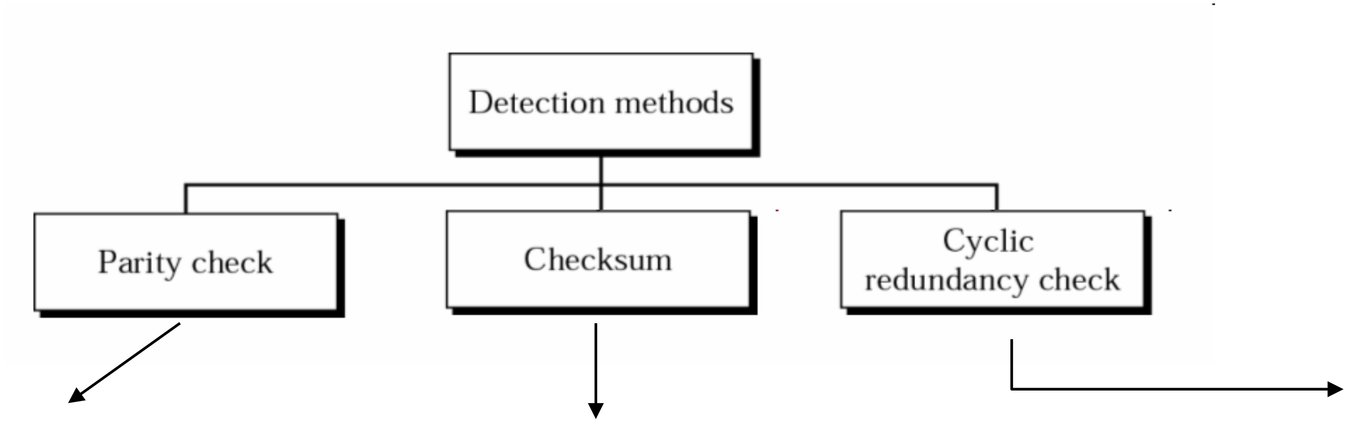
Error Control (2)

Required reading:
Forouzan 10.3
Garcia 3.9.4

CSE 3213, Fall 2015
Instructor: N. Vlajic



Single Parity

- **detects all error involving an odd # of errors**

2-D Parity

- **detects & <u>corrects</u> 1-bit errors**
- **detects all 2- and 3- bit errors**
- **detects some 4-bit errors**

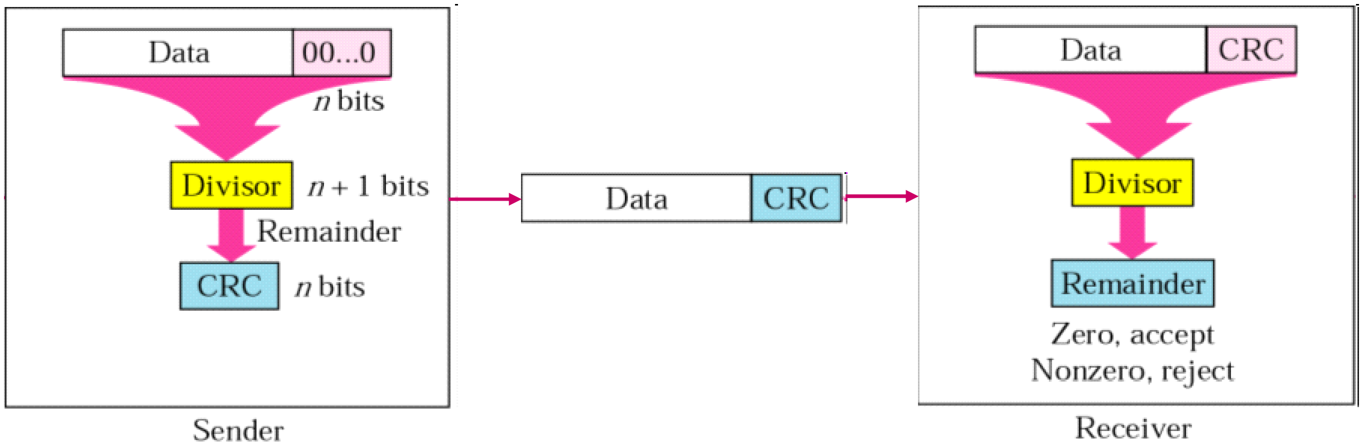
Internet Checksum

- **detects all errors involving an odd # of bits**
- **detects most errors involving an even # of bits**

