Resource Utilization and Quality-of-Control Trade-off for a Composable Platform

Juan Valencia*, E.P. (Eelco) van Horssen†, Dip Goswami*, W.P.M.H. (Maurice) Heemels†, and Kees Goossens*
Control Systems Technology† and Electronic Systems* Groups - Eindhoven University of Technology
{j.valencia, e.p.v.horssen, d.goswami, w.p.m.h.heemels, k.g.w.goossens}@tue.nl

***Abstract*—This paper deals with implementation of *feedback* controllers on embedded platforms and investigates the trade-off between Quality-of-Control (QoC) and resource utilization. In particular, we consider a setting where the embedded platform executes multiple applications including the control application under consideration. Such a setting is common in domains like automotive where consolidation of several applications is desirable for cost reasons. While tackling inter-application interference is a challenge, our platform offers composability using resource *virtualization* allowing for interference-free application development and cycle-accurate timing behavior. In this work, from the feedback control perspective, we show that platform timing behavior can be characterized by a *finite*, *known* and *periodic* set of sampling intervals for a given resource allocation. Utilizing the platform timing, we show that the control design problem can be transformed into a classical discrete-time Linear Quadratic Regulator (LQR) problem which can be efficiently solved to obtain optimal QoC for a given resource allocation. Our method is validated both in simulation and experiments, considering a Multiple-Input and Multiple-Output (MIMO) control application.**

I. Introduction

Consolidation of multiple applications into a single platform is a trend [1] in cost-sensitive industries for obvious reasons (e.g., a significant reduction in infrastructure and complexity of hardware management and maintenance). The challenge in realizing this consolidation is the deployment of effective coexistence mechanisms to allow for interference-free application development. One way to achieve this is to create multiple virtual processors per physical processors by using a micro-kernel. In this work, the virtualization is realized in the Composable and Predictable System on Chip (CompSOC) platform [2]. The physical processors, their interconnections, and memories are virtualized following Time-Division Multiplexing (TDM) policies. Unlike, Earliest Deadline First or Rate-Monotonic, the work-non-conservative nature of the TDM policies allows for interference-free execution. In a virtualized platform, a fraction of the platform resources can be allocated to a control application, such that the controller can run without interference. We address the problem of resource-efficient implementation of feedback controllers on such a platform.

Given a resource allocation, it is important to design the controller in an optimal way to get maximum QoC out of the allocated resource. Standard optimal control strategies typically use uniform sampling schemes [3]. For control applications, a shorter sampling interval typically leads to a higher QoC [4]–[6]. However, a shorter sampling interval implies a higher resource usage in terms of communication and computation. This clearly creates a trade-off between the resource utilization and resulting QoC and it is of importance to characterize this trade-off quantitatively in order to make educated design decisions. From resource utilization perspective, a uniform sampling scheme might not be the optimal choice [7]. A resource-efficient implementation of feedback controller often leads to a non-uniform sampling scheme [8]. This gives rise to the need for control design methodologies taking into account non-uniform sampling intervals resulting from a given resource allocation. We tackle this problem in this work based on ideas from periodic control systems [9] and connect this to the resource utilization on CompSOC.

Contributions: In this work, we first show that the non-uniform sampling intervals resulting from our composable platform can be characterized by a finite, known and periodic sequence for a given resource allocation. For such a sequence of sampling intervals, using a time-lifted reformulation [10], we recover a periodicity property [9] for which the optimal control problem can be solved using periodic Riccati equations [11]. Solving them, we obtain an optimal switched feedback controller for the resulting periodic timing behavior from the platform. As such, we present a co-design framework that allows for trade-off analysis between resource utilization and QoC for an embedded control implementation and illustrate this on a MIMO control application, both in simulation and experiments.

II. Composable Platform

We consider a tile-based architecture that offers configuration with multi-processors (processor tiles), interconnections through a Network-on-Chip (NoC), and memories (memory tiles) within the same platform. An example architecture is shown in Fig. 1. Each processor tile is mainly composed of a MicroBlaze soft-core processor. The monitor tile is for debugging purposes. The memory tile contains the external memory interface and controller, and the NoC provides interconnection between the tiles. To enable independent implementation, verification and execution of multiple applications, the platform offers composability by virtualizing all processors, interconnections, and memory resources.

A. Virtualization

For platform virtualization, we use CoMik (Composable and Predictable Micro-kernel) that creates multiple virtual processors (VPs) that can be used as dedicated resources [12]. Each VP's utilization of the underlying physical processors and their interconnections (i.e., NoC communication) are allocated in a TDM manner. Using perfectly periodic TDM policies both in the processors and their interconnections, the platform

978-3-9815370-7-9/DATE16/

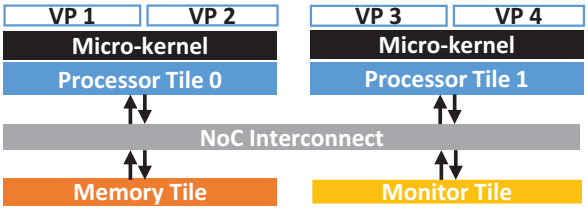

Fig. 1. Example: Composable platform using virtualization.

achieves global synchronization with a very fine (i.e., cycle accurate) time granularity.

