ŠIAULIŲ UNIVERSITETAS
TECHNOLOGIJOS IR GAMTOS MOKSLŲ FAKULTETAS
ELEKTRONIKOS KATEDRA

Arūnas Šlenderis

LPLM įrenginio resursų išnaudojimas, įgyvendinant vaizdų apdorojimo algoritmus

Magistro darbas

Vadovas
prof. dr. G. Daunys

ŠIAULIAI, 2014

ŠIAULIŲ UNIVERSITETAS
TECHNOLOGIJOS IR GAMTOS MOKSLŲ FAKULTETAS
ELEKTRONIKOS KATEDRA

TVIRTINU

Katedros vedėjas

prof. dr. G. Daunys

LPLM įrenginio resursų išnaudojimas, įgyvendinant vaizdų apdorojimo algoritmus

Signalų technologijos magistro darbas

Vadovas

prof. dr. G. Daunys

2014-06

Atliko

RM-12 gr. stud. A. Šlenderis

2014-06

Recenzentas

2014-06

ŠIAULIAI, 2014

Šlenderis A. FPGA Resources Utilization Implementing Image Processing Algorithms: Master thesis of signal technology. Research advisor prof. dr. G. Daunys: Šiauliai University, Faculty of Technology and Natural Sciences, Departament of Electronics. – Šiauliai, 2014. 49 p.

Summary

In the first chapter theory of various image and video processing algorithms are reviewed. These algorithms include edge detection, FIR filtering, and image mixing.

Second chapter goes by reviewing other science papers about image processing using FPGA devices. There are reviewed different methods and different systems and result analysis. These systems include: objects recognition in video stream, various image enhancement algorithms, image processing using Verilog programing language.

Third chapter is dedicated for tools, used to do my own research. Tools were both: software and hardware. Software tools are: Quartus II, Altera-modelsim, hardware tools: DE0-nano development board, D5M 5MP digital video camera.

In fourth chapter were reviewed video processing algorithms, which were created for research. These algorithms include: Canny and sobel edge detection, FIR filtering, two video streams mixing.

Fifth chapter is dedicated to results. The main goal of my research was to examine resources, used by FPGA CYCLONE IV device with image processing systems. In this research there were found out FPGA device logic elements usage, power consumption, system stability, effectiveness, and execution time.

Turinys

Paveikslai

Lentelės

Įvadas

Daugumos vaizdų apdorojimo sistemų tikslas yra transformuoti vaizdą į naują, pagerintą vaizdą[1]. Tokios sistemos reikalingos, nes vartotojai nori kažko naujo. Šiuo metu išmanieji įrenginiai yra populiarūs ir vartotojai neįsivaizduoja gyvenimų be jų. Išmanieji telefonai, kameros, akiniai, laikrodžiai, televizoriai ir t. t. Šie įrenginiai atlieka nemažai vaizdų apdorojimo algoritmų, kurie pagerina sąsają žmogus-mašina (angl. Human Maschine Interaction).

Norint užtikrinti interaktyvumą ir komfortą, minėti įrenginiai turi veikti realiu laiku, iš to seka, kad vaizdų apdorojimo algoritmai taip pat turi vykti realiu laiku[2]. Realus vykdymo laikas suprantamas, kai pokyčiai vyksta daugiau nei 24 kartai per sekundę arba trumpiau nei kas 41 ms.

Vienas iš realaus laiko sistemų kūrimo būdų – panaudojant lauku programuojamas logines matricas (LPLM). Jų savybės leidžia greitai, efektyviai ir naudojant mažai energijos apdoroti duomenis. Taip pat viena iš pagrindinių LPLM lustų savybių, kad juos galima pakeisti, atnaujinti, nekeičiant aparatinės dalies.

Darbo tikslas: ištirti „Altera“[3] Cyclone IV[4] LPLM įrenginio naudojamus resursus, įgyvendinant skirtingus vaizdų apdorojimo algoritmus.

Darbo uždaviniai:

1. Sudaryti sistemą, kuri paruoš vaizdą apdorojimui pasirinktu algoritmu ir apdorotą vaizdą išves į ekraną.
2. Sudaryti vaizdų apdorojimo algoritmus.
3. Atlikti vaizdų apdorojimo algoritmų simuliacijas ir tyrimus.
4. Aptarti gautus rezultatus ir padaryti išvadas.

1. Vaizdų apdorojimo algoritmų apžvalga

Šiame skyriuje bus aptariami įvairūs vaizdų apdorojimo algoritmai tokie kraštų aptikimas, RIR filtravimas, vaizdo sudėjimas.

1.1 Kraštų aptikimo algoritmai

Šiame poskyryje bus aptariami kraštų aptikimo algoritmai

1.1.1 Sobel kraštų aptikimo algoritmas

Šis algoritmas susidaro iš poros 3x3 matricų, kuriuos viena nuo kitos skiriasi pasukimu per 90 laipsnių. (1.1 pav.)[5]

-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1

1.1 pav. Sobel algoritmo 3x3 matricos S1 (kairėje) S2 (dešinėje)

1.1 pav. pavaizduotos matricos gali būti taikomos atskirai, kad apskaičiuoti gradiento dedamąsias (Gx ir Gy). Gradientas yra apibūdinamas kaip pirmoji išvestinė, o gradiento operatorius yra išvestinės operatorius. Nuolatinei funkcijai f(x,y) jos gradientas yra dvi pirmosios išvestinės kiekviena kryptimi ir apskaičiuojamas pagal (1.1) formulę.

$$\nabla f(x,y) = \begin{bmatrix} G_x & G_y \end{bmatrix}^T = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial x} \end{bmatrix} \qquad (1.1)$$

Gradiento amplitudė ir nuokrypio kampas atitinkamai apskaičiuojami pagal 1.2 ir 1.3 formules.

$$mag(\nabla f) = \left|\nabla f_{(2)}\right| = \begin{bmatrix} G_x^2 & G_y^2 \end{bmatrix}^{1/2} \qquad (1.2)$$

$$\phi(x,y) = \arctan\left(G_x / G_y\right) \qquad (1.3)$$

Šie skaičiavimai atliekami kiekvienai vaizdo taško vietai. 1.1 pav. pavaizduotomis matricomis Gx ir Gy reikšmės yra aproksimuojamos. Kiekvienai matricai atliekama sasukos skaičiavimo operacija. Viena iš matricų nustato vertikalius kraštus, kita – horizontalius. Maksimali dviejų

sasukų reikšmė naudojama kaip išėjimas – krašto amplitudė. Sasuka skaičiuojama pagal (1.4), (1.5), (1.6) formules[6].

$$g_1(x,y)=\sum_{k=-1}^{1}\sum_{l=-1}^{1}S_1(k,l)f(x+k,y+l) \quad (1.4)$$

$$g_2(x,y)=\sum_{k=-1}^{1}\sum_{l=-1}^{1}S_2(k,l)f(x+k,y+l) \quad (1.5)$$

$$g(x,y)=g_1^2(x,y)+g_2^2(x,y) \quad (1.6)$$

Jeigu g1 daugiau už g2, fiksuojamas kraštas vertikalia kryptimi. Jei vaizdo taškas yra f(x,y), šis taškas yra nurodomas kaip krašto taškas, jei funkcija patenkina vieną iš sąlygų:

$$g(x,y)>4\times\sum_{i=1}^{row}\sum_{j=1}^{list}g^2(i,j)$$

$$g_1(x,y)>g_2(x,y)$$

$$g(x,y-1)\le g(x,y)$$

$$g(x,y)\ge g(x,y+1)$$

$$g(x,y)>4\times\sum_{i=1}^{row}\sum_{j=1}^{list}g^2(i,j)$$

$$g_1(x,y)>g_2(x,y)$$

$$g(x-1,y)\le g(x,y)$$

$$g(x,y)\le g(x+1,y)$$

1.1.2 Roberts kraštų aptikimo algoritmas

Šis metodas paremtas erdvinio gradiento apskaičiavimu[5]. Tam yra naudojamos 2x2 matricos (1.2 pav.)

+1	0
0	-1

0	+1
-1	0

1.2 pav. 2x2 Roberts algoritmui naudojamos matricos

Kaip ir sobel algoritme šios matricos skiriasi pasukimu per 90 laipsnių. Tik priešingai nei sobel algoritme matricos slenka 45 laipsniu kampu per vaizdo taškus įėjime. Gradiento amplitudė kaip ir sobel algoritme apskaičiuojama pagal (1.2) formulę. Tik kampinė orientacija apskaičiuojama pagal (1.7) formulę.

(1.7)

$$\theta = \arctan(Gy / Gx) - 3\pi / 4$$

1.1.3 Laplaso Gauso algoritmas

Šis algoritmas naudoja 2-as vaizdo erdvines išvestines[5]. Taip yra sužinomi regionai kur vyksta greitas pasikeitimas vaizde. Šis algoritmas dažniausiai taikomas kai būna nufiltruojami triukšmai Gauso glodinimo filtru. Laplaso operatorius apskaičiuojamas pagal (1.8) formulę:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \qquad (1.8)$$

Laplaso aproksimacijai panaudojamos 3x3 matricos, su kuriomis atliekama sasukos operacija (1.3 pav.).

1	1	1
1	-8	1
1	1	1

-1	2	-1
2	-4	2
-1	2	-1

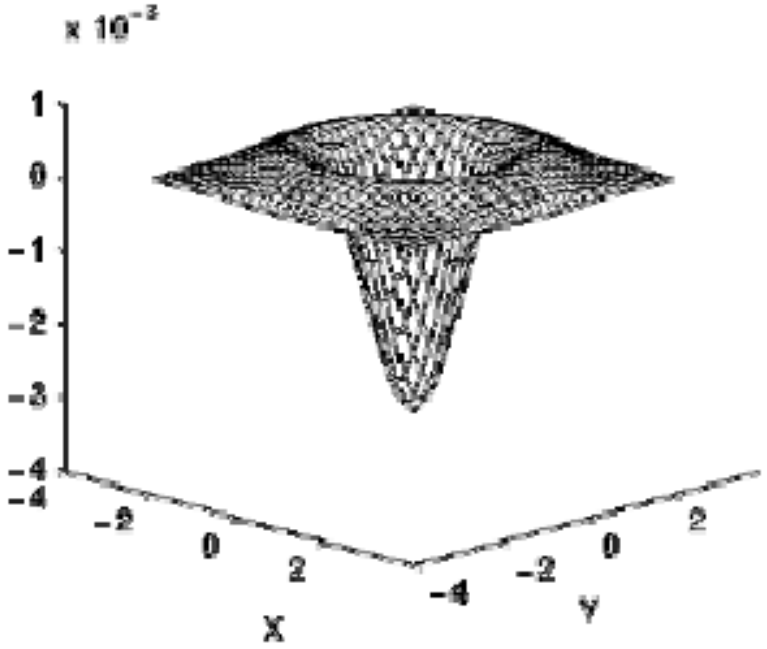
1.3 pav. 3x3 matricos naudojamos Laplaso Gauso kraštų aptikimo algoritme

Taip pat yra plačiai taikomas metodas apskaičiuoti sasuką tarp Gauso filtro ir Laplaso matricos ir tik tada atlikti sasuką su vaizdu. Tokiu būdu pasiekiamas greitesnis užduoties vykdymas, nes reikalinga mažiau aritmetinių veiksmų. Taip pat šis hibridinis filtras gali būti paruoštas iš karto, todėl nebereikia atlikti dviejų sasukų.

Algoritmo funkcija gali būti išreikšta (1.9) formule:

$$LoG(x,y) = -1/\pi\sigma^4 \left[1 - \left(\frac{x^2 + y^2}{2\sigma^2}\right)\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (1.9)$$

Grafinė funkcijos išraiška pavaizduota 1.4 pav.



1.4 pav. Grafinis Laplaso Gauso funkcijos atvaizdavimas

1.1.4 Canny algoritmas

Šis algoritmas yra atliekamas keliais žingsniais[5]. Visų pirma atliekamas vaizdo filtravimas Gauso filtru, kad panaikintų triukšmus. Toliau yra atliekama kraštų paieška, tam panaudojamas sobel algoritmas. Kitas žingsnis yra surasti kraštų kryptį, ji randama pagal (1.10) formulę.

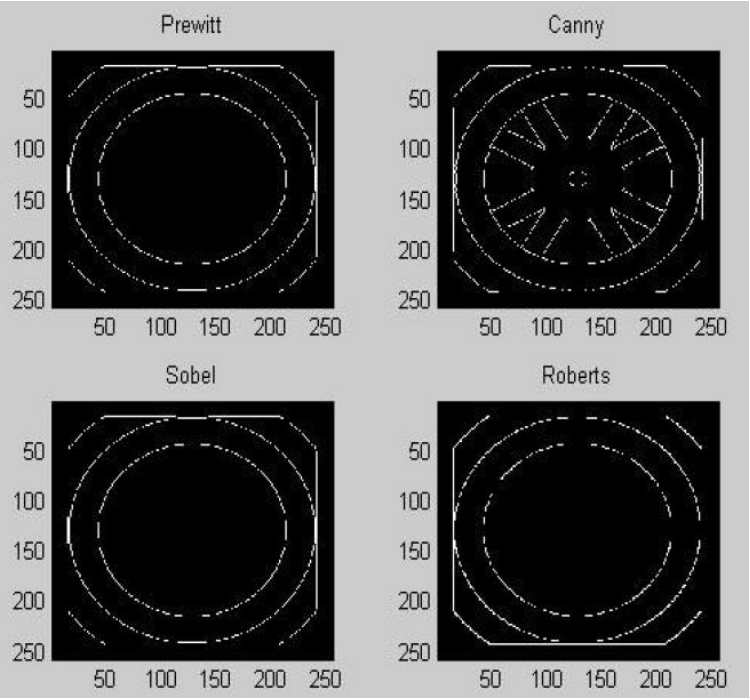
$$\theta = arctg\left(\frac{Gy}{Gx}\right) \qquad (1.10)$$

Kai kraštų kryptis aiški, reikia ją susieti su kryptimi, kuri gali būti atsekta paveiksle. Taigi visi vaizdo taškai yra suskirstomi į kryptis kuriomis jie turi būti orientuoti. Šiuos kryptys yra: 0, 45, 90, 135 laipsniai. Iš to seka, kad kraštų orientacija turi būti priskirta vienai iš minėtų reikšmių, priklausomai nuo to kuriai reikšmei kampas yra artimiausias. Pavyzdžiui, jei kampas yra 5 laipsniai, tai šis vaizdo taškas bus priskirtas 0 laipsnių.

Toliau yra atliekama ne maksimalaus slopinimo operacija (angl. non-maximum suppression). Ši operacija yra vykdoma sekant krašto liniją ir nustatant vaizdo taškų vertes į 0, kurios nėra tikri kraštai, kitaip tariant kraštas yra ploninamas.

Paskutinė operacija yra histerezė, kurios panaudojimas pašalina susiliejimus ir nutrūkimus. Tam yra panaudojama kraštų vidurkio reikšmė. Pagal ją kiekvienas krašto taškas kurio vertė didesnė už vidurkio vertę yra priskiriamas kraštui.

1.5 paveiksle pateikiamas visų kraštų aptikimo algoritmų palyginimas.



1.5 pav. Kraštų aptikimo algoritmų palyginimas

Pagal 1.5 pav. galima teigti, kad geriausiai kraštus aptinka Canny algoritmas.

1 lentelėje yra pateikiami algoritmų privalumai ir trūkumai

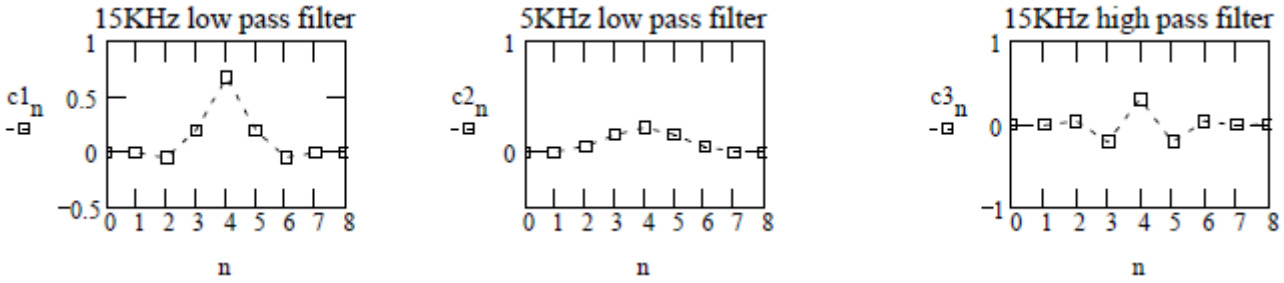
1 lentelė

Kraštų aptikimo algoritmų privalumai ir trūkumai

Algoritmas	**Privalumai**	**Trūkumai**
Sobel, Prewitt, Roberts	Kraštu ir jų orientacijų aptikimo paprastumas	Jautrumas triukšmui, netikslumas
Laplaso Gauso	Teisingų kraštų vietų radimas, tikrinama platesnė erdvė vaizde.	Keliu funkcijų atlikimas kampuose ir lenktuose vietose kur yra pilki regionai. Nerandama krašto orientacija dėl Laplaso operatoriaus
Canny	Naudoja tikimybes, nustatyti paklaidas. Pagerintas santykis signalas triukšmas. Geresnis aptikimas, ypač triukšmingomis sąlygomis	Sudėtingi skaičiavimai, užima nemažai laiko

1.2 Ribotos impulsinės reakcijos (RIR) filtrai

Tokių filtrų dažninės charakteristikos paremtos tam tikrais koeficientų rinkiniais. Skirtingų filtrų dažninės charakteristikos pavyzdžiai pavaizduoti 1.6 pav[7].



1.6 pav. RIR filtrų dažninės charakteristikos su skirtingai koeficientais

RIR filtras atlieka sasukos operaciją su iš kart žinomais koeficientais. Sasuka yra standartinė operacija naudojama vaizdų filtravimui (blukinimui, ryškinimui, suspaudimui, triukšmų šalinimui ir kt.)[8]. Sasuka apskaičiuojama pagal (1.11) formulę:

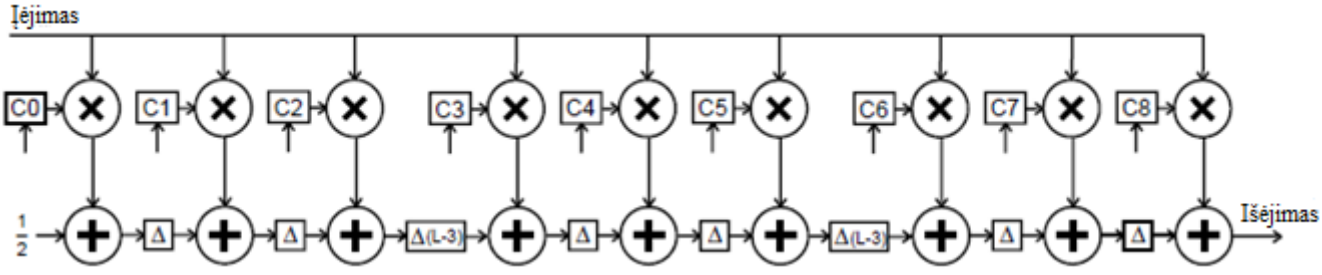
$$p = \sum_{i=0}^{n-1} v(i) \cdot c(i); \qquad (1.11)$$

čia n – filtro matricos koeficientų skaičius;

v(i) – filtruojamo vaizdo taškas;

c(i) – filtro koeficientas.

Jeigu filtro dydis yra m x m, tada kiekvienas nufiltruoto vaizdo taškas priklauso nuo m^2 gretimų vaizdo taškų ir dar nuo m^2 sandaugų ir sumų, kurių reikia kiekvienam išrinkimui.



1.7 pav. 3x3 branduolio dydžio RIR filtras, atliekantis sasukos operaciją

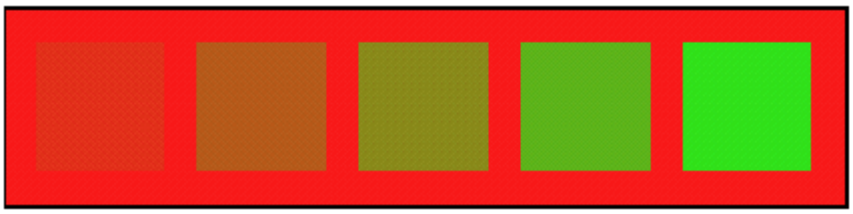
1.7 pav. pavaizduotas 3x3 sasukos operaciją atliekantis filtras.[9] Vaizdo taškai imami iš pradinio vaizdo ir iškarto padauginami iš visų 9 filtro koeficientų (C0-C8). Dalinės sumos yra

kaupiamos sumavimo elementuose. Po to tos reikšmės yra sudedamos tarpusavyje. Po kiekvienos tokios sudėties aktyvuojamas vieno vaizdo taško nuskaitymo laiko vėlinimas. Atlikus 3 sumas vienoje vaizdo eilutėje, aktyvuojamas visos vaizdo eilutės nuskaitymo laiko vėlinimas minus 3 vaizdo taškų nuskaitymo laiko vėlinimas, nes jie jau buvo užvėlinti. Toliau tokia pati operacija yra atliekama ir su kitomis dviejomis vaizdo eilutėmis. Filtro išėjime yra gaunamas nufiltruoto vaizdo taškas.

