# Turbo codes (convolutifs)

Timisoara

15-17 mars 2004

Catherine Douillard,

Claude Berrou et Michel Jézéquel, ENST Bretagne

Catherine.Douillard@enst-bretagne.fr

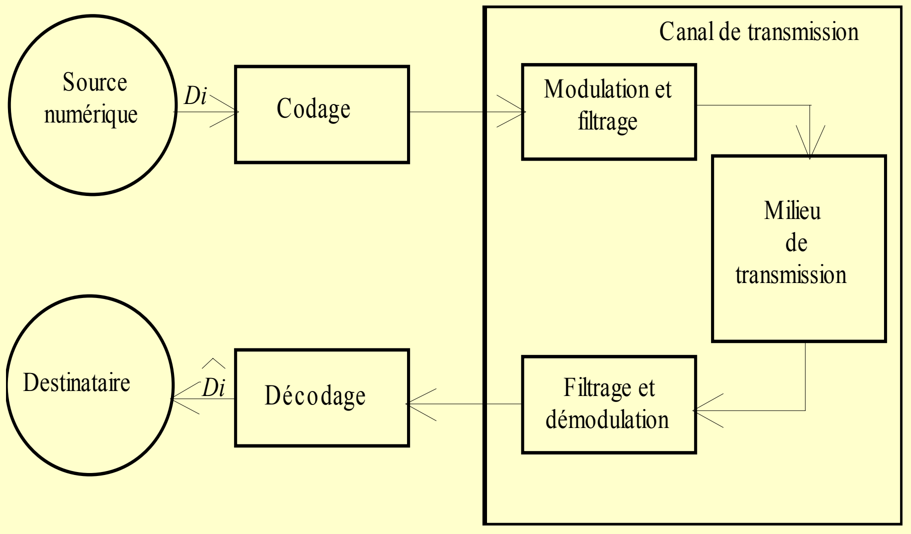
# Plan

1. Les codes correcteurs d'erreurs
   1. Généralités
   2. Limites Théoriques
2. Les codes convolutifs
   1. Les codes classiques
   2. Les codes RSC
   3. Poinçonnage et terminaison
3. Les algorithmes SISO
   1. SOVA
   2. MAP
   3. Max-Log-MAP
4. Turbo codes
   - La philosophie
   - Les différents schémas
   - Turbo décodage
5. Permutation
   - Régulière
   - Aléatoire
   - Pseudo-aléatoire
6. Turbo codes duo-binaires
   - Principe et avantages
   - Performance
7. Conclusions et perspectives

# Chapitre 1

Les codes correcteurs d'erreurs

## Une chaîne de transmission



Codes correcteur d'erreurs — Généralités

## Différentes familles de codage

Codage de source : réduire le débit d'information
Codage de canal : ajouter de la redondance
Sécurité, cryptographie
Authentification, *watermarking*
CDMA
...

Codes correcteur d'erreurs — Généralités

Séquence d'information à transmettre : $[d_1\ d_2\ \ldots\ d_i\ \ldots\ d_k]$

Addition d'une redondance linéaire : $[r_1\ r_2\ \ldots\ r_j\ \ldots\ r_{n-k}]$
(le code est dit systématique)

$$r_j = \sum_{i=1\ldots k} p_{i,j} d_i \quad \mod 2 \qquad (p_{i,j} = 0 \text{ ou } 1)$$

Forme matricielle

$$[r_1\ r_2\ \ldots\ r_j\ \ldots\ r_{n-k}] = [d_1\ d_2\ \ldots\ d_i\ \ldots\ d_k] \begin{pmatrix} p_{1,1} & \cdots & p_{1,n-k} \\ \vdots & & \vdots \\ p_{k,1} & \cdots & p_{k,n-k} \end{pmatrix}$$

Codes correcteur d'erreurs Généralités

$[d_1\ d_2\ \ldots\ d_i\ \ldots\ d_k][r_1\ r_2\ \ldots\ r_j\ \ldots\ r_{n-k}]$ : mot de code

Rendement de codage : $R = k/n$

Code $(n\ ,\ k\ ,\ d_{\min})$
de distance minimale $d_{\min}$

Le code étant linéaire, la séquence « *tout zéro* »
peut être prise comme référence

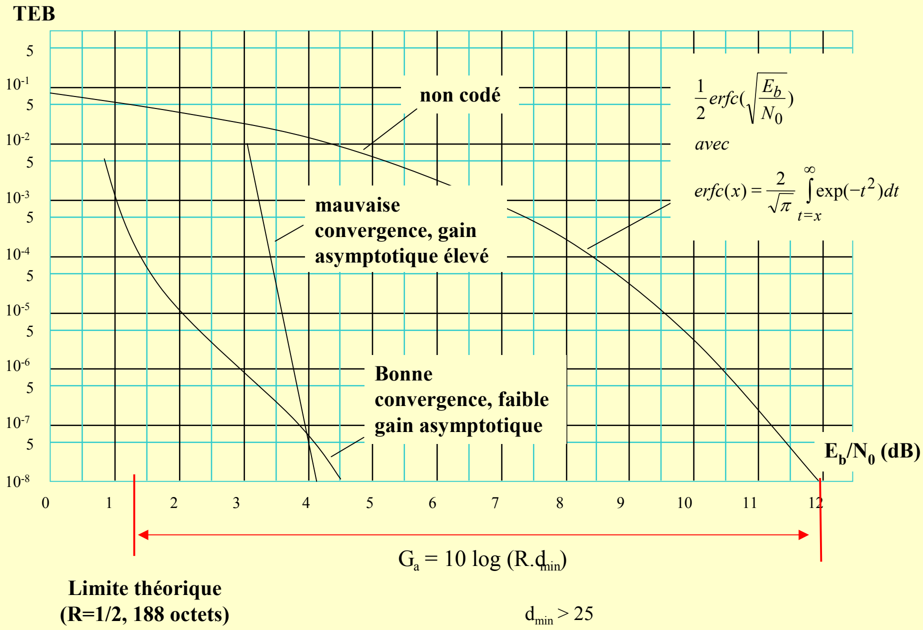
La distance $d_{min}$ est le nombre minimum de « 1 »
des mots de code $[d_1\ d_2\ \ldots\ d_i\ \ldots\ d_k]\ [r_1\ r_2\ \ldots\ r_j\ \ldots\ r_{n-k}]$
si $[d_1\ d_2\ \ldots\ d_i\ \ldots\ d_k] \neq$ « *tout zéro* »

Codes correcteur d'erreurs Généralités

Un bon code est un code avec une grande distance $d_{min}$
mais ce n'est pas tout :
- il doit pouvoir être décodé !
  (contre-exemple : les codes aléatoires)
- il doit exister un décodeur permettant d'utiliser
  la capacité de correction du code

Exemple :

MDP4, canal gaussien, taux d'erreurs cible : $10^{-8}$




Codes correcteur d'erreurs — Généralités

distinction traditionnelle mais inappropriée entre

Les codes en « bloc » : Hamming, Golay, BCH, Reed-Solomon, ...

