Elektrotehnički fakultet, Beograd
Osnovi računarske tehnike 2 (13S112ORT2)

FPGA Projekat:

Red ball bouncer

-bouncing ball endless game-

U Beogradu,
24.09.2019.

Luka Simovic 0423/17

1. Opis projekta

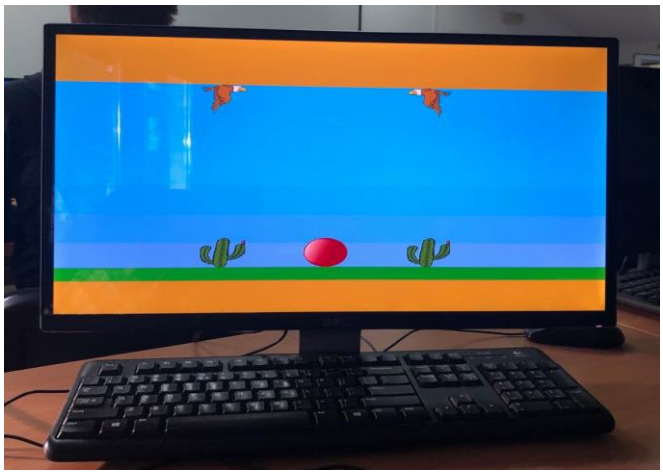
Cilj projekta jeste realizacija igrice Red ball bouncer.
Projekat uključuje sledeće periferije i protokole: monitor, tastaturu, VGA protokol.

Igru igra jedan igrač. Igra je endless game karaktera i završava se onog trenutka kada lopticu koju kontroliše igrač, pogodi jedna od prepreka koje se u tom trenutku na ekranu prikazuju.

Igrica je napravljena tako da igrač može sam da optimizuje težinu koja njemu odgovara u zavisnosti od njegove uvežbanosti. Optimizacija se reguliše tako što igrač pomoću 4 sviča gasi trenutno postavljene prepreke. Što je prepreka manje to je igra i lakša.

Pozadina igrice je nebo , sa gornjim i donjim preprekama koje se nalaze na ekranu.

Originalni izgled igrice

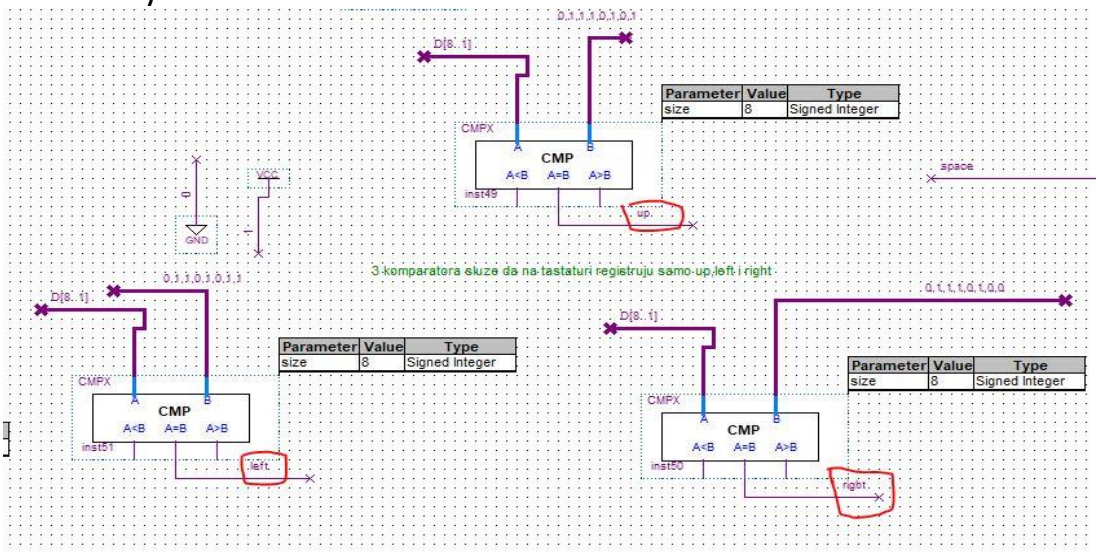


Moja bolja verzija igrice

2. Realizacija

Projekat je realizovan u Altera Quartus razvojnom okruženju za FPGA pločicu Terasic Altera DE0 Board (Cyclone® III čip) na kojoj je i testiran.

