ARM Cortex A72 Processor
CS433 Miniproject

J. Lim, A. Kokolis, S. Yesil

12/2/2016
1 Single Core Architecture

2 Memory Subsystem

3 Multicores and Multiprocessors
   Coherence
   Synchronization
Architecture Overview

- High-performance, low-power processor that implements ARMv8-A architecture.
- One to four cores in a single processor with L1 and L2 cache.

---

1 ARM Cortex A72 MPCore Processor Technical Reference Manual
Architecture Overview

Non-Processor / Level 2
- Accelerator Port
- External Cache
- AXI4-Converter
- AXI4-Converter

L2 Arbitration
- Slave
- Master
- Slave
- Fastest

Shared L2 Cache (512 KB/MB A/B)
- 512 KB/MB A/B

Cortex-A72 Processor Core
- ITLB (48 Entry)
- L1 Instruction Cache 48KB
- L1 Instruction Line (associative 64-byte cache line)
- Branch Prediction
  - Indirect Predictor
  - w/path history
  - Global History Buffer
  - Branch Target Buffer (512 entries)
  - Return Stack

Instruction Fetch
- 128 bits (4 Instructions)

3-way Instruction Decode
- Decode
- Decode
- Decode
- Up to 3 micro-ops

Register Rename
- (Virtual to Physical Register Pool, 128 rename register)

Dispatch States
- Issue (4-entry queue per issue port)
- Up to 5 micro-ops Dispatch
- Up to 8 micro-ops Issue

Commit
- WriteBack
- Retirement Buffer (128 macro-ops)
- 48-bit Virtual Address
- 44-bit Physical Address

Load-Store Unit
- Store Buffer

L1 Data Cache 32KB
- (2-way set-associative 64-byte cache line, ECC)
- 1 Load & 1 Store per cycle

Branch
- ARM Integer
- Integer Shift-ALU
- Multiply & Divide, CRC
- ARM NEON & FPU
- Quad-FMA

Branch
- Cluster 1
- Cluster 2
- Branch

Cluster 2
- 2x-Source

Cluster 1
- 2x-Source

Integer ALU & Shifter
- Integer ALU & Shifter

Retirement Buffer (128 macro-ops)
- 14-Stage

12/2/2016

2016 flagship CPU of ARM “Cortex-A72” http://www.systranet.com/
Processor Components:

- Instruction Fetch Unit & Branch Predictor
- Instruction Decode Unit & Predicated Instruction
- Instruction Dispatch Unit
- Integer Execute Unit & Pipeline & Out-of-Order Execution
- Advanced SIMD & FP Unit
- Load/Store Unit

---

Inside ARM’s Cortex-A72 microarchitecture
http://techreport.com/review/28189/
Instruction Fetch Unit & Branch Prediction

- Fetches instruction from L1 i-cache
- Sends up to 3 instructions/cycle to Instruction Decode Unit
- Prediction:
  - 2-level global history-based direction predictor
  - BTB to identify branches and provide targets for direct branches
  - Static predictor
  - Indirect predictor to provide targets of indirect branches
  - Return stack
Instruction Decode Unit & Predicated Instruction

- Decodes instruction set of:
  - AArch64 Execution State
  - AArch32 Execution State
- AArch64 – A64 Instruction Set
- AArch32 – T32 and A32 Instruction Set
Instruction Dispatch Unit

- Controls dispatch of decoded instructions
- Considers:
  - ARM core general-purpose registers
  - ASIMD Floating-Point register set
  - AArch32 CP15 & AArch64 System registers
2 ALU pipelines
Multiply-accumulate & ALU pipelines
Iterative integer divide hardware
Branch & instruction condition codes resolution logic
Forwarding & Comparator logic

Cortex-A72 Software Optimization Guide
Integer Execute Unit & Pipeline & OOO Execution

- **In order:**
  - Instruction Fetch → Instruction decode into "micro-operations" (\(\mu\)ops)
  - \(\mu\)ops proceed through register renaming and dispatch stages
  - \(\mu\)ops wait for their operands and get issued

- **Out of order:**
  - \(\mu\)ops get issued to 1 of 8 execution pipelines
  - Each execution pipeline can accept and complete one \(\mu\)op per cycle.