To achieve cycle-accurate temporal behavior, we split the TDM frame in terms of fix duration CoMik slots of length ω clock cycles and partition slots (or VPs) of length ψ clock cycles. The CoMik slots enable jitter-free context switching between VPs, and the applications only execute on the VPs. Overall, the processor utilization at the application-level is given by $\frac{\psi}{\psi+\omega}$ which indicates that a large ω is undesired as it reduces VPs utilization. An application is executed in an allocated partition slot (or VP) and is paused every time a new CoMik slot starts. Its execution is only resumed in the next partition slot assigned for the same application. The TDM frame is defined at design time with desired execution order of the VPs. In Fig. 2 this mechanism is illustrated by dividing a TDM frame into four partition and CoMik slots, where two applications have been assigned with two partition slots each.

B. Platform Resource Utilization

We use a TDM frame consisting of $N \geq 1$ partition slots. The total TDM frame duration is given by $N \times (\psi + \omega)$ clock cycles. For a set Λ of at most N applications (e.g., control and multimedia applications), the number of partition slots, i.e. the amount of resources, assigned to a unique application $\lambda \in \Lambda$ is given by the function $S : \Lambda \rightarrow \mathbb{N}_{[1,N]}$ [1]. The indices of the slots allocated to application λ are given by the set $A(\lambda) \subseteq \mathbb{N}_{[1,N]}$. Since each partition slot has ψ clock cycles, an application uses $S(\lambda) \cdot \psi$ clock cycles during one TDM frame. The resource utilization by an application $\lambda \in \Lambda$ (as a fraction of the total resource) is given by,

$$R(\lambda) := \frac{S(\lambda)}{N} \cdot \frac{\psi}{(\psi + \omega)} \cdot 100[\%]. \tag{1}$$

In Fig. 2, we show, for $N = 4$, an example of resource utilization for two unique applications $\Lambda = \{\lambda_C, \lambda_2\}$. In this example, the first and last partition slots are allocated to the application λ_C (i.e., control application under study) and the other two partition slots are allocated to application λ_2 (e.g., multimedia applications). Thus, $S(\lambda_C) = 2$, $A(\lambda_C) = \{1, 4\}$, $S(\lambda_2) = 2$, and $A(\lambda_2) = \{2, 3\}$.

In summary, the virtualization capability of the platform enables the development and execution of applications by scheduling them into customizable partition slots. This allows application designers to take into account the timing properties of the platform (e.g., slot lengths and resource allocation) in order to independently develop and verify applications on this platform whilst ensuring that there will be no interference between applications.

[1] $\mathbb{N}_{[1,N]}$ denotes the natural numbers in the interval $[1, N]$, i.e. $\mathbb{N}_{[1,N]} = \{n \in \mathbb{N} \mid 1 \leq n \leq N\}$.

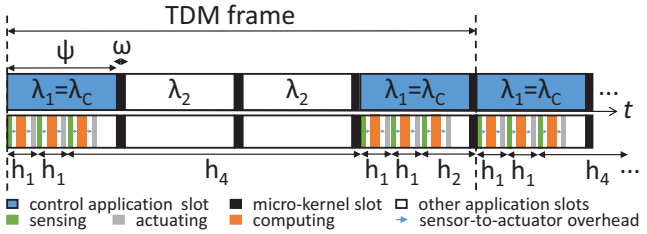

Fig. 2. Resource utilization example, for $N = 4$, with two applications $\Lambda = \{\lambda_C, \lambda_2\}$ where $S(\lambda_C) = 2$ and $A(\lambda_C) = \{1, 4\}$. The patterns repeat after the end of the TDM period. (above) The black blocks indicate the CoMik slots while the blue and white blocks indicate partitions slots for applications λ_C and λ_2, respectively. (below) Control application tasks execution and allocation-dependent sampling intervals.

Recall that, the variables ψ and ω were given as a number of clock cycles. In what follows, the length of a clock cycle is assumed to be known and the variables ψ and ω will refer to the slot time in seconds for ease of exposition.

III. Sampled-Data Control

Since the platform allows for independent development by functional and temporal separation, we focus only on the control application $\lambda_C \in \Lambda$.

The control application λ_C is related to controlling a linear time-invariant (LTI) continuous-time (CT) system represented by the differential equations

$$\dot{x}(t) = A_C x(t) + B_C u(t), \quad x(0) = x_0 \tag{2}$$

where $x(t) \in \mathbb{R}^{n^x}$ [2] is the state of the system, and $u(t) \in \mathbb{R}^{n^u}$ is the control input applied to the system at time $t \in \mathbb{R}_{\geq 0}$. Matrices $A_C \in \mathbb{R}^{n^x \times n^x}$ and $B_C \in \mathbb{R}^{n^x \times n^u}$ are constant.

The objective is to control the CT system such that the cost criterion [3]

$$J := \int_0^\infty (x^\top Q_C x + u^\top R_C u) dt, \quad Q_C \succ 0, R_C \succ 0, \tag{3}$$

is minimized, which corresponds to the classical linear quadratic regulator (LQR) problem (see [3]). We adopt the standard assumptions that (A_C, B_C) is controllable, B_C has full column rank, and $(A_C, Q_C^{\frac{1}{2}})$ is observable.

