

CS22014 实验报告模版

实验二、GPU PTX 与线程管理

姓 名: 计 春 生

学 号: 168668

班 号: xxxxxx

CS 22014 流计算结构及其程序设计
(春季, 2017)

西北工业大学
计算机学院
ERCESI

2017 年 5 月 7 日

摘 要

请在这里输入摘要内容.

版 权 声 明

该文件受《中华人民共和国著作权法》的保护。ERCESI 实验室保留拒绝授权违法复制该文件的权利。任何收存和保管本文件各种版本的单位和个人，未经 ERCESI 实验室（西北工业大学）同意，不得将本文档转借他人，亦不得随意复制、抄录、拍照或以任何方式传播。否则，引起有碍著作权之问题，将可能承担法律责任。

目 录

1 概述	5
2 PTX 指令序列产生	6
2.1 实验过程	6
2.2 实验结果和分析	6
3 PTX 与 x86 指令的比较	6
A 代码	6

1 概述

这一节请主要描述这次实验的主要实验内容。由于 latex 的中文使用了 ctex 包，所以需要 xelatex 进行编译。

- **PTX 分析**分析 NVidia GPGPU PTX 指令，判断实现矩阵运算需要的指令序列
- **与 x86 指令进行比较**比较 GPGPU 指令序列和 x86 标量指令序列实现矩阵运算的差异
- **分析 Warp 调度的过程**寻找 GPGPU-sim 中关于 Warp 调度管理的代码，分析计分牌算法
- **warp 调度的实验**分析不同的指令序列的 warp 调度能力，分别设定能够掩盖存储延时和不能掩盖存储访问延时的不同序列，分析计算性能。

表 1: 测试模式信号定义

信号名	方向	位宽	功能描述
boot	Input	1-bit	触发测试模式, 当处理器正常工作时被置为 0
test_im_wr	Input	1-bit	Instruction memory write enable in test mode,set to 0 in CPU regular process mode. In test mode, it will be set to 1 when if writing instructions to imem, otherwise it is set to 0.
test_im_re	Input	1-bit	Instruction memory read enable in test mode,set to 0 in CPU regular process mode. In test mode, it will be set to 1 when if reading instructions out, otherwise it is set to 0.
test_im_addr	Input	32-bit	Instruction memory address
test_im_in	Input	32-bit	Instruction memory data input for test mode.
test_im_out	Output	32-bit	Instruction memory data output for test mode.
test_dm_wr	Input	1-bit	Data memory write enable in test mode,set to 0 in CPU regular process mode. In test mode, it will be set to 1 when if writing data to dmem, otherwise it is set to 0.
test_dm_re	Input	1-bit	Data memory read enable in test mode,set to 0 in CPU regular process mode. In test mode, it will be set to 1 when if reading data out, otherwise it is set to 0.
test_dm_addr	Input	32-bit	Data memory address
test_dm_in	Input	32-bit	Data memory input for test mode.
test_dm_out	Output	32-bit	Data memory output for test mode.
valid	Output	1-bit	If CPU stopped or any exception happens, valid signal is set to 0.

```

    val test_im_wr      = Input(Bool())
    val test_im_rd      = Input(Bool())
    val test_im_addr    = Input(UInt(32.W))
    val test_im_in      = Input(UInt(32.W))
    val test_im_out     = Output(UInt(32.W))

    val test_dm_wr      = Input(Bool())
    val test_dm_rd      = Input(Bool())
    val test_dm_addr    = Input(UInt(32.W))
    val test_dm_in      = Input(UInt(32.W))
    val test_dm_out     = Output(UInt(32.W))

    val valid           = Output(Bool())
  }
  class Top extends Module() {
    val io              = IO(new TopIO())//in chisel3, io must be wrapped in IO(...)
    //...
    when (io.boot & io.test_im_wr){
      imm(io.test_im_addr) := io.test_im_in
    } .elsewhen (io.boot & io.test_dm_wr){
      // please finish it
    } //...
  }

```


参考文献

- [1] P. Erdős, *A selection of problems and results in combinatorics*, Recent trends in combinatorics (Matrahaza, 1995), Cambridge Univ. Press, Cambridge, 2001, pp. 1–6.