Comparing Platform-aware Control Design Flows for Composable and Predictable TDM-based Execution Platforms

JUAN VALENCIA, DIP GOSWAMI, and KEES GOOSSENS, Eindhoven University of Technology, The Netherlands

We compare three platform-aware feedback control design flows that are tailored for a composable and predictable Time Division Multiplexing (TDM)-based execution platform. The platform allows for independent execution of multiple applications. Using the precise timing knowledge of the platform execution, we accurately characterise the execution of the control application (i.e., sensing, computing, and actuating operations) to design efficient feedback controllers with high control performance in terms of settling time. The design flows are derived for Single-Rate (SR) and Multi-Rate (MR) sampling schemes. We show the applicability of the design flows based on two design considerations and their trade-off: control performance and resource utilisation. The design flows are validated by means of MATLAB and Hardware-in-the-Loop (HIL) experiments for a motion control application.

CCS Concepts: • **Computer systems organization** → **Embedded software**; **Real-time systems**; • **Computing methodologies** → *Simulation types and techniques*;

Additional Key Words and Phrases: Embedded control systems, switched linear systems, TDM execution platforms, multi rate systems, Linear Quadratic Regulator (LQR)

ACM Reference format:
Juan Valencia, Dip Goswami, and Kees Goossens. 2019. Comparing Platform-aware Control Design Flows for Composable and Predictable TDM-based Execution Platforms. *ACM Trans. Des. Autom. Electron. Syst.* 24, 3, Article 32 (March 2019), 26 pages.
https://doi.org/10.1145/3315572

1 INTRODUCTION

Feedback control applications are used in a wide range of applications developed for cost-sensitive industries that include industrial automation, consumer applications, automotive, avionics, and many others. A great number of these applications demand high performance, low cost, and a short time to market. Their control task is implemented by three sequential and repetitive operations: sense or measure data from the system under control (plant), compute the actuation signals, and apply the actuation signals to the plant such that its behaviour is regulated. In many real-life

The authors acknowledge EU Grants. No. CATRENE CT217 RESIST, No. ECSEL 692455-2 ENABLE-S3, No. 737422 SCOTT, and No. 737453 I-MECH, and NL Grant No. STW ZERO.
Authors' address: J. Valencia, D. Goswami, and K. Goossens, Eindhoven University of Technology, Postbus 513, Eindhoven, NB, 5600 MB, The Netherlands; emails: {j.valencia, d.goswami, k.g.w.goossens}@tue.nl.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
1084-4309/2019/03-ART32 $15.00
https://doi.org/10.1145/3315572

scenarios, the controllers are implemented onto embedded platforms with severe resource constraints (e.g., on computation and communication).

The current practice is to dedicate an embedded platform for each application to avoid sharing resources. This guarantees interference-free execution of applications, which is essential to achieve a high performance and reduce Time to Market (TTM). Without resource sharing, such design solutions often lead to expensive implementations due to high cost of hardware.

Bringing multiple applications within a single embedded platform is a potential solution for such expensive implementation, but it poses several challenges. The most notable among them is to deal with inter-application interferences. Common approaches to tackle this challenge include the use of multi-core embedded platforms, where each core of the platform is allocated to a single application. However, other shared resources, such as cache memories and interconnections, lead to interference between applications (Subramanian et al. 2015). An alternative approach is to *partition* (or *virtualize*) resources in both time (e.g., scheduling) and space (e.g., memory regions) such that all interference between applications due to resource sharing is avoided, and the development and integration of multiple applications in the platform is eased and sped up.

Implementing precise Time Division Multiplexing (TDM) policies onto the embedded platform execution is one technique to partition shared resources. One such example is the Composable and Predictable System on Chip (CompSOC) embedded platform (Goossens et al. 2017). This platform uses the Composable and Predictable Microkernel (CoMik), which cycle-accurately partitions the processor execution in fixed duration slots (Nelson et al. 2014). In each of these slots, an application executes without any interference from other applications. We consider TDM-based execution platforms, such as CompSOC, as the implementation platform for the feedback control applications.

Generally, the feedback control applications are required to guarantee stability and provide a required performance. However, complying with requirements and enhancing performance do not only depend on meeting timing deadlines, as is commonly regarded in real-time applications, but, as we will see, it also depends on richer timing characteristics that are derived from both the platform and the application executions.

Platform-aware model-based design of control systems has been reported to improve the control application performance (Morelli and Natale 2014). That is, the knowledge of the precise execution time information can explicitly be considered in the design of the controller to achieve a higher performance, as well as to meet the design constraints. In this work, we present feedback control design flows for efficient deployment of controllers onto composable and predictable platforms. As a study case, we exploited the CompSOC platform timing mechanisms to use equidistant and non-equidistant sampling intervals in the controller design.

Contributions: The contributions of this article are detailed as follows:

- **Single-rate (SR) design flow:** Platform configured with *equidistant* sampling. This controller design has been adapted from Valencia et al. (2015) by replacing the pole-placement design with an Linear Quadratic Regulator (LQR) controller (Åström and Murray 2008). We use the settling time as the control performance metric. Thus, the LQR is tuned to optimise the performance. We use Particle Swarm Optimization (PSO) for the LQR tuning (Medina et al. 2017). We show that the SR design flow is suitable when the demand for the performance is relatively low (i.e., a longer settling time is acceptable). Better performance can be achieved with a higher resource utilisation.
- **Multi-rate Local Optimal (MRLO) design flow:** Platform configured to obtain a finite sequence of *periodic* and *non-equidistant* sampling intervals. We further identify the most frequently occurring sampling interval (nominal sampling interval) in the sequence. We

optimise the controller design for the nominal sampling interval. We adapted this design from Valencia et al. (2015) by replacing the pole-placement controller with an LQR controller accompanied with the PSO tuning technique. We ensure the overall switching stability between the non-equidistant sampling intervals by using a Lyapunov-based Linear Matrix Inequality (LMI) stability condition. We show that the MRLO design flow is suitable when the performance demand is also high (i.e., a short settling time is required). Better performance can be achieved with a higher resource utilisation.

- **Multi-rate Global Optimal (MRGO) design flow:** Platform configured to obtain a finite sequence of *periodic* and *non-equidistant* sampling intervals. The controller is designed by using a *time-lifted* formulation of the overall system (i.e., augmented system that results from the non-equidistant sampling intervals) such that it is transformed into a classic LQR design problem. We adapted this design from Valencia et al. (2016) by adding a continuous-time (CT) LQR heuristic technique. We show that the MRGO design flow is suitable when the performance demand is high, and a higher resource utilisation is acceptable. Better performance can be achieved with a higher resource utilisation.
- **Experimental study case:** A fourth-order motion control system plant has been considered to study the design flows. This plant is mainly composed of a mechanical setup where the two masses are connected to each other by a flexible bar, and a motor is directly connected to one of the masses. Such type of plant represents the characteristics of a wide range of industrial settings (e.g., automotive, avionics, biomedical devices).
- **Hardware-in-the-Loop (HIL) experiments:** We present a HIL experiment on the CompSOC platform. In this simulation, the controllers (i.e., derived from the SR, MRLO, and MRGO design-flows) are implemented on a processor where applications run under a TDM scheduling scheme. On a separate processor, the CT plant dynamics are emulated by running the discrete-time (DT) plant dynamics at a high sampling frequency.
- **Validation:** The design flows are validated with MATLAB and HIL experiments. In both experiments, the settling time is evaluated for different amount of resources that are assigned to the control application. The comparison shows the feasibility of implementing the proposed design flows onto an embedded platform to control a real plant.
- **Design guidelines:** We provide design guidelines that explain how to select and configure the best design flow depending on the design considerations, namely settling time and resource utilisation.

The remainder of the article is organised as follows. In Section 2, we present the related work from which we have taken many inspiring ideas. The composable and predictable TDM-based execution platform used in our work is presented in Section 3. The control application and the characterisation of its timing properties are described in Section 4. In Section 5, we describe the SR design flow, followed by the Multi-Rate (MR) design flows in Sections 6 and 7. In Section 8, we present the experimental study, where we give details about the motion control study case, the MATLAB and HIL experiments, the trade-off analysis between performance and resources allocated to the application, the impact of the platform settings reconfigurations on the performance, and the suggested design design guidelines. We finally draw conclusions in Section 9.

2 RELATED WORK

This work deals with the efficient implementation of feedback control applications on embedded platforms. For decades, the development of embedded control applications has been based on the *separation of concerns* principle between theoretical control and embedded systems disciplines (Årzén and Cervin 2005). The former is focused on control design with equidistant sampling intervals with *hard* execution deadlines. The latter is focused on developing scheduling mechanisms and computational models such that control applications meet these timing requirements during runtime. This design philosophy led to simpler control system models that often restrict control performance and stringent execution models with significant resource over-dimensioning. In contrast, a huge body of work has been reported on the platform-aware design philosophy where the emphasis is on co-design of control strategies and platform configurations (Cervin et al. 2003; Chang et al. 2017; Samii et al. 2009; Valencia et al. 2016; Wolf 2009). The idea is to take into account properties of platform resources in the control design and thereby improve the control performance.

The literature on platform-aware control design can broadly be classified based on the resource category, namely, computation (Aminifar et al. 2015, 2016; Biondi et al. 2018; Cervin et al. 2003, 2011; Goswami et al. 2013; Medina et al. 2017; Samii et al. 2009; Schneider et al. 2013), communication (Bauer et al. 2014; Deng et al. 2016; Goswami et al. 2014; Roy et al. 2016), memory (Chang et al. 2017), and power (Chang et al. 2014).

The key considerations of computation-aware control design methods are the trade-off analysis between resource usage and control performance, efficient implementation and performance optimisation. A trade-off analysis between the number of processing units used for the control application and the performance is presented in Medina et al. (2017) for data intensive control loops. Integrated communication and computation (priority-based) scheduling for distributed control is solved by constraint logic programming formulation in Samii et al. (2009). The works in Aminifar et al. (2015) and Cervin et al. (2003) present analysis frameworks to analyse the effect of execution jitter on control performance. To this end, the sampling interval and delay play crucial role both in control performance and scheduling. They are often used as an *interface* between the control and embedded systems design paradigms. The optimal sampling interval is found with respect to the control performance in Cervin et al. (2011) and Goswami et al. (2013) and similarly, the delay is optimised in Schneider et al. (2013). Aminifar et al. (2016) present a response time analysis with self- and event-triggered execution of control applications. Similarly, Biondi et al. (2018) present response time analysis for controllers running under variable sampling intervals such as an engine control system.

Along the direction of communication-aware control design, there has been emphasis on control/schedule co-design considering industrial bus systems such as FlexRay (Goswami et al. 2014; Roy et al. 2016), CAN (Deng et al. 2016), as well as wireless networks (Bauer et al. 2014). The key consideration is optimisation and analysis of the control performance taking account the bandwidth restrictions, scheduling policy, and uncertainty of the communication systems in distributed implementations. Co-optimisation considering memory-mapping of control applications has been reported recently in Chang et al. (2017). In the context of electric vehicle, the power consumption by the controller becomes a crucial design parameter and hence, optimised for a longer battery life and control performance in Chang et al. (2014).

