64-Tap 16-Bit FIR Filter Design

Xingcan Chen
Department of Electrical Engineering
Columbia University
New York, NY 10027, USA
xc2807@columbia.edu

This paper describes the full design and hardware implementation of a 64-tap finite impulse response digital filter fabricated in IBM 130 nm CMOS technology. The filter handles 16-bit Q1.15 fixed-point input samples at a data rate of 10 kS/s and generates 16-bit Q7.9 output samples. The architecture uses a sequential multiply–accumulate structure running at 100 MHz to reduce silicon area while still supporting real-time signal processing. The system includes dedicated clock domain crossing logic to manage asynchronous input sampling at 10 kHz and internal computation at 100 MHz.)

I. INTRODUCTION

Finite impulse response filters serve as basic components in digital signal processing systems. These filters apply discrete-time convolution between input signals and fixed coefficient values to produce specific frequency-domain responses. Hardware realizations of FIR filters require careful tradeoffs among processing speed, power usage, and silicon area, while fixed-point arithmetic must preserve numerical accuracy during computation.

This work implements a 64-tap FIR filter intended for real-time audio signal processing. The design receives input samples at a 10 kHz sampling rate and represents them using a 16-bit Q1.15 fixed-point format. All sixty-four filter coefficients use the same Q1.15 representation. The output uses a Q7.9 format to handle the larger dynamic range introduced by accumulation. The architecture operates with two clock domains, where a 10 kHz clock aligns with external data capture and a 100 MHz internal clock supports the required computation rate.

II. SYSTEM ARCHITECTURE

A. Fixed-Point Number System)

The design uses two fixed-point formats chosen for different signal properties. Input samples and filter coefficients apply a Q1.15 representation that includes one sign bit and fifteen fractional bits. This format covers values from −1.0 to +0.999969482421875 with a resolution of 2^−15. Conversion from floating-point values to Q1.15 uses rounding with saturation, where the integer value V_q equals round(V × 2^15) and is limited to the range −32768 to 32767.

B. Top-Level Block Diagram

The system is divided into five main functional blocks as defined in the testbench architecture. The synchronizer module handles clock domain crossing between the 10 kHz input sampling domain and the 100 MHz processing domain. A three-stage flip-flop chain aligns the valid signal to the faster clock, and a holding register stores the input data. Edge detection logic produces a single-cycle pulse in the fast clock domain when a new input sample arrives.

The memory subsystem contains two distinct storage arrays. Coefficient memory stores 64 pre-loaded 16-bit values in an addressable array. External control signals enable selective writing during initialization. The data memory implements a 64-position shift register. New input samples enter at position zero. Existing samples shift to higher indices on each load cycle. Both memories provide combinational read access indexed by the tap counter.

The arithmetic logic unit integrates a 16×16 signed multiplier with a 64-bit accumulator. The multiplier processes one input sample and one coefficient in each cycle. The 32-bit multiplication result is extended to 64 bits before accumulation. The accumulator supports independent clear and enable signals. Rounding and saturation logic operates on the accumulated result to produce the final 16-bit output.

The finite state machine controls datapath activity using five defined states. The IDLE state checks the synchronizer for new input samples and resets the accumulator. The LOAD state enables shifting to update the data register. The COMPUTE state iterates through tap indices from 0 to 63 while activating multiply-accumulate operations. The OUTPUT state enables rounding and saturation for result conversion. The WAIT_ALU state allows one extra cycle for the registered output to stabilize before the output valid signal is asserted.

III. RTL IMPLEMENTATION

A. Synchronizer Module

The synchronizer transfers 16-bit data together with a valid indicator from the clk1 domain to the clk2 domain. A holding register captures the input data when valid_in is asserted on the clk1 clock edge, and this register keeps the value stable while crossing the clock boundary. Three cascaded flip-flops clocked by clk2 form a synchronization chain for the valid signal. Metastability reduction is in the first two flip-flop stages, while the third stage provides a clean synchronized output. This output is compared with its delayed version using edge detection logic. A single-cycle pulse on valid_out is generated when the synchronized signal changes from low to high. The data output samples the value stored in the holding register during this pulse.

B. Memory Module

The memory module contains two separate register arrays that support coefficient storage and data buffering. The coefficient memory consists of sixty-four individual 16-bit registers controlled by decoded write enable signals, and a load control signal gates the clock enable so that selected coefficients can be updated during initialization while continuous read access is preserved during normal operation. The data shift register is implemented using a behavioral for-loop, where the reset condition sets all register positions to zero and normal operation depends on a shift enable signal. When shifting is active, position zero loads the external input sample and positions one through sixty-three load data from their immediately preceding registers, forming a sequential shift chain that moves data toward higher indices, while a tap index input allows direct combinational read access to any stored value.

C. ALU Module

The ALU module comprises of multiplication, accumulation and output formatting. A wire assignment is a test that uses the native arithmetic operators in Verilog to do signed 16 plans 16 multiplication. Bit replication of the sign bit is used in sign extension of the 32-bit product to 64 bits.

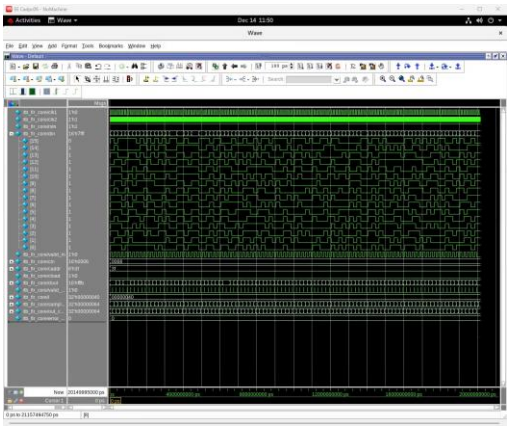


Fig. 1 waveform simulation output

The accumulator represents a registered data path with distinct next-state logic. Combinational logic chooses between cleared value, accumulated sum or held value according to control inputs.

There are two stages to output formatting. Depending on the accumulator sign, symmetric rounding adds or subtracts 2^20. The Q7.9 scale is obtained by arithmetic right shift by 21 positions. Saturation comparison looks for values that are higher than 16 bits. Saturation levels are defined by maximum and minimum constants. Conditional assignment chooses between the actual shifted value and saturated limits.

D. Top-Level Integration and FSM

The top module instantiates four functional blocks and implements the control state machine. A three-bit state encoding represents five operating states, and combinational next-state logic evaluates the current state together with input conditions. The valid signal from the synchronizer causes a transition from IDLE to LOAD, and fixed progression advances the system from LOAD to COMPUTE. Completion of the COMPUTE phase depends on the tap counter reaching its terminal value, after which the OUTPUT and WAIT_ALU states execute in sequence before control returns to IDLE. Control signals are generated through combinational logic based on the active state. In the IDLE state, the accumulator clear signal is asserted. The LOAD state enables data shifting. The COMPUTE state activates multiply-accumulate operation and increments the tap counter. The OUTPUT state enables rounding logic. The WAIT_ALU state asserts the output valid signal and captures the computed result.

IV. VERIFICATION METHOD

A. MATLAB Golden Reference

The verification flow starts with a MATLAB implementation of the filter algorithm. The reference model applies Q1.15 quantization to both input samples and filter coefficients. All multiplications and accumulations use integer arithmetic and preserve full Q2.30 precision throughout computation. Output conversion uses symmetric rounding followed by saturation to the target range. The model runs on 10,000 randomly generated input samples. The resulting data files store the quantified inputs, coefficient values, and corresponding expected outputs.

B. RTL Simulation

The testbench instantiates the FIR core and produces the required clock signals. Clock generation relies on Verilog initial blocks with forever loops. The clk1 signal toggles with a period of 100 microseconds, which corresponds to 10 kHz. The clk2 signal toggles with a period of 10 nanoseconds, which corresponds to 100 MHz. Reset assertion establishes a known startup condition for all registers. The testbench loads stimulus data from hexadecimal files into memory arrays through the $readmemh system task.

Coefficient initialization occurs before data processing. A for-loop steps through sixty-four address locations, where each step asserts the cload signal, places the target address on caddr, and drives the coefficient value on cin. This procedure runs entirely in the clk2 domain. After initialization finishes, the testbench begins presenting input samples. Input timing aligns with clk1 edges, and the valid_in signal is asserted for one clk1 cycle for each sample.

C. Logic Synthesis

Synthesis targets the IBM 130 nm scx3_cmos8rf_lpvt standard cell library. The synthesis script loads the required library files using the common.tcl configuration. Verilog source files for all design modules are provided to Synopsys Design Compiler. The tool then carries out elaboration, linking, and uniquification steps to construct the full design hierarchy.

