Hybrid Systems in Automotive Electronics Design

Andrea Balluchi[§] Luca Benvenuti[¶] Alberto L. Sangiovanni-Vincentelli[§†]

Abstract

Automotive is certainly one of the most attractive and promising application domains for hybrid system techniques. Some successful hybrid system applications to model development and control algorithm design have already been reported in the literature. On the other hand, despite the significant advances achieved in the past few years, hybrid methods are in general still not mature enough for their effective introduction in the automotive industry design processes at large. In this paper, we take a broad view of the development process for embedded control systems in the automotive industry with the purpose of identifying challenges and opportunities for hybrid systems in the design flow. We identify critical steps in the design flow and extract a number of open problems where, in our opinion, hybrid system technology could play an important role.

1 Introduction

The design of electronic control systems in the automotive industry is particularly challenging due to a number of factors.

A first factor of difficulty is the high complexity of the subsystems that compose the car and have to be operated and monitored by the control system: e.g. the engine, the electrical motor/generator in hybrid vehicles, the driveline, the vehicle body, the suspensions, the brakes, the exhaust gas treatment system, etc. Such subsystems often exhibit very complex behaviors and interact tightly one another. Furthermore, the design of next–generation x–by–wire applications (driving, steering, braking) requires accurate management of the involved interactions between the control system and the driver.

Secondly, the design of electronic control systems is subject to ever increasing demands imposed by the market in terms of vehicle performances, passengers' comfort and safety, and fuel consumption. Such demanding specifications have to be achieved in compliance with legal requirements related to emissions and safety. In particular, very stringent requirements are imposed on the dynamical behaviors of the engine and the vehicle during both fast transients and switching between operation modes.

Further challenges of the design regard the requirements for the hardware and software implementation of the control system. In fact, the implementation has to cope with very critical

[§] PARADES, Via di S.Pantaleo, 66, 00186 Roma, Italy. Tel: +39 06 6880-7923; Fax: +39 06 6880-7926; Email: {balluchi, alberto}@parades.rm.cnr.it.

[¶] Dipartimento Informatica e Sistemistica, Università di Roma "La Sapienza", Via Eudossiana 18, 00184 Roma, Italy Tel: +39 06 44585-973; Fax: +39 06 44585-367; Email: luca.benvenuti@uniroma1.it.

[†] Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720, USA. Tel: +1 510 642-1792; Fax: +1 510 643-5052; Email: alberto@eecs.berkeley.edu.

constraints on cost and reliability (safety and correctness) and constraints on power consumption, weight and position.

Finally, the overall development process in the automotive industry is subject to extremely critical time–to–market limitations, which derive by the necessity of delivering every two–three years new generations of products characterized by high contents of innovation [14, 4].

In today cars, the electronic control system is a networked system with an embedded controller dedicated to each subsystem: e.g. engine control unit, gear–box controller, ABS (Anti–lock Braking System), dashboard controller, and VDC (Vehicle Dynamic Control). The embedded controllers interact each other by communicating over a network.

However, due to the lack of an overall understanding of the interplay of sub–systems and of the difficulties encountered in integrating very complex parts, system integration has become a nightmare in the automotive industry. Jurgen Hubbert, in charge of the Mercedes-Benz passenger car division, publicly stated in 2003: "The industry is fighting to solve problems that are coming from electronics and companies that introduce new technologies face additional risks. We have experienced blackouts on our cockpit management and navigation command system and there have been problems with telephone connections and seat heating". We believe that this state is the rule, not the exception, for the leading Original Equipment Manufacturers (OEMs) in today environment. The source of these problems is clearly the increased complexity of the embedded controllers but also the difficulty of the OEMs in managing the integration and maintenance process with embedded controllers that come from different suppliers who use different design methods, different software architecture, different hardware platforms, different (and often proprietary) Real-Time Operating Systems (RTOS). In fact, most of the OEMs outsource the design and production of embedded controllers to suppliers (so–called Tier–1 companies), which in turn buy IC components and other devices by third parties (so–called Tier–2 companies). As a consequence, embedded controllers are often developed by different Tier–1 companies and are requested to operate in coordination on a same model of a car. Moreover, it is often the case that Tier–1 companies have to integrate some IPs (Intellectual Properties) provided by the OEM at different levels of details (algorithms, legacy code) and, in the near future, possibly by third parties.

Whereas on the one hand the need for standards in the software and hardware domains that will allow plug–and–play of embedded controllers is essential, on the other hand the design process for embedded controllers has to be significantly improved. The first point is the declared objective of the AUTOSAR initiative [25, 3], promoted by leading European OEMs and Tier–1 suppliers: to establish an open standard for automotive electric/electronic architectures.

Hybrid systems techniques can have an important role with respect to the second point. Successful approaches to design of control algorithms using hybrid system methodologies had been presented in the literature, e.g. cut-off control [12], intake throttle valve control [13], actual engaged gear identification [9], adaptive cruise control [30]. However, despite the significant advances of the past few years, hybrid system methodologies are not mature yet for an effective introduction in the automotive industry. On the other hand, hybrid system techniques may have an important impact on several critical open problems in the overall design flow that go beyond the classical controller synthesis step. In particular, system design is a very critical step in the today development process, which could be significantly improved by using hybrid system techniques. In this fundamental design step, system specifications are mapped into an architecture of control algorithms and their requirements.

In this paper, we analyze the design flow for embedded controllers in the automotive industry,

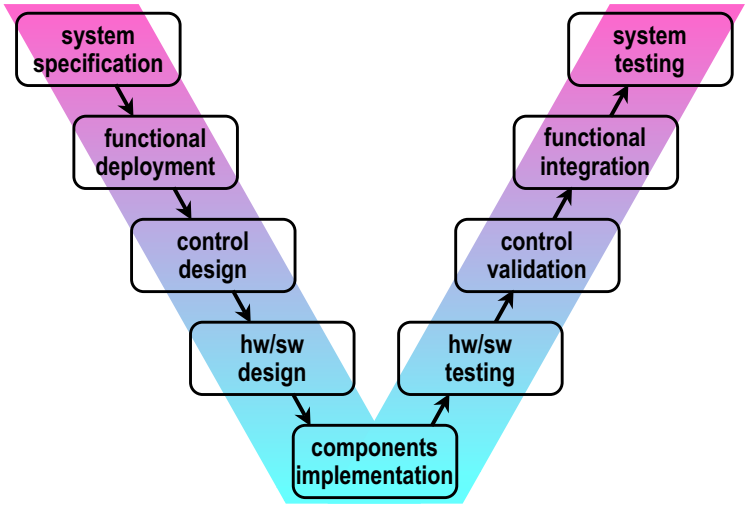

Figure 1: Design and integration flow.

with the purpose of identifying challenges and opportunities for hybrid system technologies. In Section 2, an overview of the typical design flow for embedded controllers adopted by the automotive industry is presented with particular emphasis on the Tier–1 supplier problems. In Section 3, for each design step, we identify critical phases and bottle-neck problems and we extract relevant open problems that hybrid system technologies may contribute to solve. Some conclusion remarks are given in Section 4.

2 Design scenario and design flow

In today cars, the electronic control system is a networked system with a dedicated Electronic Control Unit (ECU) for each subsystem: e.g. engine control unit, gear–box controller, ABS (Anti–lock Braking System), dashboard controller, and VDC (Vehicle Dynamic Control). The ECUs interact by asynchronous communication over a communication network specifically designed for automotive applications, such as CAN. Typically, an ECU implements a multirate control system composed of nested control loops, with frequency and phase drifts between fixed sampling–time actions and event driven actions. It may have more than one hundred I/O signals, may implement up to three hundreds control algorithms and share with the other related ECUs approximately one hundred signals (as for example, the engine control unit).

The complexity of the design of automotive ECUs is further increased by additional very critical constraints on reliability, cost and time–to–market and constraints on power consumption, weight and position.

As a consequence, a successful design, in which costly and time consuming re–design cycles are avoided, can only be achieved using efficient design methodologies that allow for component reuse and for evaluation of platform requirements at the early stages of the design flow (see [24]).

The standard design flow of automotive ECUs adopted by Tier–1 companies (subsystem suppliers)

is represented by the so–called *V-diagram* shown in Figure 1. The top–down left branch represents the synthesis flow. The bottom–up right branch is the integration and testing flow.

In particular, the synthesis flow is articulated in the following steps:

1. *System specification.* This step includes: the formalization of system level customer requirements; the completion of under–specified requirements; the abstraction at the system level of customer requirements regarding lower layers (e.g. either a control algorithm or a piece of software to be integrated in the design).

2. *Functional deployment.* In functional deployment, the system is decomposed into a collection of interacting subsystems and the specifications for each subsystem are defined. Moreover, for each subsystem, the architecture of control algorithms and their specifications are defined in order to meet the given system specifications.

3. *Control system.* This design step regards the synthesis of each control algorithm, according to the specification defined in the previous step, and its validation.

4. *HW/SW components.* The specifications for the implementation of the control algorithms are defined and the hardware and software architectures are designed.

The synthesis flow terminates with the development of the hardware, the software and possibly some electromechanical components. The right branch of the V-diagram describes the integration and testing flow whose purpose is the complete testing of the realization of the ECU and the verification of the compliance with the customer requirements. The steps of the integration and testing flow are:

- *HW/SW testing.* The correct realization of the hardware and software architectures are verified. This step includes testing of real–time implementation requirements, electrical power drivers, communication, etc.

- *Control validation.* The correct implementation of each control algorithm with respect to the given functional description is assessed by testing either its input–output response or the its behavior in closed–loop.

