**CP3106 Project Report Format**

## Introduction

This document describes the standard format for the final project report to be submitted as part of the fulfillment of the CP3106 course. Students have to ensure their reports conform to the required format before submission for evaluation.

## Final Project Report

* 1. **Length of the Report**

The total length of the main report shall not exceed **50 A4 pages**. The text of the main report shall be single space, with a font size of at least **TIMES ROMAN 12.** Appendices and other manuals can be in a smaller font size. Appendices should be kept small and bounded together with the main report. However, user manuals, programmer manuals and bulky data dictionaries should be bounded as separate volumes. Please consult your project advisor if you are unsure which materials you should include in the main report and appendices and the length of appendices. The report should be clearly written, and should include only relevant information. Indeed the inclusion of too much detail may cause the evaluation committee to doubt whether the student has really learnt how to distinguish the important issues from the trivial ones.

## Format

All reports must be prepared in the following sequence:

1. Front hard cover
2. Title page
3. Abstract
4. Acknowledgment page (if any)
5. Table of contents
6. Main report
7. References (or bibliography)
8. Appendices
9. The back hard cover

Some of the important points on the report format are explained in the following sub-sections.

## Front and Back Cover and Binding

Ring binding should be used for all reports and appendices. As sketched in the sample pages, the words “CP3106 Project Report”, the project title (bigger font size), the student’s name, the words “Department of Computer Science or Department of Information Systems and Analytics, School of Computing, National University of Singapore” and the academic year (e.g., AY 2017/2018, Semester II) should appear on the front cover.

Front Cover of the Final Report – Sample

CP3106 Project Report

**Development of a Database Link Between Mainframe and PC**

By

Chua Meng Lee

Department of Computer Science School of Computing

National University of Singapore AY2017/2018, Semester II

## Title Page

The first page of the final report should be a title page. It should consist of the words “CP3106 Project Report”, the Title (centered, bold and two size larger), the Author Name, the words “School of Computing, National University of Singapore”, 20XX, and Advisor’s name, and deliverables (e.g., number of document volumes, user manuals, software and etc.) should also be included. The cover page should fit on one page.

# Title Page – Sample

CP3106 Project Report

**Development of a Database LINK Between Mainframe and PC**

By

Chua X X

Department of Computer Science School of Computing

National University of Singapore AY2017/2018, Semester II

Advisor: Assoc Prof XXX

Deliverables:

Report: 1 Volume

Manual: 1 Volume

Software: 1 USB

## 3.2.3 Abstract Page

It consists of an abstract of the report of not more than 250 words outlining the project. The abstract should be comprehensible to readers of the report and enable them to judge the report’s potential interest. The Keywords and Subject Descriptions should follow immediately after the abstract in the same page, each with not more than five careful selected items. The descriptors should be chosen from the latest version of “The Full Computing review Classification Scheme” of the ACM Computer Review. Any suitable word that reflects the nature and content of the project may be chosen as a keyword. The student should consult the project advisor when in doubt which keywords and descriptors be used.

# Abstract - Sample

Abstract

Increasing data size and applications has generated huge demand for computation capability, and parallel computing is a potential answer. Plenty of algorithms about parallel computing have been developed, and performance simulators are created accordingly. GPGPU-Sim is one of those simulators which can characterize applications written in NVIDIA’s CUDA programming model by running NVIDIA’s parallel thread execution (PTX) virtual instruction set. But GPGPU-Sim executes all the instructions in CPU-only sequential, which means it will take more time to run the same program compared with the time the GPU version takes. Meanwhile, NVIDIA provides a profiling tool called nvprof which can characterize CUDA programs and it runs programs in GPU. In our work, we tried to overcome the gap between nvprof and GPGPU-Sim. We have modified the GPGPU-Sim to let it generate performance metrics based on statistics that are given manually. We have also designed the transformation from nvprof’s metrics to GPGPU-Sim input. The final output of our modified GPGPU-Sim will be compared with that of original GPGPU-Sim.

Subject Descriptors: https://cran.r-project.org/web/classifications/ACM.html

* + 1. Network Architecture and Design
    2. Network Protocols

C.2.4 Distributed Systems

C.4 Performance of Systems

I.2.9 Keywords:

Wireless communication, routing protocols, distributed applications, fault tolerance, sensors

Implementation Software and Hardware:

Ubuntu Linux 7.04 Feisty Fawn, TinyOS 2.x, NesC 1.2.8a, Java 1.6 SE, Xbow Motes, Tembusu cluster

## Acknowledgement

Following the abstract page, students may want to acknowledge the contributions or assistance of others to the project. It should be kept in one A4 page.

## Table of Contents

In addition to the heading of each section, sub-heading can also be used but its depth should be kept to a minimum. Details of appendices should also be given here. Students may use more than one A4 page for the content page.

i ii iii iv

1

1

3

4

5

5

7

8

9

11

11

13

14

16

23

23

27

29

40

45

49

52

55

v vi

vii

* 1. 5.2 Limitations

5.3 Recommendations for Further Work References

Appendix A – Program Listing

Appendix B – How to Use the Program

Introduction

* 1. Motivation
  2. GPGPU-Sim
  3. GPUWattch

Selection of Architecture and Benchmarks

* 1. GTX480
  2. TITAN V
  3. Benchmarks

Implementation

* 1. Extraction and read of parameters
  2. Transformation
  3. Changes to GPGPU-Sim calculation process

Results

* 1. Analysis
  2. Comparison with original GPGPU-Sim

Conclusions

5.1 Summary

Title Abstract

Acknowledgement List of Figures

**Table of Contents**

# Introduction

## Motivation

There are plenty of ways to analyze a CUDA program’s performance and GPGPU-Sim is a typical simulator. However, one of its apparent disadvantages is that it takes much more time to finish the whole process than the original CUDA program. We observe that the performance counters used in GPGPU-Sim are possible to achieve from other methods(like NVIDIA profiling tool), so it is possible to generate the equivalent power estimation of GPGPU-Sim by passing these performance counters directly to it and generate the result instead of simulating the whole process in GPGPU-Sim. This means we can get the equivalent estimation of power consumed in the format of GPGPU-Sim’s output at a much faster speed. Faster performance analysis can provide insights in developing better algorithms and architecture, even if it is an estimation instead of real data. As long as the fault is tolerable, the estimation will make sense. It is a trade-off between accuracy and speed.

## GPGPU-Sim

GPGPU-Sim was created by Tor Aamodt's research group at the University of British Columbia.[[1]]

It is a cycle-level simulator modeling contemporary graphics processing units (GPUs) running GPU computing workloads written in CUDA or OpenCL. Also included in GPGPU-Sim is a performance visualization tool called AerialVision and a configurable and extensible energy model called GPUWattch.

GPGPU-Sim and GPUWattch have been rigorously validated with performance and power measurements of real hardware GPUs. More details will be introduced at the part of Background.

## GPUWattch

GPUWattch was collaboratively developed by researchers at UTAustin, UWisconsin, and UBC.[[2]] It is an energy model based upon McPAT that is integrated with GPGPU-Sim. McPAT is an architectural modeling tool for chip multiprocessors(CMP) The main focus of McPAT is accurate power and area modeling[[3]], and a target clock rate is used as a design constraint. McPAT performs automatic extensive search to find optimal designs that satisfy the target clock frequency

# Background

## GPGPU-Sim

Because GPGPU-Sim is so important to our project, so more details need to be included. GPGPU-Sim is a cycle-level GPU performance simulator that focuses on “GPU computing”(general purpose computation on GPUs). Its microarchitecture model only reports the cycles where the GPU is busy, which means it does not model either CPU timing or PCI Express timing(i.e memory transfer between GPU and CPU). GPGPU-Sim models GPU microarchitectures similar to those in the NVIDIA GeForce 8x, 9x, and Fermi series. Its accuracy has passes careful validation. As the author says, when they use the native hardware instruction set(PTXPlus), GPGPU-Sim 3.1.0 obtains IPC correlation of 98.3% and 97.3% respectively on a benchmark suite

### Top-Level Organization

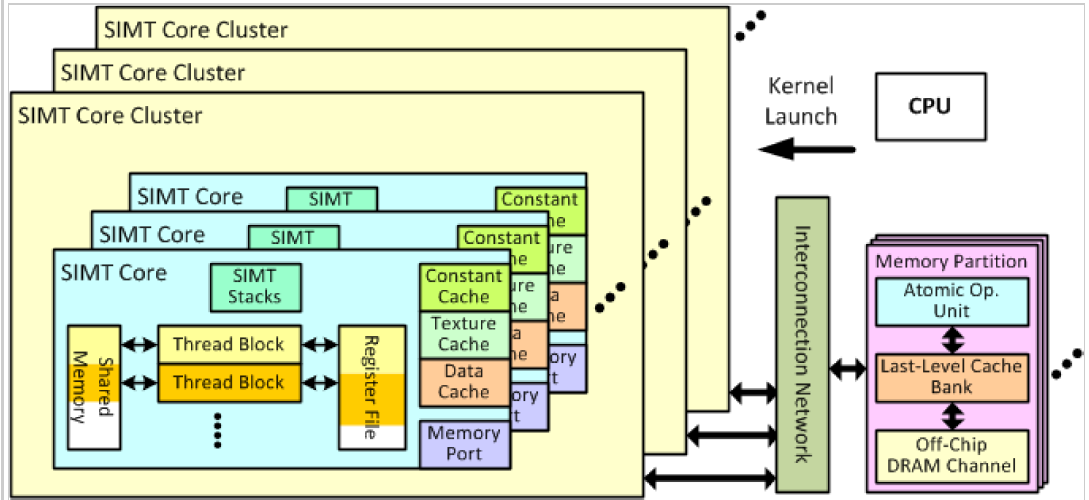
The GPU modelled by GPGPU-Sim is composed of Single Instruction Multiple Thread(SIMT) cores. Here a SIMT core is roughly equivalent to what NVIDIA calls an Streaming Multiprocessor(SM), and it models a highly multithreaded pipelined SIMD processor. A Stream Processor(SP) would correspond to a lane within an ALU pipeline in the SIMT core.

