Proceedings of the 2004 IEEE
Conference on Computer Aided Control Systems Design
Taipei, Taiwan, September 2-4, 2004

A Multi-Paradigm Modeling Approach for Hybrid Dynamic Systems

Jin-Shyan Lee[1,2], Meng-Chu Zhou[2], and Pau-Lo Hsu[1]
[1]Department of Electrical & Control Engineering
National Chiao-Tung University, Hsinchu, Taiwan
[2]Department of Electrical & Computer Engineering
New Jersey Institute of Technology, Newark, NJ, USA
jslee.ece88g@nctu.edu.tw, zhou@njit.edu, and plhsu@cc.nctu.edu.tw

Abstract

In the past years, modeling and simulation of hybrid dynamic systems (HDS) have attracted much attention. However, since simultaneously dealing with the discrete and continuous variables is very difficult, most of the models result in a unified, but more complicated and unnatural format. Moreover, design engineers cannot be allowed to use their preferred domain models. Based on the multi-paradigm modeling (MPaM) concept, this paper proposed a Petri net (PN) framework with associated state equations to model the HDS. In the presented approach, modeling schemes of the hybrid systems are separated, but combined in a hierarchical way through specified interfaces. Designers can still work in their familiar domain-specific modeling paradigms and the heterogeneity is hidden when composing large systems. An application to a rapid thermal process (RTP) in semiconductor manufacturing is provided to demonstrate the practicability of the developed approach.

Keywords: multi-paradigm modeling, hybrid dynamic systems, Petri nets.

1. Introduction

Recently, most work on system modeling and simulation has been mainly focused on either continuous variable systems (CVS) or discrete event systems (DES). The former are typically modeled by differential or difference equations to analyze physical dynamic behavior, while the latter are described based on various frameworks to capture logical and sequential behaviors, such as finite state automata, Petri nets (PN), and max-min algebra [1]. As a paradigm, modeling is a way of representing our knowledge about structure and behavior of systems so as to further answer questions about them. Both of the CVS and DES models are developed to reduce the complexity for presenting real systems. However, in practical applications, most systems possess the behavior that combines both the time-driven and event-driven dynamics together as so-called hybrid dynamic systems (HDS) or hybrid systems [2]-[4].

During the past decades, modeling and simulation of the HDS attracted much attention. Integrating continuous and discrete models is common, and several approaches have been proposed. Liu and Lee proposed a component-based approach implemented in Ptolemy II software environment [5]. Demongodin and Koussoulas proposed a differential Petri net to represent the continuous system part and the discrete-event system part of a hybrid system in a collective PN model [6]. Moreover, some commercial software has been developed, such as Simulink with the integration of Stateflow in the MATLAB platform [7], and VHDL-AMS/Verilog-AMS with the extensions of hardware description languages to analog and mixed signals [8]. However, it is very difficult to simultaneously deal with the discrete and continuous variables. Their mathematical backgrounds are completely different: recurrence versus differential equations. Moreover, most of the existing approaches mix the continuous and discrete dynamics into a unified model, such as piece-wise linear systems, hybrid Petri nets (HPN) [9], and differential Petri nets. Hence, design engineers cannot be allowed to use their preferred domain models.

Over the years, different engineering domains have come up with various modeling abstractions that best suit the design of particular kinds of systems. Multi-paradigm modeling (MPaM) is based on the premise of giving modelers the most appropriate modeling abstractions for their particular problem domain and integrating heterogeneous modeling techniques to achieve scalable designs. From an MPaM point of view, Lee and Hsu extended the statechart of the unified modeling language (UML) to design a hybrid controller of automated vehicles [10] and a three-axis motion control system [11] resulting in a clear and natural representation. Their proposed concept of the hybrid statechart is similar to the hybrid automata [12]. Generally speaking, either the statechart or automata mostly describes finite state machines that are restricted to finite state systems, while PN is better for presenting infinite state systems. Moreover, precise semantics and powerful analysis are required to prevent system deadlock and to discover the bottleneck in the present design [18]. PN provides powerful qualitative analysis and quantitative analysis, and also provides both rich visual formalism for specifying behavior and an executable notation [13]. Thus, based on the MPaM concept, this paper proposes a modeling approach within a

PN framework. With different views of the system, it could be more efficient to work with separate interacting models. When composing large systems, designers can still work in their familiar domain-specific modeling paradigms so that the heterogeneity is hidden in this way. An example of a rapid thermal process (RTP) in semiconductor manufacturing is illustrated to show the feasibility of the proposed approach.

