2011年 後学期

計算機アーキテクチャ 第二 (O)

メニーコアアーキテクチャ

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

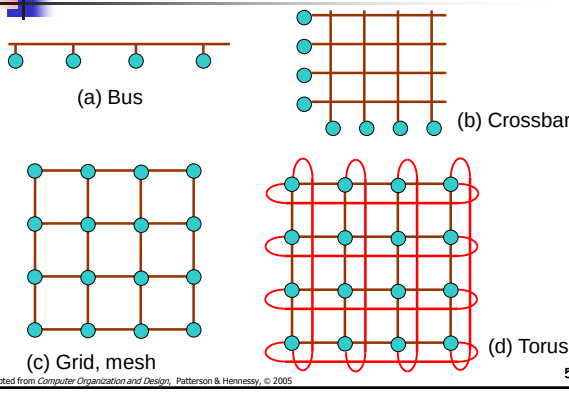
メニーコアアーキテクチャにおける重要な選択肢

- 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

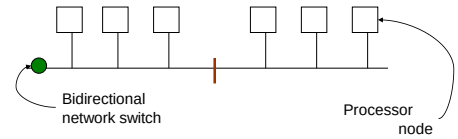
Many-core processor
（メニーコアプロセッサ）

ネットワーク

Interconnection Network

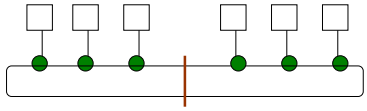


Bus Network



- 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 BE Element Interconnect Bus

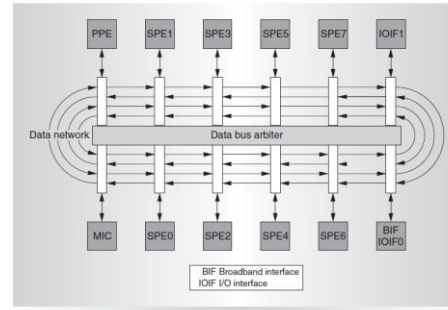
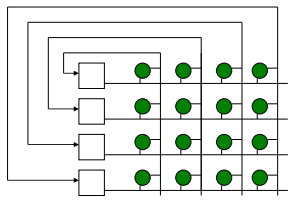


Figure 2. Element interconnect bus (EIB).

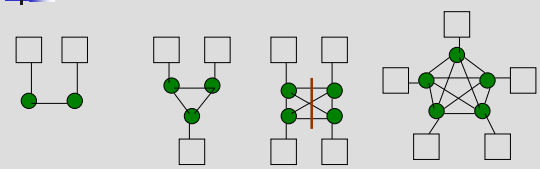
IEEE Micro, Cell Multiprocessor Communication Network: Built for Speed

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

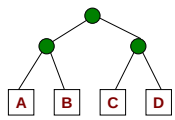
Fully Connected Network



- 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 * $(N/2)^2$

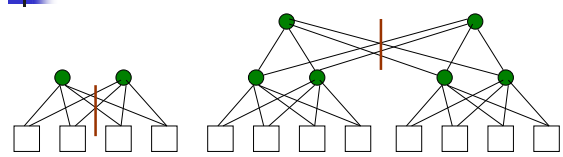
Fat Tree

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



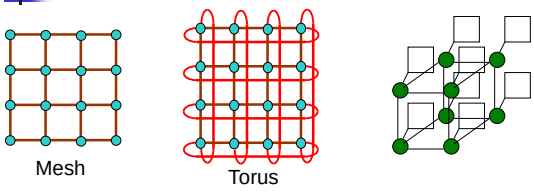
- Any time A wants to send to C, it ties up the upper links, so that B can't send to D.
  - The bisection bandwidth on a tree is horrible - 1 link, at all times
- The solution is to '**thicken**' the upper links.

