An Application of Parallel Discrete Event Simulation Algorithms to Mixed Domain System Simulation

D. K. Reed[1] S. P. Levitan[1] J. Boles[1] J. A. Martinez[1] D. M. Chiarulli[2]
Univ. of Pittsburgh, Departments of Electrical Engineering[1] & Computer Science[2]

Abstract

We present our system-level co-simulation environment for mixed domain microsystems. The environment provides synchronization and co-simulation between the Chatoyant MOEMS (Micro-Electro Mechanical Systems) simulator and ModelTech ModelSim. By using shared memory IPC (Inter-Process Communication) and PDES (Parallel Discrete Event Simulation) techniques, we achieve two orders of magnitude speedup over standard pipe/socket communication.

I. Introduction

The advent of highly integrated technologies such as MEMS (Micro-Electro-Mechanical Systems), MOEMS, s, mixed signal systems on a chip (SoC), and opto-electronic communication networks, is requiring more sophisticated and efficient tools for simulating these technologies. It is only through system level simulation that efficient design space exploration can be performed across both architectural and implementation choices.

There are a multitude of simulation packages for specific technology domains at various levels in the design abstraction hierarchy including system level simulation. However, in general, these simulators are not designed to work with others, across domains. To address this issue, these simulators must be integrated or, at the very least, interfaced together.

II. Application

The Chatoyant mixed-domain simulation environment from the University of Pittsburgh provides a broad range of models that span analog electronics, optics and MEMS devices [1]. Chatoyant is built on top of Ptolemy developed at University of California, Berkeley, and uses both Discrete Event (DE) and Dynamic Data Flow (DDF) simulation domains [2]. However, neither Chatoyant nor Ptolemy provide library components or features that can directly simulate digital design circuits written in VHDL. To handle these components, we have chosen to use the commercially available ModelSim HDL simulator for the components of a particular system that require an HDL simulator [3].

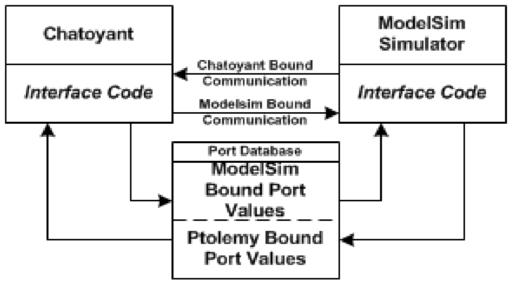
Chatoyant
Interface Code
ModelSim Simulator
Interface Code
Chatoyant Bound Communication
Modelsim Bound Communication
Port Database
ModelSim Bound Port Values
Ptolemy Bound Port Values

Figure 1. Environment for Co-Simulation between Chatoyant and ModelSim

Figure 1 shows the architecture of the co-simulation environment. The implementation of this environment must address two issues: 1. How is information exchanged accurately between a digital domain simulator with MVL9 (Multi-Value Logic 9-Level standard) conventions and an analog simulator? 2. How is synchronization and data exchange performed between these two simulators?

The first issue can be addressed by the means of technology dependent lookup tables, converting between symbolic digital values and analog voltages.

Since both simulators are discrete event simulators, the second issue can be solved with parallel discrete event simulation (PDES) techniques [4]. There are two fundamental synchronization approaches in PDES classed either as conservative or optimistic. Conservative approaches require all simulators to remain synchronized, never simulating to a future time until all simulators are ready to proceed. Optimistic approaches, on the other hand, allow any particular simulator to simulate beyond a certain point, without requiring other simulators to be synchronized. However, if a past event comes to that simulator, relative to its local time, then that simulator must “rollback” to a previous safe state.

III. Interface Design Considerations

For the environment developed here, an evaluation was performed to choose which PDES approach would be the most efficient. Both approaches are possible since ModelSim does have check-pointing and restoring methods available [3]. However, after considering that the entire co-simulation environment will be executing on one machine, with other memory and computationally intensive tasks executing, we determined that this option would be too costly in terms of memory overhead. For the system described in Section V, the amount of data required for a checkpoint file was on the order of 1 to 2MB. With typically 10 checkpoint files needed, rollback time took between 500ms to 1.5s.

