MODELING LANGUAGES FOR CONTINUOUS AND DISCRETE SYSTEMS

Peter Schwarz
Fraunhofer Institute for Integrated Circuits IIS, Design Automation Division EAS Dresden, Germany

Keywords: Behavioral model, Block diagram, Controller, Conservative quantity, Constitutive relation, DAE, Difference variable, Discrete time system, Flow variable, HDL, Interconnective constraints, Kirchhoff's law, Mixed-signal simulation, Modelica, Modeling Language, Multi-port, Network, Non-conservative quantity, Object-oriented modeling, ODE, Plant, Signal flow graph, Structural model, System simulation, VHDL, VHDL-AMS.

Contents

Summary

This chapter presents some aspects of modeling and simulation of control systems. The controller and the controlled plant have to be simulated together. Block-oriented simulators (e.g. the very popular MATLAB/SIMULINK, which is also mentioned in other articles of this theme) are widely used for this task. But in many situations it is more convenient to apply simulators which use flow and difference variables to model the systems and the interaction of their components in the electrical, mechanical, hydraulic, or fluidic domain.

Languages for behavioral and structural modeling of heterogeneous physical systems have been developed in the last years; some of them are standardized by the IEEE organization. These modeling languages (e.g. VHDL-AMS and Modelica) are supported by powerful simulators which are capable of solving continuous DAE systems together with discrete-event models (hybrid simulation, mixed-signal simulation). Modeling with (generalized) Kirchhoff's networks is very appropriate to the physical nature of the plant and may be considered as an object-oriented modeling approach.

1. Aims of Modeling Languages

For the validation of a new control system and for understanding physical phenomena, simulation is the most powerful computer-aided analysis method. Parameter variation and optimization as well as sensitivity investigations may be carried out by simulation. The *controlled plant* and the *controller* have to be modeled and simulated together (Figure 1).

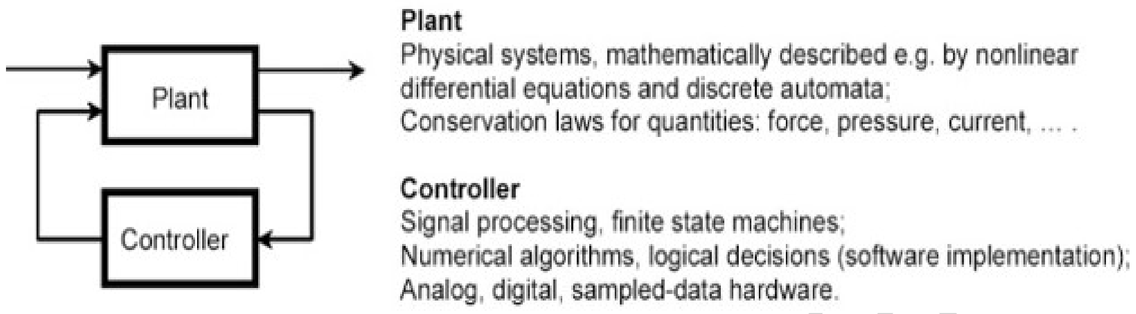

Plant
Controller
Plant
Physical systems, mathematically described e.g. by nonlinear differential equations and discrete automata;
Conservation laws for quantities: force, pressure, current, ...
Controller
Signal processing, finite state machines;
Numerical algorithms, logical decisions (software implementation);
Analog, digital, sampled-data hardware.

Figure1: Common modeling of plant and controller

From the physical point of view we have to consider:

- mixed-domain systems (mechanical, electrical, thermal, fluidic, ... phenomena),
- partially close coupling between these domains (especially in plant modeling),
- spatially distributed and lumped (concentrated) elements,
- discrete-time and continuous-time signals and systems (e.g. in electronics: analog and digital control systems).

Therefore various mathematical descriptions have to be taken into account:

- partial and ordinary differential equations,
- very large and stiff systems of differential equations to describe the continuous subsystems,
- linear as well as nonlinear differential equations, e.g. in the form of linear state-space descriptions (ODE) or nonlinear differential-algebraic equations (DAE),
- automata, BOOLEAN expressions, finite state machines (FSM), PETRI nets.

The distinction between a directed signal flow and energy-related quantities (e.g. with “effort and flow variables”, see *Modeling and Simulation of Dynamic Systems Using Bond Graphs*) is also very important for the modeling approach and the choice of a simulator. *System simulation* means, therefore, the simulation of very complex and often also heterogeneous systems. To emphasize these aspects, sometimes it is called *overall system simulation*.

To understand the basic problems of choosing a suitable simulator and a modeling approach, let us have a look at the underlying simulation principles. At first, we will focus on the simulation of continuous systems described by nonlinear differential-algebraic equations (DAE) or explicit ordinary differential equations (ODE):

$$\mathbf{f}(\mathbf{x}, \frac{d\mathbf{x}}{dt}, \mathbf{u}, t) = \mathbf{0} \tag{1a}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{h}(\mathbf{x}, \mathbf{u}, t) \tag{1b}$$

($\mathbf{x}$ = vector of system variables, $\mathbf{u}$ = vector of input variables, t = time). The user has to formulate the simulation problem description in textual or graphical form. It is the task of the input compiler to set-up an internal representation of the complete DAE or ODE system on the basis of the system topology, new user-defined models, and internal equations of build-in models, see Figure 2.

