1 D4 - TEKKOM B

FINITE STATE MACHINE

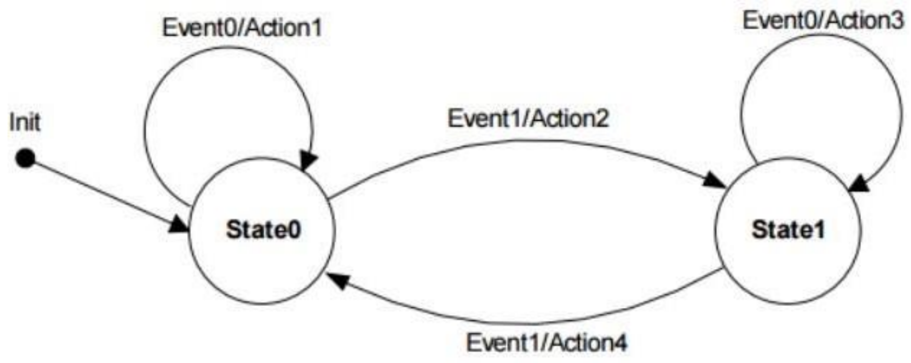
Nama	: Septian Bagus Jumantoro
Kelas	: 1 – D4 Teknik Komputer B
NRP	: 3221600039
Dosen	: Reni Soelistijorini B.Eng., MT.
Mata Kuliah	: Praktikum Rangkaian Logika 2
Hari,Tgl. Praktikum	: Jumat, 3 Juni 2022

1. **Tujuan**
    1. Menerapkan metode FSM dalam VHDL untuk sistem kendali pada simulasi trafic light
    2. Menampilkan simulasi sistem kendali pada simulasi trafic light untuk mengetahui aliran setiapperubahan state.

2. **Teori**

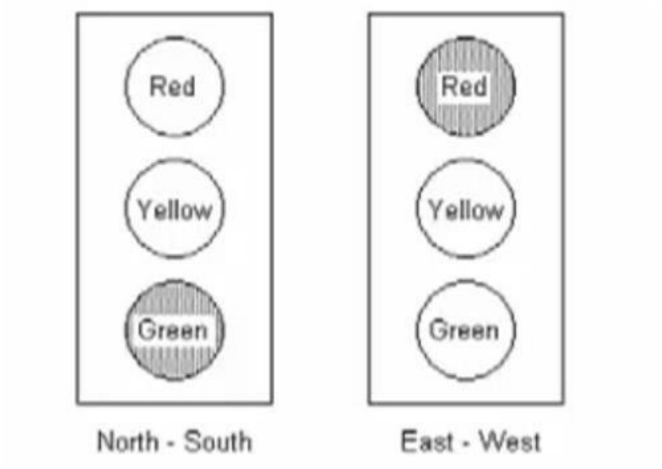
Finite State Machines (FSM) adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal berikut: State (Keadaan), Event (kejadian) dan action (aksi). Pada satu saat dalam periode waktu yang cukup signifikan, sistem akan berada pada salah satu state yang aktif. Sistem dapat beralih atau bertransisimenuju state lain jika mendapatkan masukan atau event tertentu, baik yang berasal dari perangkat luar atau komponen dalam sistemnya itu sendiri (misal interupsi timer). Transisi keadaan ini umumnya juga disertai oleh aksi yang dilakukan oleh sistem ketika menanggapi masukan yang terjadi. Aksi yang dilakukan tersebut dapat berupa aksi yang sederhana atau melibatkan rangkaian proses yang relative kompleks. Berdasarkan sifatnya, metode FSM ini sangat cocok digunakan sebagai basis perancangan perangkat lunak pengendalian yang bersifat reaktif dan real time. Salah satu keutungan nyata penggunaan FSM adalah kemampuannya dalam mendekomposisi aplikasi yang relative besar dengan hanya menggunakan sejumlah kecil item state. Selain untuk bidang kontrol, Penggunaan metode ini pada kenyataannya juga umum digunakan sebagai basis untuk perancangan protokol-protokol komunikasi, perancangan perangkat lunak game, aplikasi WEB dan sebagainya. Dalam bahasa pemrograman prosedural seperti bahasa C, FSM ini umumnya direalisasikan dengan menggunakan statemen kontrol switch case atau/dan if..then. Dengan menggunakan statemen-statemen kontrol ini, aliran program secara praktis akan mudah dipahami dan dilacak jika terjadi kesalahan logika.
DIAGRAM KEADAAN
Diagram keadaan pada dasarnya merupakan salah satu bentuk representasi dari FSM. Diagram ini secara visual menggambarkan tingkah laku yang dimiliki oleh sistem kontrol yang kompleks kedalambentuk yang lebih sederhana dan relative mudah dipahami. Dalam diagram ini, state-state yang terdapat pada sebuah sistem digambarkan sebagai lingkaran yang diberi label unik, sedangkan transisi state yang diakibatkan oleh event tertentu direpresentasikan sebagai anak panah yang berasal dari state yang ditinggalkan menuju statecvyiaiing aktif. Setiap transisi yang terjadi umumnya

