

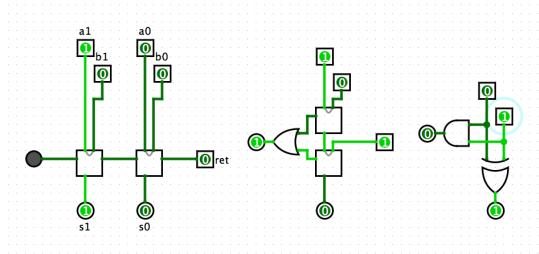
Principes de fonctionnement des machines binaires

2020–2021

Matthieu Picantin



- ◆ numération et arithmétique
- ◆ numération et arithmétique en machine
- ◆ codes, codages, compression,
- ◆ contrôle d'erreur (détection, correction)
- ◆ logique et calcul propositionnel
- ◆ circuits numériques



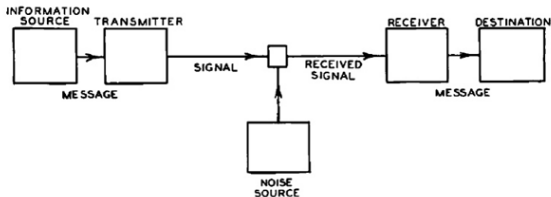


Fig. 1—Schematic diagram of a general communication system.

Détection (+retransmission)

- ♦ juste assez de redondance pour que le récepteur puisse détecter d'éventuelles erreurs & demander une retransmission

Correction

- ♦ suffisamment de redondance pour que le récepteur puisse corriger d'éventuelles erreurs

Redondance et mots de code

- ♦ mot de code (n bits) = données (m bits) + contrôle (r bits)
- ♦ 2^m mots de code (légaux) parmi 2^n mots possibles

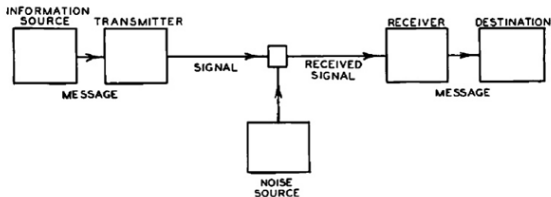


Fig. 1—Schematic diagram of a general communication system.

Détection (+retransmission)

- ♦ juste assez de redondance pour que le récepteur puisse détecter d'éventuelles erreurs & demander une retransmission
- ♦ adapté sur canal fiable

Correction

- ♦ suffisamment de redondance pour que le récepteur puisse corriger d'éventuelles erreurs
- ♦ adapté sur canal bruité

Redondance et mots de code

- ♦ mot de code (n bits) = données (m bits) + contrôle (r bits)
- ♦ 2^m mots de code (légaux) parmi 2^n mots possibles

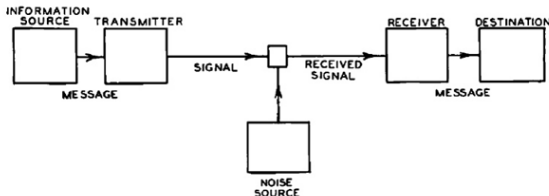


Fig. 1—Schematic diagram of a general communication system.

Détection (+retransmission)

- ♦ **juste assez de redondance** pour que le récepteur puisse **détecter** d'éventuelles erreurs & **demande une retransmission**
- ♦ adapté sur canal fiable

Correction

- ♦ **suffisamment de redondance** pour que le récepteur puisse **corriger** d'éventuelles erreurs
- ♦ adapté sur canal bruité

Redondance et mots de code

- ♦ mot de code (n bits) = données (m bits) + contrôle (r bits)
- ♦ 2^m mots de code (légaux) parmi 2^n mots possibles

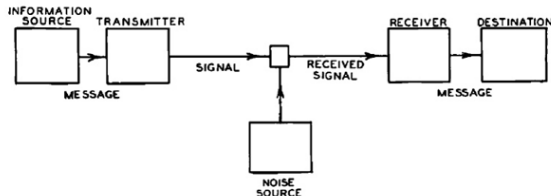


Fig. 1—Schematic diagram of a general communication system.

