

# Low Power High Performance Multicore Hardware and Software Co-Design



**Hironori Kasahara, Ph.D., IEEE Fellow**

**IEEE Computer Society President 2018**

**Professor, Dept. of Computer Science & Engineering**

**Director, Advanced Multicore Processor Research Institute**

**Waseda University, Tokyo, Japan**

**URL: <http://www.kasahara.cs.waseda.ac.jp/>**

**1980 BS, 82 MS, 85 Ph.D. , Dept. EE, Waseda Univ.**  
**1985 Visiting Scholar: U. of California, Berkeley**  
**1986 Assistant Prof., 1988 Associate Prof., 1997,**  
**Waseda Univ., Now Dept. of Computer Sci. & Eng.**  
**1989-90 Research Scholar: U. of Illinois, Urbana-**  
**Champaign, Center for Supercomputing R&D**  
**2004 Director, Advanced Multicore Research**  
**Institute, 2017 member: the Engineering Academy**  
**of Japan and the Science Council of Japan**

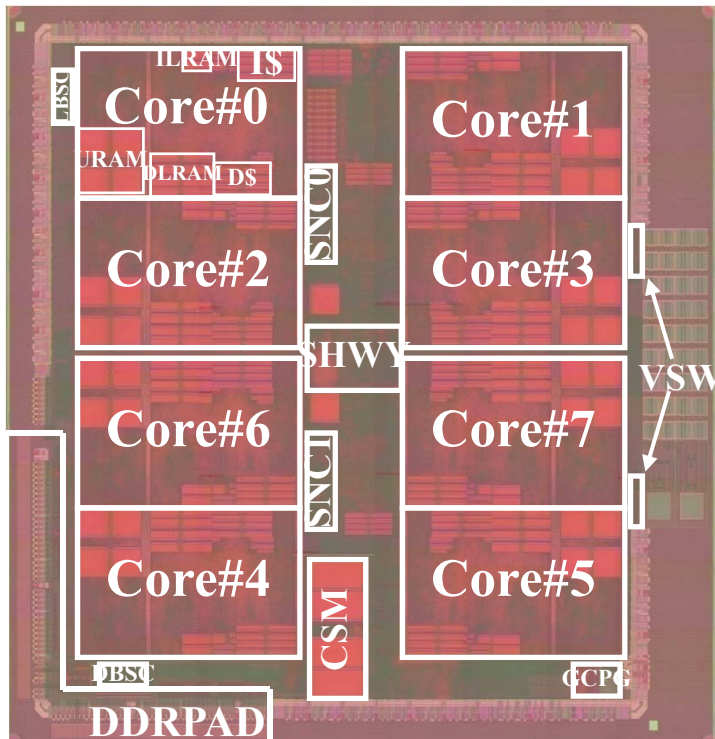
**2005 STARC Academia-Industry Research Award**  
**2008 LSI of the Year Second Prize**  
**2008 Intel AsiaAcademic Forum Best Research Award**  
**2010 IEEE CS Golden Core Member Award**  
**2014 Minister of Edu., Sci. & Tech. Research Prize**  
**2015 IPSJ Fellow**  
**2017 IEEE Fellow, IEEE Eta Kappa Nu**

**Reviewed Papers: 216, Invited Talks: 162, Granted**  
**Patents: 43 (Japan, US, GB, China), Articles in News**  
**Papers, Web News, Medias incl. TV etc.: 584**

**Committees in Societies and Government 255**  
**IEEE Computer Society President 2018,**  
**IEEE CS: BoG(2009-14), Executive Committee(2017-**  
**Multicore STC Chair (2012-), Japan Chair (2005-07),**  
**IPSJ Chair: HG for Magazine. & J. Edit, Sig. on ARC.**  
**【METI/NEDO】 Project Leaders: Multicore for**  
**Consumer Electronics, Advanced Parallelizing**  
**Compiler, Chair: Computer Strategy Committee**  
**【Cabinet Office】 CSTP Supercomputer Strategic**  
**ICT PT, Japan Prize Selection Committees, etc.**  
**【MEXT】 Info. Sci. & Tech. Committee,**  
**Supercomputers (Earth Simulator, HPCI Promo.,**  
**Next Gen. Supercomputer K) Committees, etc.**

# Multicores for Performance and Low Power

Power consumption is one of the biggest problems for performance scaling from smartphones to cloud servers and supercomputers (“K” more than 10MW) .



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”

$$\text{Power} \propto \text{Frequency} * \text{Voltage}^2$$

(Voltage  $\propto$  Frequency)

➔ Power  $\propto$  Frequency<sup>3</sup>

If Frequency is reduced to 1/4  
(Ex. 4GHz  $\rightarrow$  1GHz),  
Power is reduced to 1/64 and  
Performance falls down to 1/4 .

<Multicores>

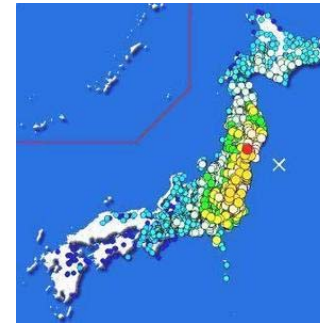
If 8cores are integrated on a chip,  
Power is still 1/8 and  
Performance becomes 2 times .

## Parallel Soft is important for scalable performance of multicore (LCPC2015)

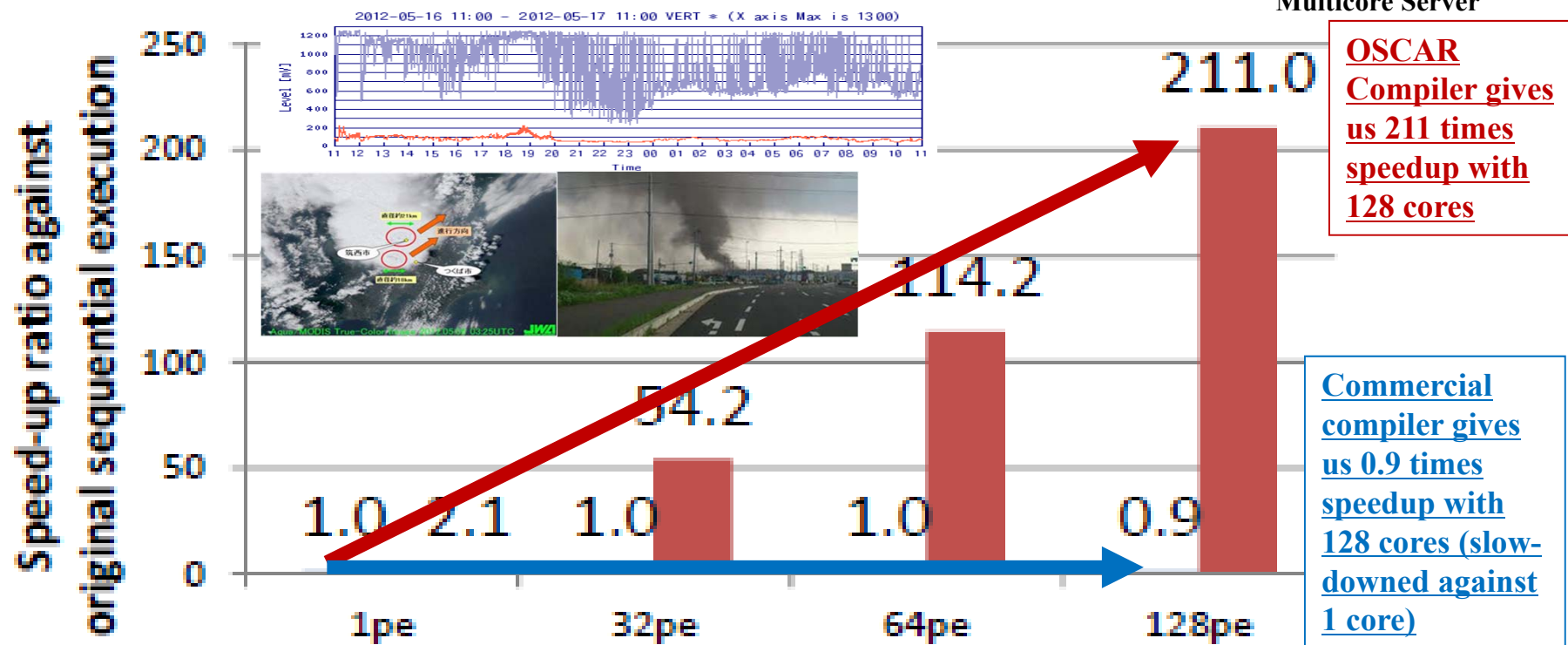
- Just more cores don't give us speedup
- Development cost and period of parallel software are getting a bottleneck of development of embedded systems, eg. IoT, Automobile

Earthquake wave propagation simulation GMS developed by National Research Institute for Earth Science and Disaster Resilience (NIED)

■ original (sun studio) ■ proposed method



Fujitsu M9000 SPARC Multicore Server

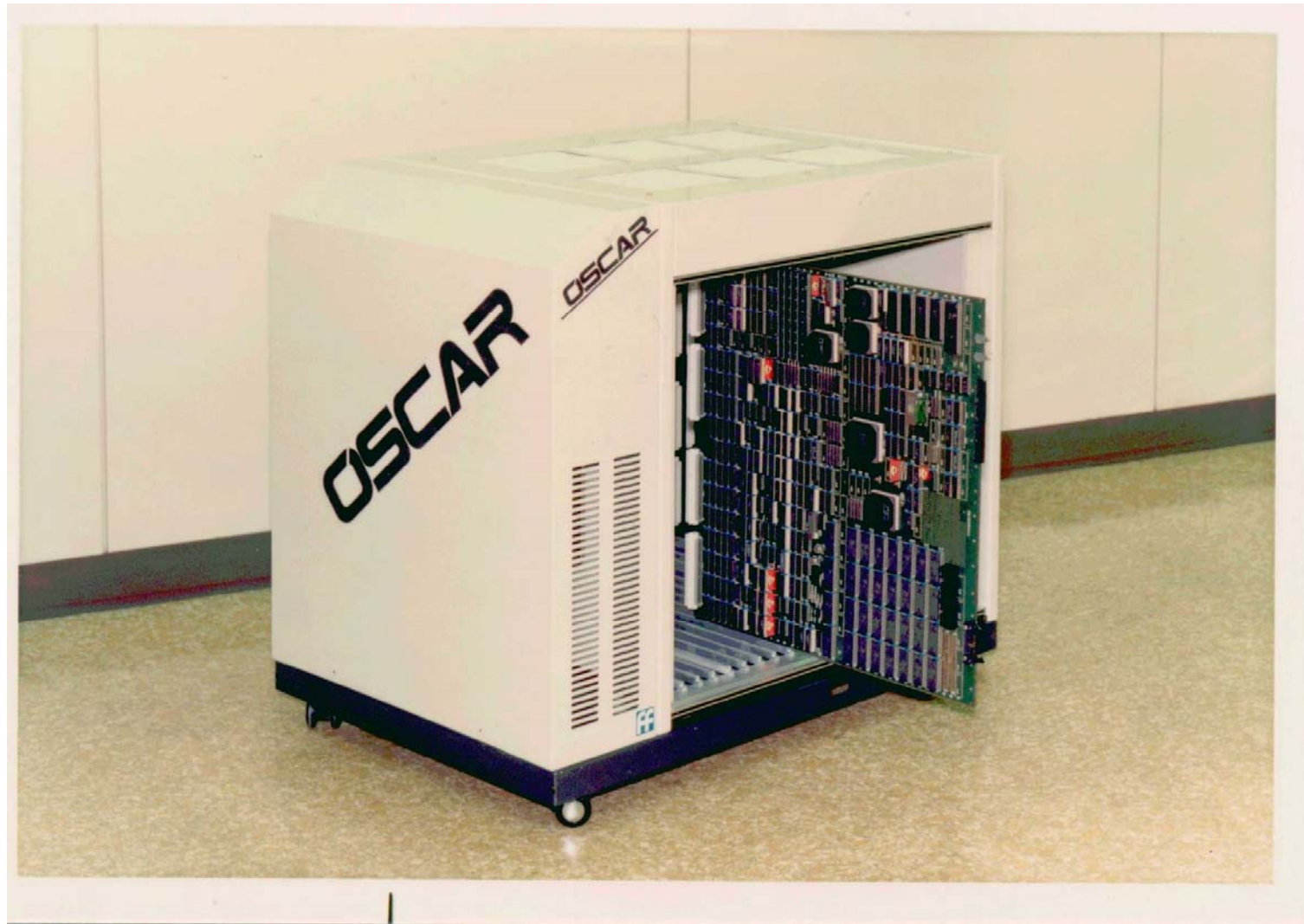


- Automatic parallelizing compiler available on the market gave us no speedup against execution time on 1 core on 64 cores
  - Execution time with 128 cores was slower than 1 core (0.9 times speedup)
- Advanced OSCAR parallelizing compiler gave us 211 times speedup with 128cores against execution time with 1 core using commercial compiler
  - OSCAR compiler gave us 2.1 times speedup on 1 core against commercial compiler by global cache optimization

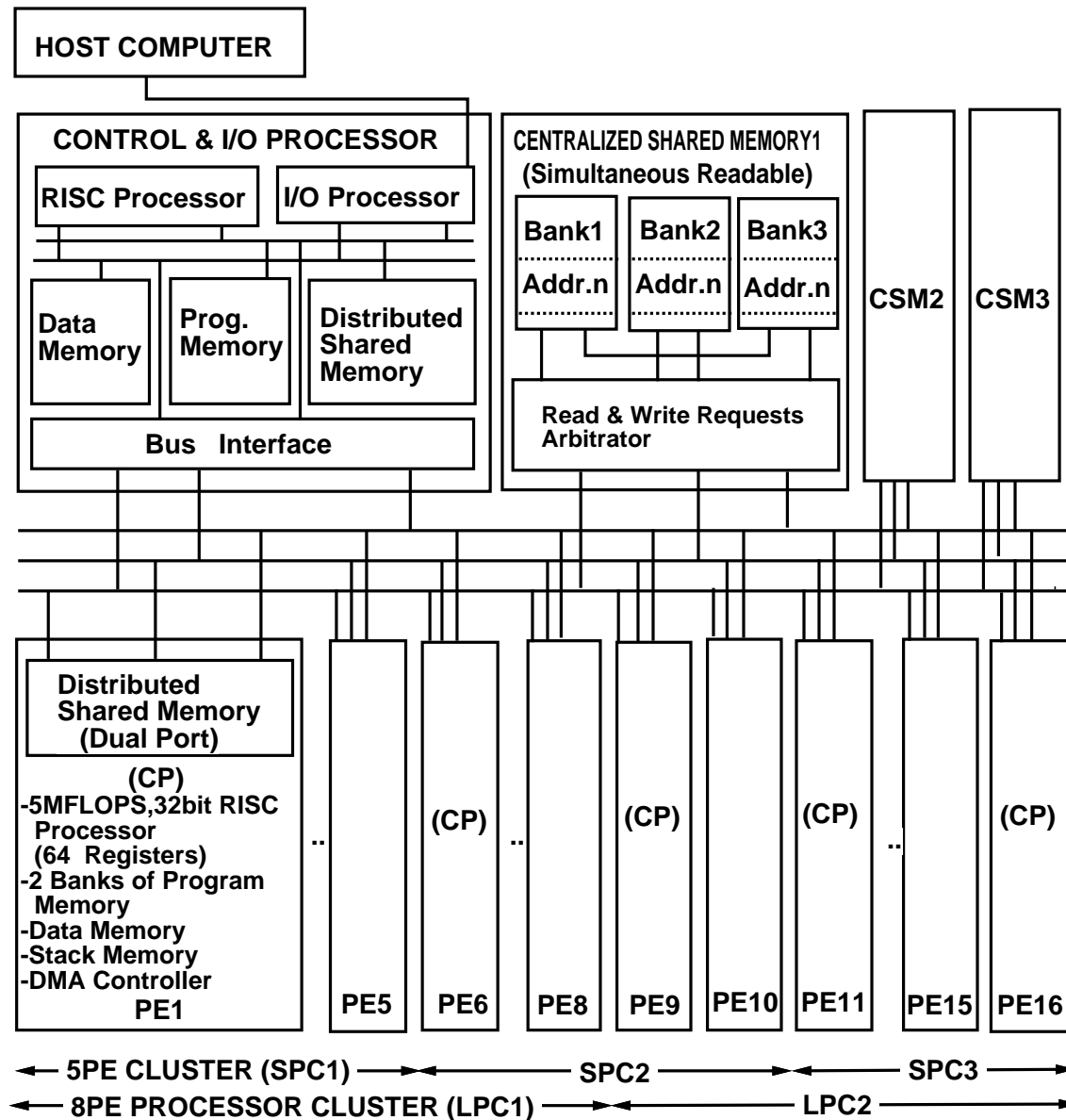
# 1987 OSCAR(Optimally Scheduled Advanced Multiprocessor)

## Co-design of Compiler and Architecture

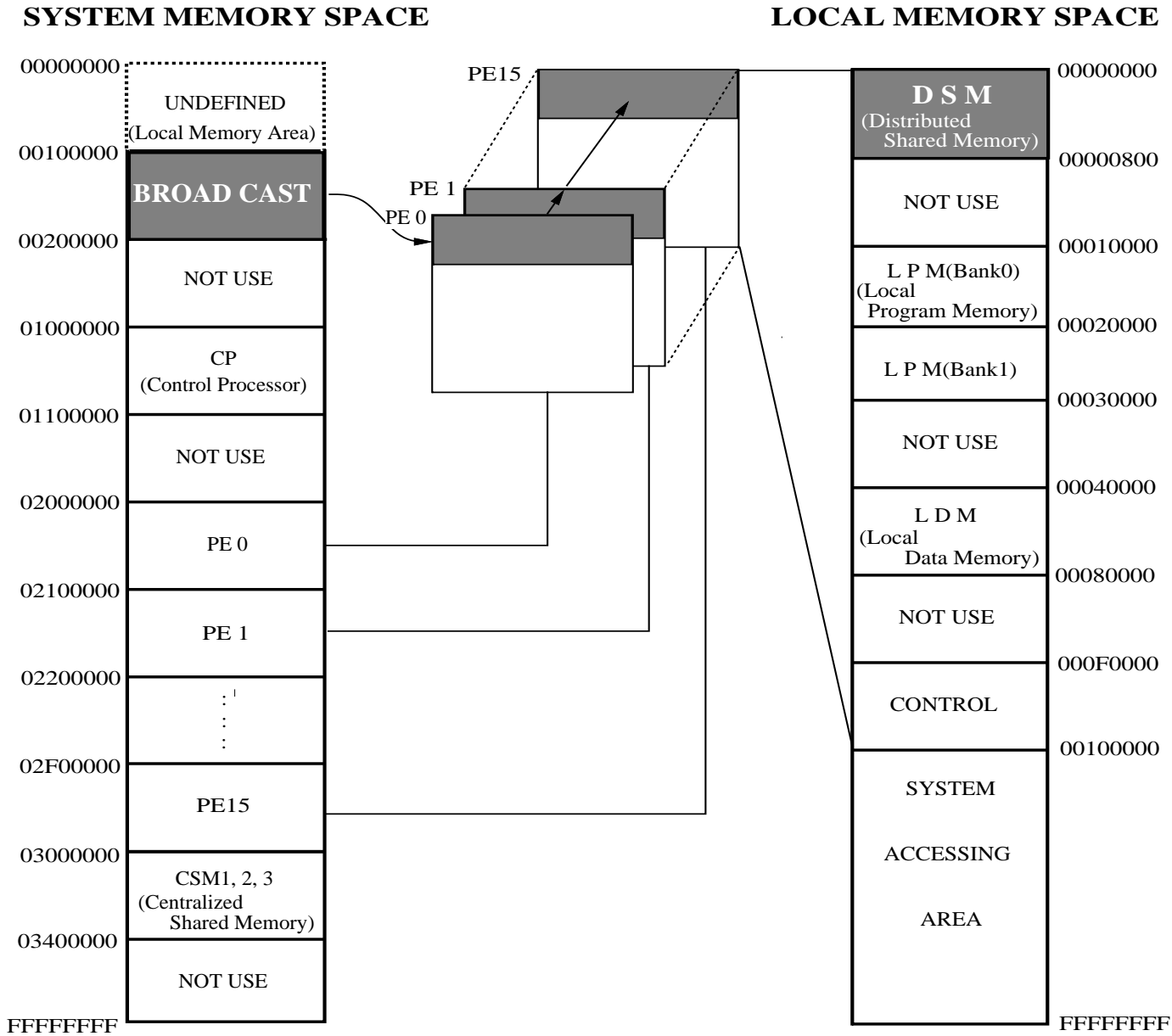
Looking at various applications, design a parallelizing compiler and design a multiprocessor/multicore-processor to support compiler optimization



# OSCAR(Optimally Scheduled Advanced Multiprocessor)

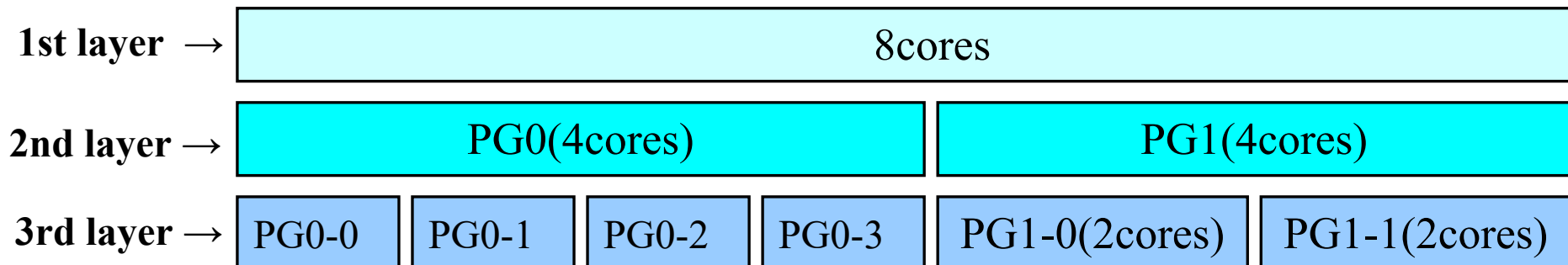


# OSCAR Memory Space (Global Address Space)



# Hierarchical Barrier Synchronization

- **Specifying a hierarchical group barrier**
  - **#pragma oscar group\_barrier (C)**
  - **!\$oscar group\_barrier (Fortran)**



# NWT



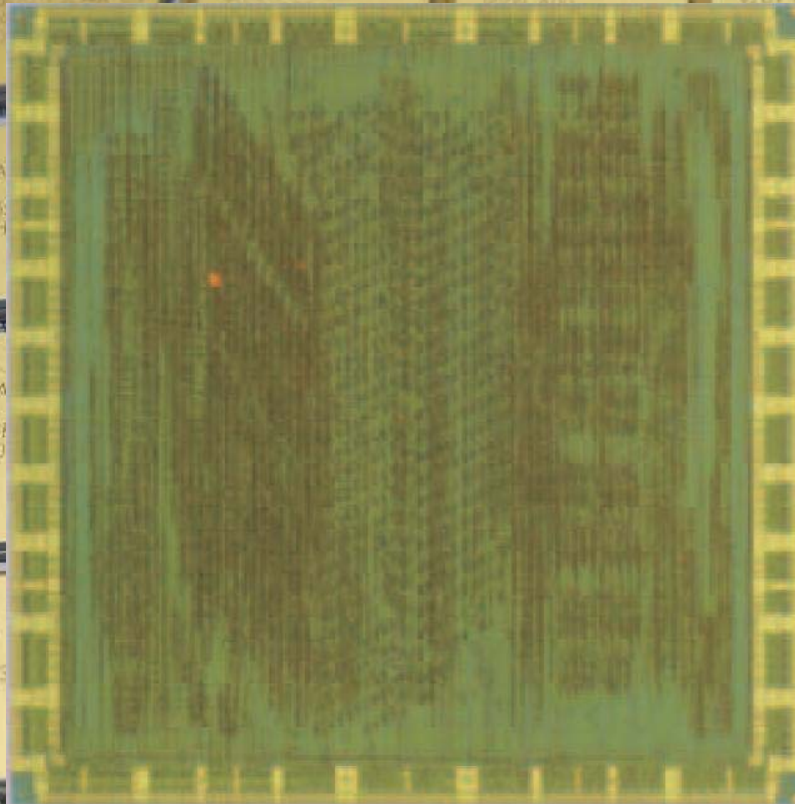
<b>Machine Cycle Time</b>	<b>9.5ns (105MHz)</b>
<b>PE Performance</b>	<b>1.68GFlops</b>
<b>PE Memory Size</b>	<b>256MB/PE</b>
<b>Crossbar Bandwidth</b>	<b>4B/cycle x 2 (send/receive simultaneous)/PE = 421MB/s x 2 /PE</b>
<b>Number of PEs</b>	<b>140PEs + 2Control Proc.</b>

**NAL computer center, Chofu, Tokyo, Feb. 1, 1993**

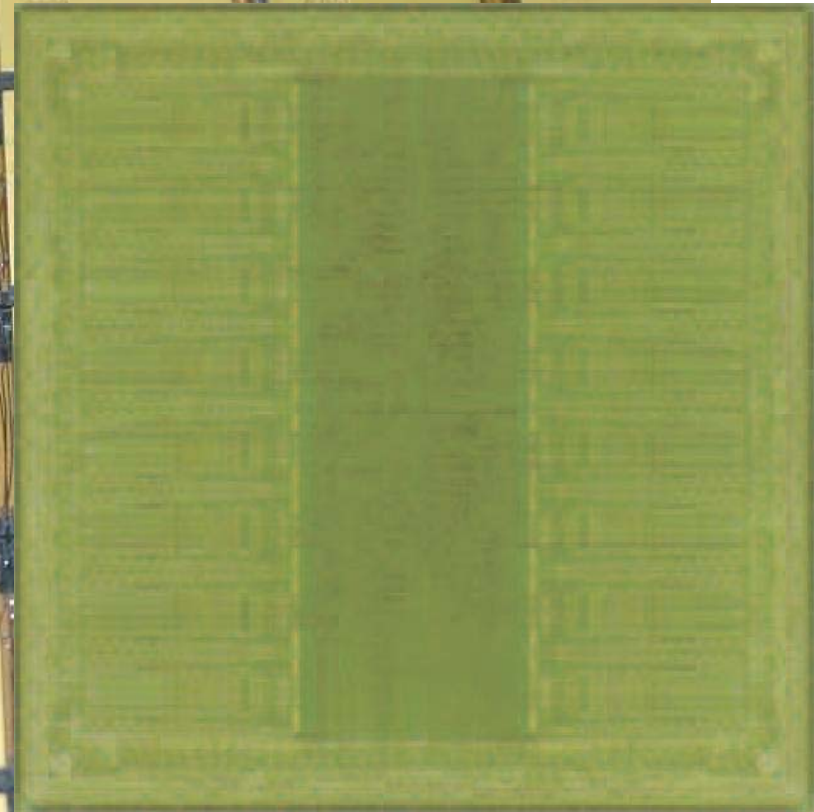


# VPP500/NWT

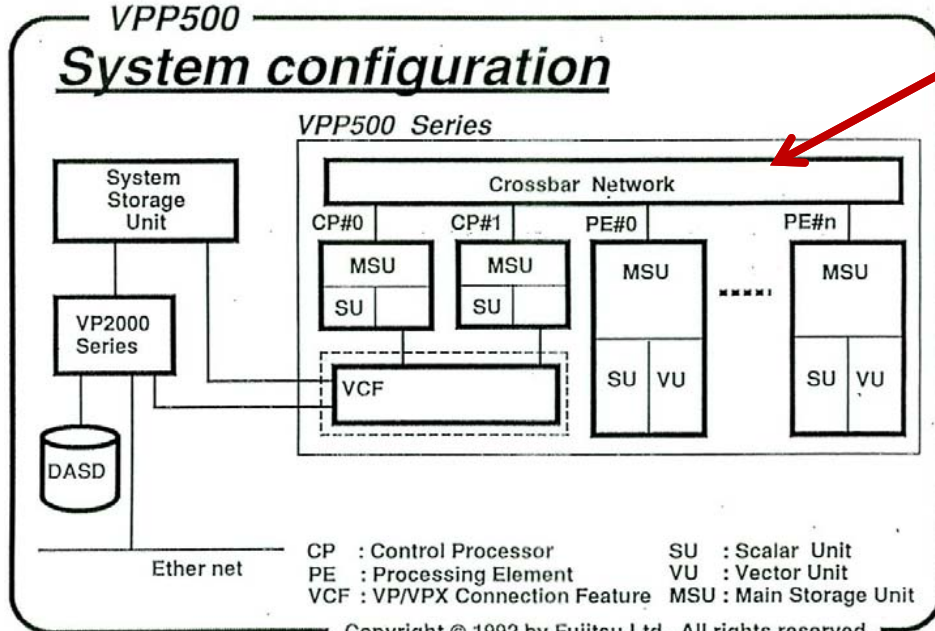
**GaAs**



**BiCMOS**



# Fujitsu Vector Parallel Supercomputer with Crossbar to a Chip

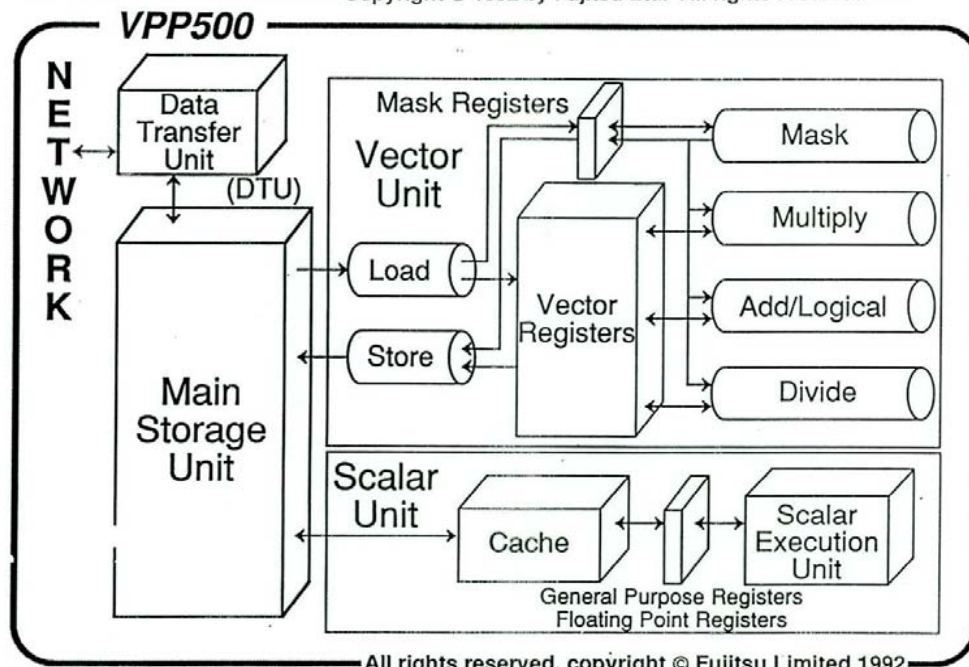
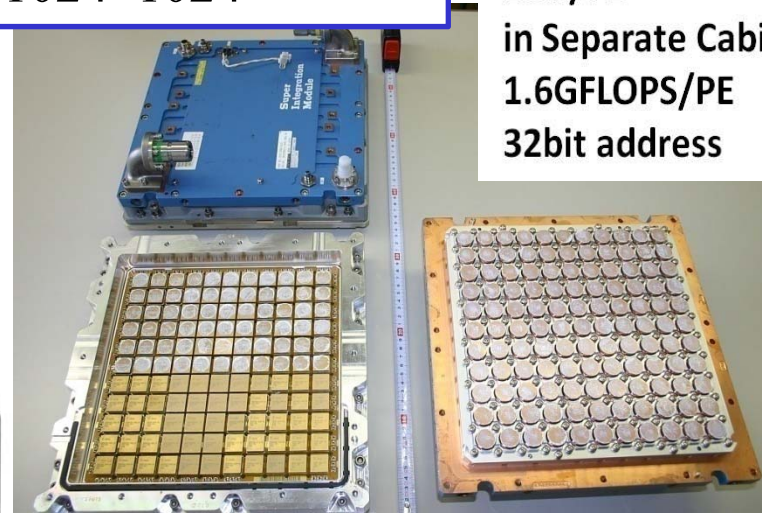


## Crossbar Network

VPP500: 256\*256  
VPP5000: 1024\*1024

## VPP500

GaAs/BiCMOS/ECL  
Water Cooling  
10ns  
1GB/PE  
in Separate Cabinet  
1.6GFLOPS/PE  
32bit address

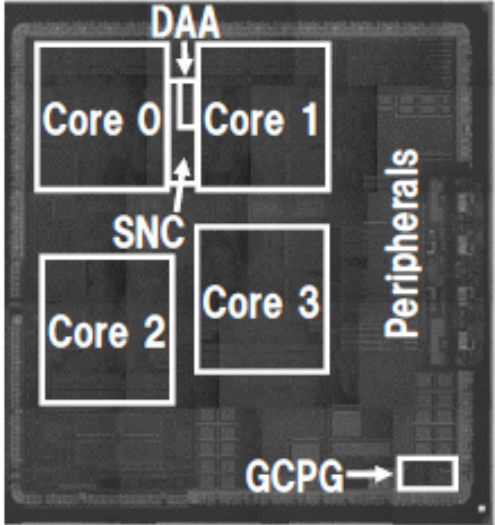
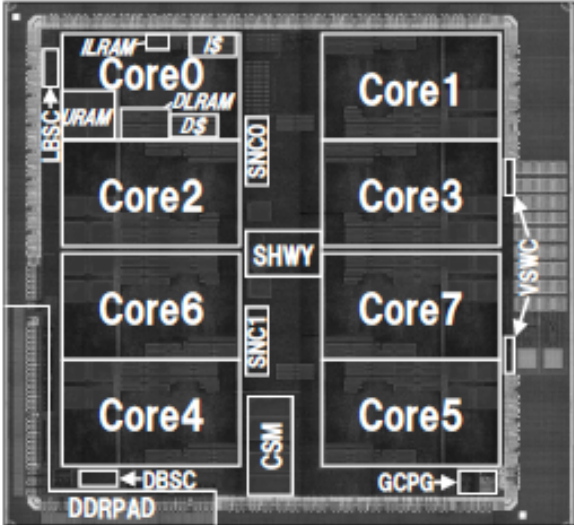
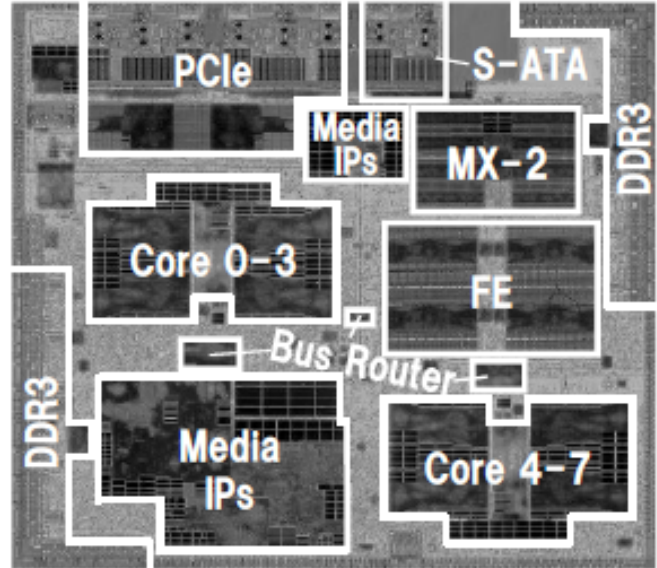


## VPP5000

0.25um CMOS  
30M Tr./Chip  
Forced Air Cooling  
3.5ns clock cycle  
8GB/PE  
on PE Board  
9.1GFLOPS/PE  
32/64 bit address

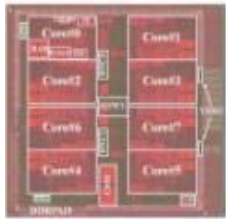


# 4 core multicore RP1 (2007) , 8 core multicore RP2 (2008) and 15 core Heterogeneous multicore RPX (2010) developed in NEDO Projects with Hitachi and Renesas

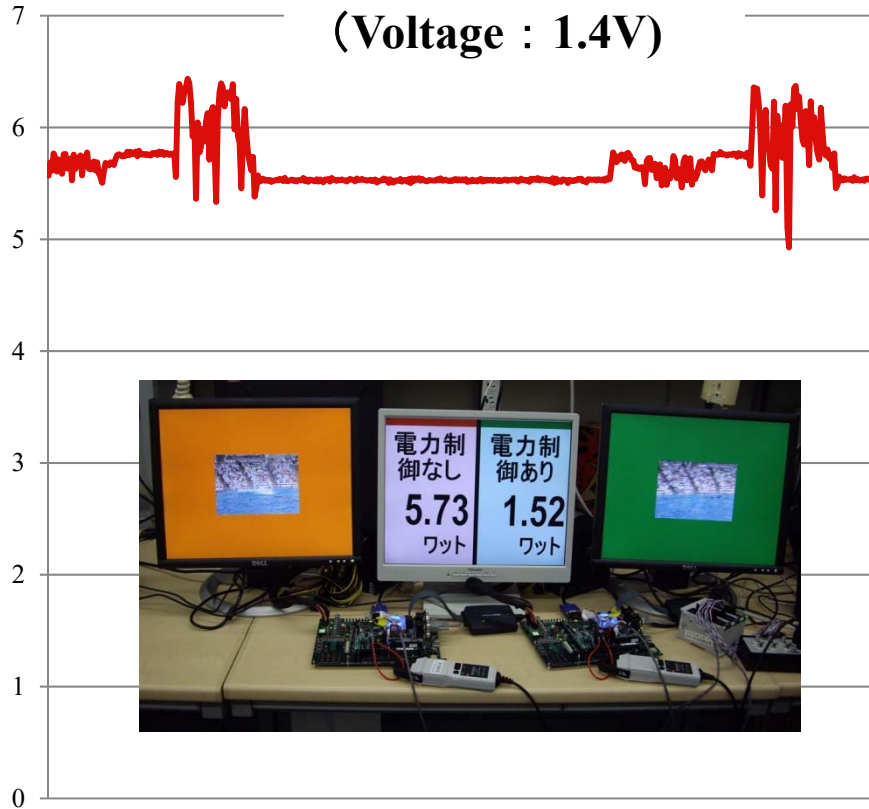
<b>RP-1</b> (ISSCC2007 #5.3)	<b>RP-2</b> (ISSCC2008 #4.5)	<b>RP-X</b> (ISSCC2010 #5.3)
		
90nm, 8-layer, triple-Vth, CMOS	90nm, 8-layer, triple-Vth, CMOS	45nm, 8-layer, triple-Vth, CMOS
97.6 mm <sup>2</sup> (9.88 x 9.88 mm)	104.8 mm <sup>2</sup> (10.61 x 9.88 mm)	153.8 mm <sup>2</sup> (12.4 x 12.4 mm)
1.0V (internal), 1.8/3.3V (I/O)	1.0-1.4V (internal), 1.8/3.3V (I/O)	1.0-1.2V (internal), 1.2-3.3V (I/O)
600MHz ,4.32 GIPS,16.8 GFLOPS	600MHz , 8.64 GIPS, 33.6 GFLOPS	648MHz, 13.7GIPS, 115GOPS, 36.2GFLOPS
11.4 GOPS/W (32b換算)	18.3 GOPS/W (32b換算)	37.3 GOPS/W (32b換算)

# 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

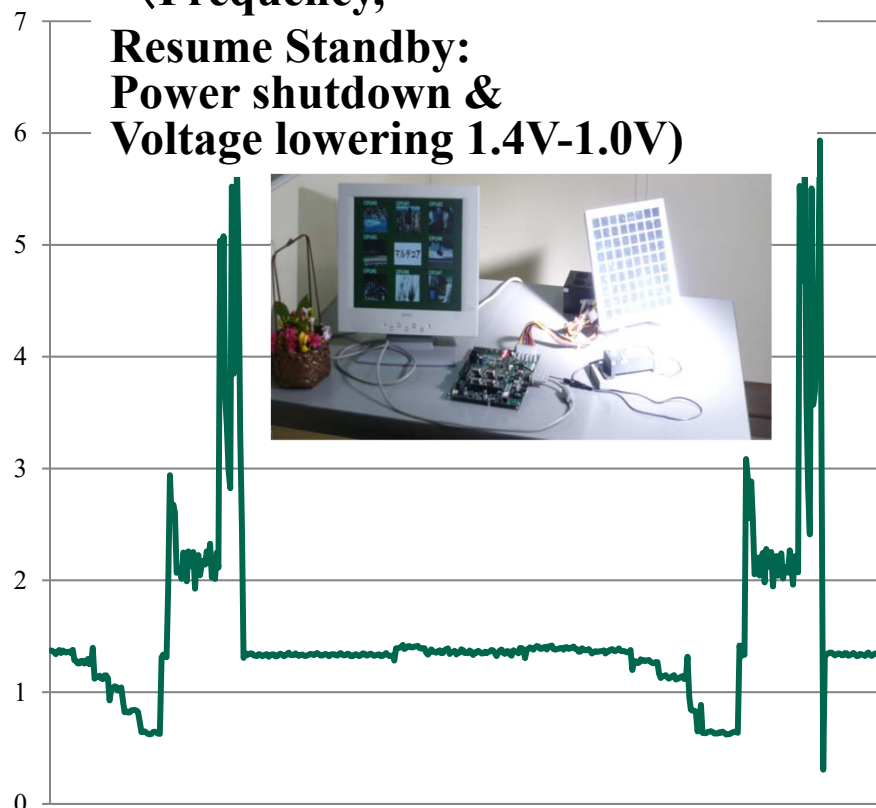


Without Power Control  
(Voltage : 1.4V)



Avg. Power  
5.73 [W]

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



Avg. Power  
1.52 [W]

73.5% Power Reduction



# OSCAR Parallelizing Compiler

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

**Multigrain Parallelization**(LCPC1991,2001,04)  
coarse-grain parallelism among loops and subroutines (2000 on SMP), near fine grain parallelism among statements (1992) in addition to loop parallelism

## Data Localization

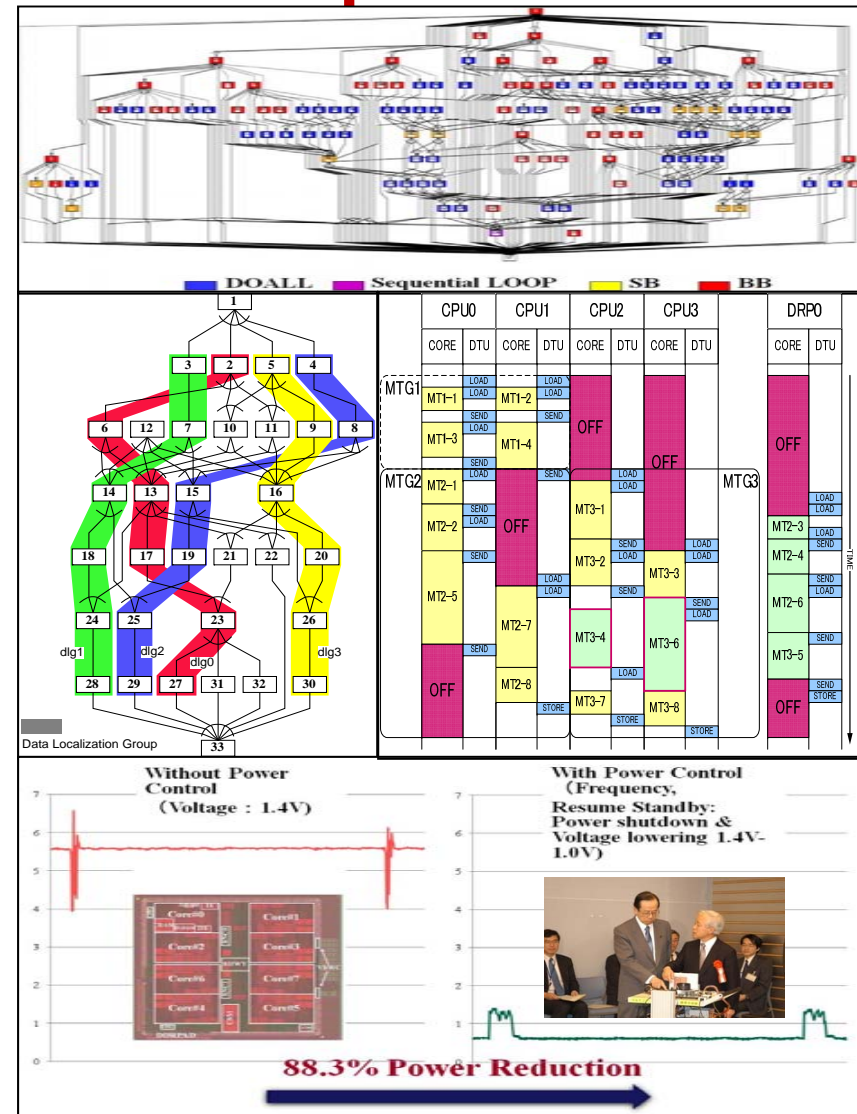
Automatic data management for distributed shared memory, cache and local memory (Local Memory 1995, 2016 on RP2,Cache2001,03)  
Software Coherent Control (2017)

## Data Transfer Overlapping(2016 partially)

Data transfer overlapping using Data Transfer Controllers (DMAs)

## Power Reduction

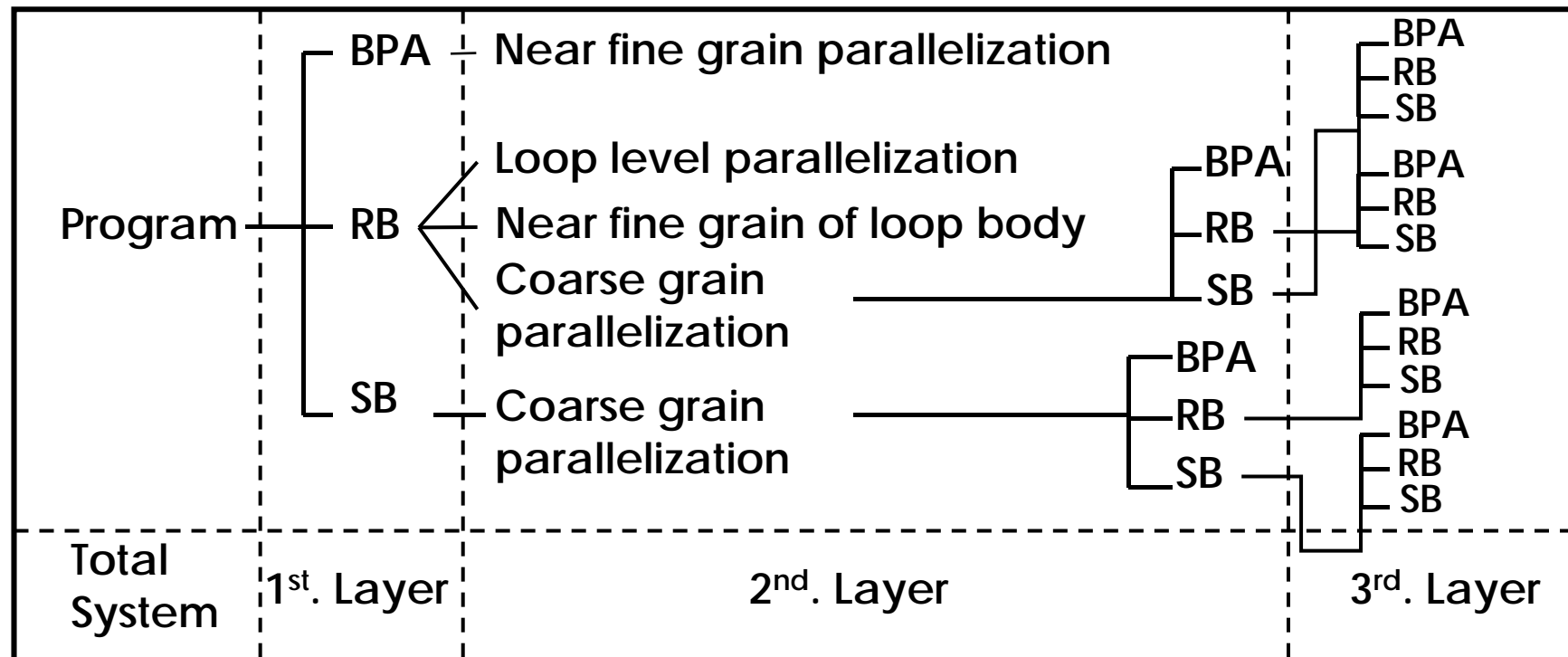
(2005 for Multicore, 2011 Multi-processes, 2013 on ARM)  
Reduction of consumed power by compiler control DVFS and Power gating with hardware supports.



