FPGA Audio Looping Station

Technical Design Document

Multi-Channel Audio Loop Station on an FPGA

CPRE 4880 Final Project

5/14/2026

John Brittain, Ian Runestad, Seth Klaassen, Enming Wang

Table of Contents

System Overview

This project implements a multi-channel audio looping station on a Zynq 7000 FPGA. The system records, stores, and plays back multiple audio loops simultaneously across 4 independent channels, functioning similarly to commercial loop station products such as the Boss RC-600 and RC-1. A key feature is a two-mode loop length arbitration system that enables both synchronized and independent loop lengths across tracks, giving the performer significant creative flexibility. User control is provided by an external ESP32-based pedal board connected to the Zynq via UART. The board exposes five footswitches (SW1-SW4 for tracks 1-4, SW5 for global reset) and ten LEDs (one red/green pair per switch) that reflect each track state in real time.

Requirements

- Record audio input and store it as a looping clip on any of 4 independent channels.
- Play back all active channels simultaneously with seamless, gapless looping.
- Support overdub: layer new audio on top of an already-playing loop.
- Accept audio from 2 independent input sources simultaneously.
- Provide per-channel volume control adjustable by the user in real time.
- All channel state changes (record / overdub / monitor) triggered via an external ESP32 footswitch pedal board connected to the Zynq over UART.
- Audio sample rate of 48 kHz, 16-bit on all channels.
- Minimum 1 minute of loop storage per channel using onboard DDR3 RAM.
- Two operating modes: Synchronized (global loop length) and Independent (per-track loop length).

Each footswitch has a dedicated red/green LED pair providing real-time visual feedback of the corresponding track state: IDLE = both off, RECORDING = red, OVERDUB = red + green, MONITOR = green.