<table>
<thead>
<tr>
<th>Decode</th>
<th>Max. Pipeline Depth</th>
<th>OOO Exec.</th>
<th>BPred</th>
<th>big.LITTLE</th>
<th>Fab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-wide</td>
<td>15</td>
<td>Yes, 5-wide</td>
<td>2-level</td>
<td>big</td>
<td>28/16</td>
</tr>
</tbody>
</table>
Advanced SIMD & FP Unit

- Supports ARMv8 ASIMD & Floating-Point Execution
- Supports optional Cryptography engine
Load/Store Unit

- Executes load & store instructions
- Encompasses the L1 data side memory system
- Services memory coherency requests from the L2 memory system
- Automatic hardware prefetcher that generates prefetches targeting the L1 D-cache and the L2 cache.
Outline

1. Single Core Architecture

2. Memory Subsystem

3. Multicores and Multiprocessors
   - Coherence
   - Synchronization
Memory Management Unit (MMU)

**MMU**: HW component for translating VA to PA

- Uses the most significant bits for translation, applications do not know that!
- Controls access permissions
- Controls table walk that accesses translation tables

**NOTE**: MMU is the HW unit that takes care of translation, TLB is just a cache of page table entries.
MMU Features

- 48-entry fully associative L1 instruction TLB
- 32-entry fully associative L1 data TLB for data loads and stores
- 4-way set associative 1024-entry L2 TLB in each processor
- Intermediate table walk caches
- TLB entries contain an ASID and VMID
  - Support context and virtual machine switches without flushes
- L1 & L2 TLB cache entries support multiple page sizes 4KB, 64KB and 1MB (plus 16MB for L2), of VA to PA mappings.
Each TLB contains besides the VA

- page size
- PA
- memory type and access permissions
- VMID
- ASID

Process of Memory Accesses:
When a processor generates a memory access the MMU

- Performs a lookup for the requested VA, current ASID & VMID in the relevant L1-I or L1-D TLB
- If there is a L1 TLB miss, it performs a lookup in the unified L2 TLB
- If L2 TLB is a miss too, performs a hardware translation table walk
L1 Organization

L1-I
- 48KB 3-way set-associative instruction cache.
- Fixed line length of 64 bytes
- Parity protection per 16 bits
- Instruction cache is PI - PT
- LRU cache replacement policy
- MBIST support

L1-D
- 32KB 2-way set-associative data cache
- Fixed line length of 64 bytes
- ECC protection per 32 bits
- Data cache that is PI - PT
- Out-of-order, speculative, non-blocking load requests to Normal memory and non-speculative, non-blocking load requests to Device memory
- LRU cache replacement policy
- Hardware prefetcher that generates prefetches targeting both the L1 data cache and the L2 cache
- MBIST support
<table>
<thead>
<tr>
<th>L1-I</th>
<th>L1-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 48KB 3-way set-associative instruction cache.</td>
<td>• 32KB 2-way set-associative data cache</td>
</tr>
<tr>
<td>• Fixed line length of 64 bytes</td>
<td>• Fixed line length of 64 bytes</td>
</tr>
<tr>
<td>• Parity protection per 16 bits</td>
<td>• ECC protection per 32 bits</td>
</tr>
<tr>
<td>• Instruction cache is PI - PT</td>
<td>• Data cache that is PI - PT</td>
</tr>
<tr>
<td>• LRU cache replacement policy</td>
<td>• Out-of-order, speculative, non-blocking load requests to Normal memory and non-speculative, non-blocking load requests to Device memory</td>
</tr>
<tr>
<td>• MBIST support</td>
<td>• LRU cache replacement policy</td>
</tr>
</tbody>
</table>

**Question:** What is the advantage and what is the disadvantage of using PI - PT Instruction and Data cache?
The L2 memory system consists of a tightly-coupled L2 cache and an integrated Snoop Control Unit (SCU)

**Characteristics:**

- Configurable L2 cache size of 512KB, 1MB, 2MB and 4MB.
- Fixed line length of 64 bytes.
- PI - PT
- Banked pipeline structures
- Inclusion property with L1 data caches
- Software-programmable pseudo-least-recently-used or pseudo-random cache-replacement policy
- Duplicate copies of the L1 data cache directories for coherency support
- Optional hardware prefetch support
- Optional Error Correction Code (ECC) support
- Register slice support for large L2 cache sizes to minimize impact on routing delays
- MBIST support
The L2 cache is partitioned into multiple banks to enable parallel operations

- The Tag array is partitioned into multiple banks to enable up to two requests to access different tag banks of the L2 cache simultaneously.
- Each tag bank is partitioned into multiple data banks to enable streaming accesses to the data banks. Each tag bank consists of four data banks.
Interesting Properties of Memory

• **Normal vs Device Memory:** Normal can be used for all code and most data regions in memory (RAM, Flash, ROM). Fewer restrictions, speculative access. Device is used for memory regions where accesses can have a side-effect (read to a FIFO or a timer).

