# A custom reduced instruction set computer-V based architecture for real-time electrocardiogram feature extraction

**Vinayak Vikram Shinde, Sheetal Umesh Bhandari, Deepti Snehal Khurge, Satyashil Dasharath Nagarale, Ujwal Ramesh Shirode**
Department of Electronics and Telecommunication Engineering, Pimpri Chinchwad College of Engineering, Pune, India

**Article Info**

***Article history:***

Received Apr 7, 2024
Revised Apr 14, 2025
Accepted Jun 10, 2025

***Keywords:***

Arithmetic logic unit
Custom architecture
Electrocardiogram control unit
Instruction set architecture
Reduced instruction set computer-V
Register transfer level
R-peak detection

**ABSTRACT**

The growing demand for energy-efficient and real-time biomedical signal processing in wearable devices has necessitated the development of application-specific and reconfigurable embedded hardware architectures. This paper presents the register transfer level (RTL) design and simulation of a custom reduced instruction set computer-V (RISC-V) based hardware architecture tailored for real-time electrocardiogram (ECG) feature extraction, focusing on R-peak detection and heart rate (HR) calculation. The proposed system combines ECG-specific functional blocks including a specialized ECG arithmetic logic unit and a finite state machine-based ECG control unit with a compact 16-bit RISC-V control core. Hardware-optimized algorithms are used to carry out pre-processing activities such high-pass and low-pass filtering as well as feature extraction processes including moving average filtering, derivative calculation, and threshold-based peak identification. Designed to reduce memory footprint and control complexity, a custom instruction set architecture supports modular reconfigurability. Functional validation is carried out by Xilinx Vivado simulating RTL components described in very high speed integrated circuit (VHSIC) hardware description language (VHDL). The present work shows successful simulation of important architectural components, complete system-level integration and custom ECG data validation. This work provides the basis for an application-specific, reconfigurable, power efficient hardware solution for embedded health-monitoring devices.

***Corresponding Author:***

Vinayak Vikram Shinde
Department of Electronics and Telecommunication Engineering
Pimpri Chicnhwad College of Engineering
Akurdi, Pune, India
Email: shindevinayak192@gmail.com

## 1. INTRODUCTION

Wearable health-monitoring systems have become increasingly important in modern healthcare, enabling real-time tracking of vital signs and early detection of medical conditions. Among these, electrocardiogram (ECG) monitoring is essential for both identifying and treating cardiovascular conditions. Wearable technology, however, requires compact, reliable, small and energy-efficient solutions to guarantee continuous use without regular charging. Technologies that operate on ECG readings in real-time while consuming little power and achieving high accuracy are in high demand. With an emphasis on wearable health-monitoring applications this paper aims to design a power-efficient custom hardware architecture that is optimized for extracting ECG characteristics. The architecture, designed at the register transfer level (RTL) using very high speed integrated circuit (VHSIC) hardware description language (VHDL), includes a reduced instruction set computer-V (RISC-V) based control core and dedicated hardware blocks for identification of key ECG features such as R-peaks and heart-rate.

The need for effective wearable health monitoring systems and the increasing incidence of cardiovascular diseases (CVDs) are propelling the area of real-time ECG feature extraction forward. One important area of research is making application-specific integrated circuits (ASICs) that are designed to handle ECG signals with little power. One method is to use ultra-low power ASICs with peak detection capabilities to diagnose CVDs early on [1]. These kinds of designs are very good at finding P, R, and T peaks in the ECG signal, which is important for diagnosing CVD, while using as little power as possible by using methods like self-adjustable discharge rates and sub-threshold region operation. Similarly, efforts have been aimed towards ASIC-based ECG processors that can predict cardiac arrhythmia [2]. These designs focus on taking out important ECG features and using classifiers, like Naive Bayes, to predict ventricular arrhythmia up to several hours in advance. This shows that proactive healthcare measures are possible. The use of artificial neural networks (ANNs) in ECG processor ASICs to find arrhythmias is also noteworthy [3]. This shows that ANN architectures and optimized R-peak detection can provide high classification accuracy with low power consumption, making them ideal for low-cost ECG monitoring. Furthermore, the development of wearable health systems needs improved hardware solutions. A study of low-power ECG signal processor designs shows important design factors and advantages of ASIC designs based on 90 nm technology [4].

Others have focused on getting heart rate (HR) information from ECG signals, using fuzzy logic-based algorithms to get very low power use [5]. This significantly reduces its power requirements while providing sufficient utility. Other methods of detecting cardiovascular diseases include developing ultra-low power ASICs that utilize forward search-based diagnostic algorithms to evaluate ECG signals in real time while consuming as little power as possible [6]. The efficiency of the algorithms is still important. Real-time QRS detection techniques use digital assessments of slope, amplitude, and width to reduce interference and improve QRS complex identification sensitivity [7].

RISC processors, particularly RISC-V, are receiving a lot of interest in embedded systems due to their flexibility and open architecture. This is in addition to advancements in ASICs. There has been discussion regarding how RISC-V microprocessors can be used in education, with a particular focus on pipeline architectures that can be used in both education and field programmable gate arrays (FPGAs) and ASICs [8]. Some methods use the wavelet transform to improve the energy efficiency of ECG recording and R-peak detection [9]. It has a high CR of 10.3 and a percentage root-mean-square difference of 0.64% when it is recording. It also has a 99.72% sensitivity for R-peak recognition and a 99.49% positive prediction rate when the data is shrunk by 13.68 times. A greater amount of work is being carried out on methods for designing RISC-V processors, which will make it simpler to make distinct types of processors [10]. Microarchitecture design and study of RISC-V instruction set architecture (ISA)-compatible processors are also looked into, especially how instruction sets affect the performance of the pipeline [11]. Verilog can also be used to build a single cycle RISC-V processor [12]. Some designs use 32-bit processors to make devices that are economical [13]. A 32-bit RISC-V based system, spanning RTL to graphic database system II (GDSII), is further designed and implemented using this method for medical applications [14].

A complete study of ECG signal reveals, together with methodologies for improved accuracy [15], how crucial automated methods and R-peak recognition are. FPGAs also allow for the construction of signal processing circuits [16]. This system has FPGA, an ECG output can help one determine the HR [17]. Chips that are highly integrated and use little power have been made for handling ECG signals in wearable tech, mostly healthcare devices [18], [19]. One study in which an R-peak identification algorithm was created and modeled using VHDL in Xilinx ISE 14.6 shows a significant method: using ECG signal analysis to FPGAs. Using the MIT-BIH Arrhythmia ECG database, the system was confirmed to consistently extract features for beat-to-beat interval identification [20]. A low-complexity feature extraction approach designed for power and resource efficiency was presented in a different study. To avoid floating-point operations, the architecture used basic computational components including comparators, shifters, and adders, which greatly lowered power consumption with just 738 pJ at 1 MHz and 198 V. Synthesized with 180 nm complementary metal oxide semiconductor (CMOS) technology, the Virtex-7 FPGA was used to construct the design, which revealed energy reductions of up to 128 times over traditional techniques [21].

FPGA-based systems have also been investigated for real-time and parallel processing of ECG signals. A suggested system was tested as a system-on-chip (SoC) design and retrieved ECG characteristics. Using the MIT-BIH Arrhythmia database, the new technique was verified against MATLAB routines, showing an overall detection tolerance of 0.01 seconds and stressing the viability of parallel processing for precise real-time analysis [22]. Low-complexity and power-efficient algorithms are absolutely necessary given the growing need for wearable ECG monitoring. One study suggested an ASIC implementation-optimized R-peak detection technique. For noise reduction it used digital and envelope filters; then the Hilbert transform detected R-peak zero-crossings. When evaluated on the MIT-BIH Arrhythmia database [23], the method attained a high detection accuracy of 99.81% and sensitivity of

99.87%. A low-power ECG signal processing method was likewise created with a RISC-V CPU to lower pacemaker power use. Built on a 90 nm technology node, the suggested RISC-V design used just 4.053223 microwatts, far less than conventional pacemakers which usually use between 60 and 100 microwatts [24].