Détection (+retransmission)

- ♦ **juste assez de redondance** pour que le récepteur puisse **détecter** d'éventuelles erreurs & **demande une retransmission**
- ♦ adapté sur canal fiable

Correction

- ♦ **suffisamment de redondance** pour que le récepteur puisse **corriger** d'éventuelles erreurs
- ♦ adapté sur canal bruité

Redondance et mots de code

- ♦ mot de code (n bits) = données (m bits) + contrôle (r bits)
- ♦ 2^m mots de code (légaux) parmi 2^n mots possibles

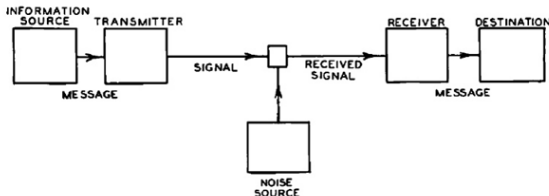


Fig. 1—Schematic diagram of a general communication system.

Détection (+retransmission)

- ♦ **juste assez de redondance** pour que le récepteur puisse **détecter** d'éventuelles erreurs & **demande une retransmission**
- ♦ adapté sur canal fiable

Correction

- ♦ **suffisamment de redondance** pour que le récepteur puisse **corriger** d'éventuelles erreurs
- ♦ adapté sur canal bruité

Redondance et mots de code

- ♦ mot de code (n bits) = données (m bits) + contrôle (r bits)
- ♦ 2^m mots de code (légaux) parmi 2^n mots possibles

Distance de Hamming

- ♦ nombre de bits différents entre 2 mots
- ♦ somme des 1 du XOR des 2 mots
- ♦ nombre minimum d'erreurs simples pour passer d'un mot à l'autre

Distance de Hamming d'un code

distance minimale entre deux mots du code:

$$d_H(C) = \min \{d_H(u, v) : u \neq v \in C\}$$

Qualité d'un code

- ♦ un code C vérifiant $d_H(C) \geq k + 1$ permet de détecter k erreurs
- ♦ un code C vérifiant $d_H(C) \geq 2k + 1$ permet de corriger k erreurs

L'ajout d'un bit de parité produit un code de distance de Hamming 2

Le code $\{0000000000, 0000011111, 1111100000, 1111111111\}$ est de distance de Hamming

Distance de Hamming

- ◆ nombre de bits différents entre 2 mots
- ◆ somme des 1 du XOR des 2 mots
- ◆ nombre minimum d'erreurs simples pour passer d'un mot à l'autre

Distance de Hamming d'un code

distance minimale entre deux mots du code:

$$d_H(\mathcal{C}) = \min \{d_H(u, v) : u \neq v \in \mathcal{C}\}$$

Qualité d'un code

- ◆ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq k + 1$ permet de détecter k erreurs
- ◆ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq 2k + 1$ permet de corriger k erreurs

L'ajout d'un **bit de parité** produit un code de distance de Hamming 2

Le code $\{0000000000, 0000011111, 1111100000, 1111111111\}$ est de distance de Hamming

Distance de Hamming

- ♦ nombre de bits différents entre 2 mots
- ♦ somme des 1 du XOR des 2 mots
- ♦ nombre minimum d'erreurs simples pour passer d'un mot à l'autre

Distance de Hamming d'un code

distance minimale entre deux mots du code:

$$d_H(\mathcal{C}) = \min \{d_H(u, v) : u \neq v \in \mathcal{C}\}$$

Qualité d'un code

- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq k + 1$ permet de détecter k erreurs
- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq 2k + 1$ permet de corriger k erreurs

L'ajout d'un **bit de parité** produit un code de distance de Hamming 2

Le code $\{0000000000, 0000011111, 1111100000, 1111111111\}$ est de distance de Hamming

Distance de Hamming

- ♦ nombre de bits différents entre 2 mots
- ♦ somme des 1 du XOR des 2 mots
- ♦ nombre minimum d'erreurs simples pour passer d'un mot à l'autre

Distance de Hamming d'un code

distance minimale entre deux mots du code:

$$d_H(\mathcal{C}) = \min \{d_H(u, v) : u \neq v \in \mathcal{C}\}$$

Qualité d'un code

- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq k + 1$ permet de détecter k erreurs
- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq 2k + 1$ permet de corriger k erreurs