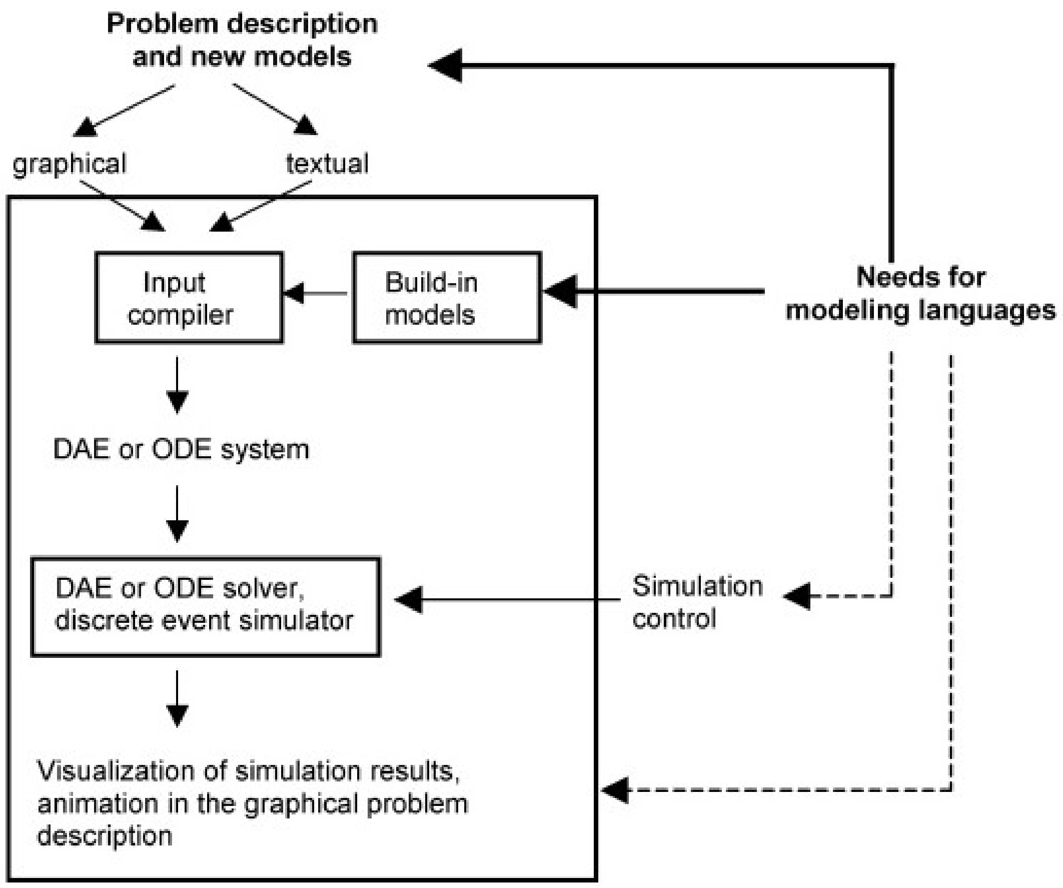
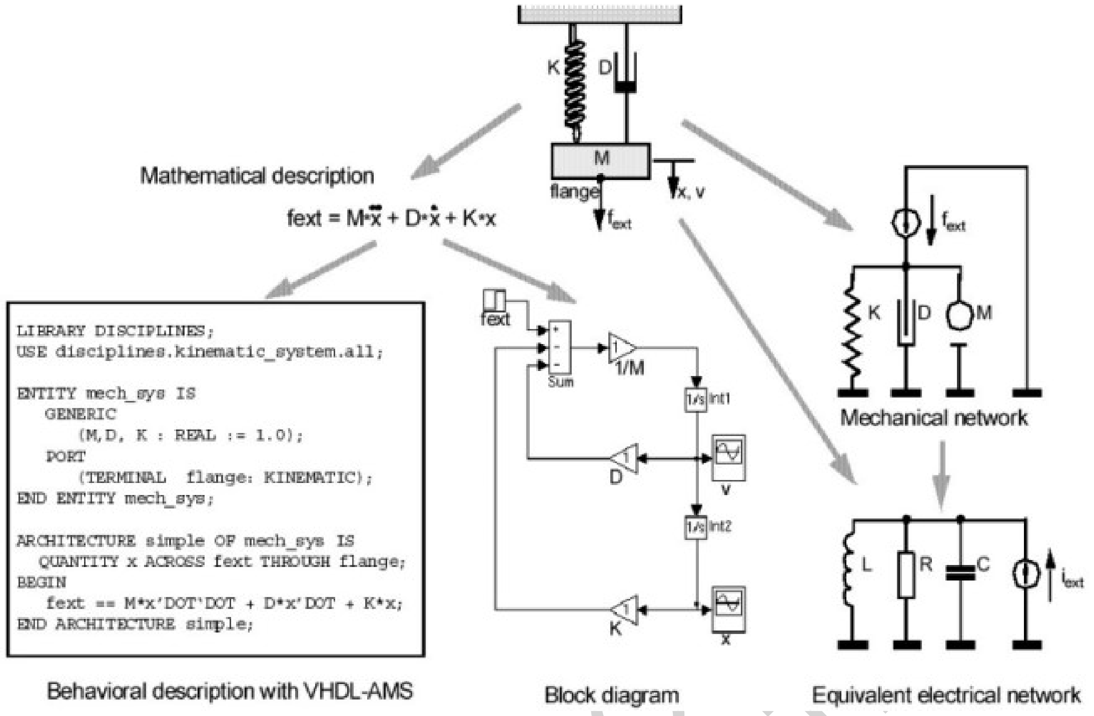

