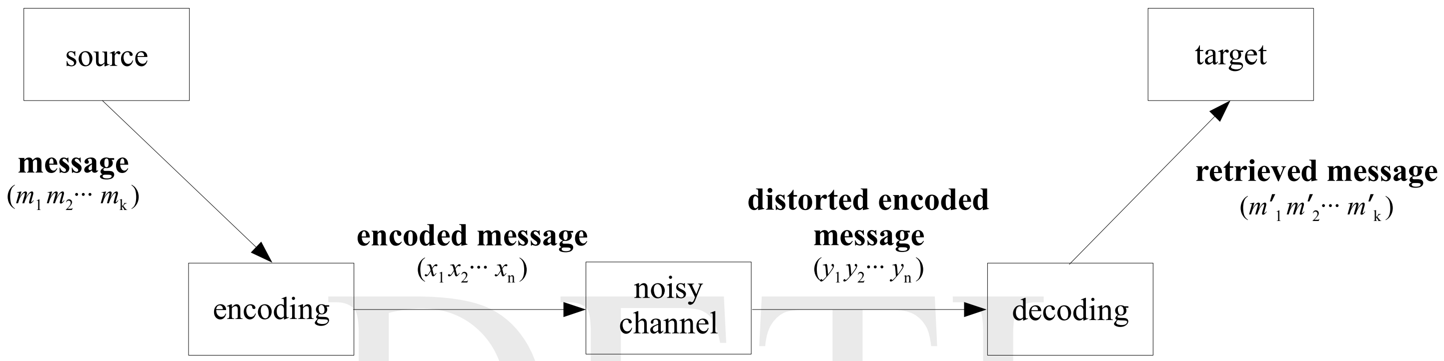
Arquitecturas de Alto Desempenho

Design Principles for Hamming Codes

António Rui Borges

Theoretical Background - 1



A message $\boldsymbol{m}$, generated by a given *source*, is expressed in k symbols of the alphabet Σ and is further encoded into a word $\boldsymbol{x}$, using a *block code*, through expansion to length n by addition of redundant information; that is, a specific code word is built by the interspersing of symbols of the alphabet Σ to the symbols of the message, following some definite rule.

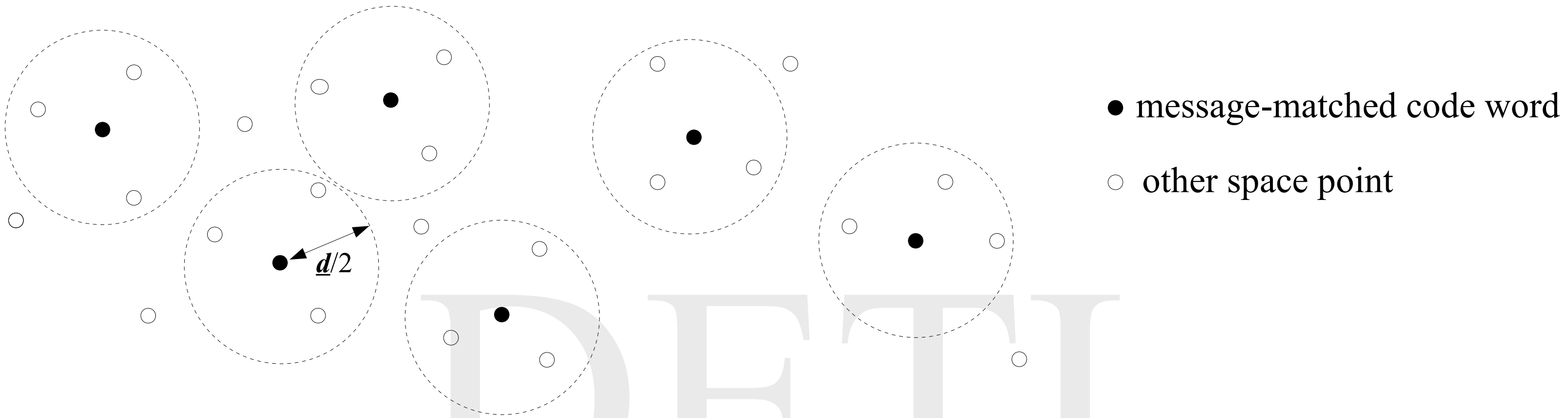
The encoded message $\boldsymbol{x}$ is next transmitted over a *noisy channel*, where the symbols may be changed according to certain probabilities that are characteristic of the channel. The received message $\boldsymbol{y}$ is finally decoded into the message $\boldsymbol{m'}$, which is retrieved by the *target*. Bear in mind that the *channel* concept must be taken in a broad sense: it can be a data transmitting medium in the traditional sense, such as copper, optical fiber, or air signal propagation, or a data storage device, such as a hard disk, a flash memory, or a dynamic RAM.

