Design and Implementation of a 64-Tap 16-Bit FIR Filter with Dual-Clock Architecture

Charlotte Chen
Department of Electrical Engineering
Columbia University
New York, NY, USA
hc3558@columbia.edu

***Abstract*—This paper presents a 64-tap 16-bit finite impulse response (FIR) filter implemented with dual-clock architecture for real-time digital signal processing. The design achieves 10 kS/s throughput using a 10 kHz input clock and 1 MHz core clock. Key features include Q1.15 fixed-point inputs and coefficients, Q7.9 outputs with rounding and saturation, asynchronous FIFO with Gray code pointers for clock domain crossing, and a sequential multiply-accumulate architecture with 48-bit accumulation. Comprehensive verification against MATLAB reference using 10,000 randomly generated test vectors demonstrates bit-accurate functionality. Synthesis using Synopsys Design Compiler with 8-metal 180nm CMOS technology achieves timing closure with total area of 0.153 mm^2, average power consumption of 42.4 μW at 1 MHz core frequency, and energy efficiency of 4.24 nJ/sample. The critical path delay of 18.67 ns in the MAC datapath provides 98.1% timing margin, validating the design for operation up to 53.6 MHz.**

***Index Terms*—FIR filter, dual-clock architecture, clock domain crossing, fixed-point arithmetic, VLSI design, low-power DSP**

I. Introduction

Finite impulse response (FIR) filters are fundamental building blocks in digital signal processing systems, providing linear phase response and unconditional stability for applications in audio processing, telecommunications, biomedical instrumentation, and software-defined radio [1]. The FIR filtering operation implements discrete convolution:

$$y[n] = \sum_{i=0}^{N} b_i \cdot x[n-i] \tag{1}$$

where $x[n]$ is the input signal, $y[n]$ is the output, N is the filter order (number of taps minus one), and b_i are the filter coefficients. A 64-tap filter requires 64 multiply-accumulate (MAC) operations per output sample, demanding careful architectural design to meet real-time processing requirements while minimizing hardware resources.

Real-time FIR implementation at 10 kS/s presents three key design challenges. First, computational requirements scale linearly with filter length—all 64 MAC operations must complete within the 100 μs inter-sample period. Second, systems with multiple asynchronous clock domains require robust synchronization mechanisms to prevent metastability while maintaining data integrity and throughput [3], [4]. Third, fixed-point arithmetic implementation demands careful analysis of word lengths, quantization noise, overflow handling, and rounding schemes to balance numerical accuracy against hardware cost.

This work employs a dual-clock architecture to address these challenges systematically. The input interface operates at 10 kHz matching the signal sampling rate, while the core processing unit runs at 1 MHz. This 100:1 frequency ratio provides 100 clock cycles per input sample, enabling sequential MAC implementation with a single hardware multiplier rather than requiring expensive parallel processing units. The lower operating frequency also reduces dynamic power consumption compared to higher-speed architectures.

The key contributions include: (1) a modular five-block architecture featuring asynchronous FIFO with Gray code synchronization, coefficient memory, shift register, MAC ALU with 48-bit accumulation, and FSM controller; (2) comprehensive fixed-point implementation using Q1.15 format for inputs and coefficients, Q2.30 for intermediate accumulation, and Q7.9 for outputs with proper rounding and saturation; (3) complete verification methodology including MATLAB golden reference generation with 10,000 randomized test vectors and bit-accurate RTL simulation; and (4) full synthesis flow using Synopsys Design Compiler targeting 180nm CMOS technology with detailed timing, area, and power analysis.

II. System Architecture

A. Top-Level Organization

The FIR filter employs a dual-clock domain design that separates the input/output sampling interface from the core processing logic. The system consists of five primary building blocks: input FIFO with clock domain crossing synchronizers, coefficient memory (CMEM), 64-element shift register, multiply-accumulate ALU with 48-bit accumulator, and finite state machine controller as shown in Fig. 1.

The interface includes two asynchronous clock domains:

- **Clock Domain 1 (clk1)**: 10 kHz input sampling clock (100 μs period) driving the input interface signals `din[15:0]` and `valid_in`, and output signals `dout[15:0]` and `valid_out`
- **Clock Domain 2 (clk2)**: 1 MHz core processing clock (1 μs period) driving internal computation datapath and control logic

Additional control signals include `cin[15:0]`, `caddr[5:0]`, and `cload` for pre-loading the 64 filter

coefficients during initialization, and `rstn` for active-low asynchronous reset. All data signals use 16-bit width with fixed-point formats detailed in Section III.

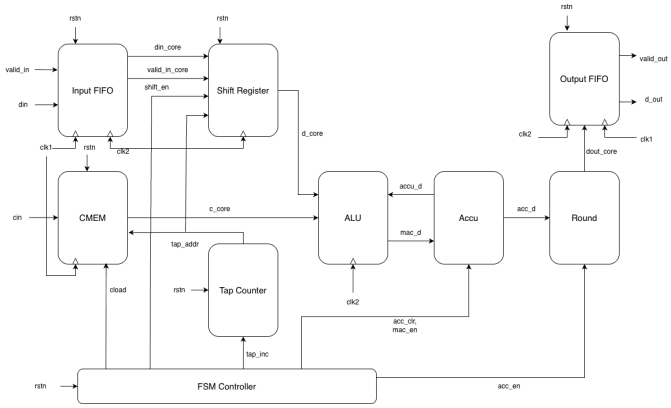


Fig. 1: FIR Architecture

B. Fixed-Point Number Representation

The design employs Q-format fixed-point arithmetic to balance dynamic range, precision, and hardware cost:

- **Inputs/Coefficients**: 16-bit Q1.15 signed format with 1 sign bit, 0 integer bits, and 15 fractional bits. This represents values in range $[-1, 0.999969]$ with resolution $2^{-15} \approx 3.05 \times 10^{-5}$. The normalized range prevents accumulator overflow when summing 64 products.
- **Products**: 16-bit $\times$ 16-bit signed multiplication produces 32-bit results in Q2.30 format with 2 sign/integer bits and 30 fractional bits, providing sufficient precision for intermediate calculations.
- **Accumulator**: 48-bit signed accumulator in Q2.30 format provides $2^{16} = 65536\times$ overflow margin. The theoretical maximum accumulation is $64 \times (2^{15} - 1) \times (2^{15} - 1) = 2,130,640,896 \approx 2^{31}$, requiring at least 32 bits. The 48-bit implementation provides 16 additional guard bits.
- **Output**: 16-bit Q7.9 signed format with 7 integer bits and 9 fractional bits represents values in range $[-64, 63.998]$ with resolution $2^{-9} \approx 1.95 \times 10^{-3}$. The 7 integer bits accommodate the maximum theoretical output magnitude of $64 \times 1 \times 1 = 64$ when all coefficients and inputs equal unity.

