

Ocenjevanje n-bitne ALU

Ocenjevanje n-bitne ALU	1
-- **** STUDENT: 64000225	3
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	3
-- **** STUDENT: 64190088	5
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Negative se mora postaviti v primeru “operacije Y”, ker gre za realizacijo zbirniškega mikroukaza TST. 5	
-- **** STUDENT: 64200100	7
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Gout je nedefiniran za operacijo seštevanja, kar je posledica napačne izvedbe na CLA seštevalniku. 7	
-- **** STUDENT: 64200112	10
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	10
-- **** STUDENT: 64200163	12
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	12
-- **** STUDENT: 64200238	14
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	14
-- **** STUDENT: 64200288	18
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	18
-- **** STUDENT: 64200296	22
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	22
-- **** STUDENT: 64200385	24
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Gout je napačen za operacijo seštevanja (ko je rezultat 0), kar je posledica napačne izvedbe na CLA seštevalniku.....	24
-- **** STUDENT: 64210113	26
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	26
-- **** STUDENT: 64210290	29
-- KOMENTARJI K OCENI NALOGE -- Matej Možek:	29
Bit N ne deluje pravilno za zmanjševanje (X-1). Primer -128 -1 ni enako -127, ampak 127. Takrat se postavi tudi bit preliava V in izhodni prenos C.	29
Bit Z ne deluje pravilno za seštevanje. Primer $14+(-14)=0$	29
Bit Z se postavi, ko je rezultat odštevanja NI enak 0 ($14-15=-1$, postavi se vam $Z=1$).	29
Bit V ni realiziran za odštevanje, povečevanje, zmanjševanje in podvojevanje ampak samo za seštevanje.....	29

-- **** STUDENT: 64210382.....	32
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	32
-- **** STUDENT: 64210384.....	34
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	34
-- **** STUDENT: 64210386.....	36
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Gout je nedefiniran za operacijo seštevanja, kar je posledica napačne izvedbe na CLA seštevalniku. 36	
-- **** STUDENT: 64210445.....	38
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Gout je nedefiniran za operacijo seštevanja, kar je posledica napačne izvedbe na CLA seštevalniku. 38	
-- **** STUDENT: 64210455.....	40
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Napake sintetizatorja:	40
ERROR:HDLCompiler:136 - "alu_cla.vhd" Line 45: OTHERS choice cannot be used in unconstrained array aggregate.	40
Podobno kot ste imeli pri cla_adder.vhd – operacije xor ne morete izvesti nad neomejenim vektorjem. Definirati morate interni signal (npr. add_sub_vector), ki ga nato povežete na vrednost (add_sub) in šele potem uporabite nad rezultatom:	40
signal add_sub_vector : std_logic_vector(n-1 downto 0); -- zdaj je velikost omejena na n elementov	40
nato ga povežete z uporabo others:	40
add_sub_vector <= (others => add_sub);.....	40
in nato uporabite pri tvorbi xor:	40
Y_mod <= (Y xor add_sub_vector);	40
-- **** STUDENT: 64210457.....	42
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	42
-- **** STUDENT: 64240429.....	44
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Pri “operaciji -1” se mora postaviti bit N=1, ker gre za negativno število na izhodu. Negative se mora postaviti v primeru “operacije Y”, ker gre za realizacijo zbirniškega mikroukaza TST.	44
-- **** STUDENT: 64240430.....	47
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	47
-- **** PREDLOGA VAJE	50
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	50

```

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

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M          : in    std_logic;  -- naèin delovanja ( '0' => aritmetični, '1' => logični )
          F          : in    std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
          X, Y       : in    std_logic_vector( n-1 downto 0 );
          S          : out   std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out   std_logic );
end alu_cla;

architecture NDV of alu_cla is

    COMPONENT cla_add_n_bit IS
        generic( n: natural := 8 );
        PORT ( Cin      : in    std_logic ;
              X, Y      : in    std_logic_vector( n-1 downto 0 );
              S         : out   std_logic_vector( n-1 downto 0 );
              Gout, Pout, Cout : out   std_logic );
    END COMPONENT;

    signal S_sig, Y_sig : std_logic_vector( n-1 downto 0 );
    signal nAddSub      : std_logic;
    signal alu_operation : std_logic_vector( 3 downto 0 );

    constant zeros_0 : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    constant one_1   : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' );

begin

    U1: cla_add_n_bit
        generic map ( n => n )

```

```

port map (
    Cin => nAddSub, X => X, Y => Y_sig, S => S_sig, Gout => Gout, Pout => Pout, Cout => Cout
);

alu_operation    <= M & F;

nAddSub          <= alu_operation( 0 );

Overflow         <= ( not X( n-1 ) and not Y_sig( n-1 ) and S_sig( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and
not S_sig( n-1 ) );

    with alu_operation select Negative    <=
        '1' when "0101",                X( n-1 ) when "1110",                Y( n-1 )
    when "1111",                S_sig( n-1 ) when others;

Zero    <= '0' when alu_operation = "0101" else
    '1' when alu_operation = "1110" and X = zeros_0 else
    '0' when alu_operation = "1110" and X /= zeros_0 else
    '1' when alu_operation = "1111" and Y = zeros_0 else
    '0' when alu_operation = "1111" and Y /= zeros_0 else
    '1' when S_sig = zeros_0 else '0';

    with alu_operation select S    <=
        S_sig when "0000",                S_sig when "0001",                S_sig when "0010",
    S_sig when "0011",                S_sig when "0100",                not zeros_0 when "0101",                X and Y
    when "1000",                X nand Y when "1001",                X or Y when "1010",                X nor Y when "1011",
    X xor Y when "1100",                X xnor Y when "1101",                X when "1110",                Y
    when "1111",                zeros_0 when others;

    with alu_operation select Y_sig <=
        Y when "0000",                not Y when "0001",                one_1 when "0010",                not one_1
    when "0011",                X when "0100",                Y when others;

end NDV;

```

```
-- *****
-- **** STUDENT: 64190088
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Negative se mora postaviti v primeru "operacije Y", ker gre za realizacijo zbirniškega mikroukaza TST.
-- *****
```

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity alu_cla is
```

```
    generic( n: natural := 8 );
```

```
    port( M      : in      std_logic;  -- naèin delovanja ( '0' => aritmetični, '1' => logični )
          F      : in      std_logic_vector( 2 downto 0 );  -- funkcijski vhod za operacije
          X, Y    : in      std_logic_vector( n-1 downto 0 );
          S      : out     std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero,  Gout, Pout : out     std_logic );
```

```
end alu_cla;
```

```
architecture NDV of alu_cla is
```

```
    component cla_add_n_bit IS
```

```
    generic( n: natural := 8 );
```

```
    PORT (
          Cin      : in      std_logic ;
          X, Y     : in      std_logic_vector( n-1 downto 0 );
          S        : out     std_logic_vector( n-1 downto 0 );
          Gout, Pout, Cout : out     std_logic );
```

```
    END component;
```

```
    signal S_sig, Y_sig : std_logic_vector( n-1 downto 0 );
```

```
    signal nAddSub : std_logic;
```

```
    signal alu_operation : std_logic_vector( 3 downto 0 );
```

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

```
    constant one : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' );
```

```
    constant ones : std_logic_vector ( n-1 downto 0 ) := ( others => '1' );
```

```
begin
```

```
    U1: cla_add_n_bit generic map ( n => n )
```

```
    port map ( Cin => nAddSub, X => X, S => S_sig, Y=> Y_sig, Cout => Cout, Pout => Pout, Gout =>Gout );
```

```

alu_operation<= M & F;
nAddSub      <= alu_operation( 0 );

Overflow      <= ( not X( n-1 ) and not Y_sig( n-1 ) and S_sig( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and not
S_sig( n-1 ) );

