Energy-Aware Compiler Optimizations

Y.N. Srikant
Department of
Computer Science and Automation
Indian Institute of Science
Bangalore
srikant@csa.iisc.ernet.in

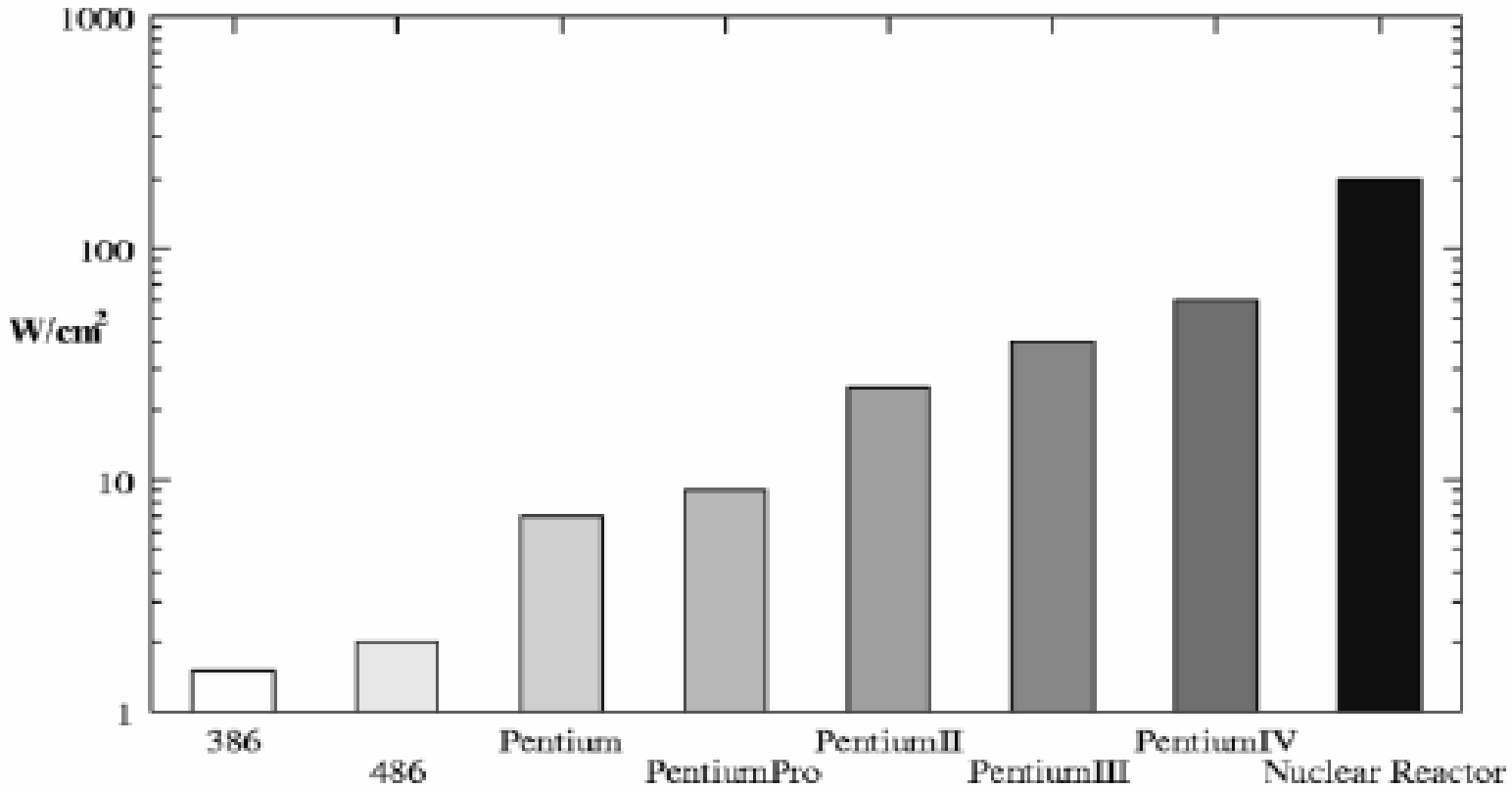
Outline of the talk

- Introduction
- Energy models
- Dynamic voltage scaling (DVS)
- Leakage energy optimization via instruction scheduling
- INTACTE: A tool for modeling interconnects

Techniques to achieve energy savings at various levels

Classification		Techniques		
Software	Application	Algorithm Design	Scheduling	Leakage Reduction
	Virtual Machine	Resource Hybernation	Memory Management	
	Compiler	Fidelity Reduction	Energy Accounting	
	Operating System	Remote Execution		
Hardware	Architecture	Clock, Memory, and Interconnect Optimizations		
	Gates	Technology Mapping	Gate Restructuring	
	Transistors	Transistor Sizing	Transistor Ordering	

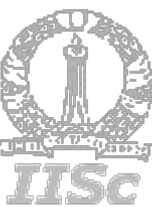
Importance of processor-level energy-aware design



Power density of Intel chips

Energy-aware compilation

- CPU voltage/frequency scaling
- Reduction of CPU bus switching
- CPU function unit voltage/clock gating
- Loop tiling
- Partitioning data cache so as to permit efficient caching
- Cache reconfiguration
- Shutting down unused memory banks



CMOS Device-level Power dissipation basics

$$PW_{device} = (\tfrac{1}{2})\ C\ V_{DD}\ V_{swing}\ a\ f + I_{leakage}\ V_{DD} + I_{sc}\ V_{DD}$$

- First factor is dynamic power dissipation and is currently dominant
- Second factor is static power dissipation and is expected to increase dramatically with shrinking device sizes
- Third factor is short-circuit power and can be controlled only by superior technology

The Cube-root rule (1)

- Assuming, C as a constant (for a given design), worst case activity (a=1), a single voltage and frequency for the whole chip, and that $f = kV$

  $PW_{chip} = K_v V^3 = K_f f^3$

  where K_v and K_f are design-specific constants

The Cube-root rule (2)

- This implies that voltage (hence frequency) reduction is the single most efficient method for reduction of dynamic power dissipation
- However, V_{DD} cannot be reduced beyond a limit
- Hence, voltage scaling combined with other techniques need to be employed to reduce power consumption

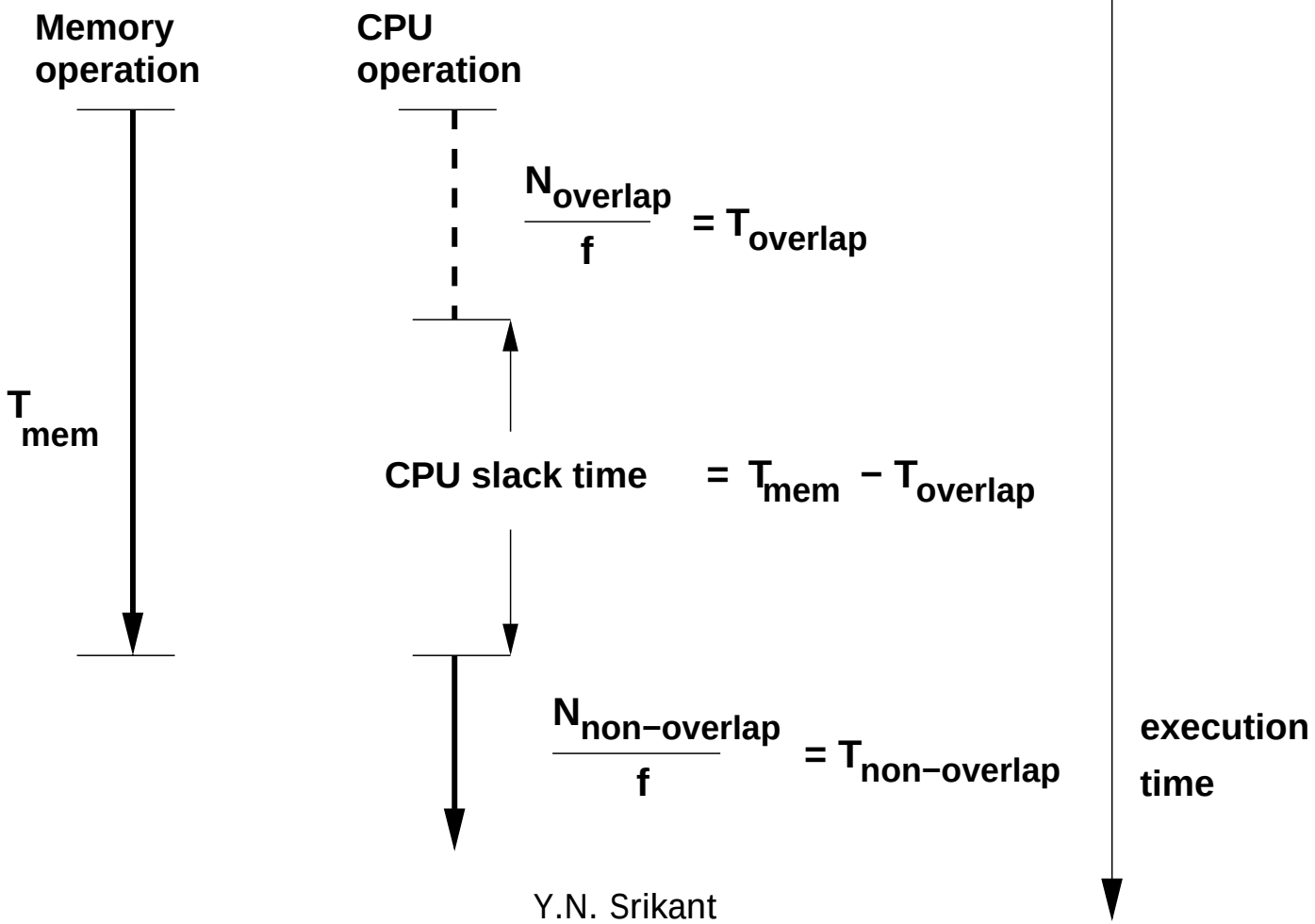
Compiler-Based Dynamic Voltage Scaling

- For memory bound programs
  - based on profiling
- Loop scheduling
  - based on loop rotation
  - suitable for multi-core architectures

Compiler DVS for Memory Bound Programs – Basic Idea

- During CPU stall (awaiting completion of memory operations)
  - scale down CPU voltage and freq
  - Save energy without performance degradation
- Memory operations are assumed to be asynchronous
- 0% – 25% energy savings with 0% - 3% performance loss

Compiler DVS – CPU Slack



DVS – An Algorithm

- Partition program into "regions" based on energy consumption at different (V,f)
  - 2 regions: one at a lower frequency and the other at f_{max}
  - Introduce frequency-changing instructions at the entry and exit of the (lower freq) region
- Finding best partitions is an optimization problem
- Time for frequency change: 100 memory accesses (10-20 μs)

Dynamic Voltage Loop Scheduling

- Repeatedly regroup a loop based on rotation scheduling
- Decrease the energy by DVS as much as possible within a timing constraint
- Not necessarily for memory bound programs

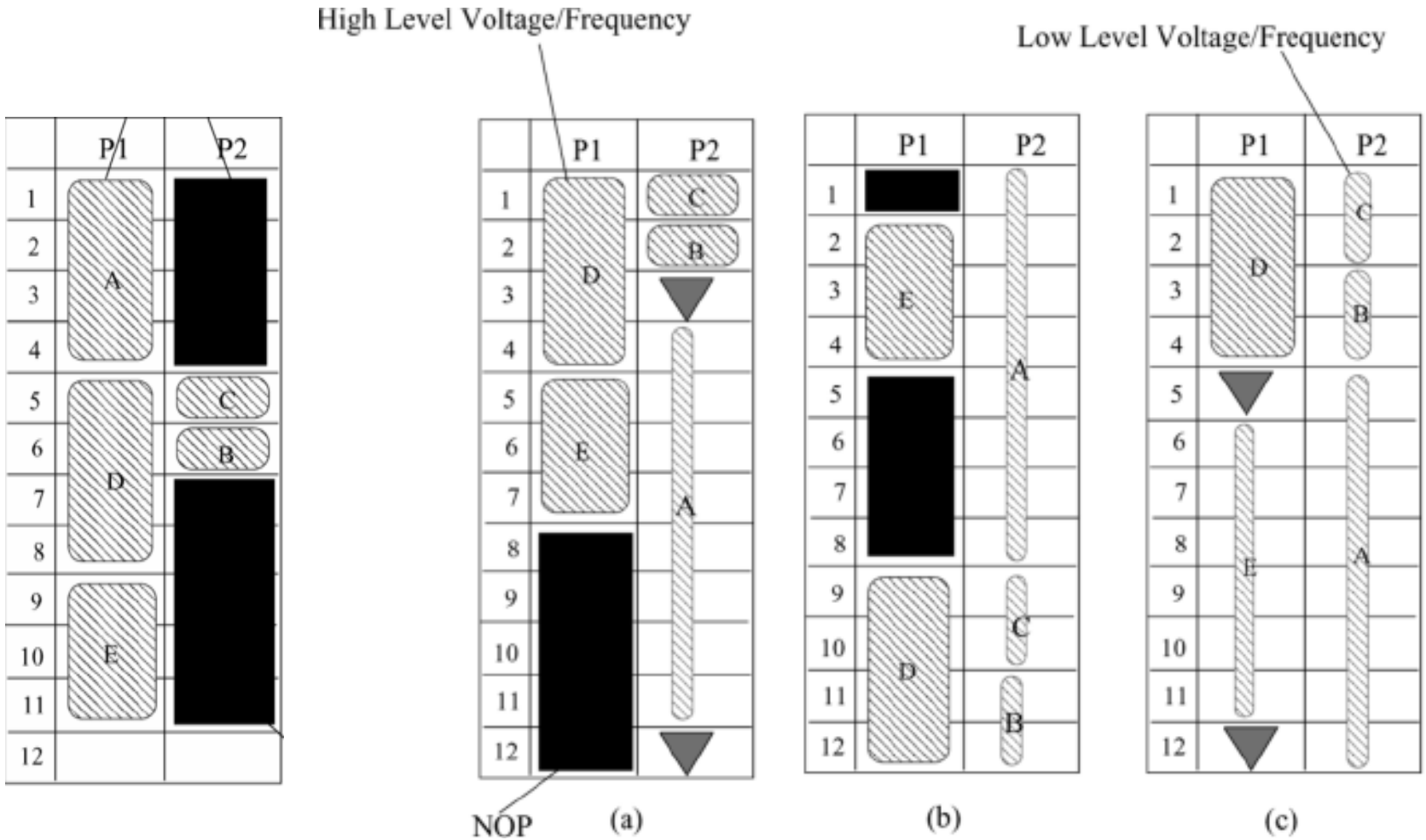
Original Loop

```
E[0]=E[−1]=E[−2]=1;
for ( i=1; i<=N; i++){
    A[i]=E[i−3]*E[i−3];
    B[i]=A[i]+1;
    C[i]=A[i]+3;
    D[i]=A[i]*A[i];
    E[i]=B[i]+C[i]+D[i];
}
```

Rotated Loop

```
E[0]=E[−1]=E[−2]=1;
A[1]=E[−2]*E[−2];                    Prologue
for ( i=1; i<=N−1; i++)
{
    B[i]=A[i]+1;
    C[i]=A[i]+3;                     Loop
    D[i]=A[i]*A[i];                  Body
    E[i]=B[i]+C[i]+D[i];
    A[i+1]=E[i−2]*E[i−2];
}
B[N]=A[N]+1;
C[N]=A[N]+3;
D[N]=A[N]*A[N];                      Epilogue
E[N]=B[N]+C[N]+D[N];
```

Figures from:Shao, et al.,
IEEE TC&S, May 2007, p445-9



Original Schedule

Schedule after first rotation

Schedule after second rotation

Schedule after third rotation

Figures from:Shao, et al., IEEE TC&S, May 2007, p445-9

DVS – Research Issues

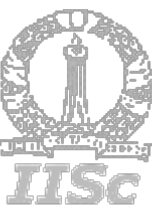
- DVS in multi-core and multiple clock domain architectures
- DVS for speculative execution architectures
- DVS for interconnection networks?

Motivation for Leakage Energy Optimization – (1)

- Leakage energy is the static dissipation energy in CPU, cache, etc.
  - The FUs are in active state, but are not doing any useful work
- With 70 nm technology, leakage energy consumption will be on par with dynamic energy consumption

Motivation for Leakage Energy Optimization – (2)

- Dual-threshold domino logic with sleep mode can facilitate fast transitions between active and sleep modes without performance penalty and moderate energy penalty
  - Can put ALU into low leakage (sleep) mode after even one cycle of idleness
- IALUs are idle for 60% of the time (on the average)



Motivation for Leakage Energy Optimization – (3)

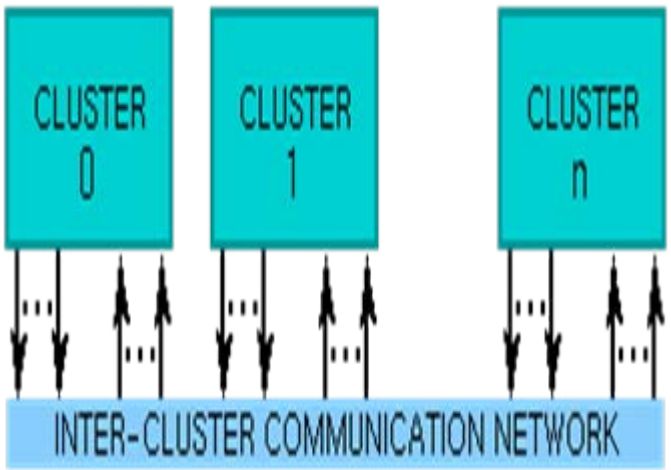
- Pure hardware scheme
  - has 26% energy overhead over ideal scheme (no overhead)
  - frequent transitions between active and sleep states
- A software-based scheme aids the hardware and together they save more energy with little performance loss

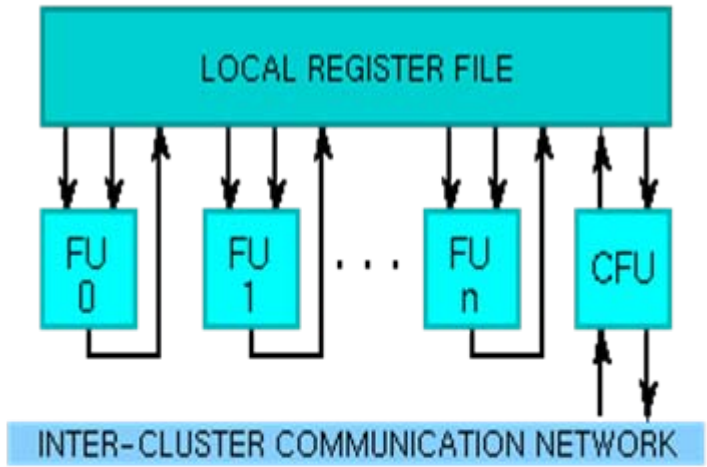
Energy-aware instruction scheduling

An integrated energy-aware instruction scheduling algorithm for clustered VLIW architectures:

- Reduces #transitions between active and sleep states and increases the active/idle periods
- Reduces the total energy consumption of FUs
- Generates a more balanced schedule which helps to reduce the peak power and step power

Clustered VLIW Architectures



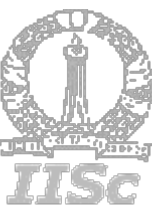


An Individual Cluster

FU Function Unit
CFU Communication Function Unit

The scheduling algorithm for clustered VLIW

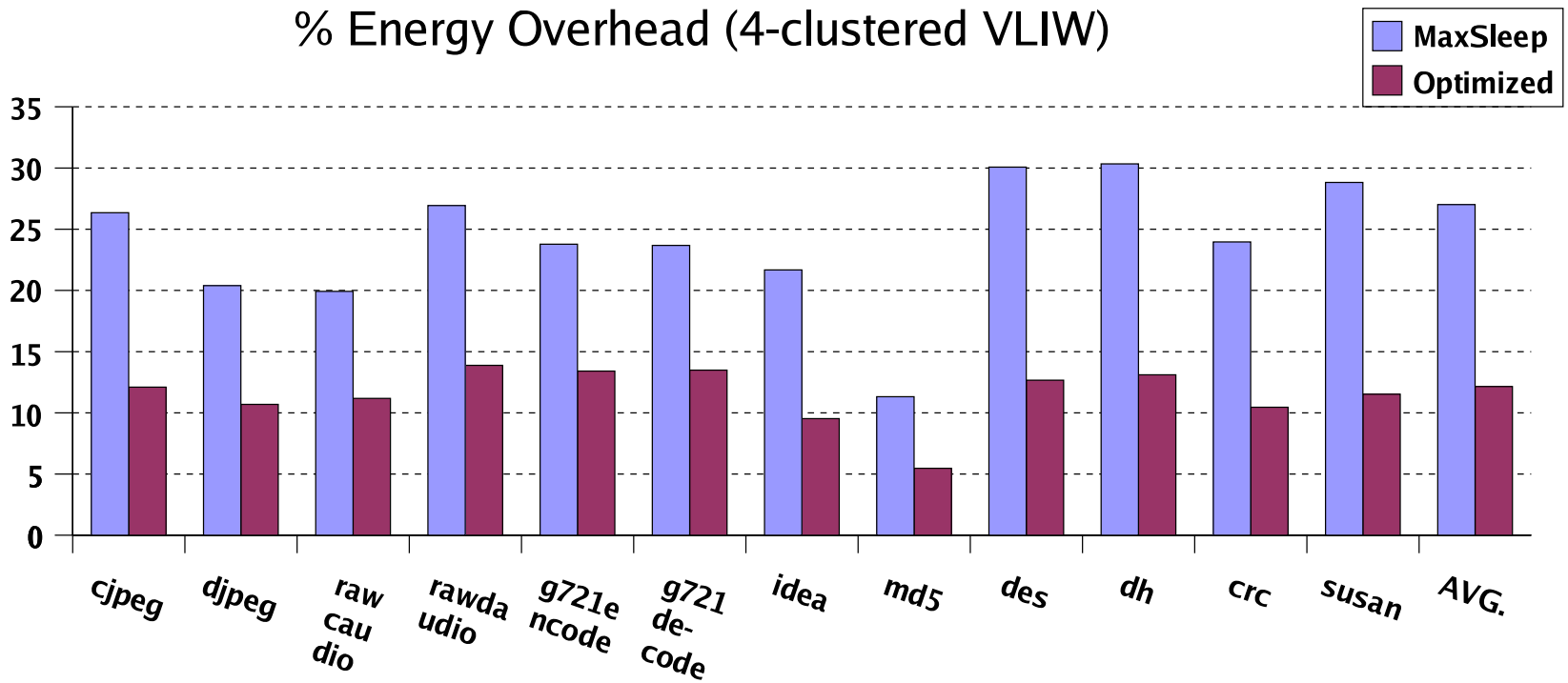
- Makes cluster assignment decisions during temporal scheduling
- Basic block scheduler using list scheduling
- Three main steps
  - Prioritizing the ready instructions
  - Assignment of a cluster to the selected instruction
  - Assignment of an FU to the selected instruction in the assigned target cluster



Results

- Comparison with hardware-only schemes
- #Transitions reduce on the average by 58.29% (4-clusters)
- Average reduction in energy overhead is 16.92% (4 clusters)

Energy Overhead (4-clusters) w.r.t No-overhead Scheme



Heterogeneous Interconnects

- An interconnect composed of two sets of wires
  - one set optimized for latency and another optimized for energy
  - less area than two sets of low latency wires
  - Instr. scheduling can help to reduce energy but maintain performance

Exploiting Heterogeneous Interconnects

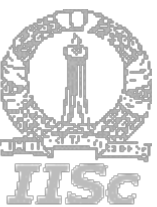
- Selectively mapping communication to the appropriate interconnect
  - urgent communications
    - low latency (high energy) path
  - non-urgent communications
    - high latency (low energy) path
  - identify urgent comm. using comm. slack (60.88% of comm. have 3-cycle slack)
  - Increase in execution time is 1.11% and reduction in comm. energy is 39% (both for a 4-cluster processor)

Research Issues in Leakage Energy Optimization

- Cache reconfigurations
- Memory bank control

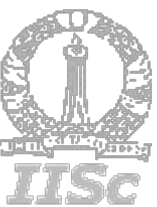
INTACTE: An Interconnect Area, Delay, and Energy Estimation Tool

- Interconnects can consume power equiv. to one core, area equiv. to three cores, and delay can account for 0.5 of L2 cache access time
  - Can be a major source of performance bottleneck
- We present an interconnect modeling tool
  - Enables co-design of interconnects with other architectural components



INTACTE – What?

- Interconnect microarchitecture exploration tool to estimate
  - Delay
  - Power
- Given the technology, area, clock frequency and latency
  - for point to point interconnect
- Analogous to CACTI



INTACTE – How?

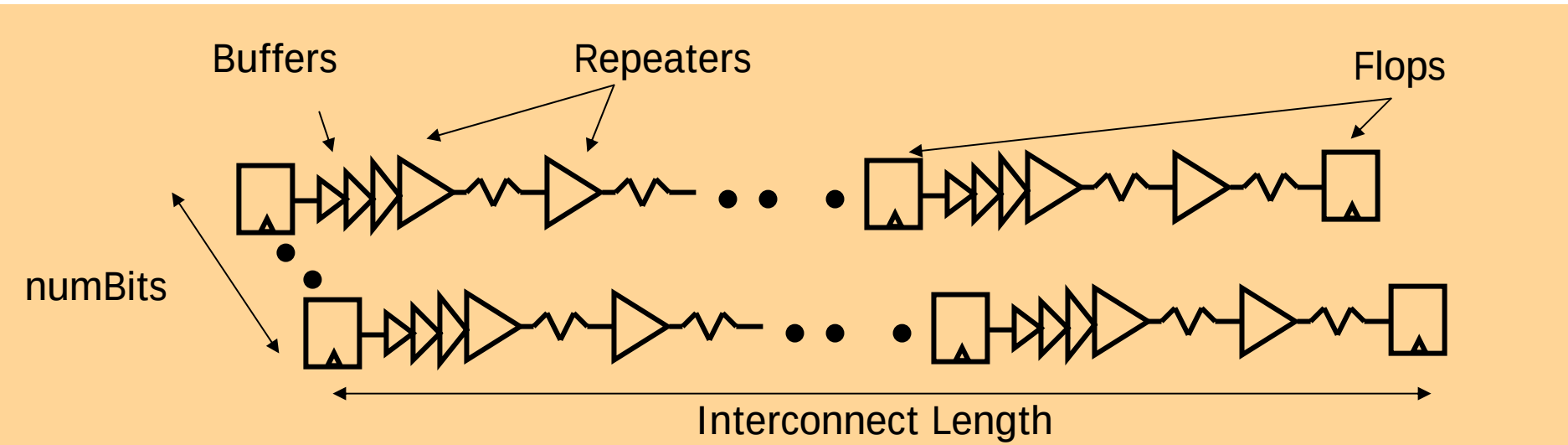
- Solves an optimization problem of minimizing power by finding the optimal values for
  - Wire width
  - Wire spacing
  - Repeater size
  - Repeater spacing

INTACTE - More

- Additional design variables - can be either constraints or determined by the tool
  - Area
  - Pipelining
- Voltage Scaling Support
  - Tool optimizes power and delay for nominal (Maximum) supply
  - Power and Delay numbers reported
    - for 32 different voltage levels separated by 15 mV from the nominal values

INTACTE Tool description (1)

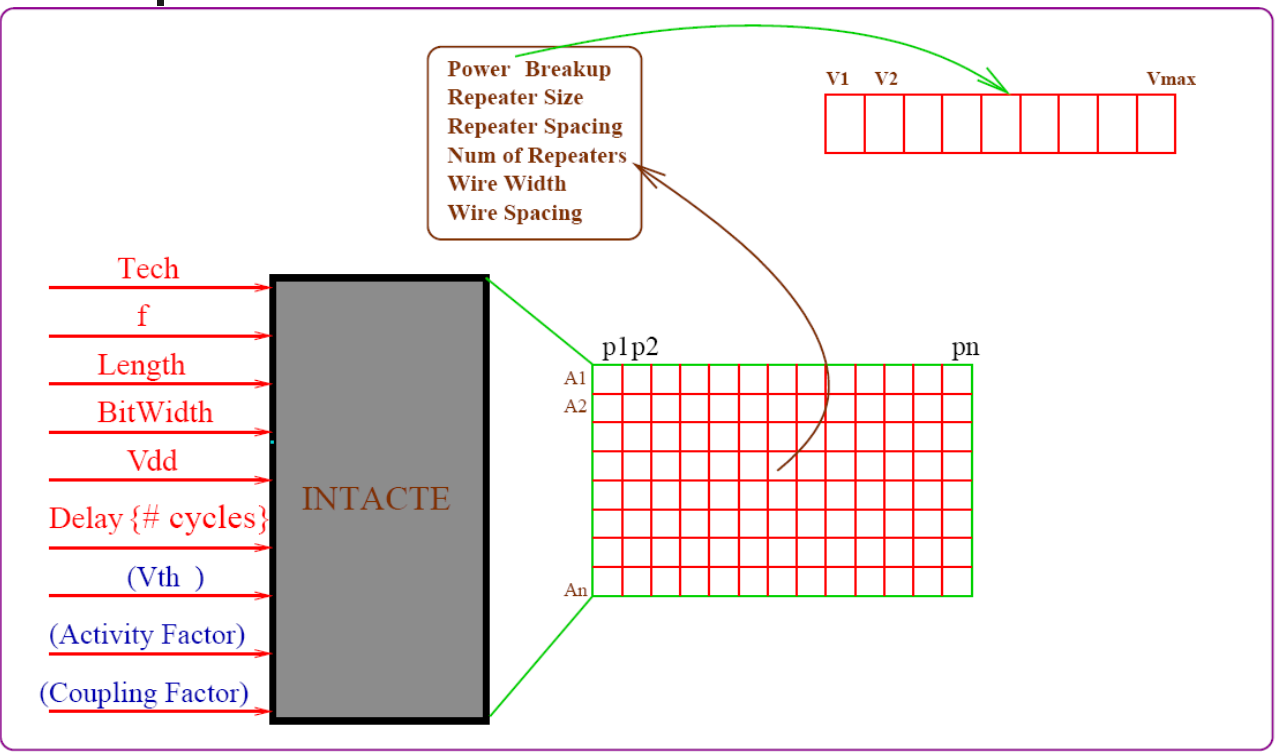
- The tool models the interconnect as consisting of a set of identical, equal length pipeline stages
- Each stage starts with a Flop driving a repeater through a set of buffers followed by equally spaced wire-repeater sections.
- All parameters for the model are taken from detailed HSPICE simulations and ITRS



INTACTE Tool description(2)

- The parameters related to the flops, repeaters, wires and buffers are pre-computed for 4 different technology nodes (90, 65, 45 and 32nm) and 32 different supply voltages.
- For each iteration of optimization, the tool computes the power and delay for each wire-repeater section.
- These values are multiplied by #repeaters and degree of pipelining and added to the pipelining overhead to get the overall power and delay numbers.
- This reduces the size of the search space

Block Diagram of INTACTE



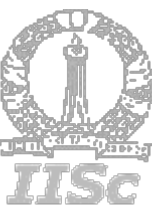
- Tool Inputs
  - Technology
    - 90,65, 45 & 32nm
  - Clock frequency
  - Length of interconnect
  - Bit width
  - Supply
  - Delay (in cycles)
  - Activity Factor
  - Coupling Factor
- Tool Outputs
  - Power, Delay versus
    - Area, Pipelining, Supply

