14th International Forum on Embedded MPSoC and Multicore

MPSoC 2014

NanoProcessor Cluster: Conversion from ILP to µTLP

July 10th, 2014

Fumio Arakawa

Nagoya Univ.

Outline

I. INTRODUCTION

II. NANOPROCESSOR CLUSTER

III. EVALUATION AND CONCLUSION

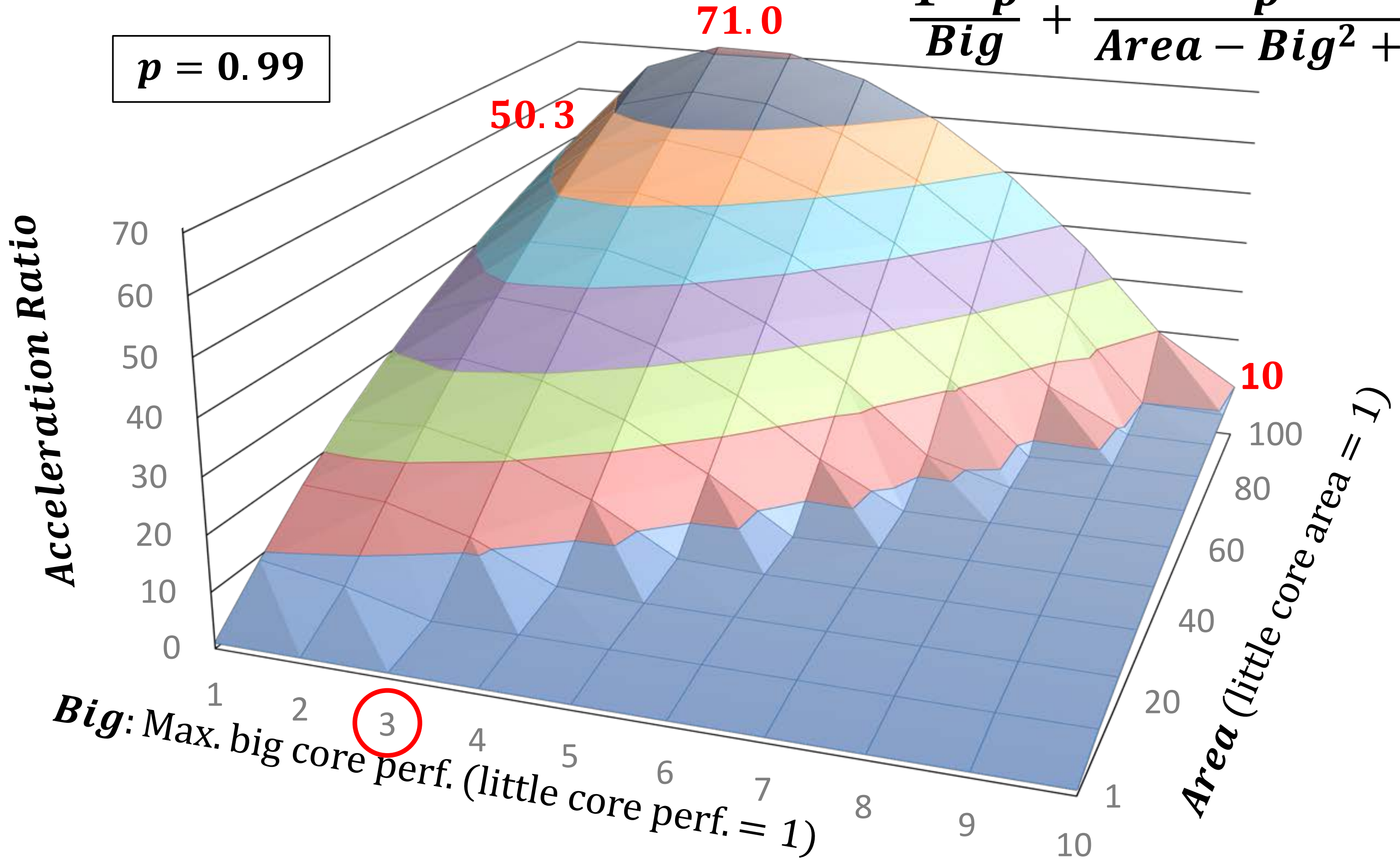
Amdahl’s Law & Pollack’s Law

- **Amdahl’s Law:** $\boldsymbol{Acceleration\ Ratio\ (\alpha)} = \dfrac{\mathbf{1}}{\dfrac{\mathbf{1}-\boldsymbol{p}}{\boldsymbol{Big}} + \dfrac{\boldsymbol{p}}{\boldsymbol{Area} - \boldsymbol{Big}^2 + \mathbf{1}}}$
  - $\boldsymbol{Area}$: total area of the cores
  - $\boldsymbol{p}$: parallelization ratio
  - $\boldsymbol{Big}$: big core perf., where little core perf. $= 1$, little core area $= 1$
- **Pollack’s Law:** $\boldsymbol{Big} = \sqrt{\boldsymbol{big\ core\ area}}$
  - We are on the wrong side of a square law.
  - Too much **ILP extraction** causes inefficiencies of the **Big** cores.
- **Heterogeneous** multi-many-core processor
  - high performance for a **fairly parallel** program
  - an inefficient, but high performance, single core
  - highly efficient, but low performance, multi-many cores
- $\boldsymbol{Area} - \boldsymbol{Big}^2 + \mathbf{1}$: number of cores in the "**Area**" with one "**big**" core

$Highly\ Parallel\ Case\ (p = 0.99)$

- **Pollack's Law:** $Big = \sqrt{big\ core\ area}$
- **Amdahl's Law:** $Acceleration\ Ratio = \dfrac{1}{\dfrac{1-p}{Big} + \dfrac{p}{Area - Big^2 + 1}}$

