Projekt

Utrka na 100m s prikazom na monitoru

#### SADRŽAJ

[1. Uvod 1](#_Toc442721296)

[2. Opis izrade sklopovskog rješenja 2](#_Toc442721297)

[2.1. VGA kontroler 3](#_Toc442721298)

[3. Opis izrade programskog rješenja 6](#_Toc442721299)

[3.1. Iscrtavanje igrača 6](#_Toc442721300)

[3.2. Kretanje igrača 8](#_Toc442721301)

[3.3. Prepoznavanje pobjednika 11](#_Toc442721302)

[4. Zaključak 15](#_Toc442721303)

[Literatura 16](#_Toc442721304)

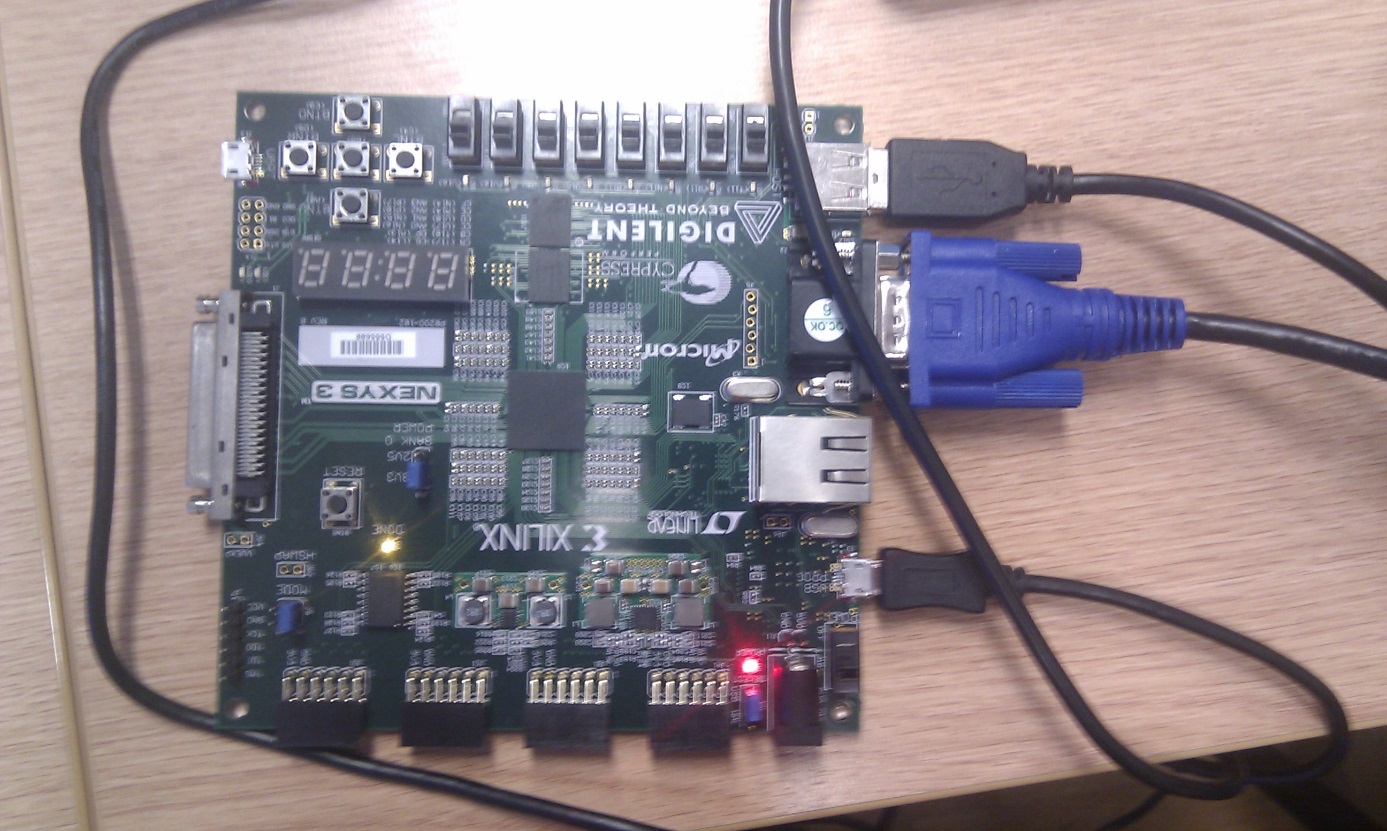
# Uvod

Cilj ovog rada je upoznavanje programskog paketa *Xilinx platform studio(XPS)* koji služi za dizajniranje računalnih sustava kroz izradu određenog zadatka. Računalni sustav koji će biti dizajniran zasnovan je na *MicroBlaze* procesoru.

U projektnom zadatku potrebno je dizajnirati računalni sustav zasnovan na spomenutom procesoru koji simulira utrku na 100 metara za 1 do 4 igrača. Unos od strane korisnika potrebno je realizirati korištenjem tipkovnice ili tipkala razvojnog sustava, a tijek utrke je potrebno prikazivati na monitoru. Zadatak je izveden izradom vlastitog VGA kontrolera u navedenom programskom paketu te izradom logike simulacije utrke u VHDL-u. Prepoznavanje pritisnute tipke te slanje podataka u registre vlastitog kontrolera obavljeno je u programskom jeziku C. Kao dodatak izvornom zadatku, implementirano je prepoznavanje pobjednika.

