**Audio Pre-Processing System**

**1. Introduction**

The advancement in digital signal processing has revolutionized audio applications, allowing the enhancement of sound quality and the application of real-time effects such as bass, treble, bandpass filtering, and echo. This project aims to design an audio processing system that uses FIR filters and FFT to apply these effects efficiently, offering high-quality audio processing for both embedded and standalone systems.

**2. Motivation and Scope**

**Motivation**

Audio processing is essential in various fields such as music production, telecommunications, and entertainment systems. Achieving precise control over sound properties like bass and treble while offering advanced effects like echo can enhance user experience significantly. This project is motivated by the need for real-time, efficient, and configurable audio processing systems.

**Scope**

The designed system will:

1. Process audio in real time.
2. Allow flexible control over audio properties like bass, treble, and bandpass.
3. Add echo effects using feedback mechanisms.
4. Be adaptable for integration into audio devices, mobile applications, and embedded systems.

**3. Problem Statement**

Traditional audio systems often lack the flexibility to provide customizable sound effects in real time. Additionally, combining multiple effects such as filtering and echo into a unified system poses challenges due to the complexity of control and data synchronization.

**Challenges Identified:**

* Real-time processing of audio data with low latency.
* Integration of multiple functionalities (bass, treble, bandpass, echo).
* Efficient control and synchronization of operations.

**4. Suggested Solution**

This project proposes an audio processing system based on **FIR filters**, **FFT (Fast Fourier Transform)**, and **MUX-controlled datapath selection**. The system will:

1. Use FIR filters for bass, treble, and bandpass processing.
2. Employ FFT for frequency-domain analysis, enabling enhanced effects.
3. Utilize an FSM to control the data flow and processing stages dynamically.
4. Incorporate a flexible datapath with multiplexers to handle different audio effects.

**5. Complete Methodology**

**5.1. Block Diagram**

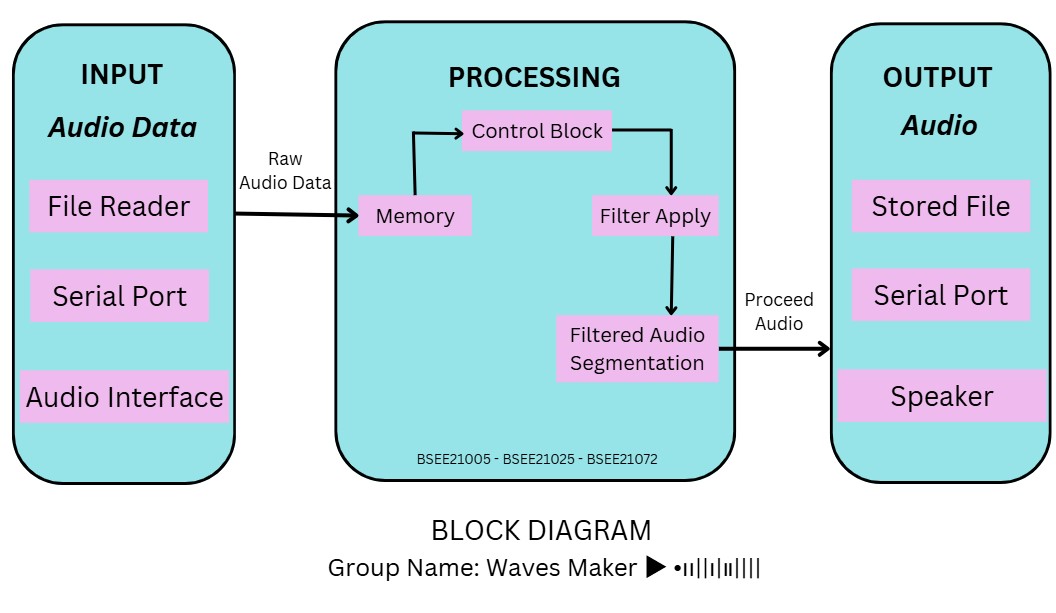


Fig I: show Block Diagram \*this diagram is made by using Canva

**Description of the Block Diagram**

The functional block diagram for the Audio Pre-Processor project represents the flow of audio data through three main modules:

**Input Module:**

* This module captures raw audio data either from a stored file or via a serial port.
* It formats the input data for further processing.

**Processing Module:**

The core module where audio enhancement occurs.

It consists of sub-blocks for filtering and effects:

* **Bass Filter:** Enhances low frequencies.
* **Treble Filter:** Boosts high frequencies.
* **Band-Pass Filter:** Isolates a specific frequency range.
* **Echo Effect (optional):** Adds echo to the audio.

Data flows sequentially or selectively through the required filters.

