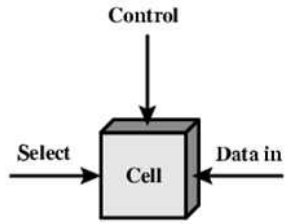
메모리 개요

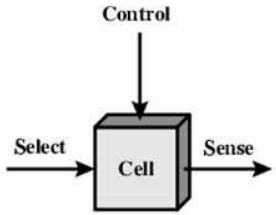
1. 메모리 및 메모리 셀

데이터 쓰기



- **메모리(Memory)**
  - Memory Cell들의 배열로 구성된 데이터 저장장소
  - 어드레스(Address)를 사용하여 셀들의 위치를 지정

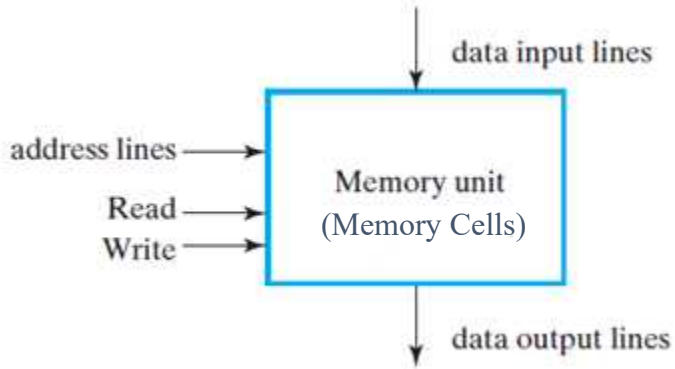
데이터 읽기



- **메모리 셀(Memory cell)**
  - 이진 데이터를 저장할 수 있는 저장공간
  - 이진값 '0' 과 '1' 을 표현할 수 있는 2 가지의 상태를 유지하는 소자.
    - 상태를 설정 → 쓰기 (write)
    - 상태를 감지 → 읽기 (read)
  - 메모리 셀 구현방법
    - 마그네틱 메모리 셀 : 자성체를 활용한 메모리 소자
    - 반도체 메모리 셀 : 반도체위에 전자회로를 사용하여 구현한 메모리 소자.

1.1 메모리 동작

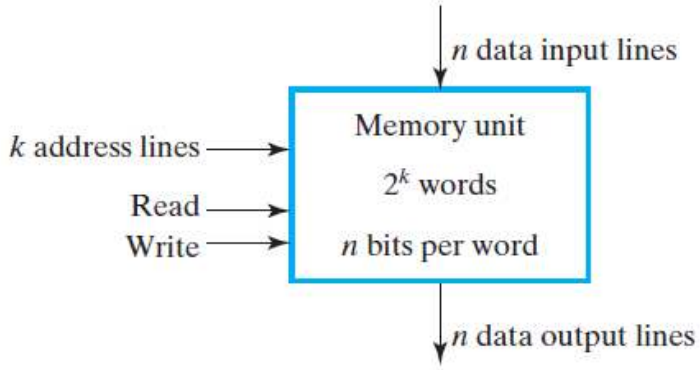
- **메모리 동작 유형**
  - 메모리 읽기 (Memory Read) : 메모리에 저장된 데이터를 메모리 밖으로 출력하는 동작.
  - 메모리 쓰기 (Memory Write) : 데이터를 메모리에 저장하는 동작.



1.2 메모리 연결

- **메모리 연결**
  - 어드레스 선택 라인 : 메모리 셀의 위치 지정
    - 메모리 셀의 수에 따라 결정
  - 데이터 라인 : 메모리 셀에 저장하거나, 메모리 셀에 저장된 데이터를 읽어올 때 데이터를 입출력 경로
  - 제어 라인 : 메모리 동작을 제어하는 신호

- **메모리 워드(Word)**
  - 메모리에 입출력되는 단위 데이터의 비트 수



1.3 메모리 접근방식

- **임의 접근(Random Access) 메모리**
  - 메모리 셀에 데이터를 읽거나 쓸 때 소요되는 시간이 메모리 셀의 위치에 상관없이 일정한 메모리.
  - 어드레스를 사용하여 기억장소를 식별한다.
  - 접근 시간은 이전 접근 장소와 무관하며, 항상 일정하다.
    - 반도체 메모리

- **직접 접근 (direct access)**
  - 각 블록이 별도 주소 (unique address) 를 가지고 있다.
  - 접근 방법은 근처 (vicinity) 로 이동한 다음 순차적 검색 (sequential search), 계수 (counting), 또는 대기 (waiting) 을 통하여 최종 위치에 도달한다.
  - 접근 시간은 가변적이다.
    - 하드 디스크

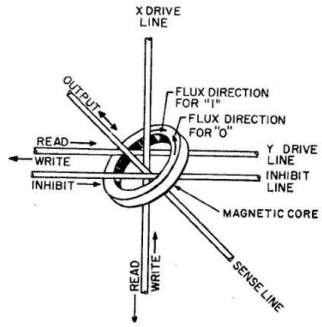
1.3 메모리 접근방식

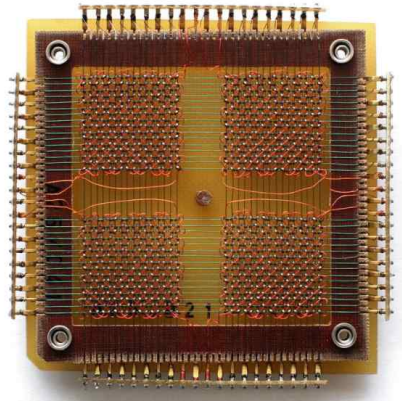
- **순차적 액세스(Sequential Access) 메모리**
  - 메모리 접근이 처음부터 순차적으로 이루어 진다.
  - 메모리 셀의 위치나 메모리 접근시 시작 위치에 따라 메모리 접근 시간이 다름.
    - 마그네틱 Tape

- **연관 접근 (associative access)**
  - 기억장소는 저장된 데이터 내용과 비교함으로써 식별된다.
  - 접근 시간은 이전 접근 장소와 무관하며, 항상 일정 : 비교 부담 (overhead)
    - cache

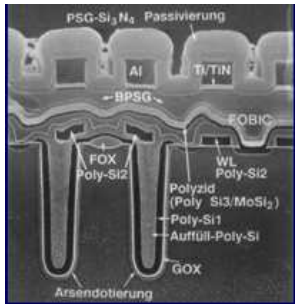
1.4 메모리 구현방식

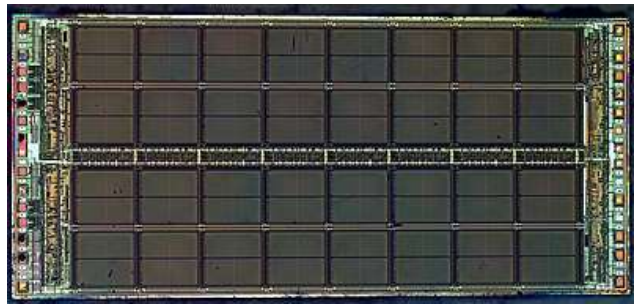
- **마그네틱 메모리(Magnetic Memory)**
  - 자성체를 활용한 메모리.
    - 마그네틱 코어 메모리
    - 하드 디스크





- **반도체 메모리(Semiconductor Memory)**
  - 반도체상에 전자회로를 사용하여 구현한 메모리.
    - RAM, ROM



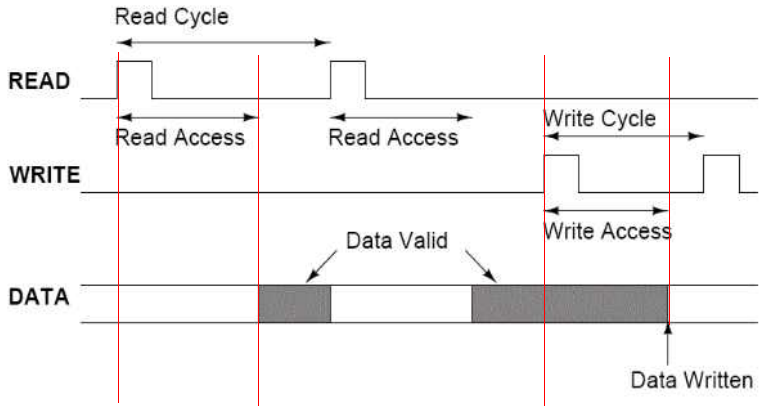


2. 메모리 성능

- **메모리 성능 표시 방법**
  - 메모리 접근 시간
    - 메모리에서 read/write 동작을 수행하는데 걸리는 시간.
  - 메모리 사이클 시간
    - 메모리 접근 시간 + 다음 접근 준비시간
  - 메모리 전송률
    - 메모리로 데이터가 전송되어 들어가거나 나오는 비율.

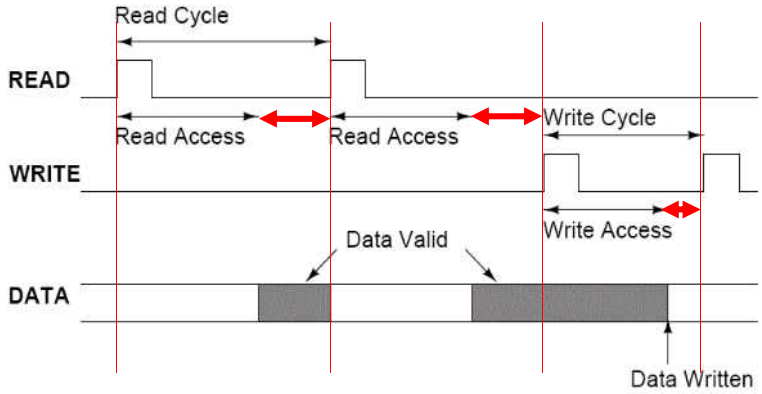
2.1 메모리 접근 시간

- **메모리 접근 시간 (Memory Access Time)**
  - 메모리에서 read/write 동작을 수행하는데 걸리는 시간.
    - Read access time : 어드레스와 제어 신호가 메모리에 도착하는 순간부터 데이터가 읽혀지는 순간까지 걸리는 시간.
    - Write access time : 어드레스와 제어 신호가 메모리에 적용되고, 데이터 쓰기가 완료될때까지 걸리는 시간.



