```
----------------------
--generic counter
--
-- 00 holds
-- 01 loads
-- 10 counts up
-- 11 resets
----------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity counter is
    generic(N: integer:=4);
    port(clk,reset : in std_logic;
     c: in std_logic_vector(1 downto 0);
     d : in  std_logic_vector(N-1 downto 0);
     q : out std_logic_vector(N-1 downto 0));
end counter;

architecture behavior of counter is
    signal tmp: std_logic_vector(N-1 downto 0);
begin
    process (clk, reset)
    begin
    if (reset='0') then
        tmp <= (others => '0');
    elsif (clk'event and clk='0')  then
    case c is
        when "00" => tmp <= tmp;
        when "01" => tmp <= d;
        when "10" => tmp <= tmp+1;
        when others => tmp <= (others => '0');
    end case;
    end if;
    end process;
    q <= tmp;
end behavior;

---------------------
--1 bit counter
--
-- 00 holds
-- 01 loads
-- 10 counts up
-- 11 resets
---------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity counter1 is
    port(clk,reset : in std_logic;
     c: in std_logic_vector(1 downto 0);
```

```
     d : in  std_logic;
     q : out std_logic);
end counter1;

architecture behavior of counter1 is
    signal tmp: std_logic;
begin
    process (clk, reset)
    begin
    if (reset='0') then
        tmp <= '0';
    elsif (clk'event and clk='0')  then
    case c is
        when "00" => tmp <= tmp;
        when "01" => tmp <= d;
        when "10" =>
            if (tmp='1') then tmp <= '0';
            elsif (tmp='0') then tmp <='1';end if;
        when others => tmp <= '0';
    end case;
    end if;
    end process;
    q <= tmp;
end behavior;

----------------------
--saturation counter 3-bit
--
-- 00 holds
-- 01 loads
-- 10 counts up
-- 11 resets
----------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity satcount3 is
    port(clk,reset : in std_logic;
     c: in std_logic_vector(1 downto 0);
     d : in  std_logic_vector(2 downto 0);
     q : out std_logic_vector(2 downto 0));
end satcount3;

architecture behavior of satcount3 is
    signal tmp: std_logic_vector(2 downto 0);
begin
    process (clk, reset)
    begin
    if (reset='0') then
        tmp <= (others => '0');
    elsif (clk'event and clk='0')  then
    case c is
```

```
        when "00" => tmp <= tmp;
        when "01" => tmp <= d;
        when "10" =>
            if (tmp="111") then tmp <= tmp;
            else tmp <= tmp+1; end if;
        when others => tmp <= (others => '0');
    end case;
    end if;
    end process;
    q <= tmp;
end behavior;
```