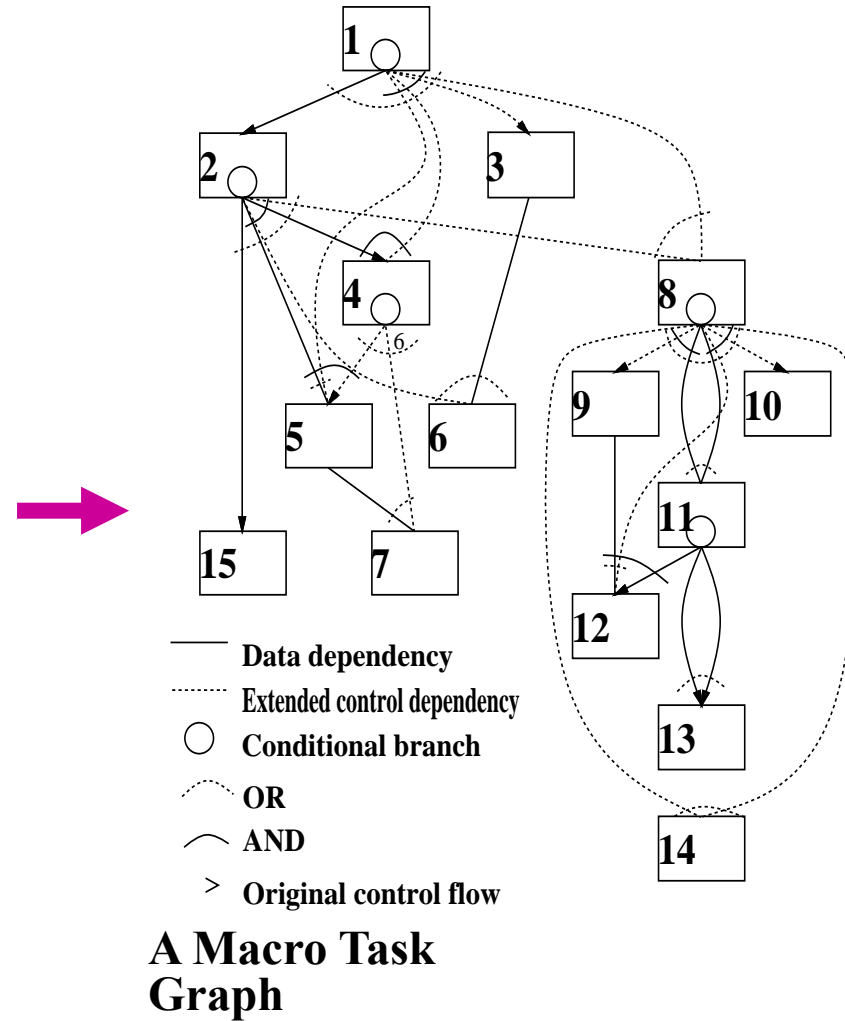
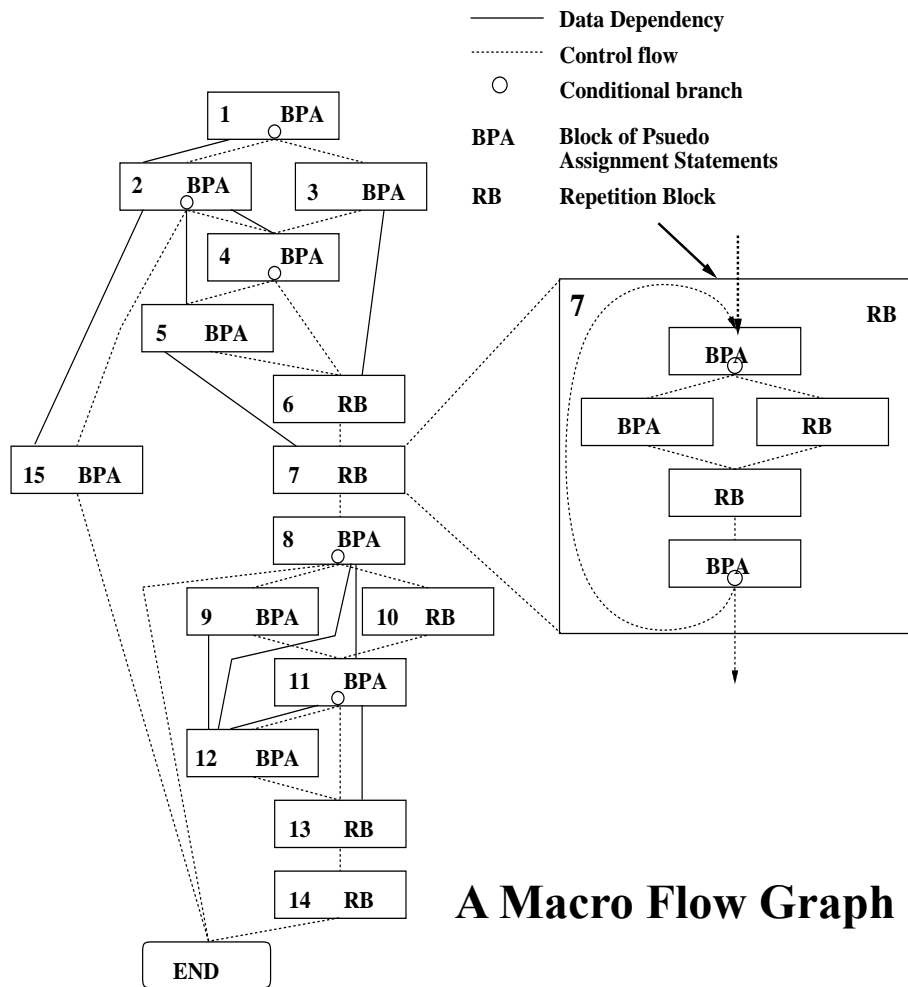
# Generation of Coarse Grain Tasks

## ■ 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)



### Earliest Executable Conditions

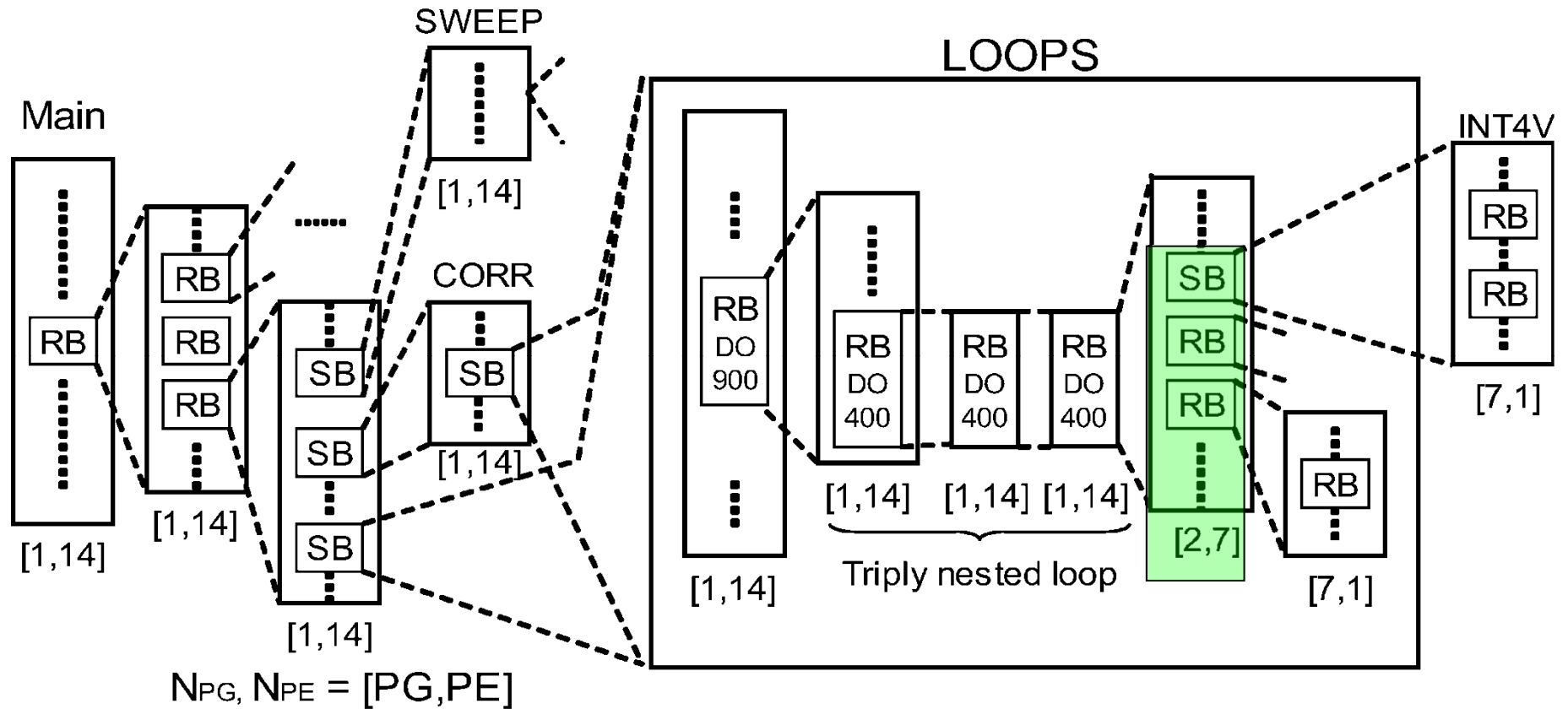
Macrotask No.	Earliest Executable Condition
1	
2	1 2
3	(1) 3
4	2 4 OR (1) 3
5	(4) 5 AND [ 2 4 OR (1) 3 ]
6	3 OR (2) 4
7	5 OR (4) 6
8	(2) 4 OR (1) 3
9	(8) 9
10	(8) 10
11	8 9 OR 8 10
12	11 12 AND [ 9 OR (8) 10 ]
13	11 13 OR 11 12
14	(8) 9 OR (8) 10
15	2 15



# Automatic processor assignment in 103.su2cor

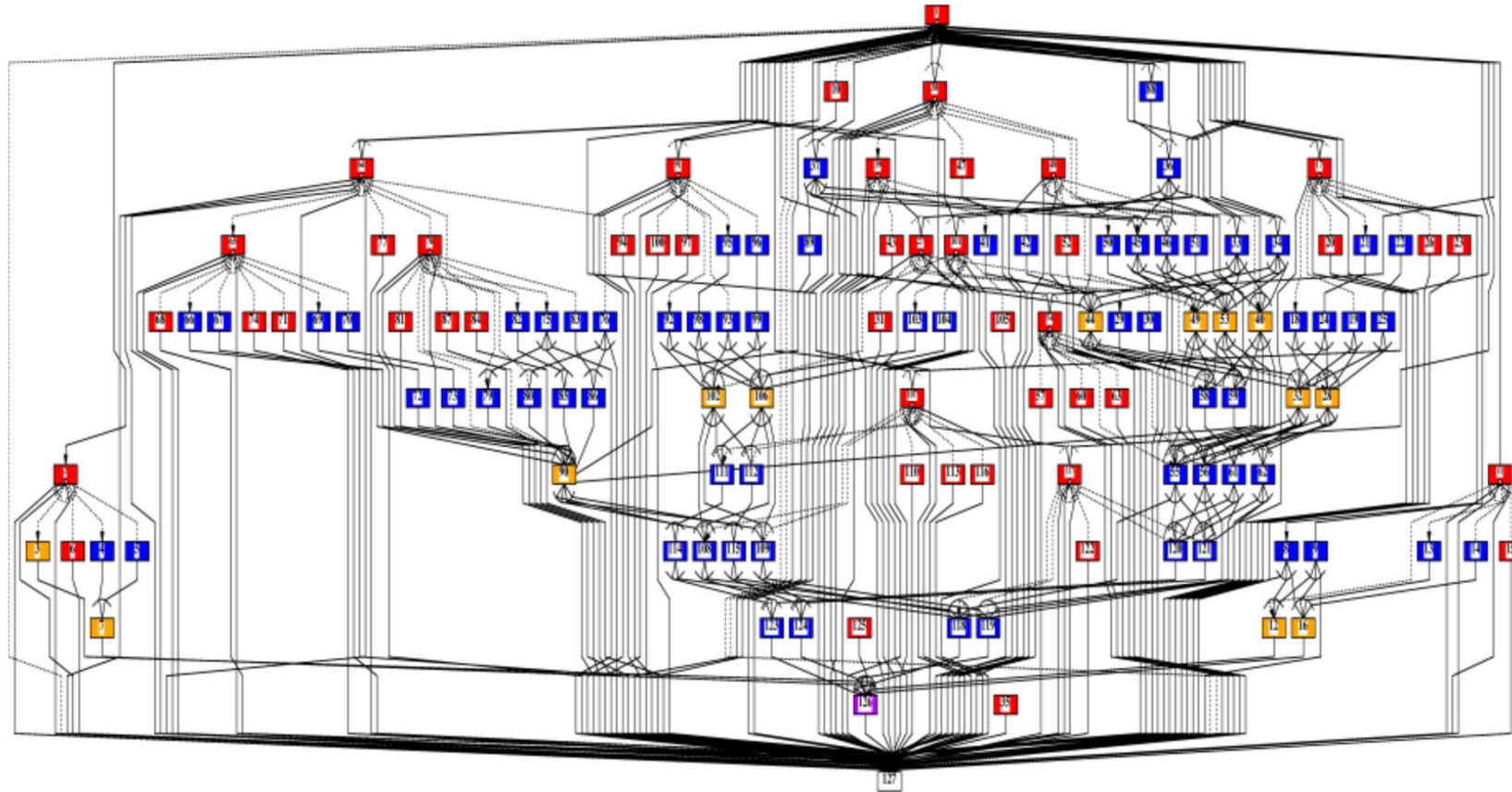
- Using 14 processors

Coarse grain parallelization within DO400



# MTG of Su2cor-LOOPS-DO400

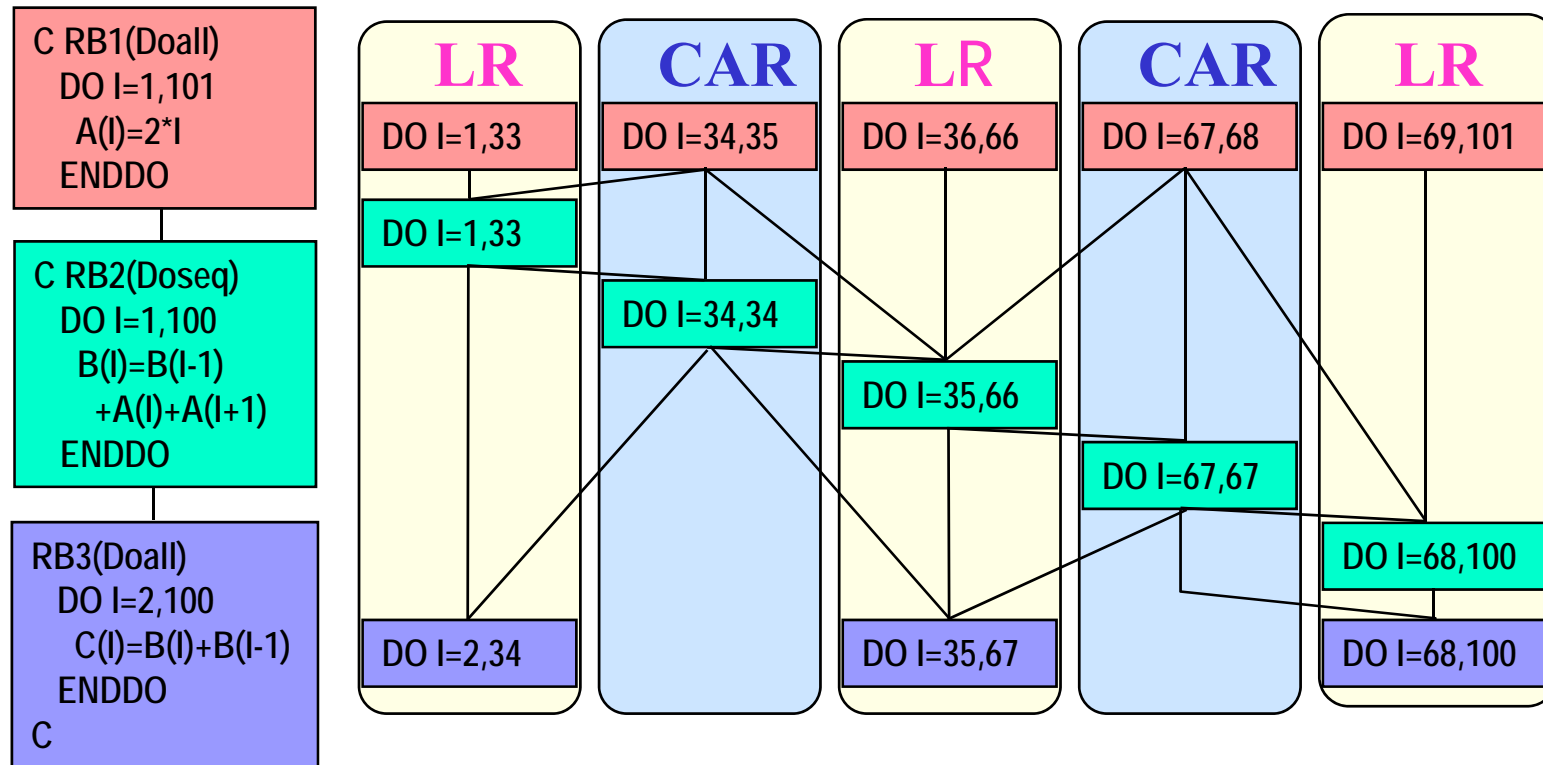
- **Coarse grain parallelism PARA\_ALD = 4.3**



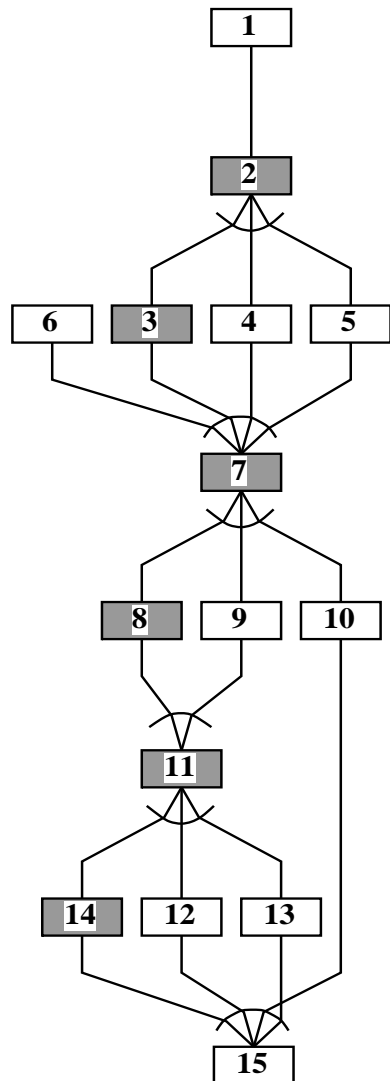
■ DOALL ■ Sequential LOOP ■ SB ■ BB

# Data-Localization: Loop Aligned Decomposition

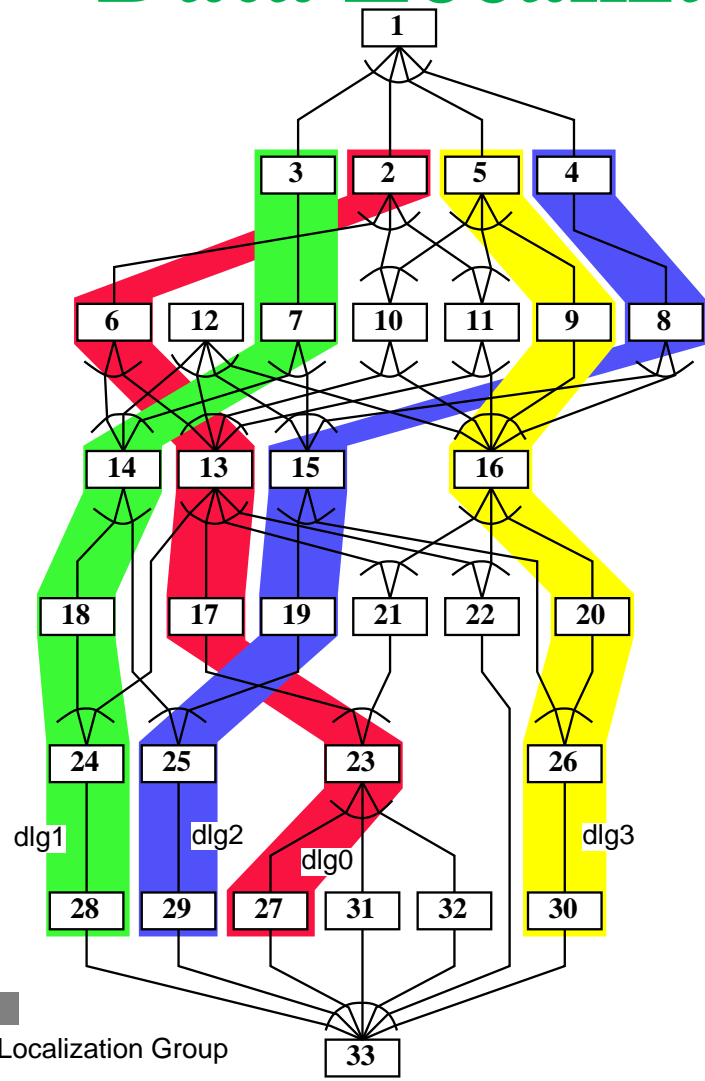
- Decompose multiple loop (Doall and Seq) into **CARs** and **LRs** considering inter-loop data dependence.
  - Most data in **LR** can be passed through LM.
  - LR: Localizable Region, CAR: Commonly Accessed Region**



# Data Localization



MTG



MTG after Division

■ Data Localization Group

PE0	PE1
12	1
2	3
6	7
4	14
8	18
15	5
19	9
25	11
29	10
13	16
17	20
22	26
21	30
23	24
27	28
	32
	31

A schedule for two processors

# An Example of Data Localization for Spec95 Swim

```

DO 200 J=1,N
DO 200 I=1,M
  UNEW(I+1,J) = UOLD(I+1,J)+
1  TDTS8*(Z(I+1,J+1)+Z(I+1,J))*(CV(I+1,J+1)+CV(I,J+1)+CV(I,J)
2  +CV(I+1,J))-TDTSDX*(H(I+1,J)-H(I,J))
  VNEW(I,J+1) = VOLD(I,J+1)-TDTSD8*(Z(I+1,J+1)+Z(I,J+1))
1  *(CU(I+1,J+1)+CU(I,J+1)+CU(I,J)+CU(I+1,J))
2  -TDTSDY*(H(I,J+1)-H(I,J))
  PNEW(I,J) = POLD(I,J)-TDTSDX*(CU(I+1,J)-CU(I,J))
1  -TDTSDY*(CV(I,J+1)-CV(I,J))
200 CONTINUE
  
```

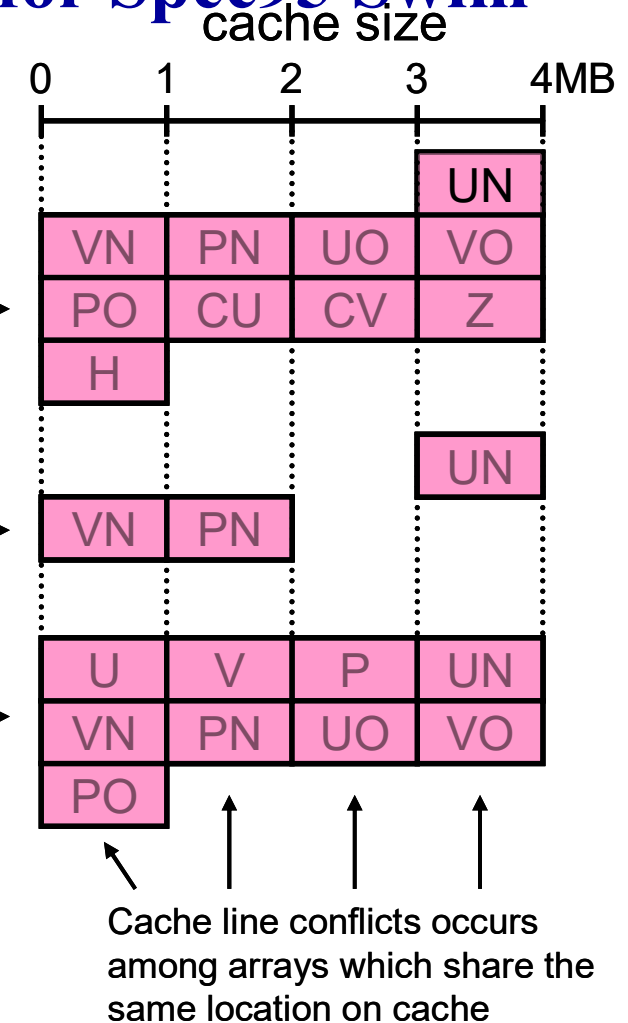
```

DO 210 J=1,N
  UNEW(1,J) = UNEW(M+1,J)
  VNEW(M+1,J+1) = VNEW(1,J+1)
  PNEW(M+1,J) = PNEW(1,J)
210 CONTINUE
  
