Article:

J. Schlatow, M. Mostl, S. Tobuschat, T. Ishigooka and R. Ernst, "Data-Age Analysis and Optimisation for Cause-Effect Chains in Automotive Control Systems," 2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES), Graz, Austria, 2018, pp. 1-9.

10.1109/SIES.2018.8442077

Data-Age Analysis and Optimisation for Cause-Effect Chains in Automotive Control Systems

Johannes Schlatow*, Mischa Möstl*, Sebastian Tobuschat*, Tasuku Ishigooka† and Rolf Ernst*
*TU Braunschweig, Institute of Computer and Network Engineering, Braunschweig, Germany
{schlatow,moestl,tobuschat,ernst}@ida.ing.tu-bs.de
†Center for Technology Innovation – Controls, Research & Development Group, Hitachi Ltd., Ibaraki, Japan
tasuku.ishigoka.kc@hitachi.com

***Abstract*—Automotive control systems typically have latency requirements for certain cause-effect chains. When implementing and integrating these systems, these latency requirements must be guaranteed e.g. by applying a worst-case analysis that takes all indeterminism and limited predictability of the timing behaviour into account. In this paper, we address the latency analysis for multi-rate distributed cause-effect chains considering static-priority preemptive scheduling of offset-synchronised periodic tasks. We particularly focus on data age as one representative of the two most common latency semantics. Our main contribution is an Mixed Integer Linear Program-based optimisation to select design parameters (priorities, task-to-processor mapping, offsets) that minimise the data age. In our experimental evaluation, we apply our method to two real-world automotive use cases.**

I. Introduction

Automotive control systems are a prominent representative of increasingly complex Cyber-Physical Systems (CPSs), particularly with regard to the emergence of Advanced Driver-Assistance Systems (ADASs). These systems are typically constrained by certain Sensor-to-Actuator (StA) delays that must not be exceeded. These delays are commonly captured as (worst-case) latencies of so-called *cause-effect chains*, which describe the data flow as it is being processed by different parts (i.e. tasks) of the software system. In particular, when it comes to multi-rate control systems, sampling of data values is applied. This leads to a time-triggered (periodic) data exchange along the cause-effect chains which is well-understood by control theory. Note that, depending on the function, i.e. the particular sensor and actuator, one distinguishes between two latency semantics: reaction time and data age (cf. [1]). The former describes the latest time it takes a control system to react to a certain sensor value whereas the latter describes the maximum age of the input data that was used for a particular output. A too large data age often reduces the control performance such that data age constraints must be considered for certain chains.

When designing such control systems in simulation, certain implementation details are not taken into account as they remain unknown at this stage. More precisely, critical requirements must still be checked and verified after implementation and integration on the target platform. For instance, worst-case analysis tackles indeterminism and predictability issues (e.g. jitter) and provides safe upper/lower bounds. When it comes to control systems, response times and latencies are the main subjects of such an analysis. As these are influenced by particular design parameters such as scheduling priorities and task-to-processor mapping, selecting parameters such that the given requirements are satisfied becomes an important objective of the software integration. In particular, we focus on data age constraints that not only should be satisfied but – for better robustness – can further be minimised.

In this paper, we present an Mixed Integer Linear Program (MILP)-based optimisation of these design parameters that we apply to two real-world automotive use cases in order to minimise the data age of given cause-effect chains. As this work was motivated by one of these use cases, we first introduce and specify its scenario in Section II. From this, we derive our system model and problem statement in Section III and summarise related work in Section IV. In Section V, we briefly present the response-time and data-age analysis for which we derive an MILP formulation in Section VI. We apply and evaluate this formulation to our use case and a similar automotive benchmark in Section VII before we summarise and conclude our findings in Section VIII.

II. Automotive use case

Table I
Task set **A** (ADAS)

Core	Task	Period	WCET	BCET
0	ISR	550 μs	20 μs	5 μs
0	Task A	100 ms	250 μs	70 μs
0	Task B	10 ms	500 μs	40 μs
0	Task C	50 ms	2600 μs	1800 μs
0	Task D	250 μs	160 μs	45 μs
0	Task E	10 ms	750 μs	200 μs
1	Task G	10 ms	200 μs	1 μs
1	Task H	50 ms	3800 μs	2600 μs
1	Task I	10 ms	110 μs	2 μs
1	Task J	10 ms	2500 μs	2 μs
1	Task K	10 ms	500 μs	3 μs
1	Task L	2000 ms	300 μs	150 μs

The following is a brief description of a real ADAS use case from Hitachi Ltd. that underlines the relevance of this work. In addition, we will later show the practical applicability of our analysis w.r.t. this use case in Section VII.

The use case comprises a dual-core system running a set of periodic real-time tasks as specified in Table I. Note that this

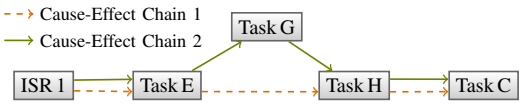

Figure 1. Task data dependencies

task set also incorporates best-effort tasks as well as a single sporadic task (ISR). Both cores employ local Static-Priority Preemptive (SPP) scheduling. Data dependencies between the tasks lead to two distributed cause-effect chains as illustrated in Figure 1. A data dependency originates when a task writes output data (e.g. sensor value) into shared memory which is read and used as input data by another task. The performance of these chains is latency critical in terms of the maximum data age.

Several obstacles arise for this use case. Due to a high load on both cores, overapproximation during Response-Time Analysis (RTA) can quickly result in an overloaded CPU. Hence, Worst-Case Execution Times (WCETs) and Best-Case Execution Times (BCETs) for this task set have been extracted from traces and summarised in Table I. The measured execution times are depicted as a Tukey boxplot in Figure 2 in which the red bar indicates the median and the box includes the data points between the 25th percentile and the 75th percentile.

