

-- **** 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: Ni pripomb.....	13
-- **** STUDENT: 64200288.....	15
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni rezultatov simulacije zaradi napak sintetizatorja – ne vem od kod se vam je na začetku datoteke vrinila beseda bus: ERROR:HDLCompiler:806 - "muxnto1_bus.vhd" Line 1: Syntax error near "busLIBRARY".	15
Če besedo odstranim, koda deluje pravilno.	15
-- **** STUDENT: 64200296.....	17
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	17
-- **** STUDENT: 64200385.....	19
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	19
-- **** STUDENT: 64210113.....	21
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	21
-- **** STUDENT: 64210290.....	23
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	23
-- **** STUDENT: 64210382.....	25
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	25
-- **** STUDENT: 64210384.....	27
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	27
-- **** STUDENT: 64210386.....	29
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	29
-- **** STUDENT: 64210445.....	31
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	31
-- **** STUDENT: 64210455.....	33

-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	33
-- **** STUDENT: 64210457		35
-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	35
-- **** STUDENT: 64240430		37
-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	37
-- **** PREDLOGA VAJE		39
-- KOMENTARJI K OCENI NALOGE	-- Matej Možek: Ni pripomb.....	39

```

-- *****
-- **** STUDENT: 64000225
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

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

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN     std_logic_vector( n_addr - 1 downto 0 );
        w            : IN     muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT    std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS
    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s      : in  std_logic_vector( n_addr - 1 downto 0 );
              w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
              f      : OUT STD_LOGIC
        );
    END COMPONENT;

BEGIN
    process ( w ) is
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    begin
        for idx in 0 to bus_width-1 loop
            for jdx in 0 to 2**n_addr - 1 loop
                mux_addr_col( jdx ) := w( jdx, idx );
            end loop;
        end loop;
    end process;

```

```

        end loop;
        mux_addr( idx )    <= mux_addr_col;
    end loop;
end process;

muxes: for idx in 0 to bus_width-1 generate
    mux: muxnto1
        generic map ( n_addr => n_addr )
        port map (
            s => s,          w => mux_addr( idx ),          f => f( idx )
        );
end generate;

```

```

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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width   : INTEGER := 8 );
    PORT (
        s           : IN    std_logic_vector( n_addr - 1 downto 0 );
        w           : IN    muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f           : OUT   std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS
type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

component muxnto1 IS
    generic( n_addr: natural := 2 );
    PORT ( s      : in   std_logic_vector( n_addr - 1 downto 0 );
          w      : in   std_logic_vector( 2**n_addr - 1 downto 0 );
          f      : OUT  STD_LOGIC
    );
END component;

BEGIN

    process( s, w )
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
        begin
            for i in 0 to bus_width - 1 loop
                for j in 0 to 2**n_addr -1 loop

```

```
        mux_addr_col( j ) := w( j,i );  
    end loop;  
    mux_addr( i )<= mux_addr_col;  
    end loop;
```

```
end process;
```

```
for_gen: for k in 0 to bus_width - 1 generate
```

```
    u0: muxnto1
```

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

```
        port map ( s => s, w => mux_addr( k ), f => f( k ) );
```

```
end generate;
```

```
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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width   : INTEGER := 8 );
    PORT (
        s           : IN      std_logic_vector( n_addr - 1 downto 0 );
        w           : IN      muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f           : OUT     std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS
    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal mux_addr : mux_addr_type := ( others => ( others => '0' ) );
    COMPONENT muxnto1 IS
        generic( n_addr: natural:=2 );
        PORT ( s:in std_logic_vector( n_addr-1 downto 0 );
        w:in std_logic_vector( 2**n_addr-1 downto 0 );
        f:OUT STD_LOGIC );
    END COMPONENT;
BEGIN
    process ( w ) is
        variable mux_addr_col: std_logic_vector( 2**n_addr-1 downto 0 );
    begin
        for i in 0 to bus_width-1 loop
            for j in 0 to 2**n_addr - 1 loop
                mux_addr_col( j ) := w( j, i );
            end loop;
            mux_addr( i )<= mux_addr_col;
        end loop;
    end process;
end architecture;

