

Placement of Chains of Real-Time Tasks on Heterogeneous Platforms under EDF Scheduling

Daniel Casini^{*†} and Alessandro Biondi^{*†}

^{*}TeCIP Institute, Scuola Superiore Sant'Anna, Pisa, Italy

[†]Department of Excellence in Robotics & AI, Scuola Superiore Sant'Anna, Pisa, Italy

Abstract—When designing a real-time system, application architects are called to settle many non-trivial decisions that may severely influence the system's performance. With modern hardware platforms always being more and more complex and equipped with heterogeneous processor cores or even hardware accelerators such as TPUs, FPGAs, or GPUs, the complexities to be faced by application architects are exacerbated. Therefore, they are called to wisely allocate the computational resources provided by the hardware platform to application tasks in such a way to meet timing requirements and optimize other goals such as energy consumption. This paper proposes a mixed-integer linear programming formulation (MILP) to solve the task-to-heterogeneous-cores allocation problem while guaranteeing the schedulability of a real-time application running on the platform under partitioned Earliest Deadline First (EDF) scheduling. A new method to derive approximate worst-case response-time bounds is also presented and leveraged to setup the MILP formulation, which allows computing and minimizing the end-to-end latency of processing chains and considers energy requirements. The approach is evaluated on a task set based on the WATERS 2019 Industrial Challenge proposed by Bosch.

Index Terms—Task Allocation, Real-Time Systems, Schedulability, EDF, Heterogeneous Platforms

I. INTRODUCTION

A vast increase in computation requirements has characterized recent years. Indeed, it is always more common to run largely computational expensive applications even on resource-constrained platforms such as embedded systems. This is the case, for example, of autonomous driving and advanced driving assistance systems (ADAS) [1, 2] or workloads generated by applications leveraging artificial intelligence [3]–[5]. In these applications, computational activities (tasks) are often required to complete within well-defined deadlines to avoid performance degradation or possibly even catastrophic consequences such as the loss of human lives.

Hardware platforms have been able to cope with such a huge computation demand by increasing their own complexity. Indeed, an always more viable and preferred choice is the usage of heterogeneous processing platforms equipped with heterogeneous cores [6] running at different speeds, offering different instruction set architectures, and being characterized by different energy consumption profiles. In some cases, such platforms also include hardware accelerators such as TPUs, GPUs, or FPGAs [7].

As a result, application designers of real-time applications are called to always make more complex decisions. Among them, a common design question is about how to properly

allocate tasks to the computational resources provided by the hardware platform.

For example, when having cores with different speed profiles, which tasks to allocate on faster cores and which on slower ones to guarantee all tasks meet their deadlines? Or, when disposing of a hardware accelerator and of tasks with multiple implementations (e.g., on a CPU or a GPU), which ones to run on the core and which on the accelerator?

These choices gain particular importance in applications where tasks communicate among themselves in a producer-consumer fashion, forming *processing chains* of communicating tasks, which are common in automotive [8, 9] and robotics applications [1, 10]. Chains might span from the sensing of a new data sample to the corresponding actuation, traversing multiple software components, possibly distributed over multiple hosts.

Bounding the end-to-end latency of such chains is crucial to ensure the temporally correct behavior of modern cyber-physical systems. However, different resource allocation choices may lead to significantly different end-to-end latencies for chains.

Contribution. To this end, this paper proposes to adopt an optimization-based approach in the form of a mixed-integer linear programming formulation (MILP) to determine the best task-to-core allocation in platforms with heterogeneous processing cores.

The optimization problem leverages real-time analysis techniques for Earliest Deadline First (EDF) scheduling [11] (an optimal scheduling algorithm on multiprocessor platforms using partitioned scheduling) and allows guaranteeing timing and energy constraints while bounding the end-to-end latency of processing chains.

However, directly computing accurate response-time bounds under EDF is not trivial and difficult to be encoded in an MILP formulation, which mandates linear constraints.

To cope with this issue, this paper adopts an approach based on demand-bounds functions [12], which are more amenable to being encoded in linear form [13, 14]. This allows guaranteeing timing constraints, but it is not enough to bound the end-to-end latency of processing chains, which requires a bound on the worst-case response time of each task involved in the chain. We therefore use insights about the relation between demand bound functions and response times to propose a novel method to compute approximate response-time bounds under

EDF, which in turn enables the computation of worst-case response-time bounds for end-to-end latencies in task chains.

We evaluate our proposal on a task set based on the WATERS 2019 Challenge provided by Bosch, showing that our modeling is capable of optimizing a representative ADAS application on the Jetson TX-2 heterogeneous platform.

The contribution of this paper builds the roots for a design-exploration tool to tackle the placement of real-time applications on a heterogeneous platform, providing significant advantages to application designers that would be otherwise forced to deploy different configurations and empirically measure response times and end-to-end latencies, ending up in a very time-consuming process with no guarantee of optimality.

Paper Structure. The remainder of this paper is organized as follows. Section II introduces the system model. Section III presents the real-time analysis theory required to encode the heterogeneous task-to-core optimization problem under EDF. Section IV presents the optimization problem. Section VI discusses the related work. Finally, Section VII concludes the paper.

II. SYSTEM MODEL

Platform and Workload Model. We consider a heterogeneous embedded platform with a set of m cores. Each core $p_k \in \mathcal{P}$ is characterized by a type, which determines the speed and the energy consumption of the core.

The workload is composed of a set of n real-time tasks $\Gamma = \{\tau_1, \dots, \tau_n\}$. Each task released a potentially-infinite sequence of instances according to a period T_i . Each instance, also called *job*, needs to complete within D_i time units from its release, where D_i is the constrained deadline $D_i \leq T_i$ of the task. A task set Γ is said to be *schedulable* if, for each task, all its instances can complete within the deadline.

Each task τ_i is characterized by a core-dependent worst-case execution time (WCET). The symbol C_i^k denotes the WCET of τ_i when it is assigned to core p_k .

