Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Hybrid Systems in Automotive Electronics Design

Andrea Balluchi, Luca Benvenuti and Alberto L. Sangiovanni–Vincentelli

Abstract— **Automotive is certainly one of the most attractive and promising application domains for hybrid system techniques. Indeed, some hybrid models and algorithms have already been successfully applied for automotive control designs. However, 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.**

I. 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 powertrain, 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.

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.

Moreover, the design is subject to critical HW/SW implementation constraints on cost, reliability (safety and correctness), power consumption, weight and position.

Finally, the overall development process 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.

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.

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. The source of these problems is clearly the increased complexity of the embedded controllers but also the difficulty of the leading Original Equipment Manufacturers (OEMs) in managing the integration and maintenance process with embedded controllers that come from different suppliers (so–called Tier–1 companies) who use different design methods, different software architecture, different hardware platforms and different (and often proprietary) Real-Time Operating Systems (RTOS).

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. Hybrid systems techniques can have an important role with respect to the second point. Successful approaches to the design of control algorithms using hybrid system methodologies had been presented in the literature, e.g. cut-off control [6], intake throttle valve control [7], actual engaged gear identification [4], adaptive cruise control [12]. 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. Nonetheless, 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 paper, we analyze the design flow for embedded controllers in the automotive industry, with the purpose of identifying challenges and opportunities for hybrid system technologies.

In particular, in Section II, 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 III, 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.

This work has been carried out in the framework of the HYCON E.U. Network of Excellence (FP6-IST-511368) and has been partially supported by the CC E.U. Project (FP5-IST-2001-33520).

A. Balluchi is with PARADES, Via S. Pantaleo, 66, 00186 Roma, Italy, balluchi@parades.rm.cnr.it

L. Benvenuti is with PARADES and with the Computers and Systems Science Dept., Università di Roma "La Sapienza", Via Eudossiana 18, 00184 Roma, Italy, luca.benvenuti@uniroma1.it

A. L. Sangiovanni–Vincentelli is with PARADES and with the EECS Dept., Univ. of California at Berkeley, CA 94720, USA, alberto@eecs.berkeley.edu

0-7803-9568-9/05/$20.00 ©2005 IEEE

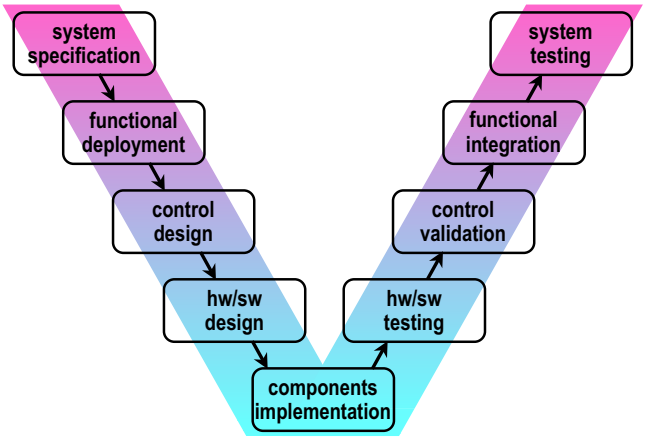

Fig. 1. Design and integration flow.

II. 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. The ECUs interact by asynchronous communication over a communication network specifically designed for automotive applications. 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 (see [1], [5]) and for evaluation of platform requirements at the early stages of the design flow. There is an increasing interest in the industrial community towards managing the complexity of the design and obtaining ECUs with guaranteed performances and reduced cost, by means of a model-based design approach. In this approach, specifications, functional architectures, algorithms, and implementation architectures are represented formally by models thus allowing, at least in principle, formal analysis and automatic synthesis.

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:

A. *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).

B. *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.

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

D. *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.

III. SYNTHESIS FLOW

In this section, we describe the synthesis part of the automotive design flow emphasizing the aspects where we believe hybrid system techniques may have an important impact.

A. System specification

System specifications define requirements on performance, driveability, fuel consumption, emissions and safety. They are given in terms of a number of operation modes characterized by different controlled variables and objectives 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.

The degree of detail given by the OEMs in describing system specifications is not uniform. Some behaviors may result only vaguely specified while some others may be very detailed so that the OEM imposes not only a system level requirement but also a particular solution to satisfy it.

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.

B. 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 [1] for more details). 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.

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.

C. 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 model and specifications analysis; *b*) algorithm development; *c*) controller validation.
3. Fast prototyping.

However, if part of the algorithms are re–used from previous designs, the entire three–step flow is often only partially performed.

In the following sections, the first two steps are discussed in details.

1.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 systems in automotive applications is apparent in many cases.

Let us consider for instance the torque generation and transmission processes of an internal combustion engine. An accurate model of the engine has a natural hybrid representation because the cylinders have four modes of operation corresponding to the stroke they are in (which can be represented by a finite-state model) while power–train 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 power–train, which, in turn, depends on the torque produced by each piston. In [2], we showed that the engine can be modeled using a hybrid system composed of interacting finite–state machines, discrete–event systems and continuous–time systems. As a second case, consider for instance the model of an automotive driveline. An accurate model of the driveline has a natural hybrid representation because the gear has different position and the clutch can be locked, unlocked or slipping. In [3], a detailed model with up to 6048 discrete state combinations and 12 continuous state variables was presented. The hybrid model accurately represents discontinuities distributed along the drive line due to engine suspension, clutch, gear, elastic torsional characteristic, tires, frictions and backlashes. Finally, models of automotive subsystems are often highly nonlinear. In engine modeling, nonlinearities arise from fluid–dynamics and thermodynamics phenomena (e.g. volumetric efficiency, engine torque, emissions) and are usually represented by piece–wise affine maps.

