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

Rajdeep Saha  Ramkushal B.

Roll Nos: IMT2023600, IMT2023601

November 25, 2025

# Problem Statement

**Objective:**

- Implement an Adaptive Noise Canceller (ANC) on an Artix-7 FPGA.
- Remove noise from a corrupted input signal in real-time.

**Constraints & Methodology:**

- **Algorithm:** Least Mean Squares (LMS) - Sign-Error Variant.
- **Hardware Constraint:** Use **Offset Binary Coding (OBC)** for the inner product calculation to replace standard multipliers with shift-add logic.
- **Verification:** Python Golden Model vs. Hardware ILA.

# Mathematical Background

## 1. Adaptive Noise Cancellation:

$$e[n] = d[n] - y[n]$$

- $d[n]$: Primary Input (Signal + Noise)
- $y[n]$: Estimated Noise (Filter Output)
- $e[n]$: Error / Clean Signal

## 2. Sign-Error LMS Update:

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

*Reduces hardware complexity by avoiding full multiplication in update step.*

## 3. OBC Inner Product:

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

*Decomposes vector multiplication into bit-serial shift-add operations.*

# System Block Diagram

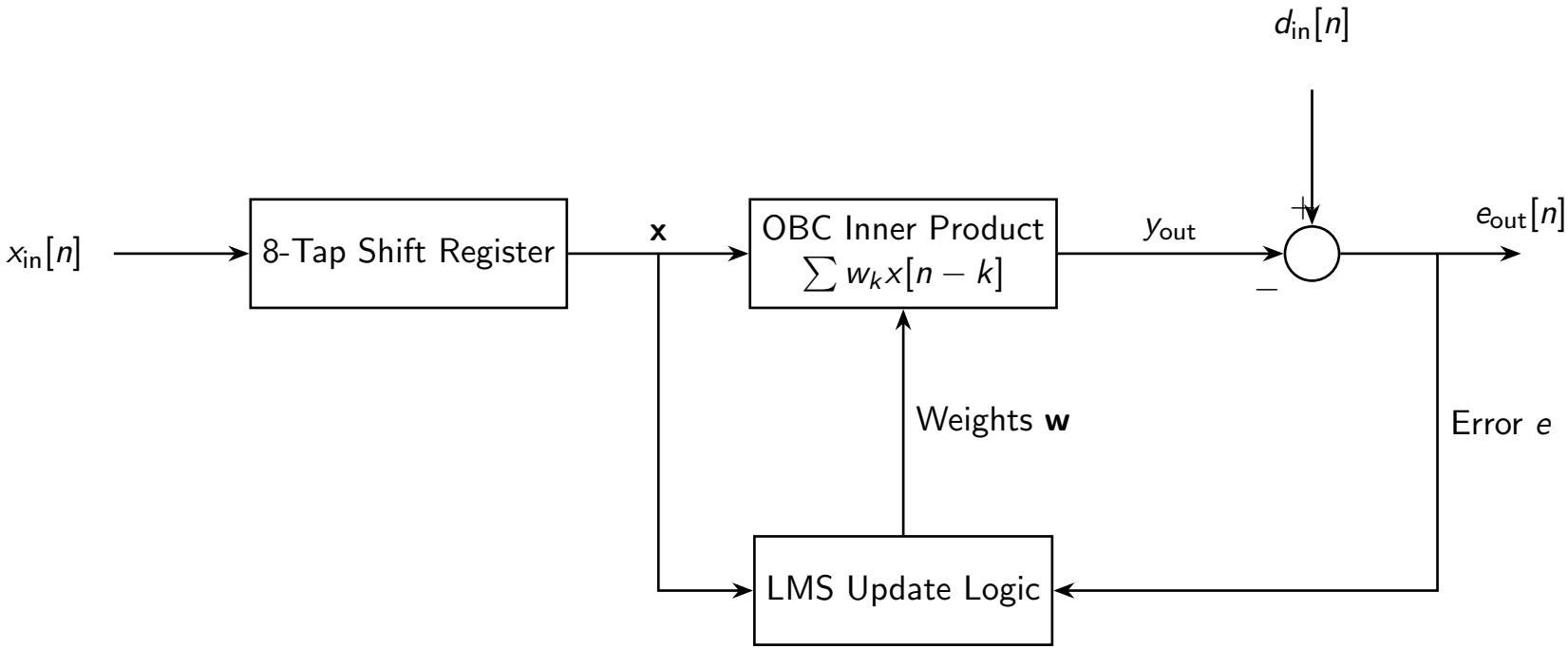



Figure: Adaptive Filter Architecture

# Hardware Implementation Details

**Target:** Artix-7 FPGA (Basys 3) using Vivado 2022.2

- **LMS Core (lms_core.v):**
  - Pipelined OBC Architecture (4 Stages).
  - **40-bit Accumulator** used to prevent overflow during bit-serial sums.
- **Memory (rom_data.v):**
  - Stores 1024 samples of noisy signal generated by Python.
  - Implemented as Synthesizable ROM.
- **Clock & Reset:**
  - 100 MHz System Clock.
  - Reset mapped to Switch 0 (V17) for manual control.

# Verification: Python Golden Model

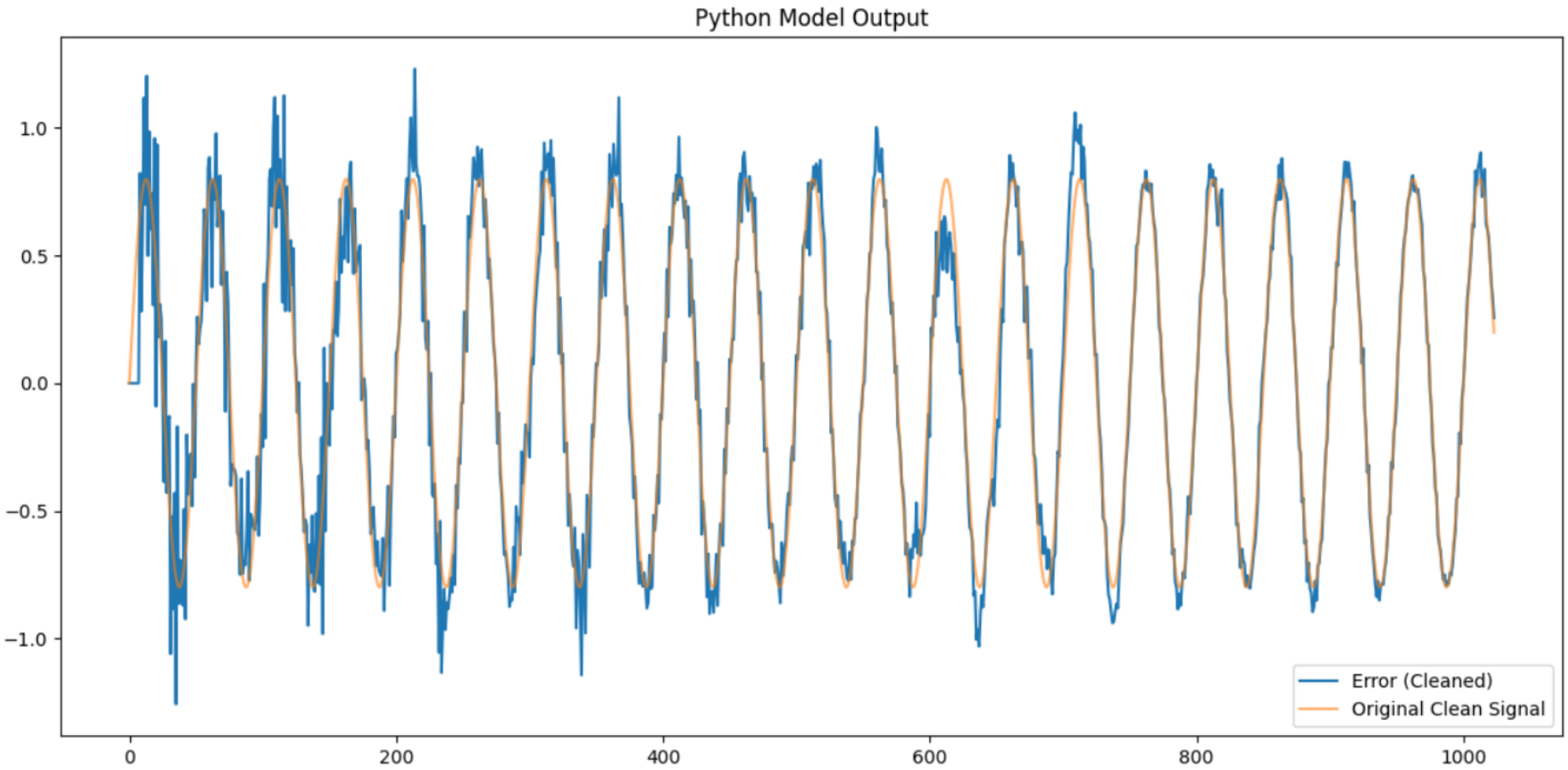



Figure: Python Output: Shows recovery of clean signal from noise.