Several algorithms have been created to identify R-peaks under real-time restrictions and computing efficiency for low-cost and portable ECG monitoring systems. One study created an efficient system for embedded processing evaluated using ECG waveform datasets. Demonstrating the feasibility of low-cost solutions for ubiquitous healthcare [25], the algorithm was effectively used in a smartphone-based ECG acquisition system. Apart from R-peak identification, predictive analytics and ECG signal classification have become more popular. Using a Naïve Bayes classifier, a completely integrated ECG signal processor (ESP) was created for ventricular arrhythmia prediction. Using MIT PhysioNet and the American Heart Association databases, the ESP validated P-QRS-T characteristics. Occupying only 0.112 mm$^2$ and utilizing 2.78 µW at 10 kHz, the architecture created using a 65 nm CMOS process revealed the practicality of ASIC-based predictive ECG analysis [26].

Among upcoming innovations in ECG signal processing are ultra-low-power feature extraction techniques employing a wavelet transform-based approach for real-time QRS complex identification. When assessed on benchmark ECG datasets [27], this technique, which was implemented in a 40 nm CMOS technology, maintained an accuracy of 99.4% and used as little as 1.2 µW. Using convolutional neural networks (CNNs), another research presented a deep learning-based ECG classification model that could distinguish between normal and pathological cardiac rhythms. Examined on a big dataset, the model showed 98.76% classification accuracy, hence proving the promise of AI-driven ECG diagnostics [28]. A multimodal technique combining ECG with photoplethysmography (PPG) for improved HR variability analysis was studied in another study. With an F1-score of 95.3% [29], this combination of bio signals enabled better accuracy in arrhythmia identification, especially in atrial fibrillation situations. At last, scientists have looked at methods of energy harvesting to run portable ECG equipment. Paving the door for self-sustaining health monitoring systems, a study showed an ECG sensor driven by a bioenergy harvesting circuit employing thermoelectric generators guaranteeing continuous operation without external batteries [30].

In conclusion, the research that has already been done shows that there are many different ways to extract features from an ECG. These include custom ASIC designs and flexible RISC-V implementations. This paper builds on these findings to present a custom RISC-V-based architecture made for extracting features from real-time ECGs. The goal is to use the best parts of both custom hardware and an open-source, flexible processor core.

## 2. METHOD

The design flow of custom RISC-V based architecture involves a systematic approach, consisting of requirement analysis, architecture design, algorithm development, RTL design and simulation and functional validation:

a. Requirements analysis:

- Application: processing ECG signals in real-time for use in wearable devices.
- Power consumption: many small wearable devices rely on coin cell batteries which generally provide a capacity of approximately 200 mAh and lithium-ion batteries with similar capacity, so a power budget of 50 mW to 100 mW is sufficient for extended battery life. This range is consistent with the power consumption of existing wearable devices and can be achieved through a combination of low-power design techniques, circuit optimisations, algorithm optimisations and careful selection of components.
- Real-time performance: the architecture is designed to process ECG signals as shown in Figure 1 in real-time for R-peak detection and HR calculation from digitized ECG signals. For the functional simulations, digitized ECG data was generated using python to emulate the output of an analog to digital converter (ADC) at a sampling rate of 200 Hz-500 Hz. This range is selected for the ADC as per the Nyquist-Shannon sampling theorem and the frequency content of the ECG signals. Clinically relevant ECG features are found between 0.05 Hz to 40 Hz range. While a lower sampling rate would be sufficient, however 200 Hz-500 Hz provides a safety margin and reduces anti-aliasing imperfections.
- Feature extraction: the system is designed to identify and extract key features from pre-processed ECG signal. The focus is on R-peak detection and hear-rate calculation due to their fundamental importance as indicators of cardiac function and their usability in various health monitoring scenarios.

Figure 1 represents an ECG waveform that represents the electrical activity of the heart and includes main components such as P-wave, QRS complex, T-wave, and PR/QT intervals. The P wave demonstrates atrial depolarization while QRS complex demonstrates ventricular depolarization necessary for heart contraction. The T wave shows ventricular repolarization and the ST segment represents hearts recovery phase.

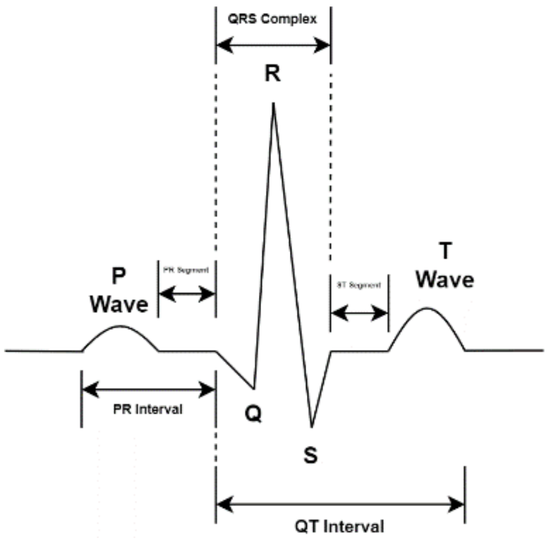

Figure 1. ECG signal waveform

b. Architecture design:

- Input module: for functional simulation purpose, the input module includes a behavioral model of a 12-bit successive approximation register (SAR) ADC. In a hardware implementation, a SAR ADC architecture would be selected due its power efficiency, good speed and reduced area making it well-suitable for wearable applications. In this paper, a behavioral model is used to generate the output of what a 12-bit ADC have. A 12-bit resolution provides sufficient precision for ECG signal analysis, capturing the dynamic range of the signal without excessive overhead. The output of the model has a sampling rate of 200 Hz to 500 Hz guided by the Nyquist-Shannon sampling theorem.
- ECG control unit (ECU): this unit is implemented as a finite state machine (FSM) to control the ECG processing stages. The FSM is responsible for high-pass filtering, low-pass filtering, R-peak detection and heart-rate calculation. The FSM design has states corresponding to each signal processing operation. Each state is responsible for a specific hardware operation or algorithm. The ECU interacts with memory unit to pass the required input signals to the ECG arithmetic logic unit (ECG_ALU) and also store the outputs of each state.
- Control unit (CU): the CU is implemented as a RISC-V based control core. A RISC-V based instruction set architecture is utilized for flexible control over system operation.
- Storage: the custom architecture features memory design intended for on-chip implementation. The function of the memory unit or memory is store data and instructions that the CU needs to access and process. The memory architecture is Von-Neumann based where program instructions and data are stored in single memory. The memory is internally bifurcated into random access memory (RAM) and read only memory (ROM) as per the usage. The size of the memory is 4 KB with 2 KB of each RAM and ROM.
- Output module: the architecture features a general-purpose input/output (GPIO) module. Its role is to communicate with external devices such as LCD displays. In a hardware implementation the GPIO interface would be utilised for transmitting heart-rate and other helpful data for real-time monitoring. In the RTL design, it simply maps a 16-bit input directly to a 16-bit output, providing a basic interface for data output during simulation.

c. Algorithm development:

Pre-processing: a first stage of processing on the digitised ECG samples. This stage includes noise reduction and baseline correction. This stage comprises of the following:

- High-pass filtering: a first-order infinite impulse response (IIR) high-pass filter is implemented to remove baseline wander, a slow drift in ECG signal. This configuration attenuates low-frequency components (like baseline wander) and passes high frequency components. The mathematical equation is discussed in the section 5.
- Low-pass filtering: a first order IIR low-pass filter is implemented to eliminate high-frequency noise. This configuration attenuates high frequency components (like noise) and passes low-frequency components. The mathematical equation is discussed in the section 5.

Feature extraction: this stage is responsible for core feature extraction process such as R-peak detection and heart-rate calculation.

– R-peak detection: a robust R-peak detection algorithm is implemented combining moving average filter, with derivative calculation and thresholding.
– Moving average filtering: this smoothens the signal by averaging it within a moving window in order to reduce the noise for derivative calculation.
– Derivative calculation: the slope of the signal is calculated using the equation specified in section 5. This is used to detect the sharp changes in the signal.
– Thresholding: detects R-peaks from the signal, and compares them with the pre-defined or reference values and declares them as R-peaks.
– HR calculation: heart-rate is calculated form the detected R-peaks by measuring the inter-beat intervals (IIB). The HR can be calculated with equation from section 5.

d. RTL design and simulation

The proposed architecture, shown in Figure 2, is designed at the RTL using VHDL, a widely used hardware description language. The design methodology involves a top module as well as individual block design approaches. Xilinx Vivado tool is used for simulation. Simulation helps in verifying the functionality of the design. Verification strategy was designed to test and check the individual modules along with the whole system. The testbenches were designed to include generating stimulus and testing for different scenarios.

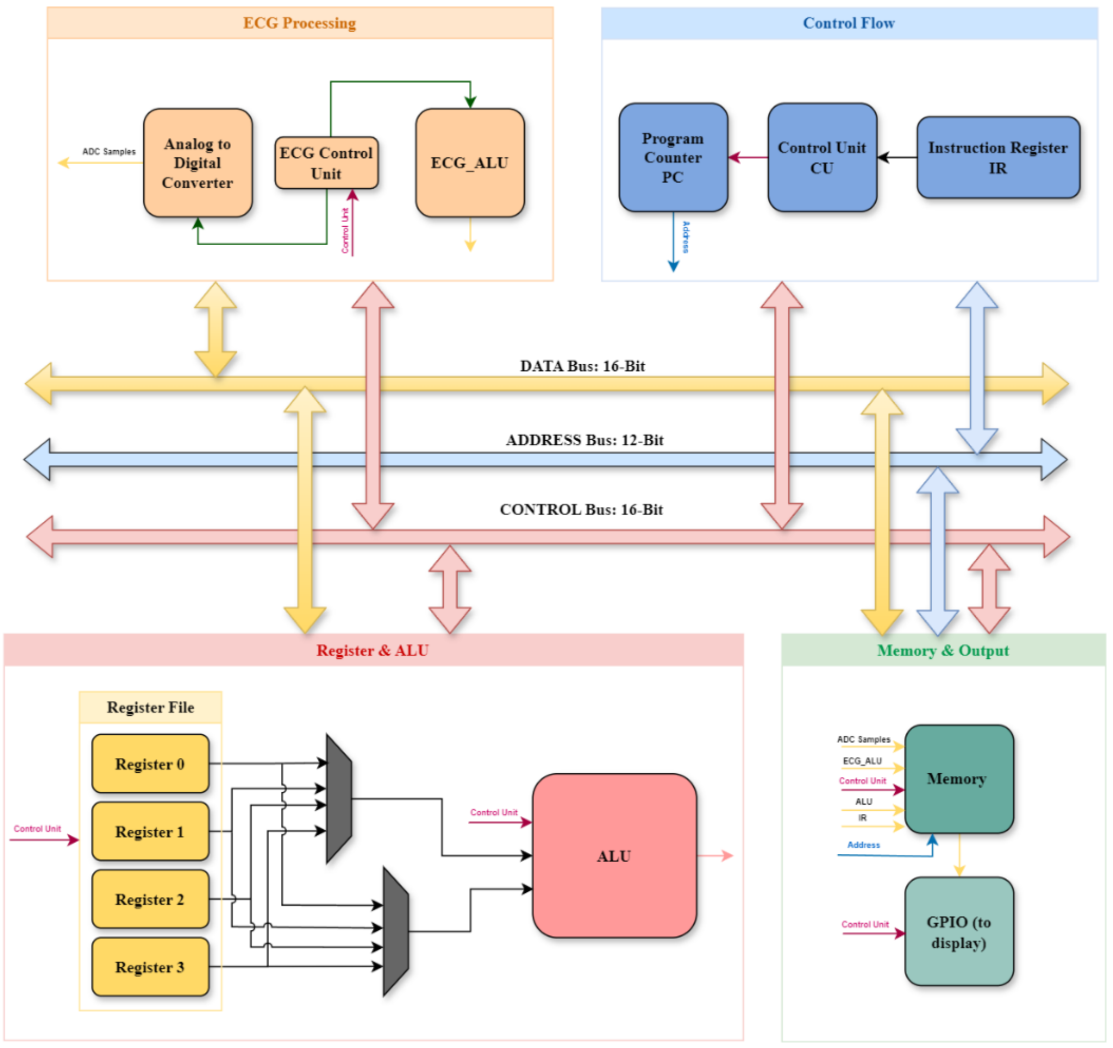

Figure 2. Custom RISC-V based architecture for ECG processing

Figure 2 represents a custom architecture for ECG signal processing with all the major components such as CU, arithmetic logic unit (ALU), memory unit, program counter (PC), ECU, and GPIO. The operations of the architecture are controlled by the CU. The CU fetches the instruction from the memory using PC and stores it in the instruction register (IR). The instruction is decoded and control signals are generated directing the flow of data between the various components. The CU sends the ECG operation signal to the ECU to proceed with ECG specific operations. The ECU fetches samples from the modelled ADC and feeds it to the ECG_ALU for processing. The final result is stored in the memory and the data is output using the GPIO. The colouring scheme used for signals in the architecture is pink for the control signals, yellow for data and blue for address signals.

## 3. IMPLEMENTATION

The custom architecture contains the following components:

### 3.1. Registers (R0-R3)

Registers play a key role in this architecture as they are used as temporary storage for operands and intermediate results. In this architecture four general purpose registers are used: R0, R1, R2, and R3. Where R3 is a return address register (RAR).

a. Return address register:
- It temporarily stores the return address during a jump or subroutine call.
- When a CALL instruction is executed the current memory address of the PC is stored in the RAR.
- When a RET instruction is executed the PC recovers the address stored in the RAR and resumes operation from that address.

b. Function:
- Used as a source and destination for ALU operations.
- Used to store data temporarily during memory transfers.

### 3.2. Memory

The memory unit is an important part of the architecture, providing storage space for both program instructions and data required for ECG signal processing. Single memory concept is used in the design due to its simplicity and reduced hardware complexity. It is internally divided into two types: data memory and program memory. The purpose of the memory unit is to store the following:

Program instructions: the instructions that control the operation of RISC-V core and ECU are stored in memory. These instructions define the sequence of operations of RISC-V based ISA and ECG operation instruction.

- ECG data samples: incoming ECG samples from the ADC are stored in the memory for processing. This includes both the raw digitized ECG samples and intermediate ECG processes results such as filtering and feature extraction.
- Filter coefficients: the coefficients used by digital filters (high-pass and low-pass) are stored in memory.
- Variables and temporary storage: the memory unit also provides storage space for variables, temporary results, immediates and stack space for function calls.

Data memory (RAM): it is of 2 KB in size. It is used to store intermediate results, ADC samples and provide stack space for temporary storage such as buffers and stack as depicted in the Table 1. Table 1 represents the structure of the data memory. It shows the address locations in memory along with their size in decimal for various applications such as general-purpose, first-in-first-out (FIFO) buffer and other buffers and stack.

Table 1. Data memory structure

| Address range | Purpose |
|---|---|
| 0x0800–0x0EFF (2048 to 3839) | General purpose data storage (intermediate calculations). |
| 0x0F00–0x0F0F (3840 to 3855) | FIFO buffer for ADC samples. |
| 0x0F10 0x0F5F (3856 to 3935) | Buffers and stack. |

Program memory (ROM): It is of 2 KB in size represented in Table 2. It is used to store instructions for sampling, filtering, R-peak detection and HR calculations. Fixed constants or values are also stored. Table 2 represents the structure of the program memory. It shows the address locations in memory along with their size in decimal for storing instructions and constants. Memory mapping: memory mapping is used to access memory patterns and reduce access time.

Table 2. Program memory structure

| Address range | Purpose |
|---|---|
| 0x0000–0x07FF (0 to 2047) | Stores instructions and constants |

### 3.3. Program counter

The PC is a fundamental register in the proposed architecture for processing ECG signals. Its main job is to maintain the address of the next instruction that needs to be fetched and executed. It acts a pointer, guiding the CU through the program code stored in memory. Without program counter CU will have no idea where to look for the next instruction and the program execution would fail. PC performs following tasks in the design:

#### 3.3.1. Instruction sequencing

The program counter makes sure that the instructions are run one after the other and in the right order. Anytime an instruction is read from the memory address that the PC points to, the PC is incremented by one to point to the next instruction in the series.

#### 3.3.2. Sequential execution

The PC is increased after each instruction read in simplest case. Because of this the instructions are executed in the order they appear in the memory, and this is called as sequential processing.

#### 3.3.3. Branching and jumping

The PC can run program instructions that are not in the order by using branching and jumping instructions:

- Branch instructions: these instructions change the flow of execution depending on the outcome of a test or comparison. When the condition is met, a new address is put into the PC. This makes the program go to different part of the code.
- Jump instructions: in jump instructions the PC is loaded with new address that changes the flow of execution without any conditions.

#### 3.3.4. Subroutine calls and returns

The PC is very important for subroutine calls and returns:

- Subroutine call: when a function or subroutine is called, the return address is generally pushed to the top of the stack and the address of the first instruction in the subroutine is loaded into the PC.
- Subroutine return: when the subroutine is done, the return address is popped from the stack and is loaded into the PC. This starts the program execution from the address when the subroutine was called.

### 3.4. Instruction register

The IR is an important component within the system design. It acts as a temporary storage area for the instruction that is currently being carried out. It’s like a buffer that holds the instruction while the CU decodes it and generates the required control signals to perform the operation. The data stored in the IR guides the design on the particular order of instructions to perform. The instruction register is implemented as a set of flipflops, the width of IR is same as of the instruction word in the ISA that is 16-bits, and the output of the IR is connected to the CU that generates the appropriate control signals.

### 3.5. Arithmetic and logic unit

The ALU performs all the mathematical and logical operations specified in the instructions. It can do simple math tasks like adding, subtracting, multiplying, and dividing. Logic functions like AND, OR, XOR, and NOT along with bitwise shift functions, such as logical shift left (SLL) and logical shift right (SRL) are supported.

### 3.6. Electrocardiogram control unit

The ECU is dedicated control block in the custom architecture, designed specifically for coordinating and managing the sequence of operations on real-time ECG signals. It acts as a traffic controller for ECG processing, ensuring that the data flows in correctly between the required modules and the necessary operations are performed in the sequential order. While the main CU handles overall system management, the ECU focuses only on ECG-specific tasks as represented in Figure 3 main functionalities of ECU are:

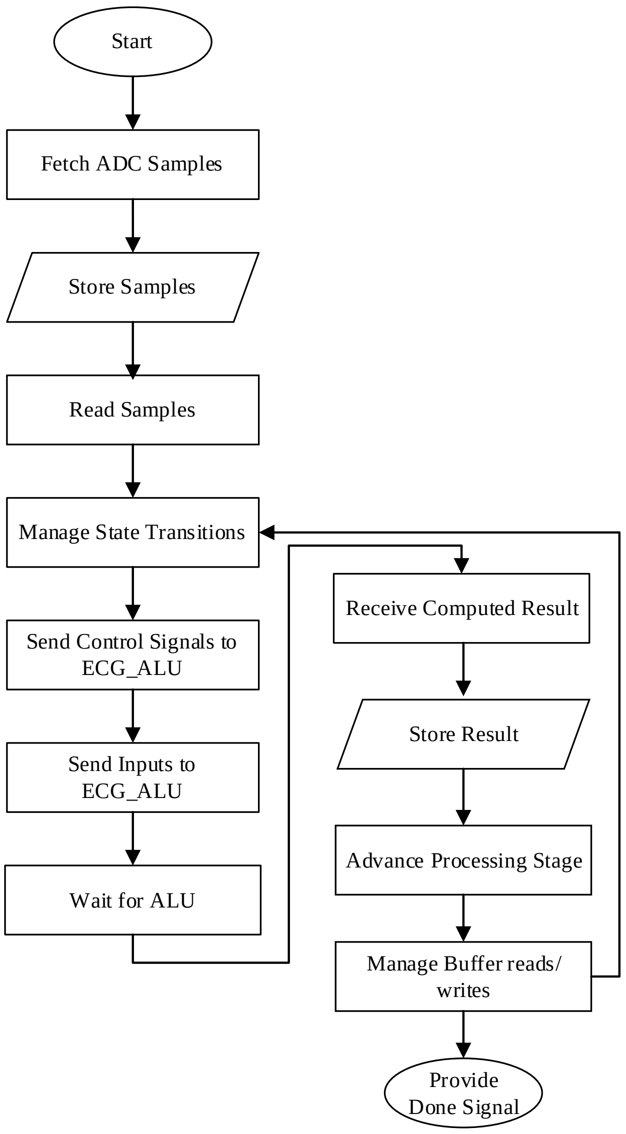

Figure 3. Flowchart depicting the ECG processing methodology

#### 3.6.1. State management

An FSM is used to build the ECU. The FSM specifies a set of states that are related to different steps in ECG processing flow, such as getting ADC data, high-pass filtering, low-pass filtering, R-peak detection, and heart-rate calculation. The FSM changes states when certain conditions are met, like completion of data gathering from ADC, or R-peak is detected. In each state it sends right control signals to the associated modules to turn them on.

#### 3.6.2. Control of data collection

The ECU is responsible for getting ECG data samples from the ADC. It generates control signals that store the ECG samples in the FIFO locations of the memory.

#### 3.6.3. Handle filtering

The ECU handles the filtering stages by:

- Reading filter coefficients from memory.
- By giving right control signals to ECG ALU to carry out high-pass and low-pass filtering operations.
- Controlling the flow of data between memory and ECG_ALU during the filtering process.

#### 3.6.4. Control feature extraction

The ECU handles feature extraction processes by:

- Sending control signals to ECG_ALU to perform required calculations to detect R-peak.
- Bringing up the thresholding logic to detect R-peaks.
- Utilizing the identified R-peaks to calculate IBI.

#### 3.6.5. Heart-rate calculation

The ECU executes the heart-rate calculation stage by:

- Using the equation defined in section 5 to calculate hear-rate.
- Storing the heart-rate in memory.

#### 3.6.6. Handling ECG_ALU

The ECU tells the ECG_ALU what arithmetic and logic operations are to be performed at each stage of ECG signal processing. This includes telling the ECG_ALU what kind of operation to do, what operands to use, and where the results should be stored.

Figure 3 demonstrates the working of an ECU with the help of a flowchart. First the ECU is initialized by the start signal from CU. The ECU begins operation by fetching the samples from ADC and storing them in the respective FIFO locations. After storing the ECU initializes ECG_ALU and reads the stored samples. The ECU sends required control signals and the read samples as input to the ECG_ALU according to state management. It waits for the ECG_ALU to compute the result and stores the computed intermediate result and proceeds to the next stage with buffer management. Finally, the system sends 'Done' signal, representing completion of the ECG processing cycle.

### 3.7. Electrocardiogram arithmetic logic unit

This is a key component of the proposed system design. The ECG_ALU is specialized for ECG tasks, such as executing high-pass or low-pass filtering, R-peak detection, and HR calculation. It performs these operations according to the control signals and inputs provided by the ECU for each state. It helps in reducing the load on ALU as it handles the ECG specific operations. This helps in reducing complexity and increasing efficiency and also reducing power consumption.

### 3.8. Control unit

The main part of the custom architecture is the CU. Its function is to decode instructions read from ROM, figure out which operations need to be performed and send control signals to all the connected blocks in the architecture to make sure they all work together.

- The CU uses the PC to get instructions from the ROM.
- Decodes these instructions as per the ISA.
- Generates the control signals for the operation of other parts, such as memory, ALU, and the GPIO.
- The CU maintains the sequence of program implementation and performs branching or subroutine call instructions.

## 4. SYSTEM ARCHITECTURE

The system architecture comprises of the following:

### 4.1. Instruction set architecture

The designed custom RISC-V based architecture uses a custom RISC-V ISA, as shown in Table 3, modified to the specific requirements of the system. The RISC-V was chosen due to its open source nature, modularity and potential for customisation allowing for an efficient instruction set for the application. Merits of the ISA are:

- Base ISA: the base ISA is RISC-V, providing a source of essential instructions for arithmetic operations, logical operations, memory access and control flow.
- Instruction format: the instruction format uses 16-bit instructions, consisting of 4-bits opcode and 12-bits operand. This short instruction format cuts down on code size and memory access load, which reduces power consumption. The use of 16-bit format provides good amount of flexibility.
- Custom instructions: one important feature of the ISA is that it has a custom ECG_OP instruction that aims to start specific ECG processing tasks within the ECU and ECG_ALU.
- Opcode encoding: the action to be performed is determined by the opcode field.

Table 3 represents the instruction set with basic instructions such as arithmetic, logical, memory access, control flow, and shift instructions. It also includes custom instruction like ECG_OP.

Table 3. Instruction set architecture

| Opcode (4 bits) | Mnemonic | Description | Operand field (12 bits) |
|---|---|---|---|
| 0000 | NOP | No operation (useful for delays, alignment) | - |
| 0001 | ADD | R1 ← R1 + R2 | R1(2 bits), R2(2 bits), Unused (8 bits) |
| 0010 | SUB | R1 ← R1 - R2 | R1(2 bits), R2(2 bits), Unused (8 bits) |
| 0011 | MUL | R1 ← R1 * R2 | R1(2 bits), R2(2 bits), Unused (8 bits) |
| 0100 | DIV | R1 ← R1 / R2 | R1(2 bits), R2(2 bits), Unused (8 bits) |
| 0101 | MOV | R1 ← R2 | R1(2 bits), R2(2 bits), Unused (8 bits) |
| 0110 | LOAD | R1 ← Memory[addr] | R1(2 bits), Address (10 bits) |
| 0111 | STORE | Memory[addr] ← R1 | R1(2 bits), Address (10 bits) |
| 1000 | AND | R1 ← R1 AND R2 | R1(2 bits), R2(2 bits), Unused (8 bits) |
| 1001 | OR | R1 ← R1 OR R2 | R1(2 bits), R2(2 bits), Unused (8 bits) |
| 1010 | XOR | R1 ← R1 XOR R2 | R1(2 bits), R2(2 bits), Unused (8 bits) |
| 1011 | SHIFT | R1 ← R1 Shift Left/Right by immediate value | R1(2 bits), Direction (1 bit), Shift Amount (5 bits), Unused (4 bits) |
| 1100 | JMP | PC ← addr | Address (12 bits) |
| 1101 | CALL | Push (PC), PC ← addr | Address (12 bits) |
| 1110 | RETURN | PC ← Pop (PC from stack) | - |

### 4.2. Register addressing

The ISA support register based addressing method for accessing operands. The architecture supports four registers, out of which three are general-purpose (R0-R2) and a return address register (R3). As there are four registers only two bits are needed to address them as shown in Table 4. Table 4 shows register addressing with binary address allocated to each register.

Table 4. Register addressing structure

| Register | Binary address | Purpose |
|---|---|---|
| R0 | 00 | General purpose register |
| R1 | 01 | General purpose register |
| R2 | 10 | General purpose register |
| R3 | 11 | Return adddress register for subroutine or functional calls |

### 4.3. Bus architecture

The custom architecture uses 16-bit bus architecture for data bus and control bus and 12-bits for address bus to communicate between various components. The bus architecture comprises of the following:

– Data bus: the 16-bit data bus allows for the transfer of data between the IR, memory unit, ECU, registers, and GPIO.
– Address bus: the 16-bit address bus is used to define memory address for the memory access operations or tasks.
– Control bus: the control bus carries control signals from the main CU that coordinates and manages the operation of different components. These signals include: i) read/write signals for memory access, ii) clock enable signals for enabling or disabling the block, and iii) reset signals for resetting the system or a block.

## 5. MATHEMATICAL ANALYSIS

The design uses following equations to perform ECG specific operations:

a. Pre-processing in the arithmetic logic unit
– High pass filter: removes baseline wander using difference equation.

$$y[n] = x[n] - x[n-1] + \alpha y[n-1] \tag{1}$$

where, x[n] is current input sample; x[n-1] is previous input sample; y[n] is filtered output; y[n-1] is previous filtered output; and α is filter coefficient (α≈0.9).
– Low pass filter: reduces high frequency noise.

$$y[n] = (1-\beta)x[n] + \beta y[n-1] \tag{2}$$

where, β is filter coefficient (β≈0.95).

b. R-peak detection

– Moving average filter: smoothens the signal by averaging it within a moving window.

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n-k] \tag{3}$$

where, N is number of samples.

– Derivative calculation: calculates slope to detect sharp changes in the signal.

$$y'[n] = x[n] - x[n-1] \tag{4}$$

– Thresholding: detects R-peaks by comparing signal values to predefined thresholds.

$$If\ x[n] > Threshold, mark\ as\ R - peak$$

c. Heart rate calculation

Calculate time intervals between successive R-peaks (Δt).

$$HR = 60/\Delta t \tag{5}$$

The (1) makes sure that slow variations are removed while important ECG features are untouched. This (2) indicates gradual signal modifications and noise reduction by combining the new sample x[n] with the earlier output y[n-1]. This (3) helps in detecting R-peaks by eliminating sudden fluctuations. By focusing on changes between successive samples (4), highlights sharp transitions like R-peaks. Final (5) is used to calculate HR by converting beat to beat intervals into beats per minute.

## 6. RESULTS

The top-level RTL schematic aligns with the proposed custom architecture, suggesting its feasibility:

Figure 4 represents a top-level RTL schematic of the proposed architecture with all major components such as CU, ALU, ECU, memory, and registers. The connections and port information of each module is clearly defined.

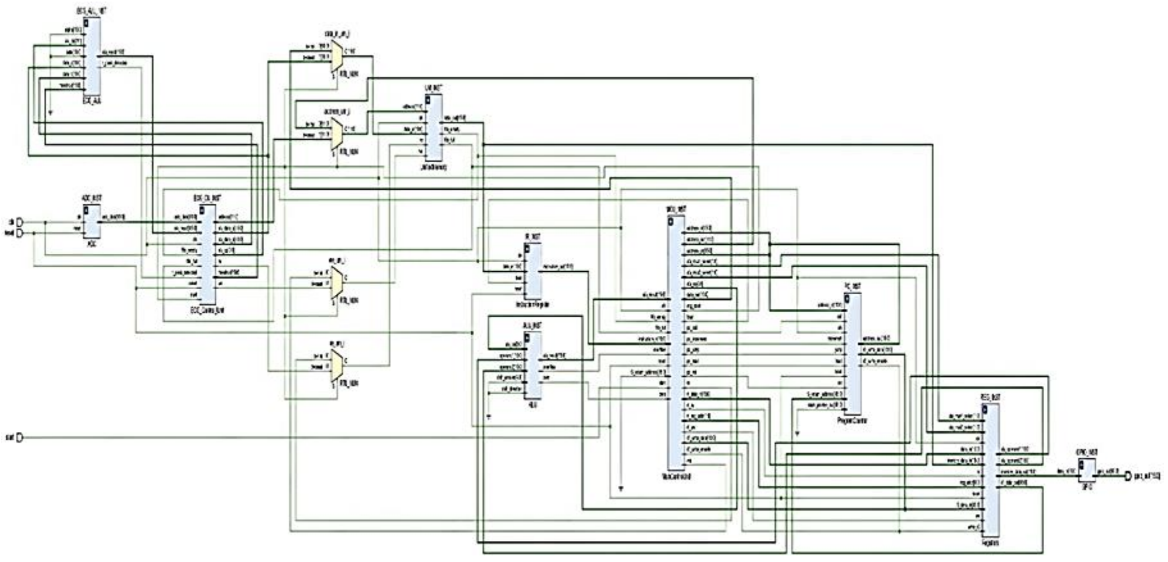

Figure 4. Top-level RTL schematic of the designed custom architecture, showing its hierarchical structure and important modules

Simulation results given in subsections below validates the functionality of the few major blocks:

## 6.1. Register

Figure 5 shows the simulation waveform for the register module. The simulation begins with a reset, making all registers to zero. Register 0 is written with 1234 and register 1 is written with 5,678 when we is high and reg_addr is set. The alu_mux1_select and alu_mux2_select are used to transfer register contents to the alu_operand1 and alu_operand2 output signals.

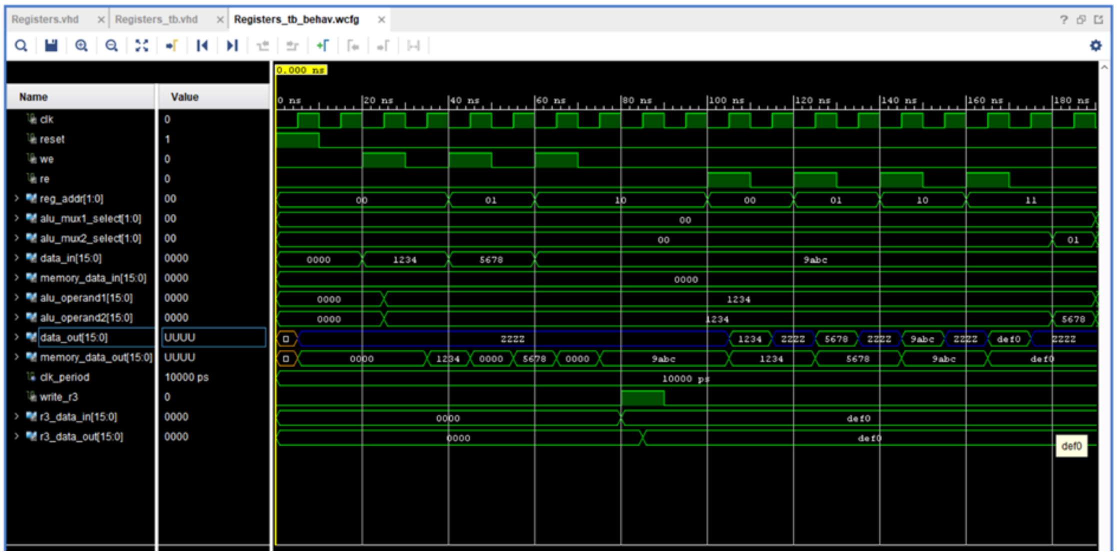

Figure 5. Register simulation waveform

## 6.2. Memory

Figure 6 shows the simulation waveform for the memory block. The memory simulation begins with writing values 0e66, 0f33 to ROM addresses 0 and 1. These are coefficient values that are stored as constants in the ROM. Subsequently, data value 1234 is written to RAM address 800. Additionally, data is written to FIFO addresses with increasing FIFO count along with tracking of empty and full conditions. Various other signals are used that define the size of each component within the memory such as RAM_SIZE, ROM_SIZE, and BUFFER_SIZE.

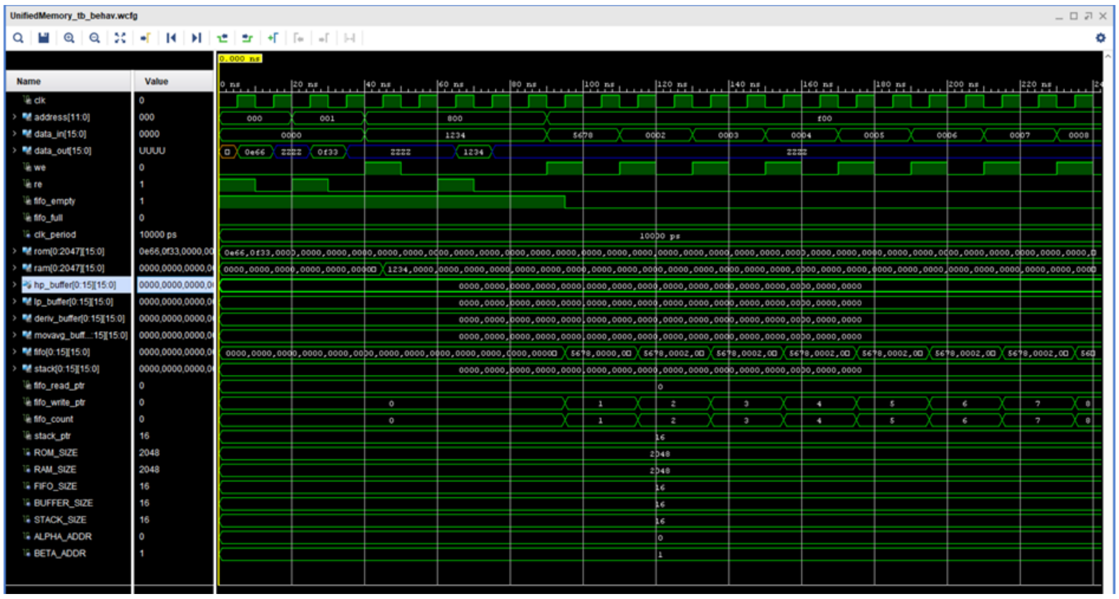

Figure 6. Memory unit simulation waveform

### 6.3. Program counter

Figure 7 represents the simulation waveform for program counter. Initially the program counter is set to 0, and is incremented to 1 when increment signal is high. Subsequently it loads value 4,660 when load signal is high and goes to address 22,136 when call is performed. For ret=1, it returns the address stored in R3 (RAR) that is 39612.

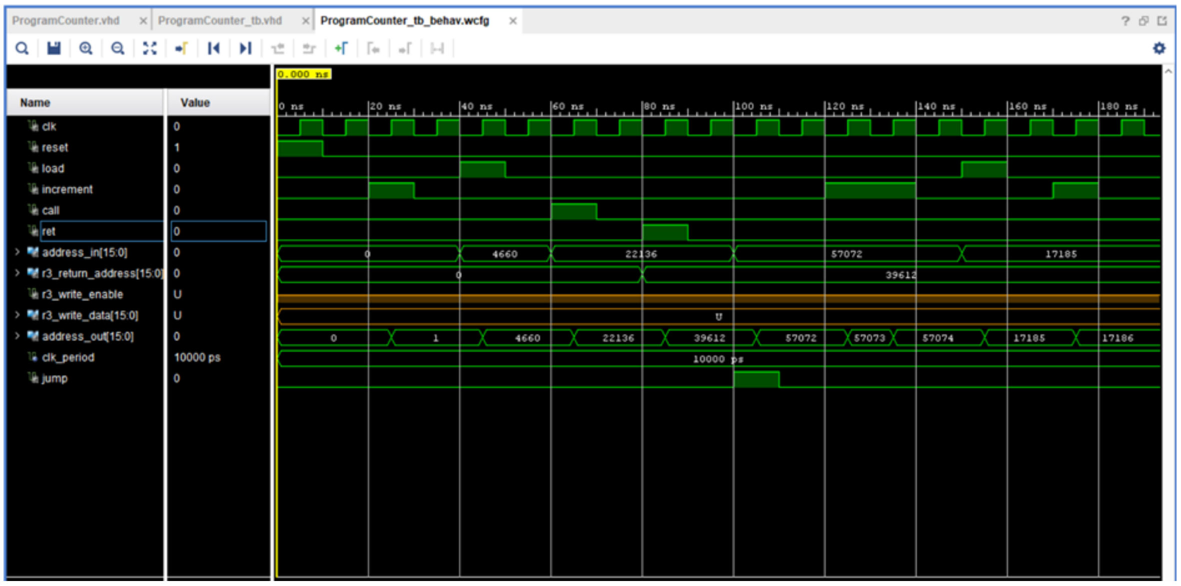

Figure 7. Program counter simulation waveform

### 6.4. Arithmetic logic unit

Figure 8 is a simulation for ALU. The ALU simulation demonstrates various operations, indicated by the alu_op signal, performed on operand1 and operand2. Initially, alu_op is set to 1, demonstrating add operation with operand1 (000a) and operand2 (0005) and result in alu_result (000f) and zero flag set to 0. Subsequently other operations are performed resulting in respective changes to alu_result, overflow and zero.

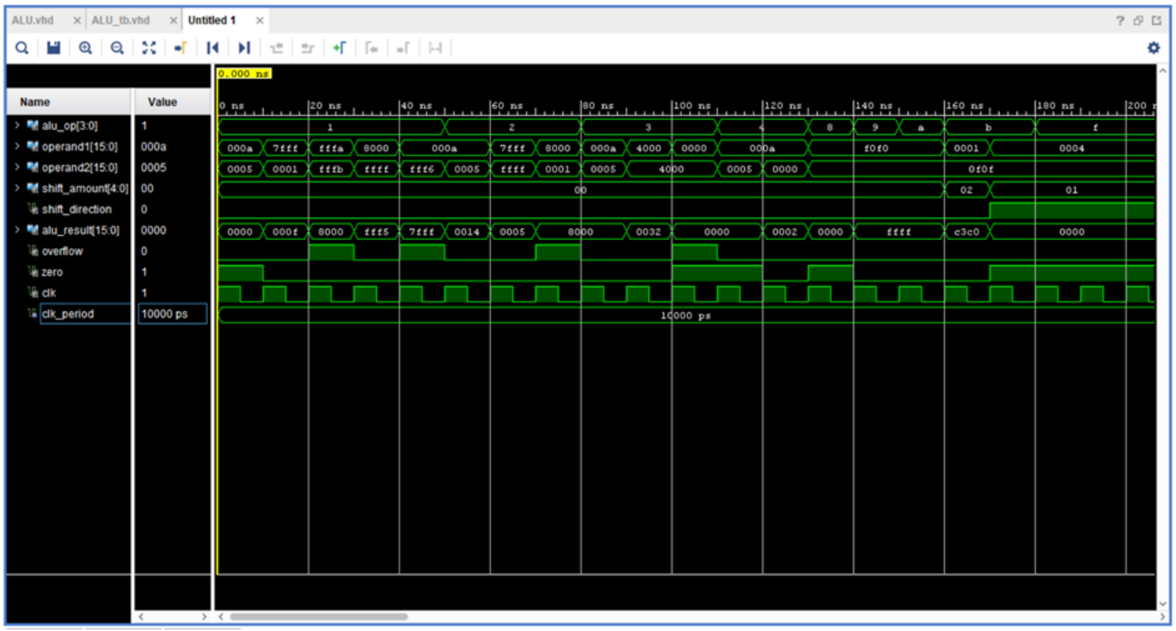

Figure 8. ALU simulation waveform

## 7. CONCLUSION

This paper presents the RTL design and simulation of a custom RISC-V based hardware architecture designed for real-time ECG signal processing in embedded health-monitoring applications. The suggested architecture achieves a balance between computational efficiency and design modularity by integrating a compact 16-bit RISC-V core with dedicated ECG-specific processing units, including an ECG-ALU and an FSM-based ECU. Hardware-optimized implementations of essential pre-processing and feature extraction stages, such as filtering and R-peak detection, enable real-time signal processing tailored to ECG analysis. The introduction of a custom instruction set is developed to support both general purpose operations and ECG tasks, offering flexible control flow while simplifying execution of signal processing routines at the hardware level. Simulation results using Xilinx Vivado confirm the functional correctness of individual RTL components. Although real-world ECG dataset testing remain as future work, the current design establishes a strong foundation for building reconfigurable and application-specific embedded architectures for biomedical devices.

## FUNDING INFORMATION

Authors state no funding involved.

## AUTHOR CONTRIBUTIONS STATEMENT

This work follows the Contributor Roles Taxonomy (CRediT). The specific contributions of each author are detailed below:

| Name of Author | C | M | So | Va | Fo | I | R | D | O | E | Vi | Su | P | Fu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vinayak Vikram Shinde | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |  | ✓ |  |
| Sheetal Umesh Bhandari |  | ✓ |  | ✓ |  | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ |  |  |
| Deepti Snehal Khurge | ✓ | ✓ | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |  |
| Satyashil Dasharath Nagarale | ✓ |  | ✓ | ✓ |  | ✓ |  |  |  | ✓ |  | ✓ |  |  |
| Ujwal Ramesh Shirode |  | ✓ |  |  | ✓ |  | ✓ | ✓ |  |  | ✓ | ✓ | ✓ |  |

| | | |
|---|---|---|
| C : **C**onceptualization | I : **I**nvestigation | Vi : **Vi**sualization |
| M : **M**ethodology | R : **R**esources | Su : **Su**pervision |
| So : **So**ftware | D : **D**ata Curation | P : **P**roject administration |
| Va : **Va**lidation | O : Writing - **O**riginal Draft | Fu : **Fu**nding acquisition |
| Fo : **Fo**rmal analysis | E : Writing - Review & **E**diting | |

## CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

## DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

## REFERENCES

[1] S. Thomas, J. S. Virdi, A. Verma, B. P. Das, K. Okada, and P. N. Singh, "A Peak-detector-based Ultra Low Power ECG ASIC for Early Detection of Cardio-Vascular Diseases," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2024, pp. 1–5, doi: 10.1109/ISCAS58744.2024.10558031.

[2] S. D. Poonguzhali, S. Kanimozhi, R. Karthika, and R. Kirthiga, "ASIC Low-Power ECG-Based Processor for Predicting Cardiac Disease," *International Journal of Engineering Research & Technology (IJERT)*, vol. 5, no. 13, 2017.

[3] C. Zhang *et al.*, "A Low-Power ECG Processor ASIC Based on an Artificial Neural Network for Arrhythmia Detection," *Applied Sciences*, vol. 13, no. 17, Art. no. 9591, 2023, doi: 10.3390/app13179591.

[4] K. Raja, S. Saravanan, R. Anitha, S. S. Priya, and R. Subhashini, "Design of a Low Power ECG Signal Processor for Wearable Health System—Review and Implementation Issues," in *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, 2017, pp. 383–387, doi: 10.1109/ISCO.2017.7856022.

[5] Y. Matsumoto, T. Tanaka, K. Sonoda, K. Kanda, T. Fujita, and K. Maenaka, "Low Power ECG Processing ASIC," *IEEJ Transactions on Sensors and Micromachines*, vol. 134, pp. 108–113, 2014, doi: 10.1541/ieejsmas.134.108.

[6] S. Jain and B. Bhaumik, “An Ultra Low Power ECG Signal Processor Design for Cardiovascular Disease Detection,” in *2015 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015, pp. 857–860, doi: 10.1109/EMBC.2015.7318497.

[7] J. Pan and W. Tompkins, “A Real-Time QRS Detection Algorithm,” *IEEE Transactions on Biomedical Engineering*, vol. 32, no. 3, pp. 230–236, 1985, doi: 10.1109/TBME.1985.325532.

[8] M. Schoeberl, “Wildcat: Educational RISC-V Microprocessors,” *arXiv*, Feb. 2025, doi: 10.48550/arXiv.2502.20197.

[9] Y. Zou *et al.*, “An Energy-Efficient Design for ECG Recording and R-Peak Detection Based on Wavelet Transform,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 2, pp. 119–123, 2015, doi: 10.1109/TCSII.2014.2368619.

[10] A. Barriga, “RISC-V Processors Design: A Methodology for Cores Development,” in *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*, 2020, pp. 1–6, doi: 10.1109/DCIS51330.2020.9268639.

[11] A. Raveendran, V. B. Patil, D. Selvakumar, and V. Desalphine, “A RISC-V Instruction Set Processor–Micro-Architecture Design and Analysis,” in *2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA),* 2016, pp. 1–7, doi: 10.1109/VLSI-SATA.2016.7593047.

[12] M. R. P., P. Niranjan, and D. K. M. J., “Design and Implementation of 32-bit RISC-V Processor Using Verilog,” in *2024 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, 2024, pp. 181–186, doi: 10.1109/DISCOVER62353.2024.10750638.

[13] D. K. Dennis *et al.*, “Single Cycle RISC-V Micro Architecture Processor and Its FPGA Prototype,” in *2017 7th International Symposium on Embedded Computing and System Design (ISED)*, 2017, pp. 1–5, doi: 10.1109/ISED.2017.8303926.

[14] G. Kanase and A. M. Al-Busaidi, “ASIC Design of a 32-bit Low Power RISC-V-Based System Core for Medical Applications,” in *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, 2021, pp. 1–5, doi: 10.1109/ICCES51350.2021.9489067.