- *Functional integration.* The correct interaction of the implemented control algorithms is tested considering an increasing number of algorithms together, to verify that their composition exhibit the behavior defined during functional deployment.

- *System testing.* The entire ECU is tested against system specification and the compliance with customer requirements is verified.

The *platform-based design* methodology proposed in [35] nicely fits the design flow described by the V-diagram and provides concepts and techniques to achieve an efficient design, aimed to maximize reuse in each design step and to obtain evaluations of platform requirements at the early stages of the design flow (see also [26]). In this context, a platform 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.

The basic tenets of the platform-based design methodology 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 [2], the application of the platform-based design methodology to the design of powertrain control systems has been described. The top-down, constraint-driven design approach proposed in the paper is articulated in five levels of abstraction: system level, function level; operation level; architecture level and component level.

Since in the automotive industry, embedded control system design is highly dominated by the need of implementing an efficient reuse to meet increasing constraints on cost and time–to–market, then a *derivative design* approach is commonly adopted (see e.g. [28, 29]). According to this approach, every two–three years a new generation of products is conceived. The design of the generation is intended to accommodate the specifications of all customers for the next years, so that for each commitment the control algorithms as well as the electrical and mechanical components are obtained by derivation from the current product generation. In the definition of the product generation, the architecture of control algorithms should be conceived is such a way to maximize future re–use, by choosing the correct granularity of partitioning for instance. The resulting ECUs are then variants of a same originating design and ideally share the highest number of parts (algorithms, software modules, hardware parts, mechanical components, etc). The derivative design approach impact all the design steps in the top–down design flow of the V-diagram and possibly the lowest part of the integration and testing flow.

Finally, there is an increasing interest in the automotive industry towards model-based design methodologies. In model-based design, specifications, functional architectures, algorithms, and implementation architectures are represented formally by models thus allowing, at least in principle, formal analysis and automatic synthesis. Using block diagram–based modeling tools, control algorithms are designed and initial validation in off–line simulation is performed. Then, models of the control algorithms are the basis for all subsequent development stages. The advantages are obvious:

- sharing models reduces the risk of mistakes and shortens the development cycles;
- design choices can be explored and evaluated much faster and more reliably;
- the result of a model–based development process is an optimized and fully tested system.

However, today there is an incomplete implementation of the model-based design approach in the development cycle of the automotive industry. In fact, model-based design is widely used for the formal representation of control algorithms, using tools such as either Simulink/Stateflow by The Mathworks [38] or ASCET by ETAS [22], but it is very superficially applied to control algorithm validation. The lack of an extensive model-based validation of the control algorithms results in major efforts in experimental validation, which is very expensive, time-consuming and achieves

only a bounded coverage of the system behavior. Due to the high cost of experimental validation, the OEMs will provide less support to Tier-1 companies for it in the future.

The partial implementation of model-based design in the automotive industry is due to:

- Insufficient investments in design process innovation. In many cases, the reduced efforts devoted to plant modeling prevent accurate model-based validation.
- Lack of methodologies suitable to address critical steps in the design flow, which are currently handled relying on the experience of the designers. A significant example in this respect is functional deployment for which there is nearly no methodological support. The low quality of today functional deployment is witnessed by the results of corresponding step in the testing flow, i.e. functional integration. In this step the large majority of the malfunctioning and noncompliance with the specification are detected, the recovering of which involves often multiple redesign cycles.
- Poor integration of the design tool chain, which is composed by different tools developed independently by different tool makers. Such tools are often connected by file transfer. However, this way of integrating tools defeats the very purpose of model–based design, introducing a high potential of errors in the transformation from one format to another and preventing formal analysis of the properties of the design.

In the rest of the paper, the synthesis flow will be analyzed in details enlightening design steps for which today there is a with weak support of methodologies and tools. As it will be observed, in many cases hybrid system techniques many significantly contribute to provide a more efficient approach to such design steps.

Regarding tool chain integration, in [5] a formal transformation across different tools is illustrated and an example of the proposed approach is reported referring to two tools that are widely used in the automotive domain: Simulink and ASCET. The proposed approach is based on the use of a common formal model, namely the synchronous reactive model of computation, which is used as the common ground to interpret system specifications given with different underlying models.

3 Synthesis flow

In this section, we describe the synthesis part of the automotive design flow covering the levels of system specification, functional deployment, control system and HW/SW components. Emphasis will be placed on the aspects which we believe hybrid system techniques may have relevant impact on, while details of the design with no relation to hybrid systems will be slightly mentioned.

The importance of developing efficient design approaches for the top layers of the design flow is due to 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. Efficient re–use of components is essential to meet the tight constraints on development time and cost and should be fostered at all levels of the design flow.

To support re–use at the functional layer according to the derivative design approach, it is necessary to develop methodologies and tools that allow evaluation of "off the shelf" control algorithms, available from previous product developments and included in the product generation, with respect to the customer requirements for the design at hand. Often, if direct re–use is not possible, requirements can be met with minor re–designs.

3.1 System specification

System specifications, issued by the OEM, define the desired behavior of the vehicle that should be achieved by the design of the control system. The specifications regard

- *performance and driveability* - dynamic behavior of the vehicle, driver assistance, detection and suppression of critical dynamic vehicle states, comfort;
- *fuel consumption*;
- *legal requirements* - environment and safety.

The specifications are defined in terms of a number of operation modes characterized by different controlled variables and regard both discrete and continuous behaviors: in fact system specifications define switching conditions between operation modes as well as the desired continuous behavior for each mode.

Discrete specifications are often given in natural language and only sometimes formalized in some discrete modeling framework. Continuous specifications are given following classical methodologies in terms of steady-state/transient response, frequency domain, robustness and parameter sensibility, disturbance rejection, control effort, cost functions, and constraints.

Often, for both discrete and continuous specifications, requirements are given by specifying requested behaviors on hybrid input/output evolutions. In addition, critical maneuvers for which the behavior requested by the specifications should be guaranteed (possibly up to some allowed degradation) are also identified.

The degree of detail given by the OEMs in describing system specifications is not uniform. Depending on the importance placed by the OEM on each single behavior, functionality or constraint, a different degree of accuracy in describing the requirement itself is used. In particular, some behaviors may result only vaguely specified: in this case, under-specified system requirements are completed by the Tier–1 supplier on the basis of its own experience while trying to maximize reusability for future developments. On the other hand, there are also behaviors that are very detailed in the customer requirements to the the point that the OEM imposes not only a system level requirement but also a particular solution to satisfy it, resulting in an undesirable (at least from an ideal point of view) over-specification.

Since these constraints are often the result of decisions based on insufficient analysis, the feasible design space may be empty thus causing unnecessary design cycles. We do believe that care must be exercised when constraint are entered at abstraction levels that are non appropriate with respect to the role of the company that specifies them.

The previous discussion shows that, since system specification regards both discrete and continuous behaviors, then:

- tools for system specifications, requirements management and system design, validation and verification must be developed to deal with hybrid models.

Moreover, since OEM requirements contain details regarding several levels of the design flow, then to achieve a complete representation of the system at system specification level,

- abstraction techniques that deal with hybrid systems for projecting lower–levels specifications back to upper–levels must be developed;

Finally, hybrid techniques and supporting tools to perform coherence and feasibility analysis at system specification level have to be developed as well.

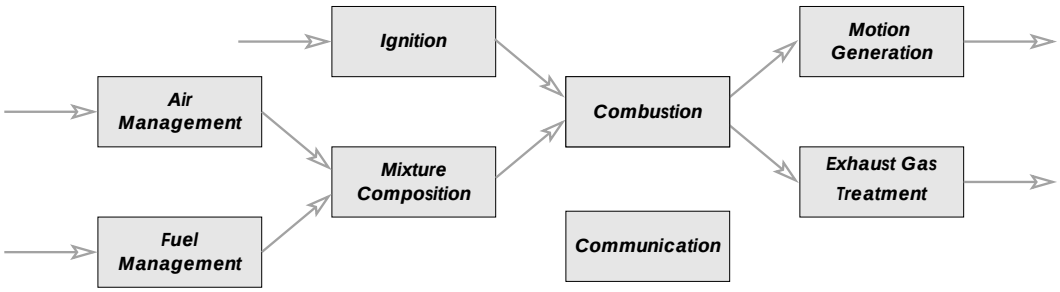

Figure 2: Functional decomposition.

3.2 Functional deployment

In a first stage of the design, the system is decomposed into a collection of interacting components. The decomposition, based on the understanding of the physical process of interest, 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 [2] for more details). A typical decomposition for engine control is shown in Figure 2. The objectives and constraints that define the system specification are distributed among the components by the functional deployment process so that the composition of the behaviors of the components is guaranteed to meet the constraints and the objectives required for the overall controlled system.

In a second stage of the functional deployment, the control algorithms architecture is defined. In particular, the set of control algorithms to be developed for each function and the topology of interconnection are determined. Furthermore, for each control algorithm, desired closed–loop specifications are defined to achieve the requested behavior for each functional component. This process is mainly guided by the experience of system engineers, with little support of methodologies and tools. The sets of measurable and actuated quantities, which will constitute the sets of, respectively, inputs and outputs to the ECU, are often defined by the OEM. In fact, the OEM often defines also sensors and actuators to be used, since they have a major impact on the cost of the control system. In addition, customer requirements may include details on the topology of the control algorithms architecture that further constrains the functional deployment process.

The results of the functional deployment design stage are: the control algorithms architecture and the desired closed–loop specification for each control algorithm.

