Balancing Resiliency and Energy Efficiency of Functional Units in Ultra-low Power Systems

M. S. Golanbari, A. Gebregiorgis, E. Moradi, S. Kiamehr, **M. B. Tahoori**

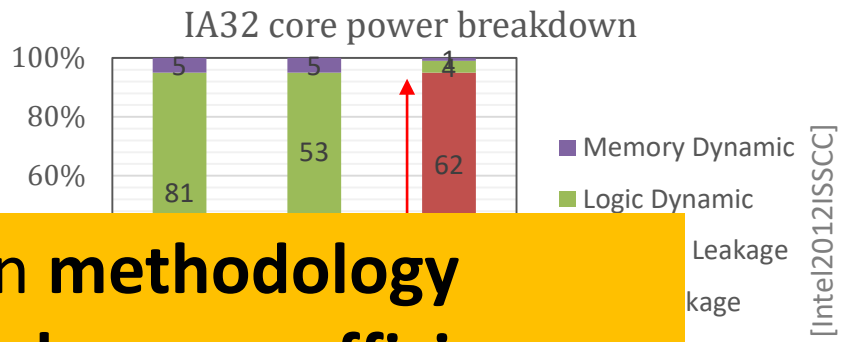
INSTITUTE OF COMPUTER ENGINEERING (ITEC) – CHAIR FOR DEPENDABLE NANO COMPUTING (CDNC)

Motivation

- Growing demand for energy efficient computing
  - **Better energy efficiency** and **higher reliability** desired

- Aggressive voltage scaling to reduce energy/power consumption
  - ✓ Best energy efficiency at Near-Threshold Voltage regime (NTV) ☺
  - ➢ Increased sensitivity to variations → reliability issues ☹
  - ➢ Large leakage power contribution at lower supply voltages ☹

- Circuit designs need to be revisited
  - Reduce the circuit idle time:
  - N



[Intel2012ISSCC]

We present a design **methodology** to improve **reliability and energy efficiency** of large **functional units**

Proposed method

- Leverage cross layer information for improving functional unit:
  - Energy efficiency and reliability
- We partition a functional unit into smaller/faster units based on:
  - Instruction pattern usage, similarity, delay analysis
  - ✓ Improve performance or reliability
    - Smaller & faster ALUs → more timing margin or better performance
  - ✓ Better energy efficiency by aggressive power gating of inactive ALUs
- Design time method: Find best ALU partitioning
  - Analyze the instruction flow of the ALU
  - Using machine-learning based clustering techniques
- Our case study: 64-bit ALU

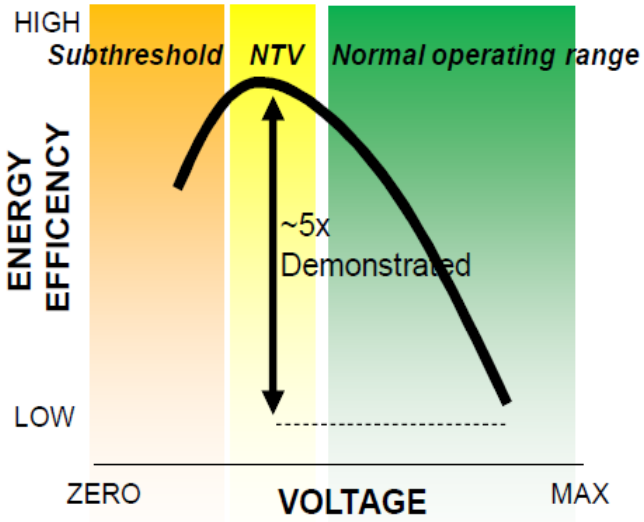
```
1453541: lda     r16,-32558(r16)
         jsr     r26,(r27)
         ldah    r29,78(r27)
         lda     r29,3152(r29)
         lda     r30,-208(r30)
         and     r16,16,r1
         bis     r31,r16,r13
         bis     r31,r18,r14
         bis     r31,r17,r15
         ldah    r2,16(r31)
         addl    r2,2,r2
         and     r16,r2,r1
         zapnot  r2,15,r2
         zapnot  r1,15,r1
         cmpeq   r1,r2,r1
         cmpult  r31,r1,r1
         lda     r1,1(r1)
         addl    r31,r15,r2
         lda     r1,4224(r31)
         lda     r2,1(r31)
         and     r13,r1,r1
```