2. MPaM for Hybrid Control

A. Multiple Models in Hybrid Control

The art of system modeling is to choose the right level of abstraction to capture the aspects worth exploring, and to ignore the irrelevant details. This paper presents an MPaM approach within a PN-based framework for hybrid control systems. As shown in Fig. 1, a high-level coordination controller (within discrete-event domain) supervises the middle-level digital controller (within discrete-time domain) through two interfaces: signal generator and event generator. The signal generator is applied to transform the action commands from the coordination controller to digital signals, while the event generator is used to trigger events according to some critical thresholds in digital signals. Then, the digital controller regulates the low-level physical plant (within continuous-time domain) via two interfacing devices: sampler and holder. The sampler is used to generate the digital signals by sampling the analog signals in the physical plant, while the holder is applied to construct the piecewise-continuous analog signals. Note that signal or event conversions through boundary interfaces could be multiple signals or events with a vector or matrix format. In practice, a coordination controller supervises more than one digital device of physical plant, especially in the fields of logical control for a manufacturing system, traffic control of a highway, and sequence control of a chemical process. Fig. 1 also shows a typical centralized and hierarchical control scheme, in which the coordination controller supervises several digital controllers in a hierarchical way.

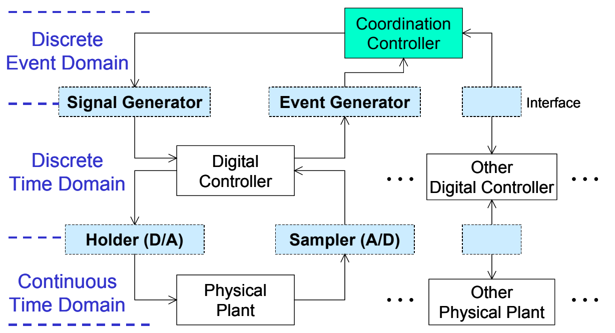

Fig. 1. Multiple domains in hybrid control systems.

In the presented approach, the discrete-event, discrete-time, and continuous-time models are connected together in a hierarchical way to form the whole systems. Even though the proposed model presentation is more compact and simpler in a natural way, it is noted that conjunct interfaces of heterogeneous models is an important issue. In this paper, the interfacing devices for the communication among models have been described.

B.PN Framework for Discrete-Event and Discrete-Time Domains

From the MPaM point of view, in the discrete-event domain, the PN is employed to model and design the coordination controller. The main motivation for using PN as hybrid models is the fact that all those good features that make discrete PN a valuable discrete-event model still be available to hybrid systems. Examples of these features are: PN does not require the exhaustive enumeration of the state space and can finitely describe systems with an infinite state space. Also, PN allows modular representation where the structure of each module is kept in the composed model. In addition, the discrete state of PN is represented by a vector and not by a symbolic label, thus linear algebraic techniques may be used for analysis.

Fig. 2 shows the proposed PN framework for modeling the discrete-event and discrete-time domains. Each operation is modeled with a *command* transition to start the operation, a progressive *working* place, a *response* transition to end the operation, and a *completed* place. Note that the start transition (drawn with a dark symbol) is a controllable event as "command" input, while the end transition is an uncontrollable event as "response" output. The working place is a hierarchical hybrid place (drawn with a triple circle), in which the state equations of under controlled systems are contained and interacted through the boundary interface. The interaction between the event-driven and time-driven domains is achieved in the following way: a token put into the working place starts the integration of the corresponding equations. Concurrently with the integration a certain number of thresholds are monitored. Each threshold is associated with a transition, i.e. the response transitions. When the threshold is crossed, it means that the corresponding event is occurring, and the attached transition is fired. The new marking is computed, and the integration of a new system starts.