Problem description and new models
graphical
textual
Input compiler
Build-in models
Needs for modeling languages
DAE or ODE system
DAE or ODE solver, discrete event simulator
Simulation control
Visualization of simulation results, animation in the graphical problem description

Figure 2: Simulator structure and modeling languages

There are also two other basically different forms of problem descriptions: structure-oriented and equation-oriented (and a mixture of them). Various graphical and textual description styles of control systems are presented in other contributions of this theme. To illustrate some description forms of the plant, different models of the same technical system are presented in Figure 3. The technical system (a very small part of a mechanical system) consists of a mass, a spring and a damping part and is modeled by the interconnection of the model elements M, K, and D.



```
LIBRARY DISCIPLINES;
USE disciplines.kinematic_system.all;

ENTITY mech_sys IS
   GENERIC
      (M,D, K : REAL := 1.0);
   PORT
      (TERMINAL  flange: KINEMATIC);
END ENTITY mech_sys;

ARCHITECTURE simple OF mech_sys IS
  QUANTITY x ACROSS fext THROUGH flange;
BEGIN
   fext == M*x'DOT'DOT + D*x'DOT + K*x;
END ARCHITECTURE simple;
```

Figure 3: Simple mechanical system

There are, principally, two ways to model such a system:

- as a mathematical description, which may be formulated as
    - * a text written in the style of a programming language
    - * a block diagram, e.g. for the popular SIMULINK simulator
- as a "physical" model with a structure similar to the structure of the original system
    - * a mechanical network
    - * an electrical network (constructed by using analogy relations)

In this example, the different model descriptions are illustrated with respect to continuous systems. Similar possibilities exist for modeling discrete systems, e.g. graphically described queuing systems and an equivalent textual GPSS model, or a state-chart model of a digital controller which may be transformed into a textual VHDL (or Verilog) model.

In the context of the various modeling possibilities, the following notation is often used:

Behavioral model: The behavior of the whole system or of a component (an element) is described by mathematical formulas. The notation may be carried out in the form of a program written in a universal programming language (FORTRAN, C, C++), in a simulator-specific description language (such as MAST for Saber), or in a standardized modeling language (e.g. CSSL or VHDL-AMS).

Structural model: A complex model results from the *interconnection* of simpler *components*. For these basic components simulator-internal models exist, the so-called build-in models or "simulator primitives". Such basic components are, e.g., resistors, capacitors, linear transfer function blocks (e.g. a PID block), adders, multipliers, and different types of signal sources. While older simulators *only* support this type of modeling, modern simulators *additionally* permit a behavioral element description formulated by the user.

A mixed structural and behavioral description is particularly efficient: the whole system is modeled by the interconnection of subsystems which are modeled by behavioral descriptions or, hierarchically, by the interconnection of other basic elements. Behavioral modeling is the most comfortable way to include numerical models resulting from identification procedures (see *Identification of Nonlinear Systems* and *Identification of Linear Systems in Time Domain*).

In many cases the decomposition of a system model into the interconnection of simulator primitives may be carried out easily. The widely used simulator MATLAB/SIMULINK with its graphical input facilities and large libraries of control system block models or the circuit simulator SPICE with its electronic component models are popular examples. But in the case of modeling complicated phenomena it is more convenient to use mathematical *formulas* and *algorithms* to describe the behavior of components. In these applications it is necessary to formulate the model equations in a user-friendly form. These requirements led to the development of **modeling languages** which can describe the interconnection of components or subsystems as well as the individual model equations of some components or subsystems.

2. Historical background

Modeling languages have a long history (see *Modeling and Simulation of Dynamic Systems*, *Simulation Software - Developments and Trends*). CSMP, ACSL, the standard CSSL, Modelica, Dymola, GPSS and SIMSCRIPT are only a few (but very important) simulation languages for general continuous, discrete or hybrid (= continuous + discrete) systems. Additional simulation languages are very popular in special application areas like mechanics and electronics.