Overall, many state-of-the-art strategies offer synthesis frameworks to schedule and map applications onto a targeted embedded platform taking into account computation, communication, memory, and power resources. Our approach differs in a number of aspects and takes inspiration from many works (see Table 1). First, we deal primarily with feedback control applications implemented onto a composable and predictable platform that allows for resource sharing between applications (e.g., CompSOC platform (Goossens et al. 2017)). This platform uses a TDM scheduling protocol that virtualises the resources that are assigned to applications into time partitions where applications execute. This virtualisation enables independent development and execution of applications, simplifying the mapping and scheduling of the applications to time partitions per application (Çela et al. 2014), unlike works that must include the scheduling of

Table 1. Comparison with State-of-the-art Works

Work	Scheduling	Performance Evaluation	Sampling	Trade-off	Design Flow	Experiments
(Arzen et al. 2000)	- rate monotonic - earliest deadline first	- error between reference and measured output	- SR	- performance vs. resource utilisation	no	- not specified
(Aminifar et al. 2012)	- static-cyclic - priority-based	- expected control performance - worst-case control performance	- SR	- runtime vs. # of applications	yes	- MATLAB
(Schneider et al. 2013)	- fixed-priority preemptive scheduling	- stability margin	- SR	- schedulability vs. performance - performance vs. utilisation	partial	- Co-sim. framework
(Goswami et al. 2012)	- time-triggered	- quadratic cost function	- SR	- performance vs. delay - performance vs. execution load	yes	- ILP-based optimisation - MATLAB
(Chang et al. 2018)	- fixed-priority preemptive scheduling	- settling time	- MR	- not specified	no	- INCHRON
our work	- TDM-based	- settling time	- SR - MR	- performance vs. utilisation - performance vs. platform settings	yes	- MATLAB - HIL

applications in their designs (Aminifar et al. 2012; Arzen et al. 2000; Chang et al. 2018; Goswami et al. 2012; Schneider et al. 2013). The time-triggered behaviour of the virtualisation mechanism brings resource utilisation limitations for classical control approaches that are based on equidistant sampling intervals (Aminifar et al. 2012; Arzen et al. 2000; Åström and Wittenmark 1990; Cervin et al. 2003; Goswami et al. 2012; Schneider et al. 2013). In contrast, we exploit frequent execution of the control application within its assigned partitions to increase the sampling frequency and potentially the control performance, at the cost of dealing with non-equidistant sampling intervals, similar to works (Chang et al. 2018; van Zundert and Oomen 2018). We present different control laws (i.e., LQR, LMI-based), and we investigate their stability and optimisation (e.g., PSO-based LQR tuning). We show how the design flows can be used to integrate manual and automated procedures for each control strategy. We use MATLAB and HIL experiments to validate our design-flows (MathWorks 2018). As seen in Table 1, many works focus on timing analysis experimentation, whereas in our work we propose a HIL framework to verify the feasibility of implementing our control designs in a real platform. We draw the design guidelines that are based on the tradeoffs that we have explored on the resources assigned to the applications, their performance, and the impact of the platform settings to the aforementioned design constraints.

In summary, our work is based on the TDM scheduling scheme that simplifies timing analysis of the applications, provides time-based performance metrics that allows to have real estimation of the effectiveness of our designs, allows for equidistant and non-equidistant sampling schemes to exploit performance, presents tradeoffs explorations to help the design flow selection, and presents HIL experiments to validate the implementation of our designs in a real embedded platform.

3 COMPOSABLE AND PREDICTABLE TDM-BASED EXECUTION PLATFORM

The CompSOC platform is configurable with processing units (processor tiles), interconnect Network on Chip (NoC), and memory units (memory tiles) (Goossens et al. 2017). An example architecture is shown in Figure 1. The processor tile is composed of a MicroBlaze soft-core processor,

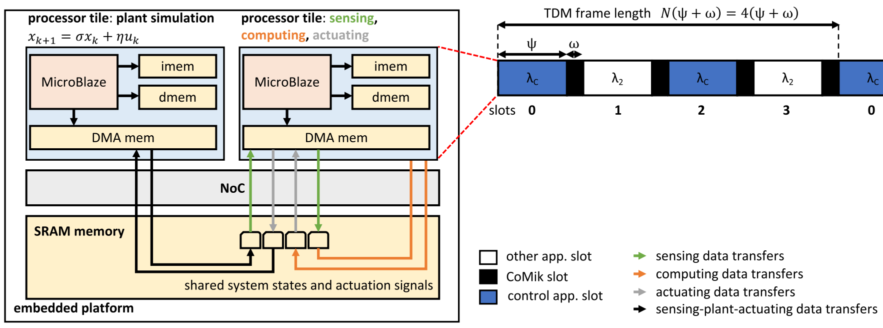


Fig. 1. Left: example of a TDM-based execution platform. Right: example for $N = 4$, with applications λ_C and λ_2 (where λ_C is the control application). The black blocks indicate the CoMik slots, while the blue and white blocks indicate the application slots for applications λ_C and λ_2, respectively. The TDM frame repeats infinitely. The numbers below the TDM frame denote the slots, which are indexed from 0 to 3.

instruction and data memory, and Direct Memory Access (DMA). The memory tile contains the Static Random-Access Memory (SRAM) memory interface, and the NoC provides the interconnections between the tiles. Resources are shared between applications with *composable* TDM arbiters, which means that the time slots allocated to each application are strictly periodic in time and of fixed duration with precision of a single clock cycle. This mechanism is used to partition the resources, such that the applications are loaded and run without affecting or being affected by other applications by even a single clock cycle. This makes it possible to implement a controller and emulate plant dynamics independently of other applications.

The CoMik micro-kernel partitions each processor execution in a TDM frame of size N slots, where each slot is composed of an application slot (i.e., where applications execute) of fixed duration ψ seconds and a CoMik slot (i.e., where the microkernel switches applications) of duration ω seconds. Thus, during runtime, each application is executed in its allocated slots and it is suspended every time a new CoMik slot starts. Its execution is only resumed in the next application slot assigned to it. This execution scheme is illustrated on the top-right side of Figure 1. Two applications λ_C and λ_2 execute independently of each other within their allocated blue and white application slots, respectively.

4 EMBEDDED CONTROL SYSTEMS

Control applications regulate the dynamical behaviour of the plant. We deal with a common class of dynamical systems that are modelled as a CT Linear Time-Invariant (LTI) system,

$$\dot{x}(t) = A_c x(t) + B_c u(t), \tag{1}$$

$$y(t) = C_c x(t), \tag{2}$$

where $x(t) \in \mathbb{R}^n$ is the state of the plant, $A_c \in \mathbb{R}^{n\times n}$, $B_c \in \mathbb{R}^{n\times m}$, $C_c \in \mathbb{R}^n$ are the state, input, and output matrices, respectively. $y(t) \in \mathbb{R}^1$ is the output of the plant. $u(t) \in \mathbb{R}^m$ is the *actuation signal* (i.e., signal used by the actuators to be applied to the plant) that is computed by the control law.

4.1 Embedded Execution

A *control task* is the sequence of *sensing* (reading of sensors), *computing* (computation of actuation signals), and *actuating* (writing to actuators) operations. The execution of these operations defines the control task execution time and the sampling interval, which are essential control design parameters.

RIGHTSLINK

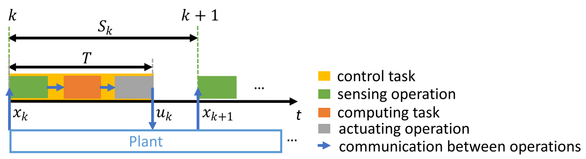


Fig. 2. Timing diagram of the control application.

- **Control task execution time T:** The control task does not execute instantaneously. Each of its operations has a finite execution time and the communication in the *sensor-to-computing*, *computing-to-actuating*, and *actuating-to-plant* paths has also finite execution time. This results in an execution that begins with the sensing operation and terminates at the end of the actuating operation, and it is denoted by T.
- **Sampling interval S_k:** The *sampling interval* is defined as the time between two consecutive sensing operations at samples k and $k+1$ and it is described by $S_k = t_{k+1} - t_k$, where $k \in \mathbb{N}_{\geq 1}$. For the implementation, we make sure that the sampling interval is longer than the control task execution time T, i.e., $S_k > T$.

These concepts are illustrated in Figure 2. In this work, we limit the scope to the delay resulting from the control task computation, since all the operations (i.e., sensing-computing-actuating) are performed within the embedded platform. Our methods can further be generalised to include the delay resulting from the communication between actuators-plant-sensors encountered in a distributed system.

4.2 Plant Discretisation

The plant dynamics, described in Equation (1), are sampled at DT instances t_k with sample index $k \in \mathbb{N}_{\geq 1}$. Thus, the state of the plant can be described in DT as $x_k = x(t_k)$. Additionally, the actuation signal is updated under a Zero-Order Hold (ZOH) actuating scheme as $u(t) = u_k$ for $t \in [t_k + T, t_{k+1} + T)$. The DT system dynamics (with time delay) can be represented by Åström and Wittenmark (1990),

$$x_{k+1} = \sigma x_k + \beta u_k + \gamma u_{k-1}, \tag{3}$$

with

$$\sigma = \Phi(S_k), \quad \beta = \Gamma_3(S_k), \quad \gamma = \Gamma_2(S_k),$$

where

$$\Phi(\tau) = e^{A_C \tau}, \quad \Gamma_1(\tau) = \int_0^\tau \Phi(s) ds B_C, \quad \Gamma_2(\tau) = \Phi(\tau - T)\Gamma_1(T), \quad \Gamma_3(\tau) = \Gamma_1(\tau - T).$$

We define the augmented system state

$$z_k = [\, x_k \quad u_{k-1} \,]', \tag{4}$$

obtaining the augmented higher-order system that can be written in a DT form,

$$z_{k+1} = \hat{A} z_k + \hat{B} u_k = \begin{bmatrix} \sigma & \gamma \\ \mathbf{0} & \mathbf{0} \end{bmatrix} z_k + \begin{bmatrix} \beta \\ \mathbf{I} \end{bmatrix} u_k, \tag{5}$$

$$y_k = \hat{C} z_k = [\, C_c \quad \mathbf{0} \,] z_k, \tag{6}$$

where **I** and **0** are the identity and zero matrices, respectively.

RIGHTSLINK

4.3 Control Performance

The objective of the feedback control application is to control the continuous-time system described in Equation (1) such that the output $y(t) \rightarrow r$ as $t \rightarrow \infty$, where r is the constant input reference signal. The time it takes for the system output $y(t)$ to reach and stay in a close region ($\leq 2\%$) around the reference value r is the *settling time*.

In this work, the control performance is measured in terms of the settling time achieved by a controller, and we define the Quality of Control (QoC) as the inverse of the settling time as follows

$$QoC = \frac{1}{\text{settling time}}, \tag{7}$$

where a shorter settling time leads to a higher QoC. Note that the presented design flows are not restricted to any performance metric and they can be used considering other performance metrics. In Section 8.4, we will consider settling time (i.e., $QoC = 1/\text{settling time}$) and Integral Time-Weighted Absolute Error (ITAE) (i.e., $QoC = 1/\sum t_k |y_k - r|$) metrics. However, since settling time has direct implications on the real-time system behavior, we mainly focus on the results based on the settling time.

4.4 Resource Allocation and Utilisation

Consider a TDM frame of size N slots, where the total TDM frame duration is given by $N(\psi + \omega)$ seconds.

