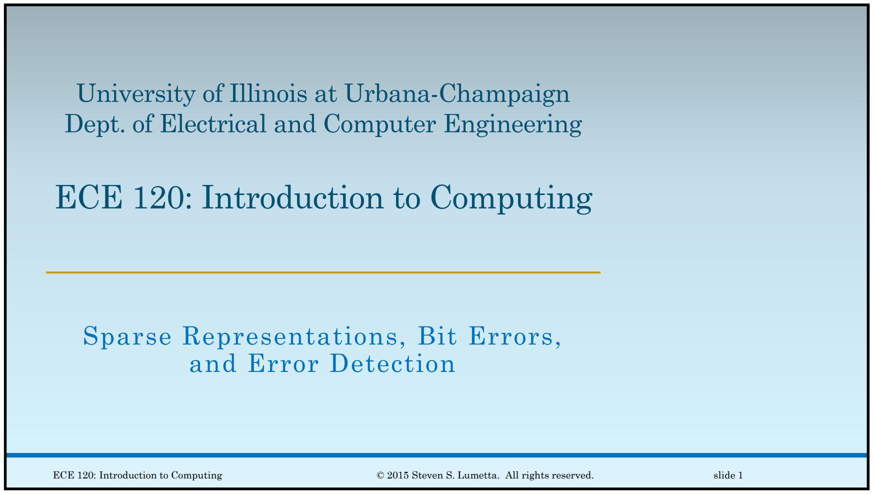
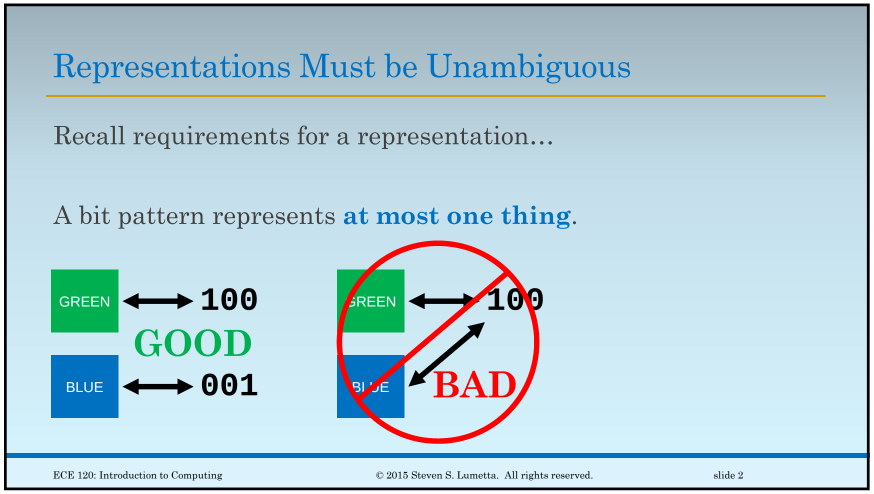


University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

# ECE 120: Introduction to Computing

## Sparse Representations, Bit Errors, and Error Detection



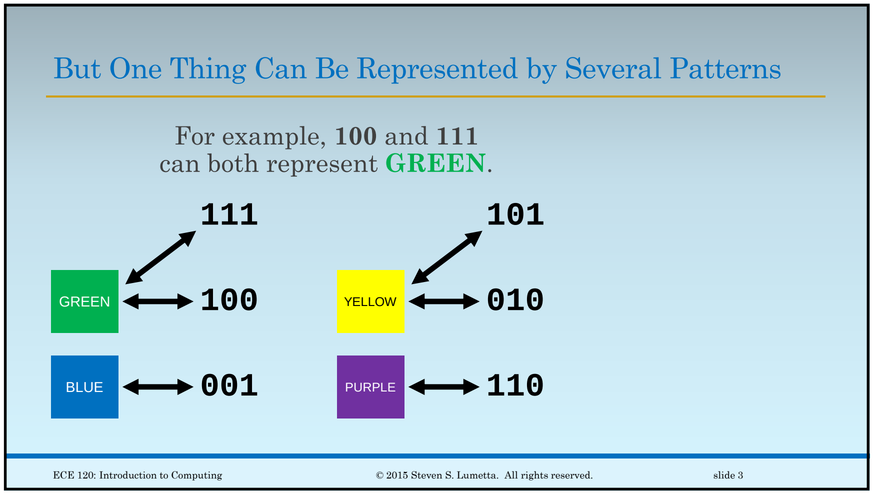
## Representations Must be Unambiguous

Recall requirements for a representation…

A bit pattern represents **at most one thing**.

