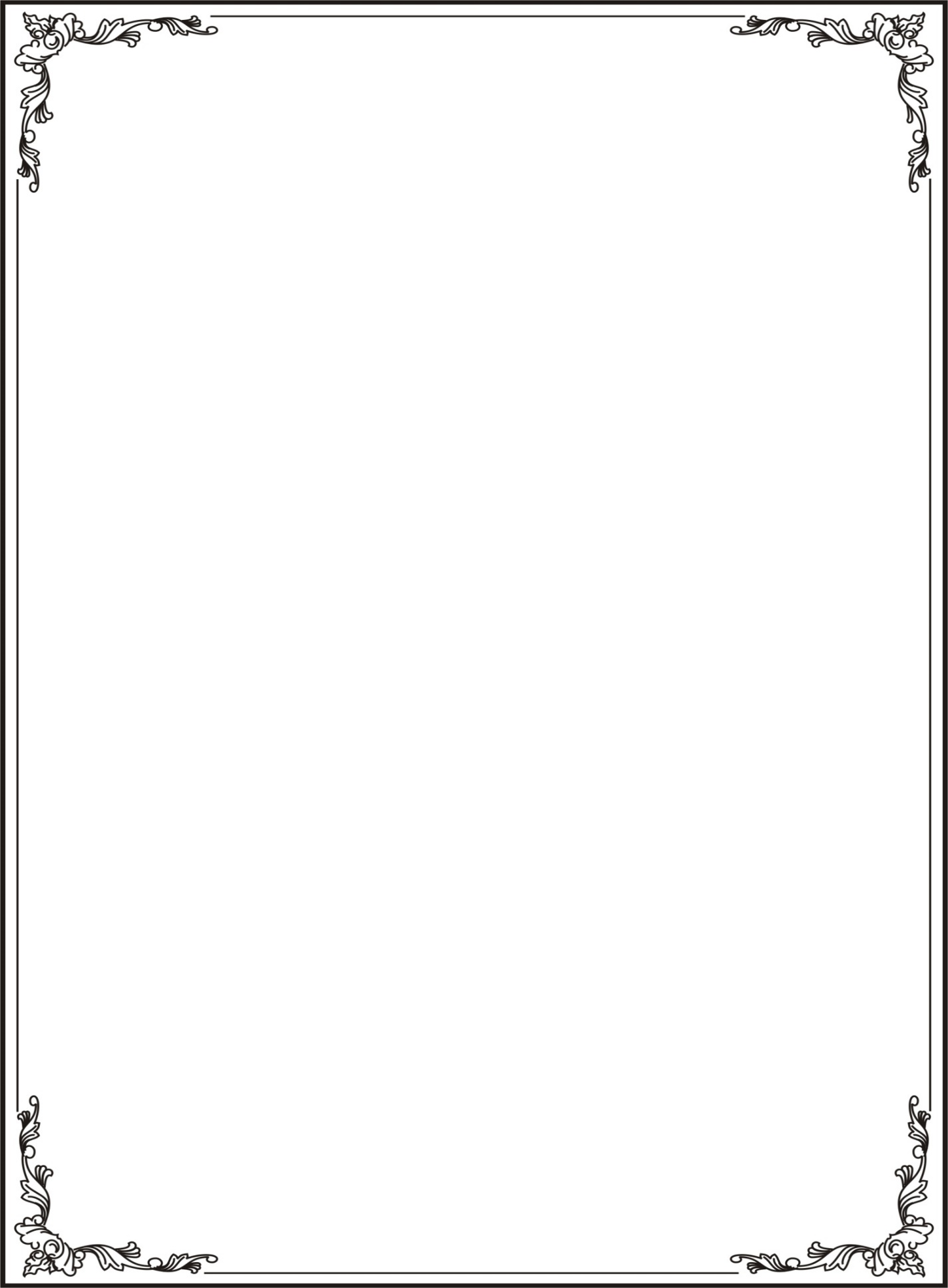
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**

A red circle with a yellow star in the middle

Description automatically generated

**BÁO CÁO BÀI TẬP LỚN**

**MÔN HỌC: ĐIỆN TỬ SỐ**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | **: Cô Trần Thị Thúy Hà** |
| **Lớp**  **Nhóm**  **Tổ thực hiện** | **: D23CQCE04-B**  **: 04**  **: 09** |
| **Họ và tên thành viên** | **: Bùi Nguyễn Thái An B23DCCE001**  **Đinh Thành Công**  **B23DCCE013** |

***Hà Nội – 2025***

MỤC LỤC

[Lời mở đầu 3](#_Toc197897856)

[I. Giới thiệu về mã Hamming và các loại lỗi thường gặp 4](#_Toc197897857)

[1. Lỗi trong truyền dữ liệu là gì? 4](#_Toc197897858)

[2. Các loại lỗi thường gặp 4](#_Toc197897859)