The conservative approach gives a solution requiring less memory. By ensuring that both simulators (one being Chatoyant and the other being ModelSim) be consistently synchronized this becomes a matter of passing event information between the simulators. Thus, the only real design issue becomes the time synchronization method.

IV. Synchronization Implementation

In order to avoid causality errors between the both simulators, the two must use common synchronization frequency. The frequency at which the simulators synchronize determines the accuracy of when, in simulation time, the events occur. For example, if a system being simulated uses 1GHz clocks then the frequency of synchronization should be on the order of picoseconds, to keep the times at which events occur in each simulator accurately synchronized.

There are two approaches using Inter-Process Communication. The first approach implements a null-message passing scheme in PDES via named pipes. The drawback with this approach is that there is overhead communicating both port and synchronization information through the pipes.

The second approach solves this problem. Instead of using named pipes as the primary conduit for synchronization, a shared memory system is used to maintain information about each port's state (Figure 1). Synchronization is checked every synch-period, but information is only updated and scheduled when necessary using a dirty-bit mechanism. This also reduces the port examination and scheduling overhead from the first approach.

V. Experimental Results

To test these approaches, a system simulation of an optoelectronic crossbar switch was used [5]. This system is composed of guided wave optics, and both optoelectronic and analog circuits (simulated in Chatoyant), and digital hardware (simulated in ModelSim). The information flow in the system consists of three stages of switches. At each stage, optical data streams are detected by photo-diodes, amplified in analog circuits, routed by digital circuitry, and sent through analog amplifiers to a vertical cavity surface emitting laser (VCSEL) array. Light is passed between stages by guided wave optics. Figure 2 shows the block layout of the system.

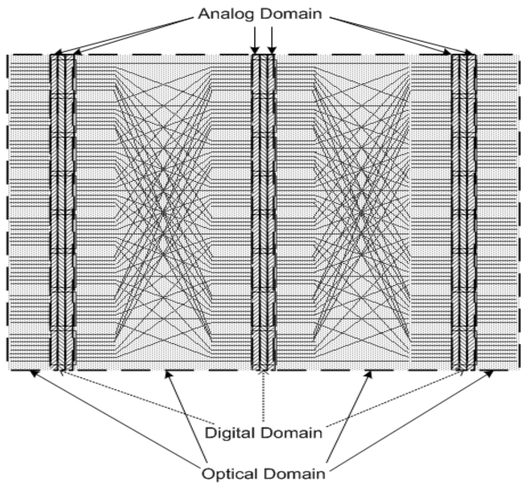

Figure 2. FIG Test System Block Diagram. Areas demarked as Digital Domain execute in ModelSim, areas demarked as Analog and Optical are executed in using Chatoyant components.

The system was simulated for 1.3us of a 1GHz clock, on a 2.4GHz 1GB P4. The first approach had a runtime of about 481.25 minutes with 1.3 million events. This was at 1ps resolution. The second approach with the same 1.3us simulation time had a runtime of 4.21 minutes with 1.3 million events. This equates to a speedup factor of 114.

VI. Conclusion

The experiment above only gives a cursory illustration of how the interface performs. Like most PDES based systems, runtimes are dependent on the system itself and the amount of event traffic at given points. However, the preliminary information presented here shows a promise of acceptable (interactive) runtimes and a useful interface for system level simulation.

References

[1] S.P. Levitan, et.al, , IEEE Trans. On CAD of ICs and Systems, vol. 22, no. 2, February 2003.
[2] J. Buck, et.al, Int. J. Comput. Simul., vol.4 pp 155-182, 1994.
[3] ModelSim SE FLI, Version 5.7g. 2003
[4] Banerjee, Prithbiraj. Parallel Algorithms for VLSI Computer-Aided Design. Prentice Hall. 1994.
[5] Donald M. Chiarulli, et.al., Optics in Computing, OSA Technical Digest (Optical Society of America, Washington DC, 2001), pp. 125-127.