UNIVERSITY of York

This is a repository copy of *Semantics-preserving cosynthesis of cyber-physical systems*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/164989/

Version: Accepted Version

Article:
Roy, Debayan, Zhang, Licong, Chang, Wanli orcid.org/0000-0002-4053-8898 et al. (2 more authors) (2018) Semantics-preserving cosynthesis of cyber-physical systems. Proceedings of the IEEE. pp. 171-200. ISSN 1558-2256

https://doi.org/10.1109/JPROC.2017.2779456

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

INVITED PAPER

Semantics-Preserving Cosynthesis of Cyber–Physical Systems

While control theory provides methods for designing provably correct controllers, there is a lack of available techniques to ensure that high-level controller models are transformed into implementations while preserving model-level semantics and safety properties. This paper reviews recent efforts to address this issue using cyber–physical system (CPS)-oriented controller/platform cosynthesis techniques.

By Debayan Roy, *Student Member IEEE*, Licong Zhang, *Student Member IEEE*, Wanli Chang, *Member IEEE*, Sanjoy K. Mitter, *Fellow IEEE*, and Samarjit Chakraborty, *Senior Member IEEE*

ABSTRACT | Software-based control of physical systems is common in domains such as automotive, avionics, and industrial automation. Safety of such systems is determined by control-theoretic properties such as stability, settling time, and peak overshoot. These properties strongly depend on the software code generated from high-level controller models, and the implementation of such code on an embedded platform. To ensure safety, the semantics of the system model considered for controller design must be faithfully preserved in the platform implementation. However, traditionally, controller design and implementation platform design are carried out in isolation, followed by their integration, which often relies on simulations to estimate the behavior of the controllers. Thus, safety properties that were proven at the model level using control-theoretic tools can no longer be established in an actual implementation. This makes the design of embedded control systems costly, error prone, and hinders certification. In this paper, we review recent efforts in control-platform cosynthesis techniques toward addressing this problem. Here, the control and the embedded systems communities have come together to adopt a cyber–physical system (CPS)-oriented design paradigm. This cosynthesis paradigm integrates the design of control algorithms and platform parameters within a holistic optimization framework and accounts for relevant details from both sides. We survey the evolution of design approaches for such cosynthesis and show how–the originally disjoint–controller and the platform design methods are gradually converging.

KEYWORDS | Control systems; cosynthesis; cyber–physical systems; embedded control systems; embedded systems; platform aware; safety

Manuscript received March 25, 2017; revised August 4, 2017; accepted August 4, 2017. Date of current version December 20, 2017. This work was supported by Deutsche Forschungsgemeinschaft (DFG) through the Technical University of Munich (TUM) International Graduate School of Science and Engineering (IGSSE).

D. Roy, L. Zhang, and **S. Chakraborty** are with the Real-Time Computer Systems, Department of Electrical and Computer Engineering, Technical University of Munich, 80333 Munich, Germany (e-mail: debayan.roy@tum.de; licong.zhang@tum.de; samarjit@tum.de).

W. Chang is with the Department of Computer Science, University of York, UK (e-mail: wanli.chang@york.ac.uk).

S. K. Mitter is with the Laboratory for Information and Decision Systems, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: mitter@mit.edu).Digital Object Identifier: 10.1109/JPROC.2017.2779456

I. INTRODUCTION

Over the last ten years the concept of cyber–physical systems (CPSs) has emphasized the integrated modeling and analysis of computational platforms and the physical processes that are controlled by such platforms. One typical class of CPSs is made up of embedded control systems. In such a system, physical processes are controlled by a piece of software running on an embedded platform. Such systems are commonly found in automotive, avionics, industrial automation, and medical devices.

These systems are often safety critical with strict requirements on stability and performance (characterized by settling time, peak overshoot, or similar metrics) [1] and must meet certain certification standards [2], [3]. Traditionally, the design of control algorithms and the embedded platforms on which such algorithms are to be implemented are designed by different groups of engineers with different expertise. Such separation of concerns while common in the general-purpose computing domain becomes problematic for embedded control systems. Here, a fundamental challenge in ensuring that safety properties at the model level (or controller design stage) hold true in an implementation requires that model-level semantics are faithfully preserved when generating implementations from the models. This is, however, not straightforward considering that current controller design methods are mostly based on idealistic assumptions on the implementation platform, such as: computing the control law takes negligible time; there are no sensor-to-controller and controller-to-actuator delays; control inputs can be computed with infinite precision; and when software code is generated from high-level models (such as those specified in Matlab/Simulink), the code generator does not introduce any side effects and accurately preserves the model level semantics. As implementation platforms become more complex, distributed, and heterogeneous, these assumptions are increasingly not true, thereby resulting in a large deviation in the behavior of an implementation from the designed controllers at the model level, and often violating safety properties that were true at the model level.

Here, the question is: How should platform configuration parameters—e.g., scheduling policies/parameters, arithmetic precision, code generation policies—be chosen so that model-level semantics are preserved in an implementation? Given an already designed (model-level) controller, the choice of such platform parameters may be restricted or in the worst case there might not be any feasible platform parameters. Since there might be multiple controllers that satisfy given stability and performance specifications (safety properties), a better approach is to determine the controller and platform configuration parameters together. In other words, by cosynthesizing the controller and platform configuration parameters—i.e., as a part of a common optimization framework—we can ensure that the two designs are compatible (or model-level semantics are preserved) and there is a larger set of parameters that may be explored [4].

In this paper, we survey such cosynthesis techniques for embedded control systems design and implementation. We first study the problems with separation of concerns, where controller design and platform implementation are carried out in isolated design spaces without sufficient knowledge of each other. Subsequently, we survey different works that follow a CPS-oriented approach and broadly classify them in terms of whether 1) the implementation platform is fixed and the control algorithm is adapted to fit the platform architecture [as in networked control systems (NCSs) where the characteristics of the wireless network such as delay and packet loss probabilities are given and the control algorithms are designed taking them into account]; 2) the control algorithm and its assumptions are given and the platform is designed to meet these assumptions as closely as possible (e.g., by designing appropriate scheduling and resource allocation policies); or 3) it is a true cosynthesis where the parameters of the control algorithms and the implementation platform are jointly determined within an integrated optimization framework.

Recently, a lot of work in this area has been done, especially in the context of automotive embedded control systems. This is because automotive software systems implement a large number of safety-critical control loops. They have to be implemented on a resource-constrained, distributed, and heterogeneous platform architecture, and increasingly need to be certified. This combination of requirements makes it a challenging and a particularly suitable domain for studying design methods for embedded control systems. Hence, most of the examples we review in this paper are from this domain. However, the problems and the solutions that we discuss in this paper are also relevant to other CPS domains. In particular, we survey several works on NCSs that are commonly found in avionics, power grid, and industrial-automation-related CPSs. We believe that it is possible to leverage the progress made in NCSs and extend existing cosynthesis approaches from the automotive domain to other CPS domains as well.

A. Separation of Concerns

Automatic control is a well-studied subject with several decades of history and a large pool of design methods. Early works on design and analysis of a control system have focused on the mathematical model of the closed-loop system, including the plant and the controller. The controller is designed such that the system is stable and certain performance requirements, e.g., settling time, peak overshoot, and energy constraints are satisfied. However, these works do not consider implementation related details such as nonnegligible and variable times for software execution and data transmission, faulty networks, and finite precision arithmetic.

Embedded platform design is also well known and is composed of several stages: 1) task partitioning and mapping; 2) frame packing (for communication messages); and 3) task and frame scheduling. Platform design and analysis consider timing properties, e.g., application latencies, periods, relative deadlines, task execution times, and message frame transmission times. The main focus has been to synthesize implementations that are schedulable (i.e., all real-time software tasks meet their deadlines), and resource efficient (i.e., minimum usage of computation and communication resources). However, this theory is not directly applicable to control applications as control requirements such as stability and performance cannot always be

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

expressed as timing properties such as deadlines and periods (and when expressed in this form, the parameters can be overly pessimistic).

B. Safety Challenges for CPS Design: The Semantic Gap

In the context of CPSs, the separate design of controllers and platform parameters leads to a semantic gap between the system models considered in the controller design and the actual implementation. On one hand, the controller design is only influenced by the physical plant. Therefore, system stability and performance are derived without considering the cyber part, i.e., the implementation on the embedded platform. However, the implementation-related timing properties such as sampling period and sensing-to-actuation delay may degrade the performance and in the worst case may also cause system instability. Thus, the semantics of the control models may not be preserved in the implementation when the controller design is oblivious to the implementation details. On the other hand, the synthesis of platform parameters is based on the software-level timing details and does not consider control-theoretic metrics such as stability and performance. An incorrect timing characterization of control properties can result in an inconsistency between models and their implementation. For example, the performance requirement can enforce a strict constraint on application latency which if not correctly modeled may not be satisfied by the implementation.

Hence, it is difficult to design a safe CPS with such separation of concerns due to the associated semantic gap. We define an embedded control system to be safe when the corresponding software implementation meets the control requirements on stability and performance even in the worst case. Now, to ensure safety with separation of concerns, the whole process is usually carried out in an iterative manner as shown in Fig. 1. Here, the controllers and platform parameters are separately designed followed by integration and testing. In case a test shows that the requirements are not met, the steps are reiterated, possibly without any systematic feedback for improvement. This paradigm relies strongly on the prior experience of engineers and can be error prone. With the increasing size and complexity of modern embedded systems, this design paradigm is not sustainable. This leads to the need for new design approaches that can guarantee safety in a correct-by-design manner and do not depend on testing.

C. Bridging the Semantic Gap: CPS-Oriented Approaches

Due to the strong dependency between controller and platform design, both the control and the embedded systems design methods are gradually moving toward a CPS-oriented design paradigm. Control theorists have started accounting for implementation details and constraints of the underlying embedded platform and are integrating them in the mathematical models for controller design. For example, properties such as the sensing-to-actuation delay, input and output jitter, packet drops, deadline misses, and finite precision arithmetic are modeled and considered in the controller design phase, so that the designed controllers are platform aware. In the same vein, embedded systems engineers have also begun to study properties of control loops and are considering them in platform design methods. These properties include stability, performance, and robustness of control loops, and steady state and transient state characteristics. Consequently, the platform parameters such as task and message schedules can be tuned according to control objectives, rather than solely on intermediate objectives such as deadlines and latencies.

These CPS-oriented approaches, as shown in Fig. 1, consider realistic details of one side while designing parameters on the other side. In particular, they mathematically translate control properties into timing characteristics and *vice versa* to bridge the semantic gap. However, these methods consider the parameters on one side as given and design the parameters on the other side accordingly, and thus, it provides limited opportunity for optimization. They may result in a suboptimal design configuration with respect to

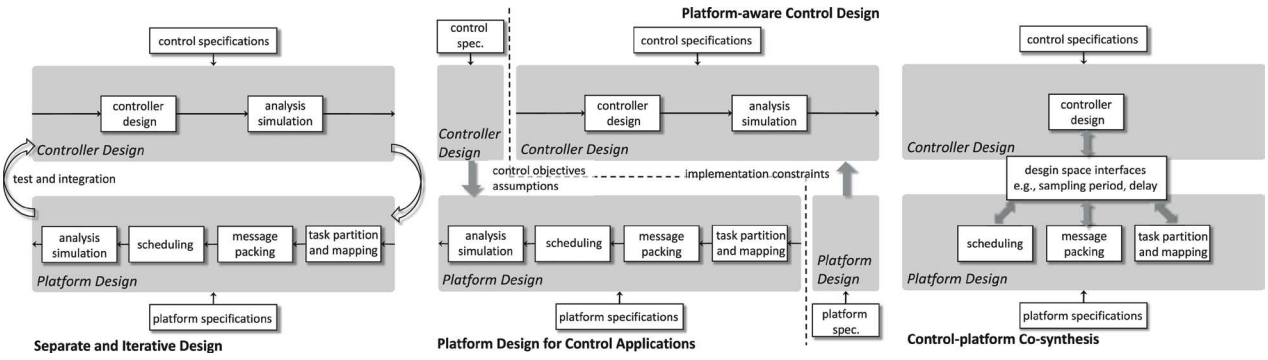


Fig. 1. ***Different design paradigms.***

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

control performance, resource efficiency, or both. In order to achieve higher design efficiency, it is important to design the control and the platform parameters together from joint specifications in a holistic optimization framework.

D. Ensuring Safety and Optimality: Cosynthesis of CPSs

In many cost-sensitive domains (such as automotive) it is not only necessary to ensure safety but it is equally desirable to achieve design optimality. In this survey, we emphasize on the importance of control-platform cosynthesis for CPSs toward ensuring safety and design optimality.

In recent years, a group of cosynthesis approaches have emerged that consider the design of control and platform parameters as a holistic optimization as shown in Fig. 1. Generally, the cosynthesis problem is formulated as a nonconvex optimization problem and is solved using a customized design space exploration (DSE) technique. The solutions provide both sets of parameters which are tuned according to certain objectives such as control performance and resource efficiency. Therefore, the synthesized parameters represent optimal design configurations. Moreover, the control model semantics are fully preserved in the implementation. This is because the controllers are designed according to the detailed constraints from the platform side and the platform parameters are synthesized considering stability and performance requirements from the control side. The synthesized parameters are, therefore, correct by design and ensure safety.

However, there exist considerable challenges that need to be addressed, if these approaches are to be applied to industrial scale applications. These challenges include handling complexity and scalability, developing closed-form optimization frameworks, inadequate toolchains, and certification issues. Moreover, existing approaches do not consider several aspects of platform architectures, e.g., memory hierarchy, heterogeneous networks, or multicore processors, all of which are common in modern embedded systems. Furthermore, they also do not take into account complex characteristics of control systems, e.g., time variance, nonlinearity, or input saturation. Hence, control-platform cosynthesis is a promising research direction with a number of open problems.

E. Paper Organization

We start with traditional problems and approaches that exist in both control theory and the embedded systems literature. In Section II, the basics of control theory are reviewed, particularly, system models, stability theorems, performance metrics, and common controller design methods. Subsequently, Section III provides the background on embedded systems design such as platform models, implementation constraints, and platform design and analysis techniques. Section IV first states the safety challenges associated with the design and implementation of CPSs. Subsequently, it reviews works on 1) how control engineers can consider implementation details in controller design; and 2) how embedded systems engineer can take into account the control properties in platform design. This is followed by Section V, where recent works on control-platform cosynthesis are studied and the general design flow for such approaches is outlined. Finally, possible future research directions and challenges are discussed in Section VI, followed by some concluding remarks (Section VII).

II. FEEDBACK CONTROL SYSTEMS

Control systems form an integral part of technological advancement in almost any field. They help in ensuring the intended functionality from machines and make processes run by adapting to the environment variables. More often than not, they are based on the theory of feedback as depicted in Fig. 2. In feedback control systems, a control action is decided based on the values of plant state variables and the reference that the plant must follow. In practice, some variables of the plant may not be measurable, and therefore, the corresponding values are estimated using an estimator. The basic idea is to mitigate the error between the plant output and the reference and therefore manipulate the plant to satisfy requirements on stability and performance. In this section, we will discuss how such a system can be mathematically modeled and the requirements can be mathematically expressed. Subsequently, we will also mention some techniques to design feedback controllers such that specified requirements are satisfied.

A. System Model

In this paper, we predominantly survey works which consider linear and time-invariant (LTI) systems with single-input–single-output (SISO). The mathematical model of the dynamic behavior of such a system in continuous time can be represented as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{1}$$

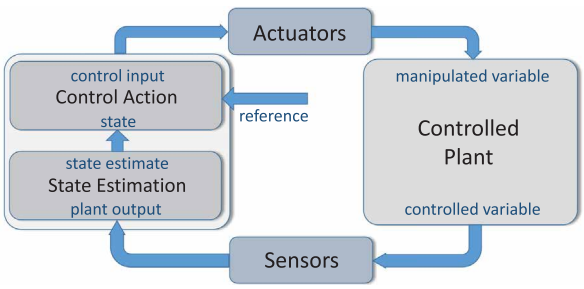


Fig. 2. ***Block diagram of feedback control systems.***

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

where vector $x(t) \in \mathbb{R}^{n\times 1}$ represents the states of the system, and $y(t)$ and $u(t)$ represent, respectively, the system output and the control input at instant t. Here, the constant matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$, and $C \in \mathbb{R}^{1\times n}$ are, respectively, the state, input, and output matrices.

Considering that the controller is implemented on an embedded platform, the control input is applied to the plant only at discrete instants $k \in \mathbb{Z}^*$. Let us assume that the time interval between two consecutive instants is a constant h and the control input to the plant is held constant until the next input is generated and applied, i.e., $u(t) = u(kh)$, where $kh \le t < (k+1)h$. This is equivalent to a system with a sample and hold device connected at the input, and correspondingly, h is the sampling period of the system. Consequently, the equivalent state–space model of the sampled data (discrete-time) system is given by

$$\begin{aligned} x[k+1] &= \phi x[k] + \Gamma u[k] \\ y[k] &= Cx[k] \end{aligned} \tag{2}$$

where the discrete-time state and input matrices ϕ and Γ can be derived from the continuous time matrices for a given sampling period h as follows:

$$\phi = e^{Ah}, \quad \Gamma = \int_0^h (e^{At}dt) \cdot B \tag{3}$$

B. Notions of Stability

For a given system model, the goal of a control engineer is to design a control law that computes the control input such that the closed-loop system satisfies specific requirements. One of the most important requirements is the stability of control loops. There are different notions of stability in control theory among which two important definitions are given here. To define stability, we must first introduce the equilibrium state x_e of a system as the state to which it converges in the absence of a control input (an unforced system), i.e., $u[k] = 0$.

1) Stability in the sense of Lyapunov: The equilibrium state x_e is stable in the sense of Lyapunov when the following holds:

$$\underset{\substack{\epsilon\in R^+ \\ k\in Z^* \\ x[0]}}{\forall} \underset{\delta\in R^+}{\exists} (||x[0]-x_e|| \le \delta) \Rightarrow (||x[k]-x_e|| < \epsilon). \tag{4}$$

Moreover, x_e is uniformly stable in the sense of Lyapunov when (4) holds and δ is independent of the initial state.

2) Asymptotic stability: x_e is said to be asymptotically stable when besides being stable in the sense of Lyapunov the following expression holds:

$$\underset{x[0]}{\forall} \underset{\delta\in R^+}{\exists} (||x[0]-x_e|| \le \delta) \Rightarrow \lim_{k\to\infty} ||x[k]-x_e|| = 0. \tag{5}$$

Moreover, x_e is uniformly asymptotically stable when δ is independent of the initial state in (5). x_e is globally asymptotically stable if, despite δ being arbitrarily large, the states finally converge to x_e.