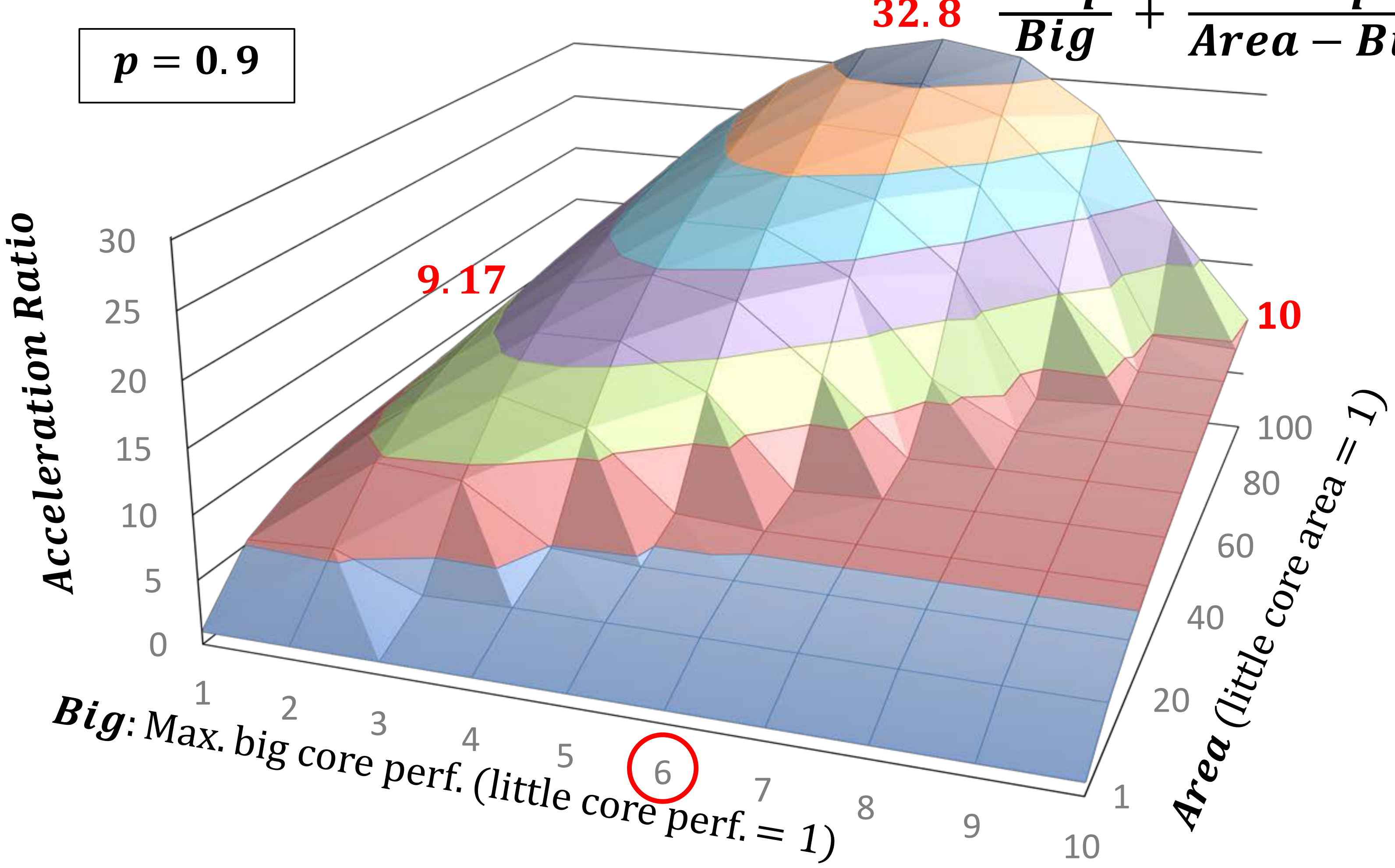
$\boxed{p = 0.99}$



$\boldsymbol{Fairly\ Parallel\ Case\ (p = 0.9)}$

- **Pollack's Law:** $\boldsymbol{Big = \sqrt{big\ core\ area}}$
- **Amdahl's Law:** $\boldsymbol{Acceleration\ Ratio = \dfrac{1}{\dfrac{1-p}{Big} + \dfrac{p}{Area - Big^2 + 1}}}$

$\boxed{p = 0.9}$



Conversion from ILP to μTLP

- Single core utilizes ILP (Instruction Level Parallelism).
  - Pipelined Architecture: Parallelism = # of the pipeline stages
  - Superscalar Architecture: Parallelism = # of issue slots
