The Evolution of the ARM Architecture Towards Big Data and the Data-Centre

8th Workshop on Virtualization in High-Performance Cloud Computing (VHPC'13) held in conjunction with SC 13, Denver, Colorado

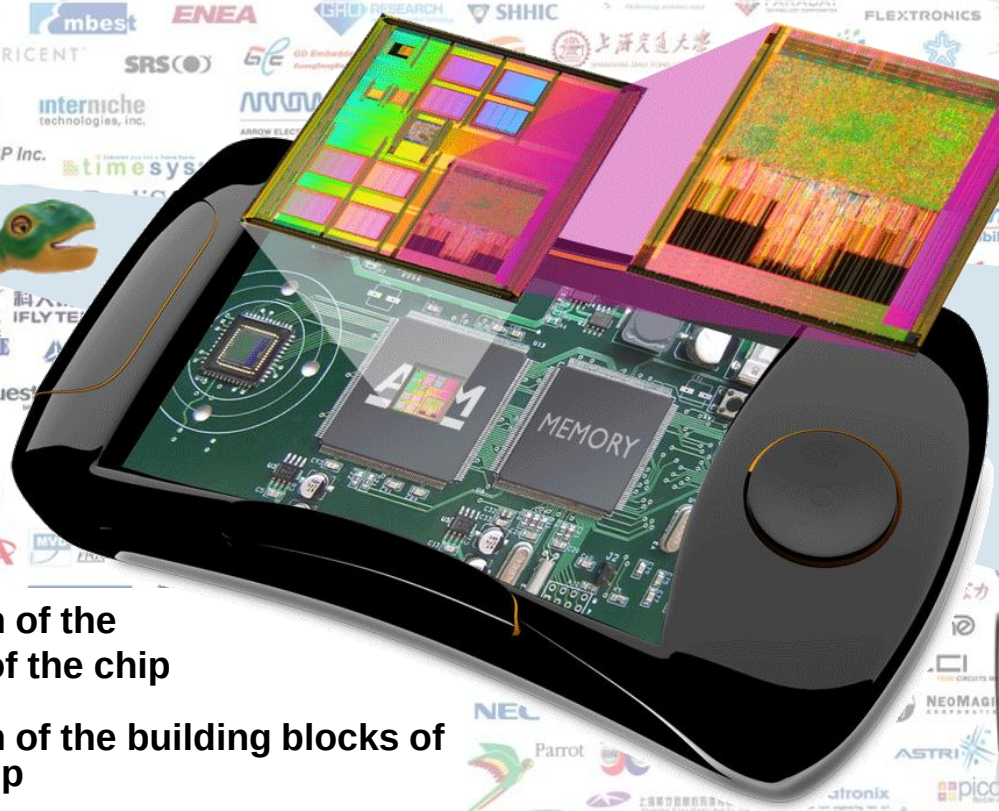
John Goodacre
Director, Technology and Systems
ARM Processor Division
ARM Cambridge

Abstract

- In recent years the rapid evolution of the ARM Architecture has lead to a potential inflection point where the architectural and compute capabilities the ARM partners deliver into the low cost, low power, consumer focused products has direct applicability to the requirement of compute in the data centre and the processing of big data. This talk will review these capabilities and the discuss the potential impact to the data centre. Finally, we'll take a glance into a project that aims to use a holistic approach to compute, ranging from 3D fabrication of an application specific, scalable compute element with virtualized access to system level resources so to balance to cost and requirement between compute, IO and memory.

ARM: Technology Supplier

- Advanced digital products are incorporating more and more ARM technology – from processor and multimedia IP to software

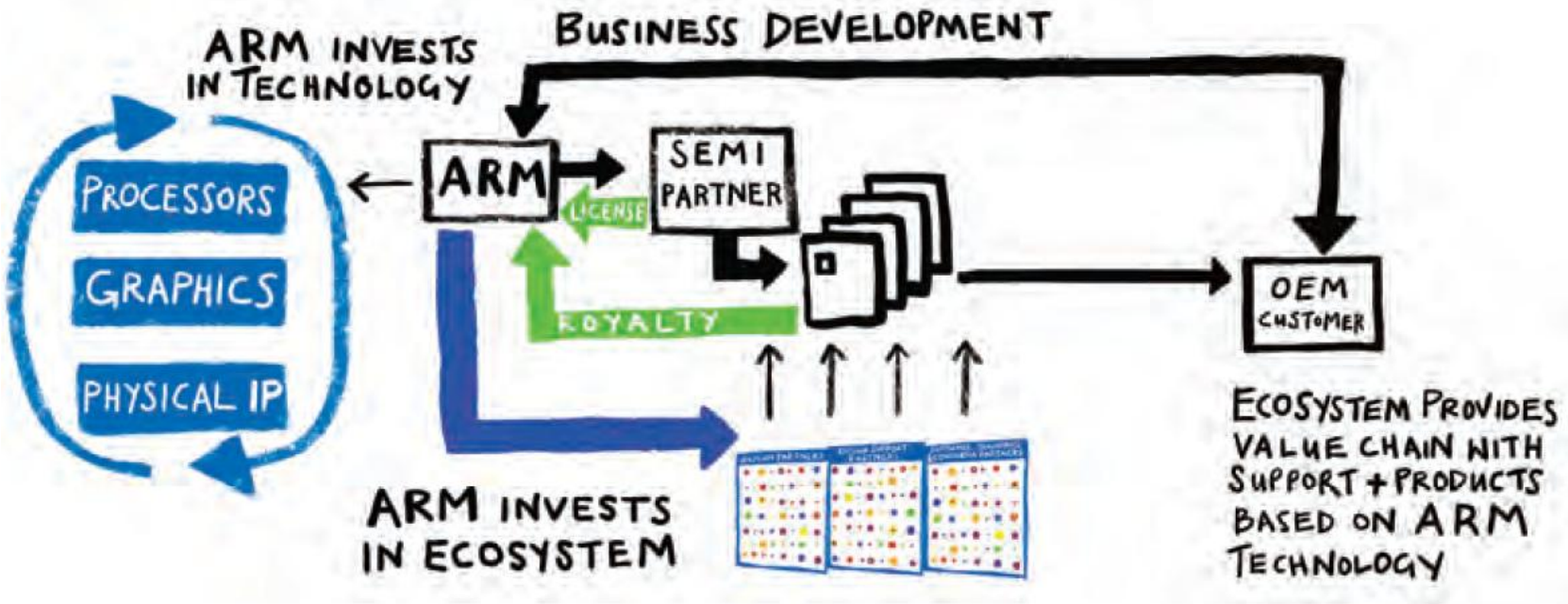


Processor IP – **Design of the brain of the chip**

Physical IP – **Design of the building blocks of the chip**

Software development tools

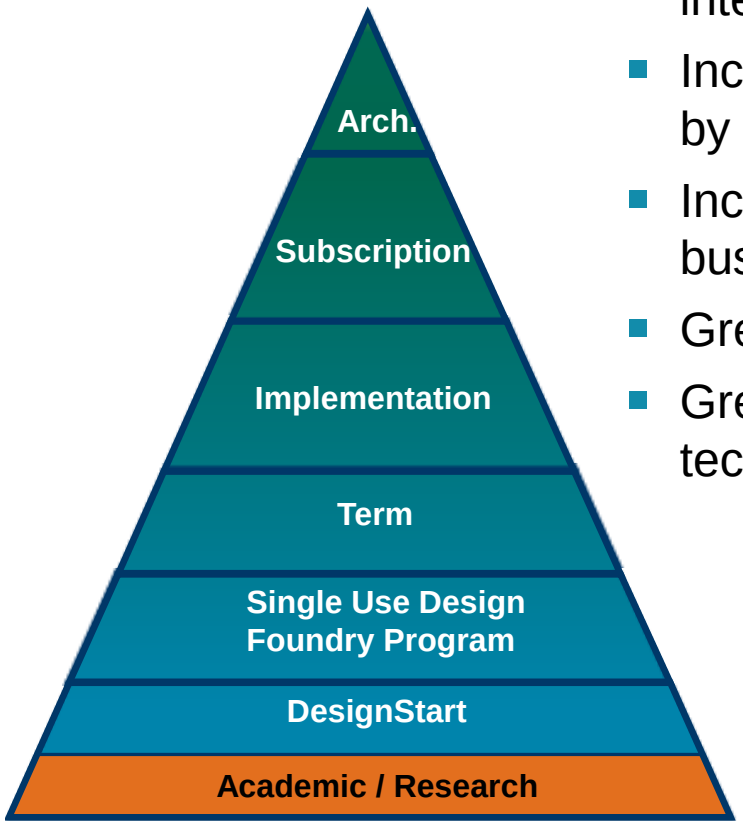
ARM IP Business Model



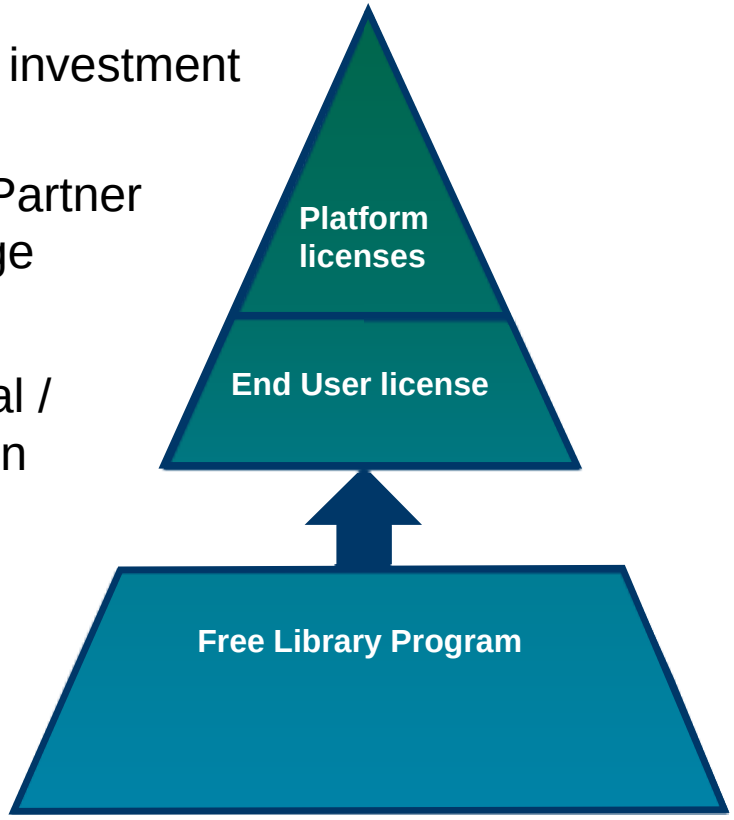
- Our customers are primarily the Semiconductor partners
- We sell mostly to their OEM
- We work with our customer to support the ecosystem