L'ajout d'un **bit de parité** produit un code de distance de Hamming 2

Le code $\{000000000, 000001111, 111110000, 111111111\}$ est de distance de Hamming

Distance de Hamming

- ♦ nombre de bits différents entre 2 mots
- ♦ somme des 1 du XOR des 2 mots
- ♦ nombre minimum d'erreurs simples pour passer d'un mot à l'autre

Distance de Hamming d'un code

distance minimale entre deux mots du code:

$$d_H(\mathcal{C}) = \min \{d_H(u, v) : u \neq v \in \mathcal{C}\}$$

Qualité d'un code

- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq k + 1$ permet de détecter k erreurs
- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq 2k + 1$ permet de corriger k erreurs

Un code 1-détecteur

L'ajout d'un **bit de parité** produit un code de distance de Hamming 2

Le code $\{000000000, 000001111, 111110000, 111111111\}$ est de distance de Hamming

Distance de Hamming

- ♦ nombre de bits différents entre 2 mots
- ♦ somme des 1 du XOR des 2 mots
- ♦ nombre minimum d'erreurs simples pour passer d'un mot à l'autre

Distance de Hamming d'un code

distance minimale entre deux mots du code:

$$d_H(\mathcal{C}) = \min \{d_H(u, v) : u \neq v \in \mathcal{C}\}$$

Qualité d'un code

- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq k + 1$ permet de détecter k erreurs
- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq 2k + 1$ permet de corriger k erreurs

Un code 1-détecteur

L'ajout d'un **bit de parité** produit un code de distance de Hamming 2

Le code $\{000000000, 000001111, 111110000, 111111111\}$ est de distance de Hamming

Distance de Hamming

- ♦ nombre de bits différents entre 2 mots
- ♦ somme des 1 du XOR des 2 mots
- ♦ nombre minimum d'erreurs simples pour passer d'un mot à l'autre

Distance de Hamming d'un code

distance minimale entre deux mots du code:

$$d_H(\mathcal{C}) = \min \{d_H(u, v) : u \neq v \in \mathcal{C}\}$$

Qualité d'un code

- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq k + 1$ permet de détecter k erreurs
- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq 2k + 1$ permet de corriger k erreurs

Un code 1-détecteur

L'ajout d'un **bit de parité** produit un code de distance de Hamming 2

Le code $\{000000000, 000001111, 111110000, 111111111\}$ est de distance de Hamming 5

Distance de Hamming

- ♦ nombre de bits différents entre 2 mots
- ♦ somme des 1 du XOR des 2 mots
- ♦ nombre minimum d'erreurs simples pour passer d'un mot à l'autre

Distance de Hamming d'un code

distance minimale entre deux mots du code:

$$d_H(\mathcal{C}) = \min \{d_H(u, v) : u \neq v \in \mathcal{C}\}$$

Qualité d'un code

- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq k + 1$ permet de détecter k erreurs
- ♦ un code \mathcal{C} vérifiant $d_H(\mathcal{C}) \geq 2k + 1$ permet de corriger k erreurs

Un code 1-détecteur

L'ajout d'un **bit de parité** produit un code de distance de Hamming 2

Un code 2-correcteur

Le code $\{000000000, 000001111, 111110000, 111111111\}$ est de distance de Hamming 5

En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données
demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

♦ bits numérotés de 1 à $n = m + r$ de gauche à droite

♦ les bits de contrôle sont placés aux positions 2, 4, 8, 16, ...

