

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

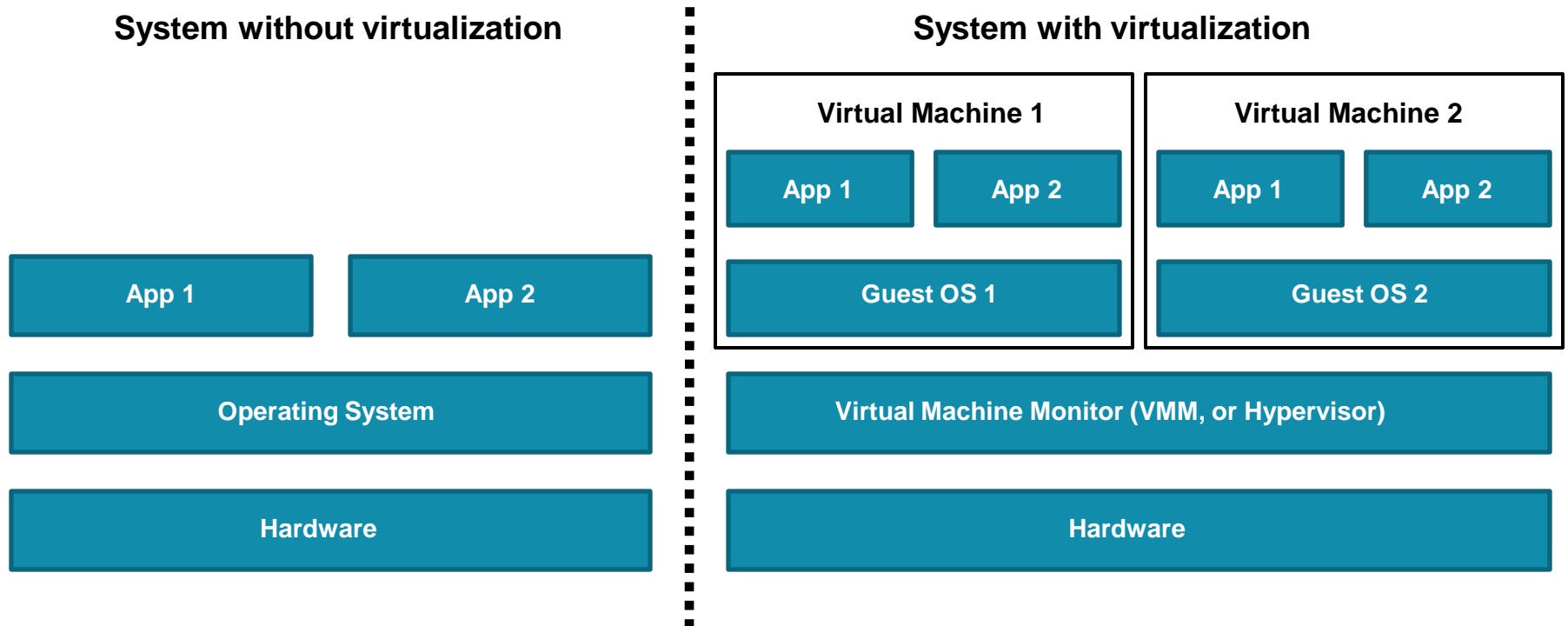
“Virtualization is execution of software in an environment separated from the underlying hardware resources”