# Opis izrade sklopovskog rješenja

U sklopovskom dijelu korišten je *Nexys3* razvojni sustav [1]. *Nexys3* je sustav baziran na Spartan 6 FPGA. Među ostalim, komponente su mu 16 MB RAM-a, USB port za programiranje i slanje podataka, USB port za UART komunikaciju i USB port za miš ili tipkovnicu, 8-bitni VGA port, 8 LED pokazivača, 5 tipkala, 8 sklopki te 4 7-segmentna pokaznika. Na slici 1. je vidljivo koji su portovi iskorišteni u ovom projektu.



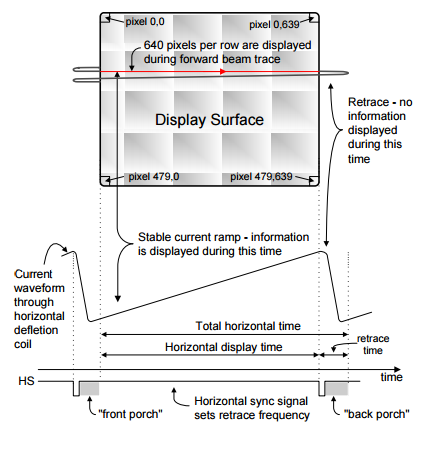
Slika 1: Nexys3 razvojni sustav sa korištenim portovima

Sustav dizajniran u programskom paketu *Xilinx Platform Studio* [2],[3], u kojem je moguće odabrati koji dijelovi sustava i s kojim svojstvima će se koristiti. Korišten je *microblaze* procesor lokalne memorije 16KB, PLB sabirnica za povezivanje perifernih modula s procesorom, generator takta frekvencije 25Mhz, RS232 Uart (koji je korišten za komunikaciju procesora s testnim prozorom u *Tera Term*-u), PS2\_Mouse\_Keyboard kontroler za eksternu tipkovnicu, DIP sklopke te tipke koji na kraju nisu korišteni, no sustav je načinjen s njima zbog početne nejasnoće izvedbe logike simulacijske utrke. Posebna komponenta je korisnički dizajniran VGA kontroler.

## VGA kontroler

VGA je standardno sučelje za kontrolu analognih monitora. VGA port na *Nexys3* sustavu se sastoji od 8-bitne boje tipa RGB (3 bita za crvenu boju, 3 bita za zelenu boju i 2 bita za plavu boju) te dva standardna sinkronizacijska signala (HSync i VSync). Time je osiguran prikaz 256 različitih boja. VGA standard seže do uporabe starih ekrana s katodnim cijevima, gdje je ekran bio pogođen snopovima elektrona i tako je nastajala slika. Postojala su tri snopa elektrona na CRT ekranima, za svaku boju po jedan. Isti princip prenio se i na LCD ekrane, iako se više ne koriste snopovi elektrona kao u CRT tehnologiji. Informacija u vidu boje na ekranu se mogla prikazati jedino ako se snop elektrona kretao u pozitivnom smjeru po ekranu, što podrazumijeva s lijeva na desno i od gore prema dolje, a za vrijeme dok se snop vraća u gornji lijevi kut ništa se ne prikazuje. Zbog toga postoji mreža po kojoj se snopovi elektrona kreću, a koja je veća od samog prikaza kojeg vidi korisnik. Ona koristi da bi se izvršila sinkronizacija pulseva horizontalnog i vertikalnog signala.

Prostor za sinkronizaciju koji se nalazi ispred prikaza na ekranu naziva se „front porch“, dok se isti iza prikaza na ekranu naziva „back porch“. Ako se snop elektrona nalazi na bilo kojem od *porch*-eva, informacija neće biti prikazana na ekran. Zbog već spomenutog principa rada koji se prenio na LCD ekrane, iako fizički ne rade na isti način kao ekrani s katodnim cijevima, na ista se svojstva mora obratiti pažnja pri rukovanju s korisnički definiranim VGA kontrolerom unutar *Xilinx*-a. Zato je u svakom slučaju gdje se definirala grafička logika uzimao u „front“ i „back porch“. „Front porch“ za HSync iznosi 144 piksela, a za VSync 39 piksela. „Back porch“ za HSync iznosi 12 piksela, dok za VSync iznosi 6 piksela. VGA prikaz na monitoru prikazan je na slici 2.



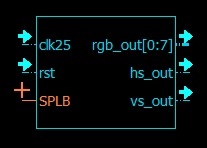
Slika 2: VGA prikaz na monitoru

VGA kontroler izrađen za realizaciju ovog projekta dizajniran je pomoću Create or Import Peripheral čarobnjaka unutar Xilinx Platform Studio programskog paketa. Njime je određen način spajanja kontrolera na procesor (PLB) te broj 32-bitnih registara za upotrebu (10). Zatim je dodan VGA port u .mpd datoteku, kako bi se kontroler mogao povezati s drugim, već dodanim komponentama, prikazano u programskom kodu 1.

|  |  |
| --- | --- |
| Linija | Kod |
| 39: | PORT clk25 = "", DIR = I |
| 41: | PORT rst = "", DIR = I |
| 43: | PORT rgb\_out = "", DIR = O, VEC = [0:7] |
| 45: | PORT hs\_out = "", DIR = O |
| 47: | PORT vs\_out = "", DIR = O |

Programski kod 1. Portovi

Dodana su dva ulazna porta, *clk25* (spojen na signal takta od 25Mhz) te *rst* (reset), i tri izlazna porta, *rgb\_out* (vektor od 8 članova za boje), *hs\_out* (horizontal sync) i *vs\_out* (vertical sync). Na sličan način dodani su portovi u generiranim HLD template datotekama *vga.vhd* te *user\_logic.vhd*. Shema kreiranog VGA kontrolera izgleda kao na slici 3.



