Multi-core API & Compiler Technology

Hironori Kasahara & Jun Shirako
Department of Computer Science & Engineering
Advanced Multicore Processor Research Institute
Waseda University, Tokyo, Japan
http://www.kasahara.cs.waseda.ac.jp

Multi-core Everywhere

Multi-core from embedded to supercomputers

- Consumer Electronics (Embedded)
  - Mobile Phone, Game, Digital TV, Car Navigation, DVD, Camera,
  - IBM/Sony/Toshiba Cell, Fujitsu FR1000, NEC/ARMMPICore&MP211, Panasonic Uniphier, Renesas SH multi-core (4 core RP1, 8 core RP2) Tilera Tile64, SPI Storm-1 (16 VLIW cores)

- PCs, Servers
  - Intel Quad Xeon, Core 2 Quad, Montvale, Nehalem (8 core), 80 core, Larrabee (32 core)
  - AMD Quad Core Opteron, Phenom

- WSs, Deskside & Highend Servers
  - IBM Power4,5,5+,6, Sun Niagara (Sparc T1,T2), Rock

- Supercomputers

High quality application software, Productivity, Cost performance, Low power consumption are important

Ex, Mobile phones, Games

Compiler cooperated multi-core processors are promising to realize the above futures
Parallelization for Multicores

• **A new era of mainstream parallel processing**
  – High performance, power efficiency and productivity on multicore processors with **increasing # cores**
  – **Software must be redesigned for multicore parallelism**
    • As was done for vector and cluster parallelism

• **Who parallelizes the program?**
  – **Parallelization by programmers**
    • New programming model/languages for improved productivity
      – Cilk, Chapel, X10, Fortress, Habanero (at Rice U)
    • Programmers express *ideal parallelism*, compilers and runtime systems extract *useful parallelism* for target multicores
  – **Parallelization by compilers**
    • Normal sequential languages
      – Fortran, C (with some restriction), etc.
    • Automatic parallelizing compilers
      – Parallelizing compiler and multicore API developed in METI/NEDO Japanese national project
METI/NEDO National Project
Multi-core for Real-time Consumer Electronics

<Goal> R&D of compiler cooperative multi-core processor technology for consumer electronics like Mobile phones, Games, DVD, Digital TV, Car navigation systems.

<Period> From July 2005 to March 2008

<Features> • Good cost performance
• Short hardware and software development periods
• Low power consumption
• Scalable performance improvement with the advancement of semiconductor
• Use of the same parallelizing compiler for multi-cores from different vendors using newly developed API

API: Application Programming Interface

(2005.7～2008.3)**

**Hitachi, Renesas, Fujitsu, Toshiba, Panasonic, NEC
OSCAR Multi-Core Architecture

CMP₀ (chip multiprocessor 0)

CPU

LPM/ I-Cache

PE₀

DTC

LDM/ D-cache

DSM

PE₁

PEₙ

Network Interface

Intra-chip connection network
(Multiple Buses, Crossbar, etc)

CSM / L2 Cache

Inter-chip connection network
(Crossbar, Buses, Multistage network, etc)

CSM: central shared mem.
LDM: local data mem.
DSM: distributed shared mem.
LPM: local program mem.
DTC: Data Transfer Controller
FVR: frequency / voltage control register
Renesas-Hitachi-Waseda 8 core RP2 Chip Photo and Specifications

- **Process Technology**: 90nm, 8-layer, triple-Vth, CMOS
- **Chip Size**: 104.8mm² (10.61mm x 9.88mm)
- **CPU Core Size**: 6.6mm² (3.36mm x 1.96mm)
- **Supply Voltage**: 1.0V–1.4V (internal), 1.8/3.3V (I/O)
- **Power Domains**: 17 (8 CPUs, 8 URAMs, common)

IEEE ISSCC08: Paper No. 4.5, M.Ito, … and H. Kasahara, “An 8640 MIPS SoC with Independent Power-off Control of 8 CPUs and 8 RAMs by an Automatic Parallelizing Compiler”
OSCAR Parallelizing Compiler

• Improve effective performance, cost-performance and productivity and reduce consumed power
  – Multigrain Parallelization
    • Exploitation of parallelism from the whole program by use of coarse-grain task parallelism among loops and subroutines, near fine grain parallelism among statements in addition to loop parallelism
  – Data Localization
    • Automatic data distribution for distributed shared memory, cache and local memory on multiprocessor systems.
  – Data Transfer Overlapping
    • Data transfer overhead hiding by overlapping task execution and data transfer using DMA or data pre-fetching
  – Power Reduction
    • Reduction of consumed power by compiler control of frequency, voltage and power shut down with hardware supports.
OSCAR Multigrain Parallelizing Compiler

- Automatic Parallelization
  - Multigrain Parallel Processing
  - Data Localization
  - Data transfer Overlapping
  - Compiler Controlled Power Saving Scheme

- Compiler cooperative Multi-core architecture
  - OSCAR Multi-core Architecture
  - OSCAR Heterogeneous Multiprocessor Architecture

- Commercial SMP machines
Generation of coarse grain tasks

- **Macro-tasks (MTs)**
  - Block of Pseudo Assignments (BPA): Basic Block (BB)
  - Repetition Block (RB): natural loop
  - Subroutine Block (SB): subroutine
Earliest Executable Condition Analysis for coarse grain tasks (Macro-tasks)

A Macro Flow Graph

A Macro Task Graph
Automatic processor assignment in su2cor

- Using 14 processors
  - Coarse grain parallelization within DO400 of subroutine

$N_{PG}, N_{PE} = [PG, PE]$
MTG of Su2cor-LOOPS-DO400

- Coarse grain parallelism PARA_ALD = 4.3
Data-Localization
Loop Aligned Decomposition

- Decompose multiple loops (Doall and Seq) into CARs and LRs considering inter-loop data dependence.
  - Most data in LR can be passed through LM.
  - LR: Localizable Region, CAR: Commonly Accessed Region
Data Localization Group
dlg0
dlg1
dlg2
dlg3

MTG

MTG after Division

A schedule for two processors
Power Reduction by Dynamic Voltage Frequency Control and Power Shutdown Control

- Shortest execution time mode

Realtime processing mode with deadline constraints
An Example of Machine Parameters for the Power Saving Scheme

- **Functions of the multiprocessor**
  - Frequency of each proc. is changed to several levels
  - Voltage is changed together with frequency
  - Each proc. can be powered on/off

<table>
<thead>
<tr>
<th>state</th>
<th>FULL</th>
<th>MID</th>
<th>LOW</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>1</td>
<td>1/2</td>
<td>1/4</td>
<td>0</td>
</tr>
<tr>
<td>voltage</td>
<td>1</td>
<td>0.87</td>
<td>0.71</td>
<td>0</td>
</tr>
<tr>
<td>dynamic energy</td>
<td>1</td>
<td>3/4</td>
<td>1/2</td>
<td>0</td>
</tr>
<tr>
<td>static power</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **State transition overhead**

<table>
<thead>
<tr>
<th>state</th>
<th>FULL</th>
<th>MID</th>
<th>LOW</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL</td>
<td>0</td>
<td>40k</td>
<td>40k</td>
<td>80k</td>
</tr>
<tr>
<td>MID</td>
<td>40k</td>
<td>0</td>
<td>40k</td>
<td>80k</td>
</tr>
<tr>
<td>LOW</td>
<td>40k</td>
<td>40k</td>
<td>0</td>
<td>80k</td>
</tr>
<tr>
<td>OFF</td>
<td>80k</td>
<td>80k</td>
<td>80k</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>state</th>
<th>FULL</th>
<th>MID</th>
<th>LOW</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL</td>
<td>0</td>
<td>20</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>MID</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>LOW</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>OFF</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>0</td>
</tr>
</tbody>
</table>

*delay time [u.t.]  energy overhead [μJ]*
Image of Generated Multigrain Parallelized Code using the developed Multicore API
(The API is compatible with OpenMP)

Centralized scheduling code

Distributed scheduling code

1st layer

Thread group0

Thread group1

2nd layer

2nd layer
Compilation Flow Using OSCAR Multicore API

Application Program
Fortran or Parallelizable C
(Sequential program)

Waseda Univ.
OSCAR
Parallelizing
Compiler
- Coarse grain task parallelization
- Global data localization
- Data transfer overlapping using DMA
- Power reduction control using DVFS, Clock and Power gating

OSCAR: Optimally Scheduled Advanced Multiprocessor
API: Application Program Interface

Parallelized Fortran or C
program with API

Proc0
Code with directives
Thread 0

Proc1
Code with directives
Thread 1

...:

Backend compiler
API Analyze
Existing sequential compiler

Backend compiler
API Analyze
Existing sequential compiler

OpenMP Compiler

OSCAR API for Real-time Low Power High Performance Multicores
Directives for thread generation, memory, data transfer using DMA, power managements

Generation of parallel machine codes using sequential compilers

Executable on various multicores

Multicore from Vendor A

Multicore from Vendor B

Shred memory servers

Machine codes

Machine codes
OSCAR API

• Targeting mainly realtime consumer electronics devices
  – embedded computing
  – various kinds of memory architecture
    • SMP, local memory, distributed shared memory, ...

• Developed with Japanese companies
  – Fujitsu, Hitachi, NEC, Toshiba, Panasonic, Renesas
  – Supported by METI/NEDO

• Based on the subset of OpenMP
  – very popular parallel processing API
  – shared memory programming model

• Six Categories
  – Parallel Execution
  – Memory Mapping
  – Data Transfer
  – Power Control
  – Timer
  – Synchronization
Parallel Execution

- Start of parallel execution
  - `#pragma omp parallel sections (C)`
  - `!$omp parallel sections (Fortran)`
- Specifying critical section
  - `#pragma omp critical (C)`
  - `!$omp critical (Fortran)`
- Enforcing an order of the memory operations
  - `#pragma omp flush (C)`
  - `!$omp flush (Fortran)`
- These are from OpenMP.
Thread Execution Model

```c
#pragma omp parallel sections
{
    #pragma omp section
    main_vpc0();
    #pragma omp section
    main_vpc1();
    #pragma omp section
    main_vpc2();
    #pragma omp section
    main_vpc3();
}
```

Parallel Execution
(A thread and a core are bound one-by-one)

VPC: Virtual Processor Core
Memory Mapping

• Placing variables on an onchip centralized shared memory (onchipCSM)
  • #pragma oscar onchipshared (C)
  • !$oscar onchipshared (Fortran)

• Placing variables on a local data memory (LDM)
  • #pragma omp threadprivate (C)
  • !$omp threadprivate (Fortran)
  • This directive is an extension to OpenMP

• Placing variables on a distributed shared memory (DSM)
  • #pragma oscar distributedshared (C)
  • !$oscar distributedshared (Fortran)
Data Transfer

• Specifying data transfer lists
  – #pragma oscar dma_transfer (C)
  – !$oscar dma_transfer (Fortran)
  – Containing following parameter directives
• Specifying a contiguous data transfer
  – #pragma oscar dma_contiguous_parameter (C)
  – !$oscar dma_contiguous_parameter (Fortran)
• Specifying a stride data transfer
  – #pragma oscar dma_stride_parameter
  – !$oscar dma_stride_parameter
  – This can be used for scatter/gather data transfer
• Data transfer synchronization
  – #pragma oscar dma_flag_check
  – !$oscar dma_flag_check
Power Control

• Making a module into specifying frequency and voltage state
  – #pragma oscar fvcontrol (C)
  – !$oscar fvcontrol (Fortran)
  – state examples
    • 100: max frequency
    • 50: half frequency
    • 0: clock off
    • -1: power off

• Getting a frequency and voltage state of a module
  – #pragma oscar get_fvstatus (C)
  – !$oscar get_fvstatus (Fortran)
Timer

- Getting an elapsed wall clock time in microseconds
  - #pragma oscar get_current_time (C)
  - !$oscar get_current_time (Fortran)
- For realtime execution
Synchronization

• Specifying a hierarchical group barrier
  – #pragma oscar group_barrier (C)
  – !$oscar group_barrier (Fortran)
Performance of OSCAR Compiler on IBM p6 595 Power6 (4.2GHz) based 32-core SMP Server

OpenMP codes generated by OSCAR compiler accelerate IBM XL Fortran for AIX Ver.12.1 by **3.3 times** on the average

Compile Option:

(1) Sequential: -O3 -qarch=pwr6, XLF: -O3 -qarch=pwr6 -qsmp=auto, OSCAR: -O3 -qarch=pwr6 -qsmp=noauto
(2) Sequential: -O5 -q64 -qarch=pwr6, XLF: -O5 -q64 -qarch=pwr6 -qsmp=auto, OSCAR: -O5 -q64 -qarch=pwr6 -qsmp=noauto
(Others) Sequential: -O5 -qarch=pwr6, XLF: -O5 -qarch=pwr6 -qsmp=auto, OSCAR: -O5 -qarch=pwr6 -qsmp=noauto
Performance of OSCAR Compiler Using the Multicore API on Intel Quad-core Xeon

- OSCAR Compiler accelerate Intel Compiler ver.10.1 by 2.1 times on the average
Performance of OSCAR compiler on 16 cores SGI Altix 450 Montvale server

OSCAR Compiler accelerate Intel Fortran Itanium Compiler revision 10.1 by 2.3 times on the average
Performance of OSCAR compiler on NEC NaviEngine (ARM-NEC MPcore)

- OSCAR compiler gave us 3.43 times speedup against 1 core on ARM/NEC MPCore with 4 ARM 400MHz cores
SH4A Multicore SoC Chip

Process Technology
90nm, 8-layer, triple-Vth, CMOS

Chip Size
97.6mm² (9.88mm x 9.88mm)

Supply Voltage
1.0V (internal), 1.8/3.3V (I/O)

Power Consumption
0.6 mW/MHz/CPU @ 600MHz (90nm G)

Clock Frequency
600MHz

CPU Performance
4320 MIPS (Dhrystone 2.1)

FPU Performance
16.8 GFLOPS

I/D Cache
32KB 4way set-associative (each)

ILRAM/OLRAM
8KB/16KB (each CPU)

URAM
128KB (each CPU)

Package
FCBGA 554pin, 29mm x 29mm

ISSCC07 Paper No.5.3, Y. Yoshida, et al., “A 4320MIPS Four-Processor Core SMP/AMP with Individually Managed Clock Frequency for Low Power Consumption”
Performance of OSCAR Compiler Using the Developed API on 4 core (SH4A) OSCAR Type Multicore

3.31 times speedup on the average for 4 cores against 1 core
Fujitsu FR-1000 Multicore Processor

FR-V Multi-core Processor

- Fast I/O Bus
  - Memory Bus: 64bit x 2ch / 266MHz
  - System Bus: 64bit / 178MHz
Performance of OSCAR Compiler Using the multicore API on Fujitsu FR1000 Multicore

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG2enc</td>
<td>1.00</td>
<td>1.94</td>
<td>2.81</td>
<td>3.76</td>
<td>1.00</td>
<td>1.80</td>
<td>2.12</td>
<td>2.54</td>
<td>1.00</td>
<td>1.98</td>
<td>2.90</td>
<td>3.75</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
</tr>
<tr>
<td>MPEG2dec</td>
<td>1.00</td>
<td>1.80</td>
<td>2.12</td>
<td>2.54</td>
<td>1.00</td>
<td>1.98</td>
<td>2.90</td>
<td>3.75</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
</tr>
<tr>
<td>MP3enc</td>
<td>1.00</td>
<td>1.98</td>
<td>2.90</td>
<td>3.75</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
</tr>
<tr>
<td>JPEG 2000enc</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
<td>1.00</td>
<td>1.98</td>
<td>2.55</td>
<td>3.47</td>
</tr>
</tbody>
</table>

3.38 times speedup on the average for 4 cores against a single core execution
### Renesas-Hitachi-Waseda 8 core RP2 Chip Photo and Specifications

<table>
<thead>
<tr>
<th>Process Technology</th>
<th>90nm, 8-layer, triple-Vth, CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip Size</td>
<td>104.8mm² (10.61mm x 9.88mm)</td>
</tr>
<tr>
<td>CPU Core Size</td>
<td>6.6mm² (3.36mm x 1.96mm)</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>1.0V–1.4V (internal), 1.8/3.3V (I/O)</td>
</tr>
<tr>
<td>Power Domains</td>
<td>17 (8 CPUs, 8 URAMs, common)</td>
</tr>
</tbody>
</table>

IEEE ISSCC08: Paper No. 4.5, M. ITO, ... and H. Kasahara, “An 8640 MIPS SoC with Independent Power-off Control of 8 CPUs and 8 RAMs by an Automatic Parallelizing Compiler”
Processing Performance on the Developed Multicore Using Automatic Parallelizing Compiler

Speedup against single core execution for audio AAC encoding

*) Advanced Audio Coding

Speedups

Numbers of processor cores

1.0
1.9
3.6
5.8
Power Reduction by OSCAR Parallelizing Compiler for Secure Audio Encoding

AAC Encoding + AES Encryption with 8 CPU cores

Without Power Control (Voltage: 1.4V)

With Power Control (Frequency, Resume Standby: Power shutdown & Voltage lowering 1.4V-1.0V)

Avg. Power: 5.68 [W]  88.3% Power Reduction  Avg. Power: 0.67 [W]
Power Reduction by OSCAR Parallelizing Compiler for MPEG2 Decoding

MPEG2 Decoding with 8 CPU cores

Without Power Control (Voltage : 1.4V)

With Power Control (Frequency, Resume Standby: Power shutdown & Voltage lowering 1.4V-1.0V)

Avg. Power 5.73 [W] 73.5% Power Reduction Avg. Power 1.52 [W]
Low Power High Performance Multicore Computer with Solar Panel

- Clean Energy Autonomous
- Servers operational in deserts
Power Reduction of MPEG2 Decoding by OSCAR Compiler on Fujitsu FR1000 Having 4 VLIW Cores

Without Power Control
(Voltage: 1.25V)

With Power Control
(Voltage: 1.25V with Clock Gating)

Average Power
1.93[W]

23% Reduction

Average Power
1.48[W]
Conclusions

- OSCAR compiler and Multicore API for consumer electronics
  - High effective performance
  - Low power consumption
  - Short software development
- The OSCAR compiler with API boosts up the vendors compiler performance on various multicores and servers
  - 3.3 times on IBM p595 SMP server using Power6
  - 2.1 times on Intel Quad core Xeon
  - 2.3 times on SGI Altix450 using Intel Itanium2 (Montvale)
  - 88% power reduction by the compiler power control on the Renesas-Hitachi-Waseda 8 core (SH4A) multicore RP2 for realtime secure AAC encoding
  - 70% power reduction on the FR1000 multicore for MPEG2 decoding