SystemC-Based Cosimulation for Global Validation of MOEMS

L.Kriaa*, W. Youssef*, G. Nicolescu*, S. Martinez*,
S. Levitan**, J. Martinez**, T. Kurzweg**, A.A. Jerraya* , B. Courtois*

TIMA Laboratory
46, Av. Felix Viallet, 38031, Grenoble, France
{ kriia , yossef, nicolesc,martinez ,jerraya, courtois}@imag.fr

Department of Electrical Engineering, University of Pittsburgh,
348 Benedum Hall, Pittsburgh, Pennsylvania, 15261
{ steve , jmarti, tim }@ee.pit.edu

Key words : validation, cosimulation, MOEMS, SystemC

1. Introduction

Recent advances in micro-electro-mechanical systems (MOEMS) have made it possible to produce entire micro-optical systems in a single chip. These systems are becoming more and more popular; compared with macro-scale optomechanical devices, micro mechanical devices are smaller, lighter, faster (higher resonant frequencies), and more rugged [1].

These systems are heterogeneous, consisting in components belonging to different application domains, abstraction levels, or being described using different specification languages. Therefore, to cope with their complexity, the MOEMS design cycle starts with a high level specification and several refinement stages are necessary for the final implementation. In such a design flow the validation is challenging the majority of the efforts being devoted to it.

An efficient approach for the heterogeneous system validation is the simulation based approach that consists in the joined simulation of the different parts of the system by using simulators and abstraction levels appropriate to each component. This kind of simulation is called co-simulation.

Several cosimulation environments are presented today in the literature. A. Goth and al.. a distributed cosimulation environment where multiple simulators run in a lock-step manner synchronization [2]. Pia [3] is an environment where (geographically) distributed cosimulation can be performed. Polis [4] provides a uniform simulation environment (based on Ptolemy Discrete Event Domain [5]). Sung and Ha [6] presents a backplane concept that enables cosimulation of the systems with heterogeneous computation models. Semeria and Ghosh [7] present cosimulation methods based on SystemC [8]. As commercial tools Mentor Graphics Seamless CVE [9], Synopsys Eaglei [10] and Coware N2C [11] present Hw/Sw cosimulation environments.

This paper addresses the applicability of a SystemC-based cosimulation methodology to the global validation of MOEMS. This methodology starts with a system model given as a netlist of heterogeneous components and allows the automatic generation of simulation models. We applied this approach to the validation of an essential component in modern optical networks, an optical cross-connect (OXC).

2. Optical cross-connect – general overview

The application that we use in this work is a free-space optical cross-connect (OXC) using mirrors actuated by electrostatic devices. OXC have been demonstrated to be essential components in modern optical networks. These devices expand the traditional role of optical networks for transmission of information by providing signal routing directly in the optical domain.

Our choice for the free-space OXC is motivated by their low losses, and the use of electrostatic devices for mirrors actuation enable a relative compatibility of fabrication with microelectronics. To model the signal propagation we use a combination of ray and gaussian beam optics. These simple models allow fast simulation, and are appropriate for the early validation of MOEMS.

The global view of the free-space OXC is presented in Figure 1. The subsystems composing this system are: a control subsystem, four electro-mechanical actuators, two sources, two collimators, a 2x2 mirror array, two focusing lens and two photodetectors.

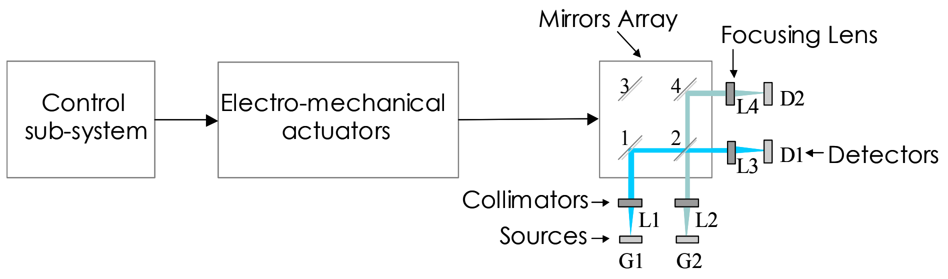


Figure 1 Optical cross-connect

We specified the control sub-system in SystemC and the electro-mechanical part in Matlab. For the behavior of optical devices (mirrors, lens, beam generators and detectors), we used C++ models from the libraries of Chatoyant [15].