Les codes « convolutifs »

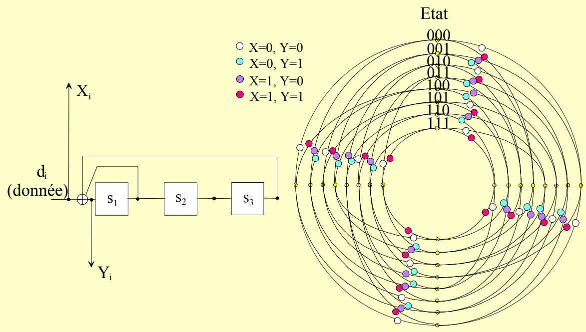
Une distinction plus naturelle prendrait en compte l'algorithme de décodage, en particulier :

- hard – dur (algébrique)
- soft – souple (probabiliste)

Exemple :

## Codes correcteur d'erreurs — Généralités

Comment transformer un code de Hamming parfait (8,4,4)




| $d_i$ | Hamming étendu | $\mathbf{Y_i}$ | $d_i$ | Hamming étendu | $\mathbf{Y_i}$ |
|---|---|---|---|---|---|
| 0000 | 0000 | **0000** | 1111 | 1111 | **1111** |
| 0001 | 0111 | **1011** | 1110 | 1000 | **0100** |
| 0010 | 1101 | **0111** | 1101 | 0010 | **1000** |
| 0011 | 1010 | **1100** | 1100 | 0101 | **0011** |
| 0100 | 1110 | **1110** | 1011 | 0001 | **0001** |
| 0101 | 1001 | **0101** | 1010 | 0110 | **1010** |
| 0110 | 0011 | **1001** | 1001 | 1100 | **0110** |
| 0111 | 0100 | **0010** | 1000 | 1011 | **1101** |

(autre exemple : le code de Golay étendu (24, 12, 8) peut être représenté par un treillis circulaire 16 états)

## Codes correcteur d'erreurs — Généralités

Code aléatoire (Shannon)

| Information | Redondance |
|---|---|
| 1 .... i ... k | 1 .... j ... n- k |
| 10000000 ... 00000000000000 | 0010010111...0110000100 |
| 01000000 ... 00000000000000 | 1001110100...0011101001 |
| 00100000 ... 00000000000000 | 1111010001...0001111011 |
| ............ | |
| 00000000 ... 00000000000001 | 1010101110...1010111101 |
| $\Sigma$ | |
| 10010110 ... 10001100010101 | 0011101011...1110010100 |

$d_{min} \approx (n-k)/4 = k.(1-R)/4R$ où $R = k/n$

## Codes correcteur d'erreurs — Généralités

$d_{min} \approx (n\text{-}k)/4 = k.(1\text{-}R)/4R$ où $R = k/n$

Exemple : $k$ = 1504 (MPEG), $R$ = 1/2

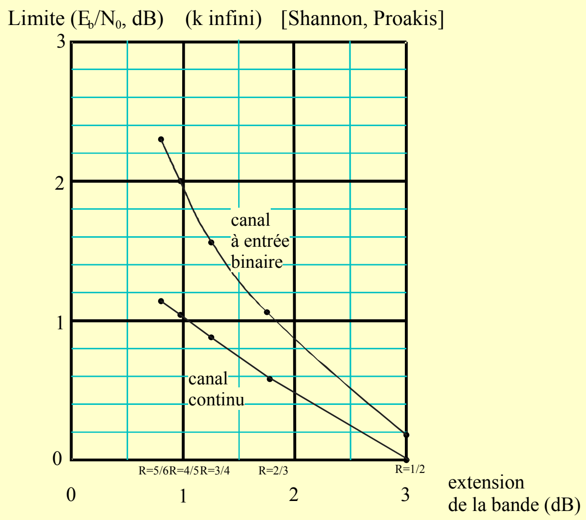
$\Longrightarrow$ $d_{min} \approx 375$ !!

(un code convolutif 64 états
a une distance $d_{min}$ = 10 pour $R$ = 1/2 !!)

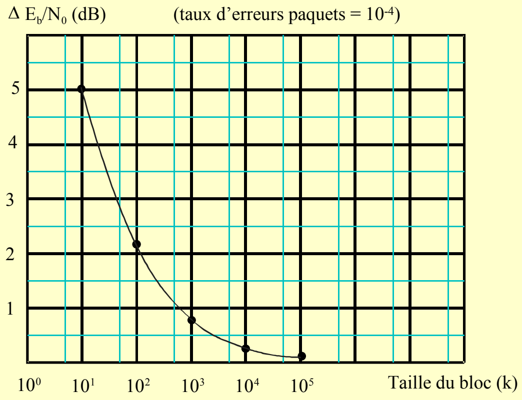
Mais non décodable en pratique ...

## Codes correcteur d'erreurs — Généralités

(sur la base d'un code aléatoire)



Codes correcteur d'erreurs — Généralités




Correction à prendre en compte pour des bocks de taille *k* :
[S. Dolinar, D. Divsalar and F. Pollara, "Code performance as a function of block size", TMO progress report 42-133, JPL, NASA].

