EEC 116 Fall 2011 Lab #4: Mixed-Signal Simulation Tutorial

Dept. of Electrical and Computer Engineering
University of California, Davis

Issued: October 10, 2011
Due: October 26, 2011, 4PM

Reading: Rabaey Section 5.5 [1].
Reference: Brunvand Chapter 7.6 [2].

OBJECTIVE

The objective of this lab is to create a test bench schematic and verify your adder design developed in Lab 2 using mixed-signal circuit simulation. You will also measure its average power consumption over a number of digital test vectors.

TOOL SETUP

No additional setup should be required for this lab.

TESTBENCH SCHEMATIC AND SIMULATION

In previous labs you have verified your circuits through analog simulation using Spectre and digital simulation using Verilog-XL. Analog simulation gives the most accurate results regarding the performance and power consumption of your circuits, but it can be very slow for large designs. Digital simulation is very quick but typically does not give particularly accurate results. *Mixed-signal* or *mixed-mode* simulation is a compromise between the two which allows the designer to specify parts of the design to be simulated using a Verilog simulator while other parts are to be simulated using Spectre. This is particularly convenient for creating digital stimulus vectors to exercise various aspects of a large digital circuit like the critical path of an adder.

V_{CC} **Power Supply Inverter** First, you will need to create some infrastructure to test your gate designs. Copy your `inv_1x` cell to another cell named `invCC_1x`. This new cell will be identical to the other, except it will use a different power supply. This is so we can isolate the power consumption of the DUTs from the inverters providing the digital stimulus. Edit the `schematic` view by replacing the `vdd` symbol with the `vcc` symbol from the `UCD_Analog_Parts` library. Copy the `inv` cell `functional` view from the `UCD_Digital_Parts` library to the `invCC_1x` cell in your EEC 116 library. If you edit the `functional` view for `invCC_1x`, a text editor (`gedit`) window will pop up with the behavioral code for the inverter. Edit the module name to match the name of your cell. Save the file and quit the editor. Edit the symbol view so that the module names, model name properties, and labels are all consistent with the name of the new cell. You will be instantiating this new inverter cell at the input and output of the DUTs.

32b Ripple-Carry Adder Next, create a schematic for a 32b ripple-carry adder in a cell named `adder32b` by instantiating eight copies of the 4b adder consisting of mirror adder cells (`adderM4b`) and connecting the addend inputs, carry input, sum output, and carry output appropriately. Create a symbol view for `adderM4b` if you haven't already. Also, create a symbol view for the 32b adder cell. Finally, create a layout and verify that it is free of DRC or LVS errors for the 32b adder. Show your schematic, symbol, and layout views to the TA or instructor and be prepared to run DRC and LVS for checkoff.

