# 병렬컴퓨터 구조
# (제 14주 차)

서울사이버대학교

오 창 환

# 학습 목표



- 병렬처리가 가능하기 위한 조건, 병렬컴퓨터의 분류 등을 설명할 수 있다.
- 배열 프로세서의 구조를 설명할 수 있다.
- 다중프로세서시스템 구조를 설명할 수 있다.

# 학습 내용



- 병렬처리가 가능하기 위한 조건, 병렬컴퓨터의 분류
- 배열 프로세서의 구조
- 다중프로세서시스템 구조

# 병렬처리가 가능하기 위한 조건 (1)



- 컴퓨터를 이용하는 응용 분야들 중에는 현존하는 초고속 컴퓨터의 성능으로도 처리 시간이 매우 오래 걸리는 것들이 많은데, 예를 들어 인공지능, 로보틱스, 신호처리, 유체 역학, 일기 예보, 입자 물리학, 우주 과학 분야 등임.
- 컴퓨터시스템의 가장 중요한 구성 요소인 CPU, 즉 프로세서의 속도가 계속 향상되고는 있지만, 필요한 정도의 시스템 성능을 얻기에는 여전히 부족함으로 최근 대부분의 고성능 컴퓨터시스템의 설계에서는 성능 향상을 위한 방법으로서 병렬처리 기술이 널리 사용되고 있음.
- 병렬처리(parallel processing)는 다수의 프로세서들이 여러 개의 프로그램들 혹은 한 프로그램의 분할된 부분들을 분담하여 동시에 처리하는 기술을 말함.

# 병렬처리가 가능하기 위한 조건 (2)



- 병렬처리가 가능하기 위한 조건은 아래와 같음.
  * 많은 수의 프로세서들로 하나의 시스템을 구성할 수 있도록
    작고 저렴하며 고속인 프로세서들의 사용이 가능해야 함.
  * 한 프로그램을 여러 개의 작은 부분들로 분할하는 것이 가능해야 하며,
    분할된 부분들을 병렬로 처리한 결과가 전체 프로그램을 순차적으로 처리한 경우와
    동일한 결과를 얻을 수 있어야 함.
- 상기의 첫 번째 조건은 반도체 기술의 발달로 인하여 저렴한 가격으로 수백 개 이상의 프로세서들을 한 시스템 내에 통합할 수 있게 되어서 만족시킬 수 있음.
  그러나 두 번째 조건을 만족시키기 위해서는 다음과 같은 문제들을 해결해야 함.
  * 문제 분할(problem partition)
  * 프로세서 간 통신(inter-processor communication)

# 병렬처리가 가능하기 위한 조건 (3)



- 문제 분할이란 병렬처리를 위하여 하나의 문제(혹은 프로그램)를 여러 개로 나누는 것을 말하지만, 프로그램들 중에는 반드시 순차적으로 처리되어야 하는 것들도 있기 때문에 병렬처리가 근본적으로 불가능한 경우도 있음.
  또한 많은 수의 프로세서들이 제공되더라도 프로그램을 그 수만큼 분할할 수가 없는 경우에는 프로세서의 이용률(utilization)이 낮아져서 원하는 만큼의 성능 향상을 얻을 수가 없기 때문에 문제 분할은 매우 중요함.
- 하나의 프로그램이 여러 개의 작은 부분들로 나누어져서 서로 다른 프로세서들에 의해 처리되는 경우에 프로세서들 간에는 데이터 교환을 위한 통신이 필요하게 됨.
  그런데 프로세서의 수가 증가하면 통신 선로의 수도 그만큼 더 많아지고, 인터페이스를 위한 하드웨어도 복잡해 짐.
  그에 따라 통신 입출력 동작을 제어하기 위한 소프트웨어 오버헤드(software overhead)와 하드웨어상의 지연 시간 때문에 통신에 소모되는 시간이 길어져서 시스템의 성능 향상에 한계가 있게 됨.

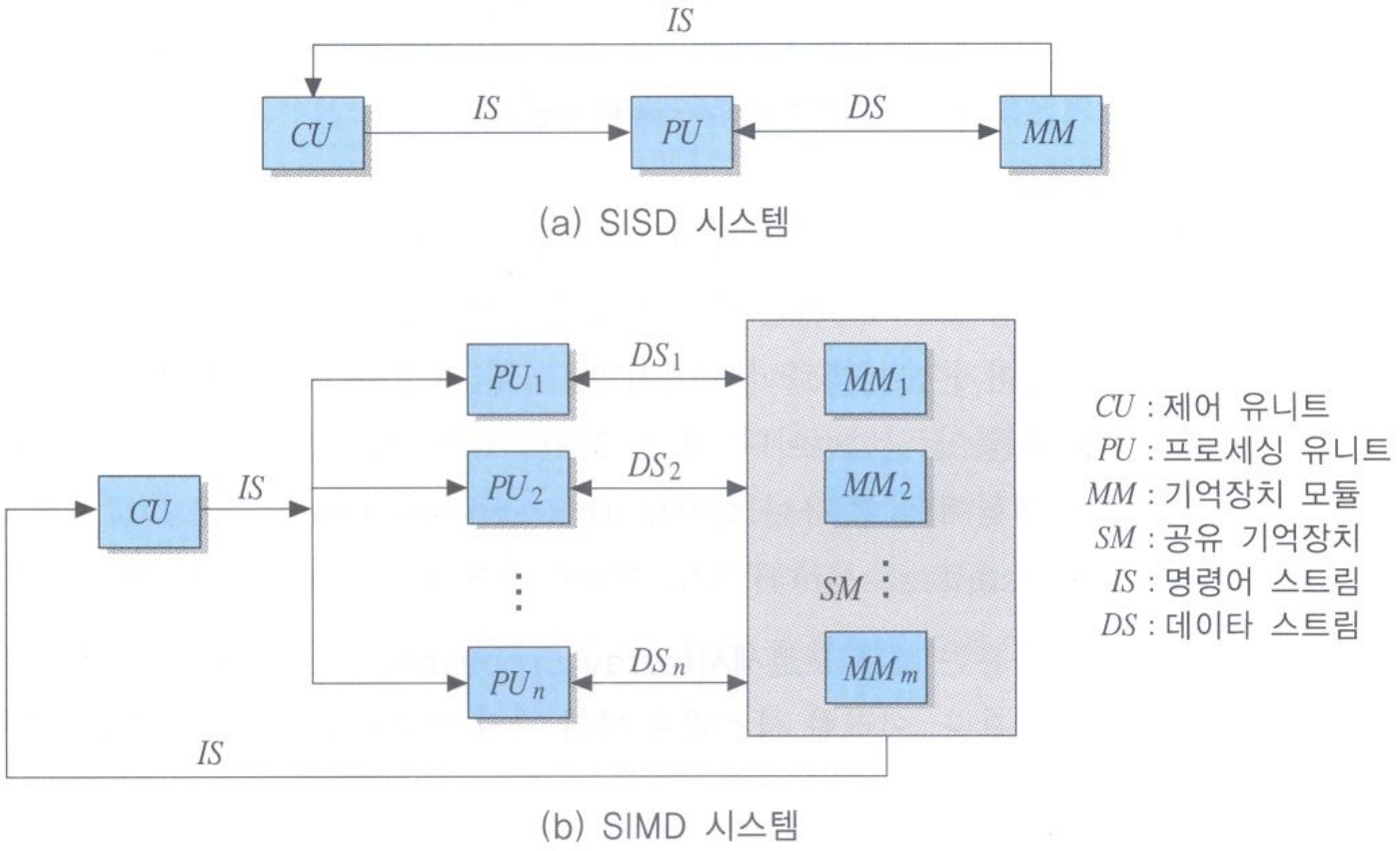
# 병렬컴퓨터의 분류 (1)



- 컴퓨터시스템을 구조적 특징에 따라 분류하는 방식으로는 Flynn의 분류가 가장 널리 사용되고 있는데, 이 분류 방식에서는 프로세서들이 처리하는 명령어 및 데이터의 스트림의 수에 따라 디지털 컴퓨터를 네 가지로 분류하고 있음.
- 스트림에는 명령어 스트림과 데이터 스트림이 있는데, 명령어 스트림(instruction stream)은 프로세서에 의해 실행되기 위하여 순서대로 나열된 명령어 코드들의 집합을 의미하고, 데이터 스트림(data stream)은 그 명령어들을 실행하는 데 필요한 순서대로 나열된 데이터들의 집합을 의미함.
- 명령어와 데이터 스트림을 처리하기 위한 하드웨어 구조에 따른 Flynn의 분류는 다음과 같음.
  * 단일 명령어 스트림 – 단일 데이터 스트림 (SISD)
  * 단일 명령어 스트림 – 복수 데이터 스트림 (SIMD)
  * 복수 명령어 스트림 – 단일 데이터 스트림 (MISD)
  * 복수 명령어 스트림 – 복수 데이터 스트림 (MIMD)

