Lecture 7

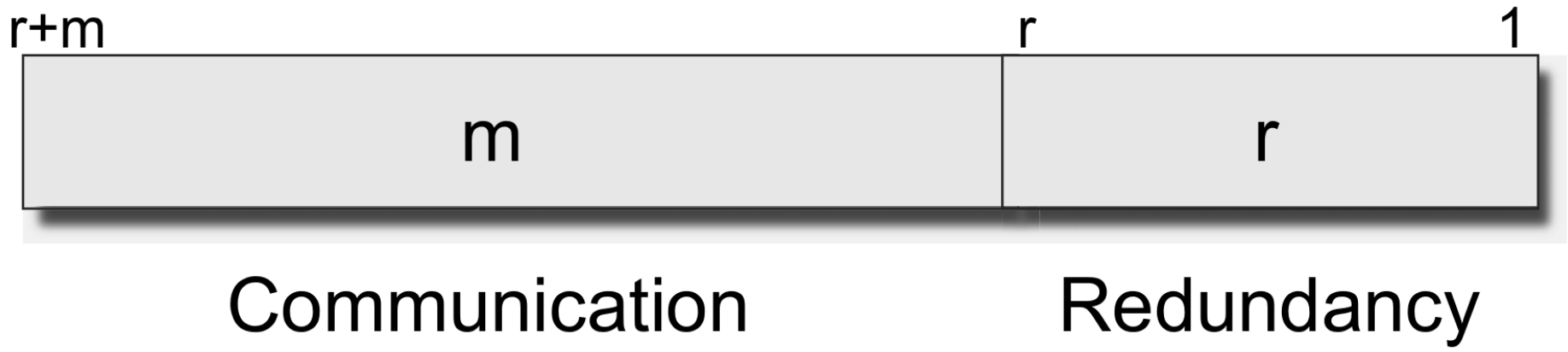
Communication Fundamentals in Computer Networks

M. Adnan Quaium

Assistant Professor
Department of Electrical and Electronic Engineering
Ahsanullah University of Science and Technology
Room – 4A07
Email – adnan.eee@aust.edu
URL- http://adnan.quaium.com/aust/cse4295

Codeword

The **codeword** *(message)* itself has a length of **n=m+r** bits and consists of the payload to be transmitted *(the communication)* and extra bits that enable the detection of a transmission error *(the redundancy)*.



Hamming Distance

Codeword $\mathbf{C_1}$ 1000100010001000
Codeword $\mathbf{C_2}$ 1000000011000000

These two codewords differ in exactly three positions.

The number of positions that differ between the two codewords a and b is referred to as the *Hamming Distance* ***H(a,b)*** of the two codewords.

If two codewords differ by *n bits*, then exactly *n single bit* errors can transform the one codeword into the other.

Hamming Distance

In order to determine the Hamming distance of two given codewords, they are linked bit by bit with the logical XOR-Operator and sum up the number of ones in the result of this operation:

$$H(a,b) = \sum_{i=0}^{n-1} a_i \oplus b_i$$

In our example, there is a Hamming distance of H(a,b)=3:

$$\begin{array}{rr} a & 1000100010001000 \\ b & 1000000011000000 \\ \hline a \oplus b & 0000100001001000 \end{array}$$

Parity Bit

The parity bit corresponds to the parity of the 1-bit in the codeword.

If the number is even then the parity bit is equal to 0, otherwise it is equal to 1.

