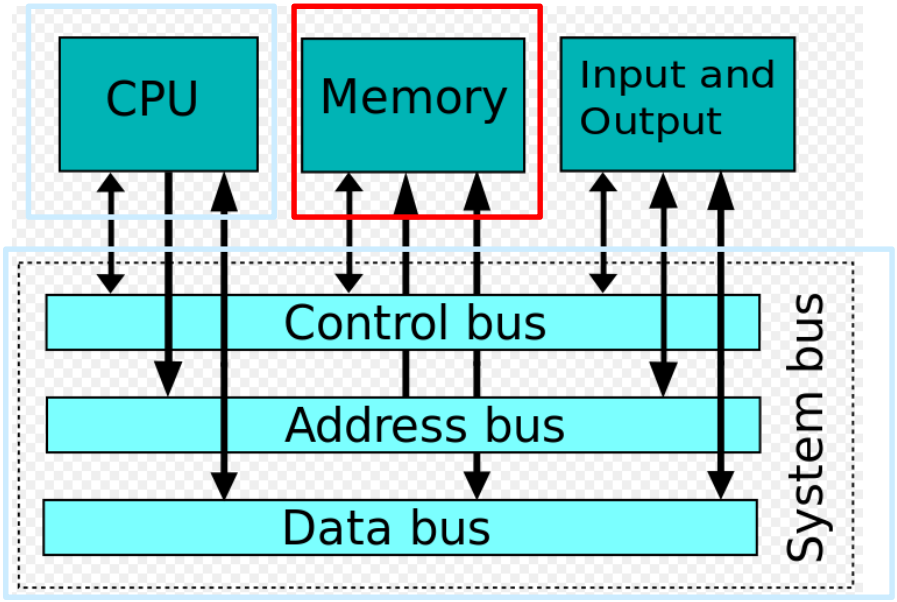
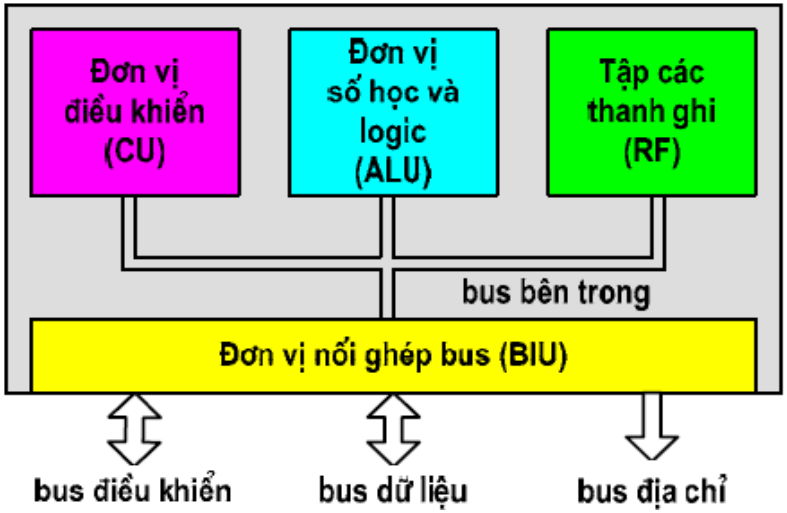
Chương 5
Hệ thống nhớ máy tính

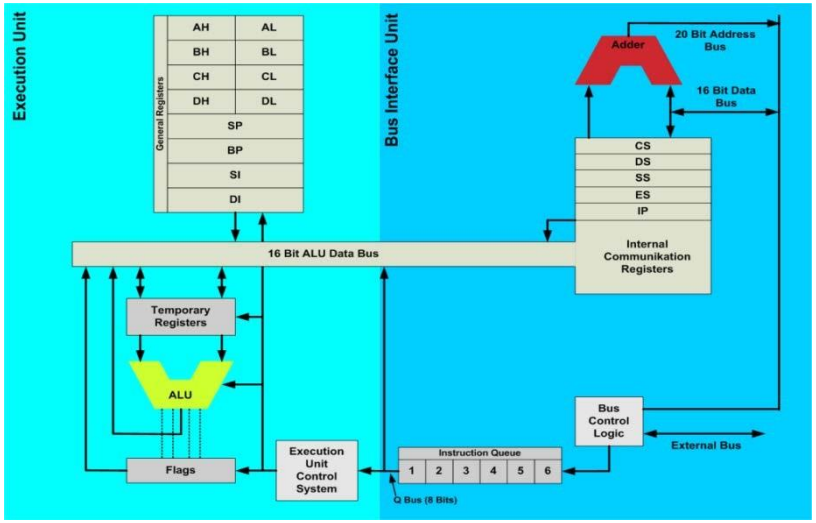


Giảng viên: ThS. Phan Như Minh

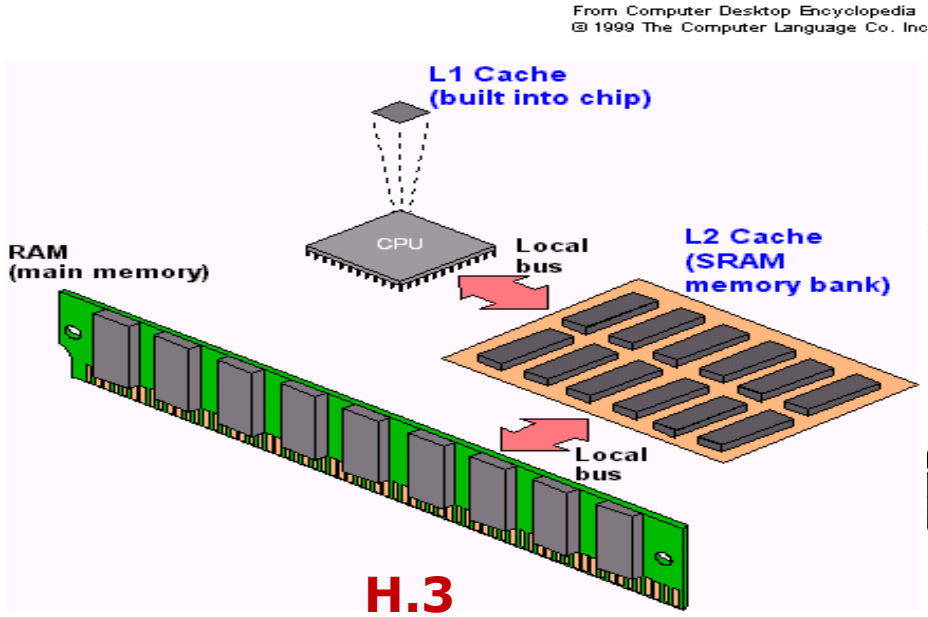
Quan sát và định vị các thành phần nhớ?



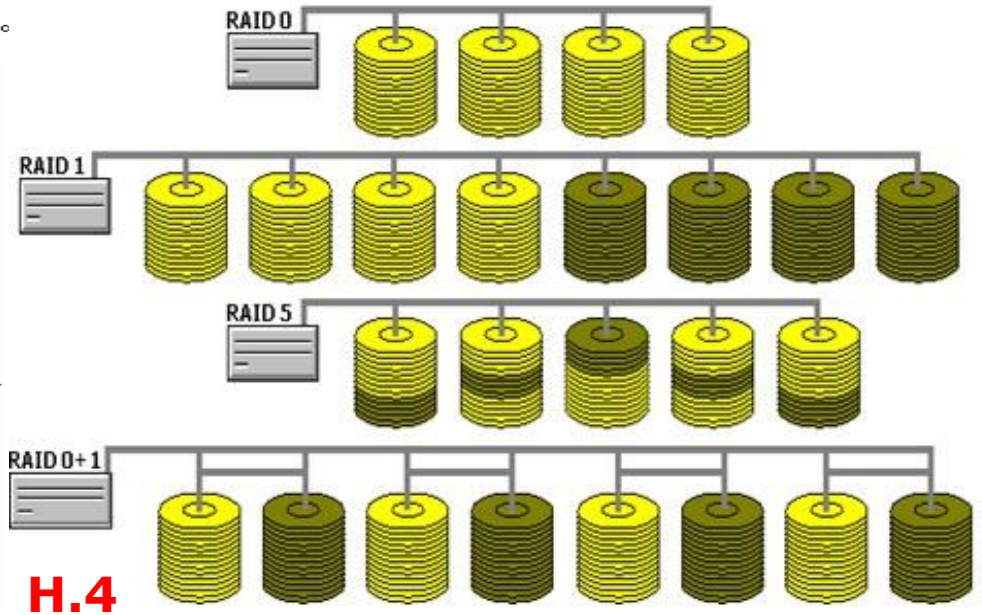
H.1



H.3



H.3



H.4

Vấn đề đặt ra trong chương này

1. Có những loại bộ nhớ nào đang được sử dụng (phân loại)?
2. Ví trị bên trong hệ thống máy tính (phân cấp)?
3. Đặc trưng, ý nghĩa của các thành phần nhớ bên trong hệ thống máy tính (chức năng)?
4. Nguyên tắc hoạt động?