**Output Module:** • Processes and outputs the enhanced audio data.

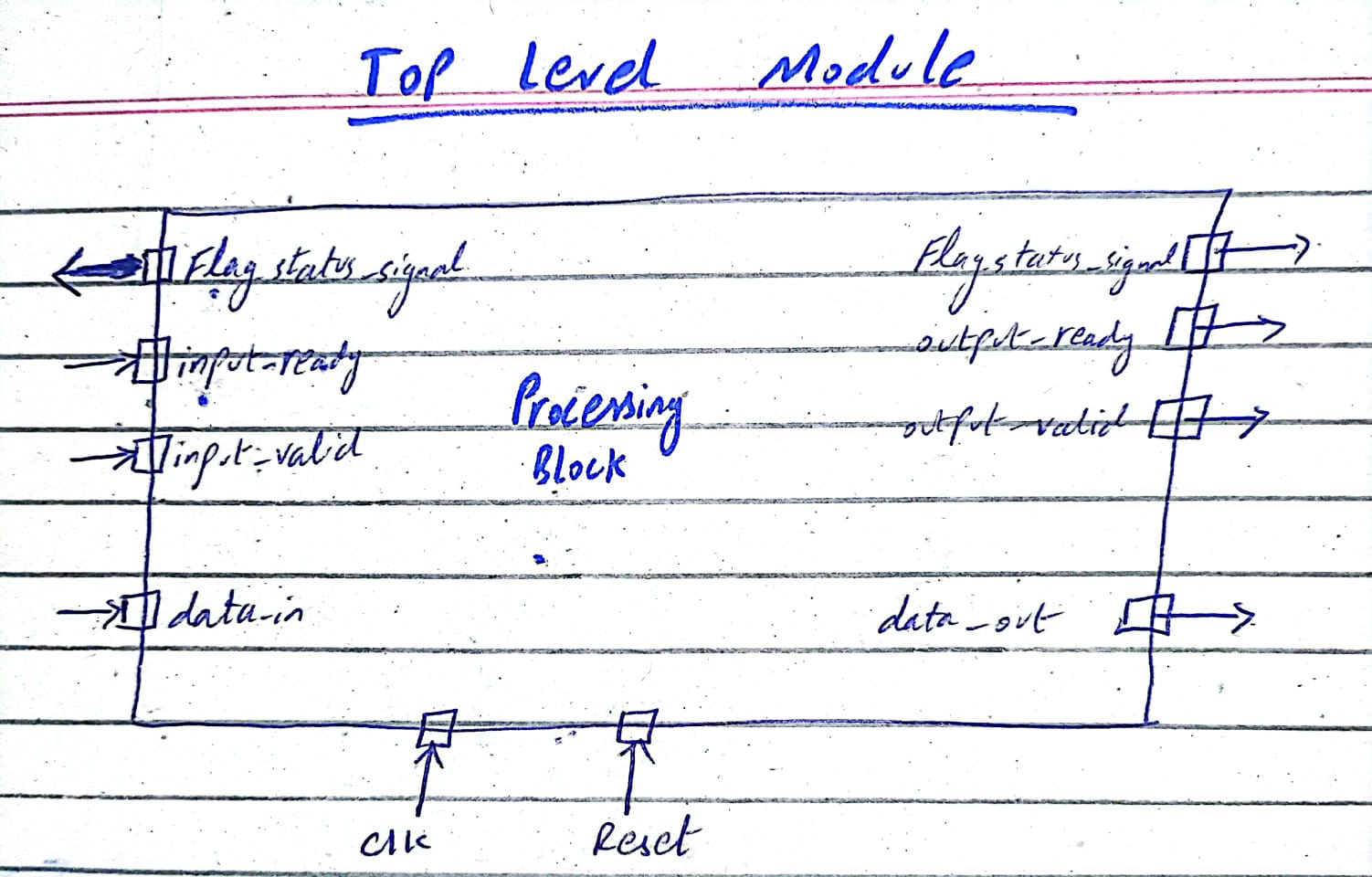
* Provides options for saving the audio to a file, transmitting it via a serial port, or playing it through a speaker.

**5.2. Datapath**

The datapath integrates the following:

* **Input Stage**: Reads data from DRAM.
* **Processing Stage**: Passes data through FIR filters (bass, treble, bandpass) or FFT for echo.
* **Control Signals**: MUX handles the selection of effects, controlled by the FSM.
* **Output Stage**: Writes processed data to DRAM or sends it to output devices.

