WHITE PAPER

Virtualization is Coming to a Platform Near You

The ARM® Architecture Virtualization Extensions and the importance of System MMU for virtualized solutions and beyond

Roberto Mijat
Software Solutions Architect

Andy Nightingale
Product Manager – ARM Processor Division

Synopsis

Virtualization in the PC and server markets has provided measurable benefits over the last few decades with advanced virtualized server systems now achieving 60% or more capacity utilization with a corresponding ROI of 769%[1]

Virtualization in the mobile and embedded space can similarly enable hardware to run with less memory [2]and fewer chips, reducing BOM[3] costs and further increasing energy efficiency [4]. Virtualization also helps to address safety and security challenges, and reduces software development and porting costs by man years.

Hardware and software design teams from many of ARM's leading silicon partners and OEMs are already planning virtualization support into their 2011 design starts – are you?

Implementing efficient virtualized systems cost effectively requires hardware support. In particular memory management can provide great challenges and have severe repercussions on system reliability and performance. To address this ARM is introducing the Virtualization Extensions to its ARM v7 architecture and the System Memory Management Unit (SMMU) Architecture. This paper examines the rationale behind this, and explores how SMMU will enable vast reductions in software costs and complexity, and at the same time aligning with the ARM's ethos of low power, high performance designs.

[1] http://h18000.www1.hp.com/products/servers/management/vse/Biz_Virtualization_White_Paper.pdf
[2] For example: Dedicated linear memory regions for H/W access outside of the OS managed space are no longer required
[3] http://www.ok-labs.com/_assets/white_paper_motorola_evoke_teardown.pdf
[4] http://www.businessweek.com/technology/content/apr2008/tc20080421_235517.htm

Introduction

Every new generation of consumer devices has to satisfy the end user's expectation for efficient energy usage, richer applications and multimedia, faster performance and an overall secure user experience. Consider an imaginary world where in 2013 embedded and mobile platforms have **no** provision for hardware virtualization:

News headlines: "Car manufacturers set to recall 3.7 million units worldwide: 'Recent automobile engine management failures are linked to software bugs in factory-fitted entertainment systems' said a spokesperson, 'one such bug has been traced to cause a glitch that can pervade into a critical control system'"

News headlines: "Consumer woes with internet enabled TVs: 1000's loose control of their DTVs through hackers exploiting security flaws in integrated web-browsing software"

Blog site entry: "My employer has just replaced my mobile hand-set and most of my favourite apps aren't supported on this device. They also tell me I can't mix my personal contacts list, calendar and e-mail on their equipment because of security risks…"

Another downside of increasing system complexity materialises in rising software costs and overheads currently running at 50% of the cost of new 32nm and 28nm designs[5]. Virtualization technologies are being employed to address this and the problem scenarios mentioned above. Virtualization also enables to make a more efficient use of the hardware, to reduce software porting overheads by consolidating newer and legacy designs, in addition to enabling IP isolation and enhancing security, safety and reliability. Such benefits form an implicit guarantee that virtualization will play an important part as the underlying technology of choice for many OEMs and semiconductor suppliers for consumer products based on future SoC designs.

The Rise of Embedded Virtualization

Virtualization is a proven technology that enables the abstraction – or virtualization – of computing resources. A relatively small control program called **Virtual Machine Monitor** (VMM) or **Hypervisor** is placed between the OS and the hardware. Typically the VMM executes in privileged mode and can host one or more operating systems – **Guest OS**s – in a sandbox[6] called **Virtual Machine**: a controlled construct of the underlying hardware. Each Guest OS operates under the illusion of exclusive access to the processors, peripherals, memory and I/O. The VMM arbitrates access to all shared resources in a similar way to how a traditional operating system enables the concurrent executions of user processes.

Virtualization in the server market

For many decades virtualization has been instrumental in improving efficiency and reducing costs in the enterprise and home entertainment arenas. The ability to run multiple operating systems concurrently on otherwise underutilised hardware resources has enabled server **consolidation** (improved utilization of a single computer) and **aggregation** (improved utilization of many computers). IBM pioneered this in the 1960s to more

[5] http://chipdesignmag.com/display.php?articleId=1252&issueId=22
[6] http://en.wikipedia.org/wiki/Sandbox_(computer_security)

efficiently utilize expensive mainframe stations, by logically partitioning them into virtual machines, and making the multitasking of many applications possible. Similar dilemmas were faced by IT departments in the 1980s and 1990s where virtualization was eventually applied to the x86 Architecture in order to address issues such as high maintenance and management costs, high infrastructure costs, and insufficient failure and disaster protection. Key features of server virtualization technologies, such as the ability to perform **live migration** of entire execution environments in case of faults or failures make for a more robust and reliable system solution. Secure **sandboxing** of un-trusted software enables system designers to efficiently address the strictest security, legal or safety design concerns. Costs associated with **portability** inherent to large quantities of legacy software are dramatically reduced. Non-existent hardware can be simulated. Virtualization enables **rapid migration** to new hardware, freeing developers to focus on differentiating features and functionality. It is no surprise that IDC predicts that virtualization technologies will command a massive $11.7 Billion market by 2011[7].

