From Embedded Multi-core SoCs to Scale-out Processors

Embedded Tutorial

Marcello Coppola,
ST Microelectronics
Grenoble, France
marcello.coppola@st.com

Babak Falsafi
EcoCloud
Lausanne, Switzerland
babak.falsafi@epfl.ch

John Goodacre
ARM Ltd.
Cambridge, UK
John.Goodacre@arm.com

George Kornaros
TEI of Crete
Heraklion, Greece
kornaros@epp.teicrete.gr

Abstract—***Information technology is now an indispensable pillar of a modern day society. CMOS technologies, which lay the foundation for all digital platforms, however, are experiencing a major inflection point due to a slowdown in voltage scaling. The net result is that power is emerging as the key design constraint for all platforms from embedded systems to datacenters. This tutorial presents emerging design paradigms from embedded multicore SoCs to server processors for scale-out datacenters based on mobile cores.***

Keywords—MPSoC, ARMv8 architecture, hardware virtualization, IOMMU, scale-out processors, total cost of ownership

I. INTRODUCTION

With the end of Dennard Scaling in sight, and a slowdown in supply voltage scaling, the entire IT industry and the corresponding digital platforms are going through a energy-induced infliction point. Energy efficiency which used to be the key design concerns of embedded and mobile systems are now the ultimate design constraint for all digital platforms from sensors to datacenters. While in mobile platforms the key concern is enhanced battery lifetime, in servers energy consumption dictates the overall operation costs with significant economic implication for datacenter owners.

This tutorial includes three presentations, covering the use of energy-efficient mobile cores in designs spanning from embedded multi-core SoCs to scale-out processors in servers. The abstracts for the presentations appear as follows. Section II introduces the next generation multicore ARM-based SoC. Section III presents *vIrtical*, an architecture for efficient virtualization in the embedded world. Section IV describes the Scale-out Processor Architecture, a methodology for building technology-scalable servers using embedded cores. We highlight the concluding remarks in Section V.

II. THE 64-BIT MULTICORE ARM-BASED SOC

Mobile usage has changed significantly in recent years, with today's consumers increasingly using their smartphones for most of their daily activities. Today's smartphones require strong capabilities and rich user experience, as they have taken over many of the functionalities of desktop computers. These include complex tasks such as media streaming, navigation and gaming, and less demanding background tasks such as voice calls, social networking, and e-mail services. As a result, the mobile phone has become an indispensable computing device for many consumers. As we continue to usher in this new era of computing, mobile device designers find themselves focusing on how to deliver devices that allow for increased data consumption, lower response time, faster and reacher connectivity, and prolonged battery life [2].

Today's embedded devices are rapidly changing to meet the ever-increasing application requirements. Systems-on-Chip (SoC) integrate more and more functionality on the chip, as advanced technology nodes allow for an unimaginable level of integration on a single device. Such devices have the need and the ability to support richer and more complex software. The demand on the memory used by these applications is continuously growing. The complexity of a single application requires addressing more memory than can be addressed by a 32-bit architecture alone.

Due to ever-increasing computational demands, designers are forced to look beyond single-core solutions to deliver powerful, energy-efficient processors. To this end the ARMv7 architecture and the Cortex-A15 processor were introduced, supporting other key market requirements such as Large Physical Address Extensions (LPAE) and hardware-assisted virtualization. LPAE introduced a new page table format supported by Cortex-A15, which allows mapping of multiple 32-bit virtual address spaces to a 40-bit physical address space using a 4KB page resolution. This fully addresses the need for the increased SoC integration and the address map restrictions that limit the capabilities of many of today's SoC devices.

Additionally, ARM's big.LITTLE processing addresses the challenge of designing a System-on-Chip (SoC) capable of delivering both higher performance and higher energy-efficiency within a single processor subsystem [4]. The Big.LITTLE design combines a high-performance Cortex-A15 multiprocessor cluster with a Cortex-A7 multiprocessor cluster offering up to 4x the energy efficiency of current

978-3-9815370-0-0/DATE13/©2013 EDAA

designs. This coupled design opens up the potential for a variety of new applications and use cases by enabling optimal distribution of the load between big and LITTLE cores and by matching application needs to the cores' computational capabilities. The Big and the LITTLE cores are architecturally compatible and expose the same functionality, including support for more than 4 GB of memory, virtualization extensions, and functional units such as NEON advanced Single Instruction, Multiple Data (SIMD) instructions for efficient multimedia processing and floating-point support. This enables applications compiled for one processor type to run on the other without modification. Because an application can run both on a Cortex-A7 and Cortex-A15, it becomes possible to map the application to the right processor based on its computational needs, performance requirements and the device's energy constraints.

