MBP-light: A Processor for Management of the Distributed Shared Memory on JUMP-1

Inoue Hiroaki† Ken-ichiro Anjo† Jun Tanabe†
Mitsuru Satoh‡ Katsunobu Nisimura† Kei Hiraki∗
Hideharu Amano†

Keio University† Fujitsu Lab.‡ The University of Tokyo∗
E-mail: mbp@am.ics.keio.ac.jp

Introduction(CC-NUMA system)

How to build a scalable Massively Parallel Processor (MPP)

⇓

- **A Distributed Shared Memory (DSM)**
  (A coherent cache is required to reduce the access latency)
- **A coherent cache is required to reduce the access latency of the DSM**
  (A kind of cache coherence is maintained)

⇓

CC–NUMA System
(Cache Coherent–Non Uniform Memory Access model)

⇓

The key factor with regards to system performance is the controller to manage the DSM

Introduction(DSM controllers)

The Stanford DASH adopts a hardwired logic

⇓

- **No flexibility**
- **No simplicity**

⇓

The Stanford FLASH and the Sequent NUMA-Q adopt core processors for the controller

What's JUMP-1 ?

JUMP-1 is an MPP developed by a collaboration of 7 Japanese universities

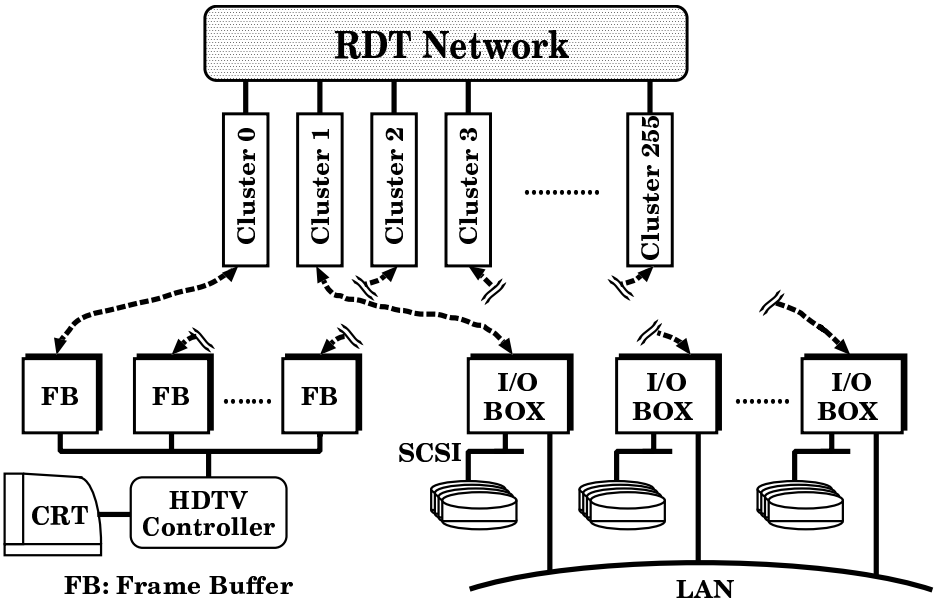
⇓

The major goal is to build an efficient DSM on an MPP with thousands of processors

⇓

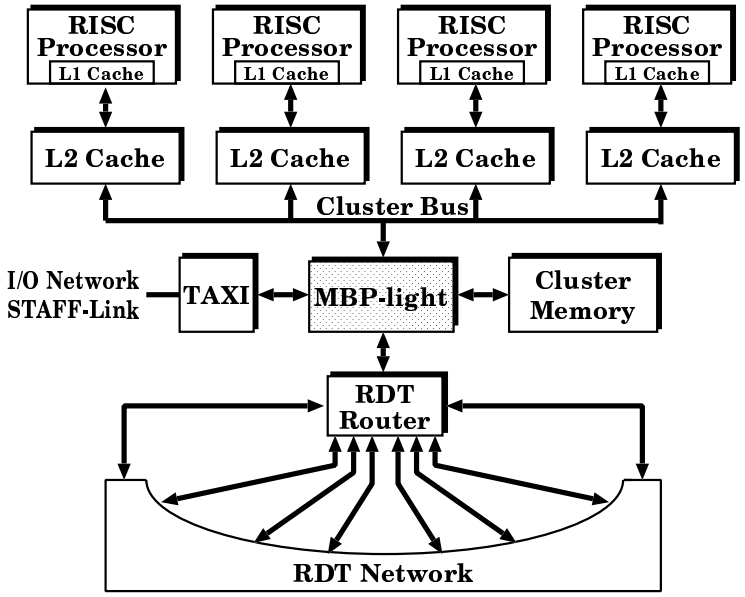
As a dedicated DSM controller MBP(Memory Based Processor)-light has developed

