Parallel Merge Sort

Kyle Verma & Hengyu Jiang

ECE 18-667 Poster Session
Dec. 6, 2013, Pittsburgh PA

1. Introduction

Observation of Merge Sort Algorithm

- Pre-simulation result of merge sort without network

Size	512	1024	2048
Single	10248	22537	49162
Bin-7	2468	7437	15886
Bin-15	2381	5006	10511

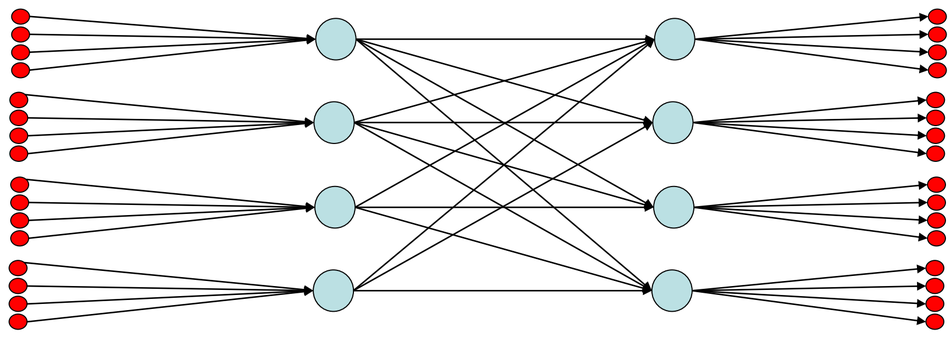
- The algorithm hints a tree structure
- Two problems with binary tree. Another topology is needed
  - Low CPU utilization
  - High communication overhead between two processors that aren't in a direct parent-child relationship
- We investigate several general purpose NoC topology

	Router Cost	Average Latency	Communication Load	Routing Algorithm	Wire Cost
Mesh	16	2.63	Varies latency May have deadlock	Complex	Low
Butterfly	8	2	Balanced Load Deadlock free	Easy	High
Mix	16	2.61	Bottleneck In critical nodes	Complex	Low

2. System Architecture

NoC Topology

- We use butterfly topology for NoC.



- Two ports for arbiter
  - As data transmission into and from the arbiter is high, one more network port is assigned to arbiter to increase bandwidth.
  - Bring 21.3% improvement.

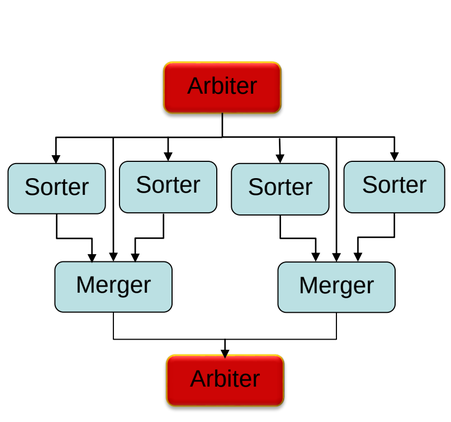
3. Module Implementation

Router Design

- Buffer-less Router Design
  - Data will be transferred from PE to PE directly, without buffering.
  - Circuit switching is used.
  - Save many cycles on handshake

PEs Design

- There are three kinds of PEs: arbiter, sorter and merger.
- Dataflow

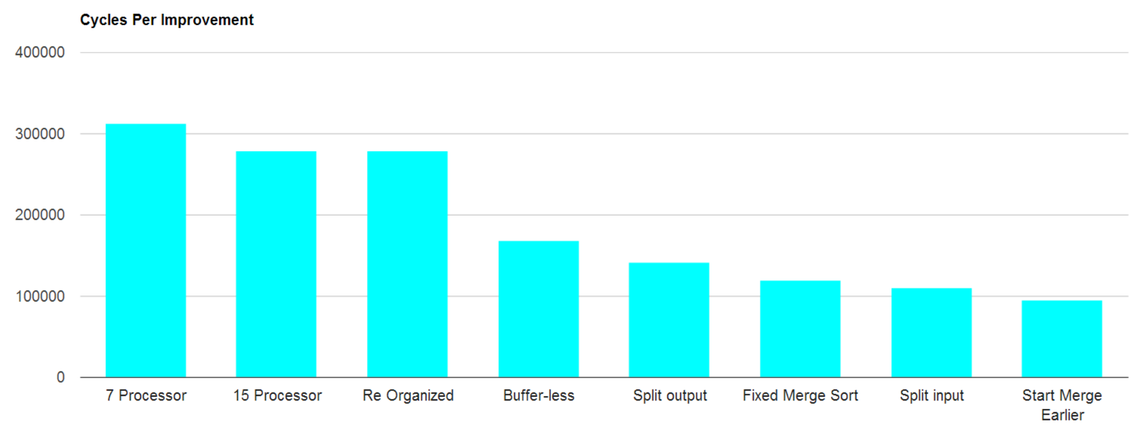


- Workflow
  - Arbiter receives data from input, delivering data to each PEs, then sorts its workload, then merge data from mergers.
  - Mergers sort their workload first, then do merge with the list they already sorted and the lists from sorter. Finally, send the merged list to arbiter.
  - Sorters their workload then send sorted lists to mergers.

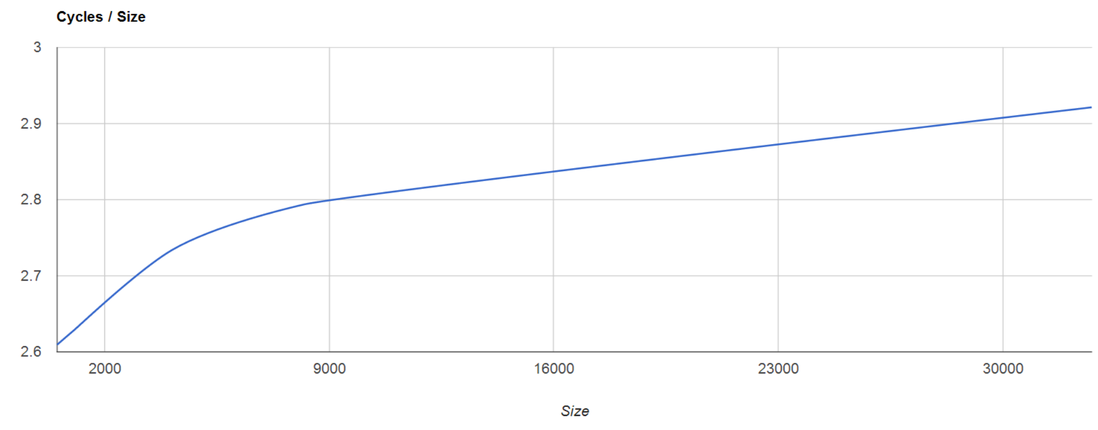
4. Results

Plot

- Performance Improvement



- Final Performance(cycle over size)



- Noise Test (Cycle over noise size)