Introduction to CMOS VLSI Design (E158)

Project 1

Harris

The goal of this project is to design a 12-bit adder with the lowest energy × delay product.

Block Specification

Function:

```
module adder(input          clk,
             input  [11:0] a, b,
             output [11:0] y);

  assign y = a + b;
endmodule
```

Inputs: each driven by two buf_1x cells from muddlib, changing concurrently with clk
Outputs: each must drive an inv_4x cell from muddlib
Objective: minimize the power × delay product of the adder (including the power drawn from V_{DD} by the input buf_1x cells and the power consumed by the inv_4x output loads)

Simulation Environment

Simulate the chip using the AMI 0.5 μm process with $\lambda = 0.3$ μm. Assume an on-chip temperature of 70 ˚C. You may choose one or more DC supply voltages between 1.2 and 5 V. You may also choose the clock period.

Several files are included in `/courses/e158/11/proj1` to help

adder_ripple:	library containing a 12-bit ripple carry adder
addertest.v:	Verilog test bench
adder.tv:	limited set of testvectors for Verilog testbench
testbench.sp:	HSPICE test bench with code to measure delay and energy
tc.sp	desired cycle time (isolated from testbench for ease of changing)
stim.vec:	digital vector file with limited set of testvectors for SPICE testbench
cycletime.pl:	a Perl script to repeatedly run HSPICE to find the minimum delay

Read through each of these files so you understand how they work. It is recommended that you work from adder_ripple and replace the ripple carry adder with something better. You are free to use cells from muddlib or to invent new cells.

Run your Verilog simulation in the same way that you did in Lab 4, using addertest.sv as the testbench.

HSPICE Simulation

HSPICE simulation involves generating a netlist from your schematic, modifying the testbench to point to your schematic, setting your target cycle time, setting your vectors, and running the simulator. You may optionally use a Perl script to automatically determine the minimum cycle time at which the circuit operates correctly on the vectors provided.

Create a new directory for your simulation (e.g. ~/proj1). Copy all the SPICE-related files from the course proj1 directory.

The first step is to generate a netlist of your schematic. A good plan is to start with the adder_ripple schematic provided so that you can test that everything works. In the schematic editor, choose Tools • Analog Environment. An Analog Design Environment window will open. Choose Setup • Simulator. Set the simulator to HSPICE (not hspiceD). Choose a project directory for the netlist; ~/cadence/simulation is fine. Then choose Simulation • Netlist • Recreate. Look at the netlist in the window that opens and check that it looks reasonable.

Next, go into the simulation directory. Read through testbench.sp and see how it works. Check that it points to the correct directory where your netlist was created. Look at stim.vec and see the default vectors. The first cycle is potentially for reset and is not used. The expected output is checked one cycle after the inputs are applied. XXX indicates don't care for an output. Edit tc.sp and change the cycle time to something conservative, such as 20 ns.

Run the simulation by typing

```
hspice testbench.sp > testbench.lis
```

The results will appear in the .lis file. The measurement results are also summarized in the .mt0 file. The simulation waveforms are in the .tr0 file where you can inspect them with Spice Explorer (sx). The .err file indicates whether there were any discrepancies with the expected outputs in the .vec file. If the file is empty, there were no errors.

Shorten tc.sp and rerun the simulation. Check if there are any errors in testbench.err. Repeat this process until you find the minimum cycle time at which the ripple carry adder works. Check the measured energy and compute the energy-delay product.

Modify the stim.vec file to include your actual test vectors. Check that the ripple carry adder still works (at a sufficiently slow cycle time) to verify that your vectors are correct.

The repeated simulations to find the cycle time get tedious. In the project directory, you can find a short Perl script called cycletime.pl that repeatedly runs the design at different cycle times, looking for errors. It reports the minimum cycle time and the EDP. Invoke it by running

```
./cycletime.pl
```

Now, netlist your improved adder design. Find its EDP. Tune your schematic and try to improve the results. Manual simulations again are tedious, so you may find that using SPICE's SWEEP capabilities or enhancing cycletime.pl to automatically tune other variables is preferable.

Deliverables

There are two deliverables and a design review in this project.

Milestone A: Schematics

Turn in a legible schematic of your adder. If you use any cells that are not part of muddlib, turn in schematics for those cells as well.

The testvectors provided are woefully inadequate to convince a skeptical engineer that the RTL is correct. Write a testplan outlining a more comprehensive set of vectors sufficient to demonstrate that the adder is likely to work, and turn in your new adder.tv along with the testplan.

Report whether the transistor-level netlist simulates in the addertest testbench for your new set of vectors with no errors.

Milestone B: Adder Optimization

Try to make your adder have the best possible energy × delay product. You are free to modify your schematics if you wish to improve a design; if so, turn in the modified designs.

Turn in SPICE simulations demonstrating the maximum operating frequency of your design, the energy at that frequency, and the energy × delay product. Use the SPICE testbench. Be careful that you choose your frequency conservatively enough that the adder will work correctly on any set of test vectors. You may wish to add test vectors to boost your confidence.

On the last day of the project, you will be given a new mystery set of test vectors. Once you have viewed these vectors, you may make no further changes to your design (including voltage or frequency). Rerun your simulations on the new vectors without changing anything. Does your design work correctly? What is its energy and energy × delay product on the new vectors?

Design Review

The adder design review will be held in class.

Complete PowerPoint slides for your adder in a template to be distributed, and email your slide to the instructor by 8 am on the day of the review. The slides will include a block diagram explaining your architecture, a Cadence schematic, and the results: supply voltage(s), energy (pJ), power (mW), operating frequency (GHz), and energy × delay product (pJ-ns, on the original and second set of test vectors).

In class, we will group the adders by their architecture and evaluate which ones look best. You should be prepared to field questions about your design. We will take a critical look at the most promising designs. This may involve opening the schematic and/or simulation in Cadence during class. Come ready to ask probing questions about the designs that may highlight implausible assumptions or erroneous results.

One caveat is that wire capacitance is not considered in this design project, making the results somewhat optimistic.

Grading

Your grade will be based on the following factors:

30%: Readable schematics simulating on schedule
5%: Convincing test vectors
30%: Claimed energy × delay product (relative to other designs in class)
20%: Adder functions at claimed delay on mystery test vectors
10%: Design review slides with readable block diagram, schematic and all data
5%: Contributions to design reviews

Hints

The SWEEP function in SPICE lets you change parameters (such as transistor sizes or supply voltages) and produce results for each parameter value. This is helpful for tuning.

There are several sections that you could read in the textbook to shave hours of experimentation off of your project. The project has been intentionally scheduled before all of these sections are reached in the book, to give you incentive to selectively read ahead and practice just-in-time learning. An ounce of analysis is worth a sleepless night of sweeps.