Digital Integrated Circuits

Class Project

Due 2 hours before lecture 7

Optimal 4-Bit "Absolute-value Detector"

Problem Description

The goal of this project is to design a **4-bit "Absolute-value Detector"** with the **minimum energy for delay that is 50% (1.5x) longer than the minimum delay**. Here "delay" refers to the worst-case propagation delay and "energy" refers to total energy drawn from V_{DD} for a given input probability distribution. You may use **gate sizing** and **supply voltage scaling** as variables. No registers (i.e. pipelining) are allowed. Assume gamma ($C_{parasitic}$ / C_{gate}) = 1 for delay modeling.

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

For a systematic approach, you can organize your work in two phases.

- In **phase-1**, use the design expertise you acquired in class to find an 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. 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.

Below is a more detailed explanation of the two phases.

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

a) Determine circuit topology that optimizes delay-energy metric.
b) Choose logic style to implement. You may mix logic families (e.g. CMOS, PTL).

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

a) Identify input vectors that will exercise critical path. Size the gates for minimum delay.
b) Consider changing gate sizing *and* V_{DD} (within range 0~1V) to achieve minimum energy while meeting the delay requirement. Assume that delay is proportional to V_{DD} as V_{DD} / $(V_{DD} - V_T)^2$ in order to find optimal V_{DD}. Assume V_T = 0.2V.

Prepare Final Report **(1/2 week)**

Prepare a **10-slide report** (template provided) representing your effort. Be crisp: 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). Report energy at minimum delay, and energy at 1.5x delay. More details are in the template.

Design Constraints (READ CAREFULLY!)

a) Supply voltage: find optimal V_{DD} (0~1V) that, in combination with gate sizing, meets the delay (1.5 of min delay) and minimizes energy.

b) Implementation choices:

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

c) Input operands:

i. Input signal is a 4-bit number represented in 2's complement format. Threshold value (3-bit binary) is fixed and you can assume a mid-range value for this Threshold.

ii. 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 (A_0-A_3) is less than or equal to 2 unit sized inverters (see below for the definition of unit-sized inverter). For design 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).

ii. The output bit is loaded with C_L = 32 unit-sized inverters.

iii. Unit-sized inverter is Wp = 650nm, Wn = 430nm, Lp = Ln = 100nm (drawn L).

Appendix A – Absolute-value Detector

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

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

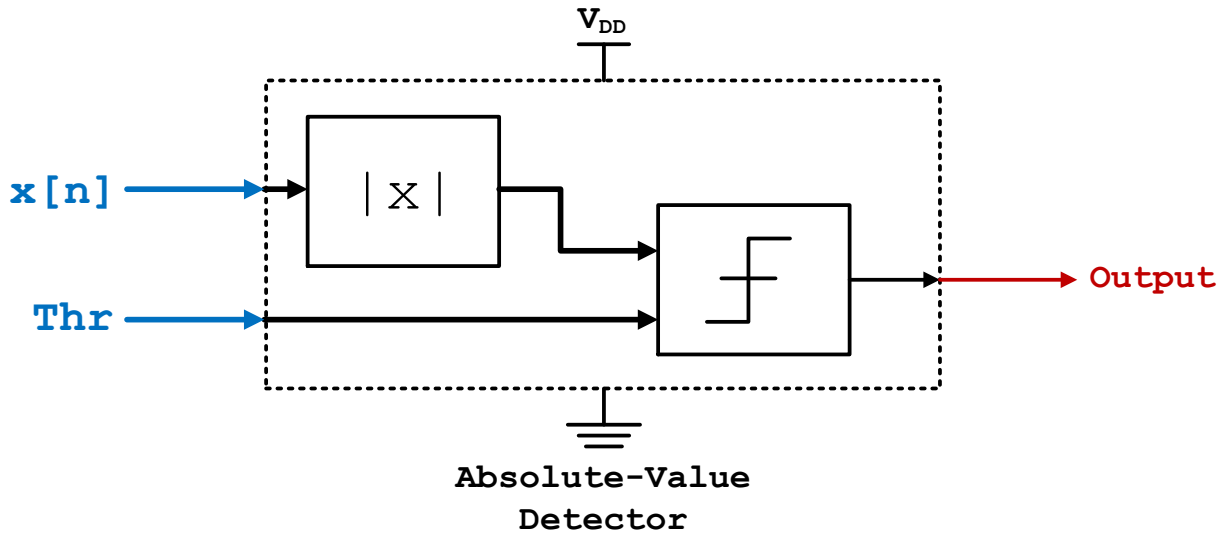

Figure A-1

As shown above there are two main components to the absolute-value detection.

(i) finding the magnitude (absolute value) of your neural signal (`x[n]`) and
(ii) 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).

Appendices B and C provide some background and references for the design of Absolute-value and comparator blocks, respectively.

Appendix B – Magnitude of a 2's Complement Number

Your input signal, `x[n]`, is given in a 2's complement representation. You should be familiar with 2's complement representation of binary numbers. If you are not, you are strongly advised to review your ECEM16 course material or refer to any of the following references:

1. "Digital Design" by M. Morris Mano.
2. "Digital Design: A Systems Approach" by William J. Dally and R. Curtis Harting.

Your input values can range from −7 to +7 and will be given in 2's complement format.

- Although −8 can be represented in 2's complement using 4 bits, you can assume that this will never occur at input. This way, your magnitude will always be 3 bits, which is compared to the given 3-bit threshold value in your comparator.

Few reminders about 2's complement numbers:

- If the most significant bit is a "`0`", the number is positive → Magnitude is the same as the number given to you.
    - Example 1: `+7` represented in 4-bit 2's complement format is `0111`.
      It's 3-bit magnitude is `111`.
    - Example 2: `+5` represented in 4-bit 2's complement format is `0101`.
      It's 3-bit magnitude is `101`.

