Integrated Control–Implementation Design[1]

Andrea Balluchi*, Luca Berardi‡, Maria Domenica Di Benedetto‡,
Alberto Ferrari*, Giovanni Girasole‡, Alberto L. Sangiovanni–Vincentelli*,†

* PARADES, Via San Pantaleo, 66, 00186 Roma, Italy
balluchi, ferrari, alberto@parades.rm.cnr.it

‡ Dip. di Ingegneria Elettrica, Università dell'Aquila, Poggio di Roio, 67040 L'Aquila, Italy
lberardi, dibenede, girasole@ing.univaq.it

† EECS Dept., University of California at Berkeley, CA 94720
alberto@eecs.berkeley.edu

Abstract

The design of embedded controllers is about developing control algorithms and their implementation satisfying tight constraints on performance and cost. To reduce design time and implementation cost, we propose a methodology based on the principles of platform-based design. The design process is decomposed into a sequence of steps that involve different levels of abstraction (platforms) related by a refinement relation. The design method is exemplified by applying it to the design of an automotive engine controller.

1 Introduction

Embedded systems are electronic components of a physical system such as a vehicle or a household appliance that typically

• Monitor variables of the physical system such as temperature, pressure, traffic, chemical composition, position, or speed via sensor measurements;
• Process this information making use of one or more mathematical models of the physical system;
• Output signals, such as controls for power circuitry to open or close valves, to ignite sparks, to open or close switches, that influence the behavior of the physical system to control its function and to optimize its performance.

These embedded systems are integrated onto the physical system itself and hidden from the user. The most challenging embedded systems, called *reactive real-time systems*, control the behavior of physical systems with tight timing constraints. We call reactive real-time embedded systems *embedded controllers*. A serious problems in controller design is the present disregard for the interaction of the control laws with their implementation. This neglect leads to long re-design cycles when the timing requirements of the applications are not met. This paper proposes a *general* methodology where the gap between functional design and implementation is bridged. To achieve this goal, we draw on the principles of *platform-based design* [8]. A platform, in this context, is a layer of abstraction that hides the *unnecessary* details of the underlying implementation and yet carries enough information about the layers below to prevent design iterations. The choice of the layers of abstraction and of the corresponding parameters are essential in the quality of the final solution of the design problem. In this paper, platforms for the embedded control design problem are defined in terms of performance parameters and appropriate cost functionals. In particular, we focus on two layers of the platform stack: functional design and the highest level of abstraction of the implementation layers.

2 Platform-based Design Methodology

The basic tenets of the Platform-based Design Methodology as exposed in [8] are:

• Regarding design as a "meeting-in-the-middle process" where successive refinements of specifications meet with abstractions of potential implementations;
• The identification of precisely defined layers where the refinement and abstraction process take place.

The layers then support designs built upon them isolating from lower-level details but letting enough information transpire about lower levels of abstraction to allow design space exploration with a fairly accurate prediction of the properties of the final implementation. The information should be incorporated in appropriate parameters that annotate design choices at the present layer of abstraction. These layers of abstraction are called *Platforms*. In this paper, a platform is defined to be *an abstraction layer in the design flow that facilitates a number of possible refinements into a subsequent abstraction layer (platform) in the design flow.* The ab-

[1]This research has been partially sponsored by PARADES, a Cadence, Magneti-Marelli and SGS-Thomson GEIE, by the European Community Projects IST-2001-33520 CC (Control and Computation) and IST-2001-32460 HYBRIDGEand by the GSRC

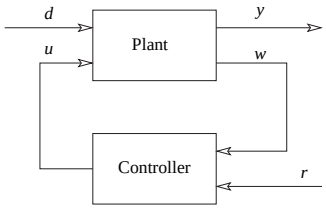

Figure 1: General scheme for the functional design of the controller *system*.

straction layer contains several possible design solutions but limits the design exploration space. During the design process, at every step we choose a *platform instance* in the platform space. Every pair of platforms, the tools and methods that are used to map the upper layer of abstraction into the lower level one is a *platform stack*. Key to the application of the design principle is the careful definition of the platform layers. Platforms can be defined at several point of the design process. Some levels of abstractions are more important than others in the overall design trade-off space. In particular, the articulation point between system definition and implementation is a critical one for design quality and time.

