Multi-platform Automatic Parallelization and Power Reduction by OSCAR Compiler

Hironori Kasahara

Professor, Dept. of Computer Science & Engineering

Director, Advanced Multicore Processor Research Institute

Waseda University, Tokyo, Japan

IEEE Computer Society Board of Governors

IEEE Computer Society Multicore STC Chair

URL: http://www.kasahara.cs.waseda.ac.jp/

OSCAR Parallelizing Compiler

To improve effective performance, cost-performance and software productivity and reduce power

Multigrain Parallelization

coarse-grain parallelism among loops and subroutines, near fine grain parallelism among statements in addition to loop parallelism

Data Localization

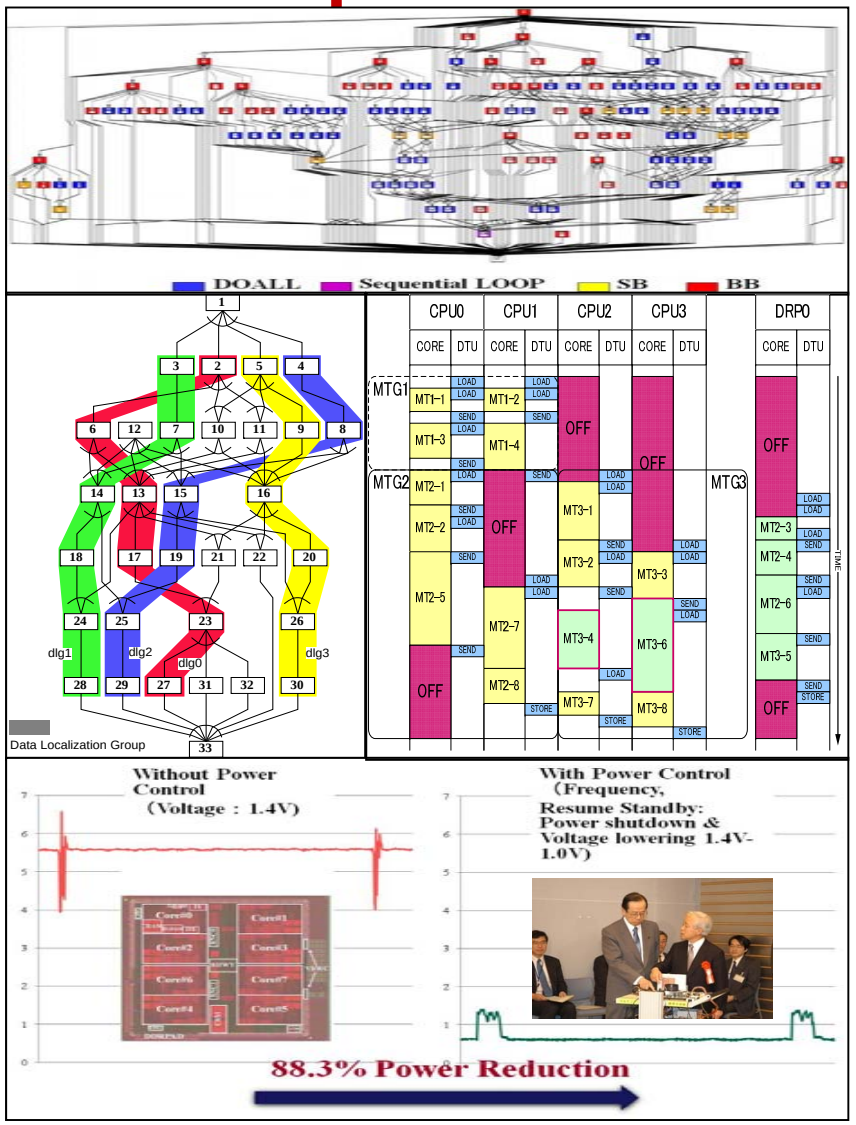
Automatic data management for distributed shared memory, cache and local memory

Data Transfer Overlapping

Data transfer overlapping using Data Transfer Controllers (DMAs)

Power Reduction

Reduction of consumed power by compiler control DVFS and Power gating with hardware supports.



Multicore Program Development Using OSCAR API V2.0

Sequential Application Program in Fortran or C
(Consumer Electronics, Automobiles, Medical, Scientific computation, etc.)

Homogeneous

Hetero

Manual parallelization / power reduction

Accelerator Compiler/ User
Add "hint" directives before a loop or a function to specify it is executable by the accelerator with how many clocks

Waseda OSCAR Parallelizing Compiler
- Coarse grain task parallelization
- Data Localization
- DMAC data transfer
- Power reduction using DVFS, Clock/ Power gating

OSCAR API for Homogeneous and/or Heterogeneous Multicores and manycores
Directives for thread generation, memory, data transfer using DMA, power managements

Parallelized API F or C program

Proc0
Code with directives
Thread 0

Proc1
Code with directives
Thread 1

Accelerator 1
Code

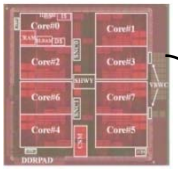
Accelerator 2
Code

Generation of parallel machine codes using sequential compilers

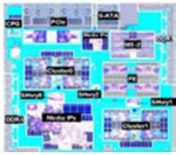
Low Power Homogeneous Multicore Code Generation
API Analyzer | Existing sequential compiler

Low Power Heterogeneous Multicore Code Generation
API Analyzer (Available from Waseda) | Existing sequential compiler

Server Code Generation
OpenMP Compiler



Homegeneous Multicore s from Vendor A (SMP servers)



Heterogeneous Multicores from Vendor B

Shred memory servers

Executable on various multicores

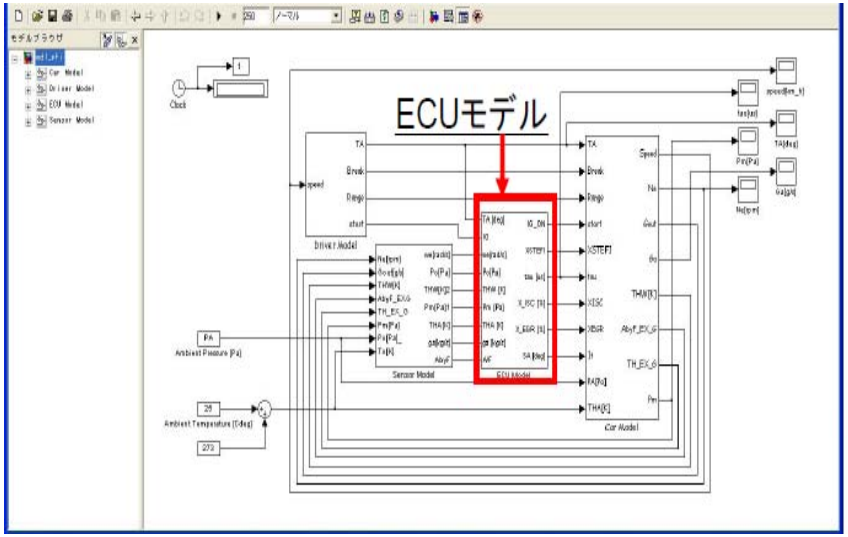
Hitachi, Renesas, NEC, Fujitsu, Toshiba, Denso, Olympus, Mitsubishi, Esol, Cats, Gaio, 3 univ.

OSCAR: Optimally Scheduled Advanced Multiprocessor
API : Application Program Interface

Model Base Designed Engine Control on V850 Multicore with Denso

Though so far parallel processing of the engine control on multicore has been very difficult, Denso and Waseda succeeded 1.95 times speedup on 2core V850 multicore processor.

Hard real-time automobile engine control by multicore



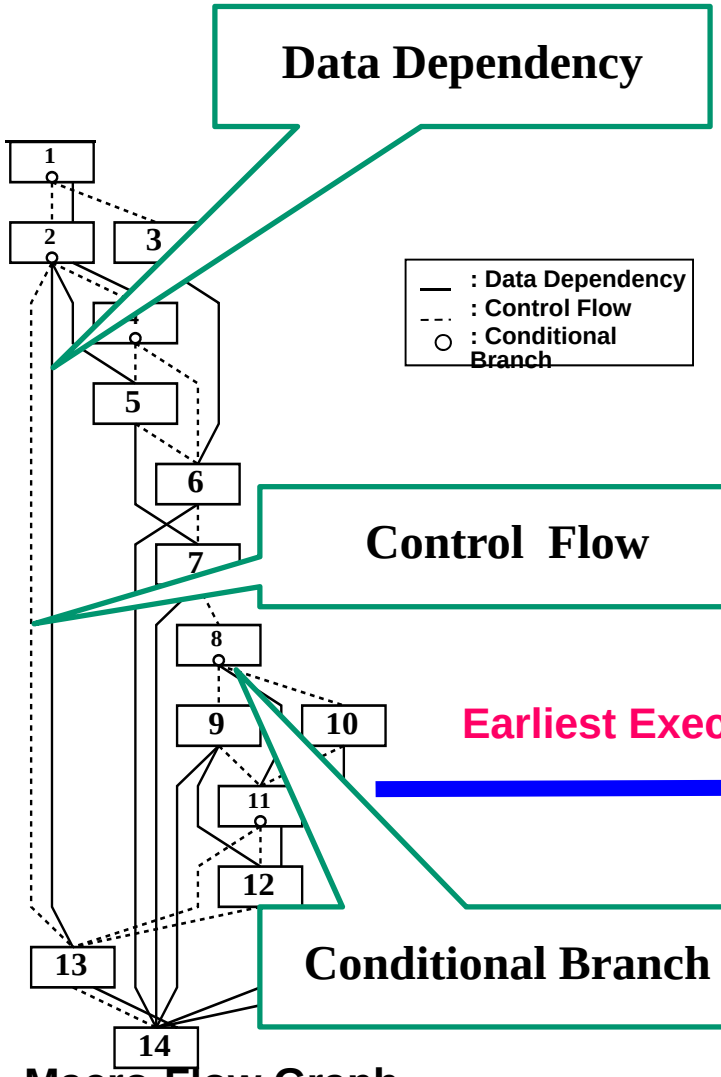
Parallelizing Handwritten Engine Control Programs on Multi-core processors

- Current automotive crankshaft program
  - Developed by TOYOTA Motor Corp
  - About 300,000 Lines
  - Difficulty of parallel processing
    - Too fine granularity
    - Many conditional branches and small basic blocks, but **no parallelizable loops**
      - Minimizing run-time overhead and improvement of parallelism are necessary
        - ➢ Current product compilers can not parallelize
        - ➢ Current accelerators are not applicable

- Automatic parallelization of a crankshaft program using multi-grain parallelization in OSCAR Compiler
  - ➢ Performance improvement and efficient multi-threaded programming development

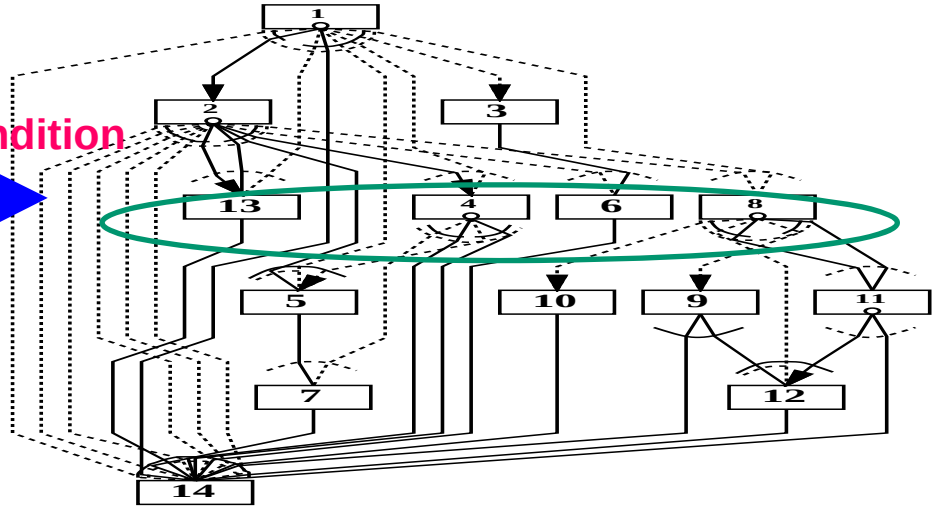
Analysis of Coarse Grain Parallelism by OSCAR Compiler

- **Decomposes a program into coarse grain tasks, or macro tasks(MTs)**
  1. **BB (Basic Block)**
  2. **RB (Repetition Block, or loop)**
  3. **SB (Subroutine Block, or function)**
- **Generate MFG(Macro Flow Graph)**
  - **Control flow and data dependencies**
- **Generates MTG(Macro Task Graph)**
  - **Coarse grain parallelism**



