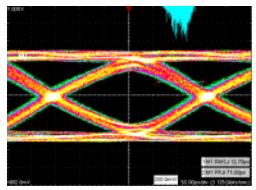
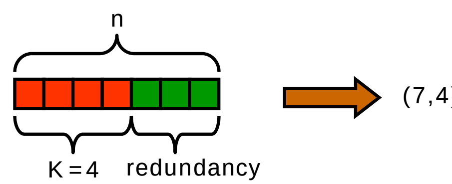
Introduction to Computer Networks

Error Detecting & Correcting Codes



Codes - Notations

- ***K*** bits of data encoded into ***n*** bits of information.
- ***n-K*** bits of redundancy
- The information data is of length ***K***
- The code word is of length ***n***
- (***n***,***k***) code



Parity

- Balances the number of 1-s in a code word
  - Even parity – add 1 to achieve an even number of 1s
    10101 → 101011
    01010 → 010100
  - Odd parity – add 1 to achieve an odd number of 1s
    10101 → 101010
    01010 → 010101
- XOR code word by bit to get even parity
  - Odd parity is NOT of the result
- (K+1, K)
- ***Detects*** all 1-bit errors

Two-Dimensional Bit Parity

- *(p·l + p + l +1, p·l)*
  - *K = p·l*
  - *n= p·l + p + l +1*
- *Can catch:*
  - *All 1-,2-,3- bit errors*
  - *Most 4-bit errors*

10110001 11000010 01101000 00110101 01 001100

↓

10110001	0
11000010	1
01101000	1
00110101	0
01001100	1
01100010	1

Hamming Codes

- **Hamming codes** are a family of linear error-correcting codes that generalize the Hamming(7,4)-code invented by Richard Hamming in 1950.
- Hamming codes can **detect up to two and correct up to one bit errors**.
- Hamming Distance – The number of positions in which 2 words differ.
  - 100101, 101001 – distance of 2
- Code word of (n,K,d)
  - Error Detection: d-1
  - Error Correction: (d-1)/2
-

Hamming Codes

- Code rate: K/n
- Hamming was interested in optimizing two parameters at once; increasing the distance as much as possible, while at the same time increasing the code rate as much as possible.
- Hamming codes are special in that they are perfect codes, that is, they achieve the highest possible rate for codes with their block length and minimum distance 3.

Hamming Codes - Construction

- Number bits from 1 and upwards
- A bit which is a power of 2 is a check bit
  - 1, 2, 4, 8….
- All other bits are data bits
  - 3, 5, 6, 7, 9, 10….
- Each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.
  - example : Bit 1 = 001

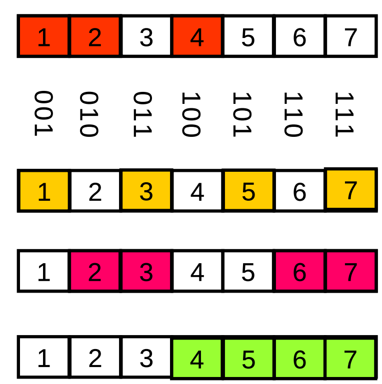
    bit 2 AND bit 1 = 001 & 010 =000

    bit 3 AND bit 1 = 001 & 011 = 001

    ….

    $\Rightarrow$ bit 1 = bit 3 $\oplus$ bit 5 $\oplus$ bit 7

Hamming Codes - Construction



Hamming Codes - Construction

- Use Generating Matrix (G) and Parity Check Matrix (H).
  - G· x =y
  - H· y = s
  - s is a null vector iff y is a code word, i.e. no parity error.
  - If s is not null, it indicates which bit had an error

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Hamming Codes - Construction

- Example: error in bit 5

$$x = (1 \quad 0 \quad 1 \quad 1)$$

$$G \cdot x = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = y$$

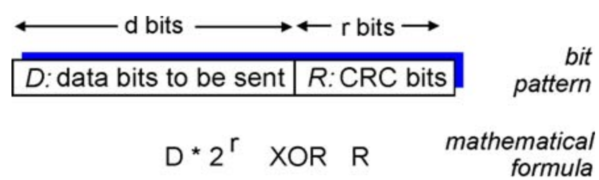
$$H \cdot \left(y + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Hamming Code (7,4)