[3. Phát hiện lỗi và sửa lỗi là gì? 5](#_Toc197897860)

[4. Các phương pháp phát hiện lỗi phổ biến 5](#_Toc197897861)

[5. Mã Hamming: Khái niệm, nguyên lý, lịch sử, ứng dụng và ưu, nhược điểm 6](#_Toc197897862)

[II. Xây dựng nguyên lí 8](#_Toc197897863)

[1. Mạch tạo mã Hamming 8](#_Toc197897864)

[2. Mạch giải mã Hamming 12](#_Toc197897865)

[III. Giới thiệu linh kiện 14](#_Toc197897866)

[IV. Sơ đồ khối và mạch cụ thể 17](#_Toc197897867)

[1. Sơ đồ khối 17](#_Toc197897868)

[2. Nguyên lí hoạt động : 18](#_Toc197897869)

[Lời kết thúc 22](#_Toc197897870)

# Lời mở đầu

Trong thời đại công nghệ số phát triển mạnh mẽ, việc trao đổi thông tin qua các hệ thống mạng và thiết bị điện tử đã trở thành một phần không thể thiếu trong cuộc sống. Tuy nhiên, quá trình truyền dữ liệu không phải lúc nào cũng diễn ra suôn sẻ. Các yếu tố như nhiễu điện từ, lỗi phần cứng hay môi trường truyền dẫn không ổn định có thể khiến dữ liệu bị sai lệch, làm ảnh hưởng nghiêm trọng đến chất lượng truyền tải và gây mất mát thông tin. Điều này đòi hỏi chúng ta phải có những phương pháp đảm bảo dữ liệu được truyền tải chính xác và an toàn.

Trong số các giải pháp khắc phục lỗi dữ liệu, mã Hamming nổi bật lên như một phương pháp hiệu quả và đáng tin cậy. Được phát minh bởi nhà khoa học Richard Hamming vào năm 1950, mã Hamming không chỉ có khả năng phát hiện mà còn tự động sửa lỗi bit đơn mà không cần truyền lại toàn bộ dữ liệu. Với nguyên lý hoạt động dựa trên việc chèn các bit kiểm tra vào chuỗi dữ liệu, mã Hamming đã trở thành một trong những giải pháp mã hóa phổ biến nhất trong lĩnh vực truyền thông và lưu trữ dữ liệu.

Nhóm chúng em đã lựa chọn đề tài này với mong muốn hiểu rõ hơn về cách thức hoạt động của mã Hamming, từ nguyên lý mã hóa, giải mã cho đến cách xây dựng mạch phần cứng thực tế. Thông qua bài tập lớn này, chúng em sẽ trình bày về các loại lỗi dữ liệu thường gặp, nguyên lý hoạt động của mã Hamming, cách xây dựng mạch mã Hamming cũng như những ứng dụng thực tiễn của nó. Hy vọng rằng, thông qua bài này, nhóm chúng em có thể mang đến một cái nhìn tổng quan và sâu sắc hơn về một phương pháp mã hóa mang tính nền tảng trong lĩnh vực kỹ thuật số.

# I. Giới thiệu về mã Hamming và các loại lỗi thường gặp

## 1. Lỗi trong truyền dữ liệu là gì?

Khi dữ liệu được truyền từ thiết bị gửi đến thiết bị nhận, nó có thể bị hỏng do các yếu tố như nhiễu điện từ, lỗi phần cứng, môi trường truyền dẫn không ổn định hoặc lỗi phần mềm.

Khi xảy ra lỗi, dữ liệu nhận được không còn giống với dữ liệu ban đầu. Sự sai lệch này được gọi là “lỗi truyền dữ liệu”.

Trong các hệ thống kỹ thuật số, dữ liệu được truyền dưới dạng chuỗi các bit (0 và 1). Nếu một bit trong chuỗi bị thay đổi từ 1 thành 0 hoặc ngược lại, thì đó được gọi là “lỗi bit”. Dù chỉ là một thay đổi nhỏ nhưng có thể làm ảnh hưởng nghiêm trọng đến hoạt động của toàn bộ hệ thống.

## 2. Các loại lỗi thường gặp

**a. Lỗi bit đơn (Single-bit error) :**

* Là lỗi xảy ra khi chỉ một bit trong chuỗi dữ liệu bị thay đổi.
* Ví dụ: chuỗi dữ liệu ban đầu là 1011001, nhưng sau khi truyền đi lại nhận được 1010001.
* Lỗi bit đơn thường xảy ra trong hệ thống truyền dữ liệu song song, nơi mỗi bit đi qua một kênh riêng và dễ bị ảnh hưởng bởi nhiễu điện.