Kretanje je realizovano pomoću standardne šeme date na labu 1. Šema služi za korišćenje strelica gore,levo i desno, 3 glavna signala potrebna za kretanje loptice pomoću tastature. Najbitniji deo šeme jesu tri 8bitna komparatora koja služe za prepoznavanje 3 make koda potrebna za kretanje loptice, koji kao logički uslovi utiču na njeno pomeranje x, odnosno y koordinate.



3 signala koja utiču na kretanje loptice

Cela šema pod nazivom tastatura je iskorišćenja za kreiranje simbola pod istim imenom koji se dalje koristi u glavnom layeru.

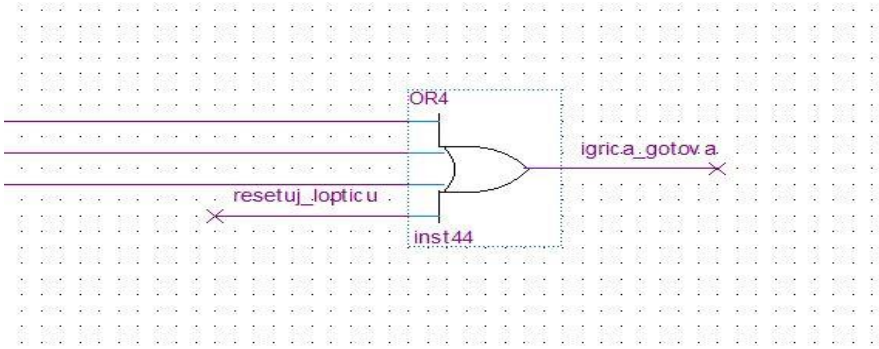
Sama loptica je komponenta koja je izvučena iz memorije, kao i ostale prepreke koje se nalaze na ekranu. Dobijanje jasne slike loptice kao i prepreka ću detaljno opisati u delu ,,**Grafika i memorija“.**

Sam prikaz loptice, kao i ostalih prepreka je regulisan ,,IS_VALID_ZA_PREPREKE” komponentom koja na izlazu izbacuje bit validnosti koji određuje da li se trenutni piksel nalazi unutar prepreke ili loptice i da li njega treba iscrtati(ako treba preko multipleksera odrediti kojom bojom piksel treba da se iscrta).

Za iscrtavanje pozadine , kao i donjeg i gornjeg dela gde se nalaze prepreke korišćene su 2 komponente koje propuštaju RED GREEN i BLUE signale predstavljene na širini 4 bita. Više o komponentama ću opisati u delu ,,**ŠEME“.**

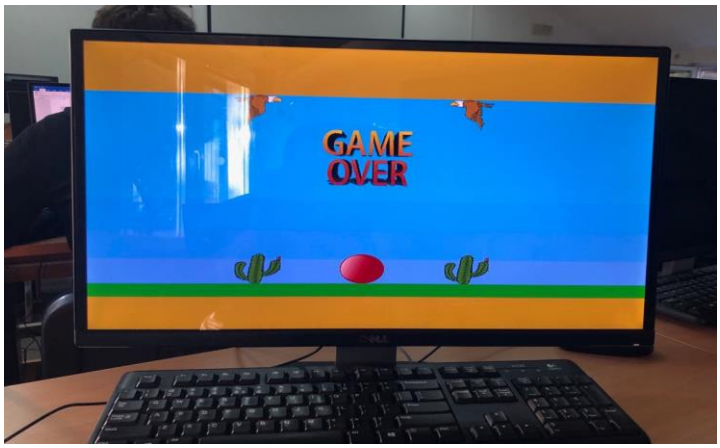
Pomeranje prepreka je realizovano pomoću registara u kojima su se početne vrednosti x koordinata postepeno umanjuju za zadati broj, samim tim se stvara efekat random pomeranja prepreka, jer se svaka x komponenta jedne od prepreka umanjuje za broj koji se razlikuje od ostalih brojeva koji služe za smanjivanje vrednosti u registru.

Realizacija kraja igre je urađena tako što pri dodiru sa jednom od prepreka, loptica, kao i ostale prepreke trenutno aktivne na ekranu, se postavljaju na svoje početne pozicije, gde se čeka taster enter za pokretanje igrice od početka. Na glavnoj šemi se nalazi jedno OR kolo sa 4 ulaza koje detektuje max 4 kolizije (loptica je udarila u jednog od 2 orla gore ili 2 kaktusa dole).



