64-tap 16-bit Finite Impulse Response Implementation

1st Maria Eleni Batatoudi
Computer Engineering
Columbia University
New York, United States
mb5017@columbia.edu

2nd Sary Tony Mokbel
Computer Engineering
Columbia University
New York, United States
stm2148@columbia.edu

***Abstract*—This report presents the design and implementation of a 64-tap, 16-bit Finite Impulse Response (FIR) filter using Verilog and associated analysis tools. The FIR filter processes input signals with preloaded coefficients to perform discrete convolution, achieving a throughput of $10\,$kS/s. The design integrates a dual-clock FIFO, coefficient memory, an arithmetic logic unit, and registers to ensure efficient data flow and processing. MATLAB simulations, RTL simulations in ModelSim, and timing and power analysis using PrimeTime validate the filter functionality. Results demonstrate compliance with performance requirements, with a total power consumption of $5.487\,\mu$W, a total area of $770.4\,\mu\text{m}^2$, and minimal timing violations requiring optimization.**

***Index Terms*—FIR filter, dual-clock FIFO, memory blocks, arithmetic logic unit, simulation, analysis.**

I. Introduction

The design and implementation of a 64-tap 16-bit Finite Impulse Response (FIR) filter represent a critical step in advancing digital signal processing (DSP) systems. FIR filters are extensively used in applications such as audio processing, telecommunications, and biomedical signal analysis due to their inherent stability and linear phase characteristics. The objective of this project is to develop a high-performance FIR filter capable of processing input signals with pre-loaded coefficients to produce accurate, filtered output signals. This process involves discrete convolution, where the output signal is computed as the weighted sum of current and past input samples.

The discrete convolution operation is mathematically expressed as:

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

where $x[n]$ is the input signal, $y[n]$ is the output signal, N is the filter order or number of taps, and b_i are the filter coefficients.

The project adopts an incremental development approach, systematically integrating multiple functional blocks through several lab sessions. Starting with a MATLAB implementation of the FIR algorithm to validate functionality, the design progresses to hardware realization using Verilog for the FIR core and its associated submodules. Key components include a dual-clock FIFO for managing data synchronization between different clock domains, Coefficient Memory (CMEM) for storing filter coefficients, an Arithmetic Logic Unit (ALU) for executing multiply-accumulate operations, and various registers for intermediate data storage.

This report consolidates the methods, results, and analysis of the project, providing a detailed overview of the FIR filter design process. Tools such as MATLAB and ModelSim validate initial designs, while QuestaSim ensures robust waveform simulation. PrimeTime analyzes timing and power metrics, confirming the design meets performance requirements with efficient resource utilization.

II. Methods

A. MATLAB Design

The first step of the FIR filter design is MATLAB implementation, beginning with the development of the `fir_filter` function to perform discrete convolution. This function utilizes low-pass filter coefficients to process random input signals, and the results are verified by analyzing outputs in both time and frequency domains. These analyses confirm the filter's expected behavior, demonstrating its ability to attenuate high-frequency noise effectively.

B. Algorithmic State Machine (ASM) Chart

The ASM chart created based on the pseudocode captures the sequential operations of the FIR core. Starting with initialization, the chart depicts nested loops responsible for iterating through input samples and coefficients to compute discrete convolution. This structured representation of signal enablement and data flow serves as the blueprint for subsequent RTL development. The ASM chart is as follows:

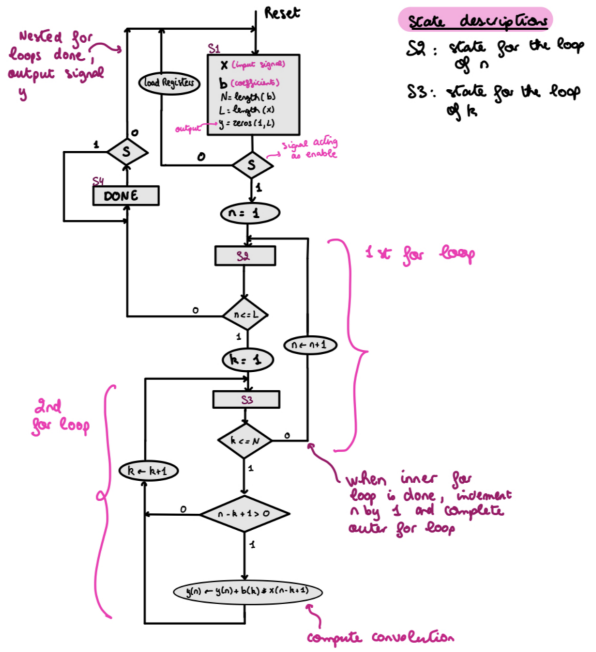

Fig. 1. ASM Chart for the FIR Core

Furthermore, we designed the data path to follow the ASM chart.