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.

1.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 (e.g. intake manifold model) 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 [9]) or piece–wise polynomial functions, would produce majors impact 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;
- higher dimension nonlinearities $R^p \rightarrow R$ could be represented and identified.

1.c) Validation: The selection of test patterns for model validation is a crucial issue in the validation process. Classical techniques allow to assess the richness of sets of test patterns for the validation of continuous models. These techniques can be used in automotive applications to assess richness of validation patterns for continuous evolutions of the plant. However, the problem remains open for hybrid model validations. This topic is further discussed in Section III.C.2.c, where automatic test pattern generation for controller validation is analyzed.

Validation of hybrid models is a very complex task not sufficiently investigated in the literature. In particular, the following open problems must be addressed:

- methodologies for automatic generation of extensive validation patterns for hybrid models;
- techniques for the assessment of the completeness of validation patterns. This problem can be formalized in the framework of reachability analysis and interesting approaches have been proposed using the concepts of structural coverage and data coverage.

2.a) Plant model and specifications analysis: Typically the design process of a control algorithm for a new application starts with the definition of a plant model based on the analysis of some experimental data obtained either with open–loop control or with some very elementary closed–loop algorithm. The assessment of classical structural properties, such as reachability, observability, stabilizability, passivity [8], on the plant model is of interest in this phase. In addition, quantitative analysis is very useful to understand the strengths and weaknesses 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.

Classical concepts and techniques for system analysis cannot be applied to hybrid systems (e.g. switching systems stability has no direct relation with subsystems poles). Unfortunately hybrid system theory has not been developed to a point to be trusted for model analysis:

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

2.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. During standard functioning, control strategies can be either at the nominal operation mode or at one of several recovery modes. A significant number of algorithms are dedicated to the computation of switching conditions 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 and safety and recovery (degraded mode activation).

Diagnostic algorithms represent a major 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 [11].

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 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 is often the case, 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 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 so far proposed for controller design using hybrid system techniques are quite complex. Usually, the application of these techniques requires designers that are trained in hybrid systems and necessitates long development times. As a consequence, the hybrid system design process results too expensive for the human resources commonly deployed in automotive system engineering. Hence, for a profitable introduction of hybrid system design techniques, it is essential that methodologies are supported by efficient tools that allow fast and easy designs. Hybrid model predictive control is a good example in hybrid system research where the development of the methodology was supported by a good effort in design tool development [10].

2.c) Controller validation: Control algorithms are validated in extensive, time-consuming and 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 even if some of the critical maneuvers may be provided by the system specifications. 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 the 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. Some classes of algorithms that require intensive and complex validation are diagnosis algorithms, safety critical algorithms and algorithms preventing the system to stall (e.g. idle speed control). 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. Hybrid system techniques can contribute significantly to the improvement of the validation process. In particular, 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.

Moreover, 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 and implementation parameters (e.g. sampling–period, latency, jitter, computation precision, etc.) for which the desired performances are achieved.

D. Hardware/Software components

The design of HW/SW implementation of ECUs follows today the standard methodologies for hardware and software development. However, 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 errors. In particular, we see value for hybrid methodologies at the boundary between control engineering and implementation design. The methodologies and the design tools in the control domain and the HW and SW implementation domains are often not integrated; this situation is the frequent cause of design errors. The specification for the HW/SW implementation must include all details necessary for a correct implementation of the algorithms, i.e., they 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 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 for software implementation and with tools for automatic synthesis for hardware design. Moreover, 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 good enough. 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.

Finally, methodologies and tools for defining and validating implementation constraints should be developed. In particular, 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 and tools.

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

IV. CONCLUSIONS

We described critically the automotive electronic design flow in use today with the intention of underlining where hybrid methods can be of use to improve the quality of design. The quality of present products is far from being satisfactory in view of the rapid advances of integrated circuit and system technology, and of the ever increasing demands on functionality and time to market. While we are optimistic that hybrid systems will be of good use in automotive electronics, the difficulties in propagating this approach to design cannot be overemphasized. A coherent set of tools and training approach should be developed to make hybrid systems and their relationship with embedded systems appealing to automotive engineers. The most obvious application of hybrid systems is for modeling and control at the highest level of abstraction, e.g. in engine control. However, we believe that a most profitable application will also be at the boundary of control design and implementation engineering where the effects of limited resources and physics on the control performance has to be captured to verify the correctness of overall system (plant and controller).

V. ACKNOWLEDGMENTS

We wish to thank Alberto Ferrari and Pierpaolo Murrieri of PARADES; Gabriele Serra, Giacomo Gentile and Walter Nesci, of Magneti Marelli Powertrain (Bologna, I); Paolo Ferracin of CNH (Modena, I); Gilberto Burgio of Ford Motor Company (Aachen, G); M.D. DiBenedetto of the University of L'Aquila, for the many discussions on the topic.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] A. Bemporad, G. Bianchini, F. Brogi, and F. Barbagli. Passivity analysis and passification of discrete-time hybrid systems. In *Proc. IFAC World Congress*, Prague, Czech Republic, 2005. Submitted.

[9] 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.

[10] 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/.

[11] 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.

[12] R. M obus, 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.