# CHƯƠNG 7: BỘ NHỚ BÁN DẪN

- **THUẬT NGỮ**
- **ĐẠI CƯƠNG VỀ VẬN HÀNH CỦA BỘ NHỚ**
  - Các tác vụ và các nhóm chân của IC nhớ
  - Giao tiếp với CPU
- **CÁC LOẠI BỘ NHỚ BÁN DẪN**
  - ROM
  - PLD
  - RAM
- **MỞ RỘNG BỘ NHỚ**
  - Mở rộng độ dài từ
  - Mở rộng vị trí nhớ
  - Mở rộng dung lượng nhớ

---

Tính ưu việt chủ yếu của các hệ thống số so với hệ thống tương tự là khả năng lưu trữ một lượng lớn thông tin số và dữ liệu trong những khoảng thời gian nhất định. Khả năng nhớ này là điều làm cho hệ thống số trở thành đa năng và có thể thích hợp với nhiều tình huống. Thí dụ trong một máy tính số, bộ nhớ trong chứa những lệnh mà theo đó máy tính có thể hoàn tất công việc của mình với sự tham gia ít nhất của con người.

Bộ nhớ bán dẫn được sử dụng làm **bộ nhớ chính** trong các máy tính nhờ vào khả năng thỏa mãn tốc độ truy xuất dữ liệu của bộ xử lý trung tâm (CPU).

Chúng ta đã quá quen thuộc với Fliflop, một linh kiện điện tử có tính nhớ. Chúng ta cũng đã thấy một nhóm các FF họp thành thanh ghi để lưu trữ và dịch chuyển thông tin như thế nào. Các FF chính là các phần tử nhớ tốc độ cao được dùng rất nhiều trong việc điều hành bên trong máy tính, nơi mà dữ liệu dịch chuyển liên tục từ nơi này đến nơi khác.

Tiến bộ trong công nghệ chế tạo LSI và VLSI cho phép kết hợp một lượng lớn FF trong một chip tạo thành các bộ nhớ với các dạng khác nhau. Những bộ nhớ bán dẫn với công nghệ chế tạo transistor lưỡng cực (BJT) và MOS là những bộ nhớ nhanh nhất và giá thành của nó liên tục giảm khi các công nghệ LSI và VLSI ngày càng được cải tiến.

Dữ liệu số cũng có thể được lưu trữ dưới dạng điện tích của tụ điện, và một loại phần tử nhớ bán dẫn rất quan trọng đã dùng nguyên tắc này để lưu trữ dữ liệu với mật độ cao nhưng tiêu thụ một nguồn điện năng rất thấp.

Bộ nhớ bán dẫn được dùng như là **bộ nhớ trong** chính của máy tính, nơi mà việc vận hành nhanh được xem như ưu tiên hàng đầu và cũng là nơi mà tất cả dữ liệu của chương trình lưu chuyển liên tục trong quá trình thực hiện một tác vụ do CPU yêu cầu.

Mặc dù bộ nhớ bán dẫn có tốc độ làm việc cao, rất phù hợp cho bộ nhớ trong, nhưng giá thành tính trên mỗi bit lưu trữ cao khiến cho nó không thể là loại thiết bị có tính chất lưu trữ khối (mass storage), là loại thiết bị có khả năng lưu trữ hàng tỉ bit mà không cần cung cấp năng lượng và được dùng như là **bộ nhớ ngoài** (đĩa từ , băng từ , CD ROM . . .). Tốc độ xử lý dữ liệu ở bộ nhớ ngoài tương đối chậm nên khi máy tính làm việc thì dữ liệu từ bộ nhớ ngoài được chuyển vào bộ nhớ trong.

Băng từ và đĩa từ là các thiết bị lưu trữ khối mà giá thành tính trên mỗi bit tương đối thấp. Một loại bộ nhớ khối mới hơn là bộ nhớ **bọt từ** (**magnetic bubble memory**, MBM) là bộ nhớ điện tử dựa trên nguyên tắc từ có khả năng lưu trữ hàng triệu bit trong một chip. Với tốc độ tương đối chậm nó không được dùng như bộ nhớ trong.

Chương này nghiên cứu cấu tạo và tổ chức của các **bộ nhớ bán dẫn**.

# 7.1 Thuật ngữ liên quan đến bộ nhớ

Để tìm hiểu cấu tạo, hoạt động của bộ nhớ chúng ta bắt đầu với một số thuật ngữ liên quan đến bộ nhớ

**- Tế bào nhớ**: là linh kiện hay một mạch điện tử dùng để lưu trữ một bit đơn (0 hay 1). Thí dụ của một tế bào nhớ bao gồm: mạch FF, tụ được tích điện, một điểm trên băng từ hay đĩa từ. . . .

**- Từ nhớ** : là một nhóm các bit (tế bào) trong bộ nhớ dùng biểu diễn các lệnh hay dữ liệu dưới dạng một số nhị phân. Thí dụ một thanh ghi 8 FF là một phần tử nhớ lưu trữ từ 8 bit. Kích thước của từ nhớ trong các máy tính hiện đại có chiều dài từ 4 đến 64 bit.

**- Byte** : từ 8 bit, đây là kích thước thường dùng của từ nhớ trong các máy vi tính.

**- Dung lượng** : chỉ số lượng bit có thể lưu trữ trong bộ nhớ. Thí dụ bộ nhớ có khả năng lưu trữ 4.096 từ nhớ 20 bit, dung lượng của nó là 4096 x 20, mỗi 1024 ($=2^{10}$) từ nhớ được gọi là "1K", như vậy 4096 x 20 = 4K x 20. Với dung lượng lớn hơn ta dùng "1M" hay 1meg để chỉ $2^{20}$ = 1.048.576 từ nhớ.

**- Địa chỉ** : là số nhị phân dùng xác định vị trí của từ nhớ trong bộ nhớ. Mỗi từ nhớ được lưu trong bộ nhớ tại một địa chỉ duy nhất. Địa chỉ luôn luôn được biểu diễn bởi số nhị phân, tuy nhiên để thuận tiện người ta có thể dùng số hex hay thập phân, bát phân

**- Tác vụ đọc** : (*Read*, còn gọi là *fetch* ), một từ nhớ tại một vị trí nào đó trong bộ nhớ được truy xuất và chuyển sang một thiết bị khác.

**- Tác vụ viết** : (ghi, *Write*, còn gọi là *store* ), một từ mới được đặt vào một vị trí trong bộ nhớ, khi một từ mới được viết vào thì từ cũ mất đi.

**- Thời gian truy xuất** (access time) : số đo tốc độ hoạt động của bộ nhớ, ký hiệu $t_{ACC}$ Đó là thời gian cần để hoàn tất một tác vụ đọc. Chính xác đó là thời gian từ khi bộ nhớ nhận một địa chỉ mới cho tới lúc dữ liệu khả dụng ở ngã ra bộ nhớ

**- Bộ nhớ không vĩnh cữu** (volatile) : Bộ nhớ cần nguồn điện để lưu trữ thông tin. Khi ngắt điện, thông tin lưu trữ bị mất. Hầu hết bộ nhớ bán dẫn là loại không vĩnh cữu, trong khi bộ nhớ từ là loại vĩnh cữu (nonvolatile).

**- Bộ nhớ truy xuất ngẫu nhiên** (Random-Access Memory, **RAM**) : Khi cần truy xuất một địa chỉ ta tới ngay địa chỉ đó. Vậy thời gian đọc hay viết dữ liệu vào các vị trí nhớ khác nhau trong bộ nhớ không tùy thuộc vào vị trí nhớ. Nói cách khác, thời gian truy xuất như nhau đối với mọi vị trí nhớ. Hầu hết bộ nhớ bán dẫn và **nhẫn từ** (bộ nhớ trong của máy tính trước khi bộ nhớ bán dẫn ra đời) là loại truy xuất ngẫu nhiên.

**- Bộ nhớ truy xuất tuần tự** (Sequential-Access Memory, **SAM**) : Khi cần truy xuất một địa chỉ ta phải lướt qua các địa chỉ trước nó. Như vậy thời gian đọc và viết dữ liệu ở những vị trí khác nhau thì khác nhau. Những thí dụ của bộ nhớ này là băng từ, đĩa từ. Tốc độ làm việc của loại bộ nhớ này thường chậm so với bộ nhớ truy xuất ngẫu nhiên.

**- Bộ nhớ đọc/viết** (Read/Write Memory, **RWM**) : Bộ nhớ có thể viết vào và đọc ra.

**- Bộ nhớ chỉ đọc** (Read-Only Memory, **ROM**): là bộ nhớ mà tỉ lệ tác vụ đọc trên tác vụ ghi rất lớn. Về mặt kỹ thuật, một ROM có thể được ghi chỉ một lần ở nơi sản xuất và sau đó thông tin chỉ có thể được đọc ra từ bộ nhớ. Có loại ROM có thể được ghi nhiều lần nhưng tác vụ ghi khá phức tạp hơn là tác vụ đọc. ROM thuộc loại bộ nhớ vĩnh cữu và dữ liệu được lưu giữ khi đã cắt nguồn điện.

**- Bộ nhớ tĩnh** (Static Memory Devices) : là bộ nhớ bán dẫn trong đó dữ liệu đã lưu trữ được duy trì cho đến khi nào còn nguồn nuôi.

**- Bộ nhớ động** (Dynamic Memory Devices) : là bộ nhớ bán dẫn trong đó dữ liệu đã lưu trữ muốn tồn tại phải được **ghi lại** theo chu kỳ. Tác vụ ghi lại được gọi là **làm tươi** (refresh).

**- Bộ nhớ trong** (Internal Memory) : Chỉ bộ nhớ chính của máy tính. Nó lưu trữ các lệnh và dữ liệu mà CPU dùng thường xuyên khi hoạt động.

