Introduction

As CMOS continues the march of scaling, its impact on system reliability becomes increasingly severe. Error correction is one of the techniques proven effective in combating reliability degradation. In this project we will implement an error correction scheme to detect and correct defects in a 4Mbit SRAM array.

The particular error correction scheme we will use is Hamming Code, named after its inventor Richard Hamming (1915~1998). Its design process can be divided to three steps.

**Step 1:** For \( n \) bits of original data, an extra \( k \) bits of parity need to be inserted to satisfy the following relation

\[
2^k \geq n + k + 1
\]

Positions of parity bits in the elongated word are \( 1^{\text{st}}, 2^{\text{nd}}, 4^{\text{th}}, \ldots, 2^{(k-1)} \). All other positions in the word are for the original data.

**Step 2:** Value of each parity bit is calculated based on parity with selected bits in the \((n+k)\) bit word

\[
P_x \oplus B_{x,1} \oplus B_{x,2} \oplus \ldots = 0
\]

where \( P_x \) is the parity bit and the \( B_x \)'s stand for the data bits selected. Selection of data bits depends on the position of \( P_x \), obeying following skip-and-check rule:

- For \( P_1 \) (the parity bit at the 1\(^{\text{st}}\) position of the word), skip 0 bits, check 1 bit, skip 1 bit, check 1 bit, …
- For \( P_2 \) (the parity bit at the 2\(^{\text{nd}}\) position of the word), skip 1 bit, check 2 bits, skip 2 bits, check 2 bits, …
- For \( P_4 \) (the parity bit at the 4\(^{\text{th}}\) position of the word), skip 3 bit, check 4 bits, skip 4 bits, check 4 bits, …
- For \( P_8 \) (the parity bit at the 8\(^{\text{th}}\) position of the word), skip 7 bit, check 8 bits, skip 8 bits, check 8 bits, …
- In general, for \( P_x \) (the parity bit at the \( x^{\text{th}} \) position), skip \((x-1)\) bit, check \( x \) bits, skip \( x \) bits, check \( x \) bits, …

**Step 3:** Decoding of a Hamming encoded word follows the same parity check equation and the same bit selection process described in step 2

\[
A_0 = P_1 \oplus B_{1,1} \oplus B_{1,2} \oplus \ldots
\]

\[
A_i = P_{2^i} \oplus B_{2^i,1} \oplus B_{2^i,2} \oplus \ldots
\]
\[ A_2 = P_4 \oplus B_{4,1} \oplus B_{4,2} \oplus \ldots \]
\[ A_3 = P_8 \oplus B_{8,1} \oplus B_{8,2} \oplus \ldots \]
\[ A_{\log_2(x)} = P_x \oplus B_{x,1} \oplus B_{x,2} \oplus \ldots \]

For single-bit errors, the address word \( A_{\log_2(x)} \ldots A_3A_2A_1A_0 \) identifies the location of the error bit.

**Example**

Follow example illustrates the use of Hamming Code for a 4-bit data.

**Step 1:** Given that \( n = 4 \), to satisfy
\[ 2^k \geq 4 + k + 1 = 5 + k \]
k = 3. Positions of parity bits in the elongated word are 1st, 2nd, and 4th. Positions of original data are 3rd, 5th, 6th, and 7th. Bit sequence in the new word is \( P_1P_2B_3P_4B_5B_6B_7 \).

**Step 2:** To calculate the value for \( P_1 \), it skips 0 bits, checks 1 bit (\( P_1 \)), skips 1 bit (\( P_2 \)), checks 1 bit (\( B_3 \)), skips 1 bit (\( P_3 \)), checks 1 bit (\( B_5 \)), skips 1 bit (\( B_6 \)), and checks 1 bit (\( B_7 \)). So the parity check equation for \( P_1 \) is
\[ P_1 \oplus B_3 \oplus B_5 \oplus B_7 = 0 \]
If the original data \( B_3B_5B_6B_7 = 1100 \), then \( P_1 = 0 \).

Similarly, \( P_2 \) skips 1 bit, checks \( P_2 \) and \( B_3 \), skips 2 more bits, and checks \( B_6 \) and \( B_7 \).
\[ P_2 \oplus B_3 \oplus B_6 \oplus B_7 = 0 \]
For \( B_3B_5B_6B_7 = 1100 \), then \( P_2 = 1 \).

Finally, \( P_4 \) skips 3 bits, and checks \( P_4 \), \( B_5 \), \( B_6 \) and \( B_7 \).
\[ P_4 \oplus B_5 \oplus B_6 \oplus B_7 = 0 \]
For \( B_3B_5B_6B_7 = 1100 \), then \( P_4 = 1 \).

The final value for the new 7-bit word is 0111100.

**Step 3:** Assume that a soft error occurs and one bit of the above word is flipped: 0111100 \( \rightarrow \) 0101100. A Hamming decoder can compute the location of this error bit as follows.
\[ A_0 = P_1 \oplus B_3 \oplus B_5 \oplus B_7 = 0 \oplus 0 \oplus 1 \oplus 0 = 1 \] (Parity check fails.)
\[ A_1 = P_2 \oplus B_3 \oplus B_6 \oplus B_7 = 1 \oplus 0 \oplus 0 \oplus 0 = 1 \] (Parity check fails.)
\[ A_2 = P_4 \oplus B_5 \oplus B_6 \oplus B_7 = 1 \oplus 1 \oplus 0 \oplus 0 = 0 \] (Parity check passes.)
\[ A_2A_1A_0 = 011 = 3 \], pointing to the 3rd bit as the source of error.

**SRAM Array**

The 4Mbit memory is organized in a 1,024 by 4,096 array. The column decoder has 6 bits, which yields the word length of 64 bits (=4096/2^6). Block diagram of the memory architecture is shown in Fig. 1.
Note that the SRAM in Fig. 1 does not include error correction codes, which will increase word length from 64 bit to $(64+k)$ bits, where $k$ is the number of parity bits. Refer to Fig. 2 below for details. First, output of the sense amplifiers needs to be Hamming decoded. When error is detected (and assume single-bit error throughout this project), the Hamming decoder should then report the bit position where error has occurred. Finally having inputs from both the sense amplifiers and the Hamming decoder, the error correction block flips the value on the error bit and generates the 64-bit data output.
Project Description

The first task is to find a partner – the project will be executed in groups of 2. Smaller and larger groups (each of which is discouraged) need special permission from Prof. Rabaey.

The project will be executed in three phases.

- Phase 1: Design of Hamming error correction (March 6 ~ March 30)
- Phase 2: Row decoder (March 30 ~ April 13)
- Phase 3: Column decoder and sense amplifiers (April 13 ~ May 6)

For Phases 1 and 2, the outcome will be a simple report, while Phase 3 culminates in the creation of a poster and a short interview, where you will have the opportunity to describe how brilliant your design is.

PHASE 1:

In this phase you will be given the opportunity to design the Hamming decoder and error correction block in Fig. 2. While the principles of error detection and correction are laid out in Introduction, it is your task to realize them using static CMOS logic gates.

More specifically, you are asked to perform following tasks:

1. Derive block diagrams of the Hamming decoder and error correction block using static CMOS gates. Prove that obtained modules indeed implement intended functions, using a provided input sequence.
2. Identify the critical timing path(s) in your design.
3. Size the logic gates in your design to minimize propagation delay of the critical path(s).

   Design constraints include:
   - Process: GPDK 90nm in the EE141 Lab Library;
   - Supply Voltage: ≤1V;
   - Loading on each sense amplifier’s output: ≤4C, where C is the gate capacitance of a unit-size inverter (NMOS: 120n/100n, PMOS: 240n/100n);
   - Loading on each bit of the final output: 4C, where C is again the gate capacitance of a unit-size inverter.

4. Simulate your design in Cadence SPECTRE and report the critical path propagation delay.

No layout work is required in Phase 1.

Report

Please use the report template provided at the website. Be sure to justify important design decisions and emphasize all the vital information. Organization, conciseness, and completeness are of paramount importance. Good reports are short and to the point. Make sure to include and
annotate important plots to illustrate your effort. Do not repeat the information we already know. Make sure to fill out the cover-page and use the correct units.

**Grading**

The quality of the report is a major part of the grade.

For Phase 1, the grade will be equally divided over the correctness and completeness of the results, and the quality of the report.

**Have fun, and good luck!**