Theoretical Background - 2

One can define the *information rate* R, which measures the slowdown of the effective data transmission, as k/n. On the other hand, given the channel characteristics, one defines the *capacity* C *of the channel* as something which, as Claude Shannon has shown, has the property that, for $R < C$, it is possible to find an *encoding*/*decoding* scheme such that the probability that $\boldsymbol{m} \neq \boldsymbol{m}'$ can be made arbitrarily small. If, however, $R > C$, no such scheme exists.

Claude Shannon, however, did not show how to build such *encoding*/*decoding* schemes. This has been the pioneering work of Richard Hamming. Presently, however, the case for the best codes in terms of the maximal number of errors that one can correct for a given information rate and code length is not clear. Existence theorems are known, but the exact bound is still an open problem.

Theoretical Background - 3



To understand how one can achieve an *error-correcting* code by increasing the number of symbols in the code word relative to the message word, one may think of each code word as a point in a n-dimensional space where a *metric* is defined, that is, a real-valued function $d(\boldsymbol{x},\boldsymbol{y})$ which, given two arbitrary space points, $\boldsymbol{x}$ and $\boldsymbol{y}$, computes the distance between them.

Now, if one can find an *encoding* scheme which matches each message word to code words that are as widely-spaced as possible within the n-dimensional space, then all space points in the neighborhood of radius $\underline{\boldsymbol{d}}/2$ of each message-matched code word, where $\underline{\boldsymbol{d}}$ represents the minimum value of the distance between two arbitrary message-matched code words, can be *decoded* as matching the corresponding message word through the use of a minimum distance criterion.

Hamming codes - 1

Hamming codes are able to detect and correct one symbol of the received message. The metric, $d(\boldsymbol{x},\boldsymbol{y})$ is here defined as the number of positions where the symbols of the code words $\boldsymbol{x}$ and $\boldsymbol{y}$ differ, the so-called *Hamming distance*.

In the binary case, the alphabet Σ is the set $\{0,1\}$ and the space of code words is F_2^n, where F_2 is the Galois field of two elements. By taking the number of redundant symbols, *parity bits*, to be r, one gets

$$n = 2^r - 1 \quad \text{for the code length}$$
$$k = n - r \quad \text{for the message length}$$

giving rise to a $[n,k]$ block code, with a minimum distance between code words of 3.

Hamming codes - 2

The encoding process is described by

$$\boldsymbol{x} = \boldsymbol{m} \cdot \boldsymbol{G}$$

where $\boldsymbol{m}$ is the message, $\boldsymbol{x}$ its code word and $\boldsymbol{G}$ the kxn code generating matrix, defined as

$$\boldsymbol{G} = \| \boldsymbol{I}_k \mid -\boldsymbol{A}^T \| \quad .$$

The *decoding* process, on the other hand, is described by

$\boldsymbol{H} \cdot \boldsymbol{y}^T = \boldsymbol{0} \quad \Rightarrow$ message bits of $\boldsymbol{y}$ are correct (there is no error)
$\boldsymbol{H} \cdot \boldsymbol{y}^T \neq \boldsymbol{0} \quad \Rightarrow$ error on bit whose position is the column number whose contents is $\boldsymbol{H} \cdot \boldsymbol{y}^T$

where $\boldsymbol{y}$ is the received encoded message and $\boldsymbol{H}$ the rxn parity check matrix, defined as

$$\boldsymbol{H} = \| \boldsymbol{A} \mid \boldsymbol{I}_r \|$$

and has as columns all the pairwise linear independent vectors of length r.

Computation of a parity-check matrix for [15,11]