Slika 3: Definirani portovi kreiranog VGA kontrolera.

Definiranim portovima dodani su pinovi u datoteci *system.ucf*, što je prikazano na sljedećem primjeru koda 2 .

Nakon dodjele pinova VGA kontroleru su dodijeljeni driveri i spojen je na PLB sabirnicu. Time je završeno hardversko rješenje.

|  |  |
| --- | --- |
| Linija | Kod |
| 78: | Net vga\_controller\_rgb\_pin<0> LOC=U7 | IOSTANDARD=LVCMOS33; |
| 79: | Net vga\_controller\_rgb\_pin<1> LOC=V7 | IOSTANDARD=LVCMOS33; |
| 80: | Net vga\_controller\_rgb\_pin<2> LOC=N7 | IOSTANDARD=LVCMOS33; |
| 81: | Net vga\_controller\_rgb\_pin<3> LOC=P8 | IOSTANDARD=LVCMOS33; |
| 82: | Net vga\_controller\_rgb\_pin<4> LOC=T6 | IOSTANDARD=LVCMOS33; |
| 83: | Net vga\_controller\_rgb\_pin<5> LOC=V6 | IOSTANDARD=LVCMOS33; |
| 84: | Net vga\_controller\_rgb\_pin<6> LOC=R7 | IOSTANDARD=LVCMOS33; |
| 85: | Net vga\_controller\_rgb\_pin<7> LOC=T7 | IOSTANDARD=LVCMOS33; |
| 86: |  |
| 87: | Net vga\_controller\_hs\_pin LOC=N6 | IOSTANDARD=LVCMOS33; |
| 88: | Net vga\_controller\_vs\_pin LOC=P7 | IOSTANDARD=LVCMOS33; |
| 89: |  |
| 90: | Net vga\_controller\_0\_rst\_pin LOC=C9 | IOSTANDARD=LVCMOS33; |

Programski kod 2. Dodani pinovi u datoteci system.ucf.

# Opis izrade programskog rješenja

Programsko rješenje obuhvaća programski kod u programskom jeziku C i kod u opisnom jeziku VHDL. U datoteci *user\_logic.vhd* u VHDL-u su određeni boja pozadine, izgled trkaće staze, izgled trkača, ispis rednog broja pri dolasku na cilj te je opisana glavna logika glede kretanja trkača. Prilikom procesa stvaranja programskog rješenja bitan je segment pažljivo rukovanje registrima vlastite dizajnirane IP jezgre, to jest VGA kontrolera, koje podrazumijeva usklađivanje zapisivanja i čitanja iz registara, odnosno usklađivanje C koda te VHDL koda. Glavna ideja prilikom spomenute izrade jest detektirati izmjenično tipkanje dviju različitih tipki na tipkovnici, reagirati na izmjenu pritiska tipki tako što će se u odgovarajući registar zapisati određena vrijednost, pročitati vrijednost iz registra i upotrijebiti je u kodu koji opisuje kretanje igrača s lijeva na desno.

|  |  |
| --- | --- |
| Linija | Kod |
| 664: | location\_x1 <= slv\_reg1(22 to 31); |
| 665: | finish\_x1 <= slv\_reg2(29 to 31); |

Programski kod 3. Varijable za spremanje podataka iz registra.

U navedenim linijama koda 3 iz datoteke *user\_logic.vhd* definirane su varijable u koje će se spremati podaci koji se nađu u registrima, koji su definirani pri dizajniranju korisničke IP jezgre. Također, vidljivo je da nije korišten cijeli registar već samo određeni raspon, kolika je procjena da je potrebno s obzirom na veličinu podatka kojeg će pohraniti. Varijabla location\_x1 prima podatak iz registra slv\_reg1 i koristit će se pri određivanju trenutne horizontalne lokacije trkača, dok varijabla finish\_x1 prima podatak iz registra slv\_reg2 i koristit će se pri iscrtavanju konačnog poretka na kraju utrke.

## Iscrtavanje igrača

Utrka broji četiri trkača. Svaki trkač ima vlastitu stazu po kojoj trči. Budući da je korišten VGA kontroler s rezolucijom 640x480, preostaje 480 piksela za smještanje 4 trkaće staze. S četiri trkača, najjednostavnije je svakome dodijeliti prostor od 100 piksela za širinu staze, što ostavlja 80 neiskorištenih piksela, koji će se jednoliko raspodijeliti, tako da prva staza započne 40 piksela od gornjeg ruba ekrana, a četvrta staza završi 40 piksela od donjeg ruba ekrana. Kako bi prikaz utrke bio što realističniji s obzirom na grafičke mogućnosti razvojnog sustava, odlučeno je da se igrači prikažu kao 2D likovi iz profila. Svaki od četiri igrača je opisan u datoteci *user\_logic.vhd* metodom „piksel po piksel“. Pojedini igrač se sastoji od glave, vrata, trupa, ruku, nogu i stopala.

Priloženi dio koda 4 opisuje izgled glave trkača. Glava je široka 7 piksela, a visoka 9 piksela. Korištena je varijabla pod nazivom location\_x1. Vrijednost registra čija se vrijednost zapisuje u spomenutoj varijabli mijenja se svaki put kada igrač naizmjence pritisne odgovarajuće tipke za pomak trkača. Budući da se trkači pomiču isključivo s lijeva na desno, to jest po x-osi, samo pri određivanju horizontal counter-a je potrebno koristiti tu varijablu, dok u slučaju vertical counter-a nije. Ovaj će kod sve piksele u rasponu od 3 piksela lijevo i desno od trenutne lokacije trkača na x-osi pročitane iz registra slv\_reg1 te sve piksele u rasponu od 65. do 74. piksela po y-osi, brojeći od gornjeg ruba ekrana ispuniti crvenom bojom.