Nội dung trình bày

1. Tổng quan về hệ thống nhớ
2. Bộ nhớ bán dẫn
3. Bộ nhớ trong
4. Bộ nhớ ngoài

5.1 Tổng quan về hệ thống nhớ

- Các đặc trưng của hệ thống nhớ
- Phân cấp bộ nhớ
- Phát hiện và hiểu chỉnh lỗi

1. Các đặc trưng của hệ thống nhớ

- Vị trí
  - Bên trong CPU:
    - tập thanh ghi
  - Bộ nhớ trong:
    - bộ nhớ chính
    - bộ nhớ cache
  - Bộ nhớ ngoài: các thiết bị nhớ,
  - Bộ nhớ lưu trữ trên mạng: cloud system, dropbox, copy, email, inbox,…
- Dung lượng
  - Số lượng ngăn nhớ, độ dài từ nhớ
  - Sides, tracks, sectors,…

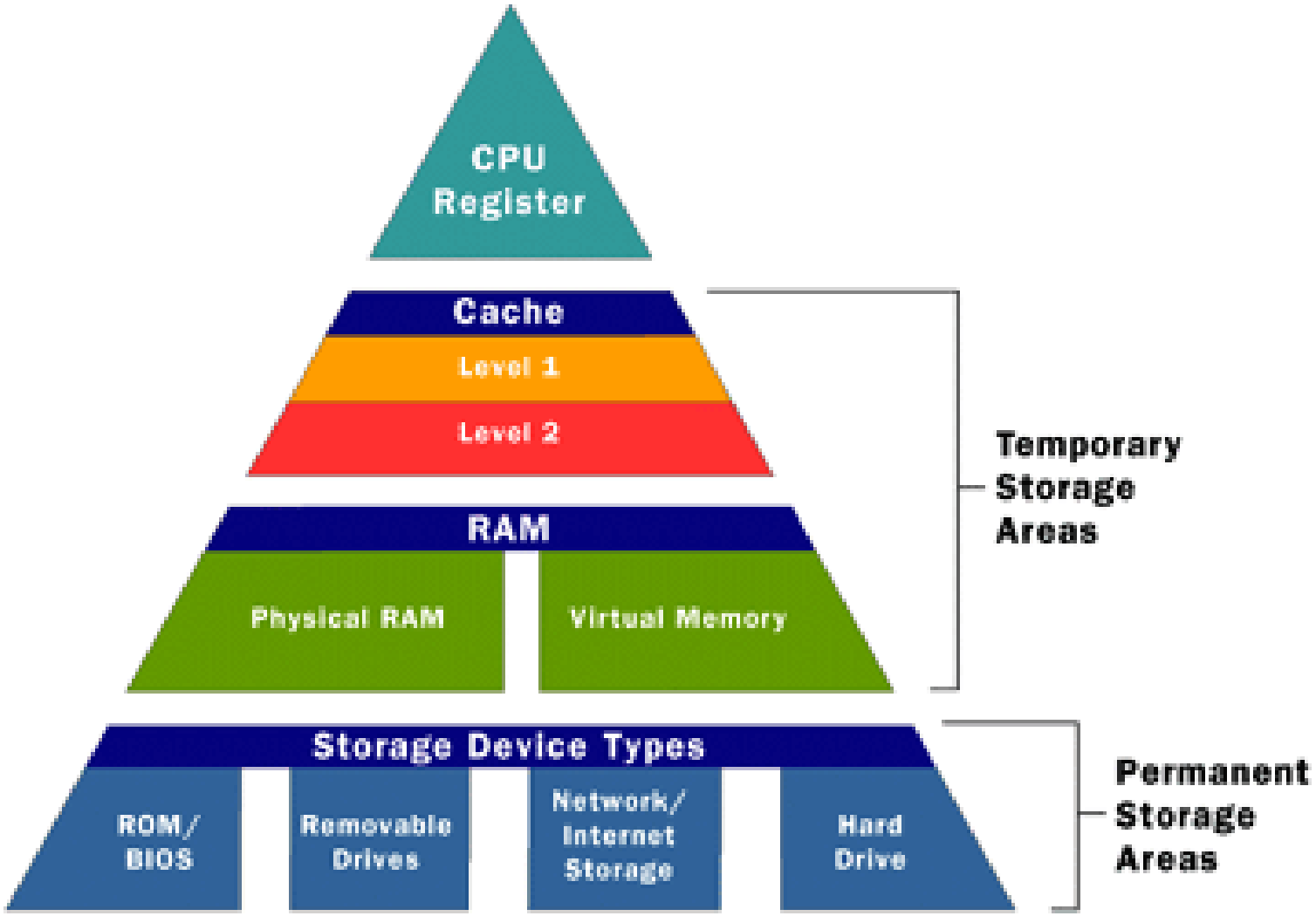
Các đặc trưng của hệ thống nhớ (tiếp)

- Đơn vị truyền
  - Từ nhớ
  - Khối nhớ
- Phương pháp truy nhập
  - Truy nhập tuần tự
  - Truy nhập trực tiếp
  - Truy nhập ngẫu nhiên
  - Truy nhập liên kết

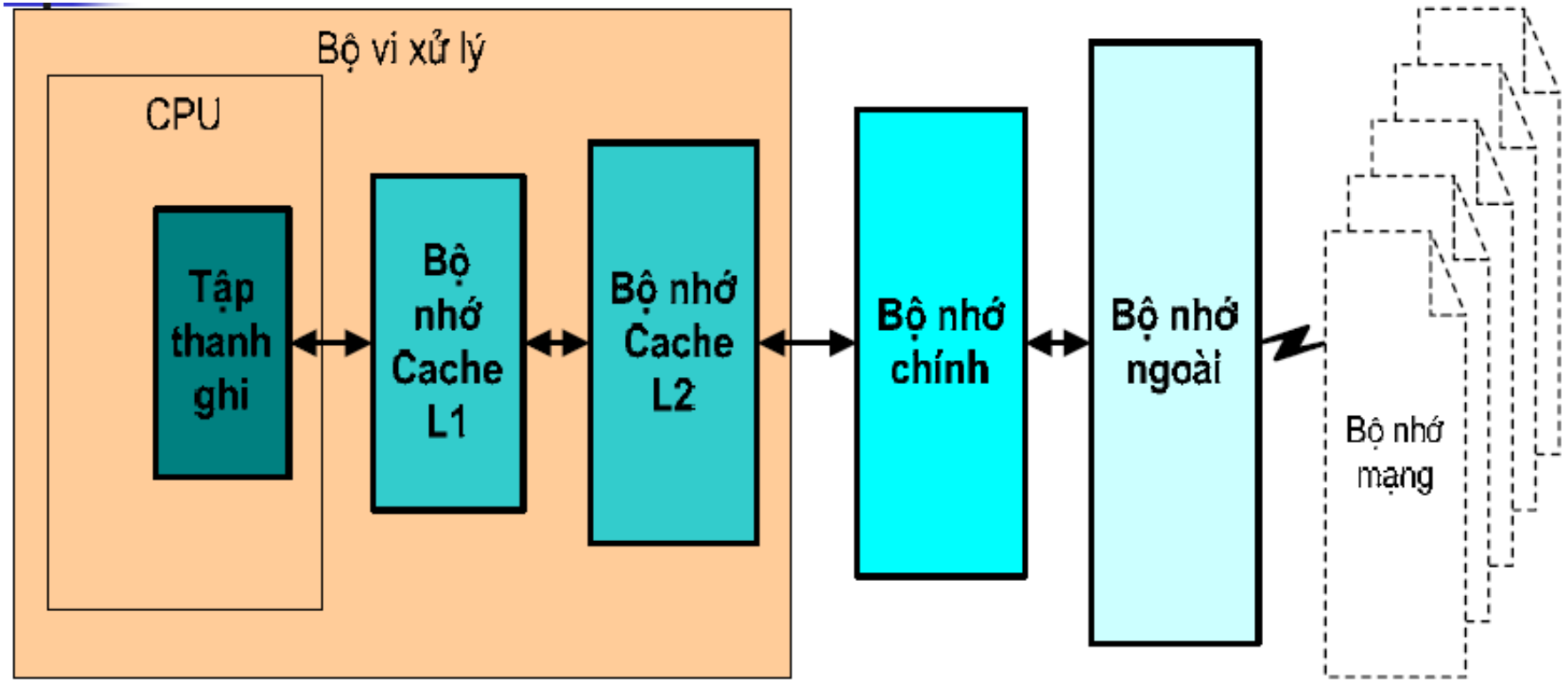
Các đặc trưng của hệ thống nhớ (tiếp)

- Hiệu năng (performance)
  - Thời gian truy nhập
  - Chu kỳ nhớ
  - Tốc độ truyền
- Kiểu vật lý
  - Bộ nhớ bán dẫn
  - Bộ nhớ từ
  - Bộ nhớ quang
- Các đặc tính vật lý
  - Khả biến / Không khả biến
  - Xoá được / không xoá được

