Software and Hardware for High Performance and Low Power Homogeneous and Heterogeneous Multicore Systems

Hironori Kasahara

Professor, Dept. of Computer Science & Engineering

Director, Advanced Multicore Processor Research Institute

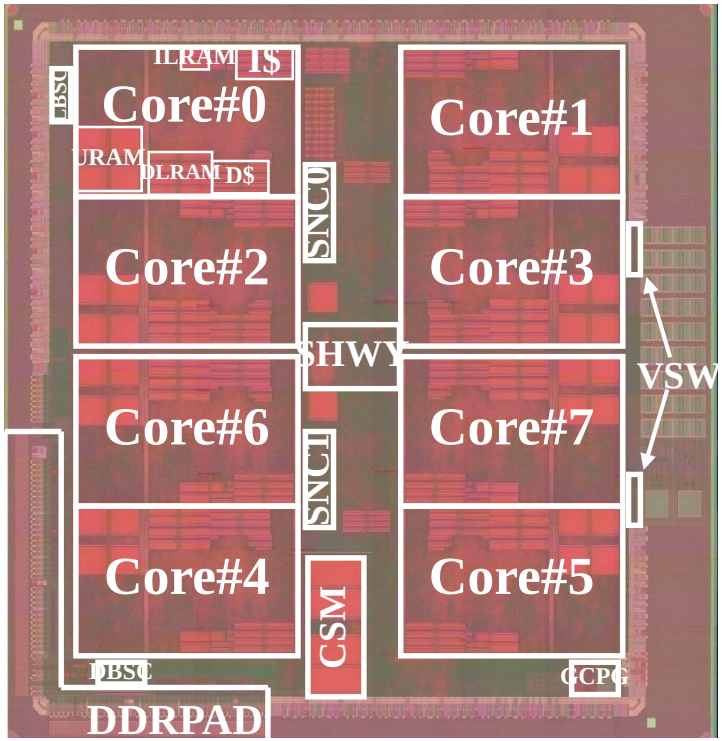
Waseda University, Tokyo, Japan

IEEE Computer Society President Elect 2017

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

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”

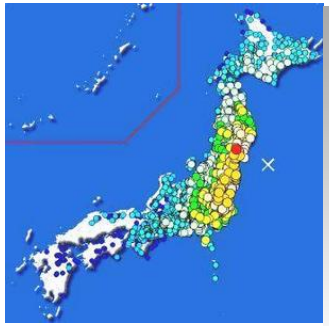
Power ∝ Frequency * Voltage^2
(Voltage ∝ Frequency)
Power ∝ Frequency^3

If Frequency is reduced to 1/4 (Ex. 4GHz→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.

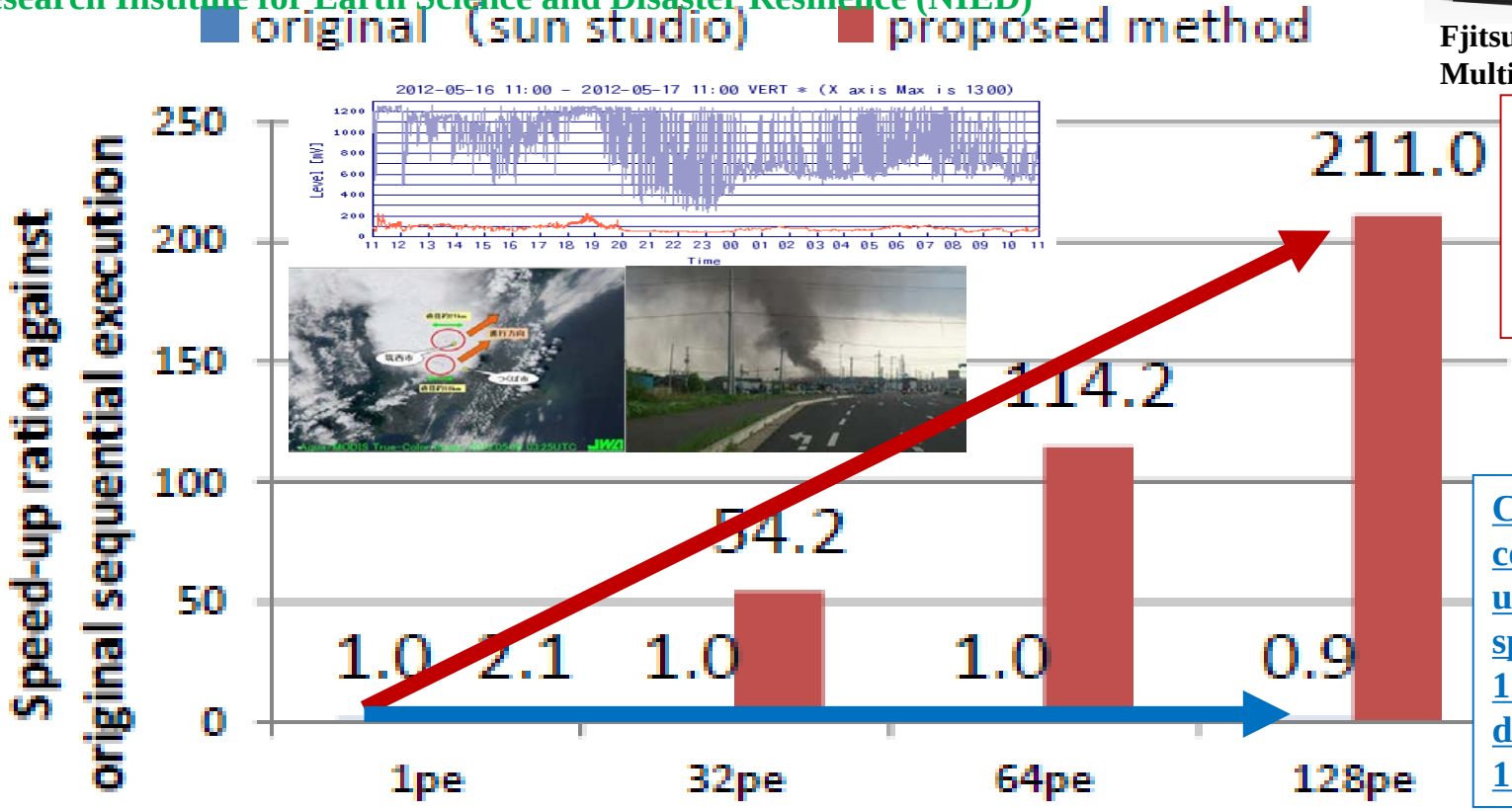
Parallel Soft is important for scalable performance of multicore

- 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)



Fjitsu M9000 SPARC Multicore Server

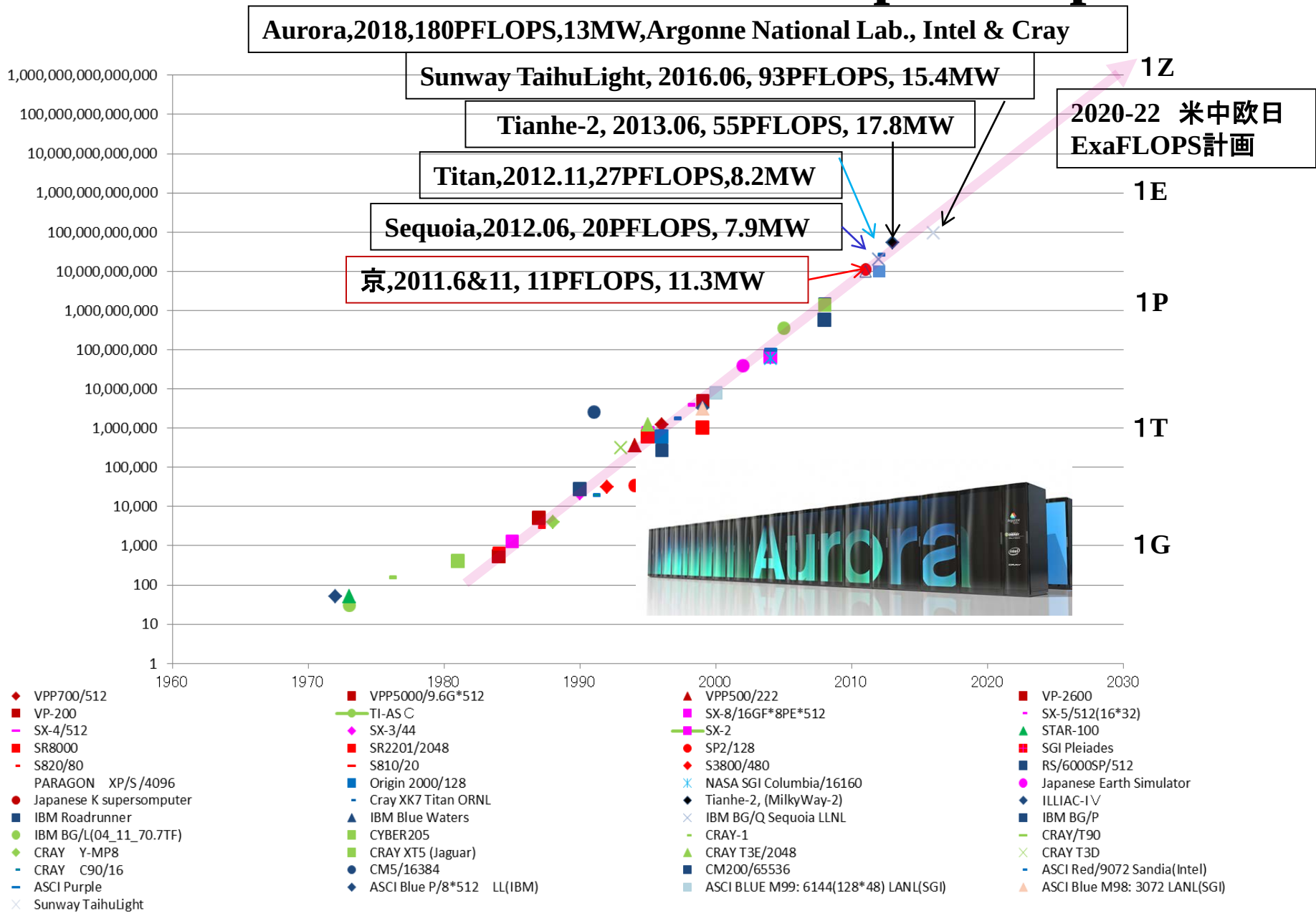


OSCAR Compiler gives us 211 times speedup with 128 cores

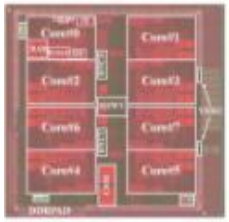
Commercial compiler gives us 0.9 times speedup with 128 cores (slow-downed against 1 core)

- 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

Trend of Peak Performances of Supercomputers

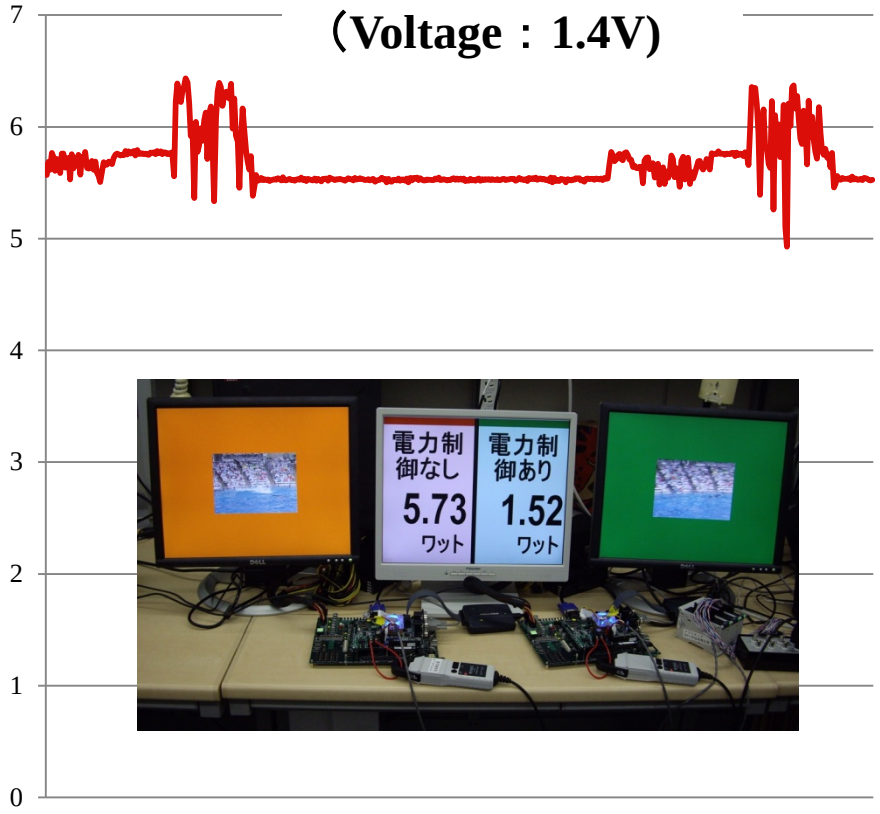


Power Reduction of MPEG2 Decoding to 1/4 on 8 Core Homogeneous Multicore RP-2 by OSCAR Parallelizing Compiler

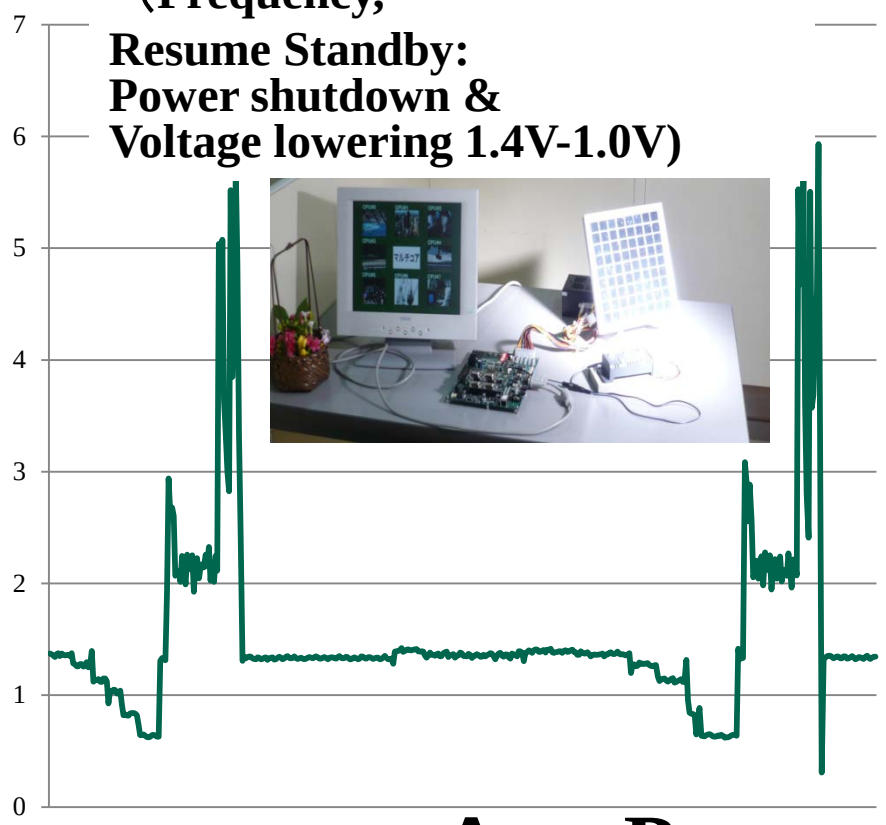


MPEG2 Decoding with 8 CPU cores

Without Power Control （Voltage： 1.4V)



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

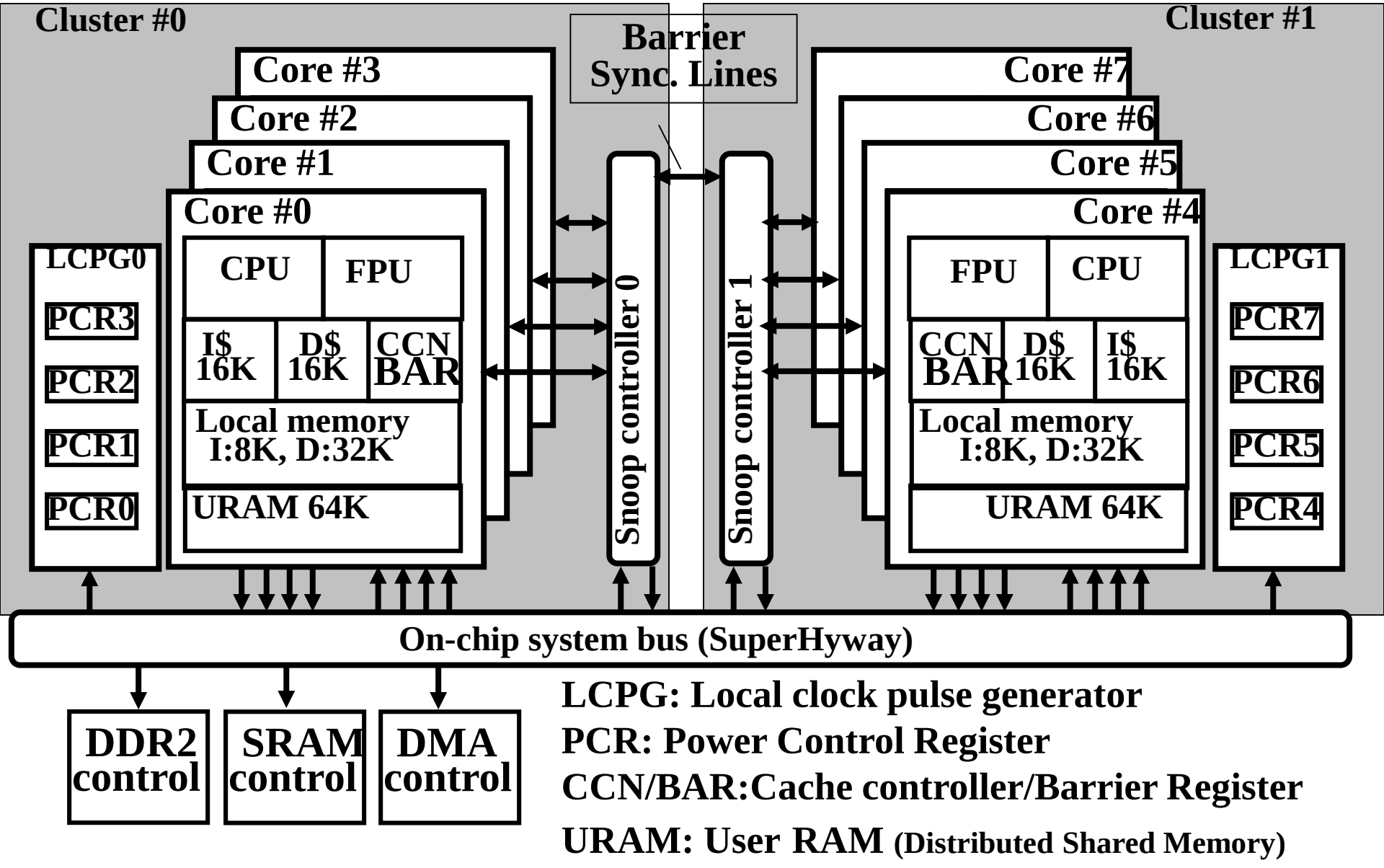


Avg. Power 5.73 [W]

73.5% Power Reduction

Avg. Power 1.52 [W]

Compiler Co-designed Multicore RP2



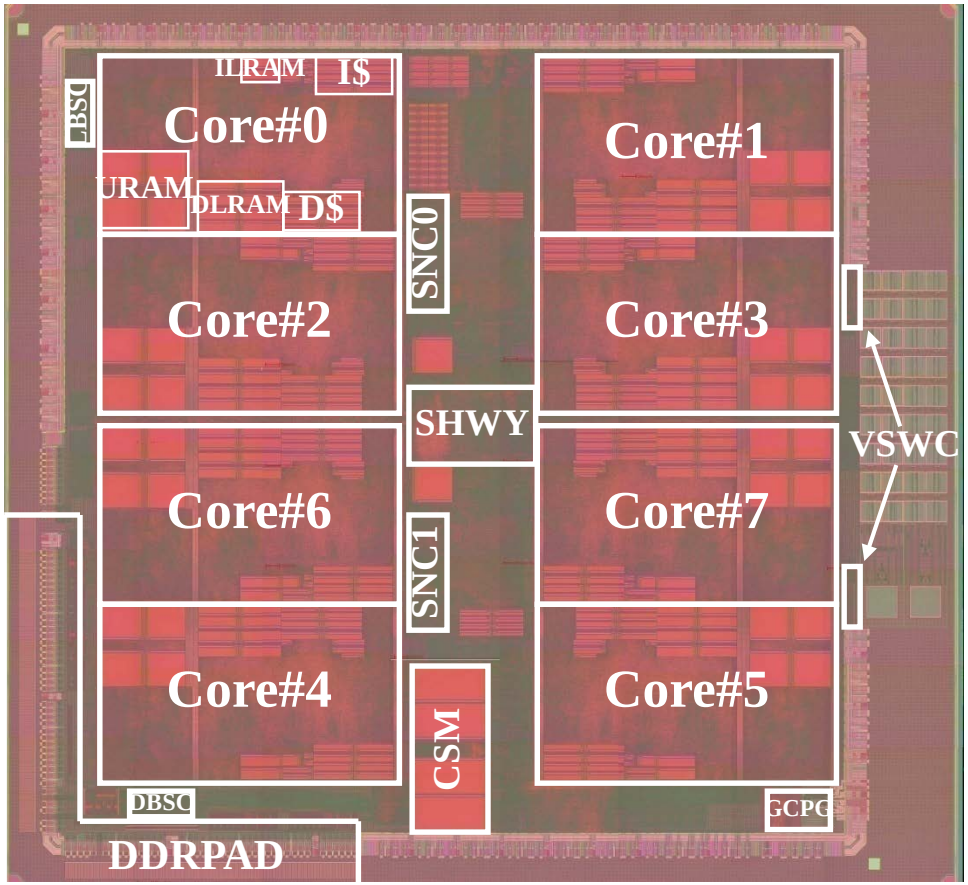
LCPG: Local clock pulse generator

PCR: Power Control Register

CCN/BAR:Cache controller/Barrier Register

URAM: User RAM (Distributed Shared Memory)

Renesas-Hitachi-Waseda Low Power 8 core RP2 Developed in 2007 in METI/NEDO project



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)

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"

Industry-government-academia collaboration in R&D and target practical applications

Protect lives

For smart life

Robots

On-board vehicle technology (navigation systems, integrated controllers, infrastructure coordination)

Protect environment

Waseda University :R&D
Many-core system technologies with ultra-low power consumption
OSCAR many-core chip

OSCAR Many-core Chip

Compiler, API

Green supercomputers

Super real-time disaster simulation (tectonic shifts, tsunami), tornado, flood, fire spreading)

Green cloud servers

Stock trading

Cool desktop servers

Consumer electronic
Internet TV/DVD

Camcorders

Cameras

Capsule inner cameras

Solar Powered Smart phones

Operation/recharging by solar cells

Medical servers

National Institute of Radiological Sciences

Heavy particle radiation planning, cerebral infarction)

Non-fan, cool, quiet servers designed for server

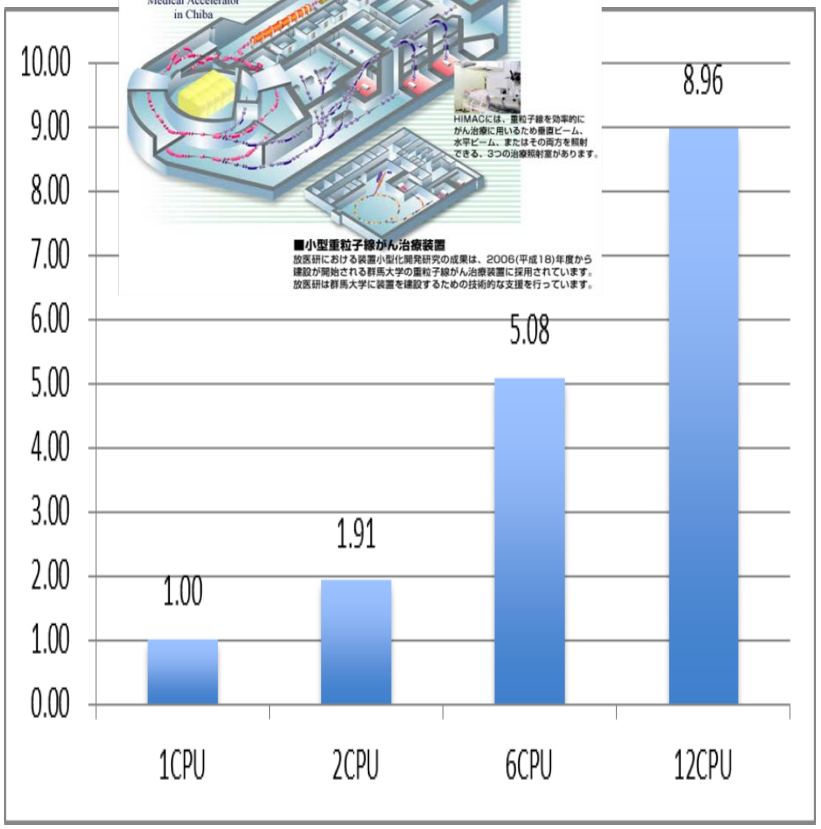
Industry

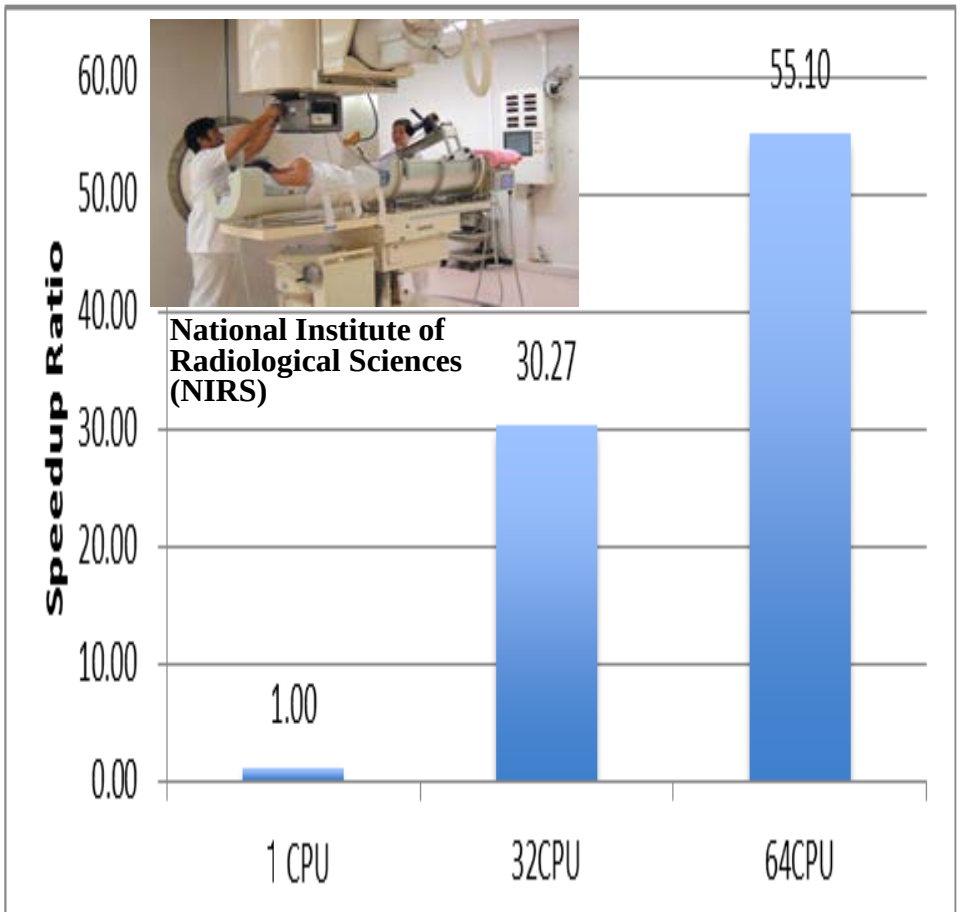
Intelligent home appliances

Supercomputers and servers

Cancer Treatment
Carbon Ion Radiotherapy

(Previous best was 2.5 times speedup on 16 processors with hand optimization)