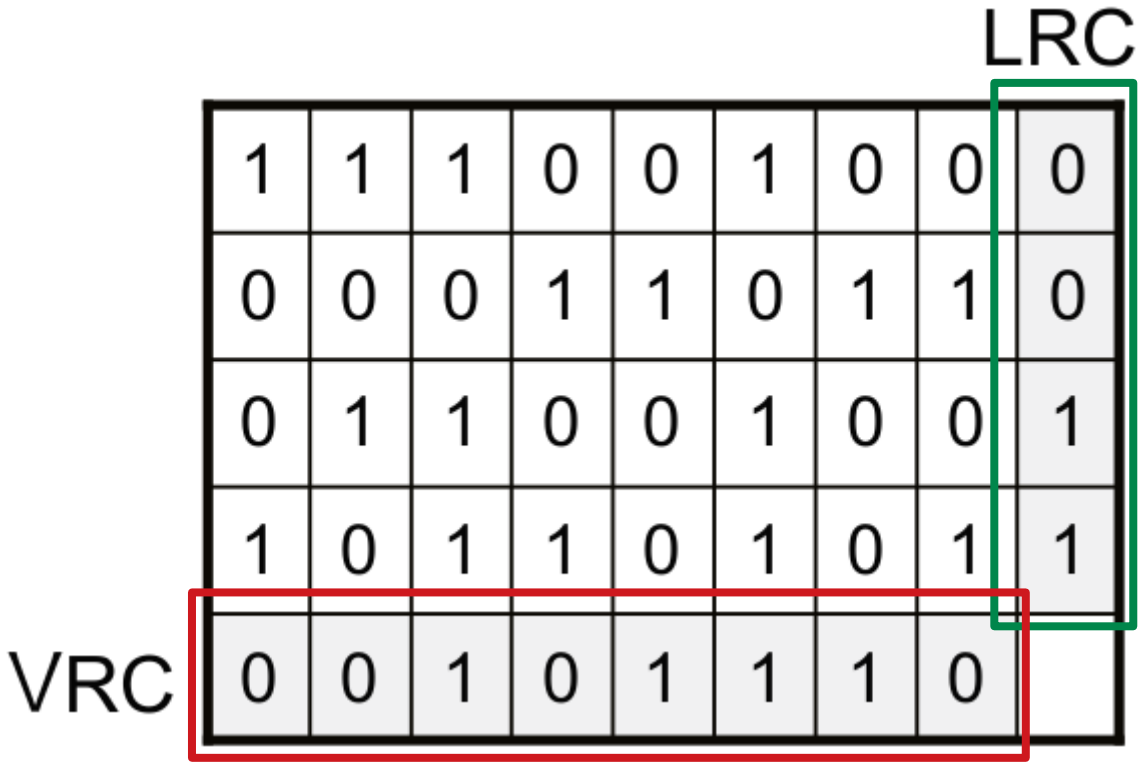
For a=1000100010001000 e.g., the number of 1-bits is equal to 4, a – the parity is therefore even.

a is added with the parity bit p=0, the code actually transmitted is a'=0100010001000100|0.

Parity Bit

The codes in which the message remains unchanged as a block, with the check bits simply appended, are referred to as systematic block codes.

If the bits of multiple code words are summarized together into a matrix, in addition to horizontally determined parity bits, *(Longitudinal Redundancy Check, LRC)* parity is also defined in columns (Vertical Redundancy Check, VRC).

