

```

1  LIBRARY ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  use IEEE.std_logic_misc.all;
5
6  entity top_10Obs is
7
8  port (
9      CLOCK_50          :in      std_logic;
10     KEY                :in      std_logic_vector(3 downto 0) ;
11     GPIO               :out      std_logic_vector(34 downto 0) );
12
13 end entity;
14
15 -----
16 ----- ARCHITECTURE -----
17 -----
18 architecture rtl of top_10Obs is
19
20     constant tau_range :integer := 20;
21
22
23     component Altplc is
24     PORT (
25         areset          : IN STD_LOGIC  := '0';
26         inclk0          : IN STD_LOGIC  := '0';
27         c0              : OUT STD_LOGIC ; -- 50Mhz
28         c1              : OUT STD_LOGIC ; -- 50Mhz
29         c2              : OUT STD_LOGIC ; -- 100 Mhz
30         c3              : OUT STD_LOGIC  -- 100 Mhz
31     );
32 end component;
33
34
35
36     component signalgenerator is
37     port (
38         clk             :in      std_logic  := 'X';
39         reset           :in      std_logic  := 'X'; -- clk
40         output          :out      std_logic -- export
41     );
42 end component signalgenerator;
43
44
45     component observer
46     generic (
47         observernumber :unsigned(15 downto 0):=x"0001" -- how many observer are
         instantiated
48     );
49     PORT (
50         clk             :in      std_logic          := 'X';
51         reset           :in      std_logic          := 'X';
52         enable_in       :in      std_logic;
53         invariance_tau  :in      std_logic_vector(7 downto 0);
54         signal_phi      :in      std_logic;
55         output          :out      std_logic;
56         enable_out      :out      std_logic
57     );
58 end component;
59
60 -----
61 -- <BEGIN_0>
62 FOR OBS_0 : observer
63     use entity work.observer(Behavioural);
64 FOR OBS_1 : observer
65     use entity work.observer(Behavioural);
66 FOR OBS_2 : observer
67     use entity work.observer(Behavioural);
68 FOR OBS_3 : observer
69     use entity work.observer(Behavioural);
70 FOR OBS_4 : observer

```

```

71     use entity work.observer(Behavioural);
72 FOR OBS_5 : observer
73     use entity work.observer(Behavioural);
74 FOR OBS_6 : observer
75     use entity work.observer(Behavioural);
76 FOR OBS_7 : observer
77     use entity work.observer(Behavioural);
78 FOR OBS_8 : observer
79     use entity work.observer(Behavioural);
80 FOR OBS_9 : observer
81     use entity work.observer(Behavioural);
82
83 -- <END_0>
84 -----
85
86 signal clk_s          : std_logic          :='0';
87 signal clk_g          : std_logic          :='0';
88 signal reset_s        : std_logic          :='0';
89 signal enable_s       : std_logic          :='0';
90 signal phi_s          : std_logic          :='0';
91 signal next_obs_s     : std_logic          :='0';
92 -----
93 -- <BEGIN_1>
94 signal add: std_logic_vector(9 downto 0):=(others=>'0');
95 signal en1            :std_logic:= '0';
96 signal en2            :std_logic:= '0';
97 signal en3            :std_logic:= '0';
98 signal en4            :std_logic:= '0';
99 signal en5            :std_logic:= '0';
100 signal en6            :std_logic:= '0';
101 signal en7            :std_logic:= '0';
102 signal en8            :std_logic:= '0';
103 signal en9            :std_logic:= '0';
104
105 -- <END_1>
106 -----
107 signal output_s       : std_logic          :='0';
108 signal tau_s          : std_logic_vector(7 downto 0) := (others => '0');
109
110
111 -----
112 ----- BEGIN OF ARCHITECTURE -----
113 -----
114 begin
115
116     signalgenerator_top : component signalgenerator
117     port map (
118         output => phi_s,
119         reset => reset_s,
120         clk => clk_g
121     );
122
123     PLL: component AltPLC
124     PORT MAP (areset => reset_s,inclck0 => CLOCK_50 ,c1 => clk_g,c0 =>clk_s) ;
125
126 -----
127 -- <BEGIN_2>
128 OBS_0: observer GENERIC MAP(observernumber => x"000A")
129     PORT MAP ( output=>add(0),clk=>clk_s,reset =>reset_s, enable_in =>enable_s,invariance_tau => tau_s,
130         signal_phi=> phi_s,enable_out=>en1) ;
131 OBS_1: observer GENERIC MAP(observernumber => x"000A")
132     PORT MAP ( output=>add(1),clk=>clk_s,reset =>reset_s, enable_in =>en1,invariance_tau => tau_s,signa
133         l_phi=> phi_s,enable_out=>en2) ;
134 OBS_2: observer GENERIC MAP(observernumber => x"000A")
135     PORT MAP ( output=>add(2),clk=>clk_s,reset =>reset_s, enable_in =>en2,invariance_tau => tau_s,signa
136         l_phi=> phi_s,enable_out=>en3) ;
137 OBS_3: observer GENERIC MAP(observernumber => x"000A")
138     PORT MAP ( output=>add(3),clk=>clk_s,reset =>reset_s, enable_in =>en3,invariance_tau => tau_s,signa

```