**b. Lỗi nhiều bit (Multiple-bit error) :**

* Là lỗi xảy ra khi có hai hoặc nhiều bit bị thay đổi trong chuỗi dữ liệu.
* Loại lỗi này có thể do sự cố lớn hơn trong hệ thống và thường khó phát hiện hoặc sửa hơn so với lỗi bit đơn.
* Nó có thể xuất hiện trong cả truyền nối tiếp lẫn truyền song song.

**c. Lỗi bùng nổ (Burst error) :**

* Là lỗi xảy ra khi một chuỗi các bit liên tiếp trong dữ liệu bị thay đổi.
* Độ dài lỗi bùng nổ được tính từ bit đầu tiên bị lỗi đến bit cuối cùng bị lỗi.
* Loại lỗi này thường xảy ra trong hệ thống truyền nối tiếp, do nhiễu kéo dài trong một khoảng thời gian ngắn.

## 3. Phát hiện lỗi và sửa lỗi là gì?

Để đảm bảo dữ liệu truyền đi được chính xác và không bị mất mát, người ta sử dụng các kỹ thuật phát hiện và sửa lỗi.

* **Phát hiện lỗi** là quá trình xác định xem có lỗi xảy ra trong dữ liệu hay không.
* **Sửa lỗi** là quá trình xác định vị trí bit bị lỗi và khôi phục lại dữ liệu ban đầu.

Việc phát hiện và sửa lỗi được thực hiện bằng cách **thêm các bit kiểm tra (redundant bits)** vào dữ liệu gốc, nhằm tạo ra một mã có khả năng tự kiểm tra.

## 4. Các phương pháp phát hiện lỗi phổ biến

**a. Kiểm tra chẵn lẻ (Parity Check) :**

* Một bit kiểm tra (bit chẵn lẻ) được thêm vào mỗi đơn vị dữ liệu.
* Có hai loại:
  + Kiểm tra chẵn (even parity): tổng số bit 1 phải là số chẵn.
  + Kiểm tra lẻ (odd parity): tổng số bit 1 là số lẻ.
* Ưu điểm: đơn giản, dễ thực hiện.
* Nhược điểm: chỉ phát hiện được lỗi bit đơn, không sửa lỗi.

**b. Kiểm tra dự phòng theo chiều dọc (LRC - Longitudinal Redundancy Check) :**

* Dữ liệu được chia thành nhiều dòng và sắp xếp thành bảng.
* Tính bit chẵn lẻ theo từng cột.
* Các bit chẵn lẻ cột được gửi kèm theo dữ liệu để kiểm tra lỗi.
* Kết hợp với kiểm tra chẵn lẻ ngang để tăng độ tin cậy.

**c. Kiểm tra dự phòng theo chu kỳ (CRC - Cyclic Redundancy Check) :**

* Dùng một đa thức sinh (generator polynomial) để chia dữ liệu gốc.
* Phần dư sau phép chia được nối vào cuối dữ liệu và gửi đi.
* Bên nhận chia lại dữ liệu + CRC, nếu dư bằng 0 thì dữ liệu hợp lệ.
* CRC được dùng rộng rãi trong mạng máy tính, truyền thông và thiết bị lưu trữ.

## 5. Mã Hamming: Khái niệm, nguyên lý, lịch sử, ứng dụng và ưu, nhược điểm

**a. Khái niệm mã Hamming :**

Mã Hamming là một loại mã kiểm tra được sử dụng để phát hiện và sửa lỗi bit đơn trong quá trình truyền dữ liệu số. Đây là một trong những phương pháp sửa lỗi đầu tiên có khả năng tự động phát hiện và sửa lỗi mà không cần gửi lại toàn bộ dữ liệu.

Trong mã Hamming, nguồn mã hóa thông điệp bằng cách thêm các bit dư thừa vào thông điệp. Các bit dư thừa này hầu hết được chèn và tạo tại các vị trí nhất định trong thông báo để thực hiện quá trình phát hiện và sửa lỗi.

Mã Hamming đặc biệt hữu ích trong các hệ thống truyền thông và bộ nhớ, nơi mà việc truyền lại dữ liệu có thể tốn kém hoặc không khả thi.

**b. Nguyên lý hoạt động của mã Hamming :**

Nguyên lý hoạt động của mã Hamming dựa trên việc chèn các bit dư thừa (parity bits) vào chuỗi dữ liệu gốc để tạo thành một chuỗi mã hóa có thể tự kiểm tra lỗi.

* Các bit dư thừa được đặt tại các vị trí có dạng 2^n (ví dụ: 1, 2, 4, 8, ...).
* Mỗi bit dư thừa có nhiệm vụ kiểm tra một tập hợp các bit theo quy tắc nhị phân, nhằm phát hiện và định vị lỗi xảy ra.
* Khi dữ liệu được nhận, hệ thống kiểm tra lại các bit dư thừa để xác định vị trí lỗi (nếu có), sau đó sửa lỗi bằng cách đảo lại giá trị bit tại vị trí đó.