```

```
end process;
mux_mux: for i in 0 to bus_width-1 generate
mux1: muxnto1
generic map ( n_addr => n_addr )
port map ( s => s,w => mux_addr( i ),f => f( i ) );
end generate;
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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN    std_logic_vector( n_addr - 1 downto 0 );
        w            : IN    muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT   std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS
type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

COMPONENT muxnto1 IS
    generic( n_addr: natural := 2 );
    PORT ( s      : in   std_logic_vector( n_addr - 1 downto 0 );
          w      : in   std_logic_vector( 2**n_addr - 1 downto 0 );
          f      : OUT  STD_LOGIC
    );
END COMPONENT;

BEGIN

    P : process ( w )
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    begin
        for i in 0 to bus_width - 1 loop
            for j in 0 to 2**n_addr - 1 loop

```

```
        mux_addr_col( j ) := w( j, i );  
    end loop;  
    mux_addr( i )<= mux_addr_col;  
end loop;  
end process;
```

```
G : for g in 0 to bus_width - 1 generate
```

```
    U : muxnto1
```

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

```
        port map( s => s,          w => mux_addr( g ),  
                );
```

```
end generate;
```

```
f => f( g )
```

```
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;
use work.reg_file_functions.all;

entity muxnto1_bus is
    generic( n_addr : integer := 2;
             bus_width : integer := 8 );
    port( s : in std_logic_vector( n_addr - 1 downto 0 );
          w : in muxnto1_bus_type( 2*n_addr - 1 downto 0, bus_width - 1 downto 0 );
          f : out std_logic_vector( bus_width - 1 downto 0 )
    );
end muxnto1_bus;

architecture NDV of muxnto1_bus is
    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2*n_addr - 1 downto 0 );
    signal mux_addr : mux_addr_type := ( others => ( others => '0' ) );
    component muxnto1 is
        generic( n_addr : natural := 2 );
        port( s : in std_logic_vector( n_addr - 1 downto 0 );
              w : in std_logic_vector( 2*n_addr - 1 downto 0 );
              f : out std_logic
        );
    end component;
begin
    process( w )
        variable mux_addr_col : std_logic_vector( 2*n_addr - 1 downto 0 );
    begin
        for i in 0 to bus_width - 1 loop
            for j in 0 to 2*n_addr - 1 loop
                mux_addr_col( j ) := w( j, i );
            end loop;
            mux_addr( i ) <= mux_addr_col;
        end loop;
    end process;
end architecture NDV;

```

```
end process;
mux: for i in 0 to bus_width - 1 generate
    U1 : muxnto1
        generic map( n_addr => n_addr )
        port map(
            s => s,          w => mux_addr( i ),          f => f( i )
        );
    end generate;
end NDV;
```

```

-- *****
-- **** STUDENT: 64200238
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN    std_logic_vector( n_addr - 1 downto 0 );
        w : IN muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f            : OUT   std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS

    component muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s      : in  std_logic_vector( n_addr - 1 downto 0 );
              w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
              f      : OUT STD_LOGIC
            );
    END component muxnto1;

    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

BEGIN

    PROCESS( w )

        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    BEGIN
        for i in 0 to bus_width-1 loop

```

```

        for j in 0 to 2**n_addr-1 loop
            mux_addr_col( j ) := w( j,i );

        end loop;
    mux_addr( i )<=mux_addr_col;
end loop;

END PROCESS;
U1 :for i in 0 to bus_width-1 generate
U2: muxnto1
    generic map( n_addr=>n_addr )
    port map(
        s => s, w => mux_addr( i ), f => f( i )
    );
end generate;

END NDV;

