This is the author's version of the work. The definitive work was published in *Proceedings of Design Automation Conference (DAC 2009), pp. 43-46, 2009.* The work is supported in part by the German Science Foundation (DFG), SFB 694.

# Designing Heterogeneous ECU Networks via Compact Architecture Encoding and Hybrid Timing Analysis*

Michael Glaß†, Martin Lukasiewycz†, Jürgen Teich†, Unmesh D. Bordoloi‡, Samarjit Chakraborty⋆

†University of Erlangen-Nuremberg, Germany
{glass,martin.lukasiewycz,teich}@cs.fau.de

‡Verimag, France
unmesh.bordoloi@imag.fr

⋆Technical University of Munich, Germany
samarjit@tum.de

## ABSTRACT

In this paper, a design method for automotive architectures is proposed. The two main technical contributions are (i) a novel hardware/software architecture *encoding* that unifies a number of design steps, i.e., resource allocation, process binding, message routing, scheduling, and parameter estimation for the processor and bus schedulers, and (ii) a hybrid scheme that allows different timing analysis techniques to be applied to different bus protocols (viz., CAN and FlexRay) within the same architecture in order to derive global performance estimates such as end-to-end delays of messages. The use of the compact encoding technique substantially reduces the underlying search space, and the hybrid timing analysis scheme allows the combination of known timing analysis techniques from the real-time systems domain. The proposed techniques were combined into a tool-chain and a real-life case study to illustrate their advantages.

## Categories and Subject Descriptors

C.3 [**Special-purpose and application-based systems**]: Real-time and embedded systems

## General Terms

Design, Performance

## Keywords

Automotive, Design Space Exploration, Timing Analysis

## 1. INTRODUCTION AND MOTIVATION

Today, it is fairly common for high-end cars to have more than 80 different electronic control units (ECUs) running a variety of distributed control applications and communicating via multiple buses and gateways implementing different communication protocols, e.g., CAN, LIN, and FlexRay. Optimally designing such a complex and heterogeneous ECU network poses several challenges, which has led to a lot of recent interest in developing design and analysis techniques specifically directed towards the automotive domain. Design methodologies for such complex embedded systems typically follow a series of design steps such as *resource allocation*, *process binding*, *message routing*, *scheduling* and *parameter estimation* for different processor and bus schedulers. Since a number of design constraints, e,g., real-time properties of safety-critical applications cannot be verified until all the design steps are complete and a concrete implementation is derived, such that multi-phase approaches hardly come up with global optimal solutions.

**Contributions:** This paper adresses the general design space exploration problem, cf. Fig. 1, by proposing an architecture encoding scheme that allows a number of hardware/software co-design tasks – such as resource allocation, process binding, message routing, scheduling, etc. – to be represented as linear constraints with binary variables. As a result, the design space exploration problem can now be formulated as a *binary search problem*, and the drawbacks associated with multi-phase design approaches can largely be avoided. The encoding scheme also substantially reduces the underlying search space and solutions to the resulting search problem are complete feasible implementations which respect all system constraints. As a result, it is now possible to use combinations of binary search and efficient heuristics for quickly solving multi-objective optimization problems arising in real-life automotive designs.

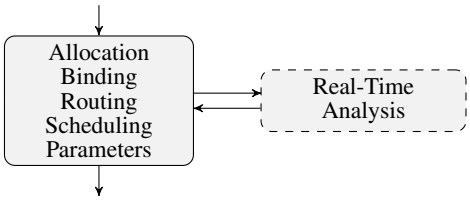


**Figure 1: Proposed design space exploration approach.**

The presented techniques are geared towards automotive architectures, consisting of a large and heterogeneous collection of ECUs, buses, and gateways running distributed control applications and processing several message streams connecting sensors to actuators. The optimization schemes clearly need to rely on timing *analysis* and performance *evaluation* techniques. Given the heterogeneous nature of the architectures in the automotive domain, having a single general-purpose timing analysis technique for all resources is difficult and often impossible. For example, different bus protocols such as CAN and FlexRay require very different modeling and analysis techniques. To get around this problem, a hybrid analysis technique is proposed which allows results from different timing analysis techniques to be composed together. In particular, it is shown how response time analysis techniques – which use a system of recurrence relations – for CAN protocol can be used with timing analysis techniques for FlexRay to derive overall timing properties of architectures containing buses of both these types.

