Hardware/Software-Cosimulation for Mechatronic System Design

G. Pelz, J. Bielefeld, G. Hess
Gerhard-Mercator-University-GH
Faculty of Electrical Engineering
Dept. of Electron Devices and Circuits
Finkenstr. 61, D-47057 Duisburg
Germany

G. Zimmer
Fraunhofer-Institute IMS and
Gerhard-Mercator-University-GH
Faculty of Electrical Engineering
Dept. of Electron Devices and Circuits
Finkenstr. 61, D-47057 Duisburg, Germany

Abstract

This paper is about a new electro-mechanical modeling paradigm based on hardware description languages which paves the way to mechatronical HW/SW-Cosimulation. The strategy is illustrated by an example from automotive system design: a processor-controlled wheel suspension following BMW's electronic damper control. Its suitability in function and performance is shown by system simulation.

1. Introduction

If electronics has to be designed for mechatronic systems, the term 'hardware' includes electronics as well as mechanics which might be of macro or micro scale. We will focus on macro-mechanics here, but most of our work applies to micro system technologies as well. Our primary goal is to support the design of electronics and software running on (embedded) controllers while keeping track of the mechanic's influence. Especially, aspects of HW/SW-Codesign should be able to be studied, e.g. deciding to do signal filtering in software or electronical hardware. In that vein, the most important objective is to preserve the hardware description language (HDL) based methodology for electronics design. Several problems arise when mechanical components are to be included in a HW/SW-Cosimulation.

- Mechanics and electronics time constants may differ by orders of magnitude. This is especially true if macro-mechanical components are taken into account. For instance, the eigenfrequency of a car's wheel is in the range of about 10 hertz. In electronics, we are talking about megahertz. One might argue that these differences would dispense us from investigating the dynamic interference between electronics and mechanics. The contrary is true. Extensive controlling algorithms are run on (embedded) controllers. Their execution time again is in the range of milliseconds, giving rise to dynamic interaction between mechanics and electronics.

- For mechanical components, appropriate models have to be supplied to include them into an electronics simulation. Within this paper we will stick to multibody mechanics with concentrated rigid bodies. Distributed models described by finite elements or partial differential equations are not discussed. Even with this restriction, it is difficult to address the vectorial nature of mechanics. A velocity in mechanics is a quantity which cannot be directly mapped onto a voltage or current, since its direction would not be taken into account. Additional to that, the system description should be accomplished by a minimal number of state variables to enable an efficient simulation.

- Even if a proper model has been developed, the question remains, how to incorporate it into a coupled simulation of mechanics and electronics.

In the following, section 2 discusses previous work on HW/SW-Cosimulation while section 3 outlines our approach. Section 4 introduces a demonstrator for mechatronic HW/SW-Cosimulation. In sections 5 and 6 several modeling prerequisites are developed, i.e. the development of efficient processor models running software and a straight-forward way from mechanical multibody systems to analog hardware description language models. Section 7 comments on mechatronic HW/SW-Cosimulation and shows some results for the demonstrator. Finally, the paper is concluded with a short summary in section 8.

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00

2 Previous Work

Many approaches for HW/SW-Cosimulation have been proposed in the last years [1, 2, 3, 4, 5, 6, 7].

1. The most straight-forward way is to model the system's processor using a hardware description language and to execute its software by simulating the model [1, 2]. The required amount of CPU-time for simulation is large but tractable if the respective model is consequently formulated in a behavioural manner.

2. In [3], simulator cores dedicated for certain processors are used to speed up simulations. The improved simulation performance has to be paid with a loss of flexibility.

3. Another method to tackle the performance problem is to further abstract the descriptions of hardware and software [4]. Here, both hardware and software are modeled by a 'process' structure. This makes sense if synthesis tools are available and applicable to generate software and hardware from these high-level descriptions. This is often not the case in mechatronic system design.

4. In [5], a workstation runs a simulator for the hardware part and directly executes a program representing the software part. This can be done, if target architecture and simulation workstation are of equal type. Typically, the resulting timing between hardware simulation and software execution does not fit anymore. Thus, dynamic interference between hardware and software, as needed in mechatronic HW/SW-Cosimulations, cannot be properly assessed.

5. Another approach maps the software portion of the system to the programming language alike part of VHDL, while the hardware is modeled in classical manner with the same language [6]. Here, again the timing between hardware and software cannot be taken into account correctly.

