

A Compiler-Directed Energy Saving Strategy for Parallelizing Applications in On-Chip Multiprocessors

Juan Chen, Yong Dong, Xue-jun Yang and Dan Wu

School of Computer Science

National University of Defense Technology, China

juanchen@nudt.edu.cn

Outline

- **Introduction**
- **On-Chip Multiprocessor**
- **Energy Saving Opportunities of Parallelizing Applications**
- **Our Approaches**
- **Experiments**
- **Conclusions**

Introduction

- **Why On-chip Multiprocessor?**
 - Increasingly employed in embedded and mobile system domain
 - Popular on-chip multiprocessors: IBM's Power4, Sun's MAJC-5200, MP98.
 - Multi-core is a trend.
- **Important issues in parallel applications**
 - Parallelism
 - Load balance
 - Energy Saving
- **Our focus: Exploiting load imbalance to save energy consumption utilizing DVS.**

Introduction (Continued)

- **Load Imbalance**
 - **Load imbalance in parallel fragments**
 - **Load imbalance in serial fragments**
- **Main strategies**
 - **Single processor is used for serial fragments while other redundant processors are shut down**
 - **Each processor's frequency and voltage are adjusted in terms of load imbalance during parallel fragments**

Related work

- **An energy saving strategy based on adaptive loop parallelization [I. Kadayif et al., DAC'02]**
- **An integer linear programming based approach for parallelizing applications in on-chip multiprocessors [I. Kadayif et al., DAC'02]**
- **Exploiting processor workload heterogeneity for reducing energy consumption in chip multiprocessors [I. Kadayif et al., DATE'04]**
- **A new load-balancing algorithm for parallel sparse matrix-vector multiplication [Hu Qingfeng et al., ICPACE'03]**

On-Chip Multiprocessor

- **Architecture:**
 - four processors, each has its own instruction and data cache
 - share off-chip DRAM memory
 - interprocessor synchronization logic
 - frequency/voltage scaling-capable processor
- **Tool: Simpower**

On-Chip Multiprocessor (Continued)

■ Simulation parameters

Simulation Parameter	Value
Number of frequency/voltage Levels	8
Lowest/Highest Supply Voltage	0.8V / 1.25 V
Maximum CPU frequency	1 GHz
Frequency Step Size	100 MHz
Frequency/Voltage Transition Penalty	10 cycles / 2.2 nJ
Technology Parameter	0.35 micron
Instruction Cache Configure	4KB 2-way associative 32 byte blocks
Data Cache Configuration	4KB 2-way associative 32 byte blocks
Data Cache Hit Latency	1 cycle
Memory Access Latency	100 cycles
Cache Energy Per Access	0.60 nJ
Per Access Energy for Memory	23.10 nJ
Energy Reduction Factor	0.1
Resynchronization Time	20 msec

Table 1. Simulation Parameters

Energy Saving Opportunities

- **Workload Assignment:**
 - **Static assignment (block, cyclic)**
 - **Dynamic assignment**
- **Special case:**
 - **irregular loop nests**

