

Cơ sở lí thuyết thông tin

Chương 1: Giới thiệu chung

TS. Phạm Hải Đăng

Phần 1: Giới thiệu chung

- Thông tin là gì?

Ví dụ: Tung đồng xu.

- Thông tin có phải là 1 đại lượng “vật lí”?

- Mục đích của môn học?

- Lí thuyết thông tin – lí thuyết mã sửa lỗi
- Nghiên cứu phương pháp xử lí thông tin như 1 đại lượng vật lí: tạo thông tin, truyền thông tin, lưu trữ thông tin, xử lí thông tin...
- Toàn vẹn thông tin.
- Thông tin trong không gian-thời gian.

Giới thiệu chung

- Lí thuyết về mã hóa sửa lỗi (Error Correction Coding).
 - Lí thuyết mã sửa lỗi liên quan tới việc xác định các nguồn gây lỗi lên thông tin số truyền tải trên kênh truyền, và phương pháp phát hiện/sửa lỗi ở phía bên nhận tin.

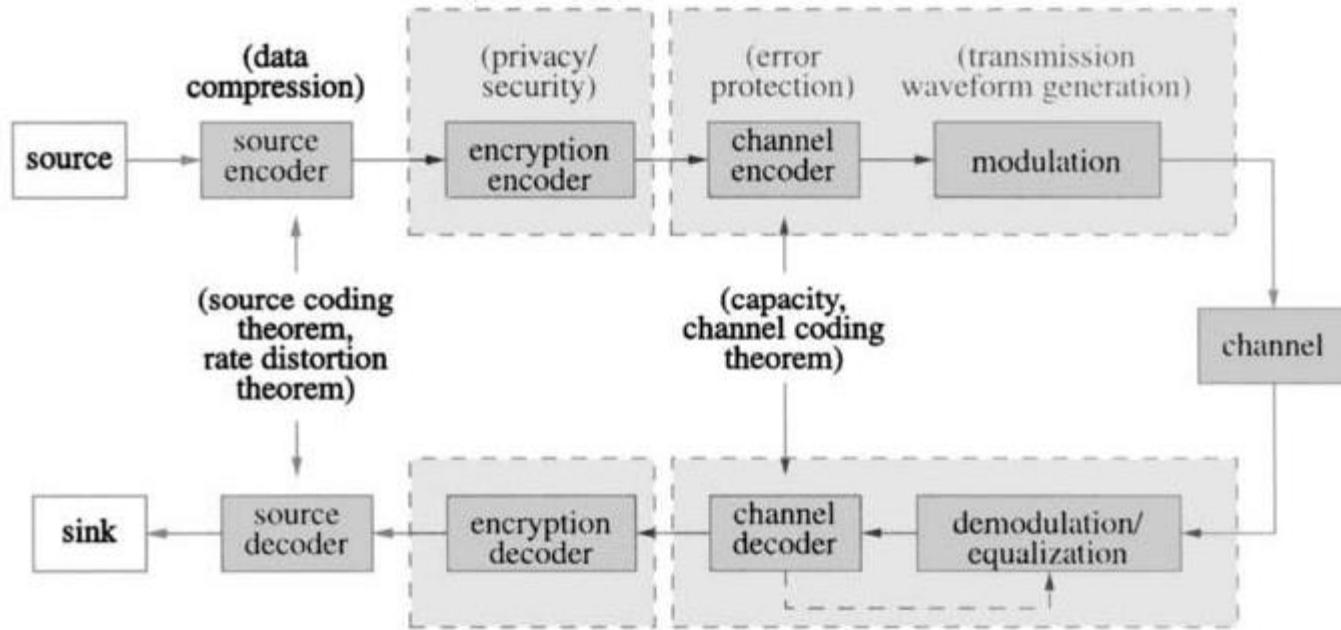
Ví dụ: Mã ISBN (international standard book number)

0 - 20 - 1 - 36186 - 8.
country publisher book no. check

Mã kiểm tra được thêm vào, thỏa mãn
điều kiện Tổng-cộng dồn của dãy số chia
Hết cho 3.

digit	cumulative sum	cumulative sum
0	0	0
2	2	2
0	2	4
1	3	7
3	6	13
6	12	25
1	13	38
8	21	59
6	27	86
8	35	121

Giới thiệu chung – hệ thống thông tin



- Hệ thống thông tin bao gồm 3 thành phần:
 - Nguồn phát tin
 - Kênh truyền tin
 - Bên nhận tin

Giới thiệu chung – hệ thống thông tin

- **Encrypter:** mã hóa, ẩn giấu thông tin ban đầu từ nguồn phát tin, nhằm tránh sự xâm phạm thông tin không mong muốn. (Toàn vẹn thông tin, bảo mật).
- **Channel Coder:** bộ mã hóa kênh. Bổ sung thông tin dư thừa nhằm cho phép việc phát hiện/sửa lỗi thông tin ở phía bên nhận tin.
- **Modulator:** bộ điều chế tín hiệu. Chuyển đổi dòng tín hiệu số (bit, digital symbol) thành dạng tín hiệu phù hợp với việc truyền dẫn trên kênh truyền.
- **Channel:** Kênh thông tin, là môi trường truyền dẫn thông tin từ nguồn tin tới bên nhận tin.

Phía nhận tin, quá trình xử lí thu nhận và xử lí thông tin tương ứng với bên phát tin: từ bộ giải điều chế (**demodulator**), tới bộ giải mã kênh (**channel decoder**) và giải mã hóa (**dencrypter**).

Ví dụ: Hệ thống thông tin BPSK

- Dòng bit thông tin được chuyển đổi thành dạng tín hiệu để truyền trên kênh truyền dẫn.
- BPSK – Binary Phase Shift Key

Ví dụ: dòng bit thông tin $b = \{b_0, b_1, b_2, \dots\}$

Tín hiệu truyền đi được ánh xạ thành các giá trị ± 1 theo công thức

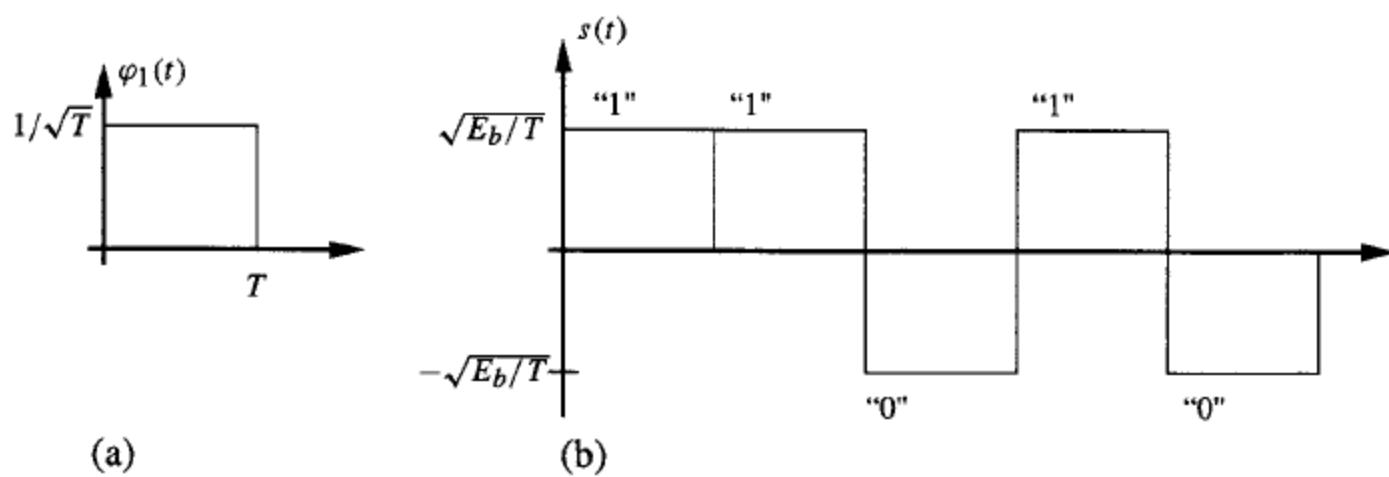
$$\tilde{b}_i = \sqrt{E_b} (2b_i - 1)$$

với E_b là năng lượng của truyền 1 bit tín hiệu

Xung đơn vị $\varphi_1(t)$ mang năng lượng đơn vị

$$\int_{-\infty}^{\infty} \varphi_1^2(t) dt = 1$$

Ví dụ: Hệ thống thông tin BPSK



- Tín hiệu truyền đi theo phương thức điều chế BPSK

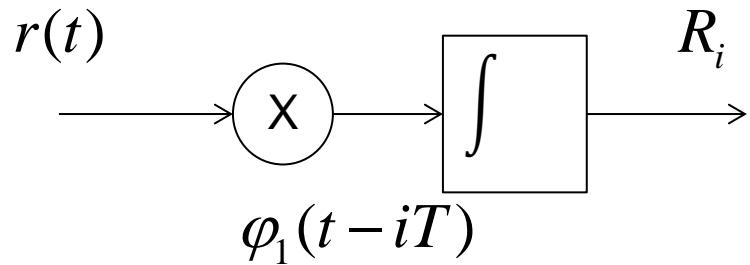
$$s(t) = \sum_i \tilde{b}_i \varphi_1(t - iT)$$

- Tín hiệu truyền trên kênh truyền dẫn bị ảnh hưởng bởi nhiễu

$$r(t) = s(t) + n(t)$$

Ví dụ: Hệ thống thông tin BPSK

Bên phía nhận tin:

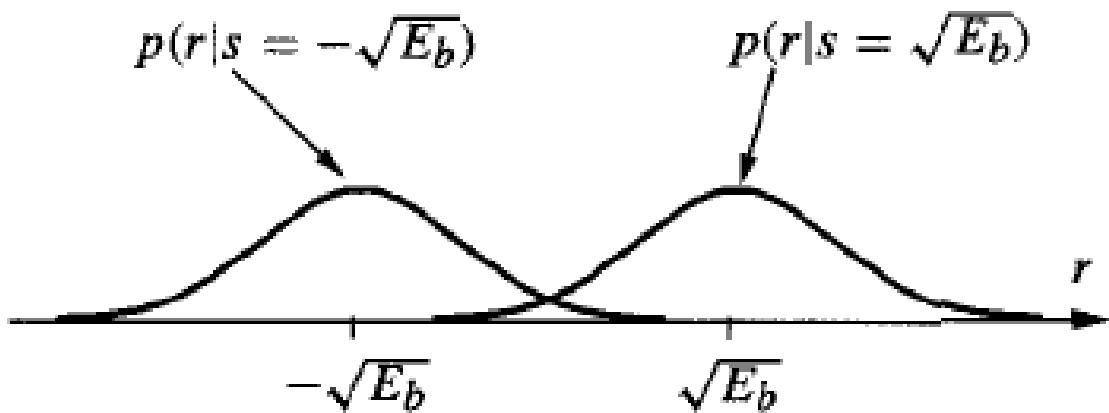


$r(t)$ là tín hiệu thu được từ kênh truyền

Tín hiệu giải điều chế BPSK

$$R_i = \int_{iT}^{(i+1)T} r(t) \varphi_1(t - iT) dt$$

Ví dụ: Hệ thống thông tin BPSK



Ví dụ: Hệ thống thông tin BPSK

Bộ giải điều chế tín hiệu đưa ra quyết định về giá trị bit thu được theo công thức:

$$\hat{s} = \arg \max_{s \in S} P(s | r)$$

Công thức Bayer

$$P(s | r) = \frac{P(r | s)P(s)}{P(r)}$$

- MAP (Maximum a posteriori) đưa ra quyết định dựa trên việc tìm cực đại của xác suất hậu nghiệm

$$\hat{s} = \arg \max_{s \in S} P(r | s)P(s)$$

- ML (Maximum likelihood): đưa ra quyết định dựa trên giả thuyết xác suất tín hiệu nguồn phát $P(s)$ là bằng nhau

$$\hat{s} = \arg \max_{s \in S} P(r | s)$$

Phần 2: Định lý Shannon

☐ Định lý Shannon

$$C = B \log_2 \left(1 + \frac{S}{N} \right)$$

Trong đó

S là công suất tín hiệu (W)

N mật độ nhiễu trung bình trong toàn kênh (W)

B băng thông (Hz)

C Dung lượng giới hạn của kênh truyền (b/s)

Định lý Shannon

Định lí Shannon cho biết dung lượng kênh truyền cho phép truyền luồng thông tin ban đầu (loại trừ thông tin dư thừa chèn thêm do mã hoá sửa lỗi) trong trường hợp kênh AWGN (Additional White Gaussian Noise) với băng thông B và tỷ lệ S/N cho trước.

Trường hợp 1

$R < C$

R tốc độ truyền tin trên đường truyền (b/s)

Cho phép truyền thông tin với tốc độ $R < C$ với xác suất lỗi BER=0

Trường hợp 2:

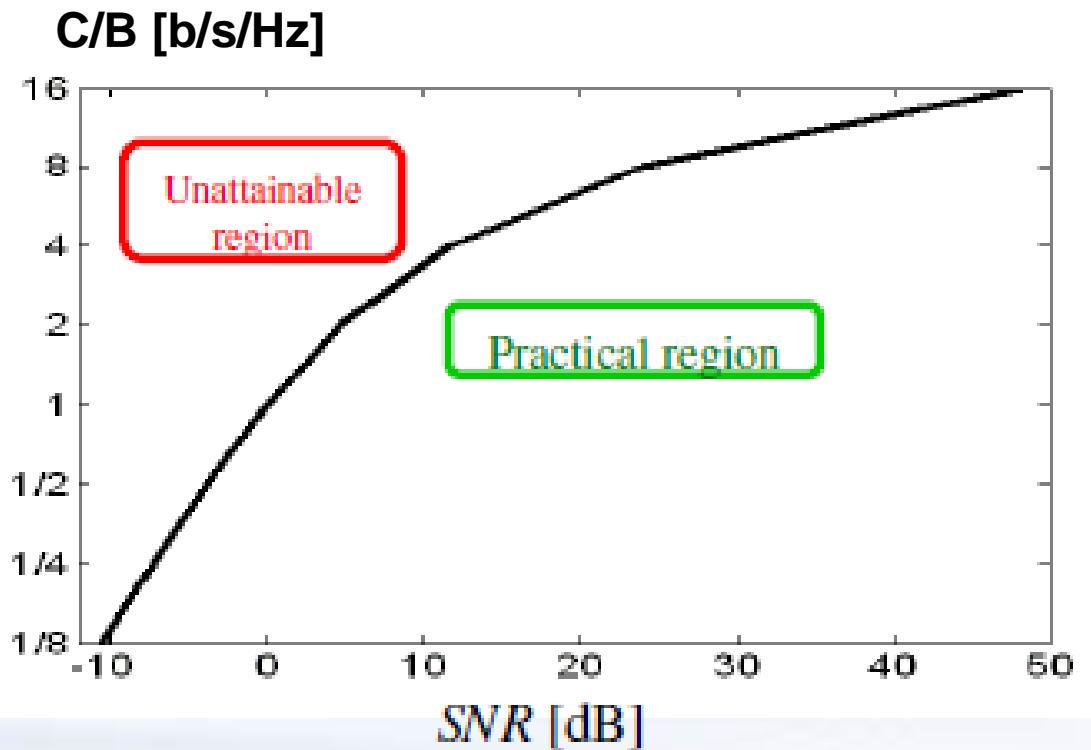
$R > C$

Tốc độ truyền tin R vượt quá dung lượng kênh truyền, không thể truyền thông tin hữu ích trên kênh truyền.

Ý nghĩa của định lí Shannon

- Định lí Shannon chỉ ra giới hạn lí thuyết về dung lượng của kênh thông tin.
- Shannon cũng chỉ ra về lí thuyết: có loại mã cho phép tăng tốc độ truyền tín hiệu trên kênh truyền tiệm cận với giới hạn Shannon.

Tuy nhiên, Shannon không chỉ ra đó là loại mã gì.



Các ví dụ giải thích định lí Shannon

Example 1: trong trường hợp SNR=20dB

Băng thông hệ thống $B=4\text{KHz}$

Tính dung lượng kênh truyền C

Các ví dụ giải thích định lí Shannon

Example 2:

Để truyền được luồng dữ liệu với tốc độ 50kbps, với băng thông là 1MHz, hãy tính điều kiện kênh truyền AWGN – tỷ số SNR cho phép?

Đáp án -14.5dB

Các ví dụ giải thích định lí Shannon

Example 3:

Trong trường hợp sử dụng phương pháp điều chế WCDMA, băng thông $B=5\text{MHz}$, tốc độ thoại $R=12.2\text{kb/s}$.

Tính tỷ lệ SNR cần thiết để truyền dữ liệu thoại trên kênh truyền.

Đáp án: $\text{SNR} = -27.7\text{dB}$

Các ví dụ giải thích định lí Shannon

Example 4:

Phương pháp MIMO

Multiple-Input-Multiple-Output

Nhiều antenna phát và nhiều antenna thu. Phổ biến trong các chuẩn WiFi thế hệ mới 802.11n/ac, 4G LTE

Mã hóa – Mã thống kê tối ưu

- Khái niệm mã hóa, các thông số của mã hóa
- Mã thống kê
 - Entropy
 - Mã Shannon-Fano
 - Mã Huffman

Mã thống kê – Khái niệm về Entropy

- Entropy trong lí thuyết thông tin là phép đo định lượng về “thông tin” của nguồn tin.
 - Nguồn tin có Entropy lớn \Leftrightarrow nội dung ngẫu nhiên
 - Nguồn tin có Entropy nhỏ \Leftrightarrow nội dung có có cấu trúc, lặp lại.
- Entropy được sử dụng trong việc mã hóa – nén thông tin. Nếu phân bố xác suất PDF của nguồn tin được biết trước, giá trị Entropy cho biết số bit trung bình cần thiết để mã hóa nguồn tin.