The ARMv7 architecture enables multiple applications to access 4GB of memory. Unfortunately, the operating system processes are also limited to a 4GB address space. Due to a constant increase in the available physical memory and ever-increasing number of simultaneously running applications managed by the OS, the address space limitation for the OS becomes a scalability bottleneck. The ARMv8 architecture has a clean and well-architected approach to allow the operating system to run within the 64-bit virtual address mode of AArch64, while the user application space can all still run in the AArch32 state [1][3]. This gives a solution that is the best of both worlds: the ability to run an effectively unlimited number of full performance 32-bit user applications, with the OS efficiently operating in the AArch64 bit mode of the ARMv8 device.

Fundamental to ARMv8 is the new instruction set, known as A64: the encoding of instructions to enable an application to utilize a 64-bit machine. ARM took the decision to introduce 64-bit architectures through a new instruction set rather than extension of the existing instruction set for many reasons. Power is the most important reason, because a new instruction set could be developed to execute code at lower power compared to the traditional approach of adding instructions to the existing instruction set. Of course, for compatibility reasons, the entire ARMv7 machine is still supported in the new ARMv8 architecture, but when running 64-bit software, this part of the machine is not being used, and the area of complex legacy it had built up does not need to be active when running in the 64-bit code, unlike other architectures where the 64-bit extension was simply added to the historical complexity and legacy of their 32-bit mode. The new ISA drew upon the years of experience of building different micro-barchitectural implementations, and it was defined so that these new processors can be more easily optimized for low-power operation – an opportunity not really offered since the first ARMv4 machine, which resulted in the now-legendary low power ARM7 processors.

The design changes introduced in ARMv8 include more advanced branch prediction techniques, addition of more 64-bit registers and integration of the floating-point units and SIMD data engine. The memory model in AArch64 provides support for 48-bit virtual and physical address spaces and a two-stage translation from an application virtual address (VA), to the intermediate physical address (IPA) used by any hypervisor, and finally to the actual physical address (PA) placed on the memory bus.

The new ARMv8 architecture offers the opportunity to optimize the software exception model and to improve the silicon utilization, providing the opportunity to lower system power consumption even further. The new architecture enables software to support the 32-bit heritage, but unlike the earlier Thumb instruction sets, the transitions between 32-bit and 64-bit execution states occur on exception boundaries, known as inter-processing, and not through the traditional ARM to Thumb interworking. This creates a strict hierarchy between the various supported privilege levels of execution.

ARM and a number of key ARM's partners are currently leading efforts to develop first processor implementations of the ARMv8 architecture, targeting various markets. Some of the markets are very new to ARM, others are much more traditional, but the key drivers are common to all of them: higher performance at lower power.

III. Heterogeneous Multicore Architecture for Embedded Virtualization

Embedded architectures are becoming more computationally powerful and complex. Today's embedded devices are capable of executing several applications that must preserve security, allow for data sharing in a coherent way, and provide the required levels of performance. At the same time, embedded devices must be more energy-efficient than ever before and use their resources and energy carefully.

Virtualization has been traditionally used in the server domain to allow for higher resource usage, flexibility, isolation and better fault tolerance, all of which are needed in the embedded domain as well. The use of virtualization is crucial for embedded systems, whose resources are inherently limited due to their energy constraints [7], and becomes more important as embedded cores enter the server market. For example, virtualization allows set-top boxes or smart TVs to run multiple virtual machines simultaneously enabling consolidation of hardware and its higher utilization. This means that multiple middleware or OS instances can run on a single device simultaneously, promoting interoperability among heterogeneous systems. In addition, virtualization allows STBs and Smart TVs to provide of support for legacy systems while upgrading to new hardware and services. Therefore, legacy services can still be used, running on virtual machines, even when the operating systems and platforms are no longer supported by the manufacturers, thus saving hardware and operational costs.

In order to provide better efficiency for embedded systems, virtualization has to address performance challenges associated with hypervisors. This is achieved through hardware support for virtualization [7][12]. Similarly to server systems, hardware virtualization techniques remove most of the software overhead for common operations that can be efficiently handled in energy-efficient hardware. The vIrtical

approach for virtualization can also bring additional benefits such as enhanced isolation and security.

Various vendors are working on virtualization technologies to separate broadband services (such as Android) from the broadcast services (such as NDS, HbbTV) available within the "Walled Garden" via multi-compartment that can be implemented using virtualization. In a smart home environment with high degree of heterogeneity in terms of services, virtualization will, for instance, provide to the home dwellers the option to select a suitable operating systems for each application or service.

The virtualization model in VIRTICAL [5][11] features software and hardware support for two hypervisors: one that supports full virtualization and the other supporting the para-virtualization model for the secure environment, which exploits the ARM Secure Execution Environment. Depending on the application domain we can use both hypervisors, or a single one. A virtual machine can accommodate an operating system or a simple API above which an application is running. The resource management is delegated to the secure environment. Isolation between virtual machines is the main advantage provided by virtualization. It is usually split in two categories: temporal isolation and spatial isolation.

The *vIrtical* system platform comprises a set of virtualizable components, such as the processor cluster Cortex A15. It introduces novel hardware components, such as IOMMU, which extends the hardware support for virtualization to I/O components and hardware accelerators (GPPA). Finally, it extends the system-level coherence not only to processors but also to hardware accelerators, enabling memory sharing between system components without any software intervention. In order to deploy the system-level coherence, the vIrtical platform uses a sophisticated NoC-based communication infrastructure based on three on-chip-networks: a cache-coherent memory NoC based on ARM's CCI-400 interconnect, a system NoC based on Spidergon STNoC and a local NoC interconnecting processors or clusters internally within the GPPA.

An architectural countermeasure for resource-constrained embedded systems to achieve energy-efficient (MOPS/W) objectives is heterogeneous system design. Heterogeneity in embedded systems is typically achieved through the adoption of accelerator-based SoC designs, where an SMP multicore host processor is coupled to various accelerators: GPU-like general-purpose programmable many-cores (GPPA in the following) and several types of Hardware Processing Units (HWPU), which implement the key computational kernels of the target application domains in hardware. Accelerator-based SoCs are already wide-spread (Qualcomm's Snapdragon, Nvidia Tegra, Apple Ax, TI OMAP) and feature an on-chip GPU-like accelerator, which the host processor can leverage to offload data-intensive computational kernels, achieving significant speedups and higher energy-efficiency even in common, general-purpose applications. While such embedded GPUs are optimized for data-parallel (SIMD) computation, the focus in the vIrtical project is on general-purpose many-core accelerators which support a more flexible execution model (both data and task parallelism), such as the Hypercore Architecture Line from Plurality [5], or ST Microelectronics Platform 2012 [10].

Managing resources in virtualized systems imposes additional challenges. The hypervisor intercepts requests to a virtual I/O device made by a device driver in the guest. Then, the requests from multiple guests must be scheduled to an underlying physical I/O device, usually via another device driver managed by the hypervisor or a privileged virtual machine with direct access to physical hardware [8][13]. After processing of I/O requests by the device, the reverse processing is performed through virtual interrupts for the virtual device managed by the guest operating system. These overheads are prohibitive particularly for high-speed devices such as fast network interfaces, or graphic processors.

To address these inefficiencies a novel I/O Memory Management Unit component (IOMMU) is architected to enable mapping of virtual addresses from multiple devices to the correct VM's physical memory locations, additionally offering enhanced protection, scatter-gather functions on distributed memory organizations, higher performance through a configurable TLB and an integrated lightweight hardware monitoring unit to facilitate dynamic system optimizations. This new IOMMU is designed in a modular way and supports address translation along with the protection functionality. IOMMU ensures device isolation by safely mapping a device to a particular guest without compromising the integrity of other guests. Additionally, the IOMMU is designed to provide increased security in scenarios without virtualization; with the aid of the IOMMU, the operating system is able to protect itself from malicious device drivers by limiting a device's memory accesses and managing the peripheral devices' permissions.

IV. SCALE-OUT PROCESSORS

We are in the midst of an information revolution, driven by ubiquitous access to vast data stores via scale-out datacenters powering cloud services that can house tens of thousands of servers that are necessary for high scalability, availability, and resilience. The massive scale of such datacenters requires an enormous capital outlay for infrastructure and hardware, often exceeding $100 million per datacenter. Similarly expansive are the power requirements, typically in the range of 5 to 15 MW per datacenter, totaling millions of dollars in annual operating costs. With demand for information services skyrocketing around the globe, efficiency has become a paramount concern in the design and operation of large-scale datacenters.

To reduce infrastructure, hardware, and energy costs, data-center operators target high computing density and power efficiency. Total cost of ownership (TCO) is an optimization metric that considers the costs of real estate, power delivery and cooling infrastructure, hardware acquisition costs, and operating expenses. Because server acquisition and power costs constitute the two largest TCO components, servers present a prime optimization target in the quest for more efficient datacenters. In addition to cost, performance is also critical in scale-out datacenters designed to service thousands

of concurrent requests with real-time constraints. The ratio of performance to TCO (performance per dollar of ownership expense) is thus an appropriate metric for evaluating different datacenter designs.

