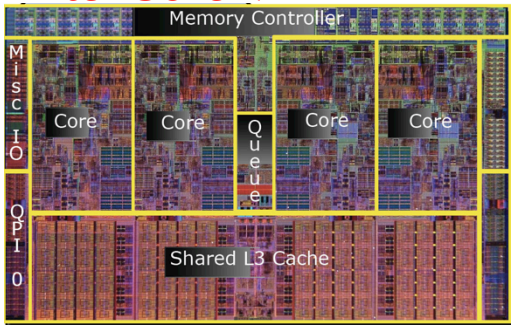
OSCAR API for Low-Power Multicores and Manycores, and API Standard Translator

Keiji Kimura, Waseda University

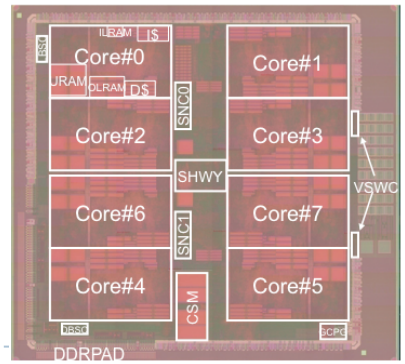
Current Multicores and Their Architecture

- Multicores are Everywhere Now!
  - Server, Desktop, and Embedded
- Multicores for Server and Desktop Usage
  - Ordinary Shared Memory Architecture
- Multicores for Embedded Usage
  - Various Kinds of Memory Architecture
    - Local Memory (Scratch Pad Memory), Distributed Shared Memory, On-chip/Off-chip Shared Memory, …
  - Power Control Mechanisms
    - DVFS, Clock Gating, Power Gating, …

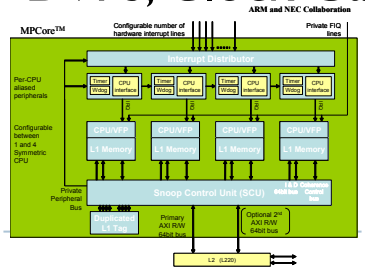
Intel Core i7



Renesas/Hitachi/
Waseda
RP2



NEC/ARM
MPCore



Current Application Development Environment for Multicores

- Parallel API
  - pthread (SMP)
    - Old thread library
  - OpenMP (Shared Memory), MPI (Distributed Memory)
    - for Scientific Applications
  - Co-array Fortran (PGAS), UPC (PGAS)
    - Language extension
  - MCAPI (Distributed Memory)
    - for Embedded Applications
    - Message Passing API
  - NO Good API for Low-Power and Real-time Multicores!
- Parallelizing Compilers for Scientific Applications (Fortran Applications)
  - Several aggressive compilers from academia
    - Polaris, SUIF, CETUS, Pluto, (OSCAR), …
    - Source-to-source Compiler
  - Parallelizing Compilers for Low-Power and Real-time Multicores?

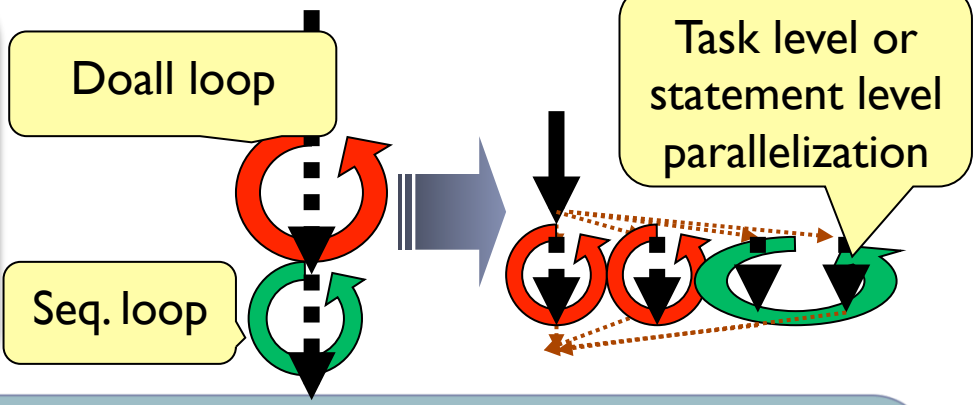
Our Effort with OSCAR Compiler

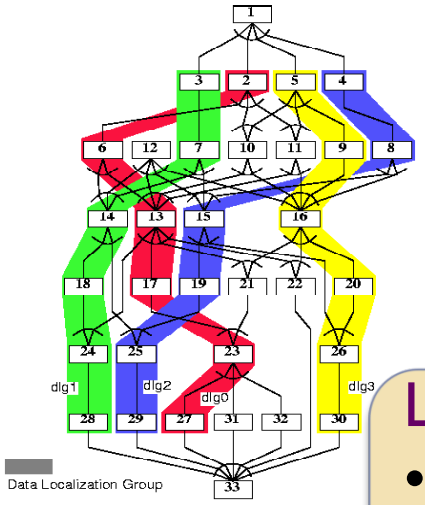
- OSCAR Compiler
  - Multigrain Parallel Processing
  - Data Locality Optimization
  - Data Transfer Optimization
  - Low-Power Optimization
  - Good for Embedded Applications in addition to Scientific Applications
- OSCAR API
  - Parallel API for Low-power and Real-time Multicores
    - Developed by CATS, DENSO, e-SOL, Fujitsu, Fujitsu Laboratory, GAIO Technology, Hitachi, MITSUBISHI Electric, NEC, Olympus, Panasonic, Renesas Electronics, Renesas Solutions, Toshiba, Toho University, Nagoya University and Waseda University in METI/NEDO Project
  - Supporting various kinds of memory architectures
    - Local Memory (Scratch Pad Memory), Distributed Shared Memory, On-chip/Off-chip Centralized Shared Memory
  - Supporting power control mechanism
    - DVFS, Clock Gating, Power Gating
  - Supporting accelerators and many-cores
    - from Version 2.0
  - Using as an interface between OSCAR Compiler and various Multicores
  - Now Open! (Download from http://www.kasahara.cs.waseda.ac.jp/)

OSCAR Automatic Paralelizing Compiler

Multigrain Parallel Processing

- Hierarchical and Global Parallelization
  - Coarse grain task parallel
  - Loop iteration parallel
  - Statement level parallel