Each task is also characterized by a per-job worst-case energy consumption e_i^k , which depends on the heterogeneous core p_k to which it is assigned. It is required that the overall task set satisfies a maximum-energy requirement in a given operational time O , i.e., that the sum of the energy consumptions of all jobs of all tasks in an arbitrary interval of time with length O is less than or equal to a threshold E . For example, this can be used to model an energy constraint for a battery-powered device, where the total energy needs to be enough to guarantee the functional behavior for a given interval of time given at design time.

The utilization U_i^k of a task τ_i on a core p_k is defined as the ratio of the WCET and the period, i.e., $U_i^k = \frac{C_i^k}{T_i}$.

Scheduling. Tasks are assigned according to the *partitioned scheduling* paradigm, i.e., each task $\tau_i \in \Gamma$ is statically assigned to only one core. This scheduling strategy showed to be particularly beneficial in fostering predictability [15, 16]. The set of tasks allocated to a core p_k is denoted with Γ_k . Each core is scheduled according to the Earliest-Deadline First scheduling algorithm [11].

Task Chains. Tasks are also characterized by functional dependencies. A *processing chain* ω_x is an ordered sequence of tasks characterized by a producer-consumer relationship, where each task τ_i may belong to multiple sequences. The set of all the chains is denoted with Ω . As for individual tasks, processing chains are characterized by a chain *end-to-end deadline* D_x^ω . In this paper, we consider time-triggered chains [17], i.e., there is no data-driven activation between tasks in the chains, which are instead released periodically.

Notation. The most important symbols used throughout the paper are summarized in Table I.

TABLE I
TABLE OF SYMBOLS.

Symbol	Description
m	number of cores
\mathcal{P}	the set of all processors
p_k	a core in set \mathcal{P}
Γ	the task set of all real-time tasks
Γ_k	the task set of real-time tasks on p_k
τ_i	a real-time task
T_i	period of τ_i
D_i	deadline of τ_i
C_i^k	WCET of τ_i when on p_k
U_i^k	utilization U_i^k of τ_i
S_i	slack of τ_i
R_i	worst-case response-time bound of τ_i
e_i^k	energy consumption of a job of τ_i on p_k
Ω	the set of all processing chains
ω_x	a processing chain
D_x^ω	the deadline of processing chain ω_x
L_x	end-to-end latency of processing chain ω_x

III. ANALYSIS

Before presenting the optimization problem, this section presents the real-time analysis theory techniques needed to guarantee timing constraints under partitioned EDF, generalizing the notation to platforms with heterogeneous cores based on the system model notation. In this section, we focus on those mechanisms that are used to encode the task-to-core placement problem.

A. Schedulability Analysis

Under partitioned scheduling, the schedulability of the overall task set Γ is determined by checking if all tasks set Γ_k of cores $p_k \in \mathcal{P}$ are schedulable.

For each core $p_k \in \mathcal{P}$, the schedulability under partitioned EDF can be checked either using the *response-time analysis* [18]–[20] or using the processor-demand criterion [12]. Response-time analysis (RTA) techniques are based on upper-bounding the worst-case response time (WCRT) R_i of each task, i.e., the maximum time span elapsed from the release to the completion of any of its instances. Under EDF, RTA techniques are rather complicated: for example, the one in [20]

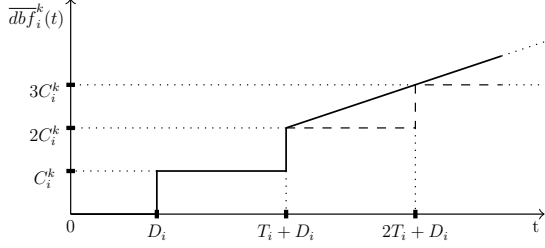


Fig. 1. Graphical representation of function $\overline{dbf}_i^k(t)$.

has $\mathcal{O}(n \cdot T_{max} \cdot L^2)$ complexity [21], where n is the number of tasks, T_{max} is the maximal period of all tasks, and L^2 is the maximal busy period length [20]. Furthermore, the response-time analysis requires solving a recursive equation, making it hard to be implemented in an optimization problem in the case of EDF¹.

Alternatively, the schedulability of a task set Γ_k of periodic, constrained-deadline, real-time tasks for an arbitrary core $p_k \in \mathcal{P}$ can be determined based on the *processor-demand criterion* (PDC) proposed by Baruah et al. [12]. As discussed later in this section, it can be approximated more easily for being implemented in an optimization problem.

The processor demand criterion builds upon the definition, for each task $\tau_i \in \Gamma$, of a demand bound function defined as follows:

$$dbf_i^k(t) = \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \cdot C_i^k. \quad (1)$$

The PDC is recalled below.

Theorem 1 (Processor Demand Criterion [12]): A task set Γ_k is deemed schedulable on a core p_k if and only if

$$\forall t \in \mathcal{D} = \cup_{\tau_i \in \Gamma_k} \{t = D_i + fT_i : t \leq L^*, f \in \mathcal{N}_{\geq 0}\}, \quad (2)$$

$$\sum_{\tau_i \in \Gamma_k} dbf_i^k(t) \leq t,$$

where bounds for L^* are available in [12, 24].

The PDC has pseudo-polynomial time complexity when the utilization is strictly less than one. Furthermore, it is not linear due to the presence of the floor operator in Eq. (1). Thus, the PDC is not a good fit for being directly implemented in a mixed-integer linear programming formulation.

Therefore, we use an approximate definition of the demand bound function and a sufficient schedulability test. The approximation has been defined as follows [14]:

$$\overline{dbf}_i^k(t) = \begin{cases} \nu_i \cdot C_i^k & \text{if } t < \nu_i T_i + D_i \\ C_i^k + U_i^k(t - D_i) & \text{otherwise.} \end{cases} \quad (3)$$

The constant parameter $\nu_i \geq 0$ allows tuning the trade-off between analysis precision and runtime required to solve the problem by regulating the number of steps of the original DBF to maintain the approximation.

¹Efficient techniques have been proposed but only for the case of fixed-priority scheduling [22, 23].

Next, we recall the sufficient schedulability test for EDF that uses $\overline{dbf}_i^k(t)$.