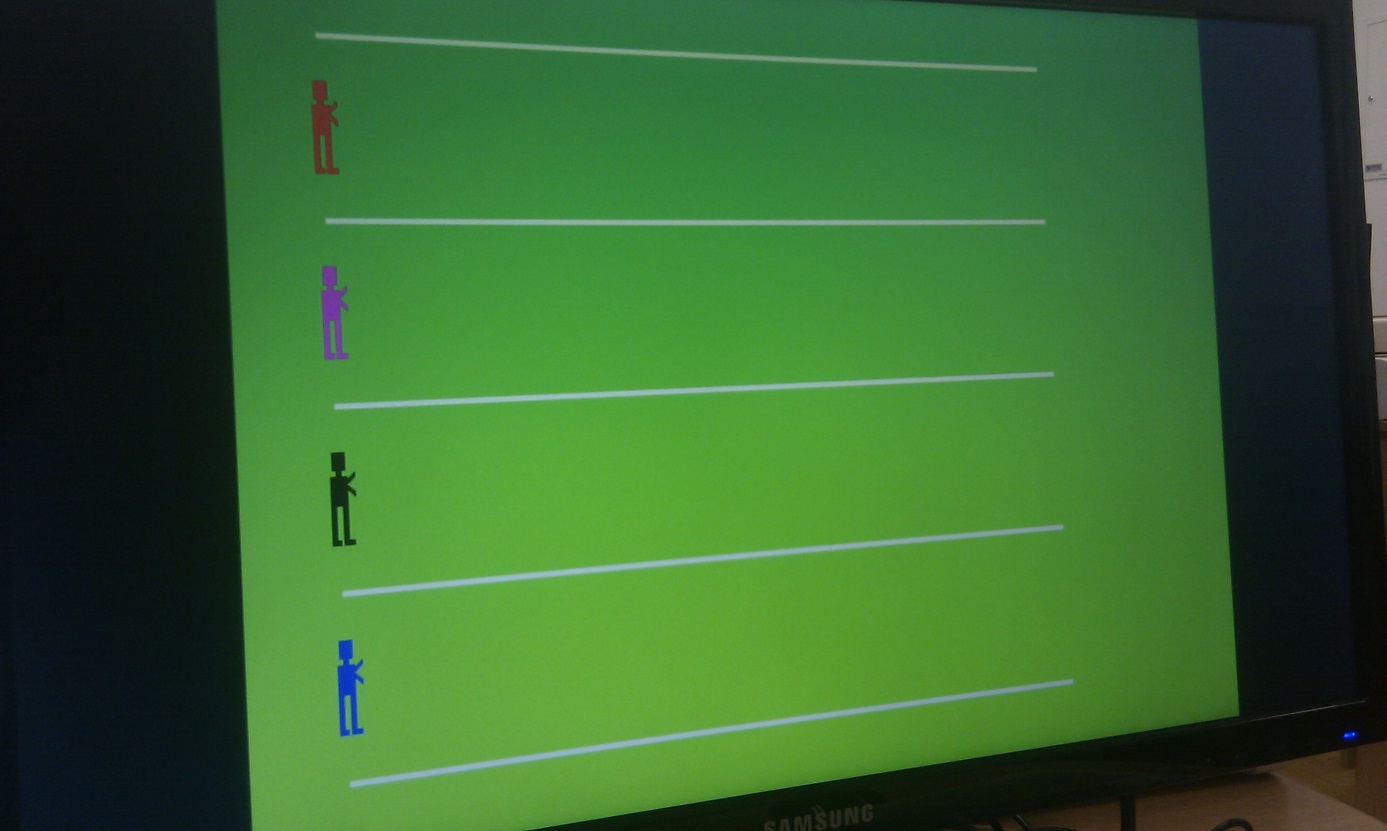
Na ekranu će se dobiti prikaz pravokutnika, koji predstavlja glavu virtualnog lika, sličnoj likovima iz video-igara sa starih igraćih konzola. Pri brojenju raspona piksela, vidljivo je da i u slučaju horizontalnog i vertikalnog prebrojavanja stoje određene konstante. Po x-osi potrebno je prvo prebrojiti 144 piksela da bi „glava“ VGA kontrolera došla do horizontalnog ruba ekrana, a po y-osi je potrebno prebrojiti 39 piksela kako bi ista došla do vertikalnog ruba ekrana. Potrebno je imati na umu obje vrijednosti pri iscrtavanju bilo kakvog oblika na ekran, jer su to vrijednosti definirane VGA standardom. Na isti način kako je na ekranu dobiven prikaz glave trkača, dobiveni su i prikazi ostalih dijelova tijela, kako bi virtualni lik barem podsjećao na stvarnog trkača. Razlike su jedino u rasponima piksela koji su obojani istom bojom, budući da su različiti dijelovi tijela na različitim pozicijama na ekranu, a i različitih dimenzija.

|  |  |
| --- | --- |
| Linija | Kod |
| 203: | if((horizontal\_counter >= 144 + location\_x1 - 3) and (horizontal\_counter <= 144 + location\_x1 + 3 ) and (vertical\_counter >= 39 + 90 - 25) and (vertical\_counter <= 39 + 90 - 16)) |
|  |  |
| ... |  |
| 226: | then |
| 228: | rgb\_out <= "11100000"; |

Programski kod 4. Iscrtavanje glave trkača.

Vrat je dimenzija 3x3 piksela, trup 9x15 piksela, noge 3x20 piksela, stopala 1x3 piksela te ruke 2x5 piksela. Sveukupno, trkač je širine 9, a visine 50 piksela. Iste dimenzije vrijede za svakog trkača, s razlikom u boji koja popunjava odgovarajući raspon piksela i u vertikalnoj poziciji, budući da su četiri trkaće trake postavljene na ekranu jedna ispod druge. Kako se horizontalna pozicija pojedinog trkača mijenja kroz igru, a vertikalna pozicija ostaje ista, za svakog je igrača određen „centar mase“, koji je korišten kao referentna točka za određivanje vertikalnog raspona u iscrtavanju pojedinih dijelova tijela.

U priloženom dijelu koda spomenuti „centar mase“ je na udaljenosti od 90 piksela, mjereno od gornjeg ruba ekrana. Ta je vrijednost dobivena tako da je uračunata vrijednost od 40 piksela, onoliko koliko je rub staze prvog trkača udaljen od gornjeg ruba ekrana te na nju dodana vrijednost od 50 piksela, što odgovara polovici vrijednosti širine pojedine trkaće staze. Kako se sredine susjednih staza međusobno razlikuju za 100 piksela, „centar mase“ sljedećeg trkača je uvećan za točno toliku vrijednost. Konačni izgled igrača pri početku utrke dan je na slici 4.



Slika 4: Izgled igrača i trkaće staze

## Kretanje igrača

Već pri opisnom kodu koji kao rezultat na ekran iscrtava virtualni lik trkača implementiran je dio koji definira kretanje igrača. Riječ je o već spomenutom registru koji pamti horizontalnu lokaciju igrača te varijabli koja prima lokaciju iz tog registra i koja se koristi u iscrtavanju lika. Međutim, prije nego se obrati pažnja na kretanje dizajniranih likova na ekranu, potrebno je prepoznati kombinaciju tipki koja pokreće trkača. Po dogovoru, to je izmjenično pritiskanje dviju različitih tipki. Postoji četiri trkača koji sudjeluju u utrci, stoga je potrebno definirati četiri različita para tipki, od kojih će svaki par biti vezan za jednog igrača. Potrebno je odabrati tipke tako da igrači ne smetaju jedan drugome. Odabrane su tipke '1' i '3', 'I' i 'P', 'F1' i 'F3' te 'F10' i 'F12'. Rješenje je dano u programskom jeziku C, budući da su u njemu sadržane funkcije *PS2\_Controller*-a. Primanje podataka s tipkovnice registrirano je predefiniranom funkcijom, a vrijednost pritisnute tipke nalazi se u varijabli pod nazivom RxBuffer, primjer u programskom kodu 5.

|  |  |
| --- | --- |
| Linija | Kod |
| 256: | BytesReceived = XPs2\_Recv(&Ps2Inst, &RxBuffer, 1); |

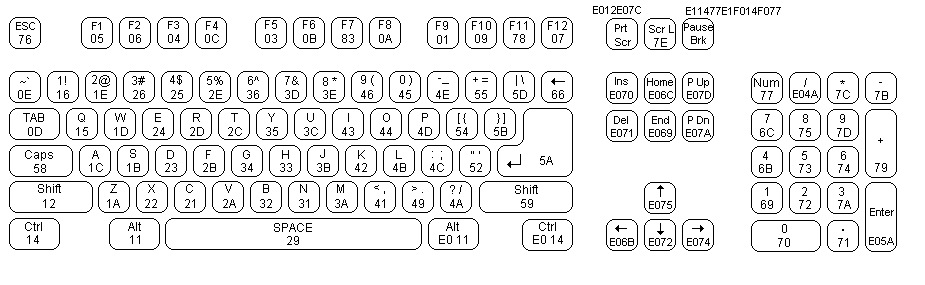
Programski kod 5.Primanje podataka s tipkovnice .

Bitno je, također, znati kako računalo detektira pritisnutu tipku na tipkovnici. Svaka tipka na tipkovnici ima svoj heksadekadski kod, koji računalo detektira njezinim pritiskom. Međutim, nakon što korisnik otpusti tipku, računalo prima kod '0xF0', što je kod za otpuštenu tipku, zatim ponovno još jednom zaprimi kod pritisnute tipke. Kako bi se ispravno izvelo zamišljeno korištenje naizmjenično pritisnutih tipki za pokretanje virtualnog trkača, izvedena je provjera pritisnute tipke na kod koji računalo primi nakon koda za otpuštenu tipku. Prepoznavanje izmjeničnog tipkanja za prvog trkača prikazano je u kodu 6.

|  |  |
| --- | --- |
| Linija | Kod |
| 85: | if(RxBuffer == 0x16){ |
| 86: | flag1=1; |
| 87: | } |
|  |  |

Programski kod 6.Kod za prepoznavanje izmjeničnog tipkanja .

Iz priloženog dijela koda vidljivo je kako se prvo ispituje je li pritisnuta tipka '1' na tipkovnici. Ukoliko jest, globalna varijabla flag1, kojoj je pri deklaraciji dodijeljena vrijednost 0, poprima vrijednost 1. Na sljedećoj slici prikazana je standardna tipkovnica s odgovarajućim heksadekadskim vrijednostima tipki.



Slika 5: Heksadekadske vrijednosti tipki.

|  |  |
| --- | --- |
| Linija | Kod |
| 88: | if(RxBuffer == 0x26 && flag1 == 1){ |
| 89: | if (location\_x1 < 506){ |
| 90: | location\_x1+=8; |
| 91: | flag1 = 0; |
| 92: | } |

Programski kod 7. Pomicanje igrača nakon provjere.