juga diikuti oleh aksi yang dilakukan oleh sistem yang dirancang. Secara praktis setiap diagram state yang dirancang akan selalu memiliki sebuah transisi awal (inisial) yang menuju salah satu state sejak sistem kontrol tersebut mulai dihidupkan. Gambar 1. berikut memperlihatkan contoh penggambarandiagram state:



Gambar 1. Contoh Diagram State Sederhana

Trafic Light



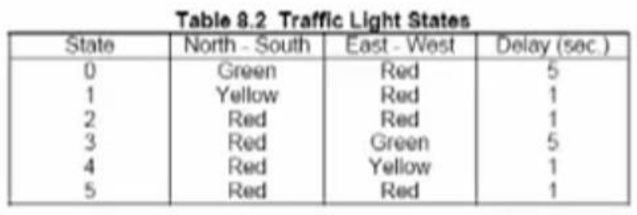
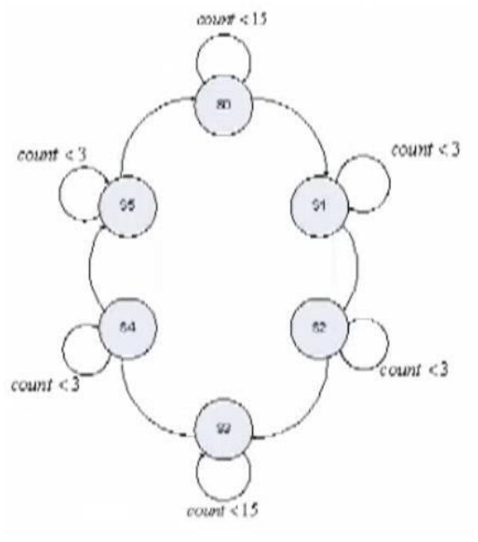


Table 8.2 Traffic Light States

State	North - South	East - West	Delay (sec.)
0	Green	Red	5
1	Yellow	Red	1
2	Red	Red	1
3	Red	Green	5
4	Red	Yellow	1
5	Red	Red	1

State Diagram for controlling Traffic Lights



3. Peralatan

1. PC yang sudah terinstal ISE 13.1
2. Xilinx Spartan 3
3. Downloader JTAG USB
4. Power Supply 5 volt

4. Langkah percobaan

1. Buatlah ***new project*** dengan nama **Lab9.**

2. Tambahkan program VHDL dibawah ini.

```
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;          cx
```

```vhdl
        signal count: STD_LOGIC_VECTOR(3 downto 0);
        constant SEC5: STD_LOGIC_VECTOR(3 downto 0):= "1111";
        constant SEC1: STD_LOGIC_VECTOR(3 downto 0):= "0011";
begin
        process (clr,clk)
        begin
                if clr='1' then
                                state <= s0;
                                count <= X"0";
                elsif clk'event and clk='1' then
                        case state is
                                when s0 =>   if count < SEC5 then
                                                        state <= s0;
                                                        count <= count + 1;
                                                else
                                                        state <= s1;
                                                        count <= x"0";
                                                end if;
                                when s1 => if count < SEC1 then
                                                        state <= s1;
                                                        count <= count + 1;
                                                else
                                                        state <= s2;
                                                        count <= x"0";
                                                end if;
                                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;
```

