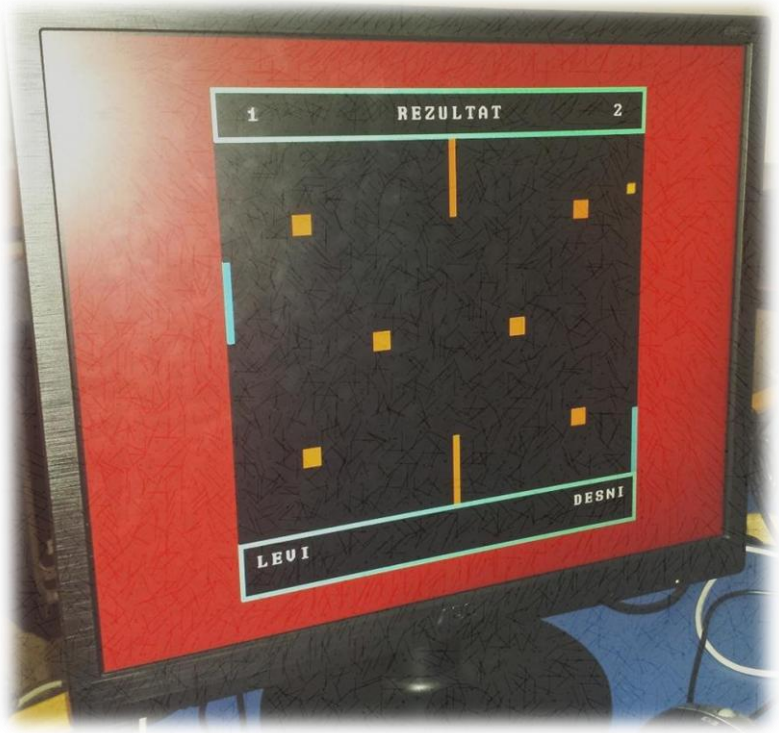
Univerza v Ljubljani
Fakulteta za elektrotehniko



