SMART MULTI-ZONE FIRE AND SMOKE DETECTION SYSTEM

VLSI PROJECT

AbdelRahman Mohamed Essam Ahmed

-AbdelRahman Moustafa Attia

-AbdelRahman Walid Morsy

-AbdelRahman Youssef Mahrous

-AbdelRahman Saad Edris

-AbdelRahman Samy Abdelhamed

-Abdullah Mohamed Ahmed Aly

-Youssef Ashraf Fathy

-Amr Hassan

-Khaled El Sayed

DR Mohamed EL-Bably

Table of Contents

Module Overview

The **FireSmokeDetector** module is a behavioral description written in VHDL, designed to monitor temperature and smoke levels across three distinct zones. It evaluates the sensor data against predefined thresholds, detects fire or smoke, and triggers appropriate alerts and actions based on the severity of the situation.

Design Components

Entity Declaration:

Defines the module's interface, including its inputs and outputs.

Key ports:

clk, rst: Clock and reset signals.

temp_data, smoke_data: 24-bit vectors holding 8-bit data for each zone.

fire_alert, smoke_alert: Output alerts for fire and smoke conditions.

critical_alert: Indicates a critical situation affecting the entire system.

send_fire_truck: Notifies fire truck dispatch for each zone.

siren_alarm, emergency_notification: Trigger siren and notify emergency services during critical conditions.

water, start_timer: Control outputs for firefighting and timing mechanisms.

Constants:

TEMP_THRESHOLD: 60°C (binary: "00111100") defines the critical temperature threshold.

SMOKE_THRESHOLD: Smoke level 48 (binary: "00110000") defines the critical smoke level threshold.

Finite State Machine (FSM):

The system uses an FSM with the following states:

READ_SENSORS: Initializes outputs and reads sensor data.

CHECK_ZONES: Compares sensor data with thresholds and determines alerts for each zone.

EVALUATE_CONDITIONS: Checks conditions for critical situations and activates responses.

CRITICAL_SITUATION: Takes necessary actions for critical conditions, including activating alarms and emergency notifications.

Signal Declarations:

pr_state, nxt_state: Hold the current and next state of the FSM.

alert_log: Internal signal to track alert status per zone.

Functional Description

Sensor Data Reading:

The READ_SENSORS state initializes system outputs to default and prepares the module for sensor data evaluation.

Zone Monitoring:

The CHECK_ZONES state evaluates temperature and smoke levels for each zone against the defined thresholds. If either condition exceeds its threshold, corresponding alerts are raised (fire_alert or smoke_alert).

Condition Evaluation:

The EVALUATE_CONDITIONS state determines if a critical situation exists. If a zone has both fire and smoke alerts, the system activates critical_alert and moves to the CRITICAL_SITUATION state. Otherwise, it continues monitoring.

Critical Response:

In the CRITICAL_SITUATION state:

Fire trucks are dispatched to affected zones (send_fire_truck).

The siren is activated (siren_alarm), and emergency services are notified (emergency_notification).

Key Features

Multi-Zone Monitoring: Supports monitoring three zones simultaneously.

Threshold Evaluation: Dynamically checks sensor data against predefined thresholds for fire and smoke.

Critical Alert Handling: Triggers coordinated responses in critical conditions, including alarms, notifications, and water sprinklers.

Finite State Machine (FSM): Provides structured control flow for state transitions and ensures efficient operation.

Code Highlights

Use of Loops:

Iterates through each zone to check sensor thresholds and set corresponding alerts:

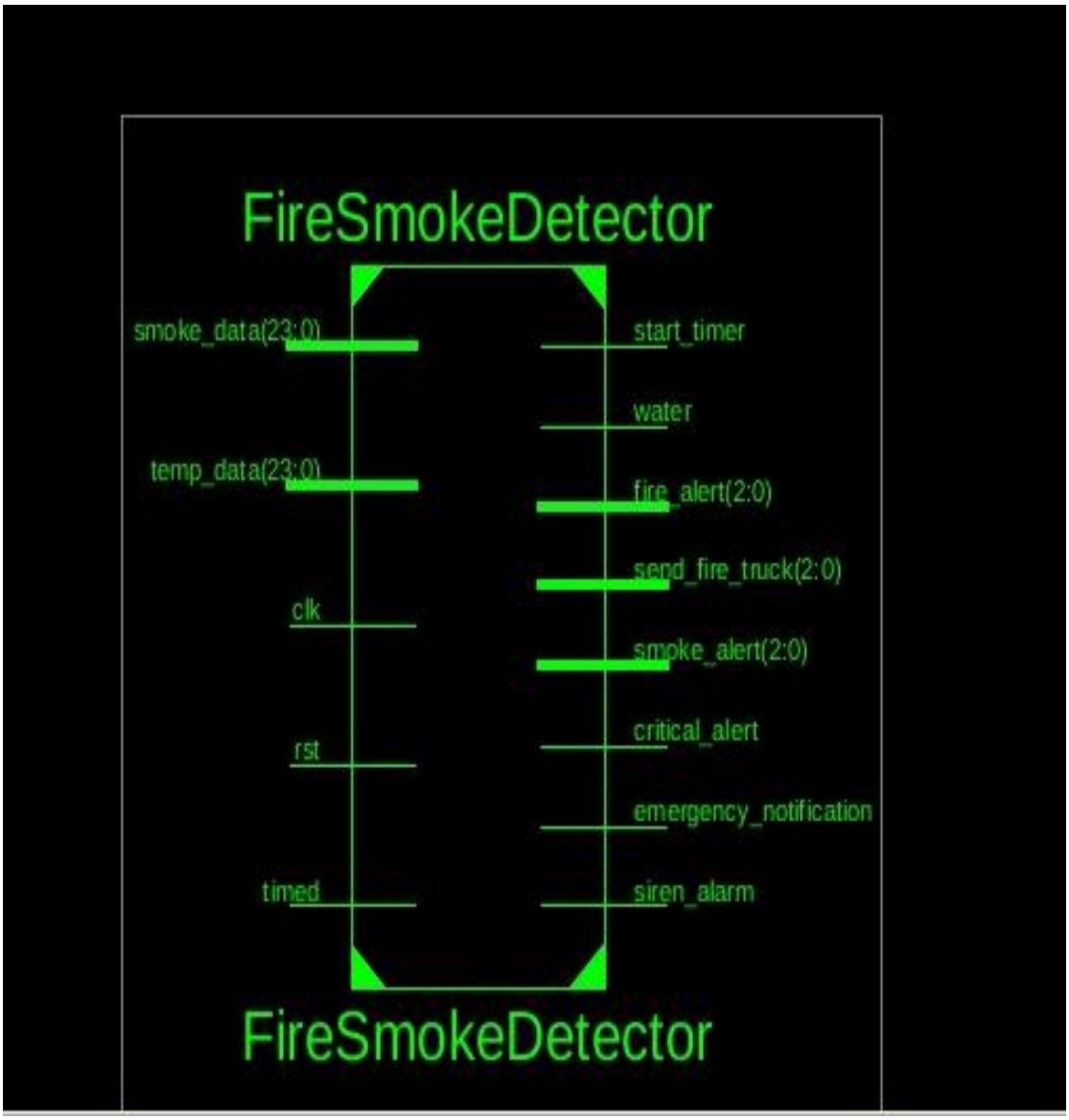
```
for i in 0 to 2 loop
  if temp_data((i+1)*8-1 downto i*8) >= TEMP_THRESHOLD then
    fire_alert(i) <= '1';
  else
    fire_alert(i) <= '0';
  end if;
end loop;
```

State Transition Logic:

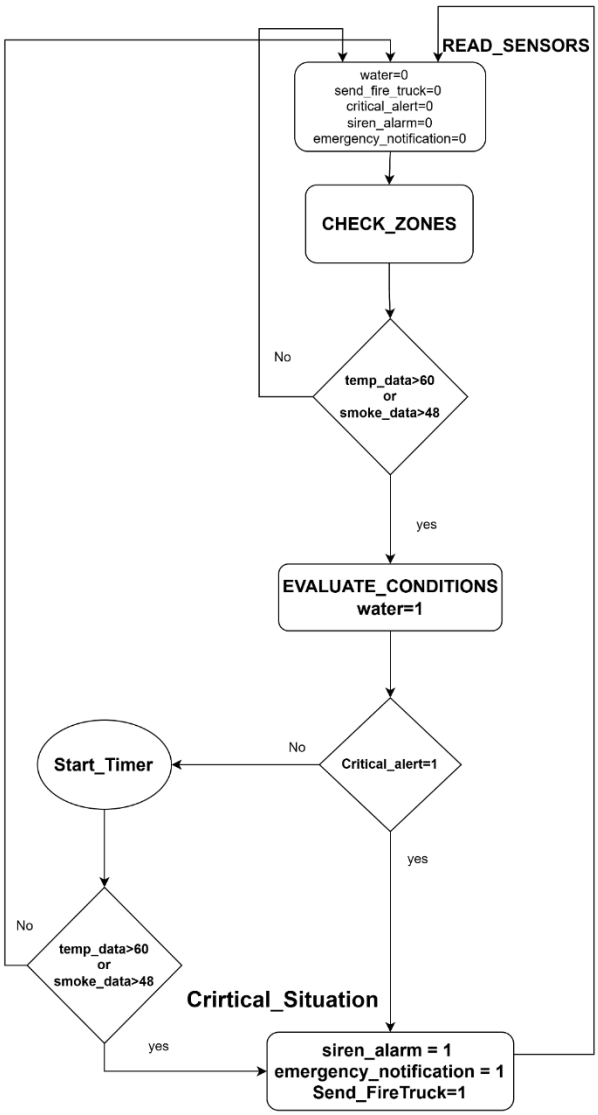
Efficiently transitions between states based on conditions:

```
case pr_state is
  when READ_SENSORS => nxt_state <= CHECK_ZONES;
  when CHECK_ZONES =>
    if ((fire_alert or smoke_alert) = "000") then
      nxt_state <= READ_SENSORS;
    else
      nxt_state <= EVALUATE_CONDITIONS;
    end if;
end case;
```

ASIC Chip


FireSmokeDetector
smoke_data(23:0)
temp_data(23:0)
clk
rst
timed
start_timer
water
fire_alert(2:0)
send_fire_truck(2:0)
smoke_alert(2:0)
critical_alert
emergency_notification
siren_alarm
FireSmokeDetector

ASM Chart:



The chart illustrates the sequence of operations and decisions in a fire and smoke detection system. It uses sensors to monitor temperature and smoke levels and takes appropriate actions when thresholds are exceeded. Below is a step-by-step breakdown:

Initial State: READ_SENSORS

The system initializes critical variables:

water = 0: Indicates the sprinkler system is off.

send_fire_truck = 0: No fire truck is dispatched.

critical_alert = 0: No critical alert is active.

siren_alarm = 0: Alarm siren is off.

emergency_notification = 0: Emergency notifications are inactive.

The system transitions to the next state to check sensor data.

CHECK_ZONES

The system reads data from sensors monitoring temperature (temp_data) and smoke levels (smoke_data).

Decision:

If temp_data ≥ 60 or smoke_data ≥ 48, it transitions to the next state.

Otherwise, it loops back to continuously monitor sensor data.

EVALUATE_CONDITIONS

If the condition is met, the water variable is set to 1, indicating the sprinkler system is activated.

Decision:

If the situation does not escalate further, no critical alert is raised, and the system continues monitoring.

Otherwise, it transitions to a critical state.

Start_Timer

A timer is activated to monitor if the situation persists or escalates further. The system evaluates the sensor data again:

If temp_data ≥ 60 or smoke_data ≥ 48 persists, it transitions to the critical situation state.

If not, it loops back for further monitoring.

Critical_Situation

If a critical condition persists:

siren_alarm = 1: The alarm system is activated.

emergency_notification = 1: Emergency notifications are sent to relevant authorities.

send_fire_truck = 1: A fire truck is dispatched to the location.

Key Features of the System:

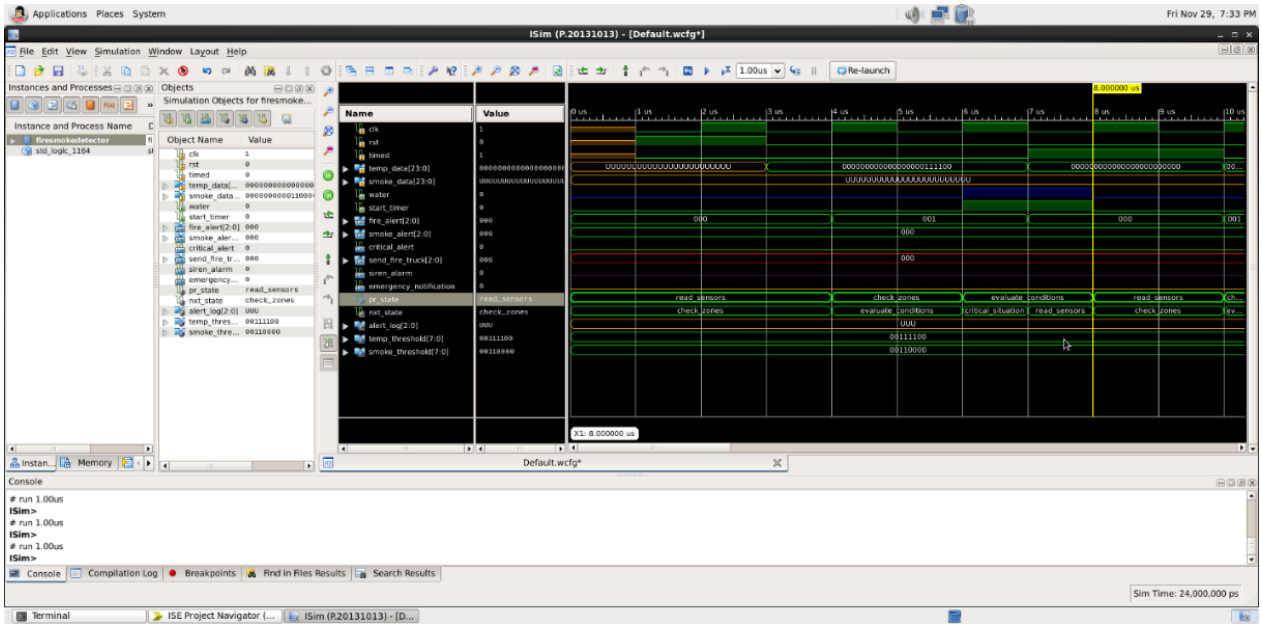
Early Detection: The system monitors both temperature and smoke levels to detect early signs of fire.

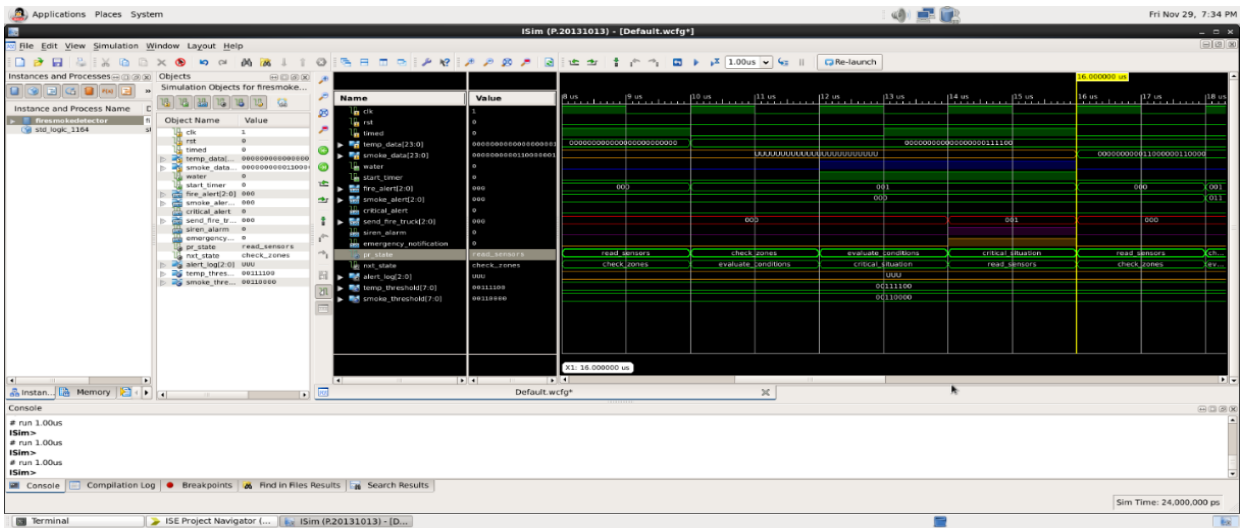
Automated Response: Activates sprinklers and sends alerts automatically upon detecting a potential fire.

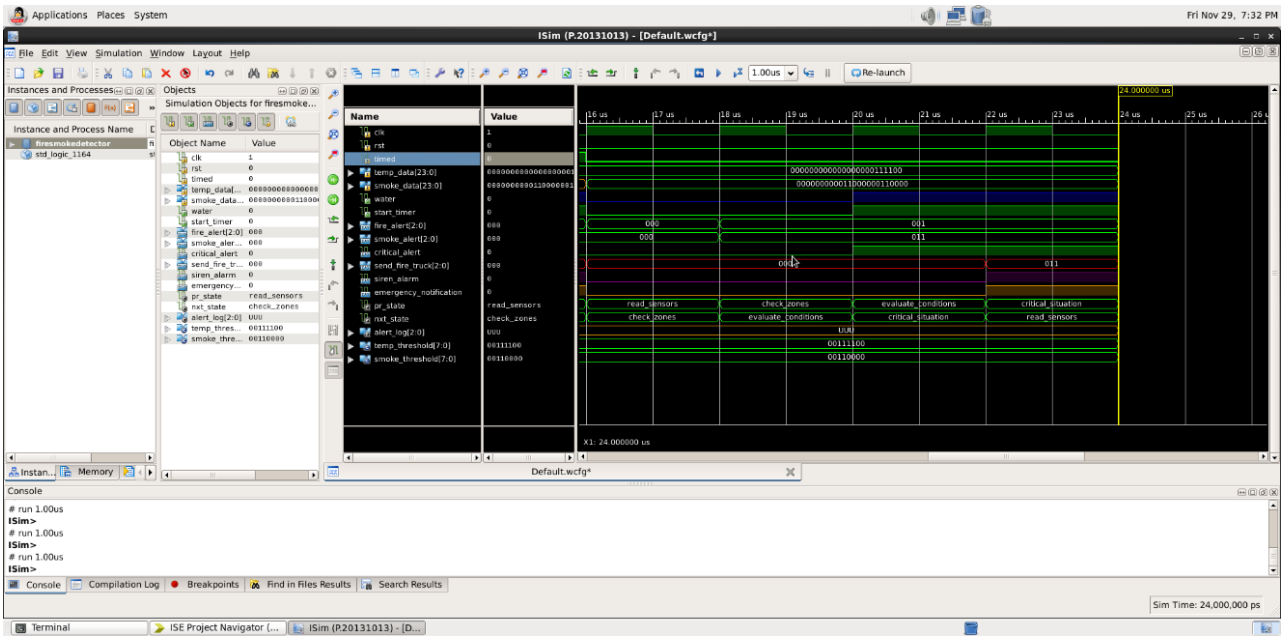
Escalation Mechanism: If the situation persists, the system transitions to a critical response mode, including siren activation and emergency dispatch.

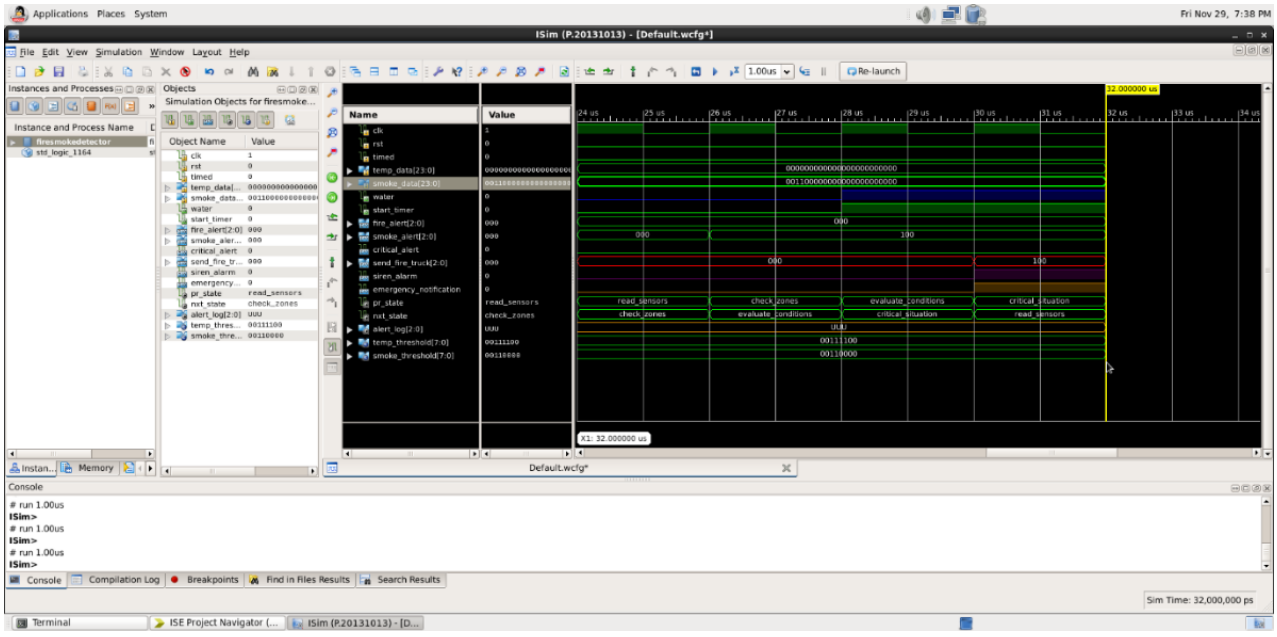
Safety and Mitigation: Aims to control the fire quickly and notify emergency services to minimize damage and ensure safety.

Simulation:






ISim (P.20131013) - [Default.wcfg*]
Console
run 1.00us
ISim>
run 1.00us
ISim>
run 1.00us
ISim>
Sim Time: 24,000,000 ps


ISim (P.20131013) - [Default.wcfg*]
Console
run 1.00us
ISim>
run 1.00us
ISim>
run 1.00us
ISim>
Sim Time: 32,000,000 ps

Full Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity FireSmokeDetector is

Port (
        clk                 : in  STD_LOGIC;
        rst                 : in  STD_LOGIC;
        timed               : in  std_logic;
        temp_data           : in  STD_LOGIC_VECTOR(23 downto 0); -- 8
bits per zone (3 zones)
        smoke_data          : in  STD_LOGIC_VECTOR(23 downto 0); -- 8
bits per zone (3 zones)
        water               : out std_logic;
        start_timer         : out std_logic;
        fire_alert          : inout STD_LOGIC_VECTOR(2 downto 0);  --
Fire alert for each zone
        smoke_alert         : inout STD_LOGIC_VECTOR(2 downto 0);  --
Smoke alert for each zone
        critical_alert      : inout STD_LOGIC;                     --
Global critical alert
        send_fire_truck     : inout STD_LOGIC_VECTOR(2 downto 0);   --
Fire truck priority signal for each zone
        siren_alarm         : inout STD_LOGIC;                     --
Siren alarm activation
        emergency_notification : inout STD_LOGIC                   --
Emergency services notification
    );
end FireSmokeDetector;

    architecture Behavioral of FireSmokeDetector is
    -- Constants for thresholds
    constant TEMP_THRESHOLD : STD_LOGIC_VECTOR(7 downto 0) :=
"00111100"; -- 60°C
    constant SMOKE_THRESHOLD: STD_LOGIC_VECTOR(7 downto 0) :=
"00110000"; -- Smoke level 48
     -- FSM states
    type state_type is ( READ_SENSORS, CHECK_ZONES,
EVALUATE_CONDITIONS,CRITICAL_SITUATION);
    signal pr_state, nxt_state : state_type;
            -- Internal signals
    signal alert_log                  : STD_LOGIC_VECTOR(2 downto
0); -- Log state per zone

    begin
    -- Sequential block for state transitions
        seq: process (clk)
    begin
        if rising_edge(clk) then
            if rst = '1' then
                pr_state <= READ_SENSORS;
            else
                pr_state <= nxt_state;
            end if;
        end if;
    end process;
            -- Combinational logic block
```

```
    comb: process (pr_state, temp_data, timed, smoke_data,
alert_log,fire_alert,critical_alert,smoke_alert,siren_alarm,emergency
_notification)
    begin
        -- Default outputs
                  case pr_state is


            -- Read sensors
        when READ_SENSORS =>
        fire_alert <= (others => '0');
        smoke_alert <= (others => '0');
        water<='0';
        start_timer<='0';
        critical_alert <= '0';
        send_fire_truck <= (others => '0');
        siren_alarm <= '0';  -- No siren by default
        emergency_notification <= '0';  -- No notification by default
        nxt_state <= CHECK_ZONES;


                        -- Check thresholds for each zone
            when CHECK_ZONES =>
                for i in 0 to 2 loop
                    if temp_data((i+1)*8-1 downto i*8) >=
TEMP_THRESHOLD then
                        fire_alert(i) <= '1';
                    else
                        fire_alert(i) <= '0';
                    end if;
                                                    if
smoke_data((i+1)*8-1 downto i*8) >= SMOKE_THRESHOLD then
                        smoke_alert(i) <= '1';
                     else
                        smoke_alert(i) <= '0';
                    end if;
                end loop;
                if ((fire_alert or smoke_alert)= "000") then
nxt_state <= READ_SENSORS;
                else
                nxt_state <= EVALUATE_CONDITIONS;
                end if;

                                        -- Evaluate alert conditions
            when EVALUATE_CONDITIONS =>
            water<='1';
                for i in 0 to 2 loop
                    if (fire_alert(i) = '1' And smoke_alert(i) = '1')
then
                        critical_alert  <= '1';
                    end if;


                end loop;
                if critical_alert = '1' then                nxt_state
<=critical_situation ;
                else start_timer<='1';
                end if;
                if timed='1' then
                for i in 0 to 2 loop
```

```
                    if temp_data((i+1)*8-1 downto i*8) >=
TEMP_THRESHOLD then
                        fire_alert(i) <= '1';
                    else
                        fire_alert(i) <= '0';
                    end if;
                                                    if
smoke_data((i+1)*8-1 downto i*8) >= SMOKE_THRESHOLD then
                        smoke_alert(i) <= '1';
                    else
                        smoke_alert(i) <= '0';
                    end if;
                end loop;
                end if;

                if ((fire_alert or smoke_alert)="000") then
nxt_state <= READ_SENSORS;
                else  nxt_state <=critical_situation ;
                end if;

                                    -- critical_situation  for each zone
and take action for critical alert
            when critical_situation =>

               for i in 0 to 2 loop
                        if fire_alert(i) = '1' or smoke_alert(i) =
'1' then
                             send_fire_truck(i) <= '1';
                        else
                             send_fire_truck(i) <= '0';
                        end if;
                end loop;
                                        -- Trigger siren and notify
emergency services if critical alert is active
                    siren_alarm <= '1';  -- Activate siren
                    emergency_notification <= '1';  -- Notify
emergency services

                nxt_state <= READ_SENSORS;

            -- Default case
            when others => nxt_state <= READ_SENSORS;
       end case;
    end process;



end Behavioral;
```

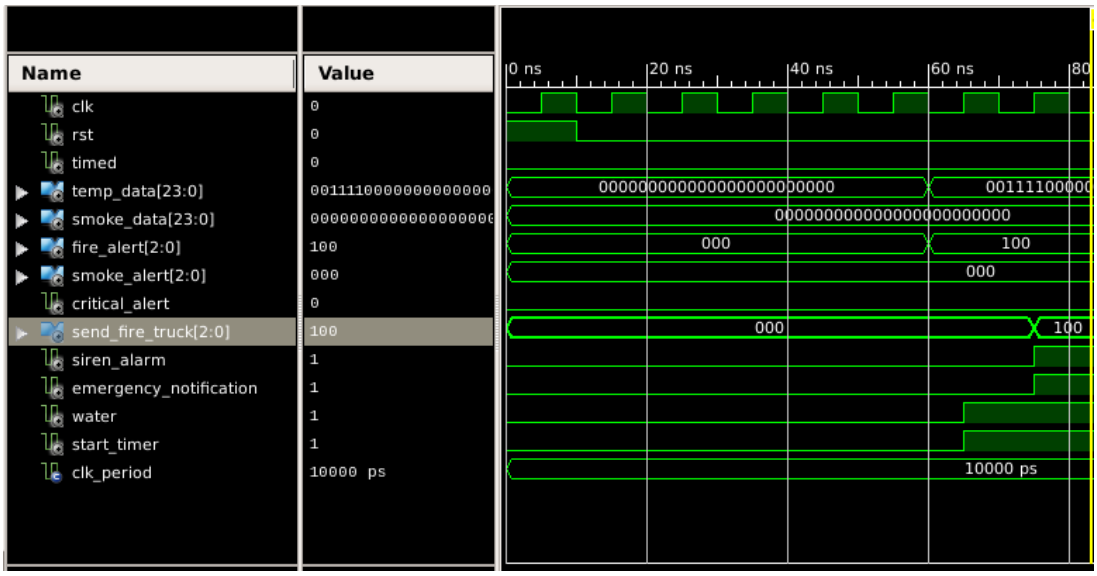
Testbench

TB Code:

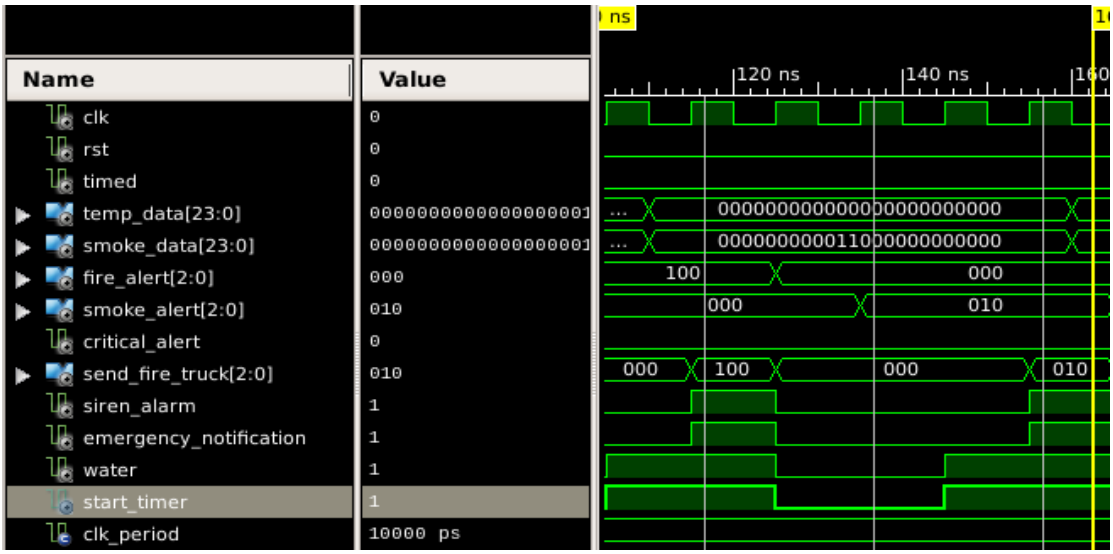
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY Testbench_FireSmokeDetector IS
END Testbench_FireSmokeDetector;

ARCHITECTURE behavior OF Testbench_FireSmokeDetector IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT FireSmokeDetector
    PORT(
         clk : IN  std_logic;
         rst : IN  std_logic;
         timed : IN  std_logic;
         temp_data : IN  std_logic_vector(23 downto 0);
         smoke_data : IN  std_logic_vector(23 downto 0);
         water : OUT  std_logic;
         start_timer : OUT  std_logic;
         fire_alert : INOUT  std_logic_vector(2 downto 0);
         smoke_alert : INOUT  std_logic_vector(2 downto 0);
         critical_alert : INOUT  std_logic;
         send_fire_truck : INOUT  std_logic_vector(2 downto 0);
         siren_alarm : INOUT  std_logic;
         emergency_notification : INOUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal clk : std_logic := '0';
   signal rst : std_logic := '0';
   signal timed : std_logic := '0';
   signal temp_data : std_logic_vector(23 downto 0) := (others => '0');
   signal smoke_data : std_logic_vector(23 downto 0) := (others => '0');

    --BiDirs
   signal fire_alert : std_logic_vector(2 downto 0);
   signal smoke_alert : std_logic_vector(2 downto 0);
   signal critical_alert : std_logic;
   signal send_fire_truck : std_logic_vector(2 downto 0);
   signal siren_alarm : std_logic;
   signal emergency_notification : std_logic;

    --Outputs
   signal water : std_logic;
   signal start_timer : std_logic;

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: FireSmokeDetector PORT MAP (
          clk => clk,
          rst => rst,
          timed => timed,
          temp_data => temp_data,
          smoke_data => smoke_data,
          water => water,
          start_timer => start_timer,
          fire_alert => fire_alert,
```

```
            smoke_alert => smoke_alert,
            critical_alert => critical_alert,
            send_fire_truck => send_fire_truck,
            siren_alarm => siren_alarm,
            emergency_notification => emergency_notification
          );

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


    -- Stimulus process
   stim_proc: process
     begin
         -- Reset the system
         rst <= '1';
         wait for CLK_PERIOD;
         rst <= '0';
           -- Test Case 1: No alerts, normal operation
         temp_data <= X"000000";  -- Low temperatures
         smoke_data <= X"000000"; -- Low smoke levels
         wait for 5 * CLK_PERIOD;
            -- Test Case 2: Fire in Zone 3
         temp_data <= X"3C0000";  -- Zone 3 above temperature threshold
         smoke_data <= X"000000"; -- No smoke
         wait for 5 * CLK_PERIOD;
           -- Test Case 3: Smoke in Zone 2
         temp_data <= X"000000";  -- Normal temperatures
         smoke_data <= X"003000"; -- Zone 2 above smoke threshold
         wait for 5 * CLK_PERIOD;

         -- Test Case 4: Fire and smoke in Zone 1 (critical alert)
         temp_data <= X"00003C";  -- Zone 1 above temperature threshold
         smoke_data <= X"000030"; -- Zone 1 above smoke threshold
         wait for 5 * CLK_PERIOD;

           -- Test Case 5: Return to normal
         temp_data <= X"000000";
         smoke_data <= X"000000";
         wait for 5 * CLK_PERIOD;

         -- Stop simulation
         wait;
     end process;

END;
```

Case1


Name
Value
clk
0
rst
0
timed
0
temp_data[23:0]
00111100000000000000
smoke_data[23:0]
00000000000000000000
fire_alert[2:0]
100
smoke_alert[2:0]
000
critical_alert
0
send_fire_truck[2:0]
100
siren_alarm
1
emergency_notification
1
water
1
start_timer
1
clk_period
10000 ps
0 ns
20 ns
40 ns
60 ns
80
00000000000000000000
00111100000
00000000000000000000
000
100
000
000
100
10000 ps

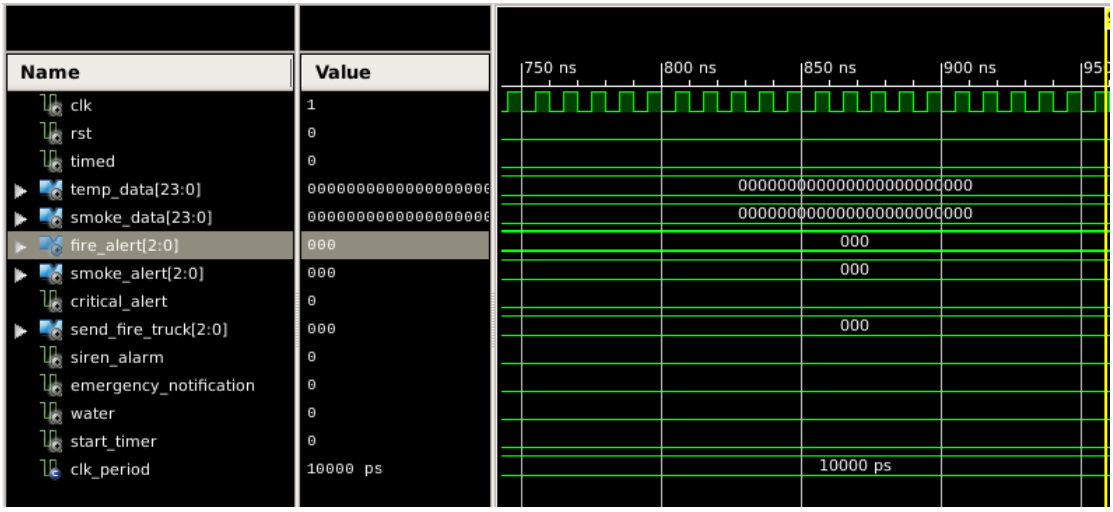
Case2


Name
Value
clk
0
rst
0
timed
0
temp_data[23:0]
00000000000000000001
smoke_data[23:0]
00000000000000000001
fire_alert[2:0]
000
smoke_alert[2:0]
010
critical_alert
0
send_fire_truck[2:0]
010
siren_alarm
1
emergency_notification
1
water
1
start_timer
1
clk_period
10000 ps
ns
120 ns
140 ns
00000000000000000000
00000000001100000000
100
000
000
010
000
100
000
010

Case3


Name
Value
clk
rst
timed
temp_data[23:0]
smoke_data[23:0]
fire_alert[2:0]
smoke_alert[2:0]
critical_alert
send_fire_truck[2:0]
siren_alarm
emergency_notification
water
start_timer
clk_period
0
0
0
001
001
1
001
1
1
1
1
10000 ps
180 ns
000
001

Case4


Name
Value
clk
rst
timed
temp_data[23:0]
smoke_data[23:0]
fire_alert[2:0]
smoke_alert[2:0]
critical_alert
send_fire_truck[2:0]
siren_alarm
emergency_notification
water
start_timer
clk_period
1
0
0
000
000
0
000
0
0
0
0
10000 ps
750 ns
800 ns
850 ns
900 ns
000
000
000
10000 ps

Conclusion:

The Fire and Smoke Detector system developed in this project provides an efficient and reliable solution for early fire detection and emergency response. Utilizing a finite state machine (FSM), the system monitors temperature and smoke levels across multiple zones, triggering appropriate actions such as activating sprinklers, generating alerts, and escalating to critical states when necessary. Features like fire truck notifications, siren activation, and emergency service alerts ensure a swift and comprehensive response in hazardous situations. This project demonstrates the effective integration of hardware and software, offering a scalable and practical approach to enhancing safety in residential, commercial, and industrial environments.

Project's Videos

https://youtu.be/mNVhdSyIqx4?si=kYYXfDDn-LVbhQ2r

https://youtu.be/DVT4KO2ii14?si=mZXTCNNvVMCR9FXa