INVITED PAPER

Codesign Methodologies and Tools for Cyber–Physical Systems

In this paper, the authors propose to codesign cyber and physical components of CPSs in a holistic environment. They present a number of codesign approaches in modeling, simulation, synthesis, verification, and validation. They also discuss open challenges in CPS codesign and possible future directions for addressing them.

By Qi Zhu, *Member IEEE*, and Alberto Sangiovanni-Vincentelli, *Fellow IEEE*

ABSTRACT | Cyber–physical system (CPS) analysis and design are challenging due to the intrinsic heterogeneity of those systems. Today, CPSs are often designed by leveraging existing solutions and by adding cyber components to an existing physical system, thus decomposing the design into two separate phases. In this paper, we argue that the codesign of the cyber and physical components would expose solutions that are better under all aspects, such as safety, efficiency, security, performance, reliability, fault tolerance, and extensibility. To do so, automated codesign tools are a necessity due to the complexity of the problems at hand. In the paper, we will discuss the key needs and challenges in developing modeling, simulation, synthesis, validation, and verification tools for CPS codesign, present promising codesign approaches from our teams and others, and point out where additional research is needed.

KEYWORDS | Codesign; cyber–physical systems (CPSs); design automation; modeling; synthesis; verification

I. INTRODUCTION

Cyber–physical systems (CPSs) such as autonomous vehicles, industrial robots, medical devices, smart buildings, and smart infrastructures promise substantial economic and societal benefits. The design of these systems, however, faces serious challenges from the fast increase of system scale and complexity, the close interaction with dynamic physical processes, the adoption of advanced and distributed embedded platforms, and the stringent requirements on a variety of design metrics such as performance, safety, security, fault tolerance, extensibility, energy consumption, and cost.

A key principal in tackling these challenges is to codesign various cyber and physical components of the system, i.e., to model, simulate, synthesize, and validate the sensing, control, computation, and communication algorithms, the software and hardware implementation platform, the mechanical components and processes, and the surrounding physical environment and human activities in a holistic environment. In fact, the term "cyber–physical systems" itself shows a unified view of heterogeneous cyber and physical components, and emphasizes the importance of analyzing their interactions to achieve better system designs.

In current practice, however, CPSs are still designed in isolated stages that address individual elements. For instance, control design is carried out with modeling of the physical processes but usually without consideration of the cyber platform. As control performance and stability significantly depend on the reliability and timing behavior of the underlying computation and communication, such isolation of control design and cyber platform implementation could lead to long production cycles, inferior systems or even infeasible designs.

To facilitate codesign of CPSs, new methodologies and tools are greatly needed. These tools should be able to capture the key interactions among various cyber

Manuscript received July 4, 2018; accepted July 23, 2018. Date of current version September 14, 2018. This work was supported in part by the National Science Foundation Directorate for Computer and Information Science and Engineering under Grant 1553757, Grant 1645964, Grant 1646381, Grant 1646641, Grant 1724341, and Grant 1739816, in part by the Office of Naval Research under Grant N00014-14-1-0815 and Grant N00014-14-1-0816; and in part by the Camozzi Group and the Toyota InfoTechnology Center. *(Corresponding author: Qi Zhu.)*

Q. Zhu is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA (e-mail: qzhu@northwestern.edu).

A. Sangiovanni-Vincentelli is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: alberto@berkeley.edu).

Digital Object Identifier 10.1109/JPROC.2018.2864271

and physical components, and to explore the entire design space in a holistic, quantitative, and automated fashion.

In the rest of the paper, we will discuss the needs and challenges for CPS codesign, and present promising codesign methodologies and tools. We organize the paper based on different design automation technology categories: modeling and simulation (Section II), synthesis (Section III), and verification and validation (Section IV). As we will see, tools in each of these categories are essential to achieve successful codesign of CPSs and ultimately effective and reliable systems.

II. MODELING AND SIMULATION

Modeling and simulation tools set the foundation for codesigning CPSs. They enable designers to capture the behavior of various cyber and physical components, specify design requirements and objectives, evaluate and validate design options based on simulations, and identify design factors that are critical for synthesis and verification.

To facilitate codesign in CPSs, it is particularly important that these tools provide the following capabilities.

- Modeling heterogeneous components in a common environment: Compared to traditional embedded systems, CPSs often involve components that have heterogeneous behavior and could be captured with different models of computation (MoCs). For instance, there are intrinsic differences between discrete cyber components and continuous physical processes. The first step for codesign is to enable modeling heterogeneity in a common environment. The component models do not have to be captured in the same language, but their interfaces should be well defined with clear syntax and semantics.
- Cosimulating heterogeneous components: Simulation is an important capability for reasoning about CPS behavior. It requires cosimulation of components that have different semantics, abstraction levels, and time scales. To enable such cosimulation, clear execution semantics have to be defined at the component interfaces and at the entire system level. Monitors and rollback mechanisms may be introduced to facilitate the simulation.
- Separating design concerns: To effectively and efficiently codesign multiple design components (e.g., a control algorithm and its embedded implementation), clearly separating their models and formalizing the interfaces is important. This separation in modeling provides the benefit of 1) exploring design alternatives of one component while reusing other components' designs; and of 2) identifying the key factors that affect the interactions among components and leveraging these factors in codesign.

In this section, we present several design environments that provide these capabilities, with particular attention to the Berkeley tools.

A. Metropolis and Metro II

The principles of heterogeneous modeling, cosimulation, and separation of concerns had been well established in the paradigm of platform-based design [1], [2], and realized in the design environment Metropolis [3], [4]. Specifically, Metropolis supports various common MoCs (e.g., dataflow, state machine, discrete event, and discrete time) with a unified language, the Metropolis meta-model (MMM). Heterogeneous components can be captured with MMM and cosimulated through a SystemC based engine [5], [6]. Furthermore, a key principle in Metropolis is to separate the modeling of system functionality (i.e., what the system does) from the modeling of architectural platform (i.e., how the system is implemented). The separated functional model and architectural model, both described in MMM along with design constraints, are then brought together through a mapping process, during which different platform implementation of the functionality are explored. This separation also enables function-architecture codesign, where different functional designs or platform choices can be easily explored without changing the other part's model (while considering the other part through mapping).

In Metro II [4], [7], [8], the second generation of Metropolis, these principles facilitating codesign are further strengthened. First, instead of requiring all components to be captured in the same MMM language, Metro II provides wrappers to support integrating and cosimulating components that are described in different languages. This is particularly important for CPSs, as components from different domains often come with their own modeling languages and simulation tools. Second, in addition to the separation of functionality and architecture, Metro II further separates the modeling of logical quantities (e.g., ordering of events) from the modeling of physical quantities (e.g., physical time and energy consumption), through the concepts of schedulers and annotators. In Metropolis, all quantities are modeled through quantity managers. The clearer separation of logical and physical aspects in Metro II helps analyzing the complex interactions among components and identifying the critical factors for codesign.

There are a number of case studies that have demonstrated the effectiveness of using Metropolis and Metro II in system codesign, from more traditional embedded systems in multimedia [9], telecommunication [10] and automotive [11] domains, to CPSs in buildings [12] and aircraft [13]. For instance, in building design automation [4], [14], a controller model described in Simulink [15] and a building plant model described in Modelica [16] are integrated into Metro II, and cosimulated with an architectural model for performance analysis and communication network synthesis.

B. Ptolemy

The Ptolemy II framework [17] is another modeling and simulation environment that facilities CPS codesign with support for heterogeneity and separation of concerns. The framework supports a variety of MoCs, such as process networks, discrete event, dataflow, synchronous reactive and continuous time, through the concept and implementation of directors. Heterogeneous components that are described with different MoCs can be integrated and cosimulated in a hierarchical model with multiple directors. For instance, a controller component governed by a synchronous dataflow (SDF) director can be integrated with a plant model governed by a continuous-time (CT) director, and cosimulated at the system level under a discrete event (DE) director. The framework also develops domain-specific ontologies for identifying misconnected components (e.g., because of unit, semantic or transposition errors) to ensure correct integration of heterogeneous components [18].

In Ptolemy II, different aspects of a system are orthogonalized based on aspect-oriented programming [19], [20], which share similar ideas to quantity managers in Metropolis. Furthermore, while the framework has mostly focused on functional modeling, it provides the capability for integrating and cosimulating architectural models. In particular, the PTIDES [21], [22] programming model enables clear separation of logical time in functional model and physical time in architectural platform, and ensures timing consistency when functionality and architecture are mapped together (a lightweight microkernel PtidyOS is presented in [23] to facilitate such mapping). The architectural models can be built within Ptolemy II as in [20], or integrated from other tools as shown below in the Metronomy project.

C. Metronomy

Leveraging the strength of Metro II and Ptolemy, the integrated Metronomy framework provides heterogeneous modeling and cosimulation capabilities to facilitate CPS codesign, in particular for verifying system timing behavior during design space exploration [13].

In Metronomy, the functional model is captured in Ptolemy, while the architectural model is described in Metro II, as shown in Fig. 1. At the system level, the two models are integrated and cosimulated through a CoSimDirector, which implements the Metro II execution semantics as a Ptolemy director. The functional model may further contain heterogeneous components—as shown in the figure, a discrete controller model and a continuous physical plant model are cosimulated with a customized discrete event director CoSimDEDirector. On the other hand, the Metro II architectural model can be simulated with a SystemC simulation engine. During model integration, it is compiled with SystemC and Metro II libraries into an executable, which is then cosimulated with the functional model via interprocess communication (governed by the CoSimDirector).

With Metronomy, control engineers or other domain experts can leverage the plethora of MoCs in Ptolemy to capture the functional design of the system. Software/hardware engineers can leverage the flexibility and expressiveness (as well as the libraries of schedulers and

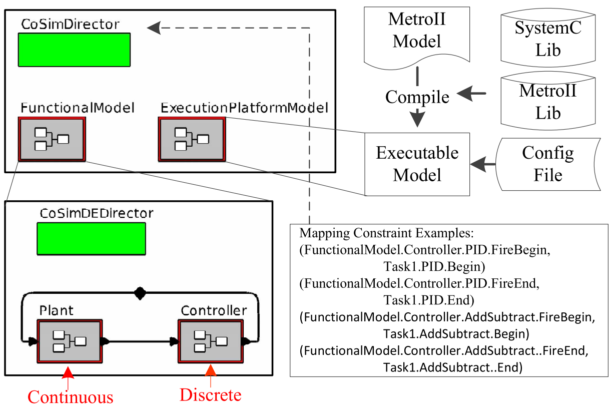

CoSimDirector
FunctionalModel
ExecutionPlatformModel
CoSimDEDirector
Plant
Controller
Continuous
Discrete
MetroII Model
SystemC Lib
Compile
MetroII Lib
Executable Model
Config File
Mapping Constraint Examples:
(FunctionalModel.Controller.PID.FireBegin, Task1.PID.Begin)
(FunctionalModel.Controller.PID.FireEnd, Task1.PID.End)
(FunctionalModel.Controller.AddSubtract.FireBegin, Task1.AddSubtract.Begin)
(FunctionalModel.Controller.AddSubtract..FireEnd, Task1.AddSubtract..End)

Fig. 1. ***Metronomy framework [13].***

annotators) of Metro II to design the architectural platform. System engineers can effectively integrate the two aspects in a cosimulation environment for codesigning various components and verifying the correctness and performance of their designs.

In particular, Metronomy exploits the contract-based design theory [24] to define an interface of timing contracts between the functional model and the architectural model. These contracts can be viewed as a set of timing assumptions and guarantees that are agreed upon between the domain experts who design the system functionality and the software/hardware engineers who implement such functionality on architectural platform. When conducting design space exploration through cosimulation, timing monitors can be implemented in Metronomy to check whether these timing contracts are being satisfied and rule out the infeasible designs.

In [13], Metronomy is applied to the design of two CPSs. In an aircraft electric power system example, a timing contract is set on the end-to-end latency (from sensing to actuation) of a control loop in the system. Through cosimulation of the controller and the implementation platform, different bounds of the timing contract are evaluated with respect to their impact on system performance and stability. The findings are then used to drive the codesign of controller and its implementation platform, e.g., whether a voltage protection mechanism should be added to the controller design when the timing bound is loose, or whether a faster communication bus should be employed in the architectural platform when the bound is tight. In another example of a printing press paper feed system, the sampling period of the paper feed controller and the operating frequency of the implementation processor are codesigned with respect to system performance, also based on cosimulation of the models and verification of the timing contracts.

