Les Codes Correcteurs d'Erreurs (CCE)

Introduction aux CCE

Application VHDL du code de Hamming

Laurent MURA
IUT de Colmar – Département Réseaux et Télécommunications

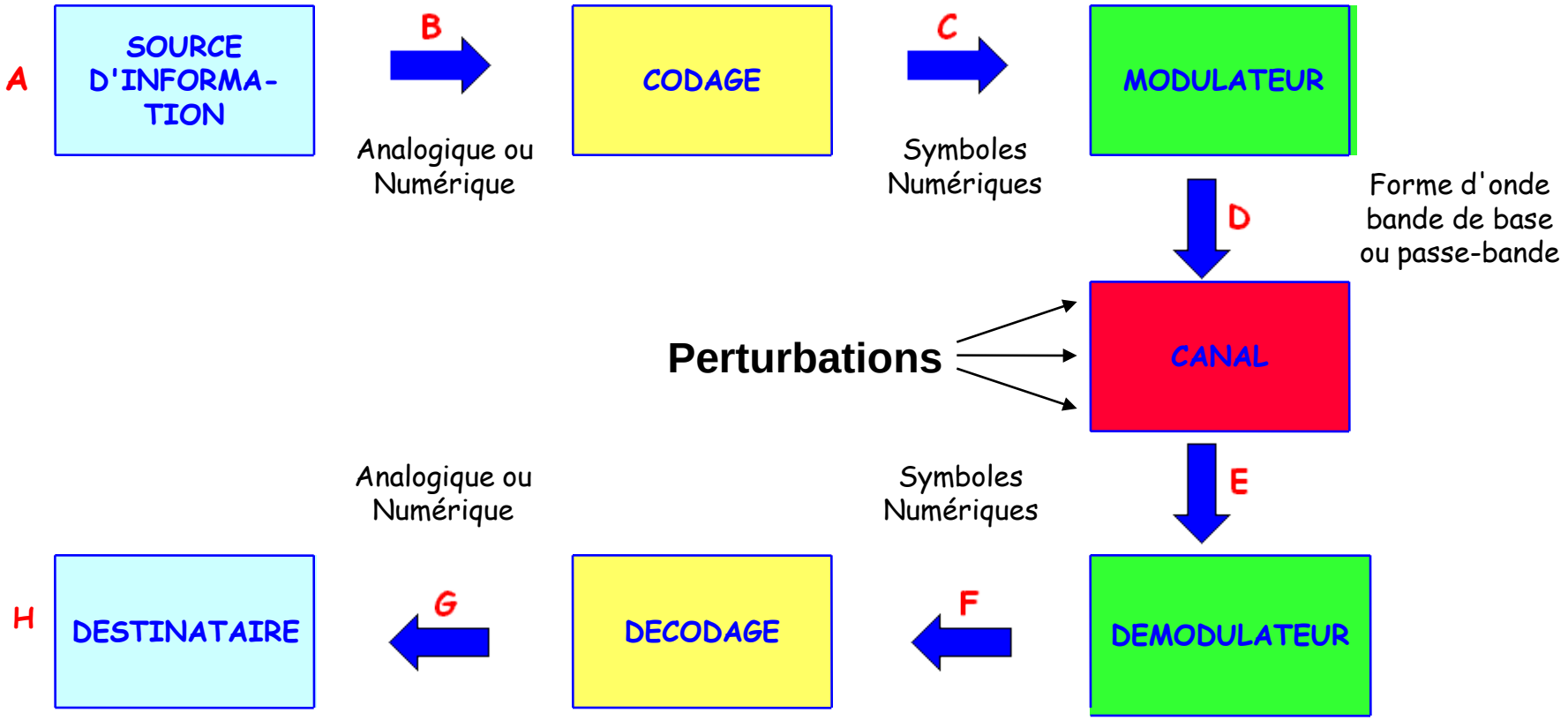
Déroulement de l'exposé

- Introduction
- Les Codes Correcteurs d'Erreurs (CCE)
- Les codes en blocs linéaires
  - Les codes simples
  - Les codes linéaires
  - Les codes polynomiaux
  - Les codes cycliques
- Les codes convolutifs
- Application au code de Hamming sur maquette UP2 d'ALTERA
- Conclusion

Introduction

Rôle du codage
Stratégies: détection et correction

Schéma général : Rôle du codage

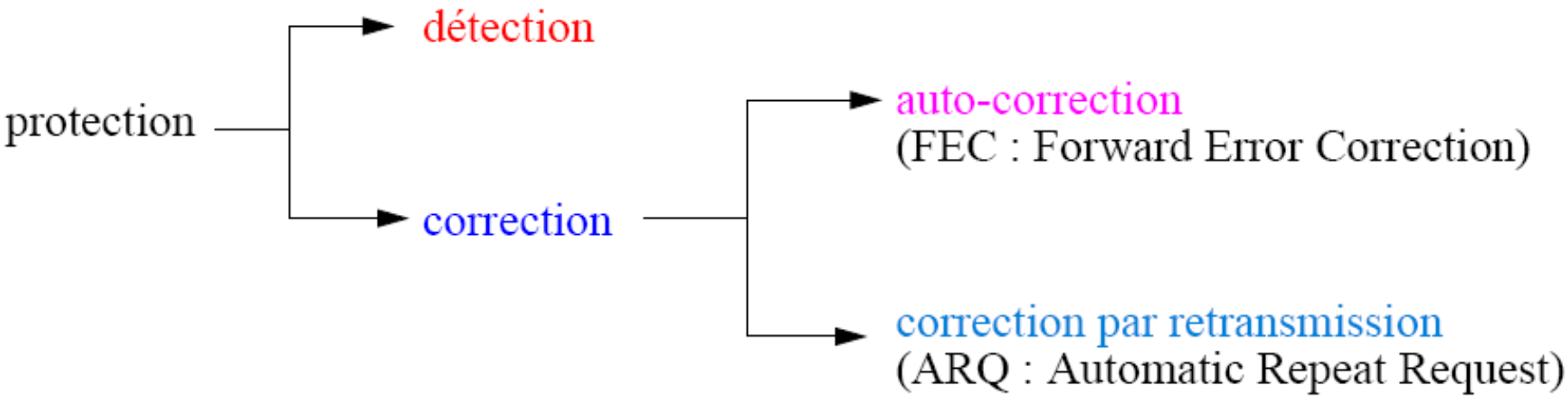


Stratégies

- Des perturbations peuvent engendrer des erreurs sur le signal transmis.
- On peut citer les perturbations suivantes:
  - bruit thermique (composants électroniques)
  - ondes EM (orage)
  - effet Doppler (déplacement)
- Dans ces conditions, la suite binaire reçue ne sera pas identique à la suite émise.

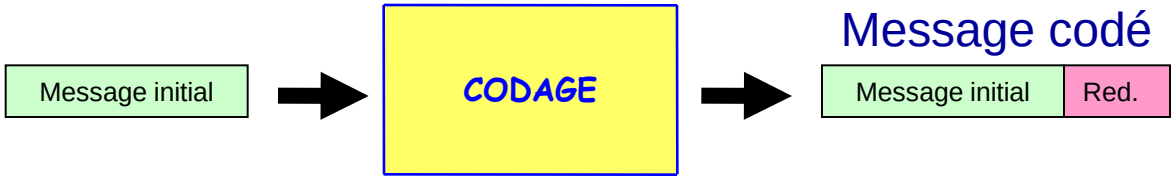
Stratégies

- Mise en oeuvre de techniques de protection contre les erreurs de transmission:



Stratégies : Détection

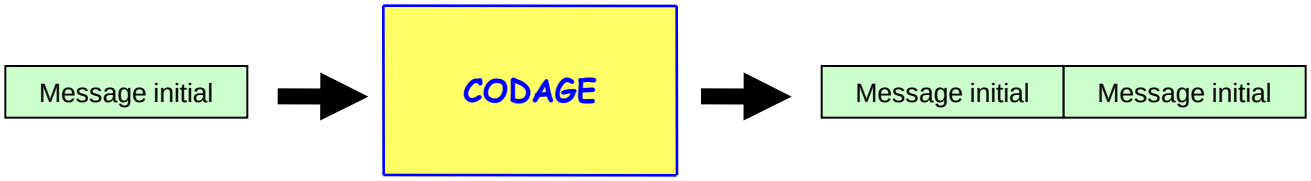
- Principe général pour la détection des erreurs de transmission :
  - un émetteur veut transmettre un message (suite binaire quelconque) à un récepteur.
  - l'émetteur transforme le message initial à l'aide d'un procédé de calcul spécifique qui génère une certaine redondance des informations au sein du message codé:



  - le récepteur vérifie à l'aide du même procédé de calcul que le message reçu est bien le message envoyé grâce à ces redondances.

Stratégies : Détection

- Exemple : la technique de détection par répétition
  - le message codé est un double exemplaire du message initial, le récepteur sait qu'il y a eu erreur si les exemplaires ne sont pas identiques:



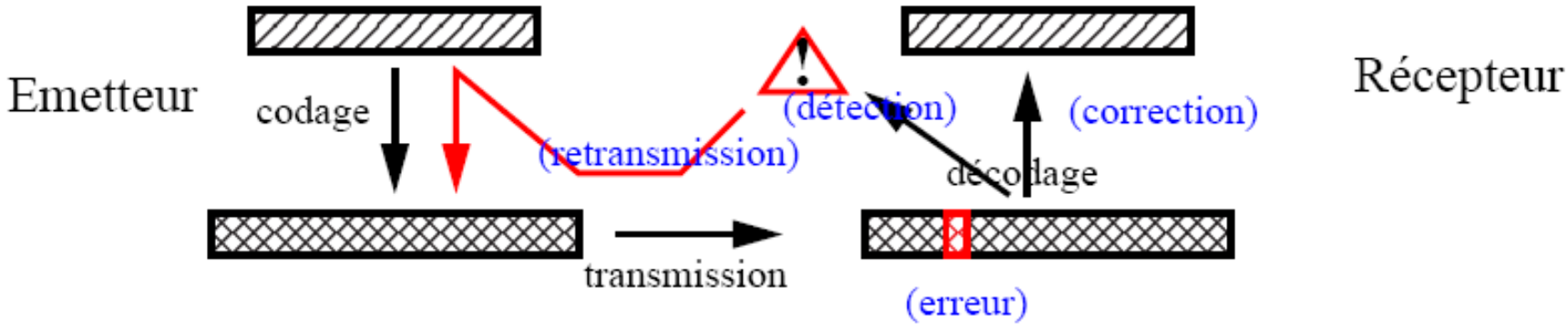
- Note : certaines erreurs sont indétectables !
  - ex. : une même erreur sur les deux exemplaires simultanément

Stratégies : Auto-correction

- Principe général pour l'auto-correction des erreurs de transmission :
  - Après détection d'une erreur, la redondance dans le message transmis est suffisante pour permettre de retrouver le message initial.
- Exemple : la technique de détection par répétition :
  - le message codé est un triple exemplaire du message initial, le récepteur suppose que le message initial correspond aux deux exemplaires qui sont identiques.
  - Note : certaines erreurs détectées ne sont pas corrigibles !!
    - ex. : une erreur différente sur au moins deux exemplaires
  - Note : certaines erreurs sont détectées et mal corrigées !!
    - ex. : une même erreur sur deux exemplaires simultanément

Stratégies : Correction par retransmission

- Principe général pour la correction par retransmission des erreurs de transmission :
  - Après détection d'une erreur, le récepteur demande à l'émetteur de retransmettre une nouvelle fois le message (codé).
  - Exemple : de très nombreux protocoles de télécommunication : X25, TCP.