2. Phân cấp hệ thống nhớ



Ví dụ hệ thống nhớ thông dụng



- Từ trái sang phải:
  - dung lượng tăng dần
  - tốc độ giảm dần
  - giá thành/1bit, byte giảm dần

3. Phát hiện và hiệu chỉnh lỗi trong bộ nhớ

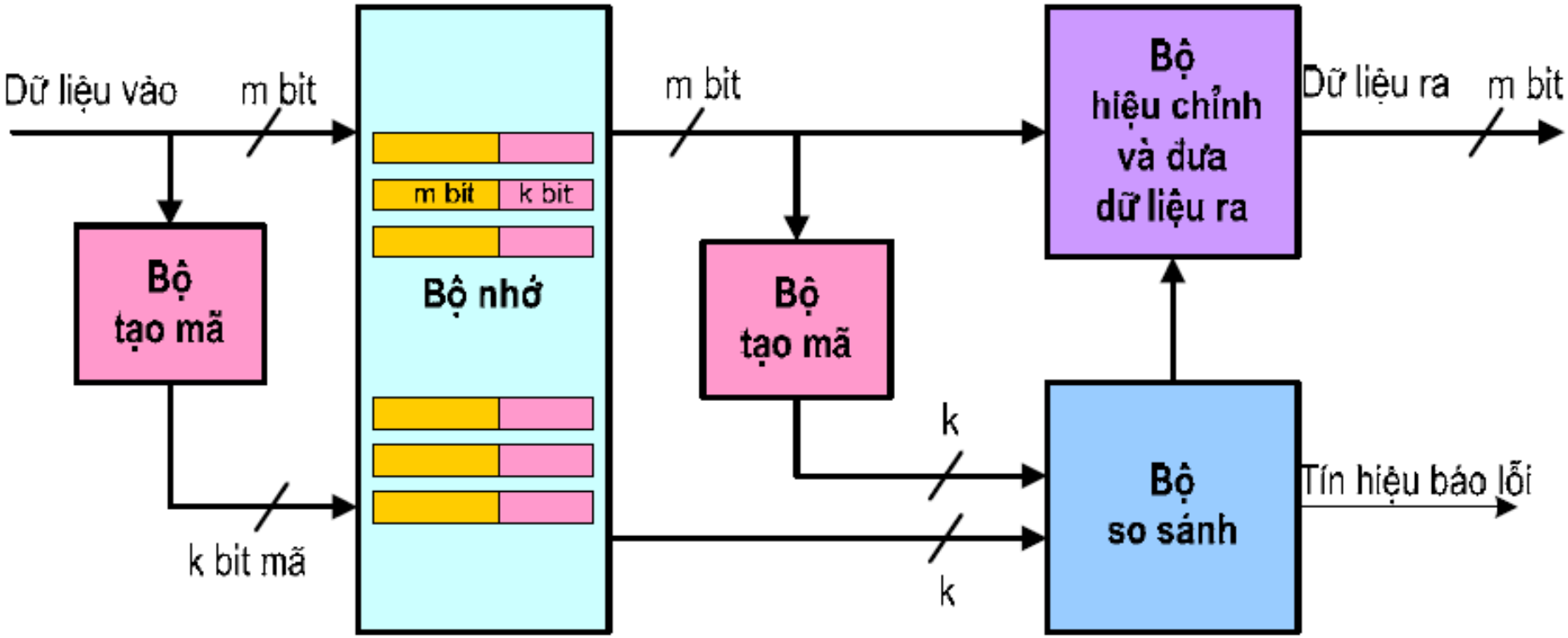
- Nguyên tắc chung: cần tạo ra và lưu trữ thêm thông tin dư thừa.
- Từ dữ liệu cần ghi vào bộ nhớ: m bit
- Cần tạo ra và lưu trữ từ mã: k bit

  => Lưu trữ (m+k) bit

VD: RAID (Redundant Array of Independent Disks)

- Khi đọc ra có các khả năng sau:
  - Không phát hiện thấy dữ liệu lỗi
  - Phát hiện thấy dữ liệu lỗi và có thể hiệu chỉnh dữ liệu thành đúng
  - Phát hiện thấy lỗi nhưng không có khả năng hiệu chỉnh => cần phát ra tín hiệu báo lỗi.

Sơ đồ phát hiện và hiệu chỉnh lỗi



Error Detection / Correction
(Phát hiện lỗi / sửa lỗi)

Error Detection / Correction

Why might we need Error detection/correction (Tại sao chúng ta có thể cần phát hiện / sửa lỗi)?

Even & Odd Parity (Chẵn lẻ & chẵn lẻ)

Error detection (Phát hiện lỗi)

Hamming code

Used for error detection & error correction
(sử dụng để phát hiện lỗi và sửa lỗi)

Parity bits

ASCII – 7 bit code (hex 00 to 7F): Mã là 7 bit

Could use "8th" bit for parity bit (8 bít cho bit chăn or lẻ):

X1011010

Even parity: make total number of "1" bits is even

01011010 (thêm 0 để tổng số bít làm bít chẵn)

Odd parity: make total number of "1" bits odd

11011010 (thêm 1 để tổng số bít làm bít lẻ)

If a parity bit is added to a bit stream, then there is a basis to check for bit(s) being corrupted.

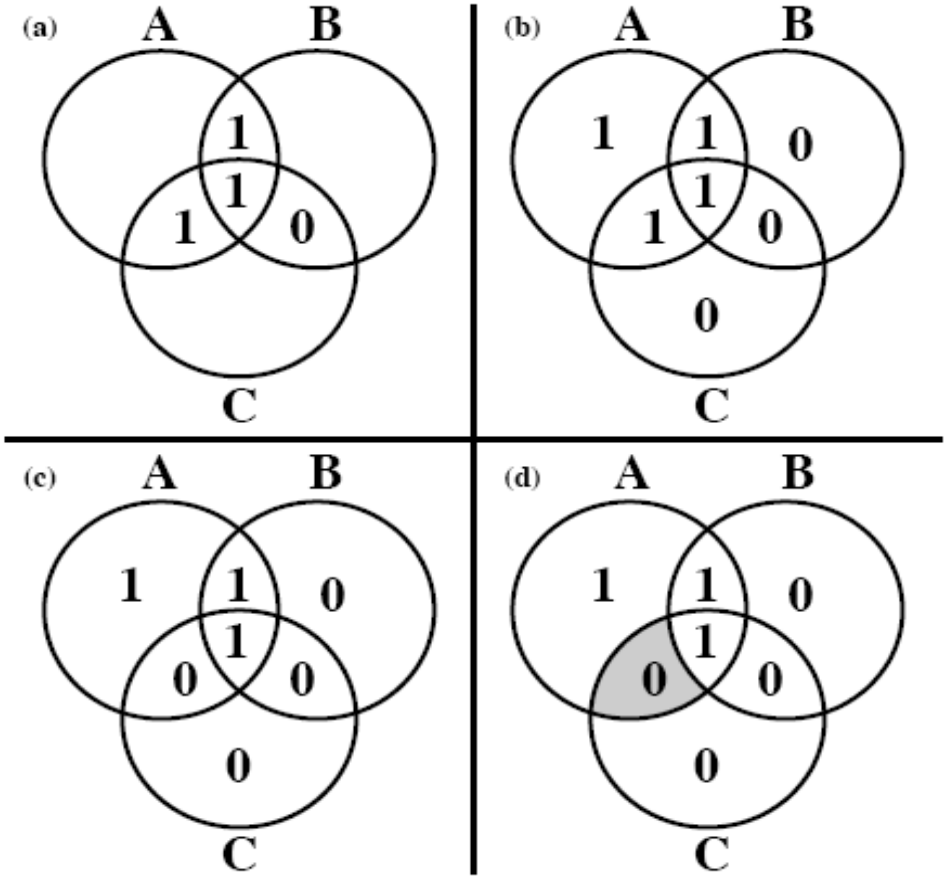
(khi thêm bít vào để kiểm tra)

Error detecting Code Example

Error Detection/correction coding:

Word:

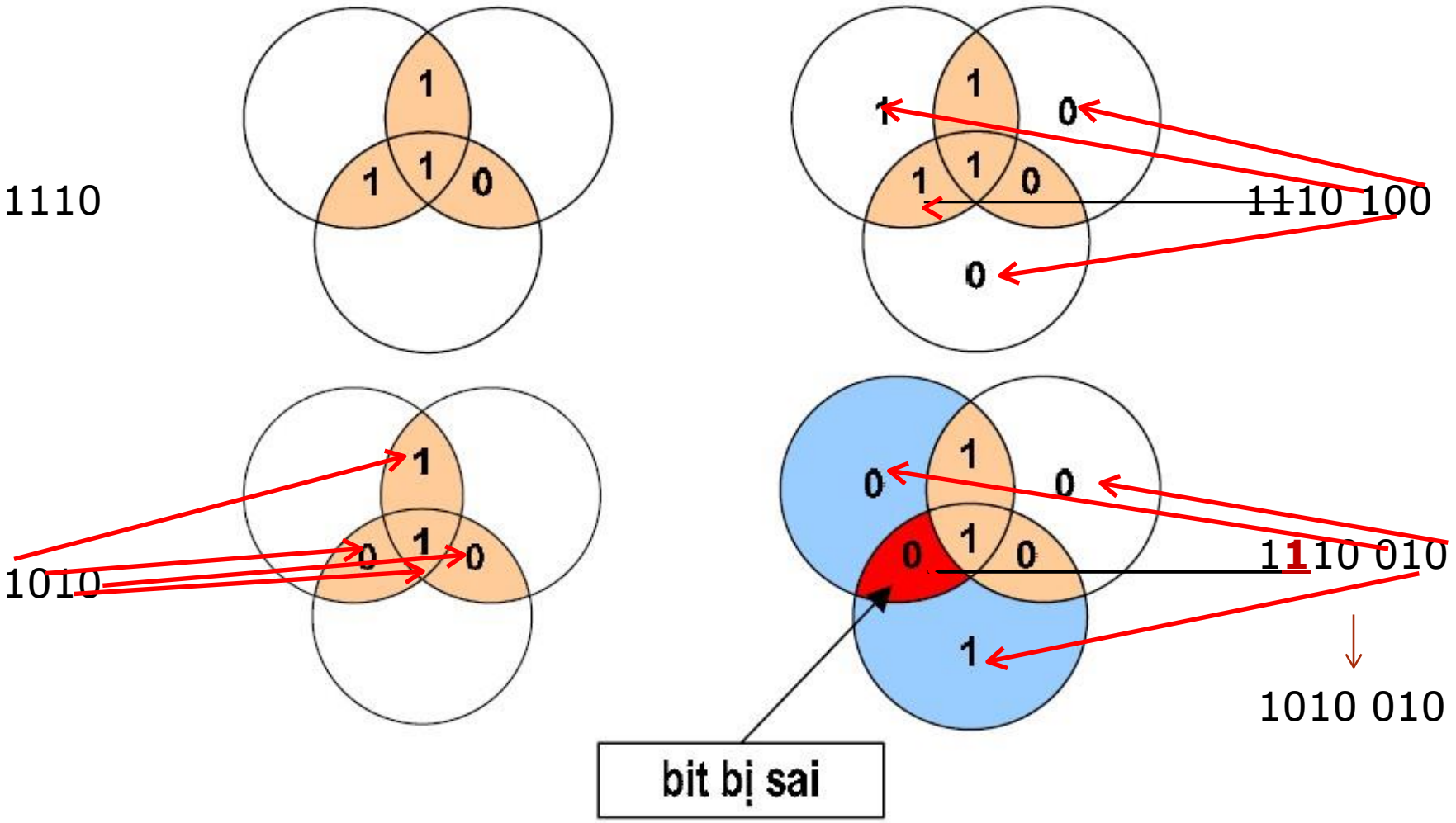
With Error:



With even parity:

Identifying error:

Ví dụ mã sửa lỗi Hamming (m=4, k=3)

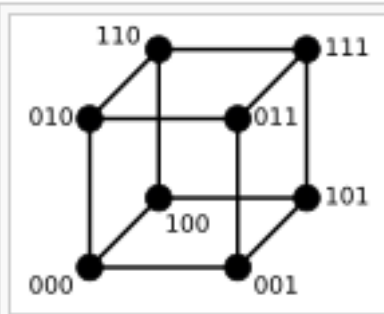


Hamming Distance (khoảng cách hamming)

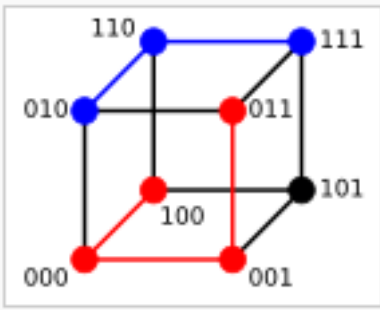
- The Hamming distance is the number of bits that have to be changed to get from one bit pattern to another (Khoảng cách Hamming là số bit phải được thay đổi để có được từ mẫu bit này sang mẫu khác)

Example: 10010101 & 1001**10**01 have a hamming distance of 2

- For any coding whose members have a Hamming distance of two, any one bit error can be detected. Why? Đối với bất kỳ mã hóa nào có thành viên có khoảng cách Hamming là hai, bất kỳ lỗi một bit nào cũng có thể được phát hiện. Tại sao?



3-bit binary cube for finding hamming distance

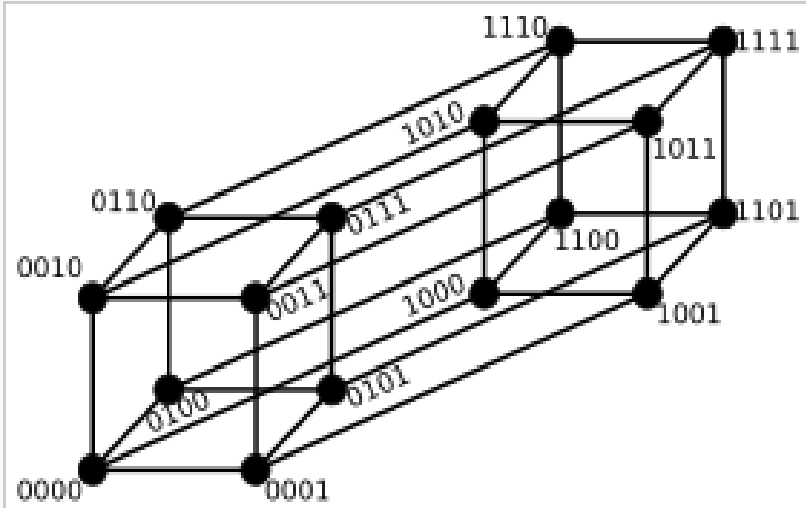


Two example distances: 100->011 has distance 3 (red path); 010->111 has distance 2 (blue path)

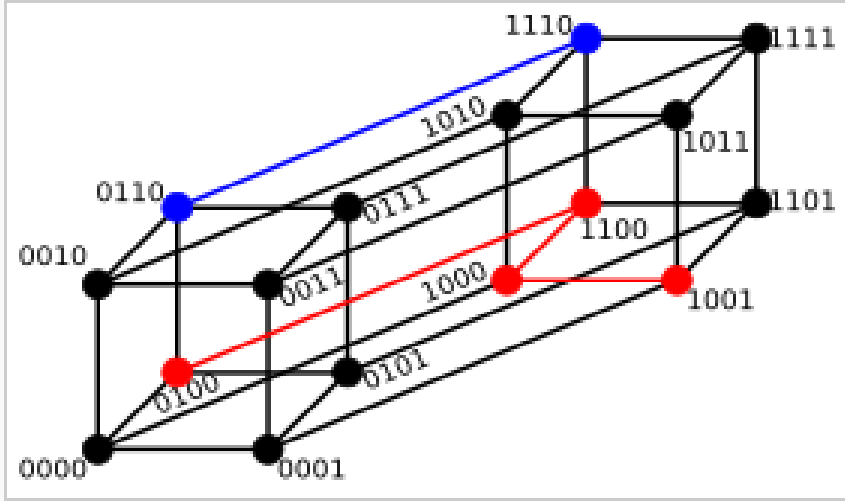
Hamming Distance

For any coding whose members have a Hamming distance of three, any one bit error can be detected and corrected. Why? Có khoảng cách Hamming là ba, bất kỳ lỗi một bit nào cũng có thể được phát hiện và sửa chữa. Tại sao?

And any two bit error can be detected. Why (Và bất kỳ lỗi hai bit có thể được phát hiện. Tại sao)?

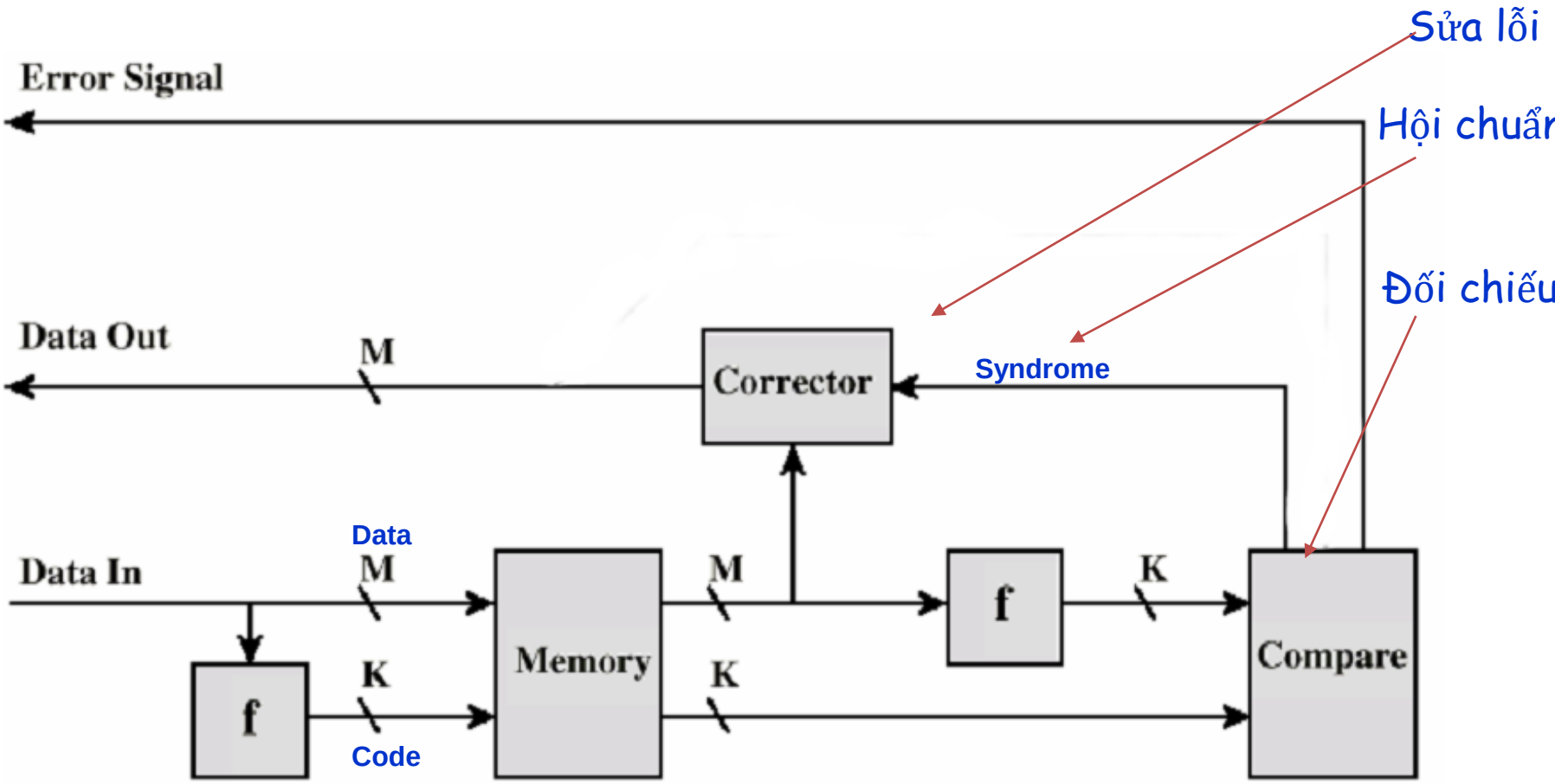


4-bit binary hypercube for finding hamming distance



Two example distances: 0100->1001 has distance 3 (red path); 0110->1110 has distance 1 (blue path)

Error Correcting Code Function



The output of the "Compare" to the "Corrector" is termed the "syndrome", and is K bits long (Kết quả đầu So bộ chỉnh sửa và được gọi là "syndrome" và có độ dài K bit)

Hamming Code Design – determining K (xác định)

To store an M bit word with detection/correction takes M+K bit words
(Để lưu trữ một từ M bit với phát hiện / chỉnh sửa phải mất các từ bit M + K

If K =1, we can detect single bit errors but not correct them
(Nếu K = 1, phát hiện các lỗi bit đơn nhưng không sửa)

If $2^K - 1 >= M + K$, we can detect, identify, and correct all single bit errors, i.e. the syndrome contains the information to correct any single bit error
(Nếu phát hiện, xác định và sửa tất cả các lỗi bit đơn, tức là "syndrome" chứa thông tin để sửa bất kỳ lỗi bit đơn nào)

Example: For M = 8:

and K = 3: $2^3 - 1 = 7 < 8 + 3$ (doesn't work)
and K = 4: $2^4 - 1 = 15 > 8 + 4$ (works!)

Therefore, we must choose K =4,
i.e., the minimum size of the syndrome is 4

Hamming Code Syndrome

- If we compare the read K bits compared with the write K bits, using an EXOR function, the result is called the "syndrome".

"So sánh các bit: read K, K write) →sử dụng hàm EXOR

- If the syndrome is all zeros, there were no errors.

→(là 0, không có lỗi).

- If there is a 1 bit somewhere, we know it represents an error.

→(là 1, Có lỗi ở đâu đó).

Increased word length for error correcting

	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
Data Bits	Check Bits	% Increase	Check Bits	% Increase
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

Hamming Code Syndrome Design Criteria

For convenience, we would like to generate a 4-bit syndrome for an 8-bit data word with the following characteristics:

- If the syndrome contains all 0s, no error has been detected.
- If the syndrome contains one and only one bit set to 1, then an error has occurred in one of the 4 check bits. No correction is needed.
- If the syndrome contains more than one bit set to 1, then the numerical value of the syndrome indicates the position of the data bit in error. This data bit is inverted for correction.

A Layout of Data and Check Bits that Achieves Our Design Criteria:

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Bit number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1

D8=0, D7=0,D6=1,D5=1, D4=1,D3=0,D2=0,D1=1

0011 1001

C1 is a parity check on every data bit whose position is xxx1

C1 = D1 exor D2 exor D4 exor D5 exor D7=1exor0exor1exor1exor0=1

C2 is a parity check on every data bit whose position is xx1x

C2 = D1 exor D3 exor D4 exor D6 exor D7=1exor0exor1exor1exor0=1

C4 is a parity check on every data bit whose position is x1xx

C4 = D2 exor D3 exor D4 exor D8=0exor0exor1exor0=1

C8 is a parity check on every data bit whose position is 1xxx

C8 = D5 exor D6 exor D7 exor D8=1exor1exor0exor0=0

Example:

Data stored = 00111001

Check bits:

$$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$
$$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

Putting it together:

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word stored as	0	0	1	1	0	1	0	0	1	1	1	1

Example:

Let us verify that this scheme works with an example. Assume that the 8-bit input word is 00111001, with data bit D1 in the rightmost position. The calculations are as follows:

$$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$
$$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

Suppose now that data bit 3 sustains an error and is changed from 0 to 1. When the check bits are recalculated, we have

$$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C2 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$
$$C4 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$
$$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

When the new check bits are compared with the old check bits, the syndrome word is formed:

	C8	C4	C2	C1
	0	1	1	1
$\oplus$	0	0	0	1
	0	1	1	0

The result is 0110, indicating that bit position 6, which contains data bit 3, is in error.

Example:

Word fetched:

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word fetched as	0	0	1	1	0	1	1	0	1	1	1	1

Check Bits:

$$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C2 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$
$$C4 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$
$$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

Comparing:

C8 C4 C2 C1

0 1 1 1 Orig Check Bits

0 0 0 1 New Check Bits

0 1 1 0 Syndrome

0110 = 6 → bit position 6 is wrong, i.e. bit D3 is wrong

Putting it all together:

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word stored as	0	0	1	1	0	1	0	0	1	1	1	1
Word fetched as	0	0	1	1	0	1	1	0	1	1	1	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Check bit					0				0		0	1

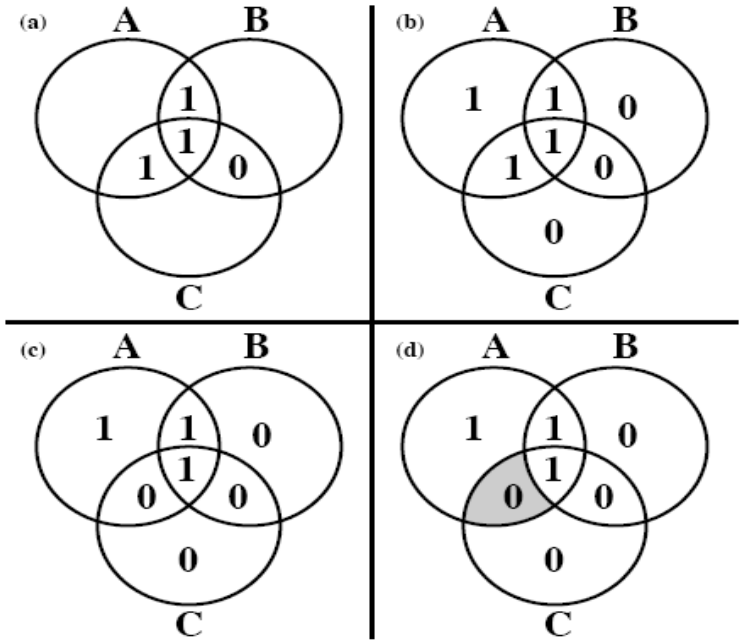
Increased word length for error correcting

	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
Data Bits	Check Bits	% Increase	Check Bits	% Increase
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

SEC-DEC Example Requires One Extra Check Bit

Single Error Correction:

Word | With Even Parity



Single Error Correction, Double Error Detection:

Word | With Even Parity | With Two Errors

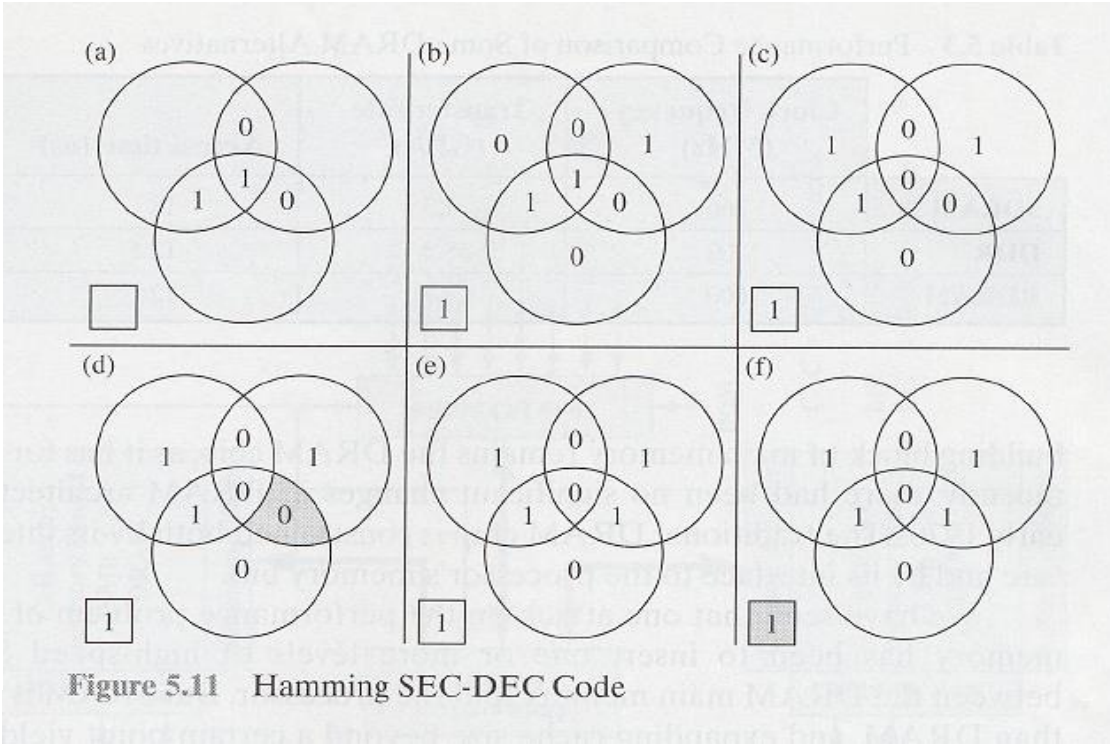


Figure 5.11 Hamming SEC-DEC Code

With Errors	Identifying Error	SEC Attempt	IS SEC Correct?	Extra Bit Confirms DE
Với lỗi	Xác định lỗi	SEC Cố gắng	là SEC đúng?	Thêm bit xác nhận DE

Mã vừa được mô tả được gọi là mã sửa lỗi đơn (SEC).

Thông thường hơn, bộ nhớ bán dẫn được trang bị sửa lỗi đơn,mã phát hiện lỗi kép (SEC-DED). Như Bảng cho thấy, các mã như vậy yêu cầu thêm một bit so với mã SEC.

Hình minh họa cách mã như vậy hoạt động, một lần nữa với từ dữ liệu 4 bit. trình tự cho thấy rằng nếu có hai lỗi xảy ra (Hình 5.11c), quy trình kiểm tra sẽ diễn ra lạc lối (d) và làm trầm trọng thêm vấn đề bằng cách tạo ra lỗi thứ ba (e). Để vượt qua vấn đề, một bit thứ tám được thêm vào được thiết lập sao cho tổng số 1 trong sơ đồ là chẵn. Các bit chẵn lẻ thêm bắt lỗi (f).

Mã sửa lỗi giúp tăng cường độ tin cậy của bộ nhớ với chi phí là thêm phức tạp. Với tổ chức 1 bit trên mỗi chip, mã SEC-DED thường coi là đầy đủ. Ví dụ, các triển khai IBM 30xx đã sử dụng mã SECDED 8 bit cho mỗi 64 bit dữ liệu trong bộ nhớ chính. Vì vậy, kích thước của bộ nhớ chính thực sự lớn hơn khoảng 12% so với người dùng. Các máy tính VAX đã sử dụng 7 bit SEC-DED cho mỗi 32 bit bộ nhớ, cho chi phí hoạt động 22%. Một số DRAM hiện đại sử dụng 9 bit kiểm tra cho mỗi 128 bit dữ liệu, với chi phí 7% [SHAR97].

Increased word length for error correcting

	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
Data Bits	Check Bits	% Increase	Check Bits	% Increase
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

5.2. Bộ nhớ bán dẫn

- Phân loại bộ nhớ
- Tổ chức bộ nhớ
- Thiết kế và ghép nối mô đun nhớ

1. Phân loại

<table>
<tr><th>Kiểu bộ nhớ</th><th>Tiêu chuẩn</th><th>Khả năng xoá</th><th>Cơ chế ghi</th><th>Tính khả biến</th></tr>
<tr><td>Read Only Memory (ROM)</td><td rowspan="2">Bộ nhớ chỉ đọc</td><td rowspan="2">Không xoá được</td><td>Mặt nạ</td><td rowspan="5">Không khả biến</td></tr>
<tr><td>Programmable ROM (PROM)</td><td rowspan="4">Bằng điện</td></tr>
<tr><td>Erasable PROM (EPROM)</td><td rowspan="2">Bộ nhớ hầu như chỉ đọc</td><td>bằng tia cực tím, cả chip</td></tr>
<tr><td>Electrically Erasable PROM (EEPROM)</td><td>bằng điện, mức từng byte</td></tr>
<tr><td>Flash memory</td><td rowspan="2">Bộ nhớ đọc-ghi</td><td>bằng điện, từng khối</td></tr>
<tr><td>Random Access Memory (RAM)</td><td>bằng điện, mức từng byte</td><td>Bằng điện</td><td>Khả biến</td></tr>
</table>

ROM (Read Only Memory)

- Bộ nhớ không khả biến
- Lưu trữ các thông tin sau:
  - Thư viện các chương trình con
  - Các chương trình điều khiển hệ thống (BIOS)
  - Các bảng chức năng
  - Vi chương trình

Các kiểu ROM

- ROM mặt nạ:
  - Thông tin được ghi khi sản xuất
  - Rất đắt
- PROM (Programmable ROM)
  - Cần thiết bị chuyên dụng để ghi bằng chương trình => chỉ ghi được một lần
- EPROM (Erasable PROM)
  - Cần thiết bị chuyên dụng để ghi bằng chương trình => ghi được nhiều lần
  - Trước khi ghi lại, xóa bằng tia cực tím

Các kiểu ROM (tiếp)

- EEPROM (Electrically Erasable PROM)
  - Có thể ghi theo từng byte
  - Xóa bằng điện
- Flash memory (Bộ nhớ cực nhanh)
  - Ghi theo khối
  - Xóa bằng điện

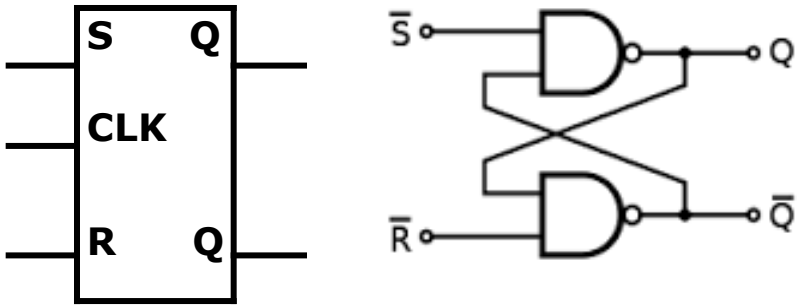
RAM (Random Access Memory)

- Bộ nhớ đọc-ghi (Read/Write Memory)
- Khả biến
- Lưu trữ thông tin tạm thời
- Có hai loại: SRAM và DRAM
  (Static and Dynamic)

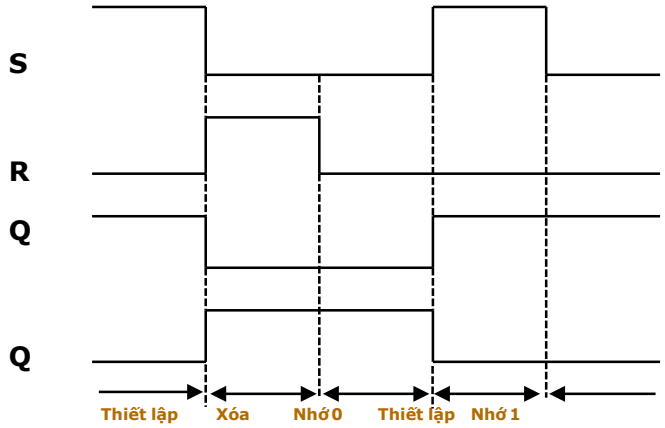
SRAM (Static) – RAM tĩnh

- Các bit được lưu trữ bằng các Flip-Flop

  =>thông tin ổn định
- Cấu trúc phức tạp
- Dung lượng chip nhỏ
- Tốc độ nhanh
- Đắt tiền
- Dùng làm bộ nhớ cache