Remark III.1. *Having large Q_C compared to R_C puts the focus on making the state small (short settling time) possibly at the cost of large control inputs. By increasing R_C, large control inputs are penalized, typically leading to a slower response.*

We are going to sample the system (2) (i.e. read out sensors) at discrete time instances t_k with sample index $k \in \mathbb{N}_{\geq 1}$. The waiting time until the first sampling instance t_1 is typically considered to be zero, i.e. $t_1 = 0$. The state at the sampling instances can then be described in discrete time (DT) by

$$x_k := x(t_k), \qquad k \in \mathbb{N}_{\geq 1}. \tag{4}$$

The execution time on the platform (i.e. reading sensors, computation of the control inputs, and updating actuators) results in a sensing-to-actuation delay T, which is detailed

[2] $\mathbb{R}^{n^x}$ denotes a (column) vector of real numbers of length n^x.

[3] The inequality $(M \succeq 0)$ $M \succ 0$ means that the matrix M is symmetric and positive (semi-)definite.

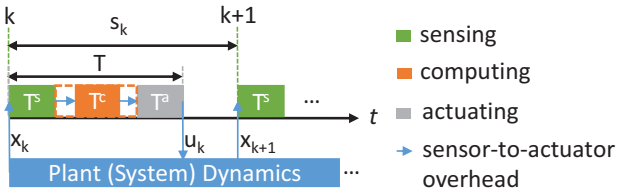

Fig. 3. Timing diagram of the control application.

in Section IV-A. We update the actuation signal in zero-order hold (ZOH) actuation scheme

$$u(t) = u_k, \qquad t \in [t_k + T, t_{k+1} + T) \tag{5}$$

where u_k are the to-be-designed piecewise constant control inputs. The sampling interval, i.e. the time between samples k and $k+1$, is denoted by s_k. In Fig. 3 the timings of the interaction between the platform and the plant are shown.

IV. Composable Embedded Control Systems

We implement the control application λ_C on a platform with an example architecture consisting of two synchronous processor tiles, one memory tile, one monitor tile, and the NoC (see Fig. 1). We map all tasks of the control application (that only runs on its allocated partition slots) onto one processor tile on the example architecture and characterize the exact timing behaviour of λ_C in the platform. The timing behaviour is then used to choose the sampling intervals.

A. Control Application Timing Properties

The control application is implemented by sequentially executing *sensing* (reading of sensors), *computing* (computation of actuation signals), and *actuating* (writing to actuators) tasks, subsequently, resulting in sensing-to-actuation delay. This results in the control application execution time or control task time, which is then defined as

$$T := T^s + T^c + T^a + T^o \tag{6}$$

where T^s, T^a and T^c are the execution times of the *sensing*, *computing*, and *actuating* tasks in the platform, respectively. Any overhead given by other operations in the *sensor-to-computing*, *computing-to-actuating*, and *actuating-to-plant* paths is captured by the overhead T^o. In our setup, we make sure that the sampling interval s_k is longer than the execution time T, i.e., $s_k \geq T$, and that the partition slot length ψ is larger than T, i.e., $T < \psi$. The execution of the application tasks is visualized in Fig. 3.

B. Periodic Sampling Pattern

A classical control implementation is to use the minimum sampling interval $s_k = T$ for all $k \in \mathbb{N}_{\geq 1}$. However, resource limitation prevents from such implementation. Instead, for a given allocation $A(\lambda_c)$, we adopt a finite sequence of sampling intervals that occur periodically with the TDM frame, according to the following scheme:

We consider the base or minimum sampling interval to be

$$h_1 := T. \tag{7}$$

Our approach is to consecutively execute the control task as many times as possible within a partition slot, i.e. $\left\lfloor \frac{\psi}{h_1} \right\rfloor$ times. If the remaining time in the partition slot is shorter than T, i.e. when $(\psi - \left\lfloor \frac{\psi}{h_1} \right\rfloor h_1) < T$, a complete execution is not possible anymore. In that case, the sensing task is delayed to the start of the next assigned partition slot. The last sampling interval in the partition slot is thus extended according to the resource allocation $A(\lambda)$. The other sampling intervals, for $2 \leq i \leq N+1$, are given by

$$h_i = T + (\psi - \left\lfloor \frac{\psi}{h_1} \right\rfloor h_1) + (i-1)\omega + (i-2)\psi, \tag{8}$$

where the four terms of (8) correspond to the execution time, the remaining time in the partition slot, the number of CoMik slots until the next assigned slot, and the number of unassigned partition slots until the next assigned slot, respectively. Then, the sequence of sampling intervals in the TDM frame can be denoted by the tuple

$$H := (h_i)_{\text{ occurring in order according to } A(\lambda_C)}. \tag{9}$$

By the above scheduling scheme, we avoid the application execution drift along TDM frames giving us a small number of elements in H. In our scheme, the number of unique elements $h_i \in H$, denoted by $\text{unique}(H)$, is limited by

$$\frac{(\text{unique}(H) - 1) \cdot \text{unique}(H)}{2} \leq N. \tag{10}$$

For example, H can have at most five unique elements for $N = 10$. For the resource allocation in Fig. 2, the periodic pattern $H = (h_1, h_1, h_4, h_1, h_1, h_2)$ is illustrated for the case where $\left\lfloor \frac{\psi}{h_1} \right\rfloor = 3$.