Les CCE

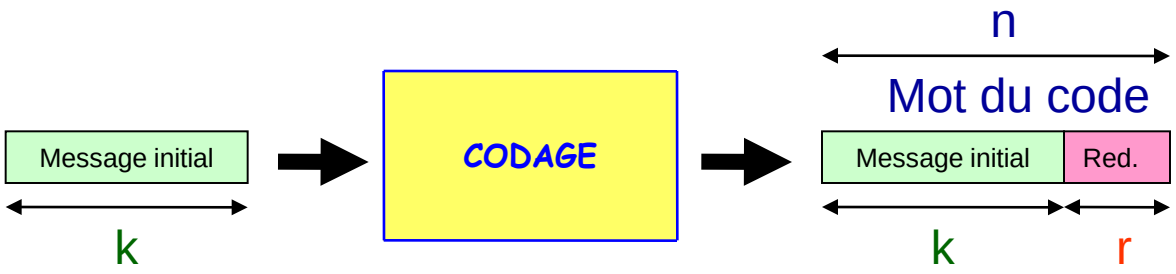
Classification
Définitions générales

Classification

- Trois grandes familles de codes :
  - les codes en blocs linéaires : le codage/décodage d'un bloc dépend uniquement des informations de ce bloc.
  - les codes convolutifs (ou récurrents) : le codage/décodage d'un bloc dépend des informations d'autres blocs (généralement de blocs précédemment transmis)
  - les codes approchant la capacité de Shannon : Turbo codes et LDPC
- Remarque : On préfère généralement les codes en blocs linéaires dans les applications téléinformatiques classiques :
  - le codage/décodage est plus simple et il y a moins de délai

Définitions

- Un code (k, n) transforme (code) tout bloc initial de *k* bits d'information en un bloc codé de *n* bits. Le code introduit une redondance puisque n >= k. Le code est systématique si les *k* premiers bits du bloc codé sont égaux aux bits du bloc initial. Alors les *r* (r=n-k) derniers bits forment un champ de contrôle d'erreur.
- On appelle mot du code, la suite de *n* bits obtenue après un codage (k, n). Le nombre *n* de bits qui composent un mot du code est appelé la longueur du code. La dimension *k* étant la longueur initiale des mots.



Définitions

- Exemple:

  mot initial = [1 0 1 1 1], mot codé = [1 0 1 1 1 1 0]

- Le rendement d'un code (k, n) est : **R = k/n, ici R = 5/7**
  - **La protection est dite forte si R tend vers 0**
  - **La protection est dite faible si R tend vers 1**
- Le poids de Hamming d'un mot est le nombre de bits à 1 qu'il contient:
  - Exemple: p(0110000) = 2
- La distance de Hamming entre deux mots de même longueur est définie par le nombre de positions binaires qui diffèrent entre ces deux mots. On l'obtient par le poids de Hamming de la somme binaire des 2 mots.

Définitions

- Exemple : d(1100110, 1010110) = p(1100110 XOR 1010110) = p(0110000) = 2

- La distance de Hamming d'un code est la distance minimum entre tous les mots du code.
  - Exemple : le code :

C= [000000 3, 3, 4, 3, 4,
0011104, 3, 4, 3
010101 3, 4, 3
0110113, 4
1000113
101101]

a une distance de Hamming $\mathbf{d_H = 3}$

Définitions

- La capacité de détection (de correction) d'un code est définie par les configurations erronées qu'il est capable de détecter (corriger). Une erreur simple (resp. double, ou d'ordre p) affecte une seule (resp. 2, ou p) position(s) binaire(s) d'un mot. Pour qu'un code ait une capacité de détection (resp. correction) des erreurs d'ordre *e*, il faut que sa distance de Hamming soit supérieure à *1+e* (resp. *1 + 2e*), ou encore:
  - **Nombre d'erreurs détectées = (d_H – 1)**
  - **Nombre d'erreurs corrigées = ((d_H – 1) / 2)**
- Exemple : distance d_H =3 ➔ capacité de détection <= 2, capacité de correction <= 1.

Exemples simples de codes en blocs linéaires

Contrôle de parité:
Bit de parité
Tableau de parité

Le contrôle de parité: Bit de parité

- Parité paire (impaire) : le poids de Hamming des mots du code est paire (impaire)
- C'est un code systématique (k, k+1) dans lequel un bit (le bit de parité) est ajouté au mot initial pour assurer la parité. Son rendement est faible lorsque k est petit.
- Exemple :
  - Transmission de caractères utilisant un code de représentation (le code ASCII sur 7 bits):

Lettre	Code ASCII	Mot codé (parité paire)	Mode codé (parité impaire)
E	1 0 1 0 0 0 1	1 0 1 0 0 0 1 1	1 0 1 0 0 0 1 0
V	0 1 1 0 1 0 1	0 1 1 0 1 0 1 0	0 1 1 0 1 0 1 1
A	1 0 0 0 0 0 1	1 0 0 0 0 0 1 0	1 0 0 0 0 0 1 1

Le contrôle de parité: Bit de parité

- L'émetteur transmet alors le message contrôlé puis le récepteur vérifie la parité du message reçu.
- Si lors de la transmission une erreur survient, la parité du message reçu est incorrecte et le récepteur décèle l'existence d'une erreur sans pour autant être capable de la localiser. Il demande alors une nouvelle émission du message.
- Si lors de la transmission deux erreurs surviennent, la parité du message reçu est correcte, les erreurs ne sont pas décelées...

Le contrôle de parité: Parité longitudinale et transversale (tableau de parité)

- Le bloc de données est disposé sous une forme matricielle (k=a.b). On applique la parité (uniquement paire) sur chaque ligne et chaque colonne. On obtient une matrice (a+1, b+1):

1 0 1 0 0 0 1 1
0 1 1 0 1 0 1 0
1 0 0 0 0 0 1 0
---
0 1 0 0 1 0 1 1

- Le rendement est très faible : a.b/ (a+1).(b+1).

Le contrôle de parité: Parité longitudinale et transversale (tableau de parité)

- Principe : Une erreur simple modifie simultanément la parité d'une ligne et d'une colonne.
- Correction : inverser le bit situé à l'intersection de la ligne et de la colonne ayant une parité incorrecte.

1 0 1 0 0 0 1 1	1 0 1 0 0 0 1 1
0 1 1 0 1 0 1 0	0 1 **1** 0 **1** 0 1 0
1 0 0 0 1 0 1 0	1 0 **0** 0 1 0 1 0
0 1 0 0 1 0 1 1	0 1 0 0 1 0 1 1

- Attention : une erreur triple peut faire croire à une erreur simple et sa correction sera inadaptée !

Les codes en blocs linéaires

Définitions
Propriétés
Codes de Hamming
Exemple

Définitions

- Les codes en blocs linéaires sont des codes dont chaque mot du code (noté *c*) est obtenu après **transformation linéaire** des bits du mot initial (noté *i*).
- Ces codes sont caractérisés par leur matrice G(k, n) (appelée matrice génératrice) telle que : i . G = c

$$\begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$$

- Utilisation du OU exclusif pour le calcul matriciel…

Définitions

- La matrice H(n-k, n) (appelée matrice de contrôle de parité) permet de savoir si un mot reçu est un mot du code, en calculant son syndrome. Si le syndrome du mot est nul, ce mot appartient au code:

Syndrome du mot reçu c' : $c'.\ H^T = 0_{(n-k)} <==> c' \in C_{(k, n)}$

L'équation $G\ .\ H^T = 0$ définit la relation entre les deux matrices.

Preuve : $c'.\ H^T = i.\ G.\ H^T = i.\ 0$

Propriétés

- La distance de Hamming d'un code linéaire est égale au plus petit poids de Hamming non nul des mots du code.
- Si un code en blocs linéaire est systématique, sa matrice génératrice s'écrit :

$$G(k, n) = [Id(k), P(k, n-k)]$$

$$Ou,\ G(k, n) = [P(k, n-k), Id(n-k),]$$

- alors sa matrice de contrôle s'écrit :

$$H(n-k, n) = [P^T(n-k,k), Id(n-k)]$$

$$Ou,\ H(n-k, n) = [\ Id(n-k), P^T(n-k,k)]$$

Propriétés : exemple

G1 =

1	0	0	0	1	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	0	1

H =

1	0	1	0	1	0
0	1	0	1	0	1

- Le mot de code c1 associé au mot i = [1 0 1 1]

est c1 = i.G1 = [1 0 1 1 0 1]

- Son syndrome s1 vaut $c1.H^T$ = [0 0] ➔ pas d'erreur
- Imaginons recevoir le code c2 = [1 0 1 0 0 1]
- Le syndrome s2 vaut $c2.H^T$ = [0 1] ➔ erreur
- Par contre pas de possibilité de correction car H a des colonnes identiques…
- Alternative par code de Hamming

Code de Hamming

- Famille de codes linéaires auto-correcteurs faciles à corriger.
- Principe de construction d'un code de Hamming dense $C(k = 2^m-m-1, n = 2^m-1)$:
  - Chaque colonne de sa matrice de contrôle de parité $H(n-k=m, n=2^m-1)$ est non-nulle et différente.
  - Propriétés :
    - leur distance minimale est égale à 3 (au moins)
    - correction des erreurs simples et détection des erreurs doubles.

Code de Hamming: Exemple

- Auto-correction :
  - le syndrome du mot reçu est identique à la colonne de la matrice de contrôle correspondant au bit à corriger.
  - si l'on trie les colonnes de H suivant leur poids binaire croissant et que les valeurs de ses colonnes couvrent l'intervalle [1, 2^m-1] alors la valeur binaire du syndrome est égale au numéro de bit erroné.
- Exemple:

H =

1	1	1	0	1	0	0
1	1	0	1	0	1	0
1	0	1	1	0	0	1

G1 =

1	0	0	0	1	1	1
0	1	0	0	1	1	0
0	0	1	0	1	0	1
0	0	0	1	0	1	1

Remarque 1: $G1*H^T = 0_{(4,3)}$

Remarque 2: les colonnes de H sont toutes différentes !

Code de Hamming: Exemple

- Soit le code c1 = [0 1 1 1 0 0 0]
- Son syndrome s1 vaut $c1*H^T$ = [0 0 0] ➔ pas d'erreur
- Soit le code c2 = [0 1 1 1 0 1 0]
- Son syndrome s2 vaut $c2*H^T$ = [0 1 0] ➔ erreur ➔ $s2^T$ se trouve dans la deuxième colonne de H ➔ erreur sur le deuxième bit de c2 !