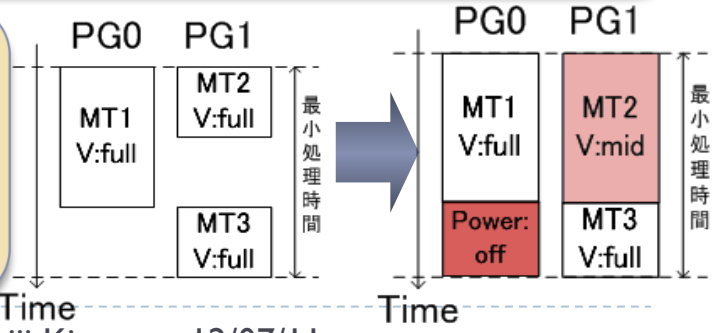


Data Locality Optimization

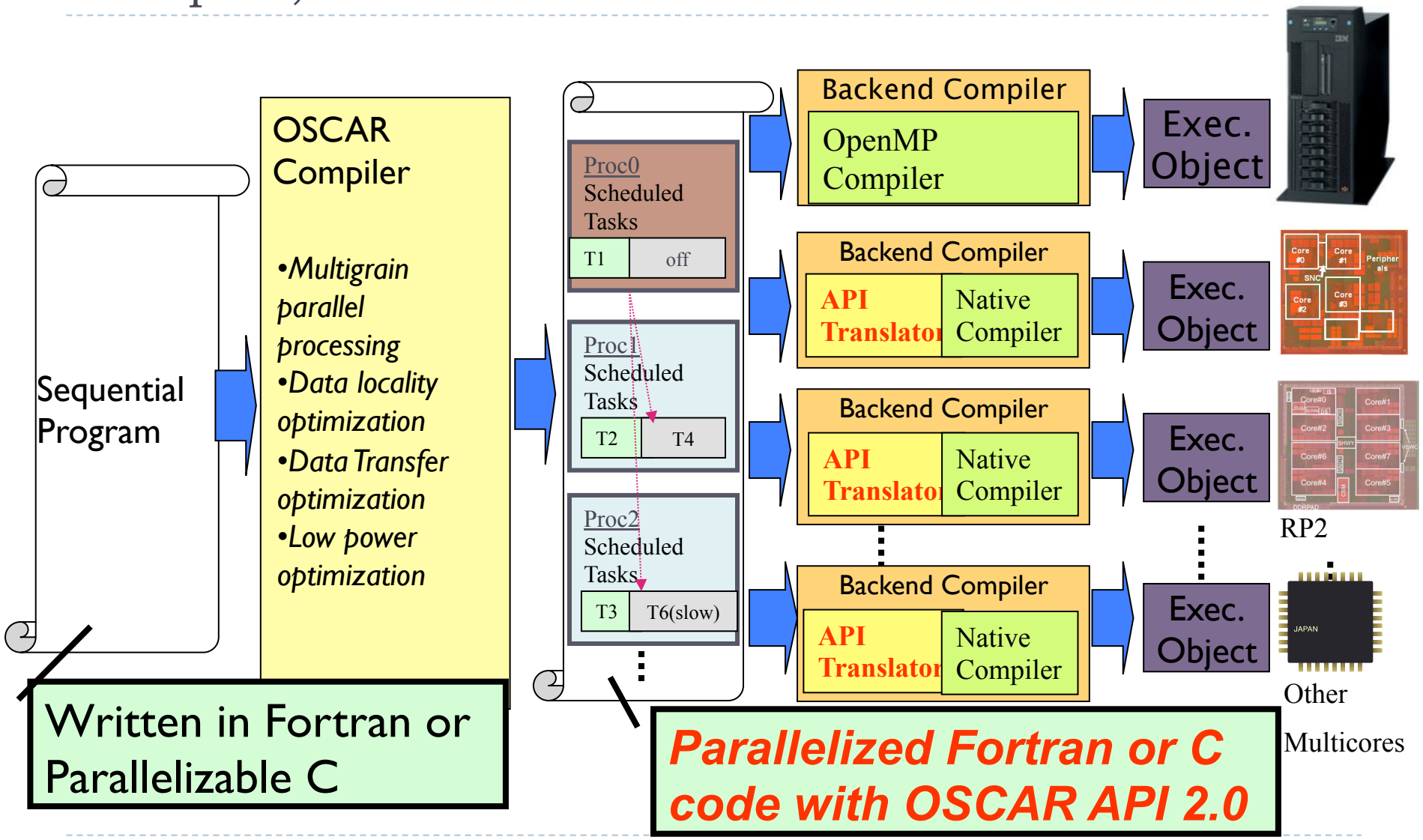
- Task (or loop) decomposition considering cache size or local memory size
- Task scheduling considering data affinity

Low power optimization

- Power scheduling with DVFS and power gating by software



Application Development Environment with OSCAR Compiler, OSCAR API and API Standard Translator

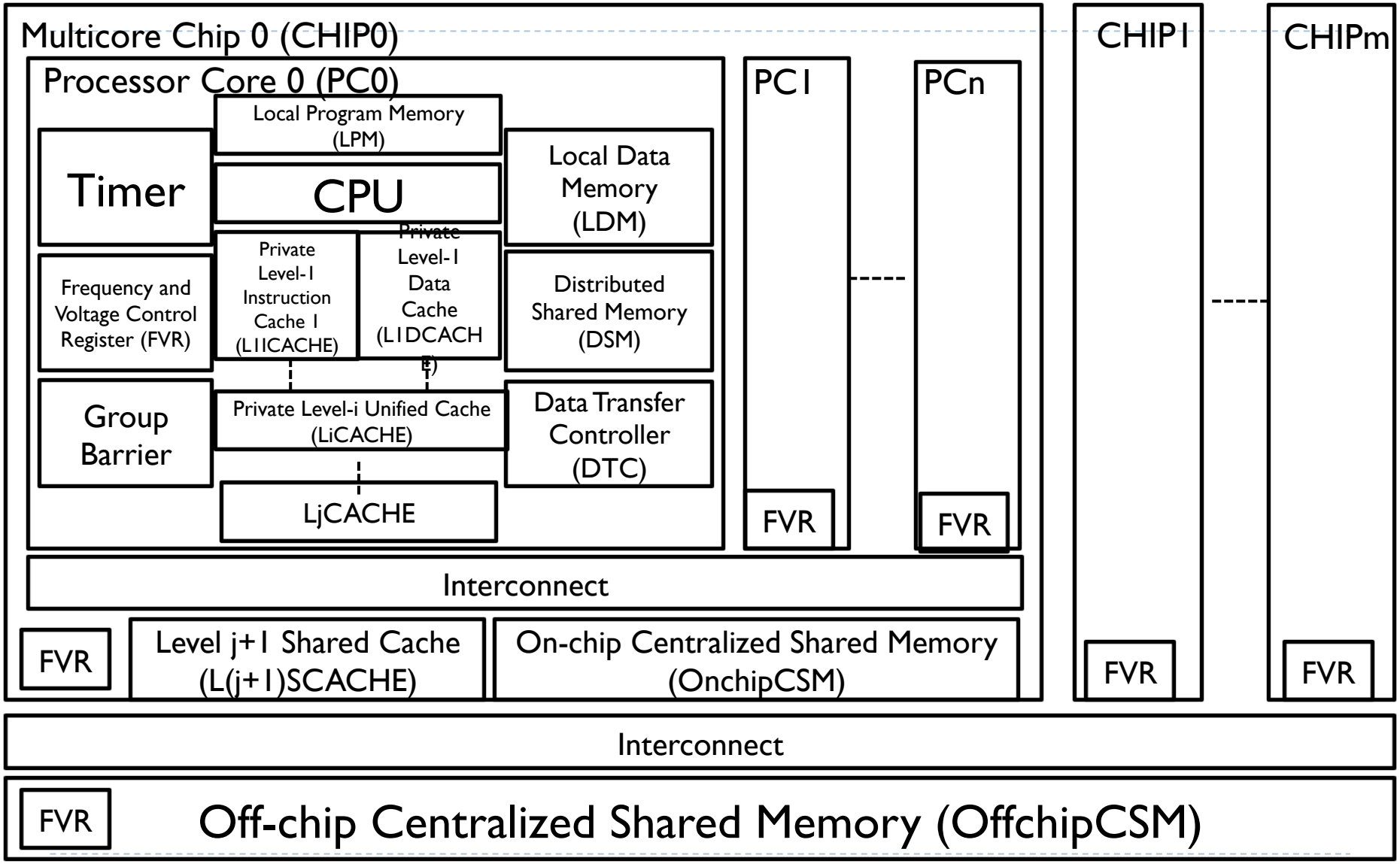


Overview of OSCAR API v2.0

- Targeting mainly real-time consumer electronics devices
  - Embedded computing
  - Various kinds of memory architecture
    - SMP, local memory, distributed shared memory, non-coherent cache ...
  - Power control mechanisms
  - Accelerators
- Based on the subset of OpenMP
  - Very popular parallel processing API
  - Shared memory programming model
  - Supporting both of C and Fortran
- Eight Categories
  - Parallel Execution
  - Memory Mapping
  - Data Transfer
  - Power Control
  - Timer
  - Synchronization
  - Accelerator
  - Cache Control

OSCAR Multicore Architecture

A Target Multicore doesn't have to equip with all modules.



