Matrix Operation Instructions in RISC-V Processor

Project specification

Callum Stew

Department of Computer Science

University of Warwick

1 Introduction

Many matrix operations involve computing many simpler calculations on the components of the matrix. These calculations are often independent and can be performed in parallel. A processor would compute these sequentially but a specific hardware implementation would be able to compute them in parallel. Adding hardware-based matrix operations to a processor instruction set would allow programs to reduce the number of cycles needed to get the result and so reduce the time taken. As the average clock speed of CPUs has seen very little increase since the early 2000s moving more complicated operations to hardware allows for run time to decrease without decreasing cycle time.

This project aims to develop hardware that can perform matrix operations and integrate that into the instructions of a RISK-V processor to allow improvement in the execution time of matrix heavy programs.

2 Background

2.1 FPGAs

As this project will use a hardware implementation, a platform is needed that can allow fast development and iterations on hardware designs. An FPGA (Field Programmable Gate Array) is a semiconductor device based around a matrix of configurable logic blocks [3]. The user can define what logic function these blocks perform and how they are connected using HDL (Hardware Description Language) programming. Hardware can be designed in HDL and constructed on the FPGA without needing to manually build the circuit.

2.2 Matrix Operations

There are four basic operations with matricies, addition, scalar multiplication, transposition and matrix multiplication. Each element of the result matricies of these functions can be independently calculated allowing for paralellism. In order to fully utalise this all elemnts of the input matricies must be accessable simultainus which limits the scalability of the size of input matrix the design can handle.

2.3 RISK-V, Rocket Chip and Chisel

For this project a processor is needed to intergrate the matrix accelerator into. RISC-V is an open standerd instruction set architecture that can be implemented in a veriaty of devices such as high-performance boards by SiFive and smaller microcrontrolers such as Espressif's ESP32-C3. This adoption means that more software is available for the RISC-V instruction such as Debian's risk64 port. To implement a RISC-V core on an FPGA the Rocket Chip Generator will be used. This is an open-source SoC generator that produces synthesizable RTL for implemntation on an FPGA. It also allows for intergration of custom accelerators as instruction set extentions [1] which is needed for this project.

Rocket Chip uses the Chisle HDL based on the Scala programming language. This allows for modern programming features to be used and then generate Verilog from this code. The Verilog code can then be synthasised for an FPGA.

3 Objectives

3.1 Hardware-Based Matrix Operations

3.2 Integration to RISC-V Core

3.3 Comparison of Hardware Solution Against Software

4 Methodology

5 Timeline

6 Resources

This project will require hardware and software resources listed below as well as open source tools such as Rocket Chip.

1. **FPGA Development Board - Nexys A7**
   This board will be used to synthesise the HDL generated by this project. It is on loan from the School of Engineering.
2. **Vivado Design Suit**
   Provides tools to synthesise and analyse HDL for FPGAs. Educational licence through the School of Engineering
3. **vivado-risk-v project** [2]
   An open-source project that generates RISC-V processors for FPGAs using Rocket Chip.
4. **Linux Workstation PC**
   A PC capable of running the Vivado Design Suit tools and the Ubunbtu 20.04 environment that is needed for the vivado-risk-v project. My PC fulfils these requirements.

7 Risks

Three main risks could harm this project. These are listed below along with mitigations and solutions.

1. **Damage to FPGA Development Board**
   This board is on loan from the School of Engineering so a replacement could be obtained. To reduce the chance of damage the board will only be transported when necessary and kept in its protective packaging when not in use.

2. **Damage to Workstation**
   Engineering computer labs could be used to run Vivado Design Suit and DCS labs can provide a Linux environment if needed.

3. **Data Loss**
   Git version control will be used to track changes and all data relevant to the project will be backed up to GitHub providing an easy way to recover data. A log will be kept of the process to generate data such as the use of vivado-risk-v so it can be repeated if necessary.

8 Legal, Social, Ethical and Professional Issues

This project should not present any social, ethical or professional issues and does not require working with people. There is no intent to profit from this project so no legal issues should arise.

References

[1] Asanovic, Krste & Avizienis, Rimas & Bachrach, Jonathan & Beamer, Scott & Biancolin, David & Celio, Christopher & Cook, Henry & Dabbelt, Daniel & Hauser, John & Izraelevitz, Adam & Karandikar, Sagar & Keller, Ben & Kim, Donggyu & Koenig, John & Lee, Yunsup & Love, Eric & Maas, Martin & Magyar, Albert & Mao, Howard & Moreto, Miquel & Ou, Albert & Patterson, David A. & Richards, Brian & Schmidt, Colin & Twigg, Stephen & Vo, Huy & Waterman, Andrew. The rocket chip generator. Technical Report UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html.

[2] Tarassov, Eugene. vivado-risk-v. github.com/eugene-tarassov/vivado-risc-v.

[3] Xilinx, AMD. What is an fpga? xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html. (Accessed October 2023).