Full virtualization

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

Para-virtualization

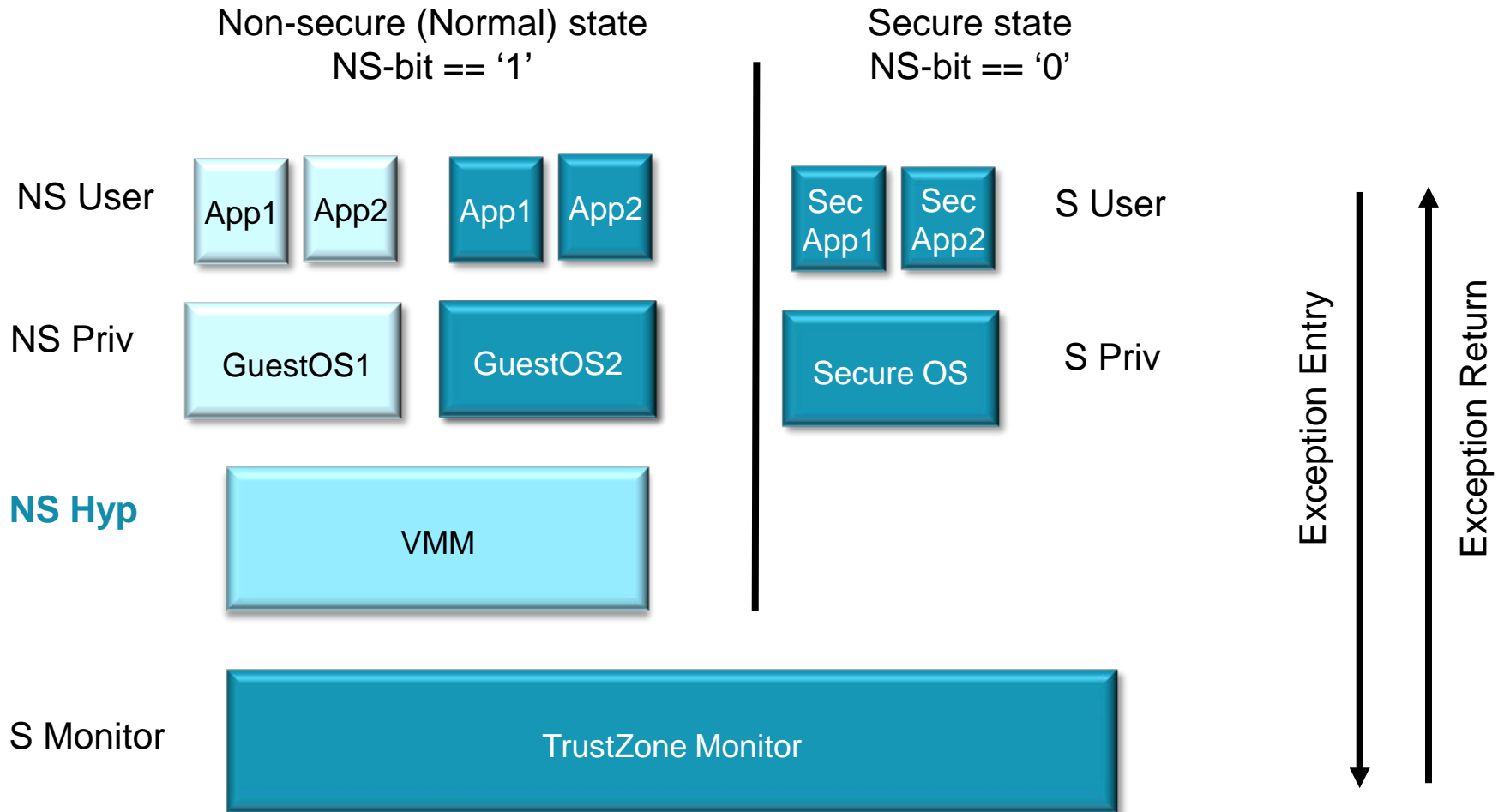
Where guest software is expressly modified



ARM hypervisor support philosophy

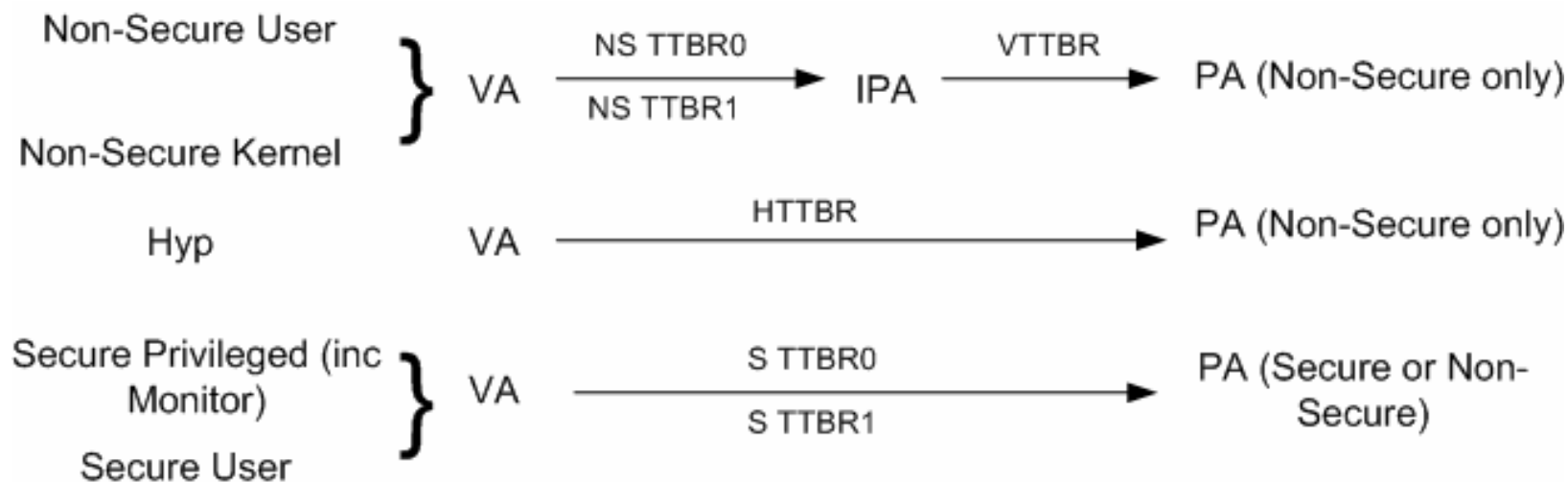
- Virtual machine (VM) scheduling and resource sharing
 - New **Hyp** mode for Hypervisor execution
- Minimise Hypervisor intervention for “routine” GuestOS tasks
 - Guest OS page table management
 - Interrupt control
 - Guest OS Device Drivers
- Syndrome support for trapping key instructions
 - GuestOS load/store emulation
 - Privileged control instructions
- System instructions (MRS, MSR) to read/write key registers
- Virtualized ID register management

ARMv7-A: Exception levels

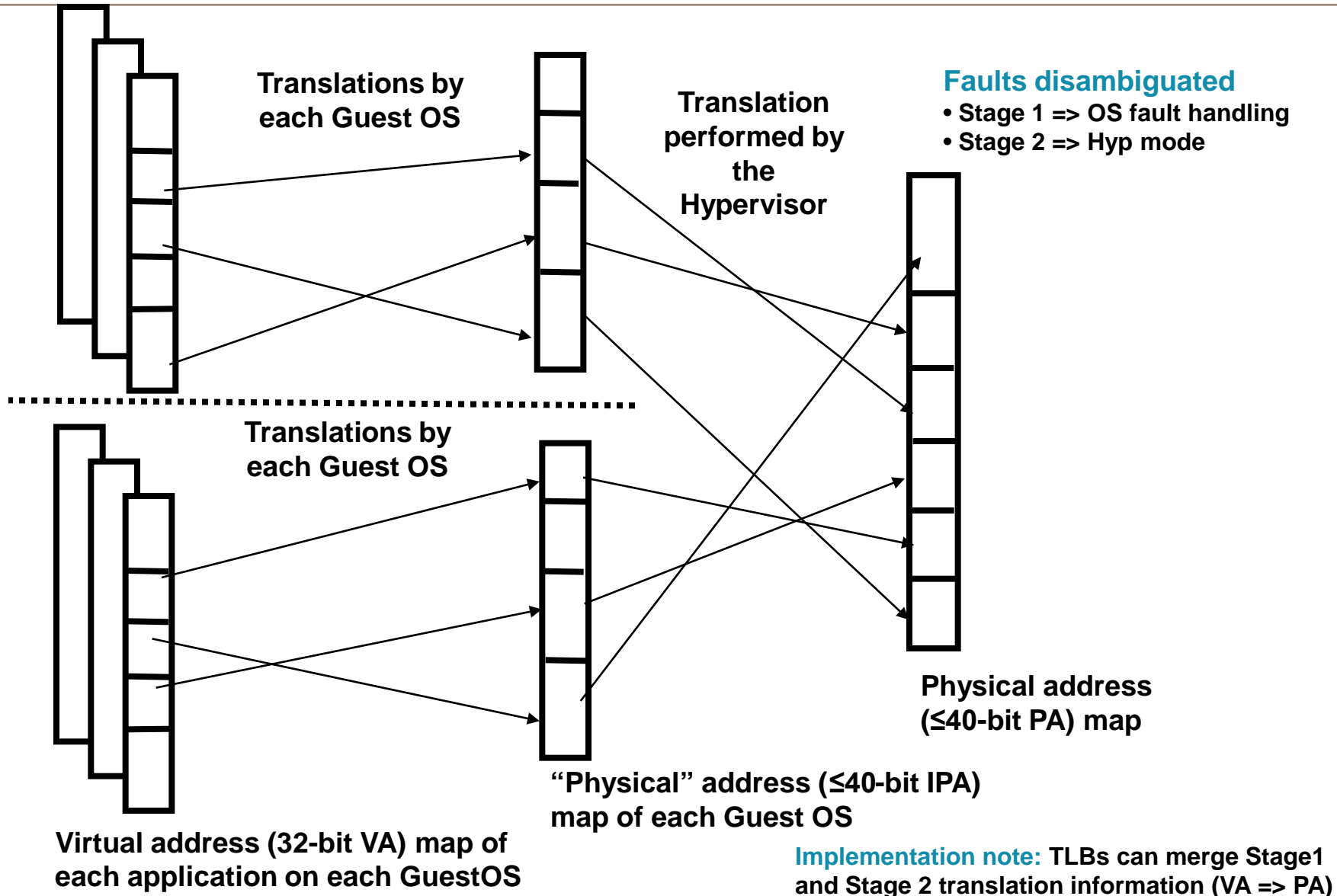


ARMv7-A Virtualization - key features 2

- 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

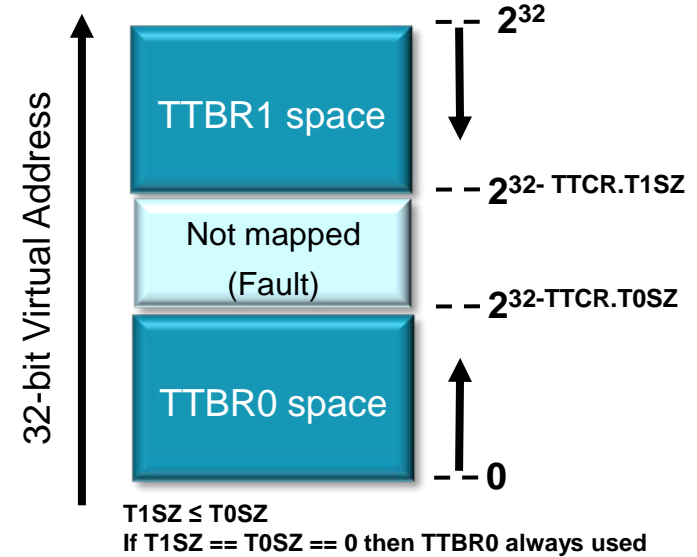


2-stage address translation



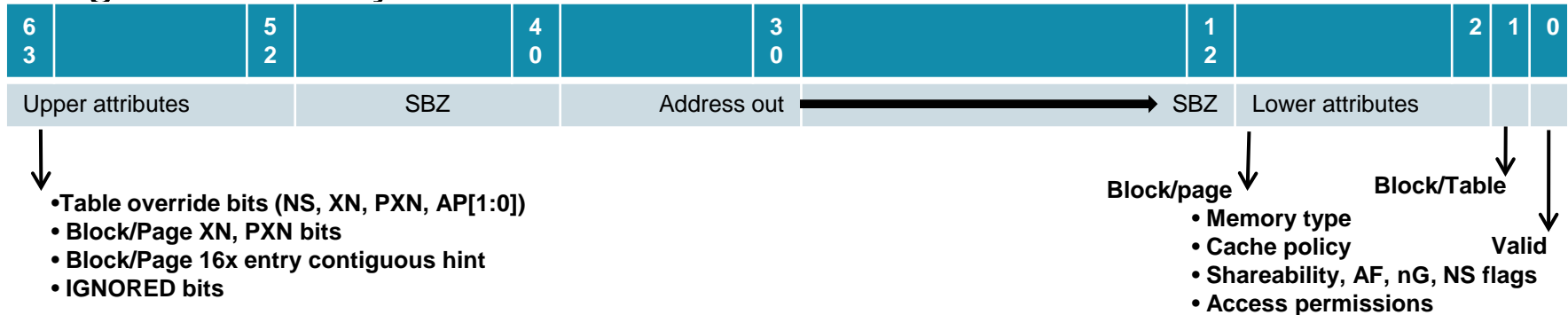
LPAE – Stage 1 (VA => {I}PA)

- 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



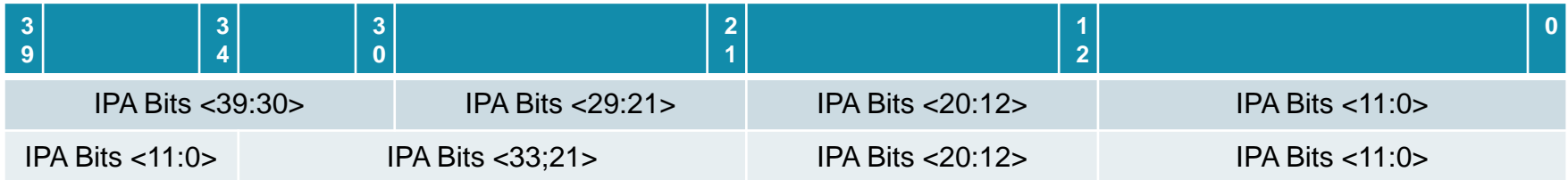
31:30	VA Bits <29:21>	VA Bits <20:12>	VA Bits <11:0>
L1	Level 2 table index	Level 3 table (page) index	Page offset address

Page Table Entry:

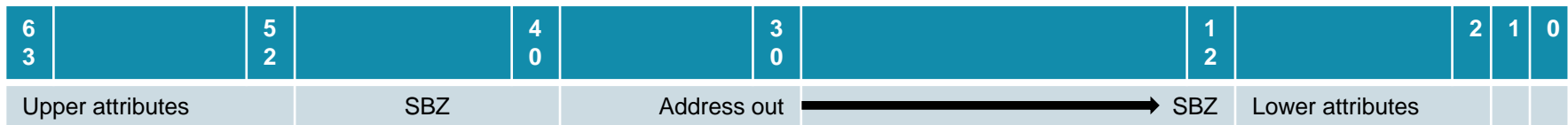


LPAAE – Stage 2 (IPA => PA)

- 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



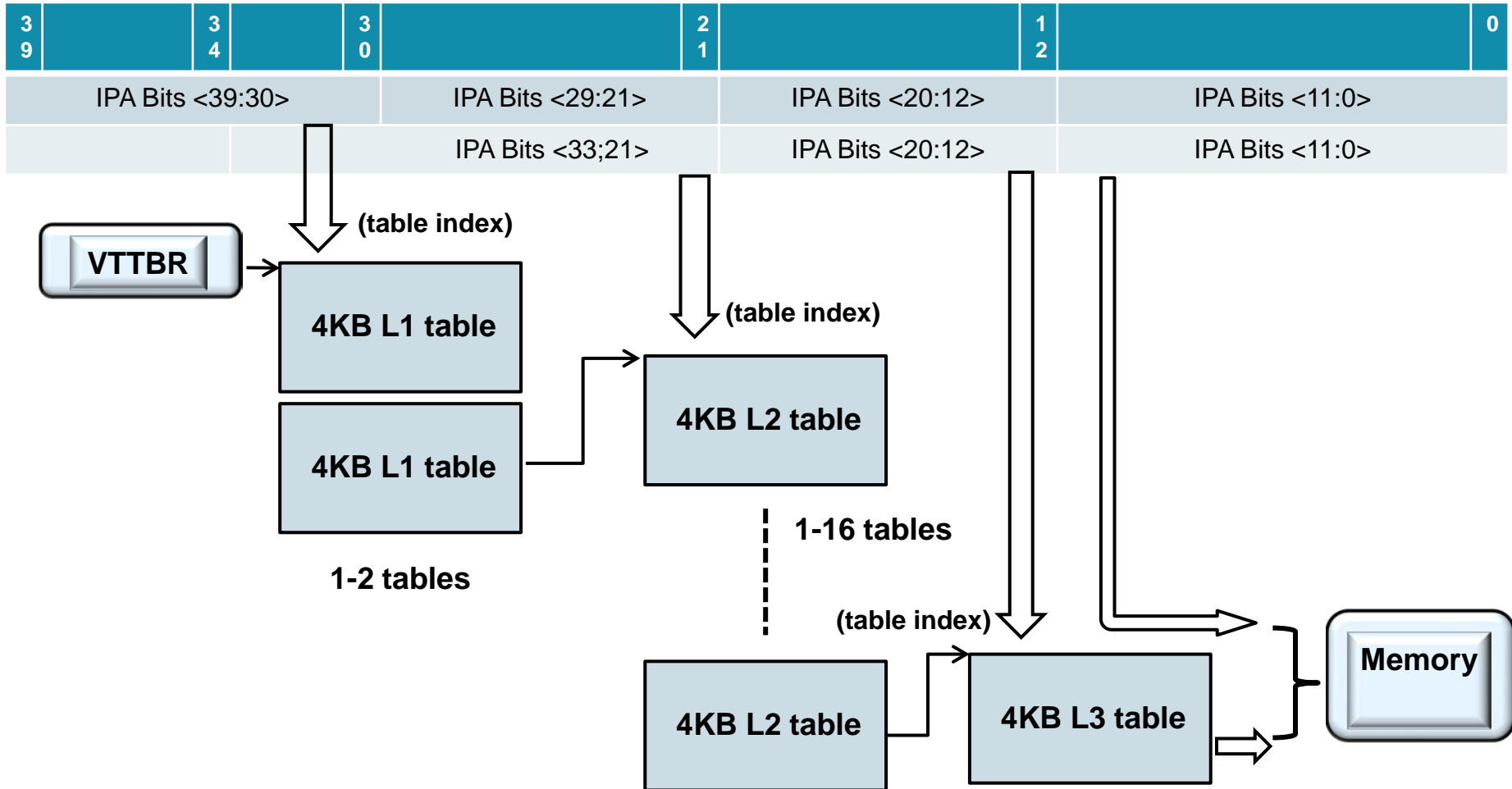
- 1x Translation Table Base register (VTTBR)
- Page table entry:



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

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

Example: Stage-2 table walk

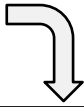


Interrupt Controller

- 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

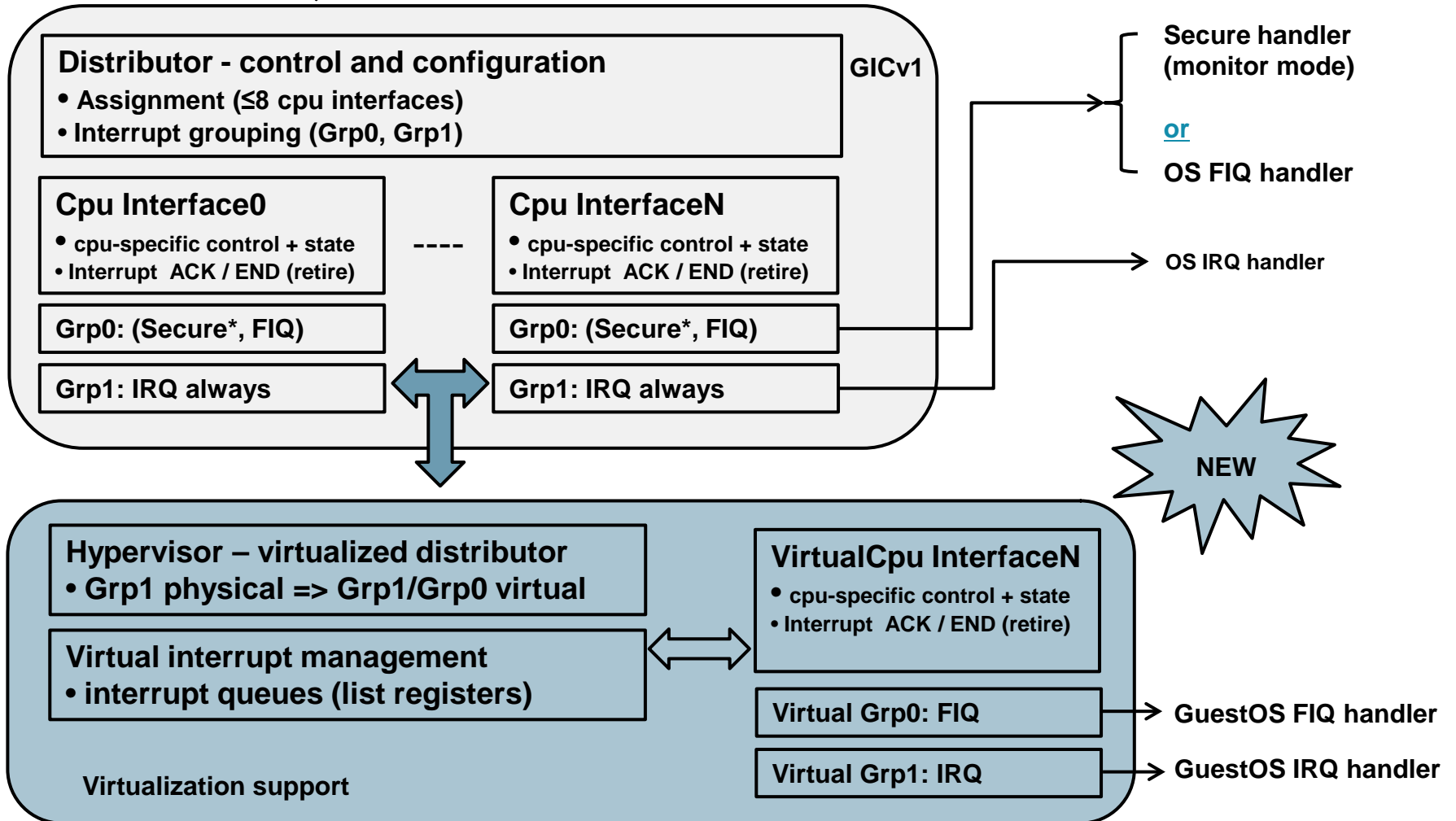
Interrupt Handling – GICv2

Interrupt sources:

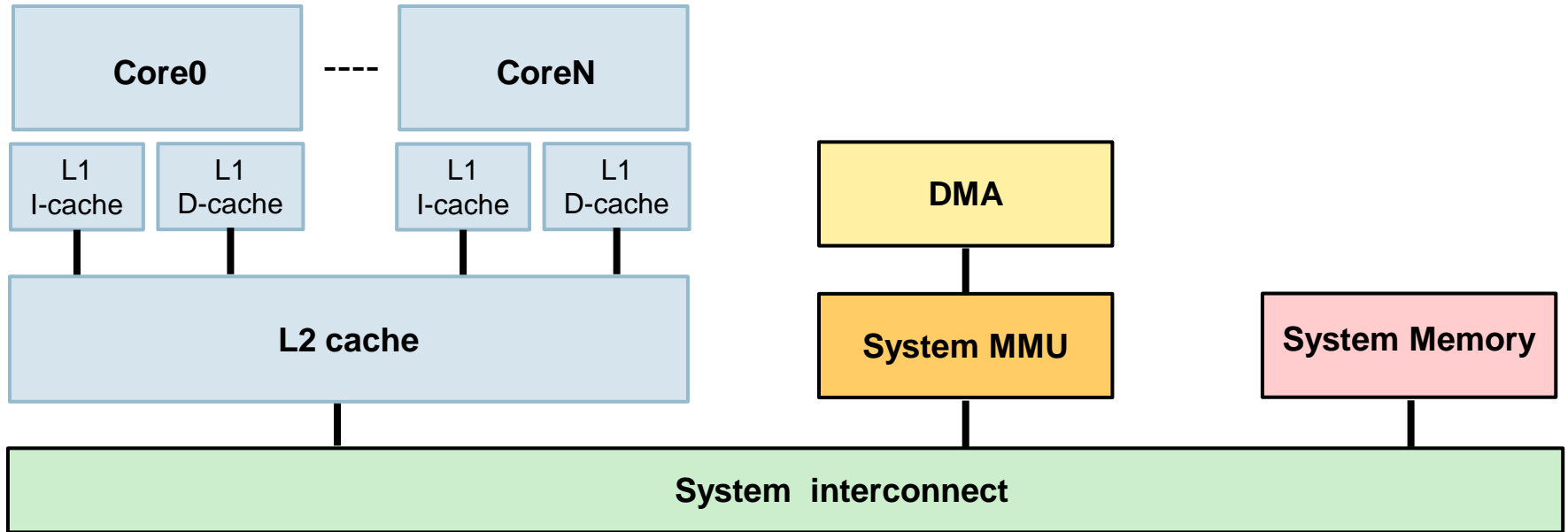


- software
- private (per cpu) peripheral
- shared peripheral

* ARM Architecture Security Extns



System MMU

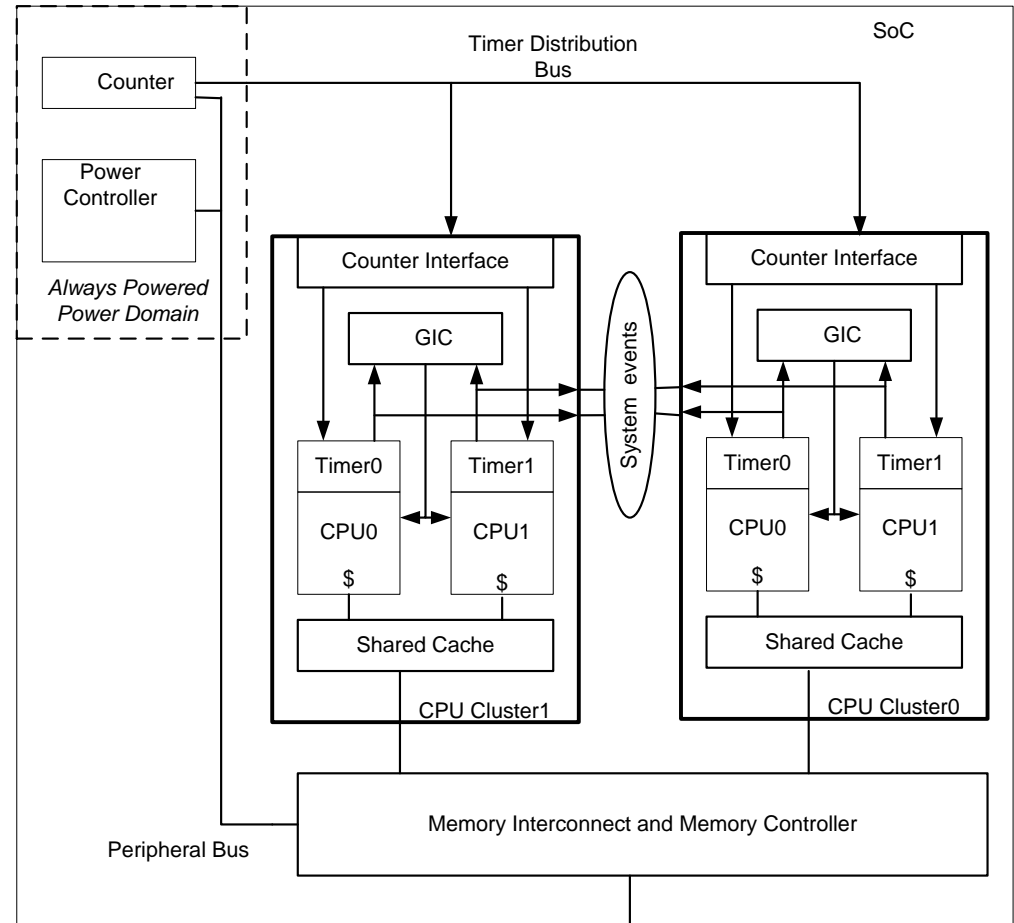


- System MMU architecture translation options from pre-set tables
 - No Translation
 - VA -> IPA (Stage 1) or IPA->PA (Stage 2) only
 - VA->PA (Stage1 and Stage2)
- Can share page tables with ARM cores – relate contexts to VMIDs/ASIDs
- Architecture specification will be published 1H-2011
 - **ARM planning to support AMBA[®] based solutions in 2011 timeframe**