3. Cosimulation environment

The cosimulation environment [12], [13], [14] starts with a system model given as a netlist of heterogeneous components and enables automatic generation of simulation models for heterogeneous systems.

3.1. Specification model

In this approach, we represent systems as a hierarchical network of modules. Each module consists of a *behavior* and *ports*. Modules are connected with each other by connecting their ports via *communication channels*. At the system level, channels hide details of protocol, for instance FIFO communication is realized using high-level communication primitives.

In the case of a heterogeneous specification, modules may be described in different languages, and channel and module may have different abstraction levels or different communication protocols. In order to enable connection we use *wrappers*. The wrapper is composed of an interface, made of two sets of ports (internal and external ports). It constitutes the interface of the module and isolates module's behavior from the rest of the system (Figure 2.a) .The *internal ports* are specific to the module and the *external ports* connect the module to external channels.

The wrapper concept enables the designer to easily specify the heterogeneous systems. For instance, in order to specify an optical module that is connected to the communication network, the interface of the optical module is specified as internal ports and that of the communication network as external ports.

3.2. Simulation model

The presented specification model is not executable because the internal structure of the wrappers is not yet fixed, only the internal and external ports are given. In order to simulate such a heterogeneous specification, interfaces adapting the different simulators or communication protocols/levels have to be generated.

These interfaces implement in fact the simulation models of the modules wrappers. In our approach, the simulation model of the wrappers consists in two types of interfaces: (1) simulator interfaces - that adapt different simulation environments to the cosimulation bus; (2) communication interfaces – that adapt different communication protocols/levels.

Figure 2 presents an example of heterogeneous specification and its corresponding simulation model.

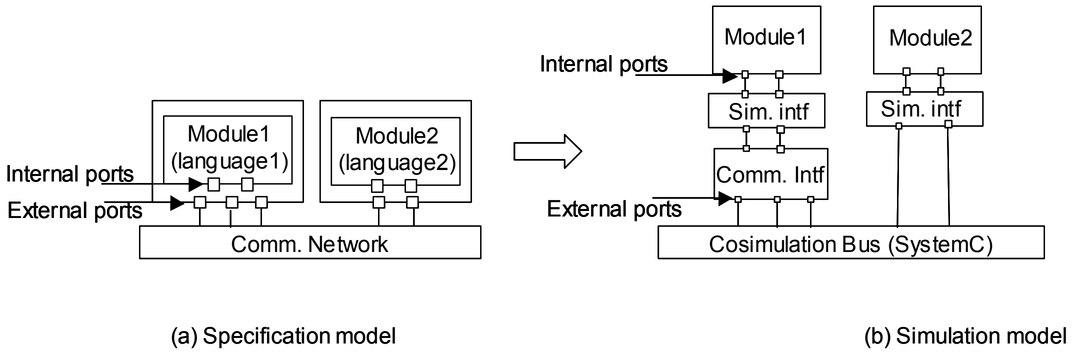


Figure 2. System specification and corresponding simulation model

A simulator interface is required when the module behavior is simulated by a different simulator than SystemC. It is specific to the used simulator.

A communication interface has a generic architecture composed of three basic elements:

- module adapter (MA) providing the internal ports (e.g. FIFO port) with required communication services (e.g. fifo_write, fifo_read, etc.). It performs also data conversion and channel resolution. The channel resolution is required since the number of internal ports can be different from that of corresponding external ports.

- internal communication media (ICM) to transfer data between module adapter and channel adapter. ICM can be an RPC (remote procedural call) relationship (i.e. the MA calls RPCs provided by channel adapters).