8.9times speedup by 12 processors

Intel Xeon X5670 2.93GHz 12 core SMP (Hitachi HA8000)

55 times speedup by 64 processors

IBM Power 7 64 core SMP (Hitachi SR16000)

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

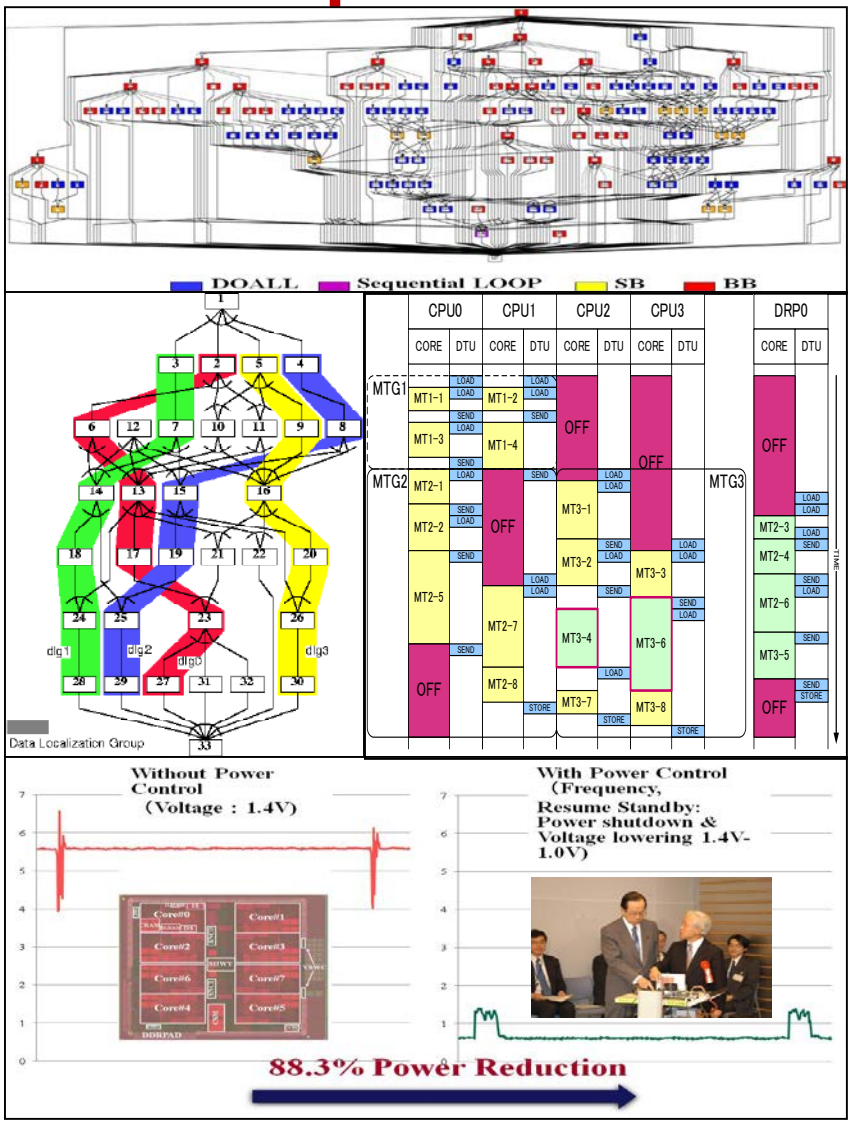
Automatic data management for distributed shared memory, cache and local memory

Data Transfer Overlapping

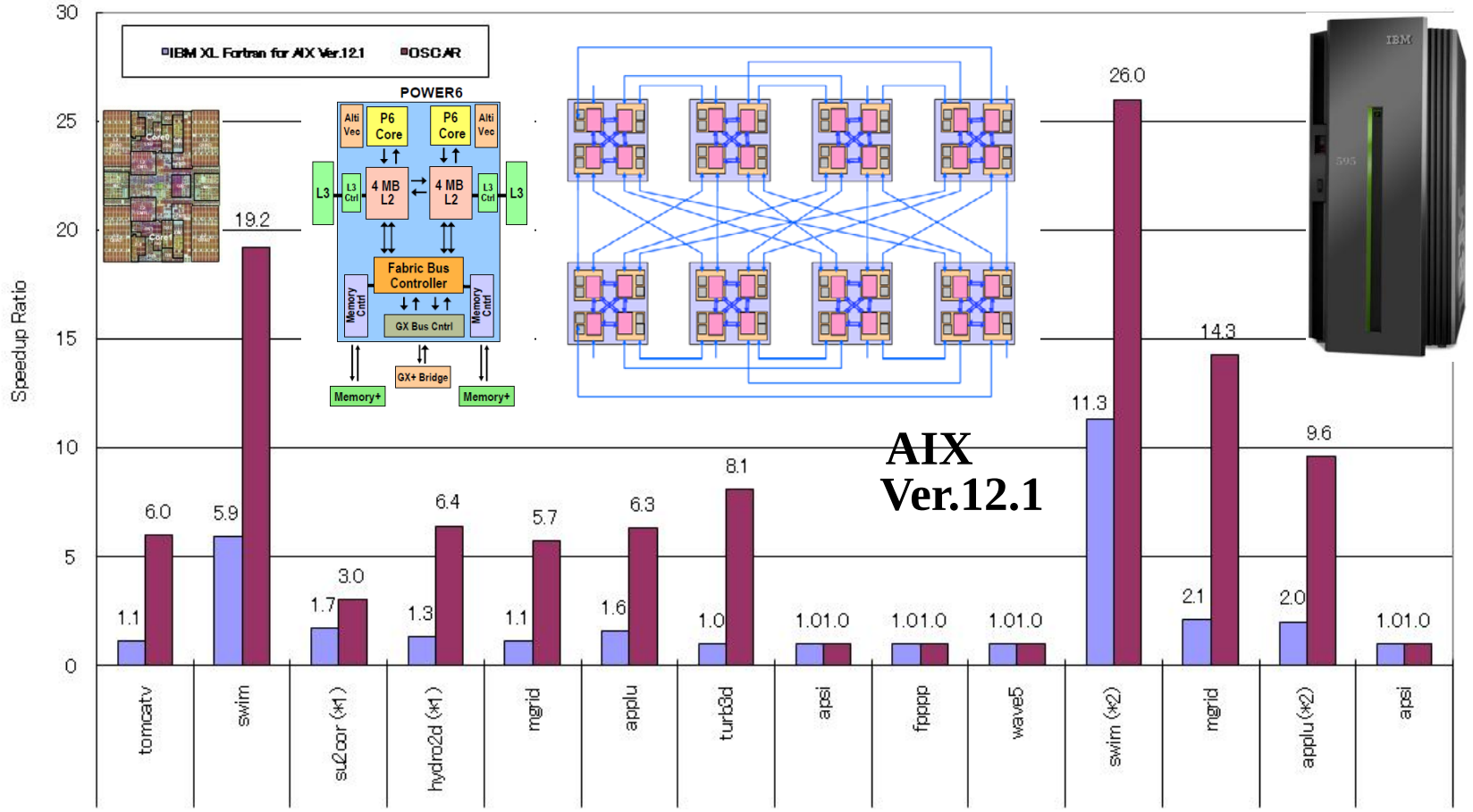
Data transfer overlapping using Data Transfer Controllers (DMAs)

Power Reduction

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



Performance of OSCAR Compiler on IBM p6 595 Power6 (4.2GHz) based 32-core SMP Server



Compi

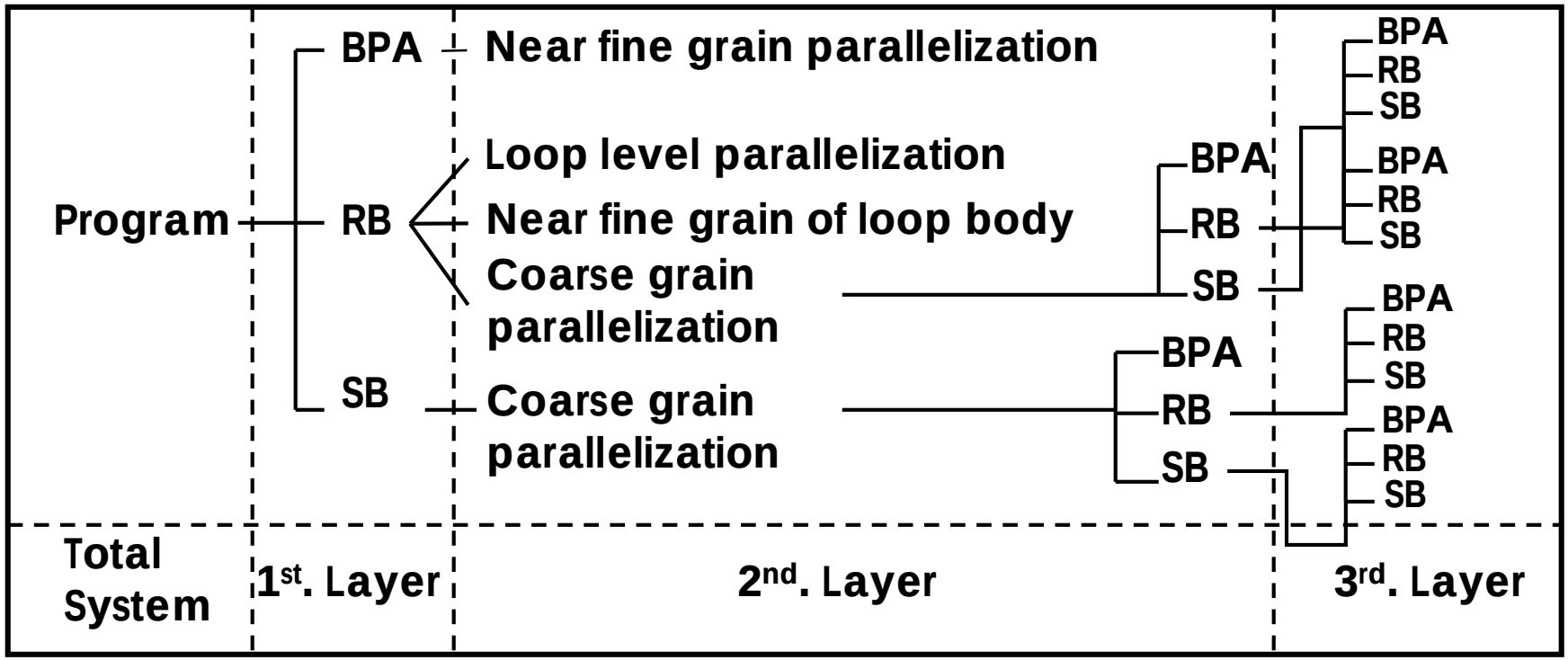
(*1) Sequential: -O3 –qarch=pwr6, XLF: -O3 –qarch=pwr6 –qsmp=auto, OSCAR: -O3 –qarch=pwr6 –qsmp=noauto

(*2) Sequential: -O5 -q64 –qarch=pwr6, XLF: -O5 –q64 –qarch=pwr6 –qsmp=auto, OSCAR: -O5 –q64 –qarch=pwr6 –qsmp=noauto

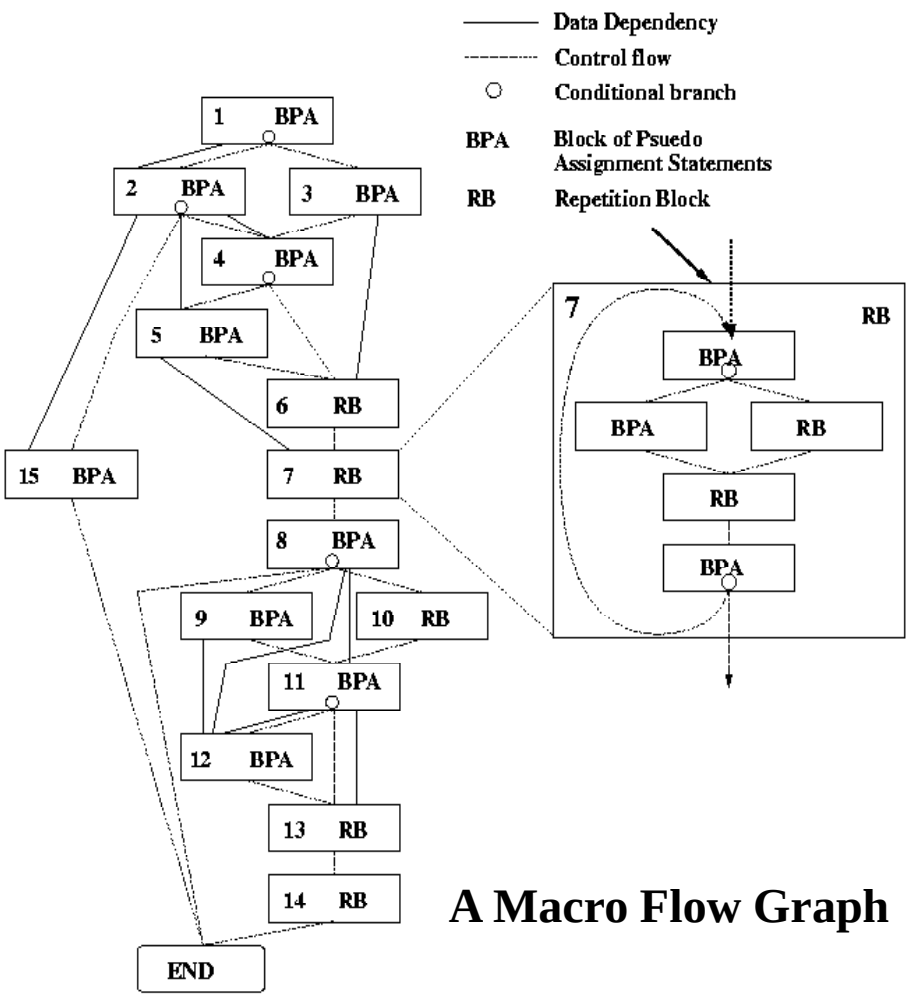
(Others) Sequential: -O5 –qarch=pwr6, XLF: -O5 –qarch=pwr6 –qsmp=auto, OSCAR: -O5 –qarch=pwr6 –qsmp=noauto

Generation of Coarse Grain Tasks

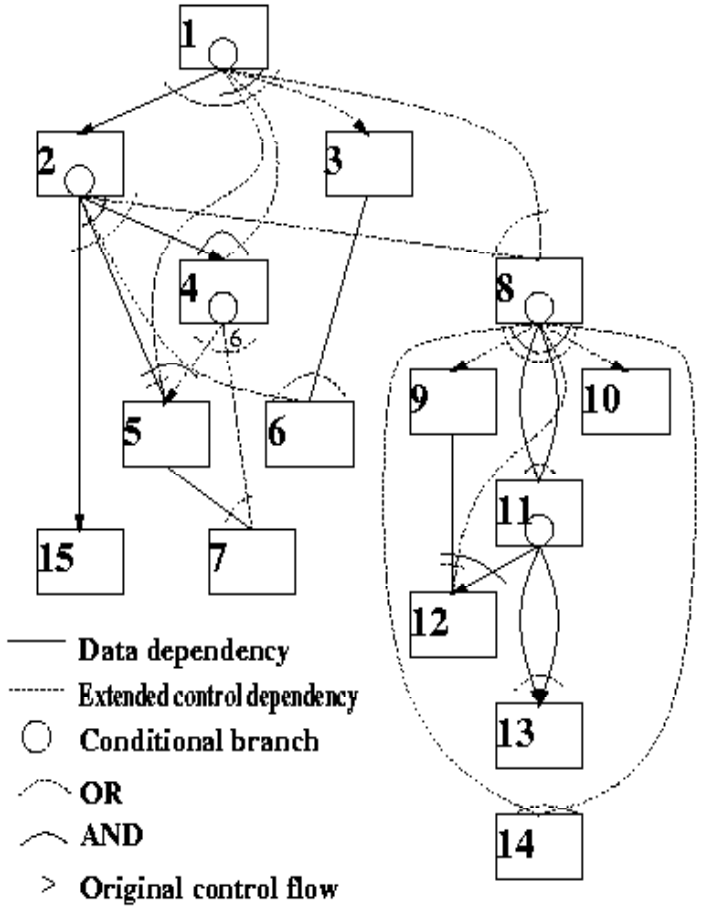
- **Macro-tasks (MTs)**
  - **Block of Pseudo Assignments (BPA): Basic Block (BB)**
  - **Repetition Block (RB) : natural loop**
  - **Subroutine Block (SB): subroutine**



Earliest Executable Condition Analysis for Coarse Grain Tasks (Macro-tasks)

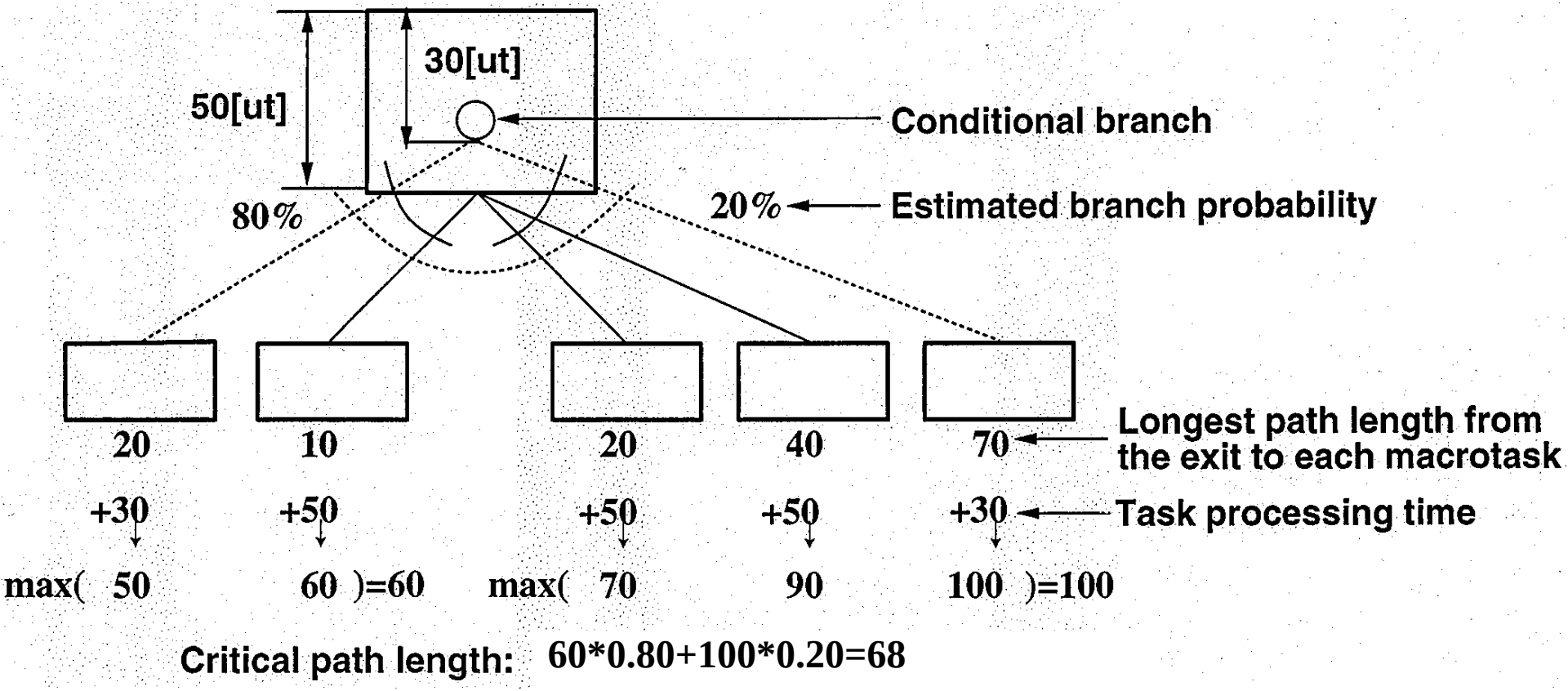


A Macro Flow Graph



A Macro Task Graph

PRIORITY DETERMINATION IN DYNAMIC CP METHOD



Earliest Executable Conditions

EEC: Control dependence + Data Dependence

Control dependences show executions of MTs are decided

Data dependences show data accessed by MTs are ready

MT2 may start execution after MT1 branches to MT2 and MT1 finish execution.

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}

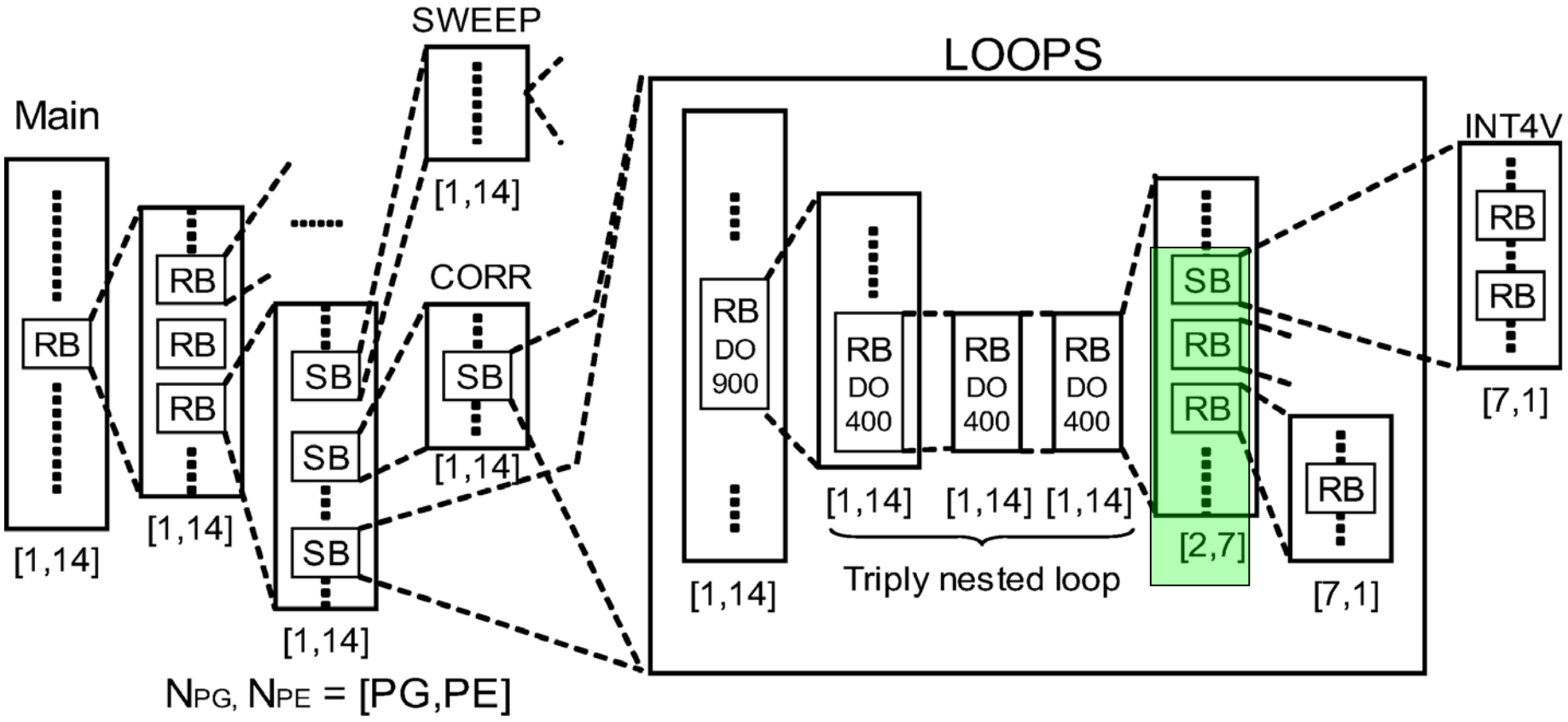
MT3 may start execution after MT1 branches to MT3.