IGRA PONG S TEŽAVNOSTMI

PROJEKT PRI PREDMETU DIGITALNA INTEGRIRANA VEZJA IN SISTEMI

Ljubljana, 10.2.2015 Adnan Tiganj

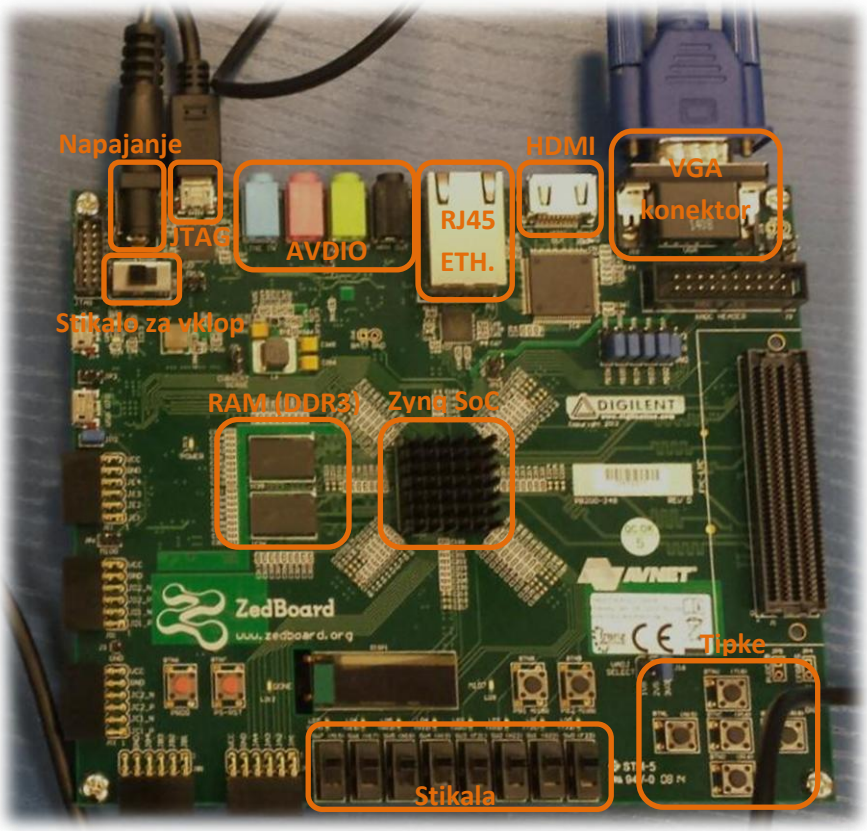
KAZALO VSEBINE

KAZALO SLIK

1. UVOD

1.1 HW – ZedBoard

Na laboratorijskih vajah pri predmetu Digitalna integrirana vezja in sistemi smo se spoznali z razvojno ploščo ZedBoard, ki temelji na *Xilinx Zynq®-7000 All Programmable SoC XC7Z020-CLG484-1* vezju. Zynq je torej t.i. *System on Chip* vezje, ki sestoji iz programirljive logike ter dveh procesorskih jedr Cortex-A9.
Na sliki 1 lahko vidimo, da je ZedBoard konkretno opremljen (512MB DDR3 RAM, gigabitni Ethernet, JTAG, veliko I/O enot, ...).



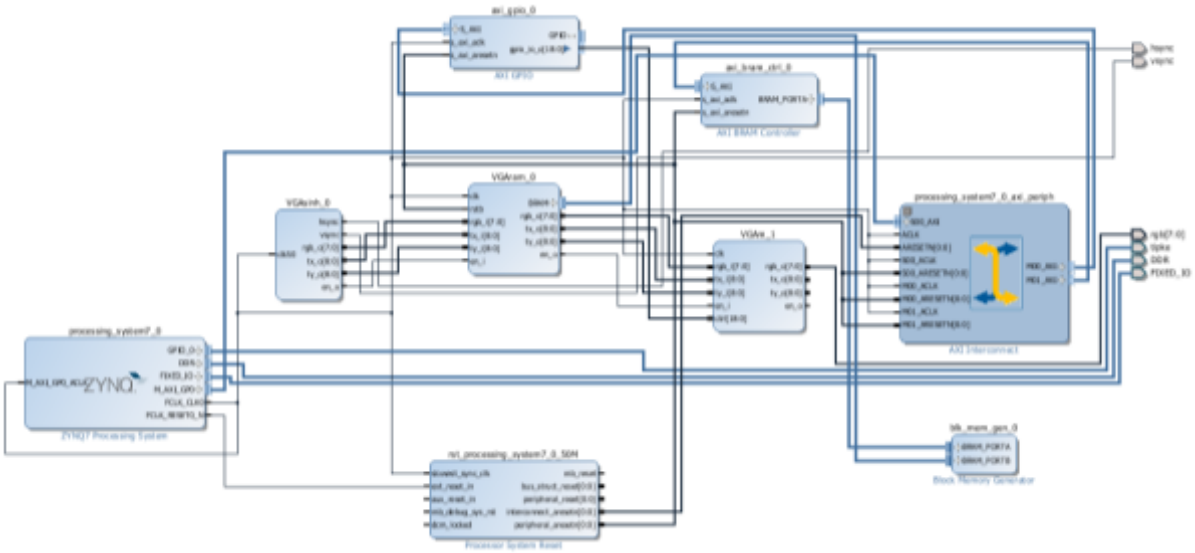
Slika 1: Razvojna plošča ZedBoard z imenom osnovnih gradnikov.

1.2 Grafični sistem

Z uporabo programskega orodja Xilinx Vivado in jezika VHDL smo opisali celoten grafični sistem. Sprva smo tipke povezali na paralelna vrata procesorja GPIO. Nato smo se lotili pisanja prve grafične komponente, ki generira sinhronizacijske signale za prikazovanje slike po standardu VGA ter izdelali IP komponento VGAsinh.

Sledila je izdelava druge grafične komponente, ki nam omogoča prikazovanje gibljive sličice ter izdelava njene IP komponente VGAm.

Tretjo grafično komponento smo izdelali za prikaz slike iz notranjega pomnilnika BRAM. Tudi tokrat smo naredili IP komponento - VGAram. Vse IP komponente smo povezali v končni sistem z vsemi IP bloki, ki je viden na sliki 2. Sledil je preizkus risanja točk (pikslov) na monitorju.



Slika 2: Končni izgled sistema z vsemi IP bloki.

Rezultat dela na laboratorijskih vajah je grafični sistem kateremu smo nato dodali še aplikacijo (grafično igro).