1.3 Alfa smulkinimo algoritmas

Vienas iš tokių algoritmų naudojamas pirminiam vaizdų apdorojimui prieš sudėjimą yra alfa smulkinimas (angl. Alpha Blending)[10]. Algoritmas paremtas apšvietimo aproksimacijomis per permatomus ir nepermatomus objektus.

Permatomumui nusakyti įvedamas permatomumo koeficientas, kuris nusako kiek šviesos energijos objektas blokuos. 0 vertė rodys, kad objektas yra skaidrus, 1 vertė – visiškai nepraleidžia šviesos. 1.8 pav. yra pavaizduoti vaizdai su skirtingais permatomumo koeficientais.



1.8 pav. Objektas su skirtingais permatomumo koeficientais (0,1, 0,3, 0,5, 0,7, 0,9)

Spalvinį koeficientą galima apskaičiuoti pagal (1.12) formulę.

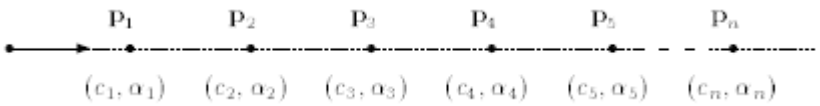
$$\alpha c + (1 - \alpha) b \qquad (1.12)$$

Alpha – permatomumo koeficientas

b – fono spalva

c – objekto spalva

Vaizde su daugiau objektų reikia atsižvelgti į kiekvieno taško permatomumą. Yra sekama linija vaizde fiksuotais tarpais yra spalvos sujungiamos kartu, kaip parodyta 1.9 pav.



1.9 pav. Sekamos linijos taškai, kuriuose atliekamas sujungimas

Spalva, kurią mato kamera yra apskaičiuojama pagal (1.13) formulę

$$\alpha_1 c_1 + (1 - \alpha_1) b_1 \qquad (1.13)$$

b1 yra spalva nustatyta kameros 1.9 pav. pavaizduotose taškuose pradedant nuo p2. (1.14) formulėje b1 yra fono spalva

$$b_1 = \alpha_2 c_2 + (1 - \alpha_2) b_2 \qquad (1.14)$$

b2 yra spalva nustatyta taškuose pradedant nuo p4.

Taigi bendro atveju formulė atrodytų taip (1.15):

$$b_i = \alpha_{i+1} c_{i+1} + (1 - \alpha_{i+1}) b_{i+1} \qquad (1.15)$$

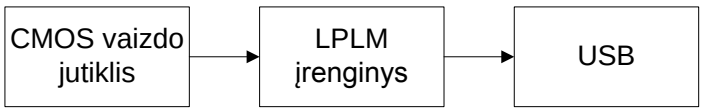
2. Panašių tyrimų apžvalga

Šiame skyriuje bus aptariamos sistemos, kuriose naudojamas vaizdų apdorojimas.

2.1 Automobilio vaizdo kameros vaizdo atpažinimo sistema

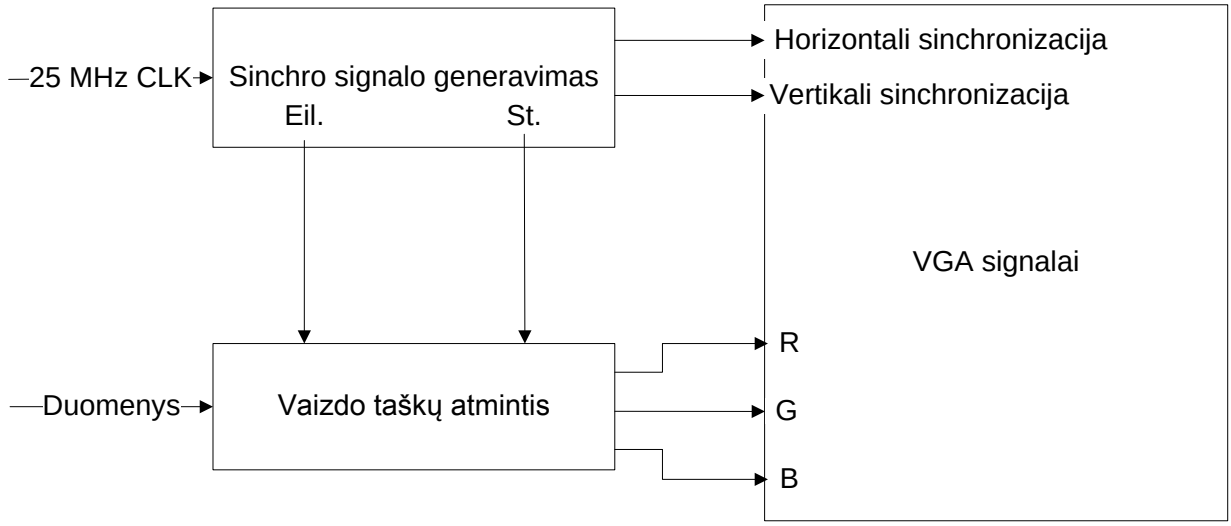
Nagrinėjama automobilio vaizdo kameros vaizdo atpažinimo sistema, panaudojant LPLM įrenginį[11].

Šioje sistemoje panaudotas 32 bitų įkeliamasis procesorius duomenų krypties valdymui. Vaizdo atvaizdavimo modulis buvo aprašytas Verilog[12] programavimo kalba. Procesorius buvo programuojamas C programavimo kalba. Sistemos struktūrinė schema pavaizduota 2.1 pav.



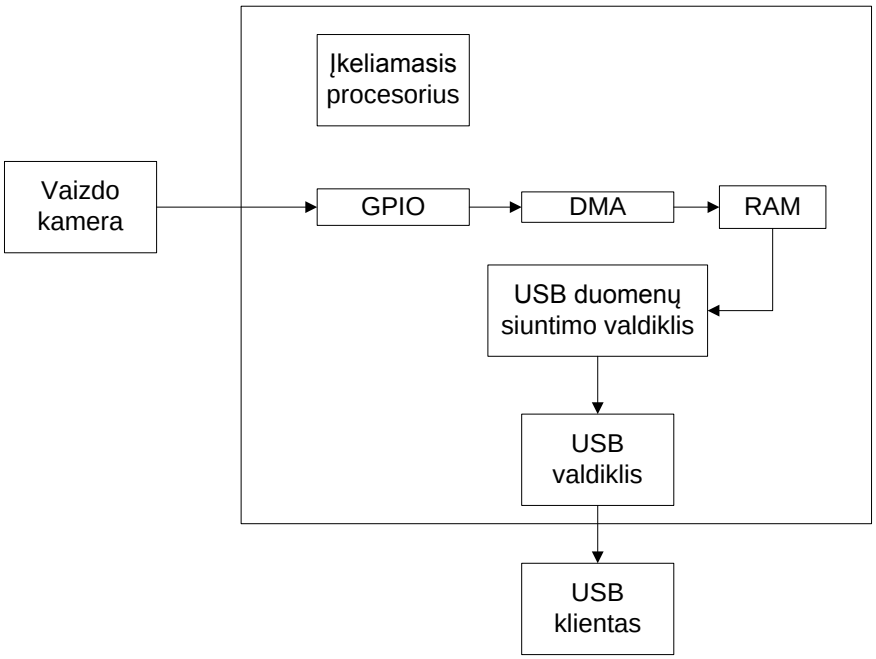
2.1 pav. Sistemos struktūrinė schema

Kadangi vaizdas yra gaunamas VGA formatu, todėl buvo panaudotas VGA valdiklis, kuris buvo integruotas bandymų plokštėje. VGA valdiklio struktūrinė schema yra pavaizduota 2.2 pav.



2.2 pav. VGA valdiklio struktūrinė schema

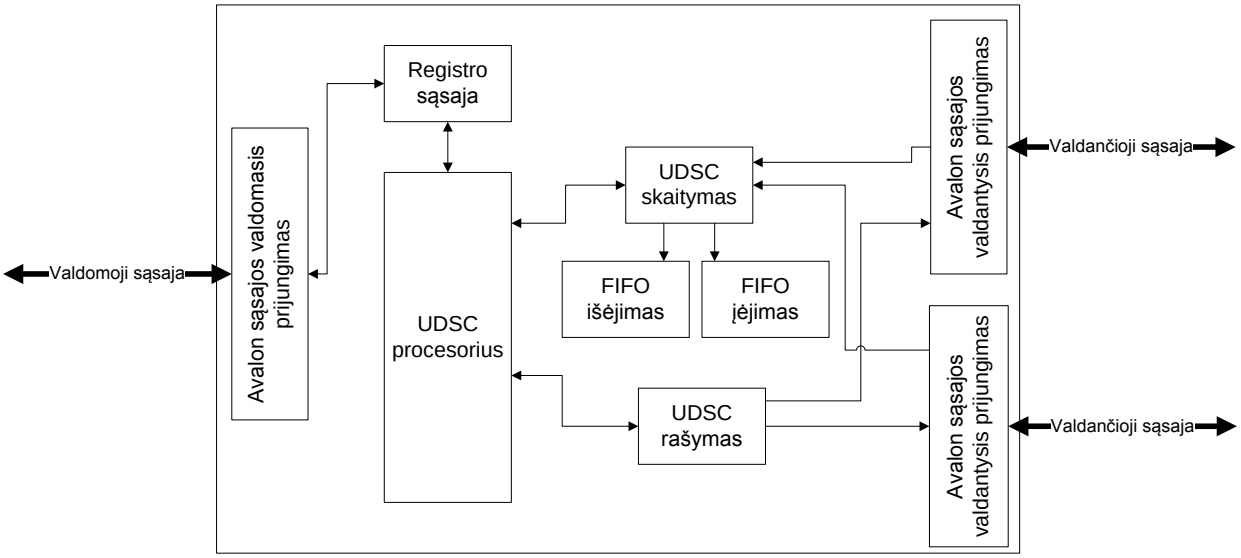
2.3 pav. pavaizduota sistemos, esančios LPLM įrenginyje struktūrinė schema.



2.3 pav. LPLM įrenginio struktūrinė schema

Šioje sistemoje įkeliamasis procesorius valdo įvairius procesus, kurie yra atliekami sistemoje. Lygiagretus įėjimas/išėjimas(PIO) naudojamas prijungti vaizdo kamerą prie sistemos. PIO Registro dydis nustatomas kas kart su taktinio dažnio kylančiu frontu ir yra lygus bitų skaičiui viename vaizdo taške. Šie vaizdo taškai tada yra siunčiami į RAM atmintį. Tiesioginės kreipties į atmintį valdiklis naudojamas palengvinti procesoriaus darbui. USB sąsaja panaudota, kad duomenys būtų siunčiami į kompiuterį tolesniam jų apdorojimui.