C. Stability Analysis

For an unforced LTI system given by $x[k+1] = \phi x[k]$ with initial state $x[0]$, we can write as follows:

$$x[k+1] = \phi^{k+1} x[0]. \tag{6}$$

Without loss of generality, we can assume $x_e = 0$ from the definition of equilibrium state. Therefore, for a system to be asymptotically stable for a finite nonzero initial state, the following must hold:

$$\lim_{k\to\infty} ||\phi^k|| = 0. \tag{7}$$

This is only possible when the eigenvalues of ϕ, i.e., λ_is, $\forall i = 1, 2, \ldots, n$, satisfy the following:

$$|\lambda_i| < 1. \tag{8}$$

Here, λ_is also represent system poles. Thus, for a system to be asymptotically stable all the poles must lie within the unit circle in a complex z-plane.

However, this constraint is only valid for LTI systems and a more powerful technique for analyzing stability of both linear and nonlinear systems is the second method of Lyapunov. According to this theorem [5], for a discrete-time unforced system $x[k+1] = f(x[k])$, where $f(0) = 0$, if a scalar continuous function $V(x[k])$ exists such that

$$\begin{aligned} &\text{i) } V(0) = 0 \quad \text{ii) } \underset{x\neq 0}{\forall} V(x) > 0 \quad \text{iii) } \lim_{||x||\to\infty} V(x) \to \infty \\ &\text{iv) } \underset{x\neq 0}{\forall} \Delta V(x[k]) = V(x[k+1]) - V(x[k]) < 0 \end{aligned} \tag{9}$$

then $x_e = 0$ is globally asymptotically stable and $V(x)$ is a Lyapunov function.

This method can be simplified and applied for LTI systems $x[k+1] = \phi x[k]$, where x_e is asymptotically stable if and only if, for a given positive–definite real symmetric matrix Q, there exists a positive–definite real symmetric matrix P such that the following criterion is satisfied:

$$\phi' P \phi - P = -Q. \tag{10}$$

Here, $V(x[k]) = x'[k]Qx[k]$ is a Lyapunov function for the system and $\Delta V(x[k]) = -x'[k]Px[k]$ [5].

D. Quality of Control

Although stability is an essential requirement, different control applications may also need to satisfy different performance criteria. The performance of a controller is often measured by a metric that quantifies the quality of control (QoC). Thus, the goal of a control engineer is to design a

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

controller which not only meets the performance criteria but preferably has a higher QoC.

The performance measures of a control loop have evolved over the years from common metrics such as peak overshoot, rise time, settling time, and steady state error, to more complex cost functions and gains. Here, we will not discuss the common metrics, however, readers are encouraged to read [5] for more insights. We list some important performance measures as follows.

1) Integral cost function: System response to a given reference $r[k]$ can be analyzed for QoC based on several cost functions [6]. These cost functions consider tracking error $e[k] = r[k] - y[k]$ and are given as follows:

$$\sum_0^\infty e[k]^2 \quad \sum_0^\infty |e[k]| \quad \sum_0^\infty k|e[k]|. \tag{11}$$

Moreover, a more general quadratic cost which is a function of system states and control input can be considered. This is represented for finite and infinite horizon, respectively, as

$$J = \frac{1}{2}[x[N]'Sx[N] + \sum_0^{N-1}(x[k]'Qx[k] + 2x[k]'Mu[k] + u[k]'Ru[k])] \tag{12}$$

and

$$J = \frac{1}{2}\sum_0^\infty (x[k]'Qx[k] + 2x[k]'Mu[k] + u[k]'Ru[k]). \tag{13}$$

Here, S and Q are symmetric positive–semidefinite matrices while R is a symmetric positive–definite matrix. S, Q, R, and M are coefficient matrices used for weighing different terms according to their dimensions or importance.

2) $\mathcal{L}_2$ gain: For a given input, let γ_u be defined as the ratio of the output and the input energy of a system and can be represented as follows:

$$\gamma_u = \frac{||y||_2}{||u||_2} = \left(\frac{\int_0^\infty ||y(t)||^2 dt}{\int_0^\infty ||u(t)||^2 dt}\right)^{1/2}. \tag{14}$$

Now, the $\mathcal{L}_2$ gain $\gamma_{\mathcal{L}_2}$ of a system is defined as follows [7]:

$$\gamma_{\mathcal{L}_2} = \sup_{u \in \mathcal{L}_2} \gamma_u. \tag{15}$$

By the second method of Lyapunov, it can be stated that a system is asymptotically stable when $\gamma_{\mathcal{L}_2}$ is finite. Moreover, the $\mathcal{L}_2$ gain gives an idea of the robustness of a system and, therefore, is used as a performance measure.

Given a performance measure, the task of a control engineer is to design a controller that optimizes the system performance. However, this is not trivial as the design parameters, e.g., control gains, affect the control performance in a nonlinear and complex manner. Therefore, engineers may often need to do extensive analysis. For example, using root locus diagram, an engineer can analyze how control gains affect the system poles which in turn influence the transient response of the system. However, with performance metrics, such as gains and cost functions, control theorists have come up with novel design approaches for optimal control.

E. Control Design

Over the years, different techniques to design controllers that stabilize the system and also optimize QoC have been developed. A naive approach could use simulation where a controller is assumed to be given and then the closed-loop system is simulated for a certain given initial condition and reference. In case design requirements are not satisfied, then a different controller is assumed heuristically and the process is repeated until a suitable controller is found. However, this iterative approach is time consuming and cumbersome, and therefore, systematic mathematical approaches are more common. Here, we discuss three such approaches.

1) Pole placement technique: This design approach [8] exploits the fact that an LTI system is asymptotically stable when (8) holds. Now, for a state-feedback controller to reject impulse disturbance, control law can be written as

$$u[k] = -Kx[k] \tag{16}$$

where vector $K \in \mathrm{R}^{1\times n}$ represents feedback control gains. Therefore, (2) can be reformulated as

$$x[k+1] = (\phi - \Gamma K)x[k]. \tag{17}$$

Now, a system represented by (17) will be asymptotically stable when the eigenvalues of $\phi_{cl} = \phi - \Gamma K$ satisfy (8). The pole placement approach exploits this, and therefore, the control gains can be calculated for single-input systems using the Ackermann's formula

$$K = [0 \quad 0 \quad \cdots \quad 1]\,\gamma_c^{-1} H(\phi) \tag{18}$$

where γ_c is the controllability matrix

$$\gamma_c = \begin{bmatrix}\Gamma & \phi\Gamma & \phi^2\Gamma & \cdots & \phi^{(n-1)}\Gamma\end{bmatrix}. \tag{19}$$

For the eigenvalues λ_is, $H(\phi)$ is given by the following:

$$H(\phi) = (\phi - \lambda_1 \mathbf{I})(\phi - \lambda_2 \mathbf{I})\cdots(\phi - \lambda_n \mathbf{I}). \tag{20}$$

However, this approach is only applicable when the system is controllable, i.e., γ_c has full rank. Otherwise, not all the eigenvalues or poles can be freely selected. This means that the system is stabilizable only when the closed-loop poles which cannot be manipulated are already stable.

2) Linear quadratic regulator (LQR): This design approach [5] not only designs an asymptotically stable system but also considers optimization of the quadratic cost given by (12) and (13). Now, the linear feedback control for the finite horizon case is given by

$$u[k] = -K[k]x[k] \tag{21}$$

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

where control gains $K[\cdot]$ can be different for different samples. Here, the gains can be calculated from a dynamic Riccati equation by iterating backwards [8]. For an infinite horizon, the gain is constant for all samples and is obtained by solving the algebraic Riccati equation until a stationary solution is reached [8].

3) Linear quadratic Gaussian control (LQG): The limitation when using LQR is that all the states must be measurable to compute the control input. However, the measured states may not be accurate as there may be some noise in the measurement or noise inherent in the system. The noisy system model can be represented as

$$\begin{aligned} x[k+1] &= \phi x[k] + \Gamma u[k] + w[k] \\ y[k] &= Cx[k] + v[k] \end{aligned} \tag{22}$$

where for the sake of simplicity w and v can be assumed to be white noise. Now, to design an optimum controller which minimizes the loss function given by (13), we can apply the separation theorem [8]. Here, we first estimate the states $\hat{x}[k]$ and then apply the LQR technique to design the controller using the estimated states $\hat{x}[k]$. In this approach, the state estimation is realized using Kalman filtering [8], where the objective is to minimize the variance of the estimation error. This can be realized, for example, using a one-step-ahead predictor, where the next states are predicted based on the current state estimations, control input, and system output which can be expressed as follows:

$$\hat{x}[k+1|k] = \phi\hat{x}[k|k-1] + \Gamma u[k] + K[k](y[k] - C\hat{x}[k|k-1]). \tag{23}$$

To minimize error variance, the Kalman gains $K[\cdot]$ can be calculated by solving the parametric optimization problem based on the predictor model in (23) [8]. This can be represented as

$$\begin{aligned} \hat{x}[k|k] &= \hat{x}[k|k-1] + K_f[k](y[k] - C\hat{x}[k|k-1]) \\ \hat{v}[k|k] &= K_v[k](y[k] - C\hat{x}[k|k-1]) \\ \hat{x}[k+1|k] &= \phi\hat{x}[k|k] + \Gamma u[k] + \hat{v}[k|k]. \end{aligned} \tag{24}$$

The Kalman filter gains, i.e., $K_f[\cdot]$ and $K_v[\cdot]$, can be obtained by solving a Ricatti equation [8].

These mathematical design approaches guarantee stability and, in particular cases, optimal performance. However, they do not consider the controller implementation on the embedded platform which may influence the system model and therefore nullify the safety guarantees. Moreover, for an embedded implementation, the resources needed by a control software must also be an important consideration in the controller design stage. Consequently, the aforementioned techniques must be extended to consider resource constraints and specific characteristics of the implementation platform.

F. Nonlinear Dynamical Systems

Although we do not consider it in this survey, an important research direction in the field of control theory is the stabilization and control of nonlinear dynamical systems. In the context of CPSs, most of the works consider linear models of the physical system as commonly found in electrical circuits, mechanical systems, and chemical processes. However, new models found in the domains of avionics, autonomous vehicles, and power grid are inherently nonlinear due to their complex interaction with the environment. Naturally, the problems of stability analysis and controller design for nonlinear systems considering the details of platform implementation have become relevant. Toward this, there have been works focusing on NCSs, where the control loop is closed over a communication network. Correspondingly, several network-specific characteristics such as time-varying network delays, packet drops, and quantization influence control properties.

Besides techniques derived from traditional nonlinear control theory, two important approaches toward solving problems of nonlinear NCS are: 1) fuzzy-model-based control; and 2) formal synthesis of hybrid control systems. There has been significant progress on these two approaches where several implementation aspects have also been considered. However, most of these works assume very abstract platform models. They do not really derive the abstraction from real platform parameters such as network schedules, communication protocols, size of gateway buffers, or switch latencies. Hence, an integrated approach that closely binds controller design with platform parameter estimation is still missing. We discuss this topic again in Section VI-B. However, for more detailed survey on fuzzy-model-based nonlinear control and formal synthesis of hybrid control systems the readers are referred to [9] and [10]–[12], respectively.

III. EMBEDDED PLATFORMS: DESIGN AND ANALYSIS

Embedded systems are widely used in various domains such as automotive, consumer electronics, healthcare, avionics, and industrial automation. In each of these domains, an underlying electrical/electronic (E/E) platform is required, which provides computation and communication services to the functional software. A typical hardware architecture for such a platform consists of one or more processing units. Fig. 3 shows an example of a distributed embedded platform. Here, each processing unit has one or more processing cores, memory systems and input/output (I/O) ports. In the case of distributed architectures, multiple processing units are connected by one or more communication bus systems. Data can be transmitted between processing units as messages packed into frames over the bus. In a large-scale system, heterogeneous bus protocols are used where communication gateways can connect different bus clusters. Toward implementing software applications on

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

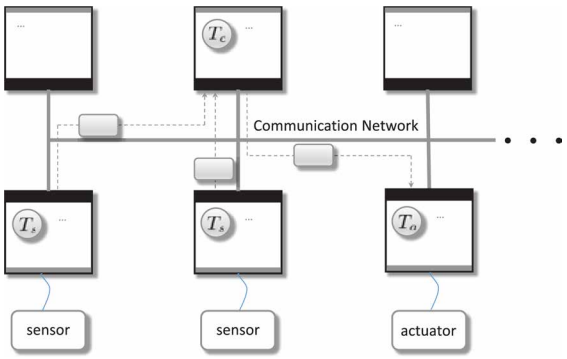


Fig. 3. ***A distributed embedded platform.***

such platforms, in this section, we will first describe the implementation model and associated implementation constraints, followed by common implementation techniques.

A. Platform Implementation Model

A software application can be implemented as several pieces of software codes called tasks. These tasks can be data dependent in the sense that the output of one task is considered as an input to another task. Two data-dependent tasks can be mapped on different processors due to reasons such as spatial distribution of sensors and actuators. In such a case, the data between them are transmitted over a communication bus via messages packed into frames. Thus, an application can be modeled as a directed task graph. Here, each vertex is a task and directed connecting lines represent data transmitted from source to target tasks. Subsequently, we will explain the timing models of tasks and data frames.

1) Task Model: Typically, in an embedded application, a task is executed multiple times triggered either a) periodically via time interrupts, i.e., in a time-triggered fashion; or b) aperiodically via events, i.e., in an event-triggered fashion. Moreover, when multiple tasks are mapped on a common processor, certain arbitration mechanisms are necessary. This is achieved by the operating system (OS), which is a software that schedules the tasks and allocates resources. Depending on the requirements of the applications, different scheduling schemes can be employed by the OS. Common scheduling schemes include time-triggered (TT) scheme (e.g., eCos), fixed-priority preemptive (FPP) scheme (e.g., OSEK), and dynamic scheduling schemes such as earliest deadline first (EDF).

In TT static scheduling, processor time allocation is preconfigured, i.e., it is known when a task will be executed by the processor. In this scheme, a periodic task is characterized by a tuple $T_i \sim \{p_i, o_i, e_i\}$, where p_i, o_i, and e_i represent, respectively, the period, the schedule offset, and the execution time. Since the execution time of a task is usually not deterministic, the worst case execution time (WCET) is used to represent the task schedule.

In contrast, in an FPP scheme, it is not known when a task will allocate a processing resource. Instead, it is decided at runtime by the OS according to the preset priority of the task. Without loss of generality, a periodic task can be characterized by the tuple $T_i \sim \{p_i, a_i, \pi_i, e_i\}$, which represent, respectively, the period, release time, priority, and execution time. The release time determines when a task instance is dispatched to be scheduled by the OS. Here, tasks are executed according to their priorities. If a task is currently running and a higher priority task arrives, the current task will be preempted. It will be resumed again when all task instances with higher priority are processed completely.

In the case of EDF scheduling scheme, the priorities of the tasks are not assigned offline, but are determined online according to remaining time to the deadline. Here, a task is represented as $T_i \sim \{d_i, e_i\}$. d_i is the relative deadline which is the time that the processor has to finish executing the task after the task release.

Each of these scheduling schemes has its own advantage such as timing predictability, resource efficiency, and implementation overhead, and the appropriate scheme can be chosen depending on requirements.

2) Frame Model: Each message frame may be packed with one or more data items and is transmitted over the bus according to different communication protocols. For example, message transmission may be achieved through a wireless medium (e.g., Zigbee, Bluetooth, and WLAN) or a wired medium (e.g., CAN, FlexRay, and Ethernet). Typical bus protocols include a) CAN, FlexRay, LIN, Ethernet, and MOST in the automotive domain; b) ProfiNet, Profibus, and EtherCAT in industrial automation; and c) AFDX in the avionics domain. Different communication protocols implement different scheduling schemes, which is determined by the media-access control (MAC) layer. For example, the CAN bus employs a fixed-priority nonpreemptive (FPNP) scheme, Ethernet implements collision sense multiple access/collision detection (CSMA/CD), EtherCAT implements polling, while FlexRay implements a hybrid scheme composed of a time-division multiple-access (TDMA)-based static segment and a flexible TDMA (FTDMA)-based dynamic segment.

For different protocols, frame timing models can be represented differently. For example, a CAN frame schedule can be represented as $fr_i \sim \{p_i, a_i, \pi_i, c_i\}$ where c_i represents the frame transmission time over the bus. However, a static FlexRay or TDMA frame timing is expressed as a tuple $fr_i \sim \{s_i, b_i, r_i\}$ where s_i is the slot id in which the frame is transmitted, b_i is the TDMA cycle when the frame is sent for the first time, and r_i represents the number of bus cycles after which the frame is sent again.

B. Implementation Constraints

The platform implementation often consists of determining various parameters associated with the processors and

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

the communication network. In processors, these parameters include task partition and mapping, and scheduling parameters such as the static task schedule or priorities. On the communication side, various parameters of the network need to be determined. The exact parameters depend on the protocol and the implementation. For example, the design parameters for CAN are the priorities of the messages. For FlexRay, this would include the whole configuration of the FlexRay communication cycle, i.e., the frame packing and frame-to-slot assignment.

Different constraints need to be considered when determining the platform parameters. These constraints may be enforced by the platform or are derived from application requirements. We discuss some important constraints as follows.

1) Processor Utilization: This is defined as the fraction of computation time for which the processor may be busy in the worst case for a given task mapping. Let us denote the WCET and the period of a task T_i as e_i and p_i, respectively. Now, for a set of tasks $\mathcal{T}(P_i)$ mapped onto a processor P_i, the processor utilization $U(P_i)$ is given by the following expression:

$$U(P_i) = \sum_{T_i \in \mathcal{T}(P_i)} \frac{e_i}{p_i}. \tag{25}$$

The processor utilization usually must be lower than a certain value, beyond which, the tasks are no longer schedulable (in practical cases, the value is much smaller the 100% for reliability reasons).

2) Bus Load: Bus load depends on the communication protocol used by the bus. For example, let us consider the FlexRay static segment. It is partitioned into N slots of equal length. In FlexRay, time is organized as an infinite repetition of 64 bus cycles. Here, the total number of slots in 64 cycles is $64N$. If Θ is the set of frames mapped onto a FlexRay bus, then the constraint on bus load U_{FR} is derived as follows:

$$U_{FR} = \sum_{fr_i \in \Theta} \frac{64}{r_i} \leq 64N. \tag{26}$$

However, for bus systems, besides the reliability, the extensibility (i.e., provision for mapping future messages) of the design also needs to be considered [13]. This is because industrial systems often follow an iterative design paradigm where the previous version is first inherited and then extended with new features. Now, if a bus is too highly loaded, it reduces the possibility of adding future messages.