**Datapath Illustration**:



A diagram of a circuit

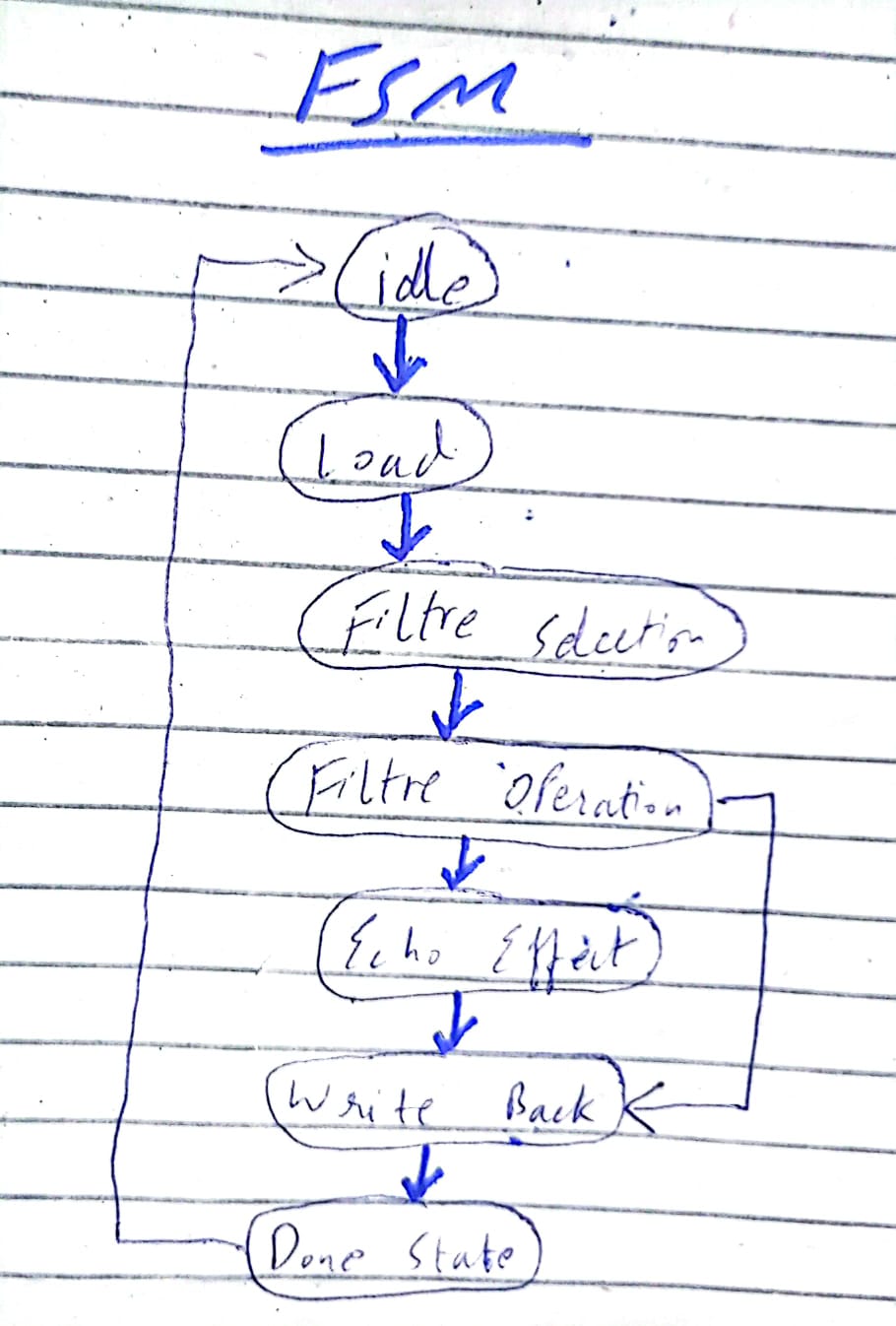
Description automatically generated with medium confidence

**5.3. FSM Design**

The FSM governs the overall operation of the system. It transitions through the following states:

1. **Idle**: Waits for the start signal.
2. **Load**: Loads input audio data.
3. **Filter Selection**: Configures the datapath to apply bass, treble, or bandpass filters.
4. **Processing**: Executes the selected filter operation.
5. **Echo Effect**: Applies echo to the processed data (if enabled).
6. **Write Back**: Outputs the processed data.
7. **Done**: Completes the operation and resets.

**FSM Diagram**:



**5.4. Implementation Steps**

**Step 1: Design the Block Diagram**

* Use DRAM to store input/output data.
* Add FIR filters and FFT modules for processing.
* Include MUX for selecting the desired operation.
* Add control and write-back units.

**Step 2: Develop the Datapath**

* Define connections between DRAM, filters, and output.
* Configure MUX for filter selection.
* Ensure proper routing of data using control signals.

**Step 3: Design the FSM**

* Define states based on functionality.
* Write state transition logic and control signal outputs.
* Integrate the FSM with the datapath for overall operation.

**Step 4: Test and Simulate**

* Simulate each component (filters, FFT, MUX) independently.
* Integrate and test the FSM with the datapath.
* Validate the system using test audio data.

**6. Desired Outcome**

**Objective**

The project aims to deliver a real-time audio processing system capable of:

* Enhancing sound with adjustable bass, treble, and bandpass filters.
* Adding echo effects with feedback control.
* Efficiently managing data flow using FSM and MUX.

**Expected Benefits**

1. Improved sound quality through precise filtering.
2. Real-time, low-latency processing for dynamic applications.
3. Adaptability for integration into various audio devices.