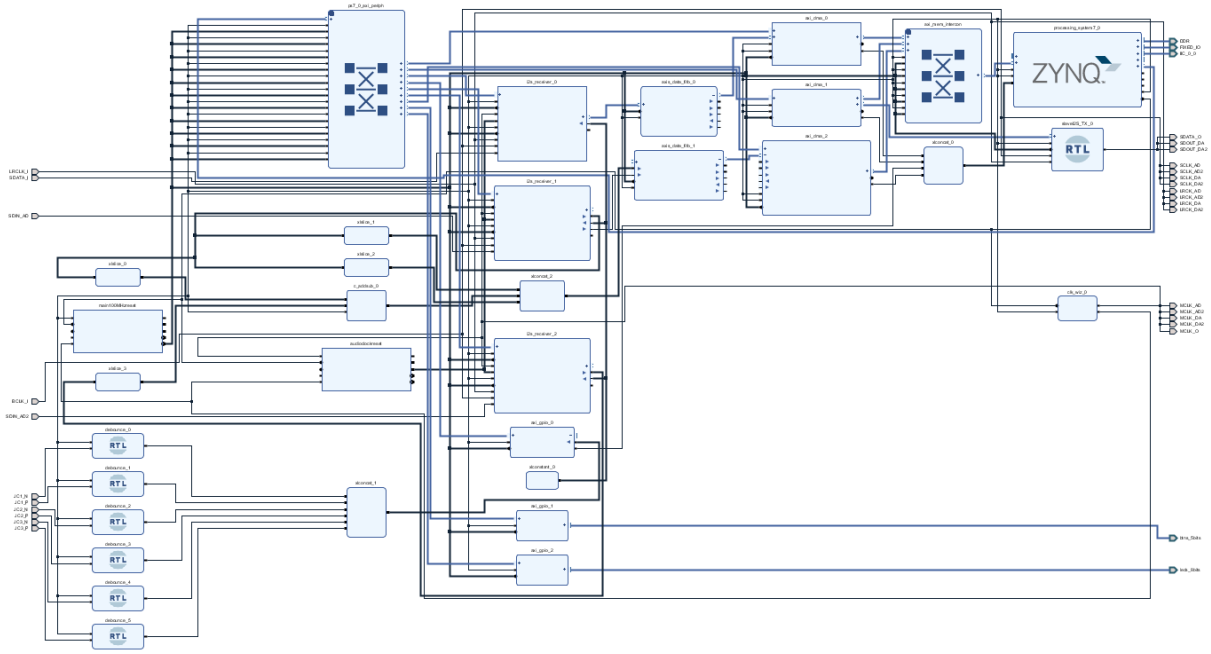
Component Summary

Component	Role
DDR3 RAM	Loop storage — minimum 1 minute per channel at 48 kHz / 16-bit
I2S Interface	Audio input/output at 48 kHz, 24-bit (downsampled to 16-bit internally)
ESP32 Pedal Board	5-switch footswitch controller with per-switch red/green LEDs; hardware-debounced switch events sent to Zynq over UART; receives LED state commands from Zynq over the same link
Track Behavior FSM	Per-track state machine: IDLE → RECORDING → OVERDUB ↔ MONITOR

Track Timer FSM	Per-track clock-tick counter measuring loop length during recording
Loop Length Arbiter FSM	Determines final loop_len per track based on operating mode
UART Interface (UART1)	Serial link at 115200 baud between Zynq UART1 (micro-USB) and ESP32; carries (SwiXOn) press events inbound and (RedLXOn/Off) / (GrnLXOn/Off) LED commands outbound

Hardware Overview

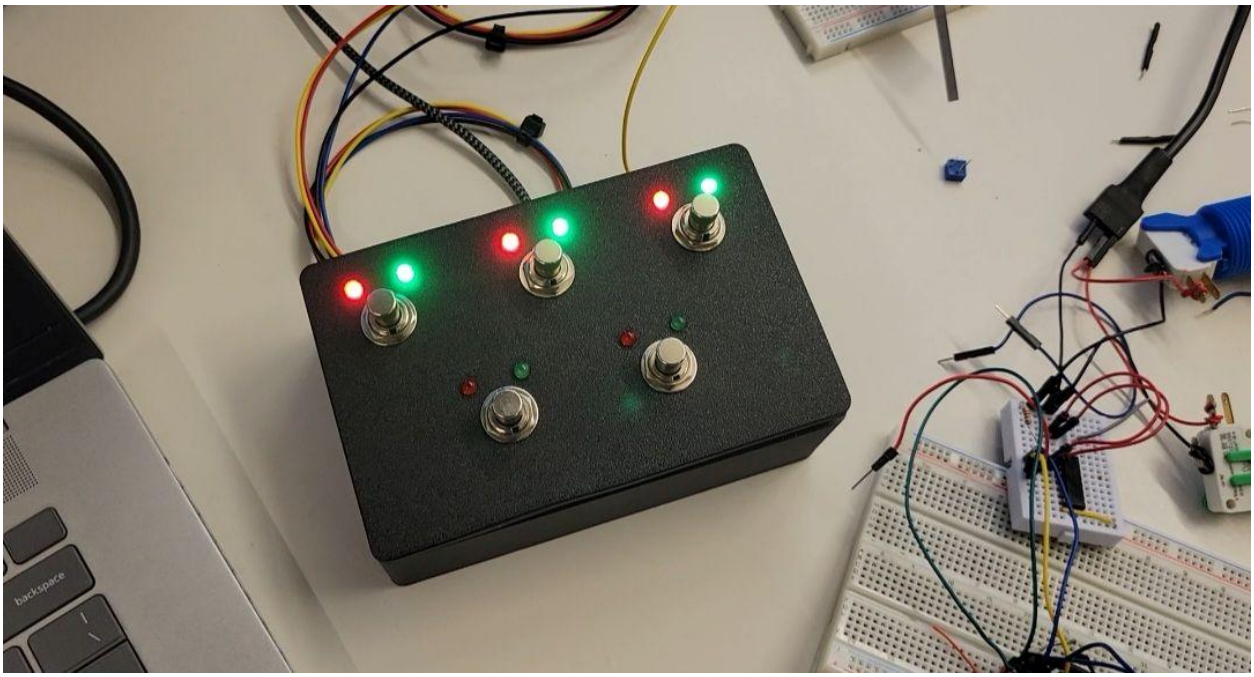
System Architecture



Vivado Block Design

Pedal Board Interface

The pedal board communicates with the Zynq over UART1 at 115200 baud using a simple parenthesis-framed ASCII protocol. The ESP32 handles all mechanical debouncing (30 ms window) before transmitting. The Zynq firmware parses incoming messages each audio chunk iteration and sends LED update commands whenever a track state transition occurs.