It was a feature of the first simulation languages that the user is responsible for the notation of model equations in a "correct order". "Simulation" was then reduced to a step-by-step evaluation of these equations (as in all sequential programming languages). "Correct order" means sorting the model equations according to the directed signal flow:

- ports of all elements ("blocks") can be partitioned into input and output ports,
- there is a directed (or uni-directional) signal flow between the blocks and, herefore, it is possible to calculate the new output signal x_i of a block i by using the output signals of preceding blocks i-1, i-2, …, which have been already calculated, as input signals of the block i:

$$x_i = f_i\left(x_{i-1}, x_{i-2}, ..., x_0\right) \qquad (2)$$

In newer simulators these strong requirements have been relaxed: the input processor of the simulator sorts the equations automatically. Nevertheless, up to now many simulators (sometimes named "block-oriented simulators") are based on the directed signal-flow paradigm. The internal mathematical description is an explicit system of differential equations (ODE):

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{3}$$

These simulators do not allow the formulation of "algebraic loops" in the modeling language or need special language constructs to force iteration procedure for solving the loop problem, see *Simulation Software - Developments and Trends*. The mathematical equivalent of an algebraic loop is a linear system of algebraic equations

$$0 = \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \tag{4}$$

which can not be solved directly by the usual ODE solvers of (3). These problems may be avoided by simulators which are constructed to solve the nonlinear DAE system (1a) with implicit numerical solution algorithms. Similar problems exist in the simulation of discrete systems: feedback loops with zero delay are not allowed in some simulators or require special iteration algorithms in more powerful discrete-event simulators.

Most simulation languages are used for textual model description but often pre-compilers generate these textual descriptions from graphical descriptions (e.g. block diagrams, signal-flow graphs, bond graphs, networks or circuit diagrams, state charts, PETRI nets). To combine textual and graphical model representations, some modern languages support both.

3. A Modeling Approach

3.1 Physical background

In this contribution we will focus on the common modeling of the plant and the controller. The main focus will be the description of continuous systems but we will also have a look at discrete and hybrid systems too. Continuous systems may be classified into:

- **conservative** physical systems: their variables are flow quantities and difference quantities, and compatibility constraints exist for these quantities;
- **non-conservative** physical systems: consisting of elements with inputs and outputs and only one type of quantities.
-

Physical domain	**Flow quantity**	**Difference quantity**
electrical	current	voltage
mechanical-translational	force	velocity
mechanical-rotational	torque	angular velocity
pneumatic	volume flow	pressure
thermal	heat flow	temperature

Table 1: Flow and difference quantities in different physical domains

A **flow quantity** (through quantity or 1-quantity) is measured at *one* point of the physical system (e.g. an electrical current or a hydraulic flow). A **difference quantity** (across quantity or 2-quantity) is measured between *two* points, e.g. a mechanical displacement or an electrical voltage. Table 1 shows some of the most important flow and difference quantities occurring in physical systems.

The terms “conservative” and “non-conservative” are very often used in the field of modeling technical systems, and must not be confused with the same term used in theoretical physics for the description of dissipative and irreversible systems! The term "conservative physical systems" is motivated in our context by the fact that compatibility constraints for flow and difference quantities are valid. They arise from the interconnection of components (or subsystems) and, therefore, the term “interconnective constraints“ is sometimes used. In electrical systems, these constraints are the well-known Kirchhoff's current law (KCL, node law) and voltage law (KVL, mesh law). Similar conservation laws for flow and difference quantities are also valid in many other physical domains.

This fact can be reflected in different classes of system models: conservative system models with conservative signals (flow and difference signals) and non-conservative systems (with non-conservative signals). We use the term "signal" as a model of all kinds of physical quantities which are able to exchange energy or information between the subsystems (in the context of VHDL, “signal” is there used in a very specific manner!). We will use the term "network" to describe a system model consisting of elements, connections between these elements, and conservative signals.

To emphasize the fact that these networks are models of physical systems in different domains, sometimes the term "generalized Kirchhoffian network" is used. Bond graphs (with effort and flow variables, see *Modeling and Simulation of Dynamic Systems Using Bond Graphs*) are closely related to networks. Their construction leads more or less directly to the formulation of an ODE system.

Non-conservative quantities are typical for controllers or digital signal-processing devices. Block diagrams (see Figure 2) or signal-flow graphs are appropriate models of these physical systems. Conservation laws do not exist on this level of abstraction. (Signals in discrete systems could also be considered as “non-conservative” but this notation is usually not applied in the discrete world.)

Languages for modeling complex heterogeneous systems have to fulfill a lot of **requirements** which are not fully supported by most of the existing modeling languages:

- mixed-domain capability (mechanics, electronics, fluidics, …);
- continuous and discrete *time* variables;
- continuous and discrete *value* variables;
- conservative and non-conservative quantities;
- structure as well as behavior may be described;
- the structure of the model should be similar to the structure of the physical system;
- models may be structured hierarchically;
- the behavior of elements or subsystems may be defined by the user (e.g. by differential

equations or automata / finite state machines FSM).

Before presenting two modeling languages which fulfill these requirements, we will discuss a very general physically-oriented modeling approach which is appropriate to model the controller as well as the plant.

3.2 The Multi-Port Approach

The *multi-port* modeling approach is illustrated in Figure 4.

- The complete system is decomposed into subsystems, the so-called multi-ports. The subsystems have ports (or terminals) to communicate with each other via their interconnections. Signals are separated into external signals between the subsystems, and internal signals within the subsystems.
- External signals may be *conservative* (flow signals e5, e6, e7, e8, e9; difference signals e1, e2 in the example) or *non-conservative* (e3, e4).
- The behavior of subsystems only depends on their port signals (and sometimes on some internal signals and their initial conditions).

