IPB
ENSEIRB
MATMECA
BORDEAUX

Module EN321

Systèmes embarqués pour les télécommunications

Année 2013-2014 / Camille LEROUX

UNIVERSITÉ DE BORDEAUX

Objectifs du module

- **Se familiariser avec les traitements en bande de base**
- **Attention particulière sur la partie codage de canal**
  - **Codes correcteurs d'erreurs (convolutifs, blocs)**
  - **Scrambleur (LFSR)**
  - **Entrelaceurs**
- **Implantation matérielle (VHDL) d'une partie de la chaîne de transmission de la norme DVB-T**
  - **Rédaction du cahier des charges**
  - **Implantation et validation VHDL des différents blocs**

Plan du module EN113

- **Codage de canal**
- **Codes en blocs (Hamming, Reed-Solomon, etc..)**
- **Codes convolutifs**
- **LFSR**
- **Chaîne d'émission de la norme DVB-T**
  - **Structure des trames**
  - **Génération des données**
  - **Scrambler**
  - **Code RS**
  - **Codes convolutifs**

Codage de canal

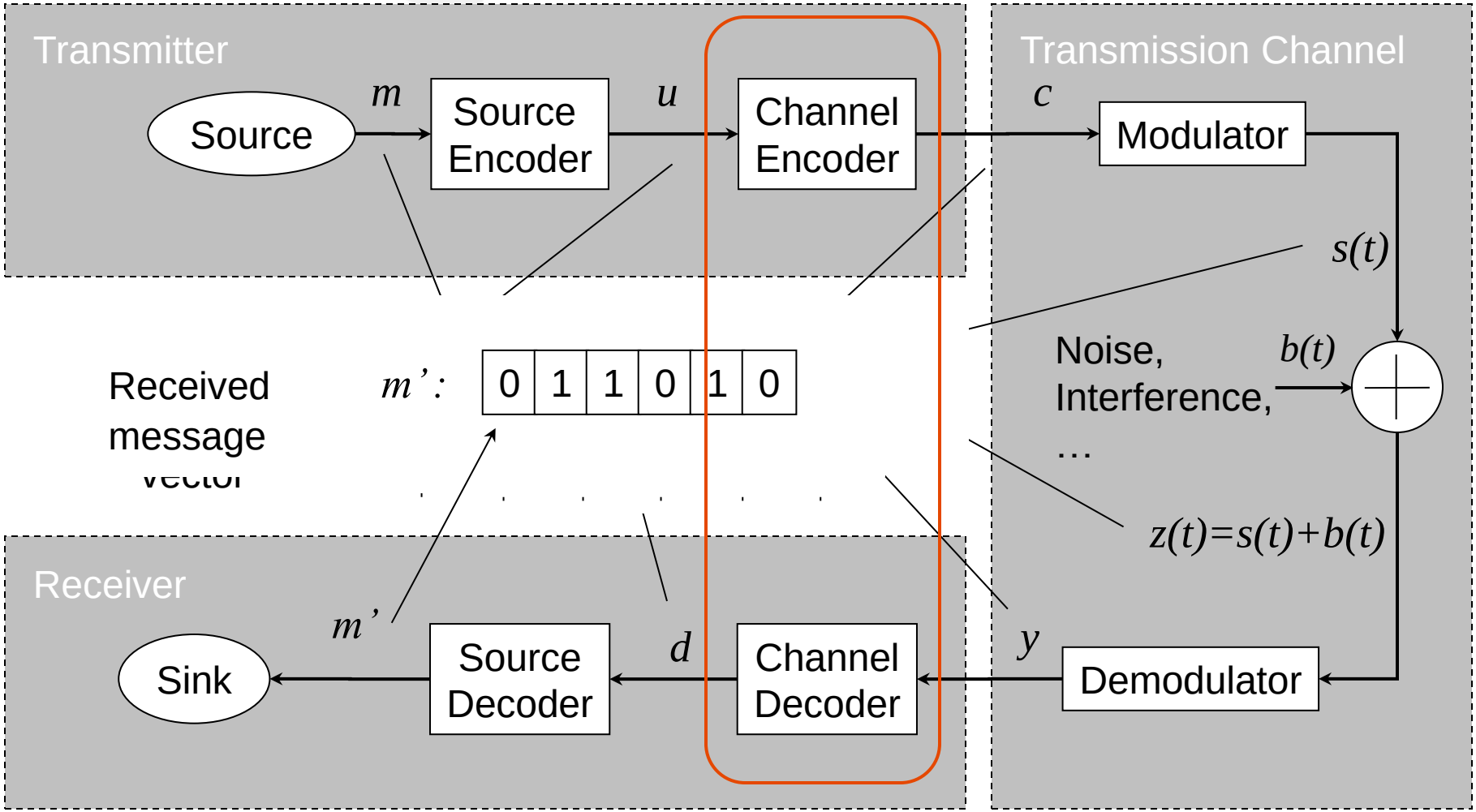
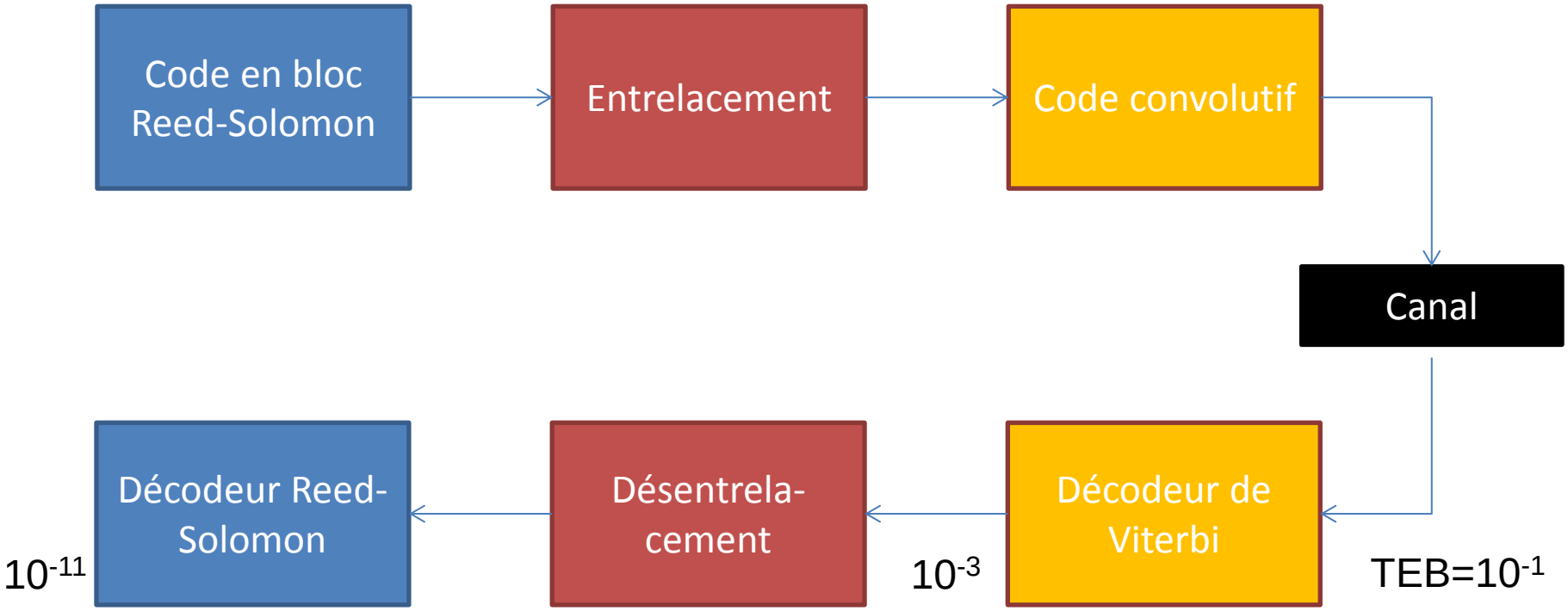
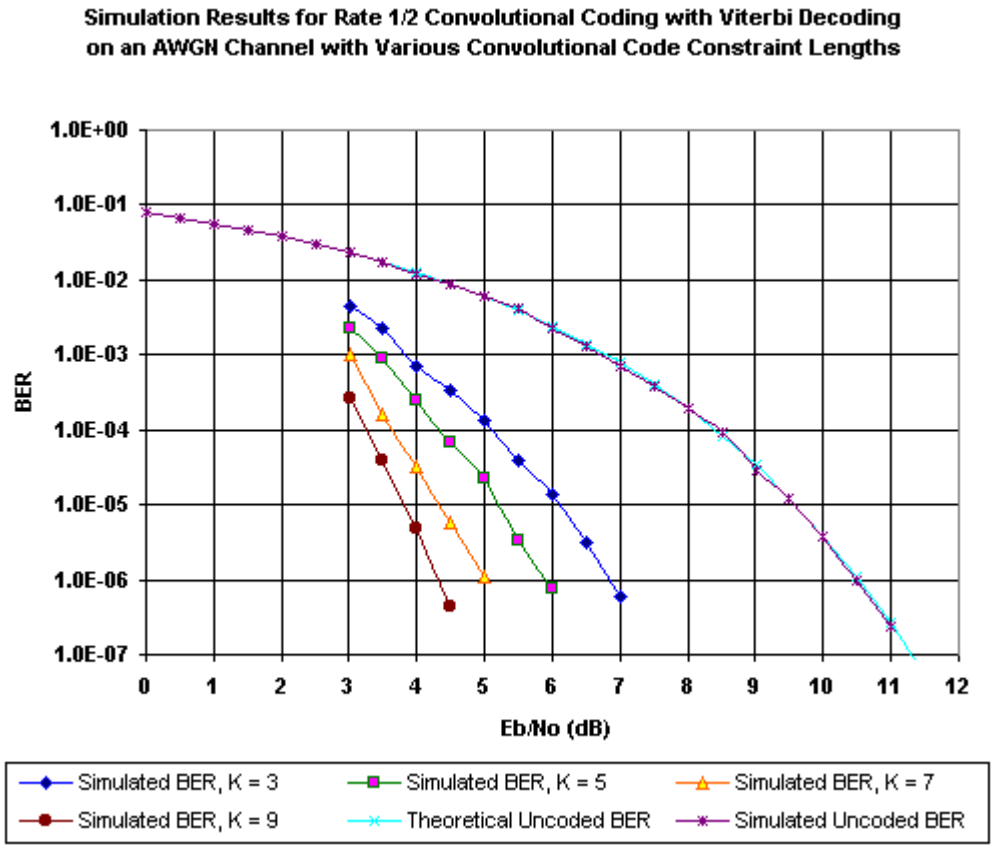