2. IGRA PONG

2.1 Opis

Zaključni del laboratorijskih vaj je predstavljal projekt v katerem smo napisali preprosto aplikacijo za naš grafični sistem. Aplikacijo smo napisali v C jeziku z uporabo SDK (*Software Development Kit*) orodja, ki je del Vivada. Osnovo pri pisanju kode sta nam predstavljali na vajah napisani funkciji *write* in *clear*. S prvo smo risali točke (piksle) na monitorju, z drugo pa hitro brisali BRAM. Aplikacijo smo na koncu preko orodja SDK sprogramirali na ZedBoard ter jo zagnali.

Osnovna ideja igre izhaja iz polovice prejšnjega stoletja. Nekaj let kasneje pa je izšla prva različica. Igra je postala popularna šele v začetku 70ih let, ko jo je podjetje Atari začelo vgrajevati v svoje igralne avtomate in poslalo na trg. Njena popularnost je pripeljala do tega, da so jo podjetja za izdelovanje igralnih konzol posnemali in izdelali svojo različico. Igro zasledimo pod različnimi imeni: *Tennis for two*, *Ping pong*, *Pong* (Atari), itd. Vsekakor pa gre za prvo komercialno uspešno igro.

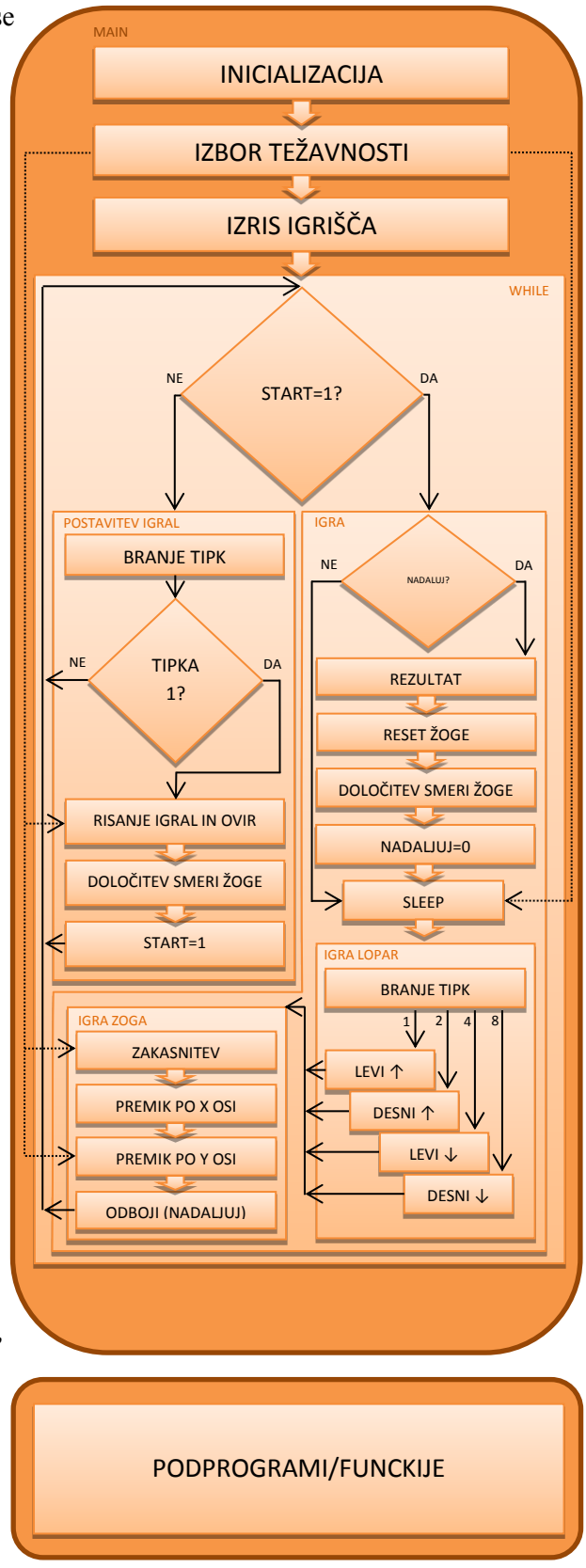
Igra z dvema loparjema in žogo postane sčasoma dolgočasna zato je bila za razvedrilo dodana izbira težavnosti kjer lahko izbiramo med različnimi hitrostmi premikanja žoge in loparjev ali pa dodamo različne ovire na igrišče.

V nadaljevanju sledi podrobnejši opis delovanja igre.