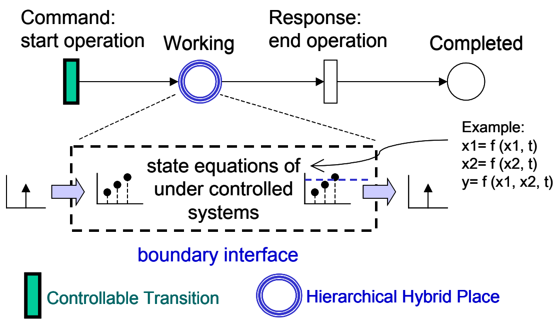

Fig. 2. Modeling the continuous dynamics within a Petri net via a hierarchical way.

C. State Equations for Discrete-Time and Continuous-Time Domains

On the other hand, in the hybrid subsystem of discrete-time and the continuous-time domains, i.e. the so-called sampled-data system, differential and difference equations or Laplace-S and Z functions are typically used to model and analyze such systems. This paper is focused on the modeling of the hybrid system from the high-level viewpoint. Design approaches of the coordination and digital controllers will be briefly mentioned in the following sections.

D. Comparison with Hybrid Petri Net

Hybrid Petri net (HPN) [9] contains both conventional discrete PN and continuous PN, as shown in Fig. 3. In this model, the discrete part is represented by the discrete places and transitions, and the continuous part is by the continuous places (drawn with a double circle) and transitions. In the continuous PN, the marking of a place is a real positive number and the firing is carried out like a continuous flow. The unique way of describing interactions between the continuous part and the discrete one is by means of self-loops. Basically, the discrete PN is applied to trigger the continuous transitions. However, this approach can only describe real and non-negative continuous variables. Moreover, a continuous place represents only one continuous variable, e.g. the x1, x2, and y in Fig, 3. In basic HPN, only continuous variables that are linear with respect to time could be represented constant speeds. It has been extended to represent other kinds of evolutions, but for each kind of evolution, a new type of continuous place has to be introduced.

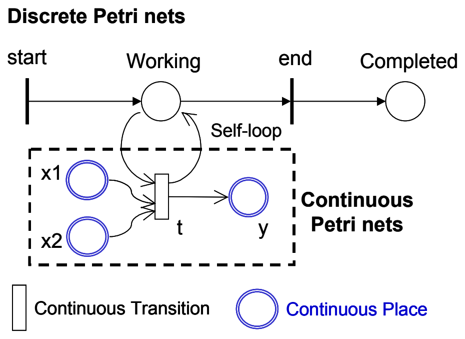

Fig. 3. Hybrid Petri net (with a net structure to model continuous dynamics).

In this paper, instead of using the net structure (i.e. the continuous PN in HPN) to model the continuous part of the system, the proposed framework, shown in Fig. 2, uses the original state equations to represent the continuous system dynamics. In this way, it can be said that the PN (event-driven part of the model) coordinates the system of difference-algebraic equations whose structure changes each time a transition is fired. Also, the system of equations (time-driven part of the model) regulates the PN evolutions by enforcing the firing of its transition via the threshold-crossing mechanism.

The proposed approach allows using the domain-specific primitive models, such as differential/difference equations, to capture low-level physical interactions, and then connected with the high-level PN in a hierarchical way. Thus, the resultant model with a natural description is more compact and simpler than HPN. Although the framework we are presenting is general in natural, in this paper we will concentrate on the specific class of semiconductor manufacturing systems that largely motivate this work.

3. Discrete-Event and Discrete-Time Models

The upper part in Fig. 1 shows a typical hybrid structure of the discrete-event and discrete-time models. In this section, we briefly review the discrete-event systems, and then describe the interfaces and controller design of such hybrid systems.

A. Discrete-Event Systems

A discrete event system is a dynamic, asynchronous system, where the state transitions are initiated by events that occur at discrete instants of time. The state of a DES may change abruptly at the occurrence of an event, and in between two events the system remains in the same state.