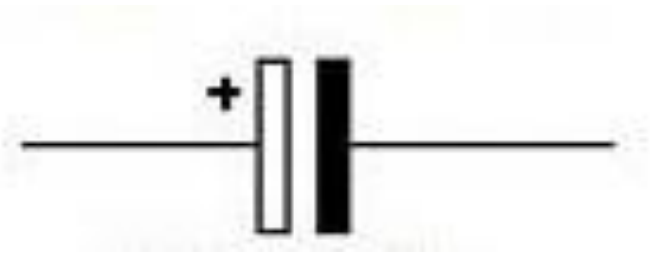
Flip Flop RS



Biểu đồ thời gian

DRAM (Dynamic) – RAM động

- Các bit được lưu trữ trên tụ điện → cần phải có mạch làm tươi
  - Mất dữ liệu sau một khoảng thời gian xác định dù vẫn được cấp điện
    - cần phải ghi lại dữ liệu đang lưu trữ
    - dùng mạch làm tươi (refresh) trên vi mạch nhớ
- Cấu trúc đơn giản
- Dung lượng lớn
- Tốc độ chậm hơn
- Rẻ tiền hơn
- Dùng làm bộ nhớ chính



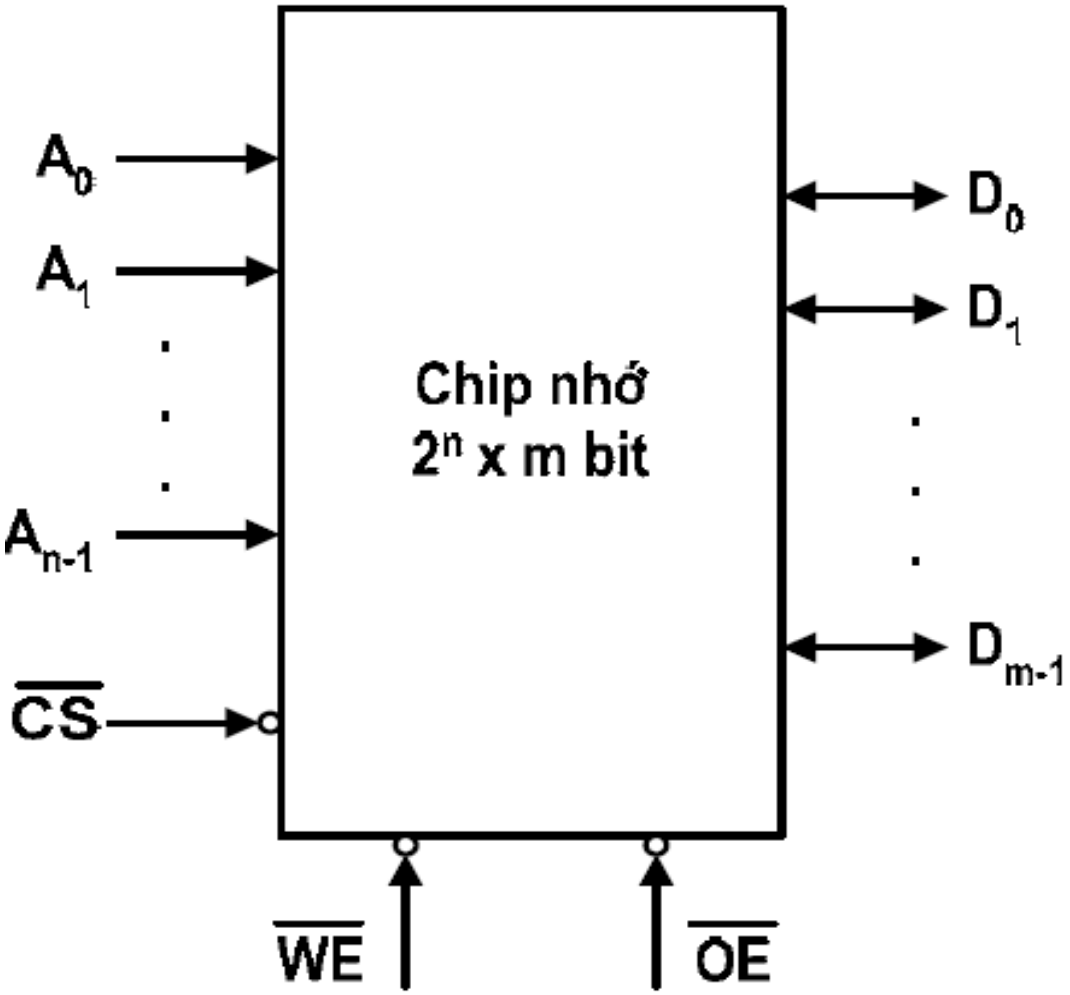
Các DRAM tiên tiến

- Enhanced DRAM
- Cache DRAM
- RAMDAC (RAM Digital-Analog Converter)
  - Dùng cho video RAM
- SDRAM (Synchronous DRAM)
  - Đồng bộ với system clock
  - Truyền dữ liệu theo khối

Các DRAM tiên tiến (tiếp)

- DDR-SDRAM (Double Data Rate-SDRAM)
  - Gởi dữ liệu 2 lần trong 1 chu kỳ clock
- DDR2 – Dual-Channel DDR
  - Dùng hai kênh bộ nhớ
- RDRAM (Rambus DRAM)
  - Dùng với Pentium-4, Itanium
  - Tốc độ clock cao, từ 400 Mhz
- SLDRAM (Synchronous-Link DRAM)
  - 64 bit dữ liệu
  - Gởi dữ liệu 2 lần trong 1 chu kỳ

2. Tổ chức của chip nhớ



Các tín hiệu của chip nhớ

- Các đường địa chỉ: $A_{n-1} \div A_0 \rightarrow$ có 2^n từ nhớ
- Các đường dữ liệu: $D_{m-1} \div D_0 \rightarrow$ độ dài từ nhớ = m bit
- Dung lượng chip nhớ = 2^n x m bit
- Các đường điều khiển:
  - Tín hiệu chọn chip CS (Chip Select)
  - Tín hiệu điều khiển đọc OE (Output Enable)
  - Tín hiệu điều khiển ghi WE (Write Enable)

(Các tín hiệu điều khiển tích cực với mức 0)

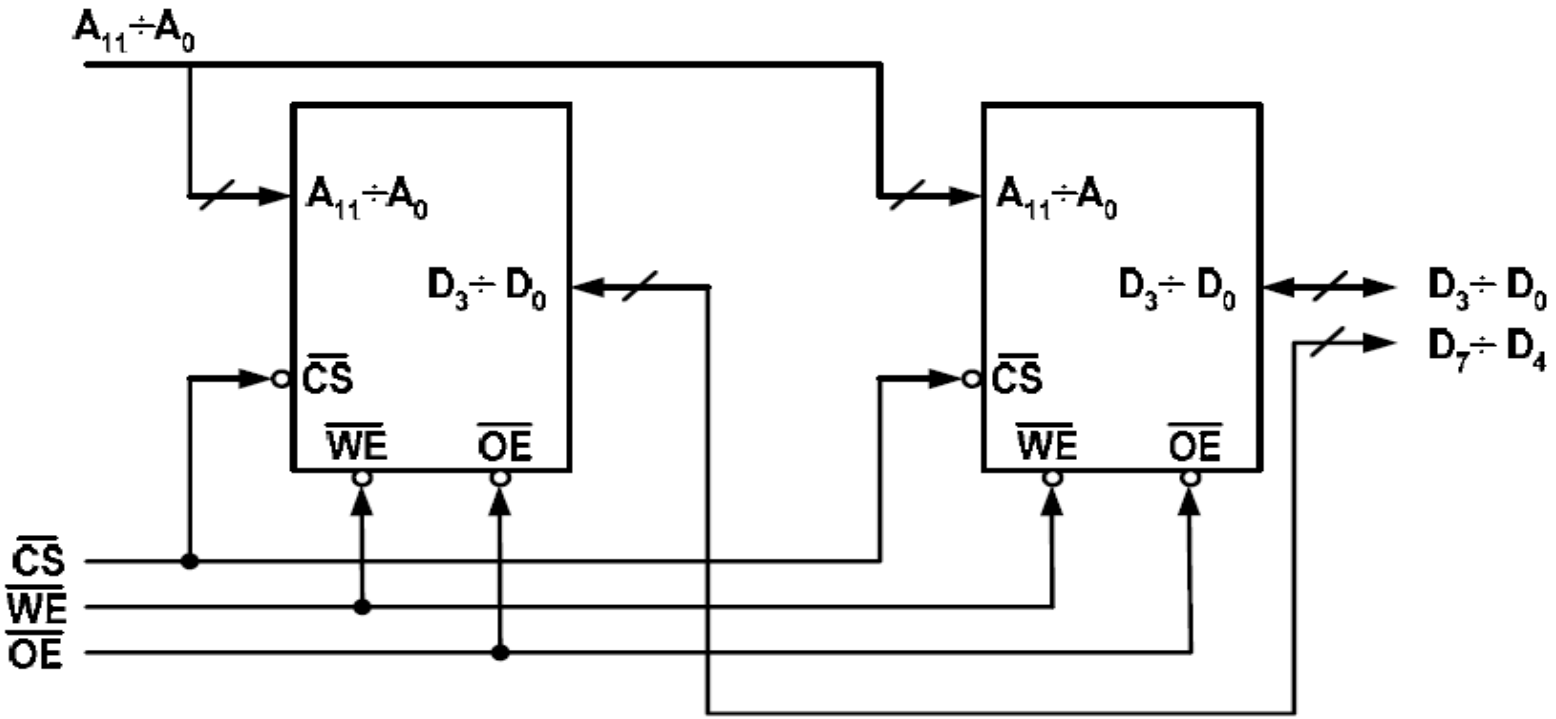
3. Thiết kế và ghép nối mô-đun nhớ bán dẫn