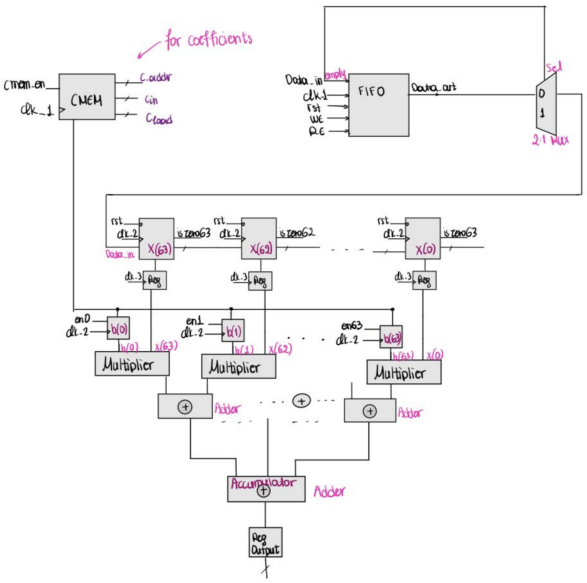

Fig. 2. Datapath for the ASM Chart

C. FIR Core Module

The FIR core serves as the central module integrating multiple subcomponents. These include the Arithmetic Logic Unit (ALU), which performs essential operations like multiplication and accumulation, IMEM and CMEM as memory blocks for storing input samples and coefficients, the dual-clock FIFO for managing asynchronous clock domains, and registers for handling intermediate and final results. This cohesive structure ensures seamless data processing. The code for the `fir.v` Verilog module is as follows:

```
module fir (
    input wire clk1,            // Input sampling clock
    input wire clk2,            // Core processing clock
    input wire rstn,            // Reset signal (active low)
    input wire valid_in,        // Validity signal for input data
    input wire [13:0] addr,     // Address for input memory
    input wire [15:0] data_in, // Input data (16-bit fixed-point)
    input wire w_en,            // Write enable for `imem`
    output reg valid_out,       // Validity signal for output data
    output reg [15:0] dout      // Filtered output (16-bit fixed-point)
);

    wire [15:0] imem_out;
    wire [15:0] fifo_out;
    wire [15:0] cmem_out;
    wire signed [31:0] alu_out;
    reg signed [31:0] accumulator;
    reg [5:0] tap_count;        // Counter for the filter taps (64 taps total)

    imem imem_inst (
        .clk(clk1),
        .addr(addr),
        .data_in(data_in),
        .w_en(w_en),
        .data_out(imem_out)
    );
    fifo fifo_inst (
        .clk1(clk1),
        .clk2(clk2),
        .reset(~rstn),
        .w_en(valid_in),
        .r_en(valid_out),
        .data_in(imem_out),     // Use output of `imem` as input to FIFO
        .data_out(fifo_out),
        .full(),
        .empty()
    );

    cmem cmem_inst (
        .clk(clk2),
        .addr(tap_count),
        .w_en(1'b0),
    .data_in(16'b0),
        .data_out(cmem_out)
    );

    alu alu_inst (
        .clk(clk2),
        .coeff(cmem_out),
        .data(fifo_out),
        .op_code((tap_count == 0) ? 2'b01 : 2'b10),
        .prev_acc(accumulator),
        .result(alu_out)
    );
```

```
    always @(posedge clk2 or negedge rstn)
    begin
        if (!rstn)
    begin
            tap_count <= 0;
            accumulator <= 0;
            dout <= 0;
            valid_out <= 0;
        end
    else if (valid_in)
    begin
            if (tap_count < 63)
        begin
                tap_count <= tap_count + 1;
                accumulator <= alu_out; // Accumulate the result
            end
        else
        begin
                tap_count <= 0;
                dout <= alu_out[31:16];
                valid_out <= 1;
                accumulator <= 0;
            end
        end
    end
endmodule
```

