ENEE244-010x Digital Logic Design

Lecture 3

Announcements

- Homework 1 due today.
- Homework 2 will be posted by tonight, due Monday, 9/21.
- First recitation quiz will be tomorrow on the material from Lectures 1 and 2.
- Lecture notes are on course webpage.
- Substitute next time.
 - Will cover the basics of Boolean Algebra

Agenda

- Last time:
 - Signed numbers and Complements (2.7)
 - Addition and Subtraction with Complements (2.8-2.9)
- This time:
 - Overflow in 2's Complement
 - Parity and Arithmetic Modulo 2
 - Error detecting/correcting codes
 - Not following presentation in textbook.

Example of Overflow in 2's complement

- Assume $\ell = 8$
- Compute:
 01110000 + 01011100

• Compute

-01110000 - 01011100 (10001111 + 1) + (10100011 +1) 10010000 + 10100100

Overflow in 2's complement

• Overflow occurs in the following cases:

Operation	Operand A	Operand B	Result
A + B	≥ 0	≥ 0	< 0
A+B	< 0	< 0	≥ 0
A-B	≥ 0	< 0	< 0
A-B	< 0	≥ 0	≥ 0

• These conditions are the same as:

Carry-In to sign position \neq Carry-Out from sign position

Aside: Please Read 2.10.1, 2.10.2 in Textbook

- Binary-Coded Decimal (BCD) Schemes
 - Basic idea: Encode decimal numbers by encoding each decimal digit by its binary representation
 - − E.g. $15_{10} \rightarrow 0001 \ 0101$
 - Look over Table 2.7, 2.8
- Unit distance codes
 - Basic idea: Encode decimal numbers so that a single bit flips between two consecutive numbers:
 - E.g. In binary, $1_{10} = 0001_2$, $2_{10} = 0010_2$. Note that 2 bits flip.
 - In Gray code: $1_{10} = 0001$, $2_{10} = 0011$. Note that a single bit flips.
 - Look over Table 2.9
- You will not be tested or quizzed on this (at this point), but these codes will come up again later in the course.

Parity and Arithmetic Mod 2

Parity

- Parity 0: A 0/1 string has an even number of 1's.
 - Example: 001011100
- Parity 1: A 0/1 string has an odd number of 1's.
 - Example: 101010000
- Given a string, can also ask about the parity of a subset of positions
 - Example: Parity of positions 1, 3, 5, 6 in the string 001011100 is 1.

Mod 2 Arithmetic

- (N mod 2) is the remainder when dividing N by 2
 - 0 when N is even
 - 1 when N is odd
- Parity of a string is the sum of the bits modulo 2
 - Example: $001011100 = 0 + 0 + 1 + 0 + 1 + 1 + 1 + 0 + 0 = 4 = 0 \mod 2 = 0$.
- Parity of a subset of a string is exactly the dot-product mod 2.
 - Example: Parity of positions 1, 3, 5, 6 in the string 001011100 is the dot product $101011000 \cdot 001011100 = 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 = 3 \mod 2 = 1.$

Codes for Error Detection and Correction

Codes

- Encode algorithm $Enc(\vec{m}) = \vec{c} \cdot \vec{m}$ is the message, \vec{c} is the codeword.
- Decode algorithm $Dec(\vec{c}) = \vec{m}$
- Typically, \vec{c} will be longer than \vec{m} and will include redundant information.
- Redundancy is useful for detecting and/or correcting errors introduced during transmission.
- Assume \vec{m}, \vec{c} are in binary.
- Would like to detect and/or correct the flipping of one or multiple bits.

Error Detection/Correction

- Basic properties:
 - Distance of a code: minimum distance between any two codewords (number of bits that need to be flipped to get from one codeword to another)

- Rate of a code: $\frac{|\vec{m}|}{|\vec{c}|}$ (length of \vec{m} / length of \vec{c})

- Distance determines the maximum number of errors that can be detected/corrected.
- Goal of coding theory is to construct codes with optimal tradeoff between distance and rate.
- Must also have efficient encoding, decoding and error correcting procedures.

Error Detection/Correction

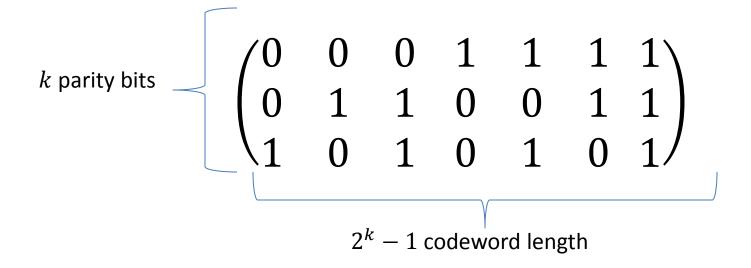
- Error detection: can detect at most *dist*-1 errors, where *dist* is the minimum distance of the code.
- Error correction: can correct at most (dist 1)/2 errors

Error Detection: Parity Check

- Encode: On input $\vec{m} = 11001010$
 - Output $\vec{c} = 11001010|b$, where b is the parity of \vec{m} . $b = 1 + 1 + 0 + 0 + 1 + 0 + 1 + 0 = 4 \mod 2 = 0$
- Decode: On input $\vec{c} = 11001010|b$, output 11001010
- Error detection:
 - If a non-parity bit is flipped
 - If the parity bit is flipped
- Can detect only one error. Why?

Error Correction for 1 Error: The Hamming Code

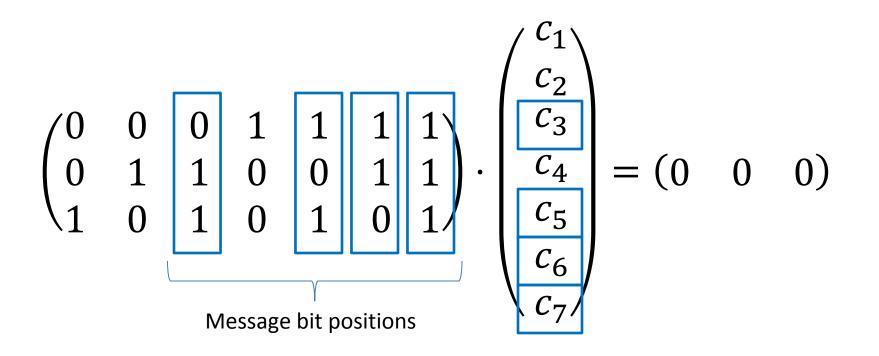
- View codeword as a vector (c_1, c_2, \dots, c_7)
- Some bits will be information bits, some bits will be parity-check bits.
- Parity-check matrix H:

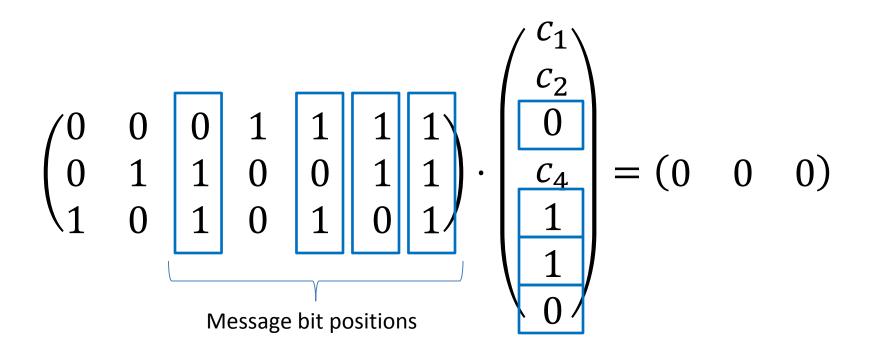


- For any codeword \vec{c} , $H \cdot \vec{c} = \vec{0}$.
- Parity-check matrix *H*:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} = (0 \quad 0 \quad 0)$$

• To encode a message $\vec{m} = m_1, m_2, m_3, m_4$





• To encode a message $\vec{m} = m_1, m_2, m_3, m_4$