```

```

DO 300 J=1,N
DO 300 I=1,M
  UOLD(I,J) = U(I,J)+ALPHA*(UNEW(I,J)-2.*U(I,J)+UOLD(I,J))
  VOLD(I,J) = V(I,J)+ALPHA*(VNEW(I,J)-2.*V(I,J)+VOLD(I,J))
  POLD(I,J) = P(I,J)+ALPHA*(PNEW(I,J)-2.*P(I,J)+POLD(I,J))
300 CONTINUE
  
```

(a) An example of target loop group for data localization



(b) Image of alignment of arrays on cache accessed by target loops

# Data Layout for Removing Line Conflict Misses by Array Dimension Padding

Declaration part of arrays in spec95 swim

before padding

after padding

PARAMETER (N1=513, N2=513)

PARAMETER (N1=513, N2=544)

COMMON U(N1,N2), V(N1,N2), P(N1,N2),

COMMON U(N1,N2), V(N1,N2), P(N1,N2),

\* UNEW(N1,N2), VNEW(N1,N2),

\* UNEW(N1,N2), VNEW(N1,N2),

1 PNEW(N1,N2), UOLD(N1,N2),

1 PNEW(N1,N2), UOLD(N1,N2),

\* VOLD(N1,N2), POLD(N1,N2),

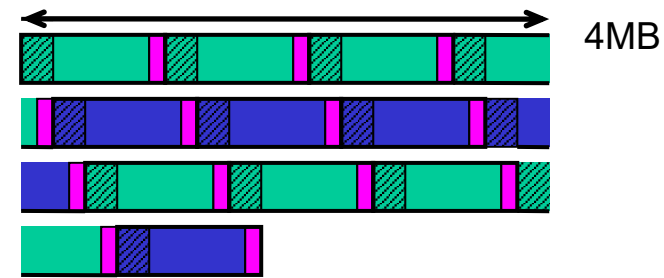
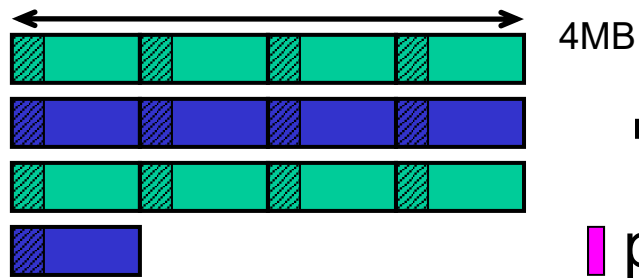
\* VOLD(N1,N2), POLD(N1,N2),

2 CU(N1,N2), CV(N1,N2),

2 CU(N1,N2), CV(N1,N2),

\* Z(N1,N2), H(N1,N2)

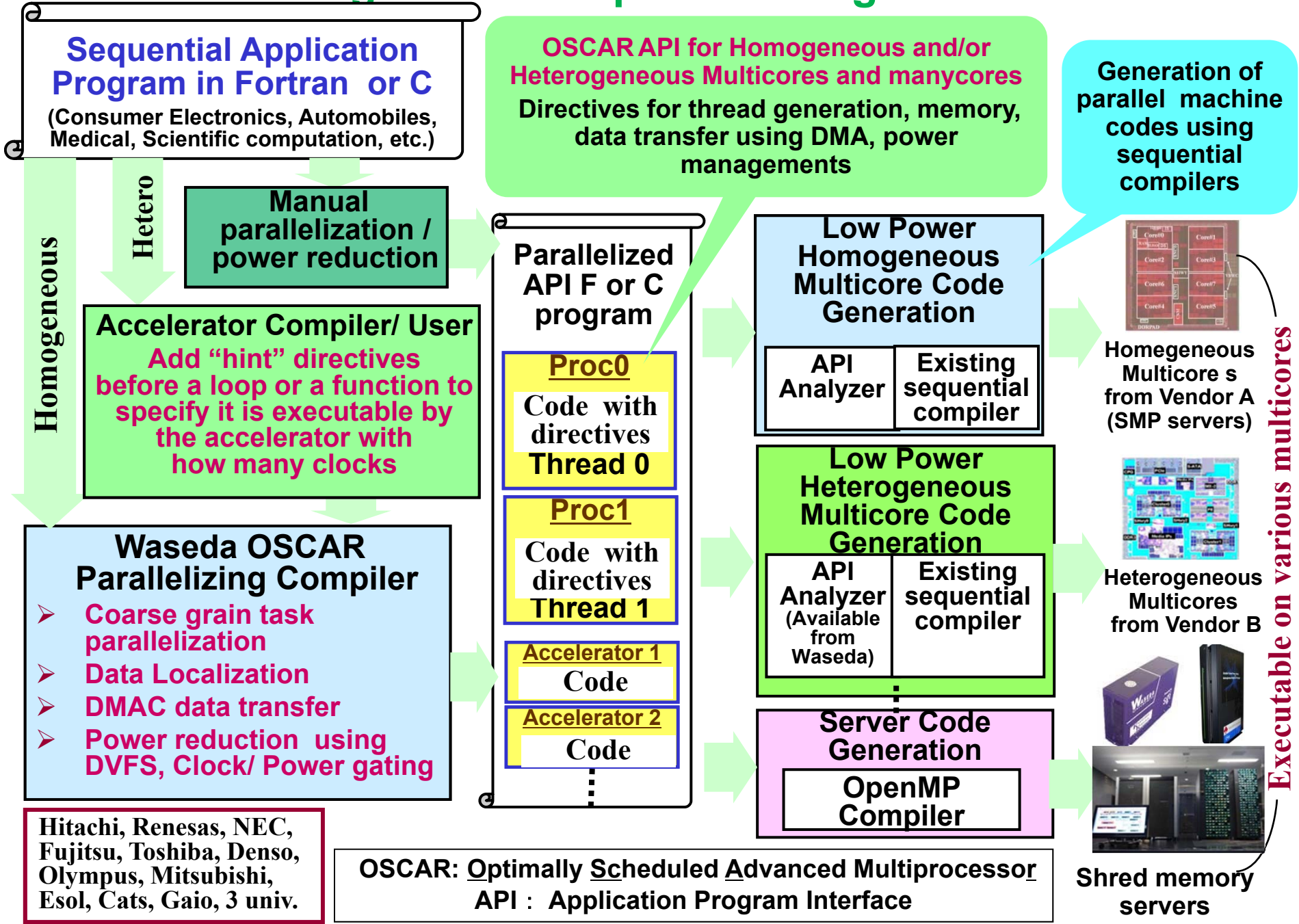
\* Z(N1,N2), H(N1,N2)



Box: Access range of DLG0

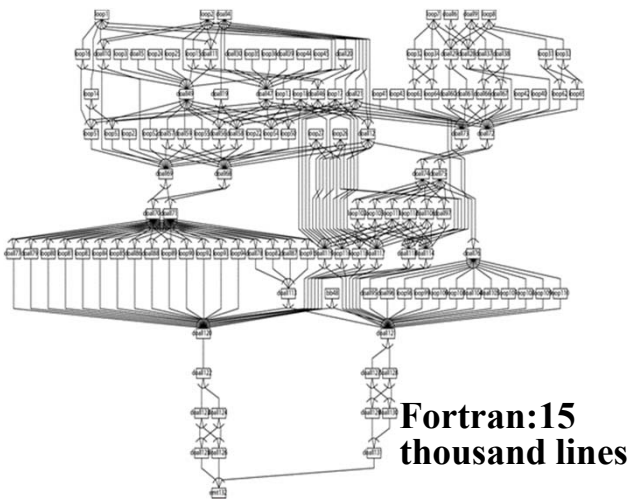
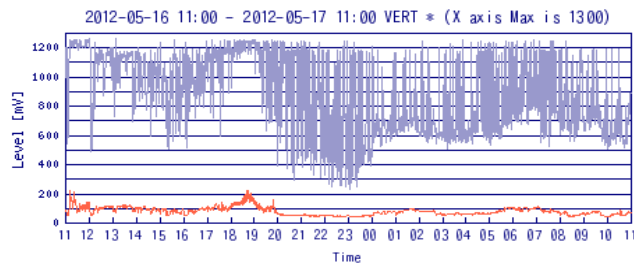


# Multicore Program Development Using OSCAR API V2.0

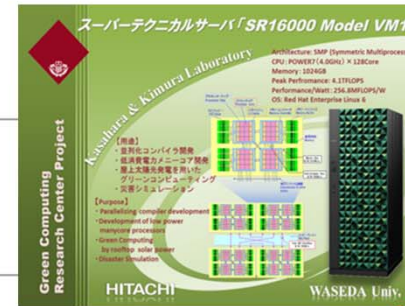
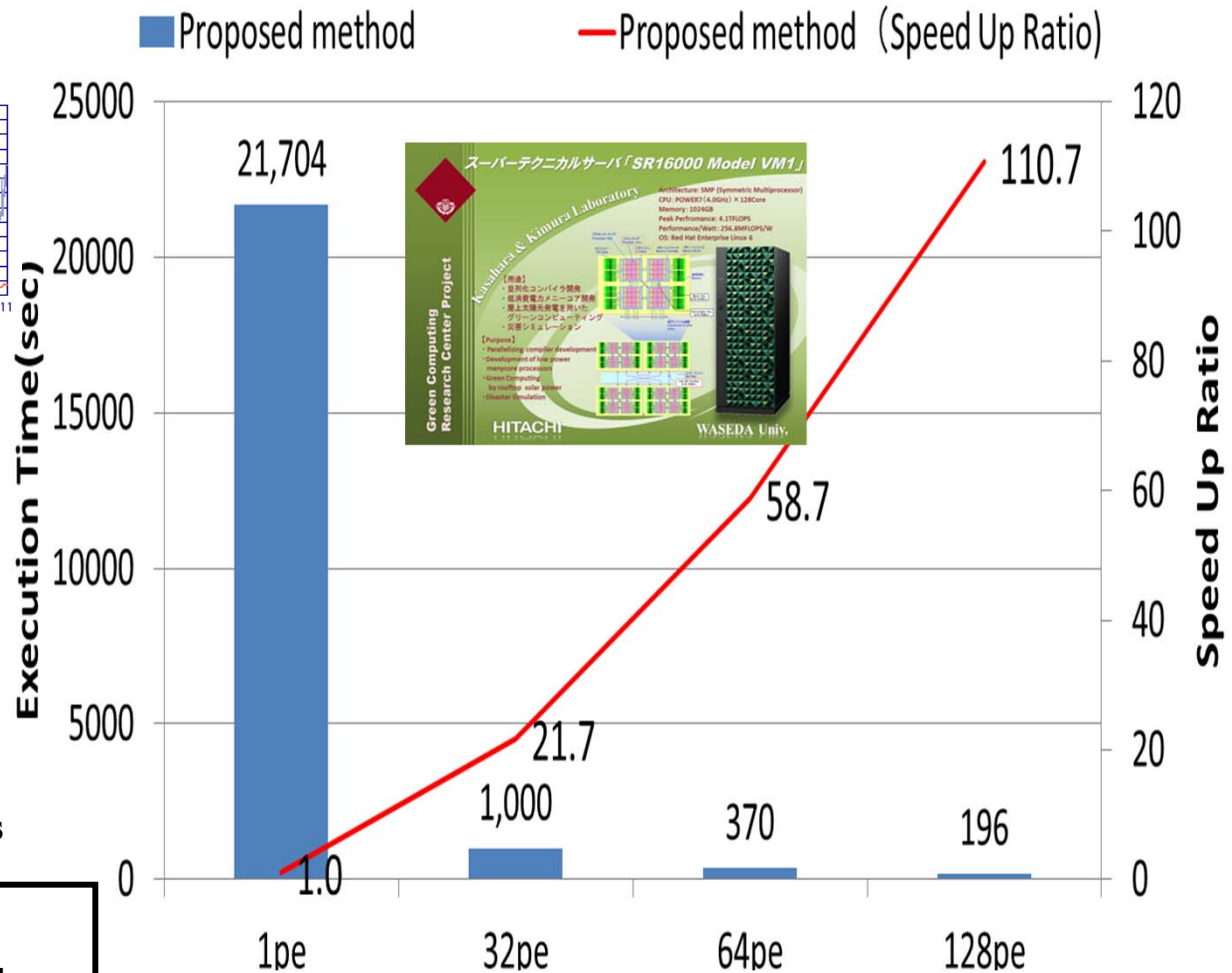


# 110 Times Speedup against the Sequential Processing for GMS Earthquake Wave Propagation Simulation on Hitachi SR16000

(Power7 Based 128 Core Linux SMP) [\(LCPC2015\)](#)

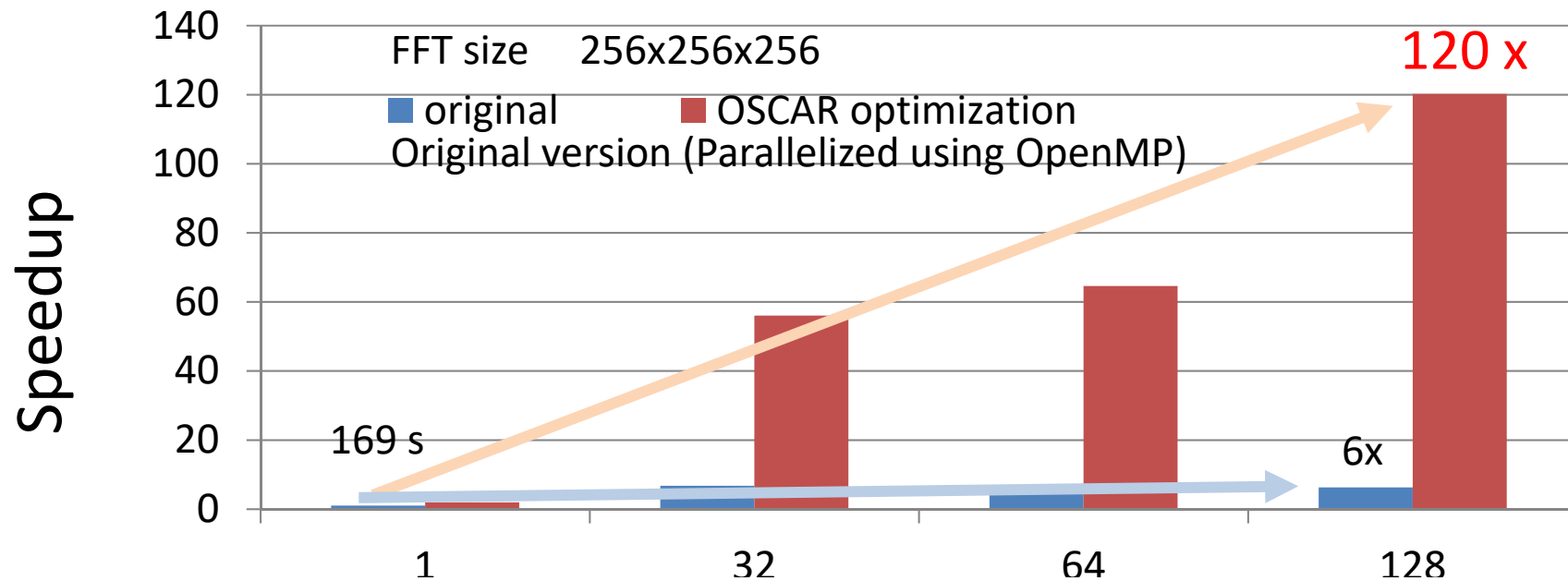


First touch for distributed shared memory and cache optimization over loops are important for scalable speedup





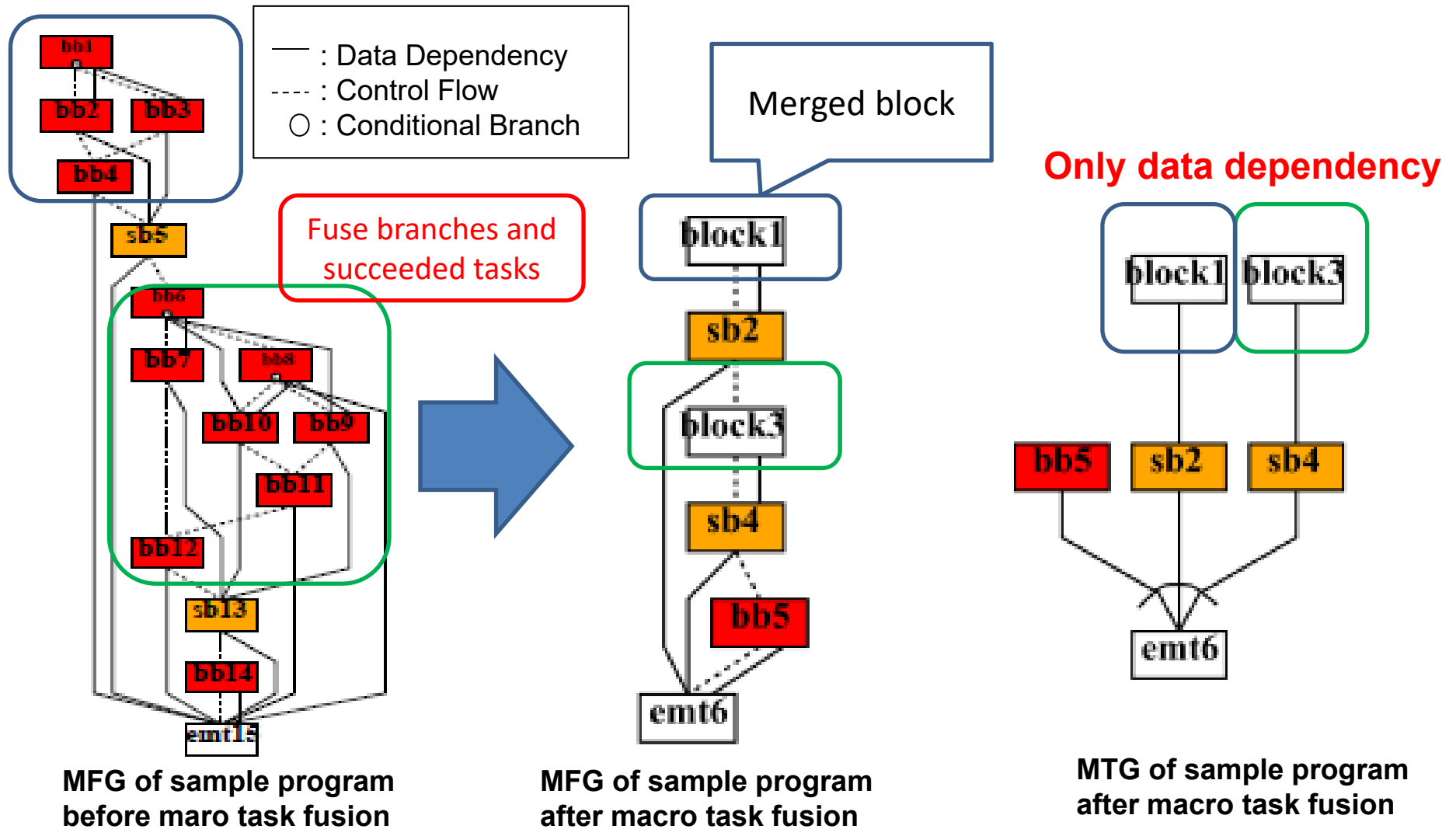
# Parallelization of 3D-FFT for New Magnetic Material Computation on Hitachi SR16000 Power7 CC-Numa Server



## OSCAR optimization

- reducing number of data transpose with interchange, code motion and loop fusion

# Parallel Processing of Automotive Engine Control: ECU Macro Task Fusion for Static Task Scheduling

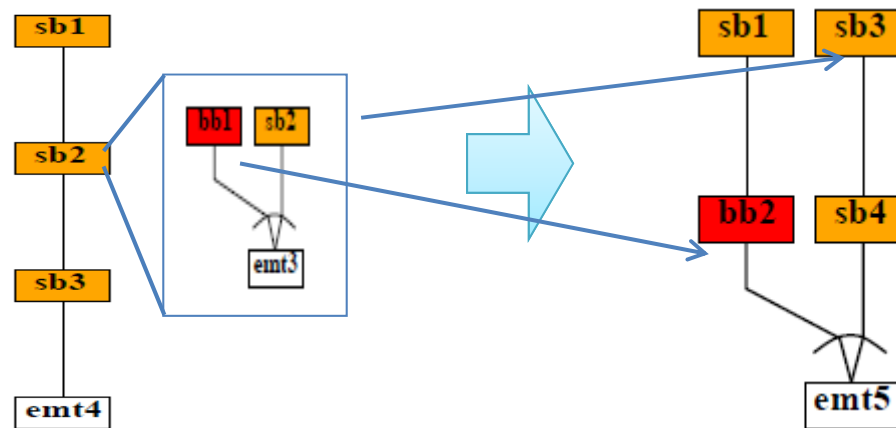


# ECU

## 3.1 Restructuring : Inline Expansion

- Inline expansion is effective
  - To increase coarse grain parallelism
- Expands functions having inner parallelism

**Improves coarse grain parallelism**



**MTG before inline expansion**

**MTG after inline expansion**

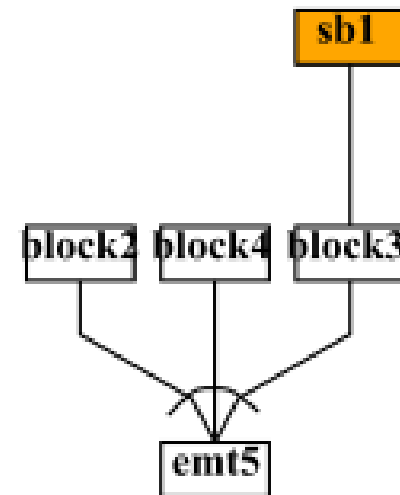
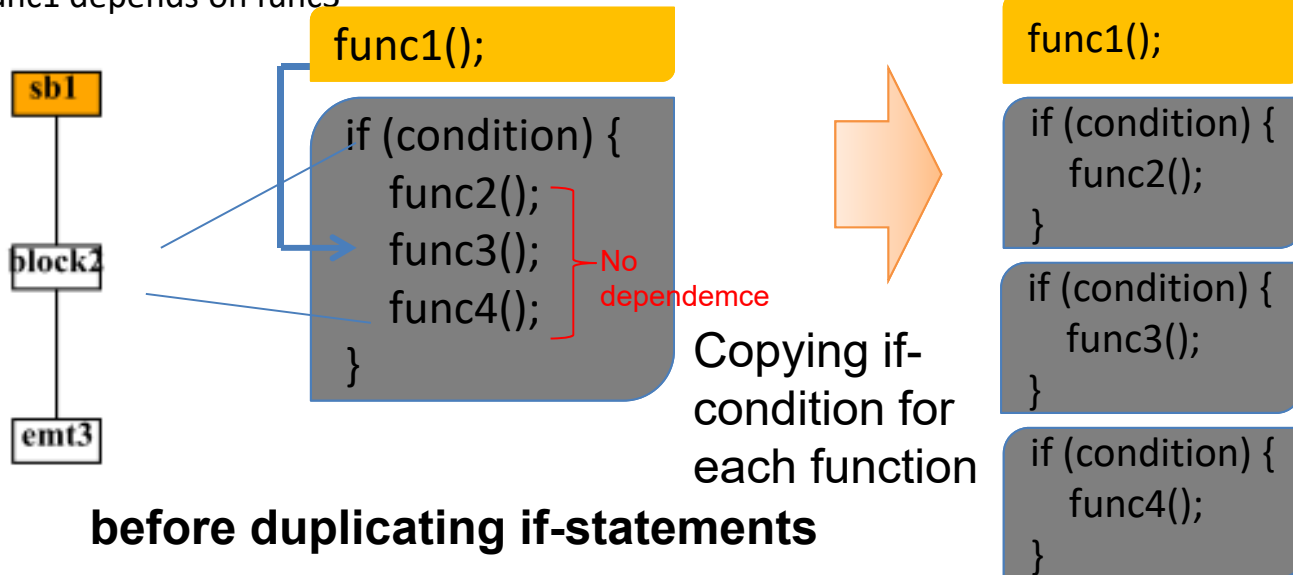
# ECU

## 3.2 Restructuring: Duplicating If-statements

- Duplicating if-statements is effective
  - To increase coarse grain parallelism
- Duplicates fused tasks having inner parallelism