- **Resource allocation:** The periodic execution of the control application λ_C requires evenly distributed allocated slots for an equidistant sampling. Considering the TDM frame of N slots, they can be numbered as $\{0, 1, 2, \ldots, N-1\}$. Thus, the resource allocation for λ_C is given by a sequence $A(\lambda_C) = (a_1, a_2, \ldots, a_M)$, where $a_i \in \{0, 1, 2, \ldots, N-1\}$. The number of slots allocated to λ_C is denoted by $M = |A(\lambda_C)|$ with $M \leq N$. $M = 1$ implies that only one slot is allocated to λ_C and the allocation is periodic with period of N slots. For the cases with $M > 1$, to ensure even distribution of slots, the following conditions are imposed,[1]

$$a_{i+1} > a_i \,\forall\, a_i \in A(\lambda_C), \tag{8}$$

$$a_M - a_{M-1} = a_{M-1} - a_{M-2} = \cdots = a_2 - a_1 = a_1 + N - a_M, \tag{9}$$

$$\mathbf{mod}(N, a_2 - a_1) = 0, \tag{10}$$

  where Equation (8) is given to define the order of the slot allocation, such that each slot allocation a_i is followed by a slot allocated in the future a_{i+1}, and not otherwise. Equation (9) conditions the slots allocation to have a period $a_M - a_{M-1} = a_{M-1} - a_{M-2} = \cdots = a_2 - a_1$. Also, the expression $a_1 + N - a_M$ considers the period from the last allocated slot a_M and the first allocated slot a_1 of the following TDM frame. Finally, the condition in Equation (10) is given to guarantee that N is a multiple integer of the separation between two consecutive allocated slots within the TDM frame and therefore to achieve periodic allocation.
- **Resource utilisation:** Utilisation is referred as the resource the control task of a control application λ_C uses as a fraction of the total TDM frame. This is given by

$$U(\lambda_C) = M \frac{ET}{N(\psi + \omega)} 100\%, \tag{11}$$

  where E is the number of executions of the control task within ψ. $E = 1$ for SR sampling (detailed in Section 5.1) and $E = \lfloor \frac{\psi}{T} \rfloor$ for MR sampling (detailed in Sections 5.1 and 6.1).

[1]**mod** denotes the modulo operator.

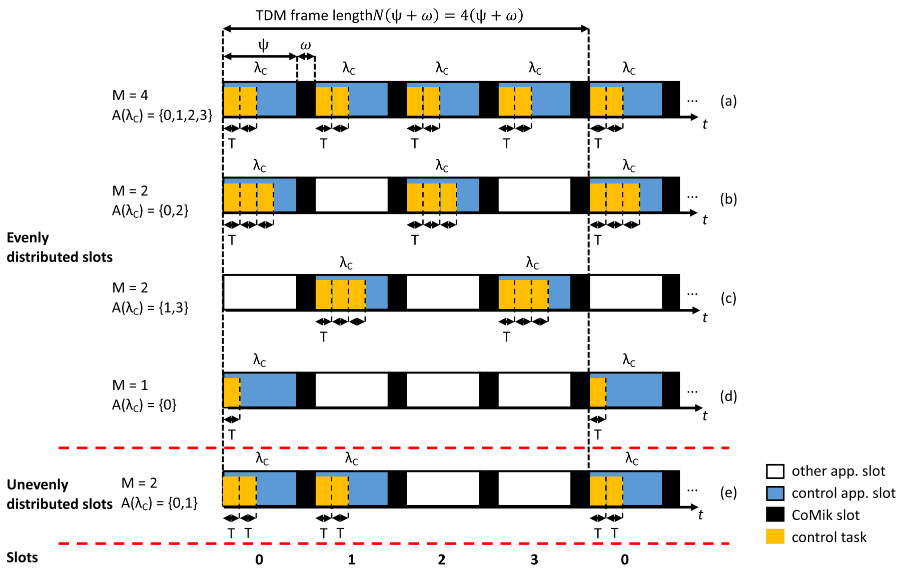


Fig. 3. Resource allocation examples for application λ_C, using a TDM frame with $N = 4$ slots. Top (a, b, c, d): evenly distributed application slots. Bottom (e): unevenly distributed application slots—this allocation is not allowed in the remainder of the article.

Note that λ_C execution is not interrupted within ψ. λ_C is designed such that within ψ the control task runs once or multiple times with execution time T.

In Figure 3, we show examples of resource allocation and utilisation for $N = 4$. Within the white slots any application may run (e.g., multimedia application), but we focus on the blue slots where λ_C executes. In the top of the figure, we present examples of evenly distributed slots. Whereas at the bottom of the figure, we present an example of unevenly distributed slots. We detail these examples (labeled from (a) to (e)) as follows:

- In (a), when all the slots are used $A(\lambda_C) = \{0, 1, 2, 3\}$, $M = 4$, and the control task runs twice within ψ, the resource utilisation $U(\lambda_C) = 4\frac{2T}{\psi+\omega}100\%$.
- In (b) and (c), for allocations $A(\lambda_C) = \{0, 2\}$ and $A(\lambda_C) = \{1, 3\}$, $M = 2$, and when the control task runs three times within ψ, the resource utilisation $U(\lambda_C) = 2\frac{3T}{\psi+\omega}100\%$.
- Alternatively, in (d) a single slot can be allocated to λ_C with $M = 1$, and when the control task runs only once within ψ, $U(\lambda_C) = \frac{T}{\psi+\omega}100\%$.
- In (e), we present an example of unevenly distributed slots allocated to λ_C, where $A(\lambda_C) = \{0, 1\}$, $M = 2$, and the control task runs twice within ψ. From Equations (8)–(10), note that the first and third conditions are met, meaning that the number of slots are correct. However, the second condition is violated with $a_2 - a_1 \neq a_1 + N - a_2$, and the periodicity is violated by not allocating evenly distributed slots.

4.5 Platform-awareness and Its Constraints

We consider the TDM-based execution scheme in the CompSOC platform illustrated in Figure 1, and the control task execution described in Section 4.1 and illustrated in Figure 2. These executions give us the precise timing information that can be used in the design of the controller. This is what we refer to as *platform-awareness* and to compare the SR and MR design flows presented in this article, we have constrained the platform-awareness with the following conditions:

- The number of slots allocated to λ_C is constrained by the conditions in Equations (8)–(10). Such an allocation allows us to directly compare the three design flows, because it allows both SR and MR sampling. It is important to notice that the allocation of slots for MR controllers is not limited to evenly distributed slots but also contiguous and unevenly distributed allocation of slots can be used, as presented in Valencia et al. (2016), where contiguous allocation led to a higher QoC.
- The control task of λ_C always starts running at the beginning of the allocated application slots and its operations (i.e., sensing, computing, actuating) execution do not spread across multiple slots.
- Given λ_C, its control task runs $E = 1$ time for the SR design flow and $E = \lfloor \frac{\psi}{T} \rfloor$ times for the MR design flows within each of its allocated application slots. To that end, we configure $\psi \geq \lfloor \frac{\psi}{T} \rfloor T$.

5 SINGLE-RATE DESIGN FLOW

In this section, we present the platform-aware design flow for feedback controllers whose execution is based on a SR sampling scheme.

5.1 Single Rate Sampling

A SR sampling (equidistant sampling interval) scheme for λ_C is achieved by customising the platform such that it meets the constraints defined in Section 4.5. Recall that for this type of sampling the control task runs only once within ψ. Thus, the single rate sampling interval for λ_C with allocation $A(\lambda_C)$, and a TDM frame with N, M, ψ, and ω is given by

$$h^{SR} = \frac{N}{M}(\psi + \omega). \tag{12}$$

Let us consider an example where the TDM frame is composed of $N = 4$ slots, and allocation with $M = 2$ slots to λ_C (depicted in Figure 4). The SR sampling interval is $h^{SR} = 2(\psi + \omega)$ seconds. At kth sample, the sampling interval is given by $S_k = h^{SR}$.

5.2 Control Design

The design of the SR controller can be done with a classical model-based control methodology (Kuo 1992), e.g., *pole-placement* or *LQR*.

Control law: The control law in this design design flow is updated with a sampling interval h^{SR} and it is of the form

$$u_k = Kz_k + Fr \quad \text{when } S_k = h^{SR}, \tag{13}$$

where K and F are the feedback and feedforward controllers, z_k is defined as per Equation (4), and r is the constant input reference signal.

Closed-loop system dynamics: Given the control law in Equation (13), the closed-loop system dynamics are obtained by using the DT augmented higher-order system from Equation (5) as

$$z_{k+1} = (\hat{A} + \hat{B}K)z_k + \hat{B}Fr. \tag{14}$$

Feedback control gain K: The feedback gain K is designed using the LQR methodology that minimises the DT cost function

$$J = \sum_{k=1}^{\infty}(z_k'\hat{Q}z_k + u_k'\hat{R}u_k), \tag{15}$$

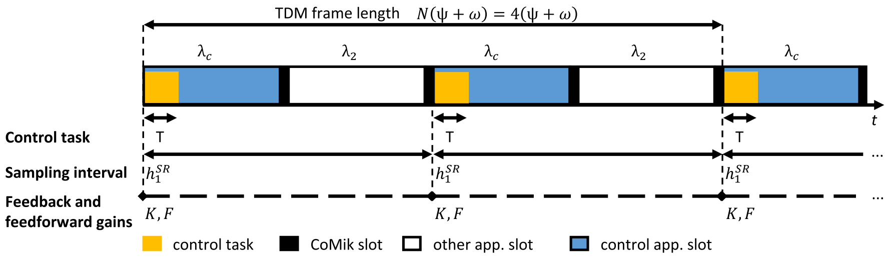


Fig. 4. SR platform-aware sampling scheme for λ_C with allocation $A(\lambda_C) = \{0, 2\}$.

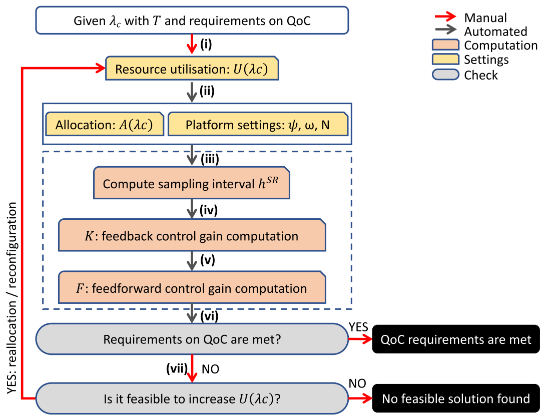

Manual
Automated
Computation
Settings
Check
Given λ_c with T and requirements on QoC
(i)
Resource utilisation: $U(\lambda c)$
(ii)
Allocation: $A(\lambda c)$
Platform settings: ψ, ω, N
(iii)
Compute sampling interval h^{SR}
(iv)
K: feedback control gain computation
(v)
F: feedforward control gain computation
(vi)
Requirements on QoC are met?
YES
QoC requirements are met
(vii)
NO
Is it feasible to increase $U(\lambda c)$?
No feasible solution found
YES: reallocation / reconfiguration

Fig. 5. SR design flow. The control design is highlighted within the dashed box.

with $\hat{Q} > 0$ and $\hat{R} > 0$, the DT state and control weighting matrices of the LQR[2]. Having large $\hat{Q}$ compared to $\hat{R}$ puts emphasis on making the state small possibly at the cost of large actuation signals but potentially leading to a short settling time. By increasing $\hat{R}$, large actuation signals are penalised, typically leading to a slower response. To minimise the settling time and maximise the QoC as per Equation (7) an LQR tuning is used to find the values of $\hat{Q}$ and $\hat{R}$. In this work, the PSO algorithm of Medina et al. (2017) has been used. This algorithm explores the tuning of the $\hat{Q}$ and $\hat{R}$ matrices as a parallel problem. It defines a swarm population composed of a finite number of individuals (i.e., random values that set the contents of the $\hat{Q}$ and $\hat{R}$ matrices). Each individual moves according to a velocity that is determined in every iteration by a random component, a personal best position of each individual, and a global best position of the swarm. Thus, in each iteration the feedback control gain K is designed and the QoC of the controller is evaluated.

Feedforward control gain F: The feedforward gain F is computed, for the closed-loop dynamic described in Equation (14), by following the design in Hellerstein et al. (2004).

