```
entity Decoder is
port( input: in std_logic_vector (3 downto 0);
sevenseg: out std_logic_vector(6 downto 0));
end Decoder;

architecture Behavioral of Decoder is

signal sevens :std_logic_vector(6 downto 0);

begin

with input select
sevens <= "1111110" when "1001",
"0110000" when "1000",
"1101101" when "0111",
"1111001" when "0110",
"0110011" when "0101",
"1011011" when "0100",
"1011111" when "0011",
"1110000" when "0010",
"1111111" when "0001",
"1111011" when "0000",
"1000111" when others;

sevenseg <= not sevens;



end Behavioral;
```

Decoder is used to decode time to seven segments display.

This subprogram is really simple, just like the decoder program what we used many times in our lab before.

```
entity FSM is
port(resetn,clock,t1,t2,w,t3: in std_logic;
t: out std_logic;
led: out std_logic_vector(5 downto 0));
end FSM;

architecture Behavioral of FSM is
type state is (S1,S2,S3,S4,S5);
signal y: state;
begin
Transitions: process (resetn, clock)
begin
if resetn = '0' then y <= S1;
 elsif (clock'event and clock='1') then
   case y is
 when S1 =>
                        if w='1' then y<=S5;
                         elsif t1='0' then y<=S1;
                                          else y<=S2; end if;
  when S2 =>
                        if w='1' then y<=S5;
                         elsif t2='0' then y<=S2;
                        else  y<=S3; end if;
 when S3 =>
                        if w='1' then y<=S5;
                         elsif t1='0' then y<=S3;
                                          else  y<=S4; end if;
 when S4 =>
                        if w='1' then y<=S5;
                         elsif t2='0' then y<=S4;
                            else y<=S1; end if;
when S5 =>
                       if w='1' then y<=S5;
                        else y<=S1; end if;
                       end case;
end if;
end process;
```

```
Outputs: process (t2,w,y,t3)
begin

led<="000000";t<='1';

case y is
  when S1 =>
    led<="100001";
 if w='1' then t<='0'; end if;
  when S2 =>
    led<="010001";
 if w='1' then t<='0'; end if;
when S3 =>
    led<="001100";
 if w='1' then t<='0'; end if;
when S4 =>
    led<="001010";
if w='1' then t<='0'; end if;
when S5 =>
if t3='1' then
    led<="010010" ; end if;
if t3='0' then
 led<="000000";     end if;
if w='0' then t<='0'; end if;


 end case;
end process;
end Behavioral;
```

What we used to do about Timer function

```
if (clock'event and clock='1') then
  ts<=ts+1; end if;
if ts= 100000000
   ts<=0;
   t<=t+1; end if;
```

This did work in functional simulation.

But, this didn’t work in FPGA board. Variable “t” never changes.

Clock is 100MHZ, maybe because “ts=100000000” only lasts 10ms. It is hard for system to detect it .

So we need to use frequency division to get a 1HZ clock.

```
entity timer is
port(clock,resetn,en,t: in std_logic;
t1,t2,t3: out std_logic;
tout: out std_logic_vector (3 downto 0));
end timer;

architecture Behavioral of timer is
component my_genpulse
generic (COUNT: INTEGER:= (10**8));
port (clock, resetn, E: in std_logic;
Q: out std_logic_vector (integer(ceil(log2(real(COUNT)))) - 1 downto 0);
z: out std_logic);
end component;
signal z, z_0, z_1, z_2,E_1, E_2,clear: std_logic;
signal Q_0, Q_1, Q_2: std_logic_vector (3 downto 0);

begin
process(t,resetn)
begin
if(t='1' and resetn='1' ) then clear<='1';
  else clear<='0'; end if;
end process;
gz: my_genpulse generic map (COUNT => 10**6)
   port map (clock => clock, resetn => clear, E => en, z => z);

g0: my_genpulse generic map (COUNT => 10)
   port map (clock => clock, resetn => clear, E => z, Q => Q_0, z => z_0);

g1: my_genpulse generic map (COUNT => 10)
   port map (clock => clock, resetn => clear, E => E_1, Q => Q_1, z => z_1);
                        E_1 <= z and z_0;

g2: my_genpulse generic map (COUNT => 10)
   port map (clock => clock, resetn => clear, E => E_2, Q => Q_2, z => z_2);
                        E_2 <= E_1 and z_1;
```

- Used genpluse program from Moddle to make a clock
- Q_2 changes from 0 to 9 and change every one second

Generate Pulse Program
(Got from Moddle)

```
with Q_2 select
t1<='0' when
"0000",
'0' when "0001",
'0' when "0010",
'0' when "0011",
'0' when "0100",
'0' when "0101",
'0' when "0110",
'0' when "0111",
'1' when "1000",
'0' when "1001",
'0' when others;
```

```
with Q_2 select
t2<='1' when
"0000",
'0' when "0001",
'0' when "0010",
'0' when "0011",
'0' when "0100",
'0' when "0101",
'0' when "0110",
'0' when "0111",
'0' when "1000",
'0' when "1001",
'0' when others;
```

```
with Q_2 select
t3<='0' when "0000",
'1' when "0001",
'0' when "0010",
'1' when "0011",
'0' when "0100",
'1' when "0101",
'0' when "0110",
'1' when "0111",
'0' when "1000",
'1' when "1001",
'0' when others;
tout<=Q_2;
```

Q_2 is a time variable, which is used to select different value for t1,t2,t3.

t1,t2,t3 are variables that are used for FSM to change state.

```
entity topfile is
port( resetn,clock,w,en: in std_logic;
segmenten: out std_logic_vector(7 downto 0);
led:out std_logic_vector(5 downto 0);
segment1: out std_logic_vector(6 downto 0));
end topfile;
architecture Behavioral of topfile is
component Decoder
port( input: in std_logic_vector (3 downto 0);
sevenseg: out std_logic_vector(6 downto 0));
end component;

component FSM
port(resetn,clock,t1,t2,w,t3: in std_logic;
t: out std_logic;
led: out std_logic_vector(5 downto 0));
end component;

component timer
port(clock,resetn,en,t: in std_logic;
t1,t2,t3: out std_logic;
tout: out std_logic_vector (3 downto 0));
end component;

signal t1,t2,t,t3:std_logic;
signal segment: std_logic_vector(6 downto 0);
signal ts: std_logic_vector (3 downto 0);
begin
f0: FSM port map(resetn=>resetn,clock=>clock,t1=>t1,t2=>t2,w=>w,led=>led,t=>t,t3=>t3);
f1: timer port map(clock=>clock,resetn=>resetn,en=>en,t=>t,t1=>t1,t2=>t2,t3=>t3,tout=>ts);
f2: Decoder port map(input=>ts,sevenseg=>segment);
segment1<=segment;
process(w,en)
begin
if w='0' then
segmenten<="01110111"; else segmenten<="11111111"; end if;
if en='0' then segmenten<="11111111";end if;
end process;
end Behavioral;
```