Both, the architecture encoding scheme, as well as the hybrid timing analysis technique have been implemented into a tool-chain which is capable of analyzing real-life case studies from the automotive electronics domain. The utility of the proposed approach is illustrated through a detailed case study later in this paper.

**Related Work:** Resource allocation and process binding strategies are studied in [1, 11], mostly based on either exact approaches like *Integer Linear Programs* (ILPs) or using meta-heuristics like *Evolutionary Algorithms* (EAs). Process and message scheduling as well as the parameter estimation has been thoroughly studied in approaches like [4]. There exist design space exploration approaches that unify some or all steps in a single exploration tool, cf., e.g., [16]. However, these approaches are not able to restrict their search space to the set of feasible implementations only. This leads to serious drawbacks in case few feasible implementations exist due to stringent constraints. Real-time analysis of automotive networks has been studied in [6, 14]. For the sake of flexibility and extensibility, the work at hand proposes a jitter propagation model that utilizes existing analysis techniques, depending on the individual resource type and used scheduler.

## 2. DESIGN SPACE EXPLORATION

This section presents the proposed one-step design space exploration approach for automotive networks. After an introduction of the used exploration model, the encoding of the tasks of resource allocation, process binding, message routing, scheduling, and pa-

*Supported in part by the German Science Foundation (DFG), SFB 694

rameter estimation as a binary search problem based on linear constraints is presented. The combination of the binary search problem and a meta-heuristic to ensure a fast convergence to the optimal implementations with respect to multiple objectives completes this section.

The exploration model is defined by a *specification* that consists of an *application* and an *architecture*. From this specification, various *implementations* can be derived by defining the *allocation* of the architecture and the *mapping* of the application, the *routing* of messages and the used *parameters*. The specification consists of an *architecture graph* $G_R$ and an *application graph* $G_T$ : The architecture is given by a directed graph $G_R(R, E_R)$. The vertices $R$ represent resources such as ECUs ($R_{ECU} \subset R$), buses like FlexRay ($R_{FR} \subset R$) or CAN ($R_{CAN} \subset R$), communication controllers ($R_{CC} \subset R$), as well as sensors and actuators. The directed edges $E_R$ indicate available communication connections between two resources. The application is given by a bipartite directed graph $G_T(T, E_T)$ with $T = P \cup M$. The vertices $T$ are either processes $p \in P$ or messages $m \in M$. Each edge $e \in E_T$ connects a vertex in $P$ to one in $M$, or vice versa. Each process can have multiple incoming edges that indicate the data dependencies to communication information of the predecessor message. On the other hand, each message has exactly one predecessor process as the sender, but a process can of course have multiple successor messages. To allow multicasts, each message can have multiple successor processes. Each process $p \in P$ can be implemented on a unique set of resources $R_p \subseteq R$. Each message $m \in M$ can be routed on a subset of resources from $R_m$ with $R_m \subseteq R$. An implementation consists of the *allocation graph* $G_A$ that is deduced from the architecture graph and a function $i$ that maps the application onto the allocation graph. The allocation is a directed graph $G_A(A, E_A)$ that is an induced subgraph of the architecture graph $G_R$. The allocation contains all resources that are available in the current implementation and the edges are induced from the graph $G_R$. Each process $p \in P$ is bound to exactly one allocated resource $i(p)$ such that $i(p) \in (A \cap R_p)$. Each message in $m \in M$ is routed on a tree that is a subgraph of the allocation such that $i(m) \subseteq G_A$ with all vertices in $R_m$. These bindings and routings have to be performed such that all data dependencies given by the following two conditions are satisfied:

1. For each message $m \in M$, the root of the routing has to equal the binding of the predecessor sender process $p \in P$. It holds:

$$\forall (p, m) \in E_T : root(i(m)) = i(p)$$

2. For each process $p \in P$ the routings of the predecessor message $m \in M$ have to be routed on the same resource as the binding of process $p$. The following holds:

$$\forall (c, m) \in E_T : i(p) \in i(m)$$

An implementation is *feasible* if all requirements regarding the process and message mapping, the data dependencies, and the system constraints are fulfilled.

