Automatic Cache and Local Memory Optimization for Multicores

Hironori Kasahara
Professor, Dept. of Computer Science & Engineering
Director, Advanced Multicore Processor Research Institute
Waseda University, Tokyo, Japan
IEEE Computer Society
President Elect 2017, President 2018
URL: http://www.kasahara.cs.waseda.ac.jp/

Hironori Kasahara Voted 2017 IEEE Computer Society President-Elect

LOS ALAMITOS, Calif., 30 September 2016 – Hironori Kasahara, a Professor of Computer Science at Waseda University in Tokyo, and Director of the Advanced Multicore Research Institute, has been voted **IEEE Computer Society 2017 President-Elect.**

Kasahara is a former member of the IEEE-CS Board of Governors, has served as chair of the IEEE-CS Multicore STC and CS Japan Chapter, and board member of the IEEE Tokyo Section. Kasahara will serve as the 2018 IEEE CS President for a one-year term beginning 1 January 2018. Kasahara garnered 3,278 votes, compared with 2,804 votes cast for Hausi A. Müller, a Professor of Computer Science and Associate Dean of Research, Faculty of Engineering at University of Victoria, Canada, and a member of IEEE-CS Board of Governors.

The President oversees IEEE-CS programs and operations and is a nonvoting member of most IEEE-CS program boards and committees. The 2016 election had a 12.69% turnout, with 6,357 ballots cast. The turnout was higher than the 2015 election with and 12.68% turnout (6,239 ballots cast) and the 2014 election with a 12.66% turnout (6,728 ballots cast).

2016 IEEE Computer Society Election Results

Press Release | Ballot counts

Posted 29 September 2016

Hironori Kasahara selected 2017 President-Elect (2018 President)

Hironori Kasahara has served as a chair or member of 225 society and government committees, including a member of the CS Board of Governors; chair of CS Multicore STC and CS Japan chapter; associate editor of IEEE Transactions on Computers; vice PC chair of the 1996 ENIAC 50th Anniversary International Conference on Supercomputing; general chair of LCPC; PC member of SC, PACT, PPoPP, and ASPLOS; board member of IEEE Tokyo section; and member of the Earth Simulator committee.

He received a PhD in 1985 from Waseda University, Tokyo, joined its faculty in 1986, and has been a professor of computer science since 1997 and a director of the Advanced Multicore Research Institute since 2004. He was a visiting scholar at University of California, Berkeley, and the University of Illinois at Urbana–Champaign's Center for Supercomputing R&D.

Kasahara received the CS Golden Core Member Award, IFAC World Congress Young Author Prize, IPSJ Fellow and Sakai Special Research Award, and the Japanese Minister's Science and Technology Prize. He led Japanese national projects on parallelizing compilers and embedded multicores, and has presented 210 papers, 132 invited talks, and 27 patents. His research has appeared in 520 newspaper and Web articles.

Past IEEE Computer Society Presidents

Chairs of the IRE Professional Group on Electronic Computers

1951-53 Morton M. Astrahan
1953-54 John H. Howard
1954-55 Harry Larson
1955-56 Jean H. Felker
1956-57 Jerre D. Noe
1957-58 Werner Buchholz
1958-59 Willis H. Ware
1959-60 Richard O. Endres
1960-62 Arnold A. Cohen
1962-64 Walter L. Anderson

Chairs of the AIEE Committee on Large-Scale Computing Devices

1946-49 Charles Concordia
1949-51 John Grist Brainerd
1951-53 Walter H. MacWilliams
1953-55 Frank J. Maginniss
1955-57 Edwin L. Harder
1957-59 Morris Rubinoff
1959-61 Ruben A. Imm
1961-63 Claude A. Kagan
1963-64 Gerhard L. Hollander

Chairs & Presidents of the IEEE Computer Society

1964-65 Keith Uncapher
1965-66 Richard I. Tanaka
1966-67 Samuel Levine
1968-69 Charles L. Hobbs
1970-71 Edward J. McCluskey
1972-73 Albert S. Hoagland
1974-75 Stephen S. Yau
1976 Dick B. Simmons
1977-78 Merlin G. Smith
1979-80 Tse-Yun Feng
1981 Richard E. Merwin
1982-83 Oscar N. Garcia
1984-85 Martha Sloan
1986-87 Roy L. Russo
1988 Edward A. Parrish
1989 Kenneth A. Anderson
1990 Helen M. Wood
1991 Duncan H. Lawrie
1992 Bruce D. Shriver
1993 James H. Aylor
1994 Laurel V. Kaleda
1995 Ronald G. Hoelzeman
1996 Mario R. Barbacci
1997 Barry W. Johnson
1998 Doris L. Carver
1999 Leonard L. Tripp
2000 Guylaine M. Pollock
2001 Benjamin W. Wah
2002 Willis K. King
2003 Stephen Diamond
2004 Carl K. Chang
2005 Gerald L. Engel
2006 Deborah M. Cooper
2007 Michael R. Williams
2008 Rangachar Kasturi
2009 Susan K. (Kathy) Land,
2010 James D. Isaak
2011 Sorel Reisman
2012 John W. Walz
2013 David Alan Grier
2014 Dejan S. Milojicic
2015 Thomas M. Conte
2016 Roger U. Fujii
2017 Jean-Luc Gaudiot
2018 Hironori Kasahara

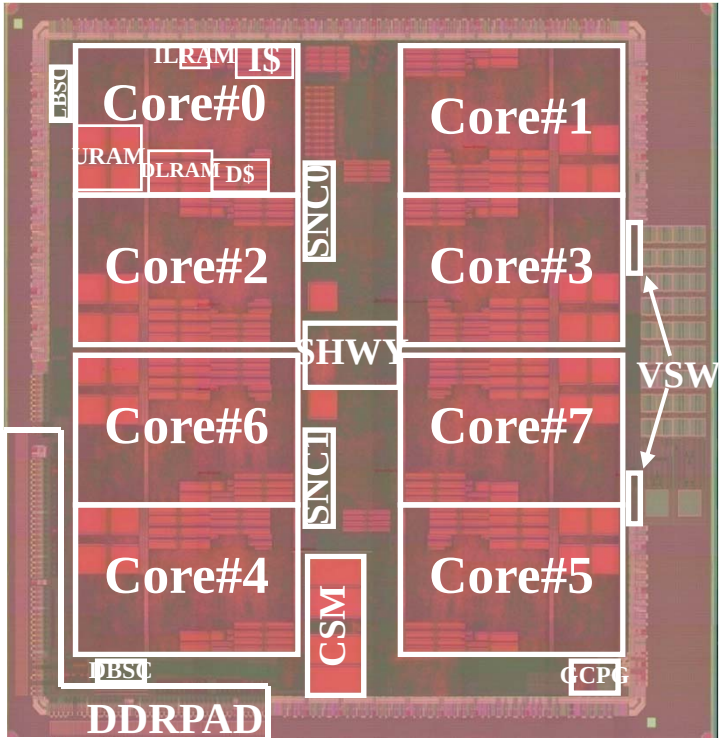
IEEE Computer Society BoG (Board of Governors) Feb.1, 2017

Toward 2018

1. Refining content and services to further improve the satisfaction of CS members;
2. Considering an incentive for volunteers to further accelerate CS activities and promptly provide technical benefits for people around the globe;
3. Offering more attractive services for practitioners in industry;
4. Providing the world's best educational content and historical treasures for future generations, which only the CS can create with our pioneering researchers (for example, the Multicore Compiler Video Series found at www.computer.org/web/education/multicore-video-series);
5. Thinking about sustainable membership fees while considering the diversity of economic situations within the 10 regions;
6. Cooperating with other IEEE societies and sister societies in a timely and efficient manner;
7. Intelligibly introducing the latest computer-related technologies to younger generations, including children, so that they can realize their technological dreams.

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)

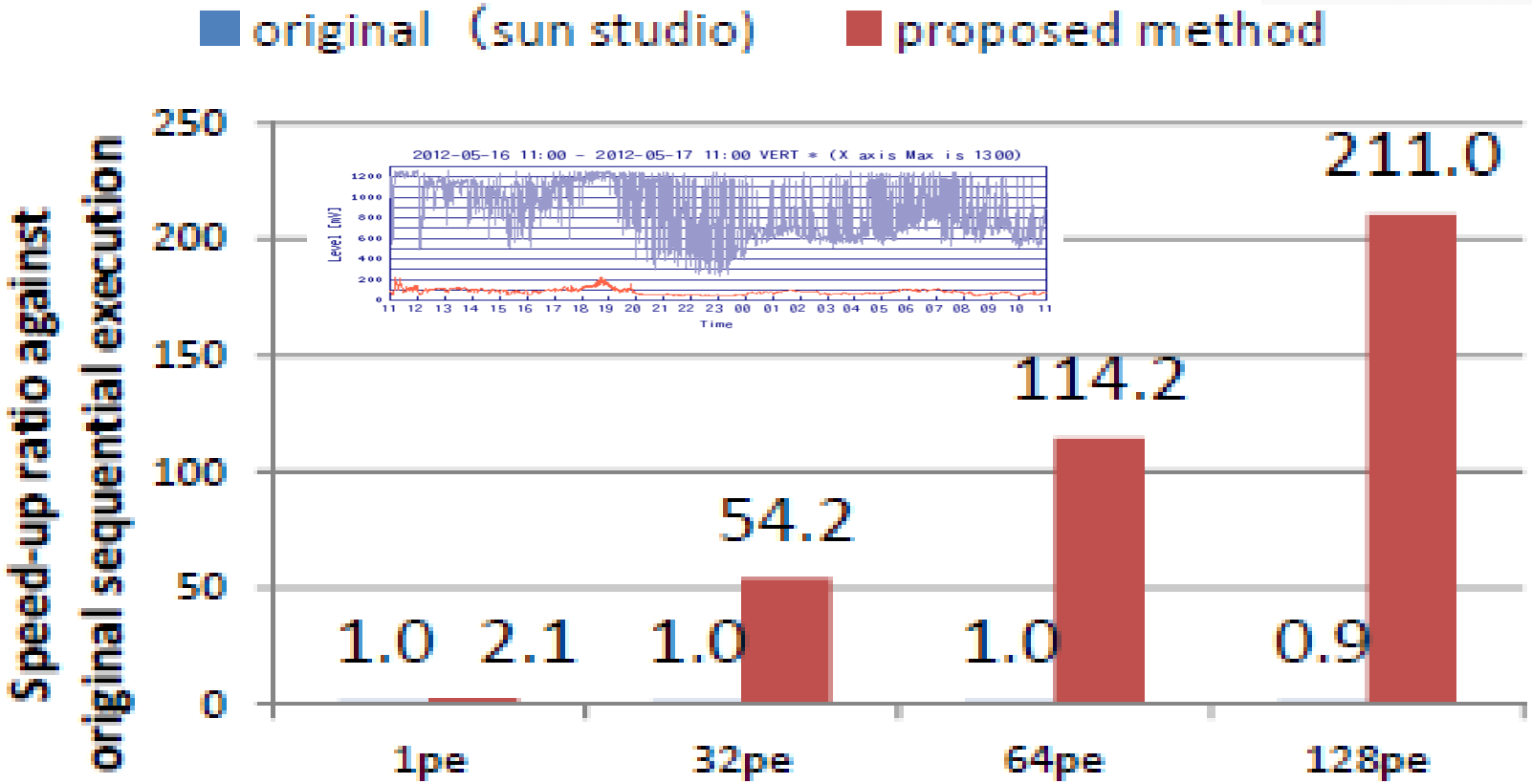
⇒ Power ∝ Frequency^3

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.

Earthquake Simulation "GMS" on Fujitsu M9000 Sparc CC-NUMA Server



With 128 cores, OSCAR compiler gave us 100 times speedup against 1 core execution and 211 times speedup against 1 core using Sun (Oracle) Studio compiler.

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

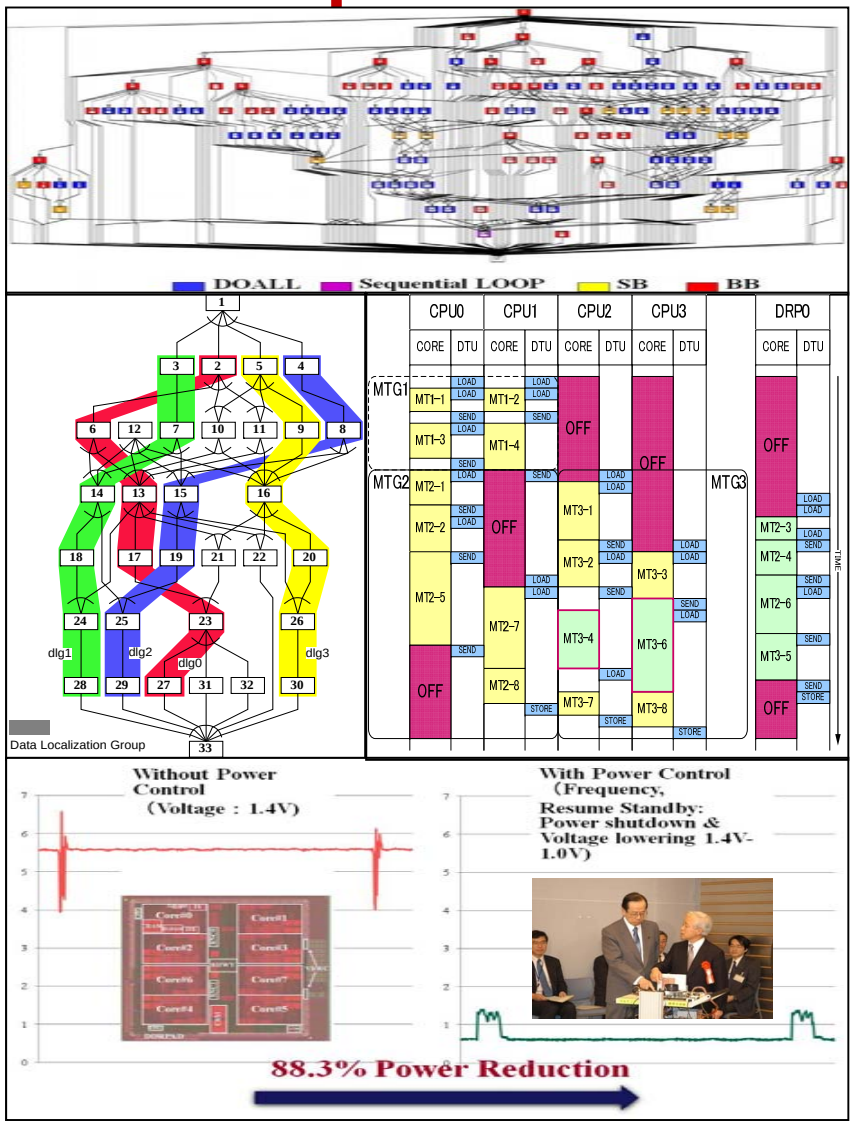
Automatic data management for distributed shared memory, cache and local memory

Data Transfer Overlapping

Data transfer overlapping using Data Transfer Controllers (DMAs)

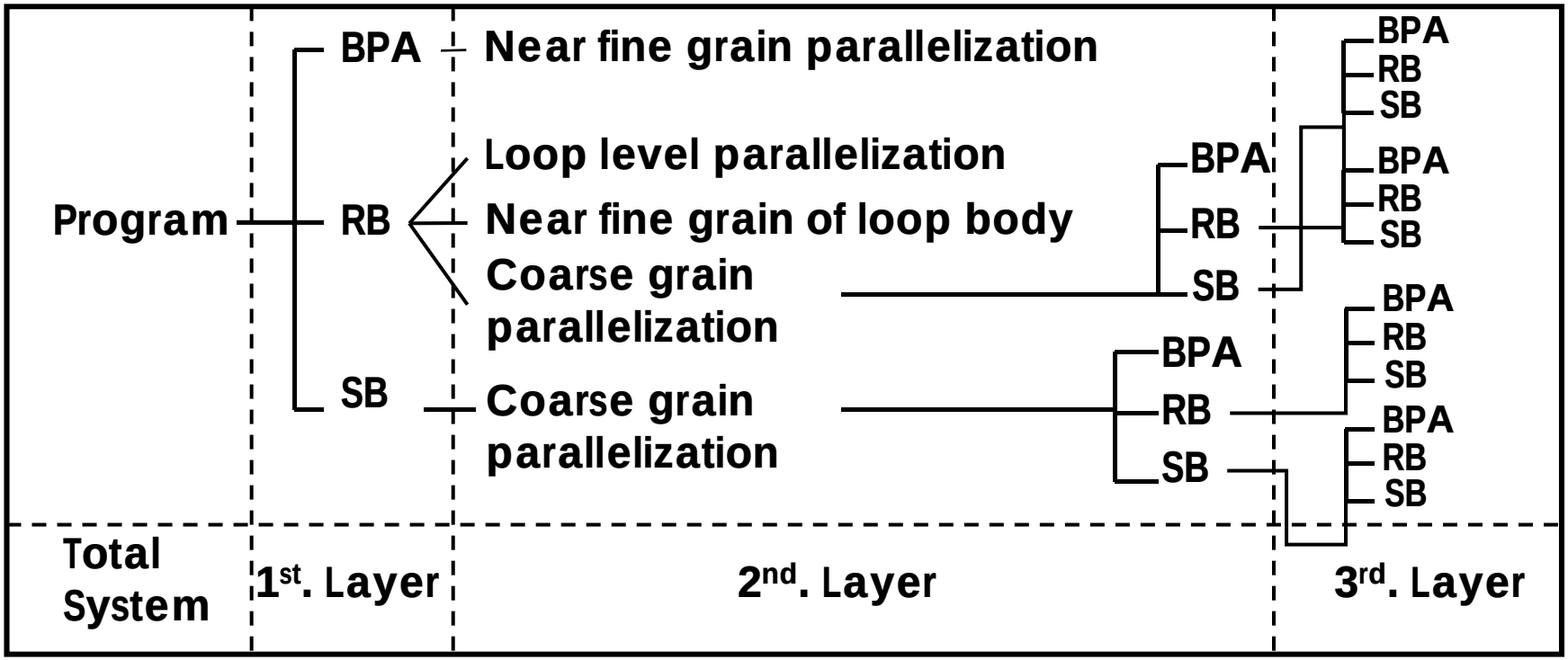
Power Reduction

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

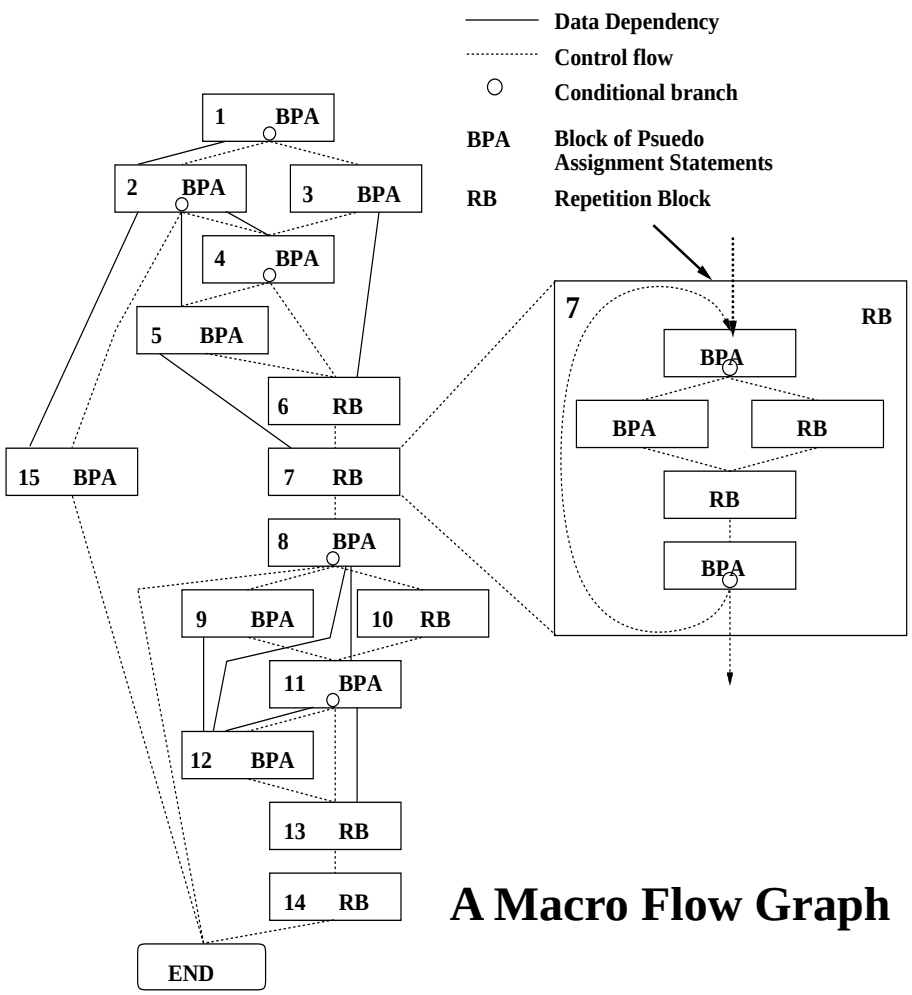


Generation of Coarse Grain Tasks

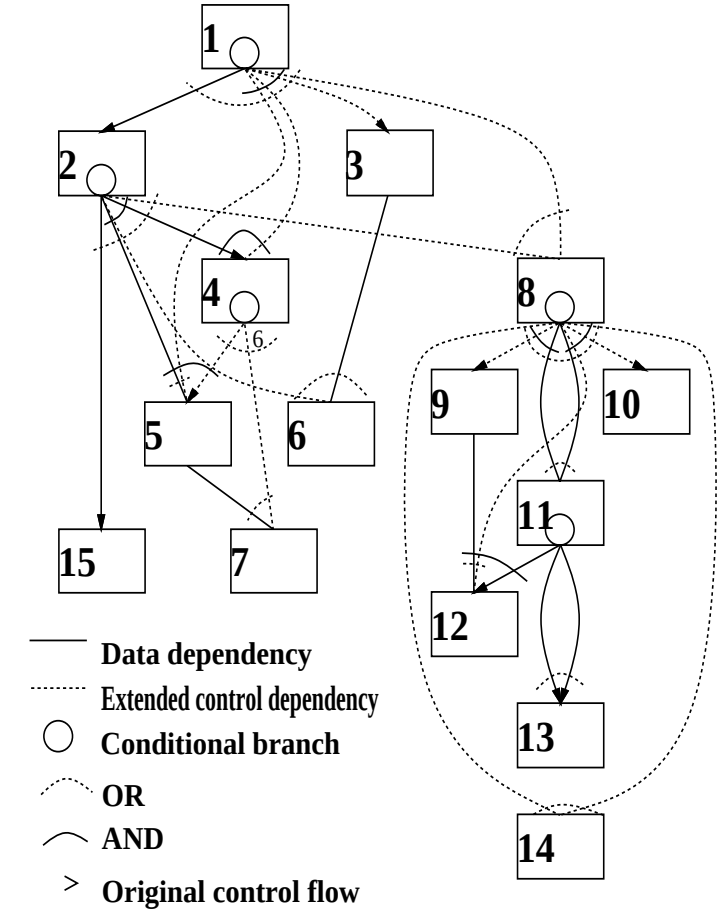
- **Macro-tasks (MTs)**
  - **Block of Pseudo Assignments (BPA): Basic Block (BB)**
  - **Repetition Block (RB) : natural loop**
  - **Subroutine Block (SB): subroutine**



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



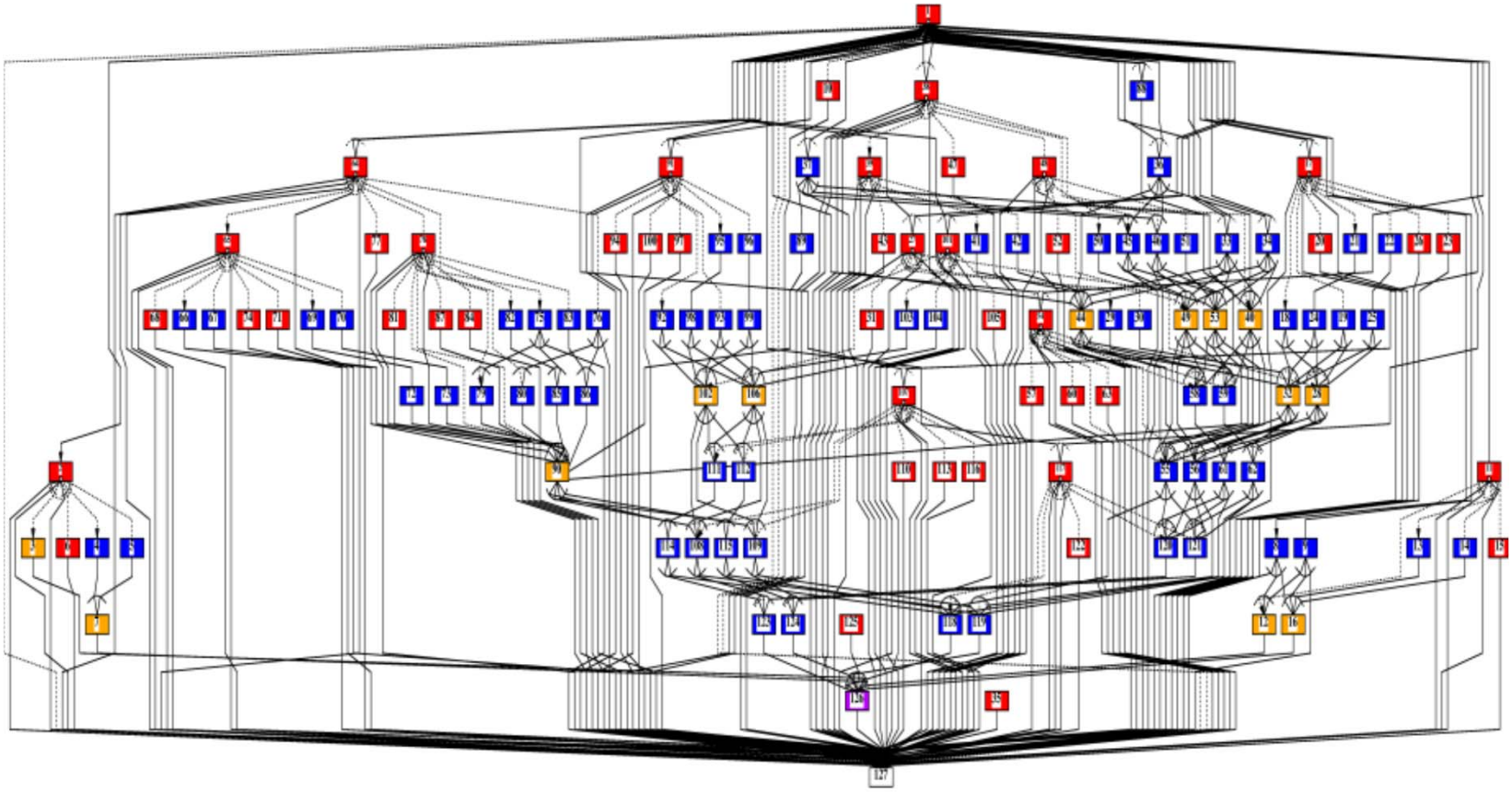
A Macro Flow Graph



A Macro Task Graph

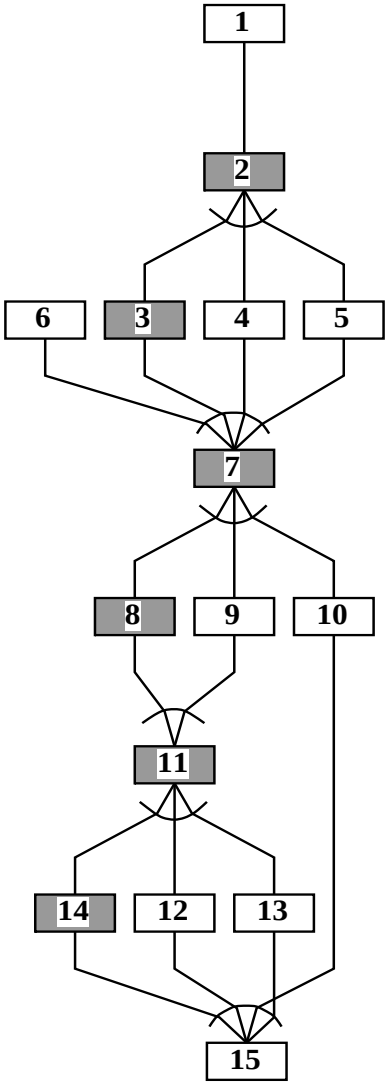
MTG of Su2cor-LOOPS-DO400

- **Coarse grain parallelism PARA_ALD = 4.3**

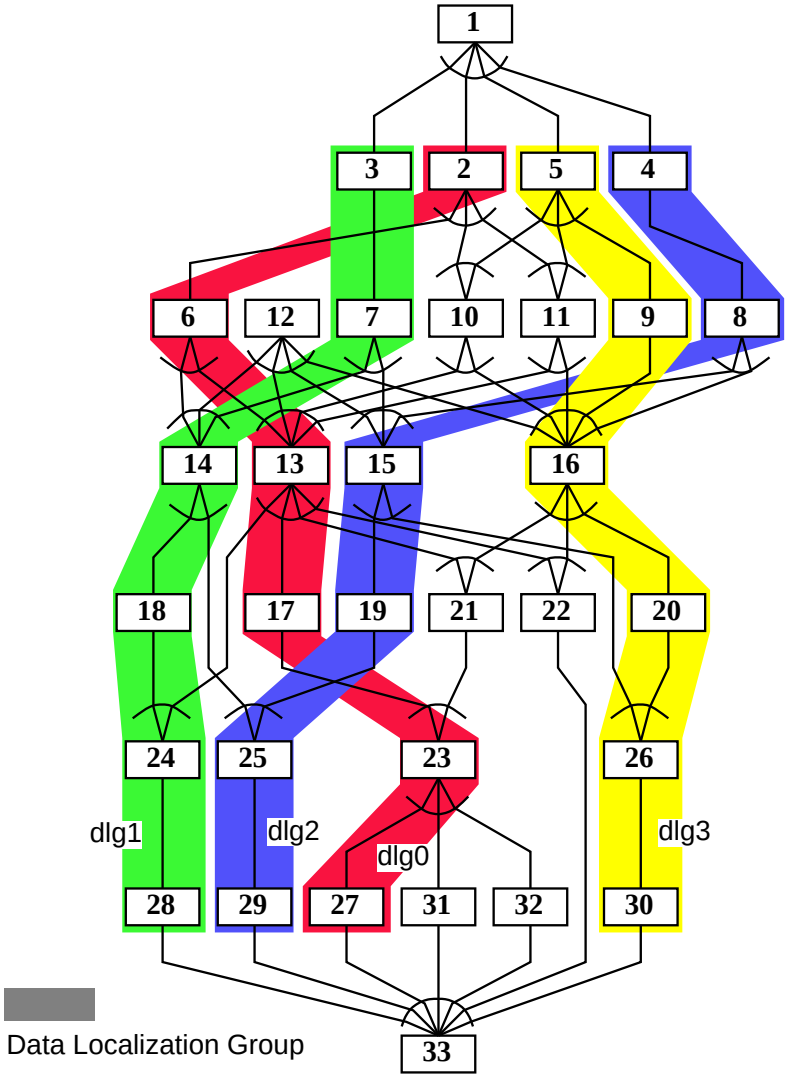


DOALL **Sequential LOOP** **SB** **BB**

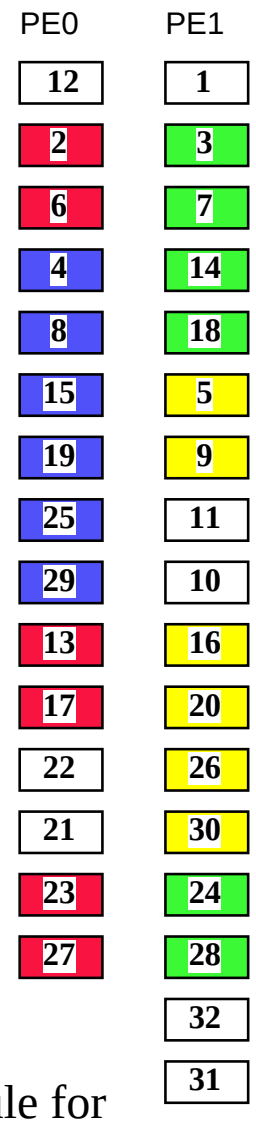
Data Localization



MTG



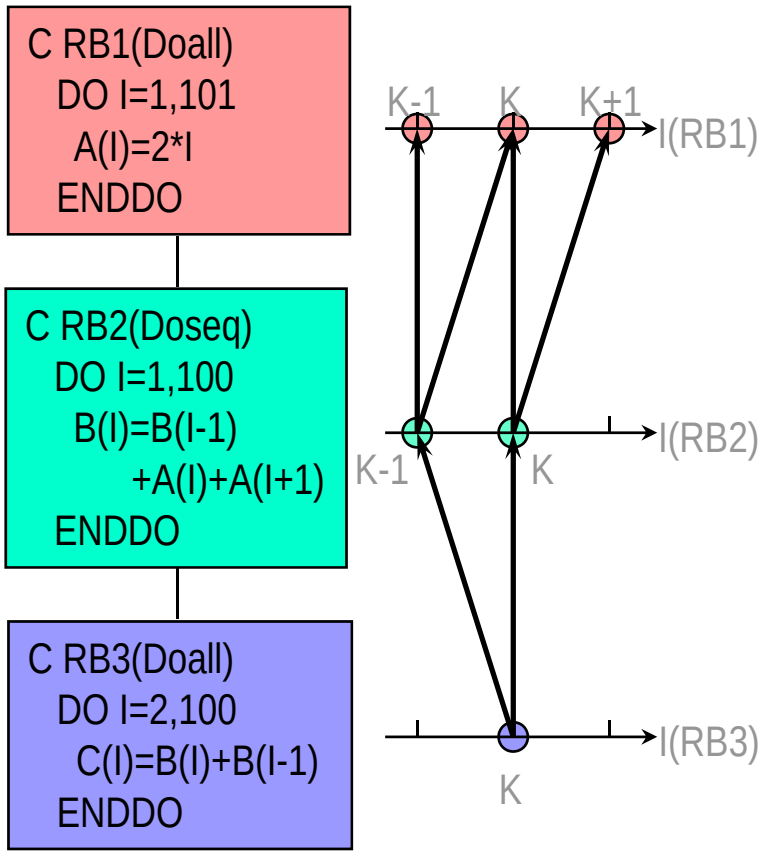
MTG after Division



A schedule for two processors

Inter-loop data dependence analysis in TLG

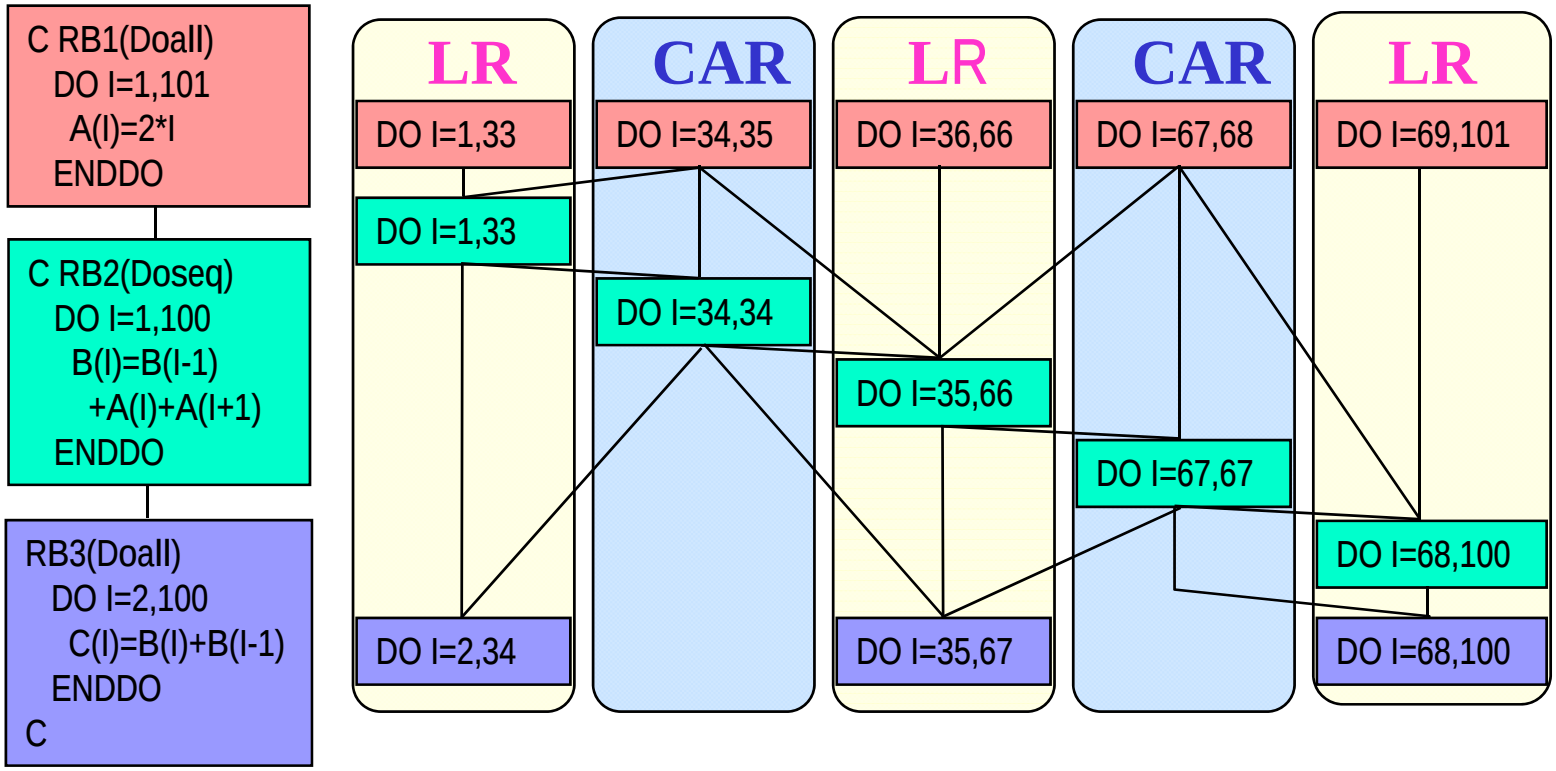
- Define exit-RB in TLG as Standard-Loop
- Find iterations on which a iteration of Standard-Loop is data dependent
  - e.g. K_{th} of RB3 is data-dep on $K\text{-}1_{th}$,K_{th} of RB2, on $K\text{-}1_{th}$,K_{th},$K\text{+}1_{th}$ of RB1



Example of TLG

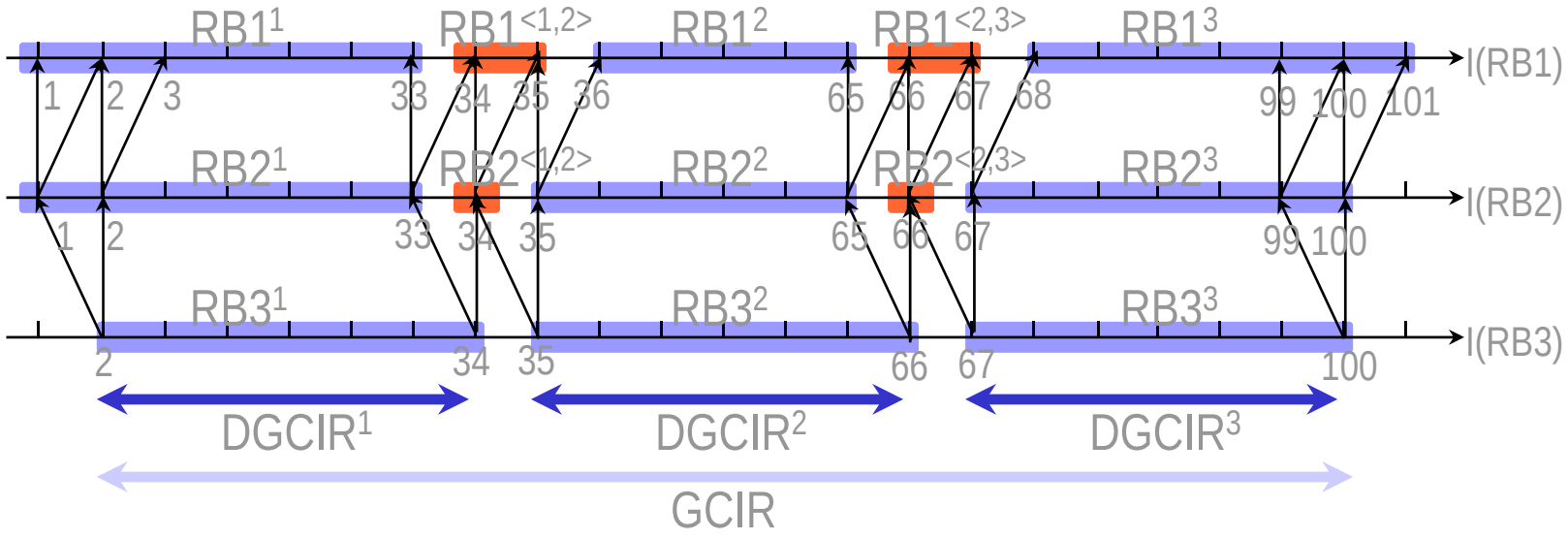
Data-Localization: Loop Aligned Decomposition

- **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**



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.



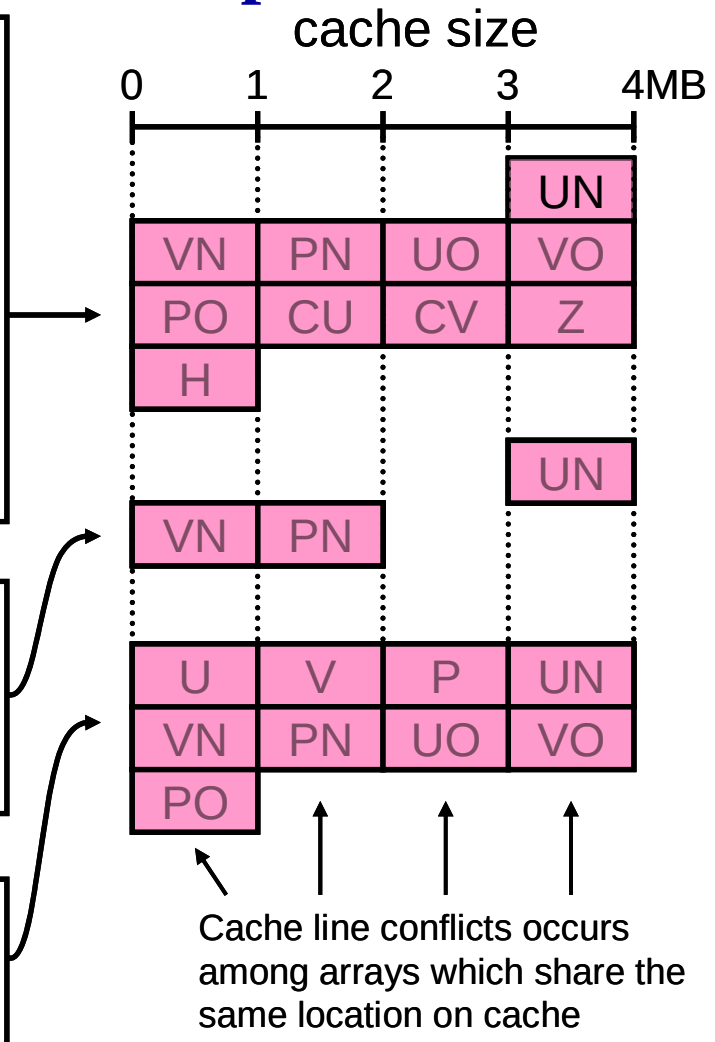
An Example of Data Localization for Spec95 Swim

```
      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

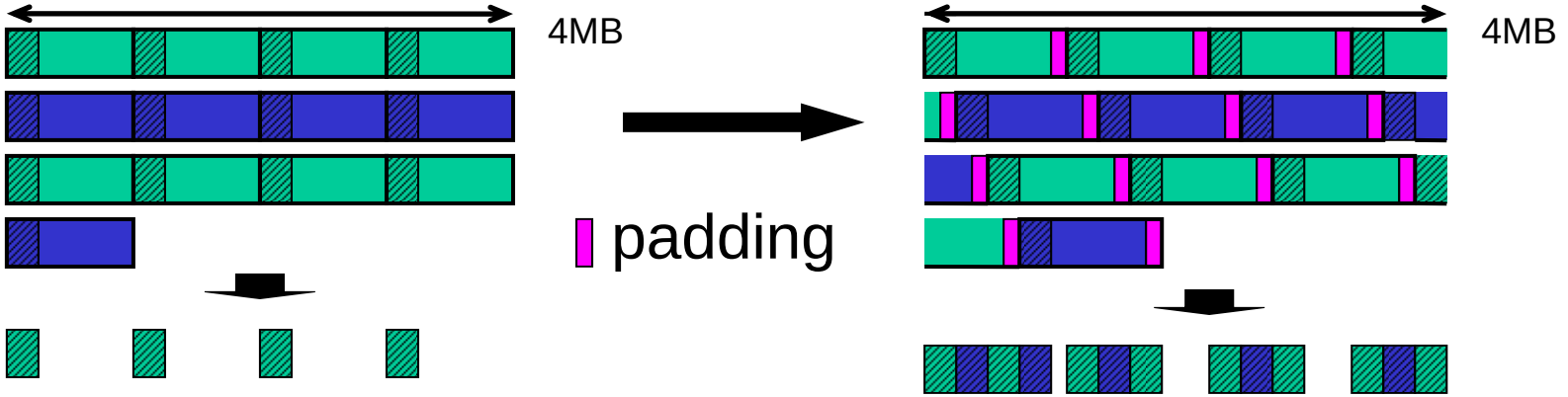
Declaration part of arrays in spec95 swim

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

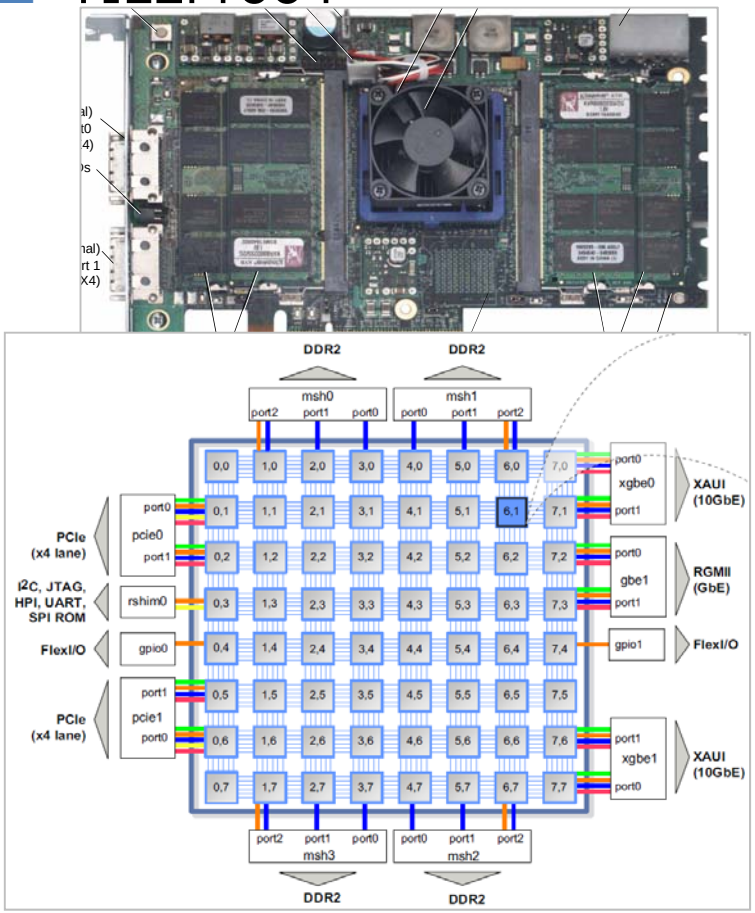
```
PARAMETER (N1=513, N2=544)

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)
```

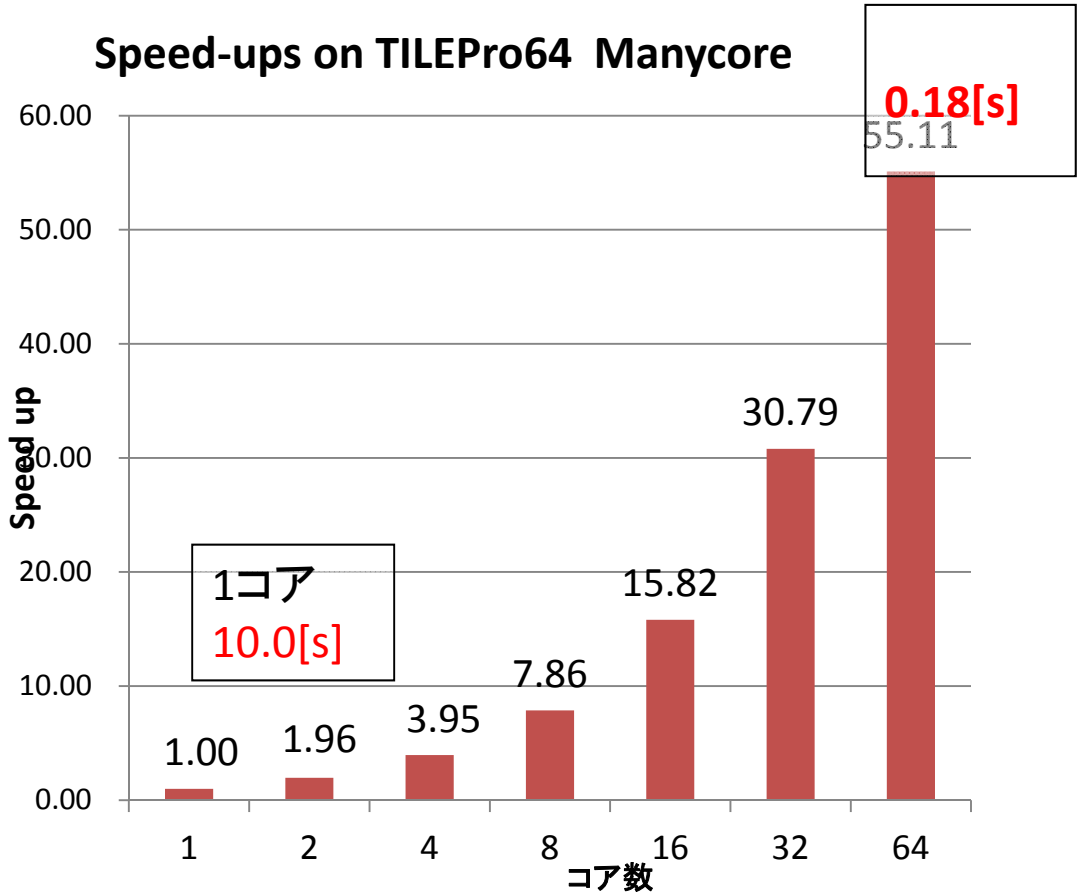


Box: Access range of DLG0

Automatic Parallelization of Still Image Encoding Using JPEG-XR for the Next Generation Cameras and Drinkable Inner Camera

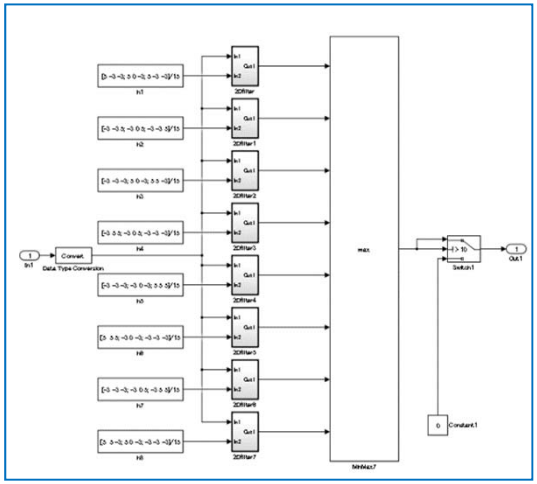
- TILEPro64





55 times speedup with 64 cores against 1 core

OSCAR Compile Flow for Simulink Applications



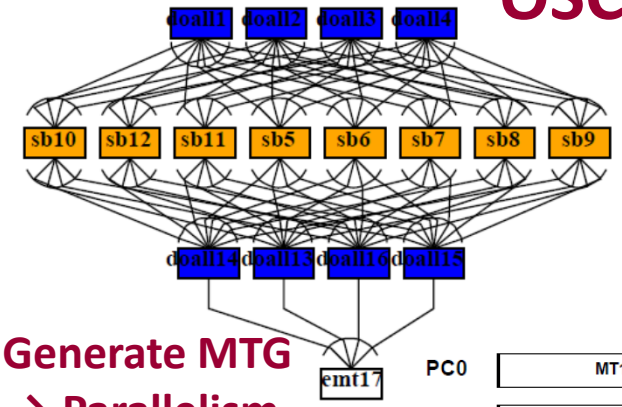
Simulink model

Generate C code
using Embedded Coder

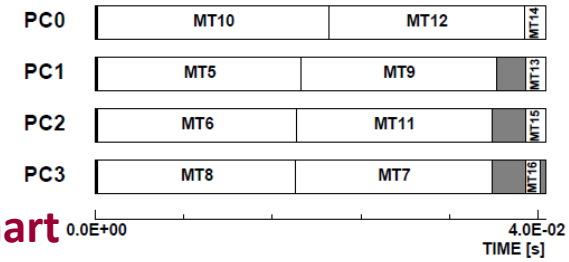
```
/* Model step function */
void VesselExtraction_step(void)
{
  int32_T i;
  real_T u0;

  /* 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



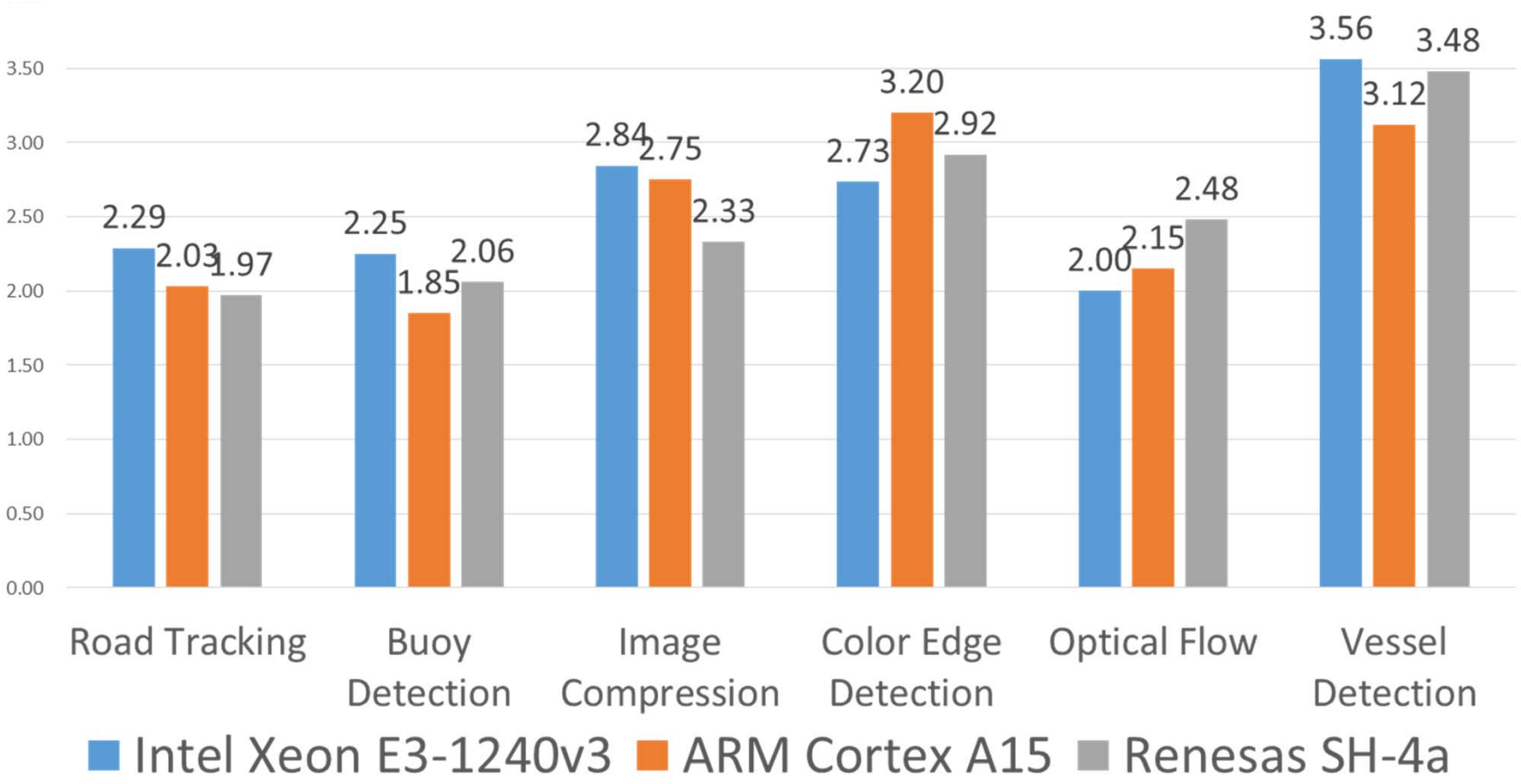
(2) Generate gantt chart
→ Scheduling in a multicore

```
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)

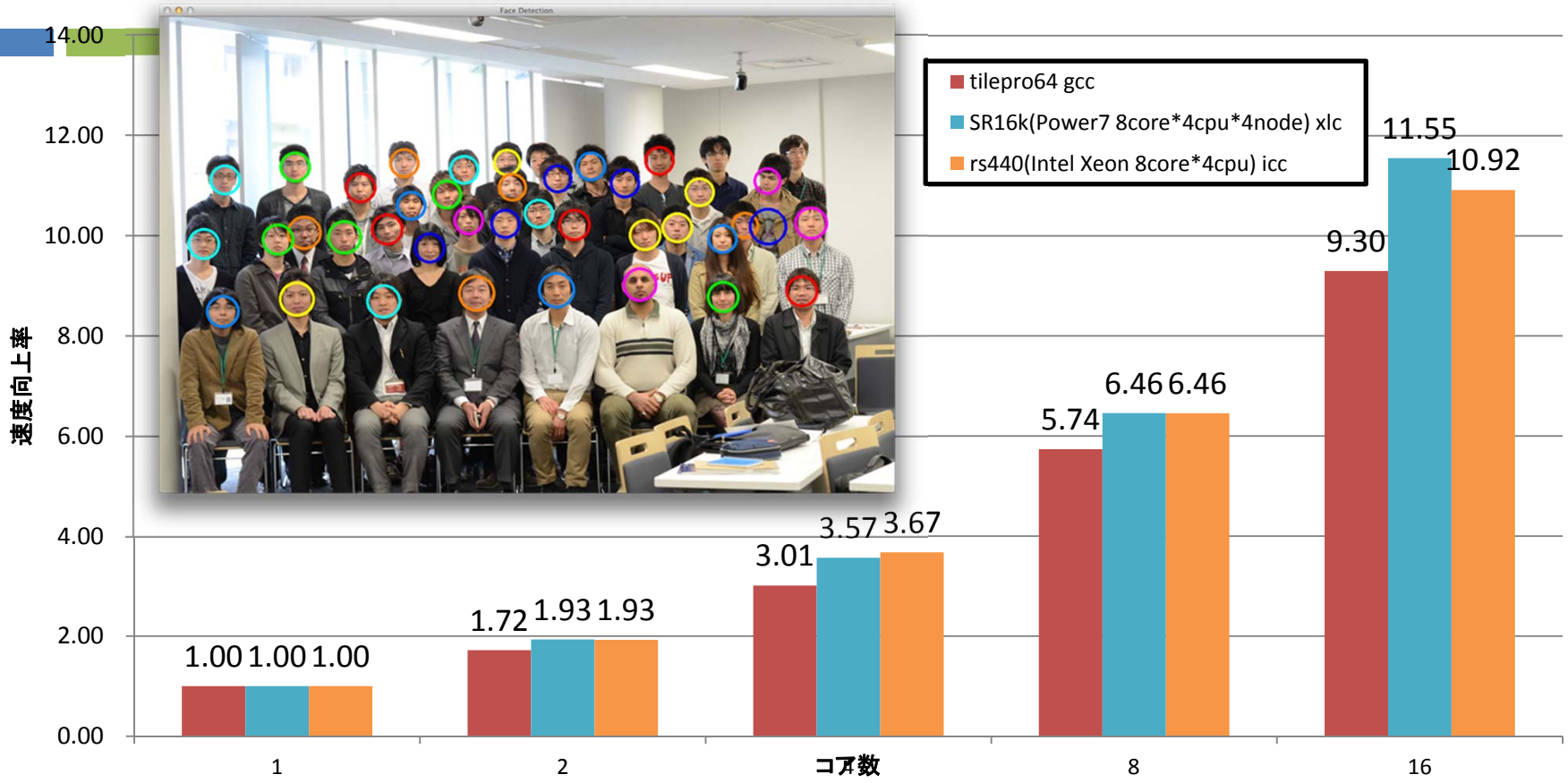
Speedups of MATLAB/Simulink Image Processing on Various 4core Multicores

(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 of Face Detection on Manycore, Highend and PC Server

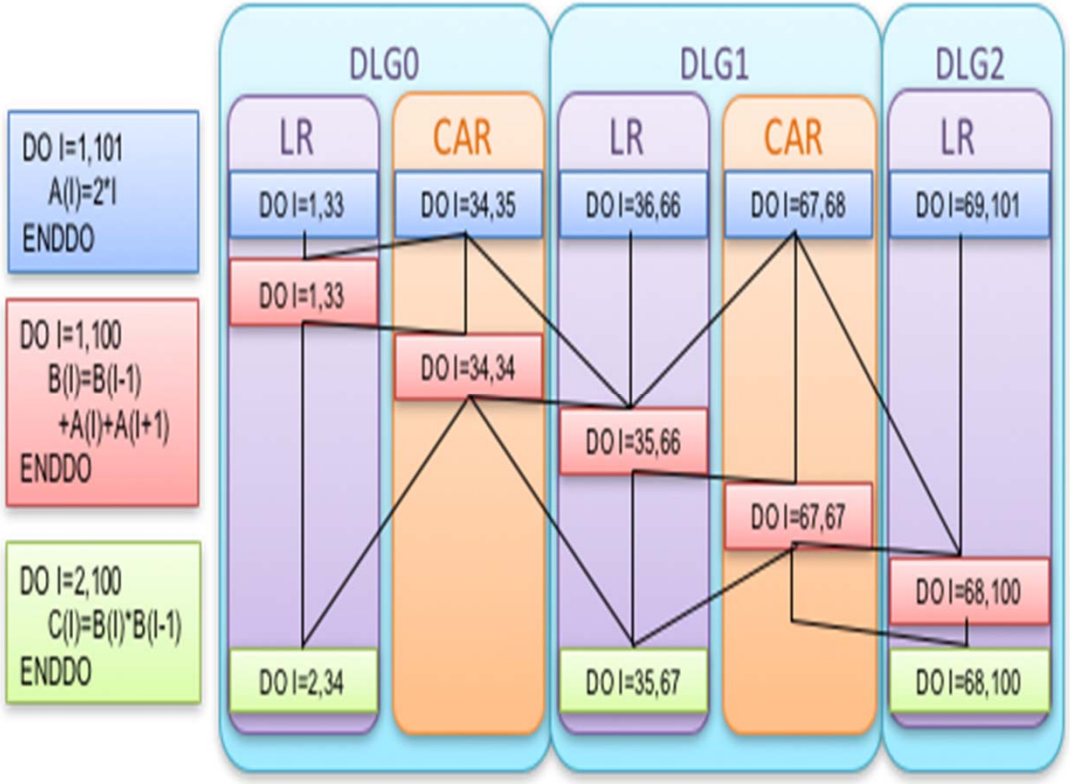


- OSCAR compiler gives us 11.55 times speedup for 16 cores against 1 core on SR16000 Power7 highend server.

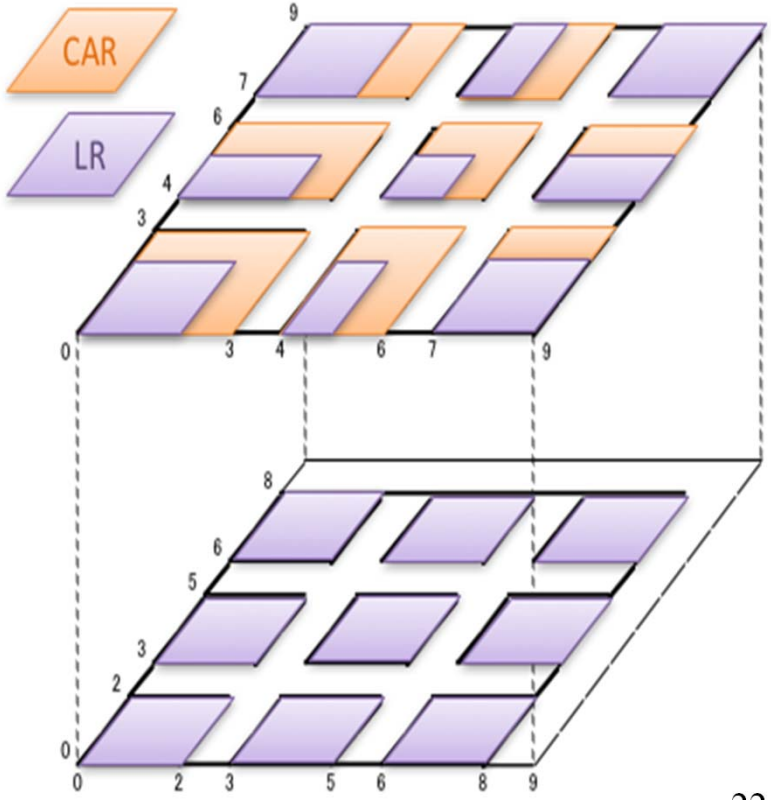
Data Localization: Loop Aligned Decomposition

- **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

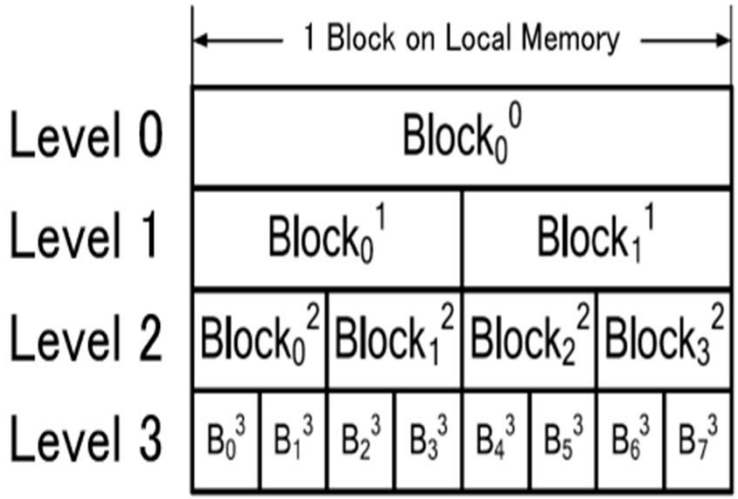


Multi-dimension Decomposition



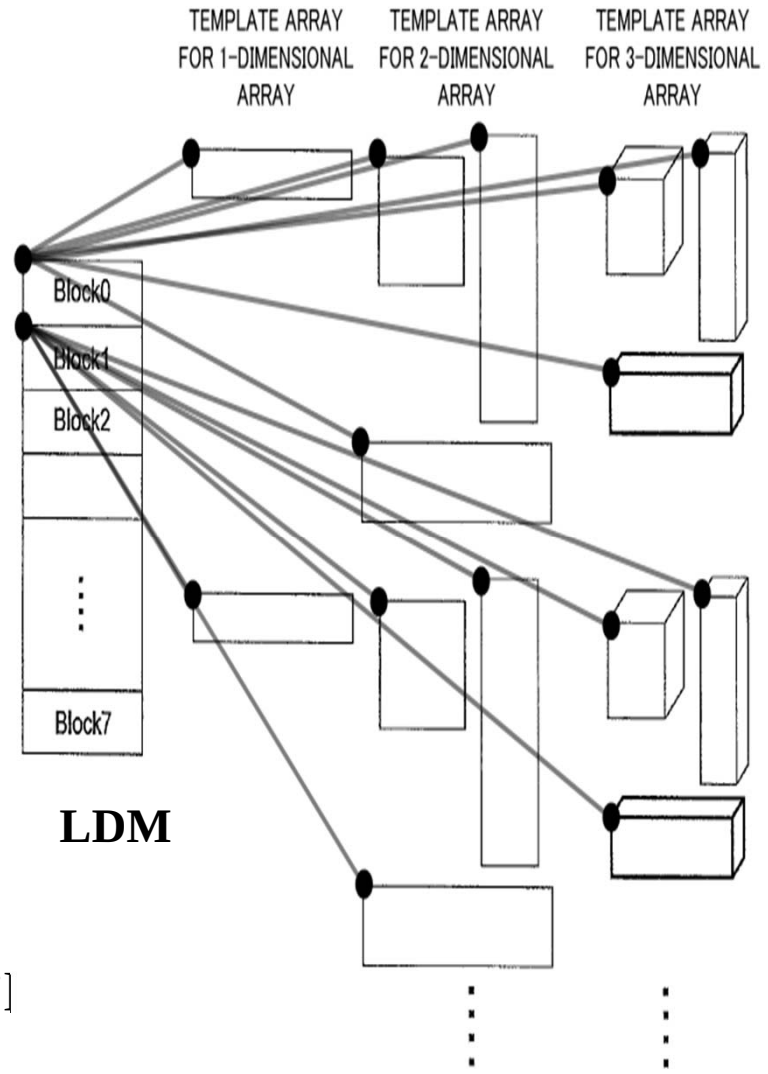
Adjustable Blocks

- Handling a suitable block size for each application
  - different from a fixed block size in cache
  - each block can be divided into smaller blocks with intege $\text{Block}_{\text{Number}}^{\text{Level}}$ 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']



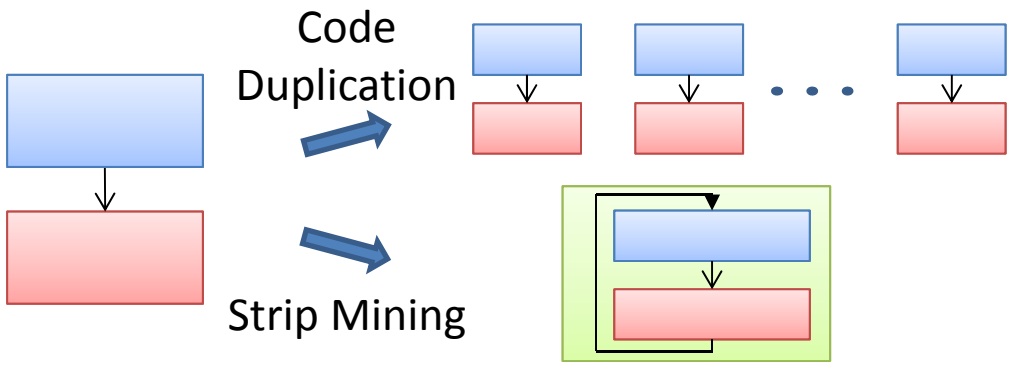
LDM

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

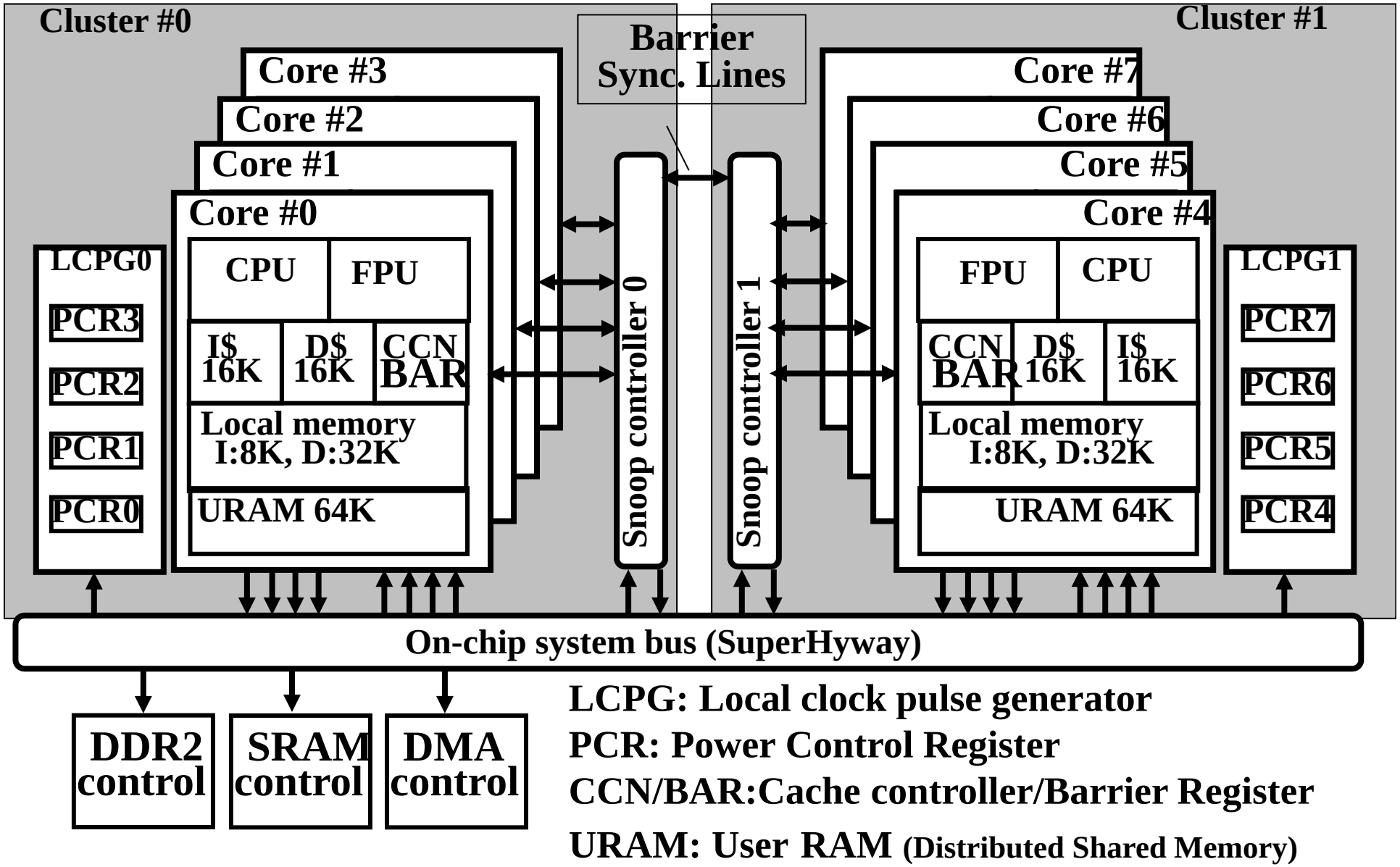