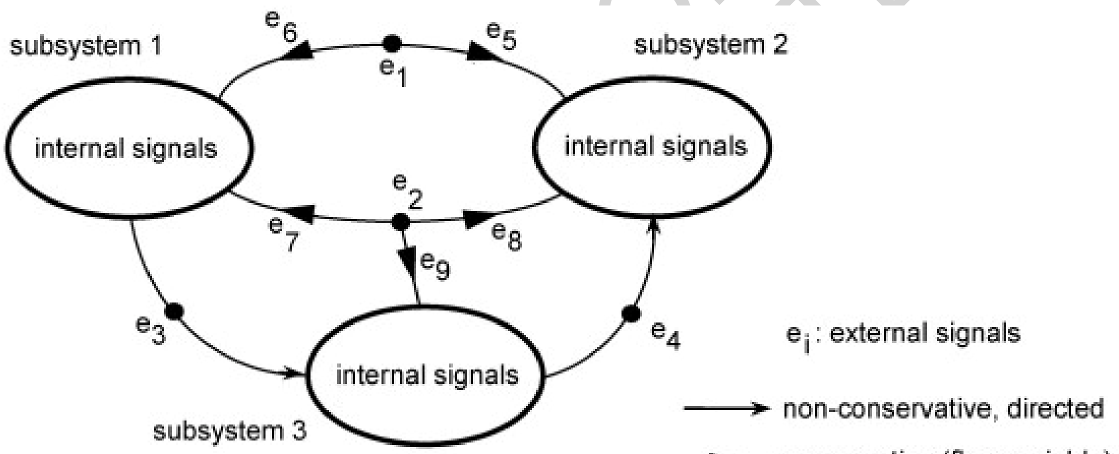


Figure 4: Multi-port modeling of heterogeneous systems

This is a very general but effective approach and it corresponds to the intuitive way of understanding complex systems: decomposing into more simple subsystems and calculating the interaction between the subsystems.

In electrical circuit theory, multi-*poles* and multi-*ports* may be distinguished as such subsystem models. But in our more general context this distinction between multi-poles and multi-ports is not substantial and, therefore, we will only use the term multi-port.

Each subsystem can be substituted by other subsystems with the same terminal behavior without any influence on the behavior of the rest of the system. The behavior of a subsystem can be expressed:

- implicitly by the connection of other subsystems (or primitives on the lowest level):

hierarchical structural refinement,

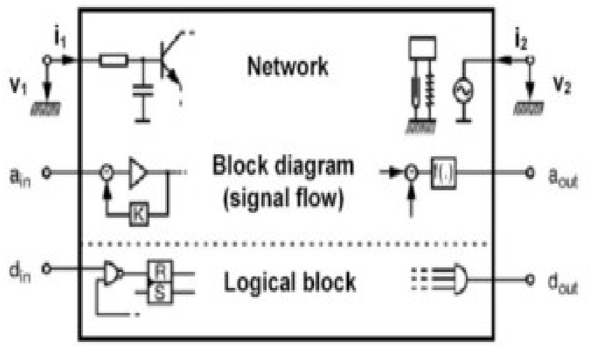
- explicitly by a set of equations (e.g. nonlinear differential-algebraic equations): "behavioral modeling" in a strict sense, or
- by a combination of both if the simulator has language constructs to formulate mixed structural-behavioral descriptions.

The key problem in setting-up the system equations is the description of the **terminal behavior** of the subsystems. All other equations are essentially constraints resulting from the connection of the subsystems, especially the node and mesh law for flow and difference signals on conservative terminals, and can be constructed automatically.

An external view of a multi-port and a system of associated model equations is shown in Figure 5. Terminal signals of the same type are assembled into vectors (written in bold types). The signals are vector-valued functions of time t. The terminal signals are divided into the following categories:

$\mathbf{v}_1, \mathbf{i}_2, \mathbf{a}_{\text{in}}, \mathbf{d}_{\text{in}}$ difference, flow, and non-conservative signals (analog and digital) which may be chosen independently as “input signals” to compute the response of the multi-port,

$\mathbf{v}_2, \mathbf{i}_1, \mathbf{a}_{\text{out}}, \mathbf{d}_{\text{out}}$ difference, flow, and non-conservative signals (analog and digital) which are dependent quantities and are calculated as “output signals”.



$$\mathbf{i}_1 = \mathbf{f}_1(\mathbf{v}_1, \dot{\mathbf{v}}_1, \mathbf{i}_2, \dot{\mathbf{i}}_2, \mathbf{a}_{in}, \dot{\mathbf{a}}_{in}, \mathbf{s}, \dot{\mathbf{s}}, \mathbf{d}_{in}, \mathbf{p}, t) \qquad \text{(E1)}$$

$$\mathbf{v}_2 = \mathbf{f}_2(\mathbf{v}_1, \dot{\mathbf{v}}_1, \mathbf{i}_2, \dot{\mathbf{i}}_2, \mathbf{a}_{in}, \dot{\mathbf{a}}_{in}, \mathbf{s}, \dot{\mathbf{s}}, \mathbf{d}_{in}, \mathbf{p}, t) \qquad \text{(E2)}$$