Schéma classique de codage de canal



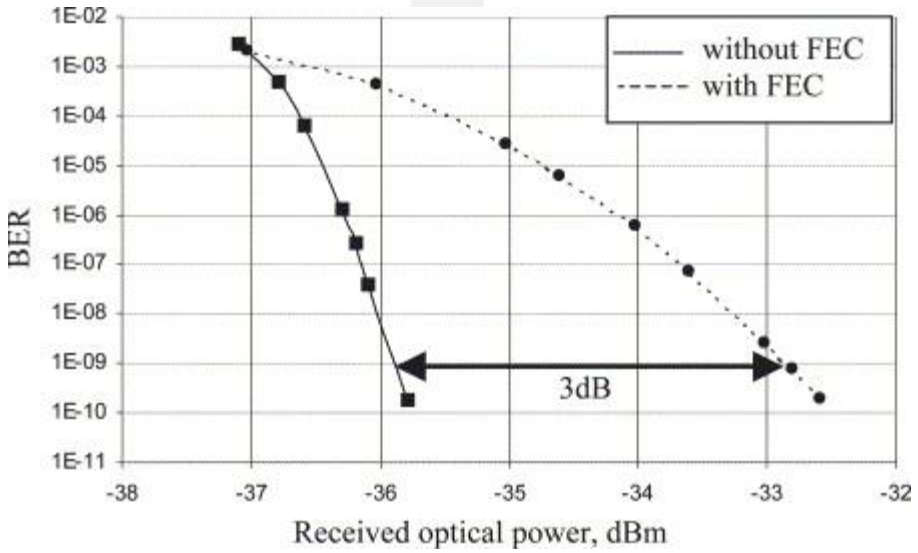
Intérêt du codage convolutif



Code convolutif + VITERBI:

- permet de tirer parti de l'information sur la fiabilité des décisions prises
- Ramène le TEB à 10^{-3}
- Laisse des *bursts* d'erreurs en sortie

Intérêt du codage Reed-Solomon



Code Reed-Solomon:

- Est efficace surtout quand TEB d'entrée < 10^{-2}
- Corrige les erreurs par bursts
- Complémentaire avec les codes convolutifs

Entrelacement

ThisIsAnExampleOfInterleaving...

ThisIs_____pleOfInterleaving...

ThisIsAnExampleOfInterleaving...

TIEpfeaghsxlIrv.iAaenli.snmoten.

TIEpfe_____Irv.iAaenli.snmoten.

T_isI_AnE_amp_eOfInterle_vin_...

Entrelacement:

- Permet de disperser les erreurs
- Permet de compenser en partie les effacements et évanouissement
- Augmente la latence du système

Codes en blocs

Binaires ou non-binaires
K : dimension du code
N : taille du code
R=K/N :rendement du code
d : distance minimum du code
t : pouvoir de correction du code

Exemple:
Code de Hamming (7,4,1) : Code binaire N=7, K=4, t=1
RS(255,239,8) : Code non-binaire N=255, K=239, R=0.937, t=8

Code à répétition: on répète le message à envoyer 3 fois. A la réception on effectue un vote à majorité pour chaque bit:
R=1/3, t=1

Existe-t-il des méthode de codage plus efficace ?

Code de Hamming

Soit **G** une matrice génératrice systématique de dimension 7x4
On utilise cet matrice pour encoder un message **u** en un mot de code **x**, tel que $\mathbf{x=uG^t}$
Le mot de code est ensuite transmis sur un canal bruité
En supposant que le canal rajoute au plus, une seule erreur, le mot recu est $\mathbf{x'=x+e_i}$
$\mathbf{e_i}$ est le vecteur d'erreur de taille **n.** Il est non nul uniquement à la position **i**
Le but du décodage est de retrouver la position de l'erreur à partir de x' et de la connaissance de **G**.
Pour cela, on définit une matrice de parité **H** telle que **HG=0**
A la réception, on calcule: $\mathbf{Hx'} = \mathbf{H(x+e_i)} = \mathbf{Hx + He_i} = \mathbf{HGu + He_i} = \mathbf{0 + He_i}$

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

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

La matrice **H** est de dimension 3x7
Chaque colonne est la représentation binaire de la position de l'erreur **i**.
Code de Hamming : n=7, k=4, R=0.571, t=1

Codes Reed-Solomon

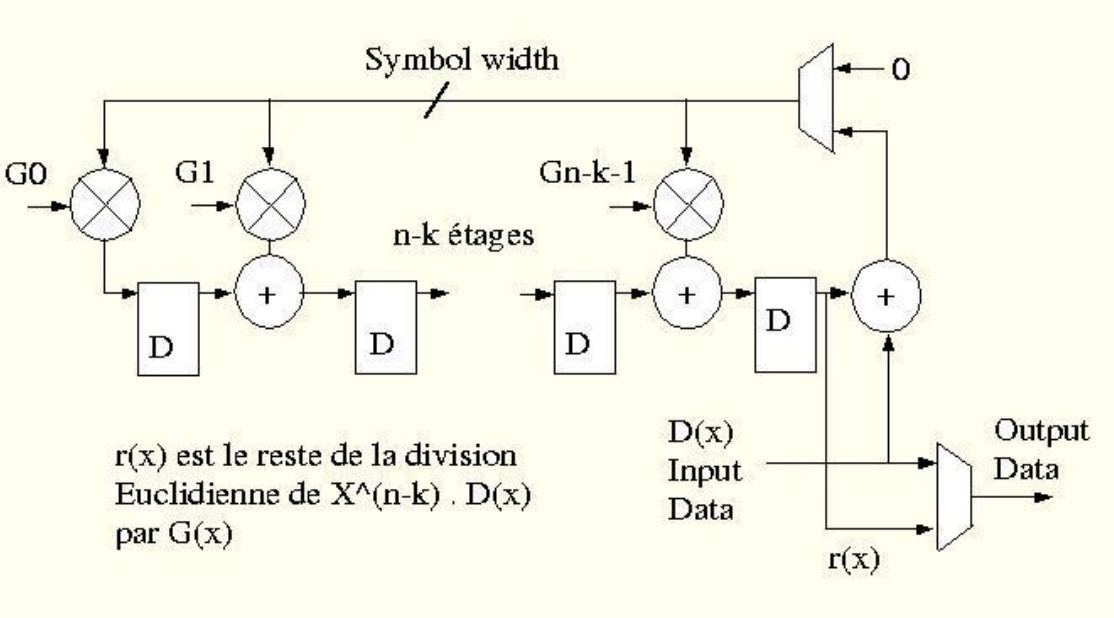
Codes à symboles non-binaires
Chaque symbole est représenté sur ***m*** bits, et prend sa valeur dans le corps de Galois **GF(2^m)**
Les codes RS disposent d'une structure algébrique basée sur les corps de Galois.
Algorithmes de décodage de complexité raisonnable.
Paramètres d'un code RS corrigeant ***t*** erreurs dans un bloc de n symboles:

$n = 2^m - 1$
$n - k = d_{min} - 1 = 2t$
$k = n - 2t$

Exemple: code DVB-S est basé sur le code RS(255,239,8) qui est ensuite raccourci à RS(204,188,8)

Les codes RS sont bons pour corriger des bursts d'erreurs (8 symboles de 8 bits par exemple)
Ils sont utilisés dans les CD/DVD, les communications optiques, etc…

Encodeur Reed-Solomon



- C'est un LFSR dans un Corps de Galois
- Multiplieurs de Galois
- Additionneurs de Galois
- Bascules D

Autres codes en blocs

Codes BCH (protection des pilotes dans DVB)

- version binaire des codes RS
- extension du code de Hamming
- faible complexité d'encodage et de décodage

Codes LDPC

- codes ayant une matrice de parité creuse
- décodage itératif à décision souple
- très bonnes performances
- complexes à implémenter

Turbocodes / turbocodes produits

- basés sur une concaténation + entrelacement
- très bonnes performances
- norme Wimax (optionnel), DVB-RCS

