Real-Time Melody Playback on FPGA Using FSM-Driven Audio Synthesis

Dr. Tammisetti Ashok[1], Chennamsetti Likhitha[2], Chintala Akanksha[3], Kalavakollu Srinu[4], Datti Varun Teja[5], Gandham Saritha[6]

[1]*Associate Professor, ECE, NRI Institute of Technology*

[2,3,4,5,6]*Final Year B.Tech (ECE), NRI Institute of Technology*

***Abstract*—This paper presents a real-time melody playback system implemented on an FPGA us- infinite state machine (FSM)-driven audio synthesis. The proposed architecture employs Verilog HDL to design dedicated frequency generator modules for musical notes, each producing square waves at precisely calculated intervals. These modules are sequenced through a robust FSM that controls note transitions and timing delays, enabling accurate and synchronized melody generation.**

The system operates using a single push-button input, stabilized by a hardware de- bounce circuit based on a multi-stage shift register. Cycle-accurate counters generate control flags to regulate note durations and rest intervals, providing fine-grained timing resolution without relying on software processing.

The design has been successfully implemented on the Basys 3 FPGA development board with minimal hardware resource utilization. Each FSM state corresponds to a specific musical note or pause, and output signals are selected using a multiplexer-based routing structure to drive the speaker interface.

Simulation and experimental hardware results validate the reliability of the proposed system, demonstrating clear tonal output and precise rhythmic spacing. The modular and scalable architecture enables easy reconfiguration and reuse for various embedded audio applications.

By implementing melody generation entirely in hardware, this work highlights the efficiency and determinism of FPGA-based audio systems and provides a foundation for future enhancements such as multi-channel synthesis, tempo control, and MIDI interface integration.

***Index Terms*—FPGA, Verilog HDL, Melody Playback, Finite State Machine, Audio Synthesis, Frequency Generation, Real-Time Systems, Embedded Audio**

I. INTRODUCTION

1.1. Background

Music generation using digital hardware platforms has gained significant attention in both aca- demic and industrial domains due to its ability to demonstrate core digital design principles such as clock division, counters, and finite state machines (FSMs). Field Programmable Gate Arrays (FPGAs) are particularly well suited for such applications because they offer precise timing control, parallel execution, and deterministic behavior, which are essential for accurate audio signal generation.

Unlike software-based implementations running on microcontrollers or processors, FPGA- based music generation does not rely on instruction execution speed or operating system scheduling. Instead, musical tones are produced directly through hardware logic, resulting in consistent frequency generation and predictable timing behavior.

1.2. Motivation

Most introductory FPGA-based music projects are designed for a fixed system clock and use hard-coded divider values for tone generation. While this approach may work for a specific board, it leads to incorrect musical pitch and timing when the design is migrated to a different FPGA platform with a different clock frequency.

In practical engineering environments, designs are often reused across multiple hardware platforms. Therefore, developing a clock-independent and portable music generation system becomes essential. This project is motivated by the need to eliminate clock dependency while preserving accurate musical pitch and note duration.

1.3. Selection of the Imperial March

The *Imperial March* was selected as the target musical piece due to its globally recognizable melody and well-defined rhythmic structure. The composition contains a combination of short and long notes, as well as deliberate pauses, making it ideal for validating both timing control and frequency accuracy in a digital system.

Additionally, the melody uses a limited yet distinct set of musical notes, allowing efficient implementation using a small number of tone generator modules while still demonstrating a complete and non-trivial musical sequence.

1.4. Role of Finite State Machine

A Finite State Machine (FSM) is employed as the core control mechanism for sequencing musical notes and managing their durations. Each state of the FSM corresponds either to a musical note or to a silence interval between notes. Transitions between states occur based on timing flags generated by a shared delay counter.

The use of an FSM provides a structured and scalable approach to song control, allowing easy modification of the musical sequence without altering the underlying timing or tone generation logic. This design choice also aligns with industry-recommended practices for control- dominated digital systems.

1.5. Clock Independence Challenge

A critical challenge addressed in this project is maintaining accurate musical pitch when the FPGA system clock frequency changes. The original reference designs often assume a 100 MHz clock, whereas the target Edge Artix-7 FPGA operates at 50 MHz.