Công thức xác định số lượng bit dư thừa cần thêm là:

2^P ≥ D + P + 1

Trong đó:

* D: Số bit dữ liệu gốc
* P: Số bit dư thừa cần thêm

**c. Lịch sử hình thành mã Hamming**

Mã Hamming được phát minh bởi Richard W. Hamming – một nhà toán học và khoa học máy tính tại Bell Labs, vào năm 1950.

Ông bắt đầu nghiên cứu vấn đề sửa lỗi sau khi gặp phải hạn chế khi sử dụng máy tính cơ học IBM, nơi chương trình tự động dừng lại khi gặp lỗi mà không thể tiếp tục. Mong muốn tạo ra một phương pháp cho phép máy tính có thể phát hiện và sửa lỗi một cách tự động, ông đã phát triển thuật toán mã Hamming đầu tiên.

Kể từ đó, mã Hamming được áp dụng rộng rãi trong các hệ thống điện tử và truyền thông.

**d. Ứng dụng của mã Hamming**

Mã Hamming được sử dụng trong nhiều lĩnh vực đòi hỏi tính chính xác cao và khả năng xử lý lỗi nhanh chóng, bao gồm:

* Bộ nhớ RAM (ECC - Error Correcting Code)
* Truyền thông vệ tinh và thiết bị mạng
* Modem và truyền dữ liệu tốc độ cao
* Bộ xử lý nhúng trong thiết bị điện tử
* Hệ thống điều khiển CNC như PlasmaCAM
* Truyền tín hiệu qua dây cáp có che chắn chống nhiễu

**e. Ưu điểm của mã Hamming**

* Có khả năng phát hiện và sửa lỗi bit đơn một cách tự động
* Giảm thiểu nhu cầu truyền lại dữ liệu khi lỗi xảy ra
* Có thể thực hiện bằng phần cứng một cách đơn giản
* Phù hợp với môi trường có tỷ lệ lỗi thấp và yêu cầu thời gian thực cao

**f. Nhược điểm của mã Hamming**

* Không thể sửa lỗi nhiều bit xảy ra đồng thời
* Nếu lỗi xảy ra tại nhiều vị trí, kết quả xác định sai vị trí lỗi có thể dẫn đến sửa sai và làm hỏng thêm dữ liệu
* Không phù hợp cho môi trường truyền dẫn có tỷ lệ lỗi cao hoặc lỗi bùng nổ kéo dài

# II. Xây dựng nguyên lí

## 1. Mạch tạo mã Hamming

* Mã Hamming là loại mã tuyến tính được R. W. Hamming đưa ra và sử dụng trong một số hệ thống thông tin.
* Mã này có khả năng sửa sai một lỗi.
* Mã Hamming có sơ đồ tạo mã và giải mã đơn giản.
* Số bít kiểm tra P và số bit tin tức D phải thoả mãn biểu thức: D + P + 1 ≤ 2^P
* Khi số bit dữ liệu tăng thì số bit kiểm tra cũng tăng. Tuy nhiên tốc độ tăng của số bit tin tức nhanh hơn nhiều so với tốc độ tăng của số bit kiểm tra.
* Do vậy, khi số bit tin tức càng lớn thì hiệu quả sử dụng bit chẵn/lẻ càng cao

→ dẫn đến việc mã Hamming trở nên phổ biến.

* Mã Hamming là một bước phát triển của kiểm tra chẵn lẻ và có khả năng sửa sai do xác định được vị trí lỗi.

**a. Mã chẵn – mã lẻ**

* Mã chẵn là mã có tổng số bit 1 là số chẵn.

Ví dụ: 11011 là mã chẵn.

* Mã lẻ là mã có tổng số bit 1 là số lẻ.

Ví dụ: 10011 là mã lẻ.

* Tương tự, một mã Hamming chẵn là mã có tổng số bit là số chẵn, tức:

D1 ⊕ D2 ⊕ ... ⊕ Dn = 0

* Mã Hamming lẻ có:

D1 ⊕ D2 ⊕ ... ⊕ Dn = 1

**b. Xác định số lượng bit kiểm tra P**

* Số lượng bit của mã Hamming tùy thuộc vào số lượng bit của chuỗi dữ liệu.
* Gọi D là số bit của chuỗi dữ liệu, gọi P là số bit kiểm tra của mã Hamming.  
   Tổng số bit phát đi là P + D

**c. Khả năng phát hiện lỗi theo số bit P**

* Với P = 1, ta xác định được một trong 2 kết quả: chuỗi dữ liệu sai hoặc đúng, nhưng không biết vị trí lỗi.
* Với P = 2, có 1 trong 4 trường hợp xảy ra: 2 phép kiểm tra đều đúng, 2 đều sai, phép thứ nhất đúng – phép thứ hai sai, hoặc ngược lại.

