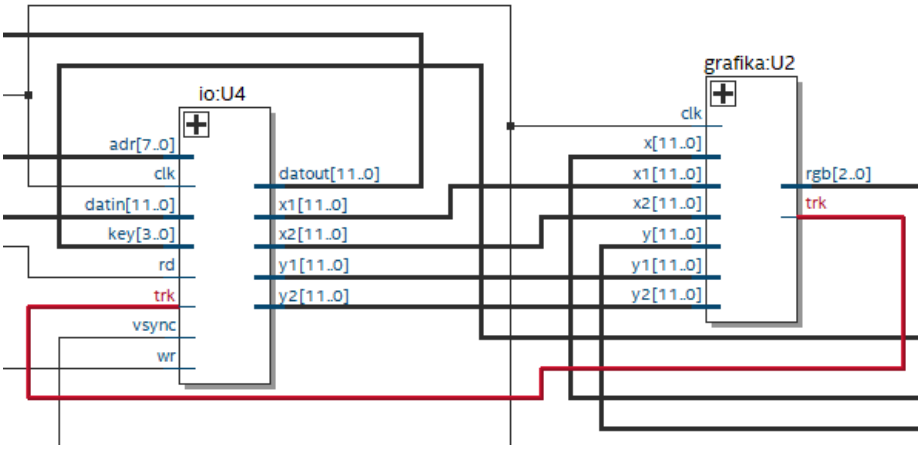
12. vaja: Grafični sistem s procesorjem (2)

Naredili bomo logiko za zaznavanje tipk in trka med dvema sličicama.



Sistem

V glavni datoteki sistem.vhd naredi nekaj novih povezav.

- Deklariraj dva 12-bitna vektorja **x2** in **y2** za določanje koordinat druge sličice in naredi v stavkih **port map** povezavo s komponento **io** in **grafika.**
- Deklariraj enobitni signal **trk** in ga poveži na komponenti **io** in **grafika.**
- Na komponento **io** poveži vektor stanja tipk (key => keys).

Grafika

Med priključke datoteke grafika.vhd dodaj vhoda **x2** in **y2** ter izhod trk. Dopolni logiko prikazovanja sličice, tako da bo prikazana druga sličica kroga na koordinatah **x2** in **y2**. Izhod **trk** naj se postavi na '1', ko je trenutna koordinata (**x**, **y**) znotraj obeh krogov, sicer pa naj ima vrednost 0.

Vhodno-izhodni vmesnik

Dopolni vhodno-izhodni vmesnik io.vhd z logiko za določanje koordinat drugega kroga (**x2**, **y2**), branjem tipk **key** in signala **trk**.

- Med priključke dodaj signale: **x2**, **y2**, **key** in **trk**.
- Logika vhodno-izhodnega vmesnika naj naredi registre na naslovih:

adr	izhodni register	adr	vhodni register
0	x1	0	int (vsync)
1	y1	1	key(0)
2	x2	2	key(1)
3	y2	3	key(2)
		4	key(3)
		5	int2 (trk)

- Vsaka tipka je vezana na svoj vhodni register zaradi enostavnejšega testiranja stanja v zbirniškem programu (naredimo le skok **jze**).
- Za signal trk naredi podobno prekinitveno logiko kot za vsync.

Program

Preizkusi delovanje grafičnega sistema s programom, ki premika krog levo in desno (glede na key(0) in key(1)) in ob trku z drugim krogom prestavi drugi krog: prog12.asm

```
start: lda x1
       outp 0
       lda y1
       outp 1
       lda x2
       outp 2
       lda y2
       outp 3
pavza: inp 0
       jze pavza
       inp 5
       jze tipke
       lda 100   ; trk = 1, x2 = 100 - x2
       sbt x2
       sta x2
       jmp start
tipke: inp 1
       jze tipke2
       jmp levo   ; key(0)=1
tipke2:inp 4
       jze pavza
       lda x1     ; key(3)=1, x1++
       add 1
       sta x1
       jmp start
levo:  lda x1     ; key(0)=1, x1--
       sbt 1
       sta x1
       jmp start
x1     db 5
y1     db 10
x2     db 0
y2     db 15
```

```
0=> lda & x"1d",
1=> outp & x"00",
2=> lda & x"1e",
3=> outp & x"01",
4=> lda & x"1f",
5=> outp & x"02",
6=> lda & x"20",
7=> outp & x"03",
8=> inp & x"00",
9=> jze & x"08",
10=> inp & x"05",
11=> jze & x"10",
12=> lda & x"21",
13=> sbt & x"1f",
14=> sta & x"1f",
15=> jmp & x"00",
16=> inp & x"01",
17=> jze & x"13",
18=> jmp & x"19",
19=> inp & x"04",
20=> jze & x"08",
21=> lda & x"1d",
22=> add & x"22",
23=> sta & x"1d",
24=> jmp & x"00",
25=> lda & x"1d",
26=> sbt & x"22",
27=> sta & x"1d",
28=> jmp & x"00",
29=> x"005",
30=> x"00a",
31=> x"000",
32=> x"00f",
33=> x"064",
34=> x"001",
others => x"000"
```

Program lahko prevedemo tudi s prevajalnikom CPU-C, ki podpira del nabora ukazov jezika C. prog12.c

```
int* px1 = 0;
int* py1 = 1;
int* px2 = 2;
int* py2 = 3;
int* sync = 0;
int* key0 = 1;
int* key3 = 4;
int* trk = 5;
int x1 = 20;
int y1 = 100;
int x2 = 0;
main()
{
 while(1) {
  *px1 = x1; *py1 = y1; *px2 = x2; *py2 = 110;
  if (*sync==1) {
    if (*trk==1)  { x2=100-x2; }
    if (*key0==1) { x1--; }
    if (*key3==1) { x1++; }
  }
 }
}
```