List of Directives (22 directives)

- Parallel Execution API
  - parallel sections (*)
  - flush (*)
  - critical (*)
  - execution
- Memoay Mapping API
  - threadprivate (*)
  - distributedshared
  - onchipshared
- Synchronization API
  - groupbarrier
- Data Transfer API
  - dma_transfer
  - dma_contiguous_parameter
  - dma_stride_parameter
  - dma_flag_check
  - dma_flag_send

(* from OpenMP)

- Power Control API
  - fvcontrol
  - get_fvstatus
- Timer API
  - get_current_time
- Accelerator
  - accelerator_task_entry
- Cache Control
  - cache_writeback
  - cache_selfinvalidate
  - complete_memop
  - noncacheable
  - aligncache

2 hint directives for OSCAR compiler
- accelerator_task
- oscar_comment

from V2.0

Image of One-time Single Level Thread Creation with OSCAR API

```
int flag1_1;
int myver;
#pragma omp \
      threadprivate(myver)
int main() {
#pragma omp parallel sections
  {
#pragma omp section
    { main_VC0(); }
#pragma omp section
    { main_VC1(); }
…
#pragma omp section
#pragma oscar \
    distributedshared(flag1_1)
    { main_VC4(); }
…
  }
  return 0;
}
```

```
void
main_VC0() {
```

MT1_1 for Core0

```
#pragma omp flush
  flag1_1 = myver;
#pragma omp flush
```

MT1_2 for Core0

```
…
}
```

```
void
main_VC1() {
```

MT1_1 for Core1

MT1_2 for Core1

```
…
}
```

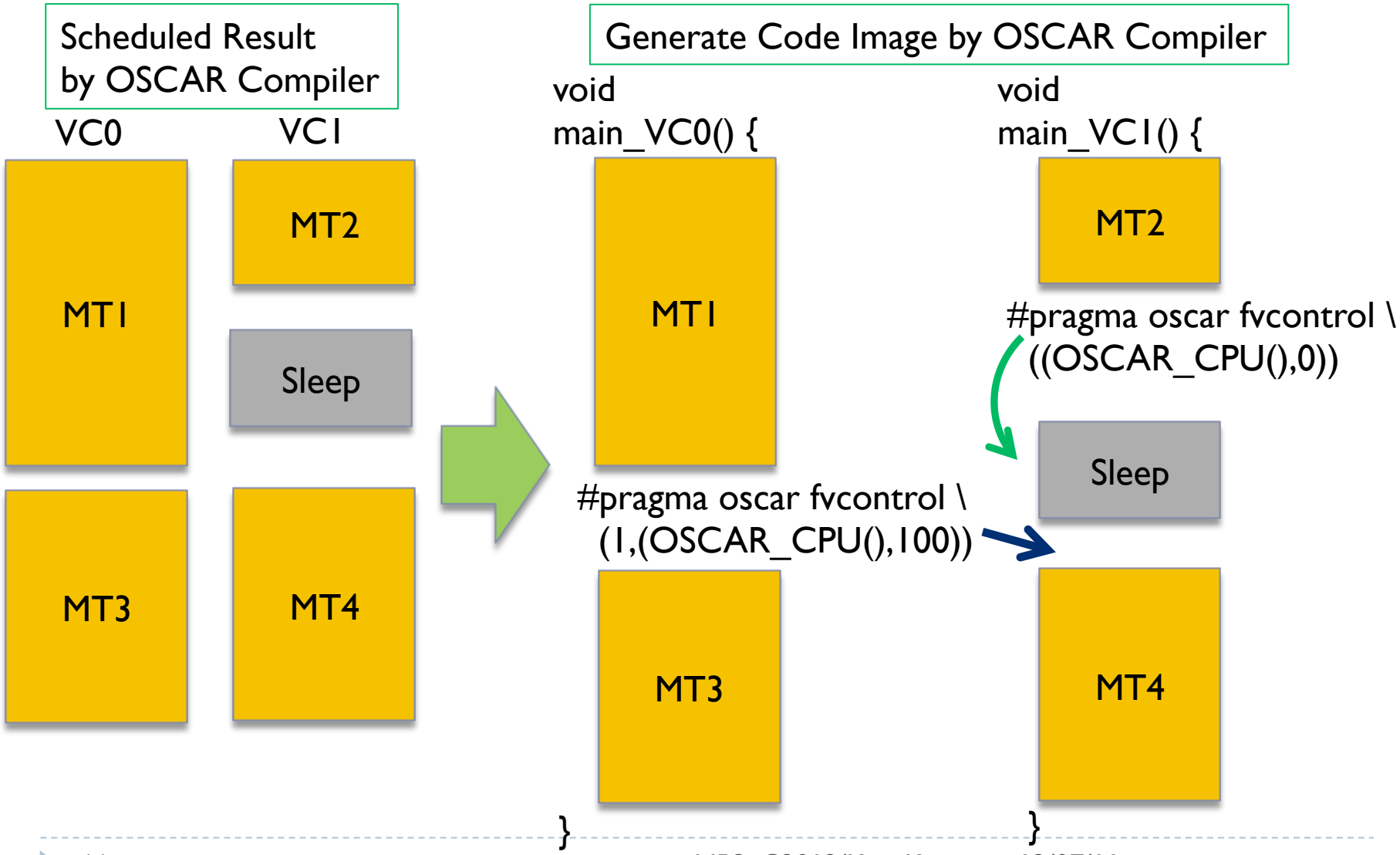
```
void
main_VC4() {
  /* waiting for
      signal from VC0 */
  do {
#pragma omp flush
  } while (flag1_1
             != myver);
```