Negative      <= S_sig( n-1 );
Zero         <= '1' when ( S_sig = zeros ) else '0';

with alu_operation select Y_sig <=
    Y          when "0000", not Y when "0001", one    when "0010", not one when "0011",      X
when "0100", Y          when others;

with alu_operation select S      <=
    S_sig when "0000", S_sig when "0001", S_sig when "0010", S_sig when "0011", S_sig when "0100", ones
when "0101", X and Y      when "1000", X nand Y when "1001",      X or Y      when "1010", X nor Y      when
"1011",      X xor Y      when "1100", X xnor Y when "1101",      X          when "1110", Y
when "1111", zeros when others;

end NDV;

```

```

-- *****
-- **** STUDENT: 64200100
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Gout je nedefiniran za operacijo seštevanja, kar je posledica napačne izvedbe na CLA seštevalniku.
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in      std_logic;    -- nain delovanja ( '0' => aritmetini, '1' => logini )
          F      : in      std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
          X, Y    : in      std_logic_vector( n-1 downto 0 );
          S      : out std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out  std_logic );
end alu_cla;

architecture NDV of alu_cla is
    component cla_add_n_bit is
        generic( n: natural:= 8 );
        port( Cin: in std_logic;
              X,Y: in std_logic_vector( n-1 downto 0 );
              S: out std_logic_vector( n-1 downto 0 );
              Gout, Pout, Cout: out std_logic
              );
    end component;

    signal      S_sig, Y_sig : std_logic_vector( n-1 downto 0 );
    signal      naddsub: std_logic;
    signal      alu_operation: std_logic_vector( 3 downto 0 );
    constant    nic: std_logic_vector( n-1 downto 0 ):= ( others=>'0' );
    constant    pEna : std_logic_vector( n-1 downto 0 ):= ( 0=>'1', others=>'0' );

begin
    --
    --
    U1: cla_add_n_bit
        generic map( n=>n )
        port map( Cin => naddsub, X=>X, Y=>Y_sig,S=>S_sig,Gout=>Gout,Pout=>Pout,Cout=>Cout );

```

```

alu_operation<=M&F;
naddsub      <=alu_operation( 0 );

Overflow      <=( not X( n-1 )and not Y_sig( n-1 ) and S_sig( n-1 ) ) or ( X( n-1 )and Y_sig( n-1 )and not S_sig( n-1 )
);
with alu_operation select Negative      <=
'1' when "0101", X( n-1 ) when "1110", Y( n-1 ) when "1111", S_sig( n-1 ) when others;

Zero  <='0' when alu_operation="0101" else
'1' when alu_operation ="1110" and X = nic else
'0' when alu_operation="1110" and X /= nic else
'1' when alu_operation="1111" and Y = nic else
'0' when alu_operation="1111" and Y /= nic else
'1' when S_sig=nic else '0';

with alu_operation select S      <=
S_sig when "0000",
S_sig when "0001",
S_sig when "0010",
S_sig when "0011",
S_sig when "0100",
not nic when "0101",

X and Y when "1000",
X nand Y when "1001",
X OR Y when "1010",
X nor Y when "1011",
X xor Y when "1100",
X xnor Y when "1101",
X when "1110",
Y when "1111",
nic when others;

with alu_operation select Y_sig <=
Y when "0000",
not Y when "0001",
pEna when "0010",
not pEna when "0011",
X when "0100",
Y when others;

```


end NDV;

```

-- *****
-- **** STUDENT: 64200112
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in    std_logic;  -- nain delovanja ( '0' => aritmetini, '1' => logini )
          F      : in    std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
          X, Y    : in    std_logic_vector( n-1 downto 0 );
          S      : out   std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out   std_logic );
end alu_cla;

architecture NDV of alu_cla is

    COMPONENT cla_add_n_bit IS
        generic( n: natural := 8 );
        PORT (
            Cin      : in    std_logic ;
            X, Y      : in    std_logic_vector( n-1 downto 0 );
            S          : out   std_logic_vector( n-1 downto 0 );
            Gout, Pout, Cout : out   std_logic );
    END COMPONENT;

    signal      nAddSub : std_logic;
    signal      Y_sig, S_sig, S_out : std_logic_vector( n-1 downto 0 );
    signal      alu_operation : std_logic_vector( 3 downto 0 );
    constant    ZERO_CONST : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    constant    one : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' );

begin

    alu_operation<= M & F;

    nAddSub      <= alu_operation( 0 );

```

```

U1: cla_add_n_bit GENERIC MAP( n => n )
    Port map( Cin => nAddSub,      X => X,      Y => Y_sig,      S => S_sig,      Gout
=> Gout,      Pout => Pout,      Cout => Cout );

WITH alu_operation SELECT
    Y_sig <= not Y      when "0001",      one      when "0010",      not one      when "0011",
    X      when "0100",      Y      when others;

WITH alu_operation SELECT
    S_out <= S_sig      when "0000",      S_sig      when "0001",      S_sig
    when "0010",      S_sig      when "0011",      S_sig      when "0100",      not
ZERO_CONST when "0101",      X and Y      when "1000",      x nand Y      when "1001",
    X or Y      when "1010",      X nor Y      when "1011",      X xor Y
    when "1100",      X xnor Y      when "1101",      X      when "1110",      Y
    when "1111",      ZERO_CONST      when others;

S      <= S_out;

Negative      <= S_out( n-1 );

Overflow      <= ( not ( X( n-1 ) ) and ( not ( Y_sig( n-1 ) ) ) and S_out( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and
( not ( S_out( n-1 ) ) ) );

Zero      <= '1' WHEN ( S_out=ZERO_CONST ) ELSE '0';

end NDV;

```

```

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

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M: in std_logic;
          F: in std_logic_vector( 2 downto 0 );
          X, Y: in std_logic_vector( n-1 downto 0 );
          S: out std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout: out std_logic );
end alu_cla;

architecture NDV of alu_cla is
    component cla_add_n_bit is
        generic( n: natural := 8 );
        port( Cin : in std_logic ;
              X, Y: in std_logic_vector( n-1 downto 0 );
              S: out std_logic_vector( n-1 downto 0 );
              Gout, Pout, Cout: out std_logic
            );
    end component;

    signal nAddSub: std_logic;
    signal Y_sig: std_logic_vector( n-1 downto 0 );
    signal S_sig, Sout: std_logic_vector( n-1 downto 0 );
    constant zero_vector: std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    signal alu_operation: std_logic_vector( 3 downto 0 );
    constant one : std_logic_vector( n-1 downto 0 ) := ( 0=>'1', others=>'0' );

begin
    U1: cla_add_n_bit
        generic map ( n => n )
        port map( Cin => nAddSub, X => X, Y => Y_sig, S => S_sig, Gout => Gout,
                  Pout => Pout, Cout => Cout

```

```

    );

alu_operation<= M & F;
nAddSub      <= alu_operation( 0 );

    with alu_operation select Sout  <= S_sig when "0000",          S_sig when "0001",          S_sig when
"0010",          S_sig when "0011",          S_sig when "0100",          not zero_vector when "0101",
    X and Y when "1000",          X nand Y when "1001",          X or Y when "1010",          X nor Y when
"1011",          X xor Y when "1100",          X xnor Y when "1101",          X when "1110",          Y
when "1111",          zero_vector when others;

    with alu_operation select Y_sig <= Y when "0000",          not Y when "0001",          one when "0010",
    not one when "0011",          X when "0100",          Y when others;

Zero  <= '1' when Sout = zero_vector else '0';
Negative  <= Sout( n-1 );
Overflow  <= ( not X( n-1 ) and not Y_sig( n-1 ) and Sout( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and not
Sout( n-1 ) );
S      <= Sout;
end NDV;

```

```

-- *****
-- **** STUDENT: 64200238
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

```

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

```

```

entity alu_cla is
  generic(
    n : natural := 8
  );
  port(
    M : in std_logic;
    F : in std_logic_vector( 2 downto 0 );
    X, Y : in std_logic_vector( n-1 downto 0 );
    S : out std_logic_vector( n-1 downto 0 );
    Negative : out std_logic;
    Cout : out std_logic;
    Overflow : out std_logic;
    Zero : out std_logic;
    Gout : out std_logic;
    Pout : out std_logic
  );
end alu_cla;