Various IP Licensing Models

- Higher license value
- Higher IP value
- Greater commercial / technical interaction
- Increasing internal investment by partner
- Increasing ARM / Partner business knowledge
- Greater flexibility
- Greater commercial / technical interaction

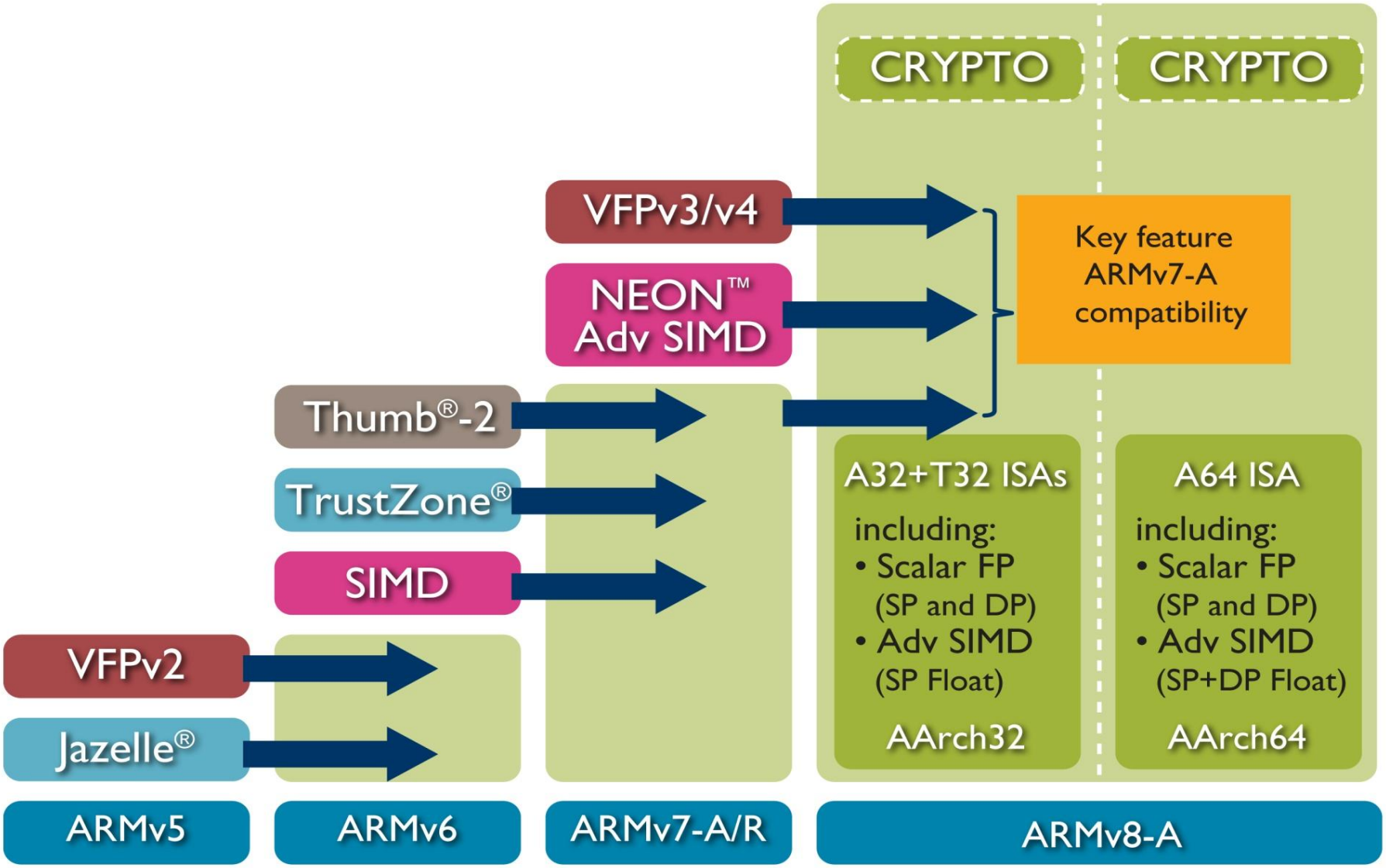


Processors

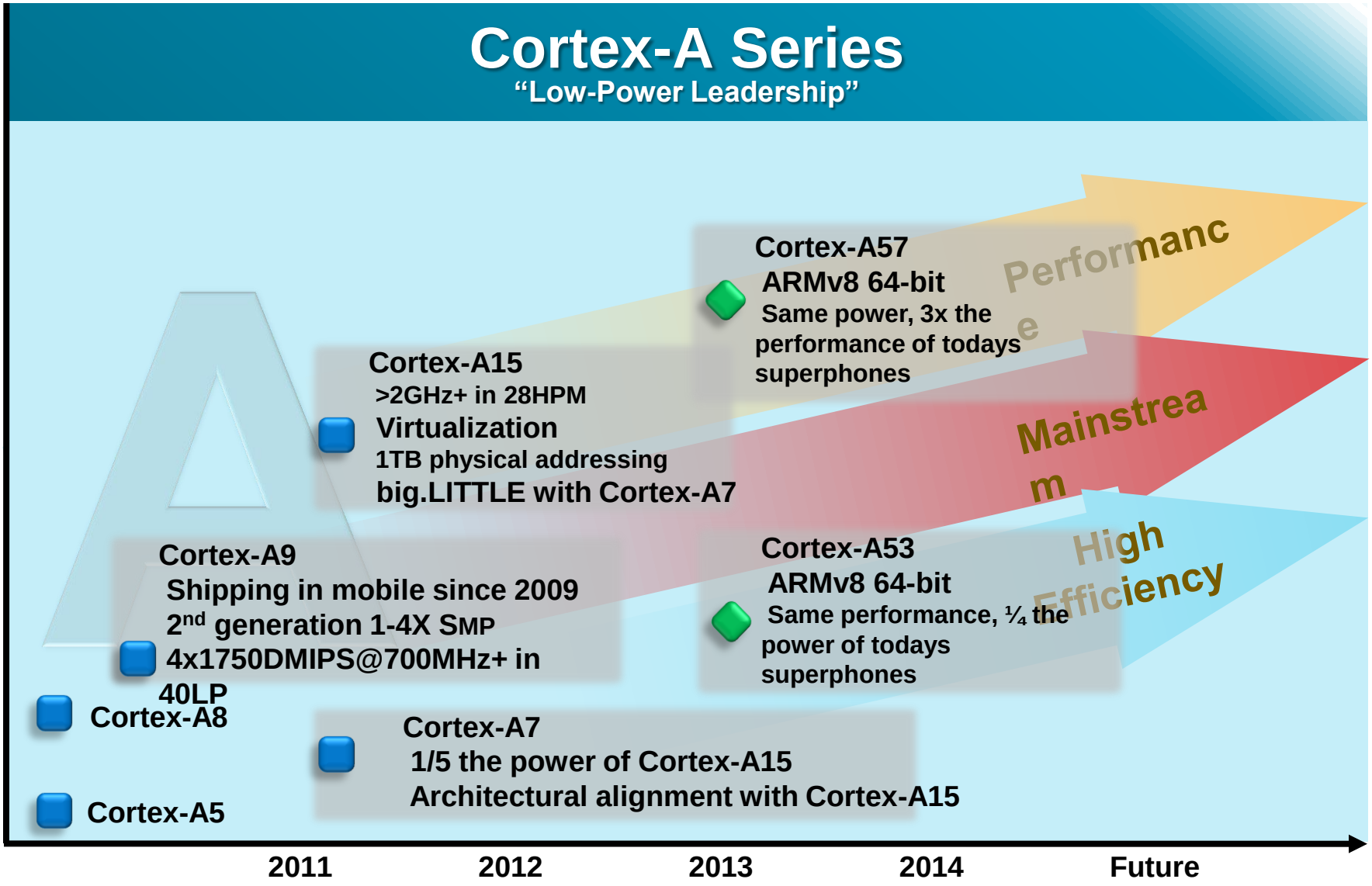


Physical IP

ARM Architecture Evolution



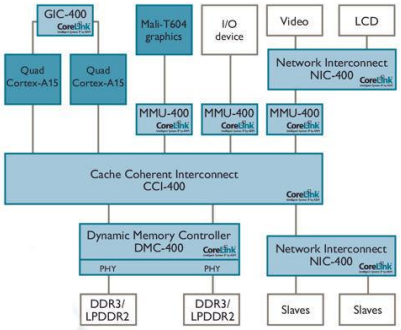
ARM Applications Processors



ARM VIRTUALIZATION EXTENSIONS

Capabilities introduced in Cortex-A15

- Full backwards compatibility with the Cortex-A9
  - Supporting the ARMv7 Architecture
- Addition of Virtualization Extension (VE)
  - Run multiple OS binary instances simultaneously
  - Isolates multiple work environments and data
- Supporting Large Physical Addressing Extensions (LPAE)
  - Ability to use up to 1TB of physical memory
- With AMBA 4 System Coherency (AMBA-ACE)
  - Other cached devices can be coherent with processor
  - Many core multiprocessor scalability
  - Basis of concurrent big.LITTLE Processing