B. Interfaces between Discrete-Event and Discrete-Time Models

Signals in discrete-event and discrete-time models are fundamentally different. When combining these models, appropriate signal conversion mechanisms need to be introduced. Most signal conversion algorithms are application-specific, and must be implemented as separate components. As shown in Fig. 4 (a), a signal generator is a device that converts the discrete event into the digital signal. This device typically uses an event as the input command (such as to push a button) to trigger a process with discrete-time dynamics. On the other hand, an event generator is the device that generates discrete events from digital signals, as shown in Fig. 4 (b). A critical job for the event generator is to find the event timestamp, which depends on the values of the state variables in the controlled system (such as a threshold-crossing event).

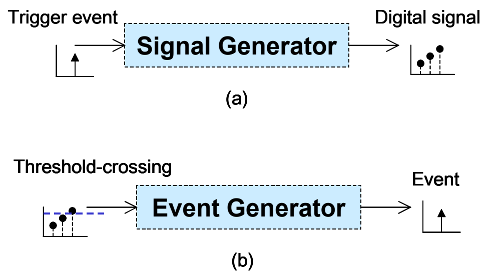

Fig. 4. Interfacing devices in high level: signal and event generators.

C. Coordination Controller Design

In this paper, we directly adopt the supervisor synthesis method from [14] to build the PN model of the coordination controller. The design procedure consists of the following steps:

Step 1) Construct the PN model of the plant.
Step 2) Construct the PN model of the specifications (e.g. the sequence of processes, operation switching, and resource constraints).
Step 3) Compose the plant and specification models to yield the coordination controller.
Step 4) Verify and refine the coordination controller to obtain a live, bounded and reversible model.

In this present scheme, the designer of coordination controllers can purely focus on the discrete-event and discrete-time domains without considering the detailed complex dynamics of the low-level physical systems.

4. Discrete-Time and Continuous-Time Models

The lower part in Fig. 1 shows a typical hybrid configuration of the continuous-time plant with discrete-time feedback controller, i.e. so-called a sampled-data system. In this section, we briefly review such kind of system, and discuss its controller design for hybrid systems.

A. Sampled-Data Systems

A sampled-data system combines both continuous and discrete-time dynamic subsystems. Because of this inherent mixture of time domains, we shall also refer to a sampled-data system as a hybrid system. Although the plant is usually a continuous-time (or analog) system, in most practical applications, the controller is a discrete-time (or digital) device. This is mainly due to the numerous advantages that digital equipments offer over their analog counterparts. With the great advances in computer technology, today digital controllers are more compact, reliable, flexible and often less expensive than analog ones.

B. Interfaces between Discrete-Time and Continuous-Time Models

There is a fundamental operational difference between digital and analog controllers, i.e. the digital system acts on samples of the measured plant output rather than on the continuous-time signal. A practical implication of this difference is that a digital controller requires special interfaces that link it to the analog world.

A digital controller (i.e. a discrete-time controller) can be idealized as consisting of three main elements: the analog-to-digital (A/D) interface, the digital computer, and the digital-to-analog (D/A) interface. As shown in Fig. 5 (a), the A/D interface, or sampler, acts on a physical variable, normally an electric voltage, and converts it into a sequence of binary numbers, which represent the values of the variable at the sampling instants. These numbers are then processed by the digital computer, which generates a new sequence of binary numbers that correspond to the discrete control signal. As shown in Fig. 5 (b), this control signal is finally converted into an analog voltage by the D/A interface, also called the holder. The digital computer implements the control algorithm as a set of difference equations, which represent a dynamic system in the discrete-time domain.

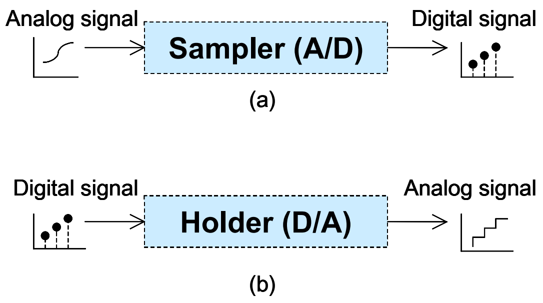

Fig. 5. Interfacing devices in low level: sampler and holder.