3 Controller Design

In this section, we begin with the formalization of the system specifications for the design of embedded controllers. The system of interest is shown in Figure 1. In this scheme, the system consists of a plant and of a feedback controller,where d models non measurable disturbances to be rejected by the controller, r denotes measurable reference signals and disturbances, w stands for the feedback outputs, u represents the control inputs and y denotes the system outputs (some of them may appear in w as well).

3.1 System Specifications

System specifications are defined in terms of a number N of inequalities of the type:

$$\bar{J}_i\left(\mathcal{J}_i\{y\}|_{x\in\mathcal{X}_i}\right) \leq 0 \qquad \text{for } i = 1 \dots N \tag{1}$$

where:

•the system of inequalities may be related to different operating modes of the system and – possibly – different requirements for each operating mode;
• $\mathcal{J}_i : \mathcal{Y}_i \to \mathbb{R}$ is a functional measuring the requested performance for the controlled system (referred to as performance functional);
• x represents the state of the system, and $\mathcal{X}_i$ represents the family of system evolutions of interest;
• y denotes the system outputs on which the functional is applied and $\mathcal{Y}_i$ is the family of output evolutions.

The evolution of the controlled system may depend on uncertain and time-varying plant parameters, action of noise and disturbances (including reference signals), initial and final conditions.

The operators

$$\bar{J}_i \circ \mathcal{J}_i : \mathcal{X}_i \to \mathbb{R}.$$

combine the values of the functionals obtained for the executions $x \in \mathcal{X}_i$, giving a unique measure of performance.

The ***performance index*** of the system is

$$\bar{J} = \begin{bmatrix} \bar{J}_1 \\ \vdots \\ \bar{J}_N \end{bmatrix} = \begin{bmatrix} \bar{J}_1(\mathcal{J}_1\{y\}|_{x\in\mathcal{X}_1}) \\ \vdots \\ \bar{J}_N(\mathcal{J}_N\{y\}|_{x\in\mathcal{X}_N}) \end{bmatrix} \tag{2}$$

Observe that while the performance functionals $\mathcal{J}_i(\cdot)$ give a measure of a particular evolution of the controlled plant, which depends itself on the factors listed above, the performance index does not. Therefore, it represents the performance metrics for the controlled system. Using the performance index, system specifications can be cast as

$$\bar{J} \leq \mathbf{0} \tag{3}$$

Performance specifications may be quite different according to the application. However, the formulation we have chosen is general enough to express most of them[2].

3.2 Functional Design

Let $\tilde{y}$ and $\tilde{x}$ respectively denote the *representation* in the system model of the physical variables y and x introduced in (1). In traditional methods, functional design is the first step in the design process. Based on a given model of the plant, a set of control strategies are designed to meet the system specifications (1). In this phase, several control strategies can be devised to address the same control problem. These strategies correspond to different choices of physical variables in d, u, w and r, and different control algorithms. Therefore, the solution of the design at the functional level is described by

• a number R of different controller structures, obtained by properly combining some candidate solutions;
• a set of parameters X_C for each controller structure.

We call the elements c in X_C the **control parameters**. A particular control system is identified by selecting a controller structure and an admissible value for the control parameters.

Because of the approximations that are necessary to build a mathematical model of a physical plant, for a

[2]For average case specifications, the performance index is defined in terms of the statistical properties of the output:

$$\bar{J}_i = E\left\{\bar{J}_i(\mathcal{J}_i\{y\}|_{x\in\mathcal{X}_i})\right\},\ i = 1, ..., N.$$

where $E\{\cdot\}$ denotes the mean value.

given controller structure $r \in \{1, \ldots, R\}$, the system specifications (1) are guaranteed when the set of control parameters and the plant model satisfy the following conservative relation

$$\forall c \in X_C, \quad \bar{J}_i = \bar{J}_i\left(\mathcal{J}_i\{y\}|_{x\in\mathcal{X}_i}\right) \leq \bar{J}_i\left(\mathcal{J}_i\{\tilde{y}(r,c)\}|_{\tilde{x}\in\mathcal{X}_i}\right) \equiv \quad J_i(r,c) \leq 0 \qquad \text{for } i = 1 \ldots N \tag{4}$$

Note that, while $\mathcal{J}_i\{\cdot\}$ is a functional that is applied to the system outputs, $J_i : \{1, \ldots, R\} \times X_C \to \mathbb{R}$ is a function of the controller structures and control parameters. The chosen model is required to be conservative with respect to the behavior of the actual system in terms of system specification (1) for the system evolutions of interest $\mathcal{X}_i$.

4 Platforms

Let $m \in \{1, \ldots, M\}$ be a possible implementation architecture and $z \in X_Z$ the overall related implementation parameters[3] (e.g. CPU clock, scheduling algorithm, task priorities, hardware frequency). In a decoupled control and implementation design, given the conservative representations of the controlled plant evolutions introduced in (4), designers first identify the values of control parameters, and then find the best implementation structure and parameters (m, z) for which the system cost is minimum (mapping phase). Then, if the performance specifications given in (1) are not met, the design cycle (i.e. control design and implementation) starts again, until closure between specification and implementation is reached. An ideal approach would be to solve the control and implementation problems together: exploring the set of all feasible solutions in the parameter space (r, c, m, z) for which conservative representations of relations (4) are satisfied and select the one with the minimum cost. Unfortunately this problem contains too many design parameters and equations, making the the solution of the problem infeasible. In the platform-based design paradigm exposed in Section 2, we represent the effects of the actual implementation parameters with an abstract model characterized by idealized parameters. Each choice of these parameters identifies a platform. Hence, the design problem is decomposed into a set of platform mapping, each of them shielding lower level details. In this view, control design is a platform mapping with as many implementation details as exposed by the implementation platform.

4.1 Implementation model

The essential issue for representing implementation platforms in an abstract way is to determine the effect of implementation platforms on control algorithms. Accuracy of measurements and actuations, and how to represent the fact that computation and communication take time are important in this respect. The task of the control designer using the principle of integrated control-platform design, is now to choose algorithms and platform parameters that are robust with respect to errors due to the computation of the control law. As argued above, it is important to classify and characterize carefully the effects that a particular implementation has on the behavior of the controlled systems. The following classification aims at representing these effects.

Control loop delay (or latency time), from the physical controller inputs w and r to the controller output u. In a digital control system the process of sensor reading, control law computation and output actuation takes a certain amount of time that is usually not negligible. A major reason of this delay is the complex sharing of computation and communication resources by several control loops. There are several known techniques to compensate a constant or varying delay even at the price of a reduced performance of the control system ([3], [4], [5]).

Quantization error, acting on the controller input and output signals (namely, w and r sensors and u actuators). In a analog-digital conversion, the digital signal is not a faithful reproduction of the analog signal, hence the conversion process bears a distortion, also called *quantization error*. Sometimes, especially in low-cost micro-controllers, the quantization error an cause an unacceptable decrease of the overall control system performance.

Sample & hold (S/H) errors on the controller channels. When a digital electronic platform is used, control signals are computed on the basis of sampled–data measurements of the plant evolution and are issued at given sampling rates. The S/H error can be expressed as the difference between the *ideal* control law synthesized considering the continuous-time dynamics of the controlled plant and the actual constant value over a fixed sampling period.