D. Other Frameworks

There are a number of other academic modeling, simulation and design frameworks for embedded systems and CPSs, such as model-integrated computing

(MIC) [25]–[28], OpenMETA [29], BIP [30]–[32], Compaan/Laura [33], Spade [34], Sesame [35], CAKE [36], ForSyDe [37], MESH [38], Mescal [39], MILAN [40], Giotto [41], CHARON [42], and a compositional real-time scheduling framework [43]–[45] supported by the CARTS tool [46] and the Real-Time Xen visualization platform [47], [48]. There are also popular industrial tools such as MATLAB Simulink [49] and SCADE suite [50], both of which follow the model-based design paradigm [51]–[54].

These frameworks typically start the design with a functional model capturing the system functionality, and then refine it to software and hardware implementations. Some frameworks also provide synthesis tools that explore the design space. For instance, the MIC framework includes a general modeling environment (GME) [55], [56] for creating domain-specific models with a UML-based metamodeling language, a universal data model (UDM) [57] for providing uniform access to the GME metamodels, a graph rewriting and transformation (GReAT) tool [58] for model transformation, and a domain-independent tool DESERT [59]–[61] for generic constraint-based design space exploration. The OpenMETA design automation tool suite [29] provides a model integration platform for precise representation of semantic interfaces among modeling domains and a tool integration platform for automated design space exploration. The model integration platform leverages GME, the model integration language CyPhyML [62] and the formal specification language FORMULA 2.0 [63] to represent components, design spaces and designs, cross-domain interactions, composition constraints, data model interfaces, models of engineering process, and model transformation. The tool integration platform features the DESERT tool for synthesis, and also integrates methods for formal verification, reliability analysis and uncertainty quantification.

The behavior–interaction–priority (BIP) framework provides a rigorous model-based and component-based design flow [30]. As a type of architecture description language (ADL), BIP supports the construction of composite and hierarchically structured components from atomic components, and captures their behavior through layered application of interactions and priorities. More specifically, it uses the concept of connectors to specify interactions between components (as synchronization constraints), and uses priorities to filter possible interactions for further specifying system behavior. The framework has been applied for the modeling and verification of resource-constrained Internet-of-Things (IoT) applications [64], [65]. Recently, DesignBIP [66], a web-based graphical design studio, has been developed to facilitate the specification of BIP models, code generation from the models, and integration with a JavaBIP tool-set [67].

III. SYNTHESIS

Synthesis methods and tools explore a vast and heterogeneous design space, including the sensing, control, computation, and communication algorithms at the functional level, the software and hardware implementation platform, the mechanical components that affect the physical environment, and the human interaction interfaces. The exploration process tries to find feasible or even optimal designs with respect to objectives and constraints on a variety of system metrics, such as performance, safety, security, fault tolerance, reliability, extensibility, energy consumption, and monetary cost.

The synthesis of CPSs has been facing several major challenges.

- The functional complexity of system cyber components is rapidly growing, particularly due to the development of intelligent features and growing system scale. For instance, with the advancement of active safety features, such as those enabling autonomous driving and advanced driver-assistance systems (ADAS), the complexity of automotive electronic systems has increased drastically in the last two decades. Modern vehicles feature about 10–100 million lines of code (up from around 1 million in 2000), thousands of software functions, tens of thousands of functional requirements, and up to 25 GB of data for processing per hour [68]–[70]. Software and electronics were featured in 90% of automotive innovations in 2012, and will continue play a dominant role moving forward [71].
- The architectural platform is becoming more distributed and networked for many CPSs, and using more advanced and complex components. In the automotive domain, the number of electronic control units (ECUs) in luxury cars has more than doubled from under 50 to more than 100 in the past decade [72], and the ECUs are evolving from simple microcontrollers to multicore CPUs with graphics units and hardware accelerators. There is also a fundamental shift from the traditional federated architecture to the integrated architecture, where one function can be distributed over multiple ECUs and multiple functions can be supported on one ECU [73]. This trend leads to significantly more sharing and contention among software functions over the architectural platform, and increases design complexity [74].
- There are strong interdependencies among various components in CPSs. For instance, how effective a control design may affect the physical process significantly depends on the performance and reliability of its cyber implementation with embedded sensing, computation, and communication components. Further, the embedded platform itself is often affected by the physical environment, such as the impact of surrounding temperature on computation and the effect of environmental interference on wireless communication. These interdependencies make it significantly more challenging to explore the design space during synthesis.
- Different system objectives place conflicting requirements on design variables and parameters. For

instance, shorter sampling and control periods usually lead to better sensing and control performance [75], [76], but may be detrimental to schedulability [77]–[81], extensibility, and fault tolerance. Deciding these design variables requires quantitative analysis and careful tradeoffs among various objectives (metrics).

- The physical environment and human activities are dynamic and hard to predict. Consequently, the workload, performance, and requirements for the cyber and mechanical components could change significantly during operation. These uncertainties present great challenges in synthesis—for instance, only considering average case behavior may lead to unreliable or unsafe scenarios, while only addressing worst case behavior could lead to overconservative or even infeasible designs.

These challenges call for new synthesis methodologies and tools that can 1) tackle the functional and architectural complexity with accurate abstractions and efficient design space exploration algorithms; 2) systematically codesign the interdependent cyber and physical components while quantitatively trading off conflicting design metrics; and 3) effectively address the system uncertainties with adaptable, robust, and extensible designs.

Next, we will present our codesign approaches for synthesizing CPSs. These works address codesign in different application domains with a common methodology as follows.

- First, we identify the major factors that affect the interplay among various system components and design metrics, and formulate these factors as a set of interface variables and constraints. The interface variables could be concrete design variables such as sampling periods, or more abstract quantities such as sensing accuracy.
- Second, we analyze and formulate how various design metrics (which may relate to different system components) could be affected by the interface variables and the constraints defined on them.
- Finally, based on the formulation and analysis from the steps above, we develop synthesis algorithms to explore the values of interface variables and constraints as well as other design variables, with respect to design metrics and requirements.

A. HVAC Controller and Sensing Platform Codesign

In [82], we presented an approach to codesign a heating, ventilation, and air conditioning (HVAC) control algorithm and an embedded sensing platform in energy-efficient buildings, as shown in Fig. 2. The codesign is motivated by our observation that the performance and energy cost of HVAC control is significantly affected by the number, location and accuracy of temperature sensors.

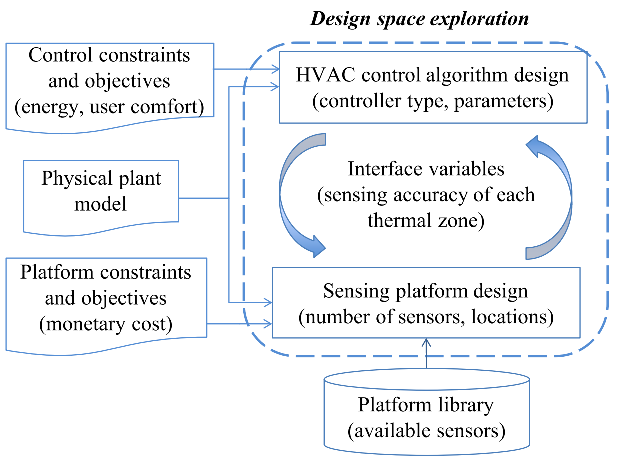

Design space exploration
Control constraints and objectives (energy, user comfort)
HVAC control algorithm design (controller type, parameters)
Physical plant model
Interface variables (sensing accuracy of each thermal zone)
Platform constraints and objectives (monetary cost)
Sensing platform design (number of sensors, locations)
Platform library (available sensors)

Fig. 2. ***Codesign of HVAC control algorithm and embedded sensing platform for energy-efficient buildings.***

In the codesign process, we first define sensing accuracy of each thermal zone, an abstract quantity that depends on the sensor selection and deployment, as the interface variables to capture the interplay between HVAC control and sensing platform design. We then evaluate the control performance (measured by a discomfort index) and energy cost of six control algorithms, ranging from simple ON–OFF control to robust model predictive control (MPC) with extended or unscented Kalman filters, under different levels of sensing accuracy. We leverage the collected data from a building testbed to analyze the relation between the abstract sensing accuracy variable of a thermal zone and the concrete number, location, and accuracy of individual sensors in that zone (which decide the sensing platform cost). Finally, based on these analysis, we explore the codesign of HVAC control algorithm and embedded sensing platform to minimize the energy cost and monetary cost while satisfying the constraints on building tenants' comfort level.

For energy-efficient buildings, holistically addressing heterogeneous components is not only important at design stage but also highly beneficial during operation time. In [83] and [84], coscheduling heterogeneous energy demands (HVAC, electric vehicles charging) and supplies (grid, renewable sources, battery storage) shows a reduction of 4% in energy cost and 15% in peak demand. In [85], coscheduling HVAC control and datacenter operations in mixed-use buildings, with shared energy supplies and cooling infrastructure, shows a reduction of 3%–17% in energy cost and 3% in carbon footprint.

B. Control Performance and Schedulability Codesign

In [81], we presented an algorithm to jointly address control performance and schedulability in controller area network (CAN)-based distributed real-time systems. In this work, the interface variables between the control function-

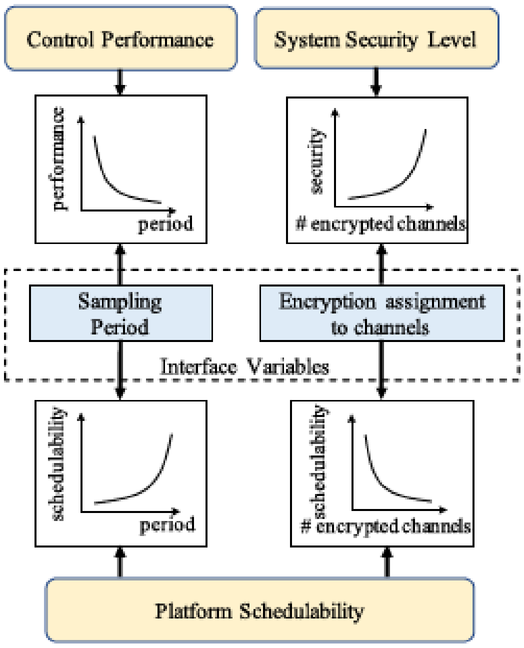


Fig. 3. *Codesign of security, control performance, and schedulability with interface variables [86].*

ality layer and the embedded implementation layer are the activation periods of tasks and messages. These are concrete design variables being explored in our codesign algorithm.

Intuitively, shorter periods for sensing and control tasks often provide better performance, but increase system load and may jeopardize schedulability (assuming the computation time stays at the same). To quantitatively address this tradeoff, the codesign algorithm first approximates the performance of each control loop in the system with a piecewise-linear function of its sampling period and end-to-end delay, and then optimizes the periods of tasks and messages by exploring the linear partitions of the approximated functions and solving a series of geometric programming (GP) problems. The optimization process sets system-level control performance as the objective function, and uses schedulability of tasks and messages as design constraints, along with latency deadlines of functional paths.

C. Security, Control, and Schedulability Codesign

In [86], we presented a cross-layer codesign framework that addresses the tradeoff between security and control performance, while guaranteeing platform schedulability for CPSs. We consider systems where multiple control loops share a common embedded platform, with messages transmitted from sensors (vision sensors, global positioning system, ultrasound, etc.) to controllers and from controllers to actuators. Attackers may eavesdrop on the message communication medium and further reconstruct the system state. This would not only result in a loss of privacy but also lead to other malicious attacks. Security techniques such as message encryption could be applied to mitigate risk, however they will also introduce computation and communication overhead, through the elongation of message transmission time, the addition of decryption tasks on computation units, and consequently the execution time increase of control tasks on the same units due to resource contention. This overhead will in turn have significant impact on system schedulability and control performance, as detailed in [86].

To quantitatively analyze the interplay among control, security, and schedulability, we identify the sampling periods of control tasks and the selection of sensing messages for encryption as the interface variables (Fig. 3). Similarly as discussed above in [81], shorter sampling periods lead to better control performance but worse schedulability, and *vice versa*. Selecting more messages for encryption enhances system security however also worsens schedulability because of the added overhead, in which case the sampling periods may have to be increased to help schedulability (i.e., to ensure each sample can be processed within its period) and thereby lead to worse control performance.

