4. vaja: Igra

Grafičnemu sistemu s procesorjem bomo dodali še eno sličico, ki jo bo krmilil program in naredili preprosto igro.

4.1 Nova sličica

a. Naredi logiko za opis še ene sličice velikosti 30 x 30 točk. Deklariraj signal in obliko za novo sličico v paketu **sprites.vhd** (tako, kot je narejena sličica žoge).

b. V datoteki **sistem.vhd** deklariraj: 30-bitni vektor za vrstico (**vrst2**), 12-bitni koordinati (**x2, y2**) in enobitni signal za barvo (**color2**).
   - napiši prireditveni stavek za **vrst2**, ki je podoben stavku **vrst**
   - v sinhronem procesu določi barvo **color2**, podobno, kot je določena **color**
   - pri določanju izhodnega signala **rgb** dodaj pogoj za **color2**, tako da bo takrat na izhodu nova barva, npr.

```
elsif color2='1' then
 rgb <= "110000";
```

   - poveži koordinati **x2** in **y2** z izhodi procesorskega sistema (v komponenti **proc** dodaj dva nova izhoda):

```
u2: proc port map (
      clk => clk,
      rst => key(0),
      pin => pin,
      pout1 => pout1,
      pout2 => x2,
      pout3 => y2
 );
```

c. Prevedi sistem in preizkusi prikaz slike na razvojni plošči.

4.2 Krmiljenje prikaza s procesorjem

a. S programom premaknimo sličico na koordinato: x2=15 in y2=30. Prenesi strojno kodo programa v datoteko **program.mif,**izvedi **Processing > Updade Memory Initialization file**, **Processing > Start > Start Assembler** in ponovno naloži kodo na razvojno ploščo.

```
        lda x2          00 : 105;
        outp 2          01 : 702;
        lda y2          02 : 106;
        outp 3          03 : 703;
zanka:  jmp zanka       04 : 404;
x2      db 15           05 : 00f;
y2      db 30           06 : 01e;
```

b. V **sistem.vhd** dodaj pogoj za prikaz sličice, ki ga bo določal procesor prek izhoda **pout1**. Izhod ima že spodnje 4 bite vezane na matrični prikazovalnik za prikaz števil, naslednji bit **pout1(4)** pa bi uporabili kot pogoj za prikaz sličice. Dodaj ta bit kot pogoj za določanje barve **color2**:

```
if x>=x2 and x<x2+30 and y>=y2 and y<y2+30 and pout1(4)='1' then
   color2 <= vrst2(to_integer(x-x2));
```

c. Dopolni program tako, da bo nastavil izhod **pout1** na vrednost 16 in tako prikazal sličico.

```
          lda slik
          outp 1
          ...
slik      db 16
```

4.3 Zaznavanje trka in točkovanje

a. V sistemu deklariraj nov enobitni signal **trk**, ki naj se postavi na '1', ko pride do trka med žogo in sličico. Pogoj za postavitev signala je, da sta hkrati **color** in **color2** postavljena na '1'. Signal **trk** naj ohranja vrednost '1', dokler je ne resetiramo s pritiskom na tipko **key(0)**, zato naj bo napisan v sinhronem procesu (znotraj pogoja za fronto ure).

b. Za preizkus pravilnosti delovanja trka bomo ob trku spremenili barvo ozadja zaslona. Popravi tisti del kode, ki določa barvo ozadja. Kadar ni trka (trk='0') naj bo barva modra "101011", sicer pa pa siva "101010":

```
else
  if trk='0' then
     rgb <= "101011";
  else
     rgb <= "101010";
  end if;
end if;
```

c. Prevedi kodo in preizkusi delovanje signala za zaznavanje trka na razvojni plošči. Ko pride do trka se mora ozadje spremeniti, ko se žoga odmakne in pritisnemo tipko S1, pa dobimo spet modro ozadje.

d. Signal za zaznavanje trka poveži z vhodom procesorja s prireditvenim stavkom:

```
pin <= x"00" & "000" & trk;
```

e. Dodaj med priključke procesorja v datoteki **cpu.vhd** enobitni izhodni signal **rd_o**. Dopolni opis procesorja tako, da bo ob ukazu za branje **(inp)** nastavil izhod **rd_o** na '1' (ne pozabi ga tudi postaviti na '0') in poveži ta signal prek **proc.vhd** v **sistem.vhd**.

Sedaj lahko v sistemu namesto tipke **key(0)** uporabiš signal **rd_o** za resetiranje trka.

f. Dopolni program, da bo zaznal trk in ob tem izbrisal sličico ter povečal rezultat, ki se prikaže na matričnem prikazovalniku ob vpisu na izhod **pout1** (zbirniški ukaz `outp 1`).