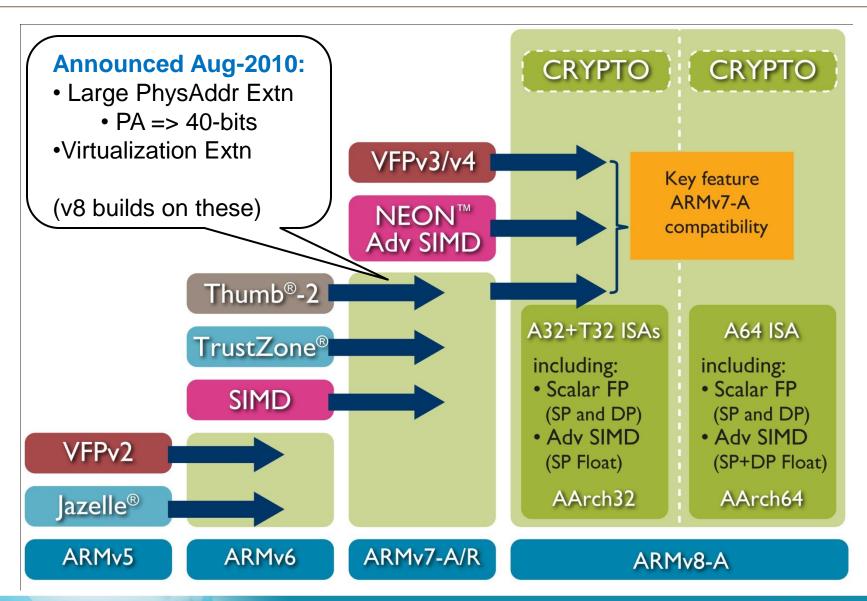
ARMv7-A Architecture Overview

David Brash
Architecture Program Manager, ARM Ltd.





ARM Architecture roadmap

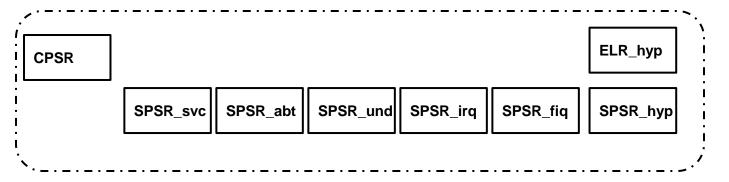


ARMv7-A registers

R0	LRO :	R0	R0][<u>R0</u>][R0	[R0]
R1	R1	R1	R1][R1	<u> R1</u>	R1
R2	<u>R2</u>	LR2	R2	R2	R2	<u>R2</u>
R3	R3 .	R3	R3	R3	l R3	R3
R4	<u>R4</u>	<u>R4</u>][R4	<u> R4</u>	R4
R5	<u>R5</u>	R5	L R5	!L R5	<u> R5</u>	<u>R5</u>
R6	<u>R6</u>	<u>R6</u>	R6	R6	<u> R6</u>	R6
R7	[R7]	<u>R7</u>][R7	R7	[R7]
R8	R8	<u>R8</u>		R8	R8 fiq	R8
R9	R9	R9	R9	R9	R9 fiq	R9
R10	<u>R10</u>	LR10	R10	R10	R10 fig	R10
R11	R11	R11	R11	R11	R11 fig	R11
R12	R12	R12	R12	R12	R12 fiq	R12
R13 (SP)	SP svc	SP abt	SP und	SP irq	SP fiq	SP hyp
R14 (LR)	LR svc	LR abt	LR und	d LR irq	LR fiq	LR (BL)

CPSR bits

- Flags: NZCVQ
- IT[7:0] status bits
- GE[3:0] Adv SIMD flags
- J (Jazelle)
- T (Thumb)
- E (endian)
- I, A, F masks
- M[4:0] (mode)



