1 D4 - TEKKOM B

DECODER 7 SEGMENT

PART 2

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, 13 Mei 2022

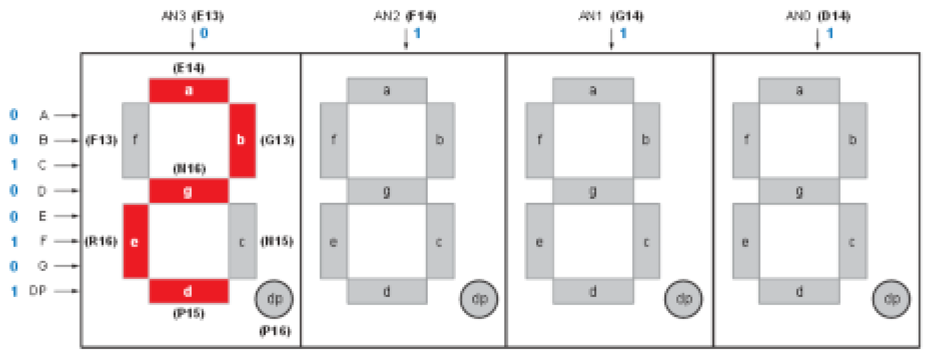
Percobaan 7 – Decoder 7 Segmen

1. Tujuan

- Mampu mengimplementasikan algoritma decoder 7 segmen.
- Dapat menampilkan nilai masukan keluaran pada 7 segmen dalam modul FPGA.

2. Teori

Pada percobaan ini akan mempelajari beberapa rangkaian kombinasional yaitu BCD to 7'segmen. Rangkaian BCD to 7'segmen adalah rangkaian kombinasional yang mengkodekan bilangan biner menjadi bilangan desimal, selanjutnya bilangan tersebut di konfigurasi menjadi tampilan seven segmen. Konfigurasi seven segmen pada board spartan 3 ditunjukan pada gambar 7.1.



Gambar 7. 1 Konfigurasi seven segmen pada board spartan3

Terdapat 4 buah sevensegmen yang disusun secara pararel dengan data 1-g dan dot di rangkai menjadi satu. Sedangkan kendali tampilan di atur melalui pin AN di setiap segmen.

3. Alat dan Bahan

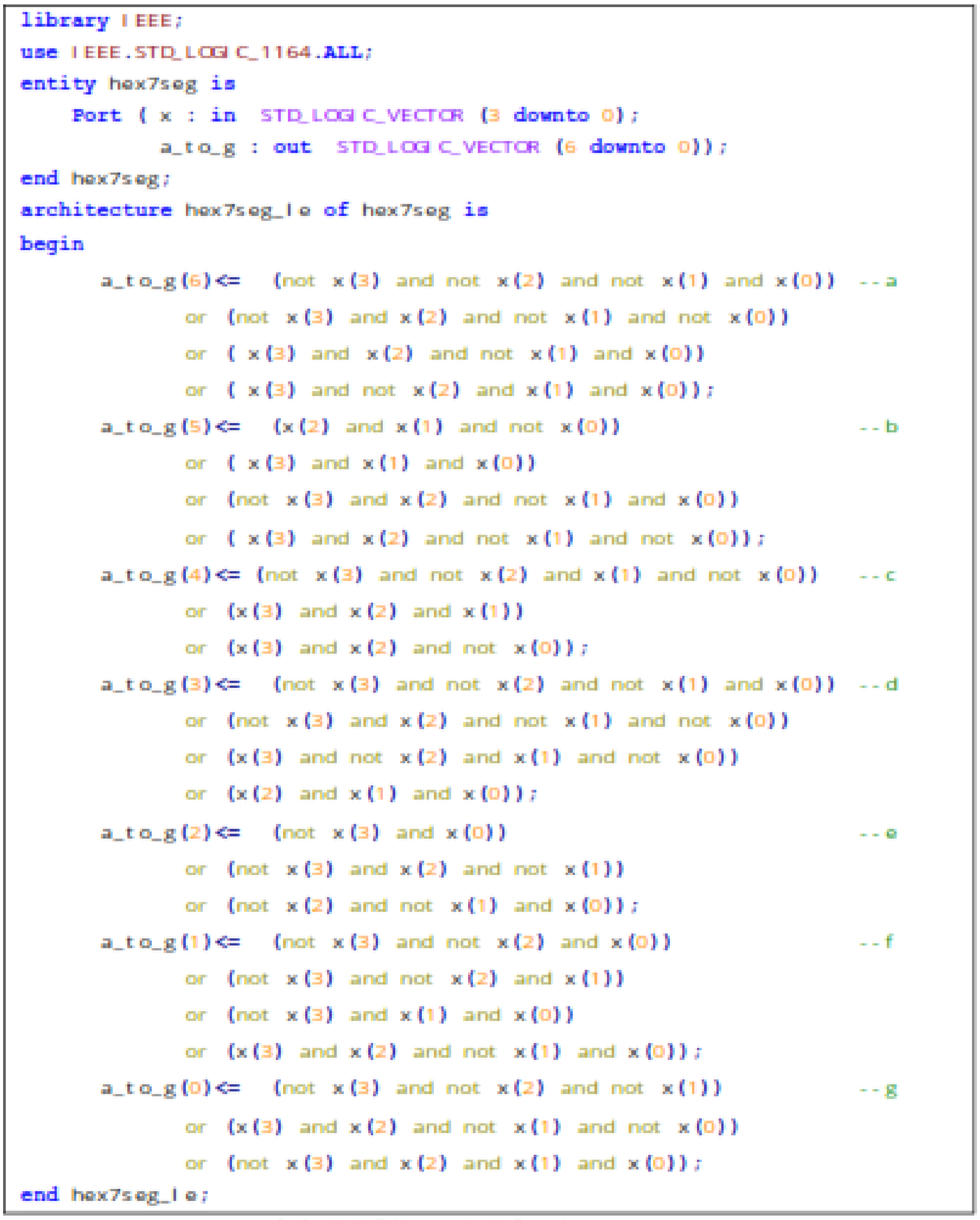
1. PC yang sudah terinstall ISE 13.1
2. Xilinx Sparatan 3
3. Downloader JTAG USB
4. Power Supply 5 volt

4. Langkah Percobaan

A. 7 Segmen Display.

Percobaan ini akan membuat program dengan masukan switch dan keluaran 7segmen.

