Systolic Array Design for Matrix Multiplication

Introduction

This report outlines the design and implementation of a systolic array in Bluespec System Verilog (BSV) to perform 4×4 matrix multiplication. The systolic array is composed of interconnected processing elements (PEs), instantiated as **mk_matrix** modules, and orchestrated by the **mk_sys_array** module. It uses pipelined FIFOs and rules to ensure data flow and compute the multiplication output in a systematic, parallel fashion.

Module Overview

1. mk_matrix Module

This module represents an individual processing element of the systolic array. Each element computes partial products and accumulates results using an integrated MAC (Multiply-Accumulate) unit. It communicates with adjacent elements via FIFOs.

Subcomponents:

- **MAC Unit:** The **mk_mac_unit** performs multiply-accumulate operations.
- **FIFOs:**
  - **ff_a**, **ff_b** (16-bit): Store input elements from matrix (A) and (B), respectively.
  - **ff_c** (32-bit): Stores intermediate results for accumulation.
  - **ff_s** (1-bit): Select line for controlling the operation mode.

Methods:

- **Input Methods (get_a, get_b, get_c, get_s):**
  - Enqueue data into respective FIFOs.
- **Output Methods (put_a, put_b, put_c, put_s, put_mac):**
  - Dequeue data or return computed results.
- **Rule rl_calculation:**
  - Fetches the first elements from FIFOs, sends them to the MAC unit, and triggers computation.

2. mk_sys_array Module

This module orchestrates the overall systolic array, comprising a 4×4 grid of **mk_matrix** modules. It manages data flow across the array using pipelined FIFOs and implements matrix multiplication through

systematic data movement and computation.

Subcomponents:

- **Matrix Processing Elements:** Sixteen **mk_matrix** instances arranged in a 4×4 grid.
- **Output FIFOs:**
  - **ff_mac0**, **ff_mac1**, **ff_mac2**, **ff_mac3** (32-bit): Store final results for each row.

Rules:

- **Rules for Propagating Matrix (B):**
  - **stage1_b**, **stage2_b**, **stage3_b**: Pass elements of (B) down rows in the array, ensuring proper alignment for multiplication.
- **Rules for Propagating Matrix (A):**
  - **stage1_a**, **stage2_a**, **stage3_a**: Pass elements of (A) across columns in the array.
- **Rules for Accumulating Results:**
  - **stage1_c** to **stage7_c**: Handle intermediate and final accumulation of results.

Methods:

- **Input Methods (get_A, get_B):**
  - Load initial rows of (A) and columns of (B) into the array.
- **Output Methods (put_B, put_index0, put_index1, put_index2, put_index3):**
  - Retrieve the final results after computation.

Design Workflow

Initialization

- **Matrix (A):** Rows are loaded sequentially into the first column of PEs using the **get_A** method.
- **Matrix (B):** Columns are loaded into the first row of PEs using the **get_B** method.

Data Propagation

- (A)'s elements propagate horizontally across the rows.
- (B)'s elements propagate vertically down the columns.

Computation

- Each PE multiplies the received (A) and (B) elements and adds the result to its accumulated value. This computation is controlled by the **rl_calculation** rule within **mk_matrix**.

Result Collection

- Partial sums are passed to the next PE along the diagonal, and final results are accumulated in the last row of PEs.
- Output FIFOs (**ff_mac0** to **ff_mac3**) store results for each row.