Classic ARM MMU

- 32-bit physical address space
- 2-level translation tables
 - Pointed to by TTBR0 (user mappings) and TTBR1 (kernel mappings but with restrictions to the user/kernel memory split)
 - 32-bit page table entries
- 1st level contains 4096 entries (4 pages for PGD)
 - 1MB section per entry or
 - Pointer to a 2nd level table
 - Implementation-defined 16MB supersections
- 2nd level contains 256 entries pointing to 4KB page each
 - 1KB per 2nd level page table
- ARMv6/v7 introduced TEX remapping
 - Memory type becomes a 3-bit index



Classic ARM MMU (cont'd)

- Other features
 - XN (eXecute Never) bit
 - Different memory types: Normal (cacheable and non-cacheable),
 Device, Strongly Ordered
 - Shareability attributes for SMP systems
- ASID-tagged TLB (ARMv6 onwards)
 - Avoids TLB flushing at context switch
 - 8-bit ASID value assigned to an mm_struct
 - Dynamically allocated (there can be more than 256 processes)



Classic ARM MMU Limitations

- Only 32-bit physical address space
 - Growing market requiring more than 4GB physical address space (both RAM and peripherals)
 - Supersections can be used to allow up to 40-bit addresses using 16MB sections (implementation-defined feature)
- Prior to ARMv6, not a direct link between access permissions and Linux PTE bits
 - Simplified permission model introduced with ARMv6 but not used by Linux
- 2nd level page table does not fill a full 4K page
- ARM Linux workarounds
 - Separate array for the Linux PTE bits
 - 1st level entry consists of two 32-bit locations pointing to 2KB 2nd level page table entries



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.



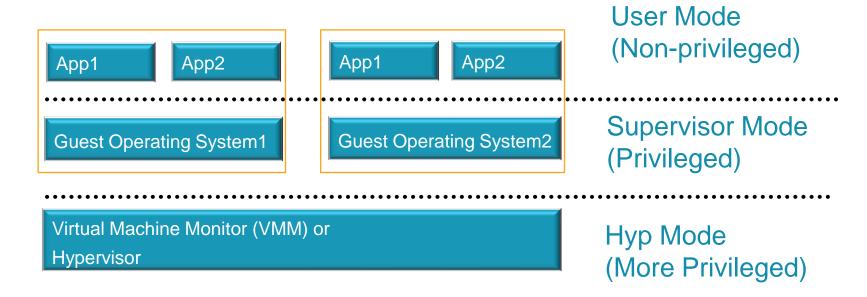
ARM hypervisor support philosophy

- 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 (MRS, MSR) to read/write key registers
- Virtualized ID register management



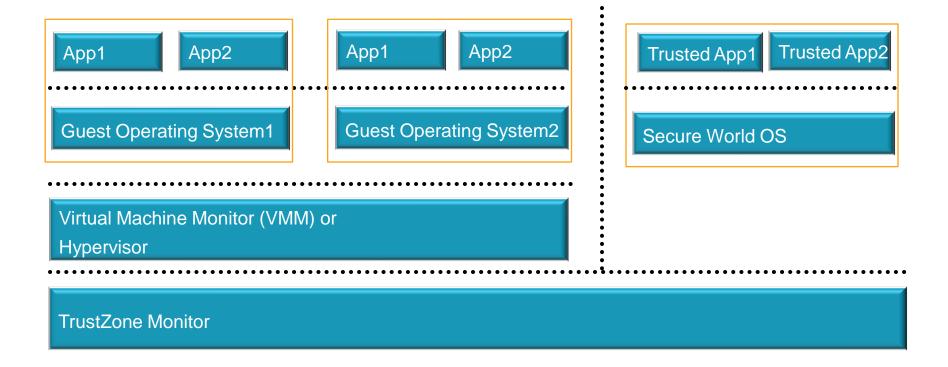
ARM virtualization extension - modes

- A new privilege layer hypervisor mode
 - Guest OS kernel given same privilege structure as for a nonvirtualized environment
 - Can run the same instructions
 - Hypervisor mode has higher privilege
 - VMM can control a wide range of OS accesses to hardware