The appeal of the above mentioned applications and benefits has prompted major technology providers to put virtualization into mobile and embedded devices[8]. Motorola, Intel and Texas Instruments have recently joined Cisco Systems as financial backers of VirtualLogix. Seeing great opportunities in this new market, Red Bend Software, market leader in Mobile Software Management solutions, acquired VirtualLogix in 2010. In late 2008 VMware acquired TRANGO Virtual Processors, who then announced the availability of their Mobile Virtualization Platform (MVP)[9]. 2009 has seen the launch of the first commercially available, fully virtualized handset in the Motorola Evoke™ QA4[10], followed in 2010 by the Motorola ATRIX™ 4G[11] dual-core hyper-phone. Open Kernel Labs' technologies shipments to have exceeded a billion devices to date[12]. Established embedded OS providers such as Wind River Systems and Green Hills Software have also made further significant investments in their embedded hypervisor offerings. This corroborating evidence confirms that embedded virtualization is soon to gain critical mass.

Exciting new Opportunities in Mobile

Virtualization technologies can be very valuable to a wide variety of mobile applications:

On a mobile device it is possible to consolidate the baseband processor with the application processor, in order to reduce BOM and shortening design and integration cycles. Unfortunately high level operating systems do not cater very well for the real time necessity of traditional communication software stacks generally found on baseband processors. For this reason it is desirable to keep the real time execution environment. This can be hosted on a virtual machine, with minimal or no porting overhead, and transparently and concurrently execute on the application processor.

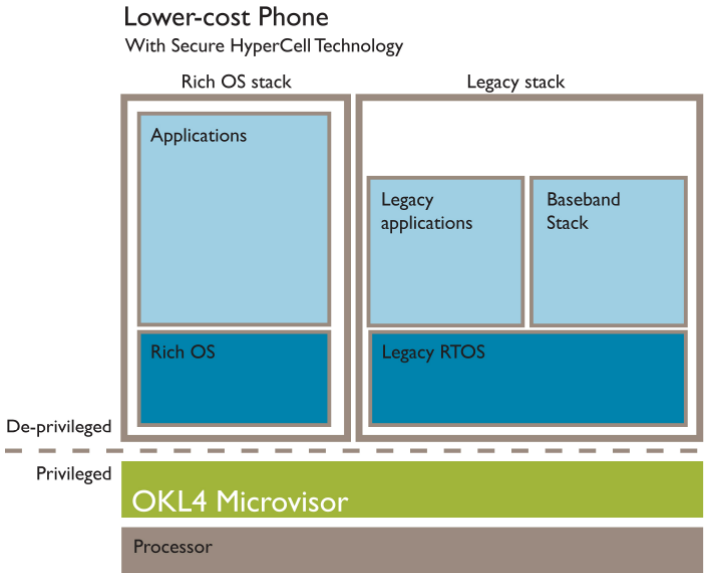

Figure 1 – Example using Open Kernel Labs' OKL4 Microvisor solution
Source: OK-labs (www.ok-labs.com)

[7] http://www.virtualization.info/2007/07/idc-predicts-virtualization-servic
[8] http://www.businessweek.com/technology/content/apr2008/tc20080421
[9] http://www.vmware.com/products/mobile/
[10] www.ok-labs.com/_assets/evoke.pdf Evoke is a trademark of Motorola Inc.
[11] ATRIX is a trademark of Motorola Inc.
[12] http://www.ok-labs.com/releases/release/open-kernel-labs-okl4-now-deployed-in-more-than-one-billion-devices

The VMM can be configured to guarantee that real time obligations are met.
Many mobile users own a personal phone as well as a work phone. This can be by choice or convenience, but more often it is dictated by restrictions imposed by corporate IT departments. With mobile virtualization it will be possible to host two (or more) Operating Systems on the same handset, simultaneously addressing the security restrictions and limitations aimed at safeguarding the integrity of the corporate network, as well as isolating the IP of open source OSs (for example Google Android) from proprietary offerings (for example Windows Mobile).

Users typically replace their handset every 18 months, motivated by the expiration of minimal contractual obligations from mobile network operators (MNOs) and the availability of newer models. In fact, sometimes a handset upgrade is the only venue to access the latest consumable technologies: for example a more powerful graphics accelerator may be needed to play the latest 3D games, or a newer modem, multimedia processor and larger screen to deliver the proper connected high-definition video experience. Having purchased many applications from one proprietary application store service, many individuals will want to move them to their new device, but this is currently not possible. Also some applications may not be available on an alternative handset/operator offering. With mobile virtualization it will be feasible to support all necessary non-native operating environments on the handset of choice, and run non-native applications on any Operating System. In addition, network operators, handset OEMs and semiconductor suppliers will be able to deploy a single software stack across multiple hardware platforms, and software programmers will not have to port applications for each operating system or platform.

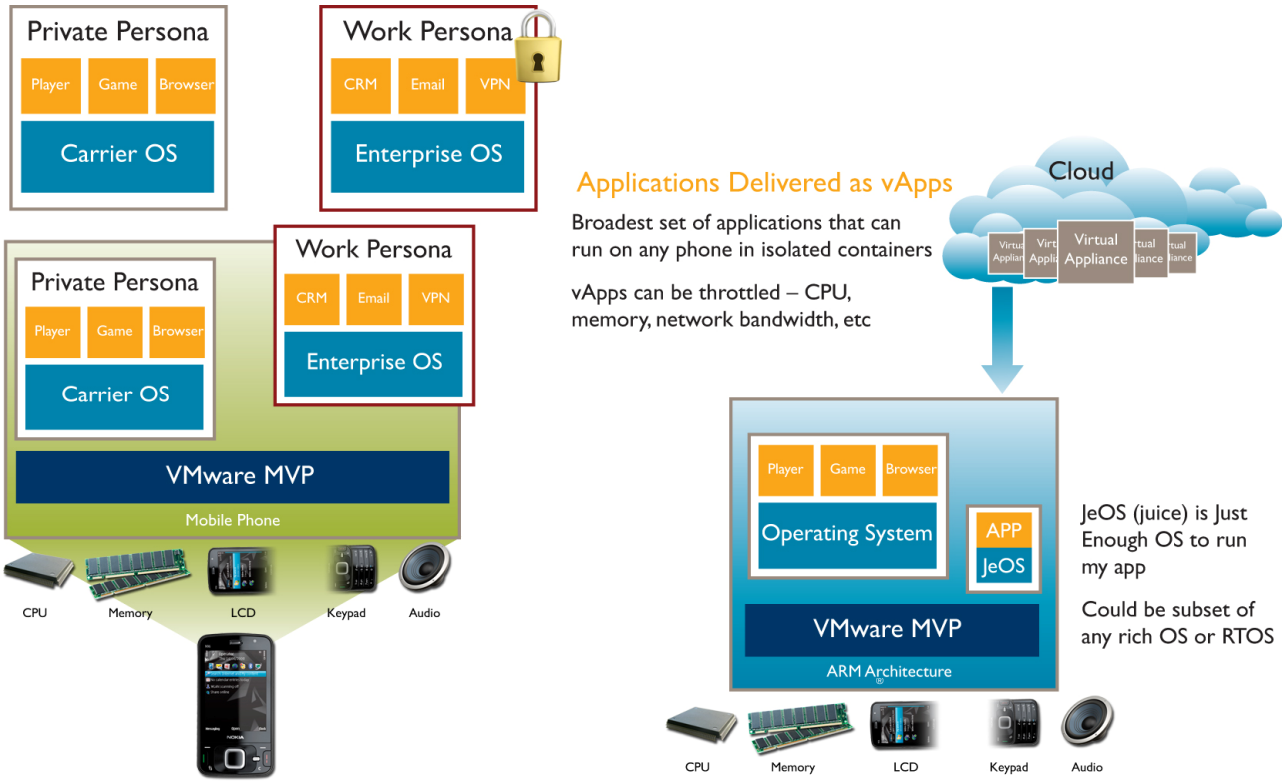

Figure 2 – Examples of mobile virtualization applications using VMWare MVP
Source: VMWare (www.vmware.com)

The list of actual and potential use cases for virtualization in embedded systems expands further.

Hardware support for Virtualization

Modern computer architectures provide multiple operational modes, each with a different level of authority (or privilege) in respect to the system's hardware resources, configurability and the execution of special instructions. The ARM architecture traditionally distinguishes between User mode and Supervisor mode[13]. The x86 architecture provides four so called Rings: from Ring-0, higher-privilege mode, to Ring-3, the lowest.

Operating Systems are generally designed to run on native hardware. The OS expects to be executing in the most privileged mode (Supervisor mode, or Ring-0) and assumes total control over the whole system. In a virtualized environment, it is the VMM that runs in privileged mode, whilst the OS will be executing at a lower privilege level (for example User mode or Ring-3).

When booting, a typical OS will try to configure the processor, memories, I/O devices and peripherals. When executing, it will expect exclusive access to such devices, including changing peripherals' configuration dynamically, directly managing the interrupt controller, replacing MMU Page Table Entries (PTEs), initiating DMA transfers, for example. There is a critical issue with this: When running de-privileged inside a Virtual Machine, the Guest OS will not be able to execute the privileged instructions necessary to configure and drive the hardware directly[14]. The VMM must handle this. In addition, the VMM may be hosting multiple Guest OSs, therefore direct modification of shared devices and memory requires cautious arbitration schemes.

Full Virtualization and Paravirtualization

The level of abstraction required to address this, and the inherent software complexity and performance overhead, are specific to the characteristics of the architecture, the hardware and the guest operating systems. The main approaches can be broadly categorized in two groups: Full Virtualization and Paravirtualization[15].
With **Full Virtualization** the guest OS is not aware of being virtualized, and it does not require any modification. The VMM traps and handles all privileged and sensitive instruction sequences, while user level instructions run unmodified at native speed (assuming ISA compatibility – otherwise emulation by binary translation is required). Full Virtualization offers the best isolation and security for virtual machines, and simplifies migration and portability as the same guest OS instance can run virtualized or on native hardware. The trade-off is in performance and complexity of the VMM. Examples of Full Virtualization include Green Hills Software's INTEGRITY® OS Secure Virtualization (ISV) solutions.