Mã thống kê – Tính giá trị Entropy

$$H(X) = - \sum_{x \in X} p(x) \cdot \log_b p(x)$$

- $H(X)$ – Entropy của nguồn tin
- X – Nguồn tin với các ký tự x
- $b=2$ - bit thông tin

Ví dụ:

symbol	Tần suất	$p(x)$	$-p(x) \cdot \log_2 p(x)$
a	5	0.45	0.52
b	2	0.18	0.45
r	2	0.18	0.45
c	1	0.09	0.31
d	1	0.09	0.31

11

2.04

$$H(X) = 2.04$$

Mã thống kê – Tính chất của Entropy

$$H(X) = -\sum p(x) \cdot \log_b p(x)$$

Ví dụ: Nguồn tin "*abracadabra*" $x \in X$

symbol	Tần suất	$p(x)$	$-p(x) \cdot \log_2 p(x)$
a	5	0.45	0.52
b	2	0.18	0.45
r	2	0.18	0.45
c	1	0.09	0.31
d	1	0.09	0.31

11

2.04

$$H(X)=2.04$$

Nguồn tin "*abracadabra*" có thể mã hóa với mã có độ dài trung bình 2.04bit/kí tự. Bản tin mã hóa theo cách này được gọi là **mã tối ưu** hay **mã hóa Entropy**.

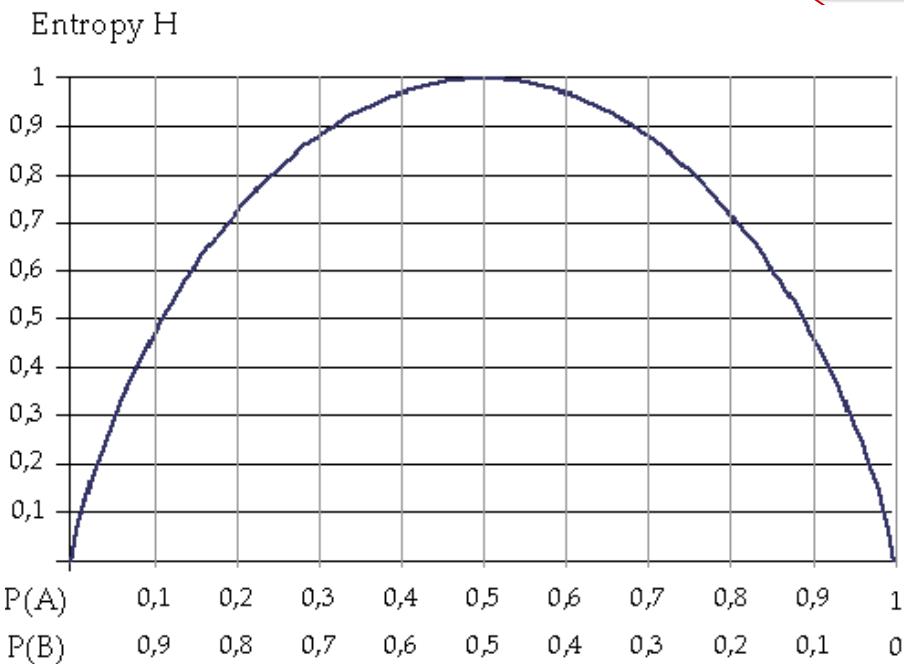
Mã thống kê – Entropy của nguồn tin nhị phân

Bản tin binary gồm 2 ký tự A,B

$$P(A) = 1 - P(B)$$

Nhận xét:

- Giá trị Entropy cực đại $H=1$ khi A và B có xác suất như nhau (0.5). Khi đó độ dài mã trung bình là 1 bit – tối ưu.
- Trong các trường hợp còn lại, $H < 1$, cần lựa chọn mã khác để đạt hiệu quả tốt hơn (code efficiency)



Mã thống kê – Định nghĩa và phân loại

- ❑ Entropy cung cấp thông tin về độ dài từ mã cần thiết cho việc mã hóa nguồn tin.
- ❑ Điều kiện tiên quyết của mã thống kê là cần biết trước xác suất xuất hiện của các kí tự (symbol) trong nguồn tin.
- ❑ Bộ mã hóa thống kê sẽ gán các từ mã (code word) có độ dài ngắn vào các kí tự có xác suất lớn, và ngược lại, gán từ mã có độ dài lớn cho các kí tự có xác suất nhỏ => Giảm kích thước của nguồn tin.
- ❑ Các thuật toán của mã hóa thống kê
 - Mã Shannon-Fano
 - Mã Huffman

Mã Shannon-Fano

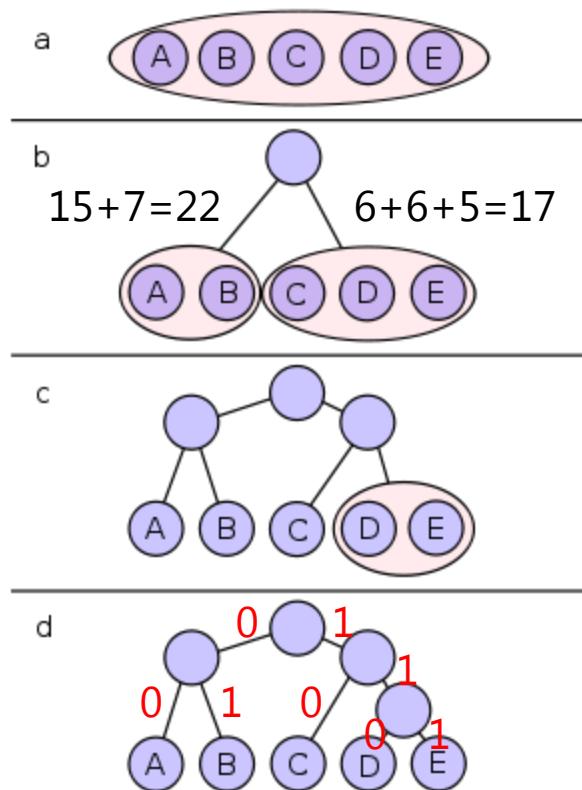
- Do Shannon và Fano độc lập xây dựng dựa trên lí thuyết Entropy.
- Mã Shannon-Fano được xây dựng nhằm tối ưu hóa độ dài của từng từ mã (code word) tiệm cận với giá trị $-\log_2 p(x)$.

Ví dụ:

symbol	Tần suất	$p(x)$	Lượng tin riêng $-\log_2 p(x)$
A	15	0.38	1.38
B	7	0.18	2.48
C	6	0.15	2.70
D	6	0.15	2.70
E	5	0.13	2.96

$$H(X)=2.1858$$

symbol	Code word
A	00
B	01
C	10
D	110
E	111



Mã Huffman

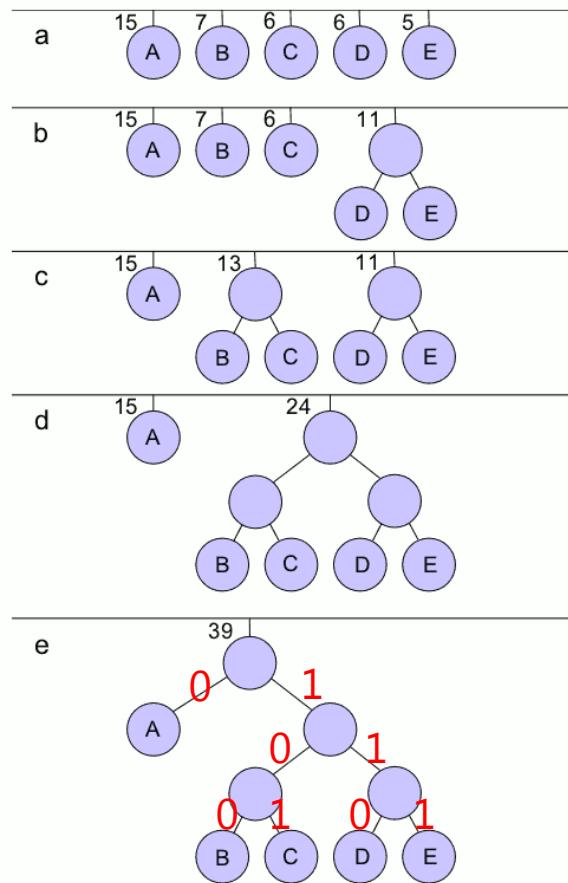
- Mã Huffman được xây dựng dựa trên lí thuyết Entropy
- Mã Huffman xây dựng cây nhị phân và gán giá trị bit từ dưới lên (bottom-up) nhằm tối ưu hóa kích thước của toàn bộ bản tin.

Ví dụ:

symbol	Tần suất	p(x)	Lượng tin riêng $-\log_2 p(x)$
A	15	0.38	1.38
B	7	0.18	2.48
C	6	0.15	2.70
D	6	0.15	2.70
E	5	0.13	2.96

$$H(X)=2.1858$$

symbol	Code word
A	0
B	100
C	101
D	110
E	111



So sánh giữa mã Shannon-Fano và Huffman

- ❑ Mã Shannon-Fano: các từ mã có kích thước gần với lượng tin riêng của kí tự (sai số ± 1)
- ❑ Mã Huffman đảm bảo kích thước của bản tin mã hóa nhỏ nhất

symbol	Shannon-Fano Code word	Huffman Code word	Tần suất	Lượng tin riêng $-\log_2 p(x)$
A	00	0	15	1.38
B	01	100	7	2.48
C	10	101	6	2.7
D	110	110	6	2.7
E	111	111	5	2.96

■ Kích thước bản tin

$$L_{Shannon} = 2bit \times (15 + 7 + 6) + 3bit \times (6 + 5) = 89bit$$

$$R_{Shannon} = 89bit / 39 = 2.28bit / symbol$$

$$L_{Huffman} = 1bit \times 15 + 3bit \times (7 + 6 + 6 + 5) = 87bit$$

$$R_{Huffman} = 87bit / 39 = 2.23bit / symbol$$

$$H(X) = 2.1858$$

Vi du

Cơ sở lí thuyết thông tin

Chương 3: Mã hóa kênh Mã Khối tuyến tính

TS. Phạm Hải Đăng

Phần 1: Khái niệm cơ bản

❑ Mã kênh/Mã sửa lỗi

- Mã hóa kênh (channel Coding) hay còn gọi là mã sửa lỗi (Error Correction coding) là kỹ thuật khống chế, phát hiện và sửa lỗi trong quá trình truyền dữ liệu qua kênh có nhiễu.
- Mã sửa lỗi sử dụng thông tin dư thừa (redundancy) được mã hóa thêm vào dữ liệu phía bên phát. Thông tin dư thừa sẽ được phía thu sử dụng để sửa lỗi - mà không cần yêu cầu phát lại tin.

Phần 1: Khái niệm cơ bản

□ Phân loại lỗi

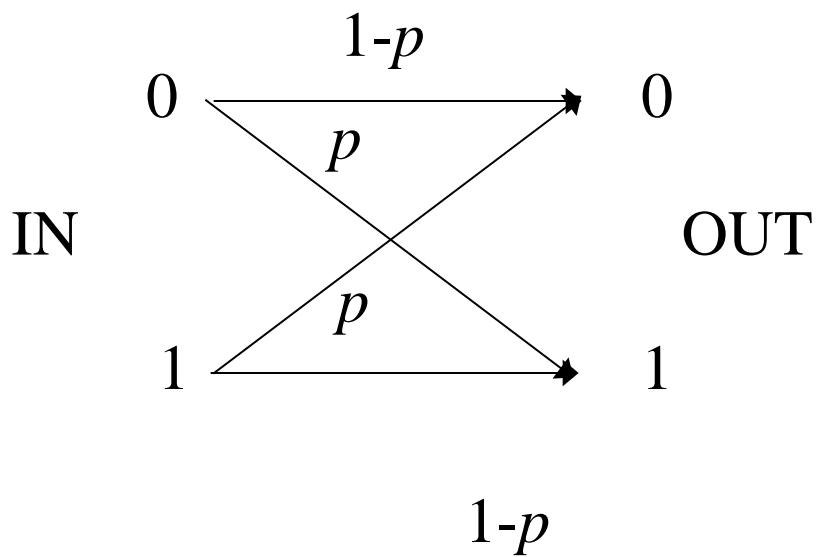
- Lỗi độc lập thống kê: Lỗi xuất hiện trong quá trình truyền tin trên kênh truyền, xuất hiện độc lập không liên quan tới nhau. Ví dụ: nhiễu Gaussian.
- Lỗi chùm: Lỗi có phân bố liên hệ với nhau.

Phần 1: Khái niệm cơ bản

Ví dụ: Kênh truyền tin không nhớ

(Binary Symmetric Memoryless Channel).

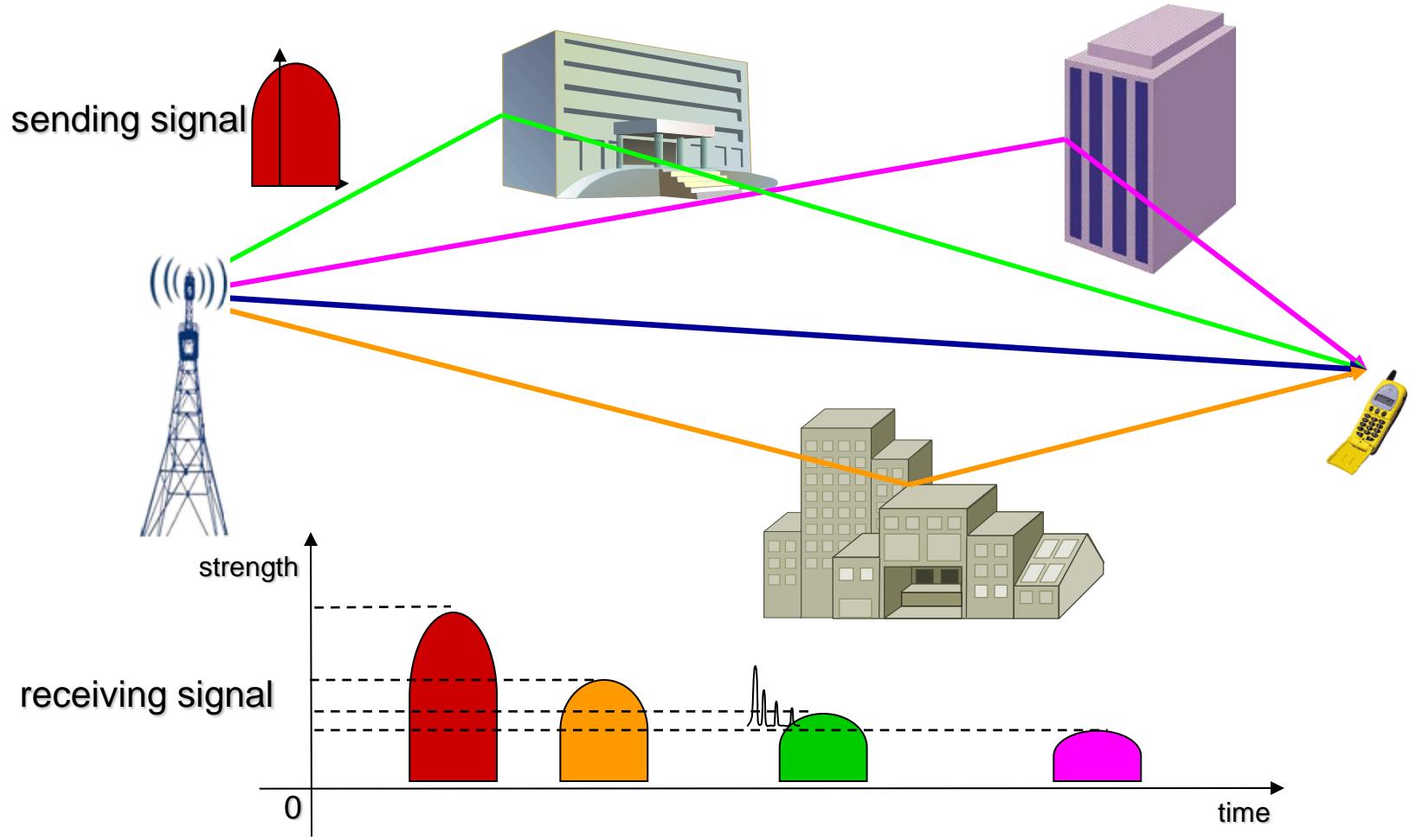
- Lỗi xảy ra với bit “0” và “1” với cùng xác suất p (symmetric)
- Lỗi xảy ra ngẫu nhiên và độc lập giữa các bit (memoryless)



p là xác suất lỗi – BER
Bit Error Rate (BER)