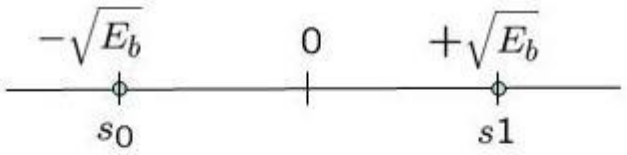
Codes polaires

- plus exotiques mais prometteurs

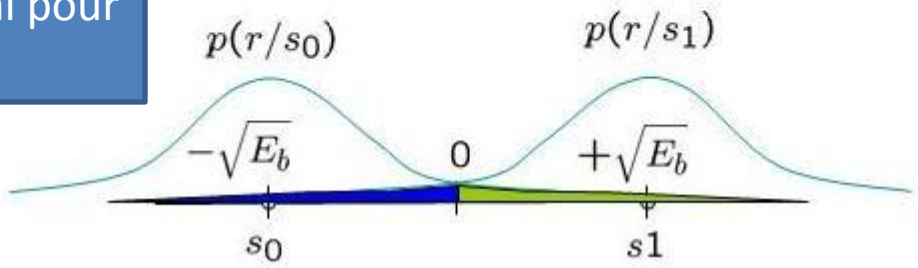
ETC…

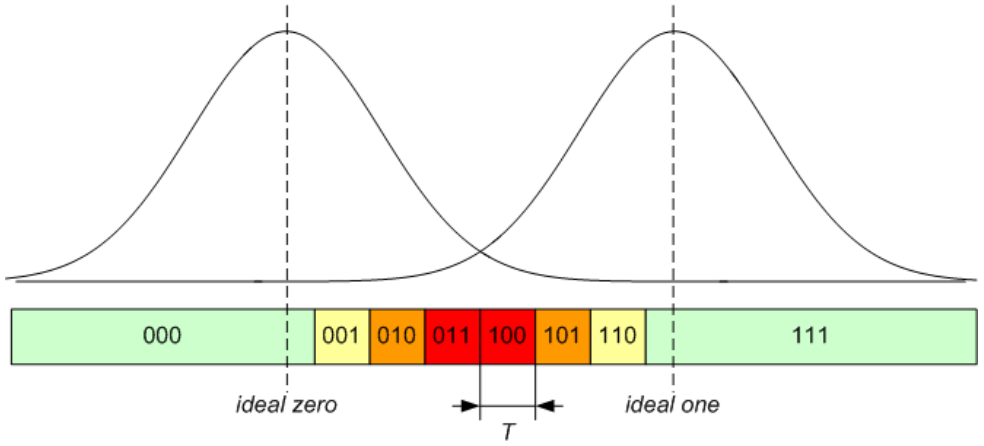
Décision dure versus décision souple

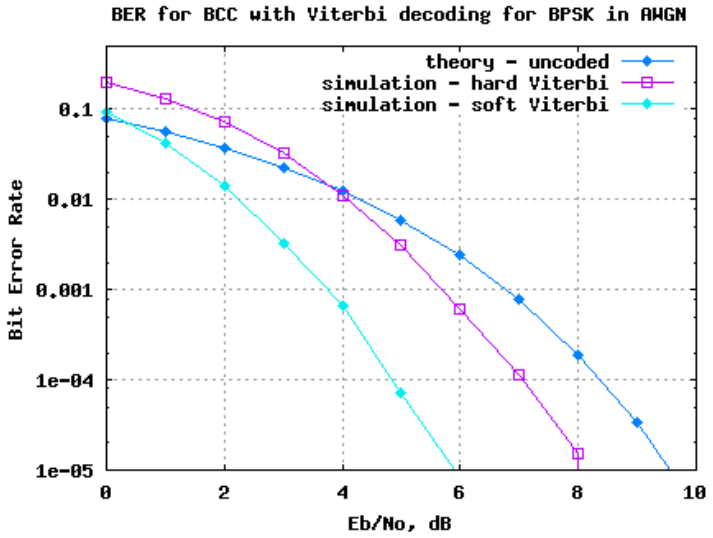
Canal Gaussien + BPSK:



Au lieu de faire un seuillage à la réception, le décodeur tient compte du niveau du signal pour quantifier la fiabilité du symbole reçu.

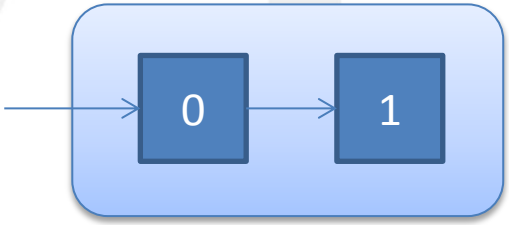






Code convolutif

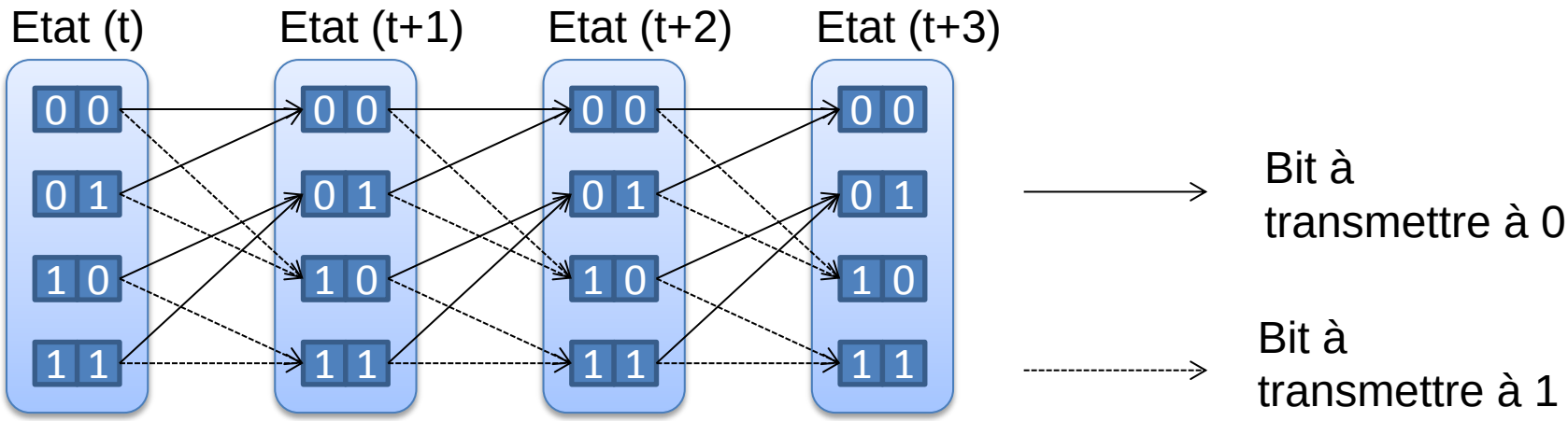
Bit à transmettre



Longueur de contrainte K=3 (1 bit à transmettre + 2 bits d'état)

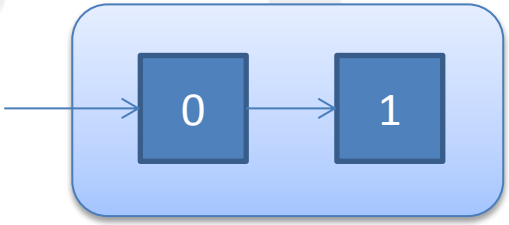
Etat du codeur à l'instant t

L'évolution de l'état du codeur est représenté par un treillis:



Code convolutif

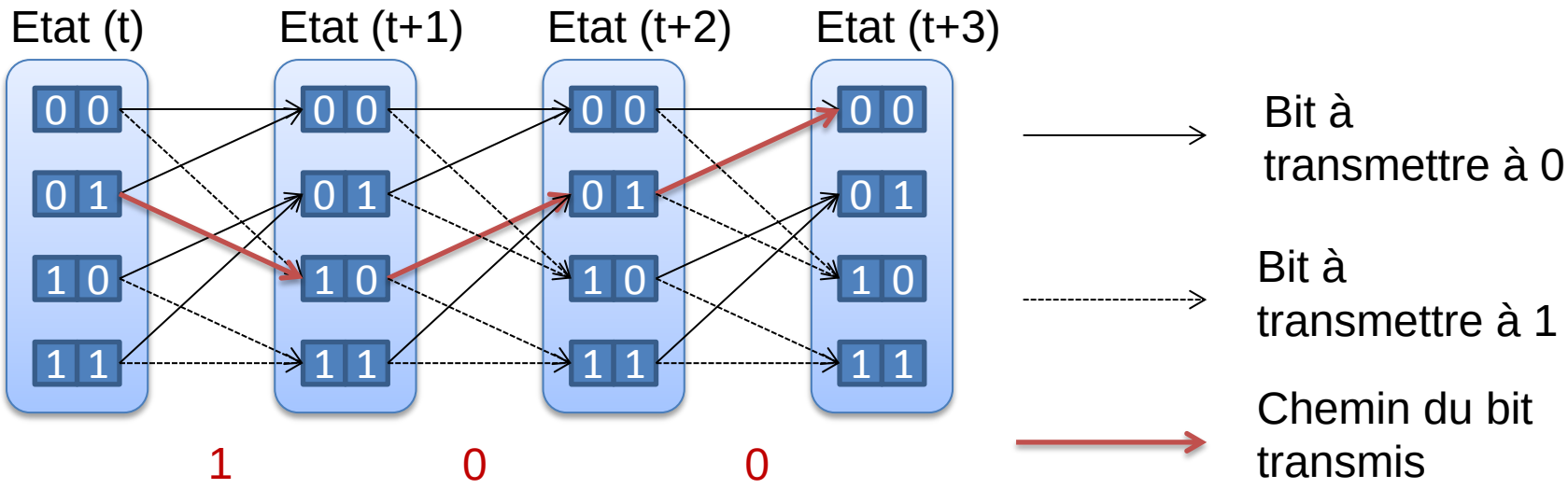
Bit à transmettre



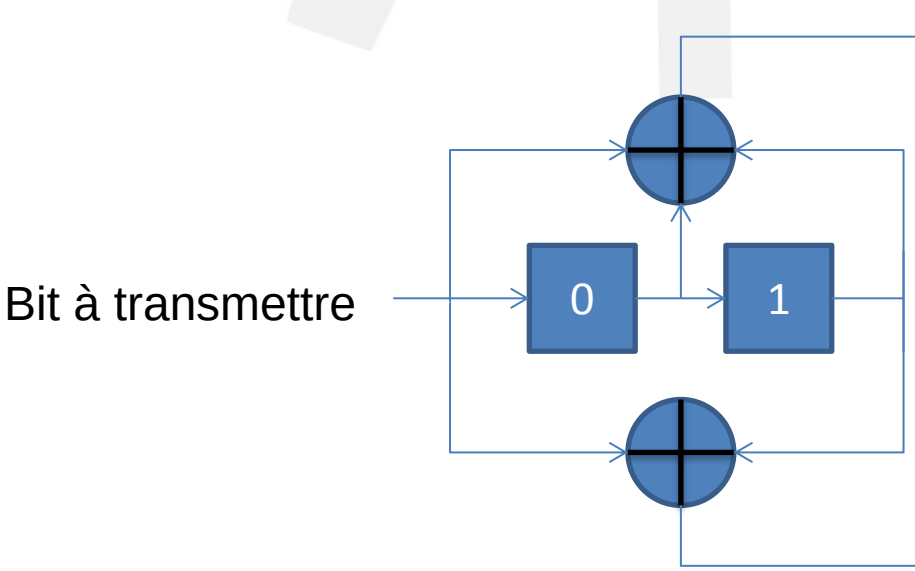
Longueur de contrainte K=3 (1 bit à transmettre + 2 bits d'état)