# Calcul des limites

Un outil de calcul des limites théoriques a été développé par un étudiant en thèse il est disponible à l'adresse :

http://www-elec.enst-bretagne.fr/turbo/

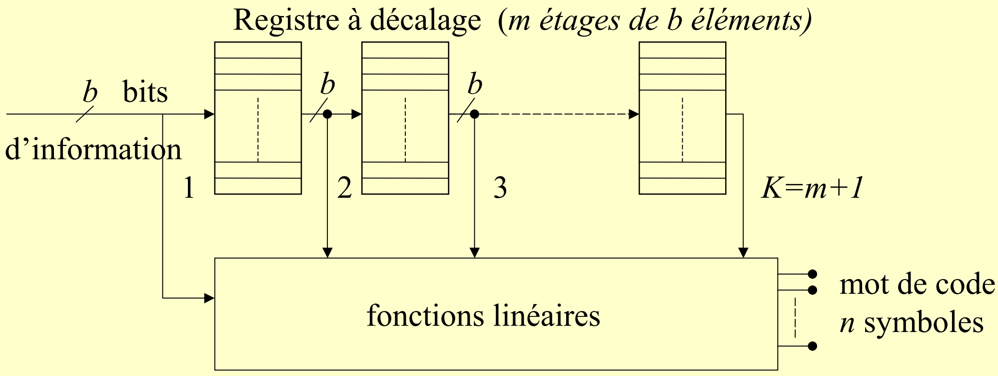
# Chapitre 2

Codes convolutifs

## Plan

- Codes convolutifs classiques
  - Représentations
  - Propriétés
- Codes convolutifs systématiques
- Codes convolutifs récursifs systématiques
- Rendement de codage et poinçonnage
- Fermeture de treillis

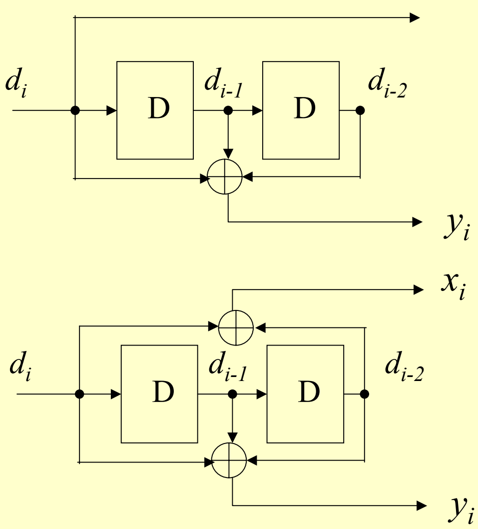
# Codes convolutifs




*K= longueur de contrainte*

*R=b/n (rendement de codage)*

# Code convolutif classique




Elias, 1954

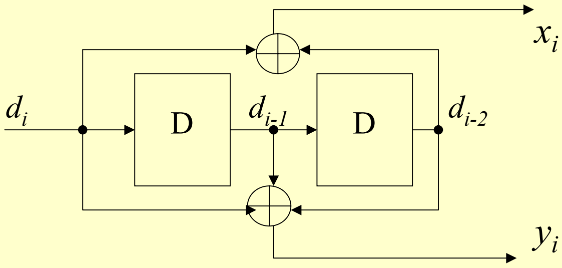
Mémoire du code : $\nu = 2$

Longueur de contrainte :
$K = \nu + 1 = 3$

$R = 1/2$

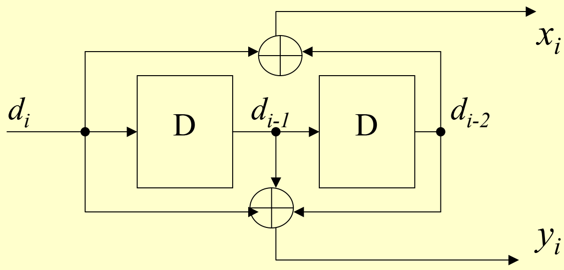
Forney, 1970

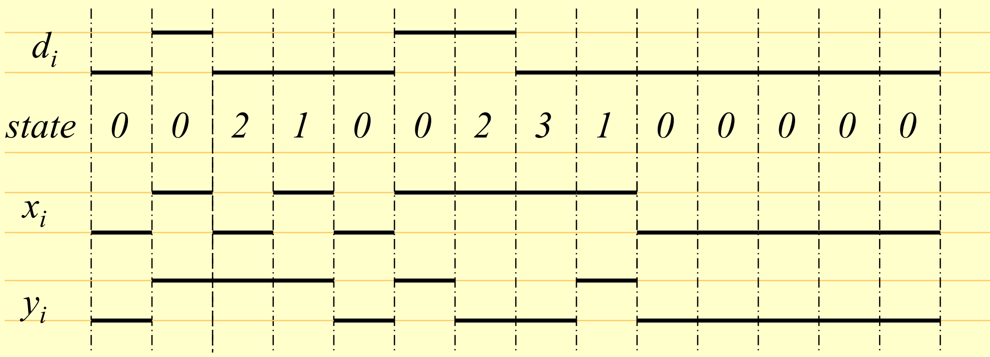
# Exemple



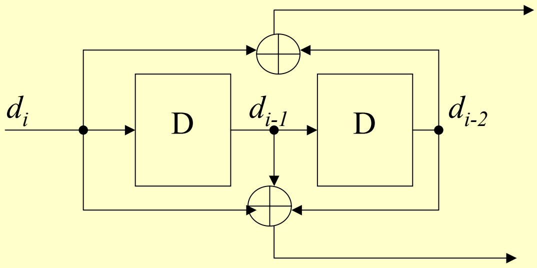

- $K$=3
- $b$=1
- $n$=2
- $R$=$b/n$=1/2

# Exemple



# Générateurs




$$x_i = \sum_{j=0}^{2} g_j^1 d_{i-j}$$

$$y_i = \sum_{j=0}^{2} g_j^2 d_{i-j}$$

$$g^1 = [g_0^1, g_1^1, g_2^1] = [1,0,1] \longrightarrow G^1(D) = g_0^1 + g_1^1 D + g_2^1 D^2$$

$$g^2 = [g_0^2, g_1^2, g_2^2] = [1,1,1] \longrightarrow G^2(D) = g_0^2 + g_1^2 D + g_2^2 D^2$$

Générateurs sous la forme octale : (5,7)

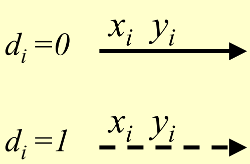
# Représentation des codes convolutifs

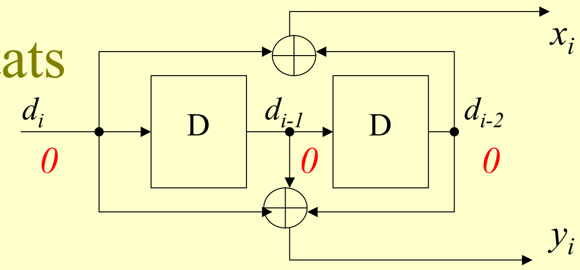
Trois formes :

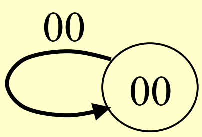
- arbre
- machine d'état
- treillis

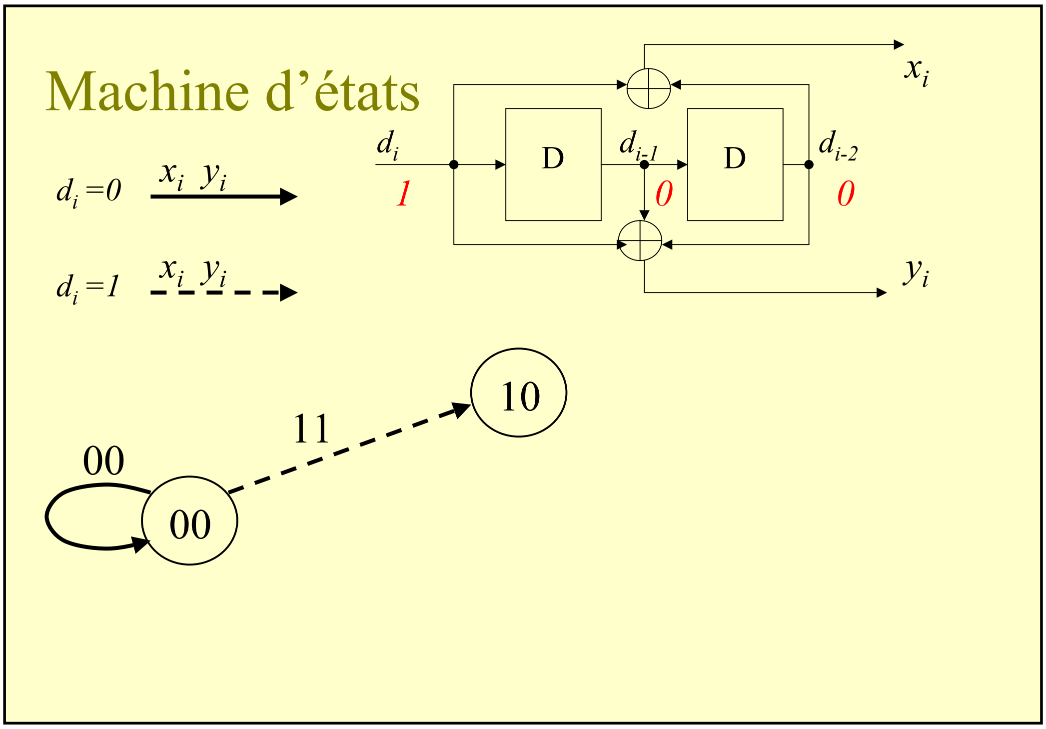
# Machine d'états

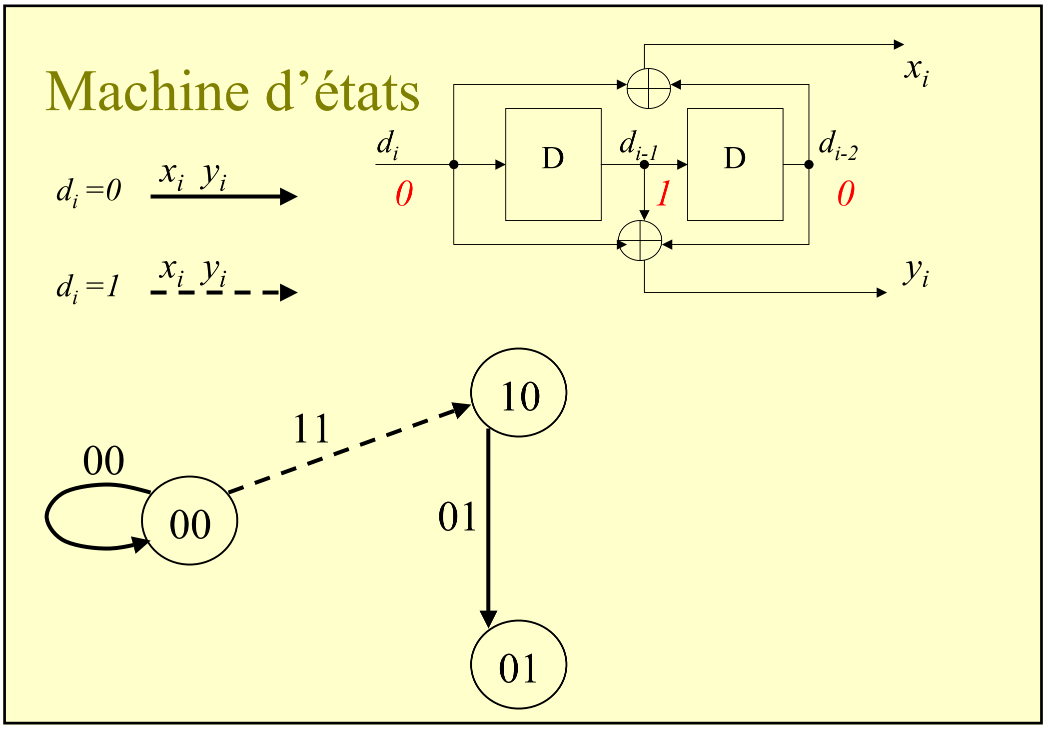
## Machine d'états

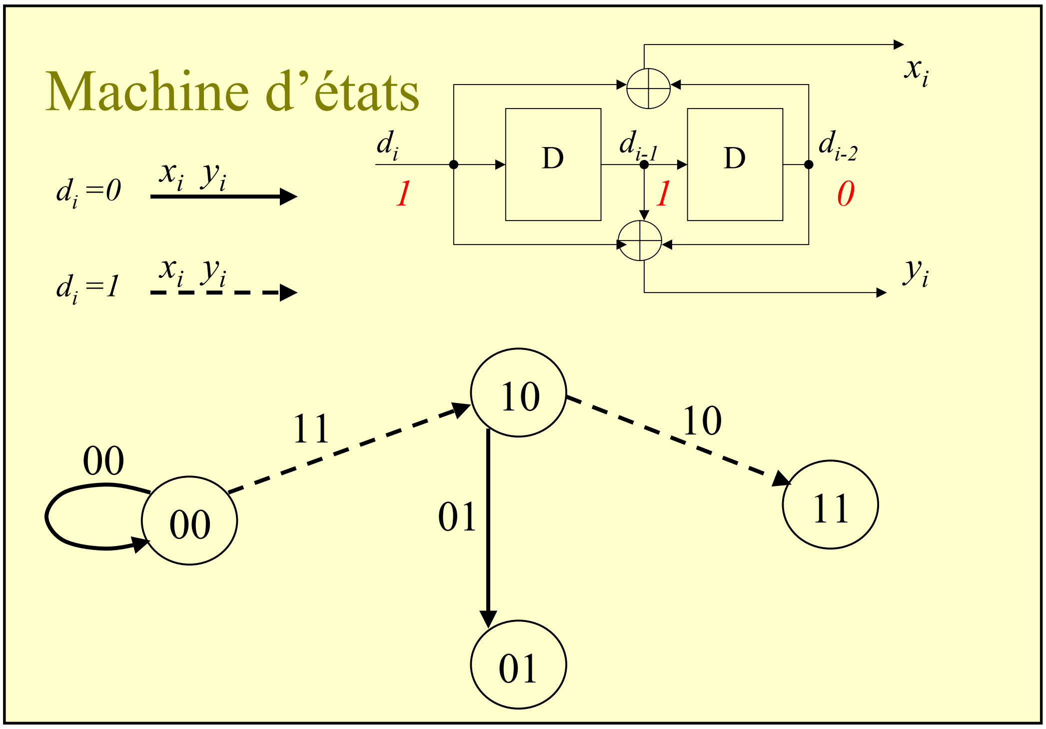


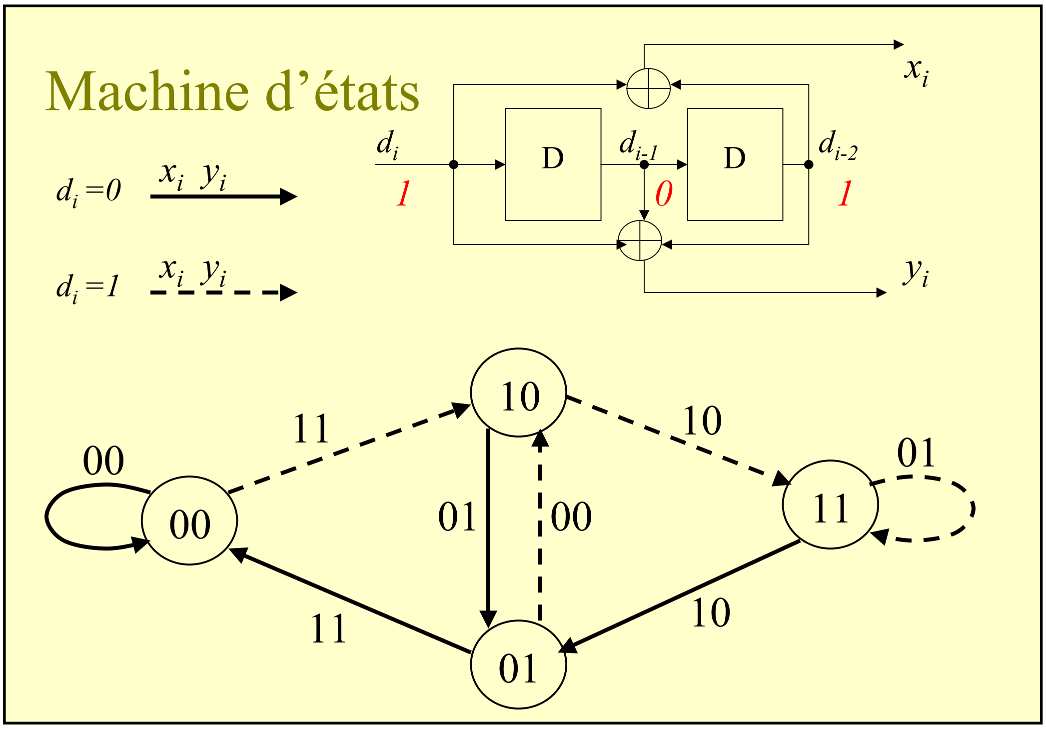




Machine d'états
$d_i = 0$
$x_i \; y_i$
$d_i = 1$
$x_i \; y_i$
$x_i$
$d_i$
D
$d_{i-1}$
D
$d_{i-2}$
1
0
0
$y_i$
10
11
00
00

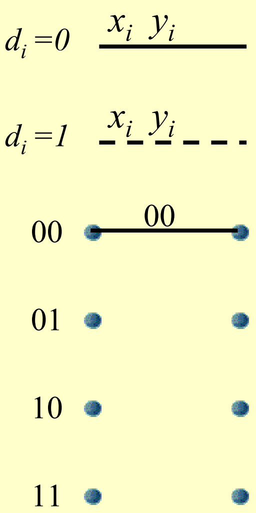

Machine d'états
$d_i = 0$
$x_i \; y_i$
$d_i = 1$
$x_i \; y_i$
$x_i$
$d_i$
D
$d_{i-1}$
D
$d_{i-2}$
0
1
0
$y_i$
10
11
00
00
01
01

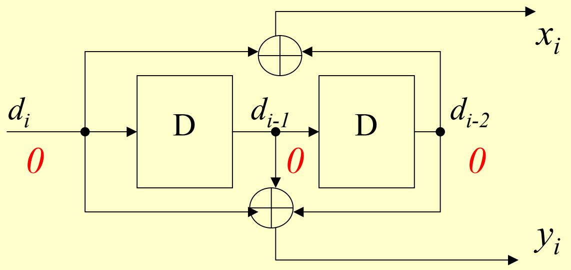

Machine d'états
$d_i = 0$ $x_i$ $y_i$
$d_i = 1$ $x_i$ $y_i$
$x_i$
$d_i$
D
$d_{i-1}$
D
$d_{i-2}$
1
1
0
$y_i$
10
11
10
00
00
01
11
01