Outline

- Background and related work
- Instruction pattern analysis
- Instruction clustering and ALU generation
- Energy, performance, and reliability improvement results
- Summary

Near-Threshold Computing

- Aggressive voltage scaling down to V_{th} (NTC)
  - High energy efficiency ($\sim 10\times$)
  - Enough performance for many application
  - Much less performance reduction compared to subthreshold operation



[Intel2012ISSCC]

Ultra-low Power	Energy Efficient	High Performance
280 mV	0.45 V	1.2 V
3 MHz	60 MHz	915 MHz
2 mW	10 mW	737 mW
1500 Mips/W	5830 Mips/W	1240 Mips/W

- Challenges to NTC
  - Reduced performance (10x)
  - Increased sensitivity to variations and asymmetric delay distribution
    → Reliability issues
    → Large and expensive design margins
  - Disproportionate ratio of dynamic and leakage power

Related work and proposed work

- Related work
  - Coarse-grained power gating of idle cores:

    [Annavaram2011HPCA, Leverich2009, Bose2012DATE]

    - The cores are turned-off when determined idle time is observed.
    - State-preservation required → high wakeup latency
  - Power-gating of execution units [Hu1004ISLPED]
    - Instruction utilization pattern not analyzed.

- Power-gating of components is costly in terms of execution time
  - Mandates a minimum time between power-on and power-off cycles.
  - More optimization opportunities can be revealed by
    - Instruction utilization pattern analysis

Outline

- Background and related work
- Instruction pattern analysis
- Instruction clustering and ALU generation
- Energy, performance, and reliability improvement results
- Summary

Instruction pattern analysis

- Reveals unused parts of a circuit based on the running application
  - Power-gating those unused parts instead of the entire component
- For ALUs executing several instructions:
  - Parts of the ALU corresponding to the instructions not executed for a long time could be power-gated.
- Analysis goal:
  - Derive metrics for partitioning a functional unit into smaller units

    → Grouping the instructions
- Full understanding of the running workload needed:
  - Simulate the execution of representative workloads (SPEC benchmarks)
  - Extract instruction stream

Instruction utilization frequency

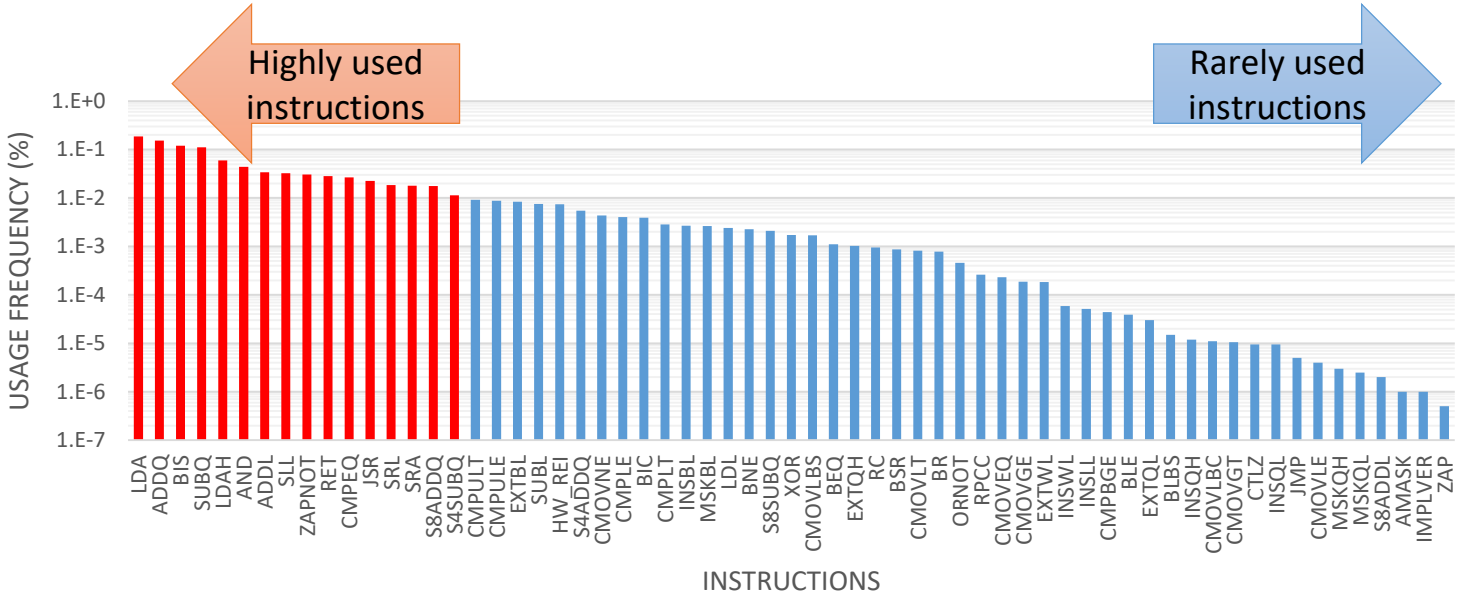
- Definition:
  - Number of cycles in which the instruction is executed divided by total cycles

instruction stream

$$Freq_A = \frac{\#S_i \in \mathbf{S}, \quad \text{such that } S_i = A}{|\mathbf{S}|}$$

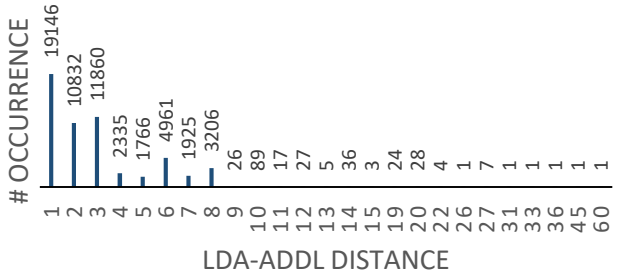
- Significant difference in utilization among the instructions
- Inversely related to the power-gating feasibility

64-bit Alpha ALU
'gzip' workload



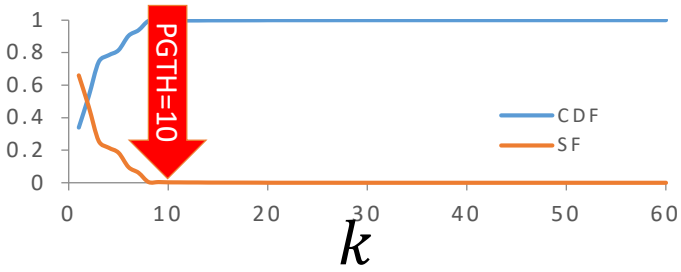
Instruction temporal distance

- Beneficial to group neighboring instructions inside one unit
  - Better power-gating interval for other functional units
- Observation from 'bzip2' workload:
  - ADDL appears after LDA on average every 2.97 instructions



- Probability distribution of observing instruction pairs (A-B) far from each other by k cycles
  - The Survival Function (SF) of the distribution
    - Represents the distance of instruction pairs
    - For a given power-gating threshold (PGTH)
  - Example: $SF_{\mathrm{LDA-ADDL}}(10) = 0.00279$
    - When power-gating threshold is 10
    - LDA−ADDL are very close: 0.279% of all occurrences have more distance

$$SF_{\mathrm{A-B}}(k) = 1 - CDF_{\mathrm{A-B}}(k)$$



Instruction similarity

- Many instructions share gates due to their similarity
  - Additions, Subtractions, Compare and XOR
  - Implementing these instructions in separate functional units imposes redundant structures
  - Preferable to group them into one functional unit
- Similarity: based on the knowledge we have about the logic implementation of different instructions
- *Three metrics:* for distance between instructions:
  - Frequency distance: $dist^{freq}_{\mathrm{A-B}} = \sqrt{Freq_{\mathrm{B}} * Freq_{\mathrm{B}}}$
  - Temporal distance: $dist^{temporal}_{\mathrm{A-B}} = SF_{\mathrm{A-B}}(PGTH)$
  - Dissimilarity: $dist^{dissimilarity}_{\mathrm{ADDL-ADDQ}} = 0$

Outline

- Background and related work
- Instruction pattern analysis
- Instruction clustering and ALU generation
- Energy, performance, and reliability improvement results
- Summary

Instruction clustering

- Partition instructions into several clusters:
  - Each cluster is implemented as one functional unit
  - Clustering goal:
    - Maximize the distance between the instructions in two functional units

  → More power-gating likelihood → lower leakage and better energy efficiency
- Agglomerative Hierarchical Clustering (AHC) algorithm
  - Instructions distance explained by:
    - Utilization frequency, Temporal distance, and Dissimilarity metrics
  - Required *pairwise distance matrix*:
    - $pdist^2_{\mathrm{A-B}} = \left(\beta . dist^{freq}_{\mathrm{A-B}}\right)^2 + \left(\gamma . dist^{temporal}_{\mathrm{A-B}}\right)^2 + \left(\lambda . dist^{dissimilarity}_{\mathrm{ADDL-ADDQ}}\right)^2$
  - single-linkage clustering: maximizing the distance between clusters
    - linkage function: $D(X,Y) = \min_{A \in X, B \in Y} pdist_{\mathrm{A-B}}$

Fine-grained power-gating prediction

- Power-gating of component:
  - when inactive phase for a component is detected at architectural level
  - Using a sleep signal on header/footer sleep transistors

- Power-gating is associated with overheads
  - Break-even when a component is not utilized for a minimum number of cycles (**PGTH**)
  - PGTH ~10 for typical technology [Hu1004ISLPED]

- Detecting the inactive phase of a functional unit:
  - Monitor the instruction buffer for a number of upcoming instructions
  - Branch prediction buffer should also be considered

Outline

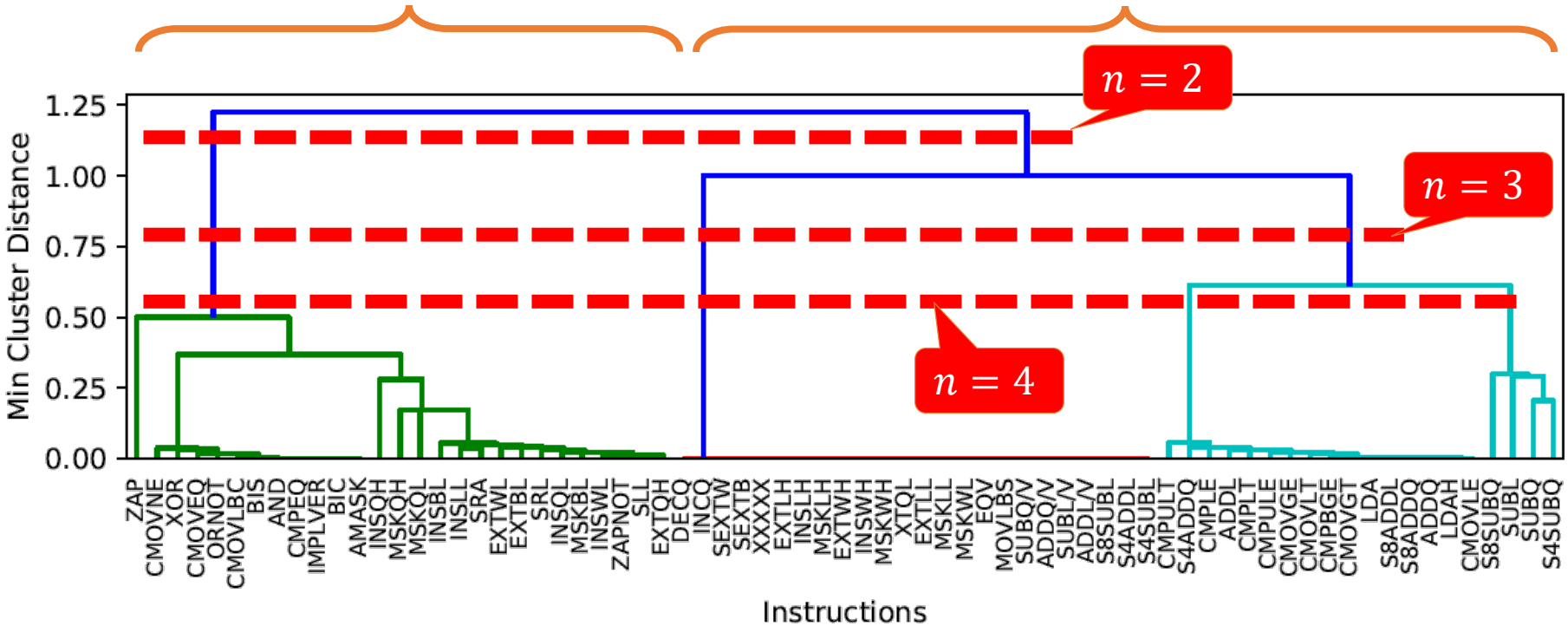
- Background and related work
- Instruction pattern analysis
- Instruction clustering and ALU generation
- Energy, performance, and reliability improvement results
- Summary

Instruction analysis setup

- Simulate the execution of representative workloads
  - Using gem5 architectural simulator, for SPEC2000 benchmarks
- 14nm and 10nm PTM transistor models
  - To evaluate the impact of technology on the proposed methodology
- Evaluate the proposed method at:
  - nominal supply voltage, near-threshold voltage region, sub-threshold region
- 64-bit ALU (Alpha ISA)
  - synthesis with Synopsys Design Compiler
  - Power and timing analysis reports extracted with Synopsys PrimeTime

Clustering results

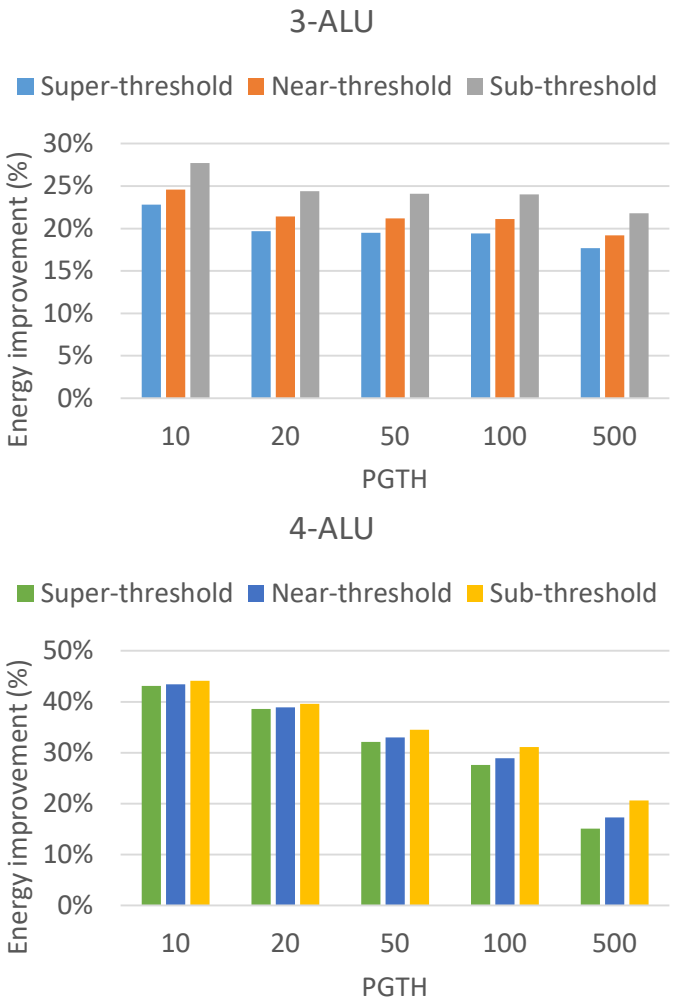
- The temporal distance metric extracted for PGTH = 100
- The ALU can be partitioned into n smaller ALUs
  - $n \leq 4$: Limited by the overhead of hardware implementation



Circuit results

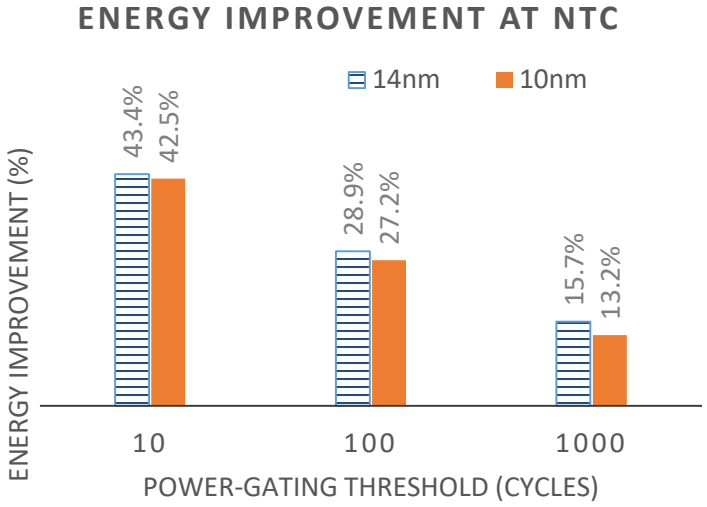
- the original ALU partitioned into:
  - three ALUs (3-ALU)
  - four ALUs (4-ALU).
- Area overhead: 3-ALU (17%), 4-ALU (19%)
- Performance improvement ~7% at NTC
  - Partitioned ALUs perform faster
- Energy improvement obtained by:
  - Calculating the percentage of the time that each smaller ALU can be power-gated
  - Calculated for several PGTH values