[15] M. S. Sanober and S. S. M. Malik, “ECG Signals Analysis: A Comprehensive Literature Review,” *International Journal of Research Publication and Reviews*, vol. 5, no. 4, pp. 3227-3235, 2024.

[16] S. Kumar, G. Singh, and M. Kaur, “FPGA Implementation of Electrocardiography (ECG) Signal Processing,” *International Journal of Engineering Sciences*, vol. 8, pp. 58–70, 2016.

[17] D. Panigrahy, M. Rakshit, and P. K. Sahu, “FPGA Implementation of Heart Rate Monitoring System,” *Journal of Medical Systems*, vol. 40, Art. no. 49, 2016, doi: 10.1007/s10916-015-0410-4.

[18] C. J. Deepu, X. Y. Xu, X. D. Zou, L. B. Yao, and Y. Lian, “An ECG-on-Chip for Wearable Cardiac Monitoring Devices,” in *Fifth IEEE International Symposium on Electronic Design, Test & Applications (DELTA),* Ho Chi Minh City, Vietnam, Jan. 13–15, 2010, pp. 1–5, doi: 10.1109/DELTA.2010.43.

[19] J. K. Kim, J. H. Oh, G. B. Hwang, O. S. Gwon, and S. E. Lee, “Design of Low Power SoC for Wearable Healthcare Device,” *Journal of Circuits, Systems and Computers*, vol. 29, no. 7, Art. no. 20500851, 2020, doi: 10.1142/S0218126620500851.

[20] H. N. Abdullah and B. H. Abd, “A Simple FPGA System for ECG R-R Interval Detection,” in *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, Hefei, China, 2016, pp. 1379–1382, doi: 10.1109/ICIEA.2016.7603800.

[21] M. Janveja, R. Parmar, G. Trivedi, P. Jan, and Z. Nemec, “An Energy Efficient and Resource Optimal VLSI Architecture for ECG Feature Extraction for Wearable Healthcare Applications,” in *2022 32nd International Conference Radioelektronika (RADIOELEKTRONIKA)*, Kosice, Slovakia, 2022, pp. 1–6, doi: 10.1109/RADIOELEKTRONIKA54537.2022.9764910.

[22] W. Hu, C. C. Lin, and L. Y. Shyu, “An Implementation of a Real-Time and Parallel Processing ECG Features Extraction Algorithm in a Field Programmable Gate Array (FPGA),” in *2011 Computing in Cardiology*, Hangzhou, China, 2011, pp. 801–804.

[23] Z. Chen, Z. Lin, H. Chen, X. Wang, and Y. Wu, “Design and Implementation of a Low-Complexity R-Peak Detection Algorithm,” in *2018 China Semiconductor Technology International Conference (CSTIC)*, Shanghai, China, 2018, pp. 1–3, doi: 10.1109/CSTIC.2018.8369334.

[24] A. B. Yandrapati, S. Nilagiri, H. K. Yakkanti, and P. Domathoti, “Design and Implementation of RISC-V Based Pacemaker,” in *2023 8th International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, 2023, pp. 123–127, doi: 10.1109/ICCES57224.2023.10192752.

[25] C. Crema, A. Depari, A. Flammini, and A. Vezzoli, “Efficient R-Peak Detection Algorithm for Real-Time Analysis of ECG in Portable Devices,” in *2016 IEEE Sensors Applications Symposium (SAS)*, Catania, Italy, 2016, pp. 1–6, doi: 10.1109/SAS.2016.7479846.

[26] N. Bayasi, T. Tekeste, H. Saleh, B. Mohammad, A. Khandoker, and M. Ismail, “Low-Power ECG-Based Processor for Predicting Ventricular Arrhythmia,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 5, pp. 1962–1974, 2016, doi: 10.1109/TVLSI.2015.2475119.

[27] L. Khriji and A. M. Al-Busaidi, “New Adaptive Thresholding-Based ECG R-Peak Detection Technique,” in *2018 IEEE 4th Middle East Conference on Biomedical Engineering (MECBME)*, Tunis, Tunisia, 2018, pp. 147–152, doi: 10.1109/MECBME.2018.8402423.

[28] M. Bhosale and A. Chitre, “Real-Time ECG Acquisition and FPGA-Based QRS Detection,” in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Pune, India, 2018, pp. 1–4, doi: 10.1109/ICCUBEA.2018.8697710.

[29] Q. Long, Y. Ren, J. Han, and X. Zeng, “VLSI Implementation for R-Wave Detection and Heartbeat Classification of ECG Adaptive Sampling Signals,” in *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, Hangzhou, China, 2016, pp. 1597–1599, doi: 10.1109/ICSICT.2016.7998814.

[30] K. Chinmayi and M. Padmaja, “VLSI Implementation of ECG Feature Extraction Using Integer Haar Wavelet and EMD Algorithm,” in *2023 First International Conference on Cyber Physical Systems, Power Electronics and Electric Vehicles (ICPEEV)*, Hyderabad, India, 2023, pp. 1–6, doi: 10.1109/ICPEEV58650.2023.10391840.

## BIOGRAPHIES OF AUTHORS

**Vinayak Vikram Shinde** completed his B.E. (Electronics and Telecommunication Engineering), from Gharda Institute of Technology, Ratnagiri, India in 2021. He is currently pursuing M.Tech. (VLSI and Embedded Systems) in Department of Electronics and Telecommunication Engineering in PCCOE, Pune. His areas of interest include RTL design, FPGA prtotyping, embedded systems, and custom processor architectures. He is proficient in VHDL, C, and Python, and has hands on experience with tools such as Xilinx Vivado, Microwind, LT Spice, and KiCad. He can be contacted at email: shindevinayak192@gmail.com.

**Prof. Sheetal Umesh Bhandari** SMIEEE, obtained her B. E., M. E. (with distinction), and Ph.D. degrees in Electronic Engineering from the University of Pune, India. She was also a visiting researcher at Polimi, Italy. She has over 22 years of experience – initially as an Entrepreneur and then as an academic (at various tiers). She is a Professor and Dean at the Department of Electronics and Telecommunication Engineering, Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University, Pune, India. Her academic and research focus is reconfigurable computing, semiconductor devices, and circuits. She can be contacted at email: sheetal.bhandari@pccoepune.org.

**Deepti Snehal Khurge** an Associate Professor at Pimpri Chinchwad College of Engineering, Pune, holds a Ph.D. from RTMNU and has 17 years of academic experience. She has experience in VLSI design, physical design, quantum computing, digital systems, and EDA tools. She has published 18 journal papers, 20+ conference papers, 3 book chapters, and 4 patents and 5 copyrights. She is actively involved in NBA and Autonomy processes and serves as Technical Program Chair for ICCUBEA 2024. Her is certifications include DAAD ProGRANT, QUantum Technologies, AVR Programming, Coursera, and AICTE initiatives. She is passionate about innovation-driven teaching, interdisciplinary research, and continuous upskilling. Her leadership in technical events, mentoring, and curriculum development demonstrates her commitment to academic excellence and institutional growth. She can be contacted at email: dipti.khurge@pccoepune.org.

**Satyashil Dasharath Nagarale** an Assistant professor in the Department of Electronics and Telecommunication Engineering at Pimpri Chinchwad College of Engineering, Pune. He has been associated with PCCOE since July 2011 and is a UGC approved faculty member. He has over 3 years of industry experience in ASIC/FPGA design, verification, and design for testability. His academic and research interests lie in the areas of FPGA-based system design, edge AI acceleration, AI-based predictive battery management systems, and embedded system development. He can be contacted at email: satyashil.nagrale@pccoepune.org.

**Ujwal Ramesh Shirode** received Ph.D. degree in Electronics and Telecommunication from Savitribai Phule Pune University, Pune, Maharashtra in May 2024. M.Tech. Degree in VLSI Technology from North Maharashtra University, Maharashtra, Jalgaon, in 2013. He is currently working as an assistant professor at the Department of Electronics and Telecommunication Engineering PCCOE, Pune, Maharashtra, India. His research interests include VLSI based low power system design and internet of things. He has 04 Granted patent in the filed of Electronics and Engineering. He can be contacted at email: ujjwal.shirode@pccoepune.org.