Control algorithm computation imprecision. Among the possible sources of computation errors, the most important are those due to fixed point arithmetic, missed deadlines and switching detection errors, which may cause major malfunctions. For real-time control systems with **hard** deadlines, missed deadlines may be catastrophic, and should be carefully avoided by design. Systems with **soft** or **firm** deadlines can temporarily allow missing deadlines at the cost of a reduced system performance ([3],[7]). Often the control algorithms require the detection of switching conditions expressed as function of measured signals and/or internal variables. The implementation of the control law must detect these switching conditions with the requested accuracy. Otherwise, the behavior of the controlled system might become unpredictable. To capture these effects, we propose to represent them in terms of perturbations on the controller input/output channels. The abstract model is illustrated in Figure 2. The disturbances n_u, n_w, n_r and the delays Δ_u, Δ_w, Δ_r represent, respectively,

[3] An implementation parameter must be given a vlaue to identify a platform instance and has an impact on its performance and cost.

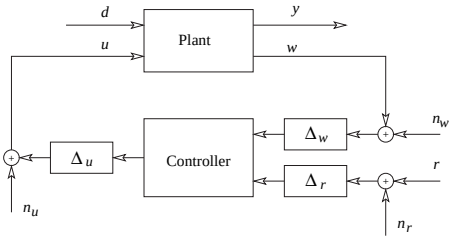

Figure 2: Abstract representation of the effects of implementation non–idealities.

value and time domains perturbations due to the implementation and acting on the control inputs u, feedback outputs w and reference signals r. Depending on the selected platform, these perturbations can be represented by different models and characterized by the abstract parameters p. The admissible time-domain perturbations, specified by Δ_w, Δ_r, Δ_u, define constraints for the implementation of the controller concerning: control loop delay and sample & hold (S/H) errors. While implementation effects due to signal quantization and imprecision on the control algorithm computation are constrained by the characterization of the disturbances n_r, n_w and n_u, modeling the admissible value domain perturbations. A set of implementation platforms with the corresponding exported parameters is defined by:

- a number S of different platform structures;
- a set of parameters X_P for each platform structure;
- a set of platform constraints

$$J_v(s,p) \leq 0, \text{for} v = 1 \ldots V \tag{5}$$

For a given platform structure $s \in \{1, \ldots, S\}$, elements $p \in X_P$ are referred to as the **platform parameters**.

4.2 Examples of Platforms

In a *ideal time-discrete* platform, computation cannot be performed in continuous fashion, but values of variables have to be sampled at discrete times. This platform does not carry information about lower bounds on sampling frequencies and allows the computation of control laws in zero time. A refinement of this ideal platform is the *non zero delay* platform where we introduce new parameters exposing the non-zero delay between the input sampling and the output action of the control. In this case, since, for cost reasons, the implementation architectures have limited computation and communication resources, the control algorithms must be mapped to a set of task to be executed on the shared resources. Operating systems are required to allocate the tasks to the CPUs. Moreover, if communications take place between CPUs and peripherals via shared resources (e.g. a bus), those must be allocated using arbitration policies. The behavior of these scheduling mechanisms is in general very hard at the control level and it is captured by platform models. The Time Triggered architecture proposed by Kopetz [3] is an abstraction that has been very effective in dealing with scheduling problems. In this scheme, each task has a priori fixed slot of time in an overall cyclic time allocation. The following is a platform view of the time-triggered architecture derived from the one presented in [6]:

- a set of periodic processes, with worst case execution time (WCET) e_i and allocated time slot ts_i;
- a constraint that guarantees the scheduling with total utilization U_{cpu}: $J_v^{(1)}(e_i, U_{cpu}, ts_i) = \sum_{i=1}^{m} \frac{e_i}{ts_i} - U_{cpu} \leq 0$
- a constraint that guarantees the control-execution latency τ lower than the smallest WCET: $J_v^{(2)}(e_i, \tau) = \tau - \min_i(e_i) < 0$

The abstraction provided by this platform allows the formalization of the process of mapping the control algorithms on a single CPU.

The choice of abstraction, including the parameters selected to represent an implementation platform, is an integral part of the design. While we can give general guidelines on how to choose them, the goal of the paper is to present a general framework where the choices can be embedded and evaluated.

