UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku

Predmet: Uvod u projektovanje VLSI sistema

Projekat: Tenkići

Faza 3: izveštaj

Rok za predaju: 2.1.2016.

Projekat radili:

Ime	Prezime	Broj indeksa
Dejan	Petković	77/2012
Predrag	Kuzmanović	49/2012

Sadržaj

Rezime

Ovaj rad predstavlja izveštaj o projektu iz predmeta *Uvod u projektovanje VLSI sistema* [1], koji je izborni predmet u sedmom semestru Odseka za elektroniku na Elektrotehničkom fakultetu u Beogradu. U ovom dokumentu opisan je postupak projektovanja jednostavnog digitalnog sistema, koji uz odgovarajući VGA (*Video Graphics Array*) displej omogućava igranje igre *Tenkići*. Korišćen je jezik za opis hardvera VHDL (*VHSIC Hardware Description Language*) [2], dok je na raspologanju za programiranje i simuliranje bio softver Altera Quartus II 15. Hardverska implementacija je odradjena na ploči DE1-SoC (*System on Chip*) sa FPGA čipom iz Altera Cyclone V SoC familije [3]. Objašnjene su ideje i problemi pri izradi svake od faza projektovanja, uz slike simulacija i odgovarajućih blok-šema.

Uvod

Zadatak projekta je da se što vernije napravi igrica *Tenkići* i implementira na FPGA čipu. Od periferija na ploči DE1-SoC korišćena su 4 tastera, 1 prekidač, 8 LED dioda, kao i priključak na koji je priključen monitor računara, koji predstavlja VGA displej dimenzija 1024x768 piksela na kojem se igrica prikazuje.

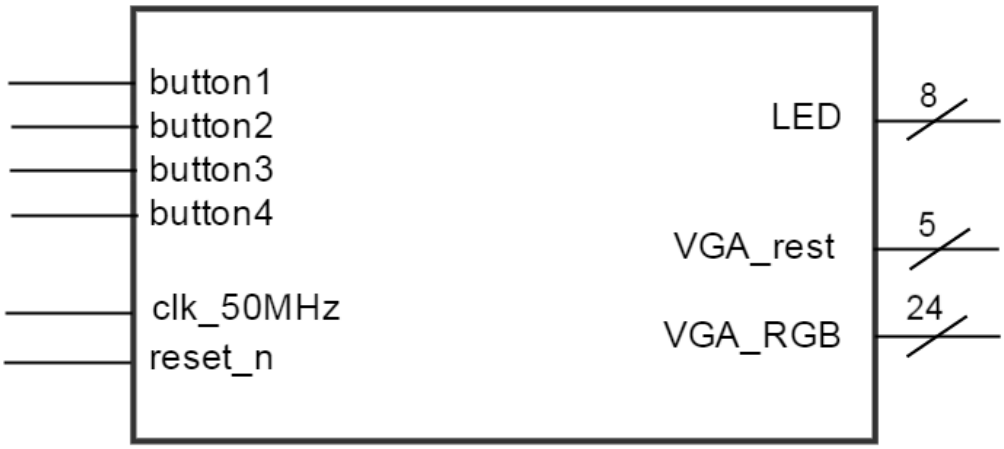
U donjem levom uglu ekrana nalazi se tenkić dimenzija 32x32 piksela. Na desnoj polovini ekrana na slučajno odabranoj poziciji nalazi se drugi tenkić koji se gađa. Protivnički tenk se nalazi na ivici brda, to jest brdo se iscrvava ispod protivničnog tenka nakon slučajnog odabiranja njegove pozicije. Igra se igra pomoću četiri tastera. Jedan taster praktično započinje igricu. Druga dva tastera služe za podešavanja trenutnog ugla pod kojim se izbacuje projektil, to jest za odabiranje jednog od pet različitih pravaca izbacivanja (pod uglovima 15°, 30°, 45°, 60° i 75° u odnosu na horizontalu, redom). Poslednji taster ispaljuje projektil dimenzija 4x4 piksela. Kontinualnim pritiskom na taj taster definiše se jačina ispaljivanja projektila koja se istovremeno ispisuje na diodama u termometarskom kodu. Na kretanje projektila utiče gravitaciona sila (tako sto se *y* komponenta brzine ravnomerno smanjuje), a nakon modifikacije projekta i promenljivi vetar (analogno, samo po *x*-osi). Po uspešnom pogotku tenkića započinje se novi ciklus igre, generisanjem nove pozicije mete.

