Parallelized FIR Filter Implementation

1. Introduction

The design processes 211 taps with a parallel factor $L = 2$ and $L = 3$, thereby reducing overall latency compared to a serial implementation while maintaining efficient resource usage.

2. Design Overview

Parameterization and Coefficient Storage:

Key parameters include:

- `N` $= 211$: Total number of taps.
- `IN_WIDTH` $= 16$, `COEF_WIDTH` $= 16$: Bit-widths for input and coefficient values.
- `MUL_WIDTH` $= 32$ and `ACC_WIDTH` $= 40$: Bit-widths for multiplication results and the accumulation register.
- `L` $= 2$: Parallel processing factor.

The FIR coefficients are stored in a local parameter array, enabling compile-time optimization and ensuring constant coefficient access.

Shift Register Implementation:

A shift register holds the most recent N input samples, where the newest sample is at index 0. This data structure is pivotal for performing convolution:

$$y[n] = \sum_{k=0}^{N-1} x[n-k] \cdot h[k]$$

where $x[n]$ represents input samples and $h[k]$ are the filter coefficients.

3. Parallel MAC Operation and FSM Control

Parallel Multiplication and Accumulation:

The design computes up to L products per cycle. A combinational loop iterates over indices:

$$\texttt{sum\_of\_products} = \sum_{j=0}^{L-1} \begin{cases} \texttt{shift\_reg}[\texttt{cycle\_count} \times L + j] \cdot \texttt{COEFF}[\texttt{cycle\_count} \times L + j] & \text{if index} < N \\ 0 & \text{otherwise} \end{cases}$$

The accumulator updates iteratively over $\lceil N/L \rceil$ cycles, reducing the total cycle count required for full convolution.

Finite State Machine (FSM):

The control logic comprises three states:

- **ST_IDLE:** Waits for a valid input; on detection, shifts the input into the register.
- **ST_ACC:** In each cycle, adds the computed partial product sum to the accumulator.
- **ST_DONE:** Upon processing all taps, asserts the output valid signal and latches the final result.

4. Testbench and Timing

A dedicated testbench instantiates the filter with $L = 2$ and $L = 3$ and applies test vectors, including an impulse (16'h4000) and subsequent zero samples. A clock generator creates a 15 ns period clock ($\approx$ 66 MHz). This timing constraint is critical for meeting performance targets in FPGA implementations.