

POWER OPTIMIZATIONS FOR GRAPHICS PROCESSORS

B.V.N.SILPA



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI**

March 2011

POWER OPTIMAZTIONS FOR GRAPHICS PROCESSORS

by

B.V.N.SILPA

Department of Computer Science and Engineering

Submitted

in fulfillment of the requirements of the degree of Doctor of Philosophy

to the



Indian Institute of Technology Delhi
March 2011

Certificate

This is to certify that the thesis titled **Power optimizations for graphics processors** being submitted by **B V N Silpa** for the award of **Doctor of Philosophy in Computer Science & Engg.** is a record of bona fide work carried out by her under my guidance and supervision at the **Department of Computer Science & Engineering, Indian Institute of Technology Delhi**. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Preeti Ranjan Panda

Professor

Dept. of Computer Science & Engg.

Indian Institute of Technology Delhi

Acknowledgment

It is with immense gratitude that I acknowledge the support and help of my Professor Preeti Ranjan Panda in guiding me through this thesis. I would like to thank Professors M. Balakrishnan, Anshul Kumar, G.S. Visweswaran and Kolin Paul for their valuable feedback, suggestions and help in all respects. I am indebted to my dear friend G Krishnaiah for being my constant support and an impartial critique. I would like to thank Neeraj Goel, Anant Vishnoi and Aryabartta Sahu for their technical and moral support. I would also like to thank the staff of Philips, FPGA, Intel laboratories and IIT Delhi for their help.

I owe my deepest gratitude to Microsoft Research India, for supporting my research by granting me MSR fellowship. I would also want to thank Intel India Pvt. Limited for funding my research. I would like to specially thank Kumar S S Vemuri for the mentorship I received from him.

This thesis is dedicated to my family members who have shown immense patience and provided me with great support during the course of my work.

B V N Silpa

ABSTRACT

Advances in Computer Graphics have led to creation of sophisticated scenes with realistic characters and fascinating effects. As a consequence, the computational complexity of graphics applications has also increased tremendously. With increasing interest in sophisticated graphics capabilities in mobile systems, energy consumption of graphics hardware is becoming a major design concern in addition to the traditional performance enhancement criteria. This motivates us to focus on designing low power graphics processors for mobile devices. We present the first comprehensive power optimization work targetting the computer graphics rendering pipeline. The power minimization is targetted at different levels of abstraction: component level, compiler level, and full system level. The main contributions of this thesis are the following:

- A custom memory architecture for low power texture memory sub-system.
- A code optimization technique that reduces the computational complexity and hence the power consumption of the geometry engine.
- System level power optimization by Dynamic Voltage and Frequency Scaling for tiled graphics processors.

Among the different steps in the graphics processing pipeline, we observe that memory accesses during texture mapping – a highly memory intensive phase – contributes 30-40% of the energy consumed in typical embedded graphics processors. This makes the texture mapping subsystem an attractive candidate for energy optimization. We argue that a standard cache hierarchy, commonly used by researchers and commercial graphics processors for texture mapping, is wasteful of energy, and propose the *Texture Filter Memory*, an energy efficient architecture that exploits locality and the relatively high degree of predictability in texture memory access patterns. Our architecture consumes 75% lesser energy for texturing in a fixed function pipeline and about 85% lesser energy in a parallel rasterization hardware. It also achieves 7% more hits than a partitioned cache generally used for multitexturing. Interestingly our proposed architecture also achieves higher performance than conventional texture mapping hardware. We also demonstrate that introduction of these filter buffers help greatly in reducing the leakage power consumption of the texture memory sub-system. Our proposed drowsy texture L1

with predictive wake-up helps in achieving 80% leakage power savings at the cost of less than 1% performance loss.

We have observed that the geometry engine also contributes significantly towards the total power consumption in modern games. This is because the creation of scenes with increasing levels of detail is resulting in escalating the amount of geometry per frame, making the performance of the geometry engine one of the computationally intensive stages of the pipeline. In this thesis we propose a mechanism to reduce the amount of computation in the geometry engine, thereby reducing the power consumption of the geometry engine and at the same time speeding up the geometry processing. This is achieved by partitioning the vertex shader into position-variant and position-invariant parts and executing the position-invariant part of the shader only on those triangles that pass the trivial reject test. Our main contributions here are : (i) a partitioning algorithm that attempts to minimize the duplication of code between the two partitions of the shader and (ii) an adaptive mechanism to enable the vertex shader partitioning so as to minimize the overhead incurred due to thread-setup of the second stage of the shader. From the results we observe a saving of upto 50% of vertex shader instructions and hence a speed-up of upto 15%. Due to significant savings on number of vertex shader instructions, we can expect attractive saving on power consumed by the geometry engine.