1. Buatlah new project dengan nama Lab7A.
2. Tambahkan program dibawah ini.
   a. 7-segmen Decoder dengan Logic Equation



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity hex7seg is
    Port ( x : in  STD_LOGIC_VECTOR (3 downto 0);
           a_to_g : out  STD_LOGIC_VECTOR (6 downto 0));
end hex7seg;
architecture hex7seg_le of hex7seg is
begin
        a_to_g(6)<=  (not x(3) and not x(2) and not x(1) and x(0))  --a
               or (not x(3) and x(2) and not x(1) and not x(0))
               or ( x(3) and x(2) and not x(1) and x(0))
               or ( x(3) and not x(2) and x(1) and x(0));
        a_to_g(5)<=  (x(2) and x(1) and not x(0))                   --b
               or ( x(3) and x(1) and x(0))
               or (not x(3) and x(2) and not x(1) and x(0))
               or ( x(3) and x(2) and not x(1) and not x(0));
        a_to_g(4)<= (not x(3) and not x(2) and x(1) and not x(0))   --c
               or (x(3) and x(2) and x(1))
               or (x(3) and x(2) and not x(0));
        a_to_g(3)<=  (not x(3) and not x(2) and not x(1) and x(0))  --d
               or (not x(3) and x(2) and not x(1) and not x(0))
               or (x(3) and not x(2) and x(1) and not x(0))
               or (x(2) and x(1) and x(0));
        a_to_g(2)<=  (not x(3) and x(0))                            --e
               or (not x(3) and x(2) and not x(1))
               or (not x(2) and not x(1) and x(0));
        a_to_g(1)<=  (not x(3) and not x(2) and x(0))               --f
               or (not x(3) and not x(2) and x(1))
               or (not x(3) and x(1) and x(0))
               or (x(3) and x(2) and not x(1) and x(0));
        a_to_g(0)<=  (not x(3) and not x(2) and not x(1))           --g
               or (x(3) and x(2) and not x(1) and not x(0))
               or (not x(3) and x(2) and x(1) and x(0));
end hex7seg_le;
```

b. 7-Segmen Decoder Case Statement

```
begin
        process(x)
        begin
                case x is                           --  abcdefg
                        when x"0"  => a_to_g <= "0000001";--0
                        when x"1"  => a_to_g <= "1001111";--1
                        when x"2"  => a_to_g <= "0010010";--2
                        when x"3"  => a_to_g <= "0000110";--3
                        when x"4"  => a_to_g <= "1001100";--4
                        when x"5"  => a_to_g <= "0100100";--5
                        when x"6"  => a_to_g <= "0100000";--6
                        when x"7"  => a_to_g <= "0001111";--7
                        when x"8"  => a_to_g <= "0000000";--8
                        when x"9"  => a_to_g <= "0000100";--9
                        when x"A"  => a_to_g <= "0001000";--a
                        when x"B"  => a_to_g <= "1100000";--b
                        when x"C"  => a_to_g <= "0110001";--c
                        when x"D"  => a_to_g <= "1000010";--d
                        when x"E"  => a_to_g <= "0110000";--e
                        when others => a_to_g <= "0111000"; --f
                end case;
        end process;
end hex7segb;
```

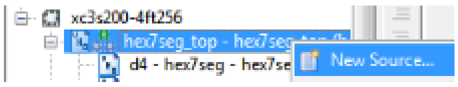
Script 7. 2. 7-Segmen Case Statement

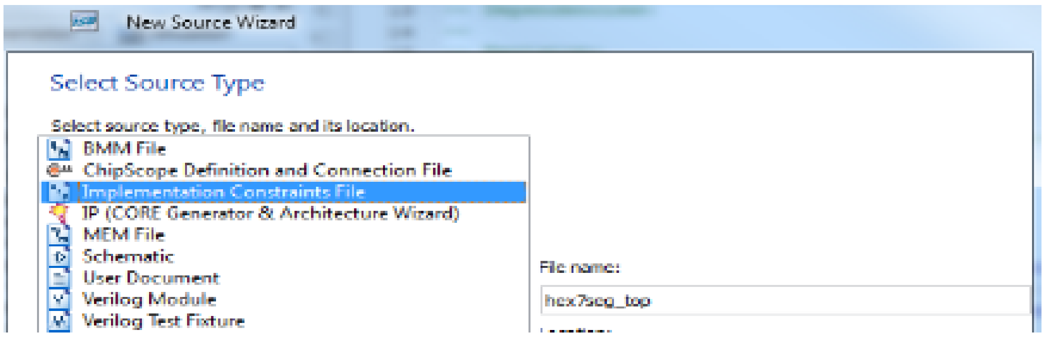
c. 7-Segmen Top Level

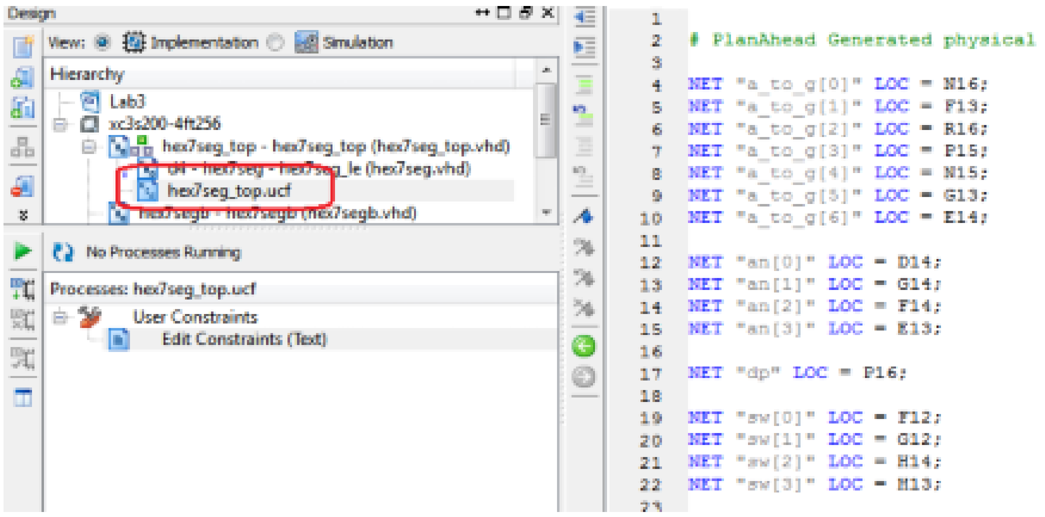
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity hex7seg_top is
    Port ( sw : in  STD_LOGIC_VECTOR (3 downto 0);
           a_to_g : out  STD_LOGIC_VECTOR (6 downto 0);
           an : out  STD_LOGIC_VECTOR (3 downto 0);
           dp : out  STD_LOGIC);
end hex7seg_top;

architecture hex7seg_top of hex7seg_top is
        component hex7segb is
                port(
                        x: in STD_LOGIC_VECTOR (3 downto 0);
                      a_to_g : out  STD_LOGIC_VECTOR (6 downto 0)
                );
        end component;
begin
        an <= "0000";           -- all digit on
        dp<= '1';               -- dp off
        d4: hex7seg port map (x=>sw, a_to_g=>a_to_g);

end hex7seg_top;
```

