ECE115C – Digital Electronic Circuits

Class Project - FINAL

Due Thursday, March 13th @ 5pm
(Submit on Gradescope)

Energy-efficient 4-Bit Absolute-value Detector

Problem Description

The goal of this project is to design a **4-bit Absolute-value Detector** with **minimal energy consumption while ensuring the delay is 40% (1.4x) longer than the minimum delay baseline.** Here, "delay" refers to practical worst-case delay, and "energy" refers to total energy drawn from V_{DD} for a given input profile.

Your design space includes:

- **Logic architecture:** Boolean logic, arithmetic units, or a combination of both.
- **Logic style:** CMOS, transmission gate-based, or a combination of both.

Once you finalize the logic architecture and logic style, you will optimize gate sizing (using the logical effort delay model) and supply voltage scaling to minimize energy. No registers are allowed; the design must be purely combinational.

Team Formation:

- Work in a group of **three students**.
- If you need help finding a partner, please notify the instructor.
- The **project sign-up sheet** is available here.

Structured Approach (Phases):

Phase 1: Logic Architecture & Gate-Level Design
Using the design methods discussed in class, identify an optimal architecture and logic style that balances speed and energy efficiency. (We will discuss adders in Week 6)

- **Architecture-level considerations:**
  - Truth table-based logic minimization,
  - Use of arithmetic functions (e.g. adders/subtractors),
  - A combination of Boolean and arithmetic-based methods.
- **Gate-level considerations:**
  - Static CMOS,
  - Transmission gate-based logic,
  - A hybrid of both.
- **Task:**
  - Sketch multiple feasible designs and select the best combination of architecture and circuit style,
  - Size the design for **minimum delay** using the logical effort delay model,
  - Keep the voltage at 1V at this stage,
  - Assume $\gamma = 0.7$ in the analytical delay model.

Phase 2: Logic Implementation & Simulation (LTSpice)

- Implement the phase-1 design in LTSpice to verify logic functionality and measure delay,
- Conduct critical-path delay analysis to determine:
  - The gates along the critical path,
  - The input pattern that includes the worst-case delay between input and output,
- Establish the minimum-delay baseline via simulation (Phase-2 simulation delay model).

Phase 3: Optimization via Gate Sizing & Supply Voltage Scaling

- Develop **logical-effort based sizing** and **voltage-dependent delay and energy models**,
- Apply gate sizing and supply voltage optimization to **minimize energy at 1.4x delay**,
- This results in the **Phase-3 analytical delay model**,
- Verify the analytical model with the simulation model, report discrepancies, and fine-tune the supply voltage as needed,
- Ensure that the final simulated delay reaches 1.4x of the minimum-delay simulation model from Phase 2.

Delay and Energy Metrics:

Analytical model is your logical effort-based delay and energy model.

- *Analytical delay* is **normalized to FO4 inverter delay** (i.e. FO4 = 4 + γ).
- *Analytical energy* is **normalized to the energy required to charge unit-inverter gate capacitance at V_{DD} = 1 V** (i.e. $E_{Cg1} = C_{in}^{INV1}$) for a **W_p/W_n = 300nm/200nm** and **L = 100nm** unit inverter.
- One unit of diffusion-capacitance energy **$E_{d1} = \gamma \cdot E_{Cg1}$**.
- These unit energy costs should be scaled according to **gate size and type, and by corresponding switching activity**.

Simulation data is obtained by LTSpice simulation.

- *Simulation delay* is expressed in **picoseconds**.
- Apply worst-case delay analysis to your circuit and determine:
  - Logic states for all inputs,
  - Latest-arriving logic input that initiates "practical" worst-case delay,
- *Simulation energy* is expressed in fJ (10^{-15}J) for the provided testbench.

Suggested Timeline (Start on Time and Finish on Time)

- **Phase 1:** 1 week (start on **Thu, 2/13**)
- **Phase 2:** 1 week (start on **Thu, 2/20**)
- **Phase 3:** 1 week (start on **Thu 2/27**)
- **Make report and double-check everything:** 1 week (start on **Thu, 3/6**)
- **Upload your report to Gradescope:** by **5pm on Thu, 3/13**

Report Guidelines:

Prepare a **concise written report** representing your effort. The report should be at most **two pages** and structured as follows:

- **Page 1:**
  - Include summary tables provided below,
  - Provide a top-level diagram of your circuit.
- **Page 2:**
  - Highlight your main design decisions and rationale,
  - Discuss trade-offs considered in achieving the design goals.

Ensure that all required elements are clearly presented and well-organized.

Phase	1	2	3
Analytical $D_{critical\text{-}path}$ [FO4]	Baseline	N/A	Optimal
Simulation $t_{critical\text{-}path}$ [ps]	N/A	Baseline	Optimal
Analytical Energy [E_{Cg1}]	Baseline	N/A	Optimal
Simulation Energy [fJ]	N/A	Baseline	Optimal

Total energy reduction [%]	Contribution from sizing	Contribution from V_{DD}
Analytical model [%]		
Simulation model [%]		
Total delay increase [%]	**Contribution from sizing**	**Contribution from V_{DD}**
Analytical model [%]		
Simulation model [%]		

Design Constraints (READ CAREFULLY!)

a) Supply Voltage and Gate Size Bounds:

- **Supply voltage, V_{DD}:** Upper bound = 1V, lower bound dictated by a 1.4x delay increase,
- **Size:** lower bound dictated by drive resistance of unit inverter, upper bound dictated by C_L,

b) Design Space Exploration:

- **Architecture Choices:**
  - Boolean logic,
  - Arithmetic-based (adder/subtractor),
  - A combination of both.
- **Logic Gate Choices:**
  - Static CMOS,
  - Transmission gate-based,
  - A hybrid of the two.

c) Logic Inputs:

- Input signal is a **4-bit number** represented in **2's complement format**.
- A fixed **3-bit unsigned binary threshold value** (provided in the final testbench) must be handled by your design for any given threshold.
- Assume each bit of the input has **equal probability** of being 0 or 1, and that the bits are **mutually independent**.

d) Loading Conditions (Input, Output):

- **Input Capacitance Constraints:**
  - The input capacitance of all inputs (A_0-A_3) must be $\leq$ 2 unit-sized inverters,
  - Inputs are driven by a **unit sized buffer** (chain of two unit-sized inverters),
  - Delay is measured from after the input driver (2 inverters) to before the load ($32C_{g1}$),
  - **A test-circuit will be provided** (ensure full functionality before using the testbench).
- **Output Loading Conditions:**
  - Each output bit is loaded with $C_L = 32\ C_{g1}$,
  - The testbench will implement this using inverters (testbench coming soon).
- Unit-Sized Inverter: W_p = 300nm, W_n = 200nm, $L_p = L_n$ = 100nm (drawn L),
  - This means $\beta = W_p/W_n = 1.5$ for static CMOS logic.
- Unit-Sized Transmission Gate: $W_p = W_n$ = 200nm, $L_p = L_n$ = 100nm.

e) Design Goals:

- **Baseline design:** optimize gate size (at V_{DD} = 1V) for minimum delay, D_{min},
- **Final design:** optimize gate size jointly with V_{DD} for minimum energy at $1.4D_{min}$.

Attached appendices provide background information. Please read them carefully.

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 ECE 115C project, you are to design one of the commonly used spike-detection algorithms, named Absolute-value detection.

Figure A 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.

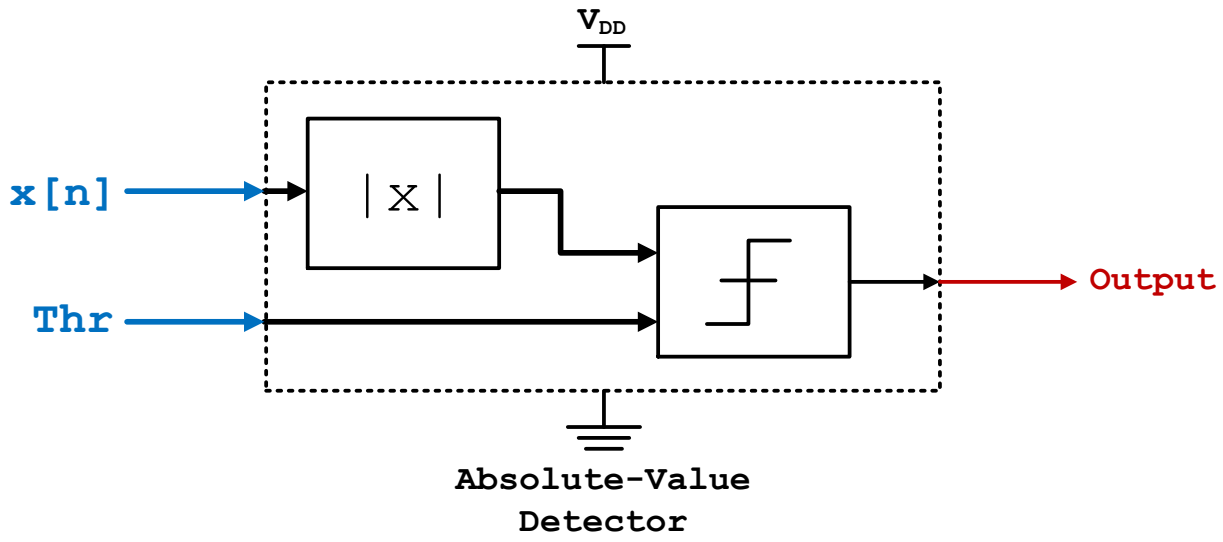

Figure A

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

Once 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 C.

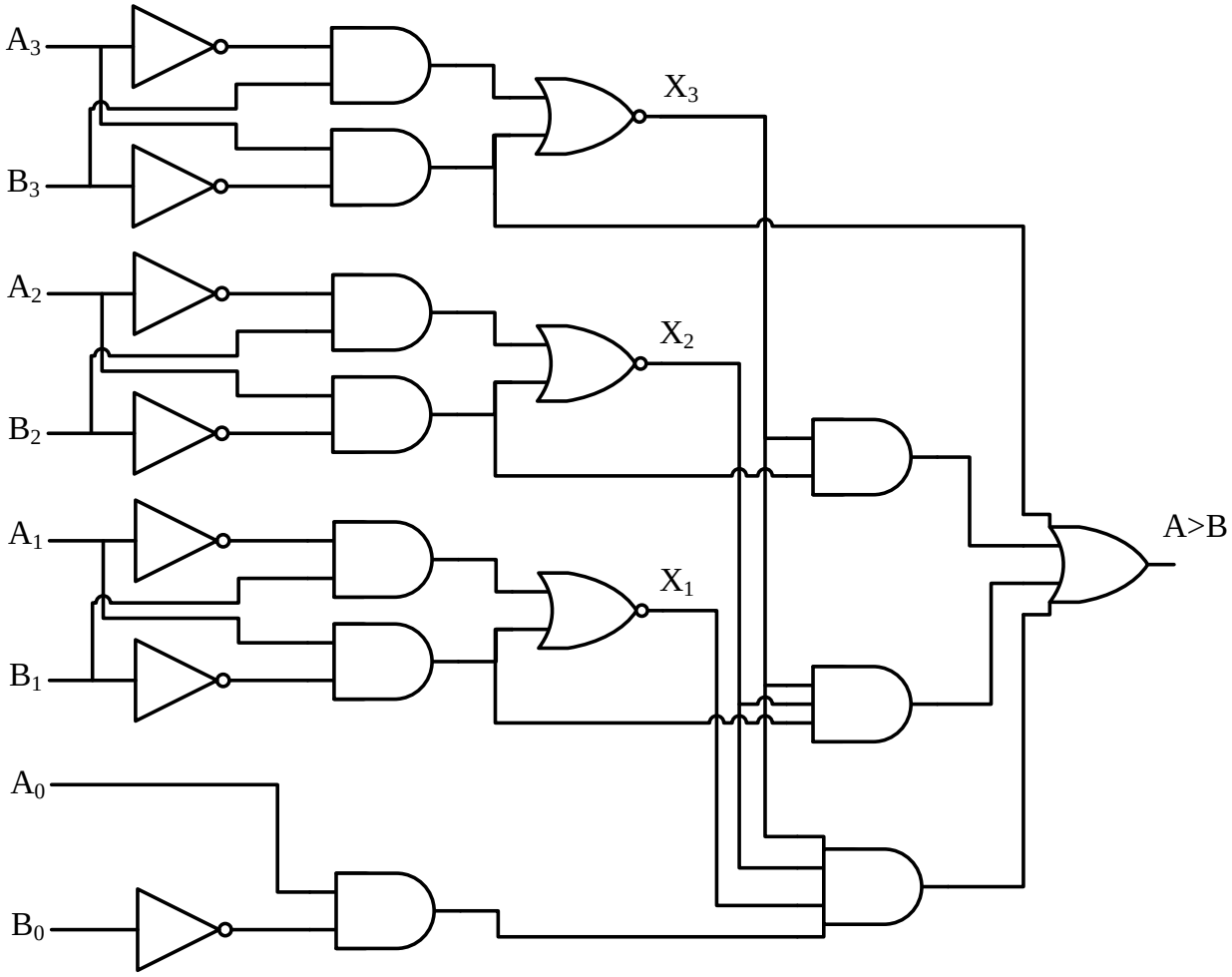

Figure C

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!