

-- **** 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: Ko se napolni, se na izhodu FIFO pojavlja 0, ne izhodne vrednosti.	
13	
Koda FIFO je sicer pravilna – ne deluje zaradi napake pri programiranju pomikalnega registra. ....	13
-- **** STUDENT: 64200288.....	15
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ko se napolni, se na izhodu FIFO pojavlja 0, ne izhodne vrednosti.	
15	
Koda FIFO je sicer pravilna – ne deluje zaradi napake pri programiranju pomikalnega registra. ....	15
-- ***** .....	15
-- **** STUDENT: 64200296.....	17
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Uporabljate stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let.	17
-- ***** .....	17
-- **** STUDENT: 64200385.....	19
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Napake sintetizatorja: .....	19
ERROR:HDLParers:3312 - "64200385/fifo.vhd" Line 67. Undefined symbol 'g'.....	19
ERROR:HDLParers:1209 - "64200385/fifo.vhd" Line 67. g: Undefined symbol (last report in this block) .....	19
ERROR:HDLParers:164 - "64200385/fifo.vhd" Line 67. parse error, unexpected END, expecting OPENPAR or TICK or LSQBRACK.....	19
WARNING:HDLParers:523 - "64200385/fifo.vhd" Line 73. The function array_of_slv_to_string does not contain any return statement.....	19
ERROR:HDLParers:3011 - "64200385/fifo.vhd" Line 73. End Identifier NDV does not match declaration, array_of_slv_to_string. ....	19
ERROR:HDLParers:3524 - "64200385/fifo.vhd" Line 73. Unexpected end of line.....	19

ERROR:HDLParasers:834 - "64210455/fifo.vhd" Line 60. Signal read_pointer has a multi source. ....	19
Izgleda se vam je v proces za izpis vrednosti FIFO prikradla nekaj odvečna črka g.☺ .....	19
Če zadevo “popravim”, FIFO deluje pravilno. ....	19
-- **** 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: Uporablajte stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let. 28	
-- ***** .....	28
-- **** STUDENT: 64210384.....	30
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	30
-- **** STUDENT: 64210386.....	32
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Uporablajte stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let. 32	
-- ***** .....	32
-- **** STUDENT: 64210445.....	34
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Uporablajte stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let. 34	
-- ***** .....	34
-- **** STUDENT: 64210455.....	36
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Rešitev je popolnoma originalna (pohvalno), sledi pa bolj navodilom prosojnic predavanj, kot navodilom domače naloge. Žal ne dobim rezultatov simulacije – treba je. spremeniti več stvari. Pri komentarju naloge (spodaj) si oglejte kodo, ki povzema vašo idejo, le da dela točno po navodilih naloge. ....	36
-- **** STUDENT: 64210457.....	39
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Uporablajte stare Synopsis knjižnice (STD_LOGIC_ARITH), česar v predlogah že ni kar nekaj let. 39	
-- **** STUDENT: 64240430.....	41
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	41
-- **** PREDLOGA VAJE .....	43
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	43

```

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

entity fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    COMPONENT shift_reg
        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( fifo_size-1 downto 0 );
        Q : OUT std_logic_vector( fifo_size-1 downto 0 );
    );
    END COMPONENT;

    constant zeros : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );

    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );

```

```

signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );

signal      reg_s : std_logic_vector( 1 downto 0 );

begin

    reg_s <= "10" when nEnable = '0' and LOAD = '1' else "00";

    sr: for i in 0 to fifo_width-1 generate
        sr1: shift_reg
            generic map ( reg_size => fifo_size )
            PORT MAP (
                clk => clk,          nCLR => nCLR,          sr_in => '0',          sl_in => fifo_in( i ),
                s => reg_s,          x => zeros,          Q => Q( i )
            );

        fifo_out( i )<= Q( i )( fifo_size-1 );
    end generate;

    PROCESS( Q ) -- process for Logging the value of FIFO
    function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
        use Std.TextIO.all;
        variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
        variable lp: line;
        begin
            for i in Q'RANGE loop      -- scroll the array of std_logic_vectors
                bv := to_bitvector( Q( i ) );    -- convert to printable value
                write( lp, bv );    -- append dynamic string
                write( lp, HT );    -- insert horizontal tab
            end loop;
            return lp.all;    -- return the concatenated string
        end;

    BEGIN
        report array_of_slv_to_string( Q );    -- write internal FIFO contents
    END PROCESS;

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 fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
    signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );

    component shift_reg
        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 component;

    signal      s : std_logic_vector( 1 downto 0 ) := ( others => '0' );