USB duomenų srauto valdiklis (UDSC) atlieka duomenų perdavimą iš USB į išorinę atmintį ir atgal be programinės įrangos intervencijos dėl galinių registrų konfigūracijos. Procesorius gali perduoti UDSC valdikliui atminties valdymo užduotis, kad lygiagrečiai atliktų kitas užduotis. 2.4 pav. pavaizduota UDSC struktūrinė schema.



2.4 pav. UDSC struktūrinė schema

Taip pat buvo panaudotas JTAG valdiklis, kad perkelti sistemą į LPLM įrenginį, sistemos identifikacijos modulis panaudotas, kad suteikti procesoriui identifikacijos numerį FIFO atmintys panaudotos kaip RAM atminties įėjimai/išėjimai.

Bandymo rezultatai pateikti 2 lentelėje

2 lentelė

Sistemos resursų išnaudojimas

Resursas	Panaudojimas	Visas kiekis	Procentinė išraiška, %
Visi loginiai elementai	1554	33216	5
Visos kombinacinės funkcijos	1266	33216	4
Dedikuoti loginiai elementai	1040	33216	3
Visi registrai	1055	—	—
Visi išvadai	425	475	89
Visa atmintis	62416	483840	13
Visi PLL	1	4	25

Buvo pakoreguotas kameros taktinis dažnis, kad sistema sugebėtų apdoroti vaizdus 23 kadrais per sekundę. Vaizdo raiška 640x480. Per USB sąsają perduodamų duomenų greitis siekė 170Mbps

2.2 Vaizdų apdorojimo algoritmai įgyvendinti Verilog programavimo kalba

Nagrinėjamos sistemos aprašytos Verilog programavimo kalba[13]. Šis tyrimas atliktas išsiaiškinti kaip Verilog programavimo kalba susidoroja su vaizdų apdorojimo algoritmais. Šia kalba parašytas algoritmas leidžia rašyti ir skaityti ASCII simbolius, bet neleidžia apdoroti .bmp ir .jpeg failų.

Pirmoji sistema – kontrasto manipuliacija. Tai buvo pasiekta priskiriant tamsiausią vaizdo taško vertę juodai spalvai, o šviesiausią – baltai. Verilog kalba parašytas kodas šiai sistemai pavaizduotas 2.5 pav.

```
if (value > threshold) begin
	tempR = Rin + valueToAdd;
	if (tempR > 256)
		Rout = 255;
	else
		Rout = Rin + valueToAdd;
	tempG = Gin + valueToAdd;
	if (tempG > 256)
		Gout = 255;
	else
		Gout = Gin + valueToAdd;
	tempB = Bin + valueToAdd;
	if (tempB > 256)
		Bout = 255;
	else
		Bout = Bin + valueToAdd;
end
```

```
if (value < threshold) begin
	tempR = Rin - valueToSubstract;
	if(tempR[8] == 1)
		Rout = 0;
	else
		Rout = Rin - valueToSubtract;
	tempG = Gin - valueToSubtract;
	if(tempG[8] == 1)
		Gout = 0;
	else
		Gout = Gin - valueToSubtract;
	tempB = Bin - valueToSubstract;
	if(tempB[8] == 1)
		Bout = 0;
	else
		Bout = Bin - valueToSubtract;
end
```

2.5 pav. Verilog kalba parašytas kodas kontrasto manipuliacijai

2.5 pav. 1 dalyje matyti, kad vaizdas yra RGB formato, todėl veiksmai atliekami su kiekviena spalva.

Kita sistema – ryškumo manipuliacija

Tokios sistemos tikslas yra paryškinti arba sumažinti tam tikrų vaizdo vietų ryškumus. 2.6 pav. esančiame kode ir pridedama arba atimama konstanta.

```
if(sign == 1) begin
	tempR = Rin + value;
	if (tempR > 256)
		Rout = 255;
	else
		Rout = Rin + value;
	tempG = Gin + value;
	if (tempG > 256)
		Gout = 255;
	else
		Gout = Gin + value;
	tempB = Bin + value;
	if (tempB > 256)
		Bout = 255;
	else
		Bout = Bin + value;
end
if(sign == 0) begin
	tempR = Rin - value;
	if (tempR[8] == 1)
		Rout = 0;
	else
		Rout = Rin - value;
	tempG = Gin - value;
	if (tempG[8] == 1)
		Gout = 0;
	else
		Gout = Gin - value;
	tempB = Bin - value;
	if (tempB_b[8] == 1)
		Bout = 0;
	else
		Bout = Bin - value;
	end
```

2.6 pav. Ryškumo manipuliacijos algoritmas

Taip pat vienas iš vaizdo apdorojimo būdų yra vaizdo inversija. Ji yra atliekama apdauginus vaizdo taškų vertes iš (-1) ir reikalui esant galima pridėti kažkokią vertę, norint labiau paryškinti ar nublankinti tam tikras vietas.

```
value2 = (Rin + Gin + Bin)/2;
value4 = (Rin + Gin + Bin)/4;
value  = (value2 + value4)/2;
     Rout = 255 - value;
     Gout = 255 - value;
     Bout = 255 - value;
```

2.7 pav. Inversija

2.7 pav. pavaizduota vaizdo inversija. Vaizdu vidurkiniams daromas, kad gauti juodai baltą vaizdą.

Kitos sistemos pavyzdys yra slenksčio nustatymas. Naudojant šį metodą galima izoliuoti objektą nuo fono. Taip pat atliekant šią operaciją visos vaizdo taškų vertės transformuojamos į dvi skirtingas vertes (balta ir juoda). Slenkstis nustatomas spendžiant lygčių sistemą, kai vertė mažesnė už nustatytą slenkstinę vertę, priskiriama yra vienai vertei, kaip didesnė – kitai. Verilog kalba parašytas kodas atrodo taip (2.8 pav.).

```
if (value > threshold) begin
      Rout = 255;
      Gout = 255;
      Bout = 255;
end
if (value < threshold) begin
      Rout = 0;
      Gout = 0;
      Bout = 0;
end
```

2.8 pav. Slenksčio nustatymo algoritmas

Toliau pateikiami tyrimo rezultatai.

2.9 pav. Kontrasto keitimas

2.10 pav. Ryškumo keitimas

2.11 pav. Inversija

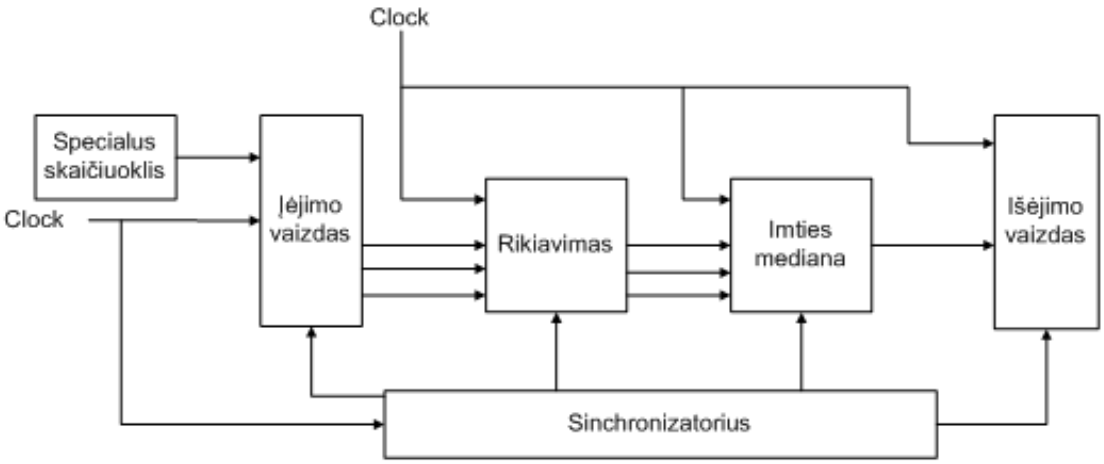
2.12 pav. Slenksčio nustatymas

Pagrindinis šio tyrimo tikslas buvo išsiaiškinti kaip Verilog programavimo kalba projektuoti vaizdo apdorojimo sistemas, taip pat kaip tinkamai susidoroti su tuo, kad teko dirbti su ASCII formato simboliais.

2.3 Vaizdų pagerinimo algoritmų tyrimas

Šiame tyrime buvo apžvelgti tokie vaizdo pagyvinimo algoritmai kaip: medianos filtras, kontrasto ištempimas, histogramos išlyginimas, neigiama vaizdo transformacija, galios dėsnio transformacija[14].

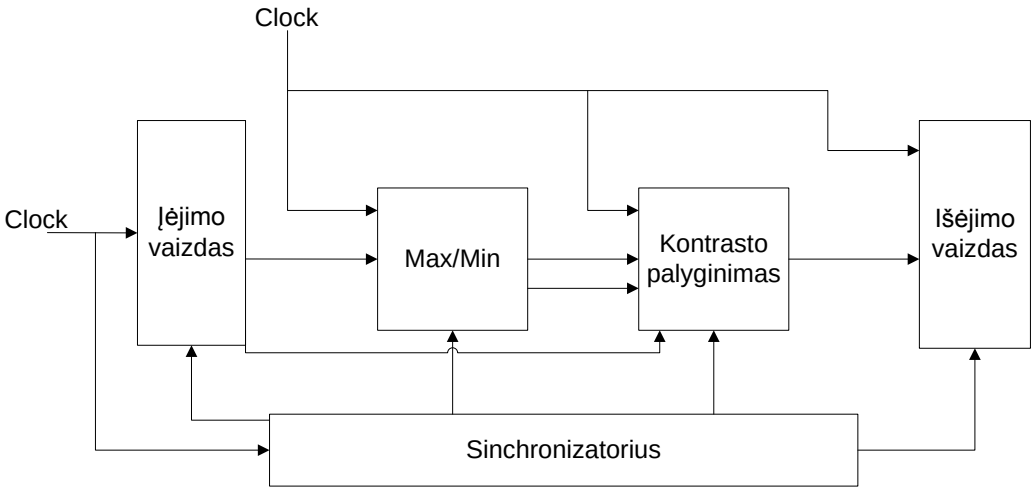
Medianos filtro struktūrinė schema pavaizduota 2.13 pav.



2.13 pav. Medianos filtro struktūrinė schema

Schemoje specialus skaičiuoklis yra suformuota filtro kaukė, kuri yra uždedama rikiavimo bloke. Kitame bloke (Imties mediana) yra keičiama vaizdo taško vertė pagal turimą kaukę.

