ERROR DETECTION

Outline

1. Error Discovery vs Recovery
2. Types of Errors
3. Vertical Redundancy (Parity)
4. Longitudinal Redundancy (2D-Parity)
5. CRC Codes
6. Checksums

Discovery vs Recovery

Errors

- Physical layer produces a virtual bit pipe.
- Nyquist's theorem gives the signal frequencies which are sufficient to carry the signal rate.
- Shannon's theorem gives the optimal channel capacity under random noise but gives no idea on how to achieve it.
- Errors still occur due to other sources such as switching effects, cross talk, lightning, etc.
- Error recovery occurs at physical through transport layer but mainly at the data link layer

Transmitted Error Problem

- After meeting a new friend at a party, you want to get his/her cell phone number: ten digits.
- Your friend hands you the telephone number on a scrap of paper as

  $$617\ 5550\boxed{?}23$$

  $$617\ 5550\boxed{*}23$$

  where ? means that the digit is unrecognizable, while * means digit is misrecognizable.
- How do you figure out the missing digit?
  - Call all possible ten numbers!
  - Ask for the number again!
  - Correct it yourself!

Error Recovery

- Error recovery takes on two forms as Error Detection and Error Correction, depending on the requirements of the application.
- **Error Detection:**
  - must be followed by retransmission:
  - most often used at higher levels, and
  - leads to retransmission strategies discussed later
- **Error Correction:**
  - most often used at the physical layer to produce bit pipe with low error rate

Types of Errors

Types of Errors: Single bit error

- **Single bit error:**
  - Just a single bit changed:
    from 0 to 1 or from 1 to 0.
  - A single bit error may affect a larger block of data bits.
  - They are more likely in parallel (because in parallel more bits are sent at the same time) than serial transmissions!

Types of Errors: Burst error

- **Burst Error:** is a contiguous block of at least two error bits (starting with an error bit and ending with an error bit).

send:	0	1	1	0	0	1	0	1	0	0	1	1
errors			↓			↓		↓				
receive:	0	1	0	0	0	0	0	0	0	0	1	1

- In a burst not all bits between two endpoints are in error!
- Length of a burst error is measured as the *distance* from the first corrupted to the last corrupted bit in this burst.
  - What is the length of the burst above?

Modeling errors

- Errors are notoriously difficult to model.
- Usually we look at the frequency of errors in an application.
  - **Error rate:** probability a bit is in error (bits are often assumed to be independent).
- **Typical error rates**
  - Wireless links: 10^{-4}
  - ISDN line: 10^{-6}
  - Optical fiber: 10^{-10}

Example

- A network channel has bit error rate p.
- How many errors do you expect in a packet of length n?

$$pn$$

- Assume errors in a packet are independent of each other.
- What is the probability a packet of length n has an error?
  - The probability that a given bit is correct is $1 - p$.
  - The probability that all bits are correct is $(1 - p)^n$.

$$\begin{aligned}
\Pr[\text{Packet has an error}] &= 1 - \Pr[\text{Packet has no error}] \\
&= 1 - (1 - p)^n \\
&\approx 1 - 1/e,
\end{aligned}$$

provided that $p = 1/n$, where e is Euler's number.

Example

- So, if the bit error rate is $1/n$ then a packet of length n will have error with a "non-negligible" probability $\sim 1 - 1/e$.
- Following packet lengths for network types
  - Wireless links: packet length $n = 10^4$ bits
  - ISDN line: packet length $n = 10^6$ bits
  - Optical fiber: packet length $n = 10^{10}$ bits

  will give you probability $\sim 1 - 1/e$ that a transmitted packet in this medium will have error!
- That's no good!

Concept of Redundancy

- Error detection uses the concept of redundancy:
  - this means adding extra bits at the source in order to detect errors at the destination.
- **Example:** repeat every bit twice.
  - Receiver will do a bit-by-bit comparison.
  - Error detection system is good but it is not efficient.
- Rather than repeat the whole string twice a shorter stream of bits could be appended.

Modelling errors

- A typical error detection/correction algorithm performs an operation

$$\text{word} \to \text{code}.$$

transforming the original word into a code word (or code for short).

