KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Magistro darbas

Atvirojo kodo vaizdo generavimo modulių realizacijų, skirtų FPGA matricoms, tyrimas bei tobulinimas

Darbą atliko: Andrius Riešutas, IFM-9/5

Darbo vadovas: prof. Eduardas Bareiša

Kaunas, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Magistro darbas

Atvirojo kodo vaizdo generavimo modulių realizacijų, skirtų FPGA matricoms, tyrimas bei tobulinimas

Recenzentas

doc. dr. E. Toldinas

2011-05-

Vadovas

prof. E. Bareiša

2011-05-

Atliko

IFM-9/5 gr. stud.
Andrius Riešutas
2011-05-26

Kaunas, 2011

Santrauka

Vis labiau modernėjant technologijoms, atsirandant naujoms specifinėms sistemoms ir poreikiams joms, turi būti tobulinamos esamos sistemos, kad atitiktų tiek profesionalų, tiek mėgėjų lūkesčius. Sistemos turi būti analizuojamos, taisomos bei pritaikomos atskirom naudotojų grupėm. Taigi visos sistemos turi būti reikalingos vartotojui ir patikimos, kad jos būtų eksploatuojamos. Ne išimtis yra ir vaizdo generavimo moduliai.

Prieš atliekant sistemų tobulinimus būtina atlikti tyrimus pasirinktai sričiai. Magistro tiriamojo darbo metu buvo surinkta medžiaga apie esamas atviro kodo vaizdo generavimo modulių realizacijas pritaikytas FPGA matricoms. Atlikta esamų projektų apžvalga, ištirti reikalavimai naudojamai sistemai bei jų veikimo savybės. Pasirinktoms realizacijom buvo atliktas vaizdo kodavimo bei perdavimo i ekraną tyrimas, taip pat atlikta projektų sintezacija, bei gautos naudojamų komponentų, laikinės ir energijos sunaudojimo ataskaitos.

Atlikus darbą buvo pastebėta, kad atviro kodo realizacijos padeda labiau susipažinti su kūrimo specifika, atsiradusiom problemom bei jų sprendimo būdais. Pastebėta, kad atviro kodo realizacijos yra tinkamos mokomiesiems tikslam, bei jas tiriant labiau priartėjama prie praktinio kūrimo.

Summary

In an increasingly technological modernization, when new specific systems and needs for them appear, current systems must be improved to meet both professional and user needs. It must be analyzed, corrected and adjusted for separate groups of users. Entire systems must be reliable and must be used to remain. Moreover, there is no exception for image modules.

Before making improvements there must be an investigation for selected areas. In masters degree research work the material was collected on existing open-source implementations of image module for FPGA arrays. Furthermore, the review on existing projects was made, in addition to this, examination on requirements and operating characteristics for system. The research was made on selected implementations, while getting more information how does the video encoding and transfer to the screen is done. Also project was synthesized, and the report of components used, power consumption and temporal information was done.

After the work was done, I have noticed that open-source implementations help become more familiar with the specifics of the development, face with errors and ways how to solve them. Those implementations suits for educational purposes, to get closer to the practical work.

Turinys

1. Įvadas

Vis labiau modernėjant technologijoms, atsirandant naujoms specifinėms sistemoms ir poreikiams joms, turi būti tobulinamos esamos sistemos, kad atitiktų tiek profesionalų, tiek mėgėjų lūkesčius. Sistemos turi būti analizuojamos, taisomos bei pritaikomos atskirom naudotojų grupėm. Taigi visos sistemos turi būti reikalingos vartotojui ir patikimos, kad jos būtų eksploatuojamos. Ne išimtis yra ir vaizdo generavimo moduliai, taip gali būti apibrėžiamos ir vaizdo plokštės.

Vaizdo plokštės (kitaip video adapteriai, vaizdo kortos, grafikos palaikymo įrenginiai) yra skirtos sugeneruoti ir perduoti vaizdą iš kompiuterio į displėjų. Vaizdo kortos gali generuoti 2D ir 3D vaizdus, taip pat gali dekoduoti MPEG-2 ar MPEG-4 video signalų formatus, gali turėti įrašymo funkciją, prievadą į televizorių ir daug kitų galimybių.

Pirmoji video plokštė buvo sukurta 1981 metais įmonės IBM. Tai buvo vienspalvio ekrano adapteris (MDA[*Monochrome Display Adapter*]), kuris veikė tik tekstinių rėžimu ir į ekraną gebėjo išvesti 80 stulpelių ir 25 eilutes (80x25). Šis įrenginys turėjo 4KB atminties ir tik vieną spalvą. Vėliau buvo pradėtos kurti plokštės, turinčios daugiau atminties, galinčios į ekraną išvesti jau ir spalvotus piešinius.

1987 m. IBM sukūrė vaizdo plokštę su VGA(*Video Graphics Array*) prievadu tai tapo kompiuterių analoginių ekranų standartu. Paplitus tokioms plokštėms jos buvo pradėtos tobulinti tokių įmonių kaip ATI, Cirrus Logic ir S3. Šios įmonės siekė padidinti galimą vaizdo rezoliuciją ir spalvų skaičių. Taigi tobulinant IBM sukurtą vaizdo plokštę buvo pasiekta 2MB vaizdo atminties ir jos galima rezoliucija buvo 1024x768 esant 256 spalvom. Taip atsirado SVGA (*Super Video Graphics Array*) standartas.

Tobulinant vaizdo plokštes 1995 metais buvo išleista SVGA plokštė su 3D vaizdavimo funkcija. Toliau sekė galimybių plėtimas ir naujų idėjų kūrimas.

Vaizdo plokštės yra naudojamos daugeliui veiksmų. Ne paslaptis, kad labiausiai video plokštės naudojamos mėgėjų, kompiuteriniams žaidimams. Taip pat jos yra naudojamos inžineriniams tikslams, sudėtingų grafinių brėžinių kūrimui, video medžiagos apdorojimui ir kt. Taigi turi būti atsižvelgiama į vartotojų norus, kuriuos galima realizuoti naudojant atviro kodo vaizdo generavimo modulius pritaikytus FPGA matricoms.

FPGA(field-programmable gate array) matrica tai integrinė mikroschema skirta konfigūruoti pagal vartotojo poreikius naudojant aparatūros aprašymo kalbą HDL(hardware description language). Pagrindiniai šių plokščių gamintojai yra Xilinx, Altera, Lattice, Actel, Quicklogic ir SiliconBlue, kurios yra įvairiausių specifikacijų, pasirenkamos pagal gaminamo produkto užimamą vietą ir

reikalaujamus resursus. Šiuo metu yra neišvengiamai tobulinamos norint išgauti aukštos raiškos (High Definition) video kokybę, kurios yra naudojamas apsaugos kameroms, medicinos tyrimams, bei kariuomenėje. Taip pat šios programuojamos matricos yra naudojamos dar nerealizuotos vienlustės sistemos funkcionavimui ir tyrimui. Visos sistemos prieš pradedant naudoti ar paleisti į rinką turi būti dokumentuojamos, analizuojamos bei testuojamos. Kaip bebūtų testuojant atsiranda klaidų, kurias reikia ištaisyti. Tačiau kai kurios klaidos išryškėja jau po sistemos išleidimo į rinką. Taigi tokiu atveju sistemai palaikyti, bei tobulinti turi būti aptarnaujamasis personalas.

Prieš atliekant tyrimą reikia apžvelgti esamas atviro kodo vaizdo generavimo modulių realizacijas pritaikytas FPGA matricoms, jų privalumus bei trūkumus, elgseną ir išsiaiškinti ką dar galima patobulinti ir kaip tą padaryti.

1.1 Tikslas

Surinkti literatūrą apie vaizdo generavimo modulius, bei surasti informacijos apie esamas atviro kodo vaizdo generavimo modulių realizacijas, pritaikytas FPGA matricoms.

Apžvelgti esamų realizacijų galimą tobulinimą.

1.2 Uždaviniai

Išanalizuoti surinktą literatūrą, apibendrinti rastą informaciją apie esamas realizacijas.

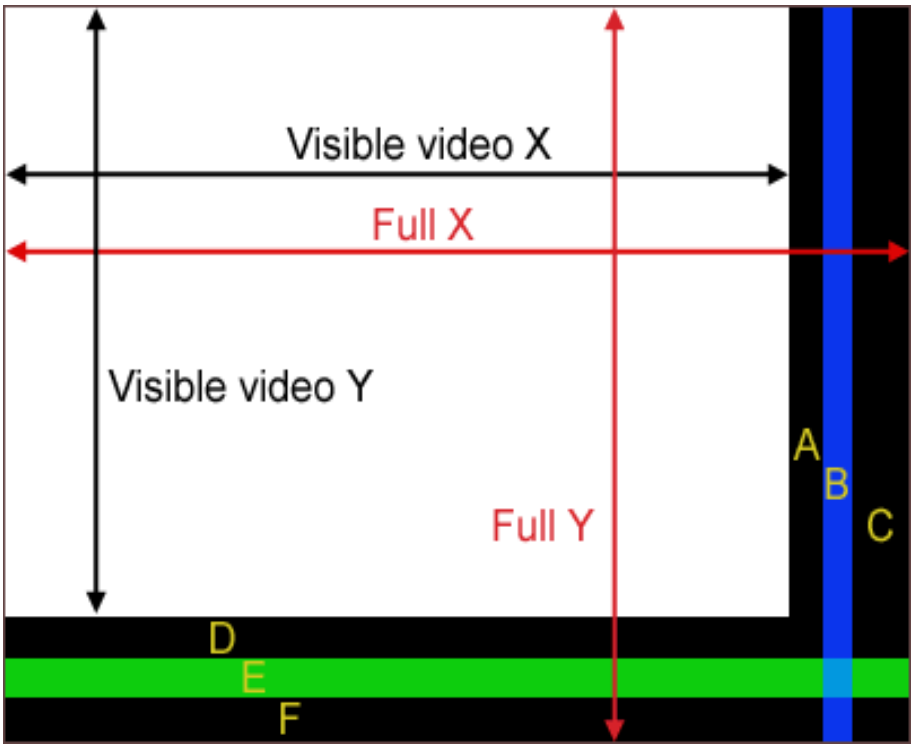
Atlikti realizacijų tyrimą, atlikti sintezavimą įvairiais programiniais sintezavimo, kūrimo paketais, atliekant sintezavimą ištaisyti atsirausiusias klaidas, gauti ataskaitinę informaciją apie susintezuotą projektą.

Aptarti kokie gali būti patobulinimai pasirinktoms realizacijoms

2. Literatūros apžvalga

2.1 VGA signalo generavimas

Tiriant vaizdo realizacijų projektus pravartu pasidomėti, kaip yra generuojamas VGA signalas. Toliau matome iliustraciją kaip yra generuojamas signalas ekrane (1pav.)



1 pav. VGA signalo generavimas

Šioje iliustracijoje matome, kad baltoji zona yra matomoji zona ekrane, kitos dalys nematomos, tačiau svarbios.

VESA(Video Electronics Standards Association) – tarptautiniai standartai, naudojami kompiuterinės grafikos kūrimui. VESA numato daug standartinių video signalų ir raiškų variantų. Raiškų variantų būna nemažai, priklausomai nuo ekrano dydžio ir tipo, pavyzdžiui: 800×600, 1024×768, 1280x780 ir kitos. Taip pat nuo displėjaus palaikymo yra naudojamas kadrų dažnis, kurių būna įvairių, pavyzdžiui 60, 70, 75 … 100Hz. Taip pat yra naudojamas taškų laikrodis (pixel clock) ir eilučių dažnis, sinchronizacijos signalai, ilgis, vieta ir standartizuotas poliariškumas.

Kuriant VGA signalą yra skaičiuojamos x ir y koordinatės ne iki standartinio ekrano dydžio, bet iki viso signalo galo. T.y. VGA režime x reikia skaičiuoti ne iki 640 taškelio bet iki 800. Tada įmanoma lengviau realizuoti sinchro signalus- po vaizdo palaukiama 16 taškų(A), po to vėl paleidžiamas sinchro signalas ir palaikoma 96 taškai (B), tada pašalinamas sinchro signalas ir

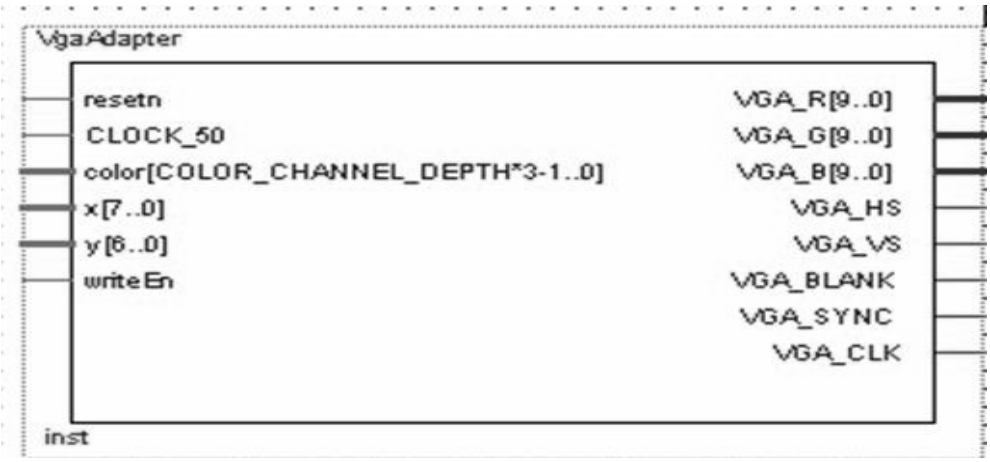
palaikoma iki 48 taškų, ir laukiama iki naujos eilutės pradžios(C). Tas pats atliekama ir su y koordinate. 1 lentelėje pateikti kai kurie VESA standartai.

1 lentelė VESA standartai

Mode	Pixel clock, MHz	VS, Hz	HS, kHz	Sinchro poliariškumas	x	y	A	B	C	D	E	F
VGA 60	25.17	60.04	31.46	- -	640	480	16	96	48	11	2	31
VGA 75	31.5	75	39.38	- -	640	480	16	96	48	11	2	32
VESA 1024×768	65	60	48.36	- -	1024	768	24	136	160	3	6	29
VESA 1280×1024	108	60	64	+ +	1280	1024	48	112	248	1	3	38

Dažniai gali šiek tiek skirtis, nes displėjus gali prisiderinti prie nedidelių nukrypimų. Tačiau patartina daryti kuo tiksliau. Darant nedidelius projektus su FPGA plokštėmis ir naudojant VGA rėžimą galima pernelyg nepaisyti elementų vėlinimo, bet gaminant sudėtingesnius, patartina ištestuoti prieš tai esamus parametrus, kad neatsirastų per dideli vėlinimai.

VGA rėžimas veikia panašiai kaip ir skaitmeninė atmintis, kuri yra atvaizduojama displėjuje. Paveikslėlyje nr.2 yra pavaizduota VGA adapterio blokinė diagrama.