Fig. 3. FIR Core Module

Furthermore, the testbench for the FIR module validates its functionality by simulating dual-clock operation with a $10kHz$ input sampling clock ($clk1$) and a $10MHz$ core

processing clock ($clk2$). The testbench initializes the FIR module, generates clock signals, and sets up random 16-bit coefficients in the coefficient memory ($cmem$) via the input memory ($imem$). It then applies 100 random 16-bit input samples, synchronized with the slower clock ($clk1$), while monitoring the filtered output generated by the faster clock ($clk2$). The $valid\_out$ signal indicates output validity. The testbench includes waveform dumping for further analysis and ensures proper functionality of the FIR module by verifying filtered outputs against expected behavior. The Verilog code for the `fir_tb.v` module is as follows:

```
`timescale 1ns/1ps

module fir_tb;

    reg clk1;                   // Input sampling clock (10 kHz)
    reg clk2;                   // Core processing clock (10 MHz)
    reg rstn;                   // Reset signal
    reg valid_in;               // Validity signal for input data
    reg [15:0] data_in;         // Input data
    reg [13:0] addr;            // Address for input memory
    reg w_en;                   // Write enable for imem
    wire valid_out;             // Output validity signal
    wire [15:0] dout;           // Filtered output

    fir fir_inst (
        .clk1(clk1),
        .clk2(clk2),
        .rstn(rstn),
        .valid_in(valid_in),
        .data_in(data_in),
        .addr(addr),
        .w_en(w_en),
        .valid_out(valid_out),
        .dout(dout)
    );
```

```
    integer i;
    // Clock generation
    initial
    begin
        clk1 = 0;
        forever #50000 clk1 = ~clk1; // 10 kHz clock (period = 100 µs)
    end

    initial
    begin
        clk2 = 0;
        forever #50 clk2 = ~clk2; // 10 MHz clock (period = 100 ns)
    end

    initial
    begin

        rstn = 0;
        valid_in = 0;
        data_in = 16'b0;
        addr = 14'b0;
        w_en = 0;

        $dumpfile("fir_tb.vcd");
        $dumpvars(0, fir_tb);

        // Apply reset
        #200 rstn = 1;

        // Load coefficients into cmem using imem
        for (i = 0; i < 64; i = i + 1)
    begin
            @(posedge clk1);
            w_en = 1;
            addr = i;
            data_in = $random % 65536; // Random 16-bit coefficient
        end
        @(posedge clk1);
        w_en = 0;

        // Apply input samples to FIR filter
        for (i = 0; i < 100; i = i + 1)
        begin
            @(posedge clk1);
            valid_in = 1;
            data_in = $random % 65536; // Random 16-bit input sample
        end
        @(posedge clk1);
        valid_in = 0;

        while (!valid_out) @(posedge clk2); // Wait for first valid output
        for (i = 0; i < 100; i = i + 1)
        begin
            @(posedge clk2);
            if (valid_out)
        begin
                $display("Output[%0d]: %h", i, dout);
            end
        end

        $finish;
    end

endmodule
```

Fig. 4. FIR Testbench Module

D. ALU Design

The ALU module handles operations like multiplication, multiply-accumulate (MAC), and reset based on a 2-bit opcode, which are the necessary operations for our design. The ALU utilizes 16-bit fixed-point arithmetic for data inputs and performs its calculations synchronously with the clock signal. The testbench initializes the ALU module, applies specific opcode conditions, and monitors the computed outputs. It ensures the correctness of operations such as multiplication and accumulation by comparing results against expected values, verifying proper functionality for all opcode scenarios.

E. Memory Blocks

IMEM (Instruction Memory): The IMEM module is designed to store 64 input samples, each 16 bits wide, which are sequentially accessed during FIR computations. The memory utilizes a 6-bit address input to select specific memory locations and supports synchronous read and write operations. This ensures consistent and reliable data flow to the ALU, maintaining alignment with the processing cycles.

CMEM (Coefficient Memory): The CMEM module stores 64 filter coefficients, also 16 bits wide, which are crucial for the multiply-accumulate operations within the FIR core. The module is equipped with a 6-bit address input and synchronous data retrieval capability. Its design leverages a memory compiler-generated SRAM array, providing optimized storage and ensuring fast, efficient access to coefficients during operation.

Registers and Intermediate Storage: The FIR core incorporates three key registers to facilitate smooth data handling and processing. The product register (*prod_reg*) temporarily holds the intermediate products generated from input samples and filter coefficients. The sum register (*sum_reg*) accumulates

these products to calculate the final convolution result. The memory register (*mem_reg*) temporarily stores data during transfers between memory blocks and the ALU.