# 병렬컴퓨터의 분류 (2)

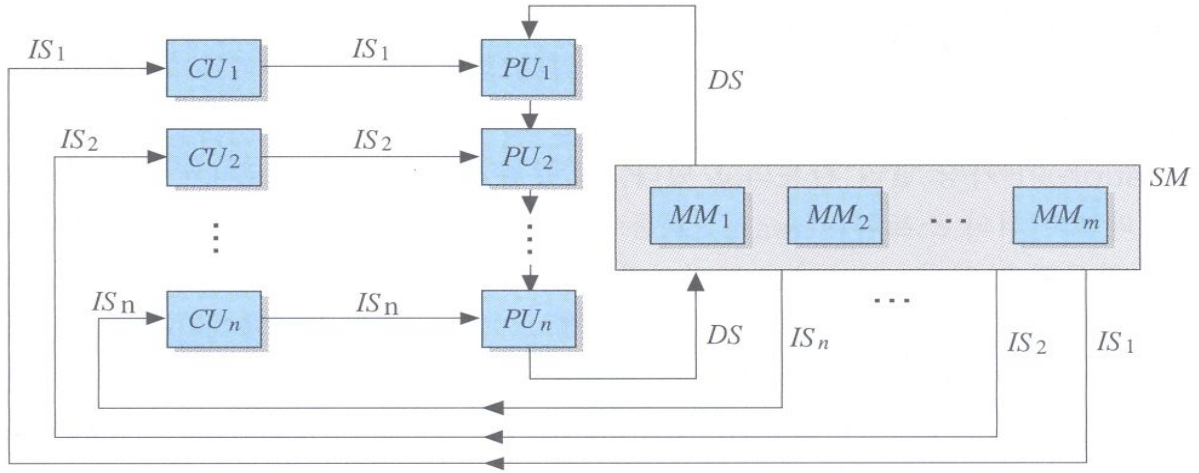




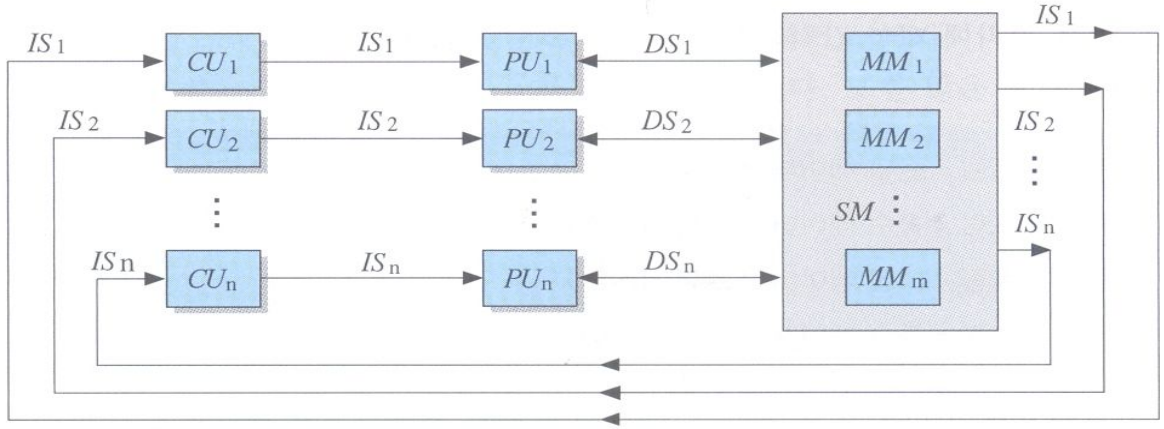

• Flynn의 분류 방식에 따른 컴퓨터시스템의 구성도

# 병렬컴퓨터의 분류 (3)






(c) MISD 시스템




(d) MIMD 시스템



• Flynn의 분류 방식에 따른 컴퓨터시스템의 구성도

# 병렬컴퓨터의 분류 (4)



- SISD 조직 : 이 조직은 한 번에 한 개씩의 명령어와 데이터를 순서대로 처리하는 단일 프로세서 시스템에 해당함.
  이러한 시스템에서는 명령어가 순서대로 실행되지만,
  실행 과정은 파이프라이닝(pipelining) 되어 있는 것이 일반적임.
- SIMD 조직 : 이 분류는 배열 프로세서(array processor)라고도 하며,
  여러 개의 프로세싱 유니트(PU)들로 구성되고,
  PU들의 동작은 하나의 제어 유니트에 의해 통제됨.
  제어 유니트는 명령어를 해독하고,
  그 실행을 위한 제어 신호를 모든 PU들로 동시에 보냄.
  그에 따라 PU들은 동일한 연산을 수행하게 되지만,
  각 연산에서 처리하는 데이터는 서로 다름.
  결과적으로 모든 PU들은 하나의 명령어 스트림을 실행하지만
  여러 개의 데이터 스트림들을 동시에 처리하게 되는 것임.

# 병렬컴퓨터의 분류 (5)



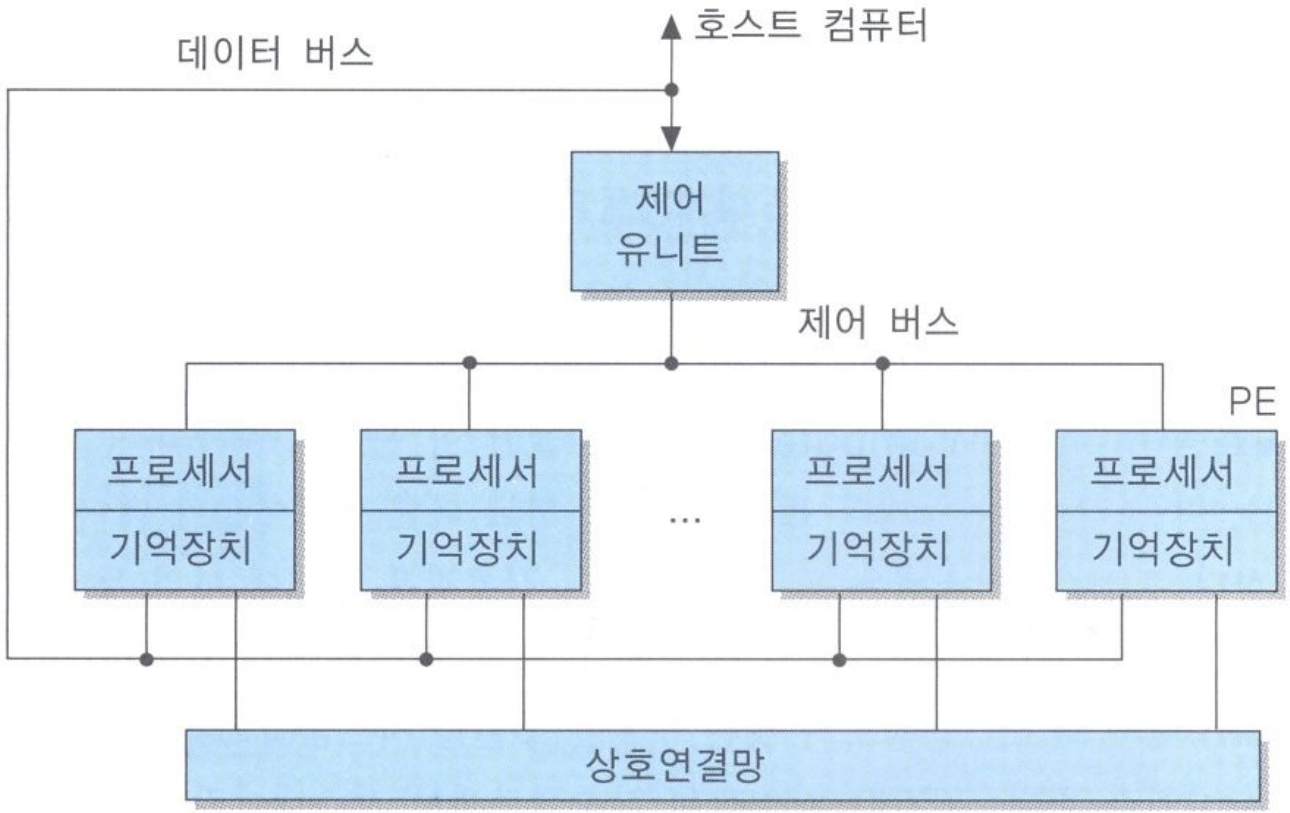
- MISD 조직 : 이 조직에서는 한 시스템 내에 n개의 프로세서들이 있고, 각 프로세서들은 서로 다른 명령어들을 실행하지만, 처리하는 데이터들은 하나의 스트림 임. 프로세서들이 파이프라인 형태로 연결되고, 한 프로세서가 처리한 결과가 다음 프로세서로 보내져 다른 연산이 수행되는 방식임. 그러나 이 조직은 설계자들의 관심을 끌지 못하고 있으며, 실제 구현된 경우도 거의 없음.
- MIMD 조직 : 대부분의 다중프로세서 시스템(multiprecessor system)들과 다중컴퓨터 시스템(multiple-computer system)들이 이 분류에 속함. 이 조직에서는 n개의 프로세서들이 서로 다른 명령어들과 데이터들을 처리함. MIMD 시스템은 프로세서들간의 상호작용 정도에 따라 두 가지로 나누어지는데, 그 정도가 높은 구조를 밀결합 시스템(tightly-coupled system)이라 하고, 그 정도가 낮은 구조를 소결합 시스템(loosely-coupled system)이라 함.
- 병렬컴퓨터는 프로그램 코드와 데이터를 병렬로 처리하는 시스템이므로 SIMD 조직과 MIMD 조직이 이에 해당함. SIMD는 배열 프로세서라 하고, MIMD는 다중프로세서 시스템이라고 함.

