Slide 0

3장. 메모리 시스템의 설계 및 접속

액세스/사이클 시간 및 메모리 대역폭

Slide 1

- 메모리 액세스 시간 (t_a) : 메모리가 주소를 받는 순간부터 데이터 버스를 구동 반응하기까지의 시간
- 메모리 사이클 시간 (t_c) : 연속된 두 번의 읽기 동작 요청 사이의 최소 시간
  - SRAM, ROM : $t_c = t_a$
  - DRAM : $t_c = 2 \times t_a$ (선 충전 시간 필요)
- 메모리 쓰기 시간 : WRITE 명령이 보내진 수간부터 저장 완료될 때까지의 시간
- 메모리 지연시간 : 마이크로프로세서가 메모리 요청후 기다려야되는 시간

액세스/사이클 시간 및 메모리 대역폭 (Cont'd)

Slide 2

- 메모리 부 시스템의 성능
  - 척도
    - 메모리 지연 시간 혹은 메모리 액세스 시간
    - 메모리 대역폭
  - 목표
    - 메모리 지연시간을 줄이고 메모리 대역폭을 늘림

메모리 대역폭

Slide 3

- 메모리의 데이터 전달속도 (메모리 대역폭) : 매초당 메모리에 전달되는 정보의 최대량
- 대역폭 : 초당 전달되는 비트 나 바이트 (비트 / 초, 바이트 / 초)
  - 예) 16Mhz 로 동작하는 80386 (32 비트 데이터 버스, 4clock 버스 사이클)
  - →16 Mbyte/sec
- 대역폭을 늘리는 방법
  - 고 속의 메모리를 사용 (가장 쉬운 방법)
  - 넓은 메모리를 사용
  - 고급 액세스 모드 사용 (니블 모드, 정적 열 모드, 페이지 모드)
  - 캐쉬메모리 사용 (가장 진보된 방법)
  - 메모리 인터리빙

메모리 소자의 종류별 특징

ROM (Read Only Memory)

Slide 4

- 비휘발성 메모리로 ROM, PROM, EPROM, EEPROM 등이 있음
- ROM : 마스크 ROM →제작시 프로그램 , 빠른 속도
- PROM : 한 번 프로그램 가능 →비쌈
- EPROM : 여러번 프로그램 가능, 자외선으로 지움
- EEPROM : 여러번 프로그램 가능, 전기적으로 데이터 지움 →너무 느림, 너무 비쌈
- FLASH 메모리 : 고 속의 읽기/쓰기 가능, 비휘발성, 전기적으로 지움 →혁신적인 메모리

RAM (Random Access Memory)

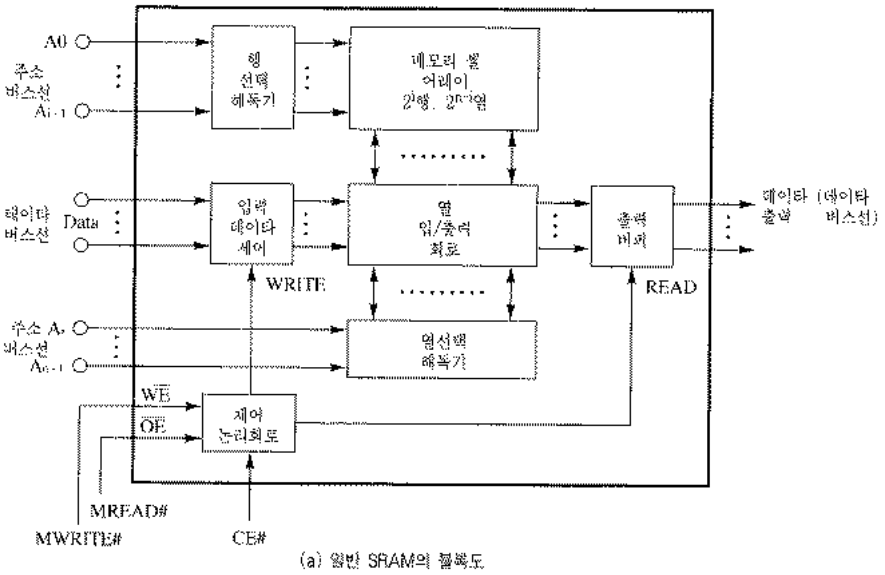
SRAM

Slide 5

- 휘발성 메모리로 정적 RAM (SRAM) 동적 RAM (DRAM) 이 있음
- SRAM
  - Flip-flop 에 데이터 저장
  - 리플래쉬 필요 없음
  - 속도 빠름
  - 외부 캐쉬 메모리로 사용됨
- SRAM 의 블록도 (그림 3.1 a)

SRAM (Cont'd)

Slide 6

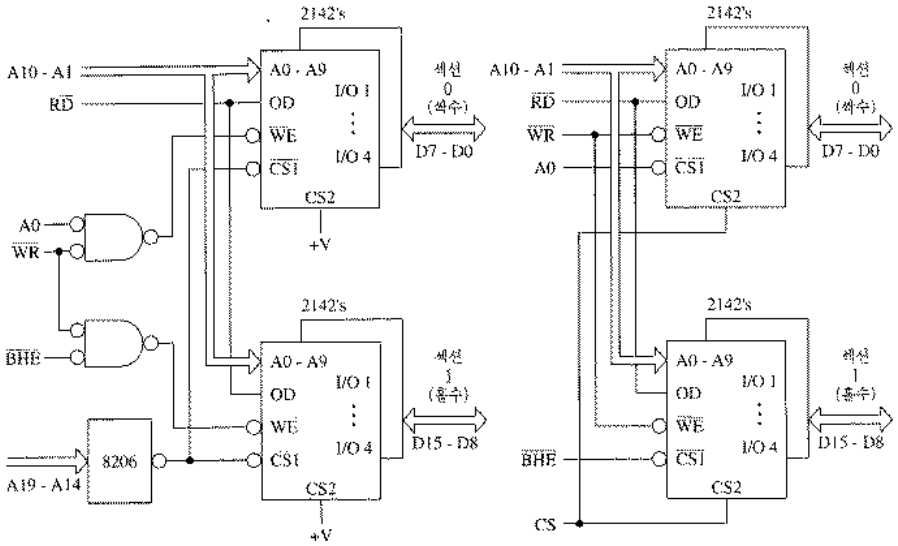


(a) 일반 SRAM의 블록도

SRAM (Cont'd)

- SRAM 으로의 접속 (그림 3.2)