As a consequence, hybrid formalisms are required to support the description of

- the functional decomposition and the desired behavior for each functional component;
- the architecture of control algorithms, sensors and actuators, for each functional component;
- the desired requirements for each control algorithm obtained from the functional deployment process.

Moreover, the development of methodologies and tools for the synthesis of functional behaviors from system specifications and for validation of the obtained control algorithm requirements w.r.t. the desired functional behaviors, are necessary.

3.3 Control system

At the control system level, the algorithms to be implemented in the architecture defined at the functional level are designed. All control algorithms have to meet the assigned specification, so that their composition within a functional component exhibits the required behavior defined during functional deployment.

In general, the design process for each control algorithm involves:

1. Plant modeling:

   a) model development;

   b) identification;

   c) validation.

2. Controller synthesis:

   a) plant and specifications analysis;

   b) algorithm development;

   c) controller validation.

3. Fast prototyping.

However, since according to the derivative design approach most of the algorithms are obtained from the current product generation, then the entire three–step flow is often only partially performed. For example, if some models of the plant interacting with the control algorithm under design are already available from previews designs, then only some adjustments of sensitive parameters, along with a coarse validation, could be sufficient to obtain reliable models. If not, rigorous identification and validation has to be performed. The complete plant modeling phase is obviously necessary either for the refinement of unsatisfactory existing models, when major changes in the plant have been made or for the development of new functionalities. The plant modeling step is discussed in details in Section 3.3.2

In the plant model and specifications analysis stage, it is first analyzed whether an available algorithm can meet the specification or a new design is needed. If the algorithm is obtained from the current product generation, then the algorithm development stage may involve some minor changes to the re–used algorithm in order to completely cover the new specification. In these cases, the controller validation stage is the most important step to ensure that re–use was successful (see Section 3.3.1). Design of new control algorithms, necessary either when re–use cannot be applied due to major changes in the specification or when new functionalities have to be developed, requires the performance of the entire three-step flow for controller synthesis. Section 3.3.3 reports a detailed illustration of the controller synthesis step.

Fast prototyping is adopted when either control algorithms are designed for new functionalities or major redesigned has occurred to meet more stringent specifications.

Before going through the details of the plant modeling and controller synthesis steps, we report below a methodology to implement derivative design.

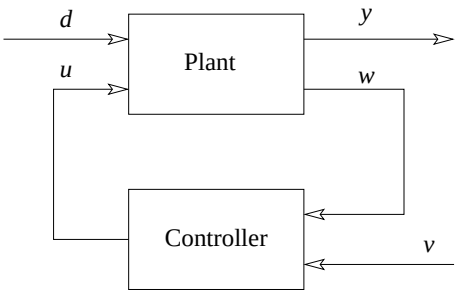

Figure 3: General scheme for control algorithm validation.

3.3.1 Derivative design

The derivative design approach to be effective in the industry development process has to be supported by methodologies and tools that allow evaluation of "off the shelf" control algorithms, available from the current product generation, with respect to the requested closed–loop performances for the design at hand. An approach based on hybrid modeling and randomized algorithms has been proposed in [1].

The problem of validating a control algorithm extracted from the current product generation with respect to the closed–loop specification established during functional deployment can be formalized similarly to a robust control problem. Consider the general closed–loop scheme depicted in Figure 3. The controller represents the control algorithm obtained from the current product generation to be evaluated, while the plant models in an abstract way the remaining part of the system interacting in closed–loop with it: i.e. a part of the physical plant, sensors, actuators and possibly other control algorithms. 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. Due to the different nature of the signals and components in the closed–loop scheme, a hybrid modeling approach has to be adopted to be able to represent the closed–loop behavior.

Let the desired specification defined during functional deployment be formalized in terms of a number N of inequalities of the type:

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

where:

- x denotes an evolution of the system state and $\mathcal{X}_i$ is the family of system evolutions of interest[1];
- y denotes an evolution of the system outputs on which the functional is applied and $\mathcal{Y}_i$ is the family of output evolutions;
- $\mathcal{J}_i : \mathcal{Y}_i \to \mathbb{R}$ 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 $\mathcal{X}_i$.

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

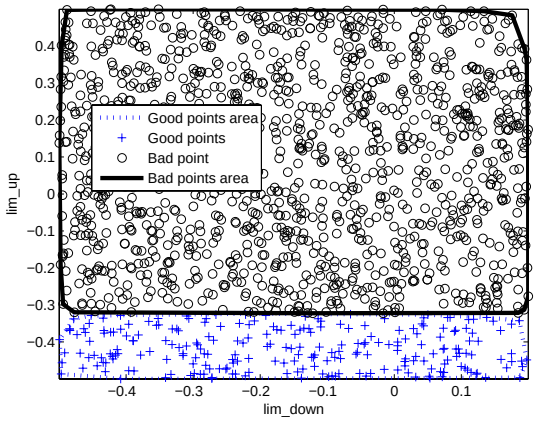

Figure 4: Set of control parameters that achieve desired functional requirements.

The set of inequalities (1) may be related to different operating modes of the system and may specify different requirements in each operating mode.

In context of platform-based design [35], the controller synthesis design step can be view as a refinement of the functional deployment into a set of control algorithms that implements the given functional requirements. Assume that the current product generation contains a set of candidate control strategies that have to be evaluated against the functional requirements (1). These strategies may correspond to different choices of physical variables in d, u, w and v, and different control algorithms. The exploration of the candidate solutions is described by

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

A particular control strategy, resulting from the mapping of given functional requirements 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

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

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. Furthermore, it is worthwhile to notice that, to guarantee the functional requirements (1), the model that produces $\tilde{y}(r, c)$ has be conservative with respect to the functionals $\mathcal{J}_i\{\cdot\}$.

In [1] a methodology and a tool that support design space exploration and validation of control strategies extracted from the current product generation are presented. In the proposed approach, performance criteria are tested for the control algorithms over parameters spaces both via randomized algorithms, i.e. letting the parameters vary according to a given probability distributions, and stochastic algorithms, i.e. letting the parameters evolve in time according to a random coefficient

stochastic differential equation. The results of the analysis are given in the parameter spaces. If the computation returns an empty control parameter set, then the corresponding control strategy extracted from the current product generation is not suitable for the design at hand since it does not achieve the desired functional requirements (1). If the returned set in the control parameter space is large, then the corresponding algorithm easily satisfies the functional requirements and it is easy to calibrate. The selection of the control strategies among those that meet the functional requirements (1) is done comparing their implementation costs, as discussed in Section 3.4.

Figure 4 reports the results obtained using the tool described in [1] to evaluate the feasible values for the lower and upper bounds of a saturation block employed in an air–to–fuel control algorithm for spark ignition engine control. The desired specification is to keep the mean–value of the air–to–fuel ratio close to the stoichiometric value. The plant model used in the validation scheme of Figure 3 represents, in an abstract and yet conservative way, the behavior of the fuel injection controlled system. The model is hybrid and contains uncertainties to achieve conservativeness.

3.3.2 Plant modeling

In this section, the three steps of the plant modeling phase, namely model development, identification, and validation, are illustrated. Aspects relevant to the introduction of hybrid system modeling techniques in automotive applications are outlined.

a) Model development

Traditionally, control engineers adopt mean–value models to represent the behavior of automotive subsystems. However, the need for hybrid system formalisms to model the behavior of subsystems in automotive applications is apparent in many cases.

Let us consider for instance the nature of input and output signals for the internal combustion engine, and the fuel injection and spark ignition subsystems. As described in Table 1, such signals can be classified in four different classes, by considering either their discrete or continuous nature in the time and value domains.

Often models of automotive subsystems are highly nonlinear. In engine modeling for instance, nonlinearities arise from fluid–dynamics and thermodynamics phenomena (e.g. volumetric efficiency, engine torque, emissions). Such nonlinearities are usually represented by piece–wise affine maps, identified by steady state measurements. In addition, since the mechanical and electromechanical components used in the automotive industry are characterized by a high production diversity and are greatly affected by aging, then controller design and validation has to be based on uncertain models of the plant that capture the main effects of variability in the plant. Uncertainties can be represented as: bounded either constant or time-varying perturbations of parameters, bounded signals and cross-coupling dynamics.

To conclude this brief discussion, we mention the increasing importance of human factors in automotive control design. Since the closed–loop system is a man–in–the–loop system, to design and validate correctly control algorithms that interact with the driver, it is necessary to understand and model the behavior of the driver. The need of modeling the human operator will become increasingly important with the expansion of x–by–wire applications (driving, steering, braking) that will require careful design of man–machine interfaces. The models of the driver must include perception (regarding also passengers as far as comfort is concerned), actuation, open-loop and closed-loop control actions. Moreover, since performance and driveability of the car are assessed

Internal Combustion Engine Hybrid Model

	discrete time		*continuous time*	
	discrete value	*continuous value*	*discrete value*	*continuous value*
inputs	spark ignition	injected fuel air charge exhaust gas conc.		engine speed
outputs	crankshaft events	air-to-fuel ratio		engine torque engine temperature engine exhaust gas

Direct Injection Fuel System Hybrid Model

	discrete time		*continuous time*	
	discrete value	*continuous value*	*discrete value*	*continuous value*
inputs		pressure valve cmd	injection signal	
outputs		injected fuel		rail pressure fuel temperature

Spark Ignition Hybrid Model

	discrete time		*continuous time*	
	discrete value	*continuous value*	*discrete value*	*continuous value*
inputs	spark command		ignition coil cmd	
outputs	spark ignition			

Table 1: Time domain and value domain classification of signals for internal combustion engine modeling.

by experienced test drivers that quote on a scale from 1 to 10 the driving feeling, then driver's models can be used to obtain analytical and repeatable specifications for driveability controllers design. The development of models for car driveability perception requires intensive study of human perception and assessment criteria, different vehicles and different drivers, driver interviews (during and after driving), data recording and analysis.

