

-- **** STUDENT: 64000225.....	3
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	3
-- **** STUDENT: 64190088.....	5
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	5
-- **** STUDENT: 64200100.....	7
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	7
-- **** STUDENT: 64200112.....	9
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	9
-- **** STUDENT: 64200163.....	11
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	11
-- **** STUDENT: 64200238.....	13
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Operacije ne delujejo, ker je zamenjana polariteta nCLR signala. .	13
--nCLR_neg <= not nCLR; -- sklepam da je to ta PRESET MIŠLJEN KAO POSTAVLJEN NA '1'	13
nCLR_neg <= nCLR; -- narobe ste sklepali	13
-- **** STUDENT: 64200288.....	16
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Operacije ne delujejo, ker je zamenjana polariteta nCLR signala. .	16
--nCLR_neg <= not nCLR; -- sklepam da je to ta PRESET MIŠLJEN KAO POSTAVLJEN NA '1'	16
nCLR_neg <= nCLR; -- narobe ste sklepali	16
-- *****	16
-- **** STUDENT: 64200296.....	18
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	18
-- **** STUDENT: 64200385.....	20
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	20
-- **** STUDENT: 64210113.....	22
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	22
-- **** STUDENT: 64210132.....	24
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	24
-- **** STUDENT: 64210290.....	26
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	26
-- **** STUDENT: 64210382.....	28
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	28
-- **** STUDENT: 64210384.....	30
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	30
-- **** STUDENT: 64210386.....	32

-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	32
-- **** STUDENT: 64210445.....		34
-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	34
-- **** STUDENT: 64210455.....		36
-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	36
-- **** STUDENT: 64210457.....		38
-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	38
-- **** STUDENT: 64240430.....		40
-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	40
-- **** PREDLOGA VAJE		42
-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	42

```

-- *****
-- **** STUDENT: 64000225
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prožen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

    COMPONENT muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT (
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2**n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk,      -- signal      ure ( prožen na sprednjo fronto )
            nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
            Q : out std_logic -- izhod ULM strukture
        );
    END COMPONENT;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal D : dff_mux_in := ( others => ( others => '0' ) );
    signal Q_sig : std_logic_vector( reg_size - 1 downto 0 );

BEGIN

```

```

FOR1: FOR i IN 0 TO reg_size-1 GENERATE
    u1: muxdff
    generic map( n_addr => 2 )
    port map (
        S => s,          D => D( i ),          clk => clk,          nCLEAR => nCLR,          Q => Q_sig( i
    );

    D( i )( 3 ) <= x( i );

    with i select D( i )( 2 ) <=
        sl_in when 0,          Q_sig( i-1 ) when others;

    D( i )( 1 ) <= sr_in when i=reg_size-1 else Q_sig( i+1 );
    D( i )( 0 ) <= Q_sig( i );
END GENERATE;

Q    <= Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64190088
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,    -- signal      ure ( prozen na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,  -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal
        D : dff_mux_in := ( others => ( others => '0' ) );

    component muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT (
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2*n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk,    -- signal      ure ( prožen na sprednjo fronto )
            nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
            Q : out std_logic -- izhod ULM strukture
        );
    end component;

    signal
        Q_sig: std_logic_vector( reg_size - 1 downto 0 );

```

BEGIN

```
D( 0 ) <=    x( 0 ) &      -- ( operacija 11 ) - nalaganje vhoda x( LSB )
            sl_in &      -- ( operacija 10 ) - pomik levo ( sl vhod gre v mesto LSB )
            Q_sig( 1 ) & -- ( operacija 01 ) - pomik desno ( mesto 1 gre v mesto LSB )
            Q_sig( 0 ); -- ( operacija 00 ) - držanje vsebine LSB
U0: muxdff generic map ( n_addr => 2 ) port map ( S, D( 0 ), clk, nCLR, Q_sig( 0 ) );

D( reg_size-1 ) <=    x( reg_size-1 ) &
                    Q_sig( reg_size-2 ) &
                    sr_in &
                    Q_sig( reg_size-1 );
U1: muxdff generic map ( n_addr => 2 ) port map ( S, D( reg_size - 1 ), clk, nCLR, Q_sig( reg_size - 1 ) );

zanka_for_gen: for i in ( reg_size-2 ) downto 1 generate
    D( i ) <= X( i ) & Q_sig( i-1 ) & Q_sig( i+1 ) & Q_sig( i );
    U2: muxdff generic map ( n_addr => 2 ) port map ( S, D( i ), clk, nCLR, Q_sig( i ) );
end generate;

Q    <= Q_sig;

end NDV;
```

```

-- *****
-- **** STUDENT: 64200100
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prozen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

    COMPONENT muxdff IS
        generic( n_addr: natural := 4 );
        PORT ( S: in std_logic_vector( n_addr - 1 downto 0 );
              D: in std_logic_vector( 2**n_addr - 1 downto 0 );
              clk,
              nCLEAR:IN std_logic;
              Q:out std_logic
        );
    END COMPONENT;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal      D : dff_mux_in := ( others => ( others => '0' ) );
    signal      Q_sig : std_logic_vector( reg_size - 1 downto 0 );

    BEGIN
    FOR1: FOR i IN 0 TO reg_size-1 GENERATE

```

```

u1: muxdff
generic map( n_addr => 2 )
port map ( S => s,D => D( i ),clk => clk,nCLEAR => nCLR,Q => Q_sig( i ) );
D( i )( 3 )  <= x( i );

with i select D( i )( 2 ) <=
sl_in when 0,
Q_sig( i-1 ) when others;
D( i )( 1 )  <= sr_in when i=reg_size-1 else Q_sig( i+1 );
D( i )( 0 )  <= Q_sig( i );
END GENERATE;
Q      <= Q_sig;
end NDV;

```



```

-- *****
-- **** STUDENT: 64200112
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prozen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

    COMPONENT muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT (
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2*n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk,      -- signal      ure ( proen na sprednjo fronto )
            nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
                                Q=>'0' )
            Q : out std_logic -- izhod ULM strukture
        );
    END COMPONENT;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal D : dff_mux_in := ( others => ( others => '0' ) );
    signal Q_sig : std_logic_vector( reg_size - 1 downto 0 ) := ( others => '0' );

BEGIN

```

```

muxdff1 : for i in 0 to reg_size - 1 generate
    U0 : muxdff
        generic map( n_addr => 2 )
        port map( S => s,          D => D( i ),          clk => clk,          nCLEAR => nCLR,          Q =>
Q_sig( i )
                );

    D( i )( 3 ) <= x( i );

    with i select
        D( i )( 2 ) <=      sl_in when 0,          Q_sig( i - 1 ) when others;

    D( i )( 1 ) <= sr_in when i=reg_size - 1 else Q_sig( i + 1 );
    D( i )( 0 ) <= Q_sig( i );
end generate;

Q      <= Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64200163
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size : natural := 4 );
    port( clk,
          nCLR,          sr_in,          sl_in : in std_logic;
          s : in std_logic_vector( 1 downto 0 );
          x : in std_logic_vector( reg_size - 1 downto 0 );
          Q : out std_logic_vector( reg_size - 1 downto 0 )
    );
end shift_reg;

architecture NDV of shift_reg is
    component muxdff is
        generic( n_addr : natural := 4 );
        port( S : in std_logic_vector( n_addr - 1 downto 0 );
              D : in std_logic_vector( 2**n_addr - 1 downto 0 );
              clk, nCLEAR : in std_logic;
              Q : out std_logic
        );
    end component;
    type dff_mux_in is array( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal D : dff_mux_in := ( others => ( others => '0' ) );
    signal Q_sig : std_logic_vector( reg_size - 1 downto 0 );
begin
    shift_register: for i in 0 to reg_size - 1 generate
        u1: muxdff
            generic map( n_addr => 2 )
            port map(
                S => s,
                D => D( i ),
                clk => clk,
                nCLEAR => nCLR,
                Q => Q_sig( i
            );
    end generate;
end;

```

```

end generate;
D( 0 ) <= x( 0 ) & sl_in & Q_sig( 1 ) & Q_sig( 0 );
D( reg_size - 1 ) <= x( reg_size - 1 ) & Q_sig( reg_size - 2 ) & sr_in & Q_sig( reg_size - 1 );
muxdff_connect: for i in 1 to reg_size - 2 generate
    D( i )( 0 ) <= Q_sig( i );
    D( i )( 1 ) <= Q_sig( i+1 );
    D( i )( 2 ) <= Q_sig( i-1 );
    D( i )( 3 ) <= x( i );
end generate;
Q <= Q_sig;
end NDV;

```

```

-- *****
-- **** STUDENT: 64200238
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Operacije ne delujejo, ker je zamenjana polariteta nCLR signala.
--nCLR_neg <= not nCLR; -- sklepam da je to ta PRESET MIŠLJEN KAO POSTAVLJEN NA '1'
nCLR_neg <= nCLR; -- narobe ste sklepali
-- *****

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity shift_reg is
  generic(
    reg_size: natural := 4    -- velikost registra
  );
  PORT (
    clk : in std_logic;      -- signal      ure ( prožen na sprednjo fronto )
    nCLR : in std_logic;     -- signal      za asinhrono brisanje
    sr_in : in std_logic;    -- zaporedni vhod za pomikanje desno
    sl_in : in std_logic;    -- zaporedni vhod za pomikanje levo
    s : in std_logic_vector( 1 downto 0 ); -- izbira operacije
    x : in std_logic_vector( reg_size-1 downto 0 ); -- vhod za vzporedno nalaganje
    Q : out std_logic_vector( reg_size-1 downto 0 ) -- vzporedni izhod registra
  );
end shift_reg;

architecture NDV of shift_reg is

  COMPONENT muxdff is
    generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
    PORT (
      S : in std_logic_vector( n_addr - 1 downto 0 );
      D : in std_logic_vector( 2*n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
      clk, -- signal      ure ( prožen na sprednjo fronto )
      nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
      Q : out std_logic -- izhod ULM strukture
    );
  END COMPONENT;

```

```

        type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal
        D : dff_mux_in;
    signal
        Q_sig : std_logic_vector( reg_size-1 downto 0 );
    constant
        zeroes : std_logic_vector( reg_size-1 downto 0 ) := ( others => '0' );
    signal
        nCLR_neg : std_logic;    -- Declare a signal for the negated value
begin

    nCLR_neg    <= not nCLR; -- sklepam da je to ta PRESET MIŠLJEN KAO POSTAVLJEN NA '1'

    D( 0 ) <= x( 0 ) & sl_in & Q_sig( 1 ) & Q_sig( 0 );
    D( reg_size-1 )    <= x( reg_size - 1 ) & Q_sig( reg_size - 2 ) & sr_in & Q_sig( reg_size - 1 );

    jojojo: FOR i IN 1 TO reg_size - 2 GENERATE
        D( i ) <=
            x( i ) &    -- ( operacija 11 ) - nalaganje vhoda x
            Q_sig( i - 1 ) &    -- ( operacija 10 ) - pomik levo
            Q_sig( i + 1 ) &    -- ( operacija 01 ) - pomik desno
            Q_sig( i ); -- ( operacija 00 ) - držanje vsebine
    END GENERATE jojojo;

    kapitalizem: FOR i IN 0 TO reg_size - 1 GENERATE
        U0: muxdff
        generic map( n_addr => 2 )
        port map(
            S => S,
            D => D( i ),
            clk => clk,
            nCLEAR => nCLR_neg,
            Q => Q_sig( i )
        );
    END GENERATE kapitalizem;

    -- volja_do_ziolenja: process( jojmene,x,sl_in,Q_sig,sr_in )
    -- begin

    -- end process volja_do_ziolenja;

    Q    <= Q_sig;

```

```
end NDV;
```

```

-- *****
-- **** STUDENT: 64200288
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Operacije ne delujejo, ker je zamenjana polariteta nCLR signala.
--nCLR_neg <= not nCLR; -- sklepam da je to ta PRESET MIŠLJEN KAO POSTAVLJEN NA '1'
nCLR_neg <= nCLR; -- narobe ste sklepali
-- *****

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity shift_reg is
  generic(
    reg_size: natural := 4
  );
  PORT (
    clk : in std_logic;
    nCLR : in std_logic;
    sr_in : in std_logic;
    sl_in : in std_logic;
    s : in std_logic_vector( 1 downto 0 );
    x : in std_logic_vector( reg_size-1 downto 0 );
    Q : out std_logic_vector( reg_size-1 downto 0 )
  );
end shift_reg;

architecture NDV of shift_reg is

  COMPONENT muxdff is
    generic( n_addr: natural := 4 );
    PORT (
      S : in std_logic_vector( n_addr - 1 downto 0 );
      D : in std_logic_vector( 2*n_addr - 1 downto 0 );
      clk,
      nCLEAR : IN std_logic;
      Q : out std_logic
    );
  END COMPONENT;

