A Co-Simulation Approach for Control Performance Analysis during Design Space Exploration of Cyber-Physical Systems

Nina Mühleis, Michael Glaß, Liyuan Zhang, and Jürgen Teich
Hardware/Software Co-Design, University of Erlangen-Nuremberg, Germany
{nina.muehleis,glass,teich}@cs.fau.de

ABSTRACT

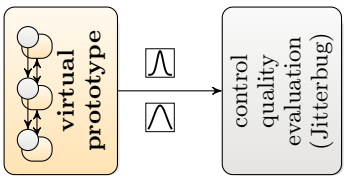
In modern embedded systems, more and more control applications are executed in a distributed fashion. Here, the architecture as well as the scheduling policies influence the control performance. Thus, considering control performance as design objective at Electronic System Level becomes mandatory. This work presents a control performance analysis approach based on the co-simulation of high level models of plants and a virtual prototype of the controllers in the system. The work in hand integrates this co-simulation in a *Design Space Exploration*, resulting in a fully automatic toolflow at the ESL that accounts for several control performance metrics as additional principle design objectives together with other design objectives such as monetary cost or energy consumption.

1. INTRODUCTION AND RELATED WORK

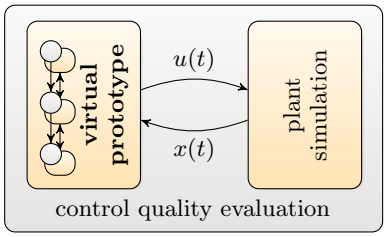
Today's distributed embedded systems as found, e. g., in the automobile and avionics domain, become more and more complex. Moreover, a growing number of control applications is integrated as a part of such systems. Hence, considering control quality during system design becomes mandatory.

Typically, a control engineer develops high-level models to design, analyze and simulate control applications, using, e. g., MATLAB Simulink. Here, a model of the controller as well as the plant exists. As soon as the modeling process is completed, the system-level designer gathers a hardware/-software model of the control application from the high-level control model. Here, the derived application consists of several periodic tasks: sampling the actual state of the system, computing the control law, and interacting with the actuator. As the high-level model is independent of the architecture, the system-level designer has to (a) allocate hardware resources, (b) bind the control application tasks to the allocated resources, (c) route messages for the data-transfer, and (d) determine a schedule for both the tasks and the send messages. In recent years, approaches for an automated *Design Space Exploration* (DSE) at the *Electronic System Level* (ESL) have been developed that investigate the design space and obtain implementations that are optimized with respect to several often conflicting *design objectives* like monetary costs, power consumption or quality of control should while fulfilling different *design constraints* like the stability of the control application.

The applicability of control applications in distributed embedded systems depends on the expected quality of the control. A key factor are end-to-end latencies of the control applications caused by contention on computation and communication resources that are shared between several applications. It is well known that not only the maximum or average delay contributes to the controller performance, but also the distribution of the delays [1]. Recent works in this domain like [2, 3] take these varying delays into account during control performance optimization by varying the periods and priorities of the tasks. However, these approaches consider a fixed architecture and task mapping and are not applicable to the problem targeted in this work. A recent approach that introduces control quality as a principal design objective into DSE at the ESL is proposed in [4] and outlined in Fig. 1(a). There, for each control application, the distributions of the delays are determined by means of a virtual prototype of the system. In a second step, each control application is investigated separately employing the Jitterbug toolbox [5]. Such an approach has two serious drawbacks: (a) The analysis is restricted to a single design objective, i. e., the *quadratic cost*. (b) The distributions of the delays in each control application are assumed to be independent. In particular, (b) does not hold in a real system. Let two tasks be executed on the same resource. Moreover, let the worst case delay for one task be that it is blocked by one instance of the other task. In this case, the worst case delay for both tasks will never occur simultaneously. However, this fact cannot be accounted such that the resulting analysis may deliver pessimistic results.



(a) coupling of a virtual protoyp and Jitterbug, cf. [4]



(b) control quality co-simulation

Figure 1: Control quality evaluation: (a) depicts the approach presented in [4] that decouples the virtual prototype from the control quality evaluation in Jitterbug while (b) depicts the proposed approach that overcomes the drawbacks of [4] by a co-simulation.