MT6 may start execution after MT3 finish execution or MT2 branches to MT4.

Automatic processor assignment in 103.su2cor

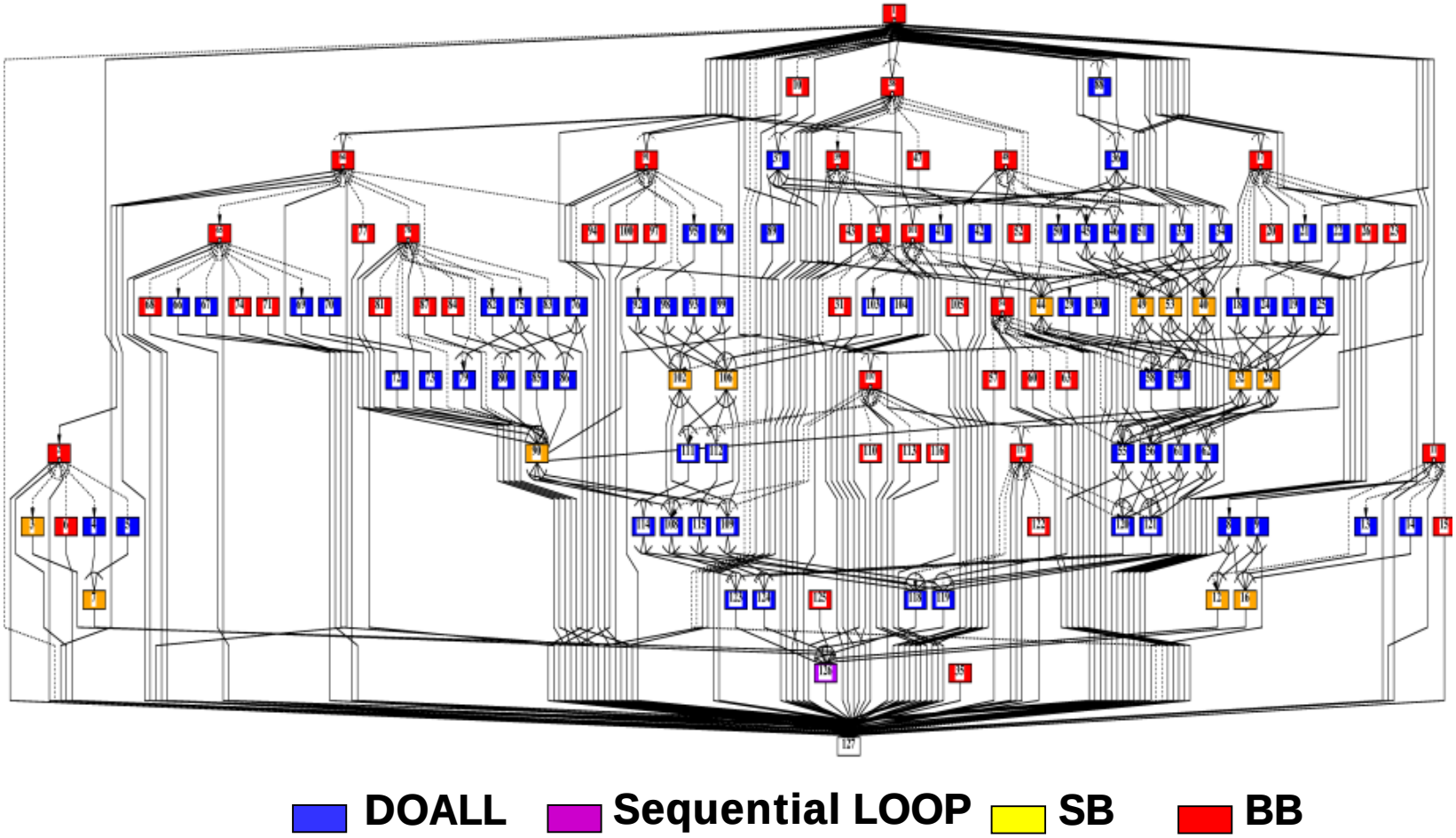
- **Using 14 processors**

Coarse grain parallelization within DO400



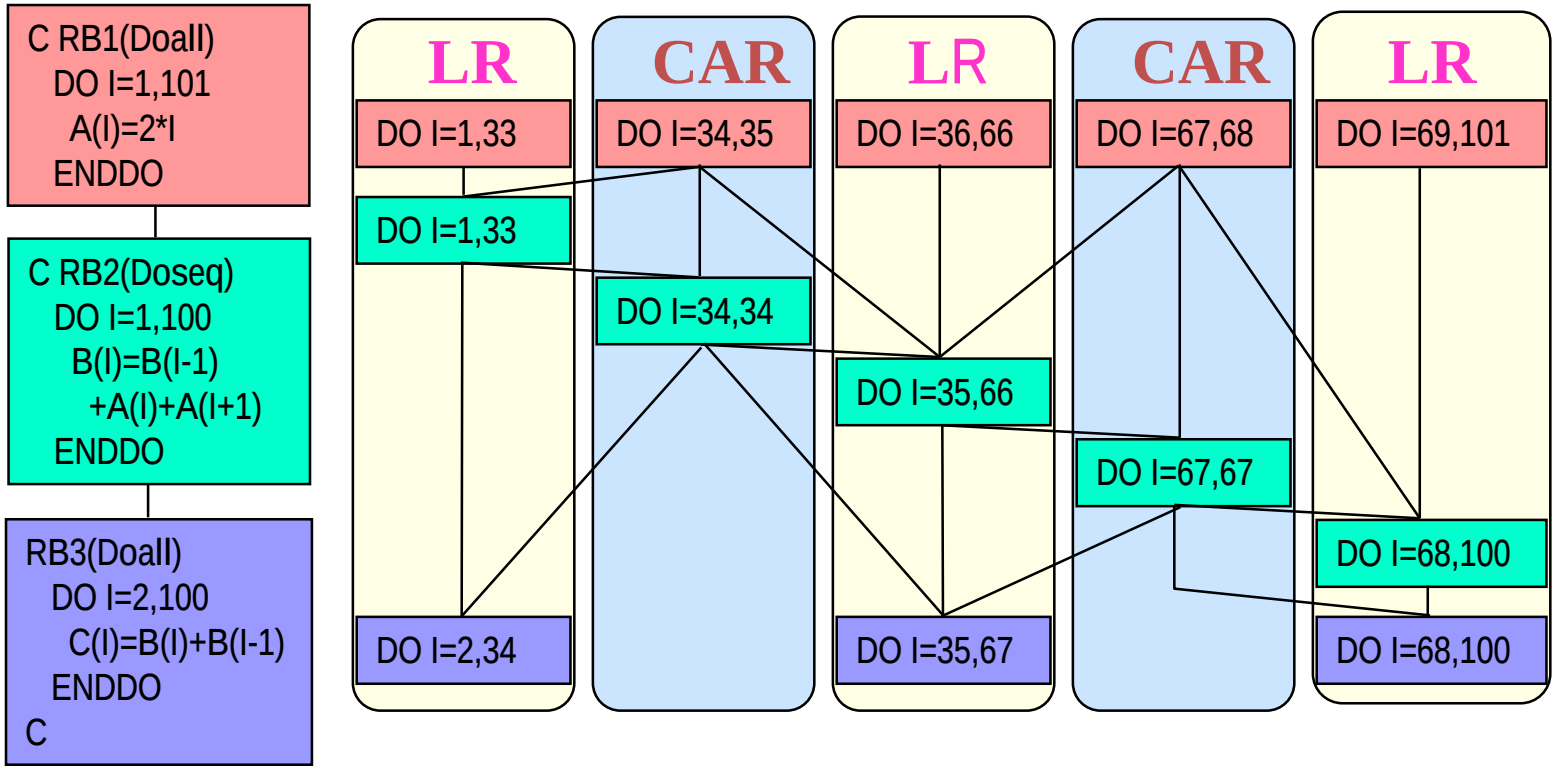
MTG of Su2cor-LOOPS-DO400

- **Coarse grain parallelism PARA_ALD = 4.3**



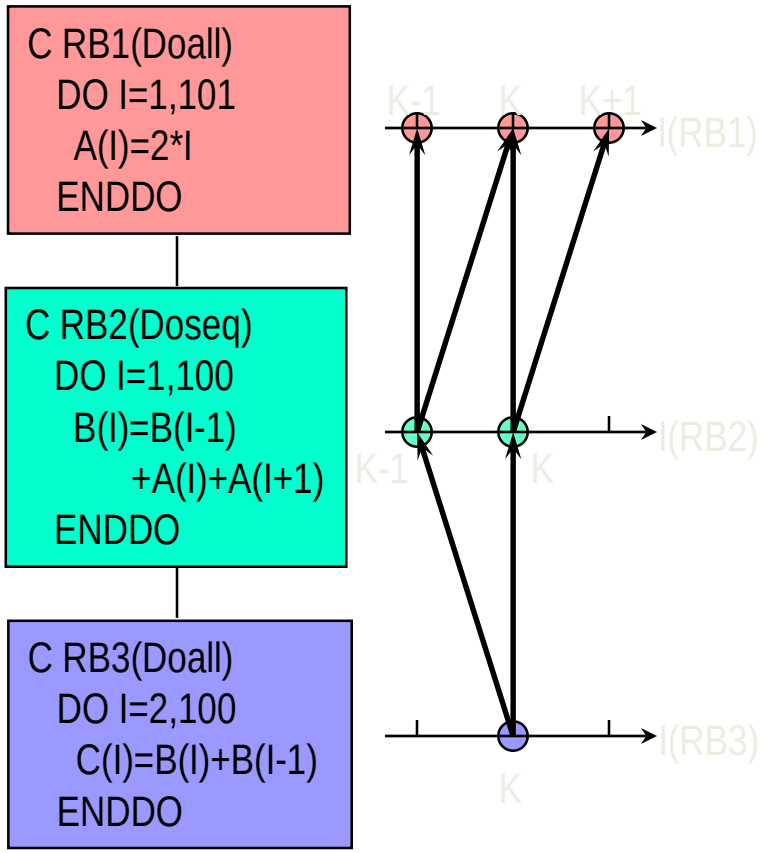
Data-Localization: Loop Aligned Decomposition

- **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**



Inter-loop data dependence analysis in TLG

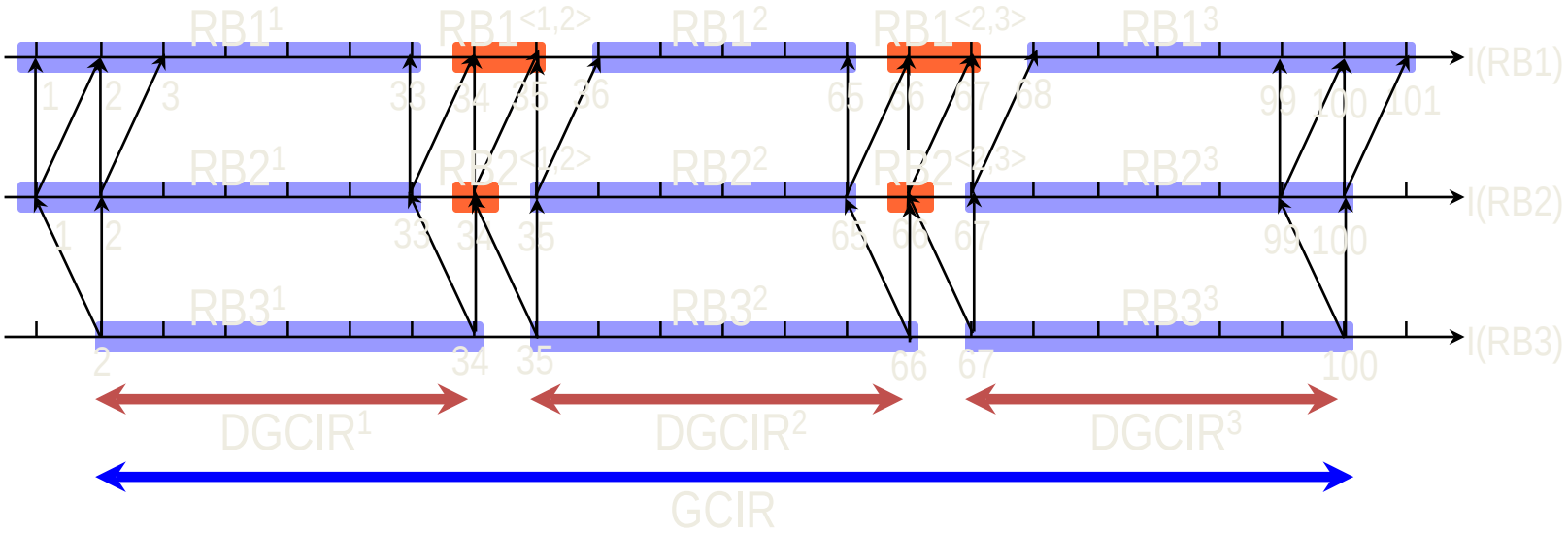
- Define exit-RB in TLG as Standard-Loop
- Find iterations on which a iteration of Standard-Loop is data dependent
  - e.g. K_{th} of RB3 is data-dep on $K\text{-}1_{th}$,K_{th} of RB2, on $K\text{-}1_{th}$,K_{th},$K\text{+}1_{th}$ of RB1 **indirectly.**



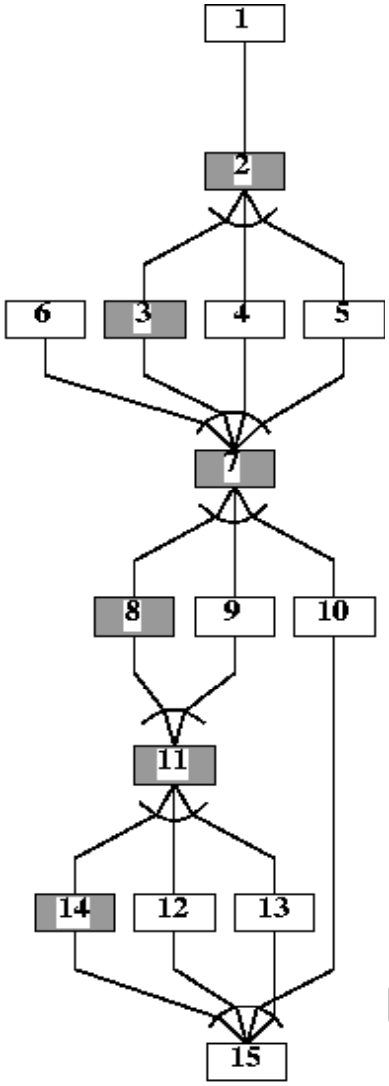
Example of TLG

Decomposition of RBs in TLG

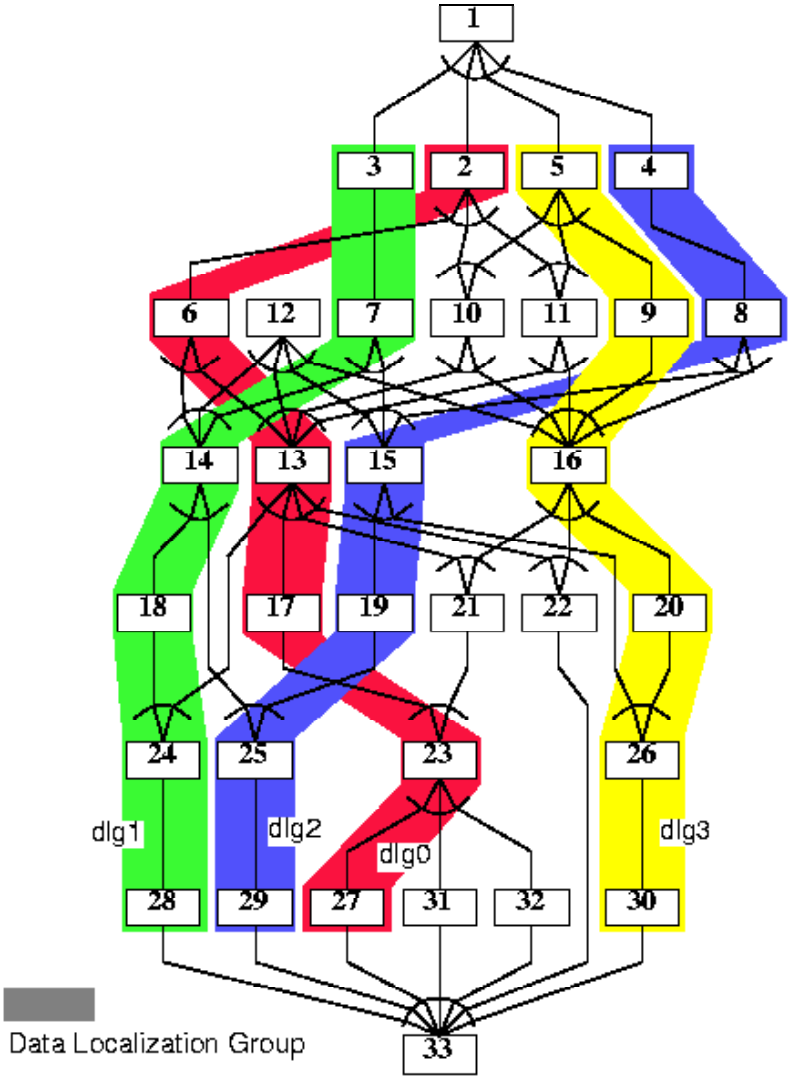
- Decompose GCIR into $DGCIR^p(1 \leqq p \leqq n)$
  - n: (multiple) num of PCs, DGCIR: Decomposed GCIR
- Generate CAR on which $DGCIR^p$&$DGCIR^{p+1}$ are data-dep.
- Generate LR on which $DGCIR^p$ is data-dep.



Data Localization



MTG



MTG after Division

A schedule for two processors

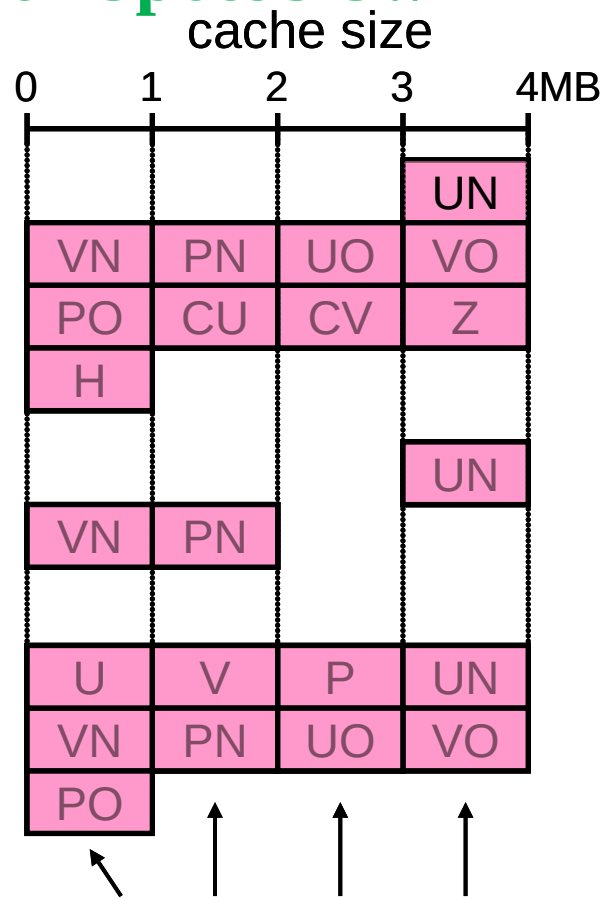
An Example of Data Localization for Spec95 Swim

```
      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)-TDTS8*(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

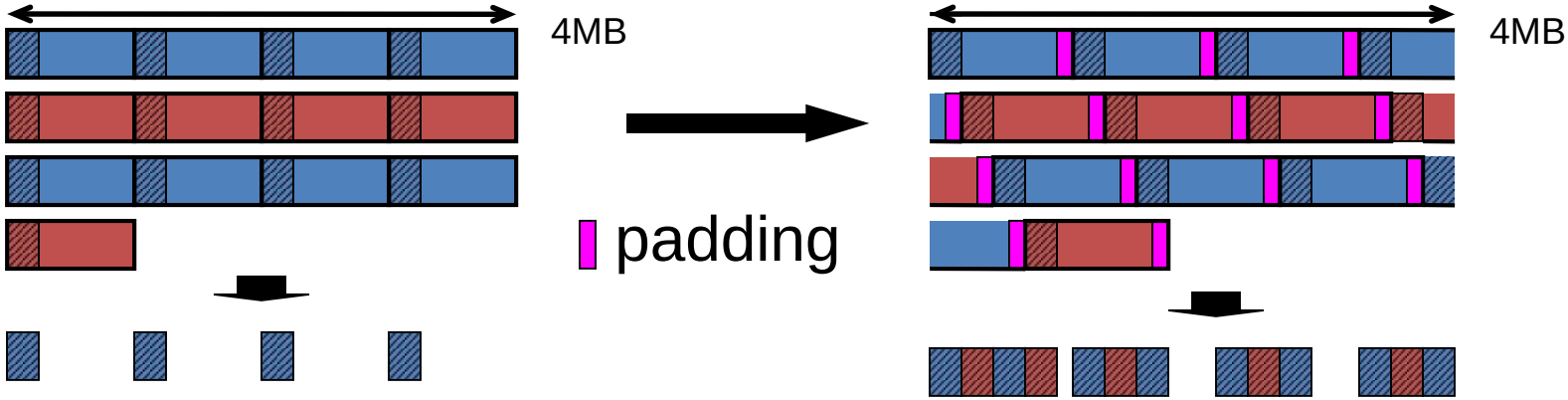
Declaration part of arrays in spec95 swim

before padding

```
PARAMETER (N1=513, N2=513)

COMMON  U(N1,N2), V(N1,N2), P(N1,N2),
*       UNEW(N1,N2), VNEW(N1,N2),
1       PNEW(N1,N2), UOLD(N1,N2),
*       VOLD(N1,N2), POLD(N1,N2),
2       CU(N1,N2), CV(N1,N2),
*       Z(N1,N2), H(N1,N2)
```

after padding

```
PARAMETER (N1=513, N2=544)

COMMON  U(N1,N2), V(N1,N2), P(N1,N2),
*       UNEW(N1,N2), VNEW(N1,N2),
1       PNEW(N1,N2), UOLD(N1,N2),
*       VOLD(N1,N2), POLD(N1,N2),
2       CU(N1,N2), CV(N1,N2),
*       Z(N1,N2), H(N1,N2)
```



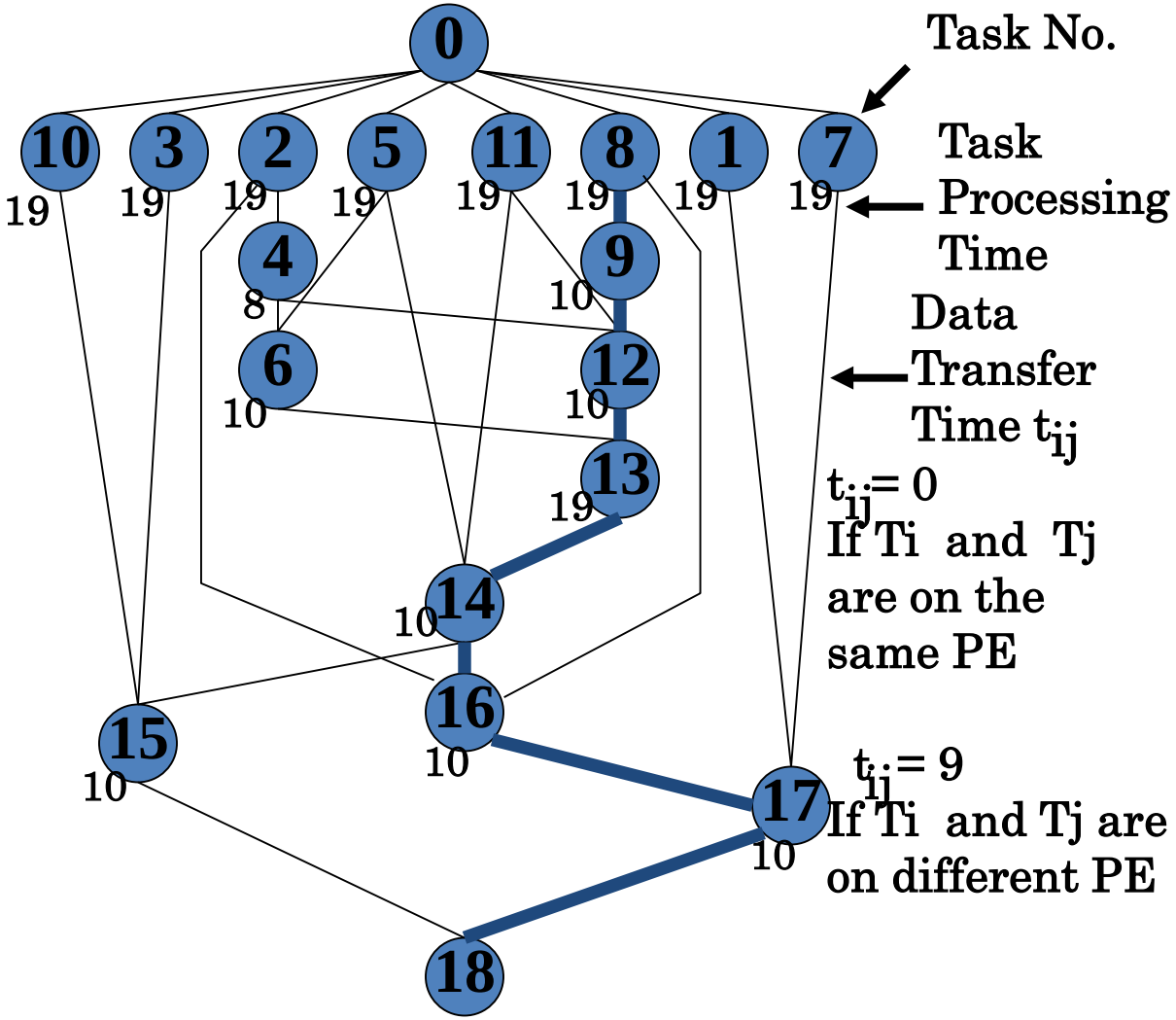
Box: Access range of DLG0

Statement Level Near Fine Grain Task

<<LU Decomposition>>
1) u12 = a12/l11
2) u24 = a24/l22
3) u34 = a34/l33
4) l54 = -l52 * u24
5) u45 = a45/l44
6) l55 = u55 - l54 * u45

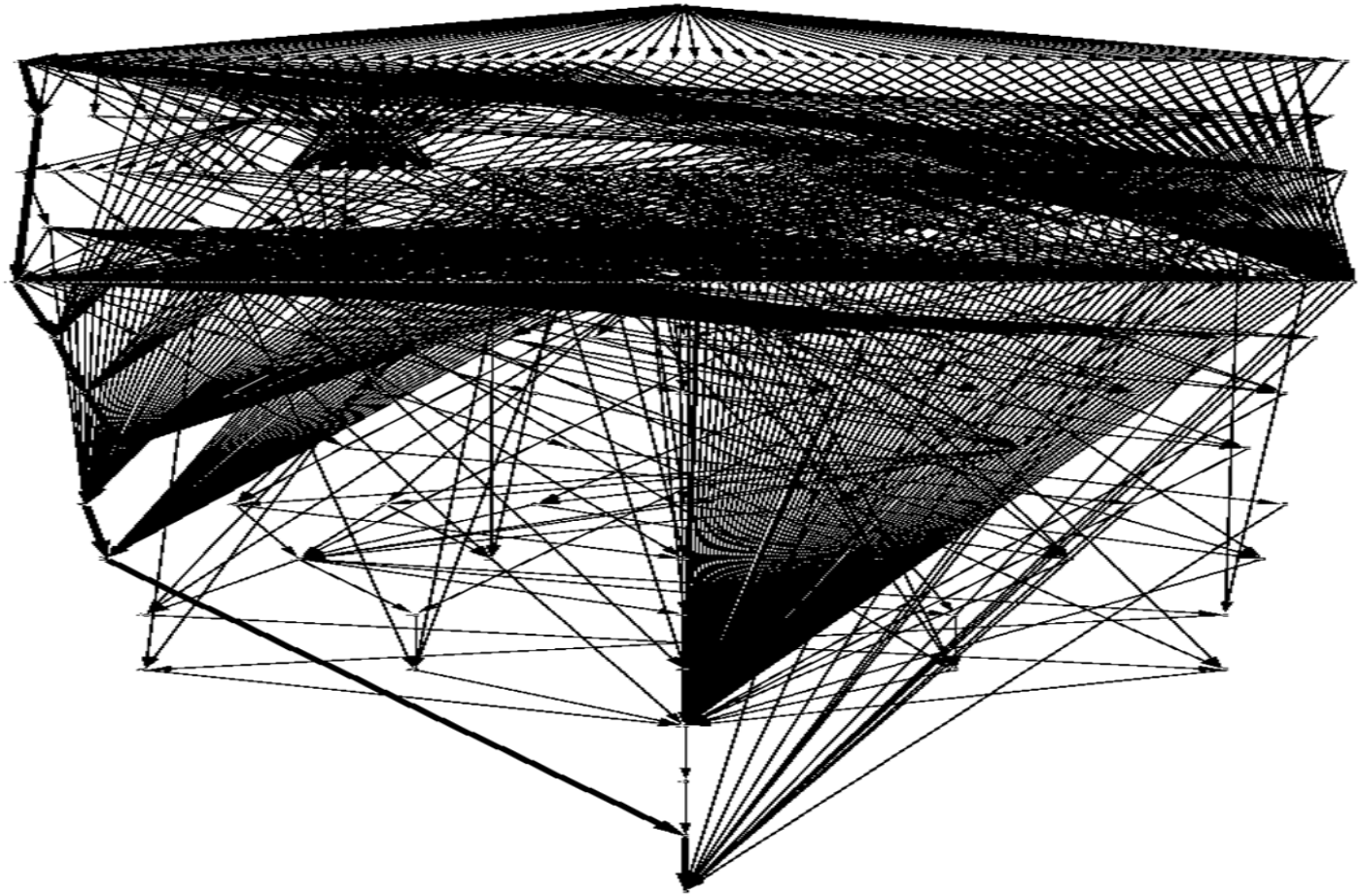
<<Forward Substitution>>
7) y1 = b1 / l11
8) y2 = b2 / l22
9) b5 = b5 - l52 * y2
10) y3 = b3 / l33
11) y4 = b4 / l44
12) b5 = b5 - l54 * y4
13) y5 = b5 / l55

<<Backward Substitution>>
14) x4 = y4 - u45 * y5
15) x3 = y3 - u34 * x4
16) x2 = y2 - u24 * x4
17) x1 = y1 - u12 * x2

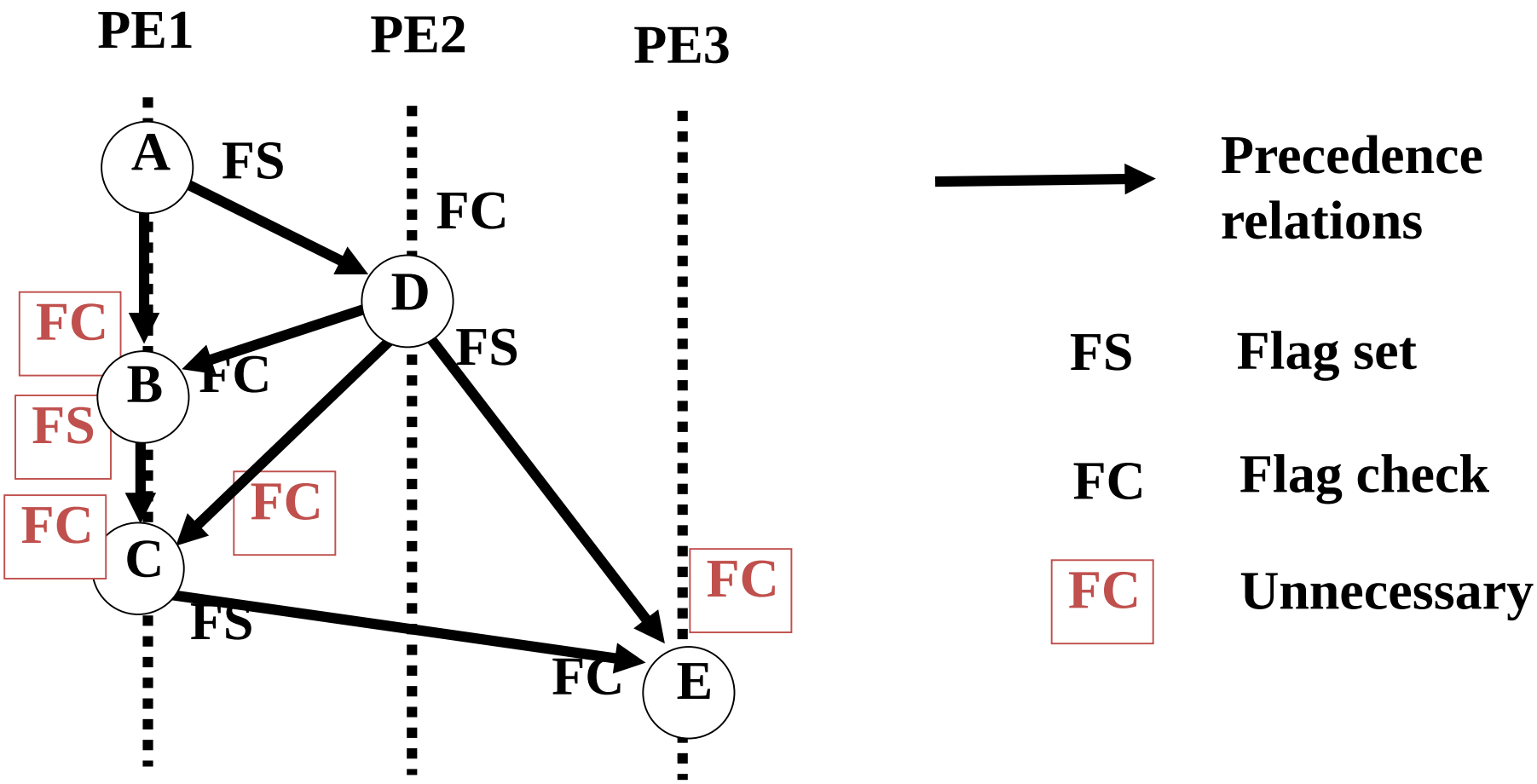


Task Graph for FPPPP

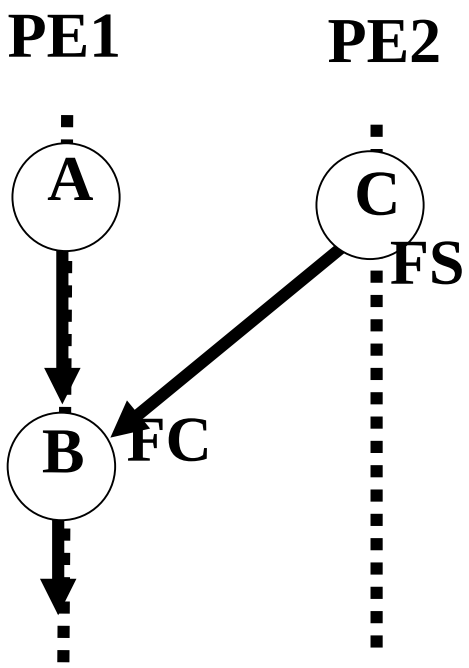
Statement level near fine grain parallelism

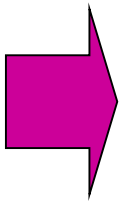


Elimination of Redundant Synchronization for Shared Data on Centralized Shared Memory after Static Task Scheduling



Generated Parallel Machine Code for Near Fine Grain Parallel Processing





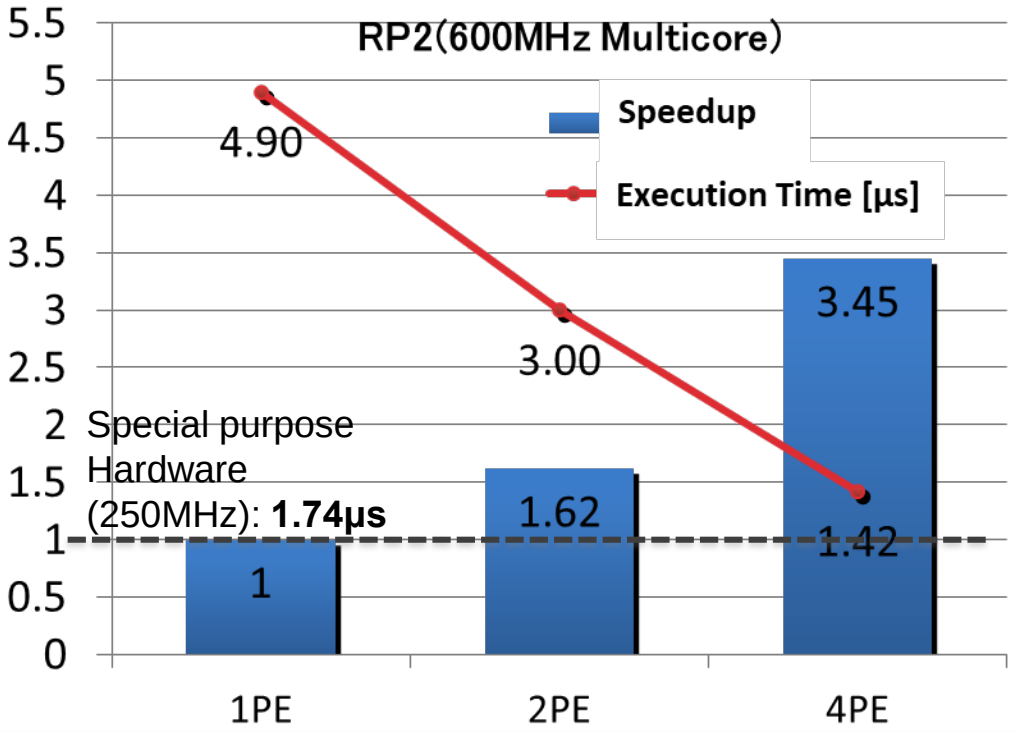
```
          PE1
; ---- Task A ----
; Task Body
  FADD R23, R19, R21
; ---- Task B ----
; Flag Check : Task C
L18:
  LDR R28, [R14, 0]
  CMP R28, R0
  JNE  L18
; Data Receive
  LDR R29, [R14, 1]
; Task Body
  FMLT R24, R23, R29
```