Pedalboard box

LED State Mapping

Each switch has one red LED and one green LED. Red indicates an active recording operation; green indicates active playback. The combined state of both LEDs uniquely encodes all four track states at a glance.

UART Protocol

Outbound (Zynq to ESP32): LED commands formatted as (RedLXOn), (RedLXOff), (GrnLXOn), (GrnLXOff) where X is switch number 1-5. Sent only on state change. Inbound (ESP32 to Zynq): switch press events formatted as (SwiXOn) where X is 1-5. SW1-SW4 map to tracks 1-4; SW5 triggers global reset.

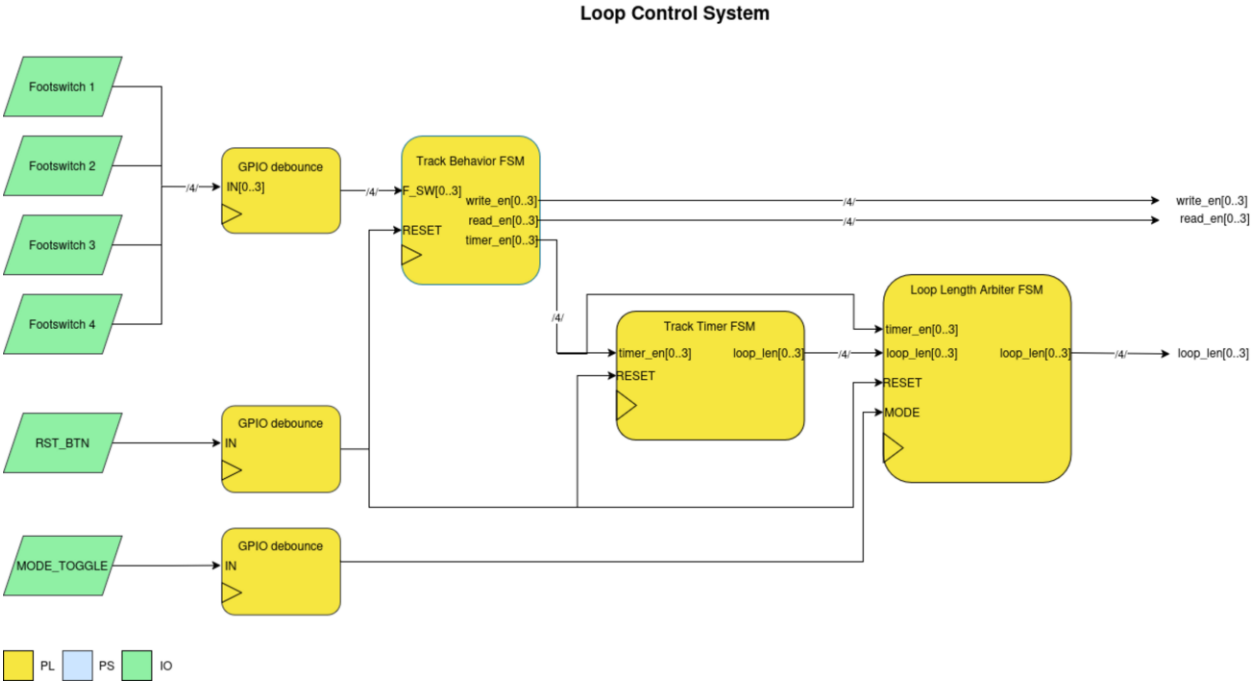
Development / Standalone Connection

During development, the ESP32 USB serial port and the ZedBoard UART1 micro-USB port are both connected to a host PC. A lightweight Python bridge script forwards bytes between the two COM ports, eliminating the need for direct board-to-board wiring during prototyping. For standalone deployment the ESP32 is wired directly to the Zynq UART0 PMOD pins.