Les codes polynômiaux

Définitions
Principe
Division binaire/polynômiale
Exemple

Définitions

- Rappel : tout vecteur peut être présenté sous une forme polynômiale :

$$U = \langle u0,u1,u2,...,un \rangle \Longleftrightarrow U(x) = u0+u1.x+u2.x2+...+un.xn$$

- Attention : les opérations sont binaires (construits sur le corps Z/2Z ➔ OU exclusif) : 1.x + 1.x = 0.x !
- Définition : Un code polynômial est un code en blocs linéaire systématique dont chacun des mots du code est un multiple du polynôme générateur (noté g(x)).
- Capacité de détection: ce type de code permet de détecter toutes les erreurs d'ordre inférieur au degré du polynôme générateur

Principe

- Possibilité de vérifier qu'un ensemble arbitraire de bits, transmis d'un émetteur à un récepteur, n'a pas été modifié pendant le transport.
- L'idée du CRC (Contrôle de Redondance Cyclique) est de transmettre, conjointement avec les données significatives, une valeur calculée par l'émetteur à partir des Data à transmettre. Le récepteur effectue le même calcul de son côté, et si le résultat est identique, alors le message n'a pas été modifié . Dans tout les autres cas, on redemande à transmettre !
- Le calcul va utiliser une opération de base : la division des polynômes (modulo 2).

Principe

- Appelons M(x) le polynôme représentant les data à coder. Son degré est représenté par **m**.
- Chacune des deux extrémités connaissent un polynôme de degré **k**, appelé **générateur**, que l'on nommera G(x).
  - a) La première étape consiste à multiplier M(x) par X^k. Le résultat P(x) = X^k * M(x) est un polynôme de degré m+k. En fait, cela provoque un décalage de M(x) de k rangs vers la gauche. Ça permet de garantir que la soustraction entre P(x) et un polynôme de degré k est toujours positive.

Principe

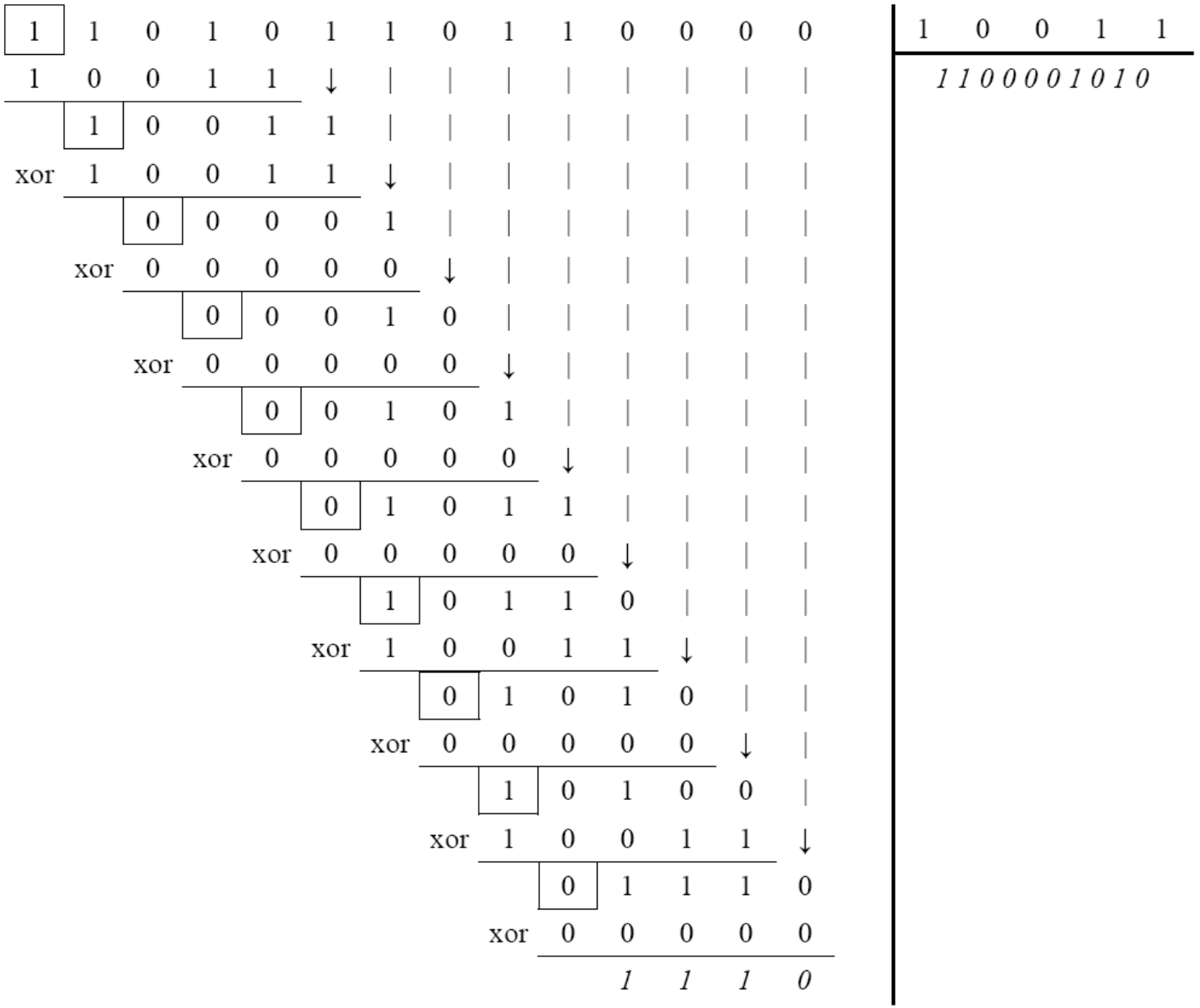
- b) On effectue la division euclidienne de P(x) par G(x) . Le calcul fournit :
  - un Quotient Q(x) , et
  - un Reste R(x) de **degré k-1**.
  - Soit : **P(x) = Q(x) * G(x) + R(x)**
- c) Le résultat final est calculé par S(x) = P(x) - R(x) = [Q(x) * G(x) + R(x)] - R(x) <=> **S(x) = Q(x) * G(x)** .

- Donc, **si l'on divise S(x) par G(x) , le reste doit être nul**, des deux côtés de la connexion, ou alors les valeurs traitées ne sont pas les mêmes

Quelques polynômes générateurs

Nom	Polynôme générateur
CRC-4	$x^4 + x^2 + x^1$
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
CRC-16 SDLC (CCITT)	$x^{16} + x^{12} + x^5 + 1$
CRC-16	$x^{16} + x^{15} + x^2 + 1$
CRC-16 Reverse	$x^{16} + x^{14} + x^1 + 1$
SDLC Reverse	$x^{16} + x^{11} + x^4 + 1$
CRC-32 (ethernet)	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$

Division binaire/polynômiale



Codes polynômiaux: Exemple

- Soit M = 1100101011 une donnée à contrôler et G = 10101 un polynôme générateur codé en binaire ($X^4 + X^2 + 1$).
- Calcul de P: le degré de G étant de 4, on multiplie par X^4 ➔ décalage de 4 rangs vers la gauche ➔ P = 11001010110000
- Calcul P / G = 11001010110000 / 10101 ➔ Le reste R est 0110
- Calcul de S: S = P - R = 11001010110000 – 0110 (OU exclusif)

  S = 11001010110110
- Remarque: le résultat S est bien constitué des **data d'origine suivi du CRC (code systématique)**.
- Le récepteur divise les données reçues S par la valeur du Générateur. Si l'opération ne donne pas un reste nul, alors il y a eu modification des données lors du transport.

Les codes cycliques

Définitions
Propriétés
Autres CCE plus performants

Définitions

- Un code cyclique (k, n) est un code en blocs linéaire (k, n) tel que toute permutation circulaire d'un mot du code est encore un mot du code.
- Exemple :
  - Un code cyclique (1, 2) possède les mots de code suivants : {01, 10} ou {00, 11}, mais pas {01, 11}.
  - Un code cyclique (1, 3) possède les mots de code suivants : {000, 111}.

Propriétés

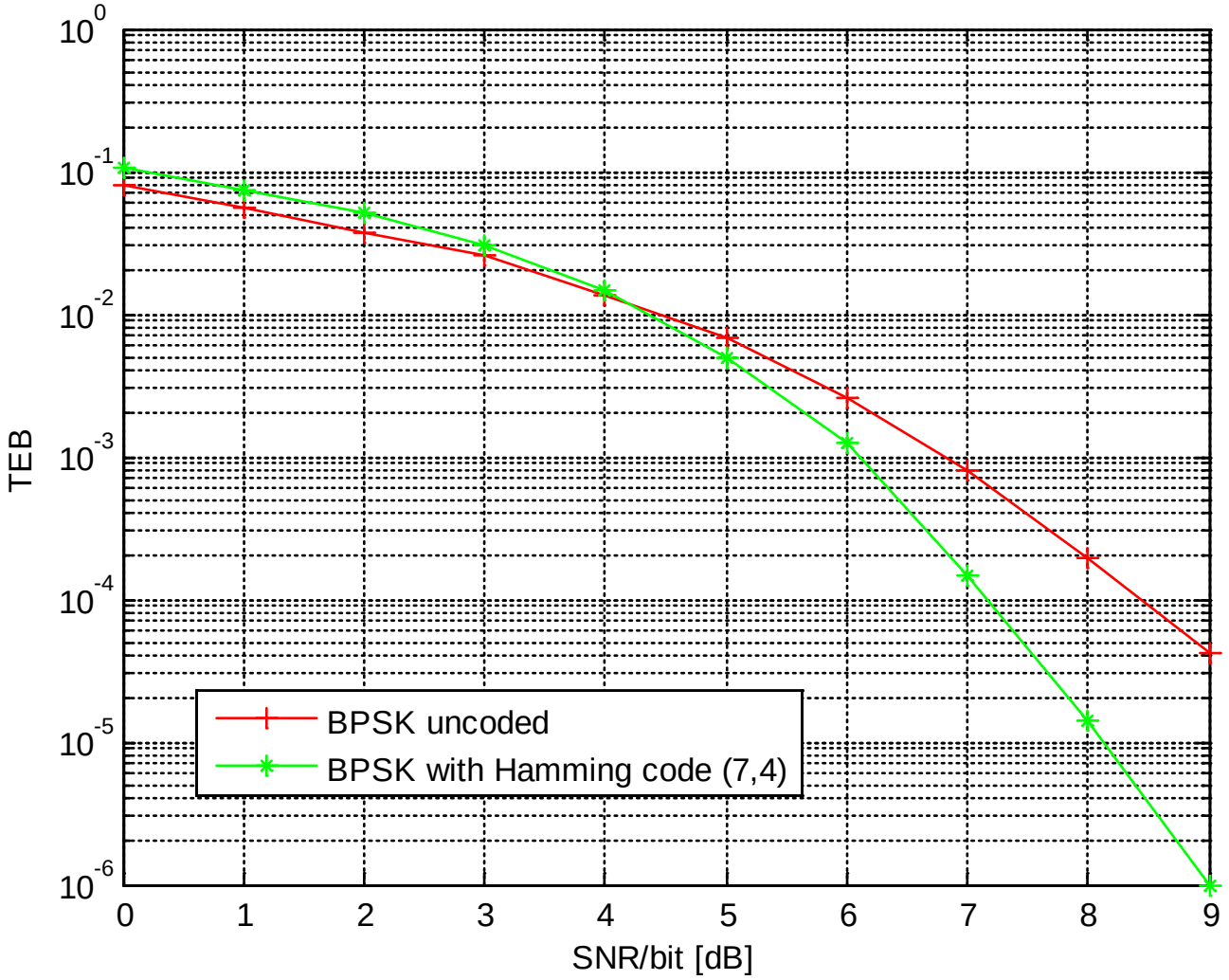
- Un code cyclique (k, n) est un code polynômial dont le polynôme générateur divise $x^n + 1$.
- Les dispositifs de codage et de décodage sont identiques à ceux des codes polynômiaux.
- Les codes cycliques sont principalement utilisés pour leur capacité de correction.
- Les codes cycliques (k, n) dont le polynôme générateur est primitif et tel que $n=2^{n-k} + 1$, sont des codes de Hamming.