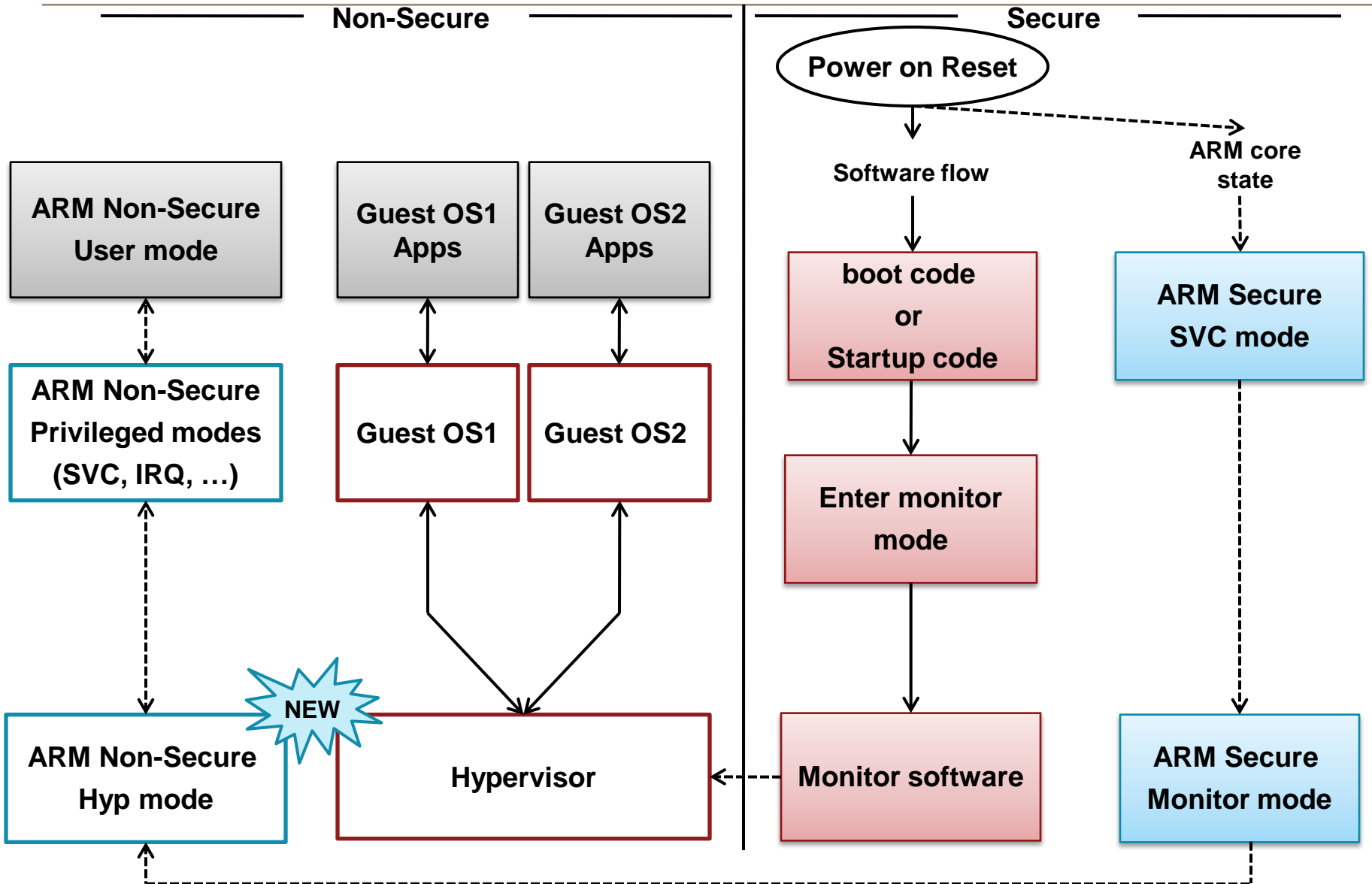
Generic timer

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

Implementation example:



Hypervisor - boot flow by example



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:



ENECA



**Mentor
Graphics**



Open Kernel Labs™

virtual**Logix**™

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

