```
LIBRARY ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.std_logic_misc.all;

entity top_10Obs is

port (
   CLOCK_50             :in       std_logic;
   KEY                  :in   std_logic_vector(3 downto 0) ;
   GPIO                 :out    std_logic_vector(34 downto 0) );

end entity;

-------------------------------------------------------------------------
------------------    ARCHITECTURE   --------------------------------
-------------------------------------------------------------------------
architecture rtl of top_10Obs is

constant  tau_range  :integer := 20;


component Altplc is
  PORT(
    areset      : IN STD_LOGIC  := '0';
    inclk0      : IN STD_LOGIC  := '0';
    c0          : OUT STD_LOGIC ; -- 50Mhz
    c1          : OUT STD_LOGIC ; -- 50Mhz
    c2          : OUT STD_LOGIC ; -- 100 Mhz
    c3          : OUT STD_LOGIC   -- 100 Mhz
  );
end component;



component signalgenerator is
  port(
   clk      :in  std_logic   := 'X';
   reset    :in  std_logic   := 'X';    -- clk
   output   :out std_logic -- export
);
end component signalgenerator;


component observer
    generic (
        observernumber :unsigned(15 downto 0):=x"0001"  -- how many observer are
 instantiated
      );
    PORT (
      clk           :in       std_logic        := 'X';
      reset         :in   std_logic         := 'X';
      enable_in          :in     std_logic;
      invariance_tau :in   std_logic_vector(7 downto 0);
      signal_phi     :in       std_logic;
      output         :out    std_logic;
      enable_out     :out  std_logic
      );
end component;

-------------------------------------------------------------------------
--  <BEGIN_0>
FOR OBS_0 : observer
  use entity work.observer(Behavioural);
FOR OBS_1 : observer
 use entity work.observer(Behavioural);
FOR OBS_2 : observer
 use entity work.observer(Behavioural);
FOR OBS_3 : observer
 use entity work.observer(Behavioural);
FOR OBS_4 : observer
 use entity work.observer(Behavioural);
FOR OBS_5 : observer
 use entity work.observer(Behavioural);
FOR OBS_6 : observer
 use entity work.observer(Behavioural);
FOR OBS_7 : observer
 use entity work.observer(Behavioural);
FOR OBS_8 : observer
 use entity work.observer(Behavioural);
FOR OBS_9 : observer
 use entity work.observer(Behavioural);

-- <END_0>
-------------------------------------------------------------------------

signal clk_s              : std_logic          :='0';
signal clk_g              : std_logic          :='0';
signal reset_s            : std_logic          :='0';
signal enable_s            : std_logic          :='0';
signal phi_s              : std_logic          :='0';
signal next_obs_s  : std_logic                 :='0';
-------------------------------------------------------------------------
-- <BEGIN_1>
signal add: std_logic_vector(9 downto 0):=(others=>'0');
signal en1          :std_logic:='0';
signal en2          :std_logic:='0';
signal en3          :std_logic:='0';
signal en4          :std_logic:='0';
signal en5          :std_logic:='0';
signal en6          :std_logic:='0';
signal en7          :std_logic:='0';
signal en8          :std_logic:='0';
signal en9          :std_logic:='0';

-- <END_1>
-------------------------------------------------------------------------
signal output_s           : std_logic       :='0';
signal tau_s          : std_logic_vector(7 downto 0) := (others => '0');


-------------------------------------------------------------------------
-----
----- BEGIN OF ARCHITECTURE ----------------------------------------------
-
-------------------------------------------------------------------------
---
begin

  signalgenerator_top : component signalgenerator
    port map (
      output => phi_s,
      reset => reset_s,
      clk => clk_g
      );

  PLL: component AltPLc
  PORT MAP (areset => reset_s,inclk0 => CLOCK_50 ,c1 => clk_g,c0 =>clk_s) ;

-------------------------------------------------------------------------
-- <BEGIN_2>
OBS_0:  observer GENERIC MAP(observernumber => x"000A")
    PORT MAP ( output=>add(0),clk=>clk_s,reset =>reset_s, enable_in =>enable_s,invariance_tau => tau_s,
signal_phi=> phi_s,enable_out=>en1) ;
OBS_1:  observer GENERIC MAP(observernumber => x"000A")
    PORT MAP ( output=>add(1),clk=>clk_s,reset =>reset_s, enable_in =>en1,invariance_tau => tau_s,signa
l_phi=> phi_s,enable_out=>en2) ;
OBS_2:  observer GENERIC MAP(observernumber => x"000A")
    PORT MAP ( output=>add(2),clk=>clk_s,reset =>reset_s, enable_in =>en2,invariance_tau => tau_s,signa
l_phi=> phi_s,enable_out=>en3) ;
OBS_3:  observer GENERIC MAP(observernumber => x"000A")
    PORT MAP ( output=>add(3),clk=>clk_s,reset =>reset_s, enable_in =>en3,invariance_tau => tau_s,signa
```