Digitalna integrirana vezja in sistemi

Šolsko leto 2015/2016

Projekt pri laboratorijskih vajah

POROČILO PROJEKTA

Labirint

Jan Markič

64110025

Ljubljana 30.1.2016

Povzetek

Za projekt pri laboratorijskih vajah sem naredili digitalni sistem, ki izrisuje labirint na VGA monitor in omogoča premikanje kvadrata po labirintu z uporabo tipk na FPGA boardu. V želji, da bi sistem deloval kot nekakšna igra z nekim smislom sem dodal še cilj (kvadrat), kateremu se spremeni pozicija ko ga dosežemo. Celoten sistem je sestavljen iz več blokov. Za osnovo sem vzel šesto laboratorijsko vajo in dodal svoj blok v katerem sem opravil računanje in določanje rgb izhoda. V tem poročilo bom opisal izvedbo svoje komponente LABIRINT in prikazal simulacije bloka. Podal bom tudi podatke o zasedenosti vezja in predstavil shemo celotnega sistema skupaj s programom v jeziku C.

Komponenta Labirint

Komponenta labirint določa *rgb* glede na vhode *rgbin*, *xy*, *en* in podatke, ki jih dobi preko AXI vodila. Izhod *rgb* pa dejansko predstavlja barvo točke na VGA monitorju. Slika 1 prikazuje komponento Labirint z vsemi vhodnimi in izhodnimi signali.

Slika 1: Komponenta Labirint

V komponenti sem opisal proces, ki najprej določi koordinati x in y premikajočega kvadrata (xp_s in yp_s) in cilja (xp_sc in yp_sc), katere dobi preko AXI vodila. Signala x in y sta koordinati točke, kateri želimo določiti barvo, določil sem jih iz vhodnega vektorja xy. Koordinate grejo od -255 do 255 in zato so vse signed 9 bitni vektorji. Na sliki 2 je prikazana določitev koordinat, katere potrebujemo pri določanju rgb izhoda.

```
x<=xy(17 downto 9);
y<=xy(8 downto 0);
x_s<=signed(x);
y_s<=signed(y);

yp<=xypr(8 downto 0);
yp_s<=signed(yp);
xp<=xypr(17 downto 9);
xp_s<=signed(xp);

ypc<=xyc(8 downto 0);
yp_sc<=signed(ypc);
xpc<=xyc(17 downto 9);
xp_sc<=signed(xpc);
```

Slika 2: Izsek kode za določitev koordinat.

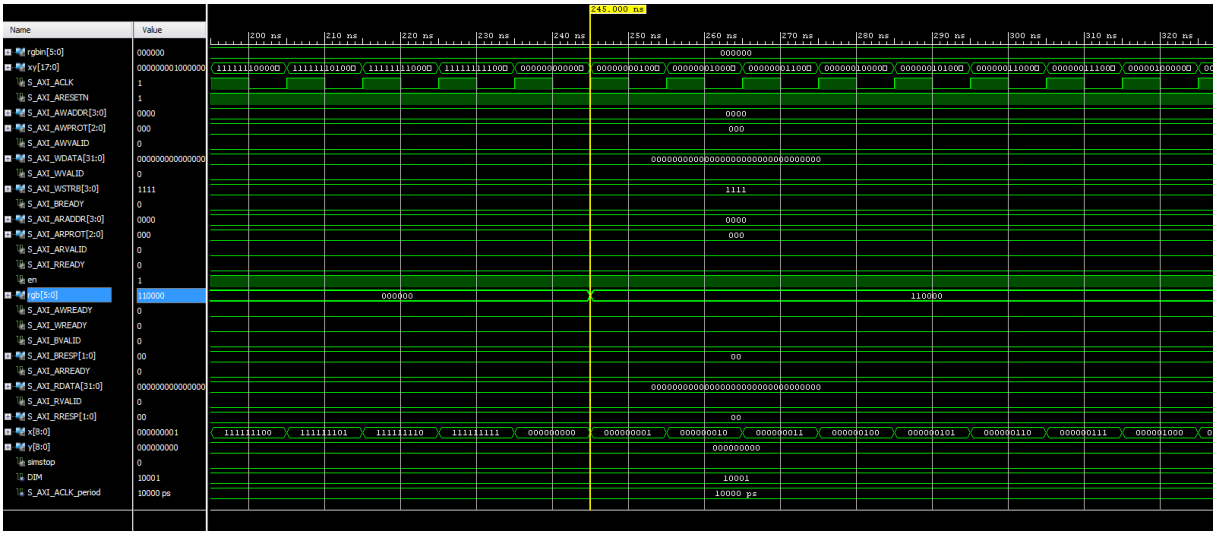
Polmer kroga ki omejuje vidno polje, pa je določen kot notranja konstanta. V mojem primeru je polmer nastavljen na 32.