  – We measure efficiency with the **Redundancy**

$$\text{Redundancy} = \frac{\text{length of code}}{\text{length of word}}$$

Parity

XOR or mod2

- If $b, b' \in \{0, 1\}$ are bits

$$b \oplus b' = \begin{cases} 0 & \text{if } b = b' = 0 \text{ or } b = b' = 1 \\ 1 & \text{otherwise} \end{cases}$$

- Sometimes we also write $b + b' \mod 2$.
- Sometimes we also use it for sequences of bits

$$(x_1 x_2 \cdots x_n) \oplus (y_1 y_2 \cdots y_n) = (x_1 \oplus y_1)(x_2 \oplus y_2) \cdots (x_n \oplus y_n)$$

- Example:

$$(01101) \oplus (10011) = 11110$$

VRC: Vertical Redundancy (or Parity) Check

- Based on bit XORing a sequence $x_1 x_2 \cdots x_n$.
- Single bit equal to the exclusive-or of the bits is added,

$$\begin{aligned} \text{Parity } (x_1 x_2 \cdots x_n) &= x_1 \oplus x_2 \oplus \cdots \oplus x_n \\ & \left(\sum_{i=1}^{n} x_i \right) \text{ mod } 2. \end{aligned}$$

- If number of 1 bits is even result is 0, otherwise 1
- Given string x of length n, the codeword of x is a string of length $n + 1$ defined by $C(x) = x\text{Parity}(x)$
- Detects a single error

VRC Check: Example

- **Example:**

word:	00101010
check bit:	1
code:	001010101

- When the receiver "receives the code" transmitted through the channel, it checks whether or not the sum of the bits is equal to 0 modulo 2.

2D Parity

LRC: Longitudinal Redundancy (or 2d-Parity) Check

- Bits placed in $m \times n$ array with $m, n \geq 2$.

  Rows are sent one after the other.

  One parity bit used for each row and column (total of $m + n$ bits).

- Detects up to 2 errors

- Corrects a single error

- Often used with ASCII stream ($m = 8$)

LRC Check: Example

- Consider a sequence of 35 bits:

$$10010100111010111000110001110011001$$

- Arrange the sequence as a 5×7 matrix and add the check bits (one for each row and column).

$$\begin{array}{ccccccc|c}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 1 & 1 & 1 & 1 &
\end{array}$$

- Transmit as a sequence.

LRC Check: Example

- The receiver arranges the sequence into a matrix

$$\begin{array}{ccccccc|c}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 1 & 1 & 1 & 1 &
\end{array}$$

  and checks the condition.

- There is a single error! Can you locate the error?

LRC Check: Example

- The receiver arranges the sequence into a matrix

$$
\begin{array}{ccccccc|c}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 1 & 1 & 1 & 1 &
\end{array}
$$

  and checks the condition.

- Here there are two errors!
- Can you locate the errors? No!
- You can only detect that two errors occurred!

LRC Check: Example

- It is even possible you will not notice any error!!!

$$
\begin{array}{ccccccc|c}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & \boxed{1} & 0 & \boxed{0} & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & \boxed{1} & 1 & \boxed{0} & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 1 & 1 & 1 & 1 &
\end{array}
$$

- If all four bits in “boxes” are in error you will not notice anything!

CRC

Cyclic Redundancy Check Codes

- Based on the theory of cyclic error-correcting codes.
- Using cyclic codes, encode messages by adding a fixed-length check value, for the purpose of error detection in communication networks.
  - First proposed by W. Wesley Peterson in 1961.
- Most commonly used error detection scheme in use are Cyclic Redundancy Check (CRC) codes.

Polynomials in Z_2

- A polynomial is an expression that can be built from constants and symbols called variables by means of addition, multiplication and exponentiation to a non-negative power.
- A polynomial in a single indeterminate x can always be written (or rewritten) in the form

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0,$$

where $a_0, \ldots, a_n$ are constants and x is the variable.

- This can be expressed more concisely by using summation notation:

$$\sum_{i=0}^{n} a_i x^i$$

Polynomials in Z_2

- Two such polynomial expressions may be added, multiplied, divided, etc
- Example: Addition

$$(x^3 + x + 1) + (x^2 + x + 1) = x^3 + x^2$$

- Example: Multiplication

$$(x^3 + x + 1) \cdot (x^2 + x + 1) = x^5 + x^4 + 1$$

Cyclic Redundancy Check Codes: Definitions

- L: length of check bits.
- K: length of data bits.
- Bit strings are treated as polynomials over Z_2.
- A bit string $s_{K-1}s_{K-2}\cdots s_1 s_0$ is encoded as a polynomial in Z_2:

$$s(x) = s_{K-1}x^{K-1} + s_{K-2}x^{K-2} + \cdots + s_1 x + s_0$$

- Addition and multiplication of polynomials is done mod2

Polynomials and Bit Sequences

- In the transformation between polynomials and bit sequences "missing coefficients of the polynomial" must be included in the bit sequence as 0s.

$$x^7 + x^5 + x^2 + x + 1$$

$$\downarrow$$

$$x^7 + 0x^6 + x^5 + 0x^4 + 0x^3 + x^2 + x + 1$$

$$\downarrow$$

$$10100111$$

- Conversely, 0s in the bit sequence are missing coefficients in the polynomial.

Polynomials

- A monomial is a number times a power of x: ax^n
  - $3x^2, 2x^7, 8$ are all monomials.
- A polynomial is a sum or difference of monomials
  - $4x^5 - 3x^2 - 1, 4x^2, 2$ are all polynomials
- When we write $P(x) = 3x^3 - 2x^2 + 1$ we say "P of x"
- To add or subtract polynomials, we just collect like terms:
  - Example: $P(x) = x^2 + 3x + 5$ and $Q(x) = 4x^3 - 2x^2 + 3x - 2$
- How do we multiply polynomials?
  - Example: $P(x) = 3x + 5$ and $Q(x) = 4x^3 + 3x - 2$

Polynomials in mod2 Arithmetic

Lets try to do all previous examples mod2

- A monomial is a number times a power of x:
  - $3x^2 = x^2, 2x^7 = 0, 8 = 0$ are all monomials.
- A polynomial is a sum or difference of monomials
  - $4x^5 - 3x^2 - 1 = x^2 - 1, 4x^2 = 0, 2 = 0$ are all polynomials
- To add or subtract polynomials, we just collect like terms:
  - Example: $P(x) = x^2 + 3x + 5 = x^2 + x + 1$ and $Q(x) = 4x^3 - 2x^2 + 3x - 2 = x$
- How do we multiply polynomials?
  - Example: $P(x) = 3x + 5 = x + 1$ and $Q(x) = 4x^3 + 3x - 2 = x$

Somehow, things get simpler mod2!

CRC Algorithm

1. **Input:** sequence of data bits of length K and a generator polynomial of degree L.

2. Append L bits (also called CRC bits) to the K data bits

   $$\boxed{\text{DATA bits}} \rightarrow \boxed{\text{DATA bits}}\boxed{\text{CRC bits}}$$

   in such a way that the resulting sequence of bits gives rise to a polynomial that is divisible by the generator polynomial.

3. Send these $K + L$ bits.

4. At the receiving end compute the data bits and check the error condition.

   $$\boxed{\text{DATA bits}}\boxed{\text{CRC bits}} \rightarrow \boxed{\text{DATA bits}}$$

Computing CRCs

- **Data bits:** $s(x) = s_{K-1}x^{K-1} + \cdots + s_1 x + s_0$
- Use a specially chosen function for generating check bits:
  $g(x) = x^L + g_{L-1}x^{L-1} + \cdots + g_1 x + 1$
  Note: $g_L = g_0 = 1$
- Divide $s(x)x^L$ by $g(x)$ and set

$$c(x) = \text{Remainder in division} \left[\frac{s(x)x^L}{g(x)} \right]$$

- **Check bits:** $c(x) = c_{L-1}x^{L-1} + \cdots + c_1 x + c_0$
- **Codeword:**

$$\begin{aligned} y(x) &= s(x)x^L + c(x) \\ &= s_{K-1}x^{L+K-1} + \cdots + s_0 x^L + c_{L-1}x^{L-1} + \cdots + c_0 \end{aligned}$$

Example: CRC (1/2)

- $s(x) = x^2 + 1, g(x) = x^3 + x^2 + 1$
- $c(x) =$ Remainder in division $\left[\frac{(x^2+1)x^3}{x^3+x^2+1}\right]$
- Elementary division gives that $c(x) = x^2 + x$.

	$x^2 + x$				
$x^3 + x^2 + 1$	x^5		$+x^3$		
	x^5	$+x^4$		$+x^2$	
		x^4	$+x^3$	$+x^2$	
		x^4	$+x^3$		$+x$
				$+x^2$	$+x$

- Codeword is $y(x) = s(x)x^3 + c(x) = x^5 + x^3 + x^2 + x$

Example CRC (2/2)

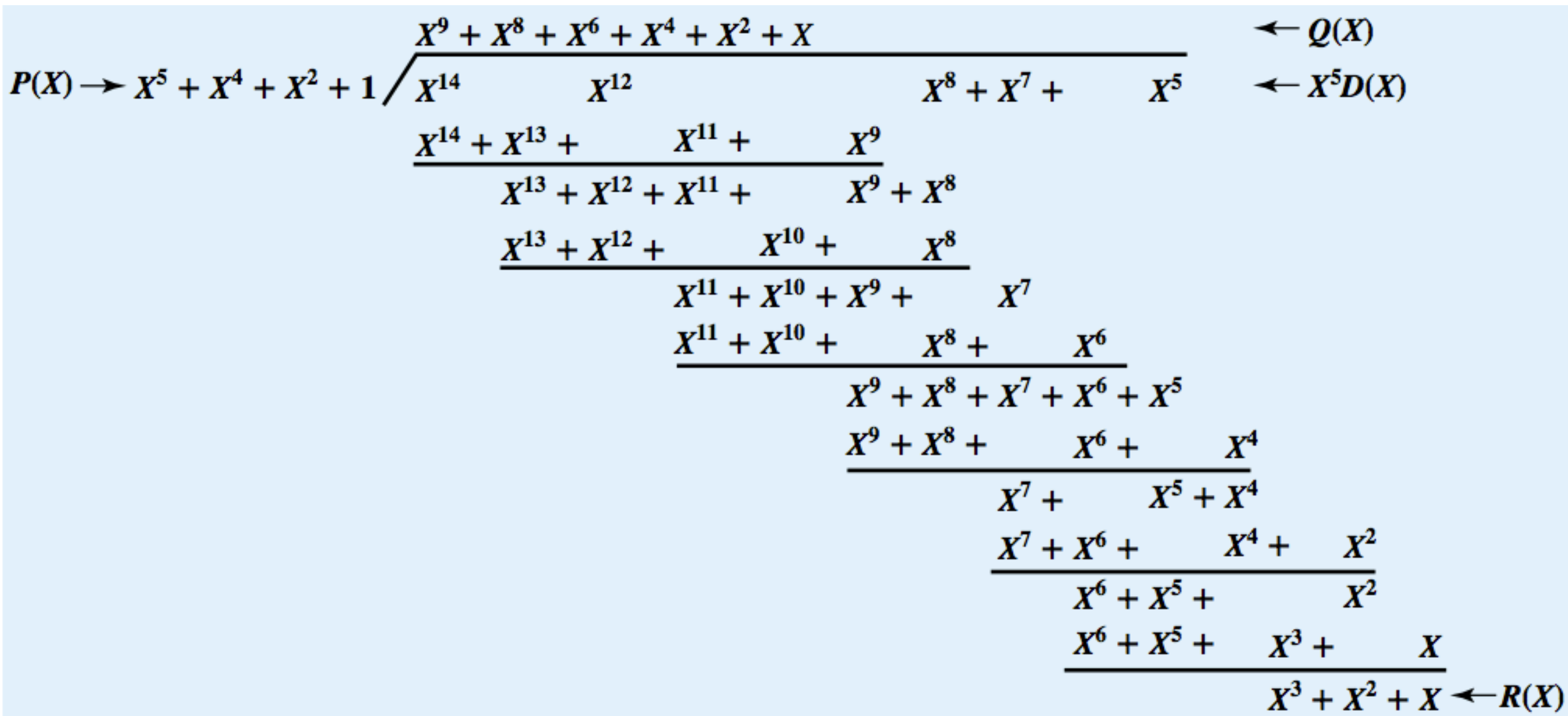
	110 (Quotient)
Generator	Message
$1101 (= x^3 + x^2 + 1)$	$101000 (= x^5 + x^3)$
Addition mod2 $\rightarrow$	1101
	1110
Addition mod2 $\rightarrow$	1101
	0110 (Remainder)
	Check bits

