Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Accurately modeling the on-chip and off-chip GPU memory subsystem

Francisco Candel *, Salvador Petit, Julio Sahuquillo, José Duato

Department of Computer Engineering, Universitat Politècnica de València, 46012 Valencia, Spain

HIGHLIGHTS

- This paper focuses on accurately modeling the entire GPU memory subsystem, both on-chip and off-chip.
- We accurately modeled critical memory components in the state-of-the-art Multi2Sim heterogeneous CPU–GPU processor simulator.
- Experimental results show that not accurately modeling these components can raise the evaluated execution time up to 3 times.

ARTICLE INFO

Article history:
Received 3 March 2016
Received in revised form
20 January 2017
Accepted 7 February 2017
Available online xxxx

Keywords:
Applied modeling and simulation
On-chip memory subsystem
Main memory controller
GDDR
Cache coherence protocol

ABSTRACT

Research on GPU architecture is becoming pervasive in both the academia and the industry because these architectures offer much more performance per watt than typical CPU architectures. This is the main reason why massive deployment of GPU multiprocessors is considered one of the most feasible solutions to attain exascale computing capabilities.

The memory hierarchy of the GPU is a critical research topic, since its design goals widely differ from those of conventional CPU memory hierarchies. Researchers typically use detailed microarchitectural simulators to explore novel designs to better support GPGPU computing as well as to improve the performance of GPU and CPU–GPU systems. In this context, the memory hierarchy is a critical and continuously evolving subsystem.

Unfortunately, the fast evolution of current memory subsystems deteriorates the accuracy of existing state-of-the-art simulators. This paper focuses on accurately modeling the entire (both on-chip and off-chip) GPU memory subsystem. For this purpose, we identify four main memory related components that impact on the overall performance accuracy. Three of them belong to the on-chip memory hierarchy: (i) memory request coalescing mechanisms, (ii) miss status holding registers, and (iii) cache coherence protocol; while the fourth component refers to the memory controller and GDDR memory working activity.

To evaluate and quantify our claims, we accurately modeled the aforementioned memory components in an extended version of the state-of-the-art Multi2Sim heterogeneous CPU–GPU processor simulator. Experimental results show important deviations, which can vary the final system performance provided by the simulation framework up to a factor of three. The proposed GPU model has been compared and validated against the original framework and the results from a real AMD Southern-Islands 7870HD GPU.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

In the recent years there has been an steady increase in the use of GPUs (Graphics Processing Units) for general purpose computing. The main reason is that general purpose computing in GPUs or simply GPGPU computing is much more energy-efficient [1] than conventional computing. In other words, for a given power budget, GPGPUs provide higher performance than their CPUs counterparts, especially when running massively parallel workloads. Because of this fact, most of the top 10 supercomputers in the top 500 list [2] rely on GPUs. For instance, the Titan supercomputer, ranged in second place of the list in November 2014, was built with Nvidia K20x devices. However, GPU programmability [3] is still harder than that of conventional computing. To deal with this shortcoming, computer architects are trying to adapt different components and mechanisms (e.g. caches and prefetching) that have successfully worked on CPUs to ease programmability.

* Corresponding author.
E-mail address: fracanma@inf.upv.es (F. Candel).

http://dx.doi.org/10.1016/j.future.2017.02.012
0167-739X/© 2017 Elsevier B.V. All rights reserved.

ARTICLE IN PRESS

The GPU architecture has been traditionally optimized to run graphic applications workloads, composed of thousands of logical threads, and that exhibit a massive parallelism. For this purpose, the GPU cores present a high computational power which come from including hundreds of processing elements, all of them working together.

In order to feed such a high number of computational elements, the GPU core must be coupled with an efficient memory subsystem. Due to this reason, GPU memory subsystems are designed to tolerate a high number of concurrent accesses.

The importance of easing the programmability of GPUs for GPGPU computing as well as the irruption in the market of *heterogeneous* computing processors [4] that combine CPUs and GPUs on the same die, open a new design space for memory hierarchy designs, which is a hot topic in computer architecture research. To implement and evaluate their approaches, academic and industry researchers need from complex and detailed simulation frameworks. These software packages are abstractions that model the functionality of real hardware and focus on those hardware components that have a significant impact on the final system performance. However, because of the fast speed at which current systems evolve, state-of-the-art simulators often miss modeling important components and, consequently, simulation results are not as accurate as they should.

This paper focuses on the memory subsystem, both on-chip and off-chip, of contemporary GPUs. We find that four main important components, which present a significant contribution to the system performance, are not precisely modeled in state-of-the-art GPU simulators with respect to a real device. In particular, three of them correspond to the on-chip memory hierarchy: (i) memory request coalescing mechanisms, (ii) miss status holding registers, and (iii) the cache coherence protocol; while the fourth component refers to the memory controller and the off-chip GDDR memory.