- channel adapter (CA) that enables the module to access the external channel. To do that, after receiving a channel access request (via MA) from the module behavior, it uses channel communication services (e.g. read_hs in the case of hand-shake communication network, etc.). To each external port, a channel adapter is assigned.

Figure 3 shows the structural details of the communication interface of module 1 in Figure 2.b.

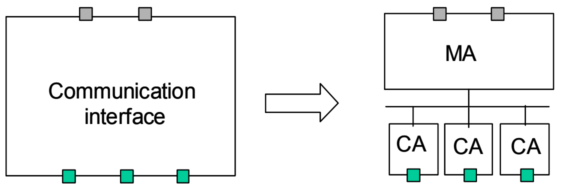


Figure 3. An example of internal structure of communication interface

3.3. Automatic generation of simulation models

Cosimulation interfaces are generated automatically using a library containing simulator interfaces and modules/channels adapters to compose communication interfaces. The flow is illustrated in Figure 4. This flow consists in two main stages:

1) Communication and simulation interfaces generation

This first stage implies the choice of simulation and/or communication interfaces from the cosimulation library:

- for each module described using an other specification language than SystemC, a correspondent simulation interface is instantiated. This consists in fact in a SystemC module encapsulating the simulator with whom communicates and is synchronized using the IPC (Inter Process Communication) mechanism. To generate the simulator interfaces we need necessary information from the system specification as follows.
    - reference to the behavior of the models
    - internal port specification (e.g. inputs/outputs and the correspondent communication protocols, data types, etc.).
- to adapt different communication protocol/abstraction levels, communication interfaces have to be generated. These interfaces are obtained by assembling module adaptors and channel adaptors selected from the cosimulation

library. Each module/channel adaptor choice is guided by the parameters of internal/external ports (e.g. communication protocol, direction, the transferred data type). In our approach, module and channel adaptors are described in SystemC.

2) Simulation models generation

The second stage consists in generation of the SystemC top-level routine *sc_main* that tight all the modules and their cosimulation interfaces together and provides the clock generation and the tracing capabilities. The *makefile* needed to compile the obtained design is also generated.

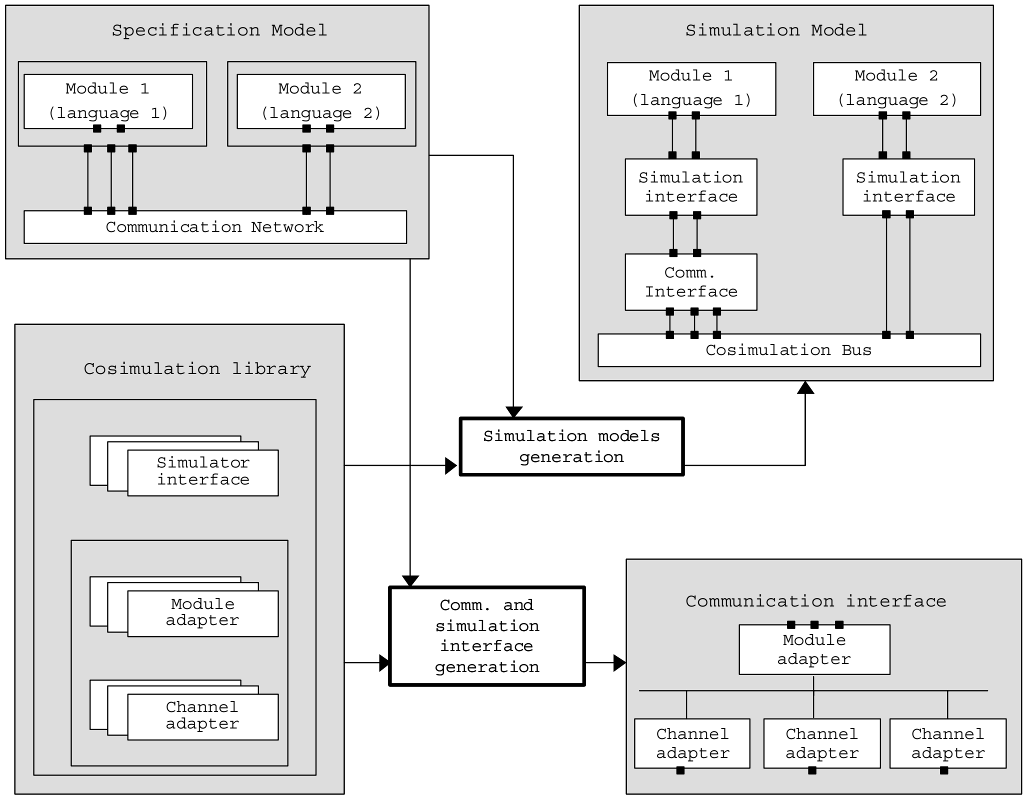


Figure 4. Flow for the generation of simulation models

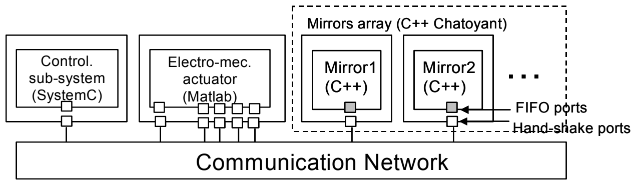
4. Experiments

To perform the overall OXC system cosimulation, the system specification model had been realized. This model is presented in figure 5.a. For the simplicity of the explanation, only two mirrors of the array are shown in the figure. Modules in the system communicate through communication channels that encapsulate simple handshake protocols. Each mirror module receives control data (i.e. the reflection command) from the electro-mechanical actuator with a FIFO communication protocol. To specify the interface of the optical modules and that of the communication channels, module wrappers have internal and external ports. As shown in figure 5.a, the internal ports specific to the module, are FIFO ports and the external ports connected to the communication channels are simple handshake ports.

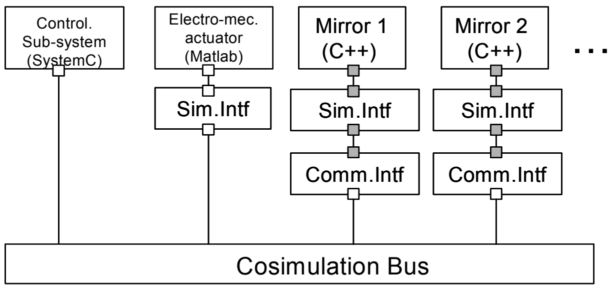
4.1. Automatic generation of the simulation model

To validate the system, the simulation model of the OXC is generated. The generated simulation model is illustrated in figure 5.b.

- Since Matlab-Simulink is used to model the electro-mechanical actuator sub-system, to adapt Matlab-Simulink simulator to the SystemC cosimulation back plane, the corresponding simulator interface is generated. The simulator interface corresponding to each C++ Chatoyant optical device model is very simple, consisting in standard SystemC interfaces that wrap C++ codes.
- The communication interfaces that adapt the communication specific to the mirrors model to the rest of the system are also generated. This is made by instantiating basic elements (module/channel adapters and internal communication bus) from the cosimulation library.


Mirrors array (C++ Chatoyant)
Control. sub-system (SystemC)
Electro-mec. actuator (Matlab)
Mirror1 (C++)
Mirror2 (C++)
. . .
FIFO ports
Hand-shake ports
Communication Network

(a) System specification model

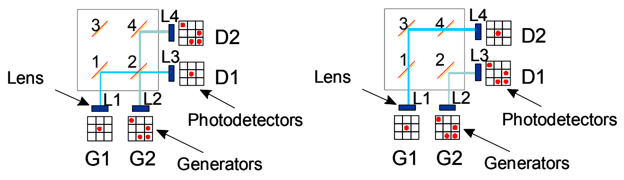


(b) Simulation model

4.2. Cosimulation results

For the experiment we used a 2x2 mirror array that have inputs from two beams generators and outputs to two photodetectors.

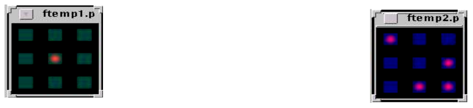
We run the cosimulation of the OXC. In the experiment, initially, the beam from G1 is detected by D1 and the beam from G2 is detected by D2. Figure 6.a shows the initial mirror configuration where two mirrors M1 and M4 are reflecting light from G1 to D1 by mirror M1 and from G2 to D2 by mirror M4. In our experiment we simulated the system evolution from this initial configuration to the final configuration (figure 6.b), where mirror M2 reflects the beam from G2 to D1 and M3 reflects the beam from G1 to D2.



(a) initial configuration (b) final configuration

Figure.5. Initial and final configuration of the optical MEM switch

Each of beam generators and each of detectors can generate/detect up to nine beams. For a better examination of the results, we parameterized G1 and G2 differently: G1 generates one of nine beams, and G2 generates four of nine beams. Figure 7 shows the generated beams by G1 and G2. The nine rectangles represent nine possible beam positions and dots represent generated beams.



a) G 1 b) G2

Figure. 7. Outputs for the beam generators

The control sub-system sends commands to the mirrors by changing the electronic voltage assigned to each mirror. The electro-mechanic actuator converts the electronic commands to the mechanic commands, i.e. the distance of mirror movement. Table 1 shows the voltage levels of electronic orders of the control sub-system and the corresponding mechanical orders converted in terms of distance for the mirror movement (see figure 5.a). As shown in the table, to change the mirror configuration from total reflecting to non-reflecting, or vice versa, the mirror needs to be moved 400 μm by the commands of the control sub-system. To do that, the control sub-system gives eleven steps of command as shown in Table 1.

The evolution of data path steering, i.e. beam reflection by the mirrors, during the simulation is illustrated in figure 9. Each line of images corresponds to each of two detectors composing the optical sub-system.

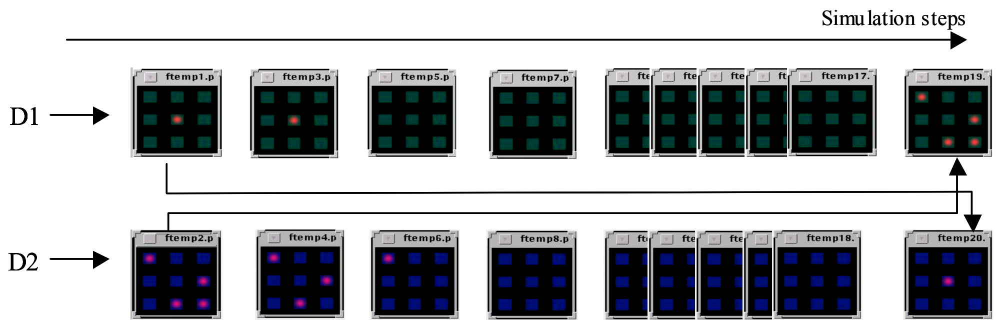


Figure 8. Cosimulation results for the optical MEM switch

Initially, mirrors M1 and M4 steer the data path, by reflecting totally the beam received from G1 to D1 and the four beams from G2 to D2, respectively. Note that, in that case, mirrors M2 and M3 are not reflecting any beam.

We can remark that at the first simulation step, D1 and D2 detect the outputs of G1 and G2, respectively, which are totally reflected by mirrors M1 and M4. During the simulation, mirrors change gradually their position according to the commands sent by the control part (in the Matlab simulator). For instance, at the second step of control commands, M4 changed its position and reflected partially - three of its four input beams. Consequently, D2 detected parts of the light generated by G2 (three beams). At the end of simulating eleven steps of command, mirrors M3 and M2 steer the data path, reflecting totally their inputs from G1 and G2, respectively. In this case, mirrors M1 and M4 are not reflecting any beam.

The overall system simulation was performed on a SAN Ultra-Sparc 1, in about 30 seconds. This enabled a fast validation of the overall system functionality, before its implementation in a final architecture.

Table 1. Mechanical/electrical orders for data path steering

Voltage (V)	Distance (μm)
0.0	0
12.7795	20
17.3641	40
20.3459	60
22.4453	80
23.9098	100

24.8782	120
25.4335	140
25.6295	160
25.6309	162
>25.6309	400

For the experiments we performed the overall validation of the free-space OXC using the presented cosimulation framework. To perform cosimulation, the system specification model had been realized. Matlab-Simulink was used to model the electro-mechanical actuator sub-system, for the specification of the optical devices we used Chatoyant and SystemC for the control sub-system model. The corresponded simulation model was generated automatically:

This enabled a fast validation of the overall system functionality, before its implementation in a final architecture.

These experiments show the power of cosimulation techniques to accommodate heterogeneous systems.

We had no specific difficulty for building the overall simulation. Starting from the cosimulation environment we had only to write the simulation interfaces for Chatoyant. The main difficulty in this project was to bring together the three different teams working on the three sub-systems in order to obtain the global specification. Optical and micro electro-mechanical designers were not used to this kind of this high level of abstraction.

We believe that cosimulation can be used for even more sophisticated applications like full car or aircraft systems.

References

[1] Wu, M.C., "Micromachining for Optical and Optoelelctronic Systems", Proc. of the IEEE, Vol. 85 No. 11, Nov. 1997

[2] A. Ghosh et al. "A Hardware-Software Co-simulatior for Embedded Systems Design and Debugging ", proc. Asia South Pacific Design Automation Conference, 1995.

[3] K. Hinnes and G. Boriello, "Dynamic Communication Models in Embedded Systems Co-simulation", Proc. Design Automation Conference, pp. 395-400, June 1997.

[4] F. Balarin et al. "*Hardware-Software Co-design for Embedded Systems*", Kluwer Academic Publishers, 1997.

[5] S. Lee and J.M. Rabaey, "A hardware software cosimulation environment", International Workshop on Hardware- Software Codesign", Cambridge, Oct. 1993.

[6] W. Sung and S. Ha, "Efficient and Flexible Cosimulation Environment for DSP Applications", *IEICE Trans on Fundamentals of Electronics, Communications and Computer Sciences*, volE81-A, no. 12, pp. 2605-2611, Dec. 1998.

[7] L. Sémeria and A. Ghosh, "Methodology for Hardware/Software Coverification in C/C++", proc. Asia South Pacific Design Automation Conference, Jan. 2000.

[8] SystemC Consortium, "SystemC Version 2.O" available at http://wwww.systemc.org

[9] Mentor Graphics, Inc., "Seamless CVE", available at http://www.mentorg.com/seamless

[10] Synopsys, Inc., Eaglei, available at http://www.synopsys.com/products/hwsw/eagle_ds.html

[11] Coware, Inc., "N2C", available at http://www.coware.com/cowareN2C.html

[12] P. Gerin, S. Yoo, G. Nicolescu, A.A. JERRAYA, "Scalable and Flexible Cosimulation of SoC Design with heterogeneous Multi-processor Target Architecture", proc. Asia South Pacific Design Automation Conference, January, 2001.

[13] S.Yoo, G. Nicolescu, D. Lyonnard, A. Baghdadi, A.A. Jerraya, "A Generic Wrapper Architecture for Multi-Processor SoC Cosimulation and Design ", Proc. of CODES 2001, Denmark, 2001.

[14] G. Nicolescu, S. Yoo, A.A. Jerraya, "Mixed-Level Cosimulation for Fine Gradual Refinement of Communication in SoC Design", Proc. of Design Automation and Test in Europe, mars, 2001.

[15] T. Kurzweg, J. Martinez, S. Levitan, P. Marchand, D. Chairulli, "Dynamic Simulation of Opitcal MEM Switches", DTIP France, April, 2001.

Lobna Kriaa is pursuing her master at Joseph Fourier University, Grenoble. Her master stage will be effectuated in the SLS group from TIMA Laboratory, Grenoble. The subject is the global validation of heterogeneous systems. Kriaa has an engineering degree from the University of Sciences, Tunisia.