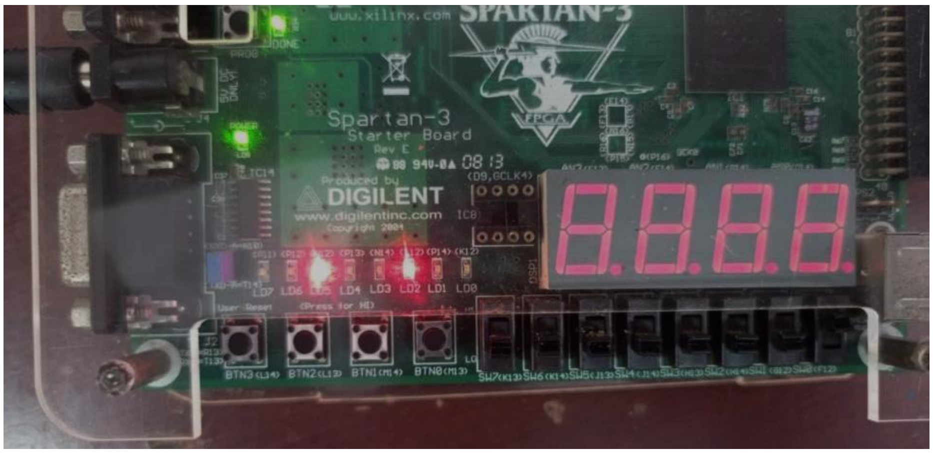
```
                              else
                                 state <= s4;
                                 count <= x"0";
                              end if;
                  when s4 => if count < SEC1 then
                                 state <= s4;
                                 count <= 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";
                              end 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";
      when others => lights <= "100001";
end case;
end process;
end Behavioral ;
```

3. Lakukan sintesis dan uplaod program ke board spartan-3, amati hasilnya dan dan demokanhasilnya ke dosen pengampu.

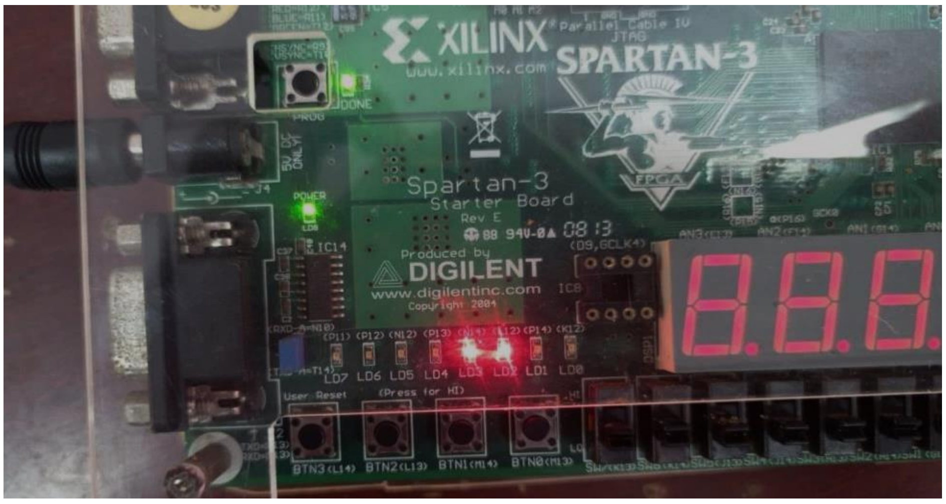
5. Hasil Percobaan

A. Langkah Percobaan

Foto Praktikum



Gambar 1 : Hasil pada FPGA dari Langkah Percobaan



Gambar 2 : Hasil pada FPGA dari Langkah Percobaan

B. Latihan

Traffic Light

Kode Program

Clock Divider

```
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;
                    clk190 : out  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";
              elsif mclk' event and mclk ='1' then
                     q <= q + 1;
              end if;

              clk48 <= q(20);
              clk190 <= q(17);

       end process;
end Behavioral;
```

Traffic Light

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

entity trafficlight is
port( clr : in STD_LOGIC;
             clk : in STD_LOGIC;
             lights : out STD_LOGIC_VECTOR(5 downto 0)
             );
end trafficlight;

architecture Behavioral of trafficlight is
type state_type is (s0,s1,s2,s3,s4,s5);
signal state: state_type;
signal count : STD_LOGIC_VECTOR(3 downto 0);
constant SEC5: STD_LOGIC_VECTOR(3 downto 0):= "1111";
constant SEC1: STD_LOGIC_VECTOR(3 downto 0):= "0011";
```

```
begin
        process (clr,clk)
        begin
                if clr='1' then
                        state <= s0;
                        count <= X"0";
                elsif clk' event and clk='1' then
                        case state is
                                when s0 =>      if count < SEC5 then
                                                                        state <= s0;
                                                                        count <= count + 1;
                                                                else
                                                                        state <= s1;
                                                                        count <= x"0";
                                                                end if;
                                when s1 =>      if count < SEC1 then
                                                                        state <= s1;
                                                                        count <= count + 1;
                                                                else
                                                                        state <= s2;
                                                                        count <= x"0";
                                                                end if;
                                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";
                                                                end 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";
                                                                end if;
                                when others => state <= s0;
                        end case;
                end if;
        end process;