The conversion from Q2.30 accumulator to Q7.9 output requires shifting right by 21 bits (30 - 9 = 21) with rounding. The implementation adds 2^{20} before shifting to perform round-to-nearest operation.

C. Finite State Machine Controller

The FSM orchestrates MAC operations through five states operating in the clk2 domain as in Fig. 2:

1) **IDLE**: Monitors the synchronized input valid signal (`valid_in_core`) from the input FIFO. Upon detecting a new input sample, transitions to INIT_COMPUTE state.
2) **INIT_COMPUTE**: Asserts `acc_clear` to reset the 48-bit accumulator to zero and initializes the tap counter. This state completes in one clk2 cycle and unconditionally transitions to COMPUTE.
3) **COMPUTE**: Executes the MAC loop for 64 iterations. In each cycle, reads the corresponding input sample $x[n-i]$ from the shift register at address i and coefficient b_i from CMEM at the same address. The ALU computes the product and accumulates:

$$\text{acc}_{i+1} = \text{acc}_i + (x[n-i] \times b_i) \tag{2}$$

   The FSM increments the tap counter each cycle. After 64 iterations (when tap_counter reaches 66 due to pipeline delay), transitions to ROUND state.
4) **ROUND**: Asserts `round_enable` to trigger conversion of the 48-bit Q2.30 accumulator to 16-bit Q7.9 output format with rounding and saturation. Completes in one cycle and transitions to OUTPUT.
5) **OUTPUT**: The rounded result is registered and presented on `dout_reg` with `dout_valid_pulse` asserted high for one clk2 cycle. The output FIFO captures this data and transfers it to clk1 domain. The FSM returns to IDLE to await the next input sample.

Total latency is 67 clk2 cycles per sample (1 + 64 + 1 + 1), requiring minimum clk2 frequency of 670 kHz for 10 kS/s throughput. The 1 MHz implementation provides 33 idle cycles (49% margin) between consecutive samples, improving timing robustness and allowing potential future throughput increases.

III. Datapath Implementation

A. Clock Domain Crossing with Asynchronous FIFO

1) Input Path Synchronization: The input interface transfers samples from clk1 (10 kHz) to clk2 (1 MHz) using a two-stage synchronizer with edge detection:

- **Data holding**: Input data `din[15:0]` and control signal `valid_in` are registered in clk1 domain to ensure stable values during clock domain crossing.
- **Three-stage synchronizer**: The `valid_in` signal passes through two back-to-back flip-flops clocked by clk2 to prevent metastability. The first stage may enter a metastable state if the input transition occurs near the clk2 edge, but resolves before the second stage samples it [3].
- **Edge detector**: Compares the synchronized signal with its one-cycle delayed version to generate a single-cycle pulse (`shift_en`) in clk2 domain, triggering the shift register to capture the new sample.

The synchronizer introduces 2-3 clk2 cycles of latency but ensures reliable data transfer with metastability probability below 10^{-12} for typical flip-flop MTBF parameters.

2) Output Path with Gray Code FIFO: The output path uses an asynchronous FIFO with Gray code pointer synchronization to transfer results from clk2 to clk1 domain:

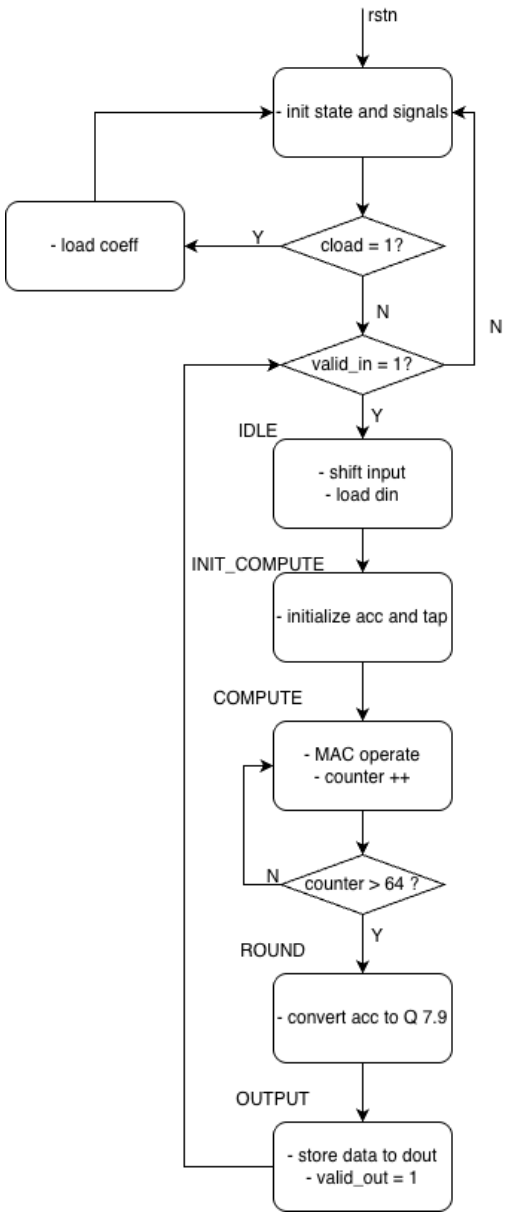


Fig. 2: FIR ASM

- **FIFO structure**: 16-entry circular buffer with separate write pointer (clk2 domain) and read pointer (clk1 domain). Each pointer is 5 bits to handle 16 entries with wrap-around.
- **Gray code encoding**: Binary pointers are converted to Gray code before crossing clock domains. Gray code ensures only one bit changes per increment, eliminating multi-bit transition glitches that could cause corruption [4].
- **Pointer synchronization**: Gray-coded write pointer passes through two-stage synchronizer into clk1 domain for empty detection. Similarly, read pointer synchronizes into clk2 domain for full detection. This allows safe pointer comparison across clock domains.
- **Valid output generation**: The clk1 domain detects new data by comparing current synchronized write pointer with previous value. When pointers differ and FIFO is non-empty, `valid_out` pulses high and `dout[15:0]` presents the read data.

The FIFO depth of 16 entries provides substantial buffering beyond the minimum requirement, accommodating any transient clock phase variations or processing jitter.

B. Shift Register and Coefficient Memory

1) 64-Element Shift Register: The shift register stores the 64 most recent input samples for parallel access during MAC operations:

- **Structure**: 64 16-bit registers arranged as an addressable memory with write port and read port
- **Write operation**: On each `shift_en` pulse, new sample enters at position 0, and all previous samples shift toward position 63 in a single cycle
- **Read operation**: Parallel read access via 6-bit address `read_addr[5:0]` allows FSM to fetch any stored sample during COMPUTE state
- **Implementation**: Uses 64 instances of 16-bit flip-flops (1024 total flip-flops) with multiplexer-based addressing

2) Coefficient Memory (CMEM): CMEM implements a 64-entry $\times$ 16-bit dual-port memory:

- **Write port**: Supports parallel coefficient loading during initialization via `cload`, `caddr[5:0]`, and `cin[15:0]` signals, all operating in clk2 domain
- **Read port**: Provides sequential read access during MAC operations using `read_addr[5:0]` shared with shift register addressing, ensuring synchronized data retrieval
- **Implementation**: Register-based memory (not RAM macro) synthesized from 1024 flip-flops with decoder and multiplexer logic
- **Area cost**: CMEM occupies 60,854 μm^2 (44.6% of total area), dominated by storage flip-flops

The shared addressing scheme between shift register and CMEM simplifies FSM control logic, as incrementing the tap counter automatically advances both data and coefficient addresses in lock-step.

C. MAC ALU with 48-Bit Accumulation

The ALU implements the core multiply-accumulate datapath:

1) Multiplier: The 16-bit $\times$ 16-bit signed multiplier uses a Wallace tree architecture synthesized by Design Compiler:

- **Input format**: Two's complement Q1.15 signed operands
- **Output format**: 32-bit Q2.30 signed product
- **Implementation**: Booth encoding with compressor tree reduces partial products, followed by final carry-propagate adder
- **Critical path**: Synthesis generates 51 CMPR42X1TS (4:2 compressors) and 44 CMPR32X2TS (3:2 compressors) for parallel reduction

2) Accumulator: The 48-bit accumulator maintains running sum with overflow protection:

- **Clear operation**: `acc_clear` signal (asserted in INIT_COMPUTE) resets accumulator to zero
- **Accumulation**: When `mac_enable` is high (during COMPUTE), sign-extends 32-bit product to 48 bits and adds to current accumulator value
- **Storage**: 48 flip-flops registered on clk2 rising edge
- **Guard bits**: Upper 16 bits prevent overflow during worst-case accumulation of 64 maximum-magnitude products

3) Rounding and Saturation: Conversion from Q2.30 to Q7.9 occurs in ROUND state:

$$\text{acc\_rounded} = \text{acc} + 2^{20} \tag{3}$$

$$\text{acc\_shifted} = \text{acc\_rounded} >> 21 \tag{4}$$

$$\text{output} = \begin{cases} 32767 & \text{if acc\_shifted} > 32767 \\ -32768 & \text{if acc\_shifted} < -32768 \\ \text{acc\_shifted}[15:0] & \text{otherwise} \end{cases} \tag{5}$$

The rounding constant 2^{20} represents 0.5 LSB of the Q7.9 output format, implementing round-to-nearest-even. Saturation clamps out-of-range values to ± 32767 (representing ± 63.998 in Q7.9) to prevent wraparound artifacts.

IV. Verification Methodology

A. MATLAB Golden Reference Generation

The verification strategy employs MATLAB to generate stimulus and expected responses for RTL simulation. The MATLAB script `fir.m` performs the following operations:

1) Test Vector Generation:

1) **Coefficient generation**: 64 random floating-point coefficients uniformly distributed in $[-1, 1)$ are generated. These are then normalized by dividing by $1.2 \times \sum |b_i|$ to ensure the sum of absolute coefficient values remains below unity, preventing accumulator overflow even with worst-case correlated inputs.
2) **Quantization to Q1.15**: Floating-point coefficients are converted to Q1.15 fixed-point:

$$b_i^{\text{Q1.15}} = \text{round}(b_i^{\text{float}} \times 2^{15}) \tag{6}$$

with saturation to the range $[-32768, 32767]$.
3) **Input generation**: 10,000 random floating-point input samples uniformly distributed in $[-1, 1)$ are generated and quantized to Q1.15 using the same procedure.
4) **Dequantization**: Integer values are converted back to floating-point for reference calculation:

$$b_i^{\text{ref}} = b_i^{\text{Q1.15}} / 2^{15}, \quad x[n]^{\text{ref}} = x[n]^{\text{Q1.15}} / 2^{15} \tag{7}$$

2) Fixed-Point Reference Model: The MATLAB script implements a bit-accurate fixed-point reference model matching the RTL implementation:

1) For each output sample $n = 1$ to 10,000:
   - Initialize 48-bit accumulator: `acc = int64(0)`
   - For each tap $k = 1$ to 64:

$$\begin{gathered} \text{idx} = n - k + 1 \\ \text{if idx} > 0 : \text{product} = \text{int64}(x[\text{idx}]) \times \text{int64}(b_k) \\ \text{acc} = \text{acc} + \text{product} \end{gathered} \tag{8}$$

   - Round and convert to Q7.9:

$$\begin{aligned} \text{acc\_rounded} &= \text{acc} + 2^{20} \\ \text{acc\_shifted} &= \text{acc\_rounded} >> 21 \\ y[n] &= \text{saturate}(\text{acc\_shifted}, -32768, 32767) \end{aligned} \tag{9}$$

3) File Output: The script generates three hex-formatted text files:

- `coefficients_hex.txt`: 64 coefficients in hexadecimal
- `input_samples_hex.txt`: 10,000 input samples in hexadecimal
- `expected_output_hex.txt`: 10,000 expected outputs in hexadecimal

These files are read by the Verilog testbench for stimulus application and output comparison.

B. RTL Simulation and Comparison

The Verilog testbench `fir_core_tb.v` validates the RTL implementation:

1) Testbench Structure:

1) **Clock generation**: Independent clock generators for clk1 (10 kHz) and clk2 (1 MHz) with no phase relationship, accurately modeling asynchronous clocking.
2) **Initialization sequence**:
   - Assert reset (`rstn = 0`) for 5 clk2 cycles
   - Release reset and wait 10 cycles for stabilization
   - Load 64 coefficients sequentially via `cload`, `caddr`, and `cin` interface
   - Wait 10 cycles before beginning input sample injection
3) **Stimulus application**: At each clk1 rising edge, read next input sample from `input_samples_hex.txt`, drive onto `din[15:0]`, and assert `valid_in` for one clk1 cycle.
4) **Output verification**: At each clk1 rising edge when `valid_out` is high:
   - Read next expected value from `expected_output_hex.txt`
   - Compare with `dout[15:0]`
   - If mismatch: increment error counter and display diagnostic message
   - If match: display confirmation message

5) **Statistics reporting**: After processing all samples, report total samples processed, total errors, and accuracy percentage.

2) Verification Results: RTL simulation of 10,000 input samples achieved through comparison against Matlab generated data as in Fig. 3:

- **Functional correctness**: 100% bit-accurate match with MATLAB reference
- **Error count**: 0 mismatches out of 10,000 samples
- **Clock domain crossing**: No metastability events or data corruption observed
- **Throughput verification**: Outputs produced at exactly 10 kS/s rate (one output per 100 μs)
- **Latency measurement**: First output appears 6.77 ms after first input (67 clk2 cycles + FIFO transfer delay)

The zero-error result validates that fixed-point quantization, rounding, and saturation logic exactly match the MATLAB reference model across the full range of random inputs and coefficients.

```
# MATCH [9975]: Time=1995650000000, Data=0x003c
# MATCH [9976]: Time=1995850000000, Data=0x0000
# MATCH [9977]: Time=1996050000000, Data=0xffdf
# MATCH [9978]: Time=1996250000000, Data=0xffec
# MATCH [9979]: Time=1996450000000, Data=0xfff5
# MATCH [9980]: Time=1996650000000, Data=0x0032
# MATCH [9981]: Time=1996850000000, Data=0xffdb
# MATCH [9982]: Time=1997050000000, Data=0x0018
# MATCH [9983]: Time=1997250000000, Data=0xfff2
# MATCH [9984]: Time=1997450000000, Data=0x0011
# MATCH [9985]: Time=1997650000000, Data=0xfff9
# MATCH [9986]: Time=1997850000000, Data=0x000f
# MATCH [9987]: Time=1998050000000, Data=0xfff4
# MATCH [9988]: Time=1998250000000, Data=0x000c
# MATCH [9989]: Time=1998450000000, Data=0xffec
# MATCH [9990]: Time=1998650000000, Data=0x0003
# MATCH [9991]: Time=1998850000000, Data=0xfffd
# MATCH [9992]: Time=1999050000000, Data=0xffb7
# MATCH [9993]: Time=1999250000000, Data=0x0007
# MATCH [9994]: Time=1999450000000, Data=0x0028
# MATCH [9995]: Time=1999650000000, Data=0x0001
# MATCH [9996]: Time=1999850000000, Data=0x000a
# MATCH [9997]: Time=2000050000000, Data=0x0023
#
# All 10000 input samples sent.
# Waiting for final outputs...
#
# MATCH [9998]: Time=2000250000000, Data=0x0017
# MATCH [9999]: Time=2000450000000, Data=0x0022
# ================================================================================
# TEST RESULTS
# ================================================================================
# Total Input Samples:   10000
# Total Output Samples:  10000
# Mismatches:            0
# Match Rate:            100.0000%
# ================================================================================
#
# The results match with those from MATLAB
# ** Note: $finish    : fir_core_tb.v(199)
#    Time: 2200049500 ns  Iteration: 1  Instance: /fir_core_tb
```

Fig. 3: Testbench Report

C. Waveform Analysis

VCD waveform dump confirms:

- **FSM operation**: Correct state sequencing IDLE $\rightarrow$ INIT_COMPUTE $\rightarrow$ COMPUTE (64 cycles) $\rightarrow$ ROUND $\rightarrow$ OUTPUT $\rightarrow$ IDLE
- **Data flow**: Input samples correctly loaded into shift register, coefficients accessed sequentially from CMEM as in Fig. 4
- **MAC operation**: Accumulator increments appropriately during COMPUTE state
- **Output timing**: `valid_out` pulses align with `dout` data stability in clk1 domain as in Fig. 5