Autres CCE plus performants

- Un code de Hamming est compact si les 2^{n-k} syndromes différents sont utilisés pour identifier les $2^{n-k}-1$ erreurs simples (+ syndrome nul !).
- Les codes BCH (Bose-Chaudhuri-Hocquenghem) sont ceux qui ont la plus grande capacité de correction d'erreurs indépendantes pour une redondance et une longueur données. Leur rendement n'est pas très élevé.
- Les codes RS (Reed-Solomon) sont des codes correcteurs très puissants. Ils peuvent être présentés comme des codes BCH dans lequel chaque bit des mots du code est remplacé par un entier défini modulo 2^v (avec $n=2^v-1$). La distance d'un code RS(m, n) est égale à n-m+1. [Utilisé dans les lecteurs CD pour éviter les erreurs en rafale]

Conclusion sur les codes en blocs

Efficacité de correction

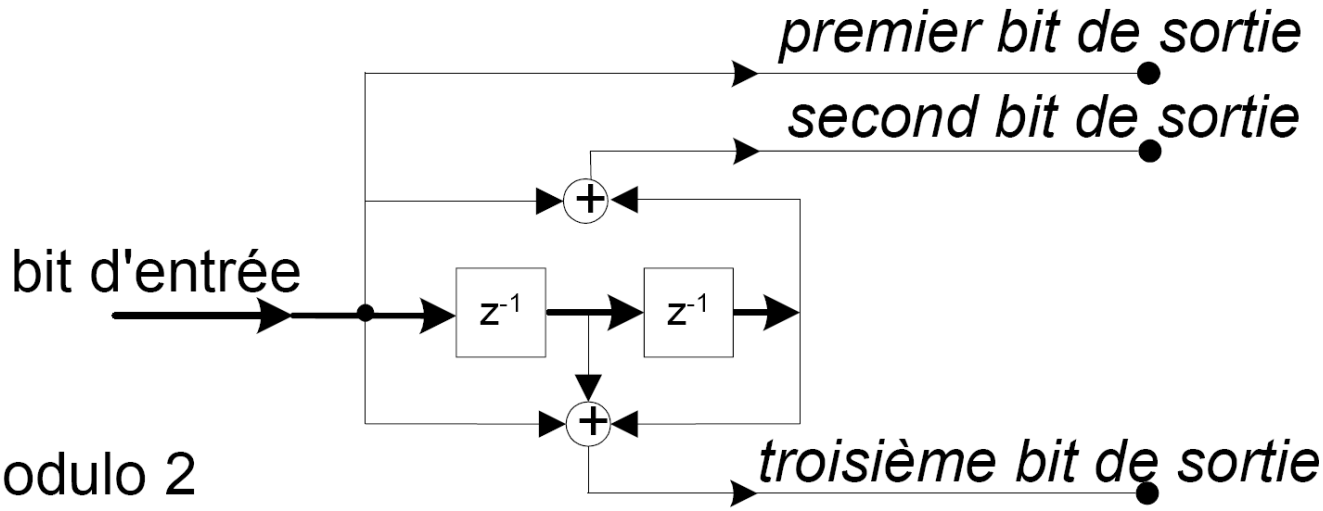


Inconvénients majeurs

- Le regroupement du message en blocs implique normalement une latence due à la nécessité de disposer de tous les symboles du message avant de procéder au codage ou au décodage. Les codes travaillant sur des grands blocs peuvent ainsi entraîner une forte latence, ce qui rend le retard algorithmique du système important.
- ➔ Regroupement du message en blocs peut être inadapté pour les applications en temps réel (ex. video, signal de parole).

Avantages majeurs

- Le décodage est « relativement » simple par rapport aux codes convolutifs car les bits du code reçu à traiter ne dépendent pas des bits précédents.
- Exemple du code convolutif (1,3):