GREEN ⟷ 100
BLUE ⟷ 001
GOOD

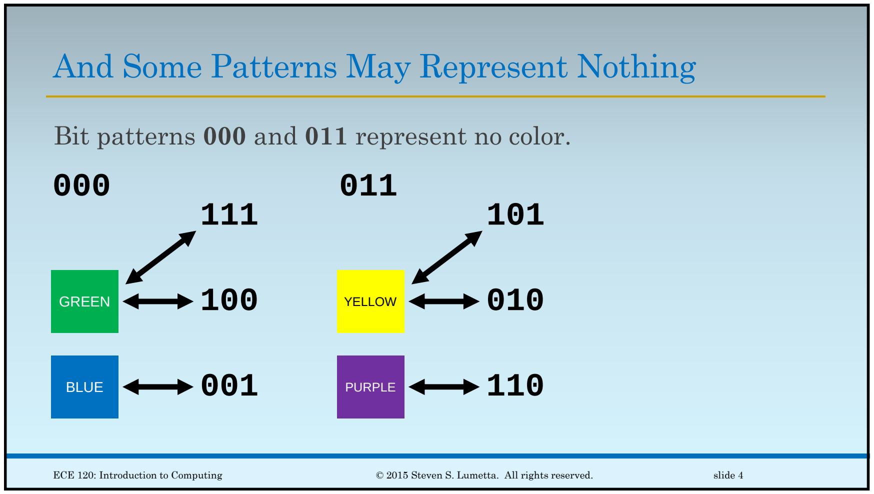
GREEN ⟷ 100
BLUE ⟷ 100
BAD



## But One Thing Can Be Represented by Several Patterns

For example, **100** and **111**
can both represent **GREEN**.

GREEN ⟷ 111
GREEN ⟷ 100
BLUE ⟷ 001
YELLOW ⟷ 101
YELLOW ⟷ 010
PURPLE ⟷ 110



## And Some Patterns May Represent Nothing

Bit patterns **000** and **011** represent no color.

000
011

GREEN ⟷ 111
GREEN ⟷ 100
BLUE ⟷ 001
YELLOW ⟷ 101
YELLOW ⟷ 010
PURPLE ⟷ 110

## Example: Binary-Coded Decimal (BCD)

BCD maps **decimal digits** into 4-bit patterns.

Each pattern is the binary value of the digit.

Six patterns are unused ($2^4 - 10 = 6$).

| | | | |
|---|---|---|---|
| 0 ↔ | 0000 | 5 ↔ | 0101 |
| 1 ↔ | 0001 | 6 ↔ | 0110 |
| 2 ↔ | 0010 | 7 ↔ | 0111 |
| 3 ↔ | 0011 | 8 ↔ | 1000 |
| 4 ↔ | 0100 | 9 ↔ | 1001 |

```
1010
1011
1100
1101
1110
1111
```

## Unused Bit Patterns Can Be Used to Detect Errors

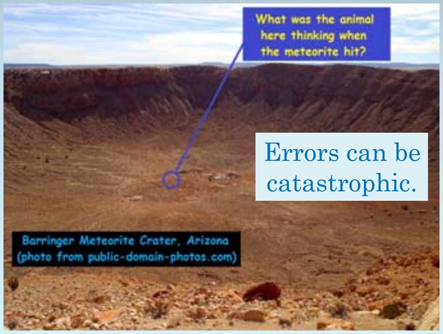
Bit patterns on the right have no meaning.

Seeing one of these patterns means that **an error has occurred**.

| | | | |
|---|---|---|---|
| 0 ↔ | 0000 | 5 ↔ | 0101 |
| 1 ↔ | 0001 | 6 ↔ | 0110 |
| 2 ↔ | 0010 | 7 ↔ | 0111 |
| 3 ↔ | 0011 | 8 ↔ | 1000 |
| 4 ↔ | 0100 | 9 ↔ | 1001 |

```
1010
1011
1100
1101
1110
1111
```

## But What Exactly is an "Error?"



Errors can be catastrophic.