Etat du codeur à l'instant t

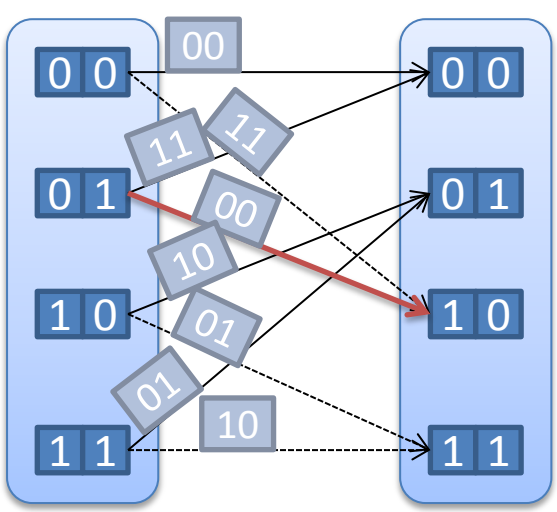
L'évolution de l'état du codeur est représenté par un treillis:

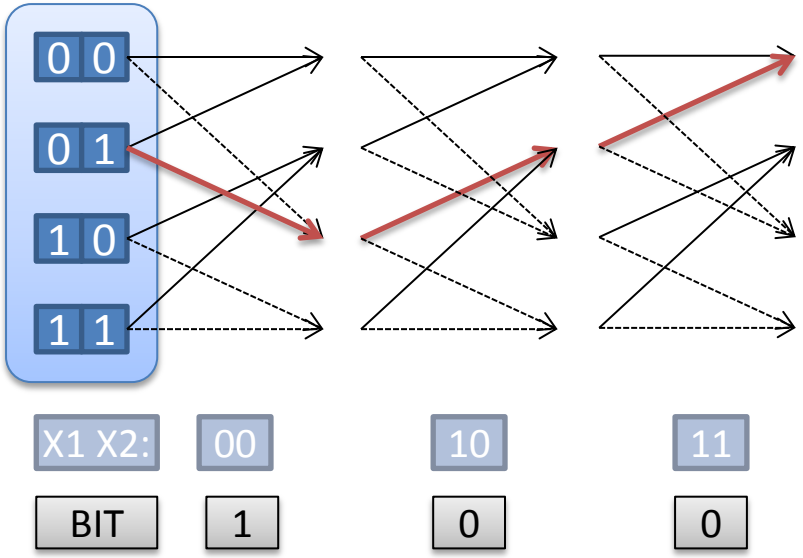


Code convolutif



Les sorties X1 et X2 représentent la signature des transitions du bit à transmettre (redondance => rendement 1/2)



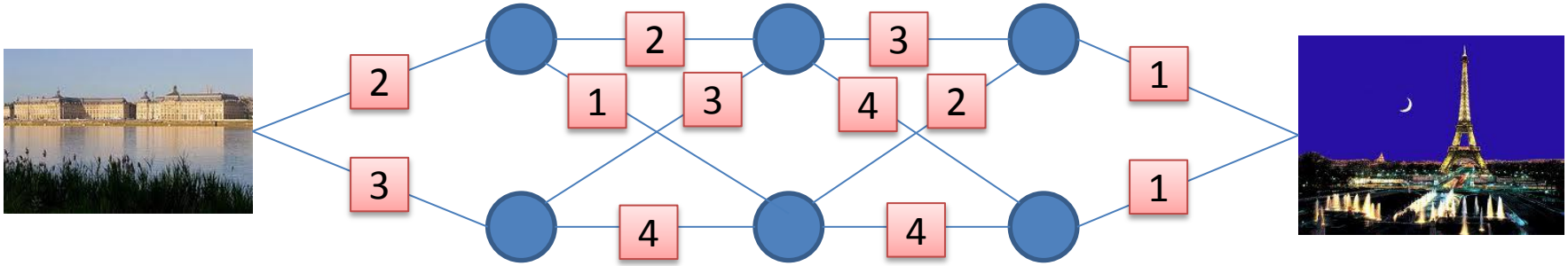


Algorithmes de décodage des codes convolutifs

- Viterbi (décision souple en entrée, dure en sortie)
  - Chemin survivant optimal
- SOVA (décision souple en entrée, souple en sortie)
  - Fiabilité approximative, complexité réduite
  - Permet de décoder les « turbo-codes »
- Forward-Backward (décision souple en entrée, souple en sortie)
  - Fiabilité exacte des bits décodés
  - +0.5dB de mieux que le SOVA

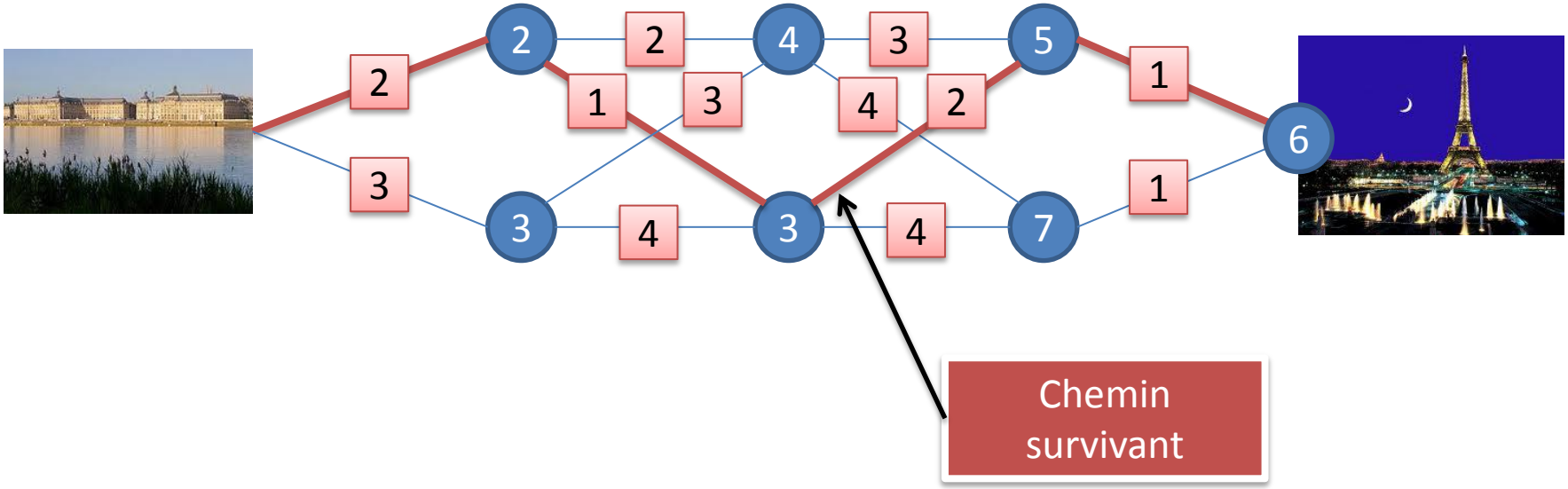
Décodeur de Viterbi

- Le décodage consiste à retrouver le chemin parcouru par le bit dans le treillis en utilisant une de recherche de maximum de vraisemblance



Décodeur de Viterbi

- A chaque nœud, la distance parcourue la plus courte est calculée et le chemin pris est mémorisé. Il faut arriver à destination pour en déduire le chemin le plus court = chemin survivant

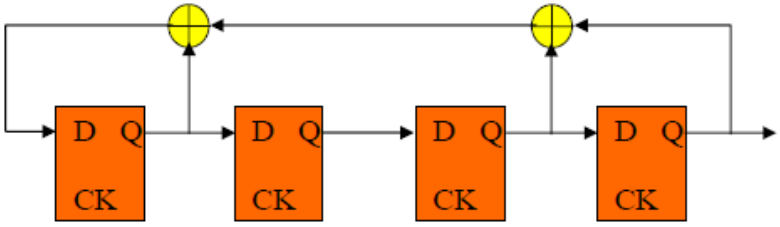


Linear Feedback Shift Registers (LFSR)