- If the most significant bit is a "`1`", the number is negative → Magnitude is found by flipping (inverting) all bits and adding a `1`.
    - Example 3: `−7` represented in 4-bit 2's complement format is `1001`.
      To find its magnitude we do the following:

```
  1001      (−7)

  0110      (bits are flipped)
+ 0001      (add 1)
  ------
  0111      (Magnitude – You can ignore the 4th bit)
```

      It's 3-bit magnitude is `111`.
    - Example 4: `−5` represented in 4-bit 2's complement format is `1011`.
      To find its magnitude we do the following:

```
  1011      (−5)

  0100      (bits are flipped)
+ 0001      (add 1)
  ------
  0101      (Magnitude – You can ignore the 4th bit)
```

      It's 3-bit magnitude is `101`.

As you noticed, you need an adder for this block. Adder designs will be covered in class and more information can be found in Chapter 11.3 of the textbook. **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**.

Appendix C – Comparator

One you have found the 3-bit magnitude of your signal you will need to compare its value with the given threshold. 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).

The strategy here is compare the most significant bits of the two numbers. If one is greater than the other, we know that number is greater. If they are equal we will move to the second most significant bit, and so on.

For your reference a 4-bit Magnitude Comparator (you need 3-bit Magnitude Comparator) is shown in Figure A-2.

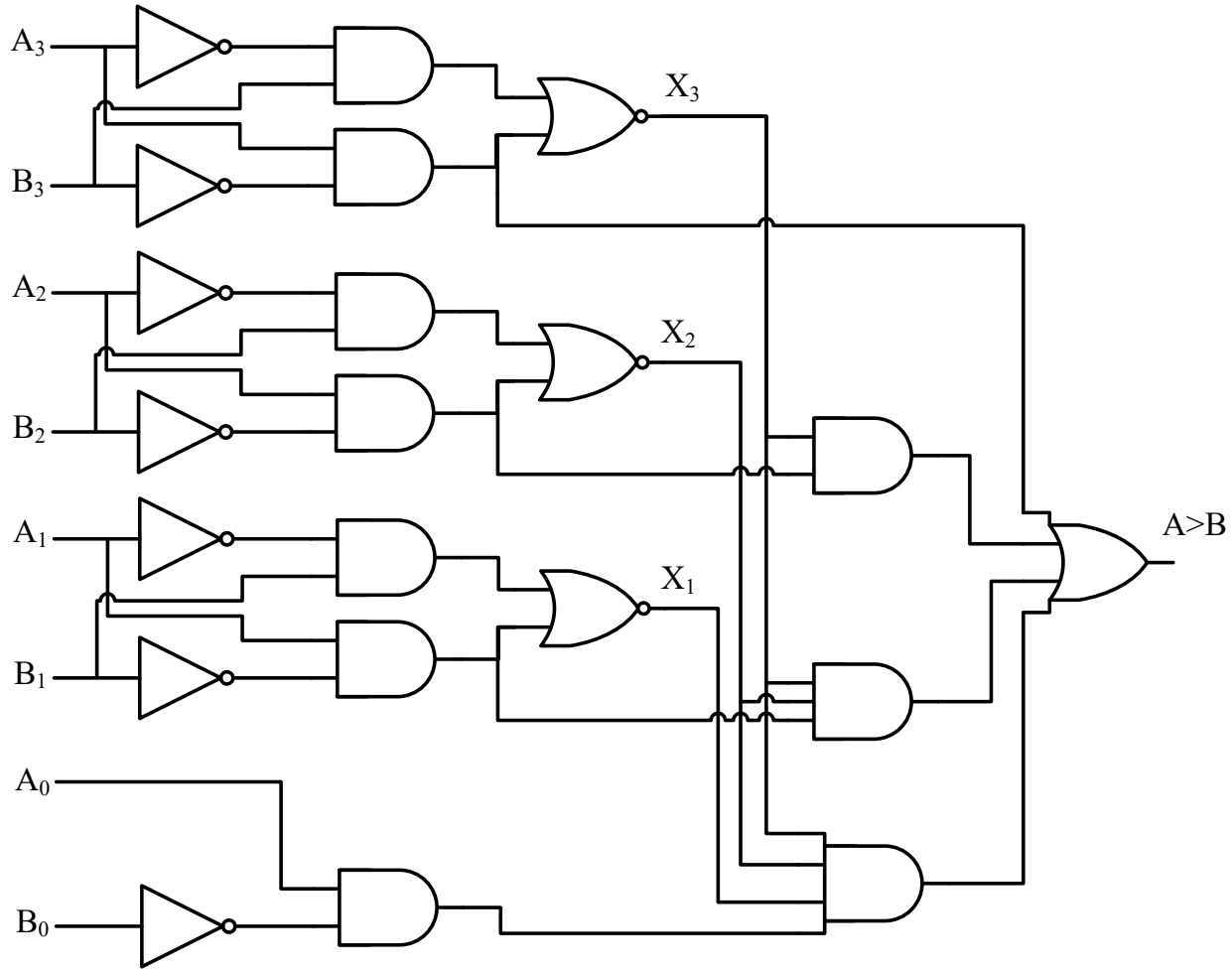

Figure A-2

Here: $(A > B) = A_3\overline{B_3} + X_3A_2\overline{B_2} + X_3X_2A_1\overline{B_1} + X_3X_2X_1A_0\overline{B_0}$, where $X_i = A_iB_i + \overline{A_i}\,\overline{B_i}$

HAVE FUN!