MT1_3 for Core4

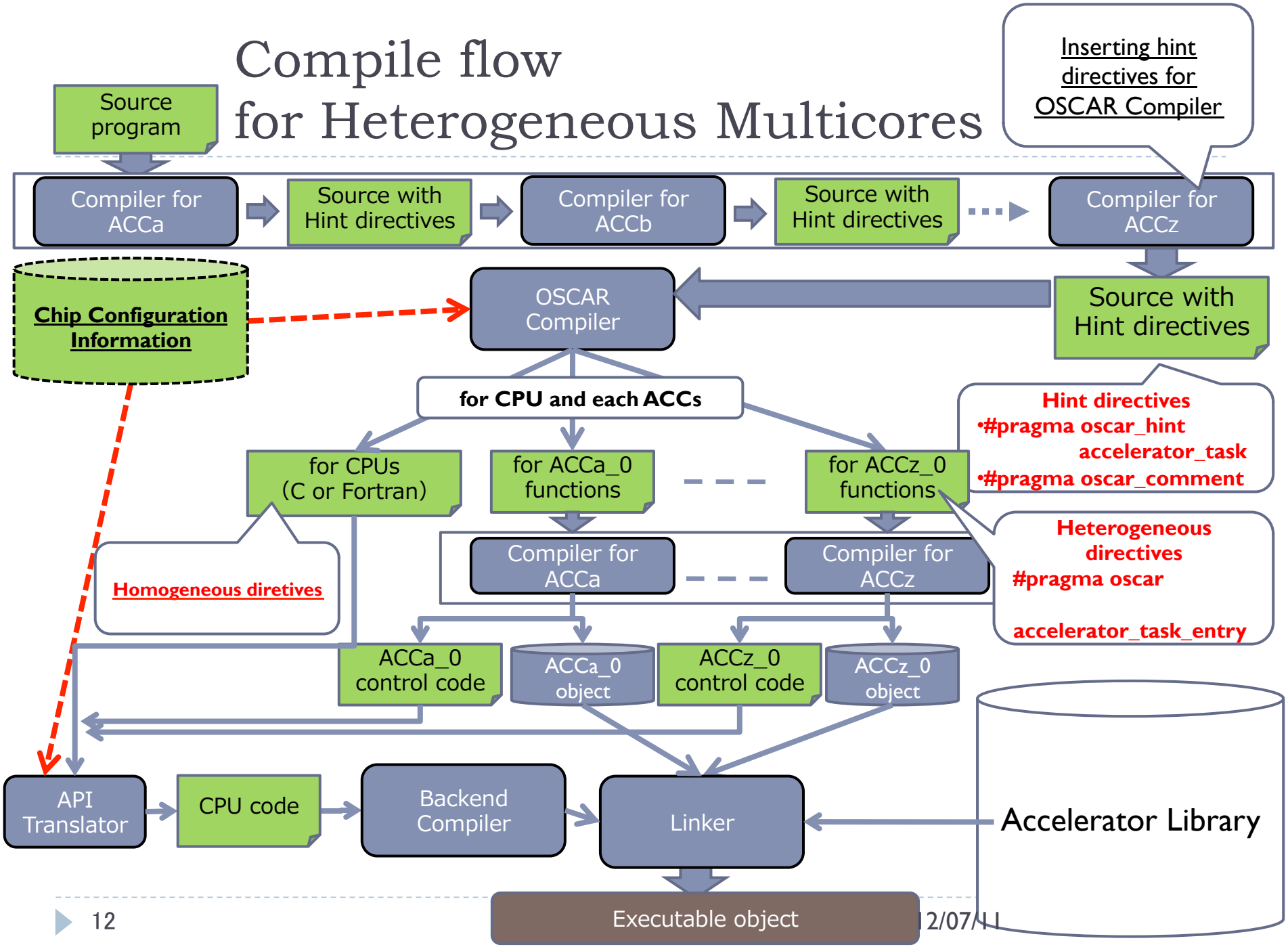
```
…
}
```

Generated code image by OSCAR Compiler

Image of Low-Power Optimization with OSCAR API



Compile flow for Heterogeneous Multicores



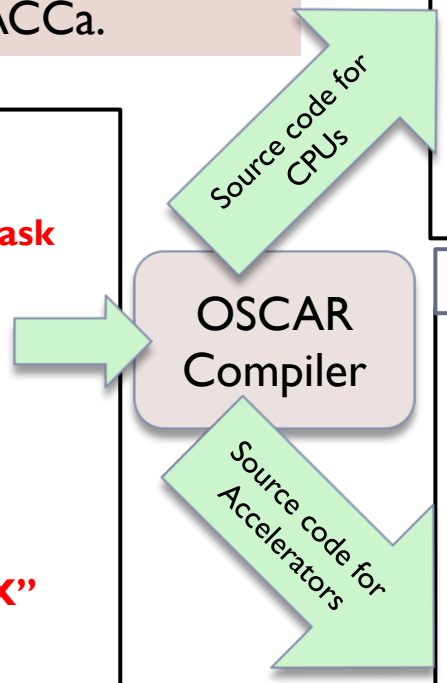
Code image with hint directives and OSCAR heterogeneous directives

"task_func" function is annotated with "oscar_hint" directive.

•loop1, loop2, loop3 can be compiled by the compiler for ACCa.
•func1 is a library function for ACCa.

Sample.c

```
main() {
  int a, b[10];
#pragma oscar_hint acccelerator_task
 (ACCa)  cycle(1000) in(a, b[0:9])
out(b[0:9])
  task_func();
}
task_func() {
  for (…) {…}// loop1
  for (…) {…}// loop2
#pragma oscar_comment "XXXXX"
  func1 (…) // func1
  for (…) {…}// loop3
}
```



Sample.omp.c — for 1CPU+1ACC

```
main() {
#pragma omp parallel sections
  {
#pragma omp section
{
   /*VC0,VPC 0 */
  …
  oscartask_CTRL0_task_func();
  …
  }
}
```

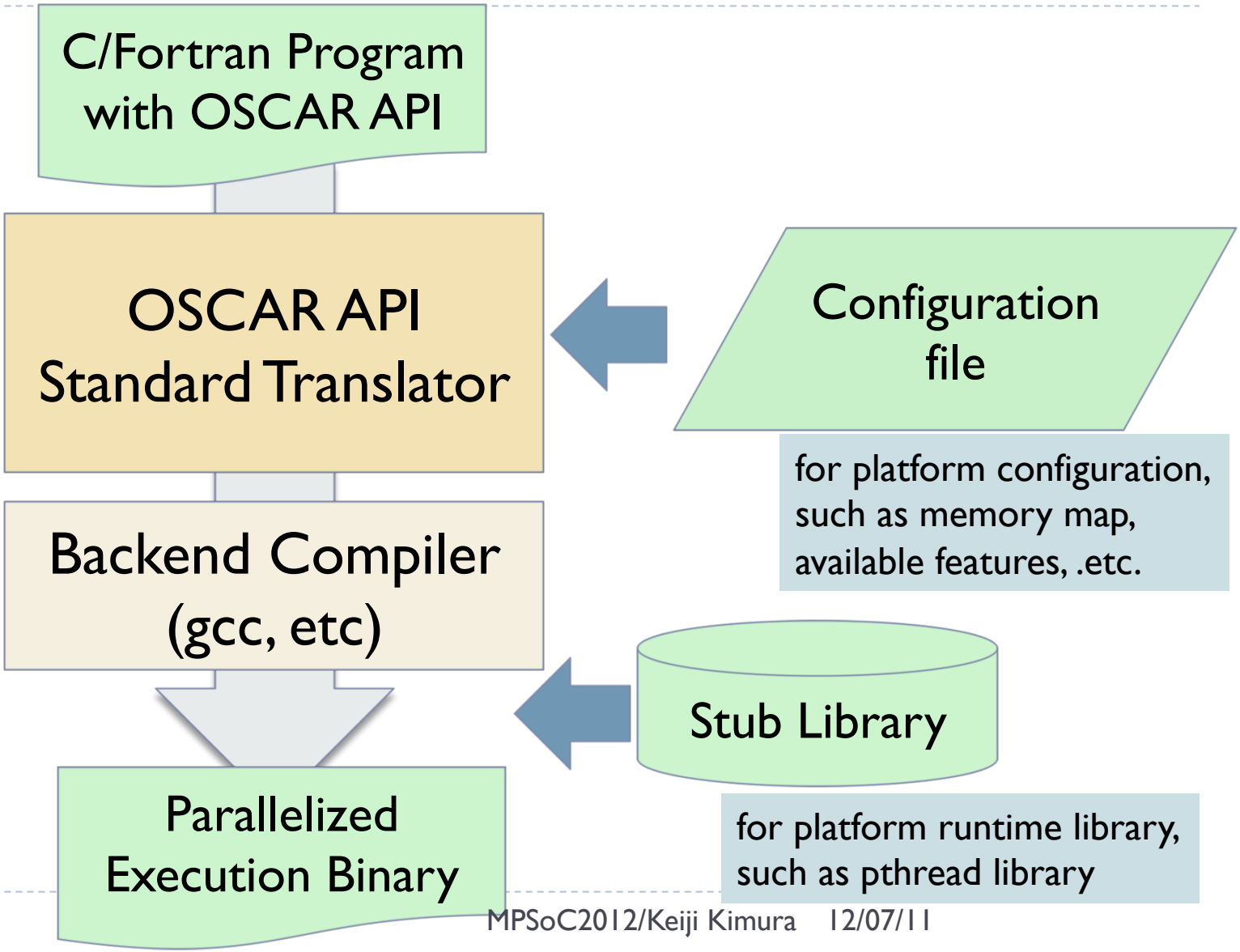
Sample.VC1.c — for ACCa_0

```
#pragma oscar accelerator_task_entry
controller(0)
oscartask_CTRL0_task_func
oscartask_CTRL0_task_func() {
 for (…) {…}// loop1
  for (…) {…}// loop2
#pragma oscar_comment "XXXXX"
  oscarlib_CTRL0_ACCEL1_func1 (…) //
func1
  for (…) {…}// loop3
}
```