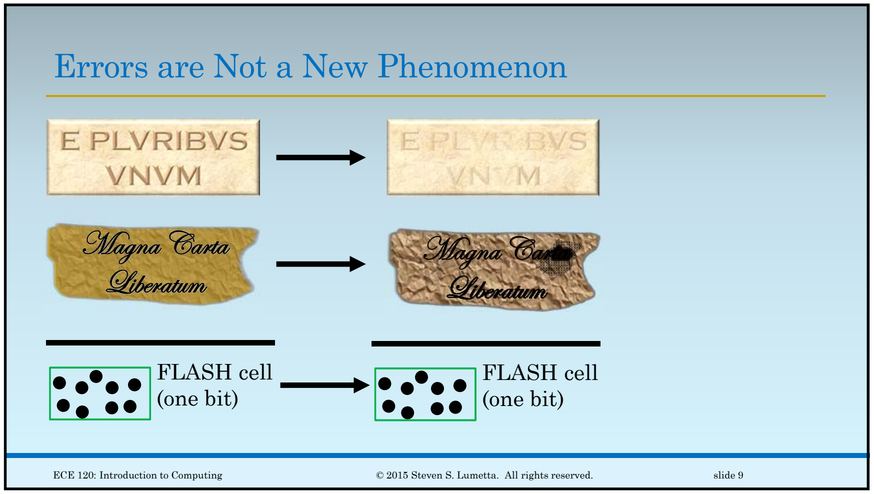
Barringer Meteorite Crater, Arizona (photo from public-domain-photos.com)

## We Need to Choose a Model for Errors

Cannot handle "all" errors.

- Catastrophes are always possible.
- But catastrophes are (hopefully) uncommon!

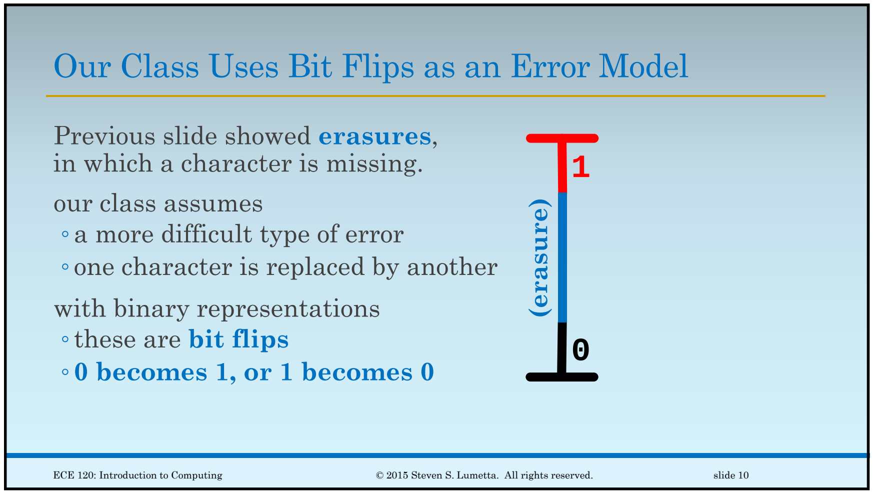
Let's focus on frequent types of errors.



## Errors are Not a New Phenomenon

E PLVRIBVS VNVM
E PLV  BVS VN  M
Magna Carta Liberatum
Magna Carta Liberatum

FLASH cell (one bit) → FLASH cell (one bit)