2.2 메모리 사이클 시간

- **메모리 사이클 시간 (Memory Cycle Time)**
  - 메모리 access time + (다음 접근에 걸리는 추가적인 시간)
  - 추가적인 시간 예
    - 신호선에서 과도 전류가 완전히 소멸되는데 걸리는 시간.
    - 읽기 동작 후 데이터가 파괴되는 저장장치의 경우, 데이터를 복원하는데 걸리는 시간.
  - 다음 메모리 read/write 제어신호 출력 시점을 결정.



2.3 메모리 전송률

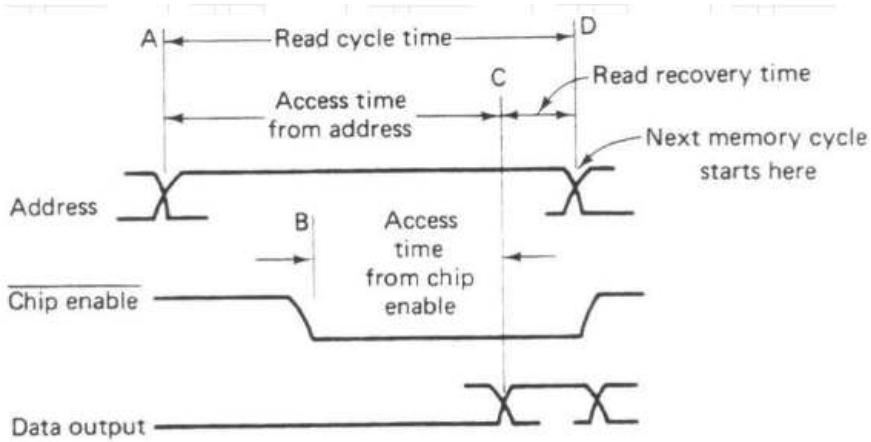
- **메모리 전송률 (Memory Transfer Rate)**
  - 메모리로 데이터가 전송되어 들어가거나 나오는 비율.
  - 임의 접근 메모리 경우, 전송률은 1/(cycle time).
  - 임의 접근 메모리가 아닌 경우, 전송률

$$T_N = T_A + \frac{n}{R}$$

T_N = N 개의 비트들을 읽거나 쓰는 데 걸리는 평균 시간
T_A = 평균 접근 시간
n **= 비트 수**
R **= 전송률, 초당 비트 수 (bits per seconds)**

3. 메모리 타이밍

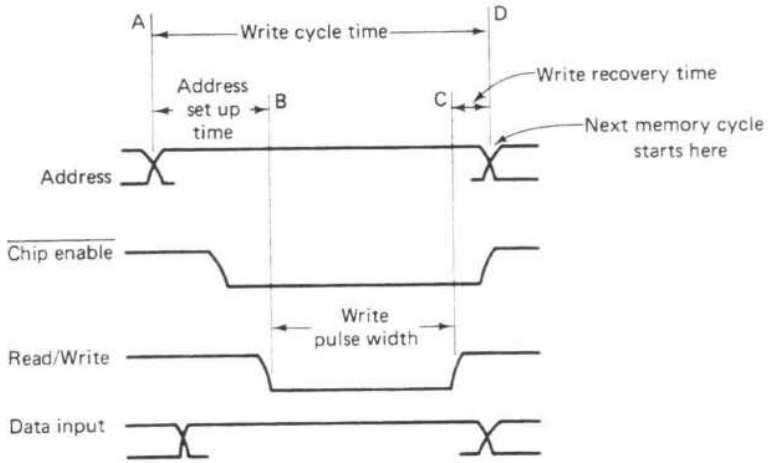
- **메모리는 클럭(Clock)을 사용하지 않고 제어신호의 변화에 따라 데이터 읽고 쓰기가 실행된다.**
- **읽기 타이밍(Read Timing)**
  - Access Time : 어드레스가 주어진 시간부터 데이터가 출력되는데 소요되는 최대 시간.
  - Read Cycle Time : 이번 Read 시작시간부터 완료되는 시점(다음 Read 를 시작할 수 있는 시점)에 소요되는 시간.



Read Timing Diagram

3. 메모리 타이밍

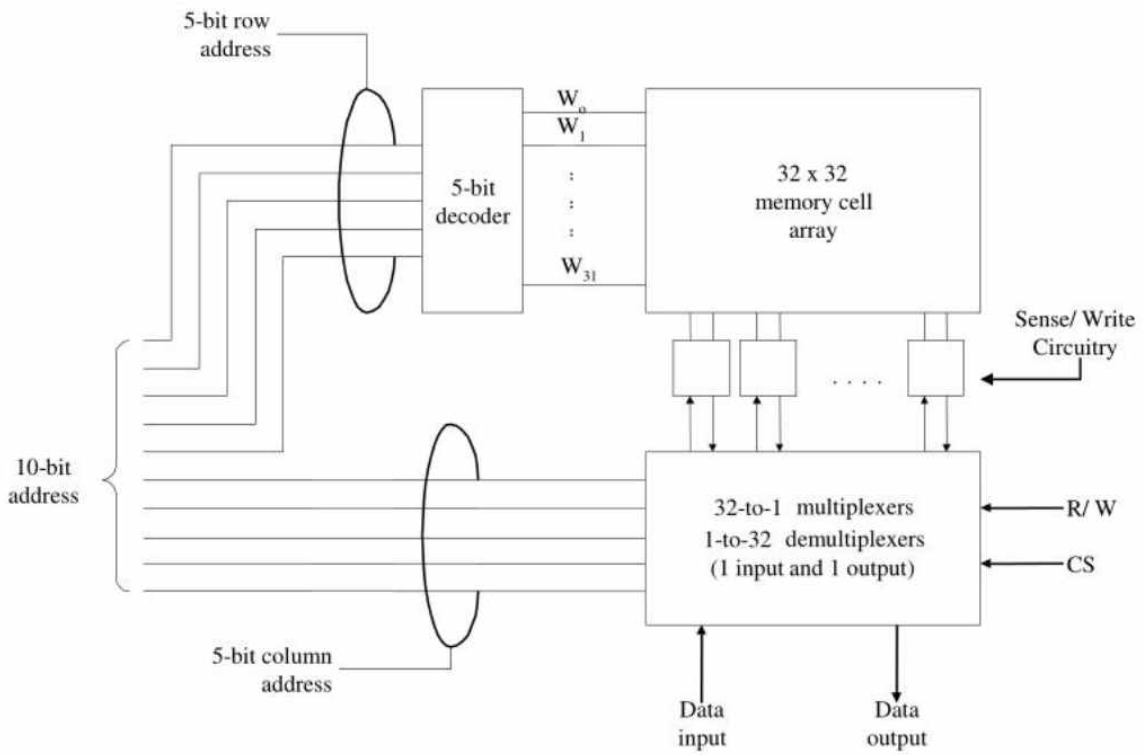
- **쓰기 타이밍(Write Timing)**
  - Write Cycle Time : 어드레스가 주어진 후 메모리 셀에 데이터가 저장되는 최대 시간.



Write Timing Diagram

메모리 구조

1. 메모리 내부구조

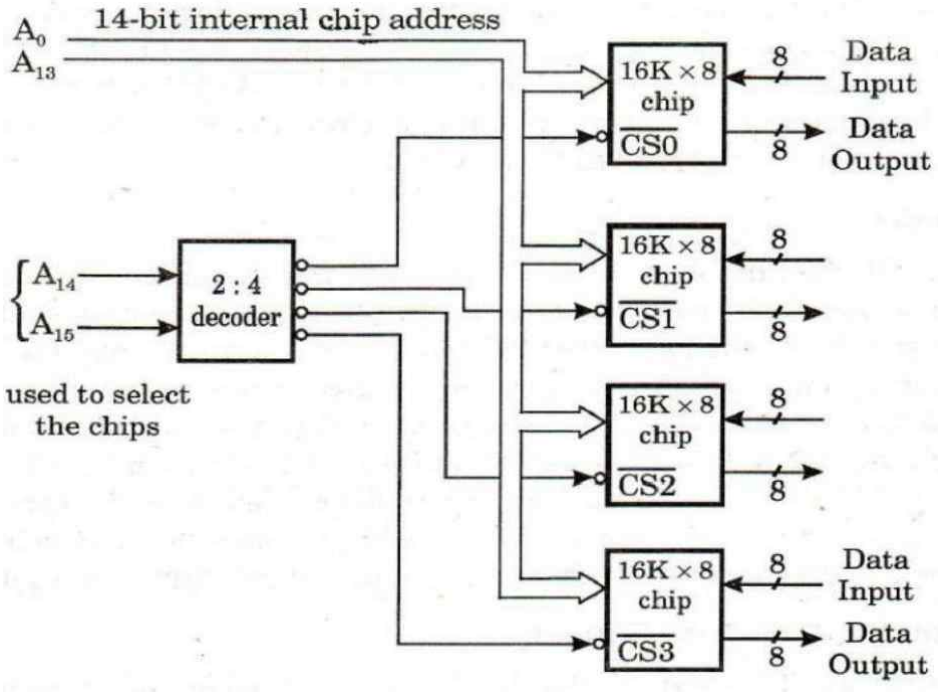

5-bit row address
5-bit decoder
W_0
W_1
W_{31}
32 x 32 memory cell array
Sense/ Write Circuitry
10-bit address
32-to-1 multiplexers
1-to-32 demultiplexers
(1 input and 1 output)
R/ W
CS
5-bit column address
Data input
Data output

2. 메모리 확장

- **메모리 확장 (Memory Expansion)**
  - 여러 개의 메모리 칩을 연결하여 대용량 메모리를 구성하는 것.
    - 용량 확장(Size/Capacity Expansion) : 메모리의 용량을 확장.
    - 워드 확장(Word-length Expansion) : 메모리 입출력 데이터 폭을 확장.

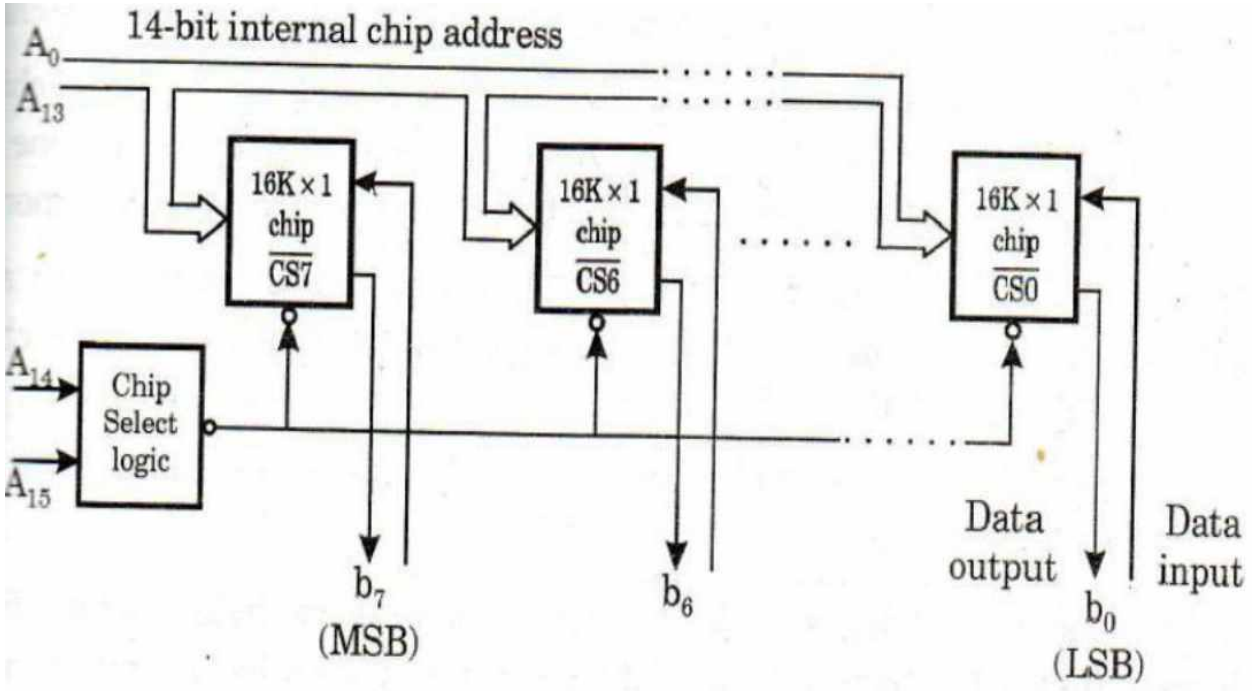
2.1 용량 확장

- 16KB(16Kx8) 메모리 4개를 사용하여 64KB 메모리로 확장



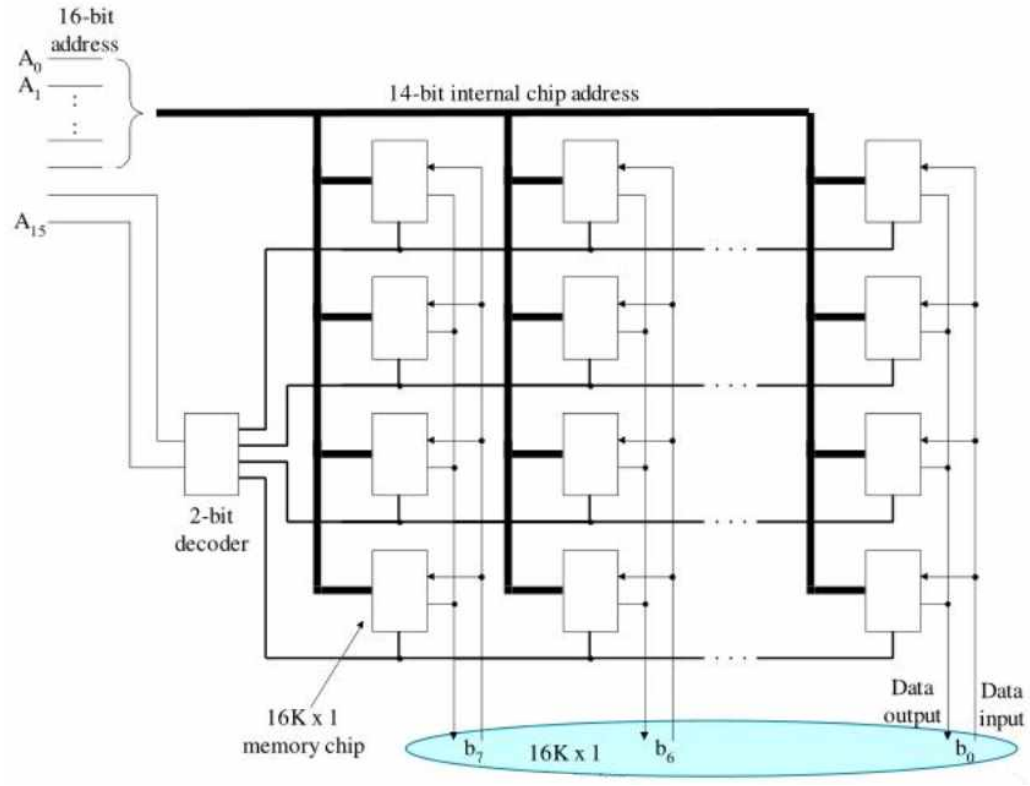
2.2 워드 확장

- 2KB(16Kx1) 메모리 8개를 사용하여 16KB 메모리로 확장



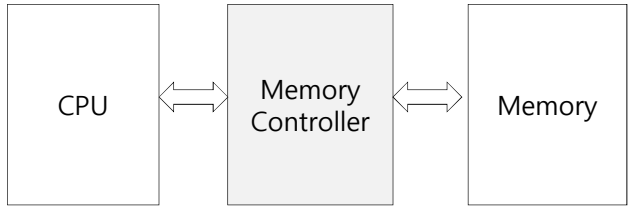
2.3 용량 및 워드 확장

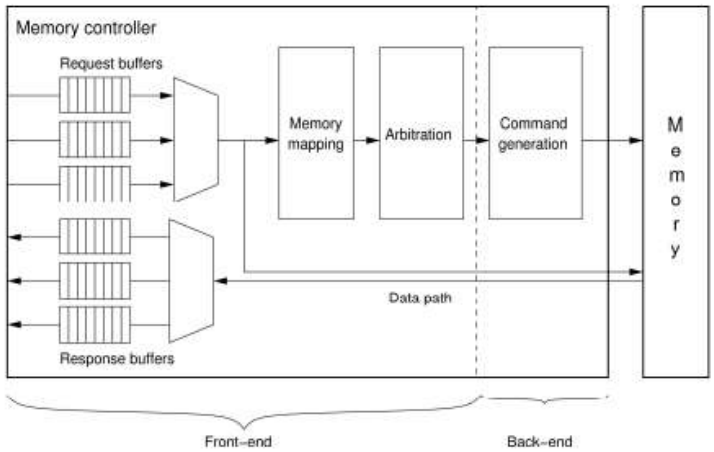
- 2KB(16Kx1) 메모리 32개(8x4)를 사용하여 164KB 메모리로 확장



3. 메모리 컨트롤러

- **메모리 컨트롤러(Memory Controller)**
  - CPU와 메모리사이에 위치하여 메모리 사용에 필요한 작업을 수행하거나 제어신호를 발생하는 장치
    - 별도 칩, 마이크로 프로세서에 내장
- **기능**
  - Front-End
    - 메모리에 대한 요청 및 응답 버퍼링.
    - Memory Mapping : 메모리 어드레스 디코딩
    - Arbitration : 메모리 접근 순서 결정
  - Back-End
    - 명령어 생성 : 타겟 메모리에 필요한 명령어 생성

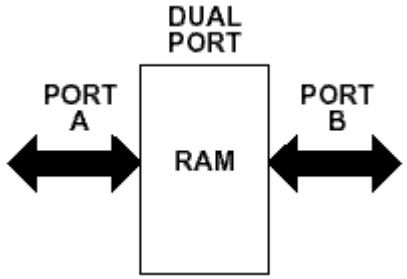


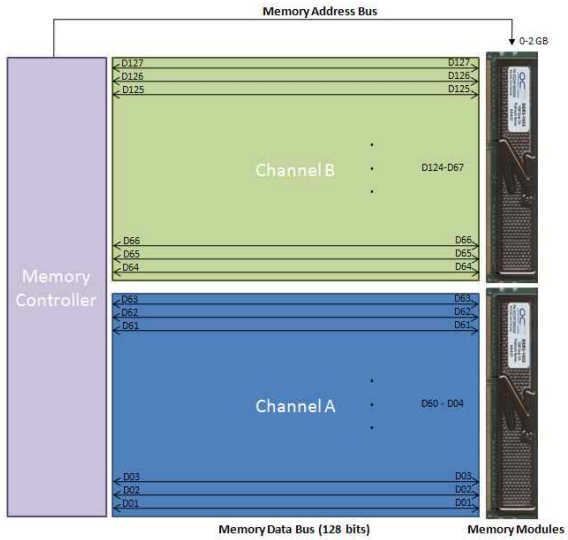


4. 메모리 대역폭 확장

- **메모리 대역폭 확장(Bandwidth Expansion)**
  - CPU와 메모리 사이의 데이터 전송율.

- **메모리 대역폭 확장 방법**
  - 메모리 접근 속도 또는 동작 주파수 향상
  - 멀티-포트(Multi-port) 메모리 사용한 입출력 포트 수 증가
    - 멀티포트 메모리 : 하나의 메모리의 입출력 포트가 2개 이상인 메모리
  - 멀티 채널 (Multi-channel) 메모리 구조
    - DRAM와 메모리 컨트롤러 사이의 통신채널을 여러 개 사용하여 메모리 전송속도를 향상시키는 방법.
  - 메모리 병렬화 (Interleaved 메모리)

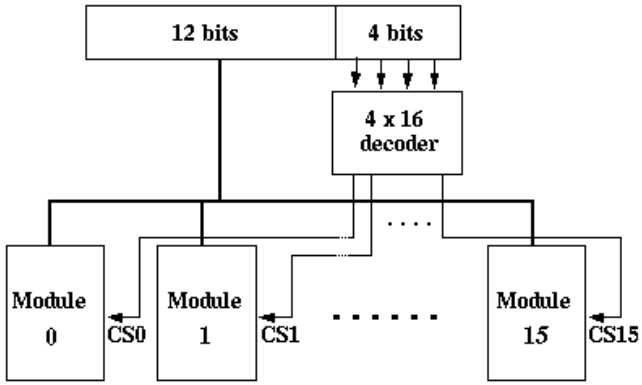




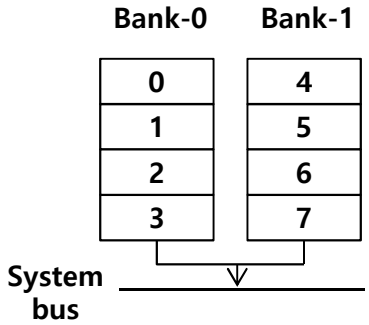
Dual-Channel Memory Architecture

5. 메모리 인터리빙

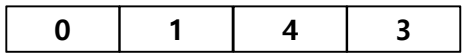
- **Interleaved 메모리**
  - 메모리를 뱅크(bank) 로 구성
  - 메모리 액세스를 병렬화



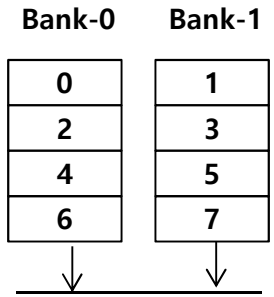
Memory Interleaving
(2^{16}=64K words, 16 modules)



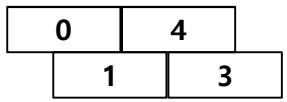
Non-interleaved



Non-interleaved Memory Access



Interleaved



Interleaved Memory Access

데이터 저장방식

1. 저장 순서

- **Byte Order**
  - 멀티 바이트로 구성된 데이터를 바이트 단위로 어드레싱하는 메모리에 저장하는 방법
    - 엔디안 (endian)

- **Big-endian**
  - Left-to-right (western culture language style)
  - 가장 낮은 어드레스에 상위 바이트 (most significant byte) 부터 차례로 저장.
  - IBM, Motorola, SUN, most RISC's, Internet

- **Little-endian**
  - Right-to-left (arithmetic operation style)
  - 가장 낮은 어드레스에 하위 바이트 (least significant byte) 부터 차례로 저장.
  - Intel, VAX

1.1 Byte Order 사례

- **저장 사례**
  - "0x12345678" 값을 4 바이트 메모리에 저장하는 방법

Address	Value
184	12
185	34
186	56
187	78

Big-Endian

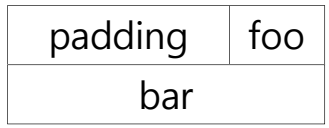
Address	Value
184	78
185	56
186	34
187	12

Little-Endian

2. 메모리 정렬 및 패딩

- **메모리 정렬(Memory Alignment)**
  - 다양한 크기의 변수로 구성된 데이터 구조체를 메모리에 할당하는 방법
  - 워드 크기에 맞춰 메모리 접근
    - 32-비트 시스템인 경우, 4 바이트 단위로 어드레싱
  - 컴파일러에서 워드 단위로 메모리 할당
    - 빈 공간에 임의의 데이터를 패딩(Padding)하여 정렬시킨다.

```
struct test {
   char foo; // 1-바이트
   int  bar; // 4-바이트
};
```



2.1 메모리 정렬 사례

- 구조체인 경우

```
struct{
   int      a;     //0x1112_1314                    word
   int      pad;   //
   double   b;     //0x2122_2324_2526_2728          doubleword
   char*    c;     //0x3132_3334                    word
   char     d[7];  //'A'.'B','C','D','E','F','G'    byte array
   short    e;     //0x5152                         halfword
   int      f;     //0x6162_6364                    word
} s;
```

Big-endian address mapping

Byte address								
00	**11** 00	**12** 01	**13** 02	**14** 03	04	05	06	07
08	**21** 08	**22** 09	**23** 0A	**24** 0B	**25** 0C	**26** 0D	**27** 0E	**28** 0F
10	**31** 10	**32** 11	**33** 12	**34** 13	**'A'** 14	**'B'** 15	**'C'** 16	**'D'** 17
18	**'E'** 18	**'F'** 19	**'G'** 1A	1B	**51** 1C	**52** 1D	1E	1F
20	**61** 20	**62** 21	**63** 22	**64** 23				

Little-endian address mapping

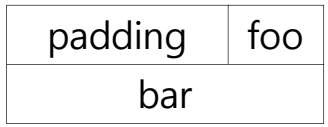
								Byte address
				11 03	**12** 02	**13** 01	**14** 00	00
07	06	05	04	**21** 0F	**22** 0E	**23** 0D	**24** 0C	08
25 0B	**26** 0A	**27** 09	**28** 08	**'D'** 17	**'C'** 16	**'B'** 15	**'A'** 14	10
31 13	**32** 12	**33** 11	**34** 10			**51** 1D	**52** 1C	18
	'G' 1A	**'F'** 19	**'E'** 18					
				61 23	**62** 22	**63** 21	**64** 20	20

3. Unpacked/Packed 할당

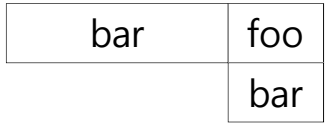
- **Unpacked 메모리 할당**
  - 워드 단위로 메모리 할당
  - 빈 공간에 임의 데이터 추가
  - 어드레싱 간단
  - 메모리 공간 낭비

- **Packed 메모리 할당**
  - 바이트 단위로 메모리 할당
  - 빈 공간없이 밀착 할당
  - 메모리 공간의 효율적 할당
  - 메모리 접근 속도 저하

```
struct test {
   char foo; // 1-바이트
   int   bar; // 4-바이트
};
```



Unpacked



Packed

3. Unpacked/Packed 할당

- **데이터 구조체에 대한 명시적 메모리 공간 할당 방법**
  - 컴파일러 지시자(__packed__)를 명시적으로 지정

```
struct __attribute__((__packed__)) test {
   char foo; // 1-바이트
   int   bar; // 4-바이트
};
```

  - 프로그램 코드에서 변수 재배치하여 패딩을 최소화하여 메모리 공간 절약

```
struct test {
   char foo; // 1-바이트
   int   bar; // 4-바이트
   char soo; // 1-바이트
   int   woo; // 4-바이트
   char pou; // 1-바이트
};
```

⇨

```
struct test {
   int   bar; // 4-바이트
   int   woo; // 4-바이트
   char foo; // 1-바이트
   char soo; // 1-바이트
   char pou; // 1-바이트
};
```

메모리 특성과 유형

1. 메모리 특성

- **접근 용이성(Accessability)**
  - 순차적 접근(Sequential) : Tape
  - 임의 접근(Random) : 반도체 메모리
  - 내용기반 접근(Associative)

- **변경 용이성(Writability)**
  - 읽기 전용(Read-Only) : ROM
  - 읽기/쓰기 가능(Read-Writable) : RAM
  - 절충형(Hybrid) : Least-Write & Most-Read, EEPROM, Flash

1. 메모리 특성

- **휘발성(Volatility)**
  - 휘발성(Volatile) : 전원공급 중단시 데이터 훼손. RAM
  - 비휘발성(Non-volatile) : 전원공급이 중단되어도 데이터 유지. ROM
  - 절충형(Hybrid) : 배터리 내장형 RAM. NVRAM(Non-volatile RAM)

- **프로그램 가능성(Programmability)**
  - OTP(One-Time Programmable) : Mask 프로그래밍 (제조자 프로그래밍)
  - MTP(Multiple-Time Programmable) : 사용자 프로그래밍

2. 메모리 분류

- **RAM(Random Access Memory)**
  - 메모리에 대한 읽기/쓰기가 모두 가능한 메모리.
  - 휘발성(Volatile) 메모리 : 전원공급 중단시 데이터 훼손.

- **ROM(Read-Only Memory)**
  - 메모리에 대한 읽기만 가능한 메모리.
  - 비휘발성(Non-volatile) 메모리 : 전원공급이 중단되어도 데이터 유지.

2.1 메모리 유형과 특성

<table>
<tr><th colspan="2">Memory Type</th><th>Category</th><th>Erasure</th><th>Write Mechanism</th><th>Volatility</th></tr>
<tr><td colspan="2">Random-access memory (RAM)</td><td>Read-write memory</td><td>Electrically, byte-level</td><td>Electrically</td><td>Volatile</td></tr>
<tr><td rowspan="4">ROM</td><td>Mask ROM (MROM)</td><td rowspan="2">Read-only memory</td><td rowspan="2">Not possible</td><td>Masks</td><td rowspan="5">Non-volatile</td></tr>
<tr><td>Programmable ROM (PROM)</td><td rowspan="4">Electrically</td></tr>
<tr><td>Erasable PROM (EPROM)</td><td rowspan="3">Read-mostly memory</td><td>UV light, chip-level</td></tr>
<tr><td>Electrically Erasable PROM (EEPROM)</td><td>Electrically, byte-level</td></tr>
<tr><td colspan="2">Flash memory</td><td>Electrically, block-level</td></tr>
</table>

RAM

1. RAM 개요

- **읽고 쓰기가 가능한 랜덤 액세스 메모리.**
  - 휘발성 (volatile) 메모리
  - 메인 메모리, 임시 저장장치로 사용

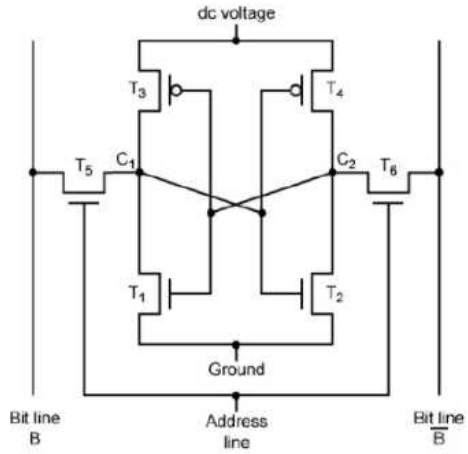
- **RAM 유형**
  - SRAM (Static RAM)
  - DRAM (Dynamic RAM)

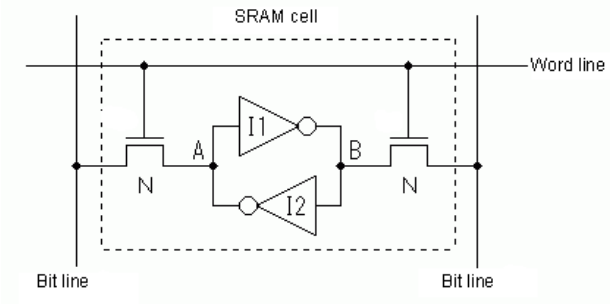
2. SRAM

- **SRAM (static RAM)**
  - 2진 값을 flip-flop 을 사용하여 저장.
  - 전력이 공급되는 동안 저장된 값을 계속 유지.

- **특성**
  - 누설 전류가 없음 : 재충전 필요없음.
  - 복잡한 구조 : 비트당 필요면적이 커서 집적도 떨어짐.
  - 고비용, 고속.

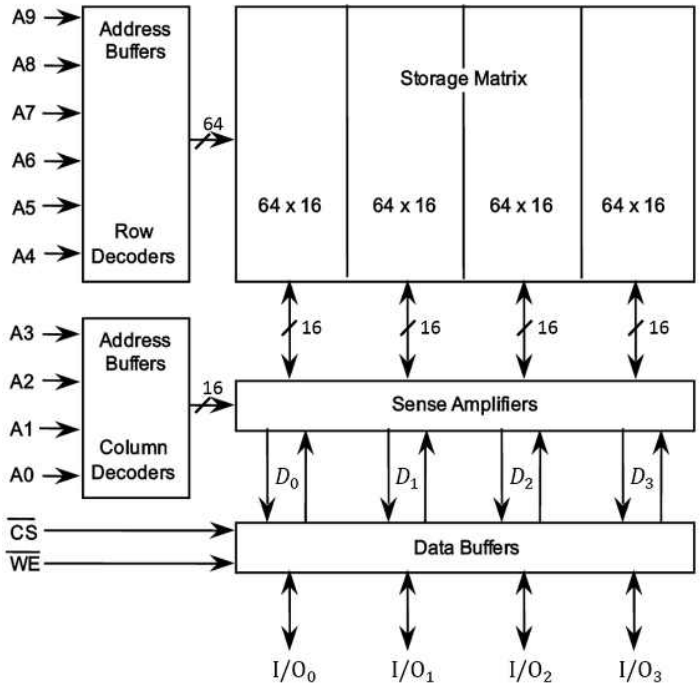
- **캐시 메모리에 사용.**





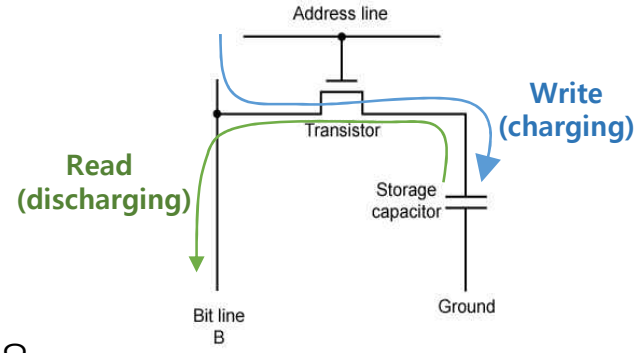
2.1 SRAM 구조

- **1024 x 4-bit SRAM**



3. DRAM

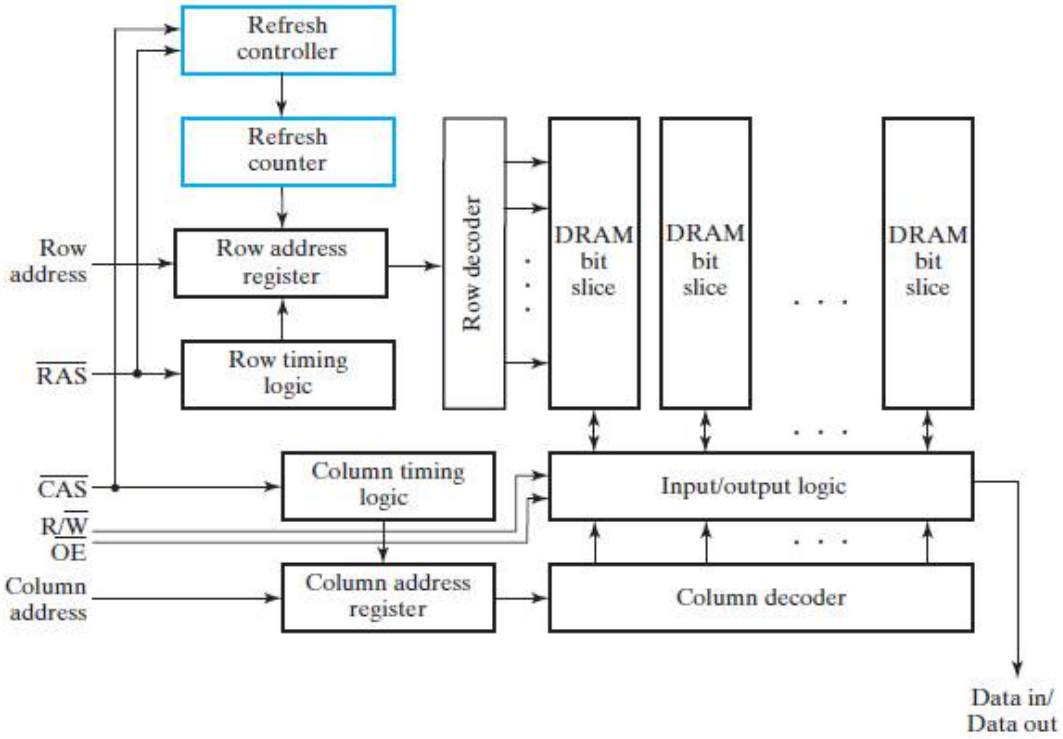
- **DRAM (dynamic RAM)**
  - 캐패시터에 전하를 충전하는 방식으로 데이터를 저장
  - 충전된 전하량으로 논리레벨 (0, 1) 을 결정.