The sampling intervals are then given, for $r := \text{card}(H)$ and $H(j)$ the j-th element in H, by $s_k = H(1 + (k-1 \mod r))$, i.e. s_k is takes the value of h_i after sample k in accordance with H. Thus, we utilize the periodicity of the sequence H in the design of the control law.

V. Control Design

This section describes the design of sampled-data optimal control taking into account periodically switched sampling pattern H resulting from a given resource utilization in the platform. The design steps, visualized in Fig. 4, are detailed for tutorial purpose, see, e.g., [3], [9], [11], [13] for more details.

A. Discrete-Time Problem

We compute the evolution of the system in discrete time by exact discretization of the system (see e.g. [13]). The evolution of the CT state after sample $k \in \mathbb{N}_{\geq 1}$ can be described as

$$x(t_k + \tau) = \begin{cases} \Phi(\tau)x_k + \Gamma_1(\tau)u_{k-1}, & \text{when } \tau \in [0, T) \\ \Phi(\tau)x_k + \Gamma_2(\tau)u_{k-1} + \Gamma_3(\tau)u_k, & \\ \qquad \text{when } \tau \in [T, t_{k+1} - t_k), & \end{cases}$$

where we distinguish between the situations before and after the actuation update, and where [4]

$$\Phi(\tau) := e^{A_C \tau}$$
$$\Gamma_1(\tau) := \int_0^\tau \Phi(s) ds B_C = [I_{n^x} \quad 0]\, e^{\begin{bmatrix} A_C & B_C \\ 0 & 0 \end{bmatrix}\tau} \begin{bmatrix} 0 \\ I_{n^u} \end{bmatrix}$$
$$\Gamma_2(\tau) := \Phi(\tau - T)\Gamma_1(T)$$
$$\Gamma_3(\tau) := \Gamma_1(\tau - T).$$

[4] $[I_{n^x} \quad 0]$ denotes the concatenation of an identity matrix of dimension n^x and a zero matrix of dimension $n^x \times n^u$.

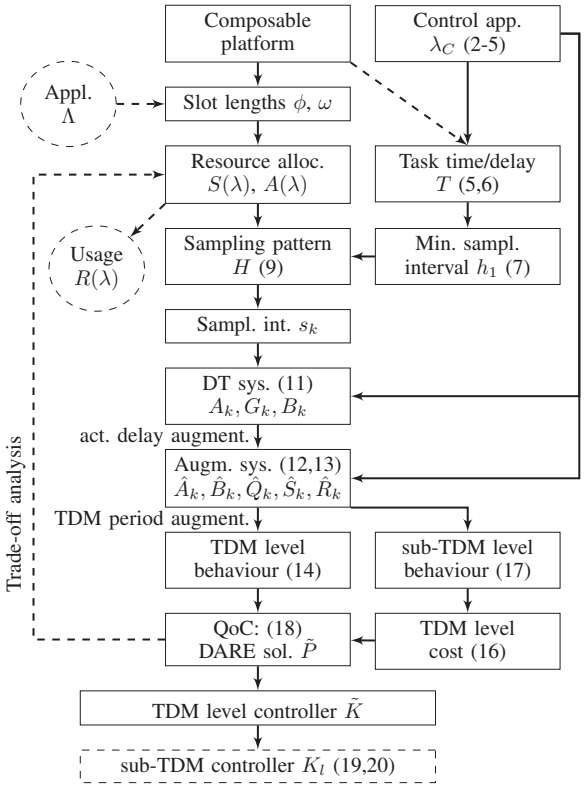

Fig. 4. Flowchart of the design procedure (with equation references).

Note that the system matrices only depend on the length of the sampling interval and not on the sample index. The system can now be represented by the discrete-time (DT) model

$$x_{k+1} = A_k x_k + G_k u_{k-1} + B_k u_k, \quad k \in \mathbb{N}_{\geq 1}, \qquad (11)$$
$$A_k := \Phi(s_k), \quad G_k := \Gamma_2(s_k), \quad B_k := \Gamma_3(s_k),$$

where $s_k = H(1 + (k-1 \mod r))$, $r := \mathrm{card}(H)$ as detailed in Section IV-B.

By augmenting the state with the delayed actuation $\xi_k = [x_k^\top, u_{k-1}^\top]^\top$ (see e.g. [10]) the system (11) can be rewritten in standard discrete-time linear time-varying (DT-LTV) form

$$\xi_{k+1} = \begin{bmatrix} A_k & G_k \\ 0 & 0 \end{bmatrix} \xi_k + \begin{bmatrix} B_k \\ I \end{bmatrix} u_k = \hat{A}_k \xi_k + \hat{B}_k u_k, \qquad (12)$$

for all $k \in \mathbb{N}_{\geq 1}$, with initial state $\xi_1 = [x_1^\top, u_0^\top]^\top$.