♦ les bits de données sont placés aux autres positions (1, 3, 5, 7, ...)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données
demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

- ♦ bits numérotés de 1 à $n = m + r$ de gauche à droite
- ♦ r bits de contrôle aux positions puissances de 2 (1, 2, 4, 8, 16, ...)
- ♦ m bits de données aux autres positions (3, 5–7, 9–15, 17, 18...)
- ♦ bits de contrôle = calcul de parité sur les bits de données
aux positions dont la décomposition en somme de puissances de 2
fait intervenir la position du bit de contrôle concerné
- ♦ détection du bit erroné (et correction) par somme
des positions des bits de contrôle non conformes à la parité



En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données
demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

- ♦ bits numérotés de 1 à $n = m + r$ de gauche à droite
- ♦ r bits de contrôle aux positions puissances de 2 (1, 2, 4, 8, 16, ...)
- ♦ m bits de données aux autres positions (3, 5–7, 9–15, 17, 18...)
- ♦ bits de contrôle = calcul de parité sur les bits de données aux positions dont la décomposition en somme de puissances de 2 fait intervenir la position du bit de contrôle concerné
- ♦ détection du bit erroné (et correction) par somme des positions des bits de contrôle non conformes à la parité



En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données
demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

- ♦ bits numérotés de 1 à $n = m + r$ de gauche à droite
- ♦ r bits de contrôle aux positions puissances de 2 (1, 2, 4, 8, 16, ...)
- ♦ m bits de données aux autres positions (3, 5–7, 9–15, 17, 18...)
- ♦ bits de contrôle = calcul de parité sur les bits de données
aux positions dont la décomposition en somme de puissances de 2
fait intervenir la position du bit de contrôle concerné
- ♦ détection du bit erroné (et correction) par somme
des positions des bits de contrôle non conformes à la parité



En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données
demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

- ♦ bits numérotés de 1 à $n = m + r$ de gauche à droite
- ♦ r bits de contrôle aux positions puissances de 2 (1, 2, 4, 8, 16, ...)
- ♦ m bits de données aux autres positions (3, 5–7, 9–15, 17, 18...)
- ♦ bits de contrôle = calcul de parité sur les bits de données
aux positions dont la décomposition en somme de puissances de 2
fait intervenir la position du bit de contrôle concerné
- ♦ détection du bit erroné (et correction) par somme
des positions des bits de contrôle non conformes à la parité

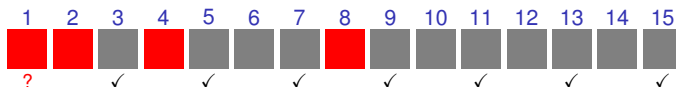


En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

- ♦ bits numérotés de 1 à $n = m + r$ de gauche à droite
- ♦ r bits de contrôle aux positions puissances de 2 (1, 2, 4, 8, 16, ...)
- ♦ m bits de données aux autres positions (3, 5–7, 9–15, 17, 18...)
- ♦ bits de contrôle = calcul de parité sur les bits de données aux positions dont la décomposition en somme de puissances de 2 fait intervenir la position du bit de contrôle concerné
- ♦ détection du bit erroné (et correction) par somme des positions des bits de contrôle non conformes à la parité

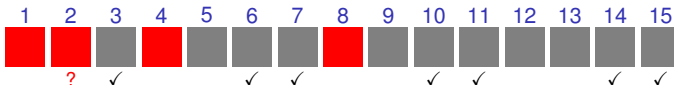


En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

- ♦ bits numérotés de 1 à $n = m + r$ de gauche à droite
- ♦ r bits de contrôle aux positions puissances de 2 (1, 2, 4, 8, 16, ...)
- ♦ m bits de données aux autres positions (3, 5–7, 9–15, 17, 18...)
- ♦ bits de contrôle = calcul de parité sur les bits de données aux positions dont la décomposition en somme de puissances de 2 fait intervenir la position du bit de contrôle concerné
- ♦ détection du bit erroné (et correction) par somme des positions des bits de contrôle non conformes à la parité

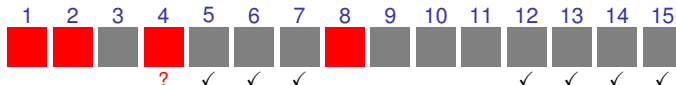


En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

- ♦ bits numérotés de 1 à $n = m + r$ de gauche à droite
- ♦ r bits de contrôle aux positions puissances de 2 (1, 2, 4, 8, 16, ...)
- ♦ m bits de données aux autres positions (3, 5–7, 9–15, 17, 18...)
- ♦ bits de contrôle = calcul de parité sur les bits de données aux positions dont la décomposition en somme de puissances de 2 fait intervenir la position du bit de contrôle concerné
- ♦ détection du bit erroné (et correction) par somme des positions des bits de contrôle non conformes à la parité