Irregular Parallel Loop Nests & Load Imbalance

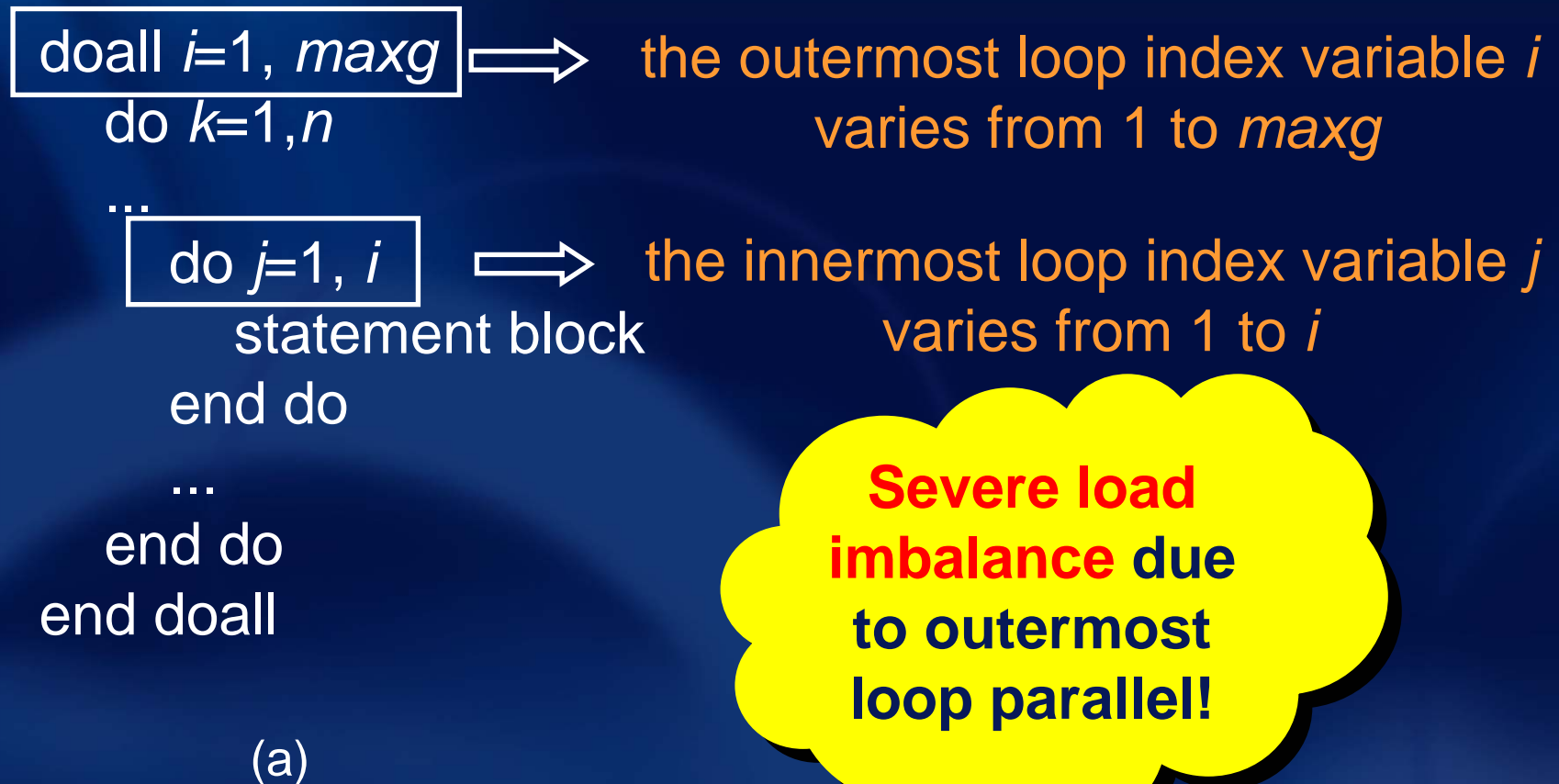
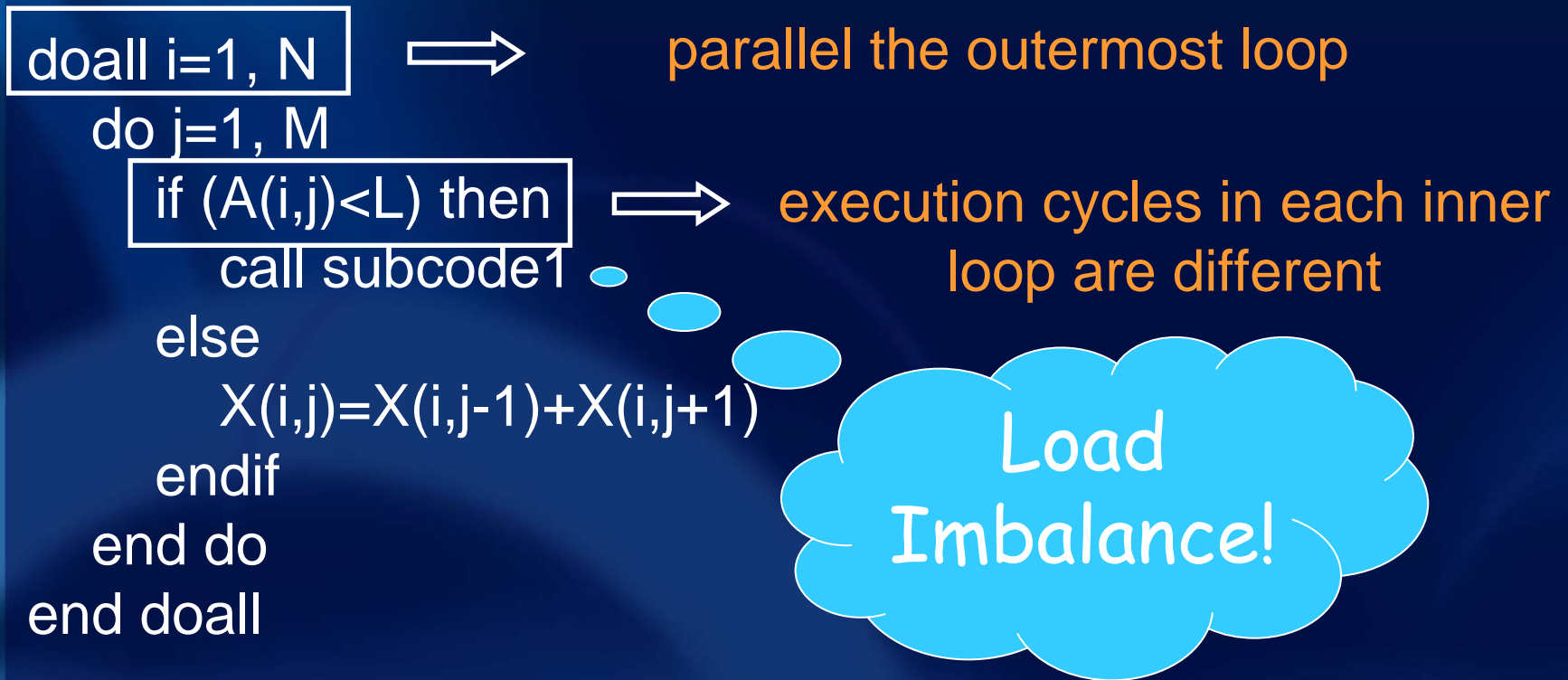