6. All the above approaches are restricted to pure electronic systems. In [7] also mechatronic aspects are discussed, but the HW/SW-Cosimulation is directed to the hardware-in-the-loop strategy, which requires a physical testbed, say a teststand for shock-absorbers, for mechanical components. Often, it is possible to develop proper mechanics models. In these cases, a pure simulation strategy which takes into account software, electronics and mechanics is to be preferred.

3 The Approach

We believe that model development for components and their coupling is the key problem for HW/SW-Cosimulation of mechatronics systems. Thus, we propose a new electro-mechanical modeling paradigm which promotes the mapping of heterogeneous system models to analog and digital hardware description languages which can be cosimulated with existing simulators. This strategy is illustrated by an example from automotive system design: a processor-controlled wheel suspension following BMW's electronic damper control.

As detailed in section 2, only the first two strategies for HW/SW-Cosimulation are suitable to extend them to mechatronic systems. We consequently employ behavioural VHDL models to achieve the required simulation performance. For instance, the micro-processor is characterized by its instruction set, some internal registers, the interrupt handling and the external signal's timing. An ADC is modelled by its translation specification and the respective timing. Nothing else is modelled.

The vectorial nature of mechanics is handled by transforming the cartesian coordinates of mechanical systems into a set of new scalar variables - so-called generalized coordinates - which can be interpreted as state variables from system theory. If the number of generalized coordinates equals the number of degrees of freedom of the mechanical system, a minimal set of ordinary differential equations can be derived. Note that the respective kinematic structure has to be taken into account to allow a proper interpretation of the simulation results.

The formulation of the mechanics models is carried out in an analog hardware description language. In this way, the problem of mechatronic HW/SW-Cosimulation is mapped to the well-investigated problem of mixed-mode electronics simulation.

4 Demonstrator: Processor-Controlled Wheel Suspension

For illustration purposes, a processor-controlled car wheel suspension is modeled and subjected to a mechatronic HW/SW-Cosimulation, see Fig. 1. Our demonstrator follows BMW's electronic damper control [10]. Its objective is to achieve a better compromise between driving safety and comfort. A micro-controller with its software sets the damper's constant - and thus the damping force - depending on the road condition. For safety reasons, excitations in the eigenfrequency domain of car body (1 Hz) and wheel (10 Hz) require

larger damping forces. If the road excitation is not critical, lower damping forces lead to better comfort. The detection of safety-relevant vibrations is done by sensing the car body acceleration. The resulting signal is processed and fed into the micro-controller, which decides on the damping constant. The mechanical part in this system consists of wheel, axle, spring, car body and a shock-absorber with switchable damping. Electronics is given through the acceleration sensor, ADC, DAC, signal processing and the micro-controller running the software.

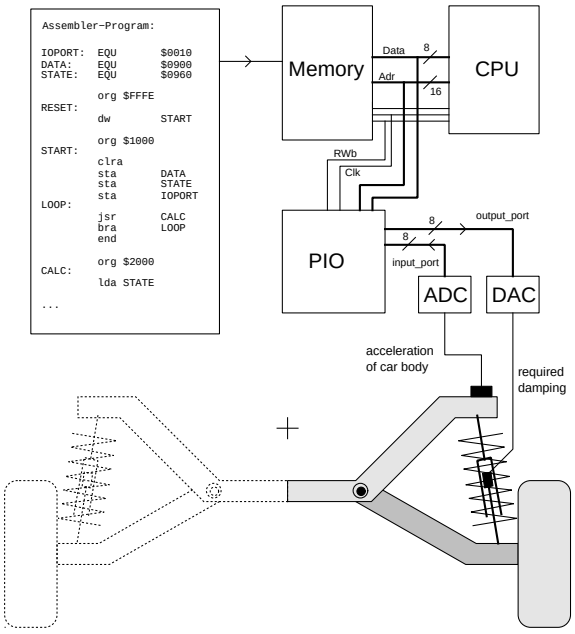

Figure 1. Processor-controlled semi-active car wheel suspension.

5 Electronics Modeling

The controller model's performance is very critical for the overall simulation time consumption, since the algorithm implemented in software is executed many times during the simulation. This calls for a behavioural model, since this type of model requires much less simulation time, than any kind of structural model. While the internal timing of the controller including processor, memory and I/O-circuitry can be neglected, it is important to model the external bus and - depending on the system - the interrupt system correctly, to enable proper simulation of the software's timing and function. This in turn is indispensible for simulating the mechatronic system's behaviour. Typical functional models, written in some other programming language, often ignore these points and are thus not suitable, even if they can be integrated into the simulation process.