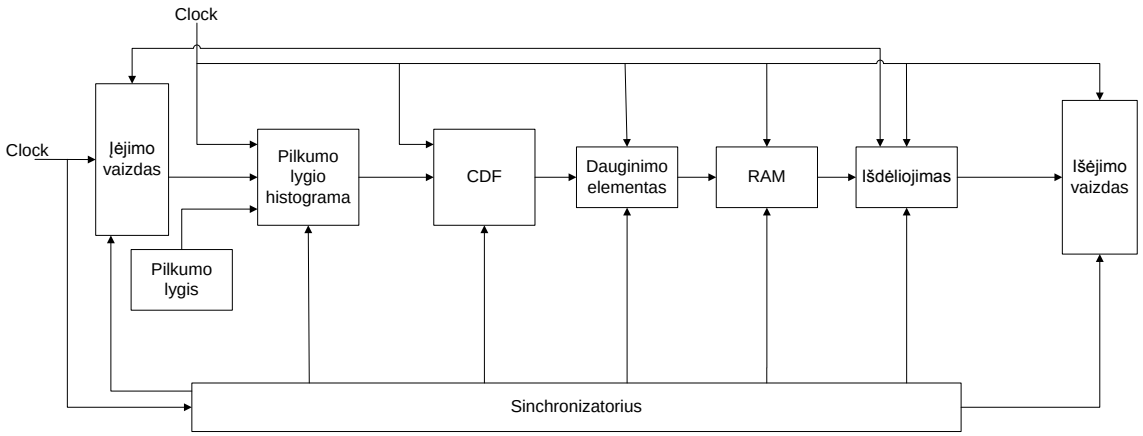
2.14 pav. pavaizduota kontrasto ištempimo struktūrinė schema



2.14 pav. Kontrasto ištempimo struktūrinė schema

Algoritmas yra panašus kaip ir 2.13 pav. pavaizduotos sistemos, tik čia yra Max/Min blokas, kuris skaičiuoja nustatytas minimalias ir maksimalias vaizdo taškų reikšmes. Kontrasto palyginimo blokas lygina ar reikšmė yra didesnė ar mažesnė už slenkstines vertes. Maksimumo atveju vaizdo taškai prilyginami 255, minimumo – 0.

2.15 pav. pavaizduota histogramos išlyginimo struktūrinė schema.

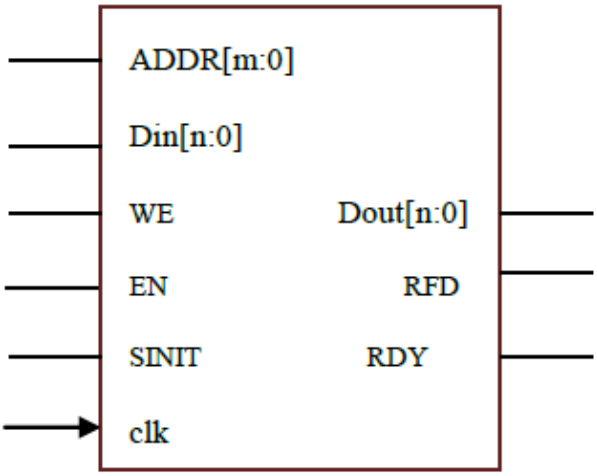


2.15 pav. Histogramos išlyginimo struktūrinė schema

3 pav. pavaizduotoje schemoje vaizdas imamas juodai baltas. Pilkumo lygio blokas skaičiuoja kiekvieno pilko vaizdo taško vertę. CDF elementas atlieka verčių skaičiavimą, kurios yra mažesnės nei nustatyta vertė. Dauginimo elementas padaugina CDF elemento reikšmes su pastovia verte. Toliau yra atliekamas vaizdo taškų išdėliojimas ir gaunamas pakeistas vaizdas.

2.4 Canny algoritmo tyrimas

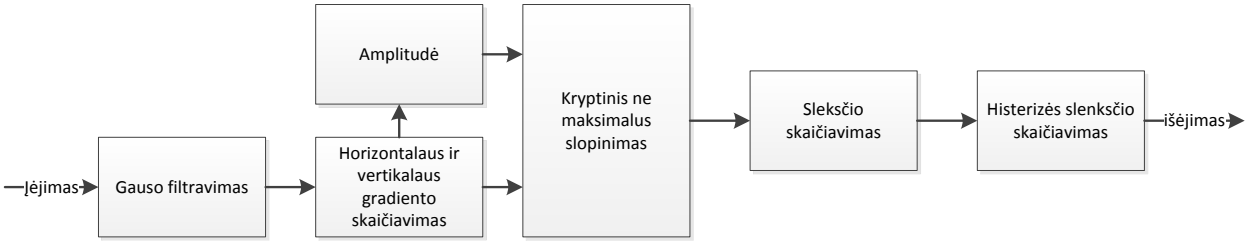
Šiame tyrime buvo atsižvelgta į Canny kraštų aptikimo algoritmą[15]. Tyrimas buvo atliekamas su bespalviu 128x128 raiškos vaizdu. Rezultatams išvesti panaudotas VGA monitorius. Įėjimo vaizdui talpinti buvo panaudota ROM atmintis (2.16 pav.)



2.16 pav. ROM atmintis, naudojama vaizdui talpinti

Operacijos atliekamos, kai sulaukiama aktyvaus taktinio dažnio šaltinio fronto. Šis nustatomas kuriant elementą ar jį modifikuojant.

Canny algoritmo struktūrinė schema LPLM įtaise pavaizduota 2.17 pav.



2.17 pav. Kraštų aptikimas pagal Canny algoritmą

Pagal 2.17 pav. pavaizduotą schemą, iš pradžių atliekamas vaizdo glodinimas pagal Gauso filtravimo algoritmą. Toliau yra atliekamas gradiento skaičiavimas, pasinaudojant gradiento filtru. Gradiento dedamosios apskaičiuojamos pagal (2.1) ir (2.2) formules.

$$G_x = I(x+1, y) - I(x-1, y) \qquad (2.1)$$

$$G_y = I(x, y+1) - I(x, y-1) \qquad (2.2)$$

Horizontaliam ir vertikaliam gradientui yra naudojamos dvi skirtingos kaukės (2.18 pav.)

0	0.5	0
0	0.5	0
0	0.5	0

a

0	0	0
0.5	0.5	0.5
0	0	0

b

2.18 pav. Kaukės, naudojamos apskaičiuoti gradientus

Kaukių reikšmės yra statomos į (2.3) ir (2.4) formules.

$$\mathrm{Gx} = (\mathrm{P2} + \mathrm{P5} + \mathrm{P8})/2 \qquad (2.3)$$

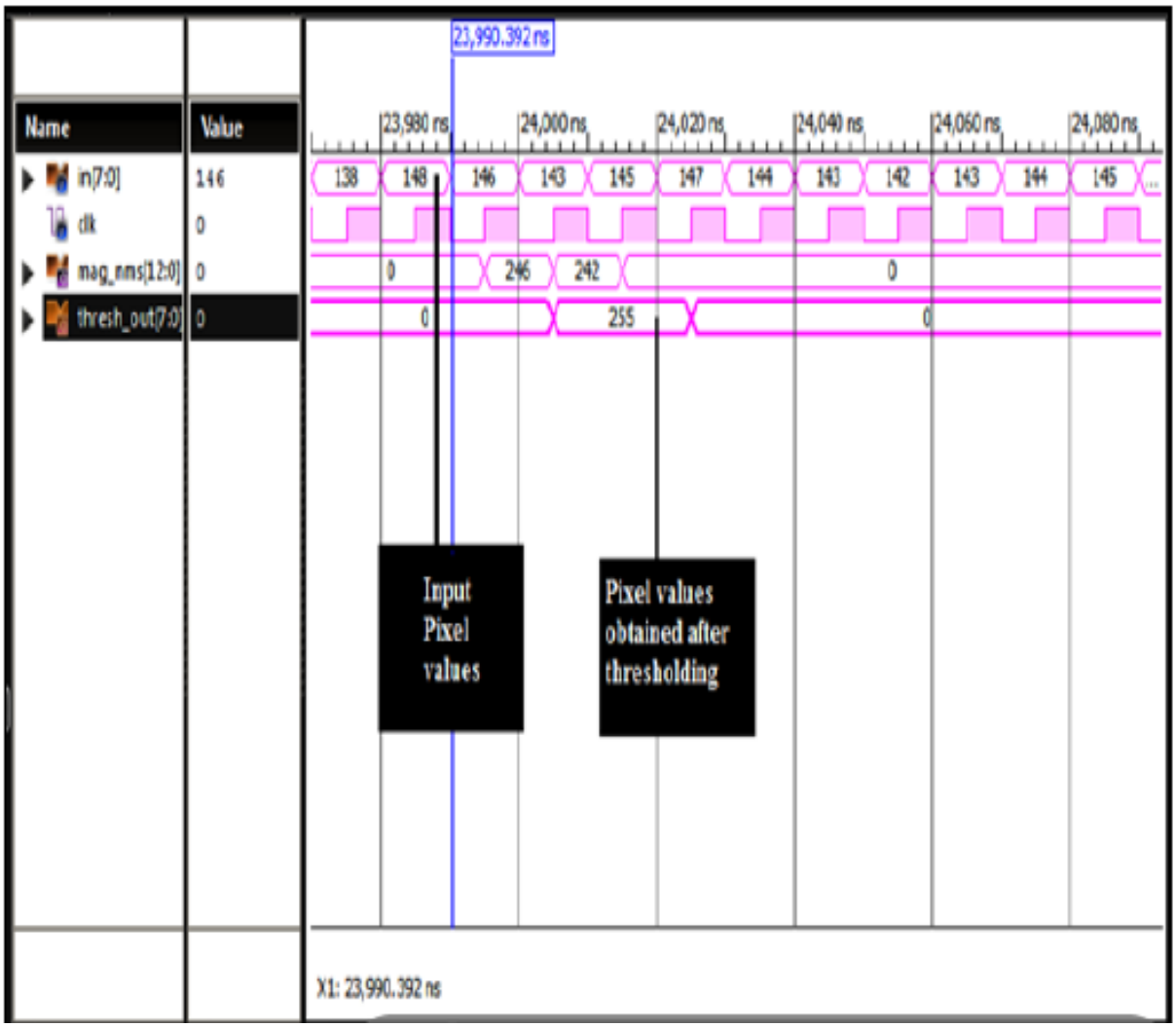
$$\mathrm{Gy} = (\mathrm{P4} + \mathrm{P5} + \mathrm{P6})/2 \qquad (2.4)$$

Amplitudės modulis skaičiuoja gradiento amplitudę ((2.5) formulė)

$$Gmag = \sqrt{Gx^2 + Gy^2} \qquad (2.5)$$

Tiesioginis ne maksimalus slopinimas yra atliekamas, norint surasti krašto kryptį pagal gradientą (ar tai bus tamsus, ar šviesus). Po tokios operacijos atsiranda šiek tiek artefaktų, kurie slopinami nustatant histerezės slenkstį.

Rezultatai pavaizduoti 2.19 pav.