Hypervisor mode and security extensions

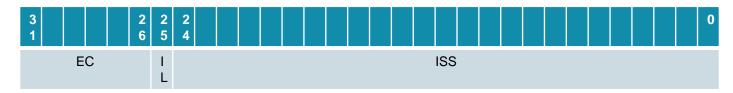
- Hypervisor mode applies to normal world
- Secure world supports a single virtual machine
- Monitor mode controls transition between worlds





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

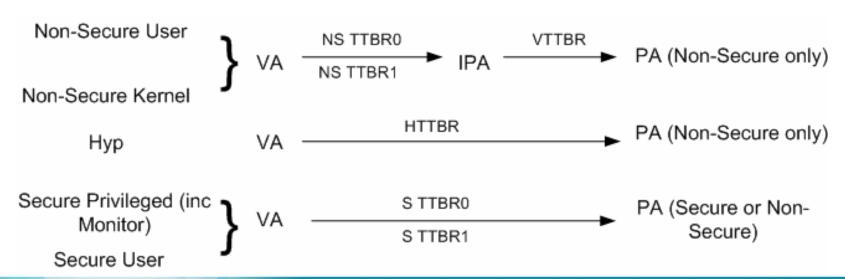


- EC: exception class (instr type, I/D abort into/within HYP)
- IL: instruction length (0 == 16-bit; 1 == 32-bit)
- ISS: instruction specific syndrome (instr fields, reason code, ...)