Slide 7



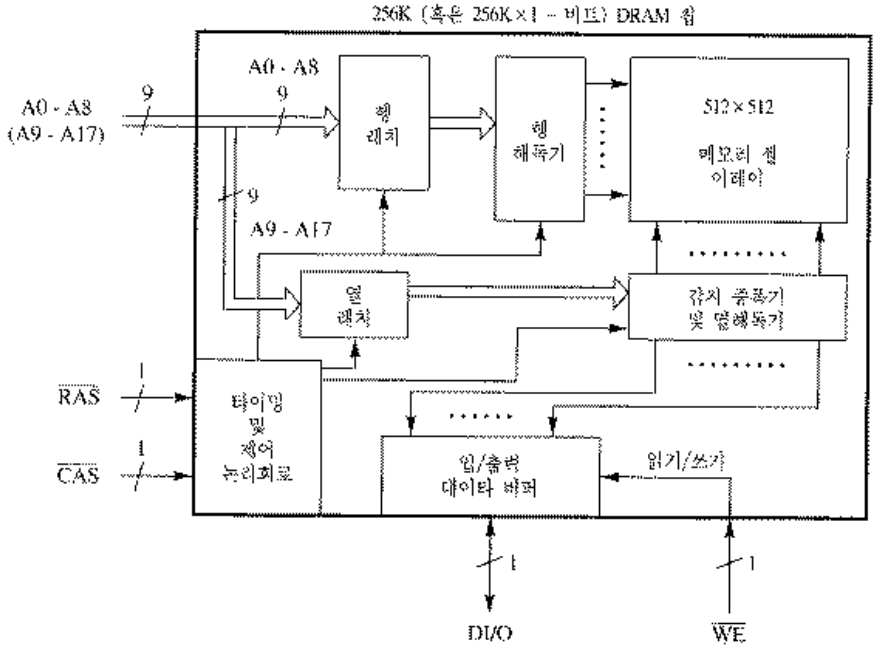
DRAM

Slide 8

- DRAM
  - capacitor 에 데이터 저장 (대용량 가능)
  - 리플래쉬 필요함 (속도 느림)
  - 액세스 시간 단축을 위해 여러 모드로 동작
  - 행/열 주소를 멀티플렉스 하여 입력함 (RAS#, CAS# 사용)
- DRAM 의 블록도 (그림 3.1 b)

DRAM (Cont'd)

Slide 9

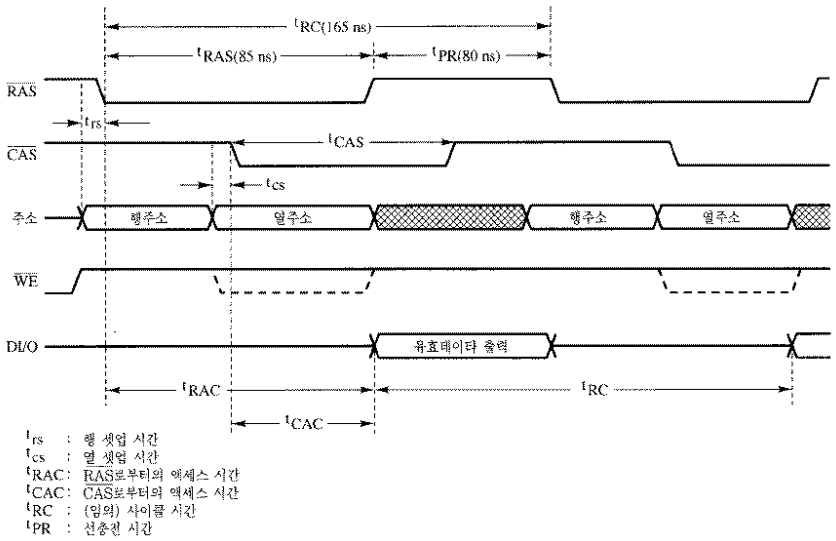


(b) 256K×1 - 비트 DRAM 블록도

DRAM (Cont'd)

Slide 10

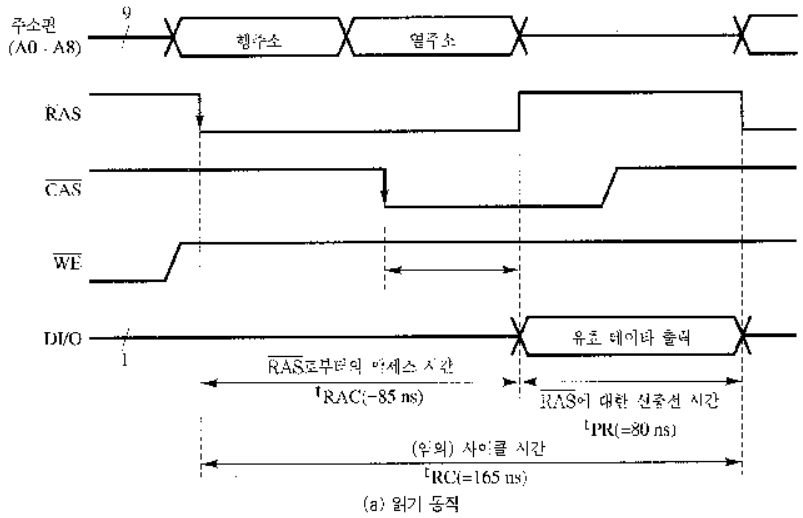
- DRAM 의 액세스 시간과 사이클 시간 (그림 3.3)



DRAM (Cont'd)

Slide 11

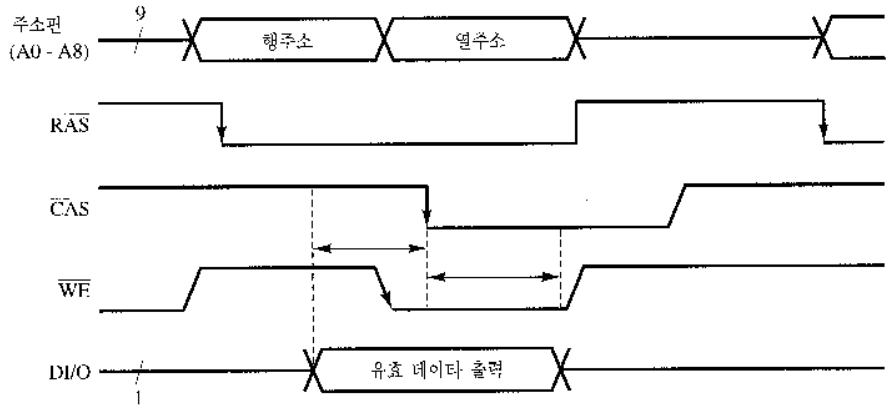
- DRAM 의 읽기 및 쓰기 타이밍 (그림 3.4)



(a) 읽기 동작

DRAM (Cont'd)