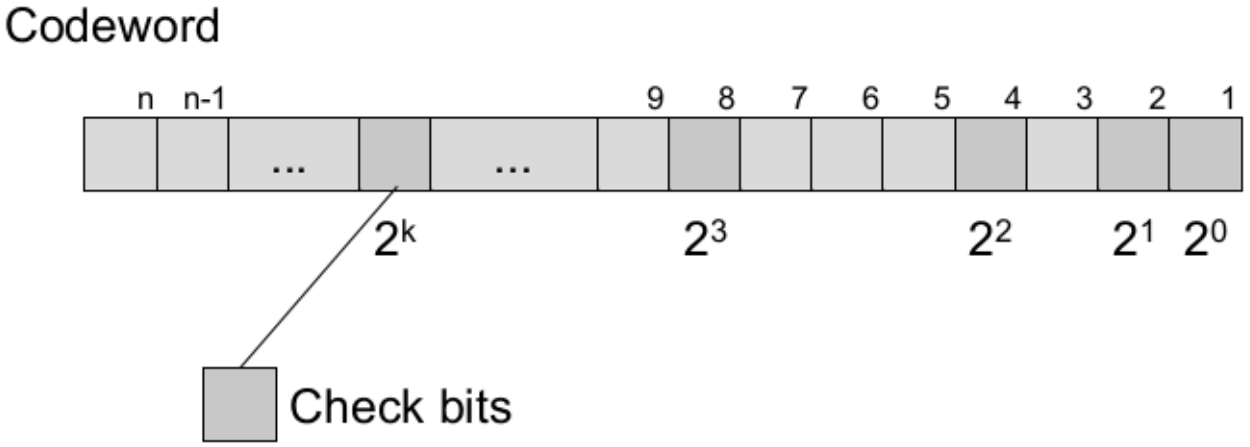


Hamming Code

The Hamming Code is a code that follows this pattern and detects all individual bit errors.

- All bits of the codeword are numbered *starting with 1*.
- Bits numbered with a power of two (1,2,4,8,16, etc.) are implemented as ***Check bits***, while the remaining bits (3,5,6,7,9,10, etc.) are filled with the m bits of the message.
- Every check bit now stands for the parity of a number of individual bits.

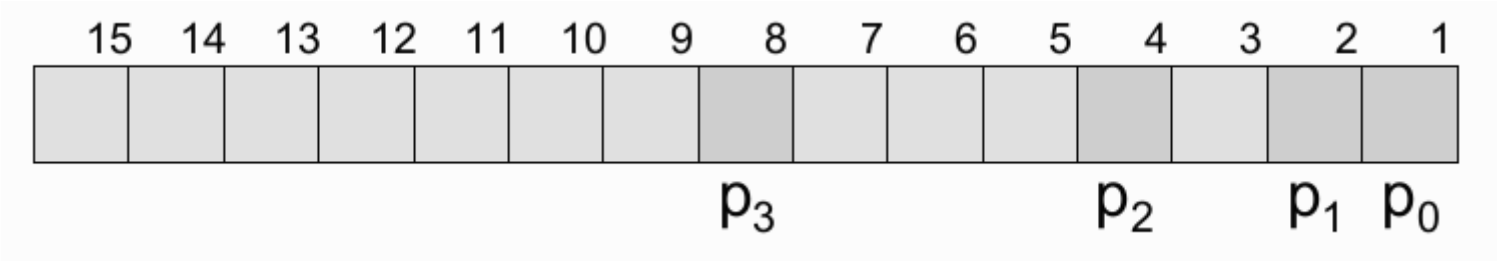
The data bit at position k, $1 \leq k \leq n$, is allocated to the check bits contained in the binary encoding of k. For example, if k=11, then $k=1+2+8=2^0+2^1+2^3$ and the k-th bit enters in the parity calculation of check bits 2^0, 2^1, and 2^3.



Hamming Code

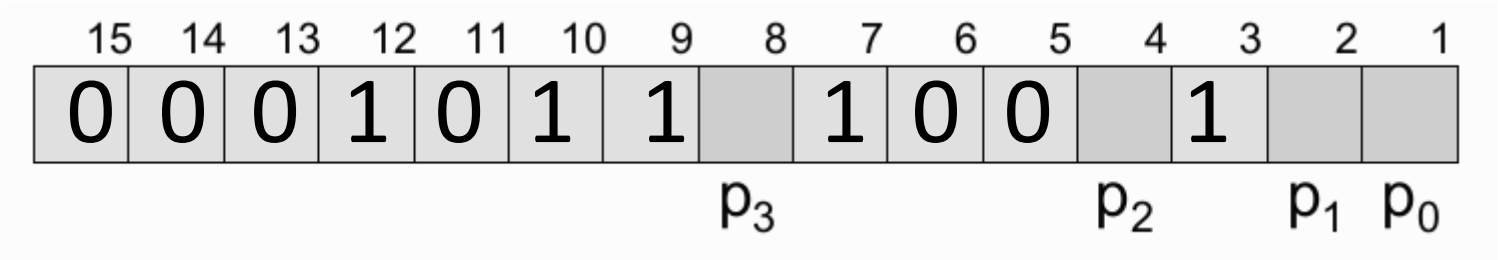
For example, let us consider a Hamming code for transmitting codewords each made up of m=11 data bits and r=4 check bits.

This code made up of n=15 bit long codewords is also known as 15/11 Hamming code.



Hamming Code

Let us assume that the following bit sequence is to be transmitted: 00010111001.



Hamming Code

Each data bit is included in a unique set of check bits, as determined its bit position in binary form.

- Check bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).
- Check bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc).
- Check bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).
- Check bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc).

$$p_0 = c_3 \oplus c_5 \oplus c_7 \oplus c_9 \oplus c_{11} \oplus c_{13} \oplus c_{15}$$
$$p_1 = c_3 \oplus c_6 \oplus c_7 \oplus c_{1o} \oplus c_{11} \oplus c_{14} \oplus c_{15}$$
$$p_2 = c_5 \oplus c_6 \oplus c_7 \oplus c_{12} \oplus c_{13} \oplus c_{14} \oplus c_{15}$$
$$p_3 = c_9 \oplus c_{10} \oplus c_{11} \oplus c_{12} \oplus c_{13} \oplus c_{14} \oplus c_{15}$$

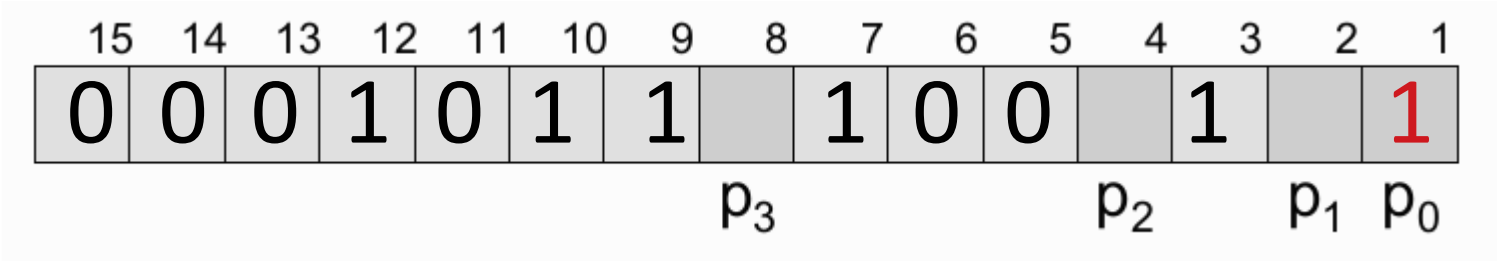
Hamming Code

To illustrate the interaction of individual check bits (position 1, 2, 4 and 8), they are shown in tabular form.

Pos	15	14	13	12	11	10	9	**8**	7	6	5	**4**	3	**2**	**1**	Parity
p_0	x		x		x		x		x		x		x		**x**	0
p_1	x	x			x	x			x	x			x	**x**		0
p_2	x	x	x	x					x	x	x	**x**				0
p_3	x	x	x	x	x	x	x	**x**								0

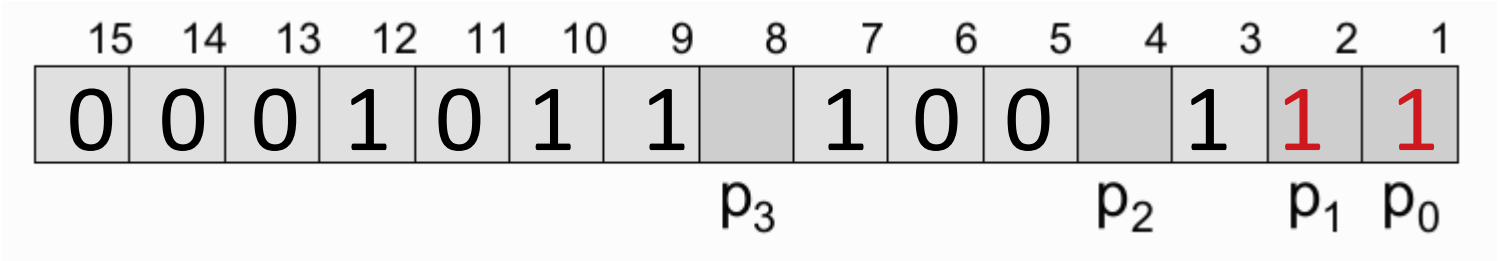
Hamming Code

Check bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).