In conclusion, plant models development requires extensive use of hybrid modeling techniques:

- hybrid deterministic and stochastic formalisms, including FSM, DES, DT, CT, PDA, for representing interacting behaviors of different nature are essential;
- such hybrid formalisms should be supported by appropriate tools for hybrid model description and simulation.

Finally, to illustrate the relevance of hybrid modeling in automotive applications, we briefly describes a hybrid model for spark ignition engine and an automotive driveline.

Spark ignition 4-stroke engine. An accurate model of a spark ignition 4-stroke engine has a natural hybrid representation because the cylinders have four modes of operation corresponding to the stroke they are in, while driveline and air dynamics are continuous-time processes. In addition, these processes interact tightly. In fact, the timing of the transitions between two phases of the cylinders is determined by the continuous motion of the driveline, which, in turn, depends on the torque produced by each piston.

More in detail, consider the hybrid model of the torque generation process and the driveline presented in [6]. For a given gear selection and clutch position, the driveline is described by a continuous time system whose state includes the driveline torsion angle, the crankshaft revolution speed, and the wheel revolution speed. The inputs of the model are the torque T produced by the engine and the wheel torque T_w.

The engine torque T is given by $\Sigma_{i=1}^{N} T^i$, where T^i is the torque generated by each piston at each cycle. The profile of T^i is determined by the phases of the cylinder, the piston position, the mass of air and the mass of fuel loaded in the cylinder during the intake phase, and on the spark ignition timing.

The 4-stroke engine cycle can be modeled by means of a finite state machine (FSM) capturing the sequential nature of the behavior of the cylinders. In fact, each cylinder cycles through the following four phases:

- *intake* (I): the piston goes down from the *Top Dead Center* (TDC) to the (*Bottom Dead Center* (BDC)) loading the air–fuel mix present in the intake manifold;
- *compression* (C): the trapped mix is compressed by the piston during its upward movement from the BDC to the TDC;
- *expansion* (E): the combustion takes place pushing down the piston from the TDC to the BDC;
- *exhaust* (H):during its upward movement, from the BDC to the TDC, the piston expels combustion exhaust gases.

However, for spark ignition engines, the torque generated by each piston is related not only to the phase of the cylinder and the air and fuel charge, but also to the spark generation process. Intuitively, spark ignition should occur exactly when the piston reaches the TDC of the compression stroke. Since the combustion process takes non-zero time to complete, then the pressure in the cylinder reaches its maximum some time after spark ignition. As a consequence, in order to achieve maximum fuel efficiency, it is convenient to produce the spark before the piston completes the compression stroke (*positive spark advance*). On the other hand, producing a spark after the piston has completed the compression phase and is in the expansion stroke (*negative spark advance*) may be used to reduce drastically (and much faster than using only the throttle valve) the value of the torque generated during the expansion run. Since spark ignition may occur either during the compression stroke or during the expansion stroke, a six state FSM is needed to model the possible behaviors of the cylinder. The cylinder FSM is shown in Figure 5. The FSM state takes one of the following values

- I, denoting *Intake*;
- BS, denoting *Before Spark*: the piston is in the compression stroke and no spark has been ignited yet;
- PA, denoting *Positive Advance*: the piston is in the compression stroke and the spark has been ignited;
- NA, denoting *Negative Advance*: the piston is in the expansion stroke and the spark has not been ignited yet;

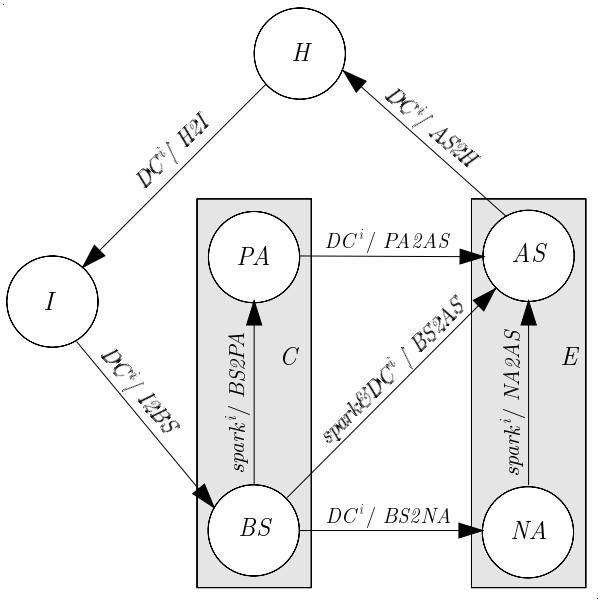

Figure 5: FSM describing the behavior of the i–th cylinder.

- AS, denoting *After Spark*: the piston is in the expansion stroke and the spark has been ignited;
- H, denoting *Exhaust*.

The cylinder FSM changes state either when a spark is given or when a dead center is reached. This last event depends on the continuous motion of the driveline and more precisely on the crankshaft angle, which defines the position of the piston. In turn, the crankshaft revolution speed depends on the torque T produced by the engine.

Finally, the torque produced by the cylinder depends on the air-fuel mixture loaded during the intake stroke. Since the air-fuel mixture is loaded in the cylinder during the intake stroke while the torque generation starts after the spark is ignited, then there is a delay between the time at which the mixture is loaded and the time at which the corresponding active torque is generated. This delay can be modeled by means of a DES synchronized with the FSM transitions. The overall model of the torque generation process for a single cylinder consists then of four communicating sub–models:

- an FSM, modeling the 4-stroke engine cycle and the spark generation process,
- a DES, modeling the discrete delay on the active torque generation, and
- two continuous time systems, modeling respectively the air intake process and the profile of the generated torque.

Driveline. A second very interesting automotive subsystem rich of discrete–continuous interactions is the driveline (see [8]). An accurate model of the driveline has a natural hybrid representation

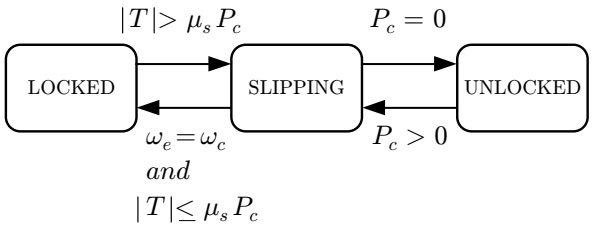

Figure 6: The hybrid model of the clutch.

because of the discontinuities due to clutch and the gear on the continuous motion of the driveline. In fact, the clutch can be modeled as a hybrid system with three discrete states: *Locked*, *Slipping*, and *Unlocked*. In Figure 6 the FSM of the hybrid model of the clutch is depicted. When the clutch is *Locked* the clutch plate and the flywheel are rigidly connected by static friction, so that their inertias are collected in a single first–order dynamics. The highest coupling torque T_{max} before incurring in clutch slipping corresponds to the maximum static friction torque, which is a function of the pressure P_c between the clutch plate and the flywheel , i.e. $T_{max} = \mu_s P_c$. When the transmitted torque T exceeds T_{max}, the system enters the state *Slipping*: the clutch plate and the flywheel are no longer strictly connected but they slip one on the other. In this case the coupling torque is due to dynamical friction and it is a function of the sliding speed. If the transmitted torque decreases, then the clutch returns in the state *Locked*, while if $P_c = 0$ the clutch enters the state *Unlocked* and $T_c = 0$. In the state *Unlocked* the crankshaft is completely decoupled from the rest of the driveline and the two systems follow independent dynamics.

For some applications, e.g. actual engaged gear identification [9], it is useful to collect in a single state the clutch *Unlocked* and *Slipping* states. In such cases, the overall system can be described by a hybrid system with 7 discrete states and four continuous state variables. The FSM describing the discrete dynamics of the model is depicted in Figure 7. The discrete state q_i, for $i = 1, \ldots, 5$, correspond to i–th gear engaged and clutch locked; location q_{RG} models reverse gear engaged; location q_N represents either driveline open (idle gear and/or clutch open) or clutch slipping. The continuous state variables are: the driveline torsion angle α, the crankshaft revolution speed ω_e, the clutch plate revolution speed ω_c, and the wheel revolution speed ω_w. When the clutch is locked, then $\omega_e = \omega_c$ so that the continuous behavior of the driveline can be described by a third order linear system with parameters depending on the selected gear.

The hybrid model has as inputs the position of the gear lever $lever \in \{1, 2, 3, 4, 5, RG, N\}$ and the torque generated by the engine while the connection pressure of the clutch plates P_c and the load wheel torque T_w are considered as disturbances. A more detailed driveline hybrid model, with 6048 discrete state combinations and 12 continuous state variables, is presented in [8]. In addition to the clutch and the gear, the proposed hybrid model describes the discontinuities in the driveline due to engine suspension, elastic torsional characteristic, tires, frictions and backlashes. This very detailed driveline hybrid model exhibits a behavior very close to the physical driveline and has been developed to be used for control algorithm validation.

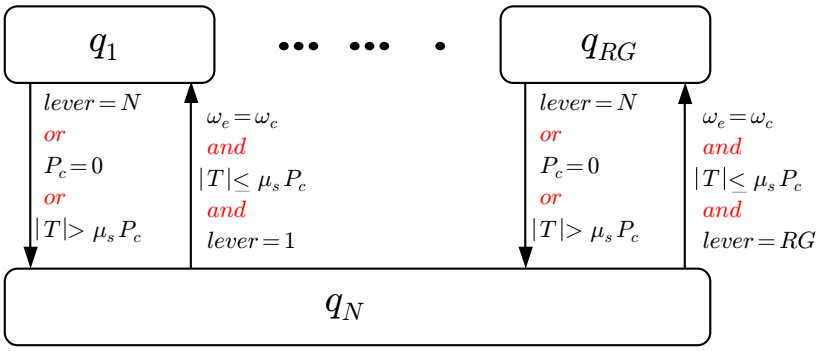

Figure 7: The hybrid model of the driveline.

b) Identification

In current practice, parameter identification is mostly based on steady–state measurements, obtained using either manually defined set–points or automatic on–line screening. Dynamic parameters are often either obtained analytically or from step responses. However, step response and other classical identification methods can be used to identify models representing standard continuous evolutions only, such as those exhibited by mean–value models. When applied to hybrid models, classical techniques can only be used to identify the plant model separately in each discrete mode. They hardly succeed in identifying parameters related to switching conditions and cannot be applied to black–box hybrid model identification.

The availability of hybrid system identification techniques using transient data, including mode switching, would allow to increase identification accuracy, reduce the amount of experimental data needed and identify all parameters in hybrid models. Efficient identification techniques for hybrid systems will also give the opportunity for modeling more complex hybrid behaviors that are currently abstracted due to the difficulties in the identification process.

Moreover, efficient hybrid techniques for the representation and identification of nonlinearities, as either piece–wise affine functions (see [16]) or piece–wise polynomial functions, would produce major impacts in the design:

- domain partition could be optimized (possibly not grid-based), achieving increased accuracy and reducing model complexity;
- parameter identification accuracy could be improved;
- high dimension nonlinearities $R^p \rightarrow R$ with $p \geq 3$, which are today represented as product of $R, R^2 \rightarrow R$ functions, could be represented and identified.

c) Validation

Model validation is the converse to identification: given a system model, the objective is to assess whether the model is consistent with experimental observations. No assumptions are made about the nature of the physical system, but the compliance of the model with the actual behavior of the system is evaluated using experimental data. As discussed above, plant models often include

unknown, bounded perturbations and unknown, bounded input signals to take into account the high uncertainty in the behavior of the components. The validation of uncertain models is a very critical task that has been studied in the literature in the case of continuous time systems, see [37, 33, 36]. Techniques have been proposed for explicit calculation of whether sufficient data for invalidation of the model has been obtained. These techniques can be used in automotive applications to assess the richness of validation patterns for the continuous evolutions of the plant.

However, established methodologies to address model validation for hybrid systems are not available yet. A critical issue is the selection of rich enough test patterns for hybrid model validation. This topic is further discussed in Section 3.3.3.c, where automatic test pattern generation for controller validation is analyzed. Some open problems related to validation of hybrid models are:

- automatic generation of validation patterns;
- assessment of the richness of validation patterns. This problem can be formalized in the framework of reachability analysis for hybrid systems. Interesting approaches have been proposed using the concepts of structural coverage and data coverage.

3.3.3 Controller synthesis

In this section, the activities related to controller synthesis are presented by discussing the three steps of: plant and specifications analysis, algorithm development, and controller validation.

a) Plant and specifications analysis

Typically, before proceeding to the actual design of a control algorithm for a new application, some experimental data on a prototype of the system to be controlled are obtained using either open–loop control or some very elementary closed–loop algorithm. Open loop simulation of the plant model is also very useful in this phase. The plant model often represents a partially controlled plant and contains the effect of some inner-loop controllers. Open loop simulation of hybrid models requires the definition of discrete time, event-based and continuous time input actions, representing either hybrid inputs and references or perturbations. The assessment of classical structural properties, such as reachability [27], controllability [31], observability [23], stabilizability [21, 32], passivity [15], etc., on the plant model is of interest in this phase. In addition, quantitative analysis is very useful to understand the easy and the critical objectives of the design. It is interesting to obtain by performance and perturbations/uncertainties analysis an evaluation of quantities such as stability margins, most critical perturbations/uncertainties, robust stability margins, reachability and observability measures in the state space. Unfortunately, hybrid system theory is not mature enough for model analysis:

- some fundamental properties have not been formally defined yet;
- tests are often not available for verifying most of the properties;
- efficient implementation of tests will be necessary for automatic evaluation, since manual testing of hybrid system properties is often prohibitively complex;
- analysis tools have be integrated with standard system engineering tools, so to be able to process directly the models.

***b*) Algorithm development**

Control algorithms are often characterized by many operation modes, that are conceived to cover the entire life–time of the product: starting from in–factory operations before car installation, configuration, first power–on, power–on, functioning, power–off, connection to diagnostic tools, and so on. During normal functioning, control strategies can be either in one of the nominal operation modes or in some recovery mode. A significant number of algorithms are dedicated to the computation of switching conditions between modes and controller initializations.

A short and by no–means exhaustive list of control actions for which hybrid system design is particularly interesting is as follows: fuel injection, spark ignition, throttle valve control (especially with stepper motor), electromechanical intake/exhaust valve control, engine start-up and stroke detection, crankshaft sensor management, VGT and EGR actuation (hysteresis management), emission control (cold start-up, lambda on/off sensor feedback), longitudinal oscillations control (backlash and elasticity discontinuities), gear–box control (servo-actuation in traditional gear shift systems), cruise control and adaptive cruise control, diagnosis algorithms (signals and functionalities on-line monitoring), algorithms for fault-tolerance, safety and recovery (degraded mode activation).

Diagnostic algorithms represent a large part of the strategies implemented in automotive ECUs. For engine control, the implementation of diagnosis algorithms is enforced by legislation: OBDII (On Board Diagnosis II) in USA and EOBD (European On Board Diagnosis) in EU. In general, these requirements specify that every fault, malfunction or simple component degradation that leads to pollutant emissions over given thresholds should be diagnosed and signaled to the driver. This requirement has a significant impact on ECU design, since it implies the development of many on–line diagnostic algorithms [20].

Both specifications and accurate models of the plant are often hybrid in automotive applications but the methodology currently adopted for algorithm development is rather crude and can be summarized as follows. The continuous functionalities to be implemented in the controller are designed based on mean–value models of the plant, with some *ad hoc* solutions to manage hybrid system issues (such as synchronization with event–based behaviors); if the resulting behavior is not satisfactory under some specific conditions, then the controller is modified to detect critical behaviors and operate consequently (introducing further control switching). The discrete functionalities of the controller are designed by direct implementation of non–formalized specifications. Design methodologies and corresponding tools for the synthesis of discrete event systems are usually not employed. The discrete behavior of the controller is not obtained by automatic synthesis of a formalized specification, as for instance it is done in hardware design. If the algorithm is not designed from scratch, but is obtained by elaborating existing solutions, as it is the case in the derivative design approach, then additional operation modes may be introduced to comply with the new specification. This results in a non–optimized controller structure. Structured approaches to the integrated design of the controller that allow to satisfy hybrid specifications considering hybrid models of the plant are not adopted as yet even though they have obvious advantages over the heuristics that permeate the present approaches.

Hybrid system techniques can significantly contribute to the improvement of control algorithm design in automotive applications. The introduction of hybrid synthesis techniques should be aimed at:

- shortening the algorithm development time;

- reducing testing effort;
- reducing calibration parameters and provide automatic calibration techniques;
- improving closed–loop performances;
- guaranteeing correct closed–loop behavior and reliability;
- achieving and guaranteeing desired robustness;
- reducing implementation cost.

Most of the analytical approaches proposed so far for controller design using hybrid system techniques are quite complex. Usually, the application of these techniques requires designers that have a deep understanding of hybrid systems and necessitates long development times. As a consequence, the introduction in the automotive industry of hybrid system design methodologies often results too much expensive. Hence, to overcome this problem and make them profitable for the industry it is essential that the methodologies be supported by efficient tools that allow fast and easy application of hybrid design. Hybrid model predictive control is a good example in hybrid system research where the development of a design methodology was supported by successful efforts in design tool development [17].

We conclude this section by presenting the hybrid design of two algorithms for engine control.

Cut-off control. A quite critical driveability objective is the control of longitudinal oscillations of the car when fast engine torque variations are requested by the driver (tip-in and tip-out). Roughly speaking, the control consists of active damping of powertrain oscillations. The problem is particularly challenging when the engine is not equipped with electronic throttle valve, since in this case only fuel injection and spark ignition controls can be used for engine torque modulation to achieve the desired damping of the oscillations. Most of the proposed approaches are based on mean–value continuous-time models of the torque generation. As a consequence, since the torque generation process has a discrete behavior, the implementation of such control strategies on a real engine may result in very poor closed–loop performance and may give rise to unpredicted unpleasant behaviors. On the contrary, a design based on a hybrid model of the engine allows to develop control laws for which closed–loop performances are guaranteed.