The Structure of JUMP-1



- Network: the RDT (Recursive Diagonal Torus)
- I/O: Fast Serial Link (STAFF-Link)

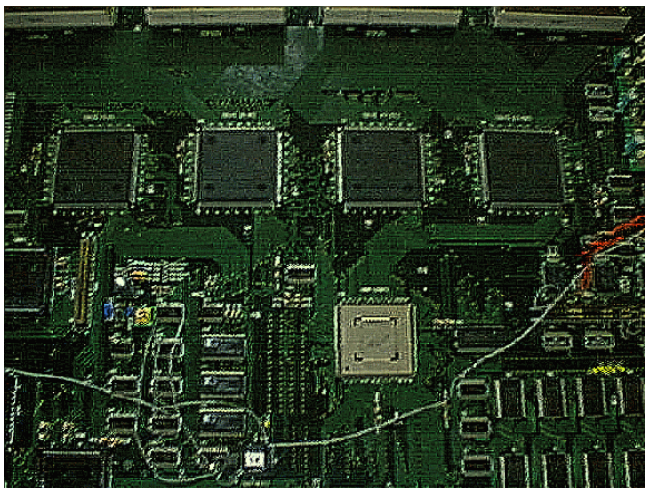
The Structure of JUMP-1 Cluster



- Four SuperSPARC+s with L2 cache
- MBP-light and the RDT router

Photographs of JUMP-1 (1/2)

Photographs of JUMP-1 (2/2)



The DSM of JUMP-1

- **Page level directory**
- **Data transfer in cache line size**
- **Cluster memory can be used as L3 cache**
- **Flexible protocol control including update policy**
- **Directory management based on multicast (RHBD schemes)**

⇓

MBP-light manages the DSM of JUMP-1

The Design Policies of MBP-light

▷ The generation and collection of acknowledgment packets in the RDT network, and the cooperation with L2 caches should be performed quickly

⇓

Custom hardwired logics

▷ The core processor must treat both header flits and data flits

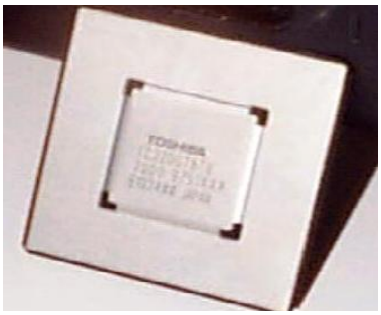
⇓

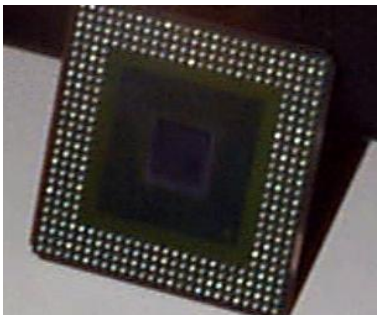
A packet buffer can be accessed as a special register
(*Buffer-Register Architecture*)

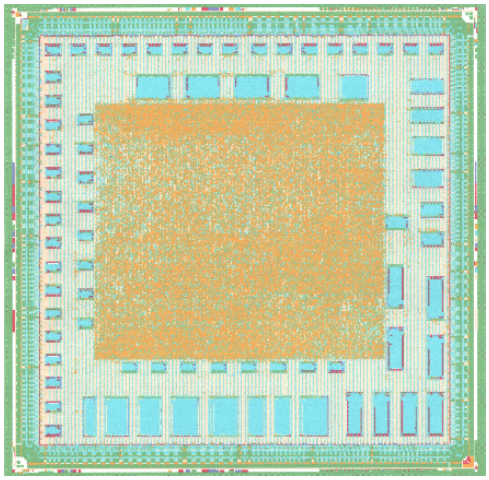
The Implementation of MBP-light

CMOS 3-layer metal embedded array from Toshiba Corp.

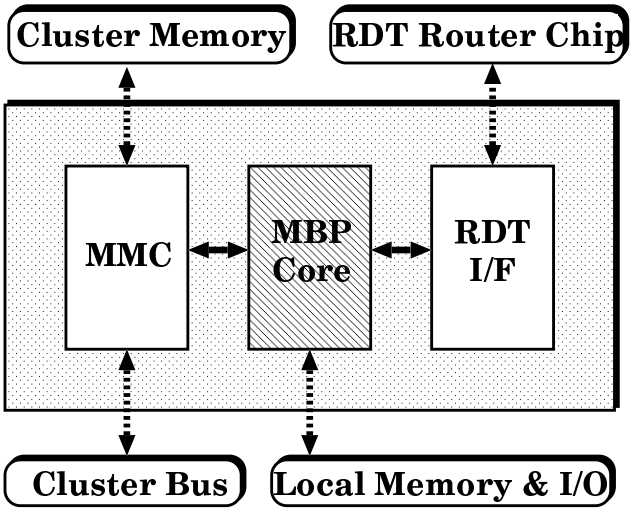
Frequency	**50 MHz**
Process rule	**0.4 μm**
Random logic	**106,905 gates**
Internal memory	**44,848 bits**
Area Utilization	**38.3 %**
The number of pin	**352**
Package	**TBGA**
Power	**3.1 W**





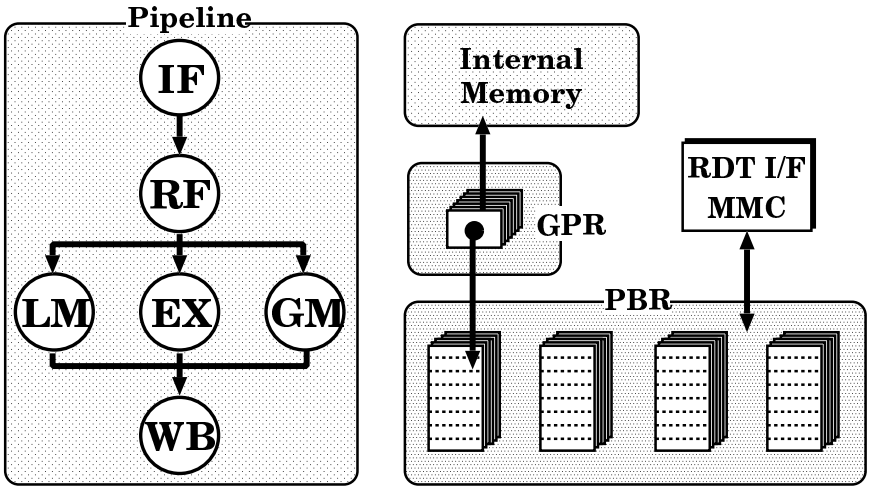


The Structure of MBP-light



- the MMC controls a cluster memory/bus
- the RDT Interface controls the RDT router
- MBP Core controls a complicated protocol

MBP Core



- A pipeline with four stages treating 21bit instructions
- Interrupt functions for protocol processing
- 16 GPRs(General Purpose Register) of 16 bit width
- 112 PBRs(Packet Buffer Register) of 68 bit width