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

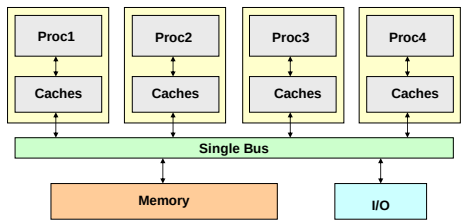
2D and 3D Mesh/Torus Network



- 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 $N^{2/3}$

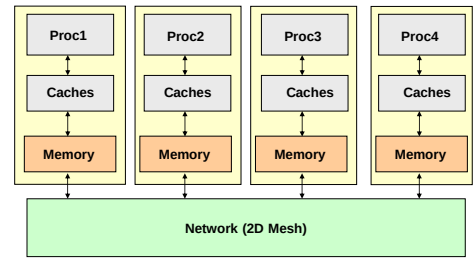
メモリ構成

単一バス結合のマルチプロセッサ/マルチコア，共有メモリ

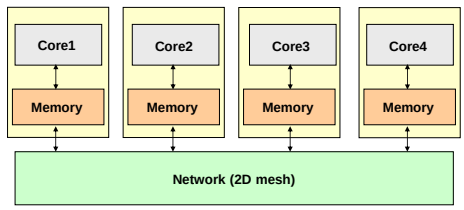


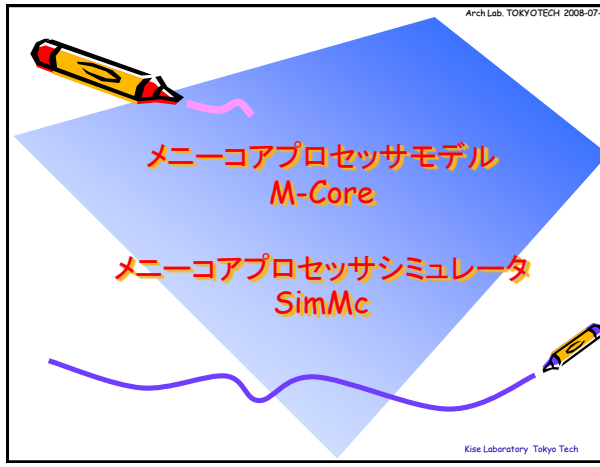
- Caches are used to reduce latency and to lower bus traffic
- Must provide hardware to ensure that caches and memory are consistent (cache coherency)
- Must provide a hardware mechanism to support process synchronization

ネットワーク結合のマルチプロセッサ，**分散メモリ**

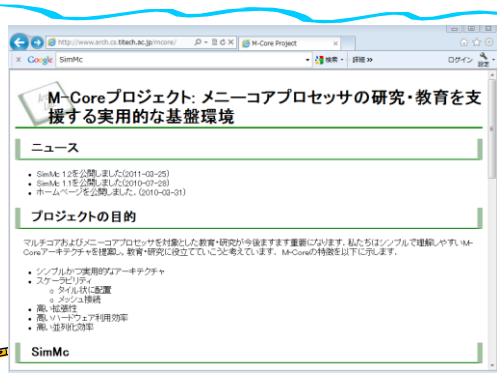


ネットワーク結合のマルチコアプロセッサ

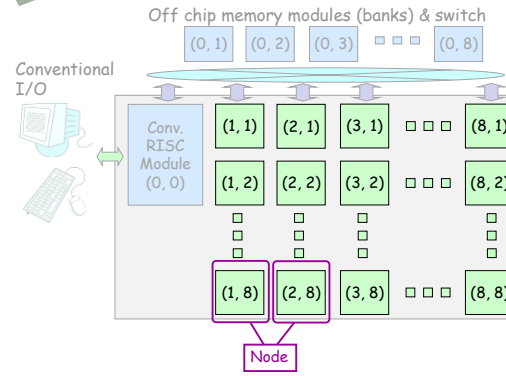




M-Coreプロジェクト www.arch.cs.titech.ac.jp/mcore/

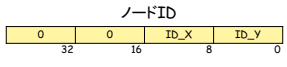


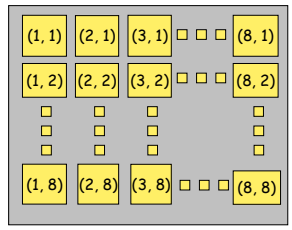
アーキテクチャモデル



M-CoreにおけるノードID

- 8ビットの整数 x, y を用いて，(x, y) の座標によりノードを指定する．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．