We mathematically formulated the relations between design metrics (security, control performance, schedulability) and interface variables (sampling periods, encryption assignment to messages) in [86]. In particular, the system security level is measured based on either observability Gramian or Kalman filter, and takes into account which messages are encrypted and to what extent. Based on this formulation, a simulated-annealing-based algorithm is developed to optimize security or control performance under schedulability constraints.

Fig. 4 shows the tradeoff between security and control performance when our approach is applied to an industrial example in the automotive domain. The generated Pareto

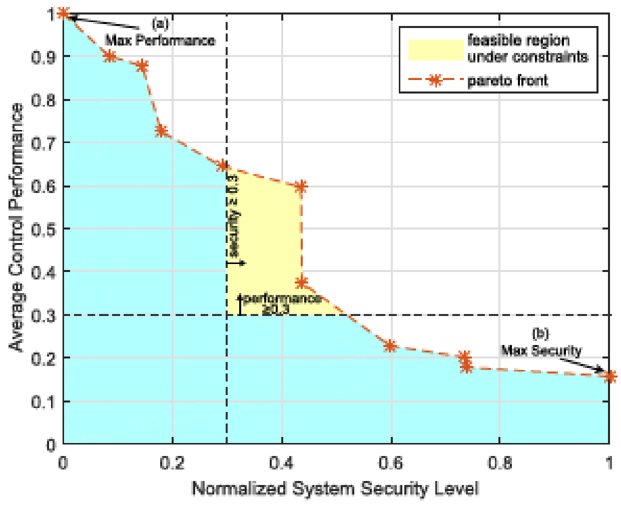


Fig. 4. *Pareto front between normalized security level and control performance for an industrial example [86]. An example feasible region denotes all feasible solutions under requirement that control performance ≥ 0.3 and security level ≥ 0.3.*

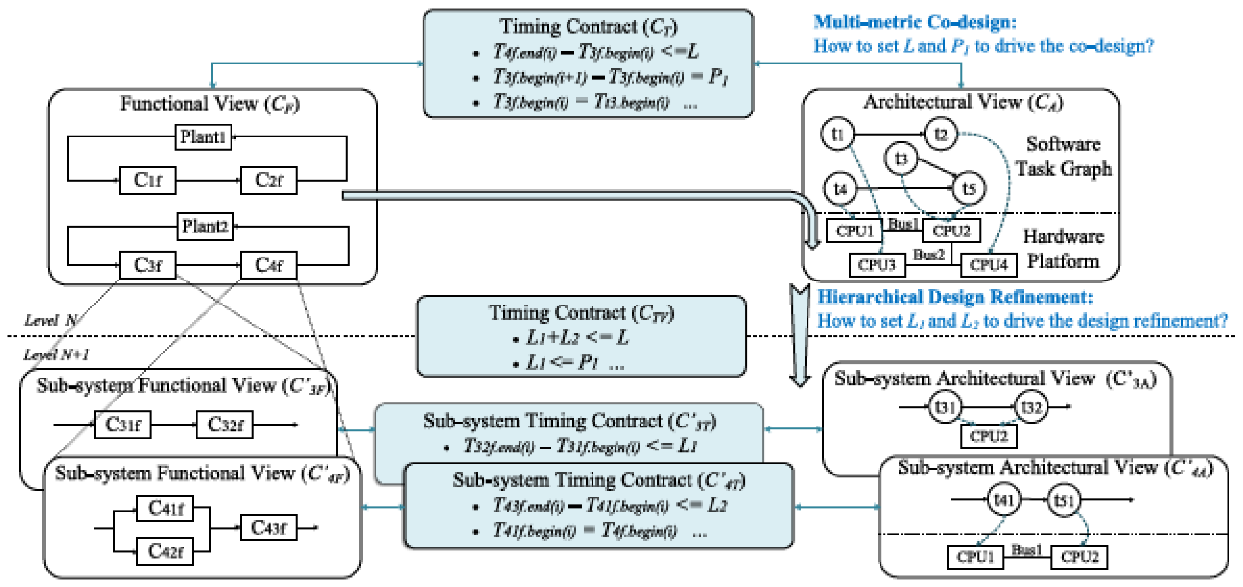

Timing Contract (C_T)
$T_{4f.end(i)} - T_{3f.begin(i)} <= L$
$T_{3f.begin(i+1)} - T_{3f.begin(i)} = P_1$
$T_{3f.begin(i)} = T_{t3.begin(i)}$ ...
Multi-metric Co-design:
How to set L and P_1 to drive the co-design?
Functional View (C_F)
Plant1
C1f
C2f
Plant2
C3f
C4f
Architectural View (C_A)
Software Task Graph
Hardware Platform
CPU1
Bus1
CPU2
Bus2
CPU3
CPU4
Level N
Level N+1
Timing Contract (C_{TF})
$L_1+L_2 <= L$
$L_1 <= P_1$ ...
Hierarchical Design Refinement:
How to set L_1 and L_2 to drive the design refinement?
Sub-system Functional View (C'_{3F})
C31f
C32f
Sub-system Timing Contract (C'_{3T})
$T_{32f.end(i)} - T_{31f.begin(i)} <= L_1$
Sub-system Architectural View (C'_{3A})
Sub-system Functional View (C'_{4F})
C41f
C42f
C43f
Sub-system Timing Contract (C'_{4T})
$T_{43f.end(i)} - T_{41f.begin(i)} <= L_2$
$T_{41f.begin(i)} = T_{4f.begin(i)}$ ...
Sub-system Architectural View (C'_{4A})

Fig. 5. *An illustrating example of timing contracts and their roles in synthesis.*

front provides a quantitative measurement of the tradeoff and identifies a feasible region that is important for making design decisions. For instance, an example feasible region shown in the figure meets the requirements that the normalized control performance should be no less than 0.3 and the system security level should be no less than 0.3. Without codesign, the designers might get a solution that violates security requirement if they only optimize for control performance [point (a) in the figure], or a solution that violates performance requirement if they simply choose to encrypt all messages [point (b)]. This example shows the close interdependency among various design metrics in CPSs and the importance to codesign them.

D. Timing Contracts for Codesign

Timing is a central element in CPS synthesis, affecting both functional correctness and various design metrics. Thus, timing variables and constraints are often identified as the interface to drive the codesign. Two examples are already introduced above—task and message activation periods along with end-to-end delays are the interface variables to drive the control performance and schedulability codesign in [81], while control sampling periods are part of the interface variables to drive the codesign of security, control performance, and schedulability in [86]. There could be many other types of timing variables and formulations of timing constraints. Below, we present a general methodology for timing-driven codesign and synthesis based on exploration of timing contracts.

The paradigm of contract-based design provides the general methodology of using contracts to reduce design complexity and represent design refinement [24], [87]–[91]. However, the exploration of timing contracts has typically been done manually or is limited to simulation-based approaches (as in our prior work [13] introduced in Section II), and it has not been used sufficiently to drive the synthesis process. Next, we use an illustrating example to demonstrate our ideas in timing contracts exploration for synthesis.

1) Multimetric Codesign With Timing Contracts: In Fig. 5, at abstraction level N, a functional view (viewpoint) C_F, and an architectural view C_A are defined to capture the system's functional behavior and architectural platform, respectively. In the contract-based design paradigm, both C_F and C_A can be defined as contracts that provide guarantees under assumptions, where guarantees and assumptions are specified as a set of assertions/constraints. In addition to C_F and C_A, a timing contract/view C_T is defined to capture the system's timing behavior and constraints.[1]

The conjunction of C_F and C_T (denoted as $C_F \sqcap C_T$, with formal definition introduced in [24] and [87]) defines the system's functional behavior under timing assumptions/constraints defined in C_T. For instance, in the figure, it defines the functional behavior when the end-to-end (sensor-to-actuator) latency of a control loop including components (subsystems) C_{3f} and C_{4f} is within L, and when the control loop's sampling period is P_1 (corresponding to the activation period of sensing component C_{3f}, where $T_{3f.end(i)}$ denotes the time associated with the end of the ith execution of C_{3f}, with other notations similarly defined). This behavior relates directly to the system's functional correctness, control performance, sensing accuracy, and other metrics related to functionality and timing.

[1]C_T may be further refined into functional and architectural timing contracts to facilitate analysis and function-architecture cosimulation as in [13].

The conjunction of C_A and C_T (i.e., $C_A \sqcap C_T$) defines the system's architectural behavior under timing constraints. This may relate directly to metrics such as schedulability, extensibility, and fault tolerance.

Often the timing constraints have opposite impact on different metrics. As discussed above, shorter end-to-end latencies and sampling periods (e.g., smaller values of L and P_1 in Fig. 5) usually lead to better performance but worse schedulability. The goal in codesign with timing contracts is to answer: how to systematically set the timing constraints as contracts between different views when considering multiple conflicting design metrics? For instance, what values should L and P_1 be to provide the best control performance while ensuring schedulability? What about to ensure certain control performance level while providing the best extensibility?

Our work introduced above [81], [86] is a first attempt at addressing these questions. The general methodology includes the following aspects.

- Timing contracts formulation: Depending on system characteristics and design focus, the constraints in timing contracts could take various mathematical formalisms. Linear, quadratic, exponential, or other functions can be used to form constraints on timing-related variables. Logic operators (e.g., conjunction, inclusive or exclusive disjunction, existential qualification) can be used for defining logic relations, particularly those related to resource contentions. More generally, formalisms such as linear temporal logic (LTL) [92], real-time logic (RTL) [93], signal temporal logic (STL) [94], logic of constraints (LOC) [95], and logical execution time (LET) [41] can be used to capture the complex requirements on timing variables and events. Leveraging these mathematical formalisms, the key in defining contracts is to identify the critical timing factors for codesign, and choose the right abstraction levels and formalisms to capture them with constraints. For instance, sensitivity analysis based on estimation and simulation can be used to evaluate whether end-to-end latencies have significant impact on extensibility, and if so, which path latencies are the most critical. Timing constraints are defined at event level (e.g., timing order between events) in [13] to drive simulation-based exploration, and defined at task and component (functional block) levels in [81], [86], and [96] to facilitate analytical synthesis algorithms. Different activation patters (e.g., periodic, sporadic, or aperiodic) and timing models (e.g., worst case, average case, or probabilistic models) could be used to define events, components, and tasks. Constraints could include hard, firm, or soft deadlines, or weakly hard deadlines that are allowed to be violated based on a defined pattern (e.g., at most m violations in any k consecutive instance) [97].
- Design metrics modeling and formulation: Once timing contracts are formulated, the next step is to derive the relations between various design metrics and the timing variables/constraints in the contracts. In some cases, these relations can be captured in closed-form formulations, such as the relation between security level and the selection of sensing messages for encryption in [86]. In other cases, approximated formulations, simulation-based curve fitting, or direct integration with simulators may have to be used, such as the relation between control performance and activation periods in [81].

  In addition to control performance, security, and schedulability, many other design metrics significantly depend on system timing behavior and should be addressed in codesign with timing contracts. For instance, extensibility represents how much a system may be changed without major redesign and reverification effort, which is imperative for large-volume and long-lifetime CPSs such as automotive and avionics systems. In our prior work [98]–[100], extensibility metrics are defined to measure how much task (or action) execution time can be increased without violating design constraints. These metrics depend on timing constraints, architectural platform factors, and synthesis choices, and should be codesigned with other metrics. Another important metric is fault tolerance. In [101], a fault tolerance metric measures the likelihood of soft errors being detected and corrected through embedded error detection (EED) or explicit output comparison (EOC) approaches without violating timing constraints. The tradeoff between fault tolerance and other metrics such as extensibility and communication cost are evident in our recent study [102], and should be addressed with timing contracts.
- Multimetric codesign algorithms: Once the timing contracts and the design metrics are formulated, synthesis algorithms need to be developed for exploring timing constraints and other design variables (e.g., task generation, allocation, and scheduling). In general, the exploration can be formulated as a constrained multiobjective optimization problem and is often NP-hard. Randomized algorithms such as simulated annealing can be applied in many cases, however, the complexity usually makes it difficult to obtain good (or even feasible) solutions for practical systems.

  For some systems and metrics, the problem may be formulated in an integrated formulation with closed-form representations of all objectives and constraints. In these cases, techniques such as mathematical programming, greedy heuristic, dynamic programming, or a combinational of them could be applied to balance algorithm optimality and complexity. For instance, mixed-integer linear programming, geometric programming, and their combination with heuristics are used in our prior synthesis work [81], [98], [99], [103]–[107]. In some cases, for complexity concern, it is necessary to develop separate exploration engines for timing constraints, for task generation, allocation and scheduling, for communication synthesis, and for other design variables (e.g., CPU frequencies as architectural variables, control algorithm parameters as functional variables), and to efficiently iterate among these engines for joint optimization.

