Fakulteta za elektrotehniko, Univerza v Ljubljani
Program: VS Aplikativna elektrotehnika, smer: Avtomatika

VGA igra: Escape angry blocks

Luka Levac

Poročilo projekta pri predmetu Načrtovanje digitalnih elektronskih sistemov

Mentor: prof. dr. Andrej Trost

Ljubljana, 2019

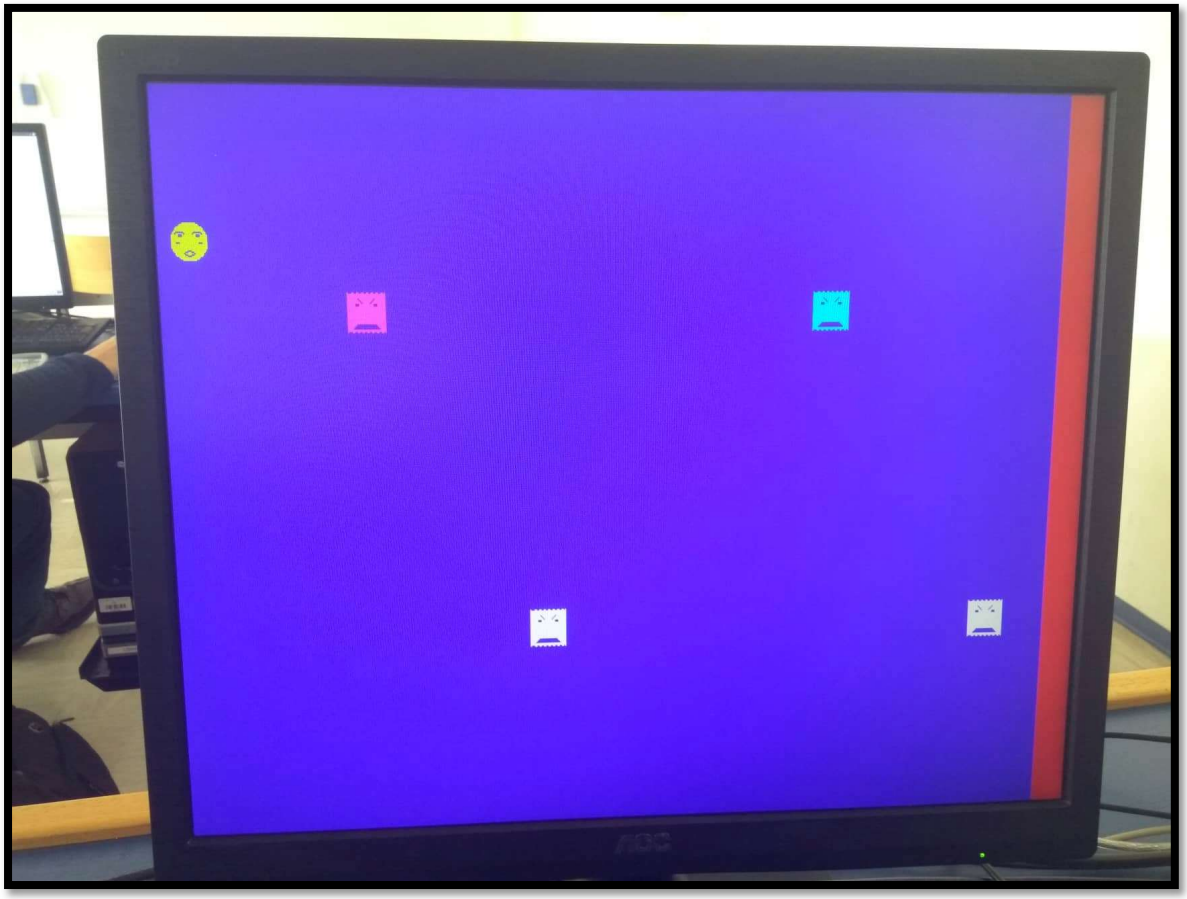
Vsebina

Povzetek projekta

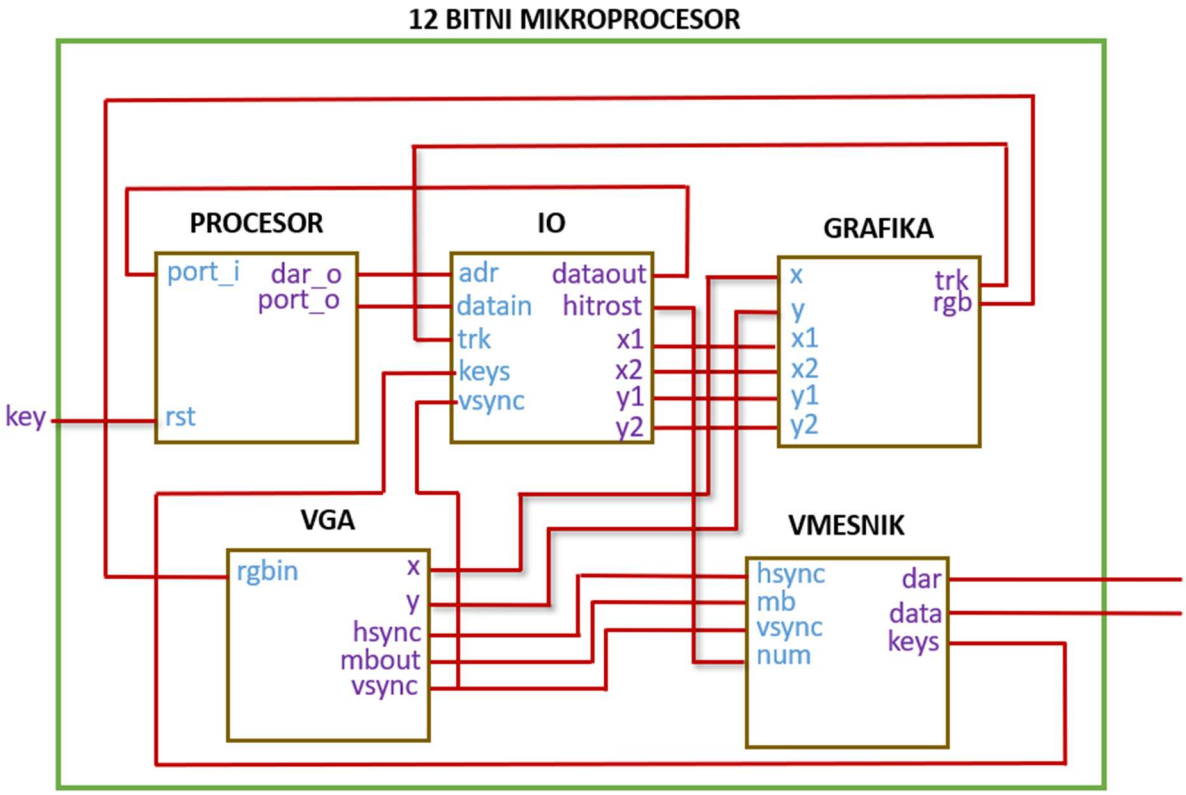
Tekom laboratorijskih vaj smo izdelali 12 bitni mikroprocesor, ki je bil sposoben izvajati preproste strojne ukaze iz pomnilnika in na koncu tudi prikazovati like na računalniškem monitorju.

Po zaključenih laboratorijskih vajah smo študentje po svojih lastnih željah projekt nadgradili za potrebe svoje video igre.

Moja končna igra je na ekranu izgledala tako:



Delovanje 12-bitnega procesorja

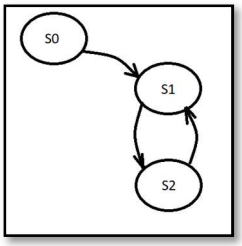


Mikroprocesor je sestavljen iz 5 medsebojno povezanih blokov, ki delujejo kot ena celota.