Scale-out workloads prevalent in large- scale datacenters rely on in-memory processing and massive parallelism to guarantee low response latency and high throughput. Although processors ultimately determine performance characteristics of a server, they contribute just a fraction of the overall purchase price and power burden in a server node. Memory, disks, networking equipment, power provisioning, and cooling all contribute substantially to acquisition and operating costs. Moreover, these components are less energy proportional than modern processors, meaning their power requirements don't scale down well as the server load drops. Thus, maximizing the benefit from the TCO investment requires getting high utilization from the entire server, not just the processor.

To achieve high server utilization, datacenters must employ processors that can fully leverage the available bandwidth to memory and I/O. Conventional server processors use powerful cores designed for a broad range of workloads, including scientific, gaming, and media processing. As a result, they deliver good performance across the workload range, but they fail to maximize either performance or efficiency on memory-intensive scale-out applications [14]. Emerging server processors, on the other hand, employ simpler core microarchitectures that improve efficiency but fall short of maximizing performance. What the industry needs are server processors that jointly optimize for performance, energy, and TCO.

Multicore processors common today are well-suited for massively parallel scale-out workloads running in datacenters. First, they improve throughput per chip over single-core designs. Second, they amortize on-chip and board-level resources among multiple hardware threads, thereby lowering both cost and power consumption per unit of work (that is, per thread). The combination of powerful cores and relatively large chip size leads us to classify conventional server processors as big-core, big-chip designs.

Recently, research has shown simple-core designs to be well-matched to the demands of many scale-out workloads, which spend a high fraction of their time accessing memory and have moderate computational intensity. Two design paradigms have emerged in this space: one type features a few small cores on a small chip (small core, small chip); the other integrates many small cores on a bigger chip (small core, big chip).

Companies including Calxeda, Marvell, and SeaMicro market small-core, small-chip processors targeted at datacenters. Despite the differences in the core organization and even the instruction set—Calxeda's [16] and Marvell's designs are powered by ARM, whereas SeaMicro uses an x86-based Atom processor—the chips are surprisingly similar in their feature set: all have four hardware contexts, dual-issue cores, a clock speed in the range of 1.1 to 1.6 GHz, and power consumption of 5 to 10 W. A processor representative of the small- core, big-chip design philosophy is Tilera's Tile-Gx3036 [15]. This server-class processor features 36 simple cores and a 9-Mbyte LLC in a tiled organization. Each tile integrates a core, a slice of the shared LLC, and a router. Accesses to the distributed LLC's remote banks require a traversal of the on-chip interconnect, implemented as a 2D mesh network with a single-cycle per-hop delay. Operating at 1.5 GHz, the Tilera-like tiled design draws approximately 28 W of power at peak load.

With this in mind, we developed a methodology for designing performance-density-optimal server chips called Scale-Out Processors (SOPs). Our SOP methodology improves data-center efficiency through a many-core organization tuned to the demands of scale-out workloads.

To understand the efficiency implications of these diverse processor architectures, we use a combination of analytic models and simulation-based studies, employing a full-system server simulation infrastructure, to estimate their performance, area, and power characteristics. Our workloads are taken from CloudSuite (http://parsa.epfl.ch/cloudsuite), a collection of representative scale-out applications that includes Web Search, Data Serving, and MapReduce. Our results corroborate earlier studies that identify efficiency benefits stemming from the use of lower-complexity cores as compared to those used in conventional server processors. However, our findings also identify an important, yet unsurprising, trend: the use of simpler cores by themselves is insufficient for maximizing processor efficiency, and the chip-level organization must be considered. More specifically, a larger chip that integrates many cores is necessary to amortize the area and power expense of uncore resources, such as cache and off-chip interfaces, by multiplexing them among the cores.

To maximize silicon efficiency on scale-out workloads, we examined the characteristics of a suite of representative scale-out applications and the demands they place on processor resources. Our findings, consistent with prior work, indicate that: (i) large LLCs are not beneficial for capturing data-center applications' enormous data footprint; (ii) the active instruction footprint greatly exceeds the Level-1 (L1) caches' capacity, but can be accommodated with a 2- to 4-Mbyte secondary cache; and (iii) scale-out workloads have virtually no thread-to-thread communication, requiring minimal on-chip coherence and communication infrastructure.

