A Heterogeneous System Architecture Conformed GPU Platform Supporting OpenCL and OpenGL

Yung Hsu
Dept. of Elec. Eng.,
National Cheng Kung University
csh01279@gmail.com

Heng-Yi Chen
Dept. of Elec. Eng.,
National Cheng Kung University
chen9611@gmail.com

Chung-Ho Chen
Dept. of Elec. Eng.,
National Cheng Kung University
chchen@mail.ncku.edu.tw

***Abstract*—Graphic Processing Unit (GPU) has powerful parallel computing ability, so it is not only used for 3D graphic application, but also for general purpose tasks. This work presents a Heterogeneous System Architecture (HSA) conformed GPU platform that supports both rendering and general computing applications. The platform can explore microarchitecture design and evaluate the performance issues for both the OpenCL and OpenGL applications.**

I. Introduction

General Purpose Computing on Graphics Processing Units (GPGPUs) are considered as a promising way to deal with heavy parallel computing tasks. Furthermore, Advanced Micro Devices, Inc. (AMD) and other founders funded the Heterogeneous System Architecture (HSA) Foundation, which aims to present an architecture of integrated heterogeneous devices, like CPU, DSP, and GPU, and make them collaborate more effectively.

In this work, we present a virtual platform conforming to the HSA programing model and the HSA Intermediate Language (HSAIL) specification. This platform has a sophisticated simulator modeling the modern GPU microarchitecture designed for Single Instruction Multiple Data (SIMD) processing. The platform also provides a simulation framework, including OpenCL and OpenGL API, the driver for simulator, and compilation flow from OpenCL kernel and OpenGL shader program to HSAIL and finally to a custom instruction set.

II. GPU Simulator Architecture

The proposed platform models the GPU microarchitecture which is composed of four major parts, task dispatcher, streaming multiprocessor (SM), fixed-function rendering units, and memory system containing DRAM and cache model.

The task dispatcher divides the whole task to several subtasks, called workgroups. Then the task dispatcher will assign these workgroups to a proper SM.

An SM which consists of various number of execution units is the main component in charge of programmable computing. Once an SM receives a workgroup, a scheduler within the SM separates the workgroup to smaller parts, called wavefront, and schedules these wavefronts to numerous execution units in the SM. The execution unit is a scalar processing core with customized instruction set based on the HSAIL binary format (BRIG). Each SM also has some special functional units. For instance, texture units are used as coordinate computing and texture filtering. Branch units handle the SIMD control divergence issue.

Fixed-function rendering units are a series of dedicated modules for graphic pipeline, such as primitive assembly, triangle setup, clipping, culling, rasterization, z-test, etc.

III. Simulation Framework

One benefit of HSAIL is to provide an efficient intermediate language for different parallel programing models to run on a variety of target machines. First, a parallel program is compiled into HSAIL assembly, and then a tool called finalizer converts the HSAIL assembly to the target platform ISA.

Our platform supports both the OpenCL and OpenGL framework. The CL Offline Compiler [1] is applied to compile an OpenCL kernel to HSAIL. Since we have no available compiler for GL shading language (GLSL) to HSAIL directly, we first compile a GLSL shader program to a pseudo assembly, NV_gpu_program_4, with the help of the Cg compiler [2]. Then we use in-house translator to convert the assembly to HSAIL. Both the CL and GLSL programs use the same finalizer to generate the real binary custom instruction.

Different APIs have their own runtime library, but all of them share the same HSA runtime, finalizer, and GPU low level driver. Notice that the graphic pipeline part is not formulated in the official HSA specification, here we extend the HSA runtime to support graphic applications.

IV. Benchmarks and Evaluation

We evaluate this platform with several benchmarks. OpenCL benchmarks are mainly the AMD sample programs. OpenGL benchmarks are programs of classic shading algorithms. On this platform, we can analyze the performance issue in different GPU microarchitecture, ISA design, task scheduling algorithms and SIMD control divergence handling mechanisms.

Acknowledgment

This work was supported by Ministry of Science and Technology of Taiwan under grant 104-2220-E-006-013.

References

[1] HSA Foundation, *CL Offline Compiler: Compile OpenCL kernels to HSAIL*, URL https://github.com/HSAFoundation/CLOC/.

[2] NVIDIA Corporation, *Cg toolkit user's manual: A developer's guide to programmable graphics. NVIDIA Corporation, 2003.*