Machine d'états
$d_i = 0$ $x_i$ $y_i$
$d_i = 1$ $x_i$ $y_i$
$x_i$
$d_i$
D
$d_{i-1}$
D
$d_{i-2}$
1
0
1
$y_i$
10
11
10
01
00
00
01
00
11
11
10
01

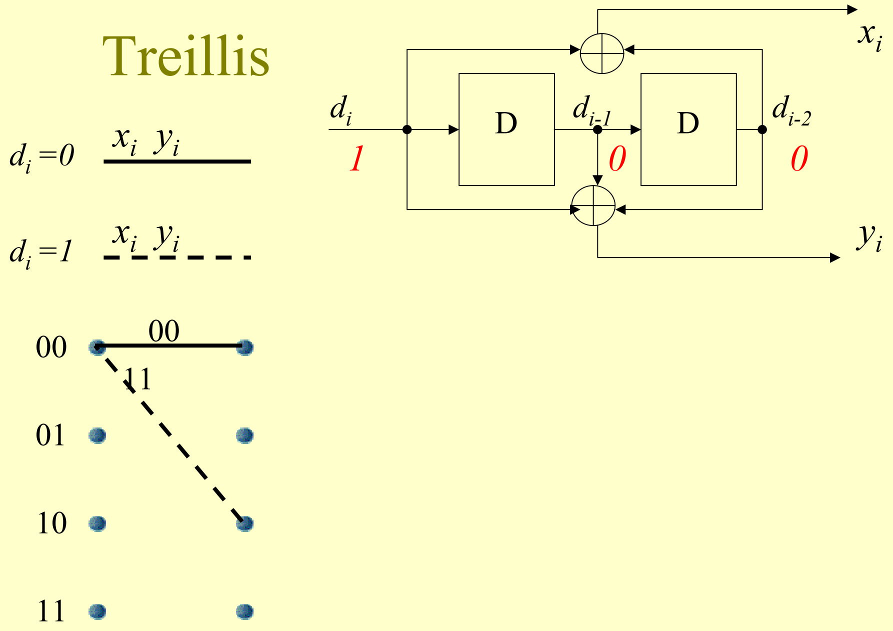
# Treillis

# Treillis

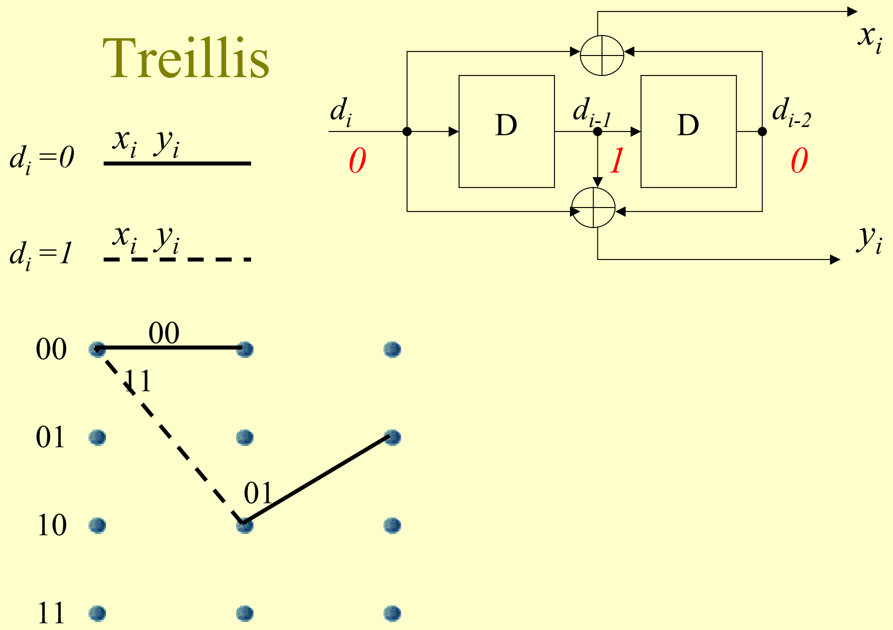


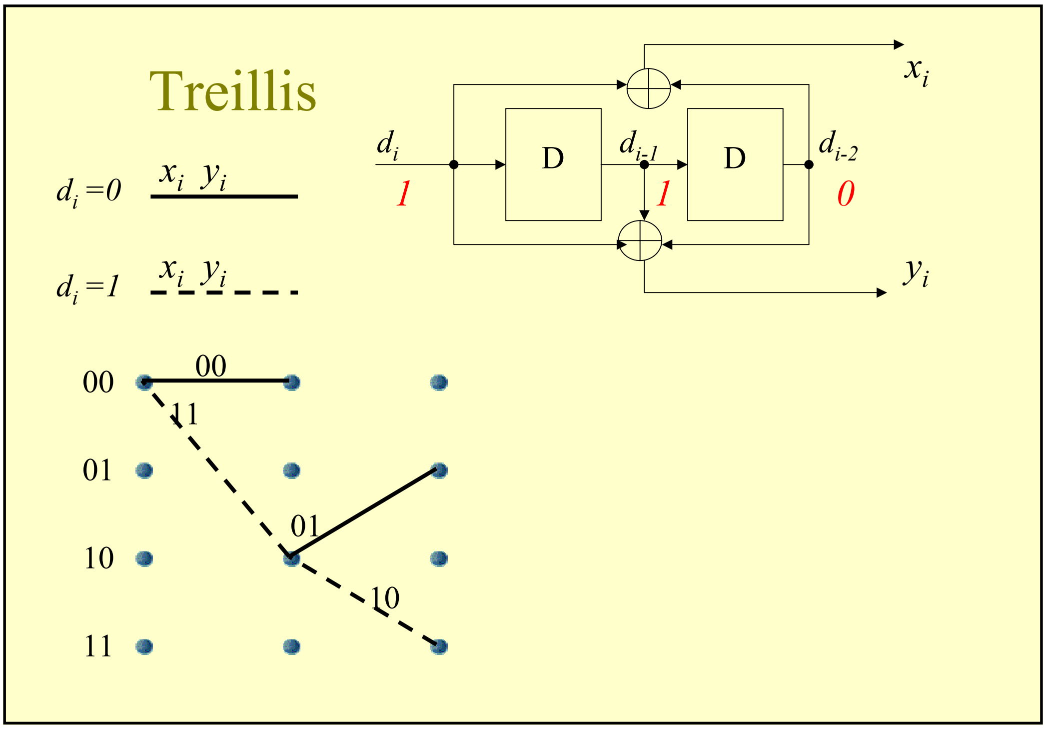



