Birdsong Seperation using FIR filters

A PROJECT REPORT

Submitted to
Amrita Vishwa Vidyapeetham

in partial fulfillment for the award of the degree
of
BACHELOR OF TECHNOLOGY IN
ELECTRONICS AND COMMUNICATION ENGINEERING

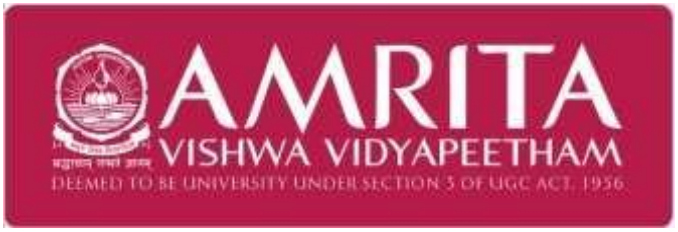
By

Haridoss Bupaal (CH.EN.U4ECE23069)

Surya Narayanan K P (CH.EN.U4ECE23052)

Supervisor

Dr. P MARAN



AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF ENGINEERING
CHENNAI – 601103

SEPTEMBER 2025

1. Introduction

This project presents the design and implementation of a **real-time Birdsong Signal Processing Chain** on an FPGA using Verilog. The main objective is to enhance the quality of birdsong audio recordings by filtering, analyzing, and normalizing the signal while suppressing background noise. The processing chain consists of four key modules:

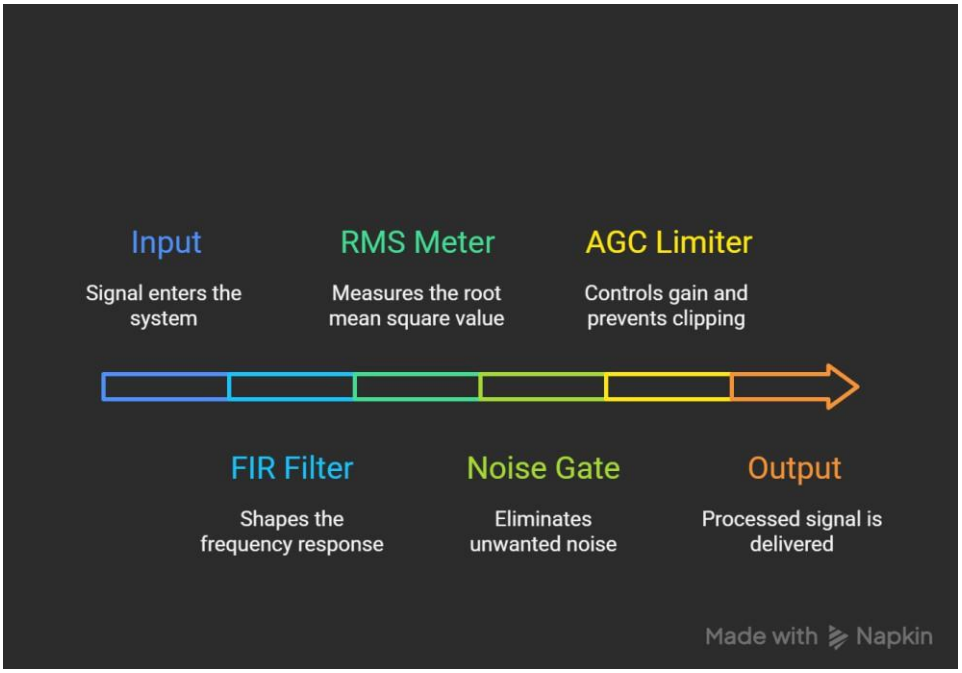
1. **FIR Band-pass Filter (2–6 kHz):** Isolates the frequency range of birdsong from other environmental sounds, ensuring that only relevant audio components are retained.
2. **RMS Meter:** Continuously measures the short-term loudness of the incoming audio signal, providing a basis for dynamic range control.
3. **Noise Gate:** Attenuates low-level background noise below a specified threshold, improving the clarity of the birdsong signal.
4. **AGC Limiter (Automatic Gain Control):** Adjusts signal amplitude in real time to maintain consistent loudness and prevent clipping during peaks.

The system operates on **16-bit signed PCM audio samples in Q1.15 fixed-point format**, allowing precise real-time processing on FPGA hardware. The output is a clean, band-passed, and dynamically normalized birdsong signal, suitable for playback, storage, or further audio analysis in bioacoustic research.

This can be FPGA-based approach provides **low-latency, deterministic processing**, making it ideal for applications requiring real-time audio enhancement in field recordings or embedded systems.

2. System Overview

2.1 Block Diagram



2.2 Features

- FPGA-friendly **Verilog RTL** implementation.
- Configurable FIR filter taps and coefficients (`coeffs.hex`).
- Noise gate with **soft-knee and floor gain** control.
- Automatic Gain Control (AGC) with **attack/release** smoothing.
- MATLAB scripts for **testbench stimulus and output conversion**.

3. Module Descriptions

3.1 FIR Filter (fir_filter.v)

- Implements a **65-tap band-pass FIR filter** (default: 2–6 kHz).
- Coefficients are pre-computed in MATLAB and loaded via `coeffs.hex`.
- Fixed-point arithmetic: Input/Output in **Q1.15**, internal MAC in **Q2.30**.