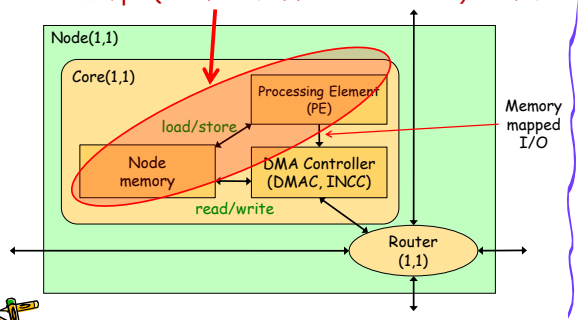


ネットワークアーキテクチャ

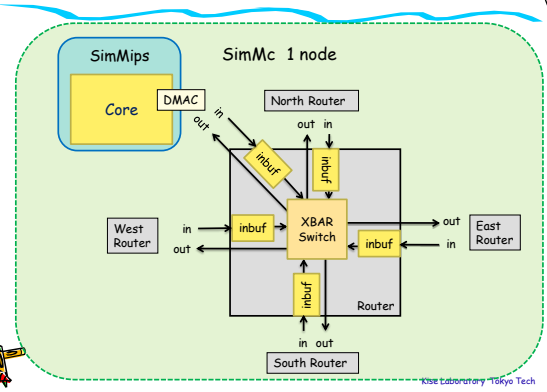
- 2D Mesh Network (2次元メッシュネットワーク)
- ルーティング
  - XY Dimension Order Routing (XY次元順ルーティング)
  - パケットはX方向に進んだ後に，Y方向に進む．
  - 同じ経路を使う複数のパケット間で，パケットの追い越しが生じない．
- ルータアーキテクチャ
  - スイッチング
    - Warm hole, no virtual channel
  - フロー制御
    - Xon / Xoff

ノードアーキテクチャ

SimMips (シングルサイクルのMIPS32プ)ロセッサ

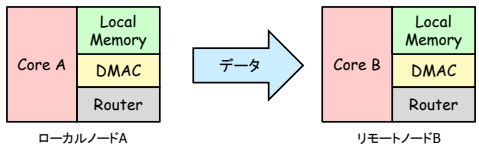


ノード構成とルータアーキテクチャ



DMA 転送：MC_dma_put

- ローカルノードAの保持するデータをリモートノードBのメモリに転送.
- コアAがMC_dma_putを呼び出し，ノードBのローカルメモリにデータを送る.
  - リモートノードのID
  - リモートノードの書き込みアドレス
  - ローカルノードの読み出しアドレス
  - 転送サイズ（バイト）
  - リモートのストライド（通常は4を指定）
  - ローカルのストライド（通常は4を指定）

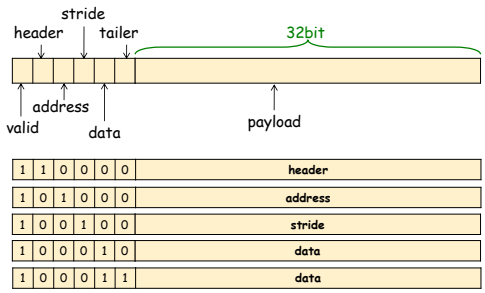


Library: Multi-Core library **MC**lib

- 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

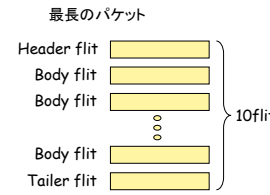
Packet および Flit の構成

- フリット(flit)は 38ビットの固定長とする

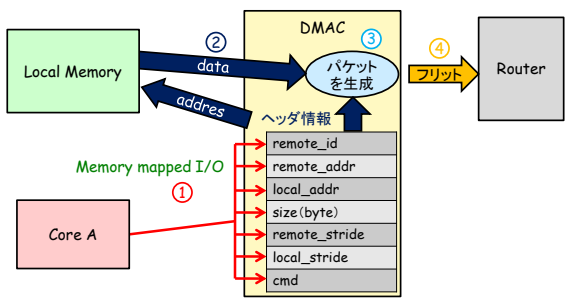


Packet および Flit の構成

- 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.

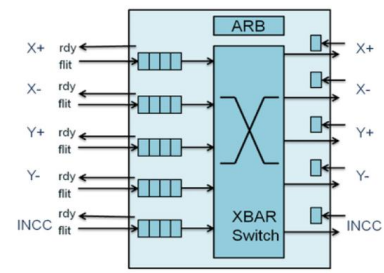


MC_dma_putの流れ - Local-Core ～ Router

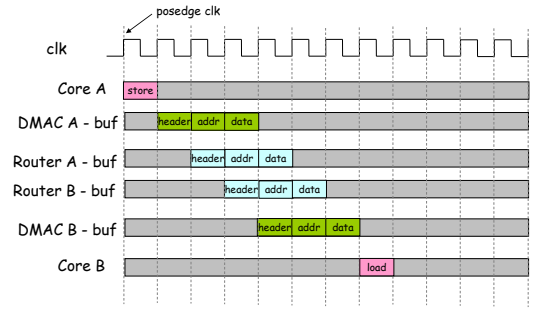


Router Architecture

パケットは入力線を経由して入力バッファに格納され，XBAR switchを通り，適切な方向へと出力される．各入出力ポートは1フリット分のビット幅を備える．Arbiterがラウンドロビン方式でパケットの調停を行う．入力バッファはFIFOであり，最大4フリットを格納する領域を備える．