With the definition of a feasible implementation, the task of the *design space exploration* can be formulated as the following multi-objective optimization problem:

DEFINITION 1 (DESIGN SPACE EXPLORATION).
optimize $f(\mathbf{x})$
subject to:
$\mathbf{x}$ is a feasible *implementation*

In real-world problems, the objective function $f$ consists of multiple functions including also non-linear equations. In single-objective optimization, the feasible set of networks is totally ordered, whereas in multi–objective optimization problems, the feasible set is only partially ordered and, thus, there is generally not only one global optimum, but a set of *Pareto solutions*. A Pareto-optimal solution is better in at least one objective when compared to any other feasible solution.

## 2.1 Model Encoding

In the following, a binary search problem is defined such that a solution $\mathbf{x}$ corresponds to a *feasible* implementation $x$, regarding the allocation of resources, the binding of processes, the routing of messages, and the parameter set. The symbolic encoding consists of the following binary variables:

- $\mathbf{r}$ - one variable for each resource $r \in R$ indicating whether this resource is allocated (1) or not (0).
- $\mathbf{p_r}$ - one variable for each pair of process $p \in P$ and available resources $r \in R_p$, indicating whether the process is bound onto the resource $r$ (1) or not (0).
- $\mathbf{m_r}$ - one variable for each message $m \in M$ and available resources $r \in R_m$, indicating whether $m$ is routed over the resource $r$ (1) or not (0).
- $\mathbf{m_{r,n}}$ - one variable for each message and resource pair indicating on which communication step $n \in \mathbb{N}$ (messages are propagated in steps) a message is routed over the resource.

The linear constraints are formulated as follows:

$$\forall p \in P : \quad \sum_{r \in R_p} \mathbf{p_r} = 1 \tag{1a}$$

$$\forall m \in M : \quad \sum_{r \in R_m} \mathbf{m_{r,0}} = 1 \tag{1b}$$

$$\forall m \in M, p \in \{\tilde{p} | (\tilde{p}, m) \in E_T\}, r \in R_p \cap R_m : \quad \mathbf{p_r} - \mathbf{m_{r,0}} = 0 \tag{1c}$$

$$\forall p \in P, m \in \{\tilde{m} | (\tilde{m}, p) \in E_T\}, r \in R_p \cap R_m : \quad \mathbf{m_r} - \mathbf{p_r} \geq 0 \tag{1d}$$

$$\forall m \in M, r \in R_m : \quad \sum_{i=1}^{n} \mathbf{m_{r,i}} \leq 1 \tag{1e}$$

$$\sum_{i=1}^{n} \mathbf{m_{r,i}} - \mathbf{m_r} \geq 0 \tag{1f}$$

$$\forall m \in M, r \in R_m, i = \{1, ..., n\} : \quad \mathbf{m_r} - \mathbf{m_{r,i}} \geq 0 \tag{1g}$$

$$\forall m \in M, r \in R_m, i = \{1, ..., n-1\} : \quad -\mathbf{m_{r,i+1}} + \sum_{\tilde{r} \in R_m \wedge e = (\tilde{r}, r) \in E_R} \mathbf{m_{\tilde{r},i}} \geq 0 \tag{1h}$$

$$\forall p \in P, r \in R_p : \quad \mathbf{r} - \mathbf{p_r} \geq 0 \tag{1i}$$

$$\forall m \in M, r \in R_m : \quad \mathbf{r} - \mathbf{m_r} \geq 0 \tag{1j}$$

$$\forall r \in R : \quad -\mathbf{r} + \sum_{m \in M \wedge r \in R_m} \mathbf{m_r} + \sum_{p \in P \wedge r \in R_p} \mathbf{p_r} \geq 0 \tag{1k}$$

Equation (1a) ensures that each process is bound exactly once. Equations (1b) and (1c) imply that each message has exactly one root that equals the used resource of the predecessor process. Analogously, for each process the predecessor messages have to be routed on the corresponding resources as stated in Equation (1d). Equation (1e) ensures that a message can pass a resource at most once such that no cycles occur in the route of a message. A message has to exist in one communication step on a resource in order to be correctly routed on this resource as implied by the Equations (1f) and (1g). Equation (1h) states that a communication is only possible between adjacent resources. The Equations (1i) and (1j) imply that a process or message, respectively, is bound or routed on an allocated resource only. On the other hand, Equation (1k) states that a resource is only allocated if at least one process is bound or a message is routed on this resource.

Besides the constraints arising from resource allocation, process binding, and message routing, additional constraints regarding the system and the parameters of its components have to be respected. In this work, an encoding for system constraints that depend on the chosen parameters of a system component is proposed. In particular, this encoding carries out the task of parameter estimation implicitly.

