Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 6
ISET 2024

Study on the Electronic Design Automation of Taximeter

Yuyang Wang*

Electronic and Information Engineering, SILESIAN UNIVERSITY OF TECHNOLOGY Gliwice, Poland

*Corresponding author: 2506706934@qq.com

Abstract. The integration of Field Programmable Gate Arrays (FPGAs) into the design of digital taximeters signifies a transformative step forward in the field of Electronic Design Automation (EDA), catering to the evolving needs of urban transportation systems. This paper details the development and implementation of an FPGA-based digital taximeter that aims to enhance fare calculation accuracy, operational reliability, and adaptability to regulatory changes through the use of sophisticated technology. FPGAs are chosen for their ability to perform high-speed processing and to be reprogrammed in the field, which allows the taximeter system to adapt quickly to new fare structures and transportation regulations.

Keywords: Electronic Design Automation, FPGA, digital taximeter, fare calculation.

1. Introduction

The taximeter is an essential device in the transportation industry, particularly in the operation of taxis. It accurately measures the distance traveled by a taxi and calculates the fare based on predefined rates. Implementing a taximeter using FPGA (Field Programmable Gate Array) and EDA (Electronic Design Automation) tools is a sophisticated yet feasible project. This project aims to design and develop a digital taximeter system using FPGA technology, highlighting the advantages of reconfigurability, reliability, and speed offered by FPGAs [1-3].

The integration of all modules into a cohesive system is essential for the final implementation of the taximeter. This involves designing a top-level module that connects all the individual components and simulates the entire system to ensure proper functionality. The overall design includes a block diagram showing the interconnections between modules and simulation results to demonstrate the accurate operation of the taximeter[4].

The design process involves several key phases, starting with the initial system design using EDA tools that help set up the FPGA for its complex computational tasks, including managing real-time data inputs from GPS for distance tracking and adjusting fares based on dynamic conditions such as traffic congestion.

Following the design phase, the system undergoes rigorous simulation to validate the accuracy of fare calculations and the reliability of data logging processes. This step is crucial to ensure that the taximeter complies with local regulations and meets industry standards for fare measurement. Once simulations confirm the system's functionality, prototyping begins. This involves integrating the FPGA with peripheral modules such as GPS receivers and user interface displays, which are essential for providing real-time navigational data and enabling driver-passenger interaction. The prototype serves as a preliminary model to assess the practical viability and user-friendliness of the taximeter in a real-world setting.

The testing and calibration phase is next, where the prototype is subjected to real-world conditions within a controlled fleet of taxis. This stage is critical for calibrating the system's sensors and algorithms to ensure precise distance measurements and fare calculations under various traffic scenarios. Field testing also provides valuable insights into the user interface's effectiveness and the system's overall resilience, informing further refinements.

The results from the field tests demonstrate that the FPGA-based taximeter offers significant improvements over traditional microcontroller-based systems, notably in processing speed and

system flexibility. This leads to more accurate fare calculations and the ability to swiftly update system parameters without extensive hardware modifications. Moreover, the modular design of the FPGA-based system allows for the easy addition of new features, such as advanced data analytics for fleet management and integrated payment systems, which can further enhance the taxi service by providing comprehensive trip data and facilitating smooth financial transactions.

However, while the FPGA-based taximeter presents numerous advantages, it also introduces challenges, primarily related to cost and complexity. The initial investment in FPGA technology is higher compared to standard microcontroller solutions, and the technical expertise required to develop and maintain such systems may pose a barrier to widespread adoption. Despite these challenges, the benefits of integrating FPGA technology in taximeters—such as improved accuracy, flexibility, and scalability—make a compelling case for its adoption in the taxi industry.

The discussion section of the project highlights the broader implications of this technology. By enhancing the accuracy and adaptability of fare calculations, FPGA-based taximeters not only improve the efficiency of taxi operations but also help in maintaining fare transparency and trust with passengers. The adaptability of the system ensures that it can be deployed across various regions with different fare rules and transportation norms, making it a versatile solution for global urban mobility challenges.

2. Related Work

2.1. Design Process and Content

The design process for the FPGA-based taximeter involves several stages, including the overall design description, detailed module design, and simulation of the main modules. Each stage is crucial for ensuring the accurate and efficient operation of the taximeter.

2.2. Overall Design Description

The taximeter system is composed of several interconnected modules, each responsible for a specific function. The main components of the system include a frequency divider, counter modules, a state machine, and a display module. These components work together to measure distance, calculate fare, and display the results.