Large Physical Addressing

- 40-bit physical addressing
  - Virtual memory (apps and OS) still has 32bit address space
  - However support more memory resident applications
- Offering up to 1 TB of physical address space
  - Supporting multiple 32bit OS images through virtualization
- What does this mean for ARM based systems?
  - More applications resident at the same time
  - Multiple resident virtualized operating systems
  - Common global physical address in many-core

Virtualization Extensions: The Basics

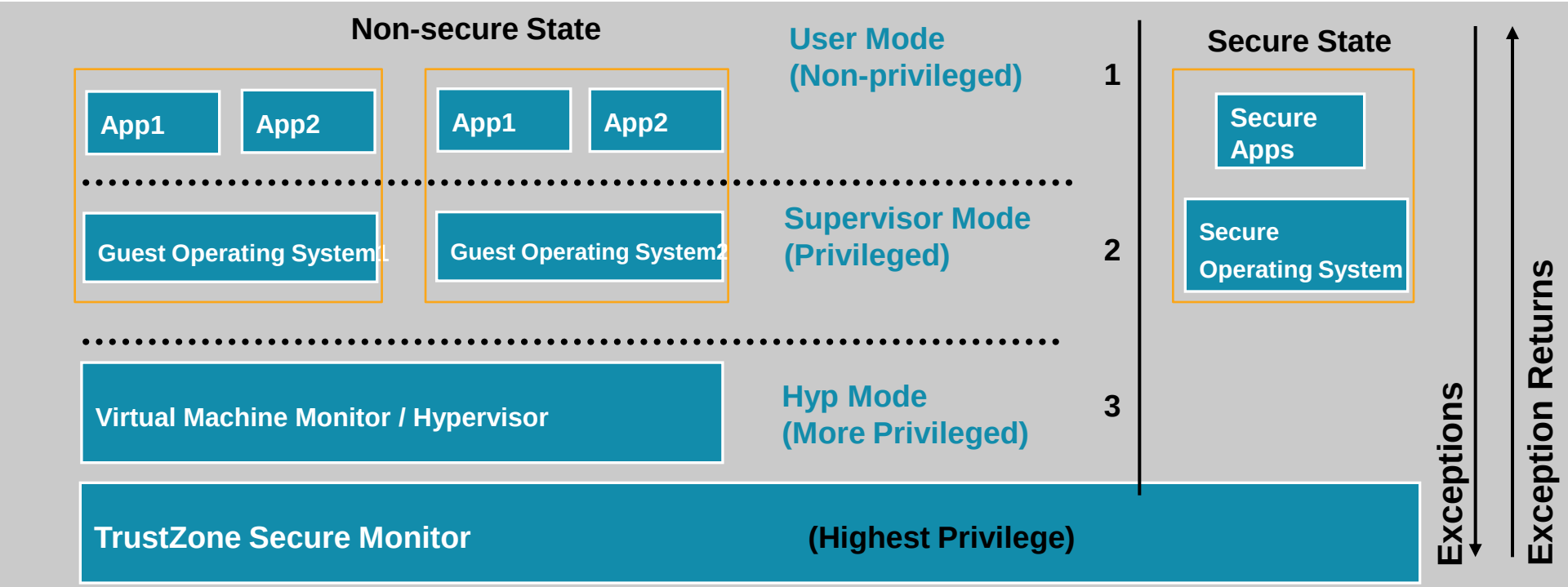
- New Non-secure level of privilege to hold Hypervisor
  - Hyp mode
- New mechanisms avoid the need Hypervisor intervention for:
  - Guest OS Interrupt masking bits
  - Guest OS page table management
  - Guest OS Device Drivers due to Hypervisor memory relocation
  - Guest OS communication with the interrupt controller (GIC)
- New traps into Hyp mode for:
  - ID register accesses and idling (WFI/WFE)
  - Miscellaneous “difficult” System Control Register cases
- New mechanisms to improve:
  - Guest OS Load/Store emulation by the Hypervisor
  - Emulation of trapped instructions through syndromes

How does ARM do Virtualization

- Extensions to the v7-A Architecture, available on the Cortex™-A15 , Cortex-A7 CPUs and future processors
  - Second stage of address translation (separate page tables) Functionality for virtualizing interrupts inside the Interrupt Controller
  - Functionality for virtualizing all CPU features, including CP15
  - Option of a MMU within the system to help virtualize IO
- Hypervisor runs in new “Hyp” exception mode / privilege
  - HVC (Hypervisor Call) instruction to enter Hyp mode
  - Uses previously unused entry (0X14 offset) in vector table for hypervisor traps
  - Hyp mode exception link register, SPSR, stack pointer
  - Hypervisor Control Register (HCR) marks virtualized resources
  - Hypervisor Syndrome Register (HSR) for Hyp mode entry reason

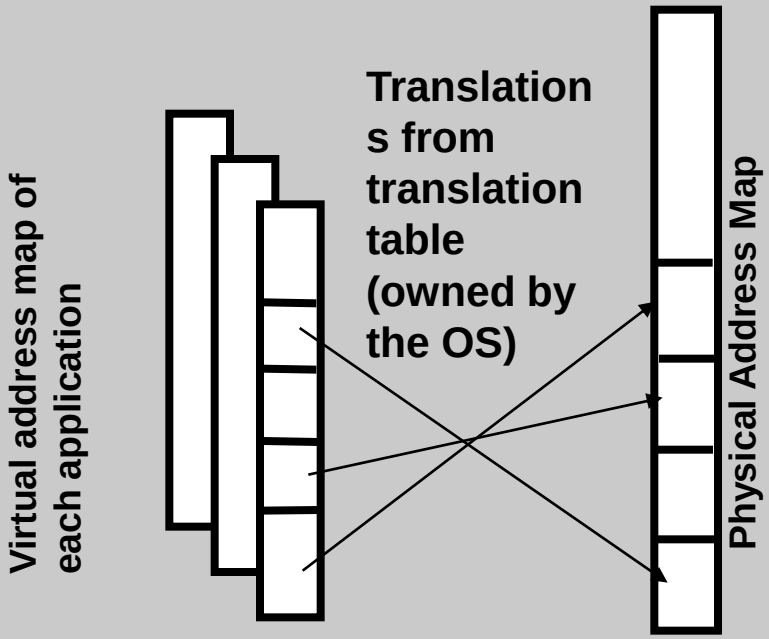
Virtualization: Third Privilege

- Guest OS same kernel/user privilege structure
- HYP mode higher privilege than OS kernel level
- VMM controls wide range of OS accesses
- Hardware maintains TZ security (4th privilege)