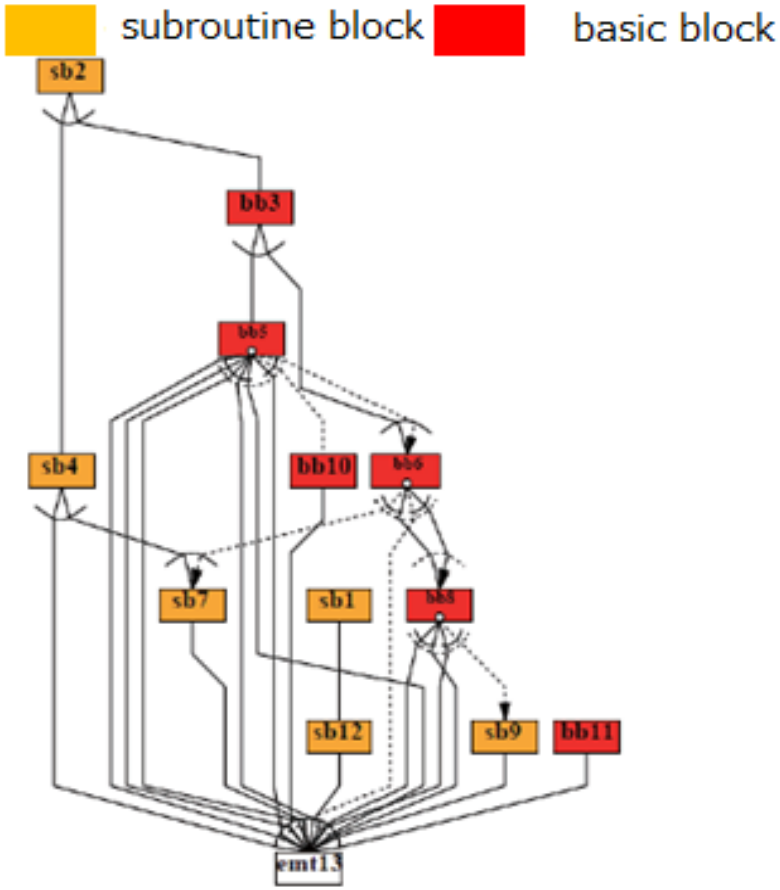
Macro-Flow Graph

Earliest Executable Condition



Macro-Task Graph

Coarse Grain Task Parallelization of Hand-written Engine Control Program

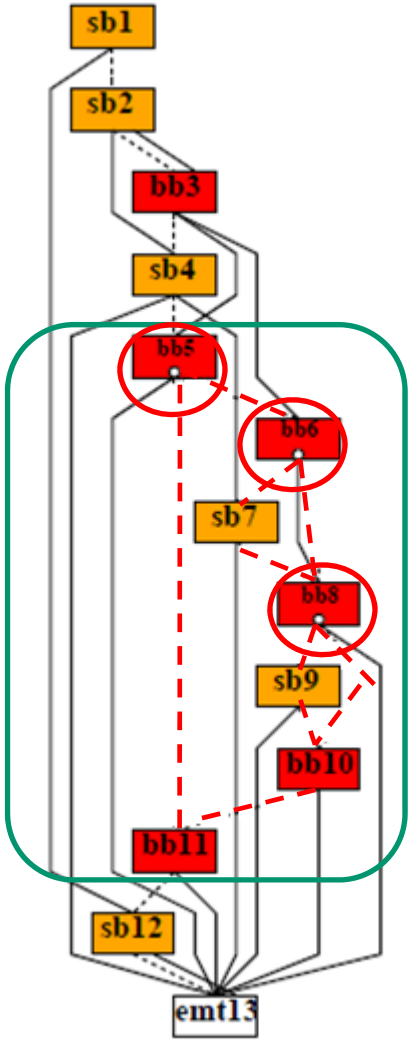


MTG of crankshaft programs

- **Loop parallelization**
  - **No parallelizable loops in engine control codes**
- **Fine grain parallelization**
  - **Each BBs are very low cost**
    - **less than 100 clock cycles**
  - **Branches prevent compilers**
- **Coarse grain parallelization**
  - **Utilize parallelism between SBs and BBs**

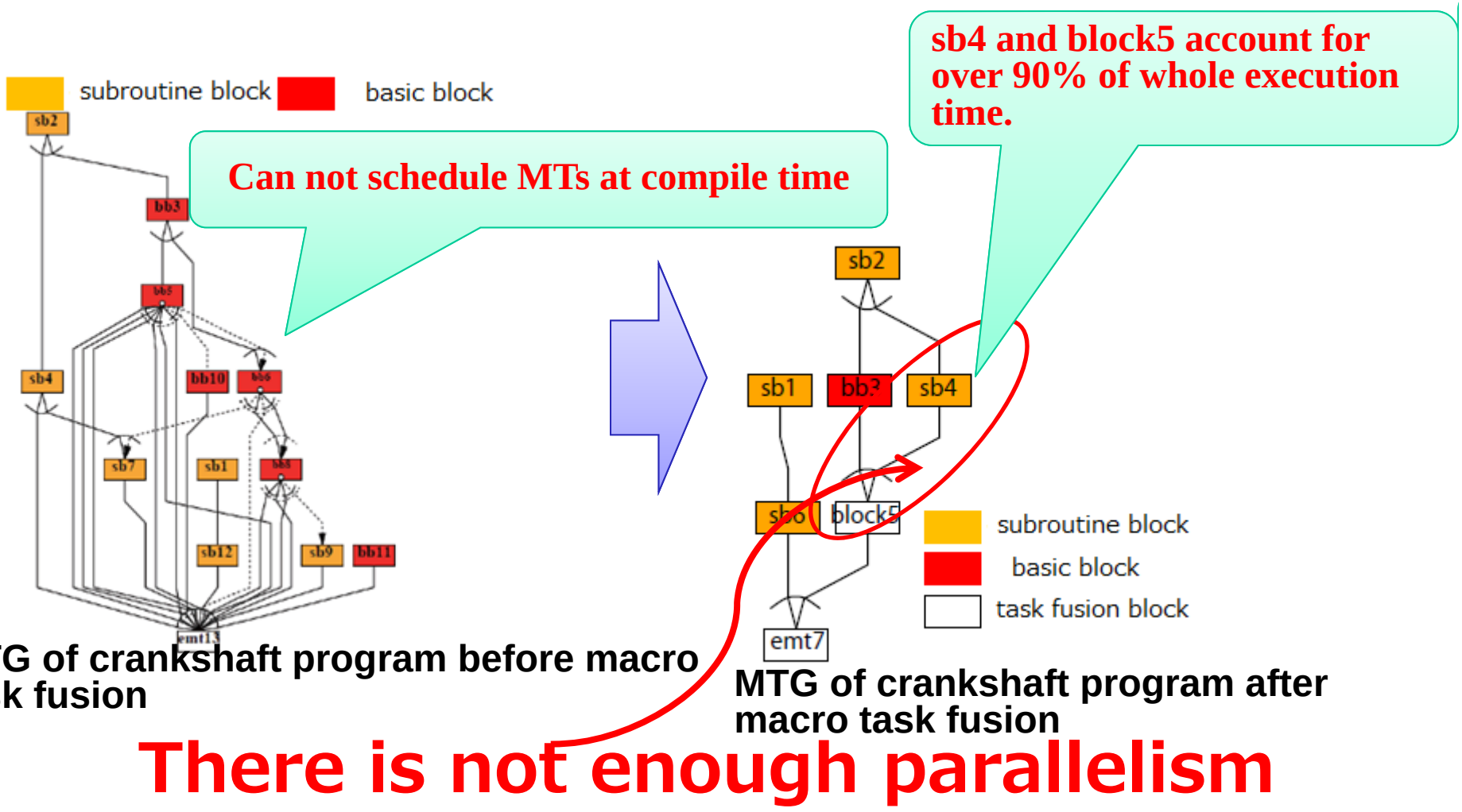
Static Task Scheduling

- **Dynamic task scheduling**
  - Prevent from traceability
  - Add run-time overhead
- **Static task scheduling**
  - Guarantee Real-time constraints
  - Ensure traceability
  - Minimize run-time overhead
- **Cannot assign BBs having braches statically**
  - Static task scheduling can be applied if the MTG has only data dependency
  - The compiler cannot see if the branch is taken or not at compile time.
- **Fuse tasks by hiding conditional branches in MFG to avoid dynamic task scheduling**
  - **Macro Task Fusion**



MFG of sample program

Analysis of A Crankshaft Program Using Macro Task Fusion

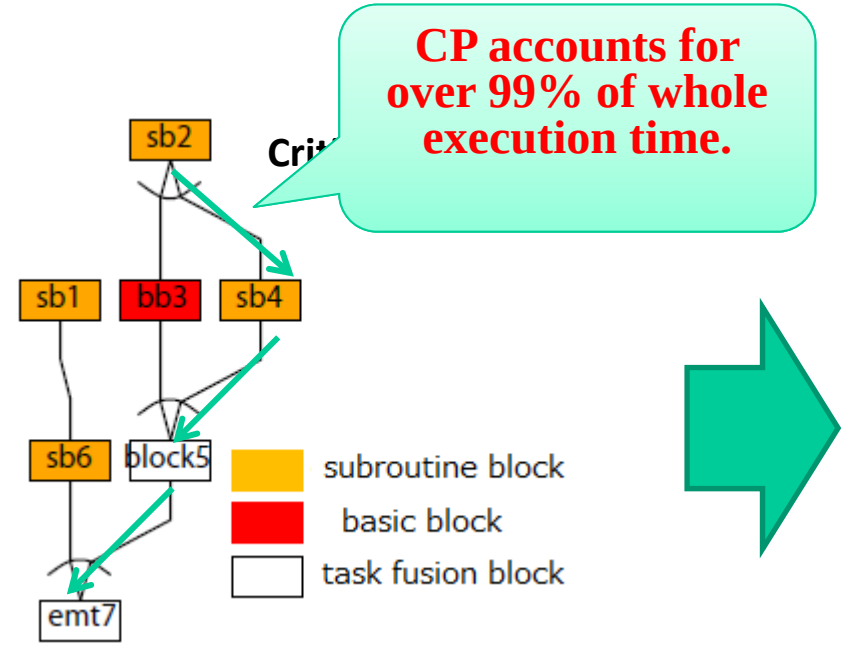


MTG of crankshaft program before macro task fusion

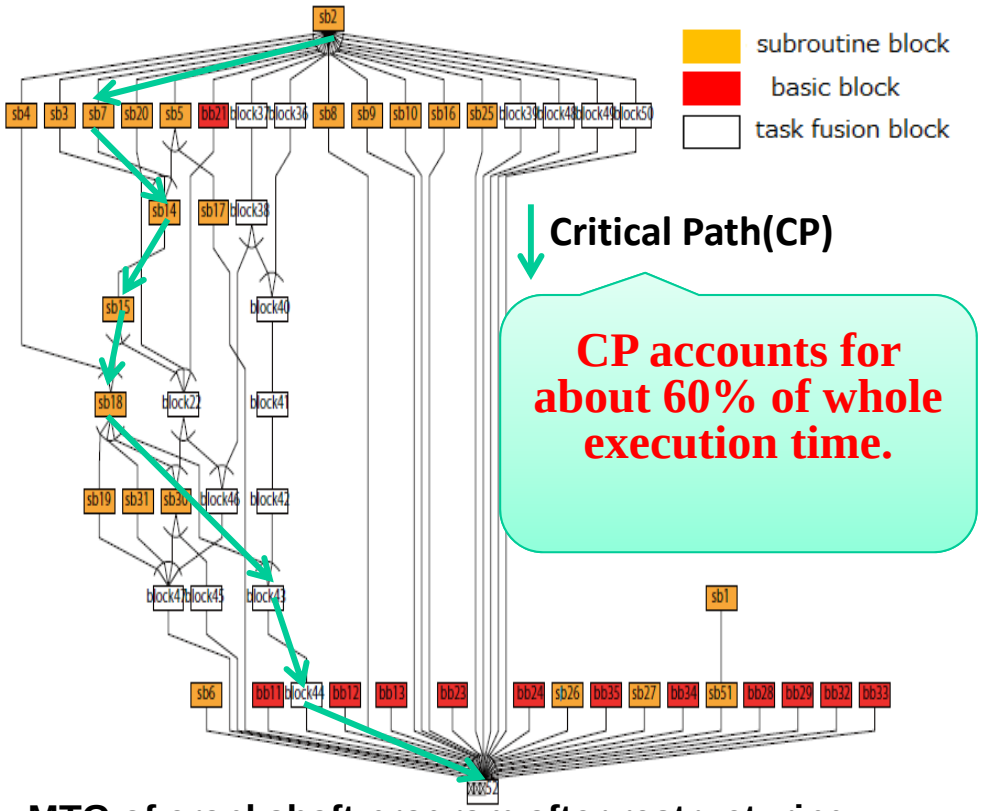
MTG of crankshaft program after macro task fusion