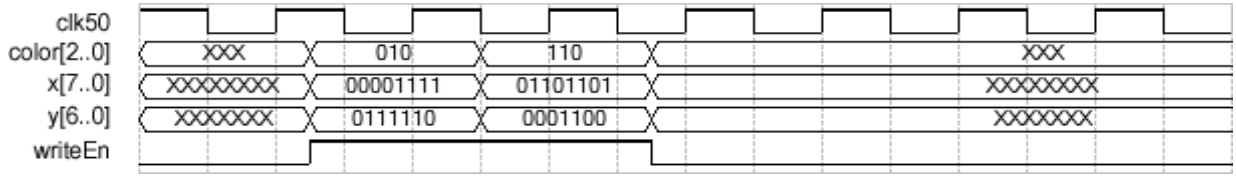
2 pav. blokinė VGA adapterio diagrama

(http://www.eecg.utoronto.ca/~jayar/ece241_06F/vga/i/VGAadapter.jpg)

Šioje blokinėje diagramoje matome, kad laikrodis(clock) yra CLOCK_50=50Hz, resetn – perkrovos signalas, nustatantis įėjimus į pradinę būseną, color[2..0] – 3 bitų spalvų magistralė, color[2] – raudona spalva, color[1] – žalia, color[0] – mėlyna. X[7..0] ir y[6..0] nurodo koordinates,

writeEn – naudojama taškų siuntimo informacija į VGA adapterį, spalva ir koordinatė yra siunčiama priekiniu frontu. VGA_R[9..0], VGA_G[9..0], VGA_B[9..0] – 10 bitų magistralės prijungtos prie skaitmeninių – analoginių keitiklių, pavertus signalą į analoginį jis yra perduodamas į displėjų. VGA_HS, VGA_VS – horizontalūs ir vertikalūs sinchrosignalai. VGA_BLANK – signalas aktyvuojantis VGA adapterį. VGA_CLK – išėjimo sinchrosignalas.

3 paveikslėlyje pavaizduota laikinė diagrama atvaizduojanti du taškus ekrane. Pirmasis yra žalias (color 010), kuris yra įrašomas į koordinates 15 ir 62, antroji spalva (color 110) įrašoma į koordinates 109 ir 12.



3 pav. Laikinė diagrama atvaizduojanti dvi spalvas koordinatėse
(http://www.eecg.utoronto.ca/~jayar/ece241_06F/vga/i/screen-timing.gif)

Šiuo metu yra ir kitokių signalų perdavimo būdų nei VGA. Kadangi daug įrenginių yra skaitmenizuojama, tokių kaip displėjai ir panašiai, reikalaujami didesni duomenų perdavimo greičiai, kuriamos skaitmeninės sąsajos su didesniais pralaidumais, pavyzdžiui DVI(Digital Visual Interface).

DVI sąsaja galima išgauti 1920×1200 rezoliuciją esant 60Hz signalui, greičiu 3.96 Gbit/s, jei vienu metu pajungiant dvi DVI sąsajas galima išgauti 2560×1600 esant 60Hz, greičiu 7.92Gbits/s.

2.2 FPGA programuojamų matricų apžvalga

Rinkoje pasirodžius FPGA (Field Programmable Gate Array) ir CPLD (Complex Programable Logic Device) programuojamosios logikos lustams, sutrumpintai vadinamiems PLL, atsirado galimybė projektuoti mažesnių matmenų spausdintines plokštes. Vis didėjanti programuojamojo lauko apimtis, universalumas ir sparčiai mažėjanti PLL kaina didina šio tipo schemų populiarumą ir paplitimą. Anksčiau norimai loginei funkcijai atlikti reikėdavo keleto ar keliolikos lustų, o dabar naudojamas vienas FPGA ar CPLD lustas. Programuojamosios logikos lustas gali būti pasirinktas priklausomai nuo reikiamo jo dydžio: loginių elementų ir trigerių skaičiaus; reikiamos maitinimo ir signalų įtampos (1,5; 3,.5; 5V).

Šio tipo lustų naudojimo teigiamybės:

1. sutaupoma nemažai spausdintinės plokštės ploto, nes sudėtingą skaitmeninę projekto dalį galima gauti naudojant vieną lustą. Taip suprojektuojama kompaktiška spausdintinė plokštė su viena logine mikroschema;

2. galima sumažinti projektavimo trukmę, kadangi spausdintinė plokštė gali būti projektuojama ir gaminama lygiagrečiai su skaitmeninės dalies projektavimu;

3. pagamintos plokštės atliekamą funkciją galima lengvai pakeisti perprogramavus CPLD arba FPGA atminties lustą, tam nereikia jos iš naujo projektuoti ir gaminti; padarius klaidą, užtenka perprogramuoti lustą. Kiekviename projektavimo etape galima keisti schemos funkciją;

4. galima padidinti schemos veikimo taktinį dažnį ir apdorojamų duomenų srautą; galima paspartinti ir supaprastinti skaitmeninio įtaiso projektavimą panaudojant jau sukurtus tipinių funkcinių mazgų bibliotekos elementus;

5. projektuotojo kūrinys yra saugus, niekas negali jo nu-kopijuoti.

Plačiai naudojamos šios XILINX gaminamos FPGA lustų šeimos: XC4000, XC4000XL, XC4000XV, Virtex, Spartan, Virtex II, Spartan IIE ,Spartan II, Virtex II Pro, taip pat CPLD lustų šeimos: XC9500, XC9500XL, XC9-500XV, CoolRunner, CoolRunner II. Kiekviena lustų šeima – tai plati lustų įvairovė, apimanti loginius elementus, trigerius, daugintuvus, atminties elementus, įėjimų ir išėjimų prievadus, taip pat ir skirtingus korpusus. Virtex II Pro programuojamosios logikos lustuose integruoti net PowerPC mikroprocesoriai (Abraitis, Bareiša, 2003).

3. Egzistuojančių atviro kodo vaizdo plokščių pritaikytų FPGA matricoms apžvalga.

3.1 VGA Sync testas

Ieškant informacijos apie vaizdo realizavimą ekrane, naudojant FPGA matricas, dažniausiai randamas pirminis daugelio atviro kodo realizacijų kūrėjų testavimo pavyzdys, kuriuo remiantis yra pradedamas kūrimas.

Ši realizacija yra sukurta naudojant VHDL programavimo kalbą, ir pritaikyta Spartan3E FPGA programuojamai matricai. Kūrėjas naudoja kompiuterio ekraną bei matricą.

Šioje realizacijoje rezultatas yra spalvos, išdėstytos stulpeliais, atvaizduotos ekrane.

3.2 TV Pong Game pritaikyta Actel FPGA matricai

Tai žaidimas – stalo tenisas (Ping Pong) aprašytas VHDL kalba. Ši vaizdo plokštė yra realizuota ant Actel APA075 ProASIC plus, tačiau ją galima testuoti ir ant kito tipo FPGA plokščių, pridėjus išorinį 25.176MHz laikrodį (clock). Išorinis laikrodis gali būti ir nevisiškai tikslus, nes dabartiniai displėjai palaiko multi-sync funkcija – gali dirbti su dideliu dažnių diapazonu. Esamo projekto puslapyje yra pateiktas VHDL kodas ir aprašymas kaip viską padaryti. Projekte yra naudojama ši techninė ir programinė įranga:

- Modelsim 5.7e
- HDL_Designer 2003b
- Precision 2003b-update1
- Actel Designer 5.0
- Trys 270 omų varžos.
- Viena 330 omų varžą.
- Viena HD15 jungtis.
- Actel ProASICplus Starter Kit
- 2 nebrangūs žaidimo pultai.
- Multi-Sync funkciją turintis displėjus.

Actel ProASICplus tipinės charakteristikos:

- Kūrimo apsauga
- Perprogramuojama plokštė
- Projektas įjungiamas įjungiant plokštę
- Naudoja mažai energijos
- Įėjimo ir išėjimų fazių bei energijos optimizavimas
- Galimybė naudoti ir automobiliuose ir armijos tikslams
- Turi JTAG išplėtimą, PCI lizdą, VGA, USB ir kitus

Projekte taip pat yra aprašytas visas žaidimo paaiškinimas, bei kaip jį realizuoti FPGA matricoje.

3.3 Pong game pritaikyta Spartan3E matricai

Realizacija pritaikyta Xilinx Spartan3E programuojamai matricai. Projektui yra panaudoti žaidimo valdymo pultai, pritaikyti matricai. Ši realizacija yra sukurta naudojant VHDL programavimo kalbą. Naudojama taip pat monohromatinė spalvų gama.

Panaudotos priemonės:

- Xilinx Spartan-3E starter kit
- Xilinx ISE design suite
- Valdymo pulteliai
- Adapteris valdymo pulteliams
- Ekranas

Xilinx Spartan-3E specifikacijos:

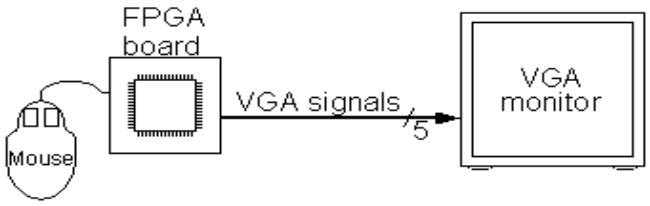
2 lentelė Spartan-3E matricų palyginimas

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)				Distributed RAM bits(1)	Block RAM bits(1)	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs	Total Slices						
XC3S100E	100K	2,160	22	16	240	960	15K	72K	4	2	108	40
XC3S250E	250K	5,508	34	26	612	2,448	38K	216K	12	4	172	68
XC3S500E	500K	10,476	46	34	1,164	4,656	73K	360K	20	4	232	92
XC3S1200E	1200K	19,512	60	46	2,168	8,672	136K	504K	28	8	304	124
XC3S1600E	1600K	33,192	76	58	3,688	14,752	231K	648K	36	8	376	156

Šios realizacijos kodas ir aprašymas yra pateiktas literatūros sąraše.

3.4 Pong Game pritaikyta Altera FPGA matricai

Panašus projektas su prieš tai esančiais, tačiau šiek tiek paprastesnis, nes kamuoliukas atsitrenkia į displėjaus „sieneles“ ir į pagrindą, kuris yra valdomas į šonus kompiuterine pele. Žaidimas yra sukurtas ant Pluto FPGA matricos ir realizuotas Verilog kalba. Toliau pateikta projekto naudojamų techninių prietaisų schema(4pav.)



4 pav. Naudojamų prietaisų schema

Projektas yra realizuotas firmos Alter a sukurtos FPGA plokštės serijos „Pluto“. 3 lentelėje pateiktos „Pluto“ plokščių charakteristikos.

3 lentelė Altera Pluto plokščių charakteristikos

Board	Pluto	Pluto-II	Pluto-3
FPGA (vendor page)	EP1K10	EP1C3	EP2C5
Datasheet (PDF)	ACEX 1K	Cyclone	Cyclone II
Logic cells	576	2910	4608
IO pins	41	51	65
PLL	No	Yes	Yes
External clocks	up to 2	up to 4	up to 4
Boot-PROM	No	1 Mbits	4 Mbits
On-board oscillator	25MHz	25MHz	25MHz
DIL8 oscillator header	No	No	Yes
Push-button	No	No	Yes
JTAG header	No	No (1)	Yes
LED(s)	1	1	2
Flashy ready	Flashy	Flashy	FlashyD
Dimensions	58x28mm	58x28mm	58x41mm

Kitos charakteristikos:

- Lengvai naudojama – valdoma per asmeninio kompiuterio RS232 išvadą.
- Įmanomas eksperimentinis įrenginių praplėtimas, lituojant papildomus įrenginius.
- Galimybė pajungti išorinius prietaisus.
- Galimybė naudoti nuolatinės srovės šaltinius.

Šiame projekte yra pateiktas projekto kodas ir aprašymas.

3.5 Galaxy projektas realizuotas ant Spartan-3E matricos

Realizacija – kosmoso šaudyklė. Šio projekto principas yra su erdvėlaiviu nukauti skraidančius ateivius. Realizacija pritaikyta FPGA matricai, bei panaudota 3bit spalvų gama. Taigi šis projektas naudoja ir spalvinius VGA kodus. Taip pat yra pritaikyti valdymo įrenginiai – nuotolinio valdymo pulteliai. Įranga taip pat panaudota kaip ir Pong realizacijoje pritaikytai Xilinx Spartan-3E matricai.

Šios realizacijos programinis kodas yra laisvai platinamas, bei turi veikimo aprašymą.

3.6 Space Invaders

FPGA matricoje realizuotas video žaidimas „Space Invaders“, projekte yra pateiktas veikimo principas, vhdl aprašai. Šis projektas yra sukurtas ant Spartan3E FPGA matricos. Šis žaidimas realizuotas tiek video tiek audio signalais ir yra skirtas displėjams su VGA prievadu. Valdymas vyksta klaviatūros pagalba.

Žaidimas vyksta kai kosminis laivas naikina keliomis eilėmis išsidėsčiusius skraidančius objektus. Šis projektas yra pritaikytas FPGA matricoms ir yra aprašytas VHDL kalba.

Xilinx Spartan3E charakteristikos:

- 50Mhz laikrodis (clock)
- 64MB operatyvioji atmintis
- 128MB atmintis
- Ethernet jungtis, JTAG prievadas, du RS232 prievadai, P/S 2 jungtis, 8 led diodai, 100 kojų išplėtimo galimybė.

3.7 Asteroids

Žaidimas realizuotas Xilinx FPGA matricoje Verilog kalba. Šis žaidimas yra dvimatėje erdvėje, kurioje erdvėlaivis šaudo ir saugosi atsitiktine tvarka atsirandančius asteroidus, kurie taip pat sukasi aplink. Pašovus asteroidą jis suskyla į mažesnes daleles, kurie vėliau pranyksta. Žaidimas yra baigiamas kai yra sunaikinami visi asteroidai arba erdvėlaivis atsitrenkia į asteroidą.

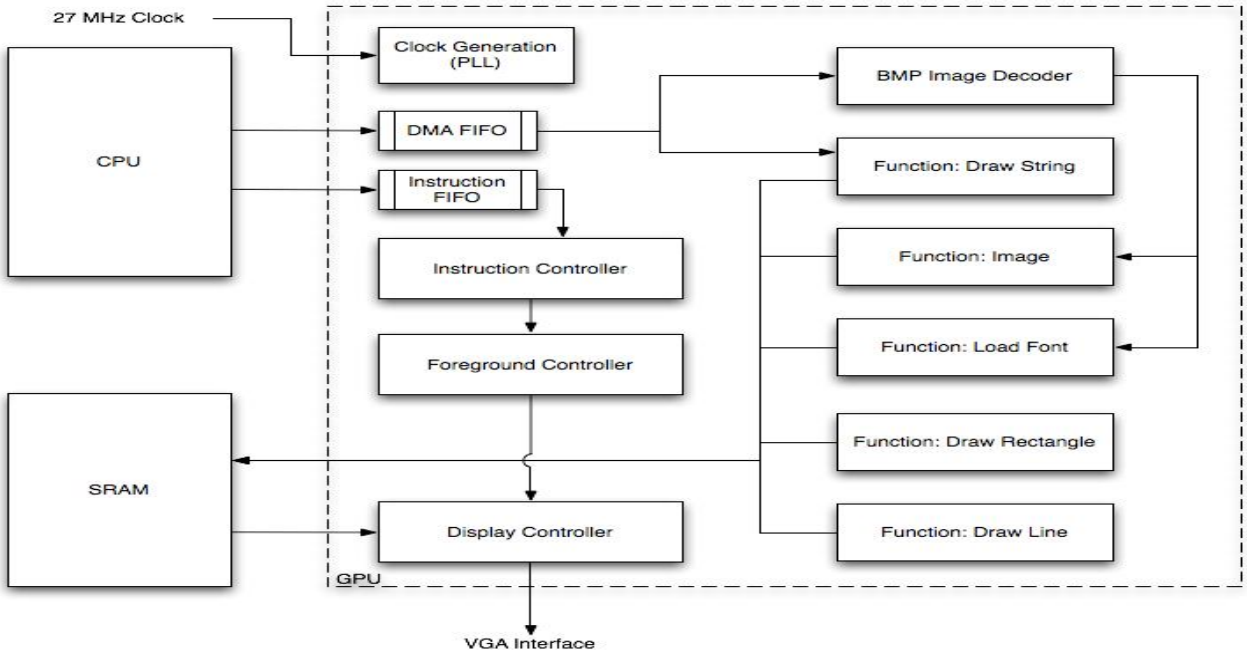
Naudojama įranga:

- FPGA matrica
- 8 šviesos diodai
- 9 Mygtukai
- 8 Jungtukai
- 16 Šešioliktainių išėjimų
- VGA displėjus
- 64 Bit Loginis analizatorius
- PS2 pelė ir klaviatūra

Projekte pilnai pateikta visa dokumentacija, diagramos ir Verilog kalba parašytas programos kodas.