In the automotive area, stringent bus load constraints are applied to the used CAN buses. The maximal rational load of a CAN bus is manufacturer-dependent and commonly between $0.4$ and $0.6$ [12]. The capacity of a high-speed CAN is $64,000$ byte/s and the low-speed CAN has a capacity of $16,000$ byte/s. The constraints are formulated as follows:

$\forall r \in R_{CAN}$ :

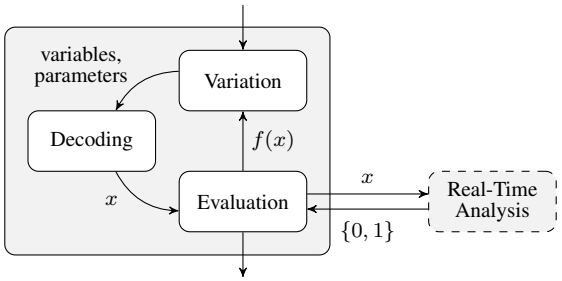


**Figure 2: Proposed design space optimization flow.**

$$\sum_{m \in M} \lceil \frac{\sigma_{CAN}(m)}{\rho(m)} \rceil \cdot \mathbf{m_r} \leq \lfloor \lambda(r) \cdot 64000 \rfloor \cdot \mathbf{r_{hi}} + \lfloor \lambda(r) \cdot 16000 \rfloor \cdot \mathbf{r_{lo}} \tag{2a}$$

$$\mathbf{r_{hi}} + \mathbf{r_{lo}} = 1 \tag{2b}$$

- $\sigma_{CAN} : M \to \mathbb{N}$ - size of bytes of each message (including additional CAN overhead per message)
- $\rho : M \to \mathbb{N}$ - period of a message
- $\lambda : R_{CAN} \to [0; 1]$ - maximal rational load of the bus

Equation (2a) states that the load of a CAN bus does not exceed the allowed bus load. Equation (2b) ensures that exactly one CAN variant, high-speed or low-speed, is chosen.

With respect to energy consumption, ECU processors can be set to different operation modes leading to different characteristics regarding computational capacity and energy consumption. The constraints are formulated as follows:
$\forall r \in R_{ECU}$ :

$$\sum_{p \in P} \lceil \frac{\sigma_{ECU}(p)}{\rho(p)} \rceil \cdot \mathbf{p_r} \leq \sum_{o \in \mu(r)} \lfloor \lambda(r) \cdot \gamma(r, o) \rfloor \cdot \mathbf{r_o} \tag{3a}$$

$$\sum_{o \in \mu(r)} \mathbf{r_o} = 1 \tag{3b}$$

- $\sigma_{ECU} : P \times R_{ECU} \to \mathbb{N}$ - instructions of a process on an ECU
- $\rho : P \to \mathbb{N}$ - period of a process
- $\mu : R_{ECU} \to 2^O$ - set of operation modes of an ECU
- $\gamma : R_{ECU} \times O \to \mathbb{N}$ - the computational capacity of an ECU under a given mode in instructions per second
- $\lambda_{ECU} : R_{ECU} \to [0; 1]$ - the maximal utilization of an ECU

Equation (3a) states that the computational load of an ECU bus does not exceed the allowed utilization, Equation (3b) ensures that exactly one operation mode is chosen.

Given a single solution $\mathbf{x}$ of this search problem, the corresponding implementation $x$ is deduced by constructing the allocation from the $\mathbf{r}$ variables, the binding for each process from the $\mathbf{p_r}$ variables, the routing of the message from the $\mathbf{m_r}$ and $\mathbf{m_{r,n}}$ variables, and the system parameters from the corresponding variables.