• **Prefetching:**
  - L1-I: sequential prefetching, after a miss request the next sequential line
  - L1-D and L2: HW - prefetcher that targets both. Some Key features are:
    * Software-programmable prefetches on any instruction fetch L2 miss (up to 4). Prefetches are allocated into the L2 cache
    * Separate mechanisms to detect and prefetch:
      - Instructions: fetch consecutive cache lines on an L2 instruction fetch access.
      - Table walk: fetch the consecutive cache lines on an L2 table walk descriptor access.
• **Cache Disabling:** Clear bit that allows cache disabling for each cache independently

• **Cache miss:** Critical Word first policy

• **Write Back - Read/Write Allocate & Write Back - No Allocate:** Two policies according to the page type

• **Reliability Issues:**
  
  • Use of parity bits for error detection
  
  • Error Correction Codes (different for L1 and L2 caches)
  
  • Memory Built-In Self Test (MBIST), to verify memory functionality and test for faults. Pros: fast in parallel testing Cons: Extra area
Outline

1. Single Core Architecture
2. Memory Subsystem
3. Multicores and Multiprocessors
   - Coherence
   - Synchronization
Single Core vs. Multi-core vs. Multi-processor

Additional concerns in multicore systems:

- Cache Coherence
- Synchronization
There are two main ways to implement cache coherency in multi-core systems:

- Software managed coherency
- Hardware managed coherency
Cortex A72 uses a hybrid version of MOESI and MESI protocols. And there are several hardware modules that control cache coherence.

- The processor cluster contains a module called Snoop Control Unit (SCU).
  - SCU is similar to the Directory Based Coherence.
The processor cluster contains a Snoop Control Unit (SCU).

- Maintains coherency between L1 data caches.
- Arbitrates accesses to L2 interfaces, for both instructions and data.
- Has duplicated Tag RAMs to keep track of what data is allocated in each core’s L1 cache.
Snoop tag arrays reduces the amount of snoop traffic between L2 caches and L1 caches.

- If line is in Modified/Exclusive state, it belongs to the L1 memory system

- If the line is invalid or in the shared state in the Snoop Tag array, then the L2 cache can supply the data
Are there any other cache like structures that require coherence among multiple cores?
Are there any other cache like structures that require coherence among multiple cores?

- Translation Lookaside Buffers (TLBs)
- TLB coherence is managed by software (operating system)
- However, it requires broadcast operations for TLB entry invalidation in other cores
- Beside cache coherence, snoop control unit also provides broadcast operations for TLB caches.
Inter-cluster Coherence

Extending the coherency to multiple clusters requires a coherent bus protocol.

- **AXI Coherence Extensions (Implemented by Cache Coherent Interconnect (CCI))**
  - Any shared access to memory in one cluster can snoop into the caches of the other clusters.
  - Additionally, as in intra-cluster coherence, it can broadcast TLB coherence messages.
Synchronization and Memory Ordering

Memory ordering issues:

- Executing code on multiple cores
- Updating the shared data simultaneously
- Updating shared data structures such as page tables
Synchronization and Memory Ordering

Memory ordering issues:

• Executing code on multiple cores
• Updating the shared data simultaneously
• Updating shared data structures such as page tables

What is the memory model of ARMv8?

• Employs weakly-ordered memory model.
• Programmer should ensure explicit ordering of memory accesses.
Synchronization and Memory Ordering

Memory ordering issues:

- Executing code on multiple cores
- Updating the shared data simultaneously
- Updating shared data structures such as page tables

What is the memory model of ARMv8?