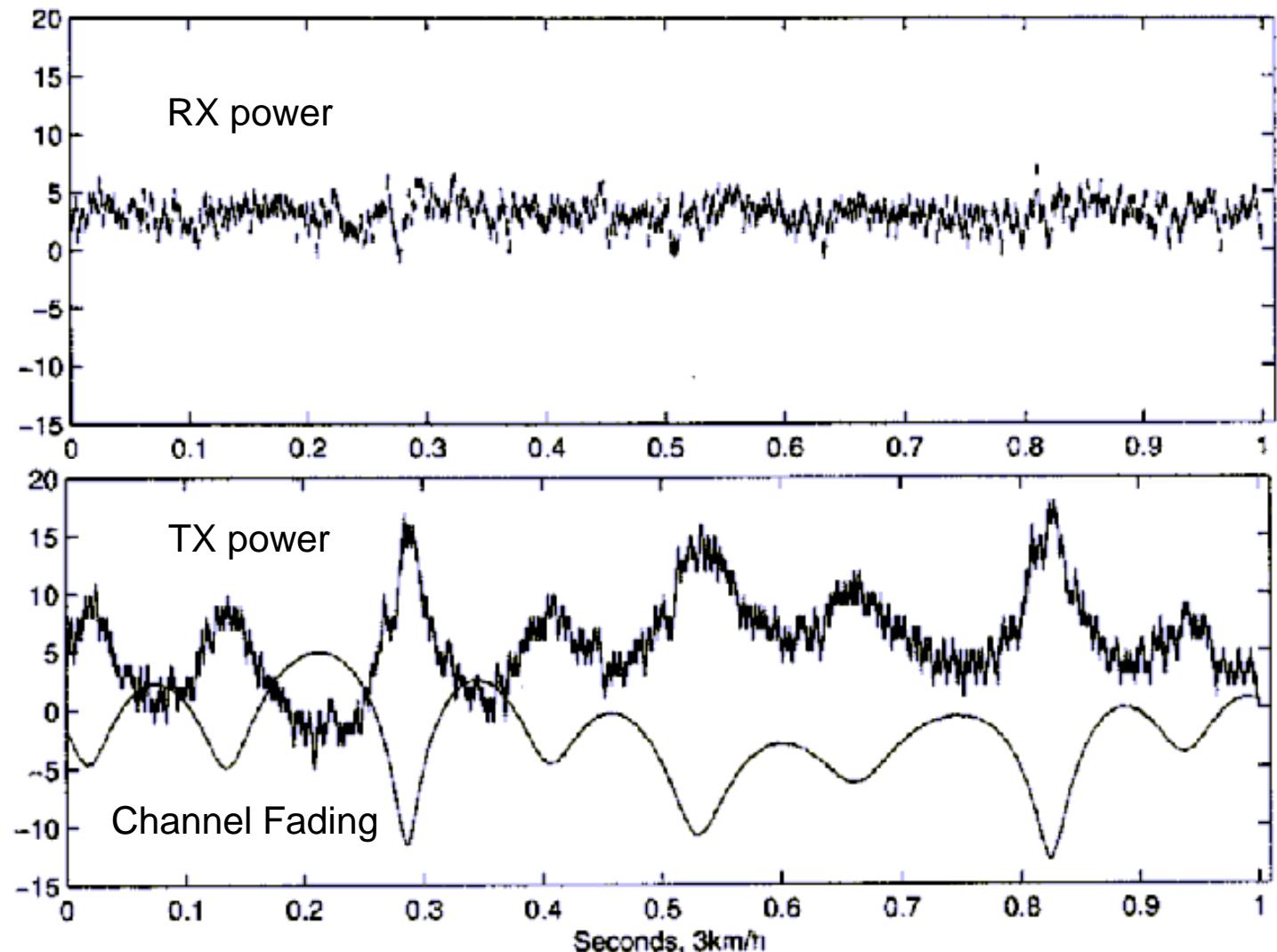
Phần 1: Khái niệm cơ bản

Ví dụ: Kênh truyền tin đa đường



Phần 1: Khái niệm cơ bản

Ví dụ: Kênh truyền tin đa đường



Phần 1: Khái niệm cơ bản

Phân loại mã sửa lỗi:

- ❑ Mã khối (block codes): thông tin được mã hóa và chèn thêm phần dư thừa theo từng khối.
 - Mã khối
 - Mã khối tuyến tính
 - Mã vòng CRC
 - Mã BCH, Reed-Solomon, LDPC
- ❑ Mã chập (Convolutional codes): thông tin được biến đổi theo các hàm truyền đạt (phép tích chập). Không có giới hạn rõ ràng giữa thông tin và phần dư thừa.
 - Mã chập (convolutional codes)
 - Mã Turbo

Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

- Tốc độ mã
- Khoảng cách Hamming (Hamming distance)
- Khoảng cách tối thiểu (minimum distance)
- Ma trận sinh, mã trận kiểm tra chẵn lẻ.

Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

Tốc độ mã

Giả thiết \mathbb{F}_2 là tập hợp 2 phần tử '0' và '1'.

\mathbb{F}_2^n biểu diễn vector n phần tử của \mathbb{F}_2

Số binary (n, k) là tập hợp 2^k điểm trong không gian \mathbb{F}_2^n

Mã (n, k) là mã chấp nhận k bit đầu vào và tạo ra n bit đầu ra.

Định nghĩa: tốc độ mã của mã (n, k) là

$$R = \frac{k}{n}$$

Ví dụ: Mã lặp (repetition code) $(n, 1)$ nhận 1 bit đầu vào và tạo ra n bit lặp lại ở đầu ra. Tốc độ mã là $R = \frac{1}{n}$

Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

Ma trận sinh

Với m biểu diễn thông tin (message).

C là từ mã (codeword) của mã lặp $(n,1)$

$$C = [m, m, m, \dots, m]$$

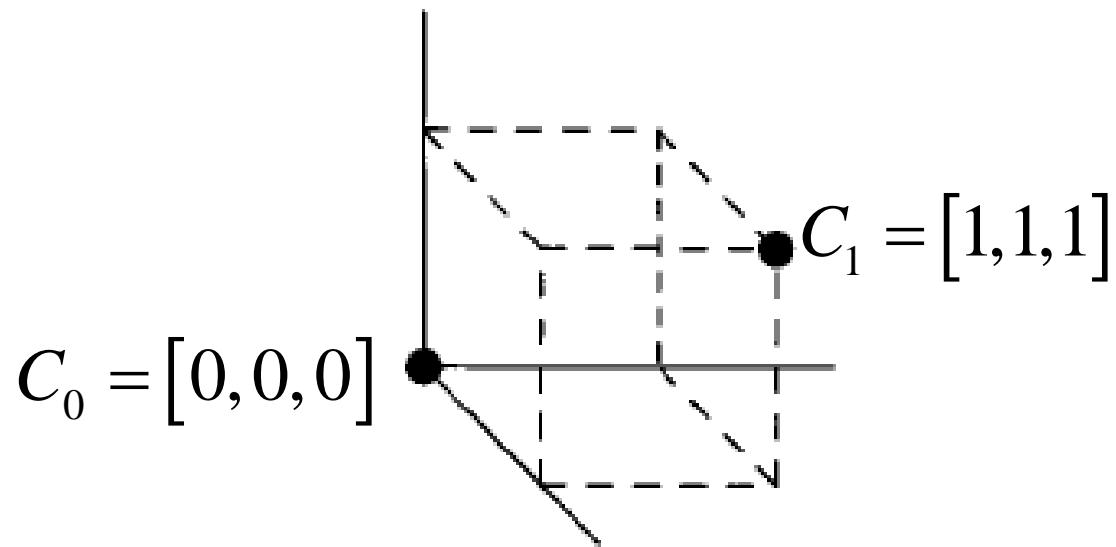
Quá trình mã hóa được biểu diễn dưới dạng ma trận. Ma trận sinh của mã lặp là G

$$C = mG$$

$$G = [1, 1, 1, \dots, 1]$$

Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

- Biểu diễn mã lặp (3,1) trong không gian



- Khoảng cách Hamming trong ma binary được tính bằng số các điểm khác biệt trong 2 từ mã.

$$C_1 = [1, 0, 1, 1, 1, 0]$$

$$C_2 = [1, 1, 0, 1, 1, 1]$$

$$d_{12} = 3$$

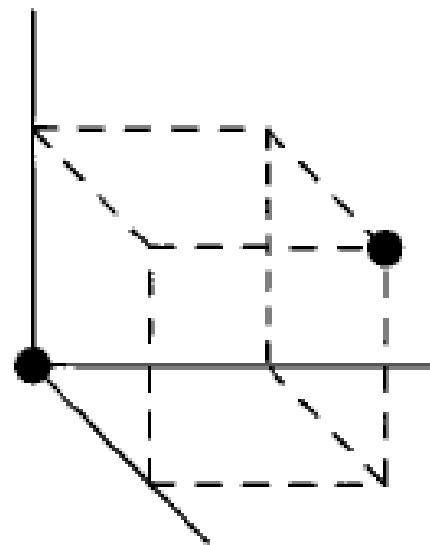
Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

- Định nghĩa : Khoảng cách tối thiểu (min distance) là khoảng cách Hamming nhỏ nhất giữa 2 từ mã bất kì.

$$d_{\min} = \min_{\mathbf{c}_i, \mathbf{c}_j \in C, \mathbf{c}_i \neq \mathbf{c}_j} d_H(\mathbf{c}_i, \mathbf{c}_j).$$

- Phương pháp giải mã ML: tìm kiếm từ mã có khoảng cách gần nhất với từ mã thu được.

$$\hat{\mathbf{c}} = \arg \min_{\mathbf{c} \in C} d_H(\mathbf{r}, \mathbf{c})$$



Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

- ☐ Liên hệ giữa khoảng cách Hamming tối thiểu và khả năng phát hiện và sửa lỗi.

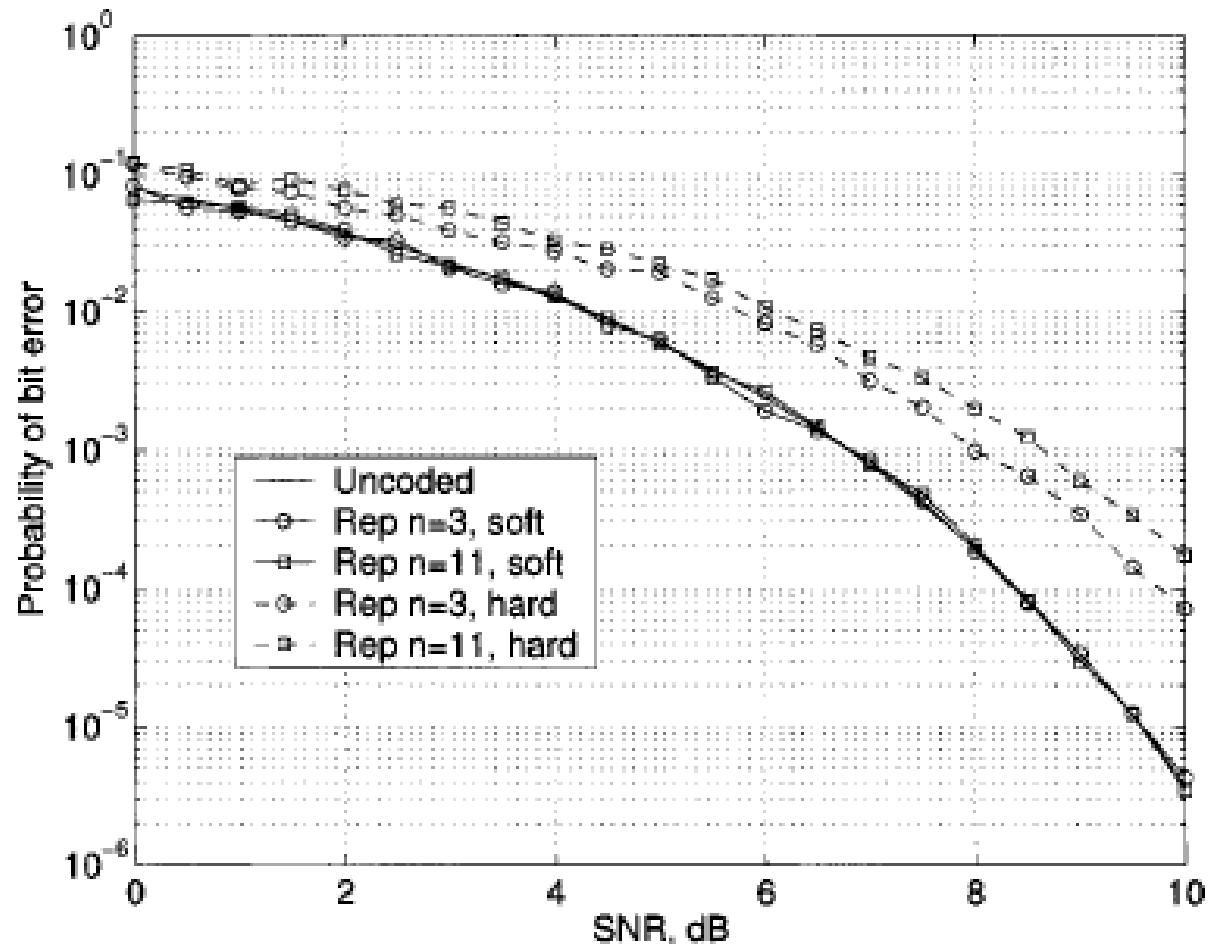
Với mã binary (n, k)

Khả năng phát hiện lỗi $d_{\min} - 1$

Khả năng sửa lỗi $t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$

Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

- ☐ Tỷ lệ lỗi bit (BER – Bit Error Rate) của mã lặp trong môi trường kênh AWGN (Additive White Gaussian Noise)



Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

□ Ví dụ mã kiểm tra chẵn

- Trong trường hợp $n = k+1$, bản tin được bổ sung thêm 1 bit kiểm tra chẵn lẻ
- Trong trường hợp số chẵn các bit '1', Bit kiểm tra chẵn lẻ có giá trị

$$q = \sum_{i=1}^k m_i \pmod{2}$$

- Trong trường hợp số lẻ các bit '1', bit kiểm tra có giá trị
$$1 - q$$
 - Bit kiểm tra chẵn lẻ được thêm vào đảm bảo số chẵn các bit '1' trong từ mã.
- Mã kiểm tra chẵn lẻ chỉ phát hiện được (tối đa) 1 lỗi, không có khả năng sửa lỗi.

Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

□ Ví dụ 1: Mã kiểm tra chẵn lẽ (6,5)

- Bản tin $m = (10110) \Rightarrow$ từ mã $c=(10110\boxed{1})$
- Bản tin $m = (11011) \Rightarrow$ từ mã $c=(11011\boxed{0})$

Phần 2: Các khái niệm cơ bản của mã hóa sửa lỗi

□ Ví dụ 2: Bảng mã kiểm tra chẵn lẻ (4,3)

Dataword			Codeword			
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	1	1	1	1

Phần 3: Mã khối tuyến tính

- ☐ Mã khối (n,k) được biểu diễn dạng vector

Bản tin $d = (d_1 \ d_2 \dots \ d_k)$

Từ mã $c = (c_1 \ c_2 \dots \ c_n)$

- ☐ Mã khối được xây dựng

$$c = dG$$

Với G là ma trận sinh

$$G = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ a_{k1} & a_{k2} & \dots & a_{kn} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix}$$

Phần 3: Mã khối tuyến tính

$$c = \sum_{i=1}^k d_i a_i$$

- Để đảm bảo 2 bản tin không có chung 1 từ mã (không thể giải mã/sửa lỗi), các hệ số a_i phải độc lập tuyến tính.
- Nếu c_i, c_k là 2 từ mã bất kì
 $c = c_i + c_k$ cũng là 1 từ mã
- Hệ quả: khối gồm toàn bit '0' cũng là 1 từ mã

Phần 3: Mã khối tuyến tính

Khả năng sửa lỗi của mã khối tuyến tính

- Khoảng cách Hamming của mã khối tuyến tính là khoảng cách Hamming nhỏ nhất của các từ mã khác '0'
- Để tìm khoảng cách Hamming nhỏ nhất, cần tìm kiểm tra 2^k từ mã để tìm khoảng cách Hamming nhỏ nhất.

Phần 3: Mã khối tuyến tính

Ví dụ 1: mã khối tuyến tính

- Với mã (4,2), có ma trận sinh

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad a_1 = [1011] \\ a_2 = [0101]$$

- Với $d=[1,1]$

$$\begin{array}{r} & 1 & 0 & 1 & 1 \\ c = & + & 0 & 1 & 0 & 1 \\ & - & - & - & - \\ = & 1 & 1 & 1 & 0 \end{array}$$

Phần 3: Mã khối tuyến tính – Giải mã sửa lỗi

Error Syndrome

- Để sửa lỗi mã khối tuyến tính, sử dụng phương pháp Error Syndrome
 - Nếu c_r là từ mã thu được ở phía thu, vector phát hiện lỗi (error syndrome) s của c_r

$$s = c_r H^T$$

- Nếu c_r bị lỗi, gọi e là vector lỗi

$$c_r = c + e$$

do đó

$$s = (c + e) H^T = cH^T + eH^T$$

$$s = 0 + eH^T$$

Vector syndrome có giá trị chỉ phụ thuộc vào vector lỗi e .

Error Syndrome

- Nhận xét: nếu cộng vector e với các từ mã khác thì vẫn thu được 1 vector syndrome.
 - Tổng cộng có $2^{(n-k)}$ syndromes, 2^n vector lỗi.
 - Ví dụ: với mã $(3,2)$, có 2 syndromes và 8 vector lỗi e . Rõ ràng không thể sửa tất cả các lỗi trong trường hợp này.
 - Ví dụ: với mã $(7,4)$ có 8 syndromes và 128 vector lỗi e .
 - Vì vậy, với 8 syndromes, ta cần bố trí các giá trị khác nhau để sửa được 7 lỗi (lỗi 1 bit) và 1 trường hợp không lỗi $s=0$.

