Energy-Oriented Compiler Optimizations for Partitioned Memory Architectures

Power & Energy Management

Light Seminar

March 29, 2001

Motivation

- Processor market is estimated >90% in embedded systems
- Embedded systems' applications spend up to 90% energy in memory system

Framework

- Partitioned memory architecture
  - Multiple memory banks (ie, 64x1MB, 32x2MB, 16x4MB, etc)
  - Multiple power modes

	Active	Standby	Napping	Power-Down	Disabled
Energy Consmp. (nJ)	3.570	0.830	0.320	0.005	0.000
Re-sync. Time (cyc.)	0	2	30	9,000	NA

Figure 1: Energy consumptions and re-synchronization times.

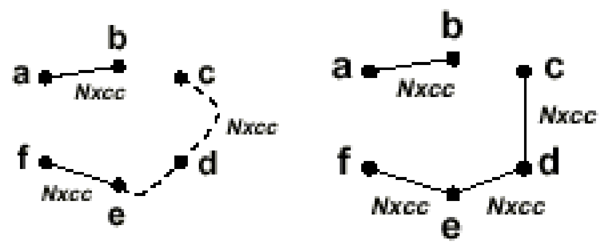
Optimizations

- Array Allocation
- Transformations (via SUIF infrastructure)
  - Loop fission
  - Loop splitting
  - Array renaming

Optimizations

- Array Allocation: Examine arrays with similar access patterns
  - Build Array Relation Graph (ARG)
    - Node == array
    - Edge weight == # of times incident arrays are accessed in loop
  - Find max weight cover
  - Place neighboring arrays into adjacent memory location
  - Bias towards large edge weights

```
for(i=0;i<N;i++)
  {a[i],b[i]}
for(i=0;i<N;i++)
  {c[i],d[i],e[i]}
for(i=0;i<N;i++)
  {e[i],f[i]}
```



Optimizations

- Transformations
  - Loop Fission

```
for(i=0;i<N;i++)
 {
   {a[i],b[i]}
   {c[i],d[i]}
 }
```

$\Longrightarrow$

```
for(i=0;i<N;i++)
  {a[i],b[i]}
for(i=0;i<N;i++)
  {c[i],d[i]}
```

```
for(...)
 {
   S_1
   S_2
   S_3
   ...
   S_{K-1}
   S_K
 }
```
,
```
for(...)
   S_1
for(...)
 {
   S_2
   S_3
   ...
   S_{K-1}
   S_K
 }
```
,
```
for(...)
 {
   S_1
   S_2
 }
for(...)
 {
   S_3
   ...
   S_{K-1}
   S_K
 }
```
,....
```
for(...)
 {
   S_1
   S_2
   S_3
   ...
   S_{K-1}
 }
for(...)
   S_K
```
,
```
for(...)
  S_1
for(...)
  S_2
for(...)
  S_3
...
for(...)
  S_K
```

Optimizations

- Transformations
  - Loop Splitting (Index Set Splitting)

$$\texttt{for}(i{=}0;i{<}N;i{+}{+})\ \{\texttt{a}[i],\texttt{b}[i]\} \Longrightarrow \begin{array}{l}\texttt{for}(i{=}0;i{<}N/2;i{+}{+})\\ \ \{\texttt{a}[i],\texttt{b}[i]\}\\ \texttt{for}(i{=}N/2+1;i{<}N;i{+}{+})\\ \ \{\texttt{a}[i],\texttt{b}[i]\}\end{array}$$

Optimizations

- Transformations
  - Array Renaming (Live Variable Analysis)
    - Feasible if **a > b** in size

```
for(i=0;i<N;i++)              for(i=0;i<N;i++)
 {a[i],c[i]}                   {a[i],c[i]}
 .....              ==>        .....
for(i=0;i<N;i++)              for(i=0;i<N;i++)
 {b[i],c[i]}                   {a[i],c[i]}
```

Evaluation System

- SimplePower?

Evaluation System

- SimplePower?
- 64MB physical memory

Evaluation System

- SimplePower?
- 64MB physical memory
- No virtual memory

Evaluation System

- SimplePower?
- 64MB physical memory
- No virtual memory
- No cache

Evaluation System

- SimplePower?
- 64MB physical memory
- No virtual memory
- No cache
- Single program environment

Evaluation System

- SimplePower?
- 64MB physical memory
- No virtual memory
- No cache
- Single program environment
- Working set size < 64MB

