MODULE 2: Data Representation

Lecture 2.6
Error Codes

Prepared By:

- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering

Virginia Tech

Lecture 2.6 Objectives

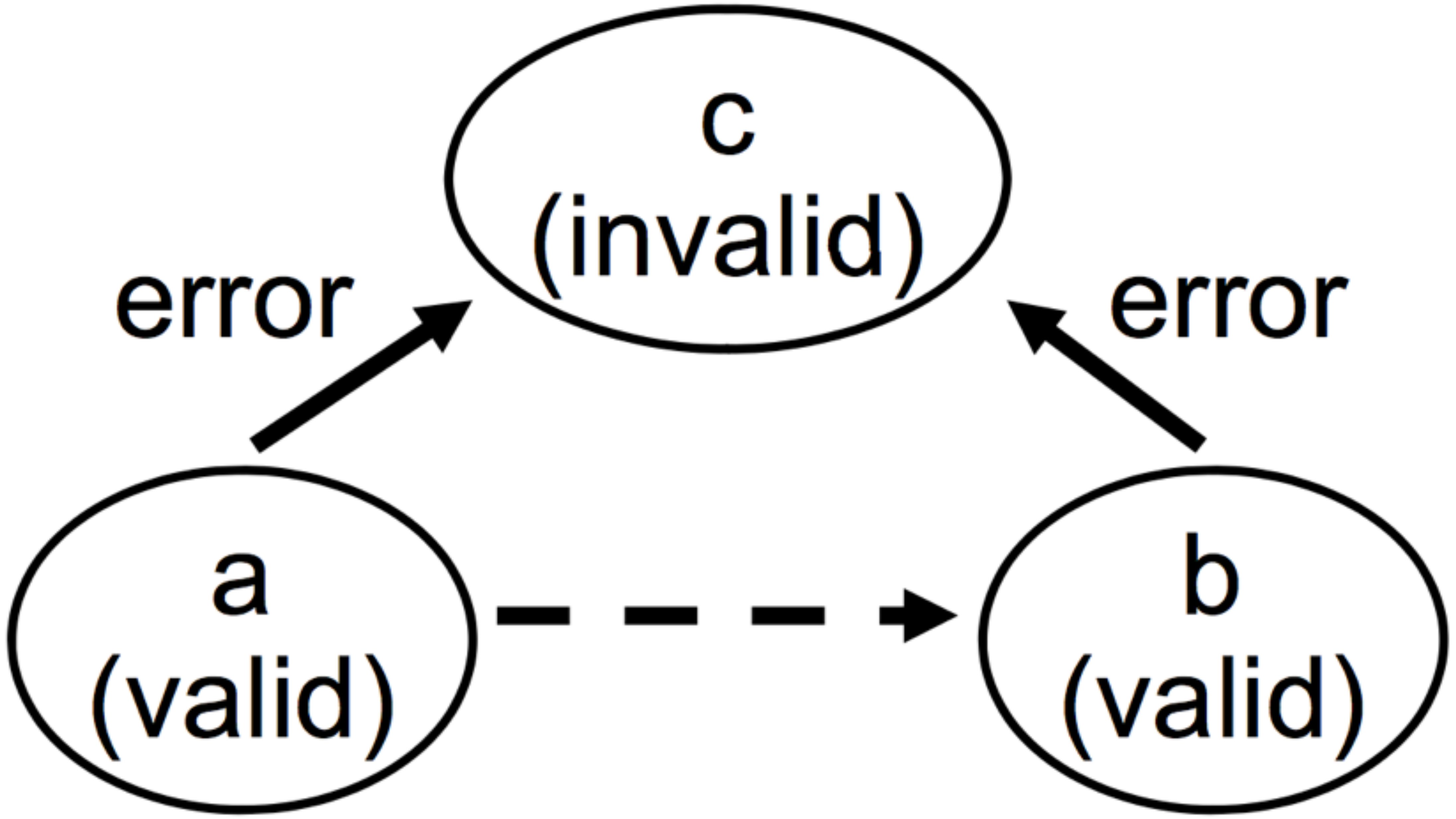
- Explain the role of error detection and error correction codes
- Calculate the Hamming distance between two codewords and the Hamming distance of a code
- Calculate the odd or even parity bit for a code word
- Determine the CRC, given a message and a generator polynomial
- Use a CRC to detect bit errors

Error Codes

- Some codes allow for:
  - Error detection (is there an error?)
  - Error correction (detect and correct error)
- To detect or correct errors, there must be redundant information, i.e., extra bits
  - The greater the redundancy, the greater the potential for error detection and/or correction
- Used in communication systems (noise) and in memory (transient or “hard” bit errors)

Code Distance

- For error detection, an error must change a set of bits from a valid code to an invalid code
- The code distance is number of bits that separate valid and invalid codes
    - Code distance of 2 required for error detection