```

```

-- *****
-- **** STUDENT: 64200288
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni rezultatov simulacije zaradi napak sintetizatorja – ne vem od kod se vam je na začetku datoteke
vrnila beseda bus: ERROR:HDLCompiler:806 - "muxnto1_bus.vhd" Line 1: Syntax error near "busLIBRARY".
Če besedo odstranim, koda deluje pravilno.
-- *****
busLIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN     std_logic_vector( n_addr - 1 downto 0 );
        w            : IN     muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT    std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS
    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s      : in  std_logic_vector( n_addr - 1 downto 0 );
              w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
              f      : OUT STD_LOGIC
        );
    END COMPONENT;
BEGIN
    process ( w ) is
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    begin
        for A in 0 to bus_width-1 loop
            for B in 0 to 2**n_addr - 1 loop

```

```

        mux_addr_col( B ) := w( B, A );
    end loop;
    mux_addr( A )<= mux_addr_col;
end loop;
end process;

Multiplexers: for A in 0 to bus_width-1 generate
    mux: muxnto1
        generic map ( n_addr => n_addr )
        port map (
            s => s,                w => mux_addr( A ),          f => f( A )
        );
    end generate;
END NDV;

```



```

-- *****
-- **** STUDENT: 64200296
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN    std_logic_vector( n_addr - 1 downto 0 );
        w            : IN    muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT   std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS
type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

COMPONENT muxnto1 IS
    generic( n_addr: natural := 2 );
    PORT ( s      : in   std_logic_vector( n_addr - 1 downto 0 );
          w      : in   std_logic_vector( 2**n_addr - 1 downto 0 );
          f      : OUT  STD_LOGIC
    );
END COMPONENT;
BEGIN
    PROCESS ( w )
    VARIABLE mux_addr_col : std_logic_vector( 2**n_addr - 1 DOWNT0 0 );
    BEGIN
        -- Pretvorba dvodimenzionalnega polja v polje vektorjev
        FOR i IN 0 TO bus_width - 1 LOOP
        FOR j IN 0 TO 2**n_addr - 1 LOOP
            mux_addr_col( j ) := w( j, i );
        
```

```

END LOOP;
mux_addr( i )      <= mux_addr_col;
END LOOP;
END PROCESS;

-- Uporaba komponent za generiranje izhodnega signala
gen_mux: FOR i IN 0 TO bus_width - 1 GENERATE
mux_inst: muxnto1
  GENERIC MAP (
    n_addr => n_addr
  )
  PORT MAP (
    s => s,
    w => mux_addr( i ),
    f => f( i )
  );
END GENERATE gen_mux;
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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN     std_logic_vector( n_addr - 1 downto 0 );
        w            : IN     muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT    std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS

    component muxnto1 is
        generic( n_addr: natural := 2 );
        PORT ( s    : in  std_logic_vector( n_addr - 1 downto 0 );
              w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
              f      : OUT STD_LOGIC
        );
    end component;

    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal
        mux_addr : mux_addr_type := ( others =>( others => '0' ) );

BEGIN

    process( w ) is
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    begin
        for i in 0 to bus_width - 1 loop

```

```

for j in 0 to 2**n_addr - 1 loop
mux_addr_col( j ) := w( j,i );
end loop;
mux_addr( i )      <= mux_addr_col;
end loop;
end process;

mbus: for i in 0 to bus_width - 1 generate
U0: muxnto1
generic map ( n_addr => n_addr )
port map ( s => s,
w => mux_addr( i ),
f => f( i ) );
end generate;

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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN     std_logic_vector( n_addr - 1 downto 0 );
        w            : IN     muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT    std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS

    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s      : in  std_logic_vector( n_addr - 1 downto 0 );
              w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
              f      : OUT STD_LOGIC
        );
    END COMPONENT;

BEGIN

    process ( w ) is
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );

    begin