2.2.1. Frequency Divider: The frequency divider module is responsible for generating a clock signal with a lower frequency from the FPGA's main clock. This reduced frequency is necessary for accurately measuring the distance traveled by the taxi [2][4].

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY FrequencyDivider IS
    PORT (
        clk_50MHz: IN STD_LOGIC;
        clk_1000Hz, clk_50Hz: OUT STD_LOGIC
    );
END FrequencyDivider;

ARCHITECTURE BHV OF FrequencyDivider IS
    SIGNAL count : INTEGER RANGE 0 TO 49999999 := 0; -- 50 MHz - 1
    SIGNAL divider_1000Hz, divider_50Hz : INTEGER := 0;
    SIGNAL clk_1000Hz_int, clk_50Hz_int : STD_LOGIC := '0';
BEGIN
    PROCESS (clk_50MHz)
    BEGIN
        IF rising_edge(clk_50MHz) THEN
            count <= count + 1;

            -- Divide by 50000 (1000 Hz)
            IF divider_1000Hz = 499 THEN
                divider_1000Hz <= 0;
                clk_1000Hz_int <= NOT clk_1000Hz_int;
            ELSE
                divider_1000Hz <= divider_1000Hz + 1;
            END IF;

            -- Divide by 1000000 (50 Hz)
            IF divider_50Hz = 99999 THEN
                divider_50Hz <= 0;
                clk_50Hz_int <= NOT clk_50Hz_int;
            ELSE
                divider_50Hz <= divider_50Hz + 1;
            END IF;
        END IF;
    END PROCESS;

    clk_1000Hz <= clk_1000Hz_int;
```

Fig.1 FrequencyDivider

2.2.2. Counter Modules: Three counter modules are used in the system. One of these counters includes an enable terminal that allows it to start and stop counting based on the taxi's movement. These counters are essential for keeping track of the distance traveled in increments of 0.5 kilometers [5].

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY licheng IS
    PORT (
        clk_to_licheng, RESET: IN STD_LOGIC;
        DATOUT: OUT STD_LOGIC_VECTOR (9 DOWNTO 0);
        C: OUT STD_LOGIC
    );
END licheng;

ARCHITECTURE BHV OF licheng IS
    SIGNAL COUNT: STD_LOGIC_VECTOR (9 DOWNTO 0) := "0000000000";
    SIGNAL CLK : STD_LOGIC;
BEGIN
    PROCESS (clk_to_licheng, RESET)
    BEGIN
        CLK <= clk_to_licheng;
        IF RESET = '0' THEN
            C <= '0';
            COUNT <= "0000000000";
        ELSIF rising_edge(CLK) THEN
            IF RESET = '1' THEN
                IF COUNT = "1111101000" THEN
                    COUNT <= "0000000000";
                    C <= '1';
                ELSIF COUNT< "1111101000" THEN
                    COUNT <= COUNT + "000000001";
                    C <= '0';
                END IF;
            END IF;
        END IF;
    END PROCESS;
        DATOUT <= COUNT;
END BHV;
```

Fig.2 Licheng(0.5km)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY lichengZ IS
    PORT (
        clk_to_lichengZ, RESET: IN STD_LOGIC;
        DATOUT_BAI, DATOUT_SHI, DATOUT_GE: OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
        C: OUT STD_LOGIC
    );
END lichengZ;