```

```

constant    zeroes : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );

begin

    s      <= "10" when ( ( nEnable = '0' ) and ( LOAD = '1' ) ) else "00";

    zanka_f_g: for i in 0 to fifo_width - 1 generate
        u0: shift_reg
            generic map ( reg_size => fifo_size )
            port map (   clk => clk,          nCLR => nCLR,          sr_in => '0',          sl_in =>
fifo_in( i ),          s => s,          x => zeroes,          Q => Q( i ) );

            fifo_out( i )<= Q( i )( fifo_size-1 );
    end generate;

    PROCESS( Q ) -- process for logging the value of FIFO
        function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
            use Std.TextIO.all;
            variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
            variable lp: line;
            begin
                for i in Q'RANGE loop          -- scroll the array of std_logic_vectors
                    bv := to_bitvector( Q( i ) );    -- convert to printable value
                    write( lp, bv );    -- append dynamic string
                    write( lp, HT );
                end loop;
                return lp.all;          -- insert horizontal tab -- return the concatenated string
            end;
        BEGIN
            report array_of_slv_to_string( Q );    -- write internal FIFO contents
        END PROCESS;

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 fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    COMPONENT shift_reg
    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( fifo_size-1 downto 0 );
        Q : OUT std_logic_vector( fifo_size-1 downto 0 ) );
    END COMPONENT;

    constant    nic : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );
    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
    signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );
    signal      reg_s : std_logic_vector( 1 downto 0 );

```

```

begin

reg_s <= "10" when nEnable = '0' and LOAD = '1' else "00";

sr: for i in 0 to fifo_width-1 generate
sr1: shift_reg
generic map ( reg_size => fifo_size )
PORT MAP (
clk => clk,
nCLR => nCLR,
sr_in => '0',
sl_in => fifo_in( i ),
s => reg_s,
x => nic,
Q => Q( i ) );

fifo_out( i )<= Q( i )( fifo_size-1 );
end generate;

PROCESS( Q )
function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
use Std.TextIO.all;
variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
variable lp: line;
begin
for i in Q'RANGE loop
bv := to_bitvector( Q( i ) );
write( lp, bv );
write( lp, HT );
end loop;
return lp.all;
end;
BEGIN
report array_of_slv_to_string( Q );
END PROCESS;
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 fifo is
    generic(
        fifo_width: natural := 4; -- dolina vhodnega podatka
        fifo_size: natural := 8 ); -- tevilno hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogoanje FIFO ( '0'-> omogoen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal      za omogoanje nalaganja ( '1'-> fifo_in se vpie )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    component 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 component;

    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
    signal Q : shift_reg_array_type := ( others => ( others => '0' ) );

    signal s_sig : std_logic_vector( 1 downto 0 ) := ( others => '0' );

```

```

    signal      zero_sig : std_logic_vector( fifo_size - 1 downto 0 ) := ( others => '0' );

begin

    s_sig <= "10" when ( nENABLE='0' and LOAD='1' ) else "00";

    shift_reg_gen : for i in 0 to fifo_width - 1 generate
        shift_reg_comp : shift_reg
            generic map( reg_size => fifo_size )
            port map( clk => clk,          nCLR => nCLR,          sr_in => '0',          sl_in =>
fifo_in( i ),          s => s_sig,          x => zero_sig,          Q => Q( i )
            );

        fifo_out( i )<= Q( i )( fifo_size - 1 );
    end generate;

    PROCESS( Q ) -- process for logging the value of FIFO
        function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
            use Std.TextIO.all;
            variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
            variable lp: line;
        begin
            for i in Q'RANGE loop          -- scroll the array of std_logic_vectors
                bv := to_bitvector( Q( i ) );    -- convert to printable value
                write( lp, bv );    -- append dynamic string
                write( lp, HT );    -- insert horizontal tab
            end loop;
            return lp.all;    -- return the concatenated string
        end;
    BEGIN
        report array_of_slv_to_string( Q );    -- write internal FIFO contents
    END PROCESS;

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 fifo is
    generic( fifo_width : natural := 4;
              fifo_size : natural := 8 );
    port( clk,      nCLR,      nEnable,      LOAD : in std_logic;
          fifo_in : in std_logic_vector( fifo_width - 1 downto 0 );
          fifo_out : out std_logic_vector( fifo_width - 1 downto 0 )
        );
end fifo;

architecture NDV of fifo is
    component 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( fifo_size - 1 downto 0 );
              Q : out std_logic_vector( fifo_size - 1 downto 0 )
            );
    end component;
    type shift_reg_array_type is array( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
    signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );
    signal      s : std_logic_vector( 1 downto 0 );
    constant    zeros : std_logic_vector( fifo_size - 1 downto 0 ) := ( others => '0' );
begin
    s      <= "10" when nEnable = '0' and LOAD = '1' else "00";
    fifo_assemble: for i in 0 to fifo_width - 1 generate
        u1: shift_reg
            generic map( reg_size => fifo_size )
            port map(

