

Ocenjevanje datoteke testnih vrednosti ALU za seštevanje in odštevanje

| | |
|--|----|
| Ocenjevanje datoteke testnih vrednosti ALU za seštevanje in odštevanje | 1 |
| -- **** STUDENT: 64000225 | 4 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). | 4 |
| -- **** STUDENT: 64200100 | 11 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). | 11 |
| -- **** STUDENT: 64200163 | 13 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek Manjka assert. | 13 |
| Zakasnitev CLA ni 10 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). | 13 |
| -- **** STUDENT: 64200238 | 15 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). | 15 |
| Solze se bodo posušile – z leti ni nič drugače, samo človek se navadi. Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). | 15 |
| -- **** STUDENT: 64200288 | 19 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). | 19 |
| -- **** STUDENT: 64200296 | 23 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). | 23 |
| Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). | 23 |
| -- **** STUDENT: 64200385 | 29 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). | 29 |
| -- **** STUDENT: 64210113 | 32 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). | 32 |
| -- **** STUDENT: 64210290 | 35 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Manjka assert. | 35 |

| | |
|--|----|
| Zakasnitev CLA ni clk_period (10 ns), ampak ca. 11 ns (glej vrednost combinatorial path delay)..... | 35 |
| -- **** STUDENT: 64210382..... | 38 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). 38 | |
| -- **** STUDENT: 64210384..... | 41 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). 41 | |
| Zakasnitev CLA ni 5 ns (2*5/2), ampak ca. 11 ns (glej vrednost combinatorial path delay)..... | 41 |
| -- **** STUDENT: 64210386..... | 44 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). 44 | |
| Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay)..... | 44 |
| -- **** STUDENT: 64210445..... | 47 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). 47 | |
| Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay)..... | 47 |
| -- **** STUDENT: 64210455..... | 50 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay). 50 | |
| -- **** STUDENT: 64210457..... | 54 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). 54 | |
| Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay)..... | 54 |
| -- **** STUDENT: 64240429..... | 57 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Manjka assert. Izbrani so samo točno določeni primeri iz testiranja CLA. Ideja je bila, da testirate na celotni obseg števil x in y (0...2**n-1)..... | 57 |
| -- **** STUDENT: 64240430..... | 63 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). 63 | |
| Zakasnitev CLA ni 5 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay)..... | 63 |
| -- **** STUDENT: 64210132..... | 66 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). 66 | |

| | |
|---|----|
| Ideja datoteke testnih vrednosti je, da preleti celoten obseg števil x in y ter na drugačen način (torej z VHDL + operatorjem) preveri, če so razlike v izračunu strojne komponente in simulatorja. Koda datoteke testnih vrednosti je ista kot pri testiranju CLA seštevalnika, kar ni poanta naloge – testirati je treba operaciji seštevanja in odštevanja preko celotnega obsega operandov X in Y. | 66 |
| ***** NASLEDNJIČ KODO NALOŽITE V USTREZEN RAZDELEK (OSTALO_x), KJER JE X ŠTEVILKA DOMAČE NALOGE ***** | 66 |

```

-- *****
-- **** STUDENT: 64000225
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****

```

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

```

```

ENTITY alu_tb IS
    GENERIC( n: natural := 8 );
END alu_tb;

```

```

ARCHITECTURE behavior OF alu_tb IS

```

```

    COMPONENT alu_cla
        GENERIC( n: natural := 8 );
    PORT(
        M : IN std_logic;
        F : IN std_logic_vector( 2 downto 0 );
        X : IN std_logic_vector( n-1 downto 0 );
        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 COMPONENT;

```

```

    -- Inputs

```

```

    signal M : std_logic := '0';
    signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
    signal X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    signal Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

```

-- Outputs
signal      S : std_logic_vector( n-1 downto 0 );
signal      Negative : std_logic;
signal      Cout : std_logic;
signal      Overflow : std_logic;
signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;

--      --      --      --      --      --      --      --      --      --      --      --      --      --      --
--      --      --      --      --      --      --      --      --      --      --      --      --      --      --
signal      X_num, Y_num : integer := 0;
signal      S_test : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
signal      X_and_Y, X_nand_Y, X_or_Y, X_nor_Y,          X_xor_Y, X_xnor_Y : std_logic_vector( n-1 downto 0
) := ( others => '0' );
signal      Neg_test, Over_test, Zero_test : std_logic := '0';

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

BEGIN

    -- Instantiate the Unit Under Test ( UUT )
    uut: alu_cla generic map ( n => n ) PORT MAP (
    M => M,
    F => F,
    X => X,
    Y => Y,
    S => S,
    Negative => Negative,
    Cout => Cout,
    Overflow => Overflow,
    Zero => Zero,
    Gout => Gout,
    Pout => Pout
    );

    -- Stimulus process
    stim_proc: process
    begin
        -- hold reset state for 100 ns.

```

```

wait for 100 ns;
for i in -( 2**( n-1 ) ) to 2**( n-1 ) - 1 loop
    X_num <= i;
    X      <= std_logic_vector( to_signed( i, X'length ) );
    for j in -( 2**( n-1 ) ) to 2**( n-1 ) - 1 loop
        Y_num <= j;
        Y      <= std_logic_vector( to_signed( j, Y'length ) );

        M      <= '0';      -- Arithmetic
        F      <= "000";    -- X + Y
        wait for 1 ns;
        assert( S_test = S ) report "Sum failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;
        assert( Neg_test = Negative ) report "Sum N failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Over_test = Overflow ) report "Sum V failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Zero_test = Zero ) report "Sum Z failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) severity error;

        F      <= "001";    -- X - Y
        wait for 1 ns;
        assert( S_test = S ) report "Dif failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;
        assert( Neg_test = Negative ) report "Dif N failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Over_test = Overflow ) report "Dif V failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Zero_test = Zero ) report "Dif Z failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) severity error;

        F      <= "010";    -- X + 1
        wait for 1 ns;
        assert( S_test = S ) report "X+1 failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;
        assert( Neg_test = Negative ) report "X+1 N failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Over_test = Overflow ) report "X+1 V failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;

```

```

        assert( Zero_test = Zero ) report "X+1 Z failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) severity error;

        F      <= "011";    -- X - 1
        wait for 1 ns;
        assert( S_test = S ) report "X-1 failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;
        assert( Neg_test = Negative ) report "X-1 N failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Over_test = Overflow ) report "X-1 V failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Zero_test = Zero ) report "X-1 Z failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) severity error;

        F      <= "100";    -- X + X
        wait for 1 ns;
        assert( S_test = S ) report "X+X failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;
        assert( Neg_test = Negative ) report "X+X N failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Over_test = Overflow ) report "X+X V failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Zero_test = Zero ) report "X+X Z failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) severity error;

        F      <= "101";    -- -1
        wait for 1 ns;
        assert( S_test = S ) report "-1 failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;
        assert( Neg_test = Negative ) report "-1 N failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Zero_test = Zero ) report "-1 Z failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) severity error;

        F      <= "110";    -- undefined operation
        wait for 1 ns;
        assert( zeros_0 = S ) report "0110 failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;

        F      <= "111";    -- undefined operation

```

```

wait for 1 ns;
assert( zeros_0 = S ) report "0111 failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;

M      <= '1';      -- Logic
F      <= "000";    -- X and Y
wait for 1 ns;
assert( X_and_Y = S ) report "X and Y failed. X:" & integer'image( i ) & " Y:" & integer'image( j
) severity error;

F      <= "001";    -- X nand Y
wait for 1 ns;
assert( X_nand_Y = S ) report "X nand Y failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) severity error;

F      <= "010";    -- X or Y
wait for 1 ns;
assert( X_or_Y = S ) report "X or Y failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;

F      <= "011";    -- X nor Y
wait for 1 ns;
assert( X_nor_Y = S ) report "X nor Y failed. X:" & integer'image( i ) & " Y:" & integer'image( j
) severity error;

F      <= "100";    -- X xor Y
wait for 1 ns;
assert( X_xor_Y = S ) report "X xor Y failed. X:" & integer'image( i ) & " Y:" & integer'image( j
) severity error;

F      <= "101";    -- X xnor Y
wait for 1 ns;
assert( X_xnor_Y = S ) report "X xnor Y failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) severity error;

F      <= "110";    -- X
wait for 1 ns;
assert( X = S ) report "X failed. X:" & integer'image( i ) & " Y:" & integer'image( j ) severity
error;

