FPGA Implementation of Adaptive LMS Filter using Offset Binary Coding (OBC)

Names: Rajdeep Saha, Ramkushal B.
Roll Nos: IMT2023600, IMT2023601

November 25, 2025

Abstract

An Adaptive Noise Canceller based on the Sign-Error LMS algorithm with Offset Binary Coding was implemented on an Artix-7 FPGA. Python-based Golden Model verification and post-implementation timing, power, resource, and critical-path reports demonstrate that the architecture is efficient, scalable, and capable of real-time DSP performance.

Contents

1 Introduction

Adaptive filters are essential components in digital signal processing, widely used for applications such as noise cancellation, system identification, and echo cancellation. This project focuses on the hardware implementation of an Adaptive Noise Canceller (ANC) on an Artix-7 FPGA.

The core objectives of this project were:

- To implement the Least Mean Squares (LMS) adaptive algorithm.
- To utilize the **Sign-Error** variant of LMS to reduce hardware complexity.
- To employ **Offset Binary Coding (OBC)** for the inner product calculation, replacing standard multipliers with shift-add logic.
- To verify the design using Python Golden Model and hardware Integrated Logic Analyzer (ILA).

2 Mathematical Background

2.1 Adaptive Noise Cancellation

The system takes two inputs: a primary signal $d[n]$ containing the desired signal plus noise, and a reference signal $x[n]$ containing correlated noise. The filter estimates the noise $y[n]$ and subtracts it from $d[n]$ to produce the clean error signal $e[n]$.

$$e[n] = d[n] - y[n] \tag{1}$$

2.2 Sign-Error LMS Algorithm

The standard LMS update rule $w_k[n+1] = w_k[n] + \mu e[n]x[n-k]$ requires significant multiplication resources. To optimize for hardware, we implemented the Sign-Error LMS variant:

$$w_k[n+1] = w_k[n] + \mu \cdot \text{sign}(e[n]) \cdot x[n-k] \tag{2}$$

Here, μ denotes the learning rate and $\mathbf{w} = [w_0, w_1, \ldots, w_{N-1}]$ represents the adaptive filter weights. This simplifies the update step to a conditional addition or subtraction of the input vector.

2.3 Offset Binary Coding (OBC)

To compute the inner product $\mathbf{y[n]} = \mathbf{w}^T\mathbf{x}$ without standard multipliers, we utilized Offset Binary Coding principles in a distributed arithmetic style logic. The inner product is decomposed into bit-slices:

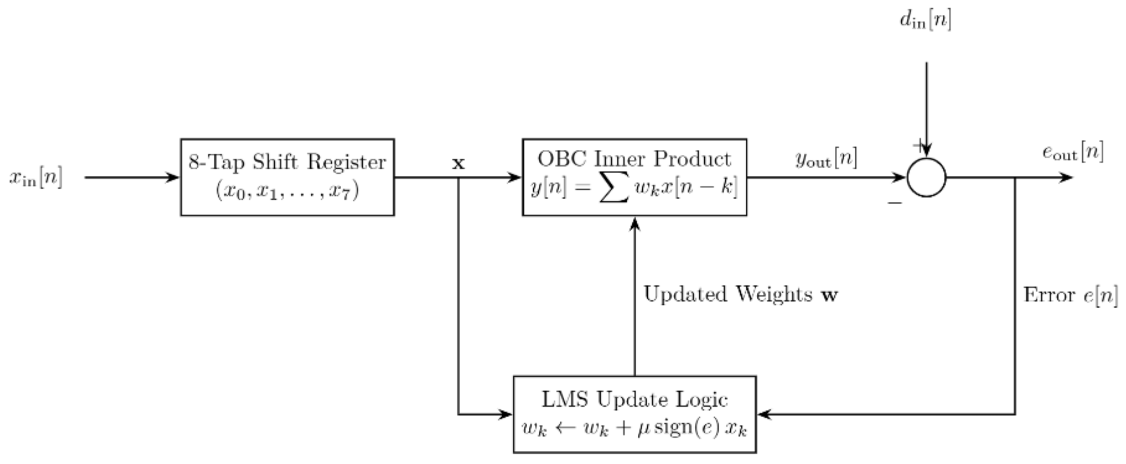
$$y[n] = \sum_{j=0}^{B-1} D_j 2^{-j} + D_{\text{offset}} \tag{3}$$

Where D_j is a partial sum of weights conditioned on the input bits, and D_{offset} is a pre-calculated correction term $(-\frac{1}{2}\sum w_k)$. This allows the filter to be implemented using only adders, shifters, and accumulators.

