FPGA Implementation of a Timer with Alarm

Privi B. Della, Clyde E. Aparente, Robert Alcantara
CEIT Department, Surigao Del Norte State University
Narciso St., Surigao City, 8400 Philippines

***Abstract*— This project presents the design and implementation of a digital timer with alarm functionality on an FPGA platform, specifically targeting the Cyclone II FPGA. The system provides real-time hour, minute, and second tracking, displayed through a multiplexed 7-segment LED interface. Leveraging the speed and parallelism of FPGA architecture, the timer ensures accurate time counting and stable operation via hardware-based logic. Core features include SS:MM format counting, time setting through push buttons, and an integrated alarm feature that activates when a preset time is reached. The system is programmed in VHDL and developed within the Quartus Prime environment, ensuring correct functionality through simulation and synthesis. With a modular and scalable design, this timer-alarm system is ideal for educational, embedded, and real-time digital applications.**

I. INTRODUCTION

In the evolving domain of embedded systems and digital design, timers with alarms serve as foundational elements in many real-time applications. Traditional microcontroller-based designs are often limited by sequential execution and software dependencies. This project explores a hardware-centric solution using an **FPGA-based timer with an alarm**, offering increased precision and parallel processing capabilities. By implementing the timer on a **Cyclone II FPGA**, the system benefits from high-speed logic operations, enabling real-time counting and immediate alarm response without software-induced delays.

The timer tracks and displays minutes and seconds on six 7-segment LED displays through a time-multiplexed scheme. Push buttons allow users to manually adjust the timer duration and set the alarm time. Once the timer reaches the alarm setting, a designated output (e.g., LED or buzzer) is triggered. All logic is implemented in VHDL and simulated using Quartus Prime Lite Edition, ensuring accurate 1 Hz pulse generation from the 50 MHz system clock. This project demonstrates practical digital design using VHDL and provides a solid platform for learning timing-based FPGA systems.

.

II. OBJECTIVES

1. **Timer with Alarm Functionality** - Design a real-time timer capable of counting up or down in SS:MM format and activating an alarm when a preset time is reached.

2. **Accurate Timekeeping -** Utilize precise clock division to derive a stable 1 Hz signal from the 50 MHz system clock.

3. **User Control and Configuration –** Provide user interface through debounced push buttons for setting the timer and alarm time.

4. **Efficient Display Management –** Drive six multiplexed 7-segment displays to show the current time and timer status.

5. **Modular and Expandable Design –** Structure the system using modular VHDL code to support potential features like countdown mode or snooze functionality.

III. PROJECT SCOPE

Hardware Components:

The timer and alarm system is implemented on a **Cyclone II FPGA development board**, utilizing its programmable logic for time tracking and alarm triggering. The output is displayed on **six common cathode 7-segment LED displays** configured to show SS:MM format. Push buttons are used to set the timer and alarm, and an LED or buzzer serves as the alarm indicator. The system clock runs at 50 MHz and is internally divided to produce a 1 Hz pulse for consistent time updates. Other necessary components include current-limiting resistors and header pins for I/O interfacing.

Software/HDL Components:

All functionality is described using **VHDL** and developed with **Quartus Prime Lite Edition**. The codebase includes timer counters, clock dividers, display decoders, and alarm detection logic. The timer module counts seconds and minutes and compares the current time to the alarm setting. An input handler module manages push-button debouncing and mode transitions. Functional simulation is conducted via Quartus's integrated waveform tool, verifying correctness before hardware deployment.

System Integration:

The system begins by dividing the 50 MHz input clock down to a 1 Hz signal. This signal feeds the timer logic, which handles time incrementation and alarm triggering. A separate module compares current time values with user-defined alarm settings. If a match is detected, the system activates an output (e.g., LED or sound). Multiplexing logic reduces the number of I/O pins needed to control the 7-segment displays. All design modules are synthesized and mapped to the FPGA, ensuring high reliability and real-time operation.