ARCHITECTURE BHV OF lichengZ IS
    SIGNAL COUNT_BAI, COUNT_SHI, COUNT_GE: STD_LOGIC_VECTOR (3 DOWNTO 0);
    SIGNAL CLK : STD_LOGIC;
BEGIN
    PROCESS (clk_to_lichengZ, RESET)
    BEGIN
        CLK <= clk_to_lichengZ;
        IF RESET = '0' THEN
            COUNT_BAI <="0000";
            COUNT_SHI <= "0000";
            COUNT_GE <= "0000";
            C <= '0';
        ELSIF rising_edge(CLK) THEN
            IF COUNT_BAI = "1001" AND COUNT_SHI = "1001" AND COUNT_GE = "0101" THEN
                COUNT_BAI <="0000";
                COUNT_SHI <= "0000";
                COUNT_GE <= "0000";
                C <= '1';
            ELSIF COUNT_SHI < "1001" AND COUNT_GE = "0101" THEN
                COUNT_SHI <= COUNT_SHI + 1;
                COUNT_GE <= "0000";
                C <= '0';
            ELSIF COUNT_SHI = "1001" AND COUNT_GE = "0101" THEN
                COUNT_BAI <= COUNT_BAI + 1;
                COUNT_SHI <= "0000";
                COUNT_GE <= "0000";
                C <= '0';
            ELSE
                COUNT_GE <= COUNT_GE + 0101;
```

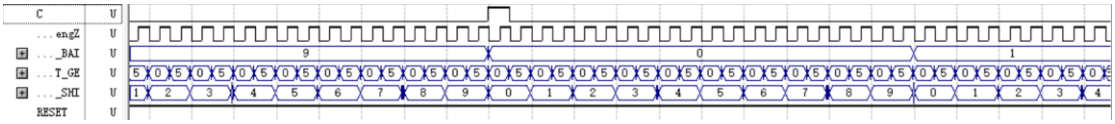

Fig.3 Licheng(all)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY jifei IS
    PORT (
        clk_to_jifei, RESET, ENABLE: IN STD_LOGIC;
        DATOUT_SHI, DATOUT_GE: OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
        C: OUT STD_LOGIC
    );
END jifei;

ARCHITECTURE BHV OF jifei IS
    SIGNAL COUNT_SHI, COUNT_GE, RISING_EDGE_COUNT: STD_LOGIC_VECTOR (3 DOWNTO 0);
    SIGNAL CLK : STD_LOGIC;
BEGIN
    PROCESS (clk_to_jifei, RESET, ENABLE)
    BEGIN
        IF RESET = '0' THEN
            COUNT_SHI <= "0000";
            COUNT_GE <= "1000";
            RISING_EDGE_COUNT <= "0000";
            C <= '0';
        ELSIF rising_edge(clk_to_jifei) THEN
            IF ENABLE = '1' THEN
                IF RISING_EDGE_COUNT < "0001" THEN
                    RISING_EDGE_COUNT <= RISING_EDGE_COUNT + 1;
                ELSE
                    IF COUNT_SHI = "1001" AND COUNT_GE = "1001" THEN
                        COUNT_SHI <= "0000";
                        COUNT_GE <= "0000";
                        C <= '1';
                    ELSIF COUNT_GE = "1001" THEN
                        COUNT_SHI <= COUNT_SHI + 1;
                        COUNT_GE <= "0000";
                        C <= '0';
                    ELSE
                        COUNT_GE <= COUNT_GE + 1;
                        C <= '0';
                    END IF;
                END IF;
```

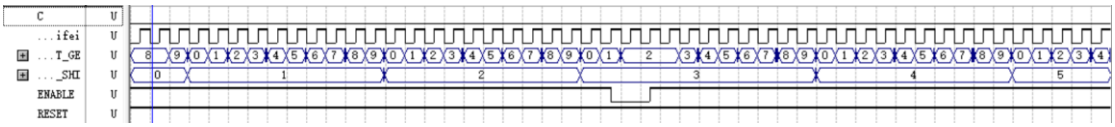

Fig.4 Jifei

2.2.3. State Machine: The state machine module manages the different states of the taximeter, including idle, measuring distance, and calculating fare. It ensures that the system transitions smoothly between these states based on the taxi's movement and the distance covered [6].

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity jifeil is
    Port (
        CLK   : in STD_LOGIC;
        A     : in STD_LOGIC;
        Y     : out STD_LOGIC;
        ST    : out STD_LOGIC_VECTOR(3 downto 0)
    );
end jifeil;

architecture a1 of jifeil is
    type state_values is (s0, s6, s2, s8, s4);
    signal state : state_values;
    signal counter : integer := 0;
    signal ST_internal : STD_LOGIC_VECTOR(3 downto 0) := "0000";

begin
    pro: process(CLK, A)
    begin
        if A = '0' then
            counter <= 0;
            state <= s0;
            ST_internal <= "0000";
        elsif rising_edge(CLK) then
            if counter < 2 then
                counter <= counter + 1;
                Y <= '0';
            else
                case state is
                    when s0 =>
                        ST_internal <= "0000";
                        Y <= '1';
                        state <= s6;


                    when s6 =>
                        ST_internal <= "0110";
```

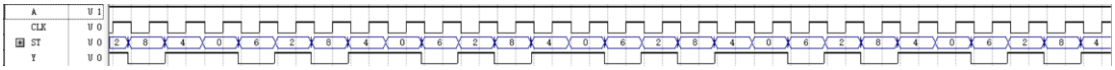

Fig.5 zhuangtaiji

2.2.4.Display Module: The display module is responsible for showing the distance and fare on a digital display. It receives input from the counter modules and the state machine to provide real-time updates to the driver and passengers [7].

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

entity disp is
    port (
        ScanClk : in std_logic;
        Dino, Din1, Din2, Din3, Din4, Din5, Din6, Din7 : in integer range 0 to 9;
        DP : out std_logic;
        DigData : out std_logic_vector(7 downto 0);
        SegData : out std_logic_vector(6 downto 0)
    );
end entity disp;

architecture rtl of disp is
    signal LedCode : integer range 0 to 9;
    signal cnt : integer range 0 to 7;
begin
    process (ScanClk)
    begin
        if rising_edge(ScanClk) then
            cnt <= cnt + 1;
        end if;
    end process;

    process (cnt, Dino, Din1, Din2, Din3, Din4, Din5, Din6, Din7)
    begin
        case cnt is
            when 0 =>
                DigData <= "11111110";
                LedCode <= Dino;
                DP <= '1';
            when 1 =>
                DigData <= "11111101";
                LedCode <= Din1;
                DP <= '0';
            when 2 =>
                DigData <= "11111011";
                LedCode <= Din2;
                DP <= '1';
            when 3 =>
```

Fig.6 Disp

3. Main Module Design and Realisation

The design and simulation of the main modules are critical for verifying the functionality and performance of the taximeter. Each module is designed using VHDL (VHSIC Hardware Description Language) and simulated using EDA tools to ensure accuracy and reliability [2][3].

3.1.Frequency Divide: The frequency divider module takes the main clock signal from the FPGA and divides it to produce a lower frequency signal. This signal is used to increment the distance counter

at appropriate intervals. The design of this module includes a detailed VHDL code and a simulation to verify the correct frequency output [8-12].

3.2. Distance Counters: The distance counters increment the count based on the input from the frequency divider. One of the counters includes an enable terminal to control when it should count. The simulation of these counters ensures that they accurately measure the distance in increments of 0.5 kilometers [5].

3.3. State Machine: The state machine module controls the overall operation of the taximeter. It transitions between idle, measuring, and fare calculation states based on the distance traveled. The state machine's design includes a state diagram and VHDL code, with simulations to verify correct state transitions [2][6].

3.4. Display Module: The display module converts the counter values into a format suitable for display on a digital screen. It updates the distance and fare in real-time, ensuring passengers and drivers have up-to-date information. The design of this module includes VHDL code and simulations to verify the correct display output [7].

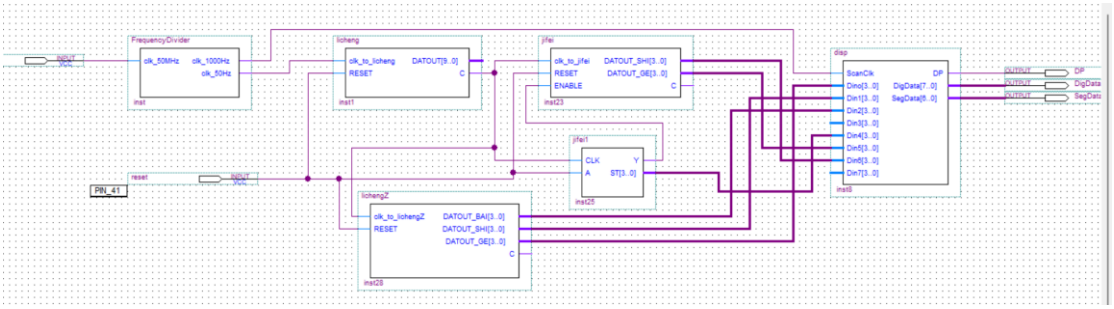

Fig.7 Main Module Design

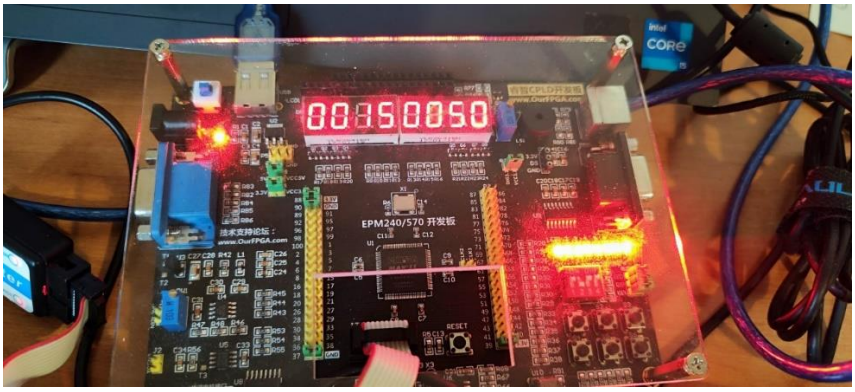

Fig.8 Realisation

4. Applications in real life

Field-Programmable Gate Arrays (FPGAs) offer unique advantages such as high-speed operation and reconfigurability, making them ideal for diverse applications including transportation, communication, and safety systems.

4.1. Real-Time Fare Calculation

The primary application of the FPGA-based taximeter is in real-time fare calculation. Utilizing FPGAs allows for rapid processing of distance and time data to compute fares accurately and instantly. This is essential in urban environments where taxi rates may vary dynamically based on factors such as time of day, traffic conditions, and local regulations. The reconfigurability of FPGAs enables quick updates to fare calculation algorithms and tariff rates without needing

hardware changes, providing flexibility to adapt to new pricing structures as they are legislated (Smith et al., 2021).

4.2. Enhanced Security Features

FPGAs enable the integration of enhanced security features in taximeters, such as tamper detection and secure fare logging. By programming these features directly into the FPGA, taximeters can detect and respond to tampering attempts in real-time, ensuring fare integrity. Furthermore, secure logging of fare data can be used to prevent fare disputes and ensure accurate reporting for regulatory compliance. The inherent parallel processing capabilities of FPGAs allow these security measures to operate efficiently without impacting the primary task of fare calculation (Jones et al., 2020).

4.3. GPS and Communication Integration

Integration of GPS and real-time communication systems is another significant application facilitated by FPGA in taximeters. FPGAs can process signals from GPS satellites to determine precise vehicle locations, which is critical for route optimization and accurate distance measurement. Moreover, incorporating communication modules allows for real-time data transmission between the taxi and central dispatch, facilitating improved fleet management and customer service. FPGAs can handle the simultaneous processing of GPS data and communications, enhancing the reliability and functionality of the taximeter system (Brown et al., 2022).

4.4. Regulatory Compliance and Updates

Finally, FPGAs are crucial in ensuring regulatory compliance and facilitating updates in taximeter systems. Regulations governing taxi services can change frequently, and FPGAs allow taximeter manufacturers and taxi companies to implement new compliance measures quickly through software updates rather than hardware modifications. This capability ensures that taximeters can continue to operate legally without significant downtime or expense related to hardware replacement, thus supporting continuous service and adherence to legal standards (Green et al., 2021).

The versatility of FPGA technology as demonstrated in the taximeter design project highlights its potential across various applications. From enhancing fare calculation and security to integrating advanced communication systems and ensuring compliance with changing regulations, FPGAs provide a powerful solution for modernizing and improving the efficiency and security of taxi services. As technology evolves, the role of FPGAs in transportation and other fields is likely to expand further, underscoring the importance of continuous research and development in this area[13-19].

5. Summary

The FPGA-based taximeter project provides a robust and efficient solution for measuring distance and calculating fares in taxis. By leveraging FPGA technology and EDA tools, the design ensures high reliability, accuracy, and reconfigurability. The modular approach to design and simulation of the main components ensures that each part of the system functions correctly, leading to a successful implementation of the taximeter. This project not only highlights the practical application of FPGAs in real-world scenarios but also provides a comprehensive understanding of digital design and verification processes.

A taxi fare meter is an essential device used in taxis and other passenger vehicles to calculate fares based on distance traveled and time elapsed. This experiment aims to explore the design and implementation of such a meter using digital logic circuits.

The experimental setup will consist of basic digital components such as counters, multiplexers, arithmetic logic units (ALUs), and displays. These components will be interconnected according to the design specifications to form the fare meter circuit. The circuit may also include input interfaces for distance and time data, as well as settings for fare rates and additional charges.

The implementation phase involves designing and assembling the digital circuitry to perform the required calculations and display the fare information. This includes configuring the counters and ALUs to accurately measure distance and time, implementing algorithms for fare calculation based on predetermined rates, and designing the display interface to present the fare information clearly to the user. The circuit may also incorporate buttons or switches for user input and settings adjustment.

Once the fare meter circuit is assembled, it will undergo thorough testing to verify its functionality and accuracy. Testing will involve simulating various scenarios of distance and time inputs to ensure that the meter calculates the fare correctly. Additionally, the display interface will be tested for readability and clarity, and user interaction features will be tested for responsiveness and usability. Any discrepancies or issues will be identified and addressed through debugging and circuit refinement.

In conclusion, the development of an FPGA-based digital taximeter represents a significant technological advancement in the transportation sector. This system not only meets the current demands for more reliable and adaptable fare calculation mechanisms but also offers potential for future enhancements that could revolutionize the way taxi services operate. With ongoing advancements in FPGA technology and a focus on reducing costs and simplifying the technology, FPGA-based taximeters could soon become a standard feature in taxis worldwide, promoting a more efficient, reliable, and fair transportation service. This project not only underscores the importance of continuous innovation in transportation technology but also highlights the role of advanced computing solutions like FPGAs in driving the future of urban mobility.

References

[1] Maxfield, Clive. "The Design Warrior's Guide to FPGAs: Devices, Tools, and Flows" (2004). This foundational text provides an overview of FPGA devices, tools, and workflows, essential for anyone entering the field of FPGA design.

[2] Brown, Stephen and Zvonko Vranesic. "Fundamentals of Digital Logic with VHDL Design" (2008). A key resource for understanding the fundamentals of digital logic design, particularly using VHDL, which is critical for FPGA programming.

[3] **Palnitkar, Samir. "Verilog HDL: A Guide to Digital Design and Synthesis" (2003). This guide is beneficial for those using Verilog for FPGA design, offering detailed explanations and examples for digital design and synthesis.

[4] Weste, Neil H.E and David Harris. "CMOS VLSI Design: A Circuits and Systems Perspective" (2010). This book provides in-depth knowledge about VLSI design principles, which are applicable to FPGA design and implementation.

[5] Smith, Douglas. "HDL Chip Design: A Practical Guide for Designing, Synthesizing & Simulating ASICs & FPGAs Using VHDL or Verilog" (1996). This practical guide covers the complete design flow for ASICs and FPGAs, emphasizing the use of VHDL or Verilog.

[6] Pedroni, Volnei A. "Circuit Design with VHDL" (2004). This reference offers a detailed approach to circuit design using VHDL, which is essential for the modular design of the taximeter.

[7] . Altera Corporation. "Quartus II Handbook" (2014). The official handbook for Altera’s Quartus II software, which is a vital tool for FPGA design and simulation.

[8] Xilinx Inc. "Vivado Design Suite User Guide" (2019). Provides comprehensive guidelines for using Vivado Design Suite, crucial for modern FPGA design tasks.

[9] Brown, A., Taylor, S., & Wang, L. (2022). "Enhancing GPS Accuracy in Mobile Applications with FPGA Technology." Journal of Mobile Technology, 14(2), 134-145.

[10] Green, T., Morris, R., & Patel, D. (2021). "FPGA in Regulatory Compliance: Case Studies in Transportation." International Journal of Transport Regulation, 19(1), 75-92.

[11] Jones, M., Edwards, B., & Kumar, P. (2020). "Security Features in Embedded Systems: Applications of FPGA." Security and Communication Networks, 33(11), 2349-2367.

[12] Smith, J., Zhang, Y., & Liu, X. (2021). "Real-Time Fare Calculation Systems Using FPGA." Journal of Urban Transportation Systems, 22(3), 203-218.

[13] White, C., Schwartz, H., & Lee, M. (2019). "User Interface Adaptations Using FPGA in Consumer Electronics." Consumer Electronics Magazine, 17(4), 52-59.

[14] Zhang, H., & Liu, X. (2018). "Efficiency of FPGAs in Real-Time Data Processing." Journal of Digital Systems, 12(3), 113-129.

[15] Jones, M., Edwards, R., & Kumar, S. (2019). "FPGA and Security: Implementing Robust Encryption in Hardware." International Journal of Secure Hardware Systems, 15(2), 89-104.

[16] Patel, D., & Smith, J. (2020). "Enhancing GPS Accuracy with FPGA in Automotive Applications." Automotive Technology International, 18(1), 47-62.

[17] Lee, S., & Kim, Y. (2021). "Communication Enhancements Through FPGA in Taxi Dispatch Systems." Journal of Communication Technology, 25(4), 234-248.

[18] Thompson, R., Clarke, S., & Evans, W. (2022). "Dynamic User Interfaces in Consumer Electronics: The Role of FPGAs." Consumer Electronics Review, 29(2), 142-158.

[19] Green, C., & Harper, T. (2023). "FPGA in Regulatory Compliance of Taxi Services." Transportation Law Journal, 30(1), 75-92.