$$\mathbf{a}_{out} = \mathbf{f}_3(\mathbf{v}_1, \dot{\mathbf{v}}_1, \mathbf{i}_2, \dot{\mathbf{i}}_2, \mathbf{a}_{in}, \dot{\mathbf{a}}_{in}, \mathbf{s}, \dot{\mathbf{s}}, \mathbf{d}_{in}, \mathbf{p}, t) \qquad \text{(E3)}$$

$$0 = \mathbf{f}_4(\mathbf{v}_1, \dot{\mathbf{v}}_1, \mathbf{i}_2, \dot{\mathbf{i}}_2, \mathbf{a}_{in}, \dot{\mathbf{a}}_{in}, \mathbf{s}, \dot{\mathbf{s}}, \mathbf{d}_{in}, \mathbf{p}, t) \qquad \text{(E4)}$$

$$\mathbf{d}_{out} = \mathbf{f}_5(\mathbf{v}_1, \dot{\mathbf{v}}_1, \mathbf{i}_2, \dot{\mathbf{i}}_2, \mathbf{a}_{in}, \dot{\mathbf{a}}_{in}, \mathbf{s}, \dot{\mathbf{s}}, \mathbf{d}_{in}, \mathbf{p}, t) \qquad \text{(E5)}$$

Figure 5: General multi-port description

In many cases it is impossible to compute the terminal behavior based on terminal signals only. Subsystems may have *internal states* and, therefore, the introduction of a vector of additional signals $\mathbf{s}$ is necessary. The terminal behavior is also determined by the values of some parameters described by a parameter vector $\mathbf{p}$. On very wide assumptions, the terminal behavior of multi-ports can be given by the equations (E1)-(E5) in Figure 5.

Equations (E1)-(E4) are differential-algebraic equations (DAE). They can be solved by numerical solution algorithms. The last equation (E5) is meant more symbolically: in contrast to equations (E1)-(E4) for calculation of continuous variables which are described very similarly in almost all modeling languages, the description of discrete systems differs considerably in the different languages and is not always formulated in the syntactical form of an “equation”. This equation (E5) is mostly solved by *discrete-event* simulation algorithms which are very popular in general system simulation and in digital electronic simulation. The combined solution of continuous and discrete equations is known as hybrid simulation, analog-digital simulation, or mixed-signal simulation.

This multi-port modeling approach is closely related to **object-oriented modeling**. Unfortunately, there is no generally accepted concept of object-orientation in modeling and simulation. From the author's point of view, the following classification is useful:

- **Object-oriented modeling**: the construction of strictly hierarchical, modular structured models. This interpretation of object-oriented modeling was emphasized by Cellier.
- **Object-oriented simulation**: each subsystem is considered as an object. Each object has its own simulation algorithm (a "method"). All objects communicate via message passing, coordinate their behavior, and so the simulation of the entire system is carried out. A similar definition is: object-oriented simulation is the *concurrent* operation of different simulators without a global controller or a master simulator.
- **Object-oriented programming**: the application of programming languages like C++ or Smalltalk (and, more general, a powerful method to design complex software systems).

All these approaches exist independently from each other but they can be combined.
We will focus on *object-oriented modeling*. **Modularity** is guaranteed by the multi-port approach because the interaction between the subsystems occurs only via the signals on the terminals (the interface). There are no global variables, and side-effects are excluded. **Hierarchy** is achieved by the structural refinement mentioned above and by the hierarchy concepts of modeling languages used for behavioral modeling.

Other aspects of object-oriented modeling are:

- The structural similarity between a functional-oriented partitioning of the original physical system and the decomposition of the system model into subsystem models.
- The subsystem models should be combined without detailed knowledge of the user about the internal model realization (“plug and play” principle and information hiding).
- The user has to give a topological description of the interconnection of the subsystems and, eventually, to model the behavior of some components. Setting-up the equations which describe the behavior of the whole system is the task of the simulator (its input language processor), and not of the user.
- With the same interface description, different behavioral descriptions may be used to obtain appropriate degrees of accuracy of the models and to carry out **multi-level-simulation** (i.e. the simulation with models described on different levels of abstraction).

-
-
-

Bibliography

Ashenden P. J., Peterson G. D., and Teegarden D. A. (2002). *The System Designer's Guide to VHDL-AMS*, 906 pp., San Francisco: Morgan Kaufmann Publishers. [This book is one of the first comprehensive presentations of VHDL-AMS.]

Atherton D. P., and Borne P. (1992). *Concise Encyclopedia of Modeling and Simulation*, 553 pp., Oxford: Pergamon Press. [The book gives a global overview on continuous-value simulation: solution of differential equations, Z-transform for sampled-data systems, identification]

Banks J. (Ed.) (1998). *Handbook of Simulation*, 861 pp., New York: Wiley. [This is a very comprehensive overview of discrete simulators, simulation methods, and a short description of widely used simulators such as GPSS/H, SIMSCRIPT and SIMPLE++.]

Breitenecker F., Ecker H., and Bausch-Gall I. (1993). *Simulieren mit ACSL* (in German), 410 pp., Braunschweig: Vieweg. See also: http://acslsim.com.

Berge J.-M., Levia Oz, and Rouillard J. (Hrg.) (since 1995). *Current Issues in Electronic Modeling*, Dordrecht: Kluwer. [Many volumes comprising current developments in the area of modeling at all abstraction levels (10 volumes so far).]