3.8 2D graphics engine

Dvimatės(2D) erdvės kūrimo varikliukas. Šis projektas yra skirtas sukurti dvimačius vaizdus, pavyzdžiui ekrano užsklandą, kuri gali būti sustabdoma, pagreitinama ir panašiai. Šiame projekte yra pateikta informacija kaip vyksta visas kūrimo procesas, bei programos kodas VHDL ir Verilog kalbomis. Yra nurodyta, kad projektas buvo bandytas ant Nios II FPGA matricos. Toliau pateikta blokinė 2D kūrimo diagrama(5pav.)



5 pav. Blokinė 2D vaizdo kūrimo diagrama

(http://instruct1.cit.cornell.edu/courses/ece576/FinalProjects/f2009/Adam_Tom/graphics_engine_12_9_09/index.html)

Šis projektas gali būti taikomas tiriant 2D kūrimą ir veikimo principą.

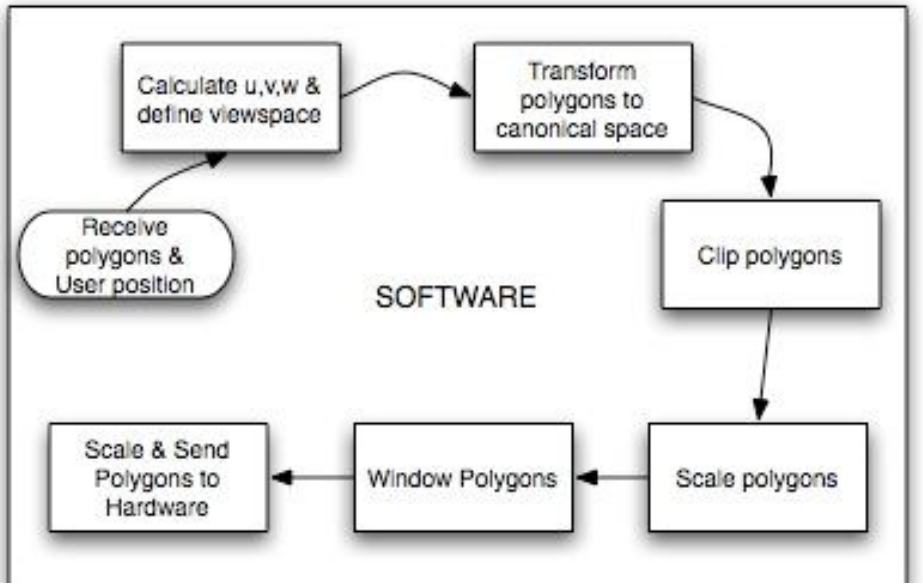
3.9 3D render engine

Šiek tiek sudėtingesnis projektas, kuriame yra paaiškinta kaip yra kuriami trimačiai(3D) vaizdai, projekto pagrindinis tikslas yra sukurti 3D figūras. Pagrindinis kūrimo principas remiasi geometrinėm ir spalvų ryškumo savybėm. Projekto kodas yra pateiktas Verilog kalba ir yra skirtas FPGA matricom.

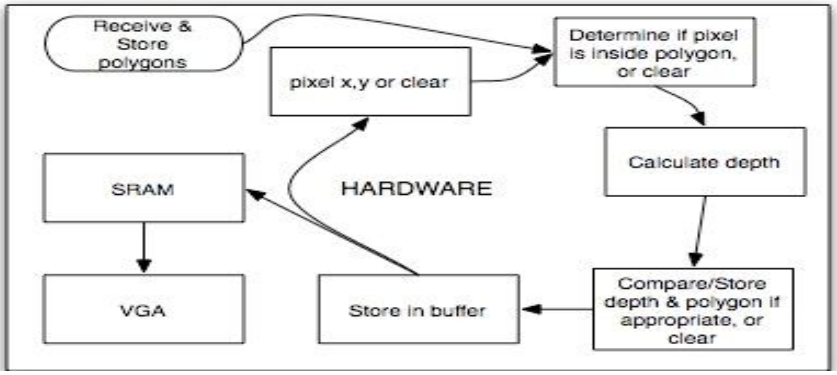
Šio projekto tyrimas padės išsiaiškinti kokie yra 3D vaizdų kūrimo principai.

3.10 SimpleGPU

GPU(Grahics processig unit) – prietaisas perduoti grafinius vaizdus. Šis projektas yra realizuotas NIOS II FPGA matricoje ir aprašytas Verilog ir C kalbomis. Taip pat yra pateiktos blokinės programinės(6pav.) bei techninės(7pav.) įrangos diagramos.



6 pav. Programinės įrangos blokinė diagrama
(http://instruct1.cit.cornell.edu/courses/ece576/FinalProjects/f2007/jas328_asl45/jas328_asl45/index.htm)



7 pav. Techninės įrangos blokinė diagrama
(http://instruct1.cit.cornell.edu/courses/ece576/FinalProjects/f2007/jas328_asl45/jas328_asl45/index.htm)

Kaip ir anksčiau minėtas „3D render engine“ šis darbas yra skirtas perduoti 3D vaizdus į kompiuterio displėjų.

4. Realizacijų tyrimas

Gaminant bet kokias realizacijas visada reikia atsižvelgti į aparatūrines bei programines galimybes. Kaip jau buvo minėta yra nemažai įmonių, kuriančių FPGA matricas, leidžiama daugelis šių programuojamų plokščių variacijų. Kiekvienas gamintojas nustato kokie sintezatoriai gali būti naudojami, kokia turi būti vartotojo aplinka, ar galima naudotis tik konkretaus gamintojo programiniu paketu ar sukurtą realizaciją galima pritaikyti ir kitoms vartotojo aplinkoms.
Pradedant tirti sudėtingesnes realizacijas galima apžvelgti ir paprastas, testinius programinius kodus, kuriuos dažniausiai kūrėjai perpratę naudoja sudėtingesnių realizacijų kūrimui.

4.1 VGA SYNC testinė realizacija

Kuriant realizacija reikia atsižvelgti į tai, kad VGA generavimo moduliai išduoda aštuonis signalus, šešis spalvų išėjimus bei du sinchronizacijos signalus(8 pav.):

```
1. entity vga is
2.          port(clk, reset : in std_logic;
3.                VGAout: out std_logic_vector(7
   downto 0);-(R,R1,G,G1,B,B1)
4.               );
5. end vga;
```

8 pav. testinio VGA išvedimo signalų pavyzdys

Taigi matome, kad pagrindiniai signalai yra spalvų išėjimai, bei sinchronizacijos signalo bei veikimo pertraukimo signalas (reset).

Sinchronizuojant signalus reikia atsižvelgti į vertikalių signalų atkūrimą, bei vėlinimą(9 Pav.).

```
1.  vcounter: process (clk, reset) begin
2.            if reset='1'then
3.                vcount <= 0;
4.            else if (clk'event and clk='1') then
5.                    if hcount=699 then
6.                        if vcount=524 then
7.                            vcount <= 0;
8.                            count<=count+1;
9.                        else
10.                           vcount <= vcount + 1;
11.                       end if;
12.                   end if;
13.               end if;
14.           end if; end process;
```

9 pav. vertikalus taškų atkūrimo skaitliukas ir vėlinimas

Atkūrus vertikalius taškus yra atkuriami ir horizontalūs, taigi reikia suderinamumą, taškų atkūrimo sinchronizaciją, tai atliekama kaip parodyta paveikslėlyje (10 pav.):

```
1.  sync: process (clk, reset)
2.       begin
3.            if reset='1'  then
4.                hsync <= '0';
5.                vsync <= '0';
6.            else
7.                if (clk'event and clk='1') then
8.                    if (hcount<=751 and hcount>=655)
    then
9.                        hsync <= '0';
10.                   else
11.                       hsync <= '1';
12.                   end if;
13.                   if (vcount<=494 and vcount>=493)
    then
14.                       vsync <= '0';
15.                   else
16.                       vsync <= '1';
17.                   end if;
18.               end if;
19.      end if;
20. end process;
```

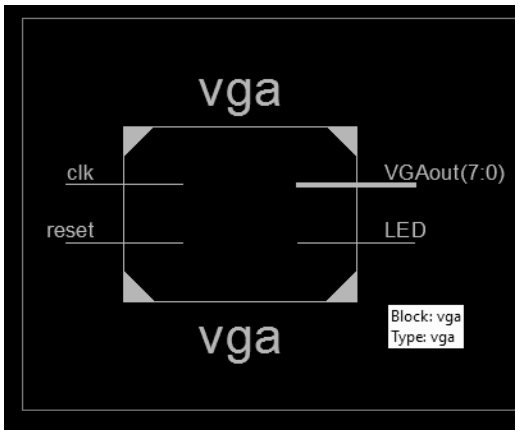
10 pav. vertikali ir horizontali sinchronizacija.

Taigi sudarius spalvų kodus ir susinchronizavus gauname testinį variantą, kai ekrane matoma nurodytų spalvų gama (11 pav.)

```
colors: process (clk, reset)
begin
    if reset='1' then
        VGA<=   "000000";
    elsif (clk'event and clk='1') then
        case hcount is
                when   0 to 39=>  VGA<= "000000";
                when  40 to 79=>  VGA<= "110000";
                when  80 to 119=> VGA<= "001100";
                when 120 to 159=> VGA<= "000011";
                when 160 to 199=> VGA<= "111100";
                when 200 to 239=> VGA<= "001111";
                when 240 to 279=> VGA<= "110011";
                when 280 to 319=> VGA<= "100000";
                when 320 to 359=> VGA<= "001000";
                when 360 to 399=> VGA<= "000010";
                when 400 to 439=> VGA<= "101000";
                when 440 to 479=> VGA<= "001010";
                when 480 to 519=> VGA<= "100010";
                when 520 to 559=> VGA<= "101010";
                when 560 to 599=> VGA<= "010101";
                when 600 to 639=> VGA<= "111111";
                when others=> NULL;
        end case;
```

11 pav. pseudo kodas, kuriame yra nurodomos spalvų koordinatės ir spalvų kodai

Susintezavus duotąjį kodą panaudojant Xilinx ISE design suite, gauname elektroninę modulio schemą(12 pav):



12 pav. blokinė VGA modulio schema

Išskleidžiant blokinę diagramą matome VGA modulį sudarančių komponentų lygį. Kadangi šį lygi sudaro gana daug elementų, šio lygio neatvaizduosiu, tik apžvelgsiu Design Suite atliktas ataskaitas.

Sintezavimo komponentų ataskaita

```
Advanced HDL Synthesis Report
Macro Statistics
# Counters                          : 3
 10-bit up counter                  : 2
 6-bit up counter                   : 1
# Registers                         : 9
 Flip-Flops                         : 9
# Comparators                       : 40
=========================================================================
Macro Statistics
# Registers                         : 35
 Flip-Flops                         : 35
=========================================================================
*                  Final Report                      *
=========================================================================
Design Statistics
# IOs                   : 11

Cell Usage :
# BELS                  : 172
#      GND              : 1
#      INV              : 2
#      LUT1             : 18
#      LUT2             : 29
#      LUT2_L           : 2
#      LUT3             : 10
#      LUT3_L           : 3
#      LUT4             : 50
#      LUT4_D           : 2
#      LUT4_L           : 12
#      MUXCY            : 18
#      MUXF5            : 4
#      VCC              : 1
#      XORCY            : 20
# FlipFlops/Latches     : 35
#      FDC              : 18
#      FDCE             : 10
#      FDE              : 7
# Clock Buffers         : 1
#      BUFGP            : 1
# IO Buffers            : 10
#      IBUF             : 1
#      OBUF             : 9
```

Laikinės ataskaitos

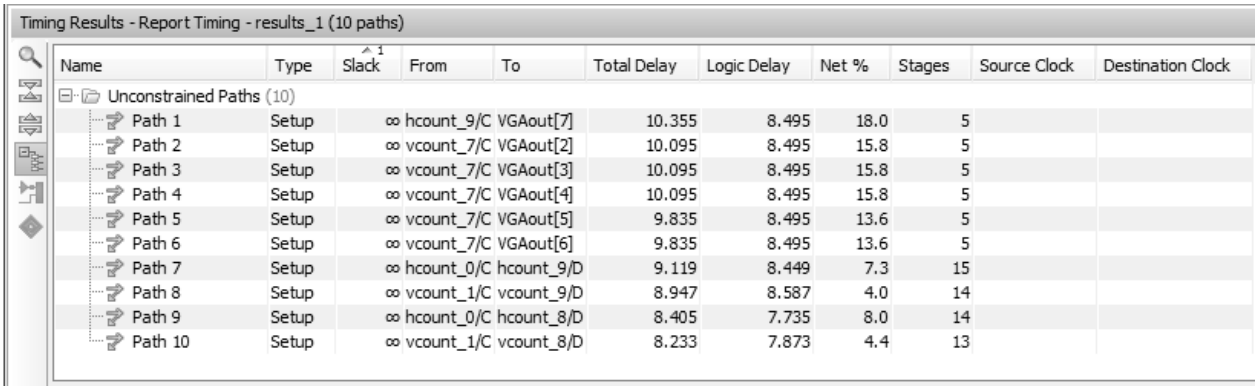
Minimum period: 5.731ns (Maximum Frequency: 174.500MHz)

Minimum input arrival time before clock: 3.994ns

Maximum output required time after clock: 7.209ns

Total memory usage is 188968 kilobytes

Netlist designer ataskaita (13 pav.)



Timing Results - Report Timing - results_1 (10 paths)

Name	Type	Slack	From	To	Total Delay	Logic Delay	Net %	Stages	Source Clock	Destination Clock
Unconstrained Paths (10)										
Path 1	Setup	∞	hcount_9/C	VGAout[7]	10.355	8.495	18.0	5		
Path 2	Setup	∞	vcount_7/C	VGAout[2]	10.095	8.495	15.8	5		
Path 3	Setup	∞	vcount_7/C	VGAout[3]	10.095	8.495	15.8	5		
Path 4	Setup	∞	vcount_7/C	VGAout[4]	10.095	8.495	15.8	5		
Path 5	Setup	∞	vcount_7/C	VGAout[5]	9.835	8.495	13.6	5		
Path 6	Setup	∞	vcount_7/C	VGAout[6]	9.835	8.495	13.6	5		
Path 7	Setup	∞	hcount_0/C	hcount_9/D	9.119	8.449	7.3	15		
Path 8	Setup	∞	vcount_1/C	vcount_9/D	8.947	8.587	4.0	14		
Path 9	Setup	∞	hcount_0/C	hcount_8/D	8.405	7.735	8.0	14		
Path 10	Setup	∞	vcount_1/C	vcount_8/D	8.233	7.873	4.4	13		

13 pav. VGA sync realizacijos Netlist designer ataskaita

Energijos bei temperatūros ataskaitos

Šios ataskaitos yra atliekamos „Xpower analyzer“ paketu.