→ 4 trường hợp này cho phép kết luận được 1 bit sai trong 3 vị trí.

* Với P = 3, có 8 khả năng xảy ra và có thể kết luận được một bit sai ở 1 trong 7 vị trí.
* Với P bất kỳ, thì có 2^P khả năng xảy ra

→ ta có thể kết luận được 1 bit sai trong (2^P – 1) vị trí.

* Vậy để có thể phát hiện 1 lỗi tại 1 vị trí cụ thể thì số P nhỏ nhất phải thỏa mãn:

2^P – 1 ≥ D + P

**d. Ví dụ cụ thể**

* Giả sử chuỗi dữ liệu cần truyền gồm 4 bit như sau: 1010.

→ Với D = 4 ⇒ P = 3 (BDT trên được thoả mãn)

* Gọi các bit kiểm tra của mã Hamming là P1, P2, P3.
* Gọi các bit dữ liệu là D1, D2, D3, D4

**e. Cách tạo mã Hamming**

* Đầu tiên cần xác định giá trị của các bit kiểm tra tính chẵn lẻ của từ mã.
* Vị trí các bit kiểm tra chẵn lẻ là vị trí 2^n (tính từ bit có trọng số nhỏ nhất).
* Bit kiểm tra chẵn lẻ có nhiệm vụ kiểm tra chẵn/lẻ (tùy theo yêu cầu), kể cả nó trong mã Hamming.
* Bit kiểm tra chẵn/lẻ P kiểm tra các bit mà nó chiếm giữ có giá trị 1 trong biểu diễn nhị phân.

**f. Các vị trí kiểm tra của các bit kiểm tra chẵn/lẻ:**

**Ex: D= 5 🡪 P= 4 🡪 The 5-bit Hamming (5, 4) code word:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** |
| **D5** | **P8** | **D4** | **D3** | **D2** | **P4** | **D1** | **P2** | **P1** |

* P1 tại các vị trí: 1, 3, 5, 7, 9, 11, 13, 15,…
* P2 tại các vị trí: 2, 3, 6, 7, 10, 11, 14, 15,…
* P4 tại các vị trí: 4, 5, 6, 7, 12, 13, 14, 15,…
* P8 tại các vị trí: 8, 9, 10, 11, 12, 13, 14, 15,…

**g. Các điều kiện chẵn/lẻ:**

* Nếu xây dựng mã Hamming chẵn, thì phải thỏa mãn biểu thức:

D1 ⊕ D2 ⊕ ... ⊕ Dn = 0

* Nếu xây dựng mã Hamming lẻ, thì phải thỏa mãn biểu thức:

D1 ⊕ D2 ⊕ ... ⊕ Dn = 1

* Các bit kiểm tra mã Hamming chèn vào vị trí 2^P và dùng cho kiểm tra chẵn lẻ.
* Các bit khác là bit thông tin.

**h. Bảng nhị phân vị trí các bit Hamming :**

| **Thập phân** | **Nhị phân** | **P1** | **P2** | **P4** | **P8** |
| --- | --- | --- | --- | --- | --- |
| 1 | 0001 | ✔ |  |  |  |
| 2 | 0010 |  | ✔ |  |  |
| 3 | 0011 | ✔ | ✔ |  |  |
| 4 | 0100 |  |  | ✔ |  |
| 5 | 0101 | ✔ |  | ✔ |  |
| 6 | 0110 |  | ✔ | ✔ |  |
| 7 | 0111 | ✔ | ✔ | ✔ |  |
| 8 | 1000 |  |  |  | ✔ |
| 9 | 1001 | ✔ |  |  | ✔ |
| 10 | 1010 |  | ✔ |  | ✔ |
| 11 | 1011 | ✔ | ✔ |  | ✔ |
| 12 | 1100 |  |  | ✔ | ✔ |
| 13 | 1101 | ✔ |  | ✔ | ✔ |
| 14 | 1110 |  | ✔ | ✔ | ✔ |
| 15 | 1111 | ✔ | ✔ | ✔ | ✔ |

2. Mạch giải mã Hamming

Để giải mã mã Hamming, cần phải kiểm tra tính chẵn lẻ trên các trường bit mà tính chẵn lẻ đã được thiết lập trước đó :

* **S1 = P1** ⊕ **pos. 3** ⊕ **pos. 5** ⊕ **pos. 7**⊕ **pos. 9** ⊕**...**
* **S2 = P2** ⊕ **pos. 3** ⊕ **pos. 6** ⊕ **pos. 7**⊕ **pos. 10** ⊕**...**
* **S4 = P4** ⊕ **pos. 5** ⊕ **pos. 6** ⊕ **pos. 7** ⊕**...**
* **S8 = P8** ⊕ **pos. 9** ⊕ **pos. 10** ⊕**...**
* Đối với mã Hamming chẵn :

Nếu S8S4S2S1 = 0000, chúng ta kết luận không có lỗi trong mã Hamming.