From the study of various modern games we observe that the workload varies significantly with time and hence can benefit from dynamic voltage and frequency scaling (DVFS) which saves the system level power consumption of the GPU. Since visual quality of graphics applications is highly dependent on the rate at which frames are processed, it is important to devise a DVFS scheme that minimizes deadline misses due to inaccuracies in workload prediction. We demonstrate that tiled-graphics renderers exhibit substantial advantages over immediate-mode renderers in obtaining access to frame parameters that help in enhancing the workload estimation accuracy. We also show that, operating at a finer granularity of “tiles” as opposed to “frames” allows early detection and corrective action in case of a mis-prediction. We propose an accurate workload estimation technique and two DVFS schemes namely (i) tile-history based DVFS and (ii) tile-rank based DVFS for tiled-rendering architectures. The proposed schemes are demonstrated to be more efficient in terms of power and performance than the frame level DVFS schemes proposed in recent literature. With a system with 8 DVFS levels, our tile-history based

DVFS scheme results in 60% improvement in quality (deadline misses) over the frame history based DVFS schemes and gives 58% saving in energy. The more sophisticated tile-rank based scheme achieves 75% improvement in quality over the frame history based DVFS scheme and results in 58% saving in energy. We have also compared the efficiency of the proposed tile-level DVFS schemes with frame-level schemes with increasing number of DVFS levels, and found that while the frame-level schemes suffer from increasing deadline misses as the frequency levels increase, the impact on our tile-level schemes is negligible. The energy per frame-rate for our scheme is the minimum, indicating that it delivers the best performance-energy results.

Contents

List of Figures	vii
List of Tables	xi
1 Introduction	1
1.1 Introduction to Graphics Processing	4
1.1.1 Application Stage	4
1.1.2 Geometry	9
1.1.3 Triangle Setup	11
1.1.4 Rasterization	12
1.1.5 Display	15
1.2 Graphics Processor Architecture	17
1.2.1 Immediate Mode Rendering Engines	18
1.2.2 Tiled Graphics Engines	23
1.3 Power Dissipation in a Graphics Processor	26
1.4 Our Contribution	28
1.5 Thesis Outline	29
2 Literature Survey	31
2.1 Programmable Units	31
2.1.1 Clock Gating	31
2.1.2 Fixed Function ALUs	33
2.1.3 Predictive Shutdown	33
2.2 Texture Unit	34
2.2.1 Low power cache configurations	34

2.2.2	Texture Compression	35
2.2.3	Clock Gating	37
2.3	Frame Buffer	37
2.3.1	Depth Buffer Compression	38
2.3.2	Color Buffer Compression	40
2.4	System Level Power Management	41
2.4.1	Power Modes	41
2.4.2	Dynamic Voltage and Frequency Scaling	41
2.4.3	Multiple Power Domains	47
2.5	Miscellaneous	48
3	Texture Filter Memory	49
3.1	Introduction	49
3.2	Texture Mapping Access Pattern	52
3.3	Architecture of Texture Filter Memory	56
3.3.1	Texture Buffer Array	57
3.3.2	Address Comparators	59
3.3.3	Controller	59
3.4	Static Power Reduction due to Texture Filter Memory	60
3.4.1	Predictive wake-up	61
3.5	Extension to other architectures and filters	64
3.5.1	Anisotropic Filtering	64
3.5.2	Parallel Texturing	65
3.5.3	Multi Texturing	66
3.6	Experiments and Results	67
3.6.1	Evaluation of the TFM architecture	67
3.6.2	Leakage Power and Delay Comparision	71
3.6.3	Parallel rasterization architecture	72
3.6.4	Multitexturing	73
3.7	Summary	74

4	Vertex Shader Partitioning	77
4.1	Introduction	77
4.2	Shader Compiler	82
4.2.1	Partitioning with Minimum Duplication	82
4.2.2	Comparison with the naïve algorithm	87
4.2.3	Selective Partitioning of Vertex Shader	88
4.3	Framework	89
4.3.1	Partitioning the assembly code	91
4.3.2	Vertex Input Buffer	92
4.3.3	Feeder Unit	92
4.4	Experiments and Results	93
4.4.1	Performance Improvement	93
4.4.2	Energy Reduction	95
4.5	Summary	96
5	DVFS for Tiled GPUs	99
5.1	Introduction	99
5.2	Workload Estimation of a Tiled Graphics Processor	104
5.2.1	Frame Rank Computation	107
5.2.2	Tile Rank Computation	110
5.2.3	Extraction of Ranks	111
5.3	DVFS Scheme	111
5.3.1	Tile History Based DVFS	112
5.3.2	Tile Rank Based DVFS	115
5.4	Experiments and Results	117
5.4.1	Performance Impact	118
5.4.2	Energy Saving	120
5.5	Summary	122
6	Conclusion and Future Work	125
6.1	Summary and conclusion	125
6.2	Future Work	127

Bibliography

129