Cache Control API

- cache_writeback
  - Write back an indicated data on a cache to an off-chip memory
- cache_selfinvalidate
  - Invalidate an indicated data from a cache
- complete_memop
  - Waite for preceding memory operation on a processor core
- noncacheable
  - Locate an indicated data onto non-cacheable area
- aligncache
  - Align an indicated data onto a cache line of the last-level-cache

API Standard Translator



Translation Example

```
int myver;
#pragma omp \
      threadprivate(myver)
int main() {
#pragma omp parallel sections
  {
#pragma omp section
    { main_VC0(); }
#pragma omp section
    { main_VC1(); }
…
#pragma omp section
    { main_VC4(); }
…
  }
  return 0;
}
```

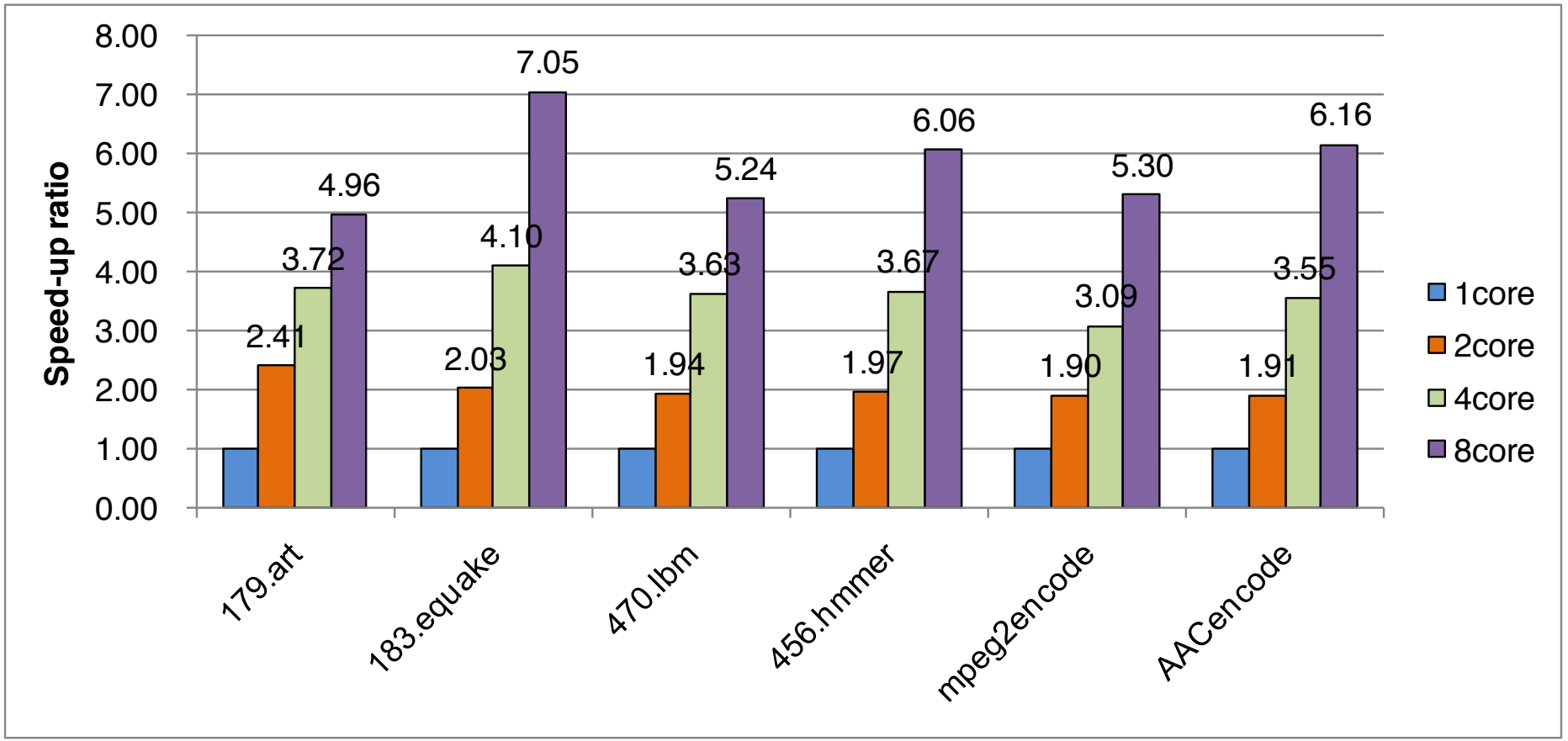
```
int __attribute((section("OSCAR_LDM")))myver;

int main() {
  int thr1, …, thr4;
  oscar_thread_create(&thr1, thread_function_001, 0);
  …
  oscar_thread_create(&thr3, thread_function_004, 3);
  thread_function_000();
  oscar_thread_join(thr1);
  …
  oscar_thread_join(thr4);
…
  }
  return 0;
}
void thread_function_000(void)
{
    main_VC0();
}
```

Evaluations

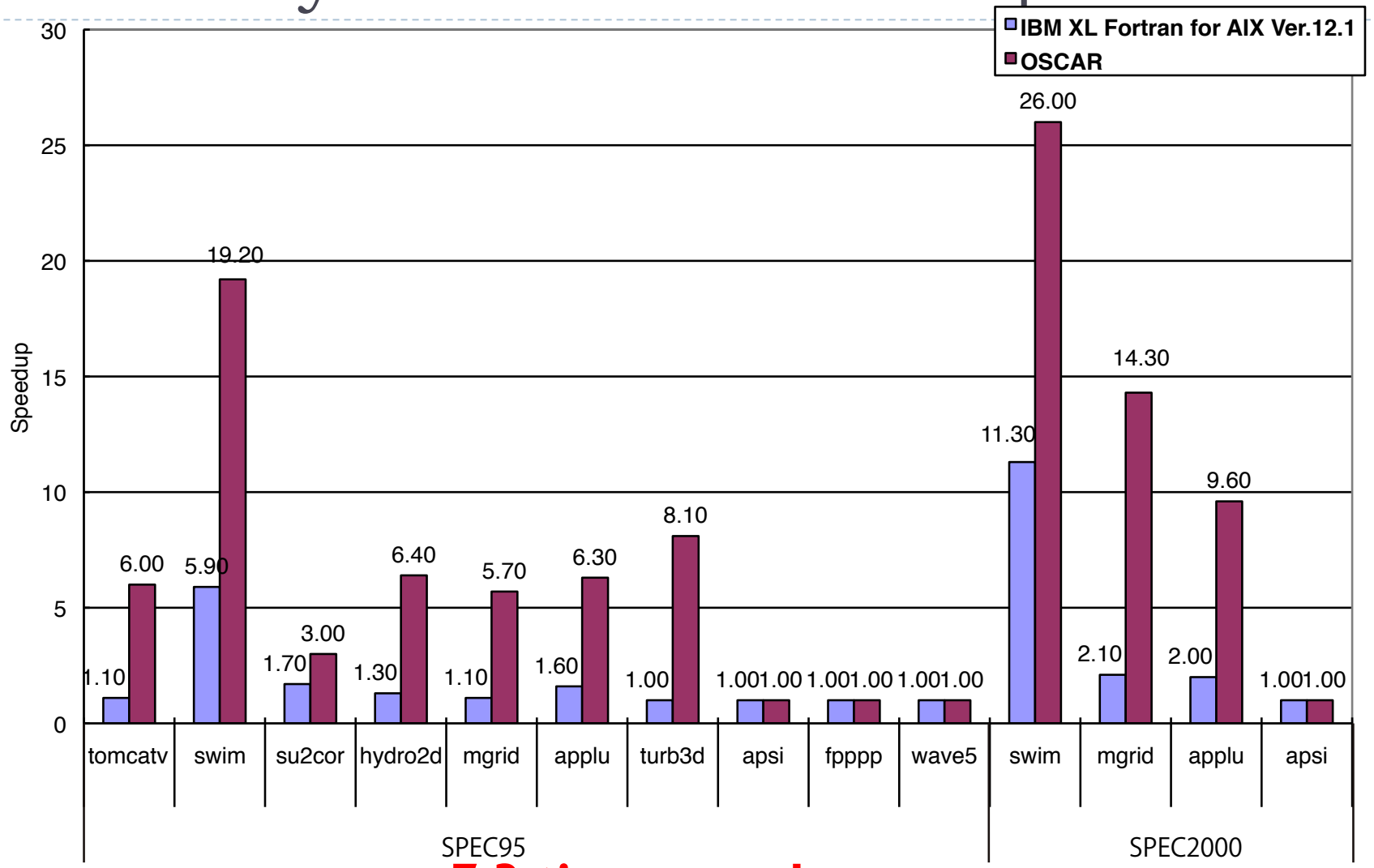
- Scalability Evaluation
  - IBM p5 550Q (Power5+ x 4chips: 8 cores)
  - Renesas, Hitachi, Waseda RP2 (SH4A 8 cores: 4 cores are used)
    - Parallelizable C Applications
  - IBM p6 595 (Power6 x 16chips: 32 cores)
    - SPEC 95, 2000 fp Applications (Fortran)
- Power Optimization Evaluation
  - RP2 with 8 cores
  - AAC Encoder available on the market from Renesas Technology
  - MPEG2 Decoder from MediaBench (modified into Parallelizable C)
  - Real-time Execution Mode