# 2 교시

# 배열 프로세서의 구조 (1)



• 배열 프로세서는 시스템 내의 모든 프로세싱 요소(processing element: PE)들이 하나의 제어 유니트(control unit: CU)의 통제 하에 동기적으로 동작하며, 기본 구성도는 아래와 같음.




• 배열 프로세서의 구조

# 배열 프로세서의 구조 (2)



- 각 PE는 프로세서와 기억장치로 구성되는데, 이러한 시스템에서 PE의 기능은 간단한 연산만 수행하는 것이므로 프로세서는 ALU와 레지스터들로만 구성됨.
  제어 유니트는 시스템 프로그램과 사용자 프로그램을 저장하기 위한 기억장치를 가지고 있는데, 프로그램 코드들은 외부의 호스트 컴퓨터(host computer)로부터 전송됨.
- 제어 유니트의 기능은 명령어들을 해석하고, 그것이 실행될 PE들을 결정하는 것임.
  그러나 스칼라 데이터(scalar data)의 계산이나 시스템 제어 명령어들은 CU가 직접 실행함.
  벡터 데이터(vector data)들은 명령어가 실행되기 전에 데이터 버스를 통하여 외부로부터 각 PE의 기억장치로 적재됨.
  이때 동시에 처리될 데이터의 수에 따라 필요한 수만큼의 PE들만 동작을 수행함.
  PE의 선택은 제어 유니트가 각 PE에 대응되는 마스크 비트(mask bit)를 1 또는 0으로 세트 함으로써 이루어짐.
  PE들 간의 데이터 교환은 상호연결망을 통해 이루어지는데, 이 동작도 제어 유니트가 관장함.

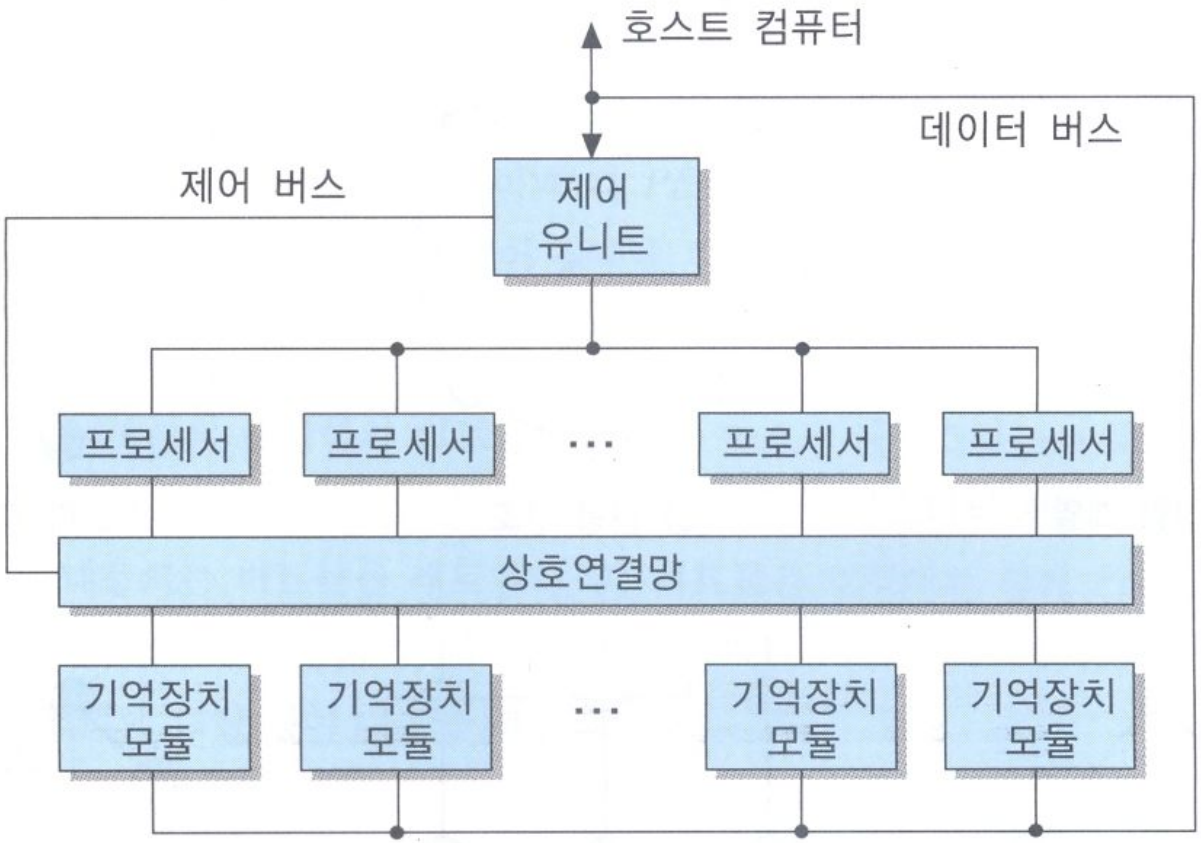
# 배열 프로세서의 구조 (3)



- 대부분의 배열 프로세서는 독립적인 시스템으로서 동작하지는 못하고,
  호스트 컴퓨터로부터 대규모 계산 작업을 의뢰 받아서 고속으로 처리하고,
  그 결과를 호스트 컴퓨터로 되돌려 보내는 계산전용 컴퓨터로 사용됨.
- 배열 프로세서는 아래 그림과 같이 기억장치 모듈들이 어느 한 프로세서에 속해 있지 않고
  모든 프로세서들에 의해 공유하는 구조를 가질 수 있음.
  여기서 프로세서의 수나 기억장치 모듈의 수가 반드시 동일할 필요는 없음.
  프로세서들은 상호연결망을 통해 원하는 데이터가 저장된 기억장치 모듈을
  액세스할 수 있으며,
  외부로부터 시스템으로 들어오거나 외부로 보내지는 모든 데이터 전송은
  데이터 버스를 통해 기억장치와 직접 이루어짐.

# 배열 프로세서의 구조 (4)






• 배열 프로세서의 다른 구조

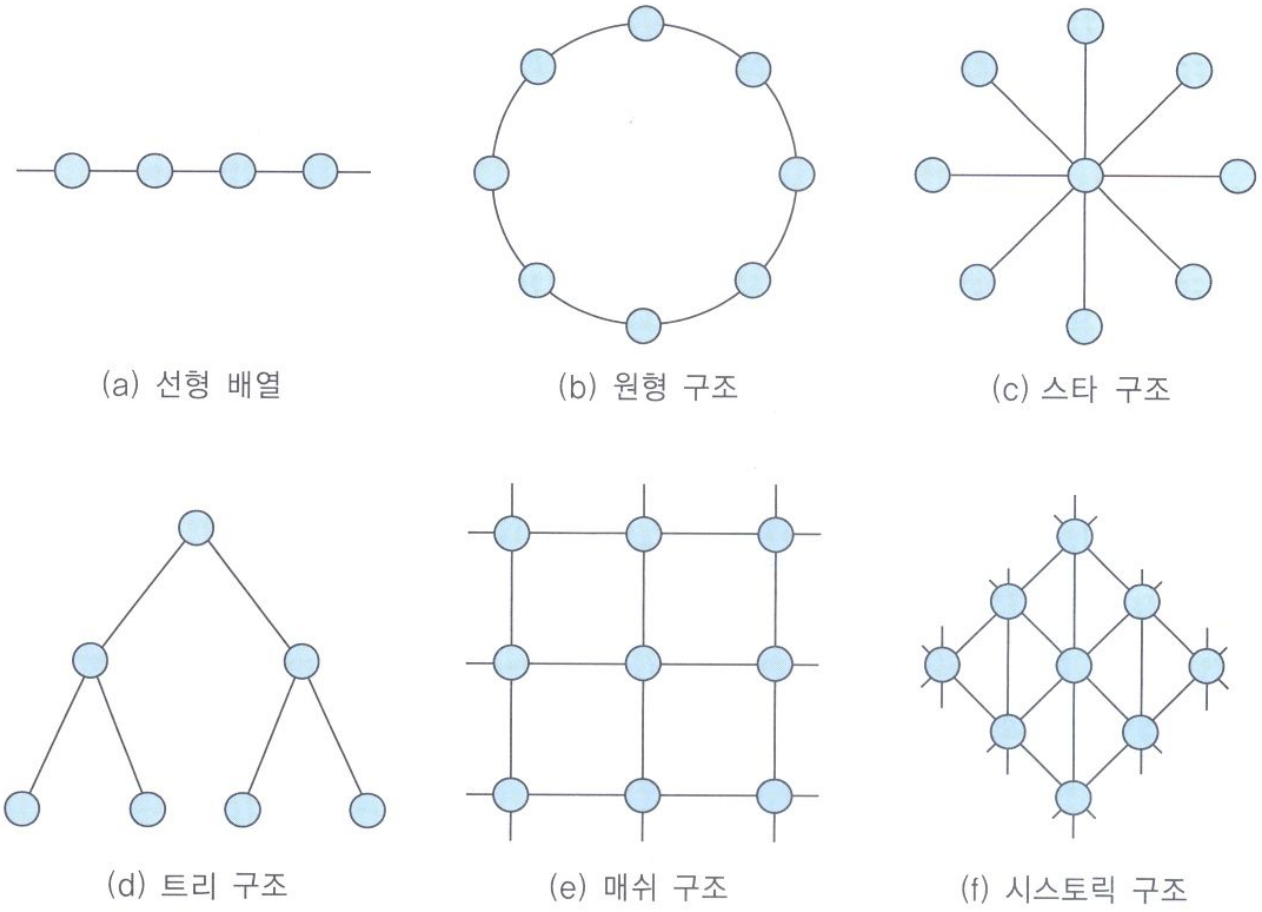
# 배열 프로세서의 구조 (5)