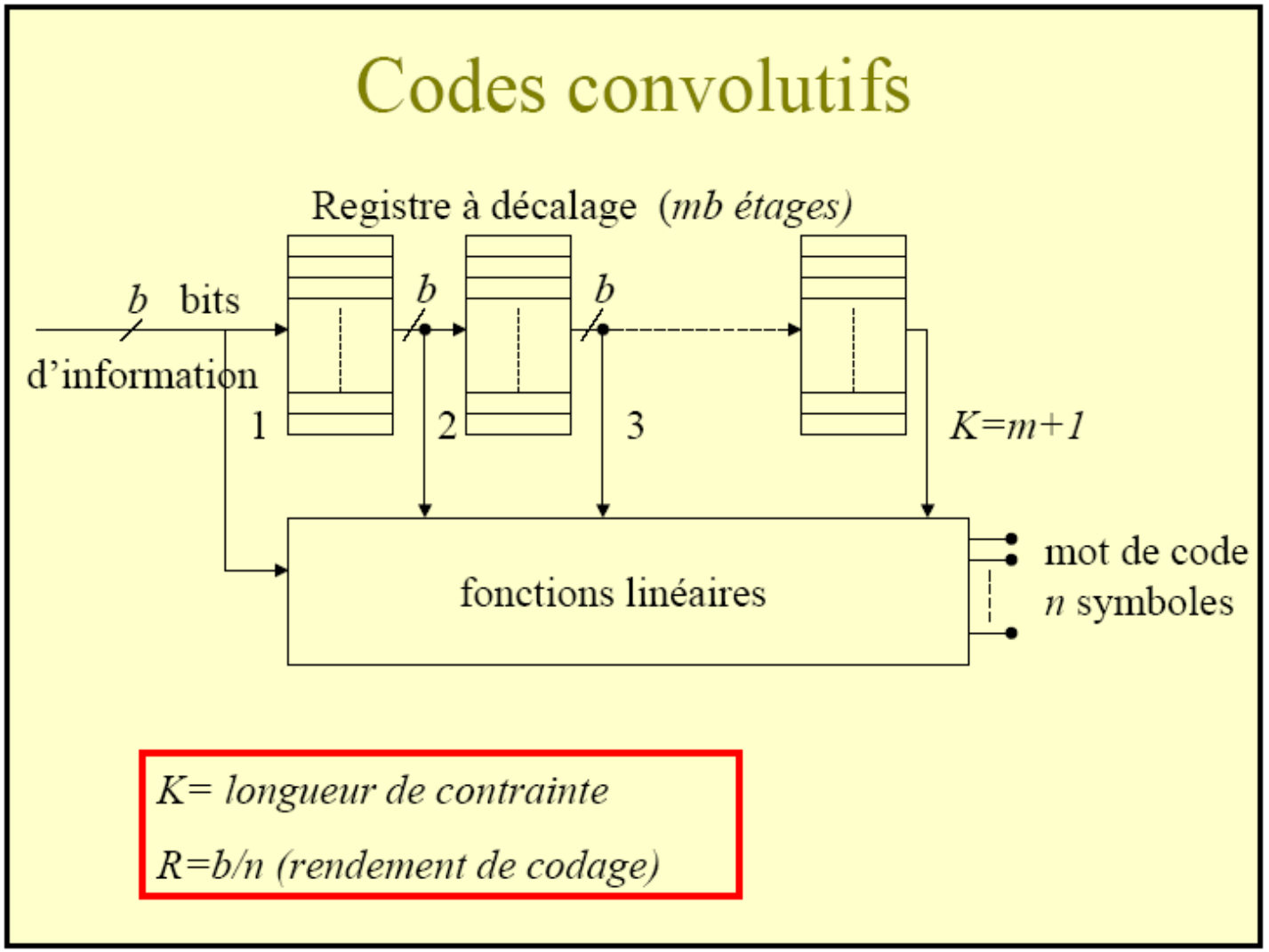
Codes Convolutifs

Principe
Exemple

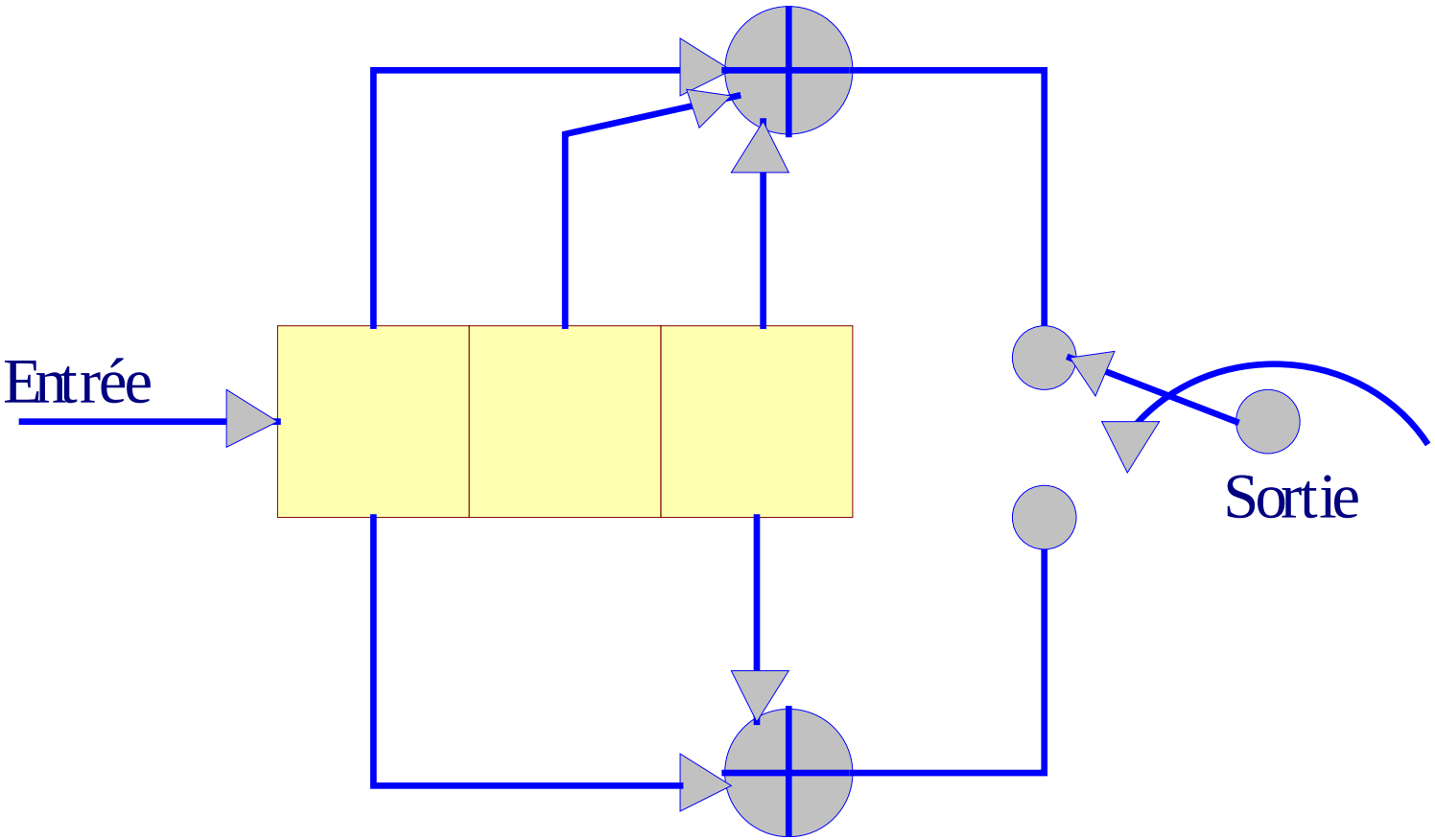
Principe

- Présenté par Elias en 1955, ce type de codage ajoute systématiquement de la redondance au message codé au fur et à mesure que les symboles du message (chacun formés de *b* bits) sont livrés au codeur. Le message codé se forme ainsi **itérativement** en utilisant un registre à décalage. Ce registre est dimensionné pour accueillir les *K* symboles les plus récents du message et la génération du message codé utilise *n* fonctions linéaires algébriques. Ces fonctions sont appelées fonctions génératrices.
- Les codes convolutifs sont plus performants que les codes en blocs linéaires pour un mot codé de même dimension.

Principe

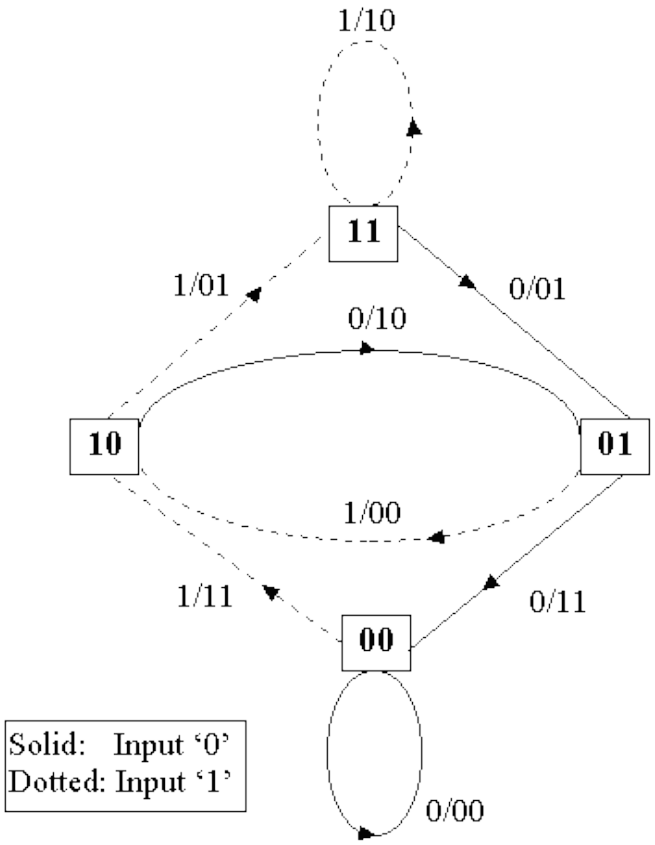

Codes convolutifs
Registre à décalage (mb étages)
b bits
d'information
b
b
1
2
3
K=m+1
fonctions linéaires
mot de code
n symboles
K= longueur de contrainte
R=b/n (rendement de codage)

Exemple: code convolutif (1,2)