```

```

        clk => clk,          nCLR => nCLR,          sr_in => '0',          sl_in => fifo_in( i ),
s => s,          x => zeros,          Q => Q( i )
    );
    fifo_out( i )<= Q( i )( fifo_size - 1 );
end generate;
process( Q )
    function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
        use Std.TextIO.all;
        variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
        variable lp: line;
        begin
            for i in Q'RANGE loop
                bv := to_bitvector( Q( i ) );
                write( lp, bv );
                write( lp, HT );
            end loop;
            return lp.all;
        end;
    begin
        report array_of_slv_to_string( Q );
    end process;
end NDV;

```

```

-- *****
-- **** STUDENT: 64200238
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ko se napolni, se na izhodu FIFO pojavlja 0, ne izhodne vrednosti.
Koda FIFO je sicer pravilna – ne deluje zaradi napake pri programiranju pomikalnega registra.
-- *****

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

entity fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

component 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 component;

```

```

type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );
signal      zeroes : std_logic_vector( fifo_size-1 downto 0 ):= ( others => '0' );
signal      s : std_logic_vector( 1 downto 0 ):= "00";

begin

s      <= "10" when nEnable = '0' and LOAD = '1' else "00";

fific: for i in 0 to fifo_width-1 generate
  U2: shift_reg generic map ( reg_size => fifo_size )
    port map(
      clk => clk,
      nCLR => nCLR,
      sr_in => '0',
      sl_in => fifo_in( i ), s => s, x => zeroes, Q => Q( i )
    );

      fifo_out( i )<= Q( i )( fifo_size - 1 );
end generate fific;

PROCESS( Q ) -- process for logging the value of FIFO
function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
use Std.TextIO.all;
variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
variable lp: line;
begin
for i in Q'RANGE loop      -- scroll the array of std_logic_vectors
  bv := to_bitvector( Q( i ) ); -- convert to printable value
write( lp, bv );      -- append dynamic string
write( lp, HT );      -- insert horizontal tab
end loop;
return lp.all;      -- return the concatenated string
end;
BEGIN
report array_of_slv_to_string( Q );      -- write internal FIFO contents
END PROCESS;

end NDV;

```

```

-- *****
-- **** STUDENT: 64200288
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ko se napolni, se na izhodu FIFO pojavlja 0, ne izhodne vrednosti.
Koda FIFO je sicer pravilna – ne deluje zaradi napake pri programiranju pomikalnega registra.
-- *****

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

entity fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
    )

        LOAD : IN std_logic;      -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek

    );
end fifo;

architecture NDV of fifo is

component shift_reg is
    generic(
        reg_size: natural := 4    -- velikost registra
    );
    PORT (
        clk : in std_logic;      -- signal      ure ( prošen 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 component;

```

```

type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );
signal      nicle : std_logic_vector( fifo_size-1 downto 0 ):= ( others => '0' );
signal      s : std_logic_vector( 1 downto 0 );

begin

s      <= "10" when nEnable = '0' and LOAD = '1' else "00";

MyFifo: for i in 0 to fifo_width-1 generate
  U2: shift_reg_generic map ( reg_size => fifo_size )
    port map(
      clk => clk, nCLR => nCLR, sr_in => '0', sl_in => fifo_in( i ), s => s, x => nicle, Q => Q( i )
    );
  fifo_out( i )<= Q( i )( fifo_size - 1 );
end generate;

PROCESS( Q )
function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
use Std.TextIO.all;
variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
variable lp: line;
begin
  for i in Q'RANGE loop
    bv := to_bitvector( Q( i ) );
    write( lp, bv );
    write( lp, HT );
  end loop;
  return lp.all;
end;
BEGIN
report array_of_slv_to_string( Q );
END PROCESS;
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 fifo is
  generic(
    fifo_width: natural := 4; -- dolžina vhodnega podatka
    fifo_size: natural := 8 ); -- število hranjenih podatkov
  PORT (
    clk,    -- signal      ure ( spremembe na sprednjo fronto )
    nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
    nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
    LOAD : IN std_logic;    -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
    fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
    fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
  );
end fifo;

architecture NDV of fifo is
  component 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 component;
  type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
  signal Q : shift_reg_array_type := ( others => ( others => '0' ) );
  signal reg : std_logic_vector( 1 downto 0 ) := ( others => '0' );
  constant zeros : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );

```

```
begin
```

```
    reg    <="10" when ( nEnable = '0' and LOAD = '1' ) else "00";
```

```
    fifo: for i in 0 to fifo_width-1 generate
```

```
        shftreg: shift_reg
```

```
            generic map ( reg_size => fifo_size )
```

```
            PORT MAP (
```

```
                clk => clk,          nCLR => nCLR,          sr_in => '0',
```

```
                s => reg,            x => zeros,            Q => Q( i )
```

```
            );
```

```
            fifo_out( i )<= Q( i )( fifo_size-1 );
```

```
        end generate;
```

```
    PROCESS( Q ) -- process for logging the value of FIFO
```

```
    function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
```

```
    use Std.TextIO.all;
```

```
    variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
```

```
    variable lp: line;
```

```
    begin
```

```
    for i in Q'RANGE loop      -- scroll the array of std_logic_vectors
```

```
        bv := to_bitvector( Q( i ) );    -- convert to printable value
```

```
        write( lp, bv );    -- append dynamic string
```

```
        write( lp, HT );    -- insert horizontal tab
```

```
    end loop;
```

```
    return lp.all;    -- return the concatenated string
```

```
    end;
```

```
    BEGIN
```

```
        report array_of_slv_to_string( Q );    -- write internal FIFO contents
```

```
    END PROCESS;
```

```
end NDV;
```

```

-- *****
-- **** STUDENT: 64200385
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Napake sintetizatorja:
ERROR:HDLParasers:3312 - "64200385/fifo.vhd" Line 67. Undefined symbol 'g'.
ERROR:HDLParasers:1209 - "64200385/fifo.vhd" Line 67. g: Undefined symbol (last report in this block)
ERROR:HDLParasers:164 - "64200385/fifo.vhd" Line 67. parse error, unexpected END, expecting OPENPAR or TICK or LSQBRACK
WARNING:HDLParasers:523 - "64200385/fifo.vhd" Line 73. The function array_of_slv_to_string does not contain any return
statement.
ERROR:HDLParasers:3011 - "64200385/fifo.vhd" Line 73. End Identifier NDV does not match declaration,
array_of_slv_to_string.
ERROR:HDLParasers:3524 - "64200385/fifo.vhd" Line 73. Unexpected end of line.
ERROR:HDLParasers:834 - "64210455/fifo.vhd" Line 60. Signal read_pointer has a multi source.
Izgleda se vam je v proces za izpis vrednosti FIFO prikradla nekaj odvečna črka g.☺
Če zadevo "popravim", FIFO deluje pravilno.
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fifo is
  generic(
    fifo_width: natural := 4; -- dolina vhodnega podatka
    fifo_size: natural := 8 ); -- tevilov hranjenih podatkov
  PORT (
    clk, -- signal ure (spremembe na sprednjo fronto)
    nCLR, -- signal za asinhrono brisanje (vsebinsa FIFO gre na 0)
    nEnable, -- signal za omogoanje FIFO ('0' -> omogoen vpis, '1' -> ohranja stanje)
    LOAD : IN std_logic; -- signal za omogoanje nalaganja ('1' -> fifo_in se vpie)
    fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
    fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
  );
end fifo;

architecture NDV of fifo is

  component shift_reg
    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( fifo_size - 1 downto 0 ); -- vhod za vzporedno nalaganje
        Q : out std_logic_vector( fifo_size - 1 downto 0 ) -- vzporedni izhod registra
    );

end component;

signal          ff_s : std_logic_vector( 1 downto 0 );
constant        zeros: std_logic_vector( fifo_size - 1 downto 0 ) := ( others=>'0' );
type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
signal          Q : shift_reg_array_type;

begin

    ff_s <= "10" when nEnable = '0' and LOAD = '1' else "00";

    ff: for i in 0 to fifo_width - 1 generate
    U1: shift_reg
        generic map ( reg_size => fifo_size )
        port map ( clk => clk, nCLR => nCLR, sr_in => '0', sl_in => fifo_in( i ), s =>
ff_s, x => zeros, Q => Q( i ) );

        fifo_out( i ) <= Q( i )( fifo_size - 1 );
    end generate;

    PROCESS( Q )
    function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
    use Std.TextIO.all;
    variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
    variable lp: line;
    begin
        for i in Q'RANGE loop
            bv := to_bitvector( Q( i ) );
            write( lp, bv );
            write( lp, HT );
        end loop;
        return lp.all; g
    end;
    begin
    report array_of_slv_to_string( Q );

```

```
END PROCESS;
```

```
end NDV;
```

```

-- *****
-- **** STUDENT: 64210113
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    COMPONENT shift_reg
        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( fifo_size-1 downto 0 );
        Q : OUT std_logic_vector( fifo_size-1 downto 0 );
    );
    END COMPONENT;

    constant zeros : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );
    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
    signal Q : shift_reg_array_type := ( others => ( others => '0' ) );

```

```

        signal      nReg : std_logic_vector( 1 downto 0 );

begin

    nReg <= "10" when ( nEnable = '0' and LOAD = '1' ) else "00";

    shift_registers: for i in 0 to fifo_width - 1 generate
    shift_reg_inst : shift_reg
    generic map ( reg_size => fifo_size )
    port map (
    clk => clk,
    nCLR => nCLR,
    sr_in => '0',
    sl_in => fifo_in( i ),
    s => nReg,
    x => zeros,
    Q => Q( i )
    );

    fifo_out( i )      <= Q( i )( fifo_size - 1 );
    end generate shift_registers;

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 fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    component shift_reg
        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( fifo_size-1 downto 0 );
        Q : OUT std_logic_vector( fifo_size-1 downto 0 )
    );
    end component;

    constant zeroes : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );

    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );

```



```

signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );

signal      s : std_logic_vector( 1 downto 0 );

begin

    s( 1 ) <= ( not nEnable ) and LOAD;
    s( 0 ) <= '0';

    gen0: for i in 0 to fifo_width-1 generate
        shift_reg0: shift_reg
            generic map ( reg_size => fifo_size )
            port map ( clk, nCLR, '0', fifo_in( i ), s, zeroes, Q( i ) );

        fifo_out( i ) <= Q( i )( fifo_size-1 );
    end generate;

    PROCESS( Q ) -- process for logging the value of FIFO
function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
use Std.TextIO.all;
variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
variable lp: line;
begin
for i in Q'RANGE loop      -- scroll the array of std_logic_vectors
    bv := to_bitvector( Q( i ) ); -- convert to printable value
write( lp, bv );          -- append dynamic string
write( lp, HT );          -- insert horizontal tab
end loop;
return lp.all;            -- return the concatenated string
end;
BEGIN
report array_of_slv_to_string( Q ); -- write internal FIFO contents
END PROCESS;

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 fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    component shift_reg is
        generic(
            reg_size: natural
        );
        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 component;

    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
    signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );

    signal      s_s : std_logic_vector( 1 downto 0 );

```

```

constant    zeroes : std_logic_vector ( fifo_size-1 downto 0 ) := ( others=>'0' );

begin

s_s    <= ( ( not nEnable ) and LOAD )&'0';

registers : for idr in 0 to fifo_width-1 generate
    shift : shift_reg
        generic map(
            reg_size => fifo_size
        )
        port map(
            clk => clk,          nCLR => nCLR,          sr_in => '0',          sl_in => fifo_in( idr ),
            s => s_s,            x => zeroes,            Q => Q( idr )
        );
    fifo_out( idr )    <= Q( idr )( fifo_size-1 );
end generate registers;

PROCESS( Q ) -- process for logging the value of FIFO
    function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
        use Std.TextIO.all;
        variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
        variable lp: line;
    begin
        for i in Q'RANGE loop      -- scroll the array of std_logic_vectors
            bv := to_bitvector( Q( i ) );    -- convert to printable value
            write( lp, bv );    -- append dynamic string
            write( lp, HT );    -- insert horizontal tab
        end loop;
        return lp.all;    -- return the concatenated string
    end;
BEGIN
    report array_of_slv_to_string( Q );    -- write internal FIFO contents
END PROCESS;

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 fifo is
    generic(
        fifo_width: natural := 4;
        fifo_size: natural := 8 );
    PORT ( clk,          nCLR,          nEnable,
           LOAD : IN std_logic;
           fifo_in : in std_logic_vector( fifo_width - 1 downto 0 );
           fifo_out : out std_logic_vector( fifo_width - 1 downto 0 )
    );
end fifo;

architecture NDV of fifo is

    COMPONENT shift_reg
        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( fifo_size-1 downto 0 );
        Q : OUT std_logic_vector( fifo_size-1 downto 0 )
    );
    END COMPONENT;

    constant zeros : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );

    type reg_outs is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size-1 downto 0 );
    signal
        Q : reg_outs;

```

```

        signal                reg_s : std_logic_vector( 1 downto 0 );

begin

    reg_s <= "10" when nEnable = '0' and LOAD = '1' else "00";

    sr: for i in 0 to fifo_width-1 generate
        sr1: shift_reg
            generic map ( reg_size => fifo_size )
            PORT MAP (
                clk => clk,                nCLR => nCLR,                sr_in => '0',                sl_in => fifo_in( i ),
                s => reg_s,                x => zeros,                Q => Q( i )
            );

        fifo_out( i )<= Q( i )( fifo_size-1 );
    end generate;

    PROCESS( Q )
    function array_of_slv_to_string( Q : reg_outs ) return string is
    use Std.TextIO.all;
    variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
    variable lp: line;

    begin
    for i in Q'RANGE loop
    bv := to_bitvector( Q( i ) );
    write( lp, bv );
    write( lp, HT );
    end loop;
    return lp.all;
    end;
    BEGIN
    report array_of_slv_to_string( Q );
    END PROCESS;

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 fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT ( clk, -- signal ure (spremembe na sprednjo fronto)
        nCLR, -- signal za asinhrono brisanje (vsebina FIFO gre na 0)
        nEnable, -- signal za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic; -- signal za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is
    COMPONENT shift_reg
        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( fifo_size-1 downto 0 );
        Q : OUT std_logic_vector( fifo_size-1 downto 0 );
    );
    END COMPONENT;

    constant zeros : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );

    type reg_outs is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size-1 downto 0 );
    signal Q : reg_outs;