Theorem 2 ([14]): A task set Γ_k is deemed schedulable on a core p_k if $\sum_{\tau_i \in \Gamma_k} U_i \leq 1$ and

$$\forall t \in \bigcup_{\tau_i \in \Gamma_k} \xi(\tau_i), \quad \sum_{\tau_i \in \Gamma_k} \overline{dbf}_i^k(t) \leq t \quad (4)$$

where

$$\xi(\tau_i) = \{sT_i + D_i\}, \quad s = 0, \dots, \nu_i. \quad (5)$$

Theorem 2 is more amenable to implementing processor demand analysis in a mixed-integer optimization problem.

First, sets $\xi(\tau_i)$ contains a linear number of points for each task. Second, $\overline{dbf}_i^k(t)$ it is a linear formulation: indeed, it is either a constant function equal to $\nu_i \cdot C_i^k$ if $t < \nu_i T_i + D_i$, or a linear function otherwise.

B. End-to-End Latency Analysis

Following the results by Davare et al. [17], the end-to-end latency L_x of a chain $\omega_x \in \Omega$ of time-driven communicating tasks can be bounded as:

$$L_x = \sum_{\tau_i \in \omega_x} (R_i + T_i) - T_f, \quad (6)$$

under the assumption that the chain is triggered synchronously with the release of the first task τ_f of the chain.

In essence, Eq. (6) assumes that each task completes an input operation just before new data becomes available, thus requiring to wait for the next job for sample updated data [17].

However, this approach requires bounding the worst-case response-time R_i of each task in the chain. As discussed in Section III-A, WCRT bounds for EDF are hard to encode in an optimization problem. Therefore, this paper uses an approach based on approximate demand-bound functions to assess schedulability. However, there is a link between the two analysis methodologies, which is discussed next.

C. From dbf analysis to response times

The link between these two analysis techniques comes from [21]. The authors of [21] proposed to compute the WCRT R_i by means of the *worst-case slack time* of each task τ_i , i.e., the length S_i of the shortest interval from a job's completion until its absolute deadline.

If S_i is known, then the WCRT can be computed as:

$$R_i = D_i - S_i. \quad (7)$$

For each task τ_i , S_i can be lower-bounded as follows.

Theorem 3 ([21]): It holds:

$$S_i \geq \min_{\forall t: D_i \leq t \leq \mathcal{L}} \left(t - \overline{sbf}_k \left(\sum_{\tau_i \in \Gamma_k} dbf_i^k(t) \right) \right), \quad (8)$$

where \mathcal{L} is a bound on the analysis interval [21] and $\overline{sbf}(x)$ denotes the minimum interval length to provide an amount x of computation capacity on the core p_k .

The formulation in [21] differs from the one required in this paper in several parts. First, it uses a general function to model the supply time provided by a core p_k , while this paper assumes that all the computing capacity of the core is dedicated to real-time tasks. It follows that $\overline{sb}f_k(x) = x$.

Therefore, the expression $\overline{sb}f_k(\sum_{\tau_i \in \Gamma_k} dbf_i(t))$ in Eq. (8) can be simplified as $\sum_{\tau_i \in \Gamma_k} dbf_i(t)$.

Second, Theorem 3 uses the complete formulation of the demand bound function and tests the condition in all the points in the interval $[D_i, L^*]$. We show next in Theorem 4 that a similar result can also be obtained when using the approximation scheme of Theorem 2.

Theorem 4: Consider a core $p_k \in \mathcal{P}$. Let $\mathcal{X} = \bigcup_{\tau_j \in \Gamma_k} \bar{\xi}(\tau_j, D_i)$, $\xi(\tau_i)$ is defined as in Eq. (5), and

$$\bar{\xi}(\tau_j, D_i) = \{t : t \in \xi(\tau_j) \wedge t \geq D_i\}. \quad (9)$$

If $\sum_{\tau_i \in \Gamma_k} U_i^k \leq 1$, it holds:

$$S_i \geq \min_{\forall t \in \mathcal{X}} \left(t - \sum_{\tau_i \in \Gamma_k} \overline{db}f_i^k(t) \right). \quad (10)$$

Proof.

Consider an arbitrary time instant $t \in \mathcal{X}$. By construction, $\overline{db}f_i^k(t, T_i) \geq dbf_i^k(t, T_i)$, hence if Eq. (8) is satisfied for a given t , also Eq. (10) is satisfied. Next, it remains to show that it is sufficient to check points in the set \mathcal{X} . Assume there exist a point $t \in [D_i, \mathcal{L}] \setminus \mathcal{X}$ (i.e., considered in Theorem 3 but not in this theorem) such that $S_i < \min_{\forall t \in \mathcal{X}} \left(t - \sum_{\tau_i \in \Gamma_k} \overline{db}f_i^k(t) \right)$.

Note that functions $dbf_i^k(t)$ (used in Theorem 3) are step-wise functions that change value only in correspondence of their discontinuities. Since \mathcal{X} includes all the discontinuities of functions $\overline{db}f_i^k(t)$ greater than or equal to D_i , if a discontinuity of $dbf_i^k(t)$ in $[D_i, \mathcal{L}]$ that is not in \mathcal{X} brings to a smaller value of S_i by Theorem 3 it means that for such a discontinuity t^* of $dbf_i^k(t)$ holds $t^* - \sum_{\tau_i \in \Gamma_k} \overline{db}f_i^k(t^*) > t^* - \sum_{\tau_i \in \Gamma_k} dbf_i^k(t^*)$ but this is impossible because by definition because $\overline{db}f_i^k(t) \geq dbf_i^k(t)$. \square

D. Energy Consumption

Battery-constrained systems, such as many embedded real-time systems, need to remain operational for a given time. This reflects in a constraint on the energy consumption of the tasks, which is influenced by the task-to-core placement and by tasks' periods. In particular, we need to enforce that the overall amount of energy required in the operation time O is less than the threshold E . Given the per-job worst-case energy consumption e_i^k , we need to know how many jobs of each task τ_i may be released in an arbitrary interval of length O . Such a number of jobs is bounded by $\left\lfloor \frac{O}{T_i} \right\rfloor$ jobs. Hence, the overall energy consumption of a task τ_i in the operational time is

computed as $e_i^k(O) = e_i^k \cdot \left\lfloor \frac{O}{T_i} \right\rfloor$. Therefore, the system needs to guarantee that:

$$\sum_{p_k \in \mathcal{P}} \sum_{\tau_i \in \Gamma_k} e_i^k(O) \leq E. \quad (11)$$

Note this constraint only serves the purpose of proving the extensibility of our MILP formulation, while we leave a finer-grained modeling of energy consumption as future work.