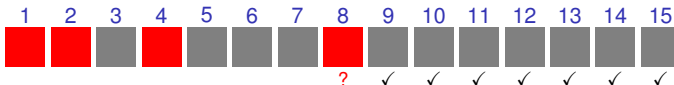


En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

- ♦ bits numérotés de 1 à $n = m + r$ de gauche à droite
- ♦ r bits de contrôle aux positions puissances de 2 (1, 2, 4, 8, 16, ...)
- ♦ m bits de données aux autres positions (3, 5–7, 9–15, 17, 18...)
- ♦ bits de contrôle = calcul de parité sur les bits de données aux positions dont la décomposition en somme de puissances de 2 fait intervenir la position du bit de contrôle concerné
- ♦ détection du bit erroné (et correction) par somme des positions des bits de contrôle non conformes à la parité

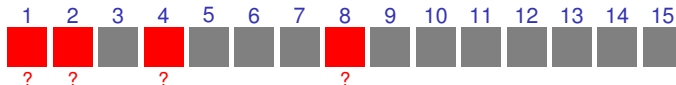


En théorie: de l'espace pour les boules (condition nécessaire)

Pouvoir corriger toute erreur simple pour m bits de données demande r bits de contrôle avec $m + r < 2^r$

En pratique : la méthode de Hamming (condition suffisante)

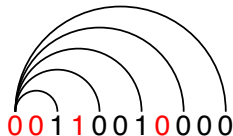
- ♦ bits numérotés de 1 à $n = m + r$ de gauche à droite
- ♦ r bits de contrôle aux positions puissances de 2 (1, 2, 4, 8, 16, ...)
- ♦ m bits de données aux autres positions (3, 5–7, 9–15, 17, 18...)
- ♦ bits de contrôle = calcul de parité sur les bits de données aux positions dont la décomposition en somme de puissances de 2 fait intervenir la position du bit de contrôle concerné
- ♦ détection du bit erroné (et correction) par somme des positions des bits de contrôle non conformes à la parité



00110010000

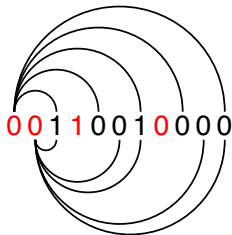
Le code de Hamming pour de l'ASCII

- ♦ valable seulement pour une erreur simple (1 bit)
- ♦ correction de rafale d'erreurs à l'aide d'une matrice



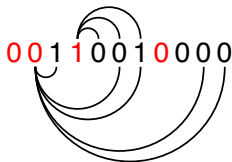
Le code de Hamming pour de l'ASCII

- ♦ valable seulement pour une erreur simple (1 bit)
- ♦ correction de rafale d'erreurs à l'aide d'une matrice



Le code de Hamming pour de l'ASCII

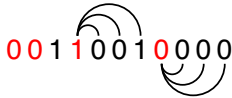
- ♦ valable seulement pour une erreur simple (1 bit)
- ♦ correction de rafale d'erreurs à l'aide d'une matrice



Le code de Hamming pour de l'ASCII

- ♦ valable seulement pour une erreur simple (1 bit)
- ♦ correction de rafale d'erreurs à l'aide d'une matrice

00110010000



Le code de Hamming pour de l'ASCII

- ♦ valable seulement pour une erreur simple (1 bit)
- ♦ correction de rafale d'erreurs à l'aide d'une matrice

00110010000

Le code de Hamming pour de l'ASCII

- ♦ valable seulement pour une erreur simple (1 bit)
- ♦ correction de rafale d'erreurs à l'aide d'une matrice

caractère

ASCII

bits de contrôle

H	1001000
a	1100001
m	1101101
m	1101101
i	1101001
n	1101110
g	1100111
	0100000
c	1100011
o	1101111
d	1100100
e	1100101

ordre de transmission des bits

00110010000
 10111001001
 11101010101
 11101010101
 01101011001
 01101010110
 01111001111
 10011000000
 11111000011
 10101011111
 11111001100
 00111000101

