Optimizing Cache Coherent Subsystem Architecture for Heterogeneous Multicore SoCs

Niu Feng
Technical Specialist, ARM

Tech Sympoisa
2016

©ARM 2016
Agenda

- Introduction
  - Challenges: Optimizing cache coherent subsystem architecture for heterogeneous multicore SoCs

- Solution components
  - Architecture prototype simulation and analysis
  - Cache coherent architecture models (processors, interconnect, and memory subsystems)

- Case study illustration

- Summary / Q&A
Growth in heterogeneity being driven by automotive

New data for training

Updated model

CPUs + HW Accelerators

CPUs + HW Accelerators
How to best communicate between processing elements?

- System-level constraint is becoming **inter-cluster** communication
  - Complex communication semantics
  - Latency and bandwidth requirements
  - Software complexity, portability and scalability
  - Power consumption (DRAM)

- **Hardware cache coherence** can simplify and optimize the system
  - But many accelerators **don’t have a cache!**
  - Heterogeneous subsystem architectures enable both
Approach

Using virtual prototypes with cache-coherent architecture performance models to enable

- Capture of application workloads (data and address)
- Efficient turn-around time for exploration
- System level analysis visibility for optimization
- Validation with high accuracy

Case study featuring:

- Synopsys Platform Architect
- ARM Fast Models and ARM Cycle Models
- Arteris Ncore cache coherent interconnect
Agenda

- Introduction
  - Challenges: Optimizing cache coherent subsystem architecture for heterogeneous multicore SoCs

- Solution components
  - Architecture prototype simulation and analysis
  - Cache coherent architecture models (processors, interconnect, and memory subsystems)

- Case study illustration

- Summary / Q&A
Synopsys Platform Architect

- Traffic-driven workload modeling
- Comprehensive library of generic and vendor specific architecture models
- Graphical assembly and analysis to explore SoC architecture tradeoffs
- Runtime configuration for simulation sweeping and sensitivity analysis
- Achieve system performance, power, and cost goals without over/under design

Deployed by leading companies worldwide to optimize SoC architecture and reduce the risk of under- and over-design
Virtual prototype solution flow

Workload capture → Exploration → Optimization → Validation

Fast Models Spec

Task Graphs and VPUs

Generic interconnect model

Ncore interconnect model

Cycle Models

Platform Architect

ARM © 2016
ARM Fast Models

- Fast, functionally complete models of all ARM IP
  - 10s to 100s of MIPS speed to enable OS boots and software development
  - Accuracy to run even the lowest level unmodified binaries
  - Timing annotation to aid system optimization
- Earliest availability models
  - Architecture models represent ISA
- Unmatched functionality
  - Caches, TrustZone, Virtualization, Crypto,
  - Multi-core/multi-cluster
- SystemC TLM interfaces to ensure wide compatibility
- Fixed virtual platforms to model reference systems

Library includes:
- Cortex-A CPUs
- Cortex-R CPUs
- Cortex-M CPUs
- CoreLink CCI Interconnect
- CoreLink CCN/CMN Interconnect
- Mali GPUs
- Mali Display Processors
- Mali Video Processors
- ARM PrimeCells
- And more…
ARM Cycle Models

- Compiled directly from ARM implementation RTL
  - 100% implementation accurate
  - All registers maintained
- Fully instrumented for in-depth analysis
  - TARMAC trace generation
  - Interactive debugging with DS-5
  - Pipelines, caches, memories, performance counters, etc
  - VCD & FSDB format waveforms
- Pin and TLM level integrations
  - Integrated with Synopsys PA and VDK
- Configurable 24/7 using ARM IP Exchange

Available for a wide range of ARM IP including CPU & Coherent Interconnect
Agenda

- Introduction
  - Challenges: Optimizing cache coherent subsystem architecture for heterogeneous multicore SoCs

- Solution components
  - Architecture prototype simulation and analysis
  - Cache coherent architecture models (processors, interconnect, and memory subsystems)

- Case study illustration

- Summary / Q&A
Use case: Vision SoC
- CPUs for general processing
- Accelerators for real-time image and data processing

Goals
- Optimize interconnect configuration to achieve target bandwidth and latency
- Minimize power and cost (cache size and DDR frequency)
- Validate results
Workload model creation and mapping

**SW application**

---

**SW execution trace**

- CPU software profile

---

**VPU platform**

---

**Data-processing application**

- Vision software profile
  - spec.docx
  - spec.vsd
  - spec.xlsx

---

**TGG**

- Generate

---

**CPU Task Graph**

---

**Vision Task Graph**

---

**Map**

---

**Map**
Platform model for exploration

Quad core VPU
- 4 processing resources
- Shared L2 cache

Vision Task Graph
Accelerator VPU
- Single resource
- No cache

Generic cache coherent interconnect (GCCI)
- 2x fully coherent CPU ports
- 2x ACE-lite accelerator ports with/without cache
- Configurable snoop

DDR controller
- 3 AXI4 ports
- Configurable DDR
Scenario: DRAM or cache?

PROBLEM
- Many systems with HW accelerators use pipeline processing
- Dedicated SRAMs cost area
- DRAM accesses are too slow and power hungry

SOLUTION
- Enable caching for accelerators hardware without a cache (can be system cache or proxy cache)
- Advantages
  - Better for sharing data between non-coherent agents (HW accelerators) and coherent agents (CPU and GPU clusters)
  - Better for sharing data between custom processing elements (pipelining!)
  - Using a cache **minimizes communication through DRAM**
    - Better latency, bandwidth and power
Scenario: DRAM vs. cache latency

In addition to lower latency:
- Lower power than DRAM
- Less area than SRAMs

Latency is the Key Metric
Scenario: DRAM vs. cache

- **Scenario:**
  - DRAM vs. cache

- **Baseline**
  - Start frame: start, end frame: end
  - CPU utilization: 60%

- **Cache**
  - Start frame: start, end frame: end
  - CPU utilization: 30%

- **DRAM Only**
  - Start frame: start, end frame: end
  - CPU utilization: 50%

- **With Cache**
  - Start frame: start, end frame: end
  - CPU utilization: 20%

- **Significant deadline violation**
  - Accelerator utilization
  - Poor cache performance

- **End-to-end deadline achieved**
  - Accelerator utilization
  - Good cache performance

- **High DDR utilization, especially when accelerators are active**
  - Low DDR utilization
Optimization

- Confirm results from exploration phase
  - Replace Generic Cache Coherent Interconnect (GCCI) with accurate Ncore model

- Tuning of design and configuration parameters
  - Outstanding transaction limits
  - Quality of Service
Confirm results from exploration phase
Validation

- Confirm results from optimization and exploration phase
  - Replace task-based workload model running on VPU with actual Software running on ARM Cycle models

- Final Tuning of design and configuration parameters
Case study results

- Use case: Vision SoC
  - CPUs for general processing
  - Accelerators for real-time image and data processing

- Goals
  - Optimize interconnect configuration to achieve target bandwidth and latency
  - Minimize power and cost (cache size and DDR frequency)
  - Validate results
Agenda

- Introduction
  - Challenges: Optimizing cache coherent subsystem architecture for heterogeneous multicore SoCs

- Solution components
  - Architecture prototype simulation and analysis
  - Cache coherent architecture models (processors, interconnect, and memory subsystems)

- Case study illustration

- Summary / Q&A
Summary

Optimizing Cache Coherent Subsystem Architecture for Heterogeneous Multicore SoCs

Use virtual prototypes with cache-coherent architecture performance models to enable
- Capture of application workloads
- Efficient turn-around time for exploration
- System level analysis visibility for optimization
- Validation with high accuracy
Thank you!

- Bill Neifert  
  email: bill.neifert@arm.com
- Pat Sheridan  
  email: pat.sheridan@synopsys.com
- Matthew Mangan  
  email: matthew.mangan@arteris.com