2) Hierarchical Design Refinement With Timing Contracts: The above timing contracts for codesigning multiple design metrics can be regarded as "horizontal" contracts between different views. There are also "vertical" timing contracts that could be defined to facilitate the design refinement across abstraction levels, which may be viewed as codesign of multiple components (subsystems).

Using Fig. 5 as an example, at the lower abstraction level $N+1$ of the system, each functional component is refined to lower level components. Functional view C'_{3F} and C'_{4F} are refined from components C_{3f} and C_{4f} at level N. Correspondingly, architectural views C'_{3A} and C'_{4A} refine parts of the architectural platform to which C_{3f} and C_{4f} are mapped. For instance, task t_3 may be a virtual task (i.e., not corresponding to a software task in the operating system) at level N, and is refined to lower level tasks t_{31} and t_{32} at level $N+1$. Tasks t_4 and t_5 at level N are refined to tasks t_{41} and t_{51} at level $N+1$, respectively.

During implementation, functional components are mapped to the tasks in architectural views at the corresponding level. For instance, in Fig. 5, C_{3f} is mapped to t_3 at level N, and C_{4f} is mapped to t_4 and t_5. At the lower level $N+1$, C_{31f} and C_{32f} are mapped to t_{31} and t_{32}, respectively (both tasks are running on $CPU2$). C_{41f} and C_{42f} are mapped to t_{41} running on $CPU1$, and C_{43f} is mapped to t_{51} running on $CPU2$. Note that both many-to-one and one-to-many mapping between functional components and architectural tasks are allowed for generality and flexibility.

Timing contracts C'_{3T} and C'_{4T} are defined and should be consistent with C_T, through constraints defined in the vertical timing contract C_{TV}. For instance, in systems that are designed following the synchronous assumption [96], [108], the end-to-end latency deadline from the beginning of C_{31f} to the end of C_{32f} (denoted by L_1) should be within the activation period P_1 of C_{3f}, to ensure that each activation of C_{3f} can be completed before its next activation. Furthermore, the sum of the end-to-end latency deadlines of C'_{3F} and C'_{4F}, denoted as L_1 and L_2, respectively, should be within the entire control path deadline L at level N. There are also timing constraints across architectural views at different abstraction levels in C_{TV}, and constraints between functional components and architectural tasks in C'_{3T} and C'_{4T} (e.g., setting $T_{31f.(i)} = T_{t31.\mathrm{begin}(i)}$ to synchronize the execution of C_{31f} and t_{31}). For simplicity, they are not shown in Fig. 5.

The goal in hierarchical refinement with timing contracts is to answer: How to assign timing "budget" as contracts for each lower level component, which has significant impact on the design of each subsystem and the overall system quality? For instance, assuming L is set to 200 ms, we will need to address questions such as " if we set L_1 to 100 ms and L_2 to 100 ms, can we find feasible implementations/mappings for C_{3f} and C_{4f}?"; "What if we set L_1 to 80 ms and L_2 to 120 ms, will it provide better extensibility for the system?"; and so on. This is challenging to address since often the timing budgets (constraints) have to be decided before the corresponding lower level components are designed (their design choices depend on the timing budgets themselves and are often carried out by different teams). Furthermore, in practical systems, there are often multiple timing constrains for each lower level component, and components that do not share the same functional path may also impact each other if they share resources on the architectural platform.

There are two important aspects in using timing contracts to address these challenges.

- Vertical timing contracts formulation: The formulation of vertical contracts may leverage the similar mathematical formalisms as discussed above for horizontal contracts. Furthermore, the vertical contracts should be able to capture timing behavior and constraints across the hierarchical structure of systems. For instance, in [96], we presented a formulation called firing and execution timing automata (FETA), which can capture the periodic timing behavior of runnables at functional layer and of software tasks at embedded platform layer. The hierarchical composition of FETAs further corresponds to the mapping of multiple runnables to a task. Having such unified FETA formalism enables cross-layer timing analysis during synthesis, and could be leveraged in formulating vertical timing contracts.
- Hierarchical timing budgets exploration: Once the vertical timing contracts are formulated, synthesis algorithms are needed for exploring timing budgets during design refinement. One direction is to develop methodologies for a first estimation of the timing complexity of components, which measures how much impact a component may have on certain timing-related design metric (e.g., schedulability, extensibility, fault tolerance). Different timing complexity measurements may be required for different metrics. As an example, for schedulability, the concepts of local utilization and alpha ratio based on FETA in [96] are shown to be effective in estimating the impact of individual runnables on system schedulability.

The estimations of timing complexity for components can then be used to drive the timing budgets assignment and ultimately the design of components, such as the software task generation and mapping for them. For example, in Fig. 5, L_2 needs to be set to drive the task generation and mapping for C_{41f}, C_{42f}, and C_{43f}. To achieve effective exploration, an iterative approach between the timing budgets assignment and the task generation and mapping is needed.

E. Other Works

There are a number of other papers that address multiple system aspects and objectives in the synthesis of CPSs. In [105], we addressed both security and timing safety requirements in mapping software tasks onto CAN-based automotive platforms. We used message authentication codes (MACs) to protect against masquerade and replay attacks on CAN networks, and developed a mixed-integer linear programming (MILP) formulation and a greedy heuristic algorithm for exploring task allocation, signal packing, MAC sharing, and priority assignment while meeting security and timing constraints.

The work in [105] improves security for CAN-based systems by addressing it together with other design concerns during the synthesis process. However, with limited bandwidth and message size, it is still very challenging to apply authentication mechanisms such as MACs on CAN networks. In [106], we further addressed security issues for next-generation automotive buses that use time-division multiple-access (TDMA) communication, such as time-triggered Ethernet [109], FlexRay [110], or time-sensitive networking [111]. We used an authentication mechanism with time-delayed release of keys, and developed an algorithm to jointly address security and timing requirements by exploring task allocation, priority assignment, network scheduling, and key-release interval length. Experiments based on time-triggered Ethernet demonstrate that both security and timing requirements can be met through the joint formulation and synthesis—and may not be met if the authentication mechanism is applied after task synthesis, even with the much larger bandwidth and message size than CAN. Following the work in [105] and [106], we proposed a general methodology for addressing security and other system objectives together during synthesis, and apply it to a vehicle-to-vehicle (V2V) application [112].

In [96], we developed a model-based synthesis flow for automotive software systems that follow the AUTOSAR standard [113]. The synthesis flow optimizes the generation of AUTOSAR runnables from synchronous functional models, the mapping of runnables onto software tasks, and the allocation of tasks onto ECU cores. It introduces the formalism of FETA to capture the worst case execution time (WCET) of functional blocks, runnables, and tasks at each activation, and then leverages the unified FETA formulation to model system timing behavior and address timing constraints throughout the entire synthesis process. Furthermore, the synthesis flow considers a variety of software engineering objectives (runnable modularity, reusability, code size, memory cost) with timing behavior in a holistic fashion. In particular, the flow focuses on trading off modularity with schedulability during runnable synthesis, and on minimizing memory cost under schedulability constraints during task synthesis. The results demonstrate the importance of using a codesign methodology in synthesis and addressing timing across the layers of functional model, runnable, and software tasks. In [108], a similar codesign formulation was developed for direct generation of tasks from synchronous models, with respect to modularity, reusability, code size, and latency.

For control applications, there are various efforts on conducting controller design with the consideration of platform impact (e.g., timing delay, packet dropping, and scheduling policy) [75], [77]–[80], [114]–[131]. In [132], Roy *et al.* discussed the issues in traditional isolated process of controller design and implementation platform design, in particular the challenges in preserving model-level semantics and safety properties during the transformation of high-level controller models into implementations. They then provided a comprehensive review of recent efforts in control-platform cosynthesis, in which the control algorithms and platform parameters are designed together.

IV. VERIFICATION AND VALIDATION

Verification and validation tools are essential in codesigning CPSs. They ensure that system specification satisfies user and mission requirements, and system implementation meets the specification. Similarly as for synthesis, verification and validation face significant challenges from increasing functional complexity, distributed and networked architectural platform, heterogeneous system components, and stringent requirements on various system metrics. They not only have to guarantee functional correctness, but also need to ensure that requirements on nonfunctional properties (e.g., performance, security, reliability) are met. Moreover, properties that are often regarded as nonfunctional in traditional computing systems, such as timing behavior, may have significant impact on functional correctness in CPSs. The close interaction between discrete cyber components and continuous physical domain, as well as the uncertainties from dynamic environment and human activities, presents further challenges in CPS verification and validation.

Next, we will first present our approach to tackle CPS verification challenges, which combines platform-aware functional verification with constrained platform synthesis in a common framework. We will then introduce the application of this methodology in a cross-layer codesign and verification framework for connected vehicles.

A. Collaborative Functional Verification and Platform Synthesis

In current practice, verification of system's functionality (abstracted through a functional model) is often conducted without much consideration of the underlying architectural platform. On the other hand, synthesis methods that explore platform design choices are often oblivious of the high-level functional requirements. Such disconnected

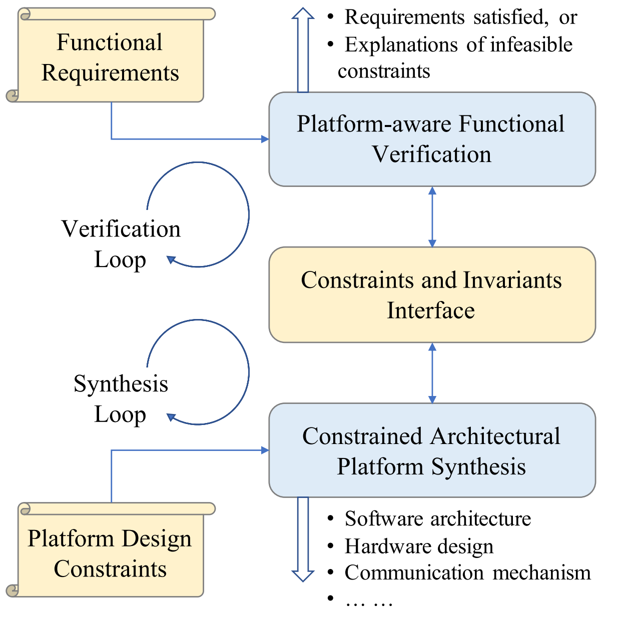

Functional Requirements
Requirements satisfied, or
Explanations of infeasible constraints
Platform-aware Functional Verification
Verification Loop
Constraints and Invariants Interface
Synthesis Loop
Constrained Architectural Platform Synthesis
Platform Design Constraints
Software architecture
Hardware design
Communication mechanism
… …

Fig. 6. ***Collaborative functional verification and platform synthesis framework for CPSs.***

approach may be feasible for traditional computing systems in which functional and nonfunctional properties can be clearly separated. However, it is unsuitable for many CPSs where nonfunctional quantities such as timing and reliability have direct and complex impact on system correctness, and using it could easily lead to infeasible designs (i.e., either cannot be verified with respect to system properties or cannot be implemented successfully) or inferior designs. In [133], we proposed a collaborative functional verification and platform synthesis framework for integrating the two currently separated steps in the design cycle, as shown in Fig. 6.

The idea in our approach is to divide the problem of verifying CPSs into two collaborative subproblems: 1) functional verification under constraints and invariants guaranteed by synthesis (e.g., verifying safety property when assuming sensor-to-actuator delay is within certain bound $[D_{\mathrm{MIN}}, D_{\mathrm{MAX}}]$); and 2) software/hardware platform synthesis under constraints and invariants specified by verification (e.g., synthesizing software design while meeting the same bound $[D_{\mathrm{MIN}}, D_{\mathrm{MAX}}]$ on sensor-to-actuator-delay).

The constraints and invariants interface can be viewed as a contract between functional verification and platform synthesis. For instance, the above example of "sensor-to-actuator delay is within bound $[D_{\mathrm{MIN}}, D_{\mathrm{MAX}}]$" is in fact an assume/guarantee contract, where the verification process assumes the delay bound will be guaranteed in the synthesis result. Thus, our framework integrates two essential aspects in CPS design and solve them systematically. By formalizing and managing the exchange of constraints and invariants through the verification-synthesis interface in coupled iterations, this approach could significantly improve the efficiency of both verification and synthesis. Next, we will introduce how we apply this methodology in codesigning and verifying connected vehicle applications, by analyzing the impact of communication delays and message losses on functional properties.