```
          PE2
; ---- Task C ----
; Task Body
  FMLT R27, R28, R29
  FSUB R29, R19, R27
; Data Transfer
  STR [R14, 1], R29
; Flag Set
  STR [R14, 0], R0
```

【W-CDMA Base Band Communication】

Near Fine Grain Parallel Processing of EAICH Detection Program on RP2 Multicore with 4 SH4A cores

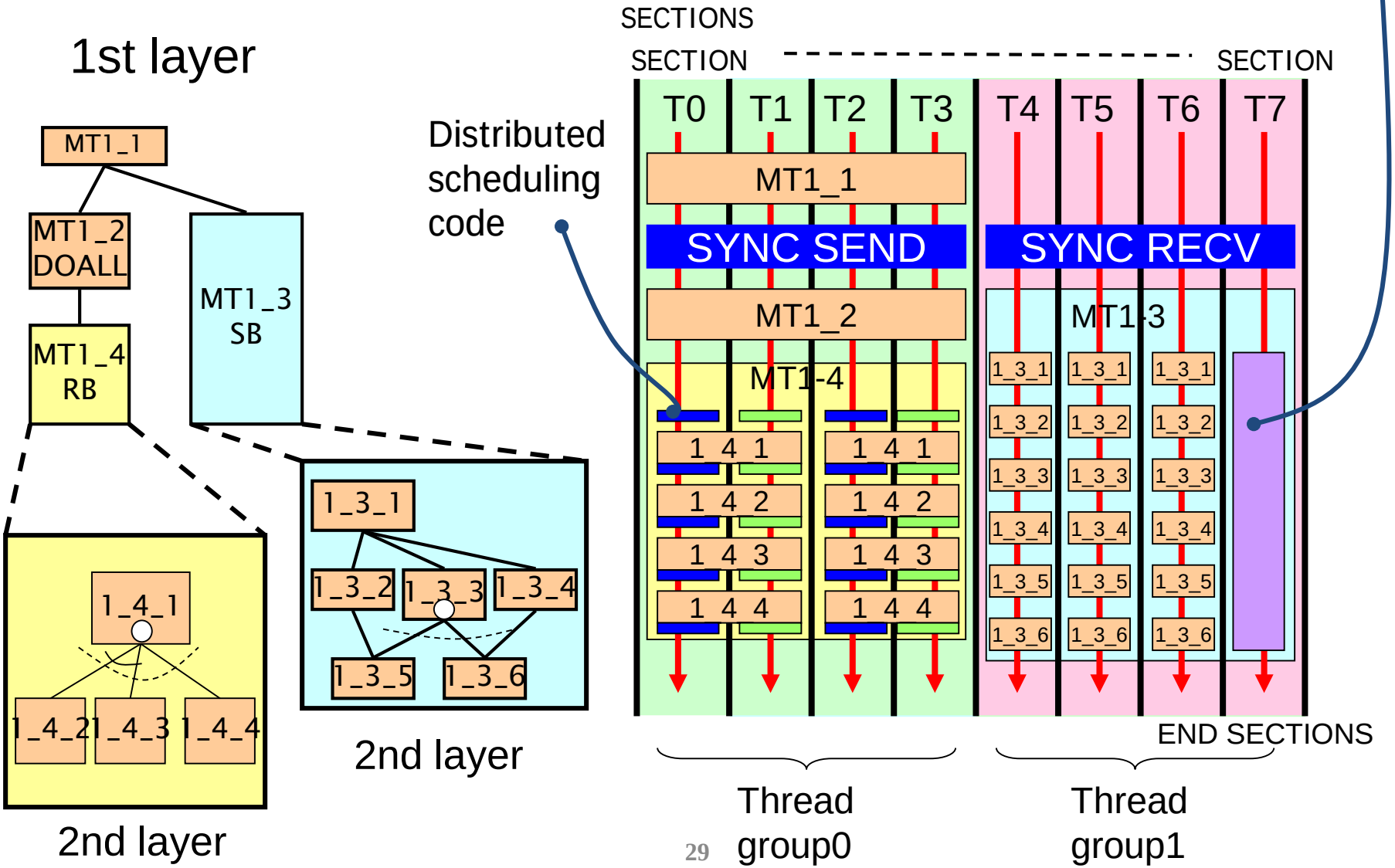
- **Hadamard transform often used in the signal processing**
- **Parallel Processing Method**
  - **Near fine grain parallel processing among statements**
  - **Static Scheduling**



1.62 times speedup for 2cores, 3.45 times speedup for 4 cores for EAICH on RP2.

Generated Multigrain Parallelized Code

(The nested coarse grain task parallelization is realized by only OpenMP "section", "Flush" and "Critical" directives.)



Code Generation Using OpenMP

- Compiler generates a parallelized program using OpenMP API
- One time single level thread generation
  - Threads are forked only once at the beginning of a program by OpenMP "PARALLEL SECTIONS" directive
  - Forked threads join only once at the end of program
- Compiler generates codes for each threads using static or dynamic scheduling schemes
- Extension of OpenMP for hierarchical processing is not required

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

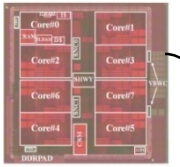
Accelerator 2
Code

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

Generation of parallel machine codes using sequential compilers



Homegeneous Multicore s from Vendor A (SMP servers)

Heterogeneous Multicores from Vendor B

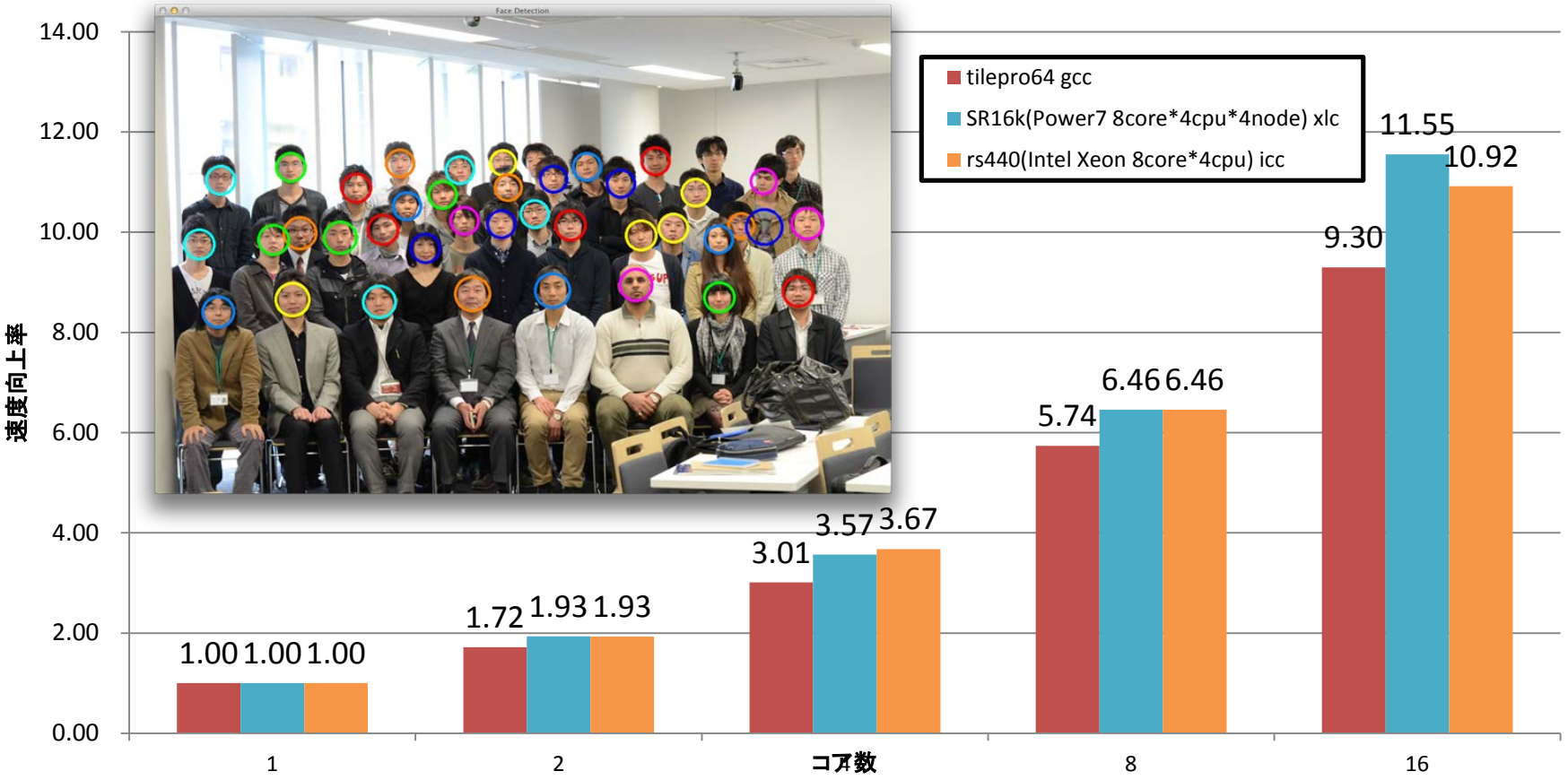
Shred memory servers

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

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

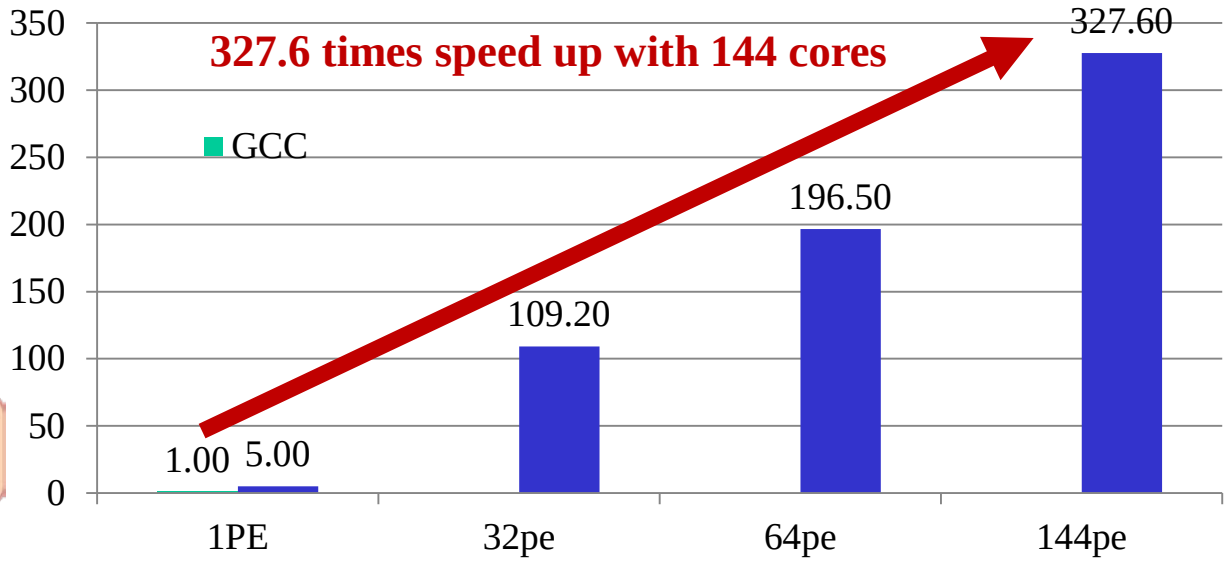


- OSCAR compiler gives us 11.55 times speedup for 16 cores against 1 core on SR16000 Power7 highend server.

Performance on Multicore Server for Latest Cancer Treatment Using Heavy Particle (Proton, Carbon Ion)

327 times speedup on 144 cores

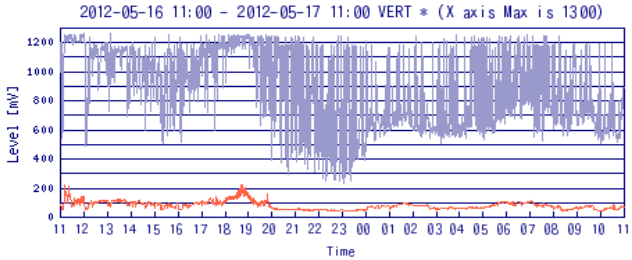
Hitachi 144cores SMP Blade Server BS500:
Xeon E7-8890 V3(2.5GHz　18core/chip) x8 chip

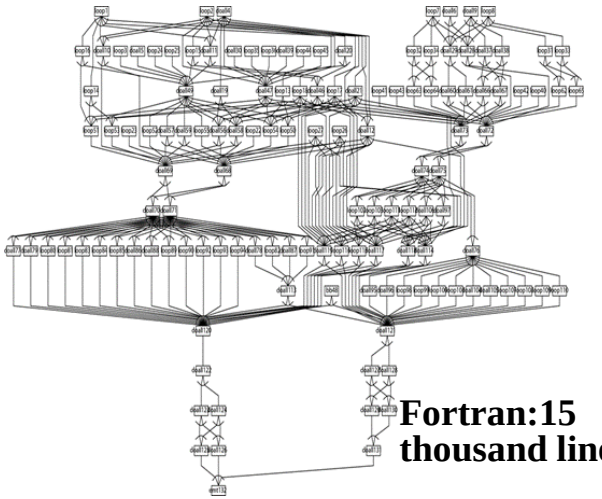


- **Original sequential execution time 2948 sec（50 minutes）using GCC was reduced to 9 sec with 144 cores（327.6 times speedup）**
  - **Reduction of treatment cost and reservation waiting period is expected**

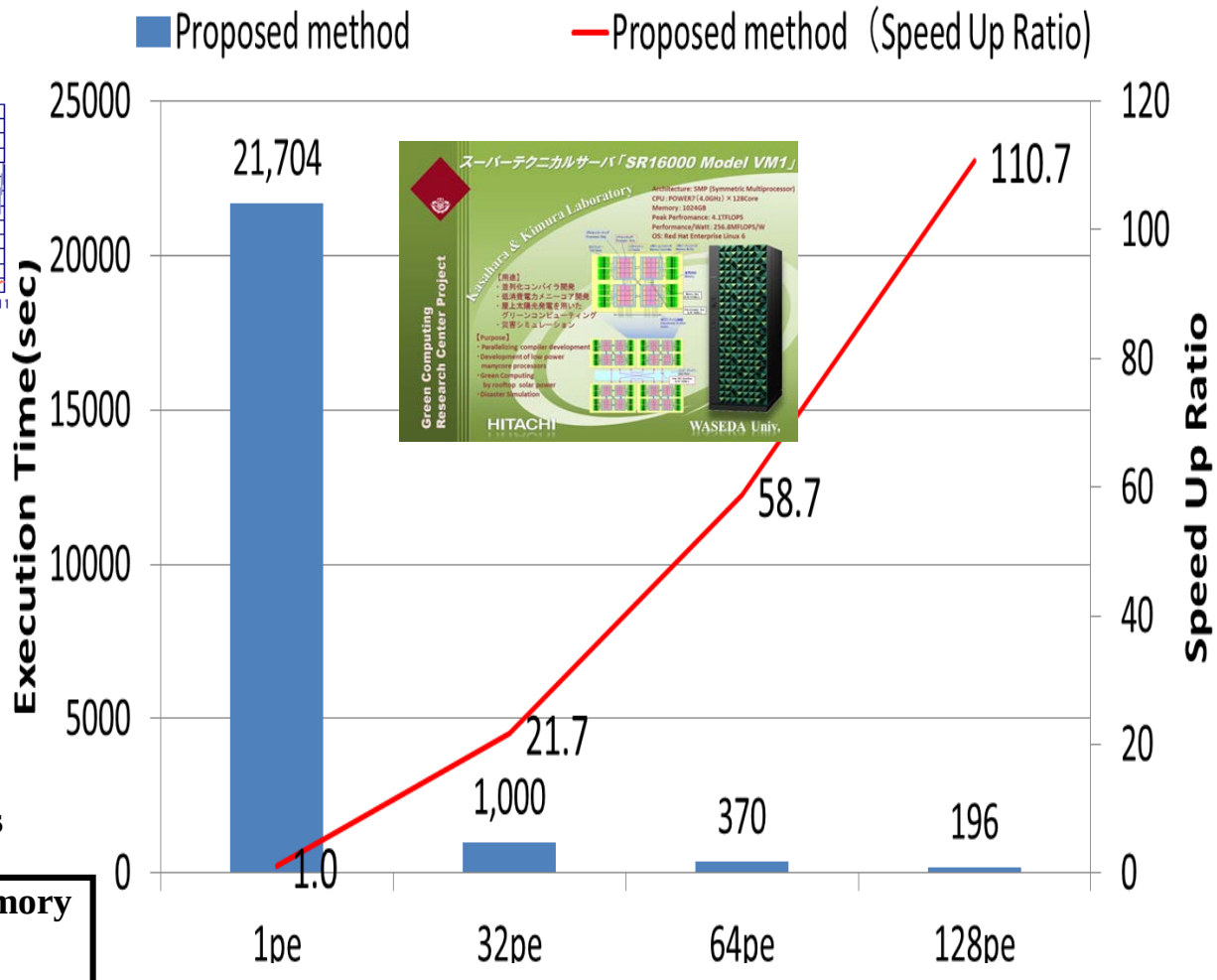
110 Times Speedup against the Sequential Processing for GMS Earthquake Wave Propagation Simulation on Hitachi SR16000

（Power7 Based 128 Core Linux SMP）





Fortran:15 thousand lines



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

Parallel Processing of JPEG XR Encoder on TILEPro64

Multimedia Applications: Sequential C Source Code

{ AAC Encoder
JPEG XR Encoder
Optical Flow Calc.

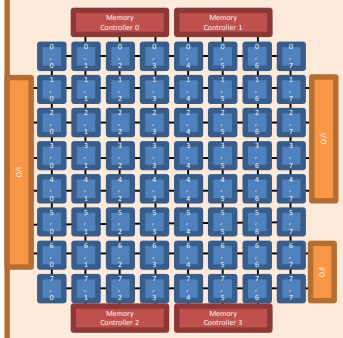
OSCAR Compiler

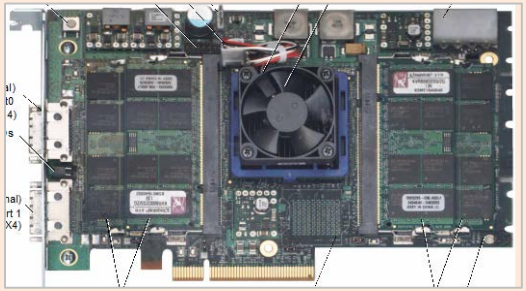
Parallelized C Program with OSCAR API

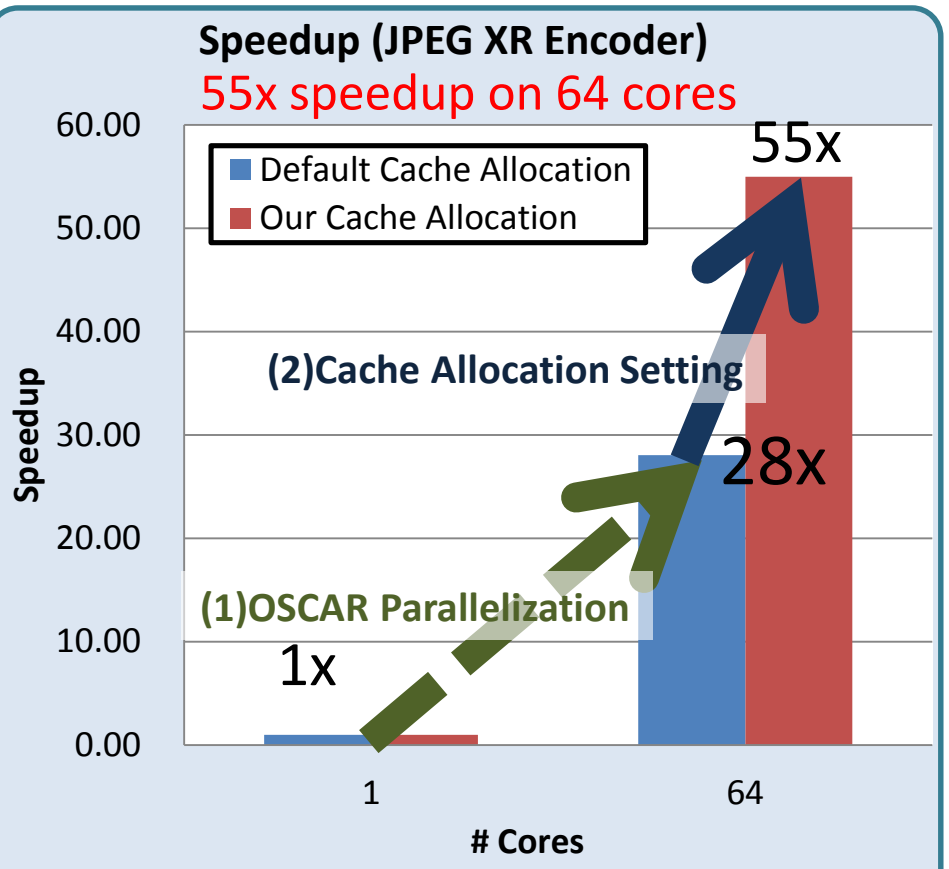
Cache Allocation Setting

API Analyzer + Sequential Compiler

Parallelized Executable Binary for TILEPro64

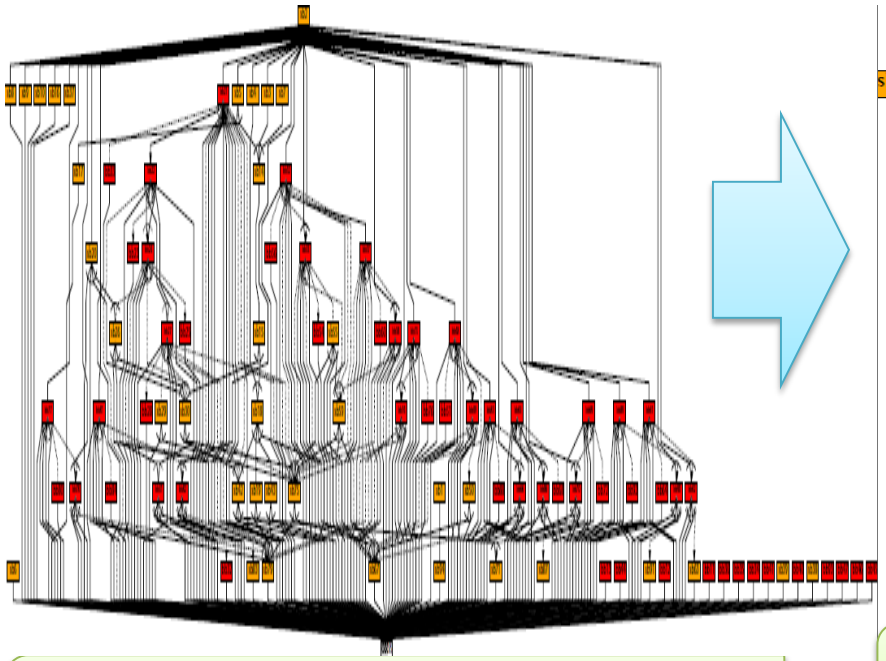




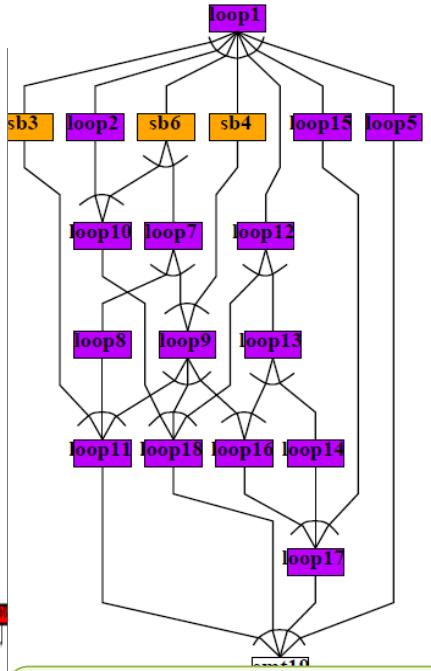


Local cache optimization:
Parallel Data Structure (tile) on heap allocating to local cache

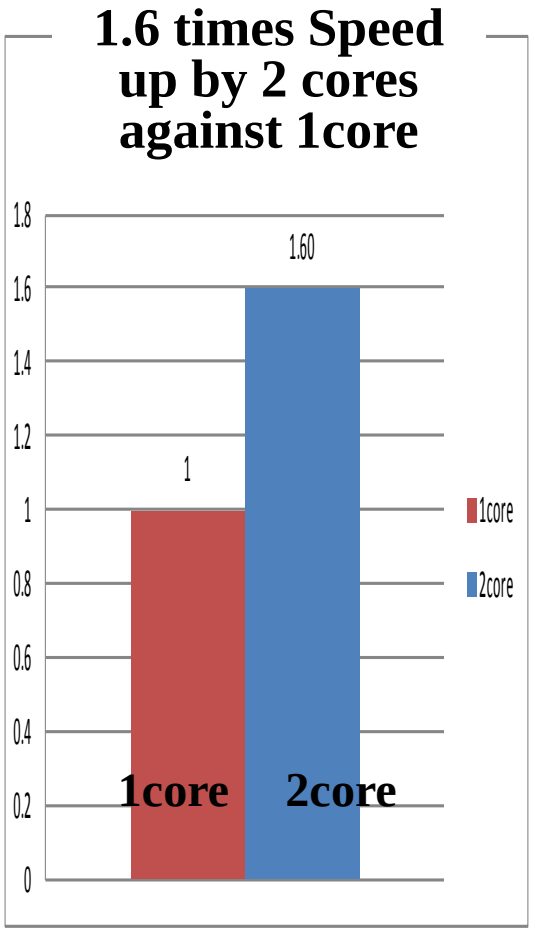
Speedup with 2cores for Engine Crankshaft Handwritten Program on RPX Multi-core Processor

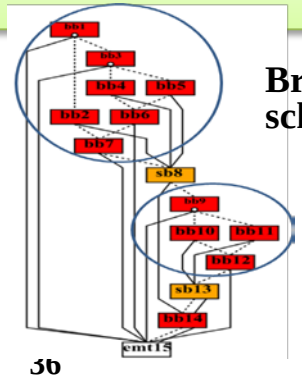


Macrotask graph with a lot of conditional branches

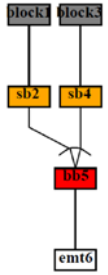


Macrotask graph after task fusion





Branches are fused to macrotasks for static scheduling



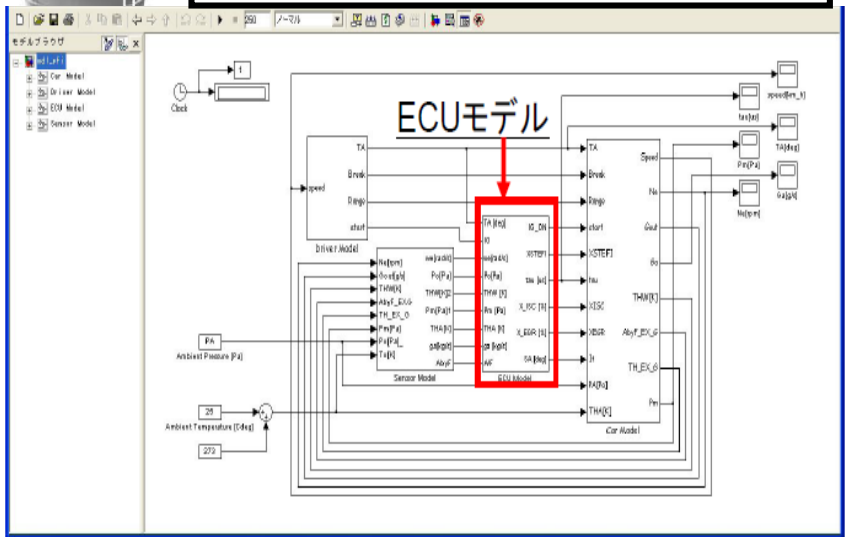
Grain is too fine (us) for dynamic scheduling.

Model Base Designed Engine Control on V850 Multicore with Denso

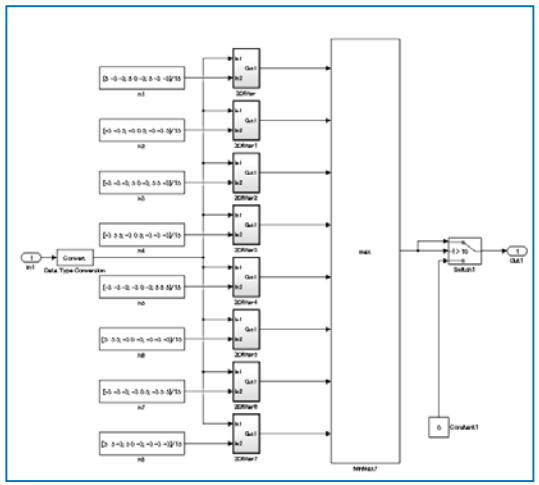
Though so far parallel processing of the engine control on multicore has been very difficult, Denso and Waseda succeeded 1.95 times speedup on 2core V850 multicore processor.

Hard real-time automobile engine control by multicore

C codes generated by MATLAB/Simulink embedded coder are automatically parallelized.



OSCAR Compile Flow for Simulink Applications



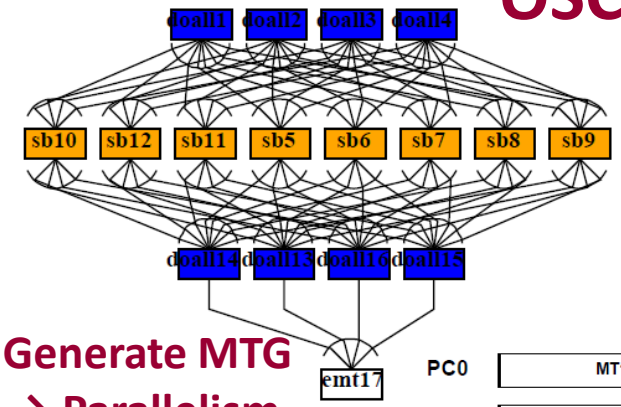
Simulink model

Generate C code
using Embedded Coder

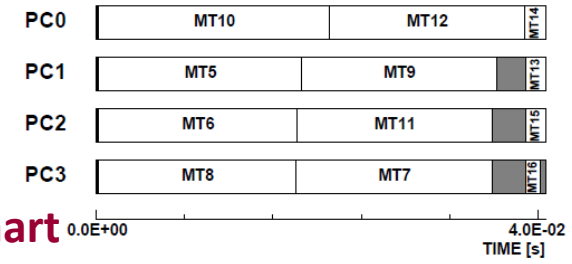
```
/* Model step function */
void VesselExtraction_step(void)
{
  int32_T i;
  real_T u0;

  /* DataTypeConversion: '<S1>/Data Type Conversion' incorporates:
   *  Inport: '<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



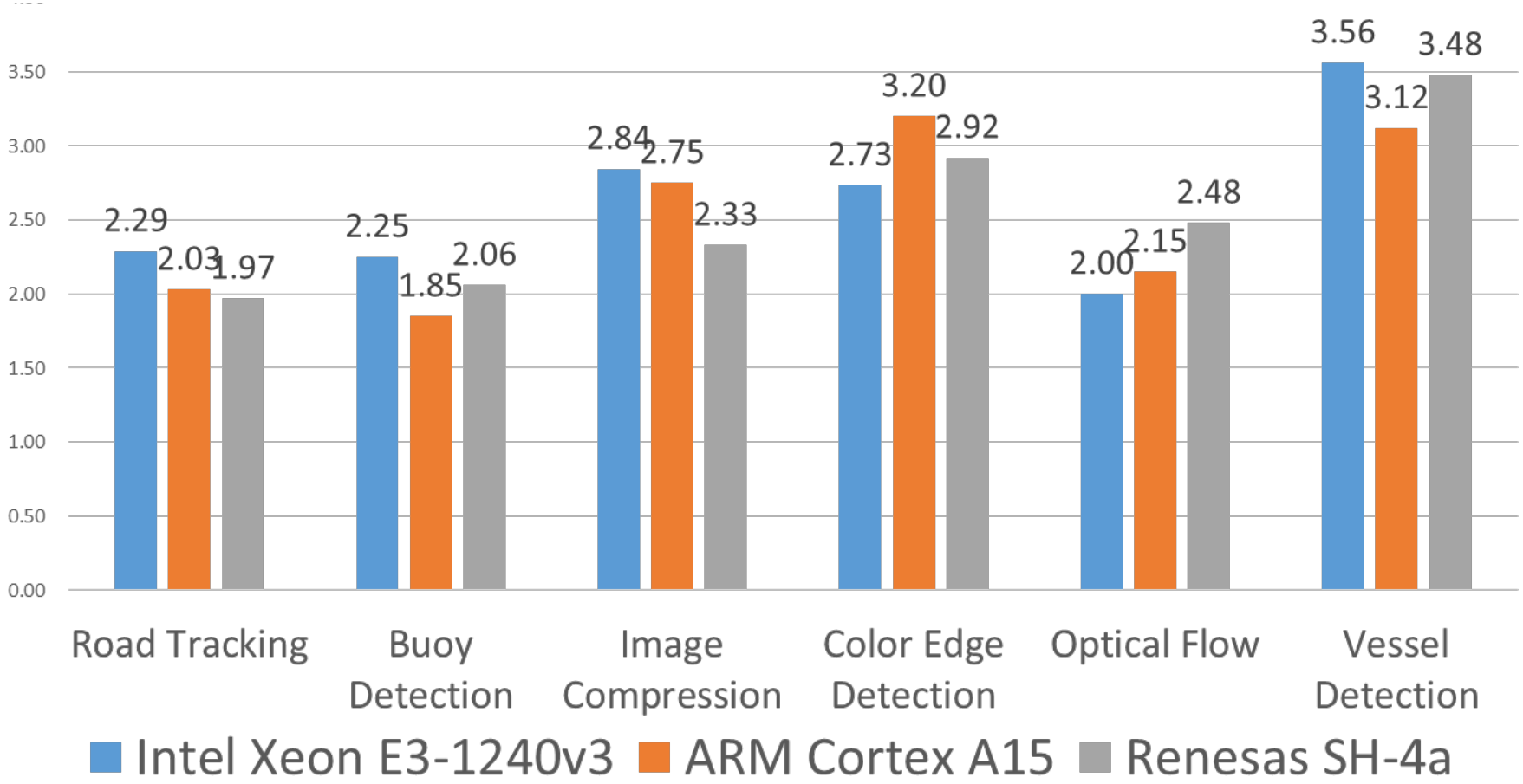
(2) Generate gantt chart
→ Scheduling in a multicore

```
void VesselExtraction_step ( )
{
    int thr1 ;
    int thr2 ;
    int thr3 ;
    {
        oscar_thread_create ( & thr1 ,
            thread_function_001 , (void*)1 ) ;
        oscar_thread_create ( & thr2 ,
            thread_function_002 , (void*)2 ) ;
        oscar_thread_create ( & thr3 ,
            thread_function_003 , (void*)3 ) ;

            VesselExtraction_step_PE0 ( ) ;

        oscar_thread_join ( thr1 ) ;
        oscar_thread_join ( thr2 ) ;
        oscar_thread_join ( thr3 ) ;
    }
}

void thread_function_001 ( void )
{
    VesselExtraction_step_PE1 ( ) ;
}
```

(3) Generate parallelized C code
using the OSCAR API
→ Multiplatform execution
(Intel, ARM and SH etc)

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

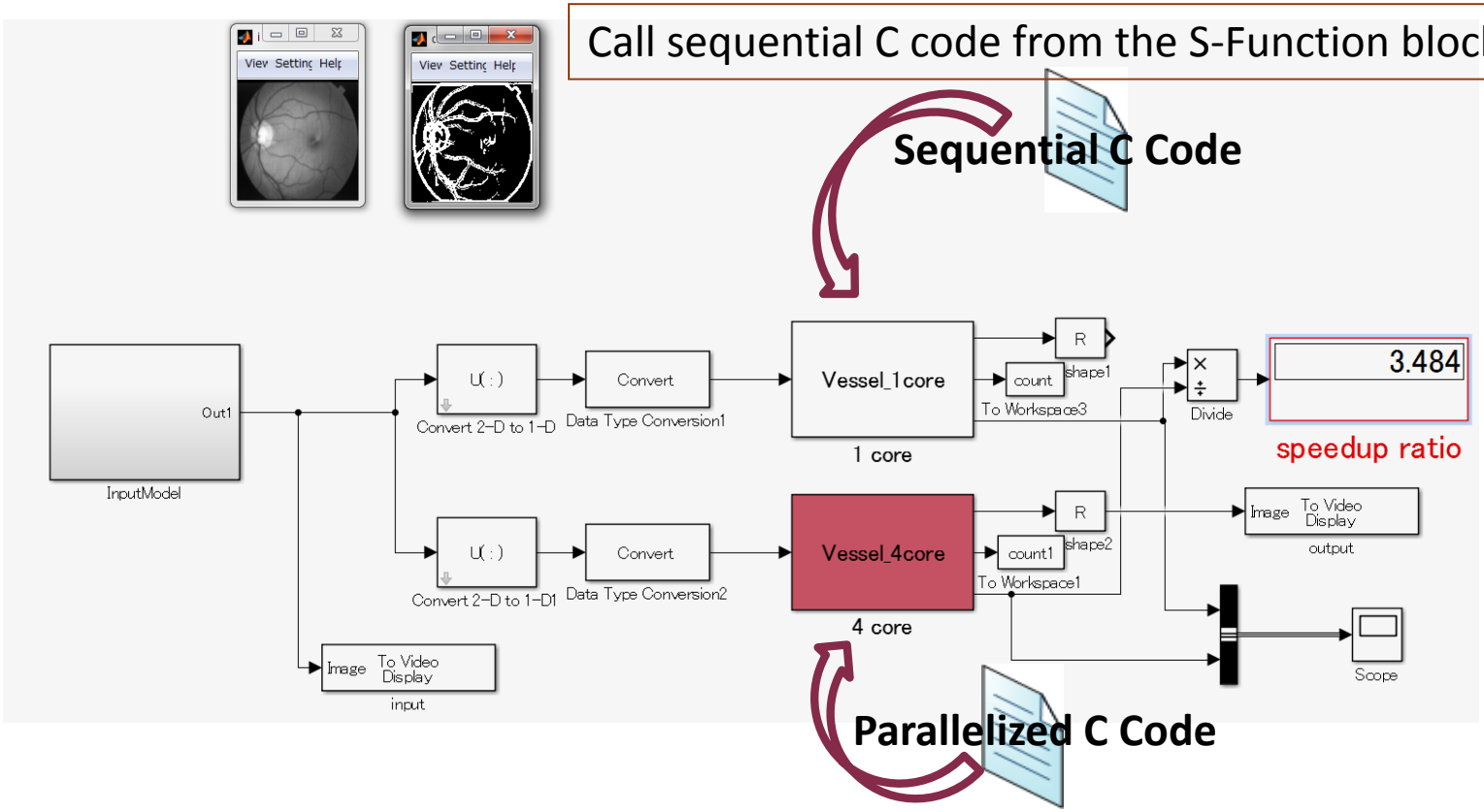
(Intel Xeon, ARM Cortex A15 and Renesas SH4A)



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/

Parallel Processing on Simulink Model

- The parallelized C code can be embedded to Simulink using C mex API for HILS and SILS implementation.