Cyclic Redundancy Check

CRC – **unlike the parity check and Internet checksum, which are based on binary addition, CRC is based on modulo-2 division**

- **redundancy bits used by CRC are derived by dividing the data unit by a predetermined divisor – the remainder is CRC**
- **CRC is, then, appended to the end of the data unit so that the resulting data unit becomes exactly divisible by the divisor**



CRC Advantages

(1) can be easily implemented in hardware
(2) can detect all single and double errors
(3) very effective in detecting burst errors !!!

Most data communications standards use CRC (polynomial) codes for error detection, e.g. IEEE 802 LAN standards, HDLC, ATM, etc.

Cyclic Redundancy Check (cont.)

Example [CRC overview]

(1) append a string of n 0s to the data unit, where (n+1) is the number of bits in the predetermined divisor (*)

(2) divide the newly elongated data (dividend) by the divisor, using binary modulo 2 division; the remainder resulting from the division is CRC

(3) replace 0s appended in step (1) with CRC – note, CRC may also consist of all 0s

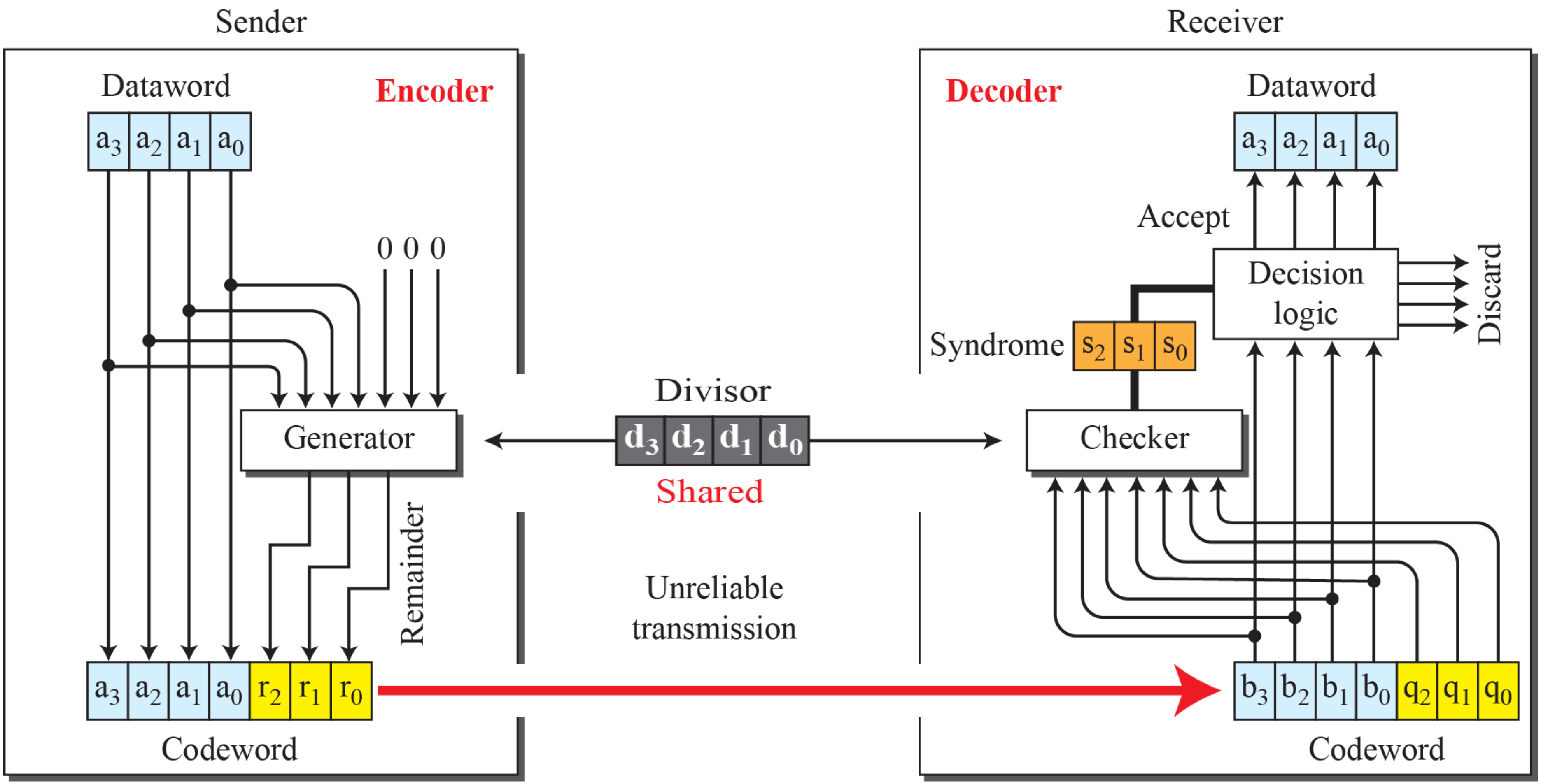
(4) receiver treats the whole string (data+CRC) as a unit and divides it by the same divisor that was used to find CRC