A discrete-time representation [13] of the cost is given by

$$J = \sum_{k=1}^{\infty} \int_{t_k}^{t_{k+1}} \begin{bmatrix} x(s) \\ u(s) \end{bmatrix}^\top \begin{bmatrix} Q_C & 0 \\ 0 & R_C \end{bmatrix} \begin{bmatrix} x(s) \\ u(s) \end{bmatrix} ds$$
$$= \sum_{k=1}^{\infty} \xi_k^\top \hat{Q}_k \xi_k + 2\xi_k^\top \hat{S}_k u_k + u_k^\top \hat{R}_k u_k, \quad \begin{bmatrix} \hat{Q}_k & \hat{S}_k \\ \hat{S}_k^\top & \hat{R}_k \end{bmatrix} \succ 0, \qquad (13)$$

where [5]

$$\hat{Q}_k := \int_0^T \star^\top \begin{bmatrix} Q_C & 0 \\ 0 & R_C \end{bmatrix} \begin{bmatrix} \Phi(s) & \Gamma_1(s) \\ 0 & I \end{bmatrix} ds + \int_T^{s_k} \star^\top \begin{bmatrix} Q_C & 0 \\ 0 & R_C \end{bmatrix} \begin{bmatrix} \Phi(s) & \Gamma_2(s) \\ 0 & 0 \end{bmatrix} ds,$$
$$\hat{S}_k := \int_T^{s_k} \begin{bmatrix} \Phi(s) & \Gamma_2(s) \\ 0 & 0 \end{bmatrix}^\top \begin{bmatrix} Q_C & 0 \\ 0 & R_C \end{bmatrix} \begin{bmatrix} \Gamma_3(s) \\ I \end{bmatrix} ds,$$
$$\hat{R}_k := \int_T^{s_k} \star^\top \begin{bmatrix} Q_C & 0 \\ 0 & R_C \end{bmatrix} \begin{bmatrix} \Gamma_3(s) \\ I \end{bmatrix} ds,$$

depend on the sampling intervals s_k and the delay T and can be computed by numerical integration.

The control problem can now be formulated as follows: Given x_1 and u_0 (i.e. ξ_1)

$$J^\star(\xi_1) := \min_{\{u_k\}_{k \in \mathbb{N}_{\geq 1}}} \quad (13) \quad \text{subject to} \quad (12).$$

In general, this problem does not have a closed-form solution for arbitrary $\{t_k\}_{k \in \mathbb{N}_{\geq 1}}$. However, on the composable platform under consideration, the sampling intervals occur in the periodic sequence H. Hence, the set of possible $\hat{A}_k$ and $\hat{B}_k$ can be precomputed and result in a discrete-time linear periodically time-varying (DT-LPTV) system [14] for which solutions to the control problem, e.g. using periodic Riccati equations, exist.

B. Periodicity: Time-lifted Reformulation

For a DT-LPTV system with period r, the dynamics and cost have the periodicity property

$$\hat{X}_{k+r} = \hat{X}_k, \ X \in \{A, B, Q, S, R\}.$$

With TDM period index j, the time-lifted reformulation [9]

$$\xi_{(j+1)r+1} = \tilde{A} \xi_{jr+1} + \tilde{B} \bar{u}_j, \qquad j \in \mathbb{N}_{\geq 0}, \qquad (14)$$

gives the dynamics over one TDM period, where [6]

$$\tilde{A} = \left[\prod_{k=r}^{1} A_k \right], \ \tilde{B} = \left[\left[\prod_{k=r}^{2} A_k \right] B_1 \quad \left[\prod_{k=r}^{3} A_k \right] B_2 \quad \cdots \quad A_r B_{r-1} \quad B_r \right].$$

and the cost can be written as

$$J = \lim_{p \to \infty} \sum_{j=0}^{p} \bar{\xi}_j^\top \bar{Q} \bar{\xi}_j + 2\bar{\xi}_j^\top \bar{S} \bar{u}_j + \bar{u}_j^\top \bar{R} \bar{u}_j \qquad (15)$$
$$= \lim_{q \to \infty} \sum_{j=0}^{q} \xi_{jr+1}^\top \tilde{Q} \xi_{jr+1} + 2\xi_{jr+1}^\top \tilde{S} \bar{u}_j + \bar{u}_j^\top \tilde{R} \bar{u}_j \qquad (16)$$
$$\tilde{Q} = \bar{A}^\top \bar{Q} \bar{A}, \ \tilde{S} = \bar{A}^\top \bar{S} + \bar{A}^\top \bar{Q} \bar{B}, \ \tilde{R} = \bar{R} + \bar{B}^\top \bar{Q} \bar{B}$$

using the augmented variables

$$\bar{\xi}_j = \begin{bmatrix} \xi_{jr+1} \\ \vdots \\ \xi_{jr+r} \end{bmatrix}, \ \bar{u}_j = \begin{bmatrix} u_{jr+1} \\ \vdots \\ u_{jr+r} \end{bmatrix}, \ \begin{aligned} \bar{Q} &= \mathrm{diag}_{k \in \mathbb{N}_{[1,r]}} \hat{Q}_k, \\ \bar{S} &= \mathrm{diag}_{k \in \mathbb{N}_{[1,r]}} \hat{S}_k, \\ \bar{R} &= \mathrm{diag}_{k \in \mathbb{N}_{[1,r]}} \hat{R}_k, \end{aligned}$$

and using the dynamics within the TDM period

$$\bar{\xi}_j = \bar{A} \xi_{jr+1} + \bar{B} \bar{u}_j, \qquad j \in \mathbb{N}_{\geq 0}, \qquad (17)$$

