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

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

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

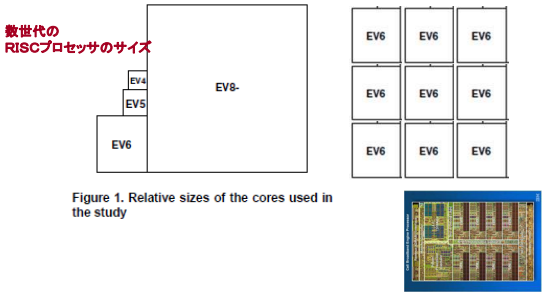


Figure 1. Relative sizes of the cores used in the study

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

A Study of an Infrastructure for Research and Development of Many-Core Processors

Koh Uehara † Shimpei Sato †
Takefumi Miyoshi ‡ Kenji Kise †

†Tokyo Institute of Technology
‡Tokyo Institute of Technology/JST

UPDAS Dec 11 2009 in Hiroshima

Motivation

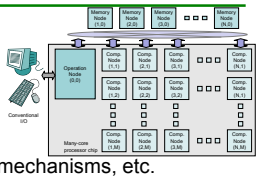
- Multi-core processors are used widely
  - dual core, quad core, etc
- Many-core processors will be realized
  - 100 core, 1000 core, and more
- Research and development will shift
  - multi-core → many-core

- We developed an infrastructure to evaluate
  - Many-core processor architectures
  - Softwares for many-core processors

Table of Contents

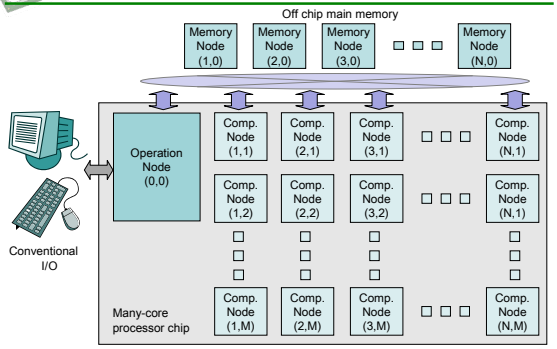
Table of Contents

M-Core: concept

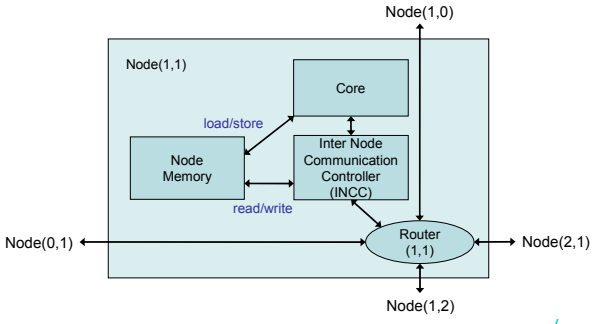


- Scalability
  - Up to 64K cores in a chip
- Effectiveness
  - Many lightweight cores without interruption control mechanisms, etc.
  - Cooperation between cores and system softwares for multi-function
- Parallelism
  - Reduced data communication overheads between cores

M-Core: architecture model

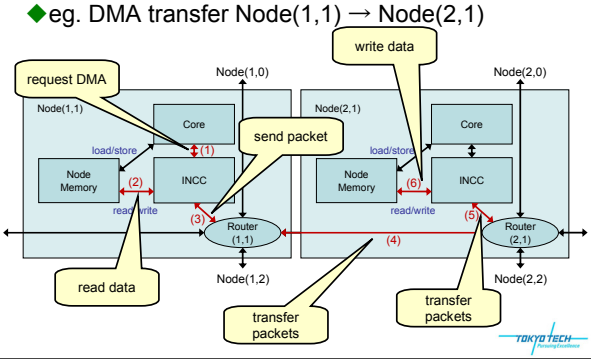


M-Core: Comp. Node



M-Core: communication flow

- eg. DMA transfer Node(1,1) → Node(2,1)

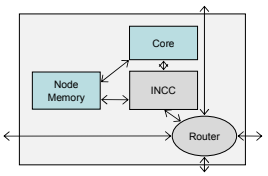


M-Core: DMA transfer

- DMA_PUT: copy data to another node
  - eg. DMA_PUT: Node (1,1) → Node (2,1)

- DMA_GET: copy data from another node
  - eg. DMA_GET: Node (1,1) → Node (2,1)