1. Model Generation in Electronic Design (1995), 173 pp..

2. Modeling in Analog Design (1995), 163 pp. [The roots of VHDL-AMS are presented here.]

3. High-Level System Modeling: Specification Languages (1995), 178 pp.

4. High-Level System Modeling: Specification and Design Methodologies (1996), 208 pp.

5. Hardware Component Modeling (1996), 151 pp.

6. Meta-Modeling: Performance and Information Modeling (1996), 209 pp.

7. Object-Oriented Modeling (1996), 168 pp.

8. HW/SW Co-Design and Co-Verification (1997), 184 pp.

9. Models in System Design (1997), 167 pp.

10. Analog and Mixed-Signal Hardware Description Languages (1997), 178 pp.

Cellier F. E. (1991). *Continuous System Modeling*, 803 pp., New York/Berlin: Springer. [In this very inspiring book the basic concepts in modeling and simulation of continuous systems together with fundamental ideas of object-oriented modeling are presented.]

Christen E., and Bakalar K. (1999). VHDL-AMS - A Hardware Description Language for analog and mixed-signal applications. *IEEE Trans.* CAS-II, 46, 1263-1272. [This paper describes the rationales of VHDL-AMS and some of the basic language constructs.]

Elmqvist H., Mattsson S.E., and Olsson H. (2002). New methods for hardware-in-the-loop simulation of stiff models. *Proc. Modelica'2002*, Oberpfaffenhofen, Germany, 59-64.

(download from http://www.Modelica.org/Conference2002/papers.shtml). [This conference paper presents an approach to real-time simulation in the context of the Modelica language.]

Fritzson P. (2004). *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*, 897 pp., New York: Wiley. [This book is currently the most comprehensive introduction to Modelica.]

Groetker, T., Liao, Stan; Martin, Grant, and Swan, S (2002). *System Design with System C*, 233 pp., Boston: Kluwer.

Haase J. (2003). Rules for analog and mixed-signal VHDL-AMS modeling. *Proc. Forum on Design and Languages (FDL'03)*, Frankfurt/Main, Germany, 98-107. [Some typical modeling errors related to the numerical solution algorithms are discussed, especially the definition of simulation problems with correct syntax but without a solution.]

Heinkel U., Padeffke M., Haas W., Buerner T., Braisz H., Gentner T., and Grassmann A. (2000). *The VHDL Reference*, 438 pp., Chichester: Wiley. [This is a "cook-book" presentation of VHDL and a short introduction to VHDL-AMS.]

Huss S.A (2001). *Model Engineering in Mixed-Signal Circuit Design*, 180 pp., Boston: Kluwer. [In this book a guide to generating accurate behavioral models in VHDL-AMS is embedded into a general top-down design methodology.]

Karnopp D. C., Margolis D. L., and Rosenberg R. C. (1990). *System Dynamics: A Unified Approach*, 528 pp., New York: Wiley. [This book gives a fascinating overview on modeling dynamic systems with bond graphs.]

Koenig H. E., and Blackwell W. A. (1961). *Electromechanical System Theory*, 520 pp., New York: McGraw-Hill. [This book is a "classical" introduction to analogies and network modeling in different physical domains.]

Kasper R., and Koch W. (1995). Object-oriented behavioral modelling of mechatronic systems. *Proc. 3rd Conf. Mechatronics and Robotics*, Paderborn, Germany, 70-84.

Mann H. (1995). Multipole and multiport approach to mixed energy-domain systems. *Proc. 1995 IEEE Int. Symp. on Circuits and Systems,* Seattle, 676-679. See also http://icosym.cvut.cz/course/.

Mattsson S.E., Otter M., and Hilding E. (1999). Modelica Hybrid Modeling and Efficient Simulation. *Proc. 38th IEEE Conference on Decision and Control CDC'99*, Phoenix, Arizona, USA, 3502-3507.

Mueller W., Rosenstiel W., and Ruf W. (2003). *SystemC: Methodologies and Applications*, 362 pp., Boston: Kluwer. [This books presents a new modeling language SystemC which is very effective in high-level modeling of hardware/software systems and will be extended to mixed-signal modeling and simulation.]

Navabi Z. (1993). *VHDL - Analysis and Modeling of Digital Systems*, 389 pp., New York: McGraw-Hill. [This is an application oriented textbook on VHDL and the discrete-event simulation principle, but presents also interesting examples of automata and finite-state machine descriptions based on VHDL.]

Neul R. et al. (1998). A modeling approach to include mechanical microsystem components into system simulation. Paris: *Proc. Design, Automation & Test Conf. (DATE'98),* 510-517.

Pantelides C. (1998). The Consistent Initialization of Differential-Algebraic Systems. *SIAM Journal of Scientific and Statistical Computing*, 213-231.

Reinschke K., and Schwarz P. (1976). *Verfahren zur rechnergestuetzten Analyse linearer Netzwerke* (in German), 335 pp., Berlin: Akademie-Verlag. [In this book different methods for formulating the equations describing generalized networks are presented.]

Remelhe M.A.P. (2002). Combining Discrete Event Models and Modelica – General Thoughts and a Special Modeling Environment. *Proc Conf. Modelica'2002,* Oberpfaffenhofen, 203-207.(http://www.Modelica.org/Conference2002/papers.shtml).