Reported numbers are averaged over different workloads

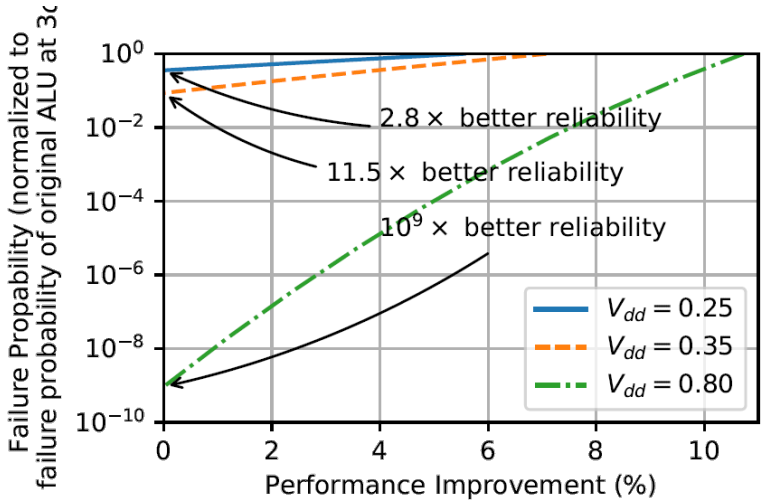


Comparison for

- Similar energy improvement trend for 10nm and 14nm



- Performance and Reliability trade-off
  - Trading performance for reliability
  - Failure probability calculated:
    - For different performance improvements
  - 3-sigma guardband respected



- Better reliability:
  - Up to 11.5x at Near-threshold
  - Up to 2.8 at Sub-threshold

$$\text{Timing Failure Probability} = 1 - \text{Reliability}$$

$$= 1 - \prod_{\text{INST}}^{\{\text{all instructions}\}} CDF_{delay,\text{INST}}(T_{clk}).$$

Outline

- Background and related work
- Instruction pattern analysis
- Instruction clustering and ALU generation
- Energy, performance, and reliability improvement results
- Summary

Summary

- Major challenges of modern ultra-low power designs:
  - Excessive leakage power and reliability issues
  - New design strategies needed to control both
- We proposed a balancing technique for functional units (e.g. ALU)
  - Partition a functional unit into smaller and faster functional units
  - Better energy efficiency by power-gating inactive functional units
  - Better reliability by trading the gained performance improvement
- A clustering method is proposed:
  - To find an optimum grouping → maximize inactivity time of functional units
- Simulation results show at NTC
  - 43.4% better energy efficiency 7%
  - performance gain or 11.5x better reliability

Thank you

Q & A

References

- [Dreslinski2009IEEE] R. Dreslinski, et al., "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits." *Proc. IEEE,* vol. 98, no. 2, pp. 253-266, Feb 2010.
- [Intel2012DAC] H. Kaul et al., "Near-threshold voltage (NTV) design: opportunities and challenges." in *DAC*, pp. 1153-1158, 2012.
- [Intel2012ISSCC] S. Jain et al., "A 280mV-to-1.2 V wide-operating-range IA-32 processor in 32nm CMOS." in *ISSCC*, pp. 66-68, 2012.
- [Annavaram2011HPCA] M. Annavaram, “A case for guarded power gating for multi-core processors,” in HPCA, 2011.
- [Leverich2009] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, “Power management of datacenter workloads using per-core power gating,” Computer Architecture Letters, 2009.
- [Bose2012DATE] P. Bose, A. Buyuktosunoglu, J. A. Darringer, M. S. Gupta, M. B. Healy, H. Jacobson, I. Nair, J. A. Rivers, J. Shin, A. Vega et al., “Power management of multi-core chips: Challenges and pitfalls,” in DATE, 2012.
- [Hu1004ISLPED] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, “Microarchitectural techniques for power gating of execution units,” in ISLPED, 2004, pp. 32–37.