```

```

    for i in 0 to bus_width-1 loop

        for j in 0 to 2**n_addr - 1 loop
            mux_addr_col( j ) := w( j, i );
        end loop;

        mux_addr( i )<= mux_addr_col;
    end loop;

end process;

PovMux: for m in 0 to bus_width-1 generate
    mux: muxnto1

        generic map ( n_addr => n_addr )
        port map (
            s => s,                w => mux_addr( m ),          f => f( m )
        );

    end generate;
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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    GENERIC(
        n_addr : INTEGER := 2;
        bus_width : INTEGER := 8
    );
    PORT(
        s : IN STD_LOGIC_VECTOR( n_addr - 1 DOWNT0 0 );
        w : IN muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT STD_LOGIC_VECTOR( bus_width - 1 DOWNT0 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS

    component muxnto1 is
        generic(
            n_addr : natural := 2
        );
        port(
            s : in std_logic_vector( n_addr - 1 downto 0 );
            w : in std_logic_vector( 2**n_addr - 1 downto 0 );
            f : out std_logic
        );
    end component;

    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal mux_addr : mux_addr_type := ( others => ( others => '0' ) );

BEGIN

```

```

wires : process( w )
begin
    for ver in 0 to bus_width-1 loop
        for hor in 0 to 2**n_addr-1 loop
            mux_addr( ver )( hor )    <= w( hor,ver );
        end loop;
    end loop;
end process;

mux_gen : for i in 0 to bus_width-1 generate
    mux : muxnto1
        generic map(
            n_addr => n_addr
        )
        port map(
            s => s,           w => mux_addr( i ),           f => f( i )
        );
    end generate mux_gen;

```

```

END NDV;

```



```

-- *****
-- **** STUDENT: 64210382
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN     std_logic_vector( n_addr - 1 downto 0 );
        w            : IN     muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT    std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS
type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
signal mux_addr : mux_addr_type := ( others => ( others => '0' ) );
COMPONENT muxnto1 IS
    generic( n_addr: natural := 2 );
    PORT ( s      : in  std_logic_vector( n_addr - 1 downto 0 );
          w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
          f      : OUT STD_LOGIC
    );
END COMPONENT;
BEGIN
process ( w ) is
    variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
begin
    for idx in 0 to bus_width-1 loop
        for jdx in 0 to 2**n_addr - 1 loop
            mux_addr_col( jdx ) := w( jdx, idx );
        end loop;
        mux_addr( idx )    <= mux_addr_col;
    end loop;
end process;

```

```

        end loop;
    end process;

    muxes: for idx in 0 to bus_width-1 generate
        mux: muxnto1
            generic map ( n_addr => n_addr )
            port map (
                s => s,                w => mux_addr( idx ),        f => f( idx )
            );
        end generate;
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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT ( s
        : IN      std_logic_vector( n_addr - 1 downto 0 );
        w
        : IN      muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f
        : OUT     std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS

    component muxnto1
        generic ( n_addr : natural := 2 );
        port(
            s : in std_logic_vector( n_addr - 1 downto 0 );
            w : in std_logic_vector( 2**n_addr - 1 downto 0 );
            f : out std_logic
        );
    end component;

    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

BEGIN

    process ( w )
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    begin

```

```

        for j in 0 to bus_width - 1 loop
            for i in 0 to 2**n_addr - 1 loop
                mux_addr_col( i ) := w( i,j );
            end loop;
            mux_addr( j )<= mux_addr_col;
        end loop;
    end process;

```

```

povezava:
for i in 0 to bus_width - 1 generate
    blok: muxnto1
        generic map ( n_addr => n_addr )
        port map(
            s => s,                w => mux_addr( i ),          f => f( i )
        );
    end generate;

```

```

END NDV;

```

```

-- *****
-- **** STUDENT: 64210386
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN    std_logic_vector( n_addr - 1 downto 0 );
        w            : IN    muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT   std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS
    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

    component muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s      : in   std_logic_vector( n_addr - 1 downto 0 );
              w      : in   std_logic_vector( 2**n_addr - 1 downto 0 );
              f      : OUT  STD_LOGIC
        );
    end component;

BEGIN

    process ( w ) is

        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );

    begin

```

```

    for st in 0 to bus_width-1 loop
        for vr in 0 to 2**n_addr - 1 loop
            mux_addr_col( vr ) := w( vr, st );
        end loop;
        mux_addr( st )      <= mux_addr_col;
    end loop;

```

```

end process;

```

```

muxs: for i in 0 to bus_width-1 generate
    mux: muxnto1
        generic map ( n_addr => n_addr )
        port map (
            s => s,                w => mux_addr( i ),
        );
    end generate;

```

```

f => f( i )

```

```

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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN     std_logic_vector( n_addr - 1 downto 0 );
        w            : IN     muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT    std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS

    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s      : in   std_logic_vector( n_addr - 1 downto 0 );
              w      : in   std_logic_vector( 2**n_addr - 1 downto 0 );
              f      : OUT  STD_LOGIC );
    END COMPONENT;

BEGIN

    process ( w ) is
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    begin

        for idx in 0 to bus_width-1 loop

```

```

        for jdx in 0 to 2**n_addr - 1 loop
            mux_addr_col( jdx ) := w( jdx, idx );
        end loop;
        mux_addr( idx )      <= mux_addr_col;
    end loop;

```

```

end process;

```

```

muxes: for idx in 0 to bus_width-1 generate

```

```

    mux: muxnto1

```

```

        generic map ( n_addr => n_addr )

```

```

        port map ( s => s,          w => mux_addr( idx ),

```

```

    end generate;

```

```

    f => f( idx ) );

```

```

END NDV;

```



```
-- *****
-- **** STUDENT: 64210455
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
USE work.reg_file_functions.all;
```

```
ENTITY muxnto1_bus IS
  GENERIC (
    n_addr : INTEGER := 2;
    bus_width : INTEGER := 8
  );
  PORT (
    s : IN STD_LOGIC_VECTOR( n_addr - 1 DOWNT0 0 );
    w : IN muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
    f : OUT STD_LOGIC_VECTOR( bus_width - 1 DOWNT0 0 )
  );
END muxnto1_bus;
```

```
ARCHITECTURE Behavioral OF muxnto1_bus IS
  TYPE mux_addr_type IS ARRAY ( bus_width - 1 DOWNT0 0 ) OF STD_LOGIC_VECTOR( 2**n_addr - 1 DOWNT0 0 );
  SIGNAL
    mux_addr : mux_addr_type := ( OTHERS => ( OTHERS => '0' ) );
```

```
  COMPONENT muxnto1 IS
    GENERIC ( n_addr : NATURAL := 2 );
    PORT (
      s : IN STD_LOGIC_VECTOR( n_addr - 1 DOWNT0 0 );
      w : IN STD_LOGIC_VECTOR( 2**n_addr - 1 DOWNT0 0 );
      f : OUT STD_LOGIC
    );
  END COMPONENT;
```

```
BEGIN
  PROCESS ( w )
    VARIABLE column_data : STD_LOGIC_VECTOR( 2**n_addr - 1 DOWNT0 0 );
  BEGIN
    FOR col_idx IN 0 TO bus_width - 1 LOOP
```

```

FOR row_idx IN 0 TO 2**n_addr - 1 LOOP
column_data( row_idx ) := w( row_idx, col_idx );
END LOOP;
mux_addr( col_idx )      <= column_data;
END LOOP;
END PROCESS;

gen_muxes: FOR col_idx IN 0 TO bus_width - 1 GENERATE
single_mux: muxnto1
GENERIC MAP ( n_addr => n_addr )
PORT MAP (
s => s,
w => mux_addr( col_idx ),
f => f( col_idx )
);
END GENERATE;
END Behavioral;