```



```

        assert( Neg_test = Negative ) report "X N failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Zero_test = Zero ) report "X Z failed. X:" & integer'image( i ) & " Y:" & integer'image( j
) severity error;

        F      <= "111";    -- Y
        wait for 1 ns;
        assert( Y = S ) report "Y failed. X:" & integer'image( i ) & " Y:" & integer'image( j ) severity
error;
        assert( Neg_test = Negative ) report "Y N failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) severity error;
        assert( Zero_test = Zero ) report "Y Z failed. X:" & integer'image( i ) & " Y:" & integer'image( j
) severity error;
        end loop;
    end loop;
wait;
end process;

```

-- Calculate sum for checking the adder result

```

test_proc: process ( X_num, Y_num, F )
variable temp_S, temp_X, temp_Y : std_logic_vector( n-1 downto 0 );
variable temp_R : integer;
begin
    temp_X := std_logic_vector( to_signed( X_num, temp_X'length ) );
    temp_Y := std_logic_vector( to_signed( Y_num, temp_Y'length ) );

    if F = "101" then
        temp_S := not zeros_0;
        Over_test <= '0';
    else
        if F = "000" then
            temp_R := X_num + Y_num;
        elsif F = "001" then
            temp_R := X_num - Y_num;
        elsif F = "010" then
            temp_R := X_num + 1;
        elsif F = "011" then
            temp_R := X_num - 1;
        elsif F = "100" then
            temp_R := X_num + X_num;
        end if;
    end if;
end test_proc;

```

```

    elsif F = "110" then
        temp_R := X_num;
    elsif F = "111" then
        temp_R := Y_num;
    else
        temp_R := 0;
    end if;

    if temp_R > 2**( n-1 ) - 1 or temp_R < -( 2**( n-1 ) ) then
        Over_test    <= '1';
    else
        Over_test    <= '0';
    end if;
    temp_S := std_logic_vector( to_unsigned( temp_R mod 2**n, temp_S'length ) );
end if;

S_test <= temp_S;
Neg_test    <= temp_S( n-1 );

if temp_S = zeros_0 then
    Zero_test    <= '1';
else
    Zero_test    <= '0';
end if;

X_and_Y      <= temp_X and temp_Y;
X_nand_Y     <= temp_X nand temp_Y;
X_or_Y       <= temp_X or temp_Y;
X_nor_Y      <= temp_X nor temp_Y;
X_xor_Y      <= temp_X xor temp_Y;
X_xnor_Y     <= temp_X xnor temp_Y;

```

end process;

END;

```

-- *****
-- **** STUDENT: 64200100
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****

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

entity alu_tb is
generic( n:natural:=8 );
end alu_tb;

architecture behavior of alu_tb is
component alu_cla
    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 component;

signal      M: std_logic := '0';
signal      F: std_logic_vector( 2 downto 0 ):= ( others=>'0' );
signal      X: std_logic_vector( n-1 downto 0 ):= ( others=>'0' );
signal      Y: std_logic_vector( n-1 downto 0 ):= ( others=>'0' );

signal      S: std_logic_vector( n-1 downto 0 );
signal      Negative: std_logic;
signal      Cout: std_logic;
signal      Overflow: std_logic;
signal      Zero: std_logic;
signal      Gout: std_logic;
signal      Pout: std_logic;

signal      Sprocs: std_logic_vector( n-1 downto 0 ):= ( others=>'0' );
signal      X8u,Y8u: integer :=0;
begin

```

```

uut:alu_cla
generic map( n=>n )
port map( M=>M,F=>F,X=>X,Y=>Y,S=>S,Negative=>Negative,Cout=>Cout,Overflow=>Overflow,Zero=>Zero,Gout=>Gout,Pout=>Pout
);

stimproc:process
begin
for i in 0 to ( 2**n )-1 loop
    X8u    <=i;
    X      <= std_logic_vector( to_unsigned( i,n ) );
    for j in 0 to ( 2**n )-1 loop
        Y8u    <=j;
        Y      <= std_logic_vector( to_unsigned( j,n ) );

        F      <="000";
        wait for 1 ns;
        assert( Sprocs=S );
        F      <="001";
        wait for 1 ns;
        assert( Sprocs=S );
    end loop;
end loop;
wait;
end process;

procs: process ( X8u,Y8u,F )
begin
if F( 0 )='0' then
    Sprocs <= std_logic_vector( to_unsigned( ( X8u+Y8u ) mod 2**n, n ) );
else
    Sprocs <= std_logic_vector( to_unsigned( ( X8u-Y8u ) mod 2**n, n ) );
end if;
end process;
end;

```

```

-- *****
-- **** STUDENT: 64200163
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek Manjka assert.
Zakasnitev CLA ni 10 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity alu_tb is
    generic( n: natural := 8 );
end alu_tb;

architecture behavior of alu_tb is
    component alu_cla
        generic( n: natural := 8 );
    port( M: in std_logic;
          F: in std_logic_vector( 2 downto 0 );
          X: in std_logic_vector( n-1 downto 0 );
          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 component;

    signal M : std_logic := '0';
    signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
    signal X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    signal Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    signal S : std_logic_vector( n-1 downto 0 );
    signal Negative : std_logic;
    signal Cout : std_logic;
    signal Overflow : std_logic;

```

```

signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;

      signal      X_int, Y_int : integer := 0;
      signal      S_comp : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

begin
  uut: alu_cla
    generic map( n => n )
    port map(
      M => M,      F => F,      X => X,      Y => Y,      S => S,
      Negative => Negative,      Cout => Cout,      Overflow => Overflow,      Zero => Zero,
      Gout => Gout,      Pout => Pout
    );
  stim_proc: process
begin
    for i in 0 to ( 2**n )-1 loop
      X      <= std_logic_vector( to_unsigned( i, n ) );
      for j in 0 to 2**n - 1 loop
        Y      <= std_logic_vector( to_unsigned( j, n ) );
        F      <= "000";
        wait for 10 ns;
        F      <= "001";
        wait for 10 ns;
      end loop;
    end loop;

    wait;
  end process;
end;

```

```

-- *****
-- **** STUDENT: 64200238
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)).
Solze se bodo posušile - z leti ni nič drugače, samo človek se navadi. Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej
vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY alu_tb IS
  GENERIC ( n: natural := 8 );
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS

  COMPONENT alu_cla
    generic(
      n : natural := 8  -- Število bitov za vhodne in izhodne signale
    );
    PORT(
      M : IN std_logic;
      F : IN std_logic_vector( 2 downto 0 );
      X : IN std_logic_vector( n-1 downto 0 );
      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 COMPONENT;

  -- Inputs
  signal M : std_logic := '0';
  signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );

```

```

signal      X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
signal      Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

-- Outputs
signal      S : std_logic_vector( n-1 downto 0 );
signal      Negative : std_logic;
signal      Cout : std_logic;
signal      Overflow : std_logic;
signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;
signal      operacija : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
signal      X_sig, Y_sig : integer :=0;
signal      all_zeros : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

BEGIN

    -- Instantiate the Unit Under Test ( UUT )
    uut: alu_cla PORT MAP (
        M => M,
        F => F,
        X => X,
        Y => Y,
        S => S,
        Negative => Negative,
        Cout => Cout,
        Overflow => Overflow,
        Zero => Zero,
        Gout => Gout,
        Pout => Pout
    );

    -- Clock process definitions

    stim_proc: process
    begin
        M      <= '0';
        for i in 0 to 2**n-1 loop
            X   <= std_logic_vector( to_unsigned( i, n ) );
            X_sig <= i;

