2. vaja: Labirint

Nadgradili bomo grafični prikaz iz prejšnje vaje, tako da bomo namesto kvadratka premikali žogo, ki bo zapisana kot sličica v pomnilniku. V ozadju bomo dodali preprost labirint in zapisali omejitve za gibanje sličice.

2.1 Gibljiva sličica

Za prikaz sličice bomo v projekt iz prejšnje vaje dodali paket sprites.vhd, v katerem je deklarirana sličica žoge velikosti 30 x 30 točk.

```
type slika30x30 is array (0 to 29) of std_logic_unsigned(29 downto 0);
constant zoga: slika30x30 := (
"00000000000000000000000000000000",
"00000000001111111111000000000000",
"00000000111111111111110000000000",
"00000001111111111111111100000000",
...

"00000000001111111111000000000000" );
```

a. Prenesi datoteko sprites.vhd v projektno mapo, odpri projekt in dodaj datoteko (npr. tako da jo v Quartusu odpreš in izbereš **Project > Add current file to project**).
V datoteki **sistem.vhd** dodaj na vrhu opisa paket:

```
library work;
use work.sprites.all;
```

Deklariraj 30-bitni nepredznačeni vektor **vrst** in zapiši stavek, ki prebere eno vrstico pomnilnika **zoga** iz naslova, ki ga določa razlika **y** koordinat trenutne točke in koordinate objekta. Vrstico beremo iz zbirke, kadar se nahajamo znotraj koordinat objekta, sicer pa jo postavimo na 0:

```
vrst <= zoga(to_integer(y-yp)) when y>=yp and y<yp+30 else (others=>'0');
```

b. Vrstica ima 30 bitov in pri določanju izhodne barve moramo izluščiti ustrezen bit s pomočjo razlike **x** koordinat. Poišči stavek, ki določa barvo kvadrata in zamenjaj izraz `color <= '1'` z izrazom, ki določi barvo z vpogledom v vrstico:

```
color <= vrst(to_integer(x-xp));
```

c. Spremeni stavke za premikanje žoge po zaslonu. V prejšnji vaji smo za upočasnitev premikanja uporabili delilnik ure, sedaj pa namesto delilnika uporabi pogoj, da sta koordinati **x** in **y** na koncu vidnega dela zaslona (`x=800 and y=600`). Ta pogoj je izpolnjen 72-krat na sekundo (frekvenca osveževanja slike), zato se bo žoga premikala nekoliko počasneje. Hitrejše premikanje dosežeš s povečevanjem vrednosti **xp** in **yp** za 3 ali 4. Prevedi opis sistema in preizkusi delovanje na razvojni plošči.

d. Dodaj pogoje, ki omejijo premikanje žoge, tako da se bo gibala v oknu 512 x 512 točk.

2.2 Labirint

V datoteki **sprites.vhd** je deklarirana tudi slika velikosti 8x8, ki predstavlja polja labirinta. Labirint bo v delu zaslona velikosti 512x512 točk in bo sestavljen iz polj velikosti 64x64.

```
signal lab: slika8x8:= (
 "01011100",
 "01000100",
 "00010000",
 "00111100",
 "00000001",
 "10000001",
 "10001001",
 "00111100" );
```

a. V **sistem.vhd** deklariraj 8-bitni nepredznačeni vektor **vrst_lab** in napiši stavek, ki prebere eno vrstico labirinta. Pri branju bomo uporabili le tri bite koordinate **y**:

```
vrst_lab <= lab(to_integer(y(8 downto 6)));
```

b. Deklariraj nov enobitni signal **color1** in zapiši pogoj, ki določa kdaj se točka nahaja v bloku labirinta:

```
if y<512 and x<512 and vrst_lab(to_integer(not x(8 downto 6)))='1' then
   color1 <= '1';
   -- dodaj pogoje za trk
else
   color1 <= '0';
```

c. Popravi pogoje za vrednost izhodne barve **rgb**, tako da bo barva bela ("111111"), kadar je signal color1 na '1'. Prevedi sistem in preizkusi na razvojni plošči !

d. Deklariraj štiri enobitne signale **ztrk, strk, ltrk** in **dtrk,** ki bodo določali ali je prišlo do trka zgornjega, spodnjega, levega ali desnega roba žoge z belim poljem labirinta. Signali za trk naj se postavijo v primeru, ko sta **x** in **y** znotraj belega polja (kjer postavimo `color1 <= '1'`) in je v tem območju tudi eden izmed robov. Npr. zgornji rob:

```
color1 <= '1';
-- dodaj pogoje za trk
  if x>=xp and x<=xp+30 and y=yp then
     ztrk <= '1';
  end if;
```

Signale uporabimo za omejitev pomikanja žogice, npr. za omejitev pomika navzgor, če je prišlo do trka na zgornjem delu:

```
if x=800 and y=600 then
  if t(0)='1' and yp>=4 and ztrk='0' then
    yp<= yp - 4;
  end if;
  ztrk <= '0';
```