CPE/EE 427, CPE 527, VLSI Design I:
Power Analysis Using Cadence Encounter®

Ashkan Ashrafi, Joel Wilder and Aleksandar Milenkovic, ECE Dept., The University of Alabama in Huntsville

1. INTRODUCTION

In this short tutorial, a simple method is discussed to evaluate power consumption of a digital integrated circuit, which has been designed by Cadence Encounter. There are two different power calculation methods: Static and Simulation based. In the static method, the probability of clock switching is given to the circuit and the power consumption of the chip is evaluated on that "guesstimate". This method is not accurate and should not be used when an accurate estimate is desired.

The simulation-based method gives results that are more realistic. In this method, parasitic components of the circuit are extracted first, and then a file (SDF file) containing both gate information and RC parasitic information is created. Then this file will be simulated along with the Verilog netlist, which has been previously created by Encounter, combined with a carefully prepared testbench, which provides the switching information file. The result of this compilation is a VCD file, which will be given to Encounter to evaluate power consumption of the chip.

The testbench plays a critical role in the power evaluation process. In the testbench, the designer must provide all the possible switching conditions for the circuit (i.e., what are the operating cases of the circuit). This can be done by applying all the possible input/clock combinations for the system. In many cases, providing all the possible combinations will lead to a gigantic VCD file. To avoid a big VCD file, the designer should reduce the number of input/clock combinations by defining a *representative* number of input vectors in the testbench. The evaluated power will provide a reliable estimate of the true power consumption of the chip if stimulus vectors are used that *represent* the actual function of the chip. In the following sections, it is shown how to evaluate power consumption of a previously placed and routed digital chip.

You will perform this work on the lab 4 accumulator design, which was based on the 0.5um AMI nwell process (lambda = 0.30um).

2. PREPARE THE CADENCE TOOLS

From your home directory, change directories into your cadence working directory:

% cd cadence

Change directories into the lab 4 encounter work area:

% cd lab4/encounter

Initialize the cadence tools:

% uah-cadence-setup

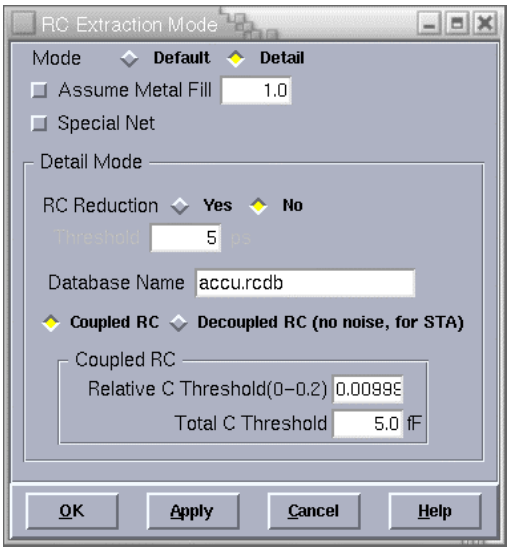
3. CREATE SDF FILE

Run Encounter in order to place-and-route the accumulator design from lab 4.

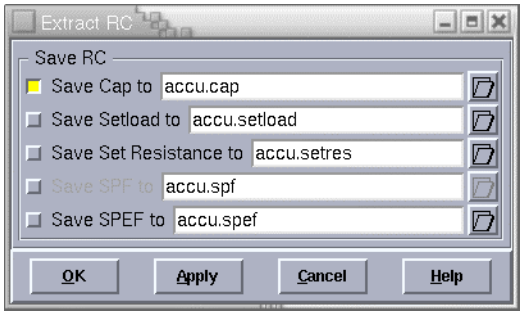
```
% encounter –init encounter.tcl
```

(Note that you can save the place-and-route results in Encounter by going to `Design->Save Design…` Then, in the future you can go to `Design->Restore Design` and load the accu.enc file in order to recover your design.)

Now go to `Timing > Specify Analysis Condition > Specify RC Extraction Mode.` A window will pop up. Change the window to this:

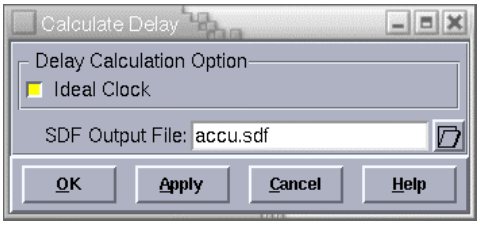


Hit “OK” and go to `Timing > Extract RC` to see this menu pop up:



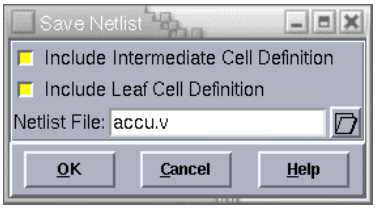
Hit "OK" again. Now RC parasitic of your design has been evaluated and you are ready to create the SDF file that contains your circuit along with the routing information and parasitic capacitances and resistances.