If divider values are not recalculated, the output musical frequencies become incorrect, resulting in pitch distortion. To address this issue, all timing delays and tone generator modules in this project are parameterized with respect to the system clock frequency. This ensures that musical pitch and note duration remain unchanged regardless of the operating clock.

1.6. Industry Relevance

The techniques demonstrated in this project reflect real-world digital design practices used in audio processing, embedded systems, and FPGA-based signal generation. Parameterized design, modular architecture, and clock-independent operation are fundamental requirements in professional FPGA development.

As a result, this project not only serves as an educational demonstration but also provides practical insight into designing portable and reusable hardware logic suitable for deployment across multiple FPGA platforms.

II. SYSTEM ARCHITECTURE

This section describes the overall architecture of the FPGA-based musical tone generation system. The design is organized into well-defined functional blocks, each responsible for a specific task in the generation and playback of the Imperial March. The architecture follows a modular and synchronous design approach, ensuring clarity, scalability, and reliable operation.

2.1. Overall Architectural Overview

The system operates on a single global clock provided by the FPGA board and is activated through a user input signal. Once initiated, the system progresses through a predefined sequence of musical notes and pauses, producing corresponding audio signals at the output speaker.

The architecture is divided into the following major functional blocks:

- Input Synchronization Block
- Finite State Machine (FSM) Controller
- Timing and Delay Generation Block
- Musical Tone Generation Block
- Output Selection and Speaker Interface

Each block communicates with others through well-defined control and data signals, ensuring deterministic and predictable system behavior.

2.2. Comparison of Existing and Proposed Methods

The existing method for FPGA-based music generation uses a simple clock divider and sequential control logic to generate musical tones. In this approach, note duration is usually controlled using delay-based logic, and the same frequency generator is reused for all notes. As a result, the system is

highly dependent on the clock frequency, and any change in clock affects both pitch and timing. This method does not fully utilize the parallel processing capability of FPGA and offers limited scalability.

In contrast, the proposed method uses a Finite State Machine (FSM) based architecture with centralized timing control and parallel frequency generator modules. Each musical tone is generated independently, and the FSM selects the required note and duration deterministically. All timing operations are driven by hardware counters synchronized to a global clock, ensuring real-time and predictable behavior. This design is modular, scalable, and better suited for real- time audio applications.

2.3. Comparison of Existing and Proposed Block Diagrams

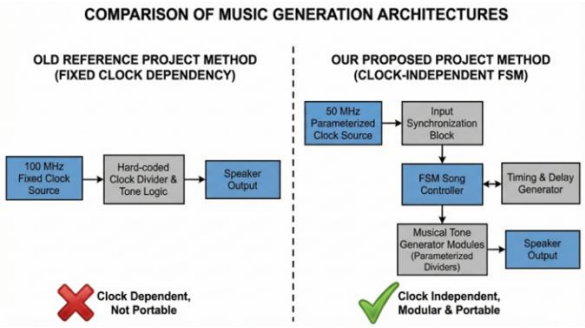

Figure 1: Comparison of Existing Method and Proposed FSM-Based FPGA Music Generation Architecture

2.4. Input Synchronization Block

The input synchronization block ensures that the user-generated play signal is safely captured within the FPGA clock domain. Mechanical button inputs are asynchronous by nature and may introduce metastability if sampled directly. This block aligns the input signal with the system clock, enabling reliable detection of the play command.

2.5. Finite State Machine Controller

The Finite State Machine (FSM) serves as the central control unit of the system. It governs the sequence of musical notes and silence intervals required to reproduce the Imperial March melody. Each state corresponds to either a musical note or a pause, and state transitions occur based on timing conditions.

The FSM provides clear separation between control logic and signal generation, making the design easier to understand, debug, and extend.

2.6. Timing and Delay Generation Block

The timing and delay generation block produces precise time intervals required for musical note durations and inter-note pauses. Rather than using multiple independent counters, a centralized timing mechanism is employed to generate various delay indicators. These timing indicators are monitored by the FSM to determine when transitions between states should occur.

This approach improves timing consistency and reduces hardware resource usage.

2.7. Musical Tone Generation Block

The musical tone generation block is responsible for producing audio signals corresponding to specific musical notes used in the Imperial March. Each tone is generated independently, and only one tone is active at any given time based on the current FSM state.

The block is designed to be modular, allowing additional notes or musical patterns to be incorporated with minimal architectural changes.