5.3 Design Flow

We propose the design flow shown in Figure 5,[3] which is composed of seven parts. (i) λ_C requirement on QoC and T shape the resource utilisation $U(\lambda_C)$. (ii) The platform settings (Section 3) and the resource allocation (Section 4.5) can be derived from the targeted resource utilisation. (iii) From

[2]The inequality ($\vartheta \geq 0$) $\vartheta > 0$ means that the matrix ϑ is symmetric and positive (semi-)definite.
[3]Note that there are manual and automated parts in all the design flows of this article. Manual parts that involve *reallocation* and *reconfiguration* comprise a new resource utilisation. Automated parts belong to one simulation program.

RIGHTSLINK Copyright Clearance Center

there, the SR sampling interval is computed (Section 5.1). (iv) The feedback control gain K is computed by using the PSO (Section 5.2). (v) The static feedforward control gain F is computed. (vi) The requirement on QoC is evaluated and if it is met, the design flow ends. (vii) If the QoC requirement is not met, then the feasibility of varying the resource utilisation $U(\lambda_C)$ is verified. If the resource utilisation can be modified, then one can either reallocate more slots to the application (i.e., increasing M as long it is $M < (N - \#$ of other applications running in the platform$)$) or change platform settings (i.e., ψ, ω, N), to derive new timings for the execution of λ_C (e.g., sampling interval h^{SR}). Otherwise, if the resource utilisation cannot be modified, then no feasible solution can be found with this design flow on this platform.

Example 5.1. Considering the configuration $\omega = 40.96\mu$s, $\psi = 2.95904$ms, $N = 10$, $M = 10$, and $T = 0.99$ms. $U(\lambda_C) = 33\%$ with sampling interval $h^{SR} = 3$ ms leading to a $QoC = 66.67$ [1/s].

6 MULTI-RATE LOCAL OPTIMAL DESIGN FLOW

In this section, we present the platform-aware design flow for feedback controllers whose execution is based on a MR sampling scheme and their performance is optimised for their nominal sampling interval.

6.1 MR Sampling

A MR sampling (finite and periodic sequence of non-equidistant sampling intervals) scheme for λ_C is achieved by customising the platform such that it meets the constraints defined in Section 4.5. Recall that for this type of sampling the control task runs $\lfloor \frac{\psi}{T} \rfloor$ times within ψ. For a given slot allocation within a TDM frame, λ_C executes according to a finite and periodic sequence of sampling intervals h_j^{MR} where $j \in \{1, 2\}$. The sampling interval h_1^{MR} occurs $\lfloor \frac{\psi}{T} \rfloor - 1$ times within each ψ, whereas h_2^{MR} occurs only once between two consecutive allocated ψ. The duration of both sampling intervals is given by

$$h_1^{MR} \geq T, \tag{16}$$

$$h_2^{MR} = h_1^{MR} + \left(\psi - \left\lfloor \frac{\psi}{h_1^{MR}} \right\rfloor h_1^{MR}\right) + \left(\frac{N}{M} - 1\right)\psi + \frac{N}{M}\omega, \tag{17}$$

where the variation of T is very small (i.e., due to the interference-free characteristics of the CompSOC platform) and $T \leq h_1^{MR} < \psi$. h_2^{MR} is equal to the summation of the last h_1^{MR} sampling interval within ψ, the remaining time within ψ that is given by $\psi - \lfloor \frac{\psi}{h_1^{MR}} \rfloor h_1^{MR}$, and the time between two consecutive allocated application slots that is given by $(\frac{N}{M} - 1)\psi + \frac{N}{M}\omega$.

In Figure 6, we illustrate the MR sampling with an example where $N = 4$, $M = 2$, and $A(\lambda_C) = \{0, 2\}$. Within each ψ, the control task runs $\lfloor \frac{\psi}{T} \rfloor = 3$ times. $h_1^{MR} \geq T$ and $h_2^{MR} = h_1^{MR} + (\psi - 3h_1^{MR}) + \psi + 2\omega$. h_1^{MR} is the nominal sampling interval, since it occurs more frequently, whereas is h_2^{MR} is the longer and less frequently occurring sampling interval. At the kth sample, the sampling interval is given by $S_k = h^{MR}$, where $h^{MR} \in \{h_1^{MR}, h_2^{MR}\}$.

6.2 Control Design

The design of the MRLO controller exploits the frequent runs of the control task within ψ to achieve a high performance. Essentially, we optimise the performance of an independent controller designed for the nominal sampling interval h_1^{MR} using the SR design flow detailed in Section 5. To guarantee that the system is stable during runtime, when the system runs between the periodically non-equidistant sampling intervals, we use a Lyapunov-based design that includes the controller designed for h_1^{MR} to obtain the controller for the sampling interval h_2^{MR}.

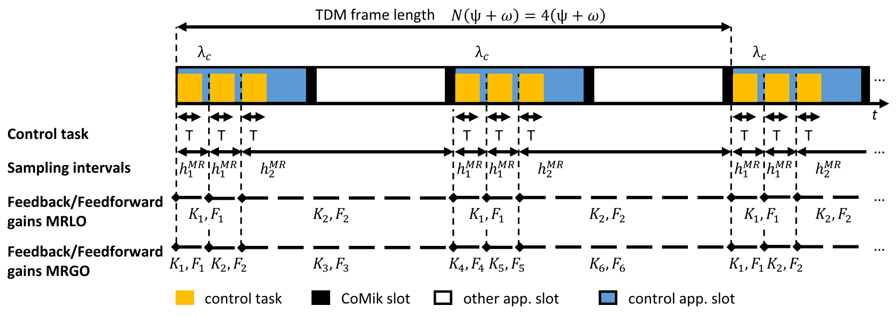


Fig. 6. MR platform-aware sampling scheme for λ_C with $A(\lambda_C) = \{0, 2\}$. The sampling intervals are h_1^{MR} and h_2^{MR}. For MRLO: The periodic sequence of non-equidistant sampling intervals along one TDM frame is given by $(h_1^{MR}, h_1^{MR}, h_2^{MR}, h_1^{MR}, h_1^{MR}, h_2^{MR})$. For MRGO Section 7.1: The periodic sequence of non-equidistant sampling intervals is given by $H = (h_1^{MR}, h_1^{MR}, h_2^{MR}, h_1^{MR}, h_1^{MR}, h_2^{MR})$.

Control law: The control law changes with the sampling intervals h_1^{MR} and h_2^{MR}, and it is given by

$$u_j = K_j z_k + F_j r \quad \text{when } S_k = h_j^{MR}. \tag{18}$$

Closed-loop system dynamics: The closed-loop dynamics depend on the sampling intervals and the control law. It is obtained using the DT augmented higher-order system dynamics in Equation (5). With sampling interval h_j^{MR} and the control law in Equation (18), the closed-loop dynamics is given by

$$z_{k+1} = (\hat{A}_j + \hat{B}_j K_j) z_k + \hat{B}_j F_j r, \tag{19}$$

where $\hat{A}_j$ and $\hat{B}_j$ are the DT augmented system matrices for the sampling interval h_j^{MR}.

Switching behaviour: The sampling intervals switch between h_1^{MR} and h_2^{MR}. Thus, the closed-loop dynamics switch between the two systems $(\hat{A}_1 + \hat{B}_1 K_1) z_k + \hat{B}_1 F_1 r$ and $(\hat{A}_2 + \hat{B}_2 K_2) z_k + \hat{B}_2 F_2 r$ according to the order of the periodic sequence of non-equidistant sampling intervals. Stability of this switched system is governed by the feedback gains K_1 and K_2. Note that the feedforward gains do not influence the stability of the overall system. Therefore, for the stability analysis, we consider the system matrices, $\alpha_1 = \hat{A}_1 + \hat{B}_1 K_1$ and $\alpha_2 = \hat{A}_2 + \hat{B}_2 K_2$. For the example in Figure 6, the switching sequence is given by $\alpha_1 \rightarrow \alpha_1 \rightarrow \alpha_2 \cdots$.

Nominal sampling interval: The focus of this controller design flow is to optimise the controller QoC by locally optimising the performance of the control gain that is designed for the nominal sampling interval. It is important to configure $\psi \geq \lfloor \frac{\psi}{T} \rfloor T$, such that the control task runs multiple times within ψ as depicted with the example in Figure 6, where $\psi \geq 3T$ leading to a nominal sampling interval h_1^{MR} that is repeated $\lfloor \frac{\psi}{T} \rfloor - 1 = 2$ times within ψ.

Nominal feedback control gain K_1: The nominal feedback control gain K_1 is computed for the nominal sampling interval h_1^{MR}, following the methodology described in Section 5.2.

Switching feedback control gain K_2: To guarantee the stability of the overall system under the switching behaviour explained above, we perform the DT *Lyapunov* stability test to find a Common Quadratic Lyapunov Function (CQLF) $P \in \mathbb{R}^{n \times n}$, such that the LMIs $P \succ 0$, $\alpha_1' P \alpha_1 - P \prec 0$, and $\alpha_2' P \alpha_2 - P \prec 0$ are feasible. With these LMIs, we evaluate the stability of the system as well as compute the switching feedback control gain K_2 using the following procedure. First, by using α_1 and α_2 on the LMIs, we solve for K_2. However, this leads to non-linear matrix inequalities (i.e., leads to a term where K_2 is multiplied by P). To solve this, we rewrite the LMIs by using the *Schur*

RIGHTSLINK

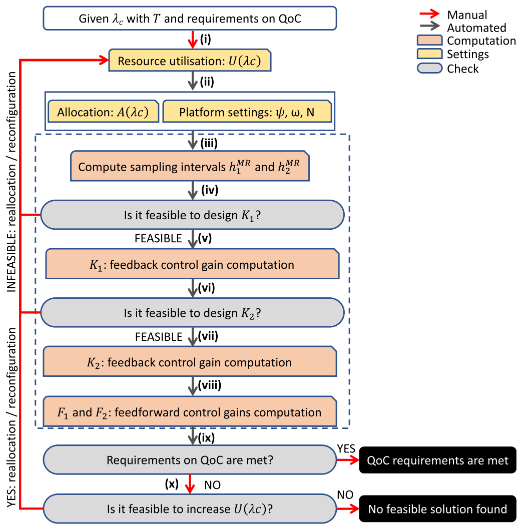

Manual
Automated
Computation
Settings
Check
Given λ_c with T and requirements on QoC
(i)
Resource utilisation: $U(\lambda c)$
(ii)
Allocation: $A(\lambda c)$
Platform settings: ψ, ω, N
(iii)
Compute sampling intervals h_1^{MR} and h_2^{MR}
(iv)
Is it feasible to design K_1?
FEASIBLE (v)
K_1: feedback control gain computation
(vi)
Is it feasible to design K_2?
FEASIBLE (vii)
K_2: feedback control gain computation
(viii)
F_1 and F_2: feedforward control gains computation
(ix)
Requirements on QoC are met?
YES
QoC requirements are met
(x) NO
Is it feasible to increase $U(\lambda c)$?
NO
No feasible solution found
INFEASIBLE: reallocation / reconfiguration
YES: reallocation / reconfiguration

Fig. 7. MRLO design flow. The control design is highlighted within the dashed box.

complement (Crabtree and Haynsworth 1969) as follows:

$$\begin{bmatrix} -P & (\hat{A}_1 + \hat{B}_1 K_1)' \\ (\hat{A}_1 + \hat{B}_1 K_1) & -P^{-1} \end{bmatrix} \prec 0, \quad \begin{bmatrix} -P & (\hat{A}_2 + \hat{B}_2 K_2)' \\ (\hat{A}_2 + \hat{B}_2 K_2) & -P^{-1} \end{bmatrix} \prec 0. \tag{20}$$

To resolve the above non-linearity, we use a variable substitution by defining $Y = P^{-1}$, where $Y \in \mathbb{R}^{n \times n}$, and we pre- and post-multiply by the linearisation operator $\mathbf{diag}(Y, \mathbf{I})$ to obtain[4]

$$\begin{bmatrix} -Y & Y(\hat{A}_1 + \hat{B}_1 K_1)' \\ (\hat{A}_1 + \hat{B}_1 K_1)Y & -Y \end{bmatrix} \prec 0, \quad \begin{bmatrix} -Y & Y\hat{A}_2' + YK_2'\hat{B}_2' \\ \hat{A}_2 Y + \hat{B}_2 K_2 Y & -Y \end{bmatrix} \prec 0. \tag{21}$$

Finally, we define

$$K_2 = WY^{-1}, \tag{22}$$

where $W \in \mathbb{R}^n$. Thus, the LMIs are reformulated as

$$\begin{bmatrix} -Y & Y(\hat{A}_1 + \hat{B}_1 K_1)' \\ (\hat{A}_1 + \hat{B}_1 K_1)Y & -Y \end{bmatrix} \prec 0, \quad \begin{bmatrix} -Y & Y\hat{A}_2' + W'\hat{B}_2' \\ \hat{A}_2 Y + \hat{B}_2 W & -Y \end{bmatrix} \prec 0, \tag{23}$$

and if there exist matrices Y and W, the system is stable with the switching between the systems α_1, and α_2, and K_2 is given by Equation (22).

