Finite Impulse Response (FIR) Filter Accelerator For SoC

Yazhuo Gao

1 INTRODUCTION

This project focuses on designing a Finite Impulse Response (FIR) filter accelerator to optimize computation in a System-on-Chip (SoC) environment. The primary objectives include:

1. Partitioning computation between hardware and software for improved performance.
2. Minimizing time to complete two FIR filter operations.
3. Achieving zero error while maintaining area efficiency.

The results compare a Hardware-Software (HW-SW) co-simulation design to a baseline Software-Only (SW-Only) implementation, highlighting the performance gains and resource overhead.

2 DESIGN OBJECTIVES

Performance: Minimize $(cycles \times critical\text{-}path\ delay)$ to improve overall system speed.
Area Efficiency: Ensure that additional hardware resources improve computation time without unnecessary area overhead.
Accuracy: Ensure the accelerator produces zero error in all FIR computations.
Compatibility: Ensure the design is synthesizable using High-Level Synthesis (HLS) tools.

3 SYSTEM DESCRIPTION

3.1 Hardware Design

The Accelerator module was implemented to perform FIR filter computations. Key features:

• Double FIR Operation: Separate coefficient buffers (`coef1` and `coef2`) and input buffers (`input_buffer1` and `input_buffer2`) ensure independent processing of the two FIR operations.

• Sliding Window: The input buffer dynamically maintains the latest 16 input values for each filter.

• FIR filter operation switch: The `ctrl` signal read from `accel_ctrl` in fir.c controls the switch between the first/second FIR filter operation.

• Pipelined Execution: The critical computation loop was optimized with pipelining to reduce latency.

3.2 Software Integration

The `fir.c` program on the Rocket CPU:

1. Configures DMA transfers to load input and coefficients into the accelerator.
2. Switches FIR filter operations using the `accel_ctrl` register.
3. Retrieves the computed results from the `z_out` FIFO.

4. Performs error checking by comparing accelerator outputs with expected values.

4 PERFORMANCE ANALYSIS

Simulator	HW-SW Design	SW-Only Design	Speedup
Rocket_sim	1447 ns	170881 ns	118.1x
VSIM	6358 ns	170881 ns	26.9x

Note: HW-SW Design runs the transfer with the accelerator and SW-Only Design runs with the systemC program only.

4.1 Simulation Time Comparison

The HW-SW design achieved a significant speedup by offloading FIR computations to the accelerator.

4.2 Critical Path Delay and Cycles

- Critical Path Delay: 2.376457 ns
- Cycles: 6339
- Throughput: 4 results per cycle

The accelerator efficiently pipelines the FIR computations to maintain a high throughput with a minimal critical path delay and cycles.

5 AREA ANALYSIS

The table below summarizes the area resources consumed by the FIR accelerator:

Area Resources used in Accelerator				
area_mux	area_func	area_logic	area_buffer	area_mem
2978.3	1600.8	1753.1	0	0
area_rom	area_reg	area_fsm_reg	area_fsm_comb	total
0	9214.2	16	16	15578.4

Comparison to RocketTile:

The RocketTile has an assumed area score of **1,500,000 units**. The FIR accelerator's total area usage of **15,578.4 units** represents only **~1.04%** of the RocketTile area, demonstrating excellent area efficiency.

6 IMPLEMENTATION DETAILS

6.1 Hardware-Specific Modifications

- Separate Buffers: Using `coef1`, `coef2`, `input_buffer1`, and `input_buffer2` ensured data separation between the two FIR operations.

• Initialization: All signals and FIFOs were initialized during reset to avoid uninitialized states.

6.2 Software Modifications

• DMA transfers were optimized to preload input and coefficients efficiently.

• Control logic ensured sequential execution of the first and second FIR operations.

7 DESIGN CHALLENGES

• Buffers and computation: Apply sliding window to store the lastest values needed for computation, and implement correct algorithm to satisfy the shifting dot product.

• Dynamic FIFO Management: The x_in, w_in, and z_out FIFOs were prone to stalling if not managed correctly. Explicit checks for Empty and Full states resolved this.

• Scheduling Constraints: SystemC constraints, such as avoiding dynamic range operations, required careful loop restructuring.

• Critical Path Scheduling: Balancing area usage and pipeline optimization to achieve better critical-path delay and cycles.

8 CONCLUSION

The HW-SW FIR filter accelerator achieved substantial speedup of 118.1x (Rocket_sim) and 26.9x (VSIM) compared to the SW-Only baseline, while maintaining area efficiency and zero error. The critical path delay of 2.376457ns and 6339 total cycles demonstrate the accelerator's efficiency. With a total area usage of only 15,578.4 units, representing ~1.04% of the RocketTile, the design balances performance and area usage effectively. The modular design also ensures scalability for additional filter operations with ctrl signal and sliding window. Future work could explore further pipelining or parallelization to enhance performance.