**- Bộ nhớ khối** (Mass Memory): Còn gọi là bộ nhớ phụ, nó chứa một lượng thông tin rất lớn ở bên ngoài máy tính. Tốc độ truy xuất trên bộ nhớ này thường chậm và nó thuộc loại vĩnh cữu.

# 7.2 Đại cương về vận hành của bộ nhớ

## 7.2.1 Các tác vụ và các nhóm chân của một IC nhớ

Mặc dù mỗi loại bộ nhớ có hoạt động bên trong khác nhau, nhưng chúng có chung một số nguyên tắc vận hành mà chúng ta có thể tìm hiểu sơ lược trước khi đi vào nghiên cứu từng loại bộ nhớ.

Mỗi hệ thống nhớ luôn có một số yêu cầu ở các ngã vào và ra để hoàn thành một số tác vụ:

- Chọn địa chỉ trong bộ nhớ để truy xuất (đọc hoặc viết)
- Chọn tác vụ đọc hoặc viết để thực hiện
- Cung cấp dữ liệu để lưu vào bộ nhớ trong tác vụ viết
- Gửi dữ liệu ra từ bộ nhớ trong tác vụ đọc
- Cho phép (Enable) (hay Không, Disable) bộ nhớ đáp ứng (hay không) đối với lệnh đọc/ghi ở địa chỉ đã gọi đến.

Từ các tác vụ kể trên, ta có thể hình dung mỗi IC nhớ có một số ngã vào ra như sau:

**- Ngã vào địa chỉ** : mỗi vị trí nhớ xác định bởi một địa chỉ duy nhất, khi cần đọc dữ liệu ra hoặc ghi dữ liệu vào ta phải tác động vào chân địa chỉ của vị trí nhớ đó. Một IC có n chân địa chỉ sẽ có $2^n$ vị trí nhớ. Ký hiệu các chân địa chỉ là $A_0$ đến $A_{n-1}$ Một IC có 10 chân địa chỉ sẽ có 1024 (1K) vị trí nhơ.

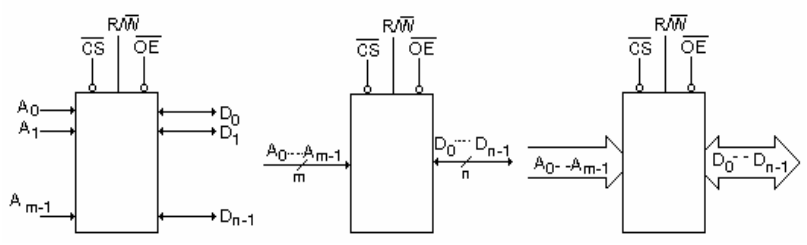
**- Ngã vào/ra dữ liệu:** Các chân dữ liệu là các ngã vào/ra, nghĩa là dữ liệu luôn được xử lý theo hai chiều. Thường thì dữ liệu vào/ra chung trên một chân nên các ngã này thuộc loại ngã ra 3 trạng thái. Số chân địa chỉ và dữ liệu của một IC xác định dung lượng nhớ của IC đó. Thí dụ một IC nhớ có 10 chân địa chỉ và 8 chân dữ liệu thì dung lượng nhớ của IC đó là 1Kx8 (8K bit hoặc 1K Byte).

**- Các ngã vào điều khiển:** Mỗi khi IC nhớ được chọn hoặc có yêu cầu xuất nhập dữ liệu các chân tương ứng sẽ được tác động. Ta có thể kể ra một số ngã vào điều khiển:

* $\overline{CS}$: Chip select - Chọn chip - Khi chân này xuống thấp IC được chọn
* $\overline{CE}$: Chip Enable - Cho phép chip - Chức năng như chân $\overline{CS}$
* $\overline{OE}$: Output Enable - Cho phép xuất - Dùng khi đọc dữ liệu
* $R/\overline{W}$ : Read/Write - Đọc/Viết - Cho phép **Đọc** dữ liệu ra khi ở mức cao và **Ghi** dữ liệu vào khi ở mức thấp
* $\overline{CAS}$: Column Address Strobe - Chốt địa chỉ cột
* $\overline{RAS}$: Row Address Strobe - Chốt địa chỉ hàng.

Trong trường hợp chip nhớ có dung lượng lớn, để giảm kích thước của mạch giải mã địa chỉ bên trong IC, người ta chia số chân ra làm 2: địa chỉ hàng và địa chỉ cột. Như vậy phải dùng 2 mạch giải mã địa chỉ nhưng mỗi mạch nhỏ hơn rất nhiều. Thí dụ với 10 chân địa chỉ, thay vì dùng 1 mạch giải mã 10 đường sang 1024 đường, người ta dùng 2 mạch giải mã 5 đường sang 32 đường, hai mạch này rất đơn giản so với một mạch kia. Một vị trí nhớ bây giờ có 2 địa chỉ : hàng và cột, dĩ nhiên muốn truy xuất một vị trí nhớ phải có đủ 2 địa chỉ nhờ 2 tín hiệu $\overline{RAS}$ và $\overline{CAS}$.

(H 7.1) cho thấy cách vẽ các nhóm chân của IC nhớ (m chân địa chỉ và n chân dữ liệu). (H 7.1b) và (H 7.1c) vẽ các chân địa chỉ và dữ liệu dưới dạng các Bus. (H 7.1b) được dùng trong các sơ đồ chi tiết và (H 7.1c) được dùng trong các sơ đồ khối.

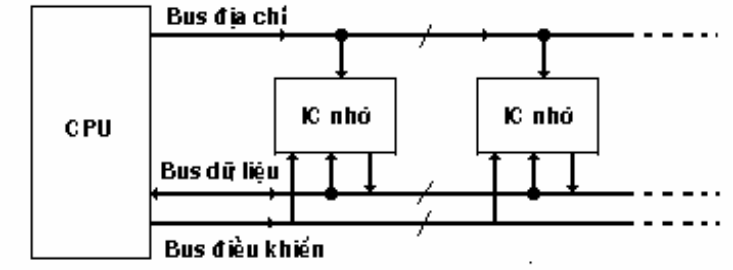



(a) (b) (c)

(H 7.1)

### 7.2.2 Giao tiếp giữa IC nhớ và bộ xử lý trung tâm (CPU)

Trong hệ thống mọi hoạt động có liên quan đến IC nhớ đều do bộ xử lý trung tâm (Central Processing Unit, CPU) quản lý. Giao tiếp giữa IC nhớ và CPU mô tả ở (H 7.2)




(H 7.2)

Một tác vụ có liên quan đến bộ nhớ được CPU thực hiện theo các bước:

- Đặt địa chỉ quan hệ lên bus địa chỉ.
- Đặt tín hiệu điều khiển lên bus điều khiển.
- Dữ liệu khả dụng xuất hiện trên bus dữ liệu, sẵn sàng để ghi vào hoặc đọc ra.

Để hoạt động của IC đồng bộ, các bước trên phải tuân thủ **giản đồ thời gian** của từng IC nhớ (sẽ đề cập đến khi xét các loại bộ nhớ)

## 7.3 Các loại bộ nhớ bán dẫn

Có 3 loại bộ nhớ bán dẫn :

- Bộ nhớ bán dẫn chỉ đọc : (Read Only Memory, ROM)
- Bộ nhớ truy xuất ngẫu nhiên : (Random Access Memory, RAM)

Thật ra ROM và RAM đều là loại bộ nhớ truy xuất ngẫu nhiên, nhưng RAM được giữ tên gọi này. Để phân biệt chính xác ROM và RAM ta có thể gọi ROM là **bộ nhớ chết** (nonvolatile, vĩnh cữu) và RAM là **bộ nhớ sống** (volatile, không vĩnh cữu) hoặc nếu coi ROM là **bộ nhớ chỉ đọc** thì RAM là **bộ nhớ đọc được - viết được** (Read-Write Memory)

- Thiết bị logic lập trình được : (Programmable Logic Devices, PLD) có thể nói điểm khác biệt giữa PLD với ROM và RAM là qui mô tích hợp của PLD thường không lớn như ROM và RAM và các tác vụ của PLD thì có phần hạn chế.

### 7.3.1 ROM (Read Only Memory)

Mặc dù có tên gọi như thế nhưng chúng ta phải hiểu là khi sử dụng ROM, tác vụ đọc được thực hiện rất nhiều lần so với tác vụ ghi. Thậm chí có loại ROM chỉ ghi một lần khi xuất xưởng.

Các tế bào nhớ hoặc từ nhớ trong ROM sắp xếp theo dạng ma trận mà mỗi phần tử chiếm một vị trí xác định bởi một địa chỉ cụ thể và nối với ngã ra một mạch giải mã địa chỉ

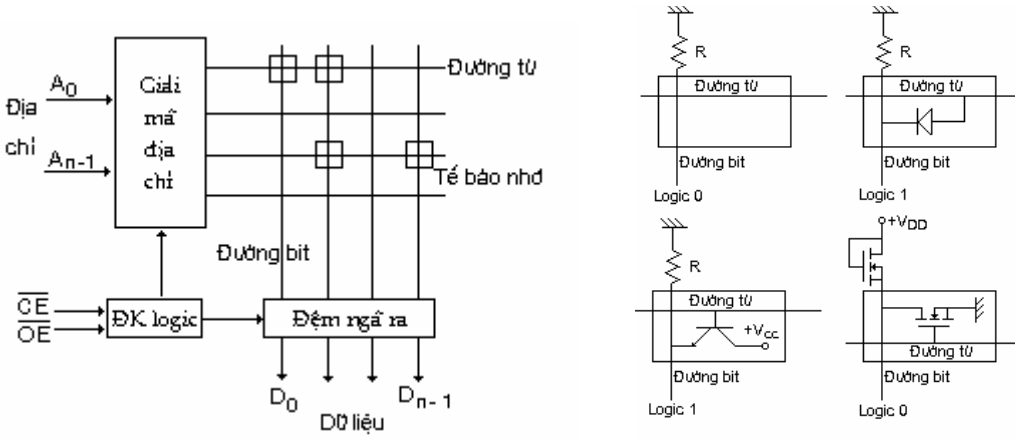
bên trong IC. Nếu mỗi vị trí chứa một tế bào nhớ ta nói ROM có **tổ chức bit** và mỗi vị trí là một từ nhớ ta có **tổ chức từ**.

