DESIGN PROJECT REPORT

EEX5351
DIGITAL ELECTRONIC SYSTEMS
WATER PUMP CONTROL UNIT (WPCU)

BY

M.N.M. FAZEEL

220260734

SUBMITED TO

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING TECHNOLOGY

THE OPEN UNIVERSITY OF SRI LANKA

AT

COLOMBO

ON

17/11/2024

System Description

Introduction

- The WPCU system is designed to automate the water pumping process to overhead tanks in three different buildings.

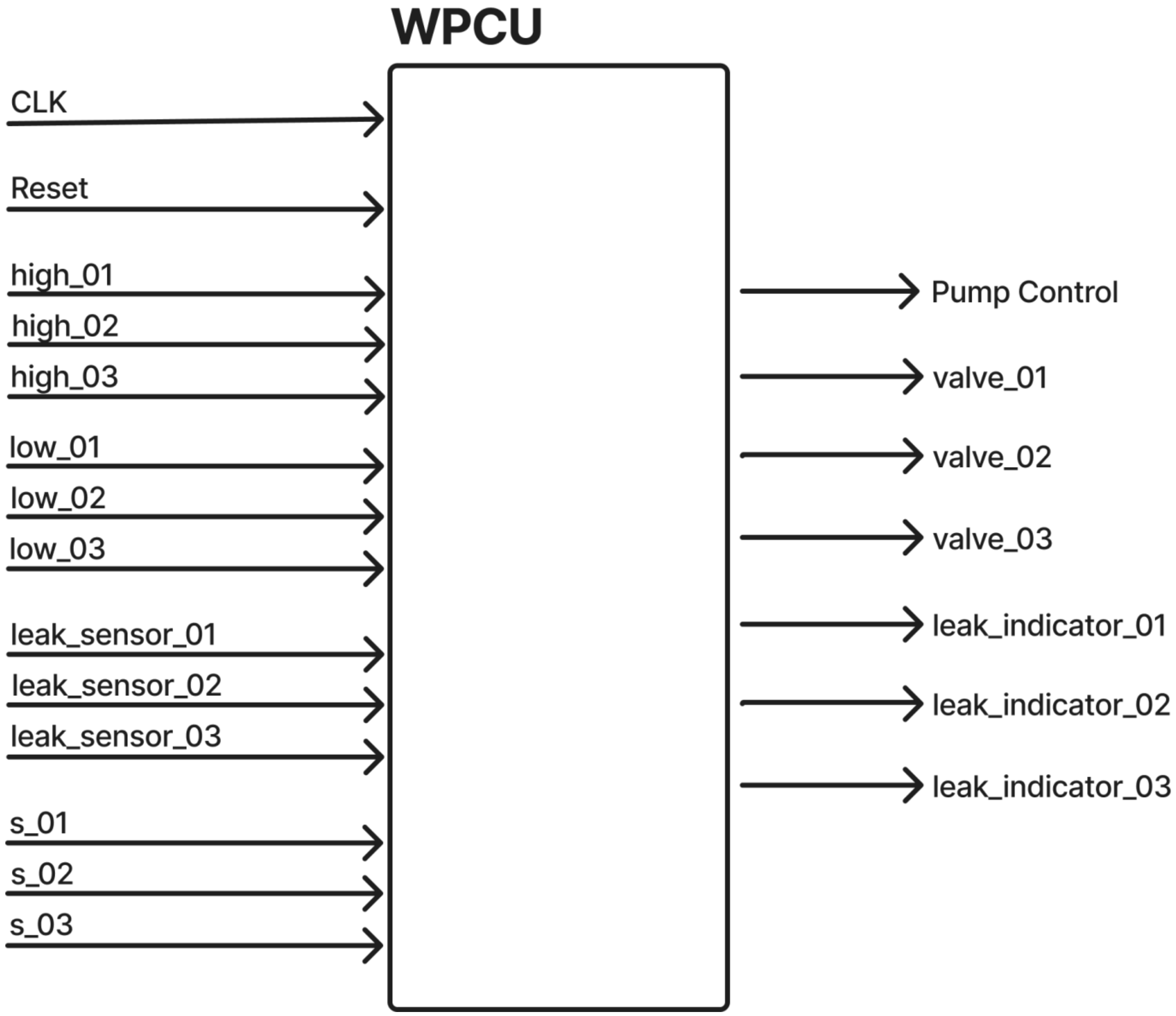
- The system ensures efficient water management by using sensors and control logic to detect tank water levels, manage pumping operations, and check for leaks.

Functional Description

- Water is pumped from a well to one tank at a time until the tank is full.
- If leakage is detected, the pump will stop and light up a warning indicator.
- The system operates based on the ON/OFF switches for each building and can be reset using a reset switch.

Block Diagram and Input Outputs of The System

- **Block Diagram**

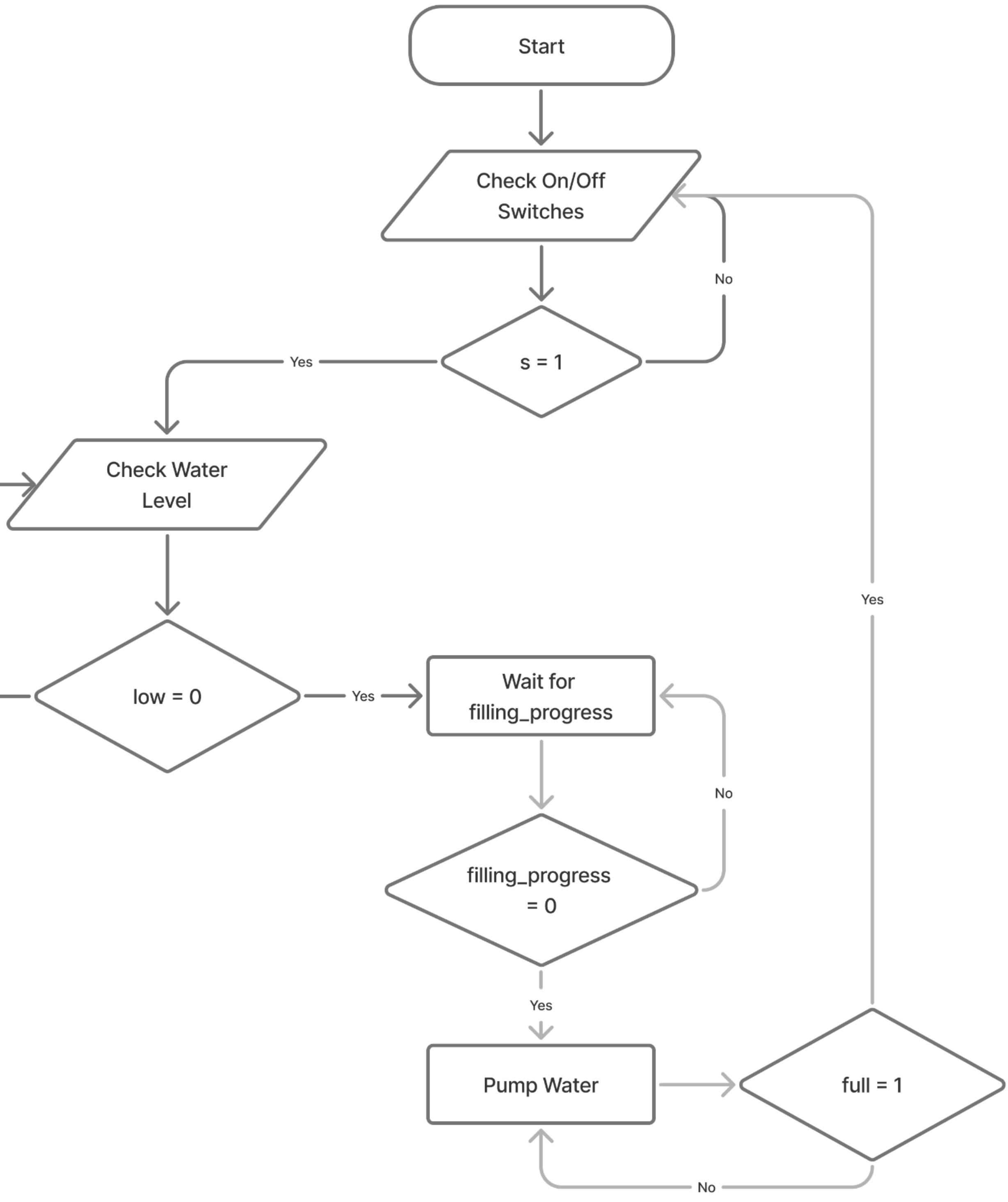


- **Inputs and Outputs**

#	**Inputs**	**Description**
1	CLK	Clock signal for the system
2	RST	Reset signal to reset the signal
3	high_01	Tank upper threshold sensor of tank 1 (1 → Contact, 0 → No Contact)
4	high_02	Tank upper threshold sensor of tank 1 (1 → Contact, 0 → No Contact)
5	high_03	Tank upper threshold sensor of tank 1 (1 → Contact, 0 → No Contact)
6	low_01	Tank lower threshold sensor of tank 1 (0 → No Contact, 0 → Contact)
7	low_02	Tank lower threshold sensor of tank 1 (1 → No Contact, 0 → Contact)
8	low_03	Tank lower threshold sensor of tank 1 (1 → No Contact, 0 → Contact)
9	leak_sensor_01	Water leak sensor of building 1 (1 → Leak Detected, 0 → No Leak Detected)
10	leak_sensor_02	Water leak sensor of building 2 (1 → Leak Detected, 0 → No Leak Detected)
11	leak_sensor_03	Water leak sensor of building 3 (1 → Leak Detected, 0 → No Leak Detected)
12	s_01	On/Off switch of building 1
13	s_02	On/Off switch of building 2
14	s_03	On/Off switch of building 3

#	**Output**	**Description**
15	pump_control	Wayer pump control signal (1 → On, 0 → Off)
16	valve_01	Control signal of solenoid valve of tank 1 (1 → On, 0 → Off)
17	valve_02	Control signal of solenoid valve of tank 2 (1 → On, 0 → Off)
18	valve_03	Control signal of solenoid valve of tank 3 (1 → On, 0 → Off)
19	leak_indicator_01	Leak indicator signal of building 1 (1 → On, 0 → Off)
20	leak_indicator_02	Leak indicator signal of building 2 (1 → On, 0 → Off)
21	leak_indicator_03	Leak indicator signal of building 3 (1 → On, 0 → Off)

Flow Chart for Single Building

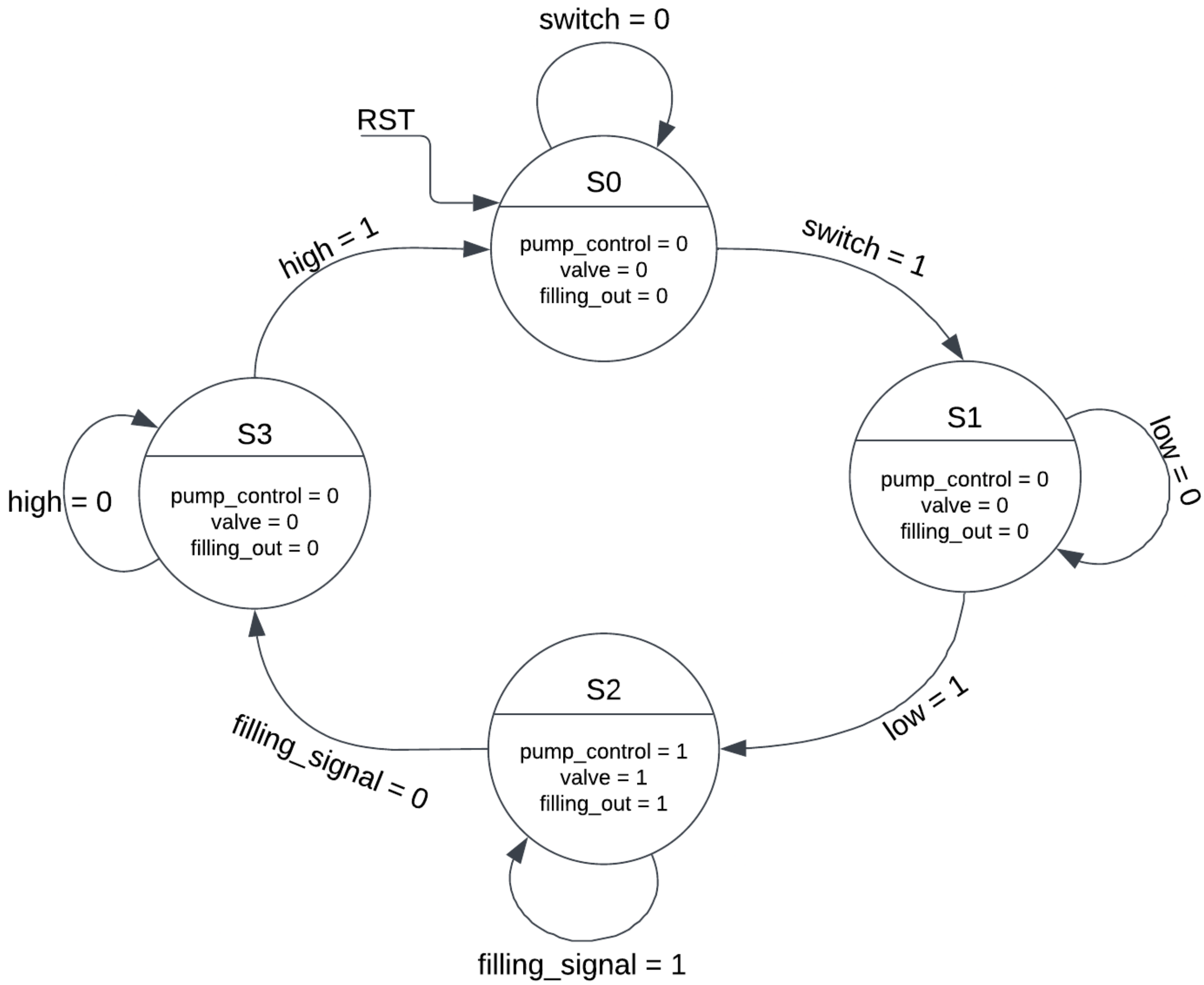


System Design

FSM (Finite State Machine) Design for Single Building

- The WPCU is controlled using a **finite state machine (FSM)** with multiple states to manage the pump and valves:
  1. **Idle (S0)**: The system waits for an ON/OFF switch signal.
  2. **Check Level (S1)**: The system checks water levels in the tanks.
  3. **Wait for Filling Signal (S2)**: Wait until system finish.
  4. **Pump Water (S3):** Water is pumped into the selected tank.

State Diagram



- Leakage sensor directly connected to the reset and warning indicator, so those sensors behave as a reset signal to the system

State Transition Table for Single Building

State	Description	Code
S0	Waiting for switch on (Idle)	00
S1	Check water level	01
S2	Waiting for filling progress flag	10
S3	Pump water	11

State Transition Table for Single Building

Q_n	Qn^+	T
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table

Inputs						Outputs						
switch	low	filling	high	S_0	S_1	S_0^+	T_0	S_1^+	T_1	pump	valve	filling_out
0	x	x	x	0	0	0	0	0	0	0	0	0
1	x	x	x	0	0	0	0	1	1	0	0	0
x	0	x	x	0	1	0	0	1	0	0	0	0
x	1	x	x	0	1	1	1	0	1	0	0	0
x	x	0	x	1	0	1	0	1	1	0	0	0
x	x	1	0	1	1	1	0	1	0	1	1	1
x	x	x	1	1	1	0	1	0	1	0	0	0

Equations for Single Building

Pump, Valve & filling flag

$$pump/valve/filling\_out \ = high \cdot S_0 \cdot S_1$$

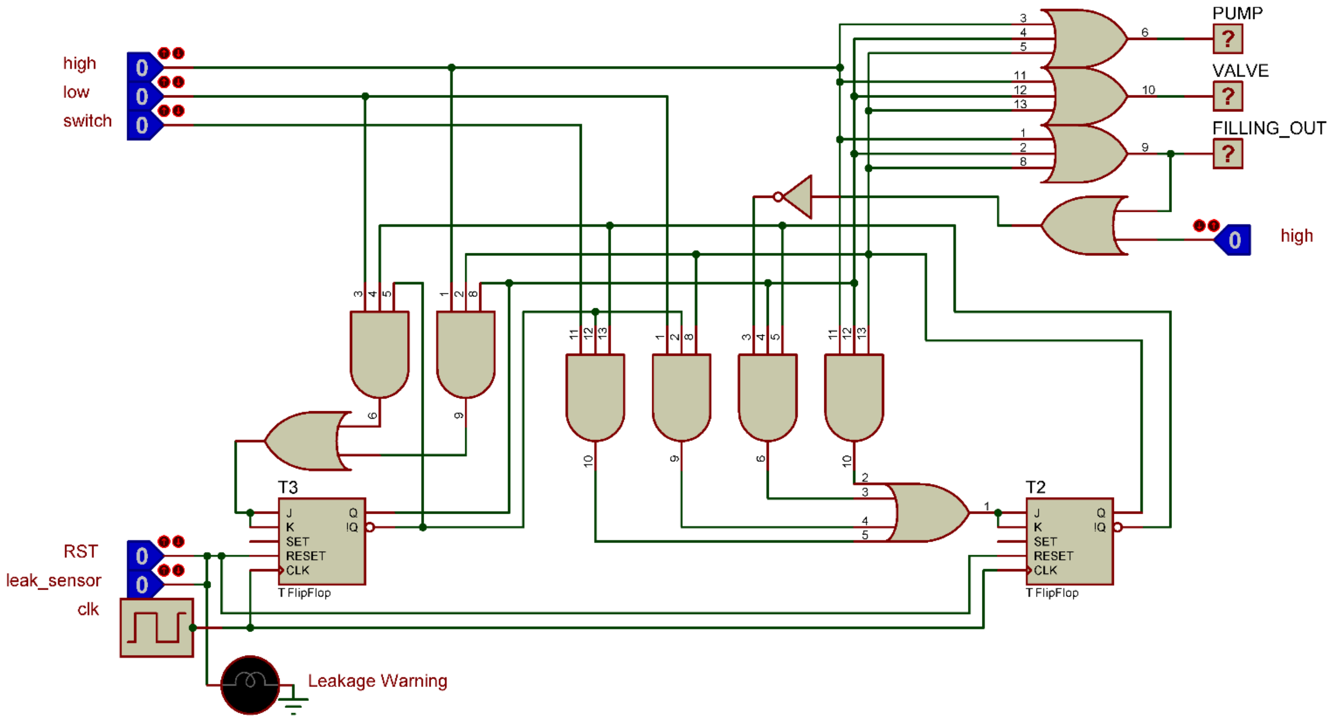
T-Flipflop 0

$$T_0 = low \cdot \overline{S_0} \cdot S_1 + high \cdot S_0 \cdot S_1$$

T-Flipflop 01

$$T_1 = switch \cdot \overline{S_0} \cdot \overline{S_1} + low \cdot \overline{S_0} \cdot S_1 + \overline{filling} \cdot S_0 \cdot \overline{S_1} + high \cdot S_0 \cdot S_1$$

Logic Circuit for Single Building



Block Implementation

- **Key blocks:**
  - Water level sensor control
  - Pump control logic
  - Signal filling progress
  - Solenoid valve control

VHDL Code for FSM (Single Building)

```
entity FSM is
  port (
    clk              : in  std_logic;
    reset            : in  std_logic;
    leak_sensor      : in  std_logic;
    pump_control     : inout std_logic;
    high             : in  std_logic;
    low              : in  std_logic;
    switch           : in  std_logic;
    valve            : inout std_logic;
--    filling_in      : in std_logic;
    filling_out      : out std_logic

  );
end FSM;

architecture Behavioral of FSM is

    -- States for building FSM
    type state_type is (S0, S1, S2, S3);
    signal current_state, next_state : state_type;
    signal pump_enable               : std_logic;
    signal filling_signal            : std_logic;

begin

    -- Leak detection logic: Reset if any leak detected
    process (leak_sensor)
    begin
        if (leak_sensor = '1') then
            pump_enable <= '0'; -- Disable pump if any leak detected
        else
            pump_enable <= '1'; -- Enable pump if no leaks detected
        end if;
    end process;
```

```
-- Motor control logic: Motor is enabled if motor_enable is '1'
process (pump_enable)
begin
    if (pump_enable = '1')  then
        pump_control <= '1';
    else
        pump_control <= '0';
    end if;
end process;


-- Signal assignments
filling_signal <= pump_control;
filling_out <= filling_signal;


-- FSM -----------------------------------------------

--Clock & Reset
process (clk, reset, leak_sensor)
begin
    if (reset = '1' or leak_sensor = '1') then
        current_state <= S0;
    elsif rising_edge(clk) then
        current_state <= next_state;
    end if;
end process;

--State transitions
process (current_state, high, low, switch, filling_signal)
begin
    case current_state is

    -- State 00
        when S0 =>
            valve <= '0';
            pump_enable <= '0';
            if switch = '1' then
                next_state <= S1;
            else
                next_state <= S0;
            end if;
```

```vhdl
        -- State 01
            when S1 =>
                valve <= '0';
                pump_enable <= '0';
                if low = '1' then
                    next_state <= S2;
                else
                    next_state <= S1;
                end if;

        -- State 02
            when S2 =>
                next_state <= S3;
                valve <= '0';
                pump_enable <= '0';
                if filling_signal = '0' then
                    -- No state change
                else
                    next_state <= S2;
                end if;

        -- State 03
            when S3 =>
                next_state <= S0;
                valve <= '1';
                pump_enable <= '1';
                if high = '1' then
                    -- Transition back to S0
                else
                    next_state <= S3;
                end if;

            when others =>
                next_state <= S0;
        end case;
    end process;

end Behavioral;
```

Testing and Verification

Test Strategy

- Simulate the FSM behavior using a VHDL testbench.
- Test cases include:
  - Activating the ON/OFF switch to start the pump.
  - Simulating water sensor signals for empty and full tanks.
  - Simulating leakage detection and the system's response.

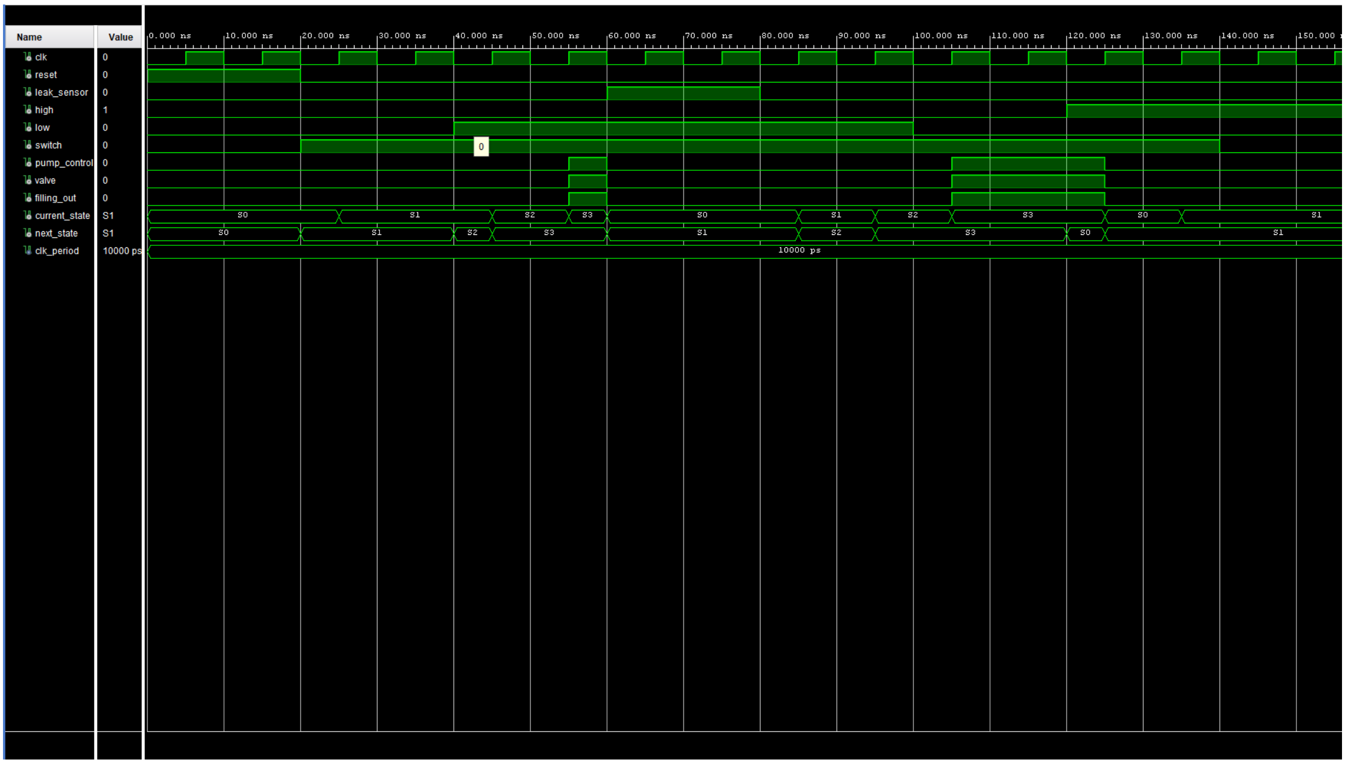
VHDL Testbench Code

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

entity FSM_tb is
end FSM_tb;

architecture tb of FSM_tb is
    signal clk, reset, leak_sensor : std_logic := '0';
    signal high, low, switch : std_logic := '0';
    signal pump_control, valve : std_logic;
    signal filling_out          :  std_logic;

    constant clk_period : time := 10 ns;
begin
    -- Instantiate FSM
    uut: entity work.FSM
        port map (
            clk => clk,
            reset => reset,
            leak_sensor => leak_sensor,
            pump_control => pump_control,
            high => high,
            low => low,
            switch => switch,
            valve => valve,
            filling_out => filling_out
        );

    -- Clock process
    clk_process : process
    begin
        clk <= '0';
        wait for clk_period / 2;
        clk <= '1';
        wait for clk_period / 2;
    end process;

    -- Test process
    test_process : process
    begin
```

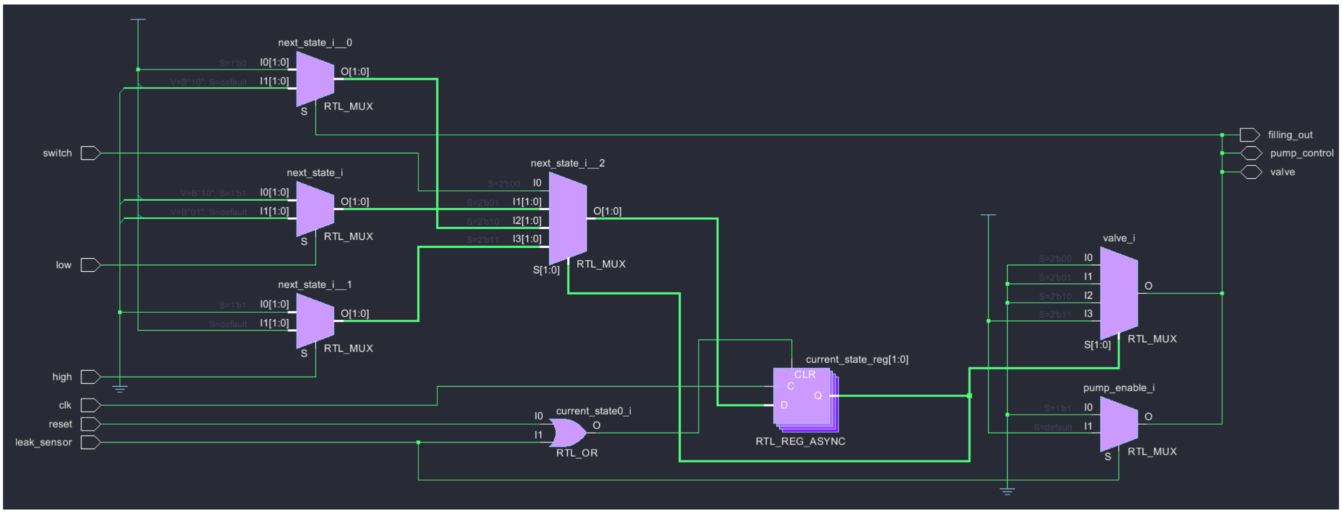
```
        -- Initial state
        reset <= '1';
        leak_sensor <= '0';
        switch <= '0';
        low <= '0';
        high <= '0';

        -- Reset the system
        wait for 20 ns;
        reset <= '0';

        -- Test Case 1: Switch pressed, no leak
        switch <= '1';
        wait for 20 ns;

        -- Test Case 2: Low signal active, pump should on
        low <= '1';
        wait for 20 ns;

        -- Test Case 3: Leak detected, pump should be off
        leak_sensor <= '1';
        wait for 20 ns;

        -- Test Case 4: Leak cleared, pump should start again
        leak_sensor <= '0';
        wait for 20 ns;

        -- Test Case 5: High signal active, pump should be off
        low <= '0';
        wait for 20 ns;
        high <= '1';  -- Simulate high signal being active
        wait for 20 ns;

        -- Test Case 6: Switch off and wait for transition to S0
        switch <= '0';
        wait for 40 ns;


        wait;
    end process;

end tb;
```

Simulation Results

Timing Diagram



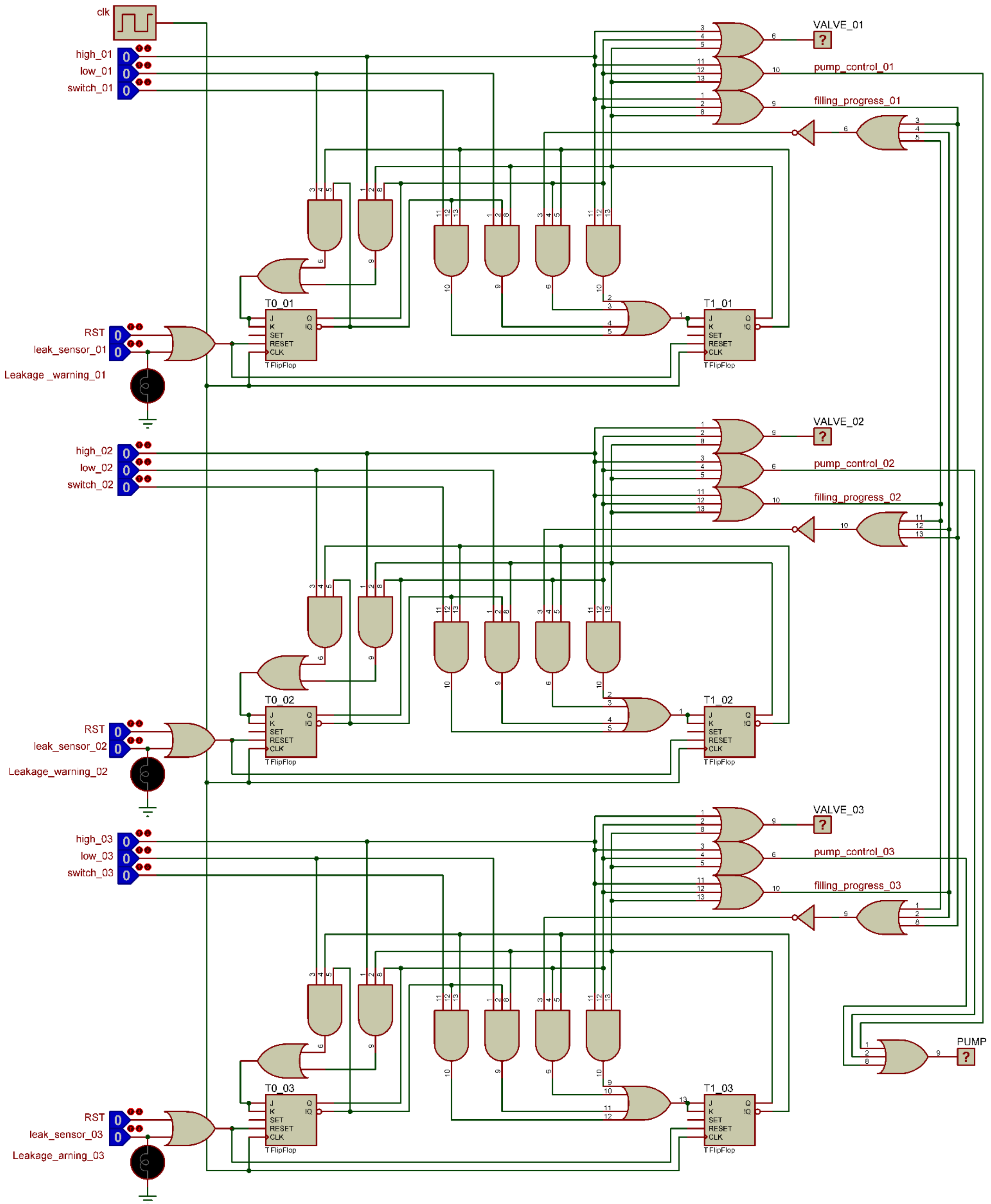
RTL Schematic



System Implementation

Complete Logic Circuit

Water is pumped from the well to one tank at a time until the tank is full by taking the filling_progress signal which is goes high when a valve is opened at that time filling_progress signal goes high and then no other valve can open until signal goes low from that this WPCU pump water to one tank at a time.



Port Mapping

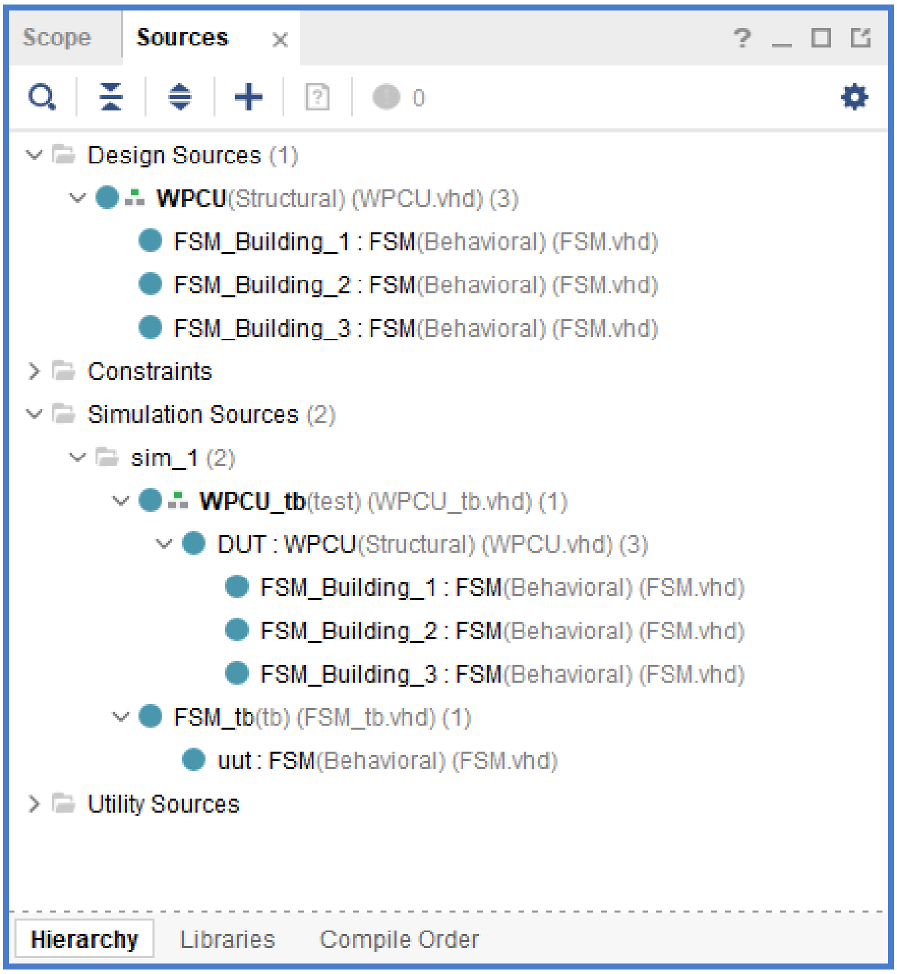
- In this system all 3 leakage sensors are acting as a reset signal to the system and there are indicators in each building and those are directly connected to the respective leakage sensor. Pump control signal is forced low until the leakage is fixed.

- There are three separate FSMs which are connected to a high-level model and pump_control signals of each FSM directed through a OR gate and connected to Pump Module.

- Leackage_signals are local to the respective FSM and when there is a leak in a particular building that FSM reset to state S0 and stays there until the leakage is fixed.

- RST is the global reset signal and using that we can reset complete system and until RST goes low system stays in state S0.

VHDL Code Hierarchy



VHDL Code for Complete System

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


entity WPCU is
  port (
    CLK                                                     : in  std_logic;
    RST                                                     : in  std_logic;
    leak_sensor_01, leak_sensor_02, leak_sensor_03          : in  std_logic;
    pump_control                                            : inout std_logic;
    high_01, high_02, high_03                               : in  std_logic;
    low_01, low_02, low_03                                  : in  std_logic;
    switch_01, switch_02, switch_03                         : in  std_logic;
    valve_01, valve_02, valve_03                            : inout std_logic;
    filling_out                                             : out std_logic

  );
end WPCU;

architecture Structural of WPCU is
    -- Signals for the output of each FSM
    signal V1, V2, V3, Fo1, Fo2, Fo3, P1, P2, P3 : STD_LOGIC;

    -- Component declaration for FSM
    component FSM
        Port (
            clk             : in  std_logic;
            reset           : in  std_logic;
            leak_sensor     : in  std_logic;
            pump_control    : inout std_logic;
            high            : in  std_logic;
            low             : in  std_logic;
            switch          : in  std_logic;
            valve           : inout std_logic;
            filling_out     : out std_logic
        );
    end component;
begin
    -- Instantiating FSM for Building 01
```

```vhdl
    FSM_Building_1 : FSM
        port map (
            clk => CLK,
            reset => RST,
            leak_sensor => leak_sensor_01,
            pump_control => P1,
            high => high_01,
            low => low_01,
            switch => switch_01,
            valve => V1,
            filling_out => Fo1
        );

    -- Instantiating FSM for Building 02
    FSM_Building_2 : FSM
        port map (
            clk => CLK,
            reset => RST,
            leak_sensor => leak_sensor_02,
            pump_control => P2,
            high => high_02,
            low => low_02,
            switch => switch_02,
            valve => V2,
            filling_out => Fo2
        );

    -- Instantiating FSM for Building 03
    FSM_Building_3 : FSM
        port map (
            clk => CLK,
            reset => RST,
            leak_sensor => leak_sensor_03,
            pump_control => P3,
            high => high_03,
            low => low_03,
            switch => switch_03,
            valve => V3,
            filling_out => Fo3
        );
```

```vhdl
    -- Assign internal signals to output ports
    pump_control <= P1 or P2 or P3;  -- Shared pump control signal
    valve_01 <= V1;
    valve_02 <= V2;
    valve_03 <= V3;
    filling_out <= Fo1 or Fo2 or Fo3; -- Combined filling status for all buildings


end Structural;
```

Testing and Verification

Test Strategy

- Simulate the FSM behavior using a VHDL testbench.
- Test cases include:
    - Activate the ON/OFF switch to start the pump.
    - Simulating water sensor signals for empty and full tanks.
    - Simulating leakage detection and the system's response and reset.

VHDL Testbench Code

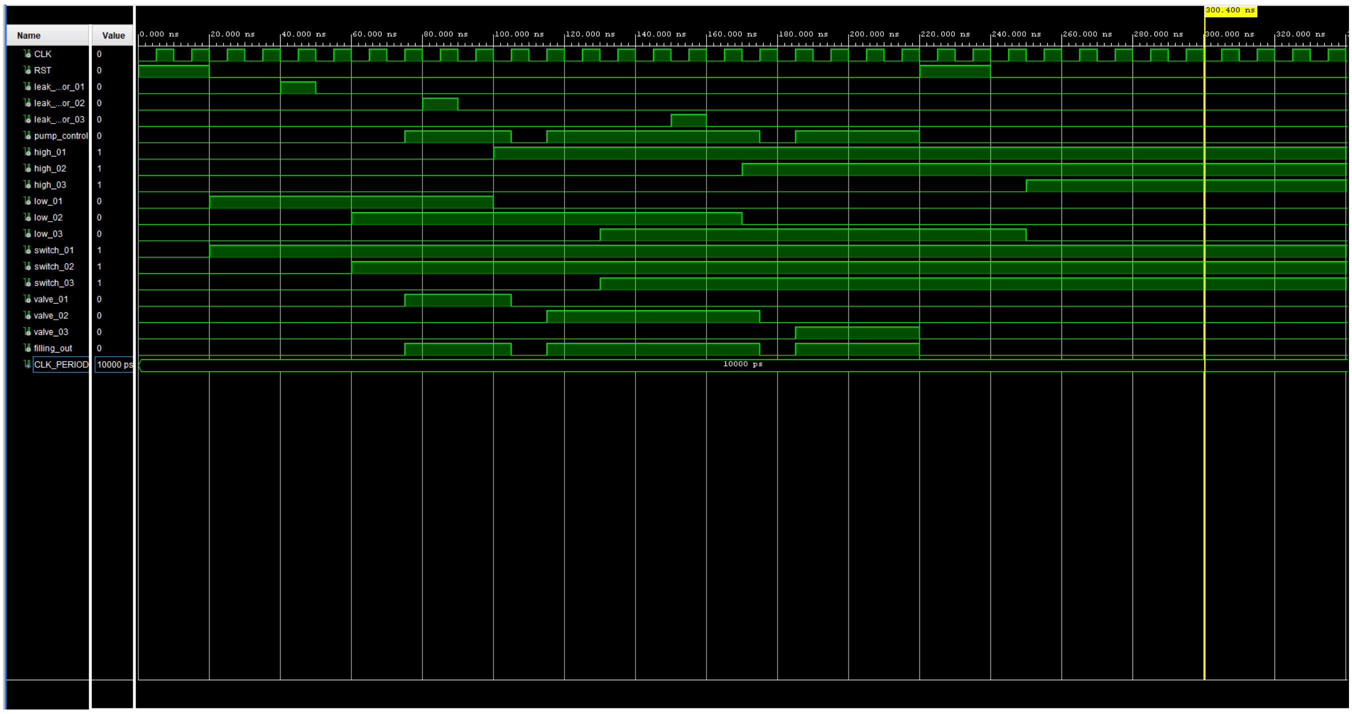
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity WPCU_tb is
end WPCU_tb;

architecture test of WPCU_tb is

    -- Component Under Test
    component WPCU
        port (
            CLK                                             : in  std_logic;
            RST                                             : in  std_logic;
            leak_sensor_01, leak_sensor_02, leak_sensor_03  : in  std_logic;
            pump_control                                    : inout std_logic;
            high_01, high_02, high_03                       : in  std_logic;
            low_01, low_02, low_03                          : in  std_logic;
            switch_01, switch_02, switch_03                 : in  std_logic;
            valve_01, valve_02, valve_03                    : inout std_logic;
            filling_out                                     : out std_logic
        );
    end component;

    -- Signals to connect to the WPCU instance
    signal CLK                                             : std_logic := '0';
    signal RST                                             : std_logic := '0';
    signal leak_sensor_01, leak_sensor_02, leak_sensor_03  : std_logic := '0';
    signal pump_control                                    : std_logic;
    signal high_01, high_02, high_03                       : std_logic := '0';
    signal low_01, low_02, low_03                          : std_logic := '0';
    signal switch_01, switch_02, switch_03                 : std_logic := '0';
    signal valve_01, valve_02, valve_03                    : std_logic;
    signal filling_out                                     : std_logic;

    -- Clock period definition
    constant CLK_PERIOD : time := 10 ns;
```

```vhdl
begin
    -- Instantiate the WPCU (DUT)
    DUT: WPCU
        port map (
            CLK             => CLK,
            RST             => RST,
            leak_sensor_01 => leak_sensor_01,
            leak_sensor_02 => leak_sensor_02,
            leak_sensor_03 => leak_sensor_03,
            pump_control    => pump_control,
            high_01         => high_01,
            high_02         => high_02,
            high_03         => high_03,
            low_01          => low_01,
            low_02          => low_02,
            low_03          => low_03,
            switch_01       => switch_01,
            switch_02       => switch_02,
            switch_03       => switch_03,
            valve_01        => valve_01,
            valve_02        => valve_02,
            valve_03        => valve_03,
            filling_out     => filling_out
        );

    -- Clock generation
    CLK_process :process
    begin
        CLK <= '0';
        wait for CLK_PERIOD/2;
        CLK <= '1';
        wait for CLK_PERIOD/2;
    end process;

    -- Test process
    test_process: process
    begin
        -- Initialize signals
        RST <= '1';
        wait for 20 ns;
        RST <= '0';
```

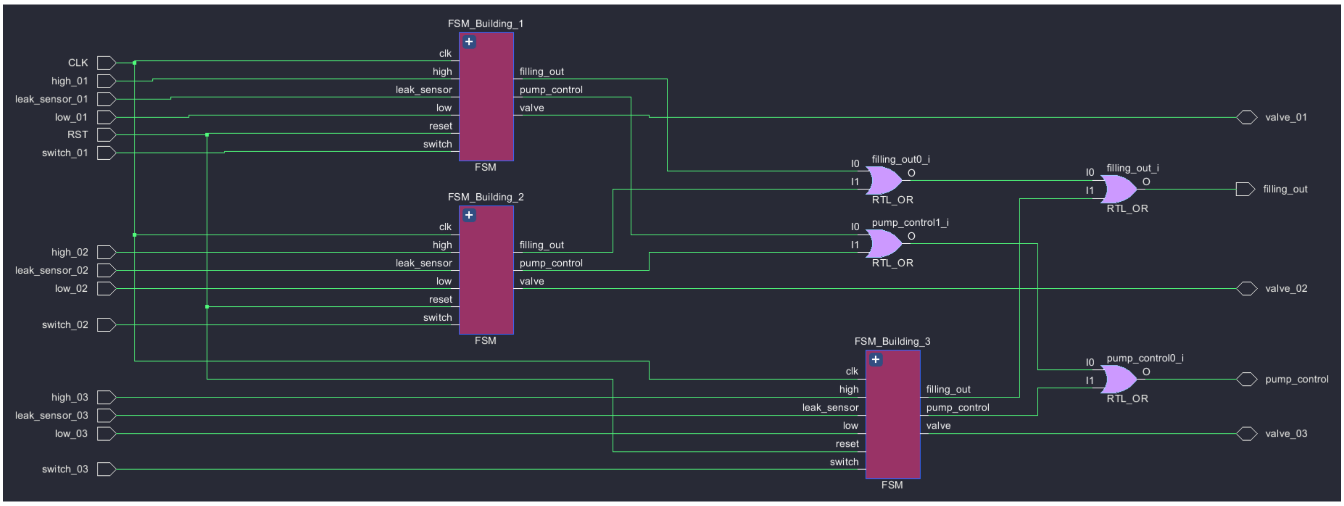
```vhdl
        -- Test Case 1: Start filling Building 1
        switch_01 <= '1'; -- Enable filling for Building 1
        high_01 <= '0';
        low_01 <= '1';    -- Low signal active to start filling
        wait for 20 ns;

        -- Leakage Test 1: Simulate leak in Building 1 during filling
        leak_sensor_01 <= '1'; -- Leak detected in building 01
        wait for 10 ns;
        leak_sensor_01 <= '0';
        wait for 10 ns;

        -- Test Case 2: Start filling Building 2 while Building 1 is still filling
        switch_02 <= '1';
        high_02 <= '0';
        low_02 <= '1';    -- Low signal active to start filling
        wait for 20 ns;

        -- Leakage Test 2: Simulate leak in Building 2 during filling
        leak_sensor_02 <= '1'; -- Leak detected in building 02
        wait for 10 ns;
        leak_sensor_02 <= '0';
        wait for 10 ns;

        -- Test Case 3: Stop filling Building 1 and start filling Building 2
        low_01 <= '0';
        high_01 <= '1'; -- High signal to stop filling Building 1
        wait for 10 ns;

        low_02 <= '1';  -- Low signal active for Building 2
        wait for 20 ns;

        -- Test Case 4: Start filling Building 3 while Building 2 is still filling
        switch_03 <= '1';
        high_03 <= '0';
        low_03 <= '1';    -- Low signal active to start filling
        wait for 20 ns;

        -- Leakage Test 3: Simulate leak in Building 3 during filling
        leak_sensor_03 <= '1'; -- Leak detected in building 03
        wait for 10 ns;
        leak_sensor_03 <= '0';
        wait for 10 ns;

        -- Test Case 5: Stop filling Building 2 and start filling Building 3
        low_02 <= '0';
        high_02 <= '1'; -- High signal to stop filling Building 2
        wait for 10 ns;

        low_03 <= '1';   -- Low signal active to start filling
        wait for 20 ns;

        -- Test Case 6: Reset test
        RST <= '1';     -- Trigger reset
        wait for 20 ns;
        RST <= '0';
        wait for 10 ns;

        -- End of test
        wait;
    end process;

end test;
```

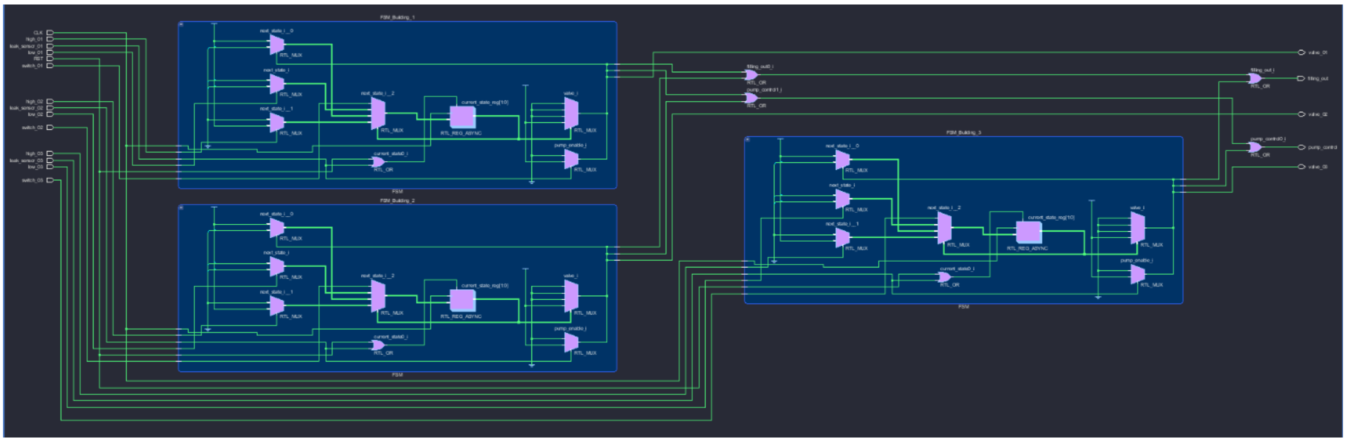
Simulation Results

Timing Diagram



RTL Schematic





Discussion

The Water Pump Control Unit (WPCU) project aimed to design a control system capable of managing water levels across three separate buildings, each with individual high, low, switch, and valve control signals. Additionally, the system included leakage sensors connected to a common motor control unit to ensure a centralized response to leak detection. To maintain water usage efficiency and ensure safety of the system, the system was designed to enforce a "filling status" condition, allowing only one tank to be filled at a time. This sequential filling process prevents overloading the pump system and ensures that all buildings receive the necessary water supply without interference or pressure issues. This system offers a practical solution for multi-building water management, ensuring optimal water levels, protection against leak-related damage, and controlled, efficient filling operations.

1. System Design and Architecture

The WPCU design used a finite state machine (FSM) as the primary control mechanism, with separate FSMs for each building's water level control. Each FSM monitors the high and low water levels, as well as a manual switch, to activate or deactivate the corresponding valve and pump. This separation of control per building ensures that water management is fit to the specific needs of each structure. The use of a single pump for all buildings also optimized the design, reducing hardware requirements and power requirement while maintaining effective water control.

Additionally, the inclusion of a global leakage detection system is a critical feature that enhances the safety and strength of the WPCU. By connecting all leakage sensors to a single reset and pump enable signal, the system can immediately disable the pump if a leak is detected in any of the three buildings. This centralized response mechanism prevents water wastage and mitigates potential water damage.

2. Challenges Faced

One of the primary challenges in developing the WPCU was ensuring that the FSM logic correctly handled simultaneous control across three buildings without interfering with one another. The design needed to account for scenarios where one building might require pumping while another could experience a leak. Managing these interactions without causing errors in pump or valve activation required careful consideration of state transitions and priority assignments.

Another challenge was the coordination of signals from multiple sensors and switches to control a single motor. The design had to ensure that the motor was only activated when necessary and that it could handle demands from multiple buildings independently. Testing and debugging the VHDL code to manage these multi-building interactions was time-intensive but crucial for reliable operation.

3. System Functionality and Performance

During testing, the WPCU system successfully demonstrated its ability to control the water levels in each building independently. The FSMs for each building responded accurately to the high, low, and switch signals, and coordinating valves to fill one tank at a time activating the valves as needed. The global leakage detection features effectively disabled the pump when a leak was simulated, validating the safety mechanism's efficacy.

However, single motor control did lead to some limitations in responsiveness. For instance, when multiple buildings required pumping at the same time, the system had to prioritize activation based on the FSM logic. While this was generally effective, the use of a single pump might introduce a slight delay in water level adjustments across all buildings, especially during high-demand scenarios because when filling_progress register goes low FSM has to start from the very first state, so it took 3 clock cycles to start filling the next tank.

4. Potential Improvements

While the WPCU project met its primary objectives, there are several areas where future improvements could enhance the system's functionality and efficiency:

- **Distributed Pump System:** Implementing multiple pumps rather than a single shared motor could reduce response time and improve the system's ability to handle high-demand situations in multiple buildings simultaneously. This would ensure that each building receives sufficient water management support independently of others.
- **Leakage Sensor Accuracy and Fault Tolerance:** Enhancing the leakage detection system to include additional sensors or redundancy mechanisms could improve reliability. For example, integrating multiple leakage sensors per building will enhance fault tolerance.
- **Advanced Control Algorithms:** Integrating more advanced control algorithms, such as proportional-integral-derivative (PID) control, could provide smoother pump activation and valve adjustments. This would allow the WPCU to better manage water levels with finer control, reducing wear on mechanical components and increasing efficiency.
- **IoT Integration for Remote Monitoring:** Adding Internet of Things (IoT) capabilities to the WPCU would allow for remote monitoring and control, enabling building managers to oversee water levels, leaks, and pump activity from a central control panel or mobile application. This addition would provide real-time data insights, leading to more proactive water management.

Conclusion

The WPCU project provided a functional and cost-effective solution for multi-building water management, demonstrating the viability of FSM-based control systems in real-world applications. By addressing the identified challenges and implementing the suggested improvements, the WPCU can be further optimized to provide even more reliable, responsive, and efficient water management. This project highlights the potential for FSM-based automation in infrastructure management and sets the stage for future enhancements through advanced control techniques and IoT integration.

[1], [2], [3], [4], [5], [6]

Citations

[1] “Automata, Finite State Machine,” YouTube. Accessed: Nov. 15, 2024. [Online]. Available: http://www.youtube.com/playlist?list=PL3MjzjRBJNQFdaMtUDwQr0l53UNrBpExq

[2] “Beginners Guide to get started with Xilinx FPGA Programming,” YouTube. Accessed: Nov. 15, 2024. [Online]. Available: http://www.youtube.com/playlist?list=PLqOe1_kmWOx33G3gOzQSajSdrTtW9shBO

[3] “Digital Electronics,” YouTube. Accessed: Nov. 15, 2024. [Online]. Available: http://www.youtube.com/playlist?list=PL-NNXPHI3pPijJsZmx_jAitLx8yks1qnp

[4] “Introduction to FPGA,” YouTube. Accessed: Nov. 15, 2024. [Online]. Available: http://www.youtube.com/playlist?list=PLEBQazB0HUyT1WmMONxRZn9NmQ_9CIKhb

[5] “FPGA Design using VHDL Lectures - YouTube.” Accessed: Nov. 15, 2024. [Online]. Available: https://www.youtube.com/playlist?list=PLZv8x7uxq5XY-IQfQFb6mC6OXzz0h8ceF

[6] “VHDL Tutorial - YouTube.” Accessed: Nov. 15, 2024. [Online]. Available: https://www.youtube.com/playlist?list=PLEdaowO6UzNENeQ2WHyGC6mlmggnnhMD6