Accelerator-Rich Architectures: Implications, Opportunities and Challenges

Ravi Iyer
Intel Corporation
E-mail: ravishankar.iyer@intel.com

Abstract - Providing high performance at ultra-low power for a domain of applications is possible by designing and integrating accelerators. Accelerators may be fixed-function, programmable or re-configurable in nature. Integration of many such accelerators in a system-on-chip (SoC) or chip-multiprocessor (CMP) introduces several major implications on architecture, power/performance and programmability. In this paper, we will provide an overview of the key challenges and outline research opportunities and challenges for accelerator-rich architectures and devices. We will also describe example solutions in some of these areas as a potential direction for further exploration.

I. Introduction

Over the last few decades, processor and platform architecture research has been focused on improving the performance of general-purpose applications by improving the capability of the general-purpose microprocessor. In the last decade, the power wall had a tremendous influence on architects, shifting their focus from higher frequency processors to multi-core processors in order to provide high performance and scalability in the general-purpose domain. With the ever-increasing focus on smaller form factors and power-constrained, battery-operated devices, the implication of the power wall continues to pose some significant challenges to the future direction for architectural solutions. As a result, many researchers are exploring the integration of accelerators on the die to provide a significant benefit in performance/power for a targeted domain of applications. A number of accelerators are already being integrated on the die for special purpose processing domains such as graphics, media, audio, security and imaging and in system-on-chip solutions [1, 2]. There are also a number of emerging application domains that demand high performance, real-time, power-efficient execution which are likely to lead to accelerator design and integration. As a result, accelerator-rich architectures are becoming mainstream and it is important to re-examine our traditional architectural research and determine the implications of accelerator integration and understand the challenges and opportunities as a result.

In this paper, the goal is to discuss the implications of accelerator-rich architectures and identify the key challenges and opportunities. We will start by describing the different types of accelerators, and then walk through the key differences between traditional CMP architectures and accelerator-rich SoC architectures. By doing so, we discuss the challenges and opportunities that accelerator integration introduces and we discuss some examples of on-going work that highlights important open questions and indicates fruitful areas of future research in this area.

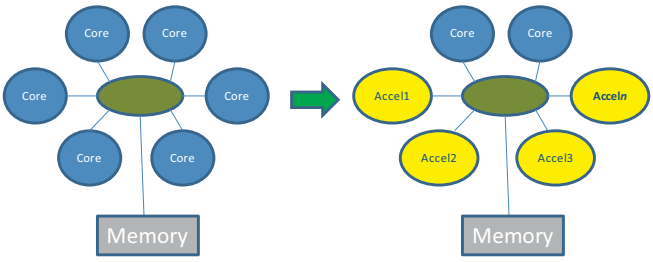

Figure 1. Accelerator-Rich Architectures

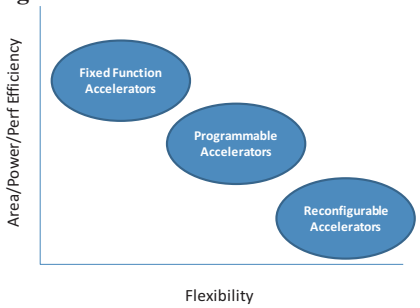

Figure 2. Accelerators: Efficiency vs Flexibility

II. Accelerator-Rich Architectures: Background

Figure 1 illustrates accelerator-rich architectures and compares it to homogeneous multi-core architectures. Accelerators are essentially designed to provide high performance with constrained power limits for domain-specific applications. Commonly known domains for acceleration includes graphics, security, image processing, etc.

When designing accelerators, two key questions immediately need to be answered. The first one is the domain for which the accelerator is being customized and designed. Emerging domains of accelerators include recognition, 3D processing, and analytics/machine learning [3, 4]. The second question is what type of accelerator makes sense to design: *fixed-function, programmable or (re)configurable.* Figure 2 illustrates the efficiency vs. flexibility considerations when designing accelerators. For application domains that are highly matured or short-lifetime products which require a specific feature, fixed function accelerators can be designed and will likely provide the best performance/power/area efficiency with very little flexibility for modifications after the product has been designed. For applications domains that are maturing or use protocols that constantly change, it is important to design programmable accelerators (at times referred to as application specific processors) that accommodate sufficient flexibility while offering a good performance/power and area advantage over general-purpose processors. For application domains that are quite wide and areas where custom logic needs to be incorporated after the product is designed, enabling

978-1-4673-0772-7/12/$31.00 ©2012 IEEE

reconfigurable accelerators may be valuable. It should be kept in mind that in general, reconfigurable accelerators provide great flexibility but at a significant cost of power/area/performance of a target application.

Another important point to keep in mind that exploration and design of these accelerators can be quite labor-intensive and costly if each accelerator is developed independently with a ASIC-style approach. Developing a methodology and a set of consistent tools where a accelerator template is used for quick exploration and design of accelerators is crucial. Typically most accelerators are based on an accelerator core or engine that provides control-plane functionality and programmable performance. Developing a configurable and extensible accelerator engine that is has a consistent base and a toolchain support it can help kickstart this approach significantly.

III. Accelerator-Rich Architectures: Implications

When compared to traditional homogeneous architectures, an accelerator-rich architecture brings with it a whole list of challenges and opportunities. Here we describe the most significant challenges and implications of accelerator-rich architectures and compare those to traditional multi-core architectures for contrast.

(a) Architectural Implications: Traditionally, architectures have been evaluated and optimized from a multi-core standpoint. With accelerators integrated, there are a number of differences between the behavior and requirements for accelerators from general-purpose cores that need to be considered. For example, memory accesses from cores are optimized for latency tolerance by introducing SRAM as cache. Furthermore, multiple cores shares the last-level of cache as a means to sharing data efficiently and minimizing resource usage. Accelerators traditionally employ SRAM as buffers and tend to use streaming models such as producer consumer flows as a means for sharing input/output data. When designing a accelerator-rich architecture, it is important to keep such behavioral differences in mind when organizing and architecting SRAM structures. An example of re-definition of SRAM for accelerators and cores can be found in [5]. Another topic of common interest to cores and accelerators is quality of service (QoS) [6]. QoS requirements for applications running on cores are typically very different from those for accelerators. Cores can sustain gradual performance degradation, but typically accelerators have coarse-grain bandwidth/latency requirements that need to be met for performance (more of a step-function than a gradual loss in performance). Such requirements need to be considered when defining the baseline architecture.

(b) Accelerator Integration and Management: A significant difference between today's accelerators and cores are also in the operating domain. Cores operate in the virtual memory domain and take advantage of memory management services to translate virtual memory addresses to physical memory addresses. Also, cores typically operate in the coherent memory domain to take advantage of private/shared caches. However, accelerators are treated as non-coherent agents (devices) in the platform architecture and traditional integrated as a peripheral I/O agent. However, it is important to consider accelerators as first class citizens in the platform with similar support for virtual memory operations as well as coherence/synchronization. A few papers have proposed this capability [8] in the past and future accelerator architectures need to find a modular and scalable architecture to achieve these capabilities. Another aspect of accelerators is the management of access to accelerators from multiple simultaneously active applications. While device drivers traditionally serve this function, it is important to consider more efficient mechanisms to enable management of accelerators for access and coupling.

(c) Accelerator Programming and APIs: The general-purpose programming model for our processor cores has served well because it makes it easy for the large mass of application developers to write programs quickly for these architectures. However, as accelerators are introduced for domain-specific applications, it becomes important to develop APIs for application developers to use that allow for modular development and portability. The ability to write a single application efficiently while retaining the capability of running it on different platforms with different levels of support for acceleration becomes critical. Several efforts (OpenCL [9] for instance) are attempting to achieve this by providing software abstractions for sending work to accelerators and for managing accelerators. Such efforts are in their early stages (mostly applicable to GPUs for now), and more advanced research for generic accelerators is required for this to quickly gain maturity for accelerator-rich architectures.

IV. Summary and Conclusions

This paper discusses accelerator-rich architectures and provides an overview of the key considerations. Accelerator-rich architectures are becoming more common because of the power wall and battery life constraints of small form factor devices. In the paper, we summarize the implications of accelerator-rich architectures and outlined key areas for further research on some key challenges and opportunities. The key areas for research are centered on accelerator design methodology, architecture implications of accelerators, integration and interfacing of accelerators and programming models and APIs for accelerator-rich architectures.

References

[1] Intel Moorestown (Next generation handheld platform), http://www.intel.com/pressroom/archive/reference/moorestown_backgrounder.pdf

[2] TI OMAP™ 5 Platform (OMAP 5430), http://www.ti.com/ww/en/omap/omap5/omap5-OMAP5430.html

[3] S. Lee, Z. Fang, et al , "Accelerating Mobile Augmented Reality on a Handheld Platform, 27th International Conference on Computer Design (ICCD), 2009

[4] S. Chan, "Challenges in Processing 3D videos," http://videoprocessing.ucsd.edu/~stanleychan/publication/presentations/2011_161B.pdf

[5] C. Fajardo, Z, Fang, R. Iyer, et al, "Buffer-in-cache: A Cost-Effective SRAM Architecture for Handheld and Embedded Platforms," published in DAC 2011

[6] R. Iyer, "CQoS: A Framework for Enabling QoS in Shared Caches of CMP Platforms," ICS 2004.

[7] R. Iyer, "CQoS: A Framework for Enabling QoS in Shared Caches of CMP Platforms," ICS 2004.

[8] P. Stillwell, et al, "HiPPAI: High Performance Portable Accelerator Interface for SoCs, HiPC'2009

[9] OpenCL specification, http://www.khronos.org/registry/cl/specs/opencl-1.0.29.pdf