2.19 pav. Canny algoritmo simuliacija

Iš simuliacijos matyti, kad labai gerai yra atskiriami kraštai. Ten kur nėra krašto yra visiškai juoda, ten kur yra – balta.

3. Vaizdų apdorojimo algoritmų tyrimo metodinės priemonės

Tyrimui buvo panaudota tiek programinė tiek aparatūrinė įranga. Programinė įranga buvo skirta sudaryti sistemą ir ja testuoti. Aparatūrinė įranga skirta išbandyti įrenginį.

3.1 Quartus II

Quartus II[16] tai kompanijos „Altera“ programinės įrangos paketas, skirtas kurti sistemas, programuojant loginių matricų lustus (angl. FPGA). Šis programinės įrangos paketas turi labai įvairių įrankių, skirtų sistemų sudarymui ir testavimui.

Sistemas galima kurti keliais būdais:

- Rašant rankiniu būdu viena iš HDL (aparatūrinės įrangos aprašymo kalba) kalbų: VHDL[17] arba Verilog.
- Braižant schemą iš standartinių arba savo sukurtų komponentų simbolių.
- Pasinaudojus įrankiu Qsys[18]

Kuriant sistemas HDL kalbomis galima žymiai lanksčiau negu kitomis programinėmis kalbomis. Galimybės praktiškai neribotos ir priklauso nuo programuotojo žinių bagažo. Šiomis kalbomis parašytas programinis kodas yra vykdomas lygiagrečiai. Programuojant šia kalba, programuotojas turi atsižvelgti į visus aspektus, todėl mažiau patyrusiems programuotojams yra didesnė tikimybė padaryti klaidų, taip pat pats programavimo ir klaidų taisymo procesas užtrunka nemažai laiko.

Kitas būdas yra braižyti schemą iš jau paruoštų komponentų. Tai pasiteisina kuriant nedideles sistemas, kuriose nereikia panaudoti daug elementų, bet sudėtingesnių sistemų kūrimui schemos gali tapti labai sunkiai skaitomos, o tai padidina klaidų tikimybę, taip pat ir klaidų aptikimo laikas darosi ilgesnis.

Kitas būdas yra pasinaudoti specialiu įrankiu Qsys, kuriame yra standartinių komponentų bibliotekos. Komponentų įvairovė yra nemaža, nuo nesudėtingų taktinio dažnio generatorių iki sudėtingų vaizdų apdorojimo algoritmų. Taip pat į šį įrankį galima integruoti ir savo aprašytą elementą. Toks būdas pasiteisina kai kuriamos sudėtingos sistemos ar reikia kokio sudėtingesnio komponento sistemai papildyti.

Quartus II programinis paketas turi labai daug įvairių įrankių, tokių kaip „TimeQuest Timing analyzer“[19], kuris skirtas analizuoti laikines sistemos charakteristikas, parodo sistemos stabilumą, aproksimuojant signalo atvykimo laiką kiekviename sistemos mazge, vėlavimo laikus. Dar vienas įrankis yra „Powerplay Power Analysis“[20]. Šis įrankis yra skirtas apskaičiuoti sistemos naudojamą galią, taip pat parodyti kiek koks įtampos šaltinis naudoja srovės. Nurodžius tinkamas

sąlygas įrankis apskaičiuoja kokio tipo ir šiluminės varžos reiktų aušintuvo. Tai labai naudinga parenkant maitinimo šaltinį.

3.2 Altera-Modelsim

Altera-Modelsim[21] yra programinės įrangos paketas skirtas atlikti tiek paprastas tiek sudėtingas LPLM sistemų simuliacijas.

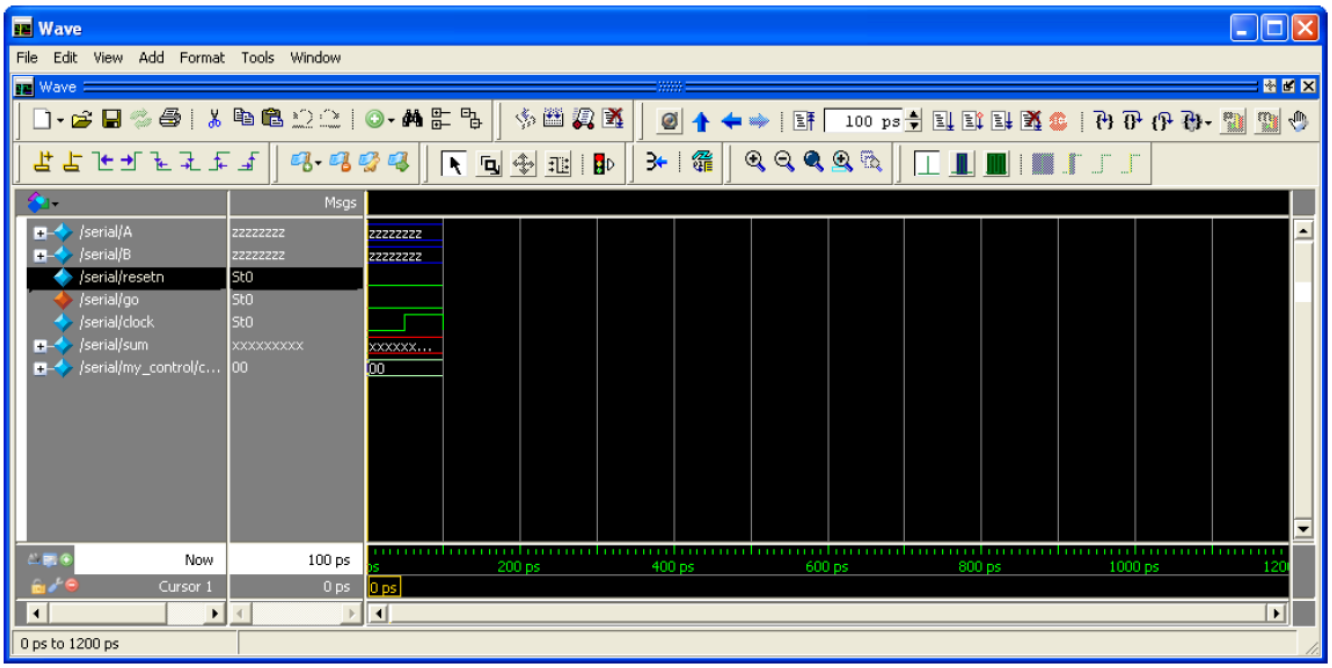
Simuliacija atliekama tada kai projektas būna sukompiliuotas Quartus II aplinkoje. Kompiliavimas turi būti atliktas būtent simuliacijai, t. y. reikia paleisti „EDA netlist“[22] kompiliavimą, kartu su pagrindine kompiliacija. Simuliacijos aplanke yra sukuriamos bylos, kurios yra reikalingos simuliacijai, ten yra aprašyta visos sistemos elgsena.

Norint simuliuoti sistemą neužtenka šios bylos, nes ten tik aprašyta sistemos elgsena, bet nėra išorinio poveikio, t. y. sistema nėra „įjungta“. Išorinį poveikį sistemai galima nurodyti dviem būdais:

- Aprašant įėjimo signalus HDL kalba
- Braižant signalus grafinėje aplinkoje kiekvienam įėjimui atskirai

Jei sistema nėra sudėtinga ir nėra daug pasikartojančių signalų, paprasčiau yra braižyti signalus, to pasėkoje galima vizualiai matyti kaip signalas atrodo, kuriuose vietose įvyksta pokyčiai laike.

Simuliacijos laikas yra tiesiogiai proporcingas sistemos sudėtingumui ir simuliavimo laikui. Pavyzdžiui, norint simuliuoti sudėtingą sistemą 1s. Simuliacija gali užtrukti ir 1 valandą. Programos simuliacijos grafinė aplinka pavaizduota 3.1 pav.



3.1 pav. Altera-Modelsim simuliacijos grafinė aplinka

3.3 De0-nano bandymų plokštė

Bandymams buvo pasirinkta DE0-nano[22] bandymų plokštė, kurioje integruotas Cyclone IV šeimos LPLM lustas EP4CE22F17C6N. Bandymų plokštė pavaizduota 3.2 pav.

3.2 pav. De0-nano bandymų plokštė

Šiame luste yra integruoti tokie elementai:

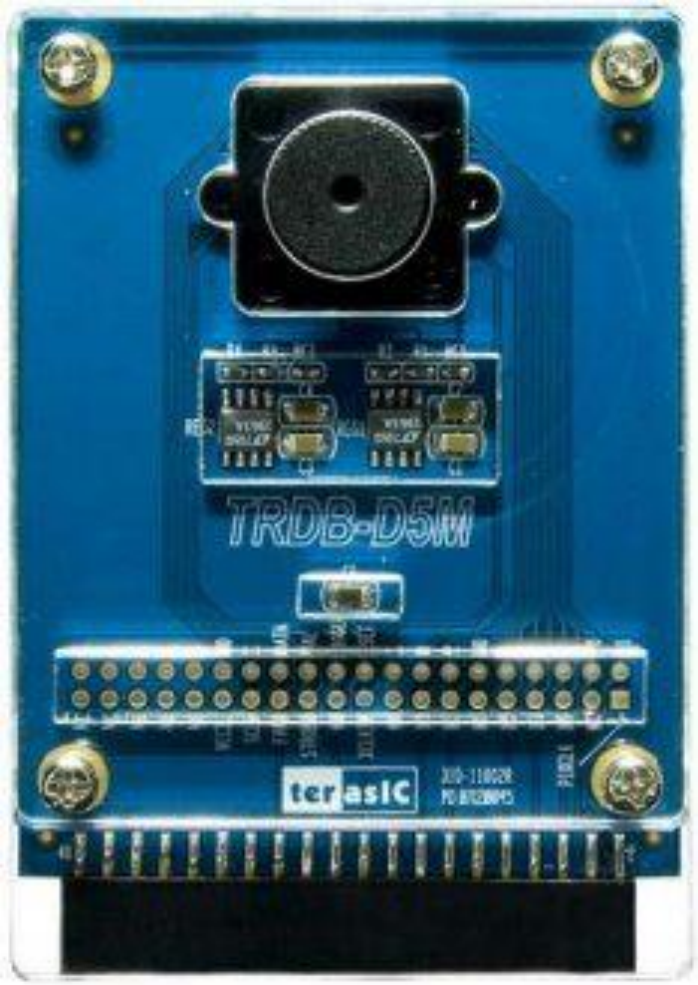
- 22320 loginių elementų (peržiūros lentelės)
- 594 kb atminties
- 66 18x18 dauginimo elementai
- 4 taktinio dažnio dalikliai
- 153 Į/I išvadai

Pačioje plokštėje yra:

- du 40 išvadų GPIO rinkiniai
- vienas 26 išvadų rinkinys su 8 analoginiais įėjimais
- 32 MB SDRAM atmintis
- 2 kb EEPROM atmintis
- 8 žali šviesos diodai
- 2 atsistatantys mygtukai
- 4 jungikliai
- 3 ašių, 13 bitų akselerometras
- 8 kanalų 12 bitų analoginis skaitmeninis keitiklis
- 50 MHz taktinio dažnio šaltinis – kvarcinis rezonatorius

3.4 D5M 5MP skaitmeninė vaizdo kamera

Vaizdo perdavimui iš išorės buvo naudojama D5M[23] vaizdo kamera. Ši kamera specialiai skirta „Altera“ bandymo plokštėms, kurios turi 40 išvadų rinkinius. Kamera pavaizduota 3.3 pav.