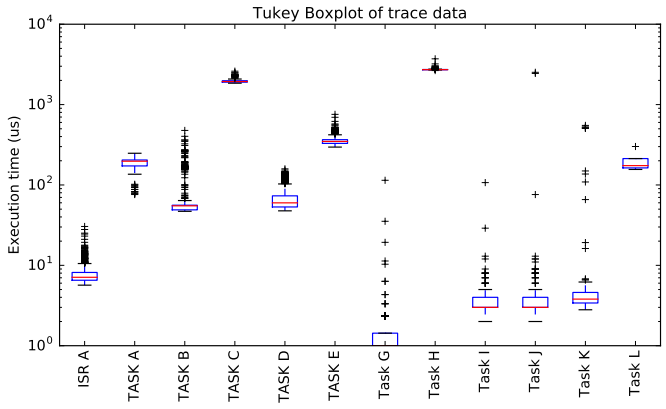

Figure 2. Tukey Boxplot of traced execution times

W.r.t. data age analysis, the time-triggered nature of the inter-task communication (sampling) must be taken into account as this imposes additional delays depending on the sampling periods and phasing.

Furthermore, as the cause-effect chains stretch across both cores, we need to take synchronisation and activation phases/offsets into account. We can assume that both cores use a common timer such that activation periods can be synchronised. However, the phasing can significantly influence the data age and the Worst-Case Response Times (WCRTs). Hence, a data age analysis must not only respect the different sampling periods but also the activation phases in order to provide reasonable bounds.

In the scope of this paper, we not only want to perform an RTA and data age analysis for this system but also want to answer the following questions:

1) Can we improve the data-age bound if we not only select scheduling priorities but also include (and wisely choose) the activation phases?
2) Can we also change the task-to-core mapping to improve the data-age bound?

III. Problem Statement and System Model

In this section, we formulate the underlying system model and problem statement for the remainder of this paper.

A. Task model

The workload is modelled by a set of periodic tasks $\{\tau_1, \ldots, \tau_n\}$ and a set of sporadic tasks $\{\tau_{n+1}, \ldots, \tau_m\}$. A job of a task τ_i is released at most every T_i time units. For periodic tasks, we only consider harmonic periods, i.e. $\forall i \neq j : \min(T_i, T_j) = gcd(T_i, T_j)$. The release of a periodic task's job can be further delayed by ϕ_i time units (activation phase) relative to the common timer. The relative arrival time of the j-th job of τ_i is denoted by $a_i(j)$. The execution time of each job is bounded from above by C_i^+ time units and from below by C_i^- time units. A job must be completed before the next job of the same task arrives (constrained deadlines). More precisely, a job has the relative deadline $D_i(j) \leq (j+1) \cdot T_i$. We denote the relative completion (exit) time by $e_i(j)$ and define the response time $R_i(j)$ as the time between $a_i(j)$ and $e_i(j)$, such that $\forall j : e_i(j) \leq D_i(j) \Rightarrow R_i(j) + \phi_i \leq T_i$. A task τ_i will read input data from shared memory at the beginning of its execution and will write output data upon completion. More formally, we assume $a_i(j) \leq r_i(j) < w_i(j) = e_i(j)$, where $r_i(j)$ and $w_i(j)$ denote the relative read and write times. Figure 3 illustrates this model for two tasks $\{\tau_i, \tau_j\}$ with $T_i = 2T_j$.

B. Cause-effect chains

A cause-effect chain Ψ is given by a sequence of tasks $\Psi = (\tau_i, \tau_{i+1}, \tau_{i+2}, \ldots)$ which corresponds to a sequence of alternating read and write events $(r_i, w_i, r_{i+1}, w_{i+1}, r_{i+2}, \ldots)$, i.e. τ_{i+1} reads data from τ_i and writes data for τ_{i+2}. We differentiate two relevant latency semantics for Ψ, reaction time and data age as specified in [1]: Reaction time denotes the maximum time it may take for an input value (e.g. sensor) to cause a reaction (e.g. actuator). Data age describes the maximum age of the input data on which an output is based. This difference results from the register-based communication for which only the last written data is stored (last-is-best) for the reader. I.e. for undersampling, data is overwritten whereas, for oversampling, the same data will be read multiple times. For data age, the adjacent preceding write thus becomes relevant when looking at a particular read (last written data). In contrast, for reaction time, the adjacent subsequent read is of interest (first reaction to the written data).

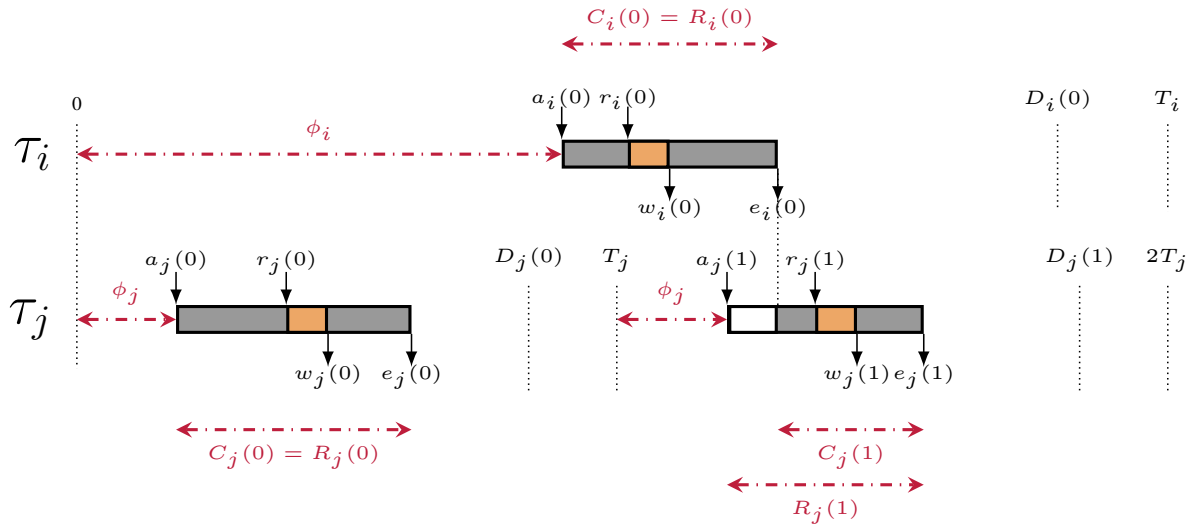

Figure 3. Illustration of our task model and terminology

C. Scheduling

The task set is scheduled on a given set of equal processors using partitioned SPP scheduling. For this, each task τ_i will be assigned a static priority π_i. Also, each task will be statically mapped to a single processors. On each processor, the pending job (released but not yet completed) with the highest priority will be scheduled. Higher-priority jobs can preempt the execution of lower-priority jobs. Note that lower numerical values of π_i correspond to higher priorities.