Go to `Timing > Calculate Delay` and you will see the following window:



Select a name for your SDF file. For instance, select "accu.sdf" and hit "OK". You should also create the final netlist file. This file is created by the automatic command line place and route process but it you will save it here to a different name.

Go to `Design > save > Netlist` and the following window will pop up:



In a terminal window, go to your encounter working directory and make sure that the files "`accu.sdf`" and "`accu.v`" are there. You should open "accu.v" using a text editor and add the following line at the beginning of the file:

```
`timescale 1ns/10ps
`celldefine
```

and the following line at the end of the file:

```
`endcelldefine
```

Now you should create the testbench. To create the testbench you can use the following command in the terminal:

```
% nedit testbench.v
```

"Nedit" is a smart text editor and it understands different languages including verilog!

The following figure shows the testbench contents. Make sure to save the "testbench.v" file before you close the text editor.

testbench.v – /home/grad/wilderj/cadence/lab4/encounter/

File Edit Search Preferences Shell Macro Windows Help

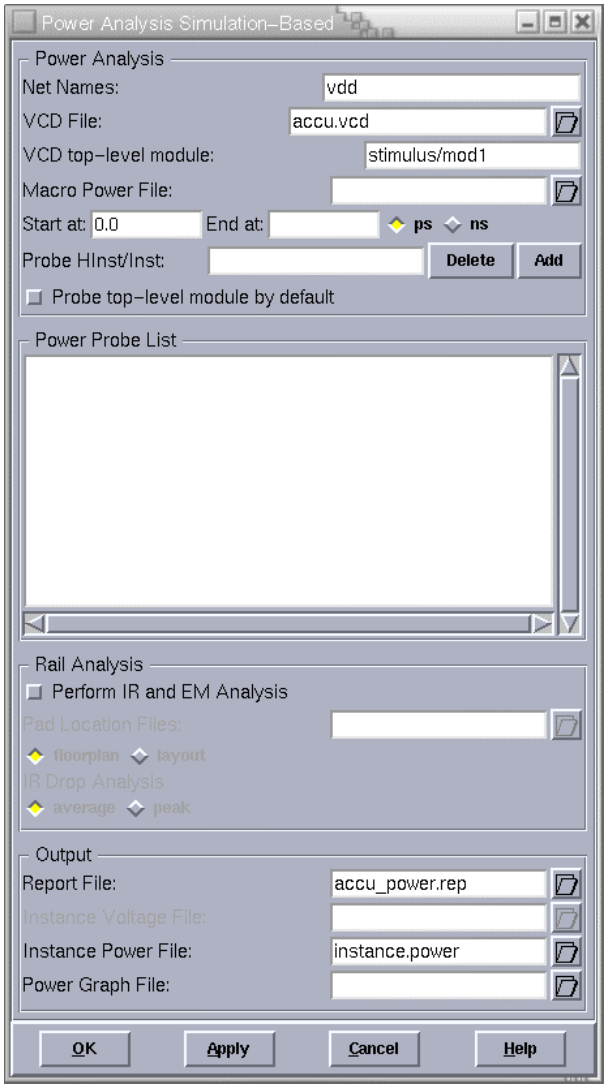
```
`timescale 1ns/10ps
`celldefine
module stimulus;

reg clk,rst;
reg [7:0] in;
wire [7:0] out;

accu mod1(in,out,clk,rst);

integer i;

 initial
     begin
        clk = 1'b0;
        forever begin #5 clk = ~clk;
                $display("At Time: %d  Accumulator Output=%d",$time,out); end
     end

initial
     begin
        #0  rst<=1;
            in<=1;
        #5  rst<=0;
     end

initial
        begin
                $dumpfile("accu.vcd");  //name of the VCD file
                $dumpvars(1,mod1);      //dump the vcd file
                $sdf_annotate("accu.sdf",mod1,,,"MAXIMUM",,); //annotate the sdf
                #2575 $finish; //finish after 65536 nSec
        end
endmodule
`endcelldefine
```

As you can see, “accu” is instantiated and the SDF file is annotated in the testbench. By running the testbench, the verilog simulator reads the SDF file and simulates “accu”, creating the VCD file that contains the switching information of the circuit based on the input test vectors. Please note that the name of the module is “`stimulus/mod1`” where “stimulus” is the name of the testbench module. Now you are ready to simulate the verilog files. At the command line run the following to simulate the files:

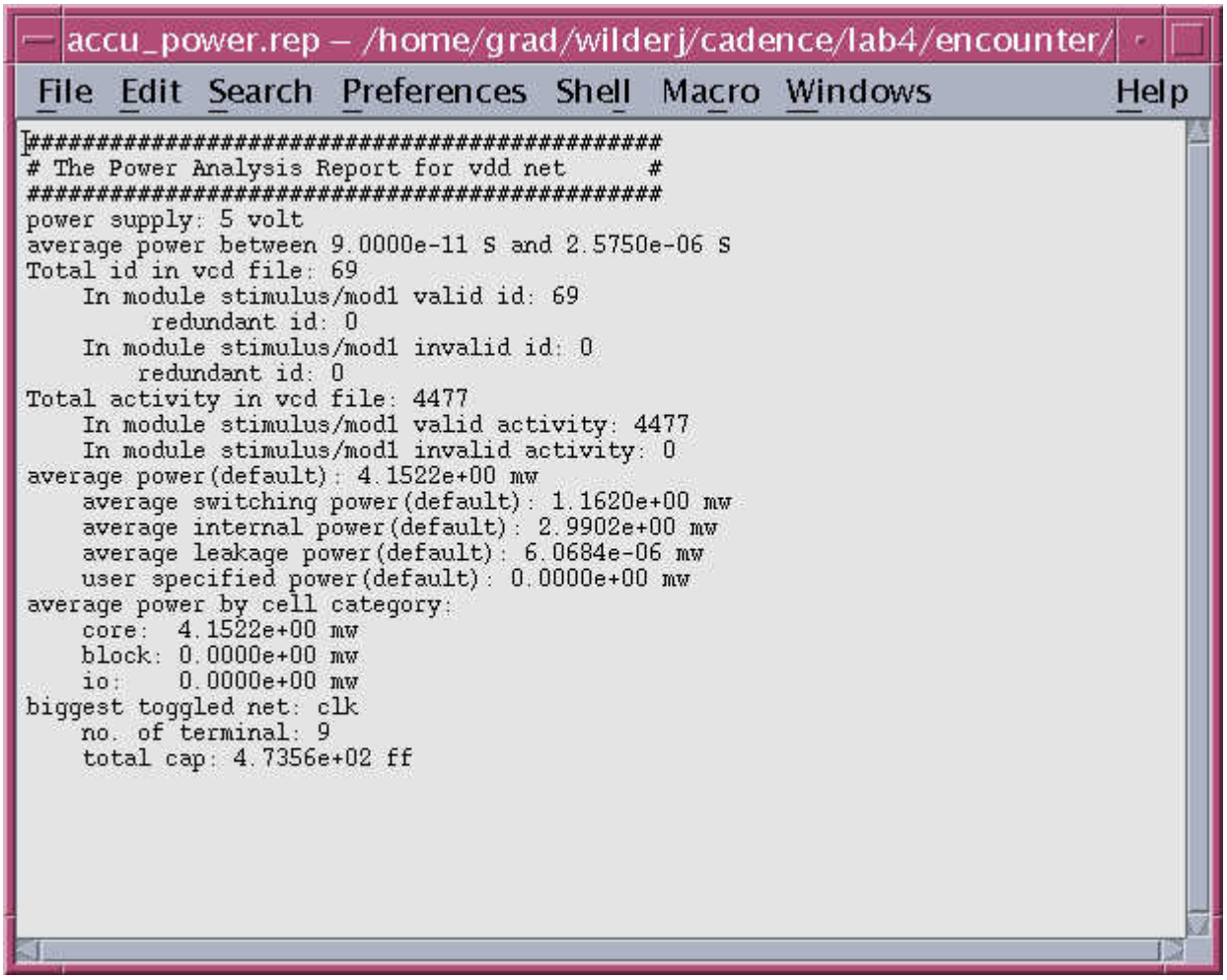
```
% ncverilog accu.v testbench.v osu05_stdcells.v +access+r
```

Running the above command creates the VCD file “accu.vcd”. Now you are ready to go back to Encounter and evaluate the power consumption of the design.

In Encounter, which had been left open in the other terminal, go to `Power > Power Analysis > Simulation-Based.` Add “Net Names”, “VCD File” name, “VCD top-level Module” and “Report File” names and change the other options as shown:



Now by hitting “OK”, Encounter will start analyzing the power of the chip and write the results in the “accu_power.rep” report file. The report file should look like this:



You used “vdd” as the net name, since you are analyzing the power consumption of the entire chip. If power analysis of other parts of the chip is needed, you should choose the net names of that section of the chip. The above report file shows output information of the power analysis. Please take a while and read the file carefully. In the file there are three different power values:

Average switching power (default): 1.162 mW
Average internal power (default): 2.9902 mW
Average leakage power (default): 6.0684e-6 mW

The first power indicates the power consumption of the chip due to the clock switching. The second power shows the cell’s internal power consumption (due to the “short-circuit” case mentioned in class), and the third power is the leakage power consumption of the circuit caused mainly by low threshold voltages. The average internal power dissipation has the most

contribution in the chip's power consumption because the circuit is dominated by flip-flops. The next line shows the total power consumption of the core, which is 4.1522 mW.

For more information on power analysis within Encounter, please open the file powerG.html in a browser window, which is located in /apps/cadence2005/SOC41/doc/encounter/.