```



```

        for j in 0 to 2**n-1 loop
Y      <= std_logic_vector( to_unsigned( j, n ) );
Y_sig <= j;
F      <= "000";
wait for 1 ns;
F      <= "001";
wait for 1 ns;
        end loop;
    end loop;

wait;
end process;
-- bom naredil še en proces za seštevanje in odštevanje, ker v prejšnjem TBju sem imel probleme z pretvorbami
med podatkovnimi tipi in mislm da bo tako lažje
-- Pa tudi baje dela vse vzporedno tako da Ni pence
Bog_pomagaj: process( F, X_sig, Y_sig ) -- izgubil sem 1 uro, ker nisem dal "( F, X_sig, Y_sig )" sedaj ko
pomislim je kar smiselno da se proces zgodi s spramembo vrednosti :D
begin
    M      <= '0';
    if F = "000" then

operacija  <= std_logic_vector( to_unsigned(
( X_sig + Y_sig ) mod 2**n, n ) );
    elsif F = "001" then

operacija  <= std_logic_vector( to_unsigned(
( X_sig - Y_sig ) mod 2**n, n ) );
    else

operacija  <= all_zeros;
    end if;

    -- wait;
end process;

    monitor_proc: process -- všec so mi procesi :D
    begin
wait for 1 ns;
if operacija /= S then
assert false

```

```
        report "napaka! operacija /= S. moral bi bit: " &  
        integer'image( to_integer( unsigned( S ) ) ) & ", je pa: " &  
        integer'image( to_integer( unsigned( operacija ) ) )  
severity error;  
        else report "Vse je kul";  
    end if;  
    end process;  
END;
```

```

-- *****
-- **** STUDENT: 64200288
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)). Zakasnitev
CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY alu_tb IS
  GENERIC ( n: natural := 8 );
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS

  COMPONENT alu_cla
    generic(
      n : natural := 8  -- Število bitov za vhodne in izhodne signale
    );
    PORT(
      M : IN std_logic;
      F : IN std_logic_vector( 2 downto 0 );
      X : IN std_logic_vector( n-1 downto 0 );
      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 COMPONENT;

  -- Inputs
  signal M : std_logic := '0';
  signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
  signal X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

```

signal          Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

-- Outputs
signal          S : std_logic_vector( n-1 downto 0 );
signal          Negative : std_logic;
signal          Cout : std_logic;
signal          Overflow : std_logic;
signal          Zero : std_logic;
signal          Gout : std_logic;
signal          Pout : std_logic;
signal          operacija : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
signal          X_sig, Y_sig : integer :=0;
signal          all_zeros : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

BEGIN

    -- Instantiate the Unit Under Test ( UUT )
    uut: alu_cla PORT MAP (
    M => M,
    F => F,
    X => X,
    Y => Y,
    S => S,
    Negative => Negative,
    Cout => Cout,
    Overflow => Overflow,
    Zero => Zero,
    Gout => Gout,
    Pout => Pout
    );

    -- Clock process definitions

    stim_proc: process
    begin
        M      <= '0';
        for i in 0 to 2**n-1 loop
            X   <= std_logic_vector( to_unsigned( i, n ) );
            X_sig <= i;
            for j in 0 to 2**n-1 loop

```

```

Y      <= std_logic_vector( to_unsigned( j, n ) );
Y_sig <= j;
F      <= "000";
wait for 1 ns;
F      <= "001";
wait for 1 ns;
      end loop;
end loop;

wait;
end process;
-- bom naredil še en proces za seštevanje in odštevanje, ker v prejšnjem TBju sem imel probleme z pretvorbami
med podatkovnimi tipi in mislm da bo tako lažje
-- Pa tudi baje dela vse vzporedno tako da Ni pence
Bog_pomagaj: process( F, X_sig, Y_sig ) -- izgubil sem 1 uro, ker nisem dal "( F, X_sig, Y_sig )" sedaj ko
pomislim je kar smiselno da se proces zgodi s spramembo vrednosti :D
begin
    M      <= '0';
    if F = "000" then

operacija  <= std_logic_vector( to_unsigned(
( X_sig + Y_sig ) mod 2**n, n ) );
    elsif F = "001" then

operacija  <= std_logic_vector( to_unsigned(
( X_sig - Y_sig ) mod 2**n, n ) );
    else

operacija  <= all_zeros;
    end if;

    -- wait;
end process;

    monitor_proc: process -- všec so mi procesi :D
    begin
wait for 1 ns;
if operacija /= S then
assert false
        report "napaka! operacija /= S. moral bi bit: " &

```

```
integer'image( to_integer( unsigned( S ) ) ) & ", je pa: " &  
integer'image( to_integer( unsigned( operacija ) ) )  
severity error;  
    else report "Vse je kul";  
end if;  
end process;  
END;
```

```

-- *****
-- **** STUDENT: 64200296
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)).
Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;

-- Uncomment the following Library declaration if using
-- arithmetic functions with Signed or Unsigned values
-- USE ieee.numeric_std.ALL;

ENTITY alu_tb IS
    generic( n: natural := 8 );
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS

    -- Component Declaration for the Unit Under Test ( UUT )

    COMPONENT alu_cla
    PORT(
        M : IN std_logic;
        F : IN std_logic_vector( 2 downto 0 );
        X : IN std_logic_vector( 7 downto 0 );
        Y : IN std_logic_vector( 7 downto 0 );
        S : OUT std_logic_vector( 7 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 COMPONENT;

    -- Inputs

```

```

signal      M : std_logic := '0';
signal      F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
signal      X : std_logic_vector( 7 downto 0 ) := ( others => '0' );
signal      Y : std_logic_vector( 7 downto 0 ) := ( others => '0' );

-- Outputs
signal      S : std_logic_vector( 7 downto 0 );
signal      Negative : std_logic;
signal      Cout : std_logic;
signal      Overflow : std_logic;
signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;

      signal      s_test, X_and_Y, X_nand_Y, X_or_Y, X_nor_Y, X_xor_Y, X_xnor_Y : std_logic_vector( n-1 downto 0 )
:= ( others => '0' );
      signal      negative_t, Over, Zero_test : std_logic := '0';
      signal      x_test, y_test, mi_t : integer := 0;

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

BEGIN

      -- Instantiate the Unit Under Test ( UUT )
      uut: alu_cla PORT MAP (
M => M,
F => F,
X => X,
Y => Y,
S => S,
Negative => Negative,
Cout => Cout,
Overflow => Overflow,
Zero => Zero,
Gout => Gout,
Pout => Pout
);

      -- Stimulus process

```



```

stim_proc: process
begin
    for i in -( 2**( n-1 ) ) to 2**( n-1 )-1 loop
        x_test<=i;
        x      <= std_logic_vector( to_signed( i, n ) );
        for mi in 0 to 1 loop
            mi_t <=mi;
            if mi = 0 then
                m      <='0';
            else
                m      <='1';
            end if;
            for fi in 0 to n-1 loop
                f      <= std_logic_vector( to_unsigned( fi, f'length ) );
                for j in -( 2**( n-1 ) ) to 2**( n-1 )-1 loop
                    y_test<=j;
                    y      <= std_logic_vector( to_signed( j, n ) );

                    wait for 1 ns;
                    if mi=0 and fi<5 then
                        assert( s_test = S ) report "add/sub Failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
                        & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                        assert( negative_t = negative ) report "add/sub negative Failed. X:" & integer'image( i ) & " Y:"
                        & integer'image( j ) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                        assert( over = overflow ) report "add/sub overflow Failed. X:" & integer'image( i ) & " Y:" &
                        integer'image( j ) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                        assert( Zero_test = zero ) report "add/sub zero Failed. X:" & integer'image( i ) & " Y:" &
                        integer'image( j ) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                    end if;