C. Digital Controller Design

Basically, two classic approaches are taken in continuous-time and discrete-time domains, respectively, for a digital controller design. The first technique, referred to as emulation [15], is the most widely applied in industry. Emulation consists in first designing an analog controller such that the closed-loop system has satisfactory properties, and then translating the analog design into a discrete one using a suitable discretization method. This technique has the advantage that the synthesis is done in continuous time, where the design goals are typically specified, and where most of the designer's experience and intuition resides. Also, the system's analog performance will in general be recovered for fast sampling. The second traditional technique consists in discretizing the plant and performing a controller design in discrete-time domain directly. The main benefit of this approach is that the synthesis procedure is again simplified, since the discretized plant is linear time-invariant (LTI) in the discrete-time domain. In this presented MPaM scheme, both approaches are acceptable. Thus, the designers of digital controllers can easily focus on the discrete-time and continuous-time domains and use their preferable approaches without considering the logical behaviors of the high-level system.

5. Application Example

This section demonstrates a practical application of the hybrid control for a rapid thermal process (RTP). The status of the hybrid RTP system at any time instant involves the upper event-driven states and lower time-driven states of all components in the system at that time.

A. Description of the RTP System

A rapid thermal processor is a relatively new semiconductor manufacturing device [16]. A schematic diagram of the RTP system is shown in Fig. 6, which is composed of 1) a reaction chamber with a door, 2) a robot arm for wafer loading/unloading, 3) a gas supply module with a mass flow controller and pressure controller-I, 4) a heating lamp module with a temperature controller, and 5) a flush pumping system with a pressure controller-II. Note that the initial state of the components in the RTP is either closed or off, except that the door is open. A realistic "recipe" of the hydrogen baking process is as follows:

Step 1)	Load the raw wafer.
Step 2)	Close the chamber door.
Step 3)	Open the gas valve to supply gases with a desired gas flow rate and pressure of 2.8 liters per minute (lpm) and 0.55 Torr, respectively.
Step 4)	Close the gas valve.
Step 5)	Turn on the heating lamp to bake the wafer with a desired baking temperature and duration of 1000 °C and 4 seconds, respectively.
Step 6)	Turn off the heating lamp.
Step 7)	Turn on the flush pump with a desired pressure of less than 0.05 Torr.
Step 8)	Turn off the flush pump.
Step 9)	Open the chamber door.
Step 10)	Unload the processed wafer.

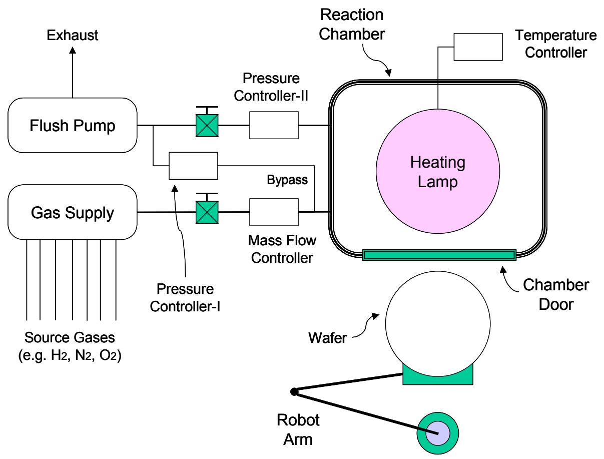

Fig. 6. Schematic diagram of the RTP system.

B. System Modeling and Simulation

By applying the design procedure in Section III-C, the PN model of RTP is constructed as shown in Fig. 7, which consists of 26 places and 20 transitions, respectively. Corresponding notations are described in Table I. After constructing the PN model, we can perform the model validation of the dynamic behavior. Due to its graphical representation, ease of manipulation, and ability to perform structural analysis, the software package ARP [17] is adopted to verify the behavioral properties of the developed PN model. Validation results reveal that the present PN model is live and bounded. The liveness property means that the system can be executed properly without deadlocks, while the boundedness property means that the system can be executed with limited resources (e.g., limited buffer sizes). Then, by interacting with the time-driven dynamics, Fig. 8 shows the pressure and temperature of one cycle for a wafer processing in RTP, including the wafer loading, gas supplying, chamber heating, chamber pumping, and wafer unloading. This simulation result indicates that the corresponding discrete events are successfully triggered according to the desired crossing thresholds of the continuous variables in the time domain.