Ngoài ra, để giảm mức độ cồng kềnh của mạch giải mã, mỗi vị trí nhớ có thể được xác định bởi 2 đường địa chỉ : đường địa chỉ hàng và đường địa chỉ cột và trong bộ nhớ có 2 mạch giải mã nhưng mỗi mạch có số ngã vào bằng 1/2 số đường địa chỉ của cả bộ nhớ.

### 7.3.1.1 ROM mặt nạ (Mask Programmed ROM, MROM)

Đây là loại ROM được chế tạo để thực hiện một công việc cụ thể như các bảng tính, bảng lượng giác , bảng logarit . . . . ngay sau khi xuất xưởng. Nói cách khác, các tế bào nhớ trong ma trận nhớ đã được tạo ra theo một chương trình đã xác định trước bằng phương pháp mặt nạ: đưa vào các linh kiện điện tử nối từ **đường từ** qua **đường bít** để tạo ra một giá trị bit và để trống cho giá trị bit ngược lại.

- (H 7.3) là mô hình của một MROM trong đó các ô vuông là nơi chứa (hay không) một linh kiện (diod, transistor BJT hay MOSFET) để tạo bit. Mỗi ngã ra của mạch giải mã địa chỉ gọi là đường từ và đường nối tế bào nhớ ra ngoài gọi là đường bit. Khi đường từ lên mức cao thì tế bào nhớ hoặc từ nhớ được chọn.

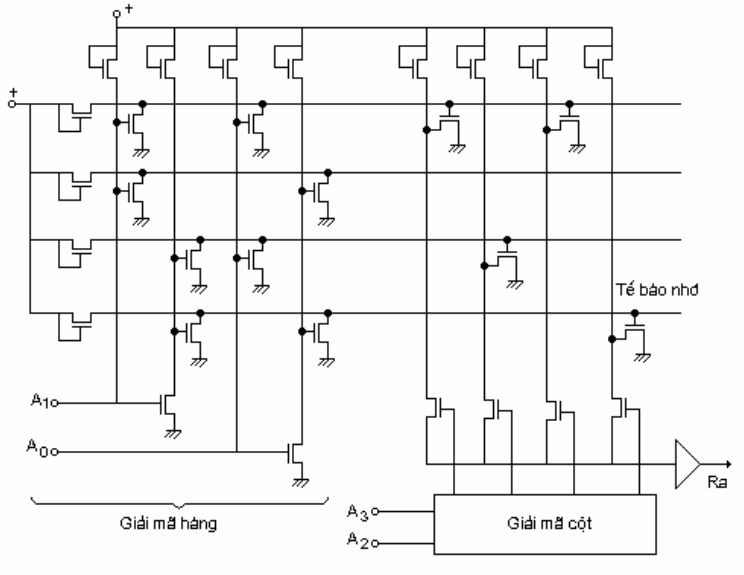



(H 7.3)

Nếu tế bào nhớ là Diod hoặc BJT thì sự hiện diện của linh kiện tương ứng với bit 1 (lúc này đường từ lên cao, Transsisstor hoặc diod dẫn, dòng điện qua điện trở tạo điện thế cao ở hai đầu điện trở) còn vị trí nhớ trống tương ứng với bit 0.

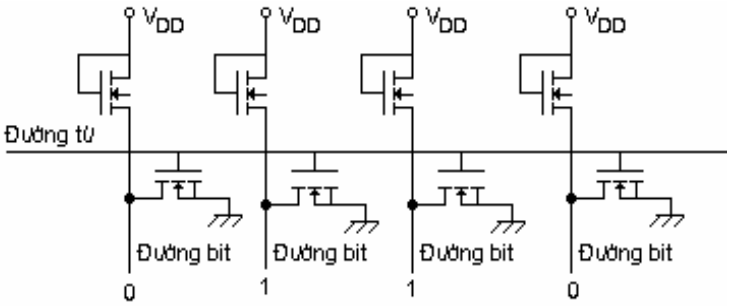
Đối với loại linh kiện MOSFET thì ngược lại, nghĩa là sự hiện diện của linh kiện tương ứng với bit 0 còn vị trí nhớ trống tương ứng với bit 1 (muốn có kết quả như loại BJT thì thêm ở ngã ra các cổng đảo).

(H 7.4) là một thí dụ bộ nhớ MROM có dung lượng 16x1 với các mạch giải mã hàng và cột (các mạch giải mã 2 đường sang 4 đường của hàng và cột đều dùng Transistor MOS và có cùng cấu trúc).




(H 7.4)

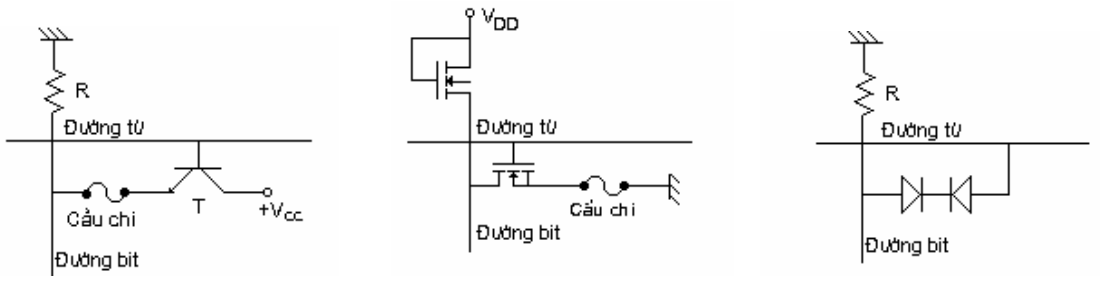
Trong thực tế, để đơn giản cho việc thực hiện, ở mỗi vị trí nhớ người ta đều cho vào một transistor MOS. Nhưng ở những vị trí ứng với bit 1 các transistor MOS được chế tạo với lớp $SiO_2$ dầy hơn làm tăng điện thế ngưỡng của nó lên, kết quả là transistor MOS này luôn luôn không dẫn điện (H 7.5), Các transistor khác dẫn điện bình thường.




(H 7.5)

### 7.3.1.2 ROM lập trình được (Programmable ROM, PROM)

Có cấu tạo giống MROM nhưng ở mỗi vị trí nhớ đều có linh kiện nối với cầu chì. Như vậy khi xuất xưởng các ROM này đều chứa cùng một loại bit (gọi là ROM trắng), lúc sử dụng người lập trình thay đổi các bit mong muốn bằng cách phá vỡ cầu chì ở các vị trí tương ứng với bit đó. Một khi cầu chì đã bị phá vỡ thì không thể nối lại được do đó loại ROM này cho phép lập trình một lần duy nhất để sử dụng, nếu bị lỗi không thể sửa chữa được (H 7.6).

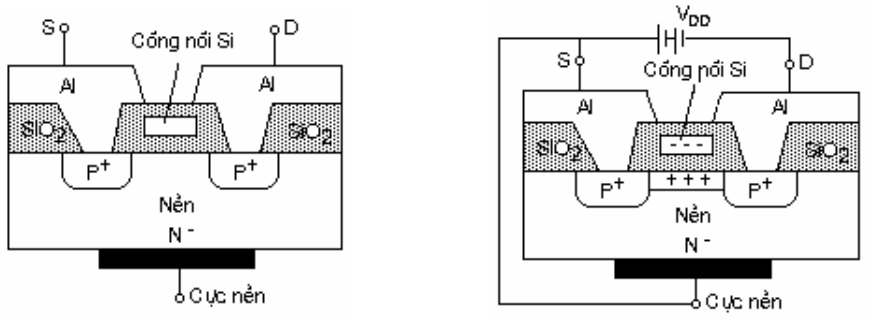



(H 7.6)

Người ta có thể dùng 2 diod mắc ngược chiều nhau, mạch không dẫn điện, để tạo bit 0, khi lập trình thì một diod bị phá hỏng tạo mạch nối tắt, diod còn lại dẫn điện cho bit 1

### 7.3.1.3 ROM lập trình được, xóa được bằng tia U.V. (Ultra Violet Erasable Programmable ROM, U.V. EPROM)

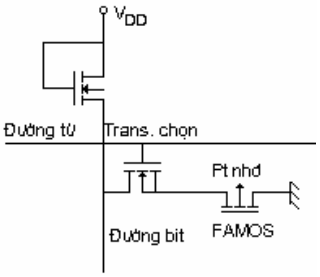
Đây là loại ROM rất tiện cho người sử dụng vì có thể dùng được nhiều lần bằng cách xóa và nạp lại. Cấu tạo của tế bào nhớ của U.V. EPROM dựa vào một transistor MOS có cấu tạo đặc biệt gọi là FAMOS (Floating Gate Avalanche Injection MOS)




(H 7.7)

Trên nền chất bán dẫn N pha loãng, tạo 2 vùng P pha đậm ($P^+$) nối ra ngoài cho 2 cực S (Source) và D (Drain). Trong lớp cách điện $SiO_2$ giữa 2 cực người ta cho vào một thỏi Silicon không nối với bên ngoài và được gọi là **cổng nổi**. Khi nguồn $V_{DD}$, phân cực ngược giữa cực nền và Drain còn nhỏ, transistor không dẫn, nhưng nếu tăng $V_{DD}$ đủ lớn, hiện tượng thác đổ (avalanche) xảy ra, electron đủ năng lượng chui qua lớp cách điện tới bám vào cổng nổi. Do hiện tượng cảm ứng, một điện lộ P hình thành nối hai vùng bán dẫn $P^+$ , transistor trở nên dẫn điện. Khi cắt nguồn, transistor tiếp tục dẫn điện vì electron không thể trở về để tái hợp với lỗ trống.