Galimų temperatūrų ataskaita

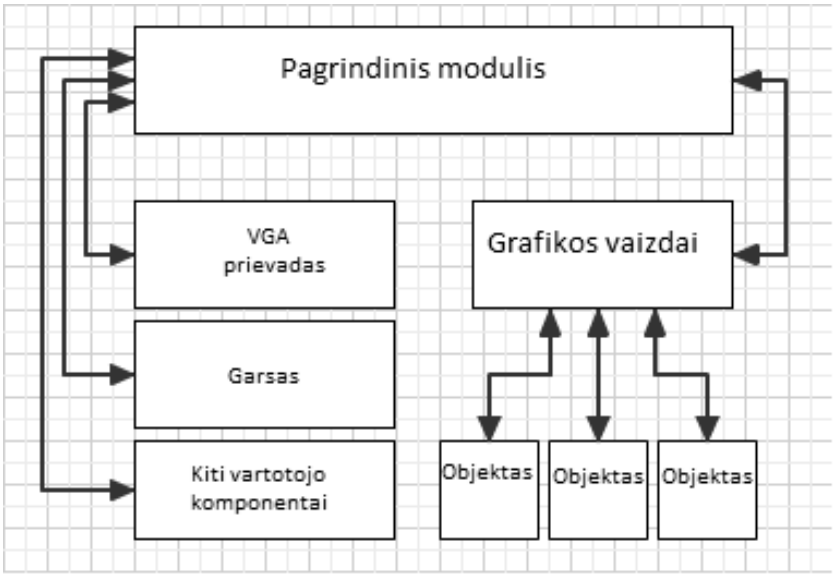
Thermal Summary	
Effective TJA (C/W)	52.1
Max Ambient (C)	83.4
Junction Temp (C)	26.6

Sunaudojamos energijos ataskaita

Power Supply Summary			
	Total	Dynamic	Quiescent
Supply Power (mW)	30.00	0.00	30.00

4.2 Grafinės realizacijos

Tiriant realizacijas reikia susidaryti planą, pagal kurį lengviau būtų analizuoti jau sukurtus atviro kodo projektus. Žemiau pateiktame paveikslėlyje matome pagrindinius realizacijų komponentus (14 pav.).



14 pav. realizacijų komponentai

Pagrindinis modulis apjungia visus realizacijos komponentus į visumą, t.y. VGA prievadą, garso modulius bei kitus vartotojo naudojamus komponentus, taip pat sugeneruotus grafikos vaizdus.

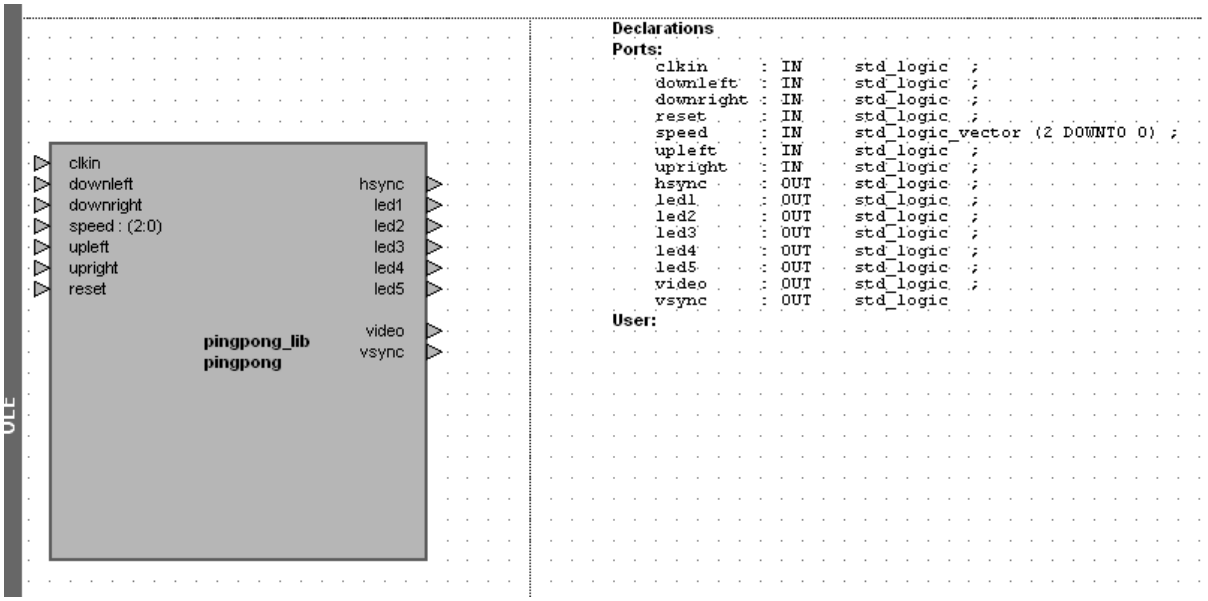
Per VGA prievadą yra vykdoma sinchronizacija tarp FPGA plokštės ir ekrano, taip pat perduodama vaizdo objektų taškų koordinatės bei savybės. Vaizdo objektai yra sudaromi realizacijos kūrėjo arba sugeneruojami iš jau esamo atvaizdo saugomo atmintyje.

4.2.1 TV Pong Game realizacija Actel FPGA

Tai žaidimas – stalo tenisas (Ping Pong) aprašytas VHDL kalba. Šis projektas yra realizuotas ant Actel APA075 ProASIC plus, tačiau jį galima testuoti ir ant kito tipo FPGA plokščių, pridėjus išorinį 25.176MHz laikrodį (clock). Išorinis laikrodis gali būti ir nevisiškai tikslus, nes dabartiniai displėjai palaiko multi-sync funkcija – gali dirbti su dideliu dažnių diapazonu.

TV Pong žaidimas yra sudarytas iš keletos komponentų, susijusių tarpusavyje tam tikrais ryšiais, taigi visi išėjimai (veiksmai) priklauso nuo užduodamų pradinių signalų (įėjimų). Vaizdo

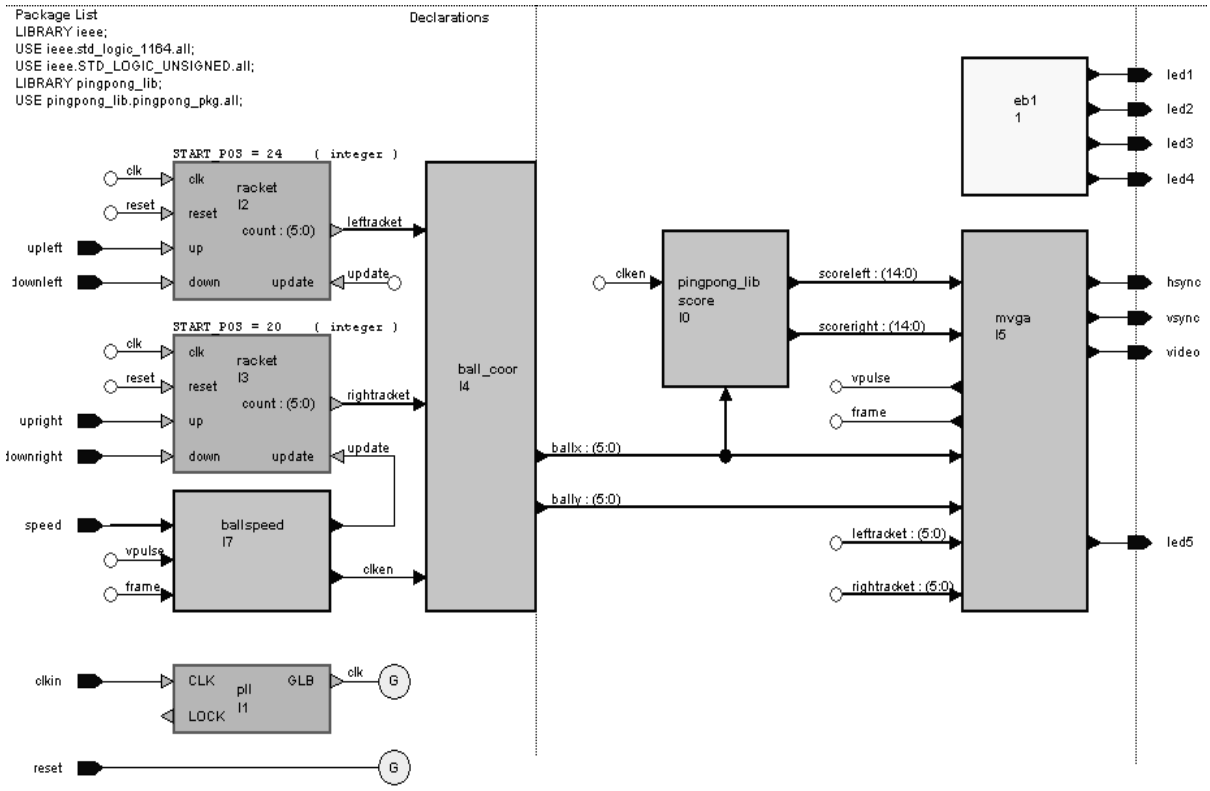
generuojamas atsižvelgiant į sinchro signalą. 15 paveikslėlyje matome blokinę TV Pong žaidimo diagramą, su atvaizduotais įėjimais ir išėjimais. Blokinė diagrama sumodeliuota panaudojant HDL Designer.



15 pav. Blokinė TV pong žaidimo diagrama

Taigi šiame paveikslėlyje matome, kad yra paduodamas sinchrosignalas, nuo kurio priklauso žaidimo veikimas. Signalai downleft, downright, upleft ir upright valdo viršutinę ir apatinę raketes. Signalai led1-led5 indikuoja paspaustus veiksmus, o vsync, hsync ir video perduoda signalus į monitorių.

Sekančiame paveikslėlyje(16 pav.) matome blokinės diagramos sandarą, atskirai pavaizduoti elementai ir ryšiai tarp jų.



16 pav. Blokinės diagramos komponentai ir ryšiai tarp jų.

RACKET(racket.vhd) – komponentai, kuriuose yra nustatomos rakečių savybės, jų judėjimo trajektorija(į kairę arba į dešinę), dydis.

BALLSPEED(ballspeed_rtl.vhd) – šis komponentas generuoja signalą (*clken*), kuris valdo komponentą ball_coor. Kamuoliuko greitis yra nustatomas šiame komponente. Kaip matėme 4 pav. greičio reikšmė yra vektorius, šiame projekte greičiausias lygis yra kai nustatoma reikšmė – „11“, o lėčiausias – „00“. Taip pat matome, kad iš šio komponento yra signalas „update“ į komponentą racket, kuriam šiuo signalu yra nustatomas raketės judėjimo greitis.

PLL(PLL.vhd) – komponentas, kuris sukuria 25.176MHz video sinchro signalą.

BALL_COOR(ball_coor_fsm.vhd) – komponentas kuriuo yra nustatomas kamuoliuko dydis ir judėjimo trajektorija.

MVGA(mvga_rtl.vhd) – mono-vga komponentas. Šis komponentas kuria HSYNC ir VSYNC signalus kurie nustato kamuoliuko, raketės ir taškų vietą monitoriuje, taip pat žalią spalvą(šiame projekte naudojama tik viena spalva, nes naudojant daugiau spalvų užimama daugiau vietos). Į vietą reikia atsižvelgti dėl to, kad ne visos FPGA plokštės turi pakankamai atminties.

SCORE(score_rtl.vhd) – komponentas, kuris skaičiuoja kiek kartų kamuoliukas atsitrenkia į priešingą pusę, ir priešininkui prideda tašką.

Taigi apjungus visus šiuos elementus, monitorių, gauname gana paprastą žaidimą, kurio pagalba galima ištirti vaizdo perdavimą į FPGA plokštes.

Pong realizacijos išvados bei ataskaitos

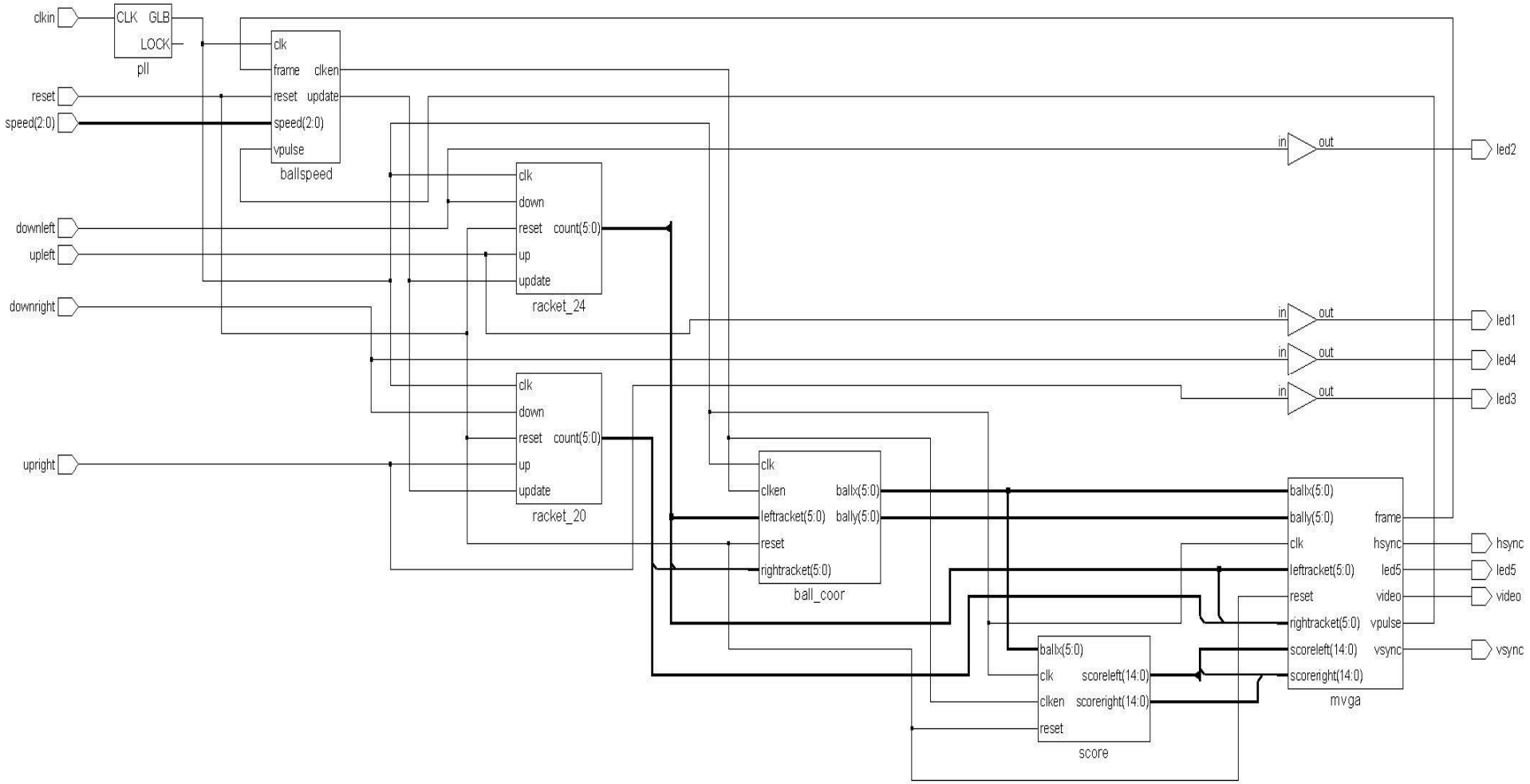
Panaudojus firmos Mentor Graphics programą, sudarytas schematinis Ping Pong žaidimo vaizdas, kuriame matomi kaip yra apjungti elementai, kokios sąsajos tarp jų, taip pat galime apžvelgti žemiausią arba aukščiausią lygmenį. Sudarytame schematiniame vaizde galima apžvelgti kiekvieno bloko sandara iš loginių elementų. Taip pat buvo gauta ataskaita apie projektą sudarančių elementų kiekį bei laikrodžio dažnį:

```
*****************************************************

Cell: pingpong    View: struct    Library: pingpong_lib

*****************************************************

Total accumulated area :
 Number of INBUF :                       9
 Number of OUTBUF :                      8
 Black Box PLLCORE :                     1
 Black Box PWR :                         1
 Number of Tiles :                     925
 Number of accumulated instances :     944
 Number of ports :                      17
 Number of nets :                      234
 Number of instances :                 209
 Number of references to this view :     0


 Cell        Library  References     Total Area

AND2         proasic3e       12 x      1     12 Tiles
AND2A        proasic3e       10 x      1     10 Tiles
AND2B        proasic3e        6 x      1      6 Tiles
AND3         proasic3e        4 x      1      4 Tiles
AND3A        proasic3e        1 x      1      1 Tiles
AND3C        proasic3e        4 x      1      4 Tiles
AO1A         proasic3e        2 x      1      2 Tiles
AO1B         proasic3e        8 x      1      8 Tiles
AO1C         proasic3e        2 x      1      2 Tiles
AO1E         proasic3e        2 x      1      2 Tiles
AOI1         proasic3e        2 x      1      2 Tiles
AOI1A        proasic3e        2 x      1      2 Tiles
AOI1B        proasic3e        2 x      1      2 Tiles
AX1          proasic3e        4 x      1      4 Tiles
AX1B         proasic3e        2 x      1      2 Tiles
AX1C         proasic3e        7 x      1      7 Tiles
AXOI5        proasic3e        1 x      1      1 Tiles
BUFF         proasic3e       14 x      1     14 Tiles
DFN1C1       proasic3e        2 x      1      2 Tiles
```

```
DFN1E0C1     proasic3e       19 x      1     19 Tiles
DFN1E0P1     proasic3e        7 x      1      7 Tiles
DFN1E1C1     proasic3e       13 x      1     13 Tiles
GND          proasic3e        2 x      1      2 Tiles
INBUF        proasic3e        9 x      1      9 INBUF
INV          proasic3e        8 x      1      8 Tiles
MX2          proasic3e        2 x      1      2 Tiles
NAND2        proasic3e        4 x      1      4 Tiles
NAND2A       proasic3e        1 x      1      1 Tiles
NAND2B       proasic3e        4 x      1      4 Tiles
NAND3        proasic3e        2 x      1      2 Tiles
NAND3A       proasic3e        7 x      1      7 Tiles
NAND3B       proasic3e        7 x      1      7 Tiles
NAND3C       proasic3e        3 x      1      3 Tiles
OA1A         proasic3e        2 x      1      2 Tiles
OUTBUF       proasic3e        8 x      1      8 OUTBUF
PLLCORE      pingpong_lib     1 x      1      1 PLLCORE
PWR          pingpong_lib     1 x      1      1 PWR
VCC          proasic3e        1 x      1      1 Tiles
XA1          proasic3e        7 x      1      7 Tiles
XAI1A        proasic3e        1 x      1      1 Tiles
XO1A         proasic3e        4 x      1      4 Tiles
XOR2         proasic3e        5 x      1      5 Tiles
XOR3         proasic3e        2 x      1      2 Tiles
ball_coor    pingpong_lib     1 x    370    370 Tiles
mvga         pingpong_lib     1 x    367    367 Tiles
```

```
*********************************************
Device Utilization for A3PE600PQFP208
*********************************************
Resource                Used    Avail   Utilization
-----------------------------------------------
IOs                     17      147      11.56%
Combinational modules   0       13824     0.00%
RAMs                    0       24        0.00%

-----------------------------------------------
Using wire table: proasic3e


                    Clock Frequency Report

      Clock                : Frequency
      ------------------------------------

      clk                  : 108.9 MHz
```