To obtain the required simulation speed this model should execute as fast as possible. The basic idea goes as follows:

- Sequential VHDL-Statements execute as fast as code written in any other programming language on the same computer.
- These statements are executed every time the VHDL-core resumes the process.

$\Rightarrow$ Write the processor model as a single process, which is only scheduled if neccessary.

The only way to decrease the execution-time of the controller-model, is the use of another programming language, that better utilizes the instruction set of the computer used for the simulation. But it is not sure, that the improvement in simulation-speed becomes significant, since the procedures required to interact with the simulator-core introduce an overhead, which is at least in the same order of magnitude as the overhead introduced by the limitations of the VHDL-compiler.

The demonstrator contains a model of a Motorola 6805 compatible microcontroller which consists of 800 Lines of VHDL-code. The model was verfied by running the testprograms for the existing microcontroller and was completed in a few days.

In addition to the model of the microcontroller, a program to be executed is necessary. The obvious source of this program are the executable files produced by the software-development system used for this controller. These files are loaded into the controller model whenever the simulation starts.

6 Mechanics Modeling

In the following, a straight-forward way from mechanical multibody systems to HDL models is shown. First, reasonable abstractions have to be identified, e.g. motivated by symmetry considerations of the geometrical configuration. Next, the kinematic structure of the system is fed into some mechanical engineering modeling tool which produces symbolic equations. Several tools are available, see [8]. The resulting equation set consists of nonlinear, ordinary, differential equations. Finally, the equations are formatted in an analog HDL. The described flow will be illustrated in the following

by performing the mechanics modelling for the demonstrator introduced above.

In modern cars, the two axles are almost equally loaded. Therefore, only one axle has to be modelled, since the motion of front and rear axle are nearly decoupled. Even more, if the road surface is of equal shape for left and right wheel, just a quarter of a car has to be taken into account for symmetry reasons. For a wheel suspension, vertical dynamics is the major effect. Therefore, the wheel suspension system can be abstracted to a two-mass system, consisting of rigid, concentrated bodies, see Fig. 2, and considering just the vertical motion of the bodies. In this way, the generalized coordinates which determine the system state can be identified in a quite natural way; they are chosen to be the vertical coordinates of the two bodies. The tire is modelled by a simple spring; its damping effect can be neglected. The resulting two mass model is a common abstraction for investigating a suspension's dynamics [11].

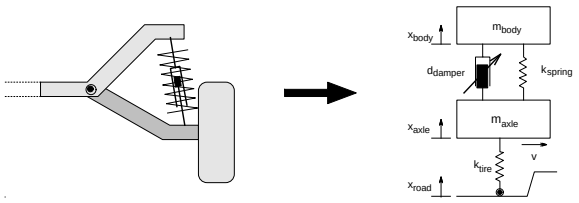

Figure 2. Abstraction of wheel suspension for mechanics modeling.

Many modeling tools for multibody systems have been devised, see [8]. We employed 'TSi Dynamics' [9] to do the basic mechanics modeling. TSi Dynamics is a Mathematica package and uses its comfortable symbolic equation handling opportunities. First, the kinematic structure as shown in Fig. 2 has to be formulated in terms of rigid bodies and joints. Two rigid bodies are taken into account: the car body (or chassis) and the axle including the wheel. The joints assure that the motion of the bodies is restricted to vertical movement. To take into account the effects of suspension spring, shock absorber and tire spring, their potential energy is formulated. Finally, TSi Dynamics automatically derived the following set of ordinary differential equations:

$$\begin{aligned} m_{body}\ddot{x}_{body} &= k_{spring}(x_{axle} - x_{body}) + \\ & \quad d_{damper}(\dot{x}_{axle} - \dot{x}_{body}) \end{aligned}$$

$$\begin{aligned} m_{axle}\ddot{x}_{axle} &= k_{tire}(x_{road} - x_{axle}) - \\ & \quad k_{axle}(x_{axle} - x_{body}) - \\ & \quad d_{damper}(\dot{x}_{axle} - \dot{x}_{body}) \end{aligned}$$

where m denotes masses, k spring constants, d damper constants and x vertical positions. Some additional features inject nonlinearities, e.g. the shock absorbers damping's constant depends on whether it is in push or pull mode and of course the controller's requirements. Finally, the model is easily formulated in an analog hardware description language. We chose MAST® [1] for this purpose.

Note that solvers for analog HDLs might run into problems for more complex mechanics models. Numerical integration is still a topic in mechanical engineering. Often, the solution of sets of mechanical differential equations depends on the clever choice of integration scheme and initial conditions.