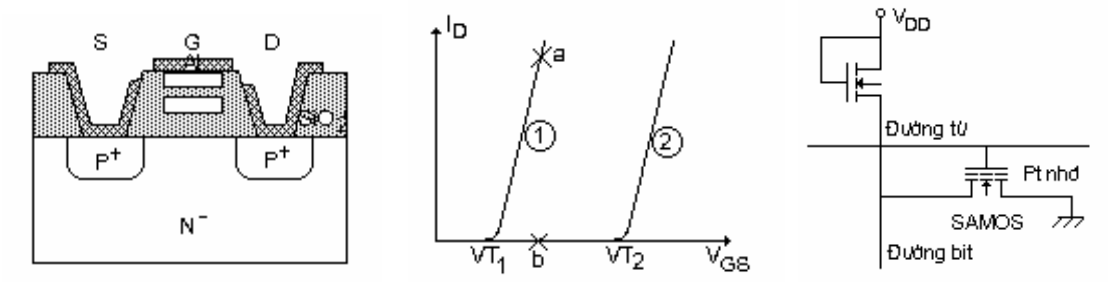
Để xóa EPROM, người ta chiếu tia U.V. vào các tế bào trong một khoảng thời gian xác định để electron trên cổng nổi nhận đủ năng lượng vượt qua lớp cách điện trở về vùng nền tái hợp với lỗ trống xóa điện lộ P và transistor trở về trạng thái không dẫn ban đầu.




(H 7.8)

Mỗi tế bào nhớ EPROM gồm một transistor FAMOS nối tiếp với một transistor MOS khác mà ta gọi là transistor chọn, như vậy vai trò của FAMOS giống như là một cầu chì nhưng có thể phục hồi được.

Để loại bỏ transistor chọn người ta dùng transistor SAMOS (Stacked Gate Avalanche Injection MOS) có cấu tạo tương tự transistor MOS nhưng có đến 2 cổng nằm chồng lên nhau, một được nối ra cực Gate và một để nổi. Khi cổng nổi tích điện sẽ làm gia tăng điện thế thềm khiến transistor trở nên khó dẫn điện hơn. Như vậy nếu ta chọn điện thế $V_c$ ở khoảng giữa $VT_1$ và $VT_2$ là 2 giá trị điện thế thềm tương ứng với 2 trạng thái của transistor ($VT_1<V_c<VT_2$) thì các transistor không được lập trình (không có lớp electron ở cổng nổi) sẽ dẫn còn các transistor được lập trình sẽ không dẫn.

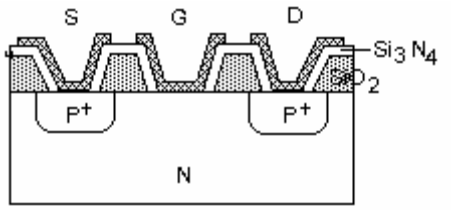



(H 7.9)

Điểm bất tiện của U.V EPROM là cần thiết bị xóa đặc biệt phát tia U.V. và mỗi lần xóa tất cả tế bào nhớ trong một IC nhớ đều bị xóa. Như vậy người sử dụng phải nạp lại toàn bộ chương trình

### 7.3.1.4 ROM lập trình được và xóa được bằng xung điện (Electrically Erasable PROM, EEPROM hay Electrically Alterable PROM, EAPROM)

Đây là loại ROM lập trình được và xóa được nhờ xung điện và đặc biệt là có thể xóa để sửa trên từng byte. Các tế bào nhớ EEPROM sử dụng transistor MNOS (Metal Nitride Oxide Semiconductor) có cấu tạo như (H 7.10).




(H 7.10)

Giữa lớp kim loại nối ra các cực và lớp $SiO_2$ là một lớp mỏng chất Nitrua Silic ($Si_3N_4$) - từ 40nm đến 650nm - Dữ liệu được nạp bằng cách áp một điện thế dương giữa cực G và S (khoảng 20 đến 25V trong 100ms). Do sự khác biệt về độ dẫn điện, electron tích trên bề mặt giữa 2 lớp $SiO_2$ và $Si_3N_4$ , các electron này tồn tại khi đã ngắt nguồn và làm thay đổi trạng thái dẫn điện của transistor. Bây giờ nếu áp một điện thế âm giữa cực G và S ta sẽ được một lớp điện tích trái dấu với trường hợp trước. Như vậy hai trạng thái khác nhau của Transistor có thể thiết lập được bởi hai điện thế ngược chiều nhau và như vậy các tế bào nhớ được ghi và xóa với 2 xung điện trái dấu nhau.

### 7.3.1.5 FLASH ROM

EPROM là loại nonvolatile, có tốc độ truy xuất nhanh (khoảng 120ns), mật độ tích hợp cao, giá thành rẻ tuy nhiên để xóa và nạp lại phải dùng thiết bị đặc biệt và lấy ra khỏi mạch.

EEPROM cũng nonvolatile, cũng có tốc độ truy xuất nhanh, cho phép xóa và nạp lại ngay trong mạch trên từng byte nhưng có mật độ tích hợp thấp và giá thành cao hơn EPROM.

Bộ nhớ FLASH ROM tận dụng được các ưu điểm của hai loại ROM nói trên, nghĩa là có tốc độ truy xuất nhanh, có mật độ tích hợp cao nhưng giá thành thấp.

Hầu hết các FLASH ROM sử dụng cách xóa đồng thời cả khối dữ liệu nhưng rất nhanh (hàng trăm ms so với 20 min của U.V. EPROM). Những FLASH ROM thế hệ mới cho phép xóa từng sector (512 byte) thậm chí từng vị trí nhớ mà không cần lấy IC ra khỏi mạch. FLASH ROM có thời gian ghi khoảng 10μs/byte so với 100 μs đối với EPROM và 5 ms đối với EEPROM

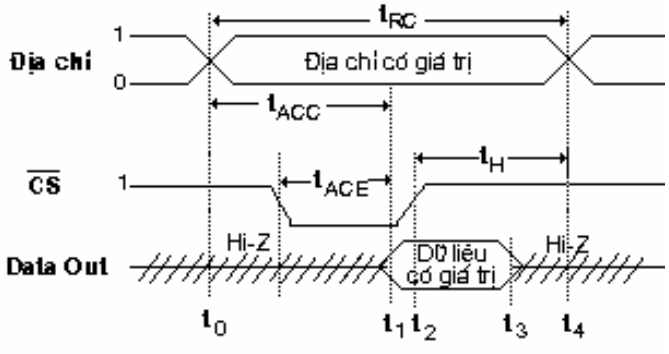
### 7.3.1.6 Giản đồ thời gian của ROM

Ngoại trừ MROM chỉ dùng ở chế độ đọc, các loại ROM khác đều sử dụng ở hai chế độ đọc và nạp chương trình.

Như vậy ta có hai loại giản đồ thời gian: Giản đồ thời gian đọc và giản đồ thời gian nạp trình.

(H 7.11) là giản đồ thời gian tiêu biểu cho một chu kỳ đọc của ROM.

Các giá trị địa chỉ, các tín hiệu R/$\overline{W}$ và $\overline{CS}$ được cấp từ CPU khi cần thực hiện tác vụ đọc dữ liệu tại một địa chỉ nào đó. Thời gian để thực hiện một tác vụ đọc gọi là chu kỳ đọc $t_{RC}$. Trong một chu kỳ đọc có thể kể một số thời gian sau:



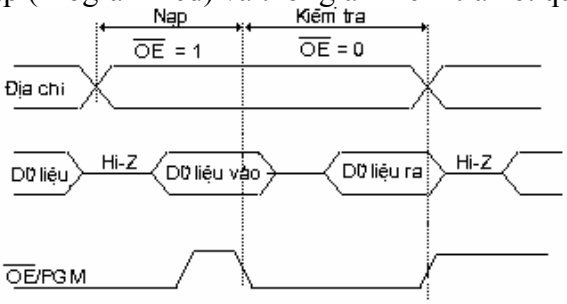

(H 7.11)

- $t_{ACC}$: Address Access time: Thời gian truy xuất địa chỉ: Thời gian tối đa từ lúc CPU đặt địa chỉ lên bus địa chỉ đến lúc dữ liệu có giá trị trên bus dữ liệu. Đối với ROM dùng BJT thời gian này khoảng từ 30 ns đến 90 ns, còn loại MOS thì từ 200 ns đến 900 ns.