[5] $A^\top Q A$ denotes $\star^\top Q A$ for any matrices A, Q of appropriate dimensions.
[6] $\prod_{k=r}^{m} A_k$ denotes (for $r \geq m$) the multiplication $A_r A_{r-1} \cdot A_{m+1} A_m$.

where

$$\bar{A} = \begin{bmatrix} I \\ A_1 \\ A_2A_1 \\ \vdots \\ \prod_{k=r-1}^{1} A_k \end{bmatrix}, \; \bar{B} = \begin{bmatrix} 0 & & & \cdots & & 0 \\ B_1 & 0 & & & & 0 \\ A_2B_1 & B_2 & 0 & & & 0 \\ \vdots & & & \ddots & \ddots & \vdots \\ \left[\prod_{k=r-1}^{2} A_k\right] B_1 & \left[\prod_{k=r-1}^{3} A_k\right] B_2 & \cdots & B_{r-1} & 0 \end{bmatrix}.$$

Now, we note that matrices $(\tilde{A}, \tilde{B}, \tilde{Q}, \tilde{S}, \tilde{R})$ in (14) and (16) do not depend on the period index j, i.e. they are time-invariant. The lifted problem thus has the standard time-invariant DT LQR form which can be solved efficiently.

C. Periodically Switched Controller Synthesis

For the time-invariant lifted reformulation (14) and (16), standard optimal control methods [3], [15] can find the optimal solution

$$J^\star(\xi_1) = \xi_1^\top \tilde{P} \xi_1, \tag{18}$$

where $\tilde{P}$ is the unique positive definite solution to the discrete-time algebraic Riccati equation (DARE)

$$\tilde{P} = \tilde{A}^\top \tilde{P} \tilde{A} + \tilde{Q} - (\tilde{B}^\top \tilde{P} \tilde{A} + \tilde{S}^\top)^\top (\tilde{R} + \tilde{B}^\top \tilde{P} \tilde{B})^{-1} (\tilde{B}^\top \tilde{P} \tilde{A} + \tilde{S}^\top).$$

Furthermore, $V(\xi) = \xi^\top \tilde{P} \xi$ is a Lyapunov function for the system at the start of the TDM period, ensuring stability for the optimal control actions

$$\bar{u}_j = -\tilde{K} \xi_{jn+1}, \quad \tilde{K} := (\tilde{R} + \tilde{B}^\top \tilde{P} \tilde{B})^{-1} (\tilde{B}^\top \tilde{P} \tilde{A} + \tilde{S}^\top).$$

Note that this is the solution to the lifted problem over a TDM period. This can be transformed into a state feedback for the original DT-LPTV system (12) leading to,

$$u_k = -K_l \xi_k, \text{ with } l = (k-1 \mod r) + 1, \; k \in \mathbb{N}_{\geq 1}, \tag{19}$$

where $\xi_k = [x_k^\top, u_{k-1}^\top]^\top$. From $P_{n+1} = \tilde{P}$, the solutions P_l for $l \in \mathbb{N}_{[1,r]}$ can be found from the solution to the standard finite horizon discrete-time dynamic Riccati equation (DDRE)

$$\begin{aligned} P_l = {} & A_l^\top P_{l+1} A_l + Q_l \\ & - (B_l^\top P_{l+1} A_l + S_l^\top)^\top (R_l + B_l^\top P_{l+1} B_l)^{-1} (B_l^\top P_{l+1} A_l + S_l^\top), \end{aligned}$$

which represents the discrete-time periodic Riccati equation (DPRE) [11] when $P_{l+n} = P_l$. The control gains K_l, $l = 1, 2, \ldots, r$, are given by

$$K_l := (R_l + B_l^\top P_{l+1} B_l)^{-1} (B_l^\top P_{l+1} A_l + S_l^\top). \tag{20}$$

Hence, (19) is the optimal feedback law which is to be applied in (5).

VI. Evaluation Results

To illustrate our methods, we simulate a modified version of a two-body oscillator model

$$A_C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & -2 & 0 \end{bmatrix} \cdot 10^4, \; B_C = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Here, we have chosen a very fast system with very small time scales close to the sampling periods of the platform under consideration. We take initial conditions $x_0 = [0 \; 10 \; 0 \; 0]^\top$, $t_1 = 0$ (i.e. $x_1 = x_0$), and $u_0 = [0 \; 0]^\top$. The weighting matrices in the cost criterion are chosen as

$$Q_C = I_4, \quad R_C = I_2 \cdot 10^{-4},$$

such that the emphasis is on having small states.

