ARM Neoverse N1

Aishik Ghosh, Anant Kandikuppa, Zane Fink
CS 433 Mini Project
Outline

1. Introduction
2. Core Microarchitecture
3. Memory Hierarchy
4. Parallelism Support
5. Special Features for Infrastructure and Security
Introduction

- ARM CPUs are RISC processors designed by Arm Holdings
- ARM Neoverse N1 is an infrastructure-focused chip that implements the Arm v8.2-A instruction set
- Hardware features make the Neoverse N1 especially suited to cloud and infrastructure applications
Introduction: Infrastructure

- Workloads commonly found in cloud environments such as AWS, Microsoft Azure, etc.
- Codes characterized by:
  - Complex branching behavior
  - JIT compiled
  - Object management
  - Garbage collection
- Concretely: web servers, datacenter applications
Core Microarchitecture

- Superscalar processor
- 11-stage out of order accordion pipeline
  - Can drop to 9-stages

*WikiChip - Neoverse N1 Microarchitecture*
Core Microarchitecture

- Core has
  - 4 way decode
  - 3 ALUs
  - 1 branch exec unit
  - 2 adv SIMD units
  - 2 load/store exec unit

WikiChip - Neoverse N1 Microarchitecture
SIMD

- Single instruction multiple data
- Adv SIMD performance
  - native quad-word, dual 128-bit units fed by separate issue queues
  - Latencies
    - Floating point add - 2 cycles
    - Floating point multiply - 3 cycles
    - Floating point multiply add - 4 cycles
Core Backend

WikiChip - Neoverse N1 Microarchitecture
Branch Prediction

- 6k-entry main branch target buffer
- 3-cycle access latency to retrieve branch target addresses (without I-cache)
- Once a prediction is made, the predicted address is stored into a 12-entry fetch queue which tracks future fetch transactions.
- Decoupled Branch Prediction
Memory Hierarchy Introduction

- Deep Cache Hierarchy L1, L2, optional shared L3, and an optional system-level cache.

- Strict inclusivity between L1 data cache and L2 cache, non-inclusivity between L1 instruction cache and L2 cache.

Arm Neoverse N1 Cloud-to-Edge Infrastructure SoCs
Memory Hierarchy Overview

- Instruction Cache (Private):
  - 64 KB
  - 4-way set-associative
  - 16B of instructions per cycle
  - 4-cycle LD-use latency.

- L1 Data Cache (Private):
  - 64 KB
  - 4-way set-associative
  - 32B per cycle
  - 4-cycle LD-use latency
Memory Hierarchy Overview

- **L2 Data Cache (Private):**
  - Configurable, 256KB-1MB in size
  - 8-way set associative
  - 64B per cycle between L1/L2 caches
  - 9-11-cycle LD-use latency
Memory Hierarchy Overview

- *(Optional) L3 Cluster Data Cache:*
  - Multiple cores can be configured in a cluster
  - Up to 2MB, 28-33 cycle LD-use latency
  - Modified Exclusive Shared Invalid (MESI) coherence protocol
  - Coherency managed through snoop filter

- *(Optional) System level Cache:*
  - Up to 128MB
  - Coherency managed through snoop filter
  - 22ns LD-use Latency
Memory Hierarchy Overview

- **Instruction TLB**
  - 48-entry, fully associative
  - Support for 4KB, 16KB, 64KB, 2MB, and 32MB page sizes

- **Data TLB**
  - 48-entry, fully associative
  - Support for 4 KB, 16KB, 64KB, 2MB, and 512MB page sizes

- **Unified (instruction/data) L2 TLB**
  - 1280-entry 5-way set associative
Memory Hierarchy Key Features

- Cache miss predictor: Bypasses cache hierarchy to avoid cache access times.
- Cache prefetcher that can detect complicated access patterns.
- Fetch-aware cache replacement policies
- Supports maximum physical memory of 256 TB
- 68 in-flight loads, 72 in-flight stores
Parallelism Support

- 4-128 single threaded cores per chip
- Private L1 & L2 caches
- Optional shared L3 cache
- Multiple clusters supported by CoreLink CMN-600 mesh interconnect
- Allows specialized cores and accelerators to work together
Corelink CMN-600 Mesh Interconnect

- Provides high-frequency, non-blocking access to shared memory resources
- Supports upto 32 coherent clusters of CPUs / Accelerators
- Provides upto 128MB of shared system cache (SC)
- Optimized for low memory latency; ~1TB/s bandwidth
Corelink CMN-600 Mesh Interconnect - SC

- **Integrated Snoop Filters**
  - Reduces number of snoop requests
  - Lowers power consumption

- **Cache Stashing**
  - Allows external agents to directly place data into L3/L2 core caches

- **Direct Memory Transfer**
  - Allows memory controllers to directly send data to the requestor
Infrastructure Specific Features

- Enhanced Virtualization Support
  - VMID extended to 16 bits
  - Hardware update of access/dirty bits in page tables
- Improved Side Channel Protection
- Supports Armv8-A architecture extensions
  - CRC32 instruction to accelerate checksum computation
  - FP16 and int8 dot product support for ML inference
- Commercial deployment - AWS Graviton 2 processor
Thank You!

Questions?
References


Appendix
Concrete Example Application

- Nginx: an open-source high-performance webserver, proxy, and load balancer
- Goal: Reduce latency of a single request, maximize number of concurrent requests that can be serviced
- Quickly servicing a request requires:
  - Low memory latency and high bandwidth to process the request and transfer the response
  - Fast context switches between user and kernel space
  - Fast instruction fetching on the CPU front end
## Memory Hierarchy Performance

<table>
<thead>
<tr>
<th>Workload stressor</th>
<th>Neoverse N1 Features</th>
<th>N1 improvement over Cortex-A72</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object management</td>
<td>Memory allocations</td>
<td>2.4x faster</td>
</tr>
<tr>
<td></td>
<td>Object/array initializations</td>
<td>5x faster</td>
</tr>
<tr>
<td></td>
<td>Copy chars</td>
<td>1.6x faster</td>
</tr>
<tr>
<td></td>
<td>Smart HW handling of SW barriers (DMBs)</td>
<td>Memory barriers elided if unnecessary</td>
</tr>
<tr>
<td>Instruction footprint</td>
<td>i-cache miss rate and branch mispredicts</td>
<td>Reduced by 1.4x</td>
</tr>
<tr>
<td></td>
<td>L2 accesses</td>
<td>Reduced by 2.25x</td>
</tr>
<tr>
<td></td>
<td>Fully HW coherent Icache</td>
<td>Accelerates VM bring up by up to 20x</td>
</tr>
<tr>
<td>Garbage collection</td>
<td>Locking throughput w/ V8.2 Arch Atomic</td>
<td>Improved by 2x</td>
</tr>
</tbody>
</table>

### Graph:

- **Java Based Benchmark (throughput)**
  - Speedup vs 1 Cortex-A72
  - Data points for Cortex-A72 and Neoverse N1
  - Bar chart showing performance improvement

---

*Arm Neoverse N1 Cloud-to-Edge Infrastructure SoCs*
Cache Stashing

Offload acceleration

1. Configure registers for task
2. Fetches data from Core memory
3. Carries out acceleration
4. Writes result into Core memory

Example application:
Offload crypto acceleration

I/O processing

1. Writes data into Core memory
2. Carries out computation
3. Processing completed
4. Reads result from Core memory or sends data for onward processing

Example application:
Packet processing in network systems

Arm DynamIQ HotChips Presentation
Direct Memory Transfer

Figure 2-1 Snooable Read DMT structure