

```

136 I_phi=> phi_s.enable_out=>en4);
137 OBS 4: observer GENERIC MAP(observernumber => x"000A")
138   PORT MAP ( output=>add(4),clk=>clk_s,reset =>reset_s, enable_in =>en4,invariance_tau => tau_s,signa
139   I_phi=> phi_s.enable_out=>en5);
138 OBS 5: observer GENERIC MAP(observernumber => x"000A")
139   PORT MAP ( output=>add(5),clk=>clk_s,reset =>reset_s, enable_in =>en5,invariance_tau => tau_s,signa
140   I_phi=> phi_s.enable_out=>en6);
141 OBS 6: observer GENERIC MAP(observernumber => x"000A")
142   PORT MAP ( output=>add(6),clk=>clk_s,reset =>reset_s, enable_in =>en6,invariance_tau => tau_s,signa
143   I_phi=> phi_s.enable_out=>en7);
142 OBS 7: observer GENERIC MAP(observernumber => x"000A")
143   PORT MAP ( output=>add(7),clk=>clk_s,reset =>reset_s, enable_in =>en7,invariance_tau => tau_s,signa
144   I_phi=> phi_s.enable_out=>en8);
144 OBS 8: observer GENERIC MAP(observernumber => x"000A")
145   PORT MAP ( output=>add(8),clk=>clk_s,reset =>reset_s, enable_in =>en8,invariance_tau => tau_s,signa
146   I_phi=> phi_s.enable_out=>en9);
146 OBS 9: observer GENERIC MAP(observernumber => x"000A")
147   PORT MAP ( output=>add(9),clk=>clk_s,reset =>reset_s, enable_in =>en9,invariance_tau => tau_s,signa
148   I_phi=> phi_s.enable_out=> next_obs_s);
149
150 --- <END_2>
-----

151
152
153 --- <BEGIN_3>
154 output_s <= and_reduce(add);
155 --- <END_3>
-----

156
157
158
159 -----FPGA OUTPUTS-----
-----

161
162
163 reset_s <= not KEY(0);
164
165 GPIO(0) <= reset_s;
166 GPIO(1) <= enable_s;
167 GPIO(2) <= en1;
168 GPIO(3) <= en2;
169 GPIO(4) <= en3;
170 GPIO(5) <= en4;
171 GPIO(6) <= en5;
172 GPIO(7) <= en6;
173 GPIO(8) <= en7;
174 GPIO(9) <= en8;
175 GPIO(10) <= en9;
176 GPIO(11) <= next_obs_s;
177 GPIO(12) <= clk_g;
178 GPIO(13) <= phi_s;
179 GPIO(14) <= clk_s;
180
181 GPIO(15) <= add(0);
182 GPIO(16) <= add(1);
183 GPIO(17) <= add(2);
184 GPIO(18) <= add(3);
185 GPIO(19) <= add(4);
186 GPIO(20) <= add(5);
187 GPIO(21) <= add(6);
188 GPIO(22) <= add(7);
189 GPIO(23) <= add(8);
190 GPIO(24) <= add(9);
191 GPIO(25) <= output_s;
192
193 tau_s      <= std_logic_vector(to_unsigned(tau_range,8));
194
195 -----SYNCHRONIZED-----
-----

196 sync:process(clk_s)

```

```

197 begin
198   if(clk_s'event and clk_s='1') then
199     if reset_s = '0' then
200       enable_s <= '1';
201     else
202       enable_s <= '0';
203     end if;
204   end if;
205 end process;
206
207 end architecture;

```