The platform has been configured with $\omega = 34.31 \cdot 10^{-6}$[s], $\psi = 477.87 \cdot 10^{-6}$[s], and a global clock frequency of 120 MHz. The execution of the control task on the platform is measured to take $T = 138 \cdot 10^{-6}$[s]. The base sampling interval is taken as $h_1 = 140 \cdot 10^{-6}$[s], which results in $\left\lfloor \frac{\psi}{h_1} \right\rfloor = 3$. The number of partition slots N is chosen to be 10 (i.e., $N = 10$).

We computed the QoC for eight different resource allocations using an analytical approach and a MATLAB-based simulation. Tab. I shows the sampling pattern H, the allocation pattern $A(\lambda_C)$ and the increasing resource usage $R(\lambda_C)$. The resulting QoC, measured in terms of the optimal quadratic cost $J^\star(\xi_1)$ (18), is computed analytically and validated by simulation (using MATLAB).

Exp. 1 and Exp. 8 represent the cases where only one slot is used or all slots are used, and typically give the worst and best performance that can be achieved in our framework, respectively. For the chosen system, we find that more resources give better QoC (lower $J^\star(\xi_1)$), which is consistent with the expectations. However, the results show significant variation of QoC for variations in the pattern. Here, contiguous allocation is found to be better than a spread allocation. Hence, by changing allocation, one may find that less resources are needed to achieve the same QoC. Our methods provide a quantitative way to analyze this trade-off, leading to educated design decisions.

Hardware-In-the-Loop (HIL) simulation: we performed a HIL simulation for one of the cases in Tab. I (Exp.#2) for validating the platform timing behavior and our results. In the HIL simulation, we synthesized the CompSOC platform described in Section II on an FPGA module (Xilinx Virtex-6) and we simulated the discrete-time system Equation (11) as illustrated in Fig. 3. Fig. 5 and Fig. 6 show the measured behavior of the first state of $x(t)$ and the first input of $u(t)$, respectively. The results from the MATLAB-based and HIL simulations closely follow each other. It should be noted that the difference between MATLAB-based and HIL simulations results from the data truncation due to the transformation from double to single precision *floating point units*. We clearly see the oscillatory behaviour of the state, which was expected. Although not visible from the figures, the simulation shows that the system is indeed asymptotically stable (implying that the state converges to the zero state). The convergence rate depends on the chosen allocation. Difference in actuation is clearly visible from the staircase signal of the input.

Discussions: we show how our design framework can be used to analyze the trade-off between resource usage, resource pattern, and QoC. We note here that the initial condition, the cost criterion, and the timing scale of the platform with respect to the plant dynamics, greatly influence the QoC as well. In order to make a good design trade-off one should consider all these factors before deciding on a choice of resource allocation. For our chosen system, we found that contiguous allocation performed better than more spread allocation. As expected, for a fixed pattern, increasing resources improves performance.

Table I. ALLOCATIONS, SENSING INTERVALS, AND PERFORMANCE FOR $N = 10$ FOR THE CONTROLLED TWO-BODY OSCILLATOR. THE SAMPLING INTERVALS ARE GIVEN BY $\{h_1, h_2, \ldots, h_{11}\} = \{140, 232, 744, 1256, 1768, 2280, 2792, 3304, 3816, 4328, 4840\}\mu s$.

$Exp.\#$	$S(\lambda_C)$	$A(\lambda_C)$	$R(\lambda_C)$	H	$J^{\star}(\xi_1)$	J^{sim}
1	1	1	$\approx 9\%$	(h_1,h_1,h_{11})	6.119	6.115
2	3	1−3	$\approx 27\%$	$(h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_9)$	3.137	3.128
3	3	1,5,7	$\approx 27\%$	$(h_1,h_1,h_5,h_1,h_1,h_3,h_1,h_1,h_5)$	3.596	3.586
4	5	1−5	$\approx 45\%$	$(h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_7)$	2.411	2.398
5	5	1,3,4,6,9	$\approx 45\%$	$(h_1,h_1,h_3,h_1,h_1,h_2,h_1,h_1,h_3,h_1,h_1,h_4,h_1,h_1,h_3)$	2.696	2.685
6	7	1−7	$\approx 63\%$	$(h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_5)$	2.015	2.003
7	7	1,2,4,5,6,8,9	$\approx 63\%$	$(h_1,h_1,h_2,h_1,h_1,h_3,h_1,h_1,h_2,h_1,h_1,h_2,h_1,h_1,h_3,h_1,h_1,h_2,h_1,h_1,h_3)$	2.158	2.146
8	10	1−10	$\approx 90\%$	$(h_1,h_1,h_2) \times 10$	1.728	1.715

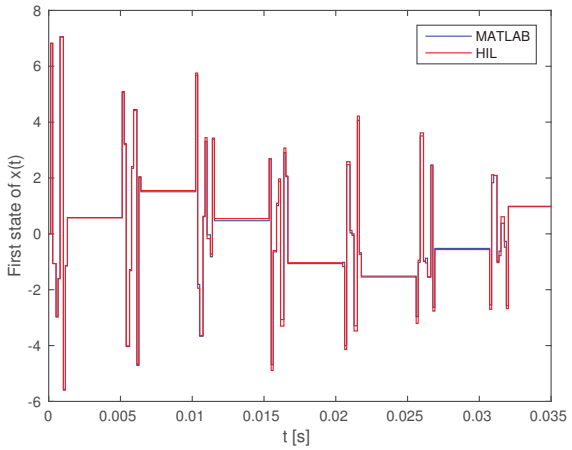

Fig. 5. Measured behaviour of the first state of $x(t)$ for Exp. $\#2$ for a section of the simulation time, using a MATLAB-based simulation and a HIL simulation.