**Improves coarse grain parallelism**

Func1 depends on func3



**after duplicating if-statements**

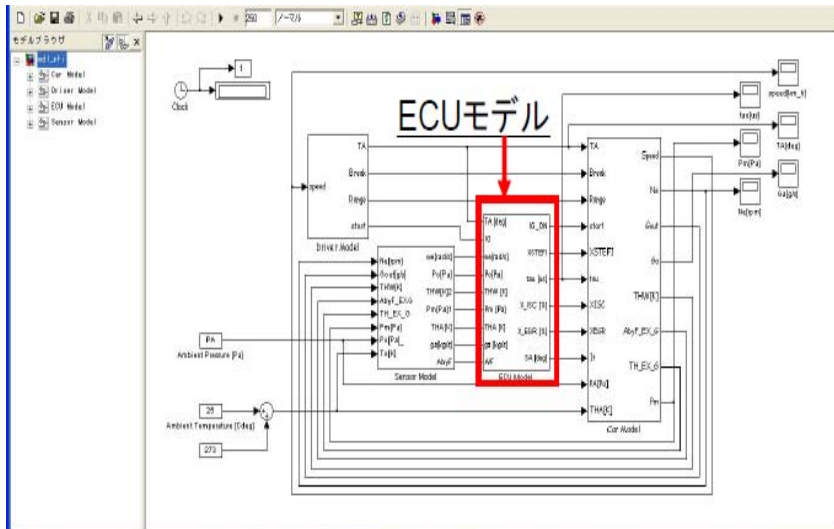
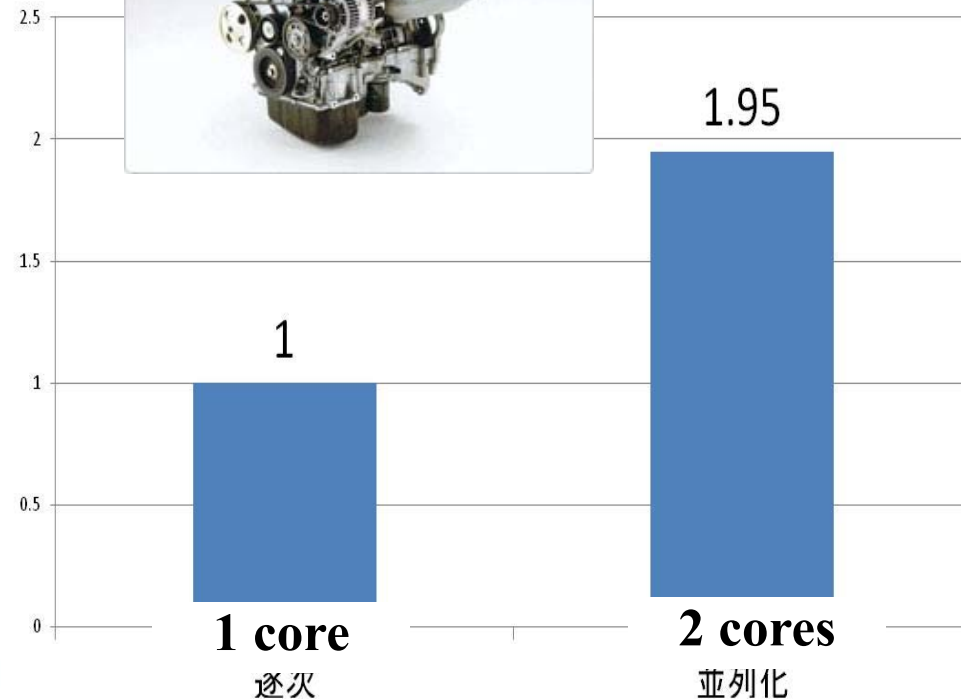


# Engine Control by multicore with Denso

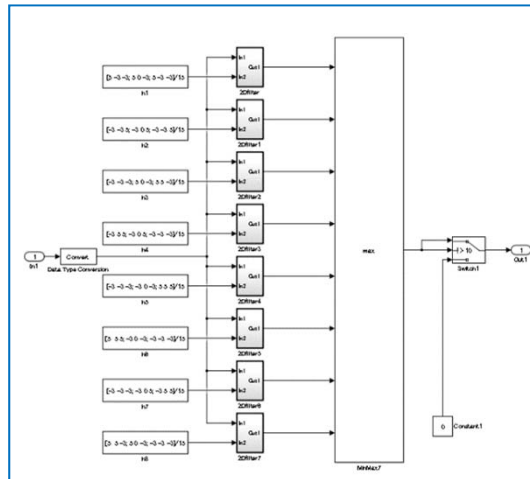
Though so far parallel processing of the engine control on multicore has been very difficult, Denso and Waseda succeeded 1.95 times speedup on 2core V850 multicore processor.



- Hard real-time automobile engine control by multicore using local memories
- Millions of lines C codes consisting conditional branches and basic blocks

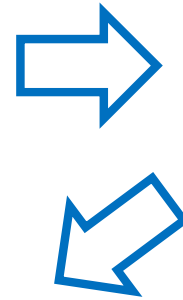


# OSCAR Compile Flow for Simulink Applications



Simulink model

Generate C code  
using Embedded Coder



```

/* Model step function */
void VesselExtraction_step(void)
{
    int32_T i;
    real_T u0;

    /* DataTypeConversion: '<S1>/Data Type Conversion' incorporates:
     * Import: '<Root>/In1'
     */
    for (i = 0; i < 16384; i++) {
        VesselExtraction_B.DataTypeConversion[i] = VesselExtraction_U.In1[i];
    }
    /* End of DataTypeConversion: '<S1>/Data Type Conversion' */

    /* Outputs for Atomic SubSystem: '<S1>/2Dfilter' */
    /* Constant: '<S1>/h1' */
    VesselExtraction_Dfilter(VesselExtraction_B.DataTypeConversion,
        VesselExtraction_P.h1_Value, &VesselExtraction_B.Dfilter,
        (P_Dfilter_VesselExtraction_T *)&VesselExtraction_P.Dfilter);
    /* End of Outputs for SubSystem: '<S1>/2Dfilter' */

    /* Outputs for Atomic SubSystem: '<S1>/2Dfilter1' */
    /* Constant: '<S1>/h2' */
    VesselExtraction_Dfilter(VesselExtraction_B.DataTypeConversion,
        VesselExtraction_P.h2_Value, &VesselExtraction_B.Dfilter1,
        (P_Dfilter_VesselExtraction_T *)&VesselExtraction_P.Dfilter1);
}
    
```

C code

## OSCAR Compiler

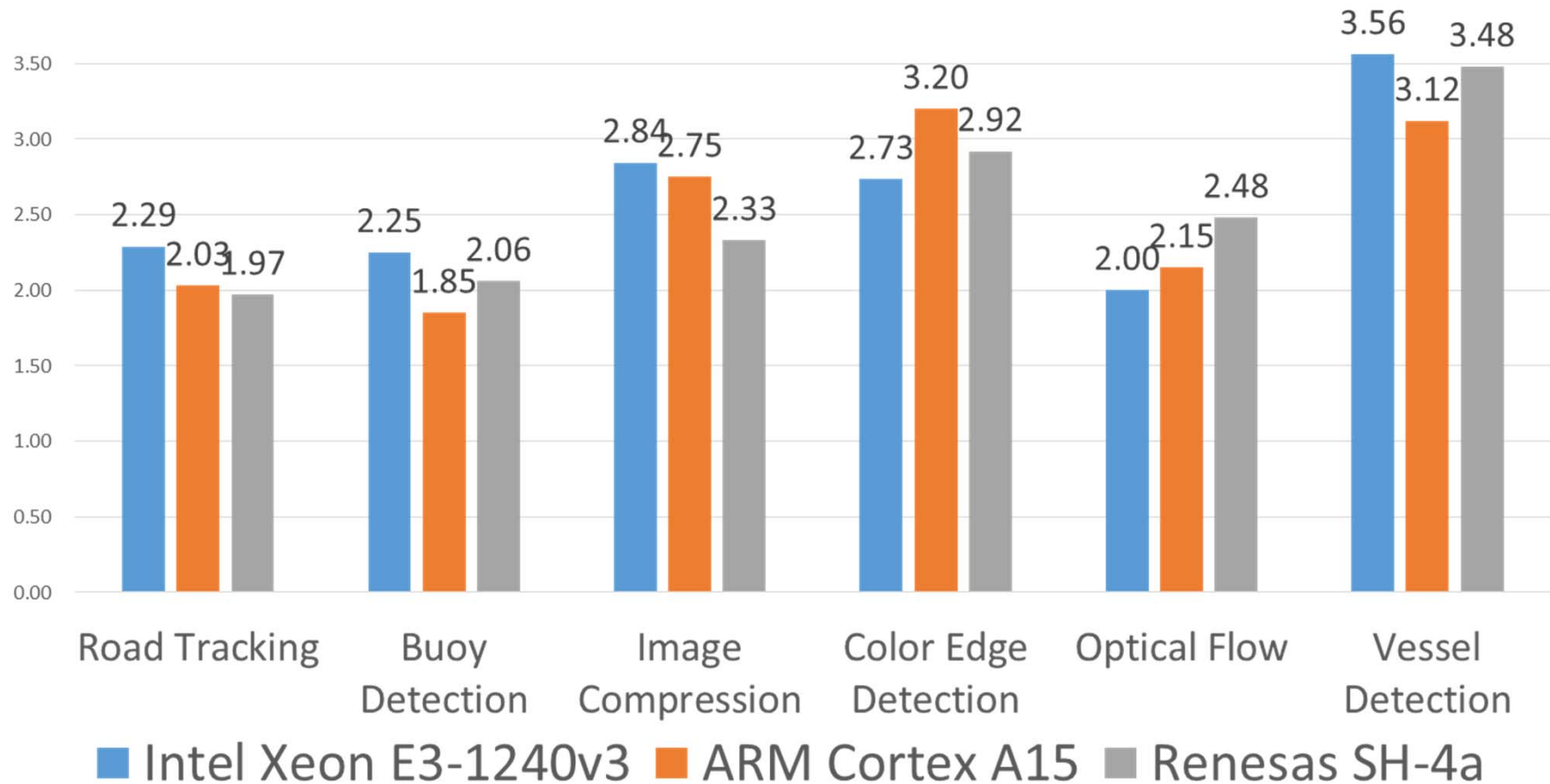
**(1) Generate MTG**  
→ Parallelism

**(2) Generate gantt chart**  
→ Scheduling in a multicore

**(3) Generate parallelized C code**  
→ Multiplatform execution  
(Intel, ARM and SH etc)

# Speedups of MATLAB/Simulink Image Processing on Various 4core Multicores

(Intel Xeon, ARM Cortex A15 and Renesas SH4A)



Road Tracking, Image Compression : <http://www.mathworks.co.jp/jp/help/vision/examples>

Buoy Detection : <http://www.mathworks.co.jp/matlabcentral/fileexchange/44706-buoy-detection-using-simulink>

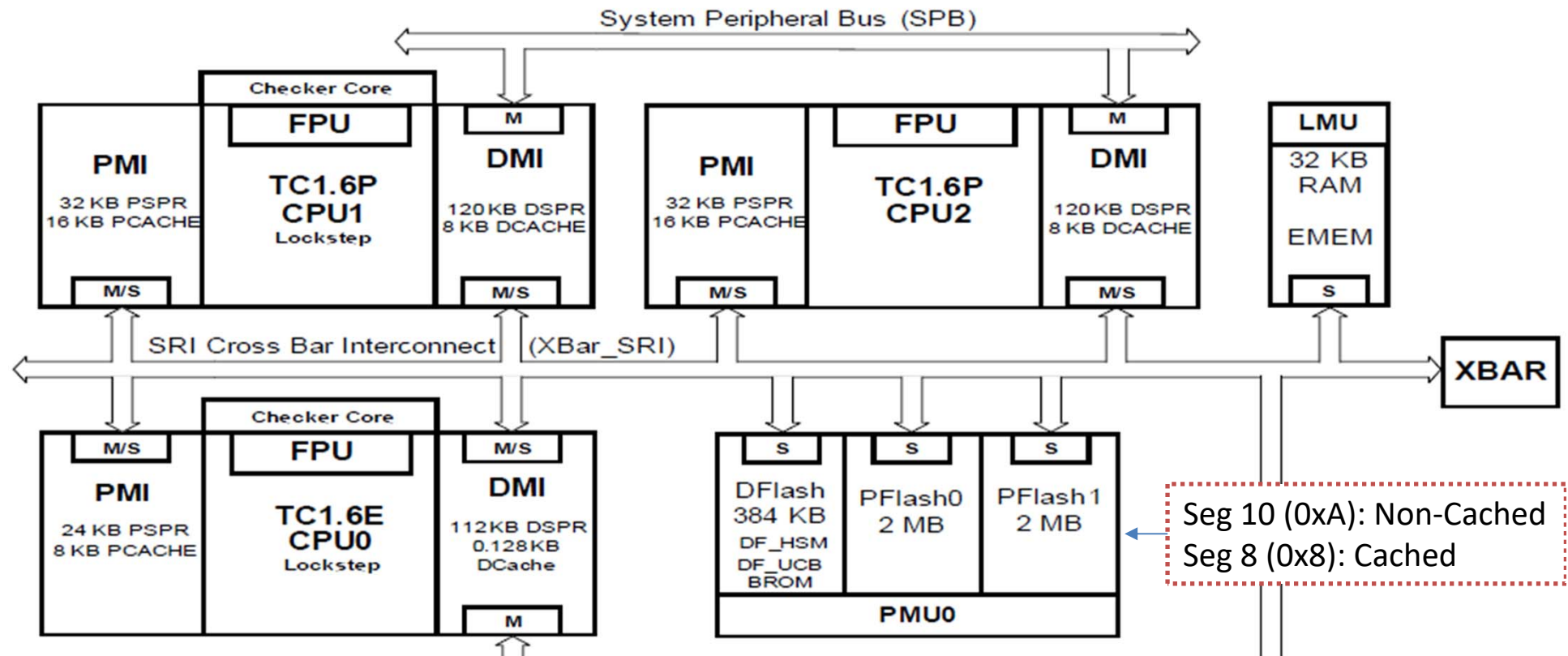
Color Edge Detection : <http://www.mathworks.co.jp/matlabcentral/fileexchange/28114-fast-edges-of-a-color-image--actual-color--not-converting-to-grayscale-/>

Vessel Detection : <http://www.mathworks.co.jp/matlabcentral/fileexchange/24990-retinal-blood-vessel-extraction/>

# Infineon AURIX TC277

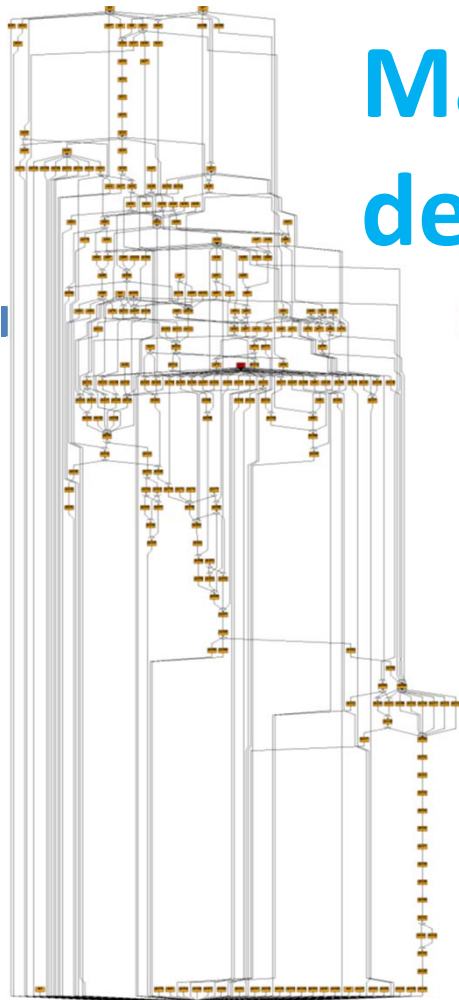
## Abbreviations :

PCACHE:	Program Cache
DCACHE:	Data Cache
DSPR:	Data Scratch-Pad RAM
PSPR:	Program Scratch-Pad RAM
BROM:	Boot ROM
PFlash:	Program Flash
DFlash:	Data Flash (EEPROM)
<b>S</b> :	SRI Slave Interface
<b>M</b> :	SRI Master Interface

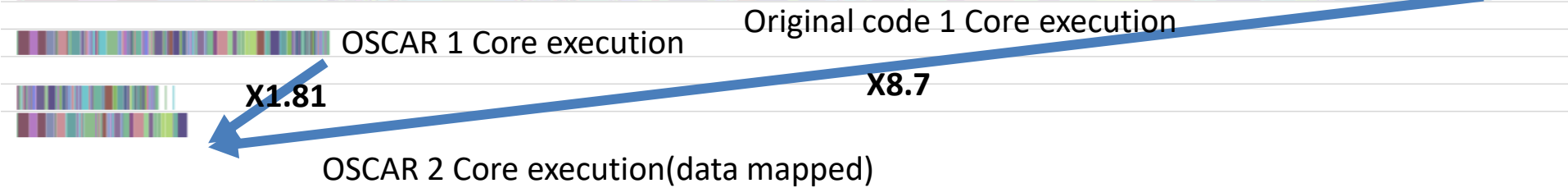
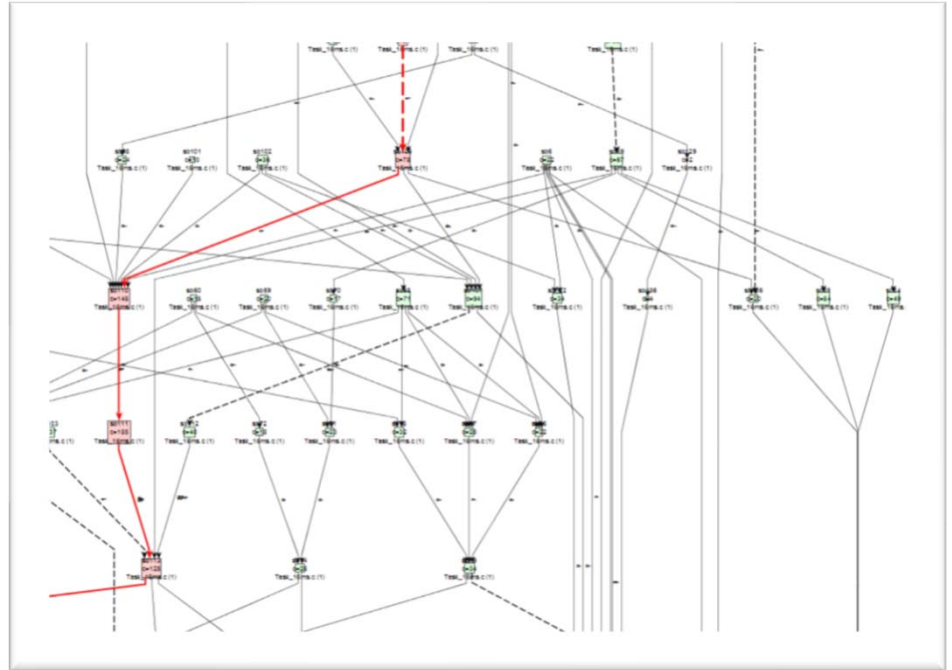




# Macrotask Graph, Dependence details and schedules

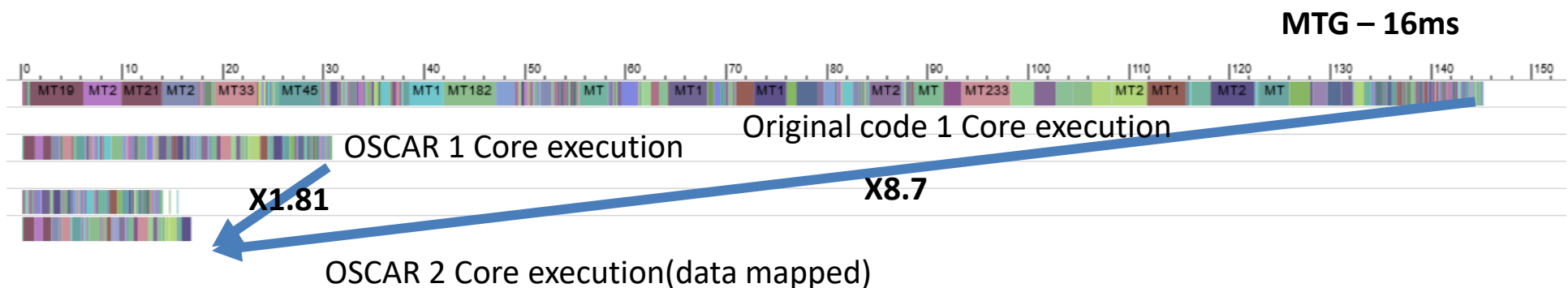


MTG – task\_16ms

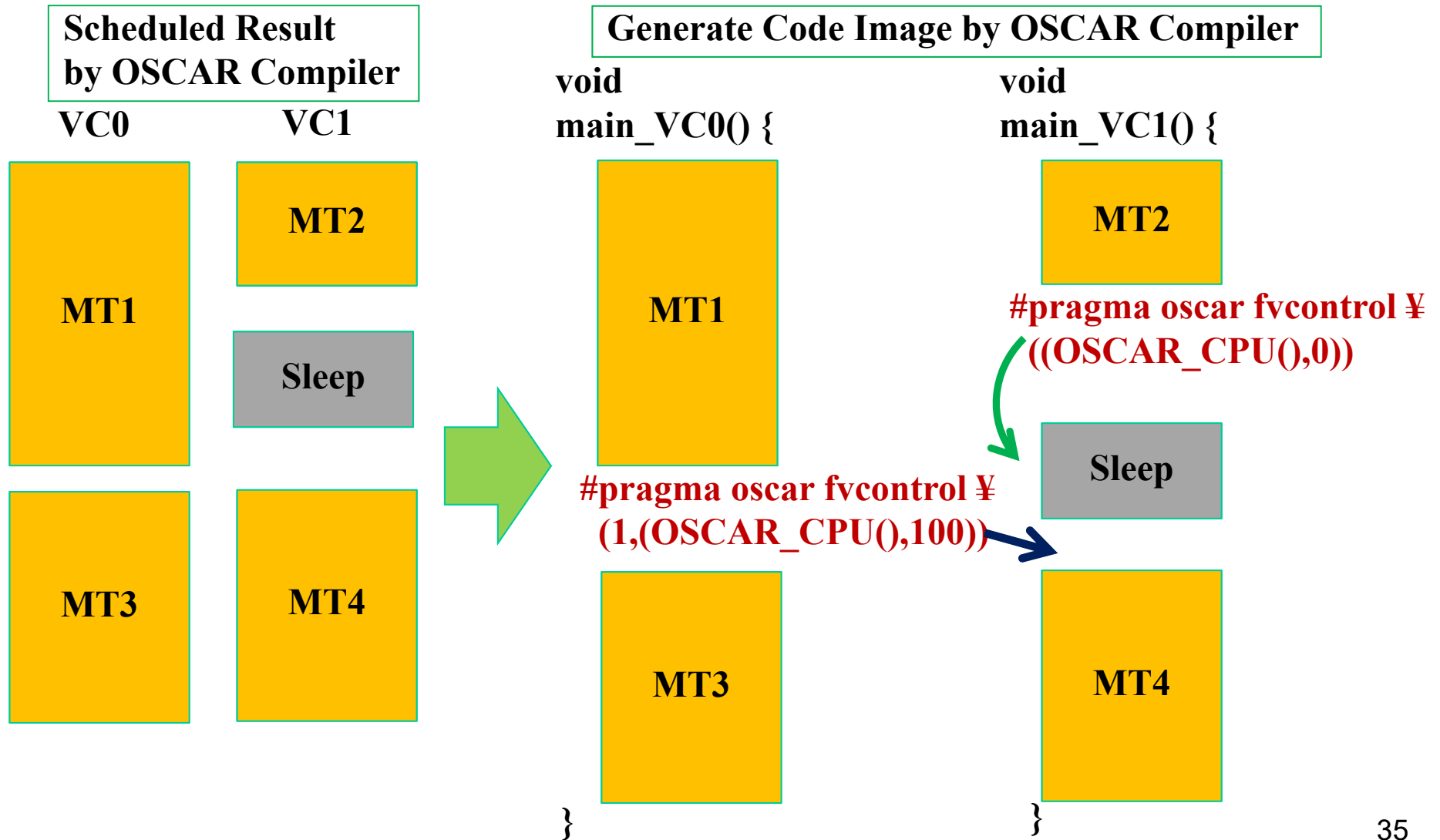


# Automatic Parallelization of an Engine Control C Program with 400 thousands lines on AUTOSAR on 2 cores of Infineon AURIX TC277