```

```

architecture NDV of alu_cla is

```

```

  COMPONENT cla_add_n_bit
    generic( n: natural := 8 );
  PORT (
    Cin : IN std_logic;      -- Vhodni prenos
    X : IN std_logic_vector( n-1 downto 0 );  -- Prvi operacijski vhod
    Y : IN std_logic_vector( n-1 downto 0 );  -- Drugi operacijski vhod
    S : OUT std_logic_vector( n-1 downto 0 );  -- Rezultat operacije
    Gout : OUT std_logic;    -- Generacijski prenos
    Pout : OUT std_logic;    -- Propagacijski prenos
    Cout : OUT std_logic     -- Koncni prenos
  );

```

```

END COMPONENT;

signal      Y_sig : std_logic_vector( n-1 downto 0 );
signal      nAddSub : std_logic;
constant    all_zeros : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

    signal    Gout_S, Pout_S : std_logic;
    signal    S_temp : std_logic_vector( n-1 downto 0 );
    constant  one : std_logic_vector( n-1 downto 0 ) := ( 0=>'1', others=>'0' );
    signal    alu_operation : std_logic_vector( 3 downto 0 );
begin

U1: cla_add_n_bit
GENERIC MAP ( n => n )
PORT MAP (
Cin => nAddSub,
X => X,
Y => Y_sig,
S => S_temp,
Gout => Gout,
Pout => Pout,
Cout => Cout
);

    alu_operation<= M & F;

nAddSub    <= alu_operation( 0 );

    with alu_operation select
    S      <=
S_temp when "0000",      -- X + Y
S_temp when "0001",      -- X - Y
S_temp when "0010",      -- X + 1
S_temp when "0011",      -- X - 1
S_temp when "0100",      -- X + X
not all_zeros when "0101", -- -1
all_zeros when "0110",    -- Not in use
all_zeros when "0111",    -- Not in use
X and Y when "1000",      -- X and Y
X nand Y when "1001",     -- X nand Y

```

```

X or Y when "1010",      -- X or Y
X nor Y when "1011",     -- X nor Y
X xor Y when "1100",     -- X xor Y
X xnor Y when "1101",    -- X xnor Y
X when "1110",           -- X
Y when "1111",           -- Y
all_zeros when others;   -- Undefined

    with alu_operation select
    Y_sig <=
"0011",          Y when "0000",          not Y when "0001",          one when "0010",          not one when
                X when "0100",          Y when others;

    Overflow    <= ( not X( n-1 ) and not Y_sig( n-1 ) and S_temp( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and not S_temp(
n-1 ) );

with alu_operation select
Negative    <= '1' when "0101", -- When alu_operation is "0101", set Negative to '1'
X( n-1 ) when "1110",          -- When alu_operation is "1110", set Negative to X( n-1 )
Y( n-1 ) when "1111",          -- When alu_operation is "1111", set Negative to Y( n-1 )
S_temp( n-1 ) when others;     -- For all other cases, set Negative to S_temp( n-1 )

process( alu_operation, X, Y, S_temp )
begin
    if alu_operation = "0101" then
        Zero <= '0';
    elsif alu_operation = "1110" then
        if X = all_zeros then
            Zero <= '1';
        else
            Zero <= '0';
        end if;
    elsif alu_operation = "1111" then
        if Y = all_zeros then
            Zero <= '1';
        else
            Zero <= '0';
        end if;
    elsif S_temp = all_zeros then
        Zero <= '1';
    end if;
end process;

```



```
else  
Zero <= '0';  
end if;  
  
end process;  
  
end NDV;
```

```

-- *****
-- **** STUDENT: 64200288
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
  generic( n: natural := 8 );
  port(
    M : in std_logic;
        F : in std_logic_vector( 2 downto 0 );
        X, Y : in std_logic_vector( n-1 downto 0 );
        S : out std_logic_vector( n-1 downto 0 );
        Negative : out std_logic;
        Cout : out std_logic;
        Overflow : out std_logic;
        Zero : out std_logic;
        Gout : out std_logic;
        Pout : out std_logic
  );
end alu_cla;

architecture NDV of alu_cla is

  COMPONENT cla_add_n_bit
    generic( n: natural := 8 );
  PORT (
    Cin : IN std_logic;      -- Vhodni prenos
    X : IN std_logic_vector( n-1 downto 0 );  -- Prvi operacijski vhod
    Y : IN std_logic_vector( n-1 downto 0 );  -- Drugi operacijski vhod
    S : OUT std_logic_vector( n-1 downto 0 );  -- Rezultat operacije
    Gout : OUT std_logic;    -- Generacijski prenos
    Pout : OUT std_logic;    -- Propagacijski prenos
    Cout : OUT std_logic     -- Koncni prenos
  );
END COMPONENT;

  signal      Y_sig : std_logic_vector( n-1 downto 0 );

```

```

signal      nAddSub : std_logic;
constant    all_zeros : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

      signal      Gout_S, Pout_S : std_logic;
      signal      S_temporary : std_logic_vector( n-1 downto 0 );
      constant    one : std_logic_vector( n-1 downto 0 ) := ( 0=>'1', others=>'0' );
      signal      alu_operation : std_logic_vector( 3 downto 0 );
begin

U1: cla_add_n_bit
  GENERIC MAP ( n => n )
  PORT MAP (
    Cin => nAddSub,
    X => X,
    Y => Y_sig,
    S => S_temporary,
    Gout => Gout,
    Pout => Pout,
    Cout => Cout
  );

      alu_operation<= M & F;

nAddSub      <= alu_operation( 0 );

  with alu_operation select
    S      <=
      S_temporary when "0000", -- X + Y
      S_temporary when "0001", -- X - Y
      S_temporary when "0010", -- X + 1
      S_temporary when "0011", -- X - 1
      S_temporary when "0100", -- X + X
      not all_zeros when "0101", -- -1
      all_zeros when "0110", -- Not in use
      all_zeros when "0111", -- Not in use
      X and Y when "1000", -- X and Y
      X nand Y when "1001", -- X nand Y
      X or Y when "1010", -- X or Y
      X nor Y when "1011", -- X nor Y
      X xor Y when "1100", -- X xor Y

```

```

        X xnor Y when "1101",      -- X xnor Y
        X when "1110",           -- X
        Y when "1111",           -- Y
        all_zeros when others;    -- Undefined

with alu_operation select
Y_sig <=
        Y when "0000",      not Y when "0001",      one when "0010",      not one when "0011",
        X when "0100",      Y when others;

Overflow <= ( not X( n-1 ) and not Y_sig( n-1 ) and S_temporary( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and not
S_temporary( n-1 ) );

with alu_operation select
Negative <=
        '1' when "0101", -- When alu_operation is "0101", set Negative to '1'
X( n-1 ) when "1110", -- When alu_operation is "1110", set Negative to X( n-1 )
Y( n-1 ) when "1111", -- When alu_operation is "1111", set Negative to Y( n-1 )
S_temporary( n-1 ) when others; -- For all other cases, set Negative to S_temporary( n-1 )

process( alu_operation, X, Y, S_temporary )
begin
    if alu_operation = "0101" then
        Zero <= '0';
    elsif alu_operation = "1110" then
        if X = all_zeros then
            Zero <= '1';
        else
            Zero <= '0';
        end if;
    elsif alu_operation = "1111" then
        if Y = all_zeros then
            Zero <= '1';
        else
            Zero <= '0';
        end if;
    elsif S_temporary = all_zeros then
        Zero <= '1';
    else
        Zero <= '0';
    end if;
end process;