- string arrives with no errors $\Rightarrow$ division yields a zero remainder
- string has been changed in transit $\Rightarrow$ division yields a non-zero remainder

(*) Why should we reserve n bits for CRC?!

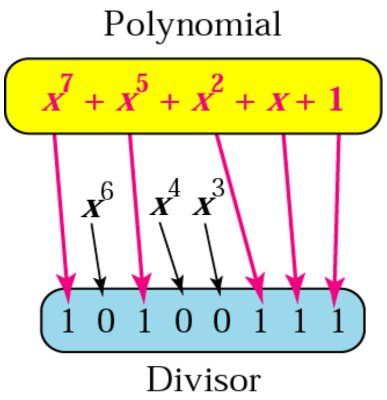
In modulo-2 (binary) division the remainder is always at least one bit shorter than the divisor.

Cyclic Redundancy Check (cont.)



Cyclic Redundancy Check (cont.)

CRC Polynomial Arithmetic – a common way of viewing the CRC process is by expressing all values as polynomials in a dummy variable X, with binary coefficients



- bit pattern $b_m b_{m-1} b_{m-2} \ldots b_1 b_0$ corresponds to

  $b_m X^m + b_{m-1} X^{m-1} + \ldots + b_1 X + b_0$

- e.g. m=7, M=11000011 $\Rightarrow$ $M(X) = X^7 + X^6 + X + 1$
- e.g. m=6, M=1100001 $\Rightarrow$ $M(X) = X^6 + X^5 + 1$
- polynomial format is useful for two reasons:
  - it is short (e.g. 1000000000000010 $\leftrightarrow$ $X^{15} + X$)
  - it can be used to easily prove the concepts mathematically
- polynomial arithmetic:
  - polynomial coefficients can only be 0 or 1
  - operations are performed in 'modulo-2' (exclusive OR) !!!

Cyclic Redundancy Check (cont.)

Example **[polynomial addition, subtraction, multiplication, division]**

Polynomial Addition:

$$(x^7 + x^6 + 1) + (x^6 + x^5) = x^7 + x^6 + x^6 + x^5 + 1$$

$$= x^7 + (1+1)x^6 + x^5 + 1$$

$$= x^7 + x^5 + 1$$

X^6 disappears as (1+1) mod 2 =0

Polynomial Subtraction:

in modulo-2 arithmetic, <u>subtraction is the same as addition</u>

Polynomial Multiplication:

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

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

$$= x^3 + 1$$

Cyclic Redundancy Check (cont.)