3 System Architecture

3.1 Block Diagram

The system consists of an 8-tap FIR filter structure with feedback control for weight updates.



$d_{\text{in}}[n]$
$x_{\text{in}}[n]$
8-Tap Shift Register $(x_0, x_1, \ldots, x_7)$
$\mathbf{x}$
OBC Inner Product $y[n] = \sum w_k x[n-k]$
$y_{\text{out}}[n]$
$+$
$-$
$e_{\text{out}}[n]$
Updated Weights $\mathbf{w}$
Error $e[n]$
LMS Update Logic $w_k \leftarrow w_k + \mu\,\text{sign}(e)\,x_k$

Figure 1: Block Diagram of the 8-Tap Adaptive Sign-Error LMS Filter with OBC

3.2 Signal Definitions & Mathematical Model

The system signals and governing equations are defined as follows:

Explanation of Signals:

- $x_{\textbf{in}}[n]$: Reference noise input.
- $d_{\textbf{in}}[n]$: Primary input (containing Clean Signal + Noise).
- $x_0 \ldots x_7$: Stored current and past 7 samples of x_{in}.
- $y_{\textbf{out}}[n]$: Estimated noise generated by the adaptive filter.
- $e_{\textbf{out}}[n]$: Cleaned signal (Noise-reduced output).

Mathematical Equations:

$$\boxed{\begin{aligned} d_{\text{in}}[n] &= s[n] + p[n] \\ y_{\text{out}}[n] &= \sum_{k=0}^{7} w_k[n]\, x_{\text{in}}[n-k] \\ e_{\text{out}}[n] &= d_{\text{in}}[n] - y_{\text{out}}[n] \\ w_k[n+1] &= w_k[n] + \mu \cdot \text{sign}(e_{\text{out}}[n]) \cdot x_{\text{in}}[n-k] \end{aligned}}$$

3.3 Hardware Implementation Details

The design was implemented in Verilog HDL using Vivado 2022.2.

- **Memory (rom_data.v):** Stores 1024 samples of noisy sine wave data generated via Python Golden Model. Implemented as a synthesizable ROM.
- **LMS Core (lms_core.v):** Implements the OBC Pipelined Architecture.
  - *Stage 1 (Offset):* Calculates the OBC offset term.
  - *Stage 2 (Slicing):* Determines the weight sum for the current bit position.
  - *Stage 3 (Accumulate):* Shifts and accumulates partial results into a 40-bit accumulator to prevent overflow.
  - *Stage 4 (Update):* Updates weights based on the sign of the error.
- **Top Level (top_lms_system.v):** Integrates the ROM, LMS Core, and Integrated Logic Analyzer (ILA) for real-time debugging.

3.4 Design Constraints

The system targets the Basys 3 FPGA board.

- **Clock:** A 100 MHz system clock is sourced from the onboard oscillator (Pin W5). This high frequency ensures the multi-cycle OBC calculation completes well within the sampling period of audio-range signals.
- **Reset:** The reset signal is mapped to **Switch 0 (Pin V17)**. This allows for a stable manual reset, which is crucial for the ILA trigger setup. A 'set_false_path' constraint is applied to this pin to safely ignore timing analysis on this asynchronous human input.
- **ILA Trigger:** The debug core is configured with 'Capture Control' enabled. This is essential because the OBC core takes multiple clock cycles to process one sample. The trigger condition 'core_done == 1' ensures the ILA only records valid output samples, filtering out intermediate calculation states.