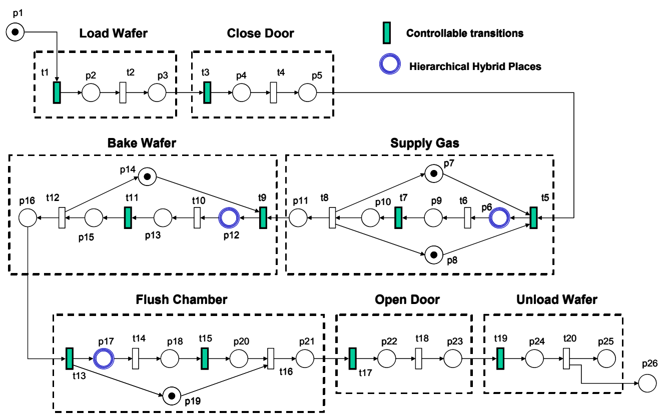

Fig. 7. Petri net model of the RTP system.

Table I
Notation for the Petri net of the RTP in Fig. 7.

Place	Description	Transition	Description
p1	Raw wafer buffer	t1	Cmd: start loading wafer
p2	Loading wafer	t2	Re: end loading wafer
p3	Loading wafer completed	t3	Cmd: start closing chamber door
p4	Closing chamber door	t4	Re: end closing chamber door
p5	Closing chamber door completed	t5	Cmd: start opening gas valve
p6	Opening gas valve	t6	Re: end opening gas valve
p7	Mass flow controller ready	t7	Cmd: start closing gas valve
p8	Pressure controller-I ready	t8	Re: end closing gas valve
p9	Opening gas valve completed	t9	Cmd: start turning on heating lamp
p10	Closing gas valve	t10	Re: end turning on heating lamp
p11	Closing gas valve completed	t11	Cmd: start turning off heating lamp
p12	Turning on heating lamp	t12	Re: end turning off heating lamp
p13	Turning on heating lamp completed	t13	Cmd: start turning on flush pump
p14	Temperature controller ready	t14	Re: end turning on flush pump
p15	Turning off heating lamp	t15	Cmd: start turning off flush pump
p16	Turning off heating lamp completed	t16	Re: end turning off flush pump
p17	Turning on flush pump	t17	Cmd: start opening chamber door
p18	Turning on flush pump completed	t18	Re: end opening chamber door
p19	Pressure controller-II ready	t19	Cmd: start unloading wafer
p20	Turning off flush pump	t20	Re: end unloading wafer
p21	Turning off flush pump completed		
p22	Opening chamber door		
p23	Opening chamber door completed		
p24	Unloading wafer		
p25	Unloading wafer completed		
p26	Processed wafer buffer		

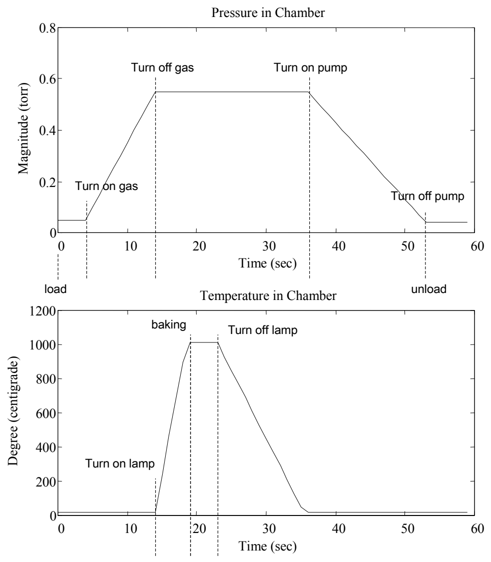

Fig. 8. Pressure and temperature in the RTP system for one cycle processing.