                    if mi=0 and fi=5 then
                        assert( s_test = S ) report "add/sub Failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
                        & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                        assert( negative_t = negative ) report "add/sub negative Failed. X:" & integer'image( i ) & " Y:"
                        & integer'image( j ) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                        assert( Zero_test = zero ) report "add/sub zero Failed. X:" & integer'image( i ) & " Y:" &
                        integer'image( j ) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                    end if;

                    if mi=0 and fi>5 then

```

```

        assert( Zero_test = zero ) report "add/sub zero Failed. X:" & integer'image( i ) & " Y:" &
integer'image( j ) & "M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
        end if;

        if mi=1 and fi = 0 then
            assert( X_and_Y = S ) report "x and y failed. X:" & integer'image( i ) & " Y:" & integer'image( j
) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
            end if;
            if mi=1 and fi = 1 then
                assert( X_nand_Y = S ) report "x nand y failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                end if;
                if mi=1 and fi = 2 then
                    assert( X_or_Y = S ) report "x or y failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
& " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                    end if;
                    if mi=1 and fi = 3 then
                        assert( X_nor_Y = S ) report "x nor y failed. X:" & integer'image( i ) & " Y:" & integer'image( j
) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                        end if;
                        if mi=1 and fi = 4 then
                            assert( X_xor_Y = S ) report "x xor y failed. X:" & integer'image( i ) & " Y:" & integer'image( j
) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                            end if;
                            if mi=1 and fi = 5 then
                                assert( X_xnor_Y = S ) report "x xnor y failed. X:" & integer'image( i ) & " Y:" & integer'image(
j ) & " M:" & integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                                end if;
                                if mi=1 and fi = 6 then
                                    assert( X = S ) report "x failed. X:" & integer'image( i ) & " Y:" & integer'image( j ) & " M:" &
integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                                    end if;
                                    if mi=1 and fi = 7 then
                                        assert( Y = S ) report "y failed. X:" & integer'image( i ) & " Y:" & integer'image( j ) & " M:" &
integer'image( mi ) & " F:" & integer'image( fi ) severity ERROR;
                                        end if;
                                    end loop;
                                end loop;
                            end loop;
                        end loop;
                    end loop;
                end loop;
            end loop;
        end loop;

```

```
wait;  
end process;
```

```
test: process ( x_test, y_test, f, mi_t )  
VARIABLE s_t, x_t, y_t : std_logic_vector( n-1 downto 0 );  
VARIABLE sum_test : integer;  
BEGIN  
x_t := std_logic_vector( to_signed( x_test, n ) );  
y_t := std_logic_vector( to_signed( y_test, n ) );
```

```
        X_and_Y      <= x_t and y_t;  
X_nand_Y    <= x_t nand y_t;  
X_or_Y      <= x_t or y_t;  
X_nor_Y     <= x_t nor y_t;  
X_xor_Y     <= x_t xor y_t;  
X_xnor_Y    <= x_t xnor y_t;
```

```
        if f = "101" then  
            S_t := minusone;  
            Over <= '0';  
        else  
            case f is  
                when "000" =>  
                    sum_test := x_test + y_test;  
                when "001" =>  
                    sum_test := x_test - y_test;  
                when "010" =>  
                    sum_test := x_test + 1;  
                when "011" =>  
                    sum_test := x_test - 1;  
                when "100" =>  
                    sum_test := x_test + x_test;  
                when "110" =>  
                    sum_test := x_test;  
                when "111" =>  
                    sum_test := y_test;  
                when others =>  
                    sum_test := 0;  
            end case;
```

```

        if sum_test > 2**( n-1 ) - 1 or sum_test < -( 2**( n-1 ) ) then
Over  <= '1';
else
Over  <= '0';
end if;
s_t := std_logic_vector( to_unsigned( sum_test MOD 2**n, n ) );
        end if;

s_test      <= s_t;
Negative_t  <= s_t( n-1 );
if s_t = vectorzero or ( mi_t=0 and( f="111" or f="110" ) ) then
Zero_test   <= '1';
else
Zero_test   <= '0';
end if;

end process;

END;

```

```

-- *****
-- **** STUDENT: 64200385
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY alu_tb IS
  GENERIC( n: natural := 8 );
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS

  -- Component Declaration for the Unit Under Test ( UUT )

  COMPONENT alu_cla
    GENERIC( n: natural := 8 );
  PORT(
    M : IN std_logic;
    F : IN std_logic_vector( 2 downto 0 );
    X : IN std_logic_vector( n-1 downto 0 );
    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 COMPONENT;

  -- Inputs
  signal M : std_logic := '0';
  signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
  signal X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
  signal Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

```

-- Outputs
signal      S : std_logic_vector( n-1 downto 0 );
signal      Negative : std_logic;
signal      Cout : std_logic;
signal      Overflow : std_logic;
signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;
signal      X_int, Y_int : integer := 0;
signal      S_comp : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

```

BEGIN

```

```

-- Instantiate the Unit Under Test ( UUT )
uut: alu_cla GENERIC MAP ( n=>n ) PORT MAP (
M => M,
F => F,
X => X,
Y => Y,
S => S,
Negative => Negative,
Cout => Cout,
Overflow => Overflow,
Zero => Zero,
Gout => Gout,
Pout => Pout
);

-- Stimulus process
stim_proc: process
begin
for i in 0 to 2**n-1 loop
    x_int <= i;
    x <= std_logic_vector( to_unsigned( i, n ) );
    for j in 0 to 2**n - 1 loop
        y_int <= j;
        y <= std_logic_vector( to_unsigned( j, n ) );
        F <= "000";
        wait for 1 ns;
    end loop
end loop
end process

```

```

        assert( S_comp = S ) report "Sum FAIL => X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;
        F      <= "001";
        wait for 1 ns;
        assert( S_comp = S ) report "Dif FAIL => X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;
        end loop;
        end loop;

wait;
end process;

comp_proc: process ( X_int, Y_int, F )
begin
    if F( 0 ) = '0' then
        S_comp <= std_logic_vector( to_unsigned( ( X_int + Y_int ) mod 2**n, n ) );
    else
        S_comp <= std_logic_vector( to_unsigned( ( X_int - Y_int ) mod 2**n, n ) );
    end if;
end process;
END;

```

```

-- *****
-- **** STUDENT: 64210113
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****

```

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

```

```

ENTITY alu_tb IS
  GENERIC( n: natural := 8 );
END alu_tb;

```

```

ARCHITECTURE behavior OF alu_tb IS
  COMPONENT alu_cla
    GENERIC( n: natural := 8 );
  PORT(
    M : IN std_logic;
    F : IN std_logic_vector( 2 downto 0 );
    X : IN std_logic_vector( n-1 downto 0 );
    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 COMPONENT;

```

-- *Inputs*

```

signal M : std_logic := '0';
signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
signal X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
signal Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

-- *Outputs*


```

signal      S : std_logic_vector( n-1 downto 0 );
signal      Negative : std_logic;
signal      Cout : std_logic;
signal      Overflow : std_logic;
signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;

      signal      unsignX, unsignY : integer := 0;
      signal      S_res : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

BEGIN

```

      uut: alu_cla
        GENERIC MAP ( n => n )
        PORT MAP (
M => M,
F => F,
X => X,
Y => Y,
S => S,
Negative => Negative,
Cout => Cout,
Overflow => Overflow,
Zero => Zero,
Gout => Gout,
Pout => Pout
);

```

```

stim_proc: process
begin

```

```

      for i in 0 to 2**n - 1 loop

        unsignX      <= i;
        X            <= std_logic_vector( to_unsigned( i, n ) );

        for j in 0 to 2**n - 1 loop
          unsignY      <= j;
          Y            <= std_logic_vector( to_unsigned( j, n ) );

```

```

        F      <= "000";
        wait for 1 ns;
        assert( S_res = S )
        report "Ses error.x:" & integer'image( i ) & "y:" & integer'image( j )
        severity error;

        F      <= "001";
        wait for 1 ns;
        assert( S_res = S )
        report "Dif erro.x:" & integer'image( i ) & "y:" & integer'image( j )
        severity error;

    end loop;
end loop;
wait;
end process;

comp_proc: process ( unsignX, unsignY, F )
begin
    if F( 0 ) = '0' then
        S_res <= std_logic_vector( to_unsigned( ( unsignX+unsignY ) mod 2**n, n ) );

    else
        S_res <= std_logic_vector( to_unsigned( ( unsignX-unsignY ) mod 2**n, n ) );

    end if;
end process;

END;

```

```

-- *****
-- **** STUDENT: 64210290
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Manjka assert.
Zakasnitev CLA ni clk_period (10 ns), ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY alu_cla_tb IS
  GENERIC( N : natural := 8 );
END alu_cla_tb;

ARCHITECTURE behavior OF alu_cla_tb IS

  -- Component Declaration for the Unit Under Test ( UUT )

  COMPONENT alu_cla
    GENERIC( N: natural := 8 );
  PORT(
    M : IN std_logic;
    F : IN std_logic_vector( 2 downto 0 );
    X : IN std_logic_vector( 7 downto 0 );
    Y : IN std_logic_vector( 7 downto 0 );
    S : OUT std_logic_vector( 7 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 COMPONENT;

  -- Inputs
  signal M : std_logic := '0';
  signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
  signal X : std_logic_vector( 7 downto 0 ) := ( others => '0' );

```

```

signal          Y : std_logic_vector( 7 downto 0 ) := ( others => '0' );

-- Outputs
signal          S : std_logic_vector( 7 downto 0 );
signal          Negative : std_logic;
signal          Cout : std_logic;
signal          Overflow : std_logic;
signal          Zero : std_logic;
signal          Gout : std_logic;
signal          Pout : std_logic;

signal          clock : std_logic;
constant        clock_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test ( UUT )
    uut: alu_cla
        GENERIC MAP( N => N )
        PORT MAP (
            M => M,
            F => F,
            X => X,
            Y => Y,
            S => S,
            Negative => Negative,
            Cout => Cout,
            Overflow => Overflow,
            Zero => Zero,
            Gout => Gout,
            Pout => Pout
        );

    -- Clock process definitions
    clock_process : process
    begin
        clock <= '0';
        wait for clock_period/2;
        clock <= '1';
        wait for clock_period/2;
    end process;

```

```

end process;

-- Stimulus process
stim_proc: process
begin
    -- hold reset state for 100 ns.
    wait for 100 ns;

    M    <= '0';

    for idx in -( 2**( n-1 ) ) to 2**( n-1 )-1 loop
        X    <= std_logic_vector( to_signed( idx,n ) );
        for idy in -( 2**( n-1 ) ) to 2**( n-1 )-1 loop
            Y    <= std_logic_vector( to_signed( idy,n ) );
            for idf in 0 to 7 loop
                F    <= std_logic_vector( to_unsigned( idf,3 ) );
                wait for 1 ns;
            end loop;
        end loop;
    end loop;

    wait;
end process;

END;

```

```

-- *****
-- **** STUDENT: 64210382
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY alu_tb IS
    GENERIC( n: natural := 8 );
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS
    COMPONENT alu_cla
        GENERIC( n: natural := 8 );
    PORT(
        M : IN std_logic;
        F : IN std_logic_vector( 2 downto 0 );
        X : IN std_logic_vector( n-1 downto 0 );
        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 COMPONENT;

    -- Inputs
    signal M : std_logic := '0';
    signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
    signal X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    signal Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

    -- Outputs
    signal S : std_logic_vector( n-1 downto 0 );

```

```

signal      Negative : std_logic;
signal      Cout     : std_logic;
signal      Overflow  : std_logic;
signal      Zero      : std_logic;
signal      Gout      : std_logic;
signal      Pout      : std_logic;

      signal      X_int, Y_int : integer := 0;
      signal      S_comp : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

BEGIN
  uut: alu_cla
    GENERIC MAP ( n => n )
    PORT MAP (
      M => M,
      F => F,
      X => X,
      Y => Y,
      S => S,
      Negative => Negative,
      Cout => Cout,
      Overflow => Overflow,
      Zero => Zero,
      Gout => Gout,
      Pout => Pout
    );

  stim_proc: process
  begin
    for i in 0 to 2**n - 1 loop
      X_int <= i;
      X      <= std_logic_vector( to_unsigned( i, n ) );
      for j in 0 to 2**n - 1 loop
        Y_int <= j;
        Y      <= std_logic_vector( to_unsigned( j, n ) );

        F      <= "000";
        wait for 1 ns;
        assert( S_comp = S ) report "Sum failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;

```

```

        F      <= "001";
        wait for 1 ns;
        assert( S_comp = S ) report "Dif failed. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;
        end loop;
    end loop;

wait;
end process;

-- Calculate sum for checking the adder result
comp_proc: process ( X_int, Y_int, F )
begin
    if F( 0 ) = '0' then
        S_comp <= std_logic_vector( to_unsigned( ( X_int + Y_int ) mod 2**n, n ) );
    else
        S_comp <= std_logic_vector( to_unsigned( ( X_int - Y_int ) mod 2**n, n ) );
    end if;
end process;

END;

```



```

-- *****
-- **** STUDENT: 64210384
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)).
Zakasnitev CLA ni 5 ns (2*5/2), ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;

ENTITY alu_tb IS
    generic( n: natural := 8 );
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS

    -- Component Declaration for the Unit Under Test ( UUT )

    COMPONENT alu_cla
    PORT(
        M : IN std_logic;
        F : IN std_logic_vector( 2 downto 0 );
        X : IN std_logic_vector( n-1 downto 0 );
        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 COMPONENT;

    -- Inputs
    signal M : std_logic := '0';
    signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
    signal X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    signal Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

```

-- Outputs
signal      S : std_logic_vector( n-1 downto 0 );
signal      Negative : std_logic;
signal      Cout : std_logic;
signal      Overflow : std_logic;
signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;

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

-- Clock
constant    clk_period : time := 5 ns;
signal      clk : std_logic;

BEGIN

      -- Instantiate the Unit Under Test ( UUT )
      uut: alu_cla PORT MAP (
      M => M,
      F => F,
      X => X,
      Y => Y,
      S => S,
      Negative => Negative,
      Cout => Cout,
      Overflow => Overflow,
      Zero => Zero,
      Gout => Gout,
      Pout => Pout
      );

      -- Clock process definitions
      clk_process :process
      begin
          clk    <= '0';
          wait for 5/2 ns;
          clk    <= '1';
          wait for 5/2 ns;

```

```

end process;

-- Stimulus process
stim_proc: process
begin
z1: for k in 0 to 1 loop -- ( 0 -> sestevanje in 1->odstevanje )
    for i in 0 to 2**n-1 loop -- x
        for j in 0 to 2**n-1 loop -- y
            F      <= std_logic_vector( to_unsigned( k, 3 ) );
            x      <= std_logic_vector( to_unsigned( i, n ) );
            y      <= std_logic_vector( to_unsigned( j, n ) );

            if F = "000" then -- sestevanje
                sum <= std_logic_vector( to_unsigned( ( i+j ) rem 2**n, n ) );
            elsif F = "001" then -- odstevanje
                sum <= std_logic_vector( to_unsigned( ( i + ( 2**n - j ) ) rem 2**n, n ) );
            end if;

-- stabilizacija izhoda
            wait for clk_period;

-- preverim "sum" = "S" ???
            assert( S = sum )
            report "Test case: X = " & integer'image( i ) &
                ", Y = " & integer'image( j ) &
                ", F = " & integer'image( k )

severity error;
            exit z1 when ( s /= sum );
        end loop;
    end loop;
end loop;
wait;
end process;

END;

```

```

-- *****
-- **** STUDENT: 64210386
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)).
Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
USE ieee.numeric_std.all;