Feedforward control gains F_1 and F_2: The feedforward gains F_1 and F_2 are computed for the closed-loop dynamics α_1 and α_2 following the design in Hellerstein et al. (2004).

6.3 Design Flow

We propose the design flow shown in Figure 7, which is composed of ten parts. (i) λ_C requirement on QoC and T shape the resource utilisation $U(\lambda_C)$. (ii) The platform settings (Section 3) and the resource allocation (Section 4.5) can be derived from the targeted resource utilisation. (iii) From there, the MR sampling intervals are computed (Section 6.1). (iv) If it is feasible to design the

[4]**diag**(a,b, ...) operator returns a diagonal matrix with the elements of the matrices a, b, ... are placed on the main diagonal.

feedback control gain K_1, then the design flow continues to part (v) or it goes to part (ii) otherwise (i.e., a different resource utilisation is selected to vary the nominal sampling interval). (v) The feedback control gain K_1 is computed by using the PSO (Section 5.2). (vi) Later, we evaluate the feasibility of the LMIs to guarantee stability using K_1 as per Equation (23). If the solution is infeasible, then it is necessary to modify the resource utilisation. That is, reallocate resources or change platform settings to derive new timings for λ_C. (vii) If the solution is feasible, then the feedback control gain K_2 is computed as per Equation (22). (viii) The static feedforward control gains F_1 and F_2, are computed. (ix) The requirement on QoC is evaluated and if it is met, the design flow ends. (x) If the QoC requirement is not met, then the feasibility of varying the resource utilisation $U(\lambda_C)$ is verified. If the resource utilisation can be modified, then one can either reallocate more slots to the application (i.e., increasing M as long it is $M < (N - \#$ of other applications running in the platform$)$) or change platform settings (i.e., ψ, ω, N), to derive new timings for the execution of λ_C (e.g., sampling interval h_1^{MR} and h_2^{MR}). Otherwise, if the resource utilisation cannot bet modified, no solution can be found with this design flow on this platform.

Unlike the SR design flow in Section 5.3, the control gains K_1 and K_2 cannot be freely designed. Since K_1 is optimised for performance, this might be an aggressive control gain that might lead to an infeasible solution to compute K_2.

Example 6.1. Considering the same configuration of Example 5.1: $\omega = 40.96\mu$s, $\psi = 2.95904$ms, $N = 10$, $M = 10$, and $T = 0.99$ms. $U(\lambda_C) = 66\%$ with sampling intervals $h_1^{MR} = 1$ ms and $h_2^{MR} =$ 2ms leading to $QoC = 142.86[1/s]$.

7 MULTI-RATE GLOBAL OPTIMAL DESIGN FLOW

In this section, we present the platform-aware design flow for feedback controllers whose execution is based on MR sampling. Their design offers a stabilising solution that transforms the switching MR system to a classic LQR control design problem, where heuristics are used to find the tuning parameters of the LQR to achieve high QoC.

7.1 MR Sampling

A MR sampling (finite and periodic sequence of non-equidistant sampling intervals) scheme for λ_C is achieved by customising the platform such that it meets the constraints defined in Section 4.5. Recall that for this type of sampling the control task runs $\lfloor \frac{\psi}{T} \rfloor$ times within ψ, with finite and periodic sequence of sampling intervals h_j^{MR} where $j \in \{1, 2\}$. This sequence is represented by the tuple

$$H = \left(h_1^{MR}, \ldots, h_1^{MR}, h_2^{MR}\right)^{\frac{N}{M}}, \tag{24}$$

where the number of sampling intervals h_j^{MR} in H is denoted by $\rho = \lfloor \frac{\psi}{T} \rfloor \frac{N}{M}$. $H(i)$ denotes the ith sampling interval in H, where $1 \leq i \leq \rho$. In this tuple h_1^{MR} occurs $\lfloor \frac{\psi}{T} \rfloor - 1$ times within each ψ, whereas h_2^{MR} occurs only once between two consecutive allocated ψ. The duration of both sampling intervals can be calculated as per Equations (16) and (17) from Section 6.1.

In Figure 6, we illustrate the MR sampling with an example where $N = 4$, $M = 2$, $A(\lambda_C) = \{0, 2\}$, and $\rho = 6$. Within each ψ, the control task executes $\lfloor \frac{\psi}{T} \rfloor = 3$ times. Thus, $h_1^{MR} > T$ and $h_2^{MR} = h_1^{MR} + (\psi - 3h_1^{MR}) + \psi + 2\omega$. The elements of the tuple are defined as $H(1) = h_1^{MR}$, $H(2) = h_1^{MR}$, $H(3) = h_2^{MR}$, $H(4) = h_1^{MR}$, $H(5) = h_1^{MR}$, and $H(6) = h_2^{MR}$. At the kth sample, the sampling interval is given by $S_k = H(i)$.

RIGHTSLINK

7.2 Control Design

The MRGO controller is designed to find a solution for the overall MR switched system to achieve high performance. Our technique transforms the overall MR control design problem to the classical LQR design by using a time-lifted reformulation.

Control law: The control law is given by

$$u_i = K_i z_k + F_i r, \tag{25}$$

where K_i and F_i are feedback and feedforward gains used when the sampling interval $S_k = H(i)$. Thus, there are ρ combinations of (K_i, F_i). This is illustrated in Figure 6, where $\rho = 6$ combinations of these control gains are used sequentially and according to the order of sampling intervals in the tuple H.

Closed-loop system dynamics: Closed-loop system considering the DT augmented higher-order system dynamics in Equation (5) and control law in Equation (25) is given by

$$z_{k+1} = (\hat{A}_i + \hat{B}_i K_i) z_k + \hat{B}_i F_i r, \tag{26}$$

where $\hat{A}_i = \hat{A}$ and $\hat{B}_i = \hat{B}$ are the augmented state and input matrices for the sampling interval $S_k = H(i)$.

Switching behaviour: Since the sampling intervals periodically repeat according to the order in H, the resulting DT system dynamics periodically switch between ρ closed-loop dynamics given by Equation (26).

Control problem: Let us first define the DT representation of the cost by

$$J = \sum_{k=1}^{\infty} \int_{t_k}^{t_{k+1}} \begin{bmatrix} x(s) \\ u(s) \end{bmatrix}' \begin{bmatrix} Q_c & \mathbf{0} \\ \mathbf{0} & R_c \end{bmatrix} \begin{bmatrix} x(s) \\ u(s) \end{bmatrix} ds = \sum_{k=1}^{\infty} z_k' \hat{Q}_i z_k + u_k' \hat{R}_i u_k, \tag{27}$$

where Q_c and R_c are the CT state and control weighting matrices, respectively. The DT state and control weighting matrices are represented by $\hat{Q}_i$ and $\hat{R}_i$, respectively (Valencia et al. 2016).

The control problem can now be formulated as follows: Given $z_1 = [x_1' \, u_0']'$

$$J^\star(z_1) = \min_{\{u_k\}}(J), \quad \text{subject to system in Equation (5)},$$

where this problem does not have a closed-form solution for arbitrary t_k. However, on the CompSOC platform, the sampling intervals occur in the periodic sequence H. Hence, the set of possible $\hat{A}_i$ and $\hat{B}_i$ can be pre-computed and result in a DT Linear Periodically Time-Varying (LPTV) system for which the control problem can be solved using periodic Riccati equations.

Time-lifted Reformulation: For a DT-LPTV system with ρ sampling intervals, the dynamics and cost have the periodicity property

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

With a TDM period index δ, the time-lifted reformulation (Bittanti and Colaneri 2008)

$$z_{(\delta+1)\rho+1} = \tilde{A} z_{\delta\rho+1} + \tilde{B} \bar{u}_\delta, \quad \delta \in \mathbb{N}_{\geq 0}, \tag{28}$$

gives the dynamics over one TDM period, where[5]

$$\tilde{A} = \left[\prod_{l=\rho}^{1} \hat{A}_i \right], \quad \tilde{B} = \left[\left[\prod_{l=\rho}^{2} \hat{A}_i \right] \hat{B}_1 \quad \left[\prod_{l=\rho}^{3} \hat{A}_i \right] \hat{B}_2 \quad \cdots \quad \hat{A}_\rho \hat{B}_{\rho-1} \quad \hat{B}_\rho \right],$$

[5] $\prod_{l=\rho}^{1} A_i$ denotes (for $\rho \geq 1$) the multiplication $A_\rho A_{\rho-1} \cdots A_2 A_1$.

and the cost can be written as

$$J = \lim_{p\to\infty} \sum_{\delta=0}^{p} \bar{z}_\delta' \bar{Q} \bar{z}_\delta + \bar{u}_\delta' \bar{R} \bar{u}_\delta = \lim_{q\to\infty} \sum_{\delta=0}^{q} z_{\delta\rho+1}' \tilde{Q} z_{\delta\rho+1} + \bar{u}_\delta' \tilde{R} \bar{u}_\delta, \tag{29}$$
$$\tilde{Q} = \bar{A}'\bar{Q}\bar{A},\ \tilde{R} = \bar{R} + \bar{B}'\bar{Q}\bar{B},$$

using the augmented variables

$$\bar{z}_\delta = \begin{bmatrix} z_{\delta\rho+1} \\ \vdots \\ z_{\delta\rho+\rho} \end{bmatrix},\ \bar{u}_\delta = \begin{bmatrix} u_{\delta\rho+1} \\ \vdots \\ u_{\delta\rho+\rho} \end{bmatrix},\ \begin{array}{l} \bar{Q} = \mathrm{diag}(\hat{Q}_i), \\ \bar{R} = \mathrm{diag}(\hat{R}_i), \end{array}$$

and using the dynamics within the TDM period

$$\bar{z}_\delta = \bar{A} z_{\delta\rho+1} + \bar{B}\bar{u}_\delta, \tag{30}$$

where

$$\bar{A} = \begin{bmatrix} \mathbf{I} \\ \hat{A}_1 \\ \hat{A}_2\hat{A}_1 \\ \vdots \\ \prod_{l=\rho-1}^{1} \hat{A}_i \end{bmatrix},\quad \bar{B} = \begin{bmatrix} \mathbf{0} & & \cdots & & \mathbf{0} \\ \hat{B}_1 & \mathbf{0} & & & \mathbf{0} \\ \hat{A}_2\hat{B}_1 & \hat{B}_2 & \mathbf{0} & & \mathbf{0} \\ \vdots & & \ddots & \ddots & \vdots \\ \left[\prod_{l=\rho-1}^{2} \hat{A}_i\right]\hat{B}_1 & \left[\prod_{l=\rho-1}^{3} \hat{A}_i\right]\hat{B}_2 & \cdots & \hat{B}_{\rho-1} & \mathbf{0} \end{bmatrix}$$

with $l \in \mathbb{N}_{[1,\rho]}$.

Feedback control gains K_i: Note that the matrices $(\tilde{A}, \tilde{B}, \tilde{Q}, \tilde{R})$ in Equations (28) and (29) do not depend on the TDM period index δ, i.e., they are time-invariant. The lifted problem thus has the standard time-invariant DT LQR form, which can be solved efficiently. For this lifted reformulation standard optimal control methods can find the optimal solution (Åström 1970; Bertsekas 2005),

$$J^\star(z_1) = z_1'\tilde{P}z_1, \tag{31}$$

where $\tilde{P}$ is the unique positive definite solution to the Discrete-Time Algebraic Riccati Equation (DARE),

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

Furthermore, $V(z) = z'\tilde{P}z$ is a Lyapunov function that ensures stability for the optimal control actions

$$\bar{u}_\delta = \tilde{K} z_{\delta\rho+1}, \tag{32}$$
$$\tilde{K} = -(\tilde{R} + \tilde{B}'\tilde{P}\tilde{B})^{-1}(\tilde{B}'\tilde{P}\tilde{A}). \tag{33}$$

This is the solution to the lifted problem over one TDM period, and can be transformed into a state feedback for the original DT-Linear Time Variant (LTV) system described with Equation (25).

From $P_{\rho+1} = \tilde{P}$, the solutions P_i can be found from the solution to the standard finite horizon Discrete-Time Dynamic Riccati Equation (DDRE),

$$P_i = \hat{A}_i'P_{i+1}\hat{A}_i + \hat{Q}_i - (\hat{B}_i'P_{i+1}\hat{A}_i)'(\hat{R}_i + \hat{B}_i'P_{i+1}\hat{B}_i)^{-1}(\hat{B}_i'P_{i+1}\hat{A}_i),$$

which represents the Discrete-Time Periodic Riccati Equation (DPRE) when $P_{i+\rho} = P_i$ (Varga 2008). The feedback control gains K_i are computed by

$$K_i = -(\hat{R}_i + \hat{B}_i'P_{i+1}\hat{B}_i)^{-1}(\hat{B}_i'P_{i+1}\hat{A}_i). \tag{34}$$

RIGHTSLINK

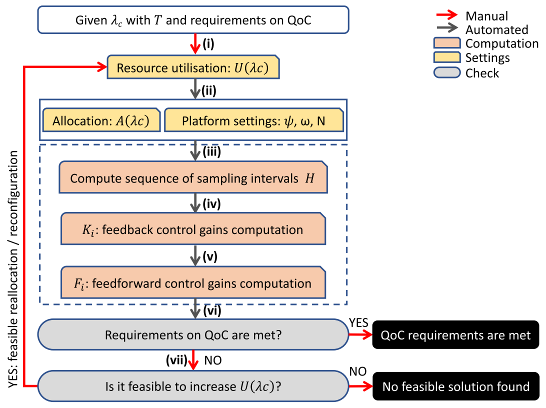

Given λ_c with T and requirements on QoC
(i)
Resource utilisation: $U(\lambda c)$
(ii)
Allocation: $A(\lambda c)$
Platform settings: ψ, ω, N
(iii)
Compute sequence of sampling intervals H
(iv)
K_i: feedback control gains computation
(v)
F_i: feedforward control gains computation
(vi)
Requirements on QoC are met?
YES
QoC requirements are met
(vii) NO
Is it feasible to increase $U(\lambda c)$?
NO
No feasible solution found
YES: feasible reallocation / reconfiguration
Manual
Automated
Computation
Settings
Check

Fig. 8. MRGO design flow. The control design is highlighted within the dashed box.

Feedforward control gains F_i: The feedforward gains are computed for the closed-loop dynamic $\alpha_{(k,i)}$ following the design in Hellerstein et al. (2004).

Q_c and R_c matrices initialisation: In this design flow, we have shown that the CT cost function is translated into its DT equivalent. This cost function is further extended for the time-lifted reformulation of the DT-LPTV representation of the system that switches between the sampling intervals in H. This means that we need to initialise the CT Q_c and R_c matrices or the DT $\hat{Q}_i$ and $\hat{R}_i$ matrices, and evaluate their impact on the QoC. To initialise the values of the Q_c and R_c matrices, we use a heuristic that consists of choosing the diagonal values of these matrices, such that the control performance is optimised in DT. To improve the design flow it would be necessary to automate the optimisation of the $\hat{Q}_i$ and $\hat{R}_i$ matrices for higher QoC. However, this procedure it not investigated in this work.

7.3 Design Flow

We propose the design flow shown in Figure 8, which is composed of seven parts. (i) λ_C requirement on QoC and T shape the resource utilisation $U(\lambda_C)$. (ii) The platform settings (Section 3) and the resource allocation (Section 4.5) can be derived from the targeted resource utilisation. (iii) From there, the periodic sequence H of non-equidistant sampling intervals is computed (Section 7.1). (iv) Thus, the feedback control gains K_i are computed as per Equation (34) following the design in Section 6.2. (v) The static feedforward control gains F_i are computed. (vi) The requirement on QoC is evaluated and if it is met, the design flow ends. (vii) If the QoC requirement is not met, then the feasibility of varying the resource utilisation $U(\lambda_C)$ is verified. If the resource utilisation can be modified, then one can either reallocate more slots to the application (i.e., increasing M as long it is $M < (N - \#$ of other applications running in the platform)) or change platform settings (i.e., ψ, ω, N), to derive new timings for the execution of λ_C (e.g., sampling intervals h_1^{MR} and h_2^{MR}). Otherwise, if the resource utilisation cannot bet modified, then no solution can be found with this design flow on the platform.

Unlike the SR and MRLO design flows in Sections 5.3 and 7.3, none of the control gains K_i can be freely designed. In this particular design flow, the design of the K_i gains are subjected to the cost function in Equation (29).

Example 7.1. Considering the same configuration of Example 5.1: $\omega = 40.96\mu s$, $\psi = 2.95904$ms, $N = 10$, $M = 10$, and $T = 0.99$ms. $U(\lambda_C) = 66\%$ with sampling intervals $h_1^{MR} = 1$ ms and $h_2^{MR} = 2$ms leading to $QoC = 250[1/s]$.

RIGHTSLINK

8 EXPERIMENTAL STUDY

In this section, we present the experimental study we have carried out to evaluate the design flows presented in Sections 5 to 7. We will dive into details of the plant that we used, the MATLAB and HIL experimets, the trade-off analysis derived from two design considerations: QoC and $U(\lambda_C)$. Moreover, we will evaluate the impact of the platform settings on QoC, and finally we present design guidelines.

8.1 Platform Configuration

The allocation of slots to applications, N, ψ, and ω, are defined at design time. ω is fixed at 40.96μs, whereas ψ is application dependent and we have set it up with values in the range of 3–30ms for the experiments reported in this article. The control task execution time is measured during runtime and it is $T = 0.99$ms.

The customisation of the platform settings has a strong influence on the control task execution time. On the one hand, choosing $\psi = T$ only allows for the SR design flow implementation, because the control task will execute once within ψ. This implies that if T is short, ψ will be also short. Thus, $\psi \rightarrow \omega$, and therefore the resource utilisation $U(\lambda_C)$ goes down significantly, because more resources will be used by the CoMik microkernel. On the other hand, choosing $\psi \gg T$ allows for both SR and MR design flows implementation. If the SR design flow is used in this scenario, then the resource utilsiation will decrease, because the control task will run once within ψ and the remaining available processing time within ψ will be unused. In essence, the SR design flow has a lower resource utilisation and this is where it falls short. To avoid this lower resource utilisation, we have used MR design flows. They use longer ψ (i.e., to avoid $\psi \rightarrow \omega$) and ψ is used as much as possible (i.e., $\lfloor \frac{\psi}{T} \rfloor$ times).

8.2 Setup: Motion Control System

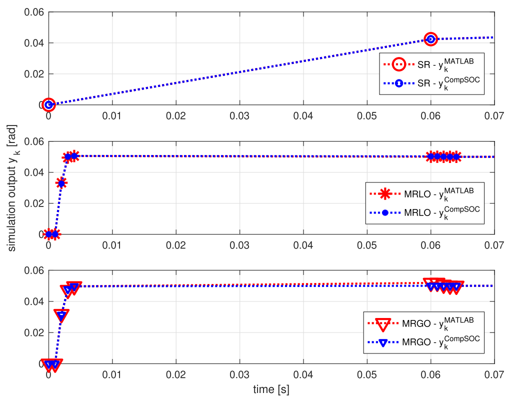
We consider a motion control system that is composed of a mechanical setup, an electrical circuit for actuation, and an embedded platform. The mechanical setup is composed of two masses connected to each other by a flexible bar. The motor is connected directly to one of the masses, and two encoders measure the rotation in each mass (Geelen et al. 2016). The electrical circuit converts the digital actuation signal to an analog input that is applied to the plant. An instance of the CompSOC platform (depicted in Figure 1) where λ_C executes, is synthesised on a Xilinx ML605 Virtex6 FPGA-based development kit (Xilinx 2018).

The mechanical and electrical circuits are described with the CT LTI model from Equation (1), where the state of the plant $x(t) = [\,\theta_1\ \theta_2\ \omega_1\ \omega_2\,]'$ is composed of angular positions (θ_1, θ_2), angular velocities (ω_1, ω_2), and the state and input matrices are defined by (adapted from the model presented in Geelen et al. (2016))

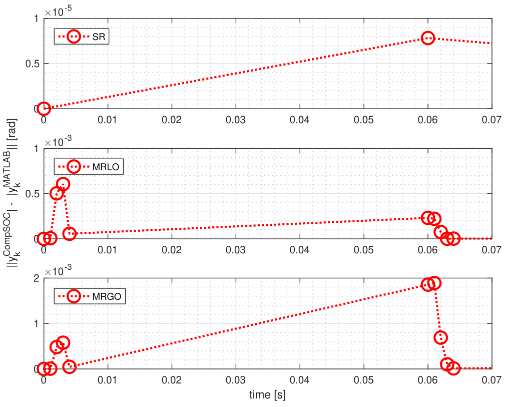
$$A_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1.0886\text{e}{+}7 & 1.0886\text{e}{+}7 & -1.9740\text{e}{+}3 & 1.4740\text{e}{+}3 \\ 1.0886\text{e}{+}7 & -1.0886\text{e}{+}7 & 1.4740\text{e}{+}3 & -1.9740\text{e}{+}3 \end{bmatrix},\ B_c = \begin{bmatrix} 0 \\ 0 \\ 997450 \\ 0 \end{bmatrix},\ C_c = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}'. \tag{35}$$

8.3 MATLAB and HIL Experiments

8.3.1 MATLAB Experiments. The MATLAB simulation is essential to verify the correct functionality of the control gains that have been designed for the previously described design flows. This experiment consists of the following steps. (i) Control design of the feedback and feedforward gains for the respective design flow (i.e., SR, MRLO, or MRGO). (ii) Simulation of the DT system by



(a) Simulation output of the MATLAB and HIL experiments.



(b) Simulation error between the MATLAB and HIL experiments.