Evaluation System

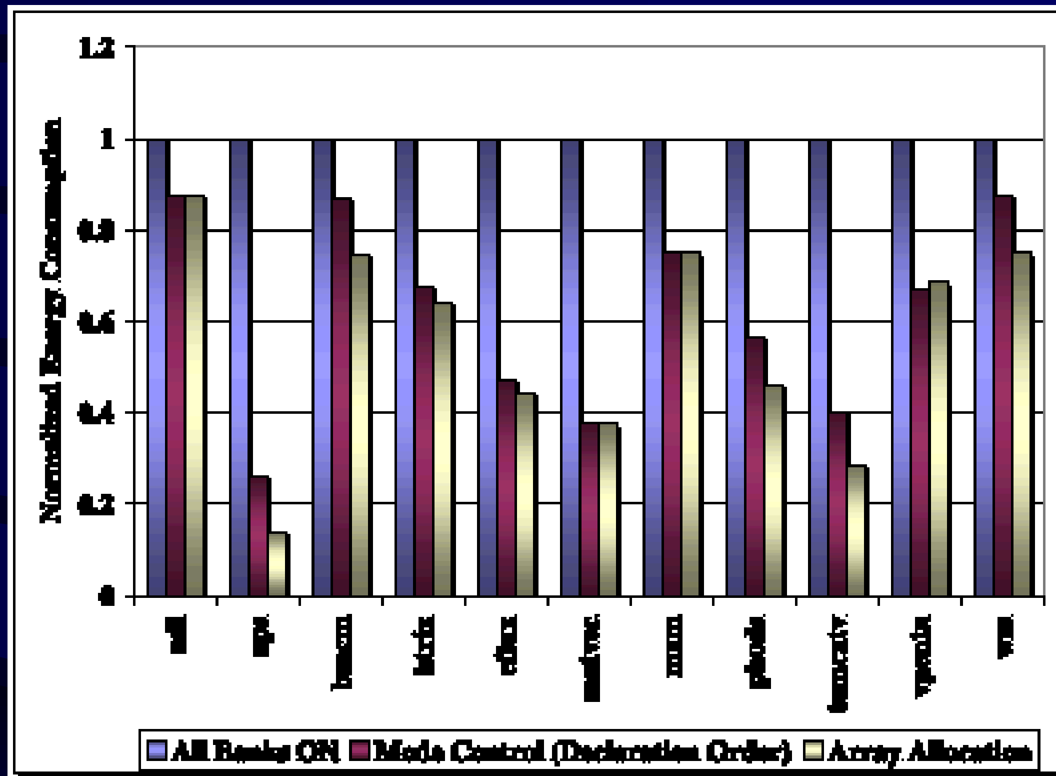
- SimplePower?
- 64MB physical memory (in most cases)
- No virtual memory
- No cache
- Single program environment
- Working set size < 64MB
- Mode Control

Benchmarks

Benchmark	Source	Data Size (MB)	Bank Configuration	Energy Consumption (mJ)
adi	Livermore	48.0	8× 8MB	3.38
aps	Perfect Club	41.5	8× 8MB	2.56
bmcm	Perfect Club	3.0	8× 0.5MB	1,040.34
btrix	Spec'92	47.3	8× 8MB	2.49
eflux	Perfect Club	33.6	16× 4MB	826.46
matvec	[1]	16.0	8× 8MB	675.86
mxm	Spec'92	48.0	8× 0.5MB	10,572.57
phods	[3]	33.0	8× 8MB	1,137.38
tomcatv	Spec'95	56.0	8× 8MB	119.78
vpenta	Spec'92	60.0	32× 2MB	2,026.66
wss	Perfect Club	3.0	8× 0.5MB	7,032.03

- Usually 1 or 2 loop nests dominate energy consumption
- Focus on most costly nest

Results



- Array Allocation: Optimal solution is NP-Hard
  - Layout based on most costly nest

