DESIGN OF A MOTORCYCLE ENGINE CONTROL UNIT USING AN INTEGRATED CONTROL-IMPLEMENTATION APPROACH [⋆]

Andrea Balluchi [*] **Maria D. Di Benedetto** [**]
Alberto Ferrari [*] **Giovanni Gaviani** [***]
Giovanni Girasole [**] **Claudio Grossi** [***]
Walter Nesci [***] **Michele Pennese** [***]
Alberto L. Sangiovanni–Vincentelli [*,****]

[*] *PARADES, Via S. Pantaleo 66, 00186 Rome, Italy,* {balluchi,aferrari,alberto}@parades.rm.cnr.it
[**] *DIEL, Università dell'Aquila, Poggio di Roio, 67040 L'Aquila, Italy,* {dibenede,girasole}@ing.univaq.it
[***] *Magneti Marelli Powertrain, Via del Timavo, 33, 40134 Bologna, Italy,* {giovanni.gaviani,claudio.grossi, walter.nesci,michele.pennese}@bologna.marelli.it
[****] *EECS Dept., University of California at Berkeley, CA 94720, USA,* alberto@eecs.berkeley.edu

Abstract: The design of automotive control systems is becoming increasingly complex as the level of performance required by car manufactures grows continuously and the constraints on cost and development time imposed by the market become tighter. A successful design, without costly and time consuming re-design cycles, can be achieved only by using an efficient design methodology that allows for component re-use and evaluation of platform requirements at the early stages of the design flow. In this paper, we illustrate the application of an integrated control-implementation design methodology, recently proposed by our group, to the development of the top few layers of abstraction in the design flow of an engine control system for motorcycles.

Keywords: Embedded systems, design methodologies, automotive control, hybrid systems.

1. INTRODUCTION

An automotive engine control unit is a reactive real-time embedded system, since it is designed to control the behavior of a physical system (namely, the engine) with tight timing constraints. Reactive real-time embedded systems, referred to as *embedded controllers* also, are the most challenging embedded systems to design.

In the standard approach to embedded controller design, a serious problem is the present disregard for the interaction of the control algorithms with their implementation. This neglect leads to long re-design cycles when the timing and accuracy requirements of the applications are not met. In (Antoniotti *et al.*, 1998) and (Balluchi *et al.*, 2002)

[⋆] The work has been conducted with partial support by the European Community Projects IST-2001-33520 CC (Control and Computation) and IST-2001-38314 COLUMBUS, and by the PRIN02 Project.

the authors proposed a general methodology to bridge the gap between functional design and implementation of embedded controllers. The proposed design methodology is based on the principles of *platform-based design*, see (Sangiovanni-Vincentelli, 2002). 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. In this paper, we describe the application of the methodology presented in (Balluchi *et al.*, 2002) to the design flow of an embedded controller in the automotive industry. In particular, we consider the design of engine control units (ECUs) for motorcycles. This application is of importance in industry since tighter laws on emission force the adoption of embedded controllers, which are considerably less complex than the ones in cars but at the same time, exhibit all the tight interactions with the plant typical of car applications that make the design of ECUs very challenging. In particular, we focus on the upper three layers of the platform-based design methodology described in (Antoniotti *et al.*, 1998), namely, the functional decomposition layer, the control strategies layer and the implementation abstraction layer. The importance of these platform layers derives from the fact that most of the critical design choices are taken in the early stages of the design flow and missteps in these stages produce costly and time consuming re-design cycles. Hybrid system techniques are extensively used to evaluate the behavior of the system at each layer (Antsaklis, 2000).
The paper is organized as follows. In Sec. 2, we briefly recall the main concepts of platform-based design. In Sec. 3, we present our approach to functional decomposition, the control strategies and the implementation abstraction layers of the motorcycle ECU. Some concluding remarks are presented in Sec. 4.

2. PLATFORM-BASED DESIGN METHODOLOGY

The basic tenets of the Platform-based Design Methodology as exposed in (Sangiovanni-Vincentelli, 2002) are:

- • Regarding design as a "meet-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.

