From analysis to optimization in the deployment of real-time distributed functions in modern automotive systems.

Marco Di Natale

Research work performed with Abhijit Davare, Paolo Giusto, Claudio Pinello, Alberto Sangiovanni-Vincentelli, Wei Zheng, Qi Zhu,

Scuola Superiore S. Anna,
GM Research and Development
University of California at Berkeley,
• Automotive architecture trends and challenges
• Architecture evaluation and selection: from analysis to synthesis
• AUTOSAR and \textit{synthesis of task models}
• Activation models and end-to-end latencies
• Problem definition: feasibility region
• MILP Optimization
• Linearization of worst-case response time and exact analysis
• Metrics, Statistical and Stochastic analysis
• Case Studies
• Way to go …
• **Maybe not your Synthesis**
  • A lot of attention has been (rightfully) dedicated to the synthesis of SW implementation (code) starting from models
  • This talk is not about the synthesis of code, but the synthesis of a task model (against real-time constraints) starting from a functional model (possibly SR/Simulink)
    – Next level up?
  • The problem is quite complex and has several dimension
  • Not suprisingly, several (semantics preservation) constraints apply
  • *Working on an architecture without assumptions on time synchronization (TTA or LTTA, time-based scheduling or priority-based scheduling)*
Functions: Active and Passive Safety

- Passive safety: reduced personal injury in event of an accident
- Active safety: avoiding an accident

Key components:
- ABS: Antilock brake system
- ACC: Adaptive cruise control
- BAS: Brake assist system
- B&W: Brake by wire
- CA: Collision avoidance
- DBW: Drive by wire
- EBD: Electronic brakeforce distribution
- EMB: Electromechanical brakes
- EMS: Electromechanical steering
- ETP: Electronic stability program
- ETC: Electronic traction control
- SbW: Steer by wire (with mechanical backup)

Timeline:
- 1980: Safety cell
- 1990: BAS, ABS, ETC
- 2000: EBD, ACC (Distronic)
- 2005: Smart adaptive controls
- 2020: Autonomous driving
Distributed Systems of Integrated Functions

Complex functional graphs

Mapped into complex distributed architectures
AUTOSAR and Architecture mapping

Virtual Functional Bus

ECU Descriptions

Deployment tools

System Constraint Description

Gateway

runnable
AUTOSAR 4.0 and the “Large scale” problem

Virtual Functional Bus

ECU1

AUTOSAR

RTE

Basic Software

ECU2

AUTOSAR

RTE

Basic Software

AUTOSAR SW-C 1

SW-C Description

AUTOSAR SW-C 2

SW-C Description

AUTOSAR SW-C 3

SW-C Description

AUTOSAR SW-C n

SW-C Description

1A 1B 1C

periodic

2A 2B

periodic

3A 3B

periodic

Sampling delay

periodic

Interference

CAN msg

BSW RTE 1B

periodic

BSW RTE 2A 2B

CAN msg
Models and deadlines

Designers are typically faced with this problem

Given a model of the functions that I would like for my system, with their time constraints ($T_i$ and $D_i$) …

… and a SW architecture, with the task and message design and priorities to optimize functional (extensibility, performance, cost …) is optimized

Define an execution architecture (possibly composing platform subsystems) … ($C_i$)

A design optimization (or synthesis) problem, rather than an analysis problem
Opportunities for synthesis

Perhaps the most important of all

System Functionality

System Architecture

Mapping

Performance Analysis

Refinement

Flow To Implementation

Periods
Delays
Activation modes

Task and message priorities

Number and type of ECU and buses
System topology

Function to task mapping
Task to ECU allocation
Signal to message mapping
X (discrete) space of design optimization variables, such as computation times, placement, priorities, periods …

**Constraints**

- Schedulability
- Communication
- Model Semantics
- Preservation
- Extensibility
Design Optimization: a simplistic view

- X (discrete) space of design optimization variables, such as computation times, periods …

**Constraints**
- Schedulability
- Communication
- Model Semantics preservation
- Extensibility

**Metrics**
- Control related
Objective

Minimization of (average case) end-to-end latencies

Subject to

- Constraints on end-to-end latencies
- Constraints on messages size
- Constraints on utilization
- Constraints on message and task deadlines
- Semantics preservation constraints

Design objectives (optimization variables)

- Placement of tasks onto the CPUs
- Packing of signals to messages
- Assignment of priorities to tasks and messages
- Definition of activation modes/synchronization model
- Period optimization
We would like to use Mathematical Programming (Convex Optimization, MILP, MIGP …)
- Simplicity
  - Problem represented with:
    - Set of decision variables
    - Constraints
    - Objective function
  - “automatically” handles cross dependency among selection choices
- Possible coding of multi-objective optimization
- Standardized approach
  - Well established technique
  - Sound theory, methods
  - Availability of commercial solvers (in essence, search engines)
- How good is your solution?
  - Provides safe estimate of optimal solution
  - Provides intermediate solutions of increasing quality

Challenge:
- Capture the problem and obtain efficient runtimes
Worst case analysis (Schedulability)

Interference $I$ from higher priority tasks on the same resource

Response Time ($r_i$)