Što se tiče same implementacije, najpre je osmišljena osnovna mašina stanja čija stanja predstavljaju posebne delove projekta. Svaki od tih delova je posebno isplaniran, instanciran i isimuliran i oni su zatim spojeni u glavni modul *tenkici.vhd*, spreman za spuštanje na ploču. Dakle, projekat je podeljen u više *.vhd* fajlova, a konstante i tipovi koji su zajednički za sve te fajlove izdvojeni su u poseban paket *tenkici_pkg*.

Za potrebe „kućne" VGA simulacije korišćen je onlajn VGA simulator [4] koji generiše sliku na osnovu tekstualnog fajla izgenerisanog od strane *ModelSim*-a, nakon simulacije. Ovaj tekstualni fajl se generiše u posebnom procesu *testbench* fajla glavnog modula, upotrebom odgovarajućih biblioteka za rad sa tekstualnim fajlovima.

1. Opis sistema i glavnog modula

Osnova sistema je zamišljena kao jedinstvena mašina stanja koja se kontroliše komandama sa tastera razvojne ploče DE1-SoC, a u isto vreme na VGA prikazuje dešavanja i eventualno pali LED diode na ploči. Blok-šema sistema je prikazana na slici 1.1.



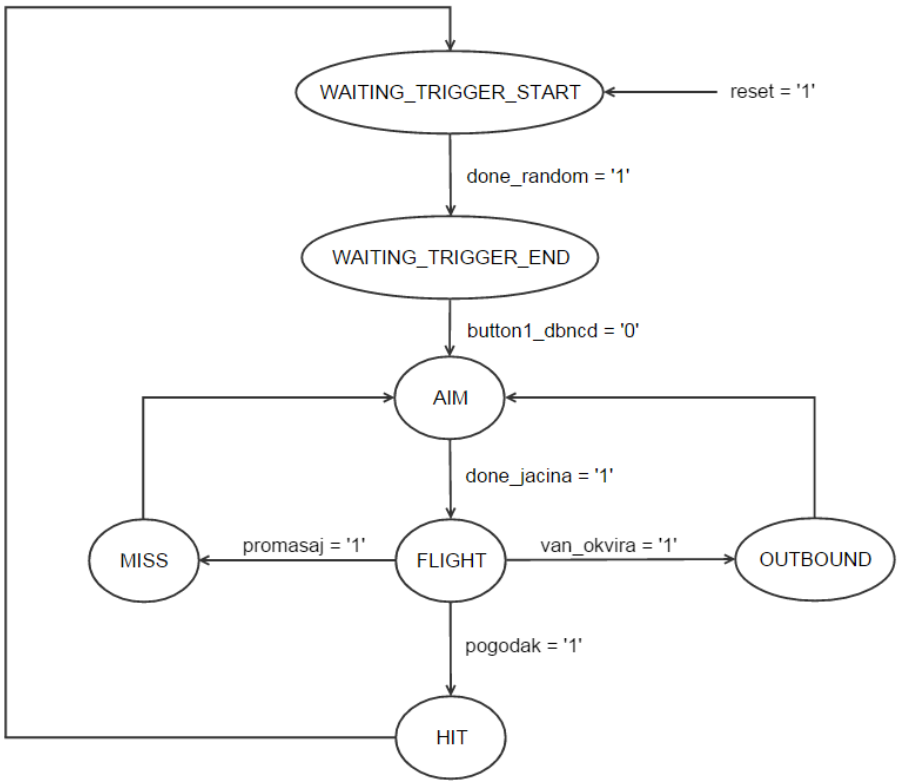
Slika 1.1 – Blok-šema sistema

Kratak pregled ulaznih i izlaznih signala sistema koji pruža bolji uvid u iste dat je u tabeli 1. Ovi signali se pomoću generičke TCL skripte modifikovane na odgovarajući način povezuju sa stvarnim pinovima periferija na razvojnoj ploči.