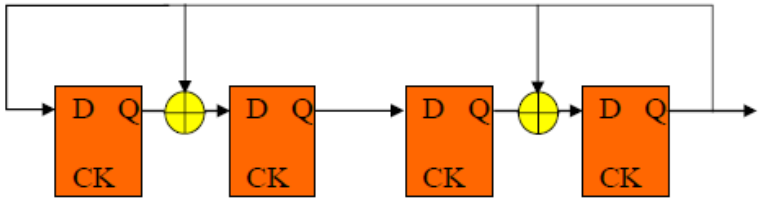
- Génération efficace de vecteurs de test
  - Flip-Flops + quelques XORs
  - Plus efficace qu'un compteur (moins de portes, fréquence plus élevée)
- CRC
- Encodeur Reed-Solomon / BCH
- Génération de séquences pseudo-aléatoire

- Deux types de LFSR
  - Boucle de retour externe (external feedback)
  - Boucle de retour interne (internal feedback) : fréquence plus élevée

External Feedback LFSR

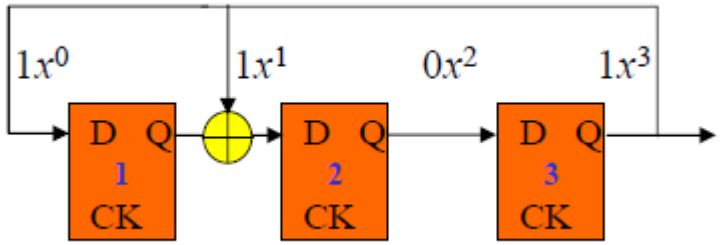


Internal Feedback LFSR



LFSR (2)

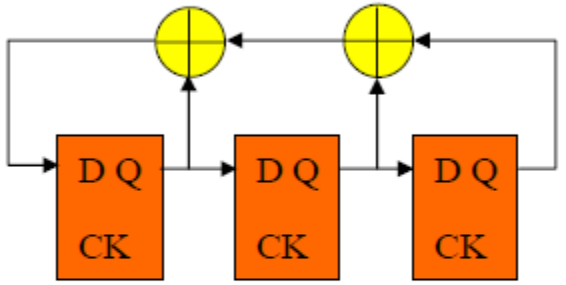
- Exemple: $P(x) = x^3 + x + 1$
- Etat de départ : tout à '1'
  - Le cycle complet dure 7 cycles d'horloges
  - Longueur maximale = 2^n-1
  - P(x) est un polynôme primitif



1	1	1	1
1	0	1	2
1	0	0	3
0	1	0	4
0	0	1	5
1	1	0	6
0	1	1	7
1	1	1	

- $P(x) = x^3 + x^2 + x + 1$

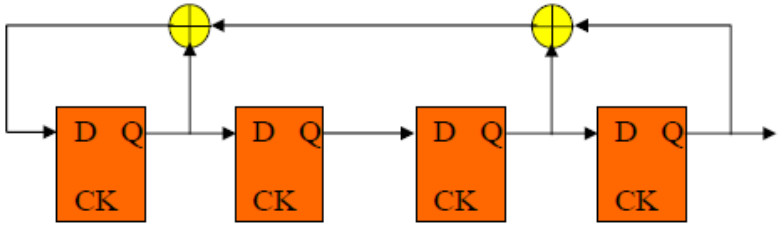
External Feedback LFSR



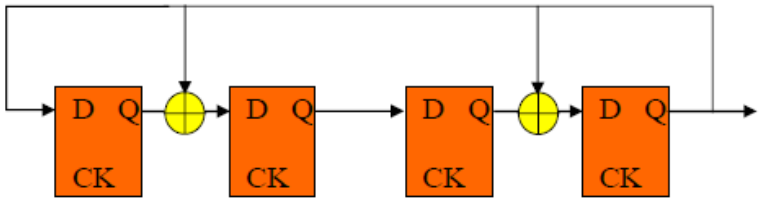
0	0	0	
0	0	0	
1	1	1	
1	1	1	
1	0	0	
1	1	0	
0	1	1	
0	0	1	
1	0	0	
0	1	0	
1	0	1	
0	1	0	

LFSR (3)

- Un LFSR est caractérisé par un ***polynôme***
  - Le polynôme définit la position des XOR
  - $P(x) = x^4+x^3+x+1$
  - n = nombre de Flip-flops = degré du polynôme



External Feedback LFSR



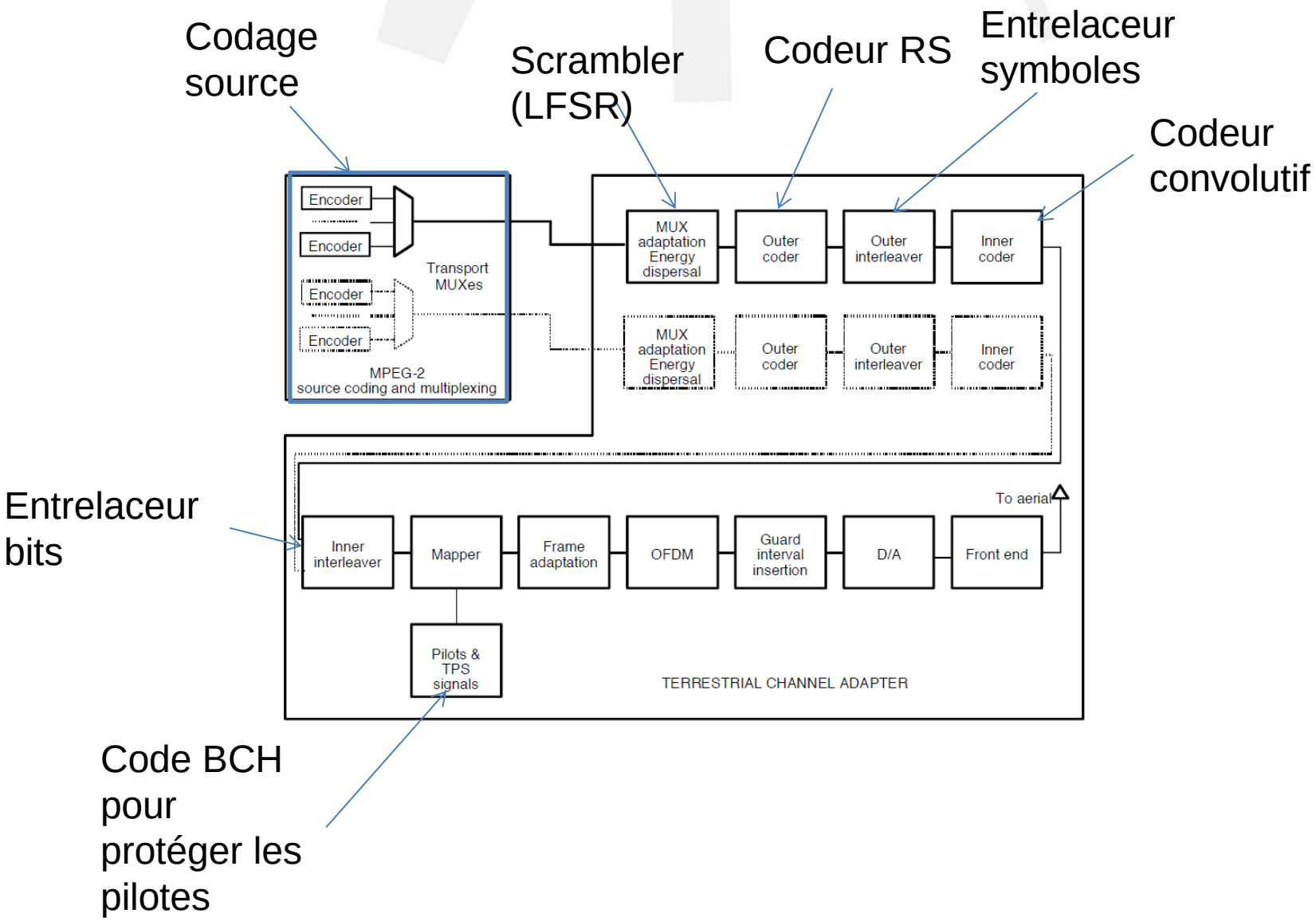
Internal Feedback LFSR

- Un LFSR génère une séquence périodique
  - Il DOIT partir d'un état non-nul sinon … que des zéros
- La longueur maximale de la séquence est 2^n-1
  - La séquence tout à zéro n'est pas générée (état bloquant)
  - Le polynôme caractéristique d'un LFSR générant une séquence de longueur maximale est un polynôme primitif
- Une séquence de longueur maximale est pseudo-aléatoire:
  - Nombre de '1' = nombre de '0' + 1

Polynômes primitifs avec un minimum de XORs

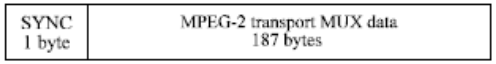
Degree (n)	Polynomial
2,3,4,6,7,15,22	$x^n + x + 1$
5,11,21,29	$x^n + x^2 + 1$
8,19	$x^n + x^6 + x^5 + x + 1$
9	$x^n + x^4 + 1$
10,17,20,25,28	$x^n + x^3 + 1$
12	$x^n + x^7 + x^4 + x^3 + 1$
13,24	$x^n + x^4 + x^3 + x + 1$
14	$x^n + x^{12} + x^{11} + x + 1$
16	$x^n + x^5 + x^3 + x^2 + 1$
18	$x^n + x^7 + 1$
23	$x^n + x^5 + 1$
26,27	$x^n + x^8 + x^7 + x + 1$
30	$x^n + x^{16} + x^{15} + x + 1$

Le codage de canal dans la norme DVB-T



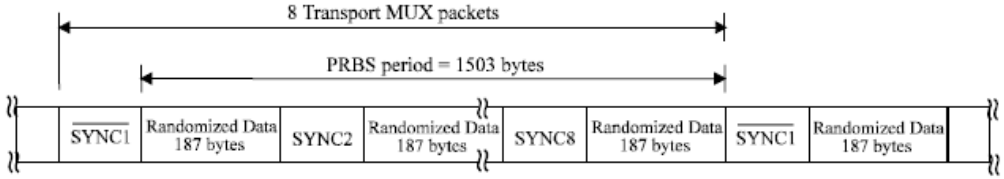
Structure des paquets et octets de synchro