```

```

        signal                reg_s : std_logic_vector( 1 downto 0 );

begin

    reg_s <= "10" when nEnable = '0' and LOAD = '1' else "00";
sr: for i in 0 to fifo_width-1 generate
    sr1: shift_reg
        generic map ( reg_size => fifo_size )
        PORT MAP (
            clk => clk,          nCLR => nCLR,          sr_in => '0',          sl_in => fifo_in( i ),
            s => reg_s,          x => zeros,          Q => Q( i )
        );
        fifo_out( i )<= Q( i )( fifo_size-1 );
end generate;

PROCESS( Q ) -- process for logging the value of FIFO
function array_of_slv_to_string( Q : reg_outs ) return string is
use Std.TextIO.all;
variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
variable lp: line;
begin
for i in Q'RANGE loop      -- scroll the array of std_logic_vectors
bv := to_bitvector( Q( i ) ); -- convert to printable value
write( lp, bv );          -- append dynamic string
write( lp, HT );          -- insert horizontal tab
end loop;
return lp.all;            -- return the concatenated string
end;
BEGIN
report array_of_slv_to_string( Q ); -- write internal FIFO contents
END PROCESS;

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 fifo is
    generic(
        fifo_width: natural := 4; -- dolina vhodnega podatka
        fifo_size: natural := 8 ); -- tevilov hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogoanje FIFO ( '0' -> omogoen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal      za omogoanje nalaganja ( '1' -> fifo_in se vpije )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    component shift_reg
        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( fifo_size-1 downto 0 );
        Q : OUT std_logic_vector( fifo_size-1 downto 0 )
    );
    end component;

    constant zeros : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );

```



```

type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );
signal      f_reg_s : std_logic_vector( 1 downto 0 );
begin

f_reg_s      <= "10" when nEnable = '0' and LOAD = '1' else "00";

      sreg: for i in 0 to fifo_width-1 generate
        sreg1: shift_reg
          generic map ( reg_size => fifo_size )
          PORT MAP (
            clk => clk,          nCLR => nCLR,          sr_in => '0',          sl_in => fifo_in( i ),
            s => f_reg_s,        x => zeros,          Q => Q( i )
          );
        fifo_out( i )      <= Q( i )( fifo_size-1 );
      end generate;

      PROCESS( Q ) -- process for logging the value of FIFO
function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
use Std.TextIO.all;
variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
variable lp: line;
begin
for i in Q'RANGE loop      -- scroll the array of std_logic_vectors
bv := to_bitvector( Q( i ) ); -- convert to printable value
write( lp, bv );          -- append dynamic string
write( lp, HT );          -- insert horizontal tab
end loop;
return lp.all;             -- return the concatenated string
end;
BEGIN
report array_of_slv_to_string( Q ); -- write internal FIFO contents
END PROCESS;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210445
-- *****
-- 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 fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,      -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,     -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable,  -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;      -- signal      za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    COMPONENT shift_reg
        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( fifo_size-1 downto 0 );
        Q : OUT std_logic_vector( fifo_size-1 downto 0 )
    );
    END COMPONENT;

    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );

```

```

signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );

constant    zeros : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );
signal      reg_s : std_logic_vector( 1 downto 0 );

begin

reg_s <= "10" when nEnable = '0' and LOAD = '1' else "00";

sr: for i in 0 to fifo_width-1 generate
    sr1: shift_reg
        generic map ( reg_size => fifo_size )
        PORT MAP (
            clk => clk,          nCLR => nCLR,          sr_in => '0',          sl_in => fifo_in( i ),
            s => reg_s,          x => zeros,          Q => Q( i )
        );

        fifo_out( i )<= Q( i )( fifo_size-1 );
end generate;

PROCESS( Q )
    function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
    use Std.TextIO.all;
    variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
    variable lp: line;
    begin
        for i in Q'RANGE loop
            bv := to_bitvector( Q( i ) );
            write( lp, bv );
            write( lp, HT );
        end loop;
        return lp.all;
    end;
BEGIN
    report array_of_slv_to_string( Q );

END PROCESS;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210455
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Rešitev je popolnoma originalna (pohvalno), sledi pa bolj navodilom prosojnic predavanj, kot navodilom
domače naloge. Žal ne dobim rezultatov simulacije – treba je. spremeniti več stvari. Pri komentarju naloge (spodaj) si
oglejte kodo, ki povzema vašo idejo, le da dela točno po navodilih naloge.
-- *****
-- ***** ZAČETEK KODE *****
-- Glavna logika za pomikanje FIFO in obravnavo podatkov
-- Definicija pomikalnega registra (FIFO struktura)
-- Pozor: Definicija polja Q je ravno obrnjena (size * width), ne (width * size)
type shift_reg_array_type is array ( fifo_size - 1 downto 0) of std_logic_vector( fifo_width - 1 downto 0);
signal Q : shift_reg_array_type := (others => (others => '0')); -- Inicializacija registra na 0

-- Signal za sledenje trenutno aktivnemu elementu FIFO
signal write_pointer : integer range 0 to fifo_size - 1 := 0;
signal read_pointer : integer range 0 to fifo_size - 1 := 0;

begin
-- Glavna logika za pomikanje FIFO in obravnavo podatkov
process(clk, nCLR)
begin
    if nCLR = '0' then
        Q <= (others => (others => '0')); -- brisanje FIFO
        write_pointer <= 0; -- ponastavljanje števca za vpis
    elsif rising_edge(clk) then
        if nEnable = '0' then
            if LOAD = '1' then
                Q(write_pointer) <= fifo_in; -- Nalaganje podatka v FIFO
                write_pointer <= (write_pointer + 1) mod fifo_size;
            end if;
        end if;
    end if;
end process;

-- Izhodni podatek iz FIFO
process(clk, nCLR)
begin
    if nCLR = '0' then