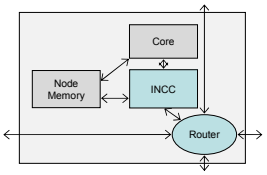
M-Core: Core and node memory spec



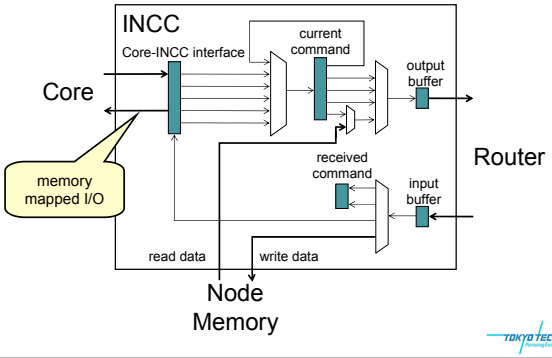
- Core
  - Single cycle processor
  - MIPS 32 ISA
- Node memory
  - 512KB
  - 1 cycle memory access latency
  - 4 ports
    - load/store
    - instruction fetch } Core
    - read
    - write } INCC

M-Core: INCC and router spec

- Topology
  - 2D mesh
- Switching technique
  - Wormhole
- Virtual Channel
  - None
- Flow Control
  - Xon/Xoff
- Routing method
  - X-Y dimension ordered routing
- Router
  - Single cycle router
  - 4(flits) x 5 input buffer



M-Core: INCC micro-architecture



M-Core: router micro-architecture

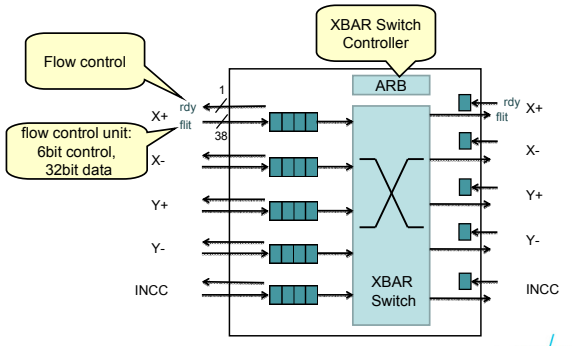


Table of Contents

SimMc: a simulator of M-Core

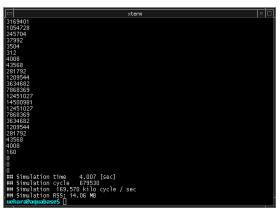
- Simulator implementation
  - Coded in C++
  - SimMips† is used
    - Core and node memory simulation
  - Simple and readable source codes
    - Easy to understand and extend
- Feature
  - Cycle level simulation
  - Change processor architecture by command line option
    - change the number of nodes
    - use another core architecture
  - Debug
    - Display packet transmission history
    - Visualize network traffic

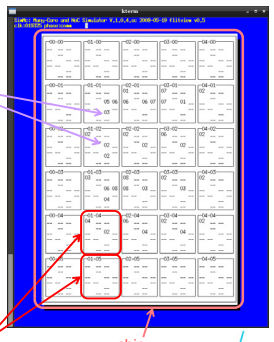
† Naoki Fujieda, et.al: *SimMips A MIPS System Simulator*: WCAE2009,
Naoki Fujieda, et.al: A MIPS System Simulator SimMips for Education and Research of Computer Science, IPSJ journal

SimMc: example of simulation

→ Network visualization

↓ Simulation result





MClib: software library for M-Core

- M-Core API
  - Provided functions
    - int MC_init(int *id_x, int *id_y, int *rank_x, int *rank_y);
    - int MC_fini();
    - int MC_get_id(int *id);
    - int MC_get_rank(int *rank);
    - int MC_clock(unsigned long long *);
    - void MC_dma_put(int dst_id, void *dst, void *src, size_t size, int dst_stride, int src_stride);
    - void MC_dma_put64(int dst_id, uint064_t dst, void *src, size_t size, int src_stride);
    - void MC_dma_get(int src_id, volatile void *dst, volatile void *src, size_t size, int dst_stride, int src_stride);
    - void MC_dma_get64(int remote_id, volatile uint064_t remote, volatile void *local, size_t size, int local_stride);
    - etc
- MClib improves
  - development efficiency of application programs
  - portability of application programs

Related work

- Many-core architectures
  - TRIPS, Raw, Tile64
    - New ISA, complicated network, etc
      → difficult to use architecture
  - M-Core
    - Conventinal ISA, simple network
- Multi/Many-core processor simulators
  - GEMS, SimFlex, M5
    - Support multi-threading, sampling and checkpointing
      → difficult to understand and extend
  - SimMc
    - Written as simple and readable source codes

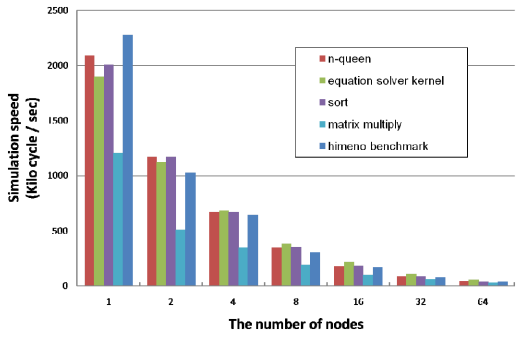
Table of Contents

Evaluation

- Benchmark programs
  - Himeno benchmark
    - Calculation size: small
  - Matrix multiply
    - Matrix size: 128 x 128
  - Equation solver kernel†
    - Matrix size: 256 x 256
  - N-queen
    - Problem size: 12
  - Bitonic sort
    - Data size: 1word x 16K
- Machine spec used in evaluation
  - Intel(R) Xeon(R) CPU X5365 @3.00GHz

† Culler, D. E., et al.: *Parallel Computer Architecture: A Hard-ware/Software Approach*, Morgan Kaufmann (1999).

