EE113 – Digital Integrated Circuits

Final Project – Optimal 800MHz 4-Bit "Absolute-value Detector"

Due Friday, January 6th

Project Description

The goal of this project is to design a **4-bit "Absolute-value Detector"** with the **minimum energy and worst-case delay of 1.25 ns**. Here "delay" refers to the worst-case propagation delay and "energy" refers to total energy drawn from V_{DD} given a specified input probability distribution. You may use **gate sizing** and **supply voltage scaling** as variables. No registers (i.e. pipelining) are allowed in the design of this project. Work in a **group of 2 students**.

You will find *appendices* that explain what an "Absolute-value Detector" is and other references you might find helpful at the end of this document. **Please read them carefully.**

For a systematic approach, you can organize your work in three phases. In **phase-1**, use the design expertise you acquired in class to find the optimum architecture that best optimizes the speed-energy goal. Do a quick sketch of several feasible options and figure out the best architecture and circuit style. You may mix circuit styles if that helps. In **phase-2**, first implement the block-level schematic of your Absolute-value Detector and verify the functionality in Spectre. Then, identify the critical path and optimize sizing for minimum delay. In the critical path evaluation, you need to determine not only the gates along the path, but also the input operands that cause worst-case delay between input and output bits. In **phase-3**, refine your rough layout sketch from phase-1 and layout your design starting with basic building blocks. Area is defined as the smallest rectangular bounding box a design can fit in. Aspect ratio of the box (long / short side) should be less than **1.5**. Below is a more detailed explanation of the steps you need to take to ensure the success of your project.

Phase 1: Choosing Topology / Circuit Style (1 week)

a) Design the logic diagram which can implement the 4-bit Absolute-value Detector

b) Determine circuit topology that optimizes delay-energy metric (hand analysis).

c) Choose logic style for the implementation. You may use any logic family you learnt from EE113.

d) Implement the schematic view of the Absolute-value Detector in Cadence.

Phase 2: Critical Path Delay Optimization & V_{DD} Scaling (1 week)

a) Check functionality of your design in Spectre.

b) Identify input vectors that will exercise critical path, and estimate its logical effort.

c) Size the gates for minimum delay (hand analysis).

d) Verify the critical path delay in Spectre under worst-case input operands.

d) Consider changing V_{DD} (within range 0~1V) to achieve minimum energy while meeting the delay requirement.

Phase 3: Layout and Verification in Spectre (2 weeks)

a) Create layout of the design and make sure it passes DRC and LVS.

b) Extract post-layout netlist and verify critical path in Spectre.

c) Submit pre-layout and post-layout netlists. We will run LVS on your design, and run worse-case test-bench to verify reported critical-path delay.

Final Report (1/2 week)

Prepare a minimum **5-page report** representing your project. Highlight your main design decisions, explain why they are the best thing in the world, and prove that they really worked out (or did not). Please identify “three most important features of your design” and “three things you would do different or improve if you are given another chance” in the final report.

Constraints (PLEASE READ CAREFULLY!)

a) Supply voltage:

Find optimal V_{DD} (0~1V) that meets the delay (1.25 ns) and minimizes energy.

b) Implementation choices:

Use only static logic (CMOS, pass-transistor logic).

c) Input operands:

Input signal is a 4-bit number represented in 2's complement format. Threshold value (3-bit binary) is fixed and will be given to you in the final test bench. Assume each bit of the input has equal probability of being 0 or 1 and the bits are mutually independent.

d) Loading conditions:

i. The input capacitance of all inputs (A0-A4) is less than or equal to 2 unit sized inverters (see below for the definition of unit-sized inverter). For simulation purposes, the inputs to your adder are driven by a unit sized buffer (chain of two unit sized inverters). The delay is measured as the delay after the input driver (2 inverters) to before the load (32 times unit sized inverters). **Test-circuit will be provided. (You also need to test your circuit before the test bench is provided to ensure full functionality).**

ii. The output bit is loaded with C_L = 32 unit sized inverters. This load will be implemented with inverters (test bench coming soon).

iii. Unit sized inverter is W_p = 720nm, W_n = 360nm, L_p = L_n = 180nm (drawn L).

d) Loading conditions:

i. You can use up to **5 metal layers**.

ii. Aspect ratio of the bounding box (long / short side) should be **less than 1.5**.

Appendix A – Absolute-value Detector

Spike-sorting algorithms have gained a tremendous interest in the research community with the recent advancements in neural signal acquisition systems. For your EE115C project this quarter, you are to design one of the commonly used spike-detection algorithms, named absolute-value detection.

Figure 1 shows the basic diagram for an Absolute-value detector that needs to be designed for your project. The inputs (shown in blue) are given to you (See the constraints for more detail). The Absolute-value detector (shown in black) is to be designed and implemented by you.

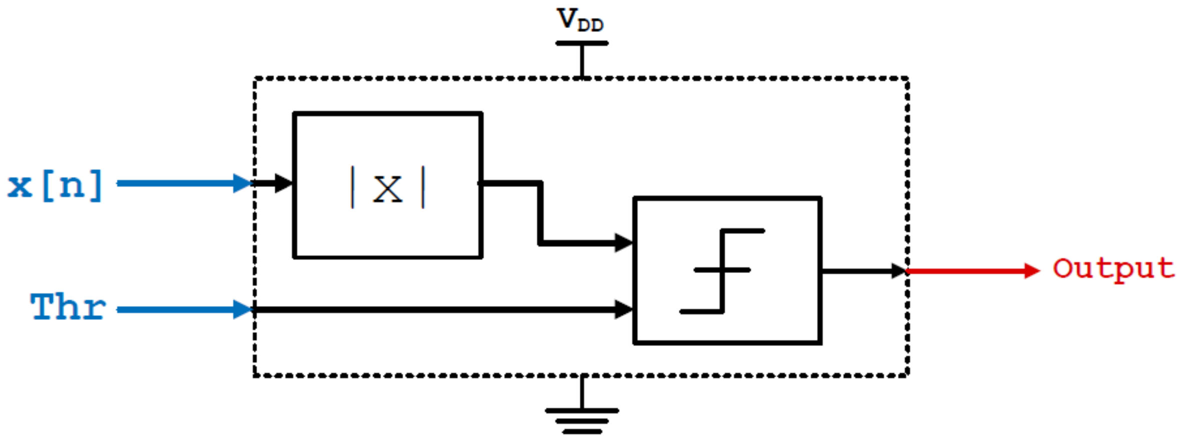

Figure 1 – Absolute Value Detector

As shown above there are two main components to the absolute-value detection.
(1) finding the magnitude (absolute value) of your neural signal (x[n]) and
(2) comparing the magnitude to the given threshold value (Thr).

If the magnitude of your signal is greater than the threshold output should display a “1” (high logic value), otherwise the output should be a “0” (low logic value).

NOTE:
(1) Your input signal, x[n], is given in a 2’s complement representation. You should already be familiar with 2’s complement representation of binary numbers. If you are not, please re-visit “Digital Fundamentals” by Thomas L. Floyd (Chapter 2).

(2) **Your input values can range from -7 to +7 and will be given to you in 2’s complement format.** Although -8 can also be represented in 2’s complement using 4 bits, you can assume that this will never be given as an input. This way, your magnitude will always be 3-bits which is compared to the given 3-bit threshold value in your comparator.

(3) As you noticed you need an adder for this block. Adder designs will be covered in class and more information can be found in Lectures 10&12. **It is important**, however, to note that you always add a 1 to your numbers. This fact can allow you to **significantly reduce the complexity of your design** and result in a much **more optimized logic**.