- 배열 프로세서를 위한 상호연결망(interconnection network)에는 연결 구조가 고정되는 정적 네트워크(static network)와 스위치를 이용하여 연결 구조를 변경시킬 수 있는 동적 네트워크(dynamic network)가 있음.
- 정적 네트워크의 토폴로지(topology)는 연결 구조의 '차원(dimension)에 따라 분류할 수 있는데, 1차원 토폴로지인 선형 배열은 파이프라인 구조에 해당함.
  2차원 토폴로지에는 원형(ring), 스타(star), 트리((tree), 매쉬(mesh), 시스토릭 배열(systolic array) 등이 있음.
  3차원 토폴로지의 예로는 완전 연결 구조, 코달 원형 구조, 큐브 구조 등이 있음.

# 배열 프로세서의 구조 (6)

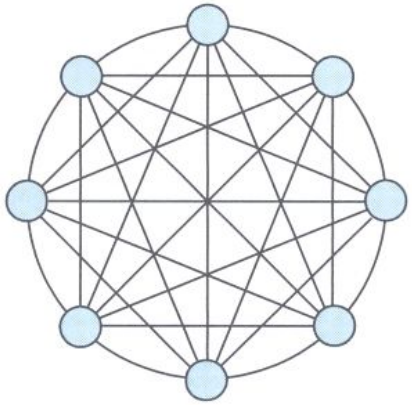




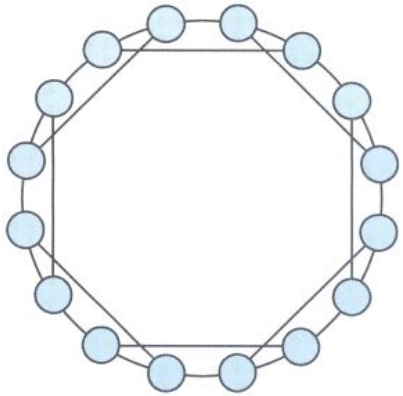

• 정적 상호연결 네트워크들의 예

# 배열 프로세서의 구조 (7)

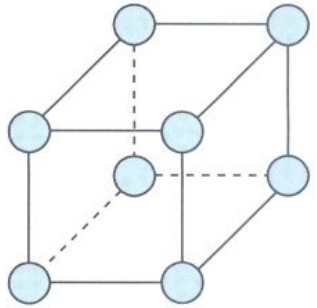




(g) 완전 연결 구조



(h) 코달 원형 구조



(i) 3-차원 큐브

• 정적 상호연결 네트워크들의 예

# 배열 프로세서의 구조 (8)

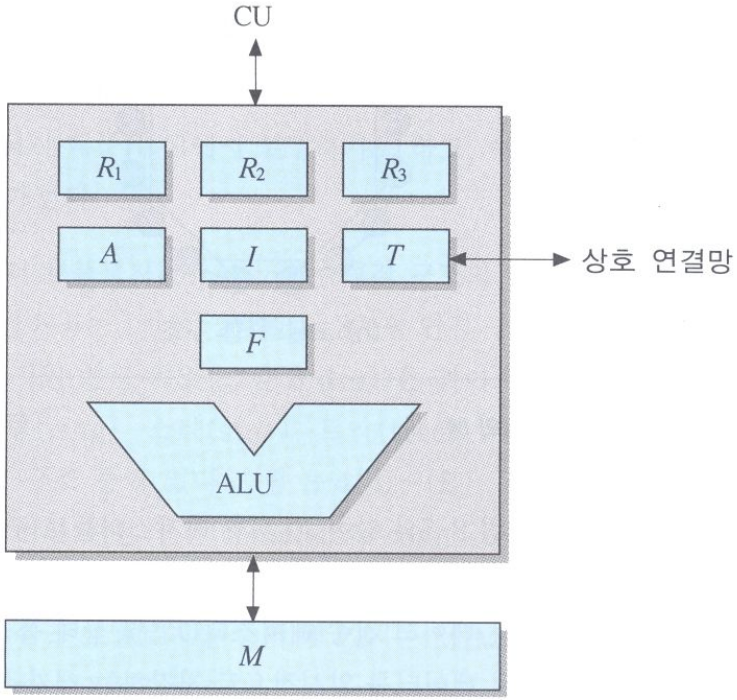


- 배열 프로세서의 각 PE는 ALU와 레지스터들로 구성된 간단한 구조를 가지고 있음.
  각 PE에는 내부 레지스터로서 데이터 레지스터들(R1, R2, R3), 주소 레지스터(A),
  인덱스 레지스터(I), 데이터 전송 레지스터(T), 상태 플래그 레지스터(F) 등이 있음.
  레지스터 T는 전송될 데이터를 일시적으로 저장하는 레지스터이므로
  데이터 전송 통로인 상호연결망과 연결됨.
  A와 I 레지스터는 기억장치 액세스를 위한 주소를 저장하는 데 사용됨.
- 각 명령어 사이클 동안에 데이터 처리에 참여하는 PE들은 제어 유니트로부터 보내져 온
  명령어들을 실행하는데, 이때 각 PE의 활성화 여부는 상태 플래그 F에 의해 지정됨.
  즉 F=1이면 PE는 명령어 실행에 참여하고, F=0이면 그 명령어 사이클 동안 PE가
  대기 상태에 들어감.

# 배열 프로세서의 구조 (9)



- 제어 유니트는 PE들의 수만큼의 비트들로 구성된 마스크 레지스터(mask register)를 가지고 있는데, PE들로 제어 신호를 전송하기 전에 그 레지스터의 비트들 중에서 활성화 될 PE들에 대응되는 비트들을 세트 함. 각 비트들은 명령어와 함께 PE들로 전송되어서 F의 상태를 결정해주게 되며, 이 비트의 값에 따라 각 PE들은 그 명령어 사이클에서의 데이터 처리에 대한 참여 여부를 결정하게 됨.




• 배열 프로세서의 PE의 내부 구조

# 3 교시

# 다중프로세서시스템 구조 (1)



(1) 공유-기억장치 시스템 구조

• 이 시스템 구조는 밀결합 구조로서, 주기억장치가 어느 한 프로세서에 속해 있지 않고 모든 프로세서들에 의해 공유됨.

각 프로세서는 특수 프로그램(하드웨어 초기화와 진단 프로그램)을 저장하고 있는 적은 용량의 지역 기억장치를 별도로 가질 수는 있으나, 운영체제와 사용자 프로그램 및 데이터들은 모두 공유 기억장치에 저장됨.

이 구조의 장점은 아래와 같음.

* 프로세서들이 공통으로 사용하는 데이터들이 공유 기억장치에 저장되므로 별도의 프로세서 간 데이터 교환 메커니즘이 필요하지 않음.
* 프로그램 실행시간 동안에 각 프로세서들이 처리할 작업들을 동적으로 균등하게 할당할 수 있으므로 프로세서 이용률을 극대화할 수 있어서 시스템 효율을 높일 수 있음.

# 다중프로세서시스템 구조 (2)



- 그러나 이 구조는 다음과 같은 단점들도 가지고 있음.
  * 프로세서들과 기억장치들 간의 통로(버스 또는 상호연결망) 상에 통신량이 많아지기 때문에 경합으로 인한 지연 시간이 길어질 수 있음.
  * 두 개 이상의 프로세서들이 공유자원(기억장치 모듈 또는 입출력장치)을 동시에 사용하려는 경우에 한 개 이외의 프로세서들은 기다려야 함.

  결과적으로 이 구조를 가진 시스템에서는 프로세서 수가 증가해도 성능 향상이 선형적으로 이루어지지 못하게 됨.