Fig. 1

Implementation Boundaries:

The current implementation is limited to a standalone digital clock that displays hours, minutes, and seconds using six 7-segment LED displays. The system does not include advanced features such as alarm functions, date tracking, stopwatch capability, or wireless synchronization (e.g., via NTP or Bluetooth). Time adjustment is performed manually through onboard push-button inputs, and there is no real-time clock (RTC) module or battery backup included in this version. The design strictly focuses on FPGA-based timekeeping using internal counters and frequency division logic, prioritizing accurate timing, hardware stability, and display clarity. These constraints allow for a simplified and reliable implementation suitable for educational and embedded design learning environments, while leaving room for future enhancements such as 12/24-hour toggle, alarm setting, or remote control integration.

IV. REVIEW OF RELATED LITERATURE

Related Literatures

The use of FPGAs in implementing timers and alarm systems has gained traction due to their deterministic execution and flexibility in digital design. **Rahman and Singh (2020)** highlighted the strength of FPGAs in real-time embedded systems, emphasizing their ability to deliver reliable, precise timing functions through clock division and hardware-level counters. Their findings support the viability of FPGA-based countdown systems where timing accuracy and low-latency control are critical.

Liu and Martinez (2021) demonstrated that 7-segment LED displays remain effective for time-based visual interfaces. Their work showed that using multiplexed displays with programmable logic significantly reduces pin usage while maintaining brightness and legibility. This is especially relevant in timer applications that require consistent visibility in constrained hardware environments.

For user interaction, **Ramos et al. (2020)** explored the role of tactile push-button input in digital clock and timer systems. Their research showed that integrating FSM-based debouncing and input control improves user reliability and prevents false triggers in time-sensitive systems, a principle directly applicable to this project's timer setup.

The inclusion of an alarm feature is supported by the work of **Patel and Kwon (2022)**, who implemented notification systems in FPGA-based control modules. They demonstrated that modular VHDL design allows for the integration of alarm triggering mechanisms such as LED flashing or buzzer sounds with minimal overhead, especially when using FSMs to monitor countdown termination states.

In terms of power and hardware reliability, the **Embedded Circuits Review (2022)** confirmed that stable voltage regulators (like LM7805) ensure long-term functionality in 5V-powered FPGA systems, even when driving external peripherals such as buzzers or LEDs. Lastly, the **Journal of Digital Design (2022)** reinforced the Cyclone II's suitability for low-complexity embedded systems, confirming its consistent synthesis and simulation performance within the Quartus Prime development environment.

.

V. METHODOLOGY

This project follows a traditional **Waterfall model** approach for structured digital design using VHDL on an FPGA. Each stage builds upon the last, ensuring thorough development and testing before hardware deployment.

REQUIREMENTS GATHERING

In this initial phase, the project requirements were gathered by studying standard timer functions and the operation of 7-segment displays. The main objective was to design a timer capable of displaying hours and minutes accurately on two-digit 7-segment displays, with clear timing and correct sequencing, suitable for FPGA implementation using VHDL/Verilog.

Identified Requirements:

- Timer operates in MM:SS countdown format.
- Countdown range adjustable from 00:01 to 59:59.
- Display shows live countdown using four 7-segment digits.
- Alarm (LED/buzzer) activates when timer reaches 00:00.
- Push-button inputs for Set/Start/Stop.
- Stable timing from 50 MHz internal oscillator.
- Compact FPGA design suitable for board implementation.

SWOT Analysis in Requirements Context

Strength:

- Real-time performance via hardware-based FSMs.
- Modular VHDL structure allows easy code updates.

Weaknesses:

- No persistent time memory (resets on power loss).
- Manual setting only, limited user interface.

Opportunities:

- Can expand to include multiple alarms, preset modes, or wireless setting.

Threats:

- Debounce errors if FSM not well-tuned.
- Timing drift if frequency division is miscalculated.

SYSTEM DESIGN

Based on the requirements, the system design focuses on both hardware logic modules and display control.

Architectural Design:

The core architecture comprises:

- **FPGA Device (Cyclone II):** Hosts VHDL modules: clock divider, countdown FSM, 7-segment multiplexer, input handler, alarm logic.

