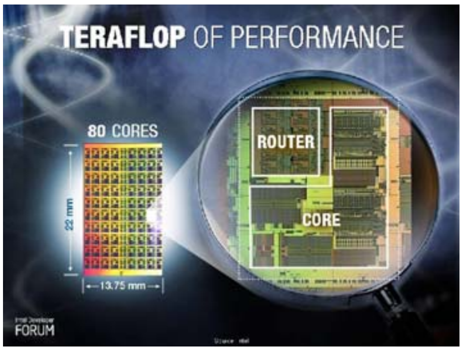
計算機アーキテクチャ特論
(Advanced Computer Architectures)

マルチコアプロセッサ・プログラミング

吉瀬 謙二　計算工学専攻
kise _at_ cs.titech.ac.jp　www.arch.cs.titech.ac.jp
W832 講義室　金曜日　13:20 – 14:50

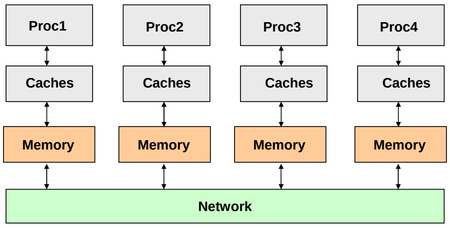
ネットワーク結合のマルチコアプロセッサ

- Intel 80-Core
  - 8 x 10 tiles
  - 2D Mesh



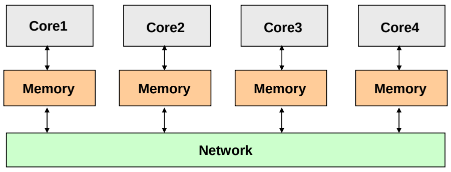
Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

ネットワーク結合のマルチプロセッサ，分散メモリ



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

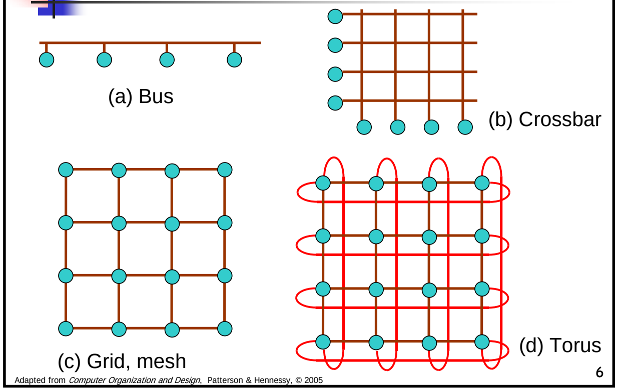
ネットワーク結合のマルチコアプロセッサ



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

ネットワーク

Interconnection Network

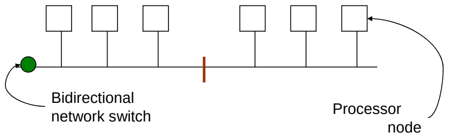


Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

Interconnection Network **Performance Metrics**

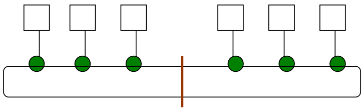
- Network cost
  - number of switches
  - number of links on a switch to connect to the network (plus one link to connect to the processor)
  - width in bits per link, length of link
- Network bandwidth (NB)
  – represents the **best** case
  - bandwidth of each link * number of links
- Bisection bandwidth (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