-- Uncomment the following Library declaration if using
-- arithmetic functions with Signed or Unsigned values
-- USE ieee.numeric_std.ALL;

ENTITY alu_tb IS
    GENERIC( n: natural := 8 );
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS

    -- Component Declaration for the Unit Under Test ( UUT )

    COMPONENT alu_cla
    PORT(
        M : IN std_logic;
        F : IN std_logic_vector( 2 downto 0 );
        X : IN std_logic_vector( n-1 downto 0 );
        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 COMPONENT;

```

```

-- Inputs
signal      M : std_logic := '0';
signal      F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
signal      X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
signal      Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

-- Outputs
signal      S : std_logic_vector( n-1 downto 0 );
signal      Negative : std_logic;
signal      Cout : std_logic;
signal      Overflow : std_logic;
signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;
-- No clocks detected in port list. Replace <clock> below with
-- appropriate port name

constant    clock_period : time := 10 ns;
signal      Xi, Yi : integer := 0;
signal      S_test : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

BEGIN

    -- Instantiate the Unit Under Test ( UUT )
    uut: alu_cla PORT MAP (
    M => M,
    F => F,
    X => X,
    Y => Y,
    S => S,
    Negative => Negative,
    Cout => Cout,
    Overflow => Overflow,
    Zero => Zero,
    Gout => Gout,
    Pout => Pout
    );

    -- Stimulus process
    stim_proc: process

```

```

begin

    for i in 0 to 2**n - 1 loop
        Xi    <= i;
        X      <= std_logic_vector( to_unsigned( i, n ) );
        for u in 0 to 2**n - 1 loop
            Yi    <= u;
            Y      <= std_logic_vector( to_unsigned( u, n ) );

            F      <= "000";
            wait for 1 ns;
            assert( S_test = S ) report "Fail SUM X:" & integer'image( i ) & " Y:" & integer'image( u )
severity error;

            F      <= "001";
            wait for 1 ns;
            assert( S_test = S ) report "Fail DIF X:" & integer'image( i ) & " Y:" & integer'image( u )
severity error;

        end loop;
    end loop;

    wait;
end process;

test_proc: process ( Xi, Yi, F )
begin

    if F( 0 ) = '0' then
        S_test <= std_logic_vector( to_unsigned( ( Xi + Yi ) mod 2**n, n ) );
    else
        S_test <= std_logic_vector( to_unsigned( ( Xi - Yi ) mod 2**n, n ) );
    end if;

end process;

END;

```

```

-- *****
-- **** STUDENT: 64210445
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)).
Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
USE ieee.numeric_std.all;

-- Uncomment the following Library declaration if using
-- arithmetic functions with Signed or Unsigned values
-- USE ieee.numeric_std.ALL;

ENTITY alu_tb IS
    GENERIC( n: natural := 8 );
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS

    -- Component Declaration for the Unit Under Test ( UUT )

    COMPONENT alu_cla
    PORT(
        M : IN std_logic;
        F : IN std_logic_vector( 2 downto 0 );
        X : IN std_logic_vector( n-1 downto 0 );
        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 COMPONENT;

```

```

-- Inputs
signal      M : std_logic := '0';
signal      F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
signal      X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
signal      Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

-- Outputs
signal      S : std_logic_vector( n-1 downto 0 );
signal      Negative : std_logic;
signal      Cout : std_logic;
signal      Overflow : std_logic;
signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;
-- No clocks detected in port list. Replace <clock> below with
-- appropriate port name

constant    period : time := 10 ns;
signal      Xval, Yval : integer := 0;
signal      S_test : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

BEGIN

    -- Instantiate the Unit Under Test ( UUT )
    uut: alu_cla PORT MAP (
    M => M,
    F => F,
    X => X,
    Y => Y,
    S => S,
    Negative => Negative,
    Cout => Cout,
    Overflow => Overflow,
    Zero => Zero,
    Gout => Gout,
    Pout => Pout
    );

    -- Stimulus process
    stim_proc: process

```



```

begin

    for i in 0 to 2**n - 1 loop
        Xval <= i;
        X <= std_logic_vector( to_unsigned( i, n ) );

        for j in 0 to 2**n - 1 loop
            Yval <= j;
            Y <= std_logic_vector( to_unsigned( j, n ) );

            F <= "000";
            wait for 1 ns;
            assert( S_test = S ) report "Fail at SUM X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;

            F <= "001";
            wait for 1 ns;
            assert( S_test = S ) report "Fail at DIF X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;

        end loop;

    end loop;

wait;

end process;

test_proc: process ( Xval, Yval, F )
begin

    if F( 0 ) = '0' then
        S_test <= std_logic_vector( to_unsigned( ( Xval + Yval ) mod 2**n, n ) );
    else
        S_test <= std_logic_vector( to_unsigned( ( Xval - Yval ) mod 2**n, n ) );
    end if;

end process;

END;

```

```

-- *****
-- **** STUDENT: 64210455
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY alu_tb IS
    GENERIC( n: natural := 8 );
END alu_tb;

ARCHITECTURE sim OF alu_tb IS
    COMPONENT alu_cla
        GENERIC( n: natural := 8 );
        PORT(
            M : IN std_logic;
            F : IN std_logic_vector( 2 downto 0 );
            X : IN std_logic_vector( n-1 downto 0 );
            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 COMPONENT;

    signal mode : std_logic := '0';
    signal func : std_logic_vector( 2 downto 0 ) := ( others => '0' );
    signal operand_X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    signal operand_Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    signal result_S : std_logic_vector( n-1 downto 0 );
    signal neg_flag : std_logic;
    signal carry_out : std_logic;
    signal overflow : std_logic;

```

```

signal          zero_flag : std_logic;
signal          gen_out   : std_logic;
signal          prop_out  : std_logic;

signal          alu_operation : std_logic_vector( 3 downto 0 );
signal          nAddSub      : std_logic;
signal          intermediate_Y: std_logic_vector( n-1 downto 0 );
signal          int_X, int_Y : integer := 0;
signal          comp_result  : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

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

```

BEGIN

```

DUT: alu_cla
  GENERIC MAP ( n => n )
  PORT MAP (
    M => mode,
    F => func,
    X => operand_X,
    Y => operand_Y,
    S => result_S,
    Negative => neg_flag,
    Cout => carry_out,
    Overflow => overflow,
    Zero => zero_flag,
    Gout => gen_out,
    Pout => prop_out
  );

alu_operation      <= mode & func;

nAddSub            <= alu_operation( 0 );

intermediate_Y     <= ( others => '0' ) when alu_operation = "0100" else
( operand_Y xor ( others => nAddSub ) ) when alu_operation( 0 ) = '1' else
operand_Y;

WITH alu_operation SELECT
result_S           <= operand_X + operand_Y WHEN "0000", -- Seštevanje
operand_X - operand_Y WHEN "0001", -- Odštevanje

```

```

operand_X and operand_Y WHEN "1000", -- Logična AND
operand_X or operand_Y WHEN "1010", -- Logična OR
operand_X xor operand_Y WHEN "1100", -- Logična XOR
operand_X xnor operand_Y WHEN "1101", -- Logična XNOR
std_logic_vector( to_unsigned( 1, n ) ) WHEN "0010", -- Dodaj 1
( others => '0' ) WHEN others; -- Neveljavne operacije

stimulus_process: process
begin
    -- Test za vse možne kombinacije X in Y
    for i in 0 to ( 2**n ) - 1 loop
        int_X <= i;
        operand_X <= std_logic_vector( to_unsigned( i, n ) );

        for j in 0 to ( 2**n ) - 1 loop
            int_Y <= j;
            operand_Y <= std_logic_vector( to_unsigned( j, n ) );

            for op in 0 to 3 loop
                func <= std_logic_vector( to_unsigned( op, 3 ) );
                mode <= '0';
                wait for 1 ns;

