Software Decomposition for Multicore Architectures

Ankit Jain, Ravi Shankar

Florida Atlantic University

Multicore Architectures: Software Reuse Challenge

- Reverse Engineer existing (legacy) software to port it on next generation Multicore Architectures
- Partition existing code over Multiple concurrent cores
- Transform existing Sequential Model based software to Concurrent Model based software

Methodology

- Ten Steps uses
  - Bottom-up Annotation
  - Middle-Out Analysis
  - Top-Down Representation
- Maximizes Software Reuse
- Performs Concurrency Modeling and Performance Simulations
- Selects MA and decomposed software architecture to meet QoS requirements

Results

- Analysis was performed on 2-Core and 4-Core architecture solutions for 2 Algorithms

Algorithm 1	Algorithm 2
Properties: 1. Master Processor – Processed CCD, CCDPP, CNTRL & REDEYE + Slave Management 2. Slave Processor(s) – Processed Parallelized CODEC 3. Pre-Expanded Instructions are passed to the Slave. Leads to heavy communication over the Bus.	**Properties:** 1. Master Processor – Slave Management 2. Slave Processor(s) – Processed Parallelized CCD, CCDPP, CNTRL, REDEYE, CODEC 3. Slaves expands the instructions locally based on code and data received. Significant reduction in usage of the Bus as compared to Algorithm 1.
Performance: 4-Core showed over 23% improvement over 2-Core implementation	**Performance:** 4-Core showed over 65% (near 3x) improvement over 2-Core implementation. Algorithm 2 also showed significant reduction in total execution cost as compared to Algorithm 1.