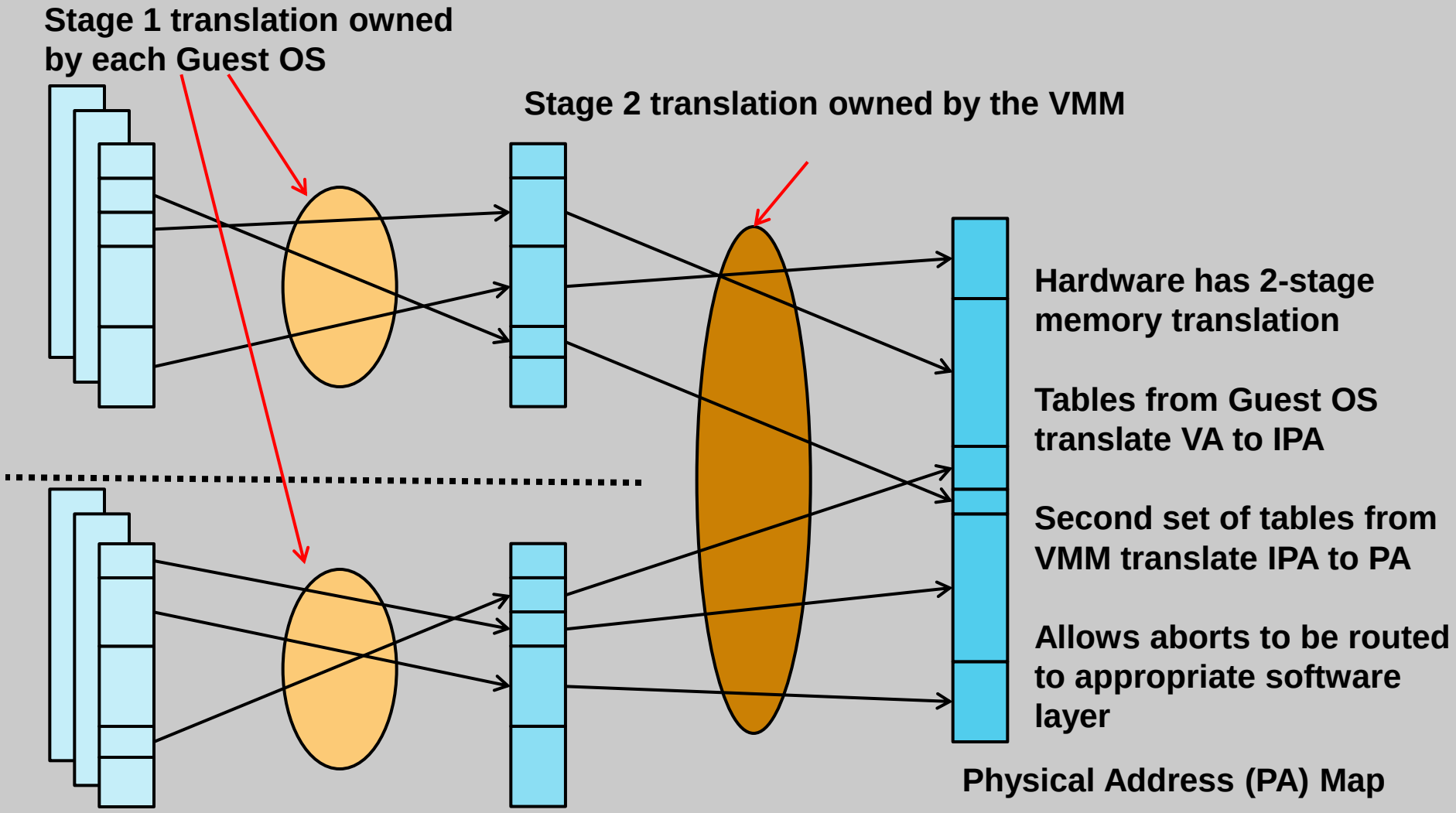


Memory – the Classic Resource

- Before virtualisation – the OS owns the memory
  - Allocates areas of memory to the different applications
  - Virtual Memory commonly used in “rich” operating systems

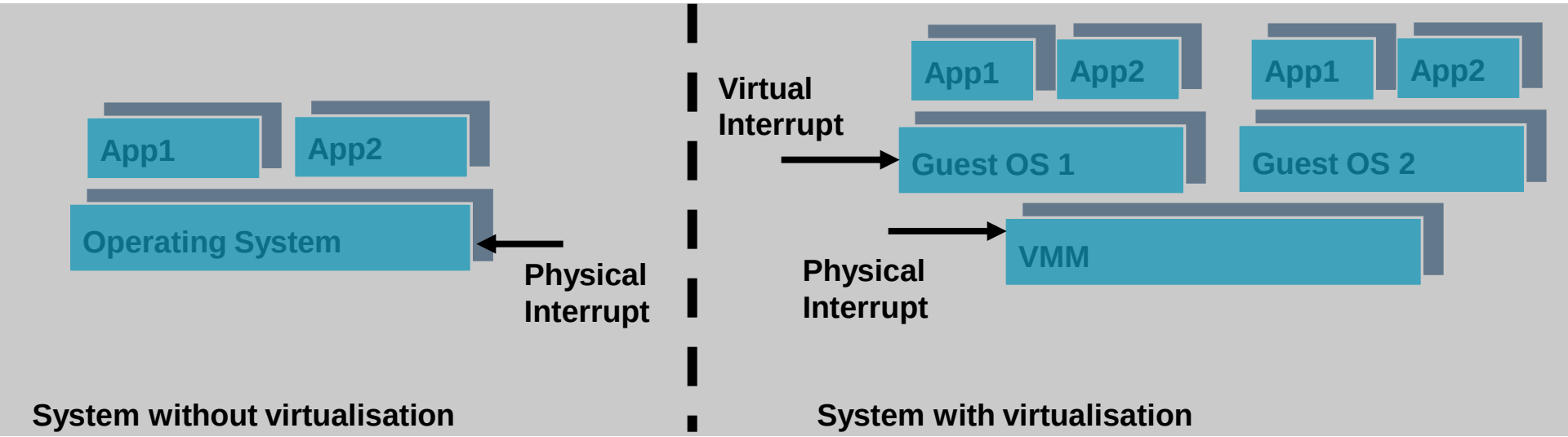


Virtual Memory in Two Stages



Classic Issue: Interrupts

- An Interrupt might need to be routed to one of:
  - Current or different GuestOS
  - Hypervisor
  - OS/RTOS running in the secure TrustZone environment
- Basic model of the ARM virtualisation extensions:
  - Physical interrupts are taken initially in the Hypervisor
  - If the Interrupt should go to a GuestOS :
    - Hypervisor maps a “virtual” interrupt for that GuestOS



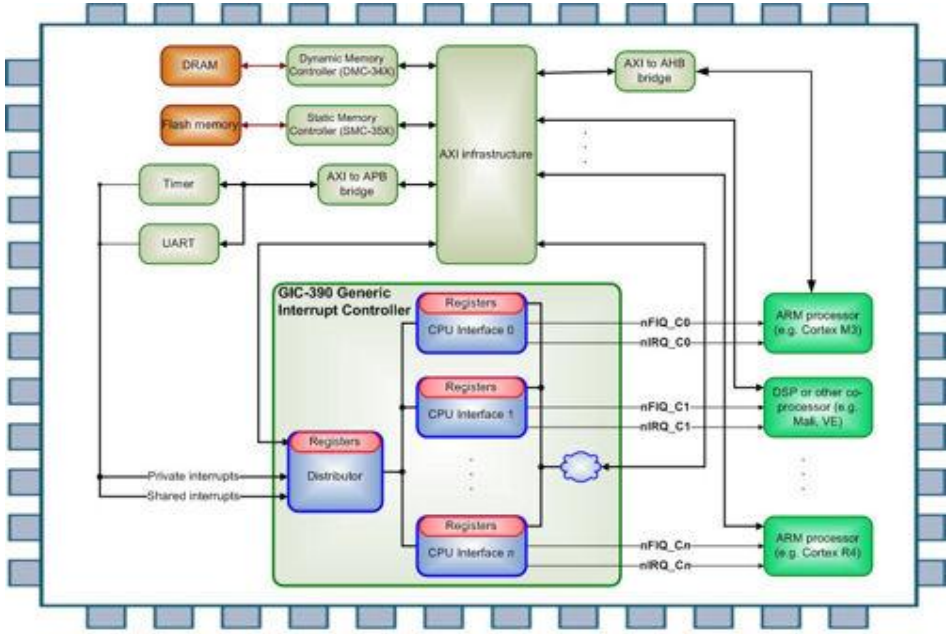
Interrupt Virtualization

- Virtualisation Extensions provides :
  - Registers to hold the Virtual Interrupt
  - CPSR.{I,A,F} bits in the GuestOS only appling to that OS
    - Physical Interrupts are not masked by the CPSR.{I.A.F} bits
    - GuestOS changes to I,A,F no longer need to be trapped
  - Mechanism to route all physical interrupts to Monitor Mode
    - Already utilized in TrustZone technology based devices
  - Virtual Interrupts are routed to the Non-secure IRQ/FIQ/Abort
  - Guest OS manipulates a virtualized interrupt controller
- Actually available in the Cortex-A9 to aid paravirtualization support for interrupts

Virtual Interrupt Controller

- New “Virtual” GIC Interface has been Architected
  - ISR of GuestOS interacts with the virtual controller
  - Pending and Active interrupt lists for each GuestOS
  - Interacts with the physical GIC in hardware
  - Creates Virtual Interrupts only when priority indicates it is necessary

- GuestOS ISRs therefore do not need calls for:
  - Determining interrupt to take [Read of the Interrupt Acknowledge]
  - Marking the end of an interrupt [Sending EOI]
  - Changing CPU Interrupt Priority Mask [Current Priority]



GIC-390 Generic Interrupt Controller in Multiprocessor System

Virtual GIC (Global Interrupt Controller)

- GIC now has separate sets of internal registers:
  - Physical registers and virtual registers
  - Non-virtualized system and hypervisor access the physical registers
  - Virtual machines access the virtual registers
  - Guest OS functionality does not change when accessing the vGIC
- Virtual registers are remapped by hypervisor so that the Guest OS thinks it is accessing the physical registers
  - GIC registers and functionality are identical
- Hypervisor can set IRQs as virtual in the HCR
  - Interrupts are configured to generate a Hypervisor trap
  - Hypervisor can deliver an interrupt to a CPU running a virtual process using “register lists” of interrupts

Resource Ownership

- Software-only approaches
  - Access to resources by GuestOS intercepted by the VMM
    - VMM interprets the GuestOS's intent
    - Provides its own mechanism to meet that intent
  - Mechanism of interception varies
    - Paravirtualisation adds a hypercall to the source code
    - Binary translation adds a hypercall to the binary
    - Exceptions in Hardware provide an trapping of operations
  - Hypercalls can be more efficient
    - More of the intent to be expressed in a single VMM entry

- Hardware assisted approaches:
  - Provide further indirection to resources
  - Accelerating trapped operations by syndrome information