These layers support designs built upon them allowing the designer to be freed 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*. 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 abstraction 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. Some layers are more important than others in the overall design trade-off space. In particular, the articulation point between functional design and implementation is a critical one for design quality and time. In the platform-based design paradigm, the effects of the actual implementation is represented by an abstract model characterized by idealized parameters. Each choice of these parameters identifies an implementation platform. In this view, control design is a platform mapping with as many implementation details as exposed by the implementation platform.

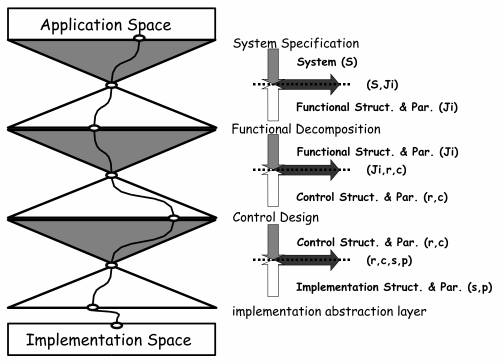
Application Space
System Specification
System (S)
(S,Ji)
Functional Struct. & Par. (Ji)
Functional Decomposition
Functional Struct. & Par. (Ji)
(Ji,r,c)
Control Struct. & Par. (r,c)
Control Design
Control Struct. & Par. (r,c)
(r,c,s,p)
Implementation Struct. & Par. (s,p)
implementation abstraction layer
Implementation Space

Fig. 1. Upper part of the platform stack.

3. INTEGRATED CONTROL-IMPLEMENTATION DESIGN OF A MOTORCYCLE ECU

In this section, we present the results of the application of the methodology for integrated control-implementation design, described in Sec. 2, to the development of an ECU for motorcycles. The design process, which includes formalization of system specifications, functional decomposition and deployment, controller selection and modeling of abstractions of potential implementations, has been carried out in Magneti Marelli Powertrain

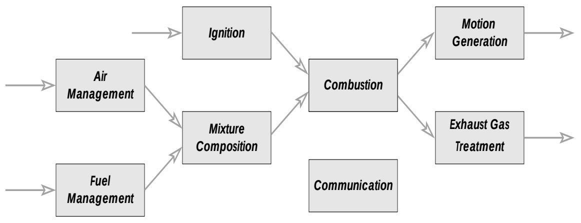

Fig. 2. Functional decomposition.

Division with the support of Parades and the University of L'Aquila. A schematic view of the design process is shown in Fig. 1.

3.1 From System Specification to Functional Decomposition

To tackle complexity, the system is decomposed into a number of interacting simpler sub-systems, called *functions*. The decomposition is based on the understanding of the physical process of interest. This first stage is clearly a key step towards a good quality design, since it leads to a design process that can be carried out as independently as possible for each component (see (Antoniotti *et al.*, 1998) for more details). By the decomposition process, the objectives and constraints that define the system specification are distributed among the components, so that the composition of the behaviors of the components, made feasible and possibly optimal with their own constraints and cost functions, is guaranteed to meet the constraints and the objectives of the overall controlled system.

3.1.1. Functional platform. In our methodology, this design step represents a first refinement of the system specifications into a platform abstraction capturing a structure of the implementation. For engine control, we model the platform at this layer with eight main functions described in Fig. 2 (see (Antoniotti *et al.*, 1998)). For the motorcycle ECU under design, the platform representation is a structure composed of the following functions:

- *Motion Generation*: driver/vehicle interface management, torque generation and transmission, synchronization.
- *Exhaust Gas Treatment*: TWC warm-up control, temperature estimation.
- *Combustion*: definition of target for spark advance and AFR, engine torque estimation.
- *Ignition*: actuation of target spark advance.
- *Mixture Composition*: AFR control, definition of target fuel mass and injection time, lambda sensor measurement and heating.
- *Fuel Management*: actuation of fuel injection, fuel compensation during transients.
- *Air Management*: air temperature and throttle valve angle measurements, cylinder aircharge estimation, throttle valve control.
- *Communication*: CAN communication with CVT and dash–board.

