**Design and implement a real-time ECG signal processing system on an FPGA using Vivado.**

**Goals:**

* Detect QRS complexes from a real-time ECG signal
* Filter out noise (baseline wander, powerline interference)
* Display or store detected heart rate

**How to Approach in Vivado**

**1. Define System Architecture**

Break the system into components:

* Signal Acquisition (simulated or real ADC interface)
* Filtering (e.g., low-pass, high-pass FIR filters)
* Feature Extraction (e.g., QRS detection using Pan-Tompkins algorithm)
* Output (LED, UART, display, etc.)

**2. Design Modules in Verilog/VHDL**

Example modules:

* filter.vhd: Implements FIR or IIR filters
* qrs\_detector.v: Edge-detection and threshold-based detection
* uart\_tx.v: To send heart rate over serial

**3. Use Vivado IP Integrator (Optional)**

* Use Xilinx’s **DSP IP cores** for filtering
* **AXI4 interfaces** for easy data transfer between blocks

**4. Test and Simulate**

* Use Vivado's **testbench** environment for simulating ECG signal
* Use a test waveform (e.g., MIT-BIH dataset or a synthetic sine wave)

**5. Implement on FPGA**

* Choose a board like **Zybo Z7**, **ZedBoard**, or **Artix-7**
* Connect inputs/outputs (ADC, UART, LEDs)

**📚 Example Applications in Biomedical Field**

You can customize the open-ended problem for:

| **Application** | **Goal** |
| --- | --- |
| ECG Monitoring | Real-time QRS detection and heart rate |
| EMG Signal Processing | Muscle signal detection and pattern analysis |
| Pulse Oximeter | Extract oxygen saturation from red/IR light |
| Neural Interfaces | Spike detection from EEG/Neural signals |

**What is a QRS Complex?**

In an ECG (Electrocardiogram) signal:

* **QRS complex** represents **ventricular depolarization** (heartbeat).
* It’s the sharp spike in ECG and **key to heart rate calculation**.

**🧩 Goal:**

**Detect QRS complexes in real-time from an ECG signal using an FPGA** (Vivado + Verilog/VHDL).

**🔁 Basic Signal Processing Pipeline**

Here’s a simplified architecture you can implement in Vivado:

**🧱 FPGA Block Diagram:**

css

CopyEdit

[ ECG Input ] ──► [ Preprocessing Filter ]

↓

[ Derivative ]

↓

[ Squaring ]

↓

[ Moving Window Integration ]

↓

[ Threshold + Peak Detector ]

↓

[ QRS Detection Output ]

**🔧 Step-by-Step Modules (Verilog or VHDL in Vivado)**

**1. Filtering Stage**

Removes baseline wander and high-frequency noise.

* Use **Low-pass FIR filter** (cut-off ~15 Hz)
* Use **High-pass filter** (cut-off ~0.5 Hz)

Can use:

* Custom Verilog filter
* Or **Vivado FIR IP core**

**2. Derivative Module**

Emphasizes rapid slope changes (QRS has steep slopes).

assign diff = ecg[n] - ecg[n-1];

**3. Squaring**

Amplifies large values, makes all values positive.

assign squared = diff \* diff;

**4. Moving Window Integration (MWI)**

Smooths out peaks over a window (e.g., 150 ms):

always @(posedge clk) begin

sum = sum - buffer[ptr] + squared;

buffer[ptr] = squared;

ptr = ptr + 1;

avg = sum / window\_length;

end

**5. Thresholding and Peak Detection**

If integrated signal > dynamic threshold → QRS detected.

if (avg > threshold) begin

qrs\_detected <= 1;

end else begin

qrs\_detected <= 0;

end

Use refractory period logic (e.g., 200 ms) to avoid detecting the same QRS multiple times.

**🧪 Input ECG Signal for Testing**

**Use:**

* **MIT-BIH ECG data** (converted to binary format)
* OR generate ECG waveform in MATLAB/Python and load to FPGA using BRAM or UART.

**🧱 Implementation in Vivado**

**Tools & Features:**

* **Vivado Design Suite**: Synthesis, IP Integration
* **Xilinx IPs**: FIR Compiler, BRAM, UART
* **Simulation**: Vivado Testbench or XSIM
* **Board**: Artix-7, Zybo Z7, or Nexys A7 (for real-time output)

**📈 Output Options**

* **LED blink** on each heartbeat (QRS)
* **UART Output**: Send heart rate to terminal
* **7-segment display**: Show BPM

**🔬 Biomedical Application Use Case**

| **Application** | **Real-time FPGA Use** |
| --- | --- |
| Heart rate monitor | QRS detection + BPM |
| Arrhythmia alert system | QRS irregularity |
| Fitness devices | Live HR monitoring |