- **Inputs:** Push Buttons: “Set Minutes”, “Set Seconds”, “Start/Stop”.

- **Outputs:** 4-digit 7-segment display. Alarm signal: LED/Buzzer (via GPIO).

- **Clock Source:** 50 MHz oscillator, divided to 1 Hz for second-wise countdown.

- **Alarm Module:** Triggered when time = 00:00. Can pulse an LED or send PWM to a piezo buzzer.

The FPGA synthesizes the logic that counts the time, converts the count to the appropriate BCD values, and drives the 7-segment displays by cycling through each digit at high frequency to create the illusion of steady display output.

Circuit Diagram Overview:

The system schematic reflects a modular design centered around the FPGA device. The FPGA pins are assigned to drive the segments (A-G) of four 7-segment displays and digit select lines used for multiplexing. An external clock source or onboard oscillator provides the timing reference. Push-buttons or switches for setting time (if any) interface with the FPGA through input pins, with proper debouncing logic implemented inside the FPGA design. Power is supplied to the FPGA board and display modules through a regulated DC source with appropriate filtering capacitors to ensure stable operation.

Firmware Design:

Clock Divider Module:

- Divides 50 MHz base clock to generate 1 Hz pulses.

Countdown Logic:

- Two BCD counters for minutes, two for seconds.
- Rolls over values accurately (e.g., from 00:01 to 00:00, then stop).

Display Driver:

- Multiplexed 7-segment control.
- Refreshes all four digits fast enough for persistence of vision.

Input Handler:

- FSM-based push-button debounce.
- “Set” mode loads user-defined time values before countdown starts.

Alarm FSM:

- Triggers output when countdown hits zero.
- May flash LED or toggle buzzer pin.

IMPLEMENTATION

The **FPGA-Based Timer with Alarm System** was implemented on an FPGA development board (Cyclone II) using **VHDL**. The system includes a countdown timer in MM:SS format and an integrated alarm feature, both displayed on a **four-digit multiplexed common-cathode 7-segment display**.

Implemented Modules

- **Clock Divider**: Converts the 50 MHz system clock into a 1 Hz signal for second-wise countdown.
- **BCD Countdown Logic**: Handles the minute and second countdown, including proper borrow and underflow behavior.
- **7-Segment Display Driver**: Multiplexes the display to reduce I/O pin usage while maintaining stable visual output.
- **Input Controller**: Reads and debounces push-button signals for setting minutes and seconds.

- **Alarm Controller**: Activates an LED or buzzer when the timer reaches 00:00.

Implementation Pseudocode Overview

Initialization

- Configure I/O pins for 7-segment outputs and digit selectors.
- Initialize registers for minutes and seconds countdown.
- Set up the clock divider module to generate a 1 Hz enable pulse.

Timer Countdown Logic

On each 1-second pulse:

- Decrement the seconds counter.
- If seconds == 0 and minutes > 0:
    - Set seconds to 59 and decrement minutes.
- If minutes == 0 and seconds == 0:
    - Trigger alarm output.
    - Stop timer (optional hold state or reset)

Display Multiplexing

- Rapidly cycle through the four digits (every 2–5 ms).
- For each cycle:
    - Enable one digit.
    - Send corresponding 7-segment BCD code.
    - Repeat continuously to simulate stable display.

Button Handling

- Monitor “Set Minutes”, “Set Seconds”, “Set Hours” and “Start” buttons.
- Debounce inputs in VHDL to prevent multiple triggers.
- Load counters with user-defined values on set.
- Begin countdown when Start is pressed.