[13] Version v6kz of the ARM Architecture introduced the Security Extensions, commercially known as the TrustZone® technology. Trustzone enables system wide security via an additional execution mode, the Secure Monitor Mode. Effectively this can be used to implement a form of full virtualization, allowing for a single guest OS, and it is supported as such in commercial offerings like Green Hills Software's INTEGRITY® Secure Virtualization (ISV) solutions.
[14] On ARM this includes: MMU instructions, SWIs, MSR and MRS instructions that read and modify the Program Status Register, MRC and MCR instructions that move values to and from a coprocessor etc.
[15] VMMs are often further categorized as Type-1 Hypervisors if they have native control of the hardware (like Green Hills Software INTEGRITY® OS, Open Kernel Labs OKL4, VMWare VMP for example), or Type-2 Hypervisors if they operate within the context of a host OS (like VMware Server, Parallels Workstation, QEMU for example). INTEGRITY® is a registered trademark of Green Hills Software Inc.

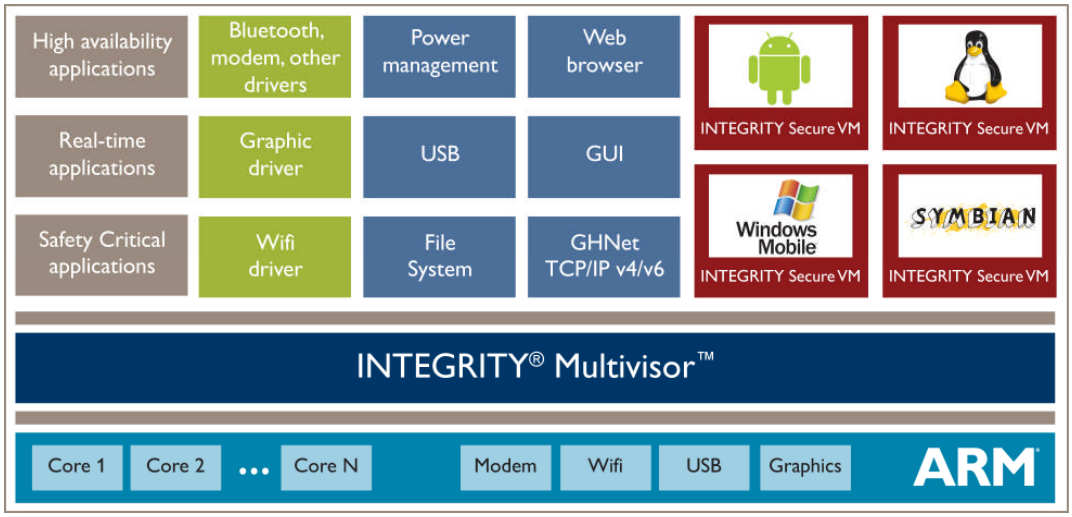

Figure 3 - Green Hills Software INTEGRITY Secure Virtualization (ISV) solutions
Source: Green Hills Software (http://www.ghs.com/)

A workaround to such overheads is **Paravirtualization**. The Guest OS is modified to have direct access to the VMM via so called hyper-calls or hypervisor calls. A special API is exposed by the VMM in order to allow Guest OSs to execute privileged and sensitive instruction sequences. This approach is endorsed by solutions such as Xen, WindRiver Hypervisor, VMware and Open Kernel Labs products, for example. Open Kernel Labs provides a Hypervisor (OKL4 Microvisor) and paravirtualized ports of popular operating Systems such as Linux, Android™ OS[16], Symbian OS™[17] and Windows Mobile™ OS[18].

The ARM Virtualization Extensions

In an ideal world neither binary translation nor Paravirtualization would be necessary and Full Virtualization would enable Guest OSs to run at near native speed. For this to happen, hardware assistance and possibly extensions to the ISA are necessary. Hardware virtualization extensions simplify existing software-only solutions by reducing and sometimes eliminating the burden of trapping and emulating instructions executed within a Guest OS. The VMM can then efficiently virtualize the entire instruction set by handling sensitive instructions using a classic trap-and-emulate model in hardware, as opposed to software. Intel® and AMD came with distinct implementations of hardware-assisted x86 virtualization: Intel® VT[19] and AMD-V™[20], respectively. ARM® is introducing virtualization support extension to its Architecture with the ARM v7 Virtualization Extensions.

The **ARM Virtualization Extensions** enhance the ARM v7-A Architecture to support the development of virtualized systems. The fundamental elements of these extensions are:

[16] Android is a trademark of Google Inc. Use of this trademark is subject to Google Permissions (http://www.google.com/permissions/index.html)
[17] Symbian OS is a trademark of Nokia.
[18] Windows and Windows Mobile are trademarks of Microsoft Corporation in the U.S. and other countries.
[19] Intel and Intel VT are trademarks of Intel Corporation in the U.S. and other countries.
[20] AMD and AMD-V are trademarks of Advanced Micro Devices Inc.