Since, in general, it is difficult to decompose the system into independent parts, the determination of the local objectives and constraints has to be the result of a careful trade-off between the desire of optimality at the global level and the ease of design for each component.

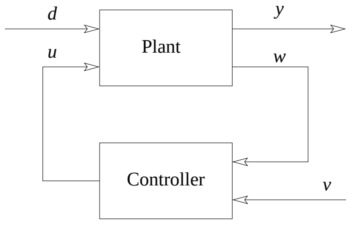

Fig. 3. General scheme for the functional design.

3.1.2. Functional refinement. The mapping of system specification to the platform model (previously described) defines the behavior for each functional component. This refinement can be cast into the classical robust control representation depicted in Fig. 3, where the controller represents the behavior of the function under design and the plant represents the remaining part of the system (plant plus other functions). Furthermore, d models measurable and unmeasurable disturbances to be rejected, v denotes reference signals and commands, w stands for feedforward and feedback signals, u represents the control inputs and y denotes the system outputs.

The behavior of the function is represented by a functional specification defined in terms of a number N of inequalities of the type:

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

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 performance of the controlled system on a particular evolution $x \in \mathcal{X}_i$ and the operator $\bar{J}_i(\cdot)$ collects the overall performances in $\mathcal{X}_i$;
- x denotes the state of the plant, and $\mathcal{X}_i$ is the family of system evolutions of interest [1] ;
- y denotes the system outputs on which the functional is applied and $\mathcal{Y}_i$ is the family of output evolutions.

[1] Which may depend on uncertain and time-varying parameters, as well as initial and final conditions.

3.1.3. ECU functional design. As examples, we report below the description of the functional specifications for two tasks of the motorcycle ECU, namely the AFR control and the fuel injection actuation. In the sequel we will refer to Fig. 3 and formalism (1).

AFR control. Function inputs produced by other functions: $w = (\mathrm{AFR}_{obj}, \mathrm{AFR})$, with AFR_{obj} desired air-to-fuel ratio, AFR actual air-to-fuel ratio, estimated from lambda sensor measurements. Function output u to the *fuel management* function: the target fuel mass q_{obj}. Disturbance $d = (m_a, \delta_q)$ with: m_a cylinder air flow, δ_q additive step disturbance on fuel mass. System output: regulation error $y = \mathrm{AFR} - \mathrm{AFR}_{obj}$. Performance functional $\bar{J} = \max(\mathcal{J}_1, \mathcal{J}_2)$ with

$$\mathcal{J}_1\{y(t)\} = \{\textit{settling time of } y(t)\} - t_s^M$$
$$\mathcal{J}_2\{y(t)\} = \{\textit{overshoot of } y(t)\} - S^M$$

with $t_s^M, S^M > 0$ upper bounds on settling time and overshoot, respectively. The class of evolutions $\mathcal{X}_1 = \mathcal{X}_2$ is given by those with initial conditions at feasible equilibrium points.

Fuel injection actuation. Function input: $w = (q_{obj}, V_{bat})$, with q_{obj} target fuel mass and V_{bat} estimated battery voltage. Function output: $u = t_{inj} \in [0, t_{inj}^M]$ injector opening time. Plant disturbance $d = (\delta_{bat}, \delta_p)$, with δ_{bat} error on battery voltage estimation and δ_p perturbation of the pressure across the injector. Class $\mathcal{X}$: any evolution under bounded disturbance d and feasible initial conditions and inputs. Denote by q the injected fuel. The performance specification is expressed in terms of the mean-value of fuel injection error $y = q - q_{obj}$ over the interval $[0, t_{inj}^M]$, that is $\mathcal{J}\{y\} = E\{|y|\} - q_{err}^M$, with $q_{err}^M > 0$ the mean-value upper bound.

3.2 From Functional Decomposition to Control Strategies

As represented in Fig. 1, the next step in the design flow consists of a refinement of the functional decomposition obtained in the previous step into a set of control strategies using given control platforms.

