Extensions to the ARMv7-A Architecture

David Brash
Architecture Program Manager, ARM Ltd.
Hotchips, August 2010
Introduction

- The ARM architecture is now pervasive in many markets

- The architecture has evolved to meet changing needs
  - ARMv7 the latest variant
  - A,R,M profiles to tailor features against requirements

- Mainframe => desktop => “the ARM world” evolution
  - Increased functionality and performance at lower power
Today’s announcements

- Two major additions to ARMv7-A
  - **Virtualization Extension**
    - New privilege level for the hypervisor
    - 2-stage address translation - OS and hypervisor levels
    - Complements the Security Extensions (TrustZone®)
  - **Large Physical Address Extension (LPAE)**
    - Translation of 32-bit virtual to ≤ 40-bit physical addresses
      (ease pressure on 4GB limit for IO and memory)
- Other architecture support
  - Generic Interrupt Controller - GICv2
  - Generic timer
  - System MMU
The ARMv7-A Virtualization Extensions

Popek and Goldberg summarized the concept in 1974:
"Formal Requirements for Virtualizable Third Generation Architectures". *Communications of the ACM* 17

- **Equivalence/Fidelity**
  - A program running under the hypervisor should exhibit a behaviour essentially identical to that demonstrated when running on an equivalent machine directly.

- **Resource control / Safety**
  - The hypervisor should be in complete control of the virtualized resources.

- **Efficiency/Performance**
  - A statistically dominant fraction of machine instructions must be executed without hypervisor intervention.
Terminology

“Virtualization is execution of software in an environment separated from the underlying hardware resources”

**Full virtualization**
Where a sufficiently complete simulation of the underlying hardware exists, to allow software, typically guest operating systems, to run unmodified

**Para-virtualization**
Where guest software is expressly modified

System without virtualization

- App 1
- App 2
- Operating System
- Hardware

System with virtualization

Virtual Machine Monitor (VMM, or Hypervisor)

Virtual Machine 1
- App 1
- App 2
- Guest OS 1

Virtual Machine 2
- App 1
- App 2
- Guest OS 2

Hardware
ARM hypervisor support philosophy

- Virtual machine (VM) scheduling and resource sharing
  - New **Hyp** mode for Hypervisor execution

- Minimise Hypervisor intervention for “routine” GuestOS tasks
  - Guest OS page table management
  - Interrupt control
  - Guest OS Device Drivers

- Syndrome support for trapping key instructions
  - GuestOS load/store emulation
  - Privileged control instructions

- System instructions \( (\text{MRS, MSR}) \) to read/write key registers
- Virtualized ID register management
ARMv7-A: Exception levels

Non-secure (Normal) state
NS-bit == ‘1’

NS User
App1
App2

NS Priv
GuestOS1
GuestOS2

NS Hyp
VMM

Secure state
NS-bit == ‘0’

S User
Sec App1
Sec App2

S Priv
Secure OS

Exception Entry
Exception Return

S Monitor
TrustZone Monitor

Non-secure (Normal) state
NS-bit == ‘1’

Secure state
NS-bit == ‘0’

Exception Entry
Exception Return

S Monitor
TrustZone Monitor
ARMv7-A Virtualization - key features 1

- Trap and control support (HYP mode)
  - Rich set of trap options (TLB/cache ops, ID groups, instructions)

- Syndrome register support

- EC: exception class (instr type, I/D abort into/within HYP)
- IL: instruction length (0 == 16-bit; 1 == 32-bit)
- ISS: instruction specific syndrome (instr fields, reason code, ...)
ARMv7-A Virtualization - key features 2

- Dedicated Exception Link Register (ELR)
  - Stores preferred return address on exception entry
    - New instruction – \texttt{ERET} – for exception return from HYP mode
  - Other modes overload exception model and procedure call LRs
    - R14 used by exception entry \texttt{BL} and \texttt{BLX} instructions

- Address translation
2-stage address translation

Faults disambiguated
• Stage 1 => OS fault handling
• Stage 2 => Hyp mode

Implementation note: TLBs can merge Stage 1 and Stage 2 translation information (VA => PA)
LPAE – Stage 1 (VA => {I}PA)

- Managed by the OS
- 64-bit descriptors, 512 entries per table
  - 4KB table size == 4KB page size
- 1 or 2 Translation Table Base Registers
- 3 levels of table supported
  - up to 9 address bits per level
    - 2 bits at Level 1
    - 9 bits at Levels 2 and 3

Page Table Entry:

<table>
<thead>
<tr>
<th>6</th>
<th>3</th>
<th>5</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>Level 2 table index</td>
<td>Level 3 table (page) index</td>
<td>Page offset address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Table override bits (NS, XN, PXN, AP[1:0])
- Block/Page XN, PXN bits
- Block/Page 16x entry contiguous hint
- IGNORED bits

T1SZ ≤ T0SZ
If T1SZ == T0SZ == 0 then TTBR0 always used

TTBR0 space

TTBR1 space

Not mapped (Fault)

TTCR.T1SZ

TTCR.T0SZ

Valid

Memory type
Cache policy
Shareability, AF, nG, NS flags
Access permissions

Block/Table

Block/page
LPAE – Stage 2 (IPA => PA)