3) Application-Level Constraints: In addition to the constraints imposed by limited platform resources, application requirements must also be considered in the design. Consider an application represented by a chain of tasks and messages as $a_i \sim T_1 \rightarrow fr_1 \rightarrow T_2 \rightarrow fr_2 \rightarrow \cdots \rightarrow T_n$. In such an application, the tasks and messages must be scheduled in a way such that the task dependency constraints are satisfied. For example, T_1 must finish before fr_1 is sent and T_2 must start only when fr_1 has arrived. Furthermore, there may be constraints on application latency, i.e., the time between the start of T_1 and the completion of T_n. This constraint is imposed on hard real-time distributed applications and correspondingly they must satisfy strict deadlines. Even the sensing-to-actuation delay of a control application can be expressed as a latency requirement.

4) Task- and Frame-Level Constraints: Application-level requirements are typically translated to deadlines of tasks and messages. Deadline constraints specify when the execution of a task (or transmission of a message) needs to be finished after its release. In this regard, the response time of a task (or a frame) is defined as the time elapsed between the release of the task (or frame) and the completion of task execution (or frame transmission). Response time not only depends on the code size in a task or the data size in a frame but also on the scheduling scheme on the processor and the bus. Denoting the response time of a task or a frame as R_i, and the deadline it must satisfy as d_i, then the deadline constraint is given as follows:

$$R_i \leq d_i. \tag{27}$$

In addition, for shared resources, tasks and messages must also satisfy constraints related to resource conflicts. Such a constraint typically states that no two tasks or messages must be allocated to the same resource at the same time.

C. Platform Analysis

Platform analysis verifies if the system design meets the specified constraints and timing requirements. The analysis techniques depend on the implemented scheduling strategy on a platform. Considering the wide spectrum of scheduling algorithms available in different domains, we would not go into the details of any specific analysis.

Early works on this topic have focused on providing schedulability tests and worst case response time (WCRT) analysis. For example, Liu and Layland [14] and Xu and Parnas [15] address the problem of schedulability tests for rate-monotonic, deadline-monotonic, and EDF scheduling schemes. Bril [16] and Bril *et al.* [17] propose the response time analysis for tasks. For example, in FPP scheduling, the WCRT of a task t_i can be computed using an iterative algorithm based on the following recurrence relation [16]

$$R_i = e_i + \sum_{\Pi_j > \Pi_i} \left\lceil \frac{R_i}{p_j} \right\rceil e_j. \tag{28}$$

In the case of communication networks, Davis *et al.* [18] address the problem of timing analysis of the CAN bus and Pop *et al.* [19] and Zeng *et al.* [20] consider the FlexRay bus.

Much effort has also been spent in deriving formal methods for compositional timing analysis. In this context, network calculus [21], real-time calculus [22], [23], and

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

SymTA/S approach [24] have been developed. These methods have been applied to processor scheduling [23] and to analyze communication networks such as FlexRay [25] and switched Ethernet topologies [26]–[28]. In addition, the problem of combined task and message timing analysis has also been addressed [29]–[31]. In this case, timing properties are considered at the application level.

D. Platform Design

Another important research focus in embedded systems is the design of platform parameters according to the constraints and timing requirements. Here, the design is mostly aimed at task mapping, frame packing, and task and message scheduling [32]–[35]. Several works also consider the combined synthesis of task and message schedules from the application-level timing requirements such as latencies and response times [36]–[39]. In these approaches, usually a constraint programming (CP) problem is formulated and solved with integer linear programming (ILP) or satisfiability modulo theories (SMTs) solvers, heuristics or metaheuristics methods. These approaches can usually guarantee that the requirements are met and further tune the parameters according to certain optimization objective(s). When multiple conflicting objectives are considered, a Pareto front can be generated or a DSE algorithm [40] can be applied to help the designer analyze different tradeoffs.

E. Hardware/Software Codesign

One additional direction of embedded systems design that has received attention in the past decade is hardware/software (HW/SW) codesign. Traditionally for electronic systems, the hardware is designed first, followed by the design of the software. HW/SW codesign approaches design both hardware and software components concurrently, thus enabling faster time to market and achieving more efficient design. However, these approaches only consider cost, performance, reliability, and power consumption as design objectives. They do not consider high-level control requirements such as stability and performance, when the software in question implements a feedback control loop. This survey focuses on the design of embedded control system from a CPS perspective and we refer the reader to [41] for a survey on HW/SW codesign. It may be noted that many optimization techniques used in HW/SW codesign, when appropriately adapted, can be utilized in the design of embedded control systems.

IV. CPS-ORIENTED APPROACHES TO EMBEDDED CONTROL SYSTEMS DESIGN

Our setup consists of a group of physical plants controlled by software running on a single processor or on a network of processors. In this section, we first study the interplay between the control system and the embedded platform using a motivational example. In doing so we point out the associated safety challenges. We will also discuss how the control and the embedded systems communities have attempted to address these challenges from their own perspectives.

A motivational example: As an example, we have considered a second-order system with the following system matrices:

$$A = \begin{bmatrix} -0.2 & 0.667 \\ -10 & -100 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 100 \end{bmatrix} \quad C = [1 \quad 0]. \tag{29}$$

For a sampling period of $h = 0.02$ s, the discrete-time matrices can be calculated using (3) as follows:

$$\phi = \begin{bmatrix} 0.9953 & 0.0057 \\ -0.0862 & 0.1349 \end{bmatrix} \quad \Gamma = \begin{bmatrix} 0.0076 \\ 0.8643 \end{bmatrix}. \tag{30}$$

Let us consider the control law as $u[k] = -Kx[k] + Fr$ where K and F are the feedback and the feedforward gains and r is the reference. Without loss of generality, we can assume $r = 0$ and use the pole placement technique described in Section II-E to calculate the feedback gain. Feedforward gain can be calculated using the final value theorem, i.e., $\lim_{t\to\infty} y(t) = r$, and is given by the following relation:

$$F = \frac{1}{C(\mathbf{I} - \varphi + \Gamma K)^{-1}\Gamma}. \tag{31}$$

For both the closed-loop poles at 0.9, K and F can be calculated as follows:

$$K = [0.7032 \quad -0.7811] \quad F = 0.8689. \tag{32}$$

Using these values, we have simulated the closed-loop system for unit step reference and plotted the output in Fig. 4.

Next, let us assume that the control code is implemented as three tasks T_s, T_c, and T_a in temporal order where the WCETs of the tasks are 1*ms*, 3*ms*, and 1*ms*, respectively. These tasks are mapped on different ECUs, each with a TT scheduler. The data between T_s and T_c are transmitted as a message m_s and the data between T_c and T_a are transmitted

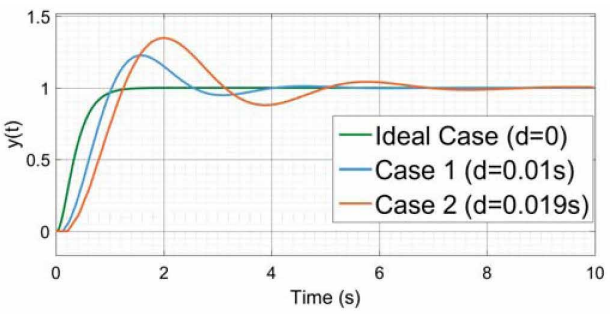


Fig. 4. ***Closed-loop simulation curves with delay.***

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

as another message m_c. The messages are transmitted over CAN. Now, let us consider two scenarios where the WCRTs of m_s and m_c are, respectively, calculated as follows: Case 1) 2*ms* and 3*ms*; and Case 2) 6*ms* and 8*ms*. We can schedule the tasks in a way such that all task dependency constraints are satisfied and the end-to-end delay is minimum. Now, for both cases, we have simulated the closed-loop system for a unit step reference signal. We have modeled the tasks T_s and T_a and the messages m_s and m_c as delay blocks of appropriate values, while T_c executes the control law followed by a delay of 3*ms*. The response curves for both the cases are shown in Fig. 4. It may be noted that the overshoot and the settling time have increased from the ideal case with no delay. Thus, we may say that the performance has deteriorated, and more importantly, it does not match the expected values that were obtained at the model level. This performance degradation can be attributed to the delays introduced in the loop which will be studied later in this section. Fig. 5 shows the interplay between the control models and the platform implementation, i.e., how the task and message schedules affect the sampling period and the closed-loop delay of the control loop. Thus, controller design without considering the timing properties of the platform implementation is unreliable and in the worst case may even result in an unstable system in spite of the model-level controller being stable.

As mentioned earlier, safety properties of embedded control systems is usually captured in terms of stability and performance guarantees. We can observe in the above example that when the controller is designed separately from the embedded platform and is oblivious to the implementation details, no such guarantees at the implementation level are possible. In practice, costly testing and integration efforts are necessary to obtain an acceptable implementation by iteratively changing the controller model and the platform parameters.

To address this issue, new controller design techniques that account for platform characteristics and resource constraints are discussed in Section IV-A. These techniques analyze properties of the platform architecture and incorporate them into the controller model, and then determine the appropriate parameters corresponding to this augmented model in order to ensure stability and performance.

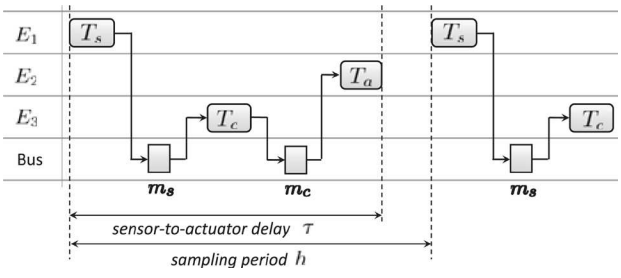


Fig. 5. ***Task/message schedules in a controller implementation.***

Similarly, on the embedded systems side, new techniques for platform design have been proposed that take into account the particular requirements of embedded control systems; we discuss them in Section IV-B. Here, a given control system is first analyzed to determine timing constraints that satisfy stability and performance requirements. Subsequently, platform parameters are synthesized taking these timing constraints into account. Both the aforementioned design paradigms have made significant progress in bridging the semantic gap between controller design and its implementation that is illustrated in the results shown in Fig. 4.

A. Platform-Aware Controller Design

The control theory community has looked into new controller design methods that take into account the implementation platform characteristics. Here we review some of the important work in this direction.

1) Sensing-to-Actuation Delay: The discrete-time control system model described in Section II-A assumes that there is a negligible time delay between sensing and actuation. This implies that the computation of control input takes negligible time. This is an idealistic assumption as embedded processors often have limited computation bandwidth and take nonnegligible time for computation. Moreover, sensors and actuators are often spatially distributed and the control loop may involve some communication over a shared network, which may introduce additional delays. Assuming a sensing-to-actuation delay (or closed-loop delay) of τ where $0 < \tau \leq h$, $u(t) = u[k-1]$ for $kh \leq t < kh + \tau$, and $u(t) = u[k]$ for $kh + \tau \leq t < (k+1)h$. Consequently, the discrete-time state-space model for the delayed system [8] becomes the following:

$$\begin{aligned} x[k+1] &= \phi x[k] + \Gamma_0 u[k] + \Gamma_1 u[k-1] \\ y[k] &= Cx[k]. \end{aligned} \tag{33}$$

Here, Γ_0 and Γ_1 for a sampling period h and a delay τ are given as follows:

$$\Gamma_0 = \int_0^{h-\tau} (e^{At}\,dt) \cdot B \quad \Gamma_1 = \int_{h-\tau}^{h} (e^{At}\,dt) \cdot B. \tag{34}$$

Consequently, we can consider an augmented state vector as $z[k] = [x[k] \quad u[k-1]]'$, for which the state–space model can be written as

$$\begin{aligned} z[k+1] &= \phi_z z[k] + \Gamma_z u[k] \\ y[k] &= C_z z[k] \end{aligned} \tag{35}$$

where ϕ_z, Γ_z, and C_z are as follows:

$$\phi_z = \begin{bmatrix} \phi & \Gamma_1 \\ 0 & 0 \end{bmatrix} \quad \Gamma_z = \begin{bmatrix} \Gamma_0 \\ \mathbf{I} \end{bmatrix} \quad C_z = [C \quad 0]. \tag{36}$$

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

This model is identical to the ideal discrete time model given by (2). Therefore, now the standard controller design methodology that was explained in Section II can be applied. This system model has been used in [42] and [43].

2) Input and Output Jitter: The closed-loop delay introduced in a controller implementation may not be a constant value and instead be time varying, resulting in output jitter. Similarly, a control task may not be sampled at regular intervals, thereby resulting in input jitter. This nondeterminism in timing may be induced by event-triggered preemptive scheduling of shared resources, or by asynchronous clocks in a distributed system, among other reasons. Moreover, this nondeterministic behavior may cause system instability or inadequate QoC. Thus, it is important to analyze the influence of input and output jitter on stability and performance of a control loop.

Toward this, there have been some work in NCSs that considers stochastic or approximate analysis of system stability [44]–[48]. However, Cervin [49] has proposed an analysis of system stability and worst case performance considering both input and output jitter. In [49], the assumption is that only the worst case delay, i.e., input and output jitter, is given without any information on the statistical distribution. Cervin introduces a novel technique to transform the closed-loop system model by adding two error paths corresponding to input and output jitter. In one error path, the influence of output jitter is modeled as an error in the actuation signal. In the second error path, the influence of input jitter is modeled as an error in the measurement. Subsequently, the stability and the performance of the transformed closed-loop system are analyzed in the frequency domain. This analysis can be very useful in practice to ensure safety of an implementation in the presence of jitter without extensive simulations or testing.

3) Processor Architecture and Operating Systems: Design of embedded systems often starts with the selection of architectural components, e.g., processors, buses, and the OS along with its scheduling policy and parameters. However, at this stage, the designer only has a very rough idea of the applications. Therefore, the choice of processor architecture and OS is almost oblivious of the applications that will run on the processor. However, there are certain OS features that invalidate the assumptions made in the controller design if not taken into account. For example, an OS such as ERCOSek, running on a processor may offer only a preconfigured set of sampling periods. Controllers implemented on such a processor must be implemented according to a sampling period selected from this set. The straightforward scheme is to find the largest sampling period H_j from the preconfigured set $\mathcal{H} = \{H_1, H_2, \ldots, H_k\}$ for which a controller can be designed satisfying the requirements. However, this might result in using an unnecessarily high sampling rate, and thereby overloading the processor.

To address this issue, Goswami *et al.* [50] have proposed to design controllers with nonuniform sampling. The idea is to choose a sampling order $\prod h_i = h_1 \rightarrow h_2 \rightarrow \cdots \rightarrow h_n \rightarrow repeat$. Here, $h_i \in \mathcal{H}$ and $h_{avg} = \frac{1}{n}\sum_{i=1}^{n} h_i$. Then, the controller is designed such that the obtained performance $J(\prod h_i)$ for the assumed sampling order satisfies the requirement $\bar{J}$ while $h_{\text{avg}} > H_j$. Higher h_{avg} implies savings in computation resource. Goswami *et al.* [50] also suggest a design technique for such a multirate controller. The controller is designed for the average sampling period h_{avg} using the pole placement technique described in Section II-E. The closed-loop system $\phi_{cl}(\prod h_i) = \phi_{cl}(h_1)\,\phi_{cl}(h_2)\cdots\phi_{cl}(h_n)$ can be analyzed for stability and performance using standard techniques.

A similar approach is also valid for multicore architectures where a TDMA-based execution policy is employed to eliminate interapplication interference or to offer compositionality. In such an architecture [51], processor time is partitioned into slots where each slot is dedicated to an application. Correspondingly, several instances of a controller may run in its TDMA slot thus resulting in a shorter sampling period. However, the last instance may either have to wait for the next dedicated slot to finish or the time gap between the last instance of the current slot and the first instance on the next slot is much larger. In both cases, the controller implementation naturally resembles a multirate case with only two sampling periods. Valencia *et al.* [42] have studied such an implementation of controllers in multicore architectures. The sampling order for a given TDMA policy is derived. Subsequently, a linear matrix inequality (LMI)-based approach is proposed to design the controller. Here, the two sampling periods result in two different subsystems and the overall system switches between the two. Therefore, it is suggested to design the controller corresponding to the shorter sampling period (the dominant one) using pole-placement technique. The stability of the overall switching system can be ensured by finding a common P such that (10) holds for both subsystems.

4) Deadline Misses, Packet Drops, and Fault Tolerance: A controller can be designed assuming a closed-loop delay and sampling period as discussed in Section IV-A1. Subsequently, in the schedulability analysis, the controller is treated as a hard real-time application where the delay is considered as a deadline and the sampling period as the dispatch period. A system is schedulable if all the WCRTs satisfy the deadline constraints. This schedulability test is conservative in nature because it relies on a critical instant (i.e., all the controllers are contending to run at the same time,) and on WCET estimates of the controller tasks (which are pessimistic in nature). Since the worst

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

case occurs rarely, this results in a poor resource utilization. Similarly, in NCSs, there is a hard constraint on in-time packet delivery. However, there may be momentary faults in the network which can corrupt the data. In highly constrained networks, there may be some packet losses as well.

In domains such as automotive, efficient utilization of resources is of utmost importance in order to minimize cost. Hence, the goal is to map as many applications as possible on a given embedded platform (single processor or network of processors). Moreover, for control applications, deadline or in-time packet delivery is not an accurate reflection of stability and QoC and designs driven by them might be overly conservative. This is because of the inherent robustness of control loops, which allows for certain deadline misses. However, quantifying such deadline miss patterns is not straightforward.

Recently, a number of works have addressed this issue [52]–[59]. Most of them consider a switched system model to represent a control-loop with deadline misses or packet drops [52]–[54], [58], [59]. There can be several ways in which deadline misses or packet drops can be modeled, i.e., 1) as an open-loop system, i.e., $u[k] = 0$; 2) as the last control input being used, i.e., $u[k] = u[k-1]$; or 3) as two consecutive misses, i.e., $u[k] = u[k-2]$. The probability of a deadline miss can either be stochastic or guided by the (m, k)-firm rule (i.e., only m out of k times deadlines can be missed).

It is natural to study the impact of deadline misses on the performance of control loops. Towards this, Geelen *et al.* [53] and Antunes and Heemels [54] have considered stochastic systems with packet drops or deadline misses. For a given ideal input signal (i.e., without any misses), mean and variance of the output signal are calculated in time domain and in frequency domain. Furthermore, van Horssen *et al.* [52] have studied performance degradation for switched systems with (m, k)-firm data losses, where a system switches between closed-loop and open-loop subsystems. Here, van Horssen *et al.* have represented the system with deadline misses as an automaton for which a stable controller can be designed using an LMI-based approach. Correspondingly, the loss in performance can be calculated by comparing the quadratic costs corresponding to the stable controller and the ideal (without misses) LQR controller. Van Horssen *et al.* [52] have further proposed that the performance can be improved at the cost of online computation by deadline-miss-aware controller updates.