Loop Control System

The loop control system handles all user input and coordinates the three FSMs that manage recording state, loop length measurement, and loop length arbitration. Footswitch press events

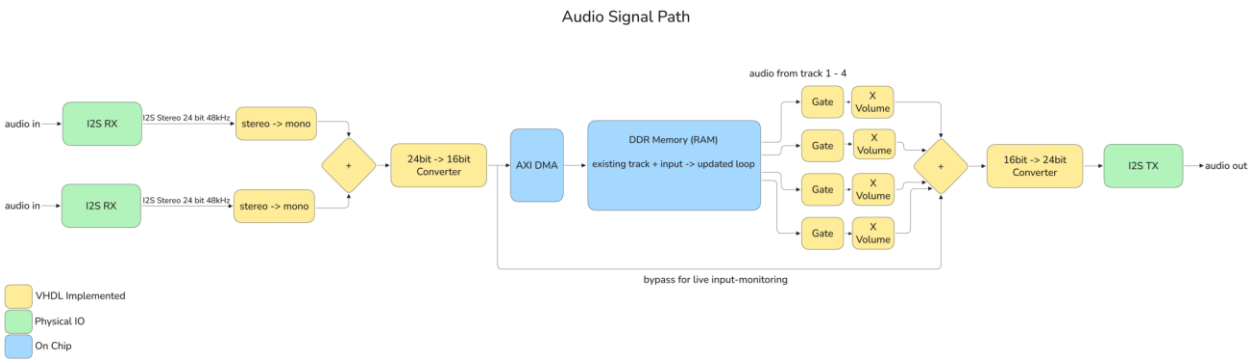
arrive from the ESP32 pedal board over UART and are parsed each audio chunk iteration. The Track Behavior FSM processes these events and drives timer_en and write/read enables. The Timer FSM counts clock ticks per track while recording is active. The Loop Length Arbiter determines the final loop_len per track based on the current operating mode. LED state is updated over UART whenever a track state transition occurs.



Loop Control System Block Diagram

Audio Signal Path

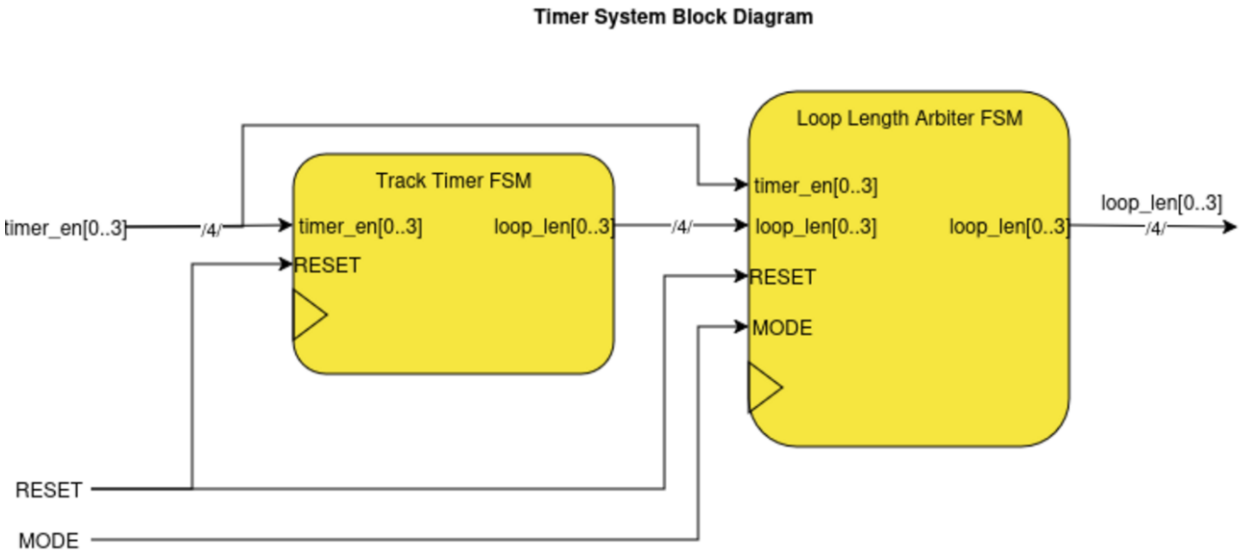
Audio enters via two I2S RX interfaces at 24-bit stereo, 48 kHz. Each stream is downmixed to mono and the two are summed before being truncated to 16-bit and written to DDR memory via AXI DMA. The DDR block mixes existing loop audio with new input for overdubbing. On playback, each of the four tracks passes through a Gate (controlled by read_en) and a Volume multiplier before being summed and upconverted to 24-bit for I2S TX output. A bypass path allows live input monitoring regardless of loop state.



