Project Report

Air Quality Monitoring and Alert System

Mohammad Fares Aljamous 1088672

Omar Mohammad 1088546

Hadi Albanna 1088677

Supervised By: Eng. Gasm Elbary and Dr. Samr

جامعة أبوظبي
ABU DHABI UNIVERSITY

Submitted: November 24, 2024

Contents

List of Figures

List of Tables

Abstract

This project describes the design, testing, and deployment of a real-time Air Quality Monitoring System that addresses the difficulty of reliably evaluating pollutant concentrations, such as carbon dioxide (CO2). The system uses an FPGA (Cyclone V) to analyze input from air quality sensors using regression algorithms to filter out noise, improving the precision of air quality measurements. Any change in air quality is visibly reflected by LEDs, while precise pollutant levels are displayed on a 7-segment display. This study discusses the engineering principles that underpin the system's operation, its efficiency in real-world circumstances, and prospective areas for future improvement.

1 Introduction

Nowadays, as CO2 levels have risen dramatically, so has global warming. Air quality is critical for human health and environmental sustainability, but current sensors frequently give noisy data, making precise assessment of pollutants such as carbon dioxide (CO2) hard. This project addresses this issue by creating a real-time Air Quality Monitoring System utilizing an FPGA (Cyclone V). By using regression approaches to remove noise, the system improves data accuracy while retaining real-time processing capabilities. Visual indications such as LEDs indicate general air quality, while a 7-segment display shows precise pollution levels. This combination of modern hardware and noise reduction ensures consistent performance under a variety of settings, assisting with urban monitoring, industrial compliance, and public health activities to improve decision-making and air quality management.

1.1 Motivation

This project is more than just building a circuit; it's about inspiring change and solving an important issue. It focuses on ensuring the safety and the healthiness of the people around us. The FPGA, is a key to making a positive impact on the world around us. Our design will be a significant step towards combining responsible technology with the urgent need to protect our environment and our future on this planet. By working on this project, we are not just making something; we are driving meaningful change.

1.2 Problem Statement

Maintaining high air quality in many urban and industrial locations is critical for protecting both human health and the environment. However, it is difficult to effectively monitor air quality. Our group intends to address this issue by developing and installing an advanced real-time air quality monitoring system. This system will employ an FPGA (Cyclone V) to interpret data from air quality sensors and apply regression algorithms to filter out noise and properly estimate air quality. The technology will use LEDs to provide a visual indicator of overall air quality while also displaying specific pollutant levels on a 7-segment display. Implementing this technology promises to provide a dependable and efficient solution for real-time air quality measurement, boosting environmental monitoring efforts, and enabling better health outcomes.

1.3 Literature Review

Air quality monitoring systems have grown to become crucial in addressing environmental and health issues. The literature has investigated a variety of existing options, each with its own strategy and technology for measuring and analyzing air quality. While these systems efficiently monitor pollutants, their benefits and limitations vary depending on the technology and approach used.

Traditional air quality monitoring systems generally use independent sensors to assess the levels of pollutants such as CO_2, particulate matter (PM), and nitrogen oxides (NO_x). These systems are simple to set up and inexpensive for basic monitoring applications. However, their accuracy is frequently compromised by noise in sensor data, climatic fluctuations, and inadequate data processing capabilities. Studies show that while these systems are commonly utilized in personal and industrial applications, they lack the sophisticated processing capability required for precise, real-time analysis and prediction [1] [2].

Microcontroller- or microprocessor-based air quality monitoring systems outperform independent sensors in terms of data processing capabilities. Microcontrollers enable simple noise reduction techniques, the integration of many sensors, and the viewing of pollution levels. While cost-effective and energy-efficient, these systems have limitations when it comes to sophisticated computations or real-time data processing. Furthermore, as noted in the literature, microcontroller systems can struggle with scalability and flexibility when advanced capabilities such as predictive modeling or data aggregation are used in smart city applications [3] [2].

The literature is increasingly pointing to FPGA technology as a possible option for real-time environmental monitoring. FPGAs enable the construction of parallel processing architectures, such as effective noise filtering and predictive algorithms, directly on the hardware. Unlike microcontrollers, FPGAs provide high-speed processing and versatility, making them ideal for dealing with noisy sensor input and conducting real-time calculations. Existing research, however, identifies obstacles, such as the greater initial cost of FPGA systems and the need for specialized knowledge in design and implementation [1] [2].

A review of the existing literature reveals a significant gap in systems for air quality monitoring that combine precision, real-time processing, and flexibility. FPGA-based systems, with their capacity to incorporate advanced noise filtering and predictive algorithms, offer a promising solution for closing this gap. This project intends to use these capabilities to create an efficient and accurate air quality monitoring system that addresses the inadequacies of previous approaches while also advancing real-time environmental monitoring technology [2] [3].

2 Design

To address the constraints of the project, we started by brainstorming and organizing our ideas into a structured plan. We then reviewed and discussed these ideas with the engineer to refine our approach. Our initial step involved building a regression model, testing it thoroughly, and experimenting with different implementation methods to identify the most effective approach. Once the core model was functional, we turned our attention to meeting the remaining requirements, such as integrating outputs like LEDs and seven-segment displays. Throughout this process, we developed the project framework using logical design principles and techniques we had learned during the course. Finally, we implemented the design by programming in VHDL, applying the skills and concepts taught throughout our studies.

2.1 Requirements Constraints, and Considerations

- Our objective was to design an air quality monitoring system by analyzing CO_2 concentration and developing a regression model to calculate the Air Quality Index (AQI). We then translated the system's logical design into VHDL code for implementation.

- Constraints:

  1. **Input Data:** The system will receive CO_2 concentration values as 8-bit digital inputs through FPGA switches. These values will be used to compute the Air Quality Index (AQI) via a regression model.
  2. **Preprocessing:** The system must efficiently handle noisy sensor data, employing techniques like regression filtering or moving average methods to ensure reliable AQI predictions.

3. **Output Specifications:** The AQI must be classified into five categories (Good, Moderate, Unhealthy, Very Unhealthy, Hazardous), which will be visually represented through LED indicators. The exact AQI value will be displayed on a 7-segment display in real-time.
4. **System Implementation:** The system design must be implemented on an FPGA, specifically the Cyclone V board, with VHDL code used to process the data, integrate the regression model, and manage the output components.

2.2 Design Process

To develop the final design, we followed a structured approach. The first step was to create the regression model, which was essential for predicting the Air Quality Index (AQI) based on CO_2 concentration. After that, we focused on determining the coefficients for the model, which allowed us to calculate the AQI. Next, we designed a logical block diagram for the regression model and experimented with different methods to represent the system structure visually. We then translated this diagram into VHDL code. The following step involved creating a blinker logic diagram to control the LED output, which was necessary to indicate specific conditions, such as when the AQI falls into certain categories. Once we had the diagram, we wrote the corresponding VHDL code. To further refine the design, we needed a state decider to evaluate the AQI and trigger the correct output to the LED unit. We implemented this logic and wrote the corresponding VHDL code as well. Finally, the AQI value needed to be displayed on a seven-segment display. To achieve this, we converted the AQI values from binary to BCD (Binary Coded Decimal), and then from BCD to a seven-segment representation. We created a diagram to guide us in writing the VHDL code for this step.

2.2.1 Basic components: Mux, Counter, and Comparator

To begin the system design and ensure a structured approach, we developed several fundamental components that simplified the implementation process. These components include the 8-to-1 multiplexer (MUX), counter, and comparator, each serving a critical role in the overall system.

1. **Multiplexer (MUX):** The 8-to-1 multiplexer was used to select one

of the eight input lines based on the three-bit selection signal. This selection mechanism is essential for directing data flow through various parts of the system, especially in cases where multiple input sources need to be routed through a single output. The VHDL implementation of the MUX is shown below, where the output is determined by the selection bits D.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

ENTITY mux8x1 is
    port(
        in0, in1, in2, in3, in4, in5, in6, in7: in std_logic;
        D: in std_logic_vector(2 downto 0);
        output: out std_logic
    );
end ENTITY;

architecture behv of mux8x1 is
begin
    process(in0, in1, in2, in3, in4, in5, in6, in7, D)
    begin
        case D is
            When "000" => output <= in0;
            When "001" => output <= in1;
            When "010" => output <= in2;
            When "011" => output <= in3;
            When "100" => output <= in4;
            When "101" => output <= in5;
            When "110" => output <= in6;
            When "111" => output <= in7;
        end case;
    end process;
end architecture;
```

2. **Counter:** The counter component was used in multiple parts of the system. It played a key role in generating the necessary clock cycles for

both the regression unit (to process the shifted input values) and the blinker logic. The counter increments its output on each clock cycle and is reset when the reset signal is active. The VHDL code for the counter is as follows:

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

ENTITY accu is
    generic (
        n: integer
    );
    port(
        reset, clk: in std_logic;
        output: buffer std_logic_vector(n-1 downto 0)
    );
end ENTITY;

architecture behv of accu is
begin
    process(clk, reset)
    begin
        if reset = '1' then
            output <= (others => '0');
        elsif rising_edge(clk) then
            output <= output + 1;
        end if;
    end process;
end architecture;
```

3. **Comparator:** The comparator logic was designed to compare two values, determining whether one is greater than, less than, or equal to the other. This comparison function is vital in controlling various decision-making processes within the system, such as triggering the correct outputs based on the AQI value. The comparator's VHDL code is as follows:

```vhdl
library ieee;
```

```
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

ENTITY comparator is
    generic (
        n: integer
    );
    port(
        c1, c2: in std_logic_vector(n-1 downto 0);
        output: out std_logic
    );
end ENTITY;

architecture behv of comparator is
begin
    process(c1, c2)
    begin
        if c1 = c2 then
            output <= '1';
        else
            output <= '0';
        end if;
    end process;
end architecture;
```

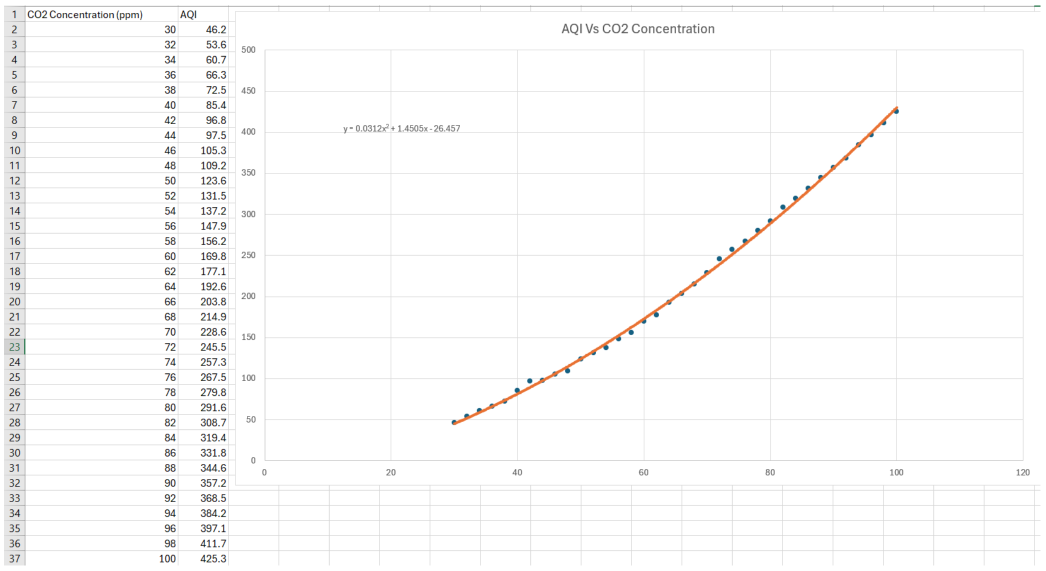
Each of these components played an integral role in the final design, enabling efficient data processing, comparison, and control. The MUX was used to route data based on selection bits, the counter generated necessary timing signals, and the comparator performed logical comparisons essential for decision-making within the system. Together, they formed the backbone of the system, allowing us to implement the AQI prediction model and manage the corresponding outputs effectively.