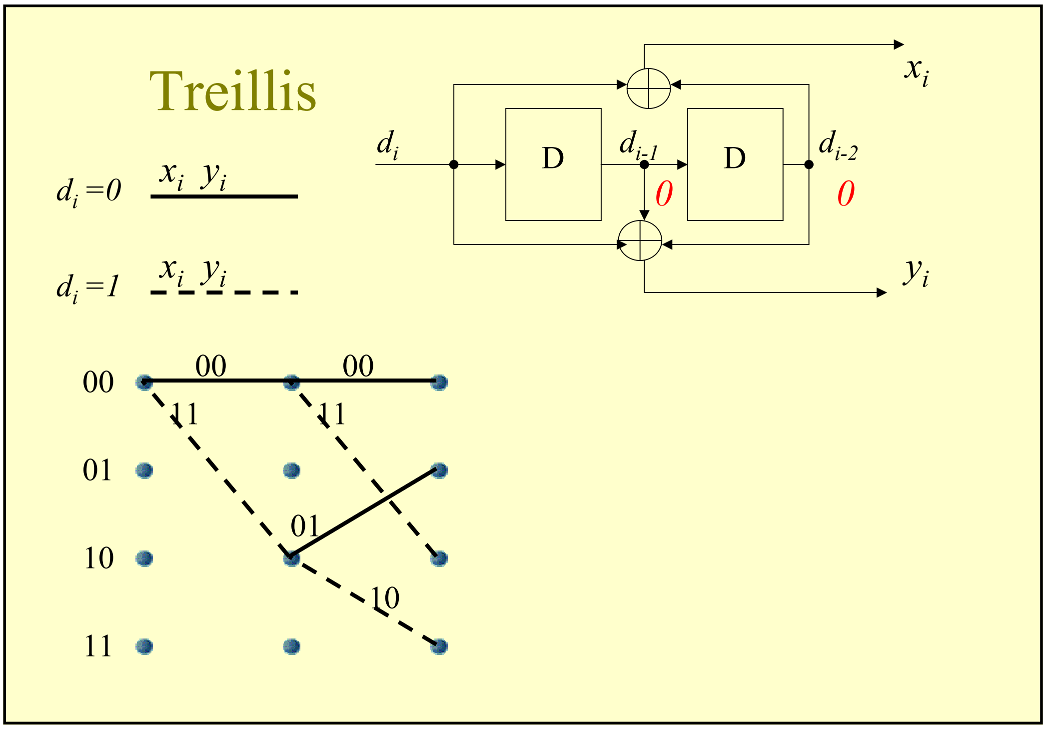
## Treillis



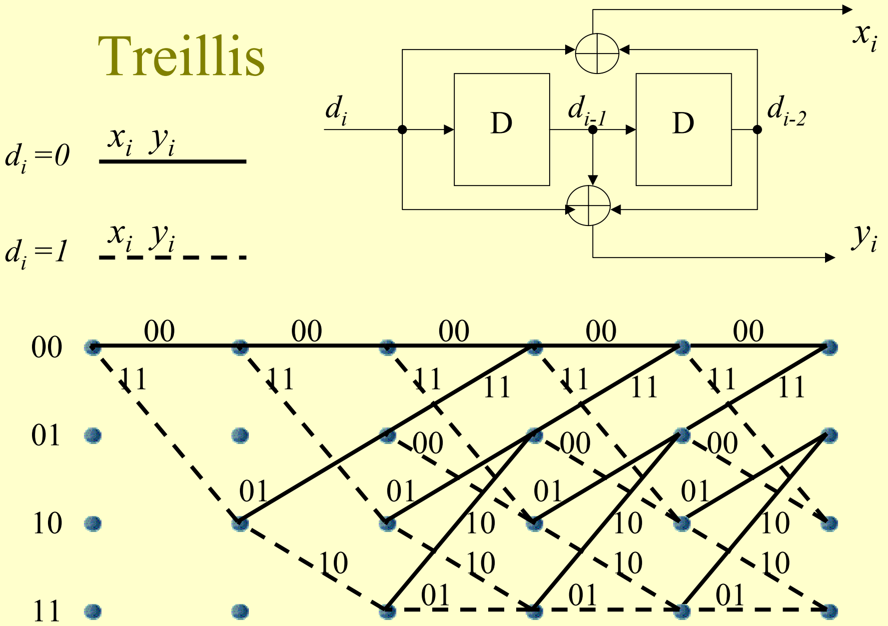

## Treillis



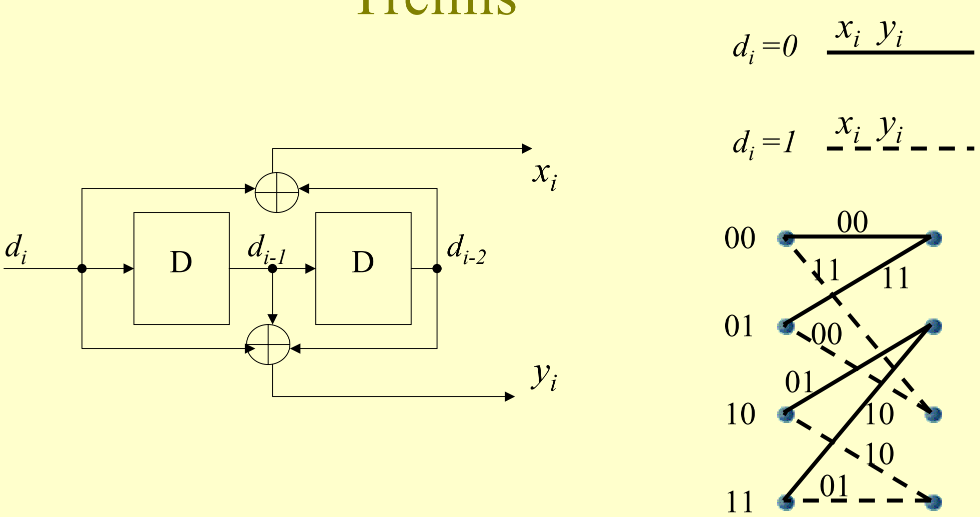

Treillis
$d_i$ =0 $x_i$ $y_i$
$d_i$ =1 $x_i$ $y_i$
$d_i$
D
$d_{i-1}$
D
$d_{i-2}$
$x_i$
$y_i$
1
1
0
00
01
10
11
00
11
01
10