        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";
```

```
                    when others => lights <= "100001";
              end case;
        end process;
end Behavioral;
```

Decoder 7 segment

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

entity decoder7seg is
port( x : in STD_LOGIC_VECTOR(15 downto 0);
             clk : in STD_LOGIC;
             clr : in STD_LOGIC;
             y : out STD_LOGIC_VECTOR(6 downto 0);
             an : out STD_LOGIC_VECTOR(3 downto 0);
             dp : out STD_LOGIC
             );

end decoder7seg;

architecture Behavioral of decoder7seg 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);
       aen <= "1111";
       dp <= '1';

       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;
       process(digit)
       begin
             case digit is
                    when "0001" => y <= "1111110";
                    when "0010" => y <= "1110111";
                    when "0100" => y <= "0111111";
                    when others => y <= "1111111";
             end case;
       end process;
```

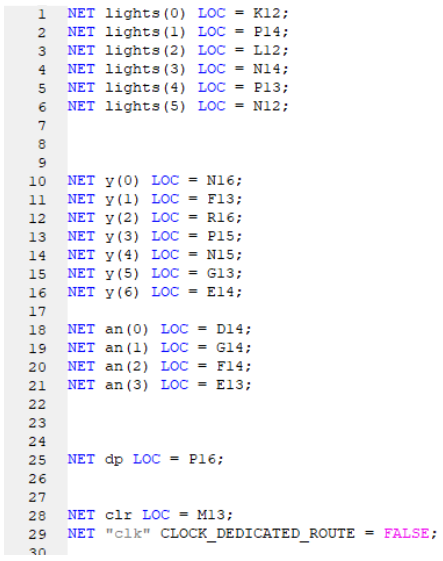
```
	process(clk,clr)

	begin

		if clr = '1' then

			clkdiv <= (others => '0');

		elsif clk' event and clk = '1' then

			clkdiv <= clkdiv+1;

		end if;

	end process;

end Behavioral;
```

Top Level

```
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;
             btn : in STD_LOGIC_VECTOR(3 downto 0);
             an : out STD_LOGIC_VECTOR(3 downto 0);
             y : out STD_LOGIC_VECTOR(6 downto 0);
             dp : out STD_LOGIC;
             --x: in STD_LOGIC_VECTOR(7 downto 0)
             lights : out STD_LOGIC_VECTOR(5 downto 0)
             );
end traffic;

architecture Behavioral of traffic is
signal clk48,clk190 : STD_LOGIC;
signal light : STD_LOGIC_VECTOR(5 downto 0);
signal x: STD_LOGIC_VECTOR(15 downto 0);

component clkdiv is
Port ( mclk : in  STD_LOGIC;
                    clr : in  STD_LOGIC;
                    clk48 : out  STD_LOGIC;
                    clk190 : out  STD_LOGIC);
end component;

component trafficlight is
port( clr : in STD_LOGIC;
             clk : in STD_LOGIC;
             lights : out STD_LOGIC_VECTOR(5 downto 0)
             );
end component;
```

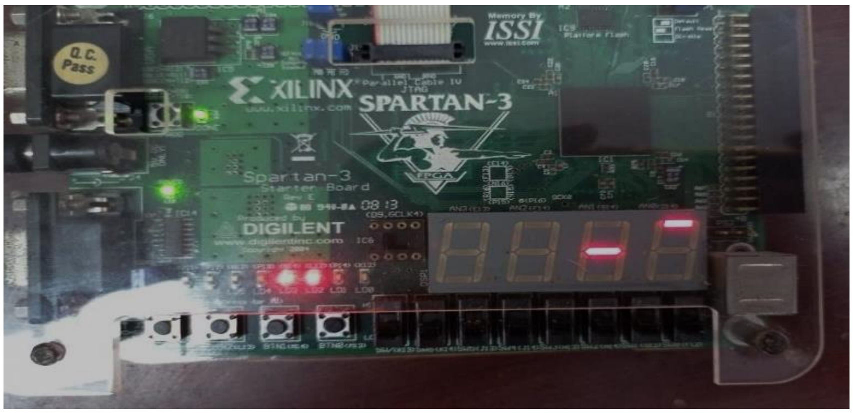
```
component decoder7seg is
port( x : in STD_LOGIC_VECTOR(15 downto 0);
clk : in STD_LOGIC;
clr : in STD_LOGIC;
y : out STD_LOGIC_VECTOR(6 downto 0);
an : out STD_LOGIC_VECTOR(3 downto 0);
dp : out STD_LOGIC
);
end component;

begin
       lights <= light;
       x <= "00000000"&'0'&light(5 downto 3) & '0'&light(2 downto 0);


       u1 : clkdiv port map(     mclk => clk,
       clr => clr,
       clk48 =>clk48,
       clk190 =>clk190);

       u2 : trafficlight port map(      clk => clk48,
       clr => clr,
       lights => light);

       u3 : decoder7seg port map(       x => x,
       clk => clk,
       clr => clr,
       y => y,
       an => an,
       dp => dp
       );
end Behavioral;
```