- $t_{ACS}$ ($t_{ACE}$): Chip select (enable) access time: Thời gian thâm nhập chọn chip: Thời gian tối đa từ lúc tín hiệu $\overline{CS}$ được đặt lên bus điều khiển đến lúc dữ liệu có giá trị trên bus dữ liệu. ROM BJT khoảng 20 ns , MOS 100 ns

- $t_H$ (Hold time): Thời gian dữ liệu còn tồn tại trên bus dữ liệu kê từ lúc tín hiệu $\overline{CS}$ hết hiệu lực

(H 7.12) là giản đồ thời gian của một chu kỳ nạp dữ liệu cho EPROM. Một chu kỳ nạp liệu bao gồm thời gian nạp (Programmed) và thời gian kiểm tra kết quả (Verify)




(H 7.12)

## 7.3.2 Thiết bị logic lập trình được (Programmable logic devices, PLD)

Là tên gọi chung các thiết bị có tính chất nhớ và có thể lập trình để thực hiện một công việc cụ thể nào đó

Trong công việc thiết kế các hệ thống, đôi khi người ta cần một số mạch tổ hợp để thực hiện một hàm logic nào đó. Việc sử dụng mạch này có thể lặp lại thường xuyên và sự thay đổi một tham số của hàm có thể phải được thực hiện để thỏa mãn yêu cầu của việc thiết kế. Nếu phải thiết kế từ các cổng logic cơ bản thì mạch sẽ rất cồng kềnh, tốn kém mạch in,

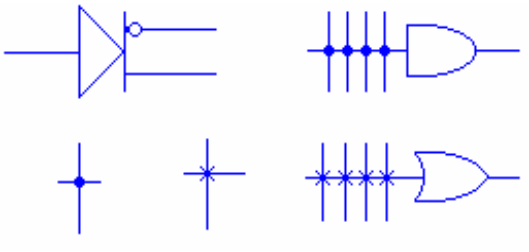
dây nối nhiều, kết quả là độ tin cậy không cao. Như vậy, sẽ rất tiện lợi nếu các mạch này được chế tạo sẵn và người sử dụng có thể chỉ tác động vào để làm thay đổi một phần nào chức năng của mạch bằng cách lập trình. Đó là ý tưởng cơ sở cho sự ra đời của thiết bị logic lập trình được. Các thiết bị này có thể được xếp loại như bộ nhớ và gồm các loại: PROM, PAL (Programmable Array Logic) và PLA (Programmable Logic Array).

Trước nhất, chúng ta xét qua một số qui ước trong cách biểu diễn các phần tử của PLD

Một biến trong các hàm thường xuất hiện ở dạng nguyên và đảo của nó nên chúng ta dùng ký hiêu đệm và đảo chung trong một cổng có 2 ngã ra.

Một **nối chết**, còn gọi là **nối cứng** (không thay đổi được) được vẽ bởi một chấm đậm (**.**) và một **nối sống**, còn gọi là **nối mềm** (dùng lập trình) bởi một dấu (x). Nối sống thực chất là một cầu chì, khi lập trình thì được phá bỏ.

Một cổng nhiều ngã vào thay thế bởi một ngã vào duy nhất với nhiều mối nối (H 7.13).



(H 7.13)

Chúng ta chỉ lấy thí dụ với mạch tương đối đơn giản để thấy được cấu tạo của các PLD, đó là các PLD chỉ thực hiện được 4 hàm mỗi hàm gồm 4 biến, như vậy mạch gồm 4 ngã vào và 4 ngã ra. Trên thực tế số hàm và biến của một PLD rất lớn.

### 7.3.2.1 PROM

(H 7.14 ) là cấu tạo PROM có 4 ngã vào và 4 ngã ra.

Có tất cả 16 cổng AND có 4 ngã vào được nối chết với các ngã ra đảo và không đảo của các biến vào, ngã ra các cổng AND là 16 tổ hợp của 4 biến (Gọi là đường tích)
Các cổng OR có 16 ngã vào được nối sống để thực hiện hàm tổng (đường tổng). Như vậy với PROM việc lập trình thực hiện ở các đường tổng.

Thí dụ dùng PROM này để tạo các hàm sau:

$O_1 = A + \overline{D}B + \overline{D}C \quad O_2 = \overline{D}\,\overline{C}BA + DC\overline{B}\,\overline{A} \quad O_3 = C\overline{B}A \quad O_4 = BA + \overline{D}\,\overline{C}$

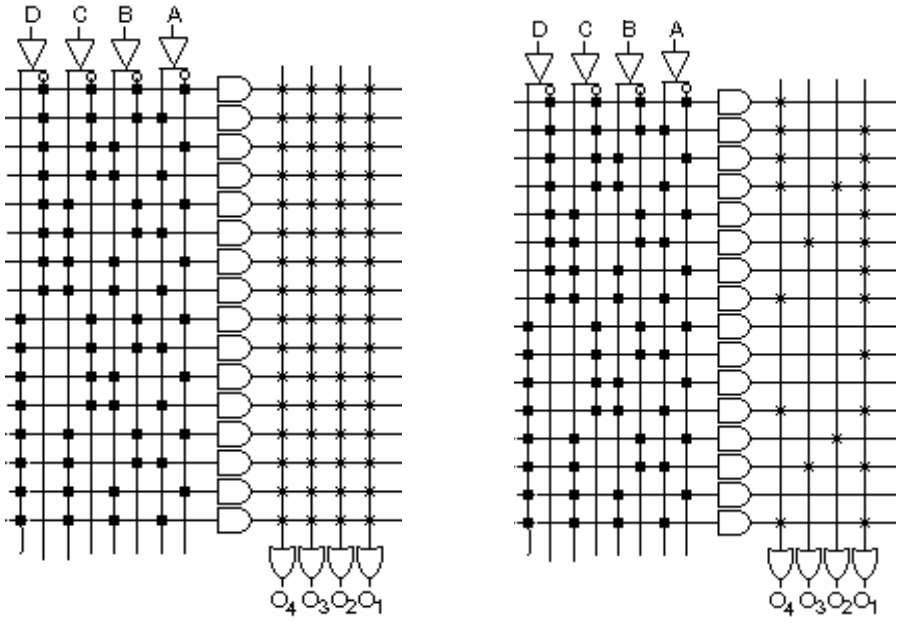
Ta phải chuẩn hóa các hàm chưa chuẩn

$$O_1 = DCBA + \overline{D}CBA + D\overline{C}BA + \overline{D}\,\overline{C}BA + DC\overline{B}A + \overline{D}C\overline{B}A + D\overline{C}\,\overline{B}A + \overline{D}\,\overline{C}\,\overline{B}A + \overline{D}CB\overline{A} + \overline{D}C\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A}$$

$$O_3 = C\overline{B}A = DC\overline{B}A + \overline{D}C\overline{B}A$$

$$O_4 = BA + \overline{D}\,\overline{C} = D\overline{C}BA + \overline{D}CBA + D\overline{C}BA + DCBA + \overline{D}\,\overline{C}BA + \overline{D}C\overline{B}\,\overline{A} + D\overline{C}\,\overline{B}\,\overline{A}$$

Mạch cho ở (H 7.14b)




(a) (b)

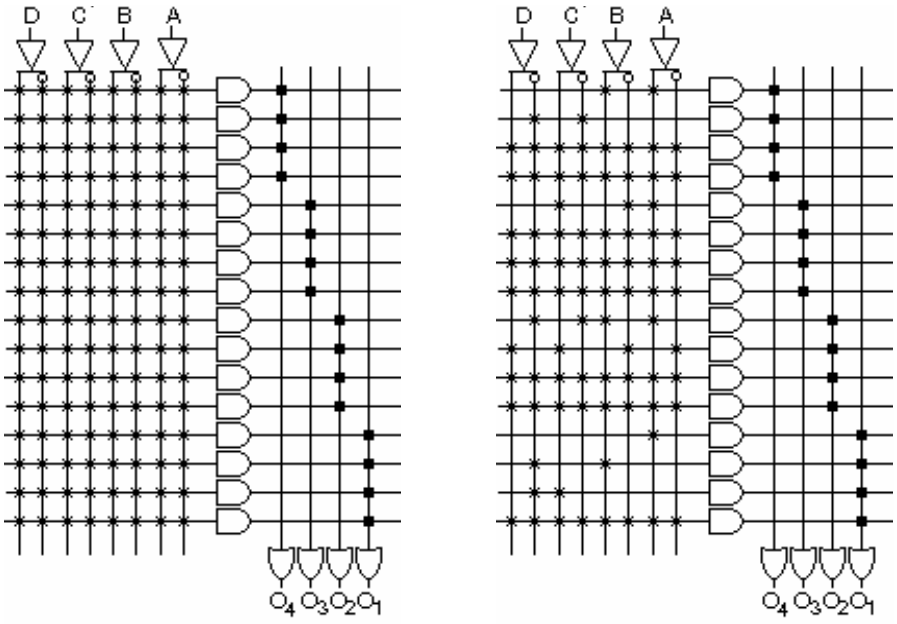
(H 7.14)

### 7.3.2.2 PAL

Mạch tương tự với IC PROM, PAL có các cổng AND 8 ngã vào được nối sống và 4 cổng OR mỗi cổng có 4 ngã vào nối chết với 4 đường tích. Như vậy việc lập trình được thực hiện trên các đường tích

(H 7.15b) là IC PAL đã được lập trình để thực hiện các hàm trong thí dụ trên:

$$O_1 = A + \overline{D}B + \overline{D}C \qquad O_2 = \overline{D}\,\overline{C}BA + DC\overline{B}\,\overline{A} \qquad O_3 = C\overline{B}A \qquad O_4 = BA + \overline{D}\,\overline{C}$$

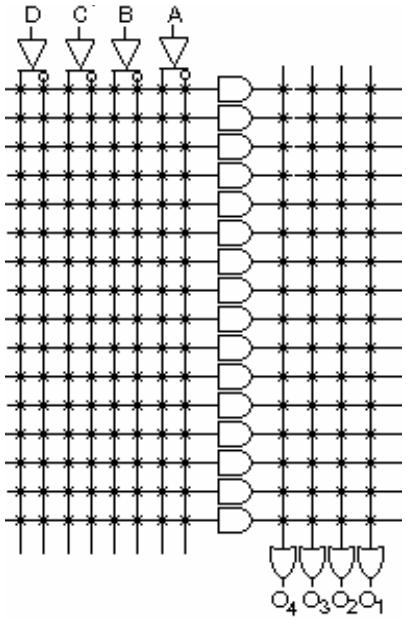



(a) (b)

(H 7.15)

### 7.3.2.3 PLA

PLA có cấu tạo tương tự PROM và PAL, nhưng các ngã vào của cổng AND và cổng OR đều được nối sống (H 7.16). Như vậy khả năng lập trình của PLA bao gồm cả hai cách lập trình của 2 loại IC kể trên.




(H 7.16)

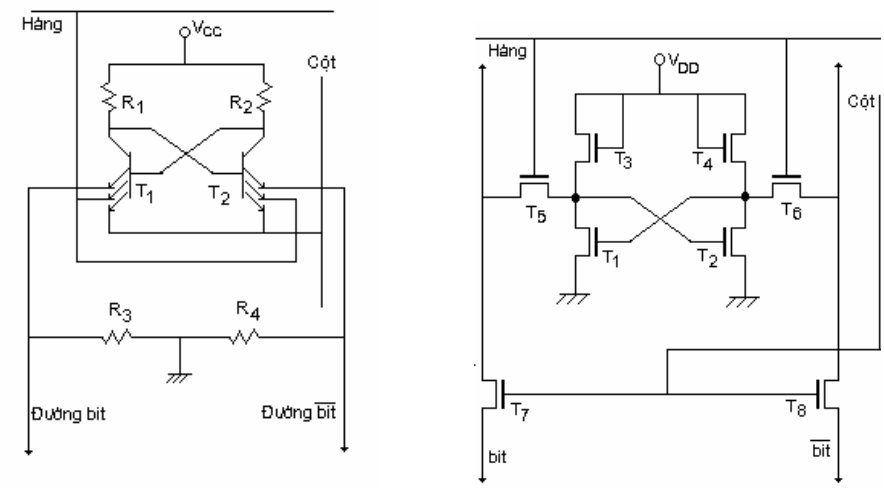
## 7.3.3 RAM (Random Acess Memory)

Có hai loại RAM : RAM tĩnh và RAM động

RAM tĩnh cấu tạo bởi các tế bào nhớ là các FF, RAM động lợi dụng các điện dung ký sinh giữa các cực của transistor MOS, trạng thái tích điện hay không của tụ tương ứng với hai bit 1 và 0. Do RAM động có mật độ tích hợp cao, dung lượng bộ nhớ thường rất lớn nên để định vị các phần tử nhớ người ta dùng phương pháp đa hợp địa chỉ, mỗi từ nhớ được chọn khi có đủ hai địa chỉ hàng và cột được lần lượt tác động. Phương pháp này cho phép n đường địa chỉ truy xuất được $2^{2n}$ vị trí nhớ. Như vậy giản đồ thời gian của RAM động thường khác với giản đồ thời gian của RAM tĩnh và ROM.

### 7.3.3.1 RAM tĩnh (Static RAM, SRAM)

Mỗi tế bào RAM tĩnh là một mạch FlipFlop dùng Transistor BJT hay MOS (H 7.17)




(a) (H 7.17) (b)

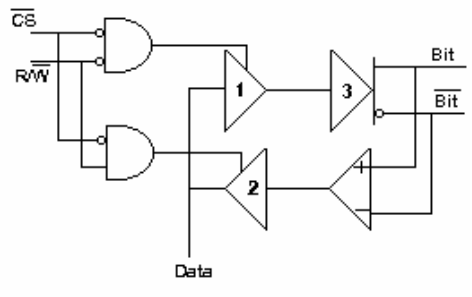
(H 7.17a) là một tế bào nhớ RAM tĩnh dùng transistor BJT với 2 đường địa chỉ hàng và cột.

Khi một trong hai đường địa chỉ hàng hoặc cột ở mức thấp các tế bào không được chọn vì cực E có điện thế thấp hai Transistor đều dẫn, mạch không hoạt động như một FF. Khi cả hai địa chỉ hàng và cột lên cao, mạch hoạt động như FF, hai trạng thái 1 và 0 của tế bào nhớ được đặc trưng bởi hai trạng thái khác nhau của 2 đường bit và $\overline{bit}$ .

Giả sử khi $T_1$ dẫn thì $T_2$ ngưng, đường bit có dòng điện chạy qua, tạo điện thế cao ở $R_3$ trong khi đó đường $\overline{bit}$ không có dòng chạy qua nên ở $R_4$ có điện thế thấp. Nếu ta qui ước trạng thái này tương ứng với bit 1 thì trạng thái ngược lại, là trạng thái $T_1$ ngưng và $T_2$ dẫn, hiệu thế ở điện trở $R_3$ thấp và ở $R_4$ cao, sẽ là bit 0. $R_3$ và $R_4$ có tác dụng biến đổi dòng điện ra điện thế.

Đối với tế bào nhớ dùng MOS, hai đường từ nối với $T_5$, $T_6$ và $T_7$, $T_8$ nên khi một trong hai đường từ ở mức thấp $T_1$ và $T_2$ bị cô lập khỏi mạch, tế bào nhớ không được chọn. Khi cả hai lên cao mạch hoạt động tương tự như trên. Trong mạch này $R_1$ và $R_2$ thay bởi $T_3$ và $T_4$ và không cần $R_3$ và $R_4$ như mạch dùng BJT.

(H 7.18) là mạch điều khiển chọn chip và thực hiện tác vụ đọc/viết vào tế bào nhớ.




(H 7.18)

OPAMP giữ vai trò mạch so sánh điện thế hai đường bit và $\overline{bit}$ cho ở ngã ra mức cao hoặc thấp tùy kết quả so sánh này (tương ứng với 2 trạng thái của tế bào nhớ) và dữ liệu được đọc ra khi cổng đệm thứ 2 mở ( $R/\overline{W}$ lên cao).

Khi cổng đệm thứ nhất mở ( $R/\overline{W}$ xuống thấp) dữ liệu được ghi vào tế bào nhớ qua cổng đệm 1. Cổng 3 tạo ra hai tín hiệu ngược pha từ dữ liệu vào. Nếu hai tín hiệu này cùng trạng thái với hai đường bit và $\overline{bit}$ của mạch trước đó, mạch sẽ không đổi trạng thái nghĩa là

nếu tế bào nhớ đang lưu bit giống như bit muốn ghi vào thì mạch không thay đổi. Bây giờ, nếu dữ liệu cần ghi khác với dữ liệu đang lưu trữ thì mạch FF sẽ thay đổi trạng thái cho phù hợp với 2 tín hiệu ngược pha được tạo ra từ dữ liệu. Bit mới đã được ghi vào.

**- Chu kỳ đọc của SRAM**

Giản đồ thời gian một chu kỳ đọc của SRAM tương tự như giản đồ thời gian một chu kỳ đọc của ROM (H 7.11) thêm điều kiện tín hiệu $R/\overline{W}$ lên mức cao.

**- Chu kỳ viết của SRAM**

(H 7.19) là giản đồ thời gian một chu kỳ viết của SRAM

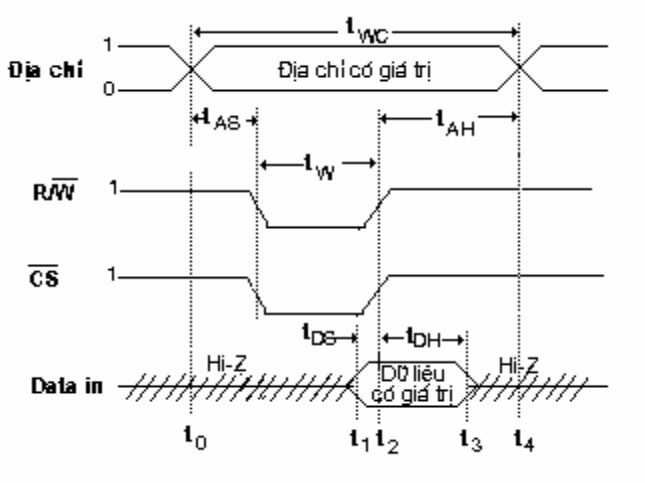
Một chu kỳ viết $t_{WC}$ bao gồm:

**- $t_{AS}$** (Address Setup time): Thời gian thiết lập địa chỉ : Thời gian để giá trị địa chỉ ổn định trên bus địa chỉ cho tới lúc tín hiệu $\overline{CS}$ tác động.

**- $t_W$** (Write time): Thời gian từ lúc tín hiệu $\overline{CS}$ tác động đến lúc dữ liệu có giá trị trên bus dữ liệu.

**- $t_{DS}$ và $t_{DH}$**: Khoảng thời gian dữ liệu tồn tại trên bus dữ liệu bao gồm thời gian trước ($t_{DS}$) và sau ($t_{DH}$) khi tín hiệu $\overline{CS}$không còn tác động

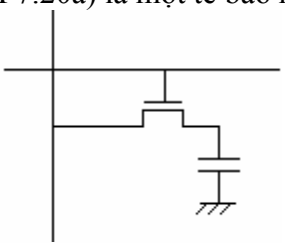
**- $t_{AH}$** (Address Hold time): Thời gian giữ địa chỉ: từ lúc tín hiệu $\overline{CS}$không còn tác động đến lúc xuất hiện địa chỉ mới.



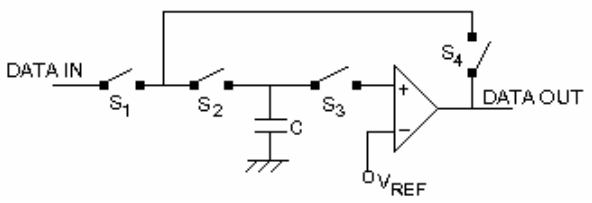

(H 7.19)

### 7.3.3.2 RAM động (Dynamic RAM, DRAM)

(H 7.20a) là một tế bào nhớ của DRAM



(a)




(H 7.20) (b)

(H 7.20b) là một cách biểu diễn tế bào nhớ DRAM trong đó đơn giản một số chi tiết được dùng để mô tả các tác vụ viết và đọc tế bào nhớ này.

Các khóa từ $S_1$ đến $S_4$ là các transistor MOS được điều khiển bởi các tín hiệu ra từ mạch giải mã địa chỉ và tín hiệu $R/\overline{W}$ .

Để ghi dữ liệu vào tế bào, các khóa $S_1$ và $S_2$ đóng trong khi $S_3$ và $S_4$ mở. Bit 1 thực hiện việc nạp điện cho tụ C và bit 0 làm tụ C phóng điện. Sau đó các khóa sẽ mở để cô lập C với phần mạch còn lại. Một cách lý tưởng thì C sẽ duy trì trạng thái của nó vĩnh viễn nhưng thực tế luôn luôn có sự rỉ điện qua các khóa ngay cả khi chúng mở do đó C bị mất dần điện tích .

Để đọc dữ liệu các khóa $S_2$ , $S_3$ , $S_4$ đóng và $S_1$ mở, tụ C nối với một mạch so sánh với một điện thế tham chiếu để xác định trạng thái logic của nó. Điện thế ra mạch so sánh chính là dữ liệu được đọc ra. Do $S_2$ và $S_4$ đóng, dữ liệu ra được nối ngược lại tụ C để làm tươi nó. Nói cách khác, bit dữ liệu trong tế bào nhớ được làm tươi mỗi khi nó được đọc.

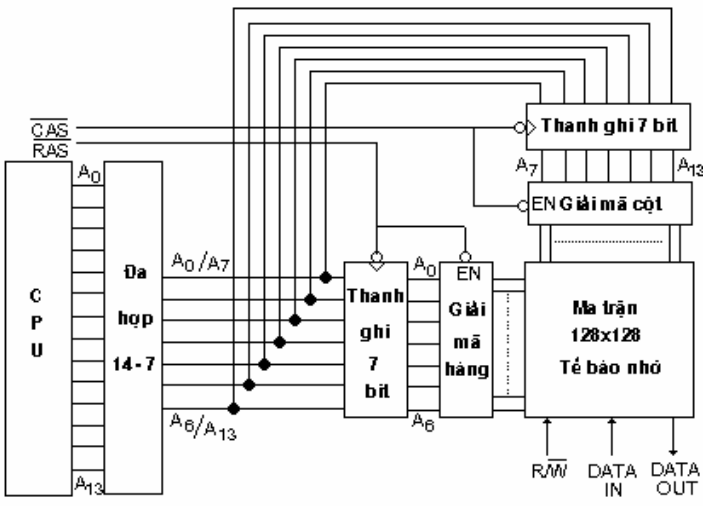
Sử dụng DRAM, được một thuận lợi là dung lượng nhớ khá lớn nhưng phải có một số mạch phụ trợ:

- Mạch đa hợp địa chỉ vì DRAM luôn sử dụng địa chỉ hàng và cột
- Mạch làm tươi để phục hồi dữ liệu có thể bị mất sau một khoảng thời gian ngắn nào đó.

### a. Đa hợp địa chỉ

Như đã nói trên, do dung lượng của DRAM rất lớn nên phải dùng phương pháp đa hợp để chọn một vị trí nhớ trong DRAM. Mỗi vị trí nhớ sẽ được chọn bởi 2 địa chỉ hàng và cột lần lượt xuất hiện ở ngã vào địa chỉ.

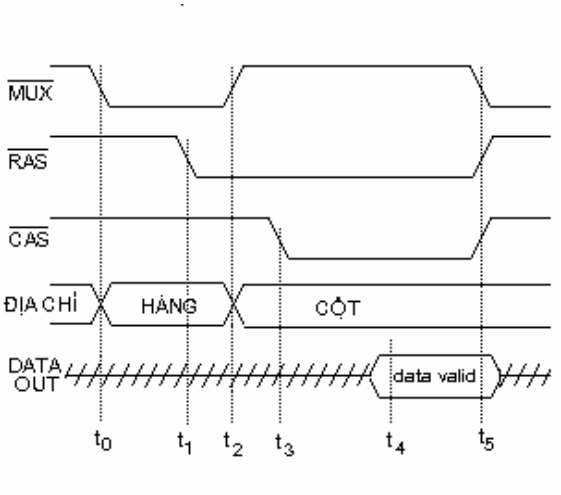
Thí dụ với DRAM có dung lượng 16Kx1, thay vì phải dùng 14 đường địa chỉ ta chỉ cần dùng 7 đường và mạch đa hợp 14 $\rightarrow$ 7 (7 đa hợp 2$\rightarrow$1) để chọn 7 trong 14 đường địa chỉ ra từ CPU (H 7.21). Bộ nhớ có cấu trúc là một ma trận 128x128 tế bào nhớ, sắp xếp thành 128 hàng và 128 cột, có một ngã vào và một ngã ra dữ liệu, một ngã vào $R/\overline{W}$ . Hai mạch chốt địa chỉ (hàng và cột) là các thanh ghi 7 bit có ngã vào nối với ngã ra mạch đa hợp và ngã ra nối với các mạch giải mã hàng và cột. Các tín hiệu $\overline{RAS}$ và $\overline{CAS}$ dùng làm xung đồng hồ cho mạch chốt và tín hiệu Enable cho mạch giải mã. Như vậy 14 bit địa chỉ từ CPU sẽ lần lượt được chốt vào các thanh ghi hàng và cột bởi các tín hiệu $\overline{RAS}$ và $\overline{CAS}$ rồi được giải mã để chọn tế bào nhớ. Vận hành của hệ thống sẽ được thấy rõ hơn khi xét các giản đồ thời gian của DRAM.




(H 7.21)

### b. Giản đồ thời gian của DRAM

(H 7.22) là giản đồ thời gian đọc và viết tiêu biểu của DRAM (Hai giản đồ này chỉ khác nhau về thời lượng nhưng có chung một dạng nên ta chỉ vẽ một)




(H 7.22)

Giản đồ cho thấy tác động của tín hiệu $\overline{MUX}$ và các tín hiệu $\overline{RAS}$ và $\overline{CAS}$. Khi $\overline{MUX}$ ở mức thấp mạch đa hợp cho ra địa chỉ hàng ($A_0$ . . . $A_6$) và được chốt vào thanh ghi khi tín hiệu $\overline{RAS}$ xuống thấp. Khi $\overline{MUX}$ ở mức cao mạch đa hợp cho ra địa chỉ cột ($A_7$ . . . $A_{13}$) và được chốt vào thanh ghi khi tín hiệu $\overline{CAS}$ xuống thấp. Khi cả địa chỉ hàng và cột đã được giải mã, dữ liệu tại địa chỉ đó xuất hiện trên bus dữ liệu để đọc ra hoặc ghi vào ( khả dụng)

### c. Làm tươi DRAM

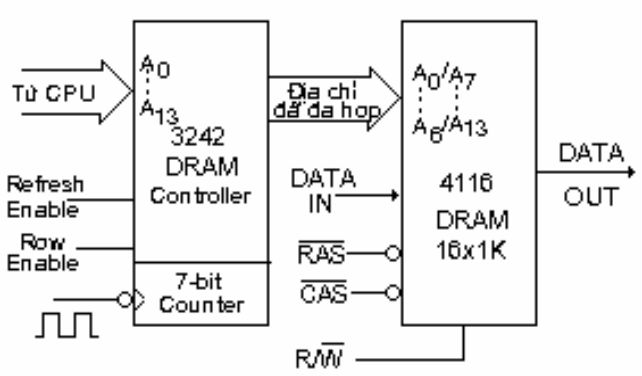
DRAM phải được làm tươi với chu kỳ khoảng 2ms để duy trì dữ liệu.

Trong phần trước ta đã thấy tế bào nhớ DRAM được làm tươi ngay khi tác vụ đọc được thực hiện. Lấy thí dụ với DRAM có dung lượng 16Kx1 (16.384 tế bào) nói trên, chu kỳ làm tươi là 2 ms cho 16.384 tế bào nhớ nên thời gian đọc mỗi tế bào nhớ phải là 2 ms/16.384 = 122 ns. Đây là thời gian rất nhỏ không đủ để đọc một tế bào nhớ trong điều kiện vận hành bình thường. Vì lý do này các hãng chế tạo đã thiết kế các chip DRAM sao cho **mỗi khi tác vụ đọc được thực hiện đối với một tế bào nhớ, tất cả các tế bào nhớ trên cùng một hàng sẽ được làm tươi**. Điều này làm giảm một lượng rất lớn tác vụ đọc phải thực hiện để làm tươi tế bào nhớ. Trở lại thí dụ trên, tác vụ đọc để làm tươi phải thực hiện cho 128 hàng trong 2 ms. Tuy nhiên để vừa vận hành trong điều kiện bình thường vừa phải thực hiện chức năng làm tươi người ta phải dùng thêm mạch phụ trợ, gọi là **điều khiển DRAM** (DRAM controller)

IC 3242 của hảng Intel thiết kế để sử dụng cho DRAM 16K (H 7.23)

Ngã ra 3242 là địa chỉ 7 bit đã được đa hợp và nối vào ngã vào địa chỉ của DRAM. Một mạch đếm 7 bit kích bởi xung đồng hồ riêng để cấp địa chỉ hàng cho DRAM trong suốt thời gian làm tươi. 3242 cũng lấy địa chỉ 14 bit từ CPU đa hợp nó với địa chỉ hàng và cột đã được dùng khi CPU thực hiện tác vụ đọc hay viết. Mức logic áp dụng cho các ngã REFRESH ENABLE và ROW ENABLE xác định 7 bit nào của địa chỉ xuất hiện ở ngã ra mạch controller cho bởi bảng

| REFRESH ENABLE | ROW ENABLE | Controller output |
|---|---|---|
| HIGH | X | Refresh address (từ mạch đếm) |
| LOW | HIGH | Địa chỉ hàng ($A_0$ . . . $A_6$ từ CPU) |
| LOW | LOW | Địa chỉ cột ($A_7$ . . .$A_{13}$ từ CPU) |




(H 7.23)

# 7.4 MỞ RỘNG BỘ NHỚ

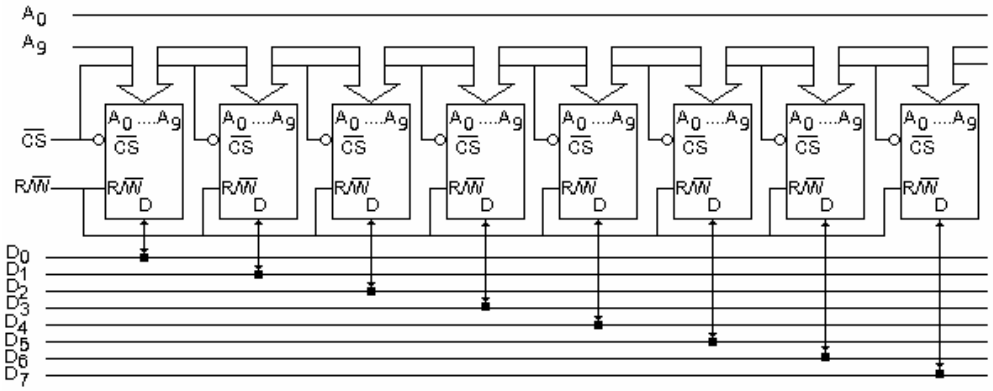
Các IC nhớ thường được chế tạo với dung lượng nhớ có giới hạn, trong nhiều trường hợp không thể thỏa mãn yêu cầu của người thiết kế. Do đó mở rộng bộ nhớ là một việc làm cần thiết. Có 3 trường hợp phải mở rộng bộ nhớ.

## 7.4.1. Mở rộng độ dài từ

Đây là trường hợp số vị trí nhớ đủ cho yêu cầu nhưng dữ liệu cho mỗi vị trí nhớ thì không đủ. Có thể hiểu được cách mở rộng độ dài từ qua một thí dụ

**Thí dụ**: Mở rộng bộ nhớ từ 1Kx1 lên 1Kx8 :

Chúng ta phải dùng 8 IC nhớ 1Kx1, các IC nhớ này sẽ được nối chung bus địa chỉ và các đường tín hiệu điều khiển và mỗi IC quản lý một đường bit. 8 IC sẽ vận hành cùng lúc để cho một từ nhớ 8 bit (H 7.24).

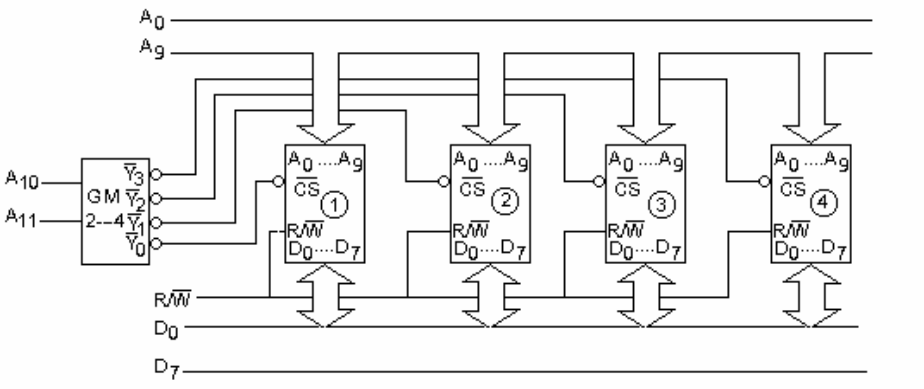



(H 7.24)

## 7.4.2 Mở rộng vị trí nhớ

Số bit cho mỗi vị trí nhớ đủ theo yêu cầu nhưng số vị trí nhớ không đủ

**Thí dụ**: Có IC nhớ dung lượng 1Kx8. Mở rộng lên 4Kx8. Cần 4 IC. Để chọn 1 trong 4 IC nhớ cần một mạch giải mã 2 đường sang 4 đường, ngã ra của mạch giải mã lần lượt nối vào các ngã $\overline{CS}$ của các IC nhớ, như vậy địa chỉ của các IC nhớ sẽ khác nhau (H 7.25). Trong thí dụ này IC1 chiếm địa chỉ từ 000H đến 3FFH, IC2 từ 400H đến 7FFH, IC3 từ 800H đến BFFH và IC4 từ C00H đến FFFH

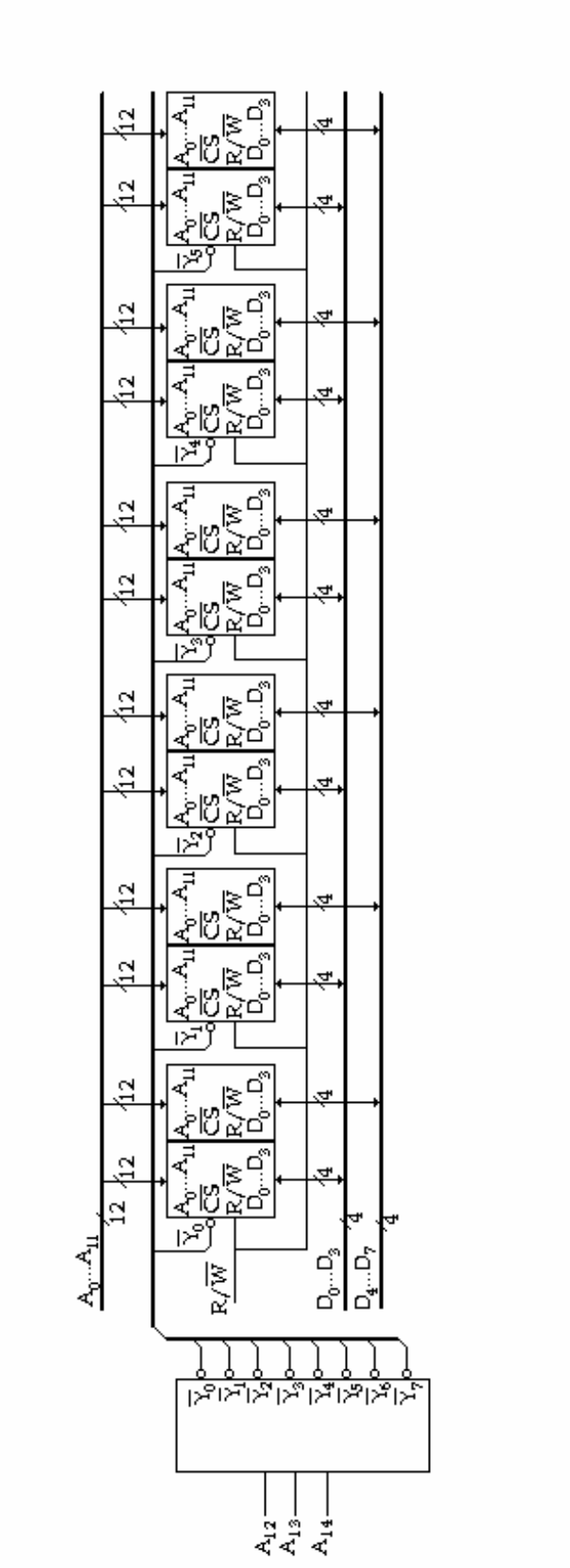



(H 7.25)

### 7.4.3 Mở rộng dung lượng nhớ

Cả vị trí nhớ và độ dài từ của các IC đều không đủ để thiết kế. Để mở rộng dung lượng nhớ ta phải kết hợp cả hai cách nói trên

**Thí dụ**: Mở rộng bộ nhớ từ 4Kx4 lên 24Kx8. Cần 6 cặp IC mắc song song, mỗi cặp IC có chung địa chỉ và được chọn bởi một mạch giải mã 3 sang 8 đường (H 7.26). Ta chỉ dùng 6 ngã ra từ $Y_0$ đến $Y_5$ của mạch giải mã




(H 7.26)

- Địa chỉ IC (1&2): 0000H - 0FFFH, IC (3&4) : 1000H - 1FFFH, IC (5&6): 2000H - 2FFFH và IC (7&8) : 3000H - 3FFFH IC (9&10): 4000H - 4FFFH và IC (11&12) : 5000H - 5FFFH

# BÀI TẬP

1. Dùng IC PROM 4 ngã vào và 4 ngã ra thiết kế mạch chuyển mã từ Gray sang nhị phân của số 4 bit.

2. Dùng IC PAL 4 ngã vào và 4 ngã ra thiết kế mạch chuyển từ mã Excess-3 sang mã Aiken của các số từ 0 đến 9.

Dưới đây là 2 bảng mã

Excess-3

| N | A | B | C | D |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 1 | 1 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 1 | 0 | 0 | 1 |
| 7 | 1 | 0 | 1 | 0 |
| 8 | 1 | 0 | 1 | 1 |
| 9 | 1 | 1 | 0 | 0 |

Aiken

| A | B | C | D |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

3. Thiết kế mạch để mở rộng bộ nhớ từ 2Kx4 lên 2Kx8

4. Thiết kế mạch để mở rộng bộ nhớ từ 1Kx4 lên 8Kx4.
Cho biết địa chỉ cụ thể của các IC

5. Thiết kế mạch để mở rộng bộ nhớ từ 2Kx4 lên 16Kx8.
Cho biết địa chỉ cụ thể của các IC