Figure 1. Irregular parallel loop nests which may cause load imbalance

Irregular Parallel Loop Nests & Load Imbalance



(b)

Figure 1. Irregular parallel loop nests which may cause load imbalance

Our Approaches

- **Load Imbalance Analyses:** the compiler estimates the load of each processor for each parallel and serial fragment
- **Frequency-Voltage Setting:** each processor is adjusted to the suitable fre./vol. values during each parallel or serial fragment
- **Optimization Approaches:** avoiding the additional energy overhead offsetting the energy benefit

Load Imbalance Analyses

- Estimate the workload of each processor during each parallel and serial fragment

For processor j ($1 \leq j \leq P$) and parallel fragment i ($1 \leq i \leq s$)

$PS_{i,j}$: the code fragment assigned to j -th processor for i -th parallel fragment

$W(PS_{ij})$: the workload of code fragment PS_{ij}

$\sum_{j=1}^P \left| W(PS_{ij}) - \max_j \{W(PS_{ij})\} \right|$: the degree of load imbalance

Load Imbalance Analyses

- **Serial Fragments – determine the execution cycles of the serial codes**
- **Parallel Fragments**
 - **estimate per-iteration cost**
 - **estimate the number of iteration**

Frequency-Voltage Setting

- **Workload as non-increasing order:**

$$W(PS_{i_1}), W(PS_{i_2}), \dots, W(PS_{ij}), \dots, W(PS_{i_P})$$

- **Normalized with respect to the largest workload:**

$$1 \geq \frac{W(PS_{i_2})}{W(PS_{i_1})} \geq \frac{W(PS_{i_3})}{W(PS_{i_1})} \geq \dots \geq \frac{W(PS_{ij})}{W(PS_{i_1})} \geq \dots \geq \frac{W(PS_{i_P})}{W(PS_{i_1})}$$

- **The highest frequency is assumed as f_{\max}**

- **Other processors' frequencies:**

$$f_{\max} \cdot \frac{W(PS_{i_2})}{W(PS_{i_1})}, f_{\max} \cdot \frac{W(PS_{i_3})}{W(PS_{i_1})}, \dots, f_{\max} \cdot \frac{W(PS_{ij})}{W(PS_{i_1})}, \dots, f_{\max} \cdot \frac{W(PS_{i_P})}{W(PS_{i_1})}$$

Frequency-Voltage Setting

- For serial fragment i

$$W(PS_{i2}) = W(PS_{i3}) = \dots = W(PS_{iP}) = 0$$

- The first processor is operated with maximum frequency and highest voltage level
- Other processors' frequencies are set zero

$$f_{\max} \cdot \frac{W(PS_{i2})}{W(PS_{i1})} = 0, \dots, f_{\max} \cdot \frac{W(PS_{ij})}{W(PS_{i1})} = 0, \dots, f_{\max} \cdot \frac{W(PS_{iP})}{W(PS_{i1})} = 0$$

Optimization Approaches

- **Some additional energy overhead may offset the energy benefit**
 - **inherit program characteristics – avoiding the unnecessary frequency transition**
 - **careless processor assignment – reusing the same processor as much as possible for the same frequency/voltage throughout the whole execution**

Experiments

■ Core program of *SWEEP3D*

```
doall i=1, maxg
  do k=1,n
    ...
    do j=1, i
      statement block
    end do
  ...
end do
end doall
```

Static block assignment



Freq/voltage
scaling

Energy optimization results

Fig. 1(a)

Experiments (Continued)

■ Before Fre. Scaling & After Fre. Scaling

Proc	Normalized Load	Initial Fre.(MHz)	Energy with initial Fre. (millijoules)	Estimated Fre.(MHz)	Actual adjusted Fre. (MHz)	Energy with Adjusted Fre. (millijoules)	Energy Savings
Pro 1	0.15	1000	160.2	96	400	40.3	74.8%
Pro 2	0.43	1000	167.5	392	400	50.2	70.0%
Pro 3	0.72	1000	170.2	724	800	90.1	47.1%
Pro 4	1	1000	175.8	1000	1000	175.8	0
Total	--	--	673.7	--	--	356.4	47.1%

Table 2. Frequency scaling and energy savings for the main load-unbalanced loop nest in SWEEP3D program

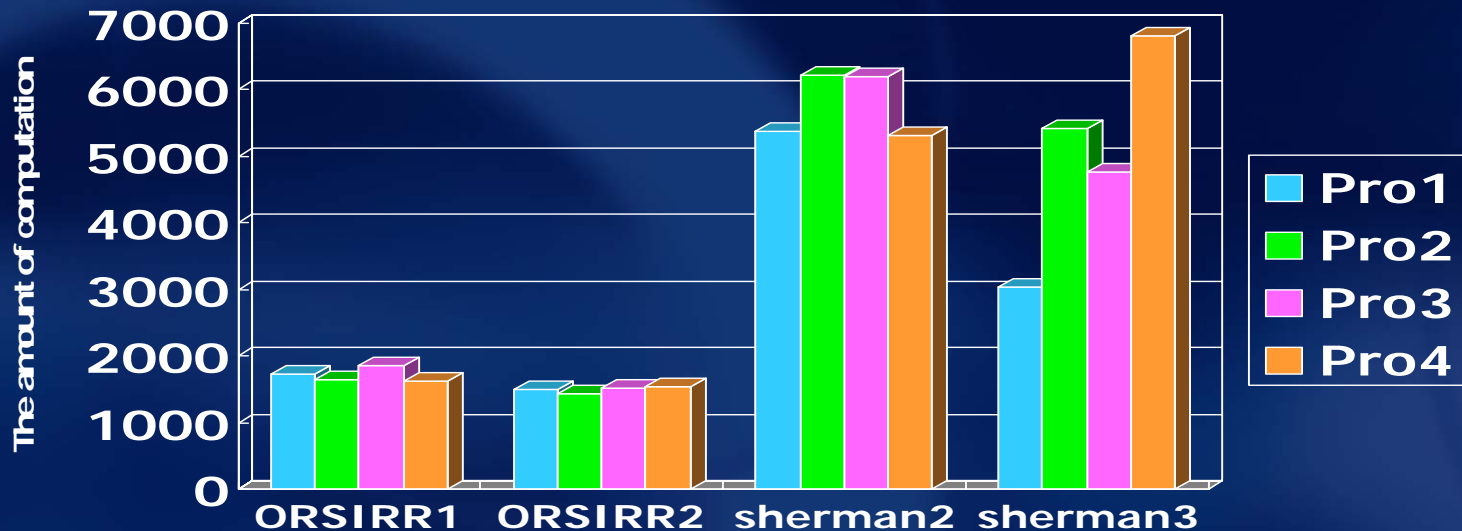
Experiments (Continued)