Another Example: Polynomial Division (1/2)

- For $D = 1010001101$ we have
  $D(X) = X^9 + X^7 + X^3 + X^2 + 1$
- For $P = 110101$ we have
  $P(X) = X^5 + X^4 + X^2 + 1$
- When we divide $D(X)$ by $P(X)$ we should end up with
  $R = 01110$, which corresponds to $R(X) = X^3 + X^2 + X$
- Lets check why this is true.

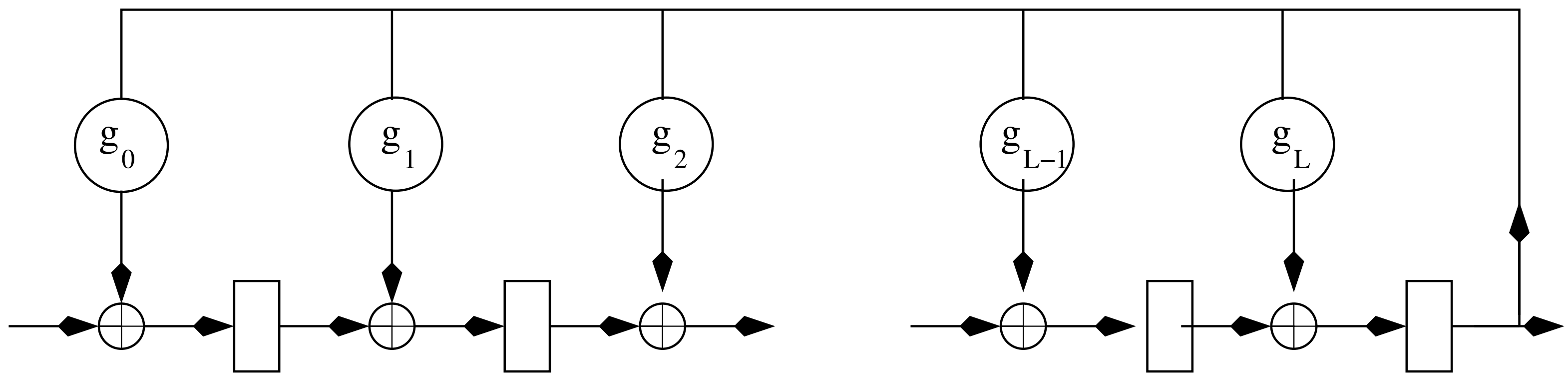
Another Example: Polynomial Division (2/2)

- Let $D(X) = X^9 + X^7 + X^3 + X^2 + 1, P(X) = X^5 + X^4 + X^2 + 1$
- Hence, $X^5 D(X) = X^{14} + X^{12} + X^8 + X^7 + X^5$
- Then the division $\frac{X^5 D(X)}{P(X)} = Q(X) + \frac{R(X)}{P(X)}$ yields



Shift-Register

- Division easily computed in hardware using shift register circuit



- Rectangles store a single bit. The circles $\oplus$ denote XOR gates. Big circles indicate multiplication by g_i.
- Register loaded with L bits: $s_{K-L+1} \cdots s_{K-1} s_K$, with s_K first.
- At each clock pulse a new bit of $s(x)$ comes in at the left. Register reads in corresponding mod2 sum of feedback plus contents of previous stage. After K shifts the switch to the right moves to the horizontal position and the CRC is read out.

Properties of CRC

Why are all code words divisible by $g(x)$, and vice versa?

- **sender** computes the code $c(x)$ from $s(x)$ and transmits $y(x) = s(x)x^L + c(x)$
- Computation is done as follows: Let $z(x)$ be quotient of $s(x)x^L/g(x)$, i.e., $s(x)x^L = g(x)z(x) + c(x)$
- Since subtraction is the same as addition mod 2 we get

$$\begin{aligned} y(x) &= s(x)x^L + c(x) \\ &= s(x)x^L - c(x) \\ &= g(x)z(x) \end{aligned}$$

  Hence, $g(x)$ divides $y(x)$.

- **Recall:** divisibility by $g(x)$ was our error detection condition!

Detecting Single Errors

- Assume receiver gets $w(x) = y(x) + e(x)$, where $e(x)$ represents the errors' polynomial.
- Receiver calculates remainder and if result is zero then accepts, otherwise detects error.
- Can a single error be undetected?
  - Code word $y(x)$ is divisible by $g(x)$.
  - Undetected means $e(x)$ divisible by $g(x)$.
  - Single error implies $e(x) = x^i$ for some i
  - But $g(x)$ has at least two non-zero terms $(x^L, 1)$ and therefore so must $e(x)$.
- It follows that single errors are detected!

Selection of CRCs

- Most important part of implementing the CRC algorithm is the selection of generator polynomial.
- Polynomial is chosen so as to maximize the error-detecting capabilities while minimizing overall collision probabilities.
- Most important attribute is length (largest degree(exponent) +1 of any one term in the polynomial), because of its direct influence on the length of the computed check value.
- Design of the CRC polynomial depends on
  - max length of block to be protected (data + CRC bits),
  - desired error protection features,
  - type of resources for implementing the CRC, and
  - desired performance

Summary of CRCs

Several CRC polynomials have been adopted by the International Telecommunication Union (ITU) and the Consultative Committee for International Telephony and Telegraphy (CCITT).

- CRC-8 (Used in ATM headers): $x^8 + x^2 + x + 1$
- CRC-16 (Used in HDLC): $g(x) = x^{16} + x^{15} + x^2 + 1$
- CRC-CCITT: $g(x) = x^{16} + x^{12} + x^5 + 1$
- CRC-32 (Used in LANs):

$$\begin{aligned} g(x) &= x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + \\ &\quad x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1 \end{aligned}$$

Checksums

Checksum

Sender:

1. Divide data $\boxed{B_1 \cdots B_k}$ into k blocks $B_1, \ldots, B_k$ each of a fixed size n bits (usually $n = 16$).
2. Append to $B_1 \cdots B_k$ the sum S (**checksum**) modulo $(2^n - 1)$.
3. send $\boxed{B_1 \cdots B_k \mid S}$

Receiver:

1. Detach blocks $B_1, \ldots, B_k$ and S from $\boxed{B_1 \cdots B_k \mid S}$
2. Check that

$$\sum_{i=1}^{k} B_i \equiv S \bmod (2^n - 1)$$

Checksum: Example

Let the input have twenty bits and let the block size be $n = 5$.

Input: 01001110110101111110

1. Break into 4 **Blocks:** $\boxed{01001}$ $\boxed{11011}$ $\boxed{01011}$ $\boxed{11110}$

2. Convert each block to integers: $\boxed{9}$ $\boxed{27}$ $\boxed{11}$ $\boxed{30}$

3. Take the sum $\bmod (2^5 - 1)$**:**

$$9 + 27 + 11 + 30 \equiv 77 \equiv 15 \bmod (2^5 - 1)$$

4. Convert to binary (block of 5 **bits):** 01111

Output (append value to input):

$\boxed{01001\ 11011\ 01011\ 11110}$ $\boxed{01111}$

This is the output transmitted by the sender.

Exercises[a]

1. Consider the following two ways to make the correction when discovering that a sequence of bits is in error: 1) Ask for the bits again. 2) Correct them yourself. Discuss advantages and disadvantages.

2. Why is is a small frame header desirable?

3. The Internet needs a point-to-point protocol (PPP) for a variety of purposes, including router-to-router traffic and home user-to-ISP traffic. Discuss some of the available protocols.

4. A bit string, 0111101111101111110, needs to be transmitted at the data link layer. What is the string actually transmitted after bit stuffing?

5. To provide more reliability than a single parity bit can give, an error-detecting coding scheme uses one parity bit for checking

[a] Not to hand in!

all the odd-numbered bits and a second parity bit for all the even-numbered bits. Discuss advantages and disadvantages.

6. What is the remainder obtained by dividing $x^7 + x^5 + 1$ by the generator polynomial $x^3 + 1$?

7. Data link protocols almost always put the CRC in a trailer rather than in a header. Why?