Automatic Multigrain Parallelization, Memory Optimization and Power Reduction Compiler for Multicore Systems

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

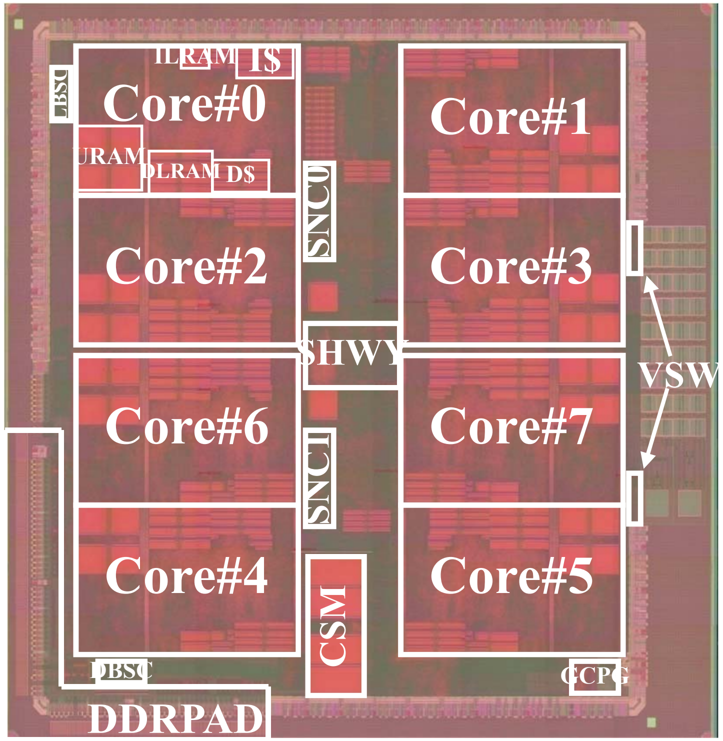
Reviewed Papers: 214, Invited Talks: 161, Published Unexamined Patent Application:59 (Japan, US, GB, China Granted Patents: 43), Articles in News Papers, Web News, Medias incl. TV etc.: 584

Committees in Societies and Government 245

IEEE Computer Society President 2018, BoG(2009-14), Multicore STC Chair (2012-), Japan Chair (2005-07), IPSJ Chair: HG for Mag. & 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"

Power $\propto$ Frequency * Voltage^2

(Voltage $\propto$ Frequency)

➡ **Power $\propto$ 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,

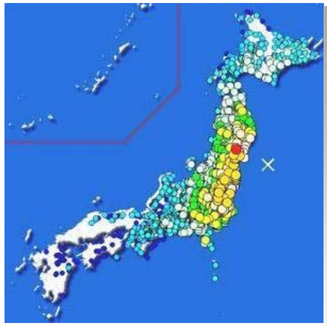
Power is still 1/8 and

Performance becomes 2 times.

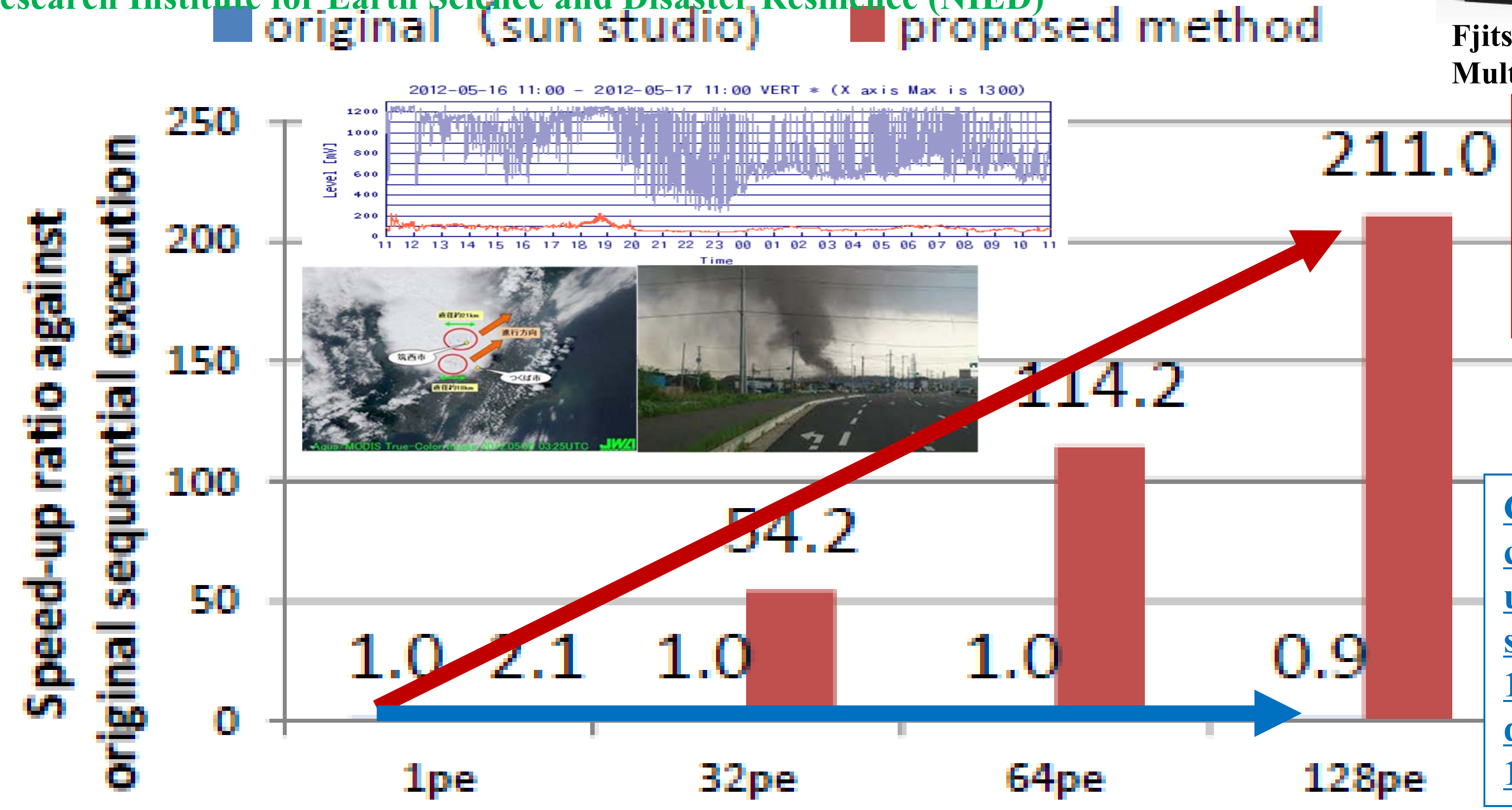
Parallel Soft is important for scalable performance of multicore (LCPC2015)

- 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)



Fjitsu M9000 SPARC Multicore Server

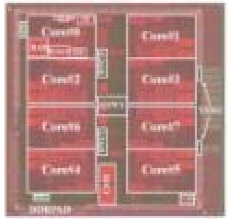


OSCAR Compiler gives us 211 times speedup with 128 cores

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

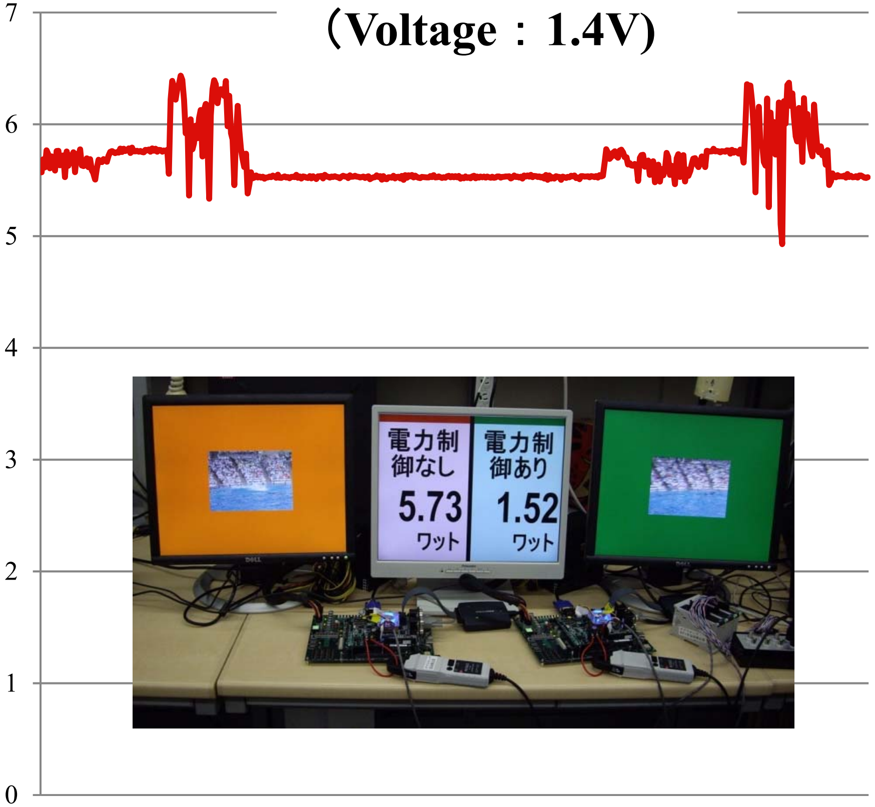
Power Reduction of MPEG2 Decoding to 1/4 on 8 Core Homogeneous Multicore RP-2 by OSCAR Parallelizing Compiler



MPEG2 Decoding with 8 CPU cores

Without Power Control

（Voltage：1.4V)



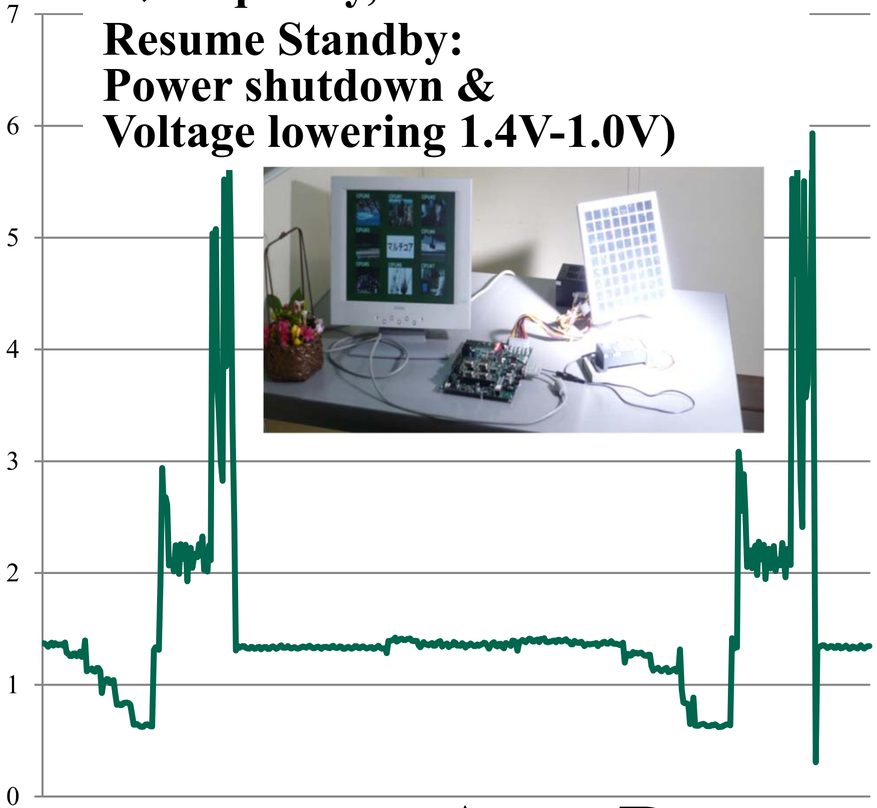
With Power Control

（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]

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)

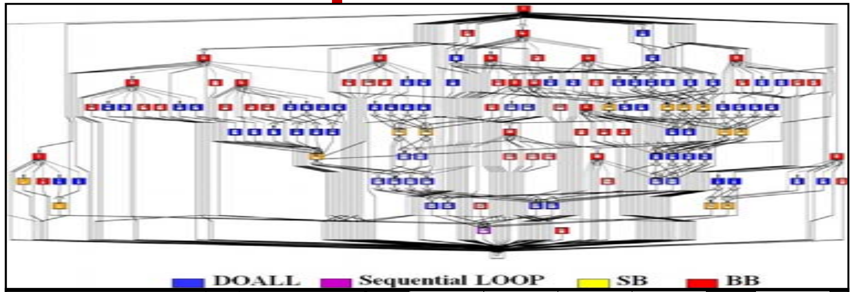
Data Transfer Overlapping(2016 partially)

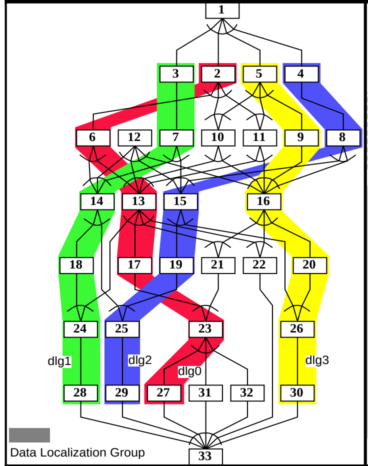
Data transfer overlapping using Data Transfer Controllers (DMAs)

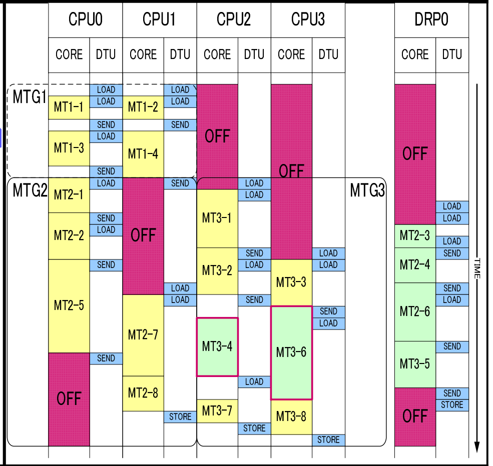
Power Reduction

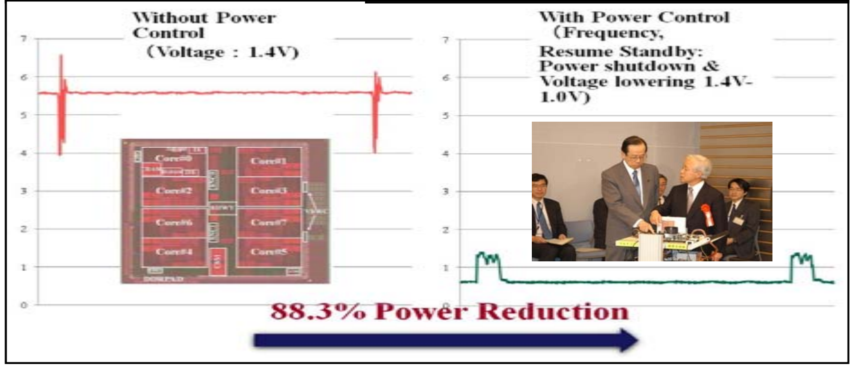
(2005 for Multicore, 2011 Multi-processes, 2013 on ARM)

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





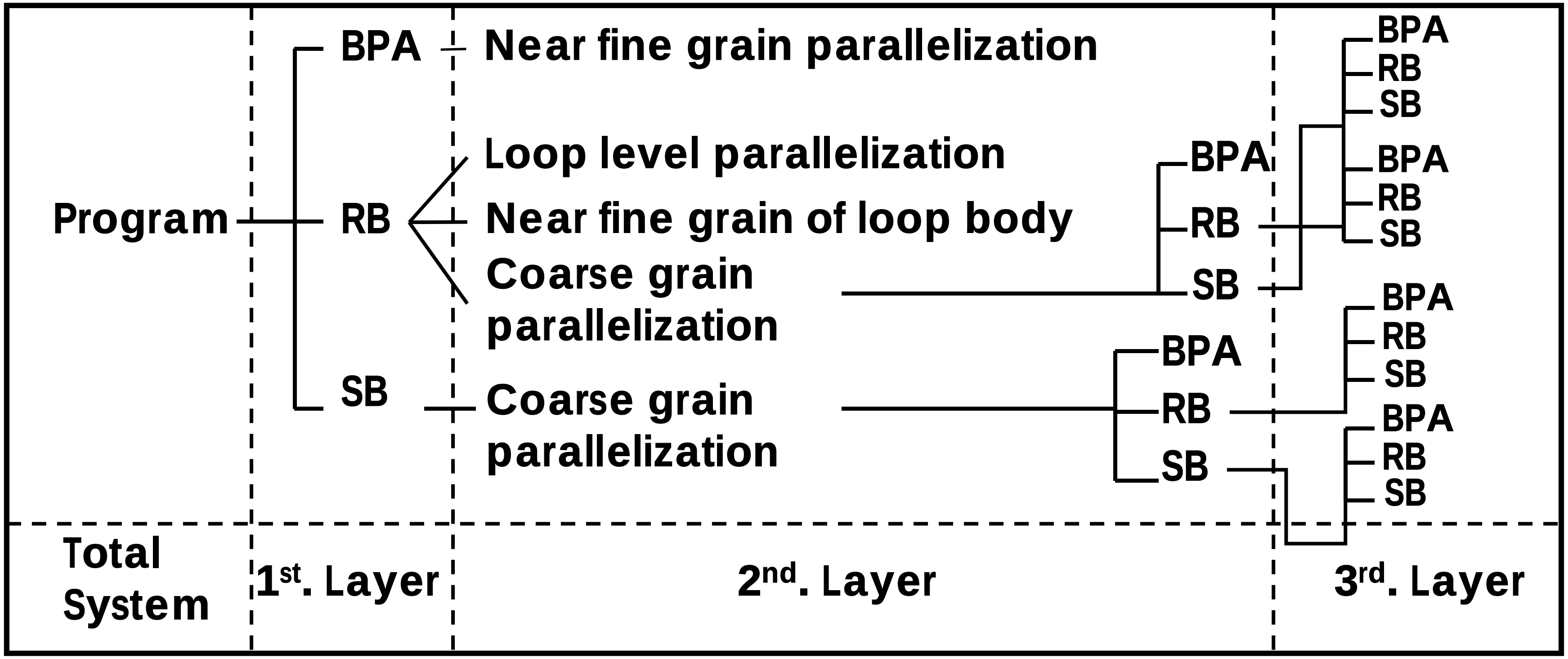




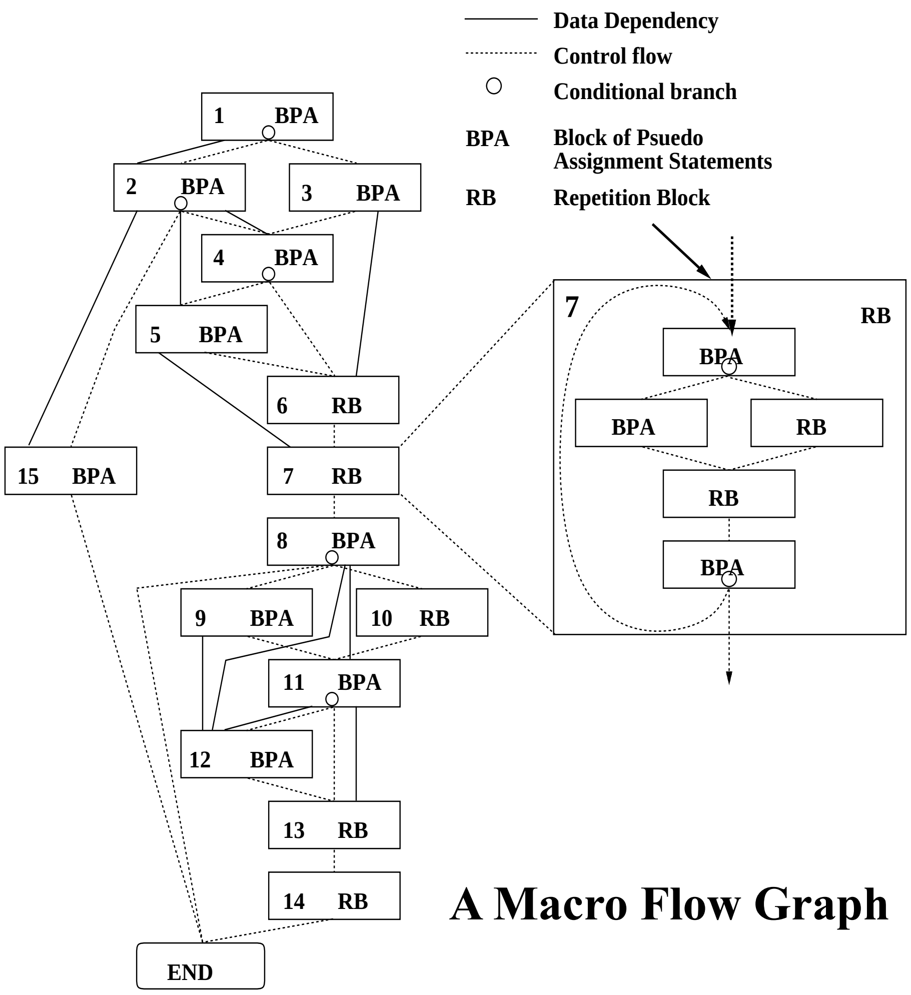
Generation of Coarse Grain Tasks

■Macro-tasks (MTs)