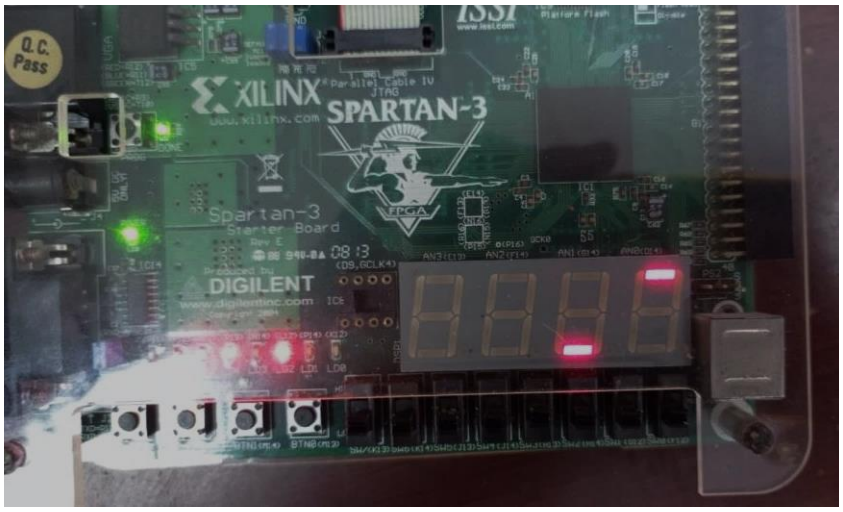
Konfigurasi Pin



```
NET lights(0) LOC = K12;
NET lights(1) LOC = P14;
NET lights(2) LOC = L12;
NET lights(3) LOC = N14;
NET lights(4) LOC = P13;
NET lights(5) LOC = N12;



NET y(0) LOC = N16;
NET y(1) LOC = F13;
NET y(2) LOC = R16;
NET y(3) LOC = P15;
NET y(4) LOC = N15;
NET y(5) LOC = G13;
NET y(6) LOC = E14;

NET an(0) LOC = D14;
NET an(1) LOC = G14;
NET an(2) LOC = F14;
NET an(3) LOC = E13;



NET dp LOC = P16;


NET clr LOC = M13;
NET "clk" CLOCK_DEDICATED_ROUTE = FALSE;
```

Foto Praktikum


Gambar 3 : Hasil pada FPGA dari Latihan


Gambar 4 : Hasil pada FPGA dari Latihan

Vending Machine

Kode Program

VM Modul

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity vm is
port(btn: in std_logic_vector (3 downto 0);
      a: out std_logic_vector (3 downto 0);
      b: out std_logic_vector (3 downto 0)
      );
end vm;

architecture Behavioral of vm is
type state_type is(btn50,btn500,btn100,btn200,
                              btn150,btn250,btn550,btn350,
                              btn650,btn750,btn600,btn700,
                              btn300,btn800,btn850,nul);
signal p: state_type;

begin

process(btn,p)
begin
     case btn is
          when "0001" => p <=btn50;
          when "0010" => p <=btn100;
          when "0100" => p <=btn200;
          when "1000" => p <=btn500;
          when "0011" => p <=btn150;
          when "0101" => p <=btn250;
          when "1001" => p <=btn550;
          when "0110" => p <=btn300;
          when "1010" => p <=btn600;
          when "1100" => p <=btn700;
          when "0111" => p <=btn350;
          when "1011" => p <=btn650;
          when "1101" => p <=btn750;
          when "1110" => p <=btn800;
          when "1111" => p <=btn850;
          when others => p <=nul;
     end case;
end process;
```

```
process(p)
begin
      case p is
            when btn50 => a<="0000"; b<="0101";
            when btn100 => a<="0001"; b<="0000";
            when btn200 => a<="0010"; b<="0000";
            when btn500 => a<="0101"; b<="0000";
            when btn150 => a<="0001"; b<="0101";
            when btn250 => a<="0010"; b<="0101";
            when btn350 => a<="0011"; b<="0101";
            when btn550 => a<="0101"; b<="0101";
            when btn650 => a<="0110"; b<="0101";
            when btn750 => a<="0111"; b<="0101";
            when btn850 => a<="1000"; b<="0101";
            when btn300 => a<="0011"; b<="0000";
            when btn600 => a<="0110"; b<="0000";
            when btn700 => a<="0111"; b<="0000";
            when btn800 => a<="1000"; b<="0000";
            when others => a<="0000"; b<="0000";
      end case;
end process;

end Behavioral;
```

7 segment decoder

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

entity x7seg 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 x7seg;

architecture percobaan7_C of x7seg 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);
             aen <= "1111" ;
             dp <= '1' ;
```

```
-- 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 "0000" => a_to_g <= "0000001" ; -- 0
                            when "0001" => a_to_g <= "1001111" ; -- 1
                            when "0010" => a_to_g <= "0010010" ; -- 2
                            when "0011" => a_to_g <= "0000110" ; -- 3
                            when "0100" => a_to_g <= "1001100" ; -- 4
                            when "0101" => a_to_g <= "0100100" ; -- 5
                            when "0110" => a_to_g <= "0100000" ; -- 6
                            when "0111" => a_to_g <= "0001111" ; -- 7
                            when "1000" => a_to_g <= "0000000" ; -- 8
                            when "1001" => a_to_g <= "0000100" ; -- 9
                            when others => a_to_g <= "1111111" ; -- null
       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 percobaan7_C;
```

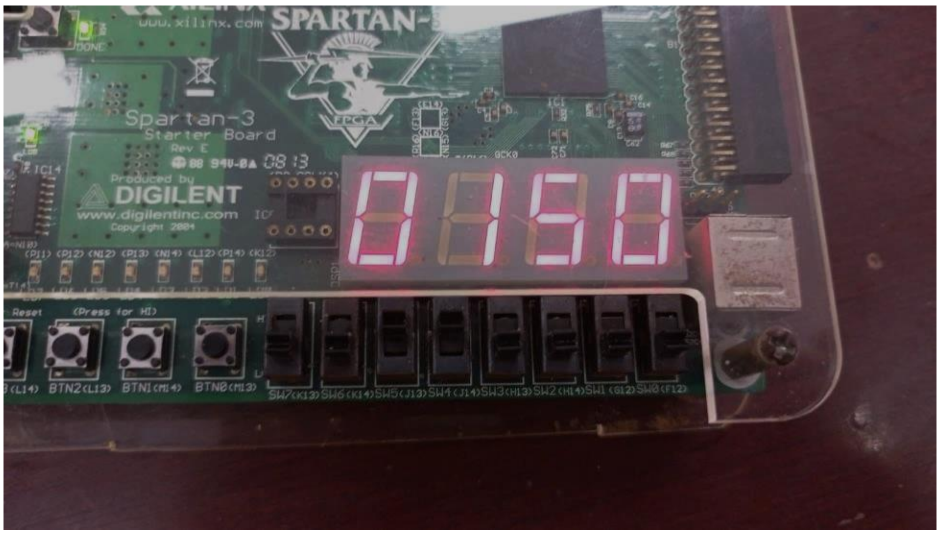
Top Level

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.NUMERIC_STD.ALL;

entity topvm is
port(input: in std_logic_vector (3 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 topvm;

architecture Behavioral of topvm is
component x7seg 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;
component vm is
port(btn: in std_logic_vector (3 downto 0);
          a: out std_logic_vector (3 downto 0);
          b: out std_logic_vector (3 downto 0)
          );
end component;

signal a: std_logic_vector (3 downto 0);
signal b: std_logic_vector (3 downto 0);
signal x: STD_LOGIC_VECTOR (15 downto 0);
begin
x<="0000" & a(3 downto 0) & b(3 downto 0) & "0000";
u1: vm port map(
                btn => input,
                a => a,
                b => b);
u2: x7seg port map(
                x =>  x,
                clk => clk,
                clr => clr,
                a_to_g => a_to_g,
                an => an,
                dp => dp);
end Behavioral;
```

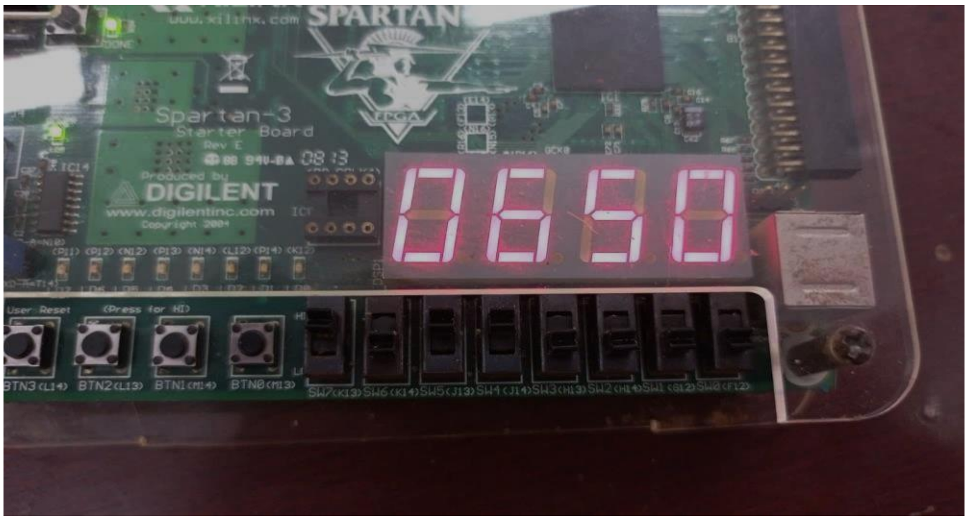
Konfigurasi Pin

```
NET "a_to_g[0]" LOC = N16;
NET "a_to_g[1]" LOC = F13;
NET "a_to_g[2]" LOC = R16;
NET "a_to_g[3]" LOC = P15;
NET "a_to_g[4]" LOC = N15;
NET "a_to_g[5]" LOC = G13;
NET "a_to_g[6]" LOC = E14;

NET "an[0]" LOC = D14;
NET "an[1]" LOC = G14;
NET "an[2]" LOC = F14;
NET "an[3]" LOC = E13;

NET "dp" LOC = P16;

NET "clr" LOC = M13;

NET "input[0]" LOC = J14;
NET "input[1]" LOC = J13;
NET "input[2]" LOC = K14;
NET "input[3]" LOC = K13;
```

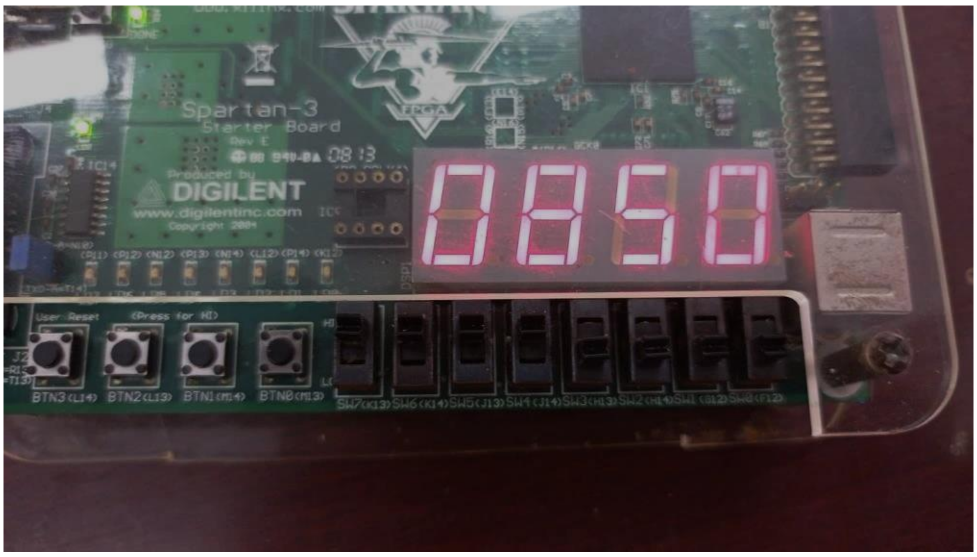
Foto Praktikum



Gambar 5 : output 7 seg display dengan input 0011

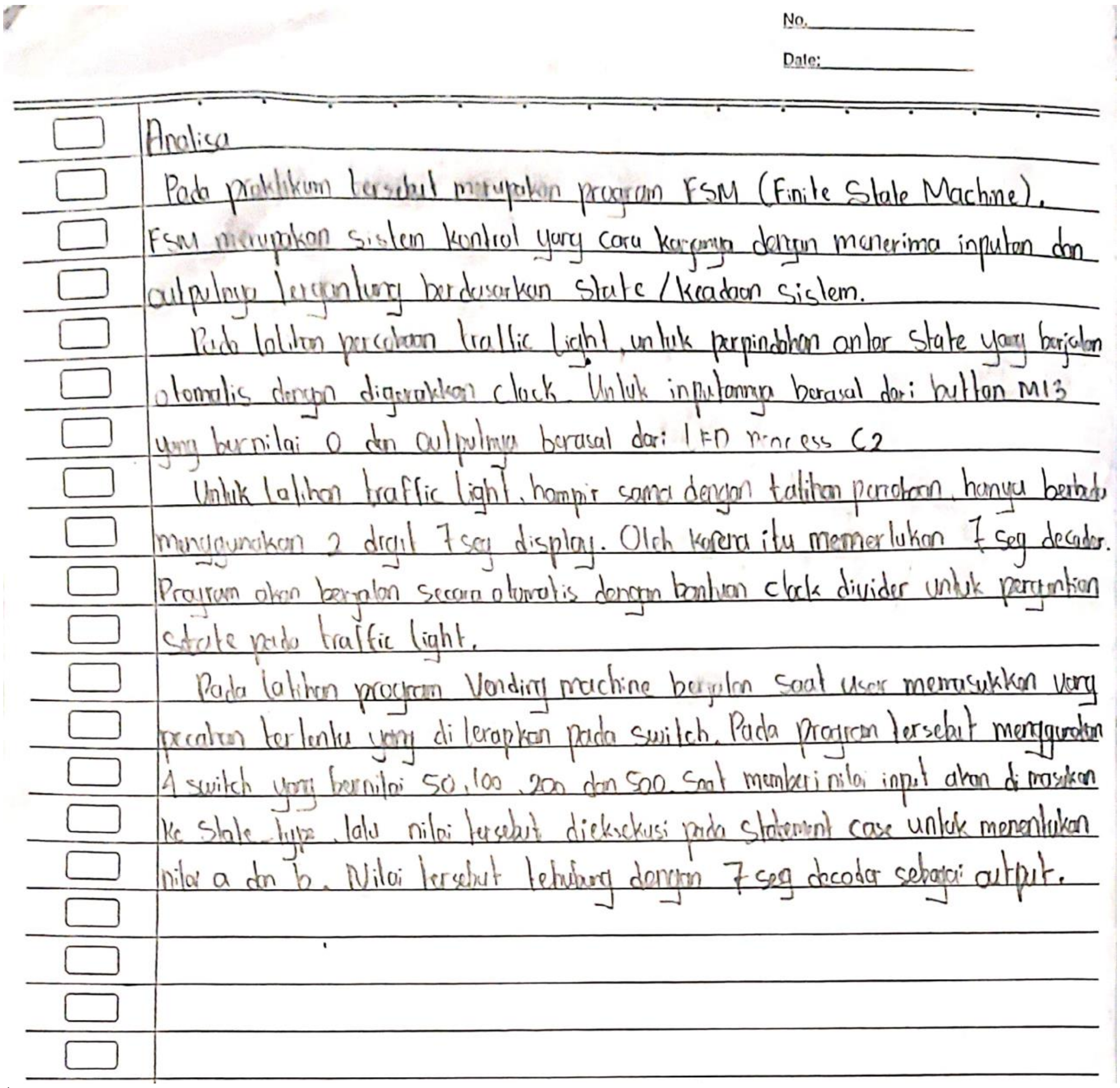


Gambar 6 : output 7 seg display dengan input 1011



Gambar 7 : output 7 seg display dengan input 1111

6. Analisa



No.

Date:

Analisa

Pada praktikum tersebut merupakan program FSM (Finite State Machine). FSM merupakan sistem kontrol yang cara kerjanya dengan menerima inputan dan outputnya tergantung berdasarkan State / Keadaan sistem.

Pada latihan percobaan traffic light, untuk perpindahan antar state yang berjalan otomatis dengan digerakkan clock. Untuk inputannya berasal dari button M13 yang bernilai 0 dan outputnya berasal dari LED process C2

Untuk latihan traffic light, hampir sama dengan latihan percobaan, hanya berbeda menggunakan 2 digit 7 seg display. Oleh karena itu memerlukan 7 seg decoder. Program akan berjalan secara otomatis dengan bantuan clock divider untuk pergantian state pada traffic light.

Pada latihan program Vending machine berjalan saat user memasukkan uang pecahan tertentu yang diterapkan pada switch. Pada program tersebut menggunakan 4 switch yang bernilai 50, 100, 200 dan 500. Saat memberi nilai input akan dimasukkan ke State type, lalu nilai tersebut dieksekusi pada statement case untuk menentukan nilai a dan b. Nilai tersebut terhubung dengan 7 seg decoder sebagai output.