# Verification: Behavioral Simulation

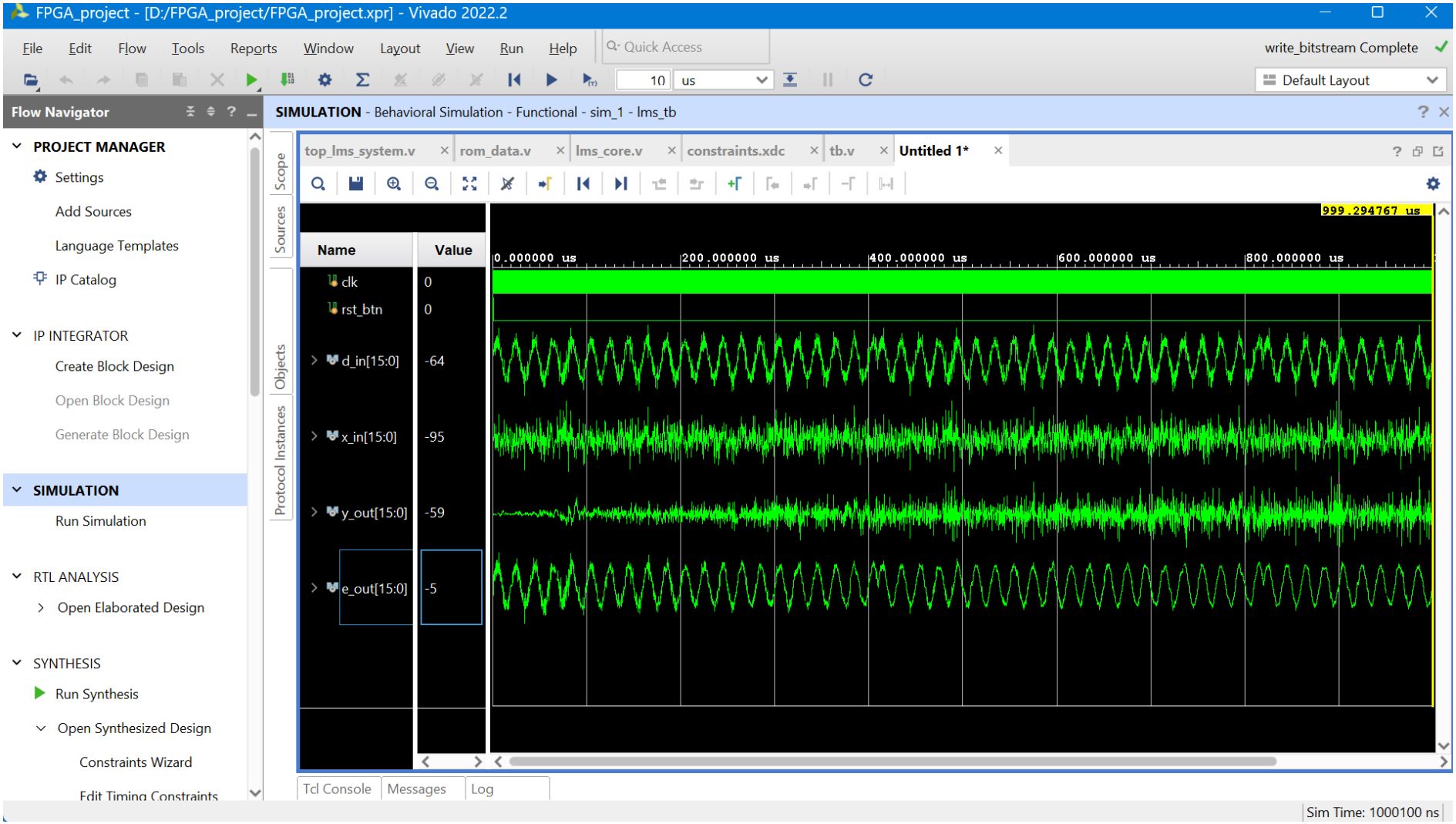



Figure: Vivado Simulation: Weights adapt, $y$ grows, and Error minimizes.

# Hardware Results: ILA Capture


Figure: Real-time ILA Capture. We get the smooth Output vs. Noisy Input.

**Note:** Used ILA Capture Control (‘core_done == 1‘) to capture valid samples.

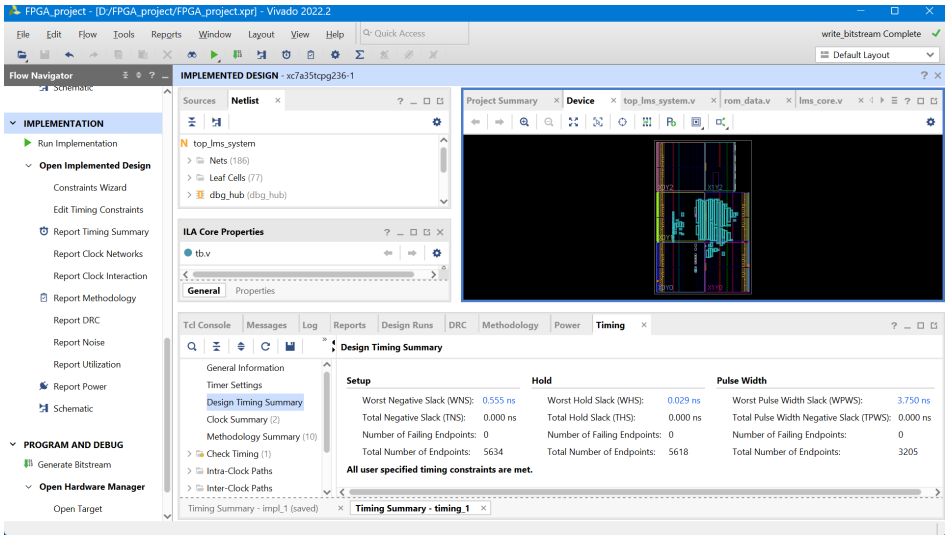
# Key Insights

- **Convergence:** The filter does not cancel noise instantly; it learns over time.
- **Adaptation:** Initially, Error contains noise. As weights update, $y_{out}$ matches the noise profile.
- **Steady State:** After convergence, subtraction yields a smooth clean signal.
- **Verification:** Hardware waveforms match the Behavioral Simulation and Python Model perfectly.

# Post-Implementation: Timing & Critical Path

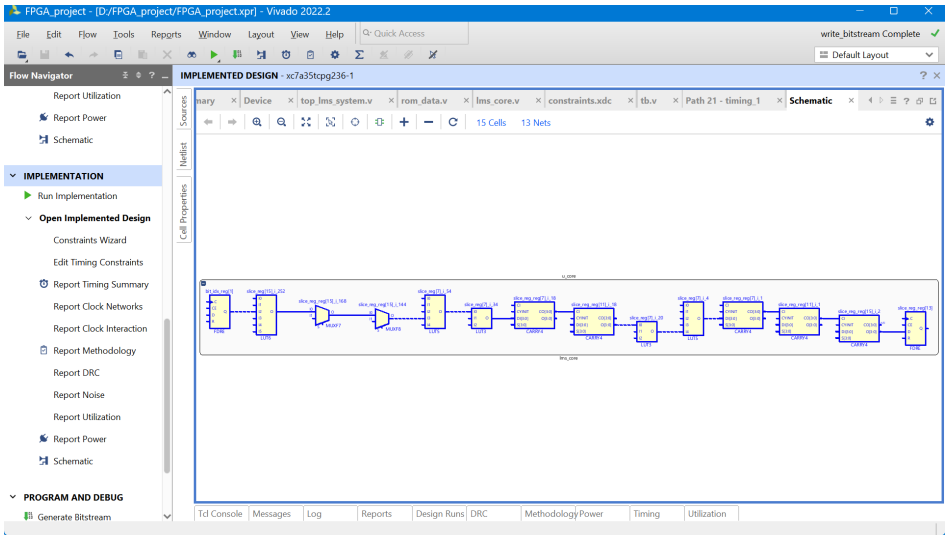
## Timing Summary:

- **WNS:** 0.555 ns (Positive Slack)
- **Target:** 100 MHz (10ns)
- **Status:** Met Constraints




## Critical Path:

- Dominated by OBC Accumulator and Carry Chain.
- Pipelining ensured timing closure.