A hybrid approach to the design of a longitudinal oscillation damping control during tip–out was presented in [12]. The control problem arises when the driver, by releasing the fuel pedal, requests no torque to the engine. In this case, an obvious strategy to minimize fuel consumption and emissions is to shut fuel injection, an operation called cut–off. However, cutting off fuel injection as soon as the gas pedal is released, causes a sudden torque reduction that may result in unpleasant oscillations compromising driving comfort. A more complex control action involves modulating the engine torque from the present value to the value corresponding to cut–off in an attempt to prevent oscillations. This control policy is implemented by slowing down air flow decay and, when air quantity is below a threshold, reducing fuel injection gradually to zero. As it is often the case, heuristic rule–based controls need extensive tuning, yield satisfactory solutions only in a limited range of operations and are hardly optimal with respect to the emissions and fuel consumption. In particular, if air reduction is too slow, when the driver releases the gas pedal and presses the clutch pedal to change gear, engine speed raises for a while, thus causing a definite reduction in passengers' comfort. Moreover, if air and/or fuel reduction is too fast, oscillations take place anyway.

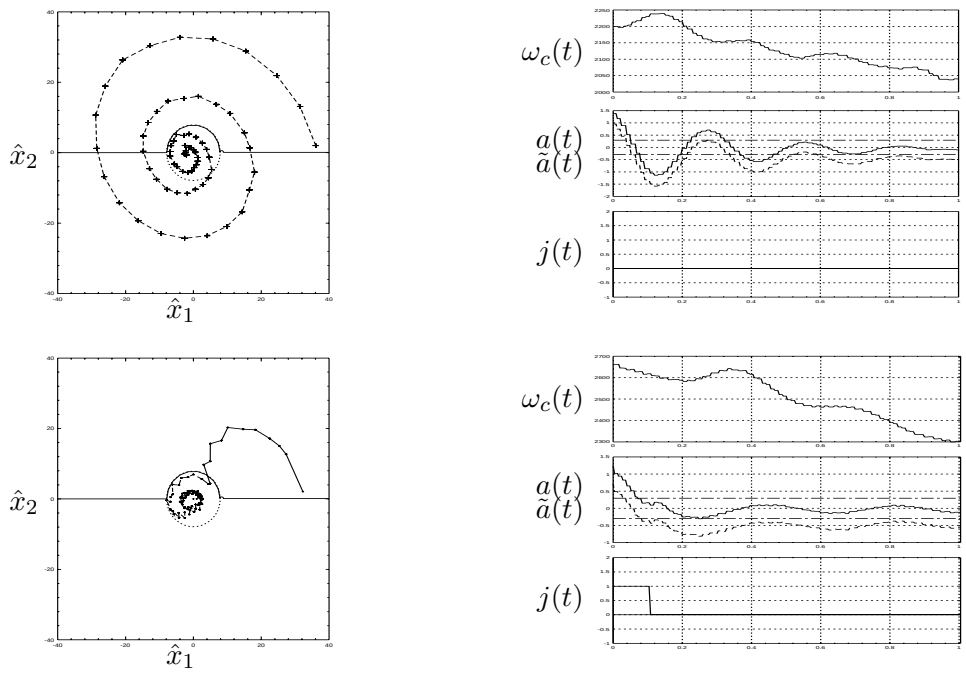

Figure 8: Evolutions of the oscillating modes to the target set $B_{\hat{\rho}}$ (left) and engine speed, accelerations and injection signal profiles (right), in an uncontrolled cut-off (top) and with the proposed hybrid control strategy (bottom).

The hybrid control algorithm presented in [12] is able to steer the evolution of the system to the fuel cut–off condition, minimizing the amplitude of the undesired oscillations. Since a hybrid model of the engine has been considered during the design, the algorithm acts on fuel injection and spark ignition once per engine cycle for each cylinder taking into account synchronization and actuators' delay. The hybrid approach adopted for synthesis of the cut–off control algorithm guarantees the correctness of the behavior when applied to the real plant. The proposed cut–off control strategy was tested at Magneti-Marelli Engine Control Division on a commercial car, a 16 valve 1400 cc engine car. The experiment was carried out driving the car in the test ring and measuring the important parameters and variables that determine the performance of the control strategy. In Figures 8 the performance achieved by the proposed hybrid cut-off strategy are compared with an instantaneous uncontrolled cut-off operation. On the left the evolution of the oscillating modes $\hat{x}$ are reported along with the switching curve that defines the regions where fuel is injected and it is shut off. The effectiveness of the proposed controller is apparent from the evolution of the vehicle acceleration and engine speed reported on the right.

Actual engaged gear identification. As a second example, consider the problem of on–line identification of the actual engaged gear. Engine control strategies achieving high performance and efficient emissions control depend critically on such identification algorithm. In fact, the knowledge of the actual engaged gear is necessary in engine torque control to compensate the equivalent inertia of the vehicle on the crankshaft and, for Diesel engines, it is very important to improve emissions

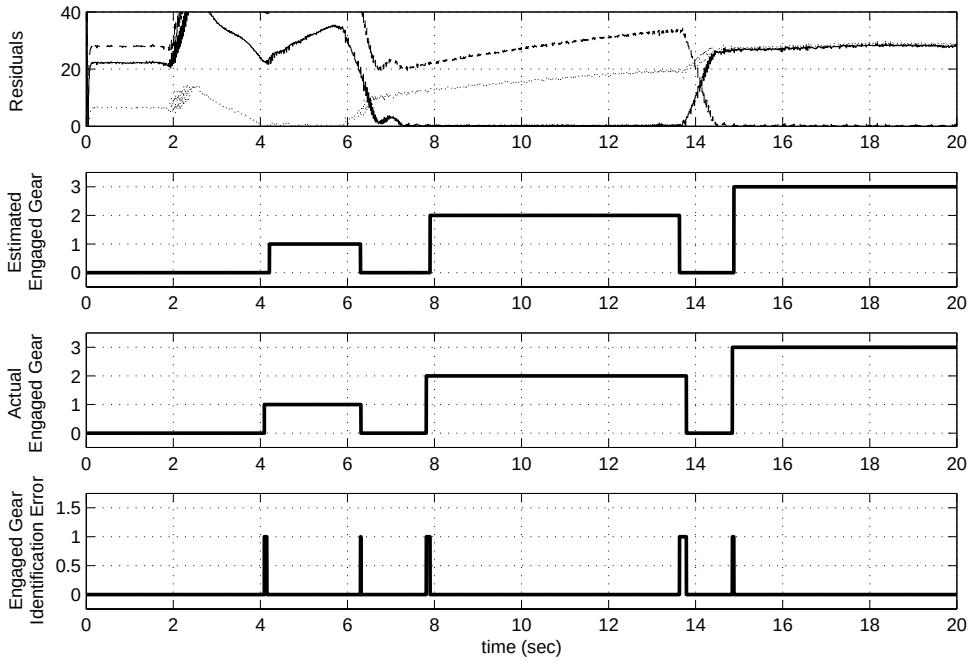

Figure 9: Gear identification with experimental data: the maneuver starts with car at rest, clutch open and first gear engaged ($q = q_N = 0$). After a clutch slipping phase ($q = q_N = 0$), the clutch is locked ($q = q_1 = 1$); later, second gear ($q = q_2 = 2$) and then third gear ($q = q_3 = 3$) are engaged, passing through idle and slipping ($q = q_N = 0$).

control. In cars equipped with manual gear shift, this information is not directly available and, at present, it is deduced by comparing the revolution speed of the wheels with the revolution speed of the crankshaft. However, since both of them are affected by oscillations due to the elasticity of the driveline and the tires, then this approach implies large time delays in the identification and may produce significant identification errors.

The identification algorithm presented in [9] is based on a hybrid model of the driveline (see the driveline model presented in Section 3.3.2.a) where the engaged gear and connection clutch state are represented as discrete states. The design problem is then formulated as the identification of the discrete state of the driveline hybrid model and an on-line identification algorithm is obtained by applying the methodology for hybrid observer design proposed in [7]. The algorithm is able to detect a change in the continuous time dynamics and identify the discrete state of the driveline hybrid model by processing the crankshaft and wheel revolution speed measurements and an estimate of the engine torque mean–value. The on–line identification is achieved by the generation of appropriate residual signals, one for each gear, which vanish when the corresponding gear is engaged and the clutch is locked. Figure 9 reports the results on actual engaged gear identification obtained in Magneti Marelli Powertrain using an Opel Astra equipped with a Diesel engine and a robotized gearbox SeleSpeed. For the validation of the identification algorithm, the estimated engaged gear is compared to the signal on actual engaged gear provided by the control unit of the robotized gearbox. The algorithm was tested on several maneuvers for a total of 250 gear engagements and it was able

to identify correctly the actual engaged gear within 250 *msec*, as requested by specification. It proved to be remarkably robust with respect to parameter uncertainties (e.g. vehicle inertia) and time-varying unknown disturbances (e.g. wheel torque and road slope).

c) Controller validation

In the automotive industry, control algorithms are validated by extensive and time-consuming — hence expensive — simulations of the closed–loop models. The designers, based on their experience, devise critical trajectories to test the behavior of the closed–loop system in the perceived worst–case conditions, in addition to the tests on the critical maneuvers that may be provided by customer requirements. Furthermore, a rough investigation on the robustness properties of control algorithms is obtained by screening the most critical parameters and uncertainties and applying critical perturbations. In the current design flow, there is no automatic approach to the validation of performance specifications. Some approaches for automatic test patterns generation are under investigation. To the best of our knowledge, there is no tool available in the market for performance analysis, robust stability, and formal verification for both continuous and discrete specification.