                -- Preverjanje rezultatov
                assert ( result_S = comp_result )
                report "Mismatch in operation: " & integer'image( i ) & " and " & integer'image( j )
                severity error;
            end loop;
        end loop;
    end loop;
    wait;
end process;

result_computation: process ( int_X, int_Y, alu_operation )
begin
    if alu_operation( 3 downto 0 ) = "0000" then
        comp_result <= std_logic_vector( to_unsigned( ( int_X + int_Y ) mod ( 2**n ), n ) );
    elsif alu_operation( 3 downto 0 ) = "0001" then
        comp_result <= std_logic_vector( to_unsigned( ( int_X - int_Y ) mod ( 2**n ), n ) );
    else

```

```
comp_result <= ( others => '0' );  
end if;  
end process;
```

```
END ARCHITECTURE sim;
```

```

-- *****
-- **** STUDENT: 64210457
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)).
Zakasnitev CLA ni 1 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
-- test of only alu addition and subtraction
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY alu_tb IS
    generic ( n:natural :=8 );
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS

    -- Component Declaration for the Unit Under Test ( UUT )

    COMPONENT alu_cla
    PORT(
        M : IN std_logic;
        F : IN std_logic_vector( 2 downto 0 );
        X : IN std_logic_vector( n-1 downto 0 );
        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 COMPONENT;

    -- Inputs
    signal      M : std_logic := '0';
    signal      F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
    signal      X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

```

signal          Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

-- Outputs
signal          S : std_logic_vector( n-1 downto 0 );
signal          Negative : std_logic;
signal          Cout : std_logic;
signal          Overflow : std_logic;
signal          Zero : std_logic;
signal          Gout : std_logic;
signal          Pout : std_logic;

-- signals for testing
signal          X_n, Y_n : integer := 0;
signal          S_check: std_logic_vector( 8-1 downto 0 ) := ( others => '0' );

BEGIN

    -- Instantiate the Unit Under Test ( UUT )
    uut: alu_cla PORT MAP (
    M => M,
    F => F,
    X => X,
    Y => Y,
    S => S,
    Negative => Negative,
    Cout => Cout,
    Overflow => Overflow,
    Zero => Zero,
    Gout => Gout,
    Pout => Pout
    );

    -- Stimulus process
    stim_proc: process
    begin
        for i in 0 to 2**n - 1 loop
            X_n    <= i;
            X      <= std_logic_vector( to_unsigned( i, n ) );
            for j in 0 to 2**n - 1 loop
                Y_n    <= j;-- priredimo j Y_int, s katerim zračunamo vsoto v drugem procesu
            end loop
        end loop
    end process

```

```

        Y      <= std_logic_vector( to_unsigned( j, n ) );

        F      <= "000";    -- seÅ;tevanje
        wait for 1 ns;
        assert( S_check = S ) report "Fail addition. X:" & integer'image( i ) & " Y:" & integer'image( j )
severity error;

        F      <= "001";-- odÅ;tevanje
        wait for 1 ns;
        assert( S_check = S ) report "Fail subtraction. X:" & integer'image( i ) & " Y:" & integer'image(
j ) severity error;
        end loop;
    end loop;

wait;
end process;

-- Calculate sum for checking the adder result
check_proc: process ( X_n, Y_n, F )
begin
    if F( 0 ) = '0' then
        S_check      <= std_logic_vector( to_unsigned( ( X_n + Y_n ) mod 2**n, n ) );
    else
        S_check      <= std_logic_vector( to_unsigned( ( X_n - Y_n ) mod 2**n, n ) );
    end if;
end process;

END;

```



```

-- *****
-- **** STUDENT: 64240429
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Manjka assert. Izbrani so samo točno določeni primeri iz testiranja CLA. Ideja je bila, da testirate
na celotni obseg števil x in y (0...2**n-1).
-- *****
library IEEE;
USE ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_MISC.ALL;      -- reduction operators ( AND_REDUCE, OR_REDUCE )
use ieee.numeric_std.all;

ENTITY alu_tb IS
    GENERIC( n: natural := 8 );
END alu_tb;

architecture testbench of alu_tb is
    component alu_cla
        generic ( n: natural := 8 );
        port(
            M : in std_logic;
            F : in std_logic_vector( 2 downto 0 );
            X : in std_logic_vector( n-1 downto 0 );
            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 component;

    -- Inputs
    signal M : std_logic := '0';
    signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
    signal X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );
    signal Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

```

-- Outputs
signal      S : std_logic_vector( n-1 downto 0 );
signal      Negative : std_logic;
signal      Cout : std_logic;
signal      Overflow : std_logic;
signal      Zero : std_logic;
signal      Gout : std_logic;
signal      Pout : std_logic;

signal      X_int, Y_int : integer := 0;
signal      S_comp : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

begin
  uut: alu_cla
    generic map ( n => n )
    port map (
      M => M,
      F => F,
      X => X,
      Y => Y,
      S => S,
      Negative => Negative,
      Cout => Cout,
      Overflow => Overflow,
      Zero => Zero,
      Gout => Gout,
      Pout => Pout
    );

  stim: process
  begin
    -- for i in 0 to ( 2**n - 1 ) loop
    -- X  <= std_logic_vector( to_unsigned( i, n ) );
    -- for j in 0 to ( 2**n - 1 ) loop
    -- Y  <= std_logic_vector( to_unsigned( j, n ) );
    -- F( 0 )  <= '0';
    -- wait for 2 ns;
    -- F( 0 )  <= '1';
    -- wait for 2 ns;

```

```
-- end loop;  
-- end loop;
```

```
M    <= '1';  
F    <= "001";  
X    <= std_logic_vector( to_signed( -86, n ) );  
Y    <= std_logic_vector( to_signed( 85, n ) );  
wait for 20 ns;
```

```
M    <= '1';  
F    <= "000";  
X    <= std_logic_vector( to_signed( -86, n ) );  
Y    <= std_logic_vector( to_signed( 85, n ) );  
wait for 20 ns;
```

```
M    <= '1';  
F    <= "001";  
X    <= std_logic_vector( to_signed( -86, n ) );  
Y    <= std_logic_vector( to_signed( 85, n ) );  
wait for 20 ns;
```

```
M    <= '1';  
F    <= "010";  
X    <= std_logic_vector( to_signed( -86, n ) );  
Y    <= std_logic_vector( to_signed( 85, n ) );  
wait for 20 ns;
```

```
M    <= '1';  
F    <= "011";  
X    <= std_logic_vector( to_signed( -86, n ) );  
Y    <= std_logic_vector( to_signed( 85, n ) );  
wait for 20 ns;
```

```
M    <= '1';  
F    <= "100";  
X    <= std_logic_vector( to_signed( -86, n ) );  
Y    <= std_logic_vector( to_signed( 85, n ) );  
wait for 20 ns;
```

```
M    <= '1';
```

```

F    <= "101";
X    <= std_logic_vector( to_signed( -86, n ) );
Y    <= std_logic_vector( to_signed( 85, n ) );
wait for 20 ns;

M    <= '1';
F    <= "110";
X    <= std_logic_vector( to_signed( -86, n ) );
Y    <= std_logic_vector( to_signed( 85, n ) );
wait for 20 ns;

M    <= '1';
F    <= "111";
X    <= std_logic_vector( to_signed( -86, n ) );
Y    <= std_logic_vector( to_signed( -86, n ) );
wait for 20 ns;

    -- all Fs are done

    -- M = 0; F=0
M    <= '0';
F    <= "000";
X    <= std_logic_vector( to_signed( 14, n ) );
Y    <= std_logic_vector( to_signed( -14, n ) );
wait for 20 ns;

M    <= '0';
F    <= "000";
X    <= std_logic_vector( to_signed( 14, n ) );
Y    <= std_logic_vector( to_signed( -15, n ) );
wait for 20 ns;

M    <= '0';
F    <= "000";
X    <= std_logic_vector( to_signed( 14, n ) );
Y    <= std_logic_vector( to_signed( 127, n ) );
wait for 20 ns;