6. Conclusion

This paper presents a multi-paradigm modeling approach within a PN framework for hybrid dynamic systems. The presented approach allows the use of preferred domain models and leads to a more natural representation of the continuous part using a hierarchical framework. Designers can still work in their familiar domain-specific modeling paradigms and the heterogeneity is hidden when composing large systems. Multiple models can then be hierarchically composed to build complex models. Thus, through combining the use of well-established theories of PN and differential/difference equations, each one being well suited for an aspect of the problem, and emphasize on the interface between the aspects.

Acknowledgement

This work was in part supported by the National Science Council, R.O.C., under Grant NSC 92-2917-I-009-005 and the Ministry of Education, R.O.C., under Grant Ex-91-E-FA06-4-4.

Reference

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston, MA: Kluwer, 1999.

[2] A. S. Morse, C. C. Pantelides, S. S. Sastry, and J. M. Schumacher, "Introduction to the special issue on hybrid systems," *Automatica*, vol. 35, no. 3, pp. 347-348, 1999.

[3] P. J. Antsaklis and M. D. Lemmon, "Introduction to the special issue," *J. Discrete Event Dynamic Syst.* (Special issue on hybrid systems), vol. 8, no. 2, pp. 101-103, 1998.

[4] P. J. Antsaklis and A. Nerode, "Hybrid control systems: An introductory discussion to the special issue," *IEEE Trans. Automat. Contr.* vol. 43, no. 4, pp. 457-460, 1998.

[5] J. Liu and E. A. Lee, "A component-based approach to modeling and simulating mixed-signal and hybrid systems," *ACM Trans. Modeling and Computer Simulation* (Special issue on computer automated multi-paradigm modeling), vol. 12, no. 4, pp. 343–368, 2002.

[6] I. Demongodin and N. T. Koussoulas, "Differential Petri nets: Representing continuous systems in a discrete-event world," *IEEE Trans. Automat. Contr.* vol. 43, no. 4, pp. 573-578, 1998.

[7] T. Harman and J. Dabney, *Mastering Simulink 4*. Englewood Cliffs, NJ: Prentice Hall, 2001

[8] K. Bakalar and E. Christen, *VHDL 1076.1: Analog and Mixed Signal Extensions for VHDL*. Tech. Rep., IEEE, 1999.

[9] R. David and H. Alla, "Petri nets for modeling of dynamics systems— A survey," *Automatica*, vol. 30, no. 2, pp. 175-202, 1994.

[10] J. S. Lee and P. L. Hsu, "An object-oriented design of the hybrid controller for automated vehicles in an AHS," *IEEE Intelligent Vehicles Symposium*, Versailles, France, June 2002, pp. 115-120.

[11] J. S. Lee and P. L. Hsu, "UML-based modeling and multi-threaded simulation for hybrid dynamic systems," *IEEE Int. Conf. Control Applications*, Glasgow, Scotland, September 2002, pp. 1207-1212.

[12] J. Lygeros, K. H. Johansson, S. N. Simic´, J. Zhang, and S. S. Sastry, "Dynamical properties of hybrid automata," *IEEE Trans. Automat. Contr.* vol. 48, no. 1, pp. 2-17, 2003.

[13] M. C. Zhou and M. D. Jeng, "Modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems: A Petri net approach," *IEEE Trans. Semicond. Manuf.*, vol. 11, no. 3, pp. 333-357, 1998. (Special section on Petri nets in semiconductor manufacturing).

[14] J. S. Lee and P. L. Hsu, "Remote supervisory control of the human-in-the-loop system by using Petri nets and Java," *IEEE Trans. Ind. Electron.*, vol. 50, no. 3, pp. 431-439, June 2003.

[15] G. Franklin, J. Powell, and M. Workman, *Digital Control of Dynamic Systems*. Reading, MA: Addison-Wesley, 1990.

[16] R. B. Fair, Rapid Thermal Processing: Science and Technology. New York: Academic, 1993.

[17] C. A. Maziero, *ARP: Petri Net Analyzer*. Control and Microinformatic Laboratory, Federal University of Santa Catarina, Brazil, 1990.

[18] J. S. Lee and P. L. Hsu, "Design and implementation of the SNMP agents for remote monitoring and control via UML and Petri nets," *IEEE Trans. Contr. Syst. Tech.*, vol. 12, no. 2, pp. 293-302, March 2004