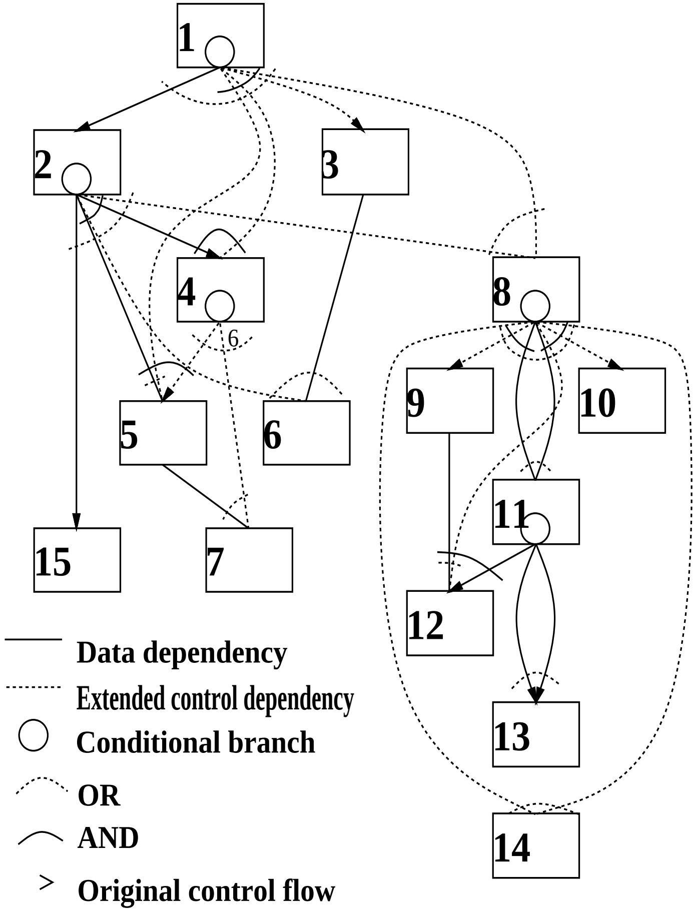
- ➢ Block of Pseudo Assignments (BPA): Basic Block (BB)
- ➢ Repetition Block (RB) : natural loop
- ➢ Subroutine Block (SB): subroutine


Program
BPA
Near fine grain parallelization
RB
Loop level parallelization
Near fine grain of loop body
Coarse grain parallelization
SB
Coarse grain parallelization
BPA
RB
SB
BPA
RB
SB
BPA
RB
SB
BPA
RB
SB
BPA
RB
SB
BPA
RB
SB
BPA
RB
SB
Total System
1st. Layer
2nd. Layer
3rd. Layer

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

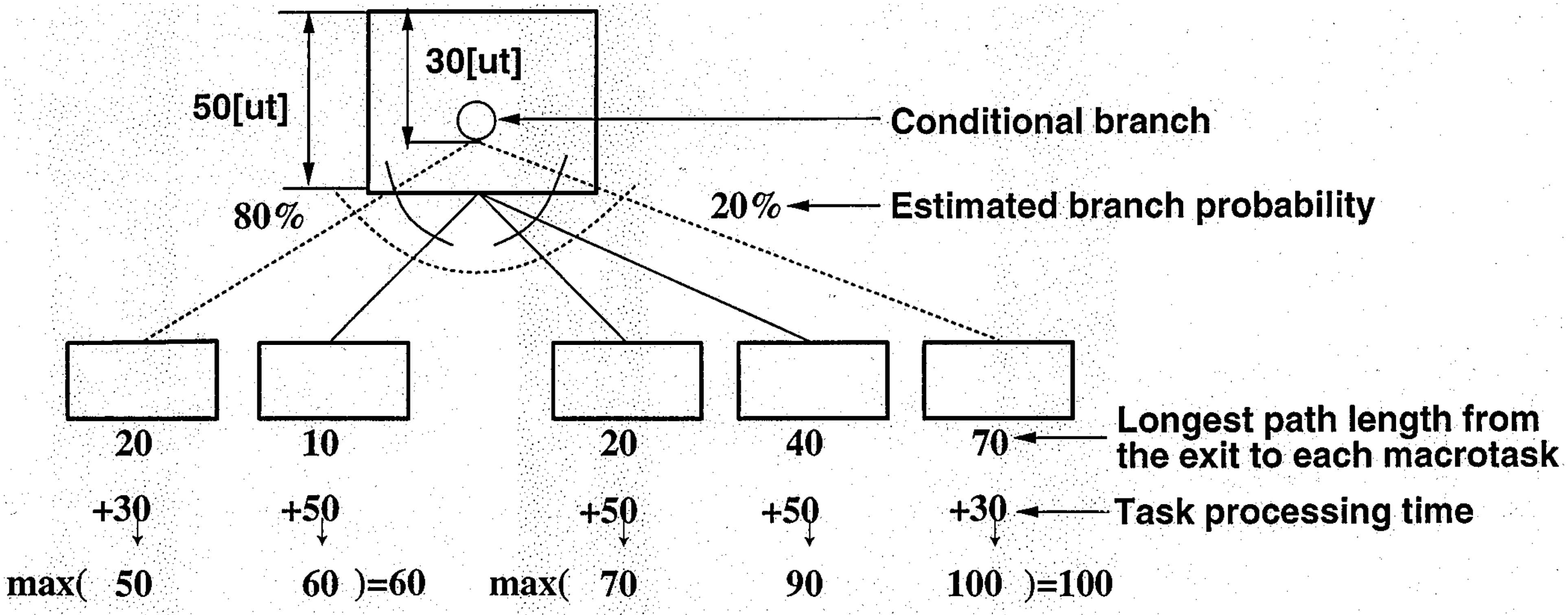


A Macro Flow Graph



A Macro Task Graph

PRIORITY DETERMINATION IN DYNAMIC CP METHOD



Critical path length: 0.80*60+0.20*100= 68

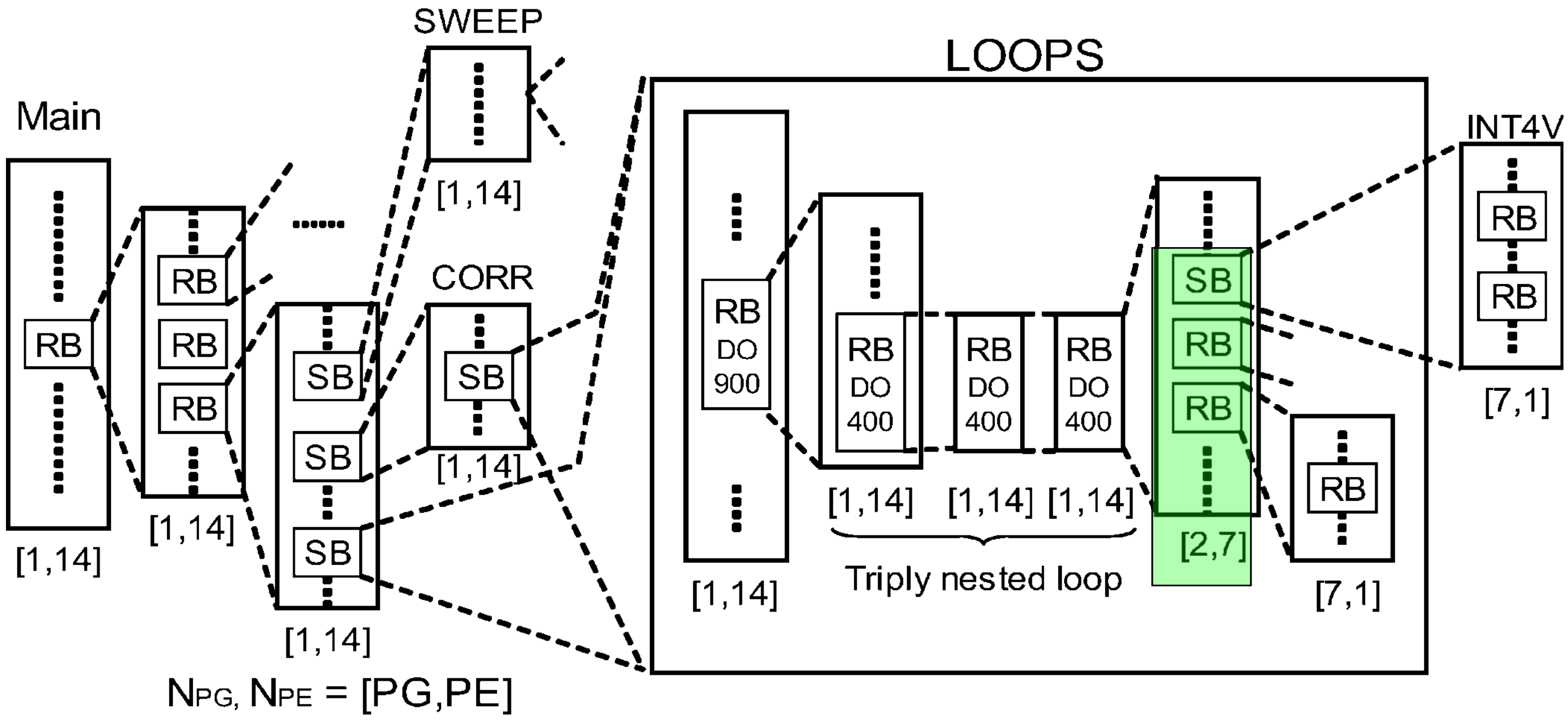
Earliest Executable Conditions

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

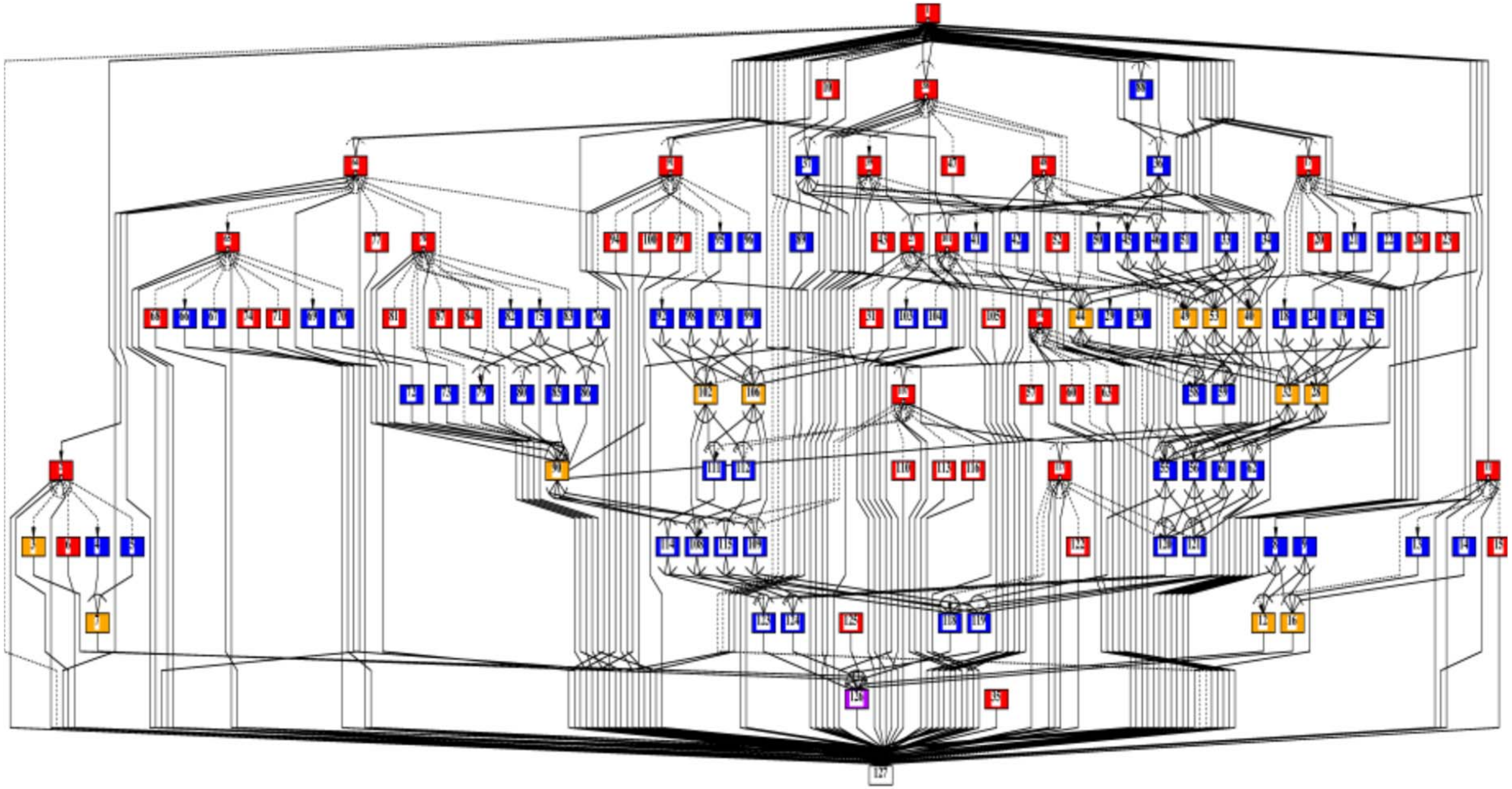
- **Using 14 processors**

Coarse grain parallelization within DO400



MTG of Su2cor-LOOPS-DO400

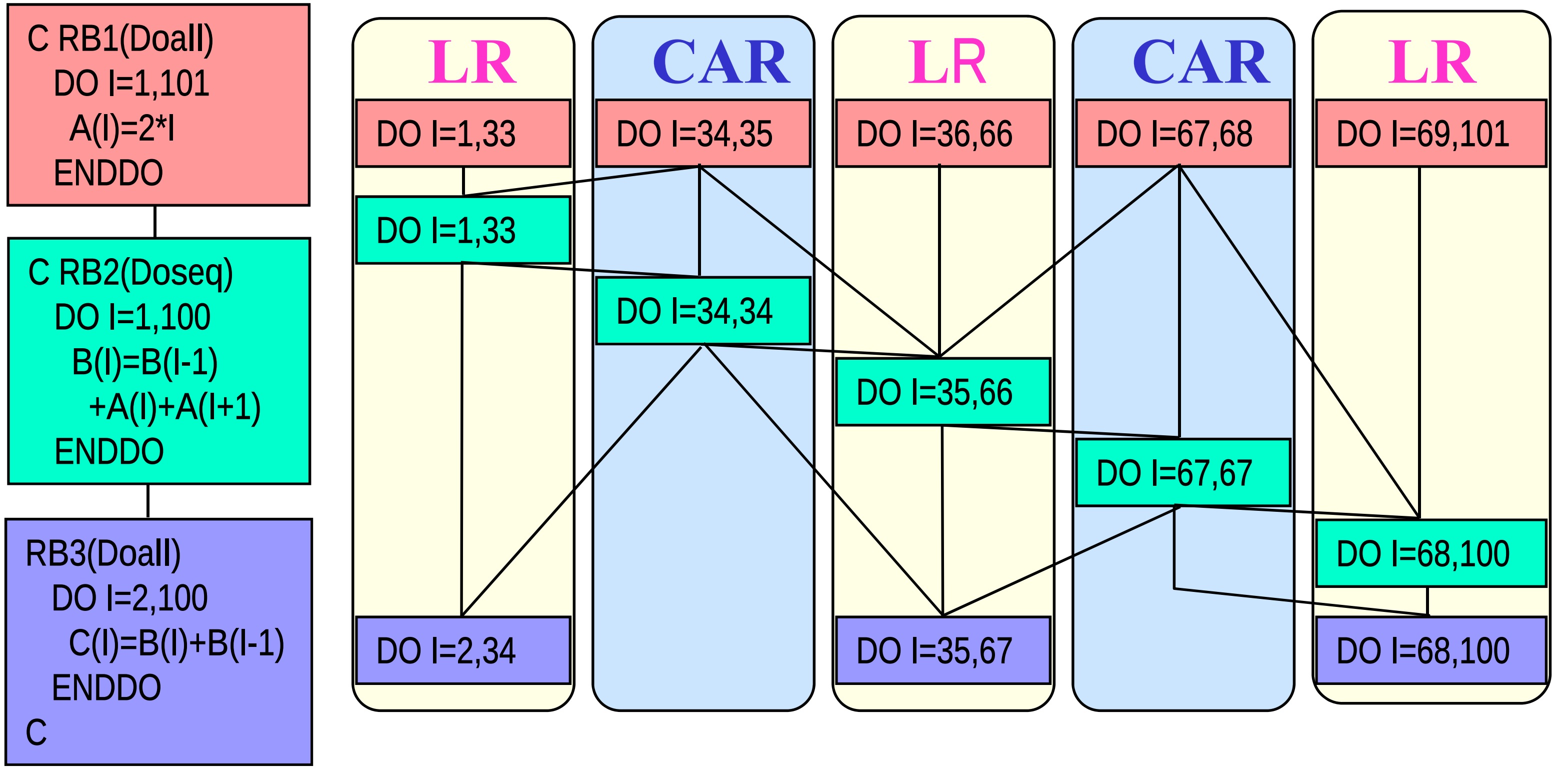
- **Coarse grain parallelism PARA_ALD = 4.3**



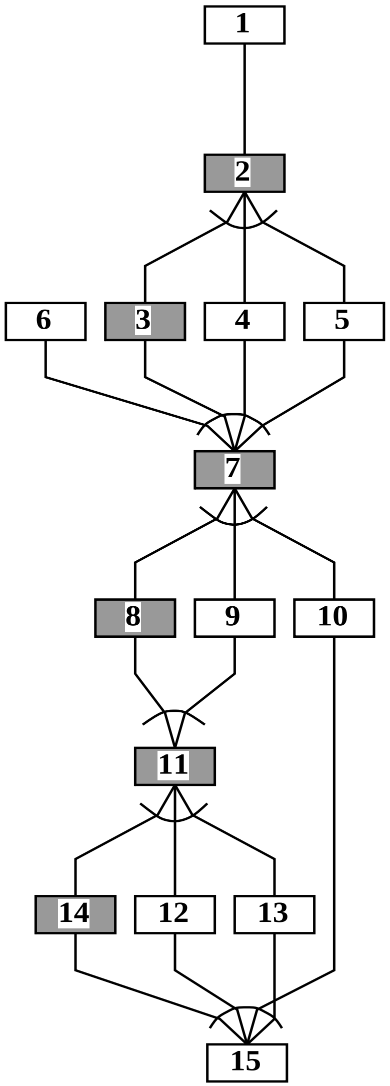
🟦 **DOALL** 🟪 **Sequential LOOP** 🟨 **SB** 🟥 **BB**

Data-Localization: Loop Aligned Decomposition

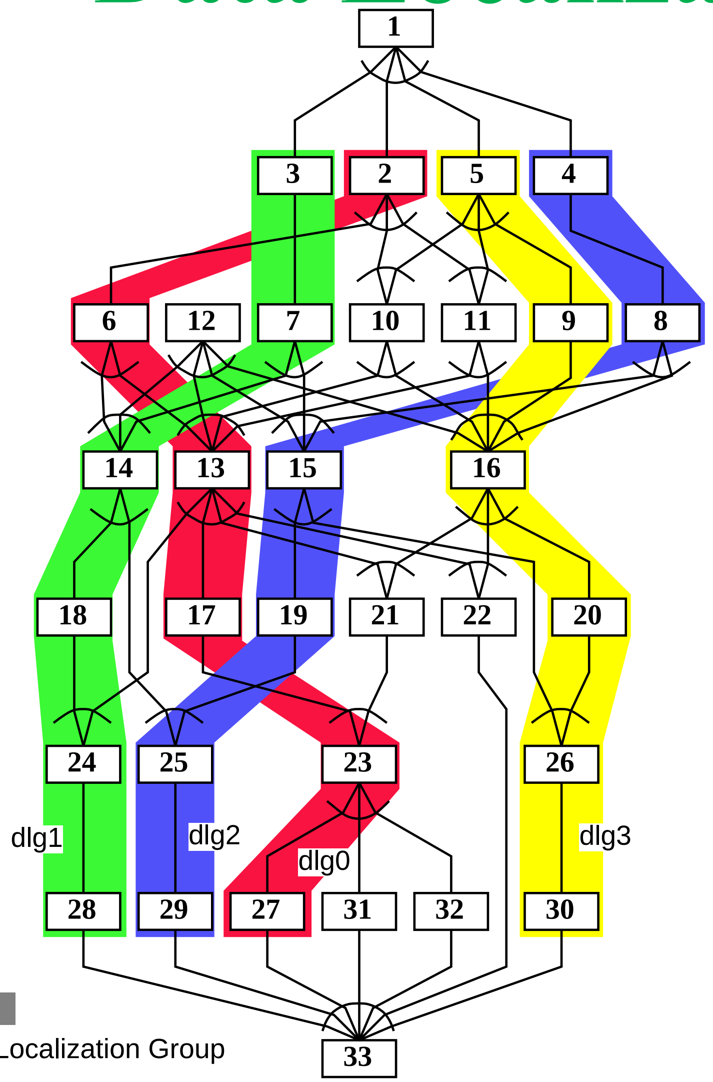
- Decompose multiple loop (Doall and Seq) into CARs and LRs considering inter-loop data dependence.
  - Most data in LR can be passed through LM.
  - LR: Localizable Region, CAR: Commonly Accessed Region



Data Localization



MTG



▨ Data Localization Group

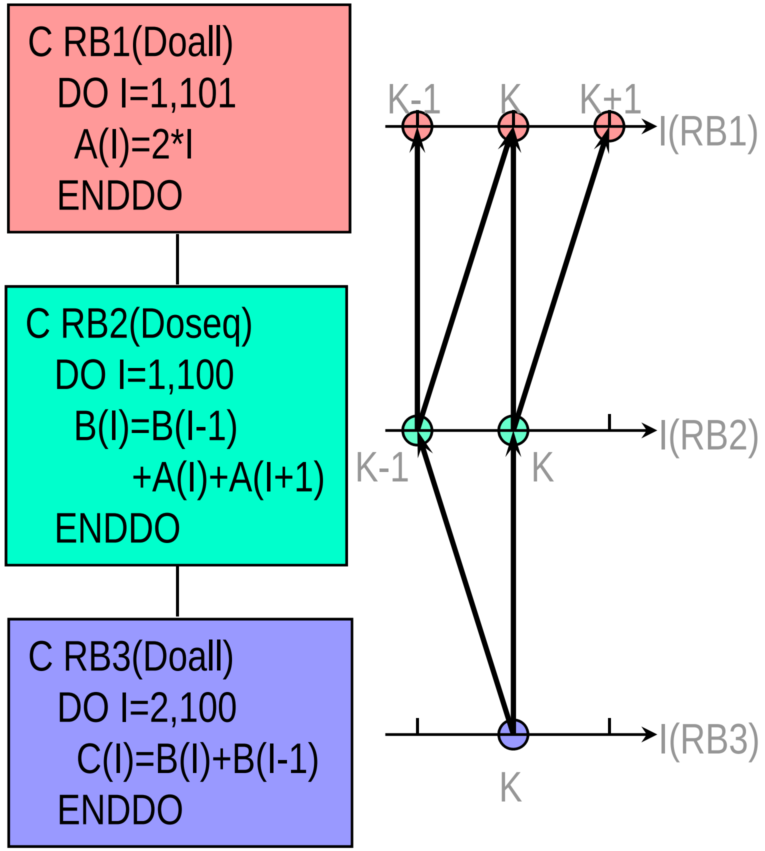
MTG after Division

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

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},$K+1_{th}$ of RB1



```
C RB1(Doall)
  DO I=1,101
   A(I)=2*I
  ENDDO
```

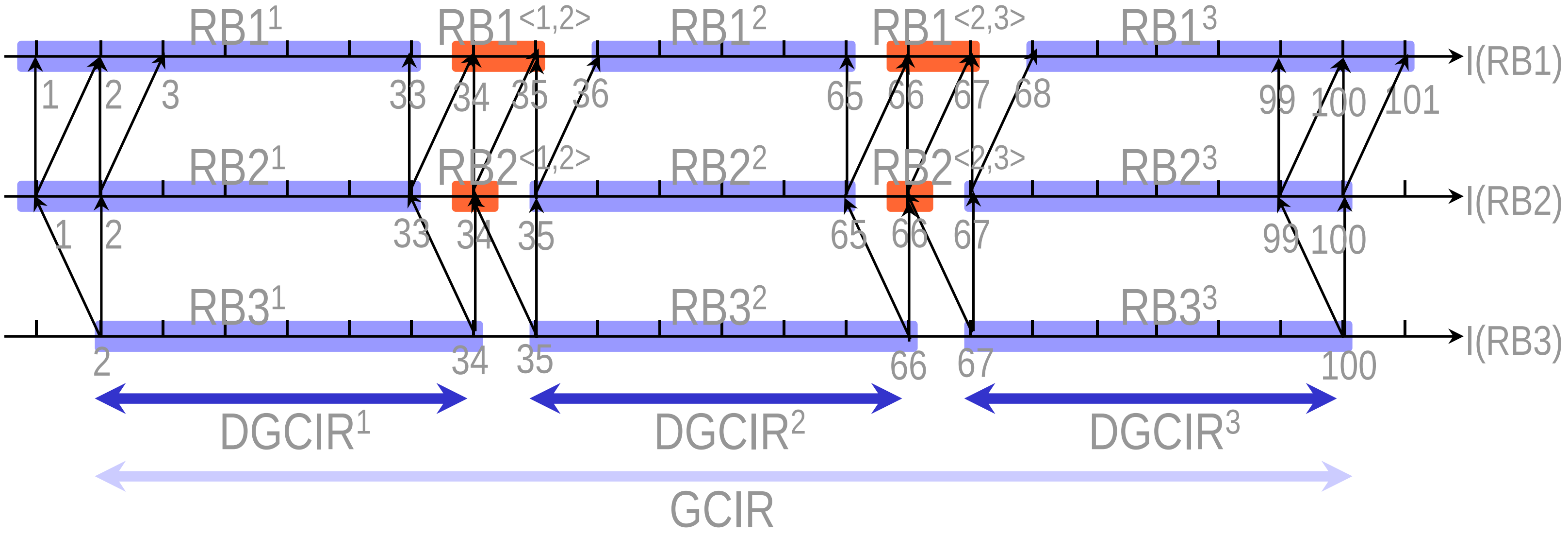
```
C RB2(Doseq)
  DO I=1,100
   B(I)=B(I-1)
       +A(I)+A(I+1)
  ENDDO
```

```
C RB3(Doall)
  DO I=2,100
   C(I)=B(I)+B(I-1)
  ENDDO
```

Example of TLG

Decomposition of RBs in TLG

- Decompose GCIR into DGCIR^p(1≦p≦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.



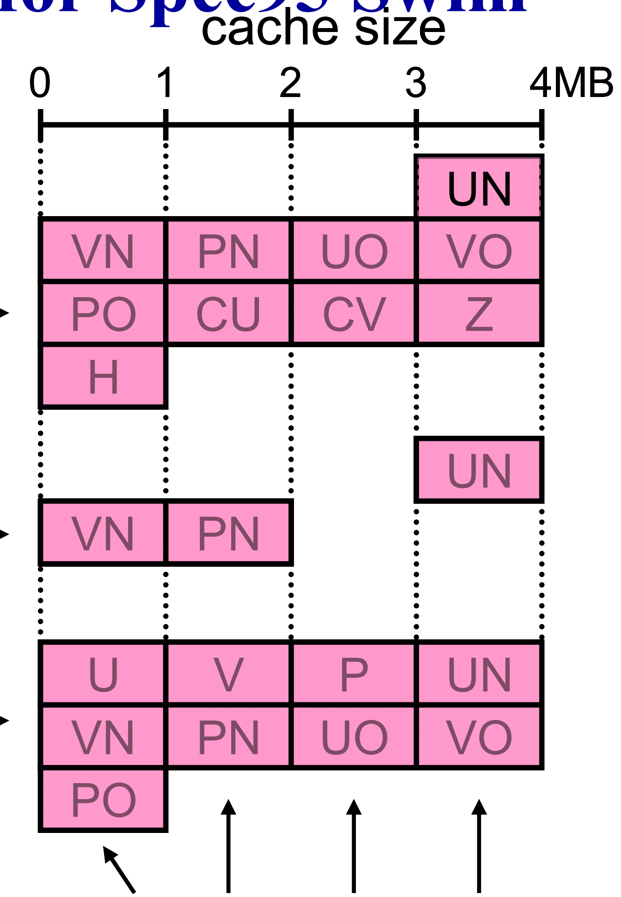
An Example of Data Localization for Spec95 Swim

```
      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



Cache line conflicts occurs among arrays which share the same location on cache

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

Data Layout for Removing Line Conflict Misses by Array Dimension Padding

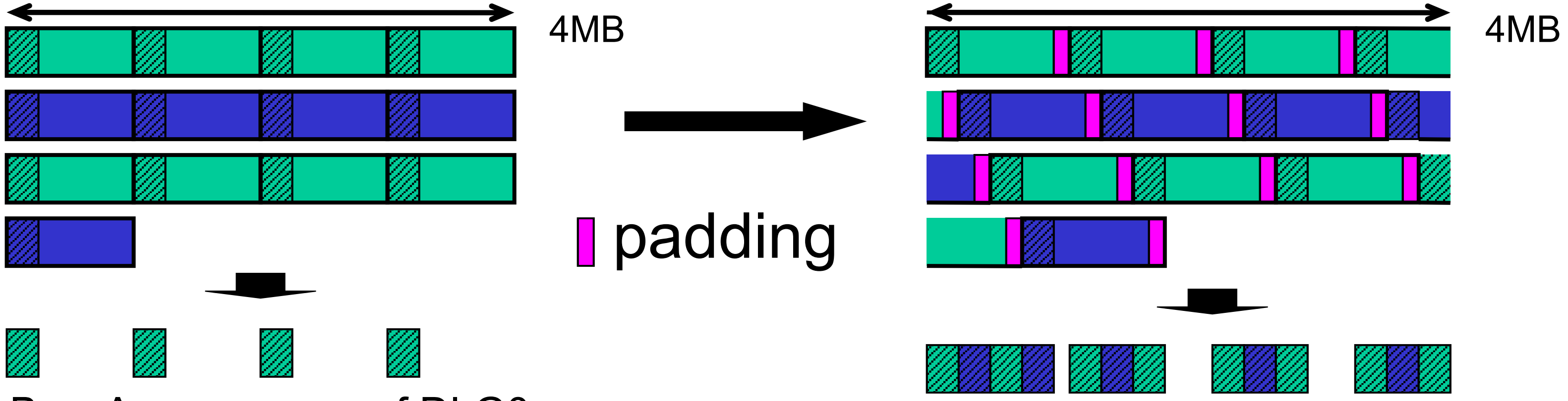
Declaration part of arrays in spec95 swim

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

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

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

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

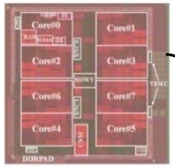
Low Power Heterogeneous Multicore Code Generation

API Analyzer (Available from Waseda)	Existing sequential compiler

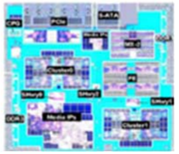
⋮

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

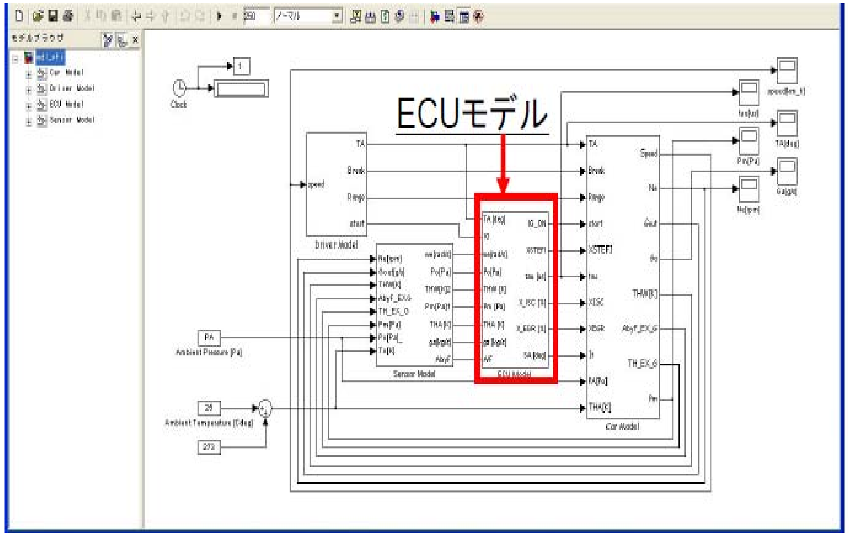
Executable on various multicores

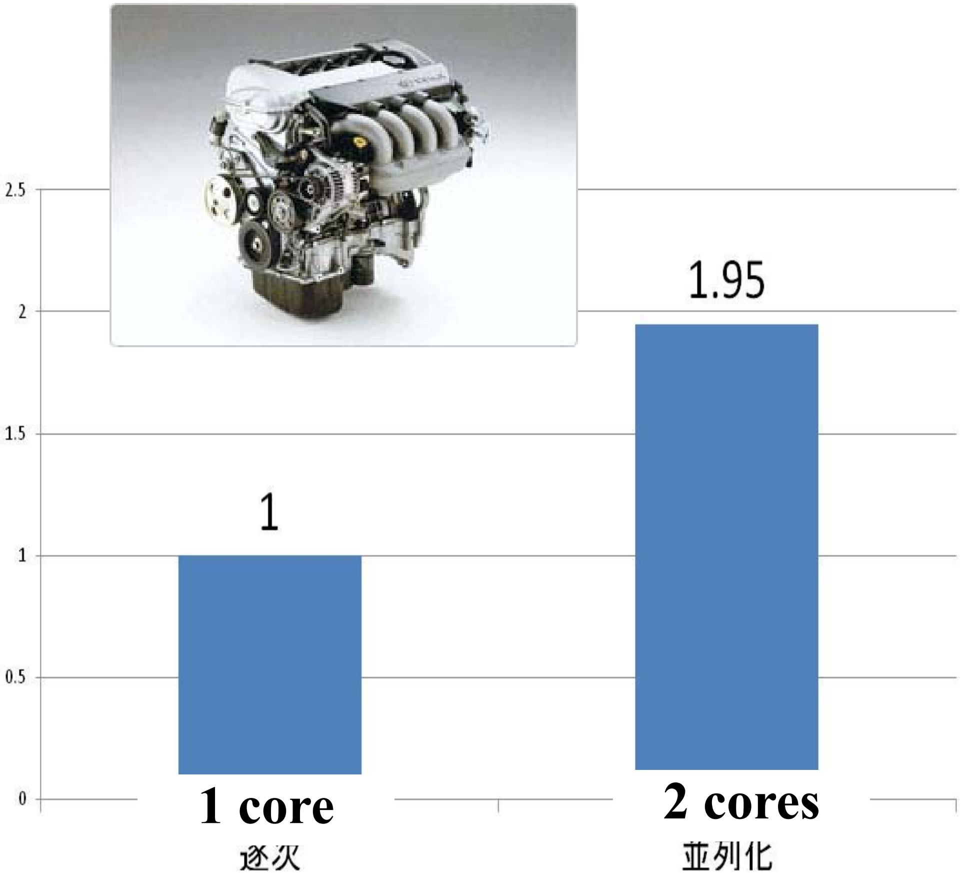
OSCAR: Optimally Scheduled Advanced Multiprocessor
API : Application Program Interface

Engine Control by multicore with Denso

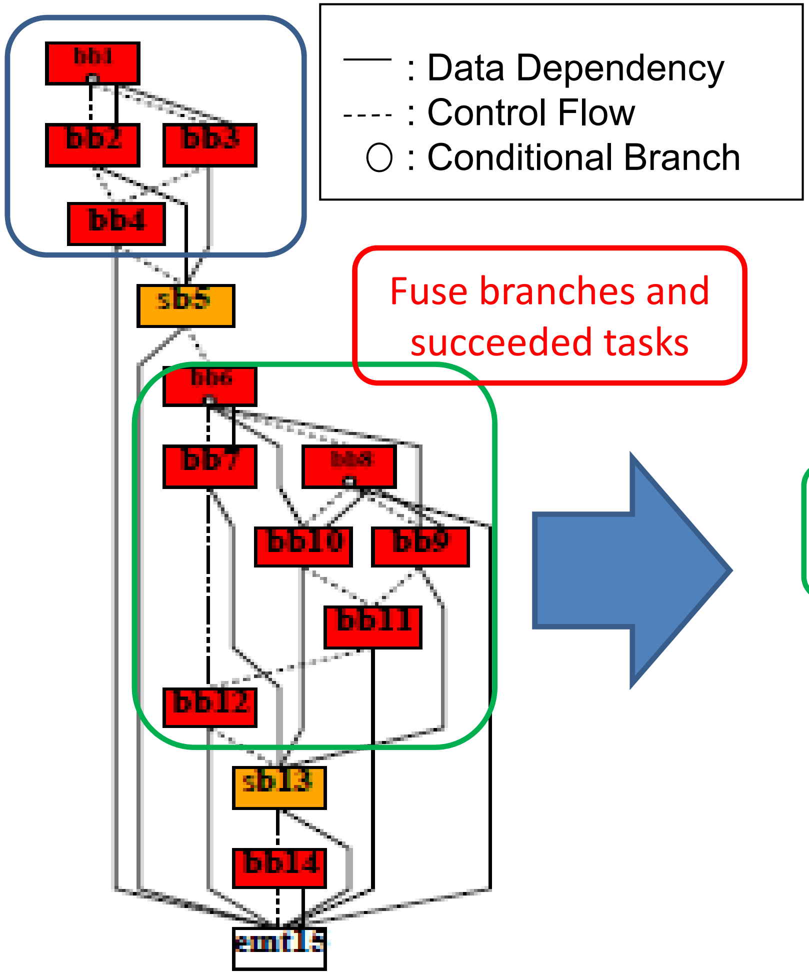
Though so far parallel processing of the engine control on multicore has been very difficult, Denso and Waseda succeeded 1.95 times speedup on 2core V850 multicore processor.

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





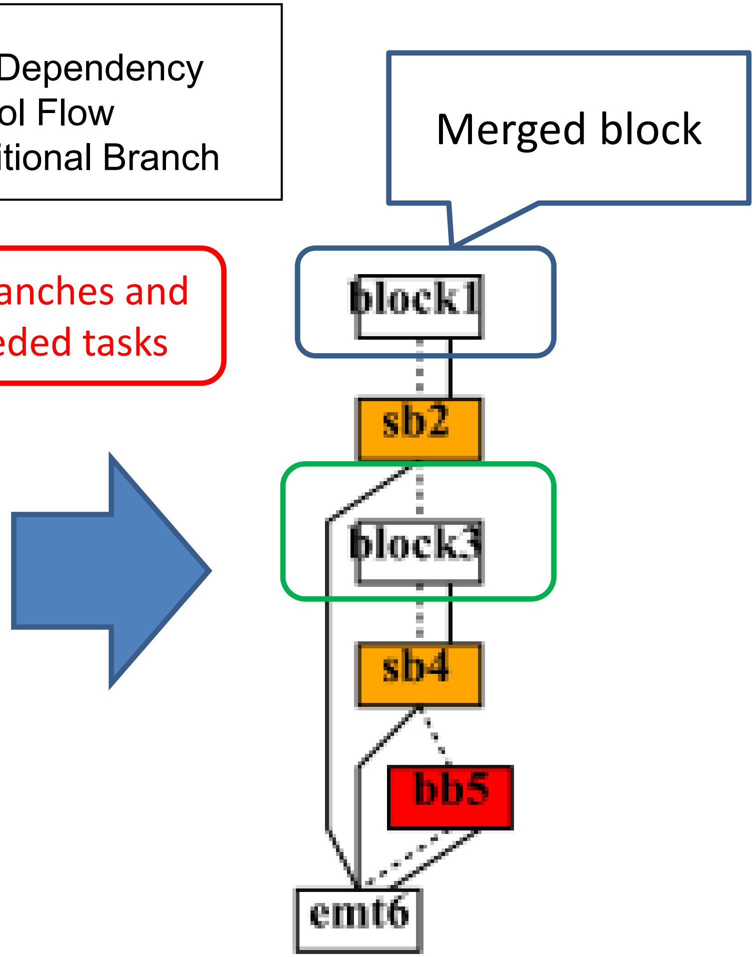
Macro Task Fusion for Static Task Scheduling



—— : Data Dependency
---- : Control Flow
○ : Conditional Branch

Fuse branches and succeeded tasks

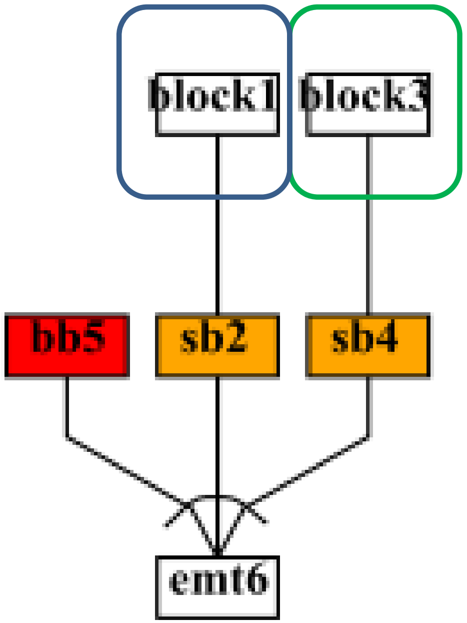
MFG of sample program
before maro task fusion



Merged block

MFG of sample program
after macro task fusion

Only data dependency

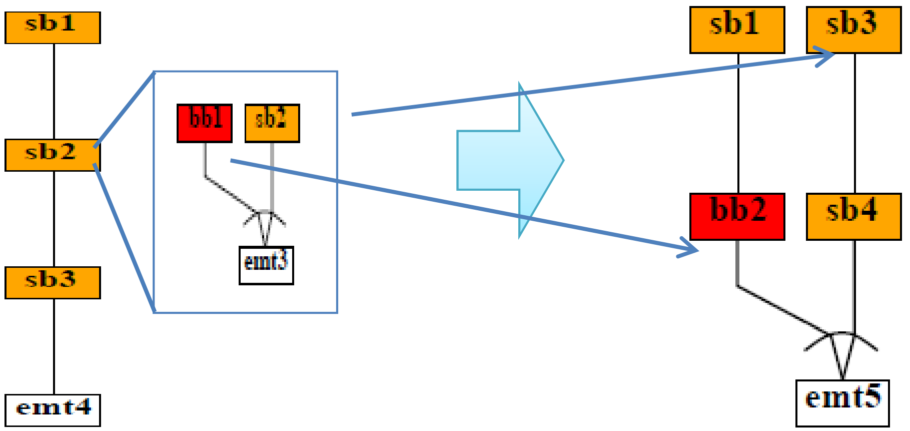


MTG of sample program
after macro task fusion

3.1 Restructuring : Inline Expansion

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

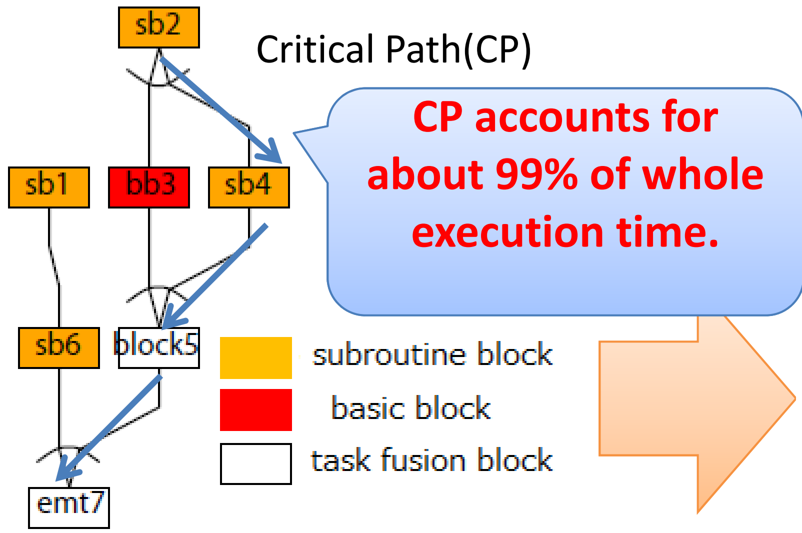
Improves coarse grain parallelism



MTG before inline expansion

MTG after inline expansion

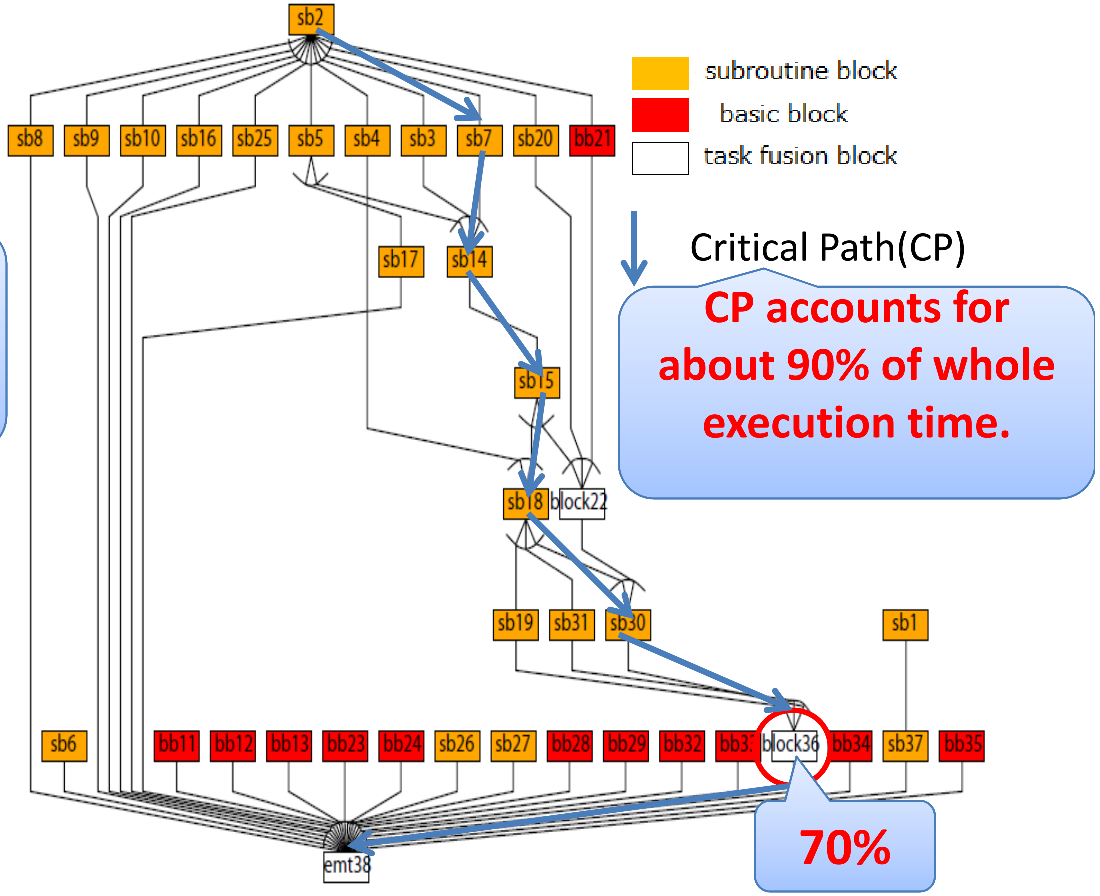
MTG of Crankshaft Program Using Inline Expansion



Critical Path(CP)

CP accounts for about 99% of whole execution time.

MTG of crankshaft program before restructuring



Critical Path(CP)

CP accounts for about 90% of whole execution time.

70%

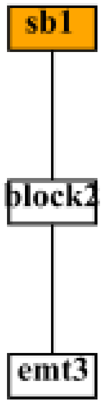
Not enough coarse grain parallelism yet!

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

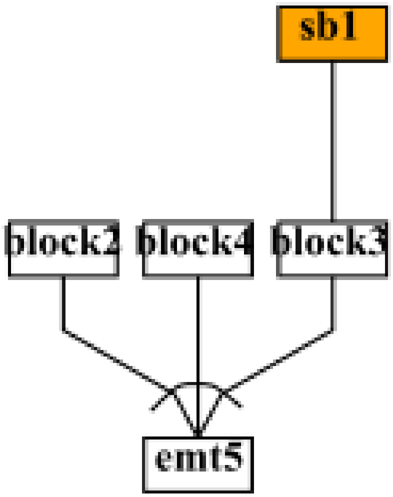


```
func1();
if (condition) {
    func2();
    func3();
    func4();
}
```

No dependemce

Copying if-condition for each function

```
func1();
if (condition) {
    func2();
}
if (condition) {
    func3();
}
if (condition) {
    func4();
}
```

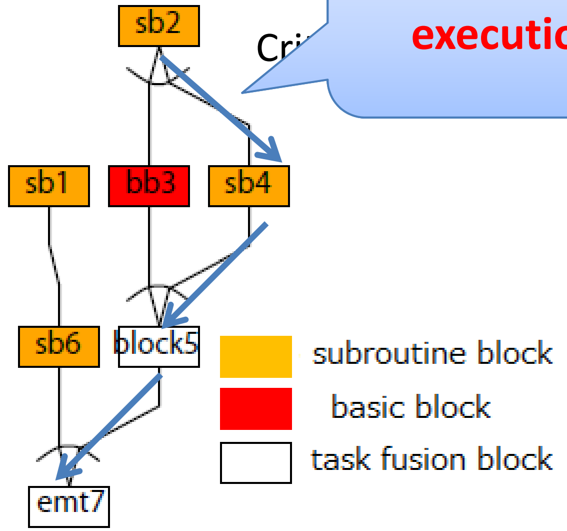


before duplicating if-statements

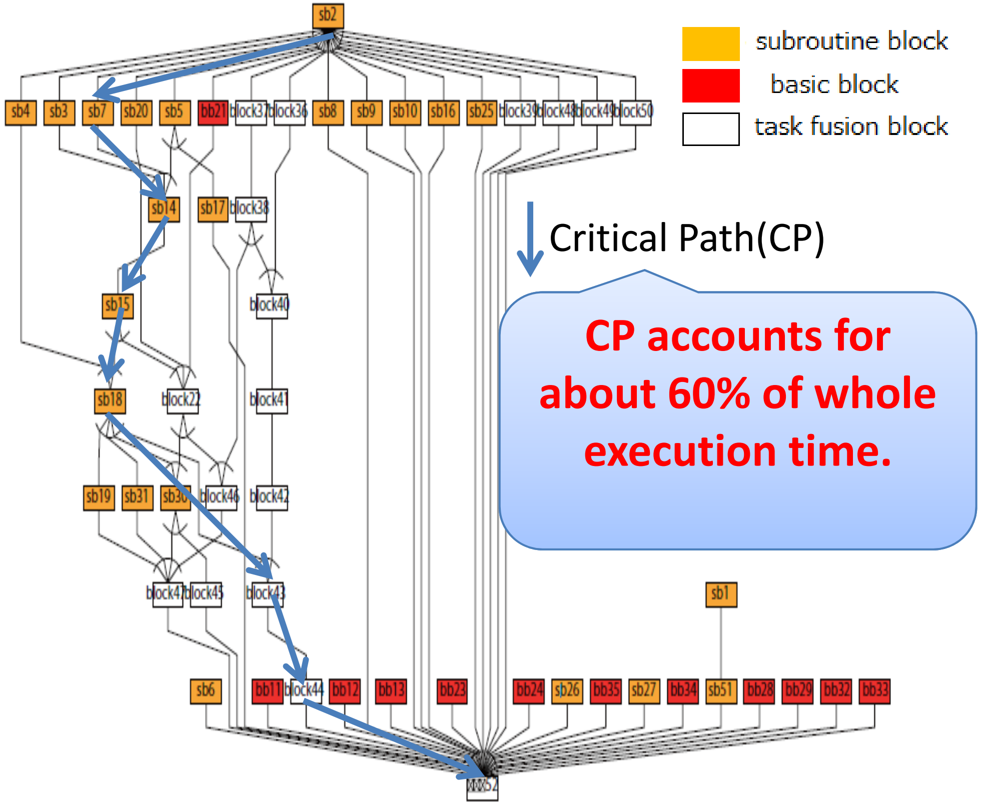
after duplicating if-statements

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

CP accounts for over 99% of whole execution time.



MTG of crankshaft program before restructuring



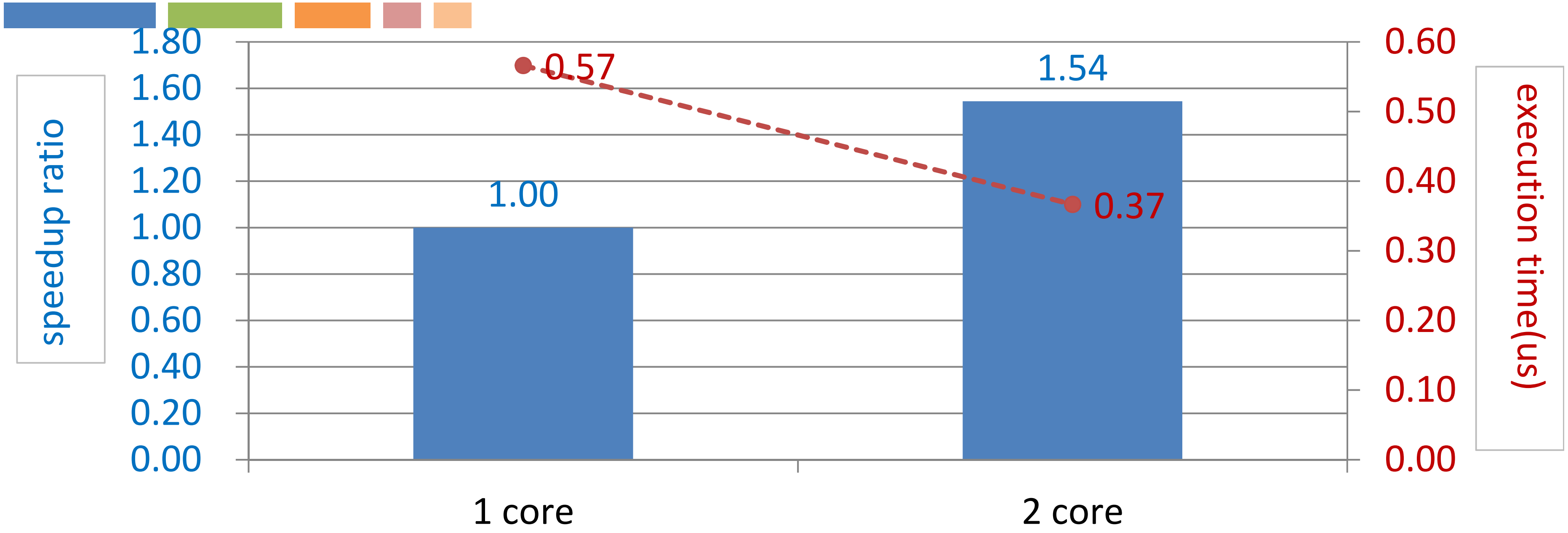
CP accounts for about 60% of whole execution time.

MTG of crankshaft program after restructuring

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

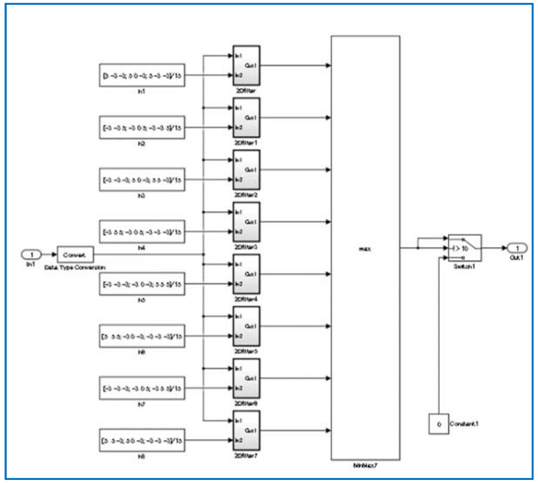
Successfully increased coarse grain parallelism

Evaluation of Crankshaft Program with Multi-core Processors



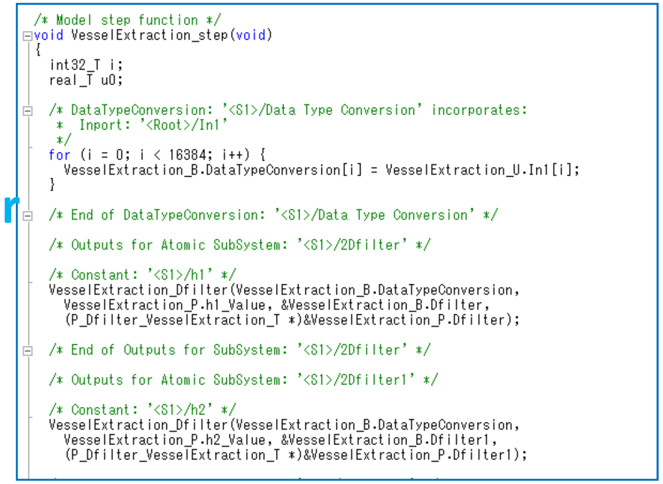
- Attain 1.54 times speedup on RPX
  - There are no loops, but only many conditional branches and small basic blocks and difficult to parallelize this program
- This result shows possibility of multi-core processor for engine control programs

OSCAR Compile Flow for Simulink Applications



Simulink model

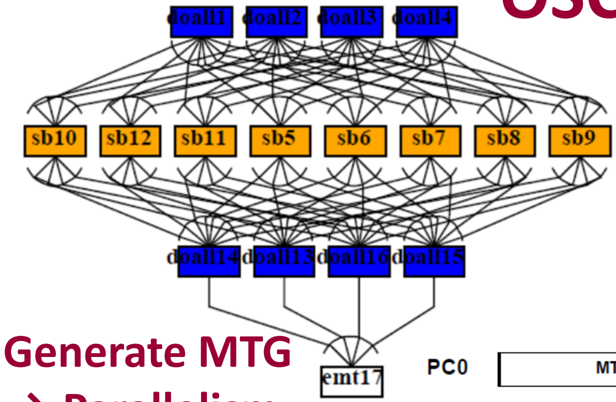
Generate C code using Embedded Coder



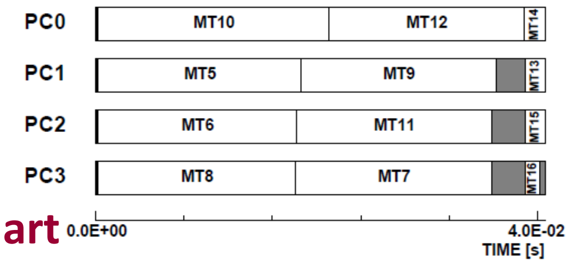
```
/* Model step function */
void VesselExtraction_step(void)
{
  int32_T i;
  real_T u0;

  /* 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

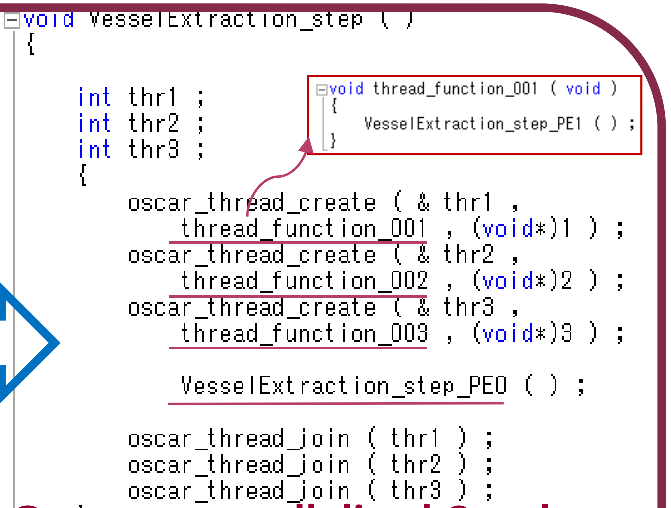
OSCAR Compiler



(1) Generate MTG
→ Parallelism



(2) Generate gantt chart
→ Scheduling in a multicore



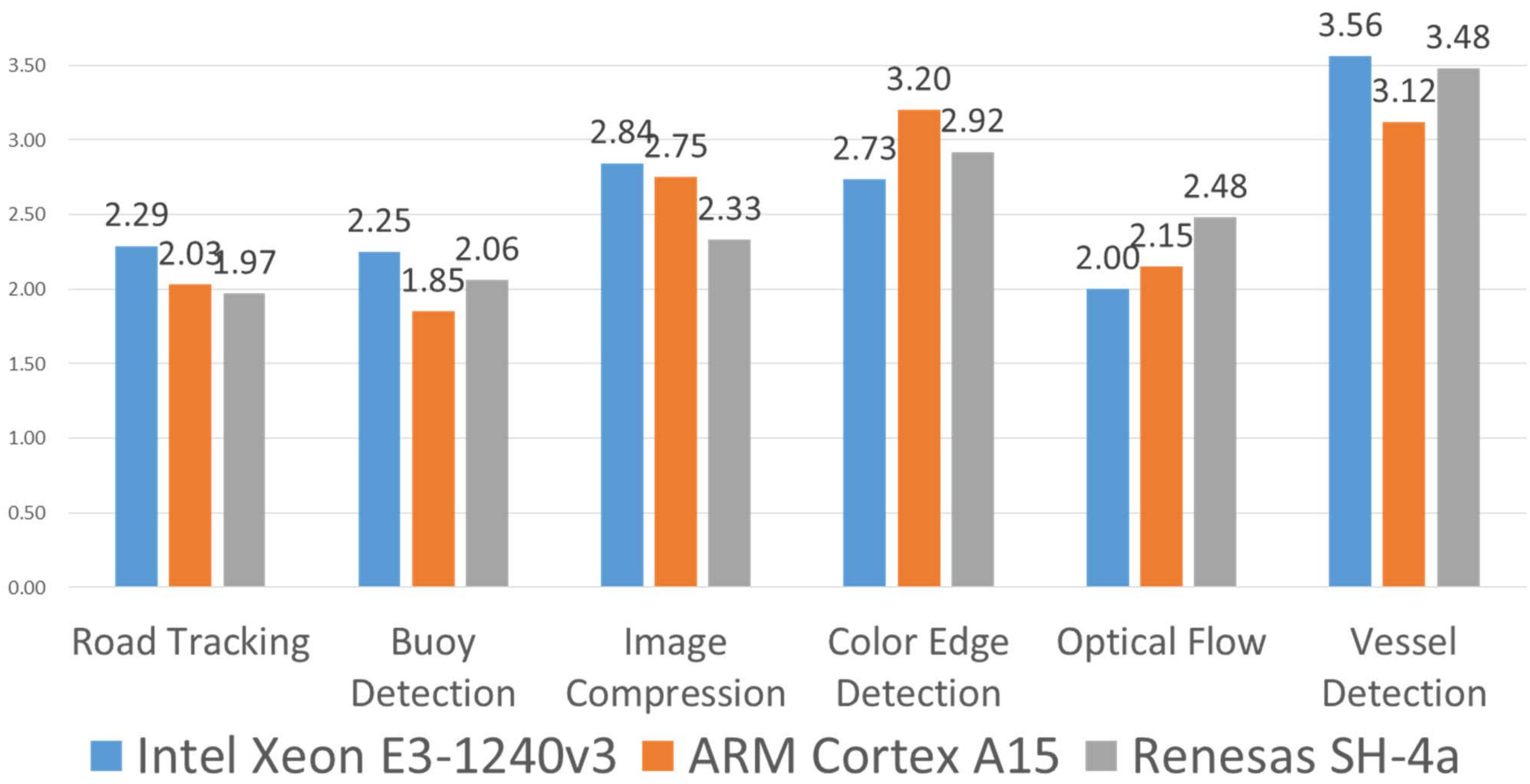
```
void VesselExtraction_step ( )
{
    int thr1 ;
    int thr2 ;
    int thr3 ;
    {
        oscar_thread_create ( & thr1 ,
            thread_function_001 , (void*)1 ) ;
        oscar_thread_create ( & thr2 ,
            thread_function_002 , (void*)2 ) ;
        oscar_thread_create ( & thr3 ,
 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

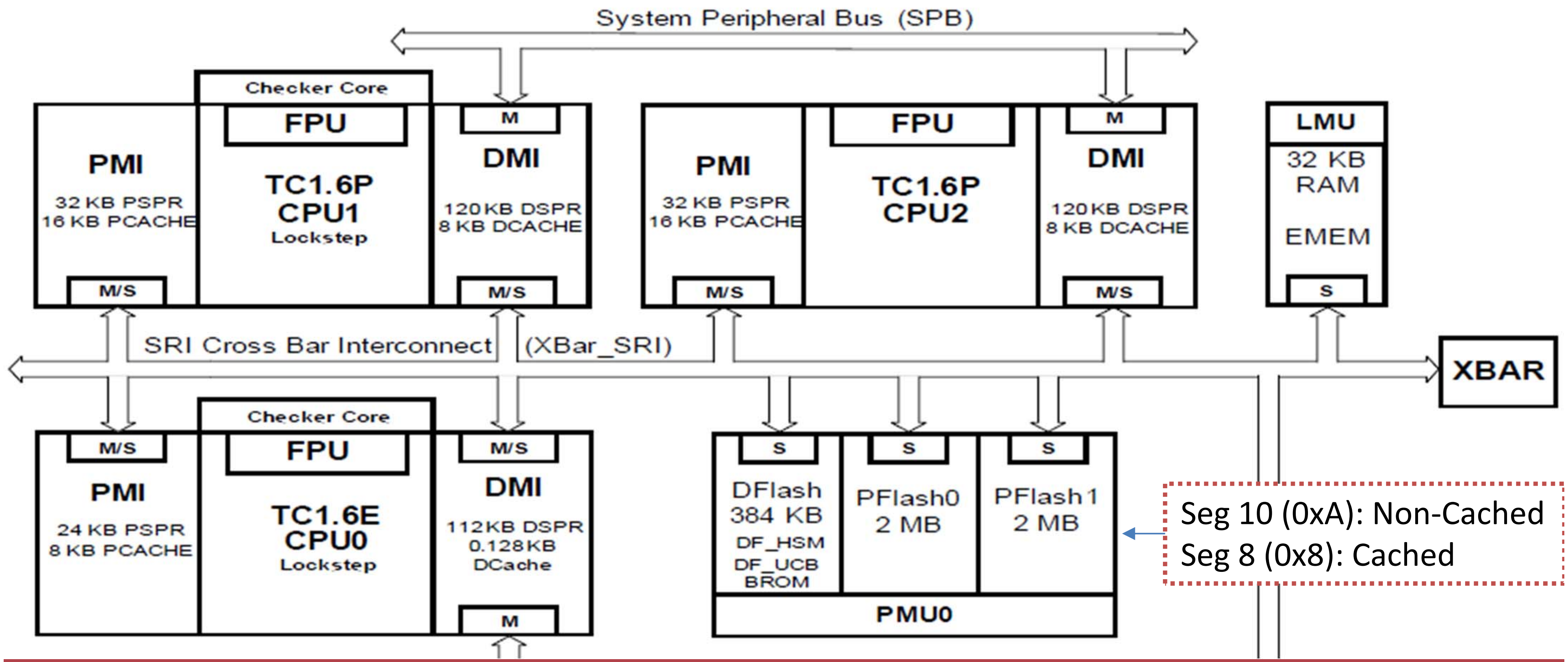
Color Edge Detection : http://www.mathworks.co.jp/matlabcentral/fileexchange/28114-fast-edges-of-a-color-image--actual-color--not-converting-to-grayscale-/

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

Infineon AURIX TC277

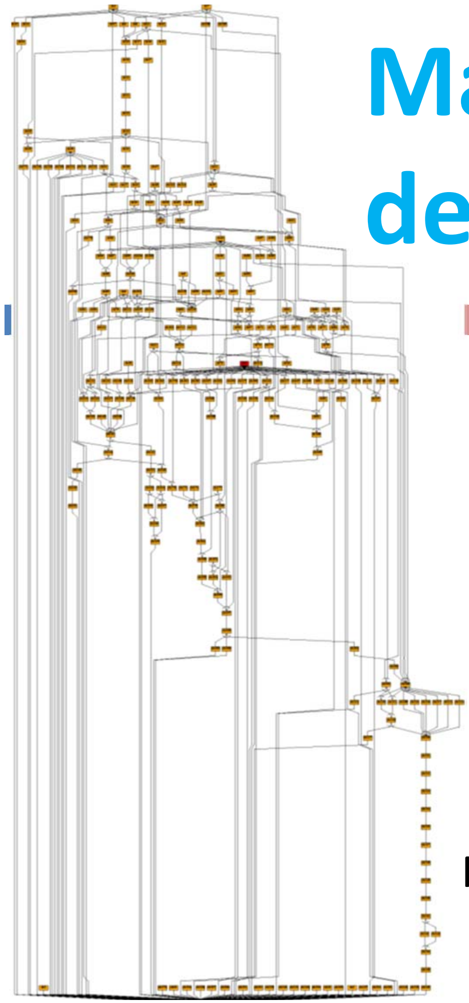
Abbreviations :

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 (EEEPROM)
S :	SRI Slave Interface
M :	SRI Master Interface

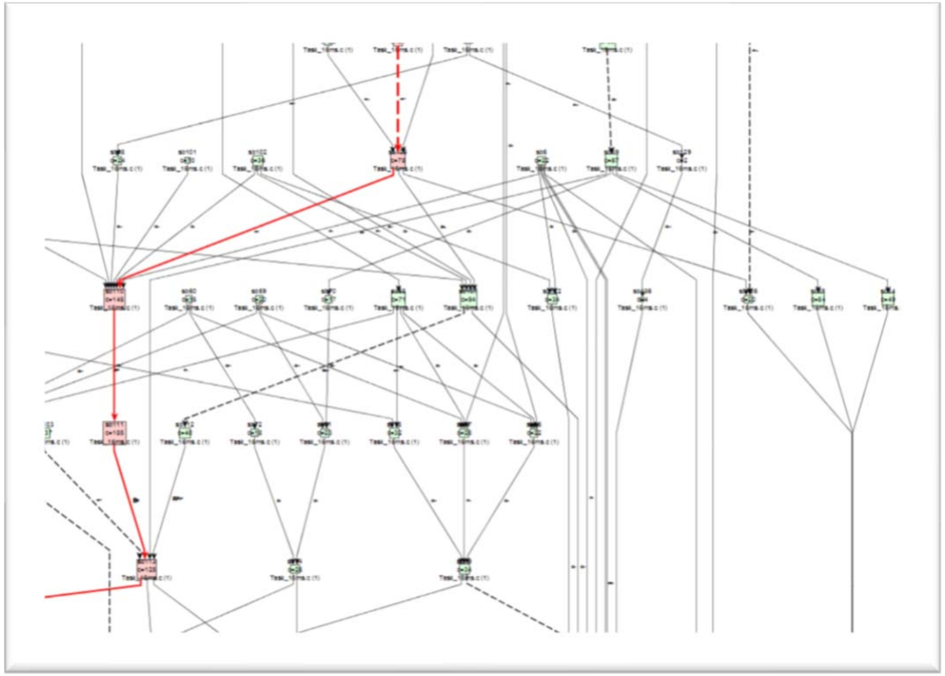


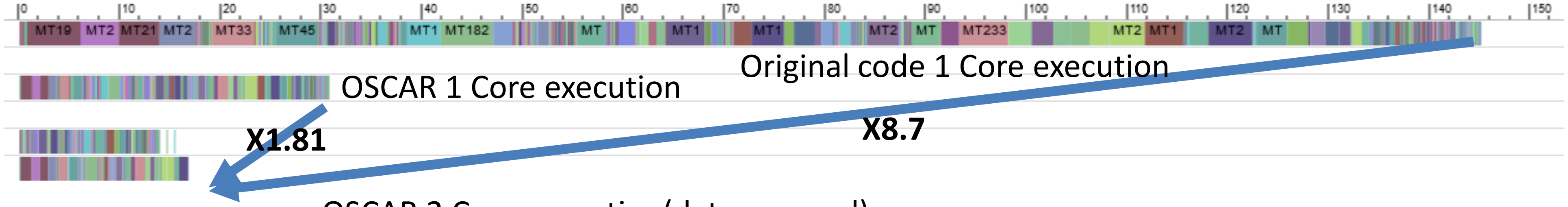
Seg 10 (0xA): Non-Cached
Seg 8 (0x8): Cached

Macrotask Graph, Dependence details and schedules



MTG – task_16ms





Original code 1 Core execution

OSCAR 1 Core execution

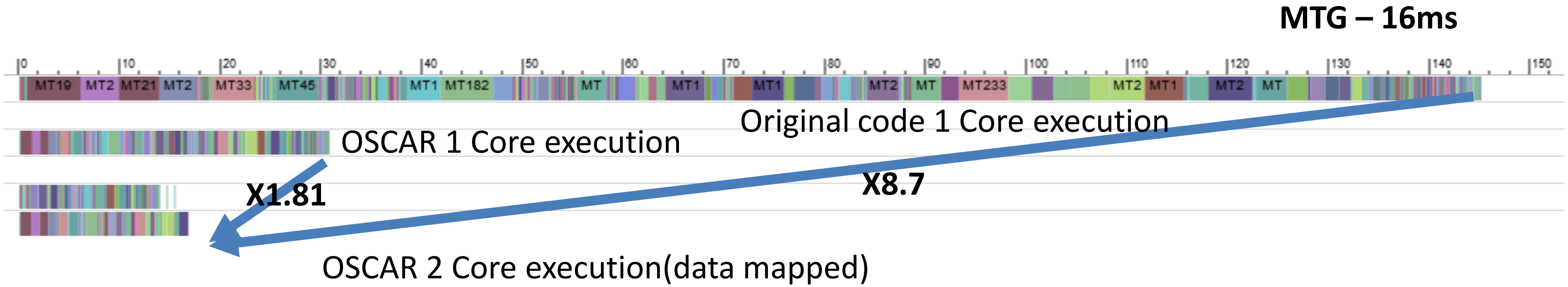
X1.81

X8.7

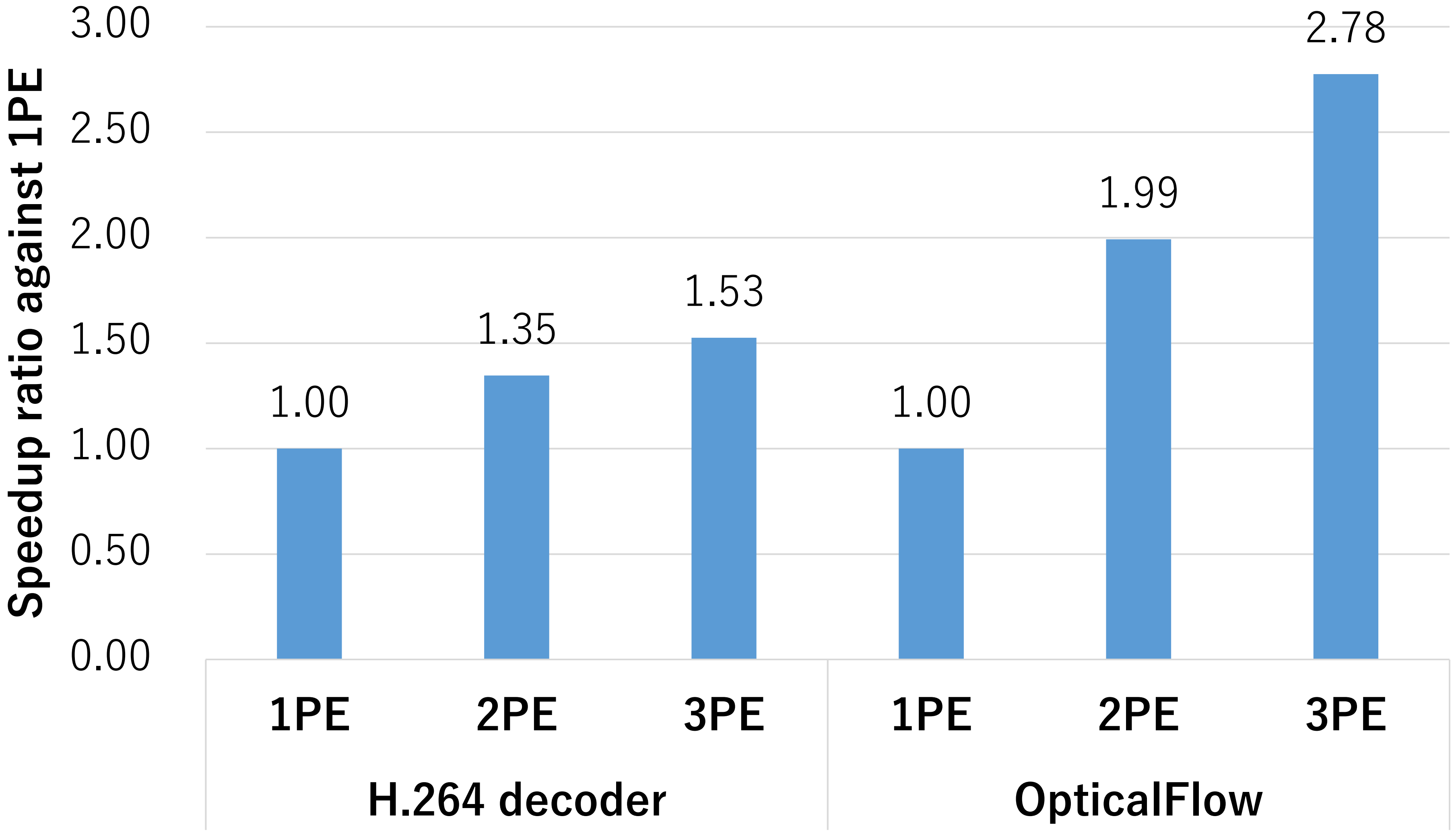
OSCAR 2 Core execution(data mapped)

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

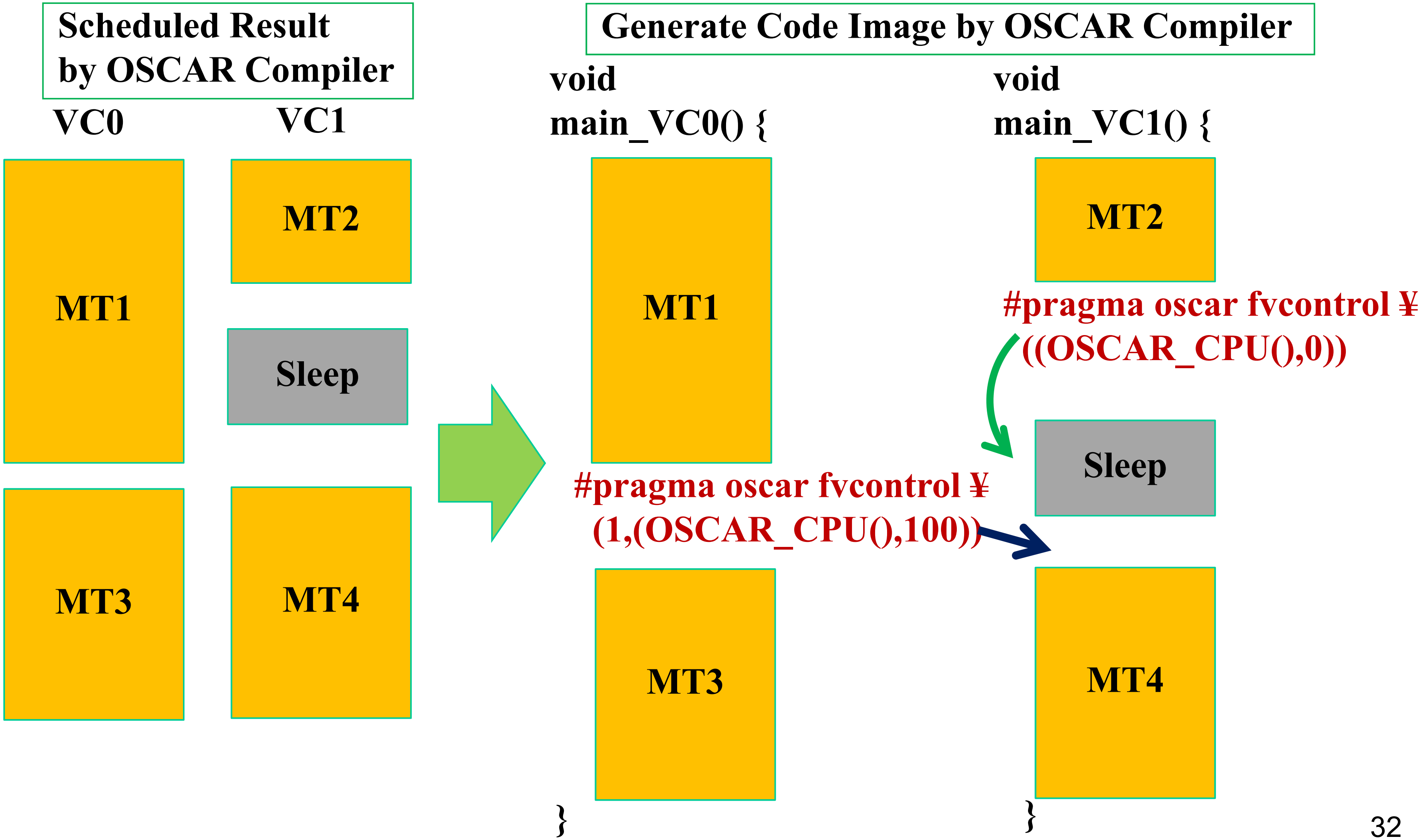
- **Original sequential** execution time on 1 core: **145500** cycles
- **Sequential execution time by OSCAR** on 1 core:**29700** cycles
  - 4.9 times speedup on 1 core against original execution by OSCAR Compilers automatic data allocation for local scratch pad memory, flush memory modules
- **2 core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



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



Low-Power Optimization with OSCAR API



Scheduled Result by OSCAR Compiler

VC0 VC1

MT1

MT2

Sleep

MT3

MT4

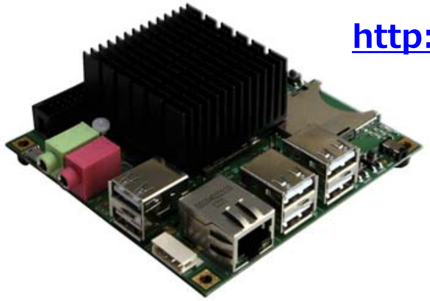
Generate Code Image by OSCAR Compiler

```
void
main_VC0() {
MT1
#pragma oscar fvcontrol ¥
(1,(OSCAR_CPU(),100))
MT3
}
```

```
void
main_VC1() {
MT2
#pragma oscar fvcontrol ¥
((OSCAR_CPU(),0))
Sleep
MT4
}
```

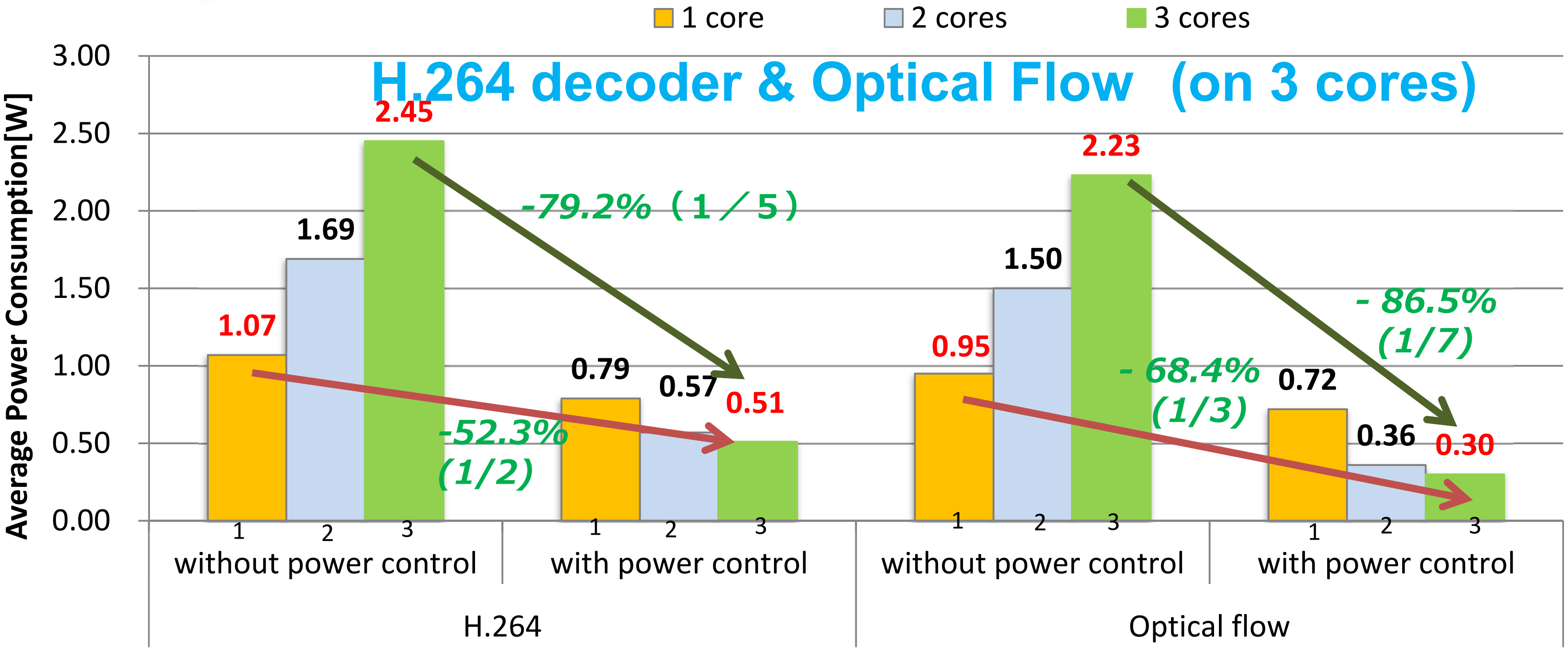
Automatic Power Reduction on ARM CortexA9 with Android

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



ODROID X2

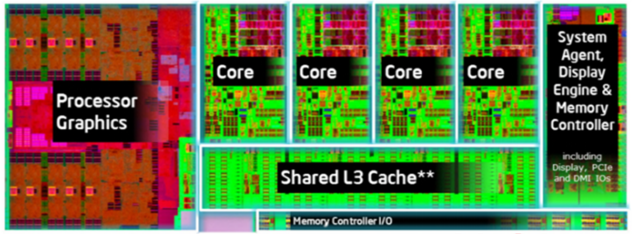
Samsung Exynos4412 Prime, ARM Cortex-A9 Quad core
1.7GHz～0.2GHz, used by Samsung's Galaxy S3



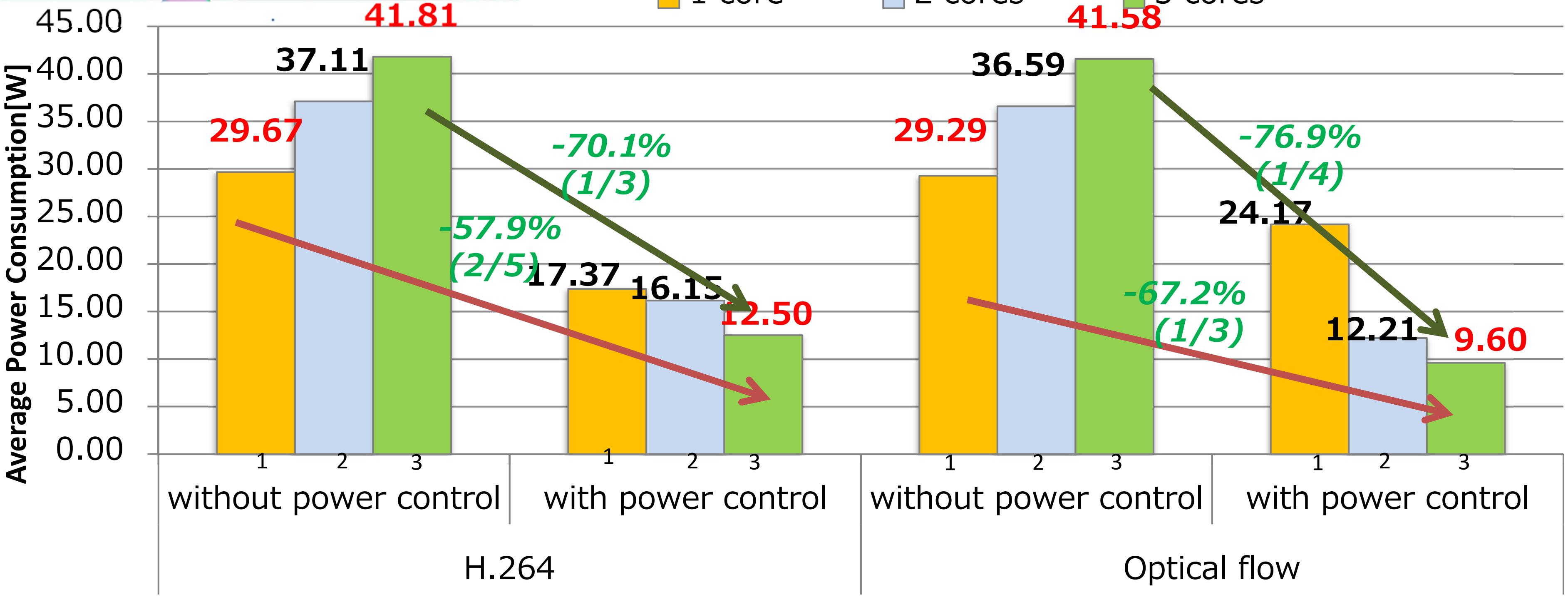
Power for 3cores was reduced to 1/5～1/7 against without software power control
Power for 3cores was reduced to 1/2～1/3 against ordinary 1core execution

Automatic Power Reuction on Intel Haswell

H.264 decoder & Optical Flow (3cores)


Processor Graphics
Core
Core
Core
Core
System Agent, Display Engine & Memory Controller
including Display, PCIe and DMI IOs
Shared L3 Cache**
Memory Controller I/O

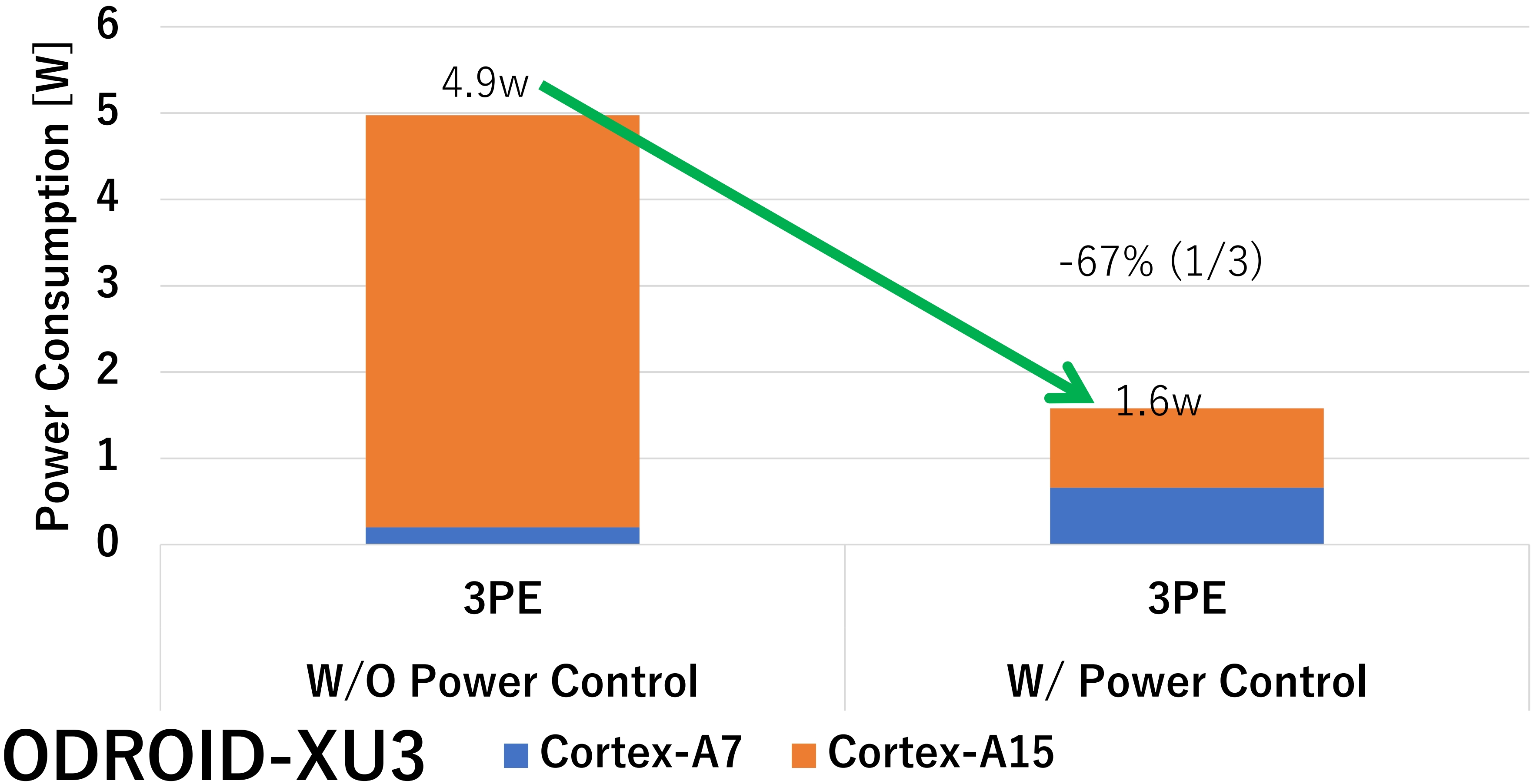
H81M-A, Intel Core i7 4770k
Quad core, 3.5GHz～0.8GHz



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

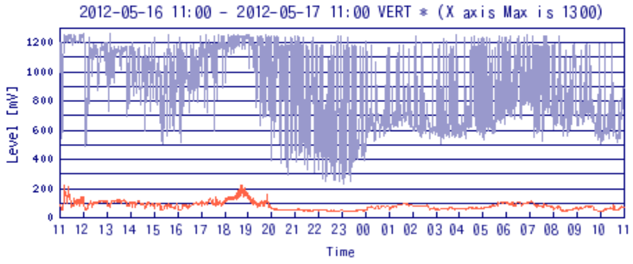
Automatic Power Reduction of OpenCV Face Detection on big.LITTLE ARM Processor

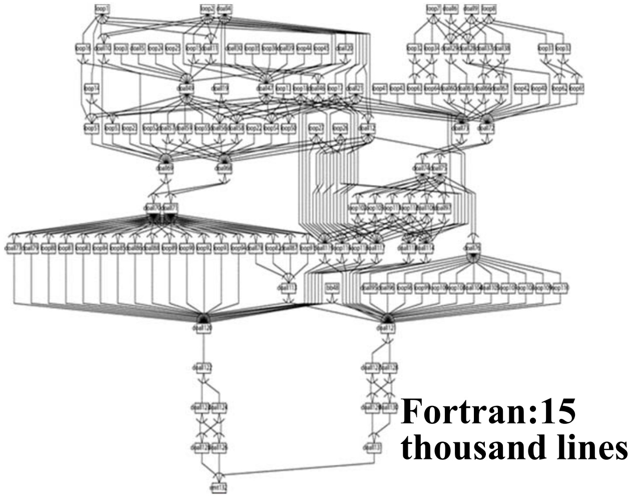


- ODROID-XU3
  - Samsung Exynos 5422 Processor
    - 4x Cortex-A15 2.0GHz, 4x Cortex-A7 1.4GHz big.LITTLE Architecture
    - 2GB LPDDR3 RAM Frequency can be changed by each cluster unit

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

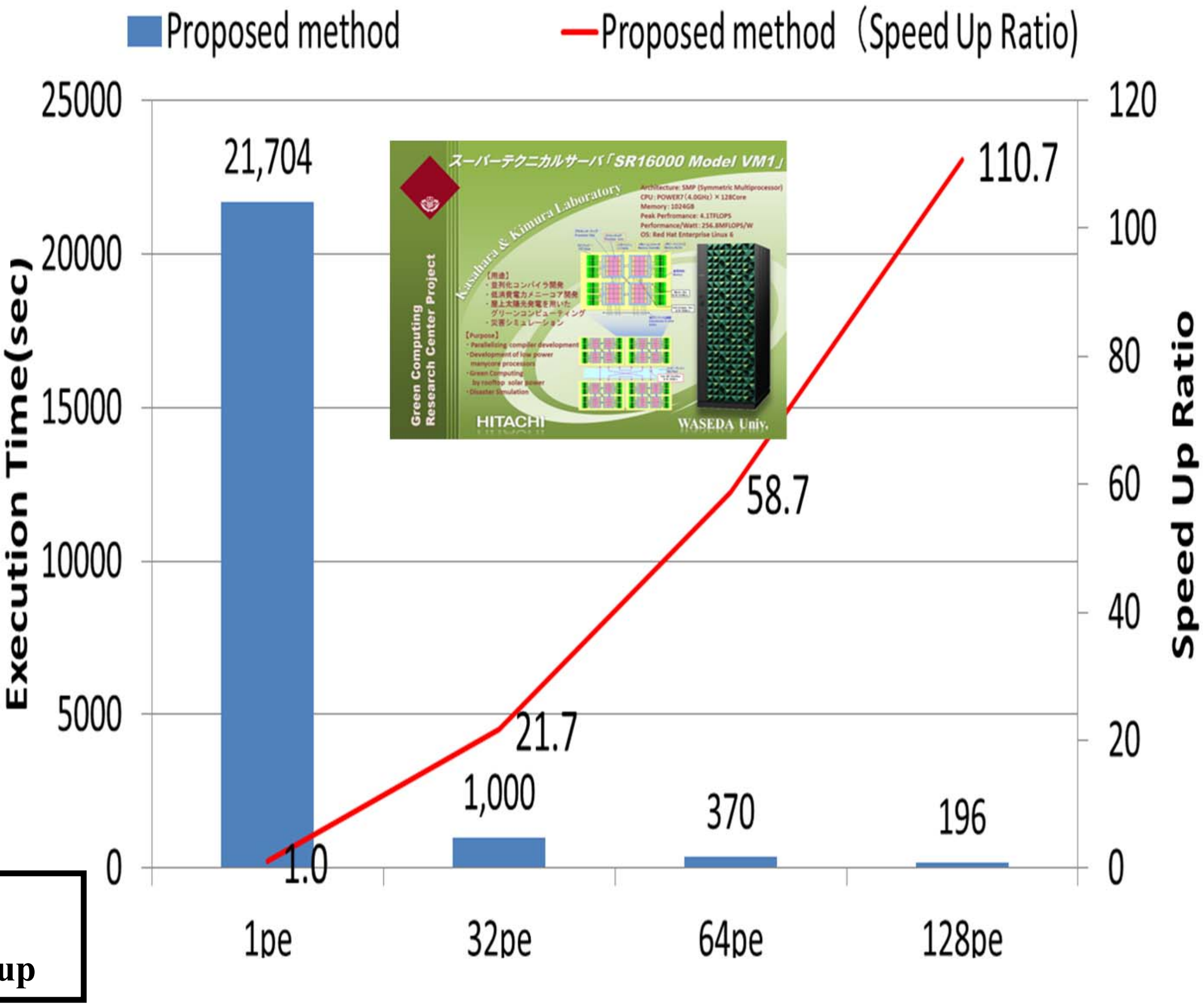
（Power7 Based 128 Core Linux SMP） (LCPC2015)





Fortran:15 thousand lines

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

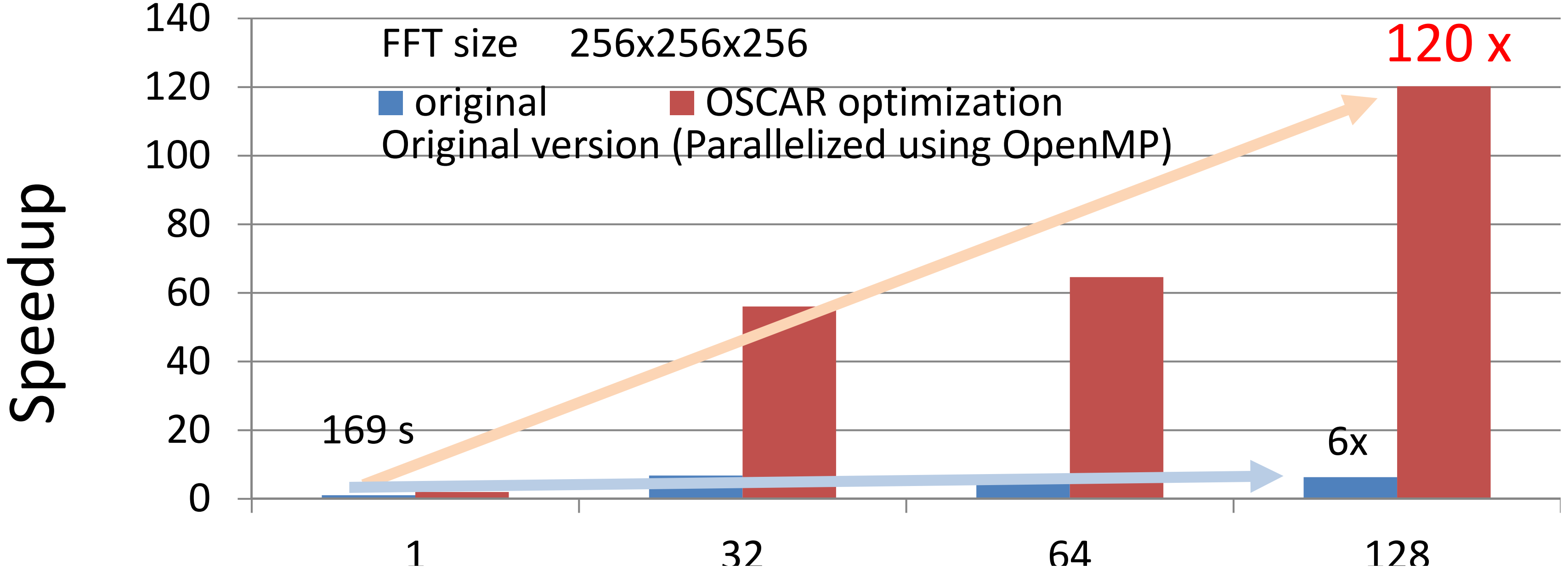


Architecture: SMP (Symmetric Multiprocessor)
CPU: POWER7（4.0GHz）× 128Core
Memory: 1024GB
Peak Perfromance: 4.1TFLOPS
Performance/Watt: 256.8MFLOPS/W
OS: Red Hat Enterprise Linux 6

【用途】
・並列化コンパイラ開発
・低消費電力メニーコア開発
・屋上太陽光発電を用いたグリーンコンピューティング
・災害シミュレーション

【Purpose】
・Parallelizing compiler development
・Development of low power manycore processors
・Green Computing by rooftop solar power
・Disaster Simulation

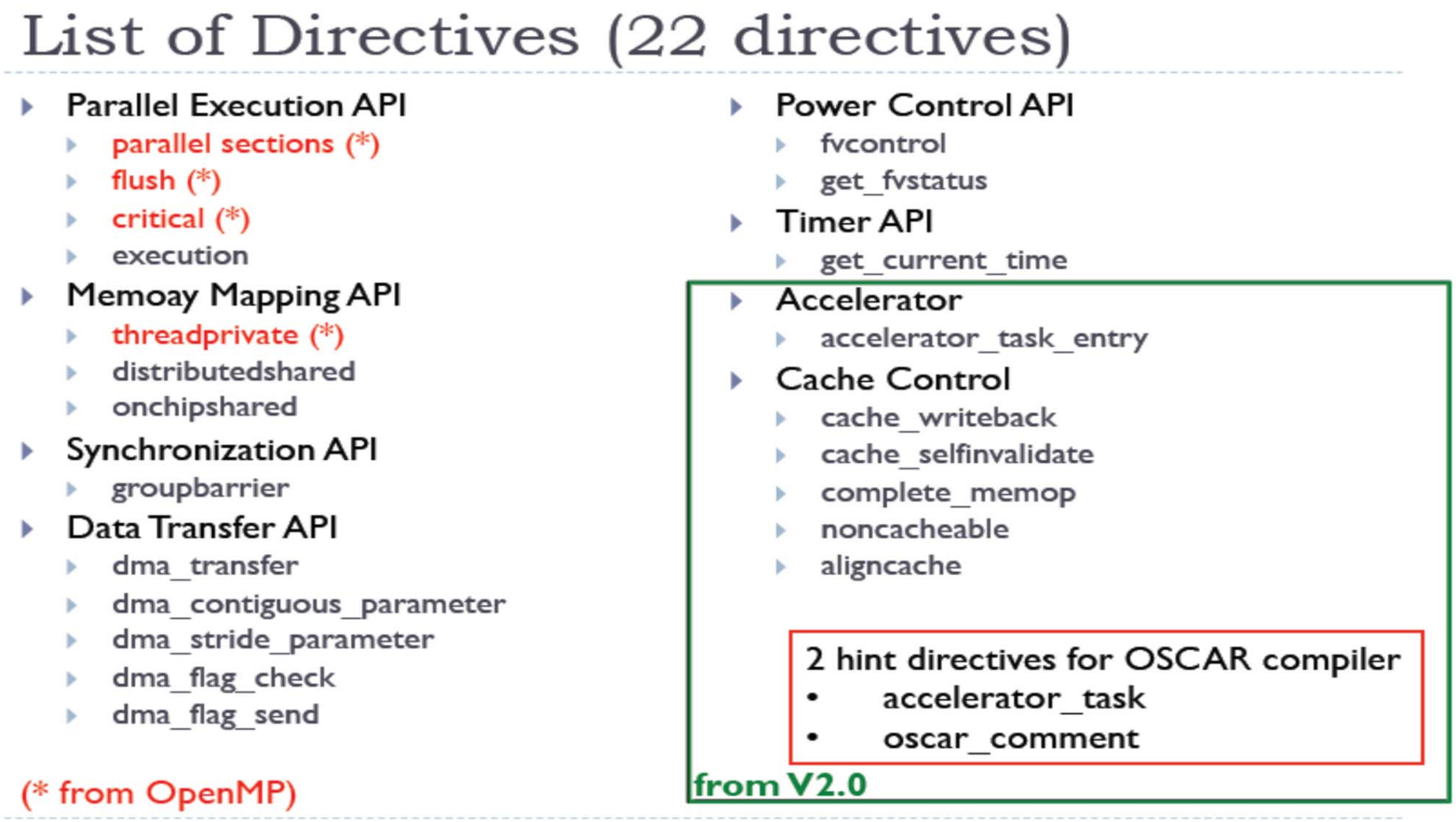
Parallelization of 3D-FFT for New Magnetic Material Computation on Hitachi SR16000 Power7 CC-Numa Server



OSCAR optimization

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

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

(* 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

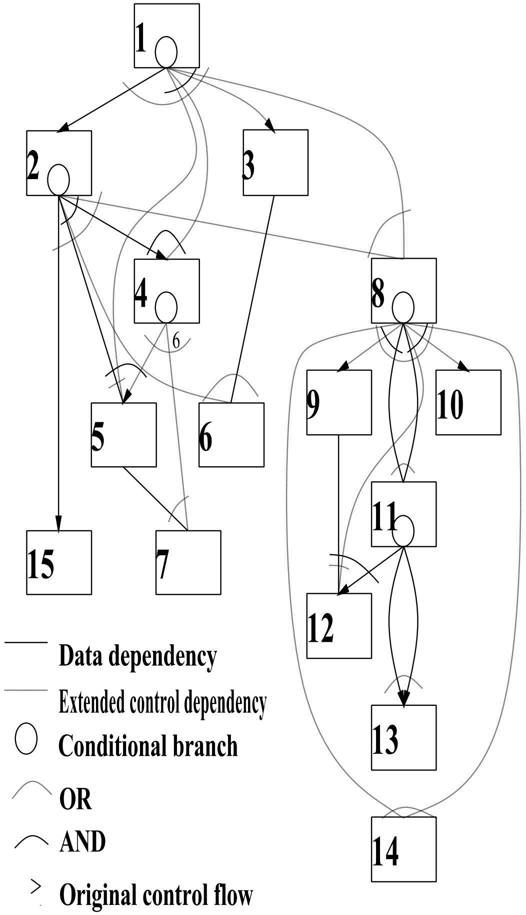
2 hint directives for OSCAR compiler

- accelerator_task
- oscar_comment

from V2.0

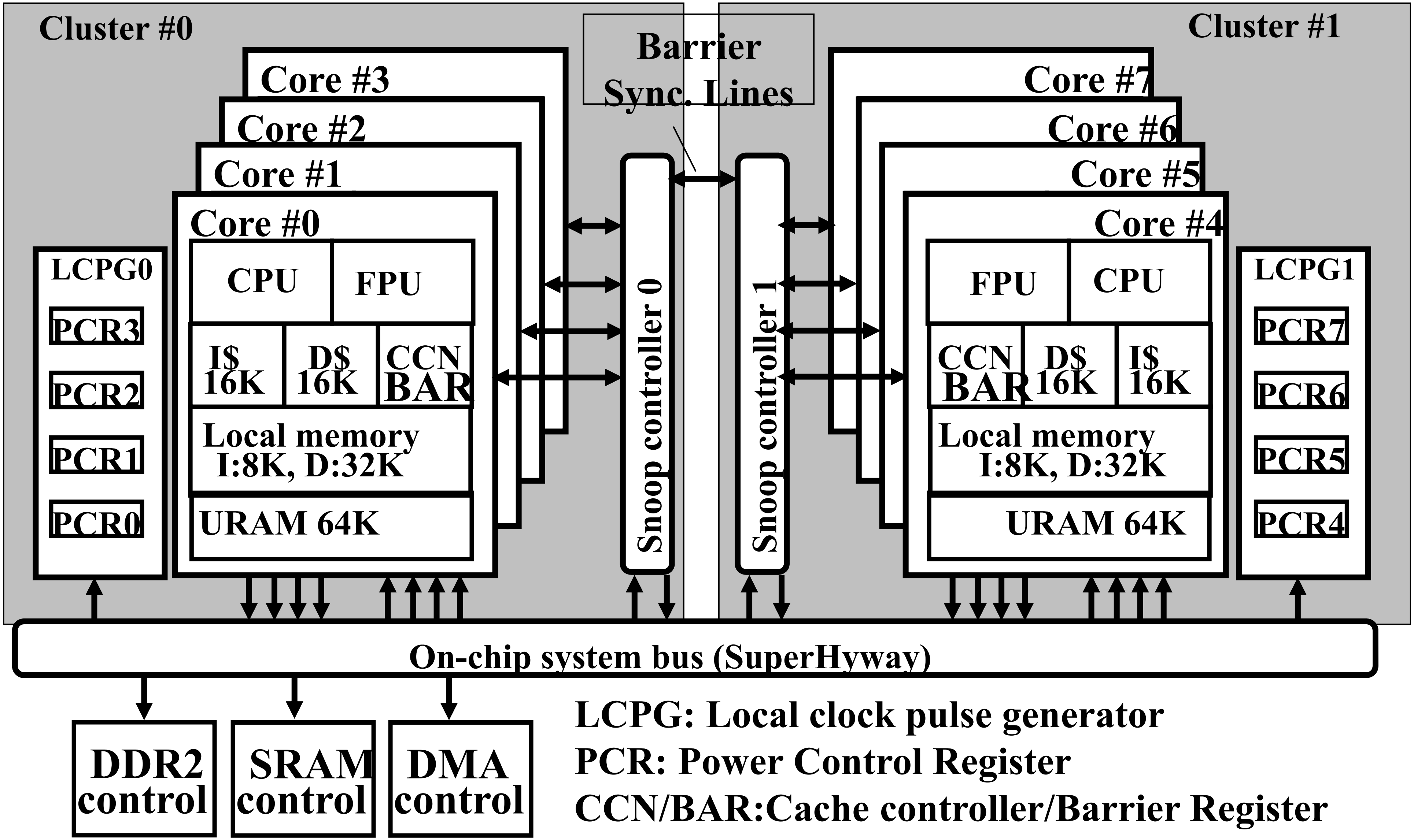
Software Coherence Control Method on OSCAR Parallelizing Compiler

- 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

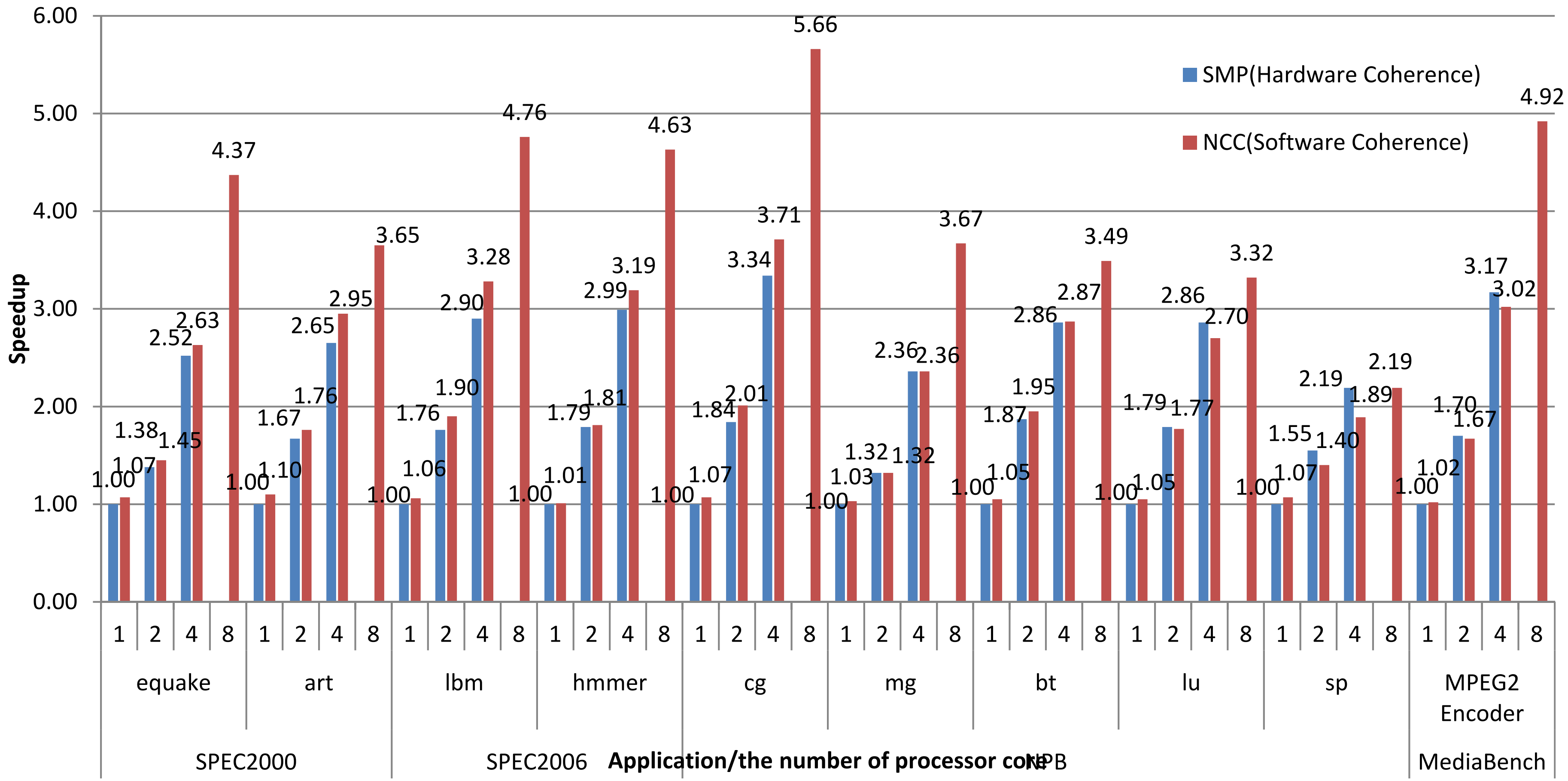
8 Core RP2 Chip Block Diagram



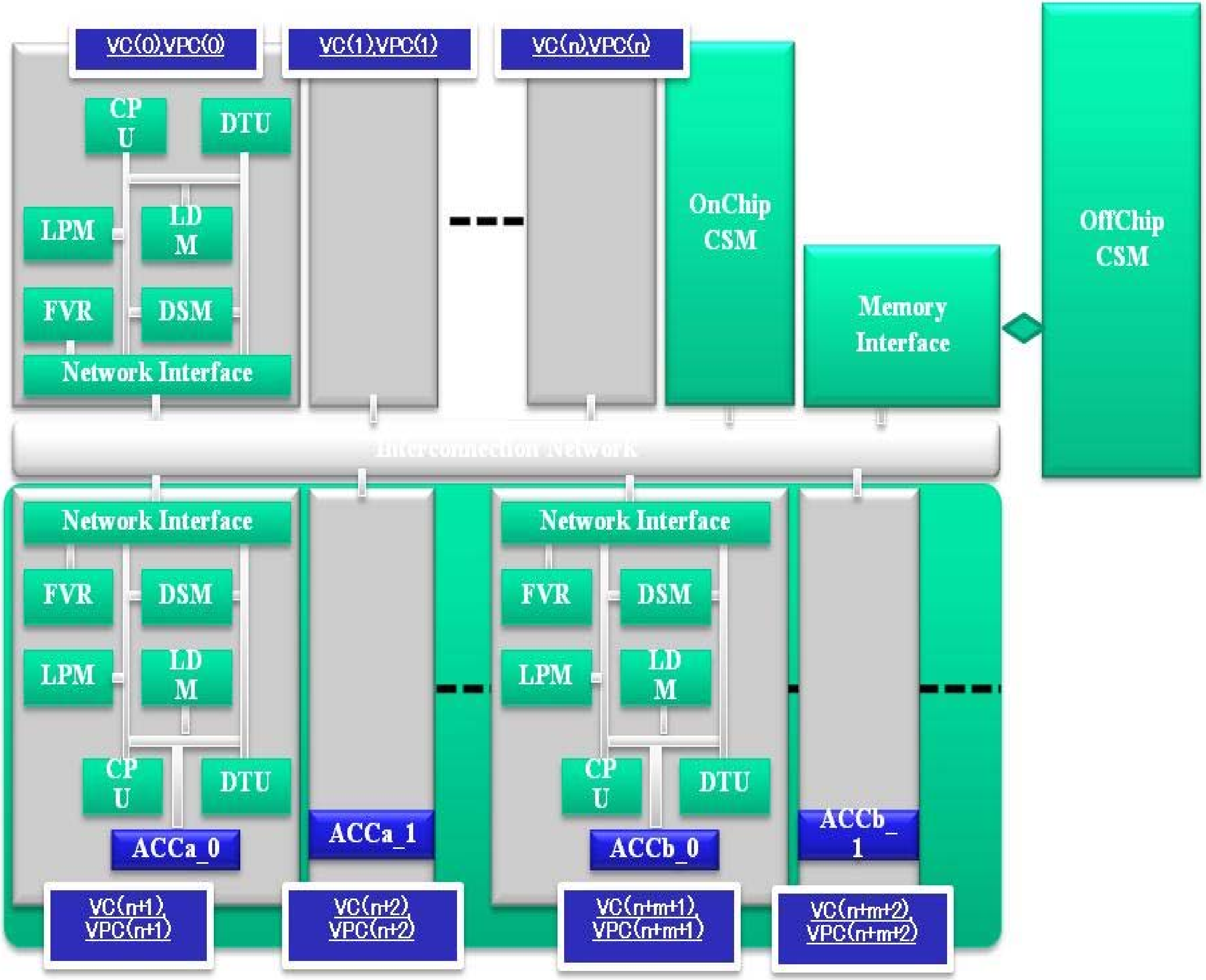
LCPG: Local clock pulse generator
PCR: Power Control Register
CCN/BAR:Cache controller/Barrier Register
URAM: User RAM (Distributed Shared Memory)

Automatic Software Coherent Control for Manycores

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



OSCAR Heterogeneous Multicore



DTU
- Data Transfer Unit

LPM
- Local Program Memory

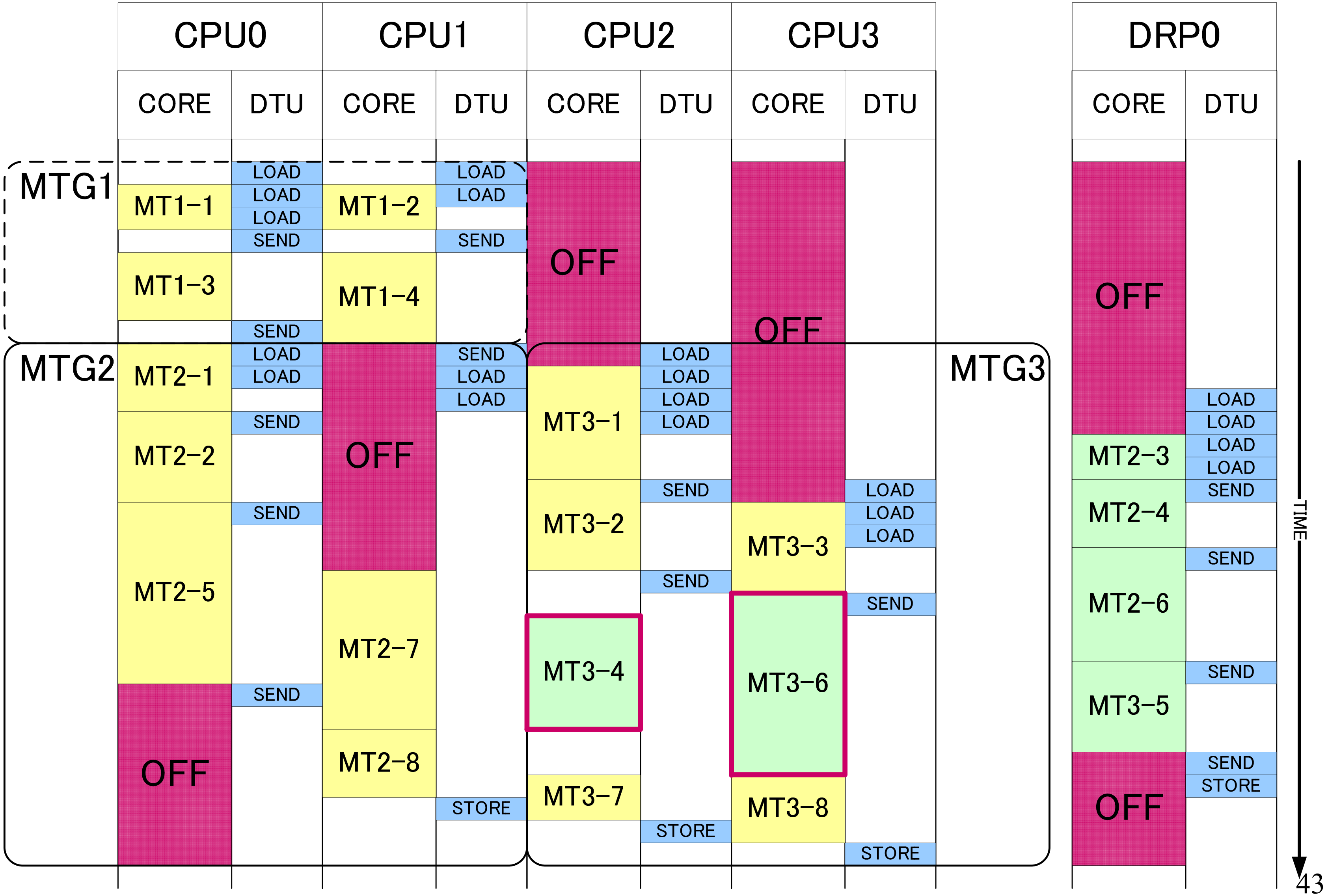
LDM
- Local Data Memory

DSM
- Distributed Shared Memory

CSM
- Centralized Shared Memory

FVR
- Frequency/Voltage Control Register

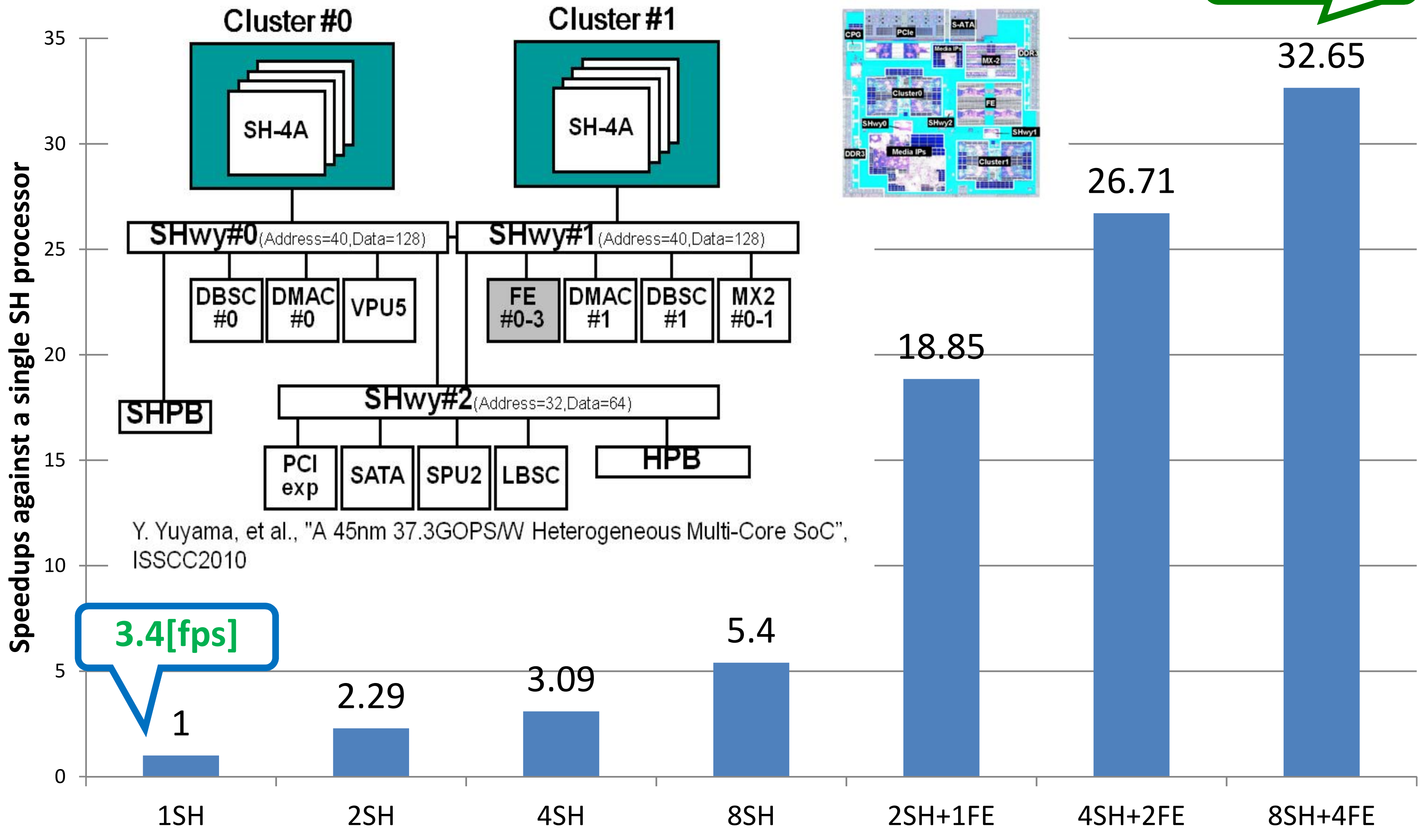
An Image of Static Schedule for Heterogeneous Multi-core with Data Transfer Overlapping and Power Control



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

(Optical Flow with a hand-tuned library)

111[fps]



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

3.4[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)

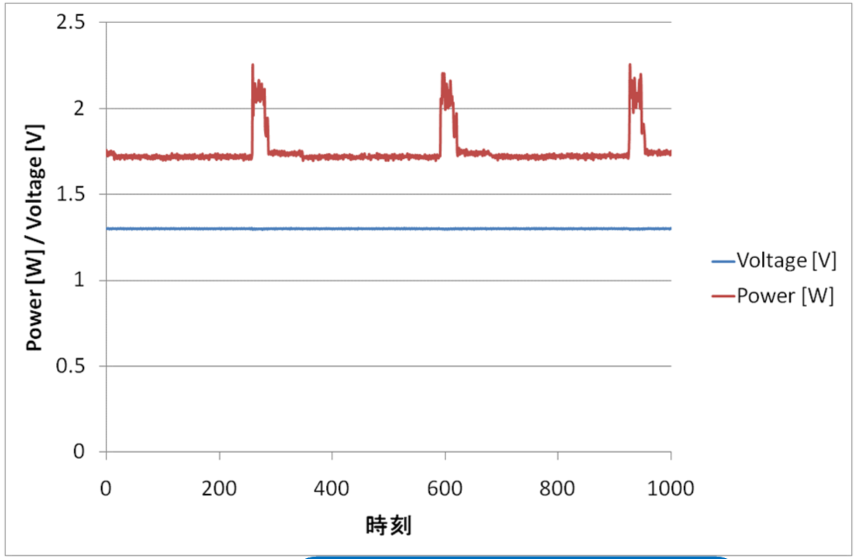
Without Power Reduction

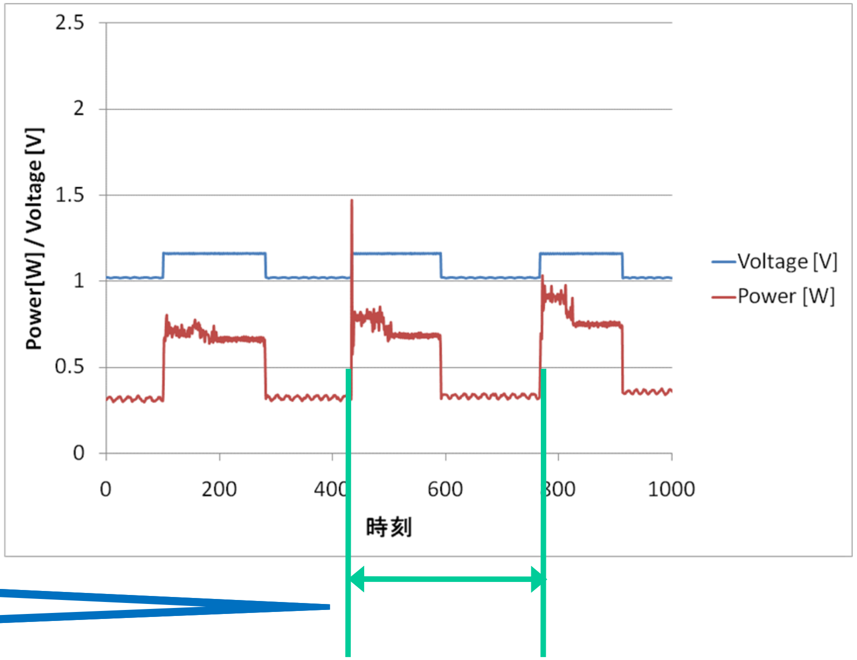
With Power Reduction by OSCAR Compiler

70% of power reduction

Average:1.76[W]

Average:0.54[W]





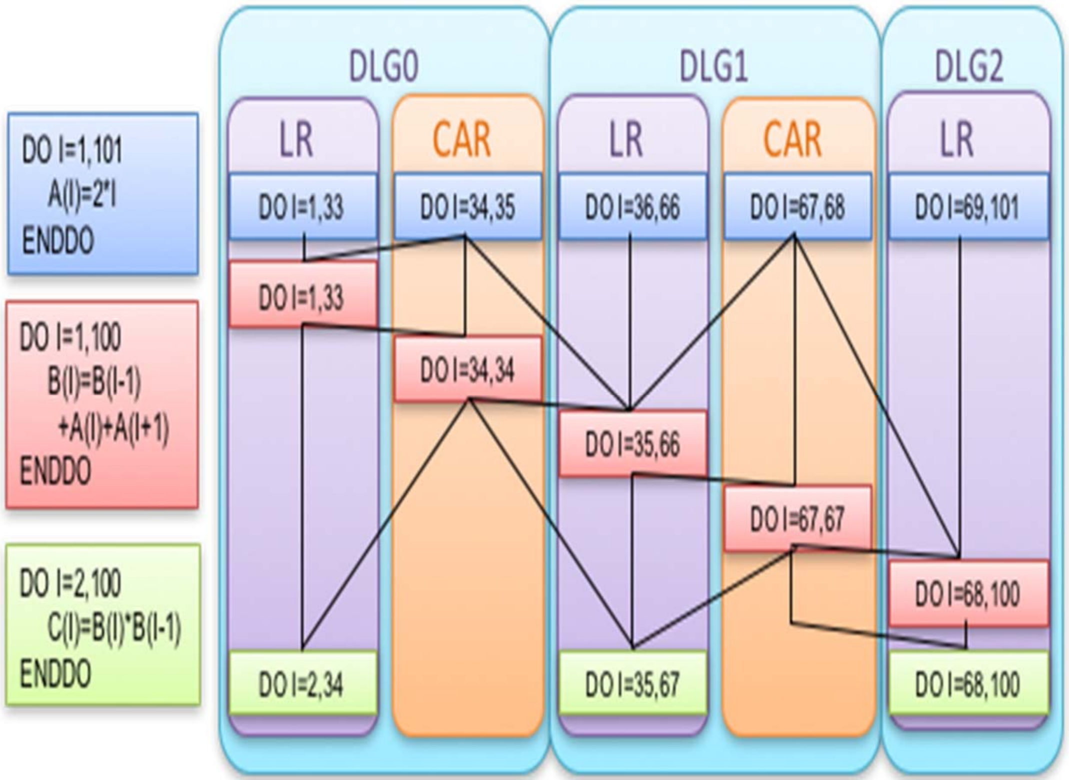
1cycle : 33[ms]
→30[fps]

Automatic Local Memory Management

Data Localization: Loop Aligned Decomposition

- Decomposed loop into 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

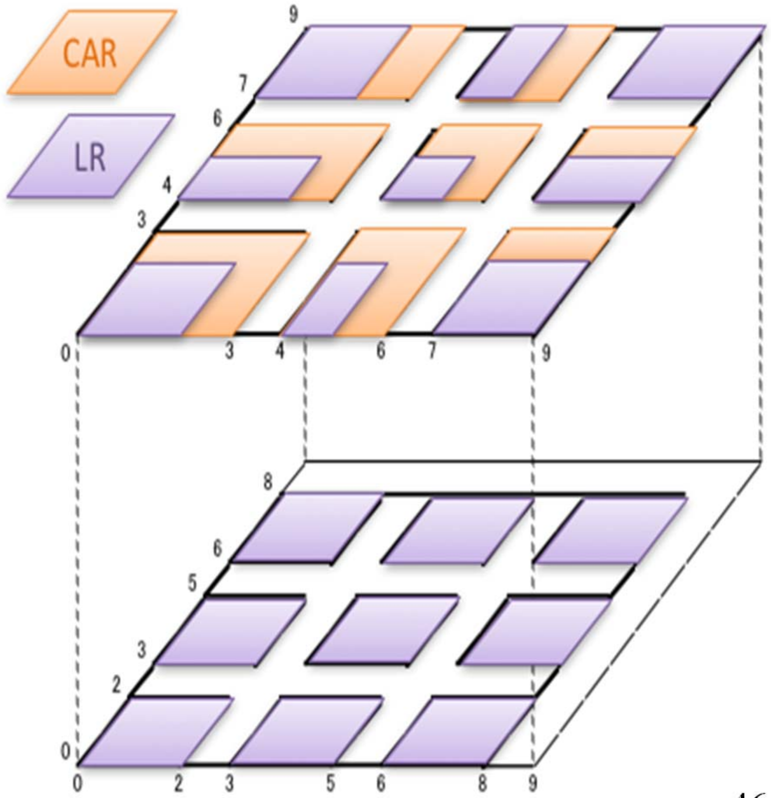