M    <= '0';
F    <= "000";

```

```

X    <= std_logic_vector( to_signed( -14, n ) );
Y    <= std_logic_vector( to_signed( 127, n ) );
wait for 20 ns;

```

```

    -- M=0; F=1
M    <= '0';
F    <= "001";
X    <= std_logic_vector( to_signed( 14, n ) );
Y    <= std_logic_vector( to_signed( 14, n ) );
wait for 20 ns;

```

```

M    <= '0';
F    <= "001";
X    <= std_logic_vector( to_signed( 14, n ) );
Y    <= std_logic_vector( to_signed( 15, n ) );
wait for 20 ns;

```

```

M    <= '0';
F    <= "001";
X    <= std_logic_vector( to_signed( 14, n ) );
Y    <= std_logic_vector( to_signed( -127, n ) );
wait for 20 ns;

```

```

M    <= '0';
F    <= "001";
X    <= std_logic_vector( to_signed( -14, n ) );
Y    <= std_logic_vector( to_signed( 127, n ) );
wait for 20 ns;

```

```

    -- Different Fs starting with 2
M    <= '0';
F    <= "010";
X    <= std_logic_vector( to_signed( 127, n ) );
Y    <= std_logic_vector( to_signed( 127, n ) );
wait for 20 ns;

```

```

M    <= '0';
F    <= "011";
X    <= std_logic_vector( to_signed( -128, n ) );
Y    <= std_logic_vector( to_signed( 127, n ) );

```

```

wait for 20 ns;

M    <= '0';
F    <= "100";
X    <= std_logic_vector( to_signed( 127, n ) );
Y    <= std_logic_vector( to_signed( 127, n ) );
wait for 20 ns;

M    <= '0';
F    <= "100";
X    <= std_logic_vector( to_signed( -128, n ) );
Y    <= std_logic_vector( to_signed( 127, n ) );
wait for 20 ns;

M    <= '0';
F    <= "101";
X    <= std_logic_vector( to_signed( -128, n ) );
Y    <= std_logic_vector( to_signed( 127, n ) );
wait for 20 ns;

M    <= '0';
F    <= "110";
X    <= std_logic_vector( to_signed( -128, n ) );
Y    <= std_logic_vector( to_signed( 127, n ) );
wait for 20 ns;

wait;
end process;

end;

```

```

-- *****
-- **** STUDENT: 64240430
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)).
Zakasnitev CLA ni 5 ns, ampak ca. 11 ns (glej vrednost combinatorial path delay).
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;

ENTITY alu_tb IS
END alu_tb;

ARCHITECTURE behavior OF alu_tb IS

    -- generisanje prirodnog broja
    constant n : integer := 8;
    -- Component Declaration for the Unit Under Test ( UUT )

    COMPONENT alu_cla
    PORT(
    M : IN std_logic;
    F : IN std_logic_vector( 2 downto 0 );
    X : IN std_logic_vector( n-1 downto 0 );
    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 COMPONENT;

    -- Inputs
    signal M : std_logic := '0';
    signal F : std_logic_vector( 2 downto 0 ) := ( others => '0' );
    signal X : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

```

```

signal          Y : std_logic_vector( n-1 downto 0 ) := ( others => '0' );

-- Outputs
signal          S : std_logic_vector( n-1 downto 0 );
signal          Negative : std_logic;
signal          Cout : std_logic;
signal          Overflow : std_logic;
signal          Zero : std_logic;
signal          Gout : std_logic;
signal          Pout : std_logic;

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

-- kreiranje kLoka
constant        clk_period : time := 5 ns;
signal          clk : std_logic;

BEGIN

    -- Instantiate the Unit Under Test ( UUT )
    uut: alu_cla PORT MAP (
        M => M,
        F => F,
        X => X,
        Y => Y,
        S => S,
        Negative => Negative,
        Cout => Cout,
        Overflow => Overflow,
        Zero => Zero,
        Gout => Gout,
        Pout => Pout
    );

    -- Clock process definitions
    clk_process :process
    begin
        clk    <= '0';
        wait for clk_period/2;
        clk    <= '1';
    end process;

```



```

        wait for clk_period/2;
end process;

-- Stimulus process
stim_proc: process
begin
z1: for k in 0 to 1 loop -- dvije moguće vrijednosti ( 0->seštevanja in 1->odštevanja )
    for i in 0 to 2**n-1 loop -- sve vrijednosti za x
        for j in 0 to 2**n-1 loop -- sve vrijednosti za y
-- postavljjanje ulaza
            F      <= std_logic_vector( to_unsigned( k, 3 ) );
            x      <= std_logic_vector( to_unsigned( i, n ) );
            y      <= std_logic_vector( to_unsigned( j, n ) );

-- odabir operacije na osnovu F
            if F = "000" then -- seštevanje
                sum <= std_logic_vector( to_unsigned( ( i+j ) rem 2**n, n ) );
            elsif F = "001" then -- odštevanje
                sum <= std_logic_vector( to_unsigned( ( i + ( 2**n - j ) ) rem 2**n, n ) );
            end if;

-- stabilizacija izlaza
            wait for clk_period;

-- provjera: da li je "sum" jednako "S"
            assert( S = sum )
            report "Test case: X = " & integer'image( i ) &
                ", Y = " & integer'image( j ) &
                ", F = " & integer'image( k ) severity error;
            exit z1 when ( s /= sum );
        end loop; -- unutrašnja petlja za y
    end loop; -- srednja petlja za x
end loop; -- vanjska petlja za Cin
-- zavretak simulacije
wait;
end process;

END;
```

```

-- *****
-- **** STUDENT: 64210132
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Povezava med alu_cla in testbench mora biti parametrizirana (manjka generic map (n=>n)).
Idea datoteke testnih vrednosti je, da preleti celoten obseg števil x in y ter na drugačen način (torej z VHDL +
operatorjem) preveri, če so razlike v izračunu strojne komponente in simulatorja. Koda datoteke testnih vrednosti je
ista kot pri testiranju CLA seštevalnika, kar ni poanta naloge - testirati je treba operaciji seštevanja in odštevanja
preko celotnega obsega operandov X in Y.
***** NASLEDNJIČ KODO NALOŽITE V USTREZEN RAZDELEK (OSTALO_x), KJER JE X ŠTEVILKA DOMAČE NALOGE *****
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY alu_tb IS
generic( n: natural := 8 );
END alu_tb;

ARCHITECTURE sim OF alu_tb IS

    component alu_cla
    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 component;

    signal M: std_logic := '0';
    signal F: std_logic_vector(2 downto 0) := (others => '0');
    signal X, Y: std_logic_vector(n-1 downto 0) := (others => '0');
    signal S: std_logic_vector(n-1 downto 0) := (others => '0');
    signal Negative, Cout, Overflow, Zero, Gout, Pout: std_logic := '0';

BEGIN

    uut: alu_cla port map (
        F => F,

```

```

        M => M,
        X => X,
        Y => Y,
        S => S,
        Cout => Cout,
        Gout => Gout,
        Pout => Pout,
        Negative => Negative,
        Overflow => Overflow,
        Zero => Zero
    );

```

```

stim_proc: process
begin

```

```

    X <= x"0E";   Y <= x"F2";

```

```

    wait for 100ns;   X <= x"0E";   Y <= x"F1";

```

```

    wait for 100ns;   X <= x"0E";   Y <= x"7F";

```

```

    wait for 100ns;   X <= x"F2";   Y <= x"7F";

```

```

    wait for 100ns;

```

```

    F(0) <= '1';

```

```

    X <= x"0E";   Y <= x"0E";

```

```

    wait for 100ns;   X <= x"0E";   Y <= x"0F";

```

```

    wait for 100ns;   X <= x"0E";   Y <= x"81";

```

```

    wait for 100ns;   X <= x"F2";   Y <= x"7F";

```

```

    wait;

```

```

end process;

```

```

END sim;

```