Simulation speed of SimMc



Performance of M-Core

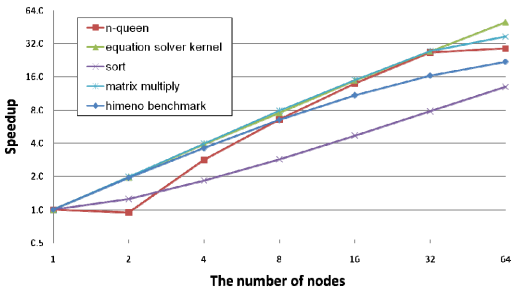
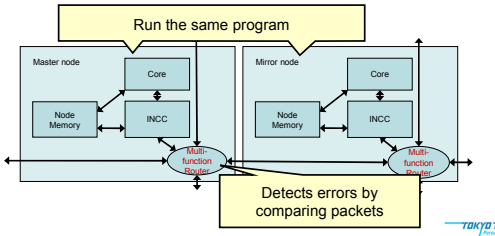


Table of Contents

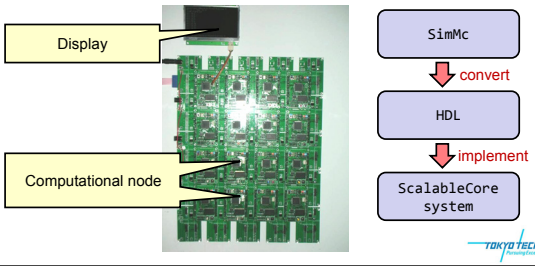
Researches using the infrastructure (1/3)

- SmartCore system
  - High-dependable, high-bandwidth and low latency many-core processors with redundant cores and multi-function routers



Researches using the infrastructure (2/3)

- ScalableCore system
  - High speed emulation system of many-core processors using FPGAs



Researches using the infrastructure (3/3)

- Cache-Core
  - Software managed cache system to reduce memory access latencies

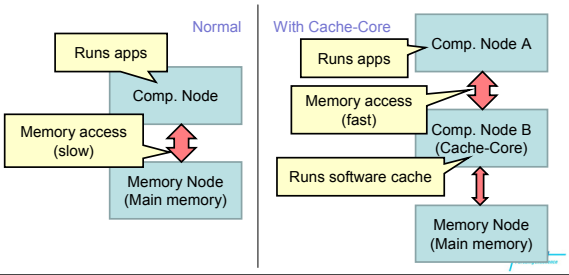


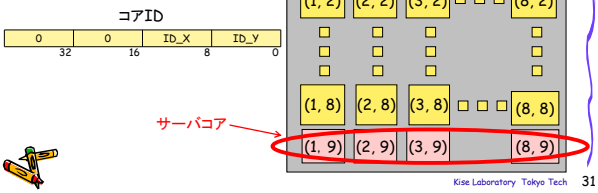
Table of Contents

Conclusion

- The infrastructure for research and development of many-core processors
  - M-Core: simple many-core architecture
  - SimMc: simulator of M-Core
  - MClib: software library for M-Core
- Evaluation
  - M-Core architecture achieves good speedup
- Researches using the infrastructure
  - The infrastructure is useful
- Future work
  - Detail the M-Core architecture
    - Operation node, memory node, cache memory, etc
  - Extend SimMc
    - Various core simulation: pipelined/out-of-ordered/different ISA
    - Cache memory simulation