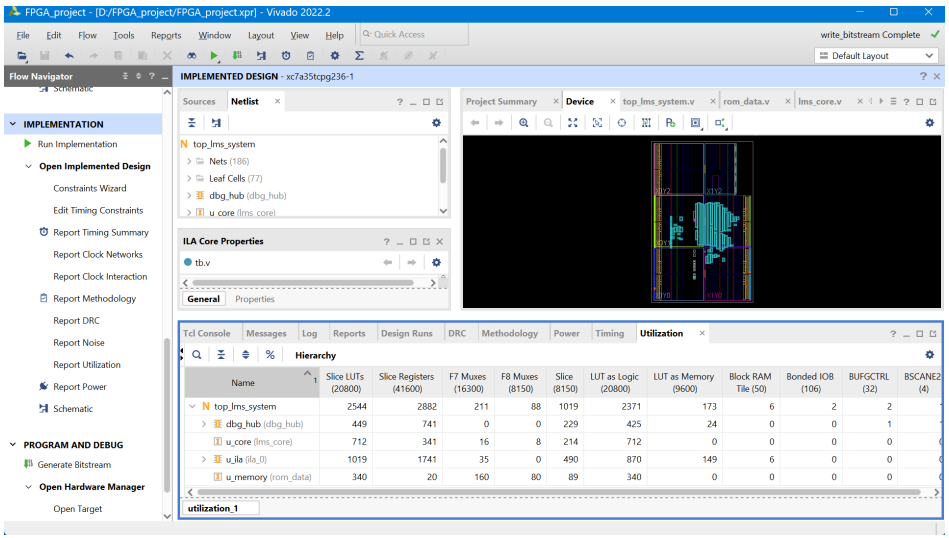
# Post-Implementation: Resource & Power

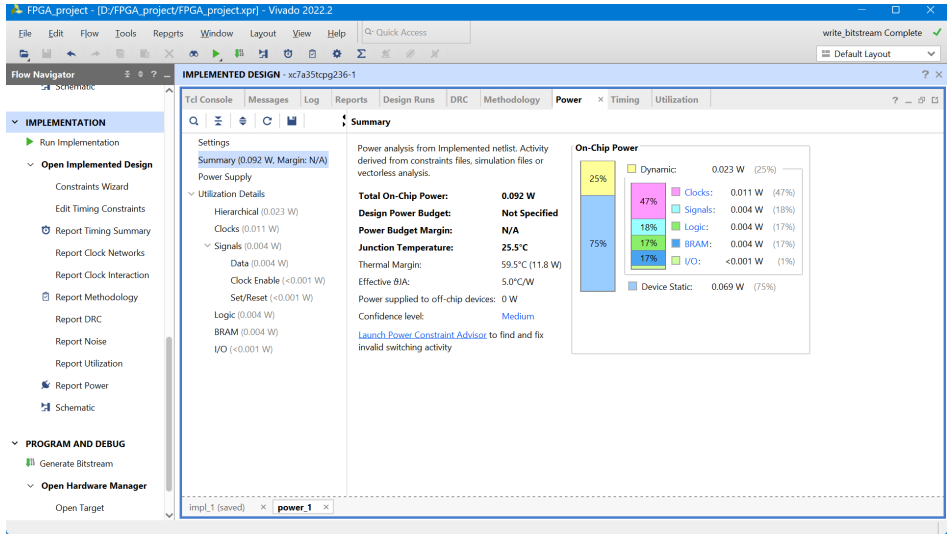
## Resource Utilization:

- Low Logic utilization.
- **Zero DSP Slices** used for filtering (OBC benefit).

## Power Consumption:

- Total On-Chip Power: **0.092 W**
- Suitable for low-power embedded applications.



# Conclusion

- Successfully implemented Adaptive Sign-Error LMS Filter on FPGA.
- **OBC Implementation:** Achieved multiplier-less inner product calculation.
- **Performance:** Met 100 MHz timing constraints with low power consumption (0.092 W).
- **Result:** Hardware output successfully removes noise, matching the Golden Model.

**Code Repository:**

`https://github.com/dipsy32/Adaptive_LMS_FPGA`

# Thank You!