- The introduction of a new Hypervisor execution mode, of higher priority than Supervisor mode[21]. This will enable the VMM to execute at a higher privilege than the Guest OSs, and the Guest OSs to execute with traditional operating system privileges, removing the need to employ Paravirtualization techniques.
- The provision of mechanisms to aid interrupt handling, with native distinction of interrupt destined to secure monitor, hypervisors, currently active Guest OSs or non-currently-active Guest OSs. This will dramatically reduce the complexity of handling interrupts using software emulation techniques and shadow structures inside the VMM.
- The provision of a System MMU to aid memory management, supporting: multiple translation contexts for multiple DMA capable masters, two levels of address translation and hardware acceleration and abstraction.
- Debug functionality aimed at enabling debugger access to individual Guest OSs.

Memory management challenges in virtualized systems

In a virtualized system, the subject of memory management is very important and can lead to substantial complexity.

One of the key functions of most operating systems is to support a stage of virtual memory management to partition the physical memory controlled by the operating system across multiple processes. In a system where each Guest OS is running inside a Virtual Machine, the memory that is being allocated by the Guest OS is not the true physical memory of the system, but instead it is an intermediate physical memory. The VMM directly controls the allocation of the actual physical memory, thereby fulfilling its role of arbiter of the shared physical resources.

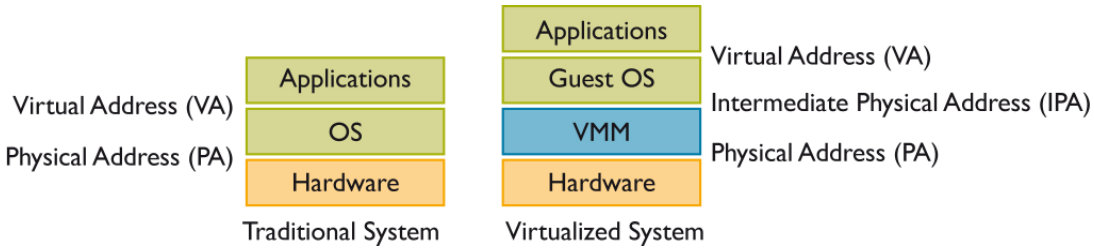

Figure 4 - Address translation stages in traditional and virtualized systems

There are two approaches to handling the two stage of address translation (VA to IPA and IPA to PA). In current systems where only one stage of memory address space translation is provided in hardware, for example using the MMU in the CPU, the hypervisor must manage the relationship between VA, IPA and PA directly. This is generally done by the hypervisor maintaining its own translation tables (called shadow translation tables), which are derived by interpreting each Guest OS translation tables. The hypervisor must ensure that all changes to the Guest OS translation tables are reflected in the shadow structures, as well as enforcing protection and redirecting access faults to the appropriate stage. The required mechanism can be complex and add performance overhead. The alternative is to use hardware assistance for both stages of translation, and this is what the ARM SMMU enables.

[21] TrustZone's Secure Monitor mode is still available as the highest priority mode in the system, and can be dedicated to perform its function as a secure execution mode for sensitive software.

Memory fragmentation

Depending on how the VMM is implemented and how the Guest OS is modified, there is a range of plausible relationships between the Intermediate Physical Address (IPA) and the Physical Address (PA) memory spaces. The IPA and the PA can be flat-mapped (removing the need for translation) or offset by a constant, with the VMM managing memory protection. This allows each OS to own a proportion of contiguous physical memory. In reality memory is expensive; therefore it is more common that any block (typically a page) of physical memory is allowed to be mapped independently to each Guest OS. This is referred to as a **fragmented relationship**. The ARM Virtualization Extensions allow for a fragmented relationship between the IPA and the PA spaces.

As the number of Guest OSs increase, the relationship between the IPA memory of a Guest OS and the PA memory can become complex. Some systems may envisage the runtime creation and destruction of Guest OSs, or regions of physical memory may be shared between multiple Guest OSs. The fragmentation of physical memory can be a real issue in systems where the use of large sections of contiguous physical memory is important.

Multiple DMA capable masters

The introduction of a new level of address translation has important implications on other data processors and peripherals in the system that may be sharing the same physical memory medium.

Generally peripheral devices are owned by the OS and programmed in PA space. Also, all devices that are capable of direct memory access (DMA) generate addresses in the PA space. In a virtualized environment these physical addresses are in fact IPAs, and therefore require further translation. Currently this would be handled by the VMM, either by software emulation or by direct device assignment techniques (native access to the device by Guest OSs arbitrated by the VMM). Either solution requires device driver porting, applicable to each Guest OS, and can add considerable complexity to the VMM as well as causing significant performance overheads[22].

This is a significant issue: Consumer electronics platforms are complex heterogeneous systems, where a wide variety of processors with direct access to the memory system co-exist. For example a typical mobile computing platform like NVIDIA®'s Tegra 2™[23], Texas Instruments®' OMAP4™ platform [24] or ST-Ericsson®'s U8500™ platform [25], includes a multi-core ARM application processor with integrated MMU for each core, an OpenGL ES 2®[26] capable Graphics Processing Unit (GPU) with its own integrated MMU, a Digital Signal Processing (DSP) unit, an Image Signal Processor (ISP), a Power Manager (PM) microcontroller and a Direct Memory Access Controller (DMAC). In the not too distant future it can be envisaged that heterogeneous distributed computing models such as OpenCL-driven General Purpose GPU (GPGPU) will be common place. In addition to the aforementioned complexity and performance overheads associated to the address translation process, there isn't a fail-proof mechanism to stop any of the DMA capable devices or processors from erroneously corrupting physical memory utilized by another Guest OS.

[22] Some VMM implementation such as Open Kernel Labs' OKL4 embedded hypervisor, mitigate these overheads by providing a flexible approach to device drivers where no additional porting is required (although for certain incremental benefits extra driver porting may be desirable) and without adding much complexity or performance overheads to the VMM.
[23] Tegra 2 is a trademark of Nvidia Corporation in the U.S. and other countries.
[24] OMAP4 is a trademark of Texas Instruments Corporation in the U.S. and other countries.
[25] U8500 is a trademark of ST-Ericsson.
[26] OpenGL is a registered trademark and the OpenGL ES logo is a trademark of Silicon Graphics Inc. used by permission by Khronos.

Guaranteeing system integrity

Another major issue can be exposed when considering virtualization for hardware consolidation in systems where one Virtual Machine is dedicated to safety or security critical purposes. For example, in an automotive subsystem, virtualization can enable the strict isolation of driving-related applications (rear-view camera, automatic parking system) and driver authentication from less critical applications associated with the infotainment and comfort systems (Head Unit, Front Electronic Module, Rear Seat Entertainment Unit etc). Malfunctions, faults or hacks in a non-critical runtime environment cannot be allowed to compromise other parts of the system.

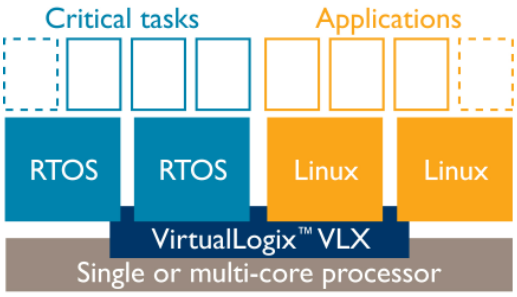

Figure 5 – VirtualLogix (now RedBend) VLX Hypervisor used to isolate critical tasks
Source: VirtualLogix (http://www.redbend.com/)

The System MMU (SMMU)

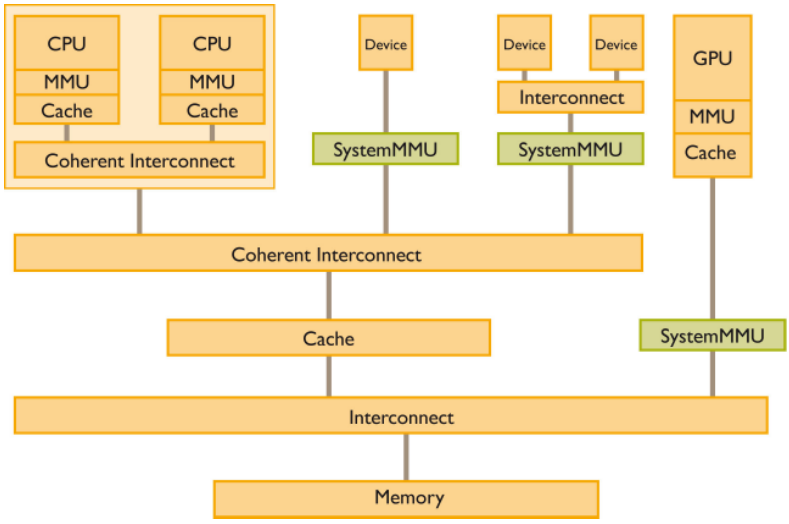

Figure 6 - Examples of where a System Memory Management Unit (SMMU) could locate in the system. Coherent interconnects ensure cache coherency between masters

In order to address all of the aforementioned issues and limitations, the ARM Architecture Virtualization Extensions introduce the System Memory Management Unit (SMMU) concept to the ARM Architecture.
A System MMU is a hardware device designed to provide address translation services and protection functionalities to any DMA capable agent in the system other than the main CPU. This includes hardware accelerators such as GPUs and Video Engines (VEs), simple DMA controllers as well as complete sub-systems. The SMMU can be implemented as a standalone device or integrated with an existing DMA capable processing unit. The diagram gives a few examples of where the SMMU could locate in the system.

The SMMU is designed for use in a virtualized system where multiple Guest OSs are managed by a VMM. To this purpose two separate address translation stages are supported. The first stage, or Stage 1, implements Virtual Address (VA) to Intermediate Physical Address (IPA) translation and is designed for use by the Guest Operating System. The second stage, or Stage 2, translates Intermediate Physical Address (IPA) to Physical Address (PA) bringing a host of benefits to the VMM. The properties and benefits of the SMMU also extend to non-virtualized systems.

The system designer has the option of implementing either stage 1 or stage 2 address translation stages for each SMMU. For example the SMMU can be placed in front of a DMA capable device that does not already have a MMU, and configured to provide Stage 1 translation to enable such device to view fragment physical memory as contiguous. The SMMU can also be configured to provide Stage 2 translations only, for example to enable a device that already has an MMU to be managed by a system hypervisor.

Stage 1 SMMU Translation

Stage 1 translation is intended to assist Operating Systems, both when running natively or inside a VM. The SMMU provides multiple benefits and addresses several issues associated with implementing otherwise costly software workarounds – as described previously in this document.

Stage 1 translation works similarly to a traditional CPU MMU. The following diagram summarises the application and functionality of Stage 1 Address Translation.

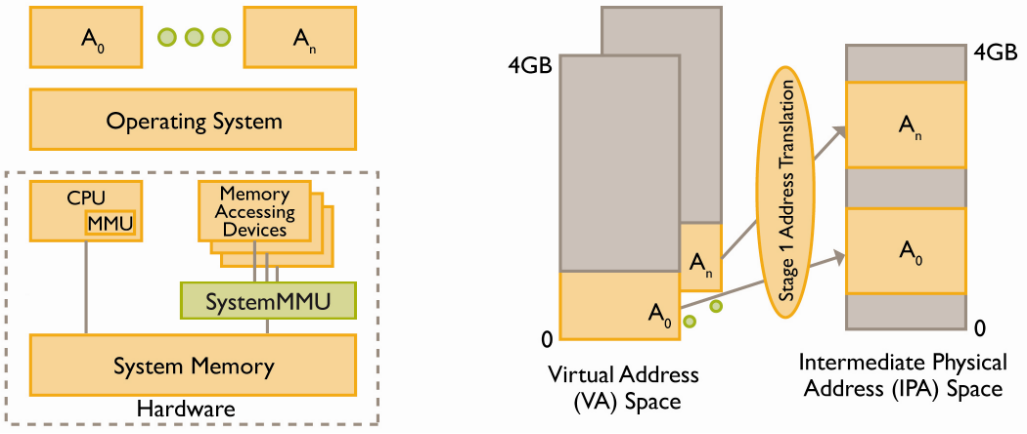

Figure 7 - Application and functionality of the Stage 1 Address Translation

Characteristically, Operating Systems cause much fragmentation of physical memory by continuously allocating and freeing space on the heap, both for kernel and applications. A system that implements a fragmented model between IPA and PA spaces, where multiple Guest OSs are sharing the same physical medium, will suffer even more because of this issue. A typical consumer device hosts applications that by nature require large amounts of contiguous memory to work efficiently. For example, a digital camera needs a large space to quickly dump pictures that have just been taken; a Smartphone device may need to play back some video content, therefore necessitating a large working buffer for the video decoder. It is very common for large contiguous blocks of physical memory not to be available for dynamic allocation due to physical memory fragmentation. A typical solution is to pre-allocate such buffers. This is very inefficient since the buffer is only required at runtime, and the additional amount of physical memory that is reserved for this purpose (and is not usable by general purpose OS activities) directly adds to the Bill of Materials (BOM). Also, in a virtualized configuration, this solution will require direct modifications to the VMM.

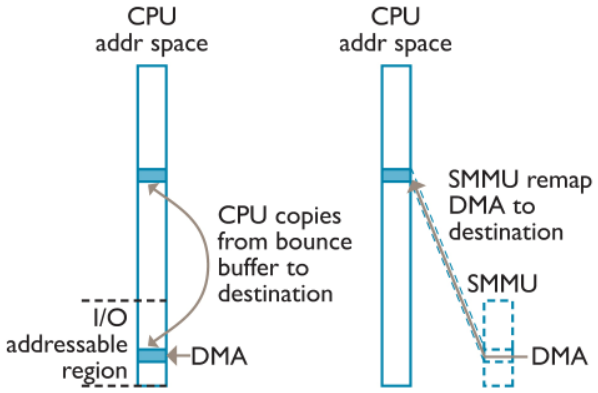

Figure 8 – Using the SMMU to avoid bounce buffers

When a DMA capable device needs to operate on fragmented physical memory, the typical solution is the adoption of software techniques such as DMA Scatter-Gather. This has complexity and performance overheads. With a SMMU this overheads can be avoided by enabling a further level of address translation, where smaller blocks of memory (down to a granularity of 4kb) in Physical Address space can be virtually gathered to expose a

contiguous block of memory at Virtual Address space (if used in a non-virtualized model) or from Intermediate Physical Address space to Virtual Address space (if used in a virtualized model)[27].

Another major issue is uncovered when some devices in the system cannot access the full range of memory available to the CPU, such as 16 or 24 bit devices on 32 bit architectures, or 32 bit devices on 64 bit architectures. Traditionally the solution has been to provide an intermediate area of memory at a low address to act as a bridge. This is known as a bounce buffer. The operating system allocates pages in an address space visible to the device and uses them as buffer pages for DMA to and from it. Once I/O completes, the content of these pages is copied by the kernel into its destination, outside of the addressable range of the I/O device. There is significant overhead to this operation as at the very least it involves copying a full page. Bounce buffers can be totally avoided by using a SMMU. Stage 1 translation can enable any DMA agent to access any address in the system without limitations associated to its bus width[28].

Stage 2 SMMU Translation

Stage 2 translation is intended to benefit Virtual Machine Monitors. The following diagram summarises the application and functionality of Stage 2 Address Translation.