Hamming Distance

- The Hamming distance between two codewords is the number of bit positions in which the codewords differ
    - Example: for code words 10001001 and 10110001, the Hamming distance d = 3
- The Hamming distance of a code is the minimum distance between any two of its codewords
- To detect all d-bit errors, need a distance d+1 code
- To correct all d-bit errors, need a distance 2d+1 code

Parity Codes

- Parity codes provide single-bit error detection
- Example with single-bit error
  - SOURCE sends: 100 0001 (ASCII “A”)
  - RECEIVER gets: 100 0011 (ASCII “C”)
- Parity codes use an extra bit — the parity bit — to ensure an odd or even number of 1’s in the code word
- Odd parity — odd number of 1’s is valid
- Even parity — even number of 1’s is valid

Odd Parity Example (1)

- Data word is 100 0001 (ASCII “A”)
- Code word is **1**100 0001 (odd parity bit added)
- Suppose 1 bit is corrupted
    - 11000011 is received
    - Received code word is known to be in error since it contains an even number of ones
    - But, we can’t determine the original word
        - Could be 1100 0010, 1100 0001, ...
        - No error correction capability

Odd Parity Example (2)

- Data word is 100 0001 (ASCII “A”) and code word is 1100 0001 (odd parity bit added)
- Suppose bits 0 and 1 are corrupted
  - 11000010 is received
  - Received code word appears to be correct
  - Parity bits can detect only an odd number of bit errors

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

1. What is the Hamming distance d between code words 10111001 and 10000000?
2. What is the value of the parity bit if you want to transmit the ASCII “B” with odd parity?

CHECK POINT

Answer:

1. The Hamming distance between code words 10111001 and 10000000 is d = 4
2. To transmit “100 0010” and achieve odd parity, the parity bit should be 1.

If you have any difficulties, please review the lecture video before continuing.

Cyclic Redundancy Check

- A powerful, widely used error detection code
- To each *k*-bit information block or stream, append *n* error checking bits
- Each valid (n+k)-bit codeword is exactly divisible by a pre-determined value
- Checking for errors:
  - Divide the codeword by the pre-determined value
  - If remainder is non-zero, an error is detected

CRC Polynomials

- CRC is a polynomial code
- Blocks of bits are represented by a polynomial
- The polynomial is used to calculate the error checking bits
- Example: represent 110001 as $x^5 + x^4 + 1$
- Standard polynomials exist, depending on how many error checking bits you want to add
  - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
  - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x + 1$

Calculating the CRC

- Suppose a *k*-bit message M and an order-*n* generator polynomial
  - Form a new block of bits by appending *n* 0s to the right of the message
  - Divide the binary equivalent of the generator polynomial into the new block of bits formed above
  - The (*n*+*k*)-bit codeword is formed by appending the *n*-bit remainder to the right of the original message M

CRC Example (1)

- Suppose we want to calculate the CRC for message 1101011011, with generator polynomial $x^4 + x + 1$
- Form the block of bits 11010110110000
- Divide 10011 into 11010110110000, find the remainder
- The CRC is the remainder, and the full codeword is the original message with the CRC appended to it

CRC Example (2)

```
          110000101
10011  | 11010110110000
         10011
         010011
          10011
          0000010110
               10011
               0010100
                 10011
                 001110  ← remainder
```

Codeword:
11010110111110

CRC Example (3)

```
                110000101
           ┌──────────────────
10011      │ 11010110111110
             10011
             ─────
             010011
              10011
              ─────
              0000010111
                   10011
                   ─────
                   0010011
                     10011
                     ─────
                     000000  ←── remainder
```

Double checking that the codeword is divisible by the generator

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Work by hand the CRC remainder calculations on the previous two slides

If you have any difficulties, please review the lecture video before continuing.

CRC Error Detection Properties

- CRC detects:
    - All single-bit errors
    - All burst errors that are ≤ *n*
    - All double bit errors, as long as the generator contains at least three 1s

Reed-Solomon (RS) Codes

- An error correction code
  - The RS(255,223) code uses 223 information bytes and 32 syndrome bytes to form 255-byte codewords
  - The RS(255,223) code can correct up to16 bit errors in the codeword
- Widely used in:
  - Data storage: CDs, DVDs, Blu-ray Discs
  - Data communications: Digital Subscriber Line (DSL), WiMax, Digital Video Broadcasting (DVB)

Summary

- If changing a bit (a 1 to a 0, or a 0 to a 1) results in an invalid code word, the error can be detected
- We may want to use error detection
  - For example, in a communications link the receiver would detect the error and explicitly or implicitly request a retransmission
- We may want to use error correction
  - For example, a corrupted data word can be corrected at the receiver without requiring retransmission
- Error detection and correction require sending or storing extra bits, so it doesn’t come for free

MODULE 2: Data Representation

Lecture 2.6
Error Codes

Prepared By:

- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering

Virginia Tech