Scalability Evaluation on IBM p5 550Q



5.8 times speedup on average

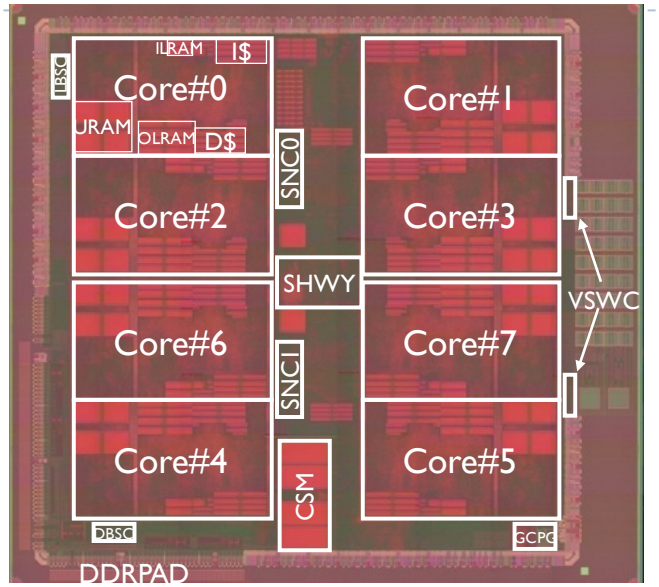
Scalability Evaluation on IBM p6 595

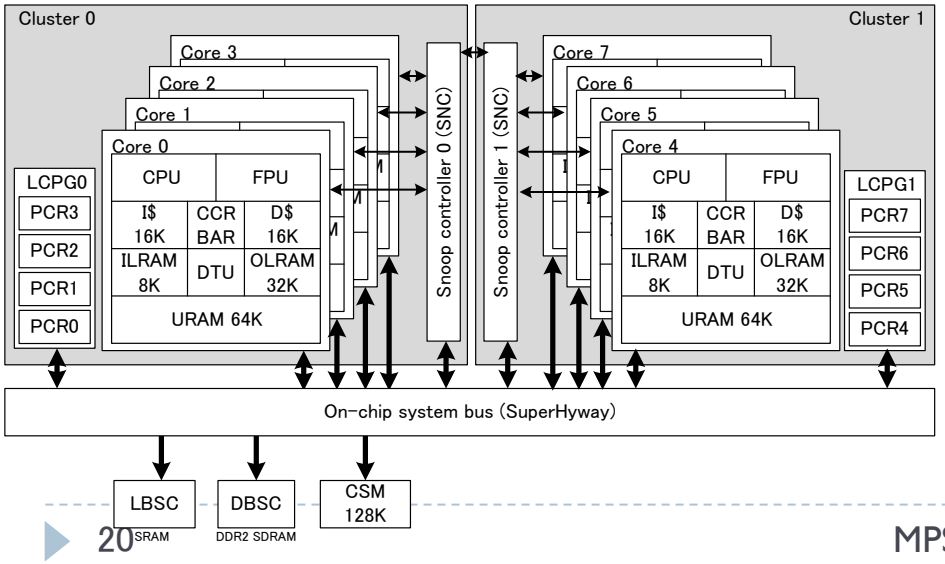


7.3 time speedup on average

3.3 times acceleration over XL Fortran

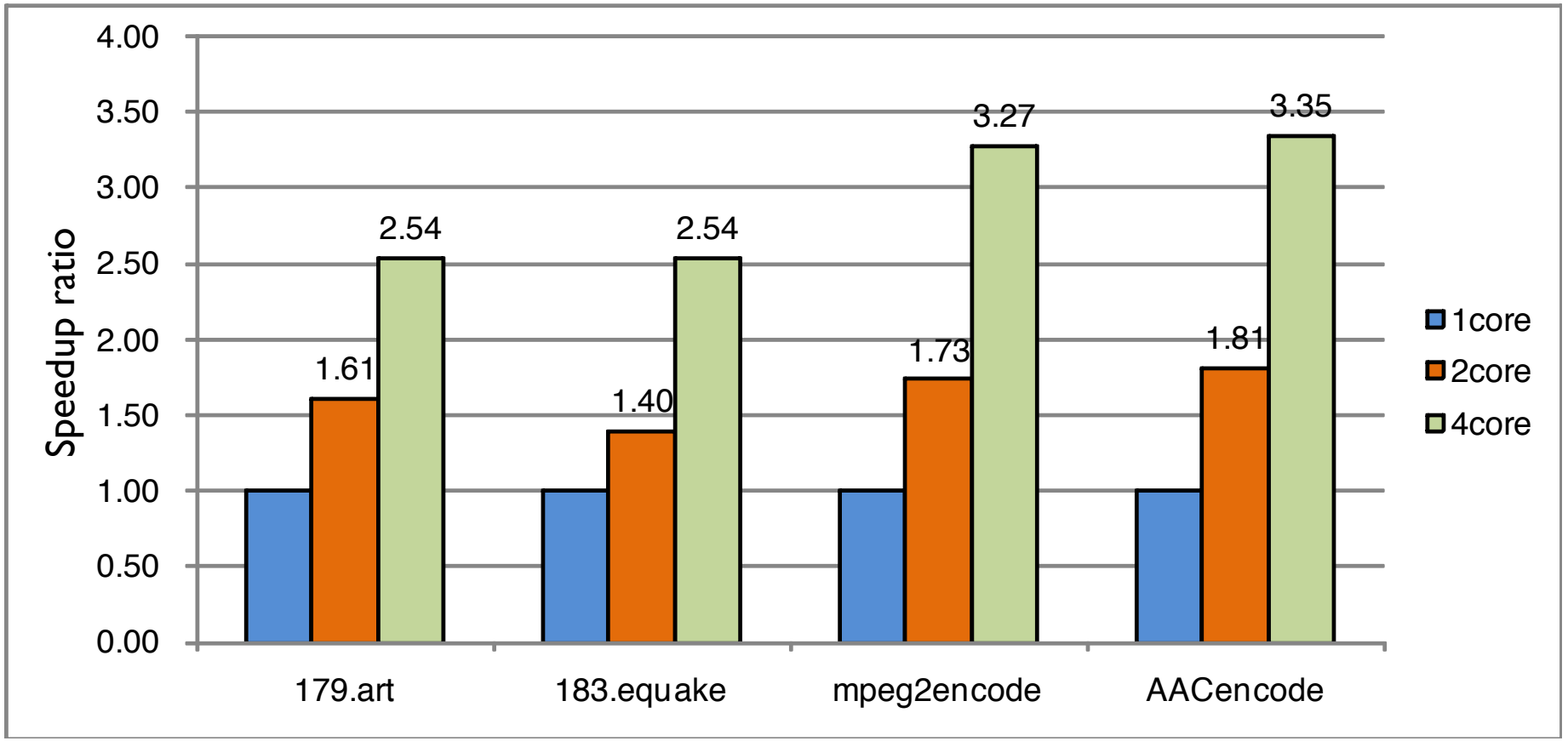
Consumer Electronics Multicore: RP2





Process Technology	90nm, 8-layer, triple-Vth, CMOS
Chip Size	104.8mm^2 (10.61mm x 9.88mm)
CPU Core Size	6.6mm^2 (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)