Contributions. To overcome the decoupling of delay distributions and to consider multiple control quality objectives,

this work proposes a co-simulation of a *virtual prototype* of the control application mapped to the system architecture and the high-level model of the physical environment, see Fig. 1(b). The co-simulation approach relies on two important aspects: (a) *Model consistency* is realized by an automatic compilation of the high-level control application to an executable application in the virtual prototype. This closes the existing gap, cf. [4], between the high-level model of the physical environment modeled by the control-engineer and the model of the control application employed for the DSE at the ESL. (b) The simulators are *synchronized* by developed so-called *s-functions*, modeled in MATLAB Simulink and automatically integrated in the virtual prototype. Thus, the synchronization of the two employed simulation approaches is transparent for the designer.

Outline. The rest of the paper is structured as follows: The employed system model for the DSE is introduced in Sec. 2. The co-simulation approach that combines the simulation of the environment in the high-level controller models and the system behavior in the virtual prototype is introduced in Sec. 3. Section 4 presents preliminary experimental results before the paper is concluded in Sec. 5.

2. SYSTEM LEVEL DESIGN

This work proposes a co-simulation approach that obtains control quality measures which are introduced as principal design objectives during *Electronic System Level* (ESL) design. The vast majority of ESL design approaches, see [6] for a survey, is inspired by the *Y-chart* approach. Given is an *application* that models the functionality and an *architecture* that models the available resources. A *Design Space Exploration* (DSE) aims at offering a set of high-quality *system-level implementations* to the designer who choses one (or several) to be refined in the next lower level of abstraction. The set of high-quality implementations results from the presence of multiple, often conflicting objectives that makes DSE a *Multi-Objective Optimization Problem*. During DSE, implementations are obtained by mapping the application onto the architecture in a process termed *system synthesis*. The quality of each implementation with respect to given objectives and its compliance with given design constraints is determined in an *evaluation* step. During this evaluation step, the proposed co-simulation is applied to obtain several control quality measures as design objectives and design constraints. This section gives a brief introduction of the used system model and system synthesis approach.

In this paper, focus is put on applications that serve as feedback controllers represented in the state space model. The structure of a quite general control system is given in Fig. 2(a): The current *state* of the system x is sampled by a *sensor* and a *controller* computes the new control signal which is send to an *actuator*. The actuator generates the *control law* u which affects the behavior of the physical system that is also called *plant*. To integrate such feedback controllers in a DSE, they are transformed to a graph-based representation. Figure 2(b) shows how the control application are mapped to the graph-based representation that is required for the DSE. Note, that the plant is not part of the application graph as this models physical environment and is respected while evaluation of implementations by the introduced co-simulation. In the following, a brief introduction of this graph-based model is given. The graph-based *specification* (see Fig. 2(b)) consists of the *architecture*, the *application*, and a relation between these two views, the *mapping constraints*:

- The architecture is modeled by a graph $g_{\mathsf{a}}(R, E_{\mathsf{a}})$ and represents all available interconnected components, i. e., hardware *resources*. The vertices $r_1, ..., r_{|R|} \in R$ represent the resources, e. g., processors, buses, sensors or actuators. The edges E_{a} model available communication links between resources.
- The application is modeled by an application graph $g_{\mathsf{t}}(T \cup C, E_{\mathsf{t}})$ that represents the behavior of the system. The vertices $t_1, ..., t_{|T|} \in T$ denote *processing tasks* and the vertices $c_1, ..., c_{|C|}$ *communication tasks*. The directed edges $e \in E_{\mathsf{t}} \subseteq (T \times C) \cup (C \times T)$ denote data dependencies between tasks.
- The relation between architecture and application is given by a set of *mapping edges* E_{m}. Each mapping edge $m_1, ..., m_{|E_{\mathsf{m}}|} \in E_{\mathsf{m}}$ is a directed edge from a task to a resource, with a mapping $m = (t, r) \in E_{\mathsf{m}}$ indicating that a specific task t can be mapped to hardware resource r.

