# ECE 378 Design Project: Audio Looper

Prof. Daniel Llamocca
William Blackburn, Syed Ahmed,
Geoffrey Williston

# Project Description

- The ideal goal was to create an audio looper and recorder
- Utilized switches, memory interface, clocks, counters, mux, combinatorial and sequential circuits, FSM, audio I/O
- Memory access

Why choose this project?

- Wanted to better understand ADC
- This project has real world applications

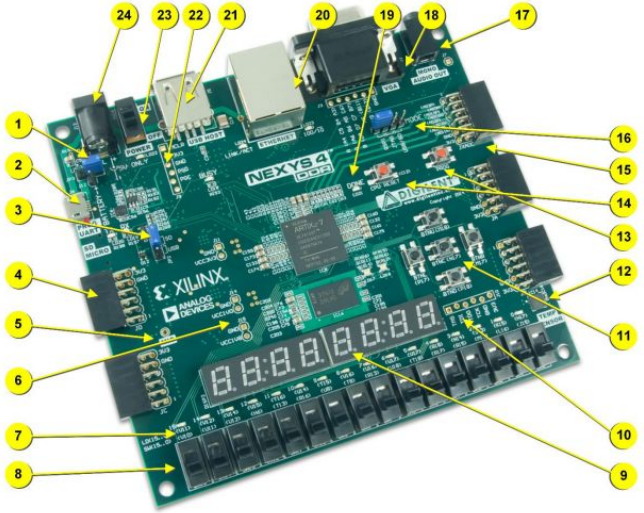



*Figure 1. Nexys4 DDR board features.*

# Project Specifics

- Half leds lit up = recording
- Full leds = playback
- Switches 12-15 used to select which track to record to
  - No track selected = no recording
- Switch 0 displays track on hex7seg display
- Switch 1 records the track
- Switch 2 plays back the selected track
  - If no tracks selected or contains no data = no output

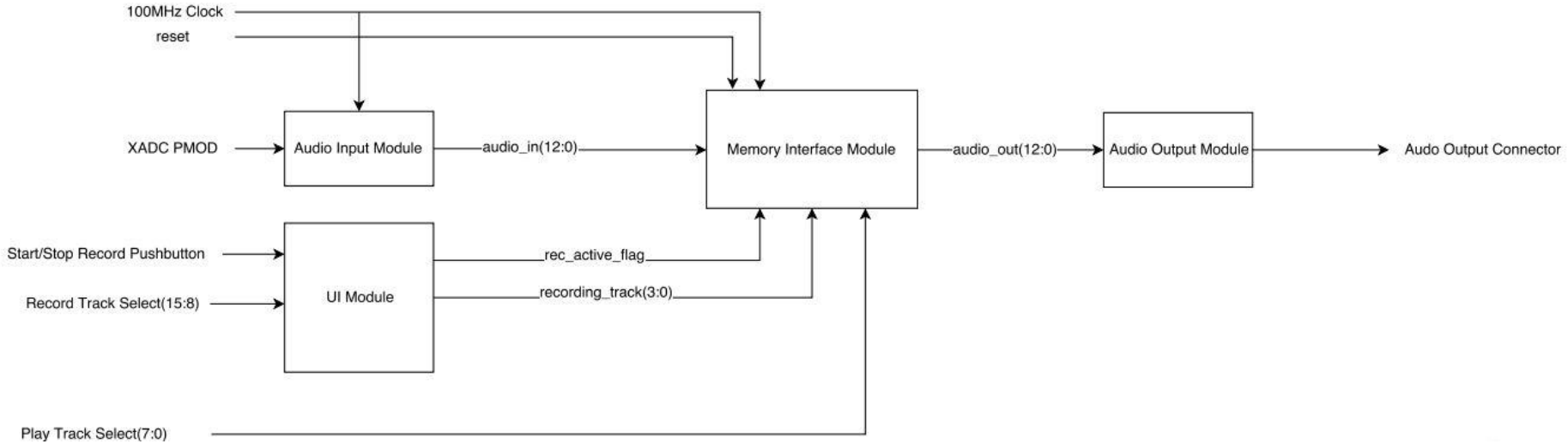
# Project Specifics

- A toggle switch used to start and stop recording.  If selected track contains previous recording data, it’s overwritten.
- Another toggle switch will be used to cancel active recording.  If selected track contains data from a previous recording, it’s kept.
- Audio output jack used for audio playback

# Project Plan

- Develop a project outline
- Block diagram top file
- Split VHDL into distinct modules
  - Memory
  - UI
  - Audio In
  - Audio Out
- Flowchart the UI and memory modules
- Implement in VHDL and debug

# VHDL: Top File



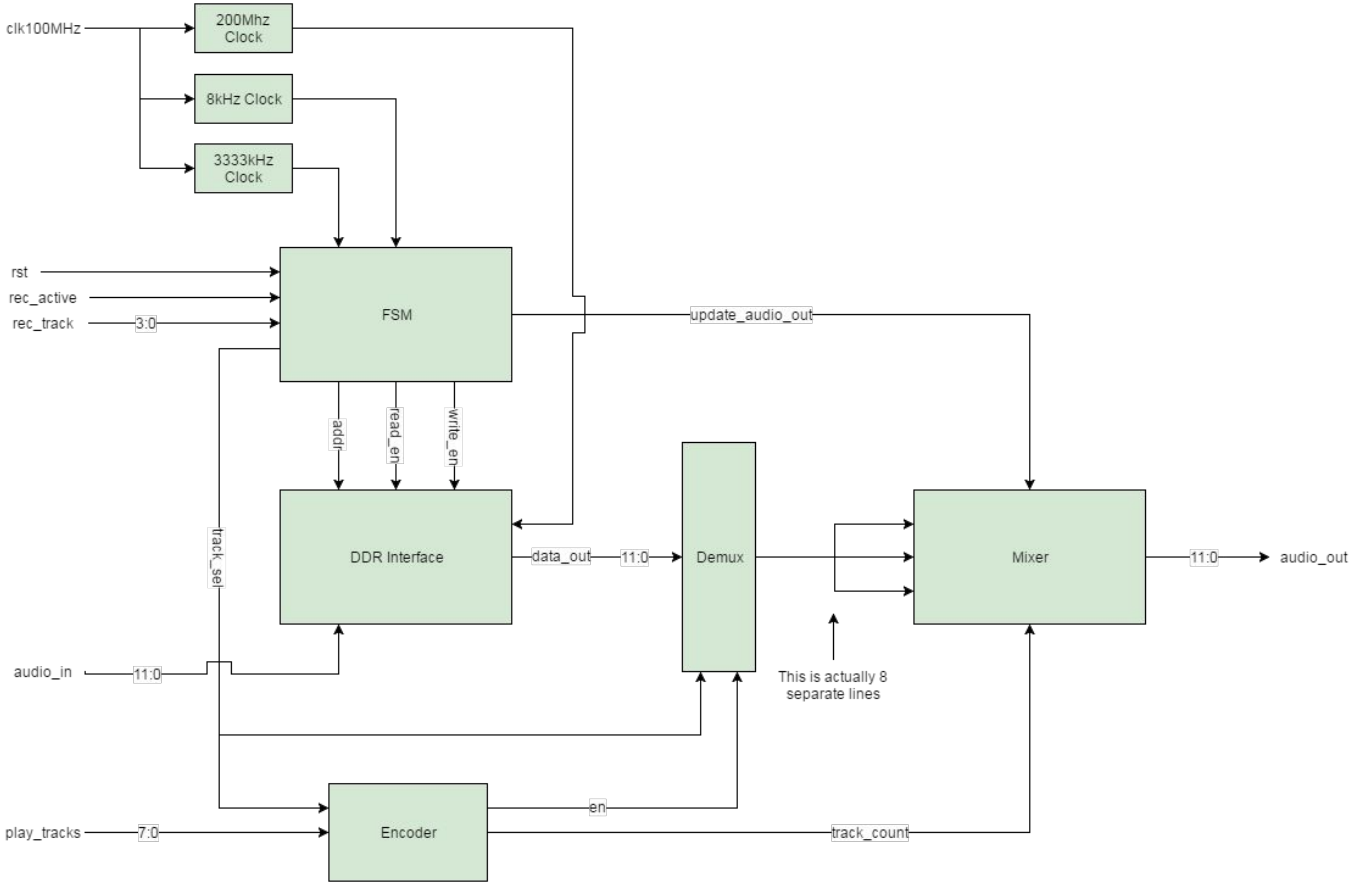
# VHDL: UI Module

- Looked at tracks to be recorded
- If record button was pressed and if only one track was selected, then it sent a recording active flag for sampling time and 3 bits specifying the track

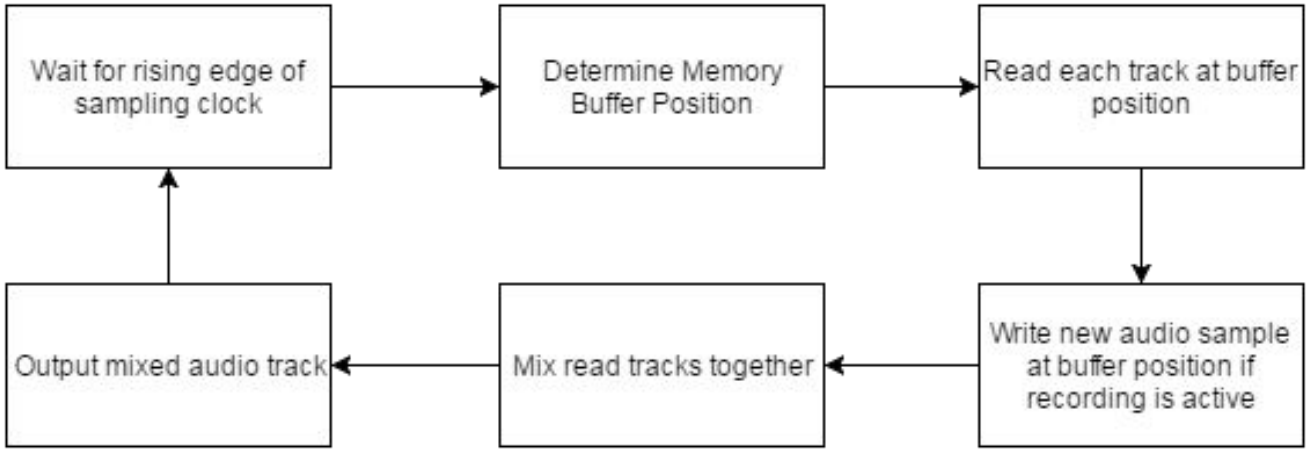
```
Transitions: process (resetn, clk)
        begin
                if resetn = '0' then
y <= S1;
                elsif (clock'event and clock = '1') then
                                case y is
                                        when S1 => if (isvalid = '1' and pressrec = '1') then y <= S2; else y <= S1; end if;
                                        when S2 => if (Vt <= 99999999) then y <= S3; else y <= S2; end if;
                                        when S3 => y <= S1;
                                end case;
                end if;
        end process;

Outputs: process (y, rec)
        begin
        active <= '0'; trackout <= "000"; encount <= '0';
                case y is
                        when S1 => encount <= '0';
                        when S2 =>
                                active <= '1';
                                with rec select
                                trackout <= "111" when "10000000",
                                                "110" when "01000000",
                                                "101" when "00100000",
                                                "100" when "00010000",
                                                "011" when "00001000",
                                                "010" when "00000100",
                                                "001" when "00000010",
                                                "000" when "00000001",
                                                "000" when others;
                                encount <= '1';
                        when S3 => active <= '0';
                end case;
        end process;
end Behavioral;
```

# VHDL: Memory Module



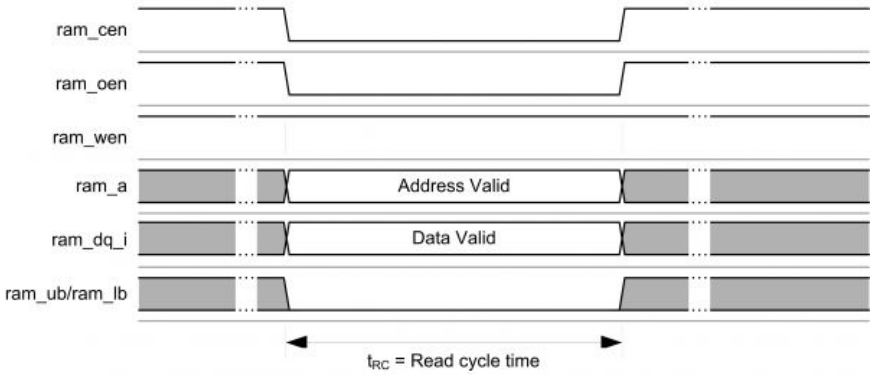
# VHDL: Memory Module

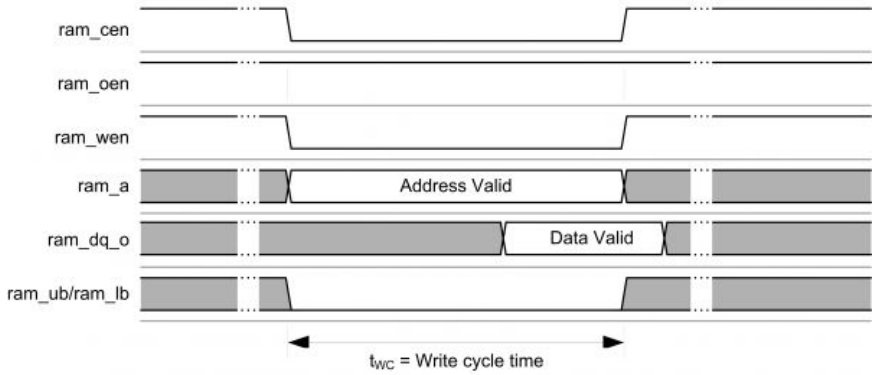


# VHDL: Memory Module: Buffer

- Nexys 4 has 128 MiB DDR2 memory ($2^{24}$, 64-bit addresses, 16-bit accessible per address)
- 8 tracks of 16-bit audio samples, 2,097,152 samples max per track
- 128 kbps sampling rate, 8kHz decimation clock, ~4.3 minutes of audio max per track
- Total memory had to be split up into 8 tracks, 480,000 samples/track was chosen (60 seconds)
- Buffer index incremented on each 8kHz decimation clock then summed with track index multiplied by buffer size
    - Example: Writing to track3, at buffer index 5824, address = 0x160FC0 = 5824 + 480000 x 3

# VHDL: Memory Module: RAM Timing



# VHDL: Audio Input Module

- Convert from PDM (pulse density modulation) to PCM (pulse code modulation)
- PCM = series of stair steps
- PDM = logic high is up, logic low is down (relative description)
- Wanted to use algorithm to convert
- Send the PCM signal to memory
- Send out that PCM signal to audio out and then convert to PDM while sending the signal out to sound generation (headphones or speaker)

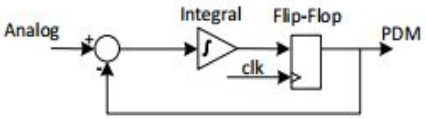



*Figure 26. Simple delta-sigma modulator circuit.*

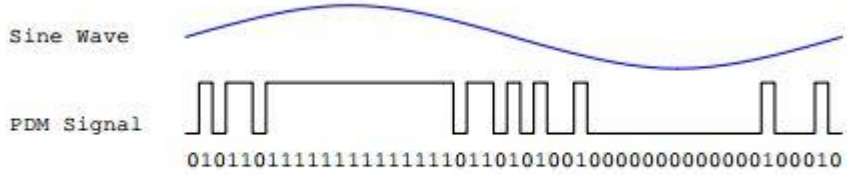



*Figure 25. PDM representation of a sine wave.*

# VHDL: Audio Output Module

- Audio output was the inverse of the audio input
- Wanted to to convert from PDM signal to PCM signal
- Then insert sound generation device to audio output jack (speakers/headphones)

# VHDL: Audio Input Module

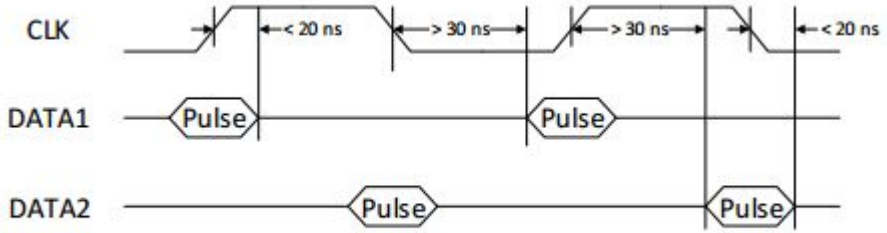
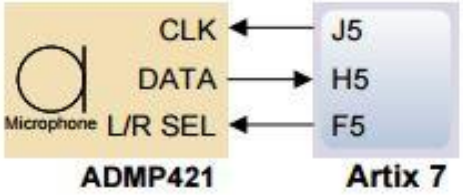


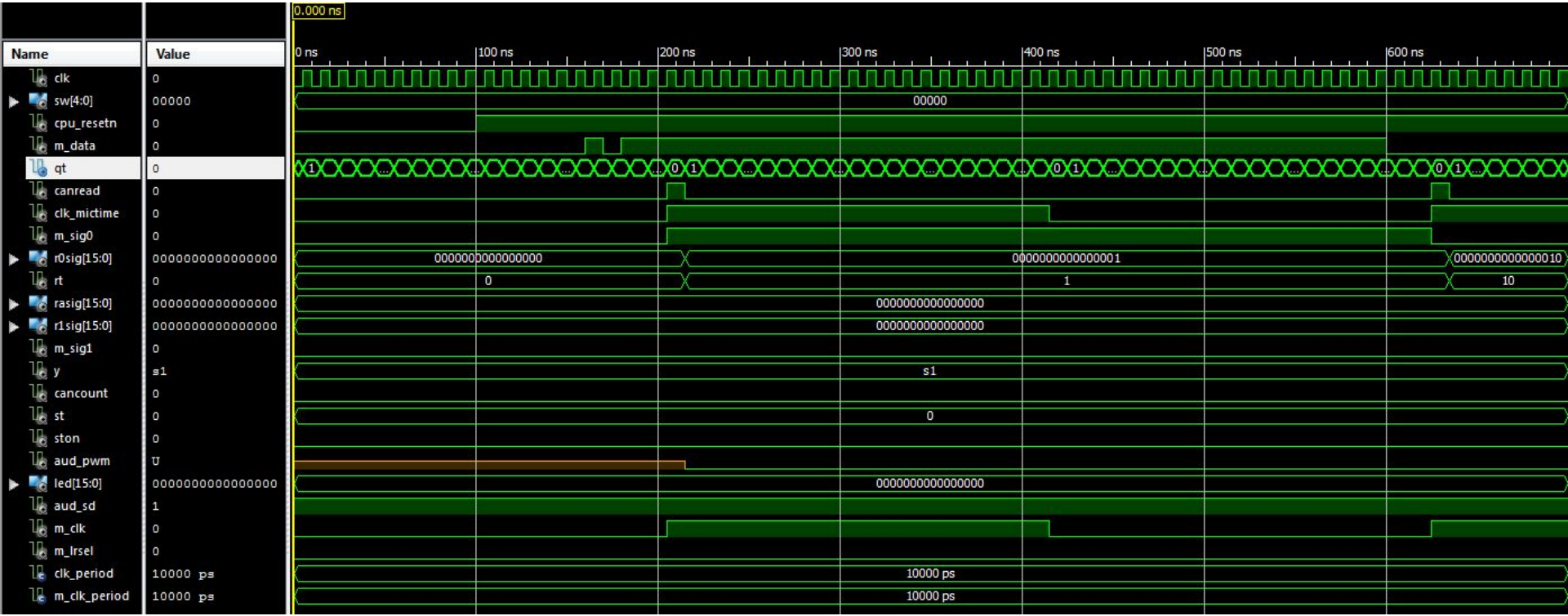

*Figure 27. PDM Timing Diagram.*




**CLK**: Clock Input to Microphone
**DATA**: Data Output Signal
**L/R SEL**: Left/Right Channel Select

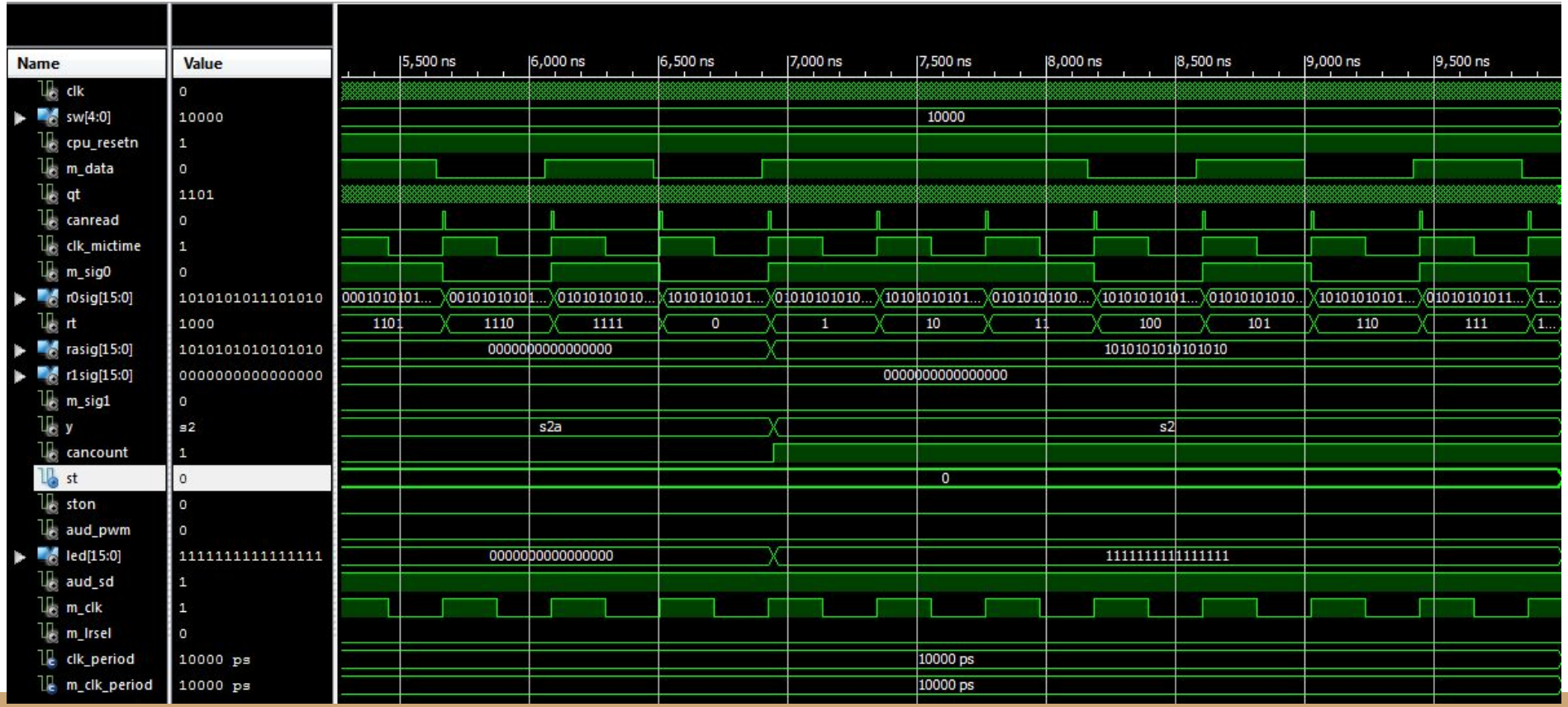
*Figure 24. Microphone block diagram.*

# Timing Diagram 1

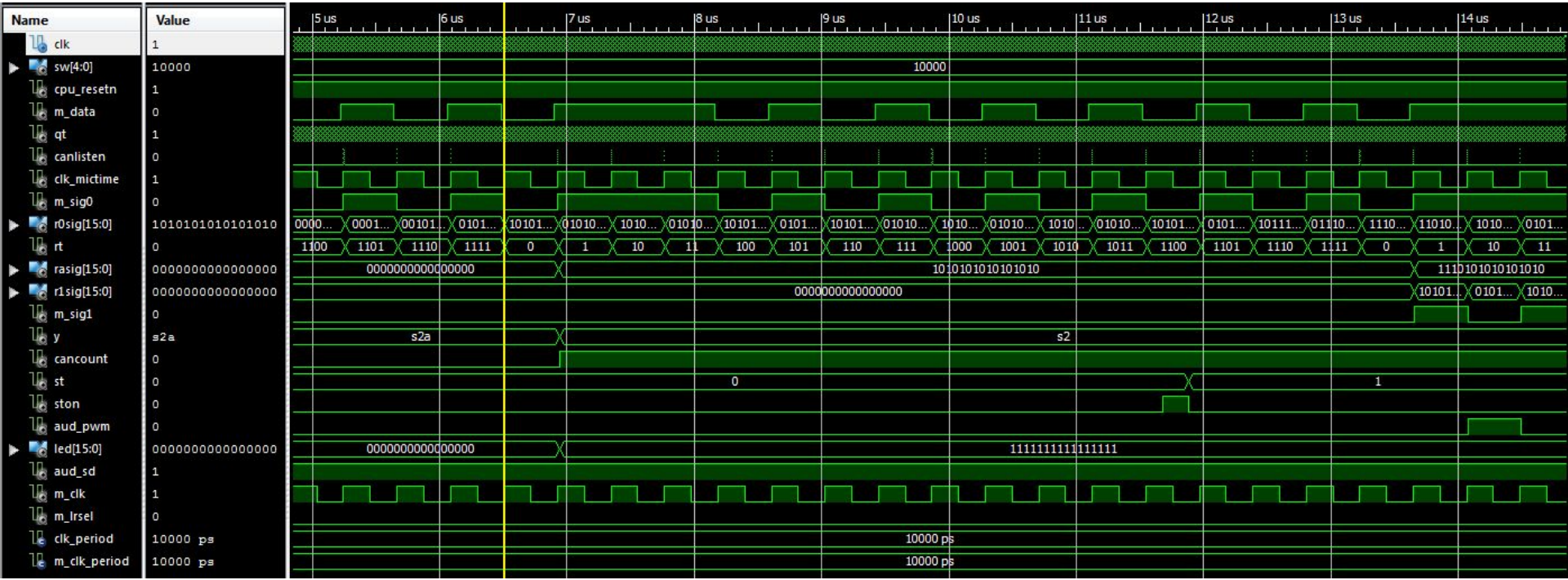


# Timing Diagram 2

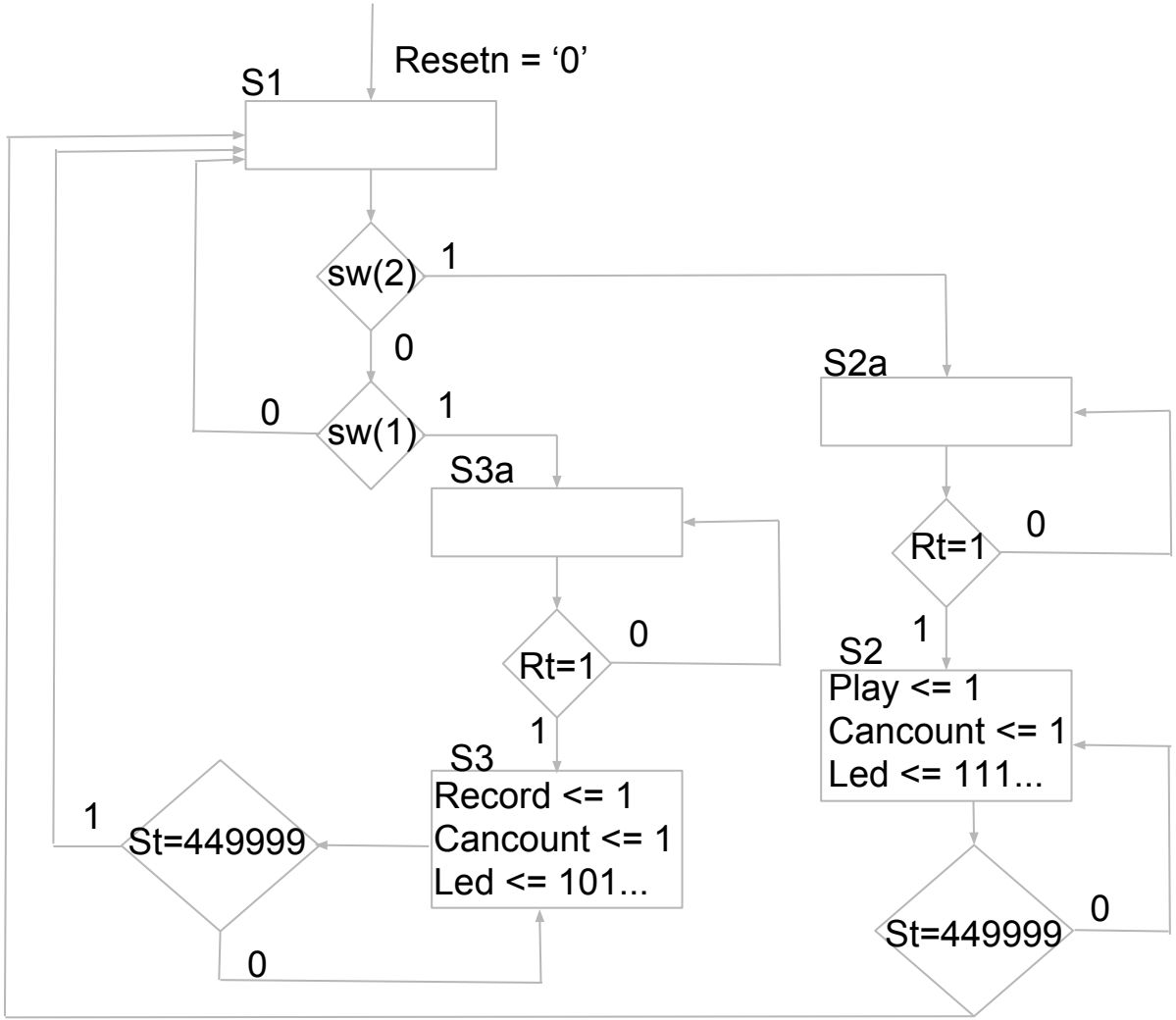
# Timing Diagram 3



# Timing Diagram 4



# ASM


Resetn = '0'
S1
sw(2)
1
0
S2a
0
sw(1)
1
S3a
Rt=1
0
1
S2
Play <= 1
Cancount <= 1
Led <= 111...
S3
Record <= 1
Cancount <= 1
Led <= 101...
1
St=449999
0
St=449999
0

# Problems Faced

- Onboard vs external mic
- Onboard mic implementation
- Audio input/output and memory clock synchronization and resulting noise
- Mixing multiple tracks
- Thought PCM ⇔ PDM was easy
- Generating testbench for VHDL relying on physical HW

# What we would have done with more time

- Reduce noise resulting from high mic sample frequency versus available memory
- Decide on external vs internal mic sooner in design process
- Research how to layer more effectively