A Risc-V based co-processor peripheral to accelerate CNN inference.

Khalid Aleem
Trinity College

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: ka476@cam.ac.uk

April 15, 2021

Declaration

I Khalid Aleem of Trinity College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 14,235

Signed:

Date:

Abstract

Write a summary of the whole thing. Make sure it fits in one page.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

Background

Chapter 3

Related Work

Chapter 4

Design and Implementation

4.1 SoC Design

4.1.1 AXI4 Protocol

`wstrb`

The AXI4 protocol comes with the disadvantage of being able write a maximum of 4 bytes per clock cycle.

4.2 Fixing Piccolo

4.2.1 Floating Point Unit

Fixing FPU typeerrors

The latest commit of the Piccolo processor, (), has a type error in the floating point unit.

We must enter, FPU.bsv and FBox_Core.bsv and prefix the RoundMode type with FloatingPoint::RoundMode.

mstatus CSR

Once the FPU is successfully compiled, and the processor is compiled with floating point support, the processor will trap when a floating point instruction is executed.

```
Trap_Info epc: 'h8000227a, exc_code: 'h2, tval: 'h1a87a787 Trap Exc: ILLEGAL_INSTRUCTION
```

4.2.2 Test Accelerator Control

When constructing an accelerator, it is important to work constructively from a minimum functioning unit.

We make use of the test bench provided by Piccolo as a basis for our simulations.

We create a simple peripheral, with a region of memory that we can write 4-byte integers into. When a special execution bit is set, we set a special accelerator busy bit and unset the execution bit. In a single cycle, as part of the accelerator, if the busy bit is set, we double the array and unset the busy bit. This allows the firmware to write an array into the accelerator memory, set the execution control bit, and stall until the busy bit is unset.

AXI4 Fabric

4.2.3 Fabric Failure

This code results in the cpu trapping: `*((uint32_t)0xC0001002) = 1337;`

`Trap_Info epc: 'h800022d6, exc_code: 'h6, tval: 'hc0001002` The exception/trap code corresponds to: TRAP EXC: STORE_AMO_ADDR_MISALIGNED

This is of particular importance to consider when we issue a command to the accelerator. In order to ensure

4.3 Matrix Reduction

4.3.1 Overview

Mathematically, the result of a matrix multiplication may be expressed accordingly:

$$C = AB$$
$$C_{ij} = \sum_{k} A_{ik} B_{kj}$$

Matrix multiplication may be expressed in a programmatic form accordingly: `C[i][j] = sum zipWith (op*) (row A i) (col B j)`

This can be implemented in two possible ways. We could approach things purely in hardware, issue a command containing the locations and dimensions of each of the matrices, and expect the hardware to write an output matrix to our memory. Alternatively, we may take advantage of the existence of firmware. It is possible that

Our peripheral contains a RegFile of internal memory which is memory mapped to the CPU via an AXI4 bus slave interface.

Offset Address Range	Size	Purpose
`0x0000-0x0020`	32 Bytes	Status
`0x0020-0x0040`	32 Bytes	Control
`0x0040-0x1000`	$\approx$ 4 KB	Memory

Status

The first byte of our status memory is to monitor, and control execution from the accelerator. We write our control command/instruction to the control memory bank

Control

Memory

4.3.2 Unit Design

4.3.3 Unit Speed-up

4.3.4 Unit Parallelism - Naive

4.3.5 Memory Parallelism

4.3.6 Memory Striping

Chapter 5

Evaluation

Chapter 6

Summary and Conclusions