```vhdl
library STD;
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_textio.all;
use STD.textio.all;

entity Counter_Test is

end Counter_Test;

architecture test of Counter_Test is

    component Counter
    port(
        clk         :   in  std_logic;
        hit_miss    :   in  std_logic;  --1 hit 0 miss
        rd_wr       :   in  std_logic;  --1 read 0 write
        start       :   in  std_logic;
        Vdd         :   in  std_logic;
        Gnd         :   in  std_logic;
        reset       :   in  std_logic;
        busy        :   out std_logic;
        rd_wr_o     :   out std_logic;
        cache_write :   out std_logic;
        rm_wr_en    :   out std_logic;
        wr_hit      :   out std_logic;
        cpu_dout_en :   out std_logic;
        mem_enable  :   out std_logic;
        write_0     :   out std_logic;  --write word0 to cache
        write_1     :   out std_logic;  --write word1 to cache
        write_2     :   out std_logic;  --write word2 to cache
        write_3     :   out std_logic  --write word3 to cache
    );
    end component;

    signal s_clk        :   std_logic;
    signal s_hit_miss   :   std_logic   := '0';
    signal s_rd_wr      :   std_logic   := 'Z';
    signal s_start      :   std_logic   := '0';
    signal s_reset      :   std_logic;
    signal s_busy       :   std_logic;
    signal s_rd_wr_o    :   std_logic;
    signal s_cache_write:   std_logic;
    signal s_rm_wr_en   :   std_logic;
    signal s_wr_hit     :   std_logic;
    signal s_cpu_dout_en:   std_logic;
    signal s_mem_enable :   std_logic;
    signal s_write_0    :   std_logic;
    signal s_write_1    :   std_logic;
    signal s_write_2    :   std_logic;
    signal s_write_3    :   std_logic;

    shared variable done    :   boolean := false;

    for asdt    :   Counter use entity work.Counter(structural);

begin

    asdt    :   Counter port map(
        clk         =>  s_clk,
        hit_miss    =>  s_hit_miss,
        rd_wr       =>  s_rd_wr,
        start       =>  s_start,
        Vdd         =>  '1',
        Gnd         =>  '0',
        reset       =>  s_reset,
        busy        =>  s_busy,
        rd_wr_o     =>  s_rd_wr_o,
        cache_write =>  s_cache_write,
        rm_wr_en    =>  s_rm_wr_en,
        wr_hit      =>  s_wr_hit,
        cpu_dout_en =>  s_cpu_dout_en,
        mem_enable  =>  s_mem_enable,
        write_0     =>  s_write_0,
        write_1     =>  s_write_1,
        write_2     =>  s_write_2,
        write_3     =>  s_write_3
    );

    c       :   process
    begin
            if done then
                wait;
            else
                s_clk <= '1', '0' after 5 ns;
                wait for 10 ns;
            end if;
    end process;

    main    :   process

        variable v_clk          :   std_logic;
        variable v_hit_miss     :   std_logic;
        variable v_rd_wr        :   std_logic;
        variable v_busy         :   std_logic;
        variable v_start        :   std_logic;
        variable v_write_0      :   std_logic;
        variable v_write_1      :   std_logic;
        variable v_write_2      :   std_logic;
        variable v_write_3      :   std_logic;

    begin
        s_reset     <= '1';
        wait for 20 ns;
        s_reset     <= '0';
        wait for 10 ns;
        s_rd_wr     <= '1';
        s_start     <= '1';
        wait for 10 ns;
        s_rd_wr     <= 'Z';
        s_start     <= '0';
        s_hit_miss  <= '0';
        wait for 190 ns;
        s_start     <= '1';
        s_rd_wr     <= '1';
        wait for 10 ns;
        s_rd_wr     <= 'Z';
        s_start     <= '0';
        s_hit_miss  <= '1';
        wait for 40 ns;
        s_rd_wr     <= '0';
        s_start     <= '1';
        s_hit_miss  <= '0';
        wait for 10 ns;
        s_rd_wr     <= 'Z';
        s_start     <= '0';
        wait for 50 ns;
        s_rd_wr     <= '0';
        s_start     <= '1';
        s_hit_miss  <= '1';
        wait for 10 ns;
        s_rd_wr     <= 'Z';
        s_start     <= '0';
        wait for 50 ns;
        done := true;
        wait;
    end process;
end test;
```