  이러한 단점들을 보완하기 위해 고속 상호연결망과 캐쉬 기억장치의 사용 등이 있음.
- 상호연결 구조에는 아래와 같은 것들이 있음.
  * 버스(Bus)
  * 크로스바 스위치(Crossbar Switch)
  * 다단계 상호연결망(Multistage Interconnection Network)

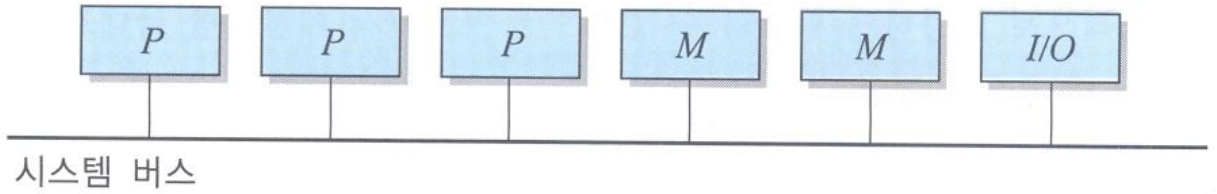
# 다중프로세서시스템 구조 (3)



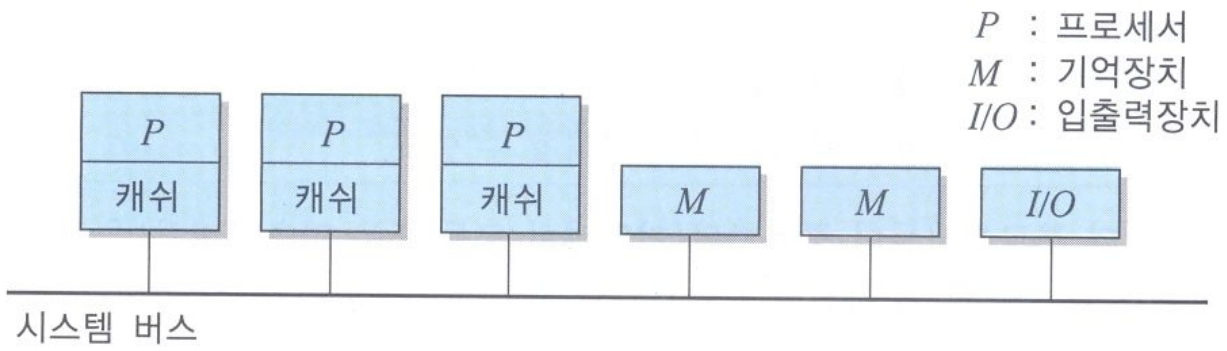
- 버스 구조는 가장 간단한 연결 방식으로서 모든 시스템 요소들이 한 개의 시스템 버스에 접속됨.
  이 구조는 하드웨어가 매우 간단하다는 장점이 있으나,
  모든 요소들간의 통신이 하나의 버스를 통해 이루어지므로
  버스 경합이 높아져서 지연 시간이 길어지는 것이 가장 큰 단점임.
  이를 보완하기 위해 버스의 전송 속도를 높이거나, 그림과 같이 각 프로세서가 캐쉬를 가지도록 하는 방법이 사용되고 있음.
  그러나 시스템 요소들을 최대 30개 정도까지 연결할 수 있을 뿐이고,
  공유-버스 구조는 프로세서의 수가 20개 정도인 중형급 컴퓨터시스템에서 주로 채택되고 있음.

# 다중프로세서시스템 구조 (4)






(a) 캐쉬가 없는 시스템




(b) 캐쉬가 있는 시스템

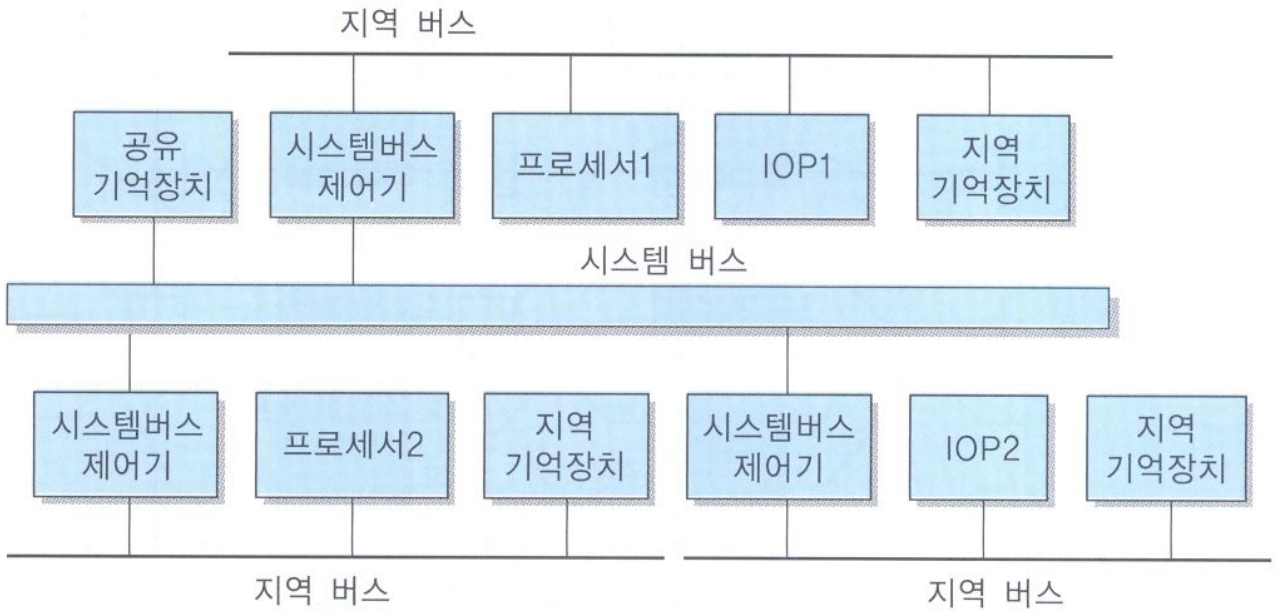
• 공유-버스 다중프로세서 시스템의 구조

# 다중프로세서시스템 구조 (5)



• 버스 경합을 줄이기 위해 버스의 수를 증가시킨 다중-버스(multiple-bus) 구조도 사용될 수 있음.

아래 그림은 각 프로세서가 자신의 지역 기억장치 및 입출력 프로세서와 연결되는 지역 버스(local bus)를 가지고 있고, 공유 기억장치를 사용할 때만 시스템 버스를 사용하는 계층버스 구조를 보여주고 있음.

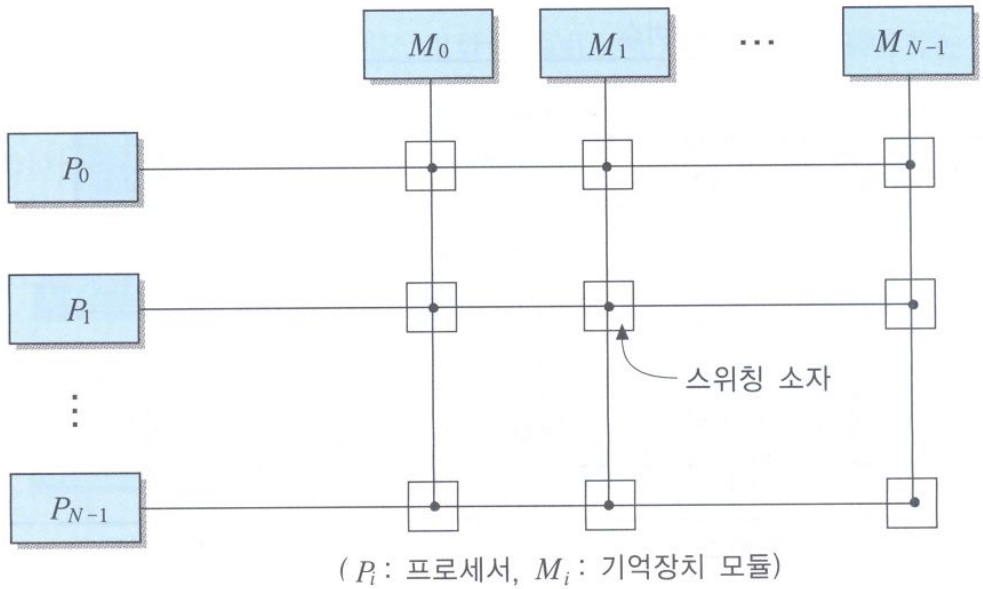



• 계층버스 구조를 가진 다중프로세서 시스템의 구조

# 다중프로세서시스템 구조 (6)



- 프로세서의 수가 많은 시스템에서는 버스 병목 현상을 줄이기 위해 높은 연결성을 제공하는 크로스바 스위치(crossbar switch) 구조를 사용함.
  이 시스템 구조에서는 각 프로세서가 서로 다른 기억장치 모듈을 액세스 하는 경우에는 최대 N개의 기억장치 액세스들이 동시에 수행될 수 있음.
  그러나 만약 두 개 이상의 프로세서들이 동일한 기억장치 모듈을 동시에 액세스 하고자 할 때는 충돌이 발생하게 되며, 이런 때는 중재를 받아서 순서대로 액세스 해야 함.