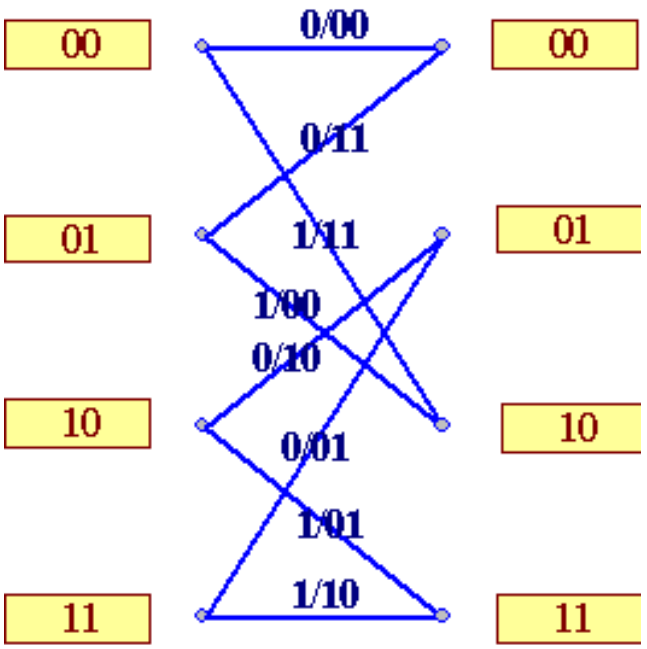


Exemple: Décodage

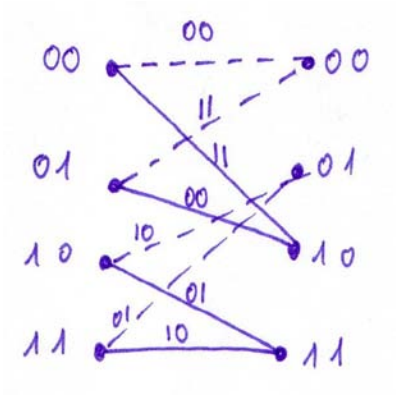
Bloc diagramme

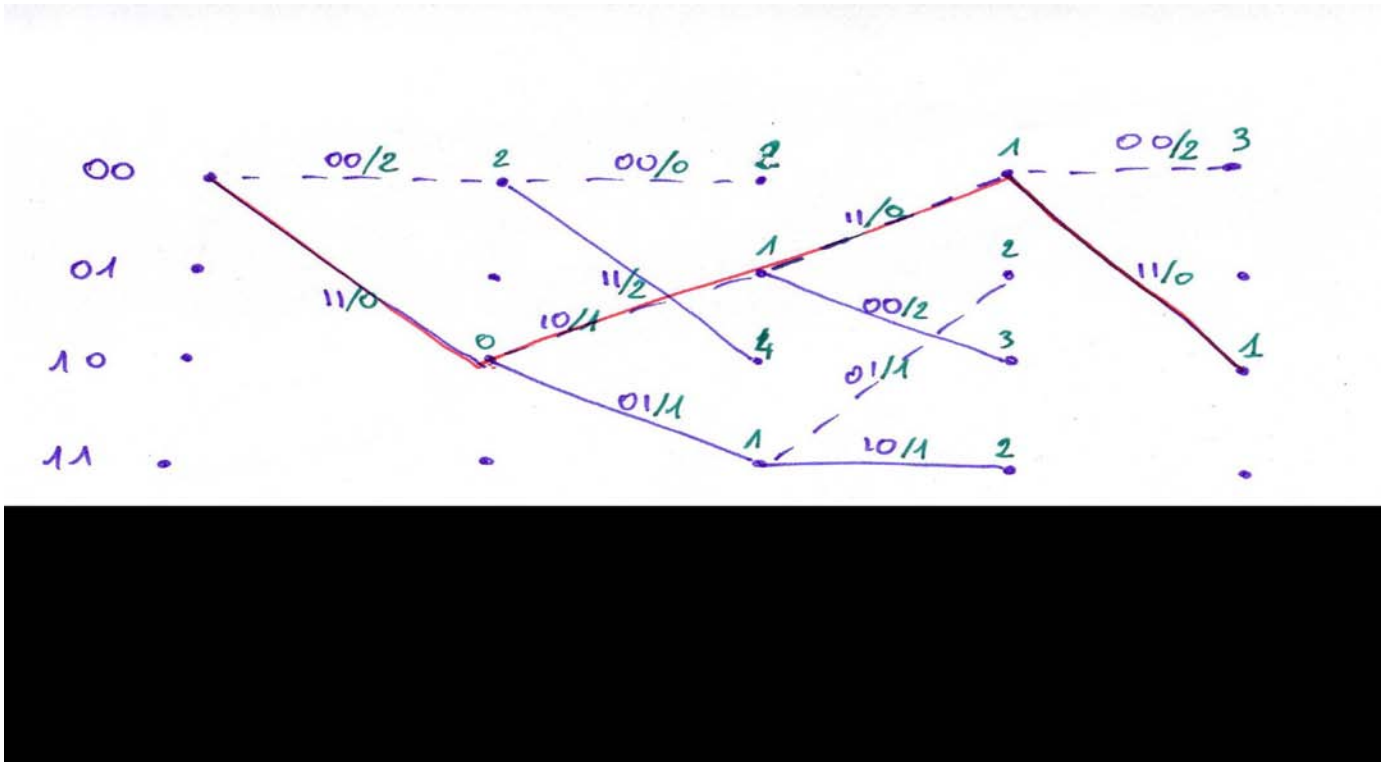


Treilli



Exemple: Décodage par algorithme de VITERBI

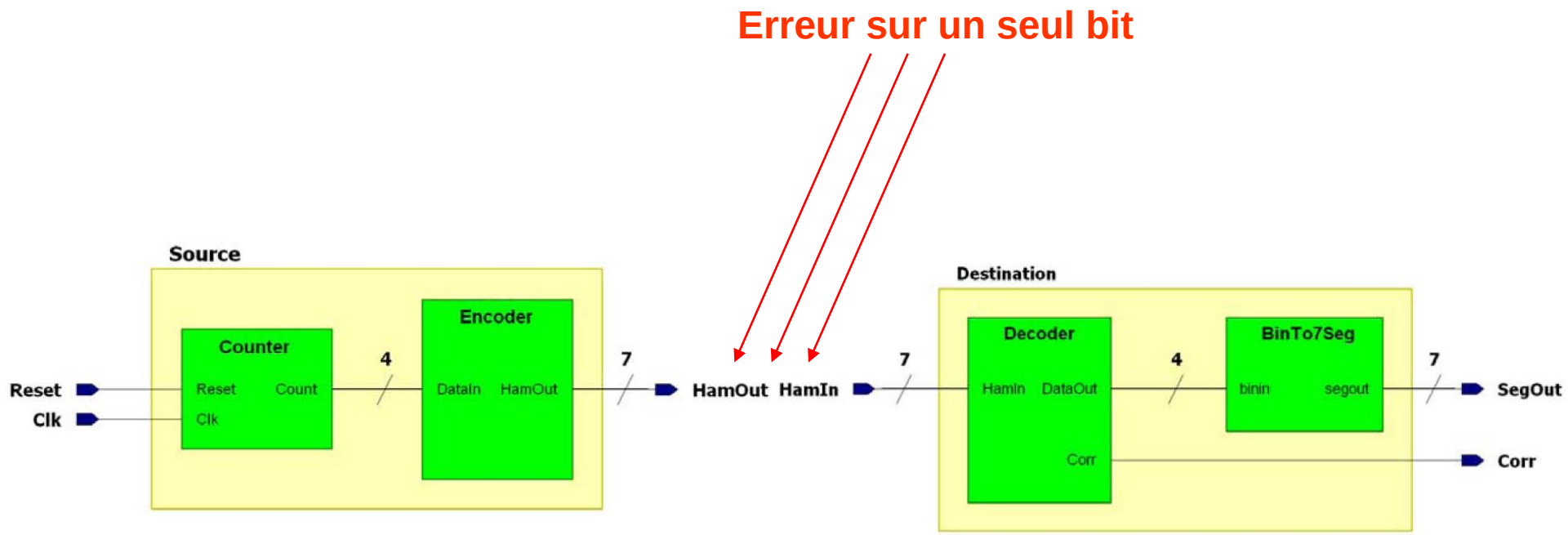




Application du code de Hamming : Maquette ALTERA UP2 - VHDL

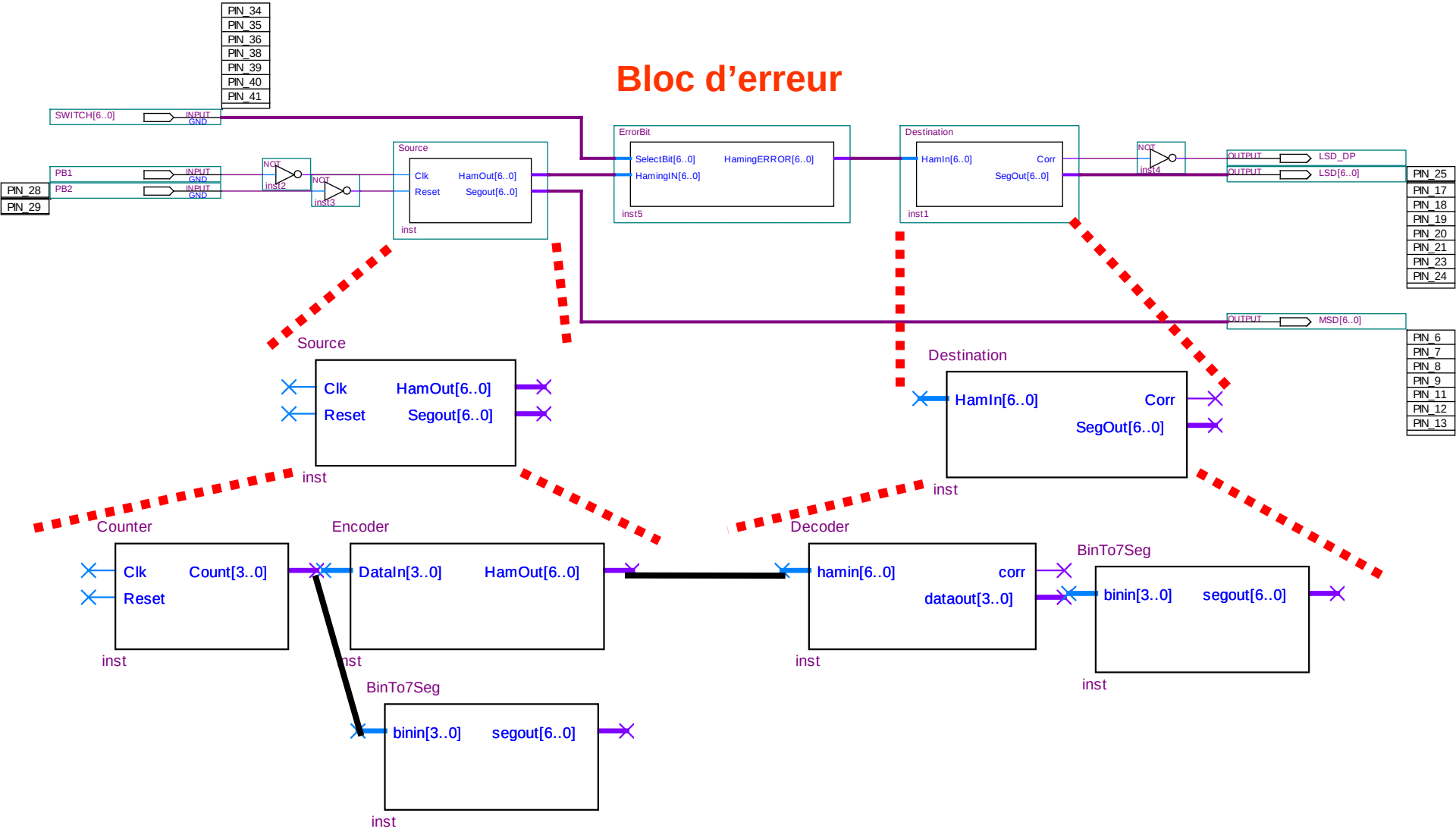
Principe
Scripts VHDL

Principe



Source: lapwww.epfl.ch (Ecole Polytechnique Fédéral de Lausanne)

Principe


Bloc d'erreur
PIN_34
PIN_35
PIN_36
PIN_38
PIN_39
PIN_40
PIN_41
SWITCH[6..0]
INPUT
GND
PB1
PB2
PIN_28
PIN_29
NOT
inst2
inst3
Source
Clk
Reset
HamOut[6..0]
Segout[6..0]
inst
ErrorBit
SelectBit[6..0]
HamingIN[6..0]
HamingERROR[6..0]
inst5
Destination
HamIn[6..0]
Corr
SegOut[6..0]
inst1
inst4
OUTPUT
LSD_DP
LSD[6..0]
MSD[6..0]
PIN_25
PIN_17
PIN_18
PIN_19
PIN_20
PIN_21
PIN_23
PIN_24
PIN_6
PIN_7
PIN_8
PIN_9
PIN_11
PIN_12
PIN_13
Counter
Count[3..0]
Encoder
DataIn[3..0]
BinTo7Seg
binin[3..0]
segout[6..0]
Decoder
hamin[6..0]
corr
dataout[3..0]

Principe : Détermination de H

- Reprenons l'exemple vu précédemment

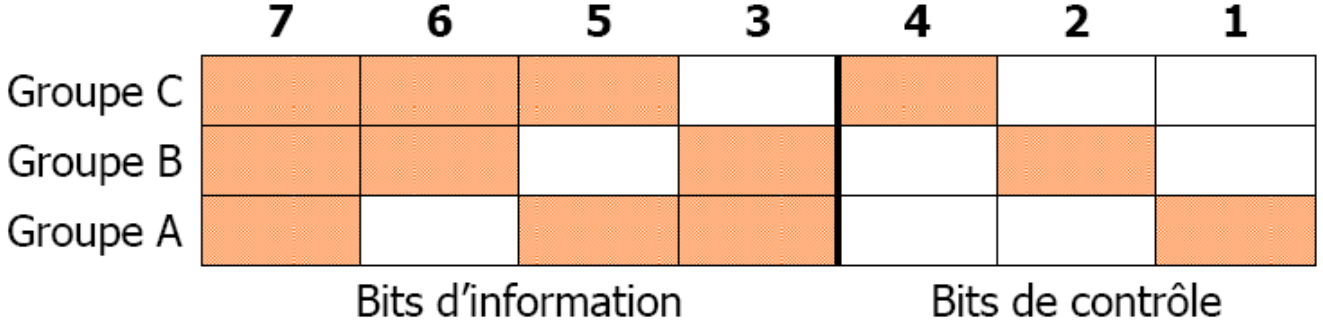
H =

1	1	1	0	1	0	0
1	1	0	1	0	1	0
1	0	1	1	0	0	1

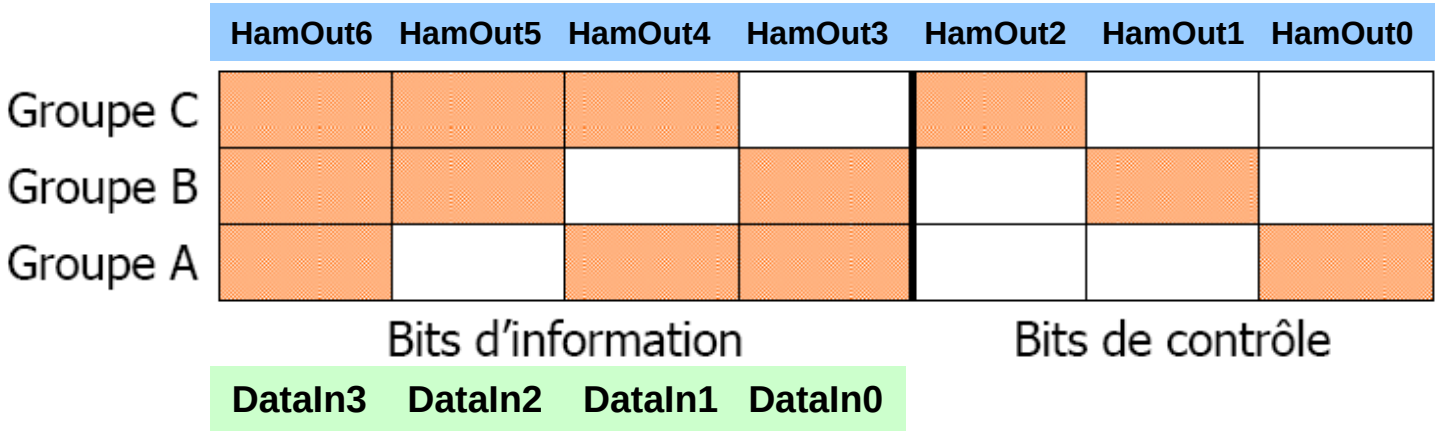
G1 =

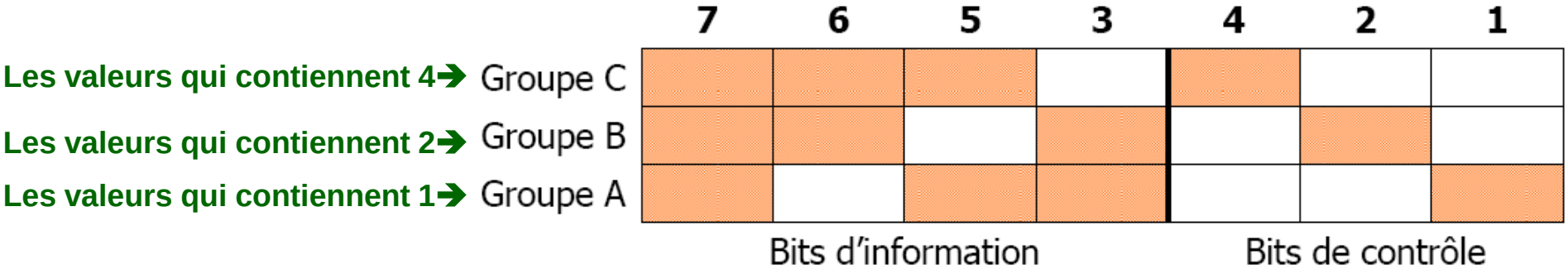
1	0	0	0	1	1	1
0	1	0	0	1	1	0
0	0	1	0	1	0	1
0	0	0	1	0	1	1

- Comment construire la matrice de contrôle de parité H ?
- Rappel: 4 bits en entrée et 7 bits en sortie ➔ H aura (7-4)=3 lignes et 7 colonnes:

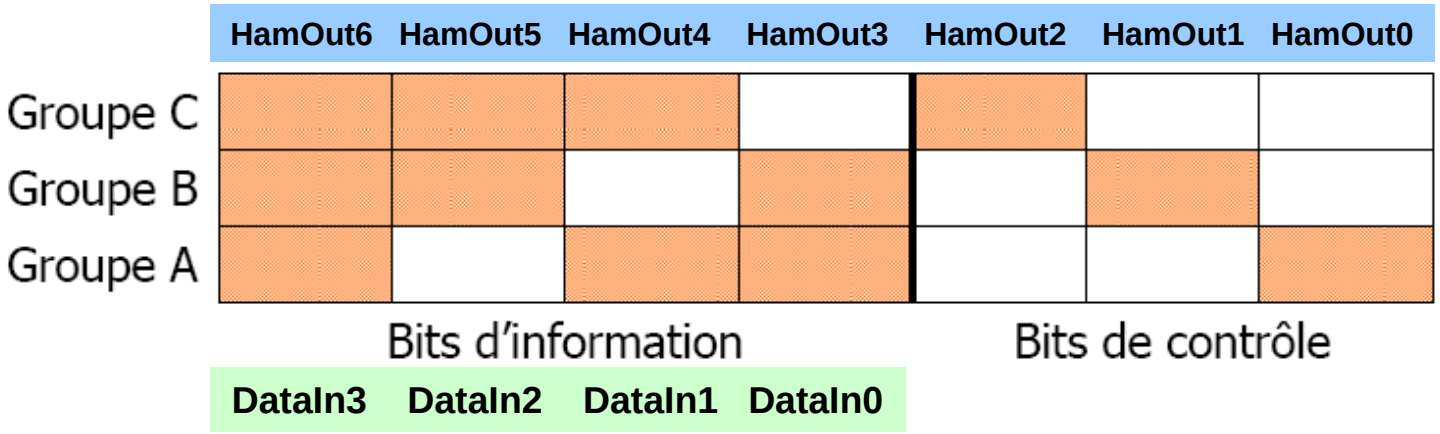


Principe : Equations émetteur





Principe : Equations émetteur



- On peut donc en déduire les équations suivantes:
  - **HamOut** (7-Bits dans notre application) étant le mot du code généré à partir de la chaîne de bits d'information **DataIn** (4-Bits dans notre application).

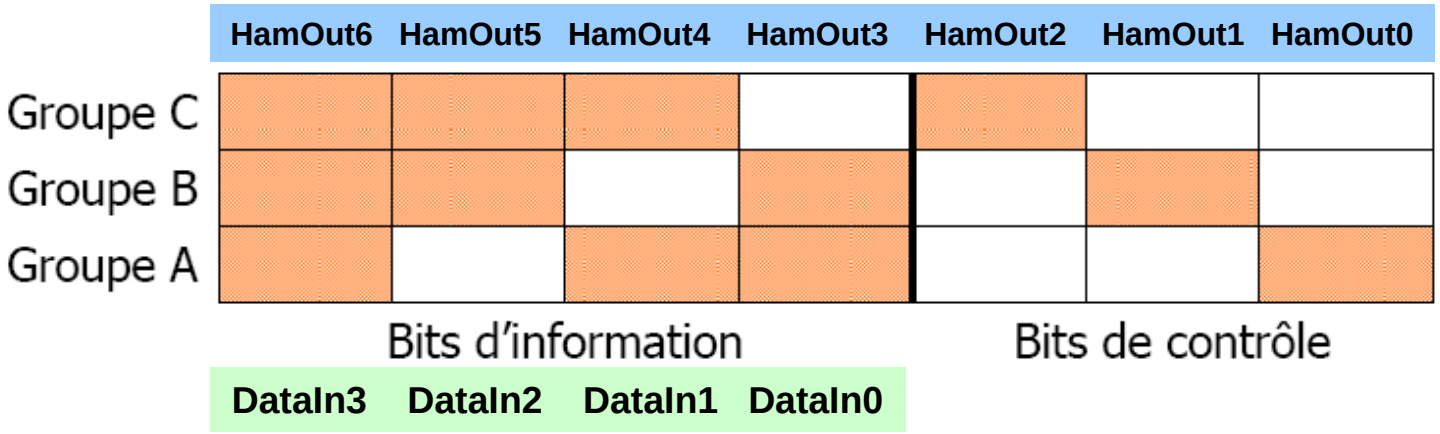
Groupe A: $HamOut_0 = DataIn_0 \oplus DataIn_1 \oplus DataIn_3$

Groupe B: $HamOut_1 = DataIn_0 \oplus DataIn_2 \oplus DataIn_3$

Groupe C: $HamOut_2 = DataIn_1 \oplus DataIn_2 \oplus DataIn_3$

Respect de la parité paire (par exemple HamOut0 est le bit de parité pour le groupe A)

Principe : Equations émetteur



- Exemple: **0 1 1 0**
- D'où:
  - HamOut0 = 0 XOR 1 XOR 0 = 1
  - HamOut1 = 0 XOR 1 XOR 0 = 1
  - HamOut2 = 1 XOR 1 XOR 0 = 0

$$HamOut_0 = DataIn_0 \oplus DataIn_1 \oplus DataIn_3$$
$$HamOut_1 = DataIn_0 \oplus DataIn_2 \oplus DataIn_3$$
$$HamOut_2 = DataIn_1 \oplus DataIn_2 \oplus DataIn_3$$

- Finalement: HamOut[6..0] = **0 1 1 0 0 1 1**

Principe : Codes en sortie de l'émetteur

- 4 bits en entrée ➔16 codes en sortie sur 7 bits !

Bits d'information	Bits de contrôle
0000	000
0001	011
0010	101
0011	110
0100	110
0101	101
0110	011
0111	000

Bits d'information	Bits de contrôle
1000	111
1001	100
1010	010
1011	001
1100	001
1101	010
1110	100
1111	111

Principe : Equations récepteur

- On détecte une erreur en contrôlant la parité de chacun des groupes de la matrice de contrôle de parité. On en détermine une valeur nommée « syndrome » (3-bits pour notre application):
  - **HamIn** étant le mot entrant dans le décodeur (7-Bits pour notre application).

Groupe A: $Syndrome_0 = HamIn_3 \oplus HamIn_4 \oplus HamIn_6 \oplus HamIn_0$

Groupe B: $Syndrome_1 = HamIn_3 \oplus HamIn_5 \oplus HamIn_6 \oplus HamIn_1$

Groupe C: $Syndrome_2 = HamIn_4 \oplus HamIn_5 \oplus HamIn_6 \oplus HamIn_2$

Principe : Equations récepteur

- Si le « syndrome » est égale à “000”, on assume qu’il n’y a pas d’erreur.
- Vérification avec l’exemple précédent :
  - HamIn[6..0] =HamOut[6..0] = **0 1 1 0 0 1 1**
  - On a :
    - Syndrome0 = 0 XOR 1 XOR 0 XOR 1 = 0
    - Syndrome1 = 0 XOR 1 XOR 0 XOR 1 = 0
    - Syndrome2 = 1 XOR 1 XOR 0 XOR 0 = 0

$$Syndrome_0 = HamIn_3 \oplus HamIn_4 \oplus HamIn_6 \oplus HamIn_0$$
$$Syndrome_1 = HamIn_3 \oplus HamIn_5 \oplus HamIn_6 \oplus HamIn_1$$
$$Syndrome_2 = HamIn_4 \oplus HamIn_5 \oplus HamIn_6 \oplus HamIn_2$$

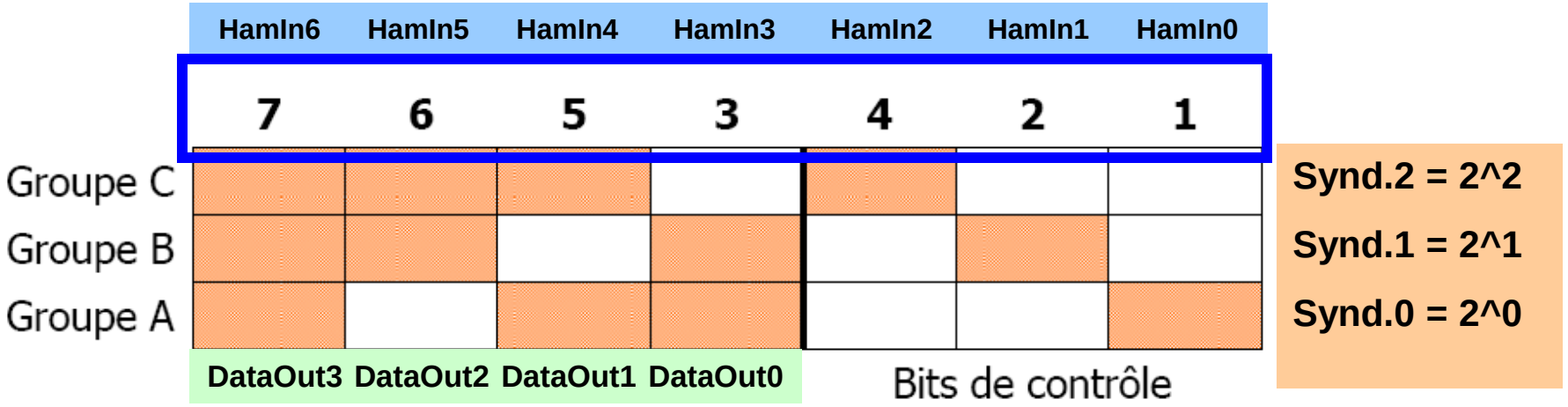
Principe : Equations récepteur

- Finalement, il y a erreur si Erreur = 1:

Erreur = (Syndrome0) OR (Syndrome1) OR (Syndrome2)

Principe : Equations récepteur

- Un « syndrome » différent de “000” indique une erreur et, dans ce cas, la valeur du « syndrome » correspond à l’indice du Bit de **HamIn** qui est faux, en se référant à la numérotation des bits de la table de contrôle de parité



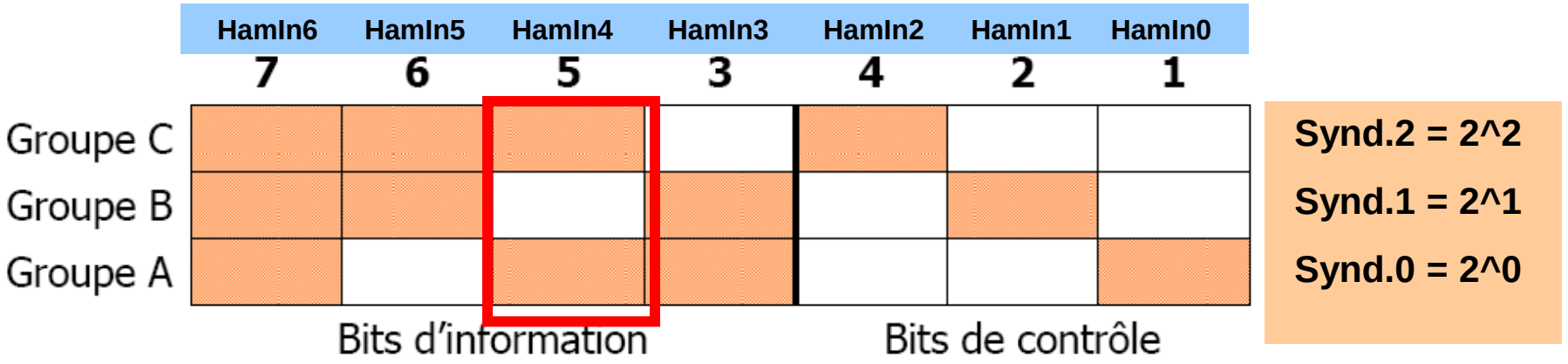
Principe : Equations récepteur

- Exemple précédent: sans erreur: HamIn[6..0] = **0 1 1 0 0 1 1**
- Avec une erreur sur le 5ieme bit on a:

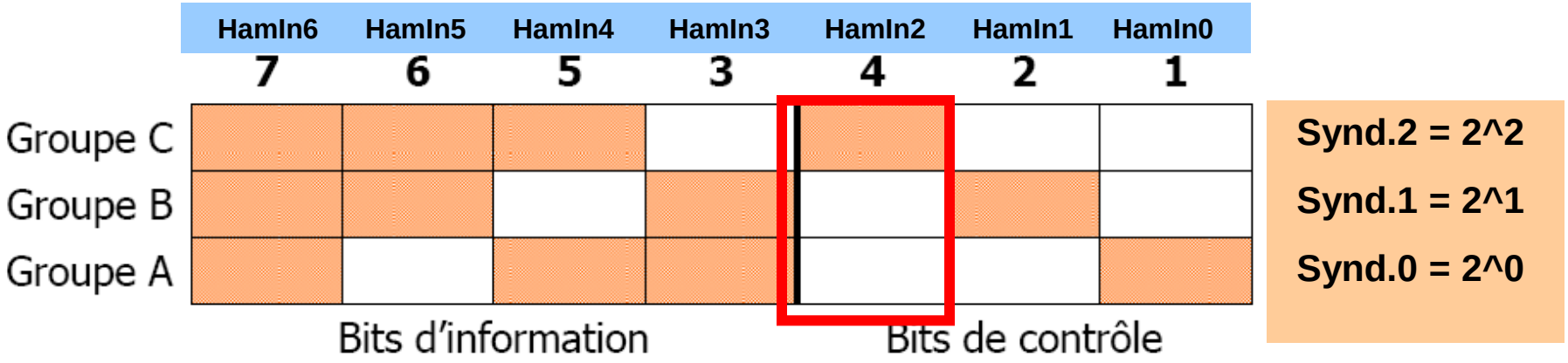
  HamIn[6..0] = **0 1 0 0 0 1 1**
- Le calcul du syndrome nous donne:
  - Syndrome0 = 0 XOR 0 XOR 0 XOR 1 = 1
  - Syndrome1 = 0 XOR 1 XOR 0 XOR 1 = 0
  - Syndrome2 = 0 XOR 1 XOR 0 XOR 0 = 1

$$Syndrome_0 = HamIn_3 \oplus HamIn_4 \oplus HamIn_6 \oplus HamIn_0$$
$$Syndrome_1 = HamIn_3 \oplus HamIn_5 \oplus HamIn_6 \oplus HamIn_1$$
$$Syndrome_2 = HamIn_4 \oplus HamIn_5 \oplus HamIn_6 \oplus HamIn_2$$

Principe : Equations récepteur



- On a bien trouvé le bit erroné !
- Autre exemple: Imaginons le **syndrome = 100**



Script « Source.vhd »

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY Source IS
PORT(
      Clk : IN std_logic;
      Reset : IN std_logic;
      HamOut : OUT std_logic_vector (6 downto 0);
      Segout : OUT std_logic_vector (6 downto 0)
      );
END Source ;

ARCHITECTURE struct OF Source IS
SIGNAL Count : std_logic_vector(3 DOWNTO 0);

COMPONENT Counter
PORT (
      clk : IN std_logic ;
      reset : IN std_logic ;
      count : OUT std_logic_vector (3 DOWNTO 0));
END COMPONENT;

COMPONENT Encoder
PORT (
      datain : IN std_logic_vector (3 DOWNTO 0);
      hamout : OUT std_logic_vector (6 DOWNTO
0));
END COMPONENT;

COMPONENT BinTo7Seg IS
PORT(
      binin : IN std_logic_vector (3 DOWNTO 0);
      segout : OUT std_logic_vector (6 DOWNTO 0)
      );
END COMPONENT ;

BEGIN
I2 : BinTo7Seg PORT MAP (Count, Segout);
I1 : Counter PORT MAP (Clk, Reset, Count);
I0 : Encoder PORT MAP (Count, HamOut);
END struct;
```

Script « Counter.vhd »

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

ENTITY Counter IS
PORT(
      Clk : IN std_logic;
      Reset : IN std_logic;
      Count : OUT std_logic_vector (3 DOWNTO 0));
END Counter ;

ARCHITECTURE Synth OF Counter IS
      signal Count_int: std_logic_vector(3 downto 0);
BEGIN
      process (Clk, Reset)
      begin
                  if (Reset = '1') then
                              Count_int <= "0000";
                  elsif (Clk'event and Clk = '1') then
                              Count_int <= Count_int
+ 1;
                  end if;
      end process;
      Count <= Count_int;
END Synth;
```

Script « Encoder.vhd »

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY Encoder IS
PORT(
    DataIn : IN std_logic_vector (3 DOWNTO 0);
    HamOut : OUT std_logic_vector (6 DOWNTO 0));
END Encoder ;
```

```
ARCHITECTURE Synth OF Encoder IS
BEGIN
    HamOut (6 downto 3) <= DataIn;
    HamOut(0) <= DataIn(0) XOR DataIn(1) XOR DataIn(3);
    HamOut(1) <= DataIn(0) XOR DataIn(2) XOR DataIn(3);
    HamOut(2) <= DataIn(1) XOR DataIn(2) XOR DataIn(3);
END Synth;
```

Script « BinTo7Seg.vhd »

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY BinTo7Seg IS
PORT(
binin : IN std_logic_vector (3 DOWNTO 0);
segout : OUT std_logic_vector (6 DOWNTO 0)
);
END BinTo7Seg ;
```

```
ARCHITECTURE Synth OF BinTo7Seg IS
BEGIN
    -- SegOut(6) -> a, SegOut(5) -> b, ...
    -- 0 = On, 1 = Off
    SegOut <= "0000001" WHEN BinIn= "0000"
ELSE
    "1001111" WHEN BinIn= "0001" ELSE
    "0010010" WHEN BinIn= "0010" ELSE
    "0000110" WHEN BinIn= "0011" ELSE
    "1001100" WHEN BinIn= "0100" ELSE
    "0100100" WHEN BinIn= "0101" ELSE
    "1100000" WHEN BinIn= "0110" ELSE
    "0001111" WHEN BinIn= "0111" ELSE
    "0000000" WHEN BinIn= "1000" ELSE
    "0001100" WHEN BinIn= "1001" ELSE
    "0001000" WHEN BinIn= "1010" ELSE
    "1100000" WHEN BinIn= "1011" ELSE
    "0110001" WHEN BinIn= "1100" ELSE
    "1000010" WHEN BinIn= "1101" ELSE
    "0110000" WHEN BinIn= "1110" ELSE
    "0111000" WHEN BinIn= "1111" ELSE
    "XXXXXXX";
END Synth;
```

Script « ErrorBit.vhd »

Script « Destination.vhd »

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY Destination IS
PORT(
      HamIn : IN std_logic_vector (6 DOWNTO 0);
      Corr : OUT std_logic;
      SegOut : OUT std_logic_vector (6 DOWNTO
0));
END Destination ;

ARCHITECTURE struct OF Destination IS
      SIGNAL DataOut : std_logic_vector(3
DOWNTO 0);
```

```
COMPONENT BinTo7Seg
PORT (
      binin : IN std_logic_vector (3 DOWNTO 0);
      segout : OUT std_logic_vector (6 DOWNTO 0)
      );
END COMPONENT;

COMPONENT Decoder
PORT (
      hamin : IN std_logic_vector (6 DOWNTO 0);
      corr : OUT std_logic ;
      dataout : OUT std_logic_vector (3 DOWNTO 0)
      );
END COMPONENT;

BEGIN
I1 : BinTo7Seg PORT MAP (DataOut, SegOut);
I0 : Decoder PORT MAP (HamIn, Corr, DataOut);

END struct;
```

Script « Decoder.vhd »

Conclusion générale

Conclusion générale:

- Les codes en blocs linéaires et les codes convolutifs sont la base des CCE.
- Il existe des stratégies de codage qui permettent une efficacité de correction encore plus performante pour des rapports du signal sur bruit très faibles.
- BIBLIOGRAPHIE:
  - http://lapwww.epfl.ch/courses/archord1/CorrErr.pdf
  - http://www.irisa.fr/prive/bcousin/Cours/G3.pdf
  - http://home.gna.org/dscratch/documentation/tx5x-rosener-controle-crc.pdf
  - http://docs.sylvain-nahas.com/crc.html