EEC 118 Spring 2011 Lab #5
Manchester Carry-Chain Adder

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

Issued: May 9, 2011
Due: May 20, 2011, 5 PM in 3173 Kemper.

1 OBJECTIVE

The objective of this lab is to use Cadence to build and simulate dynamic CMOS circuits for building fast adders.

2 PRELAB

There is no prelab for this lab.

3 MIXED-MODE SIMULATION TUTORIAL

You have so far verified your circuits through analog simulation using Spectre. 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 (for example, using a hardware description language (HDL) simulation like Verilog) 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.

Part 3.1 Mixed-Mode Testbench Schematic and Simulation Setup Create a new schematic cell view using the Library Manager for a new cell called `lab5MMtut_tb`. In this schematic, you will instantiate your DUTs (the NAND2 gate and AND2 gate you created in Lab 3), as well as additional components for testing them. In Figure 1, these additional components include a voltage source for the analog simulation power supply (just like you had in the Spectre simulation testbench) and back-to-back instances of the `inv_1x` cell from

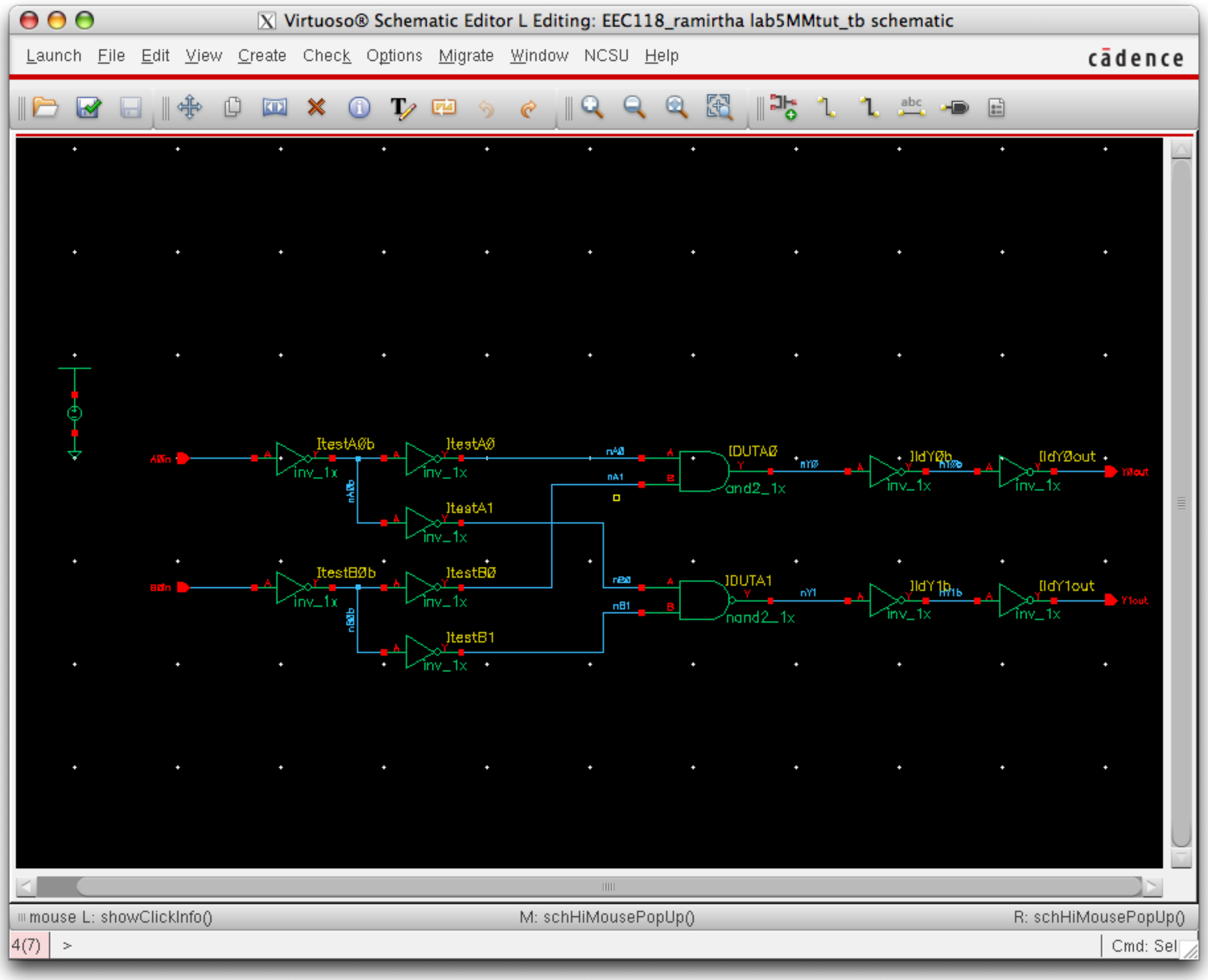


Figure 1: NAND2/AND2 gates mixed-mode cell simulation testbench schematic.

your library. 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.

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. Copy the `inv` cell `functional` view from the `UCD_Digital_Parts` library to the `inv_1x` cell in your EEC 118 library. If you edit the `functional` view for `inv_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.

In the Hierarchy Editor window, selecting one of the instance names highlights the current instance in the schematic. For the four 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 3.2 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→Design... and select `lab5MMtut_tb` and the `config` view. Choose Setup→Simulator/Directory/Host... and set the Simulator field to `spectreVerilog` and the Project Directory to `/home/<username>/eec118/simulation`. In the Setup→Model Libraries... menu, make sure you fill in the same model file location you used in earlier labs. 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. Select Setup→Stimuli→Digital... and the text editor window should appear with some initial template code in the Verilog language. Edit the file so that it looks like the code in Figure 3. Save the file and close the text editor. A copy of the file is also available on the EEC118 SmartSite. To help you understand the stimulus file, some lecture notes on Verilog are on the SmartSite and numerous other resources for learning Verilog syntax are available online.

Set the analysis mode to `tran` and configure the analysis to simulate the circuit for 40ns 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 labs. Be sure to include the digital inputs and outputs of the testbench as well as the analog inputs and outputs of the DUTs. 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 to your DUTs to 0V. Netlist and run the simulation. The waveform plot window should appear in addition to two logs: one for the digital simulations 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. Print the plot and turn it in with your project report. You should get something similar to Figure 4.

4 ADDER DESIGN AND SIMULATION

In this section of the lab, you will design and verify through mixed-mode simulation a 32 bit ripple-carry adder implemented using a Manchester carry-chain. A brief description of the circuit is provided in Chapter 11 of the Rabaey book [1]. To build up the adder, you must decide how to use design hierarchy to best effect.

4.1 Propagate and Generate

One approach to simplifying adder implementation is to use logic gates which produce *propagate* and *generate* signals. These intermediate signals can then be combined to create the sum and carry out signals for each pair of summand bits. Create a new cell called `PGgate` which takes individual input bits A and B and produces the two outputs P and G for propagate and generate, respectively. You can use any static (nonclocked) circuit style to implement these gates, including the transmission gate XOR circuits described in the book. Create both a schematic and a symbol view for the circuit and show them to the TA or instructor for checkoff.

4.2 32b Ripple-Carry Adder

You should now create a schematic for a 32b ripple-carry adder in a cell named `adder32b`. In addition to the `PGgate`, you will need cells to create the sum output bits and a circuit to transmit the carry bits from LSB to MSB. This carry circuit will use the dynamic Manchester carry-chain circuit shown in Figure 11-9 of the Rabaey book. You must decide whether you want to create a 32 pass transistor version of the carry chain or whether you want to buffer the chain using inverters to speed it up. The design choices are up to you, but you will need to document and justify them in your report. Choose transistor sizes such that the adder can operate at 100MHz. Also, create a symbol view for the 32b adder cell, which includes the summands, LSB carry in, a CLK for the dynamic circuits, the sum output, and the carry output. Show your schematic and symbol views to the TA or instructor for checkoff.

4.3 32b Ripple-Carry Adder Testbench and Stimulus

Create a new schematic cell view using the Library Manager for a new cell called `lab5_add32_tb`. In this schematic, you will instantiate your DUT, as well as additional components for testing them. You should end up with a schematic similar to Figure 5. Note that you can simplify the schematic (and save work) by using buses. The input and output pins use vector syntax (e.g., `<31:0>`) to identify signals and instances which correspond to buses and multiple copies of the same cell.

Run a mixed-mode simulation with test vectors applied using a Verilog stimulus file, similar to what you did in the tutorial. Choose vectors so that you can simulate and measure the **worst case** propagation delay of the critical path of the adder. Plot the analog output waveforms showing this worst case delay. Identify the critical path, show how your stimulus file exercises this path, show the plot, and confirm your measured propagation delay with the TA or instructor for checkoff.

Checkoff

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

Report

Write up your design decisions and simulation results in a lab report. 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. Be sure to include schematics of your circuits, your stimulus vector file listing, and any waveform plots.

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

2. Design and Verification: Describe your design choices for the propagate and generate logic, the carry-chain, and the sum logic and how these influence the delay of the critical path. Summarize how you chose the test vectors in your stimulus file to exercise this critical path.

3. Results and Discussion: Enter your critical path measurement result into the summary table. Does the result make intuitive sense? If not, explain why it might contradict your intuition. How would you go about decreasing the delay?

EEC 118 Spring 2011 Lab #5 Summary

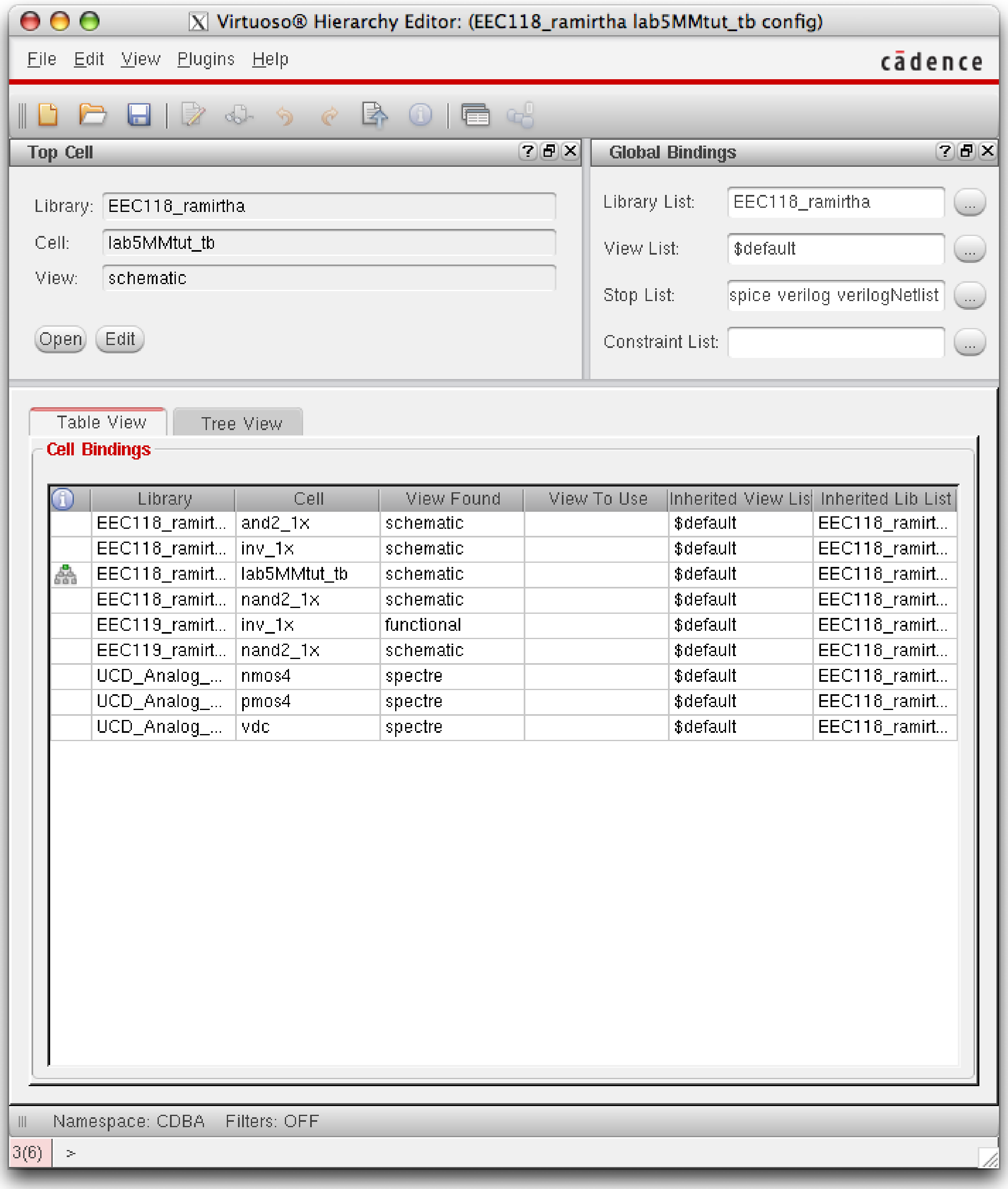
Name:

Grading:

Part	Checkoff	TA Initials	Date
3.2 Mixed-Signal Waveform Plot			
4.1 `PGgate` Schematic			
4.1 `PGgate` Symbol			
4.2 `adder32b` Schematic			
4.2 `adder32b` Symbol			
4.3 Critical Path Stimulus File			
4.3 Critical Path Waveform Plot			
	Value		
4.2 Critical Path Delay			

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.



Virtuoso® Hierarchy Editor: (EEC118_ramirtha lab5MMtut_tb config)
File Edit View Plugins Help
cādence
Top Cell
Global Bindings
Open Edit
Table View Tree View
Cell Bindings

Library: EEC118_ramirtha
Cell: lab5MMtut_tb
View: schematic

Library List: EEC118_ramirtha
View List: $default
Stop List: spice verilog verilogNetlist
Constraint List:

	Library	Cell	View Found	View To Use	Inherited View Lis	Inherited Lib List
	EEC118_ramirt...	and2_1x	schematic		$default	EEC118_ramirt...
	EEC118_ramirt...	inv_1x	schematic		$default	EEC118_ramirt...
	EEC118_ramirt...	lab5MMtut_tb	schematic		$default	EEC118_ramirt...
	EEC118_ramirt...	nand2_1x	schematic		$default	EEC118_ramirt...
	EEC119_ramirt...	inv_1x	functional		$default	EEC118_ramirt...
	EEC119_ramirt...	nand2_1x	schematic		$default	EEC118_ramirt...
	UCD_Analog_...	nmos4	spectre		$default	EEC118_ramirt...
	UCD_Analog_...	pmos4	spectre		$default	EEC118_ramirt...
	UCD_Analog_...	vdc	spectre		$default	EEC118_ramirt...

Namespace: CDBA Filters: OFF

3(6) >

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

```
// Vermix stimulus file.
// Default verimix stimulus.

initial
begin

   A0in = 1'b0;
   B0in = 1'b0;

   #5 $display("A0in = %b, B0in = %b, Y0out = %b",A0in,B0in,Y0out);
      $display("A1in = %b, B1in = %b, Y1out = %b",A0in,B0in,Y1out);

   #5 B0in = 1'b1;

   #5 $display("A0in = %b, B0in = %b, Y0out = %b",A0in,B0in,Y0out);
      $display("A1in = %b, B1in = %b, Y1out = %b",A0in,B0in,Y1out);

   #5 B0in = 1'b0;
      A0in = 1'b1;

   #5 $display("A0in = %b, B0in = %b, Y0out = %b",A0in,B0in,Y0out);
      $display("A1in = %b, B1in = %b, Y1out = %b",A0in,B0in,Y1out);

   #5 B0in = 1'b1;

   #5 $display("A0in = %b, B0in = %b, Y0out = %b",A0in,B0in,Y0out);
      $display("A1in = %b, B1in = %b, Y1out = %b",A0in,B0in,Y1out);

end
```

Figure 3: Digital stimulus file.

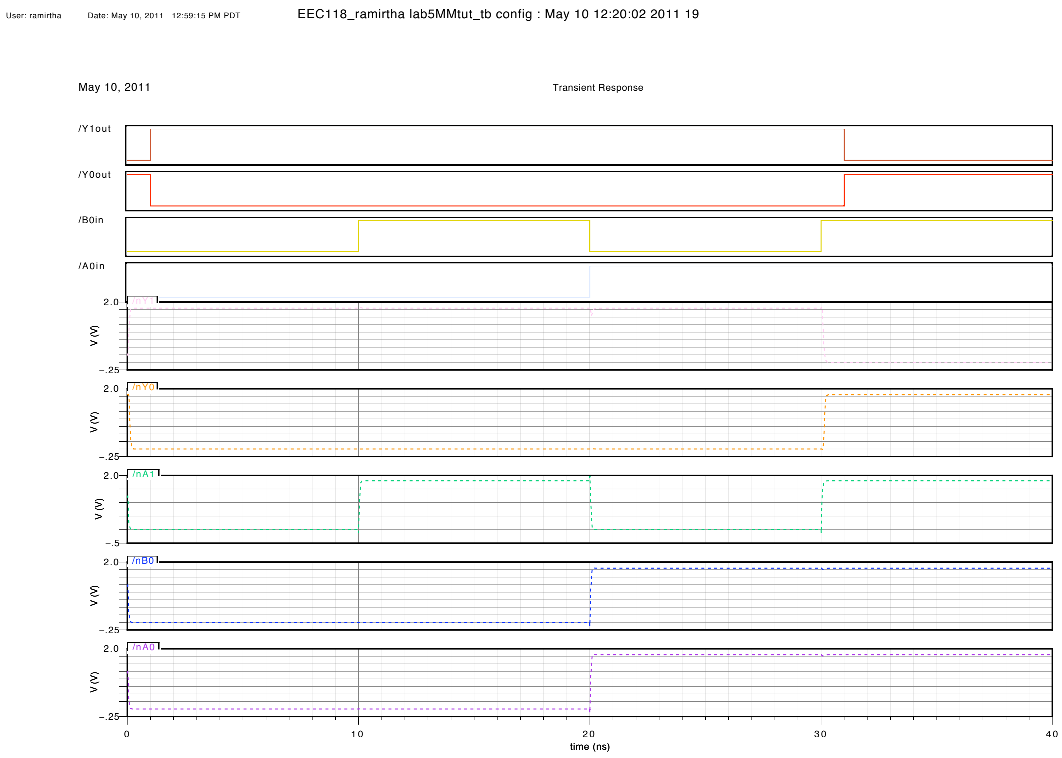


Figure 4: Mixed-signal simulation waveform plot.