Paquet MPEG-2 = 188 octets = 1 octet SYNC + 187 octets DATA



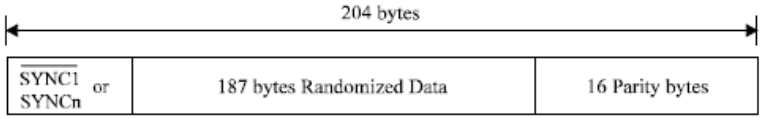
a) MPEG-2 transport MUX packet

- SYNC = 47_{HEX} ou la version complémenté not(SYNC)=$B8_{HEX}$ (tous les 8 paquets)
- L'envoi de SYNC et not(SYNC) se fait « MSB first »
- Les données sont « randomisées » Pr(0)=Pr(1)



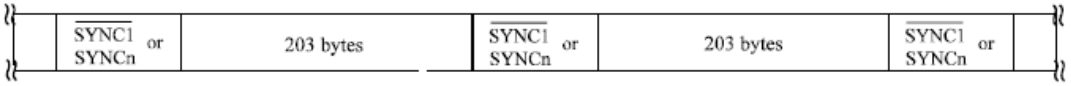
b) Randomized transport packets: Sync bytes and Randomized Data bytes

On ajoute de la redondance pour protéger les données : RS(204,188,8) ???



c) Reed-Solomon RS(204,188,8) error protected packets

Rajout de la redondance + entrelacement des octets

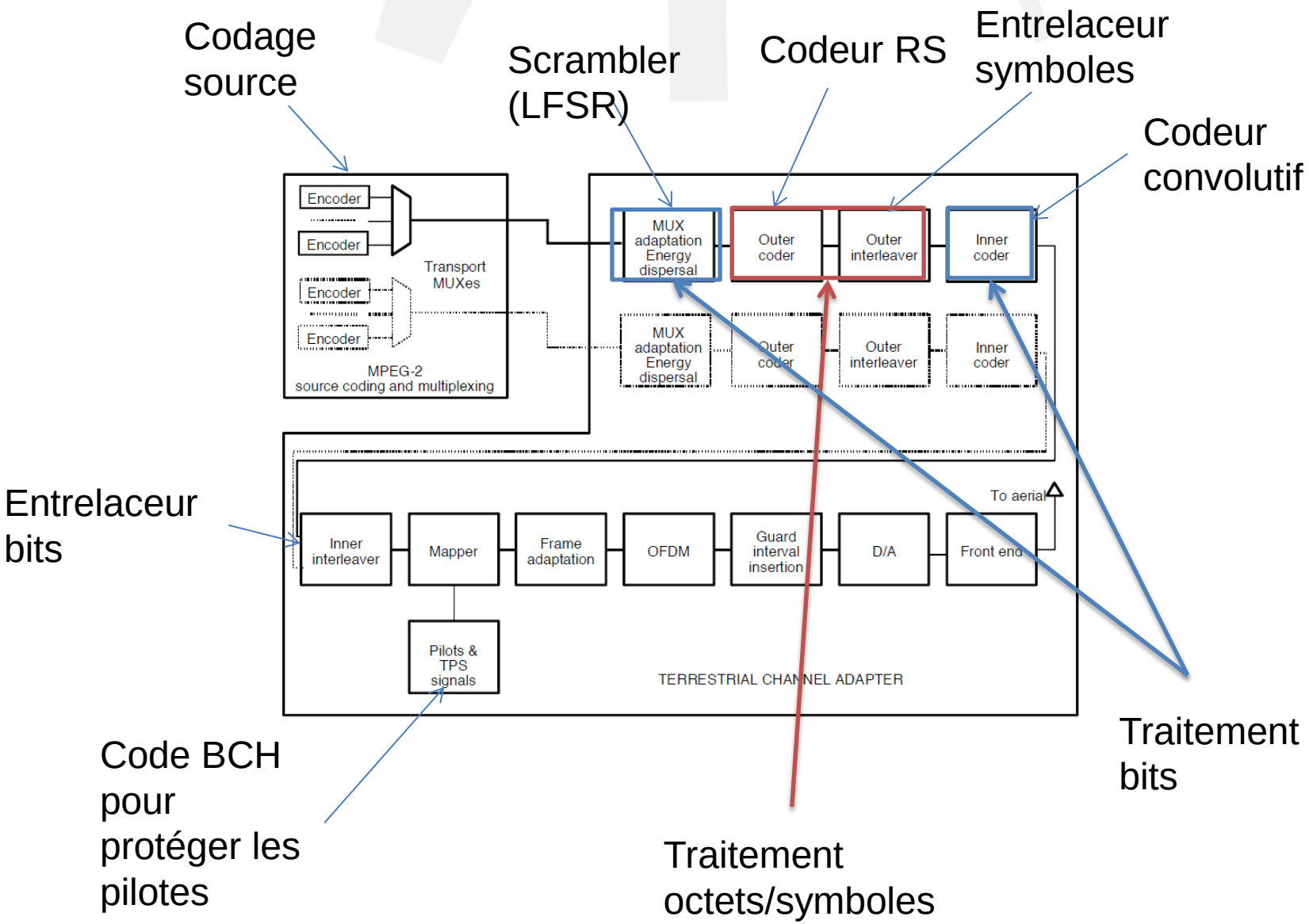


d) Data structure after outer interleaving; interleaving depth I = 12 bytes

SYNC1: Non randomized complemented sync byte
SYNCn: Non randomized sync byte, n = 2, 3, ..., 8

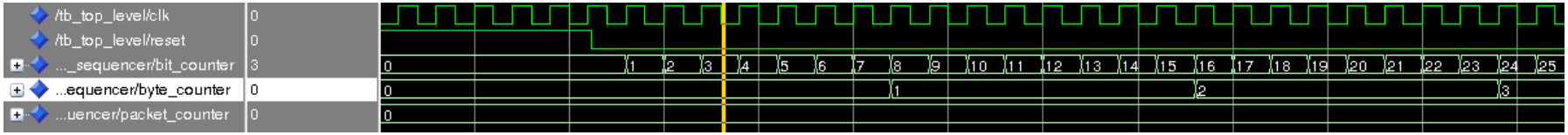
Figure 3: Steps in the process of adaptation, energy dispersal, outer coding and interleaving

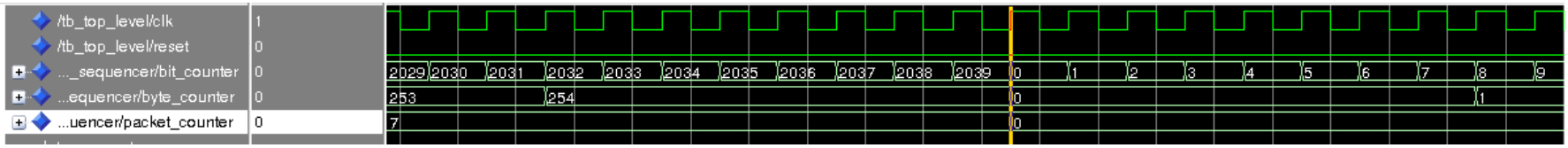
Quelle taille pour le flux de données ?



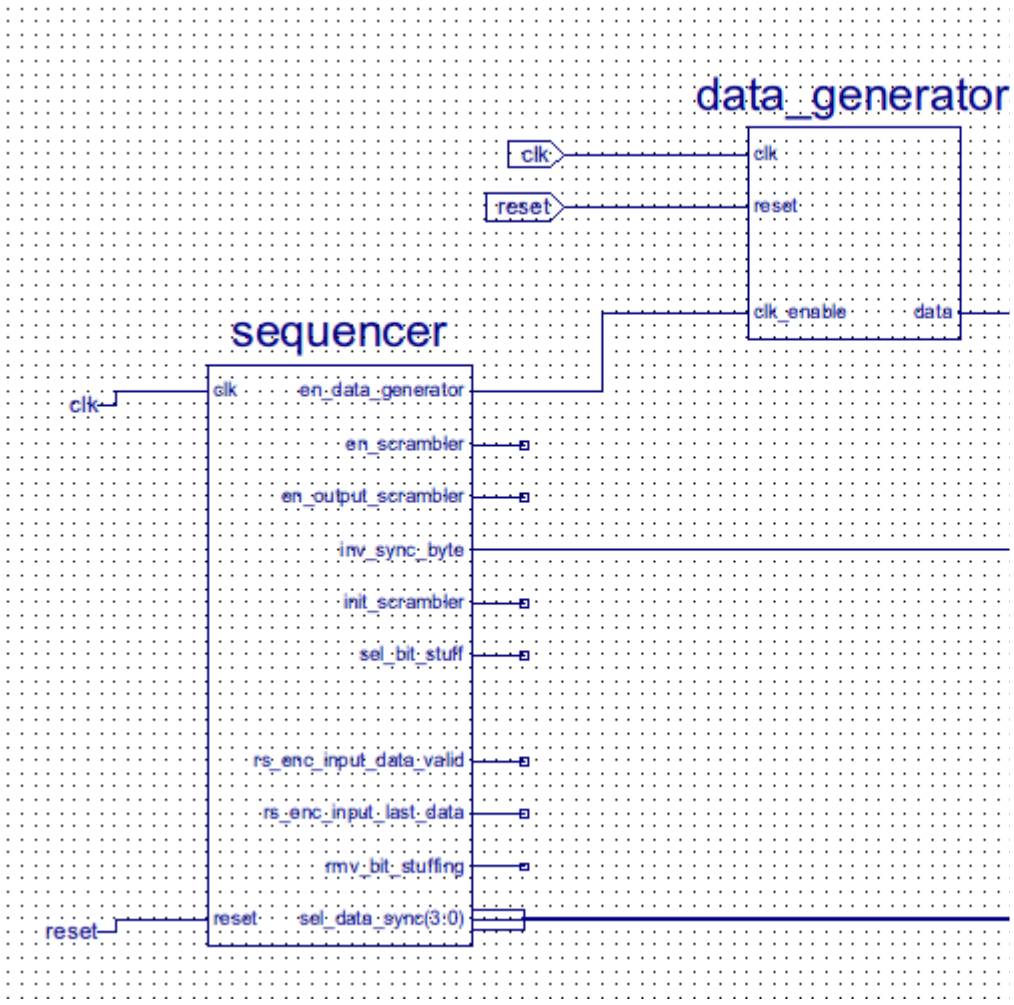
Séquencement

- Le séquenceur contient trois compteurs:
  - **Un compteur-bit:** modulo 1632 incrémenté à chaque cycle d'horloge
  - **Un compteur octet:** modulo 204 incrémenté tous les 8 cycles
  - **Un compteur paquet**: modulo 8 incrémenté tous les 204 octets = 1632 bits
- Les signaux de contrôles doivent être générés à partir de la valeur de ces trois compteurs



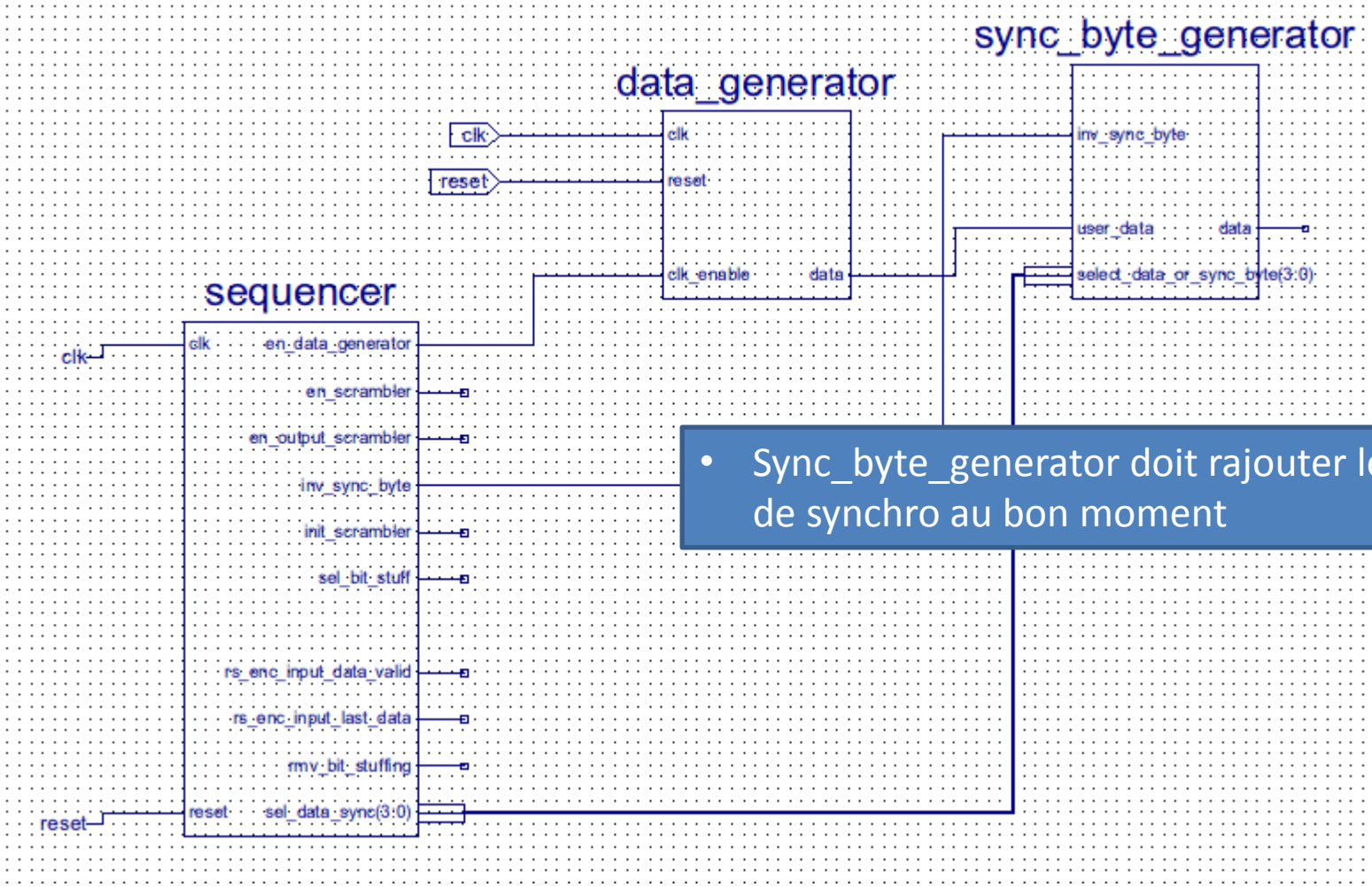


Génération des données



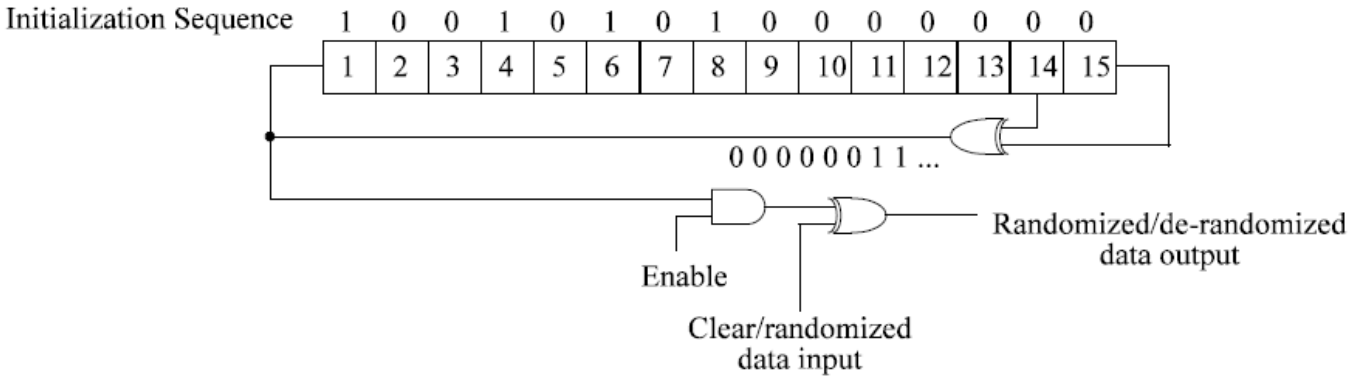
- Data_generator: fourni
- Sequenceur : fourni partiellement
- Les autres blocs : boites noires à remplir ou à créer

Octet de synchro



- Sync_byte_generator doit rajouter les octets de synchro au bon moment

Scrambler



Polynôme P(x) = ?

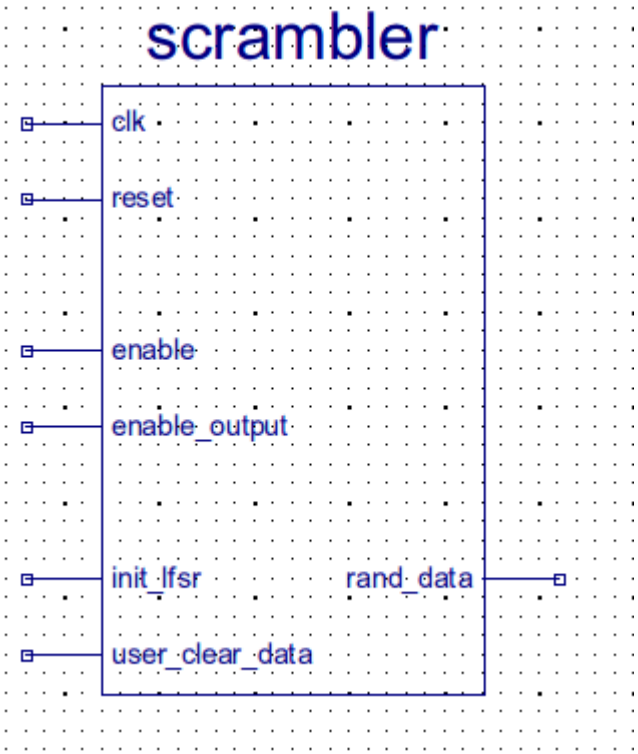
Data input (MSB first): 10111000xxxxxxxx...

PRBS sequence: 00000011...