Script 7. 3. 7-Segmen Top Level

3. Atur agar program hex7seg_top menjadi Top modul.
4. Lakukan Synthezises-XST, pastikan tidak ada error.
5. Tambahkan konfigurasi pin dengan cara tambahkan new source| Implementation Contraints File. Beri nama hex7seg_top.




New Source Wizard
Select Source Type
Select source type, file name and its location.
BMM File
ChipScope Definition and Connection File
Implementation Constraints File
IP (CORE Generator & Architecture Wizard)
MEM File
Schematic
User Document
Verilog Module
Verilog Test Fixture
File name:
hex7seg_top


Design
View: Implementation Simulation
Hierarchy
Lab3
xc3s200-4ft256
hex7seg_top - hex7seg_top (hex7seg_top.vhd)
d4 - hex7seg - hex7seg_le (hex7seg.vhd)
hex7seg_top.ucf
hex7segb - hex7segb (hex7segb.vhd)
No Processes Running
Processes: hex7seg_top.ucf
User Constraints
Edit Constraints (Text)

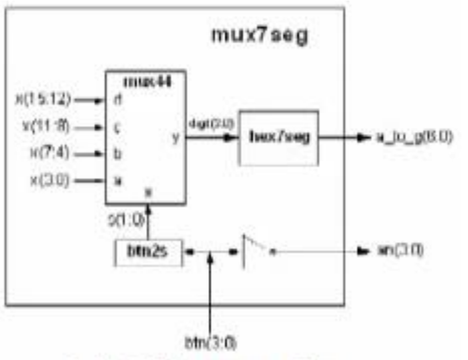
```

# PlanAhead Generated physical

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 "sw[0]" LOC = F12;
NET "sw[1]" LOC = G12;
NET "sw[2]" LOC = H14;
NET "sw[3]" LOC = H13;
```

6. Klik Pada jendela hex7seg_top.ucf dan isikan konfigurasi pin seperti gambar diatas.
7. Lakukan SynthezisesXST | Implement Design | Generate programming File sampai de ngan programming ke board Spartan-3 FPGA.
8. Atur switch sesuai dengan 1 digit angka terakhir nrp anda.
9. Foto hasil tampilan seven segmen.

B. Multiplexing 7-Segmen

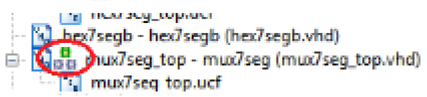
Percobaan ini untuk menampilkan data angka “1234” ke 7segmen dengan pengaturan tampilan dari enable 7segmen melaui push botton.

Blok program ditunjukkan pada gambar 7.2.



Gambar 7. 2 Blok program mux7seg

1. Buatlah new project dengan nama Lab7B.
2. Tambahkan Program baru dengan nama mux7seg_top.
3. Pastikan top modul pada mux7seg_top.vhd.


hex7seg_top.ucf
hex7segb - hex7segb (hex7segb.vhd)
mux7seg_top - mux7seg (mux7seg_top.vhd)
mux7seg top.ucf

4. Tulis program mux7seg_top.vhd berikut ini.

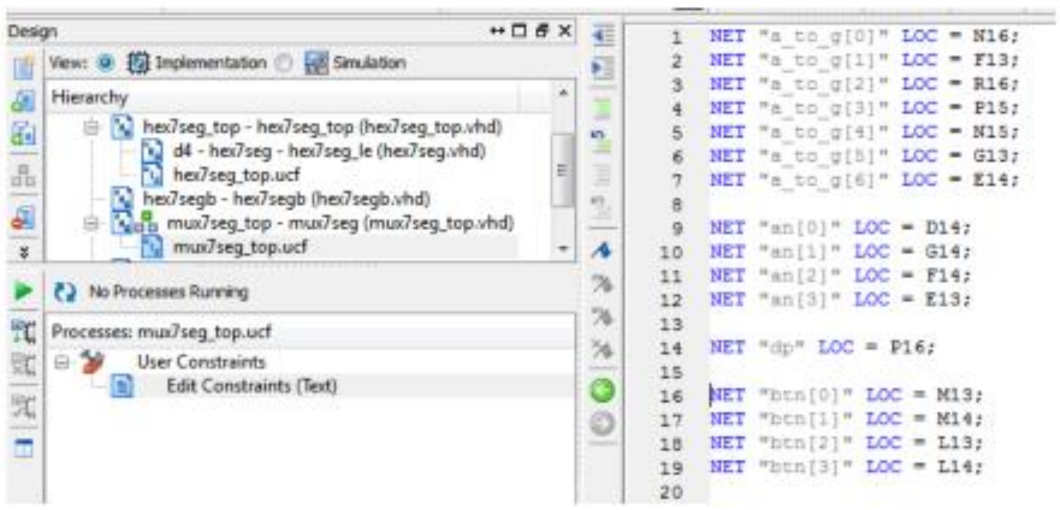
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux7seg_top is
    Port ( btn : in  STD_LOGIC_VECTOR (3 downto 0);
           a_to_g : out  STD_LOGIC_VECTOR (6 downto 0);
           an : out  STD_LOGIC_VECTOR (3 downto 0);
           dp : out  STD_LOGIC);
end mux7seg_top;

architecture mux7seg of mux7seg_top is
      signal x: STD_LOGIC_VECTOR (15 downto 0);
      signal s: STD_LOGIC_VECTOR (1 downto 0);
      signal digit: STD_LOGIC_VECTOR (3 downto 0);
```

```
begin
      x <= x"1234";
      an <= not btn;
      s(1) <= btn(2) or btn(3);
      s(0) <= btn(1) or btn(3);
      dp <='1';

-- quad 4 to 1 mux: mux44
      process(s)
      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               --  abcdefg
                when x"0" => a_to_g <= "0000001";   --0
                when x"1" => a_to_g <= "1001111";   --1
                when x"2" => a_to_g <= "0010010"; --2
                when x"3" => a_to_g <= "0000110";   --3
                when x"4" => a_to_g <= "1001100"; --4
                when x"5" => a_to_g <= "0100100";   --5
                when x"6" => a_to_g <= "0100000"; --6
                when x"7" => a_to_g <= "0001111";   --7
                when x"8" => a_to_g <= "0000000";   --8
                when x"9" => a_to_g <= "0000100";   --9
                when x"A" => a_to_g <= "0001000";   --a
                when x"B" => a_to_g <= "1100000"; --b
                when x"C" => a_to_g <= "0110001";   --c
                when x"D" => a_to_g <= "1000010"; --d
                when x"E" => a_to_g <= "0110000";   --e
                when others => a_to_g <= "0111000"; --f
            end case;
        end process;
end mux7seg;
```

Program 4. Multiplexing 7-Segmen

5. Tambahkan file .ucf untuk konfigurasi pin. Seperti dibawah ini.


Design
View: Implementation Simulation
Hierarchy
hex7seg_top - hex7seg_top (hex7seg_top.vhd)
d4 - hex7seg - hex7seg_le (hex7seg.vhd)
hex7seg_top.ucf
hex7segb - hex7segb (hex7segb.vhd)
mux7seg_top - mux7seg (mux7seg_top.vhd)
mux7seg_top.ucf
No Processes Running
Processes: mux7seg_top.ucf
User Constraints
Edit Constraints (Text)

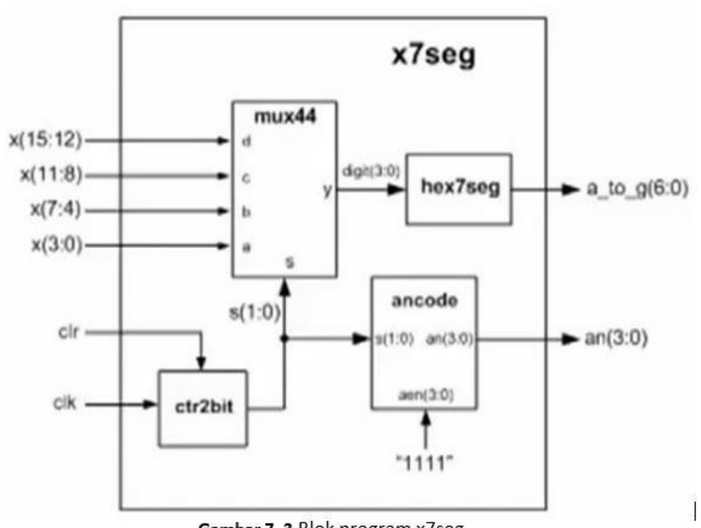
```
NET "a_to_g[0]" LOC = N16;
NET "a_to_g[1]" LOC = F13;
NET "a_to_g[2]" LOC = R16;
NET "a_to_g[3]" LOC = P15;
NET "a_to_g[4]" LOC = N15;
NET "a_to_g[5]" LOC = G13;
NET "a_to_g[6]" LOC = E14;

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 "btn[0]" LOC = M13;
NET "btn[1]" LOC = M14;
NET "btn[2]" LOC = L13;
NET "btn[3]" LOC = L14;
```

6. Jalankan programnya dan upload program ke dalam chip FPGA-nya
7. Ubah program agar tampilkan di seven segmen adalah 4 digit terakhir NRP anda.
8. Fotokan hasil tampilannya.

C. Multiplexing 7segmen Display dengan ClockDevider

Tampilan dari empat 7segmen akan diatur secara otomatis oleh ClockDevider.

Ada 2modelprogram yang akan ditampilkan yaitu ***x7seg*** dan ***x7segb***.



Gambar 7. 3. Blok program x7seg
lxxxix

x7seg.

1. Buatlah ***new project*** dengan nama **Lab7C1.**
2. Tambahkan program dibawah ini.
3. Seperti pada langkah sebelumnya. Tambahkan program vhdl berikut ini.

```
library IEEE;
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 x7seg 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 x"0" => a_to_g <= "0000001";   --0
            when x"1" => a_to_g <= "1001111";   --1
            when x"2" => a_to_g <= "0010010";   --2
```

```
                    when x"3"  => a_to_g <= "0000110";    --3
                    when x"4"  => a_to_g <= "1001100";    --4
                    when x"5"  => a_to_g <= "0100100";    --5
                    when x"6"  => a_to_g <= "0100000";    --6
   when x"7" => a_to_g <= "0001111"; --7
                    when x"8"  => a_to_g <= "0000000";    --8
                    when x"9"  => a_to_g <= "0000100";    --9
                    when x"A"  => a_to_g <= "0001000";    --a
                    when x"B"  => a_to_g <= "1100000";    --b
                    when x"C"  => a_to_g <= "0110001";    --c
                    when x"D"  => a_to_g <= "1000010";    --d
                    when x"E"  => a_to_g <= "0110000";    --e
                    when others => a_to_g <= "0111000";        --f
              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;
```

Program 5. Multiplexing 7-Segmen dengan ClockDevider A

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

entity x7seg_top is
    Port ( mclk : in  STD_LOGIC;         xci
           btn : 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_top;

architecture x7seg_top of x7seg_top 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;
      signal x: STD_LOGIC_VECTOR (15 downto 0);
begin
      x <= x"1234";
      x1 : x7seg port map
      (x=>x, clk=>mclk, clr=>btn, a_to_g=>a_to_g,an=>an, dp=>dp);
end x7seg_top;
```

Program 6. Multiplexing 7-Segmen Top Modul

4. Jalankan programnya dan upload program ke dalam chip FPGA-nya
5. Ubah program agar tampilkan di seven segmen adalah “Kelas(A/B)” – “0” - 2 digitterakhirNRP anda.
6. Fotokan hasil tampilannya.

x7segb.

1. Buatlah ***new project*** dengan nama **Lab7C2.**
2. Tambahkan program dibawah ini.
3. Seperti pada langkah sebelumnya. Tambahkan program vhdl berikut ini.

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

```vhdl
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';

-- set aen(3 downto 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";   --0
                  when x"1" => a_to_g <= "1001111";   --1
                  when x"2" => a_to_g <= "0010010"; --2
```

```
                    when x"3" => a_to_g <= "0000110";   --3
                    when x"4" => a_to_g <= "1001100";   --4
                    when x"5" => a_to_g <= "0100100";   --5
                    when x"6" => a_to_g <= "0100000";   --6
                    when x"7" => a_to_g <= "0001111";   --7
                    when x"8" => a_to_g <= "0000000";   --8
                    when x"9" => a_to_g <= "0000100";   --9
                    when x"A" => a_to_g <= "0001000";   --a
                    when x"B" => a_to_g <= "1100000";   --b
                    when x"C" => a_to_g <= "0110001";   --c
                    when x"D" => a_to_g <= "1000010";   --d
                    when x"E" => a_to_g <= "0110000";   --e
                    when others => a_to_g <= "0111000"; --f
              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;
```

Program 7. Multiplexing 7-Segmen dengan ClockDevider B

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

entity x7segb_top is
    Port ( clk : in  STD_LOGIC;
           btn : in  STD_LOGIC_VECTOR (3 downto 0);
           sw : in  STD_LOGIC_VECTOR (7 downto 0);
           a_to_g : out  STD_LOGIC_VECTOR (6 downto 0);
           an : out  STD_LOGIC_VECTOR (3 downto 0);
           dp : out  STD_LOGIC);
end x7segb_top;

architecture x7seg_top of x7segb_top is
      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;
      signal x : STD_LOGIC_VECTOR (15 downto 0);
begin
-- concatenate switch and 3 bottons
      x <= sw & btn(2 downto 0) & "01010";            -- digit 0 =A
      x2 : x7segb port map
                  (x=>x, clk=>clk, clr=>btn(3), a_to_g=>a_to_g,an=>an,
dp=>dp);

end x7seg_top;
```

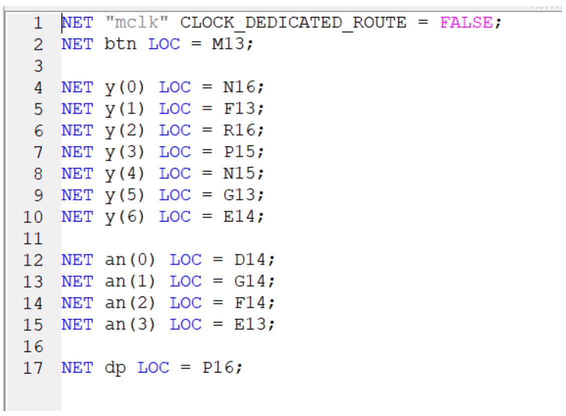
Program 8. Multiplexing 7-Segmen Top Modul B

4. Jalankan programnya dan upload program ke dalam chip FPGA-nya
5. Ubah program agar tampilkan di seven segmen adalah “CE” – 2 digit terakhir NRP anda.
6. Fotokan hasil tampilan seven segmen dan konfigurasi switch.

5. Hasil Percobaan

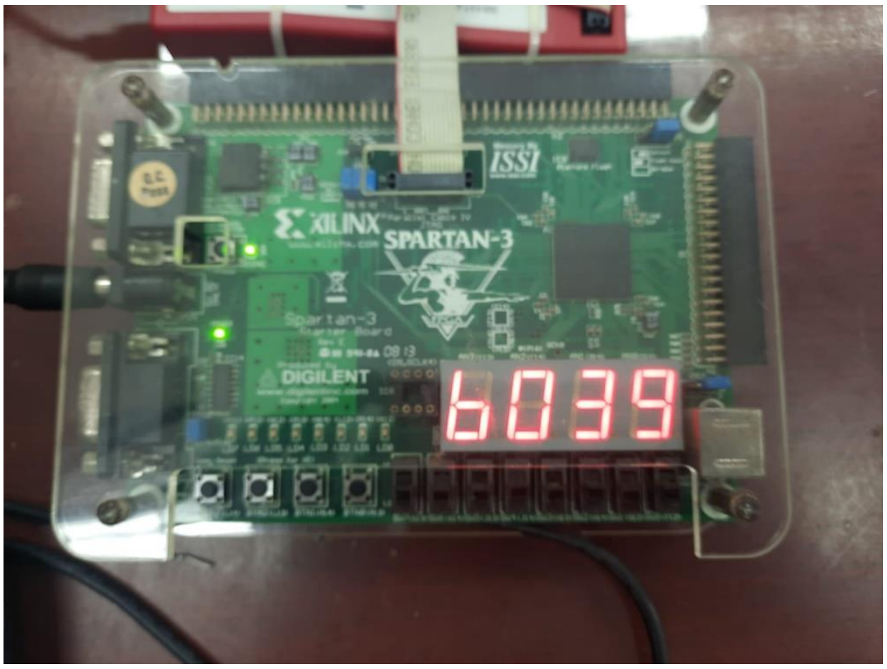
Percobaan C - A

Definisi Pin



```
NET "mclk" CLOCK_DEDICATED_ROUTE = FALSE;
NET btn LOC = M13;

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;
```

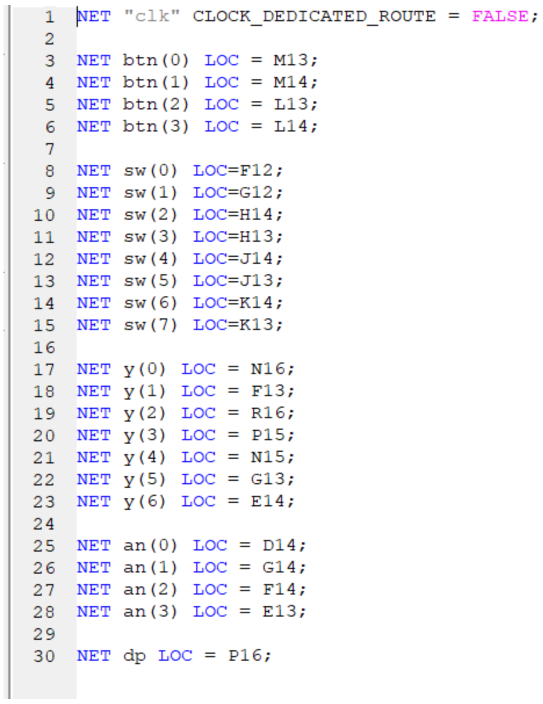
Tampilan Pada Board



Tampilan b039 pada 7 segment display

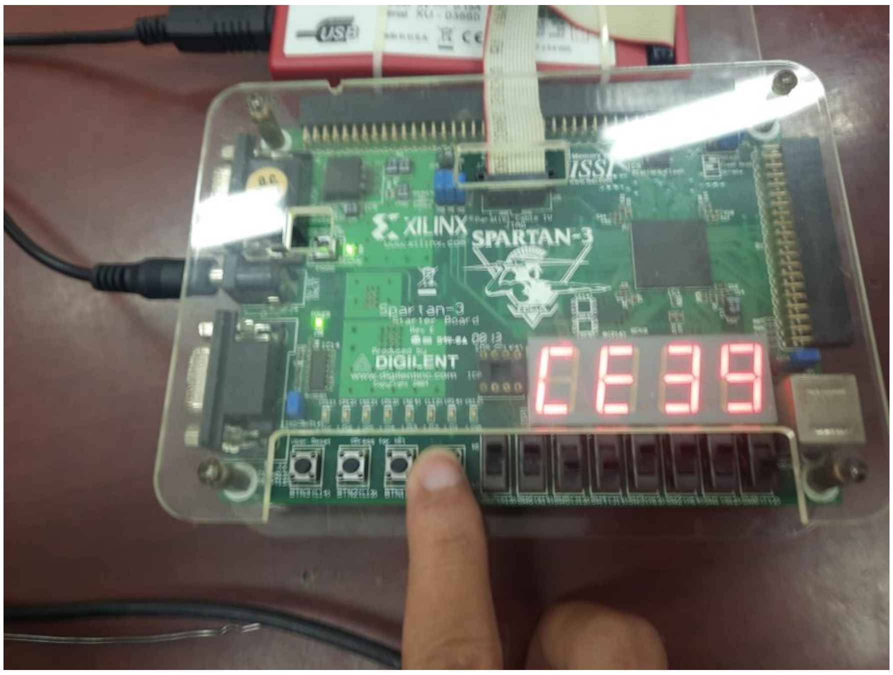
Percobaan C - B

Definisi Pin



```
NET "clk" CLOCK_DEDICATED_ROUTE = FALSE;

NET btn(0) LOC = M13;
NET btn(1) LOC = M14;
NET btn(2) LOC = L13;
NET btn(3) LOC = L14;

NET sw(0) LOC=F12;
NET sw(1) LOC=G12;
NET sw(2) LOC=H14;
NET sw(3) LOC=H13;
NET sw(4) LOC=J14;
NET sw(5) LOC=J13;
NET sw(6) LOC=K14;
NET sw(7) LOC=K13;

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;
```

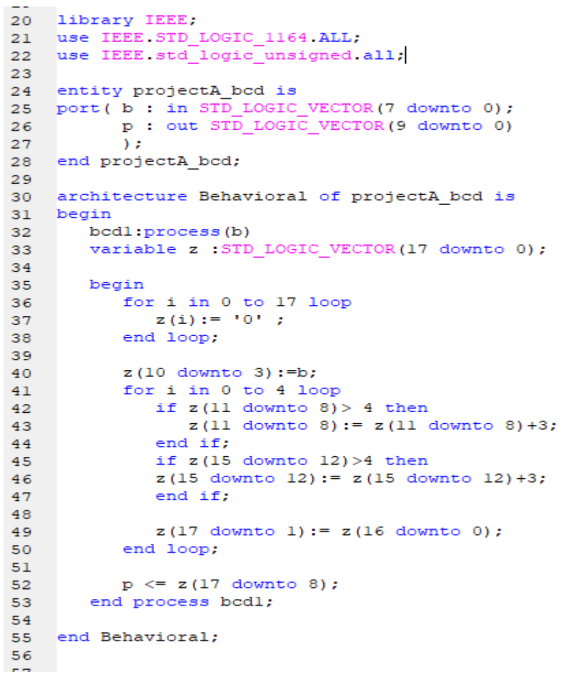
Tampilan pada Board



Tampilan CE39 pada 7 segment display

Mini Project

Source Code Binary to BCD



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

entity projectA_bcd is
port( b : in STD_LOGIC_VECTOR(7 downto 0);
      p : out STD_LOGIC_VECTOR(9 downto 0)
      );
end projectA_bcd;

architecture Behavioral of projectA_bcd is
begin
   bcd1:process(b)
   variable z :STD_LOGIC_VECTOR(17 downto 0);

   begin
      for i in 0 to 17 loop
         z(i):= '0' ;
      end loop;

      z(10 downto 3):=b;
      for i in 0 to 4 loop
         if z(11 downto 8)> 4 then
            z(11 downto 8):= z(11 downto 8)+3;
         end if;
         if z(15 downto 12)>4 then
         z(15 downto 12):= z(15 downto 12)+3;
         end if;

         z(17 downto 1):= z(16 downto 0);
      end loop;

      p <= z(17 downto 8);
   end process bcd1;

end Behavioral;
```

Source Code 7 Segment Decoder

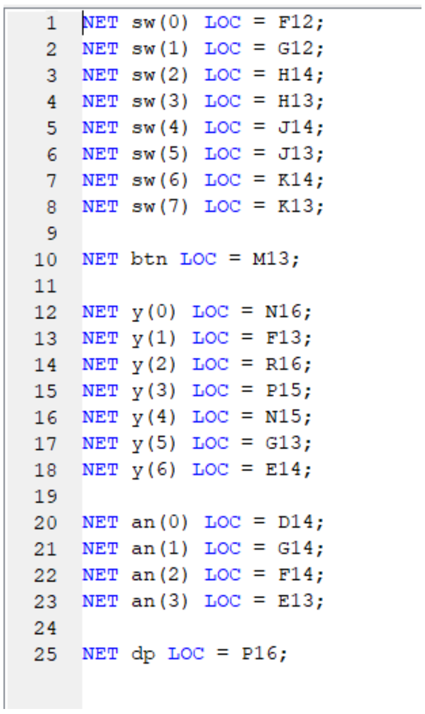
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity projectA is
port(  x : in STD_LOGIC_VECTOR(15 downto 0);
              clk : in STD_LOGIC;
              clr : in STD_LOGIC;
              y : out STD_LOGIC_VECTOR(6 downto 0);
              an : out STD_LOGIC_VECTOR(3 downto 0);
              dp : out STD_LOGIC);
end projectA;
architecture Behavioral of projectA 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 x"0" => y <= "0000001";
                        when x"1" => y <= "1001111";
                        when x"2" => y <= "0010010";
                        when x"3" => y <= "0000110";
                        when x"4" => y <= "1001100";
                        when x"5" => y <= "0100100";
                        when x"6" => y <= "0100000";
                        when x"7" => y <= "0001111";
                        when x"8" => y <= "0000000";
                        when others => y <= "0000100";
                end case;
        end process;
        process(s,aen)
        begin
                an <= "1111";
                if aen(conv_integer(s)) = '1' then
                        an(conv_integer(s)) <= '0';
                end if;
        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 Modul

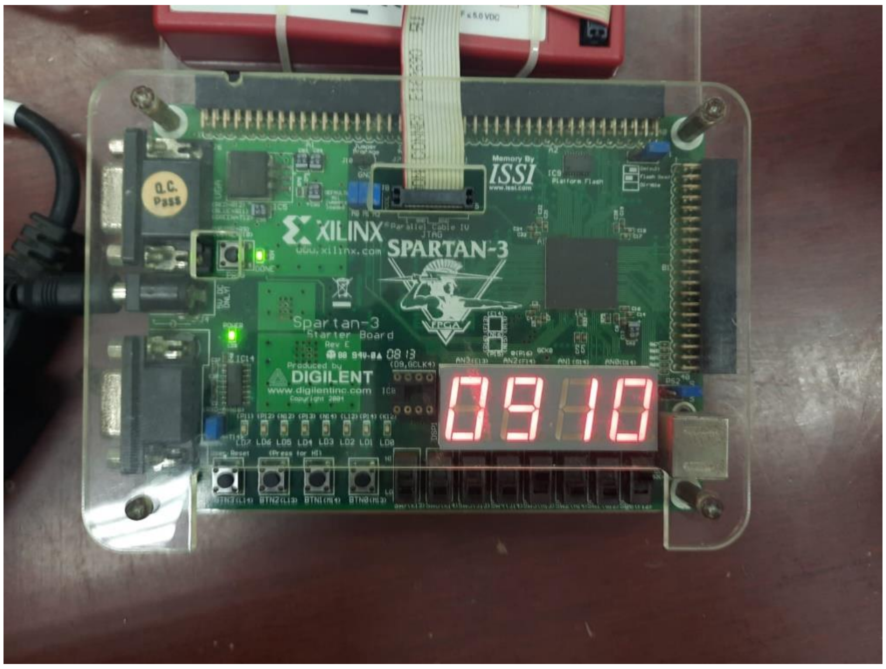
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;

entity projectA_top is
port(  clk : in STD_LOGIC;
             btn : in STD_LOGIC;
             sw : in STD_LOGIC_VECTOR(7 downto 0);
             y : out STD_LOGIC_VECTOR(6 downto 0);
             an : out STD_LOGIC_VECTOR(3 downto 0);
             dp : out STD_LOGIC
             );
end projectA_top;

architecture Behavioral of projectA_top is
        component projectA 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;
        component projectA_bcd is

                port(   b : in STD_LOGIC_VECTOR(7 downto 0);

                                p : out STD_LOGIC_VECTOR(9 downto 0)

                                );

        end component;


        signal x: STD_LOGIC_VECTOR(15 downto 0);
        signal adder : STD_LOGIC_VECTOR(7 downto 0);
        signal bcd : STD_LOGIC_VECTOR(9 downto 0);

begin

        adder <= bcd(7 downto 0)+1;
        x <= bcd(7 downto 0) & adder;

        x1 :projectA port map( x => x,

                                        clk => clk,
                                        clr => btn,
```

Definisi Pin

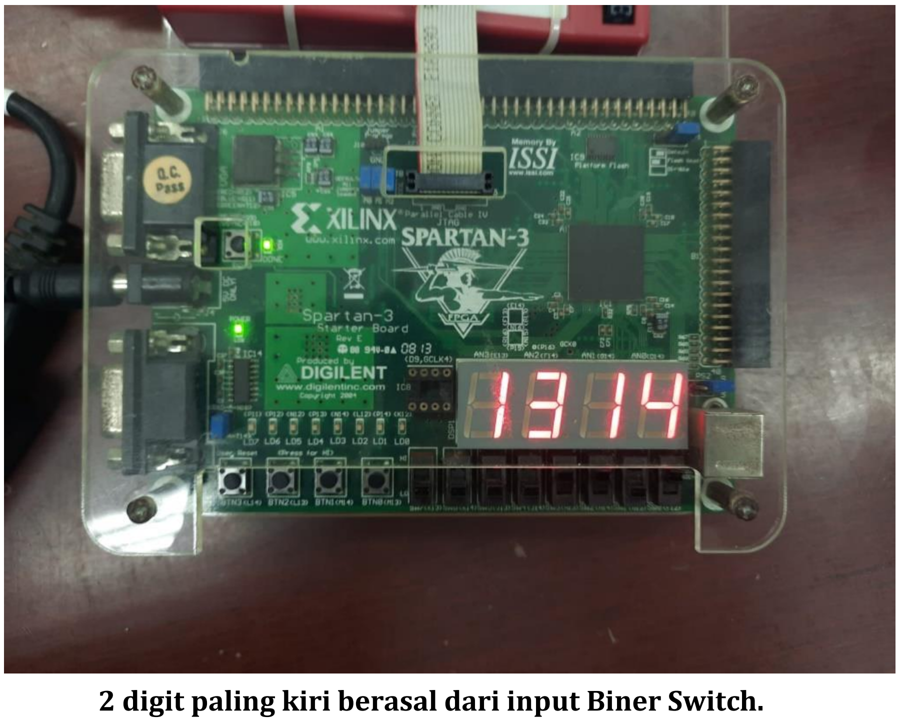


```
NET sw(0) LOC = F12;
NET sw(1) LOC = G12;
NET sw(2) LOC = H14;
NET sw(3) LOC = H13;
NET sw(4) LOC = J14;
NET sw(5) LOC = J13;
NET sw(6) LOC = K14;
NET sw(7) LOC = K13;

NET btn LOC = M13;

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;
```

Tampilan pada Board

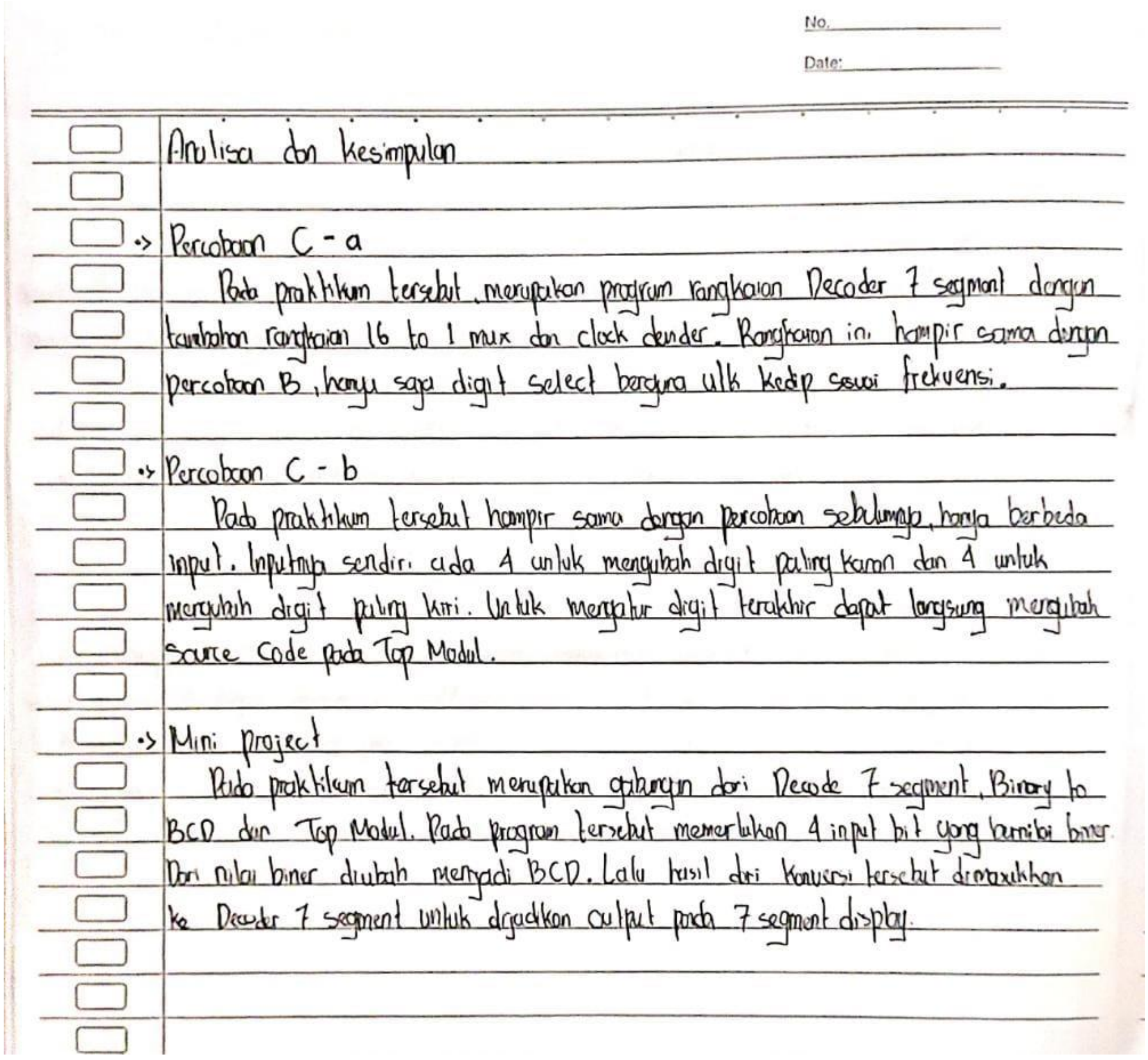


2 digit paling kiri berasal dari input Biner Switch. 2 digit dari kanan merupakan nilai dari switch dengan +1



2 digit paling kiri berasal dari input Biner Switch. 2 digit dari kanan merupakan nilai dari switch dengan +1

6. Analisa dan Kesimpulan



No.

Date:

Analisa dan kesimpulan

→ Percobaan C - a

Pada praktikum tersebut, merupakan program rangkaian Decoder 7 segment dengan tambahan rangkaian 16 to 1 mux dan clock divider. Rangkaian ini hampir sama dengan percobaan B, hanya saja digit select berguna utk kedip sesuai frekuensi.

→ Percobaan C - b

Pada praktikum tersebut hampir sama dengan percobaan sebelumnya, hanya berbeda input. Inputnya sendiri ada 4 untuk mengubah digit paling kanan dan 4 untuk mengubah digit paling kiri. Untuk mengatur digit terakhir dapat langsung mengubah source code pada Top Modul.

→ Mini project

Pada praktikum tersebut merupakan gabungan dari Decode 7 segment, Binary to BCD dan Top Modul. Pada program tersebut memerlukan 4 input bit yang bernilai biner. Dari nilai biner diubah menjadi BCD. Lalu hasil dari konversi tersebut dimasukkan ke Decoder 7 segment untuk dijadikan output pada 7 segment display.