

Bài :Mã Hamming

Mục lục

LỜI MỞ ĐẦU	2
• 1 tìm hiểu chung về mã hóa kênh.....	2	
1.1 Vị trí của mã hóa kênh trong thông tin số.....	2	
1.2 Khái niệm mã hóa kênh và phân loại.....	3	
1.2.1 Khái niệm.....	3	
1.2.2 Phân loại.....	3	
1.2.2.1 Mã khôi.....	4	
1.2.2.2 Mã chập.....	4	
1.3 Khả năng phát hiện lỗi và sửa lỗi của mã khôi.....	6	
• 2. Mã hamming	6	
<u>2 .2 Các mã trước thời kỳ của Hamming</u>	7	
2.2.1 Mã chẵn lẻ.....	7	
2.2.2 Mã hai-trong-năm.....	7	
2.2.3 Tái diễn dữ liệu.....	8	
2. 3 Mã Hamming.....	8	
2.4 Cách sửa lỗi mã hamming	9	
2. 4.1 ví dụ sửa lỗi mã hamming (7,11).....	10	
2.5 Mô hình của 1 mã 7 bit.....	11	
2.6. Ví dụ dùng (11,7) mã hamming.....	11	
2.7 Mã hamming(7,4).....	14	
2. 7.1 Ví dụ về cách dùng các ma trận thông qua GF(2).....	14	
2.8. Mã hamming và bit chẵn lẻ bỏ xung.....	17	
2.9. Một số giới hạn.....	17	
2. 9.1 Giới hạn trên hamming về số lượng từ mã.....	17	
2. 9.2 Giới hạn dưới về số lượng bit kiểm tra.....	17	
2.10.Ghi chú.....	18	
2. 11. Tài liệu xem thêm.....	19	

Lời mở đầu

Ngày nay việc truyền thông tin đi là rất quan trọng. Cùng với sự phát triển không ngừng của công nghệ mà nhu cầu truyền tin ngày càng được chú ý. Trong quá trình truyền dẫn sẽ không tránh khỏi lỗi và nhiễu. Do vậy việc sửa lỗi là rất cần thiết.

Từ đó đã có rất nhiều phương pháp sửa lỗi ra đời như: Mã chập, mã vòng, mã golay, mã BCH nhị phân, mã hamming.

Sau đây em xin tìm hiểu về mã hamming.

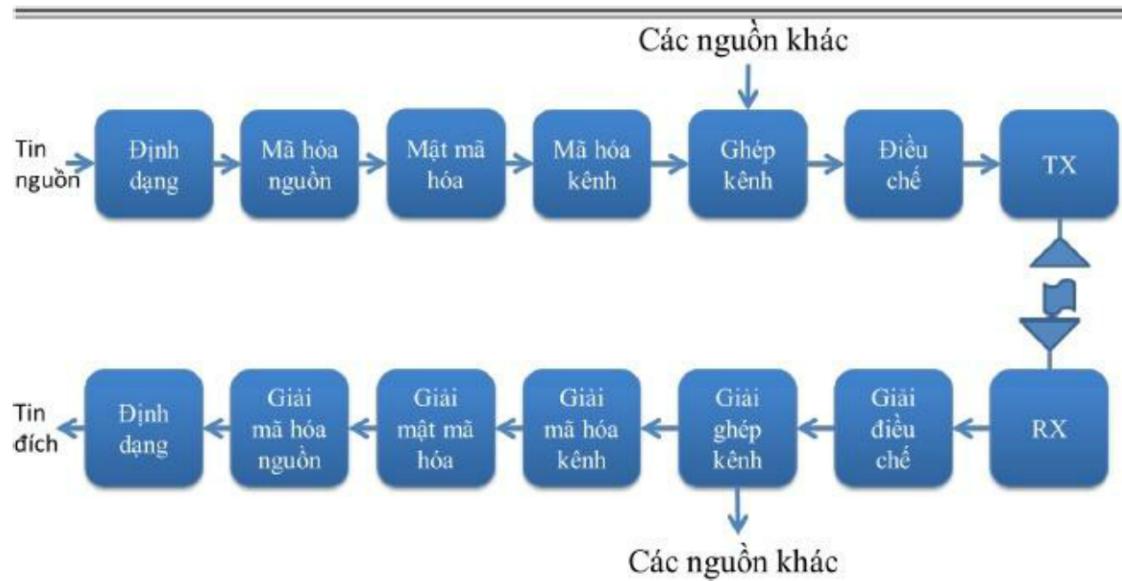
Chúng em xin chân thành cảm ơn các thầy cô đã giúp đỡ chúng em trong quá trình thực hiện bài tập này. Mặc dù đã có nhiều cố gắng nhưng trong quá trình làm đồ án, chưa có kinh nghiệm nên còn có nhiều thiếu sót. Mong các thầy cô cho em thêm ý và bổ sung thêm.

Chúng em xin chân thành cảm ơn.

1. Tìm hiểu chung về mã hóa kênh

1.1 Vị trí của mã hóa kênh trong hệ thống thông tin số

Mã hóa kênh là một khâu rất quan trọng trong hệ thống thông tin số không dây cùng với mã hóa nguồn, ghép kênh, điều chế,... để tạo ra một tín hiệu phù hợp cho việc truyền dẫn vô tuyến và tín hiệu đó có khả năng điều khiển được sự sai bit và sửa các lỗi xảy ra nếu có thể khôi phục lại gần như nguyên dạng tín hiệu tín tức mà mình truyền đi. Hình 1.1: Vị trí của mã hóa kênh truyền trong hệ thống thông tin số. Mã hóa kênh: mục đích là làm giảm xác suất sai thông tin khi truyền qua kênh truyền. Việc giảm thiểu xác suất sai dựa vào việc phát hiện sai và sửa sai có thể dẫn đến việc giảm tỉ số tín hiệu trên nhiễu (SNR) cần thiết nhờ đó giảm được công suất, tiết kiệm năng lượng. Việc sửa sai hữu hiệu cho tín hiệu SNR nhỏ sẽ thuận lợi cho việc保密, trại phổ và tăng độ chính xác của thông tin nhận - mục đích quan trọng nhất của truyền thông.