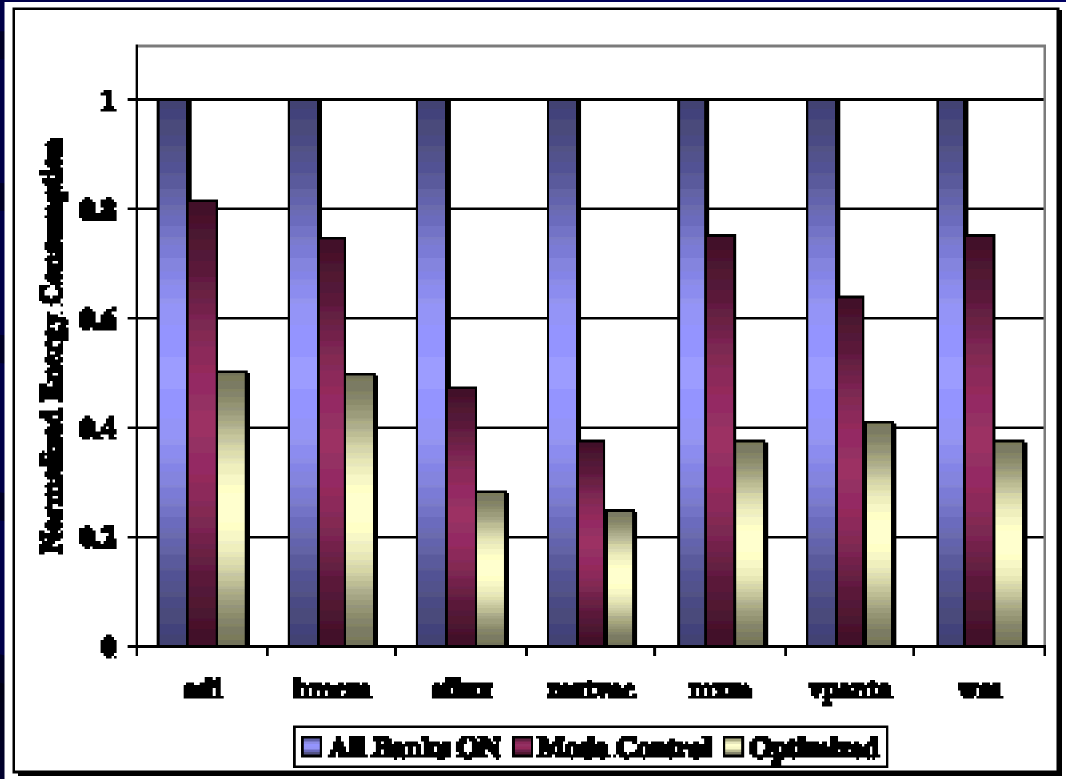
Results

Benchmark	Alternative Fission Strategies for the Most Costly Nest								
	#1	#2	#3	#4	#5	#6	#7	#8	#9
adi	47.0%	47.0%	61.2%						
aps	48.5%	48.5%	48.5%	48.5%					
eflux	47.0%	45.2%	43.3%	66.9%					
matvec	49.8%	33.2%	16.6%	58.2%					
tomcatv	49.5%								
vpenta	8.2%	23.5%	16.6%	24.9%	18.0%	16.6%	20.7%	8.2%	48.4%

Figure 11: Percentage energy improvements due to different loop fission alternatives.

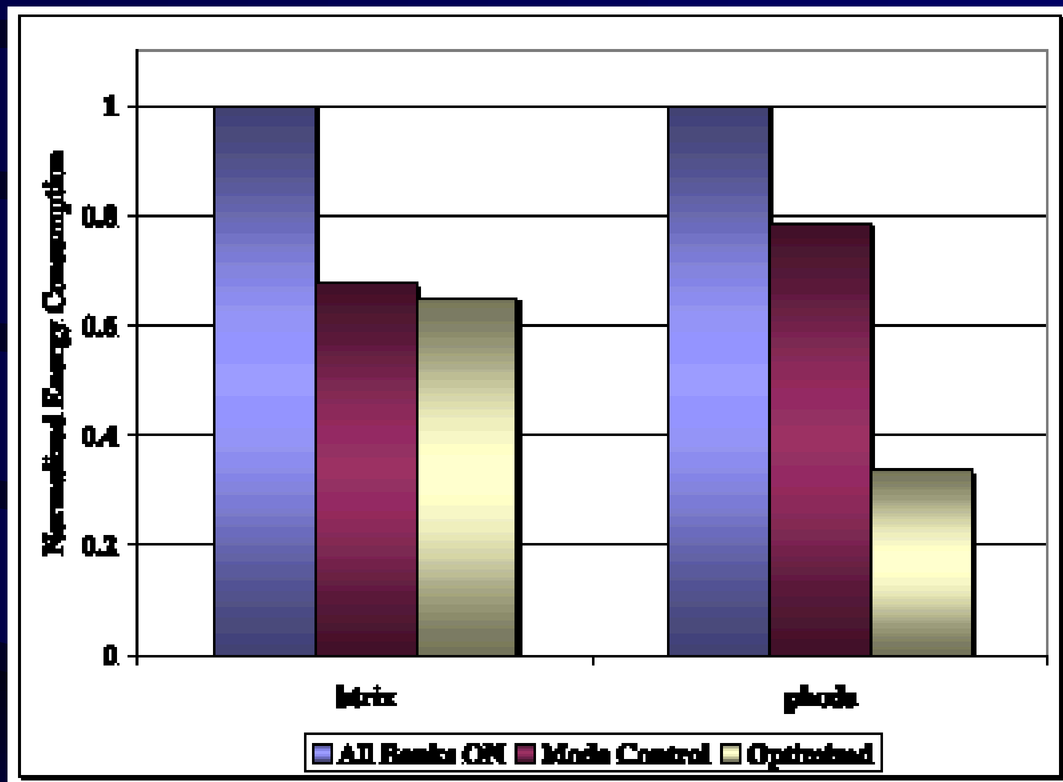
- Loop Fission
  - Loops containing single instruction?
  - Second most costly nest?
  - Average improvement: 55.5%