Scalability Evaluation on RP2

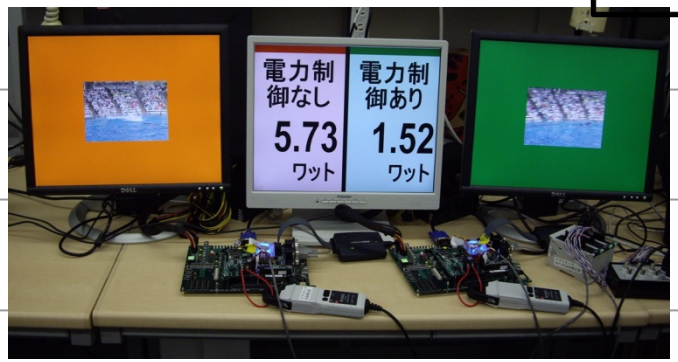


2.9 times speedup on average

Low-Power Optimization and OSCAR API on MPEG2 decoder

Without Power Control (Voltage: 1.4V)

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

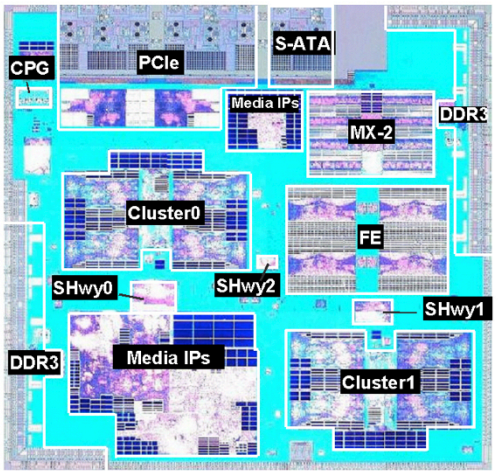


Avg. Power 5.41 [W]

76.0% Power Reduction

Avg. Power 1.30 [W]

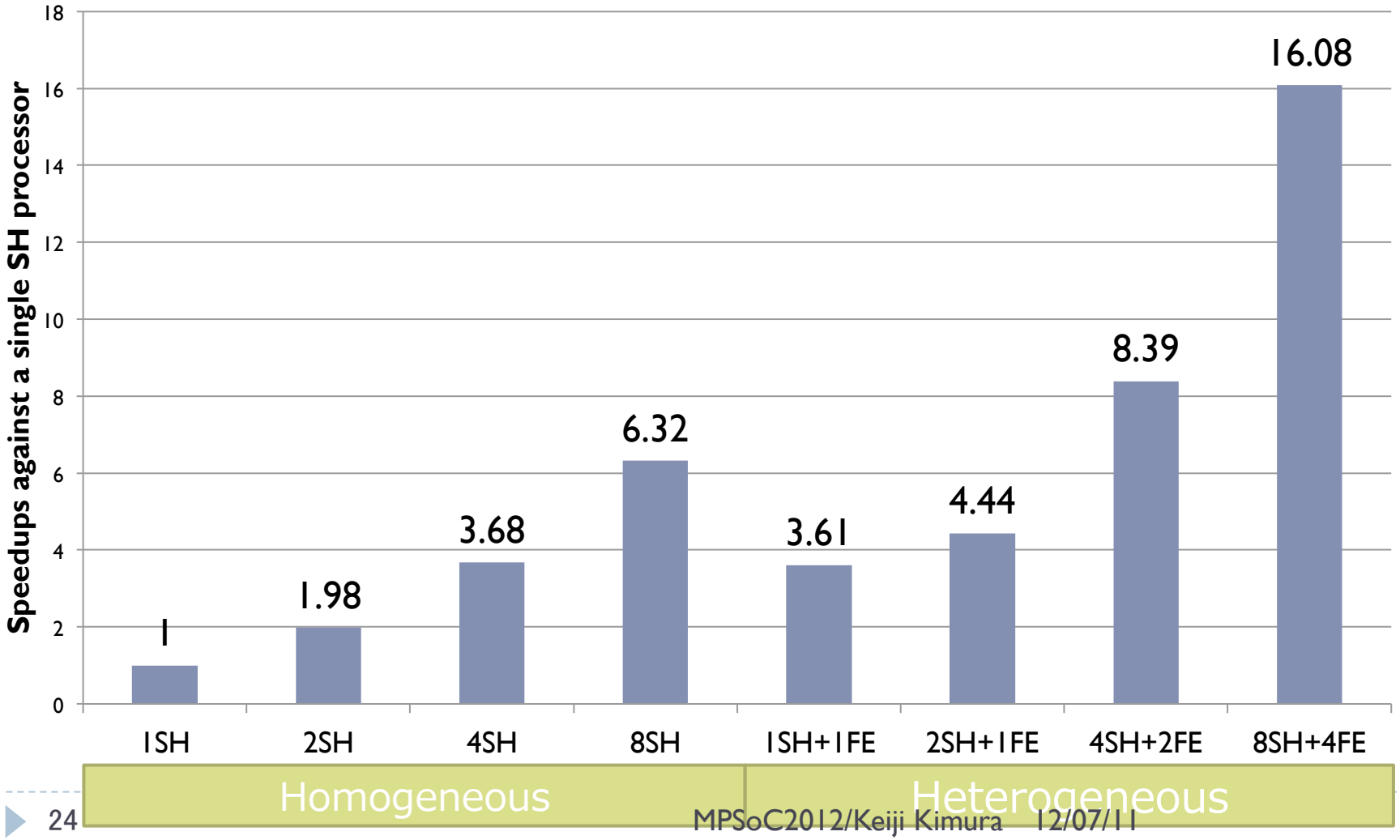
Low-power consumer electronics heterogeneous multicore “RP-X”