There is not enough parallelism

MTG of Crankshaft Program Using Inline Expansion and Duplicating If-statements



MTG of crankshaft program before restructuring

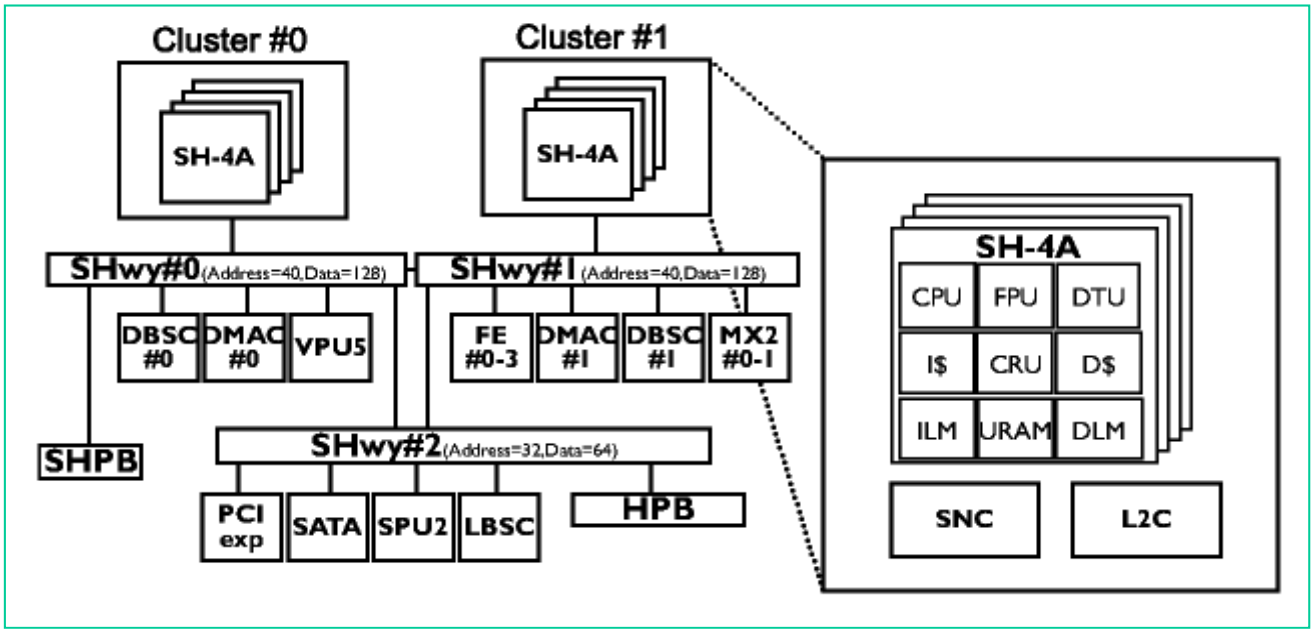


MTG of crankshaft program after restructuring

- Succeed to reduce CP
  - 99% -> 60%

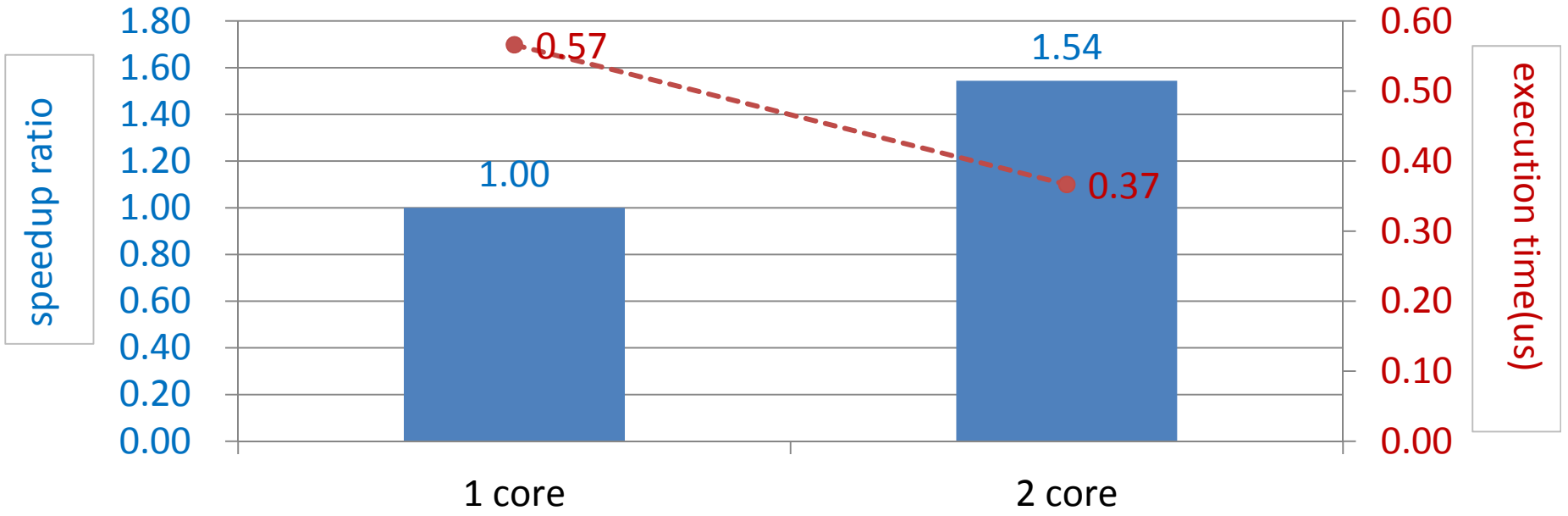
Successfully increased coarse grain parallelism

Evaluation Environment :
Embedded Multi-core Processor RPX



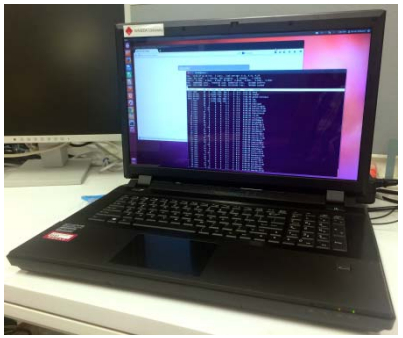
- SH-4A 648MHz * 8
  - As a first step, we use just two SH-4A cores because target dual-core processors are currently under design for next-generation automobiles

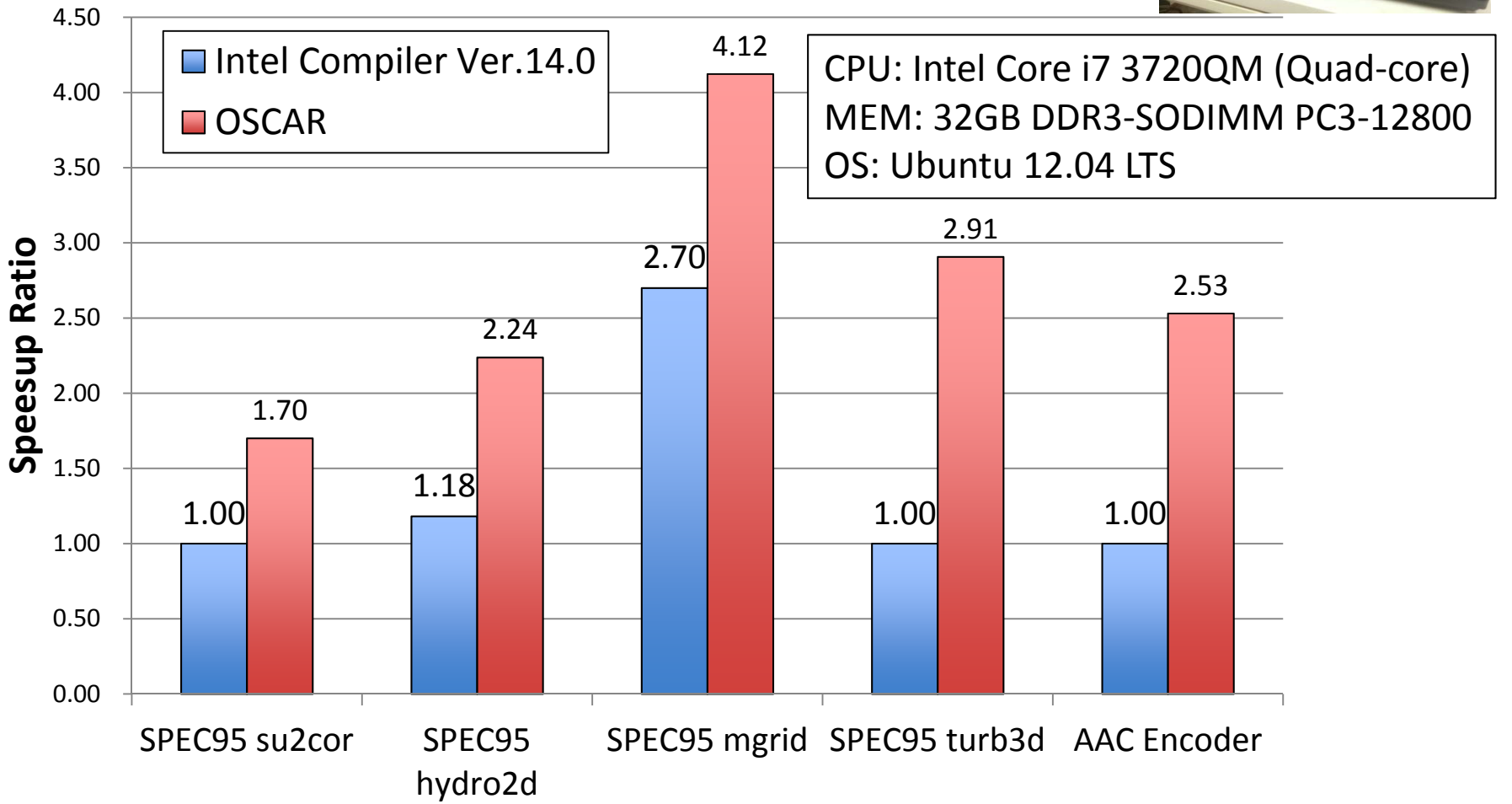
Evaluation of Crankshaft Program with Multi-core Processors



- Attain 1.54 times speedup on RPX
  - There are no loops, but only many conditional branches and small basic blocks and difficult to parallelize this program
- This result shows possibility of multi-core processor for engine control programs

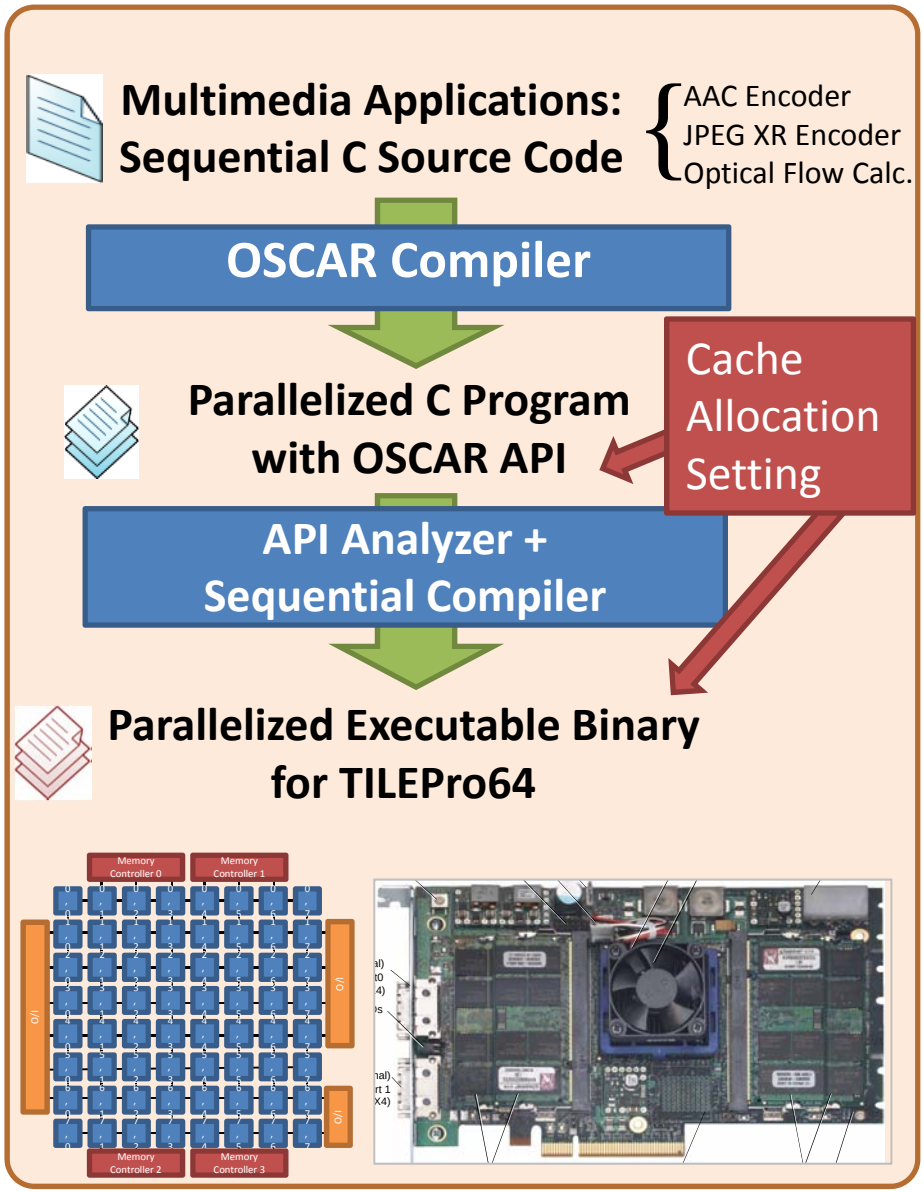
Performance of OSCAR Compiler on Intel Core i7 Notebook PC