```
DO I=1,101
  A(I)=2*I
ENDDO
```

```
DO I=1,100
  B(I)=B(I-1)
    +A(I)+A(I+1)
ENDDO
```

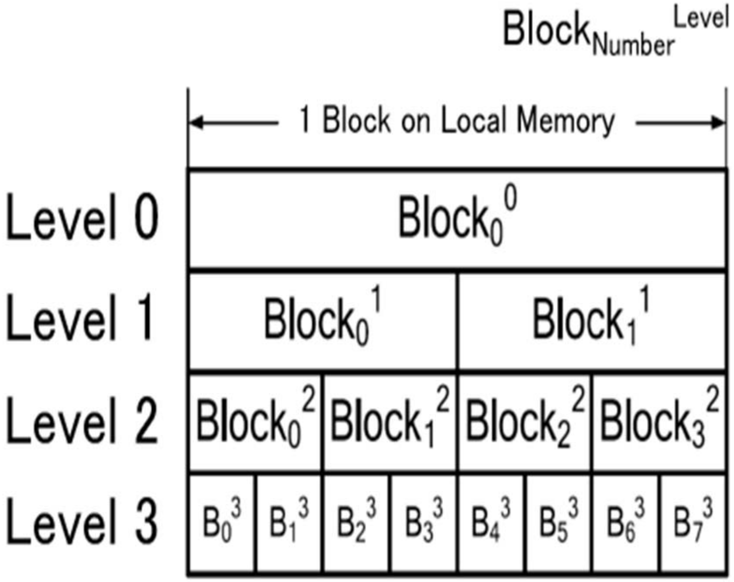
```
DO I=2,100
  C(I)=B(I)*B(I-1)
ENDDO
```

Multi-dimension Decomposition



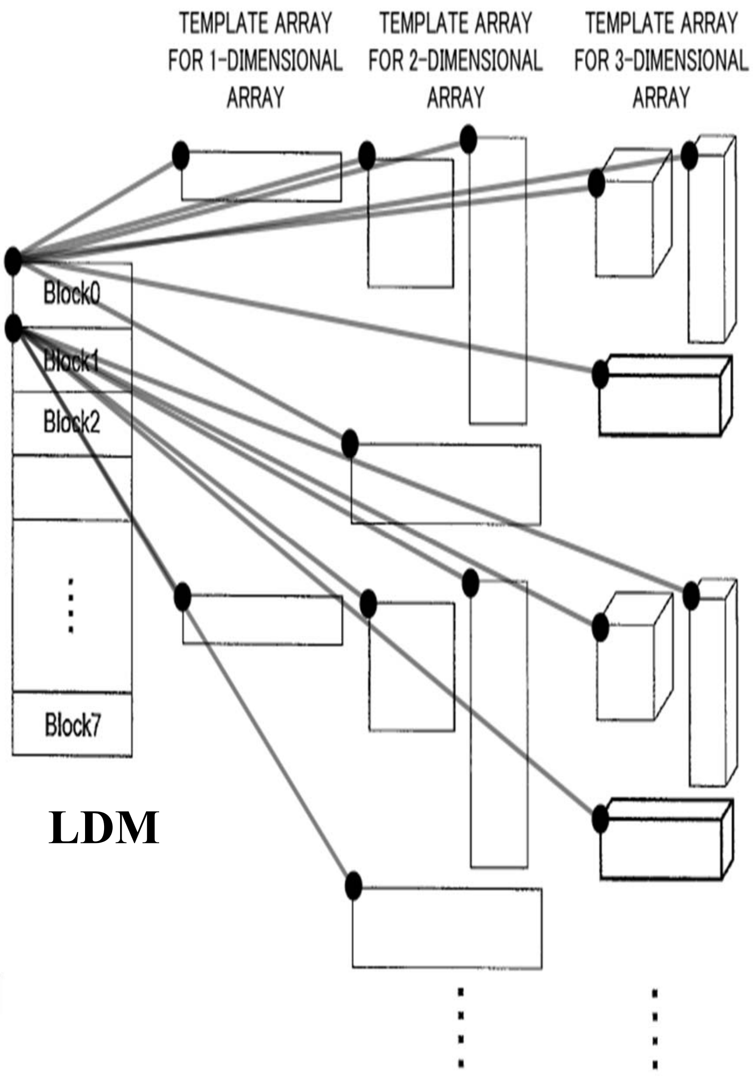
Adjustable Blocks

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

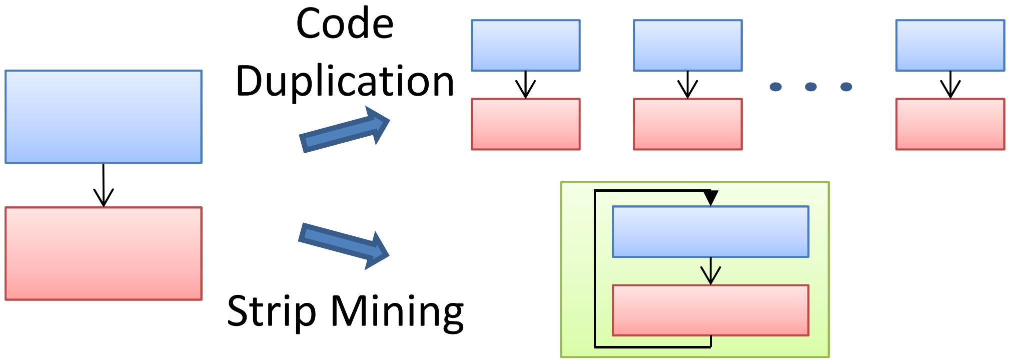

TEMPLATE ARRAY FOR 1-DIMENSIONAL ARRAY
TEMPLATE ARRAY FOR 2-DIMENSIONAL ARRAY
TEMPLATE ARRAY FOR 3-DIMENSIONAL ARRAY
Block0
Block1
Block2
Block7
LDM

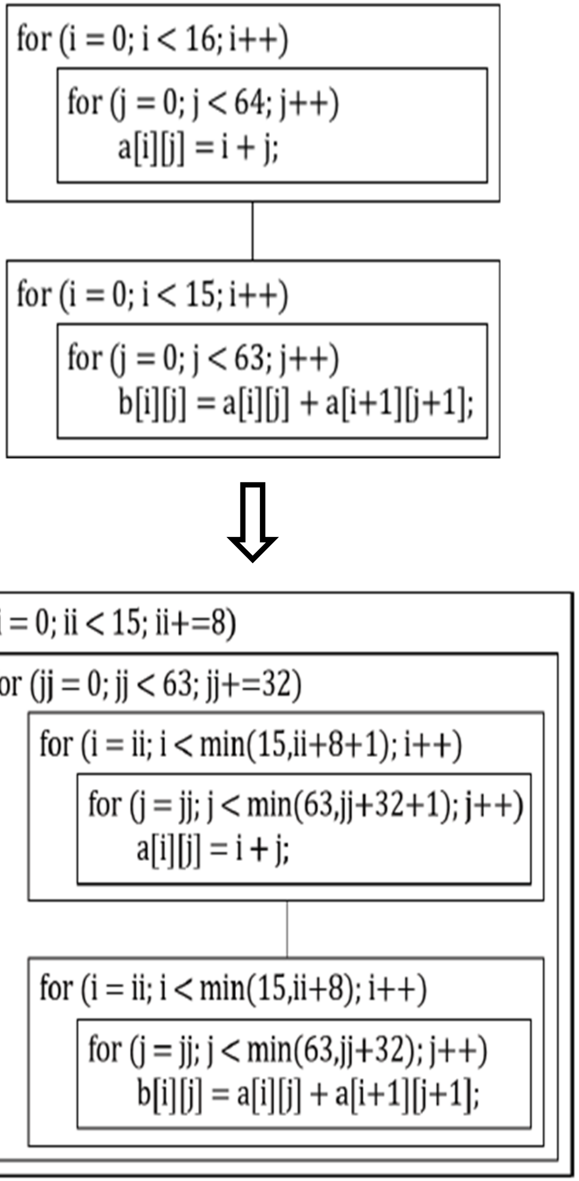
Block Replacement Policy

- 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



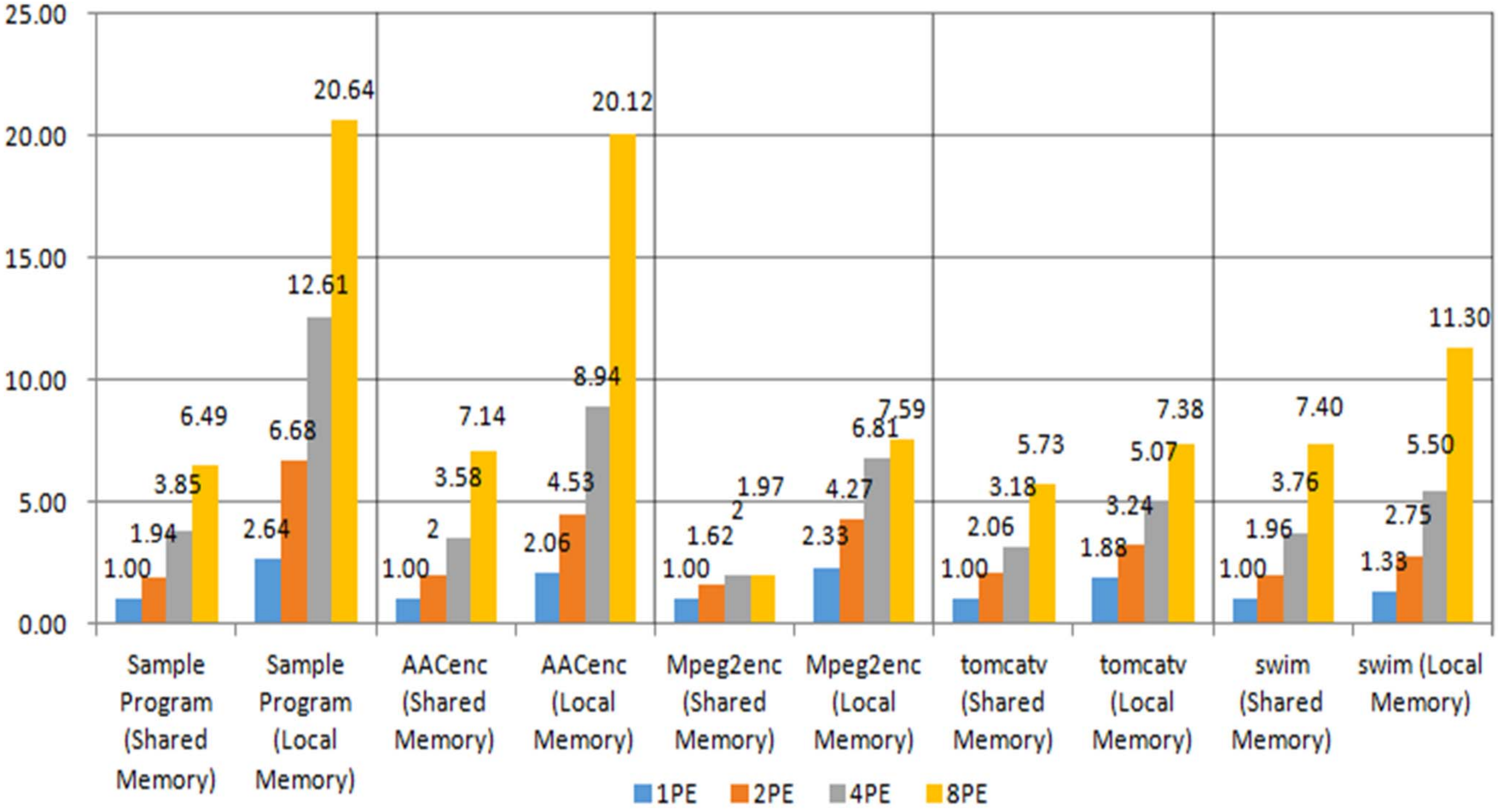


```
for (i = 0; i < 16; i++)
    for (j = 0; j < 64; j++)
        a[i][j] = i + j;

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

⇩

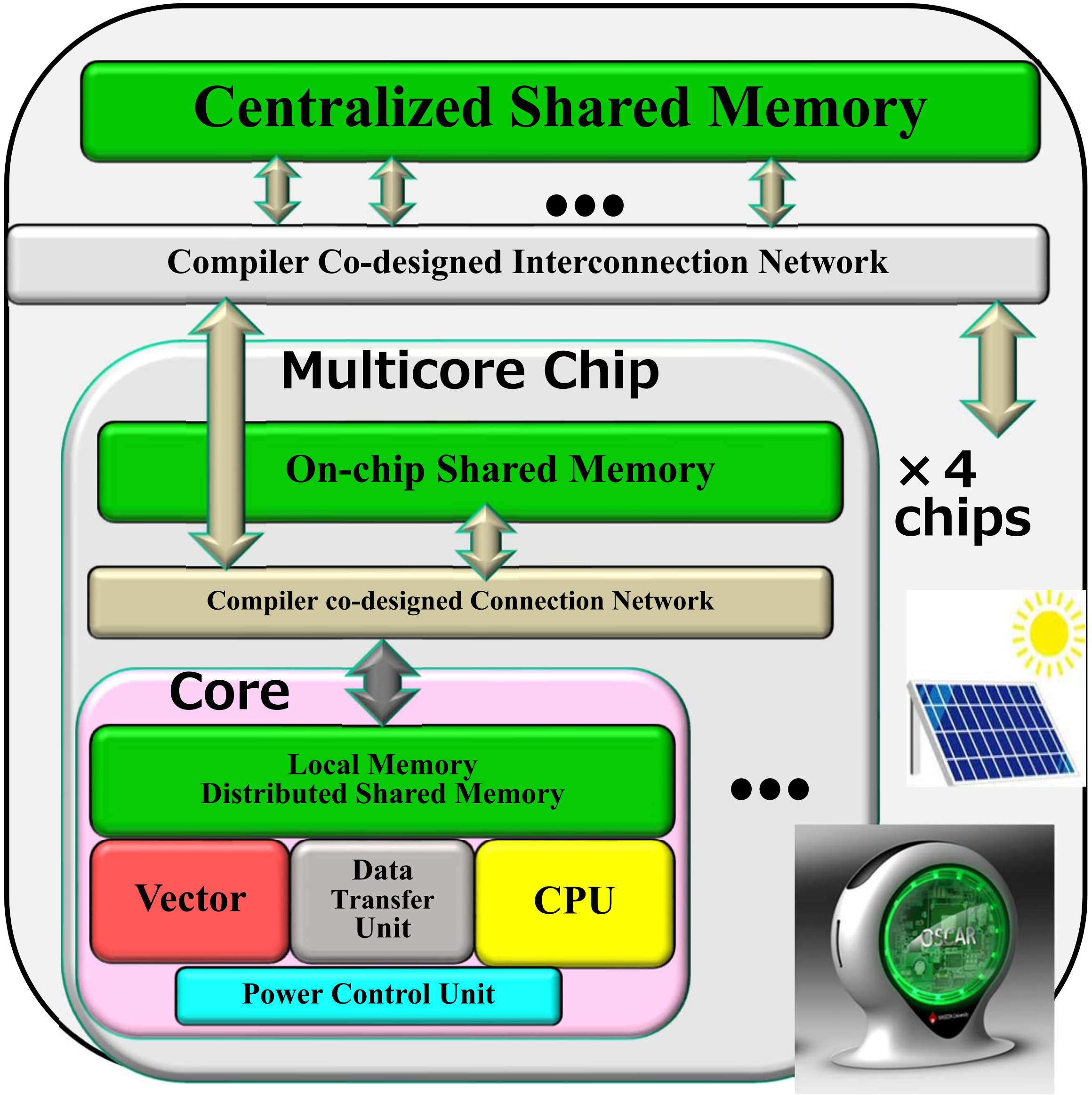
```
for (ii = 0; ii < 15; ii+=8)
    for (jj = 0; jj < 63; jj+=32)
        for (i = ii; i < min(15,ii+8+1); i++)
            for (j = jj; j < min(63,jj+32+1); j++)
                a[i][j] = i + j;

        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

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



Centralized Shared Memory

●●●

Compiler Co-designed Interconnection Network

Multicore Chip

On-chip Shared Memory

Compiler co-designed Connection Network

Core

Local Memory
Distributed Shared Memory

Vector

Data Transfer Unit

CPU

Power Control Unit

●●●

×4 chips

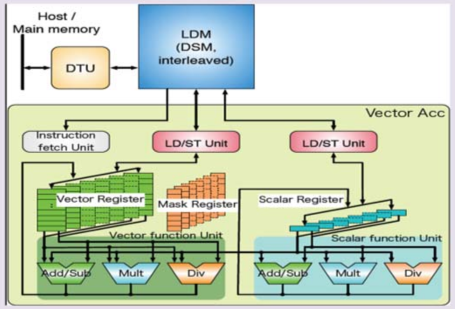
Target:

- 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

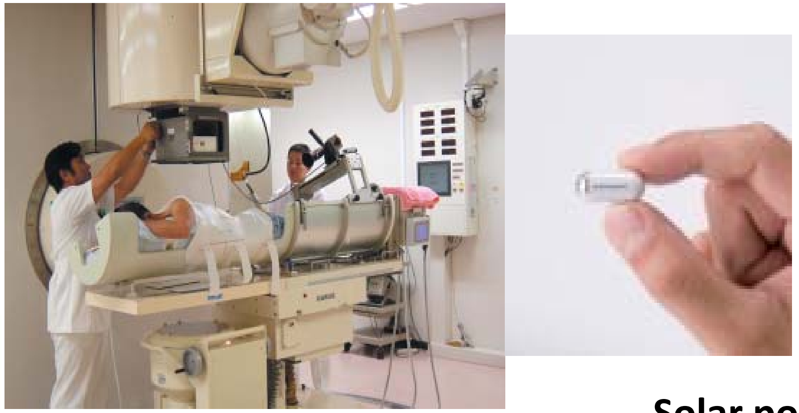
Next Generation Automobiles

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