  - Good for tightly-coupled operations, Bad for long-latency operations
- Conversion from ILP to μTLP (μ-Thread Level Parallelism)
  - μ-Thread: a divided flow of original single flow

    Flows are tightly coupled.

    Each flow requires subset of conventional CPU functions.
- Reorganizing microprocessor core to "NanoProcessor Cluster"
  - NanoProcessor: Subset of Conventional CPU Functions

    Execute a μ-Thread

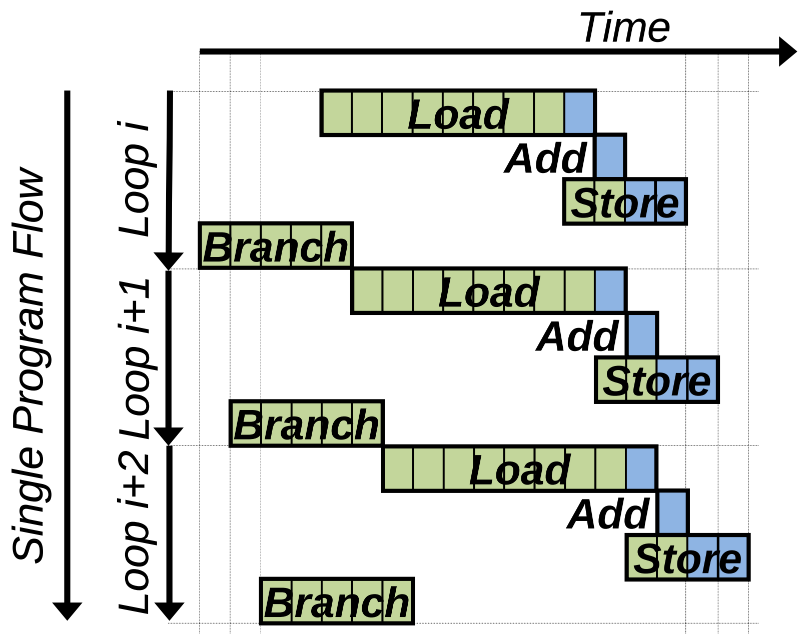
    Scalar short-pipeline processor: for limited ILP
  - Its Cluster: equivalent to microprocessor function