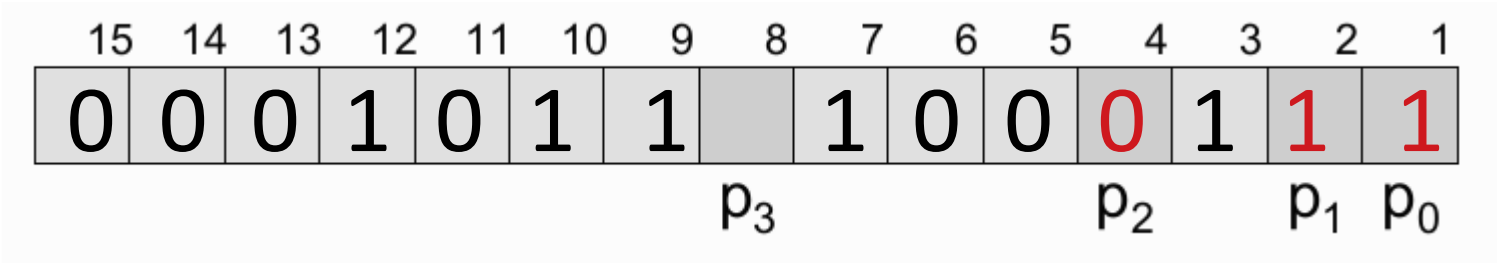
Hamming Code

Check bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc).



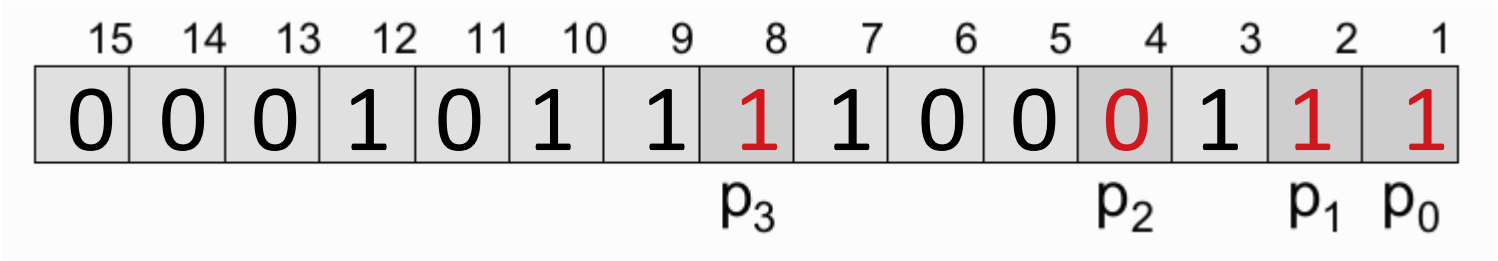
Hamming Code

Check bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).



Hamming Code

Check bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc)



Hamming Code

So the transmitted bit sequence is:

00010111001

Together with the calculated check bit p_0 - p_3 the final codeword is:

00010111100111

Pos	15	14	13	12	11	10	9	**8**	7	6	5	**4**	3	**2**	**1**	Parity
C	0	0	0	1	0	1	1	1	1	0	0	0	1	1	1	
p_0	0		0		0		1		1		0		1		**1**	0
p_1	0	0			0	1			1	0			1	**1**		0
p_2	0	0	0	1					1	0	0	**0**				0
p_3	0	0	0	1	0	1	1	**1**								0

Hamming Code

Assume that during data transmission an error has occurred in position 7, and that c_7 is inverted. This results in the incorrect code:

00010111<u>0</u>000111

Pos	15	14	13	12	11	10	9	**8**	7	6	5	**4**	3	**2**	**1**	Parity
C	0	0	0	1	0	1	1	1	**0**	0	0	0	1	1	1	
p_0	0		0		0		1		0		0		1		**1**	**1 F**
p_1	0	0			0	1			0	0			1	**1**		**1–F**
p_2	0	0	0	1					0	0	0	**0**				**1–F**
p_3	0	0	0	1	0	1	1	**1**								0–ok

Hamming Code

The parity check for p_0 - p_2 results respectively in the incorrect value 1, only p_3 is calculated correctly.

Only the place c_7 as single bit error is capable of distorting calculation of the check bit p_0 - p_2 . Adding the powers of the check bit index ($2^2+2^1+2^0$ = 4+2+1 = 7), gives the results of the incorrect place.

In this way, every bit error of the code can be detected and corrected.

Pos	15	14	13	12	11	10	9	**8**	7	6	5	**4**	3	**2**	**1**	**Parity**
C	0	0	0	1	0	1	1	1	**0**	0	0	0	1	1	1	
p_0	0		0		0		1		0		0		1		**1**	**1 F**
p_1	0	0			0	1			0	0			1	**1**		**1–F**
p_2	0	0	0	1					0	0	0	**0**				**1–F**
p_3	0	0	0	1	0	1	1	**1**								0–ok

Hamming Code

Hamming code is only viable for short codewords so that today different methods such as matrix checksums are used.

Checksum

Checksum procedures are used, for example, in the Internet protocol (IP). The most well-known procedure is the so-called **Cyclic Redundancy Check** (CRC), also called **Polynomial Code**.

The fundamental concept of the CRC procedure is representing the transmitted bits of the message as coefficient u_i, $0 \leq i \leq m-1$, of a polynomial, which could be either 0 or 1. The payload bits of the message are interpreted as follows:

$$M(x) = u_{m-1}x^{m-1} + u_{m-2}x^{m-2} + \ldots + u_1 x + u_0.$$

The polynomial M(x) is of degree m-1. The message 11000101 has e.g., exactly 8 bits and generates the polynomial $M(x)=x^7+x^6+x^2+1$.

CRC

Checksum procedures are used, for example, in the Internet protocol (IP). The most well-known procedure is the so-called **Cyclic Redundancy Check** (CRC), also called **Polynomial Code**.

The fundamental concept of the CRC procedure is representing the transmitted bits of the message as coefficient u_i, $0 \leq i \leq m\text{-}1$, of a polynomial, which could be either 0 or 1. The payload bits of the message are interpreted as follows:

$$M(x) = u_{m-1}x^{m-1} + u_{m-2}x^{m-2} + \ldots + u_1 x + u_0.$$

The polynomial M(x) is of degree m-1. The message 11000101 has e.g., exactly 8 bits and generates the polynomial $M(x) = x^7 + x^6 + x^2 + 1$.

CRC

M(x) is then divided by a **generator polynomial** G(x), used both by the sender and the receiver and the resulting remainder constitutes the block checksum to be appended. The generator polynomial is of degree r, the first and the last bits must not equal to zero, $g_r, g_0 \neq 0$:

$$G(x) = g_r x^r + g_{r-1} x^{r-1} + \ldots + g_1 x + g_0.$$

Now r zero bits are appended to the message, which corresponds to the polynomial $x^r M(x)$. $x^r M(x)$ is divided by G(x) using the calculation rules for finite fields. This results in a remainder polynomial R(x), which is r-1 degree at the most. The coefficients of R(x), $r_{r-1}, \ldots, r_0$, are appended to the message. Thus, the transmitted message corresponds to the polynomial $N(x) = x^r M(x) - R(x)$.

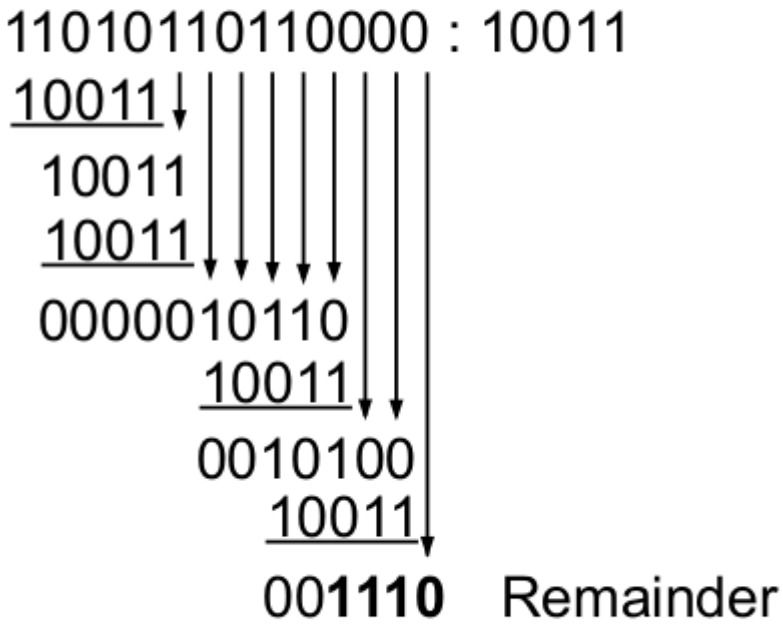
CRC

Message: 1101011011

Generator: k= 4, $G(x) = x^4 + x^1 + 1$, (10011)

Multiply message with x^4 : 1101011011 0000

Determination of remainder:



Codeword for transmission: 1101011011 1110

CRC

This polynomial is now divisible by G(x). If N(x) is transmitted without error, the receiver calculates N(x)/G(x) and gets the remainder 0. It is clear that N(x) is divisible in any case by G(x), because for any division problem it holds true that if one subtracts the remainder from the dividend, the result of the subtraction is always divisible by the divisor.

CRC

The most common standard CRC polynomials are:

$$
\begin{aligned}
\text{CRC-12} &= x^{12} + x^{11} + x^3 + x^2 + x + 1 \\
\text{CRC-16} &= x^{16} + x^{15} + x^2 + 1 \\
\text{CRC-CCITT} &= x^{16} + x^{12} + x^5 + 1 \\
\text{CRC-32} &= x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + \\
&\quad + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1
\end{aligned}
$$

CRC-12 is used for the transmission of 6-bit characters and generates a 12-bit block test sequence. CRC-16 and CCRC-CCITT are both used for 8-bit transmissions and generate a 16-bit block test sequence. Applications requiring a higher transmission security can turn to CRC-32. It generates a 32-bit block test sequence. CRC-32 is used, for example, in standard transmission technology (Ethernet, FDDI, IEEE-802). With CRC-16, or CRC-CCITT, all single bit errors as well as all doubled single bit errors and all errors of odd length are detected. Moreover, 100% of all errors are detected that are shorter than 16 bits, 99.997 % of all 17-bit burst errors and 99.998% of all burst errors of 18 bit length or more. The CRC algorithm may seem relatively complex, it can however be implemented quite easily in hardware by means of a simple sliding register operation.

Error Correction Method

*In codes for error correction a distinction is made between so-called **Block Codes** and **Convolutional Codes**.*

- Just as in error detection, also here the payload data of redundant information is added.
- The quotient m/n is referred to as the code rate, whereby n=m+r. Commonly used values are 1/2, 3/4 and 7/8.

The processing overhead for error-correcting codes is relatively high, with a correspondingly low code rate. This low code rate is responsible for the methods of error correction only being used where a re-transmission is not practical.

- An important example are terrestrial radio systems (*GSM*: Global System for Mobile Communication).