Data	Codeword	Data	Codeword
0000	0000000	1000	1110000
0001	1101001	1001	0011001
0010	0101010	1010	1011010
0011	1000011	1011	0110011
0100	1001100	1100	0111100
0101	0100101	1101	1010101
0110	1100110	1110	0010110
0111	0001111	1111	1111111

CRC – Cyclic Redundancy Check

- View data bits, D, as a binary number
- Choose r+1 bit pattern (generator), G
- Goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - Receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - Can detect all burst errors less than r+1 bits



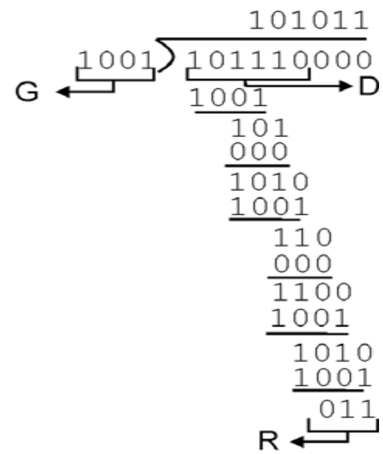
Polynomial Arithmetic Modulo 2

- B(x) can be divided by a divisor C(x) if B(x) is of higher degree.
- B(x) can be divided once by a divisor C(x) if B(x) is of the same degree as C(x).
- The reminder of B(x)/C(x) is obtained by subtracting C(x) from B(x).
- Subtracting is Simply the XOR operation

CRC - Example

- G – Generator – x^3+1
- D – $x^5+x^3+x^2+x^1$ – 101110

$$R = \text{remainder}\left[\frac{D \cdot 2^r}{G}\right]$$



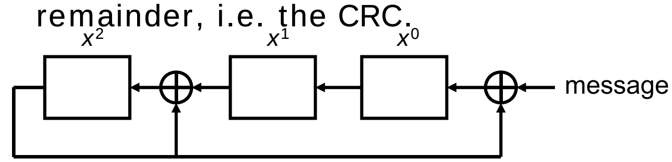
CRC Implementation in Hardware

- CRC algorithm is easily implemented in hardware using r-bit shift register and XOR gates.

- B(x) can be divided once by a divisor C(x) if B(x) is of the same degree as C(x).
- The reminder of B(x)/C(x) is obtained by subtracting C(x) from B(x).
- Subtracting is Simply the XOR operation

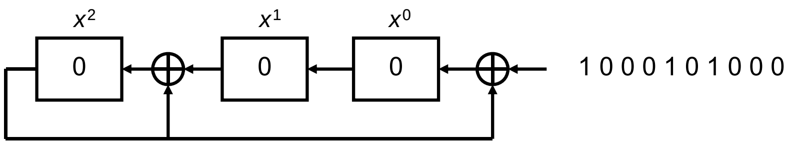
CRC Implementation in Hardware

- CRC algorithm is easily implemented in hardware using r-bit shift register and XOR gates.
- Put an XOR gate in front of bit n, if there is a term x^n in the generator polynomial.
- Below is the hardware that should be used for the generator $x^3 + x^2 + 1$.
- The message is shifted from right to left, beginning with msb and ending with r zeros.
- When all the bits have been shifted in and appropriately XORed, the register contains the remainder, i.e. the CRC.



CRC

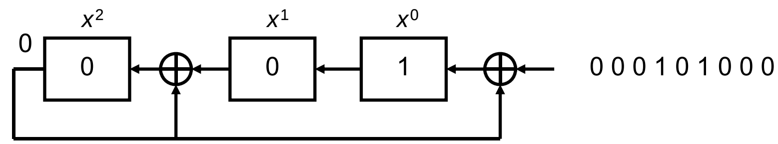
- Here's a picture for the start of the division of $(x^6 + x^2 + 1)$ divided by $(x^3 + x^2 + 1)$



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

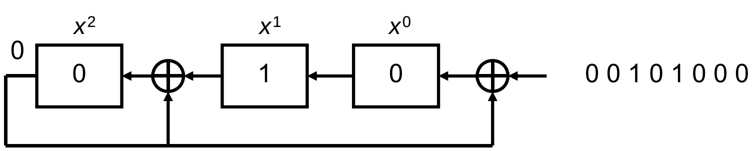
- Here's a picture for the start of the division of $(x^6 + x^2 + 1)$ divided by $(x^3 + x^2 + 1)$