D. Problem statement

Based on the above model, we can now define our problem statement: Given a task set (τ_i) including periods T_i and execution time bounds C^+/C^-, a set of cause-effect chains (Ψ_j) and the number of processors m, we want to derive a priority assignment (π_i), processor mapping $(\tau_i \to [1, m])$ and activation phases (ϕ_i) that minimise the data age latencies of the chains and guarantees the implicit response-time constraints $(R_i + \phi_i \leq T_i)$.

IV. Related work

The problem of latency computation in multi-rate systems has first been approached by Feiertag et al. [2] with a compositional framework. In particular, the authors generalise end-to-end semantics, eliminate unreachable timing paths and provide bounds for different latency semantics but exclude offsets. Moreover, data age and response time are denoted as "last-to-last" and "first-to-first" semantics respectively. Latency analysis for cause-effect chains has also been addressed by the WATERS Industrial Challenge 2017 [3] for which several solutions have been presented [4]–[8]. Most recently, the cause-effect chain analysis with different communication paradigms and varying timing information was summarised by Becker et al. [9]. Our compositional approach reduces the cause-effect chain analysis to the calculation of pairwise write-to-read delays and therefore simplifies the MILP formulation. In contrast, Becker et al. take a holistic approach by looking at data propagation paths which may include more knowledge (if available) to further tighten the analysis.

Priority assignment is a well-known problem that must be faced for design and integration of real-time systems. Existing (optimal) techniques have recently been reviewed by Davis et al. [10]. Moreover, Davis and Burns [11] published a survey about real-time scheduling for multiprocessor systems which summarises fundamental results and research challenges including task-to-processor allocation. Wieder and Brandenburg [12] have presented an MILP formulation of RTA on multiprocessing systems with shared resources as a technique to solve the task-to-processor allocation and priority assignment. However, the existing methods only consider optimality w.r.t. local deadlines, i.e. response-time constraints, rather than end-to-end latencies for cause-effect chains.

V. Response-time and data-age analysis

In this section, we summarise the RTA considering activation phases (offset) which we apply for our system model. Furthermore, we present our data-age analysis for cause-effect chains. Before proceeding, let us first clarify some additional terminology and assumptions.

Because of limited timing predictability of real systems, we can only determine lower and upper bounds for certain time values. More precisely, we consider this for w, r, a, e, ϕ, C, R, for which we denote the lower/upper bounds by a superscript $-/+$, e.g. w^-/w^+.

Note that, as we only consider independent tasks that share a common time reference we do not consider any activation jitter. Even if the common time reference experiences jitter, all tasks will be affected in the same way. On the other hand, we do consider jittering offsets which can result from small variations in execution time of the scheduler.

In consequence, we assume that the lower and upper bounds of different offsets are ordered equally:

$$\phi_i^- > \phi_j^- \Leftrightarrow \phi_i^+ > \phi_j^+ \tag{1}$$

We further denote activation events a_i, a_j that have the same reference as *coinciding* activations $a_i(0)$ and $a_j(0)$. For harmonic periods (let $T_i > T_j$), $a_j(n)$ with $n = \frac{T_i}{T_j}$ coincides again with $a_i(1)$.

A. Response-time analysis

We apply a response-time analysis based on [13] but slightly adapted for our definition of response times. As argued above, we only consider jitter that originates from varying offsets. In addition, we can safely exclude certain interferers if their offsets and periods arrange favourably:

Theorem 1. *A higher-priority task τ_j can not interfere with a task τ_i if it (i.e. τ_j) has a shorter (harmonic) period ($T_i = nT_j$ with $n \in \mathbb{N}$) and is activated after τ_i's latest completion, i.e. $\phi_j \geq \phi_i + R_i^+$.*

Proof: First, the job of τ_j that coincides with τ_i at time t cannot interfere as it is only released after τ_i completed. Let $t = 0$ without loss of generality. Because of $T_i = nT_j$ and the implicit deadlines, any job of τ_j that starts within $[0, T_i]$ will also finish within this interval and thus cannot interfere with any later job of τ_i. ∎

We thus calculate the best-case (R^-) and worst-case response-times (R^+) as follows:

$$R_i^- = C_i^- \tag{2}$$

$$R_i^+ = C_i^+ + \sum_{j \in \mathcal{I}_i} \left\lceil \frac{R_i^+ + J_j}{T_j} \right\rceil \cdot C_j^+ \tag{3}$$

with $J_j = \phi_j^+ - \phi_j^-$ and $\mathcal{I}_i = \{j | \pi_j \leq \pi_i\} \setminus \{j | \phi_j \geq \phi_i + R_i^+ \wedge T_i = nT_j\}$, $n \in \mathbb{N}$.

Eq. (3) calculates the WCRT from the WCET of task τ_i itself and the maximum preemption time from every possible interferer (τ_j) that is activated every T_j in the time window from τ_i's activation to its completion (R_i^+). Note that, an activation jitter (J_j) of the interfering tasks increases this time window. Here, we derive the activation jitter from the activation offsets. In contrast to [13], we also exclude certain interferers from the set of possible interferers as proven in Theorem 1.

B. Data-age analysis

As defined in Section III, a cause-effect chain is a sequence of read and write events. In order to perform a data-age analysis, we need to bound the distance between consecutive read and write events. Note that the distance between a read and the subsequent write event (read-write distance) is bounded by the WCRT R^+.

