Pipelined FIR Filter with Parallelism ($L = 3$)

1. Introduction and FIR Equation. A finite impulse response (FIR) filter of length $N = 211$ computes:

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

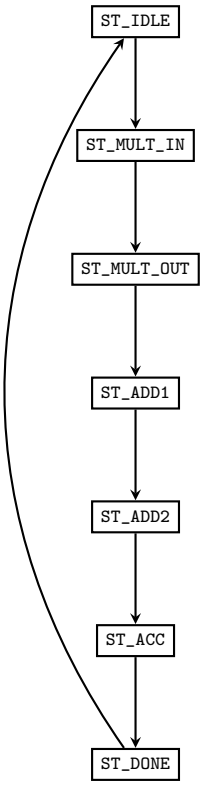
where h_k are signed 16-bit coefficients and $x[n]$ are input samples. Our Verilog design uses:

- **Parallelism** ($L = 3$) to process three MAC (multiply-accumulate) operations per iteration.
- **Pipelined stages** to improve throughput by breaking computation into smaller steps.

2. Design Structure.

- **Parameters:** `N=211`, `IN_WIDTH=16`, `COEF_WIDTH=16`, `MUL_WIDTH=32`, and `ACC_WIDTH=40`.
- **Shift Register:** A 211-entry array (`shift_reg`) that holds recent input samples (new sample enters index 0).
- **Coefficient Array:** `COEFF[0:N-1]` stores the 211 FIR taps.
- **Chunks:** The 211 taps are grouped into $\lceil 211/3 \rceil = 71$ chunks; each chunk handles 3 coefficient-sample multiplications.

3. Pipeline and FSM. The filter uses six pipeline steps: sample latch, multiply, two add stages, accumulation, and final output. The FSM states:



- **ST_MULT_IN & ST_MULT_OUT:** Input registers drive three parallel multipliers, producing three 32-bit products.
- **ST_ADD1 & ST_ADD2:** Two add stages reduce partial products into a single sum.
- **ST_ACC:** The partial sum is added to a 40-bit accumulator (`accum_reg`).
- **ST_DONE:** Outputs the final filter result and asserts `data_out_valid`.

4. Test Bench Summary. A 100 MHz clock is generated, and an impulse (`16'h4000`) is fed as input to verify the impulse response. Once `data_out_valid` is high, the output is sampled for confirmation.

5. Conclusion. This parameterizable design efficiently handles large-tap FIR filters through $L = 3$ parallelism. Multiple pipeline stages minimize the critical path by splitting multiply-and-accumulate operations. The accompanying test bench verifies correctness under impulse and zero-input scenarios.