In the same vein, Majumdar *et al.* [59] have considered $\mathcal{L}_\infty$-to-RMS gain as the performance measure, and, for a given successful packet transmission rate, an upper bound on $\mathcal{L}_\infty$-to-RMS gain is derived. This gain measure of a discrete-time LTI system with input disturbance ω and $||\omega||_\infty = \sup_{k\geq 0} ||\omega[k]||_2$ is given by the following expression and indicates how a system reacts to disturbance:

$$\sup_{||\omega||_\infty \neq 0, x[0]=0} \frac{\left(\lim\sup_{N\to\infty} \frac{1}{N} \sum_{j=0}^{N} y'[j]y[j]\right)^{\frac{1}{2}}}{||\omega||_\infty}. \tag{37}$$

The lower the gain value is, the smaller is the effect of disturbance and the better is the robustness. Considering this, an expression for the minimum successful transmission rate $r_{\min}$ is derived for which the system is stable. Later, Saha *et al.* [58] have derived that the successful transmission rate at which the performance is optimal is either the minimum possible rate $r_{\min}$ (constrained by stability) or the maximum possible rate $r_{\max}$ (constrained by network availability).

In addition, Goswami *et al.* [57] have studied a restriction on deadline misses over a finite horizon (k samples) and proposed an exponential stability criterion in terms of allowable deadline misses. This criterion forces a bound on the rate at which the system must approach the equilibrium state from a given initial state which is mathematically given by [60]

$$||x[k] - x_e|| \leq c_1 + c_2 \beta^k ||x[0]|| \tag{38}$$

where $c_1 \geq 0$, $c_2 > 0$, and $0 < \beta < 1$. Furthermore, Goswami *et al.* [57] have considered a performance measure tuple $\{S, \chi\}$ as follows:

$$S \geq \frac{||x[k+\chi] - x_e||}{||x[k]||} \times 100\%. \tag{39}$$

This implies that χ samples after the disturbance has arrived, at least $(100 - S)\%$ of the disturbance is rejected. Subsequently, the number of samples κ_s that can be missed out of χ samples can be calculated to meet the performance requirement given by S.

5) Finite Precision Arithmetic: Traditionally, controllers are designed by solving a set of differential equations in a way that stability and performance requirements are met. At the model level, no assumptions on the arithmetic precision of computations are made. However, for an embedded implementation, the control law is calculated by a processor that allows only fixed precision arithmetic operations. As a result, safety and performance guarantees are no longer valid due to quantization errors. Moreover, it may so happen that the system constantly oscillates around the equilibrium state.

To address this, there has been work on quantization-error-aware verification of control software [61]–[64]. The goal is to verify that the final implementation results in a practically stable system, i.e., the final state of the system is within a bounded region of the equilibrium state.

Furthermore, Majumdar *et al.* [65] have proposed to synthesize controllers by cooptimizing the LQR cost

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

function and the quantization error, thereby constructing a Pareto front of the two objectives. The proposed approach can be partitioned into two stages. In the first stage, an upper bound on the quantization error in the computation of control law is calculated. For a given implementation and the bounds on plant states, the range of each controller variable is analyzed and the bitwidths allocated accordingly. For given bitwidth allocations, the maximum quantization error in an arithmetic operations is calculated. Subsequently, since the computation of the control law involves multiple arithmetic operations, the quantization error accumulates accordingly. In addition to the quantization error in the computation, the approach also considers quantization error in measurement, for which the bound will simply depend on the allocated bitwidths. The errors are modeled as disturbance in the observer dynamics and the feedback gain codes. Subsequently, in the second stage, a multiobjective optimization problem is formulated and solved using particle swarm optimization (PSO) [66]. The objectives considered are LQR–LQG quadratic cost function [see (13)] and $\mathcal{L}_2$ induced gains from input disturbance to output [see (14) and (15)], where the input disturbances are the two quantization errors. Finally, the PSO algorithm generates several Pareto points that depict the tradeoff between performance and quantization error.

B. Controller-Aware Platform Design

As discussed in Section III, traditional approaches of platform design are based on timing requirements, which might be overly conservative. In the context of CPS, a more appropriate approach is to consider the control properties of the system at the platform design stage. These control properties include stability, QoC, and robustness, among others. This design approach will ensure either 1) optimal QoC of the overall system for a given platform resource; or 2) minimize resource usage while satisfying all performance and stability constraints. Toward this, in this section, we will discuss some of the platform design approaches that have been proposed in the literature.

1) Stability- and Performance-Aware Platform Design: In Section IV-A, we have discussed several platform characteristics that may influence the stability and performance of control systems if not considered at the controller design stage. Most of these platform characteristics are configurable, subject to certain constraints. For example, the sensing-to-actuation delay and jitter can be manipulated by changing the schedule parameters to the extent that the overall system still remains feasible. Now, given a set of controllers, how to determine platform parameters considering their influence on stability and performance is an important research problem.

Toward this, Mancuso *et al.* [67] have addressed the problem of calculating the priorities and periods of the control tasks. It is assumed that several control loops run on a given platform with FPP scheduling. An optimization problem is also formulated with the objective of maximizing the overall QoC. Here, the control performance of each loop is measured by the LQR cost function approximated as a linear function of sampling period and delay. Solving the optimization problem is difficult due to the nonlinear dependency of the WCRT of a task on priority assignment. Consequently, a branch and bound technique is proposed to solve the problem. Furthermore, Aminifar *et al.* [68] have proposed a scalable algorithm to determine the priorities and periods of control tasks taking into account both worst-case delay and jitter. However, only the stability of control loops is considered instead of optimizing the overall QoC. The impact of delay and jitter on stability of a control loop is studied using the Jitter Margin Toolbox [69]. Finally, a stability condition in terms of delay and jitter is derived as a linear inequality.

In the same vein, Aminifar *et al.* [70], [71] have studied server-based scheduling of control tasks. The server-based scheduling of real-time tasks is introduced to achieve isolation, i.e., misbehavior in one task will not effect the others. In [70] and [71], Aminifar *et al.* have defined periodic server as a tuple of budget Q, period P, and deadline D. It is assumed that each server is assigned to only one control task. The server ensures that Q amount of processor time is allocated to the assigned task in each period P before the deadline D. Here, the constraint is that the resource reserved by a server must be greater than or equal to the resource requirement of the assigned task. A task runs only when its dedicated server allocate processor time to it. Now, given the period and the best and the worst case execution times of a control task, it is possible to derive the best and worst case response time of the task. Corresponding to these values, nominal delay and jitter can be calculated which can be subsequently analyzed to determine the stability of the system. It may be noted here that in this server-based approach, each task can be analyzed independent of other tasks running on the same processor. The idea is to calculate the parameters of each server taking into consideration the stability and the worst case performance of the assigned control loop. Although, the server-based approach is pessimistic in nature, it offers compositionality and isolation, which are important aspects to guarantee safety in control systems (i.e., model-level guarantees are preserved in an implementation).

2) Robust-Control-Aware Platform Design and Verification: In Section IV-A4, we have mentioned that any control-loop has some inherent robustness. Hence, occasional deadline misses or packet drops may not make the system unstable or violate its performance requirements. We have also mentioned previous work that derived the minimum rate of ideal closed-loop action [(m, k)-firmness] necessary to ensure stability and performance of the system. Given these rates for multiple control loops, the task of an embedded systems engineer is to implement the corresponding controllers on an embedded platform such that these constraints/rates are satisfied.

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

In this context, Majumdar *et al.* [59] have proposed a static scheduling algorithm that satisfies the (m, k)-firmness of all the controllers and at the same time tries to maximize the overall QoC of the system. The algorithm solves a multiobjective optimization problem where (m, k)-firmness is treated as one of the constraints. The optimization objectives are the QoCs of all the control systems. Next, the multiobjective optimization problem is transformed into a single-objective problem by a weighted combination of the QoCs. Here, the weights are assigned in a way such that the control loop, which is the most sensitive to the rate of packet drops, has maximum influence on the objective. Consequently, the solution to the problem will be an undominated one in the multiobjective space.

Another body of research that studies the impact of missed control action on stability and performance is platform-aware formal verification of control software [55], [56], [72], [73]. Here, an embedded platform architecture is represented as a network of time-stamped event count automata (TS-ECA) where each message is stamped with a time as it moves from one buffer to another. Moreover, the (m, k)-firm rule can be formulated as a linear temporal logic (LTL) formula where all possible combinations of allowed misses can be represented. Subsequently, the network of TS-ECAs are model checked to verify satisfiability of the LTL formula. Thus, a control software can be formally verified using model checking to ensure that all the control applications satisfy the constraints on their deadline misses.

Furthermore, Behrouzian [73] has proposed an analytical technique to verify (m, k)-firmness. The authors assume that the control tasks are running on a processor according to a TDMA scheme where each application is assigned a dedicated slot to execute. For such an architecture, given a sampling period between the best and worst case response time, the proposed technique can calculate an upper bound on the percentage of dropped samples. The estimation is based on an analysis of a finite regular window with the assumption that tasks arrive periodically. This technique is faster than timed-automata-based approaches that were proposed earlier.

3) Application-Criticality-Aware Platform Design: Mixed criticality systems are becoming increasingly more common; here, applications of different criticality share the same resource. These applications have different requirements, e.g., hard real-time applications have strict timing requirements, while control applications have stability and performance requirements. Thus, techniques discussed in Section IV-B1 are not applicable in a straightforward manner for platform design in such cases.

Toward this, Wu *et al.* [74] have considered control and noncontrol tasks running on a processor with EDF scheduling strategy. Upper and lower bounds are assumed on sampling periods and relative deadlines of all applications. Here, the bounds for control applications are derived from stability and performance requirements. Subsequently, an optimization problem is formulated and solved where the variables are the periods and deadlines of all tasks. The objective is to maximize the overall QoC of the system while satisfying the deadline constraints of noncontrol tasks. The QoC metric of each control loop is the loss in LQR cost due to sampling period and output jitter.

Later, Schneider *et al.* [75], [76] have considered a similar problem for an FPP scheduling scheme. A multilayered scheduling approach is proposed. In this approach, real-time and control applications form the top and bottom layers, respectively. The scheduling algorithm starts with the worst priority and iteratively approaches the best priority where in each iteration one task is assigned the worst priority from the available set. In each iteration, the algorithm first tries to find a task with the longest deadline from the pool of unassigned tasks in the top layer. Correspondingly, the WCRT of the task is calculated if the current worst available priority is assigned. If the deadline is satisfied, the task is assigned the priority. Otherwise, the algorithm then tries to find the controller which remains stable and for which the performance degradation is minimum if assigned the worst available priority. If such a controller is found, then the priority is assigned. As performance degradation is a nonlinear function of sensing-to-actuation delay, the obtained overall QoC may not be the guaranteed optimum. However, this algorithm is at least more analytical than a deadline monotonic scheme and will be more useful in mixed criticality scenarios.

4) Using Hybrid Architectures: Time-triggered architectures (TTAs) have inherent timing determinism. This makes it easier to implement control algorithms on them. However, TTAs may not be resource efficient. For example, in TDMA, if a slot is allocated to a message it will be consumed irrespective of whether any data are sent and it cannot be reallocated to a different message. As a result, time-triggered slots are judiciously used. On the other hand, event-triggered architectures (ETAs) offer higher resource efficiency but timing nondeterminism makes it difficult to implement control algorithms on them. If used, the worst case delay values have to be considered, which makes a design pessimistic. Moreover, a control law need not be computed at high frequency if the controlled plant is in the equilibrium state [77], which makes TTAs unsuitable candidates if resource usage is to be maximized. Ideally, the resource allocated to a controller could be dynamically determined based on the state of the system. This fact has been exploited for hybrid platform architectures that support both TT and ET task execution and message transmission [78].

Examples of work along these lines include that of Goswami *et al.* [79]. Distributed control applications using FlexRay communication bus are studied. A controller is proposed which can switch from an event-triggered mode

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

to a time-triggered mode based on the occurrence of disturbances. The performance of such a hybrid implementation is almost equivalent to one in which only TT communication is used.

Later, Masrur *et al.* [80], [81] have proposed worst case performance analysis techniques for such a setup. In particular, these works calculate the minimum number of TT slots required such that all the applications satisfy the corresponding performance requirements. The analysis consists of two nested layers. The inner layer considers the case of a single slot assigned to several applications and investigates if such an assignment is safe. Here, two different arbitration policies are considered: 1) FPNP [80]; and 2) FPP [81]. In FPNP, when an application gets a TT slot, it stays there for a certain dwell time which is enough to stabilize the system even in the worst case. In FPP, a lower priority application may be preempted by a higher priority one. Preemption is only allowed at a point where the higher priority application if not given a TT slot will not satisfy the performance requirement. However, a retransmission cost is considered due to preemption. For both arbitration policies, worst case analysis can be carried out by extending the WCRT analysis for nonpreemptive deadline monotonic scheduling schemes. In the outer layer, a slot provisioning algorithm maps applications to slots one by one using a customized first fit heuristic. It uses the inner layer to determine the feasibility of mapping the current application to the slots which are already provisioned. If it is not feasible, then a new slot is added. It may be noted that such hybrid implementations may require runtime reconfiguration of the underlying platform [82], [83], which has been addressed in [84].

Recently, Balszun *et al.* [85] have proposed a control algorithm for mixed TT and best effort communication. The algorithm uses TT communication to guarantee worst case performance requirement while exploiting best effort communication to improve the performance and thereby achieving higher average performance.

5) Event-Triggered and Self-Triggered Control: Traditionally, a controller is implemented on an embedded platform as a group of tasks dispatched periodically. However, when the system is in steady state it is not necessary to apply control inputs as frequently as when the system is in a transient state. Moreover, periodic execution of control tasks at high frequency may be very expensive in resource-constrained embedded systems. In this context, two new implementation techniques have come up: event-triggered [86] and self-triggered control [87]. In event-triggered control, the control law is executed only when a certain error threshold is violated based on the current system states. This decision is taken by a feedback scheduler which also monitors the system states. On the other hand, self-triggered controllers calculate the current control input and also the next sampling instant based on the current system states. Therefore, they do not require an additional feedback scheduler.

In both cases, stability and performance analysis techniques that are based on a known sampling period are no longer valid. Tabuada has derived a constraint for task activation such that the system is stable with respect to measurement noise [86]. This constraint is based on the current state and the difference norm between 1) the current state; and 2) the state used in the calculation of the last control input. Here, nonlinear systems given by $\dot{x} = f(x, K(x+e))$ have been studied, where e is the measurement noise and $u = K(x+e)$ is the control input. Notion of input-to-state stability (ISS) in this case is given by [88]

$$||x[k]|| \leq \beta(x[0], k) + \gamma(\sup u[k]{:}k \in Z^*) \tag{40}$$

where the functions β and γ are of class $\mathcal{KL}$ and $\mathcal{K}$, respectively. Now, a closed-loop system is ISS with respect to the measurement noise if there exists an ISS Lyapunov function $V(x)$ such that $V(x)$ is continuous and

$$\begin{aligned} &\alpha_1(|x|) \leq V(x) \leq \alpha_2(|x|) \quad \forall x[0] \in \mathbb{R}^n \\ &\frac{\partial V}{\partial x} f(x, K(x+e)) \leq -\alpha_3(|x|) + \sigma(|e|) \end{aligned} \tag{41}$$

where $\alpha_1, \alpha_2, \alpha_3, \sigma \in \mathcal{K}_\infty$-function.

Based on the derivation in [86], Anta *et al.* have calculated the next activation time for self-triggered control that renders the closed-loop system ISS [87]. In addition, there have been other works that determine triggering conditions or trigger instants such that the system is stable, e.g., 1) stable in the sense of Lyapunov [89]; and 2) uniformly globally asymptotically stable [90]. Furthermore, a recent work also considers ISS taking output quantization [91] into account.

Besides safety, it is also important to consider control performance while designing event-triggered controllers. Toward this, Martí *et al.* [92] have proposed synthesizing triggering instants such that resource usage is minimized while maintaining optimal performance. The optimal performance is determined by the corresponding LQR cost for which the controller is designed. The trigger synthesis requires solving an online optimization problem which may be computationally expensive. Velasco *et al.* [93] have suggested approximations for solving the optimization problem in order to reduce computational complexity.

On the platform side, the problem with event-triggered or self-triggered control is that it is difficult to accurately analyze the schedulability of the system as task activation patterns are not known in advance. Toward this, Velasco *et al.* [94] have proposed a schedulability analysis of event-driven controllers. Here, the worst case activation is based on an assumed minimum interevent time. Later, Aminifar *et al.* [95] have analyzed the request bound function of a self-triggered controller for the worst case request pattern. This approach starts by discretizing

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

the space and then calculating the maximum possible sampling interval for each polytope such that the open-loop system is stable. Subsequently, a state-transition graph is constructed from which the worst case request pattern can be obtained.

Besides stability, performance, and schedulability analysis, another important component of event- or self-triggered control is its implementation, which has attracted quite some research. For example in sensor/actuator networks, Mazo and Tabuada [96] have proposed to employ a tree wave-algorithm for computing control inputs and for evaluating triggering conditions (in event-triggered control), and for calculating trigger times (in self-triggered control). Here, each sensor node computes its contribution to gain and error. Furthermore, for several control applications sharing a common platform resource, Samii *et al.* [97] first analyze the worst case schedulability based on minimum interevent time and calculate an upper bound on interevent time to ensure worst case performance. Subsequently, a dynamic scheduler that explores several schedule options at runtime is proposed. Out of these, one is selected based on the desired tradeoff between control performance and processor utilization.

One challenge in implementing self-triggered controllers is the computation time for the next activation, which may sometimes undermine the advantages of using self-triggered control. For this, Saha *et al.* [98] have proposed a hybrid implementation approach. Here, for a certain discretized region around a given operating point, trigger times are precalculated and stored in the cache. Now, if the current state is within the precalculated region, trigger times can be just fetched from the cache. However, when it is not in the cache, then the trigger time calculation task is dispatched with a very low priority such that it does not interfere with other control tasks. In the worst case, the control loop goes back to periodic execution. Furthermore, in the wireless network domain, Araújo *et al.* [99] have shown how event-triggered control can be implemented over the IEEE 802.15.4 standard. In summary, although periodic implementation has been preferred for simple design and analysis techniques, event-triggered control has also become popular for better resource efficiency.

V. CONTROL-PLATFORM COSYNTHESIS

The design approaches, discussed in Section IV, are far from being holistic. These approaches focus on the design either on the control side or on the platform side and consider the parameters on the other side as given. Thus, the opportunity for optimization is very restrictive. Toward optimal design of CPS, semantics-preserving control and platform cosynthesis approaches have emerged in recent years. In this section, we will review these approaches for both single-processor and distributed systems. Furthermore, we will present a general cosynthesis framework.