    Good for both tightly-coupled and long-latency operations

    highly efficient and high performance

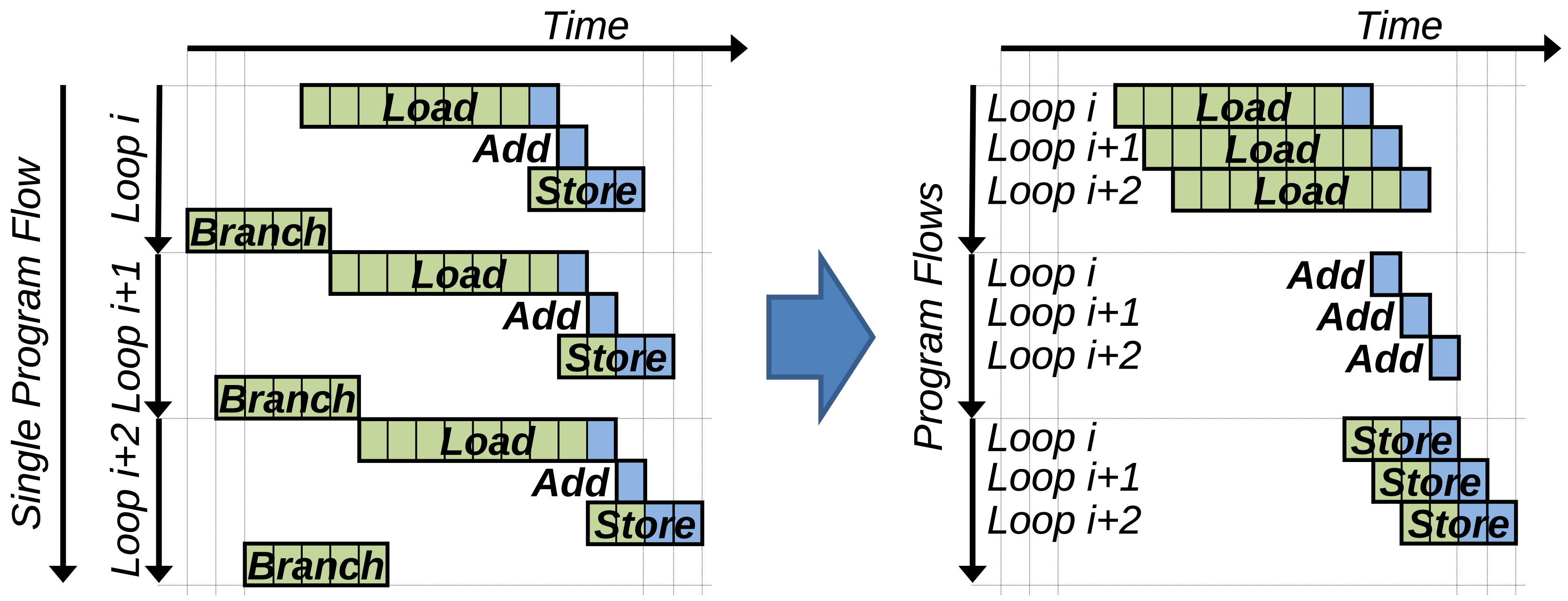
Out-of-Order Microprocessor

- Conventional out-of-order issue microprocessor
  - Mixed functions, various execution latencies
- order of instructions ≠ ideal order of processor executions
  - Limited number of logical registers visible to software
  - Reordering with large physical register file
  - Long time from assign to write: Most of registers are idle.

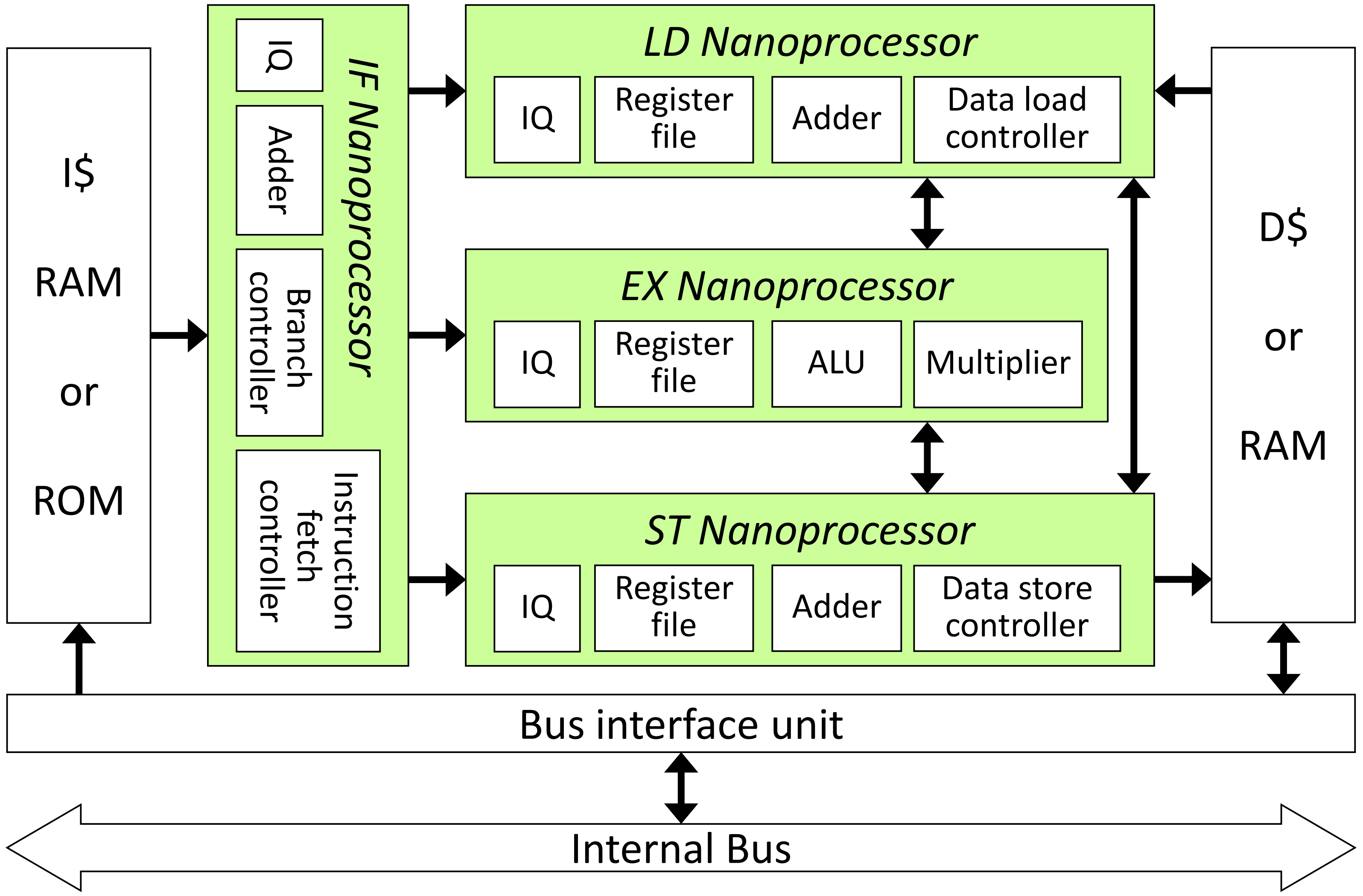


NanoProcessor Cluster

- NanoProcessor:
  - Execute its own μ-thread flow
  - Limited similar functions are enough for each NanoProcessor.
  - the same or similar latencies
  - Synchronization with Register Validity (Architectural State)
    - Write : register becomes valid
    - Last Read (New) : register becomes invalid
    - Flow change : Validate or Invalidate if the validity is changed.



Block Diagram



Sample Program

- Simple String Compare (Max. 8 Characters)

```
    IF NanoProcessor          LD NanoProcessor          EX NanoProcessor

    (caller)
         (next_PC = IF1)
1        IQ/IF 5,LD1                MOVI    R0,#ptr0
2                                   MOVI    R1,#ptr1
    (callee)
1   IF1: IQL/LD 2,EX1         LD1: LBU R0/EX,(R0)++     EX1: MOVI    R2   ,#0
2        LP/LD LD1,LD2,8      LD2: LBU R1/EX,(R1)++     EX2: SEQ C0,R2/K,R0/K
3        IQR/EX 6,RA                                    EX3: SEQ C1,R1   ,R0
4        LP/EX EX2,EX3,8                                EX4: REGV    R1   ,R0
5        BQ/EX C0.p,EX2,EX5                             EX5: SUB     R1   ,R0
6        BQ/EX C1.n,EX3,EX4                                  REGI    R2
```

Instruction Examples

Instruction	Operation
IQ/IF 5,LD1	Load 5 insts. from next_PC to the IQ of IF NanoProcessor, and update the next_PC to point LD1.
`IQL/LD 2,EX1`	Load 2 insts. from next_PC to the IQ of LD NanoProcessor, link return address to RA, and update the next_PC to point EX1.
`IQR/EX 6,RA`	Load 6 insts. from next_PC to the IQ of EX NanoProcessor, and copy RA value to the next_PC.
`LP/LD LD1,LD2,8`	Loop setting from LD1 to LD2 for 8 times.
`BQ/EX C0.p,EX2,EX5`	Branch setting from EX2 to EX5 of EX NanoProcessor if C0 == 1.
`BQ/EX C1.n,EX3,EX4`	Branch setting from EX3 to EX4 of EX NanoProcessor if ~C1 == 1.
`MOVI R0,#ptr0`	Copy an immediate value ptr0 to register R0.
`LBU R0/EX,(R0)++`	Load 1-byte unsigned value from the memory pointed by register R0 to register R0 of EX NanoProcessor, and add 1 to R0.
`SEQ C0,R2/K,R0/K`	Set a comparison result of R2 and R0 to C0, and keep the values of R2 and R0.
REGV R0,R1	Validate registers R0 and R1.
REGI R2	Invalidate register R2.

Sample Program

	IF NanoProcessor	LD NanoProcessor	EX NanoProcessor
1	IF1: IQL/LD 2,EX1		
2	LP/LD LD1,LD2,8		
3	IQR/EX 6,RA		
4	LP/EX EX2,EX3,8	LD1: LBU R0/EX,(R0)++	
5	BQ/EX C0.p,EX2,EX5	LD2: LBU R1/EX,(R1)++	
6	BQ/EX C1.n,EX3,EX4	LD1: LBU R0/EX,(R0)++	EX1: MOVI R2 ,#0
7		LD2: LBU R1/EX,(R1)++	EX2: SEQ C0,R2/K,R0/K
8		LD1: LBU R0/EX,(R0)++	EX3: SEQ C1,R1 ,R0
9		LD2: LBU R1/EX,(R1)++	EX2: SEQ C0,R2/K,R0/K
10		LD1: LBU R0/EX,(R0)++	EX3: SEQ C1,R1 ,R0
11		LD2: LBU R1/EX,(R1)++	EX2: SEQ C0,R2/K,R0/K
12		LD1: LBU R0/EX,(R0)++	EX3: SEQ C1,R1 ,R0
13		LD2: LBU R1/EX,(R1)++	
14			
15			EX4: REGV R1 ,R0
16			EX5: SUB R1 ,R0
17			REGI R2

- Mismatch at the third character
- IF & LD latencies are 3 cycles.
- Two-cycle stall for branch prediction miss
- Discard fetched value after loop exit

Evaluation with Dhrystone 2.1

- NanoProcessor architecture is effective to hide long latencies.

	SH-X		NanoProcessor Cluster			
			w/o speculative load		w/ speculative load	
Load Latency	cycles /loop	MIPS /MHz	cycles /loop	MIPS /MHz	cycles /loop	MIPS /MHz
3	282	2.02	268	2.12	225	2.53
5	337	1.69	313	1.82	249	2.29
9	509	1.12	413	1.38	302	1.89

- SH-X: embedded processor, dual issue eight-stage pipeline

#) Dhrystone performance is highly dependent on compiling option, and only relative performances are important.

NanoProcessor Cluster

Conversion from ILP to µTLP

- Good for both tightly-coupled and long-latency operations
- Highly-efficient and High performance
- Each NanoProcessor is simple and independent as a processor.
- NanoProcessors are linked by simple interface.
- Simple: Easy to design, implement, evaluate, and debug

Future Work

- Search for suitable applications
  - Embedded, Networking, HPC, ...
- More Tools, Evaluations, and Collaborations
  - Compiler, Simulator, benchmarks, ...