Slide 12



(b) 쓰기 동작

DRAM (Cont'd)

Slide 13

- 앞으로의 가정
  - DRAM 의 액세스 시간 : RAS# 으로 부터의 액세스시간 (t_a)
  - $t_a = t_{PR} \rightarrow t_c = 2 \times t_a = 2 \times t_{PR}$
  - 대역폭 예 (50ns 의 DRAM chip 의 대역폭은?)
    - $t_a = 50ns \rightarrow t_c = 100ns \rightarrow$ 10 Mbit/sec
    - 32 비트 구성에서는 (40Mbyte/sec)

DRAM (Cont'd)

Slide 14

- DRAM 의 복구
  - capacitor 로 저장함으로 수 밀리초 동안만 임시적으로 저장됨
  - 주기적으로 데이터 복구 회로 필요
  - 복구시 외부 데이터 액세스 안됨
  - 복구 사이클시 외부 복구 논리회로가 주소 및 관련 신호 줌
  - 복구 사이클과 프로세서 사이클이 동시에 들어왔을 경우 중재회로 필요

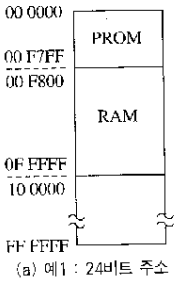
메모리 구성

Slide 15

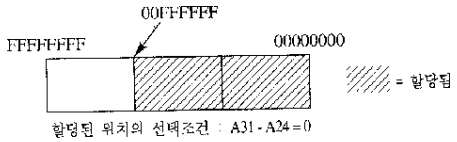
- 워드 길이 결정 : 메모리를 몇 바이트로 구성할 것인가 결정
- 메모리 주소결정 (메모리 맵) : 각 메모리 별로 주소 할당
  - ROM 은 초기화시 지정되게 해야함
  - 주소 할당의 예 (그림 3.5)

메모리 구성 (Cont'd)

Slide 16



(a) 예1 : 24비트 주소



할당된 위치의 선택조건 : A31 - A24 = 0

(b) 예2 : 32비트 주소

장 치	메모리 위치	비 고
PROM	000000 - 00F7FF	MRDC#
RAM	00F800 - 0FFFFF	MRDC#, MWTC#
8비트 입력 포트	100000 - 10FFFF	MRDC#, A28 = 1 (메모리 맵)
8비트 출력 포트	200000 - 2FFFFF	MWTC#, A29 = 1 (메모리 맵)

(c) 예3 : I/O 포트와 칩 인에이블이 있는 메모리 맵

메모리 부 시스템의 설계

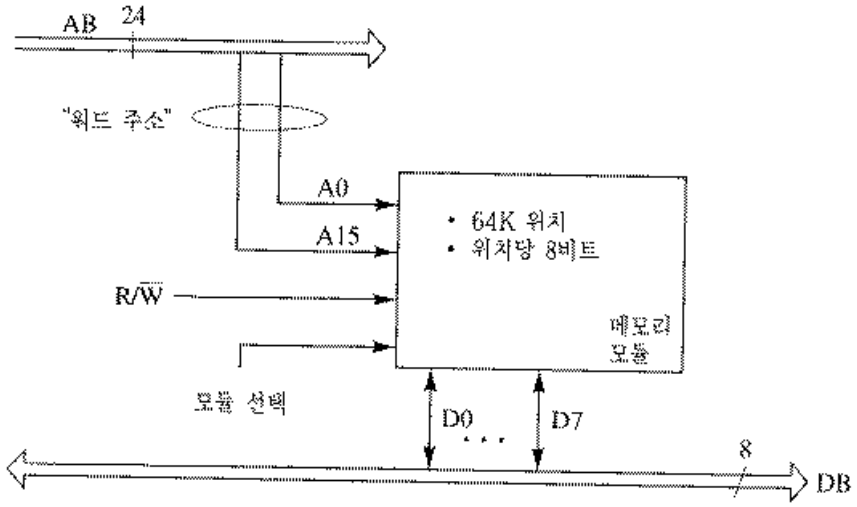
워드 길이 및 용량

Slide 17

- 사용칩 선택
- 워드 길이 결정
- 요구되는 용량 (총 바이트수) 을 만들기 위한 연결
- 상위주소 : 칩선택 신호로 사용
- 하위 주소 : 메모리 칩 내부의 위치 액세스 신호로 사용 (예 그림 3.7)

워드 길이 및 용량 (Cont'd)

Slide 18



(a) 8비트 "워드 길이"를 갖는 64K바이트 메모리 모듈

메모리 워드길이 형성

Slide 19

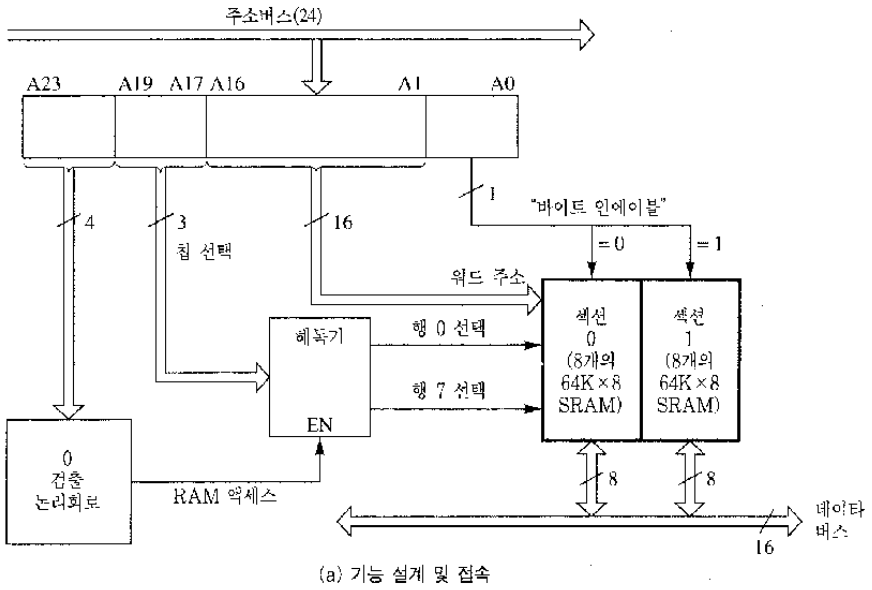
- 메모리 구성 (그림 3.8)
- 공식 : $no.\ of\ chips = \frac{wordlength\ of\ memory}{wordlength\ of\ chip}$
- 메모리의 용량 : $no.\ of\ chips = \frac{necessary\ locations\ of\ memory}{locations\ of\ chip}$
- 예) $2^n \times 2^m$ 비트 칩을 사용 $2^N \times 2^M$ 의 메모리를 만듦
  - 2^{M-m} 개의 칩이 동시에 인에이블 되어야함 (하나의 모듈)
  - 2^{N-n} 개의 모듈이 존재하고 이를 상위 주소를 이용 선택해야함
  - $2^{N-n} \times 2^{M-m}$ 의 이차원 메모리 칩이 배열 되어야함

SRAM 메모리 (설계예)

- 예 1)
  - 64K x 8 비트의 SRAM 을사용
  - 24 비트 주소 16비트 데이터를 갖는 프로세서
  - 총용량 : 1M byte (그림 3.9)

SRAM 메모리 (설계예) (Cont'd)

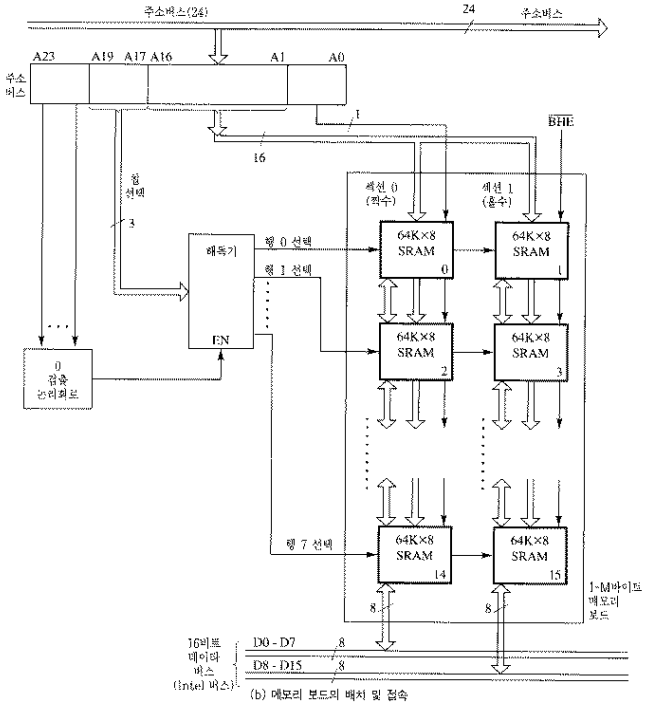
Slide 20



(a) 기능 설계 및 접속

SRAM 메모리 (설계예) (Cont'd)

Slide 21



(b) 메모리 보드의 배치 및 접속

SRAM 메모리 (설계예) (Cont'd)

Slide 22

- 예 2)
  - 4K x 1 비트 SRAM 을사용
  - 80286 사용
  - 총용량 : 8K byte (그림 3.10)

SRAM 메모리 (설계예) (Cont'd)

Slide 23

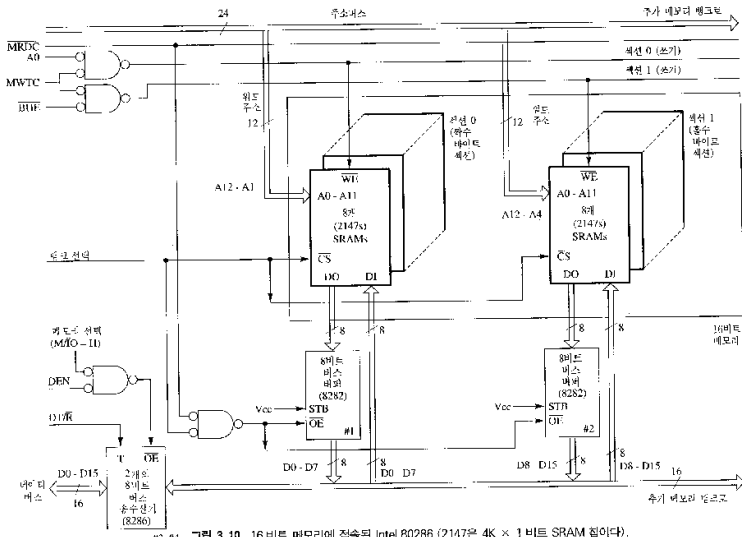


그림 3.10 16 비트 메모리에 접속된 Intel 80286 (2147은 4K × 1 비트 SRAM 칩이다).
([4]에서 인용함. Intel의 허가를 얻어 게재함. (c) 1983 Intel Corp.)

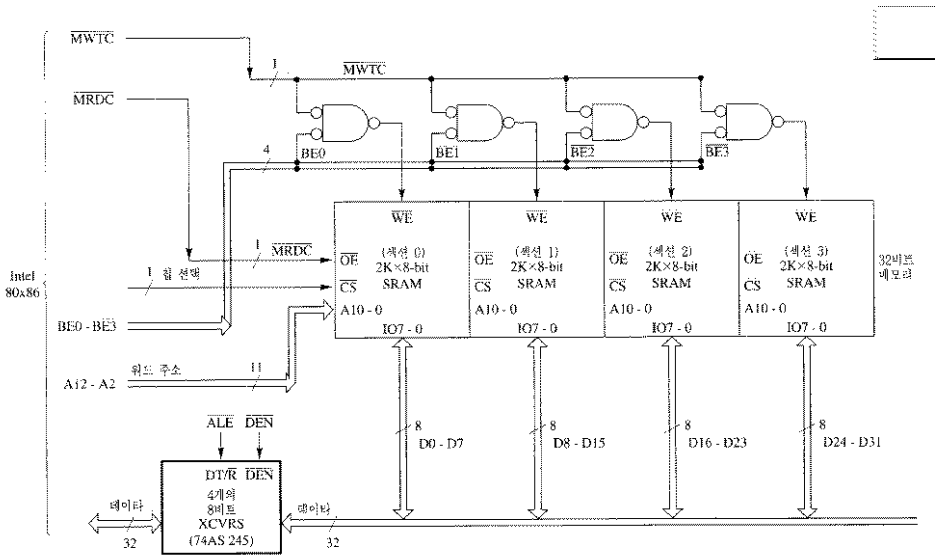
SRAM 메모리 (설계예) (Cont'd)

Slide 24

- 예 4)
  - 2K x 8 비트 SRAM 사용
  - 80X86 사용 (32 비트 메모리로)
  - 총용량 : 8K byte (그림 3.12)

SRAM 메모리 (설계예) (Cont'd)

Slide 25



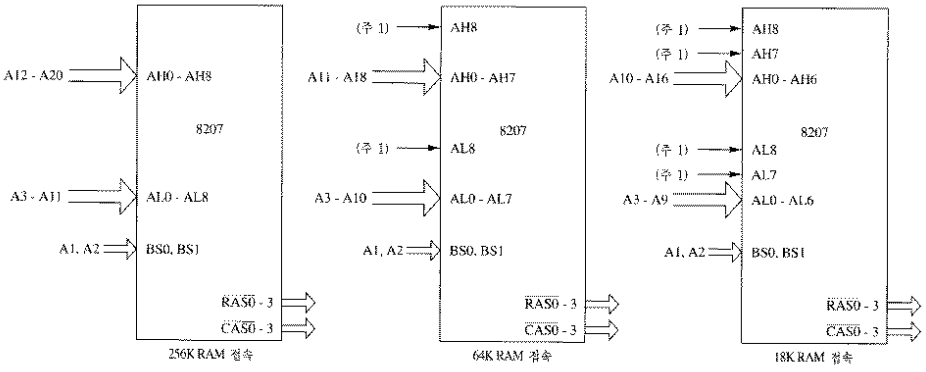
DRAM 메모리 (설계예)

Slide 26

- 8207 DRAM 제어기
  - DRAM 제어에 필요한 신호를 생성하는 칩
  - 인터리브, 비 인터리브 메모리로 256K 비트 DRAM 을 뱅크 4개까지 구동
  - 다른 DRAM 제어기들
    - TI 74ALS6301 : 1M 비트 DRAM 가능 4개 뱅크까지 구동
    - IDT 79R3721 : 1M 16M 비트 DRAM, 페이지 모드 지원, 비인터리브, 2원 인터리브 가능
  - 8207 을 사용한 메모리 접속 (그림 3.14)

DRAM 메모리 (설계예) (Cont'd)

Slide 27



주 :
1. 할당되지 않은 주소핀은 high 또는 low에 연결시켜야 한다.
2. A0와 BS[1:0]를 사용하여 프로세서 워드 내의 한 바이트를 선택한다.
3. 메모리 사이클의 뱅크 인터리빙을 허용하기 위해 연속된 메모리 액세스가 이웃 뱅크로 이루어지도록 뱅크선택 입력으로 하위 주소 비트를 사용한다.

DRAM 메모리 (설계예) (Cont'd)

Slide 28

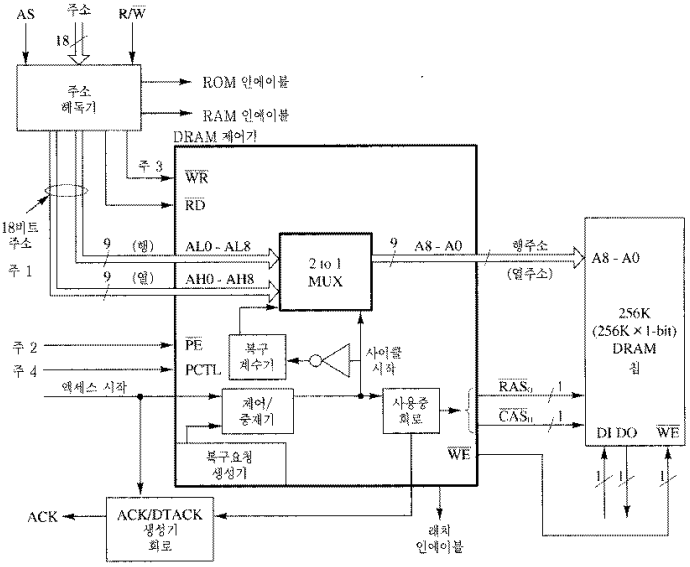
– 8207 의 뱅크 구성 (그림 3.15)

프로그램된 모드	뱅크선택 입력 BS1	BS0	RAS#/CAS#쌍 할당	
4뱅크 사용됨	0	0	RAS0#, CAS0# 뱅크 0으로	디폴드
	0	1	RAS1#, CAS1# 뱅크 1로	
	1	0	RAS2#, CAS2# 뱅크 2로	
	1	1	RAS3#, CAS3# 뱅크 3으로	
3뱅크 사용됨	0	0	RAS0#, CAS0# 뱅크 0으로	
	0	1	RAS1#, CAS1# 뱅크 1로	
	1	0	RAS2#, CAS2# 뱅크 2로	
	1	1	Bank 3 사용안됨	
2뱅크 사용됨	0	0	RAS0,1#, CAS0,1# 뱅크 0으로	
	0	1	RAS2,3#, CAS2,3# 뱅크 1로	
	1	0	Bank 2 사용안됨	
	1	1	Bank 3 사용안됨	
1뱅크 사용됨	0	0	RAS0 - 3#, CAS0 - 3# 뱅크 0으로	
	0	1	Bank 1 사용안됨	
	1	0	Bank 2 사용안됨	
	1	1	Bank 3 사용안됨	

DRAM 메모리 (설계예) (Cont'd)

Slide 29

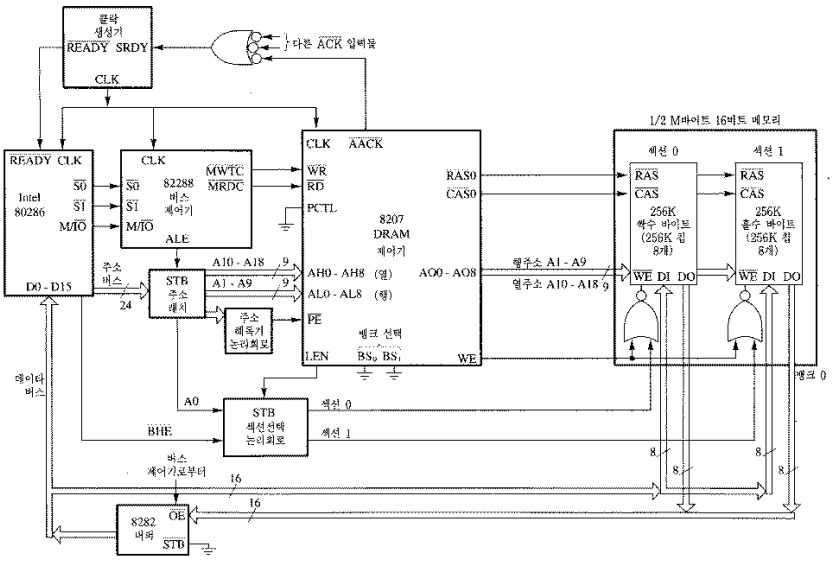
– 8207 을 사용한 접속 (그림 3.16)



DRAM 메모리 (설계예) (Cont'd)

Slide 30

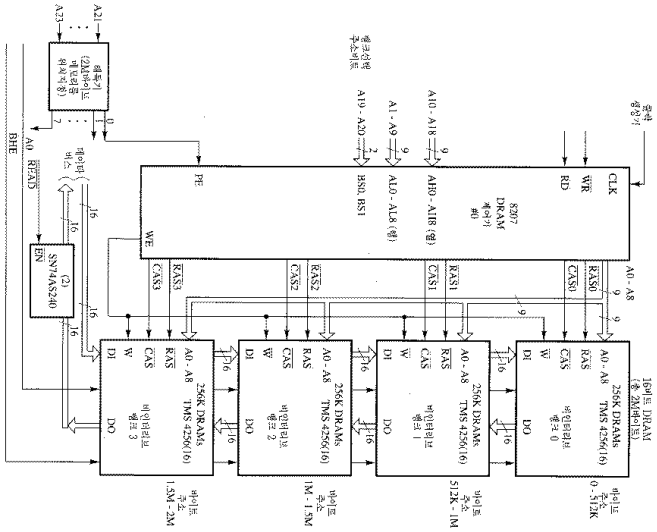
– 16비트 메모리에 접속 (그림 3.17)



DRAM 메모리 (설계예) (Cont'd)

Slide 31

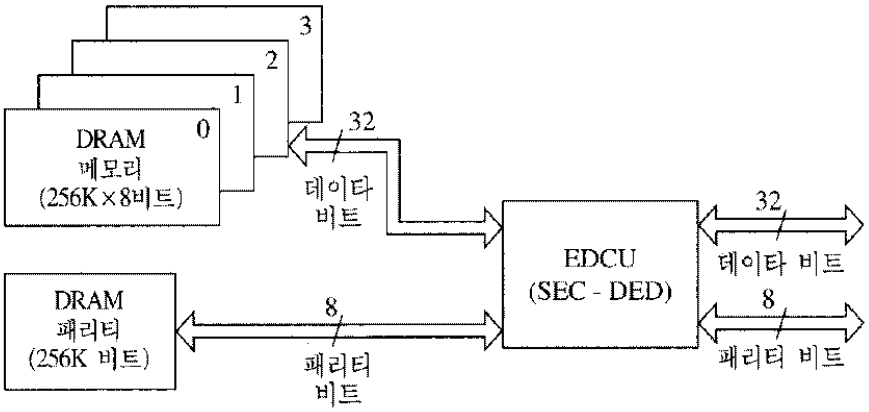
– 4개의 비인터리브 DRAM 뱅크의 접속 (그림 3.18)



DRAM 메모리 (설계예) (Cont'd)

Slide 32

- 이중 포트 메모리
  - 두 개의 다른 장치에서 하나의 메모리를 액세스 할 수 있는 메모리
- 오류검출 및 교정 장치 (그림 3.19)



메모리 인터리빙

Slide 33

인터리빙의 목적

- 저 속의 메모리 칩을 사용 고 속의 메모리 속도를 얻음
- DRAM 의 선 충전 영향을 없애기 위해
- 명령어 선 인출 및 버스 전달 모드의 실행을 위해
- 메모리 대역폭을 증가시키기 위해

인터리빙의 동기

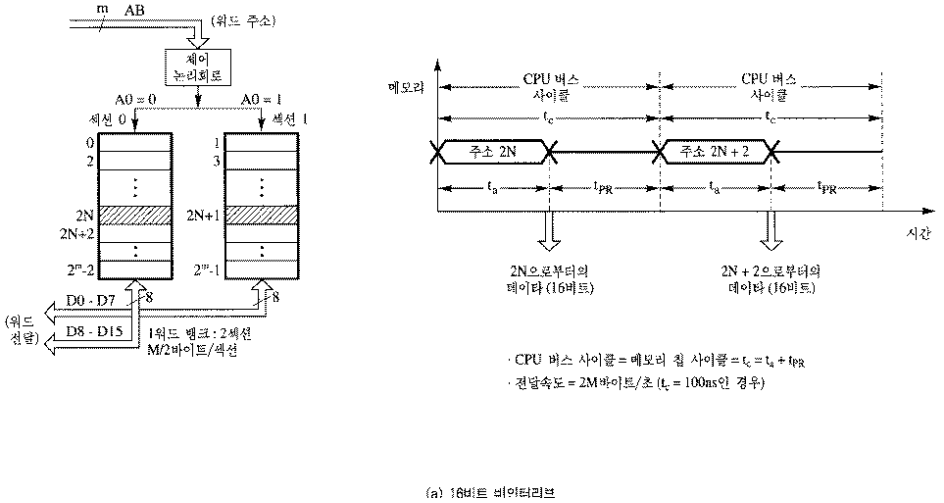
- 프로그램은 인접한 메모리를 액세스 하는 경향이 있다

  → 인접한 주소를 다른 메모리 뱅크로 설계

인터리빙의 동기 (Cont'd)

Slide 34

- 2원 인터리빙 (그림 3.23)

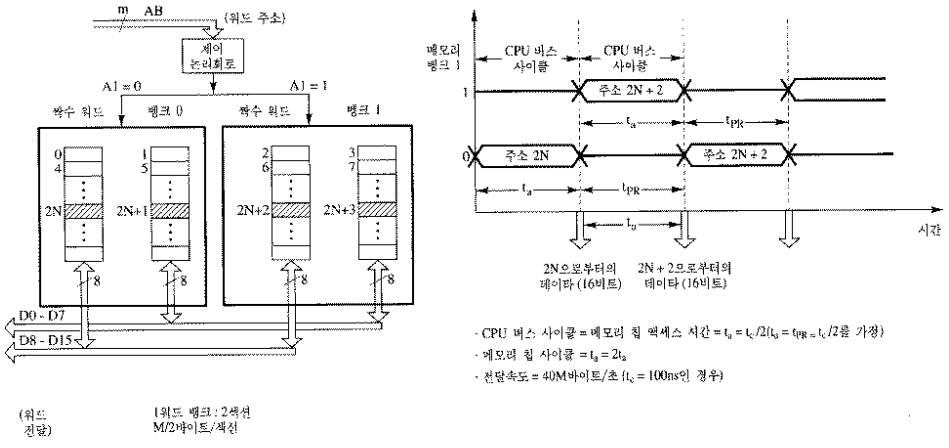


(a) 16비트 비인터리브

인터리빙의 동기 (Cont'd)

Slide 35

- 16비트 워드 인터리빙 (그림 3.23)

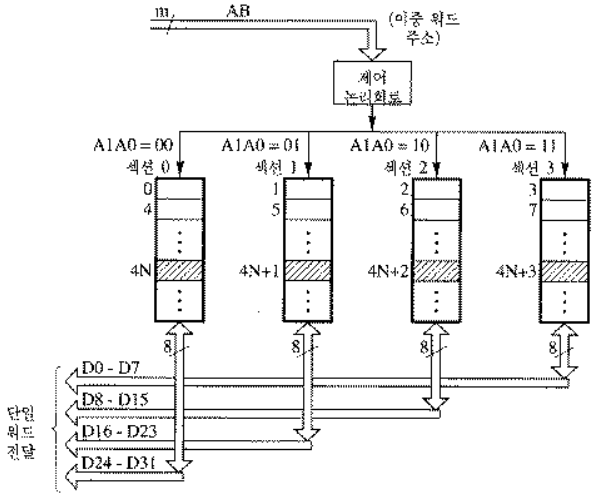


(b) 2원 워드 인터리브

인터리빙의 동기 (Cont'd)

Slide 36

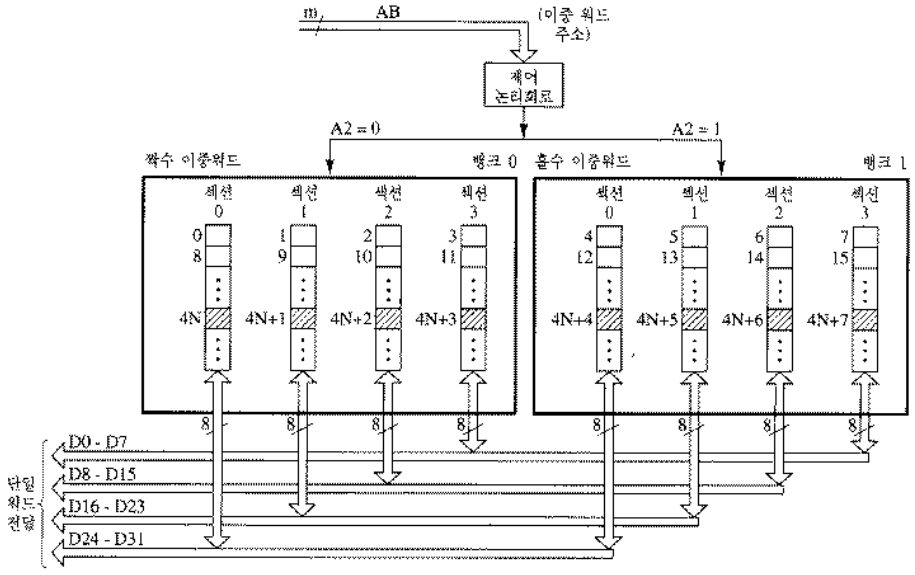
- 32비트 이중워드 인터리빙 (그림 3.24)



(a) 32비트 비인터리브 : 전달속도 = 40M바이트 / 초 (t_C = 100ns인 경우)

인터리빙의 동기 (Cont'd)

Slide 37

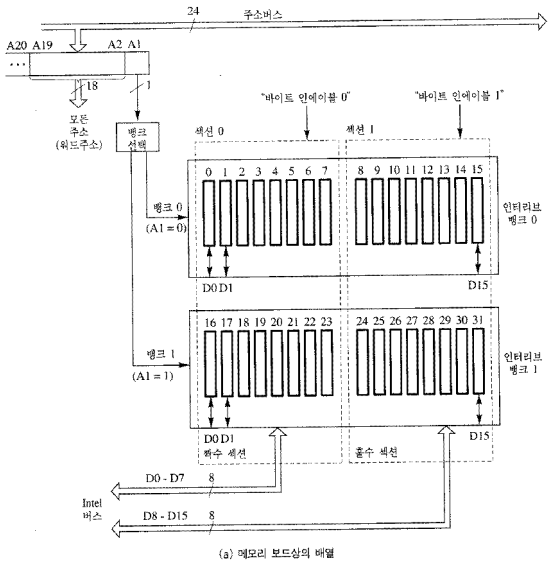


(b) 2웨 이중 워드 인터리브 : 전달속도 = 80M바이트 / 초 (t_C = 100ns인 경우)

인터리브 메모리의 설계예

Slide 38

- 2원 워드 인터리빙 (그림 3.26)

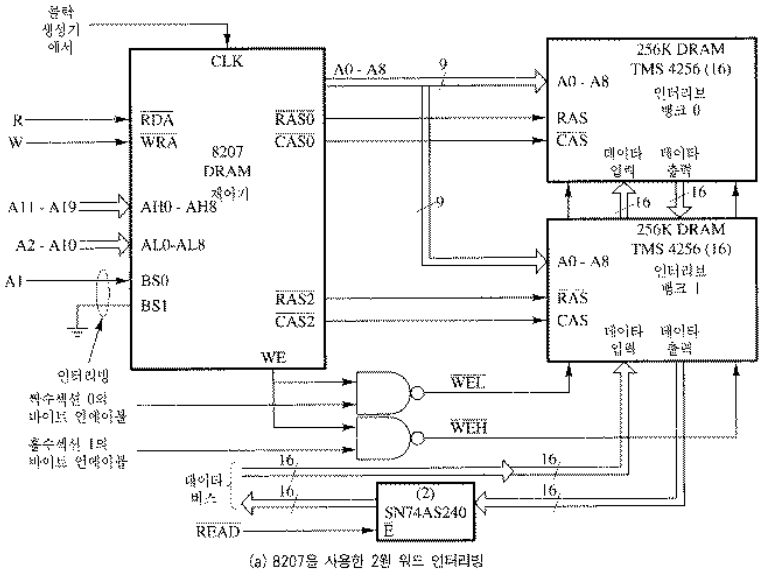


(a) 메모리 보드상의 배열

인터리브 메모리의 설계예 (Cont'd)

Slide 39

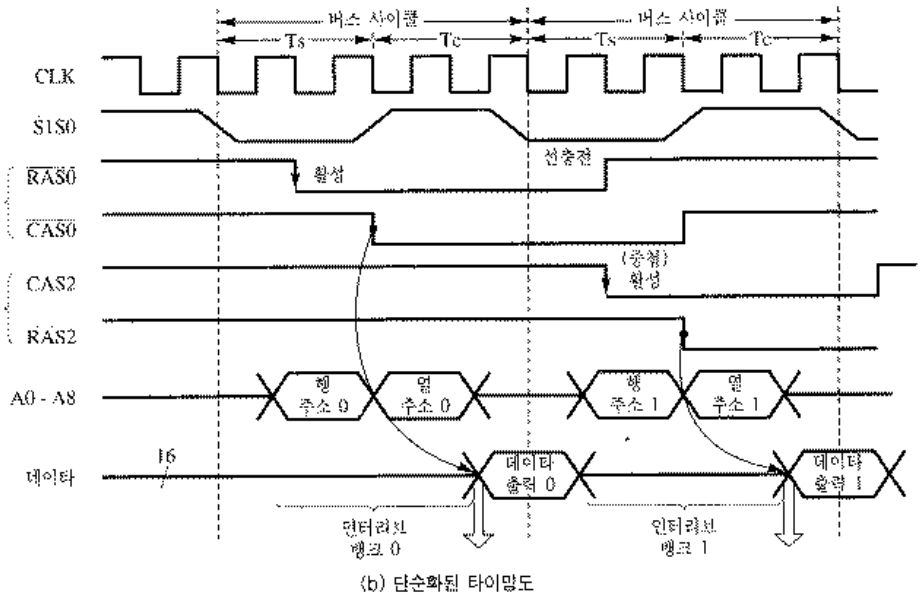
- 2원 워드 인터리브 메모리에 DRAM 제어기 사용 (그림 3.27)



(a) 8207을 사용한 2원 워드 인터리빙

인터리브 메모리의 설계예 (Cont'd)

Slide 40

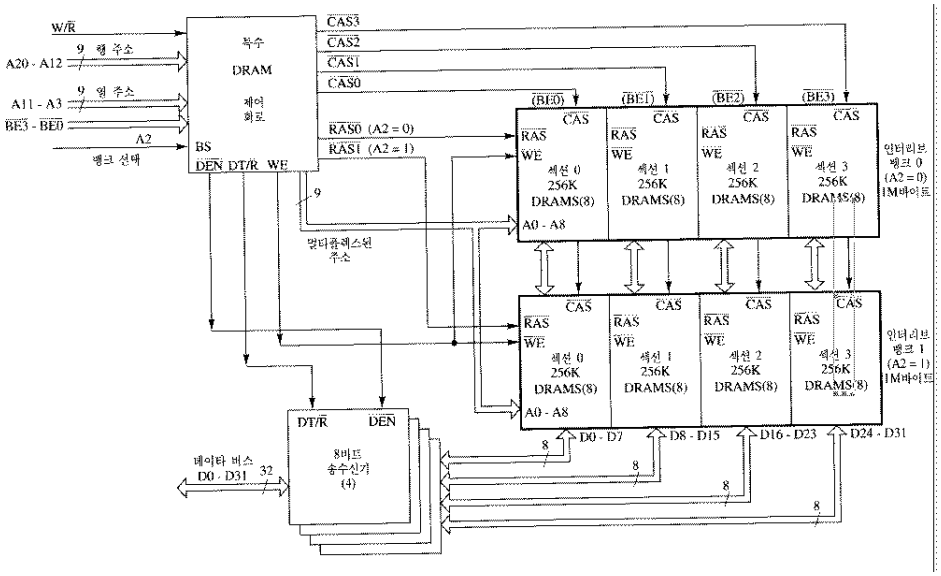


(b) 단순화된 타이밍도

인터리브 메모리의 설계예 (Cont'd)

Slide 41

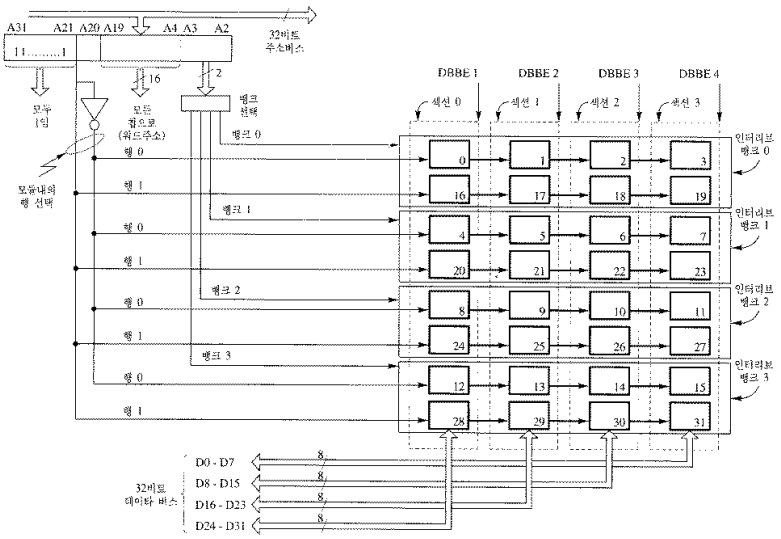
- 2원 이중워드 인터리빙 (80386) (그림 3.28)



인터리브 메모리의 설계예 (Cont'd)

Slide 42

- 4원 이중워드 인터리빙 (그림 3.29)



(a) 메모리 보드상의 배치

메모리 시간 계산

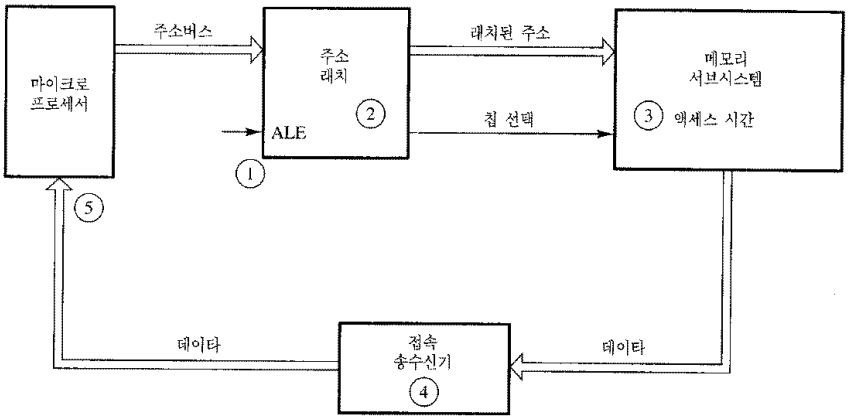
Slide 43

- 프로세서와 메모리의 속도 맞추기
  - t_c : 메모리 사이클시간
  - t_{cpu} : 마이크로프로세서의 주소사이클

    → N 원 메모리 인터리빙 필요 : $N = \frac{t_c}{t_{cpu}}$
- 메모리 지연시간 및 메모리 액세스 시간의 계산
- 메모리 모델 (그림 3.31)

메모리 시간 계산 (Cont'd)

Slide 44



지연:
① = ALE활성 지연
② = 래치의 지연
+ ③ = 액세스 시간
④ = 송수신 지연
⑤ = CPU셋업시간

합 = 메모리 지연시간 L (주소가 나온 시간부터 데이타가 CPU로 들어갈 때 까지 걸린 시간)

다른 지연요소(그림 3.17 참조):
· 주소 해독기(주소래치 후)
· DRAM 제어기 (DRAM 메모리 앞)
· 데이타 버퍼 (메모리의 출력)

(a) 하드웨어 접속에 의한 지연