$$\boldsymbol{H} = \left\| \boldsymbol{A} \mid \boldsymbol{I}_4 \right\| =$$

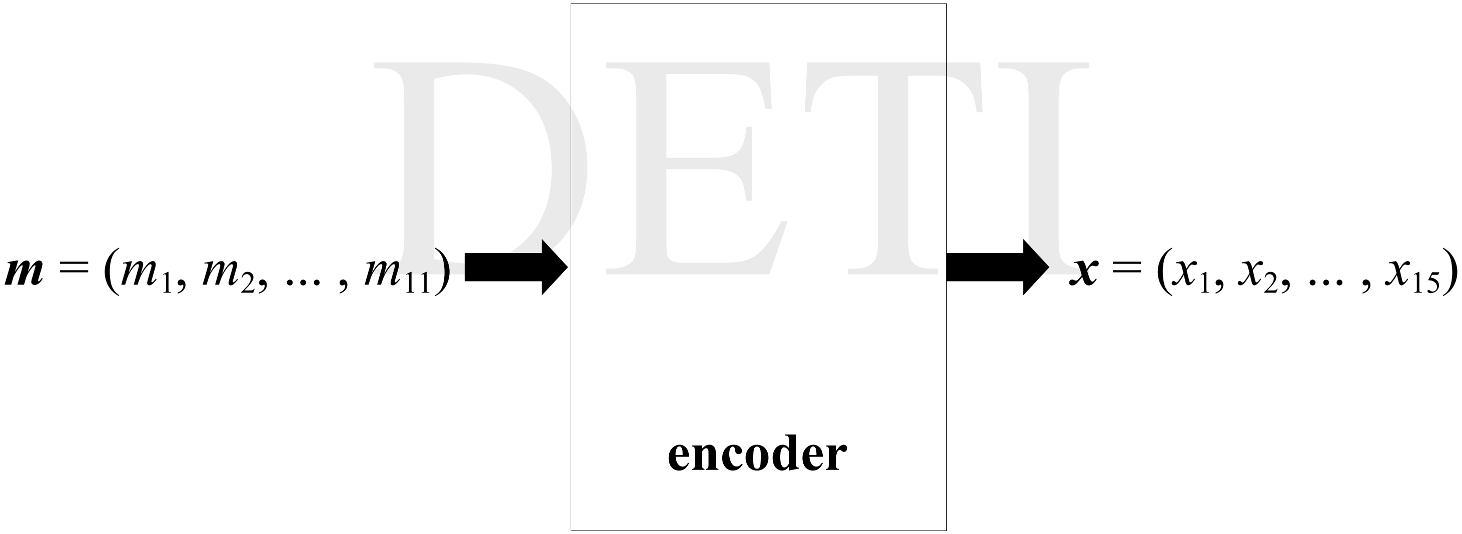
$$= \left\| \underbrace{\begin{matrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{matrix}}_{\boldsymbol{A}} \;\; \underbrace{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}}_{\boldsymbol{I}_4} \right\|$$

Computation of the associated code generating matrix

$$\boldsymbol{G} = \| \, \boldsymbol{I}_{11} \mid -\boldsymbol{A}^T \, \| =$$

$$= \left\| \; \underbrace{\begin{matrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{matrix}}_{\boldsymbol{I}_{11}} \quad \underbrace{\begin{matrix}
1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{matrix}}_{-\boldsymbol{A}^T} \; \right\|$$

Parallel implementation of the encoder for* [15,11] - *1



Parallel implementation of the encoder for [15,11] *- 2*

Straightforward implementation

$$\boldsymbol{x} = \boldsymbol{m} \times \boldsymbol{G} \quad \text{, where matrix } \boldsymbol{G} \text{ is fixed}$$

$$x_i = m_i \quad , \quad \text{for } i = 1,2, \cdots 11$$

$$x_{12} = m_1 \oplus m_2 \oplus m_3 \oplus m_7 \oplus m_8 \oplus m_9 \oplus m_{11}$$

$$x_{13} = m_1 \oplus m_4 \oplus m_5 \oplus m_7 \oplus m_8 \oplus m_{10} \oplus m_{11}$$

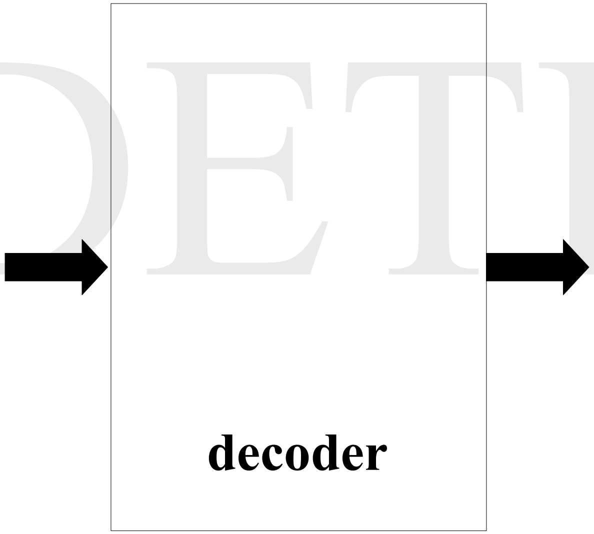
$$x_{14} = m_2 \oplus m_4 \oplus m_6 \oplus m_7 \oplus m_9 \oplus m_{10} \oplus m_{11}$$

$$x_{15} = m_3 \oplus m_5 \oplus m_6 \oplus m_8 \oplus m_9 \oplus m_{10} \oplus m_{11}$$

- 24 x-ors are needed
- 6 x-or propagation time delays in the worst case.

