

64-Tap 16-bit Dual-Clock Fixed-to-Floating-Point Finite Impulse Response (FIR) Filter Design

Yuan Jiang
M.S. in Electrical Engineering
Columbia University
yj2848@columbia.edu

Hongxin Xu
M.S. in Electrical Engineering
Columbia University
hx2389@columbia.edu

Abstract—This paper presents the design and implementation of a 64-tap, 16-bit Fixed-to-Floating-Point Finite Impulse Response (FIR) Filter with a multi-clock architecture. The design incorporates a dual-clock FIFO for clock domain synchronization, an ALU with FX16-to-FP16 conversion, and a Finite State Machine (FSM) for control. The filter was implemented in Verilog RTL and verified using QuestaSim simulation. Performance metrics, including throughput, area, power, and accuracy, were evaluated against MATLAB results. The design achieves efficient resource utilization and meets timing requirements, demonstrating its suitability for real-time digital signal processing applications.

Keywords—FIR, MATLAB, Verilog, Fixed-to-Floating-Point, Dual-Clock FIFO, QuestaSim, ASM, PrimeTime, MAC

I. INTRODUCTION

A Finite Impulse Response (FIR) filter is a digital filter whose impulse response settles to a discrete value within a finite duration. FIR filters are widely employed in digital signal processing (DSP) applications, such as communication systems, audio processing, and control systems [1]. This paper focuses on the design and implementation of a 64-tap, 16-bit FIR filter, with particular emphasis on managing asynchronous clock domains using a dual-clock FIFO and performing convolution calculations with an Arithmetic Logic Unit (ALU).

The FIR filter is designed to process 16-bit fixed-point inputs (FX) at a data rate of 10kS/s, while producing 16-bit floating-point (FP) outputs with throughput 10kS/s. Filter coefficients are pre-loaded into a memory block generated by a memory compiler, with 64 coefficients stored as 2-byte values. Input data is stored in a shift register, capable of holding the 64 most recent samples, enabling real-time convolution. A dual-clock FIFO is employed to synchronize data transfer across asynchronous clock domains, where the input clock operates at clk1 (10 kHz), and the filter core processes data at clk2 (640 kHz).

The convolution operation, expressed as:

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

is performed by ALU, which efficiently handles both multiplication and addition operations. Additionally, the ALU is equipped with functionality for Fixed-to-Floating-Point (FX-to-FP) conversion, adhering to the IEEE754 half-precision floating-point standard (FP16) [2]. The control logic for the filter

is governed by a Finite State Machine (FSM), which coordinates data movement and ensures proper operation within the core.

II. MATLAB

To validate the functionality and performance of the FIR filter, MATLAB was used to generate random input signals and filter coefficients in a 16-bit fixed-point format. The input signals consist of 100 samples represented with 1 sign bit, 8 integer bits, and 7 fractional bits, while the 64 filter coefficients use 1 sign bit, 2 integer bits, and 13 fractional bits.

The script generates the binary representations of the inputs and coefficients, converts them into fixed-point values, and performs the convolution operation using a custom multiply-accumulate (MAC) implementation. Additionally, the fixed-point convolution results are converted to FP16 floating-point format following the IEEE754 half-precision standard.

The generated input signals, filter coefficients, and golden reference outputs (both fixed-point and floating-point) are exported to text files. These files serve as the input stimulus and expected outputs for the Verilog testbench. The outputs produced by the FIR filter design are compared against the MATLAB-generated golden results to ensure correctness and accuracy.

III. DESIGN METHODOLOGY

The design of the FIR filter begins with the development of an Algorithmic State Machine (ASM) [3], which serves as a structured method for translating the design into Verilog RTL. The process starts by formulating the pseudocode for the FIR filter, outlining the sequential operations and logic required to implement the desired functionality.

Using the pseudocode as a foundation, the ASM chart is constructed to represent the control and data flow of the design. From the ASM chart, the Datapath Block Diagram and the Control ASM Chart are derived, providing a clear separation between the data processing units and control logic. This structured approach ensures an organized and efficient implementation of the FIR filter in Verilog.

A. Pseudo Code

The following pseudocode describes the operation of a 64-tap FIR filter. The variable $y[n]$ represents the output of the filter at each step, while the inner loop calculates the multiply-accumulate (MAC) operation by summing the product of the input data samples $x[n-i]$ and the corresponding filter

coefficients $b[i]$. To avoid invalid operations, a conditional check ensures the index remains within bounds. The outer loop processes all input samples sequentially, and the inner loop iterates over the 64 coefficients for each input sample, generating the filtered output signal $y[n]$.

```

x_length = length(x)
N = 64
y[] = 0
For n = 1 to x_length do
    y[n] = 0
    For i = 1 to N do
        If (n - i) >= 0 then
            y[n] = y[n] + x[n - i] * b[i]
        End If
    End For
End For
Return y[]