00110010000

Le code de Hamming pour de l'ASCII

- ♦ valable seulement pour une erreur simple (1 bit)
- ♦ correction de rafale d'erreurs à l'aide d'une matrice

caractère

ASCII

bits de contrôle

H	1001000
a	1100001
m	1101101
m	1101101
i	1101001
n	1101110
g	1100111
	0100000
c	1100011
o	1101111
d	1100100
e	1100101

ordre de transmission des bits

00110010000
 10111001001
 11101010101
 11101010101
 01101011001
 01101010110
 01111001111
 10011000000
 11111000011
 10101011111
 11111001100
 00111000101

00110010000

Le code de Hamming pour de l'ASCII

- ♦ valable seulement pour une erreur simple (1 bit)
- ♦ correction de rafale d'erreurs à l'aide d'une matrice

caractère

ASCII

bits de contrôle

H	1001000
a	1100001
m	1101101
m	1101101
i	1101001
n	1101110
g	1100111
	0100000
c	1100011
o	1101111
d	1100100
e	1100101

ordre de transmission des bits

00110010000

1011*001001

1110*010101

1110*010101

01101011001

011*1010110

011*1001111

100*1000000

111*1000011

101*1011111

111*1001100

001*1000101

Codes CRC (Cyclic Redundancy Code)

- ♦ correspondance entre rang des bits et degré des monômes
 - ▶ le mot **10011** code le polynôme $x^4 + x + 1$
- ♦ arithmétique polynomiale (soustraction modulo 2 et division euclidienne)

Utilisation d'un polynôme générateur $G(x)$

- ♦ $G(x)$ de degré r et message $M(x)$
 - ajout de r bits à 0 après le bit de poids faible de $M(x)$
 - division euclidienne par $G(x)$ modulo 2
 - le reste est le code de contrôle (à ajouter par le générateur)

Codes CRC (Cyclic Redundancy Code)

- ♦ correspondance entre rang des bits et degré des monômes
 - ▶ le mot **10011** code le polynôme $x^4 + x + 1$
- ♦ arithmétique polynomiale (soustraction modulo 2 et division euclidienne)

Utilisation d'un polynôme générateur $G(x)$

- ♦ $G(x)$ de degré r et message $M(x)$
- ♦ ajout de r bits à 0 après le bit de poids faible de $M(x)$
- ♦ division de $x^r M(x)$ par $G(x)$: reste $R(x)$
- ♦ envoi de $T(x) = x^r M(x) - R(x)$
- ♦ $T(x)$ est divisible par $G(x)$ (à vérifier par le récepteur !)

Codes CRC (Cyclic Redundancy Code)

- ♦ correspondance entre rang des bits et degré des monômes
 - ▶ le mot **10011** code le polynôme $x^4 + x + 1$
- ♦ arithmétique polynomiale (soustraction modulo 2 et division euclidienne)

Utilisation d'un polynôme générateur $G(x)$

- ♦ $G(x)$ de degré r et message $M(x)$
- ♦ ajout de r bits à 0 après le bit de poids faible de $M(x)$
- ♦ division de $x^r M(x)$ par $G(x)$: reste $R(x)$
- ♦ envoi de $T(x) = x^r M(x) - R(x)$
- ♦ $T(x)$ est divisible par $G(x)$ (à vérifier par le récepteur !)

Codes CRC (Cyclic Redundancy Code)

- ♦ correspondance entre rang des bits et degré des monômes
 - ▶ le mot **10011** code le polynôme $x^4 + x + 1$
- ♦ arithmétique polynomiale (soustraction modulo 2 et division euclidienne)

Utilisation d'un polynôme générateur $G(x)$

- ♦ $G(x)$ de degré r et message $M(x)$
- ♦ ajout de r bits à 0 après le bit de poids faible de $M(x)$
- ♦ division de $x^r M(x)$ par $G(x)$: reste $R(x)$
- ♦ envoi de $T(x) = x^r M(x) - R(x)$
- ♦ $T(x)$ est divisible par $G(x)$ (à vérifier par le récepteur !)