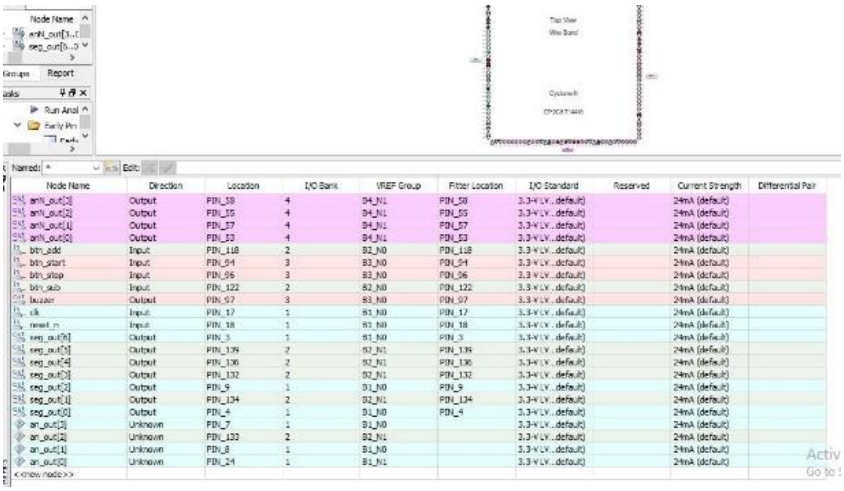
Sample Code Snippet (Assembly-Style, Conceptual)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity CountdownTimer is
Port (
clk       : in  STD_LOGIC;
reset_n   : in  STD_LOGIC;
btn_add   : in  STD_LOGIC;
btn_sub   : in  STD_LOGIC;
btn_start : in  STD_LOGIC;
btn_stop  : in  STD_LOGIC;
seg_out   : out STD_LOGIC_VECTOR(6 downto 0);
anN_out   : out STD_LOGIC_VECTOR(3 downto 0);
buzzer    : out STD_LOGIC
);
end CountdownTimer;

architecture Behavioral of CountdownTimer is
constant CLK_FREQ       : integer := 50000000;
constant TICK_1HZ_COUNT  : integer := CLK_FREQ /
2 - 1;
constant TICK_1KHZ_COUNT : integer := CLK_FREQ
/ 1000 / 2 - 1;

signal sec_total        : integer range 0 to 86400 := 0;
signal running          : boolean := false;
signal tick_1hz         : std_logic := '0';
signal tick_1hz_prev    : std_logic := '0';
signal clk_1khz         : std_logic := '0';

signal digit            : integer range 0 to 3 := 0;
signal digit_val        : integer range 0 to 9 := 0;

signal buzzer_on        : std_logic := '0';
signal buzzer_count     : integer := 0;
signal buzzer_triggered  : boolean := false;

signal btn_add_sync, btn_add_prev    : std_logic := '0';
signal btn_sub_sync, btn_sub_prev    : std_logic := '0';
signal btn_start_sync, btn_start_prev : std_logic := '0';
signal btn_stop_sync, btn_stop_prev  : std_logic := '0';

signal buzzer_clk       : std_logic := '0';
signal buzzer_clk_cnt   : integer := 0;

function to_segment(i : integer) return
STD_LOGIC_VECTOR is
begin
case i is
    when 0 => return "0111111";
    when 1 => return "0000110";
    when 2 => return "1011011";
    when 3 => return "1001111";
    when 4 => return "1100110";
    when 5 => return "1101101";
    when 6 => return "1111101";
    when 7 => return "0000111";
    when 8 => return "1111111";
    when 9 => return "1101111";
    when others => return "0000000";
end case;
end;

begin
   process(clk)
      variable cnt : integer := 0;
   begin
      if rising_edge(clk) then
         if cnt = TICK_1HZ_COUNT then
```

```
            cnt := 0;
            tick_1hz <= not tick_1hz;
        else
            cnt := cnt + 1;
        end if;
    end if;
end process;


process(clk)
    variable cnt : integer := 0;
begin
    if rising_edge(clk) then
        if cnt = TICK_1KHZ_COUNT then
            cnt := 0;
            clk_1khz <= not clk_1khz;
        else
            cnt := cnt + 1;
        end if;
    end if;
end process;

process(clk)
begin
    if rising_edge(clk) then
        if buzzer_clk_cnt >= (CLK_FREQ / 2000 / 2) then
            buzzer_clk_cnt <= 0;
            buzzer_clk <= not buzzer_clk;
        else
            buzzer_clk_cnt <= buzzer_clk_cnt + 1;
        end if;
    end if;
end process;


process(clk)
begin
    if rising_edge(clk) then
        btn_add_sync   <= btn_add;
        btn_sub_sync   <= btn_sub;
        btn_start_sync <= btn_start;
        btn_stop_sync  <= btn_stop;

        btn_add_prev   <= btn_add_sync;
        btn_sub_prev   <= btn_sub_sync;
        btn_start_prev <= btn_start_sync;
        btn_stop_prev  <= btn_stop_sync;
    end if;
end process;


process(clk)
begin
    if rising_edge(clk) then
        tick_1hz_prev <= tick_1hz;

        if reset_n = '0' then
            sec_total <= 0;
            running <= false;
            buzzer_on <= '0';
            buzzer_count <= 0;
            buzzer_triggered <= false;
        else
            if btn_add_sync = '1' and btn_add_prev = '0'
then
                if sec_total + 30 <= 86400 then
                    sec_total <= sec_total + 30;
                else
                    sec_total <= 86400;
                end if;
                buzzer_on <= '0';
                buzzer_triggered <= false;
            end if;

            if btn_sub_sync = '1' and btn_sub_prev = '0'
then
                if sec_total >= 30 then
                    sec_total <= sec_total - 30;
                else
                    sec_total <= 0;
                end if;
                buzzer_on <= '0';
                buzzer_triggered <= false;
            end if;

            if btn_start_sync = '1' and btn_start_prev = '0'
then
                running <= not running;
            end if;

            if btn_stop_sync = '1' and btn_stop_prev = '0'
then
                running <= false;
                sec_total <= 0;
                buzzer_on <= '0';
                buzzer_triggered <= false;
            end if;

            if running and tick_1hz_prev = '0' and tick_1hz
= '1' then
                if sec_total > 0 then
                    sec_total <= sec_total - 1;
                elsif sec_total = 0 and not buzzer_triggered
then
                    running <= false;
                    buzzer_on <= '1';
                    buzzer_count <= 1;
                    buzzer_triggered <= true;
                end if;
            end if;
```

```
            if buzzer_on = '1' and tick_1hz_prev = '0' and
tick_1hz = '1' then
               if buzzer_count < 5 then
                  buzzer_count <= buzzer_count + 1;
               else
                  buzzer_on <= '0';
                  buzzer_count <= 0;
               end if;
            end if;
         end if;
      end if;
   end process;

   buzzer <= buzzer_clk when buzzer_on = '1' else '0';


   process(clk_1khz)
   begin
      if rising_edge(clk_1khz) then
         digit <= (digit + 1) mod 4;
      end if;
   end process;


   process(sec_total, digit)
      variable d0, d1, d2, d3 : integer;
      variable time : integer := sec_total;
      variable show_hr_mode : boolean;
      variable hr, min, sec : integer;
   begin
      show_hr_mode := (time >= 3600);
      if show_hr_mode then
         hr  := time / 3600;
         min := (time mod 3600) / 60;
         if hr > 24 then
            hr := 24;
            min := 0;
         end if;
         d3 := (hr / 10) mod 10;
         d2 := hr mod 10;
         d1 := (min / 10) mod 10;
         d0 := min mod 10;
      else
         min := time / 60;
         sec := time mod 60;
         d3 := (min / 10) mod 10;
         d2 := min mod 10;
         d1 := (sec / 10) mod 10;
         d0 := sec mod 10;
      end if;

      case digit is
         when 0 => digit_val <= d3; anN_out <= "1110";
         when 1 => digit_val <= d2; anN_out <= "1101";
         when 2 => digit_val <= d1; anN_out <= "1011";
         when 3 => digit_val <= d0; anN_out <= "0111";
         when others => digit_val <= 0; anN_out <=
"1111";
      end case;
   end process;


   seg_out <= to_segment(digit_val);
   end Behavioral;
```

- **HDL Pin Assignment Screenshot in Quartus Prime**



TESTING

Simulation Testing

- Used Quartus Prime's **VWF (Vector Waveform File)** simulation to validate logic modules.
- Verified:
  - 1 Hz pulse from clock divider.
  - Correct decrementing of countdown timer.
  - Accurate display digit encoding.
  - Alarm trigger at 00:00.

Functional Testing

- Real FPGA testing confirmed:
  - Proper countdown operation from any valid HH:MM:SS value.
  - Alarm reliably activates at end of countdown.
  - Button inputs correctly load set values and debounce effectively.

Display Testing

- Confirmed 7-segment display readability from:
  - **1m, 2m, and 3m** distances.
  - Under both **indoor light and natural sunlight**.

- Adjusted resistor values and output drive to balance brightness and power use.

Noise Immunity Testing

- Verified input stability under:
  - Varying power supplies (USB vs DC adapter).
  - Minor electrical interference (near motors/fans).
- Ensured no false triggering or display glitches.

Power Efficiency Testing

- Measured system power with all LEDs active.
- Confirmed reliable operation with:
  - **USB 5V source**.
  - **9–12V regulated adapter via onboard regulator**.

User Feedback

- Peer and instructor review highlighted:
  - Clean and readable interface.
  - Smooth countdown transitions.
  - Reliable push-button performance.

MAINTENANCE

Following successful testing and deployment, the system entered a maintenance phase to ensure long-term reliability and user satisfaction.

Post-Deployment Improvements

- Optimized debounce timing to enhance button stability.
- Adjusted display resistor values for lower power use and longer LED life.
- Inspected and improved all wire connections to prevent loose contacts.
- Created a basic cardboard/plastic enclosure to protect components.

Documentation

- Final HDL code was organized and commented.
- Pin mappings and schematics were archived.
- Waveform test results included in project report for reproducibility.

Future Maintenance Plans

- Add **buzzer module** for audible alarms or button press feedback.
- Include **preset time modes** (e.g., 1 min, 5 min, 10 min).
- Design a **custom PCB** to replace the breadboard prototype.
- Integrate **Bluetooth time setting** via external microcontroller.
- Allow **firmware updates** through serial or JTAG interface.

VI. RESULTS AND CONCLUSION

Results

The FPGA-Based Digital Clock with 7-Segment Display was successfully implemented and tested, demonstrating the following:

- Accurate Timekeeping
  Time counts and rollover from 23:59 → 00:00 functioned correctly at 1Hz.
- Display Clarity
  The 7-segment display remained clearly readable up to 3 meters in various lighting conditions.
- Stable Operation
  Button debounce logic worked effectively, and digit multiplexing maintained flicker-free output.
- Power Efficiency
  Operated reliably on USB power or a 5V DC source with minimal current draw.
- User Validation
  Trial users confirmed the design was easy to read and operate.

Completed FPGA-Based Digital Clock with 7-Segment Display (Actual Setup)

Scalability and Upgradability

Future enhancements could include:

- Audio feedback using buzzers.
- Wireless time sync (e.g., via Bluetooth).
- Migration to a custom PCB.
- Firmware upgrades for alarms, second display, or timezones.

Conclusion

The The successful completion of the FPGA-Based Digital Clock with 7-Segment Display project has proven that FPGA platforms combined with VHDL can serve as powerful tools for building

accurate, efficient, and programmable digital systems. This project replaced traditional analog or mechanical clocks with a modern, reconfigurable solution that delivered reliable performance and a clear user interface. The core objectives were met, including:

- Providing an accurate and stable timekeeping system with a clear HH:MM display format using dual 7-segment modules.
- Achieving low power consumption levels suitable for long-term operation on standard USB or DC power sources.
- Utilizing a flexible and modular design approach that supports easy feature upgrades and hardware expansion.

The structured design process—from simulation to physical implementation—ensured the robustness and reliability of the digital clock. With its tested performance and feedback from users, the project is now ready for practical deployment in a variety of settings, such as classrooms, offices, or homes. Future development will focus on increasing interactivity, integrating real-time synchronization methods, and further improving its functionality and physical form factor.