Tabela 1 – pregled ulaznih i izlaznih signala

Naziv signala	Opis	Smer
clk_50MHz	Globalni signal takta	ulazni
reset_n	Globalni signal reseta (aktivan u nuli)	ulazni
button1	Taster 1: započinjanje nove igrice	ulazni
button2	Taster 2: nišanjenje nagore	ulazni
button3	Taster 3: nišanjenje nadole	ulazni
button4	Taster 4: ispaljivanje projektila	ulazni
LED	8-bitni signal, daje informaciju o trenutnom stanju LED dioda	izlazni
VGA_RGB	24-bitni signal, daje informaciju o boji (u RGB sistemu)	izlazni
VGA_rest	Ostali signali bitni za rad komponente *vga_sync*	izlazni

Na slici 1.2 prikazan je pojednostavljen dijagram osnovne mašine stanja sistema. Radi preglednosti, prikazani su samo signali koji vode u neko od sledećih stanja. U suprotnom se ostaje u trenutnom stanju.



Slika 1.2 – Dijagram osnovne mašine stanja sistema

Postoji **7** osnovnih stanja u mašini stanja sistema:

Waiting_trigger_start: Ovo je početno stanje, dakle i stanje u koje se dolazi po resetu sistema. U ovom stanju na displeju se iscrtava samo bela pozadina i čeka se na pritisak tastera *button1* kako bi nova partija započela, odnosno kako bi se prešlo u fazu nišanjenja. Ovde postoji prostor za nadogradnju projekta, u smislu dodavanja interaktivnog menija pre početka igre, izbora nivoa težine, izbora izgleda tenkova i okoline i slično.
Waiting_trigger_end: Ovo stanje predstavlja međustanje između početnog stanja i stanja nišanjenja. Ovo stanje je bilo potrebno da bi se rešio problem koji je nastao u ranoj fazi izrade projekta, kada se smatralo da svi spoljašnji signali dolaze isključivo sa tastera, samim tim i signal za reset. Naime, tada je bilo potrebno multipleksirati jedan od tastera, odnosno koristiti ga i za pokretanje nove igre i za ispaljivanje projektila. U tom slučaju, nije se smelo direktno nakon početnog stanja preći u stanje nišanjenja jer bi i dalje pritisnuti taster bio protumačen i kao zahtev za ispaljivanjem projektila. Demultipleksiranjem tastera, odnosno prebacivanjem signala za reset sa tastera na prekidač, logički je ovo stanje postalo nepotrebno, a oslobođen je i dodatni taster.

Aim: U stanju *aim* igrač selektuje poziciju cevke svog tenka, odnosno ugao (pravac) ispaljivanja projektila. Sa dva tastera *button2* i *button3* se nakon diferenciranja i debaunsiranja dobijaju signali *aim_up* i *aim_down* koji pomeraju cevku gore ili dole, što uzrokuje i promenu izgleda tenka (ovo obezbeđuje komponenta *crtanje*). Kontinualnim pritiskom na taster *button4* se sukcesivno pale diode, pomoću komponente *jacina* stvara i pamti signal *strength* koji se koristi u stanju leta kako bi se ažurirala brzina projektila na odgovarajući način. Po otpuštanju tastera *button4* prelazi se u stanje *flight* i započinje let projektila.
Flight: U ovom stanju se odvija let projektila, na osnovu parametara koji su dobijeni iz faze nišanjenja. Početna pozicija projektila je sam kraj cevke tenka, i određuje se u zavisnosti od odabranog ugla ispaljivanja. Početna horizontalna i vertikalna brzina određene su na osnovu signala jačine *strength* i odabranog ugla prilikom nišanjenja. Brzine su diskretnog tipa (u dimenzijama frejm po pikselu), i odabrane su tako da što vernije odgovaraju jednačinama za komponente početne brzine kod kosog hica ((1) i (2)). U ovim jednačinama v_0 odgovara jačini, a α odabranom pravcu ispaljivanja projektila. Trenutna horizontalna i vertikalna komponenta brzine sa ažuriraju tako da što vernije odgovaraju jednačinama za trenutne brzine kod kosog hica ((3) i (4)).