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

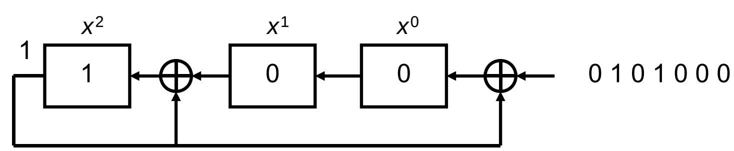
- Here's a picture for the start of the division of $(x^6 + x^2 + 1)$ divided by $(x^3 + x^2 + 1)$



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

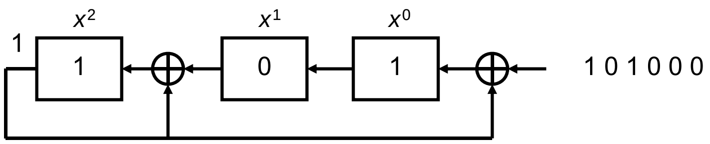
- Here's a picture for the start of the division of $(x^6 + x^2 + 1)$ divided by $(x^3 + x^2 + 1)$



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

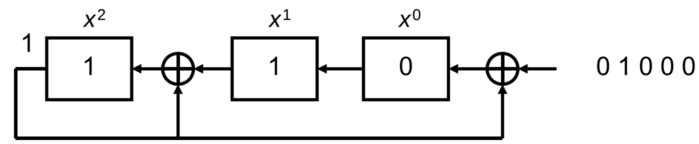
- Here's a picture for the start of the division of $(x^6 + x^2 + 1)$ divided by $(x^3 + x^2 + 1)$



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

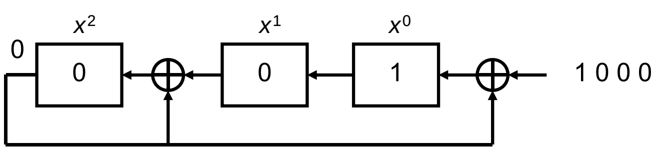
- Here's a picture for the start of the division of $(x^6 + x^2 + 1)$ divided by $(x^3 + x^2 + 1)$



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

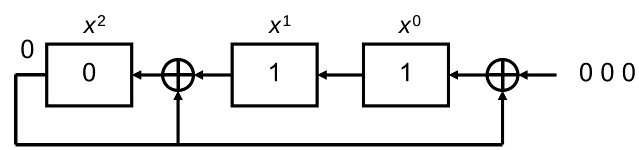
- Here's a picture for the start of the division of $(x^6 + x^2 + 1)$ divided by $(x^3 + x^2 + 1)$



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

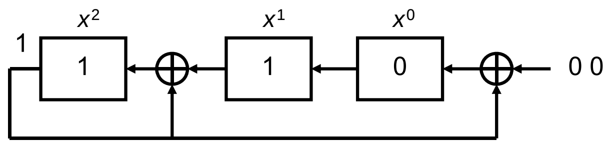
- Here's a picture for the start of the division of $(x^6 + x^2 + 1)$ divided by $(x^3 + x^2 + 1)$



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

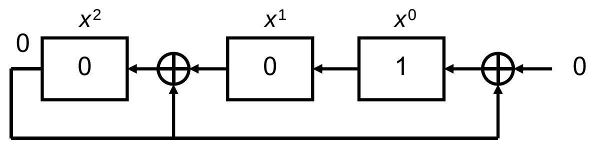
- Here's a picture for the start of the division of ($x^6 + x^2 + 1$) divided by ($x^3 + x^2 + 1$)



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

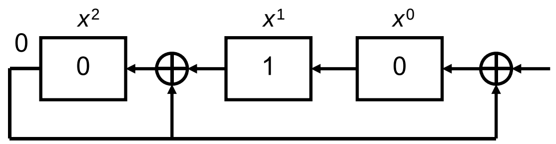
- Here's a picture for the start of the division of ($x^6 + x^2 + 1$) divided by ($x^3 + x^2 + 1$)



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input

CRC

- Here's a picture for the start of the division of ($x^6 + x^2 + 1$) divided by ($x^3 + x^2 + 1$)



- Let's do the steps. I'll list the output bit, the values in the boxes, and the remaining input