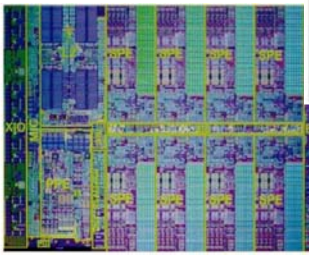
Ring Network



- 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

Cell Broadband Engine & PS3

PLAYSTATION 3試作見本（東京ゲームショウ2005バージョン）



Cell BE Element Interconnect Bus

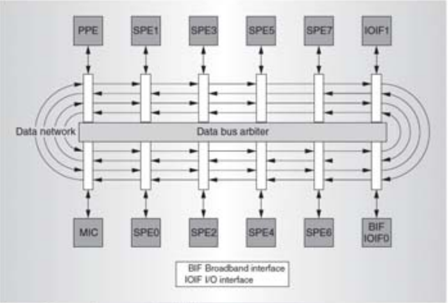
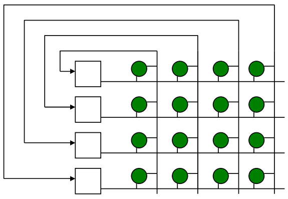


Figure 2. Element interconnect bus (EIB).

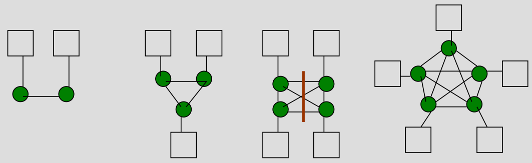
IEEE Micro, Cell Multiprocessor Communication Network: Built for Speed

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

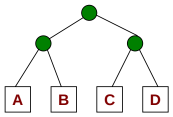
Fully Connected Network



- 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 * $(N/2)^2$

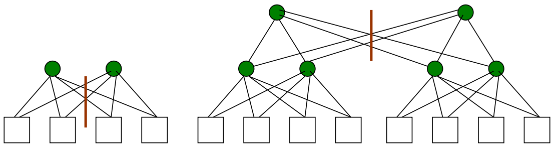
Fat Tree

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



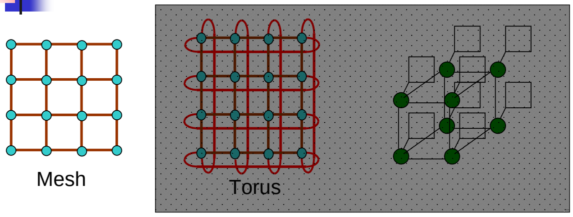
- Any time A wants to send to C, it ties up the upper links, so that B can't send to D.
  - The bisection bandwidth on a tree is horrible - 1 link, at all times
- The solution is to '**thicken**' the upper links.

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

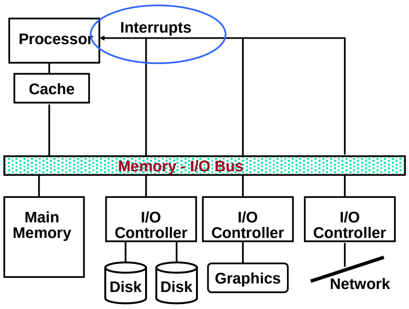
2D and 3D Mesh/Torus Network



- 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 $N^{2/3}$

割り込みとDMA

I/O Systemの利用方法と割り込み



Communication of I/O Devices and Processor

- How the processor directs the I/O devices
  - **Memory-mapped I/O**
    - Portions of the high-order memory address space are assigned to each I/O device
    - Read and writes to those memory addresses are interpreted
      as commands to the I/O devices
    - Load/stores to the I/O address space can only be done by the OS
  - **Special I/O instructions**

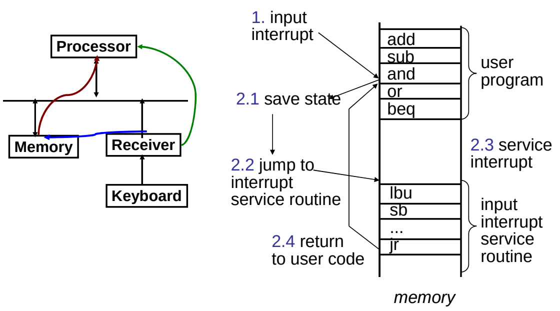
Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

Communication of I/O Devices and Processor

- How the I/O device communicates with the processor
  - **Polling** – the processor periodically checks the status of an I/O device to determine its need for service
    - Processor is totally in control – but does **all** the work
    - Can waste a lot of processor time due to speed differences
  - **Interrupt-driven I/O** – **the I/O device issues an interrupts to the processor to indicate that it needs attention**

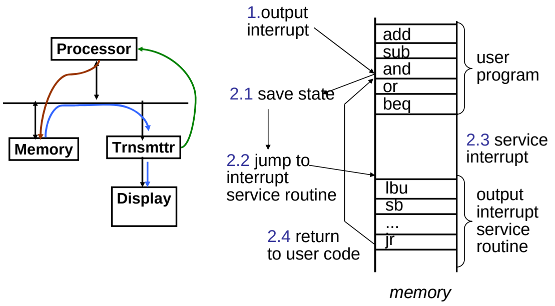
Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

Interrupt-Driven Input



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

Interrupt-Driven Output



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

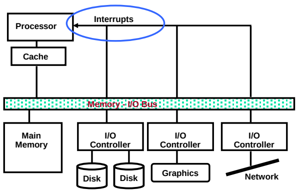
Interrupt-Driven I/O

- An I/O interrupt is **asynchronous**
  - Is not associated with any instruction so doesn't prevent any instruction from completing
    - You can pick your own convenient point to handle the interrupt
- With I/O interrupts
  - Need a way to identify the device generating the interrupt
  - Can have different urgencies (so may need to be prioritized)
- **Advantages** of using interrupts
  - No need to continuously poll for an I/O event; user program progress is only suspended during the actual transfer of I/O data to/from user memory space
- **Disadvantage** – special hardware is needed to
  - Cause an interrupt (I/O device) and detect an interrupt and save the necessary information to resume normal processing after servicing the interrupt (processor)

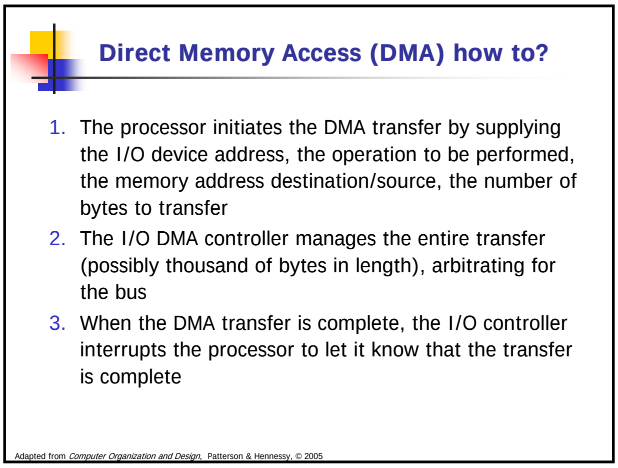
Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

Direct Memory Access (DMA)

- For high-bandwidth devices (like disks) **interrupt-driven I/O** would consume a *lot* of processor cycles
- **DMA** – the I/O controller has the ability to transfer data **directly** to/from the memory without involving the processor
- There may be multiple DMA devices in one system



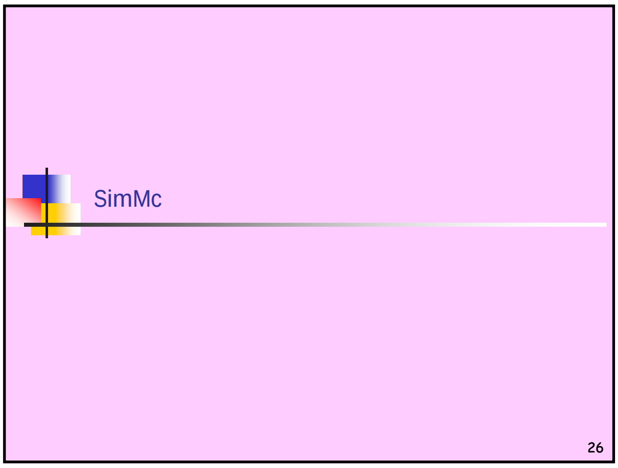
Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005



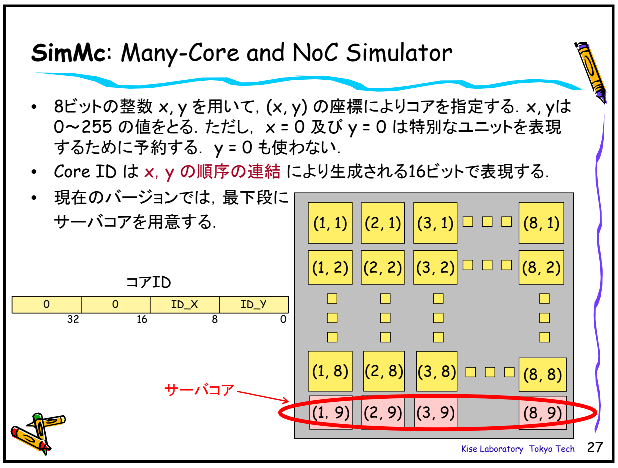
Direct Memory Access (DMA) how to?

1. The processor initiates the DMA transfer by supplying the I/O device address, the operation to be performed, the memory address destination/source, the number of bytes to transfer
2. The I/O DMA controller manages the entire transfer (possibly thousand of bytes in length), arbitrating for the bus
3. When the DMA transfer is complete, the I/O controller interrupts the processor to let it know that the transfer is complete

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



SimMc

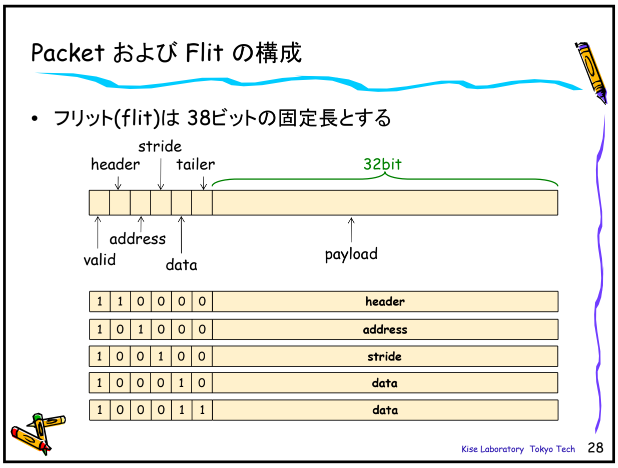


SimMc: Many-Core and NoC Simulator

- 8ビットの整数 x, y を用いて, (x, y) の座標によりコアを指定する. x, yは0～255 の値をとる. ただし, x = 0 及び y = 0 は特別なユニットを表現するために予約する. y = 0 も使わない.
- Core ID は x, y の順序の連結 により生成される16ビットで表現する.
- 現在のバージョンでは, 最下段にサーバコアを用意する.

コアID

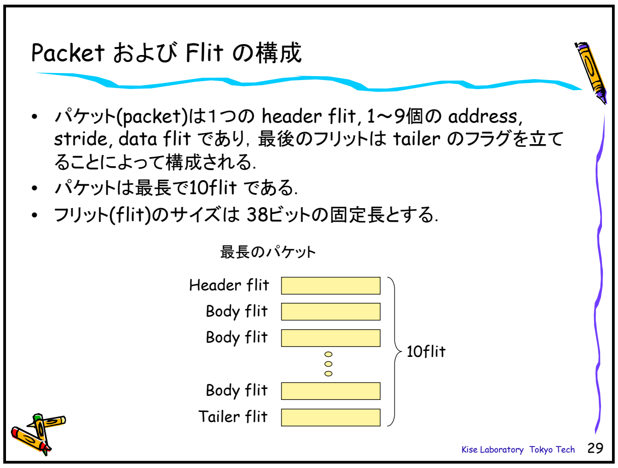
0	0	ID_X	ID_Y



Packet および Flit の構成

- フリット(flit)は 38ビットの固定長とする

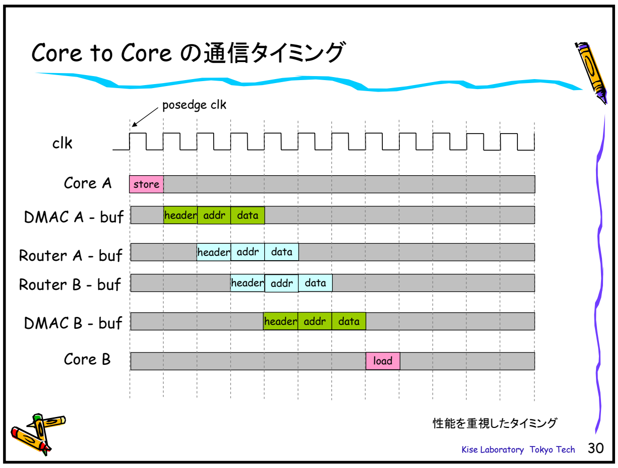
1	1	0	0	0	0	header
1	0	1	0	0	0	address
1	0	0	1	0	0	stride
1	0	0	0	1	0	data
1	0	0	0	1	1	data



Packet および Flit の構成

- パケット(packet)は1つの header flit, 1～9個の address, stride, data flit であり, 最後のフリットは tailer のフラグを立てることによって構成される.
- パケットは最長で10flit である.
- フリット(flit)のサイズは 38ビットの固定長とする.

最長のパケット



Core to Core の通信タイミング

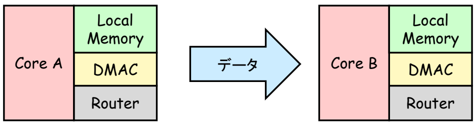
性能を重視したタイミング

Library: Multi-Core library **MC**lib Ver. 1.3

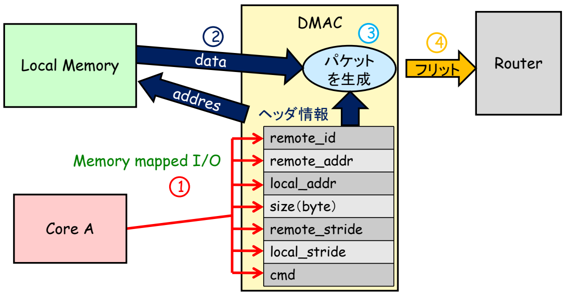
- int MC_init(int *id_x, int *id_y, int *rank_x, int *rank_y);
- void MC_finalize();
- void MC_dma_put(int dst_id, void *remote_addr, void *local_addr,
  size_t size, int remote_stride, int local_stride);
- void MC_dma_get(int get_id, int local_id, void *remote_addr,
  void *local_addr, size_t size, int remote_stride,
  int local_stride);
- int MC_printf(char *format, …);
- void MC_puts(char* s);
- int MC_sprintf(char *buf, char *format, …);
- int MC_sleep(int n);
- int MC_clock(unsigned int*);
- etc

DMA 転送：MC_dma_put

- ローカルコアの保持するデータリモートコアのメモリに転送.
- 下の例は，コアAがMC_dma_putを呼び出し，コアBにデータを送る場合.



MC_dma_putの流れ – Local-Core ～ Router



講義用の計算機の使い方

- ユーザ名 advance で 131.112.16.56 にログイン
  - linuxなど
    - ssh advance@131.112.16.56
    - 講義時に伝えたパスワードでログイン
- 学籍番号でディレクトリを作成して，そこで作業する.
    - mkdir myname
    - cd myname
- 参考ファイルをコピーして実行
    - tar …

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