#### SIMT Core Clusters

Multiple SIMT cores make up one SIMT Core Cluster. The SIMT cores in one cluster share a common port to the interconnection network.

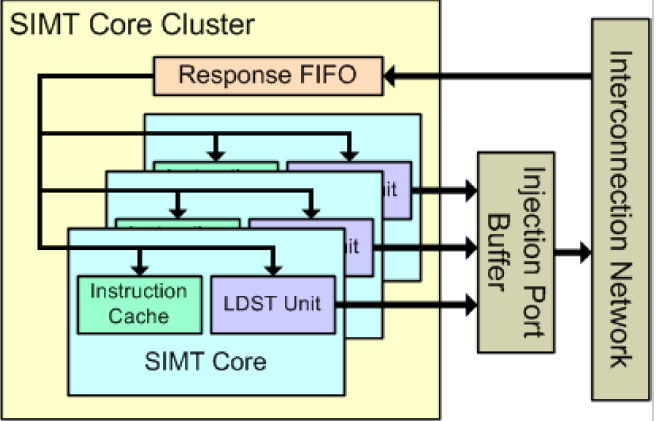
The figure below describes the overall situation, all the SIMT cores are grouped into clusters, and the clusters are connected to the interconnection network to memroy partitions that interface to graphics GDDR DRAM.

Figure :Overall GPU Architecture



The below figure describes the process that how a cluster deals with the packets ejected from the interconnection network. If the packet is a memory response servicing an instruction fetch miss, it will be directed to a SIMT Core’s instruction cache. Otherwise, it will be directed to the LDST Unit(the memory pipeline).

Figure :SIMT Core Clusters



#### SIMT Cores

A SIMT core models a highly multithreaded pipelined SIMD processor, which is roughly equivalent to what NVIDIA calls an Streaming Multiprocessor(SM).

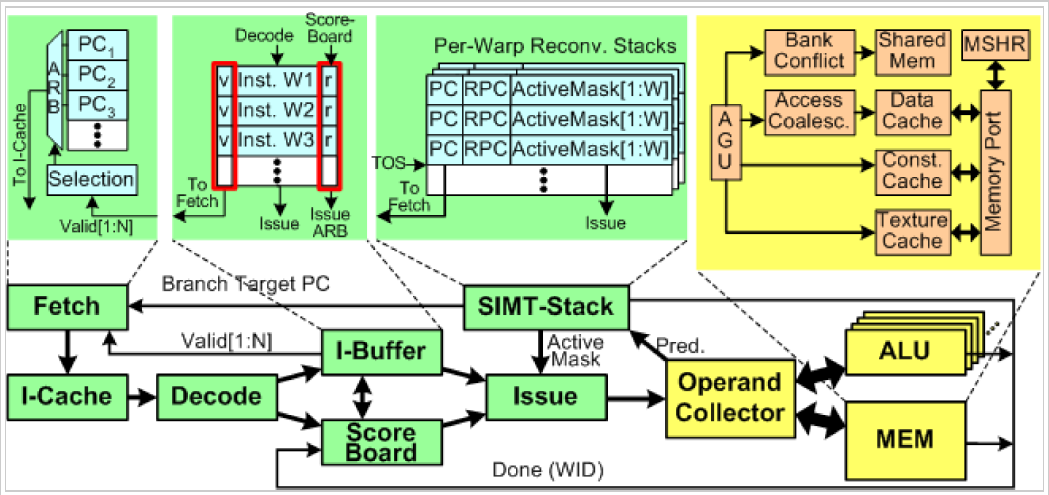
In one SM the warps are executed using pipeline technique and interleaving technique as well.

The lifecycle of one instruction consists of six stages: fetch, decode, issue, read operands, execute

and writeback,

The figure below shows the detailed microarchitecture Model of SIMT Core. For more details, can refer to GPGPU-Sim manual.

Figure :Detailed Microarchitecture Model of SIMT Core



## Reference

1. Analyzing CUDA Workloads Using a Detailed GPU Simulator
2. Jingwen Leng, Tayler Hetherington, Ahmed ElTantawy, Syed Gilani, Nam Sung Kim, Tor M. Aamodt, Vijay Janapa Reddi, GPUWattch: Enabling Energy Optimizations in GPGPUs, In proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA 2013), Tel-Aviv, Israel, June 23-27, 2013.
3. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures