

Using GPGPU-Sim to do Cost Modelling with Hybrid WCET Analysis

Adam Betts

July 31, 2012

Abstract

This documents changes made to GPGPU-sim to do cost modelling for the CARP project. It is a compendium of notes that also includes how to run it.

1 Introduction

Important notes follow:

- I have only changed v3.x of the simulator.

1.1 Simulator Source Files

For our research, we need both the CFGs of each function, a call graph of the entire kernel, and a timestamped trace of execution. GPGPU-sim provides the possibility to extract all this information, but only with some modifications to the source code. Here I explain which files and which functions were touched.

cuda-sim.cc::function_info::ptx_assemble(): Called before simulation starts to reconstruct the CFG of the kernel. In order to see the basic blocks (including PTX instructions) and the links between them, we have to add:

```
print_basic_blocks();  
print_basic_block_links();
```

shader.cc::LooseRoundRobbinScheduler::cycle() Simulates a cycle on a multiprocessor by deciding which instruction in the queue to issue. When the instruction is issued to the multiprocessor, it means that particular warp is in flight. To produce a trace of when this event occurs we have to add:

```
if(warp_inst_issued) {  
    printf("Issued: PC = 0x%04x, Warp = %d, SM = %d, cycle = %llu\n", pI  
        ->pc, warp_id, m_shader->m_sid, gpu_sim_cycle);  
}
```

1.2 Running

Running GPGPU-sim is a bit awkward. It consists of several steps:

- Set environment variable `CUDA.INSTALL_PATH` to point to the base directory of your CUDA installation. Since I am modifying v3.x and running the Fermi-based simulator (explained below), it is required that you have CUDA v4.x or greater installed and that `CUDA.INSTALL_PATH` points to that installation.
- Copy **all** the files from `v3.x/configs/arch/` to the directory where the CUDA binary lives.
- Run `'source v3.x/setup_environment'` in the directory where the CUDA binary lives. At this point, in principle, everything is good to go.
- Run the CUDA binary as normal, i.e. `'./a.out'`. If successful, you will see a splurge of output as GPGPU-sim dumps various information related to that specific run.