Due to complexity of the plant–controller interactions, the non–negligible effects of the implementation, the large uncertainties in the plant given by product diversity and aging, validation of control algorithms is one of the hottest topics in automotive industry. Today, the quality of the validation step is not satisfactory and important improvements in validation will be necessary to cope with the safety issues that will be raised by next generation x–by–wire systems. Ideally, validation and formal verification should be completely automatic. Hybrid system techniques can contribute significantly to the improvement of the validation process:

- Validation has to be supported by tools for
  - efficient simulations of hybrid closed–loop models;
  - stability and performance analysis;
  - robust stability and robust performance analysis;
  - invariant set and robust invariant set computations.
- Methodologies and tools should be developed for
  - automatic validation against formalized hybrid performance specifications;
  - automatic validation of safety relevant conditions;
  - automatic optimized test patterns generation reaching specified level of coverage.
- Interesting validation problems are related to the computation of conservative approximations for the largest sets of
  - parameter uncertainties,
  - calibration parameters,
  - implementation parameters (e.g. sampling–period, latency, jitter, computation precision, etc.),

  for which the desired performances are achieved.

- Some classes of algorithms that require intensive and complex validation are
  - diagnosis algorithms;
  - safety critical algorithms;
  - algorithms preventing the system to stall (e.g. idle speed control).

3.4 Hardware/Software components

The design of the HW/SW implementation of ECUs in the automotive industry is achieved using the most advanced methodologies for hardware and software development. An accurate design is necessary to be able to meet the strict requirements imposed on the implementation of safety critical real–time systems and achieve at the same time the very tight cost targets. HW/SW implementation of the control algorithms may offer an interesting and little explored application of hybrid formalisms as a more rigorous design approach is advocated for reducing implementation errors. In particular, hybrid methodologies could be exploited for the formalization and definition of the specification for HW/SW implementation of control algorithms. In fact, currently available methodologies and tools in the control domain and the HW and SW implementation domains are often not well integrated; this situation is a frequent cause of design errors. The specification for the HW/SW implementation must include all details necessary for a correct implementation of the algorithms, that is it must provide:

- complete description of the algorithm;
- specification of the computation accuracy;
  - in the value domain: precision for each computation chain (for fixed–point arithmetic implementation), threshold detection bounds, etc.;
  - in the time domain: bounds for latency, jitter, delay in event detection, etc.
- execution order and synchronization;
- priorities in case of resource sharing (CPU, communication, etc);
- communication specifications;
- data storage requirements (e.g. variables to be stored in EEPROM).

In addition, the specification for the HW/SW implementation should be derived from executable models, according to the model–based design approach. These models should also be integrated with tools for automatic code generation [18]. Finally, the specification for the HW/SW implementation should ideally provide executable acceptance tests that can guarantee that the computation accuracy obtained in the HW/SW implementation is compliant with the requirements. In particular,

- Tools suitable for the description of the implementation requirements of the algorithms have to:
  - support the specification of the algorithm behavior, the computation accuracy and the other implementation requirements and constraints mentioned above;

  - support description of implementation acceptance tests;
  - be efficiently integrated with software and hardware development tools and tools for automatic code generation.

- Methodologies and tools for defining and validating implementation constraints should be developed:
  - the degradation of the execution of control algorithms due to the implementation on bounded resource platforms has to be exported and modeled at the control system level to obtain constraints for the implementation;
  - these constraints should be formally specified in the HW/SW implementation requirements along with executable acceptance tests;
  - tools should support the validation of the HW/SW implementation by running the acceptance tests.

It is in the analysis of the effects of implementation on the behavior of the control algorithms, in the construction of abstract models of the implementation platform and in the constraint propagation that we see great value in hybrid technology.

An integrated control-implementation design methodology has been presented in [10] to bridge the gap between functional design and implementation of embedded control systems. According to the proposed methodology, the designer evaluates how closed–loop performance and robustness of a given control algorithm are affected by the implementation, which is represented in an abstract form. The proposed design methodology is based on the principles of *platform-based design* described in [35] and applied to the design of automotive control systems in [2]. The application of the methodology presented in [10] to the design of an engine control unit (ECU) for motorcycles was first described in [11]. Such approach is further refined in [1], where a prototype tool that supports exploration of possible candidate implementation is also presented.

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 issues in this respect. Promising approaches regarding management of time domain perturbations due to the implementation, such as latency and jitter, have been presented in [34, 19]. The main effects of a particular implementation on the behavior of the controlled system must be carefully classified and characterized.

The approach presented below is an integration of the methodology supporting derivative design described in Section 3.3.1. Schematically, they can be represented in terms of perturbations on the controller input/output channels, as illustrated in Fig. 10. 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 $s \in \{1, \ldots, S\}$;

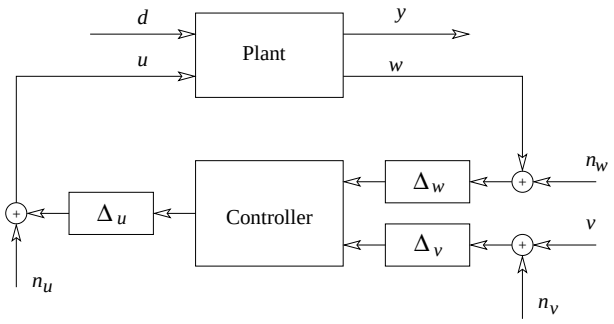

Figure 10: Abstract representation of the effects of implementation non–idealities.

- a set of platform constraints[2]

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

For a given platform structure $s \in \{1, \dots, S\}$, elements $p \in X_P^s$ are referred to as the *platform parameters*. In the control parameters and platform parameters product space, feasible mappings are given by the set

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

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).

The best implementation platform, among those described by parameters inside the set U in (4), can be selected by introducing a suitable objective function, representing an estimation of the implementation cost to be minimized. It is often the case that the cost model depends only on a subset of the parameters. The parameters belonging to the orthogonal space can be abstracted away. A very powerful method for the abstraction of non–relevant parameters is the use of formulas with existence and universal quantifiers.

As an example, consider a PI controller with gains K_P and K_I designed to control the crankshaft speed in engine control, with nominal inertia J and inertial uncertainty δ_J. Parameters K_P, K_I, J, and δ_J can be abstracted using the following formula:

$$\forall J \; \exists \; K_P, K_I \; \forall \; \delta_J \quad (\hat{c}, J, K_P, K_I, \delta_J, \hat{p}) \in U \ , \tag{5}$$

where $\hat{c}$ and $\hat{p}$ are the control and platform parameters interesting for the exploration of the possible implementation platforms. Then, by applying quantifier elimination to (5), the exploration is reduced to the *cost parameter* subspace $(\hat{c}, \hat{p})$.

The exploration of the parameter space is performed by using randomized algorithms and hybrid system techniques. More precisely, performance criteria are tested for the control algorithms over parameters spaces both via randomized algorithms, i.e. letting the parameters vary according to

[2]Typically, the schedulability and latency constraints are added.

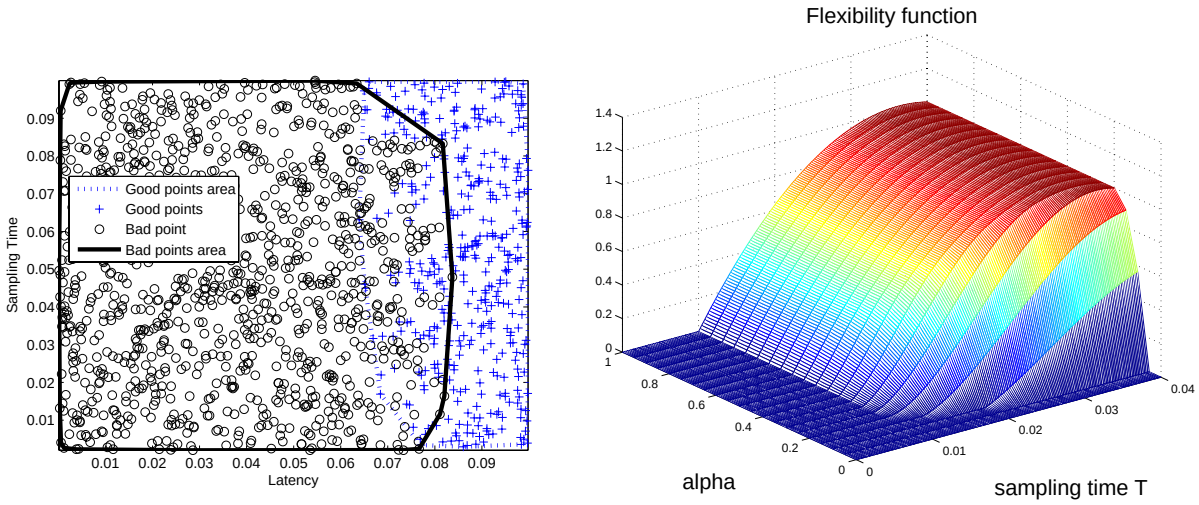

Figure 11: Implementation parameters projection (left). The flexibility function (right).

a given probability distributions, and stochastic algorithms, i.e. letting the parameters evolve in time according to a random coefficient stochastic differential equation.

By performing a random screening of the parameter space, each sample is labeled either *good* or *bad*, according to fulfillment of the closed-loop specification. Then, the convex polyhedra $\mathcal{P}_{good}$ and $\mathcal{P}_{bad}$ are produced by computing the convex hull of the subsets of *good* and *bad* samples, respectively. The subset of parameter values for which the specification is assumed to hold is given by the (non–convex, in general) polyhedron $\mathcal{P}_{good} \setminus \mathcal{P}_{bad}$. See e.g. Figure 11. Subsequently, the cost parameter subspaces of all control algorithms to be implemented in the electronic control unit are composed and the set $\mathcal{P}_{ecu}$ is defined by composing the sets $\mathcal{P}_{good} \setminus \mathcal{P}_{bad}$ obtained for each control algorithm.