$$v_{x0} = v_0 \cos(\alpha), \quad (1)$$

$$v_{y0} = v_0 \sin(\alpha), \quad (2)$$

$$v_x(t) = v_{x0}, \quad (3)$$

$$v_y(t) = v_{0y} - g \cdot t. \quad (4)$$

Pošto su brzine diskretnog tipa, pribeglo se rešenju ažuriranja vertikalne brzine promenom za konstantnu vrednost *g* nakon svakih *frames_for_g* frejmova. Takođe, horizontalna i vertikalna pozicija projektila se ažuriraju na kraju svakog iscrtavanja jednog frejma (signal *ref_tick*) tako što joj se doda odgovarajuća horizontalna, odnosno vertikalna komponenta trenutne brzine. Stanje leta može da rezultira jednim od 3 ishoda: protivnički tenk je pogođen (stanje *hit*), protivnički tenk je promašen (stanje *miss*) i projektil je otišao van okvira slike (*outbound*).
Hit: Stanje u koje se dolazi ako je igrač pogodio protivnički tenk. Ostavljen je prostor za nadogradnju ovog dela projekta, u smislu dodavanja eksplozije oko protivničkog tenka i ispisivanja poruke korisniku o uspešno završenoj partiji, ili eventualno o statistici gađanja. Logički, iz ovog stanja se prelazi u početno stanje, *waiting_trigger_start*.
Miss: Stanje u koje se dolazi ako je igrač promašio protivnički tenk, ali je projektil završio u okviru slike (projektil je pao na zemlju ili udario u brdo). Ostavljen je prostor za nadogradnju ovog dela projekta, u smislu dodavanja eksplozije oko mesta pada projektila ili eventualnog degradiranja terena. Inače, logički se iz ovog stanja prelazi u stanje *aim* i nastavlja se trenutna partija, sve dok igrač ne pogodi metu.
Outbound: Stanje u koje se dolazi ako je igrač promašio protivnički tenk, a projektil je izašao van okvira slike. Do ove situacije dolazi ukoliko je početna brzina projektila prevelika. Logički se iz ovog stanja prelazi direktno u stanje *aim* i nastavlja se trenutna partija, sve dok igrač ne pogodi metu.

2. Opis pojedinačnih modula

U ovom delu će ukratko biti predstavljene sve bitne komponente sistema koje su realizovane kao zasebni moduli.

2.1. tenkici_pkg

Radi preglednosti i smanjenog dupliranja koda svi korisnički tipovi podataka i konstante smešteni su u poseban paket *tenkici_pkg* realizovan u modulu *tenkici_pkg.vhd*, koji se onda po potrebi poziva u ostalm modulima. Na taj način je i u slučaju dodatnog testiranja ili ispravljanja grešaka dobar deo izmena lokalizovan, što se pokazalo kao dobra odluka.

2.2. diff

Ova komponenta se u projektu koristi za diferenciranje nekih ulaznih signala koji dolaze sa tastera razvojne ploče. Diferenciranje je potrebno za signale nišanjenja, jer je za jedan pritisak tastera potrebno napraviti samo jednu promenu položaja cevke tenka. Takođe se diferencira i signal za pokretanje nove partije, iz sličnih razloga.

2.3. debounce

Ova komponenta se u projektu koristi za debaunsiranje svih ulaznih signala koji dolaze sa tastera razvojne ploče. Izlazni signal *B* menja stanje na osnovu ulaznog signala *A* samo ukoliko ulazni signal traje bar *debounce_duration* uzlaznih ivica takta. Na ovaj način odstranjuju se eventualni gličevi ulaznih signala koji se javljaju kod upotrebe tastera.