Theorem 2. *The* read-write distance *for a task τ_i is bounded by the worst-case response time R_i^+.*

Proof: The read-write distance is calculated by $\max_n\{w_i(n) - r_i(n)\}$. As reading occurs after activation and writing before completion, i.e. $r_i(j) \geq a_i(j)$ and $w_i(j) \leq e_i(j)$, the read-write distance is bounded by $\max_n\{e_i(n) - a_i(n)\}$. By definition, the worst-case distance between $a_i(n)$ and $e_i(n)$ is referred to as R_i^+. ∎

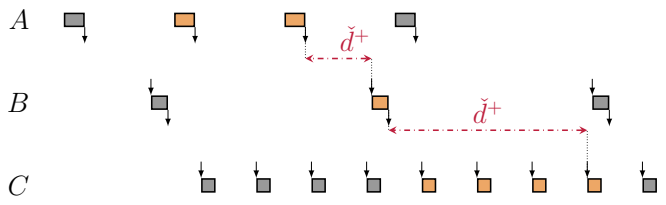

Figure 4. Write-read distances according to Eq. (12-14)

For the distance between a write and the subsequent read event (write-read distance), we must consider all possible candidates (jobs) of the producing task (writer) that may have provided the input data for a particular job of the consuming task (reader). Figure 4 illustrates this candidate search w.r.t. data-age semantics The figure depicts a Gantt chart for a cause-effect chain of three tasks A, B and C. Task B has a longer period than task A (undersampling) whereas task C has a shorter period than task B (oversampling). In this example, the jobs (candidates) that exchange data with a particular job of task B are highlighted. In order to compute the data-age delay, we are only interested in the directly preceding job of the writer (red arrows).

This candidate search must respect different sampling periods and indeterminate (but bounded) read/write times. However, by assuming harmonic periods, this search can be kept manageable. For this purpose, let us first derive the upper/lower bounds for the read and write times before we elaborate on the calculation of the data-age delay in more detail.

The lower/upper bounds for the activation and exit times of a job n of task τ_i are straightforwardly computed as follows:

$$a_i^-(n) = nT_i + \phi_i^- \tag{4}$$

$$a_i^+(n) = nT_i + \phi_i^+ \tag{5}$$

$$e_i^+(n) = nT_i + \phi_i^+ + R_i^+ \tag{6}$$

$$e_i^-(n) = nT_i + \phi_i^- + R_i^- \tag{7}$$

By definition, output data is written at completion, hence the write times are bounded by the exit times.

$$w_i^-(n) = e_i^-(n) \tag{8}$$

$$w_i^+(n) = e_i^+(n) \tag{9}$$

Input data is read at the very beginning of a job, i.e. as soon as it is scheduled for the first time. In consequence, the read time cannot be earlier than the arrival time and cannot be later than the Best-Case Response Time (BCRT) before the exit time.

$$r_i^-(n) = a_i^-(n) \tag{10}$$

$$r_i^+(n) = e_i^+(n) - R_i^- \tag{11}$$

With these bounds, we can now calculate the data-age delay (write-read distance) between τ_i and τ_{i+1}, which we denote by $\check{d}^+_{i,i+1}$. More precisely, we look at the coinciding activation cycles and decide whether the reader reads data that was written in the same or previous cycle. We distinguish two cases: oversampling and undersampling (including equal periods). Note that we can still bound the write-read distance for tasks with non-harmonic periods [4] (here: sporadic tasks, e.g. ISRs) but not leverage offsets in this case.

1) Oversampling: $(T_{i+1} < T_i)$

In case of oversampling, the same output data of τ_i is read multiple times by τ_{i+1}, i.e. we must consider the read times $r_{i+1}(n)$ with $n \in [0, \frac{T_i}{T_{i+1}} - 1]$. Depending on the write time of τ_i, τ_{i+1} might also read data from the previous cycle such that we need to take $w_i(-1)$ in addition to $w_i(0)$ into consideration. Thus, we look for the maximum between all possible read/write times taking into account that data might be produced in the previous cycle if $w_i^+(0) > r_{i+1}^-(n)$:

$$\check{d}^+_{i,i+1} \geq \max_{n \geq 0} \left(r^+_{i+1}(n) - \begin{cases} w_i^-(0) \\ w_i^-(-1) & \text{if } w_i^+(0) > r_{i+1}^-(n) \end{cases} \right) \tag{12}$$

2) Undersampling: $(T_{i+1} \geq T_i)$

For undersampling, we only need to consider a single reader job but multiple write times $w_{i+1}(n)$ with $n \in [-1, \frac{T_{i+1}}{T_i} - 1]$.

If the reader reads data produced in the same cycle, the data-age is bounded by:

$$\check{d}^+_{i,i+1} \geq \max_{n \geq 0} (r^+_{i+1}(0) - w_i^-(n)) \tag{13}$$

Otherwise, if the data may be produced in the previous cycle, we need to consider $w_i^-(-1)$:

$$\forall w_i^+(0) > r_{i+1}^-(0) : \check{d}^+_{i,i+1} \geq r^+_{i+1}(0) - w_i^-(-1) \tag{14}$$

Note that we do not consider overwrite effects in Eq. (13) and (14), i.e. if it is guaranteed that the output data is overwritten by the next job of τ_i before τ_j reads the data, these formulas are pessimistic. However, by using these bounds as a basis for our MILP formulation, we can simultaneously reduce or even eliminate overwrite effects along with the data age optimisation.

VI. MILP FORMULATION

In this section, we first summarise the MILP formulation for the RTA which is based on [12] but extended to consider offsets. Secondly, we present our MILP formulation for the data-age analysis of cause-effect chains. In combination, both parts will find activation offsets, priority assignments and processor mappings for a given task set such that the response-time constraints (deadlines) will be satisfied and data-age will be minimised.

We consider the tasks' best-case and worst-case execution times C_i^-/C_i^+, and periods T_i as constants (input parameters). In addition, as we want to find activation offsets, we introduce a constant jitter J_i to describe the variance between ϕ_i^+ and ϕ_i^-, i.e. $\phi_i^+ - \phi_i^- = 2J_i$.

We introduce the following decision variables:

- The binary variable $A_{i,k}$ which takes 1 iff τ_i is mapped to processor k.
- The binary variable $\pi_{i,p}$ which takes 1 iff τ_i has priority p.
- The integer variable $\phi_i \geq 0$ which decides the activation offset of τ_i.