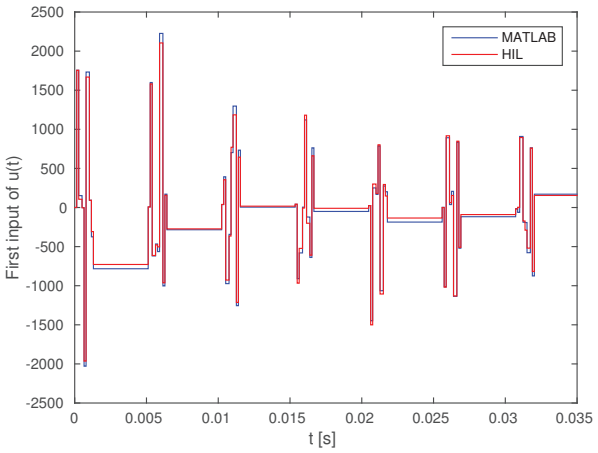

Fig. 6. First input of $u(t)$ for Exp. $\#2$ for a section of the simulation time, using MATLAB-based simulation and HIL simulation.

VII. CONCLUSIONS

We present a co-design framework for efficient implementation of feedback controllers on composable embedded platforms. The framework deals with three design considerations: the percentage resource utilization, the allocation pattern (e.g., contiguous or spread) and the (optimal) QoC. The QoC depends on both the utilization and allocation pattern. We demonstrated how our design framework can be used to analyze this trade-off in a quantitative manner thereby enabling educated designs of both the allocation pattern and the (optimal) feedback controller. For the studied example, we found that; (i) for a contiguous allocation pattern, a higher utilization provides a higher QoC, (ii) for a given allocation, a contiguous allocation pattern outperforms a spread allocation.

Acknowledgements This work was partially funded by projects STW 12697, VICI 11382 (NWO and STW), CATRENE CA505, BENEFIC, CA703 OpenES, CT217 RESIST, ARTEMIS 621429 EMC2 and 621353 DEWI.

REFERENCES

[1] R. Obermaisser, C. El Salloum, B. Huber, and H. Kopetz, "From a Federated to an Integrated Automotive Architecture," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 7, pp. 956–965, July 2009.

[2] K. Goossens, A. Azevedo, K. Chandrasekar, M. D. Gomony, S. Goossens, M. Koedam, Y. Li, D. Mirzoyan, A. Molnos, A. B. Nejad, A. Nelson, and S. Sinha, "Virtual Execution Platforms for Mixed-time-criticality Systems: The CompSOC Architecture and Design Flow," *SIGBED Rev.*, vol. 10, no. 3, pp. 23–34, Oct. 2013.

[3] K. J. Åström, *Introduction to stochastic control theory*. Elsevier, 1970.

[4] A. Cervin and P. Alriksson, "Optimal On-Line Scheduling of Multiple Control Tasks: A Case Study," in *ECRTS*, 2006.

[5] E. Bini and A. Cervin, "Delay-Aware Period Assignment in Control Systems," in *RTSS*, 2008.

[6] A. Anta and P. Tabuada, "On the Benefits of Relaxing the Periodicity Assumption for Networked Control Systems over CAN," in *RTSS*, 2009.

[7] J. Valencia, D. Goswami, and K. Goossens, "Composable platform-aware embedded control systems on a multi-core architecture," in *Digital System Design (DSD), Euromicro Conference on*, 2015.

[8] W. Heemels, K. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *IEEE Conference on Decision and Control (CDC) 2012, Hawaii, USA*, December 2012, pp. 3270–3285.

[9] S. Bittanti and P. Colaneri, *Periodic systems: filtering and control*. Springer Science & Business Media, 2008, vol. 5108985.

[10] M. Cloosterman, N. van de Wouw, W. Heemels, and H. Nijmeijer, "Stability of networked control systems with uncertain time-varying delays," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1575–1580, 2009.

[11] A. Varga, "On solving periodic riccati equations," *Numerical Linear Algebra with Applications*, vol. 15, no. 9, pp. 809–835, 2008.

[12] A. Nelson, A. Nejad, A. Molnos, M. Koedam, and K. Goossens, "CoMik: A predictable and cycle-accurately composable real-time microkernel," in *DATE*, 2014.

[13] K. J. Åström and B. Wittenmark, *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[14] R. Meyer and C. Burrus, "A unified analysis of multirate and periodically time-varying digital filters," *Circuits and Systems, IEEE Transactions on*, vol. 22, no. 3, pp. 162–168, Mar 1975.

[15] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005, 3rd Edition. Vol. 1 and Vol. 2.