3.5 Simulation Testbench

A behavioral testbench was developed to validate the logic before synthesis.

- **Clock Generation:** Generates a 100 MHz clock (period = 10ns).
- **Duration:** The simulation runs for 1,000,000 ns (1 ms). This duration is sufficient to process thousands of samples, allowing the adaptive filter enough time to converge and visibly reduce the error signal in the waveform viewer.

4 Verification and Results

4.1 Python Golden Model

A Python script was developed to simulate the Sign-Error LMS algorithm and generate the test vectors. This served as the reference for hardware verification.

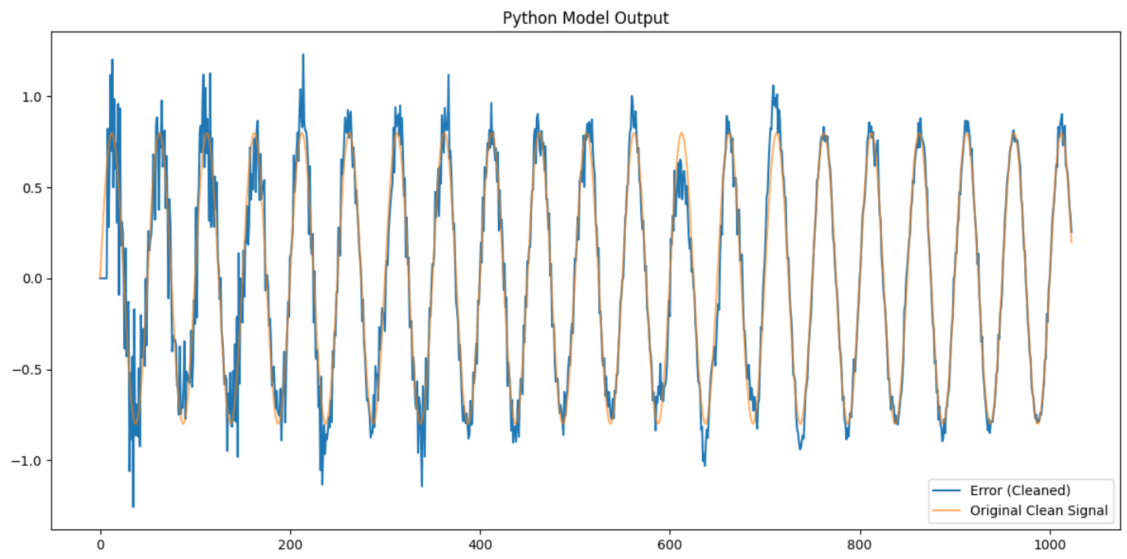
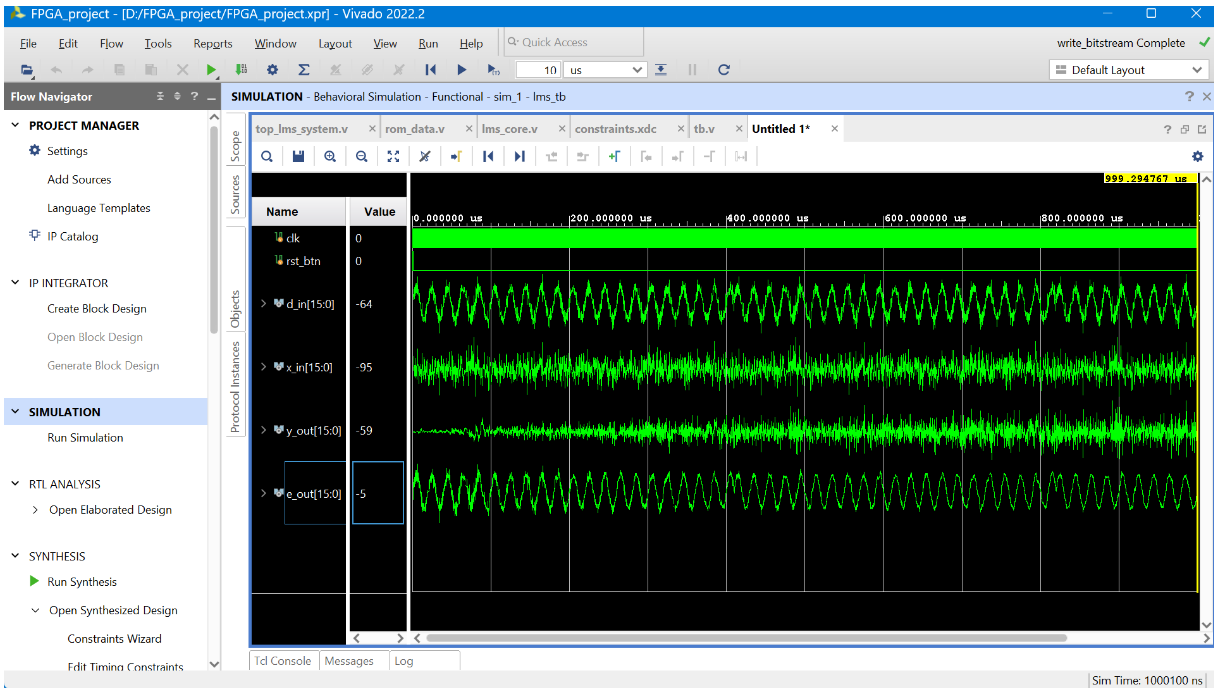


Figure 1: Python Golden Model Output: Comparison of Clean Signal vs Filtered Error

4.2 Hardware Simulation

Behavioral simulation was performed to verify the timing and logic correctness before synthesis.



Name	Value
clk	0
rst_btn	0
d_in[15:0]	-64
x_in[15:0]	-95
y_out[15:0]	-59
e_out[15:0]	-5

Figure 2: Vivado Behavioral Simulation Results showing convergence over time.

4.3 Hardware ILA Results

The design was deployed on the Basys 3 FPGA. The ILA waveform captures demonstrate the real-time noise cancellation capability. By using Capture Control, valid output samples were isolated from the calculation latency.

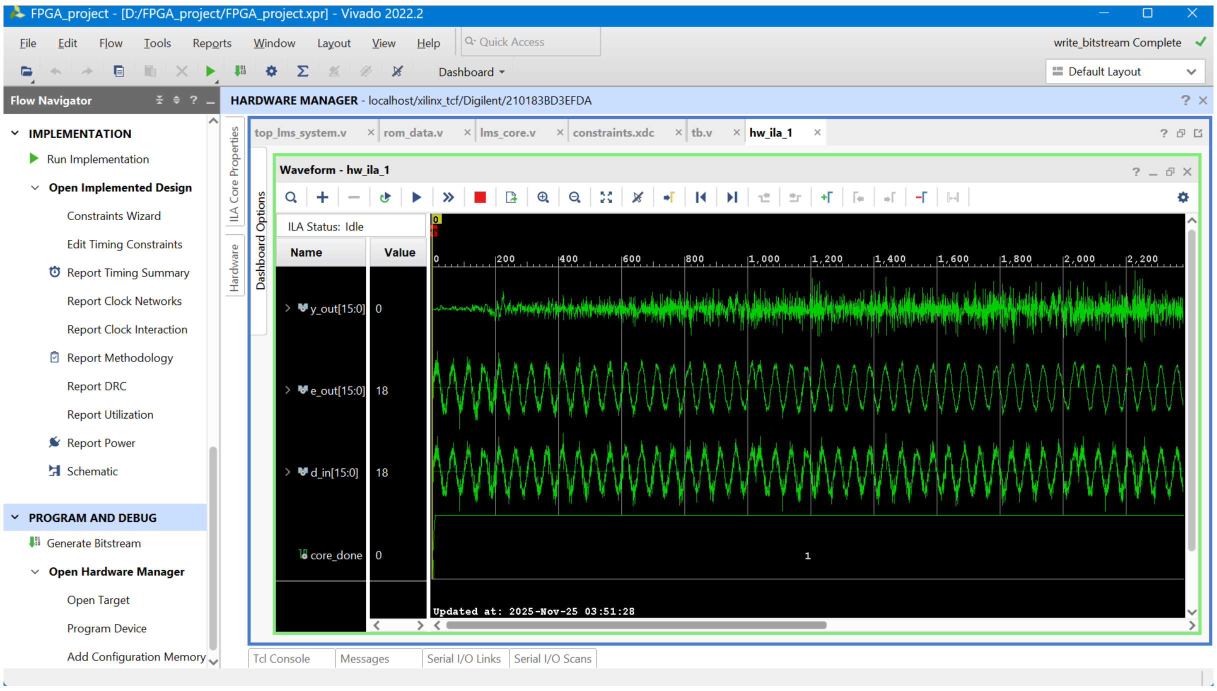


Figure 3: ILA Hardware Capture showing d_{in} (Noisy Input) vs e_{out} (Clean Output).

4.4 Key Insights

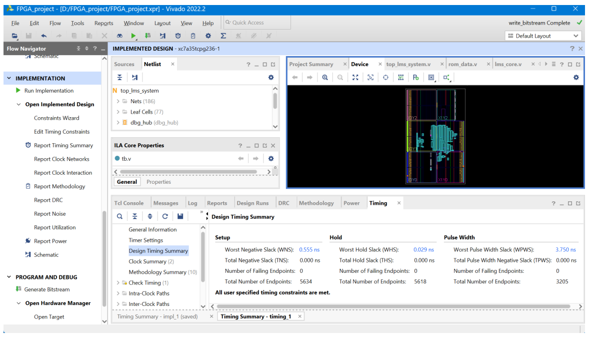
- The adaptive LMS filter does not cancel noise instantly, it requires multiple iterations for the weights to converge and learn the noise characteristics.
- Initially, the error signal $e_{\text{out}}[n]$ contains strong noise because the filter has not yet estimated the noise component accurately.
- As the weights update over time, the filter estimate $y_{\text{out}}[n]$ becomes increasingly similar to the reference noise $x_{\text{in}}[n]$, indicating successful modelling of the noise.
- After convergence, subtracting $y_{\text{out}}[n]$ from the primary input $d_{\text{in}}[n]$ results in a smooth sinusoidal signal, proving that the clean target signal $s[n]$ has been recovered.
- The similarity between behavioral simulation results and real FPGA ILA capture confirms the correctness of both the algorithm and the hardware implementation.

5 Post-Implementation Reports

After synthesis and implementation, several hardware performance reports were generated in Vivado to verify timing closure, resource efficiency, and power consumption of the proposed LMS–OBC architecture. The main results are summarised below.

5.1 Timing Summary

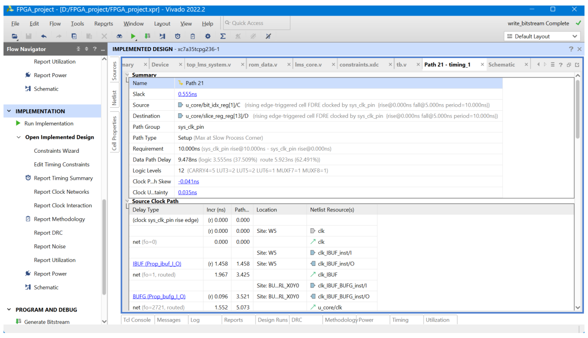
The timing report confirms that all user-defined timing constraints are satisfied. The design meets a clock period requirement of 10 ns (100 MHz) with a Worst Negative Slack (WNS) of 0.555 ns, indicating safe timing margins.


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.555 ns	Worst Hold Slack (WHS):	0.029 ns	Worst Pulse Width Slack (WPWS):	3.750 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	5634	Total Number of Endpoints:	5618	Total Number of Endpoints:	3205

All user specified timing constraints are met.

Figure 4: Timing Summary


Summary

Name	Path 21
Slack	0.555ns
Source	u_core/bit_idx_reg[1]/C (rising edge-triggered cell FDRE clocked by sys_clk_pin {rise@0.000ns fall@5.000ns period=10.000ns})
Destination	u_core/slice_reg_reg[13]/D (rising edge-triggered cell FDRE clocked by sys_clk_pin {rise@0.000ns fall@5.000ns period=10.000ns})
Path Group	sys_clk_pin
Path Type	Setup (Max at Slow Process Corner)
Requirement	10.000ns (sys_clk_pin rise@10.000ns - sys_clk_pin rise@0.000ns)
Data Path Delay	9.478ns (logic 3.555ns (37.509%) route 5.923ns (62.491%))
Logic Levels	12 (CARRY4=5 LUT3=2 LUT5=2 LUT6=1 MUXF7=1 MUXF8=1)
Clock P...h Skew	-0.041ns
Clock U...tainty	0.035ns

Source Clock Path

Delay Type	Incr (ns)	Path...	Location	Netlist Resource(s)
(clock sys_clk_pin rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: W5	clk
net (fo=0)	0.000	0.000		clk
			Site: W5	clk_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 1.458	1.458	Site: W5	clk_IBUF_inst/O
net (fo=1, routed)	1.967	3.425		clk_IBUF
			Site: BU...RL_X0Y0	clk_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.096	3.521	Site: BU...RL_X0Y0	clk_IBUF_BUFG_inst/O
net (fo=2721, routed)	1.552	5.073		u_core/clk

Figure 5: Critical Path Properties

5.2 Critical Path Analysis

The worst-case datapath was traced using Vivado's implemented schematic viewer. The path runs through the OBC accumulator and CARRY4 chain, confirming that the inner product logic is the dominant contributor to total delay, an expected outcome for adaptive FIR filters.

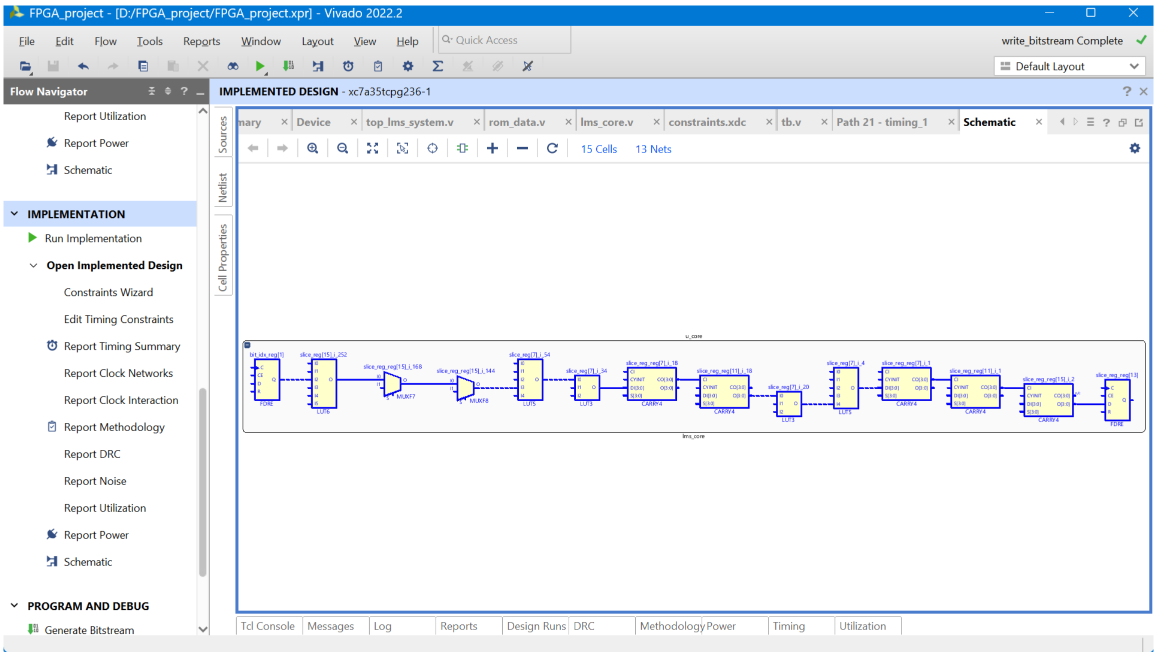
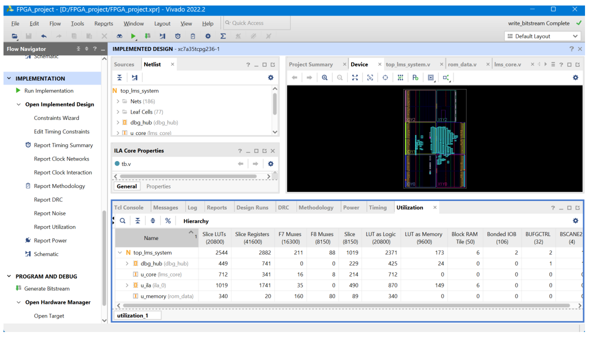