Parallel implementation of the decoder for [15,11] - *1*

$\boldsymbol{y} = (y_1, y_2, \ldots, y_{15})$



$\boldsymbol{m'} = (m'_1, \mathrm{m}'_2, \ldots, \mathrm{m}'_{11})$

Parallel implementation of the decoder for [15,11] - *2*

Straightforward implementation of the error detecting part

$$\boldsymbol{p} = \boldsymbol{H} \times \boldsymbol{y}^T \quad , \text{ where matiz } \boldsymbol{H} \text{ is fixed}$$

$$p_1 = y_1 \oplus y_2 \oplus y_3 \oplus y_7 \oplus y_8 \oplus y_9 \oplus y_{11} \oplus y_{12}$$

$$p_2 = y_1 \oplus y_4 \oplus y_5 \oplus y_7 \oplus y_8 \oplus y_{10} \oplus y_{11} \oplus y_{13}$$

$$p_3 = y_2 \oplus y_4 \oplus y_6 \oplus y_7 \oplus y_9 \oplus y_{10} \oplus y_{11} \oplus y_{14}$$

$$p_4 = y_3 \oplus y_5 \oplus y_6 \oplus y_8 \oplus y_9 \oplus y_{10} \oplus y_{11} \oplus y_{15}$$

- 28 x-ors are needed
- 7 x-or propagation time delays in the worst case.

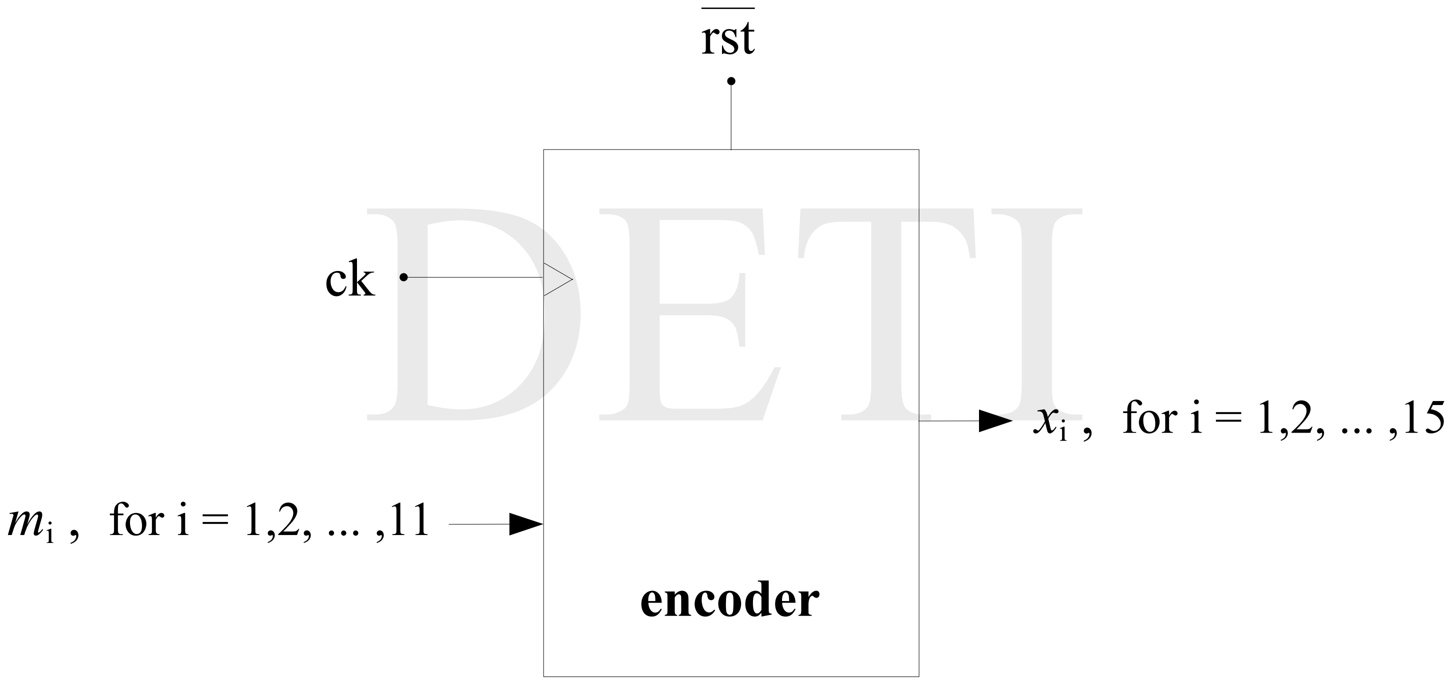
Parallel implementation of the decoder for [15,11] - *3*

Implementation of the error correcting part

$$\boldsymbol{p} \;=\; \boldsymbol{H} \times \boldsymbol{y}^{T} \;\neq\; 0 \quad \Rightarrow \quad \exists_{k} \;\; ml_{k} \;=\; \overline{y_{k}}$$

$$ml_{i} \;=\; sel_{i}(\boldsymbol{p}) \oplus y_{i} \quad , \quad \text{for } i = 1,2\,, \cdots 11$$

Bit-serial implementation of the encoder for [15,11]



Bit-serial implementation of the decoder for [15,11]