Figure 2: Scrambler/descrambler schematic diagram

- Le premier bit généré par le LFSR doit être appliqué au premier bit du premier octet suivant le premier octet de synchro
- La séquence d'initialisation doit être chargée dans le LFSR/PRBS tous les 8 paquets
- Durant l'envoi des 7 autres octets de synchro le LFSR doit continuer d'évoluer mais la sortie doit être désactivée (enable=0)

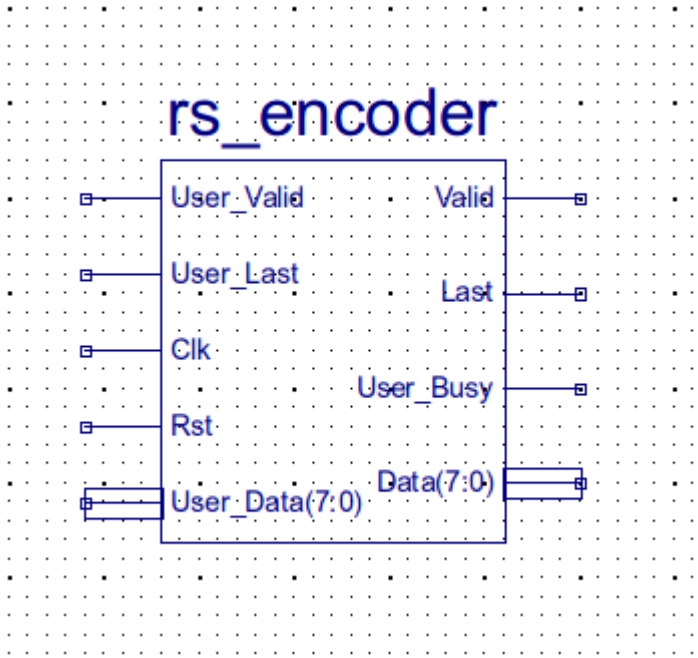
Scrambler



- Bloc séquentiel ou combinatoire ? Latence ?
- enable : gèle les flip-flop (= clk_enable)
- enable_output : active ou non la sortie (entrée de la porte ET sur le schéma)
- init_lfsr: permet d'activer la réinitialisation du LFSR
- user_clear_data: données « non randomisées »
- rand_data : données « randomisées »

Codeur RS

- Le standard requiert l'utilisation du code RS(204,188,8)
- Vous n'avez pas à écrire ce bloc, une IP vous est fournie **MAIS** l'IP est un codeur RS pour le code RS(255,239,8)
- Cette IP prend les symboles codées sur 8 bits en entrée
  - => deserializer
- Il faut envoyer des 0 pour compléter le vecteur d'information
  - => bit_stuffing (simple multiplexeur commandé par le signal *sel_bit_stuff*) => latence plus importante
  - => initialiser le LFSR à zero



Entrelaceur

Il faut simplement écrire en VHDL ce que vous voyez !

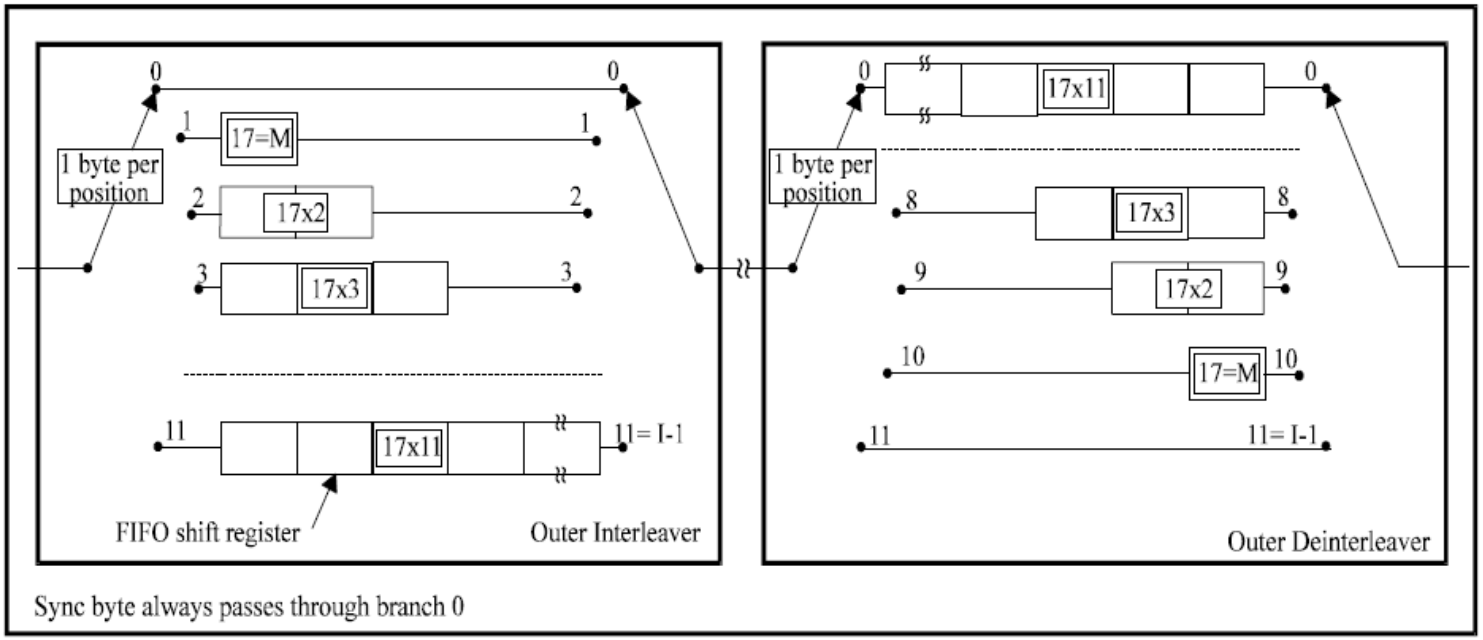


Figure 4: Conceptual diagram of the outer interleaver and deinterleaver

Codeur convolutif

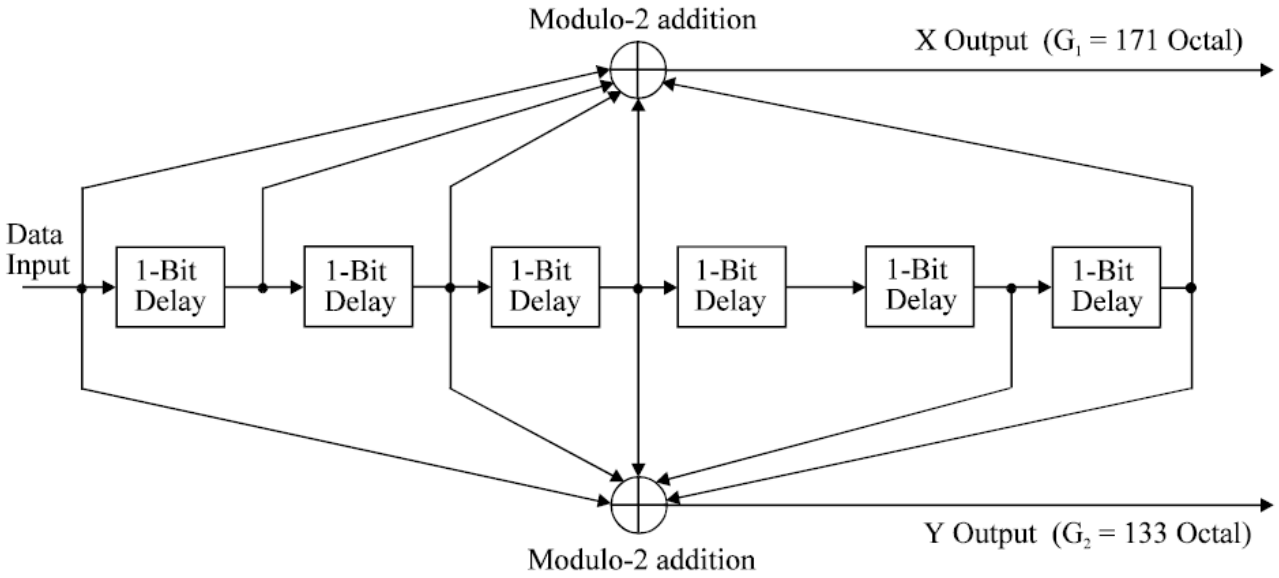


Figure 5: The mother convolutional code of rate 1/2

Table 2: Puncturing pattern and transmitted sequence after parallel-to-serial conversion for the possible code rates

Code Rates r	Puncturing pattern	Transmitted sequence (after parallel-to-serial conversion)
1/2	X: 1 Y: 1	$X_1 Y_1$
2/3	X: 1 0 Y: 1 1	$X_1\ Y_1\ Y_2$
3/4	X: 1 0 1 Y: 1 1 0	$X_1\ Y_1\ Y_2\ X_3$
5/6	X: 1 0 1 0 1 Y: 1 1 0 1 0	$X_1\ Y_1\ Y_2\ X_3\ Y_4\ X_5$
7/8	X: 1 0 0 0 1 0 1 Y: 1 1 1 1 0 1 0	$X_1\ Y_1\ Y_2\ Y_3\ Y_4\ X_5\ Y_6\ X_7$

Projet
Etape 1 : Spécification

- *Objectif: Décrire et valider (en VHDL) le début de la chaîne d'émission de la norme DVB-T: du scrambleur jusqu'au codeur convolutif*

Ecrire une spécification détaillée du système:

1. Dessiner le chronogramme de la chaîne décrite dans le standard (traitement bit/octet ?, horloge système ?, latence de chaque bloc ?, conversion série/parallèle)
2. Déterminer les blocs nécessaires et les inclure dans le schéma de la chaîne du standard
3. Pour chaque bloc:
   - Schéma des entrées/sorties
   - Description textuelle/schématique du fonctionnement
   - Chronogramme complet (horloge, signaux de contrôles, data in/out, latence)
4. Définir la procédure de vérification
   - Testbenches séparés ou testbench unique ?
   - Verif Amont/Aval ?

Projet
Etape 2 : Description/validation

- *Objectif: Décrire et valider (en VHDL) le début de la chaîne d'émission de la norme DVB-T: du scrambleur jusqu'au codeur convolutif*

Description du système

1. Pour chaque bloc:
    1. Ecrire le testbench
    2. Décrire le bloc en VHDL
    3. Valider le fonctionnement
    4. Mettre à jour la spécification
    5. Mesurer le temps passé sur chaque tâche et noter les difficultés rencontrées
    6. Faire valider par l'enseignant