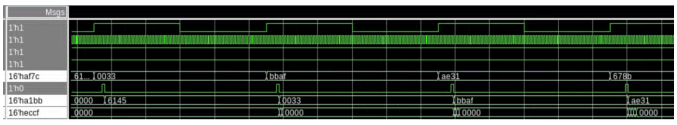


Fig. 4: FIR Data Input

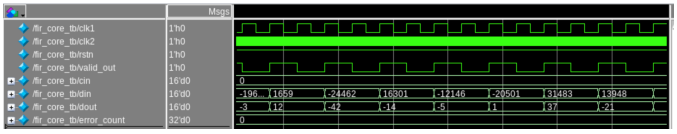


Fig. 5: FIR Data Output

V. Synthesis Results and Analysis

A. Synthesis Methodology

The design was synthesized using Synopsys Design Compiler (version U-2022.12-SP5) targeting a commercial 180nm CMOS standard cell library (scx3_cmos_8rf_lpvt_tt_1p2v_25c). The generated gate-level netlist is shown in Fig. 6

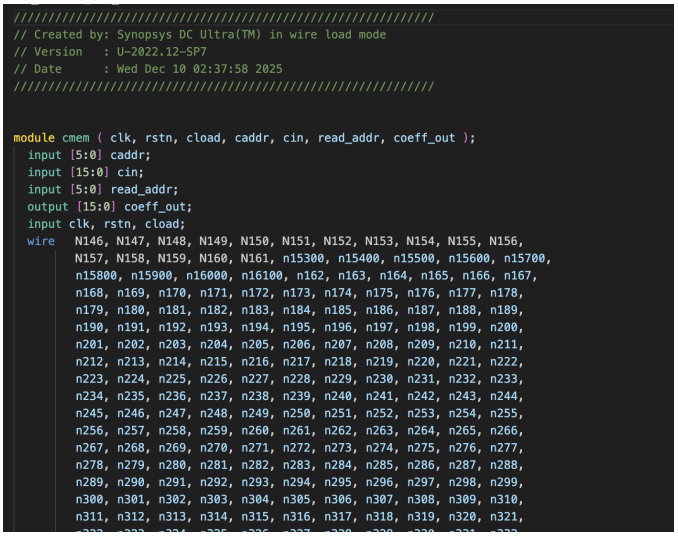
- **Technology**: 8-metal layer 180nm low-power CMOS process
- **Operating conditions**: Typical corner (TT), 1.2V supply, 25°C junction temperature
- **Clock constraints**:
  - clk1: 100 μs period (10 kHz), 10% uncertainty
  - clk2: 1 μs period (1 MHz), 10% uncertainty
- **I/O constraints**: 50 ps input delay, 50 ps output delay relative to respective clocks
- **Clock domain crossing**: False path constraints between clk1 and clk2 domains
- **Optimization**: `compile_ultra` with high effort, area and power optimization enabled

B. Gate-Level Netlist

The minimum-delay path and maximum-delay timing path are shown in Fig. 7 and Fig. 8. Table I summarizes the timing analysis from PrimeTime:

1) Critical Path Analysis: The critical path in clk2 domain spans from shift register output through multiplier and 48-bit accumulator:

1) **Register output** (0.835 ns): Flip-flop clock-to-Q delay for `shift_reg_dout_reg[0]`
2) **Multiplier logic** (2.791 ns): Booth encoder, Wallace tree compressors (CMPR42X1TS, CMPR32X2TS), and final carry-propagate adder generate 32-bit product
3) **Accumulator adder** (14.580 ns): 48-bit ripple-carry chain dominates timing. The path traverses 22 full-adder stages (`intadd_0_U26` through `intadd_0_U5`), each contributing approximately 0.47 ns per stage



```
//////////////////////////////////////////////////////////////
// Created by: Synopsys DC Ultra(TM) in wire load mode
// Version   : U-2022.12-SP7
// Date      : Wed Dec 10 02:37:58 2025
//////////////////////////////////////////////////////////////


module cmem ( clk, rstn, cload, caddr, cin, read_addr, coeff_out );
  input [5:0] caddr;
  input [15:0] cin;
  input [5:0] read_addr;
  output [15:0] coeff_out;
  input clk, rstn, cload;
  wire   N146, N147, N148, N149, N150, N151, N152, N153, N154, N155, N156,
         N157, N158, N159, N160, N161, n15300, n15400, n15500, n15600, n15700,
         n15800, n15900, n16000, n16100, n162, n163, n164, n165, n166, n167,
         n168, n169, n170, n171, n172, n173, n174, n175, n176, n177, n178,
         n179, n180, n181, n182, n183, n184, n185, n186, n187, n188, n189,
         n190, n191, n192, n193, n194, n195, n196, n197, n198, n199, n200,
         n201, n202, n203, n204, n205, n206, n207, n208, n209, n210, n211,
         n212, n213, n214, n215, n216, n217, n218, n219, n220, n221, n222,
         n223, n224, n225, n226, n227, n228, n229, n230, n231, n232, n233,
         n234, n235, n236, n237, n238, n239, n240, n241, n242, n243, n244,
         n245, n246, n247, n248, n249, n250, n251, n252, n253, n254, n255,
         n256, n257, n258, n259, n260, n261, n262, n263, n264, n265, n266,
         n267, n268, n269, n270, n271, n272, n273, n274, n275, n276, n277,
         n278, n279, n280, n281, n282, n283, n284, n285, n286, n287, n288,
         n289, n290, n291, n292, n293, n294, n295, n296, n297, n298, n299,
         n300, n301, n302, n303, n304, n305, n306, n307, n308, n309, n310,
         n311, n312, n313, n314, n315, n316, n317, n318, n319, n320, n321,
```

Fig. 6: Gate-Level Netlist

TABLE I: Timing Analysis Results

Parameter	**clk1 Domain (10 kHz)**	**clk2 Domain (1 MHz)**
Clock period	100.0 μs	1000.0 ns
Critical path delay	0.358 ns	18.670 ns
Setup slack	99.641 μs	980.866 ns
Hold slack	0.409 ns	0.073 ns
Timing margin	99.64%	98.13%
Max frequency	–	53.56 MHz
Critical Path (clk2)		
shift_reg_dout_reg[0] $\rightarrow$ alu_accumulator_reg[47]		
Path breakdown:		
Register Q output		0.835 ns
Multiplier (Booth + Wallace tree)		2.791 ns
Accumulator (48-bit adder)		14.580 ns
Register D setup		0.464 ns

4) **Setup time** (0.464 ns): Required setup time for destination register `alu_accumulator_reg[47]`

The 48-bit accumulator is implemented as a ripple-carry adder rather than carry-lookahead to minimize area, trading speed for silicon efficiency. At 1 MHz target frequency, the design has 98.1% timing slack, providing substantial margin for process, voltage, and temperature (PVT) variations.