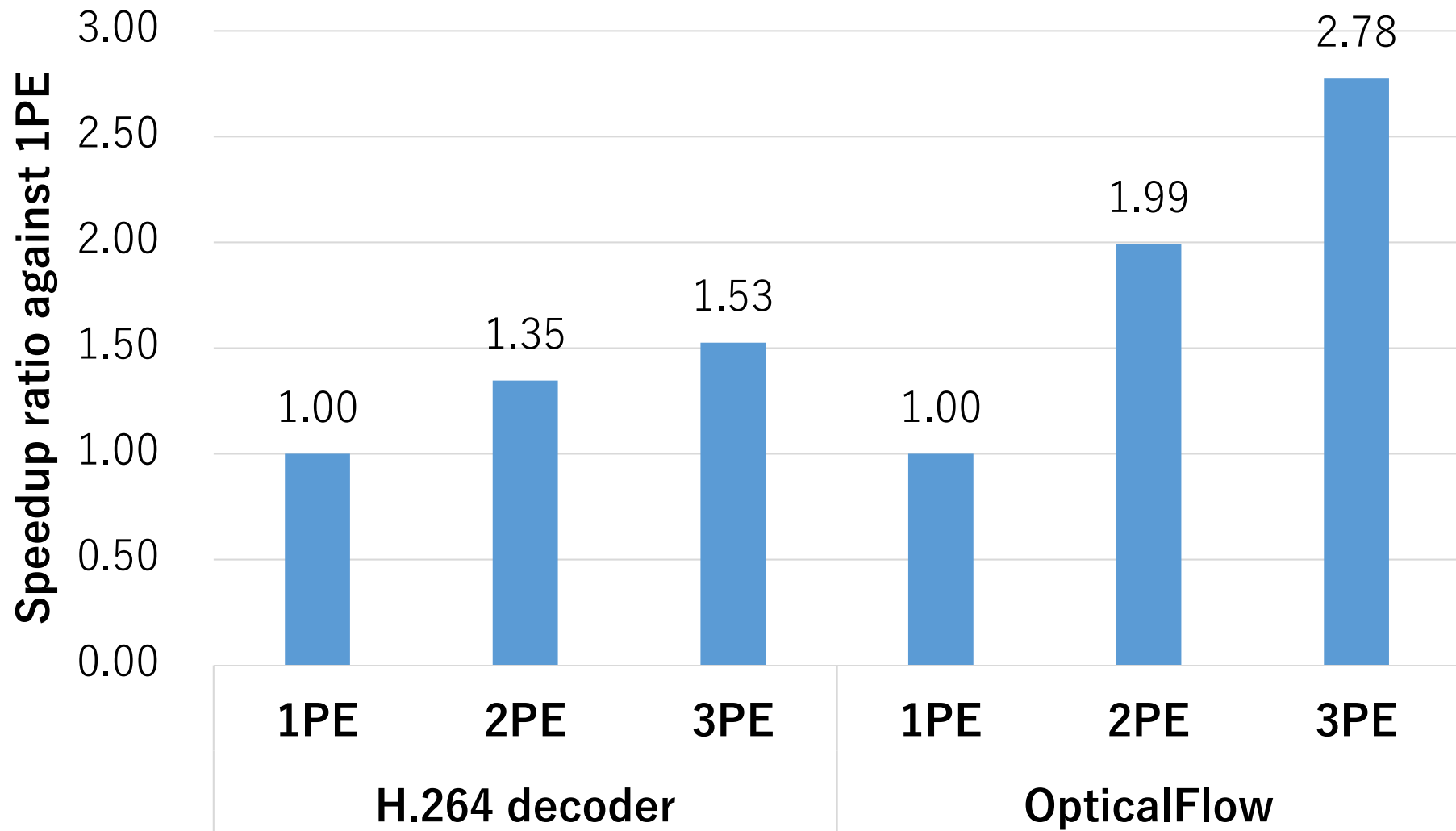
- **Original sequential** execution time on 1 core: **145500** cycles
- **Sequential execution time by OSCAR** on 1 core: **29700** cycles
  - **4.9 times speedup on 1 core** against original execution by OSCAR Compilers automatic data allocation for local scratch pad memory, flush memory modules
- **2 core execution by OSCAR** Compiler: **16400** cycles
  - **1.81 times speedup with 2 core** against **1 core execution with OSCAR Compiler**
  - **8.7 times speedup against original sequential execution.**



# Low-Power Optimization with OSCAR API



# Speedup ratio for H.264 and Optical Flow on ARM Cortex-A9 Android 3 cores by OSCAR Automatic Parallelization

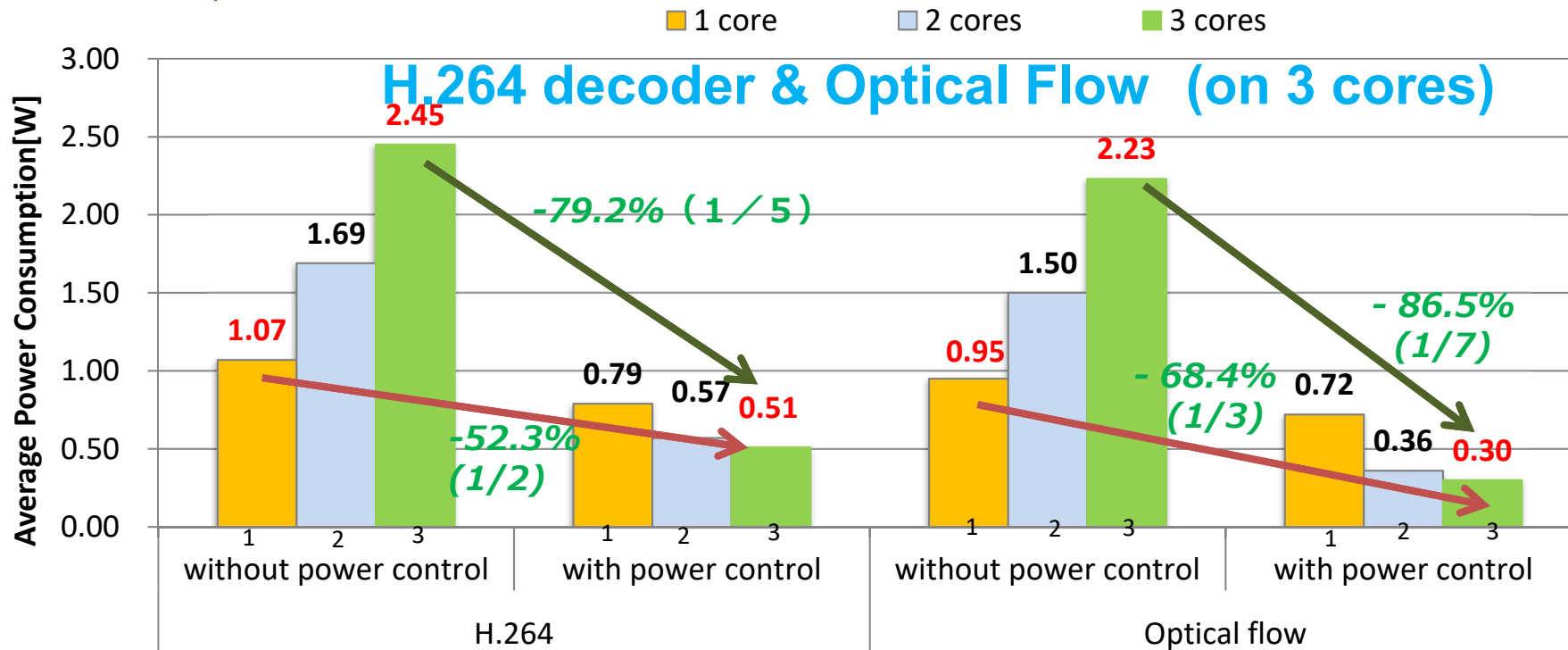
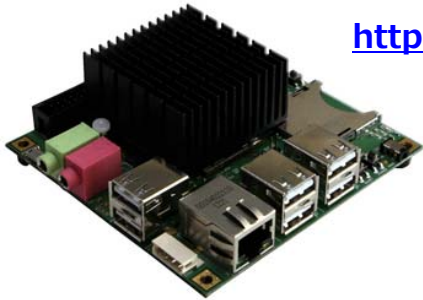


# Automatic Power Reduction on ARM CortexA9 with Android

[http://www.youtube.com/channel/UCS43INYEIkC8i\\_KIgfZYQBQ](http://www.youtube.com/channel/UCS43INYEIkC8i_KIgfZYQBQ)

ODROID X2

Samsung Exynos4412 Prime, ARM Cortex-A9 Quad core  
1.7GHz~0.2GHz, used by Samsung's Galaxy S3

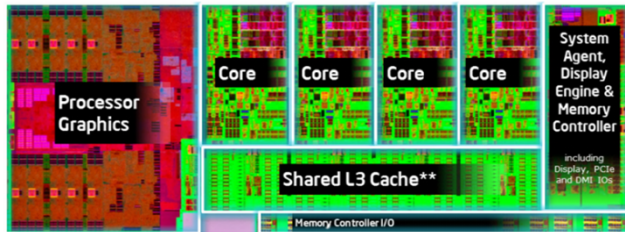


Power for 3cores was reduced to **1/5~1/7** against without software power control

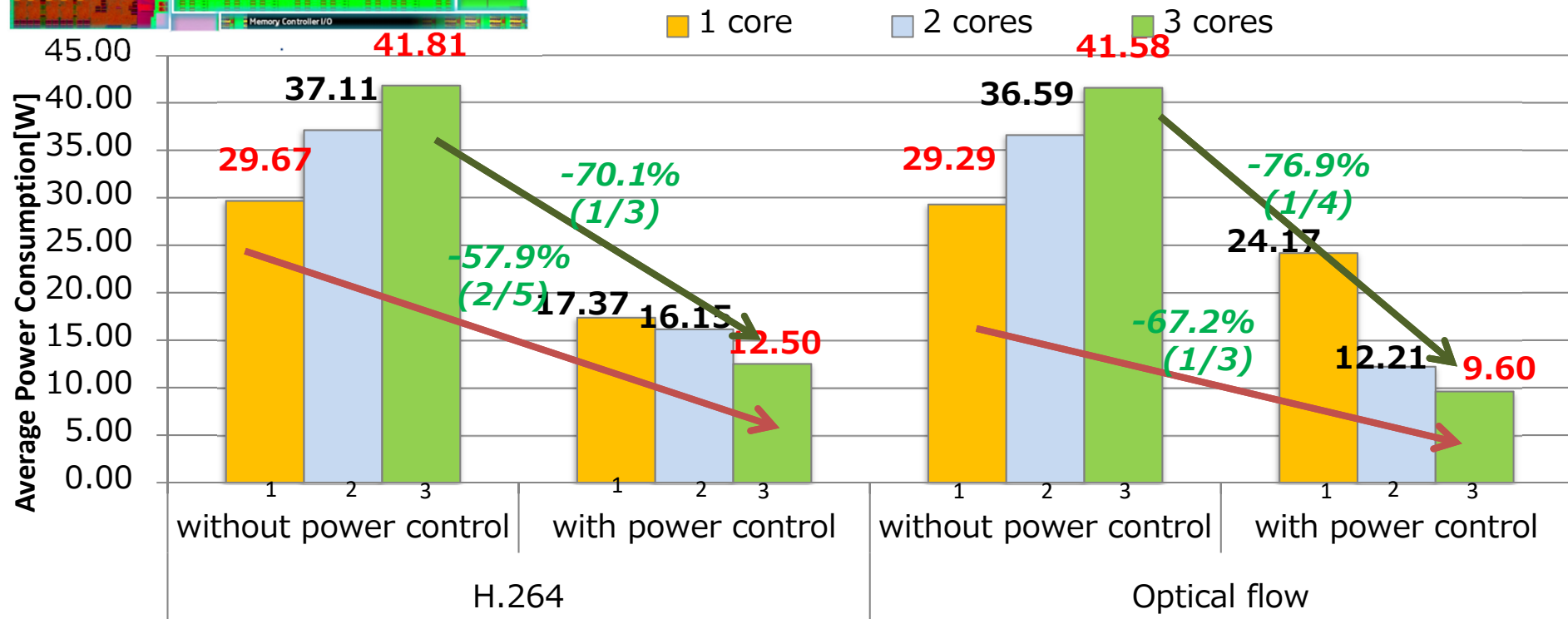
Power for 3cores was reduced to **1/2~1/3** against ordinary 1core execution

# Automatic Power Reuction on Intel Haswell

## H.264 decoder & Optical Flow (3cores)



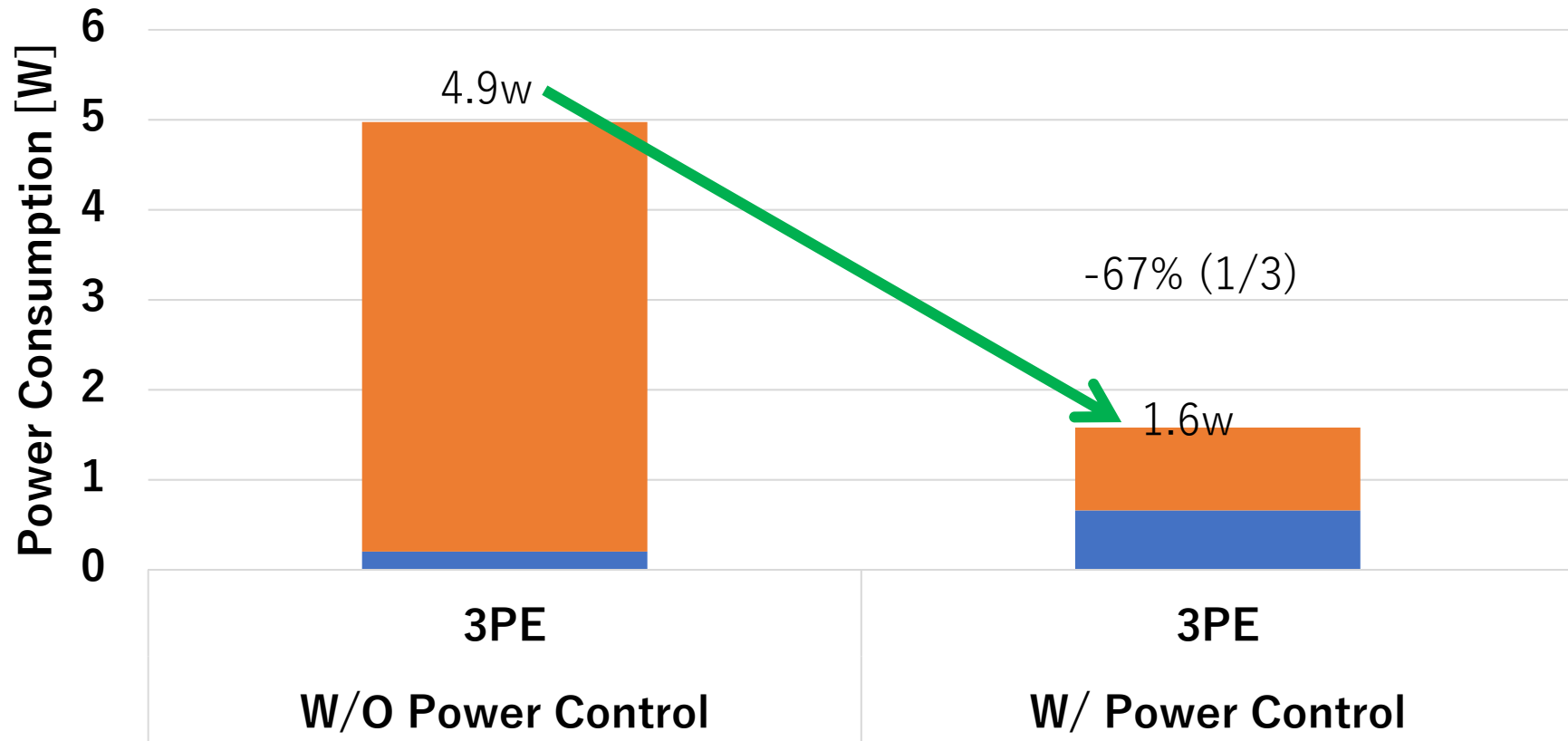
H81M-A, Intel Core i7 4770k  
Quad core, 3.5GHz~0.8GHz



Power for 3cores was reduced to **1/3~1/4** against without software power control

Power for 3cores was reduced to **2/5~1/3** against ordinary 1core execution

# Automatic Power Reduction of OpenCV Face Detection on big.LITTLE ARM Processor



- **ODROID-XU3**

■ Cortex-A7 ■ Cortex-A15

- **Samsung Exynos 5422 Processor**

- 4x Cortex-A15 2.0GHz, 4x Cortex-A7 1.4GHz big.LITTLE Architecture
- 2GB LPDDR3 RAM
- Frequency can be changed by each cluster unit

# OSCAR API Ver. 2.0 for Homogeneous/Heterogeneous Multicores and Manycores (LCPC2009Homo, 2010 Hetero)

## 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

- ▶ **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

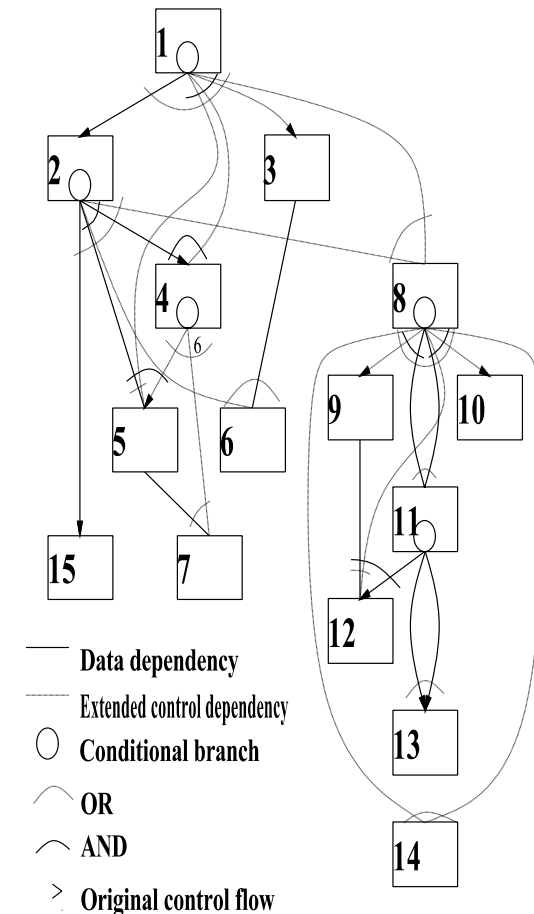
from V2.0

(\* from OpenMP)



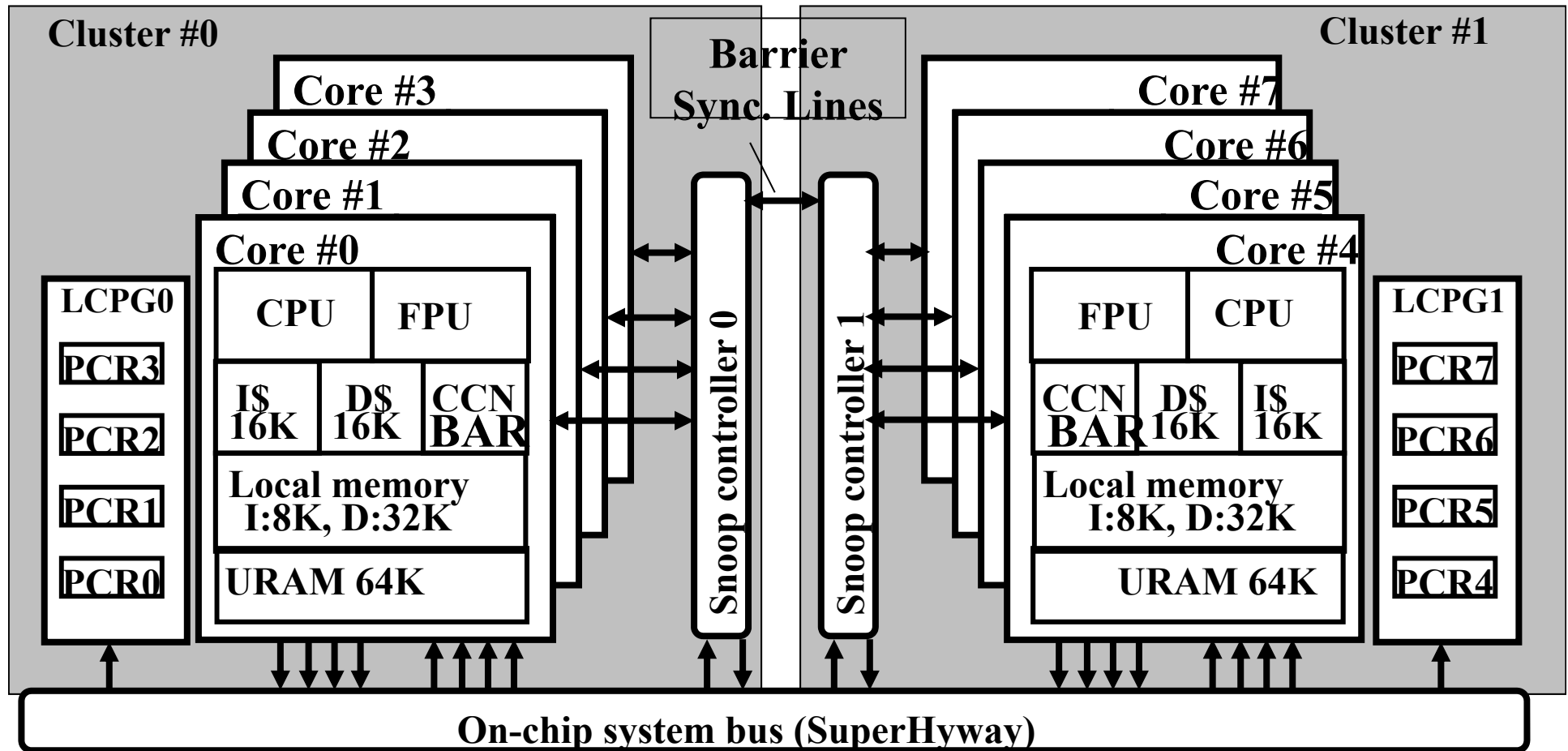
# Software Coherence Control Method on OSCAR Parallelizing Compiler

- Coarse grain task parallelization with **earliest condition analysis** (control and data dependency analysis to detect parallelism among coarse grain tasks).
- OSCAR compiler automatically controls coherence using following simple program restructuring methods:
  - To cope with stale data problems:
    - ◆ **Data synchronization by compilers**
  - To cope with false sharing problem:
    - ◆ **Data Alignment**
    - ◆ **Array Padding**
    - ◆ **Non-cacheable Buffer**