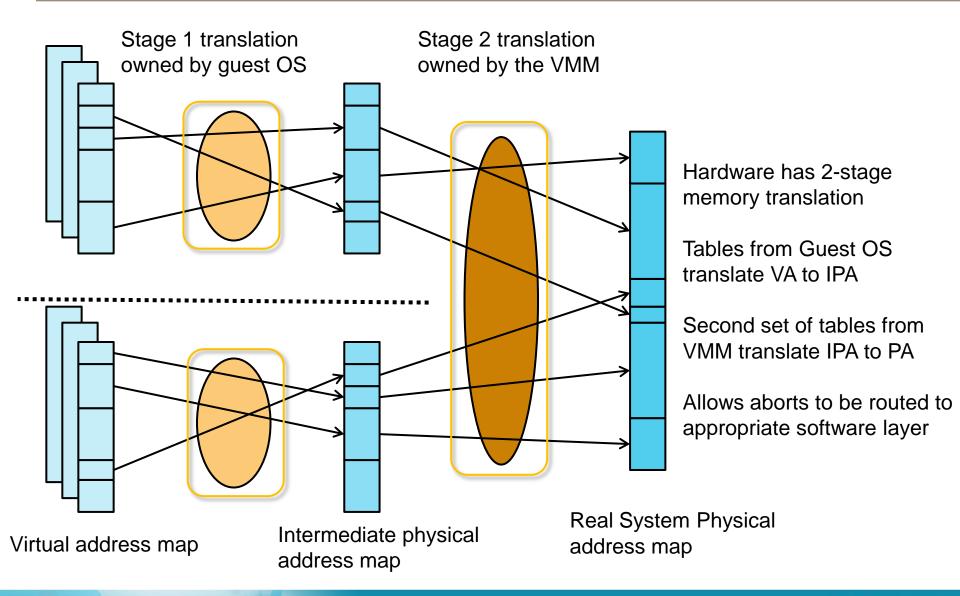
ARMv7-A Virtualization - key features 2

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

Address translation

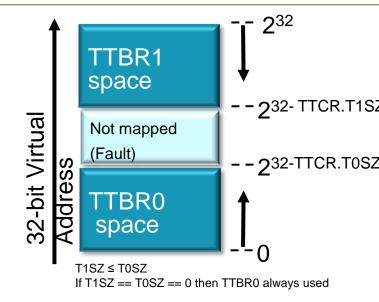


Two layers of address translation



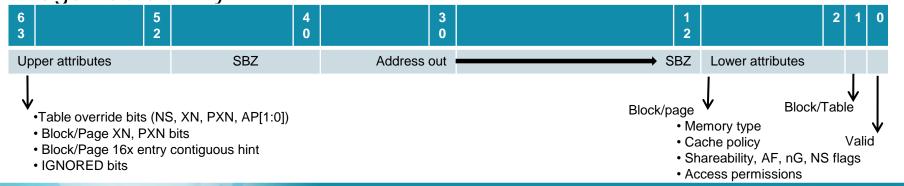
$LPAE - Stage 1 (VA => {I}PA)$

- 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



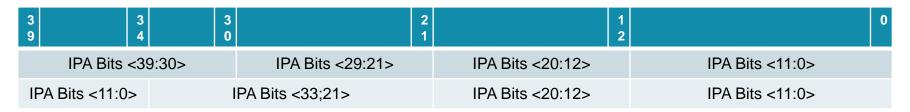
31:30	VA Bits <29:21>	VA Bits <20:12>	VA Bits <11:0>
L1	Level 2 table index	Level 3 table (page) index	Page offset address

Page Table Entry:

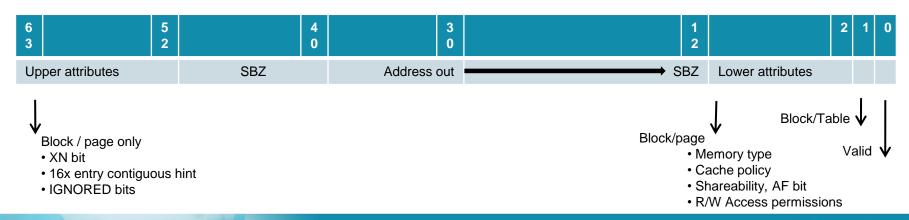


LPAE – Stage 2 (IPA => PA)

- 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 2⁴⁰ address space
 - Up to 1-16 contiguous tables allowed at L2; support 2³⁰ 2³⁴ address space



- 1x Translation Table Base register (VTBR)
- Page table entry:



Linux Support for ARM LPAE

Catalin Marinas ELC Europe 2011



ARM LPAE Features

- 40-bit physical addresses (1TB)
- 40-bit intermediate physical addresses (guest PA space)
- 3-level translation tables
 - Pointed to by TTBR0 (user mappings) and TTBR1 (kernel mappings)
 - Not as restrictive on user/kernel memory split (can use 3:1)
 - With 1GB kernel mapping, the 1st level is skipped
 - 64-bit entries in each level
- 1st level contains 4 entries (stage 1 translation)
 - 1GB section or
 - Pointer to 2nd level table
- 2nd level contains 512 entries (4KB in total)
 - 2MB section or
 - Pointer to 3rd level



ARM LPAE Features (cont'd)

- 3rd level contains 512 entries (4KB)
 - Each addressing a 4KB range
 - Possibility to set a contiguity flag for 16 consecutive pages
- LDRD/STRD (64-bit load/store) instructions are atomic on ARM processors supporting LPAE
- Only the simplified page permission model is supported
 - No kernel RW and user RO combination
- Domains are no longer present (they have already been removed in ARMv7 Linux)
- Additional spare bits to be used by the OS
- Dedicated bits for user, read-only and access flag (young) settings