From the specification of the system that implicitly includes all design alternatives, a system level *implementation* has to be deduced, respectively synthesized. This implementation corresponds to the found hardware/software system that will be implemented. The synthesis thereby involves the following steps:

- The *allocation* $\alpha \subseteq R$ denotes the subset of the available resources that are actually used and implemented in the embedded system.
- The *binding* $\beta \subseteq E_{\mathsf{m}}$ is a subset of the mapping edges in which each processing task is *bound* to a hardware resource that executes this task at runtime.
- The *routing* $\gamma \subseteq R$ of each communication task is a subset of resources over which a communication task is routed.
- The *schedule* ϕ can be either static (with predefined start times of the tasks) or dynamic (with assigned periods or deadlines to the tasks).

Thus, an implementation is given as a tuple $(\alpha, \beta, \gamma, \phi)$.

3. CONTROL QUALITY CO-SIMULATION

In order to consider control quality during DSE, three quality metrics of feedback control systems are introduced as principal design objectives. Afterwards, the proposed co-simulation approach that enables to determine the quality metrics is presented.

3.1 Quality of Control Loops

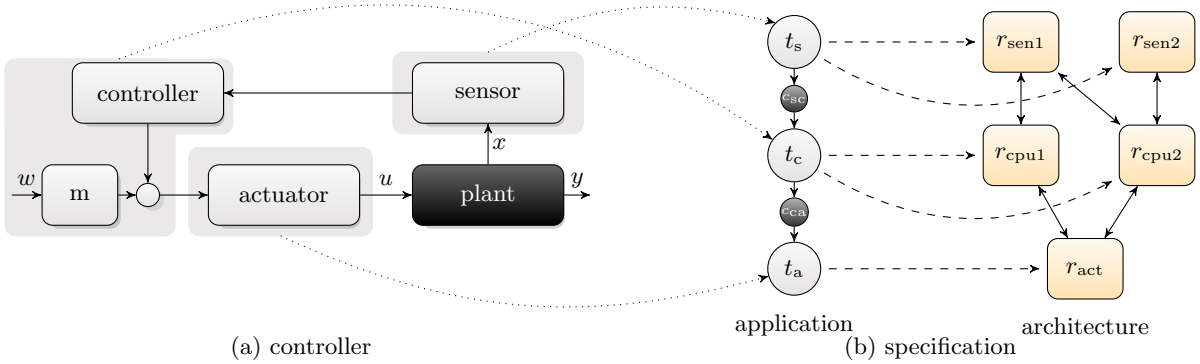
Evaluating feedback control systems requires a method to assign quality metrics to an implemented controller. In [7], the authors consider three control performance metrics and analyze them analytically. In this work, three quality metrics are introduced: (1) the *transition time* (2) the *peak overshoot*, and (3) the *quadratic cost*.

Consider a system being in a state x_s. At time t_1, a new reference value x_e is given to the system. The task of the controller is to find the control law u so that the system ends in state x_e.

The transient time J_1 defines the time to get from the start state x_s to x_e, that is reached at t_2:

$$J_1 = t_2 - t_1 \tag{1}$$

The quadratic cost is a well known metric in the area of optimal control theory. Here, the error while getting from x_s



(a) controller (b) specification

Figure 2: Shown is (a) a controller as a state space model. Its proposed transformation into a specification, see (b), is depicted by dotted edges. The resulting application consist of three tasks with a sensor task t_s modeling the sensor, a control task t_c modeling the controller including the input vector, and an actuator task t_a modeling the actuator. An architecture is depicted in the specification as well, consisting of two different sensors suitable to carry out the sensor task, two CPUs suitable to execute the control task, and an actuator that carries out the actuator task. The possible mapping of tasks to resources is depicted by the dashed edges.

to x_e as well as the control energy u needed for this transition is estimated by J_2.

$$J_2 = \int_{t_1}^{t_2} x(t)^T \cdot Q \cdot x(t) + u(t)^T \cdot W \cdot u(t)\,\mathrm{d}t. \qquad (2)$$

The matrices Q and W prioritize the different states in the vector x and u.

The third common quality metric is the peak overshoot. This is the maximum amplitude of the systems output while the transition from x_s to x_e:

$$J_3 = \max_{t_1 \le t \le t_2} (x(t)) \qquad (3)$$

In the synthesized system implementation, several control applications are executed in parallel in a distributed fashion. However, the occurring resource contention on shared components like processors and buses significantly influences the delay from sampling the state of the plant to updating the control signal. These varying sensor-actuator delays are known to heavily influence the quality metrics. Thus, the approach proposed in this work takes varying sensor-actuator delays into account while calculating the quality numbers.

3.2 Design Space Exploration

The DSE is an iterative process that generates implementations consisting of a concrete allocation, binding, scheduling and routing. In the proposed approach, the implementation obtained by the system synthesis is used to generate a *Virtual Prototype* (VP). This VP consists of the architecture, the control application mapped to the architecture, resource characteristics like operations per seconds executable per second, the determined routes for messages, and the derived schedules. The control application is a SystemC implementation of the high-level model gathered from the control engineer. This high-level model given in MATLAB Simulink is automatically transformed to the SystemC implementation to guarantee model consistency between the models employed by the performance simulation using the VP and the plant simulation in the high-level model.

Using a discrete-event-simulation of the VP, the delays caused by computation, communication, and scheduling are determined. Combining this discrete-event-simulation with a plant simulator in a co-simulation gives the chance to observe system behavior under delay characteristics for the VP. The plant that represents the physical environment is modeled using MATLAB Simulink [8]. By observing the plant simulation, the desired quality numbers can be determined and returned as the quality numbers of the implementation under analysis. Note, that this approach allows quality evaluation of each physical environment modeled in MATLAB Simulink that is a widely used modeling tool.

3.3 Co-Simulation

Although MATLAB Simulink offers a precise simulation of controller and plant, it is not possible to consider varying sensor to actuator delays. Therefore, the control application is modeled in SystemC while the plant is modeled in Simulink. The SystemC modeled sensor reads the actual state x of the plant periodically. The sampling times are given as: $p_1, \ldots, p_i, \ldots p_m$. After sampling the state $x(p_i)$ a SystemC module computes the control law $u(p_i)$. The time delay from sensor to actuator caused by computation and scheduling is defined as d_i. This means, after sampling the sensor value at p_i, the control law u_i is updated at $p_i + d_i$. This delay d_i is determined by the discrete-event-simulation of the VP. Now, the plant has to be updated under the delay conditions. Therefore, the control law is updated as follows:

$$u(t) = \begin{cases} u(p_{i-1}), & p_i \le t < p_i + d_i \\ u(p_i), & p_i + d_i \le t < p_{i+1} \end{cases}. \qquad (4)$$

As both simulations run in parallel but influence each other, synchronization mechanisms are necessary. The co-simulation synchronize at each period, when the SystemC sensor reads the state of the plant. Furthermore, the system is synchronized whenever the control law changes. To do so, the SystemC simulation has to be the simulation initiator. Such event driven synchronization mechanisms were also proposed [9], but not introduced in the context of cyber-physical systems.

3.4 Control Quality Observation

To determine the overall control quality, it is necessary to observe the outputs of the plants over time. This can be done by observing the outputs of the plant simulation.

The control performance metrics are observed for one test case. Here the plant is set to an initial state x_s. At a defined time, a new reference input is given to the plant simulation and the state transmission of x is observed. As the focus is put on linear time-invariant (LTI) systems, the observed behavior is representative for all other state transition.

Now, observing one state transmission enables to determine the quality metrics that depend on the current delays during transmission. Let several applications with different periods be mapped to one resource, delays resulting from resource contention vary over time. To respect this fact, the state transitions are observed several times for each plant. With this approach, the delays of the different applications are dependent on each other.

This work observes the three quality metrics introduced in Sec. 3.1 for each control application in the system. During DSE this metrics can either be combined to one design objective or can be respected as three individual design objectives. Combining the metrics leads to the following overall performance for each introduced metric J_1, J_2, and J_3:

$$J_i = \sum_{p \in P} \lambda_p J_{ip}, i \in \{1, 2, 3\} \tag{5}$$

where $|P|$ is the number of plants in the system and λ_p is a weighting factor for control loop p. Note that there are different possibilities to combine the quality metrics and take them into account as different design objectives in the DSE. However, increasing the number of design objectives by a large $|P|$ may deteriorate the DSE.

An important constraint is that each control loop has to be stable. A plant is defined as stable if a steady state is reached after a defined time when a new reference value is given to the system. Otherwise, the overall quality J is set to $-\infty$. As a result, a design constraint is introduced as:

$$\forall p \in P : J_p \geq 0 \tag{6}$$

4. PRELIMINARY EXPERIMENTS

In a preliminary case-study, the applicability of DSE in the area of control-scheduling-co-design is shown, see [4].