OSCAR API Ver. 2.0 for Homogeneous/Heterogeneous Multicores and Manycores

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

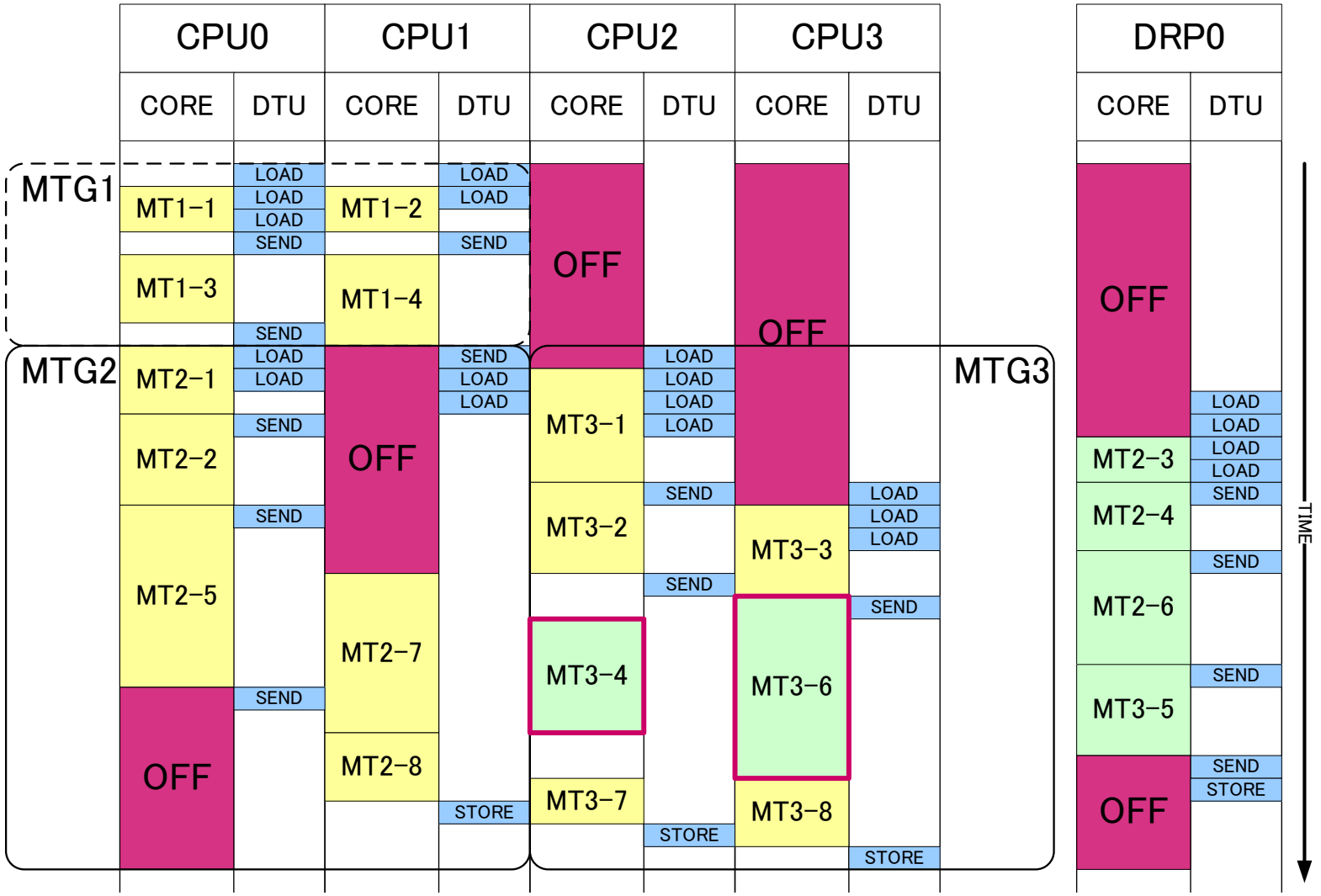
(* from OpenMP)

- 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

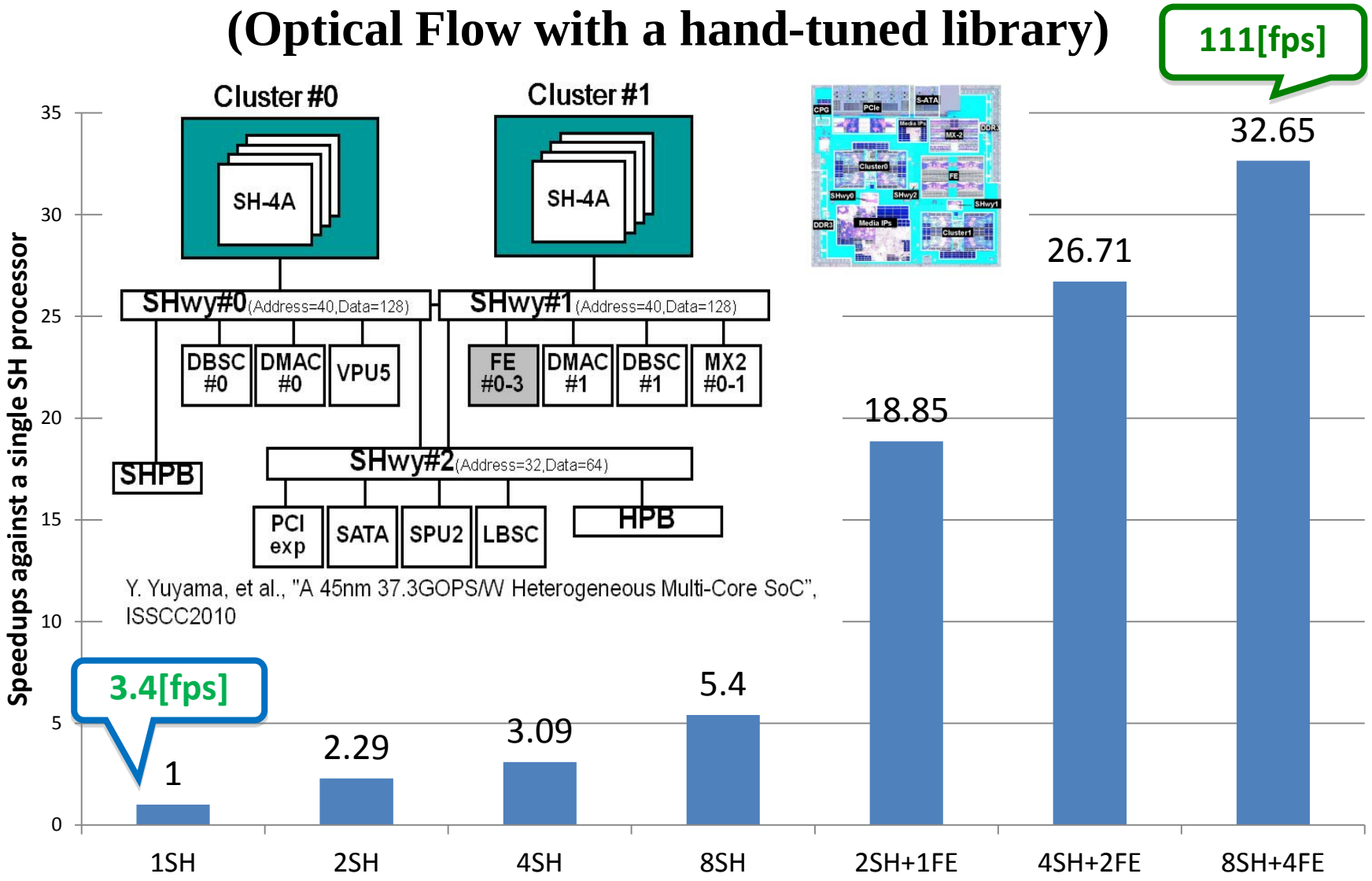
from V2.0

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

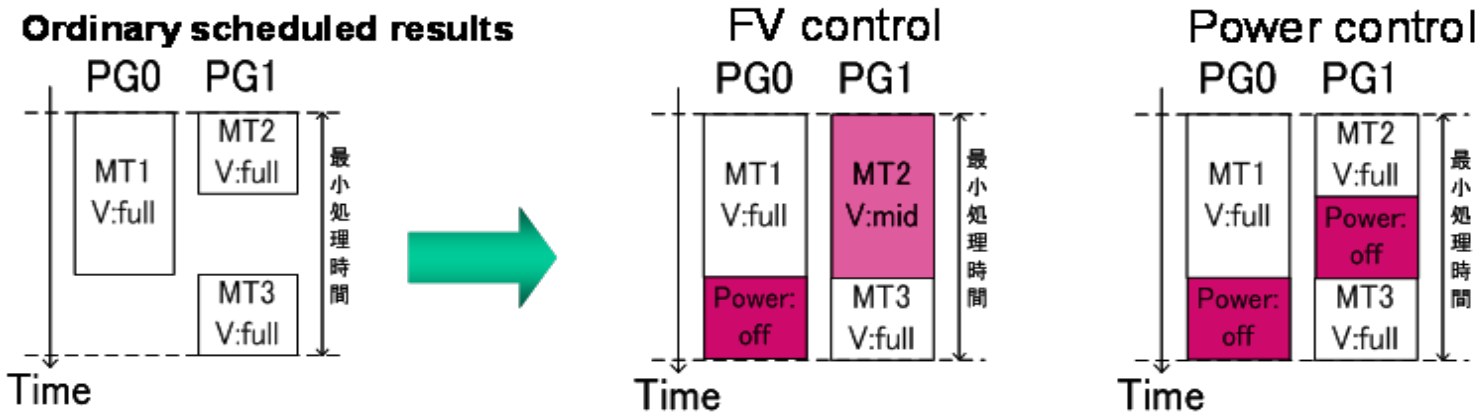
(Optical Flow with a hand-tuned library)



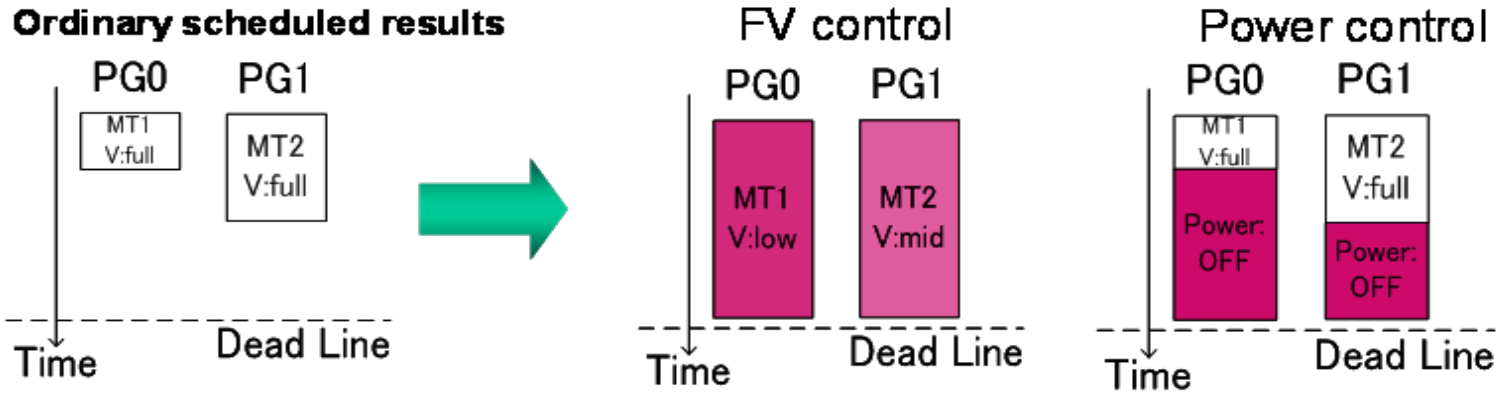
Power Reduction by Power Supply, Clock Frequency and Voltage Control by OSCAR Compiler

Frequency and Voltage (DVFS), Clock and Power gating of each cores are scheduled considering the task schedule since the dynamic power proportional to the cube of F (F^3) and the leakage power (the static power) can be reduced by the power gating (power off).

- **Shortest execution time mode**



- Realtime processing mode with dead line constraints



In this Fig. Frequency Full, Mid, Low

Power OFF: Power Gating

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**

state	FULL	MID	LOW	OFF
frequency	1	1 / 2	1 / 4	0
voltage	1	0.87	0.71	0
dynamic energy	1	3 / 4	1 / 2	0
static power	1	1	1	0

- **State transition overhead**

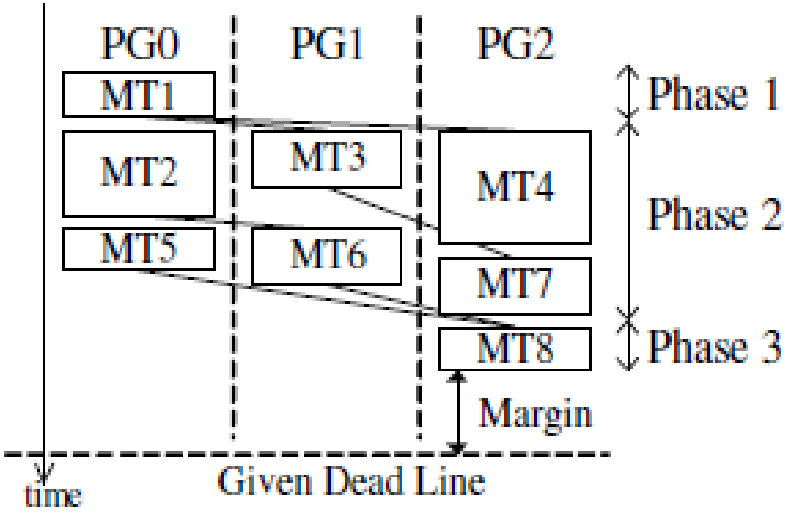
state	FULL	MID	LOW	OFF
FULL	0	40k	40k	80k
MID	40k	0	40k	80k
LOW	40k	40k	0	80k
OFF	80k	80k	80k	0

delay time [u.t.]

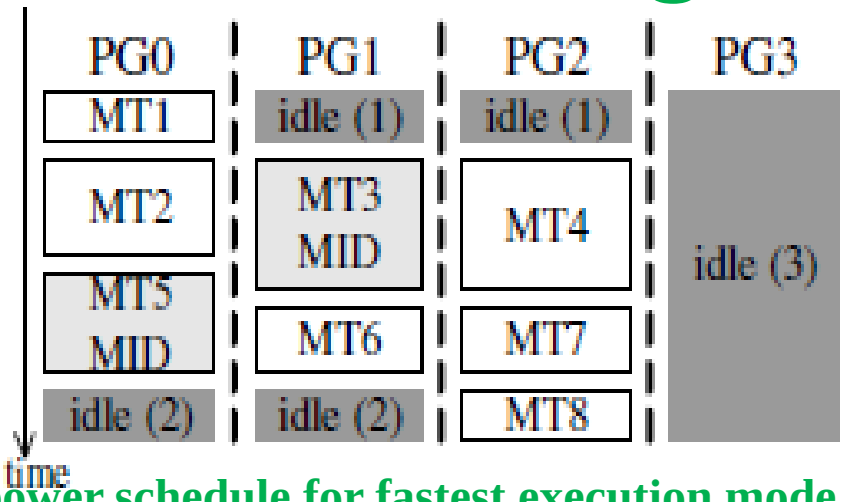
state	FULL	MID	LOW	OFF
FULL	0	20	20	40
MID	20	0	20	40
LOW	20	20	0	40
OFF	40	40	40	0

energy overhead [μJ]

Power Reduction Scheduling



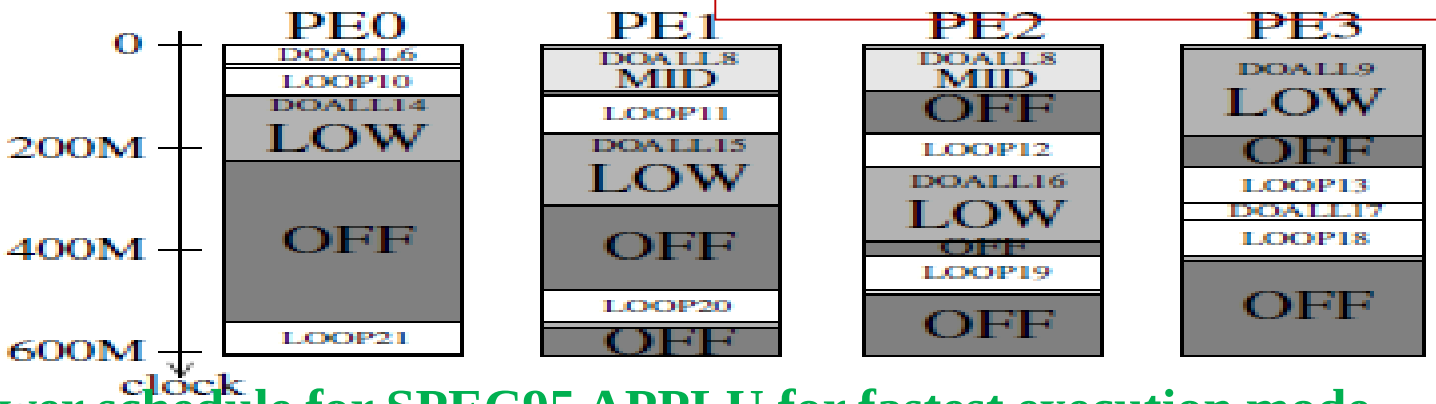
A macrotask graph assigned to 3 cores



A power schedule for fastest execution mode

Realtime scheduling mode
MTs 1,4,7,8 are on Critical Path (CP)

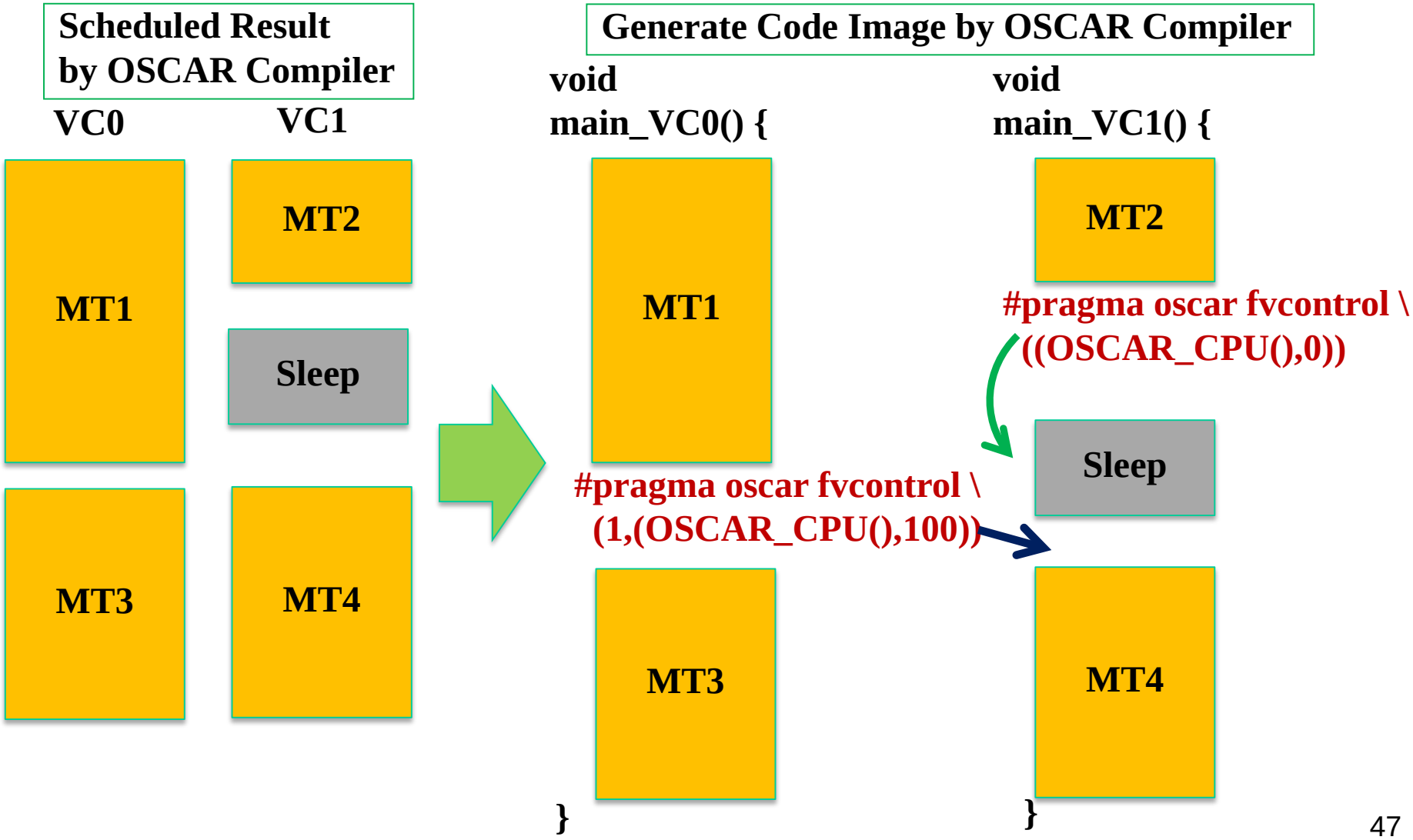
1) **Reduce frequencies (Fs) of MTs on CP considering dead line.**
2) **Reduce Fs of MTs not on CP. Idle: Clock or Power Gating considering overheads.**



A power schedule for SPEC95 APPLU for fastest execution mode

Doall6, Loop 10,11,12,13, Doall 17, Loop 18,19, 20, 21 are on CP

Low-Power Optimization with OSCAR API



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

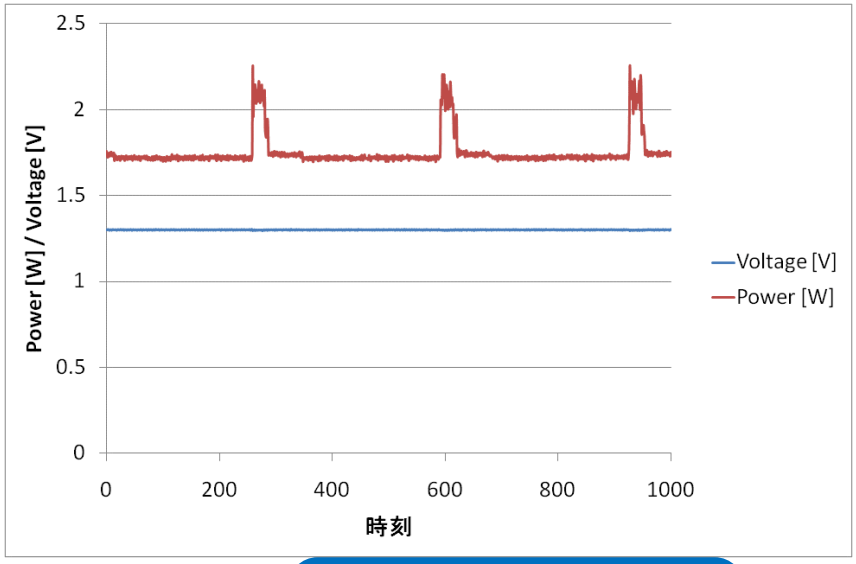
Without Power Reduction

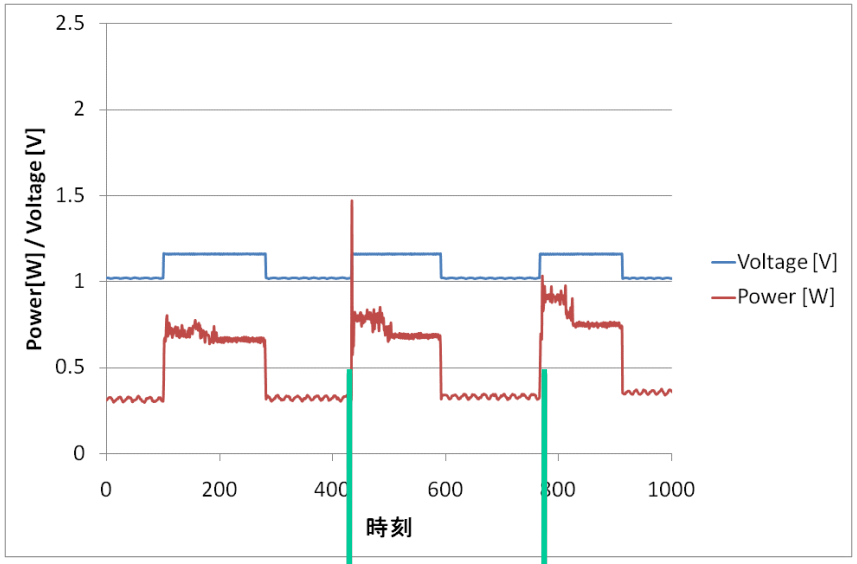
With Power Reduction by OSCAR Compiler

70% of power reduction

Average:1.76[W]

Average:0.54[W]



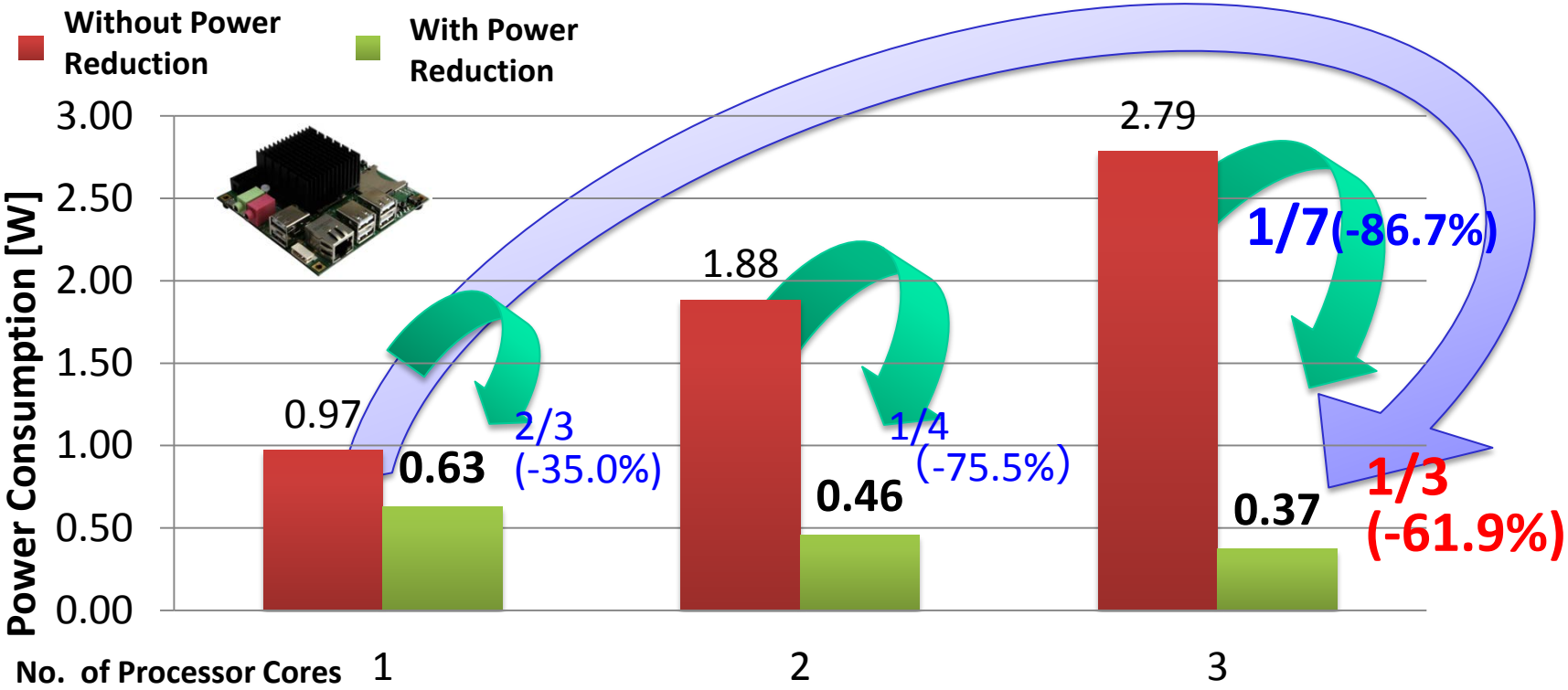


1cycle : 33[ms]
→30[fps]

Automatic Power Reduction for MPEG2 Decode on Android Multicore

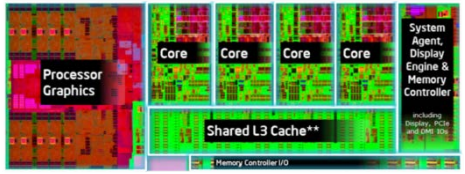
ODROID X2 ARM Cortex-A9 4 cores

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



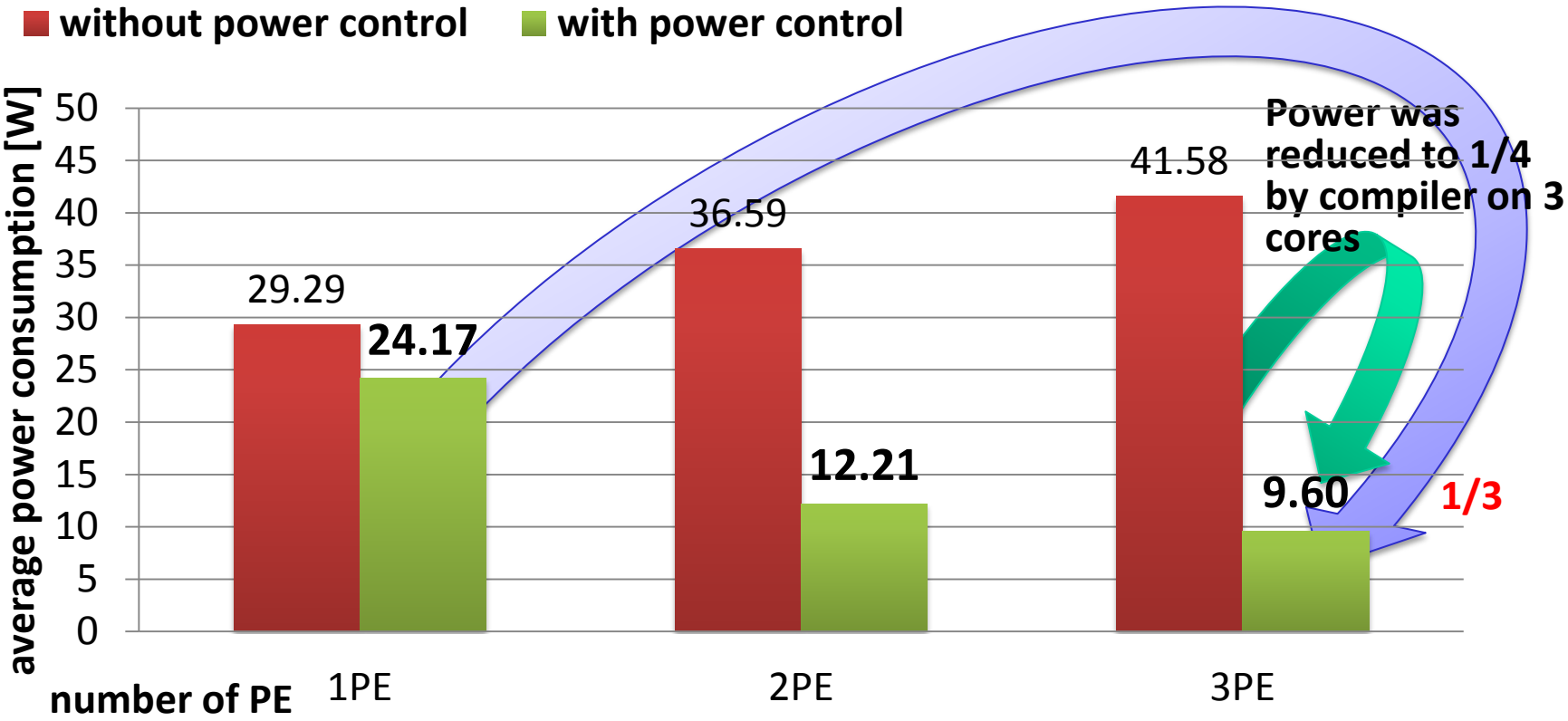
- On 3 cores, Automatic Power Reduction control successfully reduced power to 1/7 against without Power Reduction control.
- 3 cores with the compiler power reduction control reduced power to 1/3 against ordinary 1 core execution.

Power Reduction on Intel Haswell for Real-time Optical Flow



Intel CPU Core i7 4770K

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



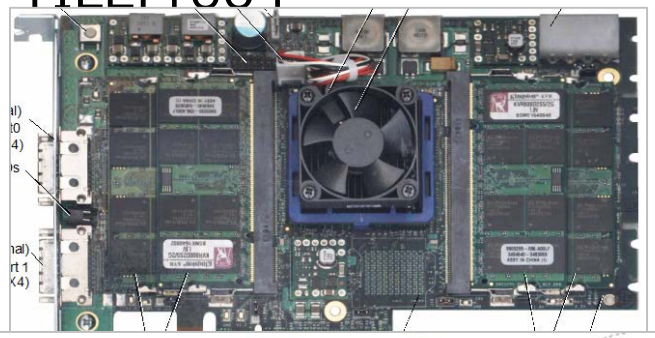
Power was reduced to 1/4 (9.6W) by the compiler power optimization on the same 3 cores (41.6W).

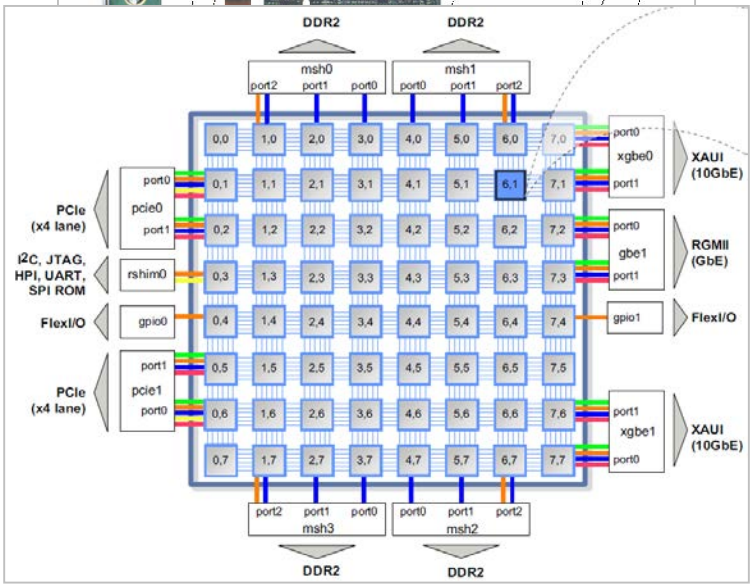
Power with 3 core was reduced to 1/3 (9.6W) against 1 core (29.3W) .

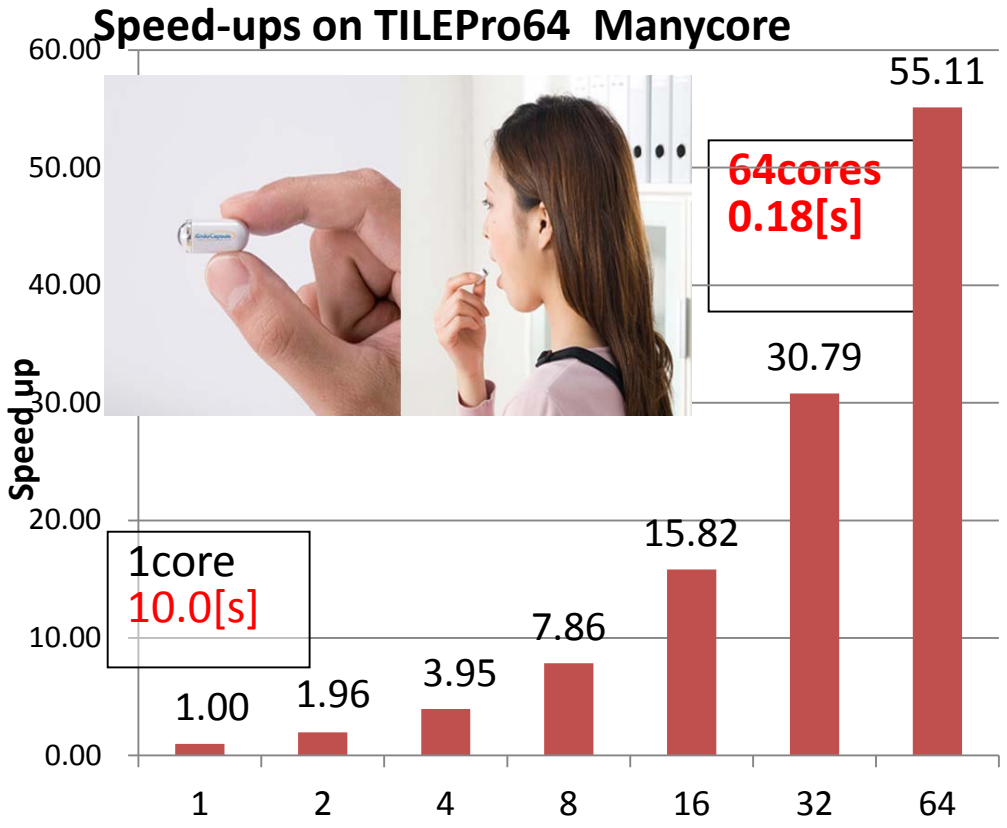
Automatic Parallelization of JPEG-XR for Drinkable Inner Camera (Endo Capsule)

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

- TILEPro64

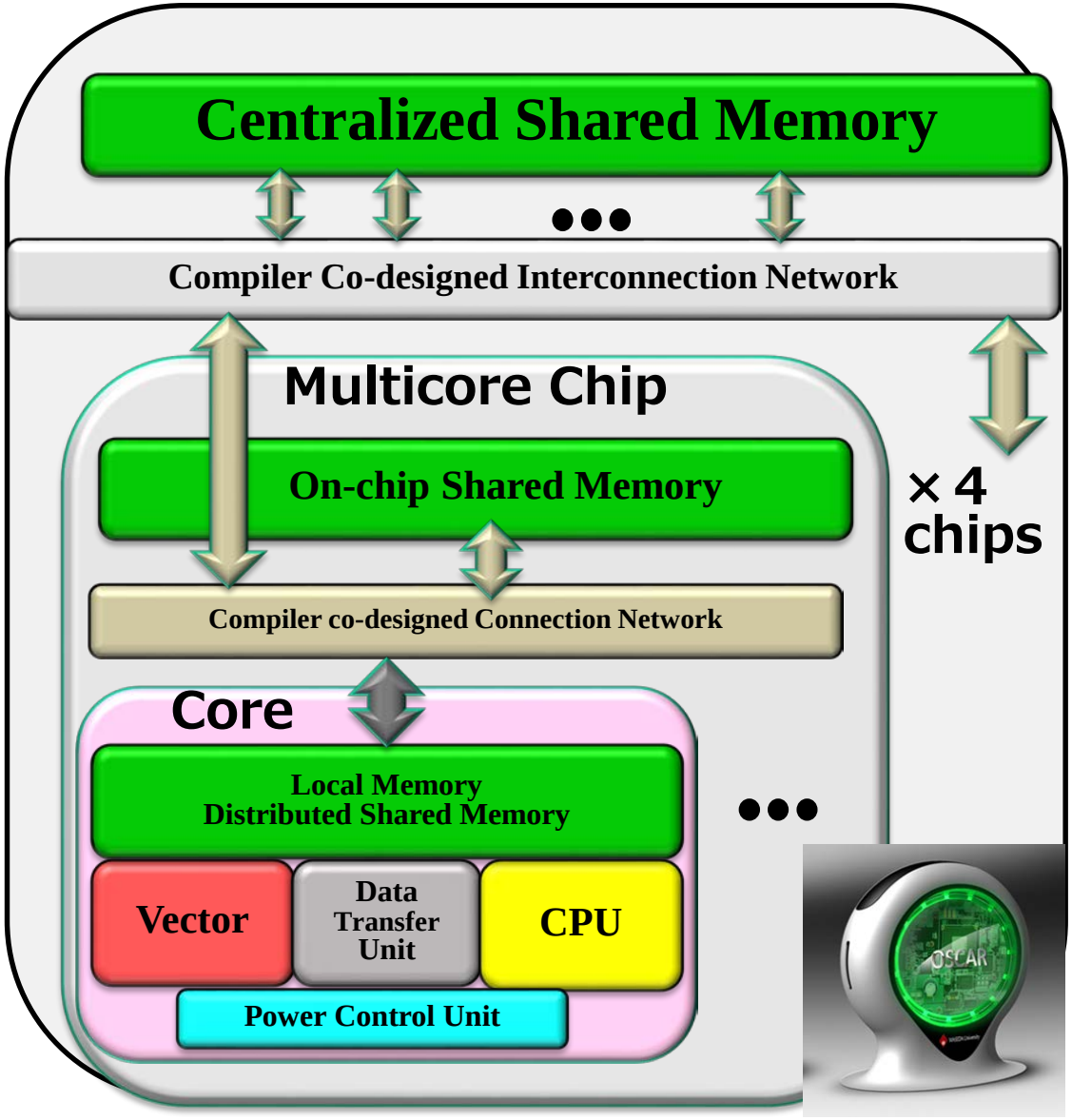


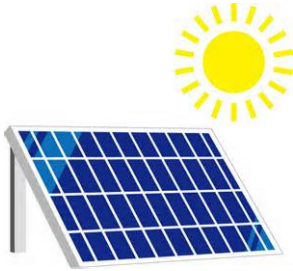




55 times speedup with 64 cores

OSCAR Vector Multicore and Compiler for Embedded to Severs with OSCAR Technology

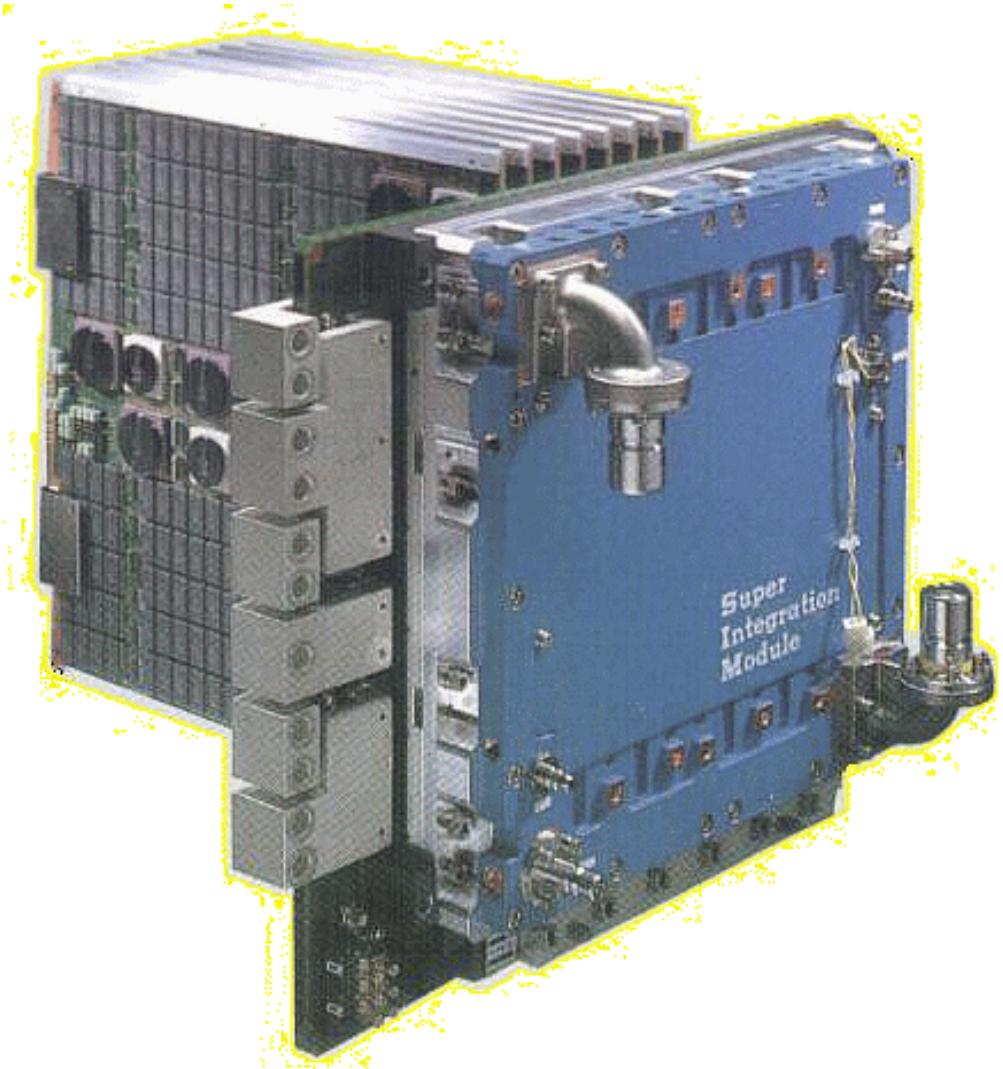




Target:

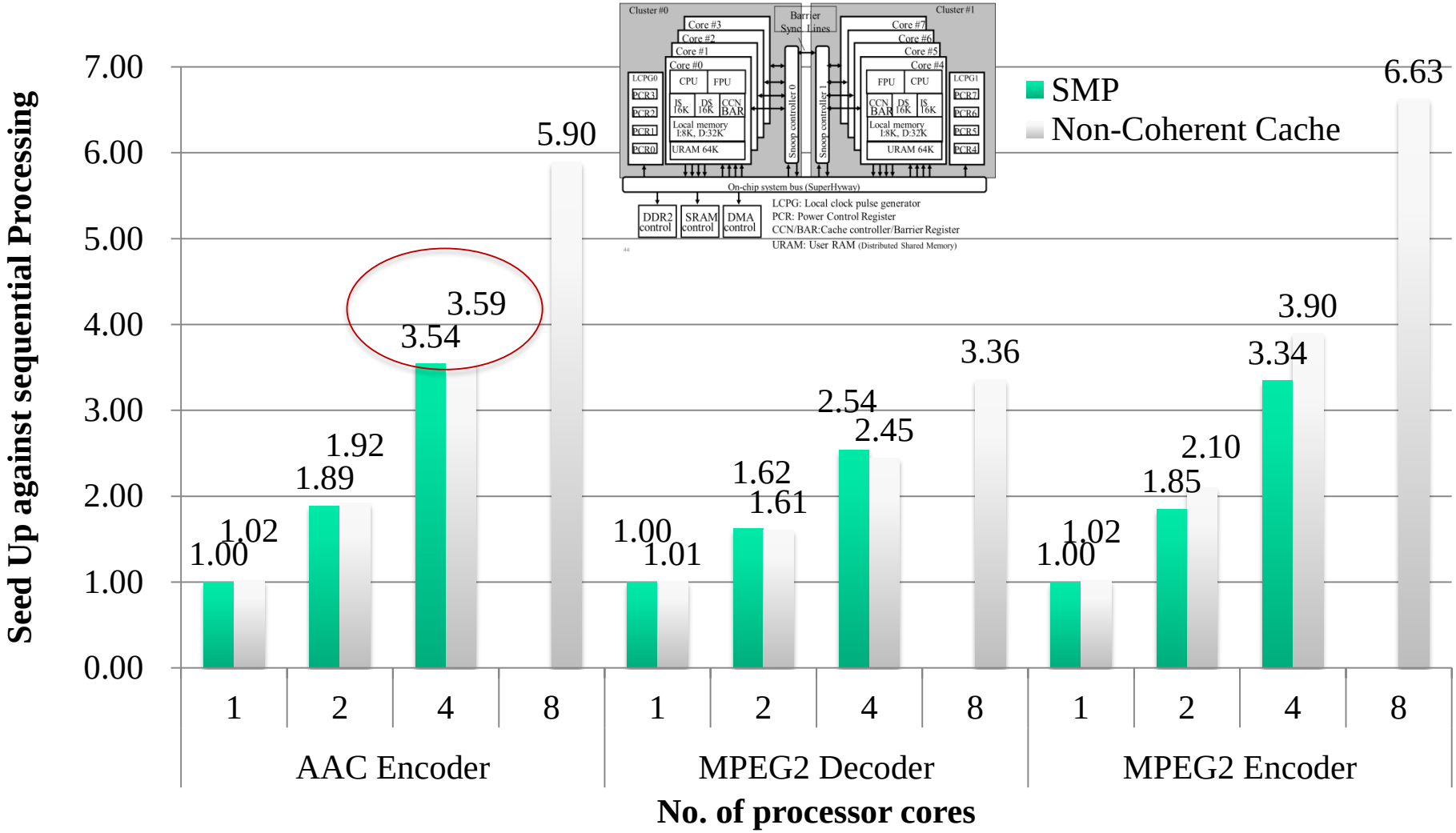
- **Solar Powered with compiler power reduction.**
- **Fully automatic parallelization and vectorization including local memory management and data transfer.**

Fujitsu VPP500/NWT: PE Unit



Performance of OSCAR Compiler Software Coherence Control

- **Faster or Equal Processing Performance up to 4cores with hardware coherent mechanism on RP2.**
- **Software Coherence gives us correct execution without hardware coherence mechanism on 8 cores.**

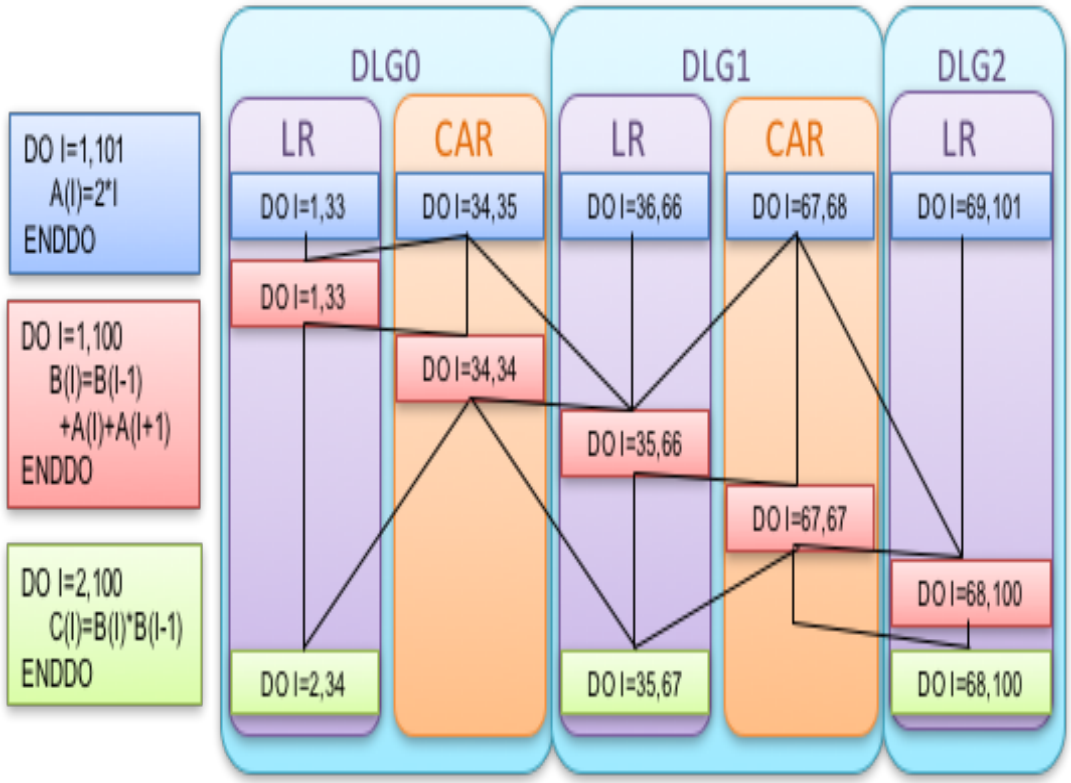


Automatic Local Memory Management

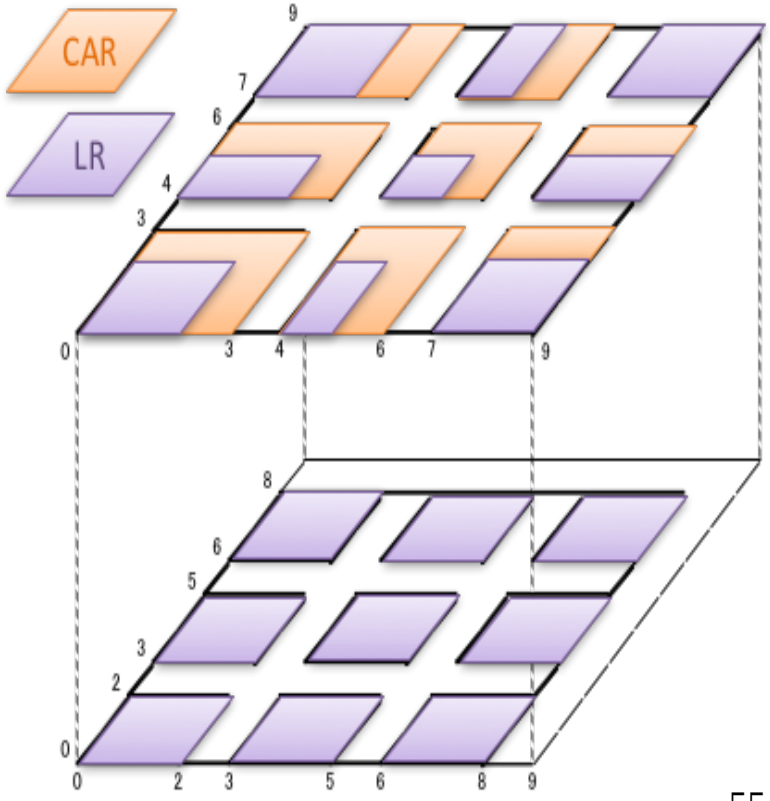
Data Localization: Loop Aligned Decomposition

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

Single dimension Decomposition

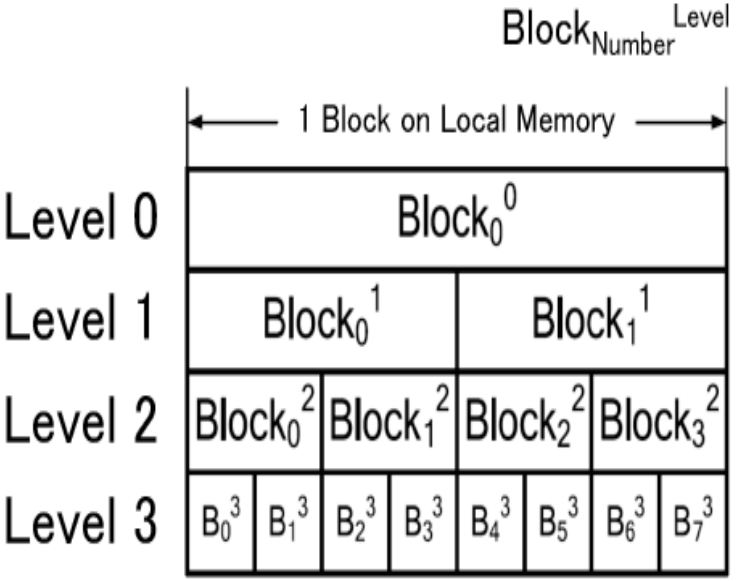


Multi-dimension Decomposition



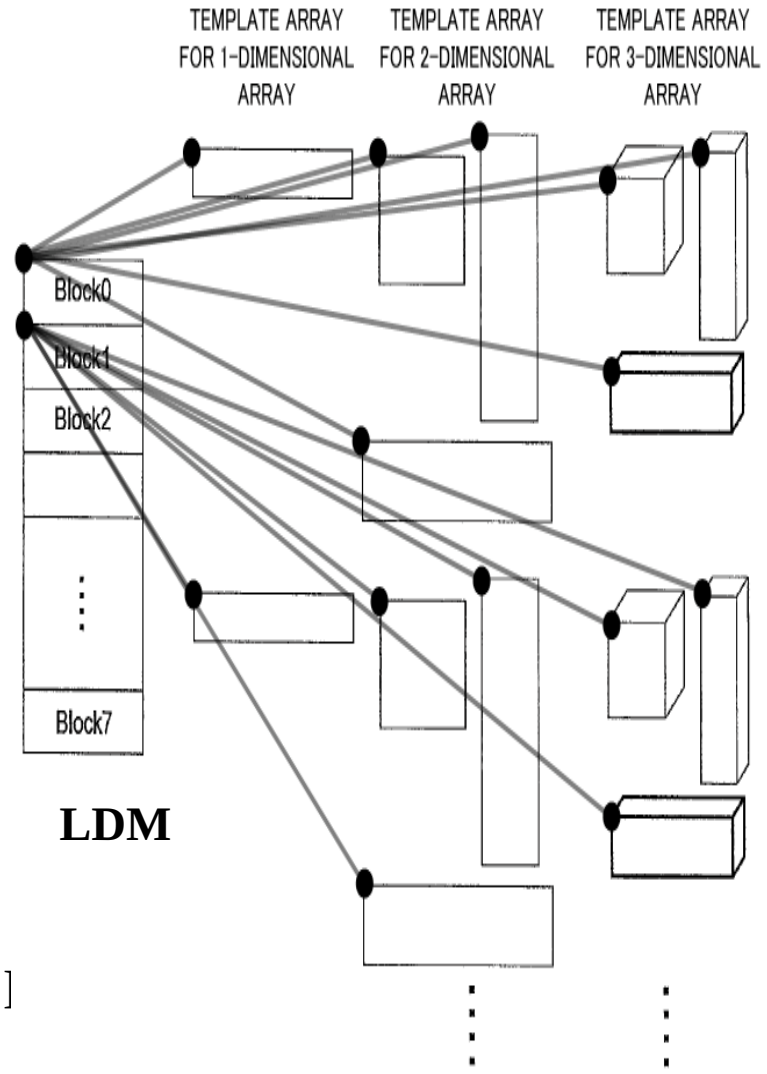
Adjustable Blocks

- Handling a suitable block size for each application
  - different from a fixed block size in cache
  - each block [illegible] blocks with integer divi[illegible] arrays and scalar varia[illegible]



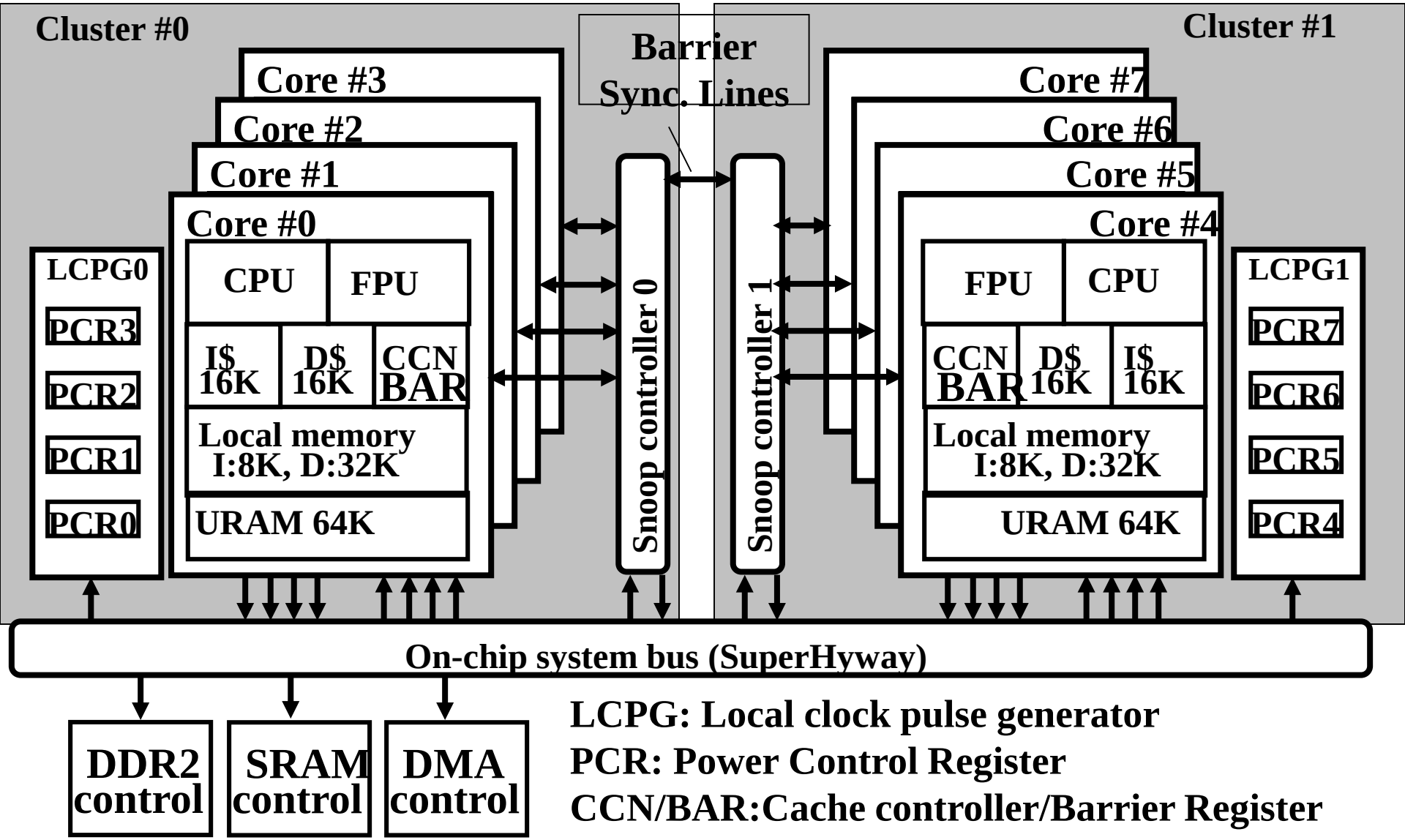
Multi-dimensional Template Arrays for Improving Readability

- 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] -> TA[offset + i' * L + j' * M + k']
  - Template Arrays provide readability
    - A[i][j][k] -> TA[Block#][i'][j'][k']



LDM

8 Core RP2 Chip Block Diagram



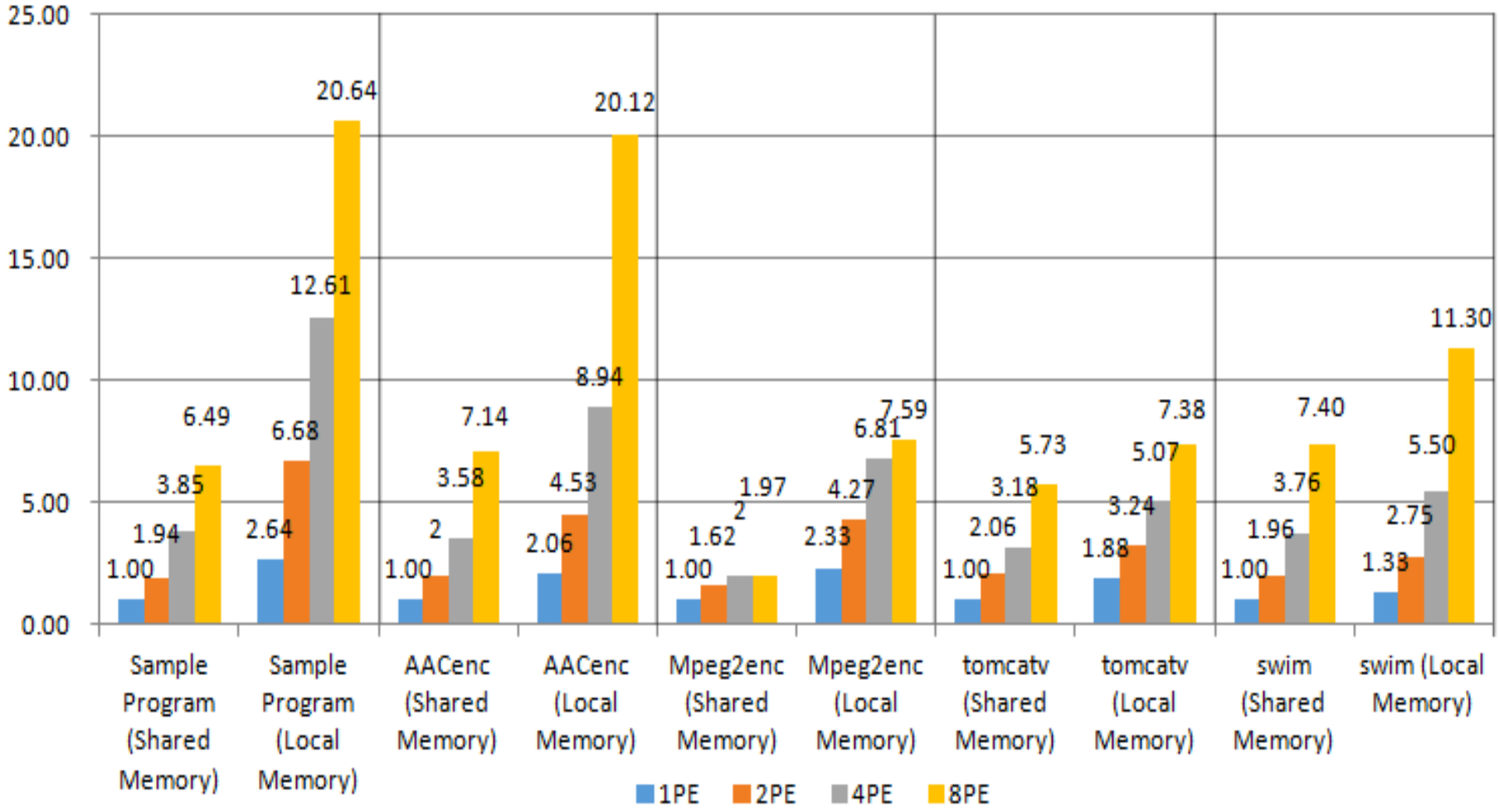
LCPG: Local clock pulse generator

PCR: Power Control Register

CCN/BAR:Cache controller/Barrier Register

URAM: User RAM (Distributed Shared Memory)

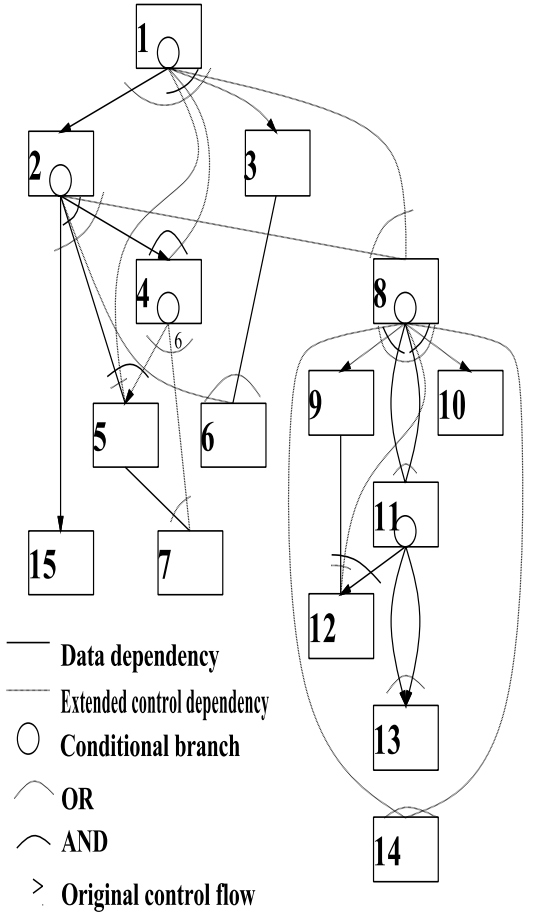
Speedups by the Local Memory Management Compared with Utilizing Shared Memory on Benchmarks Application using RP2



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

Software Coherence Control Method on OSCAR Parallelizing Compiler

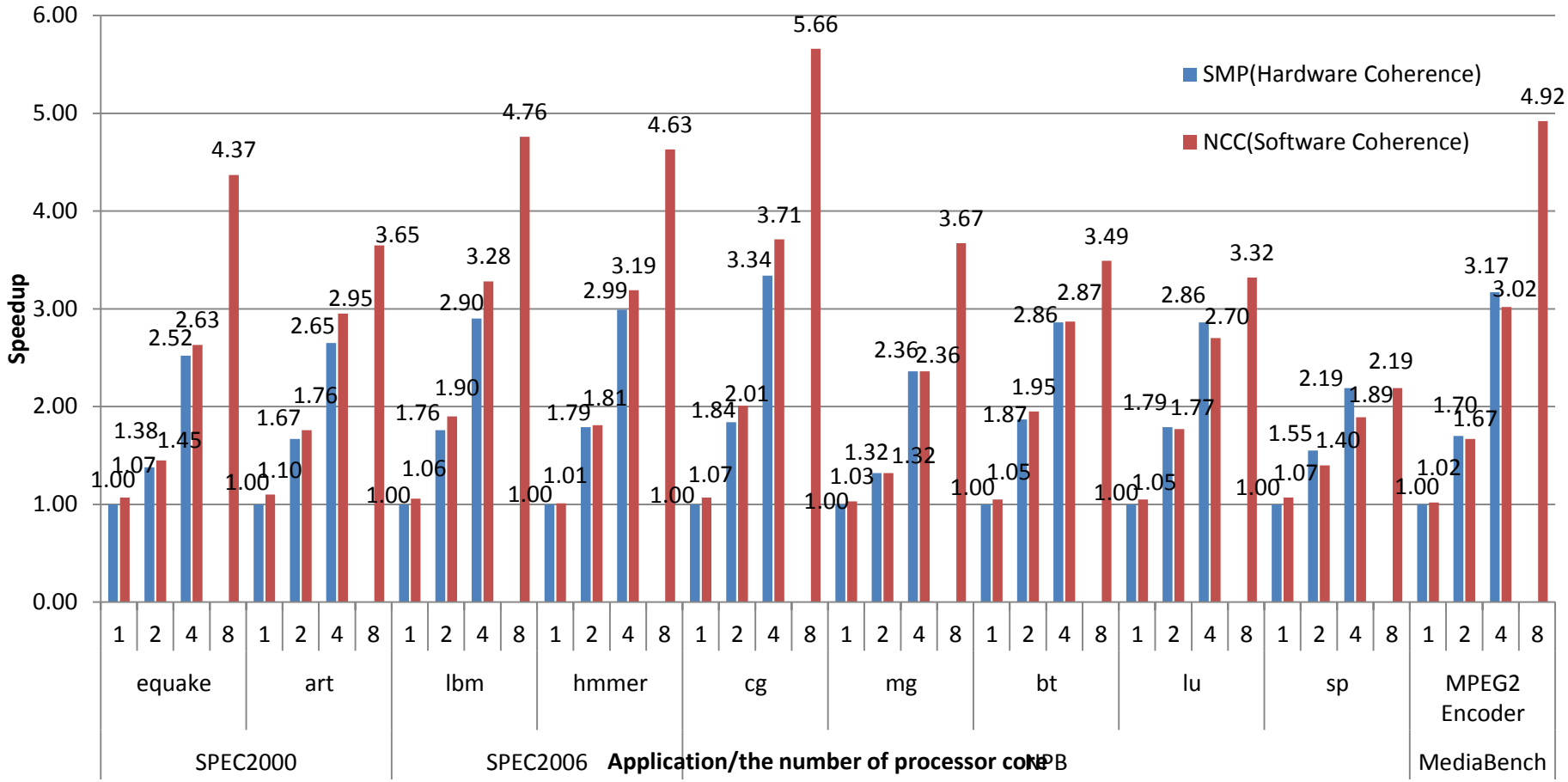
- Coarse grain task parallelization with earliest condition analysis (control and data dependency analysis to detect parallelism among coarse grain tasks).
- OSCAR compiler automatically controls coherence using following simple program restructuring methods:
  - To cope with stale data problems:
  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



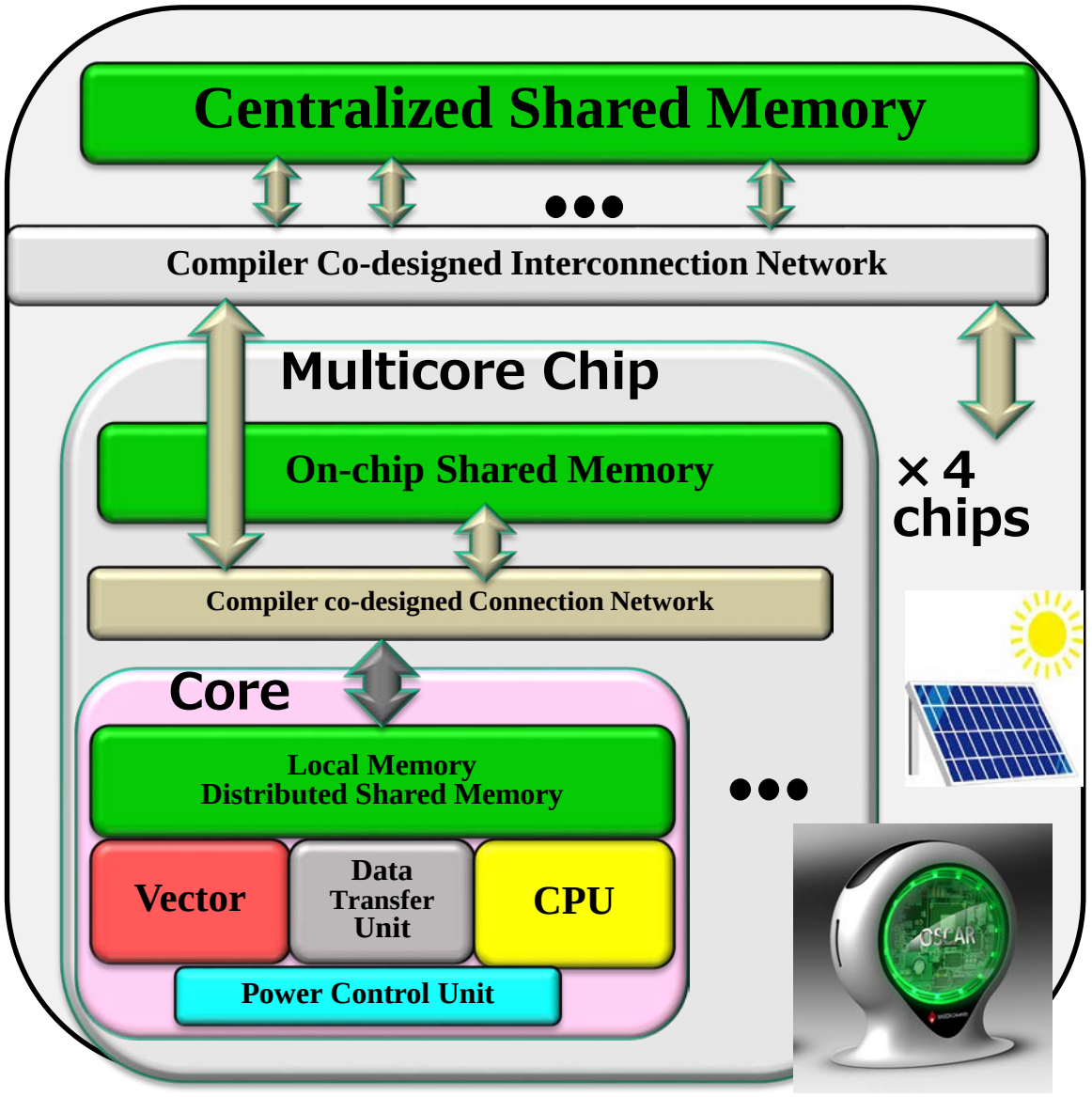
MTG generated by earliest executable condition analysis

Automatic Software Coherent Control for Manycores

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



OSCAR Vector Multicore and Compiler for Embedded to Severs with OSCAR Technology



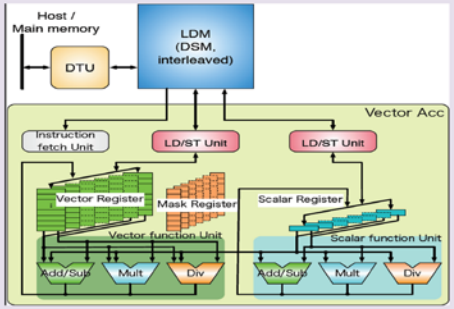
Target:

- **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

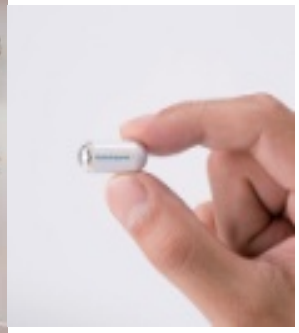
Next Generation Automobiles

- 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

Summary

- To get speedup and power reduction on homogeneous and heterogeneous multicore systems, collaboration of architecture and compiler will be more important.
- Automatic Parallelizing and Power Reducing Compiler has succeeded speedup and/or power reduction of scientific applications including "Earthquake Wave Propagation", medical applications including "Cancer Treatment Using Carbon Ion", and "Drinkable Inner Camera", industry application including "Automobile Engine Control", and "Wireless communication Base Band Processing" on various multicores.
  - For example, the automatic parallelization gave us 110 times speedup for "Earthquake Wave Propagation Simulation" on 128 cores of IBM Power 7 against 1 core, 327 times speedup for "Heavy Particle Radiotherapy Cancer Treatment" on 144cores Hitachi Blade Server using Intel Xeon E7-8890 , 1.95 times for "Automobile Engine Control" on Renesas 2 cores using SH4A or V850, 55 times for "JPEG-XR Encoding for Capsule Inner Cameras" on Tilera 64 cores Tile64 manycore.
- In automatic power reduction, consumed powers for real-time multi-media applications like Human face detection, H.264, mpeg2 and optical flow were reduced to 1/2 or 1/3 using 3 cores of ARM Cortex A9 and Intel Haswell and 1/4 using Renesas SH4A 8 cores against ordinary single core execution.
- For more speedup and power reduction, we have been developing a new architecture/compiler co-designed multicore with vector accelerator based on vector pipelining with vector registers, chaining, load-store pipeline, advanced DMA controller without need of modification of CPU instruction set.