Experimental Results

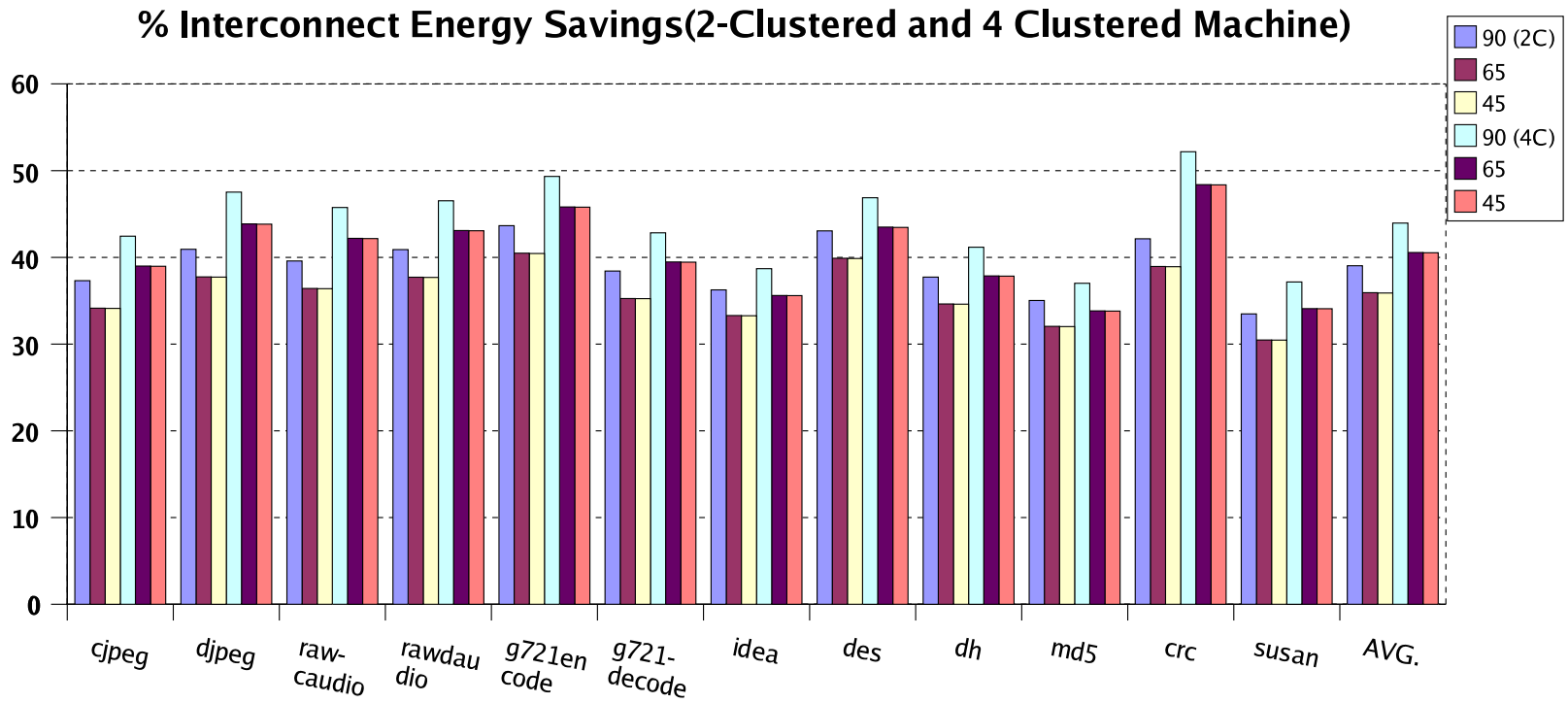
- Demonstrate the accuracy of the tool
  - various trends in interconnect power and performance have been exhibited
  - detailed HSPICE simulations have been carried out to validate the results.

Architectural Tradeoff Evaluation

- Architectural tradeoffs in having two heterogeneous wires can be evaluated using our tool
- Architect provides length, no. of bits, target technology, operating voltage, and delay estimates
- Tool provides a set of possible interconnect design options to choose from



Interconnect Energy Savings



References

1. C-H. Hsu and U. Kremer, The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction, PLDI 2003.
2. Z. Shao, *et al.*, Real-Time Dynamic Voltage Loop Scheduling for Multi-core Embedded Systems, IEEE Tr.Circuits & Systems, May 2007.

References

3. K.Ananda Vardhan and Y.N. Srikant, Transition-aware Scheduling: ..., Computing Frontiers, 2005.
4. Rahul Nagpal, and Y.N. Srikant, Compiler Assisted Leakage Energy Optimization for Clustered VLIW Architectures, EMSOFT, October 2006.
5. Rahul Nagpal, and Y.N. Srikant, Exploring Energy-Performance Tradeoffs for Heterogeneous Interconnect Clustered VLIW Processors, HiPC, December 2006.
6. Rahul Nagpal, Arvind Madan, Bharadwaj Amrutur, and Y.N. Srikant, INTACTE: An Interconnect Area, Delay, and Energy Estimation Tool for Microarchitectural Explorations, CASES, October 2007.

Thank You