Glavni del komponente pa je *if* stavek v katerem se določa izhod rgb. V prvem delu procesa se izračunajo koordinate kroga x2 in y2. Središče kroga je enako koordinati premikajočega kvadrata (16x16 točk), katerega premikamo s tipkami, s tem da prištejem 8, da se pomaknem v središče. V drugem delu procesa pa se preveri če je točka xy na ekranu (en = '1') in če je točka del premikajočega ali ciljnega kvadrata. V tem primeru dodeli izhodu rgb vrednost »110000« kar predstavlja rdečo barvo. Če točka ne ustreza zgornjim pogojem se preveri ali se nahaja znotraj kroga (x2 + y2 < r2). Če točka ustreza temu pogoju prevzame izhodni signal rgb kar vrednost vhodnega signala rgbin, kateri pride iz VGA_RAM komponente. V RAM-u so shranjene barve za vsako točko na zaslonu (celoten labirint), komponenta VGA_RAM pa priskrbi moji komponenti xy koordinate in rgb signal za vsako točko. Če koordinata xy ne ustreza nobenemu zgornjemu pogoju, izhod rgb postavi na »000000« (črna barva). Koda celotnega procesa je prikazana na sliki 3.

```
p1: process(s0_axi_aclk)
    begin
        if rising_edge(s0_axi_aclk) then

           x2 <= signed(x_s - (xp_s+8)) * signed(x_s - (xp_s+8));
           y2 <= signed(y_s - (yp_s+8)) * signed(y_s - (yp_s+8));

           if (((x_s >= xp_s) and (x_s <= (xp_s+15)) and (y_s >= yp_s) and (y_s <= (yp_s+15)) and en='1' ) or
              ((x_s >= xp_sc) and (x_s <= (xp_sc+15)) and (y_s >= yp_sc) and (y_s <= (yp_sc+15)) and en='1'))
              then
              rgb <= "110000";
              else
                  if (en = '1' and (x2 + y2 <= r2)) then
                      rgb <= rgbin;
                      else rgb <= "000000";
                  end if;
           end if;
        end if;
    end process;
```

Slika 3: Koda procesa.

Delovanje komponente Labirint sem nato še simuliral. Uporabili sem testno strukturo test_krog.vhd ki sem jo malenkost spremenil za moje potrebe. Izsek iz simulacije za prvo vrstico je podan na sliki 4.



Slika 4: Izsek iz simulacije za prvo vrstico.

X koordinata se spreminja od -17 do 17 in nato spet isto v novi vrstici (y) vse do 18 vrstice. Začetek kvadrata sem v testni strukturi nastavil na (0,0). Testna struktura ustvari tudi datoteko slika.txt, kjer se vidi celoten izrisan kvadrat 16*16. Iz simulacije lahko vidimo, da se izhod rgb res postavi na 110000 v območju kvadrata.

Na sliki 5 je prikazana datoteka slika.txt ki jo ustvari testna struktura. Za rgb enak 000000 nariše –, za ostale vrednosti pa *.