Taigi matome kiek elementų sudaro schemą. Schematinis vaizdas yra pavaizduotas 17 pav.



17 pav. Schematinis Pingpong projekto vaizdas sudarytas su programa Precision.

4.2.2 Pong žaidimo realizacija Xilinx FPGA

Kito autoriaus Ping Pong žaidimo realizacija yra pritaikyta FPGA matricai Spartan3E, firmos Xilinx. Tyrimo metu pastebėta, kad daugelis atviro kodo vaizdo generavimo modulių realizacijų yra sukurta būtent Spartan3E matricai. Pagrindinis modulis perduodantis vaizdą į kompiuterio vaizduoklį yra aprašytas vga_sync.vhdl faile, kaip pastebėta daugelis kūrėjų pasinaudoja būtent šiuo aprašu. Jis yra sukurtas kaip mokomoji medžiaga kuriant realizacija su Xilinx Spartan P. Chu knygoje. Joje yra aprašymas su paaiškinimais. Taigi autoriui teko tik pritaikyti savo kuriamai realizacijai.

VGA sinchronizavimas

Pagrindiniai perdavimo komponentai yra 25MHz sinchro signalas, vertikalios bei horizontalios sinchronizacijos taškų nustatymo komponentai, bei konstantos, kurios nustato vėlinimus, taškų skaičių ekrane. Vertikalus nustatymas gali vykti iki 523 taškų, bei horizontalus iki 799 (kai ekrano taškų skaičius yra 640x800) ir priklauso nuo sinchro signalo. Taškų koordinatės veikimo metu yra nustatomos prieš tai nurodytomis koordinatėmis (konstantomis ir jų galimoms variacijoms). Labiau galima iliustruoti VHDL pseudo kodu (18 pav.):

```
sync_pulse = 1 when
    counter >= (display_area +
                front_porch)
and
    counter < (display_area +
               front_porch +
               sync_pulse)
```

18 pav. Pseudo kodas iliustruojantis taškų nustatymą

Horizontalus ir vertikalus vga signalo sinchronizavimas yra reguliuojamas skirtingais impulsais. Matome, kad impulsas priklauso nuo ekrano dydžio, bei nurodytų taškų skaičiaus. Taip pat vga_sync.vhd faile yra nustatomi x ir y koordinatės kaip 10 bitų vektoriai. Taigi jeigu teisingai yra priskiriami kintamieji taškų skaičius yra 640x480, o dažnis 60Hz, kurį palaiko daugelis kompiuterinių ekranų.

Vaizdo perdavimas į ekraną.

Vaizdas į ekraną, kaip anksčiau buvo minėta, yra perduodamas video signalo perdavimo komponentu (vga_sync.vhdl), kuriuo yra nustatomos taškų koordinatės ir perduodamas vaizdas į RGB valdiklį.

Konkretus realizacijos vaizdas (kamuoliuko savybės, rakečių savybės) yra saugomas matricos vidinėje atmintyje (RAM). Taigi vaizdo generavimui yra supaprastinamas darbas, nes nereikia papildomai kurti objektų, tai gali užimti daugiau laiko ir reikalauti daugiau FPGA matricos resursų. Taigi vaizdas yra paimamas iš atminties, suskirstomas į taškus, kurių savybės ir koordinatės yra suskirstomos pagal konkrečią seką. Šioje realizacijoje yra naudojamas tik monochromatinis vaizdas (juoda ir balta spalvos), todėl galioja tik viena savybė - taško koordinatė. Kaip matome 19 paveikslėlyje, pseudo kodas suskirsto objekto atvaizdą į matematinę matricą (pavyzdyje matricos tipas 2x4), kur reikšmė atitinka “0”, matoma balta spalva, kur “1” generatorius parenka juodą spalvą.

```
type rom_type is array(0 to 1) of
    std_logic_vector(0 to 3);
constant SQUARES: rom_type :=
(
    "1010",
    "0101"
);
```

19 pav. Objektų vaizdo generavimas

Nuskaičius objekto vaizdo savybes reikia nurodyti kaip šios koordinatės turi kisti ekrane 20 paveikslėlyje matome pseudo kodą, kuris yra kaip pavyzdys, kaip turi kisti vaizdas keičiant objekto taškų koordinates.

```
row_addr <= px_y(3 downto 0);
col_addr <= px_x(3 downto 0);

pixel <= pixel_data(col_addr);

img_data <= "000" when pixel = '1'
else "111";
```

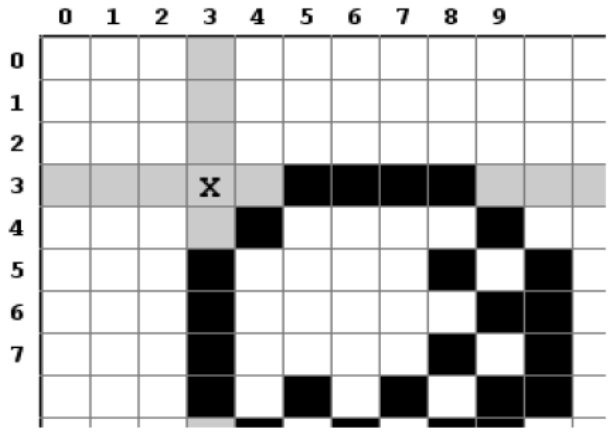
20 pav. Koordinačių kitimas

Sugeneruotas vaizdas yra perduodamas į ROM atmintį, kuri automatiškai priskiria taškų vektorius ir kokios turi būti paimto taško savybės (pixel_data(col_addr)).

Taigi sudarant sudėtingesnius vaizdus (spalvotus) reikalaujama daugiau atminties, tarkim norint perduoti spalvotą vaizdą, turi būti naudojamos ir kitos spalvos, kurios yra sudaromos iš RGB paletės.

Objekto kitimas ekrane

Pong realizacijoje judantis objektas yra kamuoliukas ir raketės. Norint, kad būtų matomas judantis vaizdas ekrane reikia nustatyti x ir y koordinačių vektorius. Taigi jei žinome objekto taškų matricos dydį, reikia tik nustatyti pradinę koordinatę ir pakeisti viso objekto koordinates. 21 pav. matome, kad paimta kamuoliuko pradine koordinatė yra 3,3.



21 pav. vektorių nustatymas

Po to koordinatės yra keičiamos į 0,0. Pagal šią schemą keičiant koordinates matomas judantis vaizdas. Realizuojant VHDL kodu (PONG.vhdl faile) šią operaciją, ji atrodytu taip (22 pav.):

```
row_addr <= px_y(3 downto 0) -
            ball_top_y(3 downto 0);

col_addr <= px_x(3 downto 0) -
            ball_top_x(3 downto 0);
```

22 pav. kamuoliuko koordinačių išrinkimas

Šiuo pseudo kodu yra realizuojamas pradinių koordinačių išrinkimas. Išrinkus pradines koordinates reikia realizuoti objekto judėjimą, kuris programiškai atliekamas taip (23 pav.):

```
rgb <= img_data when
  (px_x >= ball_top_x and
   px_x < ball_top_x + BALL_SIZE and
   px_y >= ball_top_y and
   px_y < ball_top_y + BALL_SIZE) else
       "111";
```

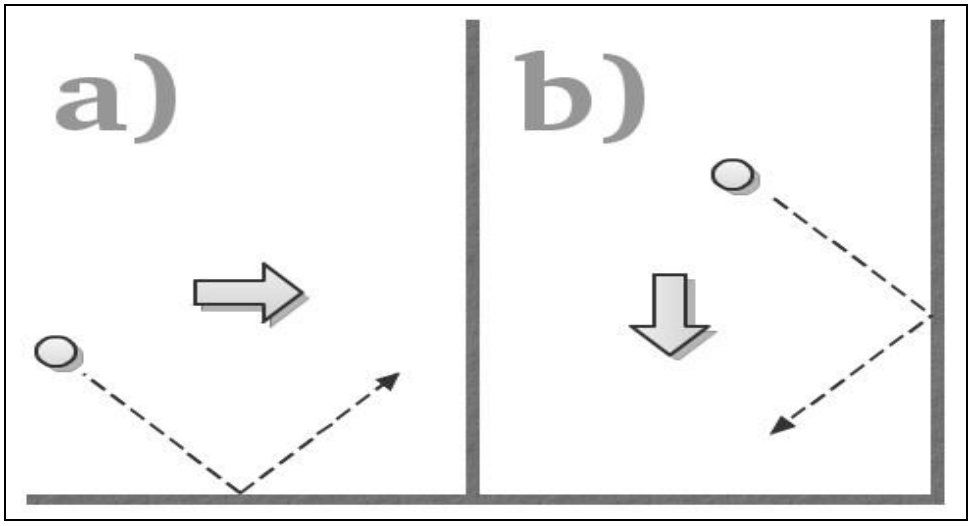
23 pav. Koordinačių atnaujinimas

Taigi kaip matome į RGB yra išvedamos atnaujintos kamuoliuko judėjimo koordinatės.

Kamuoliuko judėjimas

Sudėtingiausias objektas šioje realizacijoje yra kamuoliuko judėjimas. Norint, kad vartotojas galėtų naudotis šia realizacija, būtina suderinti kamuoliuko bei rakečių judėjimo greitį. Šis greitis yra aprašomas kaip kamuoliuko taškų koordinačių atnaujinimas. Tačiau atkuriant koordinates reikia atsižvelgti į VGA vaizdo išdavimo greitį.

Kamuoliuko koordinatėms nustatyti ir naudojamas skaitliukas(counter). Skaitliukas skaičiuoja iki tol kol kamuoliukas atsitrenkia į raketę arba pataiko į priešingą sieną, tada priešininkui yra priskaičiuojami taškai. Taip pat kuriant kamuoliuką reikia atsižvelgti ir į jo sekančią trajektoriją atsimušus į raketę arba i horizontalias sienas(24 pav.).



