Building High Performance, Power Efficient Cortex and Mali systems with ARM CoreLink

Robert Kaye
Agenda

- Once upon a time… ARM designed systems
- Compute trends
- Bringing it all together with CoreLink™ 400 Series
- What’s inside the box
  - Coherency, virtualization, end-to-end QoS
- Compute sub-systems
- Any questions?
Once Upon a Time… Over 20 Years Ago

ARM Design Philosophy

ARM Processors are:-
- Small in size
- Low cost
- Suitable as macrocells
- Efficient
- Low power consumption
- Low heat generation

The BBC Model B
The system includes:
- Keyboard / mouse
- Display
- Pen / cassette interface
- Analogue / 10 digit IO

Preserving the Pen

Advantages of ARM

Mission
Compute Trends – Clients & the Cloud

- Personal computing in the post-smartphone era has changed everything
  - Every screen is connected to the cloud
    - Connected life with presence = my services must follow me
  - Power budget – want to do more within the same budget
    - Software, security, camera as user interface, augmented reality…
    - What gets processed where is a function of bandwidth

- Technology implications:
  - CPU/GPU/VPU throughput, memory sub-system, software programmer’s model
  - Hardware must be built to meet evolving software requirements
Bringing It All Together

- It’s all about the system!
  - Coherency, virtualization, non-blocking & hierarchical interconnect, power management, end-to-end QoS
  - Software wants to see hardware as resources:
    - Details e.g. registers – pah!
    - Want common API’s – e.g. OpenCL
  - Software community want standardization of hardware resources
  - Hardware assistance in the right place improves consistency & software portability

- …system optimization is key
  - …and not just the CPU
ARM – Easy to Select the Right System IP

- We call it the CoreLink 400 Series
  - So let’s take a closer look at what’s inside the box…

<table>
<thead>
<tr>
<th>Product</th>
<th>Name</th>
<th>Headline Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Interconnect</td>
<td>NIC-400</td>
<td>Hierarchical network interconnect, improved clock and power management, AXI4 and APB4 support</td>
</tr>
<tr>
<td>Quality of Service</td>
<td></td>
<td>Enhanced bandwidth regulation</td>
</tr>
<tr>
<td>Thin Links</td>
<td></td>
<td>Point-to-point packetization - reduces wiring congestion</td>
</tr>
<tr>
<td>QOS Virtual Networks</td>
<td></td>
<td>Avoid cross-stream AND head-of-line blocking</td>
</tr>
<tr>
<td>Cache Coherent Interconnect</td>
<td>CCI-400</td>
<td>Dual cluster CCI - 2 ACE/3 ACE-Lite ports, virtualization signalling, barriers, clock gating</td>
</tr>
<tr>
<td>Dynamic Memory Controller</td>
<td>DMC-400</td>
<td>QoS, virtual networks, LPDDR2/DDR3 – smooth evolution to future memories e.g. WideIO</td>
</tr>
<tr>
<td>System Memory Management</td>
<td>MMU-400</td>
<td>Stage 2 memory translation, ARMv7 virtualization extensions compliant</td>
</tr>
<tr>
<td>Generic Interrupt Controller</td>
<td>GIC-400</td>
<td>Share interrupts across clusters, ARM v7 virtualization extensions compliant</td>
</tr>
</tbody>
</table>

**System IP for Cortex-A15 & Mali-T604**

- Cache Coherent Interconnect
  - Coherent caches shared by processors & I/O
  - AMBA® 4 coherency extensions
  - Non-blocking virtual networks
- New high efficiency memory controller
  - 1/2/4 channels @1066MHz
  - >90% interface utilization
  - LPDDR2/DDR3
- System MMU for I/O virtualization
- Network Interconnect for SoC connectivity
  - Reduced routing and power, end-to-end QoS
- Compute subsystems designed and optimized by ARM
System IP for Cortex-A15 & Mali-T604

- **Cache Coherent Interconnect**
  - Coherent caches shared by processors & I/O
  - AMBA® 4 coherency extensions
  - Non-blocking virtual networks

- **New high efficiency memory controller**
  - 1/2/4 channels @1066MHz
  - >90% interface utilization
  - LPDDR2/DDR3

- **System MMU and GIC for virtualization**

- **Network Interconnect for SoC connectivity**
  - Reduced routing and power, end-to-end QoS

- **Compute subsystems designed and optimized by ARM**
Why Coherency?