3.3 pav. D5M 5MP skaitmeninė vaizdo kamera

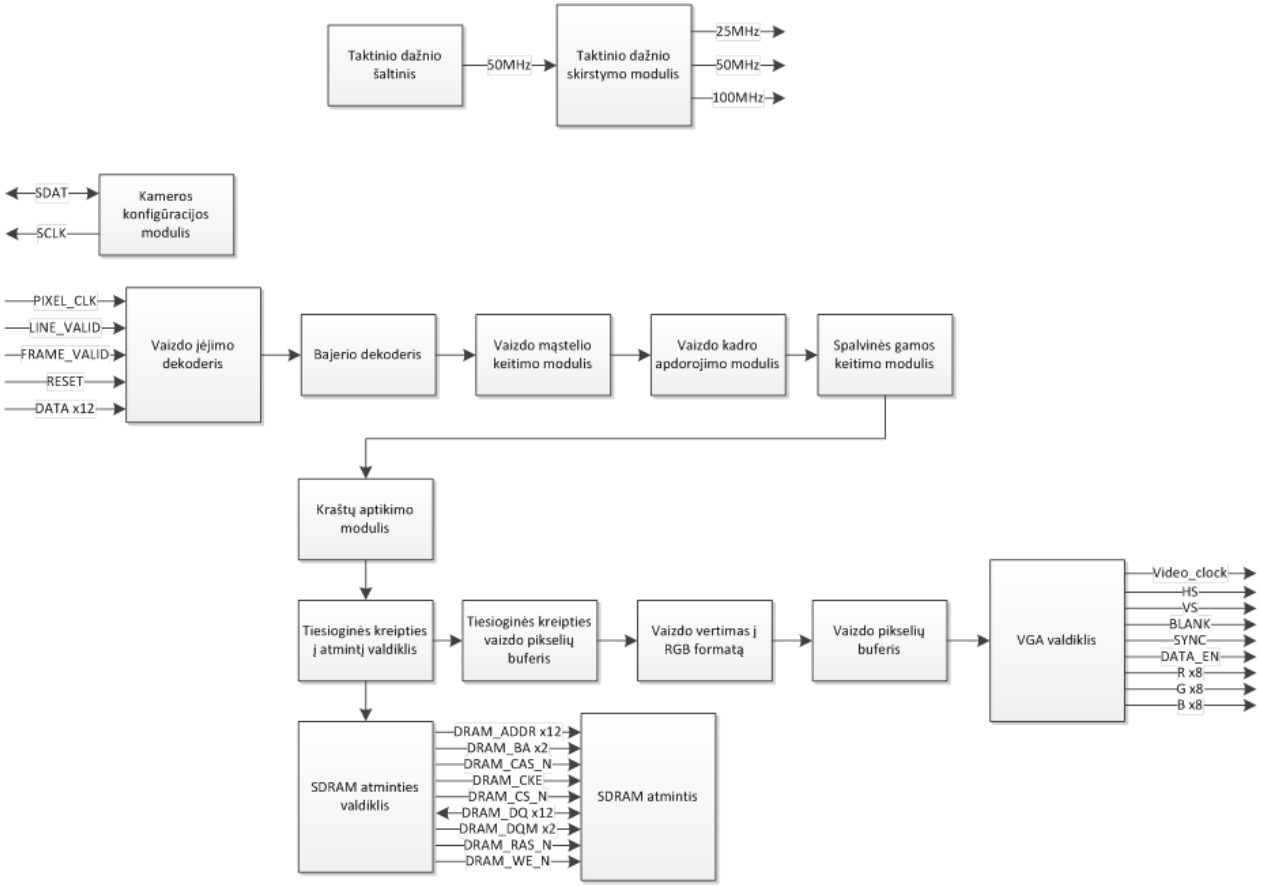
Ši kamera gali veikti tiek filmavimo, tiek fotografavimo režime. Išėjime yra formuojamas Bajerio[24] spalvinis formatas. Maksimali galima raiška – 2592x1944, 15 kadrų per sekunde. VGA raišką (640x480) gali perduoti iki 70 kadrų per sekundę.

4. Vaizdų apdorojimo sistemų sudarymas ir bandymai

Šiame skyriuje bus aprašomi vaizdų apdorojimo algoritmų tyrimai, pateikiamos simuliacijos rezultatai ir jų apibendrinimas. Bus tiriami tokie algoritmai, kaip kraštų aptikimas (Canny ir Sobel), RIR filtravimas, taip pat dviejų vaizdų sujungimas.

4.1 Canny Kraštų aptikimo algoritmo tyrimas

Pirmoji tirta sistema buvo, panaudojant Canny kraštų aptikimo algoritmą. Šis algoritmas yra įgyvendintas „Altera“ universitetinės programos pakete, kuris yra nemokamas ir laisvai prieinamas. Kadangi reikėjo apdoroti video medžiagą realiu laiku, todėl buvo pasitelkta D5M vaizdo kamera ir iš jos gautą vaizdą reikėjo paruošti apdorojimui. Tam tikslui buvo sudaryta sistema, kuri nuskaito vaizdą ir po tam tikrų veiksmų jį išveda į ekraną su išryškintais taškais. Šios sistemos struktūrinė schema pavaizduota 4.1 pav.



4.1 pav. Sistemos, skirtos nuskaityti vaizdą ir atlikti kraštų aptikimą struktūrinė schema

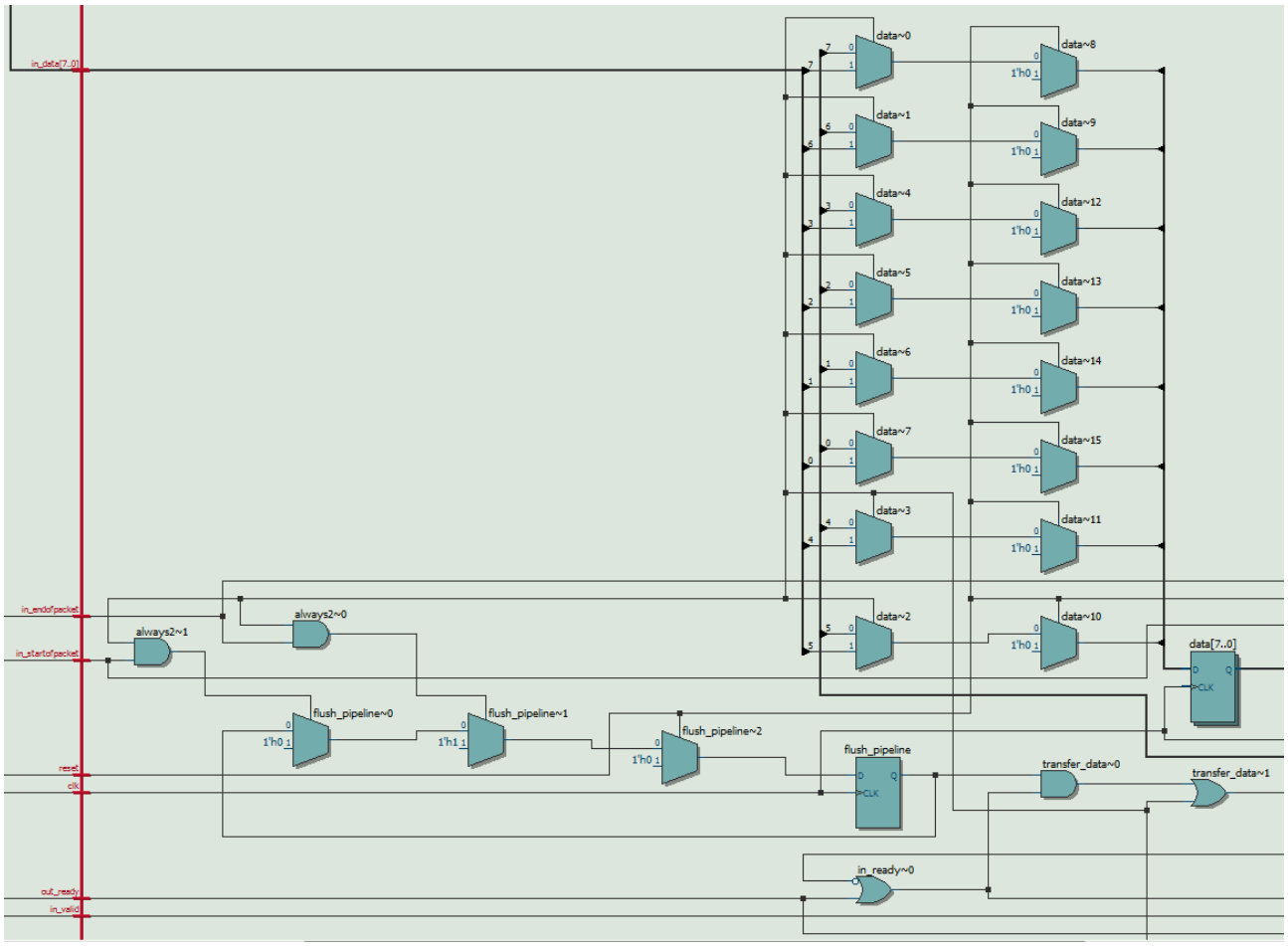
4.1 pav. pavaizduoti moduliai atlieka tokias operacijas:

- Kameros konfigūracijos modulis. Šis modulis skirtas nustatyti jungiamos kameros tipą ir vaizdo raišką. Duomenys perduodami per I2C sąsaja. Nustatoma maksimali 2592x1944 vaizdo raiška.