2.2 Delovanje

Na desni je prikazan *flow chart* igre. Znotraj *main* funkcije se sprva izvede inicializacija spremenljivk. Nato se v delu kode kjer izberemo težavnost sprva izpiše pozdrav »PONG GAME – IZBERI TEZAVNOST« (slika 3), po določenem času se napis izbriše in sledi branje tipk s katerimi izberemo težavnost igre tako, da vplivamo na spremenljivke za določanje hitrosti premika loparjev, hitrosti premika žoge in postavitve ovir. Na ekranu se nam izpiše izbrana težavnost (slika 4). S pritiskom na desno tipko izberemo normalni način igre (brez ovir ter z normalno hitrostjo premika loparjev in žoge). S pritiskom na zgornjo tipko izberemo »SPEEDY« težavnost brez ovir kjer sta hitrosti loparjev in žoge primerno hitrejše napram normalni težavnosti. S pritiskom na levo tipko izberemo »SUPER SPEED« težavnost brez ovir kjer sta hitrosti loparjev in žoge maksimalni. S pritiskom na spodnjo tipko izberemo težavnost s sredinskimi ovirami (zgoraj in spodaj). Hitrost premikanja loparjev in žoge je enaka kot pri normalni težavnosti. S pritiskom zgornje in spodnje tipke hkrati pa izberemo »ULTIMATE« težavnost kjer sta aktivirani obe vrsti ovir – sredinski oviri ter ovire v obliki kock, ki so razporejene po celem igrišču. Igra je malenkost hitrejša. V primeru, da nobena tipka ni pritisnjena, se uporabi normalna težavnost.

Sledi izris igrišča kjer s *for* zanko znotraj *for* zanke po x in y osi rišemo zgornji in spodnji okvir, ki hkrati z eno od stranic predstavljata zid za odboj žoge. V zgornjem okviru s številko izpišemo rezultat za oba igralca ter napis »REZULTAT«. V spodnjem pa ime igralca (»LEVI« in »DESNI«).



Šele v *while* zanki se igra dejansko izvaja. Sprva pogledamo vrednost spremenljivke *start*. Ob začetku igre je spremenljivka *start* vedno enaka 0, kar pomeni, da se izvede del kode »POSTAVITEV IGRAL«. V tem delu sprva beremo tipke dokler ni pritisnjena desna tipka (tipka 1). Ko je pritisnjena, izvedemo del kode za risanje igral in ovir, če so le te aktivirane. Nato se naključno (*rand()* funkcija) določi smer premikanja žoge po x in y osi (vrednosti so lahko samo 1 ali 2; če je smer premikanja po eni osi enaka 0, se žoga premika samo po (eni) osi, ki ni enaka 0, kar pa ni

zaželjeno). Za konec postavimo spremenljivko *start* na 1, kar pomeni, da bomo v naslednjem krogu *while* zanke prešli na del kode »IGRA«, kjer lahko končno začnemo igrati.

V delu kode »IGRA« sprva pogledamo, če je spremenljivka *nadaljuj* enaka 1, kar bi pomenilo, da je enemu od igralcev med igranjem žoga zletela izven igrišča (*out*). Torej v primeru, da se izvede del kode »NADALJUJ«, se sprva v podprogramu *stetje* ustreznemu igralcu izpiše povečan rezultat (v primeru, da eden od igralcev prvi pride do rezultata 9, se izpiše kdo je zmagal (slika 5), igra se zaključi in po nekaj sekundah zakasnitve, se igra resetira, kar pomeni, da se vrnemo na začetek *main* funkcije). Nato se žoga postavi na začetno pozicijo (sredina igrišča), igrala in ovire se osvežijo (izbris in ponoven izris). Žogi se zopet naključno določi smer premikanja. Spremenljivko *nadaljuj* postavimo na 0.

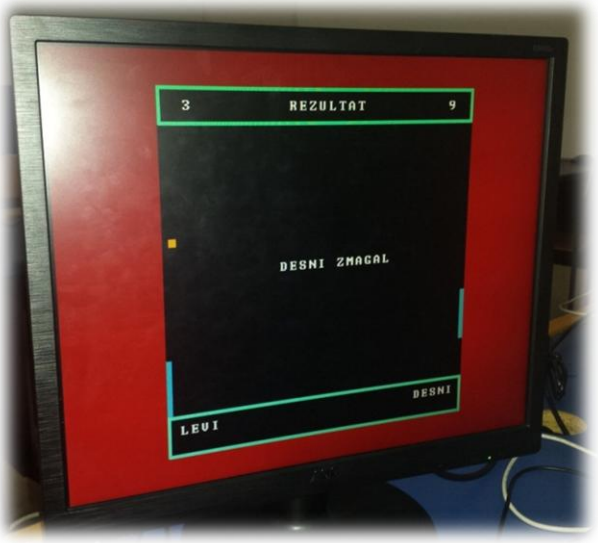
Spremenljivka je na začetku igre vedno enaka 0, kar pomeni, da del kode »NADALJUJ« preskočimo. Sledi zakasnitev za določanje hitrost igre (*sleep*), na katero vplivamo neposredno iz izbora težavnosti igre. Nato v delu kode »IGRA LOPAR« beremo tipke in ob pritisku na določeno tipko premaknemo ustrezni lopar v ustrezno smer. Ob pritisku na desno tipko (tipka 1) premaknemo levi lopar gor, ob pritisku na zgornjo tipko (tipka 2) premaknemo desni lopar gor, z levo tipko (tipka 4) premaknemo levi lopar dol in s spodnjo tipko (tipka 8) premaknemo desni lopar dol.

V delu kode »IGRA ŽOGA« sprva z zakasnitvijo na katero neposredno vplivamo iz izbora težavnosti igre, določimo hitrost premikanja žoge. Nato glede na hitrost premika žogice po x osi premaknemo žogico v tej smeri in enako za premik žogice po y osi.

Na koncu se izvede del kode »ODBOJI«, ki spremlja položaj žoge po x in y osi, ter glede na to in na položaj loparjev, stene in ovir (če so le te aktivirane) določi smer odboja. V primeru, da žoga zleti mimo loparja v *out*, se poveča rezultat nasprotnemu igralcu, spremenljivka *nadaljuj* pa se postavi na 1 tako, da se v naslednjem krogu zanke izvede omenjeni del kode »NADALJUJ«.

Slika 3: Napis ob začetku igre kjer izberemo težavnost.

Slika 4: Izpis izbrane težavnosti (primer Ultimate).



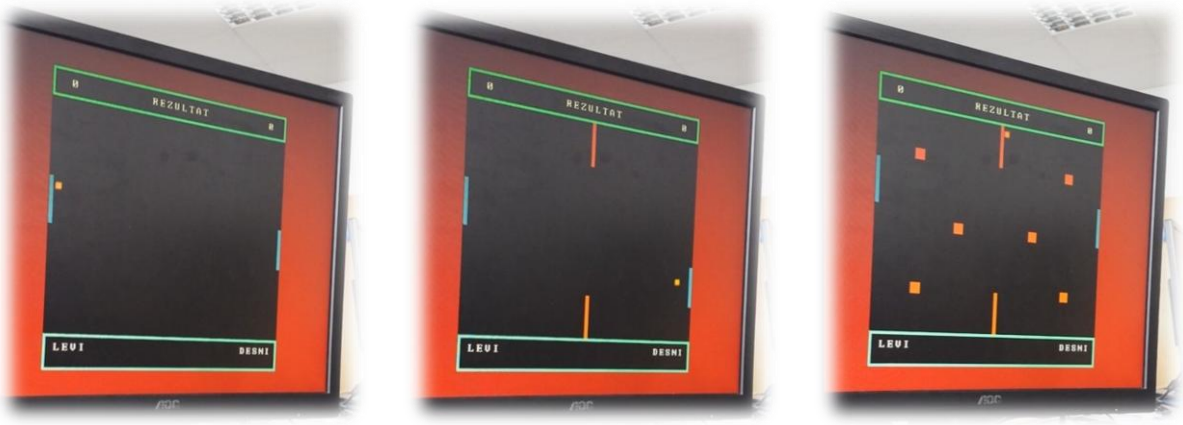
Slika 5: Konec igre ter izpis zmagovalca.

2.3 Igrišča

V omenjenih petih težavnostih se pojavijo tri različna igrišča. Prvo igrišče, ki je prikazano na sliki 6 (levo), se pojavi v prvih treh težavnostih, saj se razlikujejo le v hitrosti premikanja žoge in loparjev.

Pri drugem igrišču, ki je prikazano na sliki 6 (sredina), se pojavijo sredinske ovire. Igrišče je uporabljeno v težavnosti »ovire«.

Tretje igrišče, ki se pojavi v zadnji težavnosti tj. »ultimate«, je sestavljeno s sredinskimi ovirami in s kockastimi ovirami, ki so razporejene po celem igrišču (slika 6, desno).



Slika 6: Normalno igrišče (levo), igrišče s sredinskimi ovirami (sredina) ter igrišče s sredinskimi in kockastimi ovirami (desno).

2.4 Izseki kode

Izbira težavnosti:

```
tipke = XGpioPs_Read(&gp, 2); //PREBERE TIPKE ZA IZBOR TEZAVNOSTI
switch(tipke)
{
		case 1: //DESNO - NORMAL
		{
			hitrost_l=3000; //POSTAVI NOVO HITROST IGRE
			hitrost_s=4; //POSTAVI NOVO HITROST ZOGICE
			izbor(1); //IZPISE IZBRANO TEZAVNOST
			sleep(2); //POCAKA 2 SEKUNDI
			clear(); //POBRISE
		break;
		}
		case 2: //GOR - SPEEDY
		{
			hitrost_l=2500;
			hitrost_s=3;
			izbor(2);
			sleep(2);
			clear();
		break;
		}
		case 4: //LEVO - SUPER SPEED
		{
			hitrost_l=1500;
			hitrost_s=3;
			izbor(3);
			sleep(2);
			clear();
		break;
		}
		case 8: //DOL - OVIRA
		{
			hitrost_l=2500;
			hitrost_s=4;
			ovira=1; //AKTIVIRA OVIRI NA SREDINI
			izbor(4);
			sleep(2);
			clear();
		break;
		}
		case 10: //GOR-DOL - ULTIMATE
		{
			hitrost_l=2500;
			hitrost_s=4;
			ovira=1; //AKTIVIRA OVIRI NA SREDINI
			ovira_k=1; //AKTIVIRA KOCKASTE OVIRE
			izbor(5);
			sleep(2);
			clear();
		break;
		}
		default: //CE NI NOBENA TIPKA PRITISNJENA
		{
			hitrost_l=3000;
			hitrost_s=4;
			izbor(1);
			sleep(2);
			clear();
		break;
		}
}
```

Odboji žoge (le nekaj primerov za x in y os)

```
switch (premik_x) //POGLEDAMO ZA ODBOJE GLEDE NA POZICIJO
SKOKICE PO X OSI
{
        case 14: //ODBOJ OD LEVEGA LOPARJA
        {
        if(premik_y > (levi_y-48) && premik_y < (levi_y+48))
        //ODBOJ OD LOPARJA
                {
                smer_x=1; //ODBOJ V DESNO
                }
                else //OUT
                {
                rezultat_d++;
                nadaljuj=1;
                }
        break;
        }

        case 497: //ODBOJ OD DESNEGA LOPARJA
        {
        if(premik_y > (desni_y-48) && premik_y < (desni_y+48))
        //ODBOJ OD LOPARJA
                {
                smer_x=2; //ODBOJ V LEVO
                }
                else //OUT
                {
                rezultat_l++;
                nadaljuj=1;
                }
        break;
        }
        case 268: //ODBOJ OD SREDINSKE OVIRE Z DESNE STRANI
        {
        if(ovira==1 && premik_y > (ovira_y_1-42) && premik_y <
                (ovira_y_1+42)) //ODBOJ OD ZGORNJE OVIRE
                {
                smer_x=1; //ODBOJ V DESNO
                }
        if(ovira==1 && premik_y > (ovira_y_2-42) && premik_y <
                (ovira_y_2+42)) //ODBOJ OD SPODNJE OVIRE
                {
                smer_x=1; //ODBOJ V DESNO
                }
        break;
        }
}
```

```
switch(premik_y) //POGLEDAMO ZA ODBOJE GLEDE NA POZICIJO
SKOKICE PO Y OSI
{
        case 113: //ODBOJ Z ZGORNJE STRANI OD ZGORNJIH OVIR
        {
        if (ovira_k==1 && premik_x > (ovira_x_1-12) && premik_x <
                (ovira_x_1+12)) //LEVA
                {
                smer_y=1;
                }
        if (ovira_k==1 && premik_x > (ovira_x_4-12) && premik_x <
                (ovira_x_4+12)) //DESNA
                {
                smer_y=1;
                }
        break;
        }
        case 137: //ODBOJ S SPODNJE STRANI OD ZGORNJE OVIRE
        {
        if(ovira==1 && premik_x > 246 && premik_x < 270)
                {
                smer_y=2;
                }
        break;
        }
        case 364: //ODBOJ Z ZGORNJE STRANI OD SPODNJE OVIRE
        {
        if(ovira==1 && premik_x > 246 && premik_x < 270)
                {
                smer_y=1;
                }
        break;
        }
        case 385: //ODBOJ S SPODNJE STRANI OD SPODNJIH OVIR
        {
        if (ovira_k==1 && premik_x > (ovira_x_1-12) && premik_x <
                 (ovira_x_1+12)) //LEVA
                {
                smer_y=2;
                }
        if (ovira_k==1 && premik_x > (ovira_x_4-12) && premik_x <
                 (ovira_x_4+12)) //DESNA
                {
                smer_y=2;
                }
        break;
        }
}
```