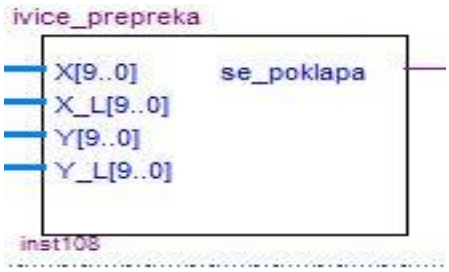
Glavni signal za zaustavljanje igrice

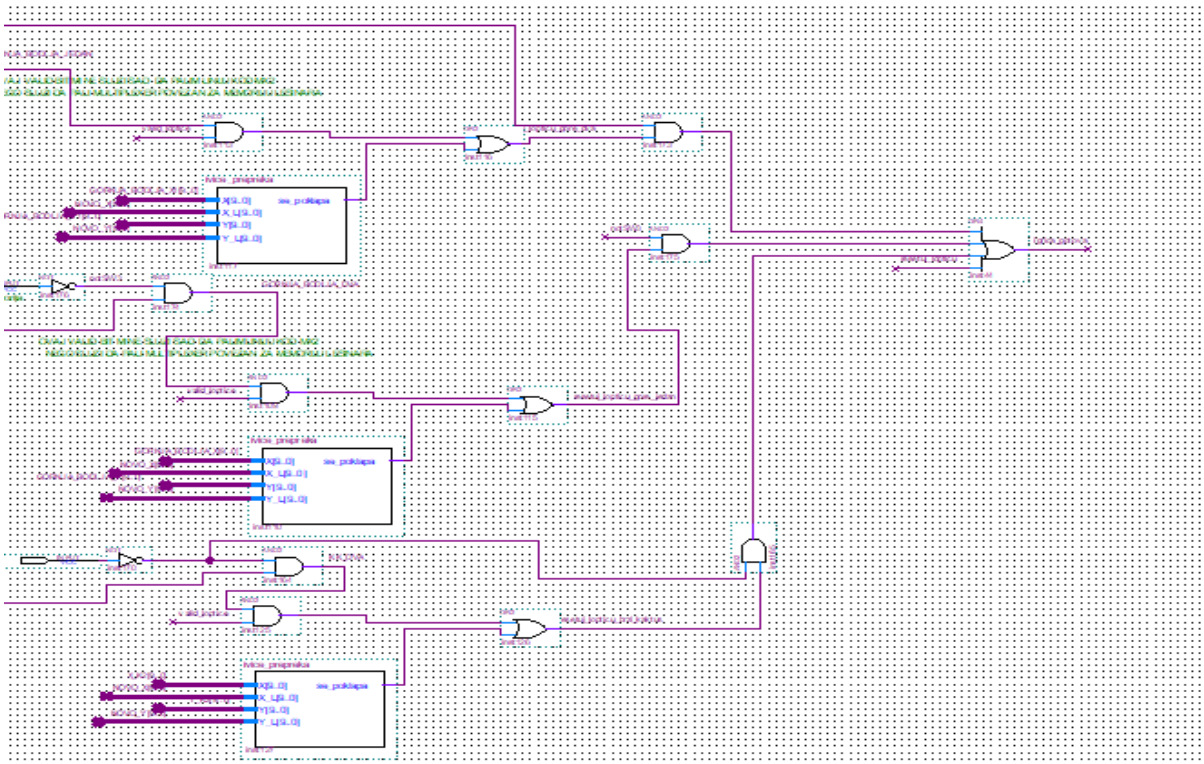
Signal ,,igrica_gotova'' dalje uz kombinaciju sa ,,space'' signalom koji dolazi nakon pritiska tastature dalje utiče na zaustavljanje loptice kao i svih ostalih prepreka, nakon prethodnog prikaza ,,GAME OVER'' slike izvučene iz memorije kao indikator za kraj igre.



Game over indikator za kraj igre

Sudar sa jednom od prikazanih prepreka , jeste ulaz na jedan od 4 inputa gore navedenog or4 kola koje reguliše kraj igre. Sam ulaz je regulisan ,,**ivice_prepreka"** komponentom koja uz kombinaciju sa and i or kapijama reguliše da li kolizija postoji i pomaže pri tačnijem određivanju da li se desilo poklapanje trenutnog piksela loptice i prepreke.





Deo koji regulise da li kolizija postoji

Realizacija kompletne slike se izvršava preko nekoliko redova multipleksera koji treba da propuste tačno određenu R, G, B kombinaciju , koja na ekranu boji piksel određenom bojom. Za realizaciju multipleksera koristio sam multiplekser sa 2 ulaza podesivnog broja bitova.

3. Grafika i memorija

Za prikazivanje png slika kao grafike u ovom projektu, upotrebljeni su memorijski moduli dostupni u Quartus okruženju koji koriste ugrađenu memoriju FPGA pločice.

Koristio sam u Python-u kod za pretvaranje png slike iz 24-bitnih u 12-bitnu paletu boja, a zatim i u memorijski, mif fajl kojim je inicijalizovana memorija. Svaka memorija je širine 13 bitova od kojih prvih 12 predstavljaju crvenu, zelenu i plavu komponentu boje, a 13. bit je bit validnosti, odnosno naznaka da li se radi o providnom ili obojenom pikselu. Sama veličina svakog memorijskog modula zavisi od veličine slike, kao i broja slika u okviru tog memorijskog modula. Svaka slika je linearizovana po redovima u memoriji. Širine svih slika su stepeni dvojke kako bi se adresna funkcija mogla izračunati jednostavnim bitovnim pomeranjem.

4. Dalja unapređenja i mane

Dalja unapređenja bi obuhvatala:

- Poboljšanje grafike igrice (svaka prepreka je ustvari kvadrat veličine 64x64 što znači da orao u vrhu i ako se ne dodirnu a loptica se dovoljno približi tom kvadratu u kom je iscrtan orao, desiće se kolizija. Ista situacija je i za kaktus).
- Dodavanje novih prepreka. Povećanje broja prepreka i mogućnost da se igrica dodatno oteža.
- Dodavanje supermoći loptici što omogućuje da igrač duže igra igricu.

Mane projekta koje bismo istakli su:

- Prepreke, orao i kaktus, koji su iscrtani u kvadratima 64x64 piksela, zapravo nisu kvadrati te veličine, samim tim je i kolizija malo neprecizna.

S obzirom da sam u potpunosti svestan da kolizija nije u potpunosti ispravna, rešenje ovog problema bi bilo maunelno iscrtavanje orla i kaktusa piksel po piksel. S obzirom da sam projekat radio sam , izvršenje istog zadatka bi oduzelo previše vremena i realizacija preostalog dela projekta ne bi bila dovoljno kvalitetna. Takodje slika koju bih u tom

trenutku iscrtavao, ne bi izgledala lepo kao orao koji je izvučen iz memorije.

5. Šeme

Bazične komponente korišćenje za realizaciju projekta:

- ***ADD*** = višerazredni sabirač sa podesivim brojem bitova
- ***SUB*** = višerazredni oduzimač sa podesivim brojem bitova
- ***CMPX*** = višerazredni komparator sa podesivim brojem bitova
- ***REGX*** = višerazredni registar sa podesivim brojem bitova i operacijama učitavanja, pomeranja u desno, brisanja, inkrementiranja, dekrementiranja
- ***CONSTX*** = konstanta sa podesivim brojem bitova
- ***CLK_DIVIDER*** = skaliranje ulaznog takta na zadati takt
- ***RaisingEdge*** *= šema za detekciju uzlaznog takta*
- ***MP2X*** = multiplekser sa 2 ulaza podesivog broja bitova

Glava šema realizuje projekat kombinujući sledeće ključne šeme:

VGA protokol

- ***VGAController*** (šema za generisanje trenutnih koordinata koje se iscrtavaju, kao i signala V_SYNC, H_SYNC, DISP_EN)
- ***Tastatura*** (šema za generisanje rada tastature, kao i regulisanje signala left, right, up, space)
- ***IS_VALID_ZA_PREPREKE*** (šema za generisanje valid bita za iscrtavanje loptice ili jedne od prepreka. Na input se primaju trenutno X,Y, koordinate X i Y prepreke ili loptice, širina i duzina , a na output je signal koji reguliše da li se taj piksel nalazi unutar prepreke ili lopte ili ne).
- ***IVICE_PREPREKA*** (šema koja reguliše koliziju)
- ***BACKGROUND (***šema koja reguliše boje koje treba da prikazu gradijent iscrtavanja neba)
- ***MESTO_KRETANJA_LOPTICE*** (šema koja iscrtava dva pravougaonika gore i dole i generiše boje za dobijanje braon i zelene boje kocke.)

- **Memorija**
  - ***Kaktus*** (memorijski modul u kom se nalazi slika kaktusa)
  - ***gameoverr***(memorijski modul u kom se nalazi slika za indikaciju kraja igre)
  - ***lesinar*** (memorijski modul u kom se nalazi slika orla(lešinara)).
  - ***Ball*** (memorijski modul u kom se nalazi slika crvene lopte)