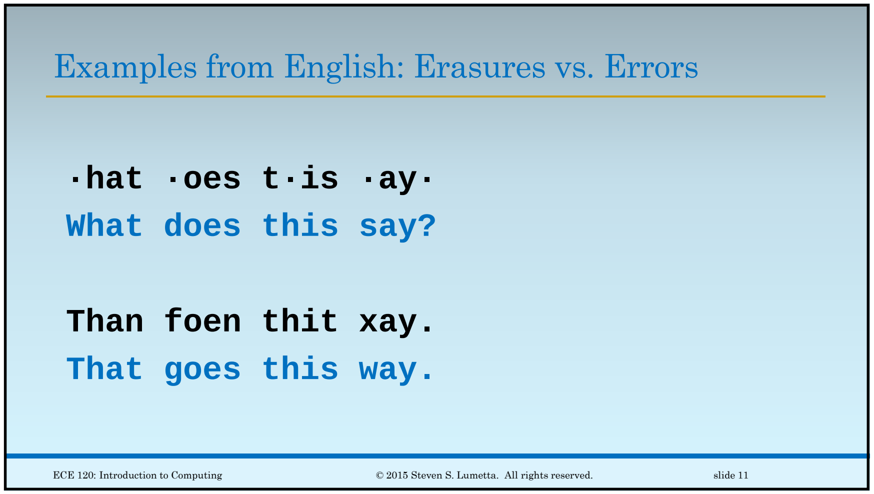


## Our Class Uses Bit Flips as an Error Model

Previous slide showed **erasures**,
in which a character is missing.

our class assumes
- a more difficult type of error
- one character is replaced by another

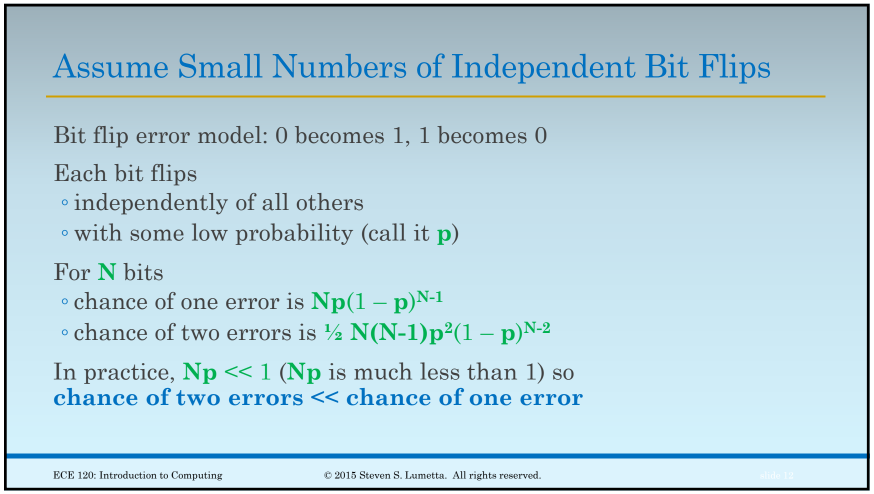
with binary representations
- these are **bit flips**
- **0 becomes 1, or 1 becomes 0**



## Examples from English: Erasures vs. Errors

**·hat ·oes t·is ·ay·**
**What does this say?**

**Than foen thit xay.**
**That goes this way.**



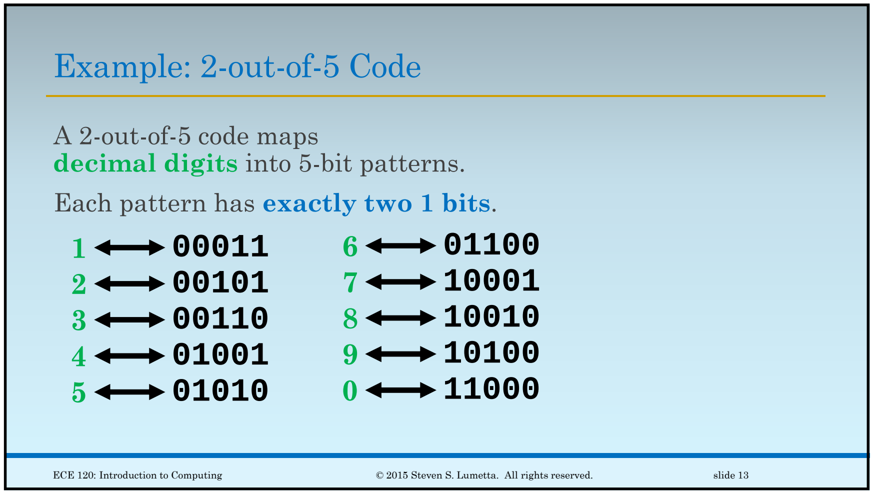
## Assume Small Numbers of Independent Bit Flips

Bit flip error model: 0 becomes 1, 1 becomes 0

Each bit flips
- independently of all others
- with some low probability (call it $p$)

For $N$ bits
- chance of one error is $Np(1 - p)^{N-1}$
- chance of two errors is $\frac{1}{2} N(N-1)p^2(1 - p)^{N-2}$

