1 D4 - TEKKOM B

COUNTER

Nama	: Septian Bagus Jumantoro
Kelas	: 1 – D4 Teknik Komputer B
NRP	: 3221600039
Dosen	: Reni Soelistijorini B.Eng., MT.
Mata Kuliah	: Praktikum Rangkaian Logika
Hari,Tgl. Praktikum	: Jumat, 27 Mei 2022

1. Tujuan

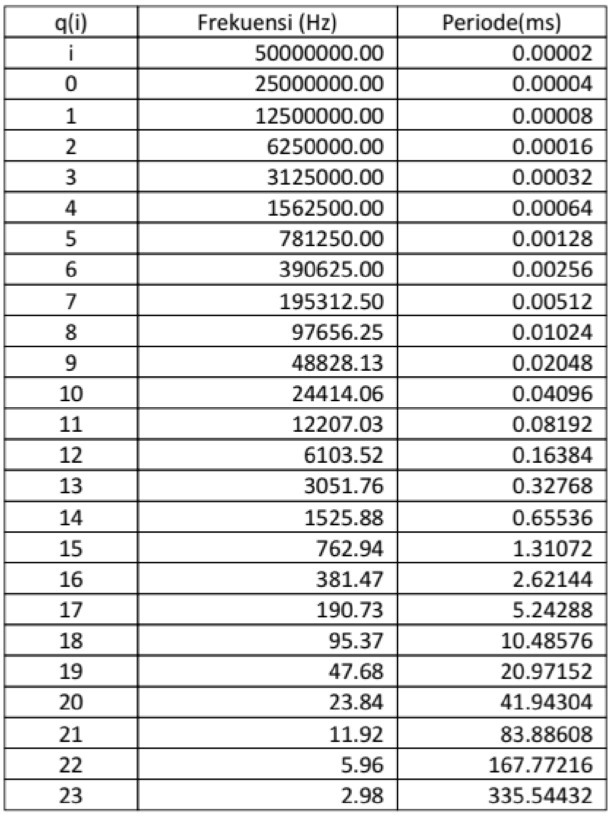
Dapat menjelaskan rangkaian pembagi clock. Dapat membuat rangkaian Counter modulus 10k.

2. Teori

Clock Divider

Frekuensi dan periode dari Xilinx Spartan 3 adalah 24-bit counter dengan clock sebesar 50MHz. Counter dapat digunakan untuk membagi frekuensi *f* dari clock, dimana frekuensi output *q(i)* adalah$f_i=f/2^{i+1}$. Tabel 2.1 menyatakan pembagian frekuensi.

Tabel 8.1 Clock divide frequencies



q(i)	Frekuensi (Hz)	Periode(ms)
i	50000000.00	0.00002
0	25000000.00	0.00004
1	12500000.00	0.00008
2	6250000.00	0.00016
3	3125000.00	0.00032
4	1562500.00	0.00064
5	781250.00	0.00128
6	390625.00	0.00256
7	195312.50	0.00512
8	97656.25	0.01024
9	48828.13	0.02048
10	24414.06	0.04096
11	12207.03	0.08192
12	6103.52	0.16384
13	3051.76	0.32768
14	1525.88	0.65536
15	762.94	1.31072
16	381.47	2.62144
17	190.73	5.24288
18	95.37	10.48576
19	47.68	20.97152
20	23.84	41.94304
21	11.92	83.88608
22	5.96	167.77216
23	2.98	335.54432

Counter

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

entity Counter is
    generic(N : integer :=8);
      Port ( clk : in  STD_LOGIC;
              clr : in  STD_LOGIC;
              q : out  STD_LOGIC_VECTOR (N-1 downto 0));
end Counter;

architecture Behavioral of Counter is
      signal count : std_logic_vector (N-1 downto 0);
begin
      process (clk, clr)
      begin
            if clr =' 1' then
                  count <= (others => '0');
            elsif clk'event and clk=' 1' then
                  count <= count + 1;
            end if;
            q <= count;
      end process;
end Behavioral;
```

```
--Example 4
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;
            clk25 : out STD_LOGIC;
            clk190 : out STD_LOGIC;
            clk3 : out STD_LOGIC
      );
end clkdiv;
architecture clkdiv of clkdiv is
signal q : STD_LOGIC_VECTOR(23 downto 0);
begin
      --clock divider
      process(mclk, clr)
      begin
```

```
            if clr =' 1' then
                  q <= X"000000";
            elsif mclk'event and mclk = '1' then
                  q <= q + 1;
            end if;
      end process;

      clk25 <= q(0);          --25Mhz
      clk190 <= q(17);  --190Hz
      clk3 <= q(23);          --3Hz

end clkdiv;
```

Skrip 8.1 Clock Divider

Pada skrip 8.1 adalah contoh 24-bit counter yang mempunyai 3 output, yaitu 25Mhz clock (clk25),clock 190Hz (clk190), dan clock 3Hz (clk3). Untuk memodifikasi component *clkdiv* untuk bisa menghasilkan output frekuensi yang berbeda dengan menggunakan tabel 8.1.

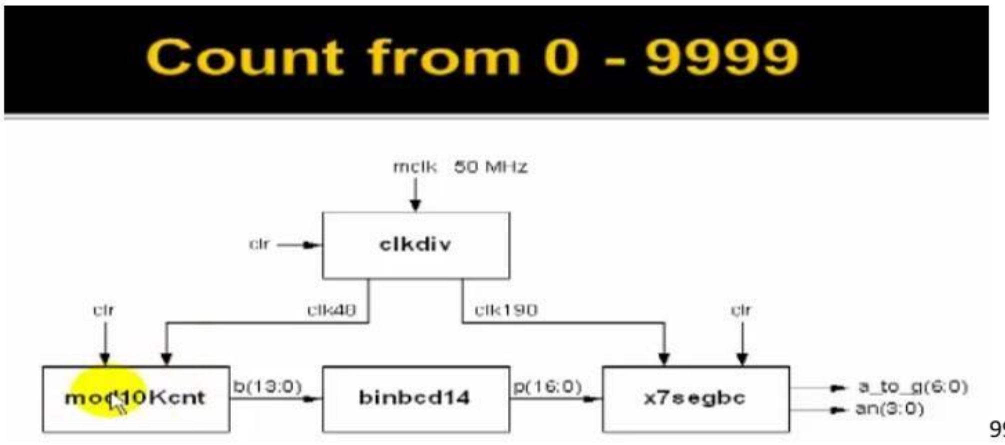
3. Peralatan

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

4. Langkah Percobaan

Pada percobaan ini akan mendisain counter 0-9999. Masukan rangkaian ini adalah mclk 50Mhz danbutton Reset. Keluaran di seven segmen(a_to_g) dan enable(an(3:0)).

1. Buatlah ***new project*** dengan nama **Lab8.**
2. Tambahkan program VHDL dibawah ini.



Clock Devider

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

entity Clockdiv is
      port (      Clk25Mhz: in STD_LOGIC;
                  Clk: out STD_LOGIC);
      end Clockdiv;

architecture Behavior of Clockdiv is
      constant max: integer := 25000000;
      constant half: integer := max/2;
      signal count: integer range 0 to max;
begin
      process
      begin
            wait until Clk25Mhz'EVENT and Clk25Mhz = '1';
            if count < max then
                  count <= count + 1;
            else
                  count <= 0;
            end if;

            if count < half then
                  Clk <= '0';
            else
                  Clk <= '1';
            end if;
      end process;
end Behavior;
```

Clock devider 48Hz dan 190Hz

```vhdl
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
      -- clock devider                       c
```

```vhdl
      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(19);
            clk190 <= q(17);
      end process;
end Behavioral ;
```

Counter modulus 10K.

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

entity mod10kcnt is
    Port ( clr : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           q : out  STD_LOGIC_VECTOR (13 downto 0));
end mod10kcnt;

architecture Behavioral of mod10kcnt is
	signal count :STD_LOGIC_VECTOR (13 downto 0);
begin
	-- modulo 10k
	process(clk,clr)
	begin
		if clr = '1' then
				count <= (others => '0');
		elsif clk'event and clk ='1' then
			if	conv_integer(count) = 9999 then
				count <= (others => '0');
			else
				count <= count + 1;
			end if;
		end if;
	end process;
	q <= count;
end Behavioral;
```

Biner 14 bit to BCD

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

entity binBCD14 is
    Port ( b : in  STD_LOGIC_VECTOR (13 downto 0);
           p : out  STD_LOGIC_VECTOR (16 downto 0));
```

```
end binBCD14;

architecture Behavioral of binBCD14 is

begin
	process(b)
		variable z : STD_LOGIC_VECTOR (32 downto 0);
	begin
		for i in 0 to 32 loop
			z(i) := '0';
		end loop;

		z(16 downto 3) := b;
		for i in 0 to 10 loop
			if z(17 downto 14) > 4 then
				z(17 downto 14) := z(17 downto 14) + 3;
			end if;
			if z(21 downto 18) > 4 then
				z(21 downto 18) := z(21 downto 18) + 3;
			end if;
			if z(25 downto 22) > 4 then
				z(25 downto 22) := z(25 downto 22) + 3;
			end if;
			if z(29 downto 26) > 4 then
				z(29 downto 26) := z(29 downto 26) + 3;
			end if;
			z(32 downto 1) := z(31 downto 0);
		end loop;
		p <= z(30 downto 14);
	end process;
end Behavioral;
```

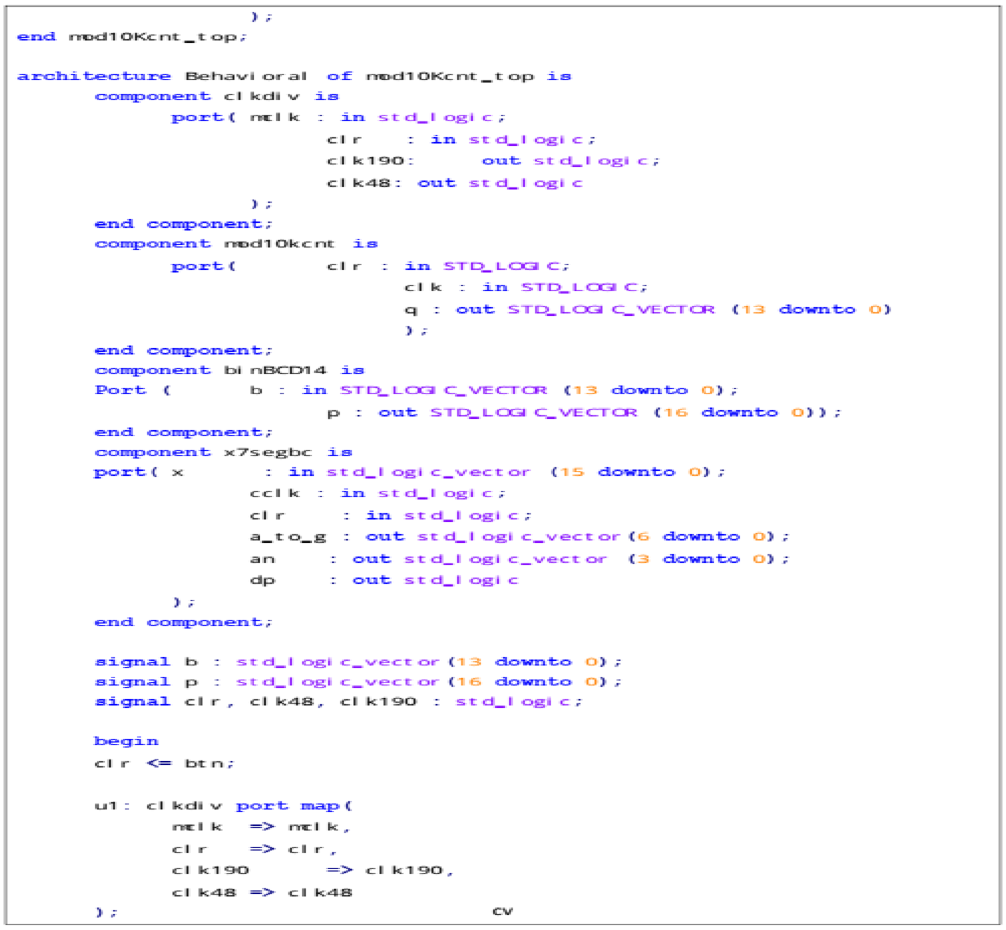
x7segbc

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

entity x7segbc is
      port( x       : in std_logic_vector (15 downto 0);
                  cclk : 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 x7segbc;                          cii
```

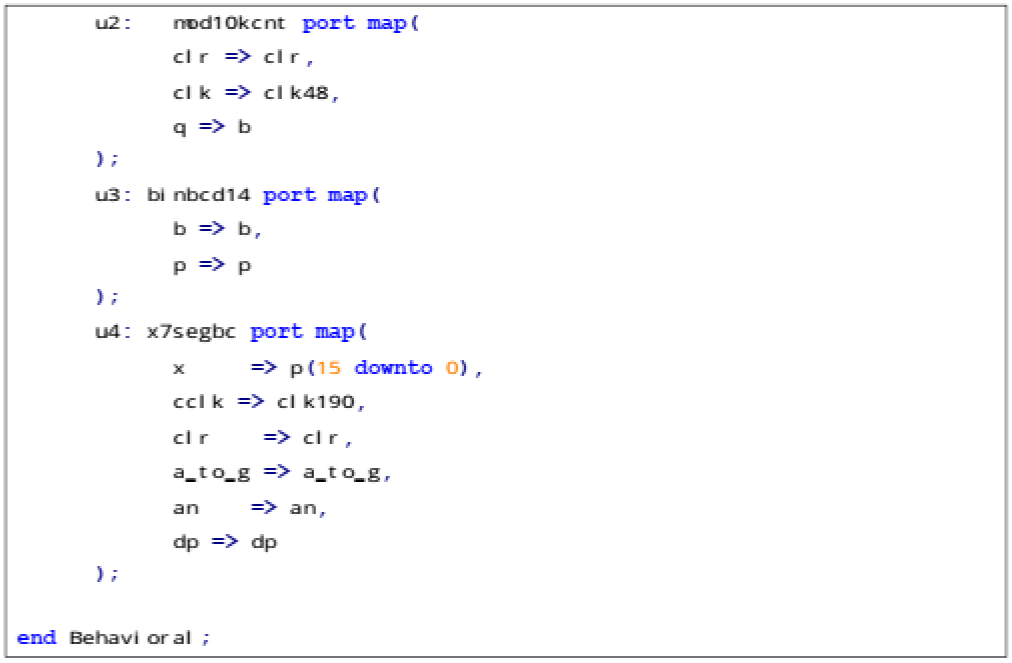
```
architecture Behavioral of x7segbc is
      signal s :  std_logic_vector(1 downto 0);
      signal digit : std_logic_vector(3 downto 0);
      signal aen : std_logic_vector(3 downto 0);
begin
      dp <= '1';
      -- set aen(3 downto 0) for leading blanks
      aen(3) <= x(15) or x(14) or x(13) 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';

-- Quad 4-to-1 MUX:      mux44
      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;

-- 2 bit counter
      process(cclk, clr)
      begin
            if clr='1' then
                  s <= "00";
            elsif cclk'event and cclk='1' then
                  s <= s + 1;
            end if;
      end process;
end Behavioral;
```

Mod10Kcnt_top

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity mod10Kcnt_top is
      port( mclk  : in std_logic;
                  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
```



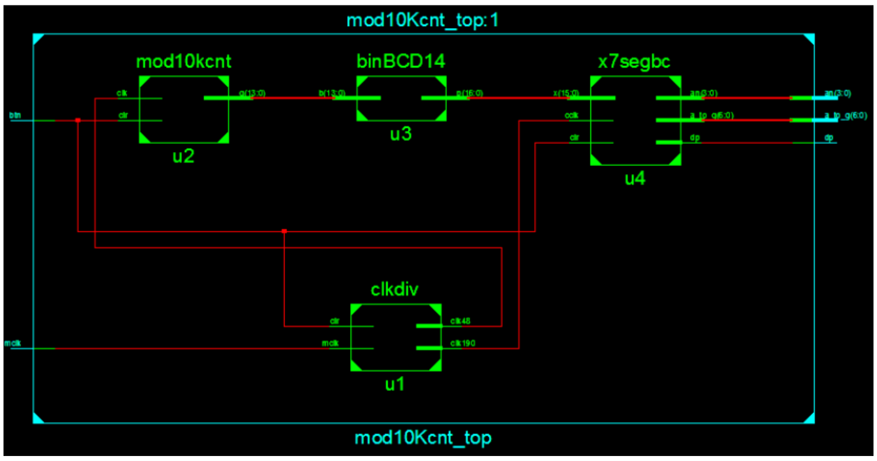
```
                              );
end mod10Kcnt_top;

architecture Behavioral of mod10Kcnt_top is
      component clkdiv is
            port( mclk : in std_logic;
                        clr    : in std_logic;
                        clk190:     out std_logic;
                        clk48: out std_logic
                  );
      end component;
      component mod10kcnt is
            port(       clr  : in STD_LOGIC;
                              clk  : in STD_LOGIC;
                              q  : out STD_LOGIC_VECTOR (13 downto 0)
                              );
      end component;
      component binBCD14 is
      Port (      b : in STD_LOGIC_VECTOR (13 downto 0);
                        p : out STD_LOGIC_VECTOR (16 downto 0));
      end component;
      component x7segbc is
      port( x       : in std_logic_vector (15 downto 0);
                  cclk : 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 b : std_logic_vector(13 downto 0);
      signal p : std_logic_vector(16 downto 0);
      signal clr, clk48, clk190 : std_logic;

      begin
      clr <= btn;

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



```
      u2:    mod10kcnt port map(
             clr => clr,
             clk => clk48,
             q => b
      );
      u3: binbcd14 port map(
             b => b,
             p => p
      );
      u4: x7segbc port map(
             x      => p(15 downto 0),
             cclk => clk190,
             clr     => clr,
             a_to_g => a_to_g,
             an     => an,
             dp => dp
      );

end Behavioral;
```

3. Pastikan program **Mod10Kcnt_top** menjadi top module.
4. Lakukan sintessis, sehingga menghasilkan RTL schematics seperti dibawah ini.

Hasil RTL Shcematics



5. Upload program ke board spartan, setelah sukses demokan hasilnya ke dosen pengampu

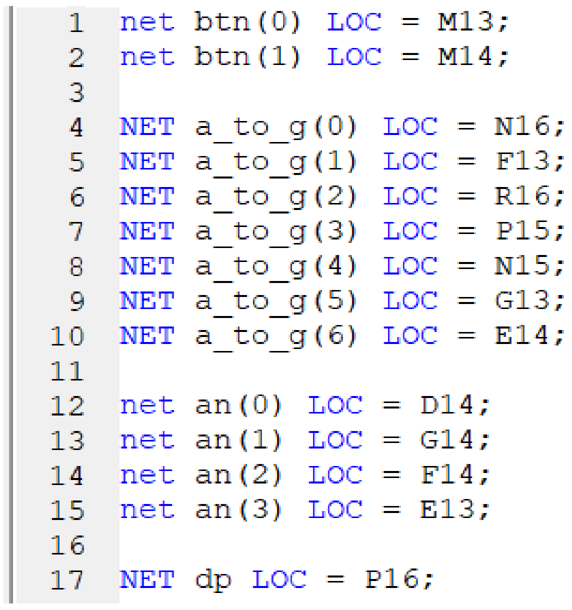
5. Latihan

Buatlah program stop watch, dengan menggunakcavni push button untuk tombol start, pause, reset.

6. Hasil Percobaan

a. Langkah Percobaan

Pendefinisian Pin



```
net btn(0) LOC = M13;
net btn(1) LOC = M14;

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;
```

RTL Skematik

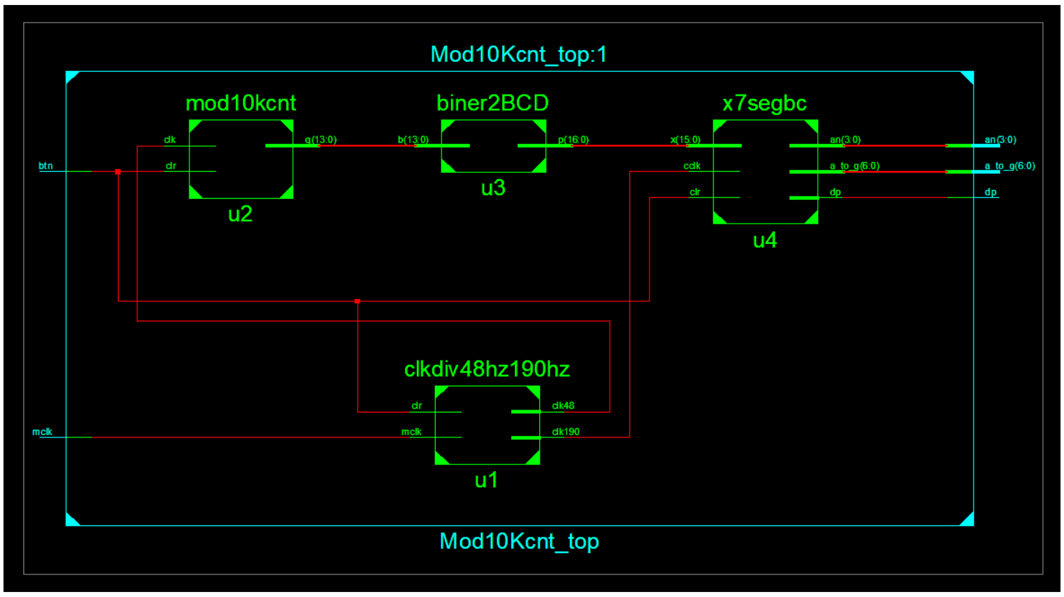
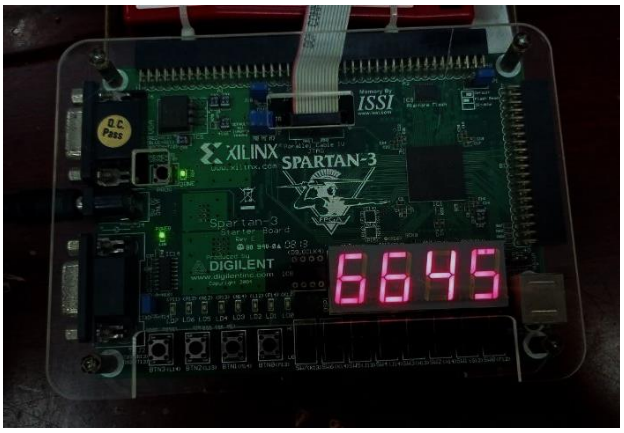
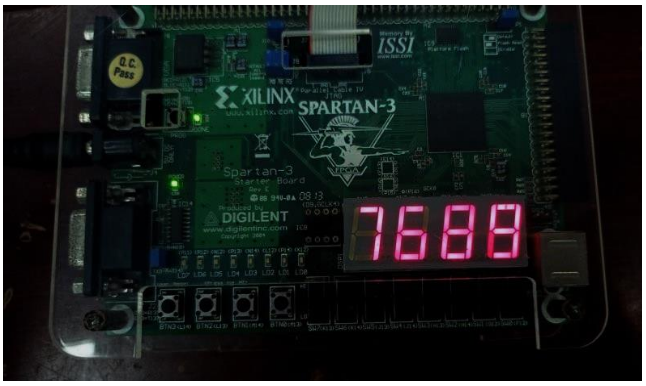


Foto Percobaan pada FPGA Board



Gambar Output 6645 pada 7 Segment Display



Gambar Output 7688 pada 7 Segment Display

b. Mini Project

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"; --format hexadesimal
             elsif mclk' event and mclk = '1' then
                    q <= q + 1;
             end if;
             clk48 <= q(19); clk190 <= q(17);
       end process;
end Behavioral ;
```

Modulus

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

entity mod10kcnt is
               Port (clr : in STD_LOGIC;
                              clk : in STD_LOGIC;
                              start : in STD_LOGIC;
                              pause : in STD_LOGIC;
                              q : out STD_LOGIC_VECTOR (13 downto 0));
end mod10kcnt;

architecture Behavioral of mod10kcnt is
               signal count:STD_LOGIC_VECTOR (13 downto 0);
               signal flagrun:std_logic;
begin
               -- modul o 10k
               process(clk,clr,start,pause)
               begin
                      -- reset and counter
                      if clr = '1' then
                              count <= (others => '0');
                      elsif clk' event and clk = '1' then
                              if start = '1' and pause = '0' then
                                     flagrun <= '1';
                              elsif start = '0' and pause = '1' then
                                     flagrun <= '0';
                              end if;
                              if conv_integer (count) = 9999 then
                                     count <= (others => '0');
                              elsif flagrun = '1' then
```

```
                                  count <= count + 1;
                           end if;
                    end if;
             end  process;
             q <= count ;
end Behavioral ;
```

Binary to BCD

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

entity binBCD14 is
       Port (b : in STD_LOGIC_VECTOR (13 downto 0);
                    p : out STD_LOGIC_VECTOR (16 downto 0));
end binBCD14;
architecture Behavioral of binBCD14 is
begin
       process(b)
       variable z : STD_LOGIC_VECTOR (32 downto 0);
       begin
       for i in 0  to 32 loop
              z (i ) := '0';
       end loop;
       z (16 downto 3) := b;
       for i in 0 to 10 loop
              if z (17 downto 14) > 4 then
                     z (17 downto 14) := z (17 downto 14) + 3;
              end if;
              if z (21 downto 18) > 4 then
                     z (21 downto 18) := z (21 downto 18) + 3;
              end if;
              if z (25 downto 22) > 4 then
                     z (25 downto 22) := z (25 downto 22) + 3;
              end if;
              if z (29 downto 26) > 4 then
                     z (29 downto 26) := z (29 downto 26) + 3;
              end if;
              z (32 downto 1) := z (31 downto 0);
       end loop;
       p <= z (30 downto 14);
       end process;
end Behavioral ;
```

Decoder 7 Segment

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

entity x7segbc  is
      port(x: in std_logic_vector (15 downto 0); cclk : 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 x7segbc ;

architecture Behavioral of x7segbc 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);
begin
      dp <= '1' ;
      --set aen( 3  downto 0)    f or l eadi ng bl anks
      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' ;

      --     Quad   4- t o- 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;
```

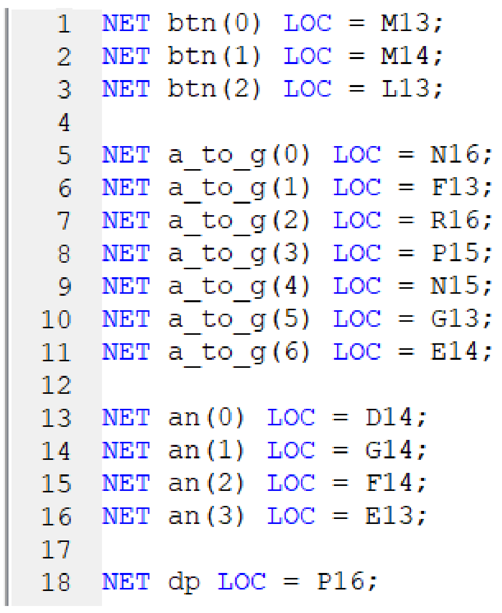
```
-- Di gi t sel ect : ancode
       process(s ,aen)
       beginan  <= "1111" ;
              if aen(conv_integer (s ))='1'then
                     an(conv_integer (s ))<='0' ;
              end if;
       end process;

       -- 2 bi t count er
       process(cclk, clr)
       begin
              if clr ='1'    then
                     s  <= "00" ;
              elsif cclk' event and  cclk='1'   then
                     s <= s + 1;
              end if;
       end process;
end Behavioral ;
```

Top Modul

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

entity mod10kcnt_top is
port( mclk : in std_logic;
               btn : in std_logic_vector(2 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 mod10kcnt_top;

architecture Behavioral of mod10kcnt_top is
               component clkdiv is
                       port( mclk : in std_logic;
                                       clr : in std_logic;
                                       clk190: out std_logic;
                                       clk48: out std_logic
                                );
end component;
               component mod10kcnt is
                       port( clr : in STD_LOGIC;
                                       clk : in STD_LOGIC;
                                       start : in std_logic;
                                       pause : in std_logic;
                                       q : out STD_LOGIC_VECTOR(13 downto 0)
                                );
end component;
               component binBCD14 is
                        Port ( b : in STD_LOGIC_VECTOR(13 downto 0);
                                        p : out STD_LOGIC_VECTOR(16 downto 0));
end component;
               component x7segbc is
                        port( x : in std_logic_vector(15 downto 0);
                                        cclk : 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;
```

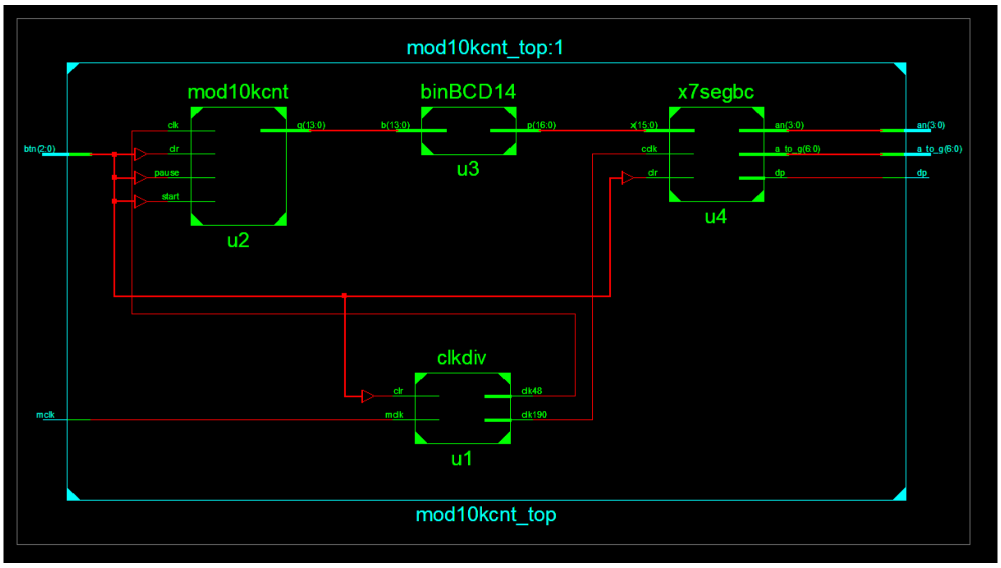
```vhdl
signal b : std_logic_vector(13 downto 0);
signal p : std_logic_vector(16 downto 0);
signal clr, start, pause, clk48, clk190 : std_logic;

begin
clr <= btn(0);
pause <= btn(1);
start <= btn(2);

u1: clkdiv port map(
             mclk => mclk,
             clr => clr,
             clk190 => clk190,
             clk48 => clk48
             );
u2: mod10kcnt port map(
             clr => clr,
             start => start,
             pause => pause,
             clk => clk48,
             q => b
             );
u3: binBCD14 port map(
             b => b,
             p => p
);
u4: x7segbc port map(
             x => p(15 downto 0),
             cclk => clk190,
             clr => clr,
             a_to_g => a_to_g,
             an => an,
             dp => dp
);
end Behavioral;
```

Pendefinisian Pin

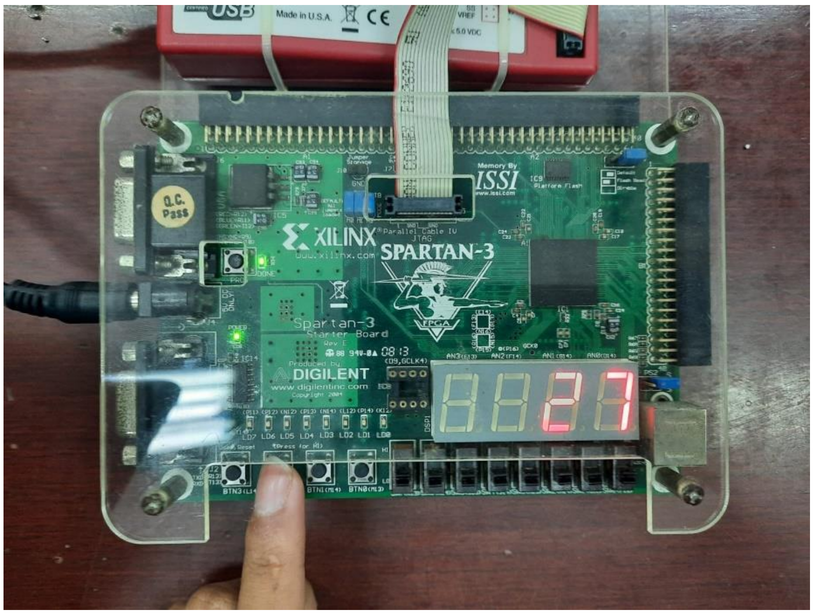


```
NET btn(0) LOC = M13;
NET btn(1) LOC = M14;
NET btn(2) LOC = L13;

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;
```

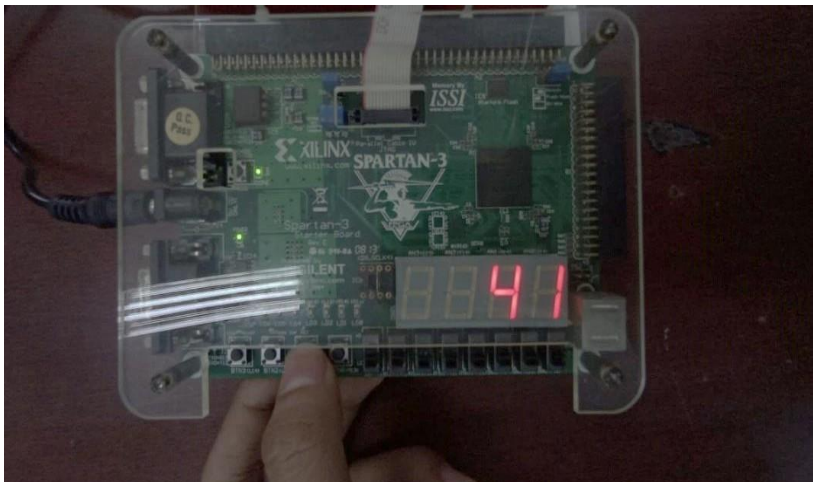
RTL Skematik



Hasil Pada FPGA Board



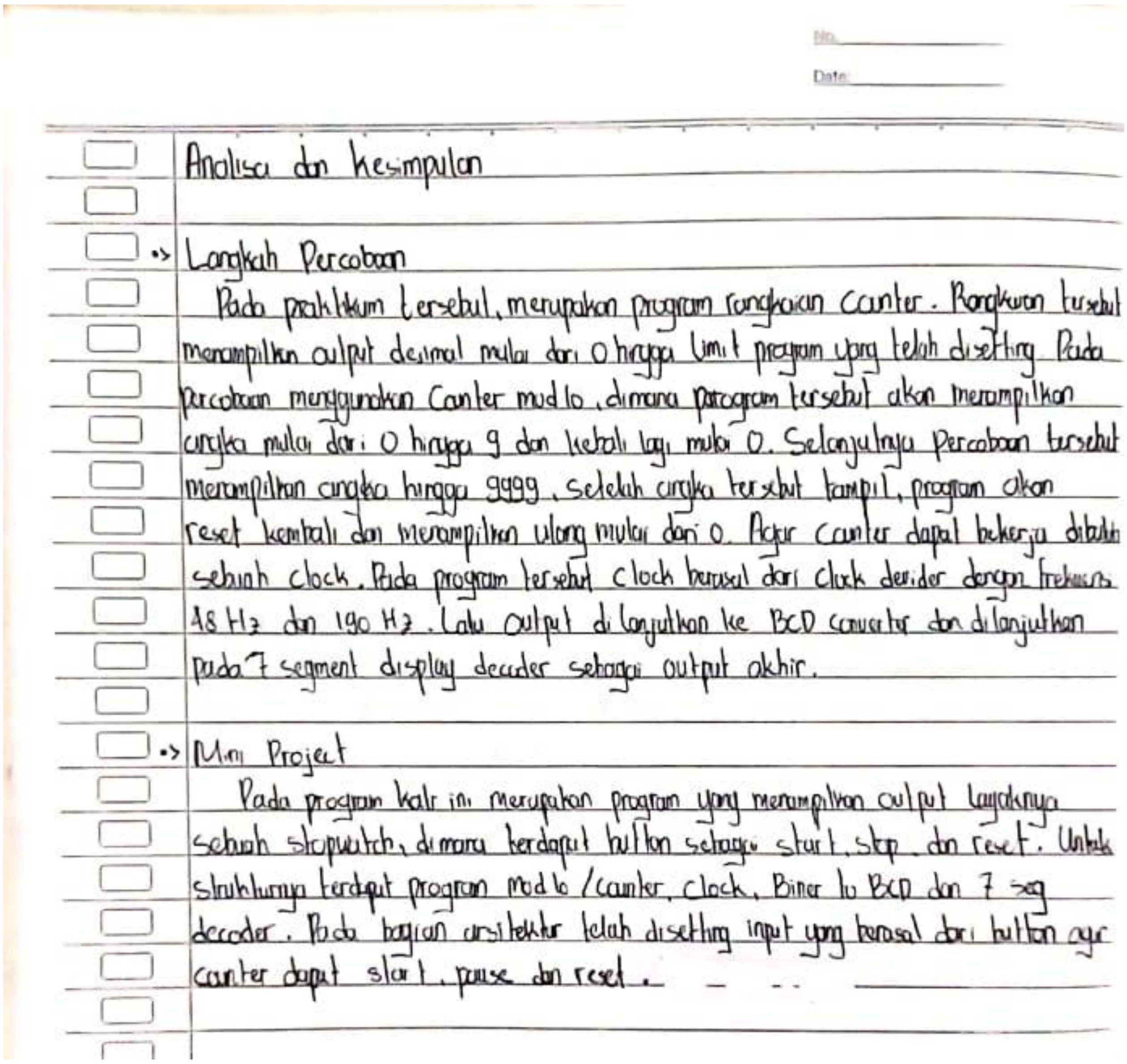
Gambar pada stopwatch sedang di mulai



Gambar pada stopwatch sedang di pause

Gambar pada stopwatch sedang di reset

7. Analisa



No.

Date:

Analisa dan kesimpulan

→ Langkah Percobaan

Pada praktikum tersebut, merupakan program rangkaian Canter. Rangkaian tersebut menampilkan output desimal mulai dari 0 hingga limit program yang telah disetting. Pada percobaan menggunakan Canter mod 10, dimana parogram tersebut akan menampilkan angka mulai dari 0 hingga 9 dan kembali lagi mulai 0. Selanjutnya percobaan tersebut menampilkan angka hingga 9999. setelah angka tersebut tampil, program akan reset kembali dan menampilkan ulang mulai dari 0. Agar canter dapat bekerja dibutuh sebuah clock. Pada program tersebut clock berasal dari Clock devider dengan frekuensi 48 Hz dan 190 Hz. Lalu output dilanjutkan ke BCD converter dan dilanjutkan pada 7 segment display decoder sebagai output akhir.

→ Mini Project

Pada program kali ini merupakan program yang menampilkan output layaknya sebuah stopwatch, dimana terdapat button sebagai start, stop, dan reset. Untuk struktur-nya terdapat program mod 10 / canter, clock, Biner ke BCD dan 7 seg decoder. Pada bagian arsitektur telah disetting input yang berasal dari button agar canter dapat start, pause dan reset.