A. Response-time analysis

The RTA is formulated by the following MILP constraints; for a more detailed explanation of these, please refer to [12]. Every task must be mapped to exactly one of the m processors:

$$\forall \tau_i : \sum_{k=1}^{m} A_{i,k} = 1 \tag{15}$$

Every task must be assigned to exactly one priority:

$$\forall \tau_i : \sum_{p=1}^{n} \pi_{i,p} = 1 \tag{16}$$

Every priority must be assigned at most once:

$$\forall 1 \leq p \leq n : \sum_{\tau_i} \pi_{i,p} \leq 1 \tag{17}$$

The additional binary variable $V_{x,i}$ is forced to 1 iff τ_x and τ_i are mapped to the same processor:

$$\forall \tau_x : \forall \tau_i, \tau_x \neq \tau_i : \forall k, 1 \leq k \leq m : V_{x,i} \geq 1 - (2 - A_{i,k} - A_{x,k}) \tag{18}$$

In order to distinguish between higher from lower priority tasks, the binary variable $X_{i,x}$ is introduced and shall take 1 iff τ_i has a higher priority than τ_x (lower p = higher priority). This is implemented by the following constraints which force $X_{i,x}$ to 0 iff τ_x is on a higher priority and by enforcing that either $X_{i,x}$ or $X_{x,i}$ is 1:

$$\forall \tau_x : \forall \tau_i, \tau_x \neq \tau_i : \forall p, 1 \leq p \leq n-1 : X_{i,x} \leq \sum_{j=p+1}^{n} \pi_{x,j} + (1 - \pi_{i,p}) \tag{19}$$

$$\forall \tau_x : \forall \tau_i, \tau_x \neq \tau_i : X_{i,x} \leq 1 - \pi_{i,n} \tag{20}$$

$$\forall \tau_x : \forall \tau_i, \tau_x \neq \tau_i : X_{i,x} + X_{x,i} = 1 \tag{21}$$

Eq. (19) is only effective if τ_i has priority p as otherwise, $(1 - \pi_{i,p})$ will be > 0 so that $X_{i,x}$ can assume 1. Furthermore, if τ_x has not a lower priority, i.e. $\sum_{j=p+1}^{n} \pi_{x,j} = 0$, the right side will evaluate to 0 such $X_{i,x}$ must assume 0. Note that Eq. (20) does the same for the lowest priority task which is not covered by Eq. (19).

Now, we can formulate the response-time bounds. The WCRT of τ_i is bounded by the task's WCET plus any interference as follows:

$$\forall \tau_i : R_i^+ = C_i^+ + \sum_{\tau_x, \tau_x \neq \tau_i} H_{i,x} \cdot C_x^+ \tag{22}$$

Where $H_{i,x} \leq \left\lceil \frac{T_i}{T_x} \right\rceil$ is an integer variable that bounds the number of executions of τ_x that may interfere with τ_i. If τ_x has a smaller period (and deadline), it can only interfere if its offset ϕ_x is smaller than the completion time of τ_i, i.e. $R_i^+ + \phi_i$. Note that $H_{i,x}$ shall be 0 if τ_i is assigned a higher priority ($X_{i,x} = 1$) or not mapped to the same processor ($1 - V_{i,x} = 1$).

$$\forall \tau_x, \tau_x \neq \tau_i : \tag{23}$$
$$H_{i,x} \geq -\frac{T_i}{T_x}(1 - V_{i,x} + X_{i,x}) + \begin{cases} \frac{R_i + \phi_i - \phi_x}{T_x} & \text{if } T_i = nT_j \\ \frac{R_i + J_j}{T_x} & \text{else} \end{cases}$$

The response-time constraint (implicit deadline) is easily formulated by:

$$\forall \tau_i : R_i + \phi_i + J_i \leq T_i \tag{24}$$

B. Data-age analysis

First, we introduce variables for $w_i^{+/-}$ and $r_i^{+/-}$ and encode the previously derived bounds for $w_i^{+/-}(0)$ and $r_i^{+/-}(0)$ (cf. Eq. (4)-(11)) into the following constraints by replacing ϕ_i^- and ϕ_i^+ with $\phi_i \pm J_i$:

$$\forall \tau_i : \quad r_i^- = \phi_i - J_i \tag{25}$$
$$\forall \tau_i : \quad r_i^+ = w_i^+ - R_i^- \tag{26}$$
$$\forall \tau_i : \quad w_i^- = \phi_i - J_i + R_i^- \tag{27}$$
$$\forall \tau_i : \quad w_i^+ = \phi_i + J_i + R_i^+ \tag{28}$$

Note that R_i^- is provided by the task's BCET, i.e. C_i^-, which is constant. Furthermore, we can calculate the $w_i^{+/-}(n)$/$r_i^{+/-}(n)$ for $n > 0$ by adding the term nT_i.

The data age of a cause-effect chain $\Psi_j = (\tau_1, \ldots, \tau_k)$ is captured in the integer variable $DA_j \geq 0$ and calculated by the following constraint as the sum of all response times and data-age delays plus the worst-case activation offset:

$$\forall \Psi_j : DA_j = \phi_1 + J_1 + \sum_{i=1}^{k-1} (R_i^+ + \check{d}_{i,i+1}) \tag{29}$$

Here, the data-age delays are provided by the integer variables $\check{d}_{i,i+1}^+ \geq 0$. In order to constrain these delays, we need to encode the equations (12)-(14) presented in Section V-B into linear constraints. For this purpose, we use binary decision variables by which we can mask the conditional cases in (12) and (14).

For the oversampling case, we introduce the binary variable $O_{j,i,n}$ which is forced to 0 iff the write time of the 0-th job of the i-th task in Ψ_j can be later than the read time of the n-th job of the reader (i.e. $w_i^+(0) > r_{i+i}^-(n)$). Note that we encode this for every chain Ψ_j, for every writer in this chain ($1 \leq i \leq k-1$), only if the reader has a shorter period ($T_{i+1} < T_i$). In this case, the following constraint must be satisfied for every n:

$$\forall \Psi_j : \forall 1 \leq i \leq k-1 : T_{i+1} < T_i : \forall 0 \leq n \leq \frac{T_i}{T_{i+1}} - 1 :$$
$$T_i \cdot (1 - O_{j,i,n}) \geq w_i^+ - (r_{i+1}^- + nT_{i+1}) \tag{30}$$

If the right side of Eq. (30) is negative (or zero), the left side always holds. The right side becomes positive if $w_i^+(0) > r_i^-(n)$ and is bounded from above by T_i. In this case, $O_{j,i,n}$ must be 0 to satisfy the constraint.

Now, we can encode Eq. (12) into two linear constraints that represent the two cases:

$$\forall \Psi_j : \forall 1 \leq i \leq k-1 : T_{i+1} < T_i : \forall 0 \leq n \leq \frac{T_i}{T_{i+1}} - 1 :$$
$$\check{d}_{i,i+1}^+ \geq r_{i+1}^+ + nT_{i+1} - w_i^- \tag{31}$$
$$\check{d}_{i,i+1}^+ \geq r_{i+1}^+ + nT_{i+1} - (w_i^- - T_i) - 2O_{j,i,n}T_i \tag{32}$$

Note that Eq. (32) corresponds to the conditional case in Eq. (12), i.e. $w_i^+(0) > r_{i+1}^-(n)$, in which case $O_{j,i,n} = 0$. Note that the constraint is not effective once the right side becomes negative. This is the case when $O_{j,i,n} = 1$ as $2T_i$ is an upper bound for $r_{i+1}^+(n) - w_i^-(-1)$, which is calculated by the first part of the constraint.

For the undersampling case (including equal periods), we similarly introduce a binary variable, $U_{j,i}$, which is forced to 0 if the write time of the i-th task in Ψ_j can be later than the read time of the reader, i.e. $w_i^+(0) > r_{i+1}^-(0)$:

$$\forall \Psi_j : \forall 1 \leq i \leq k-1 : T_{i+1} \geq T_i :$$
$$T_{i+1} \cdot (1 - U_{j,i}) \geq w_i^+ - r_{i+1}^- \tag{33}$$

Similar to the oversampling case, we encode Eq. (13) and Eq. (14) by the following constraints:

$$\forall \Psi_j : \forall 1 \leq i \leq k-1 : T_{i+1} \geq T_i : \forall 0 \leq n \leq \frac{T_{i+1}}{T_i} - 1 :$$
$$\check{d}_{i,i+1}^+ \geq r_{i+1}^+ - (w_i^- + nT_i) \tag{34}$$
$$\forall \Psi_j : \forall 1 \leq i \leq k-1 : T_{i+1} \geq T_i :$$
$$\check{d}_{i,i+1}^+ \geq r_{i+1}^+ - (w_i^- - T_i) - 2U_{j,i}T_{i+1} \tag{35}$$

For the non-harmonic case, we conservatively bound $\check{d}$ by the maximum distance between two write events, which is calculated by its period plus response-time jitter:

$$\check{d}_{i,i+1}^+ \geq T_i + R_i^+ - R_i^- \tag{36}$$

C. Optimisation

In order to minimise the data age, we formulate the objective function as follows. Note that this must also minimise the response-time variables as, otherwise, the values will not be tight, i.e. we are looking for the smallest upper bound. We thus formulate the objective function such that it first minimises the data-age variables (DA_j) and, as a second objective, minimises the response-time variables (R_i). We achieve this by multiplying the data-age variables with the upper bound on the response time (here: T_i) such that no reduction in response time can be traded against an increase in data age:

$$\sum_{\tau_i} \left(R_i^+ + T_i \cdot \sum_{\Psi_j} DA_j \right) \tag{37}$$

VII. Evaluation and experiments

In this section, we evaluate our MILP formulation on the ADAS use case as presented in Section II. In addition, we apply the same approach to a public domain automotive benchmark [14] from Bosch which specifies an engine control system and has also been used for the industrial challenge of the WATERS'16 and WATERS'17 workshop [3].

We implemented the response-time and data-age formulation from Section VI using Zuse Institut Mathematical Programming Language (ZIMPL) [15] in order to generate the particular `.lp` files for the different use cases; the implementation is available online[1]. The `.lp` files can be read by several solvers. In the scope of this paper, we used SCIP[2] with SoPlex[3]. For validation of the response-time and data-age results, we further used pyCPA[4] as reference.

A. ADAS use case

We performed two experiments using the task set from Table I. In the first experiment, we kept the initial task to processor mapping and only assigned priorities and offsets. In the second experiment, we allowed the MILP to also choose the mapping. We name the experiments $\mathbf{A}.\pi\phi$ and $\mathbf{A}.\pi\phi A$ respectively so that the names show what parameters are assigned by the MILP. For all experiments, we used an offset jitter of $J_i = 20\,\mu s$.

The results are summarised in Table II. By choosing optimal offsets for the initial scenario, we achieve a data age of approx. $73\,ms$. The main contributors to this are the rather long WCRT of Task C and a large write-to-read distance between Task H and Task C. We conclude from this that the predetermined task to processor mapping only offers a limited optimisation potential. On the other hand, by changing the mapping in the second experiment, we can reduce the data age to approx. $12\,ms$.

Note that the chains in this use case are representing two slightly different paths in the data dependencies (cf. Figure 1). Moreover, all tasks have equal or longer periods so that only undersampling is considered in this use case. We therefore applied our approach to a second use case in the following section.

B. Engine-control use case

This use case is based on a real-world engine control application that has been made publicly available as an automotive benchmark [14]. The system comprises four cores and 21 tasks which have been specified as an Amalthea model[5] from which we extracted the task set as shown in Table III.

The model specifies two distinct cause-effect chains that represent inter-task communication between harmonic and non-harmonic tasks (i.e. sporadic ISRs) with oversampling and undersampling as illustrated in Figure 5.

[1] https://www.ida.ing.tu-bs.de/pub2018/schlatow2018dataage.zip
[2] http://scip.zib.de/
[3] http://soplex.zib.de/
[4] https://bitbucket.org/pycpa
[5] http://www.amalthea-project.org

Table II
$\mathbf{A}.\pi\phi$ and $\mathbf{A}.\pi\phi A$ results (ϕ and WCRT in μs)

	$\mathbf{A}.\pi\phi$			$\mathbf{A}.\pi\phi A$			
Task	π	ϕ	WCRT	π	ϕ	A	WCRT
ISR	3	n/a	20	5	n/a	1	20
Task A	11	20	770	9	30	1	770
Task B	9	1100	5950	4	7990	2	500
Task C	10	3860	27190	6	3860	2	2600
Task D	5	50	180	8	50	1	180
Task E	8	1100	4450	2	8990	2	750
Task G	4	6620	200	1	9780	2	200
Task H	7	20	3800	10	20	2	3800
Task I	2	9370	110	5	7880	2	110
Task J	6	4120	2500	12	20	1	9490
Task K	1	9480	500	3	8490	2	500
Task L	12	20	4100	11	20	1	1750
Chain	Data Age			Data Age			
Chain 1			72655 μs				11885 μs
Chain 2			73093 μs				12323 μs

Table III
Task set **E** (engine control)

Core	Task	Period	WCET	BCET
0	ISR_9	6000 μs	260 μs	124 μs
0	ISR_8	1700 μs	213 μs	91 μs
0	ISR_7	5007 μs	228 μs	121 μs
0	ISR_6	1100 μs	22 μs	10 μs
0	ISR_5	900 μs	181 μs	90 μs
0	ISR_4	1504 μs	257 μs	116 μs
0	ISR_10	700 μs	22 μs	11 μs
0	ISR_11	5011 μs	215 μs	96 μs
1	Angle_Sync	6660 μs	2664 μs	913 μs
1	Task_1ms	1 ms	536 μs	175 μs
2	Task_200ms	200 ms	97 μs	49 μs
2	Task_20ms	20 ms	7328 μs	2523 μs
2	Task_50ms	50 ms	2160 μs	919 μs
2	Task_5ms	5 ms	653 μs	255 μs
2	Task_2ms	2 ms	283 μs	97 μs
2	Task_100ms	100 ms	6593 μs	2188 μs
2	Task_1000ms	1000 ms	96 μs	47 μs
3	Task_10ms	10 ms	8199 μs	2792 μs
3	ISR_2	9502 μs	13 μs	7 μs
3	ISR_1	9501 μs	25 μs	10 μs
3	ISR_3	9503 μs	17 μs	8 μs

Similar to the previous use case, we perform two experiments: one with predetermined mapping ($\mathbf{E}.\pi\phi$) and one with optimised mapping ($\mathbf{E}.\pi\phi A$).

The results are summarised in Table IV. Although it takes significantly more processing time to find the optimal solution (cf. Section VII-C), a near-optimal solution is often found after a short time as explained in the next section.

C. Statistics and summary

Table V shows the statistics for our experiments. More precisely, it shows the number of tasks, cores, chains, and the system load (utilisation) for our experiments. Note that a system load of 1.26 indicates that the (multi-core) system is loaded with 126 % of the maximum single-core CPU utilisation. Additionally, we listed the number of variables and constraints (after presolving, i.e. after the solver eliminated

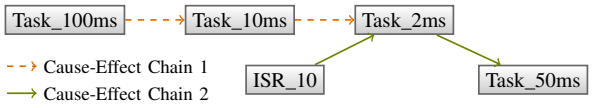

Figure 5. Task data dependencies in engine-control use case

Table IV
E.$\pi\phi$ AND **E**.$\pi\phi A$ RESULTS (ϕ AND WCRT IN μs)

Task	**E**.$\pi\phi$ π	ϕ	WCRT	**E**.$\pi\phi A$ π	ϕ	A	WCRT
ISR_9	20	n/a	1395	12	n/a	0	1179
ISR_8	14	n/a	438	19	n/a	2	1090
ISR_7	21	n/a	2340	13	n/a	0	2824
ISR_6	2	n/a	44	3	n/a	0	57
ISR_5	10	n/a	225	17	n/a	2	481
ISR_4	19	n/a	1135	11	n/a	0	897
ISR_10	1	n/a	22	1	n/a	0	22
ISR_11	18	n/a	653	18	n/a	2	696
Angle_Sync	9	n/a	5880	20	n/a	2	6543
Task_1ms	3	20	536	9	356	0	618
Task_200ms	17	20	21999	8	116	1	97
Task_20ms	11	2269	13428	6	10492	1	9488
Task_50ms	8	20	2443	5	582	1	2160
Task_5ms	5	4044	936	21	20	0	4336
Task_2ms	4	1697	283	14	259	2	283
Task_100ms	12	174	21523	7	1227	1	8753
Task_1000ms	16	20	21619	10	20	1	96
Task_10ms	6	1737	8199	15	20	3	8199
ISR_2	7	n/a	8212	2	n/a	0	35
ISR_1	15	n/a	8254	4	n/a	0	82
ISR_3	13	n/a	8229	16	n/a	2	300
Chain	Data Age			Data Age			
Chain 1	160415 μs			134207 μs			
Chain 2	5249 μs			4683 μs			

superfluous constraints and variables). Note that the engine-control use case (**E**) has about three times as many variables and about five times as many constraints as the ADAS use case (**A**). Furthermore, we recorded the number of solutions and the time it took the solver to find a first solution (feasibility) and to determine the optimal among all solutions (optimality). *Feasibility* refers to finding a first solution that meets all task deadlines irrespective of the resulting data age.

As expected for MILPs, the first solution can be determined in a few seconds whereas solving for optimality may take a long time (up to 377 s for **A** and more than 280 h for **E**). However, as the primal-dual algorithm [16] implemented by state-of-the-art solvers always bounds the gap to the optimum so that it can be interrupted once this gap gets small enough. For the **E** use case, we found the first near-optimal solution (0% gap) after approx. 15min[6]. In our case, we aborted the **E** use case after it already reached a gap <0.01 % for several hours.