- 크로스바 스위치를 이용한 시스템 구조

# 다중프로세서시스템 구조 (7)



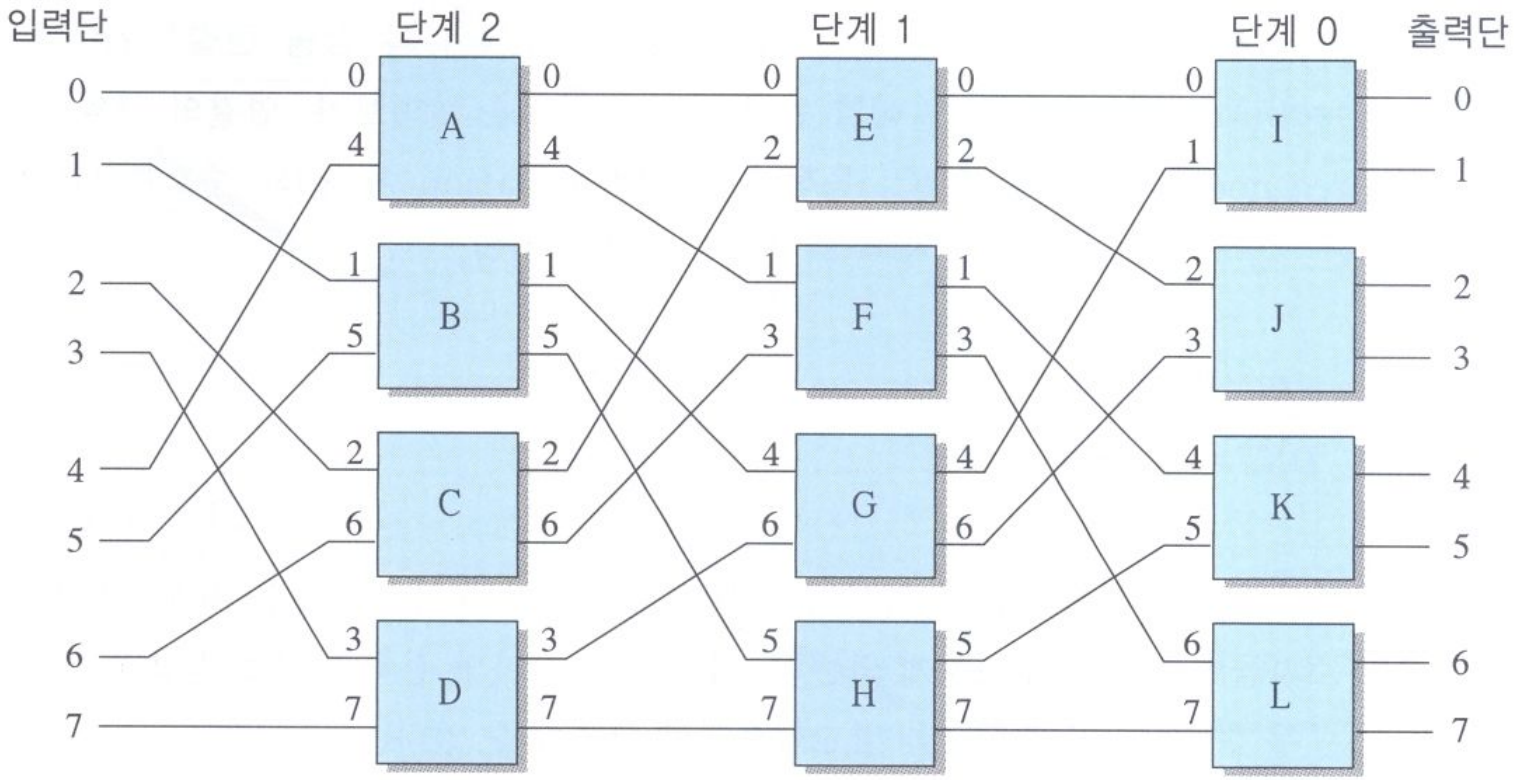
- 크로스바 스위치의 기본 개념을 이용하면서도 하드웨어 복잡성을 줄여주는 다단계 상호연결망(Multistage Interconnection Network: MIN)도 여러 가지가 제안되고 있는데 그 한 예로서 오메가 네트워크(Omega network)가 있음.
- 아래 그림은 8개의 근원지 노드(예: 프로세서)들과 8개의 목적지 노드(예: 기억장치 모듈)들이 오메가 네트워크에 의해 연결되어 있는 구조를 보여주고 있음.
- MIN 구조에서 입력단과 출력단이 각각 N개인 경우에 필요한 단계(stage)의 수(s)와 각 단계의 스위칭 소자(switching element)들의 수(m)를 구하는 일반식은 아래와 같음.

$s = \log_2^N$

$m = N/2$

# 다중프로세서시스템 구조 (8)



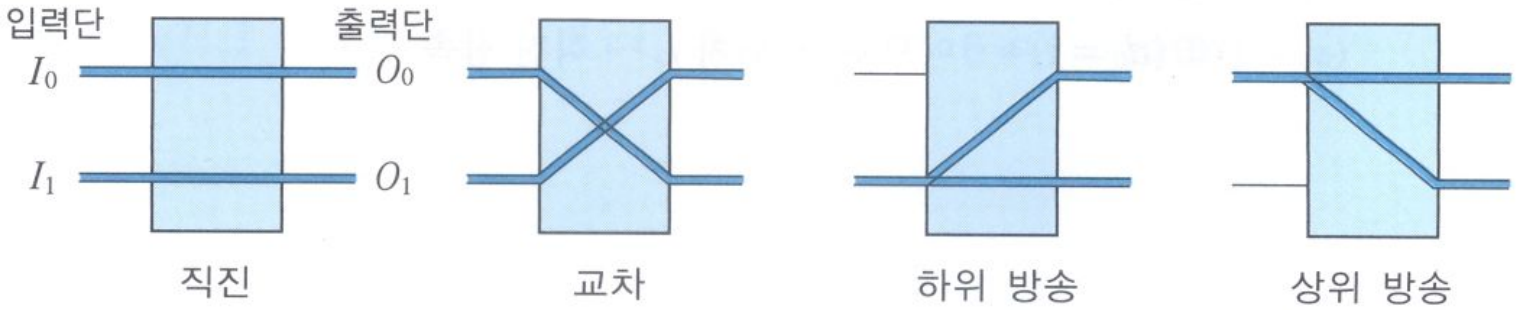



• 오메가 네트워크의 구조

# 다중프로세서시스템 구조 (9)



- 각 스위칭 소자들은 두 개의 입력단과 두 개의 출력단을 가지는 논리 회로로 구성되며, 각 입력과 출력을 연결해주는 접속 방식(connection mode)에는 네 가지가 있음.
  - * 직진(straight) : 같은 위치의 입출력 단자들이 서로 접속되는 방식
  - * 교차(swap) : 서로 다른 위치의 입출력 단자들이 접속되는 방식
  - * 하위 방송(lower broadcast) : 하단의 입력 단자가 모든 출력 단자들로 접속되는 방식.
  - * 상위 방송(upper broadcast) : 상단의 입력 단자가 모든 출력 단자들로 접속되는 방식.




- 스위칭 소자의 접속 방식들

# 다중프로세서시스템 구조 (10)



- 근원지 노드와 목적지 노드 사이의 접속 경로는 여러 개의 단계들을 거치면서 해당 스위칭 소자들에 의해 이루어지게 되는데, 각 스위칭 소자의 접속 방식을 결정해주는 제어 신호는 두 주소의 2진 비트들을 비교함으로써 생성될 수 있음.

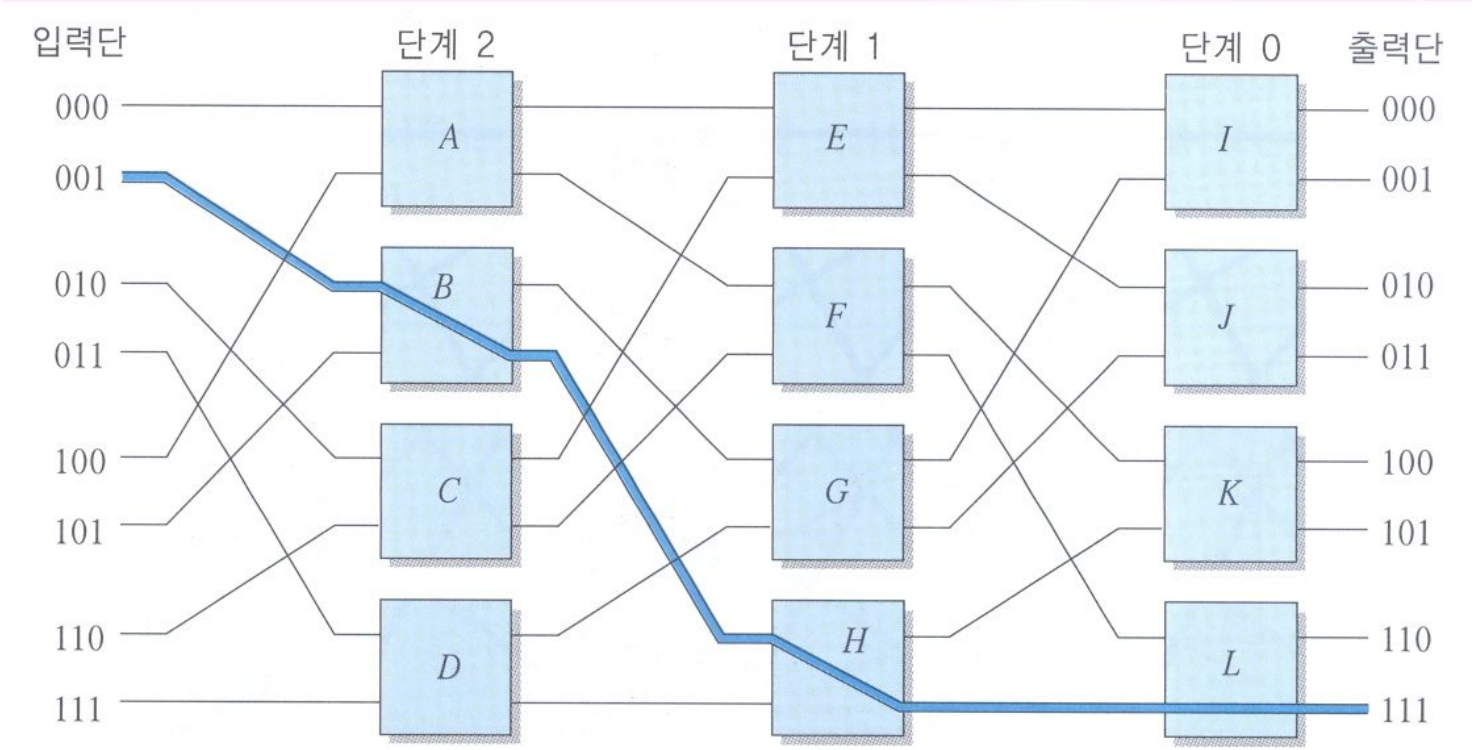
  * 만약 입력단의 주소와 출력단의 주소의 i번째 비트가 서로 다르면, i번째 단계의 스위치는 교차 접속됨.

  * 만약 i번째 비트가 서로 같으면, i번째 단계의 스위치는 직진 접속됨.

  예를 들어서 입력단이 001이고 출력단이 111인 경우에 동일한 위치의 비트를 서로 비교해 보면 첫 번째 비트와 두 번째 비트가 서로 다르고 세 번째 비트가 서로 같음. 따라서 첫 번째 단계와 두 번째 단계의 스위치는 교차 접속하고, 세 번째 단계는 직진 접속하므로 아래 그림과 같이 경로가 설정되는 것임.

# 다중프로세서시스템 구조 (11)






• 노드 1과 노드 7사이에 설정된 경로

# 다중프로세서시스템 구조 (12)



(2) 분산-기억장치 시스템 구조

- 소결합 구조(loosely-coupled structure)에서는 프로세서들이 기억장치를 공유하지 않고, 각 프로세서들이 자신의 기억장치를 별도로 가지고 있음.
  이와 같은 구조를 가진 시스템을 분산-기억장치 컴퓨터시스템이라고 하며,
  다중-컴퓨터 시스템(multiple-computer system)이라고도 부름.
  프로세서 간 통신량은 공유-기억장치 시스템에 비해 크게 줄어들지만, 프로세서 간 통신이 메시지(message) 형태로 이루어지기 때문에, 통신 프로토콜에 의한 지연 시간이 증가함.
  통신 채널은 시스템 내의 컴퓨터 모듈 간에 백 플레인(backplane)을 통해 구성되기도 하고, 이더넷(Ethernet)과 같은 통신 네트워크를 이용하여 독립적인 컴퓨터시스템들을 접속하여 분산 시스템(distributed system)으로 구성하는 경우도 있음.
- 이러한 시스템들에서 사용되는 상호연결망 구조에는 선형 배열 구조, 원형 구조, 트리 구조, 매쉬 구조 등이 있음.

# 다중프로세서시스템 구조 (13)



(가) 선형 배열 구조

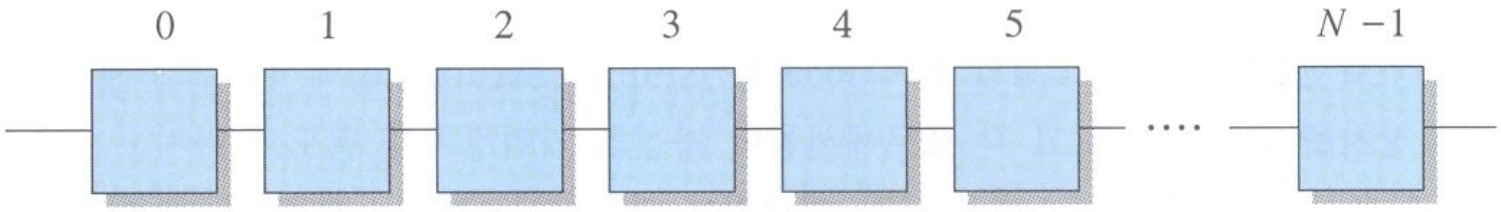
• N개의 노드들이 N-1개의 링크들에 의해 차례대로 연결된 형태를 선형 배열(linear array) 구조라고 부름.

이 구조의 네트워크 지름은 N-1이므로 다른 구조들에 비해 평균 통신 시간이 가장 길음.

여기서 네트워크 지름이란 네트워크 내에서 가장 멀리 떨어져 있는 노드들 간의 거리(즉, 링크의 수)를 말함.

선형 배열 구조는 연결 토폴로지가 간단하며, 각 링크에서 동시에 전송 동작이 일어날 수 있으므로 버스 구조보다 동시성이 더 높다는 장점이 있음.

그러나 통신에 걸리는 시간이 노드들 간의 거리에 따라 서로 다르며, 노드의 수가 많아지면(N이 커지면) 통신 시간이 매우 길어진다는 단점이 있음.




• 선형 배열 구조

# 다중프로세서시스템 구조 (14)



(나) 원형 구조

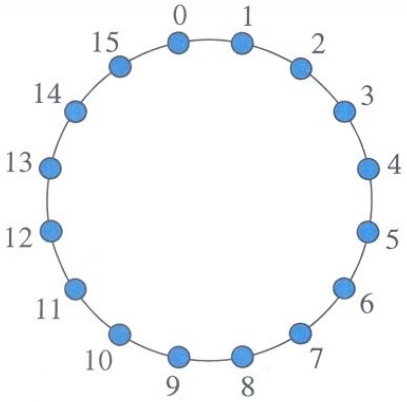
- 선형 배열 구조에서 0번 노드와 N-1번 노드를 서로 연결해주는 링크가 하나 추가되면 원형(ring) 구조가 됨.

  이 구조의 네트워크 지름은 각 링크가 양방향성이면 $\lfloor N/2 \rfloor$가 되고, 단방향성이면 N-1이 됨.

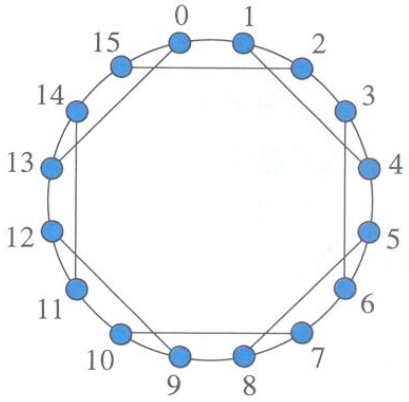
  원형 구조의 노드들 사이에 링크를 한 개 혹은 두 개씩 추가시킴으로써 새로운 구조를 구성할 수 있는데 이러한 변형된 구조를 코달 원형(chordal ring) 구조라고 부르며, 링크의 수가 증가될수록 네트워크 지름은 감소됨.

# 다중프로세서시스템 구조 (15)

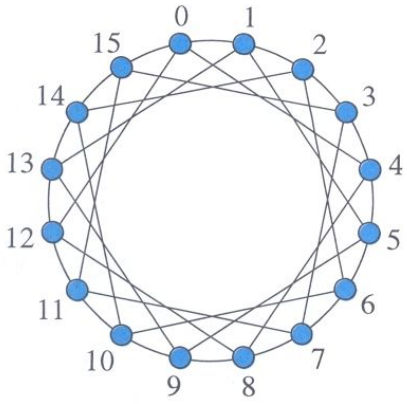





(a) 원형 구조($d$=2)




(b) 코달 원형($d$=3)




(c) 코달 원형($d$=4)

서울사이버대학교 SEOUL CYBER UNIVERSITY

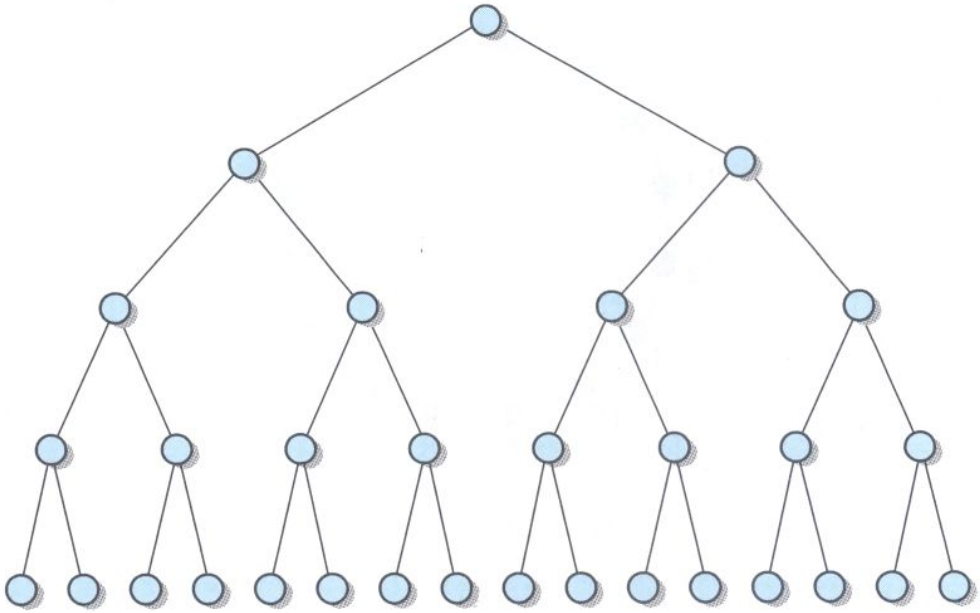
• 원형 구조와 코달 원형 구조

# 다중프로세서시스템 구조 (16)



(다) 트리 구조

- 아래 그림은 31개의 노드들이 5 층의 2진 트리(binary tree) 구조로 연결된 모습을 보여주고 있음. 이러한 트리 구조가 완전히 안정된 모습을 가지기 위해서는 층(level)의 수를 k라고 할 때 $N = (2^k - 1)$개의 노드들이 필요함.
  이 구조의 네트워크 지름은 2(k-1)인데, 시스템 요소들의 수가 증가함에 따라 성능이 선형적으로 향상되는 구조로 알려져 있으나 네트워크 지름은 비교적 큰 편임.

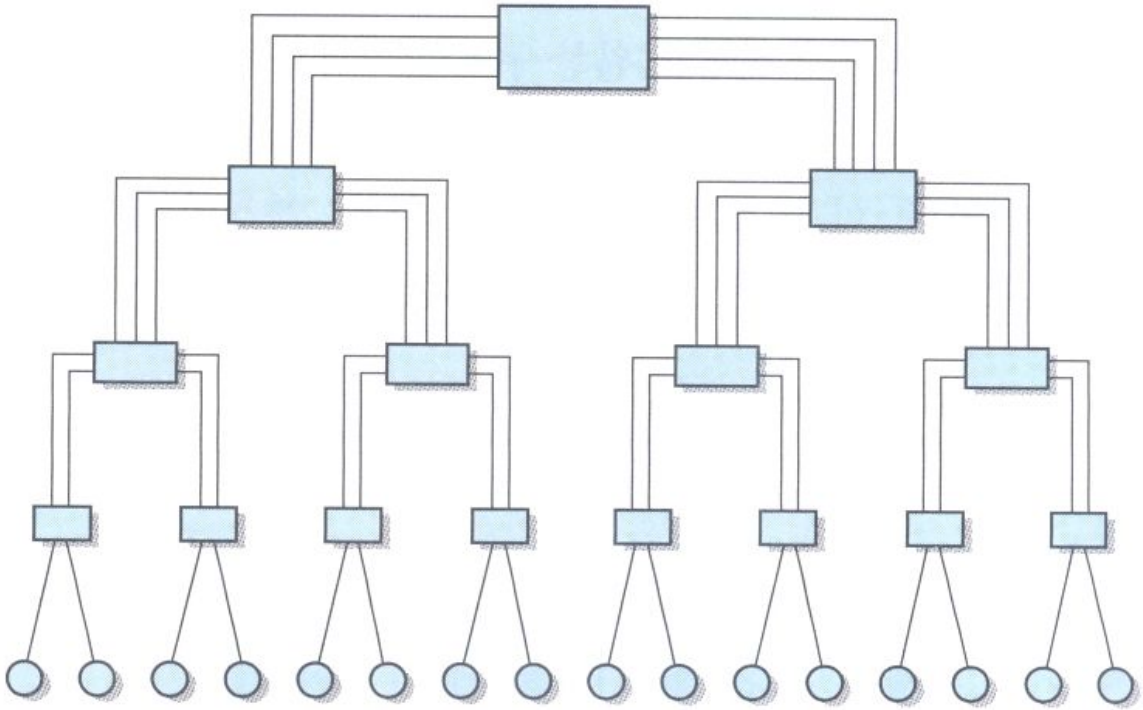


• 2진 트리 구조

# 다중프로세서시스템 구조 (17)



• 2진 트리 구조는 패트 트리(fat tree) 구조로 변형될 수 있는데 이 구조는 상위 층으로 올라갈수록 노드 간의 통신 채널 수가 증가하는 특징을 가지고 있음.



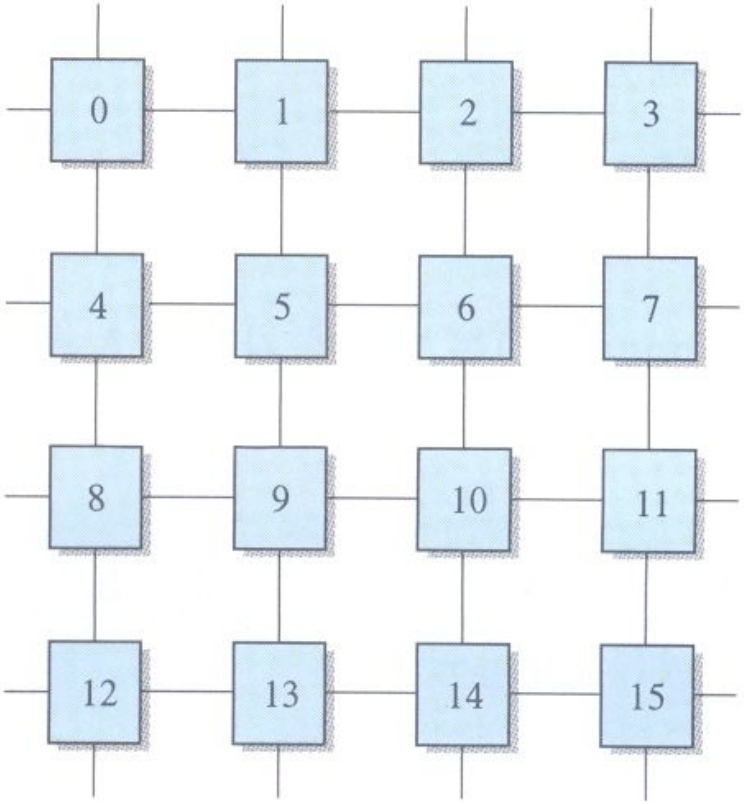
• 2진 패트 트리 구조

# 다중프로세서시스템 구조 (18)



(라) 매쉬 구조

• 노드들을 2차원 배열로 연결하여 각 노드가 4개의 주변 노드들과 직접 연결되는 구조를 매쉬 네트워크(mesh network)라고 부름.




• 매쉬 네트워크의 기본 구조

# 셀프 테스트



• Flynn의 분류에서 복수의 명령어 스트림과 복수의 데이터 스트림을 가지는 컴퓨터 시스템 구조는 무엇인가?

* MIMD

해설) MIMD는 복수의 명령어 스트림과 복수의 데어터 스트림을 갖는 컴퓨터 시스템을 말함.

• 시스템 내의 모든 프로세싱 요소(processing element: PE)들이 하나의 제어 유니트(control unit: CU)의 통제 하에 동기적으로 동작하는 프로세서를 무엇이라고 하는가?

* 배열 프로세서

해설) 배열프로세서는 SIMD, 즉 하나의 제어 유니트에 여러 개의 프로세싱 요소들을 가지는 컴퓨터 시스템을 말함.

• 다중프로세서시스템 구조에서 N개의 노드들이 N-1개의 링크들에 의해 차례대로 연결된 형태를 갖는 구조는 무엇인가?

* 선형 배열 구조

해설) 선형배열구조에서는 N개의 노드들이 N-1개의 링크들로 순서적으로 연결된 구조임.

# 요점 정리



- 병렬처리(parallel processing)는 다수의 프로세서들이 여러 개의 프로그램들 혹은 한 프로그램의 분할된 부분들을 분담하여 동시에 처리하는 기술을 말함.
- 명령어와 데이터 스트림을 처리하기 위한 하드웨어 구조에 따른 Flynn의 분류에는 SISD, SIMD, MISD, MIMD 등이 있음.
- 배열 프로세서는 시스템 내의 모든 프로세싱 요소(processing element: PE)들이 하나의 제어 유니트(control unit: CU)의 통제 하에 동기적으로 동작함.
- 공유-기억장치 시스템 구조는 밀결합 구조로서, 주기억장치가 어느 한 프로세서에 속해 있지 않고 모든 프로세서들에 의해 공유됨.
- 분산-기억장치 시스템 구조는 소결합 구조(loosely-coupled structure)로서 프로세서들이 기억장치를 공유하지 않고, 각 프로세서들이 자신의 기억장치를 별도로 가지고 있음.
- 공유-기억장치 시스템 구조에서 상호연결 방법에는 버스, 크로스바 스위치, 다단계 상호연결망 등이 있음.
- 분산-기억장치 시스템의 상호연결 방법에는 선형 배열, 원형, 트리, 매쉬 구조 등이 있음.