Treillis
$d_i$ =0 $x_i$ $y_i$
$d_i$ =1 $x_i$ $y_i$
$d_i$
D
$d_{i-1}$
D
$d_{i-2}$
$x_i$
$y_i$
0
0
00
01
10
11
00
00
11
11
01
10

# Treillis

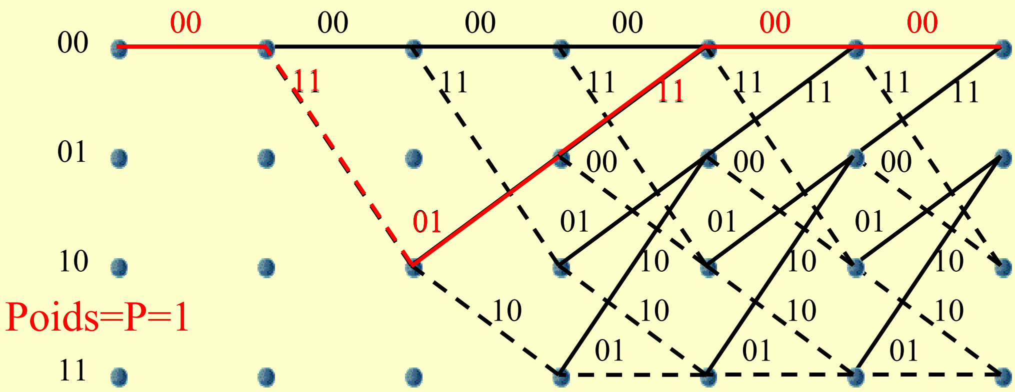



# Treillis



## Distance

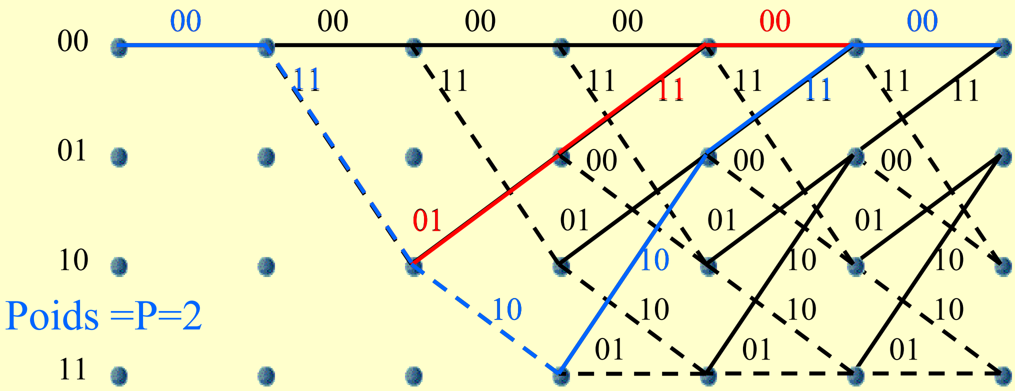
(*K*=3) (5,7) -> distance libre = $d_{libre}$=5




Distance de Hamming 0 + 2 + 1 + 2 + 0 + 0 =5

## Distance

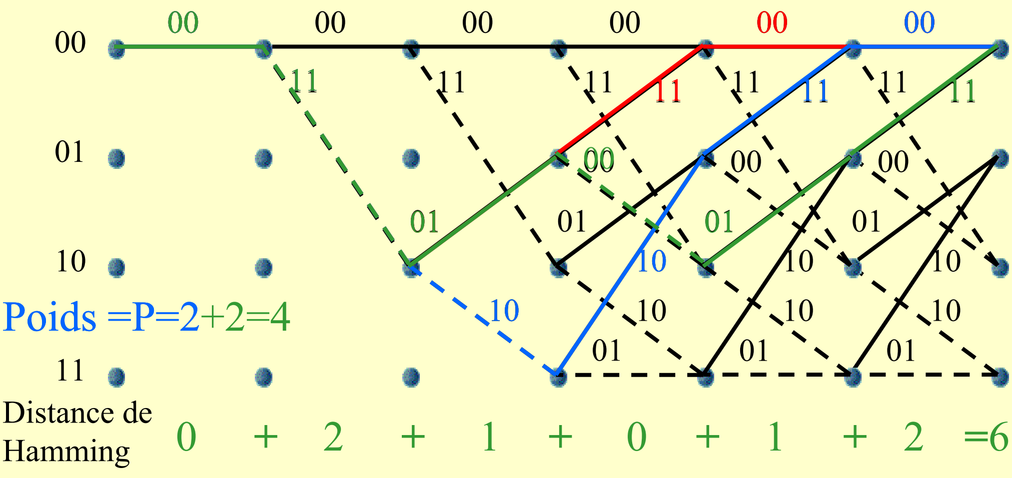
(*K*=3) (5,7) -> distance libre = $d_{libre}$=5




Distance de Hamming 0 + 2 + 1 + 1 + 2 + 0 =6

# Distance

($K$=3) (5,7) -> distance libre = $d_{libre}$=5




# Distance

- Longueur de contrainte : $K$ = 3
- Générateurs (5,7)
  - Distance libre minimale = 5
  - Spectre des distances:
    - Distance = 5 (1 cas), P=1
    - Distance = 6 (2 cas), P=4
    - Distance = 7 (4 cas), P=12
    - ….