```



```

type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
signal
    D : dff_mux_in := ( others => ( others => '0' ) );
signal
    Q_sig : std_logic_vector( reg_size-1 downto 0 );
constant
    nicle : std_logic_vector( reg_size-1 downto 0 ) := ( others => '0' );
signal
    CLR : std_logic;
begin

CLR    <= not nCLR;

D( 0 ) <= x( 0 ) & sl_in & Q_sig( 1 ) & Q_sig( 0 );
D( reg_size-1 )    <= x( reg_size - 1 ) & Q_sig( reg_size - 2 ) & sr_in & Q_sig( reg_size - 1 );

Zanka1: FOR i IN 1 TO reg_size - 2 GENERATE
    D( i ) <=
        x( i ) &
        Q_sig( i - 1 ) &
        Q_sig( i + 1 ) &
        Q_sig( i );
    END GENERATE Zanka1;

Zanka2: FOR i IN 0 TO reg_size - 1 GENERATE
    U0: muxdff
    generic map( n_addr => 2 )
    port map(
        S => S,
        D => D( i ),
        clk => clk,
        nCLEAR => CLR,
        Q => Q_sig( i )
    );
    END GENERATE Zanka2;
Q    <= Q_sig;
end NDV;

```

```

-- *****
-- **** STUDENT: 64200296
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Uporabljate stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let.
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prožen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;      -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is
    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal D : dff_mux_in := ( others => ( others => '0' ) );
    signal Q_sig : std_logic_vector( reg_size-1 downto 0 ) := ( others => '0' );
    COMPONENT muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT ( S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2*n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk,      -- signal      ure ( prožen na sprednjo fronto )
            nCLEAR : IN std_logic;      -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
            Q : out std_logic -- izhod ULM strukture
        );
    END COMPONENT;
BEGIN
    mux1: for i in 1 to reg_size-2 generate

```

```

        D( i ) <= x( i )&Q_sig( i-1 )&Q_sig( i+1 )&Q_sig( i );
    end generate;

D( 0 ) <=x( 0 )&    -- ( operacija 11 ) - nalaganje vhoda x( LSB )
    sl_in&        -- ( operacija 10 ) - pomik levo ( sl vhod gre v mesto LSB )
    Q_sig( 1 )& -- ( operacija 01 ) - pomik desno ( mesto 1 gre v mesto LSB )
    Q_sig( 0 ); -- ( operacija 00 ) - držanje vsebine LSB
mux2: for i in 0 to reg_size-1 generate
    U0: muxdff
        generic map( n_addr=>2 )
        port map(
            s=>s, D=>D( i ), clk=>clk, nCLEAR=>nCLR, Q=>Q_sig( i )
        );
    end generate;
D( reg_size-1 ) <=x( reg_size-1 )&Q_sig( reg_size-2 )&sr_in&Q_sig( reg_size-1 );
Q <=Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64200385
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk, -- signal           ure ( prozen na sprednjo fronto )
        nCLR, -- signal          za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in, -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic; -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

    component muxdff is
        generic( n_addr: natural := 2 );      -- stevilo naslovov ULM strukture
        PORT (
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2*n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk, -- signal           ure ( proÅžen na sprednjo fronto )
            nCLEAR : IN std_logic; -- signal          za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
            Q : out std_logic -- izhod ULM strukture
        );
    end component;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal D_vm : dff_mux_in;
    signal Q_sig : std_logic_vector( reg_size - 1 downto 0 );

BEGIN

```

```

stavek: for i in 0 to reg_size - 1 generate
U0: muxdff
    generic map ( n_addr => 2 )
    port map ( S => s, D => D_vm( i ),          clk => clk,          nCLEAR => nCLR,          Q => Q_sig( i ) );
end generate;

D_vm( 0 )    <=    x( 0 ) &
                  sl_in &
                  Q_sig( 1 ) &
                  Q_sig( 0 );

znk: for i in 1 to reg_size - 2 generate
    D_vm( i )    <=
        x( i ) &
        Q_sig( i-1 ) &
        Q_sig( i+1 ) &
        Q_sig( i );
    end generate;

    D_vm( reg_size - 1 )    <=
        x( reg_size - 1 ) &
        Q_sig( reg_size - 2 ) &
        sr_in &
        Q_sig( reg_size - 1 );

Q    <= Q_sig;
end NDV;

```

```

-- *****
-- **** STUDENT: 64210113
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Uporabljate stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let.
-- *****

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prožen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is
    COMPONENT muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT (
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2**n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk,      -- signal      ure ( prožen na sprednjo fronto )
            nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
            Q : out std_logic -- izhod ULM strukture
        );
    END COMPONENT;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal      D : dff_mux_in;
    signal      Qreg : std_logic_vector( reg_size - 1 downto 0 );

BEGIN

```

```

D( 0 )      <= x( 0 ) & sl_in & Qreg( 1 ) & Qreg( 0 );

D( reg_size - 1 ) <= x( reg_size - 1 ) & Qreg( reg_size - 2 ) & sr_in & Qreg( reg_size - 1 );

GEN_SHIFT_REG: for idx in 1 to reg_size - 2 generate
D( idx )      <= x( idx ) & Qreg( idx - 1 ) & Qreg( idx + 1 ) & Qreg( idx );
end generate GEN_SHIFT_REG;

GEN_MUXDFF: for idx in 0 to reg_size - 1 generate

ULM_MODULE: muxdff
generic map ( n_addr => 2 )
port map (
S => s,
D => D( idx ),
clk => clk,
nCLEAR => nCLR,
Q => Qreg( idx )
);

end generate GEN_MUXDFF;

Q      <= Qreg;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210132
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,    -- signal      ure ( prožen na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,  -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal
        D : dff_mux_in := ( others => ( others => '0' ) );
        Q_sig : std_logic_vector( reg_size - 1 downto 0 );

    component muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT ( S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2*n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk,    -- signal      ure ( prožen na sprednjo fronto )
            nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
            Q : out std_logic ); -- izhod ULM strukture
    END component;

BEGIN

```



```

D( 0 ) <= x( 0 ) & -- ( operacija 11 ) - nalaganje vhoda x( LSB )
                sl_in & -- ( operacija 10 ) - pomik levo ( sl vhod gre v mesto LSB )
                Q_sig( 1 ) & -- ( operacija 01 ) - pomik desno ( mesto 1 gre v mesto LSB )
                Q_sig( 0 ); -- ( operacija 00 ) - držanje vsebine LSB

D( reg_size-1 ) <= x( reg_size-1 ) & -- ( operacija 11 ) - nalaganje vhoda x( MSB )
                Q_sig( reg_size-2 ) & -- ( operacija 10 ) - pomik levo ( mesto reg_size-2 gre v mesto MSB )
                sr_in & -- ( operacija 01 ) - pomik desno ( sr vhod gre v mesto MSB )
                Q_sig( reg_size-1 ); -- ( operacija 00 ) - držanje vsebine MSB

U0: muxdff generic map ( n_addr => 2 ) port map ( S, D( 0 ), clk, nCLR, Q_sig( 0 ) );

U1: muxdff generic map ( n_addr => 2 ) port map ( S, D( reg_size-1 ), clk, nCLR, Q_sig( reg_size-1 ) );

G0: for i in 1 to reg_size-2 generate

    U2: muxdff generic map ( n_addr => 2 ) port map ( S, D( i ), clk, nCLR, Q_sig( i ) );
    D( i )( 0 ) <= Q_sig( i );
    D( i )( 1 ) <= Q_sig( i+1 );
    D( i )( 2 ) <= Q_sig( i-1 );
    D( i )( 3 ) <= x( i );

end generate;

Q <= Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210290
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,    -- signal           ure ( prozen na sprednjo fronto )
        nCLR,   -- signal           za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,  -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;           -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

    component muxdff is
        generic(
            n_addr: natural
        );
        port(
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2**n_addr - 1 downto 0 );
            clk, nCLEAR : IN std_logic;
            Q : out std_logic
        );
    end component;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal D_s : dff_mux_in := ( others => ( others => '0' ) );
    signal Q_s : std_logic_vector( reg_size-1 downto 0 );

```

BEGIN

```
    field : for idreg in 0 to reg_size-1 generate
        dff: muxdff
            generic map(
                n_addr => 2
            )
            port map(
                S => s,                D => D_s( idreg ),        clk => clk,        nCLEAR => nCLR,        Q =>
Q_s( idreg )
            );
        end generate field;

    connections : for idc in 1 to reg_size-2 generate
        D_s( idc )    <= x( idc )&Q_s( idc-1 )&Q_s( idc+1 )&Q_s( idc );
    end generate connections;

    D_s( 0 )        <= x( 0 )&sl_in&Q_s( 1 )&Q_s( 0 );
    D_s( reg_size-1 ) <= x( reg_size-1 )&Q_s( reg_size-2 )&sr_in&Q_s( reg_size-1 );

    Q    <= Q_s;

END NDV;
```

```

-- *****
-- **** STUDENT: 64210382
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Uporabljate stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let.
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,    -- signal      ure ( prožen na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,  -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is
    COMPONENT muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT (
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2**n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk,    -- signal      ure ( prožen na sprednjo fronto )
            nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
            Q : out std_logic -- izhod ULM strukture
        );
    END COMPONENT;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal      D : dff_mux_in;
    signal      Q_sig : std_logic_vector( reg_size - 1 downto 0 );

BEGIN

```

```

FOR1: FOR i IN 0 TO reg_size-1 GENERATE
    u1: muxdff
    generic map( n_addr => 2 )
    port map (
        S => s,          D => D( i ),          clk => clk,          nCLEAR => nCLR,          Q => Q_sig( i
    );

    D( i )( 3 ) <= x( i );

    with i select D( i )( 2 ) <=
        sl_in when 0,          Q_sig( i-1 ) when others;

    D( i )( 1 ) <= sr_in when i=reg_size-1 else Q_sig( i+1 );
    D( i )( 0 ) <= Q_sig( i );
END GENERATE;

Q    <= Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210384
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );    -- velikost registra
    PORT ( clk,    -- signal            ure ( prožen na sprednjo fronto )
          nCLR,    -- signal            za asinhrono brisanje ( vsebina registra gre na 0 )
          sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
          sl_in : IN std_logic;    -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
          s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
          x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
          Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
        );
end shift_reg;

architecture NDV of shift_reg is

    COMPONENT muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT ( S : in std_logic_vector( n_addr - 1 downto 0 );
              D : in std_logic_vector( 2*n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
              clk,    -- signal            ure ( prožen na sprednjo fronto )
              nCLEAR : IN std_logic;    -- signal            za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
              Q : out std_logic    -- izhod ULM strukture
            );
    END COMPONENT;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal D : dff_mux_in := ( others => ( others => '0' ) );
    signal Q_sig : std_logic_vector( reg_size - 1 downto 0 );

BEGIN

```

```

FOR1: FOR i IN 0 TO reg_size-1 GENERATE
    u1: muxdff
    generic map( n_addr => 2 )
    port map (
        S => s,          D => D( i ),          clk => clk,          nCLEAR => nCLR,          Q => Q_sig( i
    );

    D( i )( 3 ) <= x( i );

-- To dela:
    with i select D( i )( 2 ) <= sl_in when 0,          Q_sig( i-1 ) when others;

-- To crasha simulacijo:
-- D( i )( 2 ) <= sl_in when i=0 else Q_sig( i-1 );

    D( i )( 1 ) <= sr_in when i=reg_size-1 else Q_sig( i+1 );
    D( i )( 0 ) <= Q_sig( i );
END GENERATE;

Q <= Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210386
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Uporabljate stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let.
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prozen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;      -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

component muxdff IS
    generic( n_addr: natural :=4 ); -- stevilo naslovov ULM strukture
    PORT (
        S : in std_logic_vector( n_addr - 1 downto 0 );
        D : in      std_logic_vector( 2**n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
        clk,      -- signal      ure ( proen na sprednjo fronto )
        nCLEAR : IN std_logic;      -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )

        Q : out std_logic      -- izhod ULM strukture
    );
END component;

type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
signal      D : dff_mux_in := ( others => ( others => '0' ) );

```



```

signal      Q_sig : std_logic_vector( reg_size - 1 downto 0 );

BEGIN

G0: for i in 0 to reg_size - 1 generate

    D( 0 ) <= x( 0 ) &  -- ( operacija 11 ) - nalaganje vhoda x( LSB )
                    sl_in &  -- ( operacija 10 ) - pomik levo ( sl vhod gre v mesto LSB )
                    Q_sig( 1 ) &  -- ( operacija 01 ) - pomik desno ( mesto 1 gre v mesto LSB )
                    Q_sig( 0 );  -- ( operacija 00 ) - drÅžanje vsebine LSB

    U0: muxdff generic map ( n_addr => 2 ) port map ( S, D( i ), clk, nCLR, Q_sig( i ) );

    D( i )( 3 ) <= x( i );
    with i select D( i )( 2 ) <=
        sl_in when 0,          Q_sig( i-1 ) when others;

    D( i )( 1 ) <= sr_in when i=reg_size-1 else Q_sig( i+1 );
    D( i )( 0 ) <= Q_sig( i );

end generate;

Q      <= Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210445
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prožen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

    COMPONENT muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT (
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2*n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk,      -- signal      ure ( prožen na sprednjo fronto )
            nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
            Q : out std_logic -- izhod ULM strukture
        );
    END COMPONENT;

    signal Q_sig : std_logic_vector( reg_size - 1 downto 0 );

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal D : dff_mux_in := ( others => ( others => '0' ) );
BEGIN

```

```

FOR1: FOR i IN 0 TO reg_size-1 GENERATE
    u0: muxdff
    generic map( n_addr => 2 )
    port map (
        S => s,          D => D( i ),          clk => clk,          nCLEAR => nCLR,          Q => Q_sig( i
    );

    D( i )( 3 ) <= x( i );

    with i select D( i )( 2 ) <= sl_in when 0, Q_sig( i-1 ) when others;

    D( i )( 1 ) <= sr_in when i=reg_size-1 else Q_sig( i+1 );
    D( i )( 0 ) <= Q_sig( i );
END GENERATE;

Q    <= Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210455
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Uporabljate stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let.
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prozen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is
    COMPONENT muxdff IS
        generic( n_addr: natural := 4 );
        PORT (
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2**n_addr - 1 downto 0 );
            clk,      nCLEAR : IN std_logic;
            Q : out std_logic
        );
    END COMPONENT;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal      D : dff_mux_in;
    signal      Q_sig : std_logic_vector( reg_size - 1 downto 0 );

BEGIN
    FOR1: FOR i IN 0 TO reg_size-1 GENERATE
        u1: muxdff

```

```

generic map( n_addr => 2 )
port map (
    S => s,          D => D( i ),      clk => clk,      nCLEAR => nCLR,      Q => Q_sig( i
);

D( i )( 3 ) <= x( i );

with i select D( i )( 2 ) <=
    sl_in when 0,      Q_sig( i-1 ) when others;

D( i )( 1 ) <= sr_in when i=reg_size-1 else Q_sig( i+1 );
D( i )( 0 ) <= Q_sig( i );
END GENERATE;

Q      <= Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210457
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Uporabljate stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let.
-- *****

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prožen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is
    COMPONENT muxdff IS
        generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
        PORT (
            S : in std_logic_vector( n_addr - 1 downto 0 );
            D : in std_logic_vector( 2**n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
            clk,      -- signal      ure ( prožen na sprednjo fronto )
            nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
            Q : out std_logic -- izhod ULM strukture
        );
    END COMPONENT;

    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal      D : dff_mux_in;
    signal      Q_sig : std_logic_vector( reg_size - 1 downto 0 );

BEGIN

```

```

FOR1: FOR i IN 0 TO reg_size-1 GENERATE
    u1: muxdff
        generic map( n_addr => 2 )
        port map (
            S => s,          D => D( i ),          clk => clk,          nCLEAR => nCLR,          Q => Q_sig( i
        );

        D( i )( 3 ) <= x( i );

-- To dela:
    with i select D( i )( 2 ) <=
        sl_in when 0,          Q_sig( i-1 ) when others;

-- To crasha simulacijo:
-- D( i )( 2 ) <= sl_in when i=0 else Q_sig( i-1 );

        D( i )( 1 ) <= sr_in when i=reg_size-1 else Q_sig( i+1 );
        D( i )( 0 ) <= Q_sig( i );
    END GENERATE;

    Q <= Q_sig;

end NDV;

```

```

-- *****
-- **** STUDENT: 64240430
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,      -- signal      ure ( prožen na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,    -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture NDV of shift_reg is

component muxdff
    generic( n_addr: natural := 4 ); -- stevilo naslovov ULM strukture
    PORT (
        S : in std_logic_vector( n_addr - 1 downto 0 );
        D : in std_logic_vector( 2*n_addr - 1 downto 0 ); -- vektor vhodov ULM strukture
        clk,      -- signal      ure ( prožen na sprednjo fronto )
        nCLEAR : IN std_logic; -- signal      za asinhrono brisanje ( nCLEAR = '0' se postavi
Q=>'0' )
        Q : out std_logic -- izhod ULM strukture
    );
end component;

type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
signal      D_sig : dff_mux_in;
-- pomocna varijabla Q
signal      Q_sig : std_logic_vector( reg_size - 1 downto 0 );

```



```
BEGIN
```

```
-- Povezovanje komponente ULM modula s podano entiteto pomikalnega registra z uporabo povezovalnega ( PORT MAP )  
stavka
```

```
povezovanje:
```

```
for i in 0 to reg_size - 1 generate
```

```
    U0: muxdff
```

```
        generic map ( n_addr => 2 )
```

```
        port map ( S => s,          D => D_sig( i ),          clk => clk,          nCLEAR => nCLR,
```

```
        Q => Q_sig( i ) );
```

```
-- LSB mesto pomikalnega registra
```

```
    LSB:
```

```
        if i = 0 generate
```

```
            D_sig( i )    <= x( i ) & sl_in & Q_sig( i+1 ) & Q_sig( i );
```

```
        end generate;
```

```
-- MSB mesto pomikalnega registra
```

```
    MSB:
```

```
        if i = reg_size - 1 generate
```

```
            D_sig( i )    <= x( i ) & Q_sig( i-1 ) & sr_in & Q_sig( i );
```

```
        end generate;
```

```
-- Ostanek
```

```
    ostanek:
```

```
        if i /= 0 and i /= reg_size - 1 generate
```

```
            D_sig( i )    <= x( i ) & Q_sig( i-1 ) & Q_sig( i+1 ) & Q_sig( i );
```

```
        end generate;
```

```
end generate;
```

```
Q    <= Q_sig;
```

```
end NDV;
```

```

-- *****
-- **** PREDLOGA VAJE
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift_reg is
    generic( reg_size: natural := 4 );      -- velikost registra
    PORT (
        clk,    -- signal      ure ( prozen na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina registra gre na 0 )
        sr_in,  -- zaporedni vhod za pomikanje desno ( takrat gre sr_in->MSB )
        sl_in : IN std_logic;              -- zaporedni vhod pri pomikanju levo ( takrat gre sl_in->LSB )
        s : in std_logic_vector( 1 downto 0 ); -- izbira operacije pomikalnega registra
        x : in std_logic_vector( reg_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( reg_size - 1 downto 0 ) -- vzporedni izhod registra
    );
end shift_reg;

architecture ideal of shift_reg is
    type dff_mux_in is array ( reg_size - 1 downto 0 ) of std_logic_vector( 3 downto 0 );
    signal
        D : dff_mux_in := ( others => ( others => '0' ) );

    signal
        Q_sig : std_logic_vector( reg_size - 1 downto 0 );

    COMPONENT muxdff IS
        generic( n_addr: natural := 2 );
        PORT (
            S : in std_logic_vector( n_addr-1 downto 0 );
            D : in std_logic_vector( 2*n_addr-1 downto 0 );
            clk, nCLEAR : IN std_logic;
            Q : out std_logic
        );
    END COMPONENT;

begin
    --      Postavitev mest v pomikalnem registru je:
    --      ( MSB mesto je fizično skrajno desno, LSB mesto je fizično skrajno levo )

```

```

-- S1 S0
-- 1 1 : Vzporedno nalaganje x => Q
-- 1 0 : Pomikanje levo ( v smeri od LSB do MSB )
-- 0 1 : Pomikanje desno ( v smeri od MSB do LSB )
-- 0 0 : Držanje vsebine

D( 0 ) <= x( 0 ) & -- ( operacija 11 ) - nalaganje vhoda x( LSB )
          sl_in & -- ( operacija 10 ) - pomik levo ( sl vhod gre v mesto LSB )
          Q_sig( 1 ) & -- ( operacija 01 ) - pomik desno ( mesto 1 gre v mesto LSB )
          Q_sig( 0 ); -- ( operacija 00 ) - držanje vsebine LSB

L0: FOR i IN 1 TO reg_size - 2 GENERATE
    D( i ) <=
        x( i ) & -- ( operacija 11 ) - nalaganje vhoda x
        Q_sig( i - 1 ) & -- ( operacija 10 ) - pomik levo
        Q_sig( i + 1 ) & -- ( operacija 01 ) - pomik desno
        Q_sig( i ); -- ( operacija 00 ) - držanje vsebine
    END GENERATE;

D( reg_size - 1 ) <=
    x( reg_size - 1 ) & -- ( operacija 11 ) - nalaganje vhoda x
    Q_sig( reg_size - 2 ) & -- ( operacija 10 ) - pomik levo ( sl vhod gre v mesto 0 )
    sr_in & -- ( operacija 01 ) - pomik desno ( sr vhod gre v mesto MSB )
    Q_sig( reg_size - 1 ); -- ( operacija 00 ) - držanje vsebine

L1: FOR i IN 0 TO reg_size - 1 GENERATE
    U0: muxdff generic map ( n_addr => 2 )
        port map ( S, D( i ), clk, nCLR, Q_sig( i ) );
    END GENERATE;

Q <= Q_sig;

PROCESS( D ) -- process for logging the value of shift register
    function array_of_slv_to_string( D : dff_mux_in ) return string is
        use Std.TextIO.all;
        variable bv: bit_vector( D( D'left )'range ) := to_bitvector( D( D'left ) );
        variable lp: line;
        begin
            for i in D'RANGE loop

```

```
        bv := to_bitvector( D( i ) );
        write( lp, bv );
        write( lp, HT );
    end loop;
    return lp.all;
end;

BEGIN
    report array_of_slv_to_string( D );    -- write internal shift register contents
END PROCESS;

end ideal;
```