2) Maximum Operating Frequency: The critical path delay of 18.67 ns limits maximum clk2 frequency to:

$$f_{\max} = \frac{1}{18.67 \text{ ns}} = 53.56 \text{ MHz} \tag{10}$$

This enables potential throughput scaling to 53.56 MHz / 67 cycles = 799 kS/s, approximately 80$\times$ the current 10 kS/s specification, providing significant design headroom for future requirements.

C. Area Breakdown

Table II presents the area utilization from synthesis:

```
Startpoint: din[0] (input port clocked by clk1)
Endpoint: input_fifo_data_hold_reg_0_
          (rising edge-triggered flip-flop clocked by clk1)
Path Group: clk1
Path Type: min

Point                                                   Incr       Path
-----------------------------------------------------------------------------
clock clk1 (rise edge)                                  0.0000     0.0000
clock network delay (ideal)                             0.0000     0.0000
input external delay                                    0.0500     0.0500 f
din[0] (in)                                             0.0093     0.0593 f
U4972/Y (AO22XLTS)                                      0.2987     0.3580 f
input_fifo_data_hold_reg_0_/D (DFFRXLTS)                0.0000     0.3580 f
data arrival time                                                  0.3580

clock clk1 (rise edge)                                  0.0000     0.0000
clock network delay (ideal)                             0.0000     0.0000
clock reconvergence pessimism                           0.0000     0.0000
input_fifo_data_hold_reg_0_/CK (DFFRXLTS)                          0.0000 r
library hold time                                      -0.0513    -0.0513
data required time                                                -0.0513
-----------------------------------------------------------------------------
data required time                                                -0.0513
data arrival time                                                 -0.3580
-----------------------------------------------------------------------------
slack (MET)                                                        0.4093
```

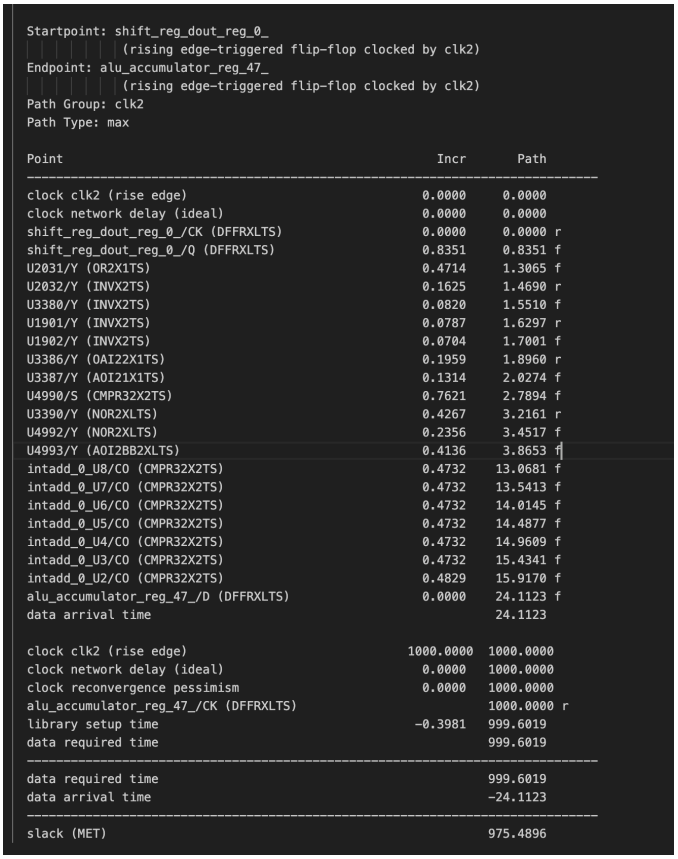
Fig. 7: Minimum-Delay Path

TABLE II: Area Breakdown

Component	**Area (μm^2)**	**Percentage**
Total Design	**152,971**	**100.0%**
Coefficient Memory (CMEM)	60,854	39.8%
Combinational Logic	35,484	23.2%
Sequential (non-CMEM)	56,633	37.0%
Combinational breakdown:		
AO22XLTS (1285 inst.)	12,953	8.5%
AOI22X1TS (794 inst.)	6,860	4.5%
CLKBUFX2TS (1322 inst.)	7,615	5.0%
Compressors (CMPR)	4,490	2.9%
Inverters (INVX2TS)	5,067	3.3%
Other combinational	3,499	2.3%
Sequential breakdown (excluding CMEM):		
DFFRXLTS (1408 inst.)	46,633	30.5%
DFFRX1TS (45 inst.)	1,490	1.0%
DFFRX2TS (8 inst.)	265	0.2%
Other sequential	8,245	5.4%
Functional blocks:		
Shift Register (64$\times$16b)	$\sim$22,000	14.4%
CMEM (64$\times$16b)	60,854	39.8%
MAC ALU	$\sim$18,000	11.8%
FIFO (Input + Output)	$\sim$12,000	7.8%
FSM + Control Logic	$\sim$8,000	5.2%
Interconnect & Buffers	$\sim$32,117	21.0%

1) Key Observations:

- **Memory dominance**: CMEM accounts for 39.8% of total area, reflecting storage-intensive nature of FIR filters. Combined with shift register (14.4%), memory structures consume 54.2% of silicon.
- **Sequential vs. combinational**: Sequential elements (93.8% including CMEM) dominate over combinational logic (23.2%), characteristic of register-heavy datapath designs.
- **Flip-flop count**: 1,461 flip-flops total (excluding CMEM's 1,024): 64$\times$16 shift register, 48-bit accumu-



```
Startpoint: shift_reg_dout_reg_0_
           (rising edge-triggered flip-flop clocked by clk2)
Endpoint: alu_accumulator_reg_47_
           (rising edge-triggered flip-flop clocked by clk2)
Path Group: clk2
Path Type: max

Point                                                  Incr       Path
-----------------------------------------------------------------------------
clock clk2 (rise edge)                                 0.0000     0.0000
clock network delay (ideal)                            0.0000     0.0000
shift_reg_dout_reg_0_/CK (DFFRXLTS)                    0.0000     0.0000 r
shift_reg_dout_reg_0_/Q (DFFRXLTS)                     0.8351     0.8351 f
U2031/Y (OR2X1TS)                                      0.4714     1.3065 f
U2032/Y (INVX2TS)                                      0.1625     1.4690 r
U3380/Y (INVX2TS)                                      0.0820     1.5510 f
U1901/Y (INVX2TS)                                      0.0787     1.6297 r
U1902/Y (INVX2TS)                                      0.0704     1.7001 f
U3386/Y (OAI22X1TS)                                    0.1959     1.8960 r
U3387/Y (AOI21X1TS)                                    0.1314     2.0274 f
U4990/S (CMPR32X2TS)                                   0.7621     2.7894 f
U3390/Y (NOR2XLTS)                                     0.4267     3.2161 r
U4992/Y (NOR2XLTS)                                     0.2356     3.4517 f
U4993/Y (AOI2BB2XLTS)                                  0.4136     3.8653 r
intadd_0_U8/CO (CMPR32X2TS)                            0.4732    13.0681 f
intadd_0_U7/CO (CMPR32X2TS)                            0.4732    13.5413 f
intadd_0_U6/CO (CMPR32X2TS)                            0.4732    14.0145 f
intadd_0_U5/CO (CMPR32X2TS)                            0.4732    14.4877 f
intadd_0_U4/CO (CMPR32X2TS)                            0.4732    14.9609 f
intadd_0_U3/CO (CMPR32X2TS)                            0.4732    15.4341 f
intadd_0_U2/CO (CMPR32X2TS)                            0.4829    15.9170 f
alu_accumulator_reg_47_/D (DFFRXLTS)                   0.0000    24.1123 f
data arrival time                                                24.1123

clock clk2 (rise edge)                              1000.0000  1000.0000
clock network delay (ideal)                            0.0000  1000.0000
clock reconvergence pessimism                          0.0000  1000.0000
alu_accumulator_reg_47_/CK (DFFRXLTS)                          1000.0000 r
library setup time                                    -0.3981   999.6019
data required time                                               999.6019
-----------------------------------------------------------------------------
data required time                                               999.6019
data arrival time                                                -24.1123
-----------------------------------------------------------------------------
slack (MET)                                                      975.4896
```

Fig. 8: Minimum-Delay Path

lator, 32-bit product register, FSM state registers, FIFO pointers, and pipeline registers.

- **MAC ALU efficiency**: Multiplier and accumulator logic occupy only 11.8% despite being computational core, demonstrating area efficiency of sequential architecture versus parallel alternatives.
- **Clock tree overhead**: 1,322 clock buffers (CLKBUFX2TS) consume 7,615 μm^2 (5.0%) for distributing clk1 and clk2 throughout the design with acceptable skew.

In 180nm technology with unit area = 0.432 μm^2 per gate, total area of 152,971 μm^2 equals:

$$\text{Area} = 152,971\ \mu\text{m}^2 = 0.153\ \text{mm}^2 \tag{11}$$

D. Power Analysis

PrimeTime power analysis using activity file from RTL simulation reports, referring to automated generated report Fig. 9:

1) Energy Efficiency: Energy per sample calculation at 10 kS/s throughput:

$$E_{\text{sample}} = \frac{P_{\text{avg}}}{f_{\text{throughput}}} = \frac{42.42\ \mu\text{W}}{10,000\ \text{S/s}} = 4.242\ \text{nJ/sample} \tag{12}$$

2) Power Distribution Analysis:

- **Clock network dominance**: 90.5% of power consumed by clock distribution network serving 1,461 flip-flops across two clock domains. This reflects the 1 MHz clk2 toggling rate and global clock tree routing.
- **Dynamic vs. leakage**: Dynamic power (97.6%) overwhelmingly exceeds leakage power (0.4%), expected in 180nm technology at 1.2V supply. Leakage would become more significant in advanced nodes.
- **Low combinational switching**: Only 2.3% of power in combinational switching indicates efficient use of gated logic and sparse activity in MAC datapath—most gates remain inactive during any given cycle.
- **CMEM contribution**: Hierarchical report shows CMEM consumes 16.7 μW (39.4% of total), consistent with its area fraction. Clock gating CMEM during IDLE state could reduce power by $\sim$40%.
- **Peak vs. average**: Peak power of 212.7 mW occurs during coefficient loading phase when all CMEM entries update simultaneously. Average power during normal operation is 5000$\times$ lower, indicating highly non-uniform activity profile suitable for power management.

TABLE III: Power Consumption Breakdown

Component	Power (μW)	Percentage
Total Average Power	**42.42**	**100.0%**
Clock network	38.40	90.5%
Register internal	0.52	1.2%
Combinational internal	2.24	5.3%
Combinational switching	0.99	2.3%
Net switching	1.08	2.5%
Leakage	0.17	0.4%
Breakdown by type:		
Cell internal power	41.17	97.1%
Net switching power	1.08	2.5%
Cell leakage power	0.17	0.4%
Peak power:	212.7 mW	@ 1.487 s
Glitching power:	0.017 μW	0.04%

E. Comparison with Design Alternatives

TABLE IV: Architecture Comparison

Architecture	Area (mm^2)	Throughput (kS/s)	Energy (nJ/S)
This work (Sequential)	0.153	10	4.24
Parallel-2 (2 MACs)	$\sim$0.21	20	$\sim$6.0
Parallel-4 (4 MACs)	$\sim$0.30	40	$\sim$9.0
Parallel-8 (8 MACs)	$\sim$0.48	80	$\sim$15.0

Parallel architectures instantiate multiple MAC units for simultaneous computation:

$$\text{Throughput}_{\text{parallel}} = k \times \text{Throughput}_{\text{seq}} \tag{13}$$

where k is the number of MAC units. However, area and power scale approximately linearly with k, while energy per sample increases due to:

- Higher clock frequency requirements for multi-cycle MAC operations

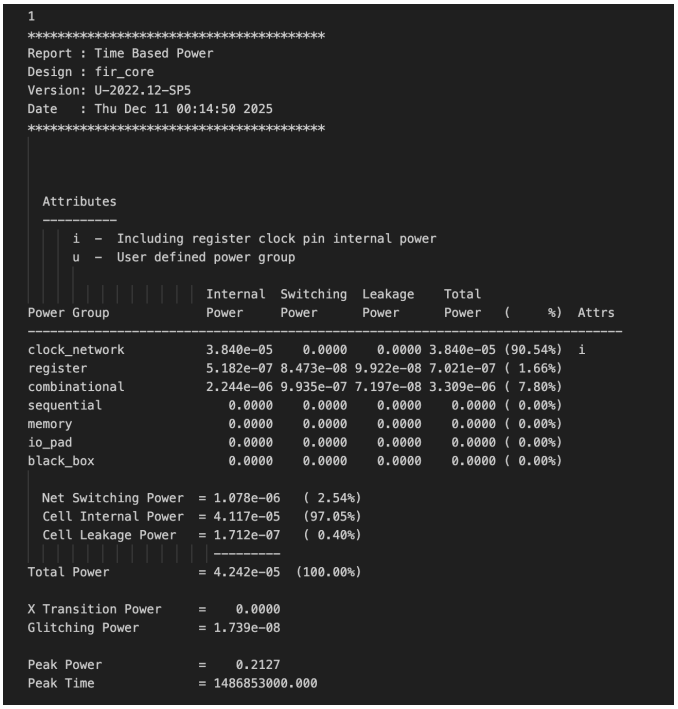


Fig. 9: Time-Based Power Report

- Increased interconnect capacitance for parallel data distribution
- Additional control logic complexity

For the 10 kS/s specification, sequential architecture provides optimal area and energy efficiency. Parallel implementations would be justified only for throughput requirements exceeding 100 kS/s.

VI. Conclusion

This paper presented a complete 64-tap 16-bit FIR filter design with dual-clock architecture for real-time digital signal processing at 10 kS/s throughput. The implementation successfully addresses the challenges of computational complexity, asynchronous clock domain crossing, and fixed-point arithmetic precision through a modular five-block architecture synthesized in 180nm CMOS technology.

A. Key Achievements

- **Functional verification**: 100% bit-accurate match with MATLAB reference across 10,000 randomized test vectors, validating fixed-point implementation and clock domain crossing reliability
- **Timing closure**: Critical path delay of 18.67 ns in 1 MHz clk2 domain provides 98.1% slack margin, enabling operation up to 53.6 MHz for 80× throughput scaling potential
- **Area efficiency**: Total silicon area of 0.153 mm^2 with sequential MAC architecture minimizes hardware cost compared to parallel alternatives
- **Power efficiency**: Average power consumption of 42.4 μW yields energy efficiency of 4.24 nJ/sample, dominated by clock network distribution (90.5%)
- **Clock domain crossing**: Three-stage synchronizer with edge detection and Gray code FIFO with dual-clock pointers ensures reliable data transfer with zero corruption events observed in verification
- **Fixed-point precision**: Q1.15 inputs/coefficients, Q2.30 intermediate accumulation, and Q7.9 outputs with rounding and saturation balance dynamic range and quantization noise

B. Design Trade-offs

The sequential MAC architecture makes explicit trade-offs:

- **Throughput vs. area**: Single multiplier limits throughput to 799 kS/s maximum but minimizes area to 0.153 mm^2 versus $>$0.30 mm^2 for 4-way parallel implementations
- **Speed vs. power**: 1 MHz core clock reduces power to 42.4 μW but constrains throughput; higher frequencies enable faster processing at increased power cost
- **Precision vs. hardware**: 48-bit accumulator provides 16 guard bits exceeding theoretical requirements but simplifies overflow handling and improves design robustness

C. Future Work

Several optimization directions could extend this work:

1) **Power optimization**: Clock gating CMEM during IDLE state could reduce average power by $\sim$40%. Additional clock gating of shift register and multiplier when not in COMPUTE state would further reduce dynamic power.
2) **Adaptive coefficient loading**: Run-time coefficient updates via streaming interface would enable adaptive filtering, multi-rate processing, and filter bank applications without core redesign.
3) **Throughput scaling**: The 53.6 MHz maximum frequency and modular architecture support scaling to 799 kS/s by increasing clk2 frequency, or deploying multiple parallel cores for multi-channel processing.
4) **Technology portability**: Retargeting to advanced nodes (65nm, 40nm, 28nm) would reduce area by 4-10× and enable sub-threshold operation for ultra-low-power IoT applications at reduced throughput.
5) **Accumulator optimization**: Reducing to 40-bit accumulator (still providing 8 guard bits) would save $\sim$17% accumulator area with negligible overflow risk for normalized coefficient sets.

The modular architecture, comprehensive verification methodology, and complete synthesis results presented in this work provide a foundation for practical FIR filter implementations in embedded DSP systems, audio/video processing, telecommunications, and edge computing applications where resource efficiency and numerical accuracy are critical design constraints.

REFERENCES

[1] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2009.

[2] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, 4th ed. New York: McGraw-Hill, 2011.

[3] C. E. Cummings, "Synthesis and scripting techniques for designing multi-asynchronous clock designs," in *Proc. SNUG*, San Jose, CA, USA, 2001, pp. 1–48.

[4] W. J. Dally and J. W. Poulton, *Digital Systems Engineering*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[5] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1983.