```

B. ASM Chart

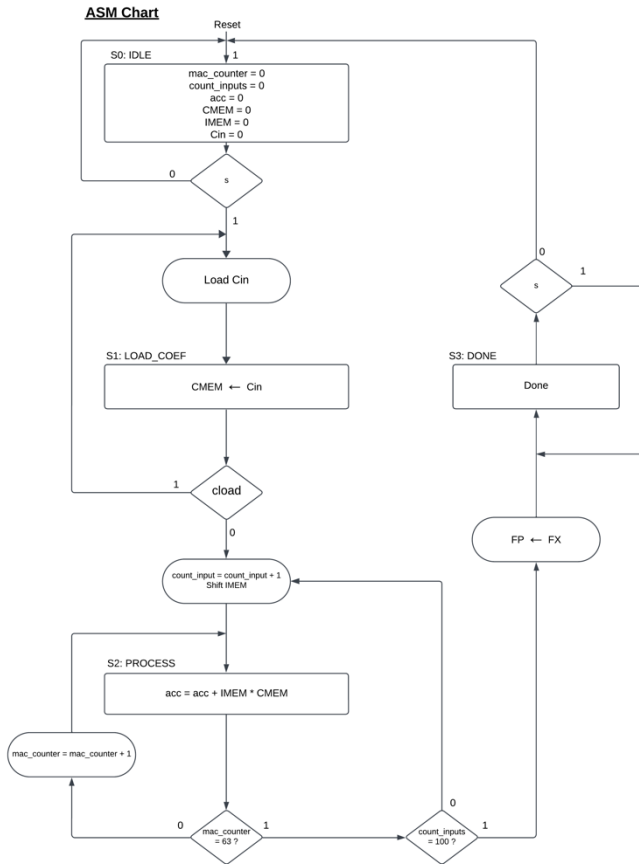


Fig.1 ASM Chart

The ASM chart in Fig. 1 consists of four states: IDLE, LOAD_COEF, PROCESS, and DONE. In the IDLE state, the system waits for a reset to complete and for the coefficient loading signal to be asserted. Upon activation, the system transitions to the LOAD_COEF state, where the 64 filter coefficients are sequentially loaded into memory. Once the

loading process is complete, the system enters the PROCESS state, where input data is fetched through a dual-clock FIFO, and the multiply-accumulate (MAC) operation is performed using the input data and coefficients. The results are converted to floating-point format and output sequentially after all 100 inputs. After processing all input samples and generating the required outputs, the system transitions to the DONE state, indicating the completion of the FIR filter operation.

C. Datapath Block Diagram

The diagram Fig. 2 illustrates the functional architecture of the 64-tap FIR filter, highlighting key modules and their interactions to achieve synchronized data processing and convolution operations. Input data (din) arrives at a rate of 10 kHz (controlled by $clk1$) and is first written into the FIFO, which serves as a synchronizer to bridge the asynchronous clock domains ($clk1$ and $clk2$). The data is subsequently read from the FIFO at 640 kHz ($clk2$) and passed to the Input Memory (IMEM_reg), implemented as a shift register to store the 64 most recent input samples. Simultaneously, the Coefficient Memory (CMEM), operating at $clk2$, provides preloaded coefficients (cin) required for convolution. These coefficients are accessed sequentially using the coefficient address ($caddr$) and supplied to the ALU.

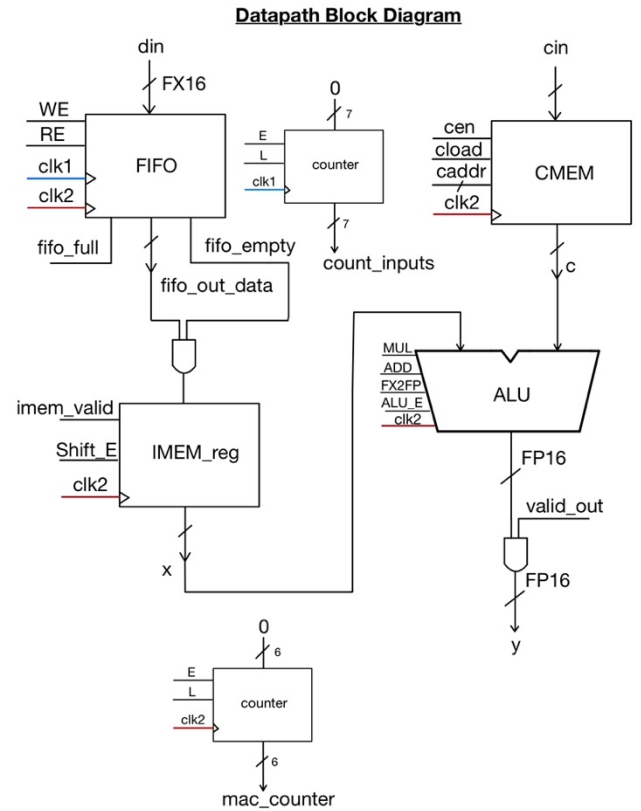


Fig.2 Datapath Block Diagram

The Arithmetic Logic Unit (ALU) is the core computation module, performing the multiply-accumulate (MAC) operation for convolution based on input data (x) and coefficients (b), where the intermediate results are converted from fixed-point to

D. Control ASM chart

valid output data. The `count_inputs` counter tracks the number of processed input samples, and when 100 outputs are generated, the system transitions to the DONE state. In this state, the control path disables the ALU (`ALU_E = 0`) and IMEM operations, signaling the completion of the FIR filter process.

IV. RTL DESIGN

At the input stage, a FIFO bridges the asynchronous clock domains by accepting input data (din) at clk1 and synchronizing it for further processing at clk2. The IMEM shift register receives data from the FIFO and shifts first in order to store the 64 most recent input samples, enabling real-time convolution. Coefficients are preloaded into the CMEM using the cload signal and addressed through the 6-bit caddr control. The FSM transitions through four states: IDLE, LOAD_COEF, PROCESS, and DONE. In the LOAD_COEF state, coefficients are loaded into CMEM. Once complete, the PROCESS state begins, where the multiply-accumulate (MAC) operation iterates over the 64 taps.