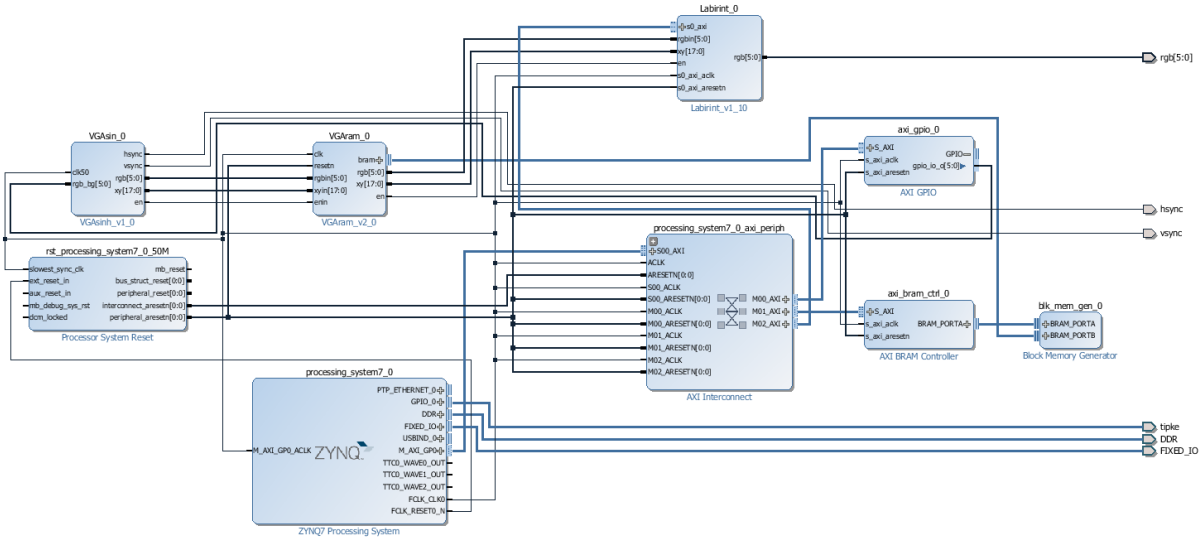
Slika 5: Datoteka slika.txt.

Pod Utilisation Report sem si ogledal tudi zasedenost vezja. Tabela je podana spodaj:

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	290	0	53200	0.55
LUT as Logic	290	0	53200	0.55
LUT as Memory	0	0	17400	0.00
Slice Registers	145	0	106400	0.14
Register as Flip Flop	145	0	106400	0.14
Register as Latch	0	0	106400	0.00
F7 Muxes	0	0	26600	0.00
F8 Muxes	0	0	13300	0.00

Celotni sistem

Komponento Labirint sem umestil v celotni sitem . Shema celotnega sistema je podana na sliki 5.



Slika 5: Celotni sistem.

Celotni sistem je sestavljen iz procesorskega dela, kjer se tudi generira ura sistema (clk=50Mhz) in ostali kontrolni signali. Del tega je tudi blok, ki predstavlja sistemski reset, periferni bloki AXI vodila kateri dostopajo preko AXI vodila do registrov in RAM pomnilnika kjer se shranjujejo vrednosti barve posameznih točk. Sistem vsebuje še VGAsinh blok kateri ima funkcijo VGA krmilnika, ki poskrbi za hsync in vsync signale. Ta pa je nato preko komponente VGAram povezan na mojo komponento Labirint. Tako ima naš sistem izhode hsync, vsync in rgb, ki so potrebni za izris slike na VGA monitorju.