2.2.2 Regression Model Derivation:

1. The first step in the process was to create a regression model that could predict the Air Quality Index (AQI) based on CO_2 concentration, measured in parts per a hundred thousand, alongside the corresponding

AQI values, the reason for using parts per a hundred thousand instead of parts per million is that in that we were forced to use 8-bit inputs, and with 8-bits, the maximum number you can represent is 255, so by using parts per hundred thousand the maximum we can represent is 2550 parts per million instead of 255 parts per million. We began by inputting the data into an Excel file and using the plot feature to visualize the relationship between CO_2 levels and AQI. From this, we determined that a parabolic equation Equation 1 best fits the data.

$$y = C_2 \cdot x^2 + C_1 \cdot x + C_0 \tag{1}$$



	CO2 Concentration (ppm)	AQI
2	30	46.2
3	32	53.6
4	34	60.7
5	36	66.3
6	38	72.5
7	40	85.4
8	42	96.8
9	44	97.5
10	46	105.3
11	48	109.2
12	50	123.6
13	52	131.5
14	54	137.2
15	56	147.9
16	58	156.2
17	60	169.8
18	62	177.1
19	64	192.6
20	66	203.8
21	68	214.9
22	70	228.6
23	72	245.5
24	74	257.3
25	76	267.5
26	78	279.8
27	80	291.6
28	82	308.7
29	84	319.4
30	86	331.8
31	88	344.6
32	90	357.2
33	92	368.5
34	94	384.2
35	96	397.1
36	98	411.7
37	100	425.3

Figure 1: Using Excel For Regression Model.

2. Next, we focused on calculating the coefficients for the regression equation. This required setting up a system of equations with the data. Since the matrix of CO_2 values applied to the equation was not square, direct inversion was not feasible. To solve this, we multiplied both sides of the equation by the transpose of the matrix, which allowed us to solve for the coefficient vector C using the formula $(X^T * X)^{-1} * (X^T * Y)$, where Y represents the output AQI values.

3. During this process, we observed that the C_0 coefficient was generating

negative values, which were unrealistic given that AQI cannot be negative. To address this, we set C_0 to zero, as negative values were not consistent with the physical meaning of the AQI.

4. To calculate the exact coefficient values, we implemented Equation 2 in MATLAB, as shown in the code below. Once the coefficients were determined, we substituted them into the regression equation Equation 3, allowing us to calculate the AQI for any given CO_2 concentration and determine its corresponding AQI value.

$$C = (X^T * X)^{-1} * (X^T * Y) \tag{2}$$

```

Ain = [30:2:100]';
B = [46.2, 53.6, 60.7, 66.3, 72.5, 85.4, 96.8, 97.5, 105.3,
    109.2, ...
       123.6, 131.5, 137.2, 147.9, 156.2, 169.8, 177.1, 192.6,
          203.8, ...
       214.9, 228.6, 245.5, 257.3, 267.5, 279.8, 291.6, 308.7,
          319.4, ...
       331.8, 344.6, 357.2, 368.5, 384.2, 397.1, 411.7,
          425.3]';

A = [Ain Ain.^2];

x =(A'*A)^-1*A'*B
```

5. After computing the coefficients, the regression model was finalized as:

$$y = 0.5977x + 0.0374x^2 \tag{3}$$

This equation relates CO_2 concentration (x) to AQI (y). Equation 3 was verified by substituting various CO_2 values and comparing the resulting AQI predictions with observed data, then we were able to write the coefficients as powers of two that can be represented by logical shifts that would be applied on the input binary number.

$$y = (2^{-1} + 3 \cdot 2^{-5} + 2^{-8}) \cdot x + (2^{-5} + 3 \cdot 2^{-9})x * x \tag{4}$$

6. The logic diagram which is shown in Figure 2 illustrates the flow of data in the system. Each block represents a specific operation, such as shifting, adding, and multiplying, which are key steps in calculating the AQI from the CO_2 concentration. The diagram visualizes the interactions between components like the adder, shifter, and multiplier, which process the data before the final AQI value is determined.

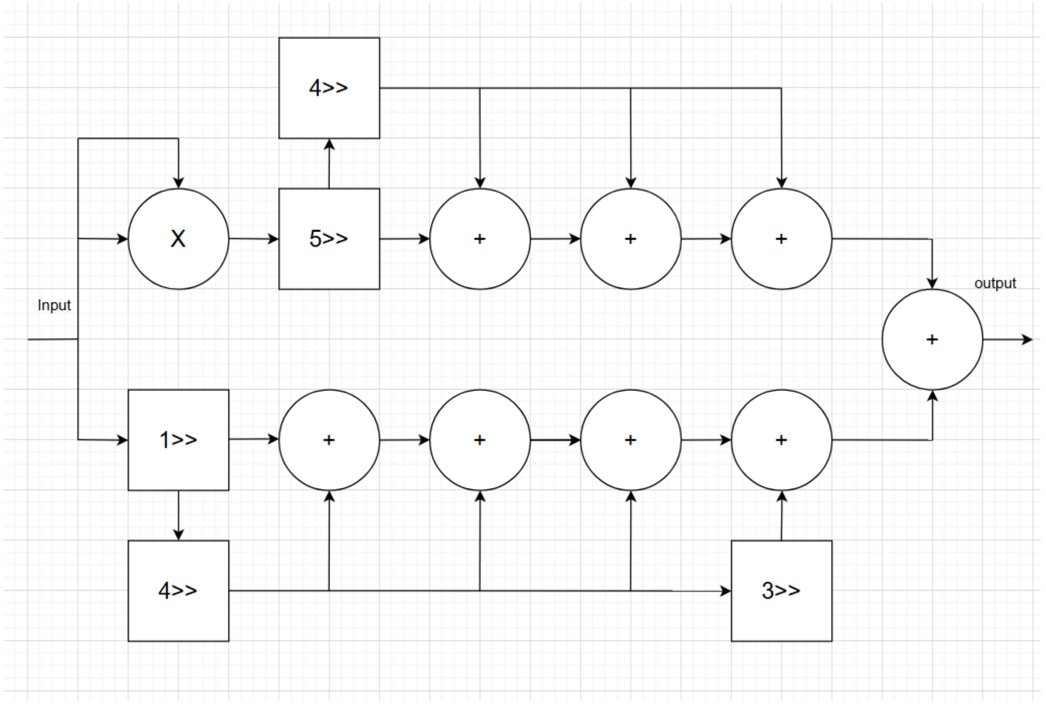


Figure 2: Regression Unit Diagram.

7. Finally, we implemented the regression model in VHDL to enable hardware computation of AQI values based on CO_2 input. Below is the VHDL code used for this purpose. The concentration input represents the CO_2 concentration in parts per a hundred thousand, and the output gives the computed AQI. The design makes use of bitwise shifts, multipliers, and adders to compute the regression model Equation 3 in hardware.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

ENTITY reg is
       port(
              concentration: in std_logic_vector(7 downto 0);
              output: out std_logic_vector(15 downto 0)
       );
end ENTITY;

architecture behv of reg is
       signal x, xs1, xs5, xs8, x2, x2s5, x2s9:
           std_logic_vector(24 downto 0);
       signal s1, s2, s3, s4, s5, s6, s7, s8:
           std_logic_vector(24 downto 0);

       begin
              process(concentration)
                     begin
                            x <= "00000000" &
                                concentration(7 downto 0) &
                                "000000000";
                            x2 <= concentration *
                                concentration & "000000000";
                            xs1 <= '0' & x(24 downto 1);
                            xs5 <= "0000" & xs1(24 downto 4);
                            xs8 <= "000" & xs5(24 downto 3);
                            x2s5 <= "00000" & x2(24 downto
                                5);
                            x2s9 <= "0000" & x2s5(24 downto
                                4);
        s1 <= x2s5 + x2s9;
        s2 <= s1 + x2s9;
        s3 <= s2 + x2s9;
        s4 <= xs1 + xs5;
        s5 <= s4 + xs5;
        s6 <= s5 + xs5;
        s7 <= s6 + xs8;
        s8 <= s7 + s3;
                            output <= s8(24 downto 9) +
                                s8(8);
            end process;

end architecture;
```

2.2.3 Blinker Implementation:

The blinker system is an integral part of the design, used to generate periodic signals essential for visual feedback in digital systems. This component operates by dividing the input clock frequency into a much lower frequency, enabling the toggling of an LED. Below, we describe the system's purpose, design steps, and implementation details.

1. **Significance and Application of the Blinker:** The primary role of the blinker is to visually indicate the state of a digital system. For example, in hardware systems where real-time processing or comparisons are performed, blinking LEDs are often used as status indicators. In the context of our project, this periodic blinking serves as feedback, allowing users to confirm that the system is active and functioning correctly. It is particularly useful in embedded systems to signal specific operations, such as a successful reset, a completed cycle, or error states, depending on the design requirements.

2. **Overview of Blinker Logic:** The blinker system leverages a hierarchical architecture, combining counters (accumulators) and comparators to divide the high-frequency clock signal into a slower toggling output. This modular approach ensures that the frequency and duty cycle of the blinking signal can be adjusted independently by modifying

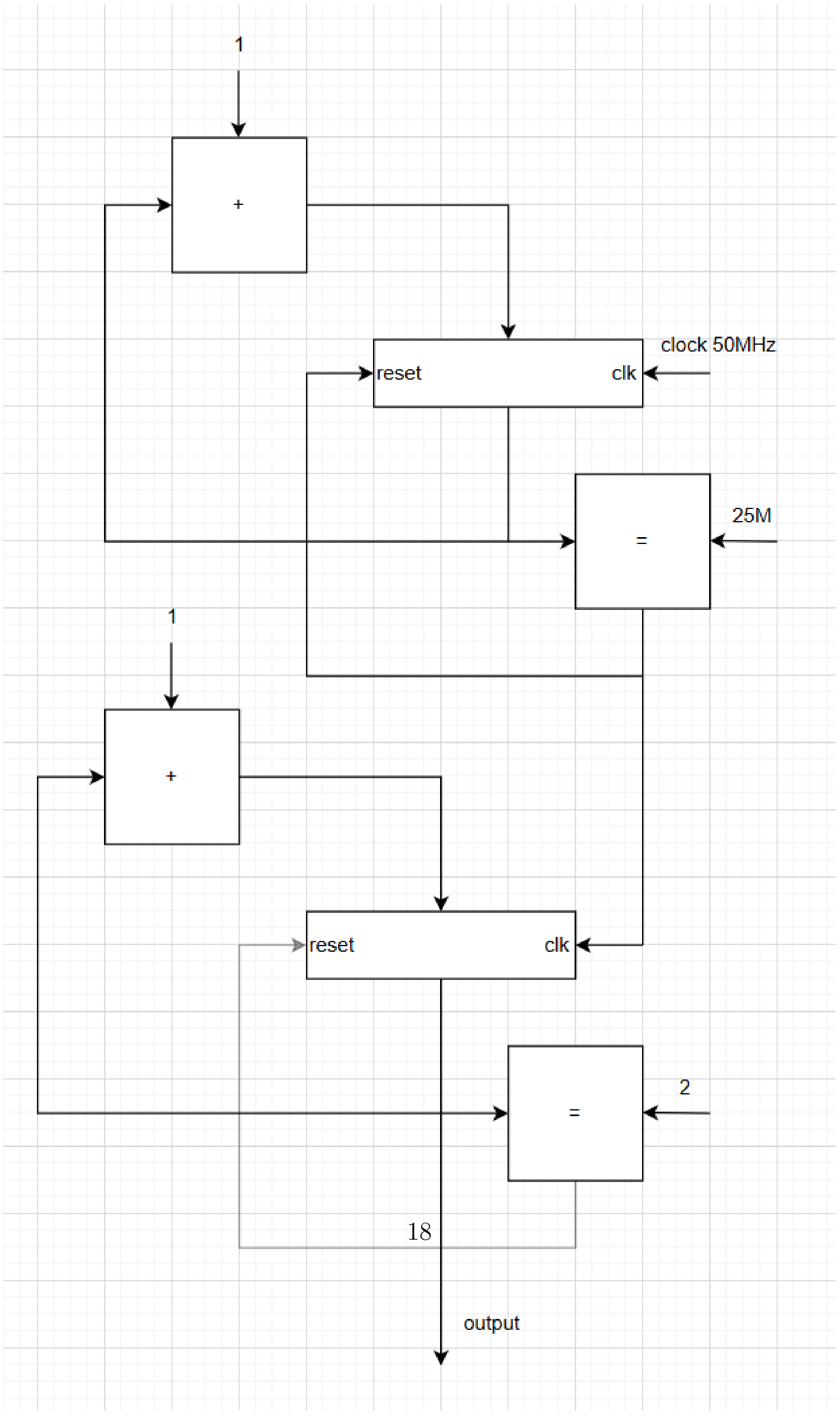
the comparator thresholds or the clock source. The primary components of the system are:

- **Clock Divider:** Reduces the 50 MHz system clock to a lower frequency suitable for blinking. For instance, a frequency of 1 Hz corresponds to a 1-second on-off period.
- **Counter-Comparator Combination:** Accumulators increment with each clock cycle and are reset when the count reaches a predefined threshold. Comparators monitor the counter values and generate the required reset signals.
- **Output Logic:** Produces the final output signal based on the least significant bit (LSB) of the counter, ensuring a consistent toggling pattern.

3. **Logic Diagram:** The structure of the blinker system is illustrated in Figure 3. This diagram highlights the interplay between the accumulators, comparators, and the clock signal in generating the blinking output. As shown, the system consists of two levels of counters and comparators. The first stage handles the clock division, while the second stage ensures that the output toggles at the required frequency.

4. **Step-by-Step Implementation:** The implementation of the blinker system involves the following key steps:

   - **Step 1: Clock Frequency Division**
     The high-frequency clock (50 MHz) drives the first counter (Accumulator 1), which increments on each clock cycle. A comparator (Comp1) resets this counter when it reaches a count equivalent to a predefined period, such as 25 million cycles for a 0.5-second interval. This ensures that the output of Accumulator 1 toggles at a much lower frequency.
   - **Step 2: Secondary Frequency Control**
     The second counter (Accumulator 2) increments with each reset of the first counter. A second comparator (Comp2) monitors this counter and resets it after a specific number of cycles (2). This provides an additional layer of control, determining the overall toggling pattern of the output.


1
+
clock 50MHz
reset
clk
=
25M
1
+
reset
clk
=
2
output

- **Step 3: Output Signal Generation**
  The final blinking signal is derived from the least significant bit (LSB) of the second counter. This ensures a periodic toggling of the output at the desired frequency and duty cycle, driving an LED or other visual indicators.

5. **VHDL Implementation:** The system is implemented using VHDL, as shown in the code snippet below. The code uses modular components, such as accumulators and comparators, to achieve a clean and scalable design.

Listing 1: VHDL Code for the Blinker System

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

ENTITY blinker is
    port(
        clock: in std_logic;
        output: out std_logic
    );
end ENTITY;

architecture behv of blinker is
    Component comparator is
        generic (
            n: integer
        );
        port(
            c1, c2: in std_logic_vector(n-1 downto 0);
            output: out std_logic
        );
    end component;

    Component accu is
        generic (
            n: integer
        );
        port(
                reset, clk: in std_logic;
                output: buffer std_logic_vector(n-1 downto 0)
            );
        end component;

        signal f1: std_logic_vector(25 downto 0);
        signal f2: std_logic_vector(1 downto 0);
        signal o1, o2: std_logic;

    begin
        comp1: comparator
            generic map (n => 26)
            port map (c1 => f1,
                      c2 => "01011111010111100001000000",
                      output => o1);

        comp2: comparator
            generic map (n => 2)
            port map (c1 => f2,
                      c2 => "10",
                      output => o2);

        acc1: accu
            generic map (n => 26)
            port map (reset => o1,
                      clk => clock,
                      output => f1);

        acc2: accu
            generic map (n => 2)
            port map (reset => o2,
                      clk => o1,
                      output => f2);

        output <= f2(0);
    end architecture;
```

6. **Detailed Logic Explanation:**

   - **Accumulator 1:** This 26-bit counter increments on every clock cycle and resets when the comparator detects that the threshold count (25 million) is reached. This creates a signal with a period of 0.5 seconds.

- **Comparator 1:** Monitors the count of Accumulator 1 and generates a reset signal for both Accumulator 1 and the clock input of Accumulator 2.
- **Accumulator 2:** A 2-bit counter that increments with each reset of Accumulator 1, toggling the output signal every second reset (producing a 1-second period).
- **Final Output:** The least significant bit of Accumulator 2 is used as the output, ensuring a clean toggling pattern.

2.2.4 State Decider and LED Output Control

The state decider and LED control modules are vital components in the system, designed to interpret Air Quality Index (AQI) data and provide visual output using LEDs. These components work in tandem to assess the system state and communicate it to the user effectively. Below is a detailed breakdown of their purpose, design logic, and implementation.

State Decider Module:

1. **Purpose:** The state decider processes the AQI input and determines the corresponding state of the system based on predefined AQI ranges. Each state is represented by a unique 3-bit output code (D). This ensures that the system can dynamically adapt to different AQI levels, categorizing them into appropriate states for further processing.

2. **Design Logic:** The state decider implements a series of conditional checks to classify the AQI input into one of the following ranges:

    - **State 0 (000):** AQI ≤ 50 (Good air quality)
    - **State 1 (001):** $51 \leq$ AQI ≤ 100 (Moderate air quality)
    - **State 2 (010):** $101 \leq$ AQI ≤ 150 (Unhealthy for sensitive groups)
    - **State 3 (011):** $151 \leq$ AQI ≤ 200 (Unhealthy)
    - **State 4 (100):** AQI > 200 (Very unhealthy)

- **Default State (111):** For undefined values.

These thresholds are encoded as binary values in the VHDL code, with conditional statements ensuring precise classification.

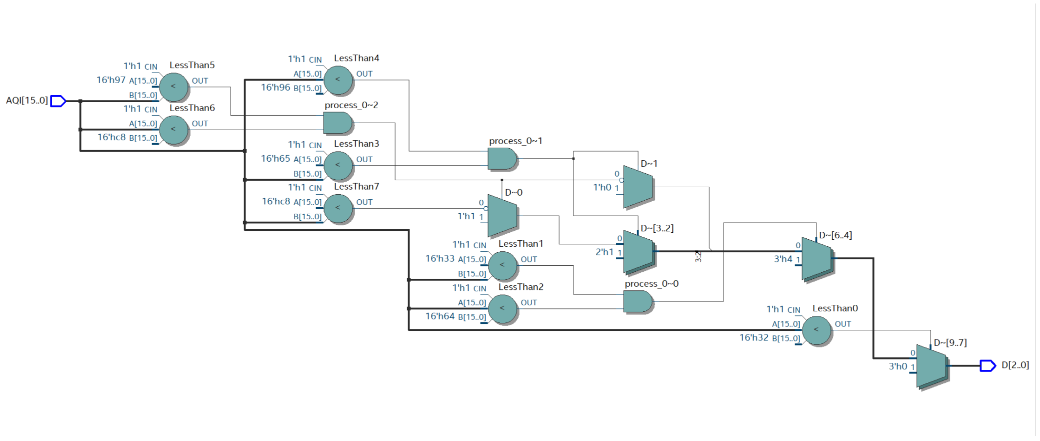


Figure 4: State Decider Diagram.

3. **VHDL Implementation:** The state decider is implemented using the following VHDL code:

Listing 2: VHDL Code for State Decider Module

```vhdl
library ieee;
use ieee.std_logic_1164.all;

ENTITY stateDecider is
    port(
        AQI: in std_logic_vector(15 downto 0);
        D: out std_logic_vector(2 downto 0)
    );
end ENTITY;

architecture behv of stateDecider is
begin
process(AQI)
begin
    if (AQI <= "0000000000110010") then
        D <= "000";
        elsif (AQI >= "0000000000110011" and AQI <=
            "0000000001100100") then
            D <= "001";
        elsif (AQI >= "0000000001100101" and AQI <=
            "0000000010010110") then
            D <= "010";
        elsif (AQI >= "0000000010010111" and AQI <=
            "0000000011001000") then
            D <= "011";
        elsif (AQI >= "0000000011001000") then
            D <= "100";
        else
            D <= "111";
        end if;
    end process;
    end architecture;
```

The **process** block continuously evaluates the AQI input and updates the output **D** to represent the current state.

LED Output Control Module:

1. **Purpose:** The LED output module translates the state determined by the state decider into a visual indication using four LEDs. It combines state-deciding logic with the blinker module to provide dynamic and state-dependent output patterns.

2. **Design Logic:** The LED control module integrates three components:

   - **State Decider:** Maps AQI to a 3-bit state (D) for categorization.
   - **Blinker:** Produces a periodic signal that toggles specific LEDs based on the current state.
   - **Multiplexer (MUX):** Directs input signals (static or blinking) to the LEDs, depending on the state output (D).

   Each LED is configured to illuminate or blink in specific states, as shown in the Figure 5 below.

3. **VHDL Implementation:** The LED control logic is implemented as follows:

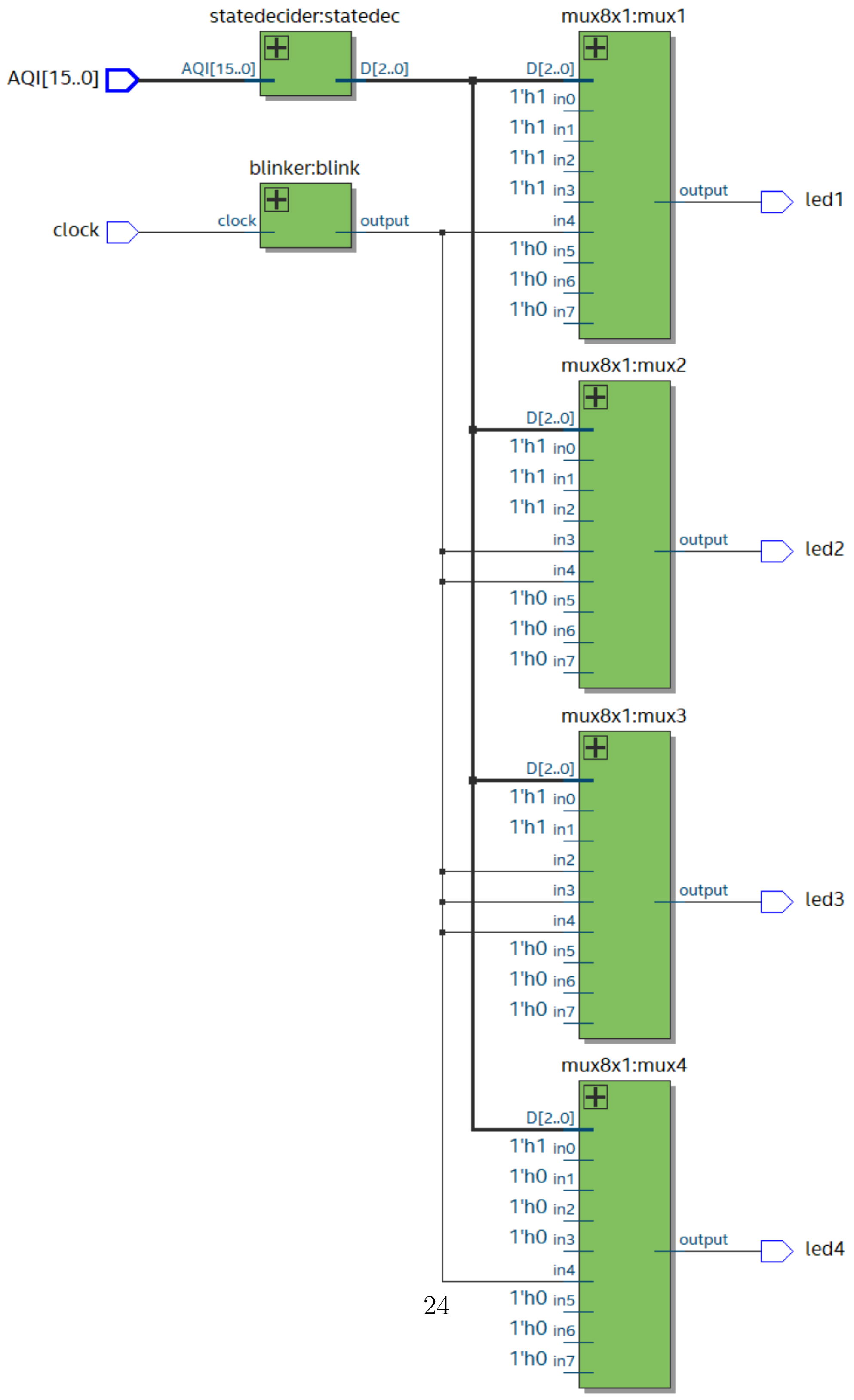


Figure 5: LED Control Diagram.

Listing 3: VHDL Code for LED Control Module

```vhdl
library ieee;
use ieee.std_logic_1164.all;

ENTITY LEDcontrol is
    port(
        AQI: in std_logic_vector(15 downto 0);
        clock: in std_logic;
        led1, led2, led3, led4: out std_logic
    );
end ENTITY;

architecture behv of LEDcontrol is
    component stateDecider is
        port(
            AQI: in std_logic_vector(15 downto 0);
            D: out std_logic_vector(2 downto 0)
        );
    end component;

    component blinker is
        port(
            clock: in std_logic;
            output: out std_logic
        );
    end component;

    component mux8x1 is
        port(
            in0, in1, in2, in3, in4, in5, in6, in7: in
                std_logic;
            D: in std_logic_vector(2 downto 0);
            output: out std_logic
        );
    end component;

    signal d: std_logic_vector(2 downto 0);
    signal b: std_logic;
begin
    statedec: stateDecider port map(AQI => AQI, D => d);
```

```
        blink: blinker port map(clock => clock, output => b);

        mux1: mux8x1 port map(
            in0 => '1', in1 => '1', in2 => '1',
            in3 => '1', in4 => b, in5 => '0',
            in6 => '0', in7 => '0', D => d, output => led1
        );
        mux2: mux8x1 port map(
            in0 => '1', in1 => '1', in2 => '1',
            in3 => b, in4 => b, in5 => '0',
            in6 => '0', in7 => '0', D => d, output => led2
        );
        mux3: mux8x1 port map(
            in0 => '1', in1 => '1', in2 => b,
            in3 => b, in4 => b, in5 => '0',
            in6 => '0', in7 => '0', D => d, output => led3
        );
        mux4: mux8x1 port map(
            in0 => '1', in1 => '0', in2 => '0',
            in3 => '0', in4 => b, in5 => '0',
            in6 => '0', in7 => '0', D => d, output => led4
        );
    end architecture;
```

4. **Detailed Functionality:** The LED output is determined as follows:

    - In states 0 and 1, all LEDs remain static (1).
    - In higher states as **state 4**, some LEDs blink to indicate severity (using the blinker output signal (b)).
    - The multiplexer selects the appropriate signal for each LED based on the state output **D**, ensuring flexibility and modularity.

2.2.5 Binary to BCD and BCD to Seven-Segment Display Implementation:

The Binary to BCD and BCD to Seven-Segment Display systems are essential for converting binary numerical data into human-readable formats. These components enable digital systems to represent numerical outputs on a seven-segment display, commonly used in embedded systems for user interaction.

Below, we describe the purpose, design, and implementation details of these systems.

1. **Significance and Application:** The primary role of these systems is to facilitate human-readable feedback from digital systems:

    - **Binary to BCD Conversion:** Binary data, widely used for processing in digital systems, must be converted to Binary-Coded Decimal (BCD) for display on decimal-based systems.
    - **BCD to Seven-Segment Display:** The BCD representation is mapped to segment control signals, enabling the correct illumination of segments to form readable numerical digits on a seven-segment display.

    In the context of this project, these systems allow for precise representation of computed Air Quality Index (AQI) values on a seven-segment display for user-friendly output.

2. **Overview of the Logic:** The Binary to BCD and BCD to Seven-Segment systems work in a pipeline to process binary inputs and drive the display:

    - **Binary to BCD Conversion:** Converts a 16-bit binary input into four 4-bit BCD outputs (‘d1‘, ‘d2‘, ‘d3‘, ‘d4‘), each representing one decimal digit.
    - **BCD to Seven-Segment Mapping:** Maps the BCD digits to segment control signals (‘a‘ to ‘g‘) for the seven-segment display.

3. **Logic Diagram:** The structure of the Binary to BCD and BCD to Seven-Segment systems is illustrated in Figure 6. The figure shows the stepwise flow of data through the conversion and display process. As shown, the pipeline consists of a binary-to-BCD converter feeding into a BCD-to-seven-segment driver.

4. **Step-by-Step Implementation:** The implementation involves the following steps:

    - **Step 1: Binary to BCD Conversion**
      The Binary to BCD system converts a 16-bit binary input into four 4-bit BCD digits. This is achieved using case-based mapping

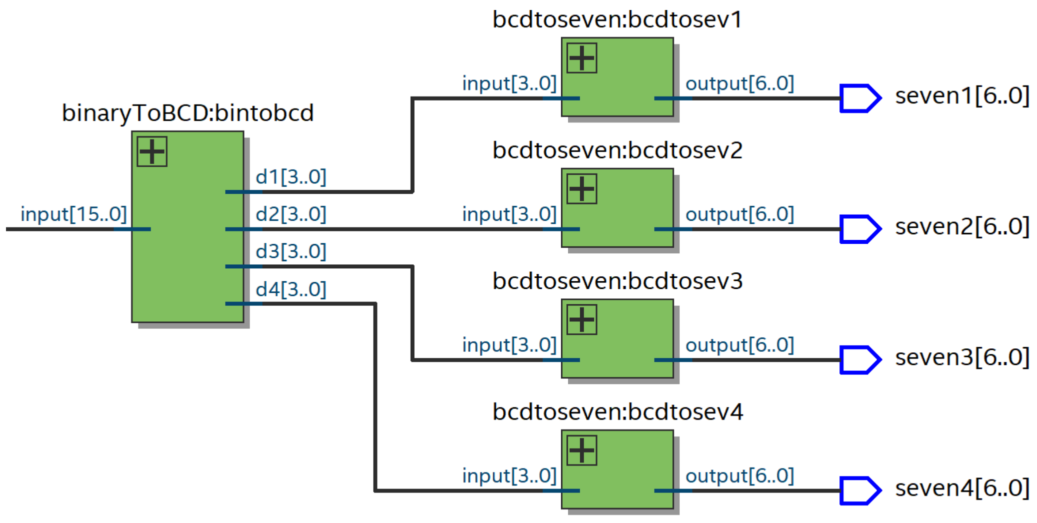


Figure 6: Binary to BCD and Seven-Segment Logic Diagram.

in VHDL. Each binary input is matched to its corresponding BCD representation, which is output as ‘d1‘, ‘d2‘, ‘d3‘, and ‘d4‘.

- **Step 2: BCD to Seven-Segment Mapping**
  The four BCD outputs are individually processed by the BCD-to-seven-segment converter. Each BCD digit is mapped to a specific pattern of the seven segments (‘a‘ to ‘g‘), ensuring correct numerical representation on the display.

- **Step 3: Driving the Seven-Segment Display**
  The segment signals control the illumination of the seven-segment display. For example, the BCD digit ‘0101‘ (representing the number 5) maps to the segment pattern ‘0100100‘, which lights up the segments required to display the digit 5.

5. **VHDL Implementation:** The system is implemented in VHDL using modular design for both subsystems. Key snippets are provided below:

Listing 4: VHDL Code for Binary to BCD Conversion

```
ENTITY binaryToBCD is
    port(
        input: in std_logic_vector(15 downto 0);
        d1, d2, d3, d4: out std_logic_vector(3 downto 0)
```

```vhdl
    );
end ENTITY;

architecture behv of binaryToBCD is
begin
    process(input)
        begin
            case input is
                when "0000000000000000" => d1 <= "0000"; d2
                    <= "0000"; d3 <= "0000"; d4 <= "0000";
                    -- 0
                when "0000101000010110" => d1 <= "0010"; d2
                    <= "1000"; d3 <= "0101"; d4 <= "0010";
                    -- 2582
                when others => d1 <= "0000"; d2 <= "0000";
                    d3 <= "0000"; d4 <= "0000"; -- Default
            end case;
    end process;
end architecture;
```

Listing 5: VHDL Code for BCD to Seven-Segment Conversion

```vhdl
ENTITY bcdtoseven is
    port(
        input: in std_logic_vector(3 downto 0);
        output: out std_logic_vector(6 downto 0)
    );
end ENTITY;

architecture behv of bcdtoseven is
begin
    process(input)
        begin
            case input is
                when "0000" => output <= "0000001"; -- 0
                when "0001" => output <= "1001111"; -- 1
                when "0010" => output <= "0010010"; -- 2
                when "0011" => output <= "0000110"; -- 3
                when "0100" => output <= "1001100"; -- 4
                when "0101" => output <= "0100100"; -- 5
                when "0110" => output <= "0100000"; -- 6
                when "0111" => output <= "0001111"; -- 7
                when "1000" => output <= "0000000"; -- 8
                when "1001" => output <= "0000100"; -- 9
                when others => output <= "1010101"; --
                    Default
            end case;
        end process;
    end architecture;
```

6. **Detailed Logic Explanation:**

    - **Binary to BCD:** This process uses a case-based mapping for simplicity, where each binary input corresponds to a specific set of BCD outputs.
    - **BCD to Seven-Segment:** The seven-segment display is controlled by activating specific segments to represent each decimal digit.
    - **Pipeline Integration:** The outputs from the Binary to BCD system feed directly into the BCD to Seven-Segment system, forming a complete pipeline from binary input to visual display.

2.3 System Overview

The air quality monitoring system is constructed as a modular framework, with distinct components working together to calculate, process, and display the Air Quality Index (AQI) derived from CO_2 concentration data. It is built around four main modules: the regression model, state decider, blinker unit, and display driver (Binary to BCD and BCD to Seven-Segment). The CO_2 concentration data is initially processed by the regression module, which computes the AQI using a predefined parabolic equation. This AQI value is then analyzed by the state decider to classify it into specific categories based on threshold levels, determining the corresponding output states. The blinker module offers visual cues using LEDs to indicate system activity or specific states, while the display driver converts the AQI into a user-readable format through Binary to BCD conversion and subsequent visualization on a seven-segment display. The components are connected through defined interfaces, ensuring seamless communication and scalability, as illustrated in the system diagram. This design, implemented with VHDL, supports real-time computation, accurate outputs, and modular expandability for broader applications.

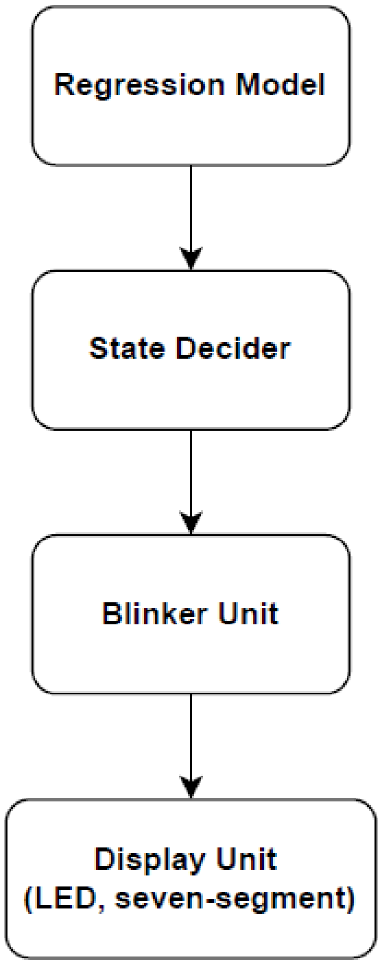


Figure 7: System Overview Diagram.

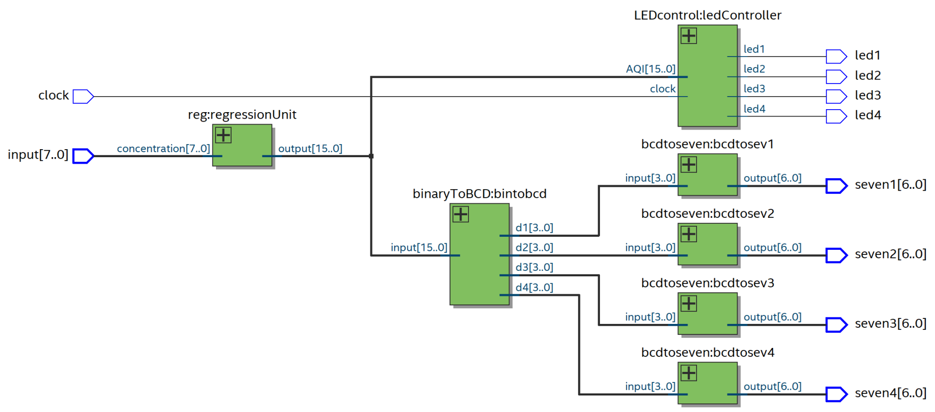
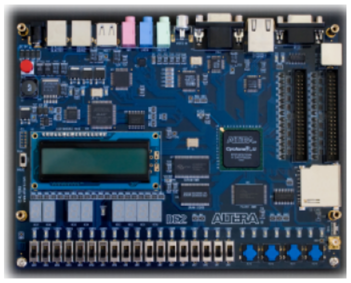


Figure 8: Full System.

2.4 Component Design

The main equipment used in this system is the Cyclone V (5CGXFC5C6F27C7N) FPGA, which serves as the processing unit for calculating the Air Quality Index (AQI). The FPGA processes the 8-bit digital sensor data simulated through the board's switches. It then uses the regression model to compute the AQI, with the results displayed on the 7-segment displays to show the exact AQI value in real-time. The LEDs on the board indicate different air quality levels. In addition, MATLAB and Excel are used for data processing and coefficient determination. MATLAB is responsible for implementing the regression model and calculating the coefficients, while Excel is used to visualize and analyze the relationship between CO2 concentration and AQI. As shown in Figure 9a, the FPGA board is configured with switches, LEDs, and 7-segment displays that provide a comprehensive solution for computing and displaying AQI values.



(a) Diagram of the Cyclone V FPGA with LEDs and 7-segment displays

(b) Altera Quartus II Software

(c) **MATLAB** used to calculate coefficient

(d) Excel used for graphs

Figure 9: List of Equipment Used

2.4.1 Regression Model Derivation

The AQI prediction model based on CO_2 concentration was developed systematically. The relationship between CO_2 levels (ppht) and AQI values was visualized using Excel, identifying a parabolic equation as the best fit, which can be seen in Equation 1. To calculate the coefficients, a system of equations was formulated, and the MATLAB-based solution which shown in Equation 2 was applied. To improve validity, C_0 was set to zero to eliminate unrealistic negative AQI values, refining the final Equation 3. The accuracy of the model was confirmed by comparing predicted AQI values with observed data. The model's computational steps—multiplications, additions, and shifts—were mapped in a logic diagram showing the interaction of components like adders, shifters, and multipliers. The regression model was then implemented in VHDL for real-time hardware execution, utilizing modular components to handle bitwise shifts, multiplications, and additions. This ensured efficient and accurate computation of AQI values directly from CO_2 inputs.

2.4.2 Blinker

The blinker system is a vital component used to provide periodic visual feedback, such as toggling an LED, to indicate the operational state of a digital system. It helps users monitor the system's activity in real-time by signaling active processes, successful operations, or error conditions. The design incorporates a hierarchical approach, utilizing accumulators and comparators to reduce the high-frequency 50 MHz clock signal to a lower frequency suitable for blinking. The system functions by using a 26-bit accumulator that increments with each clock cycle. Once it reaches a specified count, such as 25 million cycles, a comparator resets the counter, producing a signal with a 0.5-second interval. A second 2-bit accumulator, triggered by the reset of the first, continues counting and resets after two cycles, generating a final output signal with a 1-second period. This output is derived from the least significant bit (LSB) of the second accumulator, creating a consistent blinking pattern. The design, implemented using modular VHDL components like counters and comparators, ensures precise and reliable signal generation for real-time feedback.

2.4.3 State Decider and LED Output Control

The state decider and LED control systems play a key role in interpreting AQI data and providing clear visual feedback. The state decider processes AQI inputs, classifying them into one of five distinct states, ranging from "Good" (000) to "Very Unhealthy" (100), using predefined thresholds. These states are represented by a 3-bit binary code, and the system dynamically adjusts based on real-time AQI readings, with the logic implemented in VHDL for precise classification.

The LED control system uses the state information from the decider to activate four LEDs, visually indicating the air quality level. It combines the outputs from the state decider, a blinker for dynamic LED patterns, and multiplexers to assign the right signals to each LED. LEDs remain static in lower AQI states, while they blink in higher states, signaling more severe air quality. This flexible, modular design allows for a straightforward and effective way to display air quality status.

2.4.4 Binary to BCD and BCD to Seven-Segment Display

The Binary to BCD and BCD to Seven-Segment Display systems play a crucial role in converting binary data into a human-readable format on a seven-segment display. These systems are commonly used to present numerical values like the AQI in embedded systems. The Binary to BCD system first translates a 16-bit binary input into four 4-bit BCD outputs, utilizing a case-based mapping process for each binary value. Next, the BCD to Seven-Segment system takes these BCD digits and maps them to segment control signals, activating the necessary segments to form the corresponding numbers on the display. Together, these systems create a smooth flow from binary input to a visual numerical output, all achieved through a modular VHDL design that ensures effective data conversion and accurate display representation.

3 Experimental Testing and Results

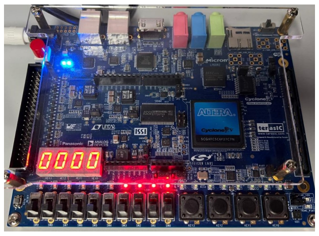
3.1 Testing Plan and Acceptance Criteria

This project was completed using the Quartus II program, so we just uploaded the code to the FPGA for testing. We used switches to enter the CO_2 concentration and checked the seven-segment display to see if it produced the correct AQI. In addition to the LEDs' behavior.

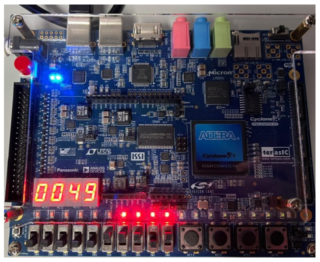
3.2 Results

3.2.1 Testing the accuracy of the regression model

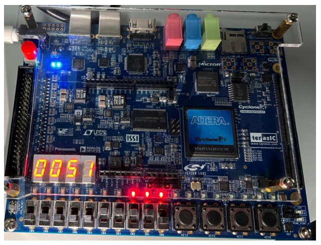
As seen in the Figure 10, the regression model we created provided the correct relationship between the input value and the output, which are the CO_2 concentration and the AQI, respectively. Whereas inputting a specific value of CO_2 yielded the equivalent value of the AQI, indicating that the regression model was designed to provide the most accurate result.



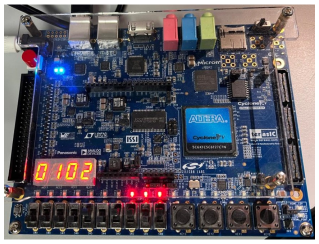
(a) State 0: Co_2 is set to zero using the switches and the AQI is indicated in the seven-segment display as zero, and the LED output is indicated as all four LEDs are on



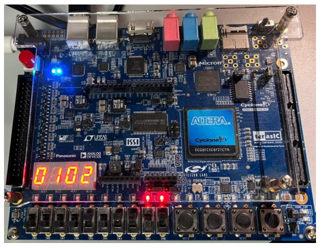
(b) State 0: Co_2 is set to 29 ppht (290 ppm) using the switches, which will be 00011101 and the AQI is indicated in the seven-segment display as 49, and the LED output is indicated as all four LEDs are on



(c) State 1: Co_2 is set to 31 ppht (310 ppm) using the switches, which will be 00011110, and the AQI is indicated in the seven-segment display as 51, and the LED output is indicated as **Three** LEDs are on

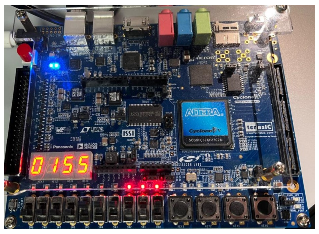


(d) State 2: Co_2 is set to 45 ppht (450 ppm) using the switches, which will be 00101101, and the AQI is indicated in the seven-segment display as 102, and the LED output is indicated as the Third LED will start to **blink** and the first two LEDs will remain on

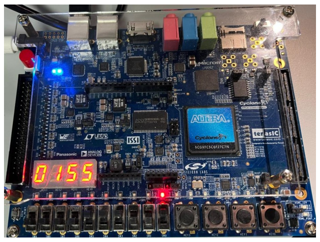


(e) State 2: as shown here and comparing with the previous Figure 10d, it shows that the third LED is **blinking**

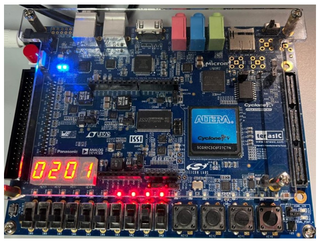
Figure 10: Testing Our System



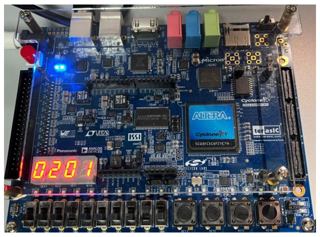
(a) State 3: Co_2 is set to 57 ppht (570 ppm) using the switches, which will be 00111001, and the AQI is indicated in the seven-segment display as 155, and the LED output is indicated as the Second and Third LEDs will start to **blink** and the first LED will remain on



(b) State 3: as shown here and comparing with the previous Figure 11a, it shows that the second and the third LEDs are **blinking** and the first one remains ON



(c) State 4: Co_2 is set to 65 ppht (650 ppm) using the switches, which will be 01000010, and the AQI is indicated in the seven-segment display as 201, and the LED output is indicated as **All LEDs** will start to **blink**



(d) State 4: as shown here and comparing with the previous Figure 11c, it shows that All LEDs are **blinking**

Figure 11: Testing Our System

3.2.2 Testing the state decider

For this particular section, we need to pay attention to the LEDs because they assist us visualize our current situation. After entering a certain value of CO_2 concentration, we examined the AQI value and the criterion specified in the project instructions. In addition, we verified the activity of the

LEDs in each level. Good Air Quality (0-50 AQI): All four LEDs are turned on, indicating optimal circumstances. Moderate Air Quality (AQI 51-100) Unhealthy for Sensitive Groups (101-150 AQI): Two LEDs ON, with a third LED flashing as a warning. Unhealthy (151-200 AQI): One LED illuminated, and two blinking. Very unhealthy (201-300 AQI): All LEDs blink to indicate hazardous air quality conditions. Which turned out to be successful in each stage, as it can be seen in the multiple examples we showed in Figure 11.

3.3 Analysis and Interpretation of Data

As seen in the previous section. We were able to determine the AQI based on the CO_2 concentration level without any difficulty. And this demonstrates how powerful FPGAs are in a variety of sectors and projects, like environmental monitoring, healthcare, and automation..

4 Conclusion

4.1 Summary

This project created a real-time Air Quality Monitoring System with an FPGA (Cyclone V), overcoming the difficulty of noisy sensor data for reliable pollutant evaluation. The system used regression algorithms to filter noise and produce precise forecasts of air quality levels, which were presented via LEDs and a 7-segment display. The FPGA's parallel processing capabilities allowed for effective real-time data handling, achieving the goal of developing a dependable and responsive monitoring tool. The findings highlight the system's effectiveness in improving air quality assessment, as well as its larger implications for environmental monitoring, and provide a scalable solution to help public health and pollution control initiatives.