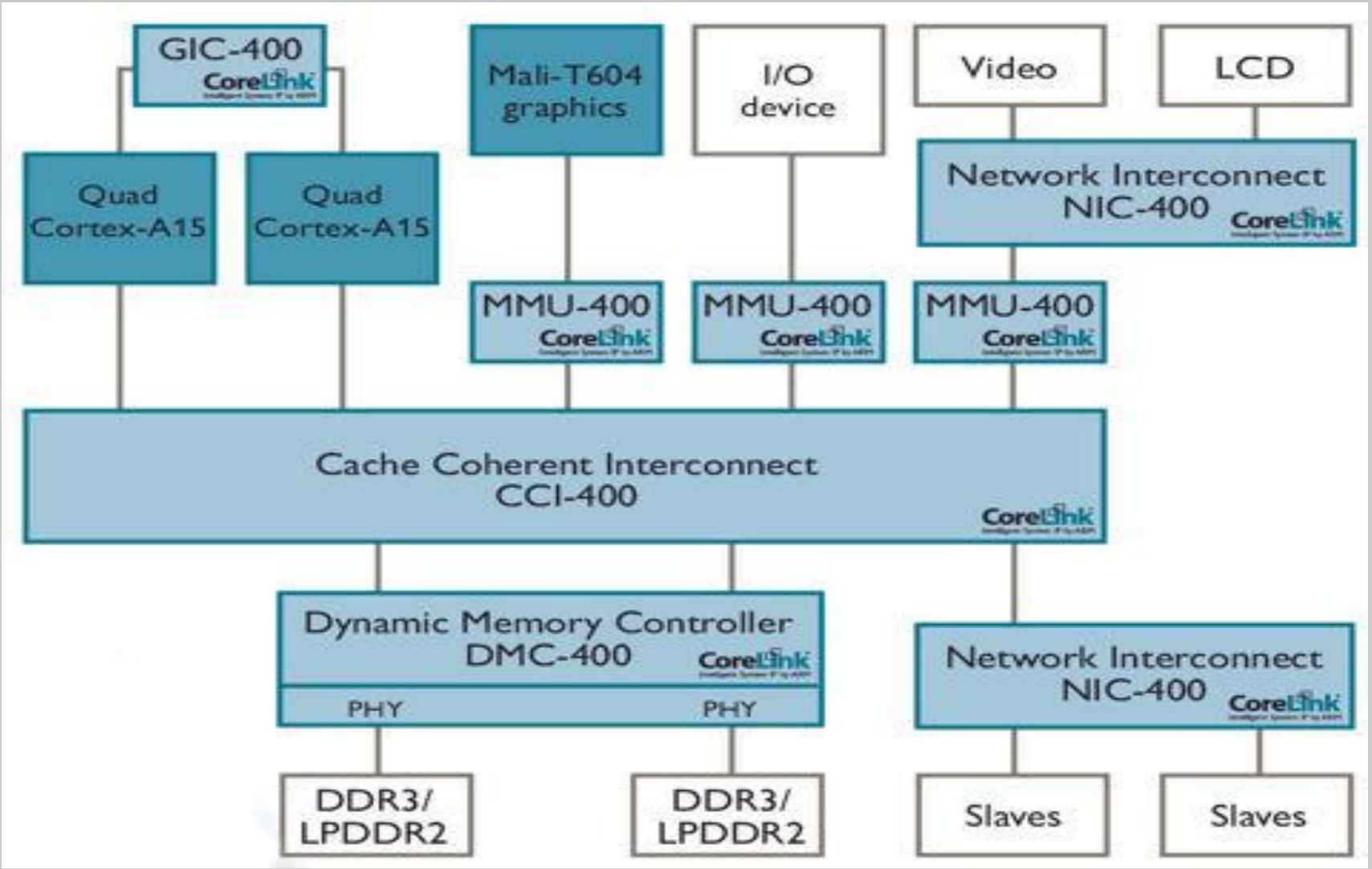
Helping with Virtual Devices

- ARM I/O handling uses memory mapped devices
  - Reads and Writes to the Device registers have specific side-effects
- Creating “Virtual Devices” requires emulation:
  - Typically reads/writes to devices have to trap to the VMM
  - VMM interprets the operation and performs emulation
  - Perfect virtualization means all possible devices loads/stores emulated
  - Fetching and interpreting emulated load/store is performance intensive
  - “Syndrome” information on aborts available for some loads/stores
  - Syndrome unpacks key information about the instruction
    - Source/Destination register, Size of data transfer, Size of the instruction, SignExtension etc
    - If syndrome not available, then fetching of the instruction for emulation still required

Devices and Memory

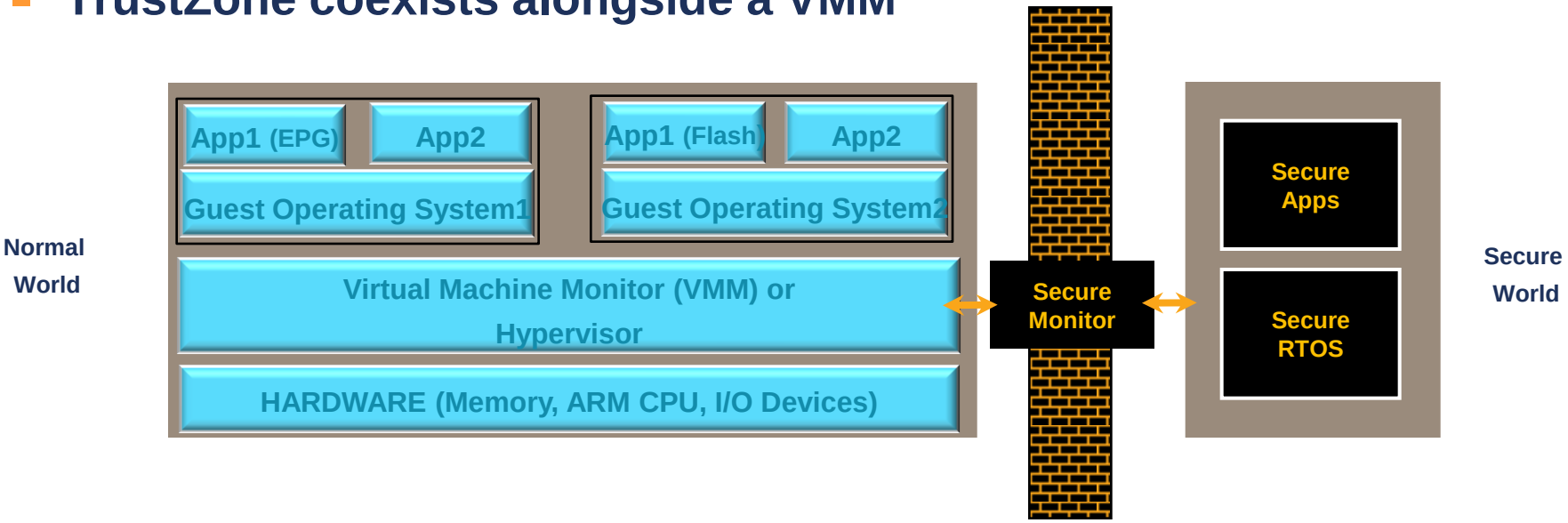
- Providing address translation for devices is important
  - Allows unmodified device drivers in the GuestOS
  - If the device can access memory, GuestOS will program it in IPA
- ARM virtualisation adds option for a “System MMU”
  - Enables second stage memory translations in the system
- A System MMU could also provide stage 1 translations
  - Allows devices to be programmed into guest’s VA space
- System MMU natural fit for the processor ACP port
  - ARM defining a common programming model
  - Intent is for the system MMU to be hardware managed using Distributed Virtual Messages found in AMBA 4 ACE

System MMU



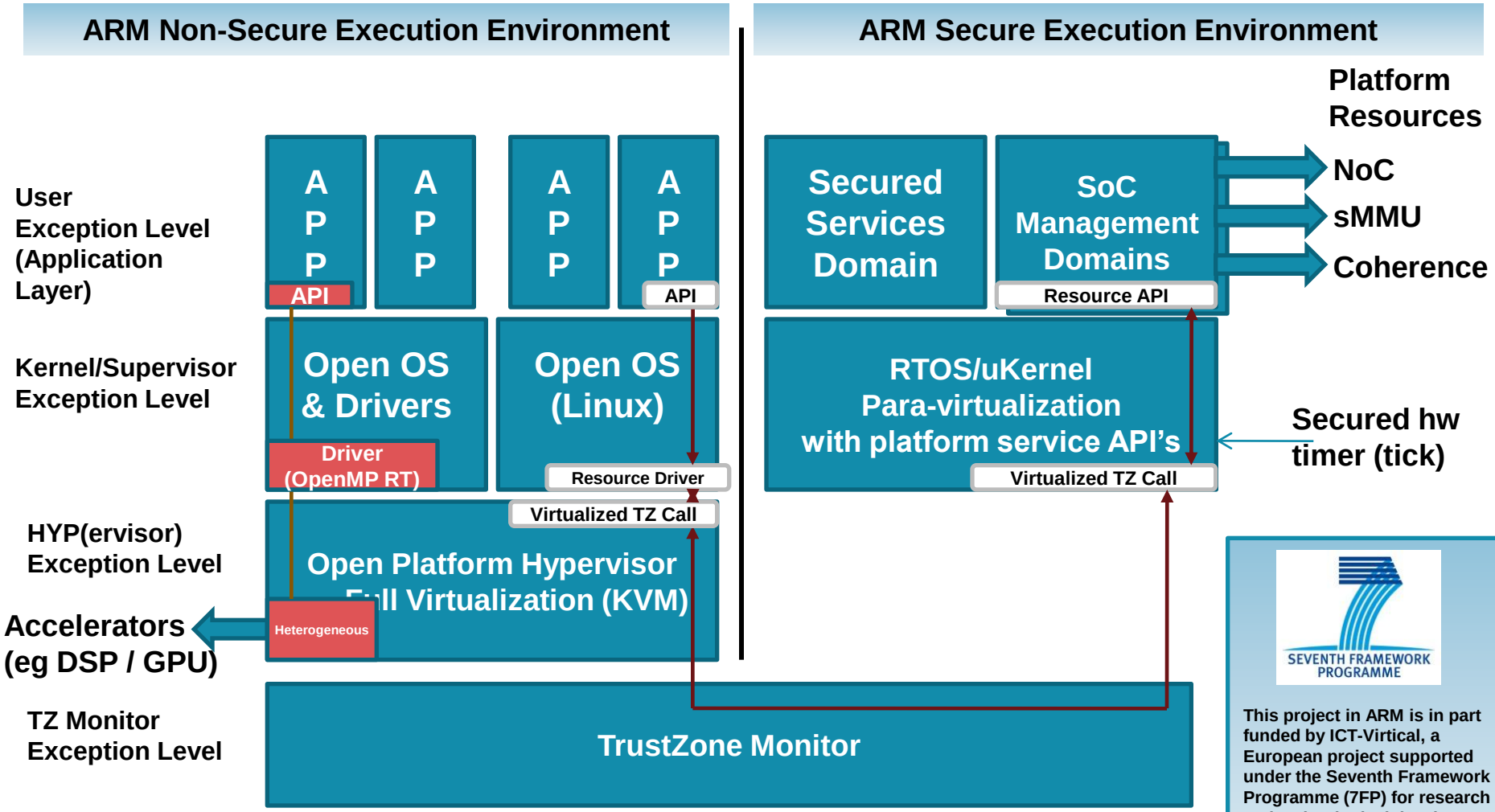
Virtualization and TrustZone

- **TrustZone coexists alongside a VMM**



- TrustZone offers a specialised type of Virtualisation
  - Only 2 ‘Worlds’ – not extendable (except through paravirtualization)
    - Although VMM can also span both worlds
  - Fourth privilege level is provided by CPU’s secure monitor mode
  - Non-symmetrical - The two ‘Worlds’ are not equal
    - Secure world can access both worlds (33bit addressing)