FPGA_project - [D:/FPGA_project/FPGA_project.xpr] - Vivado 2022.2
IMPLEMENTED DESIGN - xc7a35tcpg236-1
Schematic
15 Cells
13 Nets
u_core
lms_core
FDRE
LUT6
MUXF7
MUXF8
LUT5
LUT3
CARRY4

Figure 6: Critical Path Schematic generated from implemented design

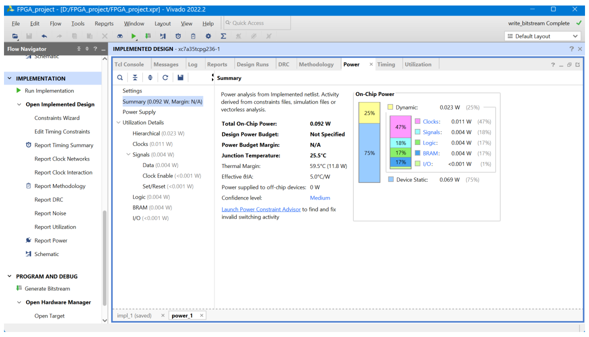
5.3 Resource Utilization

FPGA resource usage was analyzed post-implementation. The design occupies a small percentage of overall device capacity, showing high efficiency due to the OBC approach requiring no DSP slices.



Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	BSCANE2 (4)
top_lms_system	2544	2882	211	88	1019	2371	173	6	2	2	1
dbg_hub (dbg_hub)	449	741	0	0	229	425	24	0	0	1	1
u_core (lms_core)	712	341	16	8	214	712	0	0	0	0	0
u_ila (ila_0)	1019	1741	35	0	490	870	149	6	0	0	0
u_memory (rom_data)	340	20	160	80	89	340	0	0	0	0	0

Figure 7: Device Resource Utilization



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.092 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.5°C**
Thermal Margin:	59.5°C (11.8 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.023 W	(25%)
Clocks:	0.011 W	(47%)
Signals:	0.004 W	(18%)
Logic:	0.004 W	(17%)
BRAM:	0.004 W	(17%)
I/O:	<0.001 W	(1%)
Device Static:	0.069 W	(75%)

Figure 8: Power Summary

5.4 Power Report

Power estimation was performed using switching activity extracted from implementation. The design shows very low on-chip power consumption of **0.092 W**, demonstrating suitability for low-power embedded signal-processing applications.

Overall, the timing, power, and resource reports confirm that the LMS filter operates efficiently at 100 MHz, converges reliably in real time, and fits comfortably within the available FPGA resources.

6 Conclusion

The project successfully demonstrated the implementation of an adaptive filter using OBC logic on an FPGA. The 8-tap Sign-Error LMS filter effectively removed noise from the input signal. The OBC implementation reduced reliance on multipliers by leveraging shift-add logic, while the pipelined architecture ensured timing closure at 100 MHz.

7 Project Repository

- GitHub Link: https://github.com/dipsy32/Adaptive_LMS_FPGA