2.4. lfsr10

Ova komponenta se u projektu koristi kao pomoć pri generisanju pseudoslučajne pozicije protivničkog tenka na početku svake nove partije tenkića. U osnovi ove komponente nalazi se 10-bitni LFSR (*Linear Feedback Shift Register*) registar [5], čiji se sadržaj menja sinhrono sa uzlaznom ivicom osnovnog signala takta. Sekvenca promene sadržaja registra je pseudoslučajna i periodična, dakle ponavlja se svakih 1024 perioda takta.

2.5. random

Ova komponenta se u projektu koristi za generisanje psudoslučajne pozicije protivničkog tenka na početku svake nove partije tenkića. U okviru ove komponente koristi se već isprojektovana komponenta *lfsr10*, i to na sledeći način. Po nailasku pozitivnog impulsa ulaznog signala *trigger* započinje proces generisanja pozicije tenka, koji se sastoji iz nekoliko faza.

Prva faza se sastoji od učitavanja vertikalne pozicije tenka, i sastoji se u dohvatanju svih *10* bita iz LFSR registra, i to sve dok se odgovarajuća decimalna vrednost očitanog niza bita ne nalazi u dozvoljenom opsegu vertikalne pozicije tenka. Dozvoljena vertikalna pozicija je praktično po celoj visini displeja, s tim što se mora voditi računa i o dimenzijama tenka. Na kraju ove faze na izlaznom signalu *vertical* nalazi se validna vertikalna pozicija tenka.

Druga faza se sastoji od učitavanja horizontalne pozicije tenka i od dohvatanja donjih 9 bita iz LFSR registra, sve dok se odgovarajuća decimalna vrednost očitanog niza bita ne nalazi u dozvoljenom opsegu horizontalne pozicije tenka. Dozvoljeno je da se tenk nalazi samo u desnoj polovini displeja, s tim što se mora voditi računa i o dimenzijama tenka. Upravo zbog ovoga je potrebno očitavati samo 9 bita za horizontalnu poziciju tenka, jer je bit najveće težine (10. bit) uvek jednak 1 (desna polovina slike). Na kraju ove faze na izlaznom signalu *horizontal* nalazi se validna vertikalna pozicija tenka.

Nakon uspešno završene druge faze, aktivira se izlazni signal *done* u trajanju od jednog perioda takta. Ovaj signal služi kao indikator da je generisana pozicija tenka validna i da se može čitati i koristiti u nastavku od strane drugih modula u sistemu.

2.6. jacina

Ova komponenta se u projektu koristi za određivanje jačine kojom se projektil ispaljuje, u zavisnosti od vremena pritiska tastera za ispaljivanje. Ulazni signal *pali* potiče sa jednog od tastera na ploči, nakon debaunsiranja. Izlazni signal *led* se u okviru celokupnog sistema povezuje na LED diode na ploči, a predstavlja odabranu jačinu ispaljivanja prikazanu u termometarskom kodu. Odgovarajuća decimalna vrednost odabrane jačine je izlazni signal *jacina_out*, a očitava se i koristi u nastavku od strane drugih modula u sistemu pomoću izlaznog signala *done*.

2.7. pll

Ova komponenta se koristi za generisanje pomoćnog signala takta učestanosti 65MHz, koji se koristi u komponenti *vga_sync*. Kod za ovu komponentu je generisan pomoću IP (*Intellectual Property*) biblioteke.

