2012年 後学期

計算機アーキテクチャ 第二 (O)

マルチコアプロセッサ

1

マルチコア（2個～数10個）からメニーコアへ

- デスクトップPC等に搭載される高性能・汎用プロセッサのアーキテクチャは，今後，数百個のコアを搭載するメニーコアプロセッサの時代へ

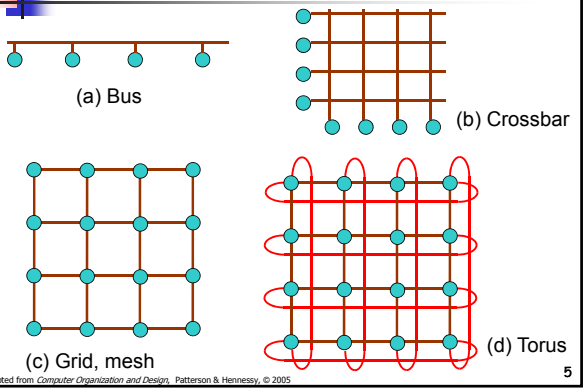
メニーコアアーキテクチャにおける重要な選択肢

- コアのアーキテクチャ
  - スーパースカラ，アウトオブオーダ実行？
  - 2-way のインオーダ・スーパースカラ程度の複雑さ
- ネットワークアーキテクチャ
  - どのようにコアやメモリを接続するのか？
- メモリアーキテクチャ
  - 共有メモリ（すべてのコアが同じメモリ空間），
  - 分散メモリ（異なるメモリ空間を持つ）？
  - キャッシュ，一貫性管理

Many-core processor
（メニーコアプロセッサ）

ネットワークトポロジー

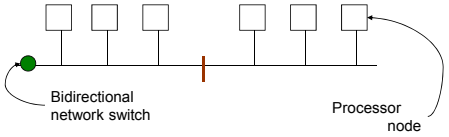
Interconnection Network



Interconnection Network **Performance Metrics**

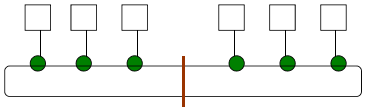
- Network cost
  - number of switches
  - number of links on a switch to connect to the network (plus one link to connect to the processor)
  - width in bits per link, length of link
- Network bandwidth (NB)
  - represents the **best** case
  - bandwidth of each link * number of links
- Bisection bandwidth (BB)バイセクションバンド幅
  - represents the **worst** case
  - divide the machine in two parts, each with half the nodes and sum the bandwidth of the links that cross the dividing line

Bus Network



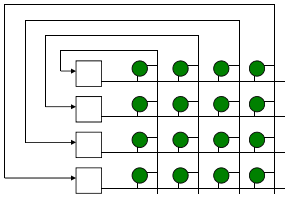
- N processors, 1 switch (●), 1 link (the bus)
- Only 1 simultaneous transfer at a time
  - NB (best case) = link (bus) bandwidth * 1
  - BB (worst case) = link (bus) bandwidth * 1

Ring Network



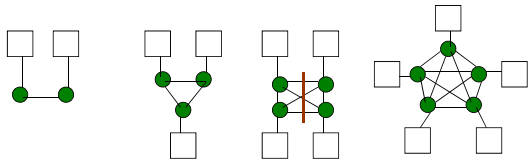
- N processors, N switches, 2 links/switch, N links
- N simultaneous transfers
  - NB (best case) = link bandwidth * N
  - BB (worst case) = link bandwidth * 2
- If a link is as fast as a bus, the ring is only twice as fast as a bus in the worst case, but is N times faster in the best case

Crossbar (Xbar) Network



- N processors, N^2 switches (unidirectional), 2 links/switch, N^2 links
- N simultaneous transfers
  - NB = link bandwidth * N
  - BB = link bandwidth * N/2

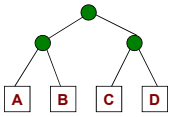
Fully Connected Network



- N processors, N switches, N-1 links/switch, (N*(N-1))/2 links
- N simultaneous transfers
  - NB (best case) = link bandwidth * (N*(N-1))/2
  - BB (worst case) = link bandwidth * $(N/2)^2$

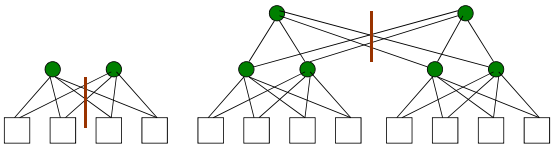
Fat Tree

- Trees are good structures.
  People in CS (Computer Science) use them all the time.
  Suppose we wanted to make a tree network.



