```vhdl
-------------------------------------------
-- Block code:  count_down.vhd
-- History:     12.Nov.2013 - 1st version (dqtm)
--                 <date> - <changes>  (<author>)
-- Function: down-counter, with start input and count output.
--          The input start should be a pulse which causes the
--          counter to load its max-value. When start is off,
--          the counter decrements by one every clock cycle till
--          count_o equals 0. Once the count_o reaches 0, the counter
--          freezes and wait till next start pulse.
--          Can be used as enable for other blocks where need to
--          count number of iterations.
-------------------------------------------


-- Library & Use Statements
-------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;


-- Entity Declaration
-------------------------------------------
ENTITY count_down IS
GENERIC (width : positive := 4);
  PORT( clk,reset_n     : IN    std_logic;
        start_i         : IN    std_logic;
        count_o         : OUT   std_logic_vector(width-1 downto 0)
        );
END count_down;


-- Architecture Declaration
-------------------------------------------
ARCHITECTURE rtl OF count_down IS
-- Signals & Constants Declaration
-------------------------------------------
CONSTANT    max_val:            unsigned(width-1 downto 0):= to_unsigned(4,width); -- convert integer value 4 to unsigned with 4bits
SIGNAL      count, next_count:  unsigned(width-1 downto 0);


-- Begin Architecture
-------------------------------------------
BEGIN


  --------------------------------------------------
  -- PROCESS FOR COMBINATORIAL LOGIC
  --------------------------------------------------
  comb_logic: PROCESS(start_i,count)
  BEGIN
    -- load
    IF (start_i = '1') THEN
        next_count <= max_val;

    -- decrement
    ELSIF (count > 0) THEN
        next_count <= count - 1 ;

    -- freezes
    ELSE
        next_count <= count;
    END IF;

  END PROCESS comb_logic;



  --------------------------------------------------
  -- PROCESS FOR REGISTERS
  --------------------------------------------------
  flip_flops : PROCESS(clk, reset_n)
  BEGIN
    IF reset_n = '0' THEN
        count <= to_unsigned(0,width); -- convert integer value 0 to unsigned with 4bits
    ELSIF rising_edge(clk) THEN
        count <= next_count ;
    END IF;
  END PROCESS flip_flops;


  --------------------------------------------------
  -- CONCURRENT ASSIGNMENTS
  --------------------------------------------------
  -- convert count from unsigned to std_logic (output data-type)
  count_o <= std_logic_vector(count);


 -- End Architecture
-------------------------------------------
END rtl;


```