ECE/CS 6770 – Lab Assignment 8

Due 20 April 2017 via Canvas

1 Introduction

Noise is a significant design issue that must be taken into consideration in any design. In this lab we will focus on the power supply noise. Power supply noise is particularly problematic for deep submicron designs due to the complexity of the chips as designs can have over a billion transistors, the complexity of the power delivery network, and constraints on wire widths in these processes that are used for power delivery. Further, as voltages are dropped and power dissipation increases, large $\partial I/\partial t$ variation will result in increasing power supply noise.

Noise is one of the reasons, as we have discussed in class, that the results of a fabricated design will vary from what we measure in traditional simulation conditions, as performed using ModelSim. Noise presents a significant problem not only to the power supply, but also in terms of both power and delay to signal wires. For example, from the perspective of the signals on a chip, any voltage increase above (below) ground (power) can result in an exponential increase in leakage of any gates that it drives. This can also result in a significant change in the delays of a signal. In the multiple input switching lab, you needed to make sure that signals were pulled fully to the power and ground rails our your delay measurements and comparisons could be substantially skewed. Likewise, as there is noise on the power supply, the delay through the gate will vary in proportion to the supply noise. For first droop voltages the voltage drops can be substantial, up to 20% or more of the nominal power supply.

In this lab you will observe the impact of the package on power delivery to a chip as well as the signal pins. We will ignore the impact of the on-chip supply noise issues.

2 Power Delivery Analysis Tools

This lab will introduce you to a tool that is employed to evaluate the quality of the on-chip power delivery of large integrated circuits. We will be using a tool from Synopsys called HSim Plus.

Chips contain both active and passive components. The models for a simple transistor are quite complex, as its operation moves through several distinct modes of operation ranging from exponential current increase in the shutoff region, linear current, and saturated current. The analysis of million- to billion-transistor designs that include the effect of power supply noise are too complex to perform using transistor models.

Thus, a common evaluation method being used today for large chips is to employ a *two phase* approach. First, the circuit is decomposed into smaller design blocks. These blocks are then simulated using an ideal power supply. The current for each of these blocks is measured, and modeled as a current source. These represent the stresses placed on the power delivery network. This is the first analysis phase. For the second phase the current sources are inserted into a model of the power delivery network. This results in a model that consists entirely of passive components. This simplified model is then simulated to determine the response of the power delivery network to the current stresses applied to it.

The HSim Plus tool performs two phase analysis on a chip. The input to HSim Plus is a spice netlist of the design and a spice control file. The simulation vectors must be provided in the control file, and the control file is compatible with spice. All of the steps of the two phases are automatic. Various parameters to the tool trade off accuracy and run time and other parameters to the phases of operation.

3 Project Definition

For this lab you will perform both HSPICE simulations and HSim Plus simulations on a small block. This block is not large enough to take advantage of the two phase analysis, but will give you an idea of the tool usage and flow.

3.1 Synthesis

Spice netlists are required to create accurate models for the two phase analysis approach used to model power delivery networks. You need to have a fully synthesized design in the 65 nm library that we are using this semester to complete this lab.

3.2 Spice Netlist Creation

In this lab you will need to generate a spice netlist for your circuit in order to perform an accurate HSPICE simulation of the design. We will use a verification tool from Mentor Graphics to perform this step.

In a previous lab you were to perform design rule checking (DRC) and layout versus schematic (LVS) verification. The LVS flow maps the schematic to a spice netlist equivalent, and does a comparison of a spice netlist of your design. Thus to perform LVS a spice netlist is required. Mentor's Calibre tool flow uses the tool v2lvs to translate a structural Verilog file into a spice netlist.

The following steps will create a spice netlist

1. You will need to source the mentor setup scripts.
2. Run the tool v2lvs.
3. Run an emacs script to properly format the spice netlist for HSPICE and HSim Plus.

The following two commands accomplish the first two steps. The resultant file will be named v2lvs.sp.

```
source /uusoc/facility/cad_common/local/setups/F13/setup-mentor
v2lvs -v mult.dcopt.v -o v2lvs.sp -addpin VDD -addpin VSS -i \
   -lsp /uusoc/facility/cad_common/Artisan/GF/cmos65g/aci/sc-adv12/lvs_netlist/cmos10sfrvt_a12.cdl
```

All engineers are experts in Emacs (right?), therefore I have written an emacs lisp script to convert the spice list used internally by the Mentor tools into one that works for HSPICE[1]. To perform the conversion, you will need to do the following. The script can be found on Canvas.

[1]This of course could have been written as a stand-alone perl script, but what is the fun in that?

1. Start up emacs with the command `emacs &`
2. Open the file v2lvs-convert.el with the command in emacs:
   `C-x C-f v2lvs-convert.el`
3. Define the convert-v2lvs function with the command:
   `M-x eval-buffer`
4. Open up your v2lvs.sp file with the command:
   `C-x C-f v2lvs.sp`
   (note: you may need to include a directory path to where the file is located)
5. Run the conversion script with the command:
   `M-x convert-v2lvs`
6. Save out the modified file with the command:
   `C-x C-s`

You now have a spice file that HSPICE and HSim Plus can use. Note that the emacs commands `C-x` means to hold down the control key and press the `x` key. The commands M-x means you can either hold down the `Meta` key and press the `x` key, or press the `Esc` key followed by the `x` key. There are ways to do this from the menu as well (if you are a windoze user).

3.3 Control List Creation

The model for a commercial package is available on Canvas. This models a package that Micron Semiconductor uses for some of its memory products. The package models resistors, capacitors, inductors, mutual inductors and other passive components that exist between a PC board and the chip. This will give you a realistic picture of what the signals on the chip side of a package really look like.

To perform simulations you will need to create a control file that contains the package and your design. I have placed a preliminary control file on Canvas that will get you started. The package is modeled with pin labelings on the PCB side (external ball grid array of the package), and the CHIP side where the chips will be wire bonded or connected with a ball grid array to the package. Please note the following regarding this control file.

- The file starts by including a number of parameters that control the operation of the simulations.
- Next the circuits are loaded. These consist of your v2lvs.sp file and the package model file. Make sure the path to your spice file is correct.
- Include the spice models for the library cells.
- Include the simulation models used for the circuit.
- Define the package as a sub-circuit with connections in this design. The PCB ports of the package will be driven with DC signals and clean waveforms.

- After your circuit you will need to place resistors and capacitors between the CHIP_DXX pins and the pins that will go to your ALU design. Most pads have a resistor placed in series with the input signals as ESD protection. They can also help protect the design against internal noise. Place a 300 ohm resistor and 75pF capacitor in series with the package input pins and your design to model these resistors and capacitors.

- Include a definition of your design. Note that there are not enough I/O pins in this package for your whole design. I suggest connecting the data inputs for both the A and B ports of your design to the chip pins CHIP_D00 through CHIP_D07. There are four other data pins available. Connect two to your clock and reset pins, and the remaining two to the low order bits of your 16 bit outputs. Make the remaining output pins available for observation by HPSICE and HSimp Plus.

- Define the input waveforms and voltages to the package from the PCB. I have included these in the sample control file on Canvas, but feel free to improve the waveforms you use in your design.

- Finally, a number of transient simulation and plotting commands are defined. Note that HSPICE will place all the internal signals into the .tr0 file due to the save command. However, for HSim Plus, you need to explicitly note with a PLOT command the signals that you need to observe. I have included all the signals in the chip where the DXXR signals are the signals after the resistor. You can either use my naming convention or change these to suit your needs.

3.4 Simulate the Circuits

Perform analysis using both HSPICE and HSim Plus. Use the following commands to run these two tools:

```
time hspice -i packagedalu.control -o results
time syn-hsimplus -i packagedalu.control -o hsim
```

The time command reports the run-time of the tool. Please include in your results the output of the `syn-hsimplus` run. This file is saved as the file hsim.log.

Note that HSPICE stores its results in a binary file .tr0, whereas HSim Plus stores its results in a binary file .fsdb. In order to plot the HSim Plus waveforms in syn-cscope you will need change the “Files of Type:” drop-down list to “FSDB” or “All Plotfiles” from the File | Open | Plotfiles popup menu.

(NOTE: It looks like the latest version of HSPICE and the older version of syn-cscope are incompatible. Make sure you are running the F15 version of HSPICE.)

4 Deliverables

Perform one simulation run in both HSim Plus and HSPICE where all internal power and ground signals are connected together, and these connect to your chip. Perform a second simulation where only one of the power and ground pins from the package are connected to your chip module. This is done by connecting only a single power and ground pin through the package to your single multiplier instance. (You would need to change one of the CHIP_PWR and CHIP_GND pins to another net name, and connect those to your multiplier power and ground.)

Include a README file that contains your description of the lab, what you learned and any problems that you encountered. Also describe all of the files that are turned in with a brief description of what they contain.

Include the files and results in a .tgz file that has the following:

1. Include the .tcl script files and the two control files that you created for your simulations. Include all log files created from the synthesis and simulation runs including the hsim.log file.

2. Plots of waveforms of the power and ground noise for both the HSPICE and HSim Plus runs. In one plot put the power signals from HSPICE and HSim Plus. In another plot the ground signals from the two tools. In a third, plot the power and ground signals together for the two tools in a single plot. Do this for the two runs, connecting your design to all power and ground pins, and one with only one of the power and ground pins connected. Include the power and ground plots for each of these runs. (Use xv to generate a .png or other file for the plots.)

3. A plot of the clock waveform into your circuit. In one plot include the PCB signal, the signal before the resistor and capacitor, and the signal after the resistor and capacitor. Do this for each of the tools. In another plot, include the signal connected to your chip after the resistor for both of the tools so you can see the difference in accuracy between the tools.

4. In the README file, include descriptions of the following:

   (a) List the run times of the two simulations. Why do you think there is a difference?

   (b) Describe the effect of the package on the clock signal.

   (c) Why do you think there are so many power and ground pins on the package?

   (d) Evaluate the difference between the simulations with a single power pin connected to your chip and multiple power pins.

   (e) What is the difference between the signals coming right off the package and those following the RC circuit?

   (f) Describe what you think the signals would look like if you ran the same simulation but included 1,000 of your ALU circuits, with a correctly extracted RLC network for the on-chip power delivery system.

   (g) What is the purpose of decoupling capacitors on a chip?

   (h) What effect does inductance have on the reset signal?