Procesor izvaja program, napisan v jeziku C, kateri v začetku iz tabele/arraya preko AXI vodila napiše barve točk labirinta v RAM nato pa preverja katere tipke so bile pritisnjene. Glede na to katera tipka je bila pritisnjena premakne kvadrat v določeno smer v primeru, da tam ni stene labirinta. Ko sta položaja premikajočega kvadrata in cilja enaka, se koordinate cilja naključno premaknejo drugam v labirint. Seveda pa program ves čas pošilja koordinate preko AXI vodila do komponente Labirint, katera poskrbi za izris objektov na VGA monitorju.

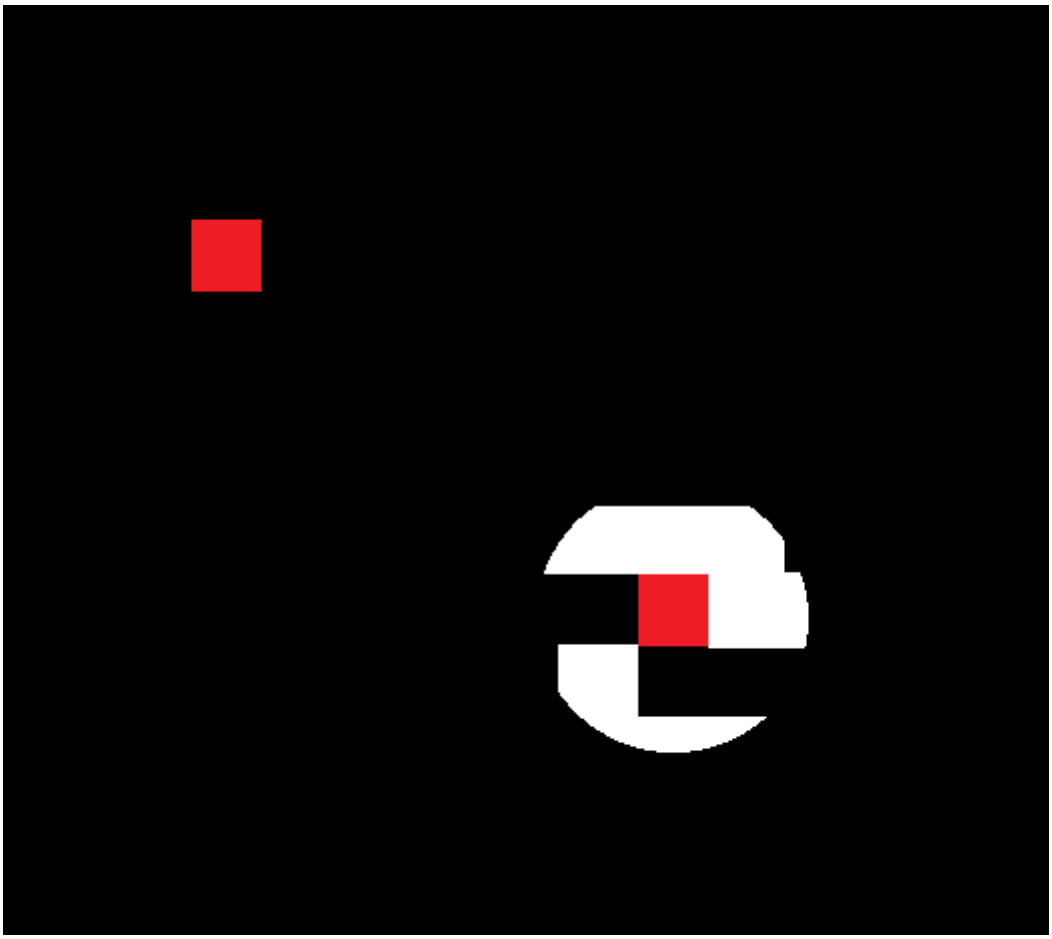
Program v jeziku C je podan spodaj skupaj s komentarji kode:

```
#include <stdio.h>
#include <stdlib.h>
#include "platform.h"
#include "xil_io.h"// neposreden dostop do registrov
#include "xgpiops.h"// gonilnik za PS GPIO (tipke)
#include "sleep.h"

char *p;
void pix (int x, int y);

void pix (int x, int y)
{
   p= (char *) (XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + (y<<9)+ x);
   *p =0x3f;
}

int main()
{
        int n=0;
        int v = 512;
        int t;

        int xp=0;
        int yp=0;

        int xk=0;
        int yk=0;
        int xycilj=0;
        int xc=0;
        int yc=0;


        int xy=0;

        int i;
        int j;

        int labi[32][32] = {
                            {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
                            {1,0,1,1,1,1,0,0,0,0,0,0,0,1,1,1,0,0,0,1,0,0,0,1,0,0,0,0,0,1},
                            {1,0,1,0,0,1,1,0,1,1,0,1,1,1,0,0,0,1,0,0,0,1,0,1,0,1,1,1,1,0,1},
                            {1,0,1,1,0,0,1,0,1,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,1,0,1,0,0,0,0,1},
                            {1,0,0,1,1,0,1,0,0,0,1,1,0,1,0,1,0,0,1,0,0,0,0,0,1,1,1,0,1,1,1,1},
                            {1,1,0,0,0,0,0,0,1,1,1,0,0,0,0,1,1,0,1,0,1,1,1,1,0,1,0,0,0,0,0,1},

                            …

                            };


        XGpioPs gp;
        XGpioPs_Config *gpCfg;

        // inicializacija PS GPIO
        gpCfg = XGpioPs_LookupConfig(XPAR_PS7_GPIO_0_DEVICE_ID);
        XGpioPs_CfgInitialize(&gp, gpCfg, gpCfg->BaseAddr);

        //prepise ves ram z 0 izbrise sliko
        for (n=0; n < 512*v; n++)
             {
             p= (char *) (XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + n);
             *p =0x00;
             }
```

```
//narise labirint oz. napise vrednosti v RAM
for(i=0;i<32;i++){
                for(j=0;j<32;j++){
                            if (labi[i][j] == 1){
                                    for(xk=0;xk<16;xk++){
                                        for(yk=0;yk<16;yk++){
                                            pix(yk+j*16,xk+i*16);
                                        }

                                    }

                            }

                }
            }



while (1)
{
    usleep(100000);

     t = XGpioPs_Read(&gp, 2); // branje stanja tipk, ki so na banki 2

    if(t==1)
    {
        if(labi[16+yp][16+xp+1]==0)
        {
        xp++;

        }
    }

    if (t==8)
    {
        if(labi[16+yp+1][16+xp]==0)
        {
        yp++;
        }
    }

    if (t==4)
    {
        if(labi[16+yp][16+xp-1]==0)
        {
        xp--;
        }
    }

    if (t==2)
    {
        if(labi[16+yp-1][16+xp]==0)
        {
        yp--;
        }
    }




    if (yp>=0)
        {
        xy=(16*xp)<<9;
        xy+=16*yp;
        }
    else {
            xy=(16*xp)<<9;
            xy=xy+16*yp+512;
         }
```

```
            if(xycilj==xy)
            {
                do{
                  yc=(rand() % 31)-15;
                  xc=(rand() % 31)-15;
                  if (yc>=0)
                  {
                  xycilj=(16*xc)<<9;
                  xycilj+=16*yc;
                  }
                  else {
                  xycilj=(16*xc)<<9;
                  xycilj=xycilj+16*yc+512;
                  }
                }while(labi[yc+16][xc+16]==1);
            }

            //polje koordinate komponenti Labirint
            Xil_Out32(XPAR_LABIRINT_0_BASEADDR,xy);
            Xil_Out32(XPAR_LABIRINT_0_BASEADDR+4,xycilj);

        }
    return 0;
}
```

Slika na VGA monitorju nato izgleda tako:



Levi kvadrat je cilj desni kvadrat pa premikamo s tipkami na FPGA boardu.