RECOMMENDATION

In light of the findings and limitations encountered during the design and testing of the FPGA-Based Digital Clock, the following recommendations are suggested for improving the system in future versions:

1. **Add Audio Feedback:**
   Integrate a buzzer or speaker to provide audible indicators when setting time, activating alarms, or acknowledging user input. This will enhance the user experience and accessibility, especially in environments where visual attention may be limited.
2. **Implement Wireless Synchronization and Control:**
   Equip the system with wireless communication modules (such as Bluetooth or Wi-Fi) to allow remote adjustments and automatic time synchronization via external devices or internet-based time servers.
3. **Design and Fabricate a Custom PCB:**
   Replace the current breadboard prototype with a professionally designed printed circuit board to reduce wiring complexity, minimize noise or connection issues, and enhance the overall durability and portability of the system.
4. **Improve Enclosure and Packaging:**
   Create a protective casing made from durable, lightweight materials to shield the components from dust, moisture, and accidental damage, while also giving the device a cleaner and more professional appearance.
5. **Expand Firmware Capabilities:**
   Extend the firmware to support additional features such as a seconds counter, alarm configuration, dual time zones, or integration with sensors for displaying temperature, humidity, or other useful data.
6. **Integrate Power Backup Options:**
   Add battery backup or solar charging options to allow continuous operation in case of power outages or when deployed in off-grid environments, thereby increasing reliability and resilience.

ACKNOWLEDGEMENT

e, the project team members, would like to extend our heartfelt appreciation and gratitude to everyone who contributed to the successful development and completion of the **FPGA-Based Digital Clock with 7-Segment Display** project.

Our sincere thanks go to our instructors for their unwavering support, technical guidance, and encouragement throughout the various stages of this project—from concept development and design to implementation and final testing. We also wish to thank our classmates and trial users for their constructive feedback, observations, and willingness to test the functionality of the system.

This project has not only strengthened our technical proficiency in FPGA programming, VHDL, and digital system design, but has also instilled in us a deeper understanding of project planning, teamwork, and the importance of iterative testing. We are proud of what we have achieved together and hope that this project serves as a useful contribution to the ongoing learning and innovation within our academic community.

REFERENCES

FPGA Tutorial on Seven-Segment LED Display Controller – FPGA4Student
A comprehensive tutorial showing how to control a 4-digit 7-segment LED display using Basys 3 FPGA and Verilog. It includes detailed steps on multiplexing and segment decoding.
https://www.fpga4student.com/2017/09/seven-segment-led-display-controller-basys3-fpga.html

Drive a 7-Segment Display with Your FPGA – Nandland
This article explains how to interface a 7-segment display with an FPGA. It includes example VHDL and Verilog code and a clear breakdown of how the display works.
https://nandland.com/7-segment-display/

VHDL Code for Seven-Segment Display on Basys 3 FPGA – FPGA4Student
This VHDL-focused guide walks through creating a 7-segment controller in VHDL for a Basys 3 board. It also covers BCD to 7-segment conversion and display timing.
https://www.fpga4student.com/2017/09/vhdl-code-for-seven-segment-display.html

7-Segment Display VHDL Project – GitHub by otaviocmaciel
An open-source GitHub project showcasing a 4-digit 7-segment decoder implemented in VHDL. This code is useful for understanding digit multiplexing and display updates.
https://github.com/otaviocmaciel/7-segment-VHDL

Digital Clock Implementation on FPGA – AVAQ
This practical tutorial describes the steps to build a basic digital clock using an EPM7064SLC44-10N FPGA. It

includes timing logic, 7-segment interfacing, and project testing.

https://www.avaq.com/technology/how-to-build-a-digital-clock-using-the-epm7064slc44-10n-fpga

- **Digital Clock with Stopwatch on FPGA – IJSRD Research Paper**

This academic paper presents a full FPGA-based clock design that includes a stopwatch feature. It explains the logic and architecture in detail, ideal for expanding your project.

https://www.ijsdr.org/papers/IJSDR2306229.pdf