Audio Signal Path

Timer System

The timer system is composed of the Track Timer FSM and the Loop Length Arbiter FSM. The Track Timer FSM runs one timer instance per track, counting clock ticks while timer_en is high (during recording). When timer_en falls, the timer latches its count and reports it to the Arbiter. The Arbiter then applies the active mode policy to determine the final loop_len output for all four tracks.



Timer System Block Diagram

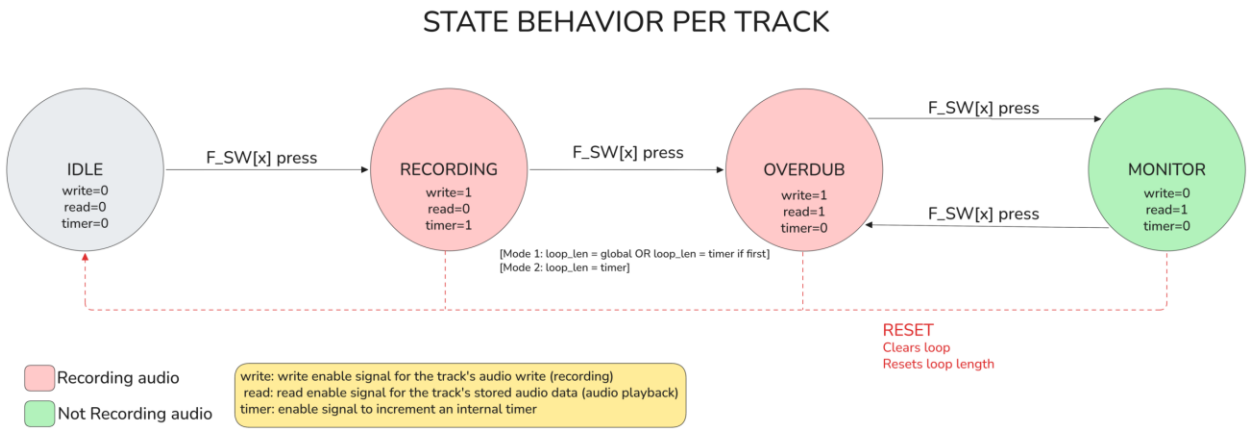
FSM Behavior

Track Behavior FSM

One instance is shared across all four tracks, with each footswitch on the ESP32 pedal board independently cycling its corresponding track through states. Switch press events arrive as UART messages (SwiXOn) and are dispatched to the appropriate track FSM. The FSM drives write_en, read_en, and timer_en signals that control both the audio path and the timer system.

State	write	read	timer	Description
IDLE	0	0	0	Track inactive, no audio written or read
RECORDING	1	0	1	Audio written to DDR, timer counting loop length
OVERDUB	1	1	0	New audio layered onto existing loop, loop playing back
MONITOR	0	1	0	Loop plays back only, no new recording

State transitions are triggered by receipt of a (SwiXOn) UART message. OVERDUB and MONITOR toggle back and forth on each press. SW5 (global reset) from any state returns all tracks to IDLE and clears loop data. Loop mode is set at startup (default: Mode 0 - Synchronized); there is no dedicated mode toggle switch in the current hardware.