7 Mechatronical HW/SW-Cosimulation

Mechatronical HW/SW-Cosimulation has to overcome several difficulties. The coupling between hardware and software resembles the one faced in classical HW/SW-Cosimulation. We propose the application of a black-box model on the level of the controller. Especially, software is not modelled which dispenses us from remodelling for every software update.

The cosimulation of digital and analog system components is a necessary prerequisite for mechatronics HW/SW-Cosimulation. Typically, digital simulation is carried out in an event-driven manner. The output of a component is recalculated if and only if the inputs have changed. The advantage of this mode of operation is due to the fact that just about 5 or 10 % of a digital circuit is active at a time. In contrast to this, analog components are simulated by solving sets of ordinary differential equations. Synchronization and efficient data exchange between analog and digital simulation have been investigated for years. Lots of so-called mixed-mode simulators have been developed in the last years.

The cosimulation of mechanical and electronical components requires a generalized view and description of analog components. If we stick to multibody systems here, the mechanics modeling is mostly performed by Lagrange's approach or by the Newton-Euler method [8]. Some of the respective tools produce sets of symbolic ordinary differential equations. Others do not,

[1] MAST is a registered trademark of Analogy Inc.

since more complicated systems tend to result in exploding sets of equations. These tools alter the system matrices for every time step. This mode of operation cannot easily be mapped to analog electronics simulators. Nonetheless, for system simulation simplified models can be employed, since the task is to verify the system's - not the component's - behaviour. Thus, we formulate symbolic mechanics models in an analog hardware description language which can be directly simulated together with other analog electronics.

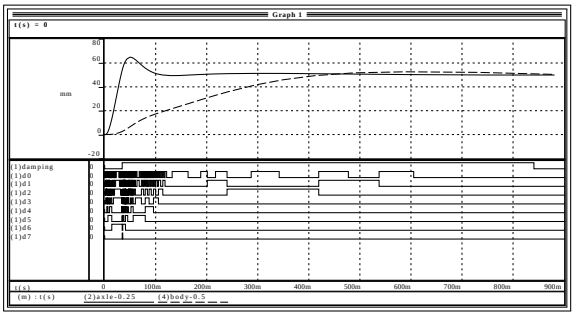

Figure 3. Simulation of semi-active car suspension: top traces show vertical, relative movement of axle (solid line) and car body (dashed line); damping values: '0'=soft damping, '1'=hard damping; bottom traces 'd0'-'d7': digital value of body acceleration.

By system simulation of a processor-controlled semi-active car wheel suspension, we show the method's suitability in function and performance. The simulation is carried out on a coupled simulator which is commercially available. It contains two simulator cores, i.e. Leapfrog® [2] (VHDL, digital) and Saber® [3] (MAST, analog) which are well-known from electronic and microelectronic circuit design. The results for an axle of a car running over a roadstep 5 cm high are shown in Figure 3. At the top, the vertical, relative movement of axle and body are depicted. At the bottom, the digital 'damping' signal shows the damping value: '0' = soft damping, '1' = hard damping. The signals 'd0' to 'd7' represent the digital value of the current body acceleration. After running over the step, wheel and car body are accelerated vertically. The car body's acceleration is fed into the controller which switches to hard damping after some time. The relative, vertical postions of car body and wheel increase to 50 mm which corresponds to the height of the step. Finally, if the car body acceleration has decreased under a certain limit, the controller switches to soft damping again.

[2] Leapfrog is a registered trademark of Cadence Design Systems.

[3] Saber is a registered trademark of Analogy Inc.

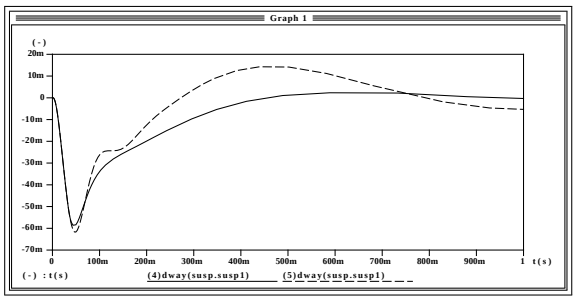

Figure 4. Shock absorber compression versus time, with switching the damping constant (solid line), without switching (dashed line), unit = m.

Fig. 4 shows results of the same simulation run and depicts the compression of shock absorber and suspension spring. After riding over the road step, the damper is compressed by about 58 mm, as depicted by the solid line. Some time later, the suspension spring has pressed the shock absorber into its previous position. For comparison purposes, the dashed line shows the shock absorber's compression without switching to hard damping. The simulation was carried out on a SUN Sparc 20 and took 63 CPU-seconds for the digital part and 37 CPU-seconds for the analog part. The digital time encloses the simulated execution of about 380000 assembler statements. Analog and digital fraction can be simulated on different workstations to accomplish a further speedup.

8 Summary

We proposed a new modeling paradigm for mechatronical HW/SW-Cosimulation which is based on hardware description languages. In this way, various domains, i.e. software, digital and analog electronics and mechanics can be cosimulated without developing new simulation tools. Moreover, model exchange between simulators is promoted.

Several modeling prerequisites were developed, the development of efficient controller models running software and a straight-forward way from mechanical multibody systems to analog hardware description language models. All these models are combined in a mechatronical HW/SW-Cosimulation suitable to as-

sess the dynamic interference between software, electronics and mechanics. Especially, the simulation of function *and* timing of software allows a closer look on the system behaviour. The system's software itself is not modeled; it can be directly fed into the controllers memory. Thus, software updates do not require to remodel the system.

The proposed strategy for mechatronical HW/SW-Cosimulation enables a thorough system analysis and is thus an important prerequisite for mechatroncal HW/SW-Codesign. For example, trade-offs, e.g. performing signal filtering in hardware or software, can be investigated.

References

[1] K. Buchenrieder, J.W. Rozenblit, "Codesign: An Overview", in: Codesign Computer-Aided Software/Hardware Engineering, IEEE Press 1994, 1-15 (also available in Proc. 1st Int. Workshop on Software/Hardware Codesign 1992)

[2] A.W. Both, B. Biermann, R. Lerch, Y. Manoli and K. Sievert, "Hardware-Software-Codesign of Application Specific Microcontrollers with the ASM Environment", Proc. EuroDAC 1994, 72-76

[3] R.K. Gupta, C.N. Coelho and G. De Micheli, "Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Software Components" Proc. 29th ACM/IEEE Design Automation Confenrence, 1992, 225-230

[4] D.E. Thomas, J.K. Adams and H. Schmit, "A Model and Methodology for Hardware-Software Codesign", IEEE Design & Test of Computers, Sep. 1993, 6-15

[5] D. Becker, R.K. Singh and S.G. Tell, "An Engineering Environment for Hardware/Software Co-Simulation", Proc. 29th ACM/IEEE Design Automation Confenrence, 1992, 129-134

[6] W. Ecker, "Using VHDL for HW/SW Co-Specification", Proc. EuroDAC 1993, 500-505

[7] H.-J. Herpel, N. Wehn and M. Glesner, "Computer-Aided Prototyping of Application-Specific Embedded Controllers in Mechatronic Systems" in: Codesign Computer-Aided Software/Hardware Engineering, IEEE Press 1994, 425-442 (also available in Proc. 1st Int. Workshop on Software/Hardware Codesign 1992)

[8] W. Schiehlen (Editor), "Multibody Systems Handbook", 1990, Springer-Verlag, Berlin Heidelberg.

[9] TSi-Dynamics User's Guide, Techno-Sciences Inc.

[10] D. Hennecke, B. Jordan, and U. Ochner, "Elektronische Dämpfer Control - eine vollautomatisch adaptive Dämpfkraftverstellung für den BMW 635 CSi", ATZ Automobiltechnische Zeitung 89, 1987, 471-479 (in german)

[11] G. Roppenecker, "Fahrzeugdynamik: Grundlagen der Modellierung und Regelung", Automatisierungstechnik 42 / 10, 1994, 429-441 (in german)

[12] J. Bielefeld, G. Pelz and G. Zimmer, "Analog Hardware Description Languages for Modeling and Simulation of Microsystems and Mechatronics", Proc. 3. Conf. on Mechatronics and Robotics, Paderborn / Germany, B.G. Teubner, Ed. J. Lueckel, Stuttgart, 1995, 85-92

[13] G. Pelz, J. Bielefeld and G. Zimmer, "Model transformation for coupled electro-mechanical simulation in an electronics simulator", Springer Journal on Microsystem Technologies, Vol. 1, No. 4, Sep. 1995, 173-177

[14] G. Pelz, J. Bielefeld, F.-J. Zappe and G. Zimmer, "Simulating Micro-Electromechanical Systems", IEEE Circuits and Devices Magazine: Simulation and Modeling, Editors: R. Saleh und A. Yang, March 1995, 10-13