Nadalje, u retku 88 (programski kod 7.) ispituje se je li pritisnuta sljedeća tipka, a to je po dogovoru, tipka '3'. Međutim, također se ispituje je li prethodno pritisnuta tipka '1' i to na način da se provjerava globalna varijabla, čija je vrijednost 1 u slučaju da je pritisnuta ta tipka. Ukoliko je to slučaj, prolazi se sljedeći uvjet, koji dopušta varijabli location\_x1 da dosegne maksimalnu vrijednost 505. Ta vrijednost predstavlja vrijednost u pikselima na ekranu do koje seže trkaća staza. Ukoliko je vrijednost spomenute varijable manja od 506, ona se povećava za 8, a globalna varijabla flag1 postavlja se na vrijednost 0, kako bi se opet mogla detektirati pritisnuta tipka '1'.

Kako bi utrka na 100 metara bila što sličnija stvarnoj utrci u smislu vremena koje je potrebno za završetak, izvedena je kalibracija brzine tipkanja (tablica 1.). Za prosjek vremena trajanja utrke na 100 metara uzeta je vrijednost od 10 sekundi. Sva tri člana projektnog tima 10 su puta pokušali u intervalu od 10 sekundi što više puta izmjenično pritisnuti dvije različite tipke. Korištena je štoperica, a rezultat svakog pojedinog člana je zabilježen.

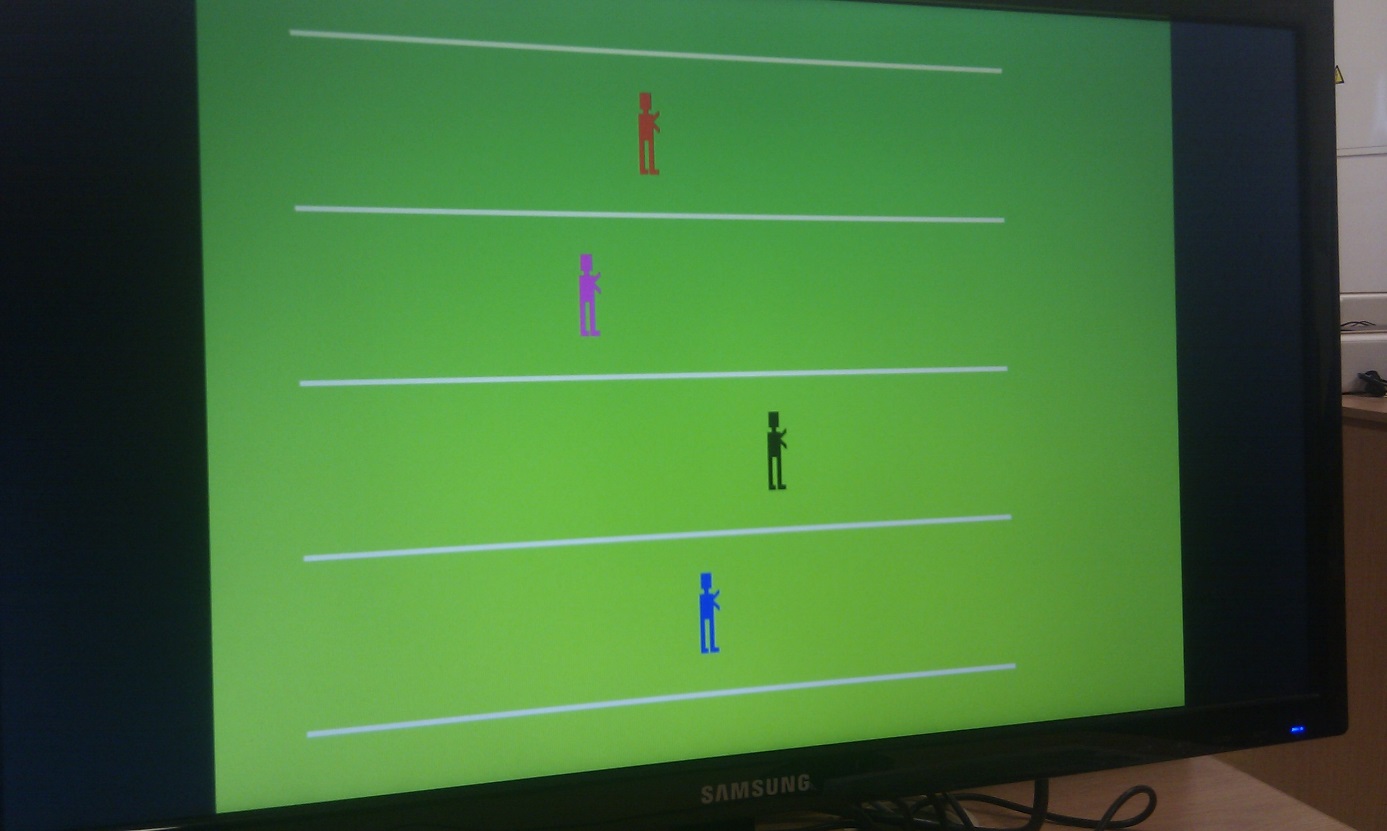
|  |  |  |  |
| --- | --- | --- | --- |
| Br. pokušaja | Prvi član | Drugi član | Treći član |
| 1. | 55x | 58x | 53x |
| 2. | 59x | 59x | 55x |
| 3. | 56x | 52x | 60x |
| 4. | 60x | 52x | 51x |
| 5. | 56x | 55x | 53x |
| 6. | 60x | 59x | 55x |
| 7. | 59x | 57x | 56x |
| 8. | 60x | 56x | 52x |
| 9. | 59x | 57x | 61x |
| 10. | 59x | 64x | 58x |

Tablica 1: Kalibracija brzine tipkanja.

Uzevši u obzir statistiku danu u tablici, prosjek broja izmjenično pritisnutih tipki u intervalu od 10 sekundi je 57 puta. Sukladno tome, nakon što je odrađen logički dio zadužen za kretanje igrača, odbrojeno je 57 izmjeničnih pritiskanja odgovarajućih tipki i detektiran trenutni položaj igrača. Kako bi se iskoristio što veći prostor na ekranu, iteracijom je odlučeno da korak igrača ne iznosi jedan, nego osam piksela. Na mjestu gdje se igrač zaustavio nakon 57 izmjeničnih tipkanja s korakom od 8 piksela, definiran je kraj staze.

Početak staze nije na početku ekrana, nego na razmaku od 50 piksela od lijevog ruba ekrana, što je u kodu definirano postavljanjem varijabli za lokaciju na početnu vrijednost 50. Slijedi kod koji izvršava upisivanje vrijednosti varijabli za lokaciju u odgovarajući registar, kako bi se kasnije u user\_logic.vhd moglo rukovati tim vrijednostima i iscrtavati trkače na odgovarajućim horizontalnim pozicijama. Dan je primjer za prvog igrača (prog. kod 8.), a sukladno tome su upisane i varijable ostalih igrača u svoje pripadajuće registre. Upisivanje vrijednosti varijabli u registre vrši se nakon svake promjene vrijednosti varijabli.

## Prepoznavanje pobjednika



Slika 6. Promijenjene lokacije igrača

|  |  |
| --- | --- |
| Linija | Kod |
| 169: | VGA\_CONTROLLER\_mWriteSlaveReg1(XPAR\_VGA\_CONTROLLER\_0\_BASEADDR,  0, location\_x1); |

Programski kod 8. Upisivanje vrijednosti varijabli za lokaciju u odgovarajući registar.

Zadnji korak ovoga projekta je prepoznavanje pobjednika utrke, odnosno, prepoznavanje konačnog poretka trkača. Glavna ideja je onemogućiti trkaču daljnje kretanje nakon što dođe do kraja trkaće staze te neposredno nakon što se zaustavi na cilju ispisati u ravnini njegove staze redni broj njegovog dolaska na cilj. Ukoliko je trkač utrku završio prvi, ispisuje se broj '1', ukoliko je završio drugi broj '2', itd. Kao u prethodnim koracima, rješenje ima dva dijela, kod u C-u i kod u VHDL-u. Slijedi kod u C-u s prigodnim objašnjenjem(programski kod 9.).

|  |  |
| --- | --- |
| Linija | Kod |
| 94: | if (flag\_poz1==0){ |
| 95: | igrac1\_cilj=1; |
| 96: | polje[br] = igrac1\_cilj; |
| 97: | br++; |
| 98: | flag\_poz1=1; |
| 99: | } |
|  |  |

Programski kod 9. Kod za provjeru završetka utrke prvog igrača.

Rješenje je izvedeno korištenjem dviju različitih globalnih varijabli. Jedna od njih je tzv. *flag*, prikladnog naziva flag\_poz1. Budući da je prikazan kod za prvog trkača, sukladno tome *flag* za drugog igrača je naziva flag\_poz2, itd. Ta globalna varijabla služi za provjeru je li igrač već prethodno došao do kraja utrke. Ukoliko jest, cjelokupan kod u vitičastim zagradama se neće izvršiti, no ako već prije nije došao do kraja, tada se u varijablu igrac1\_cilj sprema vrijednost 1. Potrebno je upotrijebiti taj mehanizam jer u protivnom će se dogoditi situacija u kojoj se igrač prestane kretati nakon što dođe do cilja, no njegov redni broj završetka utrke će se mijenjati ako korisnik ponovno pritisne tipke za pokretanje trkača. Spomenuta varijabla čuva redni broj trkača pri startu, odnosno čuva oznaku broja staze kojom trkač trči. Vrlo je bitan segment ove rutine polje cijelih brojeva naziva polje[] u koje se spremaju oznake staze pojedinog trkača sekvencijalno, kako trkači dolaze na cilj, prikazano u kodu 10.

Time je osigurano da ukoliko trkač koji trči stazom broj 1 prvi dođe na cilj, svoju oznaku staze spremljenu u varijabli igrac1\_cilj prvi ubaci u polje. Nakon što to učini, povećava brojač koji služi kao indeks za polje, tako da trkač koji dođe na cilj drugi po redu ne prepiše svoju oznaku preko njegove, nego na sljedeće slobodno mjesto u polju. Time će se očuvati poredak igrača pri dolasku na cilj. Nakon što svi igrači završe utrku i svoje oznake ubace u polje onim redom koji su stizali na cilj, potrebno je iz tog polja pročitati redoslijed trkača te odgovarajuću informaciju proslijediti odgovarajućem registru.

|  |  |
| --- | --- |
| Linija | Kod |
| 191: | for (i =0; i<4; i++){ |
| 192: | if(polje[i]==1){ |
| 193: | VGA\_CONTROLLER\_mWriteSlaveReg2(XPAR\_VGA\_CONTROLLER\_0\_BASEADDR,  0,i+1); |
| 194: | } |
| 195: | if(polje[i]==2){ |
| 196: | VGA\_CONTROLLER\_mWriteSlaveReg4(XPAR\_VGA\_CONTROLLER\_0\_BASEADDR,  0,i+1); |
| 197: | } |
| 198: | if (polje[i]==3){ |
| 199: | VGA\_CONTROLLER\_mWriteSlaveReg6(XPAR\_VGA\_CONTROLLER\_0\_BASEADDR,  0,i+1); |
| 200: | } |
| 201: | if (polje[i]==4){ |
| 202: | VGA\_CONTROLLER\_mWriteSlaveReg8(XPAR\_VGA\_CONTROLLER\_0\_BASEADDR,  0,i+1); |
| 203: | } |
| 204: | } |

Programski kod 10. Kod za određivanje poretka igrača.

