# ECE 521 Final Project Report

## Improving SPICE Simulation Time with Verilog Co-Simulation

Joseph Crop

December 6, 2010

http://people.oregonstate.edu/~cropj/Classes/ECE521

# Executive Summary

## 0.1 Project Description

SPICE simulations can be very time consuming when thousands of transistors are simulated. Often, within mixed-signal simulations, these transistors are part of digital circuits that don't require accurate modeling. For example, a designer my be using a digital circuit to calibrate an ADC at the beginning of a simulation, or count the number of oscillations of a VCO. There are also even more complex examples such as the functional testing of an entire SoC [1]. If there exists a way to model these digital circuits' functionally instead of with complex transistor models, a mixed-signal simulation can be sped up dramatically.

The primary objective of this project is to integrate a Verilog simulator into an existing mySpice simulator modified for co-simulation. A Verilog/SPICE co-simulator and state of the art hspice simulator were used to measure transient simulation times. A comparison has be made between co-simulation and transistor-only simulation.

## 0.2 Approach

This project consists of two primary objectives: design and comparison. The approach is as follows:

1. First, a c-code interface from mySpice into the Verilog simulator "ModelSim" was designed. This was done by writing code that defines interface routines with the Verilog simulator. These input/output routines are used to change and read variables as well as change the time-step of the Verilog simulation. The simulator is designed such that at each time-step, the node voltages of SPICE are evaluated as either a logic 1 or 0. These nodes are input into the Verilog simulator and result in new logic values that are converted into voltages and input into the SPICE simulator as voltage sources for the next transient time-step.

2. After modifying the mySpice simulator successfully, test circuits were simulated with and without different forms of co-simulation. The example test circuits used consisted of a ring-oscillator and digital counter or digital clock divider. The digital counter was synthesized at different bit-depths: 16, 32, and 64. The clock divider circuit was used to show that digital Verilog outputs are able to control analog SPICE circuits.

## 0.3 Accomplishments

The accomplishments of this work are two-fold: First is the completion of a custom designed Verilog co-simulator. This will allow for SPICE driven transient co-simulations to be performed in the future. Secondly, this work has shown that co-simulation is an excellent alternative to full blown mixed-signal spice simulations. By combining the required precision of analog simulations with the speed of digital simulations, simulation time can be improved by over 50% in many cases depending on the complexity of the digital system.

# Improving SPICE Simulation Time with Verilog Co-Simulation

Joseph Crop

***Abstract*—As SPICE simulations today use more and more transistors, especially in SoC designs, their simulation speed reduced dramatically. This work presents the motivation for co-simulation, a method by which SPICE can be interfaced with more abstract (usually digital) models to improve simulation speed. To show this, a co-simulator know as joeSpice is designed and implemented in c using a mySpice transient simulator and modelSim Verilog simulator.**

***Index Terms*—SPICE, Verilog, co-simulation**

## I. Introduction

SPICE simulations can be very time consuming when thousands of transistors are simulated. Often, within mixed-signal simulations, these transistors are part of digital circuits that don't require accurate modeling. For example, a designer my be using a digital circuit to calibrate an ADC at the beginning of a simulation, or count the number of oscillations of a VCO. There are also even more complex examples such as the functional testing of an entire SoC [1]. If there exists a way to model these digital circuits' functionally instead of with complex transistor models, a mixed-signal simulation can be sped up dramatically.

The following will compare the different forms of co-simulation available today, go over the implementation details of the joeSpice co-simulator, and ref ect on results obtained by using the co-simulator.

## II. Importance of Co-Simulation

Co-simulation is the process by which 2 or more simulation engines are combined to achieve either faster simulation time or simply a more robust simulation. Much research has been done on Co-Simulation engines [2]–[5]. The following will explore 2 specif c co-simulation techniques: Verilog co-simulation, and the XSPICE simulator [6].

### A. Verilog Co-Simulation

Verilog Co-Simulation is probably the most popular type of mixed-signal co-simulation used today. One of the most popular products that uses it today is Synopsys' HSIM-Plus [7]. Verilog Co-simulation integrates a SPICE simulation environment with a Verilog simulator.

*1) Pros:* Verilog co-simulation is great for SoC simulation because dgital designs will already be written in an HDL language. Therefore there is very little extra work to be done to start a simulation.

*2) Cons:* Inter-process communication can be slow when communicating with a Verilog simulator.

### B. The XSPICE simulator

The XSPICE simulator was written 1992, almost 20 years ago. As it is outdated now, one might f nd Verilog-A to be a much more advanced replacement for it. XSPICE has designed within it a series of pre-coded modules for co-simulation. Examples include digital gates, op-amps, f ip-f ops, and even RAM. The user def nes instances of these cells within the spice simulation f le much like one would for a voltage source or resistor.

*1) Pros:* One clear advantage is that these integrated, pre-compiled modules improve simulation speed because there is no need to pre-interpret any modules or inter process communication lag.

*2) Cons:* This proprietary format is not very scalable. It is only really applicable to test-benches and temporary replacements of ideal analog components.

## III. The joeSpice Co-Simulator

In order to explore Verilog co-simulation in a more controlled environment, a Verilog Co-simulator was designed. The simulator incorporates the mySpice transient simulator and interfaces with Mentor Graphics' *modelSim* Verilog simulator.

### A. Co-Simulatior Design Description

A top-level diagram of the co-simulator is shown in Figure 1. The simulator is designed to communicate using unix pipes/buffers to and from a modelSim instance. Once a Verilog file has been read into joeSpice, it is compiled into a modelSim instance. The modelSim simulator is then started and waits from commands via its stdin. Once the simulation is started, commands to set Verilog nets (set), read Verilog nets (get), and synchronize timesteps are sent from mySpice to modelSim through the bi-directional pipe.

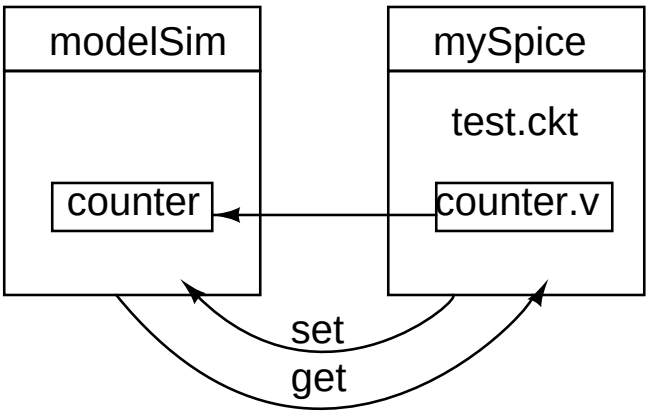


Fig. 1. Diagram of joeSpice Co-Simulator flow

### B. Simulator Control

In order to make Verilog integration fluid to the user, several new SPICE commands were added to joeSpice on top of the already existing .tran and .ic commands.

*1) .vfile file.v module_name:* The vfile card is used to readin a top-level Verilog file. Once detected, the simulator goes into co-simulation mode instead of standard transient simulation mode.

*2) .vin bus_size net_name:* The vin card defines SPICE nets that are used to control Verilog inputs. Multiple-wire buses need to be handled differently in Verilog. The *bus_size* parameter is used to generate logical buses in joeSpice so they can be entered into modelSim correctly.

*3) .vout bus_size net_name:* The vout card is similar to the vin card except it defines Verilog output nets or busses that are to be printed in the transient output only.

*4) .vdriver bus_size net_name:* The vdriver card was designed in order to make a distinction between Verilog outputs and Verilog outputs that control SPICE nets. For every driver, a voltage source is added into the circuit matrix. Due to the high complexity of multi-bus drivers, only single-net drivers are supported.

*5) .vlevel low high:* The vlevel card is used to tell the simulator what voltages to interpret as logic high and logic low. These are used as vdriver output levels and their midpoint is used as the switching threshold for SPICE → Verilog conversions.

### C. Test 1: SPICE → Verilog Communication

In order to test the software link from mySpice to modelSim a simple test circuit depicted in Figure 2 was used. The circuit consists of a simple ring oscillator (analog) driving the clock pin of a counter (digital). The counter was designed in Verilog and only exists in modelSim, whereas the oscillator exists only in SPICE.

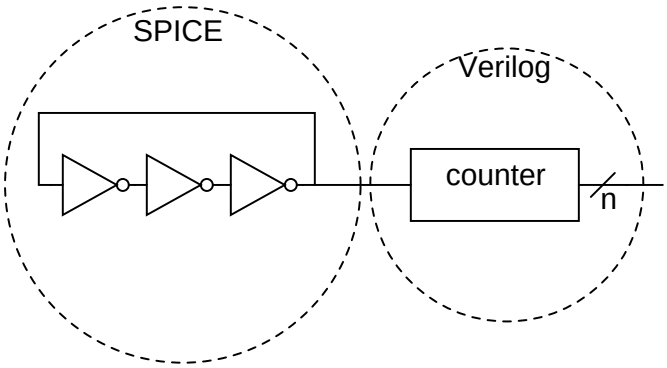


Fig. 2. Top-level Schematic of Test Circuit 1

As the simulation is run, the output of the oscillator is translated to a digital 1 or 0 and sent to modelSim for evaluation. In this case, the outputs of the Verilog portion of the system do not drive any analog nets, therefore they are simply displayed in the final transient output. A plot of the circuit in Figure 2 is illustrated in Figure 3 as taken from the joeSpice simulator output.

### D. Test 2: Verilog → SPICE Communication

The interface from Verilog to SPICE is much more complicated than the interface described for Test Circuit 1. In order to have Verilog outputs

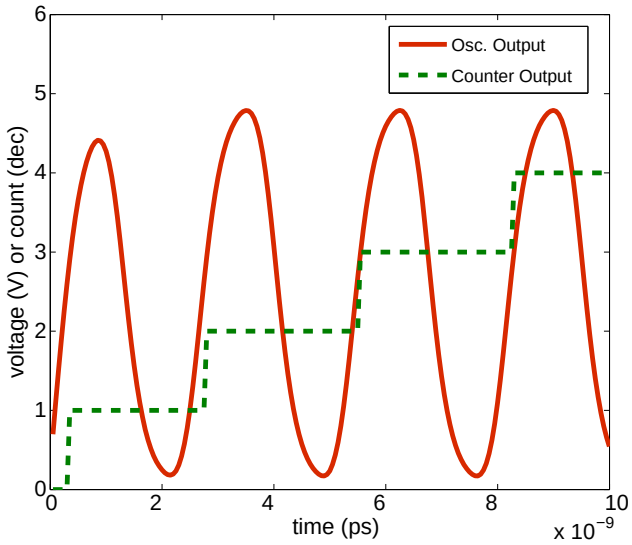


Fig. 3. Plot of Working Co-Simulator Output for Test Circuit 1

control an analog circuit, voltage sources must be added that are controlled by the Verilog output. There are many ways this can get messy. Usually is it is desirable to have a finite rise and fall time as well as an accurate driver. For this project, finite rise time was implemented in the form of a linear stepping voltage source. In order to test this fundamental interface the circuit in Figure 4 was used. This circuit consists of another ring oscillator connected to a clock divider. The clock divider output is then connected back into the SPICE input terminal of an inverter. If the output of the inverter is switching when driven by the divider we know this type of interface is working correctly.

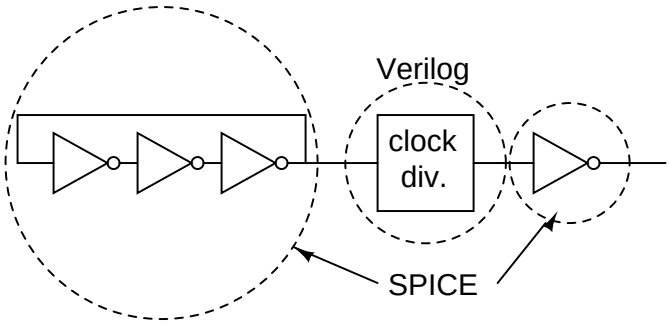


Fig. 4. Top-level Schematic of Test Circuit 2

As can be seen by the plot in Figure 5, the oscillator output is translated to Verilog and divided down. The resultant divided signal is translated back into SPICE and drives the inverter correctly.

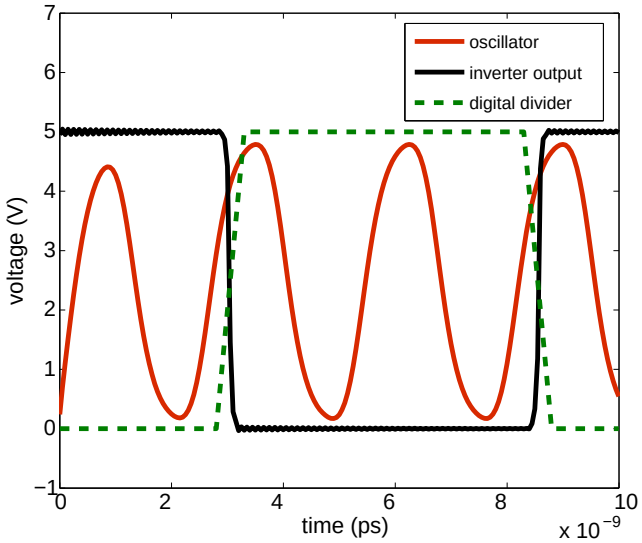


Fig. 5. Plot of Working Co-Simulator Output for Test Circuit 2

## IV. Results

Using the newly designed Verilog co-simulator, performance tests were carried out. Test circuit 1 (Figure 2) was synthesized using a 90nm IBM process for simulation in hSpice. In tandem, an identical Verilog model was used in the co-simulator. Results comparing the XSPICE simulator are unavailable because it would not compile on newer systems. The original manual makes references to a 32-bit Motorola 68000 processor with a minimum RAM requirement of 4 Mega-bytes. Four tests were run on each simulator. One with just the oscillator as a baseline, and three with varying counter sizes: 16, 32, and 64 bits. The time it took to complete the simulation for each case is recorded in Table I along with the number of mosfets used in the hSpice simulation.

The netlists and simulators were quite different in this test. The hSpice tests had complex BSIM 4 models which lead to a significantly longer simulation time for the standalone oscillator. However, The most important metric for this test is not to directly compare simulation time for each simulator. It is far more beneficial to measure the amount of time it takes between simulations of different sized digital blocks for each simulator. It is clear that with the

| Total Simulation Time | | | | |
|---|---|---|---|---|
| | No Digital | 16-bit | 32-bit | 64-bit |
| joeSpice | 0.017s | 21.26s | 19.82s | 19.21s |
| hSpice (90nm) | 3.71s | 42.29s | 83.50s | 144.39s |
| **Number of Mosfets** | | | | |
| | No Digital | 16-bit | 32-bit | 64-bit |
| joeSpice | 6 | 6 | 6 | 6 |
| hSpice (90nm) | 6 | 742 | 1494 | 2998 |

TABLE I
COMPARISON OF CO-SIMULATION VS. DIRECT SPICE

co-simulator, once the counter is added, the amount of speed degradation is virtually 0 as the size of the counter increases. For hSpice on the other hand, as more bits are added to the counter, more mosfets are added to the simulation. This results in an average speed loss of 85% as even though most mosfets are not switching, their model evaluation takes up much of the simulation time. This trend will only widen as more complex digital systems are integrated.

### A. Summary

The one clear drawback to the joeSpice co-simulator design is the use of file IO to communicate between simulators. This leads to the dramatic overhead observed by comparing the oscillator simulation to any Verilog simulation.

By looking at the results from joeSpice it can be seen that nearly the entire simulation time is taken up by the Verilog portion of the simulation, about 20 seconds. This can be easily paired with the hSpice data to result in a total simulation time of 24 seconds for any sized counter, as the oscillator only takes about 4 seconds to be simulated.

Comparing the paired results to the normal hSpice results in Table I, a speedup of almost 50% can be seen for the 16-bit counter alone.

## V. CONCLUSION

By designing and implementing a co-simulator, the complexity and requirements of co-simulation were analyzed. It has been shown that Verilog co-simulation can clearly reduce the amount of time it takes to complete a mixed-signal simulation. This leads to a clear motivation to use co-simulators such as Synopsys' HSIM-Plus. Based on the findings of this work, co-simulation engines can reduce simulation time dramatically, assuming most of the mosfets under test can be translated to Verilog models.

## REFERENCES


[1] G. G. E. Gielen, S. Member, and R. O. B. A. Rutenbar, "Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits," *Computer Engineering*, vol. 88, no. 12, pp. 1825–1852, 2000.

[2] C. Kreiner, C. Steger, and R. Weiss, "A hardware/software cosimulation environment for dsp applications," in *EUROMICRO Conference, 1999. Proceedings. 25th*, vol. 1, 1999, pp. 492 –495 vol.1.

[3] P. Gerin, S. Yoo, G. Nicolescu, and A. Jerraya, "Scalable and flexible cosimulation of soc designs with heterogeneous multi-processor target architectures," in *Design Automation Conference, 2001. Proceedings of the ASP-DAC 2001. Asia and South Pacific*, 2001, pp. 63 –68.

[4] W. Fornaciari, D. Sciuto, and F. Salice, "A two-level cosimulation environment," *Computer*, vol. 30, no. 6, pp. 109 –111, Jun. 1997.

[5] S. Yoo and A. Jerraya, "Hardware/software cosimulation from interface perspective," *Computers and Digital Techniques, IEE Proceedings* -, vol. 152, no. 3, pp. 369 – 379, May 2005.

[6] I. Cox, F.L., W. Kuhn, J. Murray, and S. Tynor, "Code-level modeling in xspice," in *Circuits and Systems, 1992. ISCAS '92. Proceedings., 1992 IEEE International Symposium on*, vol. 2, May 1992, pp. 871 –874 vol.2.

[7] Synopsys Corporation, "HSIMplus HDL Co-Simulation," 2010. [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification/DesignAnalysis/Pages/HDLCo-Simulation.aspx