2.8. vga_sync

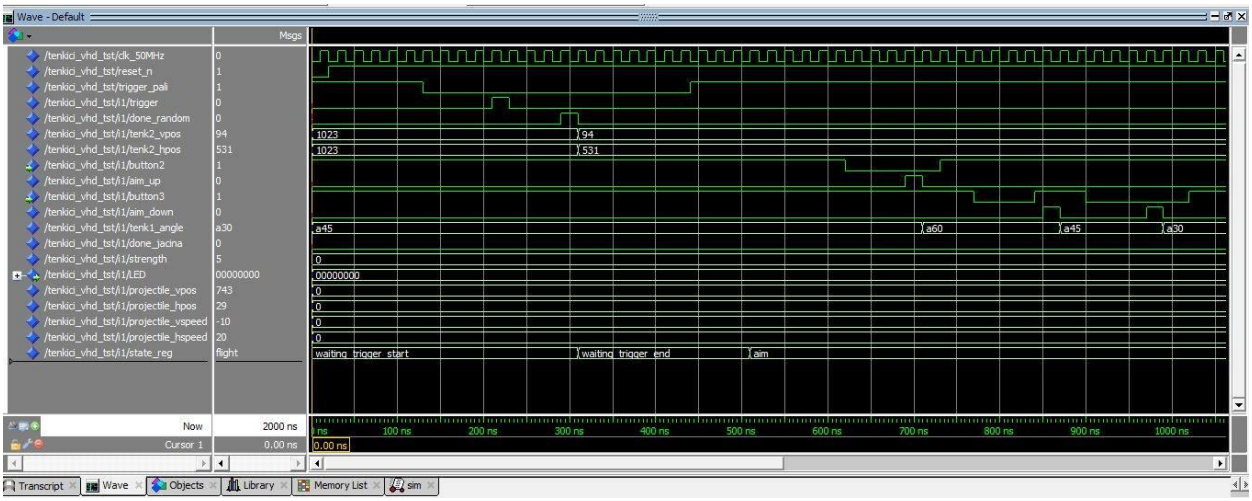
Ova komponenta predstavlja interfejs prema VGA kontroleru na razvojnoj ploči. Služi za generisanje odgovarajućih izlaznih signala pomoću kojih se vrši iscrtavanje na VGA displeju. Kod za ovu komponentu je preuzet sa časova vežbi iz predmeta *Uvod u projektovanje VLSI sistema*.

2.9. crtanje

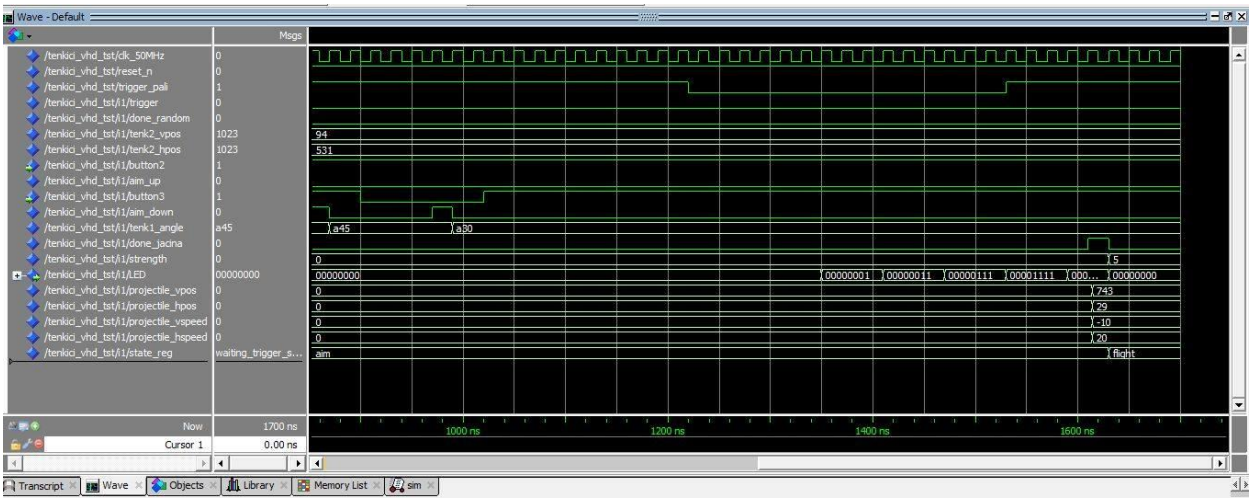
Funkcija ove komponente je iscrtavanje svega što se u igrici vidi, dakle iscrtava oba tenka, planinu i projektil u letu. Tačnije, ona prosleđuje komponenti *vga_sync* boju piksela koji trenutno iscrtava, piksel po piksel, u vidu izlaznog signala *color*. Na osnovu trenutne pozicije oba tenka, planine, projektila, kao i ugla cevi levog tenka, komponenta po prioritetima „gleda" koje boje treba da bude piksel koji se odrađuje. Ukoliko se piksel nalazi u jednom od dva okvira 32x32 piksela koji predstavljaju tenkove „poteže" se za bitmapom sačuvanom u primitivnoj ROM memoriji koja se nalazi u paketu za igru *Tenkići*. Ova komponenta ima sopstveni signal reseta, koji je aktivan kad se osnovna mašina stanja ne nalazi u stanjima *aim* ili *flight* (tačnije, displej je tada beo ili crn).

3. Testiranje i hardverska implementacija

Kao što je pomenuto, po završetku opisa svakog modula, prvo su pisani *testbench* fajlovi koji bi verifikovali ispravnost tog modula, a zatim po uspešno odrađenom testiranju i eventualnom ispravljanju grešaka, modul bi bio inkorporiran u celokupan sistem. Po sintezi celog sistema napravljen je konačni *testbench* fajl, koji ispituje funkcionalnost sistema. Na slikama 3.1 i 3.2 prikazani su rezultati simulacije za startovanje igrice i nišanjenje, kao i za ispaljivanje projektila, redom.

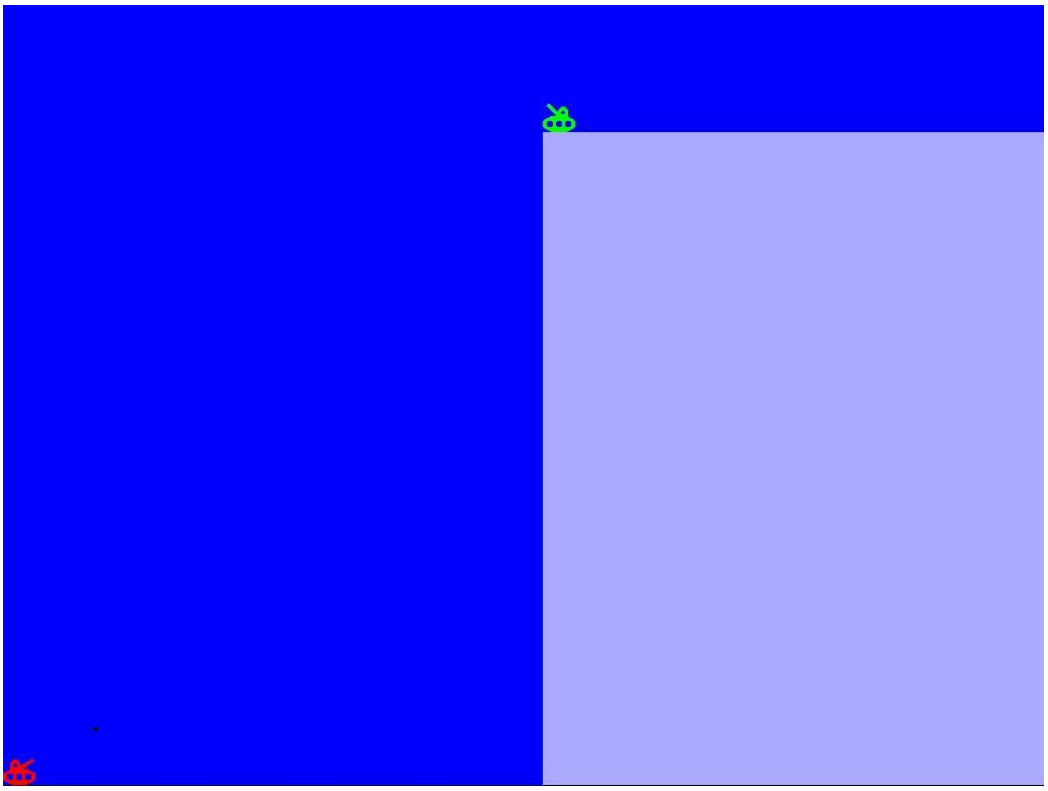


Slika 3.1 – Rezultati simulacije za startovanje igrice i nišanjenje



Slika 3.2 – Rezultati simulacije za ispaljivanje projektila

Ograničenje u testiranju je predstavljao nedostatak razvojne ploče i VGA displeja, koji bi omogućili vizuelni prikaz izgleda igrice. Pomenuti VGA simulator je stoga bio od značaja, međutim pomoću njega dobijeno samo par iscrtanih frejmova, jer bi u suprotnom ulazni fajl za iscrtavanje bio ogroman. Na slici 3.3 prkazan je izgled jednog frejma koji je dobijen pomoću simulatora.



Slika 3.3 – Izgled jednog frejma dobijen na osnovu VGA simulatora

Implementacija projekta na razvojnoj ploči (faza 2 projekta) je obavljena uspešno. Pri pravljenju TCL skripte za mapiranje pinova na razvojnoj ploči sa signalima u glavnom modulu sistema korišćena je električna šema DE1-SoC ploče [6]. Na „odbranu“ su donesene dve verzije projekta, sa resetom na prekidaču i na jednom od 4 tastera, a izabrana je prva. Nakon dodatnih sitnih izmena kod je ponovo uspešno spušten na ploču, a zatim je obavljeno par „estetskih“ izmena: boja projektila je promenjena u žutu, brzina malo povećana, itd. Modifikacija zadata od strane dežurnog asistenta je bila dodavanje uticaja vetra u igricu. To je odrađeno analogno kao gravitacija, tako što su se u procesu odabiranja slučajne pozicije protivničkog tenka očitavala i 2 najniža bita sa LFSR registra u cilju dobijanja slučajnog smera i jačine vetra pre svake nove partije igre.

Zaključak