B. Cross-Layer Design and Verification of Connected Vehicles

Next-generation autonomous and semi-autonomous vehicles will not only percept the environment with their own sensors, but also communicate with surrounding vehicles and infrastructures to improve vehicle safety and transportation efficiency. The design, verification, and validation of various V2X [i.e., vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I)] systems involve multiple layers, from application functionality to vehicular communication networks and to the software and hardware of individual vehicles [134]. They are concerned with stringent requirements on timing, safety, security, dependability, cost, and resources across these layers.

The underlying sensing, computation, and communication platform, within vehicles and between them, fundamentally influences the correctness of functional properties at the V2X application layer. For instance, the time it takes a vehicle to broadcast its position, velocity, and acceleration to nearby vehicles has a significant impact on the size of safety zones in which the vehicles can collaboratively perform collision avoidance [133]. This timing is in turn influenced by many platform decisions and factors, such as the choice of sensors and their accuracies, the availability and capability of computation resources, the in-vehicle communication latencies over buses, the design of V2X communication protocol and the environment disturbance on such communication, and the overhead incurred by added security measurements for V2X and in-vehicle communications. Thus, the correctness of a functional property at the application layer, such as "vehicles in a platoon should always maintain a safe distance from one another," inherently depends on the design decisions and execution behavior of the underlying platform.

Based on the above observation, we have been developing CONVINCE, a cross-layer modeling, exploration, and validation framework for connected vehicles (Fig. 7). One of the ideas of CONVINCE is to integrate the functional verification of V2X and self-driving applications with platform synthesis of inter-vehicle communication and intra-vehicle (software and hardware) architecture, following the general methodology introduced earlier in this section. The CONVINCE framework includes mathematical models, synthesis, verification and validation algorithms, and a heterogeneous simulator in a holistic environment. As shown in Fig. 7, it explores a variety of design options with respect to constraints and objectives across system layers, and it takes into consideration of environment disturbance and possible security attacks.

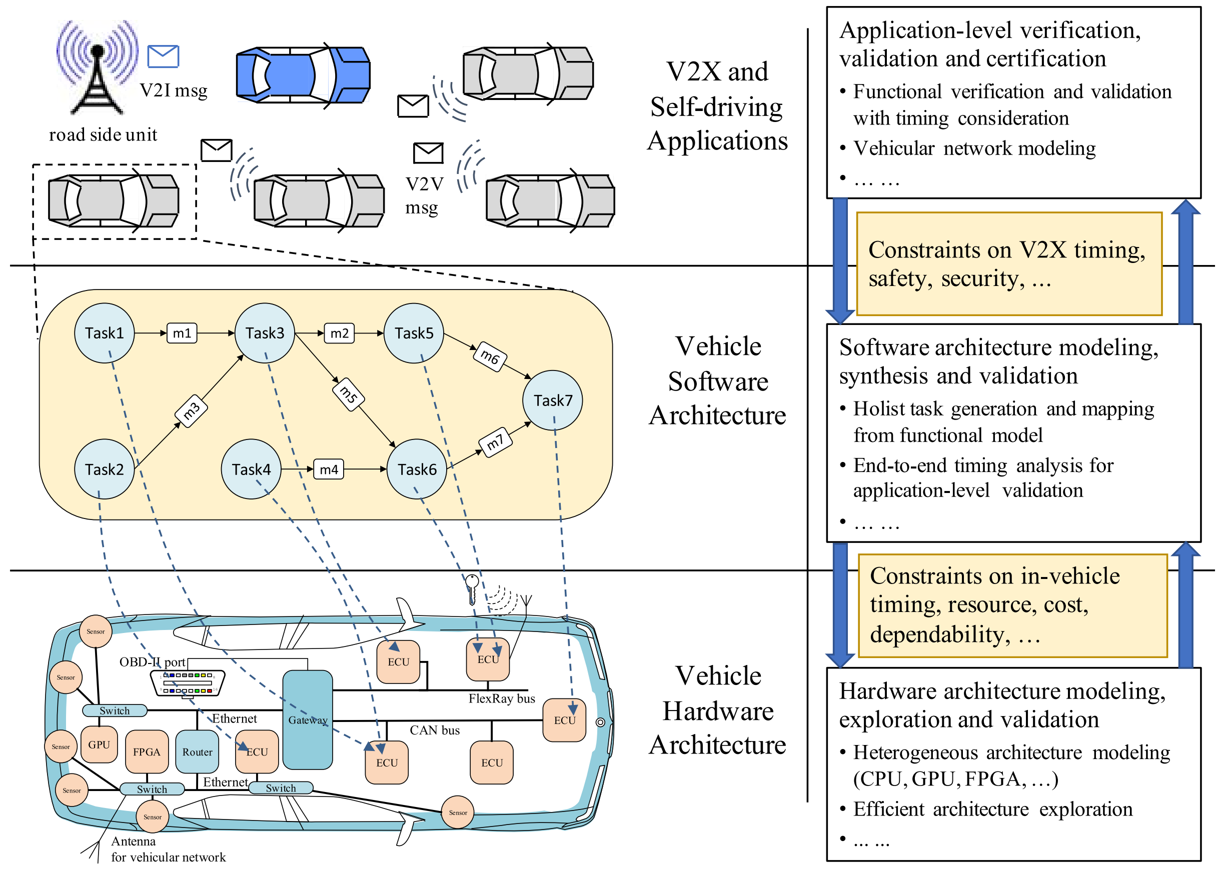

V2I msg
road side unit
V2V msg
V2X and Self-driving Applications
Application-level verification, validation and certification
• Functional verification and validation with timing consideration
• Vehicular network modeling
• … …
Constraints on V2X timing, safety, security, …
Task1
Task2
Task3
Task4
Task5
Task6
Task7
m1
m2
m3
m4
m5
m6
m7
Vehicle Software Architecture
Software architecture modeling, synthesis and validation
• Holist task generation and mapping from functional model
• End-to-end timing analysis for application-level validation
• … …
Constraints on in-vehicle timing, resource, cost, dependability, …
Sensor
OBD-II port
Switch
Ethernet
Gateway
ECU
FlexRay bus
CAN bus
GPU
FPGA
Router
Antenna for vehicular network
Vehicle Hardware Architecture
Hardware architecture modeling, exploration and validation
• Heterogeneous architecture modeling (CPU, GPU, FPGA, …)
• Efficient architecture exploration
• … …

Fig. 7. *CONVINCE: Cross-layer modeling, exploration, and validation framework for next-generation connected vehicles.*

In [134], we presented the preliminary design of CONVINCE and a case study of the collaborative adaptive cruise control (CACC) application, where every vehicle communicates with its preceding vehicle and adaptively maintains a safe distance. Through mathematical analysis and simulations, the case study demonstrate the impact of V2V communication delays on the performance of CACC.

In [135], we applied CONVINCE to the design of centralized autonomous intersection management, a complex V2X application that is significantly affected by the underlying platform. In a centralized autonomous intersection, an intersection manager accepts requests from approaching vehicles via V2I messages and schedules the order for those vehicles to cross the intersection. Previous papers in the literature assume perfect communication between vehicles and infrastructures, and do not explicitly consider communication delays [136]–[140]. However, significant message delays (up to several hundred milliseconds in the worst case) and losses could happen in vehicular network under dense traffic [141]–[143], and the situation could be even more severe when the communication channels are under malicious jamming or flooding attacks [134], [144]. In those cases, the previous approaches that lack consideration of messages delays and losses may lead to system deadlock or unsafe situations.

To address the above issue, we presented a delay-tolerant intersection management protocol in [135], and applied CONVINCE for the modeling, simulation, and verification of the protocol. The framework is able to verify that the protocol's safety property is guaranteed under any circumstance, and that its liveness and deadlock-free properties are guaranteed if the maximum communication delay is bounded and known. This bound should then be used as a constraint for the synthesis of the underlying communication and computation platform, i.e., as a constraint in the verification-synthesis interface in Fig. 6. Simulation results also demonstrate that 1) our protocol may significantly improve the intersection efficiency in normal situations (i.e., communication delay is within hundreds of milliseconds); and 2) the protocol performance worsens when communication delay increases, and thus it is important to quantitatively analyze such impact and be able to answer questions such as "for autonomous intersection to outperform traffic lights, what bound should the communication delay satisfy and how to ensure such bound in platform synthesis."

V. CONCLUSION AND FUTURE DIRECTIONS

We discussed the importance and challenges of holistically addressing modeling, synthesis, and verification of CPSs. We presented emerging design methodologies, algorithms, and tools for addressing CPS codesign, and introduced their applications in domains such as automotive systems, vehicular networks, and energy-efficient buildings.

There are still many open challenges remaining in CPS codesign. Below, we discuss some promising future directions for addressing them.

- Automated model generation and update: While a number of model-based frameworks have been proposed for capturing and analyzing CPSs (some of which are introduced in Section II), developing and maintaining those formal or semiformal models remains quite challenging in practice. There is typically a steep learning curve for designers to get familiar with the syntax and semantics of the frameworks, and to effectively apply them in modeling complex systems. This is especially the case in CPS codesign, as designers often need to model intrinsically heterogeneous components with different semantics and/or languages. The model development process is usually time-consuming and error-prone, and maintaining and updating the models throughout their lifetime may take even more effort. Thus, developing an automated or semiautomated process for model generation and update is of great interest. We envision an environment where designers only need to provide high-level informal descriptions of the system (e.g., in English) and models are automatically generated/updated based on those descriptions. To realize this, advanced techniques such as natural language processing (NLP) could be leveraged, as well as intuitive graphical interfaces and comprehensive design libraries. In addition, an interactive human-tool interface, in which designers could incrementally check, query, and modify the models, should facilitate the model development/update process (an interactive interface should also facilitate synthesis and verification as discussed below). Automated abstraction/rewriting techniques such as "lifting" are also promising directions [145].
- Agile model abstraction and integration: CPS codesign requires integration of heterogeneous cyber and physical components that are captured at various levels of abstraction. Moreover, based on the specific codesign problems, different abstraction levels may be chosen for the same component. For instance, in our approach to the codesign of an HVAC controller and sensing platform for buildings [82], a simplified resistor–capacitor (RC) network model from [146] is used to capture the room thermal dynamics, and is shown to be effective for facilitating the choices on different controllers and sensors. This model, however, may not be sufficiently accurate for designing the controllers. Detailed thermal dynamics models (such as the ones built in EnergyPlus [147]) could provide the needed accuracy but are often too complex for control design. Thus, models that are between these two abstraction levels need to be explored, through techniques such as model order reduction [148], [149] and data-driven modeling [150]–[155]. From the perspective of codesign tools, it is important to facilitate agile exploration of different model abstractions, such as the exploration of different thermal dynamic models in building design and operation. This will require well-defined component interfaces and system execution semantics that can support "plug-and-play" of models at different abstraction levels.
- Comprehensive cross-layer cosynthesis: The design of sensing, control, computation, and communication algorithms at the functional layer is closely intertwined with the embedded software and hardware design at the platform layers. The codesign/cosynthesis approaches presented in Section III analyze and leverage such cross-layer interdependencies to improve the overall system metrics such as performance, safety, and security. While showing promising results, these approaches have only explored the surface of cross-layer cosynthesis in CPSs. There is great potential to further investigate the interactions between CPS components across different system layers, discover and quantify the dependencies in their design choices, and develop holistic formulations and algorithms to cosynthesize them. For instance, HVAC control algorithms may be codesigned with the sensing platforms from scratch, beyond just being selected from existing controllers as in [82]. Vision-based sensing algorithms could be codesigned with the computation and communication platforms to enable efficient real-time video processing, as we started exploring in [156].
- Efficient algorithms for exploring heterogeneous design space: Compared with traditional embedded systems, codesign problems in CPSs often involve a more heterogeneous and complex design space. The problem formulations could include a large number of discrete and continuous design variables, and many nonlinear or even nonconvex constraints. Thus, when developing algorithms to explore such design space, it is usually infeasible to directly apply mathematical programming techniques such as linear programming and geometric programming, and ineffective (too slow or too far from the optimal solutions) to directly use randomized methods such as simulated annealing and genetic algorithms. New approaches need to be developed for efficient exploration of the heterogeneous codesign space in CPSs. Techniques such as approximation algorithms, greedy heuristics, parallel

randomized search, and learning-based methods are promising directions for tackling complexity. Furthermore, methods that can quickly estimate the bounds of design metrics for partial solutions (i.e., when only part of the design variables are decided) should facilitate pruning the design space and accelerate the exploration process.

