Harris

Introduction to CMOS VLSI Design (E158)

Project 1

Spring 2008

The goal of this project is to design a 20-bit adder with the lowest energy × delay product.

Block Specification

Function:

```
module adder20(input         clk,
               input  [19:0] a, b],
               output [19:0] y);

  assign y = a + b;
endmodule
```

Inputs: each driven by buf1x cells from muddlib, changing concurrently with clk

Outputs: each must drive an inv4x cell from muddlib

Objective: minimize the power × delay product of the adder (including the power drawn from V_{DD} by the input inv1x cells and the power consumed by the inv4x output loads)

Simulation Environment

Simulate the chip using the AMI 0.5 μm process with λ = 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 the Charlie folder under project1 to help you:

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
null.sp:	empty header file to keep Electric happy
mosisami600:	SPICE models for AMI 0.5 μm process
duration.sp:	file included by test bench to set simulation duration
stim.vec:	digital vector file with limited set of testvectors for SPICE testbench
adder.jelib:	sample Electric library with ripple carry adder including input and output loading

Read through each of these files so you understand how they work. It is recommended that you work from adder.jelib and replace the ripple carry adder with something better. To produce a SPICE netlist from an Electric schematic, first choose Preferences • Tools •

SPICE and "Use Header Cards from File:" null.sp. This overrides Electric's tendency to include bogus SPICE models at the start of a deck. Use the Tool • Simulation (SPICE) • Write Spice Deck… command to generate adder.spi from Electric. Look at the netlist. It should be correct except that it has an extraneous `.end` statement on the last line; delete this.

Run SPICE using hspui. Get in the habit of looking for warnings and errors in the .lis file after each run. The testbench generates a testbench.err file containing a list of the outputs that don't match expectations. The errors may occur because of a logic bug in the schematic (unlikely if it passes Verilog simulation) or because the cycle time is too short; by cranking down the cycle time and looking for errors, you can determine the maximum operating frequency.

You are free to use cells from muddlib or to invent new cells. The schematic does not produce diffusion capacitance estimates in the SPICE netlist; this is a weakness of the CAD flow that we will tolerate; commercial flows would do a better job of estimating diffusion and intra-cell wiring capacitances.

Place capacitors in your schematic to represent the lumped capacitances of all wires crossing 4 or more bits. Assume that cells are in 20 rows (one per bit of the adder), and that each row is 110 λ tall. Assume an average wire capacitance of 0.2 fF/μm.

The test fixture checks if each output has settled to the correct value by the start of the next clock cycle. Delay is measured as the clock period at which the simulations were performed without error. Not that this includes the delay of the input buffer in the cycle time. Energy is the average energy per addition (clock period). Note that this includes the energy consumed by the input buffers and output inverters.

Deliverables

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

Milestone A: RTL

Turn in commented gate-level structural RTL implementing 20 bit addition. The RTL should be at the gate level; e.g.. NAND, NOR, AOI gates. Page 49 of CMOS VLSI Design has an example of a gate-level description of a full adder. At your discretion, the gate modules may be described using `assign` statements or `tranif` statements.

The four 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 RTL simulates in the addertest testbench for your new set of vectors with no errors.

Milestone B: Schematics

Turn in readable printouts of your adder schematics. Be sure to include wire capacitance where necessary. Netlist the schematics to Verilog and report whether the transistor-level netlist simulates in the addertest testbench with no errors using your testvectors. Your schematics should match the RTL at the gate level; you are free to update the RTL if you wish.

Milestone C: Adder Optimization

Try to make your adder have the best possible energy × delay product. You are free to modify your schematics and RTL 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. If so, add them at the end of stim.vec and change the SIMCYCLES parameter in duration.vec to reflect the number of new vectors.

On the last day of the project, you will be given a new mystery set of test vectors (in duration.sp and stim.vec). 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

Give a presentation describing your adder. Your presentation should include at least one block diagram explaining your architecture (not an Electric schematic). It should conclude with your 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). You have 6 minutes to explain your design and 3 minutes to field questions from your classmates.

After each presentation, the members of the class will be asked to complete a form rating their confidence that your adder could really be built meeting your claimed energy and frequency (neglecting the capacitance of metal and poly wires within cells) and pointing out any flaws in the modeling or simulation. Grades will be influenced by your ability to convince your peers that your results are credible and on your ability to catch flaws in your peers' designs.

Grading

Your grade will be based on the following factors:

10%: Well-written RTL simulating on schedule
10%: Readable schematics simulating on schedule
30%: Claimed energy × delay product
40%: Confidence that adder could meet claimed performance
10%: Contributions to design reviews and accurate reviews of other adders

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.