Hence, a cost model $H(\hat{c}, \hat{p})$ is defined in the overall cost parameter space. In [10], the use of *flexibility* functions as cost models was proposed. The underlining idea is to guide the exploration towards parameters that can be easily achieved by platforms at lower levels of abstraction. In this way, the risk of expensive design cycles that span several platforms is minimized and a better platform choice is offered while approaching the implementation level. 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 can be an increasing function of the sampling time and the latency as depicted in Figure 11. In fact, 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. The point corresponding to the minimum cost, according to the cost model $H(\hat{c}, \hat{p})$ in the set $\mathcal{P}_{ecu}$ is obtained by an optimization algorithm. Finally, optimal values – maximizing robustness – for the remaining parameters of each control

algorithm abstracted away by the quantifier elimination are selected.

4 Conclusions

In this paper, we analyzed in details the design flow for electronic control systems in the automotive industry, with the purpose of identifying bottle–neck problems that either cause inefficiency in the development process or limit the design. We pointed out that the development and application of hybrid system techniques could be very profitable for the solution of some of these problems and could significantly contribute to the improvement of the design quality.

The most obvious applications for hybrid system techniques regard plant modeling and control algorithm design. But the potential impacts of hybrid systems should not be considered limited to these domains. For instance, they can successfully be applied to address problems at the boundary between control algorithm design and hw/sw implementation, since they allow the designer to capture the effects of limited resources and physical constraints on the performance of the controlled system and check the correctness of the design. An additional promising application of hybrid system techniques is in system design, today an extremely critical step. Hybrid formalisms can be applied to represent system specifications and deploy them in an architecture of control algorithms and requirements.

However, hybrid system techniques at the current state–of–the–art present clear limitations for their efficient introduction in the automotive industry design flow. A coherent set of methodologies, supported by efficient tools, is not available yet, the proposed approaches are not integrated within the overall development flow and, apart from few exceptions in some companies, control engineers are not trained in hybrid system techniques.

5 Acknowledgments

We wish to thank Alberto Ferrari and Pierpaolo Murrieri from PARADES; Gabriele Serra, Giacomo Gentile and Walter Nesci, from Magneti Marelli Powertrain (Bologna, I); Paolo Ferracin from CNH (Modena, I); Gilberto Burgio from Ford Research Center (Aachen, G) for the many discussions on the topic. This work has been partially supported by the CC (Control and Computation) E.U. Project (grant FP5-IST-2001-33520) and the HYCON E.U. Network of Excellence (grant FP6-IST-511368).

References

[1] A. Agostini, A. Balluchi, A. Bicchi, B. Piccoli, A. L. Sangiovanni-Vincentelli, and K. Zadarnowska. Randomized algorithms for platform–based design. Submitted to 44th IEEE Conference on Decision and Control, 2005.

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

[3] AUTOSAR. www.autosar.org.

[4] E.-H. Azibi and J.-C. Sardas. Computer-aided process engineering and transformation of the process design activity in automotive industry. In *Proc. of the 2002 IEEE International Conference on Systems, Man and Cybernetics*, October 2002.

[5] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, U. Freund, E. Schlenker, and H.-J. Wolff. Correct-by-construction transformations across design environments for model-based embedded software development. In *Proc. of the Design Automation and Test in Europe Conference*, pages 1044–1049, March 2005.

[6] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, C. Pinello, and A. L. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems: Challenges and opportunities. *Proceedings of the IEEE*, 88, "Special Issue on Hybrid Systems" (invited paper)(7):888–912, July 2000.

[7] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. L. Sangiovanni-Vincentelli. Design of observers for hybrid systems. In C.J. Tomlin and J.R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 76–89. Springer-Verlag, Stanford, CA, 2002.

[8] A. Balluchi, L. Benvenuti, C. Lemma, P. Murrieri, and A. L. Sangiovanni-Vincentelli. Hybrid models of an automotive driveline. Tech. rep., PARADES, Rome, I, December 2004.

[9] A. Balluchi, L. Benvenuti, C. Lemma, A. L. Sangiovanni-Vincentelli, and G. Serra. Actual engaged gear identification: a hybrid observer approach. In *to be presented at 16th IFAC World Congress*, Prague (CZ), July 2005.

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

[11] A. Balluchi, M. D. Di Benedetto, A. Ferrari, G. Gaviani, G. Girasole, C. Grossi, W. Nesci, M. Pennese, and A. L. Sangiovanni-Vincentelli. Design of a motorcycle engine control unit using an integrated control-implementation approach. In *Proc. 1st IFAC Workshop on "Advances in Automatic Control"*, pages 218–225, Salerno, Italy, April 2004.

[12] A. Balluchi, M. D. Di Benedetto, C. Pinello, C. Rossi, and A. L. Sangiovanni-Vincentelli. Hybrid control in automotive applications: the cut-off control. *Automatica*, 35, Special Issue on Hybrid Systems:519–535, March 1999.

[13] M. Baotic, M. Vasak, M. Morari, and N. Peric. Hybrid theory based optimal control of electronic throttle. In *Proc. of the IEEE American Control Conference, ACC 2003*, pages 5209–5214, Denver, Colorado, USA, June 2003.

[14] M.B. Barron and W.F. Powers. The role of electronic controls for future automotive mechatronic systems. *IEEE/ASME Transactions on Mechatronics*, 1(1):80–88, March 1996.

[15] A. Bemporad, G. Bianchini, F. Brogi, and F. Barbagli. Passivity analysis and passification of discrete-time hybrid systems. 2005. Submitted.

[16] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Trans. Automatic Control*, 2004. Accepted for publication as a regular paper.

[17] A. Bemporad, M. Morari, and N. L. Ricker. *Model Predictive Control Toolbox for Matlab – User's Guide.* The Mathworks, Inc., 2004. http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/.

[18] K.R. Butts. An application of integrated casexacsd to automotive powertrain systems. In *Proc. of the 1996 IEEE International Symposium on Computer-Aided Control System Design*, pages 339–345, Dearborn, MI, September 1996.

[19] A. Cervin, J. Eker D. Henriksson, B. Lincoln, and K. E. Årzén. How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. *IEEE Control Systems Magazine*, 23(3):16–30, June 2003.

[20] J. Chen and R.J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems.* Number 3 in Series on Asian Studies in Computer and Information Science. Kluwer International, 1999.

[21] R.A. Decarlo, M.S. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–82, July 2000.

[22] ETAS. ASCET. http://www.etas.de.

[23] Jelel Ezzine and A.H. Haddad. Controllability and observability of hybrid systems. In *Proceedings of the 1988 American Control Conference*, pages 41–6, 1988.

[24] B.S. Heck, L.M. Wills, and G.J. Vachtsevanos. Software technology for implementing reusable, distributed control systems. *IEEE Control Systems Magazine*, pages 21–35, February 2003.

[25] H. Heinecke, K.-P. Schnelle, H. Fennel, J. Bortolazzi, L. Lundh, J. Leflour, J.-L. Mat/'e, K. Nishikawa, and T. Scharnhorst. Automotive open system architecture - an industry-wide initiative to manage the complexity of emerging automotive e/e-architectures. In *Proc. of Convergence 2004*, number 2004-21-0042, Detroit, MI, October 2004.

[26] K. Keutzer, S. Malik, R. Newton, J. Rabaey, and A.L. Sangiovanni-Vincentelli. System level design: Orthogonalization of concerns and platform–based design. *IEEE Transactions on Computer-Aided Design*, 2000.

[27] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3), March 1999.

[28] G. Martin. The future of high-level modelling and system level design: Some possible methodology scenarios. In *Proc. of 9th IEEE/DATC Electronic Design Processes Workshop (EDP2002)*, Monterey, CA, April 2002.

[29] G. Martin. Guest editor's introduction: The reuse of complex architectures. *IEEE Design and Test of Computers*, 19(6):4–5, November/December 2002.

[30] R. Mobus, M. Baotic, and M. Morari. Multi-object adaptive cruise control. In O. Maler and Eds. A. Pnueli, editors, *Hybrid Systems: Computation and Control, HSCC 2003*, volume 2623 of *Lecture Notes in Computer Science*, pages 359–374. Springer Verlag, 2003.

[31] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, controllability, observability. In *Hybrid Systems*, Lecture Notes in Computer Science 736, pages 317–356. Springer-Verlag, 1993.

[32] S. Pettersson and B. Lennartson. Lmi for stability and robustness of hybrid systems. In *Proceedings of 16th American Control Conference*, volume 3, pages 1714–18, Albuquerque, NM, USA, June 1997.

[33] K. Poolla, P. Khargonekar, A. Tikku, J. Krause, , and K. Nagpal. A time-domain approach to model validation. In *Proc. of the IEEE American Control Conference (ACC1992)*, pages 313–317, 1992.

[34] K. E. Årzén, A. Cervin, J. Eker, and L. Sha. An introduction to control and scheduling co-design. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

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

[36] R. Smith and G.E. Dullerud. Continuous-time control model validation using finite experimental data. *IEEE Trans. on Automatic Control*, 41(8):1094–1105, August 1996.

[37] R.S. Smith and J.C. Doyle. Model validation: A connection between robust control and identification. *IEEE Trans. on Automatic Control*, 1992.

[38] The MathWorks. Matlab, Simulink, Stateflow. http://www.mathworks.com.