- Managed by the VMM / hypervisor
- Same table walk scheme as stage 1, now up to 40-bit input address:
  - 2x contiguous 4KB tables allowed at L1; support $2^{40}$ address space
  - Up to 1-16 contiguous tables allowed at L2; support $2^{30} - 2^{34}$ address space

1x Translation Table Base register (VTTTBR)

Page table entry:

- Block / page only
  - XN bit
  - 16x entry contiguous hint
  - IGNORED bits

- Block/Table
  - Memory type
  - Cache policy
  - Shareability, AF bit
  - R/W Access permissions

Valid
Example: Stage-2 table walk

<table>
<thead>
<tr>
<th>3</th>
<th>9</th>
<th>3</th>
<th>4</th>
<th>3</th>
<th>0</th>
<th>2</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPA Bits &lt;39:30&gt;</td>
<td>IPA Bits &lt;29:21&gt;</td>
<td>IPA Bits &lt;20:12&gt;</td>
<td>IPA Bits &lt;11:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPA Bits &lt;33:21&gt;</td>
<td>IPA Bits &lt;20:12&gt;</td>
<td>IPA Bits &lt;11:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- VTTBR
- 4KB L1 table
- 4KB L1 table
- 1-2 tables
- 4KB L2 table
- 1-16 tables
- 4KB L2 table
- 4KB L3 table
- Memory
Interrupt Controller

- ARM standardising on the “Generic Interrupt Controller” [GIC]
  - Supports the ARMv7 Security Extension
    - Interrupt groups support (Nonsecure) IRQ and (Secure) FIQ split
    - *Distributor* and *cpu interface* functionality

- GICv2 adds support for a *virtualized cpu interface*
  - VMM manages physical interface & queues entries for each VM.
  - GuestOS (VM) configures, acknowledges, processes and completes interrupts directly. Traps to VMM where necessary.
  - Hypervisor can virtualize FIQs from the physical (Nonsecure, IRQ) interface.

- \{I,A,F\} masks for Hypervisor (physical) and Guest (virtual) interrupts
Interrupt Handling – GICv2

**Interrupt sources:**
- Software
- Private (per CPU) peripheral
- Shared peripheral

**Distributor - control and configuration**
- Assignment (≤8 CPU interfaces)
- Interrupt grouping (Grp0, Grp1)

**CPU Interface0**
- CPU-specific control + state
- Interrupt ACK / END (retire)
- Grp0: (Secure*, FIQ)
- Grp1: IRQ always

**CPU InterfaceN**
- CPU-specific control + state
- Interrupt ACK / END (retire)
- Grp0: (Secure*, FIQ)
- Grp1: IRQ always

**Hypervisor – virtualized distributor**
- Grp1 physical => Grp1/Grp0 virtual

**Virtual CPU InterfaceN**
- CPU-specific control + state
- Interrupt ACK / END (retire)

**Virtual interrupt management**
- Interrupt queues (list registers)

**Virtualization support**

**NEW**

Secure handler (monitor mode)

OS FIQ handler

OS IRQ handler

**GuestOS FIQ handler**

**GuestOS IRQ handler**
System MMU

- System MMU architecture translation options from pre-set tables
  - No Translation
  - VA -> IPA (Stage 1) or IPA->PA (Stage 2) only
  - VA->PA (Stage1 and Stage2)

- Can share page tables with ARM cores – relate contexts to VMIDs/ASIDs

- Architecture specification will be published 1H-2011
  - ARM planning to support AMBA® based solutions in 2011 timeframe
Generic timer

- Shared “always on” counter
- Fast read access for reliable distribution of time.
- \( \geq \) ComparedValue or \( \leq 0 \) timer event configuration support
- Maskable interrupts
- Offset capability for virtual time
- Event stream support
- Hypervisor trap support
Hypervisor - boot flow by example

Non-Secure

- ARM Non-Secure User mode
- ARM Non-Secure Privileged modes (SVC, IRQ, ...)
- ARM Secure SVC mode
- NEW

Secure

- Power on Reset
- Software flow
- boot code or Startup code
- Enter monitor mode
- Monitor software
- ARM Secure Monitor mode

Guest OS1

- Apps

Guest OS2

- Apps

Hypervisor

Guest OS1

NEW

User mode

NEW

Privileged modes

NEW

NEW
Eco-system

- Specification details available end of this quarter
  - ARM ARM RevC (preliminary data): infocenter.arm.com

- ARM’s first implementation well advanced
  - Codenamed **Eagle** – expect to hear more this year

- 3rd party engagement
  - Review of the specifications
  - Early access to an architecture model
    - Development of full and para-virtualization solutions underway
  - Engaging with the industry experts:

  - Enea
  - Mentor Graphics
  - Green Hills Software
  - Open Kernel Labs
  - VMware
Summary

- A natural progression of well established features applied to an architecture long associated with low power.

- ‘Classic’ uses and solutions will apply to ARM markets:
  - Low cost/low power server opportunities
  - Application OS + RTOS/microkernel smart mobile devices
  - Feature split by ‘Manufacturers’ versus User’ VM environment
    - Automotive, home, ...

- Opportunities for new use cases?
  - Today’s entrepreneurs/ideas
    => tomorrow’s major businesses
Thank you