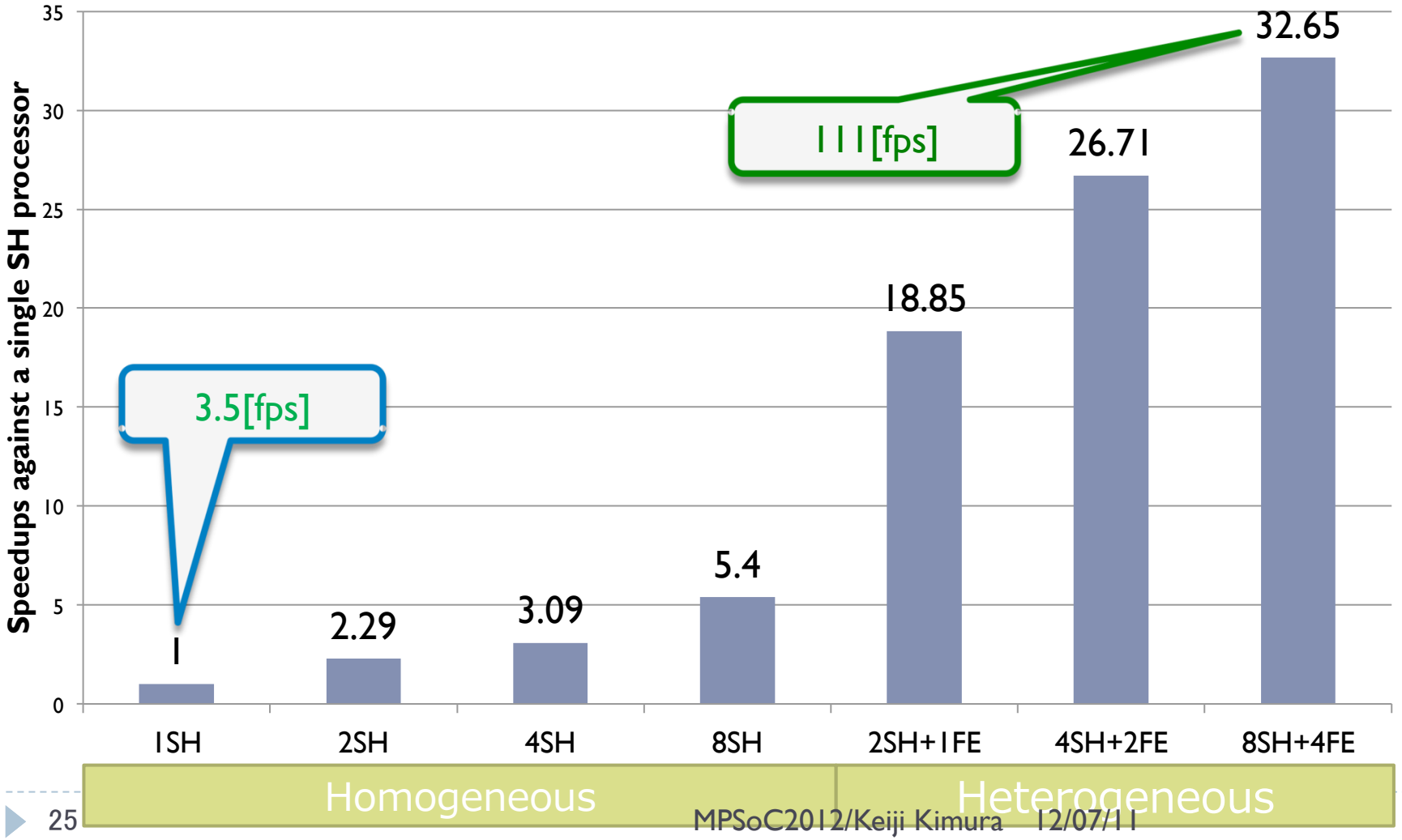
Y.Yuyama, et al., "A 45nm 37.3GOPS/W Heterogeneous Multi-Core SoC”, ISSCC2010

developed by Renesas, Hitachi, Titech, and Waseda

Scalability of AAC



Scalability of Optical Flow



Low power optimization on Optical Flow

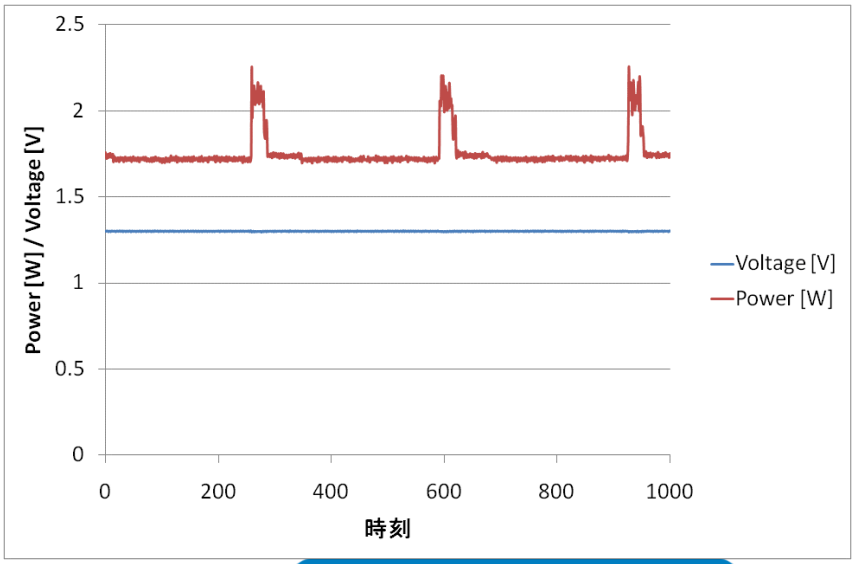
Without Power Reduction

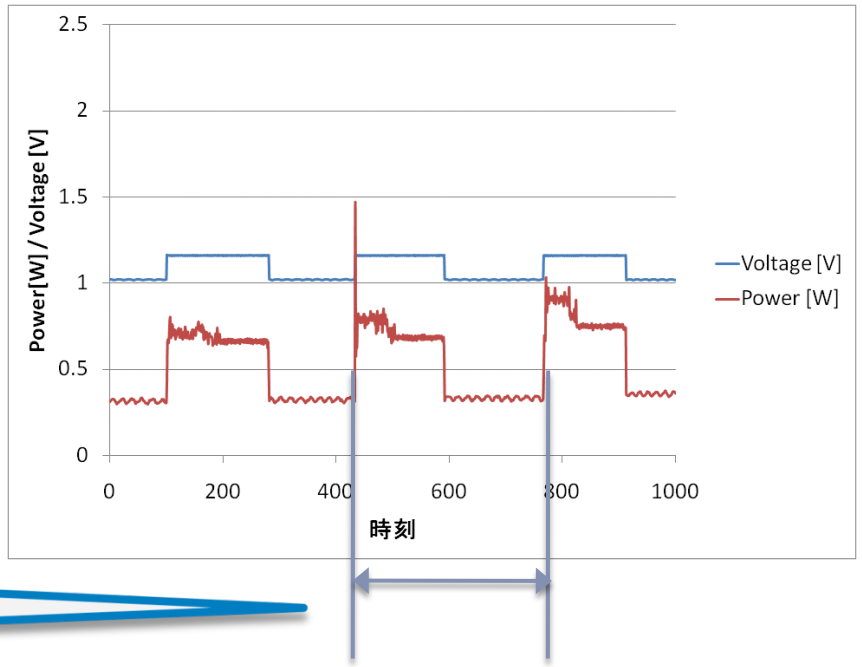
With Power Reduction by OSCAR Compiler

70% of power reduction

Average:1.76[W]

Average:0.54[W]





1cycle : 33[ms]
→30[fps]

Conclusions

- OSCAR API v2.0
  - works with OSCAR compiler
  - supporting various kinds of memory architecture
    - Local memory, Distributed shared memory, on-chip shared memory, coherent cache, and non-coherent cache
  - supporting frequency and voltage control, per-core clock-off and per-core shutdown
  - supporting accelerators
- OSCAR API standard translator
  - Portable translator for various platforms
- The combination of OSCAR compiler, OSCAR API and the standard translator give us scalable performance improvement and low-power over various multicores.
- OSCAR API specification is available from our web site.
  - http://www.kasahara.cs.waseda.ac.jp/