Driven by these observations, we developed the SOP design methodology that extends the small-core, big-chip design space by optimizing the on-chip cache capacity, core count, interconnect delay, and number of interfaces to the off-chip memory in a way that maximizes computing density and throughput. At the heart of an SOP is a coarse-grained building block called a pod—a stand-alone multicore server. Each pod features a modestly sized 2- to 4-Mbyte LLC for capturing the active instruction footprint and commonly accessed data structures. The small LLC size reduces the cache access time and leaves more chip area for the cores. To further reduce the latency of performance- critical LLC accesses, SOPs use a high-bandwidth crossbar interconnect instead of a multi-hop point-to-point network. The number of

cores in a pod is empirically chosen in a way that maximizes cache utilization without causing thrashing or penalizing interconnect area and delay.

The SOP architecture achieves scalability through tiling at the pod granularity up to the available area, power, or memory bandwidth limit. The multiple pods share the off-chip interfaces to reduce cost and maximize bandwidth utilization. The pod-based tiling strategy reduces chip-level complexity and provides a technology-scalable architecture that preserves each pod's optimality across technology generations. Compared to a tiled design, a SOP increases the number of cores integrated on the former's greater on-chip computing capacity. However, as our results demonstrate, the SOP's greater chip-level processing capability is beneficial from a TCO perspective despite the increased power draw at the chip level.

V. CONCLUSIONS

In this paper, we presented the evolution of embedded processors from smartphones to datacenter servers. We introduced efficient 64-bit addressing and the ARM V8 architecture, the need for heterogeneous embedded computing and opportunities it provides. We further argued for the need for efficient, hardware-assisted virtualization in the embedded world, discussing its opportunities and challenges. Finally, we introduced a methodology for designing technology-scalable and efficient scale-out server processors that facilitates the design of optimal multi-core configurations, which divide the server processor's real-estate into performance-optimal modules that couple many lean cores with a small last-level cache to maximize throughput per area and minimize total cost of ownership at the datacenter level.

REFERENCES

[1] ARM, ARMv8 Architecture, [online] url: http://www.arm.com/products/processors/instruction-set-architectures/armv8-architecture.php

[2] J. Goodacre, "Matching cores to demands in always on mobile applications", [online] url: http://embedded-computing.com/articles/matching-cores-demands-always-mobile-applications/,Nov. 2012

[3] J. Goodacre, "Technology Preview: The ARMv8 Architecture", White Paper, Nov. 2011

[4] Peter Greenhalgh, ARM, "Big.LITTLE Processing with ARM Cortex™-A15 & Cortex-A7", Sep. 2011, available from: http://www.arm.com/files/downloads/big.LITTLE_Final.pdf.

[5] Plurality Ltd. The HyperCore Processor www.plurality.com/hypercore.html

[6] G.Kornaros, M.Grammatikakis, M.Coppola, "Towards Full Virtualization of Heterogeneous NoC- based Multicore Embedded Architectures", In Proc of 10th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, Dec. 2012

[7] S. Rixner, "Breaking the Performance Barrier: Shared I/O in virtualization platforms has come a long way, but performance concerns remain", ACM Queue –Virtualization, Jan/Feb 2008

[8] M Rosenblum and C Waldspurger, "I/O Virtualization", ACM Queue, Nov. 2011

[9] J. Smith and R. Nair, "Virtual machines: versatile platforms for systems and processes", Morgan Kaufmann Publishers Inc., 2005

[10] STMicroelectronics and CEA, "Platform 2012: A Many-core programmable accelerator for Ultra-Efficient Embedded Computing in Nanometer Technology", Nov. 2010, available from: www.calxeda.com/wp-content/uploads/2012/06/ECX1000-Product-Brief-612.pdf

[11] VIRTICAL: "SW/HW extensions for heterogenous multicore platforms", url: http://www.virtical.eu/

[12] P. M. Wells, K. Chakraborty, and G.S. Sohi, "Dynamic heterogeneity and the need for multicore virtualization", SIGOPS Oper. Syst. Rev. 43, 2, pp 5-14, 2009

[13] P. Willmann, S. Rixner, A. L. Cox, "Protection strategies for direct access to virtualized I/O devices", USENIX 2008 Annual Technical Conference on Annual Technical Conference, pp. 15—28, 2008

[14] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, , and B. Falsafi. "Clearing the clouds: a study of emerging scale-out workloads on modern hardware", In Proc. of the Int'l Conference on Architectural Support for Programming Languages and Operating Systems, Mar. 2012

[15] B. Wheeler. Tilera sees opening in clouds. Microprocessor Report, 25(7):13–16, July 2011.

[16] Calxeda, ''Calxeda EnergyCore ECX-1000 Series'', May 2012; www.calxeda.com/wp-content/uploads/2012/06/ECX1000-Product-Brief-612.pdf