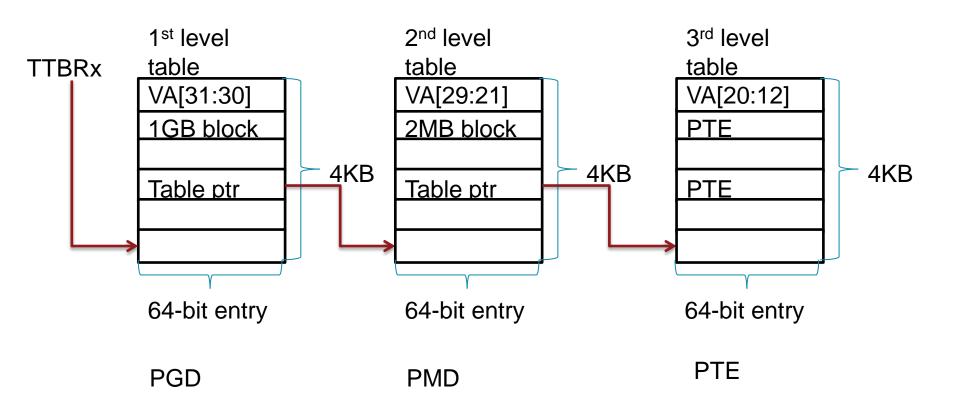


ARM LPAE Features (cont'd)

- ASID is part of the TTBR0 register
 - Simpler context switching code (no need to deal with speculative TLB fetching with the wrong ASID)
 - The Context ID register can be used solely for debug/trace
- Additional permission control
 - PXN Privileged eXecute Never
 - SCTLR.WXN, SCTLR.UWXN prevent execution from writable locations (the latter only for user accesses)
 - APTable restrict permissions in subsequent page table levels
 - XNTable, PXNTable override XN and PXN bits in subsequent page table levels
- New registers for the memory region attributes
 - MAIR0, MAIR1 32-bit Memory Attribute Indirection Registers
 - 8 memory types can be configured at a time



ARM LPAE Features



Linux and ARM LPAE

- Linux + ARM LPAE has the same memory layout as the classic MMU implementation
 - Described in Documentation/arm/memory.txt
 - 0..TASK_SIZE user space
 - PAGE_OFFSET-16M..PAGE_OFFSET-2M module space
 - PAGE_OFFSET-2M..PAGE_OFFSET highmem mappings
- Highmem is already supported by the classic MMU
 - Memory beyond 4G is only accessible via highmem
 - Page mapping functions use pfn (32-bit variable, PAGE_SHIFT == 12, maximum 44-bit physical addresses)
- Original ARM kernel port assumed 32-bit physical addresses
 - LPAE redefines phys_addr_t, dma_addr_t as u64 (generic typedefs)
 - New ATAG_MEM64 defined for specifying the 64-bit memory layout



- Hard-coded assumptions about 2 levels of page tables
 - PGDIR_(SHIFT|SIZE|MASK) references converted to PMD_*
- swapper_pg_dir extended to cover both 1st and 2nd levels of page tables
 - 1 page for PGD (only 4 entries used for stage 1 translations)
 - 4 pages for PMD
 - init mm.pgd points to swapper pg dir
- TTBR0 used for user mappings and always points to PGD
- TTBR1 used for kernel mapping:
 - 3:1 split TTBR1 points to 4th page of PMD (2 levels only, 1GB)
 - Classic MMU does not allow the use of TTBR1 for the 3:1 split
 - 2:2 split TTBR1 points to 3rd PGD entry
 - 1:3 split TTBR1 points to 1st PGD entry



- The page table definitions have been separated into pgtable*-2level.h and pgtable*-3level.h files
 - Few PTE bits shared between classic and LPAE definitions
 - Negated definitions of L_PTE_EXEC and L_PTE_WRITE to match the corresponding hardware bits
 - Memory types are the same and they represent an index in the TEX remapping registers (PRRR/NMRR or MAIR0/MAIR1)
- The proc-v7.S file has been duplicated into proc-v7lpae.S
 - Different register setup for TTBRx and TEX remapping (MAIRx)
 - Simpler cpu_v7_set_pte_ext (1:1 mapping between hardware and software PTE bits)
 - Simpler cpu_v7_switch_mm (ASID switched with TTBR0)
- Current ARM code converted to pgtable-nopud.h
 - Not using 'nopmd' with classic MMU



- Lowmer is mapped using 2MB sections in the 2nd level table
 - PGD and PMD only allocated from lowmem
 - PTE tables can be allocated from highmem
- Exception handling
 - Different IFSR/DFSR registers structure and exception numbering arch/arm/mm/fault.c modified accordingly
 - Error reporting includes PMD information as well
- PGD allocation/freeing
 - Kernel PGD entries copied to the new PGD during pgd_alloc()
 - Modules and pkmap entries added to the PMD during fault handling
- Identity mapping (PA == VA)
 - Required for enabling or disabling the MMU secondary CPU booting, CPU hotplug, power management, kexec