- More processors & More shared data
  - Increase in processor cores
  - Increase in accelerator engines including multimedia, e.g. 3D
  - Goal is more performance for less power

- Shared data needs to be managed – ensure consistency

- Three ways to guarantee coherency of shared data
  - Disable caching = *lowest performance*
  - Software managed coherency = *SW overhead for cache maintenance*
  - Hardware managed coherency = *highest performance, scalable*
New Applications Need Coherency

- OpenCL provides access to the vast processing power of Mali™-T604
- Applications include:
  - Video editing and effects
  - Camera & image processing (e.g. smile detection)
  - Image recognition (e.g. automotive lane detection)
  - Gesture recognition systems
  - Game engines (physics engines, particle physics)
  - Photorealistic ray tracing
- And Artificial Intelligence…
Hardware Coherency – Past and Present

- ARM MPCore™ processors support scaling up to quad core SMP
  - All processors see the same view of memory
  - Performance scaling for applications

- Accelerator Coherence Port (ACP) allows sharing of MPCore caches
  - Limited to one MPCore™ processor
  - Bandwidth shared with processor interface
    - Limited throughput

- Bringing hardware coherency to the system – AXI Coherency Extensions (ACE)
  - CoreLink CCI-400 Cache Coherent Interconnect - enables scalable coherency
  - Multiple processor clusters – up to 8 Cortex™-A15 cores
  - Multiple accelerator engines with increased bandwidth
Reduce Your Cache Maintenance Costs

- Without hardware coherency, software must manage caches
  - Flushing and invalidation of data requires many CPU cycles
  - Data is written to main memory (DDR)
  - This burns power, increases latency and occupies the CPU
  - Cache maintenance software is notoriously difficult to debug

- AMBA 4 ACE allows hardware to manage cache coherency
  - Caches do not need to be flushed or invalidated
  - Processor can do useful work instead, or enter low power state
  - External memory accesses are reduced
    - No wasted cache flushing
    - Shared data can now be read directly from processors caches!

Hardware coherency simplifies software & processor spends less time maintaining caches – good for power and performance
Why Virtualization?

- Virtualization: create multiple logical devices from one physical device
- Popek & Goldberg (1974) gave the requirements for virtualization
  - Equivalence / Fidelity
    Underlying H/W is transparent to S/W
  - Resource control / Safety
    System protection using sandboxing
  - Efficiency / Performance
    Low overhead virtual machine
- Meeting these 3 requirements within the SoC is a key driver in the ARM world
Virtualization Everywhere

- Servers
  - Widely used for consolidation of tasks on to CPUs
- Computing
  - Enables multiple client OS, e.g. Windows on Linux
- Motorola Evoke QA4, world’s first fully virtualized smartphone
  - Common H/W view presented to S/W across product range
- Mainstream games consoles
  - Implement virtual machines for legacy S/W
- Next gen STB and DTV hardware
  - Sandboxing of user space to protect device firmware
Virtualization in Mobile and Embedded

- Innovation driving change (EETimes, April 2010)
  "Low power processors now incorporate the same kind of hypervisor hardware acceleration enjoyed by desktop and server processors."

- Emerging virtualization uses
  - Split mobile personalities
  - Next generation mobile devices
  - DTV with internet apps
  - In-vehicle infotainment
  - Gaming systems

- 2010/11 mobile & embedded design starts
  - Including virtualization hardware support now
ARM Solution – Cortex-A15, MMU-400 and GIC-400

- ARM adds ‘virtualization extensions’ standard to ARMv7 architecture in 2010
- Cortex-A15 first processor with native ‘hypervisor mode’
- Mali-T604 GPU runs in virtual address space
- MMU-400 maps Mali accesses to physical address for the hypervisor
- MMU-400 virtualizes other masters
- GIC-400 virtualizes interrupts
CoreLink MMU-400 and GIC-400

- IO virtualization with distributed TLB maintenance messaging
- Stage 2 address translation for hypervisor support
- ARMv7 virtualization extension architecture compliant

- Generic Interrupt Controller for multiple Cortex-A15 clusters
- IRQs and FIQs securely managed by hypervisor for each OS
- ARMv7 virtualization extension architecture compliant
CoreLink Delivers Efficient Virtualization

- **Equivalence / Fidelity**
  - Cortex-A15 processor works with MMU-400 and GIC-400 to present a ‘virtual hardware view’ to S/W
  - H/W can be upgraded as required without modifying S/W

- **Resource control / Safety**
  - Hypervisor in complete control of the virtualized resources
  - Cortex-A15, MMU-400 and GIC-400 sandbox each OS

- **Efficiency / Performance**
  - Address translations are performed in hardware
    - More efficient and simpler than para-virtualization
    - No need to migrate device drivers into the hypervisor
  - No need to trap and process IO accesses or interrupts
  - Evidence from typical DMA test case:
    - 1.5% overhead vs. 36% overhead for S/W only solution
Why End-to-End Quality of Service?

- Systems use external memory
  - Higher data – GB/s
  - Higher frequency
  - Contention for scarce bandwidth

- Engineering challenges
  - Need low latency for CPU
  - High bandwidth for GPU
  - LCDC needs deterministic latency

- So what?
  - Need common QoS scheme across interconnect & memory controller
  - Maximise performance & power efficiency
End-to-End Quality of Service

Interconnect

- Traffic regulation on entry
  - Maximum bandwidth limits
  - Outstanding transaction management

- Dynamic priority
  - Uses QoS value in NIC-301, NIC-400
  - Changes priority to meet target latency

- Virtual Networks
  - Remove blocking through system

Memory Controller

- Scheduler aims for high memory efficiency whilst meeting QoS requirements
- Support for latency regulation and arbitration with QoS value
- Timeout mechanism for streaming and real-time traffic
- Separate read and write queues
SoC Energy Efficiency is a Function of…

- Many cores high performance computing
  - Energy efficient Cortex, Mali processors
  - But significant % of energy used in the memory system, especially off-chip
- Efficient use of limited off-chip memory bandwidth
  - DMC-400, >90% of max theoretical utilization
  - CCI-400, NIC-400, QoS, QVN keep processors performing without compromising bandwidth utilization
- Higher cache utilization by software
  - Coherency CCI-400 enables more on-chip accesses, less off-chip accesses
  - Lowering power and raising performance!
The ARM Compute Sub-System

- **For software**
  - Common Cortex-A + Mali graphics and video platform
    - For maximum software performance & availability
  - CoreSight provides visibility
    - For software debug and performance optimisation
  - TrustZone and System MMU
    - Hardware virtualisation to protect applications

- **System PPA**
  - Cortex-A5/A8/A9/A15, Mali-55/200/400 MP, Mali-VE3/6
    - A range of scalable processing performance to fit your power & area budget
  - CoreLink Network Interconnect and Dynamic Memory Controllers
    - To maximise efficient use of shared main memory
  - Verification and Performance Exploration (VPE) tool
    - For early and rapid system design trade-off decisions

- **Ease of use**
  - AMBA Designer
    - Fast and reliable IP configuration and sub-system assembly
  - Example system designs for faster time to market
  - Support & maintenance from a single quality IP provider
CoreLink 300 Series for AMBA 3

- NIC-301 Network Interconnect
  - Hierarchical design
  - Advanced Quality of Service (QoS) for performance and latency

- Level 2 Cache Controller
  - Increase CPU performance
  - Reduce external memory accesses

- Dynamic Memory Controllers
  - LPDDR2, DDR2
  - LPDDR, DDR, NVM

- Programmable DMA Controller
  - Off load the CPU
  - Multi-channel

Raising SoC performance and power efficiency
Summary – It’s All About the System

- Coherency, virtualization, non-blocking & hierarchical interconnect, power management and end-to-end QoS

- Software wants to see hardware as resources
  - Want common API’s – e.g. OpenCL

- The software community want standardization of hardware resources

- Hardware assistance in the right place simplifies the software programmers view; improving consistency & software portability and reducing power consumption

- System optimization is key…and not just the CPU
Thank You

Please visit www.arm.com for ARM related technical details

For any queries contact < Salesinfo-IN@arm.com >