Error Detection and Correction

Outline for Today

1. Exclusive-or operation (XOR, $\oplus$)
   - Definition
   - Some interesting applications
   - Hamming Distance
Exclusive Or

For boolean variables $p$ and $q$, we use 0 for false and 1 for true.

**Definition:** For boolean variables $p$ and $q$, the exclusive or of $p$ and $q$, denoted $p \oplus q$, is defined by:

<table>
<thead>
<tr>
<th>$\oplus$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

For same length bit strings $x, y$ (called **words**), we apply the operation bitwise. So suppose that $x = x_1x_2...x_n$, and $y = y_1y_2...y_n$, where for every $i$, $x_i$ and $y_i$ are bits. Then $z = x \oplus y$, where $z = z_1...z_n$, and $z_i = x_i \oplus y_i$.

**Definition:** Let $x$ be a bit string. Then $\overline{x}$ is the complement of $x$, in which every bit is flipped compared to $x$. That is, if $x = x_1x_2...x_n$, then $\overline{x} = y_1y_2...y_n$, where for every $i$, $y_i = 1$ if $x_i = 0$, and $y_i = 0$ if $x_i = 1$. 
Basic Properties of $\oplus$

1. **Commutativity**: $x \oplus y = y \oplus x$

2. **Associativity**: $(x \oplus y) \oplus z = x \oplus (y \oplus z)$

3. **Identity**: Let 0 represent the bit string with all 0 entries. Then for any bit string $x$, $x \oplus 0 = x$.

4. **Complement**: Let 1 represent the bit string with all 1 entries. Then $x \oplus 1 = \overline{x}$.

5. **Inverse**: $x \oplus x = 0$ and $x \oplus \overline{x} = 1$. 
Applying $\oplus$ to Non-negative Integers

- Convert each operand to binary
- Add leading zeros as needed to make the bit strings the same length
- Apply the $\oplus$ operation to the two bit strings/words
- Convert the result to the corresponding non-negative integer
Small Applications of ⊕

- bit selection
- toggling
- exchange
- storage for doubly-linked lists

More detailed discussion of these: course pack
Selecting a Bit

**Problem:** Given three boolean variables \( x, y \) and \( u \), compute \( w \) such that \( w = x \) if \( u = 0 \) and \( w = y \) if \( u = 1 \).

**Solution:** Set \( w = ((x \oplus y) \land u) \oplus x \)

**Proof:** in-class exercise

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>u</th>
<th>( x \oplus y )</th>
<th>( (x \oplus y) \land u )</th>
<th>( ((x \oplus y) \land u) \oplus x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Forming a Word by Selecting Bits from Two Words

Now we extend the bit selection idea to words.

Suppose that $x$, $y$ and $u$ are words (same length bit strings). We want to define a word $w$ such that, for every $i$,

$w_i = x_i$ if $u_i = 0$, and
$w_i = y_i$ if $u_i = 1$.

Since we apply $\oplus$ and $\land$ bitwise when we apply them to bit strings, it follows from our previous proof that:

$w = ((x \oplus y) \land u) \oplus x$.

**Example:** Let $x = 10110$ and $y = 01011$, and set $u = 00110$. What is $w$?
Toggling a Boolean Variable

**Goal:** Given two boolean variables $p$ and $q$, write a single assignment statement that *toggles* the value of another boolean variable $x$ between $p$ and $q$.

- If $x = p$, then the assignment statement should set $x$ to $q$
- If $x = q$, then the assignment statement should set $x$ to $p$

**Solution:** Assume that $x$ is either equal to $p$ or $q$. Our assignment statement is $x = x \oplus (p \oplus q)$.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$x$</th>
<th>$p \oplus q$</th>
<th>$x \oplus (p \oplus q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Exercise:** complete truth table to verify that the assignment statement is correct.
Word Toggling

For two words \( p \) and \( q \), we want an assignment statement that toggles a variable \( x \) between \( p \) and \( q \).

- If \( x = p \), then the assignment statement should set \( x \) to \( q \)
- If \( x = q \), then the assignment statement should set \( x \) to \( p \)

In our previous proof, we showed (bit by bit) that the assignment statement \( x = x \oplus (p \oplus q) \) toggles between \( p \) and \( q \), assuming that \( x \) is initially either \( p \) or \( q \).
Exchanging Values of Two Boolean Variables

**Problem:** Swap the values of two boolean variables $x$ and $y$ without using a temporary variable.

**Solution:** Use the following 3 assignment statements:

$x = x \oplus y$
$y = x \oplus y$
$x = x \oplus y$

**Proof:** Suppose that initially $x = R$ and $y = S$. We need to show that after the 3 assignment statements, $x = S$ and $y = R$.

After the first assignment, $x = R \oplus S$. After the second assignment,

$y = (R \oplus S) \oplus S$

$= R \oplus (S \oplus S)$ by associativity

$= R \oplus 0$ by inverse property

$= R$ by identity property.

After the 3rd assignment, $x = (R \oplus S) \oplus R = S$ by the associativity and inverse properties. $\square$
Exchanging Values of Two Words

**Problem:** Swap the values of two words $x$ and $y$ without using a temporary variable.

Since $\oplus$ is a bitwise operator, our proof on the previous slide works, and the following assignment statements work:

$x = x \oplus y$

$y = x \oplus y$

$x = x \oplus y$

$y = x \oplus y$

**Question:** What happens if $x$ and $y$ are two references to the same memory location?
Doubly-Linked Lists

_Avoid clever tricks like the plague!_
–Edsger Dijkstra

**Goal:** Instead of storing two pointers, a left pointer and a right pointer, only store one value.

**Trick:** Only store the XOR of the left and right pointers.

- Works if you are arriving at a node from one of the neighbors
- Won’t work if you are using an outside pointer to a node

**Question:** Why does it work??
Hamming Distance

**Definition:** The Hamming distance between two words $x$ and $y$ is the number of 1s in $x \oplus y$.

**Note:** You can also think of the Hamming distance as the number of bit flips needed to change $x$ into $y$.

**Definition:** A distance function $d : S \times S \to \mathbb{R}$ is **metric** if it satisfies the following properties:

- **Non-negativity:** $\forall x \forall y d(x, y) \geq 0$
- **Distinctness:** $\forall x \forall y d(x, y) = 0$ if $f x = y$
- **Symmetric:** $\forall x \forall y d(x, y) = d(y, x)$
- **Triangle inequality:** $\forall x \forall y \forall z d(x, y) + d(y, z) \geq d(x, z)$

**Definition:** A **metric space** is a set $S$ with an associated metric distance function $d$. 
Hamming Distance

Claim: For any non-negative integer $k$, Hamming distance defines a metric distance function over the set of all words of length $k$.

Lemma: Let $(S, d)$ be a metric space. Let $k \in \mathbb{N}$, and let $d'(x, y) = \sum_{1 \leq i \leq k} d(x_i, y_i)$, for all $x = (x_1, x_2, \ldots, x_k)$ and $y = (y_1, \ldots, y_k)$ in $S^k$. Then $(S^k, d')$ is a metric space also.

Proof: Exercise
Hamming Distance is Metric

**Theorem:** For any non-negative integer $k$, Hamming distance defines a metric distance function over the set of all words of length $k$.

By our lemma, all we need to prove is that Hamming distance defines a metric space over \{0, 1\}, the set of all words of length 1.

But this is easy (exercise).