Mặt khác, nếu nó có giá trị khác không, nó chỉ ra vị trí bit bị lỗi.

Ví dụ, nếu S8S4S2S1 = 0110, thì bit 6 bị lỗi. Để sửa lỗi này, ta lật lại bit 6 để sửa lỗi.

* Đối với mã Hamming lẻ :

Nếu S8S4S2S1 = 1111, chúng ta kết luận không có lỗi trong mã Hamming. Mặt khác, nếu nó khác 1111, thì chỉ ra vị trí bit bị lỗi: 

Ví dụ, nếu S8S4S2S1 = 0110, thì bit 9 bị lỗi. Để sửa lỗi này, ta lật lại bit 9 để sửa lỗi.

* **Tổ hợp mã Hamming(ví dụ) :**

Giả sử ta chọn mã Hamming chẵn thì các bit kiểm tra được xác định như sau (sử dụng mã Hamming (7,4)):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Vị trí | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Bit | P1 | P2 | D1 | P4 | D2 | D3 | D4 |

Giả sử dữ liệu D1 = 1, D2 = 0, D3 = 1, D4 = 0.

* P1 kiểm tra các vị trí 1, 3, 5, 7:

P1 ⊕ D1 ⊕ D2 ⊕ D4 = 0 ⇒ P1 ⊕ 1 ⊕ 0 ⊕ 0 = 0 ⇒ P1 = 1

* P2 kiểm tra các vị trí 2, 3, 6, 7:

P2 ⊕ D1 ⊕ D3 ⊕ D4 = 0 ⇒ P2 ⊕ 1 ⊕ 1 ⊕ 0 = 0 ⇒ P2 = 0

* P4 kiểm tra các vị trí 4, 5, 6, 7:

P4 ⊕ D2 ⊕ D3 ⊕ D4 = 0 ⇒ P4 ⊕ 0 ⊕ 1 ⊕ 0 = 0 ⇒ P4 = 1

***→ Mã Hamming là: 1011010***

* Nếu S1 = S2 = S34= 0, không có lỗi xảy.
* Nếu S1 = 1, S2 = S4 = 0, 1 trong các bit ở vị trí 1, 3, 5, 7 bị lỗi. Nhưng S2 = S4 = 0, có nghĩa là các vị trí 2, 3, 6, 7 và 4, 5, 6, 7 đúng. Vậy bít sai ở vị trí 1.

**Giả sử bản tin nhận được là: 1011110**

(Tức là bit thứ 5 bị sai, từ 0 thành 1)

Ta tính các syndrome S1, S2, S4 như sau:

* S1 = P1 ⊕ D1 ⊕ D2 ⊕ D4 = 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1
* S2 = P2 ⊕ D1 ⊕ D3 ⊕ D4 = 0 ⊕ 1 ⊕ 1 ⊕ 0 = 0
* S4 = P4 ⊕ D2 ⊕ D3 ⊕ D4 = 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1

Tổ hợp syndrome là:

S4S2S1 = 101₂ = 5₁₀

→ Bit sai nằm ở vị trí 5. Do đó, ta biết vị trí cần sửa lỗi chính là vị trí số 5 trong chuỗi mã Hamming.

# III. Giới thiệu linh kiện

A close-up of a chip

Description automatically generated**Một số IC được sử dụng trong mạch:**

* **IC 74164 :** là 8-bit serial-in, parallel-out shift register, dùng để chuyển đổi dữ liệu nối tiếp sang song song, mở rộng số lượng cổng đầu ra, tạo chuỗi tín hiệu điều khiển và lưu trữ tạm thời dữ liệu. Nó thường được sử dụng trong các mạch LED ma trận, màn hình LCD và hệ thống đồng bộ hóa tín hiệu.

**A table with black text

Description automatically generated**

**A diagram of a circuit

Description automatically generated**

* **IC 74HC373** là 8-bit latch dùng để lưu trữ tạm thời dữ liệu và điều khiển đầu ra, thường dùng trong hệ thống vi xử lý và bộ nhớ.

A close-up of a chip

Description automatically generated**A white sheet with black text

Description automatically generatedA diagram of a circuit diagram

Description automatically generated**

* A close-up of a chip

  Description automatically generated**IC 7493** là 4-bit binary ripple counter, dùng để đếm xung nhịp hoặc chia tần số, thường dùng trong mạch đếm, tạo thời gian trễ và bộ chia tần số.

A table of numbers and letters

Description automatically generated

**A table with black text and white text

Description automatically generated**

* **IC 74150** là 16-to-1 multiplexer, dùng để chọn một trong 16 tín hiệu đầu vào và xuất ra một tín hiệu duy nhất, thường dùng trong mạch ghép kênh, hệ thống chuyển mạch và xử lý tín hiệu số

**A table of letters and numbers

Description automatically generated**A diagram of a circuit diagram

Description automatically generated

* **IC 74154** là 4-to-16 decoder/demultiplexer, dùng để giải mã địa chỉ hoặc phân phối tín hiệu từ 4 đầu vào sang 16 đầu ra, thường dùng trong bộ nhớ, hệ thống chọn thiết bị và mạch điều khiển.