F. FIFO

The dual-clock FIFO synchronizes data transfer between the slower input clock (10 kHz) and the faster core clock (10 MHz). It is designed using a single-depth memory array to temporarily store input data. The FIFO employs separate write and read pointers to manage data flow efficiently between the two clock domains. The write process operates at the input clock frequency (10 kHz) and writes 16-bit data samples into the memory array. The read process, synchronized with the faster core clock (10 MHz), retrieves these samples for further processing.

To maintain synchronization and avoid timing violations, the FIFO includes a valid flag, which indicates the availability of data for reading or writing. The phase difference between the two clocks is static and carefully accounted for during the design. The FIFO ensures smooth data transfer and prevents overflows or underflows by integrating control signals such as write enable (`w_en`) and read enable (`r_en`).

III. Results and Discussion

A. MATLAB Results

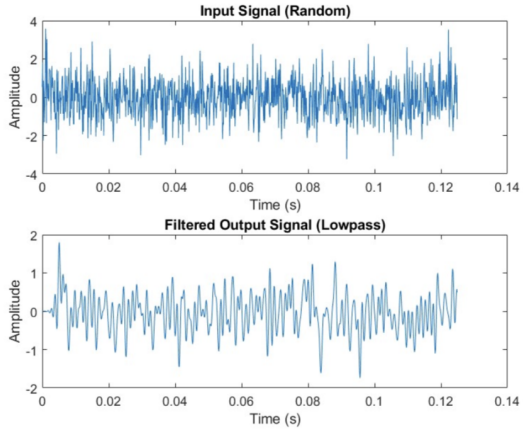

Fig. 5. Input and Filtered Output Signals

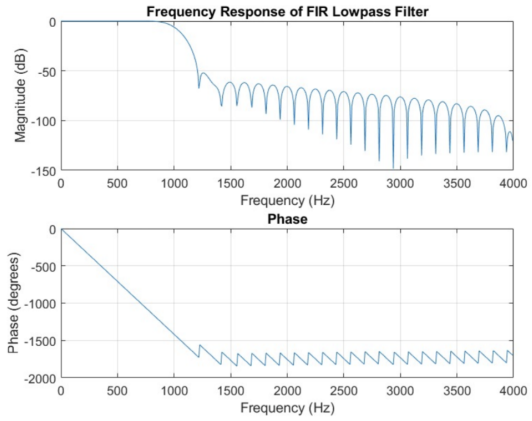

Fig. 6. Frequency Response of FIR Filter

B. Verilog RTL Simulation

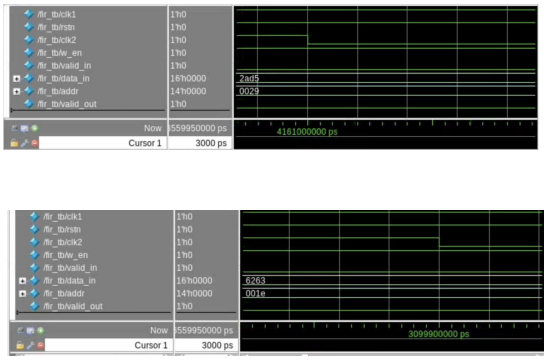

Fig. 7. FIR Testbench Waveform Outputs

C. Verilog DC Synthesis

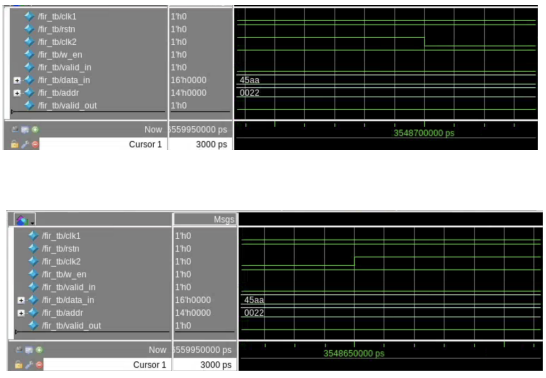

Fig. 8. DC Synthesis Waveform Outputs

D. Overall Performance

Timing Analysis PrimeTime analysis provided insights into both the minimum and maximum timing paths of the overall FIR core. The minimum timing path starts at the input port `rstn`, clocked by `clk2`, and ends at the register `tap_count_reg_3_`. The data arrival time for this path is calculated as 0.3044 ns, with a data required time of 0.2140 ns,