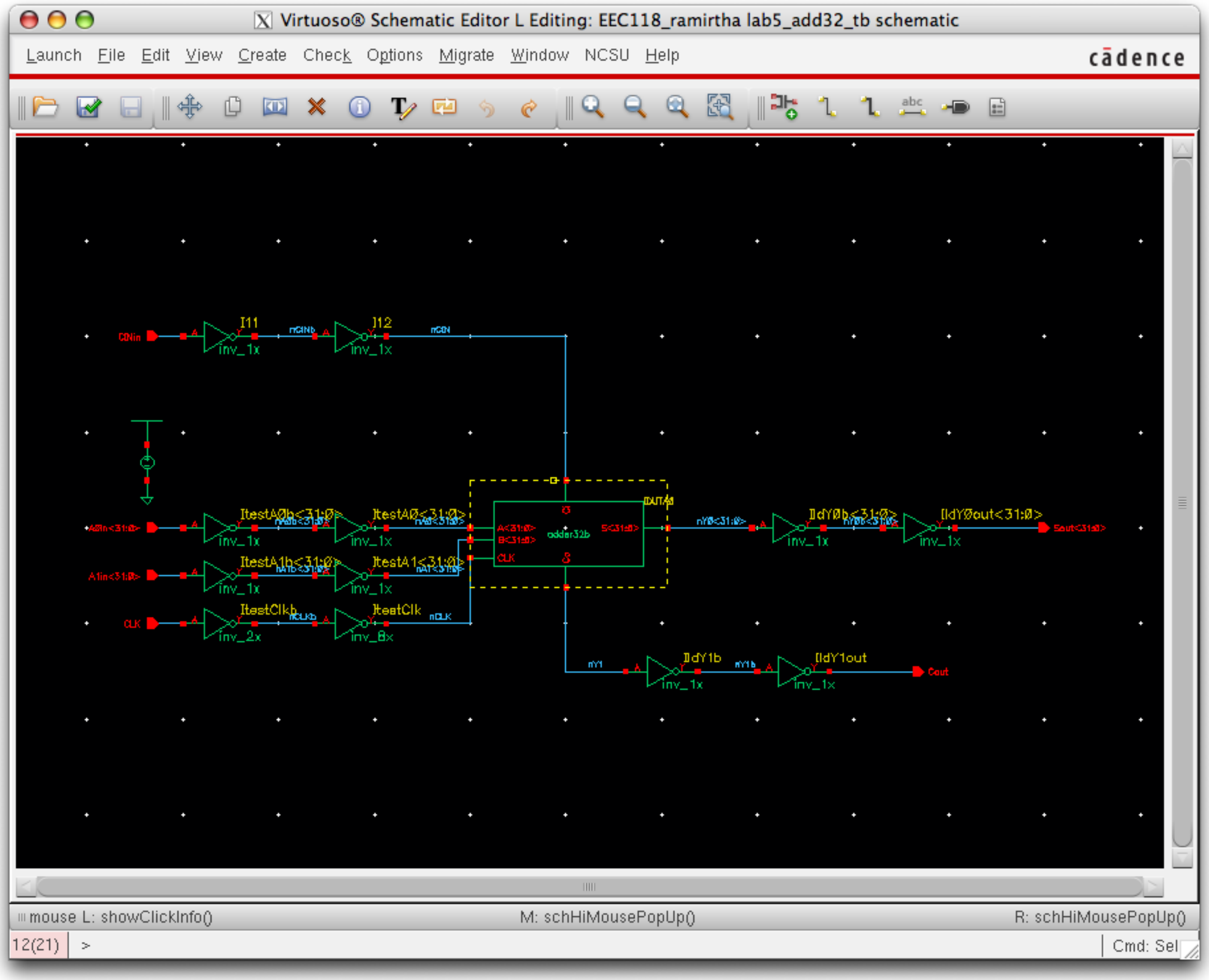


Figure 5: Mixed-mode `adder32b` simulation testbench schematic.