**MTG generated by  
earliest executable  
condition analysis**

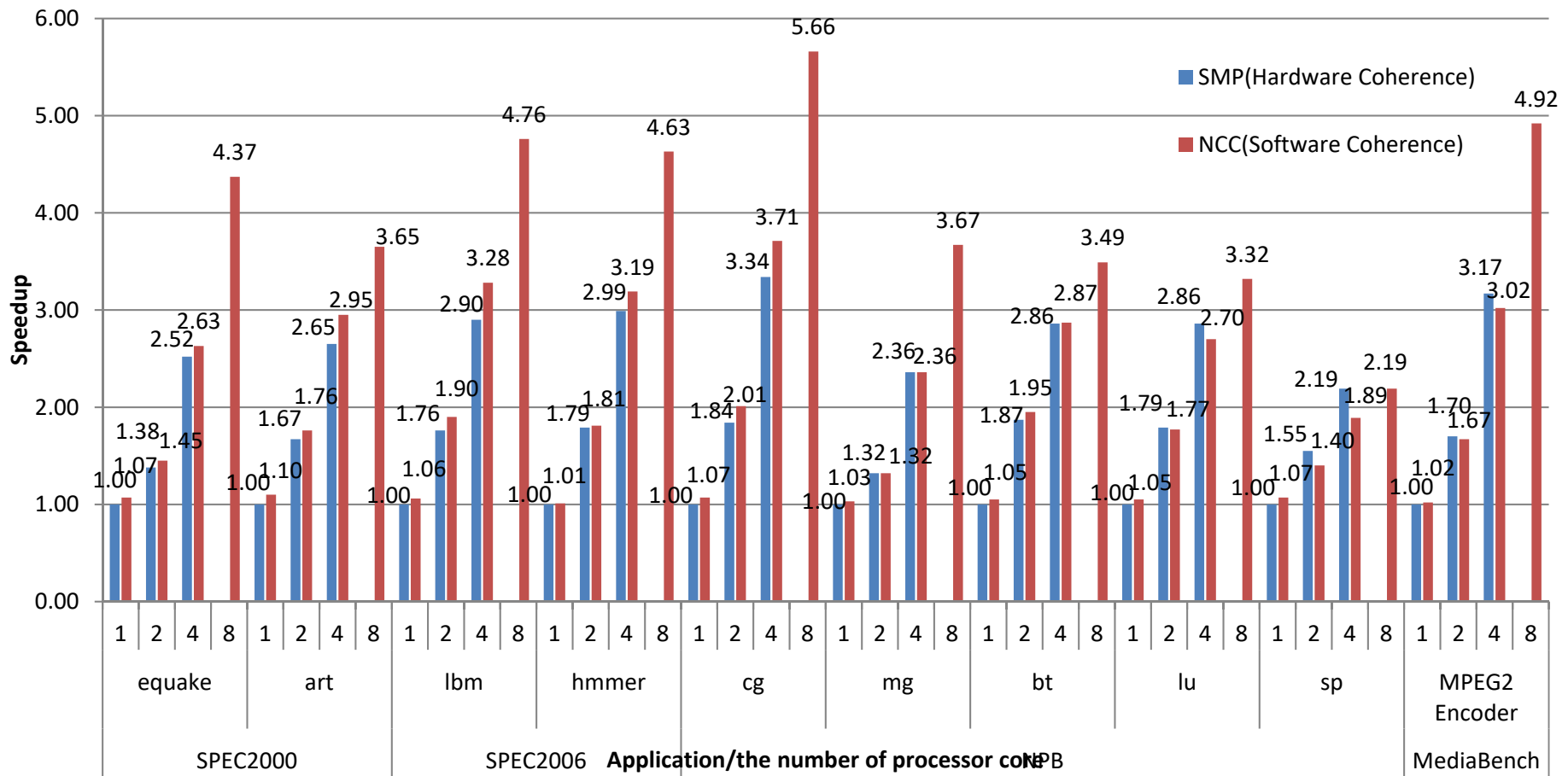
# 8 Core RP2 Chip Block Diagram



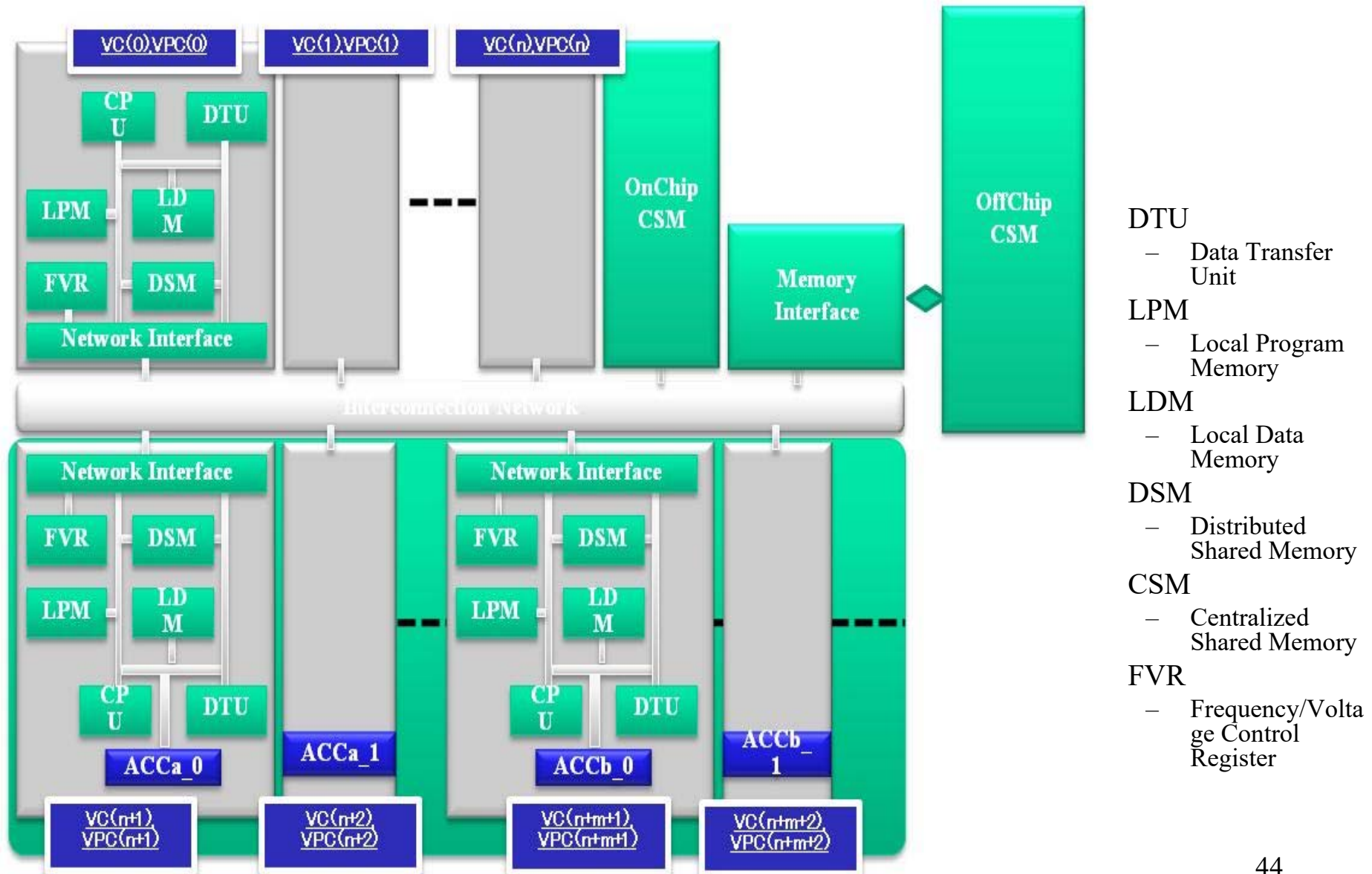
**LCPG:** Local clock pulse generator  
**PCR:** Power Control Register  
**CCN/BAR:** Cache controller/Barrier Register  
**URAM:** User RAM (Distributed Shared Memory)

# Automatic Software Coherent Control for Manycores

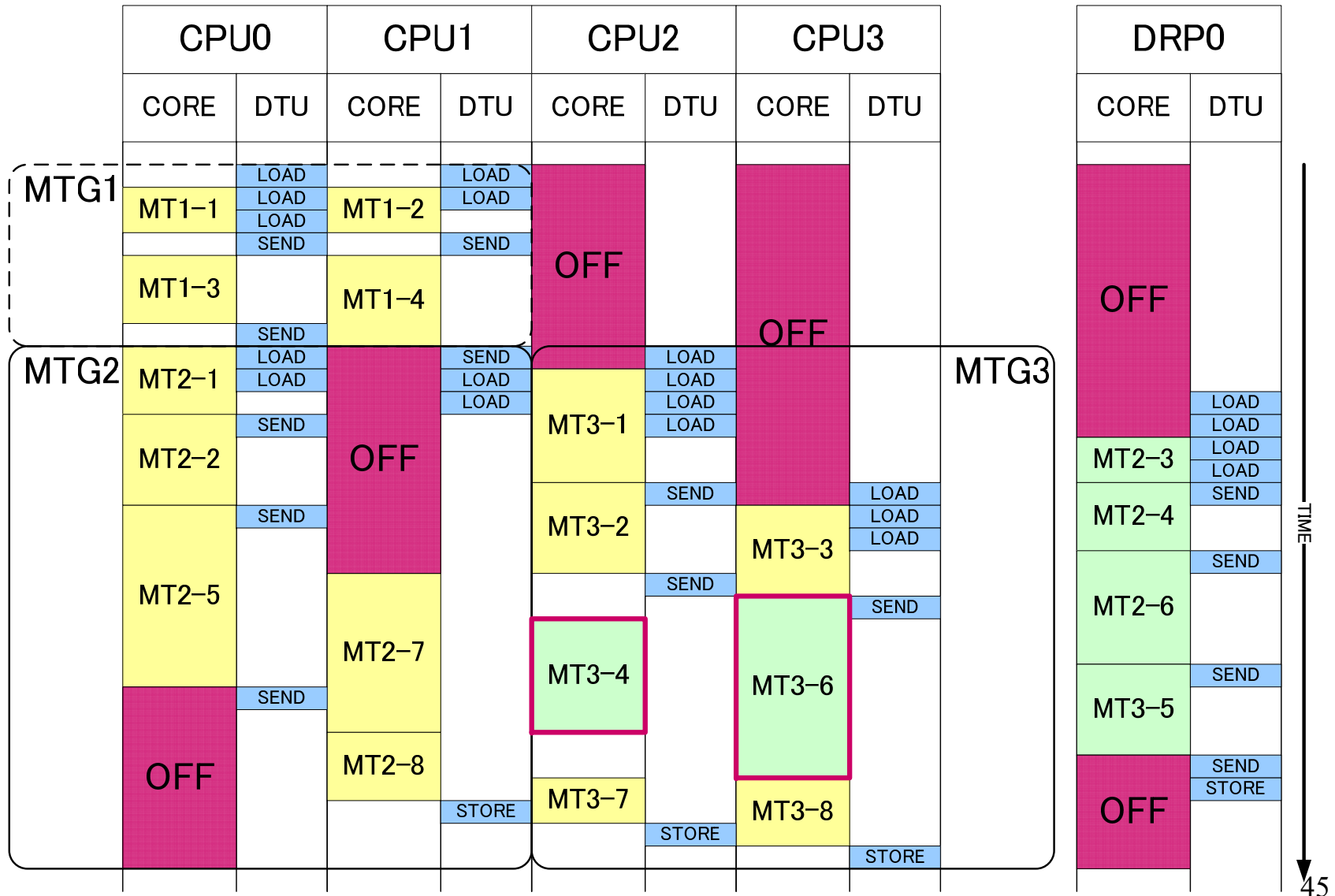
## Performance of Software Coherence Control by OSCAR Compiler on 8-core RP2



# OSCAR Heterogeneous Multicore

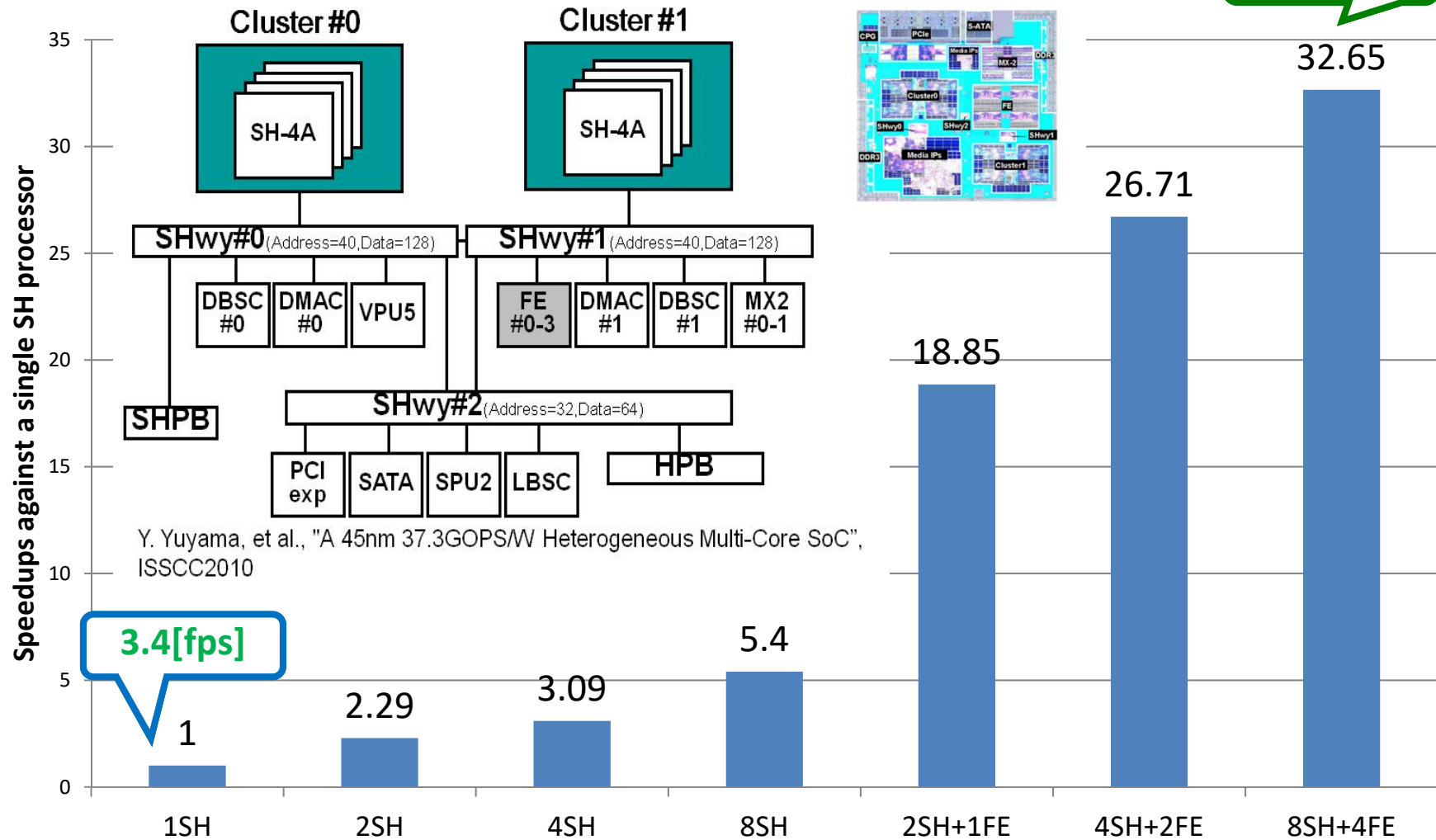


# An Image of Static Schedule for Heterogeneous Multi-core with Data Transfer Overlapping and Power Control



# 33 Times Speedup Using OSCAR Compiler and OSCAR API on RP-X (Optical Flow with a hand-tuned library)

111[fps]



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

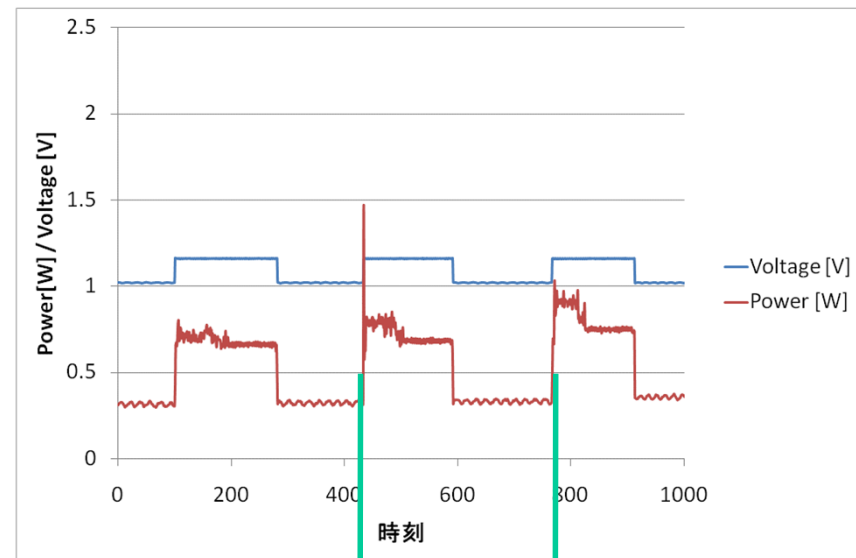
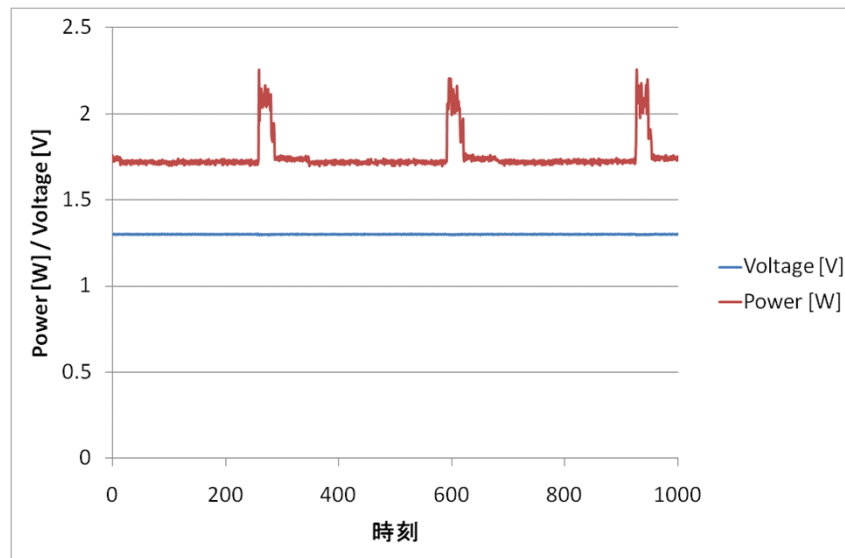
**Without Power Reduction**

**With Power Reduction by OSCAR Compiler**  
**70% of power reduction**

**Average: 1.76[W]**



**Average: 0.54[W]**



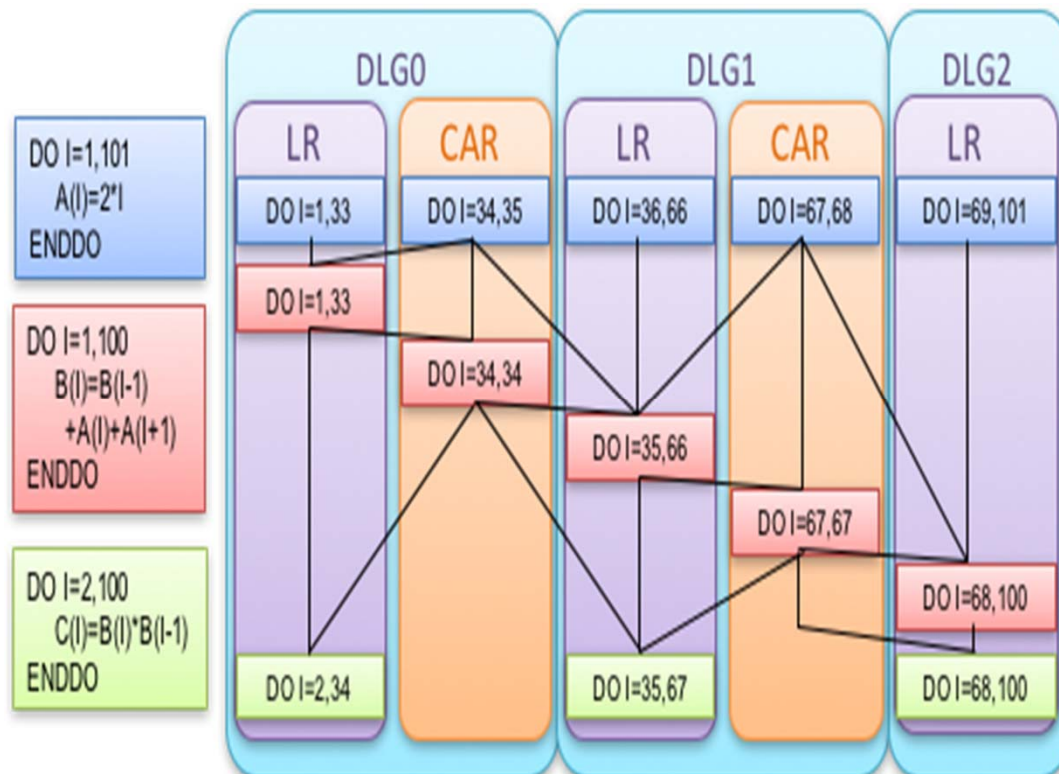
**1cycle : 33[ms]  
→30[fps]**

# Automatic Local Memory Management

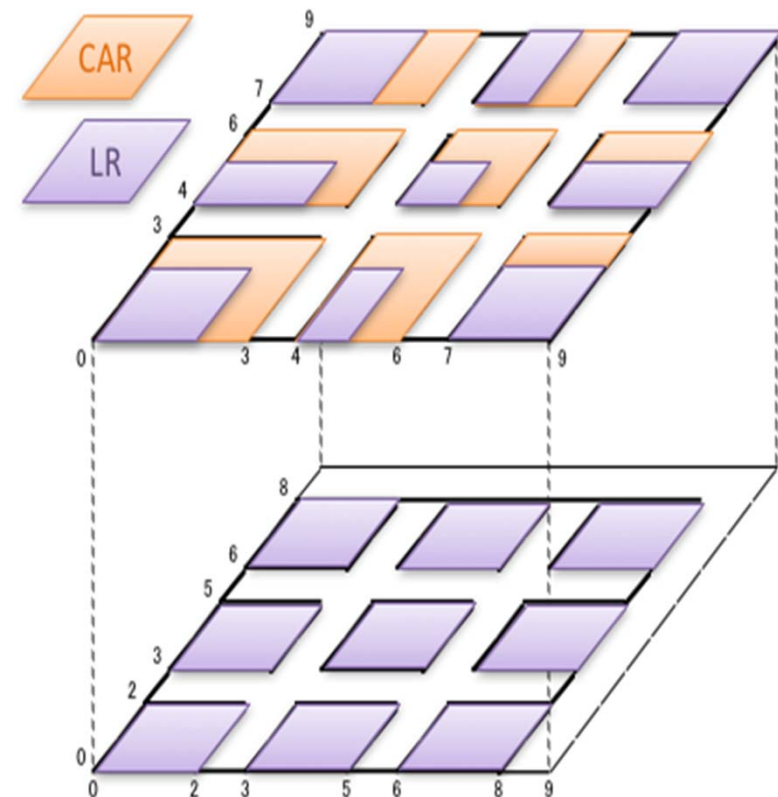
## Data Localization: Loop Aligned Decomposition

- Decomposed loop into LR and CARs
  - LR (Localizable Region): Data can be passed through LDM
  - CAR (Commonly Accessed Region): Data transfers are required among processors

### Single dimension Decomposition



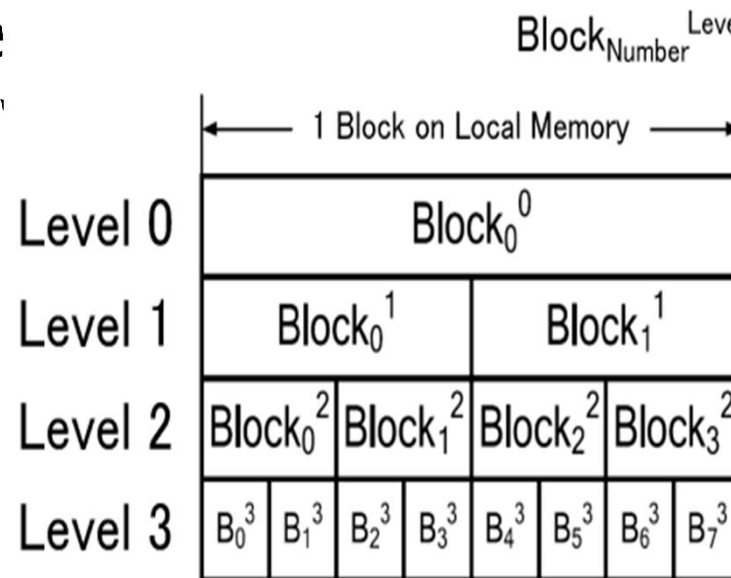
### Multi-dimension Decomposition





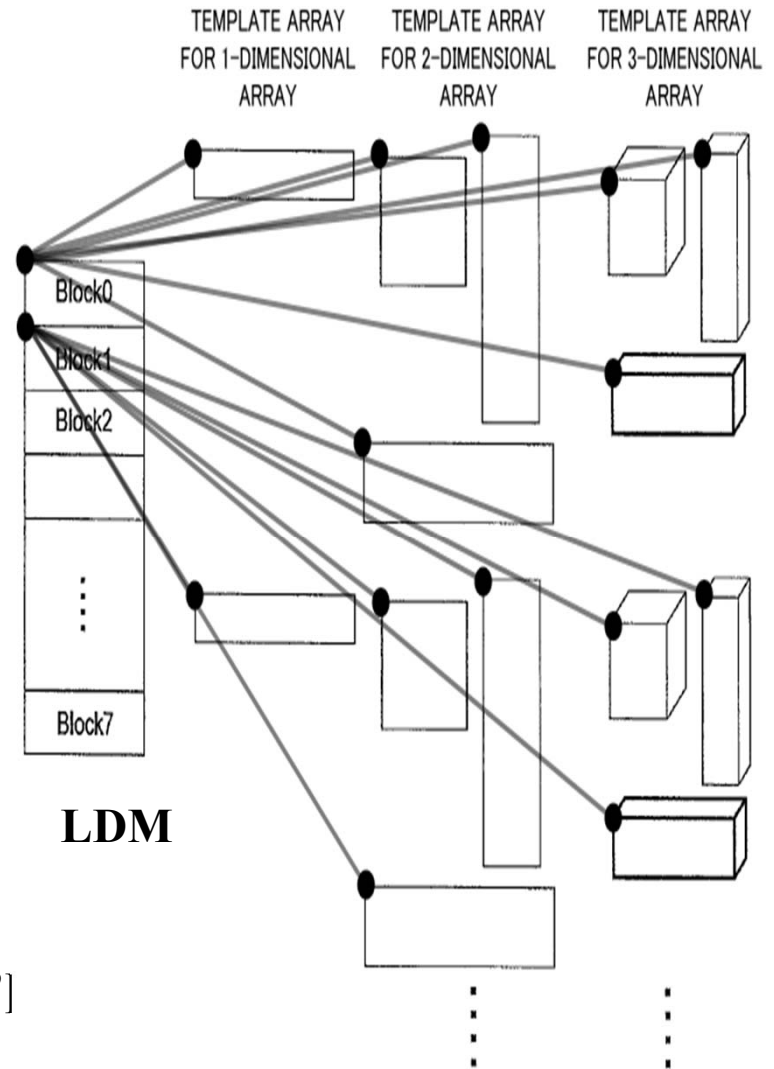
# Adjustable Blocks

- Handling a suitable block size for each application
  - different from a fixed block size in cache
  - each block can be divided into smaller blocks with integer and scalar