Phần 3: Mã khối tuyến tính – Giải mã sửa lỗi

Bảng liệt kê lỗi

- Bảng liệt kê lỗi được xây dựng như sau:

c_1 (all zero)	c_2	c_M	s_0
e_1	$c_2 + e_1$	$c_M + e_1$	s_1
e_2	$c_2 + e_2$	$c_M + e_2$	s_2
e_3	$c_2 + e_3$	$c_M + e_3$	s_3
...
e_N	$c_2 + e_N$	$c_M + e_N$	s_N

Các hàng đều có chung giá trị syndrome



Các hàng khác nhau có vector syndrome khác nhau

Bảng có 2^k cột (tương ứng với các từ mã hợp lệ) và 2^{n-k} hàng (số lượng các syndrome)

Phần 3: Mã khối tuyến tính – Giải mã sửa lỗi

- Bảng liệt kê lỗi được xây dựng theo cách
 - Liệt kê tất cả các lỗi 1 bit
 - Liệt kê tất cả các lỗi 2 bit
 -
- Kiểm tra đảm bảo các lỗi được bổ sung vào bảng có vector syndrome khác nhau. Việc xây dựng bảng kết thúc khi sử dụng hết các vector syndrome.
- Để giải mã:
 - Tính vector syndrome theo công thức $s = 0 + eH^T$
 - Tra bảng tương ứng, tìm vector lỗi e tương ứng.
 - Cộng modulo vector e và từ mã thu được để giải mã

$$c = c_r + e$$

Phần 3: Mã khối tuyến tính – Mã hệ thống

- Mã hệ thống có phần thông tin và phần kiểm tra được phân tách trong từ mã, phần thông tin là không thay đổi so với ban đầu
- Ma trận sinh của mã hệ thống có dạng

$$G = \begin{bmatrix} & \xleftarrow{k} & & \xleftarrow{R} & & & \\ & 1 & 0 & .. & 0 & p_{11} & p_{12} & .. & p_{1R} \\ & 0 & 1 & .. & 0 & p_{21} & p_{22} & .. & p_{2R} \\ .. & .. & .. & .. & .. & .. & .. & .. & .. \\ & 0 & 0 & .. & 1 & p_{k1} & p_{k2} & .. & p_{kR} \end{bmatrix} = [I | P]$$

$$R = n - k$$

Phần 3: Mã khối tuyến tính – Mã hệ thống

- ☐ Với mã hệ thống, ma trận kiểm tra được xây dựng

$$G = [I | P] \quad \text{and so} \quad H = [-P^T | I]$$

- ☐ Ví dụ: mã (7,4) với khoảng cách Hamming $d_{min}= 3$

$$G = [I | P] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad H = [-P^T | I] = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Phần 3: Mã khối tuyến tính – Mã hệ thống

Tìm vector syndrome trong ví dụ

- Từ mã thu được $c_r = [1101001]$

$$s = c_r H^T = [1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1] \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [0 \ 0 \ 0]$$

Phần 4: Mã Hamming

- Mã Hamming là dạng đặc biệt của mã khối tuyến tính
- Với mỗi $r \geq 2$
 - Từ mã có độ dài $n = 2^r - 1$
 - Bản tin có độ dài $k = 2^r - r - 1$
 - Tốc độ mã $R = \frac{k}{n} = 1 + \frac{r}{2^r - 1}$
 - Khoảng cách Hamming $d_{\min} = 3$, khả năng sửa 1 lỗi
- Mã Hamming là mã có tốc độ mã R lớn nhất với cùng khoảng cách Hamming $d_{\min} = 3$

Phần 4: Mã Hamming (7,4)

□ Mã Hamming (7,4) dạng hệ thống

$$G = [I | P] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad H = [-P^T | I] = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

□ Mã Hamming (7,4) dạng không hệ thống

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Phần 4: Mã Hamming (7,4)

- Ví dụ: mã Hamming (7,4) không hệ thống

$$d = 1011$$

$$c = 1110000$$

$$+ 0101010$$

$$+ 1101001$$

$$= 0110011$$

$$e = 0010000$$

$$c_r = 0100011$$

$$s = c_r H^T = e H^T = 011$$

- Chú ý: giá trị vector syndrome là vị trí lỗi.

Phần 4: Mã Hamming (7,4) – BER

- ❑ For a given channel bit error rate (BER), what is the BER after correction (assuming a memoryless channel, i.e., no burst errors)?
- ❑ To do this we will compute the probability of receiving 0, 1, 2, 3, errors
- ❑ And then compute their effect

Bit Error Rates after Decoding

- Example – A (7,4) Hamming code with a channel BER of 1%, i.e., $p = 0.01$

$$P(0 \text{ errors received}) = (1 - p)^7 = 0.9321$$

$$P(1 \text{ error received}) = 7p(1 - p)^6 = 0.0659$$

$$P(2 \text{ errors received}) = \frac{7 \times 6}{2} p^2 (1 - p)^5 = 0.002$$

$$P(3 \text{ or more errors}) = 1 - P(0) - P(1) - P(2) = 0.000034$$

Bit Error Rates after Decoding

- Single errors are corrected, so,
 $0.9321 + 0.0659 = 0.998$ codewords are correctly detected
- Double errors cause 3 bit errors in a 7 bit codeword, i.e.,
 $(3/7)*4$ bit errors per 4 bit dataword, that is $3/7$ bit errors per bit.
Therefore the double error contribution is $0.002*3/7 = 0.000856$

Bit Error Rates after Decoding

- The contribution of triple or more errors will be less than 0.000034 (since the worst that can happen is that every databit becomes corrupted)
- So the BER after decoding is approximately $0.000856 + 0.000034 = 0.0009 = 0.09\%$
- This is an improvement over the channel BER by a factor of about 11

Perfect Codes

So,

$$2^R \geq 1 + n$$

to correct up to 1 error

$$\geq 1 + n + \frac{n(n-1)}{2}$$

to correct up to 2 errors

$$\geq 1 + n + \frac{n(n-1)}{2} + \frac{n(n-1)(n-2)}{6}$$

to correct up to 3 errors

↑ If equality then code is Perfect

Only known perfect codes are SEC Hamming codes and TEC Golay (23,12) code ($d_{\min}=7$).

Using previous equation yields

$$1 + 23 + \frac{23(23-1)}{2} + \frac{23(23-1)(23-2)}{6} = 2048 = 2^{11} = 2^{(23-12)}$$

Cơ sở lí thuyết thông tin

Chương 4: Mã vòng CRC

TS. Phạm Hải Đăng

Phần 1: Khái niệm cơ bản

☐ Định nghĩa Mã vòng

- Mã vòng là mã khối tuyến tính $C(n,k)$.
- Nếu c là từ mã của mã vòng $C(n,k)$, các dịch vòng của từ mã c cũng là từ mã của mã vòng $C(n,k)$.

$$c = (c_0, c_1, \dots, c_{n-1})$$

$$c^{(1)} = (c_{n-1}, c_0, c_1, \dots, c_{n-2})$$

- ☐ Cấu trúc dịch vòng giúp cho việc tính toán mã hóa và giải mã, tính toán vector syndrome trở nên dễ dàng.

Phần 1: Khái niệm cơ bản

☐ Biểu diễn mă vòng dưới dạng đa thức

$$c(x) = c_0 + c_1x + c_2x^2 + \dots + c_{n-1}x^{n-1}$$

$$c^{(1)}(x) = c_{n-1} + c_0x + c_1x^2 + \dots + c_{n-2}x^{n-1}$$

- Mỗi từ mă $c(x)$ đều có bậc lớn hơn hoặc bằng $n-k$, nhỏ hơn hoặc bằng $n-1$

$$(c_{n-1}, c_0, c_1, \dots, c_{n-2}) \leftrightarrow xc(x)$$

$$(c_{n-2}, c_{n-1}, c_0, \dots, c_{n-3}) \leftrightarrow x^2c(x)$$

⋮

$$(c_1, c_2, \dots, c_{n-1}, c_0) \leftrightarrow x^{n-1}c(x)$$

Phần 1: Khái niệm cơ bản

☐ Đa thức sinh $g(x)$

- Chỉ có duy nhất một đa thức sinh $g(x)$ với mỗi mă vòng.
- Bậc của đa thức sinh $g(x)$ phải nhỏ hơn hoặc bằng $n-k$.
- Đa thức từ mă $c(x)$ phải chia hết cho đa thức sinh $g(x)$.

$$g(x) = g_0 + g_1x + g_2x^2 + \dots + g_{n-k}x^{n-k}$$

$$g_0 = g_{n-k} = 1$$

☐ Đa thức từ mă đều có thể biểu diễn dưới dạng

$$c(x) = m(x)g(x)$$

trong đó $m(x) = m_0 + m_1x + m_2x^2 + \dots + m_{k-1}x^{k-1}$ là đa thức
bản tin

Phần 1: Khái niệm cơ bản

☐ Tính chất của đa thức sinh

- Đa thức sinh $g(x)$ luôn được là đa thức con của đa thức $x^n - 1$
- Tất cả các đa thức con của đa thức $x^n - 1$ với bậc $(n-k)$ đều có thể sử dụng làm đa thức sinh.
- Do $x^n - 1$ chia hết cho $g(x)$

$$x^n - 1 = h(x)g(x)$$

trong đó $h(x) = h_0 + h_1x + h_2x^2 + \dots + h_kx^k$

$$h_0 = h_k = 1$$

$h(x)$ là đa thức kiểm tra của đa thức sinh $g(x)$ của mă vòng (n,k) .

Phần 1: Khái niệm cơ bản

☐ Ví dụ:

$$x^{15} - 1 = (1 + x)(1 + x + x^2)(1 + x + x^4)(1 + x + x^2 + x^3 + x^4)(1 + x^3 + x^4)$$

- Các đa thức con có bậc 1, 2, 4, 4, 4.
- Đa thức $g(x) = (1 + x + x^2)(1 + x + x^4)(1 + x + x^2 + x^3 + x^4)$. được sử dụng cho mă vòng (15,5)
- Đa thức $g(x) = (1 + x)(1 + x + x^4)$ có thể sử dụng cho mă vòng (15, 10)

Bảng mã chuẩn CRC

Table 4.6: CRC Generators

CRC Code	Generator Polynomial
CRC-4	$g(x) = x^4 + x^3 + x^2 + x + 1$
CRC-7	$g(x) = x^7 + x^6 + x^4 + 1$
CRC-8	$g(x) = x^8 + x^7 + x^6 + x^4 + x^2 + 1$
CRC-12	$g(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-ANSI	$g(x) = x^{16} + x^{15} + x^2 + 1$
CRC-CCITT	$g(x) = x^{16} + x^{12} + x^5 + 1$
CRC-SDLC	$g(x) = x^{16} + x^{15} + x^{13} + x^7 + x^4 + x^2 + x + 1$
CRC-24	$g(x) = x^{24} + x^{23} + x^{14} + x^{12} + x^8 + 1$
CRC-32a	$g(x) = x^{32} + x^{30} + x^{22} + x^{15} + x^{12} + x^{11} + x^7 + x^6 + x^5 + x$
CRC-32b	$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Phần 2: Mã hóa và giải mã không hệ thống

- ☐ Vector bản tin $\mathbf{m} = [m_0 \quad m_1 \quad \dots \quad m_{k-1}]$
biểu diễn dạng đa thức $m(x) = m_0 + \dots + m_{k-1}x^{k-1}$
 - ☐ Từ mã dạng không hệ thống (nonsystematic)
- $$\begin{aligned} c(x) &= m(x)g(x) \\ &= (m_0g(x) + m_1xg(x) + \dots + m_{k-1}x^{k-1}g(x)) \end{aligned}$$
- ☐ Quá trình mã hóa không hệ thống biểu diễn ma trận

$$c(x) = [m_0 \quad m_1 \quad m_2 \quad \dots \quad m_{k-1}] \begin{bmatrix} g(x) \\ xg(x) \\ x^2g(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix}$$

Phần 2: Mã hóa và giải mã không hệ thống

☐ Biểu diễn dạng ma trận

$$c(x) = [m_0 \ m_1 \ m_2 \ \dots \ m_{k-1}] \begin{bmatrix} g(x) \\ xg(x) \\ x^2g(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix}$$

$$\mathbf{c}_m = [m_0, m_1, \dots, m_{k-1}] \begin{bmatrix} g_0 & g_1 & \cdots & g_r \\ g_0 & g_1 & \cdots & g_r \\ g_0 & g_1 & \cdots & g_r \\ \ddots & \ddots & & \ddots \\ & g_0 & g_1 & \cdots & g_r \\ & g_0 & g_1 & \cdots & g_r \end{bmatrix}$$

Phần 2: Mã hóa và giải mã không hệ thống

- Ví dụ: với mã vòng $n=7$

$$g(x) = (x^3 + x + 1)(x + 1) = 1 + x^2 + x^3 + x^4$$

Ma trận sinh G dạng không hệ thống được biểu diễn

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

- Bảng quan hệ giữa bản tin và từ mã (dạng vector và dạng đa thức)

m	$m(x)g(x)$	code polynomial	codeword
(0,0,0)	$0g(x)$	0	0000000
(1,0,0)	$1g(x)$	$1 + x^2 + x^3 + x^4$	1011100
(0,1,0)	$xg(x)$	$x + x^3 + x^4 + x^5$	0101110
(1,1,0)	$(x + 1)g(x)$	$1 + x + x^2 + x^5$	1110010
(0,0,1)	$x^2g(x)$	$x^2 + x^4 + x^5 + x^6$	0010111
(1,0,1)	$(x^2 + 1)g(x)$	$1 + x^3 + x^5 + x^6$	1001011
(0,1,1)	$(x^2 + x)g(x)$	$x + x^2 + x^3 + x^6$	0111001
(1,1,1)	$(x^2 + x + 1)g(x)$	$1 + x + x^4 + x^6$	1100101

Phần 2: Mã hóa và giải mã không hệ thống

Giải mã vòng dạng không hệ thống

$$\begin{aligned}c(x)h(x) &= m(x)g(x)h(x) \\&= m(x)(x^n - 1) \\&\equiv 0\end{aligned}$$

- Đa thức thu được phía thu $r(x)$. Để kiểm tra $r(x)$

$$s(x) = r(x)h(x) \pmod{x^n - 1}$$

- $s(x)$ là đa thức syndrome. $s(x)=0$ khi và chỉ khi $r(x)$ là từ mã.

Phần 2: Mã hóa và giải mã không hệ thống

Xây dựng ma trận kiểm tra dạng không hệ thống

$$c(x)h(x) = m(x)g(x)h(x) = m(x)(x^n - 1) = m(x) - m(x)x^n$$

- Do bậc của $m(x)$ nhỏ hơn k , do đó các hệ số $x^k, x^{k+1}, \dots, x^{n-1}$ bằng 0 trong đa thức $m(x) - m(x)x^n$

Do đó, $x^k, x^{k+1}, \dots, x^{n-1}$ có hệ số bằng 0 trong đa thức $c(x)h(x)$

$$\sum_{i=0}^k h_i c_{l-i} = 0 \text{ for } l = k, k+1, \dots, n-1.$$

- Ma trận kiểm tra dạng không hệ thống được biểu diễn dạng

$$\begin{bmatrix} h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \\ h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \\ h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \\ \ddots & & & & \ddots \\ h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \mathbf{0}$$

Phần 2: Mã hóa và giải mã không hệ thống

Xây dựng ma trận kiểm tra dạng không hệ thống

$$c(x)h(x) = m(x)g(x)h(x) = m(x)(x^n - 1) = m(x) - m(x)x^n$$

- Do bậc của $m(x)$ nhỏ hơn k , do đó các hệ số $x^k, x^{k+1}, \dots, x^{n-1}$ bằng 0 trong đa thức $m(x) - m(x)x^n$

Do đó, $x^k, x^{k+1}, \dots, x^{n-1}$ có hệ số bằng 0 trong đa thức $c(x)h(x)$

$$\sum_{i=0}^k h_i c_{l-i} = 0 \text{ for } l = k, k+1, \dots, n-1.$$

- Ma trận kiểm tra dạng không hệ thống được biểu diễn dạng

$$H = \begin{bmatrix} h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \\ & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \\ & & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \\ & & & \ddots & & \ddots & \\ & & & & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \end{bmatrix}$$

Phần 2: Mã hóa và giải mã không hệ thống

- ☐ Ví dụ: Cho mã vòng (7,3) với đa thức sinh $g(x) = x^4 + x^3 + x^2 + 1$

Xây dựng ma trận sinh và ma trận kiểm tra dạng không hệ thống.

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

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

Phần 3: Mã hóa và giải mã dạng hệ thống

Cho đa thức bản tin $m(x)$ và đa thức sinh $g(x)$.