Cilj projektovanja je bilo upoznavanje sa softverskim dizajnom digitalnih VLSI (*Very-large-scale integration*) sistema putem strukturalnog dizajniranja, to jest projektovanja manjih zasebnih komponenti i spajanja u kompletan sistem. Kroz pisanje koda u VHDL-u, simuliranje u *ModelSim*-u i testiranje na razvojnoj ploči mnogo je naučeno. Zahvaljujuci brojnim greškama koje su se javljale tokom rada, još više je naučeno. Sudeći po atmosferi u laboratoriji tokom testiranja igranja igrice, niko nije ostao nezadovoljan. Zaključuje se da je ostvareni interfejs zadovoljavajući, a sama igra pomalo teška zbog malih dimenzija tenkova i diskretnih vrednosti nišanjenja i jačine ispaljivanja. Kao mogućnost za nadogradnju, razmišljalo se o dodatnom prikazivanju eksplozije, uvođenju boljeg interaktivnog menija sa ispisivanjem reči i dodatnim slikama, kao i mogućnosti da igru igraju dva igrača. Generalno, smatra se da je projekat *Tenkići* uspešno odrađen.

Literatura

[1] http://tnt.etf.rs/~oe4upv

[2] http://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html#_Toc52606134

[3] ftp://ftp.altera.com/up/pub/Altera_Material/Boards/DE1-SoC/DE1_SoC_User_Manual.pdf

[4] http://ericeastwood.com/lab/vga-simulator/

[5] http://www.eng.auburn.edu/~strouce/class/elec6250/LFSRs.pdf

[6] http://tnt.etf.bg.ac.rs/~oe4upv/projekat/DE1-SoC-schematic.pdf

Paper title: "Project: Tanks – the game"

Abstract

This paper describes the procedure of designing a simple digital system which enables playing the game *Tanks* on a VGA (*Video Graphics Array*) display, by using VHDL (*VHSIC Hardware Description Language*) and the Altera Quartus II 15 software for programming and simulations. The hardware implementation was done on the DE1-SoC Development Kit platform built around the Altera Cyclone V SoC (*System-on-Chip*) FPGA (*Field-Programmable Gate Array)* chip. The ideas (and problems) in the preparation of each of the design phases have been concisely explained. The paper includes images of simple corresponding electrical schemes and corresponding simulations.