Therefore, the MATLAB toolbox Jitterbug [5] was used as a control quality evaluator. Both, plants and controllers are modeled in this toolbox. The delays caused by resource contention are modeled as delay distributions for each controller and passed to the toolbox. As these delays are not independent, the accuracy of the result obtained from Jitterbug may be too pessimistic. Nevertheless, this experiments show the applicability of DSE in the area of control-scheduling-co-design.

In the experimental setup, the architecture graph consists of six processing elements connected via a priority based bus. The processing units vary in their capacity i. e., the operations per second that can be execute and in their hardware costs. Ten plants are controlled by ten controllers executed in a distributed fashion on the architecture. The plant as well as the controller simulation is done in Jitterbug.

The task of the DSE is now to find a mapping and schedule so that the quadratic cost of the control application as well as the hardware cost of the resources is minimized. Therefore, the following parameters are varied during system synthesis: (1) the allocation of processing elements, (2) the mapping of control tasks, and (3) the priorities of the tasks including communication and processing tasks.

As result, the DSE finds 12 high quality solutions that vary in the hardware costs and the control quality. In comparison to a hand optimized reference implementation the DSE can improve the control performance by up to 12 %.

In future experiments with the co-simulation delay dependencies between different control applications will be considered. As the delays are not independent this correlation should not be neglected. Furthermore, Jitterbug is able to evaluate just one quality metric: the quadratic cost. Generally, more quality metrics are of interest as introduced in Sec. 3.1. These metrics can be obtained by the presented co-simulation. As the proposed toolflow is able to solve multi-objective optimization problems, each of these quality metrics will be treated as a single objective in future.

5. CONCLUSION

The quality of the control applications as well as essential requirements such as stability are becoming design aspects of utmost importance for future (distributed) embedded systems. As a remedy, this work proposes an approach at the Electronic System Level (ESL) that considers control performance as a principal design objective. To account for control performance, a co-simulation of a high-level model of the physical environment and of a *Virtual Prototype* modeling the control application mapped to the concrete architecture is proposed. For this co-simulation, model consistency between the high-level control model of the control engineer and the virtual prototype employed by the automatic Design Space Exploration (DSE) is ensured by a direct and automatic transformation approach. The result is an automatic design flow, capable of optimizing multiple control quality metrics as design objectives while respecting given design constraints like scheduling and stability.

6. REFERENCES

[1] B. Wittenmark, J. Nilsson, and M. Törngren, "Timing problems in real-time control systems," in *Proceedings of ACC '95*, 1995, pp. 2000–2004.

[2] S. Samii, A. Cervin, P. Eles, and Z. Peng, "Integrated scheduling and synthesis of control applications on distributed embedded systems," in *Proceedings of DATE '09*, 2009, pp. 57–62.

[3] S. Samii, P. Eles, Z. Peng, and A. Cervin, "Quality-driven synthesis of embedded multi-mode control systems," in *Proceedings of DAC '09*, 2009, pp. 864–869.

[4] N. Mühleis, M. Glaß, and J. Teich, "Control performance-aware system level design," in *Procedings of CPMNS '11*, 2011, pp. 15–20.

[5] B. Lincoln and A. Cervin, "Jitterbug: A tool for analysis of real-time control performance," in *Proceedings of CDC '02*, 2002, pp. 1319–1324.

[6] A. Gerstlauer, C. Haubelt, A. Pimentel, T. Stefanov, D. Gajski, and J. Teich, "Electronic system-level synthesis methodologies," *IEEE Trans. on CAD*, vol. 28, no. 10, pp. 1517–1530, 2009.

[7] H. Voit, R. Schneider, D. Goswami, A. Annaswamy, and S. Chakraborty, "Optimizing hierarchical schedules for improved control performance," in *Proceedings of SIES '10*, 2010.

[8] MathWorks, "Matlab-Simulink," http://www.mathworks.com/.

[9] F. Bouchhima, M. Briere, G. Nicolescu, M. Abid, and E. Aboulhamid, "A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation," in *Proceedings of BMAS '07*, 2007, pp. 1–6.