

# Tutorial Outline

8:30 - 8:45	Introduction and motivation
8:45 - 9:05	Sources of power in CMOS designs
9:05 - 9:30	Power analysis tools and techniques
9:30 - 10:30	Gate & functional unit design issues & techniques
10:30 - 10:50	BREAK
10:50 - 12:15	Architectural level issues and techniques
12:15 - 1:30	LUNCH
1:30 - 2:30	Low power memory system design
2:30 - 3:30	Software level issues and techniques
3:30 - 3:50	BREAK
3:50 - 4:30	Software level issues and techniques, con't
4:30 - 4:45	Future challenges

ISCA Tutorial: Low Power Design

Software.1

©MJIVN, PSU, 2000

# Motivation

- Increasing software content in embedded devices
- Traditionally consider performance

**Are performance-oriented optimizations  
suitable for energy?**

ISCA Tutorial: Low Power Design

Software.2

©MJIVN, PSU, 2000

## Sources of Software Energy Consumption

- Datapaths in integer ALU and FP units
- Cache and memory systems
- System buses (address, instruction, and data)
- Control circuitry and clock logic and distribution

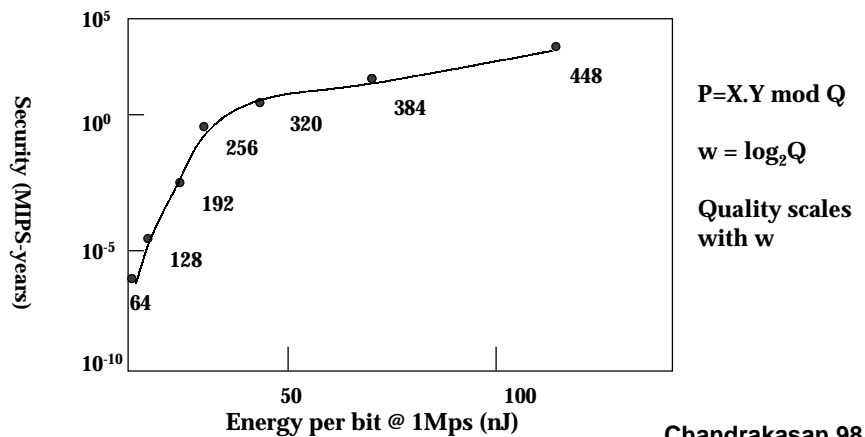
## Levels of Optimization

- Algorithm and Application Design
- Compiler Optimizations
- Operating System Control

# Algorithm- Power Tradeoffs

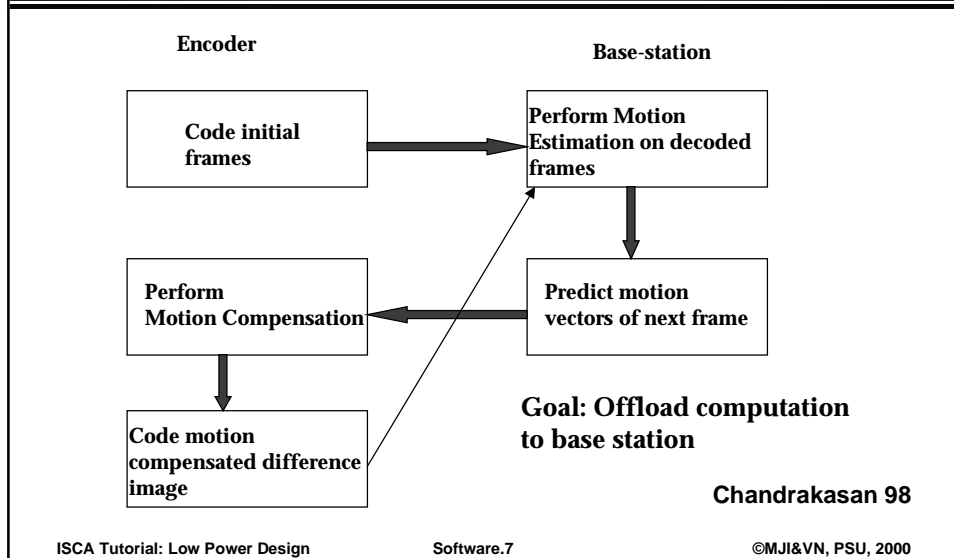
- Trading of Quality for Power
  - » Black and White Video instead of Color Video
  - » Fetch Map with less details
  - » Scalable Encryption Algorithms
  - » Network-driven partitioning of algorithms

# Energy Scalable Encryption

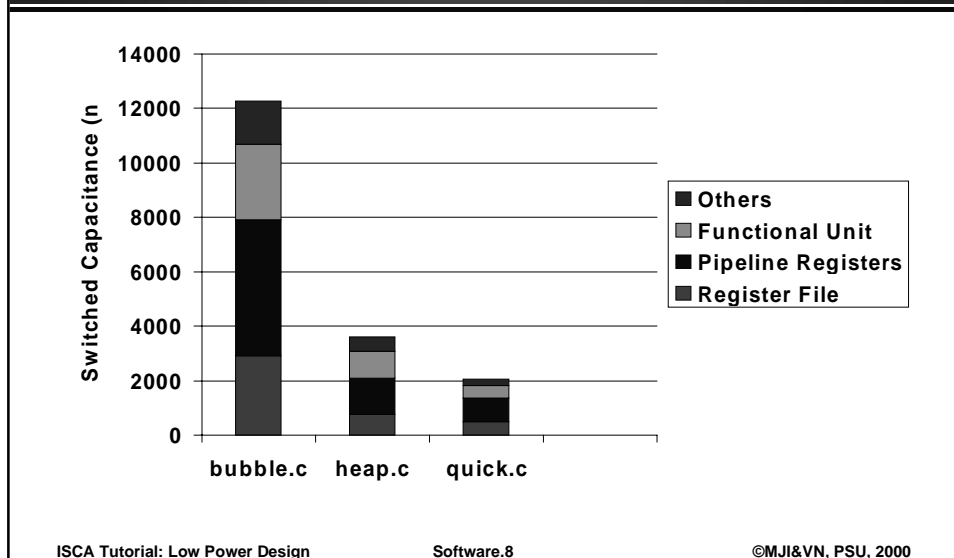


Chandrakasan 98

# Network Driven Motion Estimation



# Datapath Energy Consumption

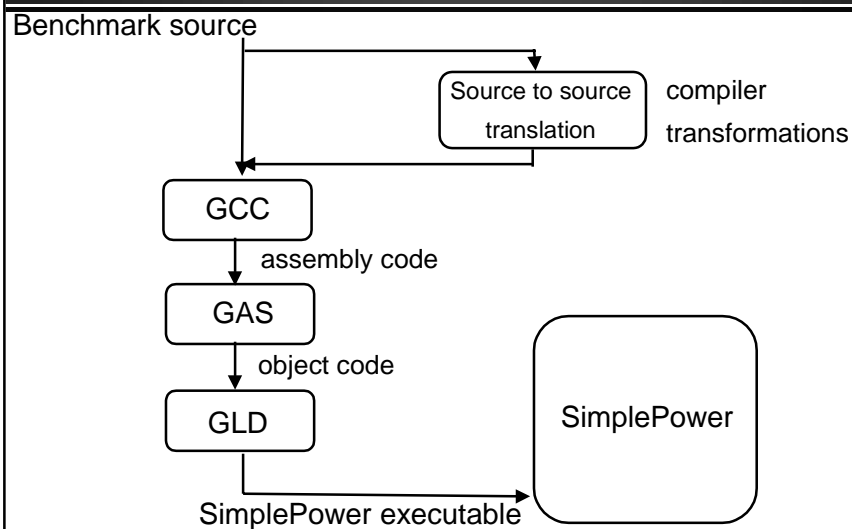


## Motivation

- Increasing software content in embedded devices
- Compilers traditionally consider performance

**Are performance-oriented optimizations suitable for energy?**

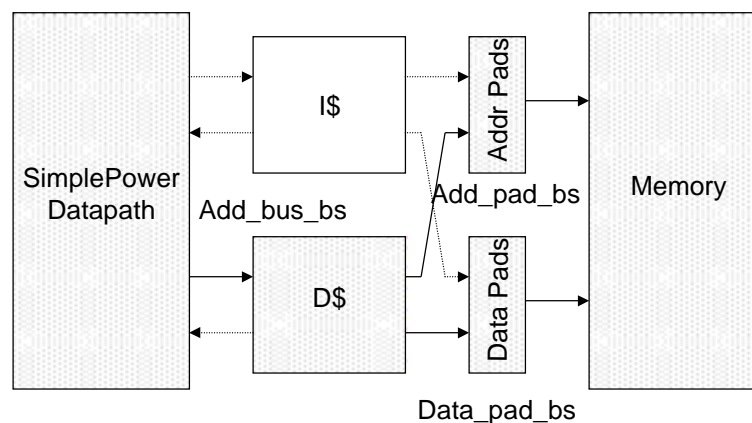
## SimplePower Compiler Framework



# Compiler Optimizations for Energy Reduction

- Reduce memory energy
- Optimal selection & scheduling of machine instructions
- Exploiting low power features of the machine

# Architectural Model



## Reducing Memory Energy

- Improve the locality of memory accesses
- Minimize the number of memory accesses
- Optimizing interactions of compiler and cache architecture
- Reduce the total memory area (in embedded systems)
- Make effective use of memory bandwidth

## Improving Locality: Loop Transformations

- Linear loop transformations
  - » Loop permutation
  - » Loop skewing
  - » Loop reversal
  - » Loop scaling
- Iteration space tiling
- Multi-loop transformations
  - » Loop Fusion/Fission

## Example: Loop Permutation

```
For(l=0; l<n; l++)  
  For(j=0; j<n; j++)  
    U[j][l]=V[j][l];
```




```
For(j=0; j<n; j++)  
  For(l=0; l<n; l++)  
    U[j][l]=V[j][l];
```

- Improves locality since array elements are accessed in sequence
- Improves both cache performance and energy

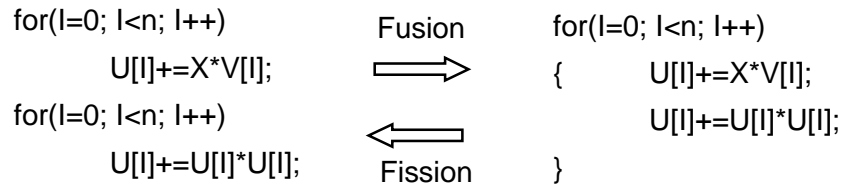
## Iteration Space Tiling

```
for(l=0; l<n; l++)  
  for(j=0; j<n; j++)  
    for(k=0; k<n; k++)  
      U[l][j]+=V[l][k]*W[k][j];
```

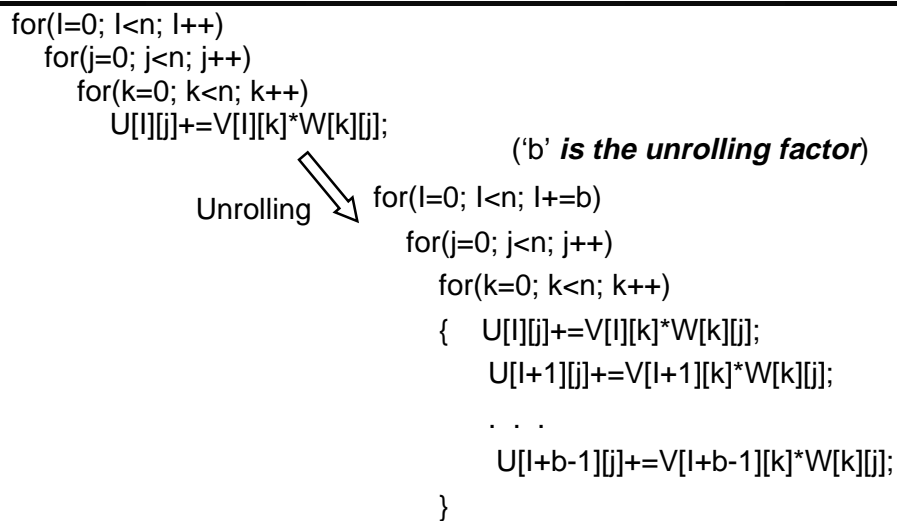
 Tiling (*t* is the tile size)

```
for(l1=0; l1<n; l1+=t)  
  for(j1=0; j1<n; j1+=t)  
    for(k1=0; k1<n; k1+=t)  
      for(l=l1; l<min(l1+t, n); l++)  
        for(j=j1; j<min(j1+t, n); j++)  
          for(k=k1; k<min(k1+t, n); k++)  
            U[l][j]+=V[l][k]*W[k][j];
```

# Loop Fusion and Fission



# Loop Unrolling



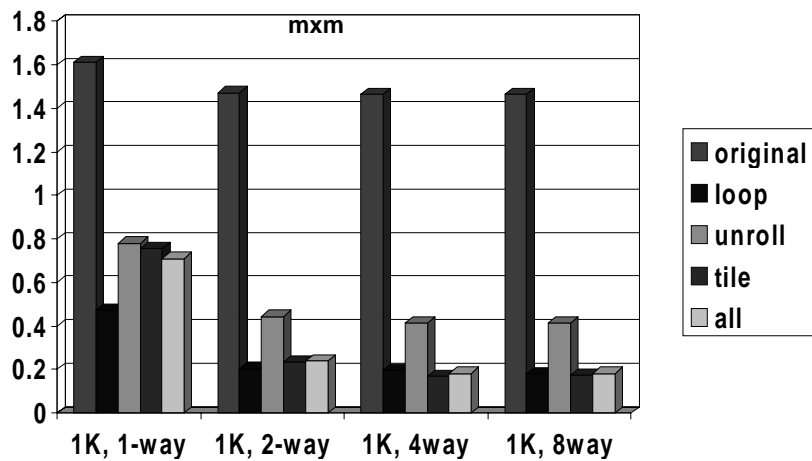
# Scalar Expansion

```
for(l=0; l<n; l++)  
{ K=U[l]+U[l-1];  
  V[l]=K+1/K;  
}
```

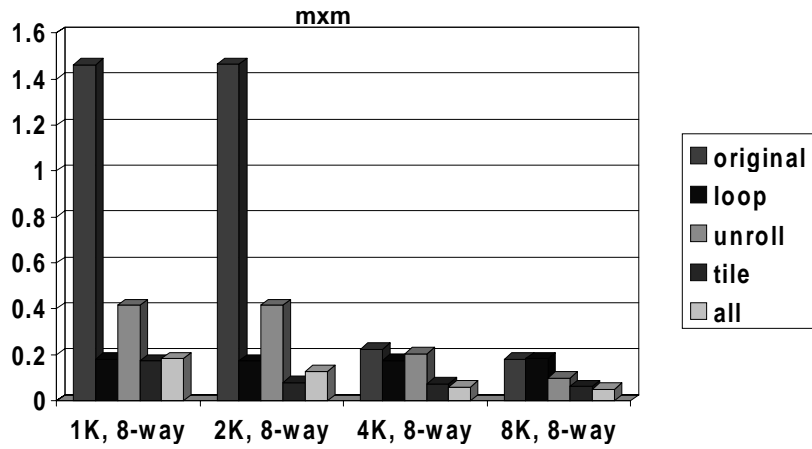
Scalar Expansion  
→

```
for(l=0; l<n; l++)  
{ K[l]=U[l]+U[l-1];  
  V[l]=K[l]+1/K[l];  
}
```

# Memory Energy (J)



## Memory Energy (J)

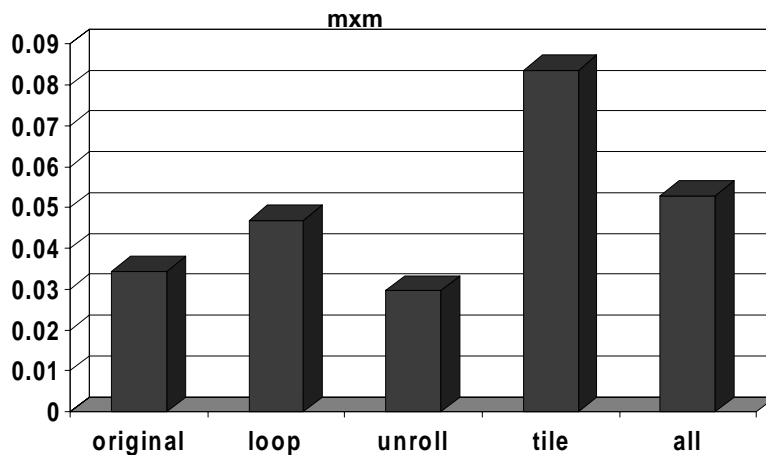


ISCA Tutorial: Low Power Design

Software.21

©MJIVN, PSU, 2000

## Datapath Energy (J)



ISCA Tutorial: Low Power Design

Software.22

©MJIVN, PSU, 2000

## Improving Locality: Data Transformations

- Linear layout transformations
  - » Dimension re-indexing
  - » Diagonal (skewed) memory layouts
- Blocked memory layouts

Data transformations might be useful where loop transformations fail, but they have their own problems (e.g. aliasing)

## Example: Dimension Reindexing

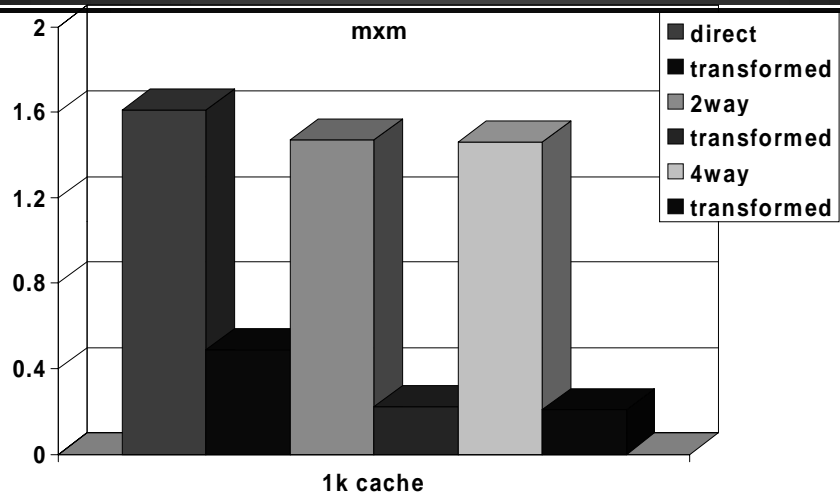
For  $I = 1, N$   
For  $J = 1, N$   
 $A[I][J] = B[J][I]$



For  $I = 1, N$   
For  $J = 1, N$   
 $A'[I][J] = B[I][J]$

- Imitates the effect of a different layout
- Should be applied with a global view
- Less negative impact on datapath energy

## Data Transformation Effects



ISCA Tutorial: Low Power Design

Software.25

©MJIVN, PSU, 2000

## Minimizing Number of Memory Accesses

- Scalar replacement enables effective use of registers
- Can be extended to larger granularities (multiple levels of caches, scratch-pad-memories)

```
For I = 1, N
  For J = 1, N
    ... A[I] ...
```

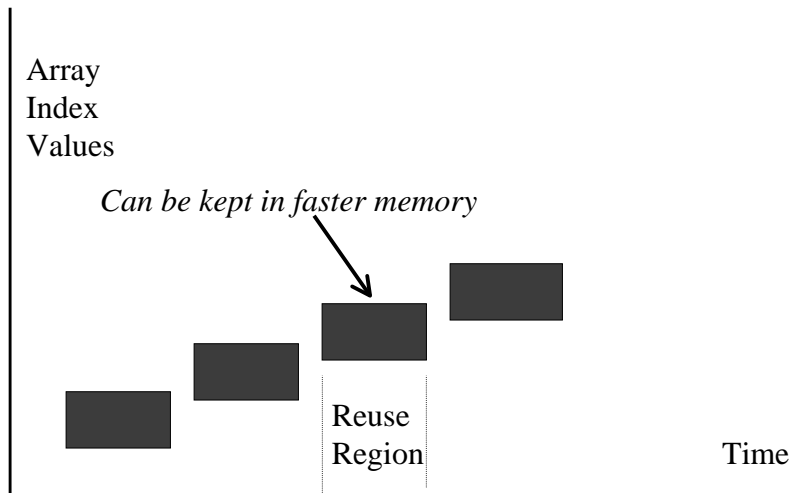
```
For I = 1, N
  x = A[I]
  For J = 1, N
    ... x ...
  A[I] = x
```

ISCA Tutorial: Low Power Design

Software.26

©MJIVN, PSU, 2000

# Exploiting Temporal Locality

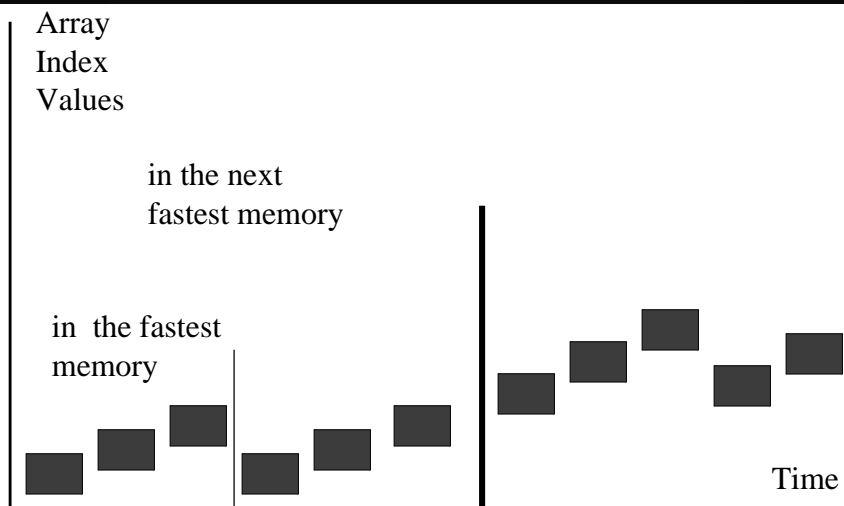


ISCA Tutorial: Low Power Design

Software.27

©MJIVN, PSU, 2000

# Exploiting Temporal Locality (Multiple Levels)



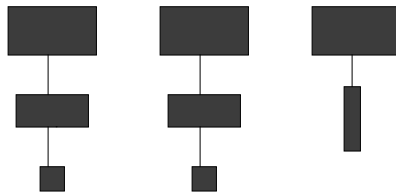
ISCA Tutorial: Low Power Design

Software.28

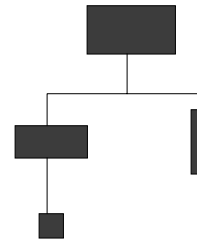
©MJIVN, PSU, 2000

# Memory Trees

individual memory trees



combined memory tree



# Reducing Memory Area

**For I = 1, N**

... = C[I]

**For I = 1, N**

B[I] = A[I]

**For I = 1, N**

... = C[I]

**For I = 1, N**

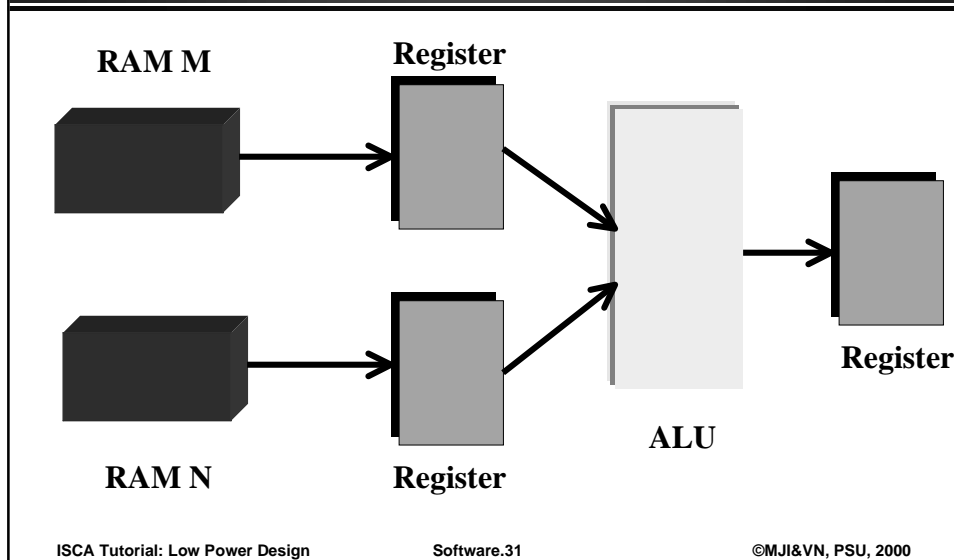
C[I] = A[I]

Reusing the same  
memory space

- » Can reduce capacity misses
- » Can lead to smaller memory in embedded design

*last use*

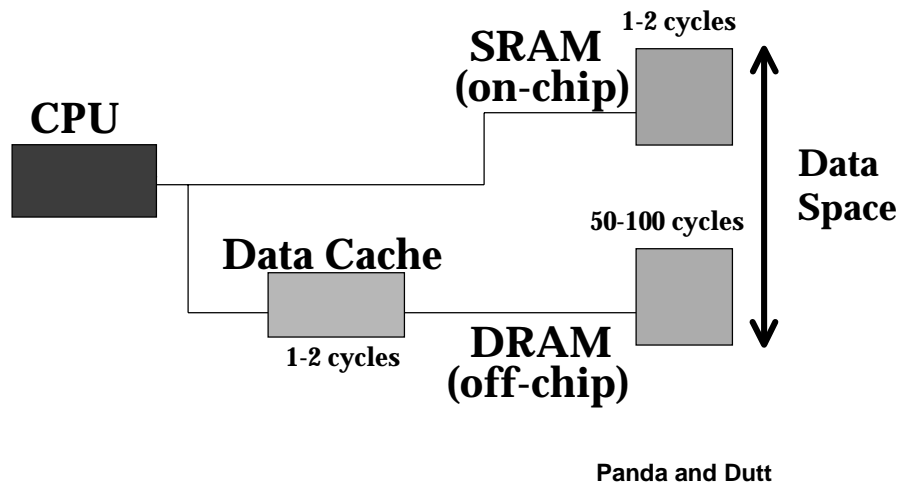
## Dual Memory Loads (Architecture)



## Dual Memory Loads (Example)

$(X*Y)+Z$	Case 1	Case 2	Case 3
M	X,Y,Z	X,Y	X,Z
N		Z	Y
	LD B,X LD C,Y LD A,Z; MUL B,C ADD A,B	DLD B,X;A,Z LD C,Y MUL B,C ADD A,B	DLD B,X;C,Y LD A,Z; MUL B,C ADD A,B
Energy	10.57pJ	9.32pJ	8.85pJ

## Scratch-Pad-Memory (SPM)



ISCA Tutorial: Low Power Design

Software.33

©MJIVN, PSU, 2000

## Data Partitioning Steps

- All scalars are mapped to the SPM
- All arrays larger than SPM are mapped into off-chip memory and accessed through the cache
- For the remaining arrays: Conflicting arrays go to different memories
- Experiments show 30-33% improvement in memory latencies

ISCA Tutorial: Low Power Design

Software.34

©MJIVN, PSU, 2000

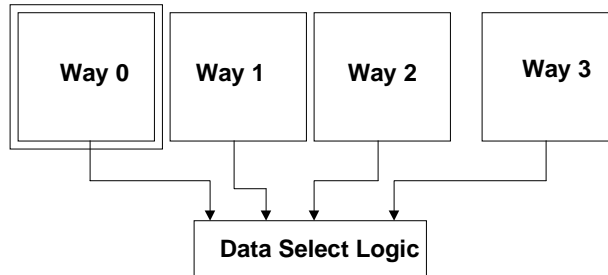
## Features Affecting Data Partitioning

- Scalar variables and constants
- Array sizes
- Life times of arrays
- Access frequency of arrays
- Access pattern and potential conflict misses

## Interaction of Optimizations and Cache Architectures

- Multiple Access Caches
- Sequential Predictive Accesses
  - » Most Recently Used Way Cache(MRU)
  - » Column Associative Cache (CA)
  - » Hash-Rehash Cache (HR)
- Selective Way Caches - Albonesi
  - » Activate different number of ways dynamically

## Most Recently Used Cache



ISCA Tutorial: Low Power Design

Software.37

©MJIVN, PSU, 2000

## Evaluation of Different Cache Architectures

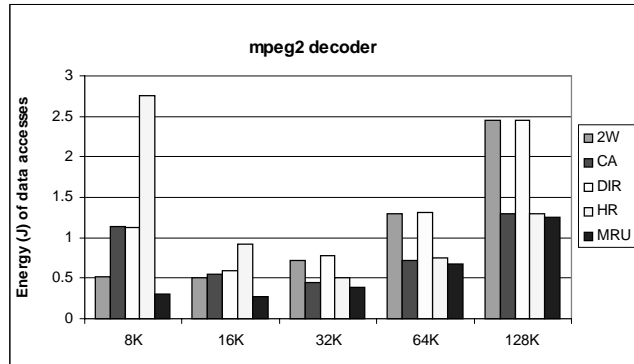
- MRU caches consume the least energy for all sizes of caches and for all the benchmarks
- The total memory system energy saving of the CA cache gets ahead of the two-way set-associative caches as cache size grows
- HR caches show poorer energy consumption compared to other multiple access caches

ISCA Tutorial: Low Power Design

Software.38

©MJIVN, PSU, 2000

## Evaluation of Different Cache Architectures



ISCA Tutorial: Low Power Design

Software.39

©MJIVN, PSU, 2000

## Influence of Compiler Optimizations - Optimization Flags

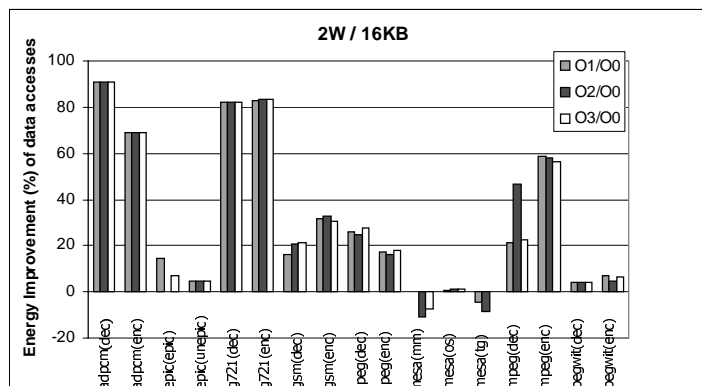
- O1
  - » peephole optimizations (redundant instruction elimination, etc.)
- O2
  - » Induction variable elimination, local/global common sub-expression elimination, etc.)
- O3
  - » Loop unrolling, in-lining, software pipelining

ISCA Tutorial: Low Power Design

Software.40

©MJIVN, PSU, 2000

## Influence of Compiler Optimizations - Energy Consumption of Data Accesses



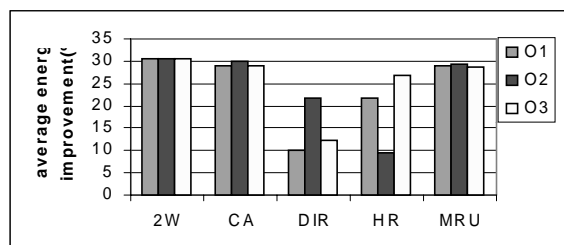
ISCA Tutorial: Low Power Design

Software.41

©MJIVN, PSU, 2000

## Influence of Compiler Optimizations - Energy Consumption of Data Accesses

- Compiler optimizations make possible considerable energy reduction for all cache configurations ← reduced number of data references

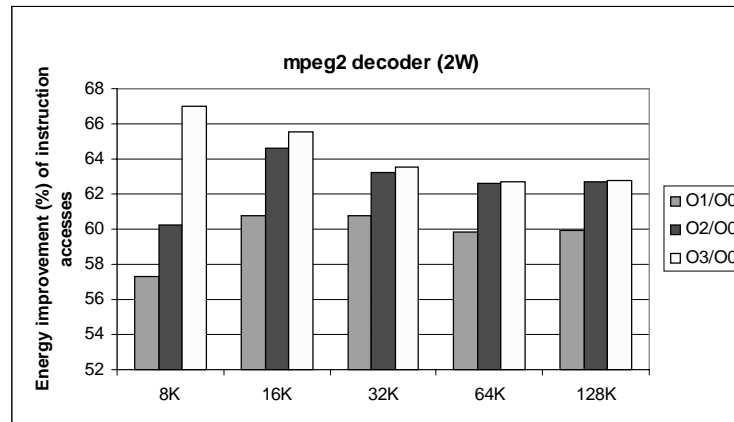


ISCA Tutorial: Low Power Design

Software.42

©MJIVN, PSU, 2000

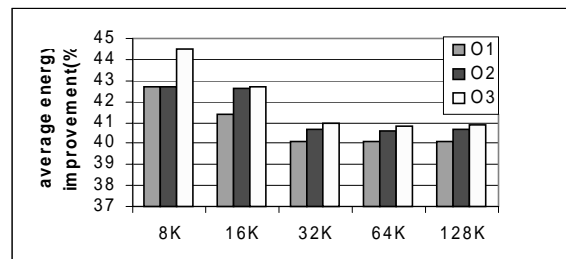
## Influence of Compiler Optimizations - Energy Consumption of Instruction Accesses



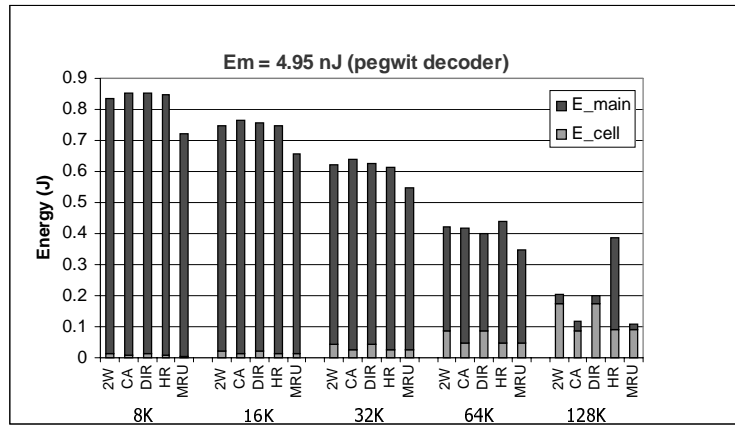
Kim 2000

## Influence of Compiler Optimizations - Energy Consumption of Instruction Accesses

- Two-way set-associative cache
- Energy savings for all the benchmarks  
 ← reduced instruction references



# Influence of Technology Changes

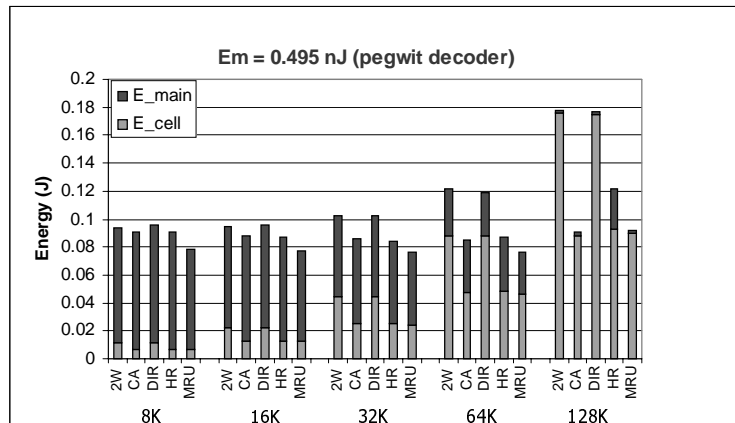


ISCA Tutorial: Low Power Design

Software.45

©MJIVN, PSU, 2000

# Influence of Technology Changes



ISCA Tutorial: Low Power Design

Software.46

©MJIVN, PSU, 2000

## Minimizing Code Space

- Storage assignment
  - » Address assignment to variables
  - » Effective use auto-increment/decrement addressing modes
- Code compression
  - » Pure Software Approach
    - Common sequences are extracted and placed in a directory
    - Instances of these sequences are replaced by mini-subroutine calls
  - » Hardware Approach
    - New instructions are defined
    - Flexible dictionary structures with architectural support

## Storage Assignment

- Many DSPs have limited addressing modes (e.g., TMS320C5)
- They use address registers to access memory
- No indexing modes, have **auto-increment/decrement** modes
- Compiler support is needed to minimize the number of explicit assignments to address registers

## Offset Assignment Problem

- Simple offset assignment
  - » One address register
  - » Works with +1/-1 offset
  - » Works on a single basic block at a time
- General offset assignment
  - » Multiple registers, larger offset values, inter-basic block analysis

Liao

## Simple Offset Assignment

$$\mathbf{c = c + d + f}$$

$$\mathbf{a = h - c}$$

$$\mathbf{b = b + e}$$

$$\mathbf{c = g - b}$$

$$\mathbf{a = a - c}$$

Based on order of  
declaration:

a,b,c,d,e,f,g,h

Based on order of first use:

c,d,f,h,a,b,e,g

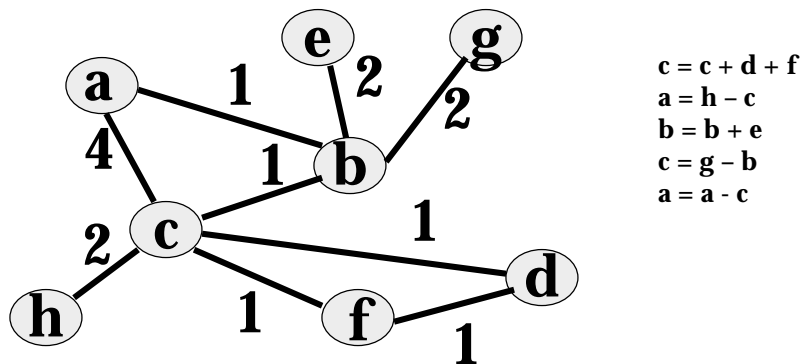
Based on compiler analysis:

a,c,h,g,b,e,f,d

## Simple Offset Assignment

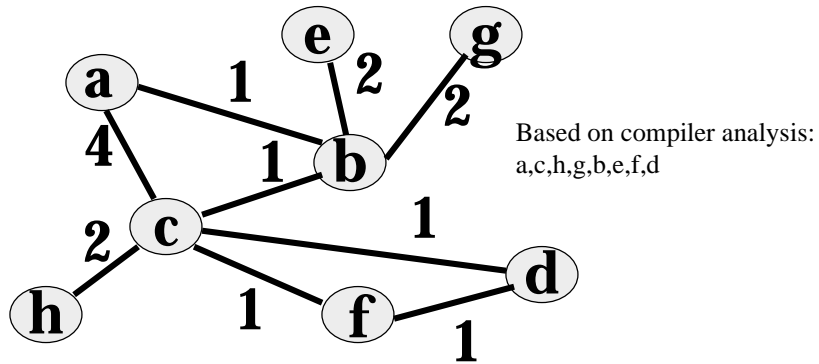
- Access sequence for “z=x op y” is “xyz”
- Access sequence for an ordered set of operations is the concatenated access sequences for each operation in the appropriate order
- Cost of assignment: number of adjacent accesses of variables that are not assigned to consecutive places
- Can be formulated as Graph Covering

## Access Graph



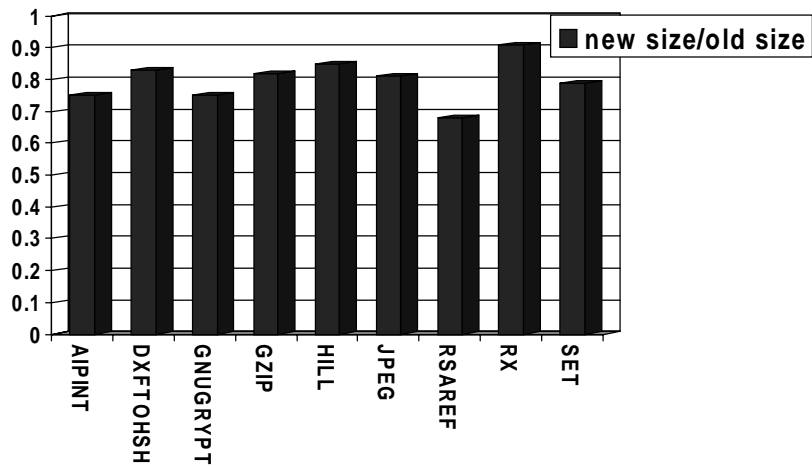
**c,d,f,c,h,c,a,b,e,b,g,b,c,a,c,a**

## Access Graph



(selected paths are in pink)

## Hardware Compression



## Compiler Optimizations for Energy Reduction

- Reduce memory energy
- Optimal selection & scheduling of machine instructions
- Exploiting low power features of the machine

## Instruction Selection & Ordering

- There are usually many possible code sequences that accomplish the same task
- Techniques:
  - » Instruction packing
  - » Instruction re-ordering (scheduling)
  - » Operand ordering/swapping

## Different Instruction Scheduling Algorithms

- Performance-oriented Scheduling (List Scheduling)
- Energy-oriented Scheduling : Top-Down approach
- Energy-oriented Scheduling : Bottom-up approach
- One-step look-ahead Scheduling

Parikh 2000

## Energy-oriented Scheduling (Top-Down Approach)

- In the input DAG,
  - » Each node – base cost denoting the average energy consumed by that instruction.
- Circuit-state cost (inter-instruction cost)
  - » Energy consumed while scheduling one instruction after the other, due to the switching activity.

## Energy-oriented Scheduling (Top-Down Approach) (Contd.)

- DAG traversed from the root towards the leaves,(Top-Down) selecting a node to be scheduled from the Candidate set.
- When there are more than one nodes in the Candidate set, the next node is selected so that the circuit state cost incurred is minimum.

## Energy-oriented Scheduling (Bottom-Up Approach)

- Works in reverse order (Bottom to Top)
- Candidate Set : A set of nodes that do not have any successors or all of whose successors have been scheduled.
- First select last instruction to be executed from the Candidate Set.
- Then, select the instruction that will precede previously selected instruction, each time trying to minimize the circuit state cost.

## Energy Scheduling (Look-ahead Approach)

- Variant of top-down scheduling.
- Instead of considering only the next immediate node to schedule, consider next two (or more) nodes at a time.

## Experiments

- Several DAGs of different sizes (in terms of nodes and edges)
- Given a number of nodes and edges, a DAG is randomly generated
- A set of six different instructions
- For performance computation:
  - » cost (number of cycles) for each instruction
  - » latency of 1, if there is a data dependence between two instructions
- For energy computation:
  - » An energy table with base (execution) cost and inter-instruction cost

## Experiments (Contd.)

Table 1 : Energy Cost Table (Proposed by Tiwari et al)

Instrucion Name	Base Cost [pJ]	Circuit-state Effects [pJ]					
		LOAD	DLOAD	ADD	MULT	LOAD;ADD	LOAD;MULT
LOAD	1.98	0.13	0.15	1.19	0.92	1.25	1.06
DLOAD	2.37	0.15	0.17	1.19	0.92	1.32	1.06
ADD	0.99	1.19	1.19	0.26	0.53	0.86	0.99
MULT	1.19	0.92	0.92	0.53	0.66	0.79	0.96
LOAD;ADD	2.10	1.25	1.32	0.86	0.79	0.40	0.53
LOAD;MULT	2.25	1.06	1.06	0.99	0.96	0.53	0.79

## Performance Impact

- Performance point of view :
  - » Performance-oriented scheduling improves execution time
  - » Performance improvement more evident with larger # of nodes - about 4-6% improvement
  - » Amongst energy-oriented scheduling algorithms, top-down → better most of the times.

# Energy Comparison

## ● Energy point of view :

- » Energy-oriented scheduling improves energy consumption.
- » % energy improvement increases with # of nodes.
- » Comparing top-down approach with performance-oriented and random scheduling,
  - » about 10-15% improvement.
- » Look-ahead approach → greediness restricts the energy-optimized scheduling of instructions yet to be scheduled.
- » Overall, top-down approach produced comparably better results.

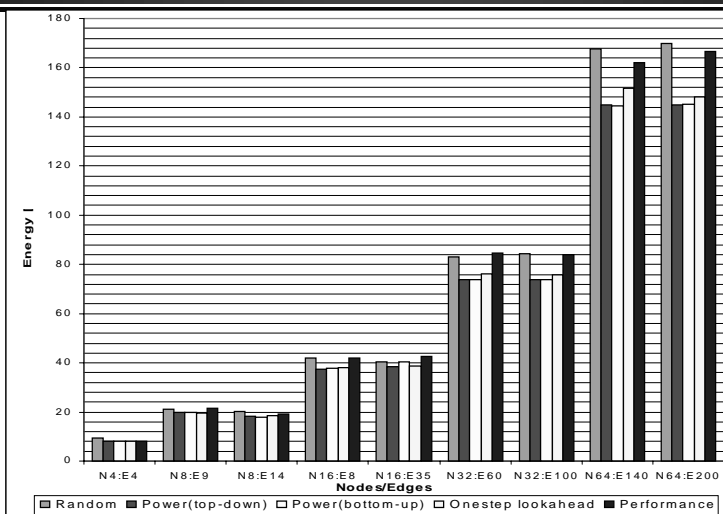
ISCA Tutorial: Low Power Design

Software.65

©MJIVN, PSU, 2000

## Results

### Energy Scheduling for Different Scheduling Algorithms



ISCA Tutorial: Low Power Design

Software.66

©MJIVN, PSU, 2000

## Chances of Improvement

- **Energy-oriented Approach:**
  - » Selection of first node to be scheduled:
    - » Very first schedulable instruction encountered.
  - » Selection in case of a tie (two or more instructions having the same circuit-state cost):
    - » Very first instruction encountered among the conflicting ones.
- **Performance-oriented Approach:**
  - » Selection in case of a tie (two or more instructions having the same delay):
    - » Very first instruction encountered among the conflicting ones.

Not the best approach !! → Unified Scheduling Algorithms

## Energy-with-Performance Scheduling

- **In top-down approach :**
  - » To select the first node to be scheduled and to resolve ties : performance into consideration.
  - » First node to be scheduled = Node with max. delay
  - » In case of ties : Node with max. delay, from among the conflicting node, selected.

## Performance-with-Energy Scheduling

- In performance-oriented scheduling:
  - » Ties resolved by considering circuit-state cost.
  - » In case of ties : node which results in minimum circuit-state cost from among the schedulable nodes is selected.

## Energy-Performance Scheduling

- Product of delay and inter-instruction cost as criteria to select next node.
- Energy\*delay → good compromise between energy & performance oriented scheduling.
  - » For first node – only performance(delay) considered because no previously scheduled instruction yet !!

# Evaluation of Scheduling Techniques

## Evaluated six different algorithms

- » Random
- » Energy-oriented (Top-Down Approach)
- » Performance-oriented
- » Energy-with-Performance
- » Performance-with-Energy
- » Energy-Performance

## Sets of input for experiments as before

- » Different sizes of DAG
- » Randomly generated DAG
- » Set of six different instructions
- » Same energy and performance costs associated with nodes

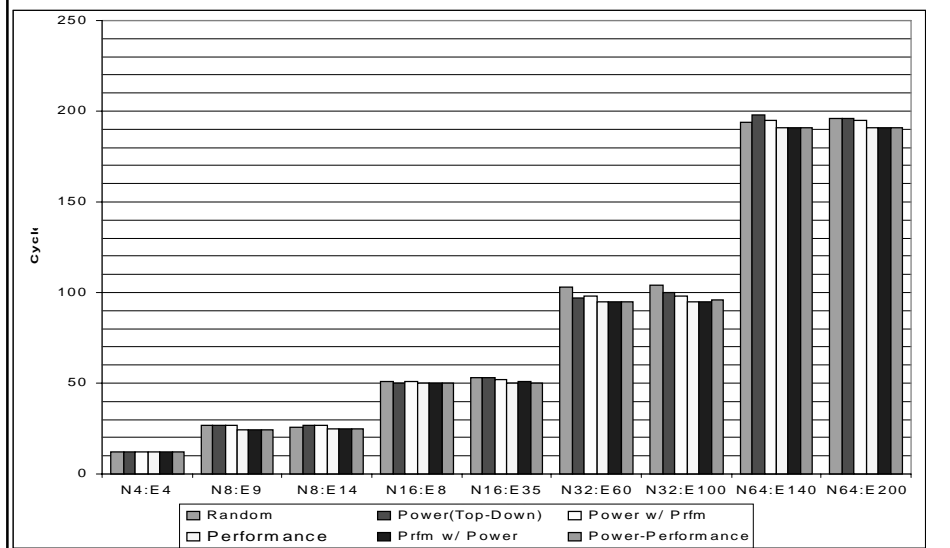
# Performance Impact

## ● Performance point of view :

- » Performance-with-energy – mostly same result as pure performance scheduling.
- » Energy-with-performance showed improvement in performance as compared to pure energy scheduling.
- » Energy-performance – same as pure performance scheduling (except one case !)

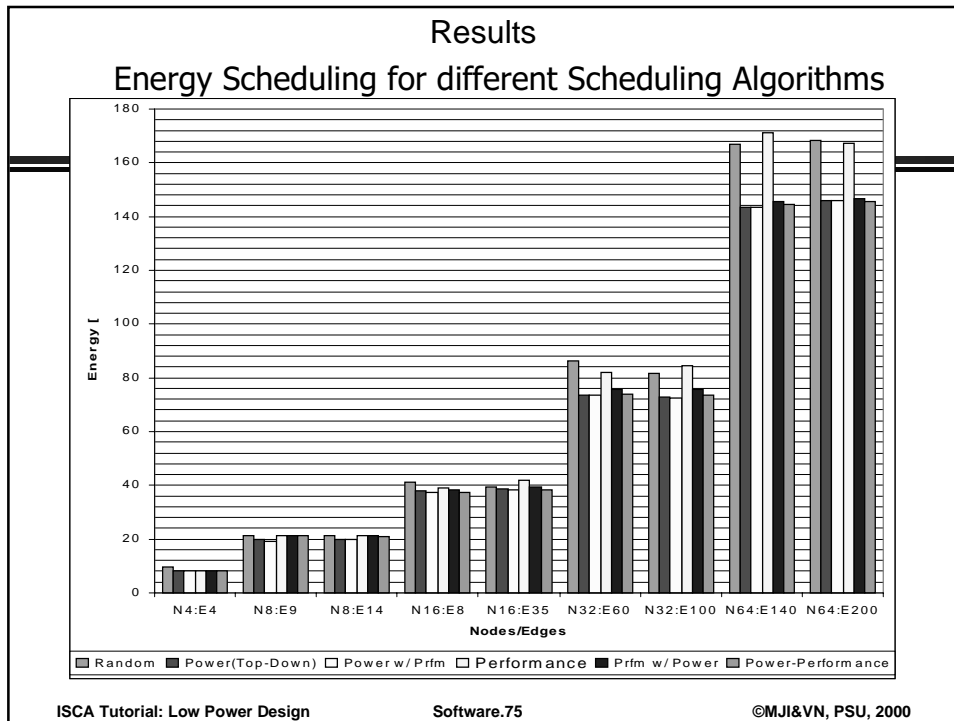
## Results

### Execution cycles for different scheduling algorithms



## Energy Consumption

- Energy point of view :
  - » Except for random and pure performance scheduling , the energy results for all other scheduling schemes are similar.
  - » Performance-with-energy performed very well as compared to pure performance scheduling : most of the times, large # of ties for selection of next node.



## Scheduling Summary

- The best scheduling from performance point of view is not necessarily the best scheduling from energy point of view.
- The unified scheduling algorithms did show improvements from performance and energy point of view as compared to their pure versions
- Need data dependencies to capture more precise information
- Need to capture interacting instructions in pipeline not just next instruction

## Scheduling for Peak Power

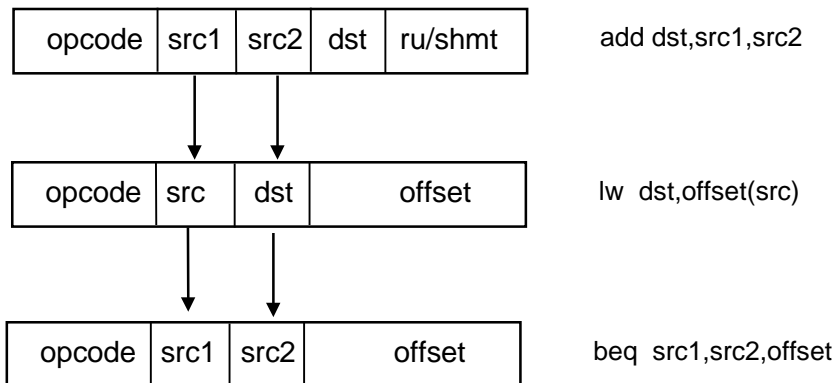
- Schedule operations in same cycles based on peak power dissipation
- Based on the units that a given instruction uses - can use activity sensitivity models

Toburen et. al.

## Register Re-labeling

- A post-compilation optimization
- Exploits corresponding fields in consecutive instructions
- Reduces bit switches on the instruction bus

## Register Relabeling Example



ISCA Tutorial: Low Power Design

Software.79

©MJIVN, PSU, 2000

## Solution Steps

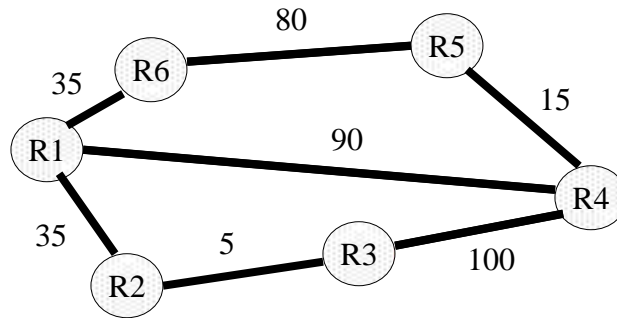
- Construct a Register Transition Graph
  - » Compiler analysis to record all consecutive transitions between all possible pairs of registers
  - » Profiling (use training inputs to create sample traces)
- Determine important paths
  - » Paths that contain edges with high transition counts
- Relabel the registers

ISCA Tutorial: Low Power Design

Software.80

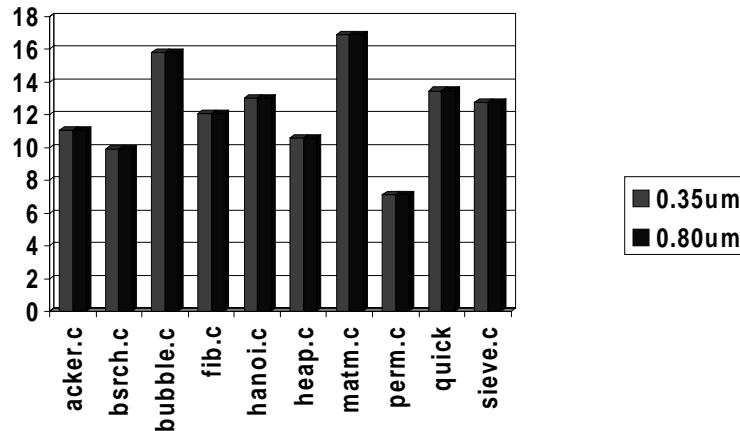
©MJIVN, PSU, 2000

## Register Transition Graph



Registers with large transition counts should be labeled using minimum Hamming distance

## Instruction Data Bus Switch Capacitance Reduction (%)



# Compiler Optimizations for Energy Reduction

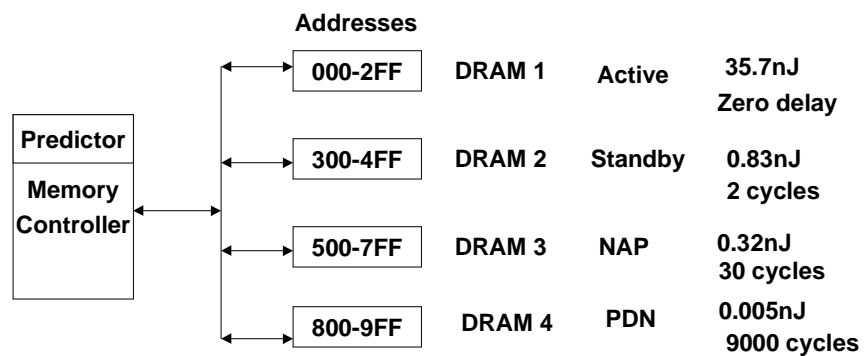
- Reduce memory energy
- Optimal selection & scheduling of machine instructions
- Exploiting low power features of the machine

ISCA Tutorial: Low Power Design

Software.83

©MJl&VN, PSU, 2000

# Memory Power Management

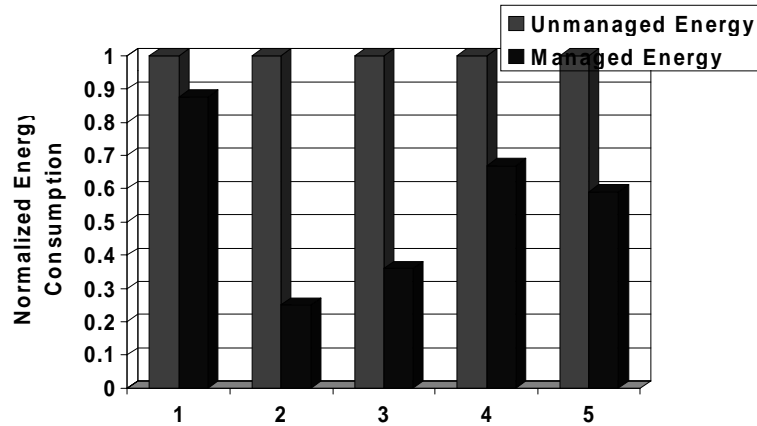


ISCA Tutorial: Low Power Design

Software.84

©MJl&VN, PSU, 2000

## Compiler Directed Turn-Off



ISCA Tutorial: Low Power Design

Software.85

©MJIVN, PSU, 2000

## Operating System Power Management

- ACPI Specification: Intel, Microsoft, Toshiba
- CPU Scheduling
- File Management - Disk Spin down
- Memory Management
- Network Interfaces

ISCA Tutorial: Low Power Design

Software.86

©MJIVN, PSU, 2000

## Advanced Configuration and Power Interface (ACPI)

- Key element in OS Directed Power Management
- Evolves existing collection of BIOS code, APM APIs etc. into a well specified power management and configuration system
- Applicable to all type of devices: mobile, handheld and desktops

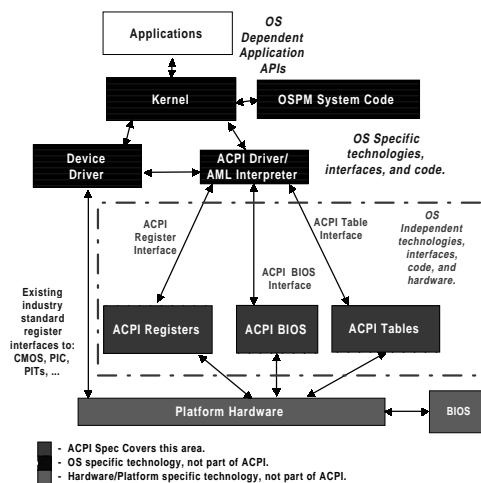
## Goals of ACPI and OSPM

- Enable all PCs to implement motherboard configuration and power management functions, using appropriate cost/function tradeoffs.
- Enhance power management functionality and robustness.
  - » Gathering power management information from users, applications, and the hardware together into the OS.

# Goals of ACPI and OSPM

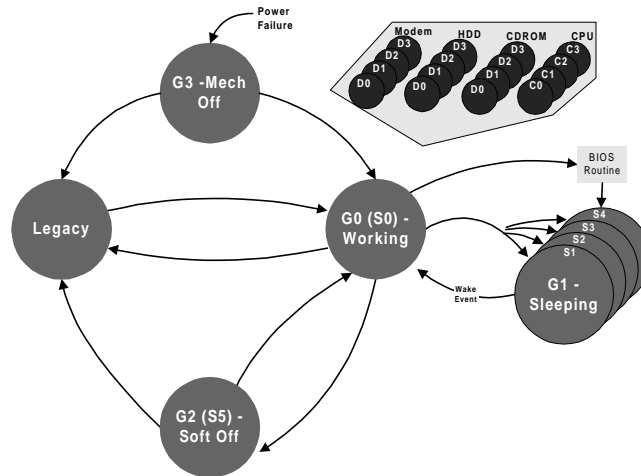
- Allows inexpensive power managed hardware to support very elaborate power management policies.
- Unification of power management algorithms in the OS will reduce opportunities for miscoordination and will enhance reliability

# ACPI Overview



ACPI

# Power States



ACPI

ISCA Tutorial: Low Power Design

Software.91

©MJIVN, PSU, 2000

# Criteria Influencing State Selection

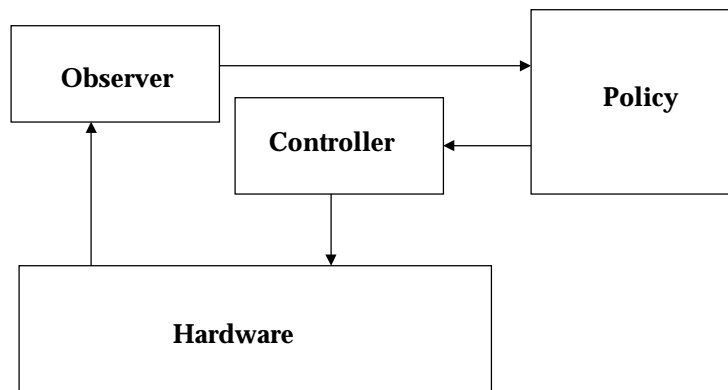
- Power consumption - how much power the device uses.
- Device context - how much of the context of the device is retained by the hardware.
- Device driver - what the device driver must do to restore the device to full on.
- Restore latency - how long it takes to restore the device to full on.

ISCA Tutorial: Low Power Design

Software.92

©MJIVN, PSU, 2000

# Power Manager



Benini 98

ISCA Tutorial: Low Power Design

Software.93

©MJl&VN, PSU, 2000

# Policy Optimization

- Greedy: Turn-off device as soon as they are inactive
  - » Problem 1: How to choose most appropriate state when multiple power modes are available
  - » Problem 2: Finite wake-up times and consequent performance penalty
  - » Solution: Need adaptive techniques => Need system monitoring capability (observer)

ISCA Tutorial: Low Power Design

Software.94

©MJl&VN, PSU, 2000

## System Monitoring

- Information Required
  - » Inter-arrival times of requests
  - » Service times of request
- Obtain time-stamped events
  - » (t,r,s) - where t is time stamp, r is resource identifier and s is the resource-dependent identifier of the type of event
  - » Modify device drivers of resources
  - » Monitor kernel activity for CPU usage

## Observer Requirements

- Low perturbation of normal system
- Flexibility
  - » Number and type of observed resources should be dynamically controllable
- Accuracy
  - » Time-Stamping events with high-accuracy

## Power Management Policies

- Processor - Dynamic Voltage Scaling and Speed Setting
- Disks - Spin down and reliability concerns
- Memory - Selective Shutdown of Components
- Transceivers - Quality, distance and Channel Conditions

ISCA Tutorial: Low Power Design

Software.97

©MJIVN, PSU, 2000

## Dynamic Voltage Scaling

- Slow Down Processor to Fill Idle Time
- More Delay => Lower Operational Voltage
- Runtime Scheduler determines processor speed and selects appropriate voltage
- Transitions delay for frequencies  $\sim 150\mu\text{s}$
- Potential to realize 10x energy savings



ISCA Tutorial: Low Power Design

Software.98

©MJIVN, PSU, 2000

## Scheduling for Variable Voltage

- Real-time tasks: Average Power (AP) = 1W

task	Arrival	Deadline	Cycles
A	0	6	5
B	5	20	5

- Partial System Shutdown:  $AP = 0.5\text{ W}$



- Voltage Scheduling:  $AP = 0.28\text{ W}$



## Speed Setting Algorithms

- PAST - Faster when busy, slower when Idle
  - » run cycles - number of non-idle cycles in previous interval
  - » idle cycles - number of idle CPU cycles
    - Hard (e.g. disk wait): Scheduling cannot help
    - Soft (e.g. network request packet)
  - » excess cycles - cycles left over from previous interval because of slow execution

## Prediction Schemes

- PAST: Predicts next interval is as busy as last completed interval.
- LONG\_SHORT: Lookup the last 12 run\_percents. 3 most recent values constitute short-term and the remaining 9 constitute long-term. Our prediction is for the next cycle to be weighted average of these values  
$$0+.3+.5+1+1+1+.8+.5+.3+4(.1+0+0)/(9+4(3)) = 0.28$$

ISCA Tutorial: Low Power Design

Software.101

©MJIVN, PSU, 2000

## Observations

- Energy Savings depend on interval between speed adjustments
  - » too fine grain: less power saved if usage is bursty
  - » too coarse grain: excess cycles built up during slow interval - affects response time
  - » having too low a speed results in less energy efficient schedules (1.0V instead of 2.2V) - due to excess cycles

ISCA Tutorial: Low Power Design

Software.102

©MJIVN, PSU, 2000

## Low Power File System

- Predictive shut down of disks
  - » Spin mode consumes 1 Watt
  - » Sleep mode consumes 25 milliWatts
- Latency Constraints
  - » Seek latency: 10 milliseconds
  - » Spinup Latency: 2 seconds
- OS must balance between the need for low latency file access and need to reduce disk power

Kester

## Requirements for a Low Power File System

- Fine-grained disk spindown
- Whole-file prefetching cache
- 8-16MB of low power, low read latency memory

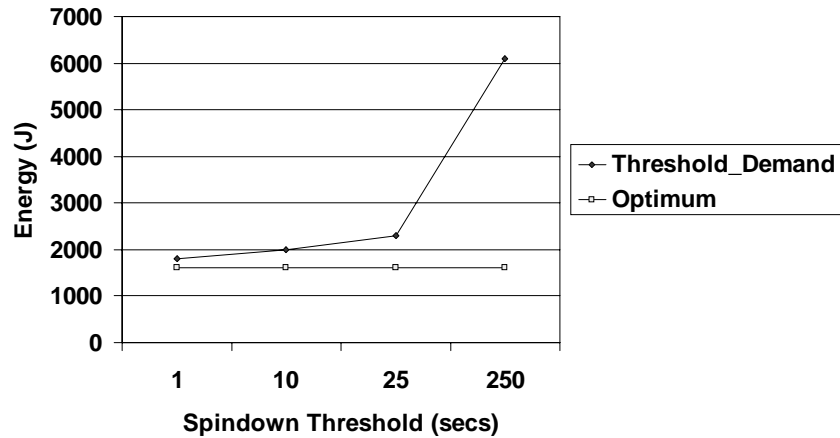
## Factors Influencing File System Design

- Number of Disk Spinups affects reliability
  - » friction induced wear
- Disk Spinups is a function of
  - » read/write inter-request time
  - » disk spindown delay (energy consumed during spindown and spinup can keep disk spinning for x seconds)

## Spindown Prediction Techniques

- Threshold\_Demand: Spun down after a fixed period of inactivity and is spun up upon the next access
- Optimum: Offline algorithm - Oracle

## Influence of Spindown Thresholds

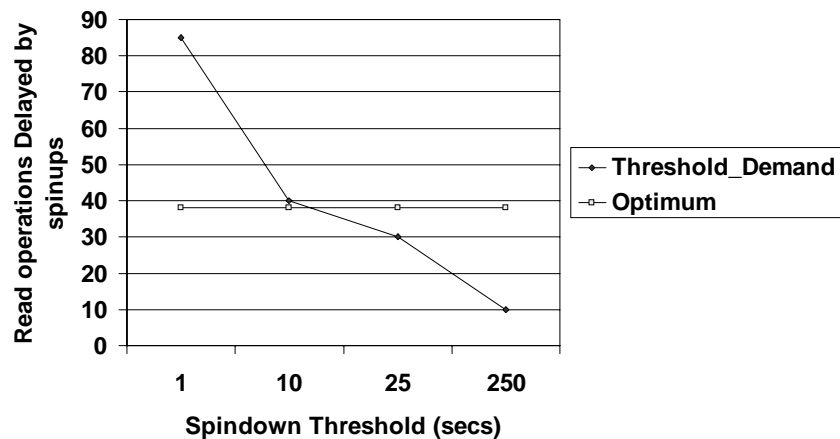


ISCA Tutorial: Low Power Design

Software.107

©MJIVN, PSU, 2000

## Influence of Spindown on Delays on Read Accesses



ISCA Tutorial: Low Power Design

Software.108

©MJIVN, PSU, 2000

## Other Techniques

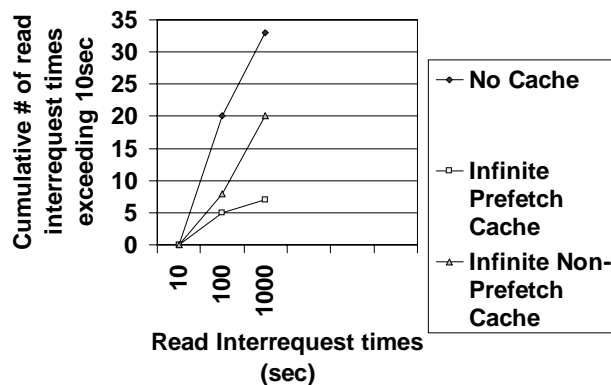
- Predictive Techniques
  - » Uses information about previous access
- Stochastic Learning Techniques
- Use of Buffers
  - » To batch writes
  - » Prefetch all blocks of an open file
    - eliminates a large number of inter-request times that are large
    - permits fine grain spin-down without access penalty

ISCA Tutorial: Low Power Design

Software.109

©MJIVN, PSU, 2000

## Influence of Prefetching and Caches on Inter-request Times



ISCA Tutorial: Low Power Design

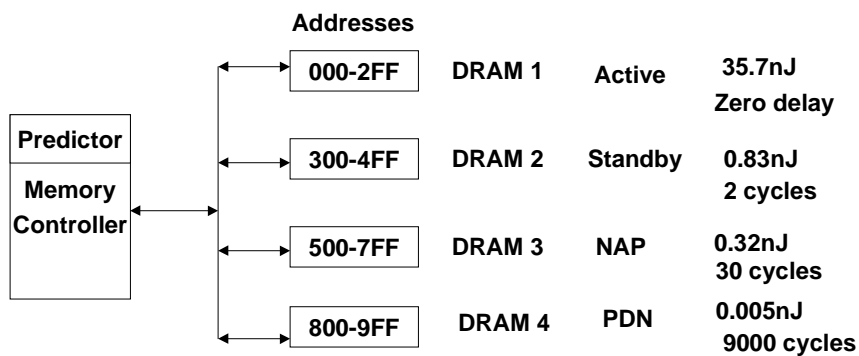
Software.110

©MJIVN, PSU, 2000

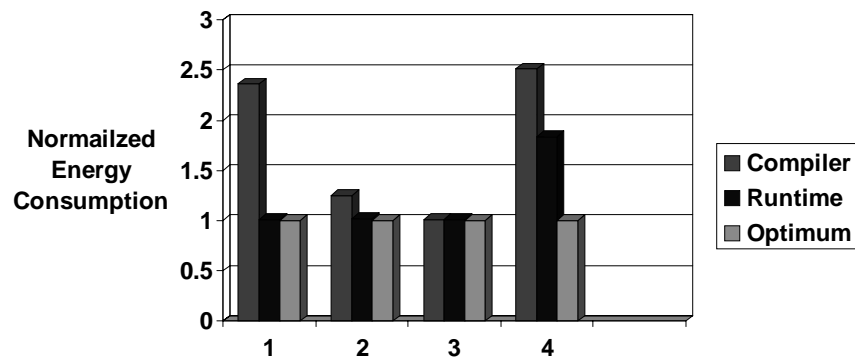
## Size of Prefetch Buffer

- Powering up 8MB of DRAM for a prefetch cache consumes as much energy as a spinning disk !!
- Prefetching requires small, low power buffer
- Use Flash Memory
  - » Reads: 325 milliwatts
  - » Problem: Slow write latencies

## Memory Power Management



## Predictive Dynamic Turnoff



ISCA Tutorial: Low Power Design

Software.113

©MJIVN, PSU, 2000

## Other Memory Management Issues

- Virtual Memory Management
  - » Memory energy is dominated by access count than size
  - » Accesses increase due to allocation and deallocation also.
  - » Use policies based on whether allocation/deallocation dominate or actual accesses dominate
- Data Structure Choices: Arrays, Linked Lists, Doubly Linked Lists, etc.

ISCA Tutorial: Low Power Design

Software.114

©MJIVN, PSU, 2000

## Key References

- F. Catthoor et. al. Global communication and memory optimizing transformations for low power signal processing systems. In Proc. the IEEE Workshop on VLSI Signal Processing, pages 178-187, 1994.
- P. R. Panda et. al. Architectural exploration and optimization of local memory in embedded systems. In Proc. ISSS'97, Antwerp, Sept 1997.
- W-T. Shiue et .al. . Memory exploration for low power, embedded systems. In Proc. DAC'99, New Orleans, Louisiana, 1999.
- F. Catthoor et. al. Custom memory management methodology -- exploration of memory organization for embedded multimedia system design. Kluwer Academic Publishers, June, 1998.
- S. Y. Liao. Code Generation and Optimization for Embedded Digital Signal Processors. Ph.D. Thesis, Dept. of EECS, MIT, Cambridge, Massachusetts, June 1996.
- V. Tiwari et. al. Instruction Level Power Analysis and Optimization of Software, Journal of VLSI Signal Processing Systems, Vol. 13, No. 2, August 1996.

## Key References

- M. C. Toburen et. al. Instruction scheduling for low power dissipation in high performance processors. In Proc. the Power Driven Micro-architecture Workshop in conjunction with the ISCA'98, Barcelona
- H. Y. Kim et. al. Multiple Access Caches: Energy Implications, In Proc. IEEE CS Annual Workshop on VLSI, pp. 37-42, April 27-28, 2000, Orlando, FL
- A. Parikh et. al. Instruction Scheduling Based on Energy and Performance Constraints, In Proc. IEEE CS Annual Workshop on VLSI, pp. 53-58, April 27-28, 2000, Orlando, FL
- M. Kandemir et. al. Influence of compiler optimizations on system power, In Proc. the 37th Design Automation Conference (DAC'00), Los Angeles, California USA, June 5-9, 2000.
- V. Delaluz et. al. Memory Energy Management Using Software and Hardware Directed Power Mode Control, Technical Report CSE-00-004, Dept. of CSE, Penn State Univ.

# Key References

- ACPI - Advanced Configuration and Power Interface  
<http://www.teleport.com/~acpi>
- Kester Li. Towards a low power file system. Technical Report UCB-CSD-94-914, University of California at Berkeley, 1994.
- K. Govil et. al., "Comparing algorithms for dynamic speed-setting of a low-power CPU," in Proc. ACM Int'l Conf. on Mobile Computing and Networking, pp. 13--25, Nov. 1995.
- F. Douglass et .al., "Thwarting the Power Hungry Disk," Proceedings of the 1994 Winter USENIX Conference
- M. Weiser et. al., Scheduling for Reduced CPU Energy, Proc. 1<sup>st</sup> USENIX OSDI, pp. 13-23, Nov 1994.
- L. Benini et. al., Monitoring System Activity for OS-Directed Dynamic Power Management, Proc. ISLPED, pp. 185-190, 1998.
- A. Chandrakasan et.al. Design of a low-power wireless camera, IEEE CS Annual Workshop of VLSI, 1998
- K. Roy et .al. Low Power Design Methodologies for Systems-on-Chips, 12th International Conference on VLSI Design, 1998