KRMILNA ENOTA

Procesor oz. krmilna enota je enota v kateri v kateri so shranjene »definicije« strojnih ukazov in hkrati enota ki skrbi za osnovne korake izvajanja teh ukazov (beri ukaz iz pomnilnika, povečuj programskih števec, izvedi ukaz…). Imamo tri osnovna stanja:

- S0, začetno stanje (reset): prog. Števec na 0 in napreduje na S1.
- S1, zajemi: ukaz iz pomnilnika se shrani in programski števec se nastavi. Nekaj izjem med strojnimi ukazi se izvede v tem stanju (ukazi s katerimi skočimo na nek drugi del programa).
- S2, izvedi: podatek iz pomnilnika se prebere in ukaz se shrani.



IO

Vhodno-izhodna enota. Povezana je z vsemi bloki/enotami:

- Od vmesnika prejema podatke kot so npr. stanje tipk na razvojni ploščici.
- Iz grafike prejema podatke o trku.
    - Te podatke preko porta *dataout* posreduje krmilni enoti katera potem glede na shranjen program izvede operacije/dogodke na podlagi teh podatkov.
- Na vhod *datain* procesor pošilja 12 bitne podatke. Iz teh podatkov IO enota »izlušči«:
    - vrednosti koordinat x1, y1, x2, y2
        - Te koordinate nam podajajo trenutno pozicijo žogice oz. jeznih kock na ekranu. Te koordinate IO enota potem preko enako poimenovanih izhodov pošlje v grafiko, ki te elemente izriše na računalniški monitor.

- hitrost
  - Spremenljivka, ki je potrebna za spreminjanje hitrosti pomikanja kock na ekranu (hitrost se spremeni vsakič ko igralec pride ko konca ekrana). Ta spremenljivka se potem preko izhoda *hitrost* prenese v vmesnik na led matriko, ki glede na trenutno hitrost prikazuje različno sličico.
- Na naslednji način beremo podatke:

```
if rising_edge(clk) then
-----------------------------------------
 if (wr = '1') and (adr = 0) then
    x1 <= datin;
 elsif (wr = '1') and (adr = 1) then
    y1 <= datin;
```

VMESNIK

Enota, ki bere stanje tipk na razvojni ploščici in prikazuje sličice na led matriki. Te enote osebno nisem veliko spreminjal. Spremenil sem le sličice za led matriko in namesto, da se na matriki prikazujejo vrednosti tipk, sem na matriko povezal spremenljivko hitrost, ki podaja hitrost pomikanja kock na ekranu:

```
);
-----------------------------------------
num <= hitrost;    -- prikazuj hitrost
```

GRAFIKA

Enota, ki skrbi za prikaz likov na ekranu.

- Iz IO enote prejema vektorje x1, x2, y1, in y2, katere odšteje od vektorjev x in y, katere prejme iz enote VGA, da vidi če so koordinate trenutno znotraj koordinat likov. Če so, potem se liki izrišejo na ekran

```
-----------------
xt1 <= x-x1;
yt1 <= y-y1;
-----------------
xt2 <= x-x2;
yt2 <= y-y2;
-----------------
```

- Like izriše na naslednji način:

```
if ((xt1 < 32)) and (yt1 < 32) then

  adr <= yt1(4 downto 0) & xt1(4 downto 0);

  data_sig <= rom(to_integer(adr));
  data <= data_sig;

  if (data_sig = '1') then
     rgb <= "110";
  elsif (data_sig = '0') then
     rgb <= "001";
  end if;
```

  - Primerja se z št. 32 zato, ker so liki sestavljeni iz arrayev 32x32.
  - Liki so tudi dveh barv. Glede na to ali je '1' ali '0' na trenutni vrednosti arraya se like izriše drugačne barve.
- V grafiki se preverja tudi ali je prišlo do trka dveh elementov. Preverja se samo, če se koordinate dveh likov »križajo«:

```
if (xt1<32 and yt1<32 and xt2<32 and yt2<32) or (xt1<32
   trk <= '1';
else
   trk <= '0';
end if;
```

- V svojem projektu sem dodal tudi štiri kocke, ki se pomikajo gor in dol po ekranu. Dve se pomikata dol in dve istočasno navzgor. To sem naredil preko enega para koordinat (x2, y2), ki jih grafika prejema iz IO enote. Vrednost teh koordinat se spreminja v procesorju (povečujejo se dokler ne dosežejo roba ekrana, potem se ponovno začnejo povečevati od začetne vrednosti in to se ponavlja, dokler žogica ne pride do cilja). Pomikanje vseh štirih na enkrat preko enega para koordinat, sem storil na naslednji način:

```
xt2 <= x-x2;
yt2 <= y-y2;
-----------------
x3 <= x2 + 400;
xt3 <= x - x3;
yt3 <= yt2;
-----------------
y4 <= 570 - y2; --da gre od spodaj navzgor
x4 <= x2 + 150;

xt4 <= x - x4;
yt4 <= y - y4;
-----------------
x5 <= x2 + 550;

xt5 <= x - x5;
yt5 <=  yt4;
-----------------
```

  - x-om sem dodajal ali odšteval vrednosti, zato, da so kocke razporejene po ekranu.
  - Pri zadnjih dveh kockah pa sem spremenil tudi y. Da so se kocke lahko pomikale v obratno smer, sem od prvotne vrednosti y odštel 570 (pribl. vrednost slikovnih znakov po y osi ekrana).
- Kocke imajo tudi svojo array znakov za drugačen videz od žogice.
- V CPU programu sem pomikanje izvedel tako:

```
lda y2
add hitrost
sta y2
sbt 600
jze reset1
```

  - Y2 se vsakič poveča za '*hitrost*' in ko doseže število 600 se izvede podprogram reset1:

```
reset1: lda y2
        sbt 600
        sta y2
        jmp start
```

  - Kjer se y ponastavi na 0.
- Prav tako sem omejil lik 1 tako, da ko pride do konca ekrana po x-osi, se ponastavi nazaj na začetek ekrana in vrednost hitrosti se poveča:

```
lda x1
sbt 780
jze reset2
```

```
reset2: lda 0
        sta x1
        lda hitrost
        add 2
        sta hitrost
        lda 0
        sta y2
        jmp start
```

- x1 gre nazaj na prvotno vrednost, hitrost se poveča za 2 in y2 (kocke) gre prav tako nazaj na začetno vrednost.

VGA

Enota, ki generira signale vsync, hsysnc, mbout ter x in y. Signal vysinc se proži 72x na sekundo (vsakič, ko se osveži slika na monitorju).

Sinteza vezja

Flow Status	Successful - Wed Jan 15 17:41:49 2020
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	sistem
Top-level Entity Name	sistem
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	878 / 22,320 (4 %)
Total registers	325
Total pins	14 / 154 (9 %)
Total virtual pins	0
Total memory bits	3,072 / 608,256 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

- Število uporabljenih logičnih elementov: 878 od 22320 (4%).
- Število vseh uporabljenih registrov: 325.
- Število uporabljenih pinov: 14 / 154 (9%).
- Uporabljen spomin: 3071 bitov od 608.256 bitov (manj kot 1%).

Delovanje igre

Cilj igre je, da se žogica z obrazom prebije iz leve strani ekrana do desne strani ekrana mimo vseh kock, ki jo poskušajo uničiti. Ko žogice pride do konca, »napreduje« na naslednjo stopnjo, kjer se kocke hitreje pomikajo gor in dol. Vsakič, ko pride žoga do konca, se kocke pričnejo hitreje pomikati. Glede na trenutno hitrost je na led matriki prikazana drugačna sličica.