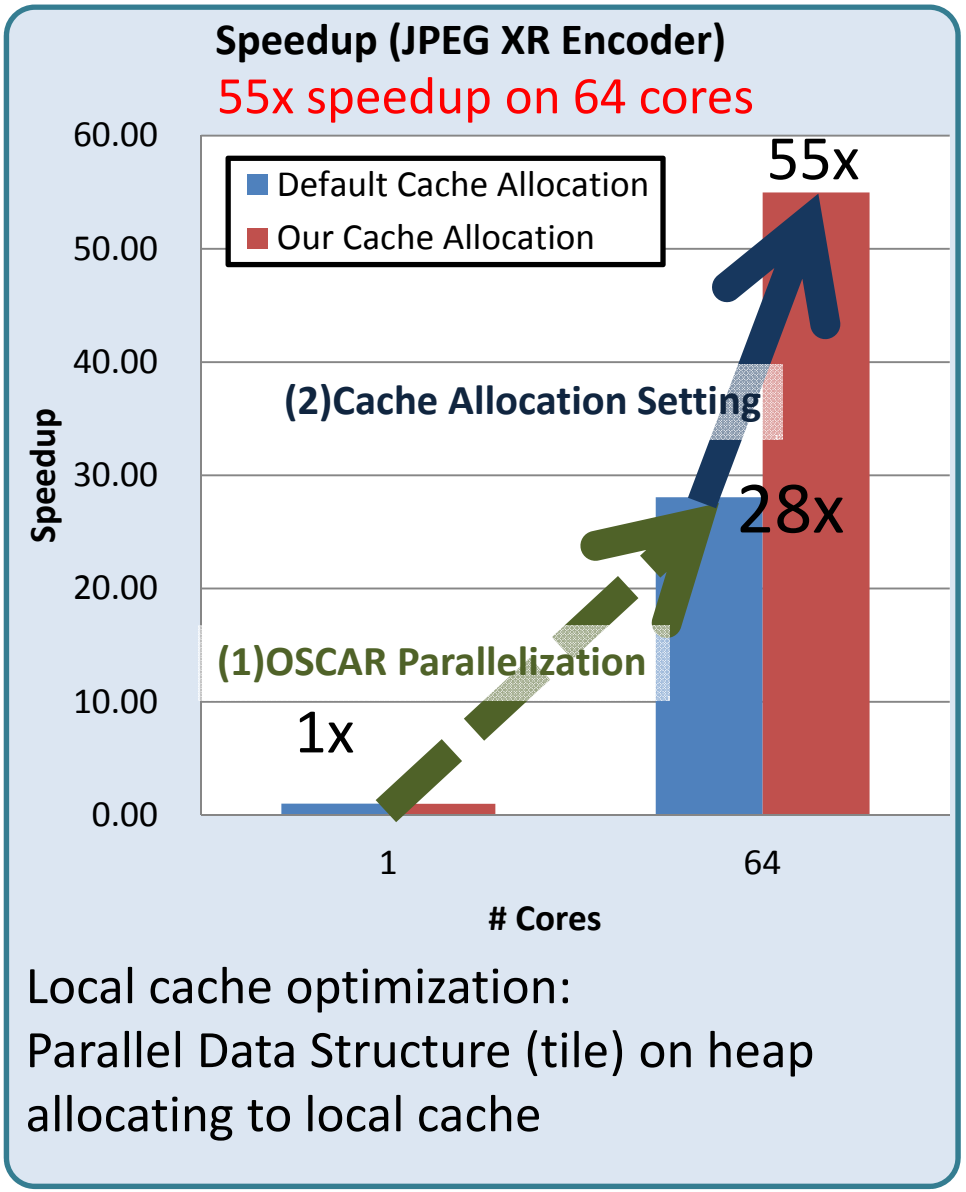




- OSCAR Compiler accelerate Intel Compiler about **2.0 times** on average

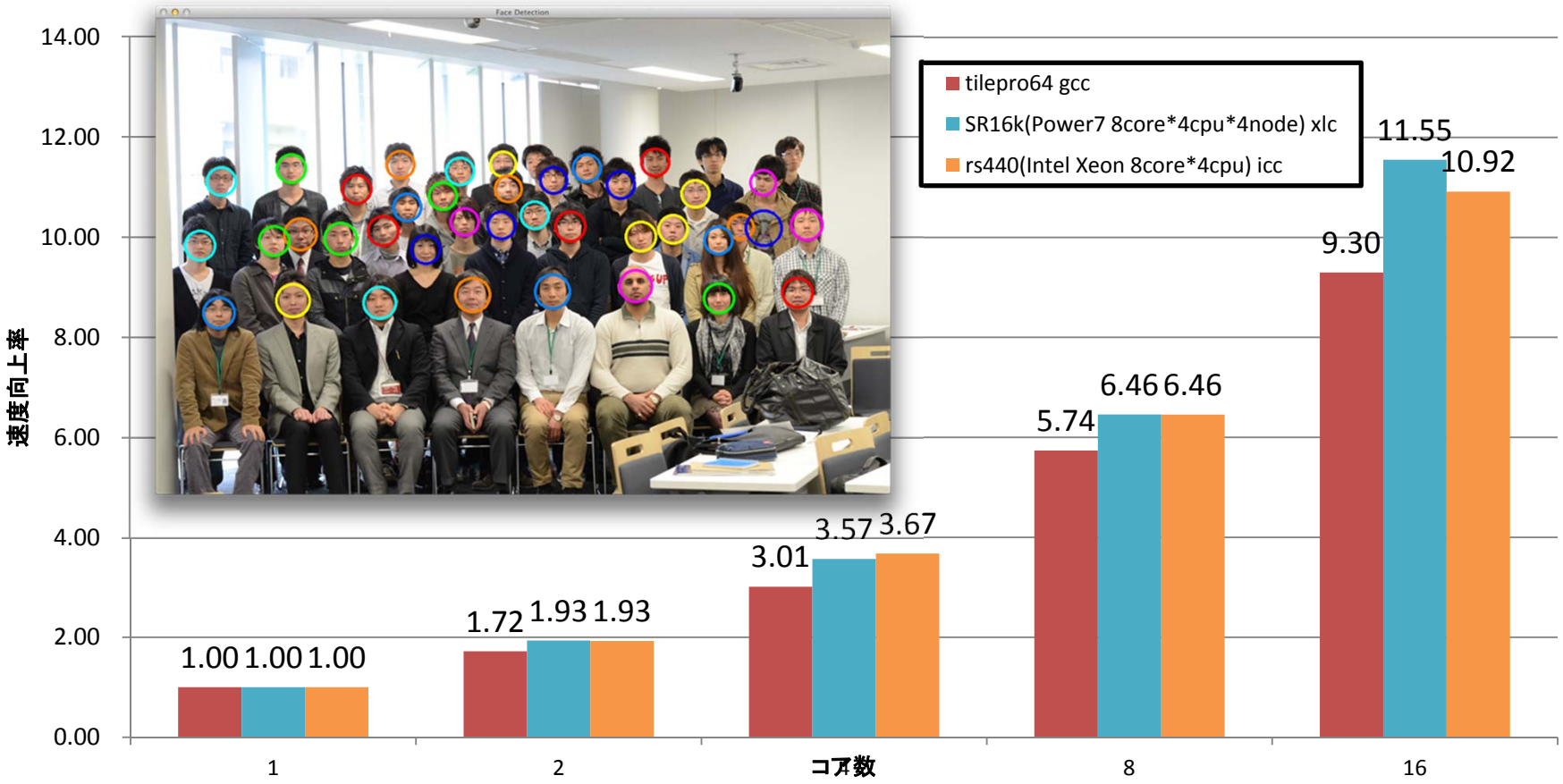
Parallel Processing of JPEG XR Encoder on TILEPro64





Local cache optimization:
Parallel Data Structure (tile) on heap allocating to local cache

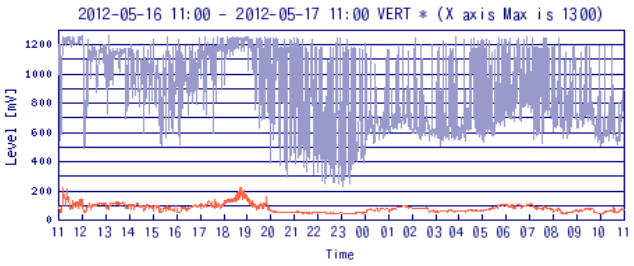
Parallel Processing of Face Detection on Manycore, Highend and PC Server

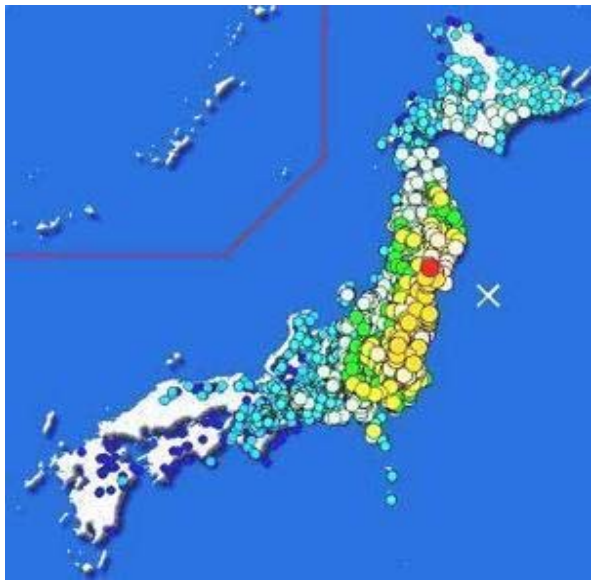


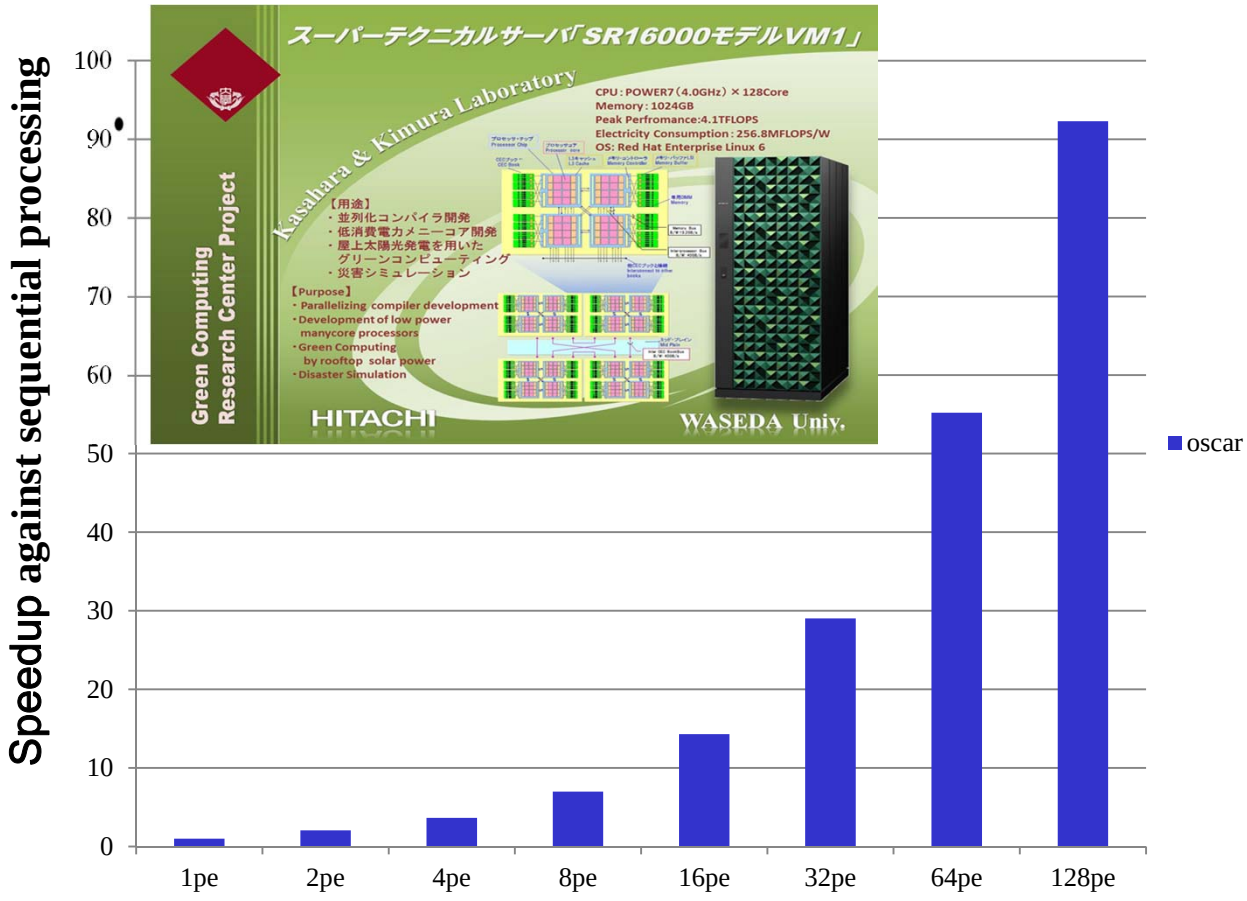
- OSCAR compiler gives us 11.55 times speedup for 16 cores against 1 core on SR16000 Power7 highend server.

92 Times Speedup against the Sequential Processing for GMS Earthquake Wave Propagation Simulation on Hitachi SR16000

（Power7 Based 128 Core Linux SMP）







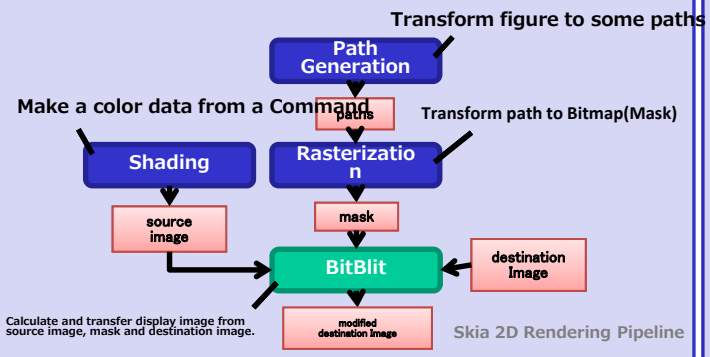
Profile-Based Automatic Parallelization and Sequential Program Tuning for Android 2D Rendering on Nexus7

Waseda University

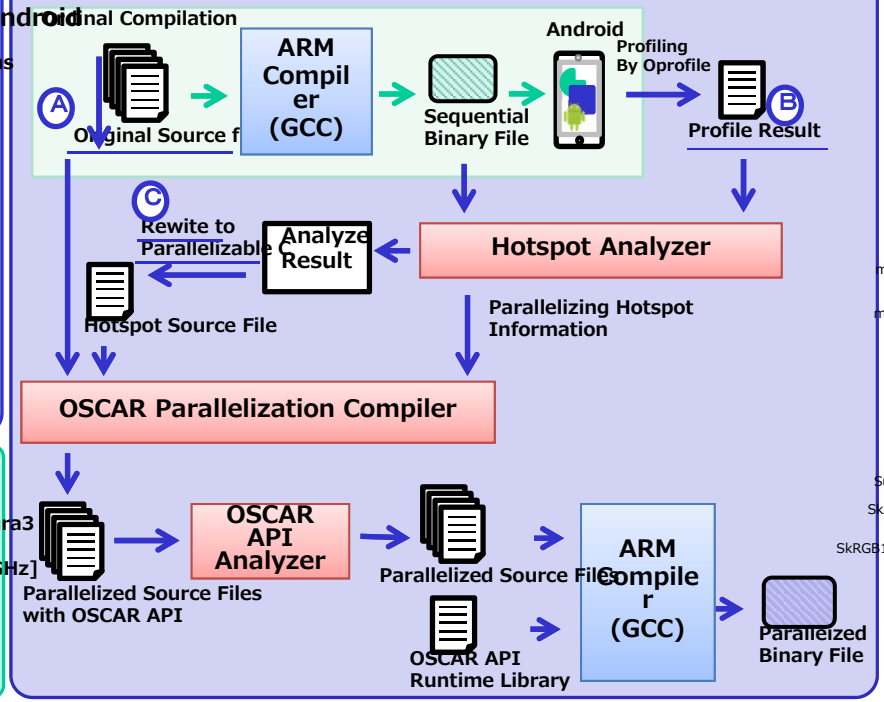
- OSCAR Compiler
- Skia
- Multicore

A Android 2D Rendering "Skia"

Standard library which performs 2D rendering on Android

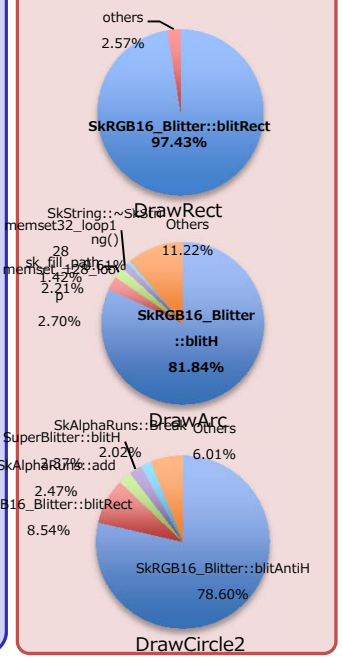


Profile-Based Automatic Parallelization



B "Skia" Profiling

Execution flow is different on each benchma
High load on the BitBlit process.



C Tuning Method for "Skia" [DrawRect]

Original Source Code

```
void SkRGB16_Blitter::blitRect(int x, int y, int width, int height) {
    SkASSERT(x + width <= fDevice.width() && y + height <= fDevice.height());
    uint16_t* SK_RESTRICT device = fDevice.getAddr16(x, y);
    unsigned    deviceRB = fDevice.rowBytes();
    SkPMColor src32 = fSrcColor32;

    while (--height >= 0) {
        blend32_16_row(src32, device, width);
        device = (uint16_t*)((char*)device + deviceRB);
    }
}
```

After Tuning

```
void SkRGB16_Blitter::blitRect(int x, int y, int width, int height) {
    SkASSERT(x + width <= fDevice.width() && y + height <= fDevice.height());
    uint16_t* SK_RESTRICT device = fDevice.getAddr16(x, y);
    unsigned    deviceRB = fDevice.rowBytes();
    SkPMColor src32 = fSrcColor32;

    SkRGB16_Blitter_blitRect_oscar(width, height, device, deviceRB, src32);
}

void SkRGB16_Blitter_blitRect_oscar(int width, int height, uint16_t *device,
                                    unsigned deviceRB, SkPMColor src32) {
    int i;
    uint16_t* deviceTMP;

    for (i = height; i > 0; i--){
        deviceTMP = (uint16_t*)((char*)device + (deviceRB * (height - i)));
        blend32_16_row(src32, deviceTMP, width);
    }
}
```

Separate C++ File

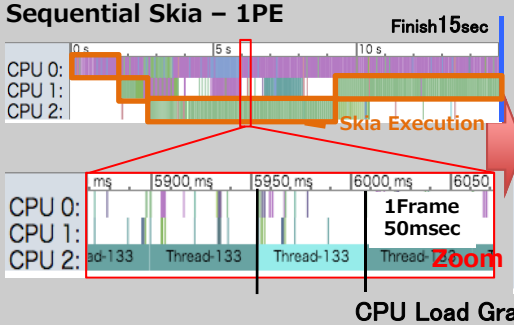
True dependency on variable deviceRB is

Google NEXUS 7

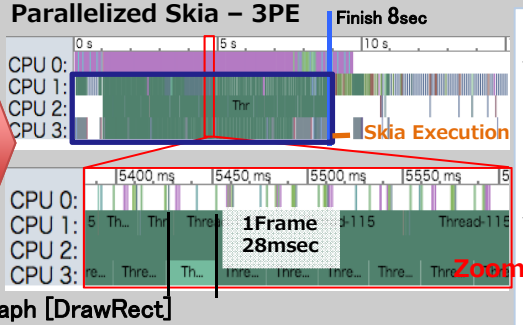
NVIDIA Tegra3 Chip
Processor : NVIDIA Tegra3
ARM Cortex A9 - 4Core
Clock Frequency : 1.2[GHz]

Evaluations

Sequential Skia – 1PE



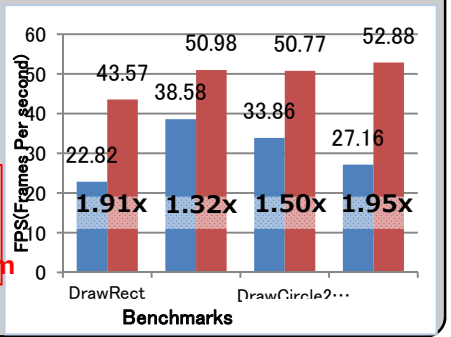
Parallelized Skia – 3PE



CPU Load Graph [DrawRect]

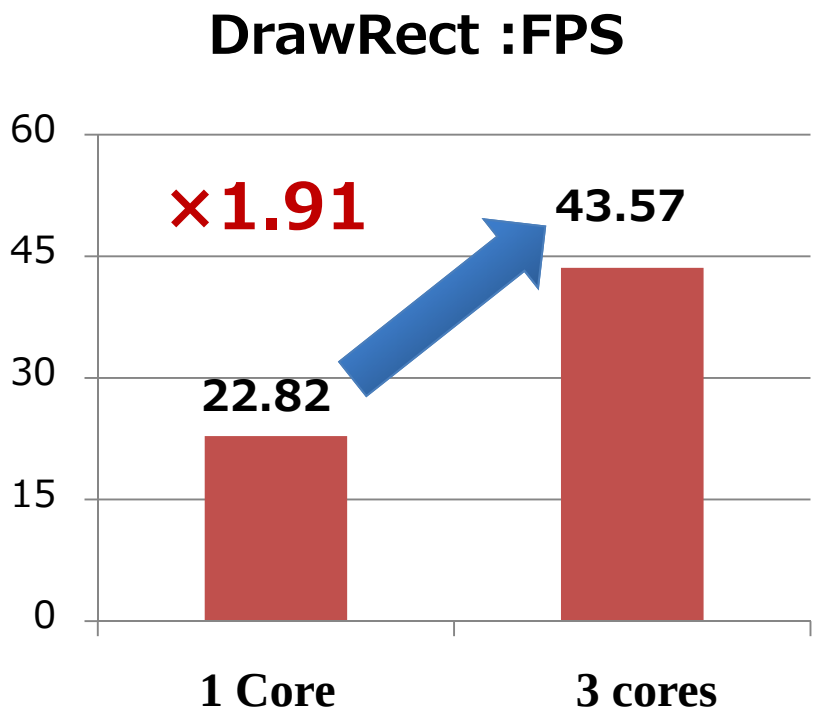
Performance

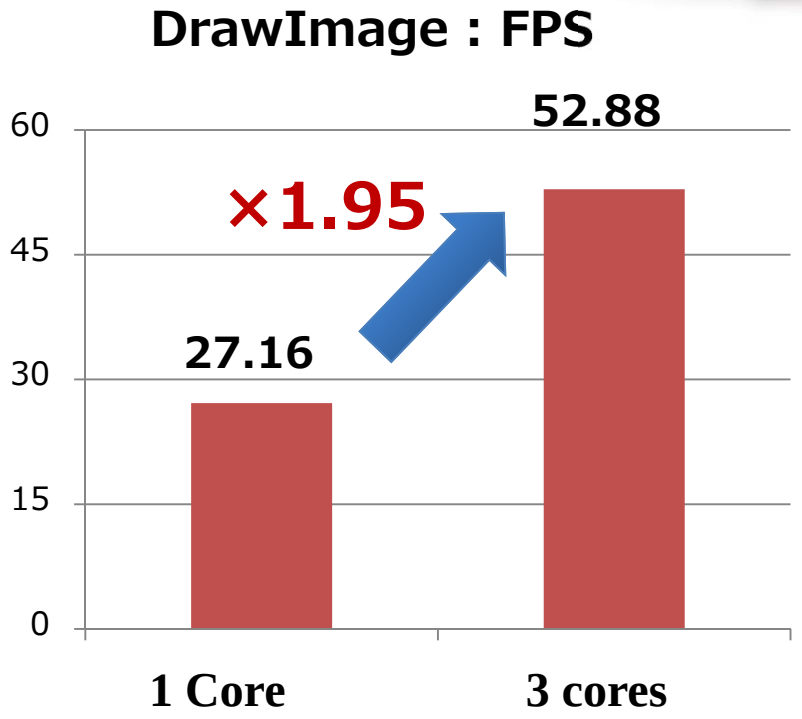
0xbenchmark 2D Canvas Test



Parallelization of 2D Rendering Engine SKIA on 3 cores of Google NEXUS7

http://www.youtube.com/channel/UCS43lNYEIkC8i_KIgFZYQBQ



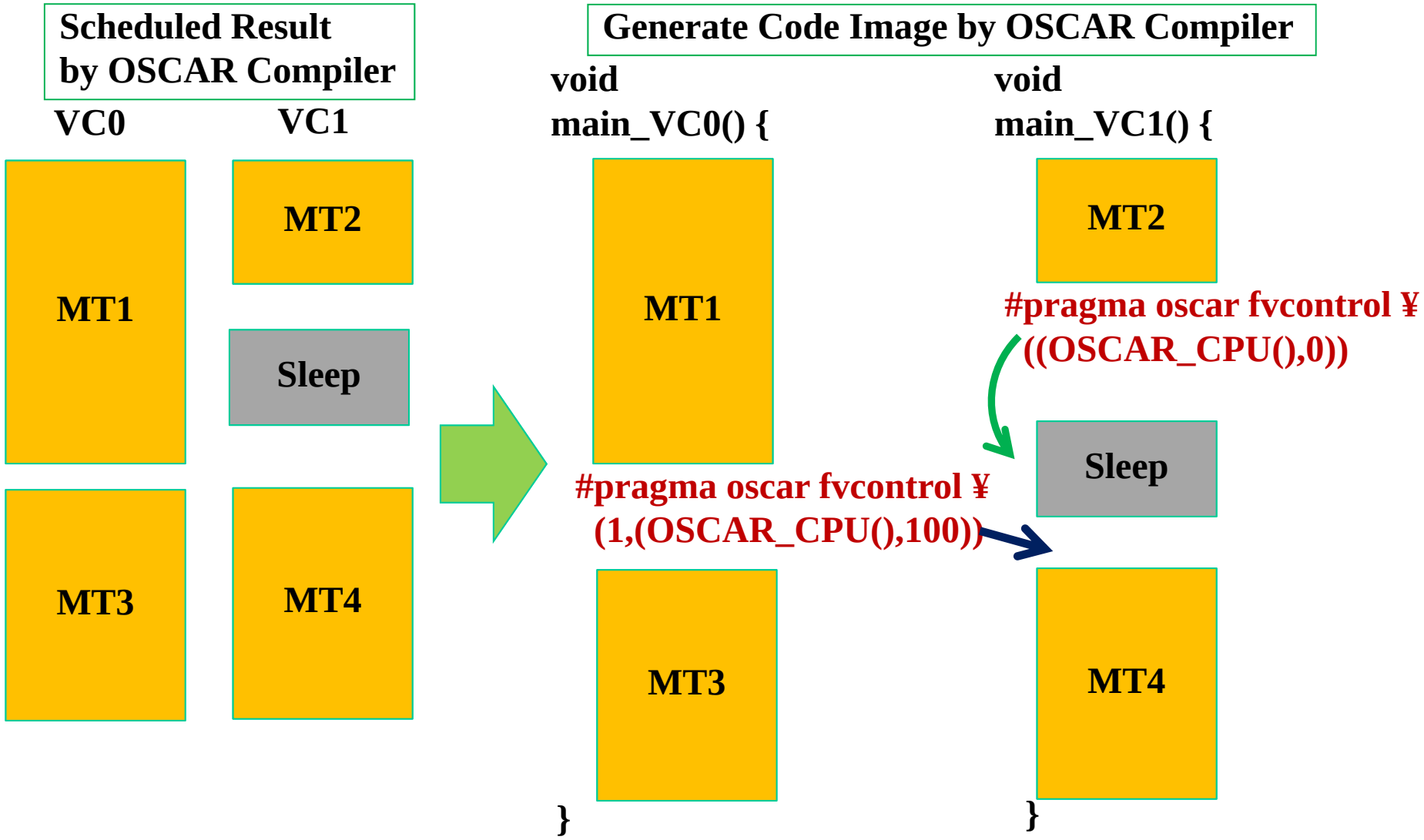


On Nexus7, 3 core parallelization gave us

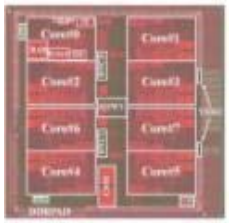
for DrawRect **1.91** speedup

for DrawImage **1.95** speedup

Low-Power Optimization with OSCAR API

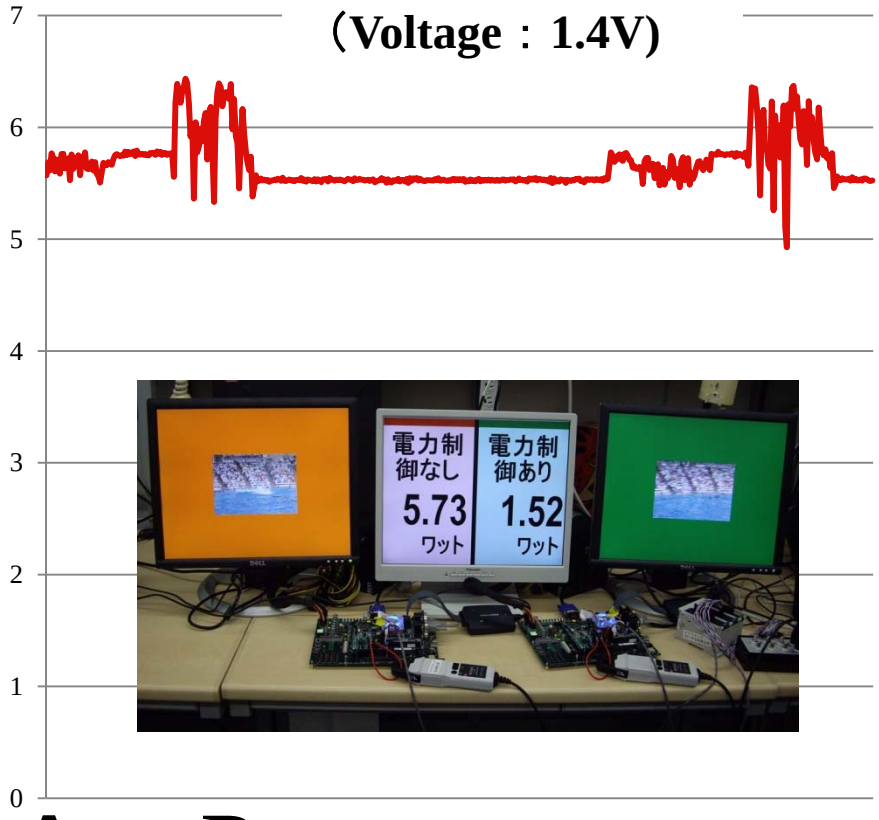


Power Reduction of MPEG2 Decoding to 1/4 on 8 Core Homogeneous Multicore RP-2 by OSCAR Parallelizing Compiler

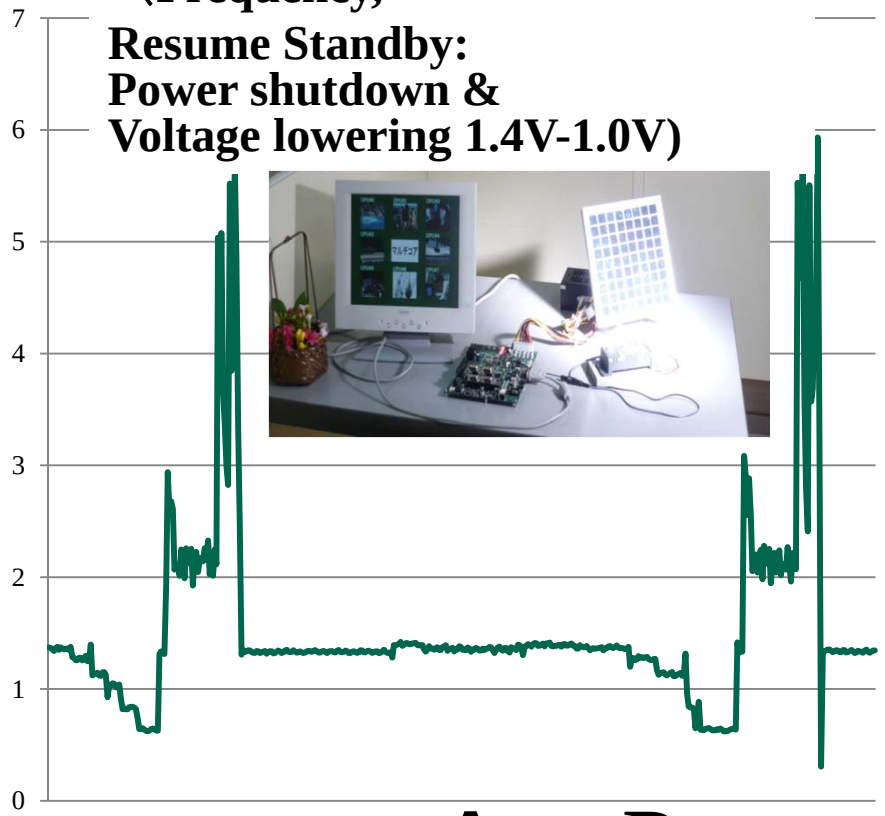


MPEG2 Decoding with 8 CPU cores

Without Power Control （Voltage：1.4V)



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



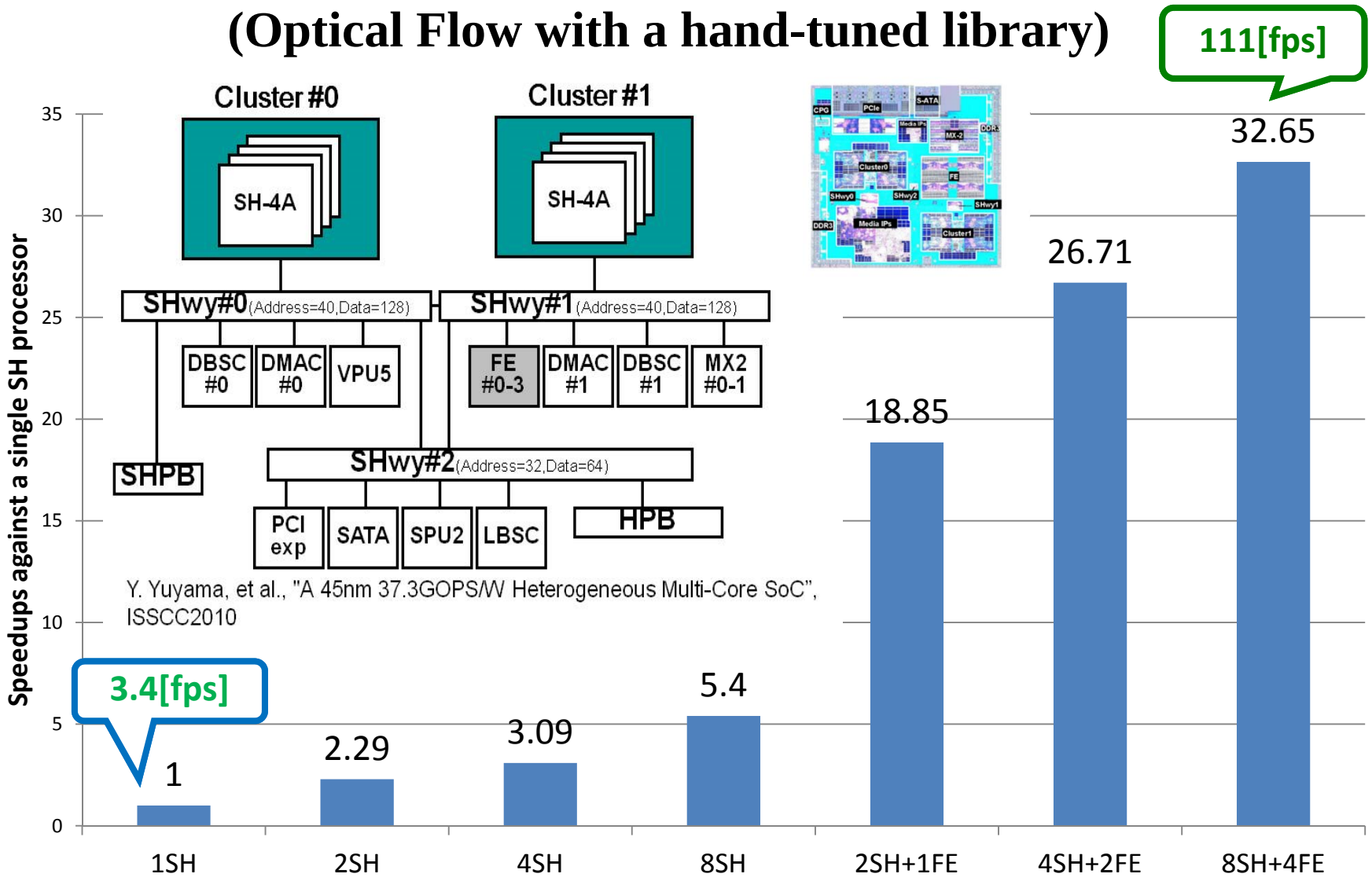
Avg. Power 5.73 [W]

73.5% Power Reduction

Avg. Power 1.52 [W]

33 Times Speedup Using OSCAR Compiler and OSCAR API on RP-X

(Optical Flow with a hand-tuned library)



Power Reduction in a real-time execution controlled by OSCAR Compiler and OSCAR API on RP-X (Optical Flow with a hand-tuned library)

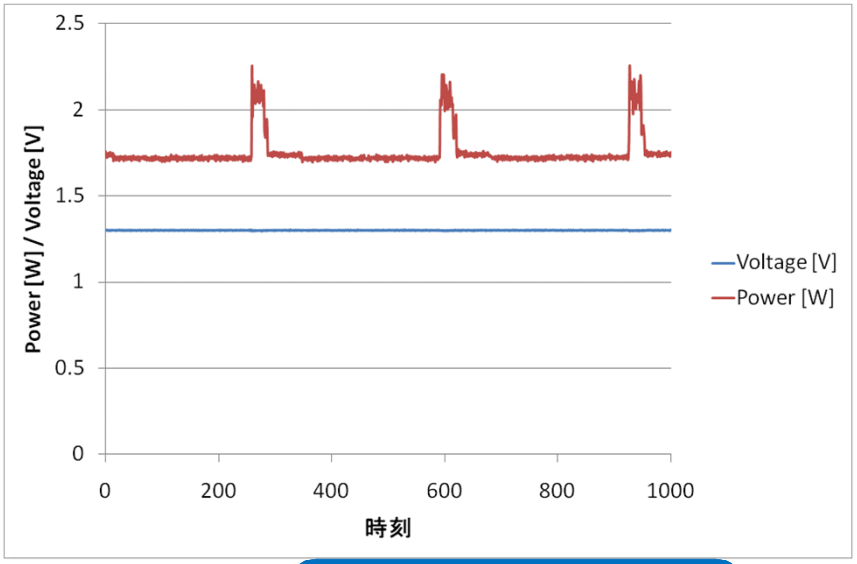
Without Power Reduction

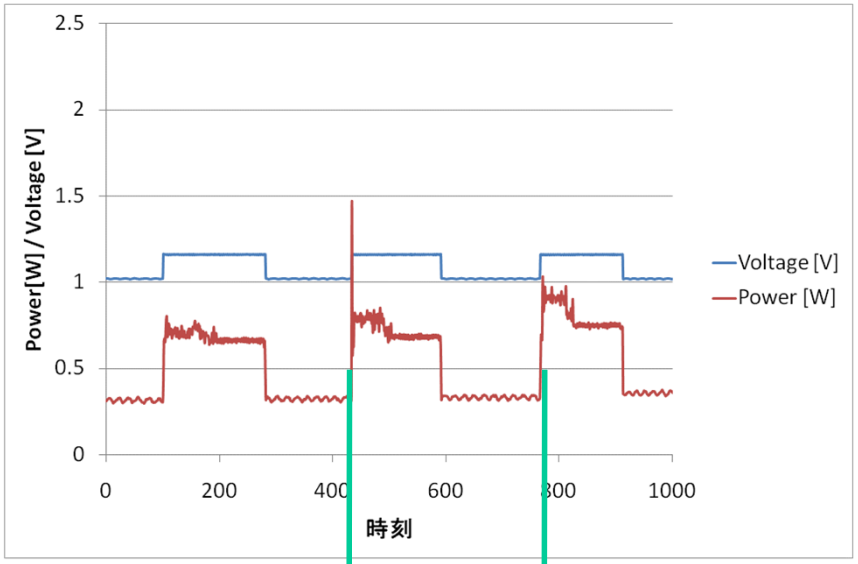
With Power Reduction by OSCAR Compiler

70% of power reduction

Average:1.76[W]

Average:0.54[W]



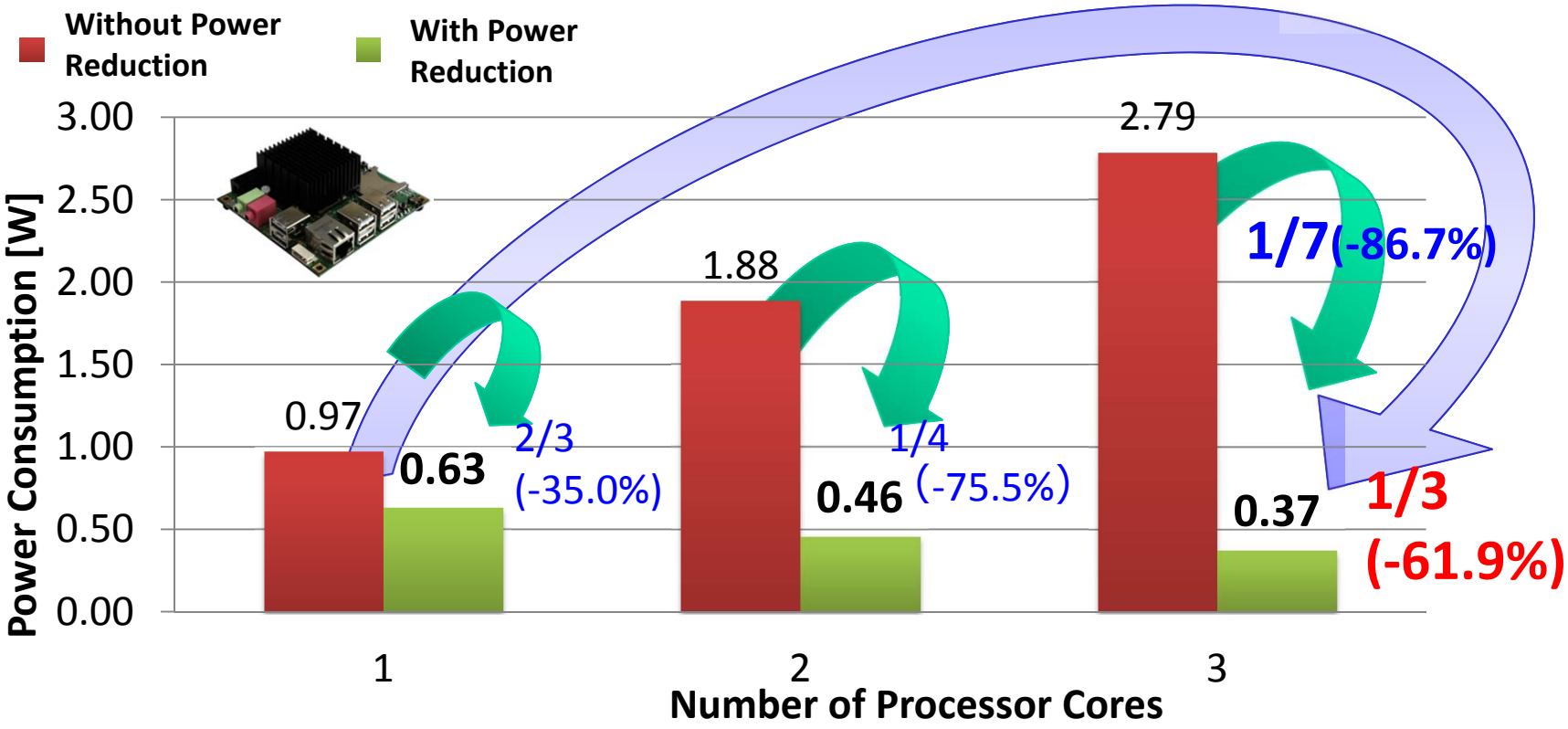


1cycle : 33[ms]
→30[fps]

Automatic Power Reduction for MPEG2 Decode on Android Multicore

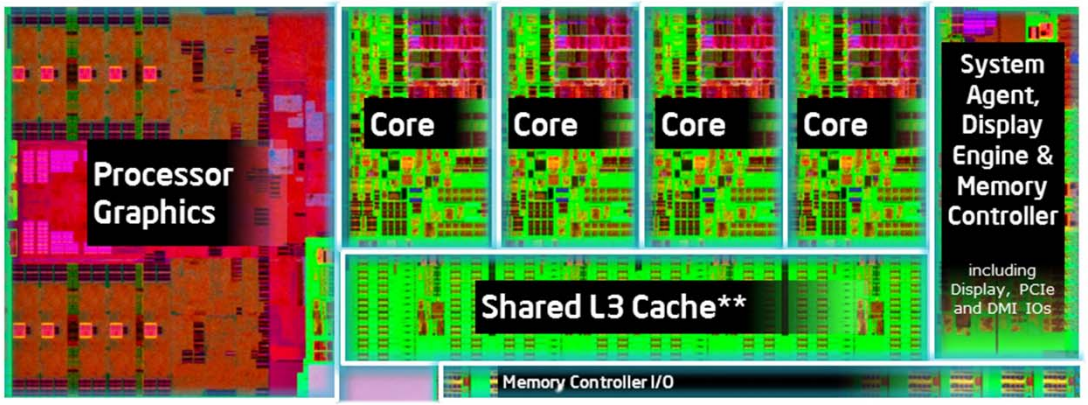
ODROID X2 ARM Cortex-A9 4 cores

http://www.youtube.com/channel/UCS43lNYEIkC8i_KIgFZYQBQ



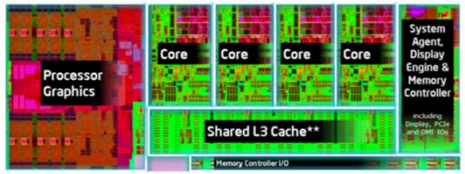
- **On 3 cores, Automatic Power Reduction control successfully reduced power to 1/7 against without Power Reduction control.**
- **3 cores with the compiler power reduction control reduced power to 1/3 against ordinary 1 core execution.**

Automatic Power Reduction on 4 core Intel Haswell

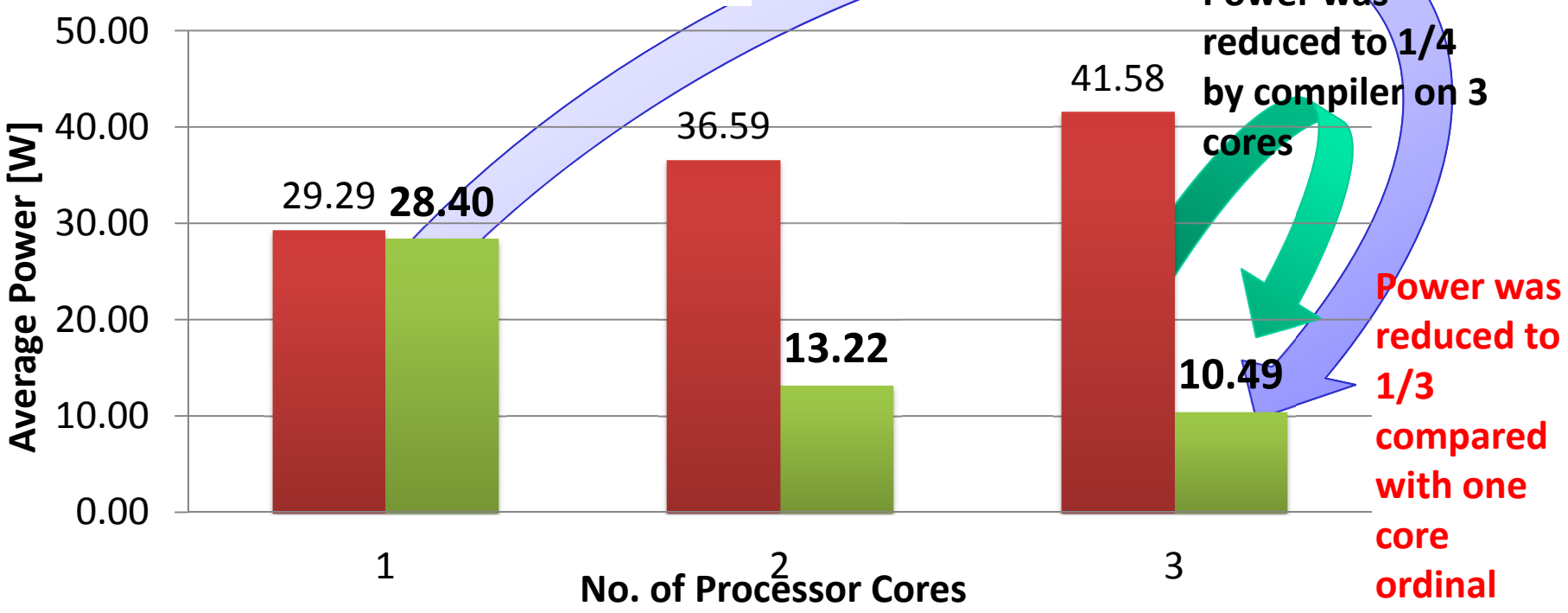


- Haswell Processor
  - OS Ubuntu 13.10
  - Intel CPU Core i7 4770K
    - 4 cores
    - L1 Cache: Load 64Bytes/cycle, Store 32Bytes/cycle
    - L2 Cache 64Bytes/cycle
    - L3 Cache 8 MB
    - Frequency 3.5GHz~0.8MHz
  - Memory 16GB (8GB×2)

Power Reduction on Intel Haswell for Real-time Optical Flow



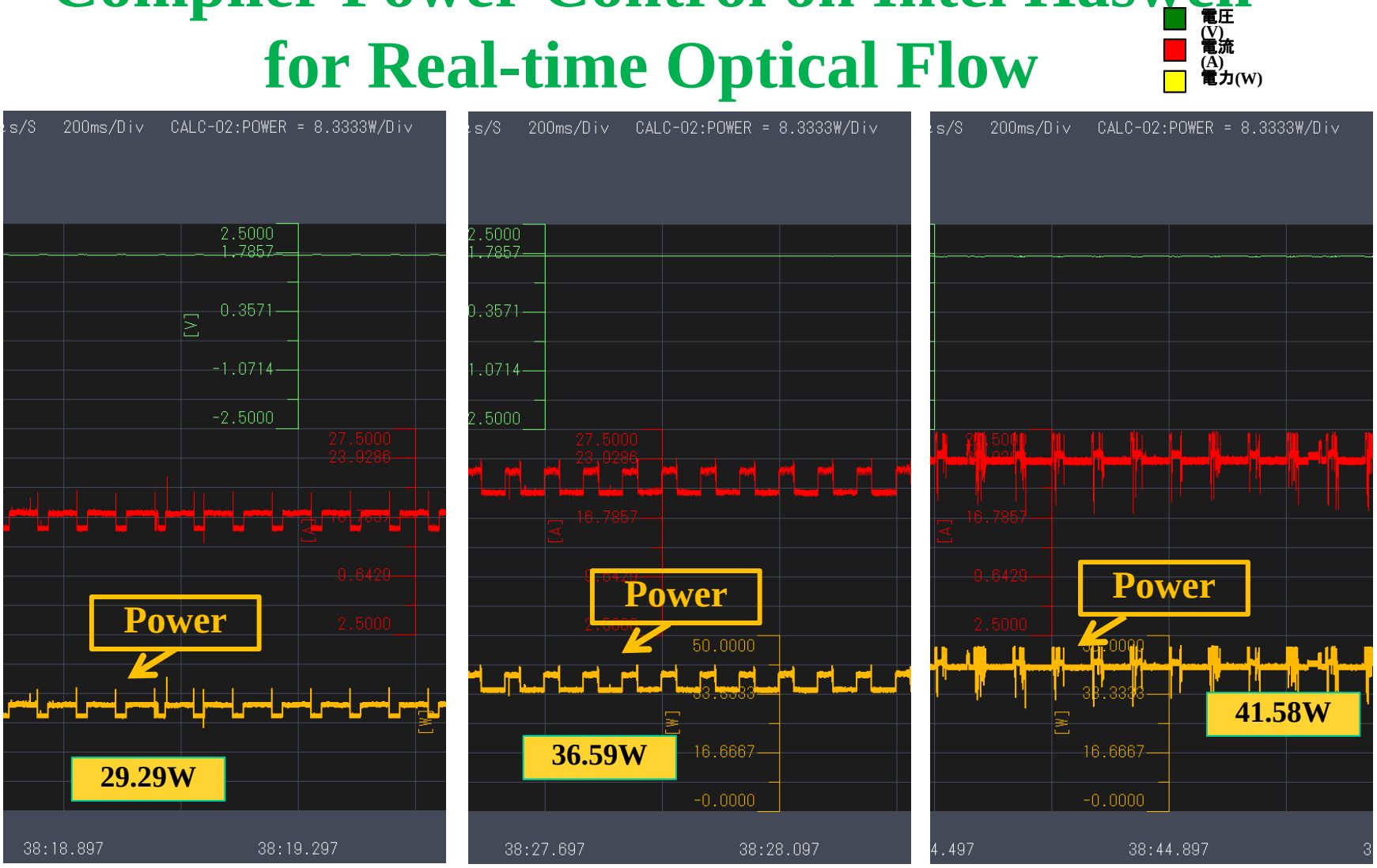
For HD 720p(1280x720) moving pictures
15fps (Deadline66.6[ms/frame])



Power was reduced to 1/4 by the compiler power optimization on the same 3 cores.

The power with 3 core was reduced to 1/3 against 1 core.

Power Waves for 1 Core to 3 Cores without the Compiler Power Control on Intel Haswell for Real-time Optical Flow



1 Core

2 Cores

3 Cores

Power Waves for 1 Core to 3 Cores with the Compiler Power Control on Intel Haswell for Real-time Optical Flow

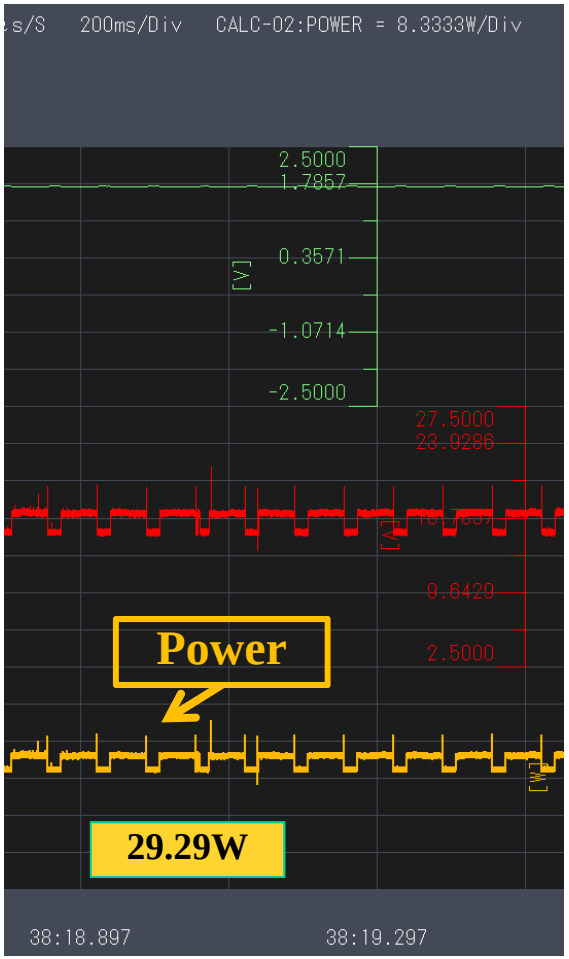
1 Core

2 Cores

3 Cores

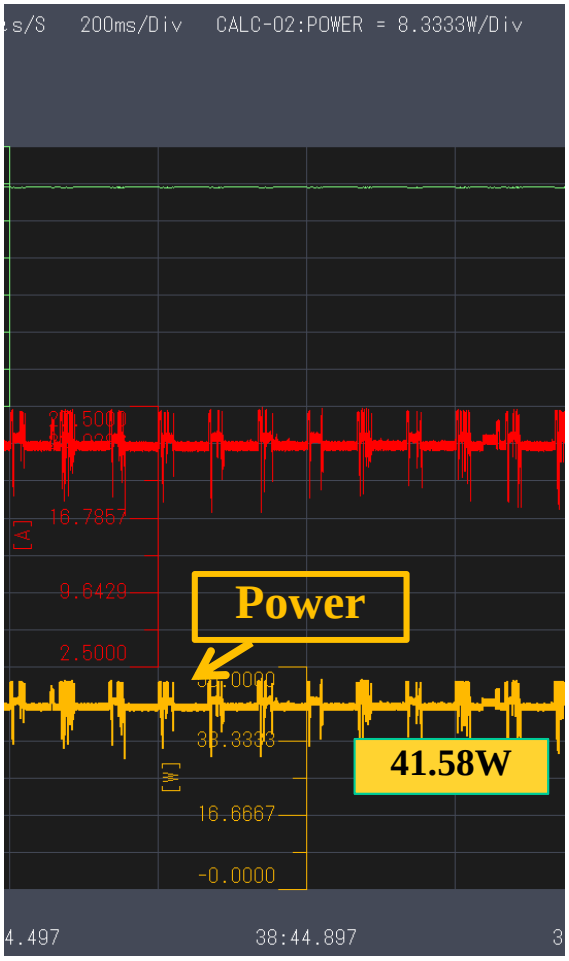
Power for 1 & 3Cores without Control vs. for 3 Cores with Control on Haswell

Without Power Control



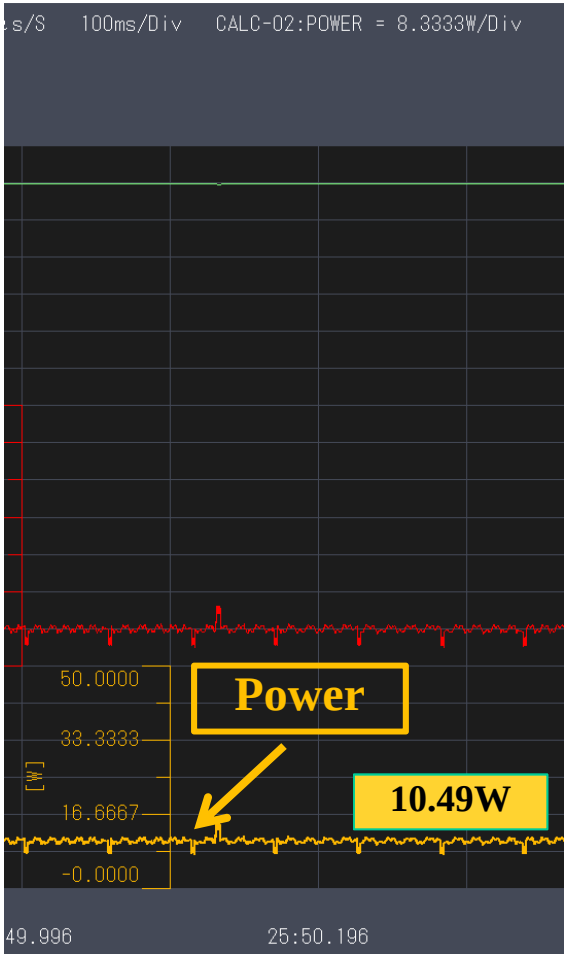
1 Core

Without Power Control



3 Cores

With Power Control



3 Cores

Future Multicore Products

Next Generation Automobiles

- Safer, more comfortable, energy efficient, environment friendly
- Cameras, radar, car2car communication, internet information integrated brake, steering, engine, moter control

Smart phones

-From everyday recharging to less than once a week
- Solar powered operation in emergency condition
- Keep health

Advanced medical systems

Cancer treatment,
Drinkable inner camera

- Emergency solar powered
- No cooling fun, No dust , clean usable inside OP room

Personal / Regional Supercomputers

Solar powered with more than 100 times power efficient : FLOPS/W

- Regional Disaster Simulators saving lives from tornadoes, localized heavy rain, fires with earth quakes