The ALU performs the convolution using inputs from IMEM and coefficients from CMEM. The MAC operation is managed by a `mac_counter`, which iterates through the 64 taps, ensuring correct sequencing of data and coefficients. The ALU also incorporates Fixed-to-Floating-Point (FX-to-FP) conversion, producing an FP16 (IEEE-754 half-precision floating-point) result. The output (`dout`) is validated through the `valid_out` signal once the computation is complete.

V. SIMULATION

The simulations presented in this report are divided into five sections. First, the Verilog testbench is introduced, followed by the QuestaSim simulation to verify the RTL design. Next, the DC report generates the gate-level netlist and provides insights into area usage, as well as hold and setup time violations. The fourth section focuses on post-synthesis analysis using the netlist file to further evaluate the design. Finally, the PrimeTime report offers detailed timing and power analysis of the circuit, providing a comprehensive evaluation of the FIR filter design.

A. Testbench

In the PROCESS state, the control path synchronizes the data flow and MAC operation. It enables the FIFO to output data to the IMEM register by asserting the imem_valid signal, while the mac_counter iterates through all 64 taps, controlling the read addresses for input samples and coefficients. Concurrently, the ALU_E signal enables the ALU to perform the multiply-accumulate operation. Upon completing 64 MAC operations, the control path asserts valid_out, signaling the availability of

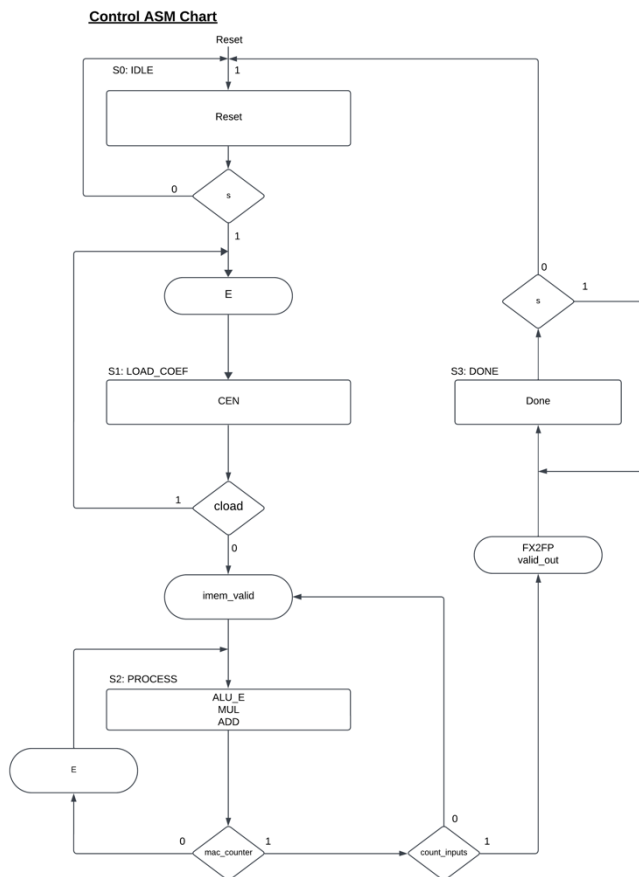


Fig.3 Control ASM Chart

emulate real-world operation. The testbench ensures coefficients are preloaded into the Coefficient Memory (CMEM) during the LOAD_COEF phase, while the input data is fed into the FIFO during the PROCESS phase. The control signals clod and valid_in enable these operations in a synchronized manner.

The testbench monitors the FIR filter's outputs (dout) and validates them against golden outputs generated from MATLAB. The comparison process ensures that the RTL design performs the multiply-accumulate (MAC) operation correctly and produces the expected fixed-to-floating-point (FX-to-FP) results. The output is read line-by-line from a file containing precomputed floating-point results and compared with dout for 100 outputs. Any mismatches are flagged with a descriptive error message, including the time and expected vs. actual values. The process also tracks the total error count to confirm correctness.

B. QuestaSim Simulation (qsim_rtl)

Running the testbench generates a QuestaSim waveform that illustrates the behavior of the inputs and outputs across the positive and negative edges of the clock signals. As described earlier, the coefficient data is initially loaded into the Coefficient Memory (CMEM) during the LOAD_COEF state, while the input data is stored into the Input Memory (IMEM) during the PROCESS state. These operations are clearly observed in Figures 4 and 5, which demonstrate the successful loading of data into both memories of the FIR core. *Note: The waveform displayed in QuestaSim shows numerical values represented in the FX16 or FP16 unsigned decimal format.*