24 pav. Kamuoliuko judėjimo trajektorija

Kaip iš paveikslėlio matome trajektorija gali kisti kamuoliukui atsimušus vertikaliai, arba horizontaliai. Taigi patikrinus ar kamuoliuko riba yra vertikalioje padėtyje, jo trajektorija aprašoma taip(25 pav.):

```
if ball_y = 0 then
    ball_v_dir_next <= '1';
elsif ball_y = SCREEN_HEIGHT -
                BALL_SIZE then
    ball_v_dir_next <= '0';
end if;
```

25 pav. kamuoliuko trajektorijos tikrinimas

Patikrinti horizontalią kliūti galima atlikti analogiškai taip pat, tik pakeitus kintamuosius į „x“ ašį. Po to būtina nustatyti kamuoliuko tolimesnes judėjimo sekas (26 pav.):

```
if ball_h_dir = '1' then
    ball_x_next <= ball_x + 1;
else
    ball_x_next <= ball_x - 1;
end if;

if ball_v_dir = '1' then
    ball_y_next <= ball_y + 1;
else
    ball_y_next <= ball_y - 1;
end if;
```

26 pav. kamuoliuko judėjimo sekos

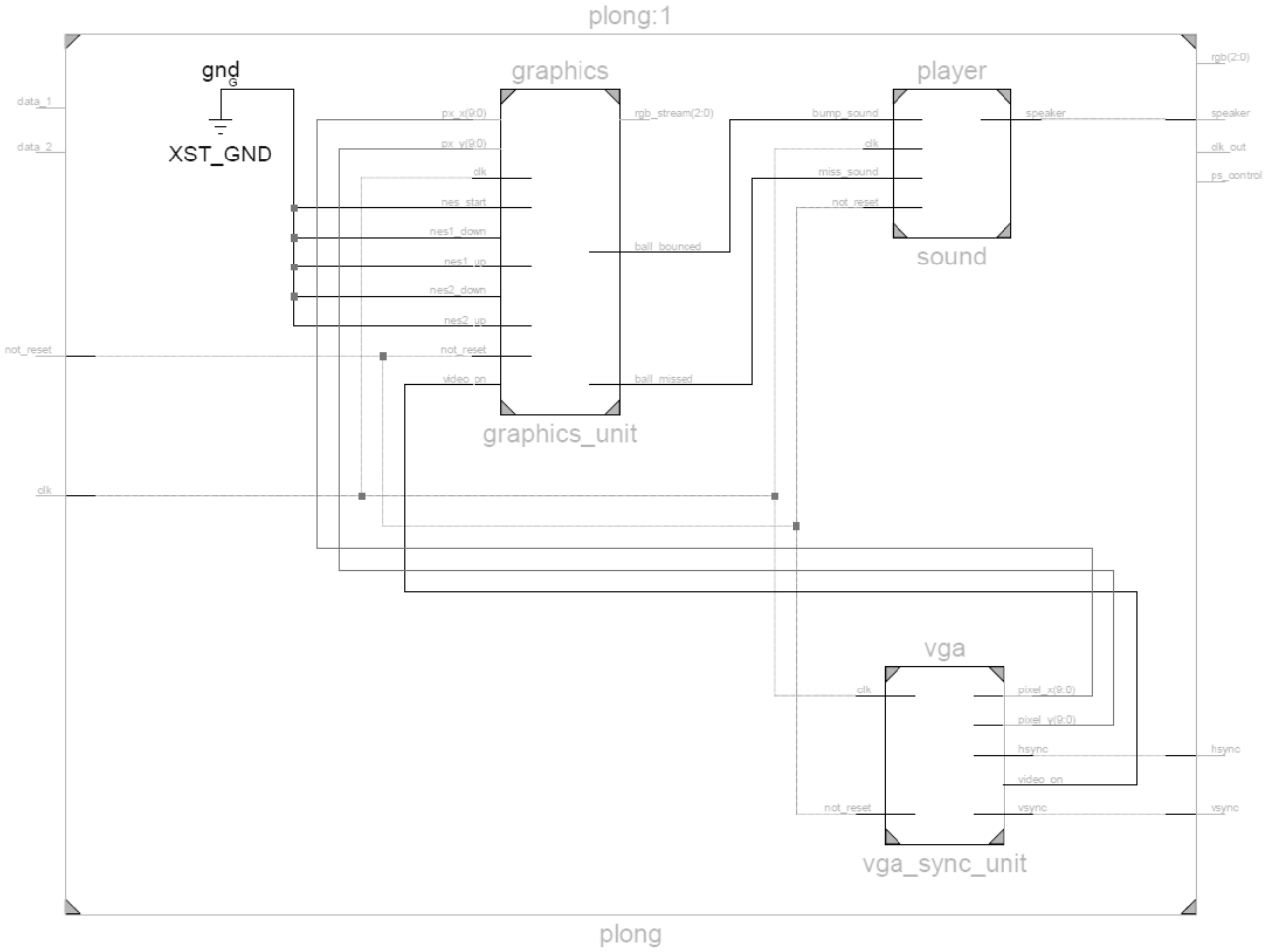
Papildomi išoriniai įrenginiai.

Šioje realizacijoje taip pat yra naudojami išoriniai valdymo įrenginiai (controller.vhd), kurie kaip ir anksčiau minėtoje realizacijoje yra pritaikyti FPGA matricai. Kaip ir kiekvienam prietaisui prijungiant prie kitų, reikalingas sinchronizuotas signalas, nebent jis atitinka esamą įrenginį.

Pong realizacijos išvados bei ataskaitos

Tiriant Pong realizaciją, pritaikytą Spartan3E matricai pastebėta, kad sinchronizacija yra pritaikyta iš anksčiau minėto pavyzdžio. Susipažinta su autoriaus kuriamais sunkumais, bei jų

sprendimo būdais. Taip pat buvo atsižvelgta į FPGA plokštės resursus, kadangi įvykdžius sintezavimą programiniu paketu Xilinx Design Suite viskas įvyko be klaidų, bei buvo gautas schematinis realizacijos vaizdas(27 pav.), bei elementų, energijos sunaudojimo ir vėlinimo ataskaitos.



27 pav. Pong Spartan-3E realizacijos sintezuotas schematinis vaizdas

Sintezavimo komponentų ataskaita

Macro Statistics
Registers : 140
Flip-Flops : 140

==========
* Final Report *
==========
Final Results
RTL Top Level Output File Name : pong.ngr
Top Level Output File Name : pong
Output Format : NGC
Optimization Goal : Speed

Keep Hierarchy : No

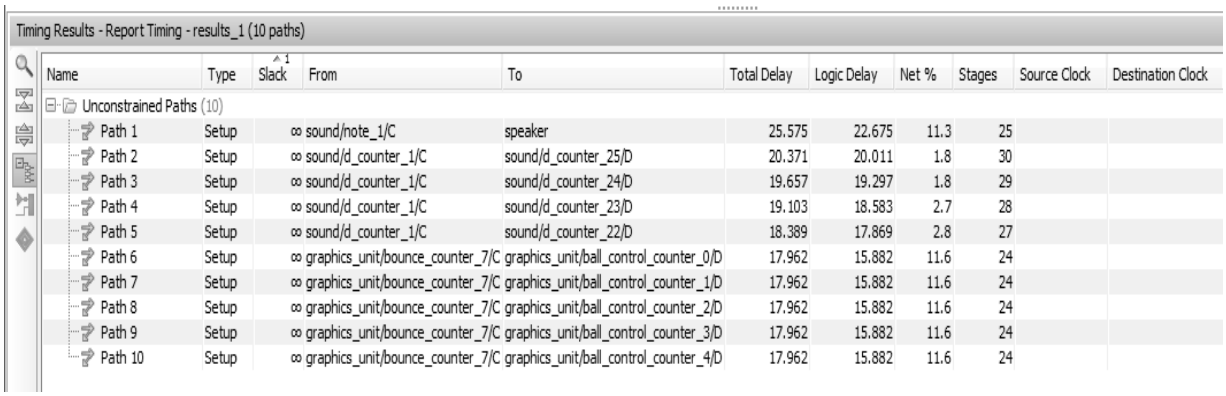
Design Statistics
IOs : 12

Cell Usage :
BELS : 876
GND : 1
INV : 41
LUT1 : 90
LUT2 : 135
LUT3 : 115
LUT3_D : 2
LUT4 : 119
LUT4_D : 3
LUT4_L : 8
MUXCY : 242
MUXF5 : 8
VCC : 1
XORCY : 111
FlipFlops/Latches : 140
FDC : 32
FDC_1 : 54
FDCE : 31
FDCE_1 : 23
Clock Buffers : 1
BUFGP : 1
IO Buffers : 4
IBUF : 1
OBUF : 3

Laikinės ataskaitos

===
Timing constraint: Default period analysis for Clock 'clk'
Clock period: 15.021ns (frequency: 66.572MHz)
Total number of paths / destination ports: 13687 / 194
Total memory usage is 205352 kilobytes

Netlist designer ataskaita (28 pav.)



Timing Results - Report Timing - results_1 (10 paths)

Name	Type	Slack	From	To	Total Delay	Logic Delay	Net %	Stages	Source Clock	Destination Clock
Unconstrained Paths (10)										
Path 1	Setup	∞	sound/note_1/C	speaker	25.575	22.675	11.3	25		
Path 2	Setup	∞	sound/d_counter_1/C	sound/d_counter_25/D	20.371	20.011	1.8	30		
Path 3	Setup	∞	sound/d_counter_1/C	sound/d_counter_24/D	19.657	19.297	1.8	29		
Path 4	Setup	∞	sound/d_counter_1/C	sound/d_counter_23/D	19.103	18.583	2.7	28		
Path 5	Setup	∞	sound/d_counter_1/C	sound/d_counter_22/D	18.389	17.869	2.8	27		
Path 6	Setup	∞	graphics_unit/bounce_counter_7/C	graphics_unit/ball_control_counter_0/D	17.962	15.882	11.6	24		
Path 7	Setup	∞	graphics_unit/bounce_counter_7/C	graphics_unit/ball_control_counter_1/D	17.962	15.882	11.6	24		
Path 8	Setup	∞	graphics_unit/bounce_counter_7/C	graphics_unit/ball_control_counter_2/D	17.962	15.882	11.6	24		
Path 9	Setup	∞	graphics_unit/bounce_counter_7/C	graphics_unit/ball_control_counter_3/D	17.962	15.882	11.6	24		
Path 10	Setup	∞	graphics_unit/bounce_counter_7/C	graphics_unit/ball_control_counter_4/D	17.962	15.882	11.6	24		

28 pav. Pong game Spartan-3E Netlist designer vėlinimo ataskaita

Energijos bei temperatūros ataskaitos

Šios ataskaitos yra atliekamos „Xpower analyzer“ paketu.

Galimų temperatūrų ataskaita

Thermal Summary	
Effective TJA (C/W)	49.0
Max Ambient (C)	83.5
Junction Temp (C)	26.5

Sunaudojamos energijos ataskaita

Power Supply Summary			
	Total	Dynamic	Quiescent
Supply Power (mW)	29.82	0.00	29.82

4.2.3 Galaxy realizacijos tyrimas

Galaxy realizacija – kosmoso šaudyklė, kurioje judantys objektai yra „ateiviai“ ir vartotojas turi savo erdvėlaiviu sunaikinti visus. Ši realizacija taip pat yra pritaikyti Xilinx firmos FPGA matricai, Spartan 3E.

Kadangi šioje realizacijoje yra naudojamos ne vien juoda bei balta spalvos, tai reikia skirti didesnį dėmesį į spalvotų objektų sudarymą ir perdavimą i ekraną. Trijų bitų taškui jau nepakanka, taigi reikia naudoti daugiau atminties įrenginyje. Realizuojant spalvotus vaizdus reikia naudoti taškų masyvus, išsikviesti RGB tipo masyvą, kuris programiškai atrodytų taip(29 pav.):

```
type rgb_array is array(0 to 3) of
    std_logic_vector(2 downto 0);
```

29 pav. pseudo kodas iškviečiantis RGB masyvą

Šis tipas yra naudojamas aprašyti vaizdo eilutėms sudaryti. Kiekvienoje eilutėje yra 4 taškai, ir kiekvienas taškas yra aprašomas kaip 3 bitų vektorius.

Sudarius eilutes jas reikia apjungti į vientisą vaizdą. Eilučių masyvas yra patalpinamas į atmintį ir taip sudaro vaizdą(30 pav.).

```
type rom_type is array(0 to 1) of
    rgb_array;
```

30 pav. apjungiamas masyvas

Vėliau yra sudaromos konstantos su taškų informacija(31 pav.):

```
constant SQUARES_2: rom_type :=
(
    ("000", "001", "010", "011"),
    ("100", "101", "110", "111")
);
```

31 pav. konstantos su taškų informacija

Taip yra sudaromi spalvoti vaizdai, prieš tai pavaizduotame pseudo kode yra atvaizduojamos aštuonios spalvos panaudojant tris bitus.

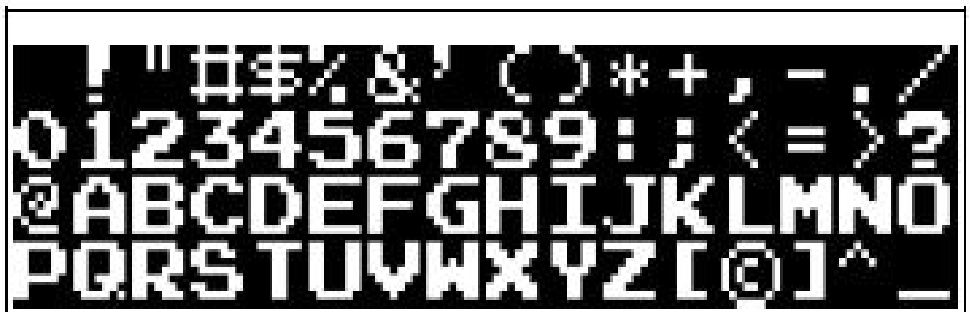
Taigi visa spalvotų vaizdų informacija yra saugoma atmintyje, masyvo pavidalu. Vėliau vaizdas yra išvedamas į ekraną atskirais taškais, kurie turi savo koordinates bei spalvą. Viskas yra valdoma vidinės atminties iš jos išskiriant informaciją.

Kadangi šioje realizacijoje objektų kitimas yra pastovus, tai yra šiek tiek paprasčiau vystyti vaizdo perdavimą į ekraną. Lėktuvėlio valdymas vyksta tokiu pačiu principu kaip ir Pong realizacijoje naudojamose raketėse. Taigi sunkiausiai įgyvendinti yra ateivių sunaikinimo vaizdą, kuris išnyksta palaipsniui, taigi automatui reikia nurodyti daugiau būsenų atnaujinant taškus.

Simbolių atvaizdavimas ekrane

Norint, kad realizacija būtų labiau pritaikyta vartotojui, reikia išvesti į ekraną informacinius pranešimus, tokius kaip surinktų taškų skaičius, kad žaidimas yra baigtas ir kitą.

Norint išvesti pranešimus į ekraną būtina sudaryti simbolius, raides, skaičius ir t.t. (32 pav.)



32 pav. Simboliai informaciniams pranešimams.

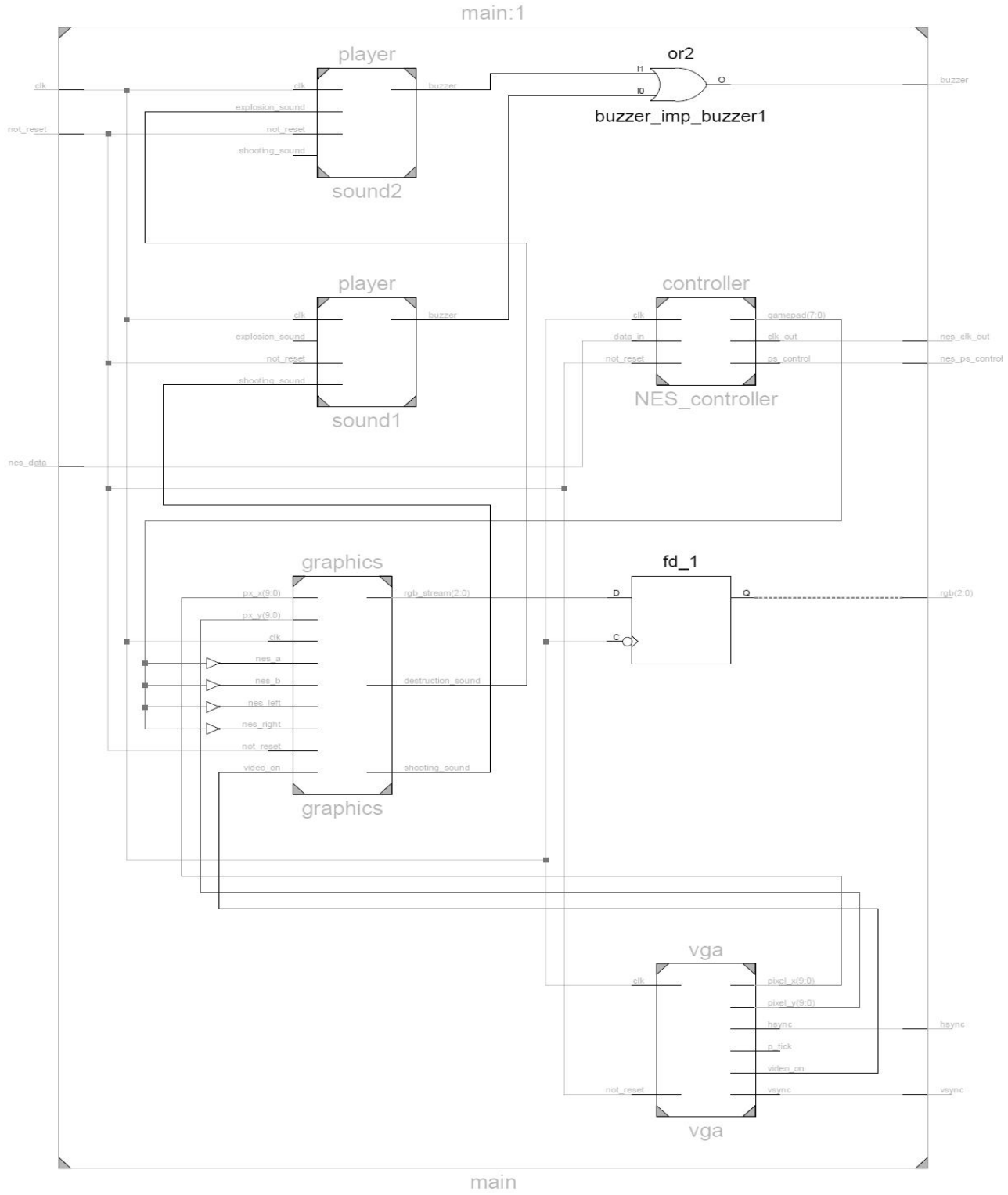
Pasirinkus norimą simbolių šriftą, simboliai yra užkoduojami ir išsaugojami atmintyje. Simbolių kodavimas vyksta taip pat kaip ir vaizdo, simboliai yra sudaromi iš loginio „1“ ir „0“. Kaip ir anksčiau minėtame projekte „Ping Pong game“, kur taško reikšmė yra 0 – balta spalva, kur 1 juoda(33 pav.):

```
"00111000", --   ###
"01101100", --  ## ##
"11000110", -- ##   ##
"11000110", -- ##   ##
"11111110", -- #######
"11000110", -- ##   ##
"11000110", -- ##   ##
"00000000", --
```

33 pav. Užkoduoto simbolio pavyzdys

Kiekvienas simbolis yra sudarytas iš 8x8 matricos, kaip ir prieš tai buvo minėta, paimamas pirmas matricos taškas ir nurodomos jo koordinatės ekrane.