## 2.2 Optimization

Common optimization approaches that are based on *Integer Linear Programs*, cf. [11], or *Evolutionary Algorithms*, cf. [8], only are either restricted to a single linear objective function or do not perform well on optimization problems with many constraints and few feasible solutions. With the linear constraints using binary variables introduced previously, the design space exploration problem as stated in Def. 1 can be carried out efficiently by using a heuristic *SAT decoding* optimization approach based on [9]. This hybrid optimization approach based on an *Evolutionary Algorithm* and a *PB (pseudo Boolean) solver* allows the optimization of multiple conflicting and non-linear objectives under linear constraints in a binary search space, cf. Fig. 2. In particular, the approach *varies* the variables of the binary search problem as well as additional parameters used, e.g., for process and message priorities or scheduling policies. These additional parameters are optimized in parallel to the binary search problem if they do not affect the feasibility of the implementation. The *decoding* solves the binary search problem with respect to the varied variables and, thus, gathers feasible implementations. Decoded implementations are *evaluated* to determine their objectives and their real-time properties are verified. The results from the evaluation guide the variable and parameter variation in the next iteration. The optimization approach iteratively improves found implementations such that with a higher runtime and more evaluated implementations, respectively, the quality of the results increases.

# 3. HYBRID TIMING ANALYSIS

Next, a timing analysis is presented that determines the real-time properties for an application by handling each process and message separately and only considering the jitter propagation. The distinction between processes and messages allows a specific analysis which has an effect on the response time delay of multicast and multihop routed messages. The jitter propagation model allows a component-wise analysis either with a common method based on recurrence functions, e.g., [7, 14], or with more sophisticated methods like the *Real-Time Calculus* (RTC) [5, 13].

## 3.1 Function Timing Analysis

Each process $p \in P$ is implemented on a single resource $i(p) \in R$ such that the worst case response time $a_p$ is calculated dependent on the scheduler of this resource. On the other hand, each message $m \in M$ is implemented on the tree $i(m) \in G_A$ such that the response $a_{m,r}$ has to be determined for each resource in the graph $i(m)$. The end-to-end latency of one path of the application $\pi$ with $\pi \subseteq T$ is determined as follows:

$$a_\pi = \sum_{p \in \pi \cap P} a_p + \sum_{m \in \pi \cap M} \sum_{r \in route(m)} a_{m,r} \tag{4a}$$

with

$$route(m) = (i(p), ..., i(\tilde{p})) \subseteq i(m) \text{ and } (p, m), (m, \tilde{p}) \in G_T \tag{4b}$$

This means that the end-to-end latency is given by the sum of all process execution times and message response times that arise on the path $\pi$.

Given a function of the application $G_F \subseteq G_T$ which is described by a directed acyclic graph, the worst case execution time of the function is:

$$a_{G_F} = \max_{\pi \in \{\text{all paths in } G_F\}} a_\pi \tag{5}$$

Instead of enumerating all paths in $G_F$ and searching for the maximal $a_\pi$, the evaluation is performed efficiently by applying the *Bellman-Ford algorithm* [2] to the graph $G_F$ with the cost for the nodes being $a_p$ or $a_{m,r}$, respectively. Thus, the complexity of Equation (5) is $O(|F| \cdot |E_F|)$ with $F$ being the nodes of $G_F$ and $E_F$ the edges, respectively.

The real-time analysis of an application requires the process execution times $a_p$ and messages response times $a_{m,r}$. For this purpose, a jitter propagation model also becomes necessary. Here, the jitter is the unwanted variation between two consecutive periodic processes executions or messages. The output jitter of the process execution is $j_p$ for $p \in P$ and a message is $j_{m,r}$ for $m \in M$ and the resource $r \in R$. The input jitter $j_p^*$ of a process $p \in p$ is system dependent if this process is time-triggered or the maximum of the jitter values of the predecessor messages $\max(j_{pred(p),i(p)})$ if the process is event-triggered, respectively. The input jitter $j_{m,r}^*$ of a message $m \in M$ with the predecessor task $p$ with $(p, m) \in E_T$ on the resource $r \in R$ is calculated as follows:

$$j_{m,r}^* = \begin{cases} j_p, & \text{if } r = root(i(m)) \\ j_{m,pred(r)}, & \text{else} \end{cases} \tag{6}$$

Where $pred(r)$ denotes the predecessor resource of the resource $r$ on the routing tree $i(m)$. Taking these jitter values into account allows an exact and flexible real-time analysis independent of the underlying methodology. With the given jitter values, each process execution time $a_p$ and messages response time $a_{m,r}$ can be calculated independently as shown exemplarily in the following subsections.

### 3.1.1 Process Execution Time (Priority Scheduling)

Given a priority-based scheduler with preemption, the worst case response time and jitter is calculated as follows:

$$a_p = c_p + \sum_{\tilde{p} \in hp(p)} \left\lceil \frac{a_p + j_p^*}{\tau_{\tilde{p}}} \right\rceil \cdot c_{\tilde{p}} \tag{7a}$$

$$j_p = a_p - c_p \tag{7b}$$

Where $c_p$ ($c_{\tilde{p}}$) is the required time for the execution of the process $p$ ($\tilde{p}$), $\tau_{\tilde{p}}$ is the period of the process $\tilde{p}$, and the function $hp(p)$ determines all processes that are implemented on the same resource as $p$ and have a higher priority.

### 3.1.2 Message Response Time on CAN (ET)

For an event-triggered (ET) CAN bus $r \in R_{CAN}$, one message $m \in M$ induces the following response time and jitter:

$$a_{m,r} = c_{m,r} + b_{m,r} + \sum_{\tilde{m} \in hp(m)} \left\lceil \frac{a_{m,r} + j_{m,r}^* - c_{m,r}}{\tau_{\tilde{m},r}} \right\rceil \cdot c_{\tilde{m},r} \tag{8a}$$

$$j_{m,r} = a_{m,r} + c_{m,r} \tag{8b}$$

Here, $c_{m,r}$ ($c_{\tilde{m},r}$) is the transmission time of the message $m$ ($\tilde{m}$) on the bus $r$, $\tau_{\tilde{m}}$ is the period of the message $\tilde{m}$, and the function $hp(m)$ determines all message that are routed over the CAN bus $r$ and have a higher priority than $m$. In contrast to the priority-based scheduler, preemption is not possible on the CAN bus and $b_{m,r}$ is added which denotes the maximal transmission time of one lower priority message.

### 3.1.3 Message Response Time on FlexRay (TT)

Each message $m \in M$ that is routed on the time-triggered (TT) *static segment* of the FlexRay bus $r \in R_{FR}$ induces the response time and jitter as follows:

$$a_{m,r} = c_{m,r} + s_{m,r} \tag{9a}$$

$$j_{m,r} = c_{m,r} + \left\lceil \frac{j_{m,r}^*}{s_m, r} \right\rceil \cdot s_{m,r} \tag{9b}$$

Where $c_{m,r}$ denotes the transmission time for a static slot and $s_{m,r}$ the maximal time difference (usually a multiple of the cycle time) between two static slots that transmit the message $m$.

The analysis of the *dynamic segment* of the FlexRay bus is based on the demand-bound criteria approach which is well-known in the real-time scheduling literature [3]. This is approach is adapted to specifically model the dynamic segment of the FlexRay protocol. The model and the associated analysis mechanisms have been integrated into a publicly-available Matlab-based tool called the Real-Time Calculus Toolbox [15].

## 4. CASE STUDY

The methodology presented in this paper, cf. Fig. 2, is realized as a tool-chain: The proposed exploration is implemented using the OPT4J framework [10] while the real-time analysis is based on recurrence relations and the RTC toolbox [15]. The methodology was applied to a case study, modeling a typical automotive subnetwork: The network architecture consists of 15 ECUs, connected via two CAN buses, one FlexRay bus, and a central gateway. The 9 sensors, and 5 actuators are connected via LIN buses to the ECUs. The application consisting of four functions, an *adaptive cruise control* (ACC), a *brake-by-wire* (BW), an *air conditioning function* (C1), and a *multimedia control* (C2), with 46 processes and 42 messages in total has to be mapped onto the given architecture.

The subnetwork is optimized in terms of the monetary cost in Euro(€) and energy consumption in Watts. Given constraints are the maximal load $40\%$ for the CAN bus, the maximal utilization $95\%$ for ECUs as well as real-time constraints regarding the end-to-end delay from the sensors to the corresponding actuators, given in Table 1.

The optimization of this subnetwork was performed by an exploration with the presented approach including a parallel optimization of priorities of the processes and messages as well as the scheduling of the messages on the static or dynamic segment of the FlexRay bus. The exploration was performed with an Evolutionary Algorithm meta-heuristic, improving the implementations iteratively by $5075$ objective evaluations within $4$ hours and $43$ minutes on an Intel Core 2 Quad $2.66$ GHz with 3GB RAM. Only 21 seconds are spent on the variation and decoding while the remaining time accounts for evaluation, in particular the real-time analysis due to a delaying file interface of the RTC.

| | energy(W) | cost(€) | ACC(ms) | BW(ms) | C1(ms) | C2(ms) |
|---|---|---|---|---|---|---|
| deadline | - | - | 100 | 50 | 30 | 50 |
| ref. | 432.5 | 214.4 | 99.7 | 30.2 | 11.8 | 32.8 |
| impl. 1 | 394.9 | 189.1 | 43.1 | 37.5 | 17.3 | 39.6 |
| impl. 2 | 396.2 | 184.7 | 41.2 | 34.0 | 15.1 | 49.0 |
| impl. 3 | 399.5 | 182.8 | 41.2 | 34.0 | 12.5 | 49.0 |
| impl. 4 | 416.7 | 182.3 | 38.1 | 28.9 | 13.9 | 39.7 |

**Table 1: Detailed results of the best found implementations.**

The results of the optimization are given in Table 1. Given is a hand-made reference implementation with the monetary cost $214.40$€ and an energy consumption of $432.5$ Watts that respects all real-time constraints. Four non-dominated high quality implementations are found improving the reference implementation in both objectives, monetary cost and energy consumption. The found implementations allow to decrease the monetary cost by $11.8\%$ to $15\%$ while decreasing the energy consumption by about $3.6\%$ to $8.7\%$ at the same time. These results are obtained by binding the processes such that a higher utilization is achieved and some ECUs become redundant and can be removed from the implementation. At the same time, the priorities of the messages and processes are varied such that the real-time constraints are still respected.

## 5. CONCLUSION

The paper presents a unified design space exploration approach for real-time automotive networks that performs the design steps of resource allocation, process binding, message routing, scheduling, and parameter optimization. A combination of a binary search problem and a meta-heuristic allows to reduce the search space to the feasible implementations only while achieving a fast convergence to the optimal implementations. For the verification of the real-time constraints, a hybrid timing analysis is introduced that combines the benefits of known analysis techniques that each fit best for particular resource types like buses, ECUs, etc. A case study modeling an automotive subsystem shows the capability of the proposed methodology.

## 6. REFERENCES


[1] N. Banerjee and R. Kumar. Multiobjective network design for realistic traffic models. In *Proceedings of GECCO '07*, pages 1904–1911, 2007.

[2] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[3] G. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer, 2005.

[4] A. Davare, Q. Zhu, M. D. Natale, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli. Period optimization for hard real-time distributed automotive systems. In *Proceedings of DAC '07*, pages 278–283, 2007.

[5] A. Hagiescu, U. D. Bordoloi, S. Chakraborty, P. Sampath, P. V. V. Ganesan, and S. Ramesh. Performance analysis of flexray-based ECU networks. In *Proceedings of DAC '07*, pages 284–289, 2007.

[6] A. Hamann, R. Racu, and R. Ernst. Formal methods for automotive platform analysis and optimization. In *Proceedings Future Trends in Automotive Electronics and Tool Integration Workshop (DATE Conference)*, 2006.

[7] M. G. Harbour, M. H. Klein, and J. P. Lehoczky. Timing analysis for fixed-priority scheduling of hard real-time systems. *IEEE Trans. Softw. Eng.*, 20(1):13–28, 1994.

[8] R. Kumar, P. K. Singh, and P. P. Chakrabarti. Multiobjective EA approach for improved quality of solutions for spanning tree problem. In *Proceedings of EMO '05*, pages 811–825, 2005.

[9] M. Lukasiewycz, M. Glaß, C. Haubelt, and J. Teich. Sat-decoding in evolutionary algorithms for discrete constrained optimization problems. In *Proceedings of CEC '07*, pages 935–942, 2007.

[10] Opt4J. The optimization framework for java. http://www.opt4j.org/, Version 1.5.

[11] D. Rajan and A. Atamtürk. A directed cycle-based column-and-cut generation method for capacitated survivable network design. *Networks*, 43(4):201–211, 2004.

[12] K. Richter and R. Ernst. How OEMs and suppliers can face the network integration challenges. In *Proceedings of DATE '06*, pages 183–188, 2006.

[13] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *Proceedings of ISCAS '00*, pages 101–104, 2000.

[14] K. Tindell, A. Burns, and A. Wellings. Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3:1163–1169, 1995.

[15] E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`, 2006.

[16] H. Zeng, A. Davare, A. Sangiovanni-Vincentelli, S. Sonalkar, S. Kanajan, and C. Pinello. Design space exploration of automotive platforms in metropolis. In *SAE Congress*, 2006.