A diagram of a dual-in-line package

Description automatically generated

# IV. Sơ đồ khối và mạch cụ thể

## 1. Sơ đồ khối

Dưới đây là sơ đồ khối mô tả quá trình truyền và sửa lỗi tín hiệu sử dụng mã Hamming:

Tín hiệu ban đầu

**Khối** tìm lỗi

Khối tạo mã Hamming

Nguồn tín hiệu

Khối phát đi tín hiệu

Khối sửa lỗi

## 2. Nguyên lí hoạt động :

A screen shot of a computer screen

Description automatically generated

**2.1 Nguồn tín hiệu :**

a. Nguyên lí hoạt động :

* Khi **MR** (U65) ở mức cao, thanh ghi dịch U65 sẵn sàng nhận và dịch dữ liệu.
* Tín hiệu **CLK** sẽ quyết định thời điểm dịch dữ liệu từ A và B vào U65. Mỗi xung CLK dịch dữ liệu sang một bit, từ **Q0** đến **Q7**.
* Dữ liệu dịch vào từ A và B sẽ lần lượt chuyển sang các bit cao hơn, tạo thành chuỗi 8 bit tại Q0-Q7.
* Khi **LE** của U58 ở mức cao, dữ liệu từ U65 được chốt và lưu vào U58.
* Khi **OE** ở mức thấp, dữ liệu từ U58 sẽ xuất ra các chân Q0-Q7 để sử dụng tiếp theo.

b. Chức năng của công tắc:

* Công tắc nối với **A** và **B** quyết định trạng thái bit đầu vào, cho phép người dùng tự chọn dữ liệu để dịch vào U65.
* Công tắc nối với **MR** để reset toàn bộ thanh ghi dịch, xóa dữ liệu nếu cần.

**2.2. Khối tạo mã Hamming:**

* Mã Hamming (8, 4) sử dụng 4 bit dữ liệu (D1-D4) và 4 bit kiểm tra (P1-P4) để tạo ra khung dữ liệu 8 bit.
* Các bit kiểm tra được tính theo công thức:
* Biểu thức kiểm tra tính chẵn:

Pc ⊕ d1 ⊕ d2 ⊕ d3 ⊕ ... ⊕ dn-1 = 0

* Biểu thức kiểm tra tính lẻ:

PL ⊕ d1 ⊕ d2 ⊕ d3 ⊕ ... ⊕ dn-1 = 1

* Tạo lỗi đơn bit: đảo bit ở cổng XOR bất kỳ.

**2.3 Khối phát đi tín hiệu:**

2.3.1. **Truyền đi tín hiệu:**

**a. Hoạt động đếm:**

* IC 7493 là **bộ đếm 4-bit** với các ngõ ra QA, QB, QC, QD tương ứng với các bit 0, 1, 2, 3 của số đếm.
* Tín hiệu xung từ công tắc được đưa vào chân CKA của IC 7493 để tăng bộ đếm.
* Mỗi xung vào sẽ tăng giá trị đếm lên 1.

**b. Giải mã và hiển thị:**

* IC 74150 (Multiplexer) nhận tín hiệu 4 bit từ IC 7493 và chọn một trong 16 ngõ vào X0-X15 để tạo ra tín hiệu cho LED 7 đoạn.
* Khi giá trị đếm là từ 0 đến 11, IC 7493 sẽ xuất tín hiệu tương ứng để hiển thị số trên LED 7 đoạn.
* Khi đếm vượt quá 11, tín hiệu từ IC 7493 sẽ được đảo bởi cổng NOT để reset lại bộ đếm, đưa về 0.

**c. Mạch reset và giới hạn:**

* Mạch sử dụng cổng AND (U69) để reset khi giá trị đếm đạt 12 (1100), đảm bảo chỉ hiển thị từ 0 đến b.

**d. Kết quả hiển thị:**

* Số trên LED 7 đoạn sẽ thay đổi từ 0 đến b khi nhấn công tắc.
* Sau khi đạt b, mạch tự động quay lại 0 và tiếp tục đếm.

2.3.2. **Thu vào tín hiệu :**

**Hoạt động của mạch:**

**a. Truyền và chốt dữ liệu từ IC 74164:**

* **Nhập dữ liệu:**
  + Dữ liệu được đưa vào hai IC 74164 thông qua các chân **A** và **B**.
  + Khi tín hiệu xung nhịp (CLK) có cạnh lên, dữ liệu từ **A** và **B** sẽ dịch lần lượt từ **Q0** đến **Q7**.
  + Hai IC 74164 nối tiếp nhau, tổng cộng có 16 bit dữ liệu song song đầu ra từ hai IC này.
* **Xóa dữ liệu:**
  + Khi chân **MR** của mỗi IC ở mức thấp, toàn bộ các bit đầu ra từ **Q0** đến **Q7** đều bị reset về 0.
  + Để mạch hoạt động bình thường, cần giữ **MR** ở mức cao.