4.3 The Platform-based Design Flow

In the control parameters and platform parameters product space, the set of feasible solutions is given by

$$\begin{aligned} U \quad = \quad & \{(r,c,s,p) \mid r \in \{1, \ldots R\},\ c \in X_C, \\ & s \in \{1, \ldots S\},\ p \in X_P \text{ such that } J_i(r,c,s,p) \leq 0, \\ & \text{for } i = 1 \ldots N+V\} \end{aligned} \tag{6}$$

where J_i include both the conservative expression of (4) with platform parameters s, r, and the platform constraints (5).

We introduce an objective function for the control-implementation platform mapping process as follows:

$$\arg \min_{(r,c,s,p) \in U} H(r,c,s,p) \tag{7}$$

If we had a way of correlating the cost of the final implementation with the parameters, then H could be chosen as an estimate of the final cost. If the platform is distant from the actual implementation, an accurate estimate of the cost is difficult to obtain. In these cases, a better solution, as demonstrated in [2], is *to adopt a function that measures the "size" of the design space where platforms at lower levels of abstraction can be selected.* If indeed the platform parameters chosen by the optimization process can be easily achieved by platforms at lower levels of abstraction, we minimize the risk of expensive design cycles that span several platforms and we offer a better platform choice while we are approaching the implementation level. The objective function that reflects these principles was called *flexibility* function in [2]. In some sense, the flexibility function is an auxiliary function that serves the purpose of a more efficient search of the design space. While the macro aspects of this function are easy to establish and can be generalized,

the actual choice of flexibility functions is the result of the experience of the designer and can be refined during re-design to reflect more accurately the difficulty of achieving the platform parameters. Consequently, there is no *a priori* best form of the flexibility function.
For example, the *flexibility* function of a discrete-time platform (3) can be an increasing function of the sampling time. The higher the sampling time, the easier is to find a platform that can support that sampling time. Note that the function has typically a steep part, where relaxing the sampling time has a great effect in enlarging the design space, and a flat one where relaxing the requirements does not pay off as much.

5 Automotive Application

In this section, we show the application of the proposed methodology to the design of the force tracking control described in [1].

Force Tracking Control. In torque tracking control the engine is to produce a torque $g(t)$ that tracks a reference torque signal $\tilde{g}(t)$. The output of interest is the tracking error $y(t) = g(t) - \tilde{g}(t)$. A natural choice for the measure of system performance is the settling time[4], referred to as $t_S^{CL}(y)$, normalized with respect to the minimum settling time:

$$\mathcal{J}\{y\} = \frac{t_S^{CL}(y)}{\min\, t_S(y)} \tag{8}$$

The performance functional is evaluated on a family $\mathcal{X}_i$ of evolutions determined by torque step references:

$$\tilde{g}(t) = \begin{cases} g_0 & \text{for:} \quad t < 0 \\ g_f & \text{for:} \quad t \geq 0 \end{cases}, \tag{9}$$

starting from the equilibrium state corresponding to the torque g_0, with $g_0 \in [g_{0min}, g_{0max}]$. Then, according to (2), we define the force tracking performance index as the average value of the normalized settling time for g_0 considered as a random parameter with a uniform probability distribution in $[g_{0min}, g_{0max}]$:

$$\bar{J} = E_{g_0 \in [g_{0min}, g_{0max}]} \{\mathcal{J}\{y\}|_{x \in \mathcal{X}_i}\} - J_0 \tag{10}$$

with $\mathcal{J}\{\cdot\}$ as in (8) and J_0 quantifying the degree of sub-optimality.

Functional design of the control algorithm. A hybrid controller that achieves optimal tracking of torque reference signals as in (9) was presented in [1]. The controller is composed of two nested loops: the intake manifold control and the torque generation control. In particular, we focus on the intake manifold control feedback. This algorithm is a discrete time control. Its sampling time T is considered as a control parameter in X_C since it determines the discretization of the physical plant based on which the controller is tuned.