In order to further evaluate the robustness of the MILP with respect to variations in the input parameters, we randomised the WCETs of **A** and evaluated the solving time and gap for ten variations. Figure 6 depicts the results, the legend shows

[6]on an Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz

Table V
EXPERIMENT STATISTICS

	A.$\pi\phi$	**A**.$\pi\phi A$	**E**.$\pi\phi$	**E**.$\pi\phi A$
#tasks	12	12	21	21
#cores	2	2	4	4
#chains	2	2	2	2
system load	1.26	1.26	3.19	3.19
#variables	325	490	1020	1511
#constraints	1425	2027	8579	10991
#solutions	7	118	≥189	≥12788
solving time[6] (feasibility)	14 s	17 s	0.8 s	431 s
solving time[6] (optimality)	29 s	377 s	>7647 m	>280h

the system load of the corresponding task set. Each line starts at the time where the solver found the first solution and ends at the optimum solution (i.e. 0 % gap) and hence shows how fast the solver was able to close the gap to the optimum. In summary, the first solution was found after 16 s to 44 s whereas the optimum was reached after 54 s to 503 s. Although there is a tendency for the MILP to take longer for systems with higher load, the orange line represents a counterexample. For this system as it took 272 s to find the optimum despite a system load of only 1.04.

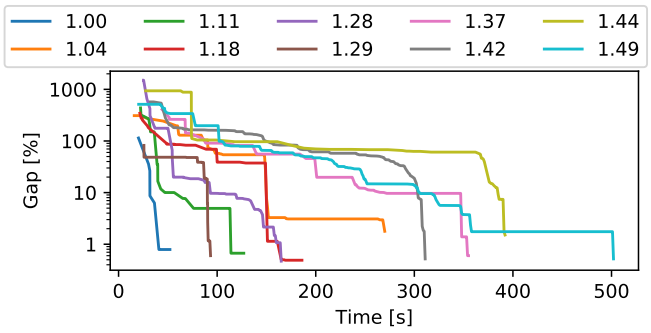

Figure 6. Solving progress for ten randomised variations of **A**.$\pi\phi A$ with different system load.

VIII. CONCLUSION

Motivated by a real-world automotive use case we identified the problem of bounding data-age latencies for cause-effect chains. Note that, although we focused on data age in this paper, our method can be analogously applied to reaction time as well as it only changes the latency semantics. We presented a data-age analysis for periodic offset-synchronised tasks that takes scheduling priorities and task-to-processor mapping into account. Based on this, we formulated MILP constraints that cover the response-time and data-age analysis for this kind of systems, which allows us to find and select the design parameters (priorities, offsets, mapping) such that the data age is minimised. Due to the exponential complexity of MILPs, however, we need to clarify the main question of its applicability, i.e. whether typical problem instances can be solved in limited time. For this purpose, we performed an

experimental evaluation of our use case as well as of a similar publicly available benchmark. Our evaluation demonstrated that feasibility is rather easy and fast, i.e. a good solution is found in short time, however, optimality is significantly harder and takes much more computing time. Given that in most realistic cases it is sufficient to fall below a certain data age threshold, feasibility is typically more important than optimality. We therefore believe that the MILP approach is applicable to real scenarios either as a design methodology or as a method to estimate how close the current solution is to a global optimum. On the other hand, there exist a great body of research regarding techniques to improve MILP efficiency such as approximations in the analysis that eliminate integer variables [6].

ACKNOWLEDGEMENT

This work was partially funded within the DFG Research Unit *CCC*, funding number FOR 1800, and partially by Hitachi Ltd. We further thank Thorsten Koch for providing ZIMPL [15].

REFERENCES

[1] AUTOSAR, "Specification of Timing Extensions, Release 4.3," http://www.autosar.org/, Nov. 2016, release 4.3.

[2] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson, "A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics," *Work. on Compositional Theory and Technology for Real-Time Embedded Systems CRTS*, 2008.

[3] A. Hamann, D. Dasari, S. Kramer, M. Pressler, F. Wurst, and D. Ziegenbein, "Waters industrial challenge 2017," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, Dubrovnik, Croatia, jun 2017.

[4] K.-B. Gemlau, J. Schlatow, M. Möstl, and R. Ernst, "Compositional analysis for the waters industrial challenge 2017," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, Dubrovnik, Croatia, jun 2017.

[5] J. Martinez, I. Sanudo, P. Burgio, and M. Bertogna, "End-to-end latency characterization of implicit and let communication models," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, Dubrovnik, Croatia, jun 2017.

[6] A. Biondi, P. Pazzaglia, A. Balsini, and M. D. Natale, "Logical execution time implementation and memory optimization issues in autosar applications for multicores," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, Dubrovnik, Croatia, jun 2017.

[7] F. Boniol, J. Forget, and C. Pagetti, "Waters industrial challenge 2017 with prelude," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, Dubrovnik, Croatia, jun 2017.

[8] J. M. Rivas, J. J. Gutierrez, J. L. Medina, and M. G. Harbour, "Comparison of memory access strategies in multi-core platforms using mast," in *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, Dubrovnik, Croatia, jun 2017.

[9] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte, "End-to-end timing analysis of cause-effect chains in automotive embedded systems," *Journal of Systems Architecture*, vol. 80, no. Supplement C, pp. 104 – 113, 2017.

[10] R. I. Davis, L. Cucu-Grosjean, M. Bertogna, and A. Burns, "A review of priority assignment in real-time systems," *Journal of Systems Architecture*, vol. 65, no. Supplement C, pp. 64 – 82, 2016.

[11] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Comput. Surv.*, vol. 43, no. 4, pp. 35:1–35:44, Oct. 2011.

[12] A. Wieder and B. Brandenburg, "Efficient partitioning of sporadic real-time tasks with shared resources and spin locks," in *Proceedings of the 8th IEEE International Symposium on Industrial Embedded Systems (SIES 2013)*, Jun. 2013, pp. 49–58.

[13] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocess. Microprogram.*, vol. 40, no. 2-3, pp. 117–134, Apr. 1994. [Online]. Available: http://dx.doi.org/10.1016/0165-6074(94)90080-9

[14] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," in *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2015.

[15] T. Koch, "Rapid mathematical prototyping," Ph.D. dissertation, Technische Universität Berlin, 2004.

[16] G. B. Dantzig, L. R. Ford, and D. R. Fulkerson, "A primal-dual algorithm for linear programs," *Linear inequalities and related systems*, vol. 38, pp. 171–181, 1956.