- Employs weakly-ordered memory model.
- Programmer should ensure explicit ordering of memory accesses.
- What might be the benefit of weakly-ordered memory model?
Synchronization and Memory Ordering

Memory ordering issues:

- Executing code on multiple cores
- Updating the shared data simultaneously
- Updating shared data structures such as page tables

What is the memory model of ARMv8?

- Employs weakly-ordered memory model.
- Programmer should ensure explicit ordering of memory accesses.
- *What might be the benefit of weakly-ordered memory model?*
  - We need to pay for the overhead of barriers when it is only needed.
- Instruction Synchronization Barrier (ISB)
- Data Memory Barrier (DMB)
- Data Synchronization Barrier (DSB)
• **ISB**: Used to ensure that any previously executed operations, such as writes to system control registers, have completed by the time the ISB completes.
• **ISB**: Used to ensure that any previously executed operations, such as writes to system control registers, have completed by the time the ISB completes

  • *What can be the effect of ISB in the instruction pipeline?*
• **ISB**: Used to ensure that any previously executed operations, such as writes to system control registers, have completed by the time the ISB completes

  • *What can be the effect of ISB in the instruction pipeline?*
    • Might cause flushing of the instruction pipeline
• **ISB**: Used to ensure that any previously executed operations, such as writes to system control registers, have completed by the time the ISB completes

  - *What can be the effect of ISB in the instruction pipeline?*
    - Might cause flushing of the instruction pipeline

• **DMB**: This barrier simply prevents re-ordering of data accesses across the barrier
• **ISB**: Used to ensure that any previously executed operations, such as writes to system control registers, have completed by the time the ISB completes

  • *What can be the effect of ISB in the instruction pipeline?*
    • Might cause flushing of the instruction pipeline

• **DMB**: This barrier simply prevents re-ordering of data accesses across the barrier

• **DSB**: Has the same ordering effect as in DMB. Also blocks the execution of any further instructions not only data accesses
• **Load-Acquire (LDAR):** All loads and stores that are after an LDAR in program order must be observed after the LDAR.

• **Store-Release (STLR):** All loads and stores preceding an STLR must be observed before the STLR.
Questions?
• Owned: This describes a line that is dirty and in possibly more than one cache. A cache line in the owned state holds the most recent, correct copy of the data. Only one core can hold the data in the owned state. The other cores can hold the data in the shared state.

• Exclusive: The cache line is present in this cache and coherent with main memory. No other copies of the memory location exist within other caches.
1. The Cortex-A53 cluster issues a Coherent Read Request.

2. The CCI passes the request to the Cortex-A53 processor to snoop into Cortex-A57 cluster cache.

3. When the request is received, cluster checks its data caches availability and responds with the required information.

4. If the requested data is in the cache, the CCI moves the data from the Cortex-A57 cluster to the Cortex-A53 cluster, resulting in a cache linefill in the Cortex-A53 cluster.

---

*aProgrammer’s Guide for ARMv8 2015*
Synchronization among threads may rely on the ability to tag particular addresses for exclusive access.

- **Load Exclusive (LDXR):** LDXR W—Xt, [Xn]

- **Store Exclusive (STXR):** STXR Ws, W—Xt, [Xn] where Ws indicates whether the store completed successfully. 0 = success.

- **Clear Exclusive access monitor (CLREX):** This is used to clear the state of the Local Exclusive Monitor.
Power Management

List of power domains:

- Each core in the device
- PCLKDBG domain:
  - Debug APB interface
  - CTI logic
  - CTM logic
- L2 cache and Snoop Tag RAMs
- A domain for:
  - L2 control
  - GIC CPU interface
  - Generic Timer logic
Power Management

- Controls both static and dynamic power dissipation
- Individual Core Power down Leakage power reduction by power-gating to the core (all core logics, SIMD, FP unit, L1 RAM, Debug, ETM, breakpoint and watchpoint logic)
- Processor powerdown without system driven L2 Flush All processor power domains are shut down and all state is lost. Lead core is defined as the last core to power down, or the first core to power up.
- Dormant Mode All processors, debug PCLKDBG, and L2 control logic are powered down. L2 cache RAMs are powered up and retain state.
- Debug Powerdown When debug facilities are not required, power is turned off to reduce leakage power.
- External debug over powerdown If any or all of the cores are powered down, the SoC can still use the debug facilities if the debug PCLKDBG domain is powered up.