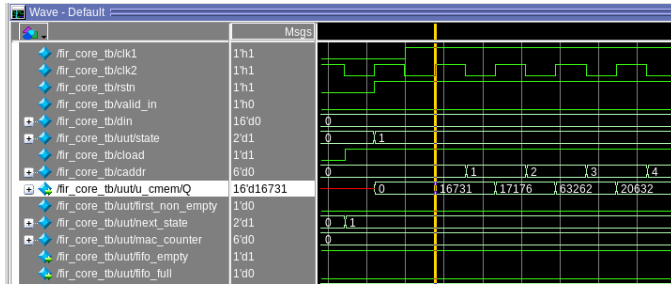


Fig.4 Coefficients Data preloaded into the CMEM

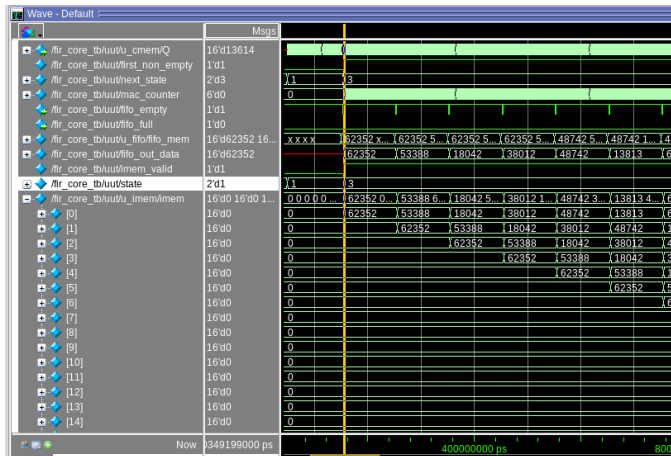


Fig.5 Input Data loaded and shifted into the IMEM

Once the data is successfully loaded, the output calculations commence. As shown in Figure 6, the output signal dout

produces the expected fixed-point (FX16) results at intervals corresponding to the 10 kHz clock (clk1), with the valid_out signal asserted to indicate valid output data.

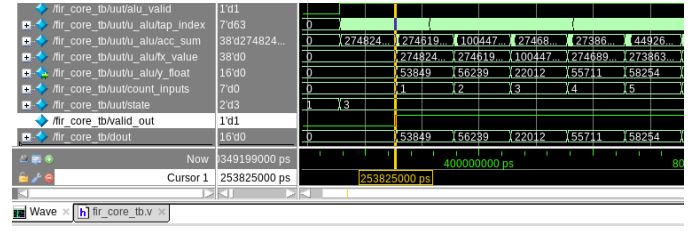


Fig.6 Output data of the FIR Filter

Furthermore, the comparison between the RTL outputs and the golden outputs generated from MATLAB is performed, as depicted in Figure 7. The error counting algorithm confirms accurate results, ensuring that the FIR filter operates correctly under the given test conditions.

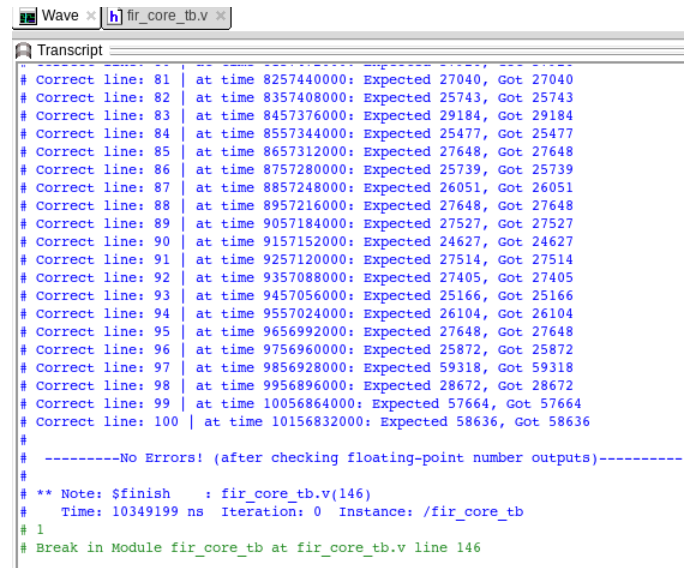


Fig.7 Total error count compared with MATLAB

C. Design Compiler (dc)

Once the RTL code was successfully tested, the design was synthesized using the Design Compiler to generate the gate-level netlist. For the Coefficient Memory (CMEM), the .lib file provided by the Memory Compiler was converted into a .db file using a pre-defined library. The Verilog code generated by the Memory Compiler was treated as a black-box, with only the .db file incorporated into the synthesis process. A portion of the resulting gate-level netlist, produced by the Design Compiler, is shown in Figure 8.

In addition to generating the gate-level netlist, the Design Compiler produced a detailed report containing timing, area, and power analysis. As illustrated in Figure 9, the report confirms that there were no timing violations, as all timing constraints were successfully met. This indicates that the synthesized design adheres to the specified clock frequencies and delay requirements.

```

Open [icon] fir_core.nlv
~EE4823FIR/dc/fir_core

// Created by: Synopsys DC Ultra(TM) in wire load mode
// Version : U-2022.12-SP7
// Date : Wed Dec 11 17:17:30 2024

module fir_core ( clk1, clk2, rstn, din, valid_in, caddr, cin, cload, dout,
valid_out, fifo_empty, fifo_full );
input [15:0] din;
input [5:0] caddr;
input [15:0] cin;
output [15:0] dout;
input clk1, clk2, rstn, valid_in, cload;
output valid_out, fifo_empty, fifo_full;
wire
done, first_non_empty, N62, u_fifo_rd_ptr_0, u_imem_N35, u_imem_N34,
u_imem_N33, u_imem_N32, u_imem_N31, u_alu_N235, u_alu_N234,
u_alu_N233, u_alu_N232, u_alu_N231, u_alu_N230, u_alu_N229,
u_alu_N228, u_alu_N227, u_alu_N226, u_alu_N225, u_alu_N224,
u_alu_N223, u_alu_N222, u_alu_N221, u_alu_N220, u_alu_N219,
u_alu_N218, u_alu_N217, u_alu_N216, u_alu_N215, u_alu_N214,
u_alu_N213, u_alu_N212, u_alu_N211, u_alu_N210, u_alu_N209,
u_alu_N208, u_alu_N207, u_alu_N206, u_alu_N205, u_alu_N204,
u_alu_N203, u_alu_N202, u_alu_N201, u_alu_N200, u_alu_N199,
u_alu_N198, u_alu_N196, u_alu_N195, u_alu_N194, u_alu_N193,
u_alu_N192, n400, n401, n402, n403, n404, n405, n406, n407, n408,
n409, n410, n411, n412, n413, n414, n415, n416, n417, n418, n419,
n420, n421, n422, n423, n424, n425, n426, n427, n428, n429, n430,
n431, n432, n433, n434, n435, n436, n437, n438, n439, n440, n441,
n442, n443, n444, n445, n446, n447, n448, n449, n450, n451, n452,