- Dung lượng chip nhớ 2^n x m bit
- Cần thiết kế để tăng dung lượng:
  - Thiết kế tăng độ dài từ nhớ
  - Thiết kế tăng số lượng từ nhớ
  - Thiết kế kết hợp

Tăng độ dài từ nhớ

- VD1:
  - Cho chip nhớ SRAM 4K x 4 bit
  - Thiết kế mô-đun nhớ 4K x 8 bit
- Giải:
  - Dung lượng chip nhớ = 2^{12} x 4 bit
  - chip nhớ có:
    - 12 chân địa chỉ
    - 4 chân dữ liệu
  - mô-đun nhớ cần có:
    - 12 chân địa chỉ
    - 8 chân dữ liệu

Ví dụ tăng độ dài từ nhớ



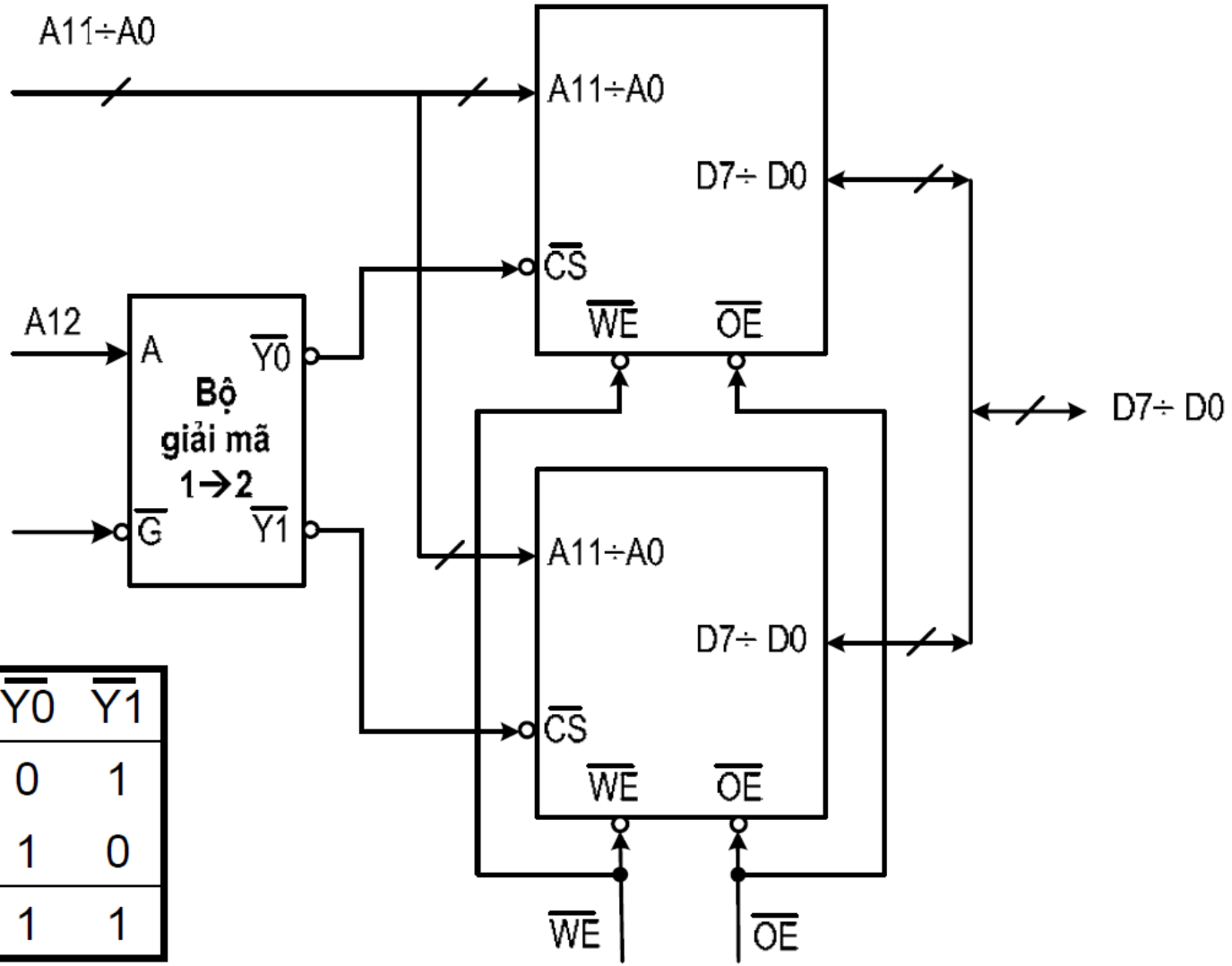
Bài toán tăng độ dài từ nhớ tổng quát

- Cho chip nhớ 2^n x mbit
- Thiết kế mô-đun nhớ 2^n x (k.m) bit
- Dùng k chip nhớ

Tăng số lượng từ nhớ

- VD2:
  - Cho chip nhớ SRAM 4K x 8 bit
  - Thiết kế mô-đun nhớ 8K x 8 bit
- Giải:
  - Dung lượng chip nhớ = 2^{12} x 8 bit
  - chip nhớ có:
    - 12 chân địa chỉ
    - 8 chân dữ liệu
  - Dung lượng mô-đun nhớ = 2^{13} x 8 bit
    - 13 chân địa chỉ
    - 8 chân dữ liệu

Tăng số lượng từ nhớ

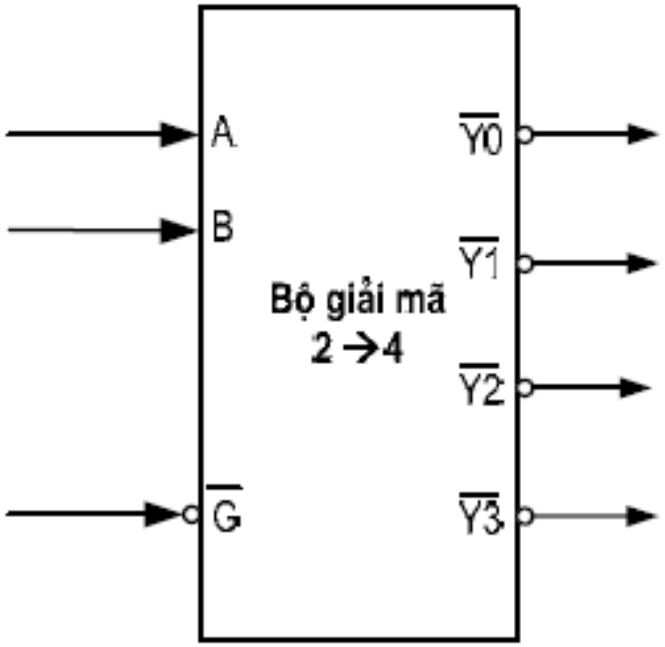


$\overline{G}$	A	$\overline{Y0}$	$\overline{Y1}$
0	0	0	1
0	1	1	0
1	x	1	1

Bài tập

- Tăng số lượng từ gấp 4 lần:
  - Cho chip nhớ SRAM 4K x 8 bit
  - Thiết kế mô-đun nhớ 16K x 8 bit
- Tăng số lượng từ gấp 8 lần:
  - Cho chip nhớ SRAM 4K x 8 bit
  - Thiết kế mô-đun nhớ 32K x 8 bit
- Thiết kế kết hợp:
  - Cho chip nhớ SRAM 4K x 4 bit
  - Thiết kế mô-đun nhớ 8K x 8 bit

Bộ giải mã 2->4



$\overline{G}$	B	A	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1