Results



- Loop Splitting
  - 61.5% reduction vs. All ON
  - 42.8% reduction vs. Mode Control

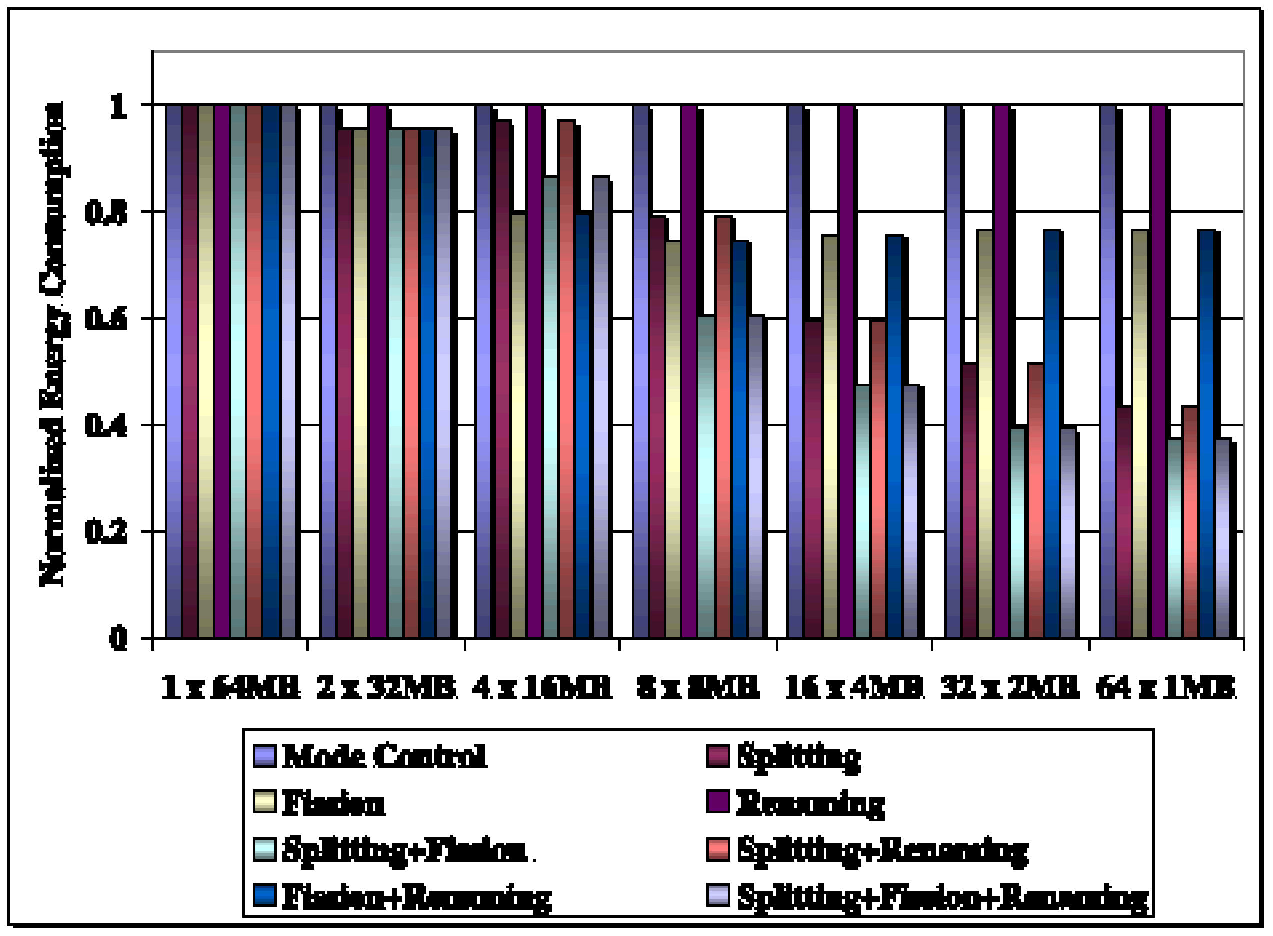
Results



- Array Renaming
  - Only 2 benchmarks; expect more opportunities in larger codes

Memory Bank Configuration

- Used **eflux**: (showed best energy improvement via fission)



Cache

- Tested 4K, 2-way/4-way set-associative
- Results are comparable; why?
  – Reduces overall # of memory references
  – Longer interaccess reference times

Concluding Remarks