```vhdl
-------------------------------------------
-- Block code:  shiftreg_p2s.vhd
-- History:     12.Nov.2013 - 1st version (dqtm)
--                 <date> - <changes>  (<author>)
-- Function: shift-register working as a parallel to serial converter.
--          The block has a load( or shift_n) control input, plus a parallel data input.
--          If load is high the parallel data is loaded, and if load is low the data is shifted towards the LSB.
--          During shift the MSB gets the value of '1'.
--          The serial output is the LSB of the shiftregister.
--          Can be used as P2S in a serial interface, where inactive value (or rest_value)  equals '1'.
-------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY shiftreg_p2s IS
  PORT( clk,set_n       : IN    std_logic;          -- Attention, this block has a set_n input for initialisation!!
        load_i          : IN    std_logic;
        par_i           : IN    std_logic_vector(3 downto 0);
        ser_o           : OUT   std_logic
        );
END shiftreg_p2s;

ARCHITECTURE rtl OF shiftreg_p2s IS
-- Signals & Constants Declaration
-------------------------------------------
    SIGNAL      shiftreg, next_shiftreg:    std_logic_vector(4 downto 0);    -- add one FF for start_bit

BEGIN

  --------------------------------------------------
  -- PROCESS FOR COMBINATIONAL LOGIC
  --------------------------------------------------
  shift_comb: PROCESS(ALL)
  BEGIN
    IF (load_i = '1') THEN            -- load parallel data + add start_bit
        next_shiftreg <= par_i & '0'; -- LSB='0' is the start_bit

    ELSE                              -- shift; shift direction towards LSB
        next_shiftreg <= '1' & shiftreg(4 downto 1);
    END IF;

  END PROCESS shift_comb;

  --------------------------------------------------
  -- PROCESS FOR REGISTERS
  --------------------------------------------------
  shift_dffs : PROCESS(clk, set_n)
  BEGIN
    IF set_n = '0' THEN
```

```
        shiftreg <= (others=>'1');
    ELSIF rising_edge(clk) THEN
        shiftreg <= next_shiftreg ;
    END IF;
  END PROCESS shift_dffs;

  --------------------------------------------------
  -- CONCURRENT ASSIGNMENTS
  --------------------------------------------------
  -- take LSB of shiftreg as serial output
  ser_o <= shiftreg(0);

END rtl;
```