```

```
end if;
```

```
end process;
```

```
end NDV;
```

```

-- *****
-- **** STUDENT: 64200296
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port(
        M      : in std_logic;  -- nain delovanja ( '0' => aritmetini, '1' => logini )
        F      : in std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
        X, Y    : in std_logic_vector( n-1 downto 0 );
        S      : out std_logic_vector( n-1 downto 0 );
        Negative, Cout, Overflow, Zero, Gout, Pout : out std_logic );
end alu_cla;

architecture NDV of alu_cla is
    component cla_add_n_bit IS
        generic( n: natural );
        port( Cin      : in std_logic;  -- nain delovanja ( '0' => aritmetini, '1' => logini )
              X, Y      : in std_logic_vector( n-1 downto 0 );
              S          : out std_logic_vector( n-1 downto 0 );
              Cout, Gout, Pout : out std_logic );
    end component cla_add_n_bit;

    signal naddsub : std_logic;
    signal alu_operation : std_logic_vector( 3 downto 0 );
    signal sig_x, sig_y, sig_s, out_s : std_logic_vector( n-1 downto 0 );
    constant vectorzero : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    constant one : std_logic_vector( n-1 downto 0 ) := ( 0=>'1', others => '0' );
    constant minus : std_logic_vector( n-1 downto 0 ) := ( others => '1' );
begin
    U1: cla_add_n_bit
        generic map ( n=>n )
        port map ( cin=>naddsub, x=>sig_x, y=>sig_y, s=>sig_s, Cout=>Cout, Gout=>Gout, Pout=>Pout );

    alu_operation<= M&F;
    naddsub      <=alu_operation( 0 );

```

```

negative    <=out_s( n-1 );
zero    <='1' when out_s=vectorzero else '0';
sig_x    <=x;
Overflow    <=( not sig_x( n-1 ) and not sig_y( n-1 ) and sig_s( n-1 ) ) or ( sig_x( n-1 ) and sig_y( n-1 )
and not sig_s( n-1 ) );

with alu_operation select out_s <= sig_s when "0000",          sig_s when "0001",          sig_s when "0010",
sig_s when "0011",          sig_s when "0100",          minus when "0101",          x and y when "1000",
x nand y when "1001",          x or y when "1010",          x nor y when "1011",          x xor y when
"1100",          x xnor y when "1101",          x when "1110",          y when "1111",          vectorzero
when others;

with alu_operation select
sig_y <= y when "0000",          not y when "0001",          one when "0010",          not one when "0011",
x when "0100",          y when others;

s    <=out_s;
end NDV;

```

```

-- *****
-- **** STUDENT: 64200385
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Gout je napačen za operacijo seštevanja (ko je rezultat 0), kar je posledica napačne izvedbe na CLA
seštevalniku.
-- *****

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in    std_logic;
          F      : in    std_logic_vector( 2 downto 0 );
          X, Y   : in    std_logic_vector( n-1 downto 0 );
          S      : out   std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out   std_logic );
end alu_cla;

architecture NDV of alu_cla is

    component cla_add_n_bit
    generic ( N: natural := 8 );
    port( Cin      : in    std_logic ;
          X, Y     : in    std_logic_vector( n-1 downto 0 );
          S        : out   std_logic_vector( n-1 downto 0 );
          Gout, Pout, Cout : out   std_logic );
    end component;

    signal      nAddSub : std_logic;
    signal      Yint : std_logic_vector( n-1 downto 0 );
    signal      Sint, Sizh : std_logic_vector( n-1 downto 0 );
    signal      alu_operation : std_logic_vector( 3 downto 0 );
    constant    z : std_logic_vector( n-1 downto 0 ) := ( others=>'0' );
    constant    one : std_logic_vector( n-1 downto 0 ) := ( 0=>'1', others=>'0' );

begin
    U1: cla_add_n_bit generic map ( n => n ) port map(
        X => X,          Y => Yint,          S => Sint,          Gout => Gout,          Pout => Pout,
        Cout => Cout,    Cin => nAddSub
    );

```



```

    );
alu_operation    <= M & F;
nAddSub         <= alu_operation( 0 );
S              <= Sizh;

Negative        <= Sizh( Sizh'left );
Zero           <= '1' when ( Sizh = z ) else '0';
Overflow        <= ( not X( X'left ) and not Yint( Yint'left ) and Sizh( Sizh'left ) ) or ( X( X'left ) and Yint(
Yint'left ) and not Sizh( Sizh'left ) );

    with alu_operation select Sizh <=
        Sint          when "0000",      Sint          when "0001",      Sint          when "0010",
        Sint          when "0011",      Sint          when "0100",      not z when "0101",      X and Y when
"1000",      X nand Y when "1001",      X or Y when "1010",      X nor Y when "1011",
        X xor Y       when "1100",      X xnor Y when "1101",      X              when "1110",      Y
        when "1111",      z              when others;

    with alu_operation select Yint <=
        Y when "0000",      not Y when "0001",      one when "0010",      not one when
"0011",      X when "0100",      Y when others;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210113
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in      std_logic;  -- naèin delovanja ( '0' => aritmetièni, '1' => logièni )
          F      : in      std_logic_vector( 2 downto 0 );  -- funkcijski vhod za operacije
          X, Y    : in      std_logic_vector( n-1 downto 0 );
          S      : out     std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out     std_logic );
end alu_cla;

architecture NDV of alu_cla is
    component cla_add_n_bit is
        generic( n: natural := 8 );
        port ( Cin      : in      std_logic ;
              X, Y      : in      std_logic_vector( n-1 downto 0 );
              S          : out     std_logic_vector( n-1 downto 0 );
              Gout, Pout, Cout : out     std_logic );
    end component;

    signal S_sig, Y_sig: std_logic_vector( n-1 downto 0 );
    signal n_add_sub: std_logic;
    signal alu_operation: std_logic_vector( 3 downto 0 );
    constant allZero: std_logic_vector( n-1 downto 0 ):= ( others => '0' );
    constant one: std_logic_vector( n-1 downto 0 ):= ( 0 => '1', others => '0' );

begin
    U1: cla_add_n_bit
        generic map ( n => n )
        port map (
            Cin => n_add_sub,    X => X,        Y => Y_sig,    S => S_sig,    Gout => Gout,        Pout => Pout,        Cout
=> Cout );

```

```

alu_operation<= M & F;
n_add_sub    <= alu_operation( 0 );

Overflow      <= ( not X( n-1 ) and not Y_sig( n-1 ) and S_sig( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and not S_sig(
n-1 ) );

with alu_operation select
S      <=      S_sig      when "0000", S_sig      when "0001", S_sig      when "0010", S_sig
      when "0011", S_sig      when "0100", not allZero      when "0101", X and Y      when
"1000",      X nand Y      when "1001", X or Y      when "1010", X nor Y      when "1011", X xor
Y      when "1100", X xnor Y      when "1101", X      when "1110", Y      when
"1111",      allZero      when others;

with alu_operation select
Y_sig <=      Y      when "0000",      not Y      when "0001",      one      when "0010",      not one
      when "0011",      X      when "0100",      Y      when others;

process( alu_operation, S_sig, X, Y )
begin
    if alu_operation = "0101" then
        Zero <= '0';
    elsif alu_operation = "1110" then
        if X = allZero then
            Zero <= '1';
        else
            Zero <= '0';
        end if;
    elsif alu_operation = "1111" then
        if Y = allZero then
            Zero <= '1';
        else
            Zero <= '0';
        end if;
    elsif S_sig = allZero then
        Zero <= '1';
    else
        Zero <= '0';
    end if;
end process;