Core to Core の通信タイミング



性能を重視したタイミング

計算機アーキテクチャと並列性

- 命令内の並列性
- 命令間の並列性
- スレッド・プロセスレベルの並列性

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

講義アンケート

- 教員コード：1600061
- 教員名：吉瀬謙二
- 科目コード：7243
- 科目名：計算機アーキテクチャ第二（O）

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

サンプルプログラム

test10

kterm

```
/**************************************************************************/
/* Many-Core Architecture Research Project       Arch Lab. TOKYO TECH */
/**************************************************************************/
#include "MClib.h"

extern int cx, cy;

/**************************************************************************/
int main(int argc, char *args[])
{
    int id_x, id_y, rank_x, rank_y;
    MC_init(&id_x, &id_y, &rank_x, &rank_y);

    printf("$$ test10: I am core (%d,%d).\n", id_x, id_y);

    MC_finalize();
    return 0;
}

/**************************************************************************/
(END)
```

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

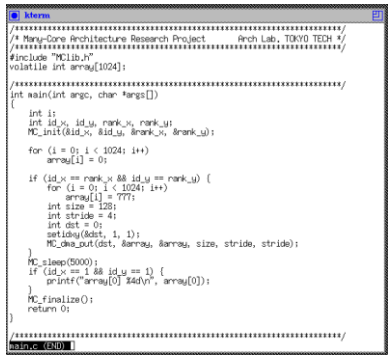
test22

kterm

```
/**************************************************************************/
/* Many-Core Architecture Research Project       Arch Lab. TOKYO TECH */
/**************************************************************************/
#include "MClib.h"

/**************************************************************************/
int main(int argc, char *args[])
{
  int i;
  int id_x, id_y, rank_x, rank_y;
  MC_init(&id_x, &id_y, &rank_x, &rank_y);

  for(i = 0; i < 2; i++) {
    unsigned long long time;

    MC_clock(&time);

    printf("!! $$ test22: I am core (%d,%d) time: %d\n",
           id_x, id_y, (int)time);
  }

main.c
```

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

test31



レポート 提出方法

- 2月17日（金）午後6時までに電子メールで提出
  - report@arch.cs.titech.ac.jp
- 電子メールのタイトル
  - Computer Architecture II（学籍番号）
- 電子メールの内容
  - 氏名，学籍番号
  - レポート
    - **PDFファイルを添付**

レポート課題：マルチコアプロセッサ プログラミング

(課題1)
プロセッサシミュレータSimMcを利用して，与えられるソーティングのプログラム(test60)を4個のコア用に並列化せよ．データ管理用に1コアを用いてもかまわない．
4個のコアを用いて，2倍以上の高速化を達成すること．
コンパイラの最適化オプションを利用しない（-O0を利用する）こと．
ソースコード及び性能向上率を示せ．また，この課題に要した時間を示すこと．

(課題2)
先の(課題1)で用いたプログラムを（必要であれば）修正して，コアの数(1,2,4,8,16)と性能向上率との関係をグラフに示せ．また，この課題に要した時間を示すこと．
ここでも，コンパイラの最適化オプションを利用しない．-O0を利用する．
並列化していない逐次プログラムの性能を1として，グラフを描くこと．

(課題3)
コンパイラの最適化オプションをO3として，コアの数と性能向上率との関係をグラフに示せ．
並列化しない逐次プログラム（O3）の性能を1として，グラフを描くこと．
また，最適化オプションの影響を議論せよ．
この課題に要した時間を示すこと．

講義用の計算機の使い方

- ユーザ名 archo で serv.arch.cs.titech.ac.jp にログイン
  - linuxなど
    - ssh archo@serv.arch.cs.titech.ac.jp
    - 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

講義用の計算機におけるSimMcの使い方

- ssh archo@serv.arch.cs.titech.ac.jp
- mkdir yourID
- chdir yourID
- cp –r /home/archo/kise/SimMc-kadai .
- cd SimMc-kadai/app/test/**test60**/

- make clean
- make
- make run

- ../../../sim/SimMc -x2 –y2 test.out

- ../../../sim/SimMc **-D4** -x2 –y2 test.out

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