Rumbaugh J., Blaha, M., and Premerlani, W. (1991). *Object-Oriented Modeling and Design*, 512 pp., Englewood Cliffs: Prentice Hall. [A "classical" book on object-oriented programming, more focused on software design than on physical modeling.]

Saleh R., Jou S.-J., and Newton A. R. (1994). *Mixed-Mode Simulation and Analog Multilevel Simulation*, 313 pp., Dordrecht: Kluwer. [Basic ideas of analog as well as mixed-signal simulation, especially in electronics, are presented in this book.]

Schoen J.M. (Edt.) (1992). *Performance and Fault Modeling with VHDL*, 415 pp., Englewood Cliffs:

Prentice Hall. [This book presents some unconventional applications of VHDL in general system simulation.]

Schwarz P. (2000). Physically oriented modeling of heterogeneous systems. *Mathematics and Computers in Simulation* 53, 333-344.

Schwarz P., and Haase J. (1998). Behavioral modeling of complex heterogeneous microsystems, Lausanne: *Proc. 1st Intern. Forum on Design Languages (FDL'98)*, 53-62, or in: Mermet, J. (Edt.) (2001). *Electronic chips & Systems Design Languages*, 314 pp., Dordrecht: Kluwer 17-30. [The paper describes the application of modeling languages in micromechanical systems simulation.]

Tiller M. (2001). *Introduction to Physical Modeling with Modelica*, 366 pp., Dordrecht: Kluwer. [This was the first Modelica textbook and contains a presentation of basic ideas and many application examples.]

Tonti E. (1976). The reason for analogies between physical theories. *Appl. Math. Modelling* 1, 37-50. [This is a "classical" paper on modeling in different physical domains.]

Villar, E. and Mermet, J. (Editors) (2003). *System Specification & Design Languages*, 355 pp., Dordrecht, Kluwer. [Some papers in this book describe the application of different mixed-signal modeling languages: Modelica, VHDL-AMS and SystemC-AMS as well as mixed-signal simulation algorithms.]

Voigt P., and Wachutka G. (1998). Electro-fluidic microsystem modeling based on Kirchhoffian network theory. *Sensor and Actuators* A 66 (1998)1-3, 6-14.

Wachutka G. (1995). Tailored modeling: a way to the 'virtual microtransducer fab' ? *Sensor and Actuators* A 46-47 (1995), 603-612. [The concept of (generalized) Kirchhoffian networks is discussed in these two papers.]

Wellstead P.E. (1979). *Introduction to Physical System Modelling*, 288 pp., London: Academic Press. [This book describes the modeling of physical systems with networks and bond graphs.]

Instead of publications, Internet site addresses should be used as references to simulators or other tools as well as to standardized modeling languages. Owing to the rapid development of such software (and the occasionally rapidly changing tool names and trade marks), the best way to up-to-date information is to check the homepages of the software providers: Simulators which are very useful in control system design are described on the following web sites:

http://www.analogy.com [Saber and modeling language MAST]

http://www.ilogix.com [Graphically oriented simulation environment Statemate]

http://www.cadence.com [Mixed-signal simulators, VHDL-AMS]

http://www.dynasim.se [Dymola, for Modelica simulation]

http://www.iti.de [Mechatronics and control system simulator ITI-SIM]

http://www.mathworks.com [Control system design and simulation environment MATLAB/SIMULINK]

http://www.ni.com/matrixx/ [Control system design and simulation environment MATRIXx]

http://www.mentor.org [Mixed-signal simulators, VHDL-AMS]

http://www.ansoft.com/products/em/simplorer/ [Mixed-signal simulator for physical systems, VHDL-AMS]

http://www.synopsys.com/ [Mixed-signal simulators, VHDL-AMS, Verilog-AMS]

Modelica: http://www.modelica.org/ . [Many links to Modelica-related publications and the Language Specification.]

VHDL-AMS (VHDL – Analog and Mixed Signal Extensions): http://www.vhdl.org/analog/ [This is the official website of the IEEE standardization working group.]

Biographical Sketch

Peter Schwarz received the diploma and the Ph.D. degree in electrical engineering from the Dresden University of Technology in 1964 and 1967, respectively. He worked in the Robotron Computer Company in Dresden and was responsible for research and development in a CAD group. From 1982 to 1991 he was the leader of the research group "Simulation" at the Central Institute for Cybernetics and Information Processes of the Academy of Sciences in Dresden. He was engaged in the development of a multi-level, mixed-signal simulator KOSIM which was used in industry and research institutes at that time. He received the Habilitation degree from the Dresden University of Technology in 1989. Since 1992 he has been working with the Fraunhofer Institute for Integrated Circuits, Design Automation Division EAS Dresden. He is the head of the Modeling and Simulation department with about 30 engineers, mathematicians, and physicists. His special interests are now multi-level, mixed-signal modeling and simulation of complex heterogeneous systems, web-based simulation and optimization, and knowledge transfer in life-long learning. Application areas are integrated circuit design, micro-electrical-mechanical systems (MEMS), telecommunications, mechatronics, and automation systems. He is member of IEEE and VDE.