CUstom Built HEterogeneous Multi-Core ArCHitectures (CUBEMACH): Breaking the Conventions

Nagarajan Venkateswaran
Director, Waran Research Foundation

Karthikeyan Palavedu Saravanan - Nachiappan Chidambaram Nachiappan
Research Trainees (2008 - 2010), Waran Research Foundation

Aravind Vasudevan - Balaji Subramaniam - Ravindhiran Mukundarajan
Former Research Trainees (2007 - 2009), Waran Research Foundation
Motivation: Heterogeneity Redefined

- Cost Effective High Performance Custom Built Heterogeneous Multi-Core Node Design for wider class applications
  - Inter and Intra core heterogeneity
- Breaking the Conventions
  - Multiple User Multiple Application without Space-Time sharing in a Cluster: Cost sharing across users
  - Single User Multiple Application without Space-Timer Sharing (non-multiprogramming): Cost sharing across applications
Overview

• Custom Built Heterogeneous Multi-Core Architectures (CUBEMACH)

• Design Space
  – Architectural Space
  – Optimization Space
    – Customer Vendor Interaction
  – Simulation Space

• CUBEMACH Design and Simulation Tool Framework

• Conclusion
Custom Built Heterogeneous Multi-Core Architectures (CUBEMACH)

• CUBEMACH promises
  – Increased Resource Utilization
  – Multiple Application Flavored Architectures
  – Elimination of Space Time Sharing at the Quantum Level during Multiple Application Execution
  – Manufacturing and Operational Cost reduction
Overview

- Custom Built Heterogeneous Multi-Core Architectures (CUBEMACH)
- Design Space
  - Architectural Space
  - Optimization Space
    - Customer Vendor Interaction
  - Simulation Space
- CUBEMACH Design and Simulation Tool Framework
- Conclusion
CUBEMACH Design Paradigm

- Architectural Space
- Optimization Space
- Simulation Space

CUBERMACH Design Paradigm

WAran Research FoundaTion
Chennai, India
Architectural Design Space - CUBEMACH

ALISA

Compiler-On-Silicon

PCOS SCOS

ALFU

DRAM SRAM

Memory

ONNET

CUBEMACH Architectural Space

WARan Research Foundation
Chennai, India
Architectural Space

• Why ALU Why Not ALFU??
  – Hardwired units
  – Design: Homogeneously Structured
  – Reduced Instruction Generation & Fetches: Employ a Higher Level ISA
  – Reduced memory-functional unit interaction
  – Helps execute multiple applications without space & time sharing
Algorithm Level Functional Unit

ALFU Types
- Algorithm Size
  - Memory in Processor
    - Type of MIP Cell
  - Delay
    - ALFU Requirements
      - Control Unit
        - Centralize/Decentralized
          - Class of Units
            - Input Bits
              - HLFU Characteristics
                - Grain Size
                  - Architecture
                    - MIP Cell Number
                      - Number of MIP Cells
                        - Class of Algorithms
                          - ALFU Types
ALU vs ALFU Instruction Generation Results

![Graph showing comparison between ALU and ALFU instructions]

- **LUD - Problem Size**
  - ALU Instructions
  - ALFU Instructions

- **Convex Hull - Problem Size**
  - ALU Instructions
  - ALFU Instructions

**WARan Research Foundation**
Chennai, India
Sample Algorithm Level Functional Units

- **Scalar Units**
  - Scalar Adder / Subtractor
  - Scalar Multiplier
  - Scalar Divider
  - Comparator
  - Sorter
  - Multiple Operand Adder
  - Min / Max Finder

- **Vector Units**
  - Inner Product

- **Matrix Centric Units**
  - Matmul
  - Matadd
  - Chain Matadd

- **Graph Theoretic Units**
  - Graph Traversal Unit – BFS, DFS
  - KL Graph Partitioning
ALISA – Algorithm Level Instruction Set Architecture

- Algorithm Level Instructions
- Triggers ALFUS
- ALISA → Multiple VLIWs
- ALISA for heterogeneous multi-cores
Hierarchical Compilation Scheme

- PCOS Partitions A Problem Into Sub-Problems – Level 1

- SCOS Partitions The Sub-Problems Into ALFU Level Instruction – Level 2
ON-Node-Network Architecture
ON-Node-Network Architecture

H- Tree Topology

- Global Router
- Local Router
- Sub-Local Router

Architectural Space Contd...
Comparison of Conventional NOCs with ONNET

<table>
<thead>
<tr>
<th></th>
<th>ONNET</th>
<th>Conventional NOCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Switch</td>
<td>MIN</td>
<td>Crossbar</td>
</tr>
<tr>
<td>Number of Routers</td>
<td>(N \times \log_2(N))</td>
<td>(N^2)</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Switching Latency</td>
<td>(\log_2(\text{Number of Inputs}) \times \text{Switch Delay})</td>
<td>(\text{Number of Inputs} \times \text{Switch Delay})</td>
</tr>
</tbody>
</table>
Overview

• Motivation
• Custom Built Heterogeneous Multi-Core Architectures (CUBEMACH)
• Design Space
  – Architectural Space
  – Optimization Space
    – Customer Vendor Interaction
  – Simulation Space
• CUBEMACH Design and Simulation Tool Framework
• Conclusion
Optimization Space

CUBEMACH Architectural Space

CUBEMACH Optimization Space

CUBEMACH Simulation Space

Multiple Applications Input

Initial Candidate Architecture Parameters

Game Theory

Selected Parameters

Simulated Annealing

Core Formation

Desired Power to Performance Ratio

Final CUBEMACH

Calculated Power to Performance Ratio

Power Model

Performance Model

WAran Research Foundation
Chennai, India
Optimization Space

• Generates Optimized CUBEMACH for input specifications such as,
  – Power – Performance – Cost
  – Initial Architecture
• Power and Performance Model
• Uses GT and SA for optimization of Power and performance
• Uses KL For Core Grouping
Sample CUBEMACH Architecture
CUBEMACH Design
Implementation: Supercomputer On Chip (SCOC) IP Cores
SCOC IP Cores

- ALFUs designed as SCOC IP Cores
- Soft IP Core
- Coarse-grained Reusable Soft IP Cores
- Scalable IP Cores
Overview

• Motivation
• Custom Built Heterogeneous Multi-Core Architectures (CUBEMACH)
• Design Space
  – Architectural Space
  – Optimization Space
  – Customer Vendor Interaction
  – Simulation Space
• CUBEMACH Design and Simulation Tool Framework
• Conclusion
CUBEMACH Simulator

- pThread based Simulator
- Evaluates candidate CUBEMACH Architecture
- Feed results to CUBEMACH Optimizer
- CUBEMACH Optimization Engine (COE) produces Optimized Architecture
- Simulation & Optimization: An iterative process
- Consists of
  - ALFU Sub-Simulator
  - COS Sub-Simulator
  - ONNET Sub-Simulator
  - Memory Sub-Simulator
CUBEMACH Simulator
What we have seen . . .

Integrated CUBEMACH Design Paradigm ...
Sample CUBEMACH Architecture:

Simulation Results

Matrix Based Algorithms

Graph Based Algorithms

WAran Research FoundaTion
Chennai, India
Sample CUBEMACH Architecture:

Simulation Results

Mixture of Algorithms

Comparison of Performance delivered by Optimized Architectures for corresponding types of Algorithms
Sample CUBEMACH Architecture:

Simulation Results

Overall Resource Utilization of:

(i) Initial CUBEMACH Architecture: Mean = 59 %
(ii) Optimized CUBEMACH Architecture: Mean = 74 %
In Initial Candidate CUBEMACH Architecture,
  • Matrix ALFUS – low usage
  • Scalar ALFUS – average usage
  • Graph ALFUS – high usage

In Optimized Candidate CUBEMACH Architecture,
  • Matrix ALFUS – high usage
  • Scalar ALFUS – high usage
  • Graph ALFUS – high usage
Conclusion

- Custom Built Heterogeneous Multi-Core Architectures (CUBEMACH) promises,
  - Increased Resource Utilization
  - Multiple application flavored architectures
  - Elimination of Space Time Sharing at the Quantum Level during Multiple Application Execution (without multiprogramming)
  - Manufacturing and Running Cost reduction
Thank You

Questions??
Customizable Compiler-On-Silicon

• What Compiler-On-Silicon?

• Why do we need Compiler-On-Silicon?

• Why go for Customizable Compiler-On-Silicon?
ONNET

Architecture uses -
- Multistage Interconnect Network
- Hardware Packetization Unit
- ONNET Design Space
  - H-Tree Structure within a Core
  - 2D Torus Across Cores
  - MIN Type
Architectural Design Space - CUBEMACH

- ALFU – Algorithm Level Functional Units
- BISA – Backbone Instruction Set Architecture
- COS – Compiler On Silicon
- ONNET – On Node Network
- Novel Cache Mapping Scheme
- SCOC IP Cores: Achieving cost effectiveness

( Super Computer On Chip - IP Cores)
On Node Network Architecture

Features -

• Communication across heterogeneous multi-cores
• Data requirements of diverse ALFUs
• High bandwidth
• Scalable
• Hierarchical Network-On-Chip
Memory

SRAM

DRAM

Mapping and Replacement Heuristic

Packet Size

No of Blocks

DRAM Size

SRAM Size

Word Length

Cache Line Size

No Of Ports

WAran Research Foundation
Chennai, India
Advantages of SCOC IP Cores

• Fully Customizable
• Greatly reduces Design-Turnaround-Time
• Physically Design Friendly
  – Constraints of Area, Power and Performance
• Constrained & Rigid Design Methodology