IEEE PARELEC 2004 Dresden, 7.-10.9.04, ISBN 0-7695-2080-4, 114 - 119

Using ModelSim, Matlab/Simulink and NS for Simulation of Distributed Systems

Uwe Hatnik, Sven Altmann
Fraunhofer Institute for Integrated Circuits, Branch Lab Design Automation
Zeunerstraße 38, D-01069 Dresden, Germany
phone +49 (0)351-4640-711/735, fax +49 (0)351-4640-703,
e-mail: Uwe.Hatnik@eas.iis.fhg.de, Sven.Altmann@eas.iis.fhg.de

Abstract

In this paper a flexible concept for simulator coupling is introduced which permits the whole system simulation of distributed systems, e.g. of complex communication systems. In our approach we use different languages (Matlab-Code, VHDL, oTcl, C/C++) and combine the specialized simulators Matlab/Simulink, ModelSim and the network simulator NS-2. Matlab/Simulink and ModelSim are shipped with a powerful foreign language interface and for NS-2 we developed a solution for including user written functions. The simulators are extended with a special coupling component implemented in C/C++. The simulator coupling bases on TCP/IP sockets, so that the simulation of the model parts can run on different host computers using different platforms.

1. Introduction

In the past years modern transmission and communication technologies (e.g. WLAN, Bluetooth, network processors, media server or internet services) changed the appearance of many applications and systems. Ubiquitous computing/networking is a new keyword and means that more and more applications in any parts of life become part of distributed systems. For example the components of modern home entertainment systems use local home networks so that several devices use a central media device. Furthermore, medical applications (telemedicine) connect doctor and patient via wide area networks like Internet. In industrial applications, automation components such as sensors, actuators and process controllers also communicate via networks. Such networks are classical fieldbus systems, but newly Industrial Ethernet gains in importance.

The modern communication networks are very complex heterogenous systems realizing local and world-wide data communication using different networks and protocols with a specified quality of service. Such communication systems consist of several devices e.g. different user devices, from powerful personal computers to small mobile phones or a **P**rogrammable **L**ogic **C**ontroller (PLC) in a factory automation network as shown in Figure 1. Any device can communicate with any other device using various net services.

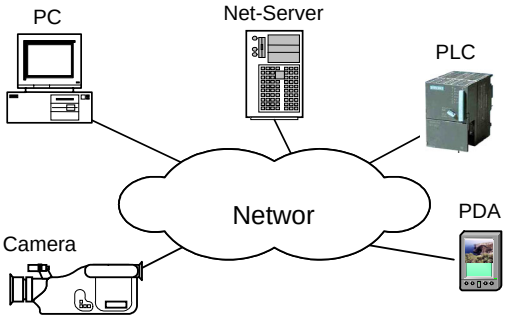

Fig. 1: Heterogenous network

One of the main challenges for the configuration and structuring of such heterogenous distributed systems is to guarantee the specified quality of service with a minimum of costs. The simulation of the whole system is a possibility to support the design process [6]. By using a simulation the internal functionality and the interactions of embedded components can be validated before their cost-intensive installation into prototypes is accomplished. The simulation of the entire system offers the chance of an optimal adjustment of the system configuration, e.g. control algorithms, special acceleration hardware, protocols, and parameters.

2. Simulator coupling

At present a single simulator processing the different continuous and discrete components, protocols and algorithms is not known. On the other side many continuous and discrete models in special tool boxes exist at the market. For both the combination and simultaneous simulation of these existing models we offer the coupling of appropriate simulation tools. The simultaneous simulation of continuous processes, digital electronic units and communication networks allows the developer to analyze various design variants and to optimize the overall system design. Furthermore, we provide a powerful simulation framework for the design of new components.

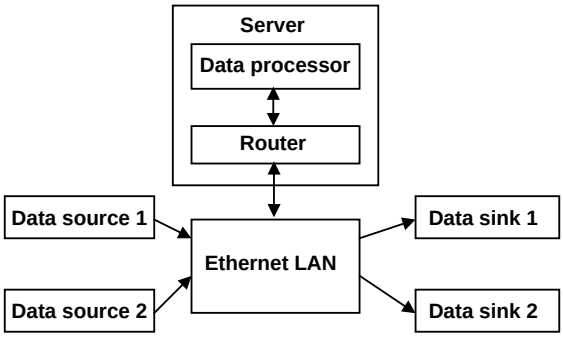

Fig. 2: Abstract view on a distributed system

An example for a distributed mixed mode network is shown in Figure 2. In this example separate data streams are generated by two data sources and are sent through a Ethernet network (LAN) to a central server which consists of a router and a data processor part. The router switches between the incoming and outgoing data streams of the data processor part. Data processing is done by several algorithms (not shown in the figure). The processed data streams are sent through the network to the data sinks.

2.1. Coupling structure

Figure 3 shows how the abstract scenario is mapped on a co-simulation environment consisting of appropriate simulators. The modelling and simulation of the data transfer over the communication network is done with NS-2 [11], which operates as the master simulator. NS-2 provides various network types (e.g. LANs or WANs), protocols and application models (e.g. programs which use HTTP or FTP) [5]. In our example we instantiate an Ethernet model, but the approach and the framework are not restricted to this scenario.

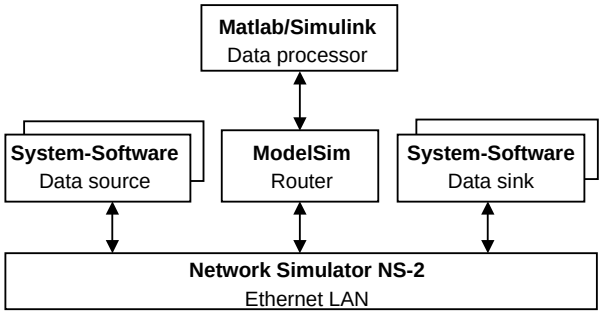

Fig. 3: Simulator coupling structure

The data streams are generated and consumed by C-applications connected to the NS-2 Ethernet network model. The data processor server components are simulated with ModelSim [9] and Matlab/Simulink [8]. ModelSim contains a VHDL model of the digital router components, so that several clients (data sources) can contact the server simultaneously. Further functions could be added, e.g. parts of the network protocol or data compression algorithms. ModelSim passes the NS-2 simulation time to the Matlab/Simulink simulator which performs analog as well as digital signal processing algorithms for instance.

The advantage of this approach is the application of specialized simulation tools for the subsystems and a modular coupling structure, so that various applications can be modeled.

2.2. Component Network Simulator NS-2

NS-2 is a tool for modelling and simulating communication networks transferring data from sources to sinks on the basis of a specified network protocol. In addition several abstract traffic generators can be included for performance and throughput analysis of the network.

Abstract client and server models inside the network model produce a basic load. These abstract models create or consume packages and can be described by some parameters like package size and package distribution functions. Models can only be developed using oTcl and C++. Other languages are currently not supported. We developed an interface to couple NS-2 with further simulators or other software components.

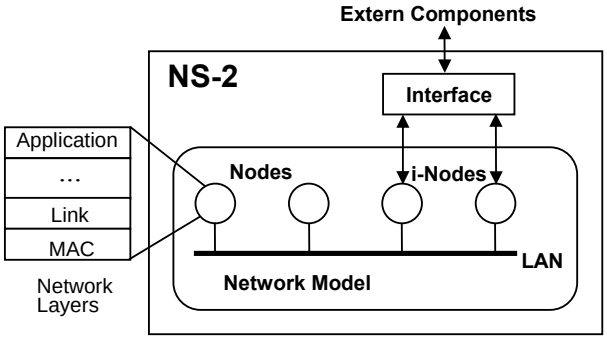

Fig. 4: NS-2 model with interface to external components

Figure 4 shows a simple model, including a Local Area Network (LAN, e. g. Ethernet) and four nodes (participants). Normally, each node contains the necessary models of the network stack (e.g. MAC Layer, Link Layer and Application Layer). The application models of the nodes exchange data packages using the LAN model. The LAN connects the nodes. To connect this network scenario with extern parts we developed special interface nodes (i-nodes), which couple the node models with the simulator interface and allows event and data exchange. The interface module is connected with

the internal simulation algorithm, so that NS-2 can work as master simulator to control extern components.

The inheritance mechanism of C++ allows a very flexible enhancement of the existing nodes, so that the interface connection can be established on different network layers.

2.3. Component User Mode Linux

In some cases, the integration of real system software is very advantageous, e.g. to get real stimuli data or to test protocol algorithms etc. The software component can be a small library but also a complete operation system including applications.

Figure 5 shows an example based on two User Mode Linux Systems. Because such Linux systems are executed like normal user processes, several systems can run on one computer. They communicate via a virtual network module, where functionality is replaced with the network simulator NS-2. The Linux systems send and receive real data packages, which pass the network models simulated by NS-2. Thereby NS-2 only uses address information of this packages, not the transmitted payload data. On this way it is possible to use already existing network models, provided by the NS-2 libraries. Furthermore, additional components which use the transmitted data can be coupled with NS-2, e. g. a VHDL simulator.

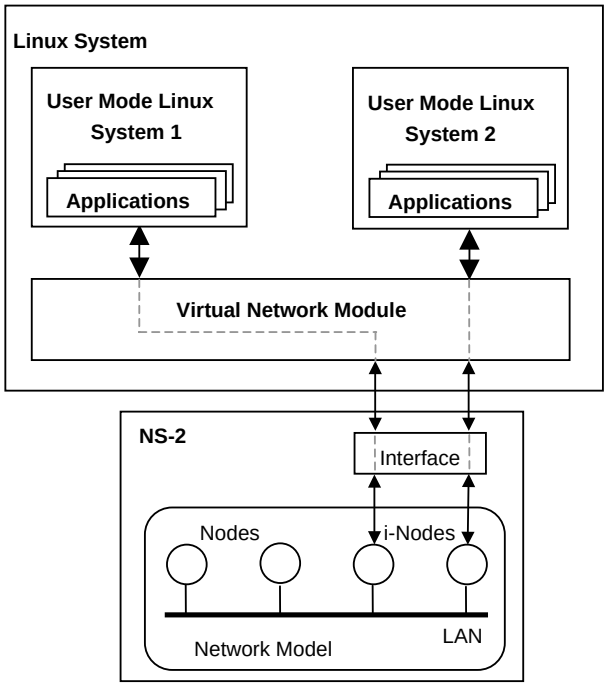

Fig 5: Co-Simulation of NS-2 and external system software

2.4. Component ModelSim

ModelSim is well suited to simulate digital components based on the hardware description languages VHDL (**V**ery High Speed Integrated Circuits **H**ardware **D**escription **L**anguage) [4] [7] and Verilog.

Generally, a VHDL model consists of several blocks, each defined by an interface (entity) and dedicated architectures. Entities are described by unidirectional and bidirectional ports, architectures can be implemented by a behavioural or a structural description. The behavioural description on an abstract VHDL modelling level allows using datatypes and programming techniques known from other programming languages. On structural level, a VHDL model mostly consists of basic logic blocks. Both description types can be merged.

In our example we apply VHDL to model the router block for the data streams sent and received via the Ethernet network. The behavioural model is depicted in figure 6.

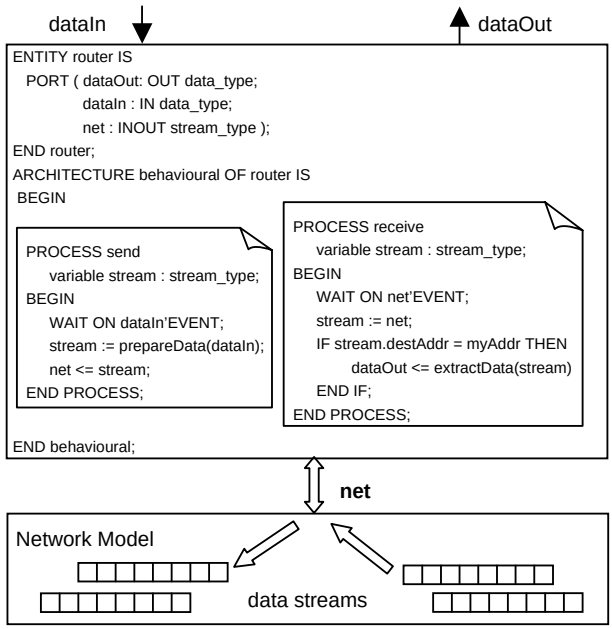

Fig. 6: VHDL model of router block

Each data stream contains some header information like source and destination address, CRC, number of payload bytes and so on. Due to the received destination address the router decides whether the stream is accepted or rejected. In acception case the payload data will be extracted and transferred to the signal processing algorithm which is modelled in Matlab/Simulink. Reversely, data received from the algorithm will be prepared with the required header information. Afterwards, the data will be sent to the dedicated recipients via the network.

In further implementation steps, the abstract behavioural VHDL description can be refined to implement the router as an ASIC for example. For all refinement steps, the co-simulation environment may not be changed. It can be used in the same manner to verify the correctness of an implementation step of the router block.

2.5. Component Matlab/Simulink

The simulator Matlab/Simulink allows a comfortable modelling and simulation of continuous or mixed discrete continuous model parts. Matlab/Simulink acquires wide application areas. For instance it is utilizable to develop a signal processing algorithm in media scenarios or to build a detailed process model in automation applications.

The tool package consists of the closely intermeshed parts Matlab and Simulink. Both represent solvers for mathematical matrix operations and differential equations, supplemented by visualization utilities and a wide range of model libraries.

A typical Matlab model is written as textual representation in the manner of a programming language extended by efficiently implemented mathematical operations. Each Simulink model consists of blocks, which also could be organized hierarchically. These blocks are connected by unidirectional signals. The Matlab/Simulink simulation algorithm can be roughly characterized as an extended dataflow algorithm.

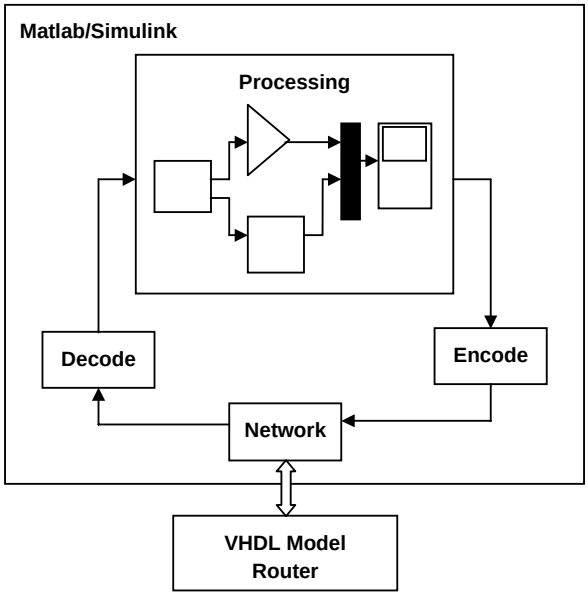

Fig. 7: Abstract Matlab/Simulink model of data processing

Figure 7 shows a simple principle for modelling data processing in Matlab/Simulink. The model consists of the processing block (containing various signal processing algorithms) and the encoder/decoder blocks for preparing data transfer. Data received from the network will be decoded and processed in various manner. The modified data stream is encoded and sent back to the network.

In our co-simulation environment, the network block is simulated by NS-2 extended by ModelSim as described above. The network block in figure 7 is replaced with a simulator coupling function.

2.6. Communication structure

The basic technology of the simulator coupling is a network wide connection via TCP/IP sockets. The advantage of this technology is that all simulators, each representing a model part (component) of the modelled system, may run on one single local computer or in the same manner on different host computers and platforms in a TCP/IP network. On this way a distributed simulation framework is constructed which also allows using a simulator in multiple instances. The environment works on Windows, Solaris an Linux platforms.

The simulator communication structure is illustrated in figure 8. Generally, all simulators may communicate with each other through a socket connection. In our example the simulators are connected by sockets as shown in figure 8. It is not necessary to use all simulators together. Coupling of only two or three simulators with each other is also possible and depends on the considered problem.

Synchronization between the simulators is done by a conservative method using simulator callbacks and blocking read/write socket routines [1][3]. Each simulator is extended with a special coupling component implemented in C/C++.