Instruction Classifications

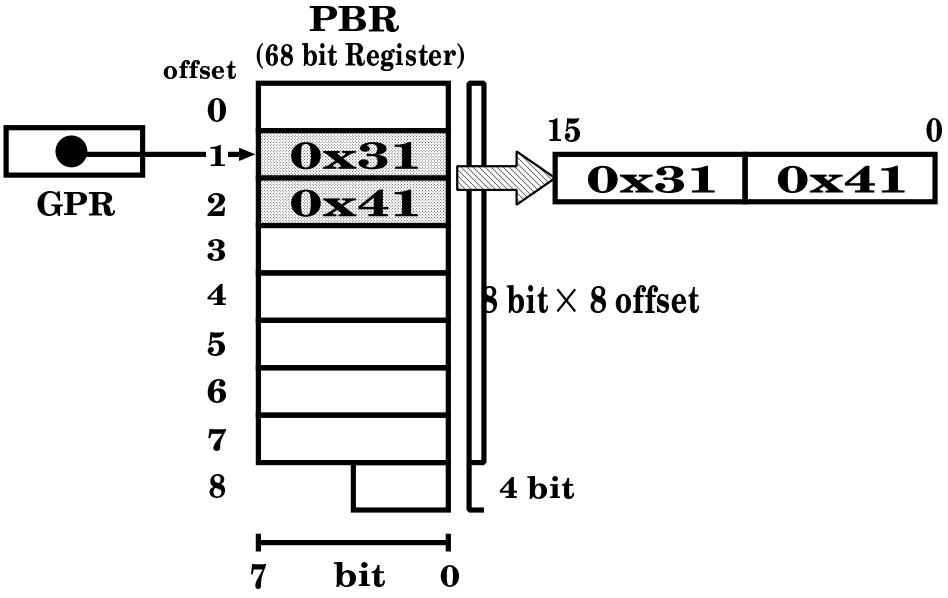
Class	Type
WGG	operate GPR–GPR
Branch	branch
BPI	operate PBR–Imm.
MPP	move PBR–PBR
MMC	control the MMC
INT	control interrupt
SPE	special instructions

Class	Type
LMA	access to local memory
WGI	operate GPR–Imm.
RDT	control the RDT Interface
WPG	operate PBR–GPR
TJ	table jump
IMA	access to internal memory
NOP	no operation

⇓

Totally, there are about 85 instructions with 14 classes

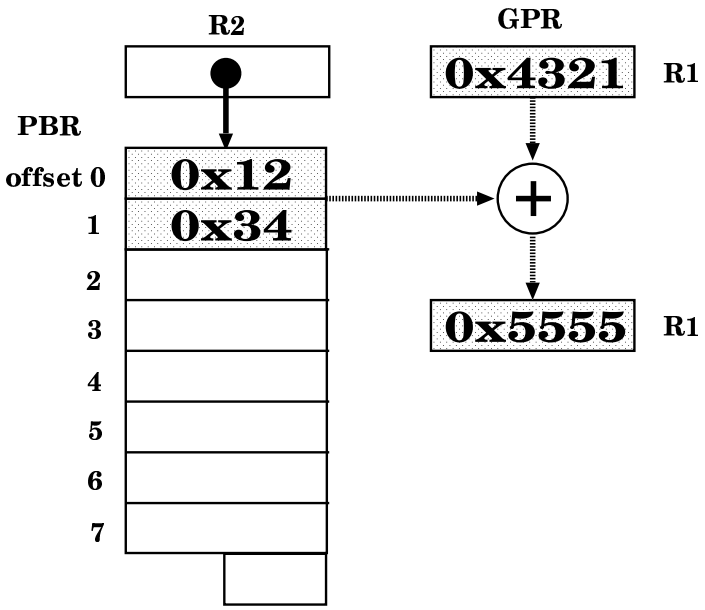
What's PBR (Packet Buffer Register) ?



- **PBR can be used as both a register and packet buffers**
- **PBR can be accessed by a pointer of GPR with 8 offsets**
  (⇒ *Buffer-Register Architecture*)

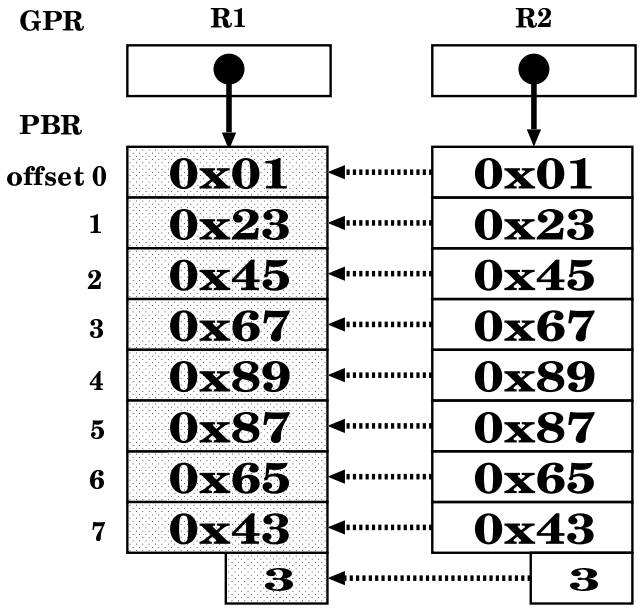
Instuction Examples – GPR-PBR – (1/2)

ADDPG R1, R2(0) /* ADD PBR GPR (WPG)*/

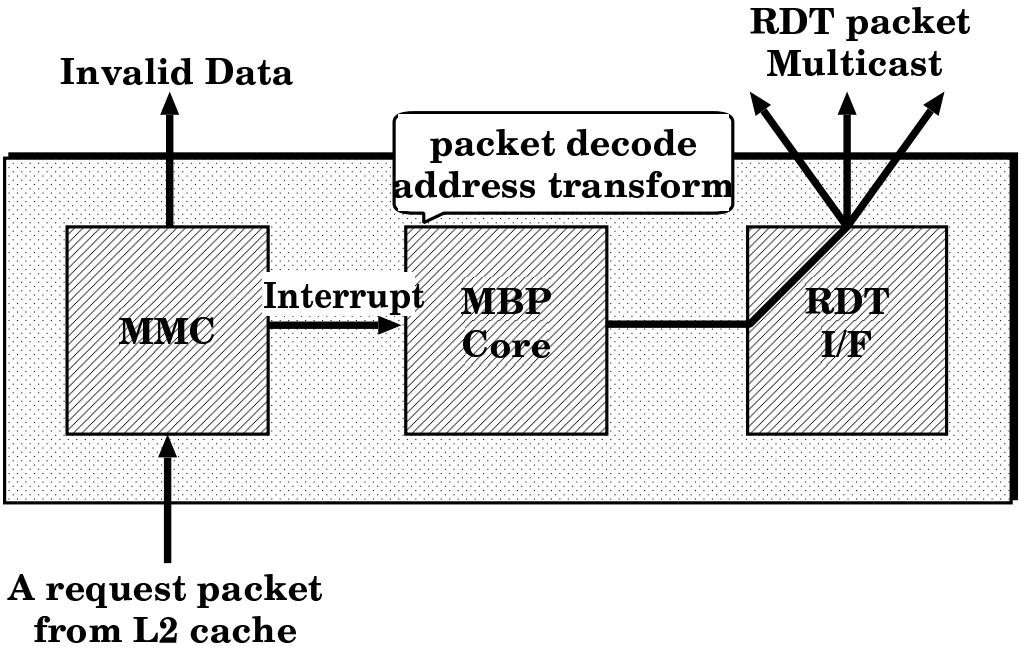


Instruction Examples – PBR-PBR – (2/2)

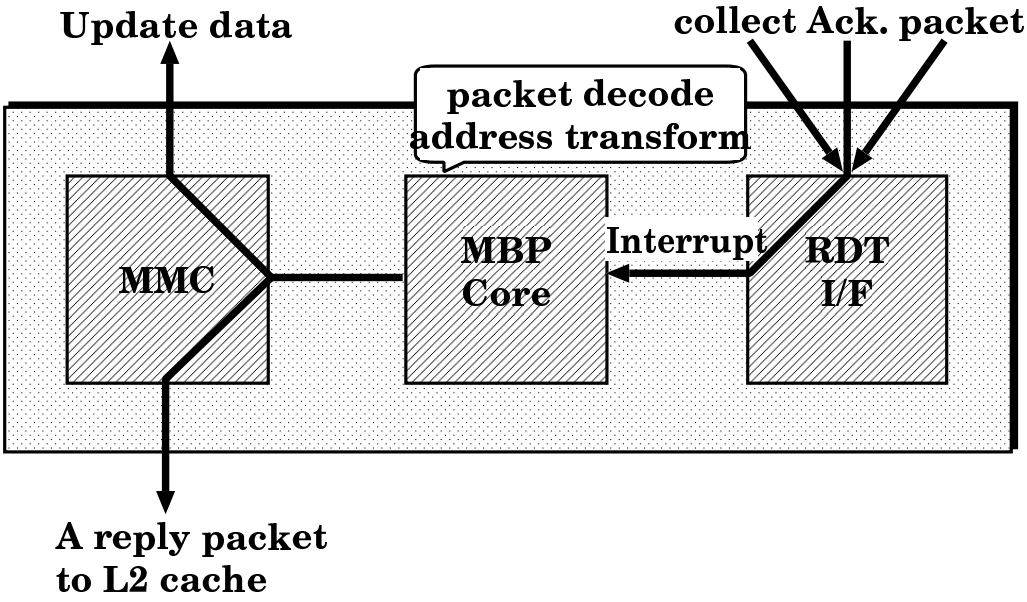
MVLPP R1, R2 /* Move Long PBR-PBR (MPP)*/



Examples of Protocol Processing(1/2)



Examples of Protocol Processing(2/2)



Evaluations

1. **Can MBP-light work efficiently,**
   compared with other CC-NUMAs' DSM controllers ?
   → The protocol processing time and the number of gates

2. **Can MBP Core work efficiently,**
   compared with a general 32bit RISC processor ?
   (*DLX* is an educational typical one)
   → The protocol processing time, the number of gates and the instruction mix

The Comparison with Other Controllers (1/2)

- **MBP-light on JUMP-1**

Block	Gates	Memory
MBP Core	35,734	6,144
the RDT i/f	19,832	38,704
the MMC	43,203	0
Total	98,769	44,848

- **The controllers on NUMA-Q**

Chip	Gates	Memory
SCLIC	140,000	152,000
OBIC	170,000	8,700
Total	310,000	160,700

The hardware amount of MBP-light is reduced more than that of the controllers on NUMA-Q

The Comparison with Other Controllers (2/2)

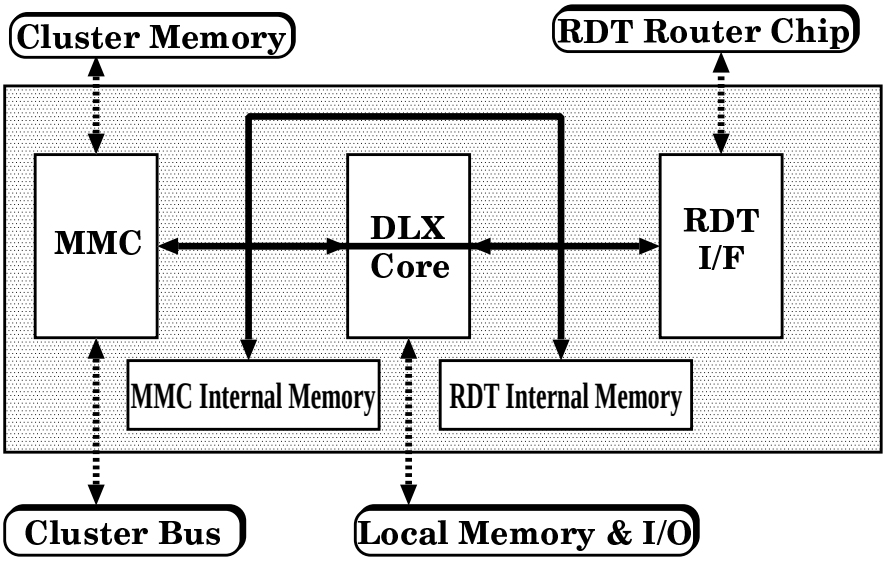
Request	Line State (Home CL)	JUMP-1	NUMA-Q
L3 cache hit	–	760 ns	800 ns
Read miss	Valid	6.9 μs	4-10 μs
Read miss	Invalid	15.3 μs	–

Protocol processing time of both system is almost same

$\Downarrow$

As the system size is larger, the difference is more noticeable

DLX as a Core Processor



DLX is an educational typical 32bit RISC processor

The Comparison with DLX

Protocol Type	Home	MBP (μs)	DLX (μs)
Read Miss	Valid	5.9	7.0
Read Miss	Invalid	14.0	16.7
Invalidation	Valid	11.0	13.4

Processor	Critical Path (ns)	Gates
MBP Core	12.7	22,973
DLX	13.3	28,612

Protocol processing time and the hardware amount of MBP Core are reduced more than those of DLX

Why is MBP Core better than the RISC processor?

In general, the 32bit RISC processor is better than the 16bit one

⇓

MBP Core has the buffer-register architecture

⇓

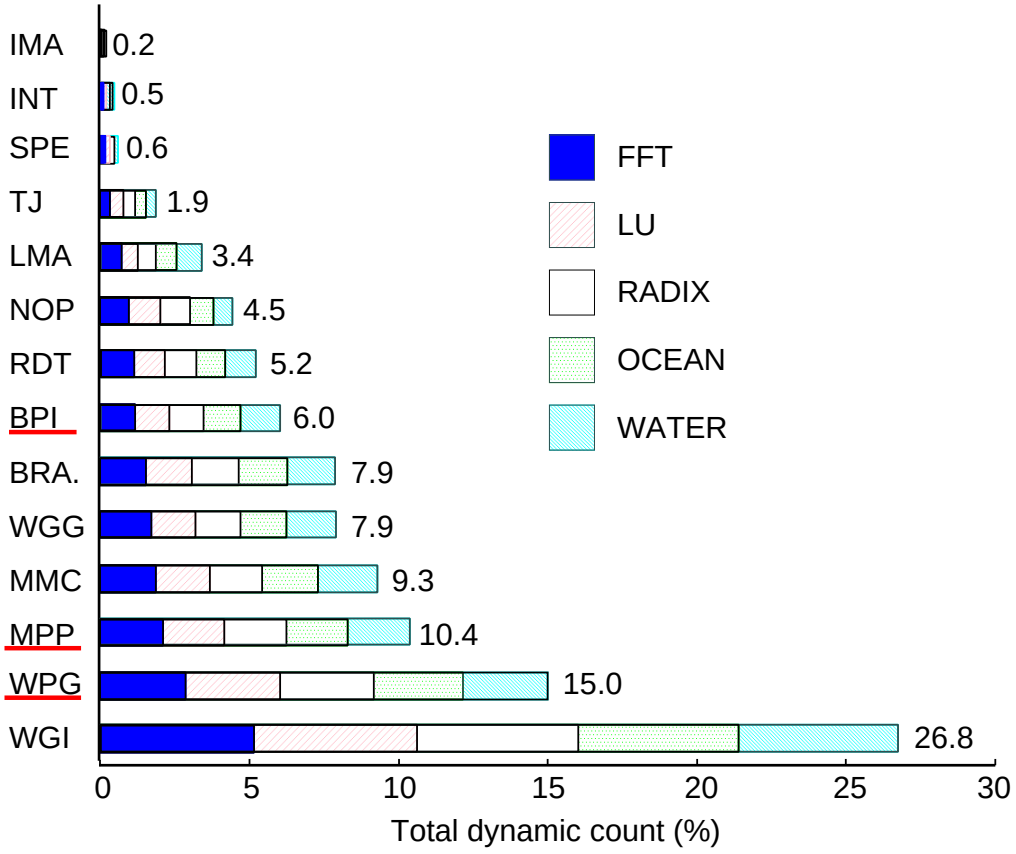
It can reduce

- **the load/store overhead to packet buffers**
- **the total hardware compared with the 32-bit RISC**

⇓

The utilization of the buffer-register architecture is the key issue

The Instruction Mix of MBP-light



Conclusions

- MBP-light which manages the DSM of JUMP-1 is designed and implemented
- MBP-light can work more efficiently to process the protocol than other CC-NUMAs' DSM controllers
- Buffer-Register Architecture adopted by MBP-light can work more efficiently than a general 32bit RISC processor

⇓

Now, JUMP-1 with 16 processors (4 clusters) is under debugging