4.2 Future Improvements and Takeaways

This study has shown that FPGAs are robust and versatile technologies that can be applied in various sectors. We developed an efficient air pollution monitoring system which can tackle the complex tasks by analyzing some of its features, such as concurrent execution and management of data in real time. Such an experience showed the potential of FPGAs in the areas of environmental monitoring, health care, and automation, which are useful in promoting creativity and resolving practical problems.

4.3 Lessons Learned

Because of this project, we acquired useful information on FPGA technology like its real time processing and interfacing with sensors and screen displays. We understood how regression techniques for noise filtering could be applied by making use of some tutorials and other related literature on the predictive algorithm. The experimentation in the laboratory plus the lessons learned from the failed attempts enabled us to make adjustments on how to design the system and in the end improve system performance. This process helped us to gain more skills on the processes of designing FPGAs and their use in monitoring in real time the environment.

4.4 Team Dynamics

- **Who was the team leader? What evidence of good leadership can you provide?** Hadi was the team leader, and he effectively managed task delegation, ensured deadlines were met, and addressed challenges collaboratively. His guidance was evident in achieving project milestones, such as implementing the regression model and integrating outputs like LEDs and seven-segment displays.

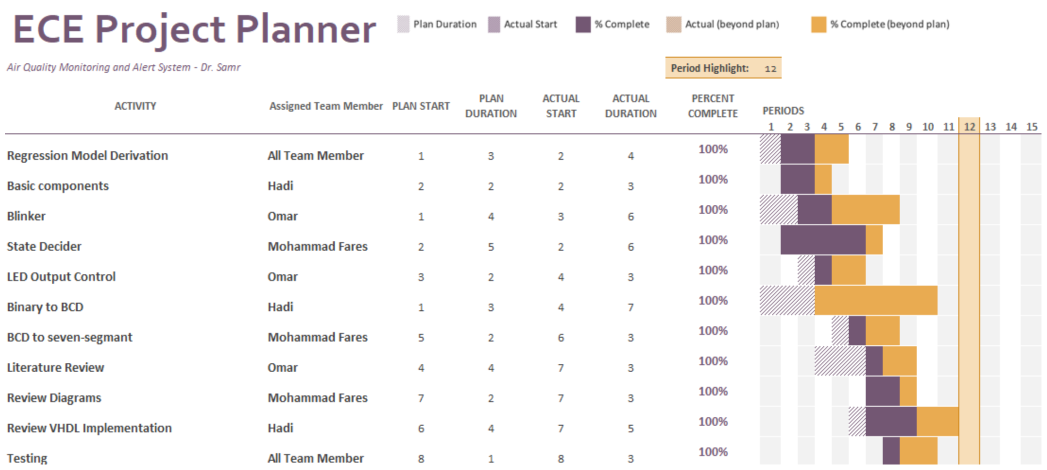
- **How did you create a collaborative and inclusive environment?** Collaboration was achieved through regular meetings in the lab and library, with communication maintained via WhatsApp and MS Teams. All members participated actively, contributing to discussions and task execution. Progress was tracked through updates and discussions.

- **What goals did you set for your team?** The team aimed to:

    1. Develop and test an AQI regression model.
    2. Implement and validate the model on an FPGA board.
    3. Design outputs (LEDs and a seven-segment display) to indicate AQI.

- **How did you plan the tasks? Did you use a Gantt Chart?** Tasks were organized systematically, potentially using a Gantt Chart as can be seen in Figure 12 for tracking milestones. Regular reviews ensured tasks were adjusted and progress was aligned with objectives.

- **Did you meet your objectives?** Yes, the system functioned successfully on the FPGA board, with deliverables like functional VHDL code and a validated regression model completed as required.



ACTIVITY	Assigned Team Member	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Regression Model Derivation	All Team Member	1	3	2	4	100%
Basic components	Hadi	2	2	2	3	100%
Blinker	Omar	1	4	3	6	100%
State Decider	Mohammad Fares	2	5	2	6	100%
LED Output Control	Omar	3	2	4	3	100%
Binary to BCD	Hadi	1	3	4	7	100%
BCD to seven-segmant	Mohammad Fares	5	2	6	3	100%
Literature Review	Omar	4	4	7	3	100%
Review Diagrams	Mohammad Fares	7	2	7	3	100%
Review VHDL Implementation	Hadi	6	4	7	5	100%
Testing	All Team Member	8	1	8	3	100%

Figure 12: Gantt Chart.

4.5 Impact Statement

- **Does your solution help society? How?** The system benefits society by offering real-time air quality monitoring, helping reduce exposure to harmful pollutants, and supporting healthier environments.

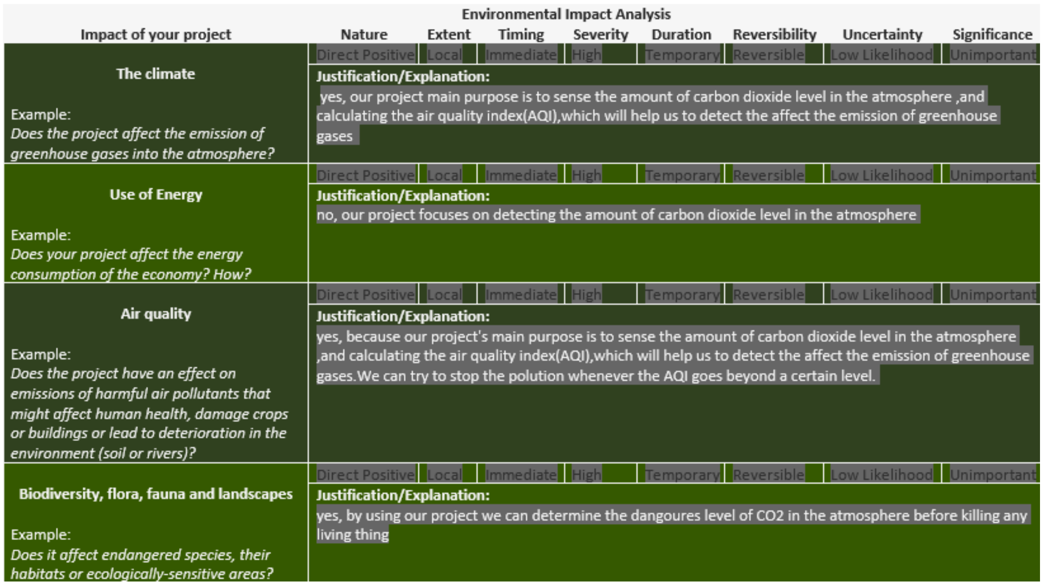
- **Are there any privacy concerns for the users?** No privacy concerns exist since only environmental data is processed, not personal information.

- **Will your solution create jobs? Remove jobs from the market?** It can create jobs in manufacturing, deployment, and data analysis without displacing existing roles.

- **Will the electronics from your project end up being electronic waste? What impact does your system have on the environment?** Modularity minimizes e-waste, allowing repairs and upgrades. The system promotes awareness of air pollution and indirectly supports sustainable energy initiatives.

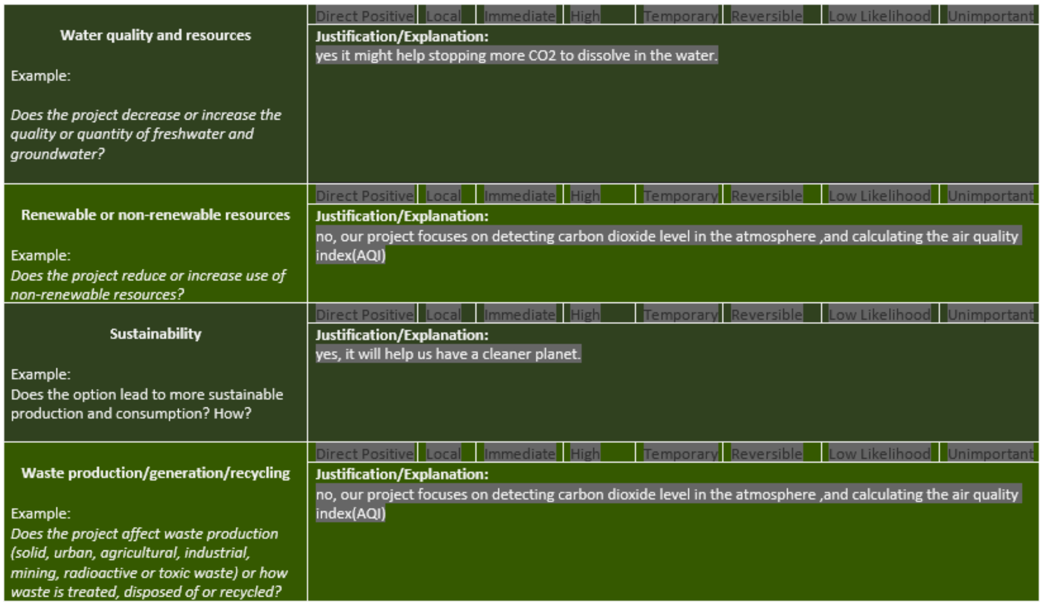
- **Does your system reduce energy demand? Increase energy demand? Is it still worth it?** It slightly increases energy demand, but the minimal impact is justified by the significant societal benefits.

- **Does your system provide comparable performance to existing ones while reducing cost, power consumption, or environmental footprint?** Yes, it offers similar or better performance with efficient energy use, cost-effectiveness, and a low environmental footprint.



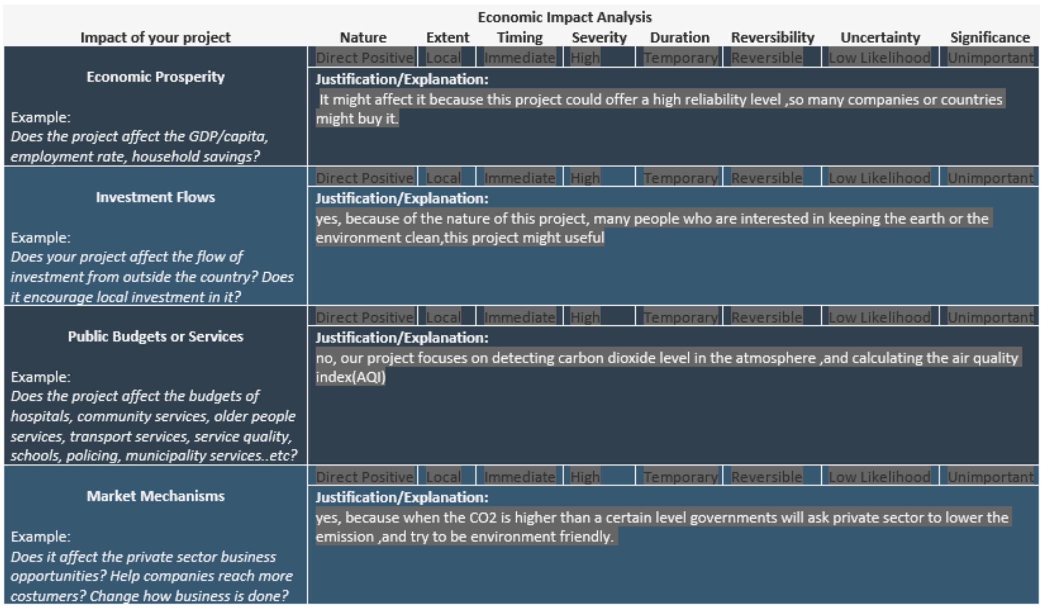
	Environmental Impact Analysis							
Impact of your project	Nature	Extent	Timing	Severity	Duration	Reversibility	Uncertainty	Significance
The climate Example: *Does the project affect the emission of greenhouse gases into the atmosphere?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, our project main purpose is to sense the amount of carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI),which will help us to detect the affect the emission of greenhouse gases							
Use of Energy Example: *Does your project affect the energy consumption of the economy? How?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on detecting the amount of carbon dioxide level in the atmosphere							
Air quality Example: *Does the project have an effect on emissions of harmful air pollutants that might affect human health, damage crops or buildings or lead to deterioration in the environment (soil or rivers)?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, because our project's main purpose is to sense the amount of carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI),which will help us to detect the affect the emission of greenhouse gases.We can try to stop the polution whenever the AQI goes beyond a certain level.							
Biodiversity, flora, fauna and landscapes Example: *Does it affect endangered species, their habitats or ecologically-sensitive areas?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, by using our project we can determine the dangoures level of CO2 in the atmosphere before killing any living thing							

Figure 13: Impact Statement.



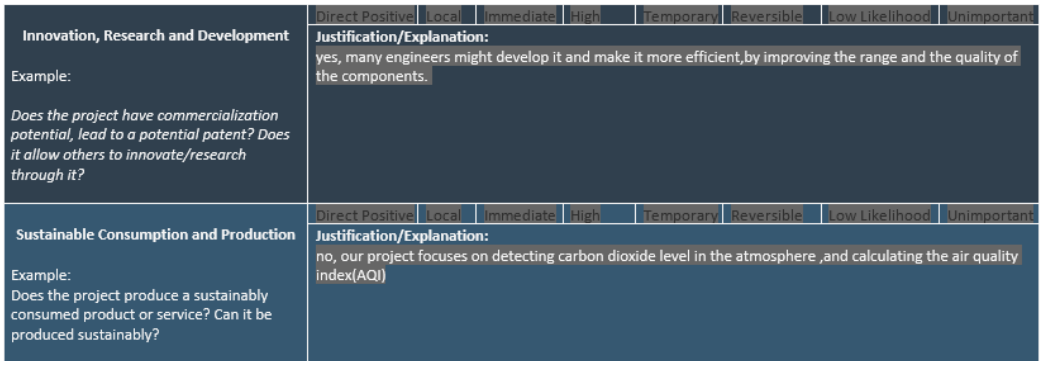
Water quality and resources Example: *Does the project decrease or increase the quality or quantity of freshwater and groundwater?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes it might help stopping more CO2 to dissolve in the water.							
Renewable or non-renewable resources Example: *Does the project reduce or increase use of non-renewable resources?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on detecting carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI)							
Sustainability Example: Does the option lead to more sustainable production and consumption? How?	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, it will help us have a cleaner planet.							
Waste production/generation/recycling Example: *Does the project affect waste production (solid, urban, agricultural, industrial, mining, radioactive or toxic waste) or how waste is treated, disposed of or recycled?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on detecting carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI)							

Figure 14: Impact Statement 2.



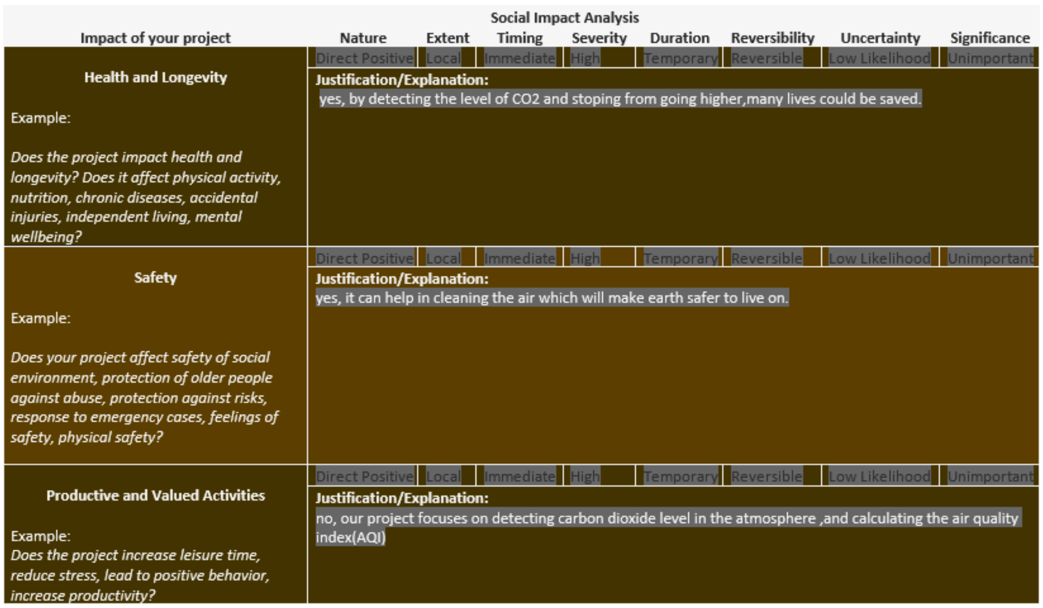
	Economic Impact Analysis							
Impact of your project	Nature	Extent	Timing	Severity	Duration	Reversibility	Uncertainty	Significance
Economic Prosperity Example: *Does the project affect the GDP/capita, employment rate, household savings?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: It might affect it because this project could offer a high reliability level ,so many companies or countries might buy it.							
Investment Flows Example: *Does your project affect the flow of investment from outside the country? Does it encourage local investment in it?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, because of the nature of this project, many people who are interested in keeping the earth or the environment clean,this project might useful							
Public Budgets or Services Example: *Does the project affect the budgets of hospitals, community services, older people services, transport services, service quality, schools, policing, municipality services..etc?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on detecting carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI)							
Market Mechanisms Example: *Does it affect the private sector business opportunities? Help companies reach more costumers? Change how business is done?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, because when the CO2 is higher than a certain level governments will ask private sector to lower the emission ,and try to be environment friendly.							

Figure 15: Impact Statement 3.



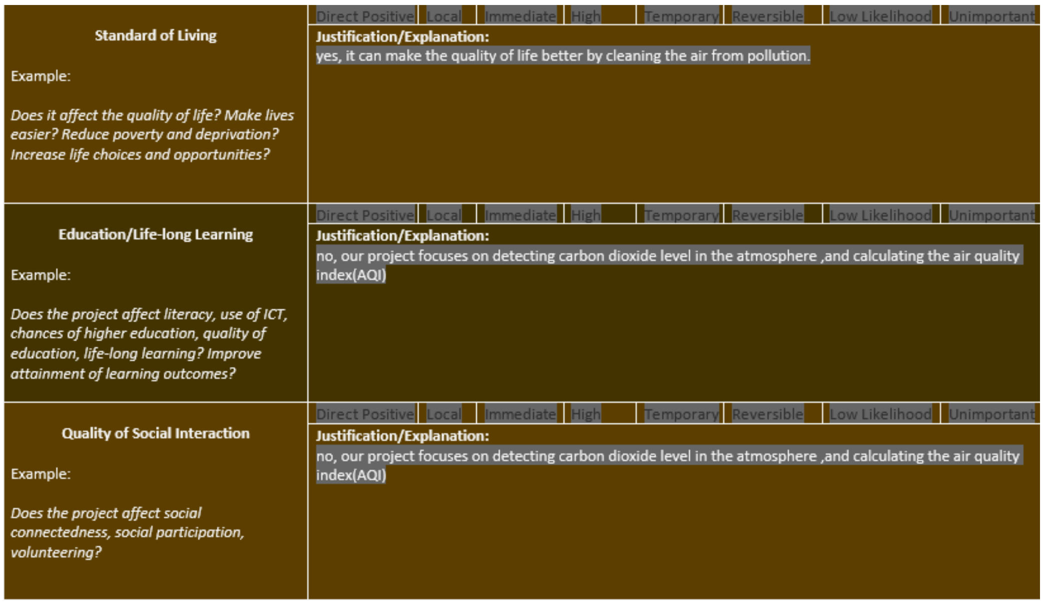
Innovation, Research and Development Example: *Does the project have commercialization potential, lead to a potential patent? Does it allow others to innovate/research through it?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, many engineers might develop it and make it more efficient,by improving the range and the quality of the components.							
Sustainable Consumption and Production Example: Does the project produce a sustainably consumed product or service? Can it be produced sustainably?	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on detecting carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI)							

Figure 16: Impact Statement 4.



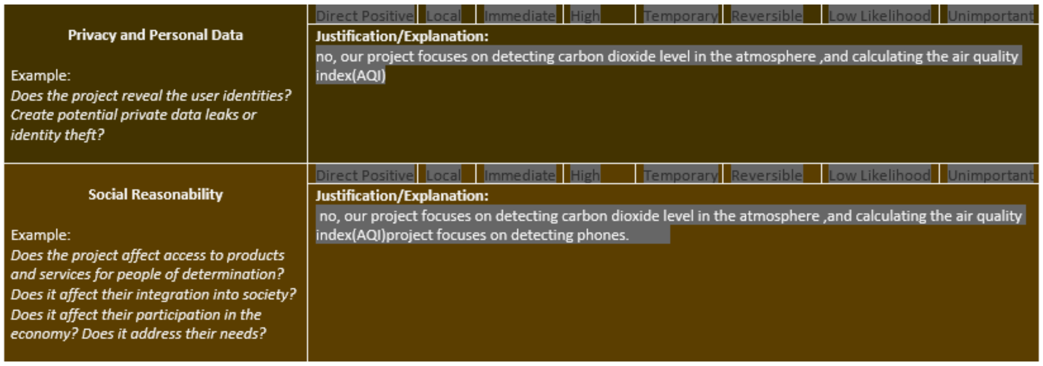
	Social Impact Analysis							
Impact of your project	Nature	Extent	Timing	Severity	Duration	Reversibility	Uncertainty	Significance
Health and Longevity Example: *Does the project impact health and longevity? Does it affect physical activity, nutrition, chronic diseases, accidental injuries, independent living, mental wellbeing?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, by detecting the level of CO2 and stoping from going higher,many lives could be saved.							
Safety Example: *Does your project affect safety of social environment, protection of older people against abuse, protection against risks, response to emergency cases, feelings of safety, physical safety?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, it can help in cleaning the air which will make earth safer to live on.							
Productive and Valued Activities Example: *Does the project increase leisure time, reduce stress, lead to positive behavior, increase productivity?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on detecting carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI)							

Figure 17: Impact Statement 5.



Standard of Living Example: *Does it affect the quality of life? Make lives easier? Reduce poverty and deprivation? Increase life choices and opportunities?*	Direct Positive | Local | Immediate | High | Temporary | Reversible | Low Likelihood | Unimportant **Justification/Explanation:** yes, it can make the quality of life better by cleaning the air from pollution.
Education/Life-long Learning Example: *Does the project affect literacy, use of ICT, chances of higher education, quality of education, life-long learning? Improve attainment of learning outcomes?*	Direct Positive | Local | Immediate | High | Temporary | Reversible | Low Likelihood | Unimportant **Justification/Explanation:** no, our project focuses on detecting carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI)
Quality of Social Interaction Example: *Does the project affect social connectedness, social participation, volunteering?*	Direct Positive | Local | Immediate | High | Temporary | Reversible | Low Likelihood | Unimportant **Justification/Explanation:** no, our project focuses on detecting carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI)

Figure 18: Impact Statement 6.



Privacy and Personal Data Example: *Does the project reveal the user identities? Create potential private data leaks or identity theft?*	Direct Positive | Local | Immediate | High | Temporary | Reversible | Low Likelihood | Unimportant **Justification/Explanation:** no, our project focuses on detecting carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI)
Social Reasonability Example: *Does the project affect access to products and services for people of determination? Does it affect their integration into society? Does it affect their participation in the economy? Does it address their needs?*	Direct Positive | Local | Immediate | High | Temporary | Reversible | Low Likelihood | Unimportant **Justification/Explanation:** no, our project focuses on detecting carbon dioxide level in the atmosphere ,and calculating the air quality index(AQI)project focuses on detecting phones.

Figure 19: Impact Statement 7.

A RTL Representation of Each Component

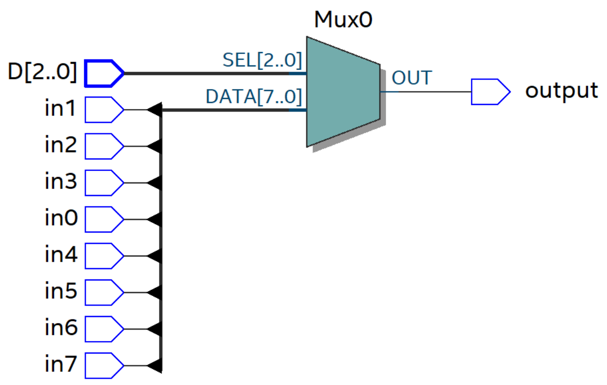


Figure 20: 8x1MUX.

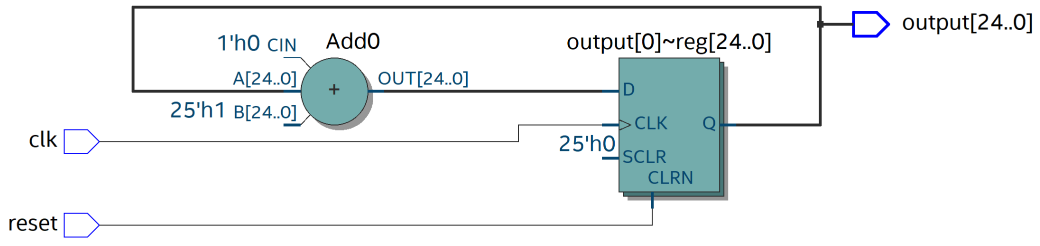


Figure 21: Counter.

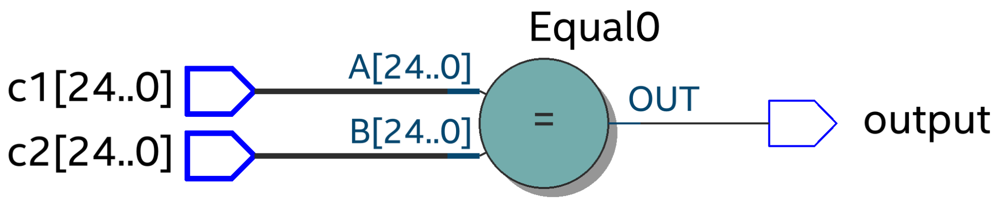


Figure 22: Comparator.

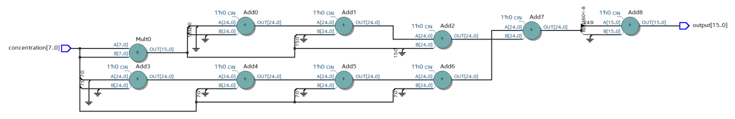


Figure 23: Regression Unit.

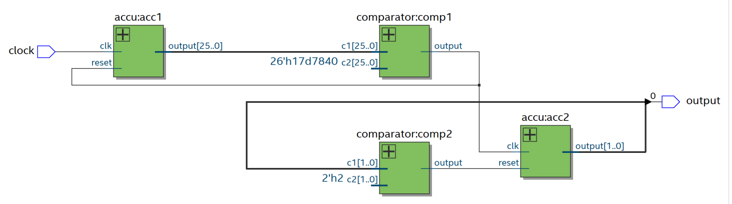


Figure 24: Blinker.

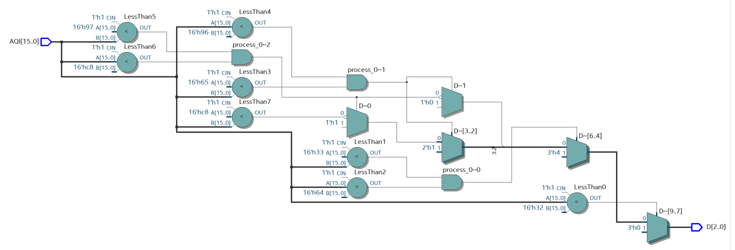


Figure 25: State Decider.

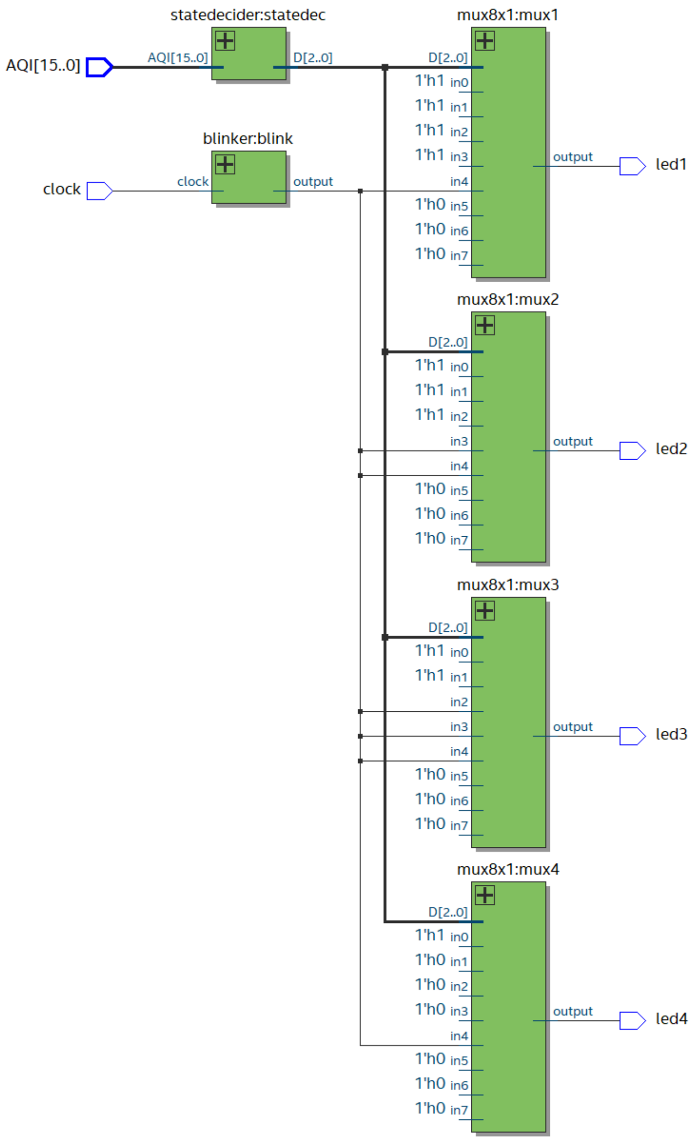


Figure 26: LED Control.

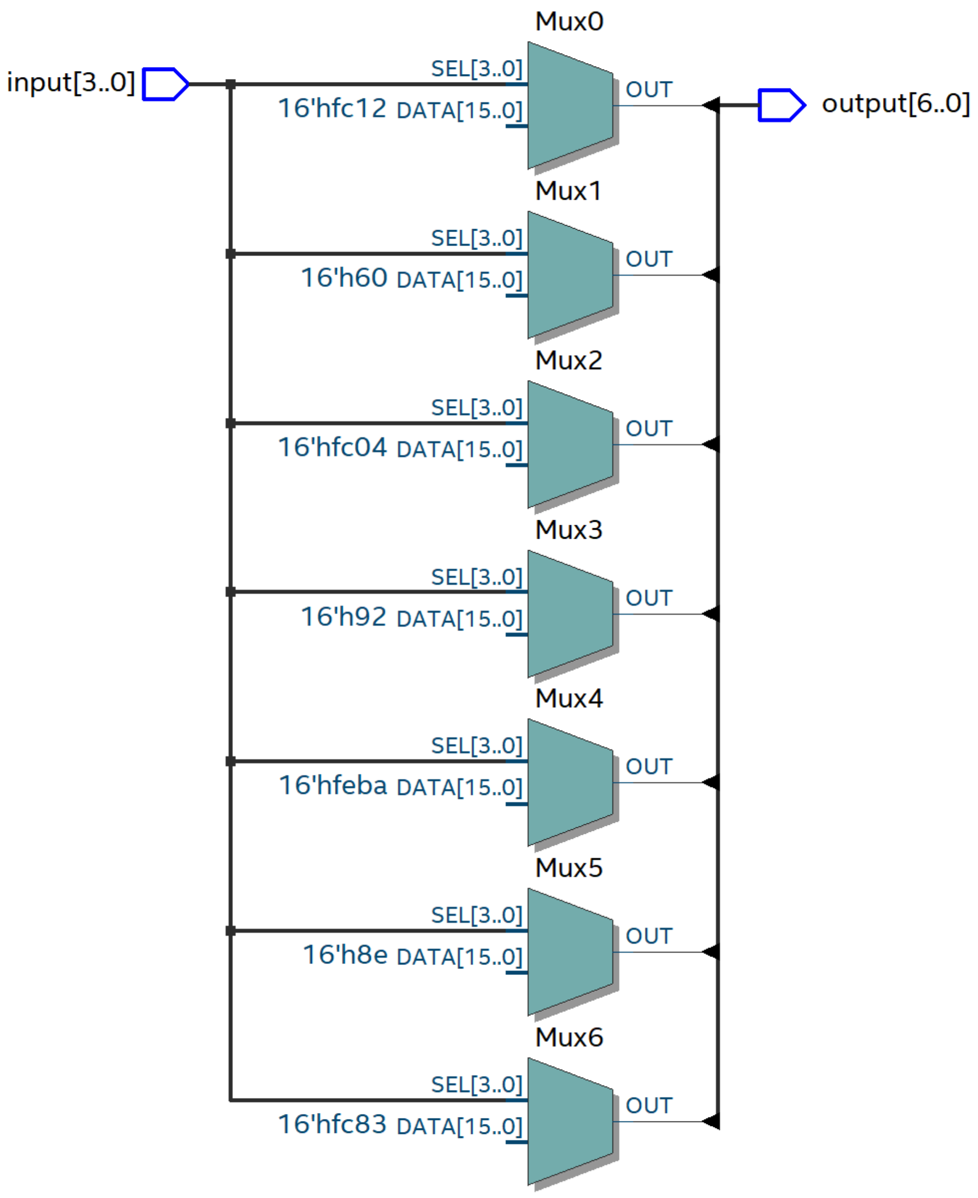


Figure 27: BCD to Seven Segments.

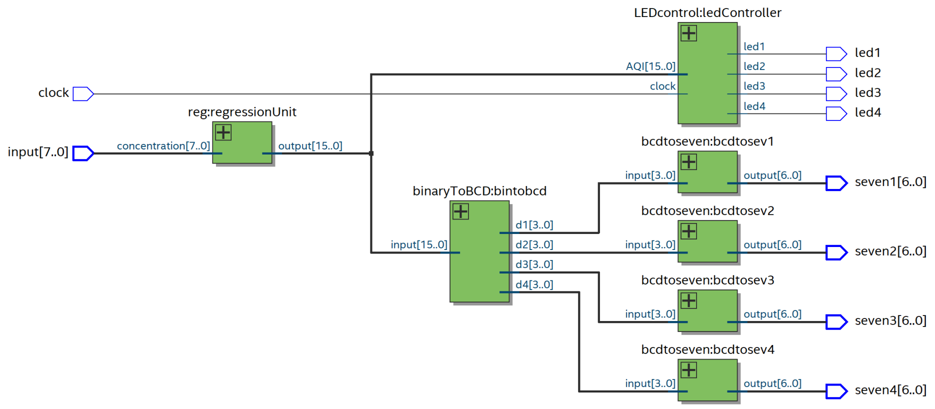


Figure 28: Full System.

References

[1] U. E. P. A. (EPA), “Managing air quality: Ambient air monitoring,” https://www.epa.gov/air-quality-management-process/managing-air-quality-ambient-air-monitoring, 2024, accessed: 2024-11-24.

[2] C. R. S. (CRS), “Ambient air monitoring systems,” https://crsreports.congress.gov, 2024, accessed: 2024-11-24.

[3] U. E. P. A. (EPA), “Low-cost air pollution monitors,” https://www.epa.gov/air-sensor-toolbox/evaluation-low-cost-air-quality-monitors, 2024, accessed: 2024-11-24.

Youtube Link for the Demo