IV. OPTIMIZATION

This section presents the optimization problem. We start presenting the variables in Section IV-A. Then, Section IV-B discusses the constraints, providing a corresponding proof for the more complex ones. In some constraints, we use the so called *big-M* formulation, denoting with M a large constant representing infinity. Finally, Section IV-C discusses two different objective functions.

A. Variables

- *Task-to-core assignment:* For each task $\tau_i \in \Gamma$, and for each core $p_k \in \mathcal{P}$, $\text{CT}_{i,k} \in \{0, 1\}$ is a binary variable set to 1 if τ_i is allocated to p_k ; 0 otherwise.
- *Tasks assigned to the same core p_k :* For each core $p_k \in \mathcal{P}$, for each task $\tau_i \in \Gamma$, and for each task $\tau_j \in \Gamma$, $\text{SP}_{i,j,k} \in \{0, 1\}$ is equal to 1 if the two tasks are allocated in the same core p_k . Otherwise, it is equal to 0.
- *Slack candidate of a task:* For each $\tau_i \in \Gamma$, for each $\tau_j \in \Gamma$, for each $t_{j,g} \in \bar{\xi}(\tau_j, D_i)$ (see Eq. (9)), $\text{SLC}_{i,j,g} \in \mathbb{R}^{\geq 0}$ is a candidate slack bound for τ_i .
- *Selector for slack candidate of a task:* For each $\tau_i \in \Gamma$, for each $t_{j,g} \in \bar{\xi}(\tau_j, D_i)$ (see Eq. (9)), $\text{SEL}_{i,j,g} \in \{0, 1\}$ is set to 1 if $t_{i,g}$ is the candidate WCRT bound.
- *Slack of a task:* For each $\tau_i \in \Gamma$, $\text{SL}_i \in \mathbb{R}^{\geq 0}$ is the slack bound of τ_i .
- *Response time of a task:* For each $\tau_i \in \Gamma$, $\text{RT}_i \in \mathbb{R}^{\geq 0}$ is an auxiliary variable for the WCRT bound of τ_i .
- *Latency of a chain:* For each $\omega_y \in \Omega$, $\text{LT}_y \in \mathbb{R}^{\geq 0}$ is a bound on the end-to-end latency of ω_y .

B. Constraints

We start presenting the constraints by enforcing each task to be allocated to only one core.

Constraint 1 (Partitioning): For each $\tau_i \in \Gamma$,

$$\sum_{p_k \in \mathcal{P}} \text{CT}_{i,k} = 1.$$

Constraint 2 is used to guarantee the schedulability of each core using Theorem 2.

Constraint 2 (EDF schedulability): For each $p_k \in \mathcal{P}$, for each $\tau_j \in \Gamma$, $\forall t_{j,g} \in \xi(\tau_j)$ (see Eq. (5)),

$$\sum_{\tau_i \in \Gamma} (\overline{db}f_i^k(t_{j,g}) \cdot \text{CT}_{i,k}) \leq t_{j,g} \quad (12)$$

Proof.

By Theorem 2, a task set Γ_k is deemed schedulable if $\forall t \in$

$\bigcup_{\tau_i \in \Gamma_k} \xi(\tau_i)$, $\sum_{\tau_i \in \Gamma_k} \overline{dbf}_i^k(t) \leq t$. the constraint is checked for each $\tau_j \in \Gamma$, $\forall t_{j,g} \in \xi(\tau_j)$, which includes all the points in the set $\bigcup_{\tau_i \in \Gamma_k} \xi(\tau_i)$. For each $p_k \in \mathcal{P}$, variable $\text{CT}_{i,k}$ is used to sum the demand of τ_i in $t_{j,g}$ only if τ_i is allocated to p_k . \square

The next constraint enforces that the overall energy consumption of the task set in the operational time O is within the energy budget E enforcing Eq. (11).

Constraint 3 (Energy Budget):

$$\sum_{p_k \in \mathcal{P}} \sum_{\tau_i \in \Gamma} e_i^k \cdot \left[\frac{O}{T_i} \right] \cdot \text{CT}_{i,k} \leq E. \quad (13)$$

Next, we define a constraint to enforce the definition of $\text{SP}_{i,j,k}$.

Constraint 4 (Tasks assigned to the same core): For each core $p_k \in \mathcal{P}$, for each task $\tau_i \in \Gamma$, and for each task $\tau_j \in \Gamma$,

$$\text{SP}_{i,j,k} \geq 1 - (2 - \text{CT}_{i,k} - \text{CT}_{j,k}), \quad (14)$$

$$\text{SP}_{i,j,k} \leq \text{CT}_{i,k}, \quad \text{SP}_{i,j,k} \leq \text{CT}_{j,k}. \quad (15)$$

Proof.

This constraint enforces the definition of $\text{SP}_{i,j,k}$ as follows. For an arbitrary core $p_k \in \mathcal{P}$ and two arbitrary tasks τ_i and τ_j , it can be either (i) neither τ_i nor τ_j is on p_k (i.e., $\text{CT}_{i,k} = \text{CT}_{j,k} = 0$), (ii) only one of the two tasks is on p_k (either $\text{CT}_{i,k} = 1$ and $\text{CT}_{j,k} = 0$ or $\text{CT}_{i,k} = 0$ and $\text{CT}_{j,k} = 1$), or (iii) both tasks are allocated to p_k .

Therefore, the constraint needs to enforce $\text{SP}_{i,j,k} = 1$ only in case (iii), and $\text{SP}_{i,j,k} = 0$ otherwise.

Case (i): Eq. (14) enforces $\text{SP}_{i,j,k} \geq -1$ and Eq. (15) enforces $\text{SP}_{i,j,k} \leq 0$. Since $\text{SP}_{i,j,k} \in \{0, 1\}$ it follows $\text{SP}_{i,j,k} = 0$.

Case (ii): Eq. (14) enforces $\text{SP}_{i,j,k} \geq 0$ and Eq. (15) enforces $\text{SP}_{i,j,k} \leq 0$ and $\text{SP}_{i,j,k} \leq 1$. It follows $\text{SP}_{i,j,k} = 0$.

Case (iii): Eq. (14) enforces $\text{SP}_{i,j,k} \geq 1$ and Eq. (15) enforces $\text{SP}_{i,j,k} \leq 1$. It follows $\text{SP}_{i,j,k} = 1$. \square

The following two constraints are used to encode the slack-bound computation according to Theorem 4. Eq. (10) involves a minimization of the quantity $t - \sum_{\tau_i \in \Gamma_k} \overline{dbf}_i^k(t)$ for each $t \in \mathcal{X}$, where \mathcal{X} is defined in Theorem 4. Constraint 5 defines the individual slack candidates of the minimization using variables $\text{SLC}_{i,j,g}$.

Constraint 5 (Slack bound candidate): For each $\tau_i \in \Gamma$, for each $\tau_j \in \Gamma$, $\forall t_{j,g} \in \bar{\xi}(\tau_j, D_i)$

$$\text{SLC}_{i,j,g} = t_{j,g} - \sum_{p_k \in \mathcal{P}} \sum_{\tau_h \in \Gamma} (\overline{dbf}_h^k(t_{j,g}) \cdot \text{SP}_{i,h,k}), \quad (16)$$

where $\bar{\xi}(\tau_j, D_i)$ is defined as in Eq. (9).

Constraint 6 encodes the minimization in Eq. (10).

Constraint 6 (Slack bound): For each $\tau_i \in \Gamma$, for each $\tau_j \in \Gamma$, $\forall t_{j,g} \in \bar{\xi}(\tau_j, D_i)$

$$\text{SL}_i \leq \text{SLC}_{i,j,g}, \quad (17)$$

$$\text{SL}_i \geq \text{SLC}_{i,j,g} - (1 - \text{SEL}_{i,j,g}) \cdot M, \quad (18)$$

with

$$\sum_{\tau_j \in \Gamma} \sum_{t_{j,g} \in \bar{\xi}(\tau_j, D_i)} \text{SEL}_{i,j,g} = 1. \quad (19)$$

Proof.

Eq. (17) enforces SL_i to be smaller than or equal to all candidates $\text{SLC}_{i,j,g}$ defined in Constraint 5. Eq. (18) and Eq. (19) enforces only one candidate to be selected as a slack bound: in particular Eq. (18) forces SL_i to be greater than or equal to *one* slack candidate $\text{SLC}_{i,j,g}$. Due to Eq. (17), the only feasible solution involves selecting the smallest $\text{SLC}_{i,j,g}$. \square

Once the slack is computed using variables SL_i , Constraint 7 bounds the worst-case response time using Eq. (7).

Constraint 7 (WCRT bound): For each task $\tau_i \in \Gamma$,

$$\text{RT}_i = D_i - \text{SL}_i \quad (20)$$

Constraint 8 computes the latency bound for the processing chains by enforcing Eq. (6).

Constraint 8 (Processing chain latency bound): For each processing chain $\omega_y = \{\tau_f, \dots\} \in \Omega$,

$$\text{LT}_y = \sum_{\tau_i \in \omega_x} (\text{RT}_i + T_i) - T_i, \quad (21)$$

Finally, Constraint 9 enforces the end-to-end deadlines for processing chains.

Constraint 9 (End-to-end deadlines): For each processing chain $\omega_y = \{\tau_f, \dots\} \in \Omega$,

$$\text{LT}_y \leq D_y^\omega. \quad (22)$$

C. Objective function(s)

We consider two different objective functions:

- 1) mM-RT: the minimization of the maximum ratio between WCRTs (as computed by variables RT_i) and the corresponding deadline D_i ;
- 2) mM-LT: the minimization of the maximum end-to-end latency of a processing chain;

Note that objective functions requires defining two additional variables RT_{MAX} and LT_{MAX} , and enforcing them to be greater than or equal to all WCRT ratios and chain latencies, respectively.

V. EVALUATION

This section evaluates the proposed approach on the benchmark task-set derived from the one proposed at the WATERS 2019 Challenge by Bosch [25]. This benchmark consists of an ADAS application to be executed on the Jetson TX-2 heterogeneous platform. The Jetson TX-2 is a heterogeneous platform with six cores: four of them are ARMv8 A57 cores running at 1.9 GHz, while two of them are ARMv8 Denver cores running at 2 GHz. The considered task set includes eight tasks: Lidar, Localization, CAN (Controller Area Network), EKF (Extended Kalman Filter), SFM (Structure From Motion), DASM (Drivers Assistance System Module),

TABLE II
TASK SET USED IN THE EVALUATION (MS).

ID	Name	T_i	WCET A57	WCET DENVER
1	Lidar Grabber	33	14,379	10,868
2	DASM	5	1,958	1,3
3	CAN Polling	10	0,632	0,6
4	EKF	15	5,011	4,430
5	Planner	15	13,939	12,437
6	SFM	33	31,055	27,812
7	Localization	400	407,811	294,808
8	Lane Detection	66	53,732	42,238

Planner, and Lane Detection: the task parameters are summarized in Table II, including their periods and the worst-case execution times for each type of core. Time values are in milliseconds. These tasks are characterized by communication relations, represented by arrows in the graph depicted in Fig. 2. Each path of the graph gives rise to a processing chain, for which we bound the worst-case end-to-end latency. The set of all the processing chains is reported in Fig. 3, together with an ID that will be used to identify them in the evaluation results.

Our MILP formulation is used to determine at design time a task-to-core placement allowing all tasks to complete within their deadlines. This involves deciding which tasks to allocate on the faster Denver cores and which ones on the slower A57 ones. Furthermore, the MILP formulation is able to optimize the provided solution according to the two objective functions defined in Section IV-C (i.e., mM-RT and mM-LT) and, thanks to its modular nature, can be easily extended to account for other constraints (e.g., offloading tasks on hardware accelerators) and objective functions in future work.

The MILP formulation has been solved with the C++ version of IBM CPLEX on an Intel Core i7-6700K @ 4.00GHz.

Results. The problem has been solved for the two objective functions, reporting in both cases a very short running time of 0.03 and 0.08 seconds for mM-RT and mM-LT, respectively. This is because our approach that makes use of approximate demand bound functions to indirectly bound the worst-case response time leads to an efficient formulation of the optimization problem with a limited number of variables and constraints, which would have been hard to achieve by directly encoding standard response-time analysis techniques for EDF scheduling, as discussed in Section III-A.

Fig. 4 reports the ratio between the worst-case response times and the deadline of each task for the two objective functions. As expected, the mM-RT objective function allows a reduction from 0.94 (which occurs for mM-LT) to 0.84 of the ratio of the SFM task, which has a WCRT very close to its deadline.

Fig. 5 reports the latencies obtained with the two objective functions, which are also reported numerically in Table III. In Fig. 5, chains have been separated into two different charts according to the magnitude of their latencies to allow easier comparison. In this case, the mM-LT objective function allows achieving a shorter end-to-end latency for chain 4 of 765 ms, which is the longest, while mM-RT reports a latency of 778 ms.

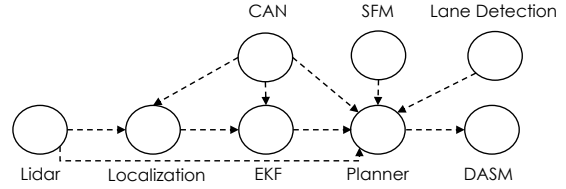


Fig. 2. Communication among tasks in the task set used for the evaluation.

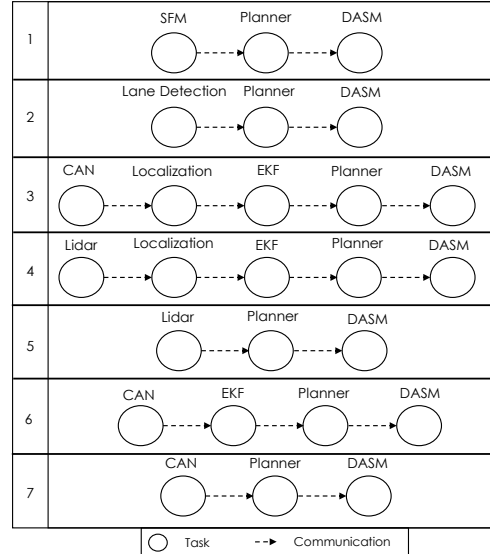


Fig. 3. Complete set of all the processing chains in the task set of Fig. 2

This is due to different task-to-core placements derived by the optimization problem when using the two objective functions.

The placement is reported in Table IV. For example, it can be observed that the solver allocates the SFM task in a (faster) Denver core when using mM-RT, while mM-LT allocates it on a A57 core. This observation explains how the better WCRT ratio of Fig. 4 has been achieved.

Furthermore, Table IV shows that the Localization task is always allocated to a Denver core because it has a WCET greater than the period when allocated to an A57 core.

Overall, the evaluation shows how the proposed approach allows deriving several non-trivial trade-offs at design time, which can significantly help application designers. Indeed, without an analysis-based optimization tool, the only alternative would be to deploy the system under several configurations, test each configuration for an adequate amount of time, collect the data, and compare each configuration, which is clearly a largely time-consuming approach with no guarantee of correctness and optimality. Instead, we answer these questions at design time while guaranteeing an optimized solution by using a MILP formulation.

VI. RELATED WORK

The problem of providing an optimized task-to-core assignment on multicore platforms has been extensively stud-

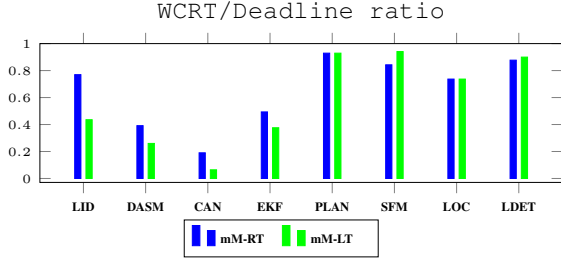


Fig. 4. WCRT/Deadline ratios under the mM-RT and mM-LT objective functions.

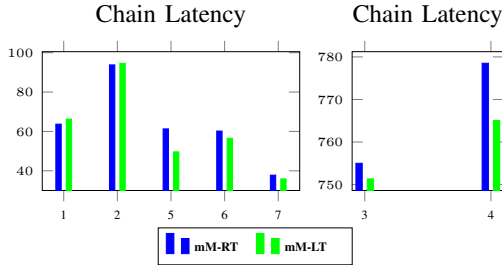


Fig. 5. Chains latencies under the mM-RT and mM-LT objective functions.

ied [26, 27]. More close to us are works considering platforms equipped with heterogeneous processing cores. This problem is known to be NP-hard in the strong sense [28]. Several solutions exist, with many of them using integer linear programming (ILP) techniques [28]–[31]. Other papers proposed heuristics methods to solve this problem, e.g., using ant-colony [32] or particle swarm optimization [33]. None of them considers the optimization of end-to-end latencies of

TABLE III
CHAIN LATENCIES (MS) FOR CHAINS IN FIG. 3.

Chain	Tasks	mM-RT	mM-LT
1	6 - 5 - 2	63,709	66,294
2	8 - 5 - 2	93,800	94,637
3	3 - 7 - 4 - 5 - 2	755,011	751,333
4	1 - 7 - 4 - 5 - 2	778,511	765,069
5	1 - 5 - 2	61,300	49,618
6	3 - 4 - 5 - 2	60,203	56,525
7	3 - 5 - 2	37,800	35,882

TABLE IV
TASK PLACEMENT.

ID	Name	core id and core type	
		mM-RT	mM-LT
1	Lidar Grabber	2 - A57	4 - A57
2	DASM	4 - A57	6 - DENVER
3	CAN Polling	3 - A57	1 - A57
4	EKF	2 - A57	1 - A57
5	Planner	1 - A57	2 - A57
6	SFM	5 - DENVER	3 - A57
7	Localization	6 - DENVER	5 - DENVER
8	Lane Detection	3 - A57	6 - DENVER

processing chains under EDF scheduling.

Some other works targeted the big.LITTLE architecture from ARM [6], e.g., by proposing partitioning heuristics for such platforms [34]–[37]. These works mainly target implicit-deadline task sets where deadlines are equal to task periods and use utilization-based tests to test schedulability. Other works considered the problem of partitioning an application on a heterogeneous platform with a hardware accelerator, considering a fixed-priority scheduling algorithm running on processor cores [38]. Zahaf et al. [39] presented the HPC-DAG model and corresponding schedulability analysis, specifically considering NVIDIA platforms.

VII. CONCLUSIONS

This paper considered the problem of providing real-time software systems application designers with the theoretical foundation for the building tool’s design-space exploration of the task-to-core allocation on heterogeneous applications under EDF scheduling. We considered applications composed of processing chains of communicating tasks, where it is essential to bound their end-to-end latencies to meet timing requirements. We highlighted that bounds for such latencies require knowing the WCRT of each task, which cannot be determined with standard analysis mechanisms based on the demand-function abstraction, which are, however, suitable to be approximated in a linear form and being encoded in an optimization problem. We used insights about the relation of the demand bound function and the worst-case response time to compute it indirectly in the optimization problem, thus making the task allocation aware of the processing chain timing requirements. We evaluated our solution on a task set based on the WATERS 2019 Industrial Challenge proposed by Bosch, showing that our solution can optimize practical applications for ADAS on the Jetson TX-2.

The next steps for future research include extending the proposed formulation to account for tasks leveraging hardware accelerators (e.g., a GPU) under EDF scheduling, for which the real-time analysis literature has only a few recent results [40, 41], the consideration of time-driven processing chains communicating under the Logical Execution Time paradigm [42, 43], data-driven processing chains [10], OS-related delays [44, 45], and reservation-based scheduling paradigms [46, 47].

ACKNOWLEDGMENT

This work has been partially supported by the EU H2020 project AMPERE under the grant agreement no. 871669.

REFERENCES

- [1] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kit-sukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, “Autoware on board: Enabling autonomous vehicles with embedded systems,” in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCP)*. IEEE, 2018, pp. 287–296.
- [2] M. Alcon, H. Tabani, J. Abella, and F. J. Cazorla, “Enabling unit testing of already-integrated ai software systems: The case of apollo for autonomous driving,” in *2021 24th Euromicro Conference on Digital System Design (DSD)*. IEEE, 2021, pp. 426–433.

- [3] L. C. Garaffa, A. Aljuffri, C. Reinbrecht, S. Hamdioui, M. Taouil, and J. Sepulveda, "Revealing the secrets of spiking neural networks: The case of izhikevich neuron," in *2021 24th Euromicro Conference on Digital System Design (DSD)*, 2021, pp. 514–518.
- [4] D. Casini, A. Biondi, and G. Buttazzo, "Timing isolation and improved scheduling of deep neural networks for real-time systems," *Software: Practice and Experience*, vol. 50, no. 9, pp. 1760–1777, 2020.
- [5] F. Restuccia and A. Biondi, "Time-predictable acceleration of deep neural networks on fpga soc platforms," in *2021 IEEE Real-Time Systems Symposium (RTSS)*, 2021, pp. 441–454.
- [6] B. Jeff, "Big. little system architecture from arm: saving power through heterogeneous multiprocessing and task context migration," in *DAC*, 2012, pp. 1143–1146.
- [7] Xilinx, *Zynq UltraScale+ Device - Technical Reference Manual*, 2020.
- [8] F. Rehm, D. Dasari, A. Hamann, M. Pressler, D. Ziegenbein, J. Seitter, I. Sañudo, N. Capodiecì, P. Burgio, and M. Bertogna, "Performance modeling of heterogeneous hw platforms," *Microprocessors and Microsystems*, vol. 87, p. 104336, 2021.
- [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, pp. 104 – 113, 2017.
- [10] T. Blass, D. Casini, S. Bozhko, and B. B. Brandenburg, "A ROS 2 response-time analysis exploiting starvation freedom and execution-time variance," in *2021 IEEE Real-Time Systems Symposium (RTSS)*, 2021, pp. 41–53.
- [11] S. Baruah, A. Mok, and L. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *[1990] Proceedings 11th Real-Time Systems Symposium*, 1990, pp. 182–190.
- [12] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Real-time systems*, vol. 2, no. 4, pp. 301–324, October 1990.
- [13] K. Albers, F. Bodmann, and F. Slomka, "Advanced hierarchical event-stream model," in *2008 Euromicro Conference on Real-Time Systems*. IEEE, 2008, pp. 211–220.
- [14] N. Fisher, T. P. Baker, and S. Baruah, "Algorithms for determining the demand-based load of a sporadic task system," in *Proceedings of the 12th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2006)*, Sydney, Australia, August, 16-18 2006.
- [15] A. Biondi and Y. Sun, "On the ineffectiveness of 1/m-based interference bounds in the analysis of global EDF and FIFO scheduling," *Real-Time Systems*, vol. 54, no. 3, pp. 515–536, July 2018.
- [16] B. Brandenburg and M. Gül, "Global scheduling not required: Simple, near-optimal multiprocessor real-time scheduling with semi-partitioned reservations," in *Proceedings of the 37th Real-Time Systems Symposium (RTSS 2016)*, Porto, Portugal, November 29 - December 2 2016.
- [17] A. Davare, Q. Zhu, M. Di Natale, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli, "Period optimization for hard real-time distributed automotive systems," in *2007 44th ACM/IEEE Design Automation Conference*, June 2007.
- [18] S. Bozhko and B. B. Brandenburg, "Abstract Response-Time Analysis: A Formal Foundation for the Busy-Window Principle," in *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), 2020, pp. 22:1–22:24.
- [19] J. Palencia and M. Harbour, "Offset-based response time analysis of distributed systems scheduled under edf," in *15th Euromicro Conference on Real-Time Systems, 2003. Proceedings.*, 2003, pp. 3–12.
- [20] M. Spuri, "Analysis of Deadline Scheduled Real-Time Systems," INRIA, Research Report RR-2772, 1996, projet REFLECS. [Online]. Available: <https://hal.inria.fr/inria-00073920>
- [21] N. Guan and W. Yi, "General and efficient response time analysis for edf scheduling," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014, pp. 1–6.
- [22] M. Park and H. Park, "An efficient test method for rate monotonic schedulability," *IEEE Transactions on Computers*, vol. 63, no. 5, pp. 1309–1315, 2012.
- [23] P. Pazzaglia, A. Biondi, and M. D. Natale, "Simple and general methods for fixed-priority schedulability in optimization problems," in *Proceedings of the International Conference on Design, Automation and Test in Europe (DATE 2019)*, March 2019.
- [24] F. Zhang and A. Burns, "Schedulability analysis for real-time systems with EDF scheduling," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1250–1258, April 2009.
- [25] A. Hamann, D. Dasari, F. Wurst, I. Saudo, N. Capodiecì, P. Burgio, and M. Bertogna, WATERS Industrial Challenge 2019.
- [26] G. Buttazzo, E. Bini, and Y. Wu, "Partitioning real-time applications over multicore reservations," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 302–315, 2011.
- [27] P. Pazzaglia, A. Biondi, and M. Di Natale, "Optimizing the functional deployment on multicore platforms with logical execution time," in *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019, pp. 207–219.
- [28] S. K. Baruah, "Partitioning real-time tasks among heterogeneous multiprocessors," in *International Conference on Parallel Processing, 2004. ICPP 2004*. IEEE, 2004, pp. 467–474.
- [29] S. K. Baruah, V. Bonifaci, R. Bruni, and A. Marchetti-Spaccamela, "Ilp-based approaches to partitioning recurrent workloads upon heterogeneous multiprocessors," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, 2016, pp. 215–225.
- [30] —, "Ilp models for the allocation of recurrent workloads upon heterogeneous multiprocessors," *Journal of Scheduling*, vol. 22, no. 2, pp. 195–209, 2019.
- [31] A. Biondi, G. Buttazzo, and M. Bertogna, "A design flow for supporting component-based software development in multiprocessor real-time systems," *Real-Time Systems*, vol. 54, no. 4, pp. 800–829, 2018.
- [32] H. Chen, A. M. K. Cheng, and Y.-W. Kuo, "Assigning real-time tasks to heterogeneous processors by applying ant colony optimization," *Journal of Parallel and Distributed Computing*, vol. 71, no. 1, pp. 132–142, 2011.
- [33] M. Pongothai, A. Rajeswari, and V. Kanishkan, "A heuristic based real time task assignment algorithm for heterogeneous multiprocessors," *IEICE Electronics Express*, vol. 11, no. 3, 2014.
- [34] H.-E. Zahaf, R. Olejnik, G. Lipari *et al.*, "Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms," *Journal of Systems Architecture*, vol. 74, pp. 46–60, 2017.
- [35] A. Mascitti, T. Cucinotta, and L. Abeni, "Heuristic partitioning of real-time tasks on multi-processors," in *IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 2020, pp. 36–42.
- [36] A. Mascitti, T. Cucinotta, and M. Marinoni, "An adaptive, utilization-based approach to schedule real-time tasks for arm big. little architectures," *ACM SIGBED Review*, vol. 17, no. 1, pp. 18–23, 2020.
- [37] A. Mascitti, T. Cucinotta, M. Marinoni, and L. Abeni, "Dynamic partitioned scheduling of real-time tasks on arm big. little architectures," *Journal of Systems and Software*, vol. 173, p. 110886, 2021.
- [38] D. Casini, P. Pazzaglia, A. Biondi, and M. Di Natale, "Optimized partitioning and priority assignment of real-time applications on heterogeneous platforms with hardware acceleration," *Journal of Systems Architecture*, vol. 124, p. 102416, 2022.
- [39] Z. Houssam-Eddine, N. Capodiecì, R. Cavicchioli, G. Lipari, and M. Bertogna, "The hpc-dag task model for heterogeneous real-time systems," *IEEE Transactions on Computers*, 2020.
- [40] M. Günzel, G. von der Brüggen, and J.-J. Chen, "Suspension-aware earliest-deadline-first scheduling analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 4205–4216, 2020.
- [41] F. Aromolo, A. Biondi, and G. Nelissen, "Response-Time Analysis for Self-Suspending Tasks Under EDF Scheduling," in *34th Euromicro Conference on Real-Time Systems (ECRTS 2022)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 231, 2022, pp. 13:1–13:18.
- [42] T. A. Henzinger, B. Horowitz, and C. M. Kirsch, "Giotto: A time-triggered language for embedded programming," in *International Workshop on Embedded Software*. Springer, 2001, pp. 166–184.
- [43] P. Pazzaglia, D. Casini, A. Biondi, and M. Di Natale, "Optimal memory allocation and scheduling for DMA data transfers under the LET paradigm," in *58th Design Automation Conference (DAC)*, 2021.
- [44] D. B. de Oliveira, D. Casini, R. S. de Oliveira, and T. Cucinotta, "Demystifying the real-time linux scheduling latency," in *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- [45] D. B. de Oliveira, D. Casini, and T. Cucinotta, "Operating system noise in the linux kernel," *IEEE Transactions on Computers*, pp. 1–12, 2022.
- [46] D. Casini, L. Abeni, A. Biondi, T. Cucinotta, and G. Buttazzo, "Constant bandwidth servers with constrained deadlines," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems (RTNS 2017)*, Grenoble, France, October, 4-6 2017.
- [47] I. Shin and I. Lee, "Compositional real-time scheduling framework," in *25th IEEE International Real-Time Systems Symposium*. IEEE, 2004, pp. 57–67.