- Interactive synthesis and verification interface: CPS design is often an iterative process, during which the system specifications and constraints are continuously refined and the tradeoffs among different metrics are constantly carried out. It is therefore important to develop an interactive interface between designers and design tools (e.g., synthesis and verification tools), to enable flexible additions and updates of specifications/constraints, and to support designer queries such as "is it possible to further improve metric A?" and "what if constraint X is removed or relaxed by quantity Y?" This interface will facilitate a nimble design process, better leverage designers' expertise, and ultimately improve design quality and productivity. To realize such interface, new synthesis and verification tools are needed to assess system feasibility (with respect to both functional and nonfunctional requirements) under an incomplete set of constraints, estimate the bounds of design metrics for all feasible implementations (if there are any), identify design bottlenecks when there is no feasible solution or certain metrics are not good enough, and leverage previous synthesis/verification results (rather than restarting from scratch) when new constraints are added or existing ones are modified.
- Runtime coadaptation: While this paper mostly focuses on tools and methods for design-time modeling, synthesis, and verification of CPSs, it is equally important to investigate and leverage the interdependencies between components for runtime adaptation of CPSs, especially for those systems that operate in uncertain environment and with changing missions. For example, when a robot faces adversarial environment, it could holistically adapt its operation across system layers by adopting more robust sensing and control algorithms, applying stronger security mechanisms in software and hardware, and possibly reducing nonessential tasks to meet resource constraints. When an intelligent building detects emergency situations (e.g., fire or breakout), it could adapt to a different operation mode, with changes across HVAC and lighting control algorithms, computation and communication infrastructures, and sensing platforms. Such level of coadaptation will require new synthesis and verification methods that are efficient and reliable enough for runtime usage. For instance, fast online algorithms could be developed with the help from offline synthesis/verification that consider different operating scenarios.

REFERENCES

[1] K. Keutzer, S. Malik, A. R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System-level design: orthogonalization of concerns and platform-based design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 12, pp. 1523–1543, Dec. 2000.

[2] L. Carloni, F. D. Bernardinis, C. Pinello, A. Sangiovanni-Vincentelli, and M. Sgroi, "Platform-based design for embedded systems," in *The Embedded Systems Handbook*. Boca Raton, FL, USA: CRC Press, 2005.

[3] 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.

[4] F. Balarin *et al.*, "Platform-based design and frameworks: Metropolis and metro II," in *Model-Based Design of Heterogeneous Embedded Systems*, G. Nicolescu and P. Mosterman, Eds. Boca Raton, FL, USA: CRC Press, 2009.

[5] F. Balarin, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, Y. Watanabe, and G. Yang, "Concurrent execution semantics and sequential simulation algorithms for the metropolis meta-model," in *Proc. 10th Int. Symp. Hardw./Softw. Codesign (CODES)*, May 2002, pp. 13–18.

[6] G. Yang, A. Sangiovanni-Vincentelli, Y. Watanabe, and F. Balarin, "Separation of concerns: Overhead in modeling and efficient simulation techniques," in *Proc. 4th ACM Int. Conf. Embedded Softw. (EMSOFT)*, New York, NY, USA, 2004, pp. 44–53. [Online]. Available: http://doi.acm.org/10.1145/1017753.1017765

[7] A. Davare *et al.*, "A next-generation design framework for platform-based design," in *Proc. Design Verification Conf. (DVCON)*, Feb. 2007.

[8] A. Davare *et al.*, "METROII: A design environment for cyber-physical systems," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 1, pp. 49-1–49-31, Mar. 2013. [Online]. Available: http://doi.acm.org/10.1145/2435227.2435245

[9] A. Davare, Q. Zhu, J. Moondanos, and A. Sangiovanni-Vincentelli, "JPEG encoding on the Intel MXP5800: A platform-based design case study," in *Proc. 3rd Workshop Embedded Syst. Real-Time Multimedia*, Sep. 2005, pp. 89–94.

[10] D. Densmore, A. Simalatsar, A. Davare, R. Passerone, and A. Sangiovanni-Vincentelli, "UMTS MPSoC design evaluation using a system level design framework," in *Proc. Design Autom. Test Eur. (DATE)*, Mar. 2009, pp. 478–483. [Online]. Available: http://www.gigascale.org/pubs/1939.html

[11] H. Zeng, A. Davare, A. Sangiovanni-Vincentelli, S. Sonalkar, S. Kanajan, and C. Pinello, "Design space exploration of automotive platforms in metropolis," SAE Tech. Paper 2006-01-1468, 2006.

[12] Y. Yang, Q. Zhu, M. Maasoumy, and A. Sangiovanni-Vincentelli, "Development of building automation and control systems," *IEEE Design Test Comput.*, vol. 29, no. 4, pp. 45–55, Aug. 2012.

[13] L. Guo, Q. Zhu, P. Nuzzo, R. Passerone, A. Sangiovanni-Vincentelli, and E. A. Lee, "Metronomy: A function-architecture co-simulation framework for timing verification of cyber-physical systems," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Oct. 2014, pp. 1–10.

[14] Y. Yang, A. Pinto, A. Sangiovanni-Vincentelli, and Q. Zhu, "A design flow for building automation and control systems," in *Proc. 31st IEEE Int. Real-Time Syst. Symp. (RTSS)*, 2010, pp. 105–116.

[15] *Simulink*. [Online]. Available: http://www.mathworks.com

[16] *Modelica*. [Online]. Available: http://www.modelica.org

[17] J. Eker *et al.*, "Taming heterogeneity—The Ptolemy approach," *Proc. IEEE*, vol. 91, no. 1, pp. 127–144, Jan. 2003.

[18] P. Derler, E. A. Lee, and A. S. Vincentelli, "Modeling cyber-physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 13–28, Jan. 2012.

[19] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin, *Aspect-Oriented Programming*. Berlin, Germany: Springer-Verlag, 1997, pp. 220–242. [Online]. Available: https://doi.org/10.1007/BFb0053381

[20] I. Akkaya, P. Derler, S. Emoto, and E. A. Lee, "Systems engineering for industrial cyber-physical systems using aspects," *Proc. IEEE*, vol. 104, no. 5, pp. 997–1012, May 2016.

[21] Y. Zhao, J. Liu, and E. A. Lee, "A programming model for time-synchronized distributed real-time systems," in *Proc. 13th IEEE Real Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2007, pp. 259–268.

[22] J. Zou, S. Matic, E. A. Lee, T. H. Feng, and P. Derler, "Execution strategies for ptides, a programming model for distributed embedded systems," in *Proc. 15th IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2009, pp. 77–86.

[23] J. Zou, S. Matic, and E. A. Lee, "PtidyOS: A lightweight microkernel for ptides real-time systems," in *Proc. IEEE 18th Real Time Embedded Technol. Appl. Symp.*, Apr. 2012, pp. 209–218.

[24] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-based design for cyber-physical systems," *Eur. J. Control*, vol. 18, no. 3, pp. 217–238, 2012.

[25] J. Sztipanovits and G. Karsai, "Model-integrated computing," *Computer*, vol. 30, no. 4, pp. 110–111, Apr. 1997.

[26] A. Ledeczi *et al.*, "Composing domain-specific design environments," *Computer*, vol. 34, no. 11, pp. 44–51, Nov. 2001.