**b. Chốt và lưu trữ dữ liệu từ IC 74HC373:**

* **Nhận dữ liệu từ 74164:**
  + Đầu ra song song từ các IC 74164 được đưa tới các chân đầu vào **D0** đến **D7** của IC 74HC373.
  + Dữ liệu này sẽ được chốt lại khi tín hiệu **LE** của IC 74HC373 ở mức cao.
* **Chốt dữ liệu:**
  + Khi **LE** ở mức cao, dữ liệu hiện tại từ các chân **D0** đến **D7** sẽ được chốt và giữ ổn định tại các đầu ra **Q0** đến **Q7** ngay cả khi tín hiệu xung nhịp thay đổi.
  + Khi **LE** chuyển về mức thấp, dữ liệu sẽ được giữ nguyên và không bị thay đổi bởi tín hiệu xung nhịp nữa.
* **Xuất dữ liệu:**
  + Tín hiệu **OE** điều khiển đầu ra ba trạng thái của IC 74HC373.
  + Khi **OE** ở mức thấp, dữ liệu từ **Q0** đến **Q7** sẽ được đưa ra mạch ngoài.
  + Khi **OE** ở mức cao, các đầu ra sẽ ở trạng thái cách ly (tri-state), không ảnh hưởng tới mạch ngoài.

**c. Phần điều khiển xung nhịp và reset:**

* **Tạo xung nhịp:**
  + Tín hiệu xung nhịp được tạo từ nút nhấn và điện trở **R3**, cho phép người dùng tạo tín hiệu thủ công.
  + Tín hiệu này được đưa tới chân **CLK** của hai IC 74164 để dịch bit dữ liệu.
* **Reset đồng thời:**
  + Hai IC 74164 có chân **MR** nối chung, cho phép reset đồng thời toàn bộ mạch khi cần thiết.

**2.4 Khối tìm lỗi:**

Dùng các cổng XOR để tìm Si

Tổ hợp S8S4S2S1 được biểu thị qua các LED để phát hiện lỗi.

**2.5 khối sửa lỗi:**

* **IC 74154** nhận **4 bit đầu vào (A, B, C, D)** từ các bit kiểm tra parity (p₁, p₂, p₄, ...).
* Các chân **E1, E2** (chân enable) được kích hoạt để IC hoạt động.
* Dựa trên tổ hợp bit parity, IC giải mã và xuất tín hiệu ở **1 trong 16 chân đầu ra** (tương ứng vị trí bit lỗi).
* Vì đầu ra ở mức tích cực thấp nên cần sử dụng cổng NOT
* Sử dụng cổng XOR để so sánh giữa bit của mã Hamming và bit cần sửa lỗi

# Lời kết thúc

Qua bài tập lớn này, nhóm chúng em đã có cơ hội tìm hiểu sâu hơn về mã Hamming – một trong những giải pháp hiệu quả nhất để phát hiện và sửa lỗi bit đơn trong quá trình truyền dữ liệu. Từ việc tìm hiểu lý thuyết về các loại lỗi phổ biến, nguyên lý hoạt động của mã Hamming cho đến cách xây dựng mạch phần cứng thực tế, chúng em đã nhận thấy tầm quan trọng của mã Hamming trong việc nâng cao độ tin cậy và hiệu quả của các hệ thống truyền thông hiện đại.

Mã Hamming không chỉ đơn giản là một kỹ thuật mã hóa, mà còn là một minh chứng cho sự sáng tạo và nỗ lực không ngừng của các nhà khoa học trong việc cải thiện chất lượng truyền tải dữ liệu. Với khả năng phát hiện và sửa lỗi bit đơn, mã Hamming đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực như viễn thông, lưu trữ dữ liệu, hệ thống nhúng và cả trong các bộ nhớ RAM hiện đại.

Chúng em hy vọng rằng, qua bài tập lớn này, nhóm đã có thể mang đến cho cô và các bạn một cái nhìn rõ nét hơn về vai trò và ý nghĩa của mã Hamming trong thế giới số hóa. Chúng em xin gửi lời cảm ơn chân thành đến cô đã hướng dẫn và tạo điều kiện cho nhóm chúng em thực hiện đề tài này, cũng như cảm ơn các bạn đã đóng góp ý kiến.

Chúng em xin kính chúc cô luôn mạnh khỏe và chúc các bạn luôn học tập tốt. Xin chân thành cảm ơn!

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |