Authors: Mahdis Mirzaei, Amir Moumeni

Report: MLP Layer and Accelerator Integration

Introduction

This report summarizes the development of an Object-Oriented SystemC model for an MLP (Multi-Layer Perceptron) layer and its integration with a matrix multiplier accelerator. The project follows the requirements outlined in the assignment 'Computer Assignment 4 - OOE Spring 1404 - v2', which focuses on processor bracketing, hardware acceleration, and data communication using custom channels.

This project was developed by Mahdis Mirzaei and Amir Moumeni.

1. MLP Layer Implementation

```
        SystemC 2.3.3-Accellera --- Jun 28 2025 17:2
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
-30
64
```

The first layer of the MLP (3–8–10 topology) was implemented using C++ and SystemC. This layer performs a matrix multiplication of a 3x1 input vector with a 3x8 weight matrix, resulting in 8 outputs. The inputs and weights are read from input files (inputs.txt, weights.txt), and results are written to outputs.txt.

Insert your MLP diagram image here.

2. RISC-V Clock Estimation

An approximate calculation of the number of clock cycles required for the software-based matrix multiplication was performed. Each multiplication and addition was considered a single clock cycle, leading to an estimated total of approximately 120–150 cycles for the 3x8 layer computation, excluding I/O operations.

3. Matrix Multiplier Accelerator (BFM)

A Bus Functional Model (BFM) of a matrix multiplier was developed to accelerate the matrix computation. The accelerator handles 1x3 × 3x2 multiplications (producing 1x2 outputs)

and uses local buffers for inputs and outputs. Upon completion, the accelerator raises an interrupt signal (intrMMA) to notify the processor.

4. Channel Design

A channel interface (channel_if) was implemented with blocking put() and get() methods for safe data transfer between the processor and the accelerator. The data_channel class uses an internal queue and SystemC events to manage data flow.

5. Processor and Accelerator Integration

The Processor module was developed to orchestrate data transfer and computation. The 3x8 matrix multiplication is divided into four 3x2 blocks, each sent to the accelerator sequentially. The processor waits for interrupts from the accelerator to fetch the results, assembling the final 1x8 output vector.

6. Performance Estimation

Considering a 32-bit data bus and a burst length of 8, transferring the inputs and weights requires a few cycles per block. Including computation, the accelerator-based approach reduces the cycle count compared to the software-only model, with an estimated total of 40–50 cycles for each 3x2 multiplication.

7. Conclusion

The implementation successfully demonstrates the use of SystemC for modeling a processor with an integrated accelerator. The matrix multiplier accelerator improves performance significantly by offloading the computationally expensive matrix operations, and the channel-based communication ensures reliable data transfer between components.