# Multi-dimensional Template Arrays for Improving Readability

- a mapping technique for arrays with varying dimensions
  - each block on LDM corresponds to multiple empty arrays with varying dimensions
  - these arrays have an additional dimension to store the corresponding block number
    - $TA[Block\#][\ ]$  for single dimension
    - $TA[Block\#][\ ][\ ]$  for double dimension
    - $TA[Block\#][\ ][\ ][\ ]$  for triple dimension
    - ...
- LDM are represented as a one dimensional array
  - without Template Arrays, multi-dimensional arrays have complex index calculations
    - $A[i][j][k] \rightarrow TA[offset + i' * L + j' * M + k']$
  - Template Arrays provide readability
    - $A[i][j][k] \rightarrow TA[Block\#][i'][j'][k']$



# Block Replacement Policy

## □ Compiler Control Memory block Replacement

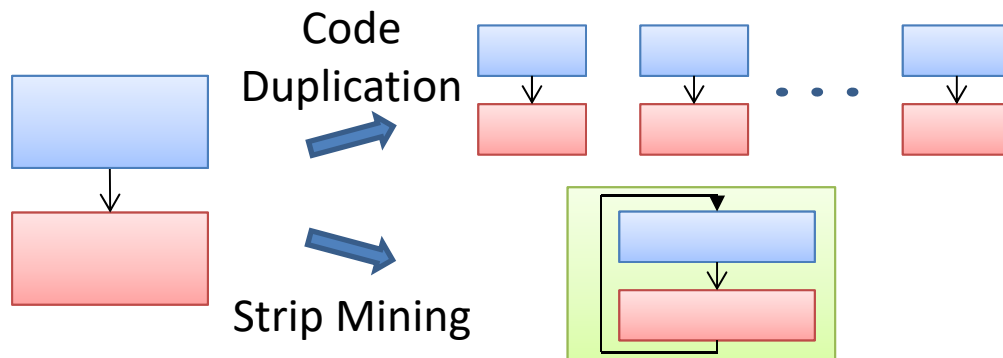
- using live, dead and reuse information of each variable from the scheduled result
- different from LRU in cache that does not use data dependence information

## □ Block Eviction Priority Policy

1. (Dead) Variables that will not be accessed later in the program
2. Variables that are accessed only by other processor cores
3. Variables that will be later accessed by the current processor core
4. Variables that will immediately be accessed by the current processor core

# Code Compaction by Strip Mining

- ❑ Previous approach produces duplicate code
  - generates multiple copies of the loop body which leads to code bloat
- ❑ Proposed method adopts code compaction
  - based on strip mining
  - multi-dimensional loop can be restructured



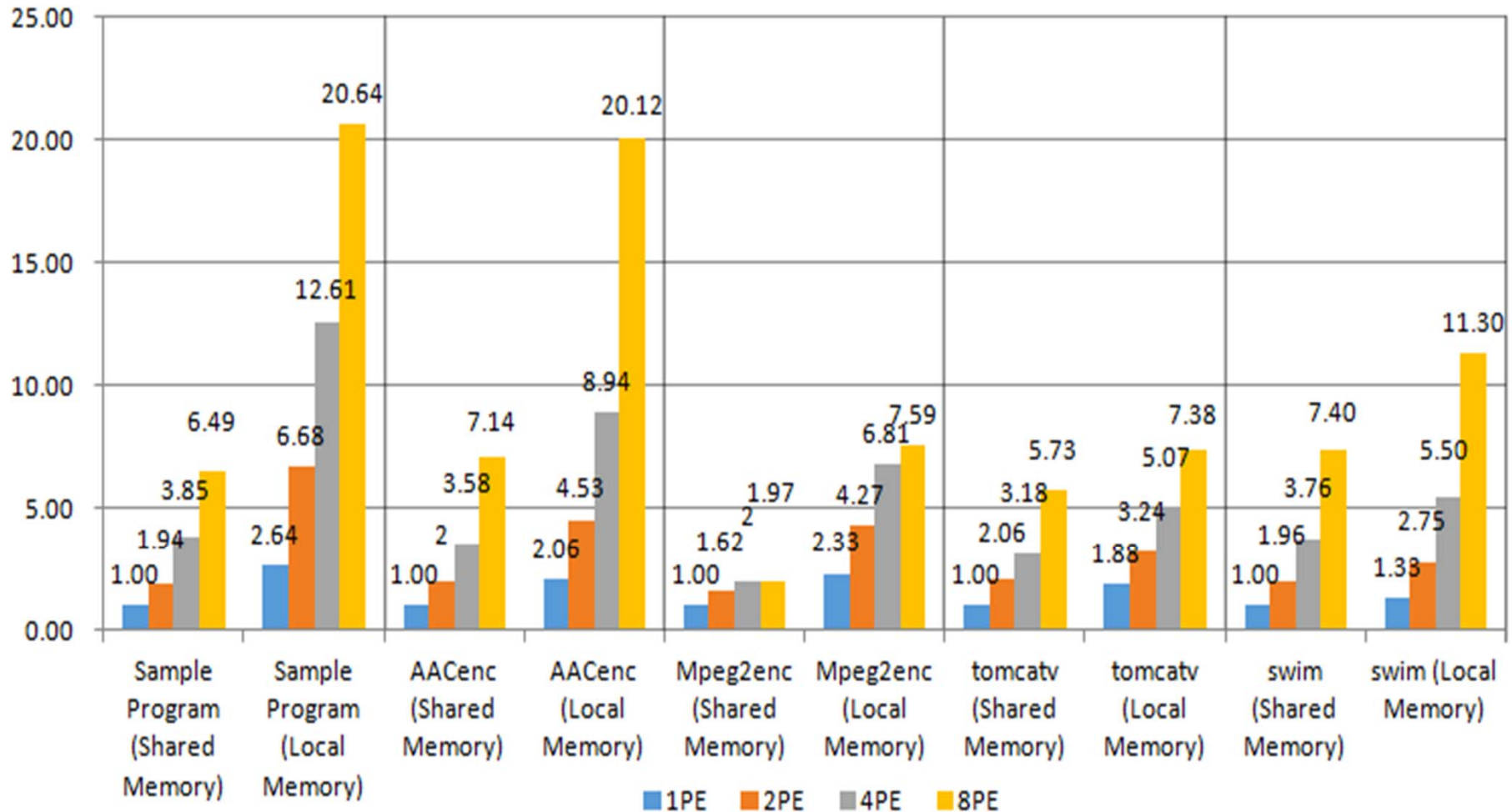
```
for (i = 0; i < 16; i++)  
  for (j = 0; j < 64; j++)  
    a[i][j] = i + j;
```

```
for (i = 0; i < 15; i++)  
  for (j = 0; j < 63; j++)  
    b[i][j] = a[i][j] + a[i+1][j+1];
```



```
for (ii = 0; ii < 15; ii += 8)  
  for (jj = 0; jj < 63; jj += 32)  
    for (i = ii; i < min(15, ii + 8 + 1); i++)  
      for (j = jj; j < min(63, jj + 32 + 1); j++)  
        a[i][j] = i + j;  
  for (i = ii; i < min(15, ii + 8); i++)  
    for (j = jj; j < min(63, jj + 32); j++)  
      b[i][j] = a[i][j] + a[i+1][j+1];
```

## Speedups by the Proposed Local Memory Management Compared with Utilizing Shared Memory on Benchmarks Application using RP2

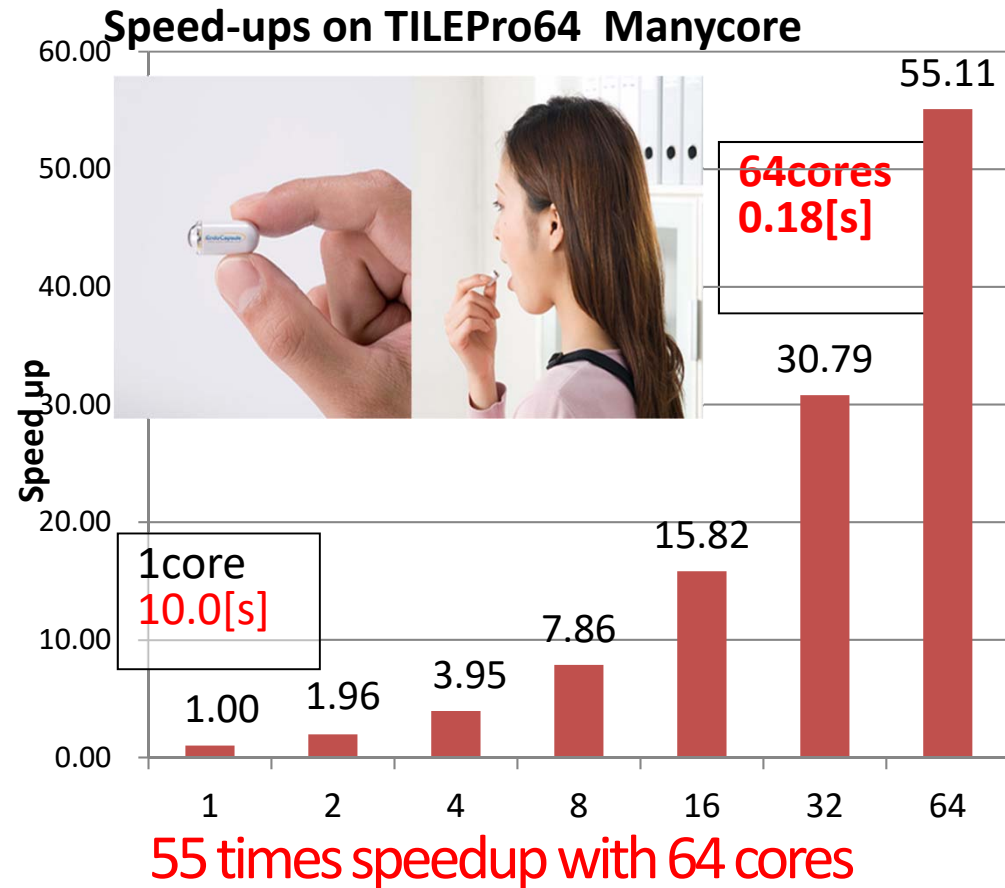
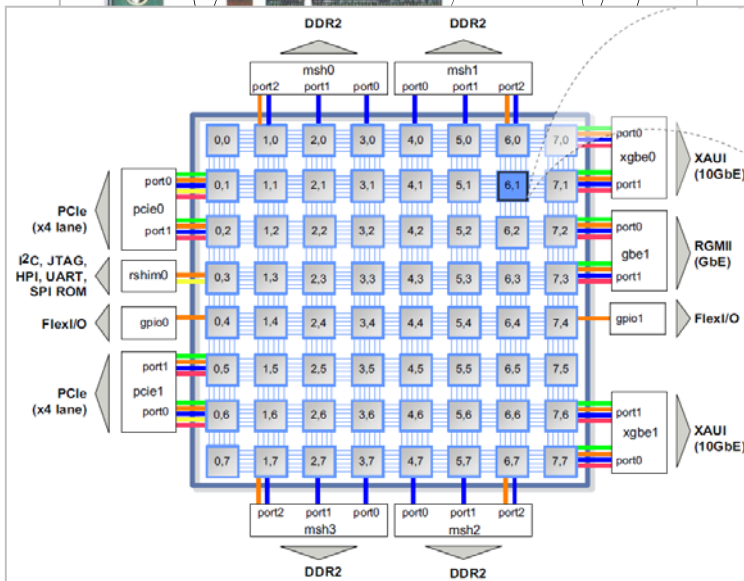


**20.12 times speedup for 8cores execution using local memory against sequential execution using off-chip shared memory of RP2 for the AACenc**

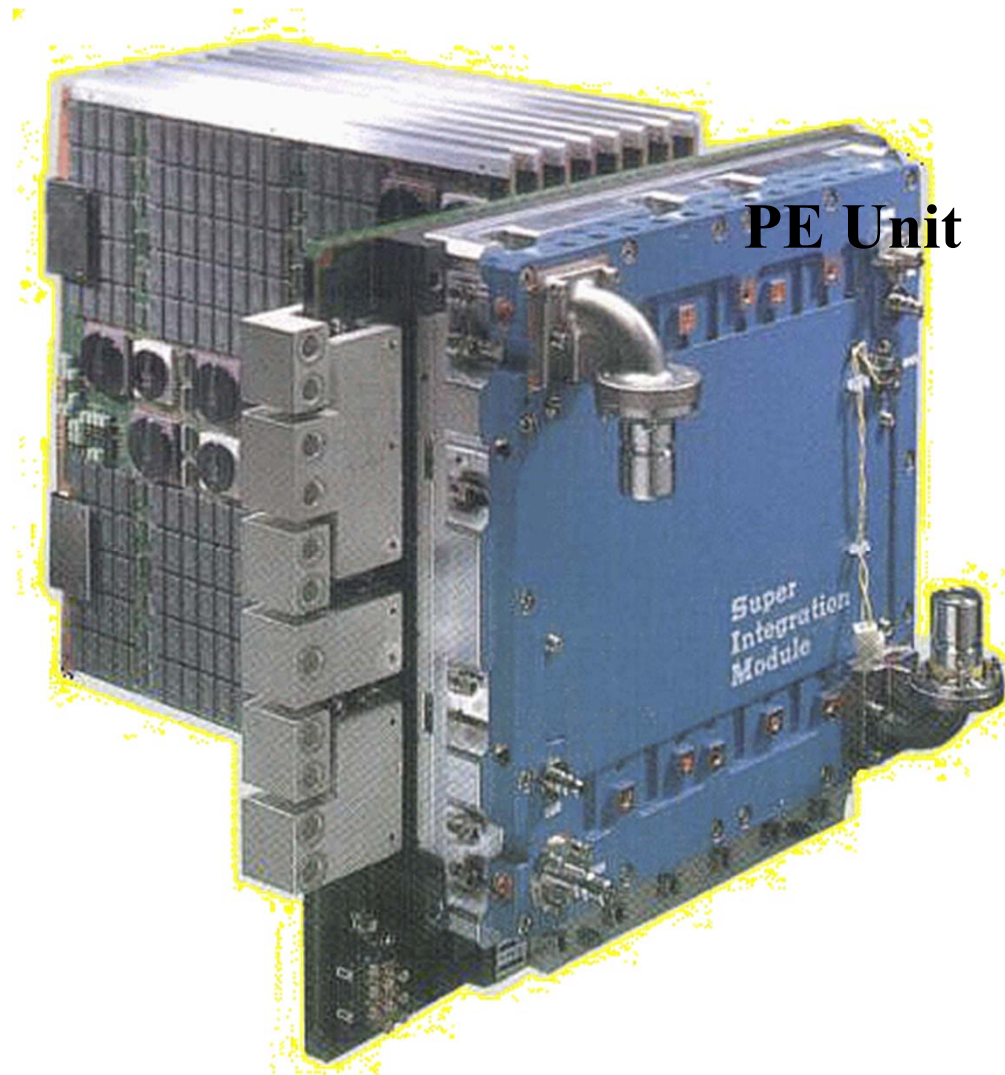
# Automatic Parallelization of JPEG-XR for Drinkable Inner Camera (Endo Capsule)

10 times more speedup needed after parallelization for 128 cores of Power 7. Less than 35mW power consumption is required.

- TILEPro64

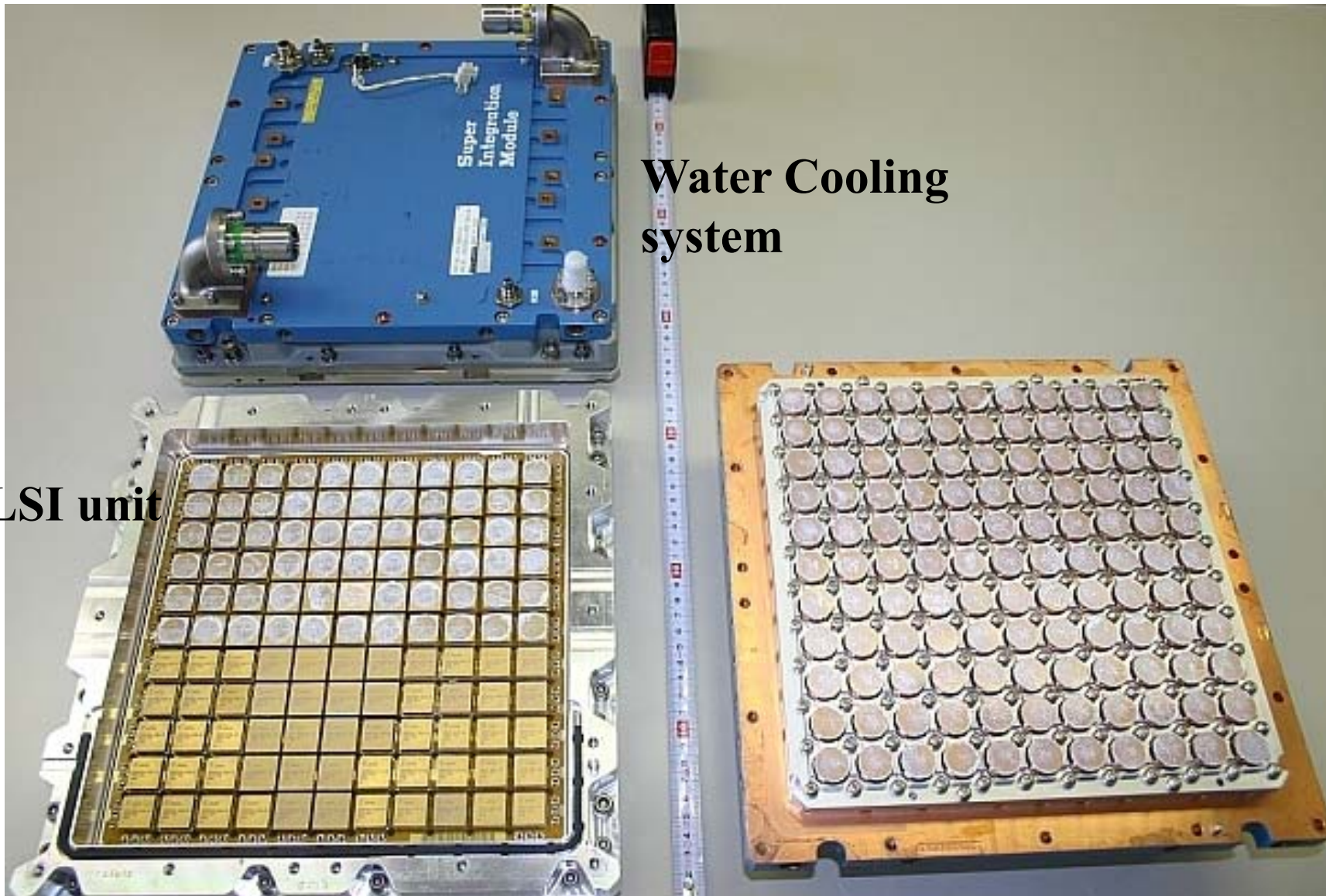


# VPP500/NWT



LIMITED

# VPP500/NWT

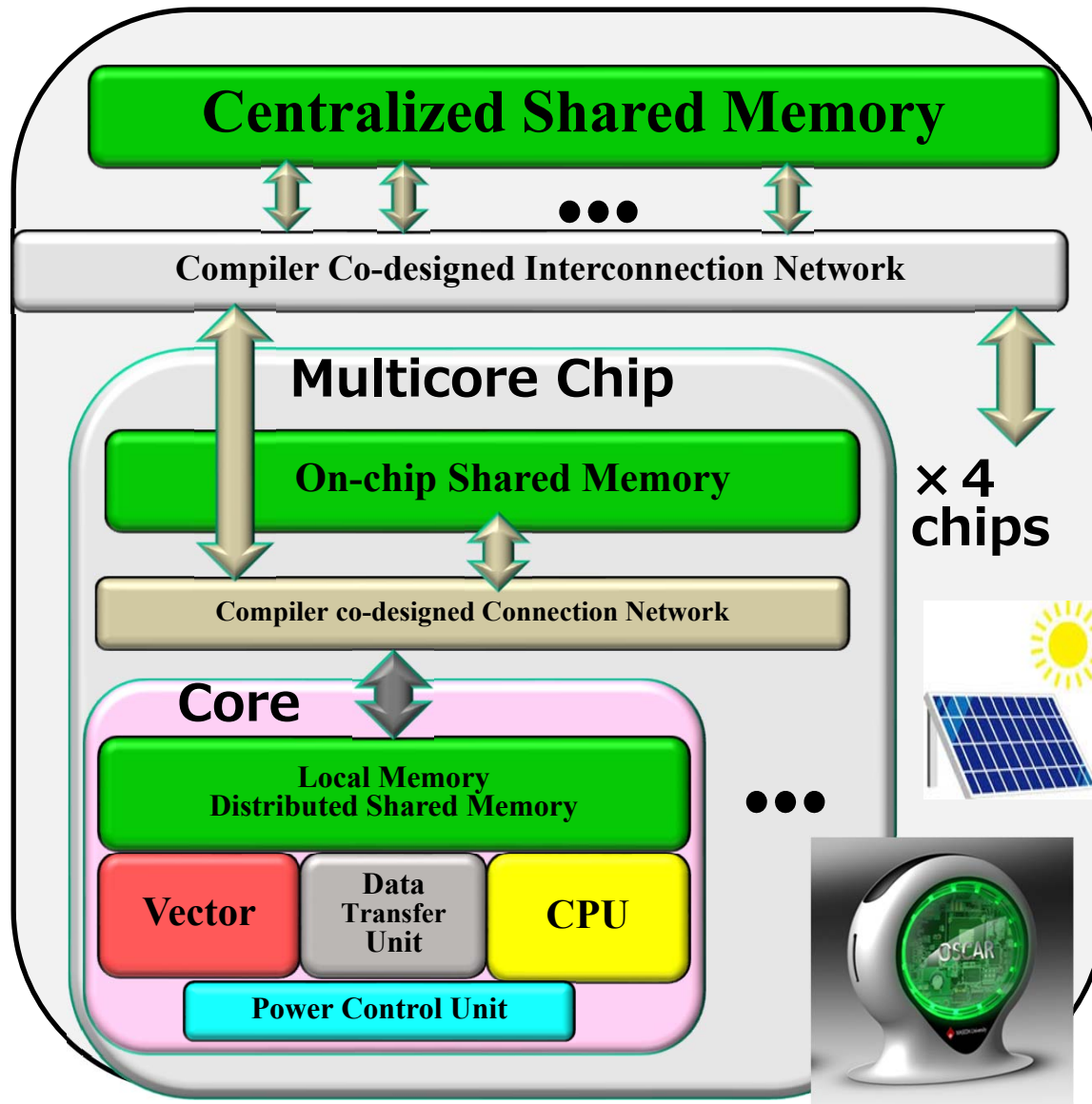


**PE LSI unit**

**Water Cooling  
system**



# OSCAR Vector Multicore and Compiler for Embedded to Servers with OSCAR Technology



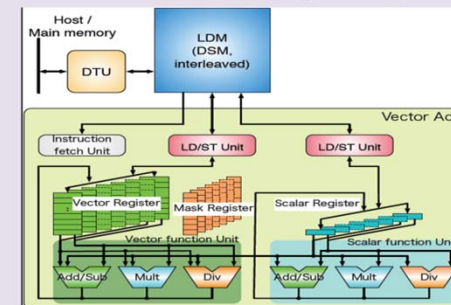
Target:

- Solar Powered
- Compiler power reduction.
- Fully automatic parallelization and vectorization including local memory management and data transfer.

## Vector Accelerator

### Features

- Attachable for any CPUs (Intel, ARM, IBM)
- Data driven initiation by sync flags



### Function Units [tentative]

- Vector Function Unit
  - 8 double precision ops/clock
  - 64 characters ops/clock
  - Variable vector register length
  - Chaining LD/ST & Vector pipes
- Scalar Function Unit

### Registers[tentative]

- Vector Register 256Bytes/entry, 32entry
- Scalar Register 8Bytes/entry
- Floating Point Register 8Bytes/entry
- Mask Register 32Bytes/entry



# Future Multicore Products with Automatic Parallelizing Compiler



## Next Generation Automobiles

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