Sorting AsaP2

- **Snake Sort** algorithm
  - Insertion sort inside each processor
  - Same cost to build sort on the chip
  - Exact same code in every processor
  - Simple but not highly efficient
  - Every memory accessed by every core
  - Requires distribution of processors
  - Local processors in core
  - Merging processors
  - Each command packet signals sorting processors that the block has completed and is ready for merging
  - No globally-accessible memory such as a cache
  - Extremely power efficient
  - Limited sorting algorithm possible

Many-Core Array as a Co-Processor

- General purpose CPU example kernel sort code
  - 6 clock cycles / record
  - 100kB sorted data
  - No need for a CPU to reattach data
  - SAISort Kernel
  - Pseudo code of the main Serial Array of Insertion Sort (SAISort) kernel used in all of the sorts:

Sorting With 100-Byte Records

- Sorting 10-GB of 100-Byte records including a 10-Byte key
  - All of this on two processors
  - 100kB record size

Phase 1 Sorting on AsaP2

- `Database` records
  - 96 GB - 96 TB of data
  - Very high performance
  - Large sort of the sort, so we targeted that to give the most benefits
  - Serial CPUs already handle merging single improvements

SAISort Phase 1 Sorting on AsaP2: Using Large Shared Memories

- Heuristics showing the active percentages of the processors
  - In the Snake Sort, activity reaches a plateau of 20% for most of the mid processors
  - In the Row Sort, the input and output processors are the bottleneck
  - Processors consume essentially zero power while stalled
  - This shows one run, with multiple runs, there would be overlap, removing some of the under-utilized processors

Processor Activity During Sort

- Current / Future Work
  - SAISort performs 3.1 to 4.9 MSorts per second
  - “Dynamic Sort” algorithm
    - Sorts data performed on 3.1 MSorts of incoming keys
    - Processor will dynamically allocate the processor to between two and the fastest
      - Depending on the processor to dynamically resize based on the data
      - Should show special for very specific use of data
      - Effective sort of the data with the highest merging overhead in the end
  - Core Scaling
    - Run simulations on how the sorting varies at various core with different block size of processor

---

**Sorts With Very Large Data Sets**

- “External” sorts typically utilize two distinct phases:
  - Sort all records that fit in memory
  - We only focus on this phase since it is where the greatest improvements
  - Successively merge lists into one first sorted list

Memory

- Many-core array as a functional unit
- Many-core array computes entire applications
- No specialized instructions
- No specialized memory hierarchy
- On-chip shared memories
- Memory to work very well for DM1/blocks, extended applications
- This project requires minimal translation extension to enterprise workloads
- Many-core array as a co-processor
- Many-core array can compute entire applications with limited overhead
- Project commenced June 2011

---

**Modular Sorting on a Fine-Grained Many-Core Processor Array**

Aaron Stillmaker and Bevan Baas

Department of Electrical and Computer Engineering
University of California, Davis

VLSI Computation Laboratory

- Sorting 10 GB of 100-Byte records including a 10-Byte key
- All this on two processors
- 100kB record size
- AsaP2 sorting uses 1.3x less energy
- AsaP2 sorting achieves up to its higher throughput per area

---

Slide 2

- Example Enterprise Application: Database RegExp + Sort + Statistics

---

Slide 4

- Modular Sorting on a Fine-Grained Many-Core Processor Array
- Many-Core Array as a Co-Processor
- Sorts With Very Large Data Sets
- Phase 1 Sorting on AsaP2

---

Chip Memories

- AsaP2, Intel Core 2 Duo
- Mem

---

Slide 5

- Sorting Results With 100-Byte Records
- Sorting Results With 100-Byte Records

---

Slide 10

- Processor Activity During Sort

---

Slide 20

- Current / Future Work

---

Slide 21

- Example Enterprise Application: Database RegExp + Sort + Statistics

---

Slide 22

- Sorting Results With 100-Byte Records
- Sorting Results With 100-Byte Records

---

Slide 23

- Processor Activity During Sort

---

Slide 24

- Current / Future Work

---

Slide 25

- Example Enterprise Application: Database RegExp + Sort + Statistics

---

Slide 26

- Processor Activity During Sort

---

Slide 27

- Current / Future Work