- Uses pgd_alloc() and pgd_free()
- When PHYS_OFFSET > PAGE_OFFSET, kernel PGD entries may be overridden
- swapper_pg_dir entries marked with an additional bit L_PGD_SWAPPER
 - pgd_free() skips such entries during clean-up

Current Status and Future

- Initial development done on software models
 - Tested on real hardware (FPGA and silicon)
- Parts of the LPAE patch set already in mainline
 - Mainly preparatory patches, not core functionality
 - Aiming for full support in Linux 3.3
- Hardware supporting LPAE
 - ARM Cortex-A15, Cortex-A7 processors
 - TI OMAP5 (dual-core ARM Cortex-A15)
- Other developments
 - KVM support for Cortex-A15 implemented by Christoffer Dall at Columbia University
 - Uses the Virtualisation extensions together with the LPAE stage 2 translations



Reference

- ARM Architecture Reference Manual rev C
 - Currently beta, not publicly available yet
- Specifications publicly available on ARM Infocenter
 - http://infocenter.arm.com/
 - ARM architecture -> Reference Manuals -> ARMv7-AR LPA Virtualisation Extensions
- Linux patches ARM architecture development tree
 - Hosts the latest ARM architecture developments before patches are merged into the mainline kernel
 - git://github.com/cmarinas/linux.git
 - When the kernel.org accounts are back
 - git://git.kernel.org/pub/scm/linux/cmarinas/linux-arm-arch.git



ARM Architecture System Features

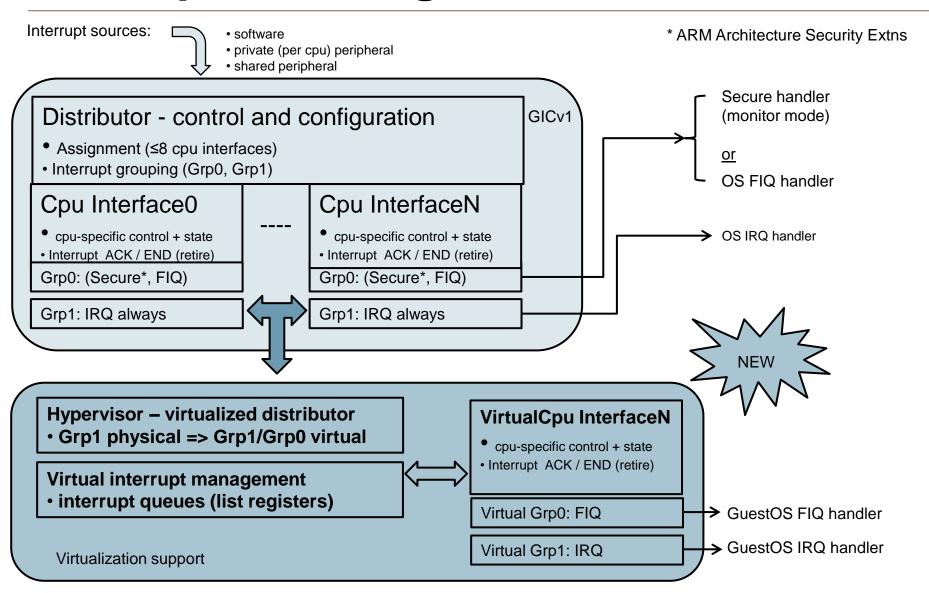


Interrupt Controller

- 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.



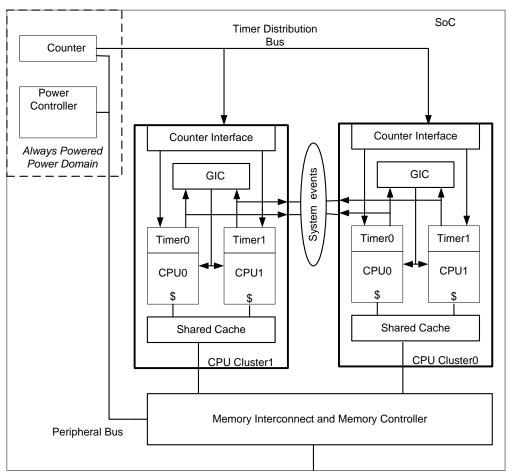
Interrupt Handling – GICv2



Generic Timer

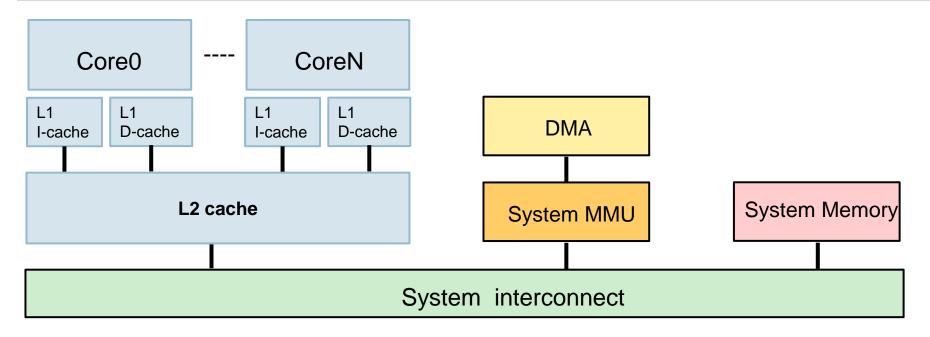
- Shared "always on" counter
- Fast read access for reliable distribution of time.
- ≥ ComparedValue or ≤ 0 timer event configuration support
- Maskable interrupts
- Offset capability for virtual time
- Event stream support
- Hypervisor trap support

Implementation example:





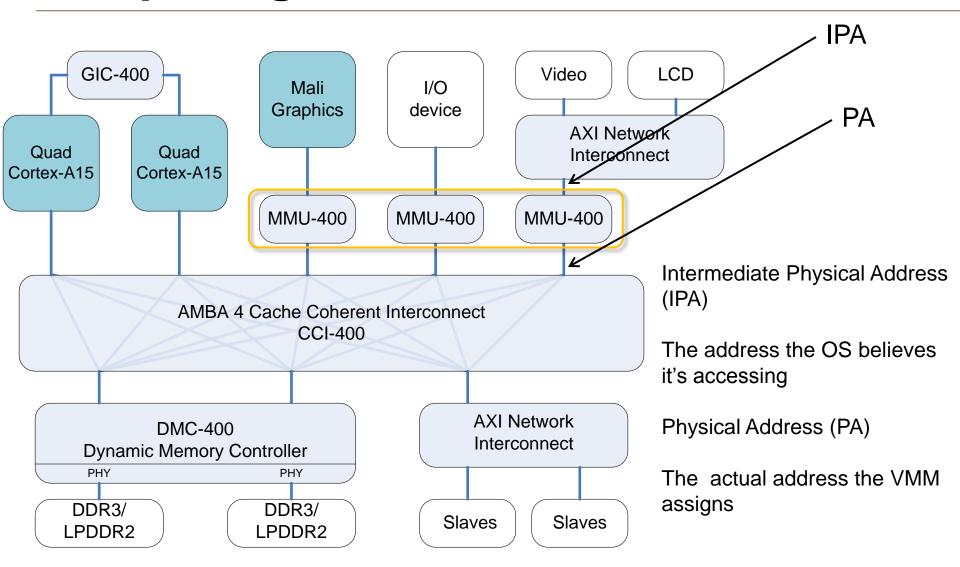
System MMU



- 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



Completing the SoC



To summarize: ARMv7-A additions

- 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/ideastomorrow's major businesses





Thank you