Hình 1.1: vị trí của mã hóa kênh truyền trong hệ thống thông tin số

Mã hóa kênh: mục đích là làm giảm xác suất sai thông tin khi truyền qua kênh truyền. Việc giảm thiểu xác suất sai dựa vào việc phát hiện sai và sửa sai có thể dẫn đến việc giảm tỉ số tín hiệu trên nhiễu (SNR) cần thiết nhờ đó giảm được công suất, tiết kiệm năng lượng. Việc sửa sai hữu hiệu cho tín hiệu SNR nhỏ sẽ thuận lợi cho việc保密, trại phổ và tăng độ chính xác của thông tin nhận- mục đích quan trọng nhất của truyền thông

1.2 Khái niệm mã hóa kênh và phân loại

1.2.1 Khái niệm

Mã hóa kênh là việc đưa thêm các bit dư vào tín hiệu số theo một quy luật nào đó, nhằm giúp cho bên thu có thể phát hiện và thậm chí sửa được lỗi xảy ra trên kênh truyền

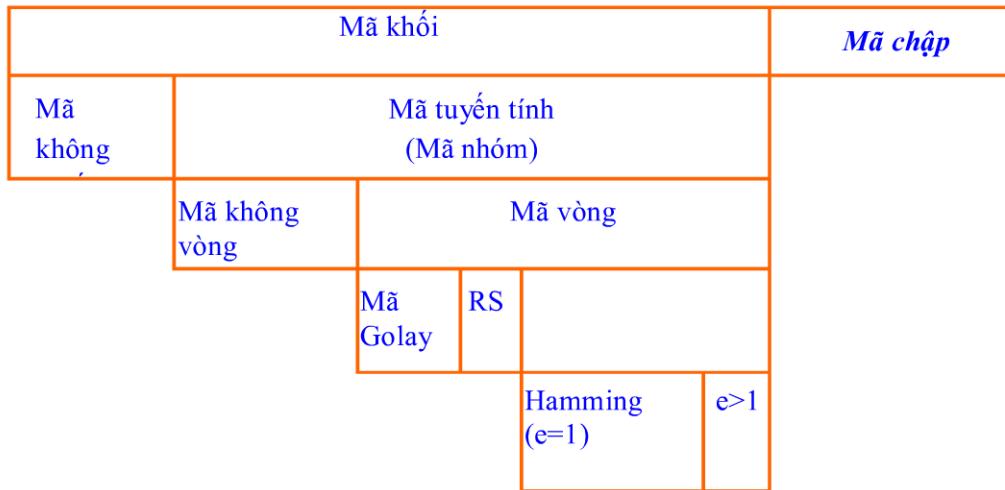
-**Mục đích :** của lý thuyết Mã hóa trên kênh truyền là tìm những mã có thể truyền thông nhanh chóng, chứa đựng nhiều từ mã tự hợp lệ và có thể sửa lỗi hoặc ít nhất phát hiện các lỗi xảy ra. Các mục đích trên không phụ thuộc vào nhau, và mỗi loại mã có công dụng tối ưu cho một ứng dụng riêng biệt. Những đặc tính mà mỗi loại mã này cần cùn tùy thuộc nhiều vào xác suất lỗi xảy ra trong quá trình truyền thông.

1.2.2. Phân loại

Có 2 loại:

1. Mã khối
2. Mã chập

Mã hóa điều khiển lỗi



Hình: phân loại mã điều khiển lỗi

1.2.2.1. Mã khối : Được đặc trưng bởi 2 số nguyên n và k và một đa trận sinh hay đa thức sinh.

Mã khối bao gồm mã khối tuyến tính, mã khối không tuyến tính, mã kiểm tra chẵn lẻ, mã vòng (mã CRC, mã hamming).

Mã khối tuyến tính: có các từ mã tương ứng 1-1 với các phân tử thuộc nhóm toán học. Mã tuyến tính có chứa từ mã gồm toàn số 0 và có tính chất đóng, chẵng hạn đối với mã tuyến tính nhị phân với 2 từ mã C_i và C_j bất kỳ ta luôn có $C_i + C_j = C_k$, C_k cũng là một từ mã. Việc có chứa từ mã gồm toàn số 0 và tính chất đóng làm cho việc tính toán đối với mã tuyến tính đặc biệt dễ.

Mã vòng: là một lớp con của mã khối tuyến tính không có từ mã gồm toàn số 0. Một mã khối tuyến tính được gọi là mã vòng needless sau một lần dịch vòng một từ mã thì cũng được một từ mã thuộc cùng bộ mã.

Mã golay: là một loại mã vòng được sửa sai nhiều lỗi, mã golay (23,12) có khả năng sửa được 3 lỗi cho từ mã dài 23 bit. Trong thực tế có hai phương pháp giải mã : phương pháp kasami và giải mã tìm kiếm có hệ thống.

Mã BCH nhị phân : là một loại mã vòng được hocquenghen tìm ra năm 1959 sau đó được Bose và chaudhuri tìm ra một cách độc lập năm 1960. Mã BCH có thể sửa được t lỗi trong n bit với $n = 2^m - 1$, $n - k \leq mt$, $d_{min} \geq 2t + 1$

Mã RS: có thể xem là mã BCH không nhị phân. Mã RS được tổ chức theo ký tự. Có khả năng sửa lỗi chùm.

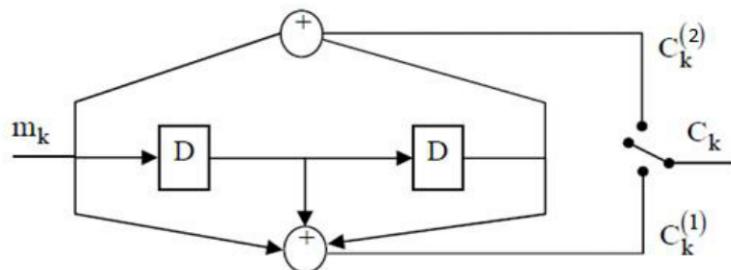
1.2.2.2. Mã chập

Mã chập cũng được đặc trưng bởi hai số nguyên n và k giống mã khối nhưng n bit ra khỏi bộ mã hóa không chỉ phụ thuộc vào k vào bit mà còn phụ thuộc vào $k-1$ bit bộ $K-1$ bit vào trước đó. K gọi là độ dài ràng buộc. Mã chập

(n, k, K) được xây dựng từ các thanh ghi dịch kK bit. Vậy xem mã chập là mã có nhớ.

Mục đích của mã hóa kênh truyền là nhằm tăng dung lượng kênh truyền, bằng cách cộng thêm vào tín hiệu những dữ liệu dư thừa được thiết kế một cách cẩn thận trước khi truyền lên kênh truyền. Mã hóa chập và mã hóa khồi là 2 dạng chính của mã hóa kênh truyền. Mã hóa chập thì dựa trên dữ liệu nối tiếp, 2 hoặc một vài bit được truyền một lúc, còn mã hóa khồi thì dựa trên một khối dữ liệu lớn tương quan (đặc trưng là khoảng vài trăm bytes). Ví dụ, mã Redsolomon là một mã hóa khồi.