Codes CRC (Cyclic Redundancy Code)

- ♦ correspondance entre rang des bits et degré des monômes
 - ▶ le mot **10011** code le polynôme $x^4 + x + 1$
- ♦ arithmétique polynomiale (soustraction modulo 2 et division euclidienne)

Utilisation d'un polynôme générateur $G(x)$

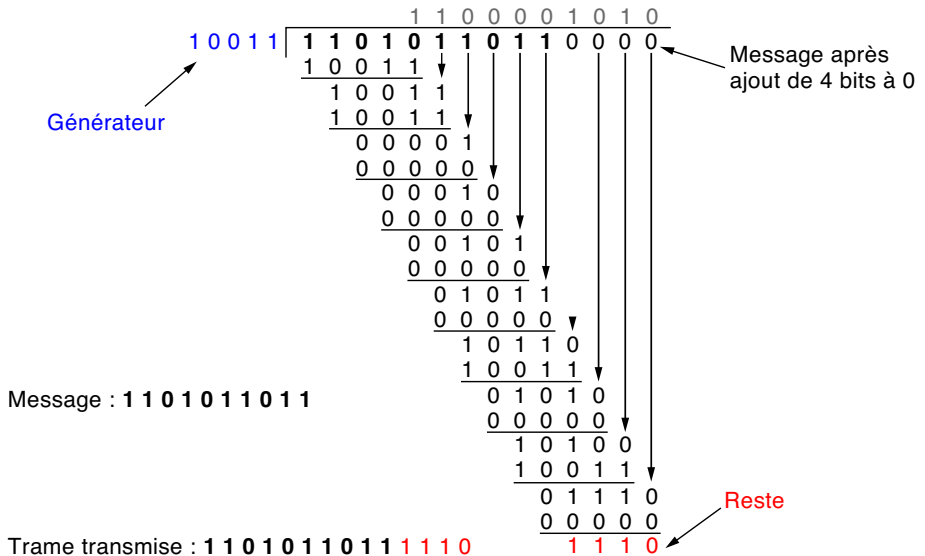
- ♦ $G(x)$ de degré r et message $M(x)$
- ♦ ajout de r bits à 0 après le bit de poids faible de $M(x)$
- ♦ division de $x^r M(x)$ par $G(x)$: reste $R(x)$
- ♦ envoi de $T(x) = x^r M(x) - R(x)$
- ♦ $T(x)$ est divisible par $G(x)$ (à vérifier par le récepteur !)

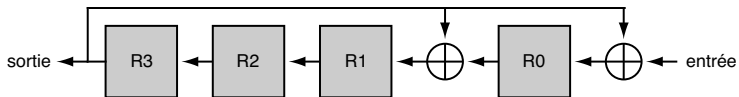
Codes CRC (Cyclic Redundancy Code)

- ♦ correspondance entre rang des bits et degré des monômes
 - ▶ le mot **10011** code le polynôme $x^4 + x + 1$
- ♦ arithmétique polynomiale (soustraction modulo 2 et division euclidienne)

Utilisation d'un polynôme générateur $G(x)$

- ♦ $G(x)$ de degré r et message $M(x)$
- ♦ ajout de r bits à 0 après le bit de poids faible de $M(x)$
- ♦ division de $x^r M(x)$ par $G(x)$: reste $R(x)$
- ♦ envoi de $T(x) = x^r M(x) - R(x)$
- ♦ $T(x)$ est divisible par $G(x)$ (à vérifier par le récepteur !)





	0	0	0	0	1101101011 0000
0	0	0	0	1	101101011 0000
00	0	0	1	1	01101011 0000
000	0	1	1	0	1101011 0000
0000	1	1	0	1	101011 0000
00001	1	0	0	0	01011 0000
000011	0	0	1	1	1011 0000
0000110	0	1	1	1	011 0000
00001100	1	1	1	0	11 0000
000011001	1	1	1	0	1 0000
0000110011	1	1	1	0	0000
00001100111	1	1	1	1	000
000011001111	1	1	0	1	00
0000110011111	1	0	0	1	0
00001100111111	0	0	0	1	

⊕	0	1
0	0	1
1	1	0