■ *Sparse Matrix-Vector Product*

- select four sparse matrices ORSIRR1, ORSIRR2, sherman2 and sherman3 from Harwell-Boeing sparse matrix set

■ Load Imbalance

Figure 5. Load imbalance due to the amount of computation in terms of algorithm provided by Aliaga and Hernandez



Experiments (Continued)

■ Energy Saving Proportion to four sparse matrix-vector products

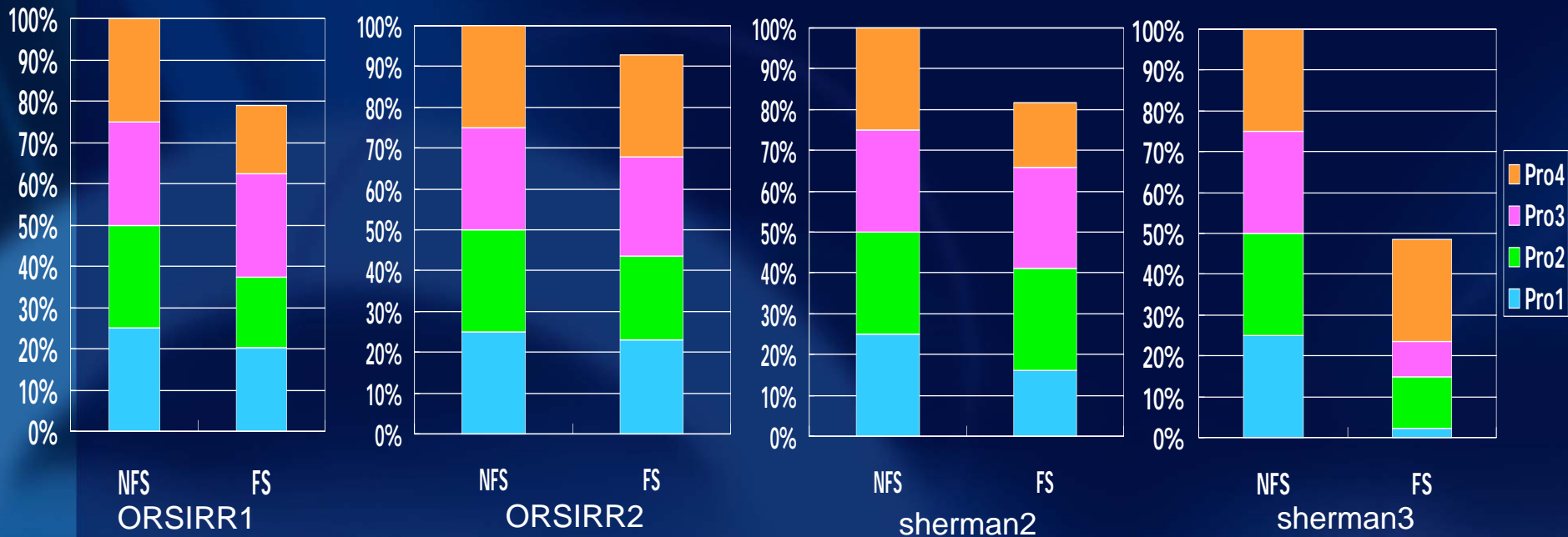
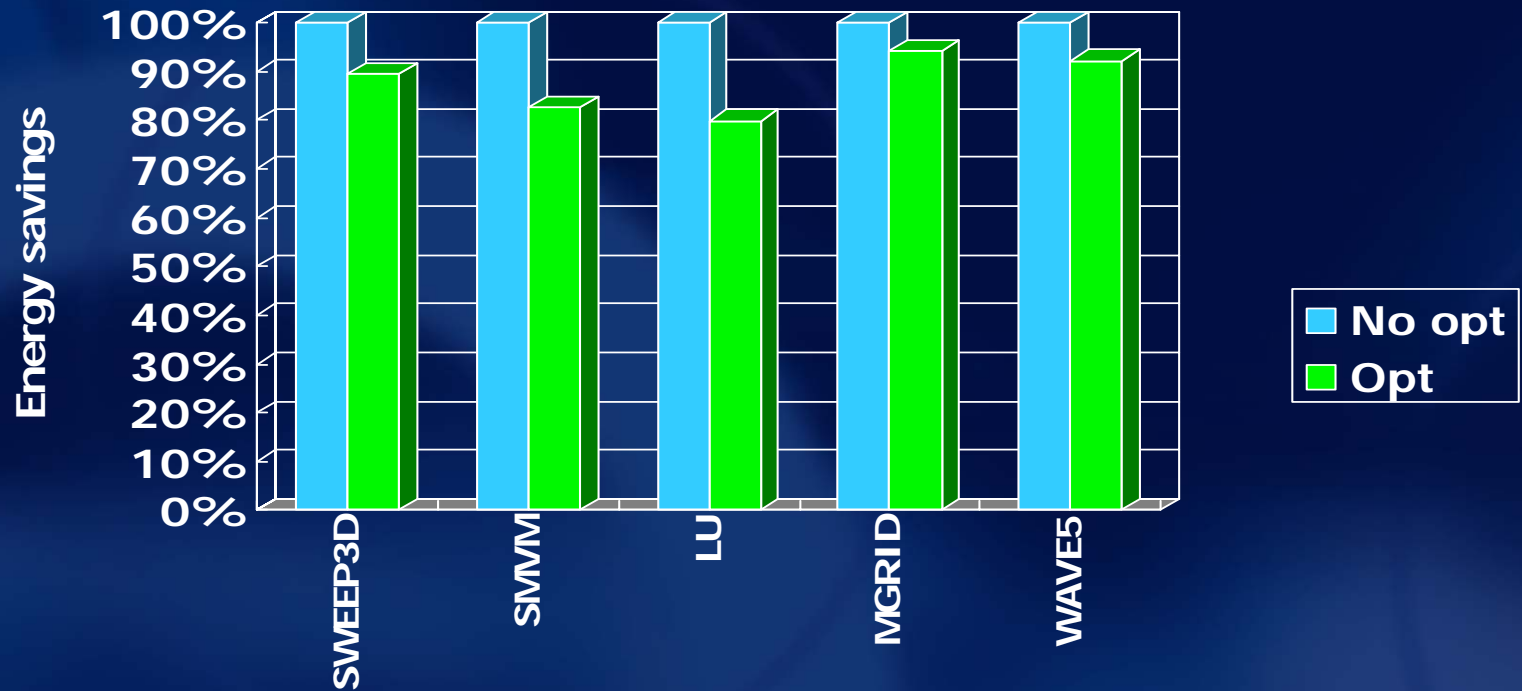


Figure 6. Energy saving proportions for four sparse matrix-vector products

Experiments (Continued)

Figure 7. Normalized energy savings for each program



Conclusions

- **On-chip multiprocessor architecture is becoming popular**
- **Two trends:**
 - **chip multiprocessing**
 - **frequency/voltage scaling**
- **Compiler-directed energy saving strategy is provided**
 - **exploiting load imbalance**
 - **utilizing DVS to save energy consumption**

Thanks!