Period ($t_i$)

Interference $I$ from higher priority msgs on the same resource

Blocking

Period ($t_i$)

Response Time ($r_i$)

Interference $I$ from higher priority msgs on the same resource
Worst Case Response Times

Not linear (or convex), not even for the single-processor case

Tasks: $$r_i = c_i + \sum_{j \in h(p(i))} \left\lfloor \frac{r_i}{t_j} \right\rfloor c_j \quad \forall o_i \in T$$

Messages: $$r_i = c_i + b_i + \sum_{j \in h(p(i))} \left\lfloor \frac{r_i - c_i}{t_j} \right\rfloor c_j \quad \forall o_i \in M$$

- Resource utilization
  - Fraction of time the resource (ECU or bus) spends processing its objects (tasks or messages)
- Utilization bounds less than x (with x<100%)
  - To allow for future extensibility

$$\left( \sum_{i : o_i \rightarrow R_j} \frac{c_i}{t_i} \right) \leq u_j \quad \forall R_j \in R$$
Periodic Activation Model

High latency, but allows decoupling the scheduling problem

End-to-end latency analysis

Periodic asynchronous activation model

\[
l_{(i, j)} = \sum_{k: \sigma_k \in P(i, j)} (T_k + r_k) \quad \text{where \ (approx.)}
\]

\[
r_i = C_i + \sum_{j \in hp(i)} \left[ \frac{r_i}{T_j} \right] C_j
\]
Event-based Activation Model

Lower latency for high priority paths, jitter increases along the path

End-to-end latency analysis

Data-driven precedence constrained activation model

\[ l_{(i,j)} = \sum_{k: o_k \in P(i,j)} w_k \]  

where (approx.)

\[ w_i = C_i + \sum_{j \in hP(i)} \left[ \frac{w_i + J_j}{T_j} \right] C_j \]
Challenges

• Linearize (or provide an efficient approximate formulation for the response time) – *partly solved*

• What are the optimization metrics?
  – Industry will give us many terms but no formal definition, and some definitions could be quite hard to achieve
    • Extensibility
    • Controls performance
    • Cost

• Design variables are very hard to tackle in a joint optimization process
  – Use stages?

• Time constraints are hardly the only constraints
  – Semantics preservation constraints
  – Reliability constraints
  – Other resources (memory, power, cost …)
Most blocks are of type feedthrough (output depends on input)

This implies a precedence constraint in the computation of the block output functions

Dependencies among outputs

Some blocks have no state
Simulink models

The result is a network of functions (output/state update) with a set of partial orders.

Each blockset is characterized by an execution rate.
• Signals are persistent

• The block semantics defines a partial order of execution
• Single task running at the base period
Runtime execution

• Single task running at the base period

Deadline violation
Runtime execution

Multiple tasks, each executing all blocks at each period

**Problem 1:** data consistency (and determinism in the communication)
Multiple tasks, each executing all blocks at each period

Problem 2: eventually D may again violate the deadline (the priority is non-RM)
Solution to 2: add a delay, now the execution order of C and D can be reversed (RM)
RT blocks: Low rate/priority to high rate/priority

COST
space: 2 additional set of variables for each link
time: overhead of RT implement.
performance: 1-unit delay (low rate period)

Low rate/priority → Protected RT → High rate/priority

COST space: 2 additional set of variables for each link
time: overhead of RT implement.
performance: 1-unit delay (low rate period)

Consistency here is guaranteed by proving there is no preemption

RT-equivalent
However...

- Delay units have a cost (in terms of memory and performance)
- It is possible to selectively preserve the precedence order (giving higher priority to the slow block) at the expense of schedulability
  - Two tasks at the same rate, one high priority, the other low priority
Q: what is the best runnable-to-task mapping?

**Pro:** No need to protect communication between E and F.

**Cons:** Less scheduling flexibility, limited priority inversion
Distributed implementation of models

A very simple model with oversampling ....
Imagine the data streams between source blocks and the multiplier/comparator are exchanged over a network. These are the results seen by the control engineer at design time.
Delays from network

An example of the trade-offs between additional functional delays and scheduling feasibility

Block A
period = 4

Block B
period = 4

Block C
period = 1

FlexRay network

A semantics preserving implementation may be difficult to schedule because of a tight communication pattern
Delays from network

Designers may be tempted to ease the scheduling problem by choosing the instance of the receiving task/block.
Delays from network

Unfortunately, by doing so, the behavior is different from the one simulated with 0-delay. Are the designers/developers fully aware of these issues? How can we help them?

(Task and message design and scheduling are in the background)
Conclusions

• Schedulability theory and worst-case timing analysis ...
  – From the run-time domain to the design domain (already happening)
  – From the analysis domain to optimization (synthesis) domain
  – Complemented by sensitivity analysis and uncertainty evaluation
  – Need efficient ways to linearize response time computation
  – And possibly to partition end-to-end deadlines
  – Need theory to formally evaluate the performance cost of adding delay units
  – Or, in general, to relate control performance to control function periods and to end-to-end latencies
  – Need to understand how to tackle large problems with multiple design variables
Thank you!