```

```

-- *****
-- **** STUDENT: 64210457
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN     std_logic_vector( n_addr - 1 downto 0 );
        w            : IN     muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT    std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS
    type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
    signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s      : in   std_logic_vector( n_addr - 1 downto 0 );
              w      : in   std_logic_vector( 2**n_addr - 1 downto 0 );
              f      : OUT  STD_LOGIC
        );
    END COMPONENT;
BEGIN
    process ( w ) is
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    begin
        for idx in 0 to bus_width-1 loop
            for jdx in 0 to 2**n_addr - 1 loop
                mux_addr_col( jdx ) := w( jdx, idx );
            end loop;
        end loop;
    end process;

```

```

        mux_addr( idx )    <= mux_addr_col;
    end loop;
end process;

muxes: for idx in 0 to bus_width-1 generate
    mux: muxnto1
        generic map ( n_addr => n_addr )
        port map (
            s => s,          w => mux_addr( idx ),          f => f( idx )
        );
    end generate;
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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS
    generic(
        n_addr      : INTEGER := 2;
        bus_width    : INTEGER := 8 );
    PORT (
        s            : IN      std_logic_vector( n_addr - 1 downto 0 );
        w            : IN      muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f            : OUT     std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE NDV OF muxnto1_bus IS

component muxto1
    generic ( n_addr : natural := 2 );
    port (
        s : in std_logic_vector( n_addr - 1 downto 0 );
        w : in std_logic_vector( 2**n_addr - 1 downto 0 );
        f : out std_logic
    );
end component;

type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

BEGIN

    process( w )
        variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    begin

```

```

        for i in 0 to bus_width - 1 loop
            for j in 0 to 2**n_addr - 1 loop
                mux_addr_col( j ) := w( j, i );
            end loop;
            mux_addr( i ) <= mux_addr_col;
        end loop;
    end process;

    pov:
    for i in 0 to bus_width - 1 generate
        mux: muxnto1
            generic map ( n_addr => n_addr )
            port map (
                s => s,                w => mux_addr( i ),          f => f( i )
            );
        end generate;
    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;
USE work.reg_file_functions.all;

ENTITY muxnto1_bus IS

    generic( n_addr      :    INTEGER := 2;
             bus_width   :    INTEGER := 8 );
    PORT (
        s      :    IN    std_logic_vector( n_addr - 1 downto 0 );
        w      :    IN    muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1
DOWNT0 0 );
        f      :    OUT   std_logic_vector( bus_width - 1 downto 0 )
    );
END muxnto1_bus;

ARCHITECTURE ideal OF muxnto1_bus IS

COMPONENT muxnto1 IS
    generic( n_addr: natural := 2 );
    PORT (
        s : in    std_logic_vector( n_addr - 1 downto 0 );
        w : in    std_logic_vector( 2**n_addr - 1 downto 0 );
        f : OUT   STD_LOGIC
    );
END COMPONENT;

type mux_addr_type is array ( bus_width - 1 downto 0 ) of std_logic_vector( 2**n_addr - 1 downto 0 );
signal      mux_addr : mux_addr_type := ( others =>( others => '0' ) );

BEGIN
    bus_mux_process: PROCESS( w )
    variable mux_addr_col : std_logic_vector( 2**n_addr - 1 downto 0 );
    BEGIN

```

```

        FOR j IN 0 TO bus_width - 1 LOOP
            FOR i IN 0 TO 2**n_addr - 1 LOOP
                mux_addr_col( i ) := w( i, j );
            END LOOP;
            mux_addr( j ) <= mux_addr_col;
        END LOOP;
    END PROCESS;

muxloop: FOR k IN 0 TO bus_width - 1 GENERATE
    Ui: muxnto1 generic map ( n_addr => n_addr )
        port map ( s => s, w => mux_addr( k ),
        );
    END GENERATE;

END ideal;

```

f => f(k)

