International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

FPGA Based Defibrillator Pulse Simulation Using ECG Signal Detection

J Stella Mary[1], K.Harika[2], K.Jahnavi[3], P.Navya[4], G.Himasree[5]

[1]Associate Professor; Department Of Electronics And Communication Engineering Bhoj Reddy Engineering College For Women Hyderabad India.

[2,3,4,5]B.Tech Students ; Department Of Electronics And Communication Engineering Bhoj Reddy Engineering College For Women Hyderabad India.

Mail Id; harikakandi2005@gmail.com[2],kuppajahnavi2005@gmail.com[3], palvainavya09@gmail.com[4], gundalahima9@gmail.com[5]

Abstract

This work proposes a hardware–software co-design architecture for real-time electrocardiogram (ECG) signal transmission and classification using FPGA technology and embedded machine learning. Initially, the ECG signal is preprocessed in MATLAB, where the biomedical waveform is converted into hexadecimal two's complement representation and stored as a memory initialization file. The formatted data is then imported into Block RAM (BRAM) using Xilinx Vivado IP and deployed on a Basys3 FPGA development board.A custom UART transmitter module is implemented to sequentially fetch ECG samples from BRAM and transmit them through serial communication at a baud rate of 9600. On the receiving end, a Raspberry Pi collects the incoming data via UART and reconstructs the 16-bit signed ECG signal. The recovered waveform is subsequently analyzed using a Random Forest–based machine learning model to determine whether the ECG pattern corresponds to a normal or abnormal condition.

The proposed design demonstrates efficient integration of FPGA-based digital hardware with embedded intelligence for biomedical signal processing. The architecture ensures reliable data transmission, reduced hardware complexity, and scalability for advanced signal analysis. This framework highlights the effectiveness of FPGA–embedded platforms for real-time physiological monitoring and intelligent healthcare diagnostics.

Keywords

ECG, FPGA, Basys3, UART Communication, Embedded Machine Learning, Random Forest, Raspberry Pi, Biomedical Signal Processing, Real-Time Monitoring, Hardware–Software Co-design.

Introduction

Cardiovascular diseases remain one of the most significant causes of mortality across the globe and continue to pose serious challenges to public health systems. Heart-related complications such as arrhythmia, myocardial infarction, heart failure, and sudden cardiac arrest account for millions of deaths annually. Many of these conditions can be effectively managed or prevented if abnormalities are detected at an early stage. Consequently, continuous monitoring of cardiac activity and early diagnosis have become essential for improving survival rates and patient outcomes. The rapid evolution of intelligent healthcare technologies has enabled the integration of advanced digital systems into biomedical applications, allowing accurate, automated, and real-time diagnostic support.The electrocardiogram (ECG) is a widely adopted non-invasive technique used to monitor the electrical activity of the heart. It captures electrical impulses generated during the cardiac cycle and provides critical insights into heart rhythm, conduction pathways, and structural irregularities. Typical ECG signals consist of distinct waveform components including the P-wave, QRS complex, and T-wave, each corresponding to specific physiological events in the cardiac cycle. Analysis of these waveforms enables detection of arrhythmias, conduction abnormalities, and ischemic conditions. Due to its diagnostic reliability and simplicity, ECG monitoring has become a fundamental tool in both clinical cardiology and portable health monitoring systems.

Conventional ECG interpretation is generally performed using hospital-grade equipment, where trained medical professionals manually analyze recorded signals. Although manual evaluation is reliable, it can be time-consuming and susceptible to human error, particularly when processing long-duration recordings or monitoring multiple patients simultaneously. Furthermore, manual interpretation requires skilled expertise, which may not always be available in remote or resource-constrained settings. These limitations highlight the necessity for automated ECG analysis systems capable of providing fast and accurate diagnostic assistance.Recent advancements in embedded systems, digital signal processing, and machine learning have facilitated the development of intelligent biomedical monitoring solutions. Automated ECG analysis systems are capable of processing large volumes of physiological data and identifying abnormal patterns with high accuracy. Machine learning algorithms have shown significant promise in classifying ECG signals by learning complex patterns associated with normal and pathological cardiac conditions. By training models on representative datasets, these systems can support

clinical decision-making and enhance early detection of cardiovascular disorders.

This work presents a hardware–software co-design framework for automated ECG signal classification. The proposed system integrates MATLAB-based preprocessing, FPGA-based memory storage and UART transmission, and Raspberry Pi-based machine learning classification. Initially, ECG signals are preprocessed in MATLAB to remove noise and format the data for hardware implementation. The processed samples are stored in FPGA Block RAM and transmitted using a custom UART communication module. The Raspberry Pi receives the transmitted data, reconstructs the ECG waveform, and applies a Random Forest algorithm to classify the signal as normal or abnormal. The proposed architecture demonstrates the feasibility of combining FPGA hardware acceleration with embedded intelligence for real-time cardiac monitoring and intelligent healthcare applications.

Problem Statement

Modern healthcare applications require efficient and automated techniques for ECG signal transmission and abnormality detection. Continuous cardiac monitoring is crucial for early diagnosis of heart disorders and prevention of severe complications. However, traditional ECG monitoring systems either rely on software-based processing, which may introduce latency, or use expensive clinical equipment that is not suitable for portable or remote deployment. Software-only solutions often suffer from computational overhead when handling large biomedical datasets, while specialized medical devices may be costly and inaccessible in resource-limited environments.

Motivation

The motivation for this research stems from the increasing demand for intelligent and cost-effective healthcare monitoring solutions. Early detection of cardiac abnormalities plays a vital role in preventing severe cardiovascular diseases. Conventional ECG systems rely either on expensive equipment or software-based analysis that may not provide real-time performance. Advances in digital hardware, embedded systems, and machine learning technologies have made it possible to design compact, efficient, and intelligent biomedical monitoring systems. FPGA-based hardware acceleration enables fast and deterministic signal processing, while machine learning algorithms improve classification accuracy and automation. Furthermore, low-cost platforms such as Basys3 FPGA and Raspberry Pi allow the development of portable healthcare solutions that can be deployed in remote or home-based environments. These factors collectively motivate the design of an integrated hardware–software architecture for real-time ECG classification.

Objectives

The primary objective of this work is to design and implement a hardware–software co-design system for real-time ECG signal transmission and classification. The system aims to preprocess ECG signals using MATLAB, convert them into a format suitable for FPGA memory initialization, and store the data in Block RAM using a COE file. A UART transmitter module is implemented in Verilog to enable sequential transmission of ECG samples from the FPGA. Reliable communication between the Basys3 FPGA and Raspberry Pi is established to ensure accurate data transfer. On the receiver side, the Raspberry Pi reconstructs the 16-bit signed ECG samples and applies a Random Forest machine learning algorithm for classification. The system further aims to detect normal and abnormal ECG patterns and validate end-to-end performance, ensuring data integrity and accuracy throughout MATLAB preprocessing, FPGA transmission, and embedded classification stages.

Verilog Hardware Description Language

Verilog is a hardware description language widely used for modeling, simulation, and implementation of digital systems. It was originally developed in the mid-1980s by Gateway Design Automation as a proprietary language for hardware modeling and simulation. The early versions incorporated concepts from existing hardware description languages as well as programming languages such as C, making it easier for engineers to adopt. Over time, the language evolved significantly, particularly after its acquisition by Cadence Design Systems in the early 1990s. To encourage industry-wide adoption, the language documentation was released through Open Verilog International (OVI), which eventually led to its standardization. Since then, Verilog has become one of the most widely used languages for digital design and verification.

Hardware Description Languages (HDLs) are used to describe digital systems at various abstraction levels, particularly at the Register Transfer Level (RTL), where data flow between registers and logical operations are specified. Verilog supports hierarchical design methodologies, enabling designers to represent complex systems using multiple levels of abstraction. It can model digital circuits at behavioral, RTL, gate, and switch levels. This flexibility allows designers to describe both high-level functional behavior and low-level structural details within the same language. Verilog syntax is concise and similar to C, which simplifies learning and implementation for digital system designers.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

Design Methodologies in Verilog

Digital systems designed using Verilog typically follow either bottom-up or top-down methodologies. The bottom-up approach begins with low-level components such as transistors and logic gates, gradually combining them to form higher-level modules and eventually the complete system. Although this method offers detailed control, it becomes inefficient for complex systems. In contrast, the top-down design approach starts with system-level specifications and progressively refines the design into smaller modules. This method allows early testing, improves modularity, and enhances design scalability. In practice, most modern digital systems adopt a hybrid approach that combines both top-down and bottom-up strategies to balance flexibility and efficiency.

Verilog provides several important features that support efficient digital design. It is case-sensitive, allows mixing of abstraction levels, and provides constructs for modeling algorithmic behavior, data flow, and structural connectivity. The language supports simulation, synthesis, and verification within a unified framework. Additionally, Verilog includes predefined keywords, data types, and operators for describing combinational and sequential logic. These capabilities make it suitable for implementing complex digital architectures.

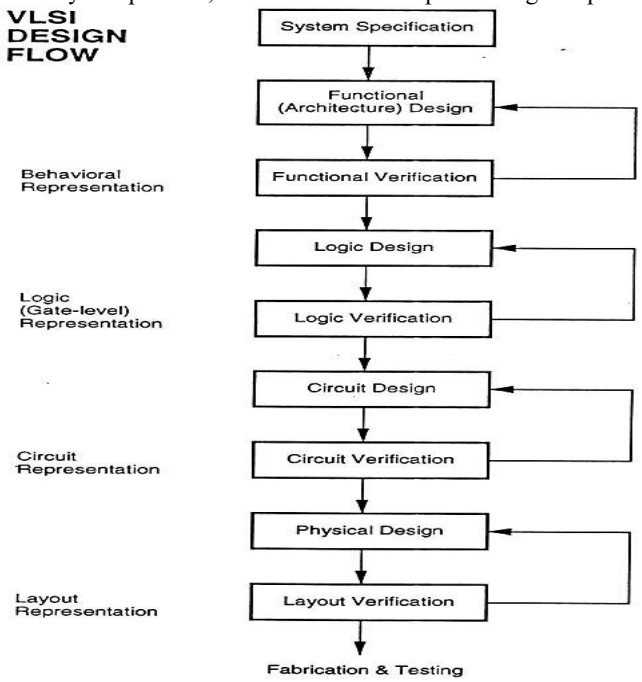

Fig 1 VLSI Design Flow

VLSI Design Flow

The Very Large Scale Integration (VLSI) design process follows a structured sequence of steps that begins with system specification and ends with fabrication and testing. The first stage involves defining system requirements such as functionality, performance, power consumption, and physical constraints. Based on these specifications, the architectural design is developed, where major functional units and data paths are identified. This stage determines parameters such as processing units, memory organization, and interconnection structures.

Following architectural design, behavioral or functional design is carried out, where system operations are described in terms of inputs, outputs, and timing relationships. The design is then translated into Register Transfer Level (RTL) representation during the logic design phase. RTL descriptions, typically written in Verilog, define data transfers, control signals, and arithmetic operations. These descriptions are verified through simulation before proceeding further.

In the circuit design phase, the RTL representation is converted into gate-level implementation while considering timing and power constraints. The resulting netlist is then used for physical design, where geometric layouts of circuit components and interconnections are created. Layout verification ensures that the physical design meets design rules and functional requirements. Finally, fabrication is performed using semiconductor manufacturing processes, followed by packaging and testing to ensure proper functionality of the integrated circuit.

Verilog Modules and Data Representation

The fundamental building block in Verilog is the module, which encapsulates a specific hardware function. Modules communicate with external components through input, output, and bidirectional ports. Each module can be instantiated multiple times to create hierarchical designs. Identifiers in Verilog represent module names, variables, and

instances, and they are case-sensitive. The language supports two primary data types: nets and registers. Nets represent physical connections between components, while registers store values assigned during procedural execution. These constructs enable representation of both combinational and sequential logic circuits.

Verilog also supports multiple abstraction levels. Behavioral modeling describes system functionality using procedural constructs such as always and initial blocks. RTL modeling defines data transfers between registers with explicit clock control. Gate-level modeling represents circuits using logic primitives, while switch-level modeling describes transistor-level behavior. Designers often combine these abstraction levels within a single project to achieve efficient implementation.

Methodology

The proposed ECG classification framework adopts a hardware–software co-design approach to achieve reliable signal transmission and accurate classification. The overall architecture integrates FPGA-based data handling with Raspberry Pi–based machine learning analysis. The workflow is organized into five major stages:

1. ECG signal preprocessing using MATLAB
2. BRAM initialization on FPGA through COE file
3. UART-based ECG data transmission

ECG Signal Conversion in MATLAB

The ECG waveform is initially extracted from a dataset stored in .mat format. The raw floating-point values are processed and converted into signed 16-bit integer representation to ensure compatibility with FPGA memory constraints. Since Block RAM initialization requires formatted memory content, the processed signal is converted into hexadecimal two's complement format and saved as a COE file.

Processing steps include:

- Loading ECG samples from dataset
- Scaling and converting values to signed 16-bit integers
- Converting integer values into hexadecimal format
- Creating COE file with memory_initialization_radix = 16
- Exporting the file for FPGA memory configuration

This stage ensures seamless mapping between MATLAB-generated data and FPGA memory structure.

FPGA BRAM Initialization

The generated COE file is imported into the Block Memory Generator IP within the Xilinx Vivado environment. The BRAM configuration parameters are defined as follows:

- Data width: 16 bits
- Memory depth: determined by number of ECG samples
- Initialization source: COE file generated in MATLAB

Each BRAM address corresponds to a single ECG sample. The memory operates as a static repository, enabling deterministic and sequential data retrieval for transmission.

UART Transmission Module Design

A custom UART transmitter is designed using Verilog HDL to read ECG samples from BRAM and send them serially. The module controls address generation, data capture, and byte-wise transmission.

Transmission sequence:

1. Address counter generates BRAM addresses sequentially
2. Two clock cycles are introduced for synchronous memory access
3. Retrieved data is stored in a temporary register
4. The 16-bit sample is divided into two bytes:

UART configuration:

This configuration provides stable and reliable communication between FPGA and Raspberry Pi.

Raspberry Pi Data Reception

The Raspberry Pi serves as both receiver and classification unit. Python-based serial communication is used to acquire transmitted data.

Reception steps:

- UART interface configured to 9600 baud
- Incoming data read as byte pairs
- Two bytes combined to form 16-bit values

The recovered signal is then forwarded to the classification stage.

Block Diagram

The logical block diagram of the system consists of the following modules:

1. MATLAB Block – Converts ECG signal and generates COE file
2. FPGA BRAM Block – Stores ECG samples
3. UART Transmitter Block – Serial data transmission
4. Raspberry Pi Receiver Block – Data reconstruction
5. Random Forest Block – Signal classification

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

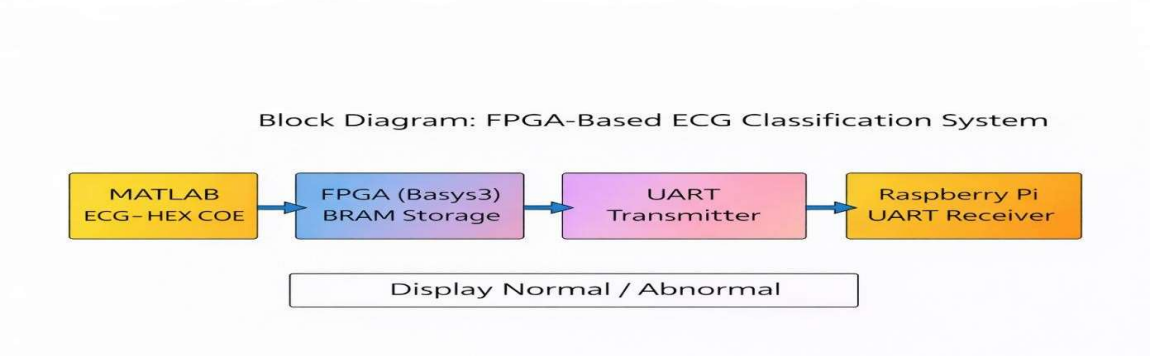

Fig 2 : Block Diagram of FPGA Based ECG Classification System

System Flowchart

The system flow begins with ECG preprocessing in MATLAB, followed by hexadecimal conversion and BRAM initialization on FPGA. The stored samples are transmitted through UART to the Raspberry Pi. The received data is reconstructed and processed using the Random Forest classifier to determine whether the ECG signal is normal or abnormal.

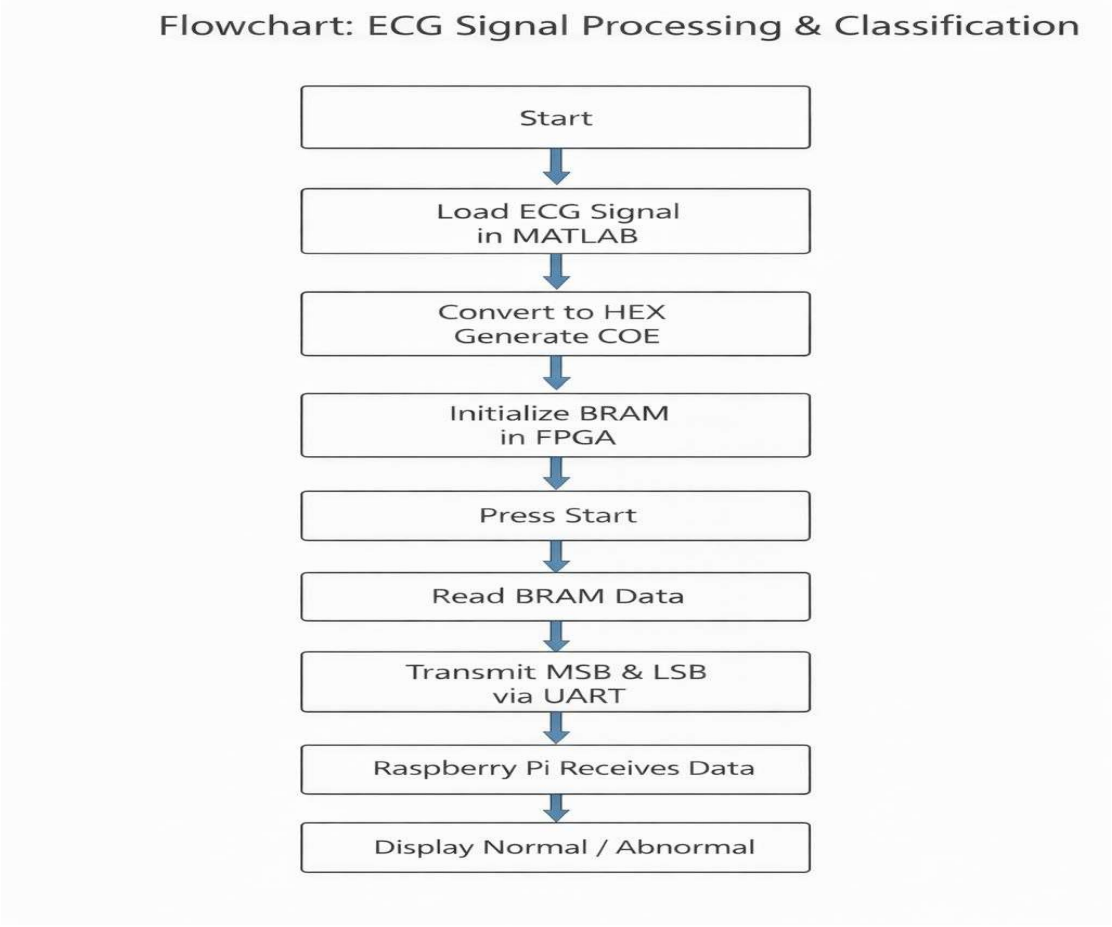

Fig 3 Flowchart-ECG Signal Processing & Classification

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

Results

RTL Schematic

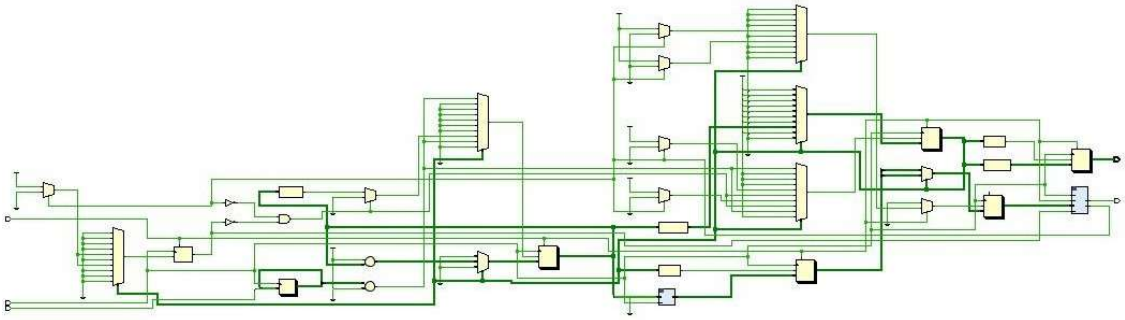

Fig 4 RTL Schematic

The Register Transfer Level (RTL) schematic illustrates the structural implementation of the proposed FPGA-based ECG transmission system. The diagram presents the interconnection of combinational and sequential logic elements used to achieve data storage, control, and communication functionality.

The architecture begins with input control signals that propagate through combinational logic blocks responsible for signal conditioning and decision-making. Logical operators such as AND, OR, and NOT gates generate intermediate control signals required for module coordination.

Multiplexer units are incorporated to select appropriate data paths based on control inputs. These components enable flexible routing of ECG samples between memory and transmission modules. Sequential elements such as registers and flip-flops are employed to store intermediate values and maintain synchronization with the system clock. These registers ensure stable data transfer and prevent timing violations.

The processed signals propagate through interconnected modules and produce the final outputs. Signal paths represented in the schematic demonstrate the data flow between BRAM, control logic, and UART transmitter blocks. The RTL representation assists in verifying the structural correctness of the design before synthesis and implementation.

MATLAB Visualization of ECG Signal and R-Peak Detection

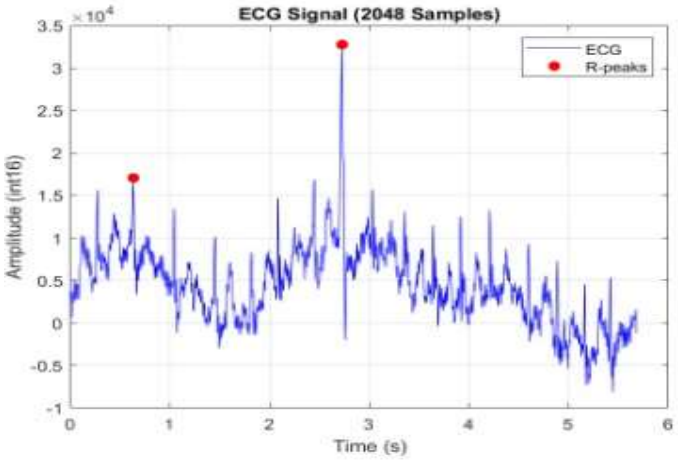

Fig 5 Matlab Visualization

The MATLAB visualization displays 2048 ECG samples with highlighted R-peaks. The R-peaks correspond to prominent spikes in the ECG waveform and are essential for heart rate estimation. The successful identification of these peaks confirms that the preprocessing and peak detection algorithms operate correctly. This validation ensures that the ECG data prepared for FPGA storage retains clinically meaningful characteristics.

Summary Report of Heart Rate and Rhythm Classification

The summary output presents computed metrics such as average heart rate and rhythm regularity. The analysis indicates a mean heart rate of approximately 28.65 bpm, which falls outside the normal physiological range. Based on the extracted features, the signal is categorized as abnormal. This report represents the final diagnostic interpretation derived from signal processing.

Raspberry Pi Real-Time ECG Processing and System Output

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

The Raspberry Pi terminal output demonstrates real-time reception and classification of ECG samples. The Python-based application reads transmitted data, reconstructs the waveform, and applies a pre-trained Random Forest model. The system successfully outputs the classification result as either "Normal" or "Abnormal," confirming correct hardware–software integration.

Simulation Results

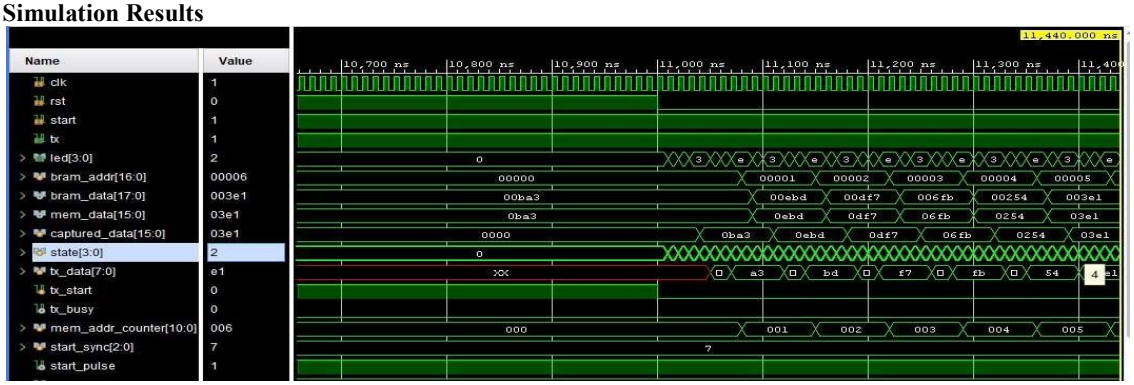

Fig 6 Simulation

The simulation waveform verifies timing behavior and functional correctness of the FPGA design. The clock signal maintains synchronization across modules, while the reset signal initializes the system into a known state. When the start signal is asserted, the Finite State Machine (FSM) controls sequential operations.

The BRAM address counter increments to access stored ECG samples. Retrieved values appear on the memory data lines and are captured into internal registers. These values are then formatted for UART transmission and presented as serial bytes. The transmission status is monitored using the busy signal, ensuring that new data is sent only after the previous frame is completed. The simulation confirms correct sequencing, timing alignment, and reliable UART communication.

Signal Transmission Results

The communication between FPGA and Raspberry Pi was successfully validated. The following observations were recorded:

- ECG samples were transmitted without corruption
- Each 16-bit sample was divided into two 8-bit UART frames
- Raspberry Pi reconstructed signed values correctly
- No data loss occurred during transmission

The reconstructed waveform closely matched the original MATLAB signal, confirming the correctness of BRAM initialization and UART data transfer.

Classification Results

The Random Forest model was trained using labeled ECG datasets. After receiving transmitted data, the Raspberry Pi processed the reconstructed waveform and performed classification. The system correctly categorized ECG signals into normal or abnormal classes. The classification output was displayed on the terminal, demonstrating successful real-time inference.

Conclusion

This work presents a hardware–software co-design approach for ECG signal transmission and classification. ECG samples were processed in MATLAB and stored in FPGA Block RAM using a COE initialization file. A custom UART transmitter enabled reliable communication between the Basys3 FPGA and Raspberry Pi. The Raspberry Pi reconstructed the received data and applied a Random Forest classifier to determine whether the ECG signal was normal or abnormal.

The system achieved accurate data transfer, correct signal reconstruction, and reliable classification results. The proposed architecture demonstrates the feasibility of combining FPGA-based data handling with embedded machine learning for biomedical signal analysis. The modular design supports scalability and can be adapted for real-time health monitoring applications.

Future Scope

Several enhancements can be considered to improve system capability and extend its applications.

Real-Time ECG Acquisition

Future implementations can integrate real-time ECG acquisition modules such as AD8232 to capture live physiological signals instead of using prerecorded datasets.

High-Speed Communication

The UART interface can be replaced with high-speed communication protocols such as SPI, USB, or Ethernet to support higher sampling rates.

Advanced Machine Learning Models

More sophisticated algorithms such as Convolutional Neural Networks, Support Vector Machines, or deep learning architectures can be explored to improve classification accuracy.

Wireless and IoT Integration
Wireless communication technologies including Wi-Fi, Bluetooth, or LoRa can enable remote monitoring and cloud-based healthcare connectivity.

References
[1] J. Pan and W. J. Tompkins, "A real-time QRS detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. BME-32, no. 3, pp. 230–236, Mar. 1985.
[2] P. S. Addison, "Wavelet transforms and the ECG: A review," *Physiological Measurement*, vol. 26, no. 5, pp. R155–R199, Oct. 2005.
[3] V. Sriram and C. Eswaran, "FPGA implementation of real-time ECG signal processing," *International Journal of Electronics and Communications*, vol. 66, no. 12, pp. 1067–1073, 2012.
[4] Y. Zhang, X. Wang, and L. Zhao, "Hardware-efficient real-time ECG monitoring using FPGA," in *Proceedings of the IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2015, pp. 1–4.
[5] A. Kumar, R. Singh, and P. Sharma, "Raspberry Pi-based health monitoring system," *International Journal of Computer Applications*, vol. 180, no. 32, pp. 20–24, 2018.
[6] S. Osowski, L. T. Hoai, and T. Markiewicz, "ECG beat classification using neural network," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 3, pp. 536–540, Mar. 2004.
[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
[8] U. R. Acharya, H. Fujita, V. K. Sudarshan, J. E. Koh, and S. L. Oh, "Automated diagnosis of arrhythmia using ECG signals," *Information Sciences*, vol. 405, pp. 190–198, Sep. 2017.