Application of Multi-domain and Multi-language Cosimulation to an Optical MEM Switch Design

G. Nicolescu, S. Martinez, L. Kriaa, W. Youssef, S. Yoo, B. Charlot, A. Jerraya
TIMA Laboratory
46, Av. Felix Viallet, 38031, Grenoble, France
{nicolesc,martinez,kriia,yossef,charlot,jerraya}@imag.fr

Abstract

This paper addresses the applicability of a cosimulation methodology to multi-domain and multi-language systems design. This methodology starts with a system model given as a netlist of heterogeneous components and enables the systematic generation of simulation models for multi-domain and multi-language heterogeneous systems. For experiments, we used a complex multi-domain application: an optical MEM switch.

1. Introduction

Today’s embedded systems are getting more and more heterogeneous including multi-domain (electronic, mechanical, and optical) components. Systems such as the optical MEMS (micro-electro-mechanical systems) are moving from abstract ideas to marketable products e.g. switching, scanning, projection, printing, etc. Examples of optical MEMS are the optical-switches that are becoming increasingly popular due to the many advantages that they propose over typical optical fiber optic switches (e.g. small, fast, reliable, and eventually will be inexpensive to produce) [1].

Designing such systems is challenging in two aspects: heterogeneous design languages and high complexity of design to integrate heterogeneous components. Due to their heterogeneity these systems are described using different domain-specific languages (e.g. C language and HDL for software and hardware design, MATLAB for electro-mechanical sub-systems, etc.). To cope with the design complexity of refining heterogeneous components and their integration, the design process requires a top-down flow from a high-level specification down to a low-level implementation. In this case, the validation of system design at any stage of the design flow is challenging. The key issue is the coordination between the different sub-systems that manipulate different communication concepts and are described using different design languages. The execution of heterogeneous models is called cosimulation.


The main contribution of this paper is to present a case study applying a multi-domain and multi-language cosimulation methodology for the validation of a heterogeneous system, the optical MEM switch. This cosimulation methodology [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.

This paper is organized as follows. The next section introduces the application, an Optical MEM Switch. Section 3 presents the conceptual framework of the cosimulation methodology that we used. Section 4 gives details for the cosimulation of the Optical MEM switch and presents the results that we obtained. These results are commented in section 5 and finally we conclude this paper in section 6.

2. Optical MEM Switch

The application that we choose for experiments is an optical MEM Switch. The global view of the system is presented in figure 1. It is composed of three sub-systems:

- The control sub-system that will be finally mapped on a processor (control sub-system in figure 1). It calculates electronic orders (voltage) commanding motion of the mechanical mirrors in the optical sub-system.
**The electro-mechanical actuator that transforms the voltage from the output of the control sub-system into mechanical orders, in terms of positions, for each of the mirrors.**

**The optical sub-system that is composed of a 2x2 mirror array, two light generators (G1, G2 in figure 1) and two beams detectors (D1, D2 in figure 1). Four lenses (L1-L4) are used to collimate or focus the light.**

The switching operation of MEM optical switch is achieved through the mechanical motion of mirrors steering the data path from the inputs (G1 and G2 in figure 1) to the desired outputs (D1 and D2 in figure 1). This mechanical motion is controlled by the control sub-system.

To perform switching operation, the control sub-system commands the position of the mirrors. As shown in figure 2.a, depending on their position, mirrors may reflect totally (d=dmax), partially or do not reflect their inputs (d = 0). Figure 2.b shows four mirrors that have two inputs from two beam generators (G1 and G2) and two outputs to two light detectors (D1 and D2). In the figure, the control sub-system controls the reflection of each of mirrors in the manner shown in figure 2.a.

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. The cosimulation framework

This section presents the conceptual framework of a cosimulation tool based on SystemC. More specifically, we present the basic concepts used for the specification and simulation of heterogeneous systems. More details on this environment can be found in [12], [13], [14].

#### 3.1. Specification models for heterogeneous systems

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.

![Figure 1. Optical MEM switch](image)

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, in this case 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 (Fig. 3.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, to specify an optical module that is connected to the communication network, the interface of the optical module is specified as the internal port specification and that of the communication network as the external port specification.

Actually for the system specification we use an extension of SystemC. Three new concepts are used:

- the virtual module, consisting of the module and its wrapper
- the virtual port, grouping the corresponding internal and external ports having a conversion relationship. Thus, a wrapper may be composed of several virtual ports.
- the virtual channel grouping several channels having a logical relationship (e.g. channels belonging to the same communication protocol).

#### 3.2. Simulation of heterogeneous systems

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. 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 3 presents an example of heterogeneous specification and its 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.

To illustrate how these elements compose a communication interface, figure 4 shows the structural details of the communication interface of module 1 in figure 3.b.

As shown in the figure, this communication interface example has a module adapter that has two internal port connections and three external port connections. The expansion of this communication interface results in a module adapter managing the two internal ports and three channel adapters managing the external ports. In the figure, the internal communication media is shown as a set of lines each of which can be an RPC relationship.

3.3. Automatic generation of cosimulation interfaces for heterogeneous specification

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 5. 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 adapter 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.
4.1 Automatic generation of the simulation

4.1. Automatic generation of the simulation model

4.1.1 Automatic generation of the simulation model

4.1.1.1 Automatic generation of the simulation model

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.

Details for the generation of the simulation model in the case of optical switch design will be given in section 4.

4. Cosimulation of the Optical MEM Switch

To perform the overall system cosimulation, the system specification model had been realized. This model is presented in figure 6.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 6.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 optical MEM switch is generated. The generated simulation model is illustrated in figure 6.a.

Figure 5. Flow for the generation of simulation models

2) Simulation models generation

• 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.

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 (see figure 5.b).

We run the cosimulation of the optical MEM system. In the experiment, initially, the beam from G1 is detected by D1 and the beam from G2 is detected by D2. Figure 7.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 7.b), where mirror M2 reflects the beam from G2 to D1 and M3 reflects the beam from G1 to D2.
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 8 shows the generated beams by G1 and G2. The nine rectangles represent nine possible beam positions and dots represent generated beams.

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.

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>
<thead>
<tr>
<th>Voltage (V)</th>
<th>Distance (µm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>12.7795</td>
<td>20</td>
</tr>
<tr>
<td>17.3641</td>
<td>40</td>
</tr>
<tr>
<td>20.3459</td>
<td>60</td>
</tr>
<tr>
<td>22.4453</td>
<td>80</td>
</tr>
<tr>
<td>23.9098</td>
<td>100</td>
</tr>
<tr>
<td>24.8782</td>
<td>120</td>
</tr>
<tr>
<td>25.4335</td>
<td>140</td>
</tr>
<tr>
<td>25.6295</td>
<td>160</td>
</tr>
<tr>
<td>25.6309</td>
<td>162</td>
</tr>
<tr>
<td>&gt;25.6309</td>
<td>400</td>
</tr>
</tbody>
</table>

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.
5. Analysis of the results

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.

6. Conclusion and future work

This paper presented a case study applying a multi-domain and multi-language cosimulation methodology for the validation of optical MEMS, using an optical MEM switch application. The used cosimulation methodology is based on automatic generation of cosimulation interfaces starting form a heterogeneous specification where the different modules may use different communication concepts or may be described in different languages.

Future work on this case study will focus on interfacing RTL level simulation of the control block (using a processor Instructor Set Simulator) with the rest of the components. Also, we plan to perform simulation for an improved version of the system, where we optimize the control of the mechanical mirrors motion, by closing the loop between the control part and the optical part of the system.

Acknowledgments

We would like to acknowledge Steven Levitan, Timothy Kurzweg and José Martinez from the University of Pittsburgh for providing us the simulation models of the optical devices and for the enriching discussions concerning MEMS modeling and simulation.

References