[27] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty, "Model-integrated development of embedded software," *Proc. IEEE*, vol. 91, no. 1, pp. 145–164, Jan. 2003.
[28] *Model Integrated Computing*. [Online]. Available: http://www.isis.vanderbilt.edu/research/MIC
[29] J. Sztipanovits, T. Bapty, X. Koutsoukos, Z. Lattmann, S. Neema, and E. Jackson, "Model and tool integration platforms for cyber-physical system design," *Proc. IEEE*, to be published.
[30] A. Basu *et al.*, "Rigorous component-based system design using the BIP framework," *IEEE Softw.*, vol. 28, no. 3, pp. 41–48, May/Jun. 2011.
[31] A. Basu, M. Bozga, and J. Sifakis, "Modeling heterogeneous real-time components in BIP," in *Proc. 4th IEEE Int. Conf. Softw. Eng. Formal Methods (SEFM)*, Washington, DC, USA, Sep. 2006, pp. 3–12.
[32] S. Bliudze and J. Sifakis, "The algebra of connectors—Structuring interaction in BIP," in *Proc. 7th ACM IEEE Int. Conf. Embedded Softw. (EMSOFT)*, Salzburg, Austria, Sep./Oct. 2007, pp. 11–20.
[33] T. Stefanov, C. Zissulescu, A. Turjan, B. Kienhuis, and E. Deprettere, "System design using kahn process networks: The compaan/Laura approach," in *Proc. Conf. Design Autom. Test Eur.*, 2004, p. 10340.
[34] P. Lieverse, T. Stefanov, P. van der Wolf, and E. Deprettere, "System level design with Spade: An M-JPEG case study," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2001, pp. 31–38.
[35] C. Erbas, S. C. Erbas, and A. D. Pimentel, "A multiobjective optimization model for exploring multiprocessor mappings of process networks," in *Proc. 1st IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, Oct. 2003, pp. 182–187.
[36] E. A. de Kock, "Multiprocessor mapping of process networks: A JPEG decoding case study," in *Proc. 15th Int. Symp. Syst. Synth.*, 2002, pp. 68–73.
[37] T. Raudvere, I. Sander, A. K. Singh, and A. Jantsch, "Verification of design decisions in ForSyDe," in *Proc. 1st IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, Oct. 2003, pp. 176–181.
[38] J. Paul and D. Thomas, "A layered, codesign virtual machine approach to modeling computer systems," in *Proc. Conf. Design Autom. Test Eur.*, 2002, p. 522.
[39] K. K. A. Mihal, *Mapping Concurrent Applications Onto Architectural Platforms*, H. T. A. Jantsch, Ed. Boston, MA, USA: Kluwer, 2003, pp. 39–59.
[40] A. Bakshi, V. Prasanna, and A. Ledeczi, "MILAN: A model based integrated simulation framework for design of embedded systems," in *Proc. Workshop Lang., Compil., Tools Embedded Syst.*, Jun. 2001.
[41] T. A. Henzinger, B. Horowitz, and C. M. Kirsch, "Giotto: A time-triggered language for embedded programming," *Proc. IEEE*, vol. 91, no. 1, pp. 84–99, Jan. 2003.
[42] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee, "Modular Specification of Hybrid Systems in CHARON," in *Proc. 3rd Int. Workshop Hybrid Syst. Comput. Control (HSCC)*. London, U.K.: Springer-Verlag, 2000, pp. 6–19. [Online]. Available: http://dl.acm.org/citation.cfm?id=646880.759760
[43] I. Shin and I. Lee, "Compositional real-time scheduling framework," in *Proc. 25th IEEE Int. Real-Time Syst. Symp.*, Dec. 2004, pp. 57–67.
[44] I. Shin and I. Lee, "Compositional real-time scheduling framework with periodic model," *ACM Trans. Embedded Comput. Syst.*, vol. 7, no. 3, pp. 30:1–30:39, Apr. 2008.
[45] A. Easwaran, M. Anand, and I. Lee, "Compositional analysis framework using EDP resource models," in *Proc. 28th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Dec. 2007, pp. 129–138.
[46] CARTS *(Compositional analysis of Real-Time Systems)*. [Online]. Available: http://rtg.cis.upenn.edu/carts/index.php
[47] J. Lee *et al.*, "Realizing compositional scheduling through virtualization," in *Proc. IEEE 18th Real Time Embedded Technol. Appl. Symp.*, Apr. 2012, pp. 13–22.
[48] S. Xi, J. Wilson, C. Lu, and C. Gill, "RT-Xen: Towards real-time hypervisor scheduling in Xen," in *Proc. Int. Conf. Embedded Softw. (EMSOFT)*, Oct. 2011, pp. 39–48.
[49] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The digraph real-time task model," in *Proc. 17th IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2011, pp. 71–80.
[50] *SCADE*. [Online]. Available: http://www.esterel-technologies.com/products/scade-suite/
[51] J. C. Jensen, D. H. Chang, and E. A. Lee, "A model-based design methodology for cyber-physical systems," in *Proc. 7th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jul. 2011, pp. 1666–1671.
[52] D. De Niz, G. Bhatia, and R. Rajkumar, "Model-based development of embedded systems: The sysweaver approach," in *Proc. 12th IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2006, pp. 231–242.
[53] *General Motors Developed Two-Mode Hybrid Powertrain With MathWorks Model-Based Design; Cut 24 Months Off Expected Dev Time*. [Online]. Available: http://www.greencarcongress.com
[54] *Automakers Opting for Model-Based Design*. [Online]. Available: http://www.designnews.com
[55] A. Ledeczi *et al.*, "The generic modeling environment," in *Proc. Workshop Intell. Signal Process.*, vol. 17. Budapest, Hungary, 2001, p. 1.
[56] J. Davis, "GME: The generic modeling environment," in *Proc. Companion 18th Annu. ACM SIGPLAN Conf. Object-Oriented Program., Syst., Lang., Appl. (OOPSLA)*, New York, NY, USA, 2003, pp. 82–83. [Online]. Available: http://doi.acm.org/10.1145/949344.949360
[57] E. Magyari *et al.*, "UDM: An infrastructure for implementing domain-specific modeling languages," in *Proc. 3rd OOPSLA Workshop Domain-Specific Modeling (OOPSLA)*, Anahiem, CA, USA, Oct. 2003.
[58] D. Balasubramanian, A. Narayanan, C. van Buskirk, and G. Karsai, "The graph rewriting and transformation language: GReAT," in *Proc. Electron. Commun. (EASST)*, vol. 1, 2006.
[59] S. Neema, J. Sztipanovits, and G. Karsai, "Constraint-based design-space exploration and model synthesis," in *Embedded Software* (Lecture Notes in Computer Science), vol. 2855. Springer, 2003, pp. 290–305.
[60] H. Neema *et al.*, "Design space exploration and manipulation for cyber physical systems," in *Proc. 1st Int. Workshop Design Space Explor. Cyber-Phys. Syst. (IDEAL)*. Berlin, Germany: Springer-Verlag, 2014.
[61] Z. Lattmann *et al.*, "Verification and design exploration through meta tool integration with openmodelica," in *Proc. 10 th Int. Modelica Conf.* Lund, Sweden: Linköping Univ. Electronic Press, 2014, pp. 353–362.
[62] Z. Lattmann *et al.*, "Towards automated evaluation of vehicle dynamics in system-level designs," in *Proc. ASME Int. Design Eng. Tech. Conf. Comput. Inf. Eng. Conf. (IDETC/CIE)*, Chicago, IL, USA, 2012, pp. 1131–1141.
[63] *FORMULA 2.0: A Language for Formal Specifications*. Berlin, Germany: Springer-Verlag, 2013. [Online]. Available: https://www.microsoft.com/en-us/research/publication/formula-2-0-langua%ge-formal-specifications/
[64] A. Lekidis, E. Stachtiari, P. Katsaros, M. Bozga, and C. K. Georgiadis, "Using BIP to reinforce correctness of resource-constrained IoT applications," in *Proc. 10th IEEE Int. Symp. Ind. Embedded Syst., (SIES)*. Siegen, Germany, Jun. 2015, pp. 245–253.
[65] A. Lekidis, P. Bourgos, S. Djoko-Djoko, M. Bozga, and S. Bensalem, "Building distributed sensor network applications using BIP," in *Proc. IEEE Sensors Appl. Symp. (SAS)*, Apr. 2015, pp. 1–6.
[66] A. Mavridou, J. Sifakis, and J. Sztipanovits, "DesignBIP: A design studio for modeling and generating systems with BIP," ArXiv e-prints, Tech. Rep., May 2018.
[67] S. Bliudze, A. Mavridou, R. Szymanek, and A. Zolotukhina, "Exogenous coordination of concurrent software components with JavaBIP," *Softw. Pract. Exper.*, vol. 47, no. 11, pp. 1801–1836, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2495
[68] *Connected Car, Automotive Value Chain Unbound*, McKinsey&Company, New York, NY, USA, Sep. 2014.
[69] R. N. Charette, "This car runs on code," *IEEE Spectrum*, to be published.
[70] A. Sangiovanni-Vincentelli, "Quo vadis, SLD? Reasoning about the trends and challenges of system level design," *Proc. IEEE*, vol. 95, no. 3, pp. 467–506, Mar. 2007.
[71] *The Road to 2020 and Beyond: What's Driving the Global Automotive Industry?* McKinsey&Company, New York, NY, USA, Aug. 2013.
[72] *Consolidation in Vehicle Electronic Architectures*, Roland Berger Strategy Consultants, Munich, Germany, Jul. 2015.
[73] M. Di Natale and A. Sangiovanni-Vincentelli, "Moving from federated to integrated architectures in automotive: The role of standards, methods and tools," *Proc. IEEE*, vol. 98, no. 4, pp. 603–620, Apr. 2010.
[74] S. A. Seshia, S. Hu, W. Li, and Q. Zhu, "Design automation of cyber-physical systems: Challenges, advances, and opportunities," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 9, pp. 1421–1434, Sep. 2017.
[75] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *Proc. Real-Time Syst. Symp.*, Dec. 2008, pp. 291–300.
[76] K. J. Aström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Upper Saddle River, NJ, USA: Prentice-Hall, 2007.
[77] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, "On task schedulability in real-time control systems," in *Proc. 17th IEEE Real-Time Syst. Symp.*, Dec. 1996, pp. 13–21.
[78] 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. Exhib. (DATE)*, Apr. 2009, pp. 57–62.
[79] S. Samii, P. Eles, Z. Peng, P. Tabuada, and A. Cervin, "Dynamic scheduling and control-quality optimization of self-triggered control applications," in *Proc. IEEE 31st Real-Time Syst. Symp. (RTSS)*, Dec. 2010, pp. 95–104.
[80] D. Goswami, M. Lukasiewycz, R. Schneider, and S. Chakraborty, "Time-triggered implementations of mixed-criticality automotive software," in *Proc. Design Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2012, pp. 1227–1232.
[81] P. Deng, Q. Zhu, A. Davare, A. Mourikis, X. Liu, and M. D. Natale, "An efficient control-driven period optimization algorithm for distributed real-time systems," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3552–3566, Dec. 2016.
[82] M. Maasoumy, Q. Zhu, C. Li, F. Meggers, and A. Vincentelli, "Co-design of control algorithm and embedded platform for building HVAC systems," in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2013, pp. 61–70.
[83] T. Wei, Q. Zhu, and M. Maasoumy, "Co-scheduling of HVAC control, EV charging and battery usage for building energy efficiency," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2014, pp. 191–196.
[84] T. Wei *et al.*, "Battery management and application for energy-efficient buildings," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–6.
[85] T. Wei, M. A. Islam, S. Ren, and Q. Zhu, "Co-scheduling of datacenter and HVAC loads in mixed-use buildings," in *Proc. 7th IEEE Int. Green Sustain. Comput. Conf.*, Nov. 2016, pp. 1–8.
[86] 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.
[87] A. Benveniste *et al.*, "Contracts for system design," Res. Rep. RR-8147, Nov. 2012. [Online].

Available: https://hal.inria.fr/hal-00757488

[88] P. Derler, E. A. Lee, S. Tripakis, and M. Törngren, "Cyber-physical system design contracts," in *Proc. ACM/IEEE 4th Int. Conf. Cyber-Phys. Syst.*, Apr. 2013, pp. 109–118.

[89] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis, "Multiple viewpoint contract-based specification," in *Formal Methods for Components and Objects*, F. S. Boer, M. M. Bonsangue, S. Graf, and W.-P. Roever, Eds. Berlin, Germany: Springer-Verlag, 2008, pp. 200–225. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-92188-2_9

[90] P. Nuzzo *et al.*, "A contract-based methodology for aircraft electric power system design," *IEEE Access*, vol. 2, pp. 1–25, 2014.

[91] P. Nuzzo, J. Finn, A. Iannopollo, and A. Sangiovanni-Vincentelli, "Contract-based design of control protocols for safety-critical cyber-physical systems," in *Proc. Design Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–4.

[92] A. Pnueli, "The temporal logic of programs," in *Proc. 18th Annu. Symp. Found. Comput. Sci.*, Oct. 1977, pp. 46–57.

[93] F. Jahanian and A. Mok, "Safety analysis of timing properties in real-time systems," *IEEE Trans. Softw. Eng.*, vol. SE-12, no. 9, pp. 890–904, Sep. 1986.

[94] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[95] X. Chen, H. Hsieh, F. Balarin, and Y. Watanabe, "Logic of constraints: A quantitative performance and functional constraint formalism," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 8, pp. 1243–1255, Aug. 2004.

[96] P. Deng, F. Cremona, Q. Zhu, M. D. Natale, and H. Zeng, "A model-based synthesis flow for automotive CPS," in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2015, pp. 198–207.

[97] G. Bernat, A. Burns, and A. Liamosi, "Weakly hard real-time systems," *IEEE Trans. Comput.*, vol. 50, no. 4, pp. 308–321, Apr. 2001.

[98] Q. Zhu, Y. Yang, E. Scholte, M. D. Natale, and A. Sangiovanni-Vincentelli, "Optimizing extensibility in hard real-time distributed systems," in *Proc. 15th IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2009, pp. 275–284.

[99] Q. Zhu, Y. Yang, M. D. Natale, E. Scholte, and A. Sangiovanni-Vincentelli, "Optimizing the software architecture for extensibility in hard real-time distributed systems," *IEEE Trans. Ind. Informat.*, vol. 6, no. 4, pp. 621–636, Nov. 2010.

[100] Q. Zhu, P. Deng, M. Di Natale, and H. Zeng, "Robust and extensible task implementations of synchronous finite state machines," in *Proc. Design Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2013, pp. 1319–1324.

[101] Y. Gao, S. K. Gupta, and M. A. Breuer, "Using explicit output comparisons for fault tolerant scheduling (FTS) on modern high-performance processors," in *Proc. DATE*, 2013, pp. 927–932.

[102] H. Liang, Z. Wang, B. Zheng, and Q. Zhu, "Addressing extensibility and fault tolerance in can-based automotive systems," in *Proc. IEEE/ACM Int. Symp. Netw.-Chip (NOCS)*, Nov. 2017, p. 10.

[103] W. Zheng, Q. Zhu, M. D. Natale, and A. Sangiovanni-Vincentelli, "Definition of task allocation and priority assignment in hard real-time distributed systems," in *Proc. 28th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Dec. 2007, pp. 161–170.

[104] Q. Zhu, H. Zeng, W. Zheng, M. D. Natale, and A. Sangiovanni-Vincentelli, "Optimization of task allocation and priority assignment in Gard real-time distributed systems," *ACM Trans. Embed. Comput. Syst.*, vol. 11, no. 4, pp. 85:1–85:30, 2012.

[105] C. Lin, Q. Zhu, C. Phung, and A. Sangiovanni-Vincentelli, "Security-aware mapping for CAN-based real-time distributed automotive systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2013, pp. 115–121.

[106] C. W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-aware mapping for TDMA-based real-time distributed systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2014, pp. 24–31.

[107] A. Davare, Q. Zhu, M. D. Natale, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli, "Period optimization for hard real-time distributed automotive systems," in *Proc. Design Autom. Conf. (DAC)*, Jun. 2007, pp. 278–283.

[108] P. Deng, Q. Zhu, M. Di Natale, and H. Zeng, "Task synthesis for latency-sensitive synchronous block diagram," in *Proc. 9th IEEE Int. Symp. Ind. Embedded Syst. (SIES)*, Jun. 2014, pp. 112–121.

[109] "Time-triggered Ethernet," SAE Tech. Paper AS6802, 2011.

[110] M. Lukasiewycz, M. Glass, P. Milbredt, and J. Teich, "Flexray schedule optimization of the static segment," in *Proc. CODES+ISSS Conf.*, Jun. 2009, pp. 363–372.

[111] M. D. J. Teener *et al.*, "Heterogeneous networks for audio and video: Using IEEE 802.1 audio video bridging," *Proc. IEEE*, vol. 101, no. 11, pp. 2339–2354, Nov. 2013.

[112] C. Lin, B. Zheng, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-aware design methodology and optimization for automotive systems," *ACM Trans. Design Autom. Electron. Syst.*, vol. 21, no. 1, pp. 18-1–18-26, Dec. 2015. [Online]. Available: http://doi.acm.org/10.1145/2803174

[113] *AUTOSAR*. [Online]. Available: http://www.autosar.org

[114] Y. Ma, D. Gunatilaka, B. Li, H. Gonzalez, and C. Lu, "Holistic cyber-physical management for dependable wireless control systems," *ACM Trans. Cyber-Phys. Syst.*, 2018.

