INTEGRATION, the VLSI journal ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

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

Efficient modeling and analysis of energy consumption for 3D graphics rendering

Lidong Xing [a,*], Tao Li [b], Hucai Huang [b], Qingsheng Zhang [b], Jungang Han [b]

[a] School of Microelectronic, Xidian University, Xi'an 710071, China
[b] Xi'an University of Posts and Telecommunications, Xi'an 710121, China

ARTICLE INFO

Keywords:
Multiprocessor
GPU (Graphics Processing Unit)
Performance optimization
Energy estimation models
Parallelism

ABSTRACT

This paper proposes new models of GPU energy consumption from the perspectives of hardware architects and graphics programmers by performing an architecture-independent analysis of the classical graphics rendering pipeline which is still in widespread use today. The detailed analysis includes graphics rendering workload, memory bandwidth and energy consumption . Although the models are derived from classical 3D pipeline, they are extensible to programmable pipelines. There are many factors that affect the performance and energy consumption of 3D graphics rendering, such as the number of textures, vertex sharing, level of details, and rendering algorithms. The proposed models are validated by our simulation study and used to guide our 3D graphics hardware design and 3D graphics programming in order to optimize performance and energy consumption of our GPU prototypes which have been successfully fabricated in SMIC 0.13 μm CMOS technology.

1. Introduction

VLSI technology has entered the era of billions of transistors on a single chip. Integration density brought forth the fast development of parallel processing engines such as many core CPUs and massively parallel GPUs. These parallel engines also brought with them tremendous energy consumption. Since GPUs are widely used in mobile devices, desk top and laptop computers, and high performance computers, GPU energy optimization is of paramount importance to industry.

This paper presents a detailed energy model of 3D graphics rendering. The results are architecture independent and are applicable to most type of architectures. Different from previous modeling effort, our work builds a model that can help (1) GPU architects in optimizing their design, and (2) graphics programmers in optimizing their programs.

Modern GPUs [1–3] employ separate shader architectures [4] as well as unified shader architectures [5,6]. Memory bandwidth constraints may limit the performance of unified shader GPUs and software implementations may affect the load balance of separate shader GPUs.

We present models of GPU energy consumption from the perspectives of hardware architects and graphics programmers. Detailed analysis of energy estimate in every stage of a classical 3D rendering pipeline is given here. Different from the previous research, we emphasize that the analysis makes no assumptions about the hardware architecture, so that the results are widely applicable. Our research employs forward analysis and builds detailed models from known graphics rendering algorithms. We should point out that the focus of this paper is on 3D graphics rendering, rather than general purpose GPU computing (GPGPU). The results presented here might not be directly applicable to GPGPU.

Previous research on GPU performance and power analysis was mainly found in these areas, (1) characterization of GPU workloads, (2) methods and results of measuring existing GPUs and model building based on measurement results, and (3) GPU performance and power analysis from the perspective of software and application. These results come from GPU users rather than GPU designers. Since an exhaustive description of all the previous research is difficult, we only review the representative efforts in GPU energy estimation.

Workload characterization studies either the static aspect or the dynamic aspect of 3D graphics and typically employs either simulation study, or experimental measurement, or a combination of both, plus mathematical modeling. Characterizations on 3D static workload and dynamic workload are found in [7,8]. The work was done on such characteristics as memory access bandwidth, the number of vertices per primitive, and triangles processed per frame. The research uses a combination of simulation work and experimental measurement, using well-known benchmarks. A characterization on the workload of 3D games is found in

* Corresponding author.
E-mail address: zmy_xld@163.com (L. Xing).

http://dx.doi.org/10.1016/j.vlsi.2016.02.009
0167-9260/

[9], which characterizes GPU games, in such attributes as average vertex and pixel shader instructions, utilization different types of primitives, bus bandwidths, and triangle sizes (at different stages). A signature-based estimation technique for predicting 3D graphics workloads is presented in [10]. They showed that monitoring specific parameters of the 3D pipeline provides better prediction accuracy over conventional approaches. Signature-based prediction is computationally efficient. A fundamental difference between signatures and standard history-based predictors is that signatures capture previous outcomes as well as the cause that led to the outcome, and use both to predict future outcomes. A study [11] using the Quake 3 and XRace games as benchmarks on three mainstream mobile system-on-chip architectures reveals that the geometry stage is the main bottleneck in 3D mobile games and confirms that game logic significantly affects energy consumption.