Matlab/Simulink provides several alternatives to include user specific model parts. We use the C-Mex interface [8] to integrate the coupling routines into a Simulink S-Function.

ModelSim provides the Foreign Language Interface (FLI) [9]. FLI allows the user to implement own functions, procedures or as used in our example complete architectures in C/C++. FLI also addresses internal simulator control routines like start, stop, schedule and much more.

For NS-2 an appropriate interface has to be developed since NS-2 has no C-code plugin. For that purpose, we extend the NS-2 simulation algorithm with a C-model interface called NSI (NS-Interface). At model level the C-model can be addressed via dedicated network nodes (i-nodes) as shown in figure 4.

The coupling routines are compiled with VC++ for Windows2000 and with gcc for Solaris and Linux platforms.

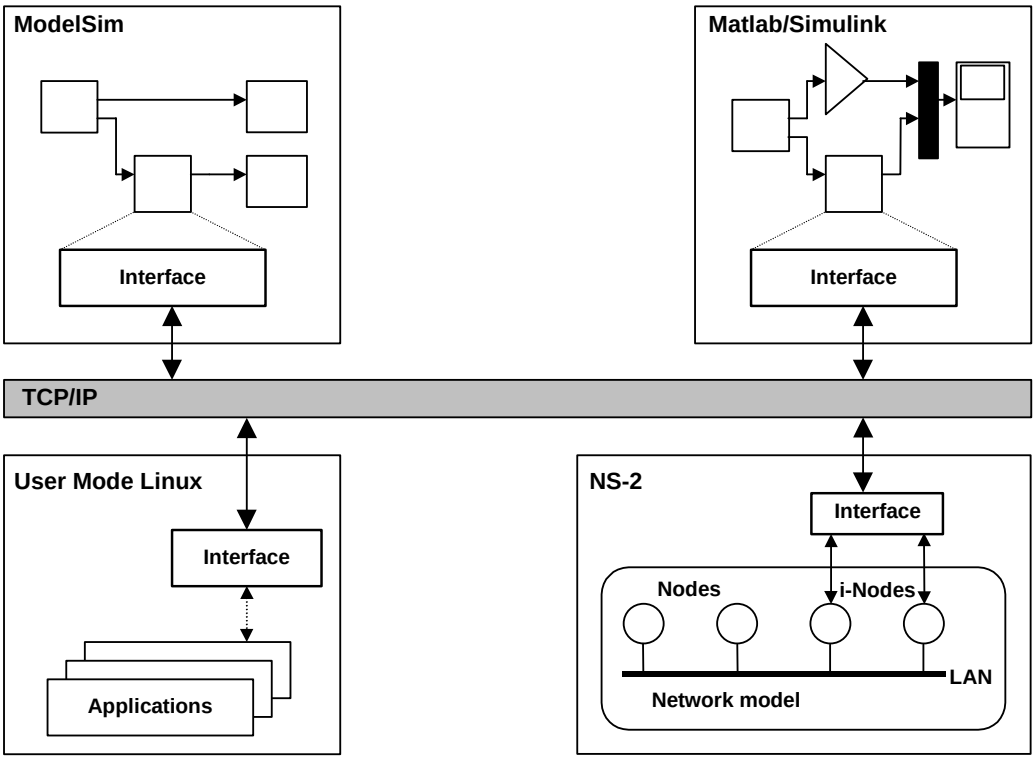

Fig 8: Simulator communication structure

3. Simulation

Each simulator is handled in the known manner using integrated design browser, source code debugger and other usefully features.

The evaluation of the simulation results is done alternatively online with the visualization tools by ModelSim, Matlab/Simulink and NS-2 and/or in a postprocessing step. The analysis of the simulation results is carried out with the following targets:

- check of the system functionality, performance, and security
- evaluation of the circuit power on behaviour and the system startup
- system dynamics in time-critical load cases and error situations
- optimization of selected operating parameters.

For postprocessing, extensive log files may be created from special monitoring components during the simulation. In our example a bus monitor connected to the Ethernet in NS-2 logs every transferred data telegram. These simulation traces are analyzed and visualized with standard office applications or own postprocessing tools as shown in Figure 9.