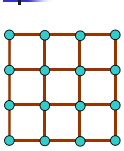
- Any time A wants to send to C, it ties up the upper links, so that B can't send to D.
  - The bisection bandwidth on a tree is horrible - 1 link, at all times
- The solution is to '**thicken**' the upper links.

Fat Tree

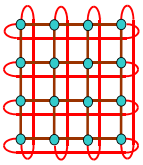


- N processors, log(N-1)*logN switches, 2 up + 4 down = 6 links/switch, N*logN links
- N simultaneous transfers
  - NB = link bandwidth * N log N
  - BB = link bandwidth * 4

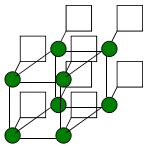
2D and 3D Mesh/Torus Network



Mesh



Torus



- N processors, N switches, 2, 3, 4 (2D torus) or 6 (3D torus) links/switch, 4N/2 links or 6N/2 links
- N simultaneous transfers
  - NB = link bandwidth * 4N or link bandwidth * 6N
  - BB = link bandwidth * 2 $N^{1/2}$ or link bandwidth * 2 $N^{2/3}$

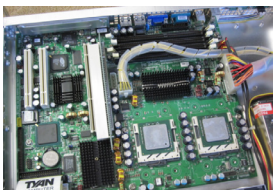
Interconnection Network Comparison

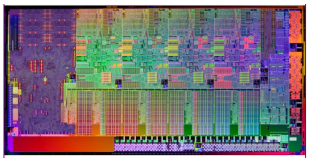
- For a **64** processor system

	Bus	Ring	2D Torus	6-cube	Fully connected
Network bandwidth	1	**64**	**256**	192	**2016**
Bisection bandwidth	1	**2**	**16**	32	**1024**
Total # of switches	1	**64**	**64**	64	**64**
Links per switch		**2+1**	**4+1**	6+7	**63+1**
Total # of links (bidi)	1	**64+64**	**128+64**	192+64	**2016+64**

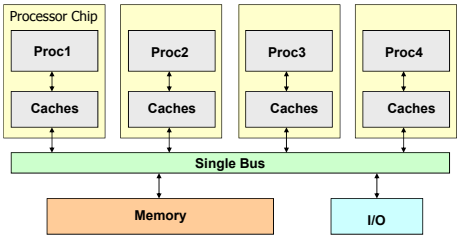
メモリ構成とネットワークアーキテクチャ

マルチプロセッサ と マルチコア



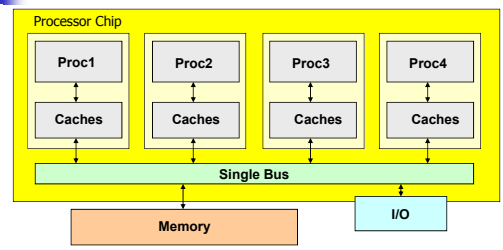


単一バス結合のマルチプロセッサ，共有メモリ



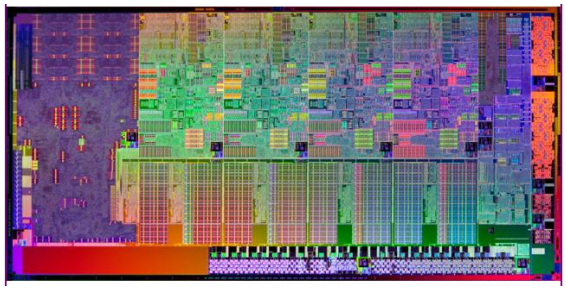
- Caches are used to reduce latency and to lower bus traffic
- Must provide hardware to ensure that caches and memory are consistent (cache coherency)
- Must provide a hardware mechanism to support process synchronization

単一バス結合のマルチコア，共有メモリ



- Caches are used to reduce latency and to lower bus traffic
- Must provide hardware to ensure that caches and memory are consistent (cache coherency)
- Must provide a hardware mechanism to support process synchronization

