One ECU with 3 cores
  - Output within 2 minutes

# Contributions and Message

**Contributions:**

- Automatic mapping tool for automotive functionality
  - Handles multicores, AUTOSAR
  - Considers ISO 26262
- Utilization based schedulability test
  - Takes into account the AUTOSAR communication type

**Message:**

- Our proposed SA-based optimization approach is able to find, in a short time, schedulable implementations