Spanning Hypervisor Framework



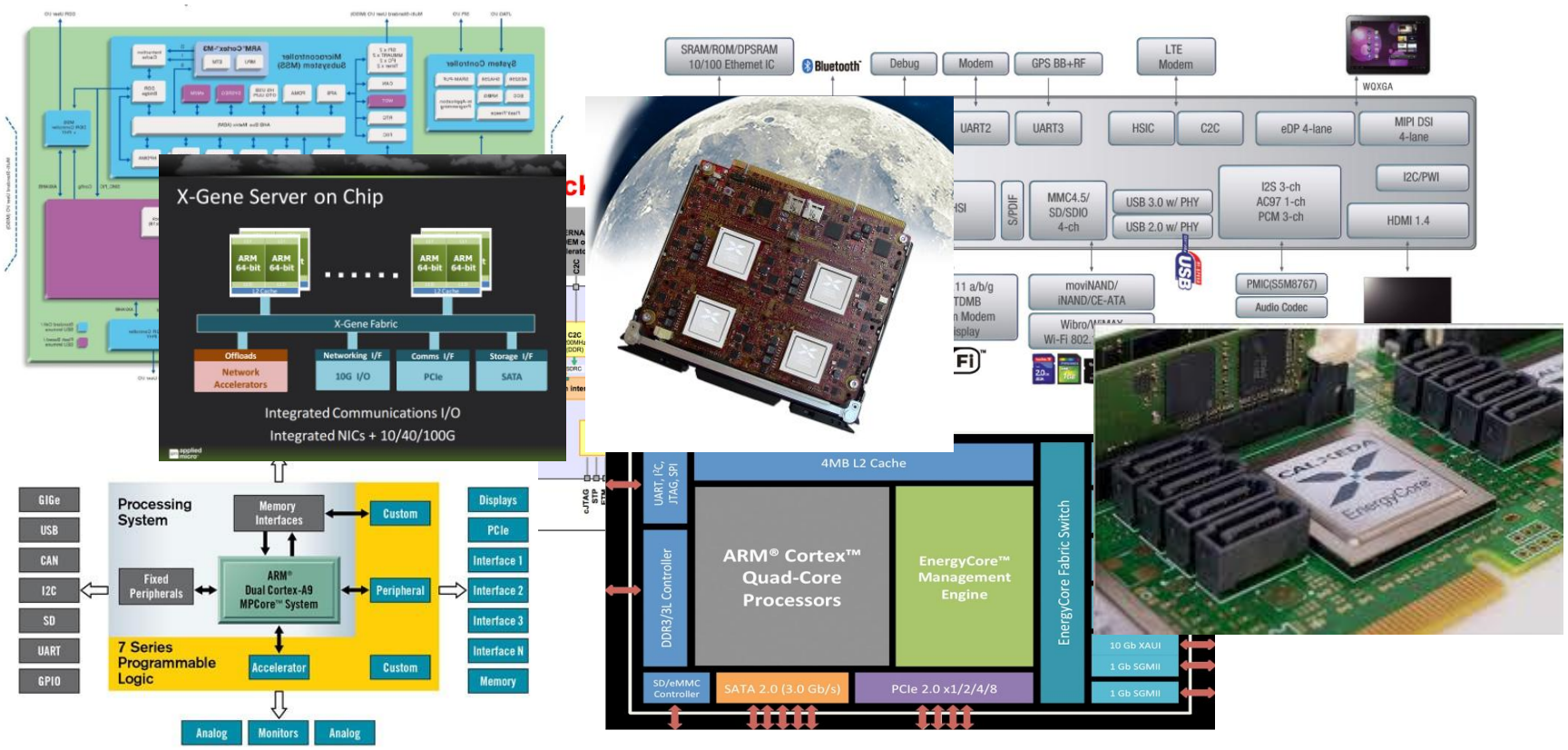
Enabling secure and mix-criticality systems through virtualization

SEVENTH FRAMEWORK PROGRAMME

This project in ARM is in part funded by ICT-Virtical, a European project supported under the Seventh Framework Programme (7FP) for research and technological development

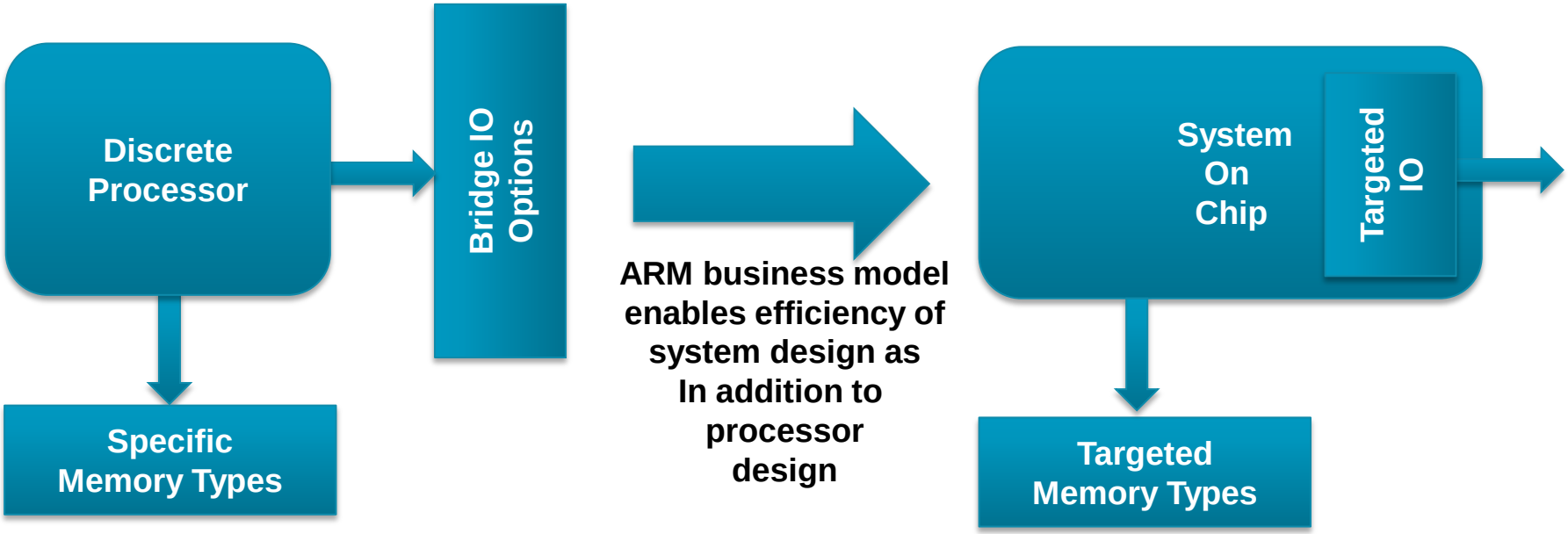
Project website
http://www.virtical.eu

Flexibility Through Design



- Same “core” technology
  - thousands of application specific solutions
  - Built by hundreds of ARM technology licensee partners