Figure 9 shows the current, mean and maximum values for latency of data stream transfers during the network. The plot clarified that the quality of service of the data transfer strongly depends on the current latency values for single packets and not only on the mean values for a whole data stream. The reason of this behaviour is the nondeterministic character of data transmission over the modelled Ethernet LAN. Because many participants are active on the same transmission medium some of the telegrams are destroyed by bus collisions and the transfer must be repeated. These repetitions result in the alternating curve for the current latency values and affects the quality of service significantly.

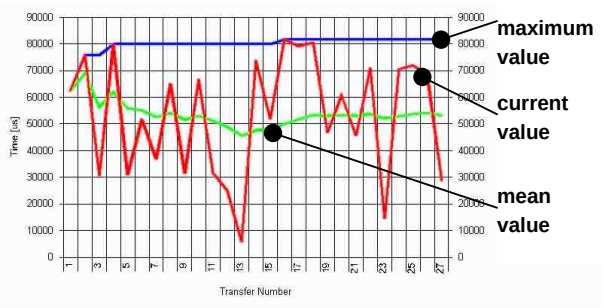

Fig 9: Latency analysis by postprocessing

4. Conclusions

We have shown that a relative simple interfacing with other simulators and system design tools is possible. The main advantage of our approach is that appropriated simulation tools for domain specific tasks can be combined within a common co-simulation environment to simulate complex heterogenous systems. Existing models run on their appropriate simulator and need not be redesigned for other less appropriate single simulation tools.

The presented solution is generic and flexible enough to be useful for a wide range of scenarios. One of its various applications of the coupling was illustrated by a distributed data transfer scenario. Another application is the simulation of distributed factory automation systems e.g. including a programmable logic controller, a fieldbus protocol and a process model [2][10].

The User Mode Linux interface supports the integration of a wide range of application software.

Connecting the simulators via TCP/IP sockets allows to distribute the simulation in a company network. On this way computing power and costly software licences could be used efficiently.

5. References

[1] Altmann, S.; Donath, U.: *Using ModelSim and Matlab/Simulink for System Simulation in Automotive Engineering,* Mentor Graphics Users' Group 2003 (MUG 2003), Sonthofen, October 9-10, 2003

[2] Altmann, S.: *Anwendung von Simulatorkopplungen zur Untersuchung hybrider Automatisierungssysteme,*. 4. GI/GMM/ITG-Workshop "Multi-Nature Systems", Ilmenau, 22. September 2003

[3] Benz, M.; Feske, K.; Hatnik, U.; Schwarz, P.: *TCP/IP Protocol Engine System Simulation. In Lecture Notes in Computer Science - Protocols for Multimedia Systems,* 6th International Conference, PROMS 2001, Enschede, The Netherlands, October 17-19, 2001,Volume 2213 / 2001, Springer-Verlag Heidelberg, ISSN: 0302-9743, 155-164

[4] Donath, U.; Schwarz, P.: *VHDL. at – Automatisierungstechnik*, Heft 9, 10, 11/2002

[5] Hatnik, U.;. Haufe, J.; Schwarz, P.: *Abstract Object Oriented System Simulation of Large Heterogeneous Communication Systems*. In "System Design Automation - Fundamentals, Principles, Methods, Examples", ISBN 0-7923-7313-8, Kluwer Academic Publishers, March 2001, pp.185 - 194 (Editors: R. Merker and W. Schwarz)

[6] Hatnik, U.; Sawitzki, S.: *Abstract Distributed Object-Oriented Simulation of Heterogeneous Communication Systems*, Proceedings of the IASTED (International Conference - Applied Simulation and Modelling), Crete, Greece, 25-28 June 2002, pp. 283-288

[7] IEEE Standard: *VHDL Language Reference Manual (1076-1993)*, New York, 1994

[8] The Mathworks, *Matlab/Simulink Users Guide, Application Program Interface Guide*, 2004

[9] Modeltech, *ModelSim Reference Manual*, 2004

[10] Schwarz, P.; Donath, U.: *Simulation - based Performance Analysis of Distributed Systems*, 5th Intern. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'97), Geneva, April 1997, pp 244-249

[11] Varadhan, K.; Fall, K.: *NS Notes and Documentation*, January 2000, http://www.isi.edu/nsnam/ns