```vhdl

LIBRARY ieee;
USE ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;


architecture Behavioural of observer is

signal inc_tau                              : unsigned(8 downto 0):= "000000000";
signal count,count_next                 : unsigned(15 downto 0):= x"0001";
signal count_p,count_p_next              : unsigned(15 downto 0):= x"0002";
signal cycle,cycle_next                : unsigned(15 downto 0) := x"0000";
signal direction,direction_next        : std_logic := '1';


signal enable_logic                    : std_logic := '0';
signal output_next                              : std_logic := '0';
begin --BEGIN ARCHITECTURE

-- parallel logic
inc_tau <= unsigned(invariance_tau) + to_unsigned(1,9) ;
enable_logic <= enable_in and not reset;

-- changes cycle up from 0 to observernumber and down back to 0
comb_cycle: process(cycle,direction,enable_logic)
begin --changes cycle_next, direction, changeDirection
  if(direction = '0' and enable_logic = '1') then
    if(cycle = 0)then
      direction_next <= '1';
      cycle_next <= cycle + 1;
    else
      direction_next <= '0';
      cycle_next <= cycle - 1;
    end if;
  elsif(direction = '1' and enable_logic = '1') then
    if(cycle = observernumber)then
      direction_next <= '0';
      cycle_next <= cycle - 1;
    else
      direction_next <= '1';
      cycle_next <= cycle + 1;
    end if;
  else
    direction_next <= direction;
    cycle_next <= cycle;
  end if;
end process comb_cycle;


 -- main logic of the observer
comb_logic: process(inc_tau,count,count_p,cycle,signal_phi,enable_logic)
begin
  if ( (cycle = observernumber  or cycle = 0) and enable_logic = '1') then -- m cycles passed
    if(signal_phi = '0') then   -- w(phi) = 0
         count_next   <= x"0001";
      count_p_next <= x"0002";
      output_next  <= '0';
    elsif(count_p <= inc_tau) then
      count_next   <= count   + 1; --every clock cycle
      count_p_next <= count_p + 1 ;
        output_next       <= '0';
    else
      count_next   <= count;
      count_p_next <= count_p;
      output_next       <= '1';
    end if;
  elsif(count_p <= inc_tau and enable_logic = '1') then
    count_next   <= count + 1; --every clock cycle
    count_p_next <= count_p + 1 ;
      output_next       <= '0';
  elsif(count_p > inc_tau and enable_logic = '1') then
    count_next   <= count;
    count_p_next <= count_p;
    output_next  <= '1';
  else
    count_next   <= count;
    count_p_next <= count_p;
    output_next  <= '0';
  end if;
end process comb_logic;

  --the synchronisation logic
sync: process(clk,enable_logic)
  begin
    if(clk'event and clk = '0')then
      if(enable_logic = '1') then
        cycle           <= cycle_next;
        direction       <= direction_next;
        count           <= count_next;
        count_p            <= count_p_next;
            output    <= output_next;
        enable_out      <= '1';
      else
        cycle           <= x"0000";
            direction       <='1';
        count   <= x"0001";
        count_p <= x"0002";
            output    <= '0';
            enable_out <='0';
      end if;
    end if;
  end process sync;
end architecture ; --END ARCHITECTURE
```