3.2 RMS Meter (`rms_meter.v`)

- Computes a running RMS approximation using an **IIR filter**.
- Output: unsigned **Q1.15** estimate of short-term loudness.
- Configurable smoothing factor (`ALPHA_SHIFT`).

3.3 Noise Gate (`noise_gate.v`)

- Attenuates signals below a threshold (`THRESH_Q15`).
- Supports **hard gating** or **soft gating with floor gain**.
- Soft-knee transition defined by `SOFT_KNEE_RANGE`.

3.4 AGC Limiter (`agc_limiter.v`)

- Maintains target loudness (`TARGET_Q15`).
- Uses attack/release smoothing to follow envelope changes.
- Gain clamped between `MIN_GAIN_Q15` and `MAX_GAIN_Q15`.
- Includes a final **hard limiter** to prevent clipping.

3.5 Top-Level Chain (`birdsong_chain.v`)

- Connects all modules in sequence.
- Single clock (`clk`) and reset (`rst`).
- Input: `x_in` (Q1.15 PCM).
- Output: `y_out` (processed Q1.15 PCM).

4. Testbench

File: `tb_birdsong_chain.v`

- Stimulus: reads input samples from `birdsong.hex` (16-bit signed).
- Captures processed output to `output.hex`.
- Clock: 100 MHz (10 ns period).
- Simulation flow:
  - Reset system.
  - Apply input samples.
  - Capture processed output.
  - Save to `output.hex`.

5. MATLAB Scripts

5.1 `birdsong_to_hex.m`

- Reads a `.wav` file (`birdsong.wav`).
- Normalizes signal and writes **16-bit Q1.15 PCM samples** to `birdsong.hex`.
- Designs band-pass FIR filter (2–6 kHz, 65 taps).
- Normalizes FIR passband gain to unity.
- Saves coefficients in `coeffs.hex`.
- Generates a preview file `filtered_preview.wav` (software reference).

5.2 `hex_to_wav.m`

- Reads simulation output (`output.hex`).
- Converts back to PCM and saves as `birdsong_out.wav`.
- Allows quick listening and duration validation.

6. Usage Guide

6.1. Project Setup

- Open **Vivado** → **Create New Project**.
- Select **RTL Project** → Check *Do not specify sources currently* if you want to add them later.
- Choose a project name and location (e.g., `birdsong_chain_project`).
- Select your target FPGA device/board.

6.2. Add Source Files

You have two categories of Verilog files:

- **Design Sources (RTL):**

```
birdsong_chain.v
fir_filter.v
rms_meter.v
noise_gate.v
agc_limiter.v
```

- **Simulation Sources (Testbench):** `tb_birdsong_chain.v`

Steps

1. In the **Sources window** → Right-click **Design Sources** → *Add Sources* → Add the five RTL files.
2. Right-click **Simulation Sources** → *Add Sources* → Add the testbench file.

6.3. Add Memory Initialization Files

Two .hex files are required for simulation:

- coeffs.hex → Filter coefficients.
- birdsong.hex → Audio input samples.[**save them in srcs or sim**]

Steps

1. In the **Sources window** → Right-click **Simulation Sources** → *Add Sources*.
2. Select **Add or Create Simulation Sources**.
3. Browse and add both paths of coeffs.hex and birdsong.hex.
4. Confirm they appear under Simulation Sources.This ensures $readmemh("filename.hex", …) can find them at runtime.

6.4. Simulation Execution

- From the **Flow Navigator**, select **Run Simulation → Run Behavioral Simulation.**
- Vivado will compile the RTL, load the .hex files, and start the testbench.
- During simulation:
  a. tb_birdsong_chain.v reads input samples from birdsong.hex.
  b. fir_filter.v loads FIR coefficients from coeffs.hex.
  c. Processed samples are written to a new file:
     fh = $fopen("output.hex", "w");
     $fwrite(...);

6.5. Locating the Output File

- Vivado saves output.hex in the **simulation working directory.**
- Default path (for XSIM):***<project_dir>/simulations/sim_1/behav/xsim/***
- Verify after simulation completes; output.hex should contain 16-bit signed samples in hex format.

6.6. Post-Processing

- Copy output.hex back into your MATLAB workspace.
- Run hex_to_wav.m to convert the hex samples into a playable .wav file.
- Validate by listening to birdsong_out.wav and comparing with your MATLAB preview file.

7. Workflow Summary

- Prepare audio input (birdsong.hex) and FIR coefficients (coeffs.hex) in MATLAB.
- Add Verilog RTL and testbench files to Vivado.
- Add .hex data files as **simulation sources**.
- Run behavioral simulation.
- Retrieve output.hex from Vivado's simulation directory.
- Convert output.hex back to .wav in MATLAB for validation.

REFERENCE:

birdsong-audio-on-ZedBoard : Refer for the code

xup_high_level_synthesis_design_flow : This is the main workshop/tutorial repository for using Vitis HLS to target boards like PYNQ-Z2, PYNQ-ZU, KV260.