Systolic Array Matrix Multiplication: A Scalable Hardware Design

Noah Bean

February 23, 2025

1 Introduction

Matrix multiplication is a fundamental operation in numerous applications, including machine learning, signal processing, and scientific computing. Traditional software implementations on general-purpose processors can be inefficient for large matrices due to sequential execution and memory bottlenecks. This project presents a hardware-accelerated solution using a systolic array architecture, implemented in SystemVerilog, to perform matrix multiplication efficiently for arbitrary $n \times n$ matrices. The design is scalable, pipelined, and adheres to industry best practices, making it suitable for modern FPGA or ASIC deployments.

2 Design Overview

A systolic array is a grid of processing elements (PEs) that rhythmically compute and pass data to neighbors, resembling a heartbeat. For matrix multiplication $C = A \times B$, where A and B are $n \times n$ matrices, the design employs an $n \times n$ array of PEs. Each PE computes one element of the result matrix $C[i][j]$ by accumulating partial products as follows:

$$C[i][j] = \sum_{k=0}^{n-1} A[i][k] \cdot B[k][j]$$

Data flow is orchestrated such that matrix A moves rightward through rows, and matrix B moves downward through columns, enabling parallelism and reducing memory access overhead.

3 Implementation Details

The implementation consists of two primary SystemVerilog files:

3.1 Design Module: `systolic_array_matrix_mul.sv`

This module defines the systolic array with the following features:

- **Parameterization**: Configurable matrix size (`N`) and data width (`DATA_WIDTH`), defaulting to 3×3 and 16 bits.
- **PE Grid**: An $n \times n$ array of PEs, each performing multiply-accumulate (MAC) operations. The PE module (`pe`) pipelines inputs A and B and accumulates results.
- **Control Logic**: A finite state machine (FSM) manages computation over $2n - 1$ cycles, feeding skewed inputs and capturing results when complete.

- **Inputs/Outputs**: Matrices A and B as 2D arrays, with C as the output, plus control signals `start` and `done`.

A snippet of the PE instantiation is shown below:

```
generate
    for (i = 0; i < N; i++) begin : row_gen
        for (j = 0; j < N; j++) begin : col_gen
            pe #(.DATA_WIDTH(DATA_WIDTH)) pe_inst (...);
        end
    end
endgenerate
```

3.2 Testbench: `systolic_array_matrix_mul_tb.sv`

The testbench verifies the design:

- **Stimulus**: Initializes matrices A and B with test values (e.g., $A_{i,j} = i \cdot N + j + 1$).
- **Verification**: Computes expected results and checks against C after `done` is asserted.
- **Best Practices**: Uses modern SystemVerilog constructs, self-checking logic, and proper clock/reset sequencing.

4 Results and Performance

For $n = 3$, the design completes in 5 cycles ($2n - 1$), achieving full parallelism across 9 PEs. Compared to the original 3x3 design, which used a linear pipeline, this systolic array scales seamlessly to larger n, reduces latency from $O(n^3)$ to $O(n)$ cycles, and eliminates hardcoded memory constraints.

5 Conclusion

This project successfully implements a scalable systolic array for matrix multiplication, optimized for hardware efficiency and flexibility. Future enhancements could include wider accumulators to prevent overflow, support for rectangular matrices, or integration with external memory interfaces for real-world applications.