Advantage of a Targeted Solution

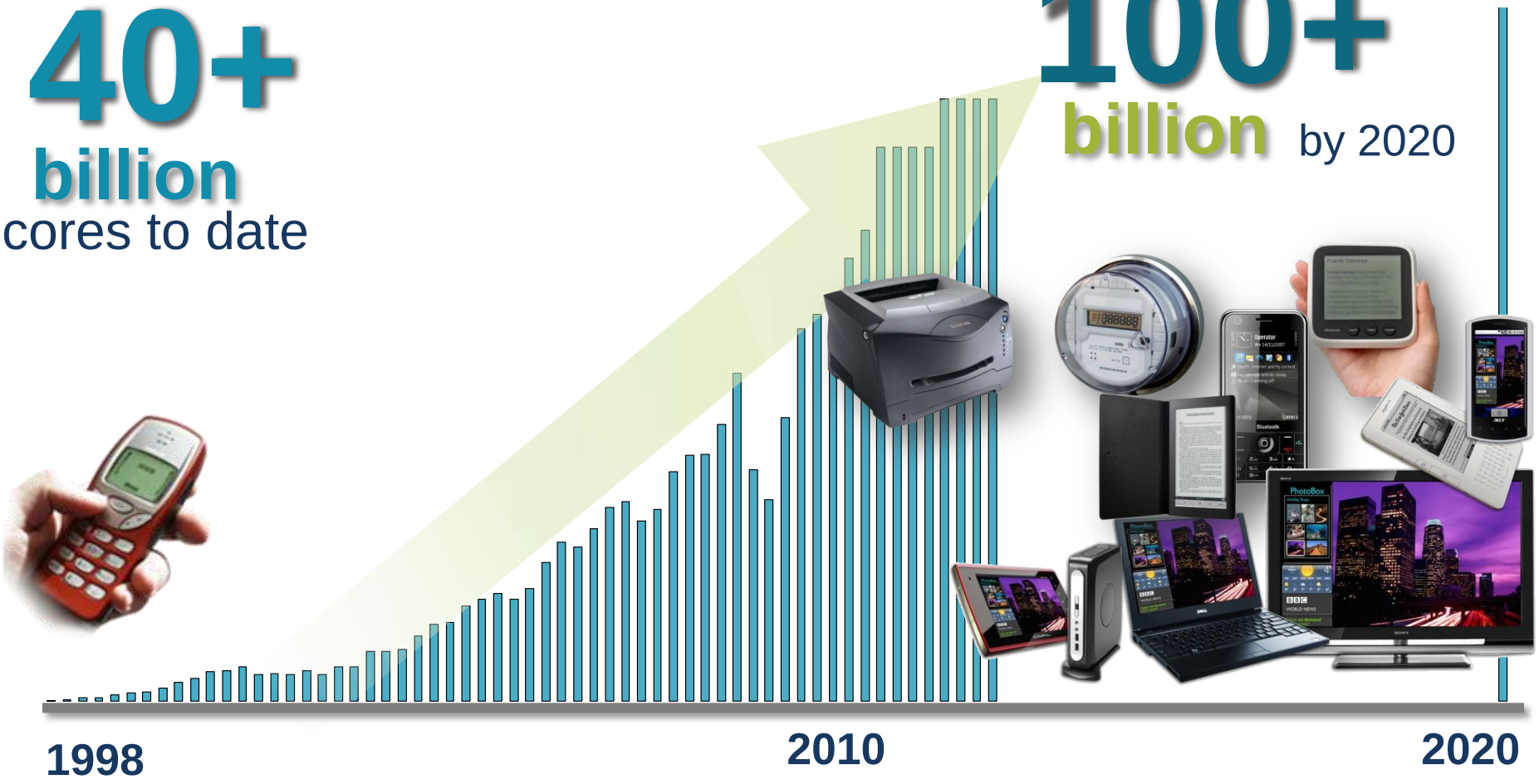


- One-size can not fit all applications
  - Compromise across all important metric, compute vs memory vs IO
  - Energy overhead from abstractions (eg PCIe) between IO and CPU
- ARM model supports hundreds of companies building hundreds of application optimized System on Chip solutions
  - The challenge is ROI – cost of development vs. market opportunity

COTS-on-Silicon

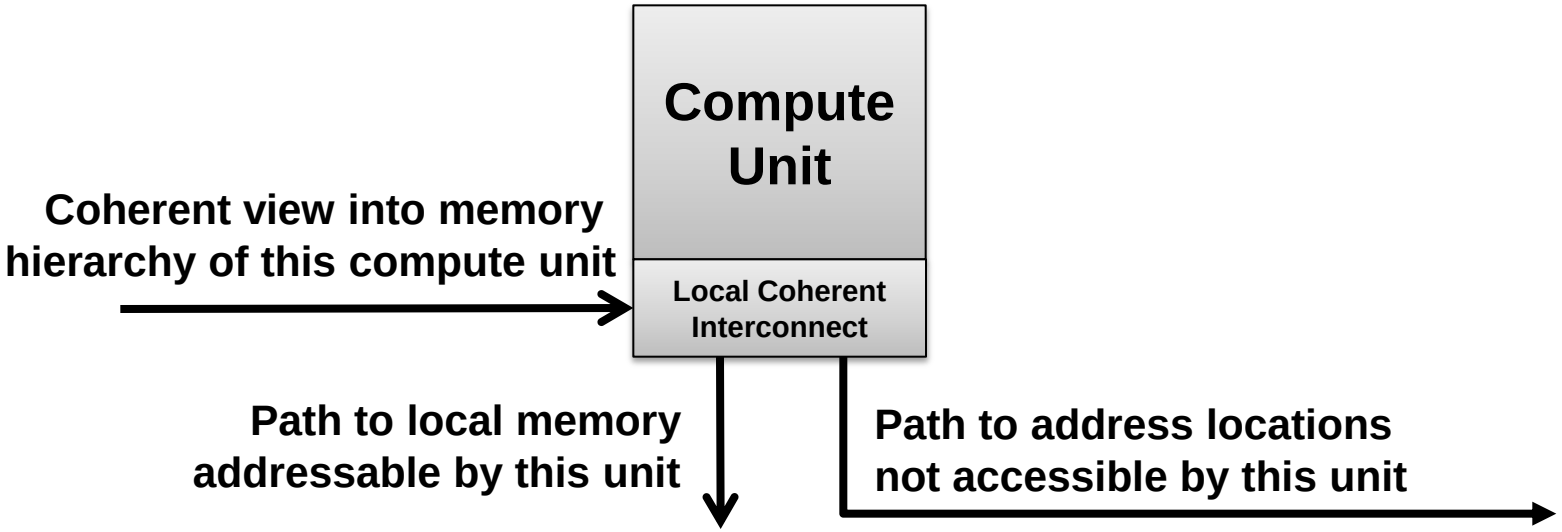
40+ billion cores to date

100+ billion by 2020



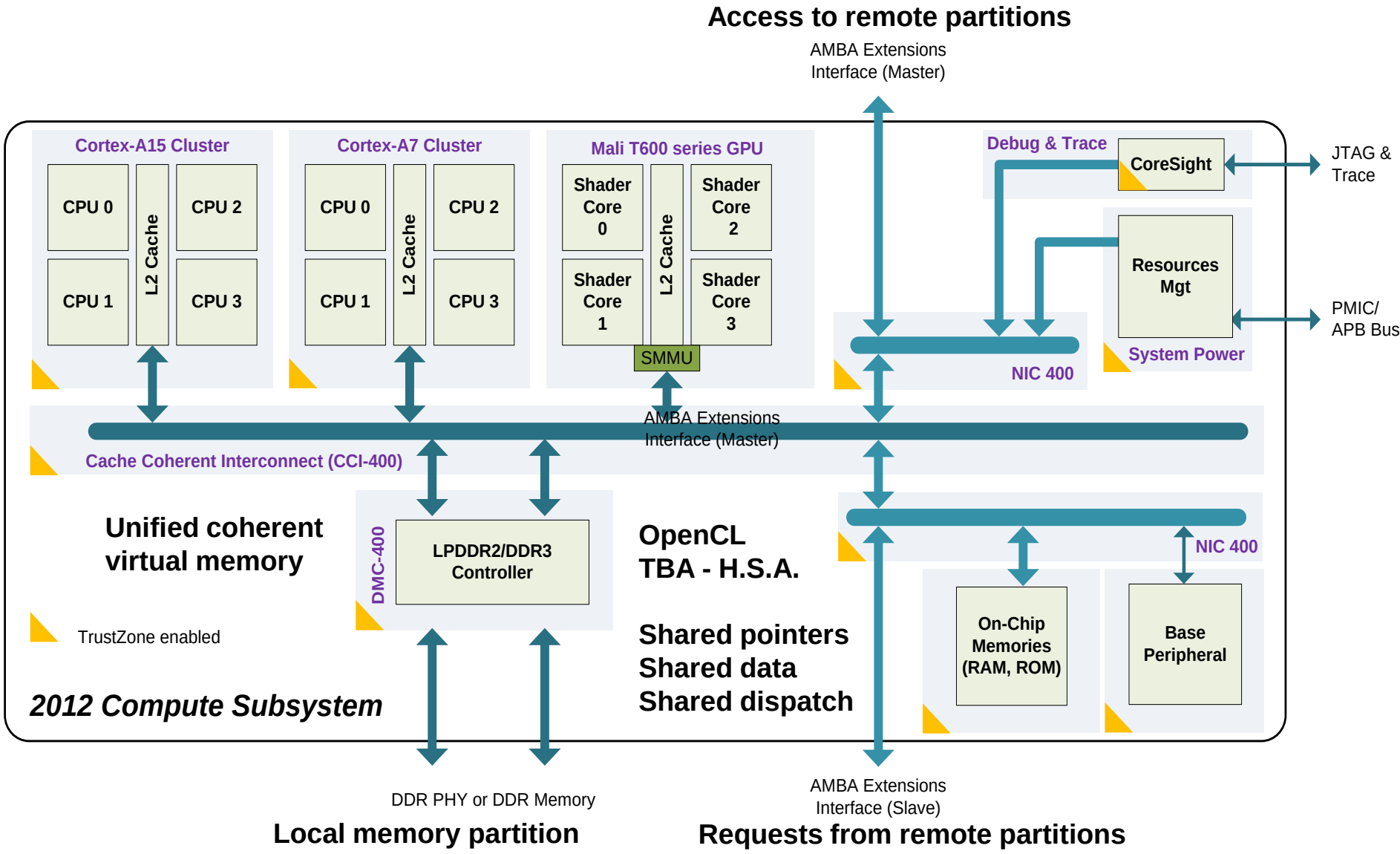
Approx 10B additional core in 2013

Scalability Concept: Compute Unit



- The Compute Unit unifies a number of localized compute resources
  - Potentially both general purpose and other local heterogeneous accelerators
- Provides coherent and symmetric access across local resources
  - Enabling a SMP capable operating system and resource sharing
- Each Compute Unit is registered at a partition within a system's global address space (GAS), including units with heterogeneous capability
  - Any unit can access any remote location in the GAS (including cached)
  - DMA can transfer between (virtually address cached) memory partitions