To quantify the impact on performance of these components, we enhance the modeling of the GPU memory subsystem in a state-of-the-art GPU simulator, we quantify the impact of each component on the system performance, and we validate all the components working together by comparing the results of the proposal to the execution time on a AMD Southern-Islands 7870HD GPU. For this purpose, we used the Multi2Sim simulation framework [5], widely used in both the academia and the industry. Experimental results show that each of the studied components, if not accurately modeled, can result in important (e.g. in a factor of $2\times$ or $3\times$) performance deviations in the simulated results.

The remainder of this work is organized as follows. Section 2 presents a relevant subset of current GPU simulators. Section 3 describes the Southern Islands architecture and its programming model. In Section 4, the proposed Multi2Sim extensions are described in detail. Section 5 presents the experimental results. Section 6 provides the accuracy improvements achieved by the proposed extensions. Finally, in Section 7 some concluding remarks are drawn.

2. Related work

GPU research simulators are relatively young and still maturating. In fact, the number of available GPU simulation frameworks is nowadays much lower than that of CPU simulators. The main reasons of this lack of tools is that GPU manufacturers provide little information about the architecture of their processors as well as the fact that the architecture of modern GPUs has been and is quickly evolving, hampering the development of detailed architectural simulators which require an established and well-known model. In spite of this fact, due to the growing use of GPUs, some GPU simulation frameworks have become recently available. Below, we describe a representative set of them.

GPGPU-Sim [6,7] is currently one of the most referenced GPU simulators. It is a detailed cycle by cycle simulator that supports CUDA version 3.1. It models a GPU microarchitecture similar to the Nvidia GeForce 8x, 9x, and Fermi series. GPGPU-Sim also simulates the interconnection network between GPU cores and memory modules.

Recently, the Gem5 [8] computer system simulator platform was combined with GPGPU-Sim to model a heterogenous CPU–GPU system. Moreover, GPGPU-Sim version 3.2.0 integrates GPUWattch [9], an energy model based on McPAT [10]; a power, area, and timing modeling framework. However, due to its dependence on Nvidia drivers, which only support OpenCL 1.1, GPGPU-Sim does not provide support for the execution of GPGPU benchmark suites like that provided by AMD [11] with modern OpenCL code.

Barra [12] is a parallel GPU functional simulator. It is based in the UNISIM framework [13] and it implements both a CUDA driver emulator and an Nvidia Tesla GPU simulator. In this way, Barra can execute directly unmodified CUDA programs and generate statistics at the instruction level. The major shortcoming of this simulator is that it does not model the GPU microarchitecture, thus it cannot be used to evaluate possible enhancements in the memory subsystem. In addition, this framework only supports a rather old CUDA version 2.2.

Multi2Sim [14,5] is an accurate cycle by cycle execution driven simulation framework for CPU–GPU heterogeneous computing. Release and development versions of Multi2Sim are available. It provides a fully configurable memory subsystem with several cache levels and interconnection networks. Multi2Sim implements several GPU architectures from both AMD (Evergreen, Southern Islands) and Nvidia (Fermi) as well as CPU architectures like x86, MIPS-32 and ARM. The Multi2Sim developer team is currently modeling the HSA heterogeneous architecture [15], where both CPU and GPU share the same memory subsystem. Finally, Multi2Sim includes its own implementation of OpenCL and CUDA libraries. In this way, it can provide dynamic information about CPU–GPU interaction by instrumenting OpenCL and CUDA calls.

In summary, we chose Multi2Sim because (i) it simulates a heterogeneous CPU–GPU cycle by cycle, (ii) it implements the recent AMD GPU core architectures called GCN [16], (iii) it includes its own OpenCL and CUDA libraries, and (iv) support for the HSA architecture is being developed.

3. Southern Islands GPU programming model and architecture

This section provides some background on how contemporary GPUs work. To this end, we focus on the state-of-the-art *Southern Islands* GPU from AMD introduced in 2012 which, to the best of our knowledge, is the most recent GPU architecture implemented on a detailed simulator framework. To understand this system, two main axis must be considered: (i) its programming model, and (ii) its architecture, which consists of multiple cores sharing the same memory hierarchy. Below, both axis are discussed.

3.1. The OpenCL programming model

Two main programming frameworks, CUDA [17] from Nvidia and OpenCL [18] from the Khronos group, are currently being used for developing programs targeting GPGPUs and other kinds of computing devices. OpenCL is, "*de facto*", an industry standard programming model [19]. There are OpenCL implementations that work on devices from different brands such as Intel, AMD, ARM, or even Nvidia, while CUDA is only supported in GPUs manufactured by Nvidia.

The OpenCL specification [20] defines a platform model and an execution model. The platform model is an abstraction of the real

Download English Version:

https://daneshyari.com/en/article/6873225

Download Persian Version:

https://daneshyari.com/article/6873225

Daneshyari.com