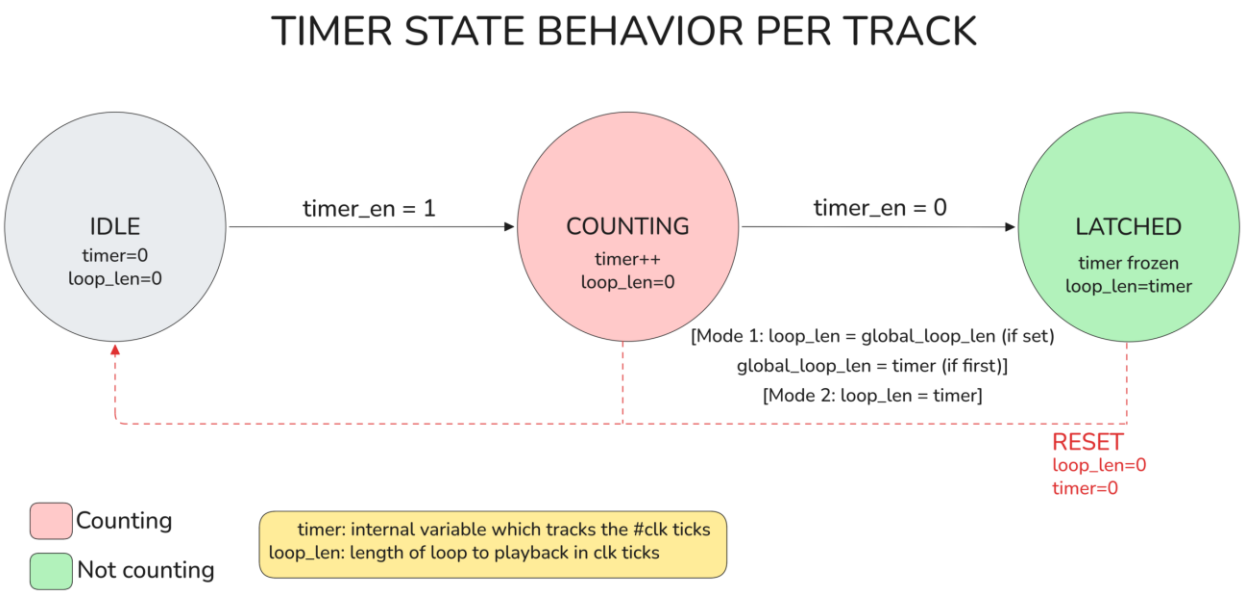


Track Behavior State Diagram

Track Timer FSM

One timer FSM instance per track. The timer counts clock ticks (at the system clock rate) while timer_en is high. On the falling edge of timer_en, the count is frozen and reported as loop_len to the Loop Length Arbiter. RESET clears both the timer and loop_len registers.

State	timer	loop_len	Description
IDLE	0	0	Waiting for timer_en to go high
COUNTING	timer++	0	Incrementing on every clock tick
LATCHED	frozen	timer	Count frozen and available to Arbiter

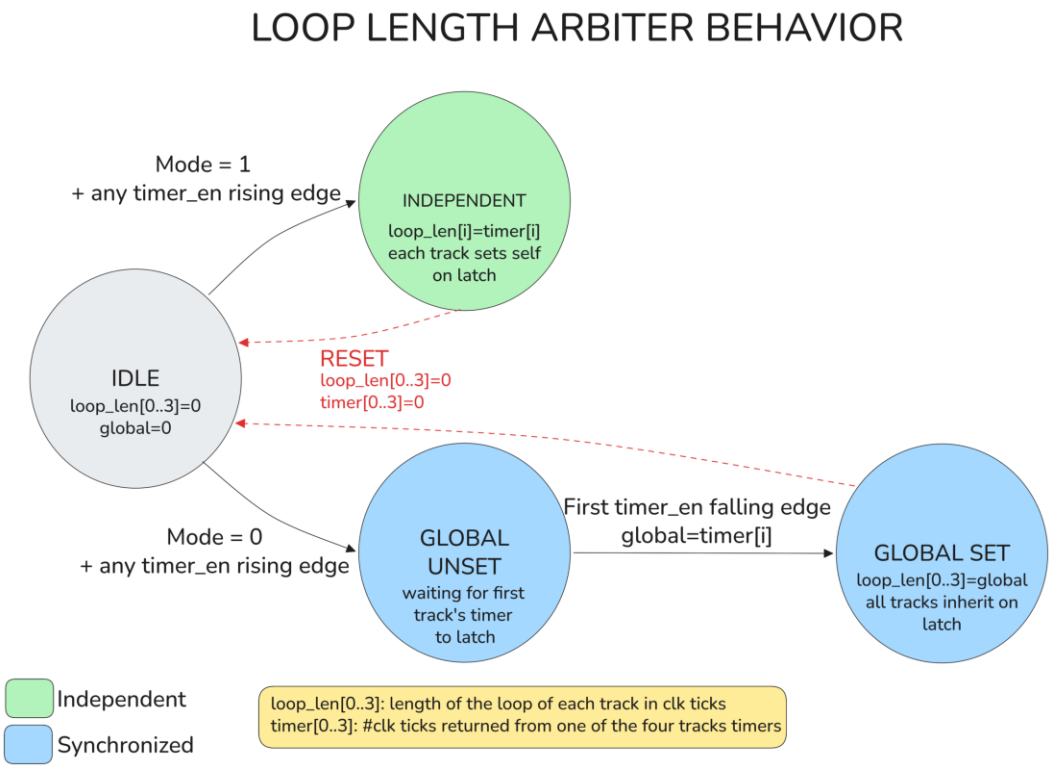


Timer Behavior State Diagram

Loop Length Arbiter FSM

The Arbiter is the most architecturally significant component of the design. It takes the four timer values from the Track Timer FSMs and determines the final loop_len for each track based on the MODE input. This enables the two core operating modes of the device.

State	Description
IDLE	All loop_len and global values zeroed. Waiting for any track to begin recording.
INDEPENDENT	Mode=1. Each track sets its own loop_len from its own timer on latch.
GLOBAL_UNSET	Mode=0. Waiting for the first track to finish recording and set the global length.
GLOBAL_SET	Mode=0. Global length established. All subsequent tracks inherit this loop_len on latch.



Loop Length Arbiter Behavior State Diagram

Operating Modes

The operating mode determines how loop lengths are coordinated across tracks. In the current hardware implementation the mode is fixed at startup via firmware (default: Mode 0 - Synchronized). Mode switching is available in software and can be re-enabled by mapping a secondary input gesture to the loop_mode flag.

Mode 0 - Synchronized

The first track recorded sets a global loop length. All subsequent tracks automatically inherit this length when they finish recording, regardless of when the footswitch is pressed. This locks all tracks to the same rhythmic grid, making it ideal for building up cohesive, layered loops in a live performance context - analogous to standard looper pedal behavior.

Mode 1 - Independent

Each track measures and keeps its own loop length independently. Tracks are not time-locked to one another, allowing different loop lengths to coexist simultaneously. This enables the performer to build a full song arrangement of distinct sections - a short repeating figure on one track, a longer chord progression on another - and trigger them as needed. This mode turns the device from a traditional looper into a live arrangement and composition tool.

I2S Interface

Configuration

Parameter	Value
Role	Slave — LRCK and SCK generated by host (Zynq 7000)
MCLK	12.2880 MHz
LCLK (Sample Rate)	48 kHz
Bit Depth	24-bit per channel (input); 16-bit internal storage; 24-bit output

Timing Details

The I2S protocol requires data to be clocked on the FALLING edge of SCLK. The first bit of data is not clocked in until the first complete serial clock cycle has passed after LRCK changes state. This results in a one-cycle delay that must be accounted for in all shift register implementations - other samples will be offset by one bit if this is not handled correctly.

Output Specification

Parameter	Value
Resolution	24-bit per channel
Sample Rate	48 kHz
Channels	Stereo (L + R)

References

[1] Boss, "RC-600 Loop Station," https://www.boss.info/global/products/rc-600/

[2] Boss, "RC-1 Loop Station," https://www.boss.info/global/products/rc-1/

[3] Xilinx / AMD, "Zynq-7000 SoC Technical Reference Manual," https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf

[4] JEDEC, "Serial Peripheral Interface Standard I2S," https://www.sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf