Nama : Septian Bagus Jumantoro

Kelas : 1 D4 Teknik Komputer B

NRP : 3221600039

1. Source Code
   - **fire_alarm.vhd (Top Module)**

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

entity fire_alarm is
    port(   button: in std_logic;
            clr: in std_logic;
            clk: in std_logic;
            fire: in std_logic;
            sensor: out std_logic;
            alarm: out std_logic;
            hydrant: out std_logic;
            a_to_g : out STD_LOGIC_VECTOR (6 downto 0);
            an  : out STD_LOGIC_VECTOR (3 downto 0);
            dp  : out STD_LOGIC);
end fire_alarm;

architecture Behavioral of fire_alarm is
    component clkdiv is
            port( mclk : in std_logic;
                  clr : in std_logic;
                  clk48: out std_logic
                );
    end component;
    component x7segb is
```

```
                port ( x : in STD_LOGIC_VECTOR (15 downto 0);
                       clk : in STD_LOGIC;
                       clr : in STD_LOGIC;
                       a_to_g : out STD_LOGIC_VECTOR (6 downto
0);
                       an : out STD_LOGIC_VECTOR (3 downto 0);
                       dp : out STD_LOGIC );
        end component;
        type state_type is (disarm,arm,intrution);
        signal state: state_type;
        signal clk48: std_logic;
        signal x: std_logic_vector(15 downto 0);

begin

        u1: clkdiv port map(
                mclk => clk,
                clr => clr,
                clk48 => clk48
                );

        process(button, fire, clk48)
        begin
                if clr='1' then
                        state <= disarm;
                elsif clk48' event and clk48='1' then
                        case state is
                                when disarm =>          if button = '1' then state <= arm;
                                                                else state <=
disarm;
                                                                end if;
```

```
                    when arm =>   if button = '0' and fire = '1' then state <=
intrution;
                                  elsif button = '1' and fire =
'0' then state <= disarm;
                                  else state <= arm;
                                  end if;
                    when intrution =>       if button = '0' and fire = '1' then
state <= intrution;
                                                elsif button
= '1' and fire = '0' then state <= disarm;
                                                else state
<= intrution;
                                                end if;
                end case;
                sensor <= fire;
        end if;
    end process;

    process(state, fire, clk48)
    begin
        case state is
            when disarm => alarm <= '0'; hydrant <= '0'; x <=
"0100000100110011";
            when arm => alarm <= '0'; hydrant <= '0'; x <=
"0100010000010010";
            when intrution => if fire = '1' then alarm <= '1'; hydrant <= '1';
                                  else alarm <= '1';
hydrant <= '0';
                                  end if;
        end case;
    end process;

    u2 : x7segb port map
```

```
                    (x=>x, clk=>clk, clr=>clr, a_to_g=>a_to_g, an=>an, dp=>dp);

    end Behavioral;
```

- **clkdiv.vhd**

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

entity clkdiv is
Port (mclk : in  STD_LOGIC;
          clr : in STD_LOGIC;
          clk48 : out STD_LOGIC);
end clkdiv;

architecture Behavioral of clkdiv is
signal q:std_logic_vector(23 downto 0);
begin
    process(mclk, clr)
    begin
          if clr = '1' then
                q <= x"000000"; --format hexadesimal
          elsif mclk' event and mclk = '1' then
                q <= q + 1;
          end if;
          clk48 <= q(22);
    end process;
end Behavioral ;
```

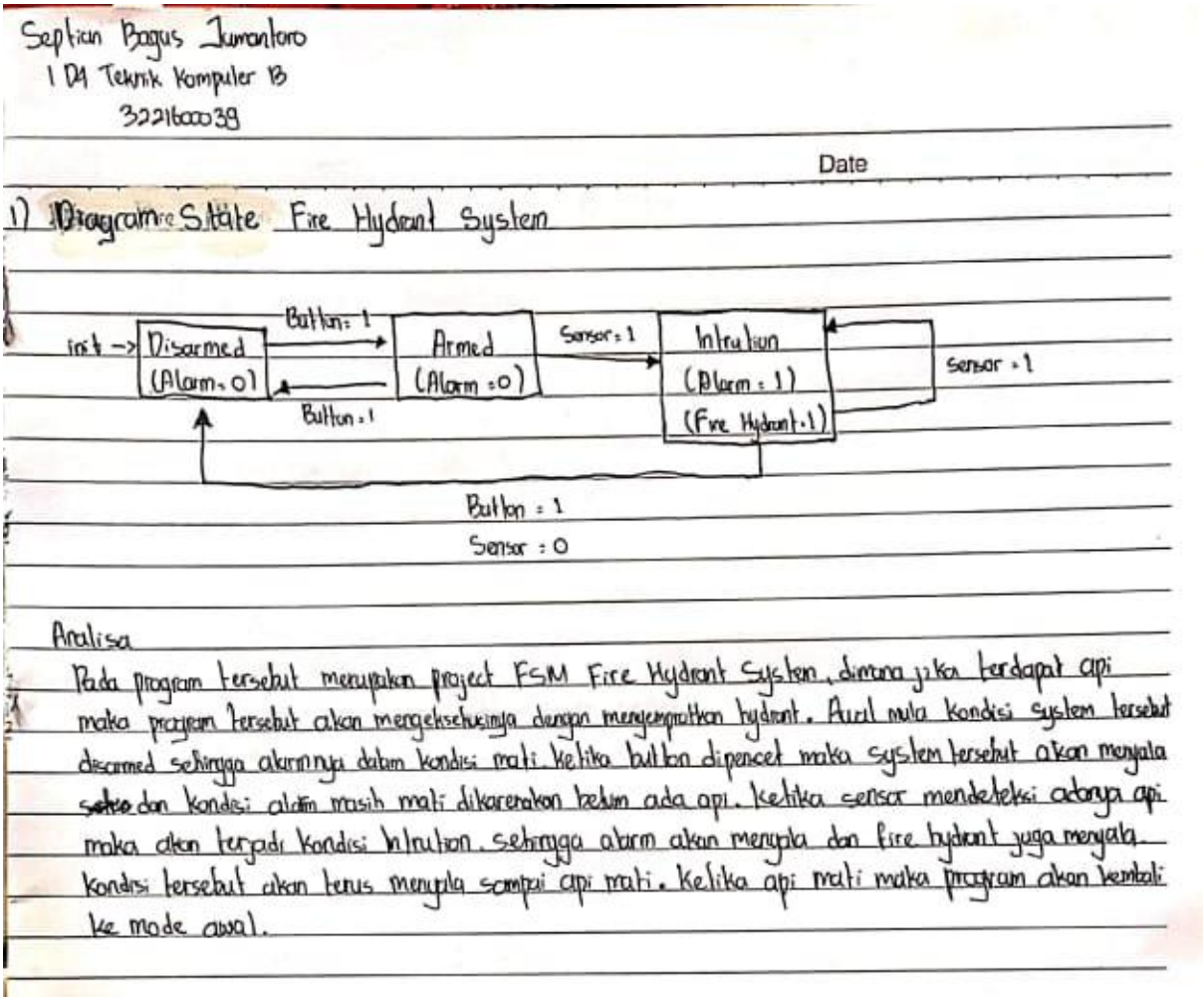
- **x7segb.vhd**

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

entity x7segb is
	port( x : in STD_LOGIC_VECTOR (15 downto 0);
			clk : in STD_LOGIC;
			clr : in STD_LOGIC;
			a_to_g : out STD_LOGIC_VECTOR (6 downto 0);
			an : out STD_LOGIC_VECTOR (3 downto 0);
			dp :  out STD_LOGIC);
end x7segb;

architecture x7seg of x7segb is
		signal s : STD_LOGIC_VECTOR (1 downto 0);
		signal digit : STD_LOGIC_VECTOR (3 downto 0);
		signal aen : STD_LOGIC_VECTOR (3 downto 0);
		signal clkdiv : STD_LOGIC_VECTOR (20 downto 0);
begin
		s <= clkdiv(20 downto 19);
		dp <= '1';
		aen <= "1111";

-- set aen(3 downton 0) for leading blenks
--		aen(3) <= x(15) or x(14) or x(13) or x(12);
--		aen(2) <= x(15) or x(14) or x(13) or x(12) or x(11) or x(10) or x(9) or x(8);
--		aen(1) <= x(15) or x(14) or x(13) or x(12) or x(11) or x(10) or x(9) or x(8) or
x(7) or x(6) or x(5) or x(4);
--		aen(0) <= '1'; -- digit 0 always on
```

```
-- quad 4 to 1 mux
		process(s,x)
		begin
			case s is
				when "00" => digit <= x(3 downto 0);
				when "01" => digit <= x(7 downto 4);
				when "10" => digit <= x(11 downto 8);
				when others => digit <= x(15 downto 12);
			end case;
		end process;

-- 7 segment decoder: hex7seg
		process(digit)
		begin
			case digit is
				when x"0" => a_to_g <= "0000001";
				when x"1" => a_to_g <= "0000001";
				when x"2" => a_to_g <= "0001001";
				when x"4" => a_to_g <= "1111111";
				when others  => a_to_g <= "0111000";
			end case;
		end process;

-- digit select: ancode
		process(s,aen)
		begin
			an <= "1111";
			if aen(conv_integer(s)) = '1' then
				an(conv_integer(s)) <= '0' ;
			end if;
```

```
        end process;

-- clock devider
        process(clk,clr)
        begin
                if clr = '1' then
                        clkdiv <=(others => '0');
                elsif clk' event and clk = '1' then
                        clkdiv <= clkdiv + 1 ;
                end if;
        end process;

end x7seg;
```

2. Analisa



3. Source Code Traffic Light

Date

2) Source Code:

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

entity traffic is port (
        clr: in STD_LOGIC;
        clk: in STD_LOGIC;
  lights: out STD_LOGIC_VECTOR (5 downto 0);
end traffic;
architecture Behavioral of traffic is type
state_type is (S0, S1, S2, S3, S4, S5);
signal
state: state_type;
Signal count: STD_LOGIC_VECTOR (3 downto 0); constant
SEC 5: STD_LOGIC_VECTOR (3 downto 0):= "1111"; constant
SEC 1: STD_LOGIC_VECTOR (3 downto 0); = "0011"; constant
Process (clr, clk)
    begin
   if clr = '1' then
   state <= S0;
   count <= x"0";
   else if if event and clk = '1' then
   case state is
   when S0 => if count < secs then
    state <= S0;
   count <= count + 1;
    else
    state <= S1;
   count <= x"0";
   end if;
   When S1 => if count < sec 1 then
    state <= S1;
   count <= count +1;
    else
   state <= S2;
   count <= x"0";
   end if;
```

KKY

Date

```
when S2 => if count < SEC1 then state <= S2;
count <= count+1;
else state <= S3;
count <= x"0"; end if;
when S3 => if count < SEC5 then state <= S3;
count <= count+1; else state <= S4;
count <= x"0"; else if;
when S4 => if count < SEC1 then state <= S4;
count <= count+1;
else state <= S5;
count <= x"0";
end if;
when S5 => if count < SEC1 then state <= S5;
count <= count+1;
else
state <= S0;
count <= x"0";
else if;
when others => state <= S0;
end case;
end if; end process;
C2: process (state) begin case state is
when S0 => lights <= "100001";
when S1 => lights <= "100010";
when S2 => lights <= "100100";
when S3 => lights <= "001100";
when S4 => lights <= "010100";
when S5 => lights <= "100100";
end case;
end process;
end behavioral;
```