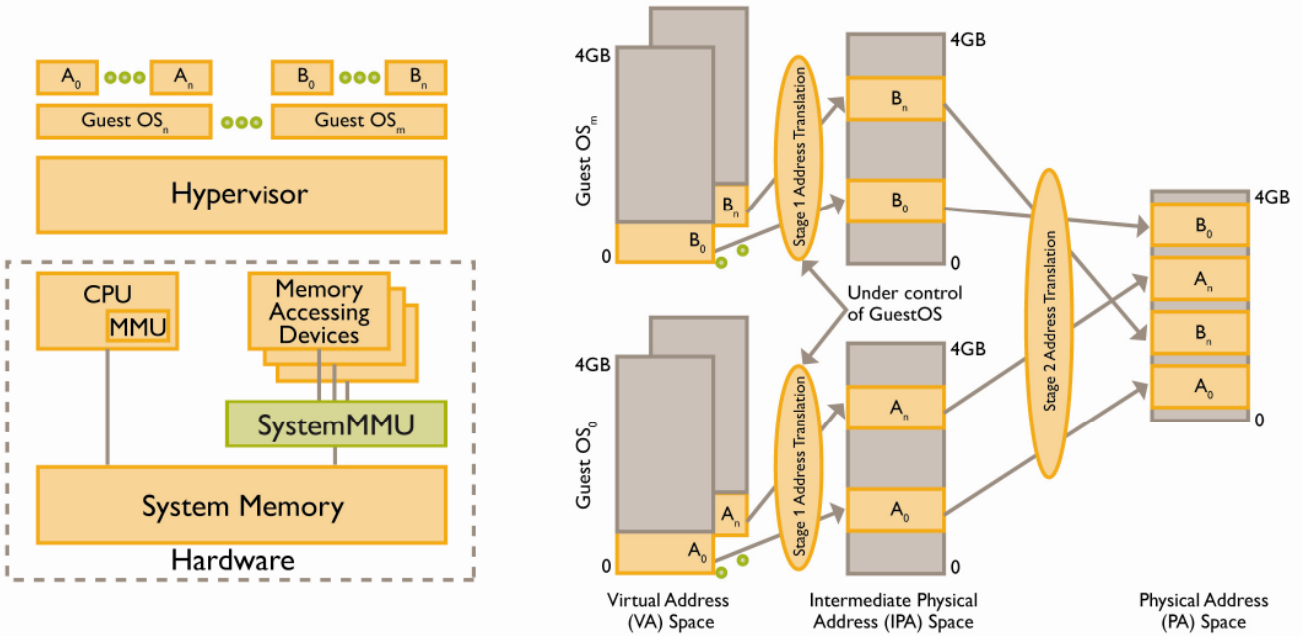

Figure 9 - Application and functionality of the Stage 2 Address Translation

[27] This is similar in principle to using a GART (Graphics Address Remapping Table). The GART was designed to allow graphics processors to read textures directly from system memory, using address translation to gather them into a contiguous region mapped to an address visible to the graphics processor (known as the graphics aperture). The GART is fixed to the size of the graphics aperture and provides no memory protection mechanisms. The SMMU can translate all addresses and in addition it also provides protection mechanisms.

[28] To address physical memory larger than 4Gbytes the second stage translation system can be used instead.

Adding a SMMU device for Stage 2 translation removes the need for the hypervisor to manage shadow translation tables totally in software. Hardware assisted address space translation has clear performance benefits.

With Stage 2 address translation, the SMMU enables any Guest OS to directly configure all DMA capable devices in the system whilst sharing with them the same address space at IPA level.

The SMMU can also be configured to ensure that devices operating on behalf of one Guest OS are prevented from corrupting memory of another Guest OS.

The Stage 2 address translation system is based on a translation scheme with a 64-bit descriptor to allow it to address a physical memory larger than 4GBytes.

Providing hardware separation between the two stages of address translation enables to clearly define the ownership of the different stages between the Guest OS (Stage 1) and the VMM (Stage 2). Translation faults can therefore be routed by hardware to the appropriate level of software, allowing management functions (TLB management, MMU enabling, register configurations) to be handled at the appropriate stage of the translation process. This can improve performance by greatly reducing the number of entries into the VMM.

Conclusion

This paper discusses just some of the compelling roles of virtualization in the mobile and embedded computing space, building on the current success of virtualization in the server market. Many market leaders in hardware and software development for the ARM architecture are investing now in virtualization technologies, opening up several exciting opportunities for the ARM ecosystem.

By introducing the Virtualization Extensions to its architecture, and in particular the System Memory Management Unit Architecture, ARM delivers a cohesive solution that enables vast reductions in software costs and complexity with the following benefits:

- Reduction in software porting overheads by consolidating new and legacy designs
- Hardware assisted protection of devices for enhancing security
- Promoting system robustness and reliability for safety critical systems
- Reduced BOM costs, more efficient use of the hardware and memory resources

The ARM Architecture is synonymous with low power and efficiency. The Virtualization Extensions are no exception to this, and further enable ARM to address markets where the benefits of virtualization have been well understood, for example the server market. Hardware and software design teams from many leading ARM silicon partners and OEMs are already planning virtualization support into their 2011 design starts – are you?

For more information on how you can integrate SMMU into your 2011 design starts, please contact:
andy.nightingale@arm.com