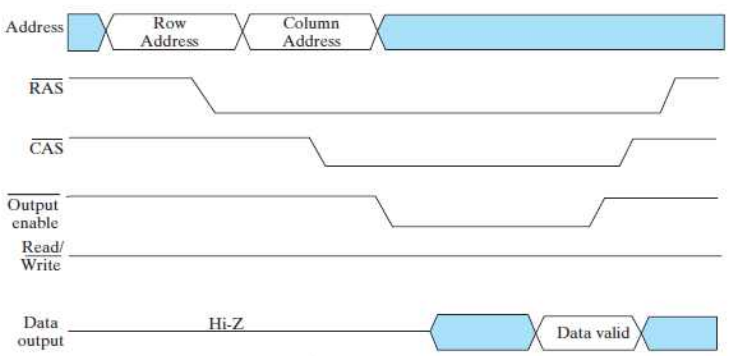
- **특성**
  - 충전된 전하의 누설 (leakage)
    - 전원이 공급된 상태에서도 재충전(refresh)이 필요
      - 재충전 회로 (refresh circuits) 와 재충전 시간 필요
  - 간단한 구조.
    - 비트당 필요 면적이 작아 집적도가 높음.
    - 집적도 향상 → 메모리 공간 확대 → 어드레스 라인 증가 → 입출력 핀 증가 → 패키지 크기 증가 → 제품크기 증가
  - 어드레스 다중화 (multiplexing) : 어드레스 핀 수 절약
    - RAS (Row Addr. Strobe), CAS (Column Addr. Strobe)
  - 저비용, 저속

- **메인 메모리에 주로 사용**

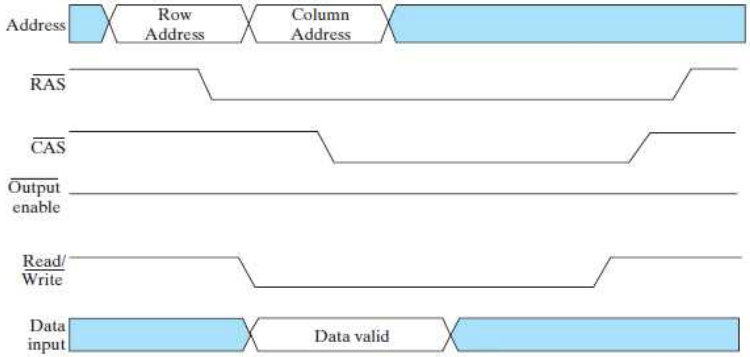
3.1 DRAM 구조


Refresh controller
Refresh counter
Row address
Row address register
Row timing logic
RAS
Row decoder
DRAM bit slice
DRAM bit slice
DRAM bit slice
CAS
Column timing logic
R/W
OE
Input/output logic
Column address
Column address register
Column decoder
Data in/ Data out

3.2 DRAM Timing



Read Timing Diagram



Write Timing Diagram

4. 고성능 DRAM

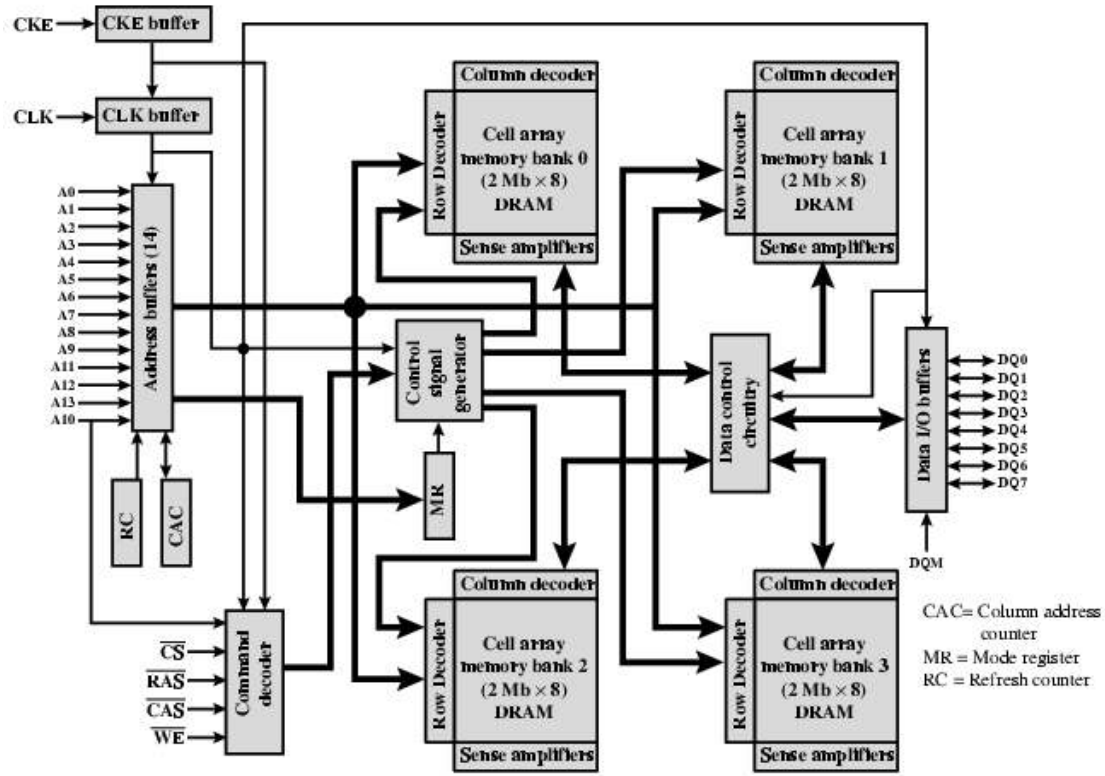
- SDRAM (Synchronous DRAM)
- DDR (Double Data Rate DRAM)
- RDRAM (Rambus DRAM)
- CDRAM (Cache DRAM)

4.1 SDRAM

- **메모리 액세스가 외부 클럭에 동기화**
  - 큰 데이터 블록 전송에 최적.

- **SDRAM 특성**
  - Burst mode 사용
    - 첫 번째 액세스 이후, 어드레스 셋업 시간 (setup time) 과 행, 열 라인의 pre-charge 시간을 절약할 수 있다.
  - Multiple-bank 구조
    - 2 또는 4 개 뱅크로 분리.
    - 온-칩 병렬처리 가능성 향상.
  - Mode register 사용
    - SDRAM 을 특정 용도로 특화 가능.
    - burst type, burst length, nCAS latency 를 제어할 수 있음.

4.1.1 SDRAM 구조


CKE
CKE buffer
CLK
CLK buffer
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A11
A12
A13
A10
Address buffers (14)
RC
CAC
MR
Control signal generator
Command decoder
CS
RAS
CAS
WE
Column decoder
Row Decoder
Cell array memory bank 0 (2 Mb × 8) DRAM
Sense amplifiers
Cell array memory bank 1 (2 Mb × 8) DRAM
Cell array memory bank 2 (2 Mb × 8) DRAM
Cell array memory bank 3 (2 Mb × 8) DRAM
Data control circuitry
Data I/O buffers
DQ0
DQ1
DQ2
DQ3
DQ4
DQ5
DQ6
DQ7
DQM
CAC= Column address counter
MR = Mode register
RC = Refresh counter

4.1.2 SDRAM 제어신호

- **제어신호는 클럭의 rising edge 에 동기**
- **DQM (data mask)**
  - 데이터 입출력을 제한함.

A0 to A13	Address inputs
CLK	Clock input
CKE	Clock enable
$\overline{CS}$	Chip select
$\overline{RAS}$	Row address strobe
$\overline{CAS}$	Column address strobe
$\overline{WE}$	Write enable
DQ0 to DQ7	Data input/output
DQM	Data mask

*** DQM = 1 : Data is not output from/ written to memory.**

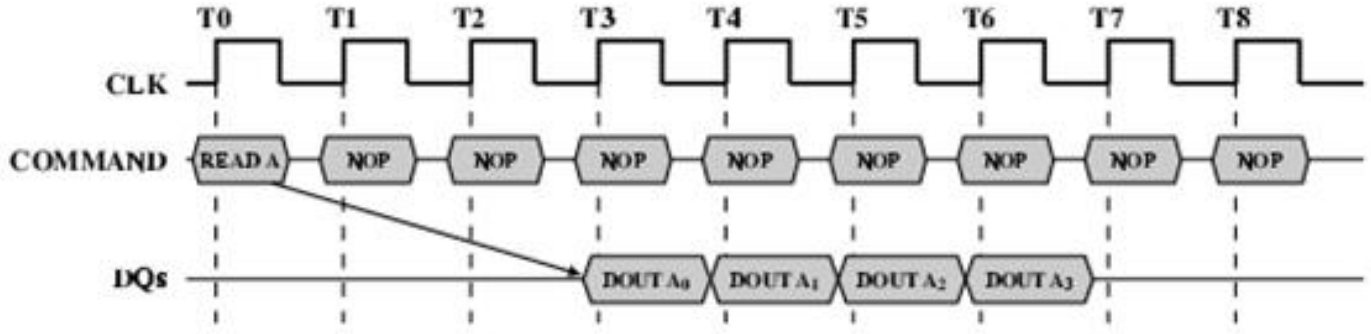
4.1.3 SDRAM 명령어

- 제어 신호와 어드레스 라인을 사용해서 명령어 전달.

/CS	/RAS	/CAS	/WE	BA*n*	A10	A*n*	Command
H	x	x	x	x	x	x	Command inhibit (No operation)
L	H	H	H	x	x	x	No operation
L	H	H	L	x	x	x	Burst Terminate: stop a burst read or burst write in progress.
L	H	L	H	bank	L	column	Read: Read a burst of data from the currently active row.
L	H	L	H	bank	H	column	Read with auto precharge: As above, and precharge (close row) when done.
L	H	L	L	bank	L	column	Write: Write a burst of data to the currently active row.
L	H	L	L	bank	H	column	Write with auto precharge: As above, and precharge (close row) when done.
L	L	H	H	bank	row		Active (activate): open a row for Read and Write commands.
L	L	H	L	bank	L	x	Precharge: Deactivate current row of selected bank.
L	L	H	L	x	H	x	Precharge all: Deactivate current row of all banks.
L	L	L	H	x	x	x	Auto refresh: Refresh one row of each bank, using an internal counter. All banks must be precharged.
L	L	L	L	0 0	mode		Load mode register: A0 through A9 are loaded to configure the DRAM chip. The most significant settings are CAS latency (2 or 3 cycles) and burst length (1, 2, 4 or 8 cycles)

From : Wikipedia

4.1.4 SDRAM 읽기 타이밍



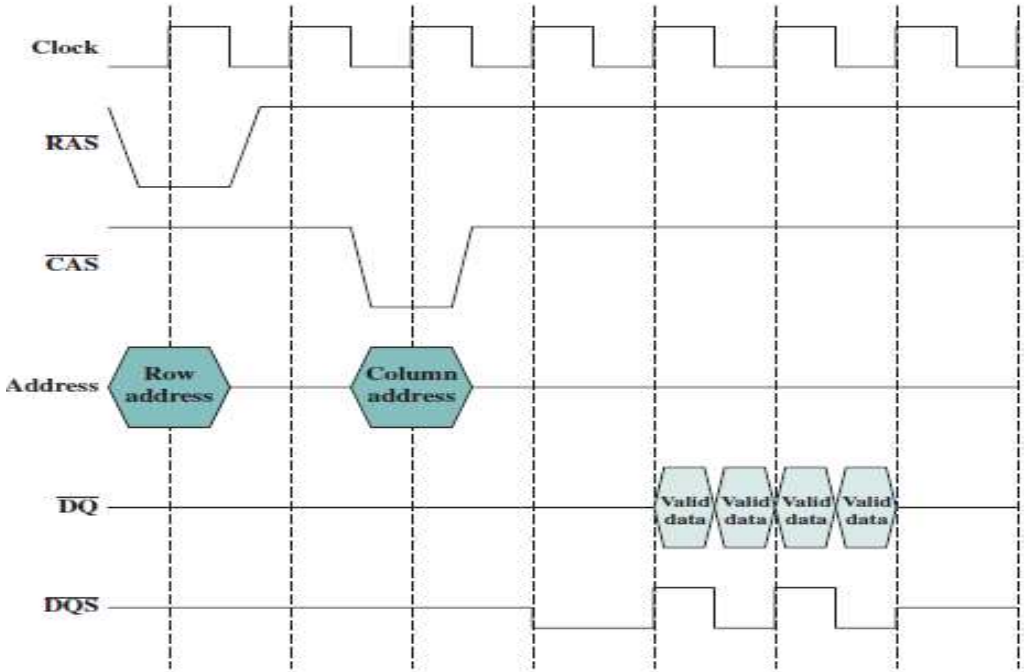
SDRAM Read Timing (Burst Length = 4, $\overline{\text{CAS}}$ latency = 2)

4.2 DDR SDRAM

- **SDRAM 을 개선**
  - SDRAM
    - Single data rate SDRAM
      - 클럭당 1-워드의 데이터를 전송.
      - Rising edge 사용

- **DDR SDRAM (double-data-rate SDRAM)**
  - 클럭당 데이터를 두번전송.
  - Rising edge 와 falling edge 를 모두 사용

4.2.1 DDR SDRAM Timing



RAS = Row address select
CAS = Column address select
DQ = Data (in or out)
DQS = DQ select

Wikipedia 인용

4.2.2 DDR SDRAM

- **DDR1**
  - 4-bank : 2-비트 뱅크 어드레스 (BA0, BA1), 선인출 버퍼 (pre-fetch buffer) 를 4-비트로 확장.
  - DDR-266 (133MHz), DDR-333 (166MHz), DDR-400 (200 MHz)

- **DDR2**
  - 8-bank : 3-비트 뱅크 어드레스 (BA0, BA1, BA2), 선인출 버퍼를 8-비트로 확장.
  - DDR2-400 (200MHz), DDR2-533 (266), DDR2-667 (333MHz), DDR2-800 (400MHz)

- **DDR3**
  - 대역폭 (bandwidth) 와 외부 bus rate 을 2 배로 향상.
  - DDR3-800 (400MHz), DDR3-1066 (533MHz), DDR3-1333 (667MHz), DDR3-1600 (800MHz)

- **DDR4**
  - <1.2V, 2GB/s, 2.133GHz, less power consumption than DDR3.
  - DDR4-4266(2133MHz)

4.2.3 DDR SDRAM 모듈

- **DDR2 Modules**

Standard name	Memory clock (MHz)	Cycle time (ns)	I/O bus clock (MHz)	Data rate (MT/s)	Module name	Peak transfer rate (MB/s)	Timings[4][5] (CL-tRCD-tRP)	CAS latency (ns)
DDR2-400B DDR2-400C	100	10	200	400	PC2-3200	3200	4-4-4 3-3-3	20 15
DDR2-533B DDR2-533C	133.33	7.5	266.67	533.33	PC2-4200*	4266.67	4-4-4 3-3-3	15 11.25
DDR2-667C DDR2-667D	166.67	6	333.33	666.67	PC2-5300*	5333.33	5-5-5 4-4-4	15 12
DDR2-800C DDR2-800D DDR2-800E	200	5	400	800	PC2-6400	6400	6-6-6 5-5-5 4-4-4	15 12.5 10
DDR2-1066E DDR2-1066F	266.67	3.75	533.33	1066.67	PC2-8500*	8533.33	7-7-7 6-6-6	13.125 11.25

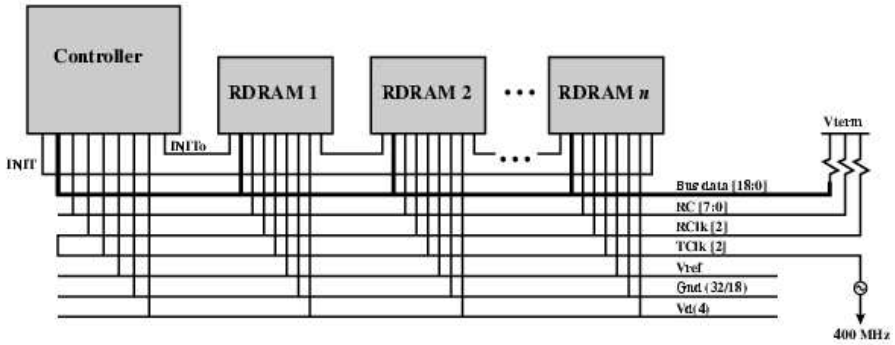
PC2-8500

240-pin DIMM : dual in-line memory module (desktop)
200-pin SO-DIMM : small outline DIMM (notebook)

Wikipedia 인용

4.3 RDRAM

- **RDRAM(RAMBUS DRAM)**
  - Rambus 사에서 개발하고, 인텔에서 Pentium 과 Itanium 프로세서에 채용한 DRAM 구조.
    - 1 Controller + up to 320 RDRAM chips
  - SDRAM 과 경쟁.
- **비동기 블록 프로토콜 사용**
  - 처음 480ns 액세스 후에, 1.6 Gbps 전송가능, 클럭당 2번 데이터 전송.
- **제어버스**
  - RC[7:0] for address & control
  - RAS, CAS, R/W, CE 사용안함.



4.3.1 RDRAM 사양

- **Module spec.**

Designation	Bus width (bits)	Channels	Clock rate (MHz)	Bandwidth (MB/s)
PC600	16	Single	266	1066
PC700	16	Single	355	1420
PC800	16	Single	400	1600
PC1066 (RIMM 2100)	16	Single	533	2133
PC1200 (RIMM 2400)	16	Single	600	2400
RIMM 3200	32	Dual	400	3200
RIMM 4200	32	Dual	533	4200
RIMM 4800	32	Dual	600	4800
RIMM 6400	32	Dual	800	6400

Too expensive !!

Wikipedia 인용

4.4 CDRAM

- **CDRAM (Cache DRAM)**
  - Mitsubishi 사 개발.
  - SRAM 캐시 (16 kb) 를 일반 DRAM 칩에 내장시킨 것.

- **SRAM 캐시 용도**
  - 64-비트 라인들로 구성된 실제 캐시로 사용.
    - 임의 액세스하는 경우에 효과적임.
  - 데이터 블록을 직렬 액세스 (serial access) 하기 위한 버퍼로 사용.
    - 사용례 : 비트-맵 스크린을 갱신 refresh 하기 위해 데이터를 DRAM 에서 SRAM 버퍼로 선인출한 후, 다음 데이터 액세스는 SRAM 버퍼에서 액세스한다.

ROM

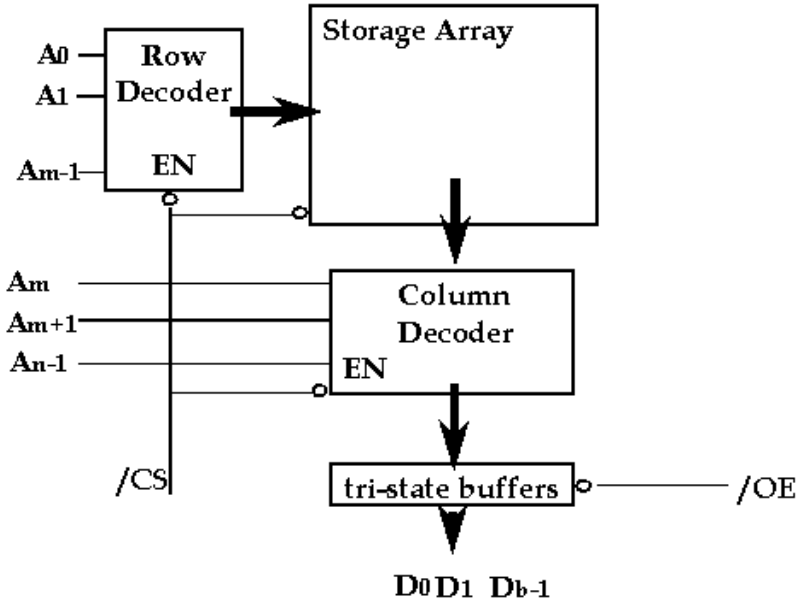
1. ROM 개요

- **ROM (Read Only Memory)**
  - 읽기 전용 메모리

- **비 휘발성 (nonvolatile) 메모리**
  - 전원공급이 없어도 저장된 데이터 값 유지

- **ROM 유형**
  - OTP (One-Time Programable)
    - MROM
    - PROM
  - MTP(Multiple-Time Programmable)
    - EPROM
    - EEPROM
    - Flash

1.1 ROM 구조

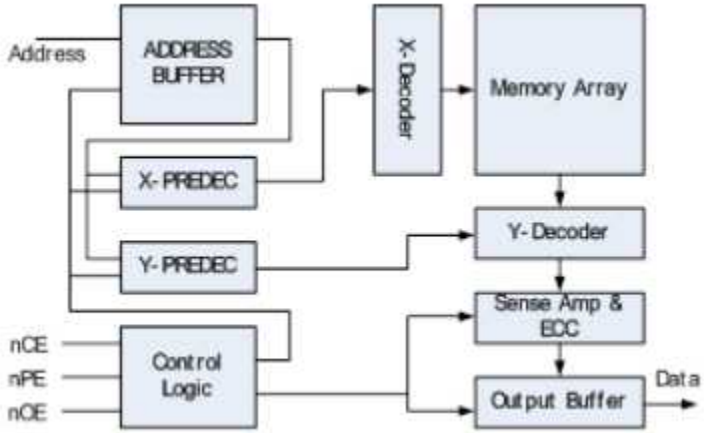

A0
A1
Am-1
Row
Decoder
EN
Storage Array
Am
Am+1
An-1
Column
Decoder
EN
/CS
tri-state buffers
/OE
D0 D1 Db-1

2. MROM

- **Mask ROM (MROM)**
  - 제조과정에서 데이터 값 쓰기
  - 사용자 데이터 변경 불가
    - 영구 데이터 저장용.
  - 소량인 경우 고비용, 대량 생산인 경우 최저 비용.
    - 높은 고정비용

3. PROM

- **Programmable ROM (PROM)**
  - 현장에서 한 번만 쓰여질 수 있다. (OTP : one-time programmable)
  - 제조가 완료된 후에 사용자가 필요에 따라 임의로 쓸 수 있다.
    - 특별한 장치 (ROM writer) 가 필요.
  - 융통성과 편이성 제공 : 대량 생산에 유용



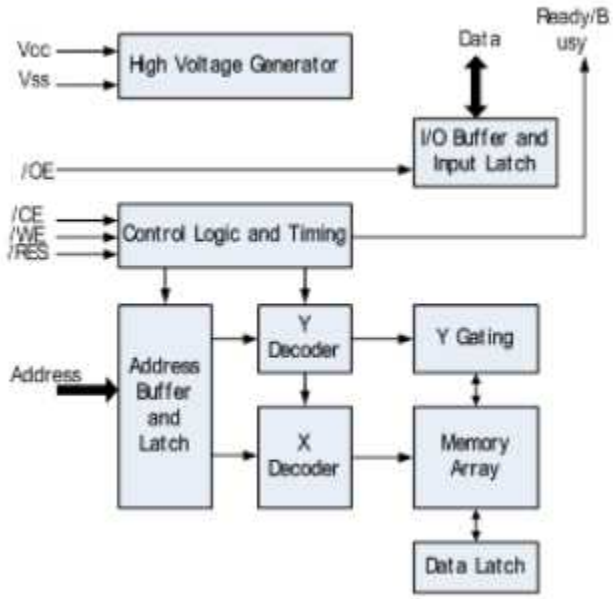
4. EPROM

- **Erasable PROM (EPROM)**
  - 자외선 (UV) 으로 데이터 삭제
  - EPROM writer/programmer 를 사용하여 데이터 저장
  - 데이터 저장후 차광 씰 부착 보관

Gang programmer : EPROM writer for mass production

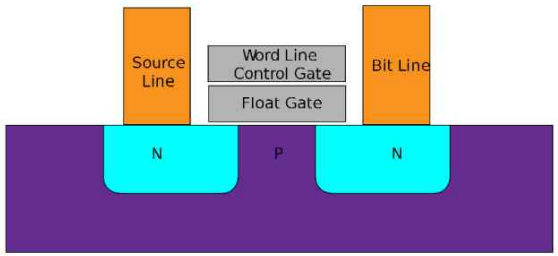
5. EEPROM

- **Electrically Erasable PROM (EEPROM)**
  - 전기적으로 바이트 단위로 데이터 삭제.
  - 읽기보다 쓰기 동작에 많은 시간 소요



6. 플래시 메모리

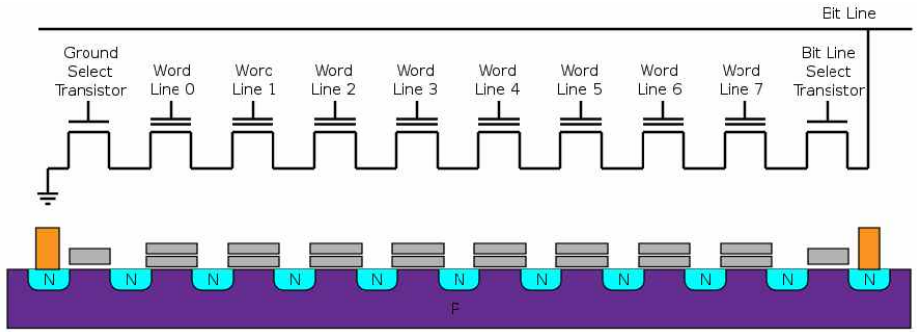
- **EEPROM 의 특수 형태**
  - 비휘발성 메모리
  - EPROM 과 EEPROM 의 중간 정도의 가격과 성능
  - EPROM 보다 빠른 데이터 삭제
  - 높은 집적도.
  - 블록 단위 삭제 : 바이트 단위 삭제 불가



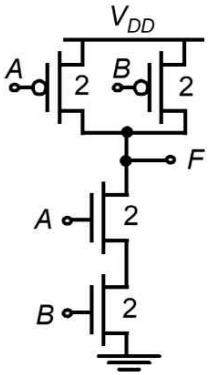
- **단점**
  - 블록단위 삭제
  - 메모리 마모 (wear) : 유한한 삭제-쓰기 동작
    - Wear leveling : 메모리 셀을 균등하게 사용함으로써 메모리 전체 수명을 연장하는 작업
    - Over provisioning : 플래시 메모리에 필요한 작업 (wear leveling, garbage collection, bad block manage 등) 에 필요한 예비 공간을 준비하는 작업.

6.1 NAND형 플래시

- **NAND 플래시**
  - 1987 도시바 NAND 형 플래시 메모리 발표
  - 빠른 삭제 및 쓰기 시간
  - 작은 칩 면적 사용, 긴 수명 (lifetime)
  - 블록 단위 액세스 (not random access)
  - 대용량 저장장치에 적합.



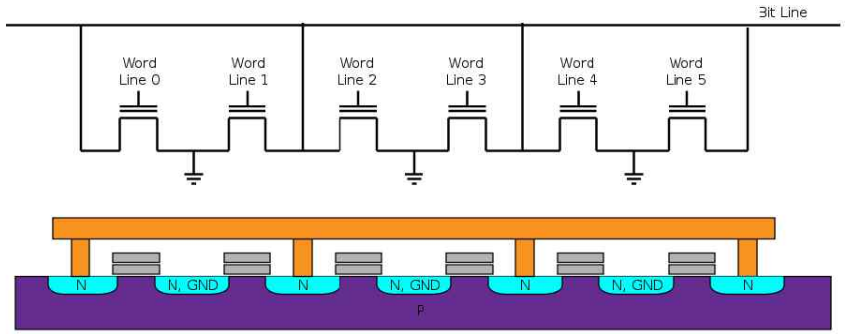
NAND Flash Memory Structure



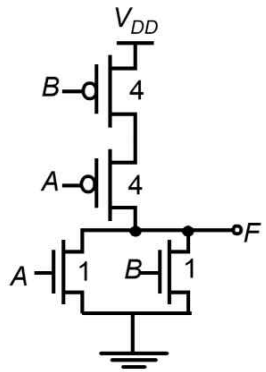
NAND Gate Circuit

6.2 NOR형 플래시

- **NOR 플래시**
  - 1988 인텔 NOR-형 플래시 메모리 발표
  - 임의 액세스 (random access) 가능
  - 프로그램 ROM 에 적합.



NOR Flash Memory Structure



NOR Gate Circuit

6.3 NAND 와 NOR 플래시 메모리 비교

Attribute	NAND	NOR
Main Application	File Storage	Code execution
Storage capacity	High	Low
Cost per bit	Better	
Active Power	Better	
Standby Power		Better
Write Speed	Good	
Read Speed		Good

Type of flash memory	Endurance rating (Erases per block)
SLC NAND	100,000
MLC NAND	5,000 to 10,000 for medium-capacity applications; 1,000 to 3,000 for high-capacity applications
TLC NAND	1,000
SLC (floating-gate) NOR	100,000 to 1,000,000
MLC (floating-gate) NOR	100,000

end