Taip pat buvo atsižvelgta į FPGA plokštės resursus, kadangi įvykdžius sintezavimą programiniu paketu Xilinx Design Suite viskas įvyko be klaidų, bei buvo gautas schematinis realizacijos vaizdas(34 pav.), bei elementų, energijos sunaudojimo ir vėlinimo ataskaitos.



34 pav. Sintezuotas Galaxy realizacijos schematinis vaizdas.

Sintezavimo komponentų ataskaita

Final Register Report

Macro Statistics
Registers : 542
Flip-Flops : 542

```
=========================================================================
*                    Design Summary                     *
=========================================================================
Top Level Output File Name         : main.ngc

Primitive and Black Box Usage:
------------------------------
#      BELS                 : 2793
#      GND                  : 1
#      INV                  : 37
#      LUT1                 : 46
#      LUT2                 : 206
#      LUT3                 : 212
#      LUT4                 : 430
#      LUT5                 : 217
#      LUT6                 : 653
#      MUXCY                : 532
#      MUXF7                : 103
#      MUXF8                : 23
#      VCC                  : 1
#      XORCY                : 332
# FlipFlops/Latches         : 581
#      FD_1                 : 3
#      FDC                  : 213
#      FDC_1                : 32
#      FDCE                 : 106
#      FDCE_1               : 147
#      FDE_1                : 1
#      FDP_1                : 5
#      FDPE                 : 10
#      FDPE_1               : 25
#      LD                   : 39
# Clock Buffers             : 2
#      BUFG                 : 1
#      BUFGP                : 1
# IO Buffers                : 10
#      IBUF                 : 2
#      OBUF                 : 8
```

Laikinės ataskaitos

Timing constraint: Default period analysis for Clock 'clk'
Clock period: 15.961ns (frequency: 62.653MHz)
Total number of paths / destination ports: 515547 / 831
Total memory usage is 233336 kilobytes

Netlist designer ataskaita (35 pav.)

Name	Type	Slack	From	To	Total Delay	Logic Delay	Net %	Stages	Source Clock	Destination Clock
Unconstrained Paths (10)										
Path 1	Setup	∞	vga/v_count_8/C	rgb_reg_0/D	11.189	2.964	73.5	16		
Path 2	Setup	∞	vga/v_count_8/C	rgb_reg_2/D	10.952	2.969	72.9	16		
Path 3	Setup	∞	vga/v_count_8/C	rgb_reg_1/D	10.680	2.964	72.2	16		
Path 4	Setup	∞	graphics/missile/y_coordinate_0/C	graphics/missile/missile_ready/D	9.191	2.257	75.4	14		
Path 5	Setup	∞	graphics/missile/y_coordinate_0/C	graphics/origin_y_5/CE	9.175	2.003	78.2	13		
Path 6	Setup	∞	graphics/missile/y_coordinate_0/C	graphics/origin_y_7/CE	9.175	2.003	78.2	13		
Path 7	Setup	∞	graphics/missile/y_coordinate_0/C	graphics/origin_y_4/CE	9.146	2.003	78.1	13		
Path 8	Setup	∞	graphics/missile/y_coordinate_0/C	graphics/origin_y_6/CE	9.146	2.003	78.1	13		
Path 9	Setup	∞	graphics/missile/y_coordinate_0/C	graphics/origin_y_8/CE	9.146	2.003	78.1	13		
Path 10	Setup	∞	graphics/missile/y_coordinate_0/C	graphics/origin_y_9/CE	9.146	2.003	78.1	13		

35 pav. Galaxy Netlist designer vėlinimo ataskaita

Energijos bei temperatūros ataskaitos

Šios ataskaitos yra atliekamos „Xpower analyzer“ paketu.

Galimų temperatūrų ataskaita

Thermal Summary	
Effective TJA (C/W)	42.4
Max Ambient (C)	84.5
Junction Temp (C)	25.5

Sunaudojamos energijos ataskaita

Power Supply Summary			
	Total	Dynamic	Quiescent
Supply Power (mW)	11.23	0.06	11.17

5. Vaizdo generavimo moduliu realizacijų pritaikytų FPGA matricoms tobulinimas

Tobulinant bet kokius projektus ar net atskirus objektus reikia atsižvelgti į realizacijos apimtis. Ne visi projektai gali būti daug tobulinami, ne visiems galima pridėti papildomų elementų ir panašiai. FPGA plokštės turi ribotą atmintį, komponentų galinčių apdirbti duomenis. Be viso to, būtina atsižvelgti ir prijungtus įrenginius, turi būti suderinami dažniai, įėjimų reikšmės. Reikia atsižvelgti į ekrano parametrus, kad spėtų atkurti grafinius objektus, kad turėtų reikiamą skaičių spalvų ir panašiai.

Po atlikto tyrimo matyti, kad tobulinant realizacijas galima pritaikyti ir kitos firmos programuojamoms matricoms, kadangi TV Pong projektas yra pritaikytas tik Actel ProASIC, pakeitus bibliotekas galima naudotis ir kitomis matricomis bei programiniais paketais, tuo pačiu galima priskirti daugiau spalvų, jei leidžia galimybės FPGA matricos, kuriai yra perdaromas projektas. Taip pat galima pridėti įvairių variacijų, pridėti garsus atsimušinėjant kamuoliukui, pridėti įvairių lygių ar kitų žaidimą pagyvinančių elementų.

Pong realizacija pritaikyta Spartan-3E FPGA matricai yra labiau pažengusi nei prieš tai minėta. Šiame projekte gali būti naudojamos spalvos, taip pat naudojami garsai. Tobulinant šią realizaciją, siūlyčiau keisti žaidimo lygius, greitinant kamuoliuko judėjimą, tačiau reikia atsižvelgti į tai, kad greitėjant kamuoliukui (greičiau atnaujinant kamuoliuko taškų koordinates) galima susidurti su ekrano sinchronizacijos problemomis, tarkim, jei kamuoliuko taškai kistų kas vieną, kamuoliukas gali išlėkti už matomos ekrano ribos. Tokiu atvejų būtina atsižvelgti ir į kamuoliuko judėjimo ribas. Paprastesnis būdas yra atnaujinti visų taškų koordinates, tačiau reikia atsižvelgti ir į ekrano sugebėjimą atkurti vaizdą. Taip pat šioje realizacijoje galima pridėti tam tikrus pasunkinimus, tarkim, keičiantis lygiui mažinti raketės dydį, tai galima padaryti mažinant grafikos atvaizdavimo matricą. Be to galima pridėti papildomas sienas, kad būtų sunkiau pasiekiamas priešininkas. Tai realizuojant reikia įvesti papildomus grafinius elementus ir nustatyti konkrečią jų vietą ekrane, tačiau kamuoliuko judėjimo trajektorijos funkcijoje reikia pakeisti elgseną, kadangi atsiradus papildomai sienai kamuoliuko atsimušimams atsiranda daugiau kliūčių. Kaip jau buvo minėta, šiai realizacijai galima pritaikyti ir kitas spalvas, galima uždėti ekrano vaizdą, pakeisti vietoj balto fono, tačiau viskas priklauso nuo FPGA matricos galimybių.

Galaxy realizacijoje jau yra panaudotos spalvos, taigi šioje realizacijoje galima tik surasti klaidas ir pridėti skirtingus sunkumo lygius. Taip pat galima padaryti, kad ne tik erdvėlaivis numušinėja ateivius, bet ir ateiviai gali numušti erdvėlaivį.

Apžvelgus tobulinimus, pagrindas šiame tiriamajame darbe gali būti pritaikymas įvairesniam kiekiui FPGA plokščių. Kadangi šios realizacijos yra pagamintos naudojant ir kitomis programavimo kalbomis, jos yra ištobulintos, pridėti sunkesni lygiai, pataisytos klaidos. Taigi tobulinant realizacijas pritaikytas FPGA matricoms, galima atsižvelgti ir į kitomis programavimo kalbomis parašytomis realizacijomis.

6.Išvados

Apžvalgos metu buvo susipažinta su esamais atviro kodo vaizdo generavimo modulių realizacijų projektais, naudojama technine bei programine įranga. Susipažinta su VGA adapterio veikimo principu, įėjimų bei išėjimų signalais. Atlikta FPGA plokščių analizė, pagal kurią buvo nuspręsta, kurį programuojamos matricos tipą pasirinkti. Taip pat rasta medžiagos bei projektų, kuriais buvo remtasi atliekant tiriamąjį darbą.

Tiriant realizacijas, susipažinta su programiniais vartotojo paketais tokiais kaip:

- Mentor Graphics „FPGA Advantage“, kurį sudaro blokinių diagramų kūrimo įrankis, keletas sintezatorių, modeliavimo įrankiai, dalinai sudaromos projekto ataskaitos.
- Xilinx „ISE Design Suite“, kurį sudaro sintezavimo, modeliavimo, takelių sudarymo, naudojamos energijos analizavimo bei kitais įrankiais, taip pat sudaromos viso projekto ataskaitos.

Pastebėta, kad paketo pasirinkimas lemia darbo sėkmę, kadangi kiekvienas programinis paketas dažniausiai būna pritaikytas konkrečiai FPGA matricai. Tiriant realizacijas buvo bandoma dirbti anksčiau minėtais paketais, tačiau klaidingai pasirinkus trūkdavo technologinių bibliotekų, ar atsirasdavo klaidų, kurių kitas paketas nerodo. Tačiau tai nėra vienintele bėda, kiekvienas kūrėjas turi savo programavimo stilių, bei savaip priskiria kintamuosius, kuriuos yra sunkiau suprasti kitam realizaciją tiriančiam asmeniui. Kadangi atviras kodas yra platinamas laisvai ir gali būti daug skirtingų realizacijų variacijų, sunku jas suderinti vieną su kita.

Tyrimo metu buvo susipažinta su vaizdo objektų kūrimo bei realizavimo ekrane būdais, susipažinta su kokiais sunkumais susiduria tokių realizacijų kūrėjai, bei jų sprendimo būdais.

Atliekant atvirko kodo vaizdo generavimo modulių realizacijų tobulinimus būtina atsižvelgti į FPGA matricos technines charakteristikas, kad tobulinimas atitiktu turimą sistemą. Jeigu bus kuriama rimtesnė realizacija, remiantis esamomis, galima pritaikyti sistemų kūrimo modelius, kad realizacija būtų labiau dokumentuota, ištestuota ir palaikoma. Kadangi matricų kaina atspindi technines galimybes, atitinkančias šių dienų reikalavimus, pasirinkus atitinkamą variantą galima realizacijų perduodama vaizdą ištobulinti iki sudėtingų objektų detalių, spalvų derinio, bei jų atkūrimo greičio ekrane. Taip pat reikia atsižvelgti į tobulinamos realizacijos naudą, kuri gali būti ateityje kuriama ne tik atviru kodu, bet ją pakankamai ištobulinus pasiūlyti savo darbą vaizdo generavimo modulių kūrėjams.

Atviro kodo realizacijos gali būti naudingos mokomiesiems tikslams ir mažiau naudojamos praktikoje, kadangi projektai būna pritaikyti labiau patyrusiems naudotojams, o ne vartotojų visumai.

Literatūros sąrašas:

1. V. Abraitis, E. Bareiša. Programuojamųjų lustų testavimo metodai. Elektronika ir elektrotechnika. Programinės įrangos katedra, Kauno technologijos universitetas. 2003. Nr. 5(47). 43 P.
2. http://www.ht-lab.com/freecores/pingpong/pingpong.html TV Pong Game
3. http://www.fpga4fun.com/PongGame.html Pong Game
4. http://www.fpgaarcade.com/spc_main.htm Space Invaders
5. http://web.mit.edu/6.111/www/f2005/projects/jrv23_Proje ct_Final_Report.pdf Asteroids
6. http://instruct1.cit.cornell.edu/courses/ece576/FinalProjects/f2009/Adam_Tom/graphics_engine_12_9_09/index.html 2D graphics engine
7. http://instruct1.cit.cornell.edu/courses/ece576/FinalProjects/f2009/ng292/ng292/index.html 3D render engine
8. http://instruct1.cit.cornell.edu/courses/ece576/FinalProjects/f2007/jas328_asl45/jas328_asl45/index.htm SimpleGPU
9. http://members.optusnet.com.au/jekent/FPGA.htm informacija apie FPGA plokštes bei jų panaudojimo galimybes
10. http://instruct1.cit.cornell.edu/courses/ece576/FinalProjects/ Darbai su aprašymais skirti FPGA matricoms.
11. http://www.maxim-ic.com/app-notes/index.mvp/id/3605 Informacija ir istorija apie vaizdo plokštes.
12. http://www.dmoz.org/Science/Technology/Electronics/Design/Hardware_Description_Languages/ Informacija apie HDL kalbą.
13. http://www.theinquirer.net/inquirer/news/1023755/graphics-card-launched-fpga-fiddlers Straipsnis apie išleistą į pardavimą atviro kodo vaizdo plokštę.
14. http://www.design-reuse.com/articles/21108/deinterlacing-hd-video.html Deinterlacing with FPGA for HDTVs
15. www.vabolis.lt/page/4/ VGA signalo generavimas
16. http://www.eecg.utoronto.ca/~jayar/ece241_06F/vga/vga-interface.html VGA Adapter Interface
17. http://university.eve-team.com/files/ASAP2009.pdf FPGA –based RTL Emulation for Embedded Software Development
18. http://www.tomshardware.com/reviews/tft-connection,931-7.html DVI Dual Link
19. http://newsroom.intel.com/community/intel_newsroom/blog/2010/12/08/leading-pc-companies-move-to-all-digital-display-technology-phasing-out-analog Digital Visual Interface