```

```

        read_pointer <= 0;                                -- ponastavljanje števca za branje
    elsif rising_edge(clk) then
        if nEnable = '0' then
            read_pointer <= (read_pointer + 1) mod fifo_size;
        end if;
    end if;
end process;

-- Asinhrono branje - podatek je **vedno** na voljo
fifo_out <= Q(read_pointer);
-- ***** KONEC KODE *****

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

entity fifo is
    generic(
        fifo_width: natural := 4; -- Dolžina vhodnega podatka
        fifo_size: natural := 8  -- Število hranjenih podatkov
    );
    PORT(
        clk : in std_logic;      -- Signal      ure ( spremembe na sprednjo fronto )
        nCLR : in std_logic;     -- Signal      za asinhrono brisanje ( FIFO gre na 0 )
        nEnable : in std_logic;  -- Signal      za omogočanje FIFO ( '0' -> omogočen vpis )
        LOAD : in std_logic;     -- Signal      za nalaganje ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- Vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- Izhodni podatek
    );
end fifo;

architecture NDV of fifo is
    -- Definicija pomikalnega registra ( FIFO struktura )
    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
    signal Q : shift_reg_array_type := ( others => ( others => '0' ) ); -- Inicializacija registra na 0

    -- Signal      za sledenje trenutno aktivnemu elementu FIFO
    signal write_pointer : integer range 0 to fifo_size - 1 := 0;
    signal read_pointer : integer range 0 to fifo_size - 1 := 0;

```

```

begin
    -- Glavna logika za pomikanje FIFO in obravnavo podatkov
    process( clk, nCLR )
    begin
        if nCLR = '0' then
            -- Asinhrono brisanje ( FIFO se ponastavi )
            Q      <= ( others => ( others => '0' ) );
            write_pointer      <= 0;
            read_pointer <= 0;
        elsif rising_edge( clk ) then
            if nEnable = '0' then
                if LOAD = '1' then
                    -- Nalaganje podatka v FIFO
                    for i in 0 to fifo_width - 1 loop
                        Q( i )( write_pointer ) <= fifo_in( i );
                    end loop;
                    write_pointer      <= ( write_pointer + 1 ) mod fifo_size;
                end if;
            end if;
        end if;
    end process;

    -- Izhodni podatek iz FIFO
    process( clk )
    begin
        if rising_edge( clk ) then
            if nEnable = '0' then
                fifo_out      <= ( others => '0' );      -- Privzeto
            for i in 0 to fifo_width - 1 loop
                fifo_out( i )      <= Q( i )( read_pointer );
            end loop;
            read_pointer <= ( read_pointer + 1 ) mod fifo_size;
        end if;
    end if;
    end process;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210457
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Uporabljate stare Synopsys 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 fifo is
  generic( fifo_width: natural := 4;    -- dolina vhodnega podatka
    fifo_size: natural := 8 );    -- teviilo hranjenih podatkov
  PORT ( clk, -- signal          ure ( spremembe na sprednjo fronto )
    nCLR, -- signal          za asinhrono brisanje ( vsebina FIFO gre na 0 )
    nEnable, -- signal          za omogoanje FIFO ( '0' -> omogoen vpis, '1' -> ohranja stanje )
    LOAD : IN std_logic; -- signal          za omogoanje nalaganja ( '1' -> fifo_in se vpie )
    fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
    fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
  );
end fifo;

architecture NDV of fifo is    -- vkljucimo shift register
  component 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 component;

  signal      s : std_logic_vector( 1 downto 0 ) := ( others => '0' );
  CONSTANT    zeros : std_logic_vector( fifo_size-1 downto 0 ):= ( others => '0' );
  type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size-1 downto 0 );
  signal      Q : shift_reg_array_type := ( others => ( others => '0' ) );

```

```

begin
  s    <= "10" when ( LOAD = '1' ) and ( nEnable = '0' ) else
  "00";
  fifo: for i in 0 to fifo_width-1 generate
  fifo1 : shift_reg generic map ( reg_size => fifo_size )
  port map(
  clk => clk, nCLR => nCLR, sr_in => '0', sl_in => fifo_in( i ),
  s => s, x => zeros, Q => Q( i )
  );
  fifo_out( i )    <= Q( i )( fifo_size -1 );
end generate;

    -- razhroscevalni process

PROCESS( Q )-- process for logging the value of FIFO
function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
use Std.TextIO.all;
variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
variable lp: line;
begin
  for i in Q'RANGE loop    -- scroll the array of std_logic_vectors
  bv := to_bitvector( Q( i ) ); -- convert to printable value
  write( lp, bv ); -- append dynamic string
  write( lp, HT ); -- insert horizontal tab
  end loop;
  return lp.all;    -- return the concatenated string
end;
BEGIN
  report array_of_slv_to_string( Q ); -- write internal FIFO contents
END PROCESS;
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 fifo is
    generic(
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT ( clk, -- signal ure (spremembe na sprednjo fronto)
        nCLR, -- signal za asinhrono brisanje (vsebina FIFO gre na 0)
        nEnable, -- signal za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic; -- signal za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture NDV of fifo is

    component 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 component;

    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
    signal Q : shift_reg_array_type;
    -- pomožna spremenljivka
    signal s_temp : std_logic_vector( 1 downto 0 ) := "00";

```

```

constant    zeros : std_logic_vector( fifo_size - 1 downto 0 ) := ( others => '0' );

begin

    s_temp <= "10" when nEnable = '0' and LOAD = '1'
               else "00";

    povezovanje:
    for i in 0 to fifo_width - 1 generate
        U1: shift_reg
            generic map ( reg_size => fifo_size )
            port map ( clk => clk,          nCLR => nCLR,          sr_in => '0',          sl_in =>
fifo_in( i ),          s => s_temp,          x => zeros,          Q => Q( i )
            );
        fifo_out( i ) <= Q( i )( fifo_size - 1 );
    end generate;

    PROCESS( Q ) -- process for logging the value of FIFO
        function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
            use Std.TextIO.all;

            variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
            variable lp: line;

            begin
                for i in Q'RANGE loop          -- scroll the array of std_logic_vectors
                    bv := to_bitvector( Q( i ) ); -- convert to printable value
                    write( lp, bv );           -- append dynamic string
                    write( lp, HT );           -- insert horizontal tab
                end loop;
                return lp.all;                 -- return the concatenated string
            end;

        BEGIN

            report array_of_slv_to_string( Q ); -- write internal FIFO contents

        END PROCESS;
    end NDV;

```

```

-- *****
-- **** PREDLOGA VAJE
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fifo is
    generic (
        fifo_width: natural := 4; -- dolžina vhodnega podatka
        fifo_size: natural := 8 ); -- število hranjenih podatkov
    PORT (
        clk,    -- signal      ure ( spremembe na sprednjo fronto )
        nCLR,   -- signal      za asinhrono brisanje ( vsebina FIFO gre na 0 )
        nEnable, -- signal      za omogočanje FIFO ( '0' -> omogočen vpis, '1' -> ohranja stanje )
        LOAD : IN std_logic;    -- signal za omogočanje nalaganja ( '1' -> fifo_in se vpiše )
        fifo_in : in std_logic_vector( fifo_width - 1 downto 0 ); -- vhodni podatek
        fifo_out : out std_logic_vector( fifo_width - 1 downto 0 ) -- izhodni podatek
    );
end fifo;

architecture ideal of fifo is
    signal s : std_logic_vector( 1 downto 0 );
    type shift_reg_array_type is array ( fifo_width - 1 downto 0 ) of std_logic_vector( fifo_size - 1 downto 0 );
    signal Q : shift_reg_array_type := ( others => ( others => '0' ) );
    constant zeroes : std_logic_vector( fifo_size-1 downto 0 ) := ( others => '0' );
    constant zero : std_logic := '0';
    COMPONENT 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 COMPONENT;

begin
    -- Load new data ( shift left ) or hold contents when fifo disabled ( nEnable = '1' )
    s <= "10" when ( nEnable = '0' and LOAD = '1' ) else "00";

```

```

L1: FOR i IN 0 TO fifo_width - 1 GENERATE
    U0: shift_reggeneric map ( reg_size => fifo_size )
    port map ( clk, nCLR, zero, fifo_in( i ), s, zeroes, Q( i ) );
    fifo_out( i )<= Q( i )( fifo_size - 1 );
END GENERATE;

PROCESS( Q ) -- process for logging the value of FIFO
    function array_of_slv_to_string( Q : shift_reg_array_type ) return string is
        use Std.TextIO.all;
        variable bv: bit_vector( Q( Q'left )'range ) := to_bitvector( Q( Q'left ) );
        variable lp: line;
        begin
            for i in Q'RANGE loop
                bv := to_bitvector( Q( i ) );
                write( lp, bv );
                write( lp, HT );
            end loop;
            return lp.all;
        end;
    BEGIN
        report array_of_slv_to_string( Q );    -- write internal FIFO contents
    END PROCESS;

end ideal;

```