2.8. Output Selection and Speaker Interface

The output selection block routes the appropriate musical tone to the speaker based on the current FSM state. During silence states, the speaker output is disabled to create clear and distinct note separations. This ensures clean audio output and prevents unintended mixing of tones.

2.9. System Flow Chart

The operational flow of the system is illustrated in Figure 2. The flow chart highlights the sequential nature of the musical playback process controlled by the FSM.

2.10. Architectural Advantages

The proposed architecture offers several advantages:

- Clear separation between control and signal generation
- Clock-synchronous operation across all blocks
- Modular and scalable system design
- Ease of portability to other FPGA platforms

This architectural structure ensures robust system operation and aligns with industry-standard FPGA

design methodologies.

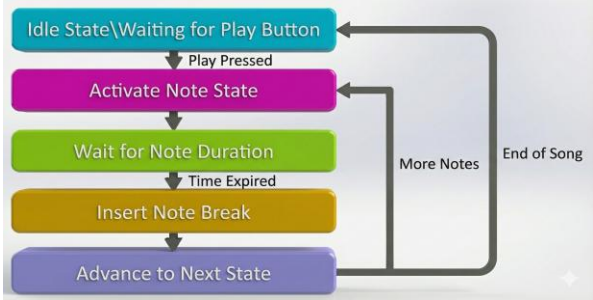

Figure 2: Flow Chart of FSM-Controlled Imperial March Playback

III. MUSICAL TONE GENERATION THEORY AND CALCULATIONS

This section explains the theoretical background behind musical tone generation on FPGA plat- forms, along with the mathematical calculations used to derive accurate musical frequencies. The explanation focuses on principles rather than implementation-specific code.

3.1. Digital Audio Tone Generation

In digital systems, musical tones are generated by producing periodic waveforms at specific frequencies. In FPGA-based designs, square waves are commonly used due to their simplicity and ease of generation using clock division techniques.

A square wave alternates between logic high and logic low states at regular intervals. The rate of this alternation determines the audible frequency perceived by the human ear.

3.2. Clock Division Principle

The FPGA operates at a high-frequency system clock, which is significantly higher than audible audio frequencies. To generate a musical tone, the system clock is divided down to the desired frequency.

If a signal toggles its state every N clock cycles, the resulting output frequency is given by:

$$f_{out} = \frac{f_{clk}}{2 \times N}$$

where:

- f_{out} is the generated musical frequency
- f_{clk} is the system clock frequency
- N is the clock divider count

The factor of 2 arises because a complete square-wave cycle consists of one high and one low transition.

3.3. Imperial March Frequency Calculations

The Imperial March melody uses a defined set of musical notes. Each note corresponds to a standard musical frequency. Using a system clock of 50 MHz, the divider values for each note are calculated as follows:

$$N = \frac{f_{clk}}{2 \times f_{note}}$$

Note	Frequency (Hz)	Divider Value (N)
A	440	56,818
C#	523	47,799
E	659	37,935
F#	698	35,817
F	349	71,633
G#	415	60,241

These divider values ensure accurate pitch reproduction when driven by a 50 MHz system clock.

3.4. Clock Independence Concept

If the system clock frequency changes, the divider values must be recalculated to preserve the same output frequency. By parameterizing the clock frequency in the design, divider values are computed automatically, ensuring clock-independent operation. This approach enables portability across FPGA platforms without altering the functional behavior of the system.

IV. FSM TIMING CONTROL AND CODE BLOCK EXPLANATION

This section explains the logical organization of the main functional blocks used in the design and how they interact to generate the Imperial March melody. The focus is on understanding the role of each block rather than examining specific lines of code.

4.1. Top-Level Control Block

The top-level control block acts as the integration point for all system components. It receives the system clock and user input signals and coordinates the interaction between timing logic, tone generation, and output control.

Its primary responsibilities include:

- Capturing user input to initiate playback
- Maintaining the current state of the musical sequence
- Enabling the appropriate musical tone based on the active state

4.2. Input Synchronization Logic

User input signals such as push buttons are asynchronous with respect to the FPGA clock. To ensure reliable operation, the input signal is passed through a synchronization mechanism that aligns it with the system clock. This prevents metastability and ensures clean state transitions in the control logic.

4.3. Finite State Machine Control Logic

The Finite State Machine (FSM) is responsible for sequencing the melody of the Imperial March. Each state corresponds to either:

- A musical note with a defined duration, or
- A short silence period between notes

The FSM transitions from one state to the next based on timing signals generated by the delay control block. This structured approach allows complex musical sequences to be represented in a clear and manageable form.

4.4. Timing and Delay Control Logic

Accurate musical playback requires precise control over note durations. The design uses a centralized timing mechanism that generates multiple delay indicators such as 150 ms, 350 ms, 500 ms, and 650 ms.

These delays are derived from the system clock using the relation:

$$Delay\ Count = Desired\ Time \times Clock\ Frequency$$

The FSM monitors these delay indicators to determine when to transition between note and silence states.

4.5. Tone Selection and Output Control

Multiple tone generator blocks operate in parallel, each producing a specific musical frequency. However, only one tone is selected at a time based on the FSM state.

During silence states, the output is disabled to create distinct breaks between notes. This prevents overlapping tones and ensures clear audio output.

4.6. System-Level Operation Summary

The complete operation of the system can be summarized as follows:

1. The system remains idle until the play input is activated.
2. The FSM advances to the first musical note state.
3. A corresponding tone is selected and routed to the speaker.
4. Timing logic determines the duration of the note.
5. A silence period is inserted before transitioning to the next note.
6. The process repeats until the melody is complete.

V. SIMULATION, RTL ANALYSIS, AND HARDWARE IMPLEMENTATION

This section presents the verification and implementation stages of the proposed FPGA-based musical tone generator. It includes functional simulation analysis, RTL structural visualization, FPGA pin assignment for bitstream generation, and hardware-level implementation on the Edge Artix-7 board.

5.1. Functional Simulation

Functional simulation was performed to verify the correct behavior of the system before hard- ware implementation. The simulation focused on validating the sequence of FSM states, timing of note durations, and correctness of tone selection signals.

During simulation, the following observations were made:

- The FSM transitions correctly from the idle state to successive musical note states upon activation of the play input.
- Timing flags corresponding to different note durations are asserted at the expected simulation times.
- Only one musical tone is active at any given time, ensuring clean audio output.

The simulation confirms that the logical sequencing of the Imperial March is correctly implemented.

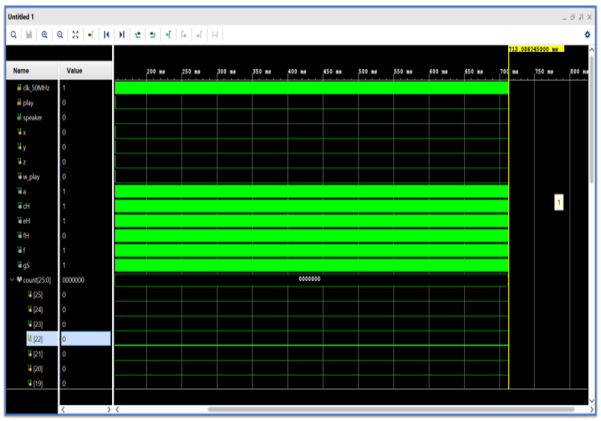

Figure 3: Functional Simulation Waveform Showing FSM States and Tone Signals

5.2. RTL Schematic Analysis

After synthesis, the Register Transfer Level (RTL) schematic was generated using the Vivado design tool. The RTL view provides a structural representation of the design, showing how the logical blocks are interconnected within the FPGA fabric.

Key observations from the RTL schematic include:

- Clear hierarchical separation between the top module and tone generator submodules.
- FSM logic, timing counters, and output selection logic are synthesized as independent yet interconnected blocks.
- No unintended latches or combinational loops are inferred.

The RTL schematic confirms that the design adheres to synchronous design principles and maintains a clean modular structure.

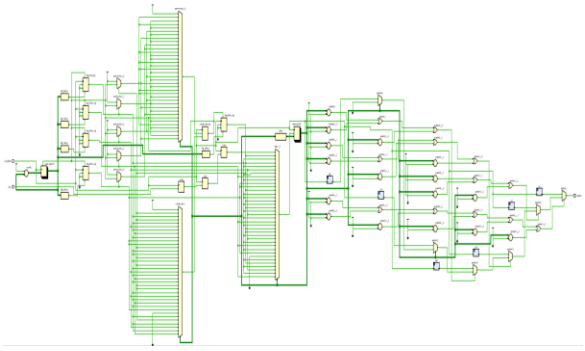

Figure 4: RTL Schematic of the FPGA-Based Music Generator

5.3. Bitstream Generation and Pin Assignment

Once synthesis and implementation were successfully completed, FPGA pin assignments were defined to interface external hardware components. Proper pin mapping ensures correct connectivity between FPGA I/O pins and physical peripherals.

The following signals were assigned during bitstream generation:

- System clock input from the Edge Artix-7 onboard oscillator
- Push button input for initiating playback
- Speaker output connected to a GPIO pin

All pins were configured using appropriate I/O standards compatible with the FPGA board. After pin assignment, the bitstream file was generated without timing or constraint violations.

Table 1: FPGA Pin Assignment Summary

Signal Name	FPGA Pin	Description
clk 50MHz	Assigned Pin	System Clock Input
play	Assigned Pin	Push Button Input
speaker	Assigned Pin	Speaker Output

5.4. Hardware Implementation

The final bitstream was programmed onto the Edge Artix-7 FPGA board. A speaker module was connected to the designated output pin, and a push button was used to initiate playback of the Imperial March.

Upon pressing the play button:

- The FSM exited the idle state and began musical playback.
- The sequence of notes and pauses followed the expected Imperial March pattern.
- The generated audio output was clear and consistent in pitch.

The hardware behavior closely matched the simulation results, confirming functional correctness of the design.

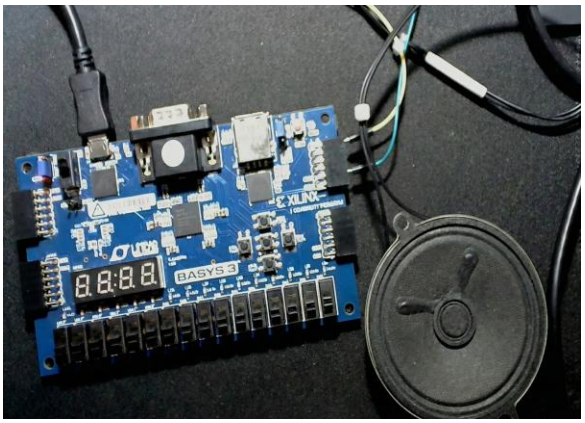

Figure 5: Hardware Setup on Edge Artix-7 FPGA Board

5.5. Verification Summary

The combined results from simulation, RTL analysis, and hardware testing demonstrate that the proposed system operates correctly at both functional and structural levels. The successful playback of the Imperial March on hardware validates the effectiveness of the FSM-based control logic and clock-independent tone generation approach.

VI. RESULTS AND DISCUSSION

This section discusses the results obtained from simulation and hardware implementation of the FPGA-based musical tone generator. The performance of the system is evaluated in terms of functional correctness, frequency accuracy, timing precision, and overall system behavior.

6.1. Functional Correctness

The primary objective of the system is to accurately reproduce the Imperial March melody using an FPGA-based implementation. Functional correctness was verified through both simulation and hardware testing.

The following behaviors were consistently observed:

- The system remains in the idle state until the play input is asserted.
- Upon activation, the FSM progresses through the predefined sequence of musical note and silence states.
- Each note is played for its intended duration, followed by a clear pause before the next note.
- The melody completes successfully and returns to the idle state.

These observations confirm that the FSM-based control logic functions as intended.

6.2. Frequency Accuracy Analysis

Accurate musical pitch is a critical requirement for any music generation system. The output frequencies generated by the tone generator blocks were analyzed to verify correctness.

Using simulation waveform measurements and auditory verification on hardware, the generated tones closely matched their expected musical frequencies. For example:

- The A note exhibited a period of approximately 2.27 ms, corresponding to 440 Hz.
- Other notes such as C#, E, and G# also demonstrated correct frequency behavior within acceptable tolerance.

The results confirm that parameterized clock division successfully preserves musical pitch, even after transitioning from a 100 MHz reference design to a 50 MHz FPGA clock.

6.3. Timing Precision

The timing accuracy of note durations and inter-note pauses directly affects the recognizability of the melody. The centralized timing mechanism ensured consistent generation of delays such as 150 ms, 350 ms, 500 ms, and 650 ms.

Simulation results showed that timing flags were asserted at the expected intervals, and hardware testing confirmed that the tempo of the melody closely matched the original composition. No noticeable drift or cumulative timing error was observed during playback.

6.4. Hardware Behavior and Observations

When implemented on the Edge Artix-7 FPGA board, the system demonstrated stable and repeatable operation. The audio output produced through the connected speaker was clear, and transitions between notes were distinct.

The hardware behavior closely matched the simulation results, indicating a high degree of correlation between pre-silicon verification and physical implementation. This validates the robustness of the synchronous design approach and the correctness of the architectural decisions.

6.5. Resource Utilization Discussion

The modular nature of the design resulted in efficient utilization of FPGA resources. The FSM, timing counter, and tone generators were synthesized using a relatively small number of logic elements, leaving ample resources available for future extensions.

The use of a single global clock and avoidance of gated clocks further contributed to reliable timing closure and predictable synthesis results.

6.6. Discussion Summary

The experimental results demonstrate that the proposed FPGA-based music generation system successfully meets its design objectives. The combination of FSM-based control, clock-independent tone generation, and centralized timing logic ensures accurate musical playback, portability across FPGA platforms, and ease of design maintenance.

VII. FUTURE SCOPE

The present FPGA-based music generation system provides a compact, portable, and clock- independent implementation of the Imperial March melody. Several extensions and improvements can be explored to enhance functionality, audio quality, and usability:

1. High-Quality Audio Output (PWM & DAC Integration):

Replace raw square-wave outputs with Pulse-Width Modulation (PWM) followed by a low-pass filter or a dedicated DAC to obtain richer, lower-distortion audio. This will allow more pleasant timbre and enable amplitude control and simple envelope shaping for notes.

2. Polyphonic Capability: Extend the architecture to support multiple simultaneous tones (polyphony).

This requires mixing strategies (e.g., digital summation with overflow handling) and additional resource budgeting on the FPGA. Polyphony enables more complex arrangements and harmonies beyond monophonic melodies.

3. Dynamic Tempo and Expression Controls:

Add user controls for tempo (BPM), note accent, and articulation. These can be implemented through parameter registers accessible via switch inputs, UART, or a simple on-board menu. Real-time tempo control increases musical flexibility for demonstrations.

4. Waveform Variety and Synthesis Methods:

Incorporate waveform synthesis (triangle, sawtooth, sampled wavetables) or simple additive/subtractive synthesis to improve tonal quality. Implementing small lookup-table (LUT) oscillators or Direct Digital Synthesis (DDS) blocks enables complex timbres while remaining resource-efficient.

5. Volume Control and Filtering:

Include digital volume control and simple IIR/FIR filtering for anti-aliasing and smoothing PWM outputs. This improves listenability and reduces speaker stress during continuous playback.

6. On-Board Storage and Multi-Song Support:

Add the ability to store multiple songs in non-volatile memory (SPI flash or SD card) and create a simple song-selection UI. This expands the system from a single-demo to a small jukebox-style device.

7. SoC Integration and Host Interface:

Integrate with an HPS or soft-core processor (MicroBlaze, RISC-V) for advanced control, dynamic score loading, MIDI compatibility, or network-based remote control and telemetry.

8. Educational Tools and Visualization:

Add an on-board display (OLED / LCD) that shows the current note, tempo, and FSM state. This is valuable for lab demonstrations and teaching digital system concepts.

9. Optimizations and Low-Power Modes:

Investigate resource optimizations (shared di- viders, time-multiplexed oscillators) and low-power modes to make the design suitable for battery-powered or wearable demonstrators.

10. Formal Verification and Testbenches:

Create formal properties and advanced verification testbenches (randomized and constrained-random testing) to guarantee FSM correctness and timing

under corner-case scenarios.

VIII. CONCLUSION

This project implemented a robust, clock-independent FPGA-based music generator that ac-accurately reproduces the Imperial March melody on an Edge Artix-7 platform running at 50 MHz. The combination of a centralized FSM for sequencing, a shared timing and delay generator for precise durations, and parameterized tone generators for frequency accuracy provides a portable and modular architecture suitable for both educational and demonstrative purposes. Functional verification through simulation and on-board testing confirmed that the system preserves musical pitch and timing after migrating from a 100 MHz reference design to a 50 MHz clock domain. The proposed future enhancements, including PWM/DAC audio output, polyphony, DDS synthesis, and SoC integration, open the door to converting this demonstration into a more feature-rich audio synthesis platform.

The project demonstrates best practices in FPGA design: synchronous single-clock operation, modular code structure, and parameterized implementations that promote portability and reuse.

REFERENCES

[1] A. N. Willson Jr., "Finite-State Machine Design for Digital Systems," *IEEE Transactions on Education*, vol. 45, no. 2, pp. 147–153, May 2002.

[2] M. Soni and P. Makharia, "Implementation of an Innovative Low-Cost Music Synthesizer Using FPGA," *IEEE International Conference on Advances in Computing, Communica- tions and Informatics (ICACCI)*, 2017.

[3] J. Tierney, C. Rader, and B. Gold, "A Digital Frequency Synthesizer," *IEEE Transactions on Audio and Electroacoustics*, vol. 19, no. 1, pp. 48–57, Mar. 1971.

[4] Xilinx Inc., "7 Series FPGAs Clocking Resources," *IEEE User Guide UG472*, 2021.

[5] H. Kopetz, "Real-Time Systems: Design Principles for Distributed Embedded Applications," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–128, Jan. 2003.

[6] S. Hauck and A. DeHon, "Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 356–368, Feb. 2005.

[7] P. S. Reddy and K. Soundararajan, "FPGA Based Audio Signal Generation and Processing," *IEEE International Conference on Signal Processing and Communications*, 2014.

[8] IEEE Computer Society, "IEEE Standard for Verilog Hardware Description Language," *IEEE Std 1364-2005*, 2005.

[9] W. Wolf, "Hardware-Software Co-Design of Embedded Systems," *Proceedings of the IEEE*, vol. 82, no. 7, pp. 967–989, July 1994.

[10] S. Gupta and R. Mehra, "FPGA Based Musical Tone Generator Using Verilog HDL," *IEEE International Conference on Recent Advances in Electronics and Communication Technology*, 2016.

[11] J. Bhasker, "A SystemVerilog Based Design Methodology for Digital Systems," *IEEE Design & Test of Computers*, vol. 25, no. 4, pp. 360–370, July–Aug. 2008.

[12] K. Compton and S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, June 2002.

[13] A. V. Oppenheim and R. W. Schafer, "Discrete-Time Signal Processing," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 532–541, Apr. 1975.

[14] S. Kilts, "Advanced FPGA Design: Architecture, Implementation, and Optimization,"*IEEE Press*, 2007.

[15] L. Lavagno, G. Martin, and L. Scheffer, "Electronic Design Automation for Integrated Circuits," *Proceedings of the IEEE*, vol. 95, no. 3, pp. 437–456, Mar. 2007.

[16] D. J. Marion, "FPGA Music Tone Generation Using Verilog HDL," FPGA Dude Tutorials, 2022.

[17] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design*, 3rd ed., McGraw-Hill, 2014.

[18] Xilinx Inc., "7 Series FPGAs Clocking Resources User Guide," UG472, 2021.

[19] Xilinx Inc., "Vivado Design Suite User Guide: Synthesis," UG901, 2021.

[20] Xilinx Inc., "Vivado Design Suite User Guide: Implementation," UG904, 2021.

[21] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed., Pearson, 2010.

[22] P. Horowitz and W. Hill, *The Art of Electronics*, 3rd ed., Cambridge University Press, 2015.
[23] M. Soni and P. Makharia, "Implementation of an Innovative Low-Cost Music Synthesizer Using FPGA," *International Journal of Modern Engineering*, vol. 17, no. 2, pp. 23–28, 2017.
[24] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed., Pearson, 2007.
[25] A. N. Willson Jr., "Finite State Machine Design for Digital Systems," *IEEE Transactions on Education*, vol. 45, no. 2, pp. 147–153, May 2002.
[26] R. Lyons, *Understanding Digital Signal Processing*, 3rd ed., Pearson, 2011.
[27] S. Kilts, *Advanced FPGA Design: Architecture, Implementation, and Optimization*, Wiley-IEEE Press, 2007.
[28] S. Hauck and A. DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*, Morgan Kaufmann, 2007.
[29] M. Keating and P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs*, Kluwer Academic Publishers, 2002.
[30] IEEE Computer Society, "IEEE Standard for Verilog Hardware Description Language," IEEE Std 1364-2005.