```

```
with alu_operation select
  Negative      <='1'      when "0101",      X( n-1 )      when "1110",      Y( n-1 )      when "1111",
  S_sig( n-1 ) when others;

end NDV;
```

```

-- *****
-- **** STUDENT: 64210290
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek:
Bit N ne deluje pravilno za zmanjševanje (X-1). Primer -128 -1 ni enako -127, ampak 127.
Takrat se postavi tudi bit preliva V in izhodni prenos C.
Bit Z ne deluje pravilno za seštevanje. Primer 14+(-14)=0.
Bit Z se postavi, ko je rezultat odštevanja NI enak 0 (14-15=-1, postavi se vam Z=1).
Bit V ni realiziran za odštevanje, povečevanje, zmanjševanje in podvojevanje ampak samo za seštevanje.
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in      std_logic;  -- način delovanja ( '0' => aritmetični, '1' => logični )
          F      : in      std_logic_vector( 2 downto 0 );  -- funkcijski vhod za operacije
          X, Y    : in      std_logic_vector( n-1 downto 0 );
          S      : out     std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out  std_logic );
end alu_cla;

architecture NDV of alu_cla is

    COMPONENT cla_add_n_bit IS
        GENERIC(
            n: natural := 8 );
        PORT( Cin      : in      std_logic ;
              X, Y      : in      std_logic_vector( n-1 downto 0 );
              S          : out     std_logic_vector( n-1 downto 0 );
              Gout, Pout, Cout : out  std_logic );
    END COMPONENT;

    CONSTANT zeros : std_logic_vector( n-1 downto 0 ) := ( others=>'1' );
    CONSTANT one   : std_logic_vector( n-1 downto 0 ) := ( 0=>'1', others=>'0' );

    SIGNAL alu_operation : std_logic_vector( 3 downto 0 );
    SIGNAL nAddSub        : std_logic;
    SIGNAL S_sig, Y_sig   : std_logic_vector( n-1 downto 0 );

```

```
begin
```

```
U1: cla_add_n_bit  
    generic map( n => n )  
    port map(      Cin => nAddSub,      X => X,      Y => Y_sig, S => S_sig, Gout => Gout,      Pout => Pout,  
              Cout => Cout  
    );
```

```
alu_operation<= M & F;  
nAddSub      <= alu_operation( 0 );
```

```
with alu_operation( 3 downto 0 ) select  
    Y_sig <=      Y      when "0000",  
               not Y    when "0001",  
               one      when "0010",  
               not zeros when "0011",  
               X        when "0100",  
               Y        when others;
```

```
with alu_operation select  
    S      <=      S_sig when "0000",  
    S_sig  when "0001",  
    S_sig  when "0010",  
    S_sig  when "0011",  
    S_sig  when "0100",  
    not zeros when "0101",  
    X and Y when "1000",  
    X nand Y when "1001",  
    X or Y when "1010",  
    X nor Y when "1011",  
    X xor Y when "1100",  
    X xnor Y when "1101",  
    X when "1110",  
    Y when "1111",  
    zeros when others;
```

```
with alu_operation select  
    Negative      <=      '1' when "0101",  
                       X( n-1 ) when "1110",  
                       Y( n-1 ) when "1111",
```

```

        S_sig( n-1 ) when others;

Overflow    <= ( not( X( n-1 ) ) and not( Y_sig( n-1 ) ) and S_sig( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and not(
S_sig( n-1 ) ) );

Zero    <=    '0' when alu_operation="0101" else
              '1' when ( alu_operation="1110" )and( X=zeros ) else
              '0' when ( alu_operation="1110" )and( X/=zeros ) else
              '1' when ( alu_operation="1111" )and( Y=zeros ) else
              '0' when ( alu_operation="1111" )and( Y/=zeros ) else
              '1' when S_sig=zeros else '0';

end NDV;

```

```

-- *****
-- **** STUDENT: 64210382
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in      std_logic;  -- tipi delovanja ( '0' => aritmetični, '1' => logični )
          F      : in      std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
          X, Y    : in      std_logic_vector( n-1 downto 0 );
          S      : out     std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out     std_logic );
end alu_cla;

architecture NDV of alu_cla is
    COMPONENT cla_add_n_bit IS
        generic( n: natural := 8 );
        PORT (
            Cin      : in      std_logic ;
            X, Y      : in      std_logic_vector( n-1 downto 0 );
            S          : out     std_logic_vector( n-1 downto 0 );
            Gout, Pout, Cout : out     std_logic );
    END COMPONENT;

    signal S_sig, Y_sig : std_logic_vector( n-1 downto 0 );
    signal n_add_sub : std_logic;
    signal alu_operation : std_logic_vector( 3 downto 0 );

    constant zeros : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    constant one : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' );
begin
    U1: cla_add_n_bit
        generic map ( n => n )
        port map (
            Cin => n_add_sub, X => X, Y => Y_sig, S => S_sig, Gout => Gout, Pout => Pout, Cout => Cout
        );

```



```

alu_operation<= M & F;

n_add_sub    <= alu_operation( 0 );

-- Y_sig is inverted when subtracting, so using Y_sig the overflow equation is the same for both operations.
Overflow    <= ( not X( n-1 ) and not Y_sig( n-1 ) and S_sig( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and
not S_sig( n-1 ) );

with alu_operation select Negative    <=
    '1' when "0101",      X( n-1 )      when "1110",      Y( n-1 )
when "1111",      S_sig( n-1 ) when others;

Zero    <= '0' when alu_operation = "0101" else
    '1' when alu_operation = "1110" and X = zeros else
    '0' when alu_operation = "1110" and X /= zeros else
    '1' when alu_operation = "1111" and Y = zeros else
    '0' when alu_operation = "1111" and Y /= zeros else
    '1' when S_sig = zeros else '0';

with alu_operation select S    <=
    S_sig      when "0000",      S_sig      when "0001",      S_sig      when "0010",
S_sig      when "0011",      S_sig      when "0100",      not zeros  when "0101",      X and Y
when "1000",      X nand Y      when "1001",      X or Y      when "1010",      X nor Y      when "1011",
X xor Y      when "1100",      X xnor Y      when "1101",      X      when "1110",      Y
when "1111",      zeros      when others;

with alu_operation select Y_sig    <=
    Y when "0000",      not Y when "0001",      one when "0010",      not one when
"0011",      X when "0100",      Y when others;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210384
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_misc.all;      -- vporabljam za redukcijske operatore

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in      std_logic;
          F      : in      std_logic_vector( 2 downto 0 );
          X, Y   : in      std_logic_vector( n-1 downto 0 );
          S      : out     std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero,  Gout, Pout : out     std_logic );
end alu_cla;

architecture NDV of alu_cla is

    COMPONENT cla_add_n_bit
    generic( n: natural := 8 );
    PORT ( Cin   : in std_logic;      -- vhodni prenosni bit
          X, Y   : in std_logic_vector( n-1 downto 0 );
          S      : out std_logic_vector( n-1 downto 0 );
          Gout, Pout, Cout : out std_logic );
    END COMPONENT;

    -- signali za internu uporabo
    signal y_temp : std_logic_vector( n-1 downto 0 );    -- y
    signal nAddSub : std_logic;      -- + in -
    signal alu_operation : std_logic_vector( 3 downto 0 );    -- izbira operacije
    signal rezultat : std_logic_vector( n-1 downto 0 );    -- rezultat
    signal s_temp : std_logic_vector( n-1 downto 0 );    -- privremeni rezultat signala
    signal over_temp : std_logic;    -- slucajevi kada ce se desiti overflow

    -- uporaba konstante ena zaradi operacija
    constant ena : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' );
    constant nic : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

```

begin

U1: cla_add_n_bit
    generic map ( n => n )
    port map (
        Cin => nAddSub,          X => X,          Y => y_temp,          S => s_temp,
Gout => Gout,          Pout => Pout,          Cout => Cout
    );

    alu_operation<= M & F;

    nAddSub      <= alu_operation( 0 );
    with alu_operation select
        y_temp<= not y when "0001",          ena when "0010",          not ena when "0011",          x
        when "0100",          y when others;

    with alu_operation select
        rezultat      <= not nic when "0101",          nic when "0110" | "0111",          x and y when "1000",
        x nand y when "1001",          x or y when "1010",          x nor y when "1011",          x xor y when
"1100",          x xnor y when "1101",          x          when "1110",          y when "1111",
        -- y ali y_signal, vseno je
        s_temp when others;