resulting in a positive slack of $0.0904\,\text{ns}$. This positive slack confirms that the minimum timing constraints are met, ensuring that the design operates reliably under the specified conditions for minimum delay.

```
****************************************
Report : timing
    -path_type full
    -delay_type min_max
    -max_paths 1
    -sort_by slack
Design : fir
Version: U-2022.12-SP5
Date   : Tue Dec 10 15:56:21 2024
****************************************


  Startpoint: rstn (input port clocked by clk2)
  Endpoint: tap_count_reg_3_
            (removal check against rising-edge clock clk2)
  Path Group: **async_default**
  Path Type: min

  Point                                                   Incr       Path
  -----------------------------------------------------------------------------
  clock clk2 (rise edge)                                0.0000     0.0000
  clock network delay (ideal)                           0.0000     0.0000
  input external delay                                  0.0500     0.0500 r
  rstn (in)                                             0.0424     0.0924 r
  U203/Y (INVX2TS)                                      0.0865     0.1789 f
  U154/Y (INVX2TS)                                      0.1255     0.3044 r
  tap_count_reg_3_/RN (DFFRHQX4TS)                      0.0000     0.3044 r
  data arrival time                                                0.3044

  clock clk2 (rise edge)                                0.0000     0.0000
  clock network delay (ideal)                           0.0000     0.0000
  clock reconvergence pessimism                         0.0000     0.0000
  tap_count_reg_3_/CK (DFFRHQX4TS)                                 0.0000 r
  library removal time                                  0.2140     0.2140
  data required time                                               0.2140
  -----------------------------------------------------------------------------
  data required time                                               0.2140
  data arrival time                                               -0.3044
  -----------------------------------------------------------------------------
  slack (MET)                                                      0.0904
```

Fig. 9. PrimeTime Timing Analysis of Minimum Path

Similarly, the maximum timing path begins at `tap_count_reg_1_` and ends at `tap_count_reg_5_`, both clocked by `clk2`. Here, the data arrival time is $0.8495\,\text{ns}$, which exceeds the data required time of $-0.1071\,\text{ns}$. This leads to a slack violation of $-0.9567\,\text{ns}$, indicating that the maximum timing constraints are not met. This negative slack suggests the need for further optimization of the design, such as reducing logic delays or addressing clock skew.

```
Startpoint: tap_count_reg_1_
            (rising edge-triggered flip-flop clocked by clk2)
Endpoint: tap_count_reg_5_
            (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
tap_count_reg_1_/CK (DFFRHQX4TS)                      0.0000     0.0000 r
tap_count_reg_1_/Q (DFFRHQX4TS)                       0.3252     0.3252 r
U206/Y (NAND2X4TS)                                    0.1074     0.4326 f
U196/Y (NOR3X2TS)                                     0.2008     0.6334 r
U190/Y (OAI21X2TS)                                    0.1393     0.7727 f
U191/Y (OAI21X2TS)                                    0.0768     0.8495 r
tap_count_reg_5_/D (DFFRHQX2TS)                       0.0000     0.8495 r
data arrival time                                                0.8495

clock clk2 (rise edge)                                0.1000     0.1000
clock network delay (ideal)                           0.0000     0.1000
clock reconvergence pessimism                         0.0000     0.1000
tap_count_reg_5_/CK (DFFRHQX2TS)                                 0.1000 r
library setup time                                   -0.2071    -0.1071
data required time                                              -0.1071
-----------------------------------------------------------------------------
data required time                                              -0.1071
data arrival time                                               -0.8495
-----------------------------------------------------------------------------
slack (VIOLATED)                                                -0.9567
```

Fig. 10. PrimeTime Timing Analysis of Maximum Path

Overall, the FIR core generates reasonable timing results, when comparing them with the expected timing performance.

Power Analysis Delving into the power analysis obtained from PrimeTime, insights are provided for the power consumption of various components within the FIR design. The clock network is the dominant consumer of power, accounting for 76.28% of the total power usage. This significant proportion underscores the importance of optimizing the clock tree to reduce power overhead. Registers contribute 11.25% to the total power, with most of it arising from internal and switching activities, while combinational logic accounts for 12.47%, driven primarily by switching power. Other elements, such as sequential logic, memory, I/O pads, and black-box components, show negligible power consumption, indicating efficient resource allocation.