```

Fig.8 Gate-Level Netlist of the FIR Core

Path Type: max		
Point	Incr	Path
clock clk2 (rise edge)	99968.00	99968.00
clock network delay (ideal)	0.00	99968.00
input external delay	0.05	99968.05 r
valid_in (in)	0.02	99968.06 r
U3837/Y (NAND2BXLTS)	0.23	99968.29 f
U3838/Y (NOR2XLTS)	0.51	99968.80 r
U3845/Y (NAND3XLTS)	0.55	99969.34 f
U3846/Y (CLKBUF2TS)	0.38	99969.72 f
U3849/Y (CLKBUF2TS)	0.25	99969.97 f
U3850/Y (OAI2BB2XLTS)	0.30	99970.27 f
u_fifo_fifo_mem_reg_3_0_0 /D (DFFQX1TS)	0.00	99970.27 f
data arrival time		99970.27
clock clk1 (rise edge)	100000.00	100000.00
clock network delay (ideal)	0.00	100000.00
u_fifo_fifo_mem_reg_3_0_0 /CK (DFFQX1TS)	0.00	100000.00 r
library setup time	-0.33	99999.67
data required time		99999.67
data arrival time		-99970.27
slack (MET)		29.40

Fig.9 Timing Report from Design Compiler

```

Open [icon] *fir_core.dcrpt
~EE4823FIR/dc/fir_core

Version: U-2022.12-SP7
Date : Wed Dec 11 17:17:30 2024
*****

Library(s) Used:

scx3_cmos8rf_lpv1_tt_1p2v_25c (File: /courses/ee6321/share/ibm13rflpvt/s)
scx3_cmos8rf_lpv1_tt_1p2v_25c.db)
USERLIB (File: /homes/user/stud/fall24/yj2848/EE4823FIR/dc/lib2db/
CMEM_tt_1p2v_25c_syn.db)

Number of ports: 62
Number of nets: 5243
Number of cells: 4995
Number of combinational cells: 3768
Number of sequential cells: 1225
Number of macros/black boxes: 1
Number of buf/inv: 971
Number of references: 52

Combinational area: 33995.520468
Buf/Inv area: 4887.360194
Noncombinational area: 39880.798740
Macro/Black Box area: 12671.385742
Net Interconnect area: undefined (No wire load specified)

Total cell area: 86547.704950
Total area: undefined
1
Information: Propagating switching activity (medium effort zero delay simula

```

Fig.10 Area Report from the Design Compiler

The Design Compiler also provided an area report, shown in Figure 10, which offers a detailed breakdown of the design's

physical footprint. This includes metrics such as the total area, the number of ports, nets, and the distribution of combinational and sequential cells, all of which are highlighted in the figure.

D. Post-synthesis dynamic timing verification (qsim_dc)

The gate-level netlist created from the Design Compiler is tested using a modified testbench to validate its functionality and capture switching activities. The modified testbench includes function calls that record cycle-accurate switching activities, generating a fir_core.vcd file for further analysis. Additionally, the runsim.do script is updated to reference the gate-level netlist and include the fir_core.syn.sdf file for SDF (Standard Delay Format) back-annotation. As shown in Figure 11, the QuestaSim transcript confirms the successful simulation of the gate-level netlist, including the completion of the SDF back-annotation process.

```

Open [icon] transcript
~EE4823FIR/qsim_dc/fir_core

# ** Warning: (vsim-3722) ../dc/fir_core/fir_core.nlv(2842): [TFMPC] - Missing
connection for port 'QN'
# Loading timing data from ../dc/fir_core/fir_core.syn.sdf
# ** Warning: (vsim-8756) Instance 'fir_core_tb.uut.u_fifo_rd_ptr_gray_reg_1' -
Negative timing check limits detected in simulation with cells modeled without
delayed copies of data or reference signals.
# Time: 0 ps Iteration: 0 Instance: /fir_core_tb File: /courses/ee6321/share/
ibm13rflpvt/verilog/ibm13rflpvt.v Line: 23343
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
# Time: 0 ps Iteration: 0 Instance: /fir_core_tb File: fir_core_tb.v
# Loading coefficients from file...

```

Fig.11 SDF Back Annotation

The modified testbench also generates a QuestaSim waveform for post-synthesis simulation, enabling detailed timing analysis. The waveform results, illustrated in Figure 12, show that the outputs of the gate-level netlist match the results of the original RTL code, confirming functional equivalence and proper synthesis. This validation step ensures that the design behaves correctly with gate-level timing delays applied.

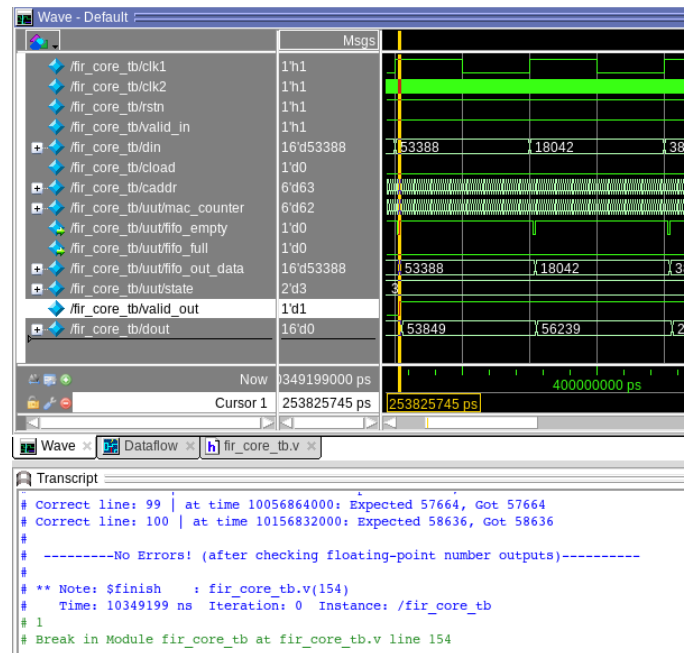


Fig.12 Post-synthesis QuestaSim Waveform

However, during the gate-level simulation, specific phenomena can be observed, such as counter glitches caused by

bit-level propagation delays. As seen in Figure 13, a temporary erroneous value (e.g., transitioning from 5 to 6 briefly showing 4) occurs due to the different flip-flop transition times in the combinational logic. This phenomenon is referred to as "Metastability" or "Glitch" and typically arises from propagation delay skew in combinational logic paths, where multiple bits update asynchronously. Such glitches are common in post-synthesis simulations where gate-level delays are explicitly modeled.

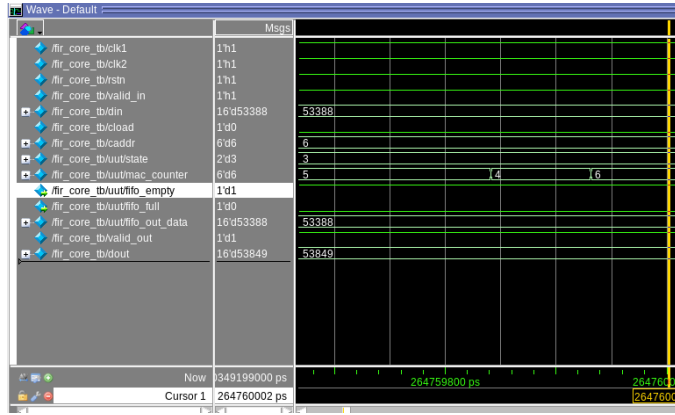


Fig.13 Transitioning from 5 to 6 briefly showing 4 (Glitch)

Additionally, the waveform reveals an initial undefined state (X state) for certain signals, as shown in Figure 14. This phenomenon is known as "Power-On Reset Indeterminacy" or "Initialization Uncertainty". At the start of simulation, registers or memory elements are uninitialized, resulting in X (unknown) values. This state persists until the reset signal (rstn) propagates through the entire design, stabilizing all registers and signals. The presence of X states highlights the importance of proper reset logic and explicit initialization in the design.

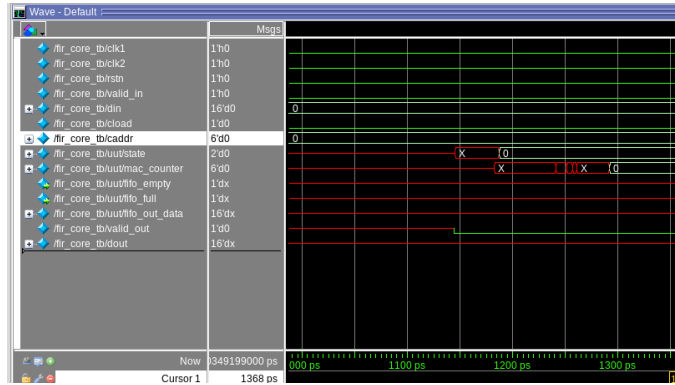


Fig.14 Initialization Uncertainty (X state)

E. Post-synthesis static timing and power verification (pt)

Running the PrimeTime script generates a comprehensive report highlighting the timing analysis of the design, including the longest and shortest path delays. As shown in Figure 15, the shortest path delay is identified, while the maximum path delay, representing the critical path, is highlighted in Figure 16. The critical path delay provides valuable insight into the circuit's timing constraints and overall performance, ensuring that the design meets the required clock frequency specifications.

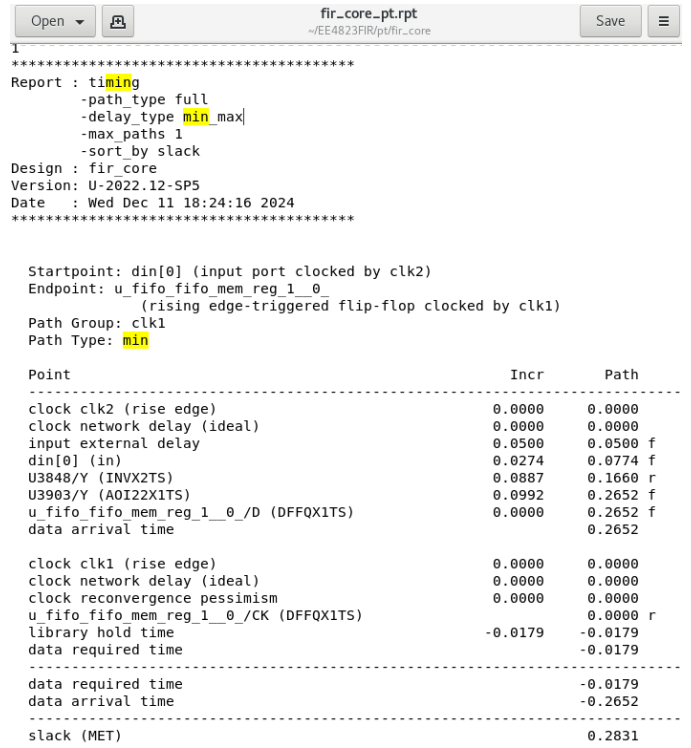


Fig.15 Shortest Path Delay

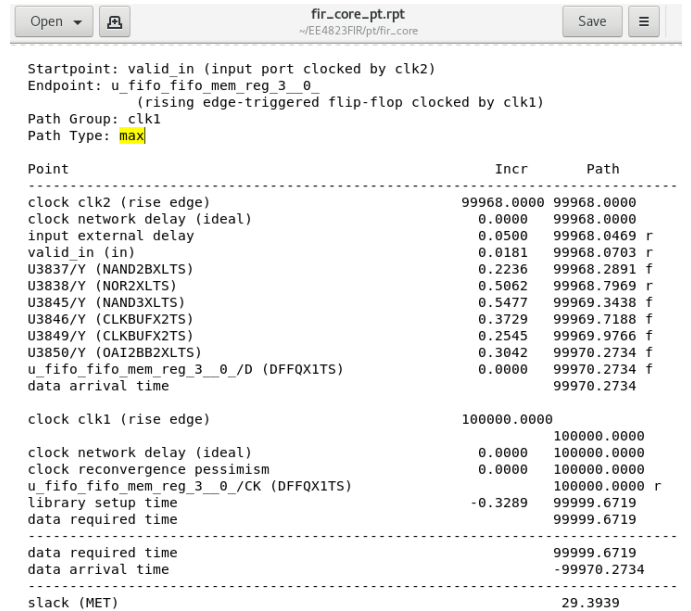



Fig.16 Critical Path Delay

In addition to timing analysis, the PrimeTime report also provides a detailed power analysis of the circuit. As illustrated in Figure 17, the total power consumption is broken down into Net Switching Power, Cell Internal Power, and Cell Leakage Power, with contributions from black-box components also included. Furthermore, the report offers a hierarchical power analysis, which provides a structured breakdown of power consumption across different design levels, as depicted in Figure


18. This analysis aids in identifying power hotspots and optimizing the design for energy efficiency.

Open ▾



fir_core_pt.rpt
~EE4823FIR/pt/fir_core

Save



Attributes

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

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
clock_network	1.176e-05	0.0000	0.0000	1.176e-05	(62.94%)	i
register	6.425e-07	7.067e-08	4.712e-08	7.603e-07	(4.07%)	
combinational	4.446e-06	1.636e-06	3.559e-08	6.117e-06	(32.75%)	
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	4.513e-08	0.0000	4.513e-08	(0.24%)	

Net Switching Power	= 1.752e-06 (9.38%)					
Cell Internal Power	= 1.684e-05 (90.18%)					
Cell Leakage Power	= 8.271e-08 (0.44%)					

Total Power	= 1.868e-05 (100.00%)					

X Transition Power	= 7.741e-11					
Glitching Power	= 8.851e-09					

Peak Power	= 0.0915					
Peak Time	= 7852955.000					

Fig.17 Time Based Power Report

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
clock_network	1.176e-05	0.0000	0.0000	1.176e-05	(62.94%)	i
register	6.425e-07	7.067e-08	4.712e-08	7.603e-07	(4.07%)	
combinational	4.446e-06	1.636e-06	3.559e-08	6.117e-06	(32.75%)	
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	4.513e-08	0.0000	4.513e-08	(0.24%)	

Net Switching Power	= 1.752e-06 (9.38%)					
Cell Internal Power	= 1.684e-05 (90.18%)					
Cell Leakage Power	= 8.271e-08 (0.44%)					

Total Power	= 1.868e-05 (100.00%)					

X Transition Power	= 7.741e-11					
Glitching Power	= 8.851e-09					

Fig.18 Hierarchical Time-Based Power Report

VI. OUTPUT ANALYSIS (METRICS)

For the FIR filter project, specific design and performance metrics were established to ensure successful implementation and evaluation. The filter core must load all inputs and coefficients into their respective memory registers while utilizing a single adder and multiplier for computation. The target throughput of the design is 10 kS/s. To achieve this, a set of comprehensive metrics was defined and evaluated throughout the project lifecycle.

1. **Input:** Randomly generated 10,000 16-bit real-valued numbers.
2. **Throughput:** Verified using the PrimeTime report with Design Compiler generated SDF annotation.
3. **Maximum Clock Frequency:** Determined from PrimeTime timing analysis with SDF annotations generated by the Design Compiler.
4. **Power Analysis:** Conducted using PrimeTime reports to evaluate the total power consumption.

5. **Energy Efficiency:** Analyzed using PrimeTime reports combined with Design Compiler SDF annotations and QuestaSim generated VCD files.
6. **Area:** Reported using the Design Compiler to determine resource utilization.
7. **Accuracy:** Measured as the Root Mean Square Error (RMSE) between the FIR filter's outputs and MATLAB-generated golden results in 32-bit floating-point format, with 16-bit inputs generated using MATLAB's random functions.

A. Area

An area analysis was conducted to verify that the design meets the specified area constraints. The total cell area represents the physical space occupied by the core and serves as an estimate of the chip size required for manufacturing. The detailed breakdown of the area, as reported in Figure 10, is as follows:

$$\text{Combinational Area: } 33995.52 \text{ mm}^2$$

$$\text{Buffer and Inverter Area: } 4887.36 \text{ mm}^2$$

$$\text{Non combinational Area: } 39880.8 \text{ mm}^2$$

By summing these contributions, the total cell area is calculated to be:

$$\text{Total Cell Area} = 86547.7 \text{ mm}^2$$

B. Timing

A timing analysis was performed to verify that the design meets the required timing metrics. The analysis provides insights into the critical path delay and the minimum path delay, which are key factors in determining the overall performance of the circuit. As shown in Figure 15 and Figure 16, the results are as follows:

$$\text{Minimum Path Delay: } 0.2652 \text{ ns}$$

$$\text{Critical Path Delay: } 99970.2734 \text{ ns}$$

C. Clock Frequencies

The clock frequency determines the operational speed of the circuit and is a critical parameter for evaluating the FIR filter core. This frequency directly impacts the system's throughput, which is discussed in the following section. The maximum operating frequency calculation is as follows:

$$freq_{max} = \frac{1}{T_{min}} = \frac{1}{29.39 \text{ ns}} = 34.03 \text{ MHz}$$

D. Throughput

A throughput analysis was conducted to ensure that the design meets the specified performance metrics. The throughput can be determined using the following formula:

$$\text{Throughput} = \frac{\text{Clock Frequency}}{\text{Cycles per Sample}} = \frac{640 \text{ kHz}}{64 \text{ cycles/sample}} = 10 \text{ kS/s}$$

E. Accuracy

An accuracy analysis was performed to ensure that the design meets the expected precision requirements. The accuracy

was quantified using the Normalized Root Mean Squared Error (NRMSE), which is defined as:

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}}$$

The Root Mean Squared Error (RMSE) is calculated using the following equation:

$$\sqrt{\frac{\sum_{i=0}^n (y_i - y_i^*)^2}{n}}$$

Here, y_i represents the full-precision MATLAB results, while y_i^* corresponds to the quantized results obtained from the RTL simulation. From the calculation:

Accuracy: Worst case: 95.30189 % | Average: 99.92025 %

F. Power and Energy Efficiency

A power analysis was performed to evaluate the total power consumption of the design, including contributions from Net Switching Power, Cell Internal Power, and Cell Leakage Power. The calculations also include black-box power contributions. The detailed breakdown of power metrics is as follows:

Net Switching Power: $1.752 \times 10^{-6} W$

Cell Internal Power: $1.684 \times 10^{-5} W$

Cell Leakage Power: $8.271 \times 10^{-8} W$

By summing these components, the Total Power is obtained as:

$$Total\ Power = 1.868 \times 10^{-5} W$$

An energy efficiency analysis was conducted to assess the design's performance in terms of energy consumption per operation. Energy efficiency is calculated using the formula:

$$Energy = Power \times Time$$

The resulting energy efficiency is:

$$Energy\ Efficiency = 0.549\ pJ/S$$

VII. CONCLUSION

In conclusion, the FIR Filter Core was successfully implemented and validated through both MATLAB and Verilog. The golden outputs generated using the MATLAB filter function matched the results obtained from the behavioral code, ensuring accuracy. The input data, coefficients, and outputs were generated using MATLAB, enabling the creation of a pseudocode for the FIR filter. This pseudocode served as the foundation for developing the ASM chart, Datapath Block Diagram, and Control ASM. The FIR core and its corresponding testbench were implemented in Verilog, with waveform simulations demonstrating the core's functionality.

The project metrics were achieved successfully. A gate-level netlist was generated using the Design Compiler, with the .sdf and .vcd files enabling delay annotation and cycle-accurate switching activity analysis. The PrimeTime report provided insights into the circuit's critical and shortest path delays, along with a detailed power analysis. This comprehensive implementation confirms the successful design, synthesis, and verification of the FIR Filter Core.

REFERENCES

- [1] Trimale, Manish B. "A review: FIR filter implementation." 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE, 2017.
- [2] Ehliar, Andreas. "Area efficient floating-point adder and multiplier with IEEE-754 compatible semantics." 2014 International conference on field-programmable technology (FPT). IEEE, 2014.
- [3] Coowar, R., and Hvd Biggelaar. "Asm Charts In Vhdl." Computers in Education Journal 14.4 (2004).