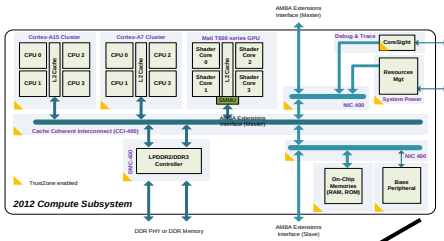
Example: Heterogeneous Compute Unit



EUROSERVER Vision: Started Sept'13

EURO SERVER

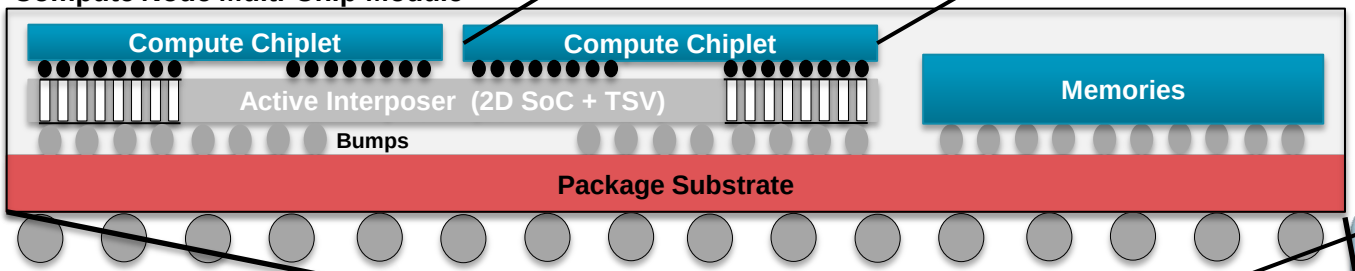
SEVENTH FRAMEWORK PROGRAMME



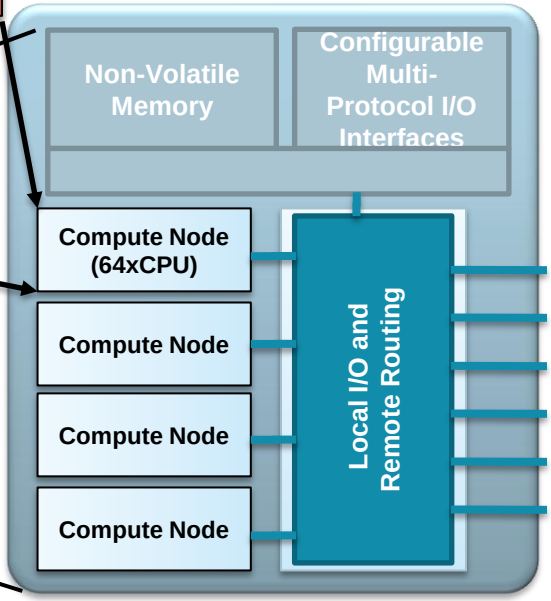
1) Start with a standard scalable compute unit "die"

2) Connect a few together along with its local memory partition

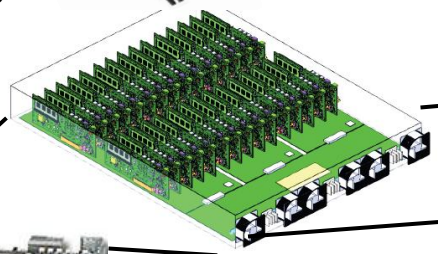
Compute Node Multi-Chip-Module

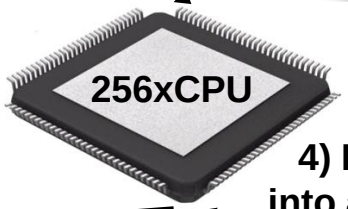


3) Put a few on a board
With shared virtualized IO's



Consumer supply





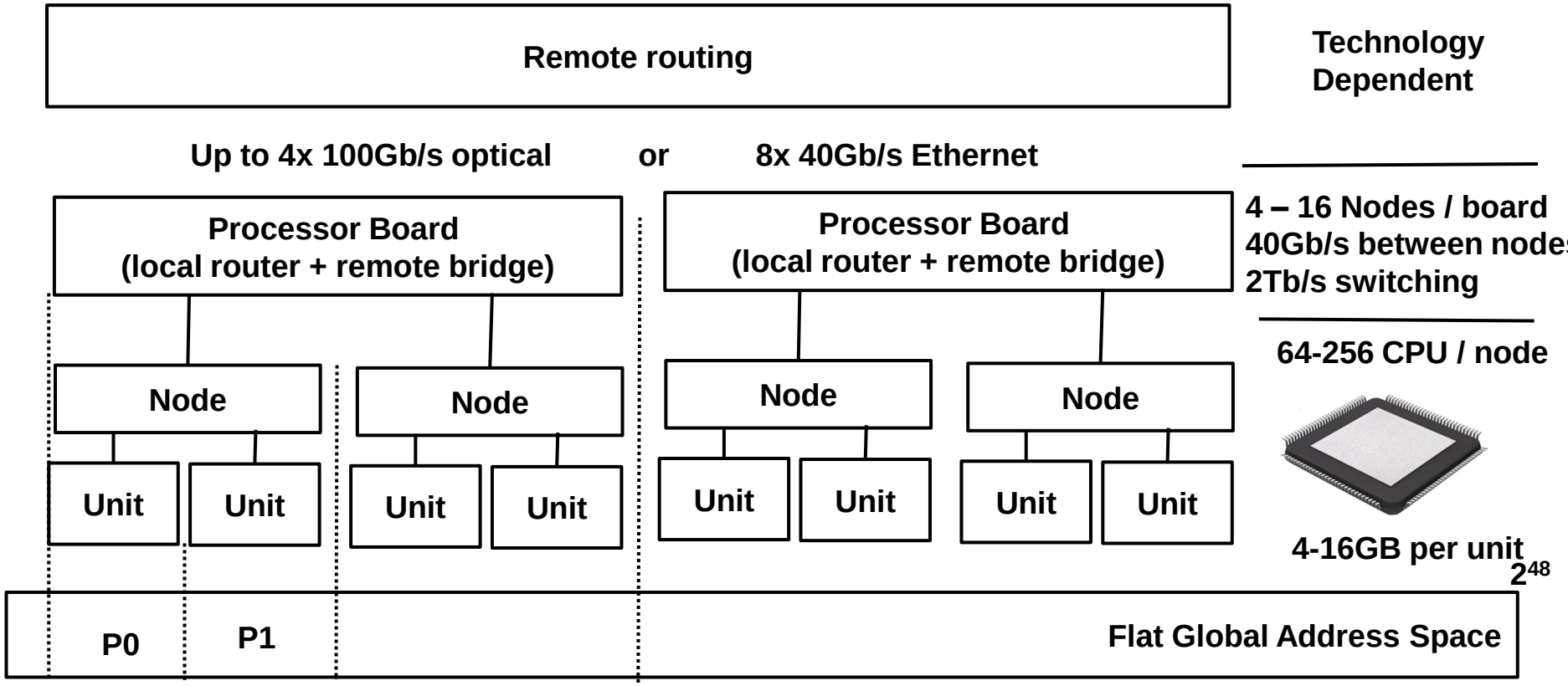
4) In future, integrate into a single MCM

Delivering a result
100x "better" than today

EUROSERVER: System Hierarchy

Component	Hierarchy	Per hierarchy	Software view	Design Choice
Rack	Shelves	8-16	System shmem MPI	Power / cooling density
Base board	Chassis shelf	2-4	Clustering (supporting virtually mapped shmem)	Thermal density and local storage requirement
Mezzanine board	IO Specialized node	2-8 processor boards	Virtually mapped regions, hostOS	Balance of IO vs communication
Processor Board	Daughter board	2 nodes < 100ns latency	Virtual mapped regions, guestOS	Physical sizing
Packaged 2D TSV flipchip	Compute Node	2-8 Units	Virtual shared memory (ccNUMA)	Balance compute vs communication
Micro-bumped Silicon “chiplet”	Compute Unit	4-16 CPU + GPGPU	Virtual memory Coherent SMP	Balance compute again memory
MCM in Package	Local Memory	8-32GB per unit	Unified memory	Primary Memory per processor

Prototype Example Configuration



8x CPU per unit, 4x unit per node, 8xG3 PCIe/40Gb/s per node, 9x nodes per local router

Latency from unit to unit via processor board ~ 0.5 us, ability to read/write L2 cache of remote unit

Use of Virtualization

- IO
  - Multiple nodes share the physical cost of ‘utility’ interfaces
  - Multiple Access Mac used to share high-speed interfaces
    - NIC masters directly into each guests local memory partition
- Processing
  - IO Node is the VMM host operating as a ccNuma machine over multiple physical compute nodes
    - Capable of 1:1 allocation between hardware and guest
- Memory
  - Any guest can be allocated memory up to the entire system image
  - Numa cost rules help with shared memory access latencies
- Communication
  - Direct processor read/write semantic between system memory
  - Shared memory MPI with DMA for long haul transfers

Summary

- ARM Architecture
  - Has evolved into a full 64bit, secure, high-performance solution capable of hardware accelerated virtualization
- ARM Licensee
  - Design both the processor core and system to target the requirements of specific applications and markets
- The flexibility opportunity of ARM based design for HPC hasn't really started
  - We see integration of the high-speed IO around the ARM core
  - We see scalable connectivity being integrated on the device
- The full opportunities of integrated system design has yet to be realized for enterprise and HPC requirement