Prolaskom kroz polje u kojemu su smještene oznake svakog igrača onim redoslijedom kojim su dolazili na cilj, ispituje se je li vrijednost na trenutnom mjestu u polju jednaka jednoj od oznaka trkača. Kada se utvrdi čija je oznaka na kojem rednom mjestu u polju, tada se u registar koji pripada određenom igraču upisuje vrijednost indeksa mjesta u polju uvećana za 1, budući da indeksi polja započinju od vrijednosti 0. Dakle, ako je igrač u prvoj stazi završio utrku prvi, tada će njegova oznaka biti na prvom, to jest nultom mjestu u polju. Kada se utvrdi da je vrijednost upisana na prvom, odnosno, nultom mjestu u polju jednaka oznaci prvog igrača, tada se u registar SlaveReg2 šalje vrijednost 1, budući da je to osvojeno mjesto trkača u prvoj stazi. Dakle, svaki trkač ima svoj odgovarajući registar u kojemu je zapisan redni broj mjesta kojeg je osvojio prilikom ulaska u cilj.

Sukladno tome, u datoteci *user\_logic.vhd* vrši se daljnja obrada vrijednosti u registrima kako bi se pored igrača koji se zaustavi na cilju ispisala znamenka koja označava njegovo osvojeno mjesto, na isti način kako se iscrtavao virtualni lik trkača i njegova trkaća staza. Slijedi kod kojim se u slučaju da je prvi igrač stigao na cilj prvi, pored trkača iscrtava brojka '1'. Slično vrijedi za slučaj da je prvi igrač stigao na cilj drugi, treći ili četvrti. Jedino što se mijenja je raspon piksela koje valja obojati kako bi se na ekranu dobio prepoznatljiv prikaz znamenke. Također, slično vrijedi za ostale igrače, a razlika je u vertikalnoj poziciji ispisane znamenke, jer je cilj da pored svakog igrača stoji znamenka njegovog osvojenog mjesta. Znamenka je ispisana crnom bojom. Prikazano u programskom kodu 11.

|  |  |
| --- | --- |
| Linija | Kod |
| 323: | if (finish\_x1 = "001") |
| 324: | then |
| 325: | if ((horizontal\_counter >= 144 + 590) and (horizontal\_counter <= 144 + 600) and (vertical\_counter >= 39 + 90 - 30) and (vertical\_counter <= 39 + 90 + 30)) |
| 326: | then |
| 327: |  |
| 328: | rgb\_out <= "00000000"; |
| 329: | end if; |

Programski kod 11.Ispisivanje poretka crnom bojom.

Slijedi slika koja prikazuje kako izgleda završetak utrke za sva četiri igrača. Na njoj je jasno vidljivo da je crni trkač u trećoj stazi došao na cilj prvi, crveni trkač u prvoj stazi drugi, plavi trkač u četvrtoj stazi treći te ružičasti trkač u drugoj stazi posljednji, odnosno, četvrti (prikazano na slici 7)777777.



Slika 7. Završetak utrke za sva četiri igrača.

# Zaključak

Projekt je uspješno odrađen, zahvaljujući dobroj pripremi na laboratorijskim vježbama te prethodno stečenih znanja, poglavito programiranja u programskom jeziku C te pisanja koda u VHDL-u. Od prvotno zadanog zadatka, učinjeno je nekoliko promjena te je simulacija utrke poprimila dovoljno realističan izgled jer prepoznatljivi su trkači, trkaće staze te osvojena mjesta. Logika kretanja igrača, njihov izgled, izgled staze, ispisivanje broja osvojenog mjesta u utrci te boja pozadine određeni su i datoteci *user\_logic.vhd* u VHDL-u. Komponenta koja podrazumijeva čitanje izmjeničnog tipkanja odgovarajućih tipki te slanje informacija za kretanje i osvojena mjesta u odgovarajuće registre odrađena je u C-u. Duljina koraka igrača te duljina staze određeni su prema stvarnim podacima utrke na 100 metara, tako da utrka traje u prosjeku 10 sekundi. Međutim, postoji žaljenje za stopostotnom funkcionalnosti simulacije, jer u slučaju da sva 4 igrača istovremeno igraju, procesor mora istovremeno obrađivati previše ulaznih podataka pa se događa da detekcija jednog ulaznog signala blokira detekciju drugoga. Stoga se trkači neće pomicati na svaki pritisak i utrka traje jako dugo. Taj bi se problem u budućnosti mogao riješiti implementiranjem više PS2 kontrolera. Međutim, dvojica igrača se mogu natjecati bez navedenih problema. Ovaj je problem ostavljen za razrješenje nekom drugom prilikom u budućnosti. Prilikom izrade projekta stečena su vrijedna i obilna iskustva i znanja, koja će zasigurno koristiti u daljnjem studiranju i napretku pojedinca.

# Literatura

|  |  |
| --- | --- |
| [1] | Digilent, »Nexys3 Board Reference Manual,« Digilent, Pullman, WA, 2013. |
| [2] | Xilinx, »Embedded System Tools Reference Manual - Embedded Development Kit,« Xilinx, 2008. |
| [3] | Xilinx, »MicroBlaze Processor Reference Guide - Embedded Development Kit EDK 10.1i,« Xilinx, 2008. |