[115] K. Zhang, J. Sprinkle, and R. G. Sanfelice, "Computationally aware switching criteria for hybrid model predictive control of cyber-physical systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 479–490, Apr. 2016.

[116] K. Zhang, J. Sprinkle, and R. G. Sanfelice, "A hybrid model predictive controller for path planning and path following," in *Proc. ACM/IEEE 6th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, New York, NY, USA, 2015, pp. 139–148. [Online]. Available: http://doi.acm.org/10.1145/2735960.2735966

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

[118] D. Soudbakhsh, L. T. X. Phan, O. Sokolsky, I. Lee, and A. Annaswamy, "Co-design of control and platform with dropped signals," in *Proc. ACM/IEEE 4th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, New York, NY, USA, 2013, pp. 129–140. [Online]. Available: http://doi.acm.org/10.1145/2502524.2502542

[119] V. Gupta, D. Spanos, B. Hassibi, and R. M. Murray, "Optimal LQG control across packet-dropping links," in *Syst. Control Lett.*, pp. 360–365, 2005.

[120] J. Nilsson. (1998). *Real-Time Control Systems with Delays*. [Online].Available: http://theses.lub.lu.se/postgrad/search.tkl?field_query1=pubid&query1=tec_163&recordformat=display

[121] 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.

[122] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S. S. Sastry, "Optimal control with unreliable communication: The TCP case," in *Proc. Amer. Control Conf.*, vol. 5, Jun. 2005, pp. 3354–3359.

[123] D. Nesic and D. Liberzon, "A unified framework for design and analysis of networked and quantized control systems," *IEEE Trans. Autom. Control*, vol. 54, no. 4, pp. 732–747, Apr. 2009.

[124] D. Nešić and A. R. Teel, "Input-output stability properties of networked control systems," *IEEE Trans. Autom. Control*, vol. 49, no. 10, pp. 1650–1667, Oct. 2004.

[125] G. C. Walsh, H. Ye, and L. G. Bushnell, "Stability analysis of networked control systems," *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 3, pp. 438–446, May 2002.

[126] G. Weiss and R. Alur, "Automata based interfaces for control and scheduling," in *Proc. 10th Int. Conf. Hybrid Syst., Comput. Control (HSCC)*. Berlin, Germany: Springer-Verlag, 2007, pp. 601–613. [Online]. Available: http://dl.acm.org/citation.cfm?id=1760804.1760852

[127] J. L. Ny and G. J. Pappas, "Robustness analysis for the certification of digital controller implementations," in *Proc. 1st ACM/IEEE Int. Conf. Cyber-Phys. Syst. (ICCPS)*, New York, NY, USA, 2010, pp. 99–108, doi: doic 10.1145/1795194.1795209.

[128] R. Majumdar, E. Render, and P. Tabuada, "Robust discrete synthesis against unspecified disturbances," in *Proc. 14th Int. Conf. Hybrid Syst., Comput. Control (HSCC)*, New York, NY, USA, 2011, pp. 211–220, doi: doic 10.1145/1967701.1967732.

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

[130] A. Anta and P. Tabuada, "On the benefits of relaxing the periodicity assumption for networked control systems over CAN," in *Proc. 30th IEEE Real-Time Syst. Symp. (RTSS)*, Washington, DC, USA, 2009, pp. 3–12, doi: doic 10.1109/RTSS.2009.39.

[131] A. Anta and P. Tabuada, "To sample or not to sample: Self-triggered control for nonlinear systems," *IEEE Trans. Autom. Control*, vol. 55, no. 9, pp. 2030–2042, Sep. 2010.

[132] D. Roy, L. Zhang, W. Chang, S. K. Mitter, and S. Chakraborty, "Semantics-preserving cosynthesis of cyber-physical systems," *Proc. IEEE*, vol. 106, no. 1, pp. 171–200, Jan. 2018.

[133] B. Zheng, W. Li, P. Deng, L. Gérardy, Q. Zhu, and N. Shankar, "Design and verification for transportation system security," in *Proc. 52nd ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[134] B. Zheng, C.-W. Lin, H. Yu, H. Liang, and Q. Zhu, "CONVINCE: A cross-layer modeling, exploration and validation framework for next-generation connected vehicles," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2016, pp. 1–8, doi: doic 10.1145/2966986.2980078.

[135] B. Zheng, C.-W. Lin, H. Liang, S. Shiraishi, W. Li, and Q. Zhu, "Delay-aware design, analysis and verification of intelligent intersection management," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, May 2017, pp. 1–8.

[136] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *J. Artif. Intell. Res.*, vol. 31, pp. 591–656, Mar. 2008.

[137] Q. Jin, G. Wu, K. Boriboonsomsin, and M. Barth, "Advanced intersection management for connected vehicles using a multi-agent systems approach," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2012, pp. 932–937.

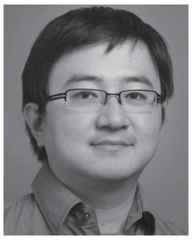
[138] R. Azimi, G. Bhatia, R. R. Rajkumar, and P. Mudalige, "STIP: Spatio-temporal intersection protocols for autonomous vehicles," in *Proc. ACM/IEEE 5th Int. Conf. Cyber-Phys. Syst. (CPS Week) ICCPS*, Apr. 2014, pp. 1–12.

[139] S. Azimi, G. Bhatia, R. R. Rajkumar, and P. Mudalige, "Reliable intersection protocols using vehicular networks," in *Proc. ACM/IEEE 4th Int. Conf. Cyber-Phys. Syst.*, 2013, pp. 1–10.

[140] F. Zhu and S. V. Ukkusuri, "A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment," *Transp. Res. C, Emerg. Technol.*, vol. 55, pp. 363–378, Jun. 2015.

[141] Y. P. Fallah and M. K. Khandani, "Analysis of the coupling of communication network and safety application in cooperative collision warning systems," in *Proc. ACM/IEEE 6th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, New York, NY, USA, 2015, pp. 228–237, doi: doic

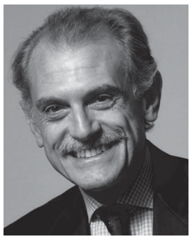
10.1145/2735960.2735975.
[142] Y. Yao, L. Rao, X. Liu, and X. Zhou, "Delay analysis and study of IEEE 802.11p based DSRC safety communication in a highway environment," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1591–1599.
[143] S. Bastani, B. Landfeldt, and L. Libman, "On the reliability of safety message broadcast in urban vehicular ad hoc networks," in *Proc. 14th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst. (MSWiM)*, New York, NY, USA, 2011, pp. 307–316, doi: doic 10.1145/2068897.2068951.
[144] Y. O. Basciftci, F. Chen, J. Weston, R. Burton, and C. E. Koksal, "How vulnerable is vehicular communication to physical layer jamming attacks?" in *Proc. IEEE 82nd Veh. Technol. Conf. (VTC Fall)*, Sep. 2015, pp. 1–5.
[145] S. Kamil, A. Cheung, S. Itzhaky, and A. Solar-Lezama, "Verified lifting of stencil computations," in *Proc. 37th ACM SIGPLAN Conf. Program. Lang. Design Implement. (PLDI)*, New York, NY, USA, 2016, pp. 711–726, doi: doic 10.1145/2908080.2908117.
[146] M. Maasoumy, A. Pinto, and A. Sangiovanni-Vincentelli, "Model-based hierarchical optimal control design for hvac systems," in *Proc. Dyn. Syst. Control Conf.*, 2011, pp. 271–278.
[147] *EnergyPlus*. [Online]. Available: https://energyplus.net/
[148] K. Deng, P. Barooah, P. G. Mehta, and S. P. Meyn, "Building thermal model reduction via aggregation of states," in *Proc. Amer. Control Conf.*, Jun. 2010, pp. 5118–5123.
[149] D. Kim and J. E Braun, "Reduced-order building modeling for application to model-based predictive control," in *Proc. Simuild 5th Nat. Conf. (IBPSA-USA)*, Madison, WI, USA, 2012.
[150] T. Wei, Y. Wang, and Q. Zhu, "Deep reinforcement learning for building HVAC control," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2017, pp. 1–6.
[151] Y. Heo and V. M. Zavala, "Gaussian process modeling for measurement and verification of building energy savings," *Energy Buildings*, vol. 53, pp. 7–18, Oct. 2012.
[152] J. Wall, Y. Guo, J. Li, and S. West, "A dynamic machine learning-based technique for automated fault detection in HVAC systems," *ASHRAE Trans.*, vol. 117, pp. 449–456, 2011.
[153] H.-X. Zhao and F. Magouls, "A review on the prediction of building energy consumption," *Renew. Sustain. Energy Rev.*, vol. 16, no. 6, pp. 3586–3592, 2012.
[154] A. H. Neto and F. A. S. Fiorelli, "Comparison between detailed model simulation and artificial neural network for forecasting building energy consumption," *Energy Buildings*, vol. 40, no. 12, pp. 2169–2176, 2008.
[155] B. Dong, C. Cao, and S. Lee, "Applying support vector machines to predict building energy consumption in tropical region," *Energy Buildings*, vol. 37, no. 5, pp. 545–553, 2005.
[156] S. Lan, R. Panda, Q. Zhu, and A. K. Roy-Chowdhury, "FFNet: Video fast-forwarding via reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6771–6780.



Qi Zhu (Member, IEEE) received the B.E. degree in computer science from Tsinghua University, Beijing, China, in 2003 and the Ph.D. degree in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 2008.

Currently, he is an Associate Professor at the Electrical Engineering and Computer Science Department, Northwestern University, Evanston, IL, USA. Prior to joining Northwestern, he was an Assistant Professor and later Associate Professor at the University of California, Riverside, Riverside, CA, USA and a Research Scientist in Intel. His research interests include model-based design and software synthesis of cyber–physical systems (CPSs), CPS security, embedded and real-time systems, and system-on-chip design.

Dr. Zhu received four best paper awards at the Design Automation Conference (DAC), the International Conference on Cyber–Physical Systems (ICCPS), and the *ACM Transactions on Design Automation of Electronic Systems*; the National Science Foundation CAREER award; and the IEEE Technical Committee on Cyber–Physical Systems (TC-CPS) Early-Career Award. He is an Associate Editor of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, and has served on the program committees for a number of conferences in design automation, CPSs, embedded systems, and real-time systems.



Alberto Sangiovanni-Vincentelli (Fellow, IEEE) is the Edgar L. and Harold H. Buttner Chair at the Electrical Engineering and Computer Science Department, University of California at Berkeley, Berkeley, CA, USA, where he has been a member of the faculty since 1976. He helped founding Cadence and Synopsys, the two leading companies in EDA. He is on the Board of Directors of Cadence, KPIT Technologies, Sonics, Expert Systems, and Cogisen. He is a member of the Investment Committee of Atlante Venture, the advisory board of Walden International and Xseed, and of the Executive Committee of the Italian Institute of Technology. He was the President of the Strategic Committee of the Italian Strategic Fund. He consulted for companies such as Intel, HP, Bell Labs, IBM, Samsung, UTC, Lutron, Camozzi Group, Kawasaki Steel, Fujitsu, Telecom Italia, Pirelli, GM, BMW, Mercedes, Magneti Marelli, ST Microelectronics, ELT, Unipol and UniCredit. He authored over 950 papers, 17 books, and two patents.

Dr. Sangiovanni-Vincentelli is a Fellow of the Association for Computing Machinery (ACM), a member of the National Academy of Engineering, and holds two honorary Doctorates from Aalborg University (Denmark) and KTH (Sweden). He earned the IEEE/RSE Maxwell Award for groundbreaking contributions that have had an exceptional impact on the development of electronics and electrical engineering, the Kaufmann Award for seminal contributions to EDA, the EDAA lifetime Achievement Award, the IEEE/ACM R. Newton Impact Award, the University of California Distinguished Teaching Award, the IEEE Technical Committee on Cyber–Physical Systems (TC-CPS) Technical Achievement Award for pioneering contributions and leadership in cyber–physical systems and design automation, the International Symposium on Physical Design (ISPD) lifetime achievement award, intended for individuals who have made outstanding contributions to the field of physical design automation over multiple decades and the IEEE Graduate Teaching Award for inspirational teaching of graduate students.