Primjer 1

U VHDL-u realizovati kolo koje obavlja funkciju 4-bitnog „sign-magnitude“ sadibarča.

Kreirati odgovarajući **testbench** pri čemu se ulazi za simulaciju učitavaju iz odgovarajućeg tekstualnog fajla. Prikazati rezultate simulacije.

Izvršiti procese **synthesize**, **translate**, **map** i **place** & **route** i očitati maksimalno kašnjenje kombinacionih kola za dati primjer.

Izvršiti implementaciju kola uz pomoć **Spartan-3E Starter Kit** razvojne platforme. Signale na ulazu kola zadavati uz pomoć odgovarajućih prekidača na razvojnoj ploči, pri čemu su ulazi kratko vezani. Izlaz kola prikazati preko LED (dioda svijetli ukoliko je vrijednost bita **1**).

Rješenje

Cijeli broj se može predstaviti u „sign-magnitude“ formatu gdje MSB predstavlja znak broja, dok se ostali biti odnose na magnitudu broja. Na primjer, broj **3** bi bio predstavljen kao **0011**, dok bi broj **-3** bio predstavljen kao **1011**, ukoliko je u pitanju 4-bitni „sign-magnitude“ format.

VHDL kod za jedno do mogućih rješenja „sign-magnitude“ sabirača dato je u listingu 1.

U radnom direktorijumu kreirati fajl **adder_test.txt**, čiji sadržaj je dat u listingu 2.

VHDL **testbench** kod dat je u listingu 3.

Prikazati rezultate simulacije.

Listing 1 - Sign-magnitude sabirač

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity sign_mag_adder is
    port (
              a : in  std_logic_vector(3 downto 0);
              b : in  std_logic_vector(3 downto 0);
              sum : out  std_logic_vector(4 downto 0)
         );
end sign_mag_adder;

architecture arc_sign_mag_adder of sign_mag_adder is
       signal mag_a, mag_b, max, min: unsigned(2 downto 0);
       signal mag_sum: unsigned(3 downto 0);
       signal sign_a, sign_b, sign_sum: std_logic;
begin
       mag_a <= unsigned(a(2 downto 0));
       mag_b <= unsigned(b(2 downto 0));
       sign_a <= a(3);
       sign_b <= b(3);
       process(mag_a, mag_b, sign_a, sign_b)
       begin
              if mag_a > mag_b then
                     max <= mag_a;
                     min <= mag_b;
                     sign_sum <= sign_a;
```

```
            else
                  max <= mag_b;
                  min <= mag_a;
                  sign_sum <= sign_b;
            end if;
      end process;
      mag_sum <= (('0' & max) + ('0' & min)) when sign_a = sign_b else
                 (('0' & max) - ('0' & min));
      sum <= std_logic_vector(sign_sum & mag_sum);
end arc_sign_mag_adder;
```

Listing 2 – adder_test.txt

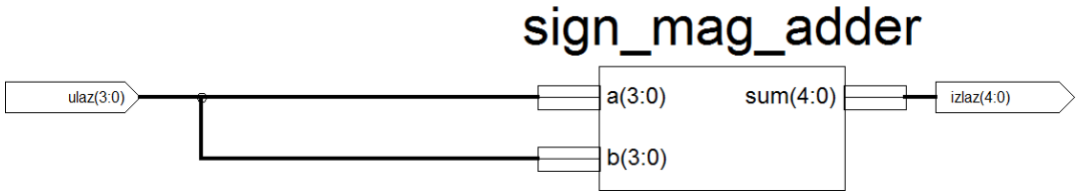
```
10    1011  1011
20    1011  1011
30    1011  1011
40    1011  1011
50    1011  1011
60    1011  1011
70    1011  1011
80    1011  1011
90    1011  1011
100   1011  1011
110   1011  1011
120   1011  1011
130   1011  1011
140   1011  1011
150   1011  1011
160   1011  1011
170   1011  1011
180   1011  1011
190   1011  1011
200   1011  1011
210   1011  1011
220   1011  1011
230   1011  1011
240   1011  1011
250   0111  0111
260   0101  0101
270   0110  0110
280   1001  1001
290   1100  1100
300   0011  0011
```

Listing 3 – Testbench fajl za “sign-magnitude” sabirač

```
entity adder_testbench is
end;

library IEEE;
use ieee.std_logic_1164.all;
use std.textio.all;
use ieee.std_logic_textio.all;
use ieee.numeric_std.all;

architecture stimonly of adder_testbench is

component sign_mag_adder
      port
      (
            a, b: in std_logic_vector(3 downto 0);
```

```
              sum: out std_logic_vector(4 downto 0)
       );
end component;

signal a, b: std_logic_vector(3 downto 0);
signal sum: std_logic_vector(4 downto 0);

begin
uut: sign_mag_adder
       port map( a => a, b => b, sum => sum);

test:
process
       variable tmp_a, tmp_b : std_logic_vector(3 downto 0);
       file vector_file : text is in "adder_test.txt";
       variable l : line;
       variable vector_time : time;
       variable r : real;
       variable good_number, good_val : boolean;
       variable space : character;
begin
while not endfile(vector_file) loop
       readline(vector_file, l);
       read(l, r, good => good_number);
       next when not good_number;
              vector_time := r * 1 ns;
                     if (now < vector_time) then
                            wait for vector_time - now;
                     end if;
       read(l, space);
       read(l, tmp_a, good_val);
       assert good_val report "bad a value";

       read(l, space);
       read(l, tmp_b, good_val);
       assert good_val report "bad b value";

       a <= tmp_a;
       b <= tmp_b;

end loop;
       assert false report "Test complete";
wait;
end process;
end;
```

U cilju verifikacije rada sistema, razvijeno je testno kolo prikazano na slici 1.



Slika 1. “Sign-magnitude” sabirač - testno kolo

Postupak za kreiranje testnog kola je sljedeći:

U okviru **Design View**, izabrati **Implementation**, označiti VHDL dizajn „sign-magnitude“ sabirača, i u okviru **Processes** dvoklik na **Create Schematic Symbol**.

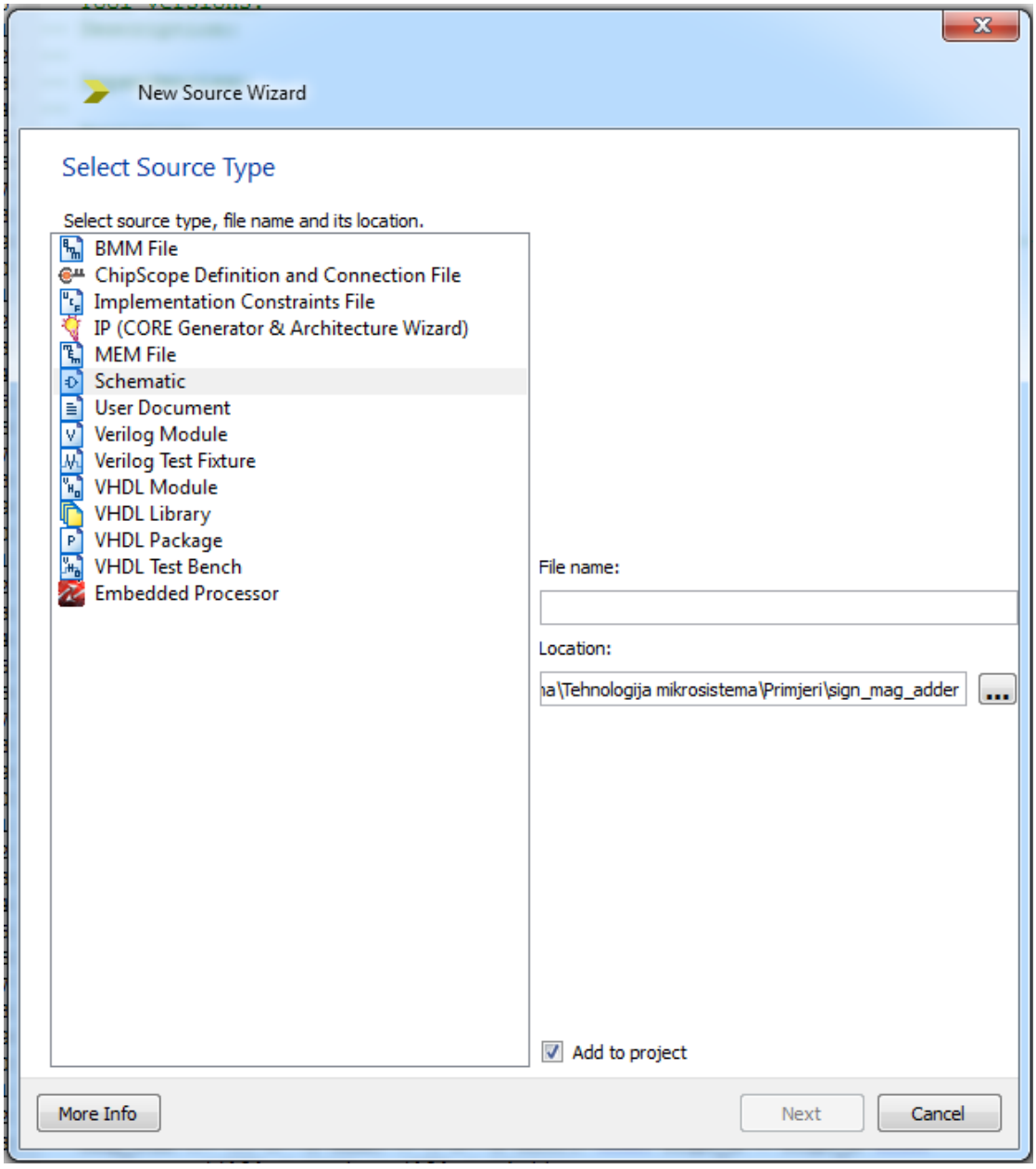
Project->New Source, izabrati **Schematic** (slika 2), unijeti naziv, **<Next>**, **<Finish>**.

Add->Symbol, izabrati kreirani simbol za „sign-magnitude“ sabirač.

Add->Wire, kratko spojiti ulaze, slika 1.

Add->I/O marker, izabrati jedan ulazni i jedan izlazni marker i povezati ih na odgovarajući način. Promijeniti naziv u skladu sa slikom 1.

U okviru **Hierarchy**, označiti kreirani **schematic** fajl, desni klik, izabrati **Set as Top Module**.



Slika 2 - Kreiranje **schematic** fajla

Izvršiti procese **synthesize**, **translate**, **map** i **place** & **route**.

Kreirati **constraints** fajl, listing 4.

Listing 4 – Constraints fajl za "sign-magnitude" sabirač

```
NET "ulaz<0>" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
NET "ulaz<1>" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP ;
NET "ulaz<2>" LOC = "H18" | IOSTANDARD = LVTTL | PULLUP ;
NET "ulaz<3>" LOC = "N17" | IOSTANDARD = LVTTL | PULLUP ;

NET "izlaz<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "izlaz<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "izlaz<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "izlaz<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "izlaz<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
```

Izvršiti implementaciju kola uz pomoć **Spartan-3E Starter Kit** razvojne platforme (pogledati uputstvo u okviru vježbi 2) i verifikovati rad kola.