AXI-Based Dot Product Accelerator

Project Overview

The implementation of a hardware accelerator for computing the dot product of two vectors stored in memory was carried out. The design consists of:

- **An AXI Master Interface:** Used to read the input data from BRAM and to write the computed 32-bit dot product back to memory.
- **AXI BRAM Controller and BRAM:** The AXI master interacts with the AXI BRAM controller to read and write data into Block RAM (BRAM).
- **Dot Product Computation Module:** This module receives the input data, performs element-wise multiplication and accumulation, and generates the final result.

The accelerator computes the dot product by multiplying corresponding elements of the two vectors and summing the results. The final 32-bit result is stored in BRAM, and the done signal is asserted high to indicate computation completion.

Implementation Details

1. **Start Signal Control:**
    - The start signal was provided directly from the testbench to initiate the operation.
2. **AXI Master and BRAM Handling:**
    - The AXI master was connected to the AXI BRAM controller, which managed BRAM access for both reading and writing data.
    - The BRAM was loaded with values by the AXI master state machine.
    - These values were then read back from BRAM and provided as inputs to the dot product computation module.
3. **Dot Product Computation:**
    - The computation module performed the multiplication and accumulation of values read from BRAM.
    - The computed 32-bit result was stored back into BRAM.
    - Upon completion of the computation, the done signal was asserted high.
4. **Control Flow:**
    - The testbench asserted the start signal.
    - The AXI master initiated loading values into BRAM.
    - The dot product computation module read values, computed the result, and stored it back into BRAM.
    - The done signal was set high to indicate the completion of computation.

Project Directory Structure

Below is the structured hierarchy of files and directories in the project:

AXI-Dot-Product-Accelerator - No AXI Lite/

```
│── diagrams/                    # Block diagrams and testbench simulation results
│   ├── Block Diagram.pdf        # Architectural overview of the design
│   ├── testbench_sim results.png   # Screenshot of simulation results
│
│── documentation/               # Contains design documentation
│
│── python_code/                 # Python scripts related to verification
│   ├── accum.py                 # Python script for computing dot product
│   ├── dot_product_output.txt   # Output file of computed dot product
│
│── src_code/                    # Source files for the hardware implementation
│   ├── controller.sv            # Control logic for the accelerator
│   ├── counter.sv                # Counter module for iterations
│   ├── design_1.v               # Main design implementation
│   ├── design_1_wrapper.v       # Wrapper for top-level module
│   ├── dot_product_accel.sv     # Core computation module for dot product
│   ├── regn.sv                  # Register handling module
│   ├── top_level.sv             # Top-level module integrating components
│   ├── IPs/                     # Custom IP cores used in the design
│       ├── axi_master_m_1.1_0/  # AXI master version 1.1
│       ├── axi_master_m_1.2_0/  # AXI master version 1.2
│       ├── dot_prod_v1.0_0/     # Dot product accelerator custom IP
│
│── testbench/                   # Testbench and related simulation files
│   ├── testbench.sv             # SystemVerilog testbench for verification
│   ├── testbench_behav.wcfg     # Waveform configuration file
│   ├── testbench_sim results.png   # Screenshot of simulation results
│
└── The_Zynq_Book_ebook.pdf      # Reference book on Zynq architecture
```

Verification and Testing

- A testbench was developed to:
    - Provide the start signal to the accelerator.
    - Model BRAM for the AXI master interface.
    - Load known 8-bit values into BRAM.
    - Verify that the computed dot product matched the expected result.
    - Ensure proper control flow transitions (idle, busy, done).
- The implementation was simulated using tools such as ModelSim and Vivado Simulator.

Conclusion

The hardware accelerator was successfully implemented with an AXI master interface handling BRAM access, and a dot product computation module performing the required operations. The absence of an AXI-Lite interface simplified control by allowing the testbench to directly manage the start signal. The design effectively computed the dot product, stored the result in BRAM, and asserted the done signal upon completion.