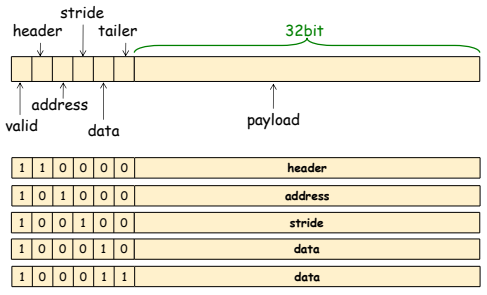
SimMc: Many-Core and NoC Simulator

- 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．



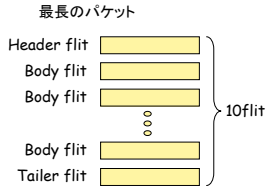
Packet および Flit の構成

- フリット(flit)は 38ビットの固定長とする

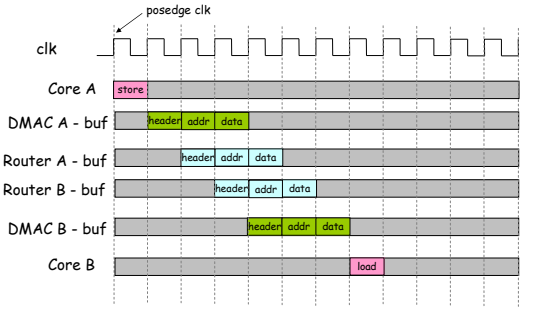


Packet および Flit の構成

- 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．



Core to Core の通信タイミング



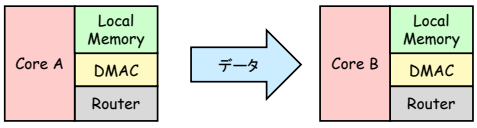
性能を重視したタイミング

Library: Multi-Core library MClib Ver. 1.3

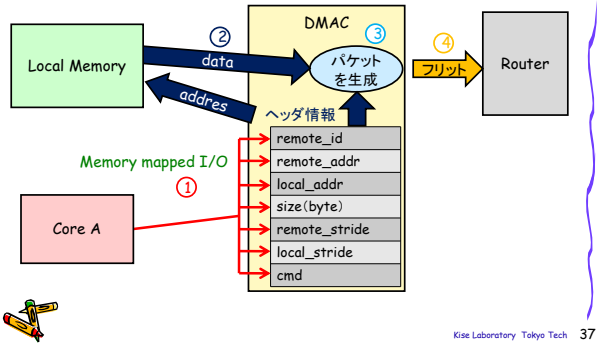
- int MC_init(int *id_x, int *id_y, int *rank_x, int *rank_y);
- void MC_finalize();
- void MC_dma_put(int dst_id, void *remote_addr, void *local_addr, size_t size, int remote_stride, int local_stride);
- void MC_dma_get(int get_id, int local_id, void *remote_addr, void *local_addr, size_t size, int 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



レポート課題：マルチコアプロセッサ（並列）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，与えられる行列積のプログラム (test64) を4個のコア用に並列化せよ．
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)
・(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．

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

レポート課題：マルチコアプロセッサ（並列）プログラミング

(課題4)
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．
4個のコアを用いて，1.3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O3 とする．
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5)
・(課題4)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．
並列化し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．
16コアの場合の速度向上率が高いほど高得点とする．

(課題6，重要)
・プロセッサシミュレータSimMcについての感想をまとめよ．
どこで苦労したか？
どの程度の時間が必要となったか？
期待する改良点．

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

レポート 提出方法

- 2010年1月21日
  講義の開始時に提出

```
[advance@sc440 ~/kise]$ tar xfz ../kadai.tgz
[advance@sc440 ~/kise]$ cd kadai/
[advance@sc440 kadai]$ cd test10
[advance@sc440 test10]$ make
make -C /home/share/M-Core-32bit//lib
make[1]: ディレクトリ `/home/share/M-Core-32bit/lib' に入ります
make[1]: `all' に対して行うべき事はありません．
make[1]: ディレクトリ `/home/share/M-Core-32bit/lib' から出ます
/home/share/cad/mipsel/usr/bin/mipsel-linux-gcc -Wall -O3 main.c -c \
                -I/home/share/M-Core-32bit//lib/
/home/share/cad/mipsel/usr/bin/mipsel-linux-gcc -static main.o /home/s
hare/M-Core-32bit//lib/MClib.o -o test.out
[advance@sc440 test10]$ make run
make -C /home/share/M-Core-32bit//sim
make[1]: ディレクトリ `/home/share/M-Core-32bit/sim' に入ります
make[1]: `all' に対して行うべき事はありません．
make[1]: ディレクトリ `/home/share/M-Core-32bit/sim' から出ます
/home/share/M-Core-32bit//sim/SimMc  test.out
## SimMc: Many-Core and NoC Simulator V.1.0.5 2009-08-27
## [SimMips: Simple Computer Simulator of MIPS Version 0.5.0 2008-11-0
5]
## DEBUG MODE 0, LOG MODE 0
## comp node (1,1) - (4,4) + server (1,5) - (4,5)
## memory node (0,0)
## path node (0,1) - (0,5), (1,0) - (4,0)
## Multi-Core library MClib v0.1.1 2009-07-29
$$ core( 2, 5) : server lock
$$ core( 1, 5) : server barrier
$$ test10: I am core (2,1).
$$ test10: I am core (3,1).
$$ test10: I am core (4,1).
$$ test10: I am core (2,2).
```

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

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