マルチコアプロセッサの例, Intel Sandy Bridge

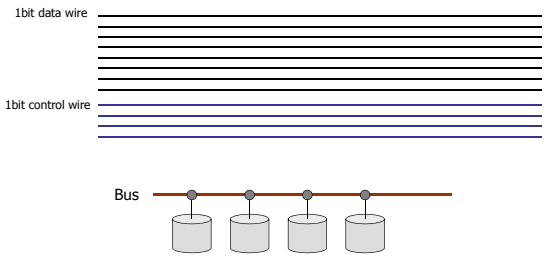


Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

19

Bus

- A **bus**（バス）is a **shared** communication link



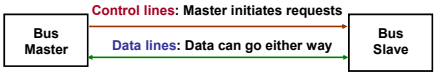
20

Bus, I/O System Interconnect

- A **bus**（バス）is a shared communication link (a single set of wires used to connect multiple subsystems)
  - **Advantages**
    - **Low cost** – a single set of wires is shared in multiple ways
    - **Versatile（多目的）** – new devices can be added easily and can be moved between computer systems that use the same **bus standard**
  - **Disadvantages**
    - Creates a communication bottleneck – **bus bandwidth** limits the maximum **I/O throughput**
- The maximum bus speed is largely limited by
  - The **length** of the bus
  - The **number** of devices on the bus

21

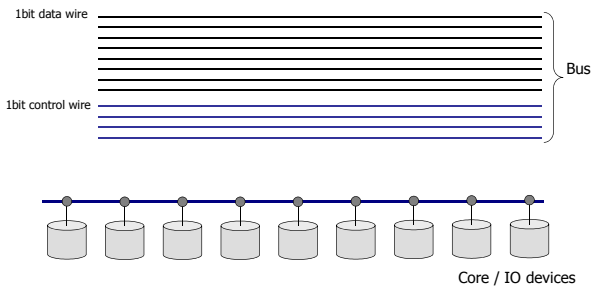
Bus Characteristics



- **Control lines**
  - Signal requests and acknowledgments
  - Indicate what type of information is on the data lines
- **Data lines**
  - Data, addresses, and complex commands
- **Bus transaction** consists of
  - Master issuing the command (and address) – request
  - Slave receiving (or sending) the data – action
  - *Defined by what the transaction does to memory*
    - **Input** – inputs data from the I/O device to the memory
    - **Output** – outputs data from the memory to the I/O device

22

The Need for Bus **Arbitration**（調停）



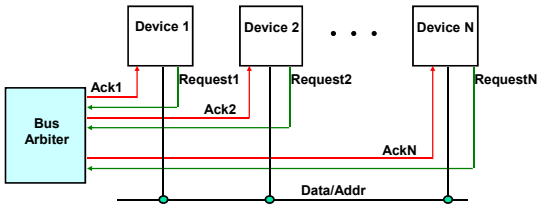
23

The Need for Bus **Arbitration**（調停）

- Multiple core / devices may need to use the bus **at the same time**
- **Bus arbitration schemes** usually try to balance:
  - **Bus priority** – the highest priority device should be serviced first
  - **Fairness** – even the lowest priority device should never be completely locked out from the bus
- **Bus arbitration schemes** can be divided into four classes
  - **Daisy chain arbitration**
  - **Centralized, parallel arbitration**
  - Distributed arbitration by collision detection
    - device uses the bus when its not busy and if a collision happens (because some other device also decides to use the bus) then the device tries again later (Ethernet)
  - Distributed arbitration by self-selection

24

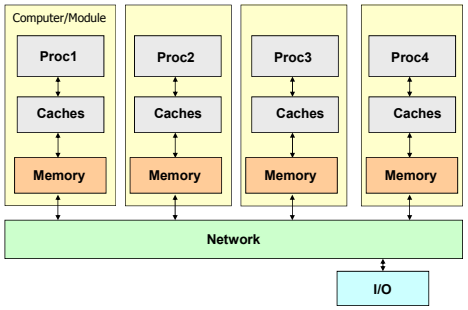
Centralized Parallel Arbitration（集中並列方式）



- **Advantages**: flexible, can assure fairness
- **Disadvantages**: more complicated arbiter hardware
- Used in essentially all processor-memory buses and in high-speed I/O buses