D. Gate-Level Simulation

Post-synthesis verification replaces the RTL description with the synthesized gate-level netlist while keeping the original testbench unchanged. Functional simulation uses Verilog models from the IBM technology library to represent standard cells accurately. The simulation command specifies the SDF file path and design hierarchy through the -sdftyp option to apply timing information. Setup and hold checks verify correct temporal behavior under annotated delays. The simulation shows that all generated outputs match the expected reference values, while any mismatch would point to synthesis-related issues or timing violations.

V. RESULTS AND ANALYSIS

A. Time Analysis

Static timing analysis identifies the critical path within the design. This path starts at the tap counter register and ends at the accumulator register. Memory read access requires 2.93 nanoseconds, the multiplier contributes an additional 1.11 nanoseconds, and the accumulator adder chain introduces 5.57 nanoseconds of delay. The total data arrival time therefore reaches 9.61 nanoseconds. A clock period constraint of 10 nanoseconds leaves 0.38 nanoseconds for library setup time, which gives a data required time of 9.62 nanoseconds. A positive slack of 0.01 nanoseconds confirms that timing requirements are met, and this narrow margin reflects operation near the highest feasible frequency for the selected architecture. The clk1 domain shows a much larger timing margin, where the longest path passes through the synchronizer holding registers and completes in 1.25 nanoseconds relative to a 100-microsecond clock period, resulting in slack greater than 99,998 nanoseconds.

B. Area Report

The total cell area is measured as 127,620 square micrometers. Conversion to square millimeters uses the technology scaling factor. The 130 nm process corresponds to a feature size of 0.13 micrometers. The area in square millimeters is calculated as 127,620 × (0.13 × 10^−3)^2, which equals 0.002157 mm². This small footprint follows from the use of a sequential processing architecture, while a fully parallel design would need sixty-four multipliers and would occupy a much larger silicon area.

C. Power Analysis

Initial synthesis power estimation reports a value of 3.52 milliwatts without using switching activity information. This estimate reflects a coarse approximation derived from default activity assumptions. Gate-level simulation generates a VCD file that captures actual signal transitions during operation. PrimeTime reads this VCD data and applies measured toggle rates to all nets in the design.

Time-based power analysis computes average power across the simulated interval. The measured total power consumption equals 3.406 milliwatts. Internal cell power contributes 3.405 milliwatts, which represents 99.98 percent of the total. Net switching power adds 0.531 microwatts, corresponding to 0.02 percent. Cell leakage power contributes 0.155 microwatts and remains negligible. The small switching component results from limited VCD signal coverage caused by file size limits.

Energy per processed sample is obtained from the ratio of power to throughput. Using a total power of 3.406 × 10^−3 watts and a sample rate of 10,000 samples per second yields 340.6 picojoules per sample.

VI. SUMMARY

TABLE I - PERFORMANCE METRICS

Metric	Value
Technology	IBM 130nm
Input Format	16-bit Q1.15
Output Format	16-bit Q7.9
Filter Taps	64
Input Clock	10 kHz
Core Clock	100 MHz
Throughput	10 kS/s
Critical Path	9.61 ns
Timing Slack	0.01 ns
Total Area	0.00216 mm²
Total Power	3.406 mW
Energy/Sample	340.6 pJ
Accuracy (NRMSE)	0%

The throughput of 10 kS/s matches the applied input sampling rate, and each input sample produces one output after completion of sixty-four multiply–accumulate operations together with a small amount of control overhead. Operation at a maximum clock frequency of 100 MHz supports this processing rate and leaves sufficient computational margin. Area efficiency arises from resource sharing, since a single multiplier–accumulator unit performs all sixty-four tap calculations in a sequential manner, whereas parallel architectures reach higher throughput at the cost of significantly increased silicon area. This approach favors reduced area while requiring multiple clock cycles to generate each output. Power consumption remains moderate due to the use of 130 nm technology and a controlled clock frequency, and the sequential structure limits simultaneous switching activity across the datapath.

VII. CONCLUSION

This work describes a full RTL-to-GDSII implementation of a 64-tap FIR filter using IBM 130 nm technology. The design balances silicon area, power consumption, and processing performance by using a sequential processing architecture. Fixed-point computation with Q1.15 and Q7.9 formats preserves numerical precision during filtering operations. Operation across two clock domains separates low-rate input sampling from high-speed internal processing.

REFERENCES

[1] Columbia University, "CSEE W4823 Advanced Logic Design Laboratory Materials," Fall 2025.