20. https://github.com/akshayp/vhdl/tree/master/pong Examples of VHDL code for processor and PingPong game.
21. http://code.google.com/p/uzebox/ [Aplankyta 2010 kovo mėn.]
22. http://www.mentor.com/products/fpga/synthesis/precision_rtl/ Precision rtl
23. http://www.mentor.com/products/fpga/hdl_design/hdl_designer_series/ RTL design analysis
24. http://www.gmvhdl.com/ VHDL Tutorial
25. http://martin.hinner.info/vga/timing.html [Aplankyta 2011 balandžio mėn]
26. Xilinx, Inc., 2008. Xilinx Spartan-3E FPGA Family Datasheet. http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf [Aplankyta 2011 balandžio mėn.]
27. Ashenden, Peter J., 1996. The Designers Guide to VHDL. San Francisco: Morgan Kaufmann
28. Xilinx, Inc., 2008. Xilinx Spartan-3E FPGA Starter Kit User Guide. http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf [Aplankyta 2011 banandžio mėn.]
29. Myers, Robert L., 2002. Display interfaces: fundamentals and standards. Chichester: John Wiley and Sons
30. Chu, Pong P., 2008. FPGA prototyping by VHDL examples. New Jersey: John Wiley & Sons VHDL kūrimo pavyzdžiai
31. ON Semiconductor, 2005. MC14021B: 8-Bit Static Shift Register.
32. http://www.onsemi.com/PowerSolutions/product.do?id=MC14021BCP [Aplankyta 2011 gegužės mėn.]
33. http://www.mikekohn.net/micro/fpga_vga.php VGA Graphics On An FPGA
34. http://www.alteraforum.com/forum/showthread.php?t=3009 VGA VHDL DE2 problem
35. http://www.pyroelectro.com/tutorials/vhdl_vga/software.html Creating VGA With VHDL: Software
36. http://www.retromicro.com/projects.html FPGA experiments
37. http://asic.co.in/projects/vga_files/vga_lcd.htm VGA/LCD controller's
38. http://blog.asgon.net/2010/07/vga-test-signal-generator-with-fpga/ VGA Test Signal Generator in VHDL
39. http://www.ece.stevens-tech.edu/~hhe/cpe487f09/lab/VGABALL.htm VGA Ball I
40. http://www.fpgaarcade.com/games.htm Programmable Gaming Hardware
41. http://pinouts.ru/Video/VGA15_pinout.shtml VGA connector pinout
42. http://www.ece.unm.edu/~yuebing/files/fall_2008/space.pdf SPACE INVADER: VHDL

Terminų žodynas

VGA (ang. Video Graphics Array) – vaizdo signalo prievadas

FPGA (ang. Field-programmable Gate Array) – programuojamos logines matricos

SYNC (ang. Synchronization) - kelių vienodų arba atitinkamų procesų suderinimas, kad jie vyktų vienu metu arba jų vyksmo periodai skirtųsi tam tikru laiko intervalu.

HDL (ang. hardware description language) – aparatūrinė programavimo kalba

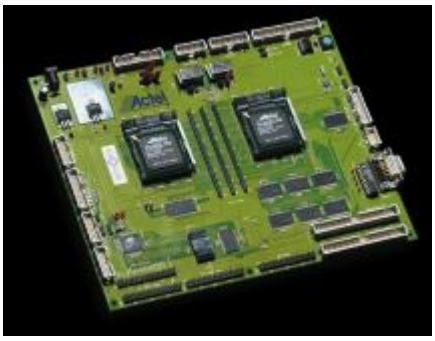
Paveikslėlių sąrašas

Lentelių sąrašas

Priedai

1. FPGA plokštės

ACTEL PROASICPLUS



1 lentelė Actel ProASICPlus palyginimas

	APA075	APA150	APA300	APA450	APA600	APA750	APA1000
System Gates	75,000	150,000	300,000	450,000	600,000	750,000	1,000,000
Max Registers	3,072	6,144	8,192	12,288	21,504	32,786	56,320
Embedded RAM Bits	27 k	36 k	72 k	108 k	126 k	144 k	198 k
Embedded RAM Blocks (256x9)	12	16	32	48	56	64	88
Max User I/Os	158	242	290	344	454	562	712
Packages	TQ100	TQ100					
	TQ144						
	FG144	FG144	FG144	FG144			
	PQ208	PQ208	PQ208	PQ208	PQ208	PQ208	PQ208
		BG456	BG456	BG456	BG456	BG456	BG456
		FG256	FG256	FG256	FG256		
				FG484	FG484		
					FG676	FG676	
						FG896	FG896
							FG1152

Xilinx Spartan-3E

Spartan-3E FPGA matricų šeimos specifikacija

Features
• Very low cost, high-performance logic solution for high-volume, consumer-oriented applications
• Proven advanced 90-nanometer process technology
• Multi-voltage, multi-standard SelectIO™ interface pins
- Up to 376 I/O pins or 156 differential signal pairs
- LVCMOS, LVTTL, HSTL, and SSTL single-ended signal standards
- 3.3V, 2.5V, 1.8V, 1.5V, and 1.2V signaling
- 622+ Mb/s data transfer rate per I/O
- True LVDS, RSDS, mini-LVDS, differential HSTL/SSTL differential I/O
- Enhanced Double Data Rate (DDR) support
- DDR SDRAM support up to 333 Mb/s

• Abundant, flexible logic resources
- Densities up to 33,192 logic cells, including optional shift register or distributed RAM support
- Efficient wide multiplexers, wide logic
- Fast look-ahead carry logic
- Enhanced 18 x 18 multipliers with optional pipeline
- IEEE 1149.1/1532 JTAG programming/debug port
• Hierarchical SelectRAM™ memory architecture
- Up to 648 Kbits of fast block RAM
- Up to 231 Kbits of efficient distributed RAM
• Up to eight Digital Clock Managers (DCMs)
- Clock skew elimination (delay locked loop)
- Frequency synthesis, multiplication, division
- High-resolution phase shifting
- Wide frequency range (5 MHz to over 300 MHz)
• Eight global clocks plus eight additional clocks per each half of device, plus abundant low-skew routing
• Configuration interface to industry-standard PROMs
- Low-cost, space-saving SPI serial Flash PROM
- x8 or x8/x16 parallel NOR Flash PROM
- Low-cost Xilinx® Platform Flash with JTAG
• Complete Xilinx ISE® and WebPACK™ software
• MicroBlaze™ and PicoBlaze™ embedded processor cores
• Fully compliant 32-/64-bit 33 MHz PCI support (66 MHz in some devices)
• Low-cost QFP and BGA packaging options
- Common footprints support easy density migration
- Pb-free packaging options
• XA Automotive version available

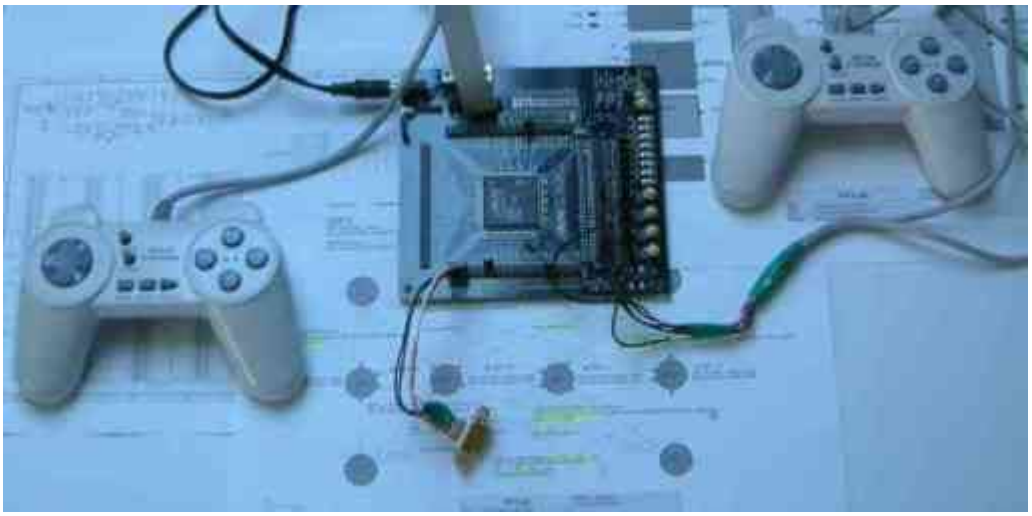
2 lentelė Xilinx FPGA matricų palyginimas

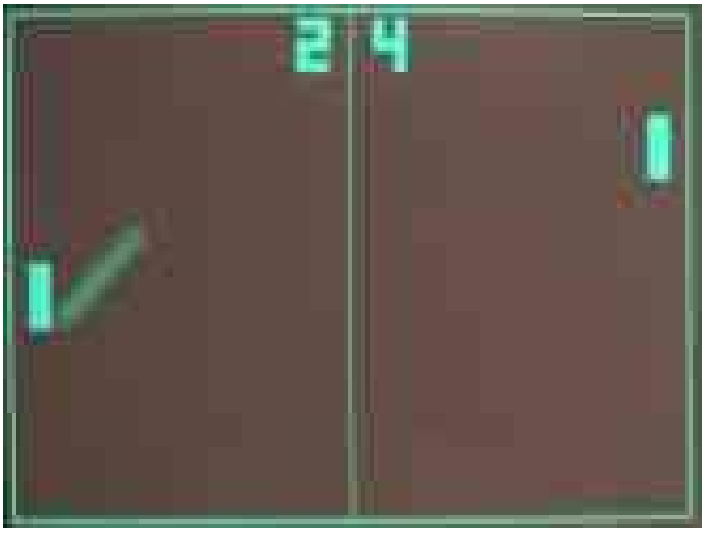
FPGA Comparison Table				
Features	Virtex-6	Virtex-5	Spartan-6	Extended Spartan-3A
Logic Cells	Up to 760,000	Up to 330,000	Up to 150,000	Up to 53,000
User I/Os	Up to 1200	Up to 1200	Up to 570	Up to 519 I/O
I/O Standards Supported	Over 40	Over 40	Over 40	Over 20
Clock Management Technology	PLL	DCM + PLL	DCM + PLL	DCM
Embedded Block RAM	Up to 38 Mbits	Up to 18 Mbits	Up to 4.8 Mbits	Up to 1.8 Mbits
Embedded Multipliers for DSP	Yes (25 x 18 MAC)	Yes (25 x 18 MAC)	Yes (18 x 18 MAC)	Yes (18 x 18 MAC)
Multi-Gigabit High Speed Serial	6.5 Gbps, beyond 11.18 Gbps	3.75 Gbps, 6.5 Gbps	3.125 Gbps	No
PCI Express Technology	Gen 1, x8, hard Gen 2, x8, hard	Gen 1, x8, hard Gen 2, x8, soft	Gen 1, x1, hard	No
Soft Processor Support	Yes	Yes	Yes	Yes

2. Realizacijų nuotraukos

2.1. TV Pong Game ACTEL FPGA

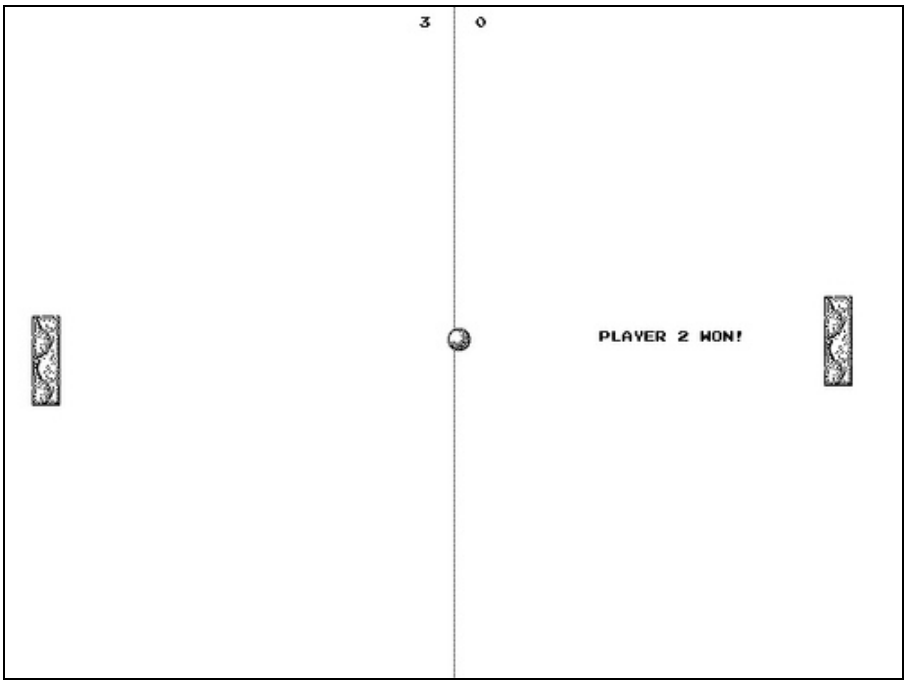
(nuotraukos iš tinklapio http://www.ht-lab.com/freecores/pingpong/pingpong.html)

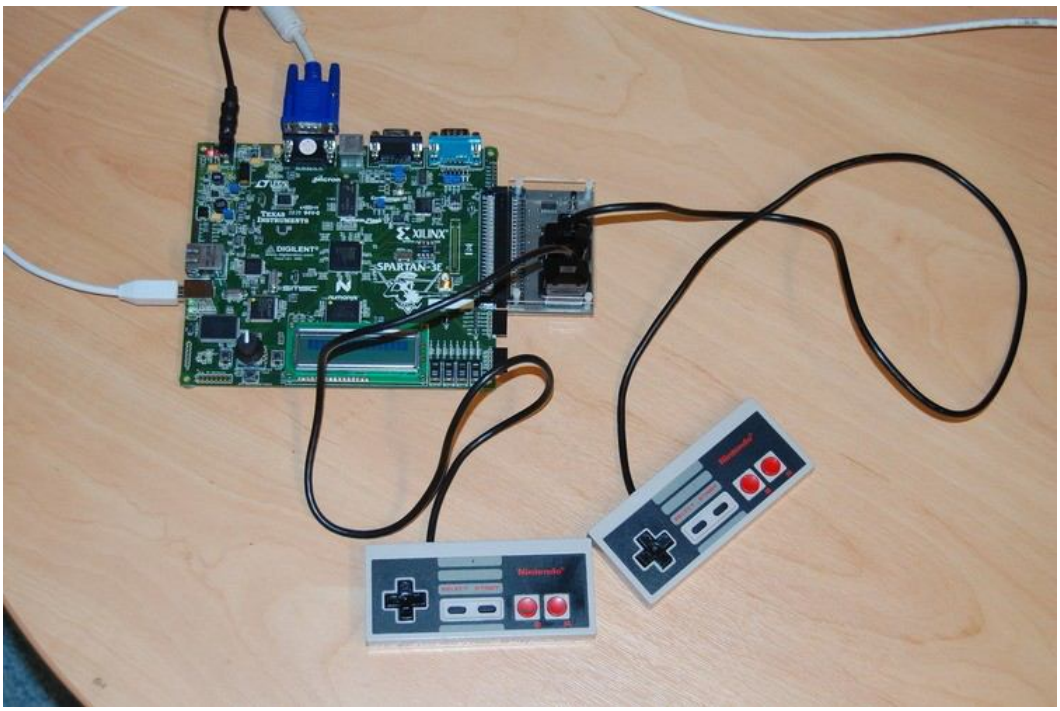




2.2. Ping Pong Spartan-3E FGPA

(Nuotraukos iš http://www.flickr.com/photos/armandas/4072103434/)





2.3. Testinė VGA Sync realizacija

(nuotaukos iš http://blog.asgon.net/2010/07/vga-test-signal-generator-with-fpga/)