Risanje loparjev

```
void lopar(char lopar, int risi_y)
{
int i,j;
int b = 50; //BARVA LOPARJA
//RISANJE LOPARJA
for(i=0; i<80; i++)
{
        for(j=0; j<8; j++)
        {
                switch(lopar)
                {
                case 0: //LEVI LOPAR
                {
                        write(j, risi_y+i-40, b);
                break;
                }
                case 1: //DESNI LOPAR
                {
                        write(504+j, risi_y+i-40, b);
                break;
                }
                }
                }
        }
}
```

Risanje kockastih ovir

```
void ovira_kocka(char brisanje, int brisi_x, int brisi_y, int risi_x, int risi_y)
{
int b = 200; //BARVA KOCKE
int i,j;
if(brisanje==1) //BRISE OVIRO
{
        for(i=0; i<20; i++)
        {
                for(j=0; j<20; j++)
                {
                        write(brisi_x+i-10, brisi_y+j-10, 0);
                }
        }
}

for(i=0; i<20; i++) //RISE OVIRO
{
        for(j=0; j<20; j++)
        {
                write(risi_x+i-10, risi_y+j-10, b);
        }
}
}
```

3. ZAKLJUČEK

Za konec je potrebno dodati, da smo aplikacijo uspešno implementirali v naš grafični sistem. Hkrati pa je potrebno poudariti, da obstaja veliko možnih nadgradenj.

Na sistemskem nivoju bi lahko vključili avdio vhodno-izhodno komponento in tako igri dodali zvokovne učinke. Glede na popularnost HDMI standarda bi lahko namesto VGA krmilnika realizirali kar HDMI krmilnik. Vključili bi lahko tudi premična stikala s katerimi bi lahko kar med igro spreminjali nastavitve, ipd.

Na aplikacijskem nivoju pa obstaja neskončno opcij za dodelavo. Za začetek bi bilo smiselno narediti ovire premične, kar bi še otežilo igro. Lahko bi med igro ob vsakem odboju spremenili vrednost spremenljivke za barvo žoge. Hitrost in velikost žoge bi se lahko spreminjali ob vsakem uspešnem odboju od loparja. To je le nekaj idej za dodelavo aplikacije.

4. LITERATURA IN VIRI

[1] Spletna stran Digitalna integrirana vezja in sistemi. Stran dostopna 9.2.2015.
http://lniv.fe.uni-lj.si/div.html - Digitalna integrirana vezja in sistemi

[2] Spletna stran Xilinx, *Zynq-7000 All Programmable SoC Overview.* Stran dostopna 9.2.2015.
http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[3] Spletna stran Xilinx, *Zynq-7000 All Programmable SoC*. Spletna stran dostopna 9.2.2015.
http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[4] Spletna stran ZedBoard, *ZedBoard*. Stran dostopna 9.2.2015.
http://zedboard.org/product/zedboard - Spletna stran ZedBoard

[5] Spletna stran Wikipedia, *Pong.* Stran dostopna 9.2.2015.
http://sl.wikipedia.org/wiki/Pong

[6] Spletna stran PongGame, *Pong Game.* Stran dostopna 9.2.2015.
http://www.ponggame.org/

[7] Spletna stran Aaroncox, *Paddle Battle.* Stran dostopna 9.2.2015.
http://www.aaroncox.net/tutorials/arcade/PaddleBattle.html

[8] Spletna stran Botskool, *Ping Pong Game in C Language.* Stran dostopna 9.2.2015.
http://www.botskool.com/tutorials/c-tutorials/ping-pong-game-c-language