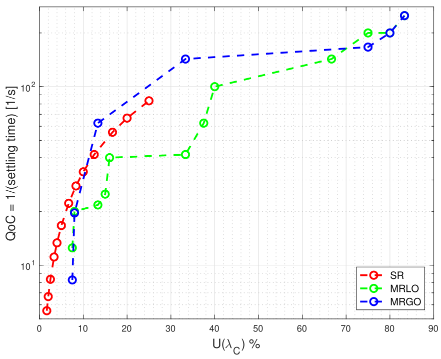
Fig. 9. MATLAB and HIL experiments comparison for the DT system output y_k.

calculating the states at each sample k, where the states depend on the selected design flow (i.e., Equation (14) for SR, Equation (19) for MRLO, and Equation (26) for MRGO). (iii) Update the sampling interval S_k at each sample k depending on the design flow (i.e., $S_k = h^{SR}$ for SR, $S_k = h^{MR}$ for MRLO, and $S_k = H(i)$ for MRGO).

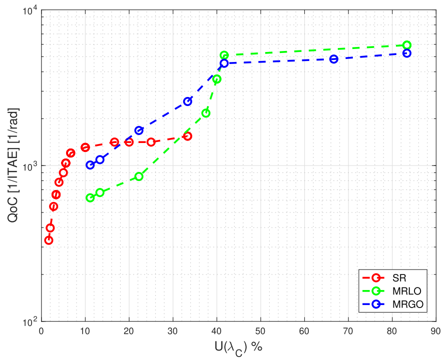
8.3.2 HIL Experiments. The HIL experiment allows for the validation of the control design flows. These are validated by implementing the controller and emulating the plant dynamics on independent processor tiles. Hence, there is an exchange of electrical signals between the controller and the plant (Karpenko and Sepehri 2006; Ogan 2015; Palladino et al. 2009; Truong 2012). We built a HIL experiment using one instance of the CompSOC platform with the architecture shown in Figure 1. One processor tile runs the control application under a TDM-based execution scheme. The other processor tile runs the DT plant dynamics at a high frequency ($\gg T$) to emulate the CT behaviour of the plant.

The HIL experiment can be divided in two parts. (i) Control application implementation, where we simulate the sensing and actuating operations as read and write operations of the system states and actuation signals to and from off-chip memory locations. In between these operations the computation of actuation signals (u_k, Equation (13); $u_{1,2}$, Equation (18); and u_i, Equation (25)) is done by using the control gains calculated off-line by the design flows described in Sections 5–7, respectively. (ii) The emulation of the CT plant dynamics is done by running the DT plant dynamics of the plant at a very high frequency, with the DT dynamics (without time delay (Åström and Wittenmark 1990)) of the plant represented by $x_{k+1} = \sigma x_k + \eta u_k$, where $\eta = \Gamma_1(\tau)$ and the state of the plant is sampled at $S_k = 100\mu$s.

8.3.3 MATLAB-HIL Comparison. In Figure 9, we compare the MATLAB and HIL experiments, with y^{MATLAB} and y^{HIL} the outputs, respectively. In these experiments, the platform has been configured with $\psi = 5.95904$ms, $N = 10$ slots, and $M = 1$ slot. The design flows compute the following timing properties: $h^{SR} = 60$ms, $h_1^{MR} = 1$ms, and $h_2^{MR} = 56$ms. In Figure 9(a) the comparisons of the simulation outputs is presented. In essence, the three design flows have been simulated and it can be seen the difference between SR (top plot) and MR (middle and bottom plots) sampling (i.e., equidistant and non-equidistant samples). Note that both MATLAB and HIL experiments show very similar results, which means that the implementation of such controller is feasible for the



(a) QoC based on settling time. QoC = 1/settling time.



(b) QoC based on ITAE. QoC = $1/\sum t_k|y_k - r|$.

Fig. 10. Comparison of the QoC with respect to the resource utilisation $U(\lambda_C)$ for each design flow.

studied motion system. In Figure 9(b) the absolute errors of the simulation outputs are shown. The error for the SR design flow (top plot) is very small due to the use of a single controller. Finally, the error for the MR design flows (middle and bottom plots) appears with the switching between h_1^{MR} and h_2^{MR} but it does not exceed 4% with respect to the static reference signal that is set to 0.05 radians (see Figure 9(a)). These results show that both type of experiments are closely matching. In what follows, we focus on analysing the results and perform trade-off analysis.

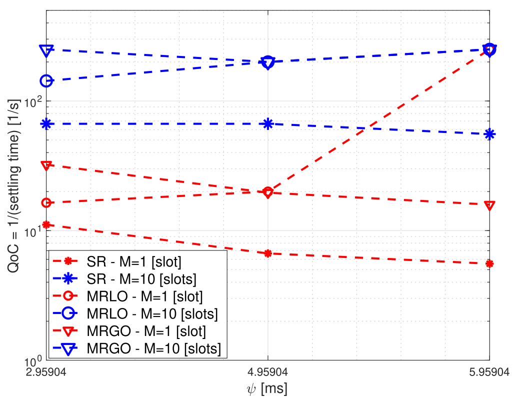
8.4 Trade-off Analysis: QoC and $U(\lambda_C)$

We run several experiments to evaluate the impact of $U(\lambda_C)$ on the QoC for each design flow. To that end, we set the platform up with $\omega = 40.96\mu$s. We varied $\psi = 2.95904, 4.95904, 5.95904$ms, $N = 1, 2, 6, 10$ slots, and $M = 1$ slot or $M = 10$ slots with $N = 10$. In Figure 10, we compare the QoC based on settling time (see Figure 10(a)) and ITAE (see Figure 10(b)) with $U(\lambda_C)$. As expected, it can be seen that the trends depend on the selected metric. We explain further results using the QoC based on the settling time (see Equation (7)). It can be seen, a common trend is the increase in QoC when more resources are assigned to λ_C. One can also note that the MRLO and MRGO design flows bring high performance with at least $\approx 40\%$ of the resources allocated to λ_C. Another interesting result is that the QoC of the SR design flow only reaches up to 25% of the resource utilisation. This is due to the fact that $\psi \gg T$. Thus, even when the TDM frame only has one application slot $N = 1$, a great part of ψ is unused. This lower platform resource utilisation is one of the shortcomings of the SR design flow.

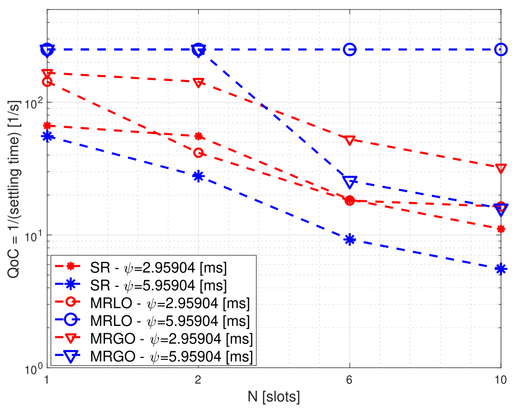
8.5 Impact of Platform Settings

As presented in the design flows, the platform settings can be reconfigured to meet design requirements. However, the CoMik slot duration ω can be considered to be limited by the implementation (i.e., we run the hardware as fast as possible and minimise ω). Thus, we focus on the application slot duration ψ and the number of slots within the TDM frame N. We varied these settings for a control task execution time $T = 0.99$ms. In what follows, we only refer to Figure 11 for ease of reading. Please refer to Table 2 to see the corresponding sampling intervals used in the presented experiments in Figure 11.

- **Application slot duration ψ:** In Figure 11(a), we present the results of the QoC in terms of ψ values. The increase in ψ, leads to a QoC deterioration of the SR design flow, since



(a) QoC vs. ψ.



(b) QoC vs. N.

Fig. 11. Impact of ψ and N to the system output for the SR, MRLO, and MRGO design flows. (a) Fixed $N = 10$ slots, varied ψ with 2.95904, 4.95904, and 5.95904ms, and varied M with 1 and 10 slots. (b) Fixed $M = 1$ slot, varied N with 1, 2, 6, and 10 slots, and varied ψ with for 2.95904 and 5.95904ms.

Table 2. Sampling Intervals for Experiments Presented in Figure 11

	ψ [ms]	N [slots]	M [slots]	h^{SR} [ms]	h_1^{MR} [ms]	h_2^{MR} [ms]
Sampling in Figure 11(a)	295904	10	1	30	1	29
	2.95904	10	10	3	1	2
	4.95904	10	1	50	1	47
	4.95904	10	10	5	1	2
	5.95904	10	1	60	1	56
	5.95904	10	10	6	1	2
Sampling in Figure 11(b)	295904	1	1	3	1	2
	2.95904	2	1	6	1	5
	2.95904	6	1	18	1	17
	2.95904	10	1	30	1	29
	5.95904	1	1	6	1	2
	5.95904	2	1	12	1	8
	5.95904	6	1	36	1	32
	5.95904	10	1	60	1	56

Top: Sampling intervals when ψ is varied (Figure 11(a)). Bottom: Sampling intervals when N is varied (Figure 11(b)).

the sampling interval h^{SR} increases. Similar results can be seen for the MRGO design flow when $M = 1$. The QoC in this case decreases with longer ψ, due to the enlargement of the sampling interval h_2^{MR} (29, 47, and 56ms for an increasing ψ) while h_1^{MR} does not change. When $M = 10$, h_2^{MR} remains constant at 2ms regardless of ψ. This leads to a fairly high and constant QoC, since the sampling interval h_{MR}^2 does not change significantly. The MRLO design flow QoC shows a different trend for which a longer ψ improves the QoC. This is due to the fact that the control task runs more often within ψ (2, 4, and 5 for an increasing ψ). For the same design flow, the impact of increasing M to 10 slots is reflected by an overall increase in the QoC due to shorter sampling intervals $h_1^{MR} = 1$ms and $h_2^{MR} = 2$ms.

RIGHTSLINK

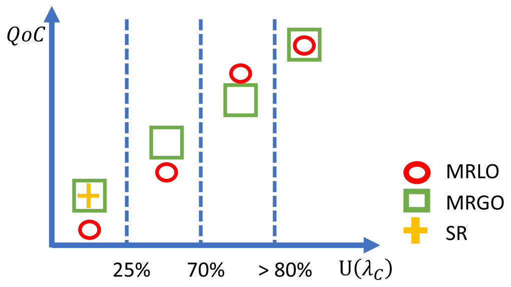


Fig. 12. Design guidelines in terms of QoC and $U(\lambda_C)$.

Table 3. Design Guidelines in Terms of the Platform Parameters

Design flow	**Choice of ψ**	**Choice of N**
SR	Small ψ is always good	- Depends on application sharing - Small is better
MRLO	Large ψ is always good	- Depends on application sharing - Small is better
MRGO	- Less sensitive to ψ - Small is recommended	- Depends on application sharing - Small is better

- **TDM frame slots number N:** In Figure 11(b), we present the results of the QoC in terms of the frame slots nummber N, for a fixed $M = 1$, while varying N with 1, 2, 5, and 10 slots, and varying ψ with 2.95904 and 5.95904ms. For the SR design flow, increasing N enlarges the sampling interval h^{SR} that leads to a QoC deterioration. For the MRLO design flow, we see two scenarios for $\psi = 2.95904$ and 4.95904ms. For the smaller ψ, we notice that the QoC decreases with larger values of N. This enlarges the TDM frame with more slots and make the sampling interval h_2^{MR} longer (2, 5, 17, and 29ms for an increasing N), which negatively influences the QoC. For the larger ψ, the controller manage to achieve the reference with a high QoC regardless of N. This happens because the control task runs more frequently within ψ, meaning that the controller can sample and control more frequently (with nominal sampling interval and feedback control gain h_1^{MR} and K_1, respectively). For MRGO design flow, the QoC presents two type of behaviours. The former one is given by a high QoC for $N = 1, 2$ slots. This results from the short sampling intervals in those configurations. The latter one is given by a decreasing QoC for $N = 6, 10$ slots, which is due to the increasing sampling interval h_2^{MR}.

8.6 Design Guidelines

One can notice that there is no optimal design flow that guarantees fastest settling time and least amount of resources. However, we see that each design flow has its benefits and drawbacks depending on the requirements and platform configurations. Therefore, we present the following design guidelines that are illustrated in Figure 12 accompanied with Table 3.

- **QoC and resource utilisation $U(\lambda_C)$:** When resource utilisastion is low (below 25%, which further depends on the platform configuration), SR and MRGO perform better than MRLO. When $U(\lambda_C)$ is in the range of 25–70%, both MRGO and MRLO can be used while MRGO performs better. When $U(\lambda_C)$ is in the range of 70–80%, both MRGO and MRLO can be used while MRLO performs better. When $U(\lambda_C) > 80\%$, both MRGO and MRLO perform equally good.

- **Choice of $\boldsymbol{\psi}$:** For the SR design flow, a smaller ψ is always good. A larger ψ is recommended for MRLO for a high $U(\lambda_C)$. MRGO is less sensitive to the choice ψ while a smaller ψ is recommended.
- **Choice of $\boldsymbol{N}$:** A smaller N is good for all the three design flows. However, a small N implies less number of applications can be executed on the platform. Depending on the number of other application that need to run on the platform, N should be chosen as small as possible.

9 CONCLUSIONS

We have presented three platform-aware control design flows that have been validated and compared using MATLAB and HIL experiments. We have shown in our experiments that each design flow can be used depending on the requirements that are given on the QoC and a targeted resource utilisation. Furthermore, we have shown how the time precision offered by composable and predictable platforms can be exploited to design SR and MR control systems considering various design constraints. For future work, a multi-rate observer/estimator module will be designed to address the cases where all states are not measurable. This is often the case in real-life physical systems.

REFERENCES

A. Aminifar, P. Eles, and Z. Peng. 2015. Jfair: A scheduling algorithm to stabilize control applications. In *Proceedings of the 21st IEEE Real-time and Embedded Technology and Applications Symposium.* 63–72.

A. Aminifar, S. Samii, P. Eles, Z. Peng, and A. Cervin. 2012. Designing high-quality embedded control systems with guaranteed stability. In *Proceedings of the IEEE 33rd Real-time Systems Symposium.* 283–292.

A. Aminifar, P. Tabuada, P. Eles, and Z. Peng. 2016. Self-triggered controllers and hard real-time guarantees. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE).* 636–641.

K. Arzen, A. Cervin, J. Eker, and L. Sha. 2000. An introduction to control and scheduling co-design. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Vol. 5. 4865–4870.

Karl-Erik Årzén and Anton Cervin. 2005. Control and embedded computing: Survey of research directions. *IFAC Proc.* Vol. 38, 1 (2005), 191–202.

Karl J Åström. 1970. *Introduction to Stochastic Control Theory.* Elsevier.

Karl Johan Åström and Richard M. Murray. 2008. *Feedback Systems: An Introduction for Scientists and Engineers.* Princeton University Press.

Karl J. Åström and Björn Wittenmark. 1990. *Computer-controlled Systems: Theory and Design* (2nd Ed.). Prentice-Hall, Inc., Upper Saddle River, NJ.

N. W. Bauer, S. J. L. M. B. van Loon, N. van de Wouw, and W. P. M. H. M. Heemels. 2014. Exploring the boundaries of robust stability under uncertain communication: An NCS toolbox applied to a wireless control setup. *IEEE Control Syst. Mag.* 34, 4 (Aug. 2014), 65–86.

Dimitri P. Bertsekas. 2005. *Dynamic Programming and Optimal Control.* Athena Scientific. 3rd ed. Vols. 1 and 2.

A. Biondi, M. D. Natale, and G. Buttazzo. 2018. Response-time analysis of engine control applications under fixed-priority scheduling. *IEEE Trans. Comput.* 67, 5 (May 2018), 687–703.

Sergio Bittanti and Patrizio Colaneri. 2008. *Periodic Systems: Filtering and Control.* Vol. 5108985. Springer Science & Business Media.

Arben Çela, Mongi Ben Gaid, Xu-Guang Li, and Silviu-Iulian Niculescu. 2014. *Resource Allocation in Distributed Control and Embedded Systems.* Springer International Publishing, Cham, 9–29.

A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K. Arzen. 2003. How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. *IEEE Control Syst. Mag.* 23, 3 (June 2003), 16–30.

A. Cervin, M. Velasco, P. Marti, and A. Camacho. 2011. Optimal online sampling period assignment: Theory and experiments. *IEEE Trans. Control Syst. Technol.* 19, 4 (July 2011), 902–910.

Wanli Chang, Dip Goswami, Samarjit Chakraborty, and Arne Hamann. 2018. OS-aware automotive controller design using non-uniform sampling. *ACM Trans. Cyber-Phys. Syst.* 2, 4, Article 26 (July 2018), 22 pages.

W. Chang, D. Goswami, S. Chakraborty, L. Ju, C. J. Xue, and S. Andalam. 2017. Memory-aware embedded control systems design. *IEEE Trans. Comput.-aided Design Integr. Circ. Syst.* 36, 4 (Apr. 2017), 586–599.

W. Chang, A. Pröbstl, D. Goswami, M. Zamani, and S. Chakraborty. 2014. Battery- and aging-aware embedded control systems for electric vehicles. In *Proceedings of the IEEE Real-time Systems Symposium*. 238–248.

Douglas E. Crabtree and Emilie V. Haynsworth. 1969. An identity for the Schur complement of a matrix. *Proc. Amer. Math.* 22, 2 (1969), 364–366.

P. Deng, Q. Zhu, A. Davare, A. Mourikis, X. Liu, and M. D. Natale. 2016. An efficient control-driven period optimization algorithm for distributed real-time systems. *IEEE Trans. Comput.* 65, 12 (Dec. 2016), 3552–3566.

W. Geelen, D. Antunes, J. P. M. Voeten, R. R. H. Schiffelers, and W. P. M. H. Heemels. 2016. The impact of deadline misses on the control performance of high-end motion control systems. *IEEE Trans. Industr. Electron.* 63, 2 (2016), 1218–1229.

Kees Goossens, Martijn Koedam, Andrew Nelson, Shubhendu Sinha, Sven Goossens, Yonghui Li, Gabriela Breaban, Reinier van Kampenhout, Rasool Tavakoli, Juan Valencia, Hadi Ahmadi Balef, Benny Akesson, Sander Stuijk, Marc Geilen, Dip Goswami, and Majid Nabi. 2017. *NoC-Based Multiprocessor Architecture for Mixed-Time-Criticality Applications*. Springer Netherlands, Dordrecht, 491–530.

D. Goswami, M. Lukasiewycz, R. Schneider, and S. Chakraborty. 2012. Time-triggered implementations of mixed-criticality automotive software. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'12)*. 1227–1232.

D. Goswami, A. Masrur, R. Schneider, C. J. Xue, and S. Chakraborty. 2013. Multirate controller design for resource- and schedule-constrained automotive ECUs. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'13)*. 1123–1126.

D. Goswami, R. Schneider, and S. Chakraborty. 2014. Relaxing signal delay constraints in distributed embedded controllers. *IEEE Trans. Control Syst. Technol.* 22, 6 (Nov. 2014), 2337–2345.

Joseph L. Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M. Tilbury. 2004. *StateSpace Feedback Control*. IEEE.

M. Karpenko and N. Sepehri. 2006. Hardware-in-the-loop simulator for research on fault tolerant control of electrohydraulic flight control systems. In *Proceedings of the American Control Conference*. 7.

Benjamin C. Kuo. 1992. *Digital Control Systems* (2nd ed.). Oxford University Press, Inc., New York, NY.

MathWorks. 2018. What Is Hardware-in-the-Loop Simulation? Retrieved from https://nl.mathworks.com/help/physmod/simscape/ug/what-is-hardware-in-the-loop-simulation.html.

Róbinson Medina, Sander Stuijk, Dip Goswami, and Twan Basten. 2017. Exploring the trade-off between processing resources and settling time in image-based control through LQR tuning. In *Proceedings of the Symposium on Applied Computing (SAC'17)*. ACM, New York, NY, 1456–1459.

M. Morelli and M. Di Natale. 2014. An MDE approach for the design of platform-aware controls in performance-sensitive applications. In *Proceedings of the IEEE Emerging Technology and Factory Automation (ETFA'14)*. 1–8.

A. Nelson, A. Beyranvand Nejad, A. Molnos, M. Koedam, and K. Goossens. 2014. CoMik: A predictable and cycle-accurately composable real-time microkernel. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'14)*. 1–4.

Ron T. Ogan. 2015. *Hardware-in-the-Loop Simulation*. Springer London, London, 167–173.

A. Palladino, G. Fiengo, F. Giovagnini, and D. Lanzo. 2009. A micro hardware-in-the-loop test system. In *Proceedings of the European Control Conference (ECC'09)*. 3833–3838.

D. Roy, L. Zhang, W. Chang, D. Goswami, and S. Chakraborty. 2016. Multi-objective co-optimization of flexray-based distributed control systems. In *Proceedings of the IEEE Real-time and Embedded Technology and Applications Symposium (RTAS'16)*. 1–12.

S. Samii, A. Cervin, P. Eles, and Z. Peng. 2009. Integrated scheduling and synthesis of control applications on distributed embedded systems. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition*. 57–62.

R. Schneider, D. Goswami, A. Masrur, M. Becker, and S. Chakraborty. 2013. Multi-layered scheduling of mixed-criticality cyber-physical systems. *J. Syst. Architect. Embed. Syst. Design* 59, 10-D (2013), 1215–1230.

L. Subramanian, V. Seshadri, A. Ghosh, S. Khan, and O. Mutlu. 2015. The application slowdown model: Quantifying and controlling the impact of inter-application interference at shared caches and main memory. In *Proceedings of the 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'15)*. 62–75.

N. Truong. 2012. Hardware-in-the-loop approach to controller design and testing of motion control systems using xPC target. In *Proceedings of the 4th International Conference on Intelligent and Advanced Systems (ICIAS'12)*, Vol. 1. 117–121.

J. Valencia, D. Goswami, and K. Goossens. 2015. Composable platform-aware embedded control systems on a multi-core architecture. In *Proceedings of the Euromicro Conference on Digital System Design*. 502–509.

J. Valencia, E. P. van Horssen, D. Goswami, W. P. M. H. Heemels, and K. Goossens. 2016. Resource utilization and quality-of-control trade-off for a composable platform. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'16)*. 654–659.

J. van Zundert and T. Oomen. 2018. LPTV loop-shaping with application to non-equidistantly sampled precision mechatronics. In *Proceedings of the IEEE 15th International Workshop on Advanced Motion Control (AMC'18)*. 467–472.

A. Varga. 2008. On solving periodic Riccati equations. *Numer. Lin. Alg. Appl.* 15, 9 (2008), 809–835.

Wayne Wolf. 2009. Cyber-physical systems. *Computer* 42, 3 (Mar. 2009), 88–89.

Xilinx. 2018. Virtex-6 FPGA ML605 Evaluation Kit. Retrieved from https://www.xilinx.com/products/boards-and-kits/ek-v6-ml605-g.html.

Received August 2018; revised February 2019; accepted February 2019

RIGHTSLINK Copyright Clearance Center