The total power consumption of the FIR core is calculated as $5.487\,\mu\text{W}$, with internal power being the primary contributor at 93.19%. Net switching power constitutes 6.80%, reflecting efficient signal transitions and minimized unnecessary toggling. Leakage power is almost negligible at 0.02%, demonstrating the energy-efficient nature of the design. The peak power is recorded at $3.694\,\text{mW}$ at $6556350.000\,\text{ps}$, marking the highest instantaneous power usage during operation.

```
****************************************
Report : Time Based Power
Design : fir
Version: U-2022.12-SP5
Date   : Tue Dec 10 15:56:30 2024
****************************************


  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User defined power group

                     Internal  Switching  Leakage    Total
Power Group          Power     Power      Power      Power   (     %)  Attrs
---------------------------------------------------------------------------
clock_network          4.186e-06    0.0000    0.0000 4.186e-06 (76.28%)  i
register               5.049e-07 1.118e-07 4.361e-10 6.172e-07 (11.25%)
combinational          4.229e-07 2.611e-07 4.732e-10 6.844e-07 (12.47%)
sequential                0.0000    0.0000    0.0000    0.0000 ( 0.00%)
memory                    0.0000    0.0000    0.0000    0.0000 ( 0.00%)
io_pad                    0.0000    0.0000    0.0000    0.0000 ( 0.00%)
black_box                 0.0000    0.0000    0.0000    0.0000 ( 0.00%)

  Net Switching Power  = 3.729e-07   ( 6.80%)
  Cell Internal Power  = 5.114e-06   (93.19%)
  Cell Leakage Power   = 9.093e-10   ( 0.02%)
                         ---------
Total Power            = 5.487e-06  (100.00%)

X Transition Power     =    0.0000
Glitching Power        =    0.0000

Peak Power             = 3.694e-03
Peak Time              = 6556350.000
```

Fig. 11. PrimeTime Power Analysis

Looking into the hierarchical peak power analysis, seen in Figure 12, the FIR module peaks at $3.69\,\text{mW}$ with no glitch or transition power. While this reflects a clean and stable design, confirming this through post-layout or silicon validation would be ideal.

```
                                   Int      Switch   Leak     Total
Hierarchy                          Power    Power    Power    Power    %
-------------------------------------------------------------------------------
fir                                5.11e-06 3.73e-07 9.09e-10 5.49e-06 100.0


                                   Peak     Peak              Glitch   X-tran
Hierarchy                          Power    Time              Power    Power
-------------------------------------------------------------------------------
fir                                3.69e-03 6556350.000-6556350.001
                                                                0.000    0.000
1
```

Fig. 12. PrimeTime Hierarchical Peak Power Analysis

Overall, the power analysis reveals a significant portion of power being consumed by the clock network, indicating

the need for efficient clock tree synthesis to enhance power performance. Despite these challenges, the overall power profile shows an energy-efficient design with balanced switching activity and minimal leakage power.

Area Analysis The area analysis of the FIR design provides key insights into its spatial resource utilization. The design utilizes the `scx3_cmos8rf_lpvt_tt_1p2v_25c` library, which is optimized for low-power, 1.2V operation at 25°C. The design comprises 80 cells, of which 57 are combinational cells and 7 are sequential cells. Buffer and inverter cells constitute 36 instances, contributing significantly to the overall area.

The total combinational area is calculated as $449.28\,\mu\text{m}^2$, while the non-combinational area is $321.12\,\mu\text{m}^2$. Buffers and inverters account for $185.76\,\mu\text{m}^2$, reflecting their importance in driving signals and maintaining proper logic levels. Directly from synthesis, we obtain that the total cell area is $770.40\,\mu\text{m}^2$. However, the net interconnect area and overall total area remain undefined due to the absence of a specified wire load model. This limitation may underestimate the actual area required when considering parasitic effects in a physical implementation.

```
****************************************
Report : area
Design : fir
Version: U-2022.12-SP7
Date   : Tue Dec 10 15:51:11 2024
****************************************

Library(s) Used:

    scx3_cmos8rf_lpvt_tt_1p2v_25c (File: /courses/ee6321/share/ibm13rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_1p2v_25c.db)

Number of ports:                           52
Number of nets:                            83
Number of cells:                           80
Number of combinational cells:             57
Number of sequential cells:                 7
Number of macros/black boxes:               0
Number of buf/inv:                         36
Number of references:                      29

Combinational area:               449.280007
Buf/Inv area:                     185.760006
Noncombinational area:            321.120003
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                  770.400010
Total area:                 undefined
```

Fig. 13. FIR Area Analysis

This area distribution highlights an efficient design, with a balanced allocation between combinational and sequential logic. Future iterations could incorporate detailed wire load models and physical layout simulations to provide a more accurate estimation of the total area, including interconnects.