A. Existing Cosynthesis Approaches

1) Single-Processor Systems: Here, the problem setting consists of a number of applications mapped on a shared processor. It is required to compute the control law and the task schedules for each of the applications. In this setting, one of the earliest approaches on integrated controller design and scheduling is proposed by Aminifar *et al.* [100]. The proposed approach optimizes the expected control quality while guaranteeing the worst case performance. Here, an application is represented by an acyclic graph of tasks. The execution time of each task is assumed to follow a distribution between the best case and the worst case values. The jitter is modeled as some stochastic disturbance. The sensitivity of a control loop is measured as gain from the stochastic control input to output. The worst case sensitivity analysis involves finding the lowest priority group to which the application must be assigned such that the worst case performance is ensured. Now, the proposed cosynthesis approach employs an iterative scheme. In each iteration, the applications are assigned periods based on a hybrid search technique. Subsequently, for given sampling periods, applications are first grouped based on worst case sensitivity analysis of the control loops. Applications with same sensitivity are grouped together. Now, based on delay and jitter analysis, LQG controllers are synthesized for expected value of delay to tackle uncertainties. Then, within each priority cluster, control applications are assigned concrete priorities such that the expected performance is optimized.

For server-based controller implementation, Aminifar *et al.* [101] have proposed extension to the earlier work [70], which is discussed in Section IV-B1. Here, controller-server codesign is considered and thus a controller is designed together with the dedicated server parameters. In the same vein, Valencia *et al.* [102] have presented a codesign framework as an extension to the work [42] (which is reviewed in Section IV-A3). A tradeoff analysis between resource utilization and QoC is offered for controllers implemented on a composable platform. Furthermore, Xu *et al.* [103], [104] consider partial codesign where task priorities are given, however, task dispatch periods need to be calculated along with the controllers. These works determine the perturbed dispatch period by exploiting the periodic delay pattern such that a finite and short hyperperiod is obtained. Subsequently, LQG controllers are designed considering the delay pattern and also the distribution of execution time.

2) Distributed Systems: For distributed embedded controllers, communication network schedules must also be calculated during the cosynthesis. Samii *et al.* [105], as one of the first few, proposed a control-platform cosynthesis approach for distributed systems. Both static–cyclic scheduling and priority-based scheduling are considered on the processor and on the bus. The design follows an iterative approach. In each iteration, sampling periods of all the applications are first selected according to a genetic algorithm. Now, for a

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

specific set of periods, in the case of static–cyclic scheduling, the schedules are synthesized and delay distributions are derived. The control gains are synthesized for each application based on the corresponding expected delay value, while the QoC is computed based on the stochastic delay using Jitterbug toolbox [106]. For priority-based scheduling, the different priority sets are iterated over, and for each set, the delay distributions are obtained through timing analysis. Correspondingly, the control gains and the associated QoC values are computed. Later, Samii *et al.* [107] extend their work with specific characterization of the FlexRay parameters. In the same vein, Aminifar *et al.* [108] have extended the approach developed for single-processor architecture [100], as mentioned in Section V-A1, with added complexity due to schedule computation for the communication bus (CAN bus).

Furthermore, Schneider *et al.* [109] have proposed a method to codesign controllers and a FlexRay-based distributed system. TT scheduling scheme is assumed on the processor and FlexRay protocol is considered on the bus. The whole approach is divided into three stages: the controller design, the platform constraints, and the platform configuration synthesis. In the first stage, each controller is designed with the sampling period selected from a preconfigured set (given by the FlexRay protocol), such that the control performance is optimized. In the second and third stages, the platform parameters are synthesized according to the selected sampling period and one sample sensing-to-actuation delay. This paper addresses the specific semantics of the FlexRay protocol and consider different performance metrics such as settling time and a modified quadratic cost function. The cost function is as follows:

$$\sum_{0}^{N} \int_{kh}^{(k+1)h} [\lambda u[k]^2 + (1-\lambda)e(t)^2]dt. \tag{42}$$

It is to be noted here that the cost for each discrete step is integrated over the sampling period h which is different from the quadratic cost usually considered in the literature. This is required to compare controllers designed for different sampling periods based on this metric. Therefore, the quadratic cost will be calculated until a certain given time T_R from which the number of samples N for a given sampling period h can be calculated as T_R/h.

Later, Goswami *et al.* [110] have assumed variation in delay during the controller design instead of one sample delay. Optimal controllers are designed for selected sampling periods and different sensing-to-actuation delays. The control performance curve depending on the period and delay is then discretized and approximated with piecewise linear functions. This function is considered together with the platform constraints into an ILP problem. The whole scheme then iterates through different combinations of the sampling periods and decides on the configuration that optimizes the overall system performance.

Recently, Roy *et al.* [111] have also considered FlexRay-based distributed systems. In aforementioned approaches, iteration over different sets of sampling periods serves as the outer loop of the cosynthesis problem. In contrast to them, Roy *et al.* [111] first design prospective optimal controllers at all possible sampling periods for each application. Then, the tables of prospective controllers and their corresponding performance are considered in the cooptimization problem. Here, a nested two-layer hybrid optimization scheme is proposed to generate a Pareto front for the objectives of overall control performance and communication resource utilization. This is also one of the first works to consider design objectives from both control and platform sides.

B. General Cosynthesis Framework

Fig. 6 shows the general design flow for control-platform cosynthesis. The cosynthesis methods usually start with modeling of the control systems and the underlying embedded platform. The model on the control side is typically the system dynamics of the control plants and limitations of the physical devices such as the actuator limit. Most of the related research works focus on LTI systems [103], [105], [109]–[111]. System modeling further includes the controller type (e.g., state feedback). On the platform side, the model includes the relevant aspects of platform architecture, for example, whether it is a single processor system [100]–[104], [112] or a distributed system [105], [108]–[111], [113], [114]. The platform model also incorporates the scheduling schemes on the processors (e.g., TT scheduling [102], [105], [109]–[111] or priority-based scheduling [100], [103]–[105]). Specifics of the bus protocols must also be considered in the platform model (e.g., FlexRay [107], [109]–[111]; CAN [107]). Furthermore, the

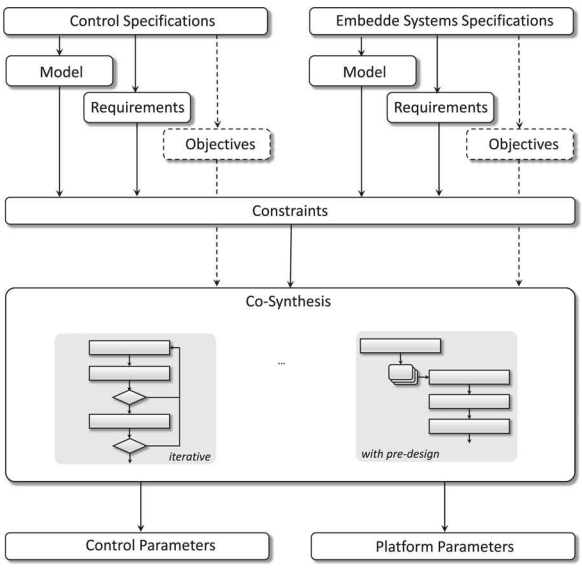


Fig. 6. ***Design flow for control-platform cosynthesis.***

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

task partitions of the control software and the execution times of the tasks are also modeled. In terms of execution time, most approaches assume that a model is available, be it WCET [109]–[111] or a distribution of the execution time [105], [108]. Usually the task mapping is also considered as provided by the specification.

Besides the models, design requirements can also be specified. They can come from both control and platform sides. Requirements from the control side are typically related to the control performance (e.g., settling time [109], [111]; cost function [105], [108], [110]) and stability. They usually reflect the QoC of the designed controller in terms of transient response, steady-state error, and energy consumption. On the platform side, the requirements may include, for example, upper bounds on processor utilization and bus load. These requirements are specified due to reasons such as limited resource bandwidth, reliability, extensibility of the system or certification requirements.

Subsequently, constraints can be formulated from the models and the requirements. On the control side, the constraints may include some minimal performance requirement and limits of the physical devices such as the input saturation. On the platform side, the constraints come from the models of the scheduling scheme, the limitations of the resources and the design requirements. The constraints are usually expressed mathematically.

Typically, the goal of a cosynthesis technique is to find a valid parameter set that satisfies all the constraints. In many a case, multiple such parameter sets can be found. Moreover, based on design requirements, optimization objectives may also be considered in the cosynthesis problem. In that case, one or more parameter sets that optimize the objectives, while being feasible, must be synthesized. Such objectives include, e.g., control performance [105], [108], [110], [111], and resource utilization [111]. Often multiple conflicting objectives are also considered. In such a scenario, tradeoff between the objectives must be explored, for example, by constructing a Pareto front [111].

Once such a synthesis problem is formulated, it needs to be solved efficiently. Assuming that the constraints are precise characterization of the semantics of the closed-loop system and the embedded platform, the design problem boils down to finding a valid parameter set that satisfies all the constraints. Common approaches used to solve the optimization problems include ILP [110], SMT, metaheuristics such as genetic algorithms [105] and PSOs. However, in the case of control-platform cosynthesis, hybrid techniques are often employed to solve the whole problem due to the complexity. It is often the case that in order to tackle complexity, the whole design problem is partitioned into several subproblems while retaining the feasible regions of design space as much as possible. Now, different subproblems may be solved using different approaches. For example, control performance often depends nonlinearly on design parameters such as control gains and closed-loop delay, and thus optimal control design problem may be solved using metaheuristic algorithms. On the other hand, TT schedule synthesis problem fits very well into the linear programming model while priority-based scheduling may require a heuristic search. Thus, different problem settings may call for different hybrid approaches and this imposes a major challenge towards considering more complex CPS architectures in the future.

VI. FUTURE OUTLOOK AND CHALLENGES

Although there have been efforts in developing cosynthesis techniques for CPS, there are still a number of open problems. In this section, we will discuss these problems and classify them as follows. 1) Architectural aspects, such as memory hierarchy, heterogeneous networks and multicore processors, must be considered in the cosynthesis. 2) Complex closed-loop dynamics, such as input saturation, time-variance and nonlinearity, can be modeled. 3) Besides being safe, CPS must also be secure, reliable, and energy efficient. We also identify several challenges toward these extensions, e.g., problem complexity, certification, and lack of tool support.

A. New Architectural Considerations

Most of the available cosynthesis techniques are specific to certain architectural consideration and cannot be trivially applied when considering additional platform details. New techniques are required to deal with them. Here, we will discuss three such architectural aspects which are becoming relevant in the context of embedded control systems.

1) Memory Hierarchy: Memory architecture plays an important part in determining the cost and size of a processor chip. Larger the storage capacity of faster memory is, higher is the cost and space required. Typically, a processor has access to several levels of memory, such that, faster the memory access speed is, smaller is the capacity. For the sake of simplicity, let us assume two levels of memory, i.e., a faster on-chip cache memory and a slower off-chip flash memory. The access speed of cache is many times faster than the flash. When a processor executes an instruction first it checks the cache and if present executes directly from cache. This is called cache hit and is very fast. In case the instruction is not in the cache, then the processor brings it from the flash to the cache and executes it. This is a cache miss. However, the next time when the instruction is called again, if it is still in the cache, it will result in a cache hit.

Consider a scenario when several control codes are running on a processor in round-robin fashion. If the cache is larger then lower is the probability of cache misses. It is desirable to have more number of cache hits than misses for control applications. This is because longer access time of cache misses will result in higher WCET, and therefore,

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

higher closed-loop delay. This in turn results in performance degradation and in worst case system instability. Thus, to improve QoC of closed-loop systems it is desirable to have larger cache, however, it will increase the cost of the system. For cost-sensitive systems, the question is can we achieve better QoC by exploiting certain characteristics of memory hierarchy and management. Along similar lines, the embedded systems community have exploited the cache reuse by code rearrangement during compile-time [115]–[117] or runtime [118]. However, this is not applicable for control applications because it is difficult to analyze the timing properties for such compile-time rearrangement in the design stage.

Until recently, to the best of our knowledge, there has not been any work on memory-aware design of embedded control systems. In [43], Chang *et al.* have proposed a novel approach to maximize cache reuse without losing timing determinism. In this approach, the schedule is still round robin, however, in each round a controller is executed multiple times in succession instead of once. In each round, the execution time of the first instance will be the same as in the case of standard round robin. However, the second and subsequent instances take less time due to cache reuse. This is because some part of the code can be expected to be in the cache if the code size is comparable to the cache size. In this scheme, the controller executes with nonuniform sampling with average sampling period less than the standard round-robin case. Consequently, we can expect that it is possible to design a controller with better performance for the reduced average sampling period. Chang *et al.* [43] have introduced a technique to design a controller for such a case. However, it does not consider determining a schedule which optimizes the overall QoC of the system. Thus, we believe it is possible to extend this idea and to do a cosynthesis of controller and platform schedules offering tradeoff between QoC and cache size.

Furthermore, modern processor chips are equipped with scratchpad memory in addition to cache. Scratchpad memories are as fast as cache but are programmable. A software code can determine which memory block to fetch and store in the scratchpad. We imagine that scratchpad-centric design of embedded controllers will be an important research topic in the future. The idea is to develop efficient scratchpad allocation algorithm to reduce code execution time and correspondingly design the controller to improve QoC. Here, program analysis techniques can help identify frequently invoked part of the code. These parts can be stored in the scratchpad thus optimizing the program execution time.

2) Heterogeneous Networks: The cosynthesis techniques for distributed CPS discussed in Section V-A2 predominantly consider a single bus. However, modern CPS such as automotive systems typically consist of several bus clusters connected via gateways. Each bus cluster serves a certain functional domain, e.g., FlexRay for chassis, high speed CAN or FlexRay for powertrain, low speed CAN and LIN for body, MOST and Ethernet for infotainment. Today, with increasing demand for advanced driver assistance systems (ADAS) and autonomous driving, the need for interdomain interaction and communication has also increased. Control applications across heterogeneous network have also emerged. Designing such applications is not a straightforward extension of existing techniques. The problem is that different communication protocols have different timing models, and therefore, require different analysis framework. For example, CAN employs FPNP scheduling while FlexRay uses TDMA for static segment and flexible TDMA for the dynamic segment. Designing an application across CAN and FlexRay will require finding TDMA schedules for FlexRay messages and priorities for CAN messages. Moreover, interdomain communication also involves transmission of messages across communication gateways. This requires additional timing analysis and buffer characterization for gateways. Therefore, the design of applications across different network domains leads to increase in design dimensions and a more complicated timing analysis.

In this context, Glaß *et al.* [119] have proposed a hybrid analysis framework where different timing analysis techniques can be composed together to determine, for example, end-to-end delay of a message. However, control applications over such heterogeneous networks are not yet considered. Control properties depend nonlinearly on timing properties, thereby adding to the complexity of the problem. Therefore, cosynthesis of controllers, heterogeneous network schedules and gateway parameters will be challenging to explore.

3) Multicore Processors: Multicore processors are becoming increasingly more popular in embedded systems due to their higher instruction throughput as compared to single-core processors. High throughput is achieved through simultaneous processing of multiple tasks in parallel on different processing cores. However, the cores may share different hardware components, e.g., memory, I/O, and on-chip bus. Simultaneous access to these shared resources may result in contention. Access to shared resources if not properly managed or synchronized may result in nondeterministic timing behavior which is difficult to analyze.

There have been few works addressing this problem from both hardware [120], [121] and software [122] perspective. Recently, Tabish *et al.* have proposed a scratchpad-centric solution [123]. They have assumed that each core has its own scratchpad with size greater than any two tasks running on the core. The access to main memory is with TDMA-based schedule via a direct memory access (DMA). The idea is that the codes for the next task can be prefetched in one half of the scratchpad while the current task is running from the other half. In this approach, there is no resource contention. Additionally, the WCETs of the tasks are also reduced

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

as the instructions are already in the scratchpad before execution. Consequently, control codes can be mapped on such an architecture to achieve higher QoC. However, large-sized and dedicated scratchpad for each core substantially increases the cost of the system which may not be acceptable in cost-sensitive domains. Therefore, we believe there is a possibility of using smaller dedicated scratchpad or shared scratchpad. And program analysis techniques may be used for appropriate memory partitioning and code mapping. Program analysis integrated with cosynthesis of controllers, processor, and memory access schedules may result in an improved overall QoC and better load balancing across the cores.

B. Complex System Dynamics

The cosynthesis approaches developed so far mostly consider LTI systems. However, systems often demonstrate complex dynamics with time variance, input saturation, and nonlinearity. Although there are works that study these aspects from control theory perspective, almost none actually evaluates the possibility of a true cosynthesis for such system dynamics.

1) Input Saturation: Primitive control design approaches do not consider any constraint on control input. However, this is not realistic as actuators often have limited range and energy is often an important factor for most control loops. Toward these considerations, model predictive control (MPC) is very popular. Typically, an MPC controller solves a constrained optimization problem online. Solution to the optimization problem gives a set of N control inputs corresponding to N time steps up to a finite horizon. The problem often considers actuator limits as constraints and energy or control quality or a combination of both as an objective. However, MPC is more applicable in process control as it requires considerable amount of time to solve the optimization problem. This is not acceptable for high-frequency machine control software running on constrained embedded platforms.

Recent works such as [124] have proposed approximate solution to the online optimization problem while preserving the guarantees on stability. This facilitates the application of MPC to high frequency control systems. Furthermore, Yao *et al.* [125] have proposed a resource-efficient implementation by exploiting the characteristics of MPC. Here, the actual system state is compared with the predicted state. If the error is more than a threshold then the optimization problem is solved otherwise the precalculated control action is applied. However, to utilize the released processor time, the underlying platform needs to be runtime adaptable. Thus, a complex scheduling algorithm must be cosynthesized along with MPC to ensure processor resource to the application based on requirement. In addition, we believe MPC will find more and more applications in resource-constrained embedded systems. Tradeoff between optimality and computation time can be explored further with self-triggered nonuniform sampling.

2) Time Variance: In control theory, adaptive control techniques were known since many decades. An adaptive controller can manipulate the control gains online based on the changing plant dynamics. Therefore, it can stabilize the system in the event of unforeseen environmental variations. Such a control technique is inherently applicable to time-varying systems. However, these techniques have not been considered for safety-critical systems until recently as it is difficult to quantify the transient performance of an adaptive control loop.

A popular adaptive control technique is model reference adaptive control (MRAC). In this technique, the error between the output of the reference model and the actual output is fed back to adapt the control gains. In order to improve the transient performance, closed-loop reference models are considered of late. Here, the error is also fed back to change the reference model.

In this regard, there have been some works [126]–[129] which quantify the transient performance using $\mathcal{L}_2$ norm of error signals. However, only very few [130]–[132] have actually tried to consider cosynthesis of controller and platform parameters. Voit and Annaswamy [131] have derived an adaptive controller considering network induced delay. Furthermore, Voit *et al.* [132] have considered codesign of adaptive controllers and shared hybrid communication bus minimizing resource utilization while guaranteeing stability.

As a future extension, one can consider a setting where multiple applications mapped on a shared platform. Here, worst case bounds on time variance can be analyzed. Correspondingly, it is possible to cosynthesize platform schedules along with adaptive controllers providing guarantees on stability and performance.

3) Nonlinearity: Nonlinear systems can be widely found in several domains of embedded systems, including avionics and automotive. However, cosynthesis of controllers and platform parameters for such systems is still an open problem. There has been significant progress in the stabilization and control of these systems based on abstraction of platform characteristics. Many related works in this direction are based on concepts such as input-to-state stability [133], [134], small gain theorem [135], passivity [136], and feedback linearization [137].

Fuzzy-model-based analysis and control of nonlinear systems have also received significant attention. Among different fuzzy models, nonlinear systems fit well into Takagi–Sugeno (T–S) models [138]. In such a model, at each sampling time the system is represented as an averaged linear model. Based on T–S models, there have been works that consider network-specific properties such as packet dropout, signal quantization, and time delays. Toward considering packet drops, data loss in T–S fuzzy-based systems is modeled as a Bernouli process. Correspondingly, 1) stability

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

is studied based on a common quadratic Lyapunov function and a fuzzy Lyapunov function [139]; and 2) design of $\mathcal{H}_\infty$ state feedback control [140], static/dynamic output feedback control [141], [142], observer-based output feedback reliable control [143], and model predictive control [144] are proposed. Toward time-delayed nonlinear systems, most works assume parameters such as maximum allowable delay bound [145], maximum allowable transfer interval [146], and delay distribution [147], [148]. Corresponding to these parameters, the stability and control of such systems can be evaluated. Furthermore, there have been some works which consider the impact of network induced signal quantization on T–S fuzzy-based nonlinear systems. They use abstracted platform models based on time-invariant logarithmic quantizer [149] or time-varying quantizer [150] to study stabilization and control problems. However, we may point out here again that all these works start with platform abstractions. Therefore, in the context of CPSs, we can leverage on these advanced theories of fuzzy-based control. We can also consider cosynthesis of platform parameters and controllers by systematically deriving an interface between the platform parameters and the abstraction models.

Another important approach to tackle complex closed-loop system dynamics is formal methods [151]. The embedded control systems naturally fall in the category of hybrid systems where physical plant is in continuous time while the corresponding control action is generated and applied in discrete time [152]. There are several approaches to study such systems [10]. One of the earliest works toward control of hybrid nonlinear systems is by Branicky *et al.* [153]. This work presents a systematic notion of hybrid systems unifying differential equations and automata. Subsequently, it proposes sufficient conditions for optimal control by deriving quasi-variational inequalities. A powerful approach to analyze such systems is the theory of hybrid automata (e.g., timed automata). It can be applied to study system properties according to complex specifications of reachability and safety, given by some LTL formulas [154] or automata on infinite strings. This approach can deal with complex nonlinear systems [155]–[159] and correspondingly can formally synthesize controllers [160]–[162] which are correct by construction. To apply this theory, the first step is finite abstraction [163]–[165] of the dynamical systems. This abstraction can already consider implementation-level imperfections such as delay, jitter, packet loss, quantization error, and limited resource. The abstracted model has a well-defined formal relation with the original system. Subsequently, given a hybrid (or timed) automata model, controller can be synthesized satisfying specifications using algorithmic theory such as two-player games [166], safety games, reachability games, and minimal and maximal fixed point theorem [167]. Guglielmo *et al.* [168] have solved the controller synthesis problem by formulating a bounded model checking (BMC) problem and subsequently solving the problem using SMT solver. The synthesized controller is then refined to be applied to the original system using information such as quantized state, according to the relations derived in the abstraction stage, e.g., bisimulation relations [169], [170], alternating simulation relations [171], and feedback refinement relations [171], [172]. There have been several works addressing the design problem of embedded controllers from the perspective of hybrid systems. However, none of them considers the synthesis of platform parameters. Therefore, the cosynthesis of controller and platform parameters considering hybrid system model is an important problem yet to be addressed and can be a prominent research direction for the future. Nevertheless, an important challenge here is the complexity of the problem and the scalability of the approach considering multiple control applications mapped on a shared platform.

C. Emerging Topics

Security, reliability, and energy efficiency have become important requirements in the design of CPS. It is important to understand how these requirements influence the system safety. Here, we will review the prospect of considering these requirements while design safe CPS.

1) Secure CPS: With modern connected systems, security has become an important concern while designing embedded systems. For example, in [173], Checkoway *et al.* have stated that the security of a modern vehicle can be compromised via a number of interfaces. These include Bluetooth, cellular radio, RFID car keys, and onboard diagnostic. Furthermore, it is reported that a malicious binary can be injected into car electronics via onboard diagnostics to which a WiFi-enabled PassThru device is connected. The malicious binary can then send preprogrammed CAN messages over the vehicular network. It is further claimed in [174] that if a malicious item can enter the internal network of a car, then it can gain control over critical components in a car such as engine or brake.

Thus, it is necessary to add security infrastructure to embedded architecture, e.g., encrypted network messages. However, it is difficult to incorporate cryptographic algorithms on the ECUs and message authentication codes (MAC) on the bus because they consume substantial computation power and communication bandwidth, respectively. These security overheads impact the timing of the applications which in turn may affect system stability and performance. Therefore, it is important to have a cosynthesis approach to the problem where controllers are designed along with cryptographic algorithms with a thorough timing analysis of the complete system.

In this regard, Zheng *et al.* have proposed a cross-layer design framework [175]. This framework combines controller design and implementation along with security integration. Thus, it offers a tradeoff analysis between degree of security, control performance, and platform schedulability. The degree of security is measured as the number of messages

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

that are encrypted. The larger the degree of security is, the lower is the platform schedulability due to resource constraints. Similarly, the interplay between control quality and platform resource usage is often via the choice of sampling periods. The more frequently the control task is invoked, the higher is the performance but the lower is the schedulability. These interdependencies are mathematically formulated into a cosynthesis problem. Subsequently, a simulated annealing algorithm is proposed to solve the problem. In the same vein, we believe that there exists scope to further exploit (m, k)-firmness and nonuniform sampling of control algorithms to achieve a better degree of security while satisfying performance requirements.

2) Energy-Efficient CPS: In electric vehicles (EVs), actuators are powered by in-vehicle battery system and current drawn from the batteries determines the actuation values. However, the actuation values are calculated by the control laws running on the processors. The battery capacity is constrained by weight and volume limitations. Additionally, battery capacity fades due to ageing calculated in terms of number of charging/discharging cycles. The battery ageing also depends on the discharging current profiles according to Peukert's law [176]. In practice, for the sake of reliability in safety-critical systems, a battery is replaced on reaching 80% capacity. Due to high battery cost, it is required to ensure battery usage in a way such that battery lifetime is enhanced. Therefore, modern battery systems typically consist of a battery management system (BMS) for this purpose.

On the other hand, controllers are designed oblivious to battery characteristics except it may consider a constraint on actuator saturation. Therefore, separate design of controllers and BMS may result in a performance gap or inappropriate battery usage where neither is desirable. An obvious alternative will be to design controllers in conjunction with BMS such that a tradeoff analysis between control performance and battery lifetime is possible. Toward this, Chang *et al.* [177] have proposed to design a direct current (dc) motor speed controller taking battery characteristics into consideration. In the same vein, Vatanparvar *et al.* have proposed design of heating, ventilation, and air conditioning (HVAC) control together with BMS [178]. This design improves battery lifetime and driving range of EVs while keeps vehicle climate within acceptable range. On average, their approach has successfully improved the battery lifetime by 14% and reduced the power consumption by 39% compared to state-of-the-art methodologies.

In this direction, we envision a more holistic cosynthesis approach where all the control loops powered by the battery system will be considered together with the BMS. Moreover, in the future, hybrid electrical energy storage (HEES) systems [179] can be considered where multiple storage elements are packed together for better energy efficiency. For such a setting, dimensioning of HEES system may also be integrated in the control/battery system codesign framework.

3) Reliable and Fault-Tolerant CPS: There has been emphasis on reliability of embedded systems or design of fault-tolerant embedded systems. A natural choice is to add redundancy [180]. However, dual redundancy can only help in error detection. This is because any mismatch between two systems only indicates fault but cannot say by certainty which one is faulty. However, triple redundancy may allow uninterrupted functionality as it can be safely assumed that at least two behave correctly. However, it results in more cost and space. Toward fault-tolerant systems, Kim *et al.* have proposed a middleware-based solution [181]. The middleware remaps and reschedules the tasks of the faulty processor to achieve full system functionality. It further considers timing analysis of hard real-time systems to ensure all deadline constraints. However, it is assumed that the system can withstand fault for certain minimum time. This may be critical for control loops running at high frequency as the system can become unstable in no time. On the other hand, control theorists regard fault as some outages in sensors or actuators. In case a fault is detected, it can be mitigated using compensation in the reference input [182].

In a safety-critical control application, it is not desirable that a fault in the underlying embedded platform propagates to the control loop and jeopardizes the safety of the system. Toward this, Goswami *et al.* [183] have considered designing controller such that the control loop is stable to intermittent hardware faults. They have characterized an intermittent hardware fault using intermittent bit flip model. They have calculated from Monte Carlo simulations the probability that a faulty sample is followed by at least N nonfaulty samples. The value of N should be such that the calculated probability is close to 1. In case of a faulty sample, let us assume that the control input $u[k]$ is held. Here, the overall system can be represented as a switched system where the system switches between faulty and nonfaulty instances. For such systems, Goswami *et al.* [183] have suggested to design two controllers. The first one ensures performance under nonfaulty execution while the second one ensures fault recovery within the next N nonfaulty executions after a faulty sample. In this direction, further research efforts are required to consider different fault scenarios while designing the controllers.

Along similar lines, it is also important to consider the impact of processor ageing on control loops. As processor ages, delay in the critical path increases which may call for a decrease in operating frequency of a processor. As a result, the execution time of control tasks will increase which may jeopardize safety. Toward this, Chang *et al.* [177] have proposed to mitigate the performance degradation by redesigning the controller. The redesign exploits energy compensation to meet the performance demand. In the current age, the negative impacts of aggressive technology scaling,

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

e.g., manufacturing variabilities and ageing, are becoming more apparent. Therefore, embedded control system design taking into consideration processor errors or ageing will gain grounds in near future.

D. Challenges

It is established that control-platform cosynthesis is necessary for safe and efficient implementation of embedded controllers. There exist some fundamental challenges that impede future advancement in this direction [184], [185].

1) Scalability and Complexity: It may be noted that the future directions discussed in this section somehow increase the dimensions of the cosynthesis problem. Examples include adding program analysis in case of memory-aware design, gateway characterization in heterogeneous networks, and DMA scheduling in multicore architectures. The problem of complexity and scalability is a big challenge in moving forward. In general, the complexity of a problem grows rapidly with increase in design dimensions, i.e., the number of parameters that needs to be synthesized. In addition, the complexity might also depend on the number and nature of the constraints.

In control-platform cosynthesis, the controller and the platform parameters are synthesized together. This increases considerably the complexity as compared to separation of concerns. Therefore, the related works explained in Section V can only be scaled to a certain size. The second problem is that usually the controller design problem cannot be formulated in a closed-form mathematical representation. Moreover, the tools and methods for controller design are different from those used in platform synthesis. The complexity problem becomes even challenging, if certain objectives need to be optimized. In this case, the solver needs to spend a lot of computation effort on proving the optimality of the solution.

Toward addressing the problem of complexity, efficient design space pruning is required. It may be possible to divide the whole design space into the controller design subspace and the platform design subspace, which are interconnected [111]. This requires a clearly defined interface between the two subspaces so that feasibility region is well preserved. The whole synthesis problem can then be solved using an efficient DSE technique. The technique may consist of heuristic search to choose a value for sampling period, evolutionary algorithms for designing the controller for a given sampling period, and linear programming for computing the schedules. Moreover, the characteristics of specific problem setting can also be exploited to reduce complexity. For example, sampling period of controllers can be restricted to some discrete values, which is enforced by some platform constraints [110], [111].

Furthermore, making a tradeoff between optimality and computational effort can also help the scalability of the approach. For example, Samii *et al.* [105] utilize a genetic algorithm to iterate through sets of sampling periods. However, the algorithm stops as soon as the cost satisfies a certain metric without finding the global optimal solution.

2) Certification and Verification: Industrial CPS, especially safety-critical control systems in domains such as avionics and automotive, need to meet certain national and international safety standards [2], [3]. They have to be certified accordingly by corresponding certification authorities. Traditionally, the certification process involves verification or extensive testing of system properties. This not only consumes a lot of time and effort but is also expensive.

Toward addressing this issue, model-based design approaches are popular which are based on accurate mathematical model of the system. Specifications expressed as mathematical expressions can be formally proved. Since cosynthesis techniques are model based, the synthesized control and platform parameters are correct by design. However, the codes generated from the models that will run on the processors may not preserve the model-level guarantees. This is due to some optimization in the compiler. Thus, further verification or testing of generated codes or compiler to prove satisfaction of safety requirements is necessary.

Over the years there have been considerable research efforts in systematic testing and verification of embedded codes. However, it is far from being effective for industrial systems. On one hand, verification requires formal mathematical proofs for correct system behavior. It employs tools such as model checking [186], [187] and theorem proving [188]. However, these tools are not scalable to industrial-sized embedded code. On the other hand, testing which usually examines system behavior for a finite set of inputs and parameters, e.g., [189], [190], does not guarantee safety in all possible scenarios. Moreover, it is difficult to achieve substantial coverage owing to the exponential increase in scale and complexity of modern embedded systems. Thus, verification or testing of complex embedded control software is an important aspect to be considered in the coming years to comply with safety certification standards.

3) Control Design and Optimization Hurdles: In controller design, the emphasis is not only on stability of closed-loop systems but also on optimal control. Therefore, for given resource constraints the requirement is to maximize QoC. However, for certain performance metrics, such as settling time and overshoot, it is difficult to come up with closed-form expressions. There does not exist any closed-form standard framework for optimal control. Therefore, often exhaustive search of design parameters, such as system poles, is employed to design an optimal controller [111]. However, for higher system orders, this approach is not scalable. Therefore, more scalable heuristics or optimization techniques are required to be developed to synthesize optimal control parameters. On the other hand, although LQR/LQG techniques gives optimal control for

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

a given cost function, determining the weight matrices which correctly represent the desired performance measure is challenging.

Moreover, considering that control performance is nonlinearly dependent on timing properties, it is difficult to integrate the controller design and the platform design problems in a closed-form mathematical formulation. Therefore, cosynthesis approaches often iterate through all feasible combinations of sampling periods to determine the optimal design configuration [105], [110]. However, based on specific platform characteristic, efficient design space pruning may also be possible. For example, Roy *et al.* [111] predesign for each application optimal controller at each possible sampling period. Consequently, in the optimization stage, it considers only those sampling periods for which the predesigned controllers satisfy the performance requirements. This is only possible as the choice of closed-loop delay is assumed to be constrained for a selected sampling period. Nevertheless, different problems may offer different opportunities for design space pruning and it is challenging to identify them.

Furthermore, the related works mostly consider that the type of controller is specified and only one type is used for all the applications. However, it is interesting to consider a system where different applications may need different control strategies. For example, time-variant plant dynamics may require adaptive control while input saturation may necessitate the use of MPC. Thus, for a heterogeneous set of applications, it is challenging to design the complete system using a single framework. As different types of controllers are designed differently, it is difficult to combine them in a single design problem. On the other hand, separation of concerns may lead to an inefficient design.

4) Toolchain Support: In industry, the design and implementation of CPS follow strict procedures from requirements to test and integration. Therefore, these steps are supported by standard and reliable commercial-off-the-shelf (COTS) tools, so that products can be developed in a systematic way. Traditionally, controllers are designed in MATLAB/Simulink and are provided as a black box to the embedded systems engineer with defined interfaces. The latter then uses platform design tools, e.g., Metropolis [191] and Metro II [192]. These tools support the synthesis of platform parameters, such as cache sizes, scheduling algorithms, and schedules, followed by the generation of final software implementation. However, these tools are restricted to the platform synthesis and do not consider the control aspect. Therefore, the algorithms described in Section V need to have well-defined interfaces with the COTS tools for controller and software codesign and implementation. We believe that a prerequisite for the applicability of cosynthesis methods in industrial systems is a systematic and possibly standardized design flow and toolchain support.

VII. CONCLUDING REMARKS

In this paper, the evolution of design approaches and the shift of design paradigm for embedded control systems are reviewed. It is established that to ensure safety of these systems, it is required to preserve the semantics of controller design in the platform implementation and *vice versa*. Corresponding to this requirement, the design paradigm is gradually moving from isolated design of controllers and platform to a more integrated approach. A group of cosynthesis approaches have emerged which synthesizes parameters on both sides by employing efficient and novel design space exploration and optimization techniques. These approaches can, therefore, effectively bridge the semantic gap between controller and platform designs. We further believe that several future extensions to the cosynthesis paradigm are possible and it will draw increasing attention in the context of CPS design. ■

REFERENCES

[1] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta, "Ensuring safety, security, and sustainability of mission-critical cyber-physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 283–299, Jan. 2011.

[2] *Medical Electrical Equipment—Part 1–11: General Requirements for Basic Safety and Essential Performance*, Standard IEC 60601-1-11, 2010.

[3] *Road Vehicles—Functional Safety—Part I*, Standard ISO 26262-1:2011, 2011.

[4] W. Chang, L. Zhang, D. Roy, and S. Chakraborty, "Control/architecture codesign for cyber-physical systems," in *Handbook of Hardware/Software Codesign*. Amsterdam, The Netherlands: Springer-Verlag, 2017.

[5] B. C. Kuo, *Digital Control Systems*, New York, NY, USA: Holt McDougal, Series in Electrical and Computer Engineering, 1980.

[6] W. C. Schultz and V. C. Rideout, "Control system performance measures: Past, present, and future," *IRE Trans. Autom. Control*, vol. AC-6, no. 1, pp. 22–35, Feb. 1961.

[7] D. G. Roberson and D. J. Stilwell, "L2 gain performance analysis of linear switched systems: Fast switching behavior," in *Proc. Amer. Control Conf.*, Jul. 2007.

[8] K. J. Aström and B. Wittenmark, "Computer-controlled systems: Theory and design," in *Information and System Sciences*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1997.

[9] J. Qiu, H. Gao, and S. X. Ding, "Recent advances on fuzzy-model-based nonlinear networked control systems: A survey," *IEEE Trans. Ind. Electron.*, vol. 63, no. 2, pp. 1207–1217, Feb. 2016.

[10] P. J. Antsaklis, X. D. Koutsoukos, and J. Zaytoon, "On hybrid control of complex systems: A survey," *Eur. J. Autom.*, vol. 32, nos. 9–10, pp. 1023–1045, 1998.

[11] B. De Schutter, W. P. M. H. Heemels, J. Lunze, and C. Prieur, "Survey of modeling, analysis, and control of hybrid systems," in *Handbook of Hybrid Systems Control—Theory, Tools, Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[12] R. Alur, "Formal verification of hybrid systems," in *Proc. Int. Conf. Embedded Softw.*, Oct. 2011, pp. 273–278.

[13] Q. Zhu, H. Liang, L. Zhang, D. Roy, W. Li, and S. Chakraborty, "Extensibility-driven automotive in-vehicle architecture design," in *Proc. Design Autom. Conf.*, 2017.

[14] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. Assoc. Comput. Machinery*, vol. 20, no. 1, pp. 46–61, 1973.

[15] J. Xu and D. L. Parnas, "Scheduling processes with release times, deadlines, precedence and exclusion relations," *IEEE Trans. Softw. Eng.*, vol. 16, no. 3, pp. 360–369, Mar. 1990.

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

[16] R. J. Bril, "*Real-time scheduling for media processing using conditionally guaranteed budgets*," Ph.D. dissertation, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, Netherlands, 2004.

[17] R. J. Bril, J. J. Lukkien, and W. F. Verhaegh, "Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption," *Real-Time Syst.*, vol. 42, nos. 1–3, pp. 63–119, 2009.

[18] R. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Syst.*, vol. 35, no. 3, pp. 239–272, 2007.

[19] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, "Timing analysis of the FlexRay communication protocol," *Real-Time Syst.*, vol. 39, nos. 1–3, pp. 205–235, 2008.

[20] H. Zeng, A. Ghosal, and M. Di Natale, "Timing analysis and optimization of FlexRay dynamic segment," in *Proc. Int. Conf. Comput. Inf. Technol.*, Jun. 2010, pp. 1932–1939.

[21] J. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Germany: Springer-Verlag, ser. Lecture Notes in Computer Science, 2001.

[22] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *Proc. Int. Symp. Circuits Syst.*, 2000, pp. 101–104.

[23] S. Chakraborty, S. Künzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. Design Autom. Test Eur. Conf. Exhib.*, 2003, p. 10190.

[24] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis—The SymTA/S approach," *Inst. Electr. Eng. Proc.—Comput. Digit. Techn.*, vol. 152, no. 2, pp. 148–166, 2005.

[25] U. D. Bordoloi, B. Tanasa, P. Eles, and Z. Peng, "On the timing analysis of the dynamic segment of FlexRay," in *Proc. Int. Symp. Ind. Embedded Syst.*, Jun. 2012, pp. 94–101.

[26] J. Diemer, D. Thiele, and R. Ernst, "Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching," in *Proc. Int. Symp. Ind. Embedded Syst.*, 2012, pp. 1–10.

[27] R. Schneider, L. Zhang, D. Goswami, A. Masrur, and S. Chakraborty, "Compositional analysis of switched Ethernet topologies," in *Proc. Design Autom. Test Eur. Conf. Exhib.*, 2013, pp. 1099–1104.

[28] F. Reimann, S. Graf, F. Streit, M. Glaß, and J. Teich, "Timing analysis of Ethernet AVB-based automotive E/E architectures," in *Proc. 18th Conf. Emerg. Technol. Factory Autom.*, Sep. 2013, pp. 1–8.

[29] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse, "System architecture evaluation using modular performance analysis: A case study," *Int. J. Softw. Tools Technol. Transf.*, vol. 8, no. 6, pp. 649–667, 2006.

[30] A. Hagiescu, U. D. Bordoloi, S. Chakraborty, P. Sampath, P. V. V. Ganesan, and S. Ramesh, "Performance analysis of FlexRay-based ECU networks," in *Proc. Design Autom. Conf.*, 2007, pp. 284–289.

[31] D. B. Chokshi and P. Bhaduri, "Performance analysis of FlexRay-based systems using real-time calculus, revisited," in *Proc. Symp. Appl. Comput.*, 2010, pp. 351–356.

[32] M. Lukasiewycz, M. Glaß, J. Teich, and P. Milbredt, "FlexRay schedule optimization of the static segment," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, 2009.

[33] W. Steiner, "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks," in *Proc. Real-Time Syst. Symp.*, 2010, pp. 375–384.

[34] D. Tamas-Selicean, P. Pop, and W. Steiner, "Synthesis of communication schedules for TTEthernet-based mixed-criticality systems," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, 2012.

[35] Z. Hanzalek, P. Burget, and P. Sucha, "Profinet IO IRT message scheduling with temporal constraints," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 369–380, Aug. 2010.

[36] F. Sagstetter, P. Waszecki, S. Steinhorst, M. Lukasiewycz, and S. Chakraborty, "Multischedule synthesis for variant management in automotive time-triggered systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 4, pp. 637–650, Apr. 2016.

[37] F. Sagstetter, M. Lukasiewycz, and S. Chakraborty, "Generalized asynchronous time-triggered scheduling for FlexRay," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 2, pp. 214–226, Feb. 2017.

[38] S. S. Craciunas and R. S. Oliver, "SMT-based task-and network-level static schedule generation for time-triggered networked systems," in *Proc. Int. Conf. Real-Time Netw. Syst.*, 2014, p. 45.

[39] L. Zhang, D. Goswami, R. Schneider, and S. Chakraborty, "Task-and network-level schedule co-synthesis of Ethernet-based time-triggered systems," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2014, pp. 119–124.

[40] M. Lukasiewycz, F. Sagstetter, and S. Steinhorst, "Efficient design space exploration of embedded platforms," in *Proc. Design Autom. Conf.*, 2015.

[41] J. Teich, "Hardware/software codesign: The past, the present, and predicting the future," *Proc. IEEE*, vol. 100, pp. 1411–1430, 2012.

[42] J. Valencia, D. Goswami, and K. Goossens, "Composable platform-aware embedded control systems on a multi-core architecture," in *Proc. Eur. Conf. Digit. Syst. Design*, 2015.

[43] W. Chang, D. Goswami, S. Chakraborty, J. Xue, L. Ju, and S. Andalam, "Memory-aware embedded control systems design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 4, pp. 586–599, Apr. 2017.

[44] R. Krtolica, Ü. Özgüner, H. Chan, H. Goktas, J. Winkelman, and M. Liubakka, "Stability of linear feedback systems with random communication delays," in *Proc. Amer. Control Conf.*, 1991.

[45] J. Nilsson, B. Bernhardsson, and B. Wittenmark, "Some topics in real-time control," in *Proc. Amer. Control Conf.*, 1998.

[46] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *IEEE Control Syst. Mag.*, vol. 21, no. 1, pp. 84–99, Feb. 2001.

[47] J. Nilsson, B. Bernhardsson, and B. Wittenmark, "Stochastic analysis and control of real-time systems with random time delays," *Automatica*, vol. 34, no. 1, pp. 57–64, 1998.

[48] E. Boje, "Approximate models for continuous-time linear systems with sampling jitter," *Automatica*, vol. 41, no. 12, pp. 2091–2098, 2005.

[49] A. Cervin, "Stability and worst-case performance analysis of sampled-data control systems with input and output jitter," in *Proc. Amer. Control Conf.*, 2012, pp. 3760–3765.

[50] D. Goswami, A. Masrur, R. Schneider, C. J. Xue, and S. Chakraborty, "Multirate controller design for resource- and schedule-constrained automotive ECUs," in *Proc. Design Autom. Test Eur. Conf. Exhib.*, Mar. 2013.

[51] K. Goossens, "Virtual execution platforms for mixed-time-criticality systems: The CompSOC architecture and design flow," *SIGBED Rev.*, vol. 10, no. 3, pp. 23–34, 2013.

[52] E. P. van Horssen, A. R. B. Behrouzian, D. Goswami, D. Antunes, T. Basten, and W. P. M. H. Heemels, "Performance analysis and controller improvement for linear systems with (m, k)-firm data losses," in *Proc. Eur. Control Conf.*, 2016, pp. 2571–2577.

[53] W. Geelen, D. Antunes, J. P. M. Voeten, R. R. H. Schiffelers, and W. P. M. H. Heemels, "The impact of deadline misses on the control performance of high-end motion control systems," *IEEE Trans. Ind. Electron.*, vol. 63, no. 2, pp. 1218–1229, Feb. 2016.

[54] D. Antunes and W. P. M. H. Heemels, "Frequency-domain analysis of control loops with intermittent data losses," *IEEE Trans. Autom. Control*, vol. 61, no. 8, pp. 2295–2300, Aug. 2016.

[55] M. Kauer, S. Steinhorst, D. Goswami, R. Schneider, M. Lukasiewycz, and S. Chakraborty, "Formal verification of distributed controllers using time-stamped event count automata," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2013.

[56] M. Kauer, D. Soudbakhsh, D. Goswami, S. Chakraborty, and A. M. Annaswamy, "Fault-tolerant control synthesis and verification of distributed embedded systems," in *Proc. Design Autom. Test Eur. Conf. Exhib.*, 2014, pp. 1–6.

[57] D. Goswami, R. Schneider, and S. Chakraborty, "Relaxing signal delay constraints in distributed embedded controllers," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 6, pp. 2337–2345, Nov. 2014.

[58] I. Saha, S. Baruah, and R. Majumdar, "Dynamic scheduling for networked control systems," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2015, pp. 98–107.

[59] R. Majumdar, I. Saha, and M. Zamani, "Performance-aware scheduler synthesis for control systems," in *Proc. Int. Conf. Embedded Softw.*, 2011, pp. 299–308.

[60] V. C. Aitken and H. M. Schwartz, "On the exponential stability of discrete-time systems with applications in observer design," *IEEE Trans. Autom. Control*, vol. 39, no. 9, pp. 1959–1962, Sep. 1994.

[61] A. Podelski and S. Wagner, "Model checking of hybrid systems: From reachability towards stability," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2006.

[62] A. Podelski and S. Wagner, "Region stability proofs for hybrid systems," in *Proc. Int. Conf. Formal Modelling Anal. Timed Syst.*, 2007.

[63] A. Anta, R. Majumdar, I. Saha, and P. Tabuada, "Automatic verification of control system implementations," in *Proc. Int. Conf. Embedded Softw.*, 2010.

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

[64] E. Feron, "From control systems to control software," *IEEE Control Syst. Mag.*, vol. 30, no. 6, pp. 50–71, Jun. 2010.

[65] R. Majumdar, I. Saha, and M. Zamani, "Synthesis of minimal-error control software," in *Proc. Int. Conf. Embedded Softw.*, 2012, pp. 123–132.

[66] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, 1995.

[67] G. M. Mancuso, E. Bini, and G. Pannocchia, "Optimal priority assignment to control tasks," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 5s, pp. 161:1–161:17, 2014.

[68] A. Aminifar, P. Eles, Z. Peng, and A. Cervin, "Stability-aware analysis and design of embedded control systems," in *Proc. Int. Conf. Embedded Softw.*, 2013, Art. no. 23.

[69] A. Cervin, B. Lincoln, J. Eker, K. E. Årzén, and G. Buttazo, "The jitter margin and its application in the design of real-time control systems," in *Proc. Int. Conf. Real-Time Embedded Comput. Syst. Appl.*, 2004.

[70] A. Aminifar, E. Bini, P. Eles, and Z. Peng, "Designing bandwidth-efficient stabilizing control servers," in *Proc. Real-Time Syst. Symp.*, Dec. 2013, pp. 298–307.

[71] A. Aminifar, E. Bini, P. Eles, and Z. Peng, "Analysis and design of real-time servers for control applications," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 834–846, Mar. 2016.

[72] M. Al Khatib, A. Girard, and T. Dang, "Verification and synthesis of timing contracts for embedded controllers," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2016, pp. 115–124.

[73] A. R. B. Behrouzian, "Sample-drop firmness analysis of TDMA-scheduled control applications," in *Proc. Symp. Ind. Embedded Syst.*, May 2016.

[74] Y. Wu, G. Buttazzo, E. Bini, and A. Cervin, "Parameter selection for real-time controllers in resource-constrained systems," *IEEE Trans. Ind. Informat.*, vol. 6, no. 4, pp. 610–620, Apr. 2010.

[75] R. Schneider, D. Goswami, A. Masrur, and S. Chakraborty, "QoC-oriented efficient schedule synthesis for mixed-criticality cyber-physical systems," in *Proc. Forum Specification Design Lang.*, 2012, pp. 60–67.

[76] R. Schneider, D. Goswami, A. Masrur, M. Becker, and S. Chakraborty, "Multi-layered scheduling of mixed-criticality cyber-physical systems," *J. Syst. Archit.*, vol. 59, no. 10, pp. 1215–1230, 2013.

[77] P. Martí, J. M. Fuertes, G. Fohler, and K. Ramamritham, "Improving quality-of-control using flexible timing constraints: Metric and scheduling," in *Proc. Real-Time Syst. Symp.*, Dec. 2002, pp. 91–100.

[78] D. Roy, M. Balszun, D. Goswami, and S. Chakraborty, "Hybrid automotive in-vehicle networks," in *Proc. Int. Symp. Netw. Chip*, 2017.

[79] D. Goswami, R. Schneider, and S. Chakraborty, "Re-engineering cyber-physical control applications for hybrid communication protocols," in *Proc. Design Autom. Test Eur. Conf. Exhib.*, 2011.

[80] A. Masrur, D. Goswami, R. Schneider, H. Voit, A. Annaswamy, and S. Chakraborty, "Schedulability analysis of distributed cyber-physical applications on mixed time-/event-triggered bus architectures with retransmissions," in *Proc. Int. Symp. Ind. Embedded Syst.*, Jun. 2011, pp. 266–273.

[81] A. Masrur, D. Goswami, S. Chakraborty, J. Chen, A. Annaswamy, and A. Banerjee, "Timing analysis of cyber-physical applications for hybrid communication protocols," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2012.

[82] L. Zhang, D. Roy, P. Mundhenk, and S. Chakraborty, "Schedule management framework for cloud-based future automotive software systems," in *Proc. Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2016, pp. 12–21.

[83] P. Mundhenk, G. Tibba, L. Zhang, F. Reimann, D. Roy, and S. Chakraborty, "Dynamic platforms for uncertainty management in future automotive E/E architectures: Invited," in *Proc. Design Autom. Conf.*, 2017, Art. no. 15.

[84] D. Majumdar, L. Zhang, P. Bhaduri, and S. Chakraborty, "Reconfigurable communication middleware for flex ray-based distributed embedded systems," in *Proc. Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2015, pp. 159–166.

[85] M. Balszun, D. Roy, L. Zhang, W. Chang, and S. Chakraborty, "Effectively utilizing elastic resources in networked control systems," in *Proc. Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, 2017.

[86] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Trans. Autom. Control*, vol. 52, no. 9, pp. 1680–1685, Sep. 2007.

[87] A. Anta and P. Tabuada, "Self-triggered stabilization of homogeneous control systems," in *Proc. Amer. Control Conf.*, Jun. 2008, pp. 4129–4134.

[88] Z.-P. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems," *Automatica*, vol. 37, no. 6, pp. 857–869, Jun. 2001.

[89] M. Velasco, P. Martí, and E. Bini, "On Lyapunov sampling for event-driven controllers," in *Proc. Conf. Decision Control*, 2009.

[90] R. Postoyan, P. Tabuada, D. Nešic, and A. Anta, "Event-triggered and self-triggered stabilization of distributed networked control systems," in *Proc. Conf. Decision Control Eur. Control Conf.*, 2011, pp. 2565–2570.

[91] M. Abdelrahim, V. S. Dolk, and W. P. M. H. Heemels, "Input-to-state stabilizing event-triggered control for linear systems with output quantization," in *Proc. Conf. Decision Control*, Dec. 2016.

[92] P. Martí, M. Velasco, and E. Bini, "The optimal boundary and regulator design problem for event-driven controllers," in *Proc. 12th Int. Conf. Hybrid Syst. Comput. Control*, 2009, pp. 441–444.

[93] M. Velasco, P. Martí, J. Yépez, F. J. Ruiz, J. M. Fuertes, and E. Bini, "Qualitative analysis of a one-step finite-horizon boundary for event-driven controllers," in *Proc. Conf. Decision Control Eur. Control Conf.*, Dec. 2011, pp. 1662–1667.

[94] M. Velasco, P. Martí, and E. Bini, "Control-driven tasks: Modeling and analysis," in *Proc. Real-Time Syst. Symp.*, 2008.

[95] A. Aminifar, P. Tabuada, P. Eles, and Z. Peng, "Self-triggered controllers and hard real-time guarantees," in *Proc. Design Autom. Test Eur. Conf. Exhib.*, 2016, pp 636–641.

[96] M. Mazo and P. Tabuada, "On event-triggered and self-triggered control over sensor/actuator networks," in *Proc. Conf. Decision Control*, 2008, pp. 435–440.

[97] S. Samii, P. Eles, Z. Peng, P. Tabuada, and A. Cervin, "Dynamic scheduling and control-quality optimization of self-triggered control applications," in *Proc. Real-Time Syst. Symp.*, 2010, pp. 95–104.

[98] I. Saha and R. Majumdar, "Trigger memoization in self-triggered control," in *Proc. Int. Conf. Embedded Softw. (EMSOFT)*, 2012.

[99] J. Araújo, M. Mazo, A. Anta, P. Tabuada, and K. H. Johansson, "System architectures, protocols and algorithms for aperiodic wireless control systems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 175–184, 2014.

[100] A. Aminifar, S. Samii, P. Eles, Z. Peng, and A. Cervin, "Designing high-quality embedded control systems with guaranteed stability," in *Proc. Real-Time Syst. Symp.*, Dec. 2012, pp. 283–292.

[101] A. Aminifar, E. Bini, P. Eles, and Z. Peng, "Bandwidth-efficient controller-server co-design with stability guarantees," in *Proc. Design Autom. Test Eur. Conf. Exhib.*, Mar. 2014, pp. 1–6.

[102] J. Valencia, E. P. van Horssen, D. Goswami, W. P. M. H. Heemels, and K. G. W. Goossens, "Resource utilization and quality-of-control trade-off for a composable platform," in *Proc. Design Autom. Test Eur. Conf. Exhib.*, 2016.

[103] Y. Xu, K.-E. Årzén, E. Bini, and A. Cervin, "Response time driven design of control systems," *IFAC World Congr.*, vol. 47, no. 3, pp. 6098–6104, 2014.

[104] Y. Xu, K.-E. Årzén, A. Cervin, E. Bini, and B. Tanasa, "Exploiting job response-time information in the co-design of real-time control systems," in *Proc. Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2015.

[105] S. Samii, A. Cervin, P. Eles, and Z. Peng, "Integrated scheduling and synthesis of control applications on distributed embedded systems," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2009, pp. 57–62.

[106] B. Lincoln and A. Cervin, "JITTERBUG: A tool for analysis of real-time control performance," in *Proc. Conf. Decision Control*, Dec. 2002.

[107] S. Samii, P. Eles, Z. Peng, and A. Cervin, "Design optimization and synthesis of FlexRay parameters for embedded control applications," in *Proc. Int. Symp. Electron. Design Test Appl.*, 2011, pp. 66–71.

[108] A. Aminifar, P. Eles, Z. Peng, and A. Cervin, "Control-quality driven design of cyber-physical systems with robustness guarantees," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2013, pp. 1093–1098.

[109] R. Schneider, D. Goswami, S. Zafar, M. Lukasiewycz, and S. Chakraborty, "Constraint-driven synthesis and tool-support for FlexRay-Based automotive control systems," in *Proc. 7th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, 2011, pp. 139–148.

[110] D. Goswami, M. Lukasiewycz, R. Schneider, and S. Chakraborty, "Time-triggered implementations of mixed-criticality automotive software," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, Mar. 2012.

[111] D. Roy, L. Zhang, W. Chang, D. Goswami, and S. Chakraborty, "Multi-objective co-optimization of FlexRay-based distributed control systems," in *Proc. Real-Time Embedded Technol. Appl. Symp.*, 2016.

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

[112] T. Gommans, D. Antunes, T. Donkers, P. Tabuada, and M. Heemels, "Self-triggered linear quadratic control," *Automatica*, vol. 50, no. 4, pp. 1279–1287, 2014.

[113] A. Aminifar, S. Samii, P. Eles, and Z. Peng, "Control-quality driven task mapping for distributed embedded control systems," in *Proc. Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2011, pp. 133–142.

[114] D. Goswami, R. Schneider, and S. Chakraborty, "Co-design of cyber-physical systems via controllers with flexible delay constraints," in *Proc. Asia South Pacific Design Autom. Conf.*, 2011, pp. 225–230.

[115] K. Pettis and R. C. Hansen, "Profile guided code positioning," in *Proc. Conf. Programm. Lang. Design Implement.*, vol. 25, no. 6, pp. 16–27, 1990.

[116] J. Kalamationos and D. R. Kaeli, "Temporal-based procedure reordering for improved instruction cache performance," in *Proc. Int. Symp. High-Perform. Comput. Archit.*, Feb. 1998, pp. 244–253.

[117] N. Gloy, T. Blackwell, M. D. Smith, and B. Calder, "Procedure placement using temporal ordering information," in *Proc. Int. Symp. Microarchitect.*, 1997.

[118] K. W. Batcher and R. A. Walker, "Dynamic round-robin task scheduling to reduce cache misses for embedded systems," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2008, pp. 260–263.

[119] M. Glaß, M. Lukasiewyc, J. Teich, U. D. Bordoloi, and S. Chakraborty, "Designing heterogeneous ECU networks via compact architecture encoding and hybrid timing analysis," in *Proc. Design Autom. Conf.*, Jul. 2009, pp. 43–46.

[120] D. Bui, E. Lee, I. Liu, H. Patel, and J. Reineke, "Temporal isolation on multiprocessing architectures," in *Proc. Design Autom. Conf.*, 2011, pp. 274–279.

[121] T. Ungerer, "Merasa: Multicore execution of hard real-time applications supporting analyzability," *IEEE Micro*, vol. 30, no. 5, pp. 66–75, Sep. 2010.

[122] S. Girbal, X. Jean, J. L. Rhun, D. G. Pérez, and M. Gatti, "Deterministic platform software for hard real-time systems using multi-core COTS," in *Proc. Digit. Avion. Syst. Conf.*, Sep. 2015, pp. 8D4-1–8D4-15.

[123] R. Tabish, "A real-time scratchpad-centric OS for multi-core embedded systems," in *Proc. Real-Time Embedded Technol. Appl. Symp.*, Apr. 2016, pp. 1–11.

[124] A. Bemporad, A. Oliveri, T. Poggi, and M. Storace, "Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations," *IEEE Trans. Autom. Control*, vol. 56, no. 12, pp. 2883–2897, Dec. 2011.

[125] Z. Yao and N. H. El-Farra, "Resource-aware model predictive control of spatially distributed processes using event-triggered communication," in *Proc. Conf. Decision Control*, Dec. 2013, pp. 3726–3731.

[126] T. E. Gibson, A. M. Annaswamy, and E. Lavretsky, "Improved transient response in adaptive control using projection algorithms and closed loop reference models," in *Proc. AIAA Guid. Navigat. Control Conf.*, 2012.

[127] T. E. Gibson, A. M. Annaswamy, and E. Lavretsky, "Closed–loop reference model adaptive control: Composite control and observer feedback," in *Proc. IFAC Int. Workshop Adaptation Learn. Control Signal Process.*, Jun. 2013, pp. 3376–3383.

[128] T. E. Gibson, A. M. Annaswamy, and E. Lavretsky, "Adaptive systems with closed-loop reference-models, part I: Transient performance," in *Proc. Amer. Control Conf.*, 2013.

[129] T. E. Gibson, A. M. Annaswamy, and E. Lavretsky, "Closed-loop reference models for output-feedback adaptive systems," in *Proc. Eur. Control Conf.*, Jul. 2013, pp. 365–370.

[130] L. Chunmao and X. Jian, "Adaptive delay estimation and control of networked control systems," in *Proc. Int. Symp. Commun. Inf. Technol.*, 2006, pp. 707–710.

[131] H. Voit and A. Annaswamy, "Adaptive control of a networked control system with hierarchical scheduling," in *Proc. Amer. Control Conf.*, Jun. 2011, pp. 4189–4194.

[132] H. Voit, A. M. Annaswamy, R. Schneider, D. Goswami, and S. Chakraborty, "Adaptive switching controllers for systems with hybrid communication protocols," in *Proc. Amer. Control Conf.*, 2012, pp. 4921–4926.

[133] Y.-E. Wang, X.-M. Sun, P. Shi, and J. Zhao, "Input-to-state stability of switched nonlinear systems with time delays under asynchronous switching," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2261–2265, Dec. 2013.

[134] Y. Wang, X. Sun, and B. Wu, "Lyapunov–Krasovskii functionals for input-to-state stability of switched non-linear systems with time-varying input delay," *IET Control Theory Appl.*, vol. 9, no. 11, pp. 1717–1722, Jul. 2015.

[135] W. P. M. H. Heemels, "Stability analysis of nonlinear networked control systems with asynchronous communication: A small-gain approach," in *Proc. Conf. Decision Control*, Dec. 2013, pp. 4631–4637.

[136] Y. Wang, M. Xia, V. Gupta, and P. J. Antsaklis, "On feedback passivity of discrete-time nonlinear networked control systems with packet drops," *IEEE Trans. Autom. Control*, vol. 60, no. 9, pp. 2434–2439, Sep. 2015.

[137] J. Lei and H. K. Khalil, "Feedback linearization for nonlinear systems with time-varying input and output delays by using high-gain predictors," *IEEE Trans. Autom. Control*, vol. 61, no. 8, pp. 2262–2268, Aug. 2016.

[138] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. SMC-15, no. 1, pp. 116–132, Jan. 1985.

[139] X. Zhang, G. Lu, and Y. Zheng, "Stabilization of networked stochastic time-delay fuzzy systems with data dropout," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 3, pp. 798–807, Mar. 2008.

[140] H. Gao, Y. Zhao, and T. Chen, "H_∞ fuzzy control of nonlinear systems under unreliable communication links," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 2, pp. 265–278, Feb. 2009.

[141] H. Li, C. Wu, and Z. Feng, "Fuzzy dynamic output-feedback control of non-linear networked discrete-time system with missing measurements," *IET Control Theory Appl.*, vol. 9, no. 3, pp. 327–335, 2015.

[142] J. Qiu, G. Feng, and H. Gao, "Fuzzy-model-based piecewise H_∞ static-output-feedback controller design for networked nonlinear systems," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 5, pp. 919–934, 2010.

[143] D. Du, "Reliable H_∞ control for Takagi-Sugeno fuzzy systems with intermittent measurements," *Nonlinear Anal. Hybrid Syst.*, vol. 6, no. 4, pp. 930–941, 2012.

[144] Y. Zhao, H. Gao, and T. Chen, "Fuzzy constrained predictive control of nonlinear systems with packet dropouts," *IET Control Theory Appl.*, vol. 4, no. 9, p. 1665–1677, 2010.

[145] H. Zhang, J. Yang, and C. Su, "T-S fuzzy-model-based robust H_∞ design for networked control systems with uncertainties," *IEEE Trans. Ind. Informat.*, vol. 3, no. 4, pp. 289–301, Apr. 2007.

[146] H. Zhang, D. Yang, and T. Chai, "Guaranteed cost networked control for T–S fuzzy systems with time delays," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 37, no. 2, pp. 160–172, Mar. 2007.

[147] C. Peng and T. C. Yang, "Communication-delay-distribution-dependent networked control for a class of T-S fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 2, pp. 326–335, Feb. 2010.

[148] E. Tian, D. Yue, and Z. Gu, "Robust H_∞ control for nonlinear systems over network: A piecewise analysis method," *Fuzzy Sets Syst.*, vol. 161, no. 21, pp. 2731–2745, 2010.

[149] M. S. Mahmoud and A.-W. A. Saif, "Robust quantized approach to fuzzy networked control systems," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 1, p. 71–81, Mar. 2012.

[150] J. Yan, Y. Xia, and L. Li, "Stabilization of fuzzy systems with quantization and packet dropout," *Int. J. Robust Nonlinear Control*, vol. 24, no. 10, p. 1563–1583, 2014.

[151] S. A. Seshia, "Combining induction, deduction, and structure for verification and synthesis," *Proc. IEEE*, vol. 103, no. 11, pp. 2036–2051, Nov. 2015.

[152] W. Kohn, J. James, A. Nerode, K. Harbison, and A. Agrawala, "A hybrid systems approach to computer-aided control engineering," *IEEE Control Syst.*, vol. 15, no. 2, pp. 14–25, Apr. 1995.

[153] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Trans. Autom. Control*, vol. 43, no. 1, pp. 31–45, Jan. 1998.

[154] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Trans. Autom. Control*, vol. 51, no. 12, pp. 1862–1877, Dec. 2006.

[155] X. Chen, E. Abraham, and S. Sankaranarayan, "Taylor model flowpipe construction for non-linear hybrid systems," in *Proc. Real-Time Syst. Symp.*, 2012.

[156] R. Testylier and T. Dang, "NLTOOLBOX: A C++ library for reachability computation of non-linear dynamical systems," in *Proc. Int. Symp. Autom. Technol. Verification Anal.*, 2013, pp. 469–473.

[157] T. Dang, O. Maler, and R. Testylier, "Accurate hybridization of nonlinear systems," in *Proc. Int. Conf. Hybrid Syst., Comput. Control*, 2010, pp. 11–20.

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

[158] E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Inf.*, vol. 43, no. 7, pp. 451–476, 2007.

[159] X. Chen and S. Sankaranarayanan, "Decomposed reachability analysis for nonlinear systems," in *Proc. Real-Time Syst. Symp.*, 2016, pp. 13–24.

[160] H. Ravanbakhsh and S. Sankaranarayanan, "Robust controller synthesis of switched systems using counterexample guided framework," in *Proc. Int. Conf. Embedded Softw.*, Oct. 2016, pp. 1–10.

[161] H. Ravanbakhsh and S. Sankaranarayanan, "Infinite horizon safety controller synthesis through disjunctive polyhedral abstract interpretation," in *Proc. Int. Conf. Embedded Softw.*, Oct. 2014, pp. 1–10.

[162] S. Ghosh, "Diagnosis and repair for synthesis from signal temporal logic specifications," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2016, pp. 31–40.

[163] M. Zamani, M. Mazo, and A. Abate, "Finite abstractions of networked control systems," in *Proc. Conf. Decision Control*, Dec. 2014, pp. 95–100.

[164] M. Khaled, M. Rungger, and M. Zamani, "Symbolic models of networked control systems: A feedback refinement relation approach," in *Proc. Conf. Commun. Control Comput.*, 2016.

[165] M. Zamani, G. Pola, M. Mazo, Jr., and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1804–1809, Jul. 2012.

[166] A. Balluchi, L. Benveunuti, T. Villa, H. Wong-Toi, and A. L. Sangiovanni-Vincentelli, "Controller synthesis for hybrid systems with lower bounds on event separation," in *Proc. Conf. Decision Control*, Dec. 1999, pp. 3984–3989.

[167] M. Rungger and M. Zamani, "SCOTS: A tool for the synthesis of symbolic controllers," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2016, pp. 99–104.

[168] L. Di Guglielmo, S. A. Seshia, and T. Villa, "Synthesis of implementable control strategies for lazy linear hybrid automata," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, 2013.

[169] A. Girard, "Controller synthesis for safety and reachability via approximate bisimulation," *Automatica*, vol. 48, no. 5, pp. 947–953, 2012.

[170] P. Bouyer, K. G. Larsen, N. Markey, O. Sankur, and C. Thrane, "Timed automata can always be made implementable," in *Proc. Int. Conf. Concurrency Theory*, 2011, pp. 76–91.

[171] M. Rungger, G. Reissig, and M. Zamani, "Symbolic synthesis with average performance guarantees," in *Proc. Conf. Decision Control*, Dec. 2016, pp. 7404–7410.

[172] G. Reissig, A. Weber, and M. Rungger, "Feedback refinement relations for the synthesis of symbolic controllers," *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 1781–1796, Apr. 2017.

[173] S. Checkoway, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. Conf. Secur.*, 2011.

[174] K. Koscher, "Experimental security analysis of a modern automobile," in *Proc. Symp. Secur. Privacy*, 2010.

[175] B. Zheng, P. Deng, R. Anguluri, Q. Zhu, and F. Pasqualetti, "Cross-layer codesign for secure cyber-physical systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 5, pp. 699–711, May 2016.

[176] D. Rakhmatov and S. Vrudhula, "An analytical high-level battery model for use in energy management of portable electronic systems," in *Proc. Int. Conf. Comput. Aided Design*, 2001.

[177] W. Chang, A. Pröbstl, D. Goswami, M. Zamani, and S. Chakraborty, "Battery-and aging-aware embedded control systems for electric vehicles," in *Proc. Real-Time Syst. Symp.*, 2014.

[178] K. Vatanparvar and M. A. Al Faruque, "Battery lifetime-aware automotive climate control for electric vehicles," in *Proc. Design Autom. Conf.*, 2015.

[179] M. Pedram, N. Chang, Y. Kim, and Y. Wang, "Hybrid electrical energy storage systems," in *Proc. Int. Symp. Low Power Electron. Design*, 2010.

[180] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini, "Fault-tolerant platforms for automotive safety-critical applications," in *Proc. Int. Conf. Compil. Archit. Synth. Embedded Syst.*, 2003.

[181] J. Kim, G. Bhatia, R. Rajkumar, and M. Jochim, "SAFER: System-level architecture for failure evasion in real-time applications," in *Proc. Real-Time Syst. Symp.*, 2012.

[182] D. Theilliol, C. Join, and Y. Zhang, "Actuator fault-tolerant control design based on reconfigurable reference input," *Int. J. Appl. Math. Comput. Sci.*, vol. 18, no. 4, pp. 553–560, 2008.

[183] D. Goswami, D. Müller-Gritschneder, T. Basten, U. Schlichtmann, and S. Chakraborty, "Fault-tolerant embedded control systems for unreliable hardware," in *Proc. Int. Symp. Integr. Circuits*, Dec. 2014, pp. 464–467.

[184] W. Chang, D. Roy, L. Zhang, and S. Chakraborty, "Model-based design of resource-efficient automotive control software," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2016, pp. 1–8.

[185] D. Roy, W. Chang, L. Zhang, and S. Chakraborty, "Automated synthesis of cyber-physical systems from joint controller/architecture specifications," in *Proc. Forum Specification Design Lang.*, 2016.

[186] C. Baier and J. Katoen, *Principles of Model Checking* (Representation and Mind Series). Cambridge, MA, USA: MIT Press, 2008.

[187] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," *Adv. Comput.*, vol. 58, pp. 117–148, 2003.

[188] M. Fitting, "First-order logic and automated theorem proving," in *Graduate Texts in Computer Science*. New York, NY, USA: Springer-Verlag, 1996.

[189] S. Sims and D. C. DuVarney, "Experience report: The Reactis validation tool," in *Proc. Int. Conf. Funct. Programm.*, 2007.

[190] M. Satpathy, A. Yeolekar, and S. Ramesh, "Randomized directed testing (REDIRECT) for simulink/stateflow models," in *Proc. Int. Conf. Embedded Softw.*, 2008.

[191] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli, "Metropolis: An integrated electronic system design environment," *Computer*, vol. 36, no. 4, pp. 45–52, Apr. 2003.

[192] A. Davare, "metroII: A design environment for cyber-physical systems," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 1s, pp. 49:1–49:31, 2013.

ABOUT THE AUTHORS

Debayan Roy (Student Member, IEEE) received the M.Sc. degree in communications engineering from the Technical University of Munich (TU Munich), Munich, Germany, in 2015, where he is currently working toward the Ph.D. degree.

He is currently a Researcher with the Chair of Real-Time Computer Systems, TU Munich. His current research interests include automotive E/E architecture and embedded control systems.

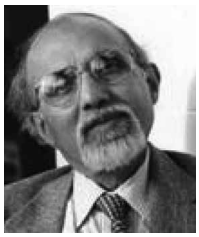
Licong Zhang (Student Member, IEEE) received the Dipl. -Ing degree in electrical and computer engineering from the Technical University of Munich (TU Munich), Munich, Germany, in 2011, where he is currently working toward the Ph.D. degree with the Chair of Real-Time Computer Systems.

His current research interests include automotive E/E architecture and software, in-vehicle communication networks and control-platform cosynthesis.

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.

Wanli Chang (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the Technical University of Munich (TU Munich), Munich, Germany, in 2017.

He is a Lecturer at the Singapore Institute of Technology, Singapore. His current research interest includes resource-aware automotive control systems.



Sanjoy K. Mitter (Fellow, IEEE) received the Ph.D. degree from the Imperial College of Science and Technology, London, U.K., in 1965.

He is currently a Professor in Electrical Engineering at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. His current research interests are communication and control in a networked environment, the relationship of statistical and quantum physics to information theory and control, and autonomy and adaptiveness for integrative organization.

Samarjit Chakraborty (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from ETH Zurich, Zurich, Switzerland, in 2003.

He is currently a Professor in Electrical and Computer Engineering at the Technical University of Munich (TU Munich), Munich, Germany, where he holds the Chair for Real-Time Computer Systems. His research interests include distributed embedded systems, embedded control systems, energy storage systems, electromobility, and sensor network-based information processing for healthcare.

Authorized licensed use limited to: University of York. Downloaded on August 24,2020 at 11:04:44 UTC from IEEE Xplore. Restrictions apply.