```
for (i = 0; i < 16; i++)
    for (j = 0; j < 64; j++)
        a[i][j] = i + j;

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];
```

8 Core RP2 Chip Block Diagram



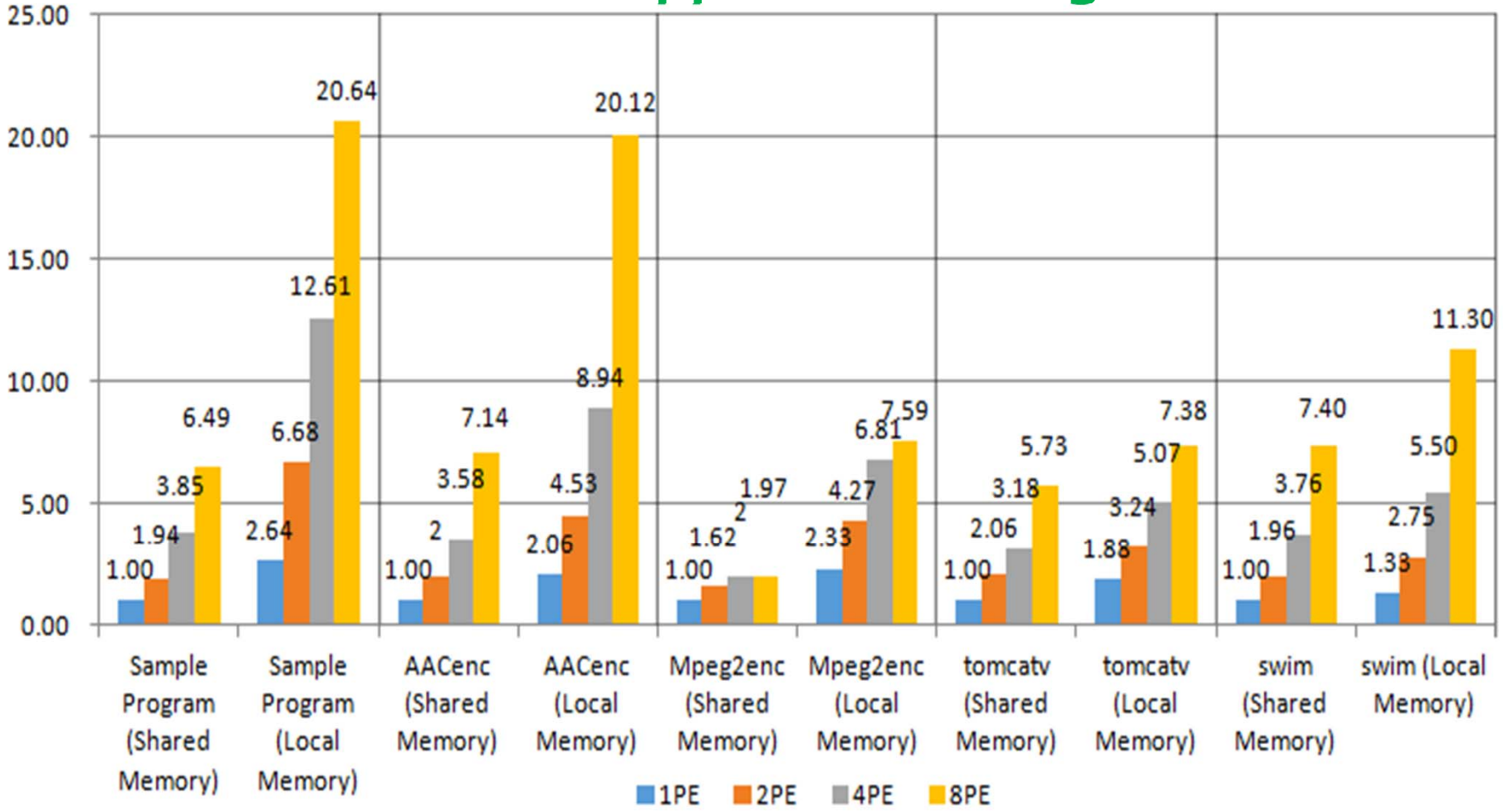
LCPG: Local clock pulse generator

PCR: Power Control Register

CCN/BAR:Cache controller/Barrier Register

URAM: User RAM (Distributed Shared Memory)

Speedups by the 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

Conclusions

- This talk introduced automatic cache and local memory management method using data localization with hierarchical loop aligned decomposition, adjustable block tailored for each application, and block replace considering block reuse distance .
- The local memory management method was implemented on the OSCAR parallelization compiler.
- The performance on the RP2 8 core multicore gave us
- for example,
  - 20.12 times speedup on 8cores using local memory against sequential execution using off-chip shared memory for the AAC encoder though the 8 core execution using shared memory gave us 7.14 times speedup.
  - 11.30 times speedup on 8cores execution using local memory against sequential execution using off-chip shared memory for the SPEC95 swim though the 8 core execution using shared memory gave us 7.40 times speedup.