3.2.1. Control platforms. On the basis of a model of the plant interacting with the functional component, a set of candidate control strategies are devised for each function. Different control strategies are conceived to allow for exploration of different solutions. These strategies correspond to different choices of physical variables in d, u, w and v, and different algorithms. Therefore, the platforms at the control strategies layer are described by

- a number R of different controller structures;
- a set X_C^r of *control parameters* for each controller structure $r \in \{1, \ldots, R\}$.

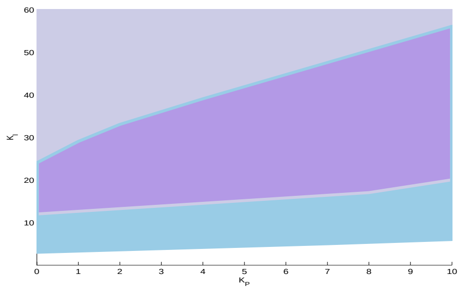

Fig. 4. AFR control parameter admissible set $J(c) \leq 0$, for settling time specification $\mathcal{J}_1$ (gray) and overshoot specification $\mathcal{J}_2$ (cyan).

3.2.2. Control refinement. A particular control strategy, resulting from the mapping of a functional solution into a control platform, is identified by selecting a controller structure and an admissible value for the control parameters. Let $\tilde{y}$ and $\tilde{x}$ respectively denote the representation in the given model of the physical variables y and x. The functional specification (1) is guaranteed for a control structure r, control parameters c, and a given plant model if

$$\begin{aligned} \bar{J}_i\left(\mathcal{J}_i\{y\}|_{x\in\mathcal{X}_i}\right) &\leq \bar{J}_i\left(\mathcal{J}_i\{\tilde{y}(r,c)\}|_{\tilde{x}\in\mathcal{X}_i}\right) \\ &\equiv J_i(r,c) \leq 0 \qquad \text{for } i = 1\ldots N. \end{aligned} \tag{2}$$

Note that, while $\mathcal{J}_i\{\cdot\}$ is a functional that is applied to the system outputs, $J_i : \{1, \ldots, R\} \times X_C^r \to \mathbb{R}$ is a function of the controller structures and control parameters. To guarantee system specification (1), the model that produces $\tilde{y}(r,c)$ has be conservative w.r.t. functionals $\mathcal{J}_i\{\cdot\}$.

3.2.3. ECU control strategies design. Using the representation of Fig. 3, controller parameters (r, c) have been identified and values for which (2) holds have been computed. For the examples considered above we obtained:

AFR control. A PI controller is designed to achieve zero asymptotic error for a fuel mass step disturbance δ_q. Control parameters are $c = (K_P, K_I, \beta)$ with: (K_P, K_I) PID tuning parameters, and β an anti-windup parameter. The abstract plant model is $\mathrm{AFR}(t) = LP(t) * \frac{m_a(t-\tau_0)}{q_{obj}(t-\tau_0)+\delta_b(t)}$, where $LP(t)$ is a unitary gain low-pass filter and τ_0 models the induction to exhaust delay. Following (2), functional specification is achieved for control parameters in the set depicted in Fig. 4.

Fuel injection actuation. A standard piecewise linear approximation of the injector characteristic is assumed: $q = 0$, if $t_{inj} < t^0$, $q = \alpha t_{inj}$, if $t_{inj} \geq t^0$. Gain α depends on the pressure across the injector. Hence, α is represented as the sum of a nominal value α_N and an uncertain component α_U (which depends on the pressure disturbance δ_p). The control algorithm is the inversion of the nominal piecewise model of the injector. The

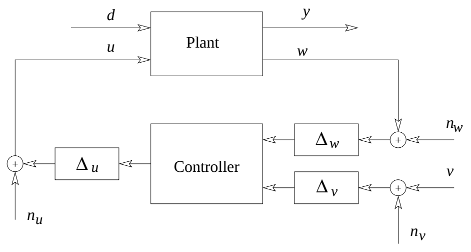

Fig. 5. Abstract representation of the effects of implementation non–idealities.

maximum value of α_U is chosen as the control parameter $c = \delta_\alpha = \max_{\delta_p} \alpha_U$. According to (2), the functional specification is guaranteed for the control parameter δ_α satisfying:

$$\delta_\alpha(t_{inj}^{M^2} - t_0^2)/2t_{inj}^M \leq q_{err}^M \ .$$

3.3 From Control Strategies to Implementation Abstract Model

Finally, referring again to Fig. 1, we describe the third step of the methodology in which control strategies are refined in an implementation abstract model.

3.3.1. Implementation platforms. The essential issue for representing implementation platforms in an abstract way is to determine the effect of implementation platforms on the controlled system performances. Accuracy of measurements and actuations, and how to represent the fact that computation and communication take time and may be affected by errors are important in this respect. The main effects of a particular implementation on the behavior of the controlled system must be carefully classified and characterized. They can be represented in terms of perturbations on the controller input/output channels, as illustrated in Fig. 5. Disturbances n_u, n_w, n_r and blocks Δ_u, Δ_w, Δ_r represent, respectively, value and time domains perturbations due to the implementation and acting on the control inputs u, feedback outputs w and reference signals v. Depending on the selected platform, these perturbations can be represented by different models and characterized by abstract parameters p. 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^s for each platform structure $s \in \{1, \ldots, S\}$;
- a set of platform constraints

$$J_v(s,p) \leq 0, \qquad \text{for } v = 1 \ldots V. \quad (3)$$

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

3.3.2. Implementation abstract model refinement. In the control parameters and platform parameters product space, feasible mappings are given by the set

$$\begin{aligned} U = \{(r,c,s,p) \mid r \in \{1, \ldots R\},\ c \in X_C^r,\\ s \in \{1, \ldots S\},\ p \in X_P^s, \text{ such that}\\ J_i(r,c,s,p) \leq 0, \text{ for } i = 1 \ldots N+V\} \end{aligned} \quad (4)$$

where J_i include both conservative expressions for (2), including the effects of the implementation platform modeled by (s,r), and the platform constraints (3). To select the best mapping, i.e. the best implementation platform, we introduce an objective function $H(r,c,s,p)$ and solve

$$\arg \min_{(r,c,s,p) \in U} H(r,c,s,p) \ . \quad (5)$$

For layers of abstraction distant from the actual implementation, H does not represent the real cost, since an accurate estimate of it would be difficult to obtain. In these cases, a better solution, as demonstrated in (Chang *et al.*, 1997), 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 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[2].

3.3.3. ECU implementation abstract model design. In the design of the motorcycle ECU, for each function the main effects of the implementation on the behavior of the controlled system have been modeled as in Fig. 5 and implementation parameters (s,p) have been identified, along with constraints (3). Then, the feasible parameter set U in (4) has been computed. For the examples considered above we have:

AFR control. When the PI controller is implemented in the digital system, the main platform parameters that affect the performance is the sampling time T_c and the worst case execution time

[2] For example, the *flexibility* function of a discrete-time platform 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.

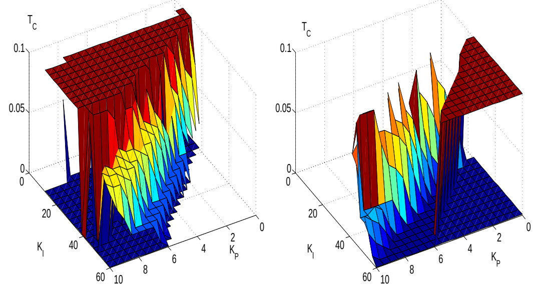

Fig. 6. Values of the platform parameter T_c that guarantee settling time specification $\mathcal{J}_1$ (left) and overshoot specification $\mathcal{J}_2$ (right), for admissible control parameters (K_P, K_I).

W_{AFR}. Hence, $p = (T_c, W_{AFR})$. Fig. 6 reports a section of the set U contained in composed parameter space $(K_P, K_I, \beta) \times T_c$ for which the specification is guaranteed.

Fuel injection actuation. Threshold t^0 depends on the battery voltage $V_{bat} - \delta_{bat}$. The platform parameters $p = (\delta_{t_0}, \lambda)$ represent: the maximal variation of the injector threshold t^0 due to the battery estimation error δ_{bat} (δ_{t_0}), and the accuracy of the injector command (λ). In the composed control and platform parameter space, the performance specification is guaranteed for

$$(\delta_\alpha(t_{inj}^{M^2} - t_0^2) + (\alpha - \delta_\alpha)\delta_{t_0}(2t_0 + \delta_{t_0}) + \\ +2\lambda(t_{inj}^M - t_0 - \delta_{t_0}))/2t_{inj}^M \le q_{err}^M \ .$$

Additional platform parameters are: W_{fi} the worst case execution time and T_{fi} the minimum cycle time of the actuation of fuel injection.

The execution time parameters can be refined to describe platform structures with different hardware/software partitioning, by writing $W_{AFR} = W_{AFR_{hw}} + W_{AFR_{sw}}$, $W_{fi} = W_{fi_{hw}} + W_{fi_{sw}}$ and

$$p = (W_{AFR_{hw}}, W_{AFR_{sw}}, \delta_{t_0}, \lambda, W_{fi_{hw}}, W_{fi_{sw}})$$

Assuming that the implementation has a single CPU, the constraint that guarantees the scheduling with total utilization U_{cpu} can be expressed as

$$J_v^{(1)}(W_i, U_{cpu}, T_i) = \sum_{i=1}^m (W_i/T_i) - U_{cpu} \le 0$$

where W_i is the worst case software execution time of the component i, and T_i is the execution period of that component. This constraint for the algorithms considered here is written as follows

$$W_{AFR_{sw}}/T_c + W_{fi_{sw}}/T_{fi} - U_{cpu} \le 0 \ .$$

Note that a different hardware/software partitioning is captured by different values of the platform parameters and different values of the objective function (5). A pure software implementation is represented with a zero value for any hardware contribution ($W_{i_{hw}}$) to the execution time. An interesting implementation platform under investigation has a fully hardware fuel injection actuation and a fully software AFR control. This implementation is expressed by the following values of the model parameters $W_{AFR_{hw}} = W_{fi_{sw}} = 0$, which shows that the scheduling problem is drastically simplified. The definition of a flexibility function for the motorcycle ECU, which will allow us to select a particular implementation platform, is currently under investigation.

4. CONCLUDING REMARKS

We illustrated the application of an integrated control-implementation design methodology, recently proposed by our group, to the development of an engine control system for motorcycles. The methodology allowed us to: 1) evaluate in terms of performance degradation the *main* effects of control algorithm implementation at the very early stage of design when the control solution is conceived; 2) formally express the constraints on the implementation platform that guarantee fulfillment of the system specification. The results documented in this paper were achieved via intense collaboration between control engineers and hardware/software designers.

REFERENCES

Antoniotti, M., A. Balluchi, L. Benvenuti, A. Ferrari, R. Flora, W. Nesci, C. Pinello, C. Rossi, A. L. Sangiovanni-Vincentelli, G. Serra and M. Tabaro (1998). A top-down constraints-driven design methodology for powertrain control system. In: *Proc. GPC98, Global Powertrain Congress*. Vol. Emissions, Testing and Controls. Detroit, Michigan, USA. pp. 74–84.

Antsaklis, P.J. (2000). Special issue on hybrid systems: theory and applications a brief introduction to the theory and applications of hybrid systems. *Proceedings of the IEEE* **88**(7), 879–1133.

Balluchi, A., L. Berardi, M. D. Di Benedetto, A. Ferrari, G. Girasole and A. L. Sangiovanni-Vincentelli (2002). Integrated control–implementation design. In: *Proc. 41st IEEE Conference on Decision and Control*. Las Vegas, NV, USA.

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

Sangiovanni-Vincentelli, A. (2002). Defining platform-based design. *EEdesign*. http://www.eedesign.com/.