    S      <= rezultat;

    Negative      <= rezultat( n-1 );
    -- ce so vsi biti 0 -> ( Zero = 1 )
    Zero      <= nor_reduce( rezultat );

    over_temp      <= ( ( not x( x'left ) ) and ( not y_temp( y_temp'left ) ) and rezultat( rezultat'left ) ) or
        ( x( x'left ) and y_temp( y_temp'left ) and ( not rezultat( rezultat'left ) ) ) );
    with alu_operation select
        overflow      <= '0' when "0101" | "0110" | "0111",          over_temp when others;
end NDV;

```

```

-- *****
-- **** STUDENT: 64210386
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Gout je nedefiniran za operacijo seštevanja, kar je posledica napačne izvedbe na CLA seštevalniku.
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in      std_logic;    -- nain delovanja ( '0' => aritmetini, '1' => logini )
          F      : in      std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
          X, Y    : in      std_logic_vector( n-1 downto 0 );
          S      : out     std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out     std_logic );
end alu_cla;

architecture NDV of alu_cla is

    COMPONENT cla_add_n_bit IS
        generic( n: natural := 8 );
        PORT (
            Cin      : in      std_logic ;
            X, Y      : in      std_logic_vector( n-1 downto 0 );
            S          : out     std_logic_vector( n-1 downto 0 );
            Gout, Pout, Cout : out     std_logic );
    END COMPONENT;

    signal      s_sig, y_sig : std_logic_vector( n-1 downto 0 );
    signal      nAddSub : std_logic;
    signal      alu_operation : std_logic_vector( 3 downto 0 );
    constant    nicle : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    constant    enica : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' );

begin

    U1: cla_add_n_bit generic map ( n => n )
        port map (

```

```

        Cin => nAddSub,          X => X,          Y => y_sig,          S => s_sig,          Gout => Gout,
        Pout => Pout,          Cout => Cout );

alu_operation<= M & F;
nAddSub      <= alu_operation( 0 );

Overflow     <= ( ( not X( n-1 ) ) and ( not y_sig( n-1 ) ) and s_sig( n-1 ) ) or ( X( n-1 ) and y_sig( n-1 )
and ( not s_sig( n-1 ) ) );

with alu_operation select
    Negative   <= '1' when "0101",          X( n-1 ) when "1110",          Y( n-1 ) when "1111",          S_sig( n-1 )
when others;

Zero  <= '0' when alu_operation = "0101" else
      '1' when alu_operation = "1110" and X = nicle else
      '0' when alu_operation = "1110" and X /= nicle else
      '1' when alu_operation = "1111" and Y = nicle else
      '0' when alu_operation = "1111" and Y /= nicle else
      '1' when S_sig = nicle else '0';

with alu_operation select
    S      <= s_sig when "0000",          s_sig when "0001", s_sig when "0010", s_sig when "0011", s_sig when
"0100",          not nicle when "0101",          X and Y when "1000",          X nand Y when "1001",          X or Y when "1010",
X nor Y when "1011",          X xor Y when "1100",          X xnor Y when "1101",          X when "1110",          Y when
"1111",          nicle when others;

with alu_operation select
    y_sig <= Y when "0000",          not Y when "0001", enica when "0010",          not enica when "0011",          X when
"0100",          Y when others;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210445
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Gout je nedefiniran za operacijo seštevanja, kar je posledica napačne izvedbe na CLA seštevalniku.
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in    std_logic;  -- naèin delovanja ( '0' => aritmetični, '1' => logični )
          F      : in    std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
          X, Y    : in    std_logic_vector( n-1 downto 0 );
          S      : out   std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero,  Gout, Pout : out   std_logic );
end alu_cla;

architecture NDV of alu_cla is

    COMPONENT cla_add_n_bit IS
        generic( n: natural := 8 );
        PORT (
            Cin      : in    std_logic ;
            X, Y      : in    std_logic_vector( n-1 downto 0 );
            S          : out   std_logic_vector( n-1 downto 0 );
            Gout, Pout, Cout : out   std_logic );
    END COMPONENT;

    signal      s_sig, y_sig : std_logic_vector( n-1 downto 0 );
    signal      nAddSub      : std_logic;
    signal      alu_oper     : std_logic_vector( 3 downto 0 );
    signal      y_temp       : std_logic;

    constant    nule : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    constant    ena  : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' );

begin

    U1: cla_add_n_bit
        generic map ( n => n )

```

```

port map (
    Cin => nAddSub,      X => X,      Y => Y_sig,      S => S_sig,      Gout => Gout,
    Pout => Pout,      Cout => Cout );

alu_oper    <= M & F;

nAddSub     <= alu_oper( 0 );

-- y_sig     <= y xor ( y_sig'range => '1' ) when nAddSub = '1' else y;

Overflow    <= ( ( not X( n-1 ) ) and ( not y_sig( n-1 ) ) and s_sig( n-1 ) ) or ( X( n-1 ) and y_sig( n-1 )
and ( not s_sig( n-1 ) ) );

with alu_oper select
    Negative    <= '1' when "0101",      X( n-1 ) when "1110",      Y( n-1 ) when "1111",      S_sig( n-1 )
when others;

Zero    <= '0' when alu_oper = "0101" else
    '1' when alu_oper = "1110" and X = nule else
    '0' when alu_oper = "1110" and X /= nule else
    '1' when alu_oper = "1111" and Y = nule else
    '0' when alu_oper = "1111" and Y /= nule else
    '1' when S_sig = nule else '0';

with alu_oper select
    S    <= S_sig when "0000",      S_sig when "0001", S_sig when "0010", S_sig when "0011", S_sig when
"0100",      not nule when "0101",      X and Y when "1000",      X nand Y when "1001",      X or Y when "1010",
X nor Y when "1011",      X xor Y when "1100",      X xnor Y when "1101",      X when "1110",      Y when
"1111",      nule when others;

with alu_oper select
    Y_sig <= Y when "0000",      not Y when "0001", ena when "0010",      not ena when "0011",      X when
"0100",      Y when others;

end NDV;

```

```

-- *****
-- **** STUDENT: 64210455
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Napake sintetizatorja:
ERROR:HDLCompiler:136 - "alu_cla.vhd" Line 45: OTHERS choice cannot be used in unconstrained array aggregate.
Y_mod <= ( Y xor ( others => add_sub ) );
Podobno kot ste imeli pri cla_adder.vhd - operacije xor ne morete izvesti nad neomejenim vektorjem. Definirati morate
interni signal (npr. add_sub_vector), ki ga nato povežete na vrednost (add_sub) in šele potem uporabite nad
rezultatom:
signal add_sub_vector : std_logic_vector(n-1 downto 0); -- zdaj je velikost omejena na n elementov
nato ga povežete z uporabo others:
add_sub_vector <= (others => add_sub);
in nato uporabite pri tvorbi xor:
Y_mod <= (Y xor add_sub_vector);
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
  generic( n: natural := 8 );
  port(
    M : in std_logic;
    F : in std_logic_vector( 2 downto 0 );
    X, Y : in std_logic_vector( n-1 downto 0 );
    S : out std_logic_vector( n-1 downto 0 );
    Negative, Cout, Overflow, Zero, Gout, Pout : out std_logic
  );
end alu_cla;

architecture NDV of alu_cla is
  COMPONENT cla_add_n_bit IS
    generic( n: natural := 8 );
    PORT (
      Cin : in std_logic;
      X, Y : in std_logic_vector( n-1 downto 0 );
      S : out std_logic_vector( n-1 downto 0 );
      Gout, Pout, Cout : out std_logic
    );
  END COMPONENT;

```



```

signal      S_sig : std_logic_vector( n-1 downto 0 );
signal      Y_mod : std_logic_vector( n-1 downto 0 );
signal      add_sub : std_logic;

constant    all_zeros : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

begin
  CLA_Instance: cla_add_n_bit
  generic map ( n => n )
  port map (
    Cin => add_sub,
    X => X,
    Y => Y_mod,
    S => S_sig,
    Gout => Gout,
    Pout => Pout,
    Cout => Cout
  );

  Y_mod <= ( Y xor ( others => add_sub ) );

  add_sub    <= M and F( 0 );

  S         <= S_sig;

  Negative   <= S_sig( n-1 );
  Zero       <= '1' when S_sig = all_zeros else '0';
  Overflow    <= ( not X( n-1 ) and not Y_mod( n-1 ) and S_sig( n-1 ) ) or
    ( X( n-1 ) and Y_mod( n-1 ) and not S_sig( n-1 ) );
end NDV;

```

```

-- *****
-- **** STUDENT: 64210457
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in      std_logic;  -- nain delovanja ( '0' => aritmetini, '1' => logini )
          F      : in      std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
          X, Y    : in      std_logic_vector( n-1 downto 0 );
          S      : out     std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out     std_logic );
end alu_cla;

architecture NDV of alu_cla is

    component cla_add_n_bit IS
        generic( n: natural := 8 );
        PORT (
            Cin      : in      std_logic ;
            X, Y      : in      std_logic_vector( n-1 downto 0 );
            S          : out     std_logic_vector( n-1 downto 0 );
            Gout, Pout, Cout : out     std_logic );
    END component;

    -- test signals
    signal S_sig, Y_sig : std_logic_vector( n-1 downto 0 ); -- sum and controlled complement
    signal n_add_sub : std_logic;
    signal alu_operation : std_logic_vector( 3 downto 0 );

    constant zeros : std_logic_vector( n-1 downto 0 ) := ( others => '0' ); -- set to 0
    constant one : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' ); -- set to ..001

begin
    U1: cla_add_n_bit generic map ( n => n )
        port map
        (

```

```

Cin => n_add_sub,  X => X,      Y => Y_sig,  S => S_sig,  Gout=>Gout,  Pout => Pout,      Cout => Cout
);

-- set operation mode
alu_operation<= M & F;
n_add_sub    <= alu_operation( 0 );

-- calculate overflow
Overflow    <= ( not X( n-1 ) and not Y_sig( n-1 ) and S_sig( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and not S_sig(
n-1 ) );

-- set negative bit
with alu_operation select
Negative    <=
    '1' when "0101",      X( n-1 )      when "1110",      Y( n-1 )
when "1111",      S_sig( n-1 ) when others;

Zero <= '0' when alu_operation = "0101" else
    '1' when alu_operation = "1110" and X = zeros else
    '0' when alu_operation = "1110" and X /= zeros else
    '1' when alu_operation = "1111" and Y = zeros else
    '0' when alu_operation = "1111" and Y /= zeros else
    '1' when S_sig = zeros else '0';

-- calculating the sum
with alu_operation select S    <=
    S_sig    when "0000",      S_sig    when "0001",      S_sig    when "0010",
S_sig    when "0011",      S_sig    when "0100",      not zeros when "0101",      X and Y
when "1000",      X nand Y    when "1001",      X or Y      when "1010",      X nor Y    when "1011",
X xor Y    when "1100",      X xnor Y    when "1101",      X      when "1110",      Y
when "1111",      zeros      when others;

-- calculating controlled complement of y
with alu_operation select Y_sig <=
    Y when "0000",      not Y when "0001",      one when "0010",      not one when "0011",
X when "0100",      Y when others;

end NDV;

```

```

-- *****
-- **** STUDENT: 64240429
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Pri "operaciji -1" se mora postaviti bit N=1, ker gre za negativno število na izhodu. Negative se mora
postaviti v primeru "operacije Y", ker gre za realizacijo zbirniškega mikroukaza TST.
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in      std_logic;  -- način delovanja ( '0' => aritmetični, '1' => logični )
          F      : in      std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
          X, Y    : in      std_logic_vector( n-1 downto 0 );
          S      : out     std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout : out     std_logic );
end alu_cla;

architecture NDV of alu_cla is
    component cla_add_n_bit IS
        generic( n: natural := 8 );
        PORT (
            Cin      : in      std_logic ;
            X, Y     : in      std_logic_vector( n-1 downto 0 );
            S        : out     std_logic_vector( n-1 downto 0 );
            Gout, Pout, Cout : out     std_logic );
    END component;

    signal      alu_operation : std_logic_vector( 3 downto 0 );

    constant zeros_vector : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    constant one          : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' );
    signal nAddSub : std_logic;

    signal Y_sig : std_logic_vector( n-1 downto 0 );
    signal X_sig : std_logic_vector( n-1 downto 0 );
    signal S_sig : std_logic_vector( n-1 downto 0 );

begin

```

```
U1: cla_add_n_bit generic map ( n => n )
```

```
port map (  
  Cin => nAddSub,  
  X => X_sig,  
  Y => Y_sig,  
  S => S_sig,  
  Cout => Cout,  
  Gout => Gout,  
  Pout => Pout  
);
```

```
X_sig <= X; -- Z is always X
```

```
alu_operation<= M&F;  
nAddSub      <= F( 0 );
```

```
Overflow      <= ( not X( n-1 ) and not Y_sig( n-1 ) and S_sig( n-1 ) ) or ( X( n-1 ) and Y_sig( n-1 ) and not S_sig(  
n-1 ) );  
Negative      <= S_sig( n-1 ); -- for some reason even if I say S_sig( 7 ) the Negative is different from S_sig( 7 )  
sometimes  
Zero <= '1' when S_sig = zeros_vector else '0';
```

```
with alu_operation select Y_sig <=  
  Y when "0000",  
  not Y when "0001",  
  one when "0010",  
  not one when "0011",  
  X when "0100",  
  Y when others;
```

```
with alu_operation select S      <=  
  S_sig when "0000",  
  S_sig when "0001",  
  S_sig when "0010",  
  S_sig when "0011",  
  S_sig when "0100",  
  not zeros_vector when "0101", -- NI V UPORABI when "0110", -- NI V UPORABI when "0111",  
  X_sig and Y_sig when "1000",  
  X_sig nand Y_sig when "1001",  
  X_sig or Y_sig when "1010",
```

```
X_sig nor Y_sig when "1011",  
X_sig xor Y_sig when "1100",  
X_sig xnor Y_sig when "1101",  
X_sig when "1110",  
Y_sig when "1111",  
zeros_vector when others;  
  
end NDV;
```

```

-- *****
-- **** STUDENT: 64240430
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_misc.all;      -- vporabljam za redukcijske operatore

entity alu_cla is
    generic( n: natural := 8 );
    port( M      : in    std_logic;      -- način delovanja ( '0' => aritmetični, '1' => logični )
          F      : in    std_logic_vector( 2 downto 0 ); -- funkcijski vhod za operacije
          X, Y    : in    std_logic_vector( n-1 downto 0 );
          S      : out   std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero,  Gout, Pout : out   std_logic );
end alu_cla;

-- Arhitektura za ALU
architecture NDV of alu_cla is

    COMPONENT cla_add_n_bit
    generic( n: natural := 8 );
    PORT ( Cin   : in std_logic;      -- vhodni prenosni bit
          X, Y   : in std_logic_vector( n-1 downto 0 );
          S      : out std_logic_vector( n-1 downto 0 );
          Gout, Pout, Cout : out std_logic );
    END COMPONENT;

    -- signali za internu uporabo
    signal y_temp : std_logic_vector( n-1 downto 0 ); -- spremljenivka y
    signal nAddSub : std_logic;      -- sestevanje in odštevanje
    signal alu_operation : std_logic_vector( 3 downto 0 ); -- izbira operacija
    signal rezultat : std_logic_vector( n-1 downto 0 ); -- rezultat operacija
    signal s_temp : std_logic_vector( n-1 downto 0 ); -- privremeni rezultat signala
    signal over_temp : std_logic;      -- slucajevi kada ce se desiti overflow

    -- uporaba konstante ena zaradi operacija
    constant ena : std_logic_vector( n-1 downto 0 ) := ( 0 => '1', others => '0' );

```

```

constant    nic : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

begin

U1: cla_add_n_bit
    generic map ( n => n )
    port map (
        Cin => nAddSub,          X => X,          Y => y_temp,          S => s_temp,
        Gout => Gout,          Pout => Pout,          Cout => Cout
    );

    -- kontrolna promjenjiva
    alu_operation<= M & F;
    -- LSB bit nam je pomemban za nAddSub
    nAddSub      <= alu_operation( alu_operation'right );
    -- podesavanje vrijednosti y u zavisnosti od operacije
    -- with select, zaradi lazje vporabe
    with alu_operation select
        y_temp<= not y when "0001",          ena when "0010",          not ena when "0011",          x
        when "0100",          y when others;

    -- odabir operacija uz pomoc varijable alu_op
    with alu_operation select
        rezultat      <= not nic when "0101",          nic when "0110" | "0111",          x and y when "1000",
        x nand y when "1001",          x or y when "1010",          x nor y when "1011",          x xor y when
"1100",          x xnor y when "1101",          x          when "1110",          y when "1111",
    -- y ali y_signal, vseno je
        s_temp when others;

    S      <= rezultat;

    -- Negativan ce je MSB = 1;
    Negative      <= rezultat( rezultat'left );
    -- proverava svih posameznih bita "rezultat"
    -- ce so vsi biti 0 -> ( Zero = 1 )
    Zero      <= nor_reduce( rezultat );
    -- za koje slucajeve ce se desiti overflow
    over_temp      <= ( ( not x( x'left ) ) and ( not y_temp( y_temp'left ) ) and rezultat( rezultat'left ) ) or
        ( x( x'left ) and y_temp( y_temp'left ) and ( not rezultat( rezultat'left ) ) );
    with alu_operation select

```



```
end NDV;      overflow      <= '0' when "0101" | "0110" | "0111",      over_temp when others;
```

```

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

entity alu_cla is
    generic( n: natural := 8 );
    port( M          : in    std_logic;  -- naèin delovanja ( '0' => aritmetièni, '1' => logièni )
          F          : in    std_logic_vector( 2 downto 0 );  -- funkcijski vhod za operacije
          X, Y       : in    std_logic_vector( n-1 downto 0 );
          S          : out   std_logic_vector( n-1 downto 0 );
          Negative, Cout, Overflow, Zero, Gout, Pout      :
    out   std_logic );
end alu_cla;

architecture ideal of alu_cla is
    constant zeroes      : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    constant one         : std_logic_vector( n-1 downto 0 ) := ( 0=>'1', others=>'0' );
    signal Sum_sig,Y_sig: std_logic_vector( n-1 downto 0 );
    signal nAddSub        : std_logic;
    -- operacija seštevanja ( nAddSub => '0' ) ali odštevanja ( nAddSub => '1' )
    signal alu_operation : std_logic_vector( 3 downto 0 );

    signal X_is_zero, Y_is_zero, Sum_is_zero : std_logic;

    component cla_add_n_bit IS
        generic( n: natural := 8 );
        PORT (
            Cin      : in    std_logic ;
            X, Y     : in    std_logic_vector( n-1 downto 0 );
            S        : out   std_logic_vector( n-1 downto 0 );
            Gout, Pout, Cout : out   std_logic );
    END component;

    for all:cla_add_n_bit use entity work.cla_add_n_bit( ideal );

begin

```

```
alu_operation<= M & F;    -- združimo naèin delovanja ALU in tip operacije
nAddSub      <= alu_operation( 0 );    -- LSB mesto bo doloèalo tip aritmetične operacije
```

-- Izbiralnik vhoda Y za različne operacije:

```
with alu_operation select
    Y_sig <=      Y                when x"0",    -- seštevanje
    not( Y )      when x"1",    -- odštevanje ( eniški komplement )
    one           when x"2",    -- prištevanje konstante 1
    not ( one )   when x"3",    -- odštevanje konstante 1
    X             when x"4",    -- prištevanje X + X
    Y             when others;
```

```
U1: cla_add_n_bit generic map ( n => n ) port map ( nAddSub, X, Y_sig, Sum_sig, Gout, Pout, Cout );
-- povezovanje sestevalnika
```

```
with alu_operation select
    Negative      <= '1'          when x"5",    -- postavi N bit v konstanti -1 na izhodu
    X( n-1 )      when x"E",      Y( n-1 )      when x"F",
Sum_sig( n-1 )    when others; -- bit predznaka
```

-- bit preliva se postavi v primerih (odvisno od operacije):

Operacija:	X(n-1)	Y(n-1)	S(n-1)
Add (X+Y)	0 (pozitiven)	0 (pozitiven)	1 (negativen)
Add (X+Y)	1 (negativen)	1 (negativen)	0 (pozitiven)
Sub (X-Y)	0 (pozitiven)	1 (negativen)	1 (negativen)
Sub (X-Y)	1 (negativen)	0 (pozitiven)	0 (pozitiven)

```
with alu_operation select
    Overflow      <= ( ( not X( n-1 ) and not Y( n-1 ) and Sum_sig( n-1 ) ) or ( X( n-1 ) and Y(
n-1 ) and not Sum_sig( n-1 ) ) ) when x"0",    -- seštevanje ( drugi operand je Y )
    ( ( not X( n-1 ) and Y( n-1 ) and Sum_sig( n-1 ) ) or ( X( n-1 ) and not Y( n-1 ) and not
Sum_sig( n-1 ) ) ) when x"1",    -- odštevanje ( drugi operand je Y );
    ( not X( n-1 ) and Sum_sig( n-1 ) ) when x"2",    -- X plus 1
( drugi operand je 1, ki ima MSB vedno 0, zato ostane samo en clen )
    ( X( n-1 ) and not Sum_sig( n-1 ) ) when x"3",    -- X minus 1
( drugi operand je 1, ki ima MSB vedno 0, zato ostane samo en clen )
    ( ( not X( n-1 ) and Sum_sig( n-1 ) ) or ( X( n-1 ) and not
Sum_sig( n-1 ) ) ) when x"4",    -- X plus X ( drugi operand je X, zato ostane samo en clen )
    '0' when others;
```

```

Sum_is_zero  <= '1' when ( Sum_sig = zeroes ) else '0';
X_is_zero    <= '1' when ( X = zeroes ) else '0';
Y_is_zero    <= '1' when ( Y = zeroes ) else '0';

with alu_operation select
    Zero    <= '0' when x"5", X_is_zero when x"E", Y_is_zero when
x"F", Sum_is_zero when others; -- bit enako nic

-- Izhodni izbiralnik ALU realizira operacije:
-- M F      operacija
-- 0 0 0 0 S = X plus Y
-- 0 0 0 1 S = X minus Y
-- 0 0 1 0 S = X plus 1
-- 0 0 1 1 S = X minus 1
-- 0 1 0 0 S = X plus X
-- 0 1 0 1 S = minus 1 ( dvojiški komplement )
-- 1 0 0 0 S = X and Y
-- 1 0 0 1 S = X nand Y
-- 1 0 1 0 S = X or Y
-- 1 0 1 1 S = X nor Y
-- 1 1 0 0 S = X xor Y
-- 1 1 0 1 S = X xnor Y
-- 1 1 1 0 S = X
-- 1 1 1 1 S = Y

with alu_operation select
    S    <= Sum_sig
when x"5", X and Y
    when x"A", X nor Y
X xnor Y
zeroes
    when x"D", X
    when others;

when x"0" | x"1" | x"2" | x"3" | x"4", not ( zeroes )
when x"8", X nand Y
when x"B", X xor Y
when x"E", Y
when x"9", X or Y
when x"C",
when x"F",

end ideal;

```