Definition of the implementation model. In addition to the sampling time T, the latency time (or control delay) αT is also used to evaluate the closed–loop performance. We assume the control delay αT to be a fixed quantity. The chosen scheduling policy should ensure $\alpha \leq 1$. We observe that while T is a control parameter since it enters in the control law design stage (see [1]), the latency α, instead, has to be considered as a platform parameter. In spite of their different semantics, both control and platform parameters are handled in the same way when it comes to the choice of the optimal or feasible values for them, as we will see shortly. A conservative approximation of (10) is given by

$$J(T, \alpha) = E_{g_0 \in [g_{0min}, g_{0max}]} \{\mathcal{J}\{\tilde{y}(T, \alpha)\}|_{\tilde{x} \in \mathcal{X}_i}\} - J_0 \tag{11}$$

where $\tilde{y}(T, \alpha)$ is given by a model representing the effects of both the sampling time T and the control delay α. Hence, the relation

$$J(T, \alpha) \leq 0 \tag{12}$$

guarantees that *the normalized settling times are smaller than J_0 on the average.*

Platform–control co-design. The first problem we have to solve in the platform–control co-design framework is the determination of the admissible set U defined in (6), namely, *the set of admissible values for T and α, such that (12) holds.*
Since $J(T, \alpha)$ as in (11) is not available in closed form, we evaluated it on a large number of simulations for different g_{0i} (for $i = 1...N$). The expected value $J(T, \alpha) = E\{\mathcal{J}(\tilde{y}_i(t; T, \alpha))\} - J_0$ is thus obtained by means of the corresponding statistical approximation[5]:

$$J_N(T, \alpha) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{J}(\tilde{y}_i(t; T, \alpha)) - J_0$$

Then, $J_N(T, \alpha)$ is interpolated with a polynomial function of the sampling time T and latency α, whose coefficients are determined using a least squares fitting algorithm applied to the observed quantities $J_N(T_k, \alpha_k)$, where $k = 1 \ldots N_{sim}$ and N_{sim} is the number of simulations carried out. Figure 3 shows the interpolating function representing the performance index $J(T, \alpha)$ and the admissible set satisfying $J(T, \alpha) \leq 0$.

[4]The settling time is defined as the time needed for the error $y(t)$ to reach and stay within a given range around zero.

[5]Since J_N is a function of independently distributed random variables g_{0i} (for $i = 1...N$), it has a normal distribution for $N \to \infty$ with expected value $J(T, \alpha)$ and variance $\sigma^2 = E\{\mathcal{J}(\tilde{y}_i(t; T, \alpha))^2\} - J(T, \alpha)^2$. Furthermore,

$$P\left\{|J_N(T, \alpha) - J(T, \alpha)| < \frac{k\sigma}{\sqrt{N}}\right\} = \frac{2}{\sqrt{\pi}} \int_0^{\frac{k}{\sqrt{2}}} e^{-u^2} du$$

gives a bound of the estimation error $J(T, \alpha) - J_N(T, \alpha)$.

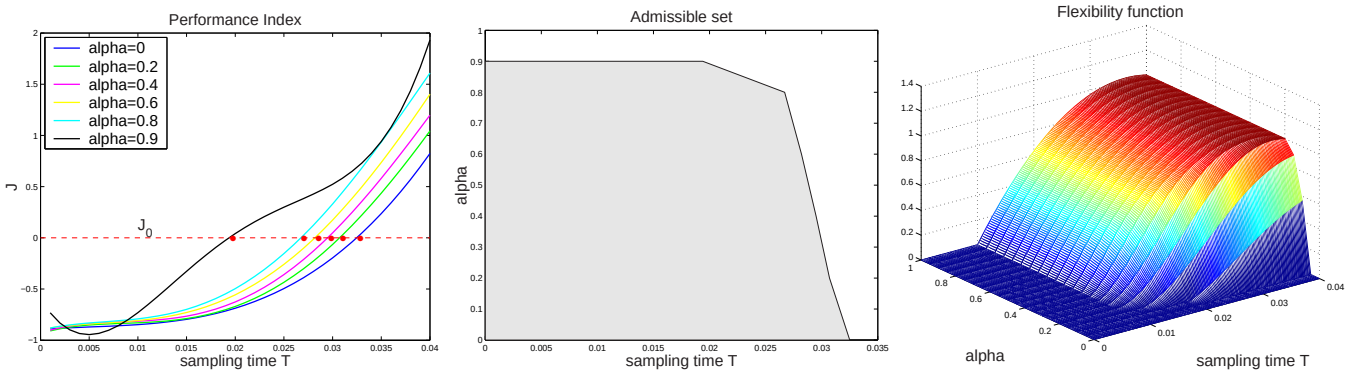

Figure 3: Profile of the performance index, the admissible set and the flexibility function.

To select *one* pair (T, α), we introduce an auxiliary flexibility function. We advocated the use of a quadratic function so that the resulting optimization problem is fairly simple to solve numerically. We chose:

$$f_{FT}(\alpha, T) = f_1(\alpha T) f_2(T) \tag{13}$$

where

$$f_1(\alpha T) = \max\{0, -0.8(\alpha T)^2 + 0.4\alpha T - 0.029\alpha T, \mathfrak{d}(\alpha T)\}$$

$$\text{with:} \quad \mathfrak{d}(\alpha T) = \left\{ \begin{array}{lll} 0 & \text{if:} & \alpha T < (\alpha T)_{max} = 8\text{msec} \\ 1 & \text{if:} & \alpha T \geq (\alpha T)_{max} \end{array} \right\}$$

models how easy it is to design an implementation that exhibits a latency time smaller than αT, and

$$f_2(T) = \max\{0, -0.92T^2 + 0.177T - 0.0015\}$$

quantifies how easy it is to implement the control loop on a sampling period of T.
The two quadratic functions $f_1(\cdot)$ and $f_2(\cdot)$ have been tuned on the basis of data obtained from our industrial partner Magneti Marelli Powertrain (Bologna, Italy) and the inputs from a number of experienced designers. The flexibility function (13), depicted in Figure 3, attains its maximum values for $(\alpha^*, T^*) = (0.4, 29.6)$ inside the admissible set.

6 Conclusions

In this paper, we addressed the problem of designing embedded controllers taking into consideration both functional and implementation aspects. In particular, we proposed a platform-based methodology where the standard controller design procedures are enriched by considering a set of parameters that represent abstractly the capabilities of an underlying implementation architecture.

Acknowledgments

The authors wish to thank R. Bafile and M. Masciovecchio for their valuable help and the numerical assessment of the methodologies introduced in this paper.

References

[1] A. Balluchi, A. Bicchi, C. Caterini, C. Rossi, and A. L. Sangiovanni-Vincentelli. Hybrid tracking control for spark–ignition engines. In *Proc. 39th IEEE Conference on Decision and Control*, volume 4, pages 3126–31, Sydney, NSW, Australia, December 2000.

[2] H. Chang, E. Charbon, U. Choudhury, A. Demir, E. Felt, E. Liu, E. Malavasi, A. Sangiovanni-Vincentelli, and I. Vassiliou. *A Top-down Constraint-driven Design Methodology for Analog Integrated Circuits*. Kluwer Academic Publishers, Boston/London/Dordrecht, 1997.

[3] H. Kopetz. *REAL-TIME SYSTEMS. Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[4] J. Nilsson, B. Bernhardsson, and B. Wittenmark. Some topics in real-time control. In *Proceedings of American Control Conference*, Philadelphia, USA, 1998.

[5] J. Nilsson, B. Bernhardsson, and B. Wittenmark. Stochastic analysis and control of real-time systems with random-time delays. *Automatica*, 34:1:57–64, 1998.

[6] L. Palopoli, C. Pinello, A. Sangiovanni-Vincentelli, L. El-Ghaoui, and A. Bicchi. Synthesis of robust control systems under resource constraints. In *Proc. HSCC2002*, LNCS-2289, Springer-Verlag, NY 2002

[7] P. Ramnathan. Graceful degradation in real-time control applications using (m, k)-firm guarantee. In *Proceedings of the IEEE Real-Time Systems Symposium*, 1997.

[8] A. Sangiovanni-Vincentelli. Defining platform-based design. *EEdesign*, February 2002. http://www.eedesign.com/.