A straightforward method for measuring per-frame energy consumption of real-time graphics workloads is discussed in [12]. The method is non-invasive, and needs no source code. It is possible to measure a much wider range of applications. Reference [13] gives an in-depth quantitative analysis of the power consumption of mobile 3D graphics pipelines, on the effects of various 3D graphics factors such as resolution, frame rate, level of detail, lighting and texture maps on power consumption. Power estimate models are built based on the analysis. Statistical models are constructed based on simulation study and experimental measurements in [14,15]. The models are constructed based on performance counters and power measurements for GPGPU applications. The SigGraph 2011 course [16] discusses the general principles of GPU power consumption and gives some tips about power optimization. References [17,18] describe GPU performance and power models derived from rendering pipeline stages and measurements of existing GPUs. An energy model for a graphics processing unit (GPU) is given in [19]. The model is based on the amount and type of work performed in various parts of the unit. It is able to accurately predict the energy that any arbitrary mix of operations will take.

In [20], GPU power consumption is investigated from a software perspective and GPGPU applications. It studies how and where the power consumption is located within a GPU board by analyzing the relations between the measured power consumption, the required time and the type of units. The considered blocks include register file, memory hierarchy and functional units. The effectiveness of GPUs for a variety of application types is discussed in [21] along with some specific coding idioms that improve program performance on the GPU. It also discusses advantages and inefficiencies of the CUDA programming model and some desirable features that might allow for greater ease of use. In [22], an important biological code that runs in a traditional CPU environment is taken in the study and is transformed to a hybrid CPU+GPU environment, which delivers an energy-delay product that is multiple orders of magnitude better than a traditional CPU environment. The research investigates via an empirical study the performance, power, and energy characteristics of GPUs for scientific computing. Low power programming tips, such as compressing textures and reducing memory bandwidth, follow drawing order (front to back), use low precision arithmetic, etc, are offered in [23]. Wimmer and Wonka [24] study different factors that contribute to the rendering time in order to develop a framework for rendering time estimation. Given a viewpoint (or view cell) and a list of potentially visible objects, they propose several algorithms that can give reasonable upper limits for the rendering time on consumer hardware.

2. The 3D graphics pipeline

Graphics rendering is to generate a 2D image frame from a 3D scene description. The classical graphics processing pipeline includes several major stages as will be presented in Section 2.1. Details of graphics primitives will be described in 2.2.

2.1. The rendering pipeline

A typical 3D graphics rendering pipeline is shown in Fig. 1. The pipeline consists of many computation tasks, including command processing by FEP (front-end processor), model-view transform of vertices and vertex normals, unitization of vertex normals, vertex shading and clamping of shading results, primitive (shape) assembly, plane clipping, projection transform, frustum clipping and division by w, back-face culling, rasterization (scan-conversion), pixel shading and fogging, and fragment operations such as tests and logic operations.

Most stages of the 3D graphics rendering pipeline perform fixed functions. In addition to the fixed function states, the rendering pipeline may also have a number of programmable stages, typically including a vertex shader stage and a pixel shader stage.

Graphics rendering is one of those tasks which exhibit the so-called “embarrassing parallelism”. Many stages in the 3D rendering pipeline can be very well parallelized, for example, the vertex shading stage and the pixel shading stage, and the parallelization of these stages can be straightforward and without much overhead. Linear speed ups are obtainable. Performance and power consumption can be extended from a single shader case to n shader case by applying a multiplicative coefficient. However,

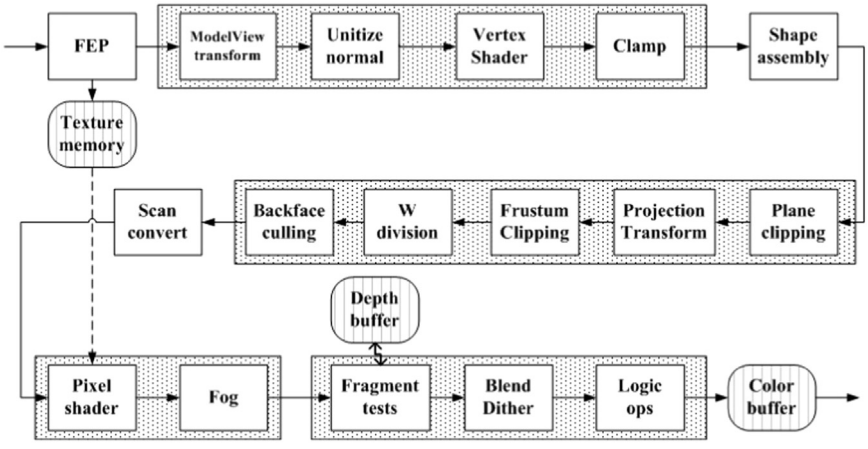

Fig. 1. The generic 3D graphics rendering pipeline.

Please cite this article as: L. Xing, et al., Efficient modeling and analysis of energy consumption for 3D graphics rendering, INTEGRATION, the VLSI journal (2016), http://dx.doi.org/10.1016/j.vlsi.2016.02.009