Documentation

Team 9

14/06/2025

1 Team members

JIBAN-UL AZAM CHOWDHURY SHAFIN

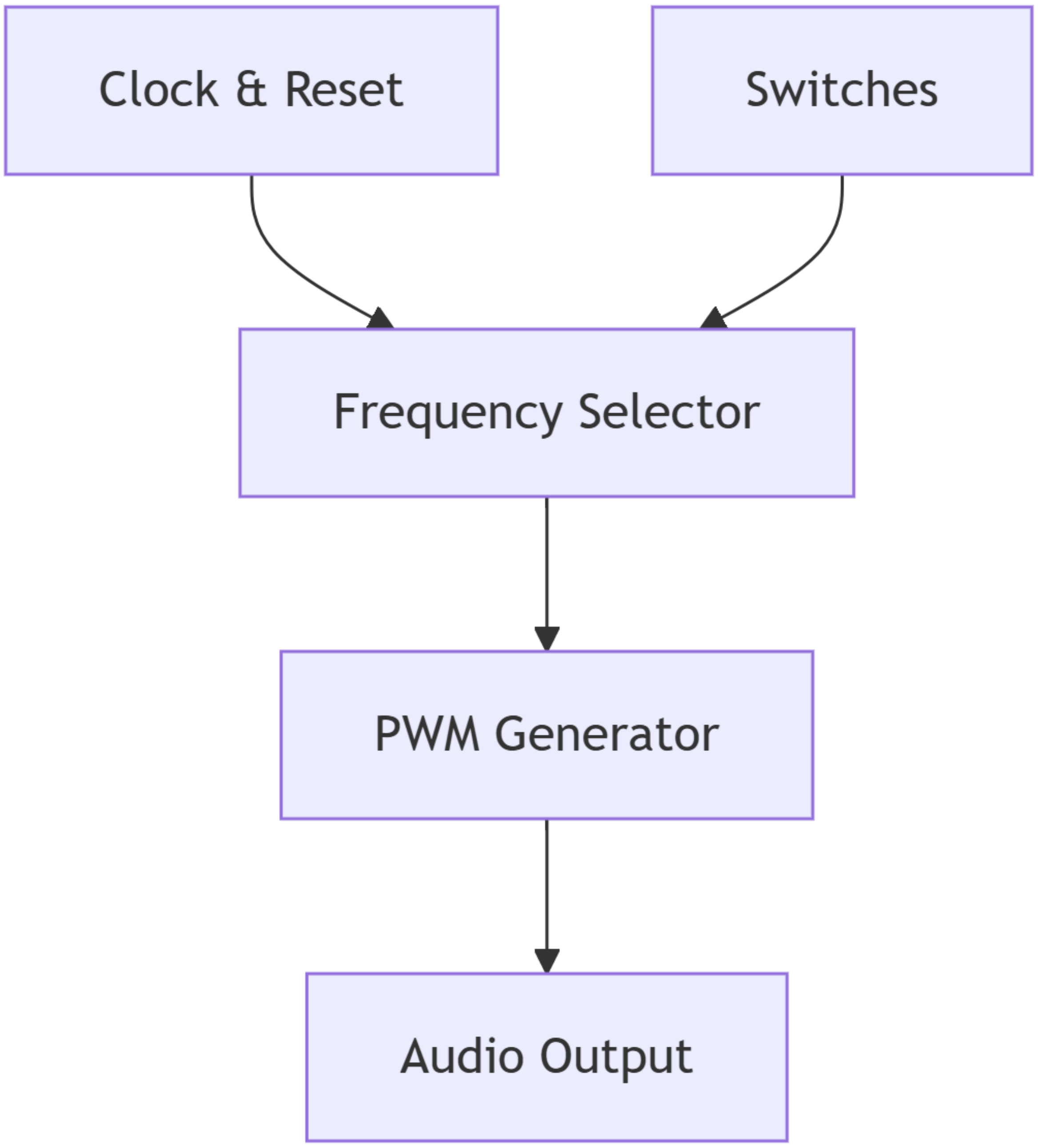
2 Concept description

This project implements a **basic digital audio synthesizer** using an FPGA(Nexys A7 FPGA board with Artix-7). It generates square wave audio tones through **PWM (Pulse Width Modulation)** based on input from **16 slide switches**. Each switch corresponds to a different audio frequency. When a switch is ON, the FPGA calculates a PWM limit to control the output waveform's frequency. The result is output through AUD_PWM (audio signal) and AUD_SD (audio enable).

🔷 **System Requirements**

- Nexys A7 FPGA board with Artix-7
- 16 slide switches (SW[0] to SW[15]) to select tone frequencies
- Audio output through 3.5mm jack (AUD_PWM and AUD_SD)
- 100 MHz input clock
- Reset button to mute output and reset PWM counter

🔷 Block Diagram:



🔷 Used Peripherals

- **Slide Switches** (SW[0] to SW[15]): To select tone
- **Audio Output Port** (AUD_PWM, AUD_SD): Sends the tone to speakers
- **Reset Button**: Resets the system

- **On-board Clock**: 100 MHz system clock

---

3 Project/Team management

Project Methodology Used

In this project, I followed a **Kanban-style task management approach**, inspired by Agile methodology. Being a solo developer, I divided the project into clearly defined tasks and organized them visually on a personal Kanban board. This helped in monitoring progress, prioritizing critical steps, and maintaining focus throughout the development cycle. Each task transitioned through phases like *To Do*, *In Progress*, *Testing*, and *Completed*.

Task Breakdown and Management Strategy

To ensure structured development and avoid confusion during later stages, I planned and executed the project in the following phases:

1. Block Diagram Design

Before diving into VHDL, I conceptualized the system by designing a **functional block diagram** using Microsoft Word. This step clarified how different components like the frequency selector, PWM generator, and audio output would interact.

2. VHDL Code Development

I wrote the synthesizer logic in **VHDL**, implementing key elements:

- Input handling for 16 switches (SW[15:0])
- PWM signal generation based on selected tone frequency
- Reset behavior for clearing internal states
- Continuous activation of the audio enable line (AUD_SD <= '1')

3. Testbench Creation and Functional Simulation

Using **ModelSim**, I created a VHDL testbench to validate the behavior of the code in simulation. This included:

- Applying reset pulses
- Simulating switch inputs one at a time
- Verifying correct frequency selection and PWM generation

4. Synthesis and Real Hardware Testing

Once simulation results were validated, I used **Vivado** to:

- Synthesize the VHDL code
- Apply XDC constraints (pin mapping)
- Generate the bitstream
- Program the design onto the **Nexys A7 FPGA board**

5. Hardware-Level Testing on Nexys A7

Once the simulation was verified, I synthesized and deployed the design to the Nexys A7 FPGA development board. I used the on-board slide switches (SW[0] to SW[15]) to select the tone frequency. The reset functionality was triggered using the BTNU push button (pin M18), mapped to the RESET signal in my VHDL code. I then connected headphones to the on-board 3.5mm audio output jack, where the AUD_PWM signal produced audible tones. This step helped me verify that the correct frequencies were generated in real-time based on switch selection.

6. Schematic and PCB Design

The schematic and PCB design phase was managed using a part of Kanban system, which helped organize each task into stages like *To Do*, *In Progress*, and *Done*. I used KiCad 8.0 to draw the complete schematic, assign correct footprints, and connect all components. After verifying the schematic, I moved to PCB layout, where I placed components and routed all tracks manually instead of using an autorouter plugin. The layout was verified with DRC (Design Rule Check) to ensure there were no errors. Once completed, I generated BOM, Gerber Files, a 3D model and exported a STEP file, which I reviewed using an online 3D viewer to confirm mechanical accuracy and completeness.

7. Version Control

All files—code, constraints, testbenches, diagrams, and documentation—were tracked using **Git**. This ensured:

- Backup and recovery
- Task tracking using commit messages
- Synchronization with a GitHub repository:” https://github.com/JAShafin/SS2025_HWE_Lab_Team9”

Team Roles

This was an **individually developed project**, so I handled all roles myself:

Role	**Responsibility**
Project Manager	Planning the timeline, organizing tasks in Kanban
System Architect	Designing block diagram and system structure

Role	Responsibility
VHDL Developer	Writing synthesizer logic in VHDL
Test Engineer	Creating testbench, running simulations
FPGA Engineer	Synthesizing, debugging on hardware
PCB Designer	Schematic + PCB design in KiCad
Documentation	Writing reports, managing Git version control

4 Technologies

To implement this project, I used the following technologies and tools:

VHDL (VHSIC Hardware Description Language): *Used to describe the hardware behavior of the system. The logic included generating different audio frequencies using PWM based on 16 input switches, handling the reset functionality, and managing audio output signals.*

Vivado: *Vivado was used for writing, simulating, synthesizing, and implementing the VHDL code. It also handled constraint management with XDC files and programming the Nexys A7 FPGA board.*

FPGA (Nexys A7 with Artix-7): *The design was tested and validated using the Nexys A7 FPGA development board. The board provided built-in switches, reset button, and audio output interface which helped in quick testing and demonstration.*

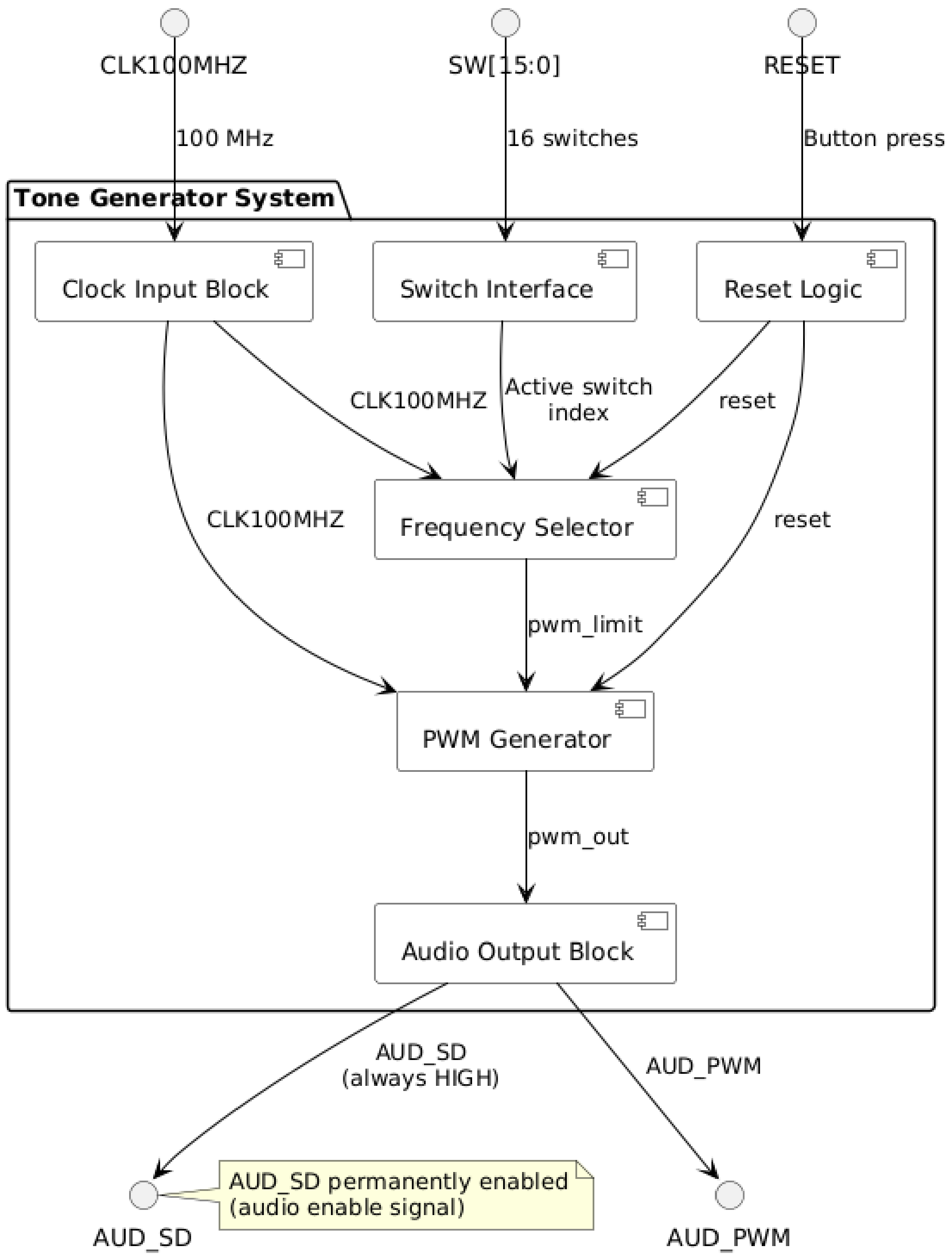
KiCad: *I used KiCad to design a custom PCB for the project. The process included schematic creation, footprint assignment, PCB layout with full manual routing, DRC checks, generating Gerber and drill files, and exporting the 3D model of the board.*

Kanban (Task Planning): *I followed a Kanban-inspired task management approach during the project. Tasks were divided into small actionable steps and tracked manually to maintain progress and ensure timely completion.*

5 VHDL and FPGA Implementation

VHDL Design Overview
The FPGA-based digital design implements a ***software-controlled audio tone generator*** *using Pulse Width Modulation (PWM). It is coded in VHDL and mapped to the* ***Xilinx Artix-7 XC7A100T-CSG324*** *device on the Nexys A7 board.*

Block Diagram



Module Breakdown

- ***Clock Input (CLK100MHZ)****: System clock at 100 MHz.*
- ***Switch Input (SW[15:0])****: Each switch maps to a predefined audio frequency between 400 Hz and 1900 Hz.*
- ***Reset (RESET)****: Synchronous reset initializes the counters and silences the output.*
- ***PWM Core****: Calculates the high/low duty cycle based on selected frequency and outputs to AUD_PWM.*

- ***Audio Output (AUD_PWM, AUD_SD)****: AUD_PWM carries the audio tone, AUD_SD is always enabled.*

Design Highlights

- *Uses one process clocked by CLK100MHZ.*
- *Frequency is selected dynamically depending on which switch is active (first high switch in priority order).*
- *A counter-based PWM logic is used to toggle the output signal with the required duty cycle.*

Testbench and Functional Verification
The testbench (tb_synth_audio_output) was created to validate the PWM generation in simulation.
Test Flow:

- *CLK100MHZ was generated with a 10 ns period.*
- *RESET was asserted for 100 ns.*
- *SW(0) and SW(5) were activated sequentially to simulate tone changes.*
- *The simulation was observed in waveform viewer (Vivado) to confirm correct duty cycle changes on AUD_PWM.*

Results:

- *Correct frequency behavior observed for each switch.*
- *AUD_SD remained constant high, confirming correct static assignment.*
- *No glitches or unintended transitions occurred during switch toggles.*

FPGA Implementation Results
The design was synthesized, implemented, and tested using ***Xilinx Vivado*** *on a Nexys A7 development board.*
XDC Pin Mapping:

- *All 16 switches were assigned individually using the correct package pins.*
- *AUD_PWM and AUD_SD were mapped to available audio header pins.*
- *Clock and reset were mapped as per board documentation.*

Post-Synthesis Summary:

- ***Timing met*** *on all paths.*
- ***No LUT or BRAM overflow****.*
- ***Minimal resource usage*** *due to simple combinational logic.*

Hardware Results

- *The FPGA was programmed with the bitstream.*
- *Pressing any switch played a tone on the speaker (connected via audio output header).*
- *RESET immediately muted the audio, confirming correct reset functionality.*

- *Each switch generated a unique and distinct tone, with frequencies increasing as higher switches were pressed.*

6 PCB Design

Schematic Implementation

The schematic was implemented using KiCad 8.0 and divided into five modular sheets to maintain clarity and organization:

- ***FPGA_Audio_Board.kicad_sch (Main Sheet)** – High-level sheet connecting all subsheets*
- ***POWER.kicad_sch** – Implements power regulation using AMS1117 LDOs to generate 3.3 V, 1.8 V, and 1.0 V from a 5 V input*
- ***FPGA_Core.kicad_sch** – Includes the Artix-7 XC7A100T-CSG324 FPGA symbol and related connections*
- ***Interfaces.kicad_sch** – Contains external connections such as the 16-bit switch header and audio output header*
- ***Utilities.kicad_sch** – Contains clock input circuitry, reset button, and JTAG programming header*

The design includes the bare minimum required components to meet the lab's evaluation board criteria. Notable parts:

- *Artix-7 FPGA (symbol only)*
- *3 AMS1117 regulators (3.3 V, 1.8 V, 1.0 V)*
- *Capacitors for regulator stability (6 × 10 µF)*
- *LED and resistor for 3.3 V power indicator*
- *16-bit input switch header*
- *Audio output header (AUD_PWM, AUD_SD)*
- *Reset button and JTAG header*

Once all symbols and footprints were assigned, an ERC (Electrical Rule Check) was performed and passed successfully. The netlist was then imported into the PCB layout.

PCB Layout

*The PCB was designed in **KiCad's PCB Editor** and manually laid out as a 4-layer board with the following layer configuration:*

- ***F.Cu (Top Layer):** Signal and power traces, also includes GND zone*
- ***In1.Cu (Inner Layer 1):** Dedicated GND plane for low-noise reference*
- ***In2.Cu (Inner Layer 2):** Contains zones for 1.0 V, 1.8 V, and 3.3 V power rails*
- ***B.Cu (Bottom Layer):** Signal routing*

*A custom rectangular board outline was defined with a final dimension of **56.5 mm × 54.0 mm**.*

Zones were carefully defined:

- ***GND Zone** on both F.Cu and In1.Cu*
- ***1.8 V Zone** on In2.Cu*
- ***3.3 V Zone** on F.Cu and In2.Cu*
- ***1.0 V Zone** on F.Cu and In2.Cu*

Track widths were selected based on net class, and ***tented vias*** *were used to reduce EMI and soldering issues.*
A ***FreeRouting plugin*** *was initially tested but did not provide satisfactory results, so* ***100% of routing was performed manually*** *to ensure clean and controlled signal flow.*
After layout:

- ***DRC (Design Rule Check):*** *Passed with zero errors*
- ***No unconnected nets or open connections remained***
- ***Copper zones were poured and filled properly***

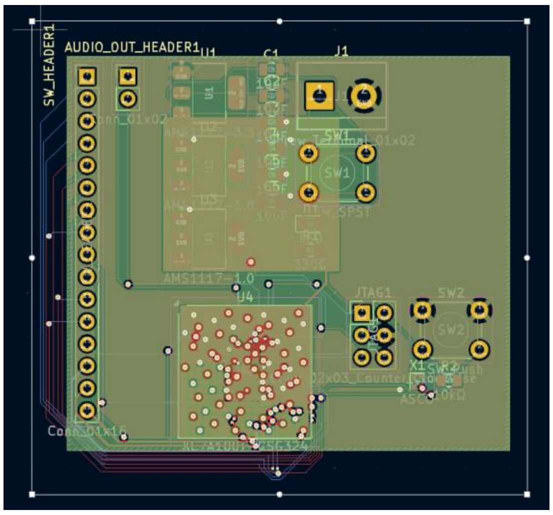


Figure 1:GND zone in In1.cu Layer

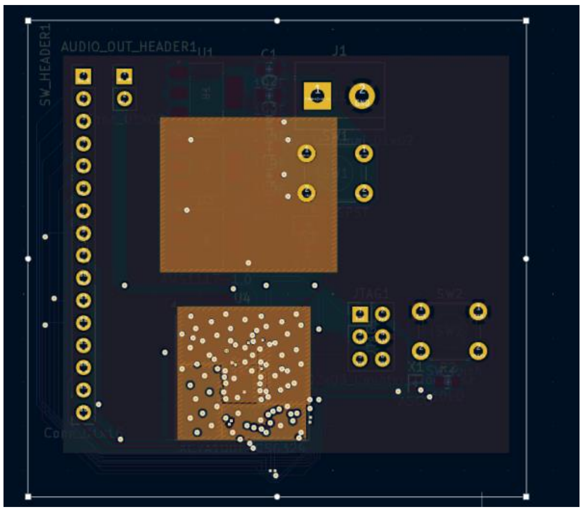


Figure 2: *Zones for 1.0 V, 1.8 V, 3.3 V power in In2.cu Layer*

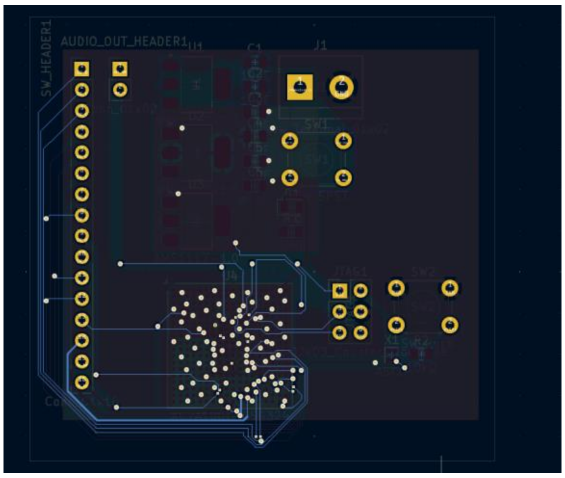


Figure 3: B.Cu (Bottom Layer): Signal routing

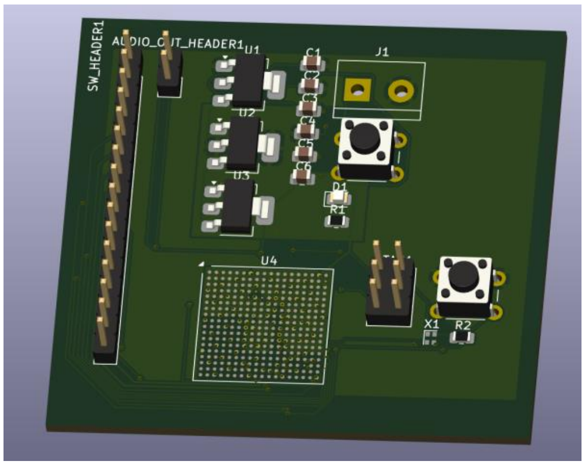


Figure 4: 3D view in KICAD

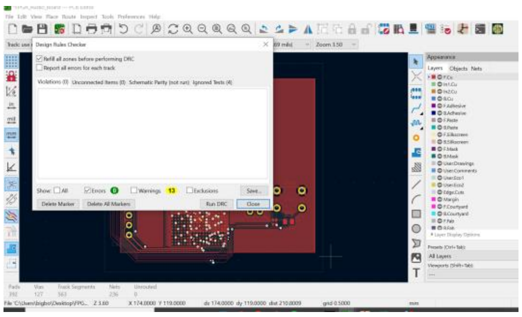


Figure 5: 0 Errors in DRC

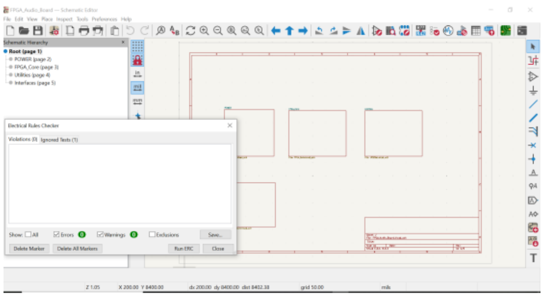


Figure 6: 0 errors in ERC

BOM Summary and Cost Estimation

The BOM was exported from KiCad and included 16 unique components:

- ***6× Capacitors** – 10 µF (0805 package)*
- ***3× AMS1117 LDO regulators***
- ***1× LED (0805)***
- ***2× Resistors***
- ***2× Tactile Switches***
- ***1× JTAG header (2x03)***
- ***1× Terminal block (5V input)***
- ***1× Audio output header***
- ***1× Switch header (16-bit)***
- ***1× Oscillator – ASCO 100 MHz***
- ***FPGA – XC7A100T-CSG324 (symbol only)***

Cost Estimation Summary

Using typical component prices from LCSC, Mouser, and DigiKey (as of June 2025), the estimated unit prices were calculated as follows:

Component Type	***Qty***	***Avg. Unit Price (EUR)***	***Subtotal (EUR)***
AMS1117 LDOs (3.3V, 1.8V, 1.0V)	***3***	***€0.15***	***€0.45***
Capacitors (10 µF, 0805)	***6***	***€0.02***	***€0.12***
LED (0805)	***1***	***€0.03***	***€0.03***
Resistors (1 kΩ, 0805)	***2***	***€0.01***	***€0.02***
Oscillator (100 MHz ASCO)	***1***	***€0.50***	***€0.50***
Headers & Connectors	***4***	***€0.20***	***€0.80***
Push Buttons (Reset, BTNU)	***2***	***€0.10***	***€0.20***
FPGA (Artix-7 XC7A100T-CSG324)	***1***	***€45.00–€55.00***	***€50.00 (avg.)***
Total Component Cost			***~€52.12***

PCB Manufacturing Cost Estimate:

Assuming standard 4-layer PCB manufacturing (FR4, 56.5 mm × 54.0 mm) using services like JLCPCB or PCBWay:

- ***Cost for 5 PCBs: ~€7–€9 total***
- ***Per-board estimate: €1.50***

Estimated Total Cost per Unit (Including PCB and FPGA):
~€53.50 to €54.00 EUR
Note: The FPGA cost is included based on average retail pricing.

3D Model and STEP File
*The board was visualized using **KiCad's 3D Viewer** to check:*

- *Footprint alignment and orientation*
- *Component height and clearances*
- *Board outline and header positions*

*A full 3D model (.step) was exported and verified in **SolidWorks 2024**. While SolidWorks had display issues initially, validation through an online 3D viewer confirmed that the STEP file was correctly generated and included all mechanical components.*

7 Results

The digital audio synthesizer project was successfully implemented and tested on the Nexys A7 FPGA board. All 16 slide switches (SW[0] to SW[15]) correctly triggered different tone frequencies ranging from 400 Hz to 1900 Hz, generated via PWM. The system responded reliably to user inputs, and the reset button immediately muted the output, confirming proper reset functionality.

In simulation (testbench), frequency transitions were verified using waveform analysis. The `AUD_PWM` output showed accurate high/low timing behavior corresponding to selected tones, and `AUD_SD` remained constantly high, confirming proper activation.

The final FPGA synthesis met all timing constraints with no resource overflow. Hardware validation was successful using on-board components (switches, reset, and audio output). A custom PCB design was also completed with clean layout, DRC success, and correct 3D visualization. Overall, the design fulfilled all project goals and provided a stable, real-time tone generation system based on hardware logic.

8 Sources/References

The following resources and tools were used during the implementation of this project:

- *GitHub repository for this project*
  Source: https://github.com/JAShafin/SS2025_HWE_Lab_Team9

- *Xilinx Vivado Design Suite documentation*
  Source: https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0002-vivado-design-hub.html
- *Digilent Nexys A7 Reference Manual*
  Source: https://digilent.com/reference/programmable-logic/nexys-a7/start
- *KiCad 8.0 Official Documentation*
  Source: https://docs.kicad.org/
- *Online 3D STEP file viewer*
  Source: https://cad-viewer.com/