Từ mã dạng hệ thống được xây dựng theo công thức sau:

- Thực hiện phép chia đa thức lấy phần dư $d(x)$

$$x^{n-k}m(x) = q(x)g(x) + d(x)$$

- Từ mã dạng hệ thống của bản tin $m(x)$ được tính như sau

$$c(x) = x^{n-k}m(x) - d(x)$$

- Từ mã $c(x)$ thỏa mãn điều kiện chia hết cho đa thức sinh $g(x)$, có bậc lớn hơn $(n-k)$ và nhỏ hơn n .
- Từ mã được biểu diễn dạng vector

$$c = [-d_0, -d_1, \dots, -d_{n-k-1}, m_0, m_1, \dots, m_{k-1}]$$

Phần 3: Mã hóa và giải mã dạng hệ thống

- Biểu diễn ma trận sinh và ma trận kiểm tra dạng hệ thống

$$\left[\begin{array}{cccc|c} g_0 & g_1 & \cdots & g_r & \mod(x^{n-k}) \\ g_0 & g_1 & \cdots & g_r & \mod(x^{n-k+1}) \\ g_0 & g_1 & \cdots & g_r & \mod(x^{n-k+2}) \\ \vdots & \ddots & & \ddots & \\ & & & g_0 & \mod(x^{n-1}) \\ & & & g_0 & g_1 & \cdots & g_r \\ & & & g_0 & g_1 & \cdots & g_r \end{array} \right] \rightarrow \mod(x^{n-1})$$

Tương đương với việc thực hiện phép chia lấy phần dư với các hàng trong ma trận G

$$x^{n-k+i} = q_i(x)g(x) + b_i(x), \quad i = 0, 1, \dots, k-1$$

Thu được ma trận sinh dạng hệ thống

$$G = \left[\begin{array}{ccccc|cccc} -b_{0,0} & -b_{0,1} & \cdots & -b_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ -b_{1,0} & -b_{1,1} & \cdots & -b_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ -b_{2,0} & -b_{2,1} & \cdots & -b_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & & & & & & \\ -b_{k-1,0} & -b_{k-1,1} & \cdots & -b_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{array} \right] = [P | I]$$

Phần 3: Mã hóa và giải mã dạng hệ thống

Ma trận kiểm tra dạng hệ thống

$$H = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & b_{0,0} & b_{1,0} & b_{2,0} & \cdots & b_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & b_{0,1} & b_{1,1} & b_{2,1} & \cdots & b_{k-1,1} \\ 0 & 0 & 1 & \cdots & 0 & b_{0,2} & b_{1,2} & b_{2,2} & \cdots & b_{k-1,2} \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & \cdots & 1 & b_{0,n-k-1} & b_{1,n-k-1} & b_{2,n-k-1} & \cdots & b_{k-1,n-k-1} \end{bmatrix} = [I | P^T]$$

Phần 3: Mã hóa và giải mã dạng hệ thống

Ví dụ: Cho đa thức sinh $g(x) = 1 + x + x^3$

Biểu diễn ma trận sinh và ma trận kiểm tra dạng hệ thống.

$$i = 0 : \quad x^3 \quad = g(x) + (1 + x) \quad b_0(x) = 1 + x$$

$$i = 1 : \quad x^4 \quad = xg(x) + (x + x^2) \quad b_1(x) = x + x^2$$

$$i = 2 : \quad x^5 = (x^2 + 1)g(x) + (1 + x + x^2) \quad b_2(x) = 1 + x + x^2$$

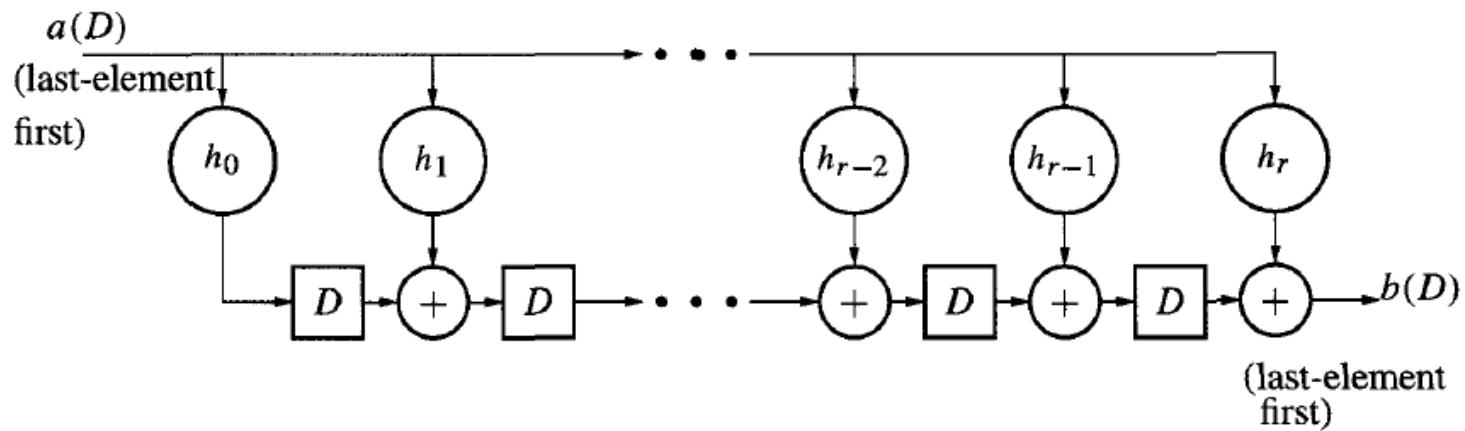
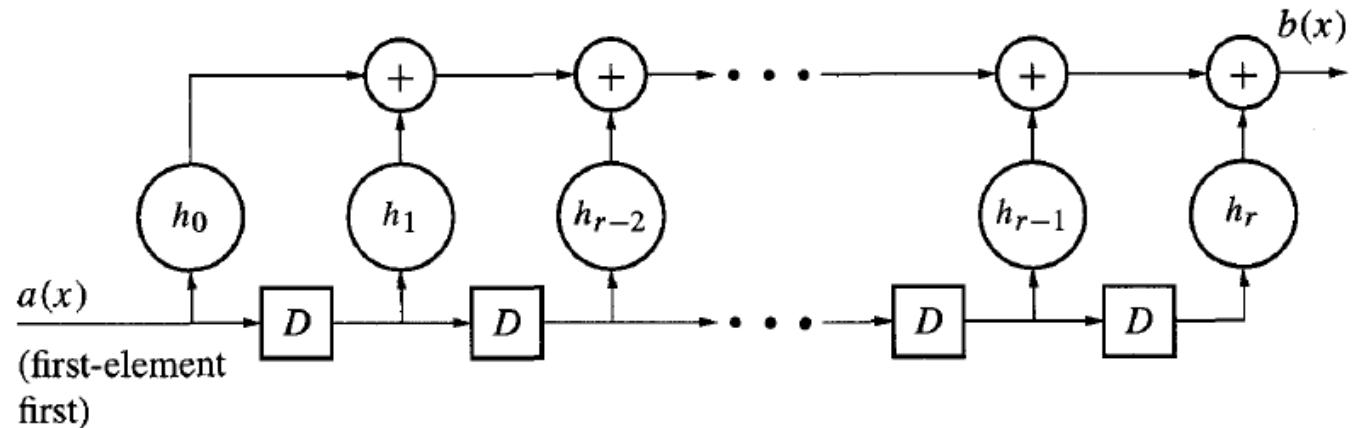
$$i = 3 : \quad x^6 = (x^3 + x + 1)g(x) + (1 + x^2) \quad b_3(x) = 1 + x^2$$

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Phần 4: Thực hiện phần cứng mã hóa giải mã vòng CRC

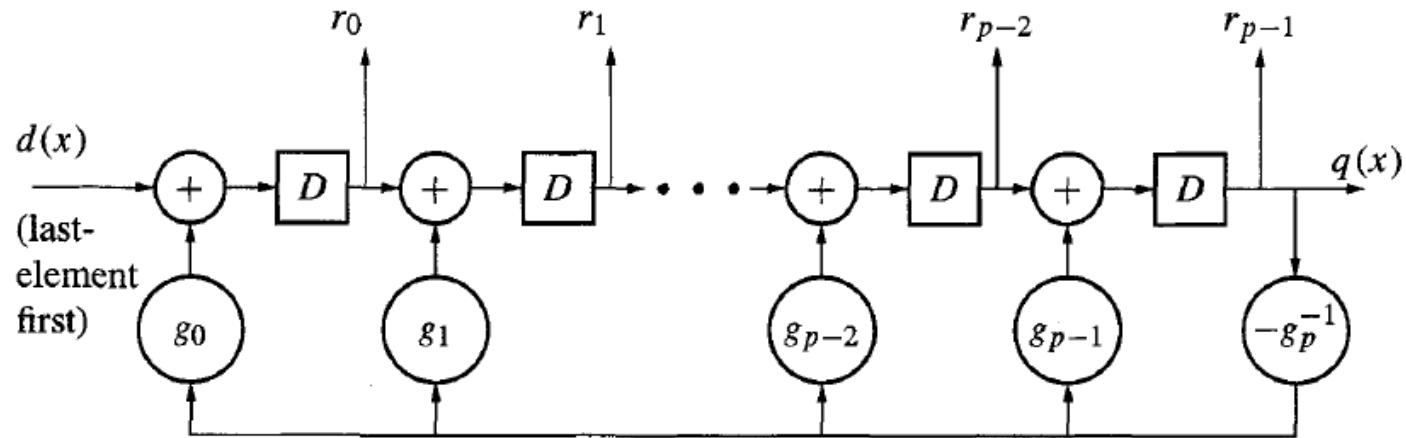
☐ Mạch nhân đa thức

(first-element first)

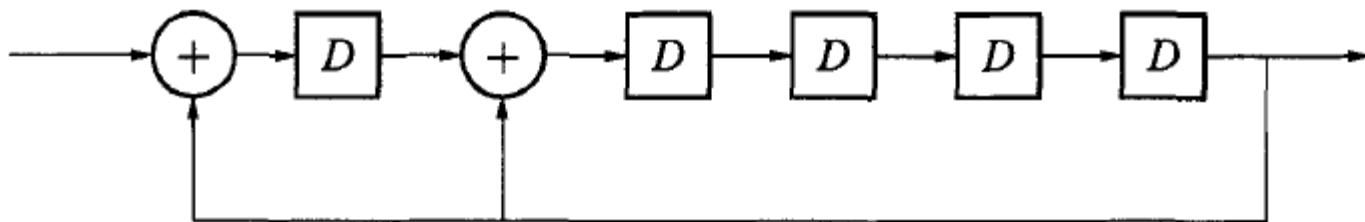


Phần 4: Thực hiện phần cứng mã hóa giải mã vòng CRC

☐ Mạch chia đa thức



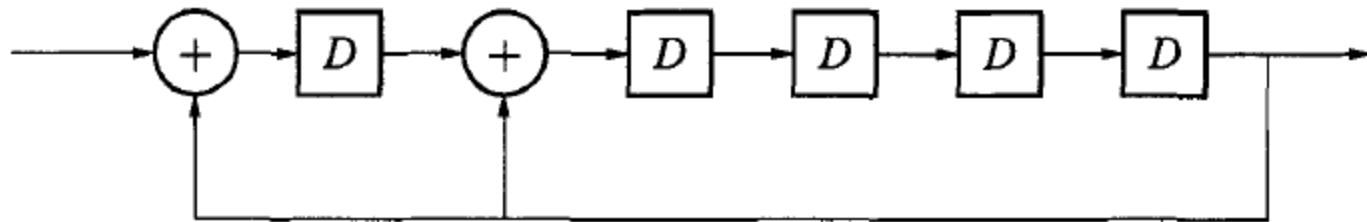
☐ Ví dụ: Mạch chia đa thức $g(x) = x^5 + x + 1$



Giá trị khởi tạo của các FF-D là ‘0’.

Phần 4: Thực hiện phần cứng mã hóa giải mã vòng CRC

- ☐ Ví dụ: Mạch chia đa thức $g(x) = x^5 + x + 1$



Bảng quan hệ đầu vào – đầu ra mạch chia $a(x) = x^8 + x^7 + x^5 + x + 1$

j	Input Symbol on jth Shift		Shift Register Contents After j Shifts					Output Symbol on jth Shift	
	bit	polynomial term	bits		polynomial representation			bit	polynomial term
0	-	-	0	0	0	0	0		
1	1	(x^8)	1	0	0	0	0		
2	1	(x^7)	1	1	0	0	0		
3	0	(x^6)	0	1	1	0	0		
4	1	(x^5)	1	0	1	1	0		
5	0	(x^4)	0	1	0	1	1	A:	$x^5 + x^7 + x^8$
6	0	(x^3)	1	1	1	0	1	B:	$x^3 + x^4 + x^5 + x^7$
7	0	(x^2)	1	0	1	1	0	C:	$x^2 + x^4 + x^5$
8	1	(x^1)	1	1	0	1	1		$x + x^2 + x^4 + x^5$
9	1	1	0	0	1	0	1	D:	$x^2 + x^4$

Cơ sở lí thuyết thông tin

Chương 5: Mã tích chập Thuật toán giải mã Viterbi

TS. Phạm Hải Đăng

Phần 1: Khái niệm cơ bản

☐ Định nghĩa Mã tích chập

- Mã tích chập là 1 dạng mã tuyến tính.
- Mã tích chập có cấu trúc giống 1 bộ lọc số - phép tích chập.
- Bộ mã hóa tích chập có thể coi như 1 tập hợp các bộ lọc số - hệ thống tuyến tính, bát biến theo thời gian.
- Đầu vào của bộ mã hóa tích chập là một dòng dữ liệu (data stream) biểu diễn dạng vector $m(x) = [m^{(1)}(x) \quad m^{(2)}(x) \quad \dots]$

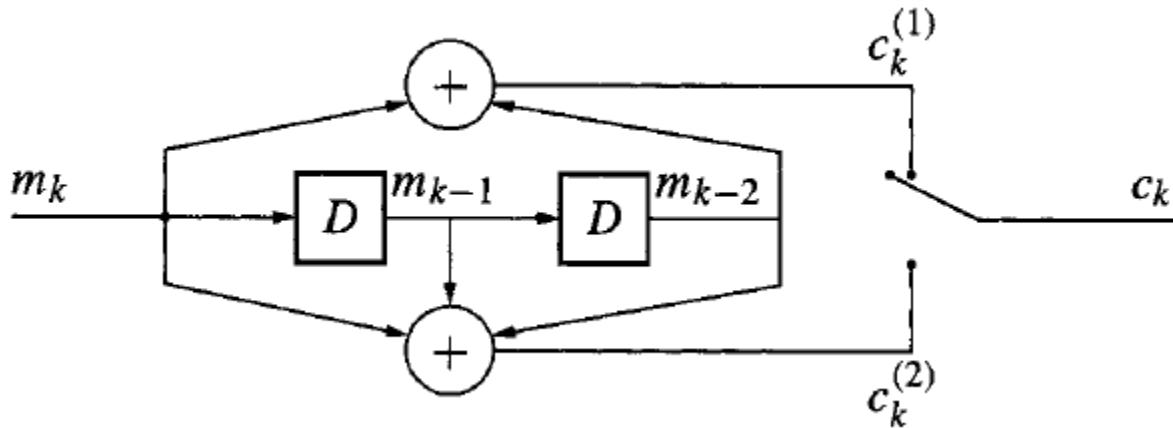
$$m^{(1)}(x) = m_0^{(1)} + m_1^{(1)}x + m_2^{(1)}x^2 \dots$$

$$m^{(2)}(x) = m_0^{(2)} + m_1^{(2)}x + m_2^{(2)}x^2 \dots$$

- Tốc độ mã $R=k/n$
- Chiều dài ràng buộc K (constraint length) là kích thước của thanh ghi (số lượng D-FF).

Phần 1: Khái niệm cơ bản

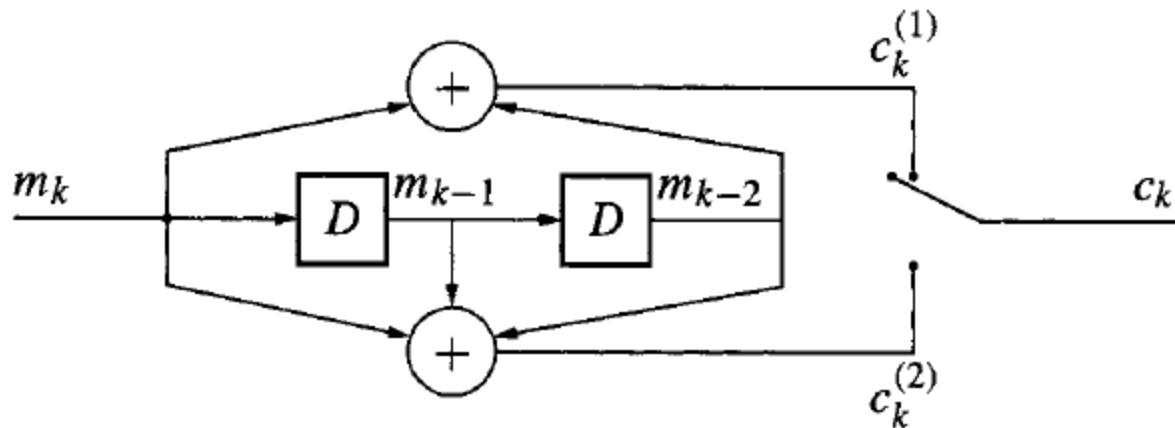
- Ví dụ: Mô hình tích chập có $R=1/2$



- Với đầu vào $\mathbf{m} = \{1, 1, 0, 0, 1, 0, 1\}$
- Đầu ra $\mathbf{c}^{(1)} = \{1, 1, 1, 1, 1, 0, 0, 0, 1\}$
 $\mathbf{c}^{(2)} = \{1, 0, 0, 1, 1, 1, 0, 1, 1\}$
- Biểu diễn đầu ra dạng vector
 $\mathbf{c} = \{11, 10, 10, 11, 11, 01, 00, 01, 11\}$

Phần 1: Khái niệm cơ bản

- Ví dụ: Mô hình tích chập có $R=1/2$



- Với đầu vào $\mathbf{m} = \{1, 1, 0, 0, 1, 0, 1\}$

Biểu diễn dạng đa thức $m(x) = 1 + x + x^4 + x^6$

- Đầu ra dạng đa thức

$$c^{(1)}(x) = m(x)g_1(x) = (1 + x + x^4 + x^6)(1 + x^2) = 1 + x + x^2 + x^3 + x^4 + x^8$$

$$c^{(2)}(x) = m(x)g_2(x) = (1 + x + x^4 + x^6)(1 + x + x^2) = 1 + x^3 + x^4 + x^5 + x^7 + x^8$$

Với đa thức sinh $g^{(1)}(x) = 1 + x^2$
 $g^{(2)}(x) = 1 + x + x^2$

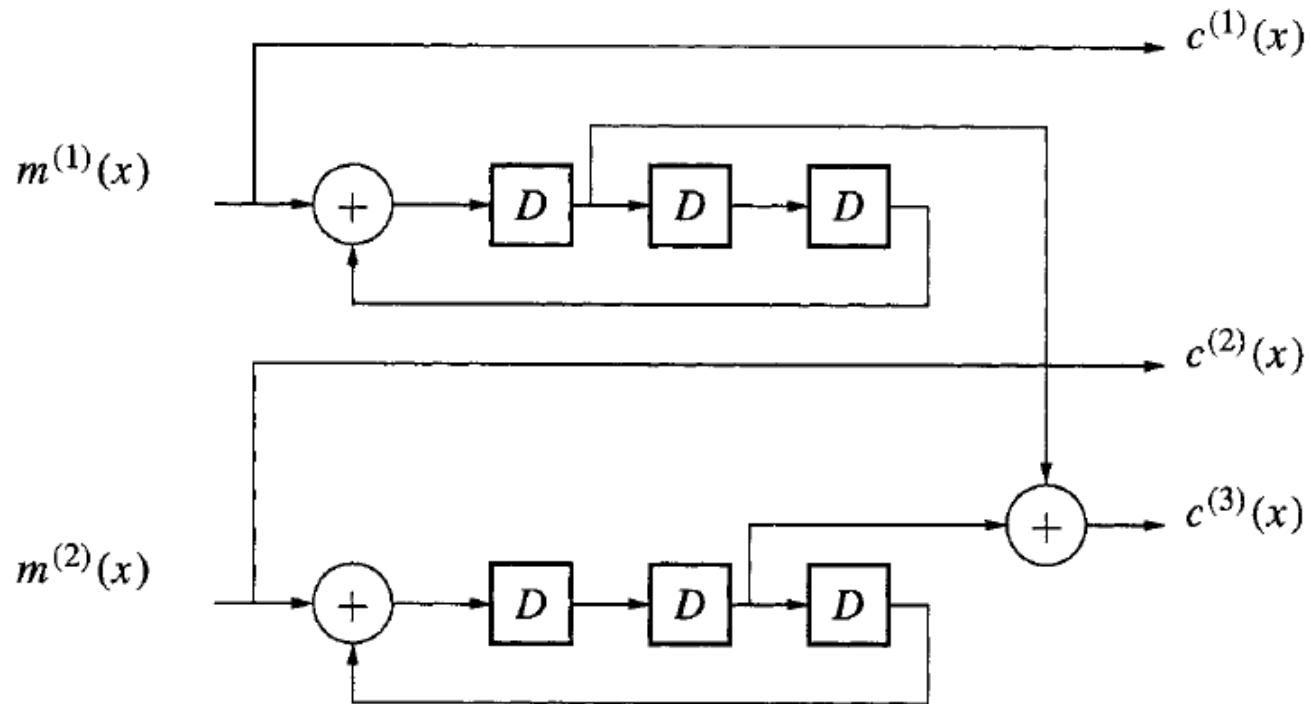
$$G_a(x) = [1 + x^2 \quad 1 + x + x^2]$$

Phần 1: Khái niệm cơ bản

- ☐ Ví dụ: Mô tích chập dạng hệ thống có ma trận đa thức sinh

$$G_1(x) = \begin{bmatrix} 1 & 0 & \frac{x}{1+x^3} \\ 0 & 1 & \frac{x^2}{1+x^3} \end{bmatrix}$$

Biểu diễn dạng sơ đồ mạch



Phần 1: Khái niệm cơ bản

- Biểu diễn dạng tổng quan của đa thức bản tin và ma trận đa thức sinh

$$\mathbf{m}(x) = [m^{(1)}(x), m^{(2)}(x), \dots, m^{(k)}(x)]$$

$$G(x) = \begin{bmatrix} g^{(1,1)}(x) & g^{(1,2)}(x) & \dots & g^{(1,n)}(x) \\ g^{(2,1)}(x) & g^{(2,2)}(x) & \dots & g^{(2,n)}(x) \\ \vdots & & & \\ g^{(k,1)}(x) & g^{(k,2)}(x) & \dots & g^{(k,n)}(x) \end{bmatrix}$$

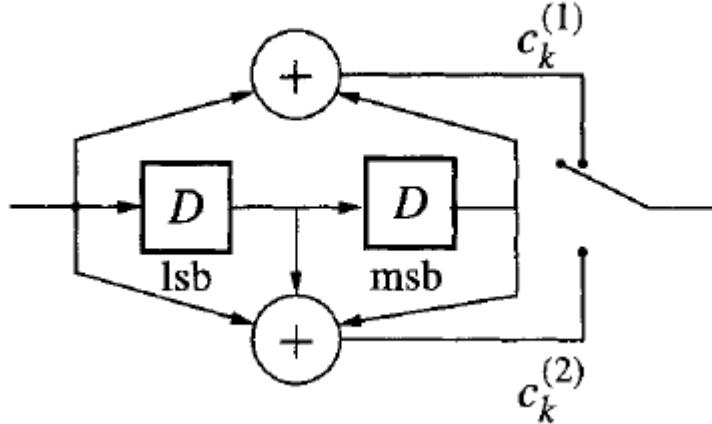
- Từ mã của mã tích chập

$$\mathbf{c}(x) = [c^{(1)}(x), c^{(2)}(x), \dots, c^{(n)}(x)] = \mathbf{m}(x)G(x)$$

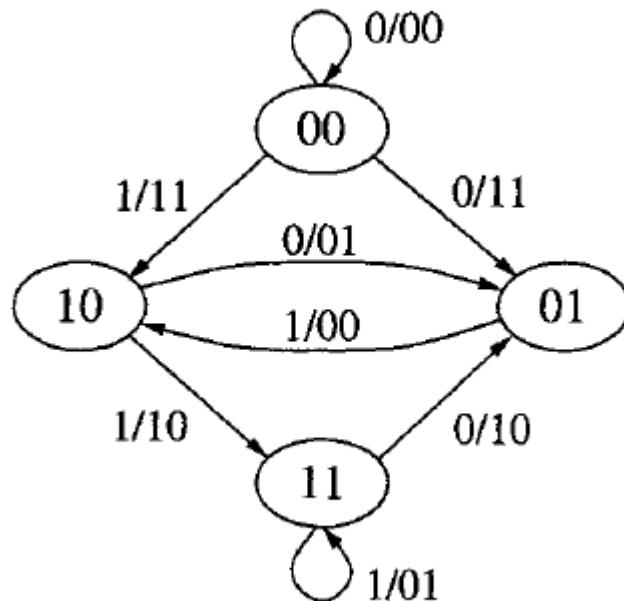
Phần 2: Biểu diễn sơ đồ trạng thái và sơ đồ lưới của mã tích chập

- Mã tích chập là một máy trạng thái (state machine), có thể biểu diễn bằng sơ đồ chuyển trạng thái

- Giá trị D-FF là trạng thái (state). Số lượng trạng thái 2^K
 - Đầu vào là kích thích chuyển trạng thái
 - Mũi tên mô tả quá trình chuyển trạng thái, với giá trị đầu vào/đầu ra.
- Ví dụ: 0/00 – Đầu vào $m=0$, đầu ra $c=[00]$



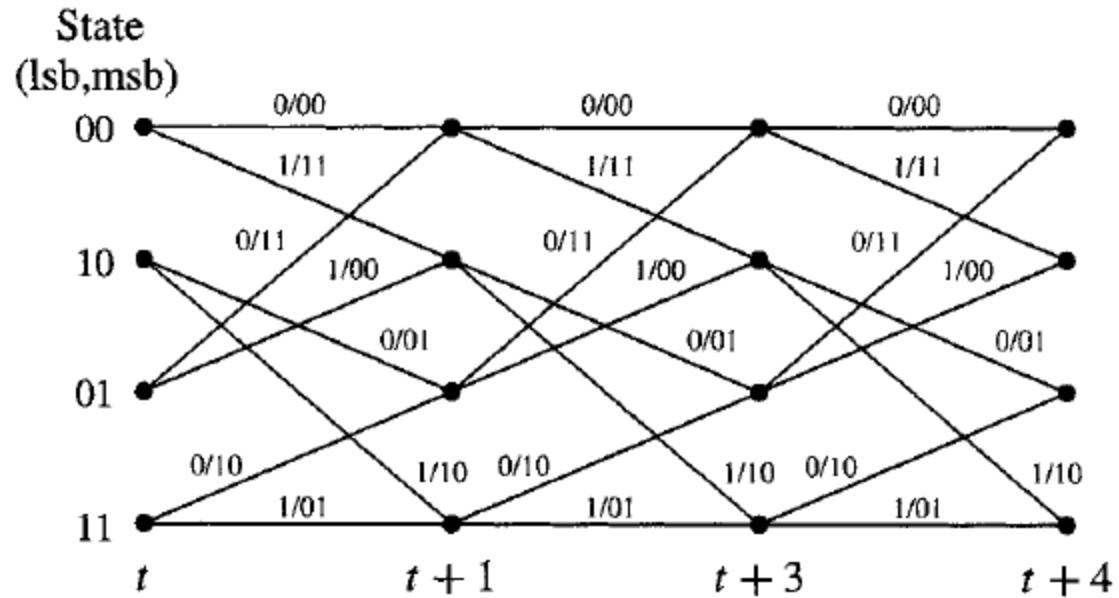
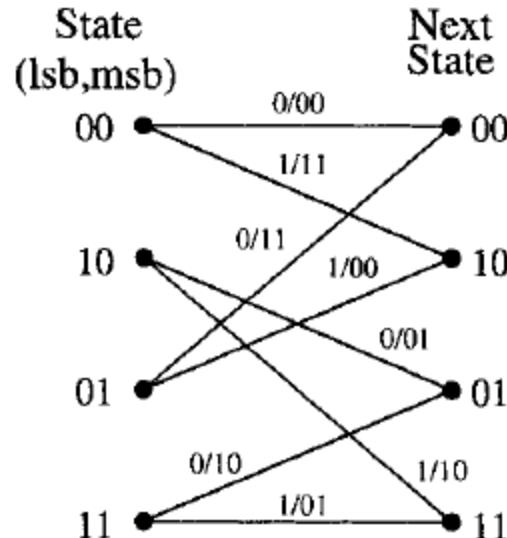
Bộ mã hóa tích chập R=1/2



Bộ mã hóa tích chập R=1/2

Phần 2: Biểu diễn sơ đồ trạng thái và sơ đồ lưới của mã tích chập

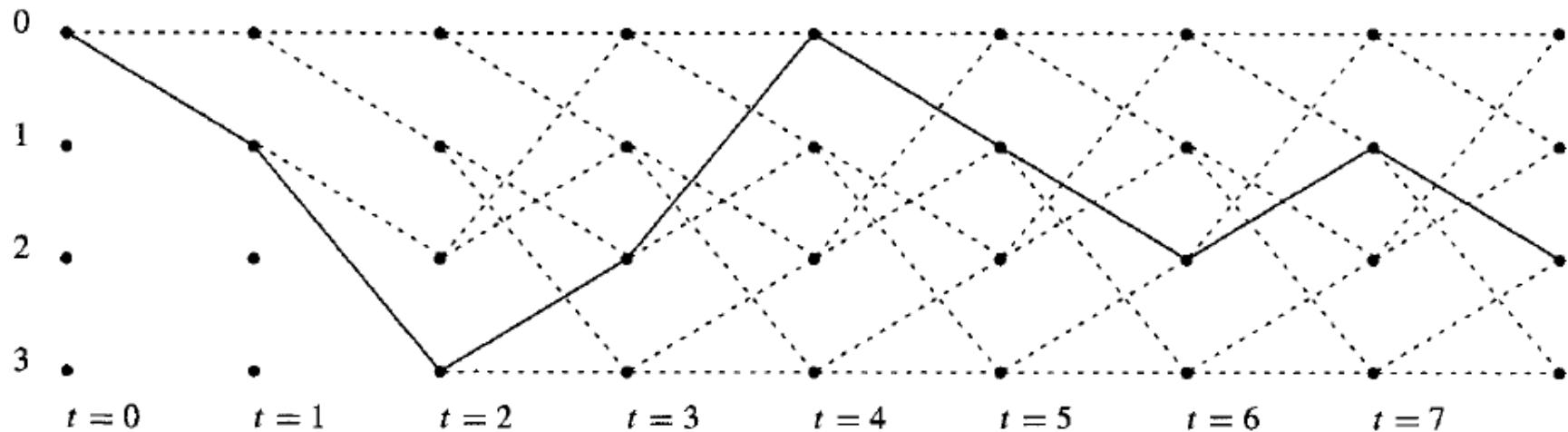
- Từ sơ đồ chuyển trạng thái, có thể chuyển sang sơ đồ lưới



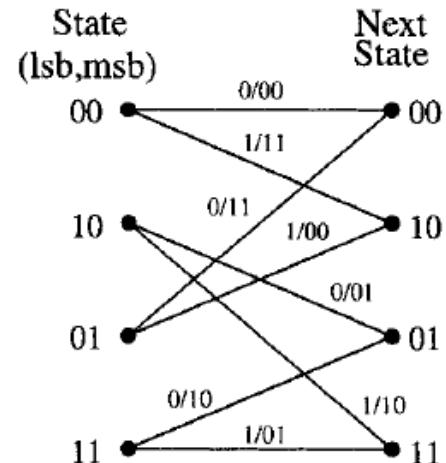
Phần 3: Thuật toán giải mã Viterbi

Sơ đồ lưới của mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$



t	Input m_k	Output c_t	State Ψ_{t+1}
0	1	11	1
1	1	10	3
2	0	10	2
3	0	11	0
4	1	11	1
5	0	01	2
6	1	00	1
7	0	01	2



Phần 3: Thuật toán giải mã Viterbi

- Thuật toán giải mã Viterbi thuộc lớp thuật toán giải mã ML (Maximum Likelihood).
- Thuật toán Viterbi là thuật toán tìm đường ngắn nhất, với quãng đường được tính toán là tổng khoảng cách Hamming của các nhánh trung gian.
 - P là trạng thái (state) đích, S trạng thái trung gian.
 - P_0 là tổng khoảng cách quãng đường tới state P với bit đầu vào giá trị '0', đi qua state S_0 . BR_0 là khoảng cách Hamming (đầu ra) của nhánh S_0-P
 - P_1 là tổng khoảng cách quãng đường tới state P với bit đầu vào giá trị '1', đi qua state S_1 . BR_1 là khoảng cách Hamming (đầu ra) của nhánh S_1-P



Algorithm 1 Path metric calculation in the conventional Viterbi algorithm

$$P_0 = PS_0 + BR_0$$

$$P_1 = PS_1 + BR_1$$

Phần 3: Thuật toán giải mã Viterbi

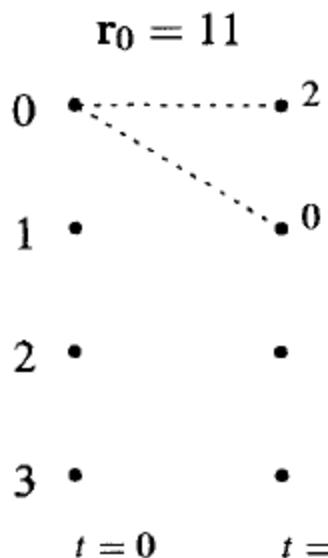
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào m=[1,1,0,0,1,0,1,0]

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ 10 \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=1$. Khoảng cách Hamming giữa đầu ra [11] và nhánh 0-0 và 0-1 lần lượt là 2 và 0



Phần 3: Thuật toán giải mã Viterbi

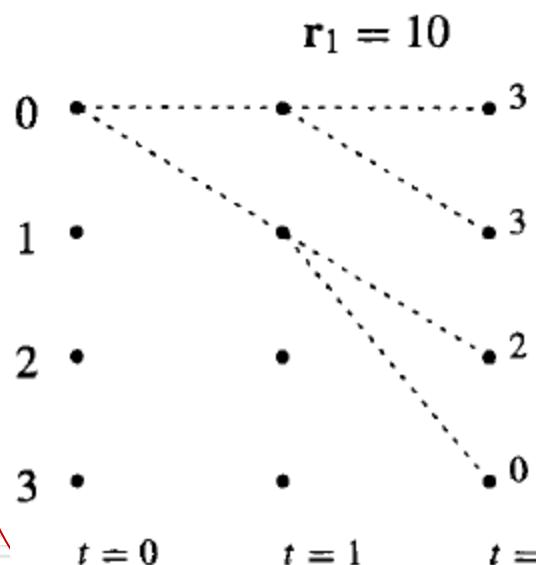
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ \underline{10} \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=2$, tính các khoảng cách Hamming tới các state



Phần 3: Thuật toán giải mã Viterbi

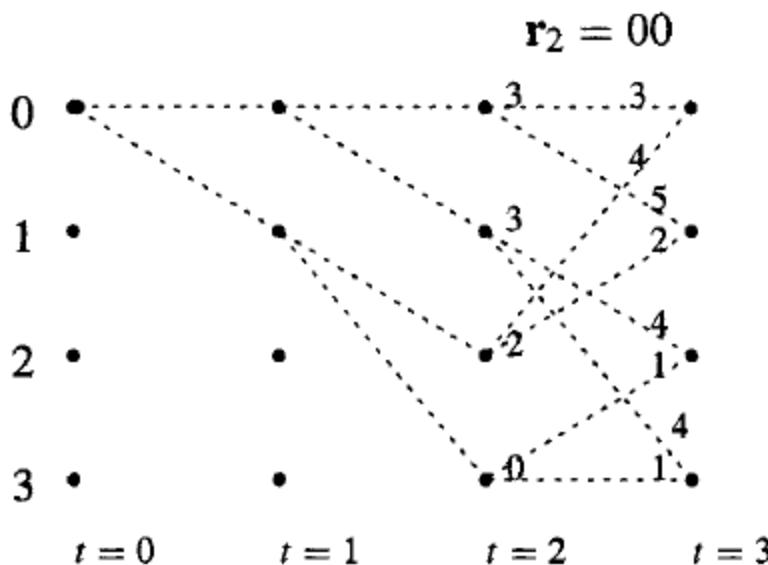
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ \underline{10} \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=3$, tính các khoảng cách Hamming tới các state



Phần 3: Thuật toán giải mã Viterbi

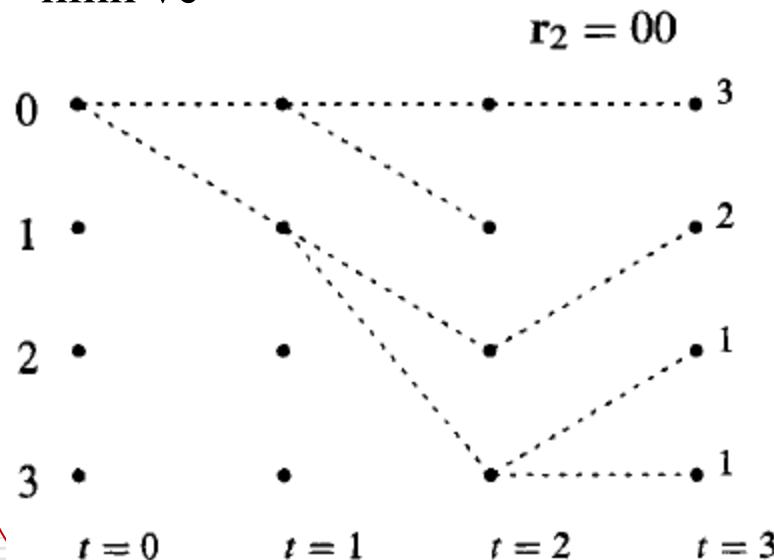
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ 10 \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=3$: xuất hiện 2 tuyến đường cùng tới 1 trạng thái. So sánh là loại bỏ tuyến đường có khoảng cách Hamming lớn. Tuyến đường giữ lại được biểu diễn như hình vẽ



Phần 3: Thuật toán giải mã Viterbi

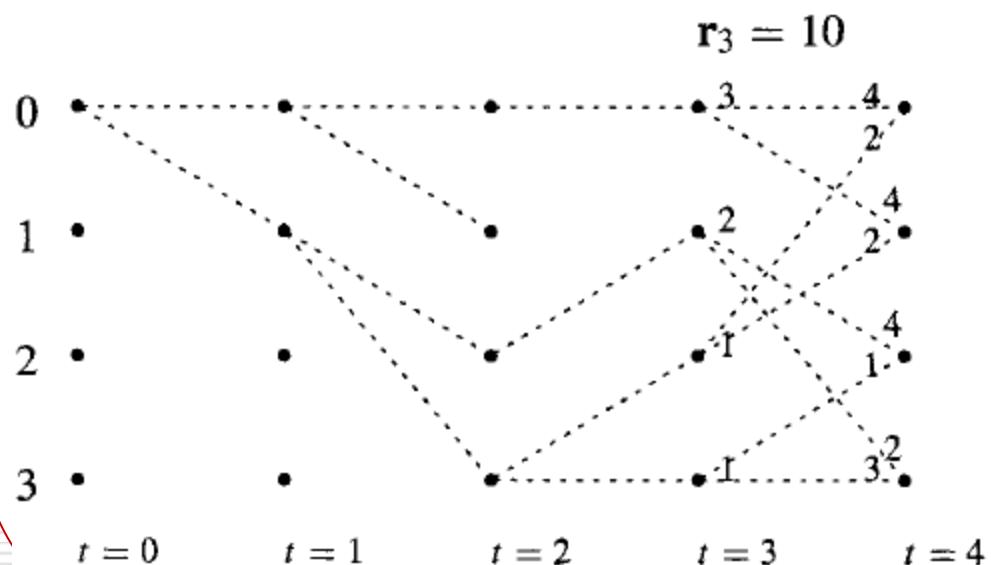
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ \underline{10} \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=4$: Mở rộng sơ đồ lưới, tính khoảng cách Hamming



Phần 3: Thuật toán giải mã Viterbi

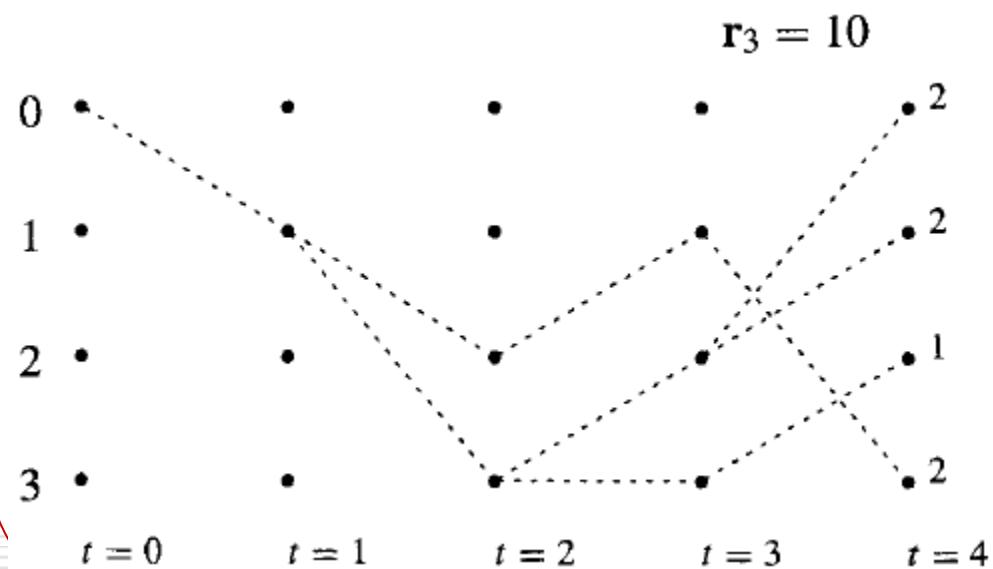
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ 10 \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=4$: Loại bỏ nhánh có khoảng cách Hamming lớn



Phần 3: Thuật toán giải mã Viterbi

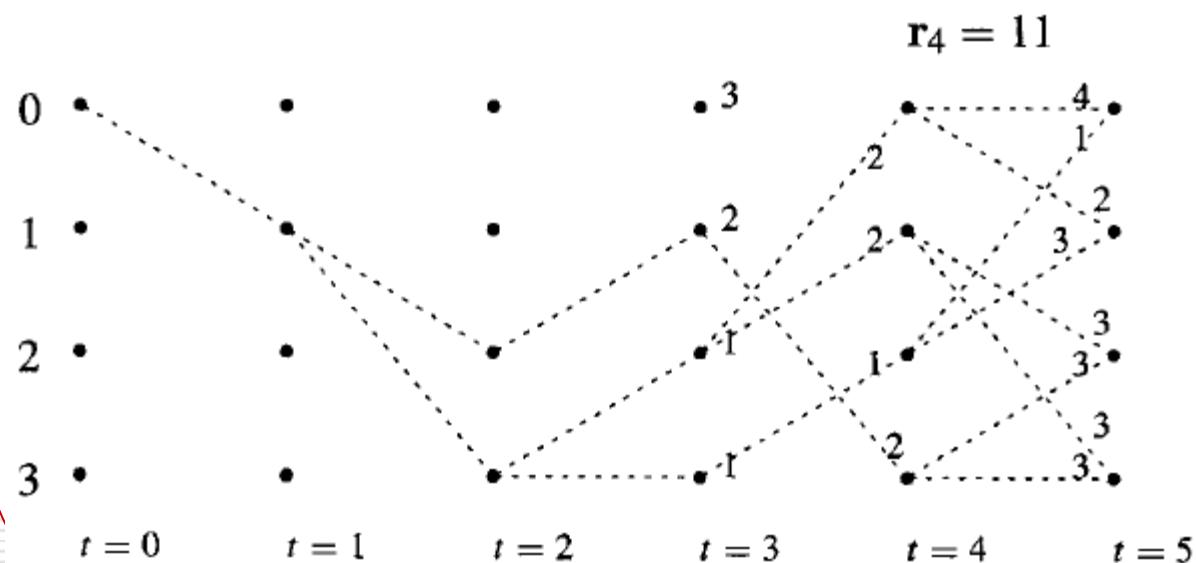
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ \underline{10} \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=5$: Mở rộng sơ đồ lưới, tính khoảng cách Hamming



Phần 3: Thuật toán giải mã Viterbi

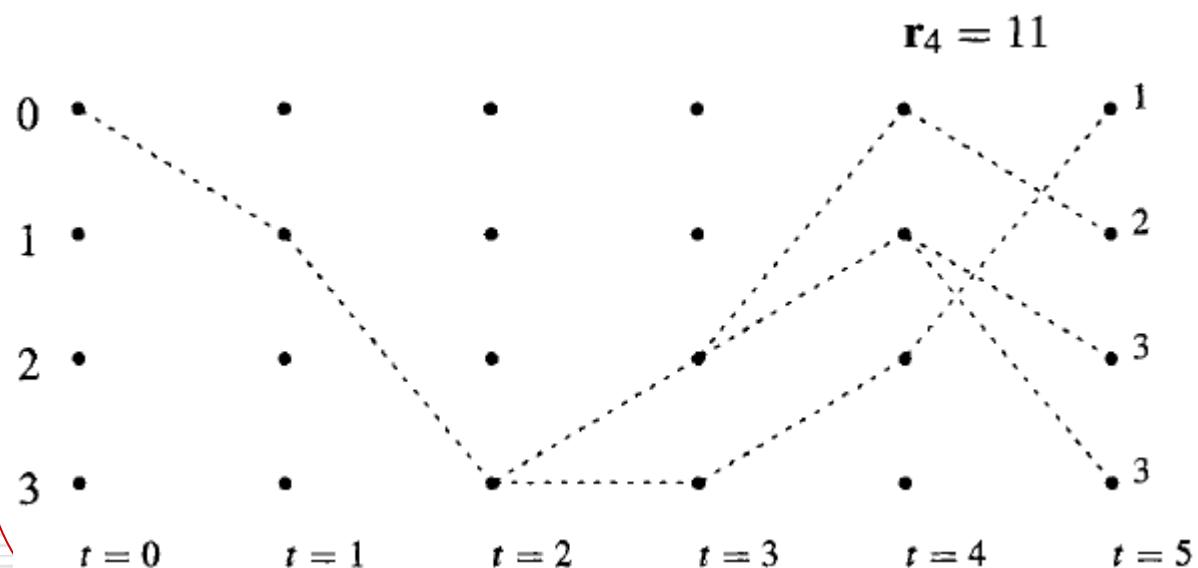
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ 10 \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=5$: Loại bỏ nhánh có khoảng cách Hamming lớn



Phần 3: Thuật toán giải mã Viterbi

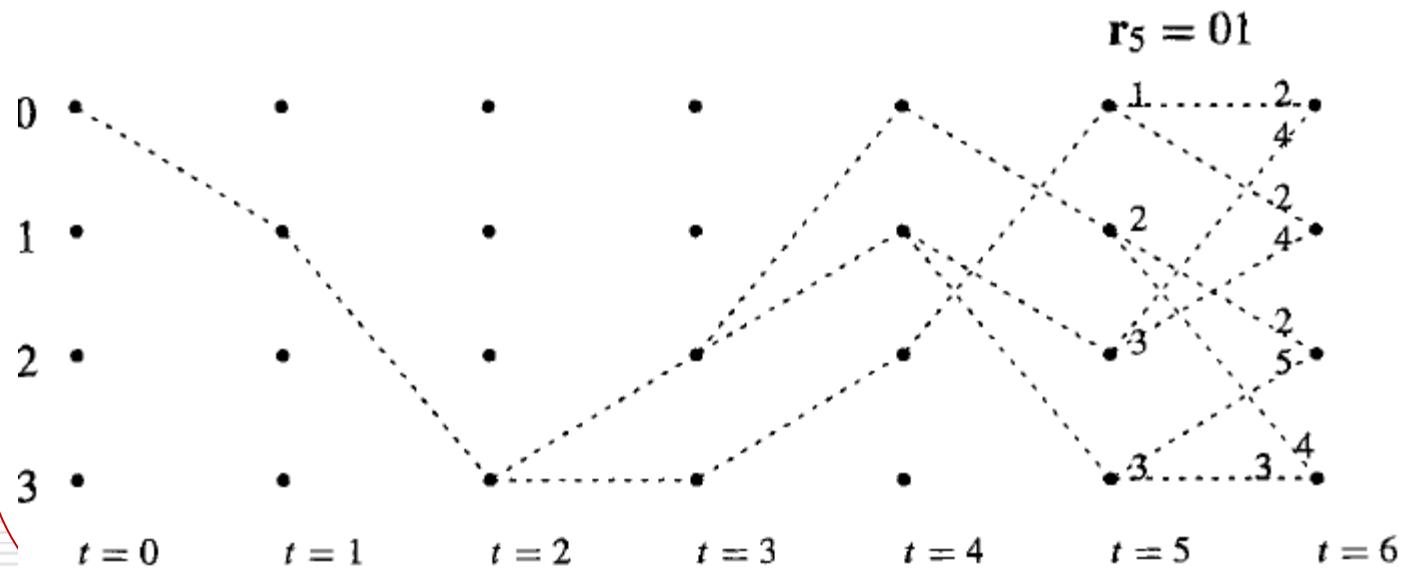
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ 10 \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=6$: Mở rộng sơ đồ lưới, tính khoảng cách Hamming



Phần 3: Thuật toán giải mã Viterbi

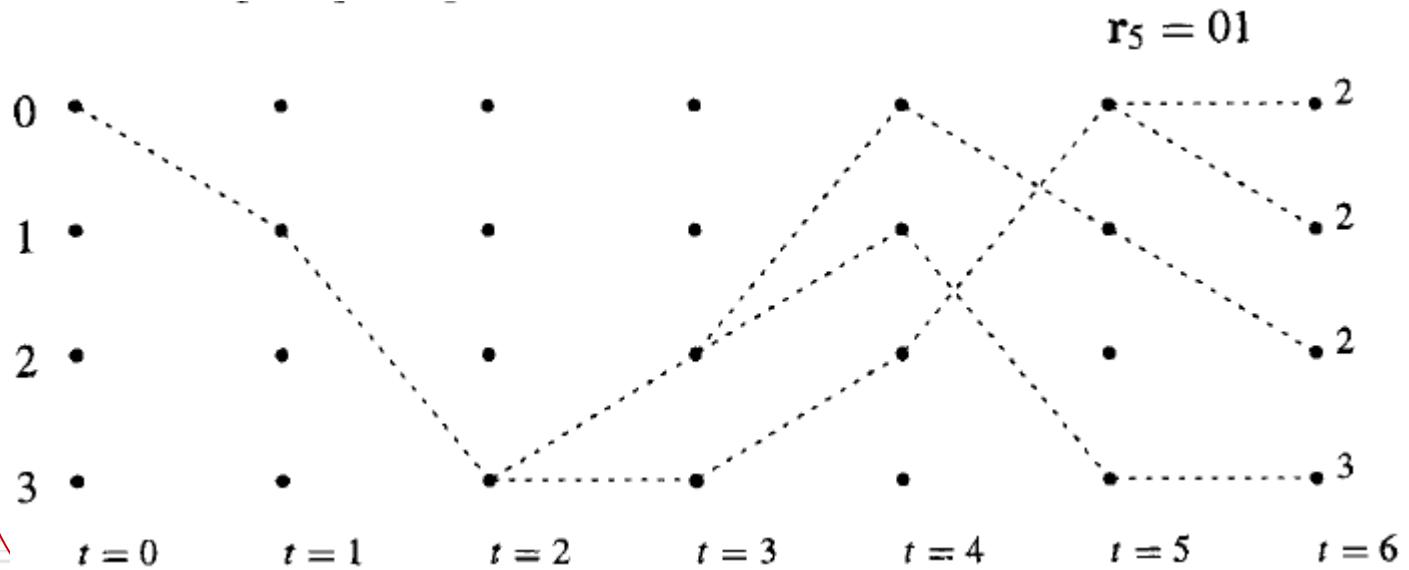
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ 10 \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=6$: Loại bỏ nhánh có khoảng cách Hamming lớn



Phần 3: Thuật toán giải mã Viterbi

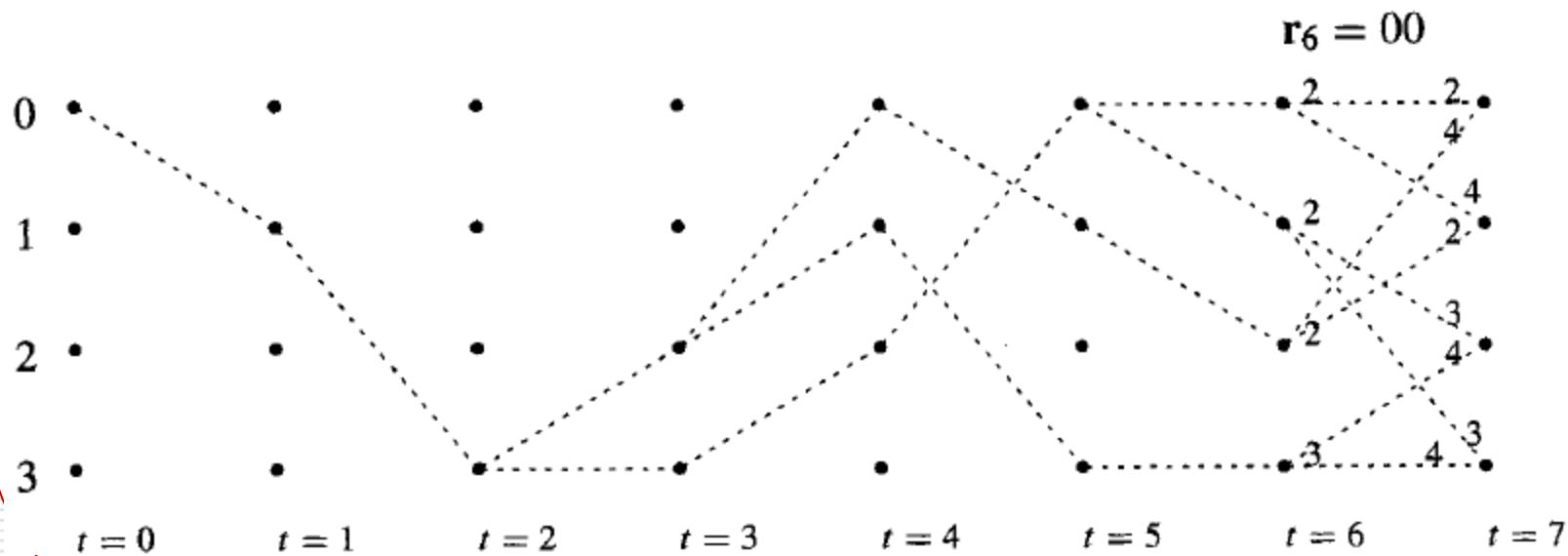
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ 10 \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=7$: Mở rộng sơ đồ lưới, tính khoảng cách Hamming



Phần 3: Thuật toán giải mã Viterbi

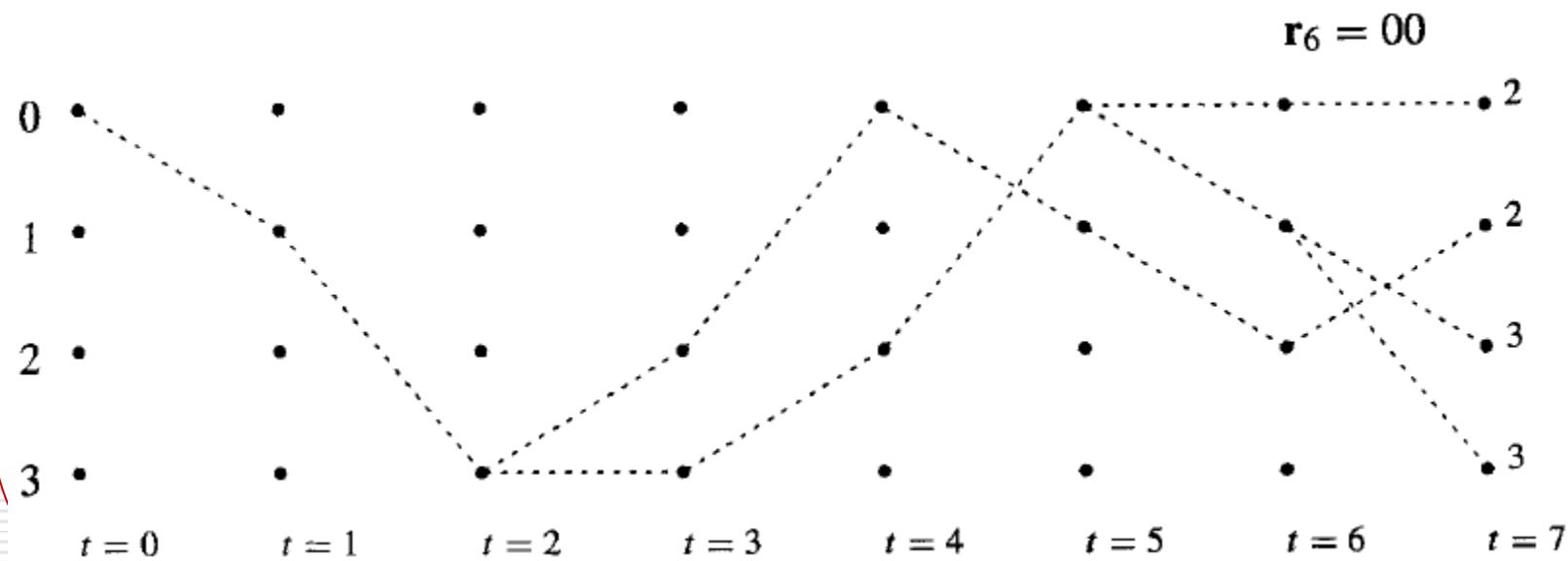
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $r = [11\ 10\ \underline{00}\ \underline{10}\ 11\ 01\ 00\ 01\ \dots] = [r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=7$: Loại bỏ nhánh có khoảng cách Hamming lớn



Phần 3: Thuật toán giải mã Viterbi

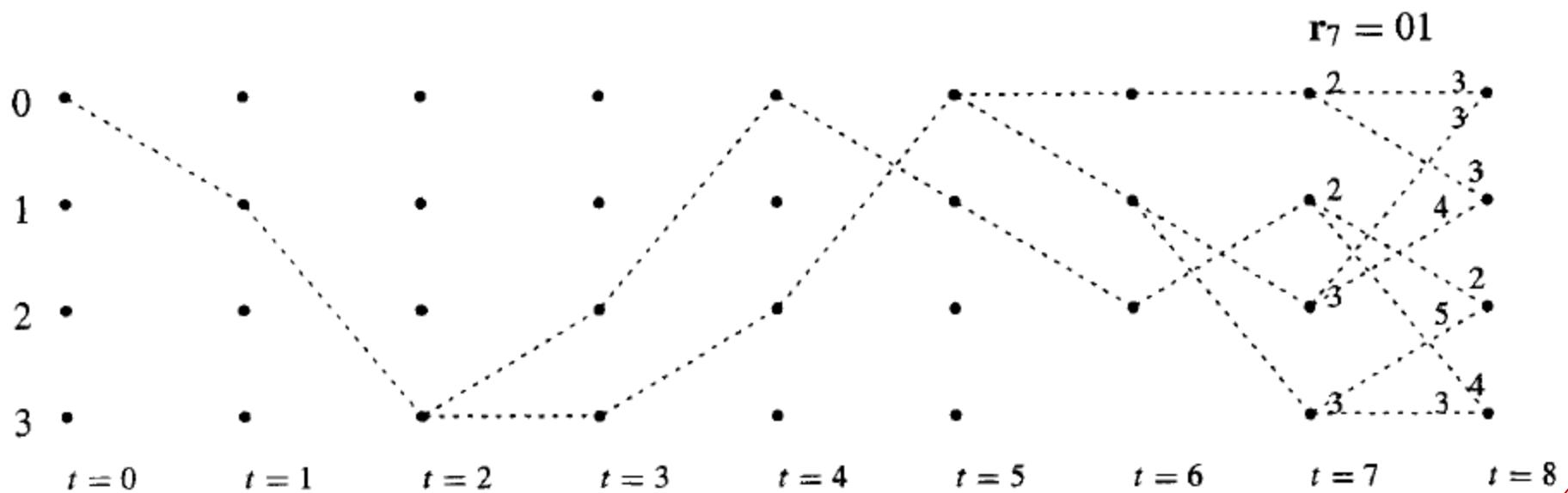
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ \underline{10} \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=8$: Mở rộng sơ đồ lưới, tính khoảng cách Hamming



Phần 3: Thuật toán giải mã Viterbi

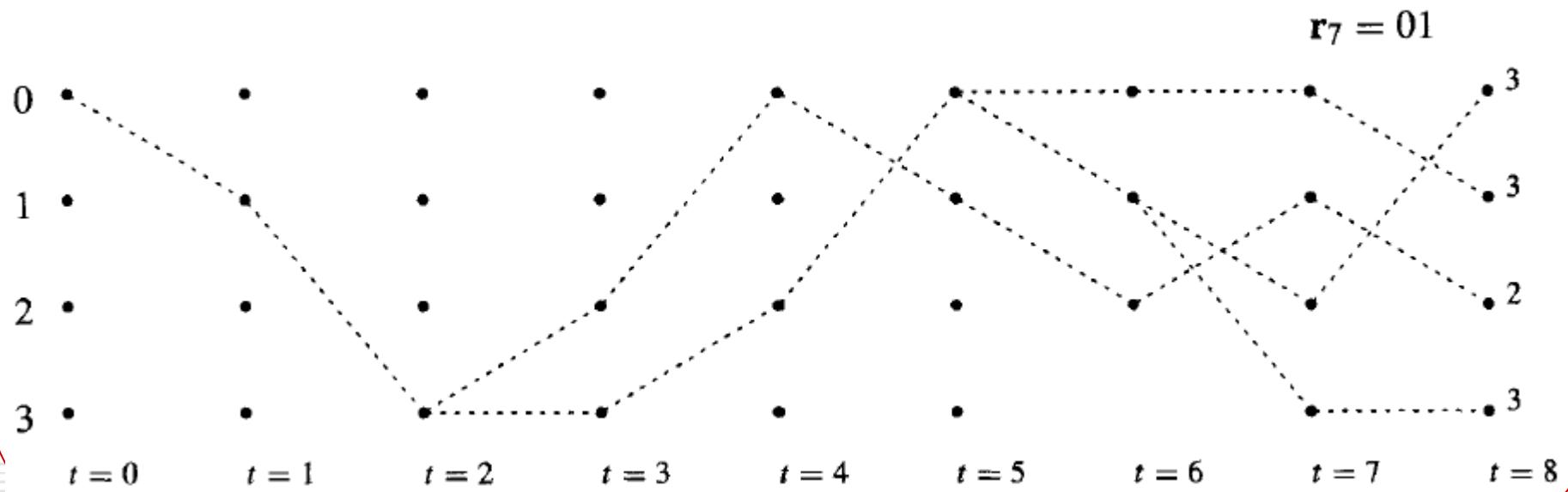
Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

Từ mã thu được $\mathbf{r} = [11 \ 10 \underline{00} \ 10 \ 11 \ 01 \ 00 \ 01 \ \dots] = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6, \mathbf{r}_7, \dots]$

Xuất phát từ trạng thái 0 (giá trị các D-FF được reset về giá trị bit ‘0’).

Tại $t=8$: Loại bỏ nhánh có khoảng cách Hamming lớn



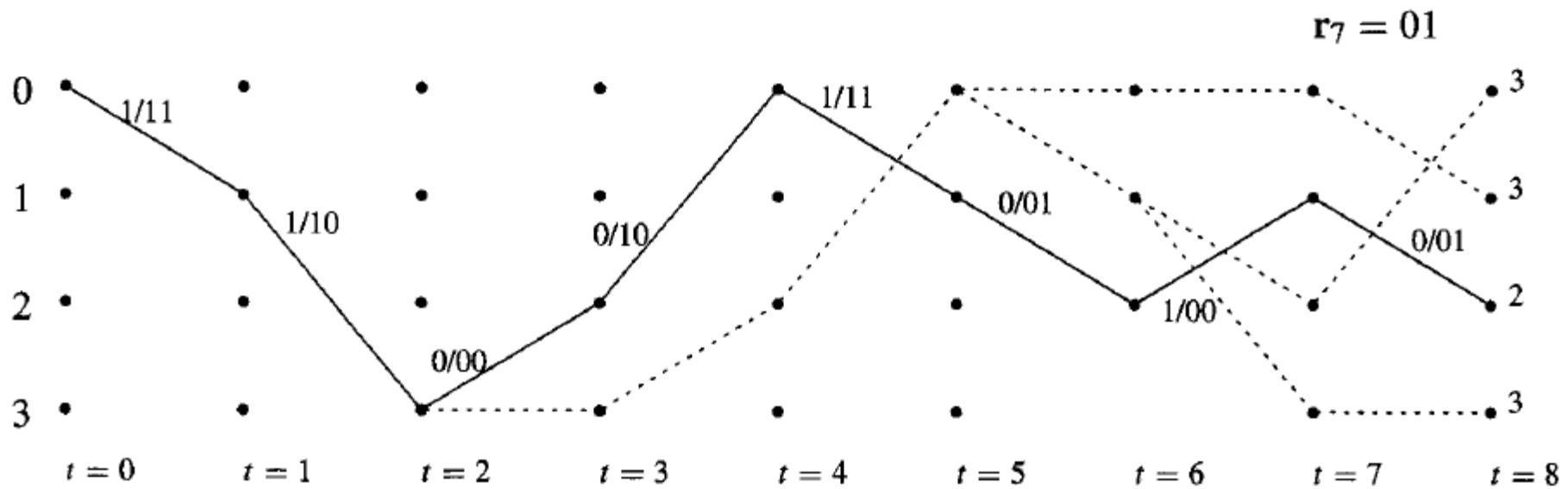
Phần 3: Thuật toán giải mã Viterbi

Ví dụ: mã tích chập $G(x) = \begin{bmatrix} 1+x^2 & 1+x+x^2 \end{bmatrix}$

Đầu vào $m=[1,1,0,0,1,0,1,0]$

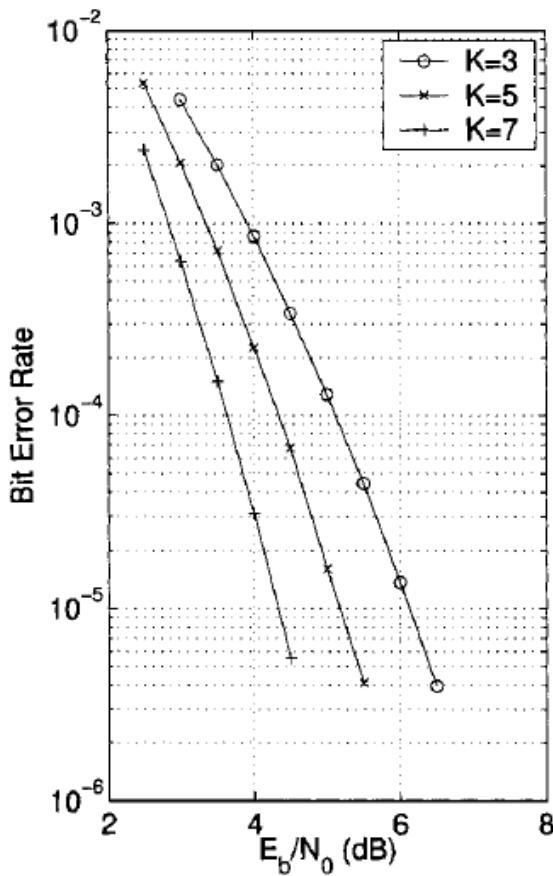
Từ mã thu được $r = [11 \ 10 \underline{00} \ \underline{10} \ 11 \ 01 \ 00 \ 01 \ \dots] = [r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7, \dots]$

- Lựa chọn tuyến đường ngắn nhất – Survival Path
- Đầu vào của tuyến đường ngắn nhất chính là thông tin cần giải mã-sửa lỗi.

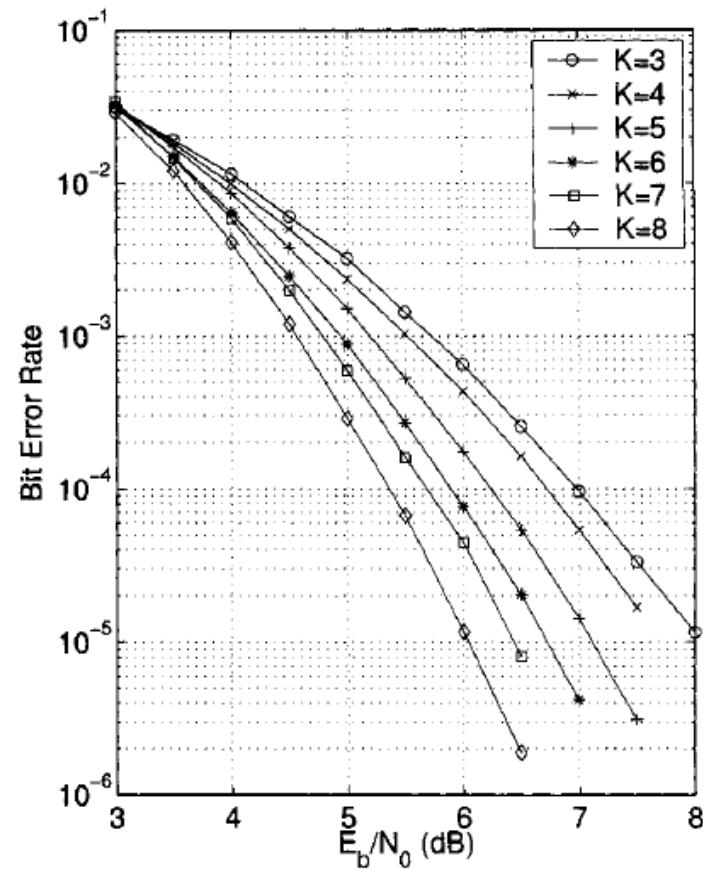


Phần 3: Thuật toán giải mã Viterbi

- ❑ So sánh khả năng sửa lỗi của mã tích chập với các độ dài ràng buộc khác nhau, và phương pháp giải mã 1-bit (Khoảng cách Hamming) và 3-bit (khoảng cách Euclid).



(a) 8-level quantization, $K = 3, 5, 7$.



(b) 1-bit (hard) quantization, $K = 3$ through 8.