Overview

Energy and efficiency are very important considerations when designing hardware systems. MANIC\(^1\) introduces vector dataflow execution in order to drastically reduce the energy use by amortizing energy costs across grouped executions of instructions. I specifically worked on adding fixed function accelerators to the design, such as sorting. I did this by writing in SystemVerilog at the register-transfer level.

Approach

The main focus was on the implementation of the various fixed function operations. The overall high level of the operation of the units was that the core would interact with each of the individual units through memory mapped IO (MMIO). This is shown in a high level in the diagram immediately below.

A bus and dedicated arbitration unit were also implemented in order to handle concurrent accesses to memory through loads and stores by the individual units.

Overall Design

Bus and Arbitration

Bus and the arbiter used round robin arbitration. They were designed to be highly parameterizable such that one could keep on adding new accelerators easily. This is necessary since we would like to have multiple accelerators working on different pieces of data concurrently.

Next Steps

I intend to keep developing and working on my designs. The next couple steps for this work currently would to get some solid results for measuring the performance, since all I have currently is correctness. This was already tough as it was to get, considering writing and designing with Verilog is much more involved than higher level programming.

Convolution

This was the most technically complicated unit. The convolution unit supports 2D dense convolution with the capability of a column wise-stride. The core dataflow and design was inspired by Eyeriss\(^2\), which has the same dataflow in the diagram above.

This specific dataflow was chosen because it enabled the most reuse of activations, filters, and partial sums as they were accumulated. As in the diagram, filters flow to the right, accumulated partials flow upwards, and activations flow diagonally.

In addition, future developments include support for packed inputs to make it possible to work with smaller datatypes, and double or quadruple throughput.

Convolution

Bus and the arbiter used round robin arbitration. They were designed to be highly parameterizable such that one could keep on adding new accelerators easily. This is necessary since we would like to have multiple accelerators working on different pieces of data concurrently.

Next Steps

I intend to keep developing and working on my designs. The next couple steps for this work currently would to get some solid results for measuring the performance, since all I have currently is correctness. This was already tough as it was to get, considering writing and designing with Verilog is much more involved than higher level programming.

Convolution

Bus and the arbiter used round robin arbitration. They were designed to be highly parameterizable such that one could keep on adding new accelerators easily. This is necessary since we would like to have multiple accelerators working on different pieces of data concurrently.

Next Steps

I intend to keep developing and working on my designs. The next couple steps for this work currently would to get some solid results for measuring the performance, since all I have currently is correctness. This was already tough as it was to get, considering writing and designing with Verilog is much more involved than higher level programming.