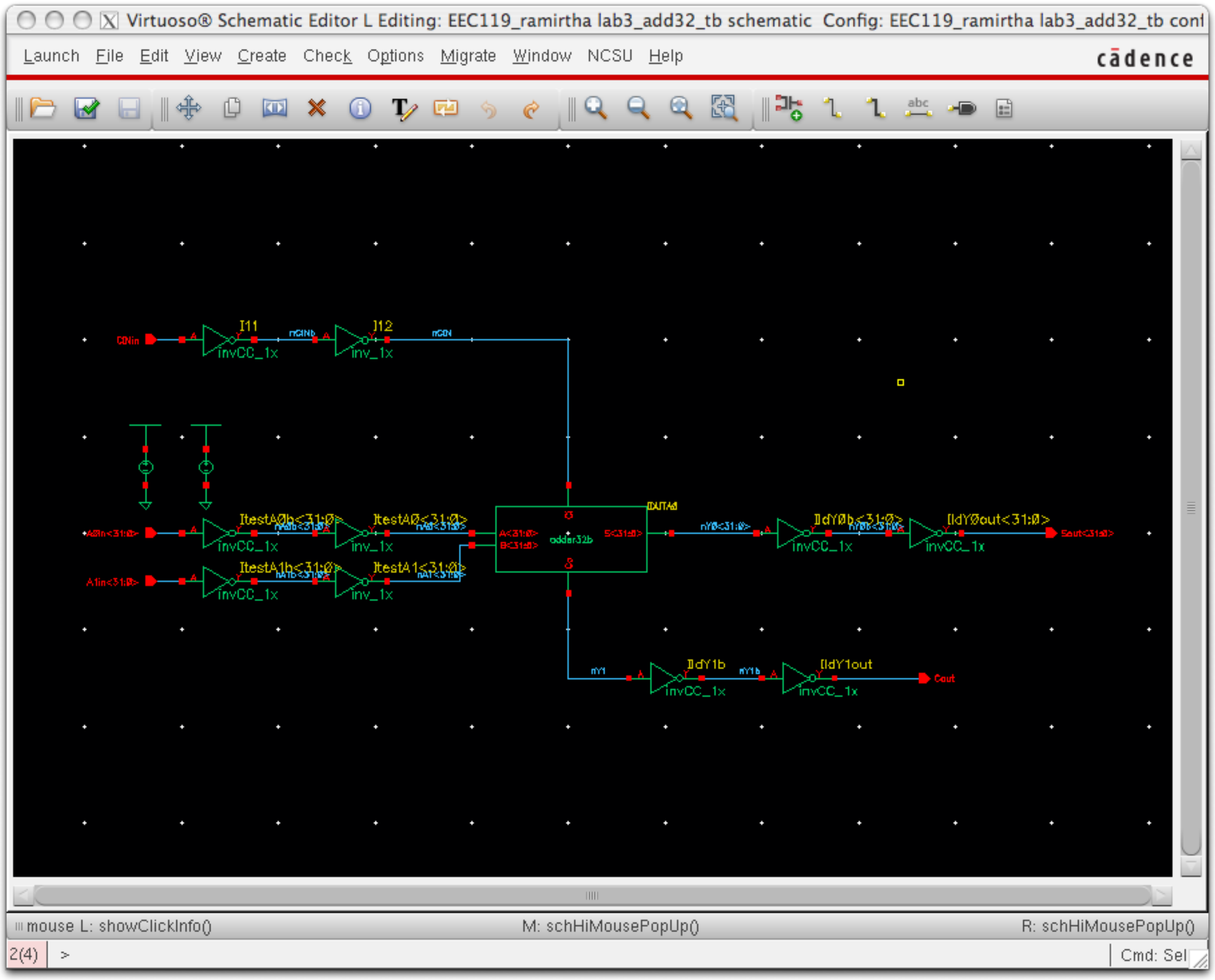


Figure 1: Mixed-mode cell simulation testbench schematic.

Part 1 Mixed-Mode Testbench Schematic and Simulation Setup Create a new schematic cell view using the Library Manager for a new cell called `lab3_add32_tb`. In this schematic, you will instantiate your DUTs, as well as additional components for testing them. In Figure 1, these additional components include two voltage sources for the analog simulation power supplies (like you had in the Spectre simulation testbench) and back-to-back instances of inverter cells from your library. Be sure to include the second DC voltage source to provide the `vcc` power supply. Having two inverters allows you to simulate one of the inverters in the digital domain and the second inverter in the analog domain to create an analog waveform to drive the DUT (which you are also simulating using the analog simulator). Similarly, the two inverters at the output of the DUT are a convenient way of converting the analog output voltage of the DUT to a digital signal which can be verified using behavioral Verilog code. Partition the design such that the `invCC_1x` inverters are at the digital input pins and output pins of the testbench. Implement the loads at the DUT output using the same inverters. However, the inverters which directly drive the inputs of the DUT should be `inv_1x` cells to account for the power associated with driving the input capacitance of the DUT.

To drive the mixed-mode simulation, you need to create a new cell view of the testbench schematic called a `config` view. Go to the Library Manager and select File→New→Cell View... Fill in the View Name field in the popup as `config`. This should automatically select **Hierarchy Editor** as the Tool. Click OK. Two nested windows should pop up named **Virtuoso Hierarchy Editor** and **New Configuration**. In the New Configuration window, select Use Template. In the Use Template dialog window, select `spectreVerilog` and click OK. The fields in the New Configuration window should now be filled in. Change the View: field to `schematic`, make sure the correct values are filled in the Library:, Cell:, and View: fields, and click OK to create the `config` view.

You should get a Hierarchy Editor window like the one shown in Figure 2. Click on the Tree View tab and some of the icons in the Tree View frame. A list of the instances and the components within each instance appears as you descend the design hierarchy, terminating with transistor instances from the `UCD_Analog_Parts` library. The key to mixed-mode simulation is to specify views which result in analog (Spectre) simulation for some cells and digital (Verilog) simulation for other cells. The `config` view describes each instance in the testbench schematic and allows you to choose the view used to simulate it. You are making this choice for a tree of cells, so any cells under the initial cell will inherit that choice by default. You can always descend the hierarchy further and change it to suit your needs. By stretching out the Instance bar at the top of the Tree View frame, you can see that every cell instance has the `schematic` view selected initially, implying that the entire simulation will be analog.

To proceed, we have to do two things: (1) specify the interface elements the mixed-mode simulation will use to connect the digital and analog portions of the circuit and (2) partition the circuit so some of the elements are simulated using Verilog. Save your `config` view and close the Hierarchy Editor window. Re-open the `config` view for the cell from the Library Manager window and select yes to open both the `config` and `schematic` views. Switch the Hierarchy Editor to Tree View and in the schematic window select Launch→Mixed Signal Options→Verimix to enable the **Verimix** pulldown menu in the menu bar at the top of the Schematic Editor window. Invoke Verimix→Interface Elements→Default Options... In the

IE Default Options popup, change the Default IE Library Name field to `UCD_Analog_Parts` and click OK. Now you can select Verimix→Interface Elements→Library and the **IE Property Editor** window will appear. You can now set the characteristics of the analog-to-digital (`a2d`) and digital-to-analog (`d2a`) interface elements, such as the input and output logic levels, rise/fall times, etc. First, set the Model IO pulldown menu to `input`. Then, set the Model Parameters `timex`, `vl`, and `vh` to 10p (10ps), 0.6 ($\sim \frac{1}{3}$ of the nominal V_{DD} of 1.8V), and 1.2 ($\sim \frac{2}{3}$ of the nominal V_{DD} of 1.8V), respectively. Click Apply. Change Model IO to `output`. Set the `d2a` rise and fall times to 2p and the output high and low to 1.8 and 0, respectively. Set `valx` to 0.9. Click OK.

In this testbench, since the inverters start with transistor-level schematics, we need to replace the inverter view to use with a `functional` (i.e., HDL) view. In the Hierarchy Editor window, selecting one of the instance names highlights the current instance in the schematic. For the five inverters at the inputs and outputs, click the View To Use field (it appears blank initially) and type in functional. Select View→Update to update the view and save the final version of the `config` view. You can double-check that your partitioning is correct by invoking

Verimix→Display Partition→All Active

in the schematic editor window. You should see highlighted in red all of the components and nets to be simulated in Spectre and highlighted in orange all the components and nets to be simulated in Verilog. The mixed-mode simulator has added implicit `d2a` and `a2d` components between the simulation domains and these are highlighted in blue. You may need to go back to the Hierarchy Editor and explicitly enter the View To Use as schematic for some of the cells. Remember, only the inverters connected to the input and output pins of the testbench schematic should be using functional views.

Part 2 Mixed-Mode Simulation

Now select Launch→ADE L from the schematic editor to start up the analog simulation environment. In the ADE window, choose Setup→Simulator/Directory/Host... and set the Simulator field to `spectreVerilog` and the Project Directory to

`/project/<username>/simulation`.

If the `spectreVerilog` option is not available, you may have to invoke Setup→Design... and select `lab3_add32_tb` and the `config` view to get it. In the Setup→Model Libraries... menu, make sure you fill in the same model file location you have used for Spectre simulations before. In the Setup→Environment... menu, click the Verilog Netlist Option.. button and make sure that the Generate Test Fixture Template, Netlist SwitchRC, Drop Port Range, and Preserve Buses boxes are all checked. Also, make sure the Generate Test Fixture Template pulldown menu is set to Verimix. Click OK twice to get back to the ADE window.

Now you need to create your digital testbench code as you did in the previous lab. Select Setup→Stimuli→Digital... and the text editor window should appear with some initial template code. Replace the code with a simple set of nested loops to exercise some test vectors through your adder and verify correct operation. An example is in the `add32test.verimix` file available on the class SmartSite. You may need to modify this file to match the names of your inputs and outputs.

Set the analysis mode to `tran` and configure the analysis to simulate the circuit for 4020ns using `conservative` accuracy. If you need to set any design variables and values, do those as well. Select signals to be plotted as you did in the previous lab. Be sure to include some digital inputs and outputs of the testbench as well as some analog inputs and outputs of the DUTs (there are too many signals to try to plot all of them, but you should include the analog and digital carry in and carry out signals and at least one bit that changes from each input and the sum output). One issue that often crops up in mixed-mode simulation is the initial convergence of the analog circuits. To help Spectre converge, invoke Simulation→Convergence Aids→Initial Condition ... from the schematic editor window and once the pop up appears, initialize the input nodes of your DUT to 0V (the same value as the Verisim digital stimulus would apply). Netlist and run the simulation. The waveform plot window should appear in addition to two logs: one for the digital simulation and the other for the analog simulation. Look over the log files and you should see that the Verilog simulation has printed out the correct values for the gate outputs. In the waveform plot window, convert the plot to a strip chart by clicking on the four horizontal rectangles icon. At the top you should see digital waveforms for the two inputs and two outputs and analog waveforms below for the other signals. Show the plot to your TA or the instructor for checkoff. Print the plot and turn it in with your project report.

Calculating Power Consumption Instantaneous power is the product of the current drawn from the power supply and the power supply voltage ($i_{DD}(t)V_{DD}$). Current waveforms can be plotted by selecting the terminals of a voltage source or other circuit element using the same menus as you used above. They will be highlighted as circles rather than the entire net being highlighted. To compute the average power, you can use the Visualization and Analysis calculator which can be invoked from the ADE window by selecting Tools→Calculator ... to integrate the current waveform, multiply it by the supply voltage, and divide by the duration of the integration interval. More details are available from an excerpt from Erik Brunvand's book [2] which is posted on the class SmartSite. Note that Brunvand's book refers to an earlier version of Cadence, so the menus and prompts may not be exactly the same as what we are using in EEC116. Compute the average power dissipation for the test vectors applied in the simple looped test bench code and record it in the summary/lab report cover sheet.

Power Characterization Stimulus A 32b adder has $2^{32} \cdot 2^{32} \cdot 2 = 2^{65}$ combinations of input vectors. It is obviously impractical to simulate all of them (it would take about 300 years on a 4GHz processor) to completely characterize the power consumption. Moreover, you have to consider sequential combinations of vectors because those may affect the power consumption as well. In this part of the lab, you must develop your own digital stimuli to provide a reasonable estimate of the average power consumption of your 32b adder. Run the mixed-mode simulation using your stimulus and be prepared to hand in a listing of the code. Consider the following things about the test vectors when designing your stimulus:

1. Number
2. Values
3. Statistical distribution
4. Sequence

You may go over your stimulus design and get feedback from the TAs or the instructor. Record the average power consumption reported by your stimulus. How does it compare to power reported using the simple test vector loop above?

Checkoff

Show your completed schematics, stimulus file listing, and all waveform plots to the TA for checkoff.

Report

You must hand in a **typewritten** report to receive credit for this lab. Your report can be brief, but must include the following sections in addition to the completed summary sheet attached at the end of this lab. The summary sheet will be the cover page of your lab.

1. Overview: Describe in one paragraph the objectives of the lab. State what you were testing and what data you expected to gather as a result of your experiments.

2. Procedure: Briefly document your methodology for acquiring the data you describe in the Overview. Describe how you measured the simulated current and computed the average power. Describe your strategy for generating your power characterization stimulus vectors. Someone reading this section should be able to easily duplicate your results by following the methodology described in this section.

3. Results and Discussion: Describe succinctly the power results captured in the completed summary sheet tables. Do the results make intuitive sense? If not, explain why they might contradict your intuition.

Acknowledgments

Parts of this lab were inspired by lab exercises developed by Prof. David Money Harris and others at Harvey Mudd College for the class E158: Introduction to CMOS VLSI Design.

References

[1] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 2003.

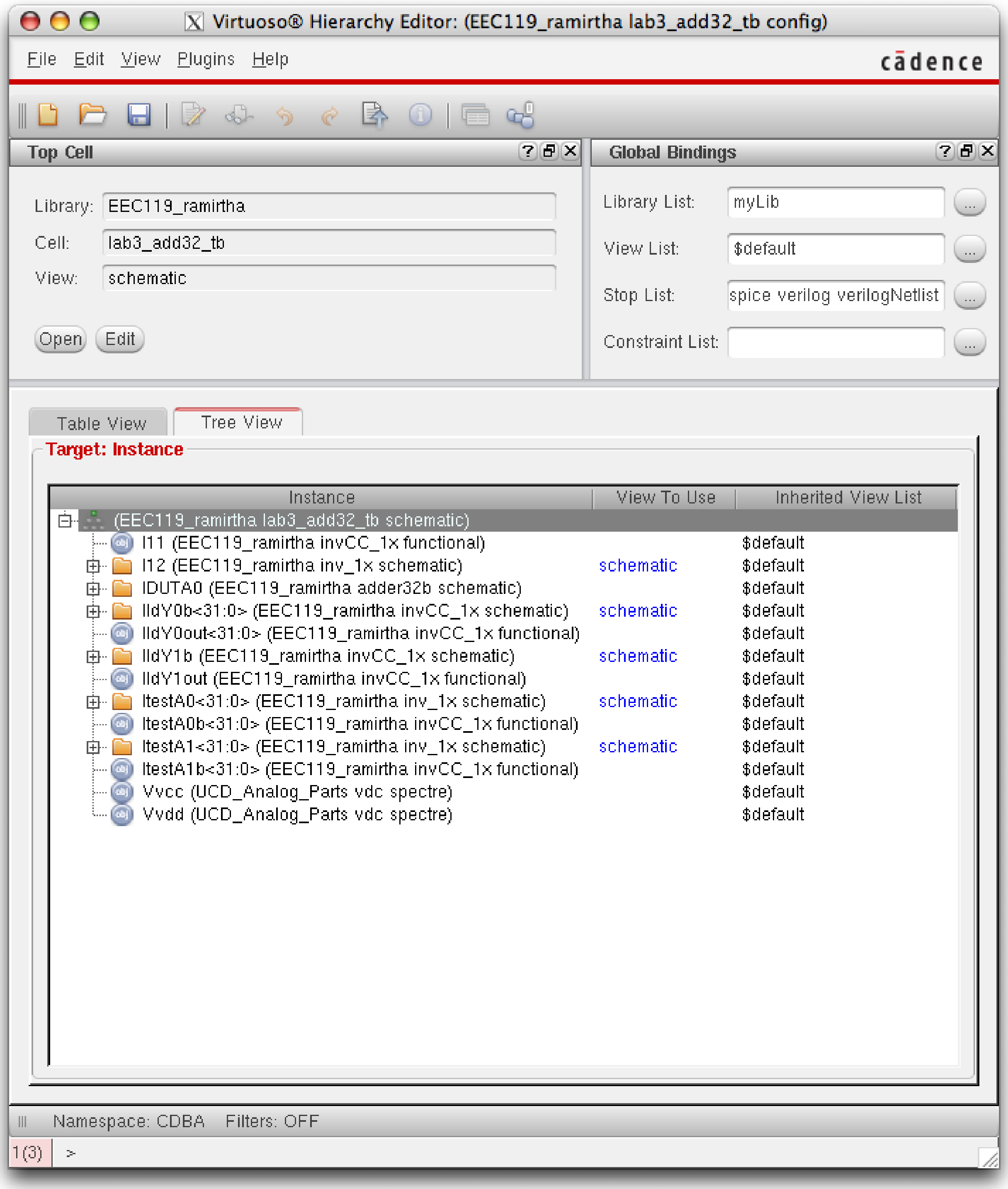
[2] E. Brunvand, *Digital VLSI Chip Design with Cadence and Synopsys CAD Tools*, 1st ed. San Francisco: Addison-Wesley, Inc., 2010.

EEC 116 Fall 2011 Lab #4 Summary

Name:

Grading:

Part	Checkoff	TA Initials	Date
`adderM4b` Symbol			
`adder32b` Schematic			
`adder32b` Symbol			
`adder32b` Layout			
`adder32b` DRC			
`adder32b` LVS			
2 Mixed-Mode Sim Waveform Plot			
2 Power Simulation Stimulus File			
	Value		
2 Power (Nested Loop Test)			
2 Power (Your Test)			



Virtuoso® Hierarchy Editor: (EEC119_ramirtha lab3_add32_tb config)

File Edit View Plugins Help

Top Cell

Library:	EEC119_ramirtha
Cell:	lab3_add32_tb
View:	schematic

Global Bindings

Library List:	myLib
View List:	$default
Stop List:	spice verilog verilogNetlist
Constraint List:	

Target: Instance

Instance	View To Use	Inherited View List
(EEC119_ramirtha lab3_add32_tb schematic)		
I11 (EEC119_ramirtha invCC_1x functional)		$default
I12 (EEC119_ramirtha inv_1x schematic)	schematic	$default
IDUTA0 (EEC119_ramirtha adder32b schematic)		$default
IIdY0b<31:0> (EEC119_ramirtha invCC_1x schematic)	schematic	$default
IIdY0out<31:0> (EEC119_ramirtha invCC_1x functional)		$default
IIdY1b (EEC119_ramirtha invCC_1x schematic)	schematic	$default
IIdY1out (EEC119_ramirtha invCC_1x functional)		$default
ItestA0<31:0> (EEC119_ramirtha inv_1x schematic)	schematic	$default
ItestA0b<31:0> (EEC119_ramirtha invCC_1x functional)		$default
ItestA1<31:0> (EEC119_ramirtha inv_1x schematic)	schematic	$default
ItestA1b<31:0> (EEC119_ramirtha invCC_1x functional)		$default
Vvcc (UCD_Analog_Parts vdc spectre)		$default
Vvdd (UCD_Analog_Parts vdc spectre)		$default

Namespace: CDBA Filters: OFF

Figure 2: Mixed-mode simulation config view Hierarchy Editor window.