25

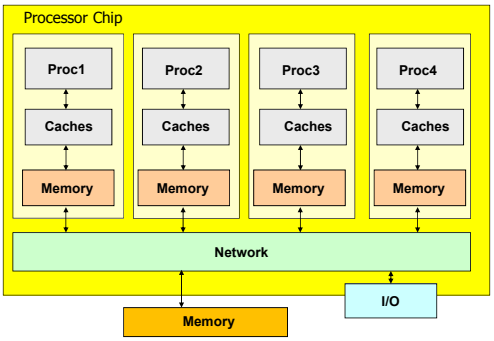
ネットワーク結合の並列計算機/クラスタ，**分散メモリ**



Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

26

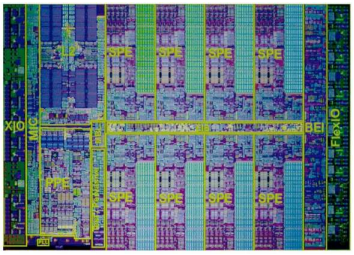
ネットワーク結合のマルチコアプロセッサ，**分散メモリ**



Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

27

Cell Broadband Engine & PS3



28

Cell BE Element Interconnect Bus

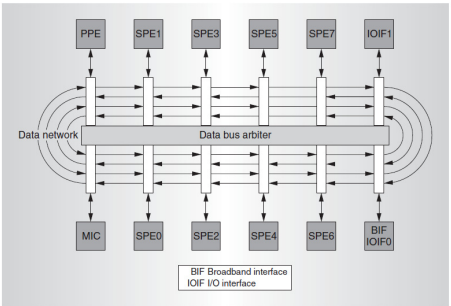


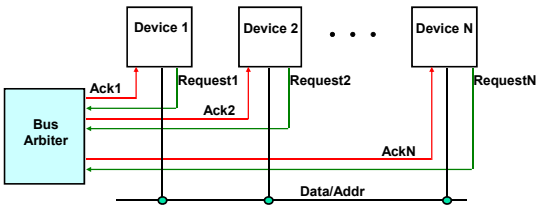
Figure 2. Element interconnect bus (EIB).

IEEE Micro, Cell Multiprocessor Communication Network: Built for Speed

Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

29

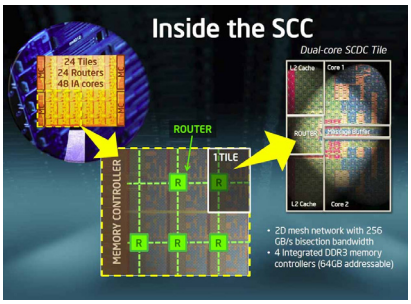
Centralized Parallel Arbitration（集中並列方式）



- **Advantages**: flexible, can assure fairness
- **Disadvantages**: more complicated arbiter hardware
- Used in essentially all processor-memory buses and in high-speed I/O buses

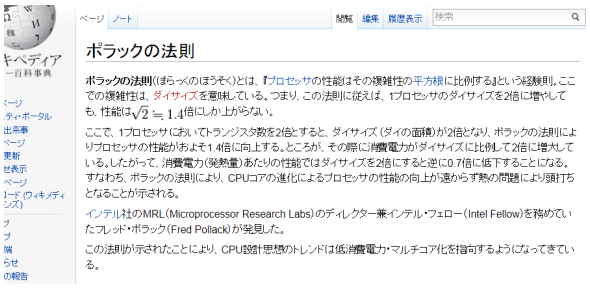
30

メニーコアプロセッサの例, Intel SCC



Intel Single-Chip Cloud Computer **(48 Core)**

ポラックの法則



ポラックの法則

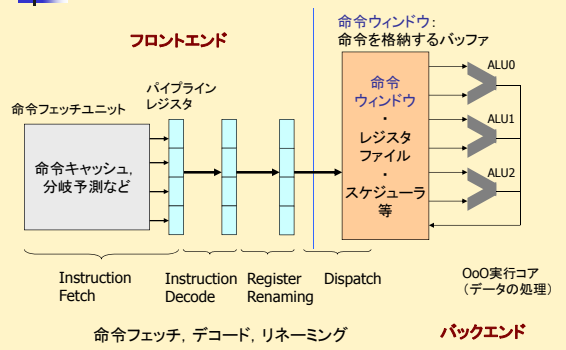
ポラックの法則(ぽらっくのほうそく)とは、『プロセッサの性能はその複雑性の平方根に比例する』という経験則。ここでの複雑性は、ダイサイズを意味している。つまり、この法則に従えば、1プロセッサのダイサイズを2倍に増やしても、性能は$\sqrt{2} \fallingdotseq 1.4$倍にしか上がらない。