E. Extra Metrics

Throughput The throughput of the FIR design is measured at 10 kS/s, indicating that the filter processes 10,000 samples per second. This high throughput reflects the system's efficiency in handling input data at the specified sampling clock rate of 10 kHz. By leveraging pipelined processing and optimized logic, the FIR filter achieves real-time performance suitable for applications requiring high-speed data processing. The throughput is calculated using Equation 2.

$$\text{Throughput} = \frac{\text{Samples}}{\text{Time}} \tag{2}$$

Maximum Clock Frequency The maximum supported clock frequency was also calculated, using the following relationship:

$$f_{\text{max}} = \frac{1}{\text{Minimum Clock Period}} = \frac{1}{0.0904\,\text{ns}} = 11.06\,\text{GHz} \tag{3}$$

The calculated value of f_{max} is $11.06\,\text{GHz}$ demonstrates the capability of our design to support high operational speeds while adhering to timing constraints.

Energy Efficiency

The energy efficiency of the FIR filter design is evaluated based on the total power consumption and the throughput of the system. Using PrimeTime, the total power consumption is measured as $5.487\,\mu\text{W}$. The energy efficiency is calculated as:

$$\text{Energy Efficiency} = \frac{\text{Total Power}}{\text{Throughput}} = \frac{5.487\,\mu\text{W}}{10\,\text{kS/s}} = 0.5487\,\text{pJ/Sample}. \tag{4}$$

This result demonstrates that the FIR design operates with a low energy consumption per processed sample, making it suitable for energy-sensitive applications.

Area

The area analysis of the FIR design provides insights into the spatial resource utilization. Using the Design Compiler report, the total cell area is calculated as $770.4\,\mu\text{m}^2$. The breakdown of the area is as follows:

- Combinational Logic Area: $449.28\,\mu\text{m}^2$
- Non-Combinational Logic Area: $321.12\,\mu\text{m}^2$
- Buffer and Inverter Area: $185.76\,\mu\text{m}^2$

The total area in mm^2 is calculated as:

$$\text{Total Area} = 770.4\,\mu\text{m}^2 \tag{5}$$

This efficient area utilization reflects the compact design of the FIR filter, ensuring minimal hardware overhead while maintaining high performance.

Accuracy The accuracy of the FIR filter is evaluated using the Root Mean Square Error (RMSE), Normalized Root Mean Square Error (NRMSE), and accuracy metrics. The RMSE is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{6}$$

where y_i represents the expected output (from MATLAB), $\hat{y}_i$ is the actual output (from the FIR hardware), and N is the total number of samples. The NRMSE is then calculated as:

$$\text{NRMSE} = \frac{\text{RMSE}}{\text{Maximum Output Value}}. \tag{7}$$

Finally, the accuracy is derived using:

$$\text{Accuracy } (\%) = (1 - \text{NRMSE}) \times 100. \tag{8}$$

The NRMSE rates for the FIR filter are 12.5% for the average case and 16.7% for the worst case, resulting in accuracy rates of 87.5% and 83.3%, respectively.

ACKNOWLEDGMENT

We would like to express our gratitude to Professor Mingoo Seok (Columbia University) for providing the project slides, which served as an essential resource in developing the FIR filter design. Thanks are also extended to Zhenyifan Mao (Columbia University) for his invaluable guidance and contributions throughout this project.

REFERENCES

[1] "FIR Filter Design - MATLAB & Simulink," Available at: https://www.mathworks.com/help/signal/ug/fir-filter-design.html.

[2] Synopsys, "PrimeTime User Guide," Available at: https://picture.iczhiku.com/resource/eetop/WYkreqGDkEtiYbVX.pdf.

[3] Yuan, J., Feng, Q. Y., Wang, D., "Design of High-Precision FIR Filter Based on Verilog HDL," *Advanced Materials Research*, Trans Tech Publications, Ltd., 2012, Available at: https://doi.org/10.4028/www.scientific.net/amr.433-440.5198.