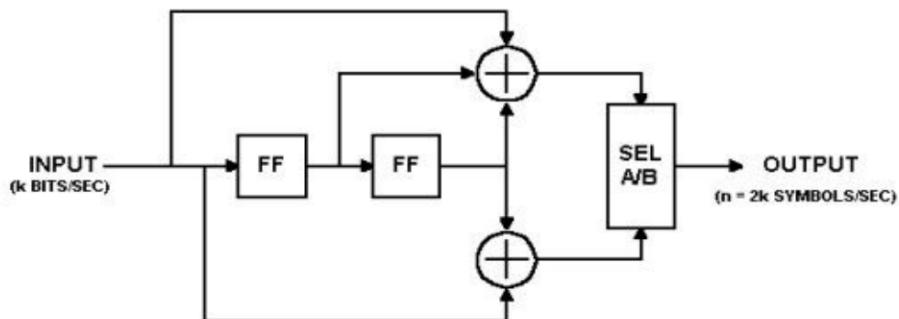
Sự khác nhau cơ bản giữa mã hóa khồi và mã hóa chập là mã hóa khồi là mã hóa không nhớ.



Sơ đồ Bộ mã hóa cho mã chập tốc độ $R = \frac{1}{2}$

Phương pháp biểu diễn mã chập: có 3 phương pháp

- + Sơ đồ lưới
- + sơ đồ trạng thái
- + sơ đồ cây



Sơ đồ Biểu diễn mã chập

1.3 . Khả năng phát hiện lỗi và sửa lỗi của mã khôi

1.3.1. Mối quan hệ giữa khoảng cách hamming với khả năng phát hiện lỗi và sửa lỗi

Công thức chỉ mối liên hệ:

$$d \geq r+s+1$$

d: là khoảng cách hamming

r: là số lỗi phát hiện được

s: là số lỗi sửa được

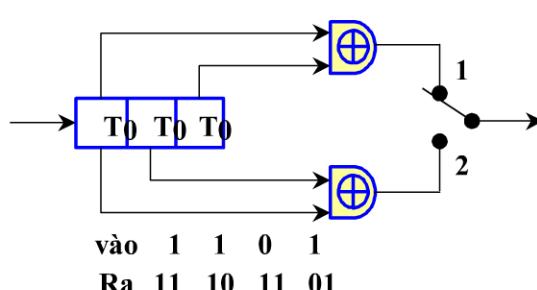
1.3.2. Mối quan hệ giữa độ dài tống cộng của từ mã và số bit tin

Thỏa mãn bất đẳng thức: $2^k \leq n+1$

$$2^k \leq n+1$$

n: là độ dài tống cộng của từ mã

k: là số bit tin trong từ mã



2. Mã hamming

2.1. Lịch sử

Trong những năm của thập niên kỷ 1940, Hamming làm việc tại [Bell Labs](#) trên máy tính [Bell Model V](#), một máy [điện cơ](#) (*electromechanical*) dùng rơ-le (*relay-based*), với tốc độ rất chậm, máy giây đồng hồ một chu kỳ máy. Nhập liệu được cho vào máy bằng những cái [thép đục lỗ](#) (*punch cards*), và hầu như máy luôn luôn gây lỗi trong khi đọc. Trong những ngày làm việc trong tuần, những mã đặc biệt được dùng để tìm ra lỗi và mỗi khi tìm được, nó nháy nháy đèn báo hiệu, báo cho người điều khiển biết để họ sửa, điều chỉnh máy lại. Trong thời gian ngoài giờ làm việc hoặc trong những ngày cuối tuần, khi người điều khiển máy không có mặt, mỗi khi có lỗi xảy ra, máy tính tự động bỏ qua chương trình đang chạy và chuyển sang công việc khác.

Hamming thường làm việc trong những ngày cuối tuần và ông càng ngày càng trở nên bực tức mỗi khi ông phải khởi động lại các chương trình ứng dụng từ đầu, do chất lượng kém, không đáng tin cậy (*unreliability*) của bộ máy đọc các thẻ đục lỗ. Năm 1950, ông đã công bố một phương pháp mà hiện nay được biết là *Mã Hamming*. Một số chương trình ứng dụng hiện thời vẫn còn sử dụng mã này của ông.

2.2. Các mã trước thời kỳ của Hamming

Nhiều mã phát hiện lỗi đơn giản đã được sử dụng trước khi có mã Hamming, nhưng không có mã nào hiệu quả bằng mã Hamming với một tổng phí tương đương.

2.2.1 Mã chẵn lẻ

Mã chẵn lẻ thêm một bit vào trong dữ liệu, và bit cho thêm này cho biết số lượng bit có giá trị **1** của đoạn dữ liệu nằm trước là một số chẵn hay một số lẻ. Nếu một bit bị thay đổi trong quá trình truyền dữ liệu, giá trị chẵn lẻ trong thông điệp sẽ thay đổi và do đó có thể phát hiện được lỗi (Chú ý rằng bit bị thay đổi có thể lại chính là bit kiểm tra). Theo quy ước chung, bit kiểm tra có giá trị **1** nếu số lượng bit có giá trị 1 trong dữ liệu là một số **lẻ**, và giá trị của bit kiểm tra **bằng 0** nếu số lượng bit có giá trị 1 trong dữ liệu là một số **chẵn**. Nói cách khác, nếu đoạn dữ liệu và bit kiểm tra được gộp lại cùng với nhau, số lượng bit có giá trị bằng 1 luôn luôn là một số chẵn.

Việc kiểm tra dùng mã chẵn lẻ là một việc không được chắc chắn cho lắm, vì nếu số bit bị thay đổi là một số chẵn (2, 4, 6 - cả hai, bốn hoặc sáu bit đều bị hoán vị) thì mã này không phát hiện được lỗi. Hơn nữa, mã chẵn lẻ không biết được bit nào là bit bị lỗi, kể cả khi nó phát hiện là có lỗi xảy ra. Toàn bộ dữ liệu đã nhận được phải bỏ đi, và phải truyền lại từ đầu. Trên một kênh truyền bị nhiễu, việc truyền nhận thành công có thể mất rất nhiều thời gian, nhiều khi còn không truyền được nữa. Mặc dù việc kiểm tra bằng mã chẵn lẻ không được tốt cho lắm, song vì nó chỉ dùng 1 bit để kiểm tra cho nên nó có số tổng phí (*overhead*) thấp nhất, đồng thời, nó cho phép phục hồi bit bị thất lạc nếu người ta biết được vị trí của bit bị thất lạc nằm ở đâu.

2.2.2 Mã hai-trong-năm

Trong những năm của thập niên kỷ 1940, Bell có sử dụng một mã hiệu phức tạp hơn một chút, gọi là mã hai-trong-năm (*two-out-of-five code*). Mã này đảm bảo mỗi một khói 5 bit (còn được gọi là *khối-5*) có chính xác hai bit có giá trị bằng 1. Máy tính có thể nhận ra là dữ liệu nhập vào có lỗi nếu trong một khói 5 bit không 2 bit có giá trị bằng 1. Tuy thế, mã hai-trong-năm cũng chỉ có thể phát hiện được một đơn vị bit mà thôi; nếu trong cùng một khói, một bit bị lộn ngược thành giá trị 1, và một bit khác bị lộn ngược thành giá trị 0, quy luật hai-trong-năm vẫn cho một giá trị đúng (*remained true*), và do đó nó không phát hiện là có lỗi xảy ra.

2.2.3 Tái diễn dữ liệu

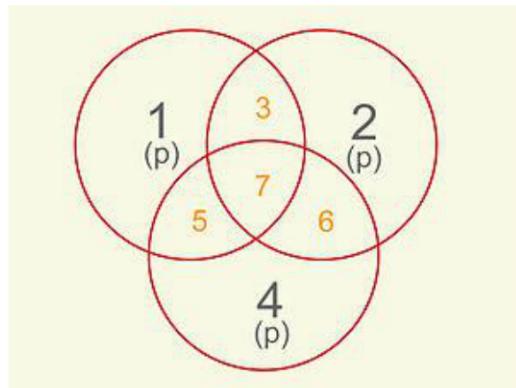
Một mã nữa được dùng trong thời gian này là mã hoạt động bằng cách nhắc đi nhắc lại bit dữ liệu vài lần (tái diễn bit được truyền) để đảm bảo bit dữ liệu được truyền, truyền đến nơi nhận trọn vẹn. Chẳng hạn, nếu bit dữ liệu cần được truyền có giá trị bằng 1, một mã tái diễn n=3 sẽ cho truyền gửi giá trị "111". Nếu ba bit nhận được không giống nhau, thì hiện trạng này báo cho ta biết rằng, lỗi trong truyền thông đã xảy ra. Nếu kênh truyền không bị nhiễu, tương đối đảm bảo, thì với hầu hết các lần truyền, trong nhóm ba bit được gửi, chỉ có một bit là bị thay đổi. Do đó các nhóm 001, 010, và 100 đều tương đương cho một bit có giá trị 0, và các nhóm 110, 101, và 011 đều tương đương cho một bit có giá trị 1 - lưu ý số lượng bit có giá trị 0 trong các nhóm được coi là có giá trị 0, là đa số so với tổng số bit trong nhóm, hay 2 trong 3 bit, tương đương như vậy, các nhóm được coi là giá trị 1 có số lượng bit bằng 1 nhiều hơn là các bit có giá trị 0 trong nhóm - chẳng khác gì việc các nhóm bit được đổi xử như là "các phiếu bầu" cho bit dữ liệu gốc vậy. Một mã có khả năng tái dựng lại thông điệp gốc trong một môi trường nhiễu lỗi được gọi là mã "sửa lỗi" (*error-correcting code*).

Tuy nhiên, những mã này không thể sửa tất cả các lỗi một cách đúng đắn hoàn toàn. Chẳng hạn chúng ta có một ví dụ sau: nếu một kênh truyền đảo ngược hai bit và do đó máy nhận thu được giá trị "001", hệ thống máy sẽ phát hiện là có lỗi xảy ra, song lại kết luận rằng bit dữ liệu gốc là bit có giá trị bằng 0. Đây là một kết luận sai lầm. Nếu chúng ta tăng số lần các bit được nhắc lại lên 4 lần, chúng ta có thể phát hiện tất cả các trường hợp khi 2 bit bị lỗi, song chúng ta không thể sửa chữa chúng được (số phiếu bầu "hòa"); với số lần nhắc lại là 5 lần, chúng ta có thể sửa chữa tất cả các trường hợp 2 bit bị lỗi, song không thể phát hiện ra các trường hợp 3 bit bị lỗi.

Nói chung, mã tái diễn là một mã hết sức không hiệu quả, giảm công suất xuống 3 lần so với trường hợp đầu tiên trong ví dụ trên của chúng ta, và công suất làm việc giảm xuống một cách nhanh chóng nếu chúng ta tăng số lần các bit được nhắc lại với mục đích để sửa nhiều lỗi hơn.

2.3. Mã Hamming

2.3.1 Khái niệm; mã Hamming là một mã sửa lỗi tuyến tính (linear error-correcting code), được đặt tên theo tên của người phát minh ra nó, Richard Hamming. Mã Hamming có thể phát hiện một bit hoặc hai bit bị lỗi (single and double-bit errors). Mã Hamming còn có thể sửa các lỗi do một bit bị sai gây ra. Ngược lại với mã của ông, mã chẵn lẻ (parity code) đơn giản vừa không có khả năng phát hiện các lỗi khi 2 bit cùng một lúc bị hoán vị (0 thành 1 và ngược lại), vừa không thể giúp để sửa được các lỗi mà nó phát hiện thấy.



2.4. Cách sửa lỗi mã hamming:

- Mã hamming là một trường hợp của mã vòng
- Mã hamming có $d = 3$, có khả năng sửa được một lỗi
- Một từ mã hamming được biểu diễn dưới dạng tổng quát $c_1c_2ic_4iic_8i\dots$ ở đây i là các bit tin và c là các bit kiểm tra.
- Các bit c là kết quả của phép đo XOR giá trị chỉ vị trí của các bit 1 với nhau. Quá trình kiểm tra lỗi bên thu diễn ra tương tự như bên phát. Nếu kết quả của phép XOR là một giá trị khác 0 thì đó chính là vị trí của bit lỗi.

2.4.1. Ví dụ xét khả năng sửa lỗi đơn của mã hamming (7,11) trong trường hợp từ mã mang tin là 1011101

Từ mã hamming có dạng c

Các bit 1 ở các vị trí 3,6,9 và 11 đổi các số này sang nhị phân

$3 \leftrightarrow 0011, 6 \leftrightarrow 0110, 9 \leftrightarrow 0111, 11 \leftrightarrow 1011$

$$\text{Tính XOR } 0011 \oplus 0110 \oplus 0111 \oplus 1011 = 1001$$

Vậy từ mã hamming phát đi là 10100110101

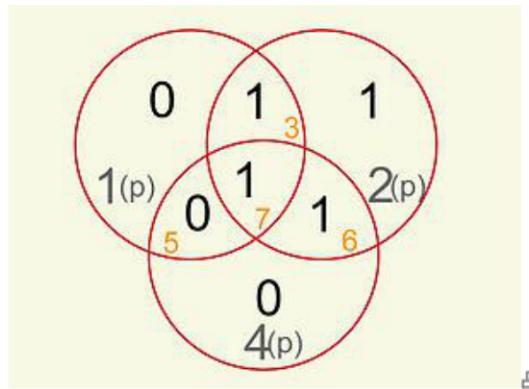
Giả sử ở bên thu thu được từ mã 10000110101. Đổi giá trị chỉ vị trí của các bit 1 sang nhị phân rồi tính XOR tương tự bên phát

$1 \leftrightarrow 0001, 6 \leftrightarrow 0110, 7 \leftrightarrow 0111, 9 \leftrightarrow 1001, 11 \leftrightarrow 1011$

$$0001 \oplus 0110 \oplus 0110 \oplus 1001 \oplus 1011 = 3$$

Từ đây phát hiện được bit lỗi là ở vị trí thứ 3. Vậy từ mã thu được sửa lại là 10100110101 giống bên phát

2.5. Mô hình của một mã 7-bit: bao gồm 4 bit dữ liệu (3,5,6,7) và 3 bit chẵn lẻ (1,2,4). Sự liên quan của các bit dữ liệu với bit chẵn lẻ được biểu hiện bằng các phần của hình tròn gối lên nhau. Bit thứ 1 kiểm tra bit thứ (3, 5, 7), trong khi bit 2 kiểm tra bit (3, 6, 7). Lưu ý, các vị trí (1,2,4 v.v.) thực ra là vị trí $2^0, 2^1, 2^2$ v.v.



Các bit dữ liệu và bit chẵn lẻ trong mối quan hệ chồng gói với nhau. Các bit chẵn lẻ được tính dùng quy luật "số chẵn". Giá trị của nhóm dữ liệu là **1100110** - các bit chẵn lẻ được in đậm, và đọc từ phải sang trái.

Càng nhiều bit sửa lỗi thêm vào trong thông điệp, và các bit ấy được bố trí theo một cách là mỗi bộ trí của nhóm các bit bị lỗi tạo nên một hình thái lỗi riêng biệt, thì chúng ta có thể xác định được những bit bị sai. Trong một thông điệp dài 7-bit, chúng ta có 7 khả năng một bit có thể bị lỗi, như vậy, chỉ cần 3 bit kiểm tra ($2^3 = 8$) là chúng ta có thể, không những chỉ xác định được là lỗi trong truyền thông có xảy ra hay không, mà còn có thể xác định được bit nào là bit bị lỗi.

Hamming nghiên cứu các kế hoạch mã hóa hiện có, bao gồm cả mã hai-trong-năm, rồi tông quát hóa khái niệm của chúng. Khoi đầu, ông xây dựng một danh mục (nomenclature) để diễn tả hệ thống máy, bao gồm cả số lượng bit dùng cho dữ liệu và các bit sửa lỗi trong một khối. Chẳng hạn, bit chẵn lẻ phải thêm 1 bit vào trong mỗi từ dữ liệu (data word). Hamming diễn tả phương pháp này là mã (8,7). Nó có nghĩa là một từ dữ liệu có tổng số bit là 8 bit, trong đó chỉ có 7 bit là các bit của dữ liệu mà thôi. Theo phương pháp suy nghĩ này, mã tái diễn (nhắc lại) ở trên phải được gọi là mã (3,1). Tỷ lệ thông tin là tỷ lệ được tính bằng việc lấy con số thứ hai chia cho con số thứ nhất. Như vậy với mã tái diễn (3,1) ở trên, tỷ lệ thông tin của nó là $\frac{1}{3}$.

Hamming còn phát hiện ra nan đề với việc đảo giá trị của hai hoặc hơn hai bit nữa, và miêu tả nó là "khoảng cách" (distance) (hiện nay nó được gọi là khoảng cách Hamming (Hamming distance) - theo cái tên của ông). Mã chẵn lẻ có khoảng cách bằng 2, vì nếu có 2 bit bị đảo ngược thì lỗi trong truyền thông trở nên vô hình, không phát hiện được. Mã tái diễn (3,1) có khoảng cách là 3, vì 3 bit, trong cùng một bộ ba, phải bị đổi ngược trước khi chúng ta được một từ mã khác. Mã tái diễn (4,1) (mỗi bit được nhắc lại 4 lần) có khoảng cách bằng 4, nên nếu 2 bit trong cùng một nhóm bị đảo ngược thì lỗi đảo ngược này sẽ đi thoát mà không bị phát hiện.

Cùng một lúc, Hamming quan tâm đến hai vấn đề; tăng khoảng cách và đồng thời tăng tỷ lệ thông tin lên, càng nhiều càng tốt. Trong những năm thuộc niên kỷ 1940, ông đã xây dựng một số kế hoạch mã hóa. Những kế hoạch này đều dựa trên những mã hiện tồn tại song được nâng cấp và tiến bộ một cách sâu sắc. Bí quyết chia khóa cho tất cả các hệ thống của ông là việc cho các bit chẵn lẻ gói lên nhau (overlap), sao cho chúng có khả năng tự kiểm tra lẫn nhau trong khi cùng kiểm tra được dữ liệu nữa.

Thuật toán cho việc sử dụng bit chẵn lẻ trong 'mã Hamming' thông thường cũng tương đối đơn giản:

1. Tất cả các bit ở vị trí là các số mũ của 2 (powers of two) được dùng làm bit chẵn lẻ. (các vị trí như 1, 2, 4, 8, 16, 32, 64 v.v. hay nói cách khác $2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6$ v.v.)
 2. Tất cả các vị trí bit khác được dùng cho dữ liệu sẽ được mã hóa. (các vị trí 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
 3. Mỗi bit chẵn lẻ tính giá trị chẵn lẻ cho một số bit trong từ mã (*code word*). Vị trí của bit chẵn lẻ quyết định chuỗi các bit mà nó luân phiên kiểm tra và bỏ qua (*skips*).
 - + Vị trí 1 ($n=1$): bỏ qua 0 bit($n-1$), kiểm 1 bit(n), bỏ qua 1 bit(n), kiểm 1 bit(n), bỏ qua 1 bit(n), v.v.
 - + Vị trí 2($n=2$): bỏ qua 1 bit($n-1$), kiểm 2 bit(n), bỏ qua 2 bit(n), kiểm 2 bit(n), bỏ qua 2 bit(n), v.v.
 - + Vị trí 4($n=4$): bỏ qua 3 bit($n-1$), kiểm 4 bit(n), bỏ qua 4 bit(n), kiểm 4 bit(n), bỏ qua 4 bit(n), v.v.
 - + Vị trí 8($n=8$): bỏ qua 7 bit($n-1$), kiểm 8 bit(n), bỏ qua 8 bit(n), kiểm 8 bit(n), bỏ qua 8 bit(n), v.v.
 - + Vị trí 16($n=16$): bỏ qua 15 bit($n-1$), kiểm 16 bit(n), bỏ qua 16 bit(n), kiểm 16 bit(n), bỏ qua 16 bit(n), v.v.
 - + Vị trí 32($n=32$): bỏ qua 31 bit($n-1$), kiểm 32 bit(n), bỏ qua 32 bit(n), kiểm 32 bit(n), bỏ qua 32 bit(n), v.v.và tiếp tục như trên.
- Nói cách khác, bit chẵn lẻ tại vị trí 2^k kiểm các bit ở các bit ở vị trí t có giá trị logic của phép toán AND giữa k và t là khác 0

2.6. Ví dụ dùng (11,7) mã Hamming

Lấy ví dụ chúng ta có một từ dữ liệu dài 7 bit với giá trị là "0110101". Để chứng minh phương pháp các mã Hamming được tính toán và được sử dụng để kiểm tra lỗi, xin xem bảng liệt kê dưới đây. Chữ **d** (data) được dùng để biểu thị các bit dữ liệu và chữ **p** (parity) để biểu thị các bit chẵn lẻ (parity bits).

Đầu tiên, các bit của dữ liệu được đặt vào vị trí tương thích của chúng, sau đó các bit chẵn lẻ cho mỗi trường hợp được tính toán dùng quy luật bit chẵn lẻ số chẵn^[1].

BÀI TẬP LỚN: MÔN TRUYỀN SÓ LIỆU

TT bit									0	1	
Vị trí bit chẵn lẻ và các bit dữ liệu	1	2	1	3	2	3	4	4	5	6	7
Nhóm dữ liệu (không có bit chẵn lẻ):											
p_1											
p_2											
p_3											
p_4											
Nhóm dữ liệu (với bit chẵn lẻ):											
Cách tính các bit chẵn lẻ trong mã Hamming (từ trái sang phải)											

Nhóm dữ liệu mới (new data word) - bao gồm các bit chẵn lẻ - bây giờ là "10001100101". Nếu chúng ta thử cho rằng bit cuối cùng bị thoái hóa (gets corrupted) và bị lật ngược từ 1 sang 0. Nhóm dữ liệu mới sẽ là "10001100100"; Dưới đây, chúng ta sẽ phân tích quy luật kiến tạo mã Hamming bằng cách cho bit chẵn lẻ giá trị 1 khi kết quả kiểm tra dùng quy luật số chẵn bị sai.

Thứ tự bit	0	1									
Vị trí bit chẵn lẻ và các bit dữ	1	2	1	3	2	3	4	4	5	6	7
Kiểm tra bit chẵn lẻ											

BÀI TẬP LỚN: MÔN TRUYỀN SỐ LIỆU

liệu

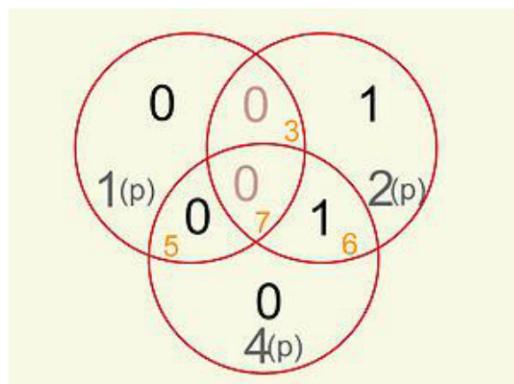
Nhóm dữ liệu nhận được:																	1	
p₁															Sai	1		
p₂															Sai	1		
p₃															Đúng	0		
p₄															Sai	1		

Kiểm tra các bit chẵn lẻ (bit bị đảo lộn có nền thăm)

Bước cuối cùng là định giá trị của các bit chẵn lẻ (nên nhớ bit nằm dưới cùng được viết về bên phải - viết ngược lại từ dưới lên trên). Giá trị số nguyên của các bit chẵn lẻ là $11_{(10)}$, và như vậy có nghĩa là bit thứ 11 trong nhóm dữ liệu (data word) - bao gồm cả các bit chẵn lẻ - là bit có giá trị không đúng, và bit này cần phải đổi ngược lại.

	p₄	p₃	p₂	p₁
Nhị phân	1	0	1	1
Thập phân	8		2	1

$\Sigma = 11$



Khi hai bit dữ liệu (3,7) có cùng bit chẵn lẻ kiểm tra tại vị trí 2^k - ví dụ (1,2) - biến đổi giá trị (lỗi trong truyền thông) thì giá trị của bit chẵn lẻ vẫn đúng như giá trị gốc (0,1)

Việc đổi ngược giá trị của bit thứ 11 làm cho nhóm

10001100100

trở lại thành

10001100101.

Bằng việc bỏ đi phần mã Hamming, chúng ta lấy được phần dữ liệu gốc với giá trị là

0110101.

Lưu ý, các bit chẵn lẻ không kiểm tra được lẫn nhau, nếu chỉ một bit chẵn lẻ bị sai thõi, trong khi tất cả các bit khác là đúng, thì chỉ có bit chẵn lẻ nói đến là sai mà thôi và không phải là các bit nó kiểm tra (not any bit it checks).

Cuối cùng, giả sử có hai bit biến đổi, tại vị trí x và y . Nếu x và y có cùng một bit tại vị trí 2^k trong đại diện nhị phân của chúng, thì bit chẵn lẻ tương ứng với vị trí đấy kiểm tra cả hai bit, và do đó sẽ giữ nguyên giá trị, không thay đổi. Song một số bit chẵn lẻ nào đấy nhất định phải bị thay đổi, vì $x \neq y$, và do đó hai bit tương ứng nào đó có giá trị x và y khác nhau. Do vậy, mã Hamming phát hiện tất cả các lỗi do hai bit bị thay đổi — song nó không phân biệt được chúng với các lỗi do 1 bit bị thay đổi.

2.7. Mã Hamming (7,4)

Hiện thời, khi nói đến **mã Hamming** chúng ta thực ra là muốn nói đến mã (7,4) mà Hamming công bố năm [1950](#). Với mỗi nhóm 4 bit dữ liệu, mã Hamming thêm 3 bit kiểm tra. [Thuật toán](#) (7,4) của Hamming có thể sửa chữa bất cứ một bit lỗi nào, và phát hiện tất cả lỗi của 1 bit, và các lỗi của 2 bit gây ra. Điều này có nghĩa là đối với tất cả các phương tiện truyền thông không có [chùm lỗi đột phát](#) (burst errors) xảy ra, mã (7,4) của Hamming rất có hiệu quả (trừ phi phương tiện truyền thông có độ nhiễu rất cao thì nó mới có thể gây cho 2 bit trong số 7 bit truyền bị đảo lộn).

2.7.1 Ví dụ về cách dùng các ma trận thông qua GF(2) [\[2\]](#)

Nguyên lý của mã Hamming bắt nguồn từ việc khai triển và mở rộng quan điểm chẵn lẻ. Việc khai triển này bắt đầu bằng việc nhân các ma trận, được gọi là **Ma trận Hamming** (Hamming matrices), với nhau. Đối với mã Hamming (7,4), chúng ta sử dụng hai ma trận có liên quan gần gũi, và đặt tên cho chúng là:

$$H_e := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

và

$$H_d := \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Các cột vectơ trong H_e là nên tảng [hạch](#) của H_d và phần trên của H_e (4 hàng đầu) là một ma trận đơn vị (identity matrix). Ma trận đơn vị cho phép

vectơ dữ liệu đi qua trong khi làm tính nhân, và như vậy, các bit dữ liệu sẽ nằm ở 4 vị trí trên cùng (sau khi nhân). Sau khi phép nhân hoàn thành, khác với cách giải thích ở phần trước (các bit chẵn lẻ nằm ở vị trí 2^k), trật tự của các bit trong từ mã (codewords) ở đây khác với cách bố trí đã nói (các bit dữ liệu nằm ở trên, các bit kiểm chẵn lẻ nằm ở dưới).

Chúng ta dùng một nhóm 4 bit dữ liệu (số 4 trong cái tên của mã là vì vậy) chủ chốt, và cộng thêm vào đó 3 bit dữ liệu thừa (vì $4+3=7$ nên mới có số 7 trong cái tên của mã). Để truyền gửi dữ liệu, chúng ta hãy nhóm các bit dữ liệu mà mình muốn gửi thành một vectơ. Lấy ví dụ, nếu dữ liệu là "1011" thì vectơ của nó là:

$$\mathbf{p} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Giả sử, chúng ta muốn truyền gửi dữ liệu trên. Chúng ta tìm tích của H_e và \mathbf{p} , với các giá trị modulo 2 [3]:

$$H_e \mathbf{p} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \mathbf{r}$$

Máy thu sẽ nhận H_d với \mathbf{r} , để kiểm tra xem có lỗi xảy ra hay không. Thi hành tính nhân này, máy thu được (một lần nữa, các giá trị đồng dư modulo 2):

$$H_d \mathbf{r} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Vì chúng ta được một vectơ toàn số không cho nên máy thu có thể kết luận là không có lỗi xảy ra.

Sở dĩ một vectơ toàn số không có nghĩa là không có lỗi, bởi vì khi H_e được nhân với vectơ dữ liệu, một sự thay đổi trong nền tảng xảy ra đối với không gian bên trong vectơ (vector subspace), tức là hạch của H_d . Nếu không có vấn đề gì xảy ra trong khi truyền thông, \mathbf{r} sẽ nằm nguyên trong hạch của H_d và phép nhân sẽ cho kết quả một vectơ toàn số không.

Trong một trường hợp khác, nếu chúng ta giả sử là lỗi một bit đã xảy ra. Trong toán học, chúng ta có thể viết:

$$\mathbf{r} + \mathbf{e}_i$$

môđulô 2, trong đó \mathbf{e}_i là vecto đơn vị đứng thứ i (ith unit vector), có nghĩa là, một vecto số 0 có một giá trị 1 trong vị trí i (tính từ 1 tính đi). Biểu thức trên nói cho chúng ta biết rằng có một bit bị lỗi tại vị trí i.

Nếu bây giờ chúng ta nhân H_d với cả hai vecto này:

$$H_d(\mathbf{r} + \mathbf{e}_i) = H_d\mathbf{r} + H_d\mathbf{e}_i$$

Vì \mathbf{r} là dữ liệu thu nhận được không có lỗi, cho nên tích của H_d và \mathbf{r} bằng 0. Do đó

$$H_d\mathbf{r} + H_d\mathbf{e}_i = \mathbf{0} + H_d\mathbf{e}_i = H_d\mathbf{e}_i$$

Vậy, tích của H_d với vecto nền chuẩn tại cột thứ i (the ith standard basis vector) làm lộ ra cột ở trong H_d , vì thế mà chúng ta biết rằng lỗi đã xảy ra tại vị trí cột này trong H_d . Vì chúng ta đã kiến tạo H_d dưới một hình thức nhất định, cho nên chúng ta có thể hiểu giá trị của cột này như một số nhị phân - ví dụ, (1,0,1) là một cột trong H_d , tương đồng giá trị với cột thứ 5, do đó chúng ta biết lỗi xảy ra ở đâu và có thể sửa được nó.

Lấy ví dụ, giả sử chúng ta có:

$$\mathbf{s} = \mathbf{r} + \mathbf{e}_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Nếu thi hành phép nhân:

$$H_d\mathbf{s} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Tích của phép nhân cho chúng ta một kết quả tương đương với cột thứ 2 ("010" tương đương với giá trị 2 trong số thập phân), và do đó, chúng ta biết rằng lỗi đã xảy ra ở vị trí thứ 2 trong hàng dữ liệu, và vì vậy có thể sửa được lỗi.

Chúng ta có thể dễ dàng thấy rằng, việc sửa lỗi do 1 bit bị đảo lộn gây ra, dùng phương pháp trên là một việc thực hiện được. Bên cạnh đó, mã Hamming còn có thể phát hiện lỗi do 1 bit hoặc 2 bit bị đảo lộn gây ra, dùng tích của H_d khi tích này không cho một vecto số không. Tuy thế, song mã Hamming không thể hoàn thành cả hai việc.

2.8. Mã Hamming và bit chẵn lẻ bổ sung

Nếu chúng ta bổ sung thêm một bit vào mã Hamming, thì mã này có thể dùng để phát hiện những lỗi gây ra do 2 bit bị lỗi, và đồng thời nó không cần trả việc sửa các lỗi do một bit gây ra. Nếu không bổ sung một bit vào thêm, thì mã này có thể phát hiện các lỗi do một bit, hai bit, ba bit gây ra, song nó sẽ cần trả việc sửa các lỗi do một bit bị đảo lộn. Bit bổ sung là bit được áp dụng cho tất cả các bit sau khi tất cả các bit kiểm của mã Hamming đã được thêm vào.

Khi sử dụng tính sửa lỗi của mã, nếu lỗi ở một bit chẵn lẻ bị phát hiện và mã Hamming báo hiệu là có lỗi xảy ra thì chúng ta có thể sửa lỗi này, song nếu chúng ta không phát hiện được lỗi trong bit chẵn lẻ, nhưng mã Hamming báo hiệu là có lỗi xảy ra, thì chúng ta có thể cho rằng lỗi này là do 2 bit bị đổi cùng một lúc. Tuy chúng ta phát hiện được nó, nhưng không thể sửa lỗi được.

2.9. Một số giới hạn

2.9.1 .Giới hạn trên hamming về số lượng từ mã

- Định lý:Nếu bộ mã (W_1, \dots, W_m) gồm các từ mã nhị phân chiều dài n có thể sửa sai các lỗi sai $\leq t$ bit, thì số lượng từ mã M phải thỏa mãn bất đẳng thức

$$M \leq \frac{2^n}{\sum_{i=0}^t \binom{n}{i}}$$

- Chứng minh; đối với mỗi từ mã W_i định nghĩa một tập hợp A_i bao gồm tất cả các dãy V_j có khoảng cách hamming đối với W_i nhỏ hơn hay bằng t .Tổng số dãy trong A_i được cho bằng

$$\sum_{i=0}^t \binom{n}{i}$$

- Để có thể sửa sai được các lỗi sai $\leq t$ bit thì tập hợp A_1, \dots, A_m phải rời nhau. Mà tổng số phần tử của tất cả các tập A_i phải nhỏ hơn hoặc bằng tổng số dãy có chiều dài n vì vậy

$$M \sum_{i=0}^t \binom{n}{i} \leq 2^n$$

2.9.2 Giới hạn dưới về số lượng bit kiểm tra

- Định lý: Nếu 1 mã truyền tính $C(n,k)$ có thể sửa sai các lỗi sai $\leq t$ bit, thì các bit kiểm tra $r = n - k$ phải thỏa mãn bất đẳng thức sau :

$$2^r \geq \sum_{i=0}^n \binom{n}{i}$$

BÀI TẬP LỚN: MÔN TRUYỀN SÓ LIỆU

- Chứng minh; Đối với mỗi vector e chúng ta có một syndrome tương ứng.
Vậy để sửa sai tất cả các lỗi sai $\leq t$ bit thì chúng ta có $t \leq n$

$$\sum_{i=0}^{t \leq n} (\)$$

vector lỗi vì vậy có $\sum_{i=0}^{t \leq n} (\)$ vector sửa sai

- Mà tổng vector sửa sai bằng $2^r n-k = 2^r r$. Thật vậy vì các vector thuộc cùng một tập coset thì có cùng 1 vector sửa sai. Do đó số lượng vector sửa sai bằng số lượng tập coset tức bằng $2^r n-k$, vì vậy chúng phải có

$$2^r r \geq \sum_{i=0}^{t \leq n} (\)$$

- Định lý này là điều kiện cần và đủ. Chẳng hạn với $n=10$, $t=2$ thì $r \geq 6$ tuy nhiên chúng ta có thể kiểm tra lại rằng để sửa được lỗi sai ≤ 2 thì phải ít nhất $r \geq 7$

2.10. ghi chú

7 bit dữ liệu	byte có bit chẵn lẻ	
	Quy luật số chẵn	Quy luật số lẻ
0000000	00000000	00000001
1010001	10100011	10100010
1101001	11010010	11010011
1111111	11111111	11111110

- Cách tính bit chẵn lẻ trong nhóm các bit dữ liệu, dùng quy luật **số chẵn** như sau: nếu số lượng bit có giá trị bằng 1 (the bit is set) là một số lẻ, thì bit chẵn lẻ bằng $1_{(2)}$ (và do việc cộng thêm một bit có giá trị $1_{(2)}$ này vào dữ liệu, tổng số bit có giá trị $1_{(2)}$ sẽ là một **số chẵn** - bao gồm cả bit chẵn lẻ, còn không thì bit chẵn lẻ sẽ có giá trị $0_{(2)}$). Ngược lại, bit chẵn lẻ dùng quy luật **số lẻ** sẽ có giá trị $1_{(2)}$, nếu số lượng các bit có giá trị bằng $1_{(2)}$ là một số chẵn - do việc thêm bit chẵn lẻ có giá trị bằng $1_{(2)}$ vào nhóm dữ liệu, tổng số bit có giá trị $1_{(2)}$ là một **số lẻ** - và bằng 0 nếu ngược lại.

- GF (nguyên [tiếng Anh](#): Galois field - hay gọi finite field), tạm dịch là "Trường Hữu Hạn". Xin xem thêm bài [tiếng Anh](#) [Finite field](#).

- môđulô (nguyên [tiếng Anh](#): Modulo) là phép tính số dư trong tính chia. Ví dụ $100/3 = 1$ (được 33 dư 1). Trong toán học, nếu có hai số nguyên a và b ,

cùng một số dư n nào đó, thì biểu thức $a \equiv b \pmod{n}$ - nói là a và b có đồng dư modulo n - có nghĩa là a và b có cùng số dư khi được chia cho n, hay nói một cách tương tự, $a-b$ là một bội số (*multiple*) của n.

2.11. xem thêm tài liệu

- [Khoảng cách Hamming](#) (*Hamming distance*)
- [Mã Golay](#) (*Golay code*)
- [Mã Reed-Muller](#) (*Reed-Muller code*)
- [Mã Reed-Solomon](#) (*Reed-Solomon code*)
- [Mã Turbo](#) (*Turbo code*)
- [Mã CRC](#)

[1] John Proakis, Digital Communications
(Chapter 8-Block and Convolutional Channel Codes), McGraw-Hill Science/
Engineering/ Math, 4th, 2000.

[2] Fu Hua Huang, Evaluation of Soft Output Decoding for
TurboCodes(chapter 2_convolution codes), Master's Thesis, 1997.

[3] Mr. Chip Fleming, [Tutorial on Convolutional Coding with Viterbi Decoding](#), Spectrum Applications, 2006

[4] Wei Chen, RTL implementation of Viterbi decoder, Master's thesis performed
in Computer Engineering, 2006

6] Một số trang web mà nhóm có tham khảo:

- <http://www.altera.com>
- <http://www.fotech.org>
- <http://www.ngohaibac.net>
- <http://www.mathwork.com>
- <http://en.wikipedia.org>
- <http://www.dsplog.com>
- <http://home.netcom.com>
- <http://gaussianwaves.blogspot.com>