ここで、1プロセッサにおいてトランジスタ数を2倍とすると、ダイサイズ(ダイの面積)が2倍となり、ポラックの法則によりプロセッサの性能がおよそ1.4倍に向上する。ところが、その際に消費電力がダイサイズに比例して2倍に増大している。したがって、消費電力(発熱量)あたりの性能ではダイサイズを2倍にすると逆に0.7倍に低下することになる。すなわち、ポラックの法則により、CPUコアの進化によるプロセッサの性能の向上が遅からず熱の問題により頭打ちとなることが示される。

インテル社のMRL(Microprocessor Research Labs)のディレクター兼インテル・フェロー(Intel Fellow)を務めていたフレッド・ポラック(Fred Pollack)が発見した。

この法則が示されたことにより、CPU設計思想のトレンドは低消費電力・マルチコア化を指向するようになってきている。

アウトオブオーダ実行プロセッサの構成



マルチコア(2個～数10個)からメニーコアへ

Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction, MICRO-36

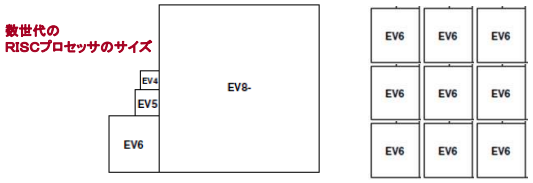


Figure 1. Relative sizes of the cores used in the study

マルチコア(2個～数10個)からメニーコアへ

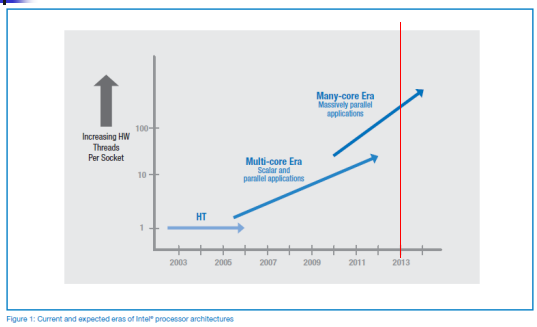
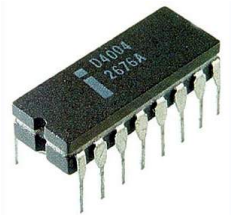


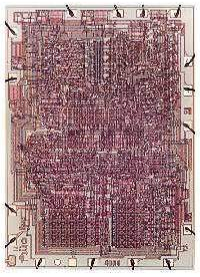
Figure 1: Current and expected eras of Intel® processor architectures

Platform 2015: Intel® Processor and Platform Evolution for the Next Decade

世界初のマイクロプロセッサ (1971)

1971年: 4004 マイクロプロセッサ

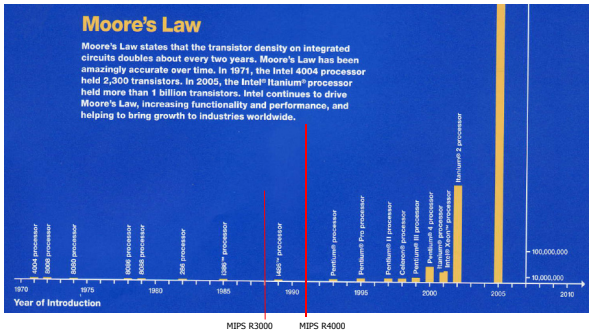




プロセッサ	出荷年	トランジスタ数
4004	1971	2,250

出典: フリー百科事典『ウィキペディア(Wikipedia)』、Intelミュージアム

Moore's Law



The free lunch is over !

IBM Software Group

"The free lunch is over!" [Sutter, 2005]

- With multi-core processors, programs written in sequential model will no longer surf on the wave of this generation of processors
  - Even though they get a little bit faster, they won't enjoy the whole improvements
- To surf in the new wave, programs need to be well-written parallel
  - Remember: Not all problems can be parallelized (regular parallelism)
- "programming languages and systems will increasingly be forced to deal well with concurrency"
  - Java has included support for concurrency since its beginning, but…
  - … Java 5.0 includes as part of the release the java.concurrent api, a tentative to improve the support to write concurrent programs
  - ISO C++ does not have support to write multithread systems
    - Although there are non-standard and non-portable alternatives such as Pthreads and OpenMP
- Fine-grained control constructs (e.g.: loops) are difficult to parallelize
  - Functional languages are naturally suited to concurrency due to its nature:
    - Immutable object instances, higher order functions, and parallelism exposed in the level of procedure calls

アナウンス

- 講義スライド，講義スケジュール
  - www.arch.cs.titech.ac.jp