In practice, $Np \ll 1$ ($Np$ is much less than 1) so
**chance of two errors << chance of one error**

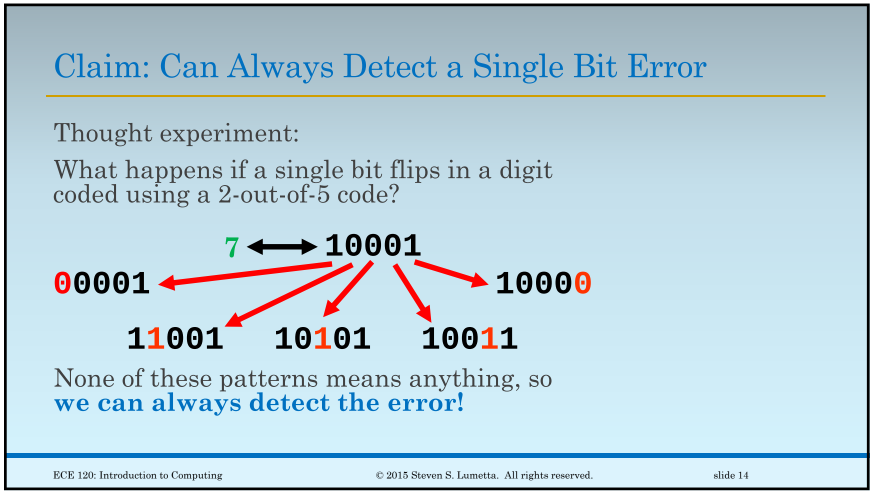


## Example: 2-out-of-5 Code

A 2-out-of-5 code maps **decimal digits** into 5-bit patterns.

Each pattern has **exactly two 1 bits**.

| | | | | | |
|---|---|---|---|---|---|
| 1 | ⟷ | 00011 | 6 | ⟷ | 01100 |
| 2 | ⟷ | 00101 | 7 | ⟷ | 10001 |
| 3 | ⟷ | 00110 | 8 | ⟷ | 10010 |
| 4 | ⟷ | 01001 | 9 | ⟷ | 10100 |
| 5 | ⟷ | 01010 | 0 | ⟷ | 11000 |

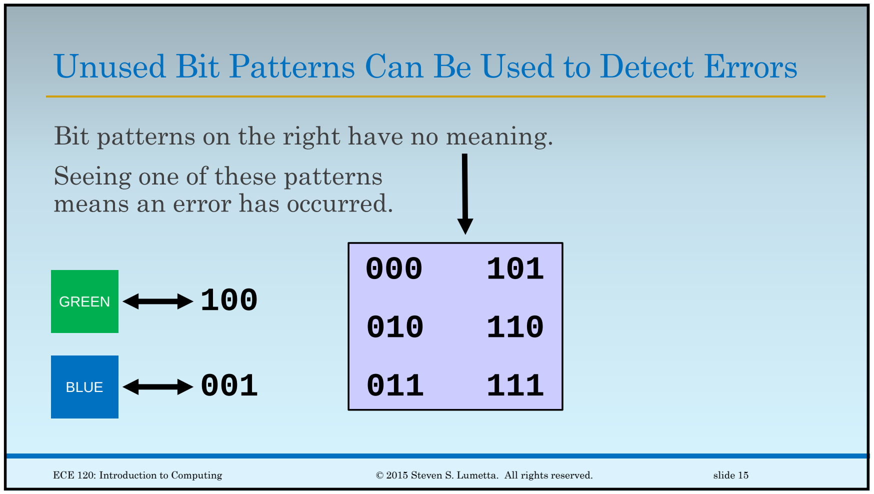


## Claim: Can Always Detect a Single Bit Error

Thought experiment:

What happens if a single bit flips in a digit coded using a 2-out-of-5 code?

None of these patterns means anything, so **we can always detect the error!**



## Unused Bit Patterns Can Be Used to Detect Errors

Bit patterns on the right have no meaning.

Seeing one of these patterns means an error has occurred.

| | | |
|---|---|---|
| GREEN | ⟷ | 100 |
| BLUE | ⟷ | 001 |

| | |
|---|---|
| 000 | 101 |
| 010 | 110 |
| 011 | 111 |