Polynomial Division:

The first term of the quotient is chosen so that when we multiply the given term by the divisor, the highest power of the resulting polynomial is the same as the highest power of the dividend.

$$x^3 + x^2 + x \quad = Q(x) \text{ quotient}$$

divisor: $x^3 + x + 1$) $x^6 + x^5$ (dividend)

$$x^6 + x^4 + x^3$$

$$x^5 + x^4 + x^3 \quad \text{(interim remainder)}$$

$$x^5 + x^3 + x^2$$

$$x^4 + x^2$$

$$x^4 + x^2 + x$$

$$x \quad = R(x) \text{ remainder}$$

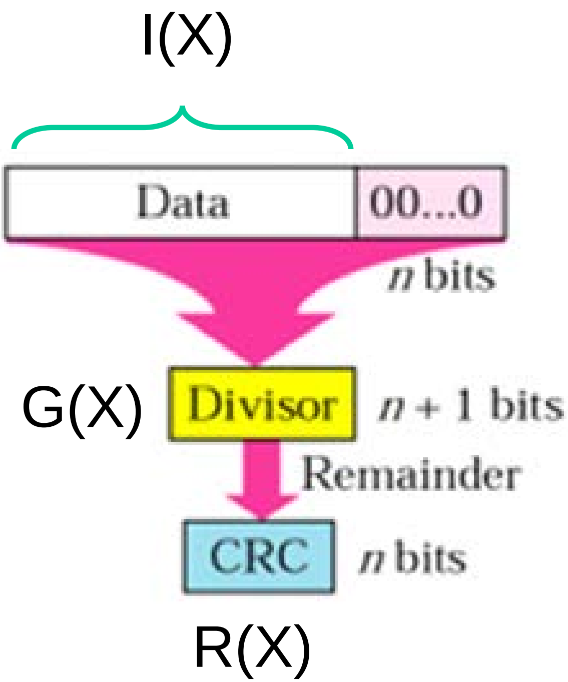
Note: Degree of R(x) is less than degree of divisor.

Cyclic Redundancy Check (cont.)

CRC Polynomial Arithmetic (cont.)

– **assume:**

divisor / generator polynomial:	**G**	**(n+1 bits)**
information:	**I**	**(k bits, k>n)**
CRC remainder:	**R**	**(≤ n bits)**
transmitted frame – I+R:	**B**	**(k+n bits)**



- **CRC process can now be described as:**

step 1) $$\frac{X^n \cdot I(X)}{G(X)} = Q(X) + \frac{R(X)}{G(X)}$$

step 2) $$B(X) = X^n \cdot I(X) + R(X)$$ ← transmitted frame

- **note, from step 2) and 1)**

$$B(X) = X^n \cdot I(X) + R(X) = [G(X) \cdot Q(X) + R(X)] + R(X)$$

and in modulo-2 arithmetic

$$B(X) = G(X) \cdot Q(X)$$

← transmitted frames, i.e. all valid codewords are multiples of the generator polynomial

Cyclic Redundancy Check (cont.)

Example **[CRC polynomial arithmetic]**

Let $G(X) = X^3 + X + 1$. Consider the information sequence 1100.
Find the transmitted codeword corresponding to the given information sequence.

$G(X) = X^3 + X + 1$, $n+1=4$

$I(X) = X^3 + X^2$

$X^3 * I(X) = X^6 + X^5$

$$
\begin{array}{r l}
 & x^3 + x^2 + x \\
\cline{2-2}
x^3 + x + 1 \,) & x^6 + x^5 \\
 & x^6 + \; x^4 + x^3 \\
\cline{2-2}
 & \; x^5 + x^4 + x^3 \\
 & \; x^5 + \; x^3 + x^2 \\
\cline{2-2}
 & \; x^4 + \; x^2 \\
 & \; x^4 + \; x^2 + x \\
\cline{2-2}
 & \; x
\end{array}
$$

$R(X) = X$

Transmitted codeword: $B(X) = X^3 * I(X) + R(X) = X^6 + X^5 + X \leftrightarrow$ **(1,1,0,0,0,1,0)**

Cyclic Redundancy Check (cont.)

Error Detection with CRC Polynomial Arithmetic

– **receiver can check whether there have been any transmission errors by dividing the received polynomial (B'(X)) by G(X)**

- **<u>if there are no errors, remainder = 0</u>**

$$B'(X) = B(X): \qquad \frac{G(X) \cdot Q(X)}{G(X)} = Q(X), \quad \text{no remainder}$$

- **<u>if remainder ≠ 0, an error is (likely) detected</u>**

$$B'(X) = B(X) + E(X): \qquad \frac{G(X) \cdot Q(X) + E(X)}{G(X)} = Q(X) + \frac{E(X)}{G(X)}$$

- **note: if error polynomial E(X) is divisible by G(X), error pattern will be undetectable !!!**

- **design of polynomial codes involves:**
  1) **identifying error polynomials we want to be able to detect**
  2) **synthesizing a generator polynomial that will not divide the given error polynomials without remainder**

Cyclic Redundancy Check (cont.)

Designing Good Polynomial Codes – G(X)

(1) Codes that Detect Single Errors

- codeword of n bits $\Rightarrow$ E_{single} = (0,0,0,1,0,0, …, 0) $\Rightarrow$ $E(X) = X^i$, $0 \le i < n$
- if G(X) has more than one term, and coefficient of X^0 is 1, all single-bit errors can be caught, regardless of bit-i's position

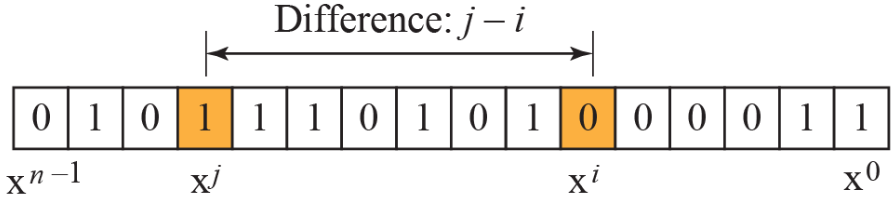
(2) Codes that Detect Double Errors

- codeword n bits $\Rightarrow$ E_{double} = (0,0,0,1,0,1, …, 0) $\Rightarrow$

  $\Rightarrow$ $E(X) = X^i + X^j$, $0 \le i < j \le n$

  $\Rightarrow$ $E(X) = X^i (1 + X^{j-i})$, $0 \le i < j \le n$

- from (1), G(X) is such that it has more than one term & cannot divide X^i $\Rightarrow$ E(X) will be divisible by G(X) only if G(X) divides $(1 + X^{j-i})$ $\Rightarrow$ so we need G(X) that does NOT divide $(1 + X^{j-i})$ without remainder, <u>for any value of i & j</u>



Cyclic Redundancy Check (cont.)

Example **[CRC generators]**

How good are the below CRC generators for detecting two isolated single-bit errors.

a) $X + 1$

Bad choice. Any two errors next to each other will not be detected.

b) $X^4 + 1$

Bad choice. Any two errors 4 bits/positions apart will not be detected.

c) $X^{15} + X + 1$

Good choice, aka PRIMITIVE POLYNOMIAL – cannot be factorized.

Cyclic Redundancy Check (cont.)

Designing Good Polynomial Codes – G(X) (cont.)

(3) Codes that Detect Odd Number of Errors

- we want to make sure that CRC performs as good as single parity check
- **E(X) has an odd number of terms, hence at X=1 $\Rightarrow$ E(1) = 1**
- **G(X) must have a factor (X+1), since there is no polynomial E(X) with an odd number of terms that has (1+X) as a factor**
  - **PROOF: assume such a polynomial, E(X), exists, then**

$$E(X) = (1+X)\,Q(X) \quad \Rightarrow \quad E(1) = (1+1)*Q(1) = 0$$

and this contradicts the fact that E(1) = 1, due to an odd number of terms

- **pick $G(X)=(X+1)*P_{primitive}(X)$ to be able to detect all single, double, & odd-number of errors**

Exercise

1. Which error detection method uses ones complement arithmetic?
   (a) single parity check
   (b) 2-D parity check
   (c) CRC
   (d) checksum

2. In cyclic redundancy checking, the divisor is ______________________ the CRC.
   (a) the same size as
   (b) 1 bit less than
   (c) 1 bit more than
   (d) 2 bits more than

3. In CRC there is no error if the remainder at the receiver is ________________.
   (a) equal to the remainder at the sender
   (b) zero
   (c) nonzero
   (d) the quotient at the sender

4. Which error detection method can detect a burst error?
   (a) the parity check
   (b) 2-D parity check
   (c) CRC
   (d) (b) and (c)