- Vaizdo įėjimo dekoderis. Šis modulis skirtas nukaityti duomenis iš kameros ir perduoti į LPLM Avalon[25] magistralę, kuri skirta perduoti srautinius duomenis. Duomenų srauto formatas yra 8 bitų Bajerio.
- Bajerio dekoderis. Modulis skirtas duomenis, užkoduotus Bajerio formatu paversti į 24 bitų RGB spalvotą vaizdą. To pasėkoje sumažėja vaizdo raiška iki 1296x972.
- Vaizdo mastelio keitimo modulis. Šis modulis skirtas sumažinti vaizdą kiek galima arčiau VGA raiškos. Atliekamas vaizdo sutraukimas per pusę ties aukščiu ir ties pločiu. Sutraukimas atliekamas panaikinant kas antrą kadro eilutę ir stulpelį. Taigi raiška yra pakeičiama į 648x486.
- Vaizdo kadro apdorojimo modulis. Skirtas „apkarpyti vaizdo kadrus. Kadangi reikia gauti 640x480 vaizdo raišką, todėl nuo kraštų yra pašalinami stulpeliai ir eilutės. Šiuo atveju nuo kiekvieno krašto yra pašalinama po 4 stulpelius ir po 3 eilutes.
- Spalvinės gamos keitimo modulis. Modulis skirtas pakeisti vaizdo formatą iš 24 bitų į 8 bitų nespalvotą vaizdą. Toks formatas reikalingas kraštų aptikimo moduliui.
- Kraštų aptikimo modulis. Šis modulis atlieka kraštų aptikimą pagal Canny algoritmą.
- Tiesioginės kreipties į atmintį valdiklis. Šis modulis talpina vaizdo kadrus į atmintį ir pasiima iš jos. Jis veikia kaip valdantysis įrenginys ir siunčia komandas atminties valdikliui.
- SDRAM atminties valdiklis. Šis modulis komunikuoja su išorinę SDRAM atmintimi, kuri yra DE0-nano vaizdo plokštėje. Šis modulis yra nustatomas kaip valdomasis įrenginys, kuris gaunas komandas iš tiesioginės kreipties į atmintį modulio. Modulis yra eksportuojamas į išorę, kad būtų atliekamas fizinis sujungimas su išorine atmintimi.
- Tiesioginės kreipties vaizdo pikselių buferis. Šis modulis naudojamas laikinai laikyti vaizdo kadrus, kurie buvo nuskaityti iš išorinės SDRAM atminties.
- Vaizdo vertimas į RGB formatą. Šis modulis skirtas paversti vaizdą iš nespalvoto į 30 bitų spalvotą RGB vaizdą, nes tokio formato reikia VGA valdikliui.
- Vaizdo pikselių buferis. Šis buferis padeda duomenis perduoti kai skiriasi taktiniai dažniai. VGA valdikliui reikia 25MHz taktavimo, tuo metu vaizdo srauto duomenys dirba 50MHz taktiniu dažniu. Duomenys yra talpinami į FIFO (Pirmas įeina, pirmas išeina) atmintį.
- VGA valdiklis. Modulis skirtas Nuskaityti srautinius duomenis ir perduoti į skaitmeninį analoginį keitiklį, tolesniam išvedimui į ekraną.

- Taktinio dažnio skirstymo modulis. Jis skirtas Padalinti taktinį dažnį pagal tai kokio dažnio reikia kiekvienam įrenginiui. VGA valdikliui reikalingas 25MHz taktinis dažnis, avalon srautinei magistralei reikalingas 50MHz dažnis, išoriniai atminčiai reikalingas 100MHz taktinis dažnis.

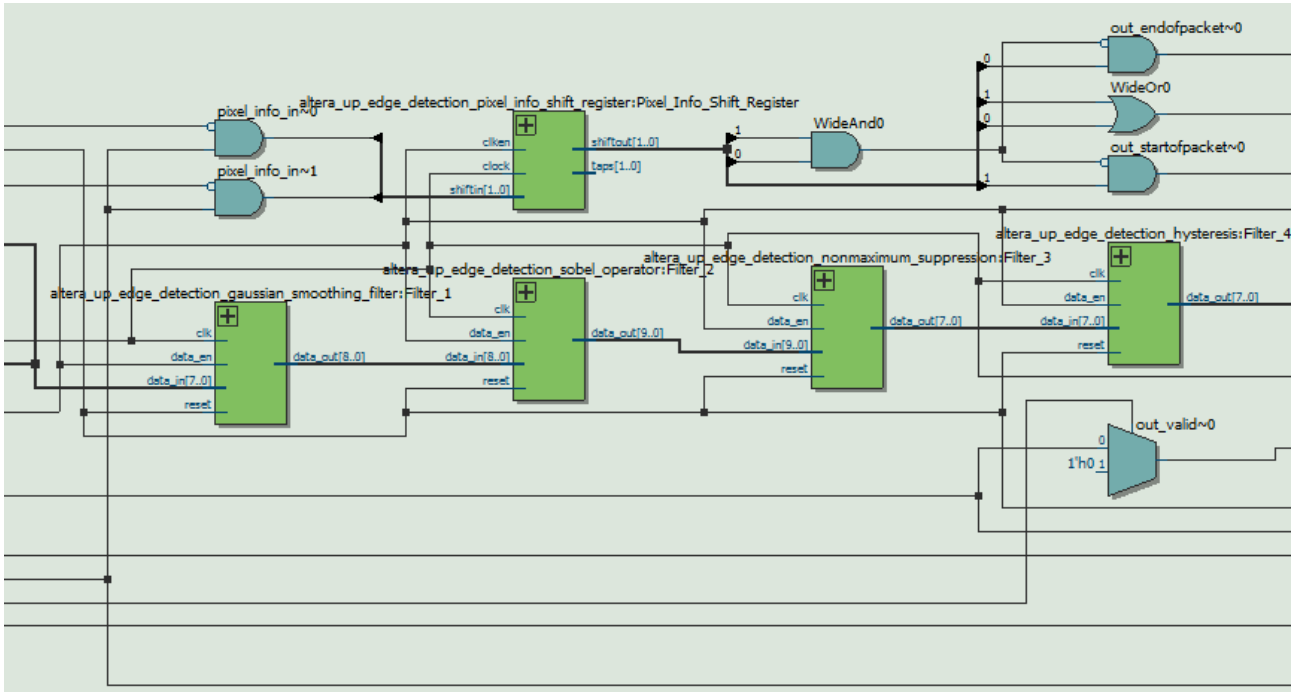
Kraštų aptikimo modulio įėjimas yra pavaizduotas 4.2 pav.



4.2 pav. Kraštų aptikimo modulio įėjimo dalis

4.2 pav. matyti, kad į modulį paduodami ne tik duomenys, bet ir kiti signalai, tokie kaip:

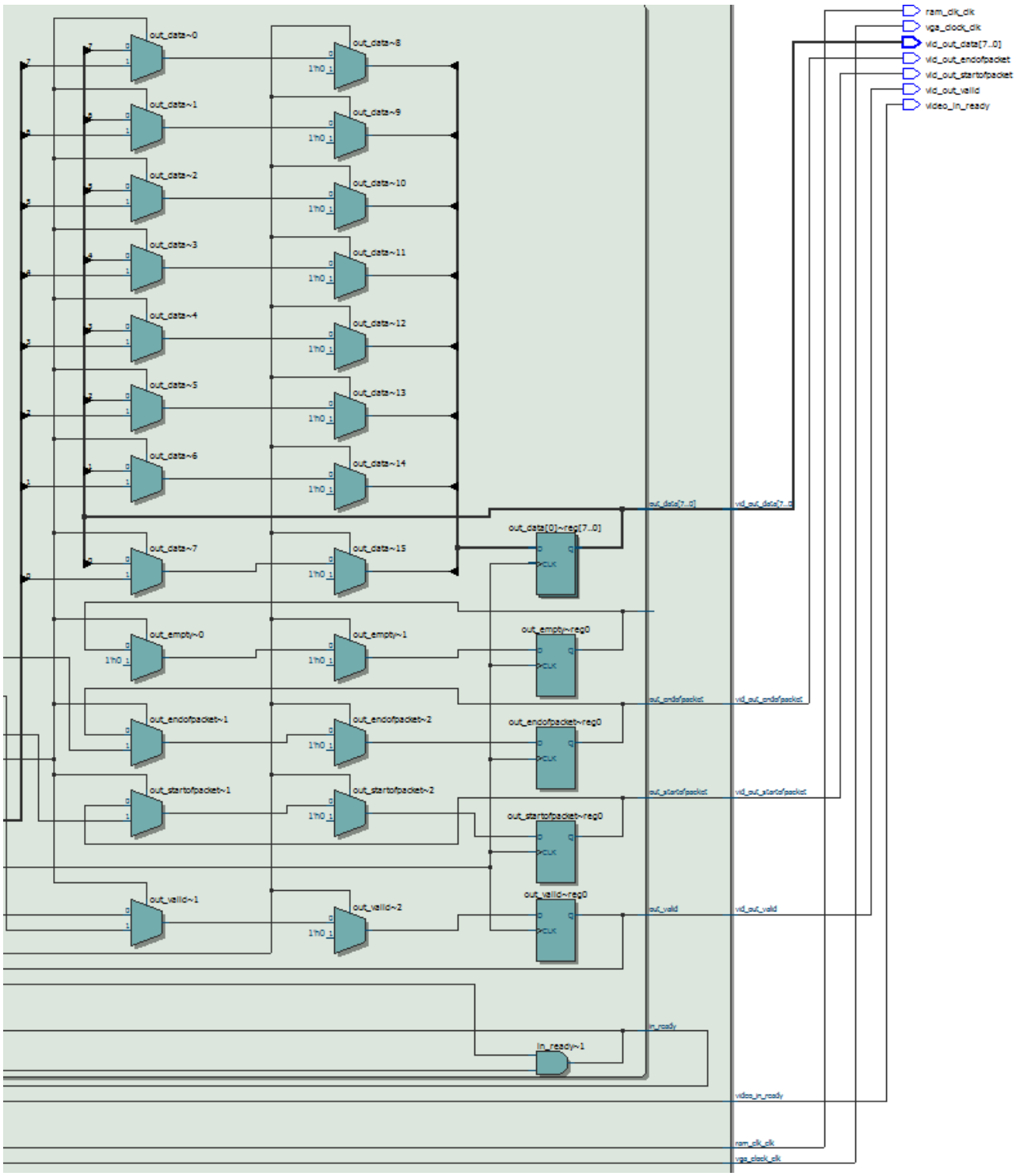
- „startofpacket“, kuris indikuoja kada turi būti pradėtas siusti duomenų paketas, šiuo atveju vaizdo kadro pradžia.
- „endofpacket“, kuris indikuoja vaizdo kadro pabaigą
- „reset“, kuris atlieka modulio atstatymą su aukštu loginiu lygiu
- „clk“, kuris yra sistemos taktavimo dažnis
- „out_ready“, kuris indikuoja, kad modulis yra „pasiruošęs“ siųsti duomenis po apdorojimo
- „in_valid“, kuris parodo, kad priimami duomenys yra tinkami apdorojimui, aktyvus signalas yra su aukštu loginių lygiu.



4.3 pav. Kraštų aptikimo modulio vykdančioji dalis

4.3 pav. pavaizduotoje schemoje matyti, iš kokių pagrindinių dalių yra sudaryta vykdančioji kraštų aptikimo dalis.

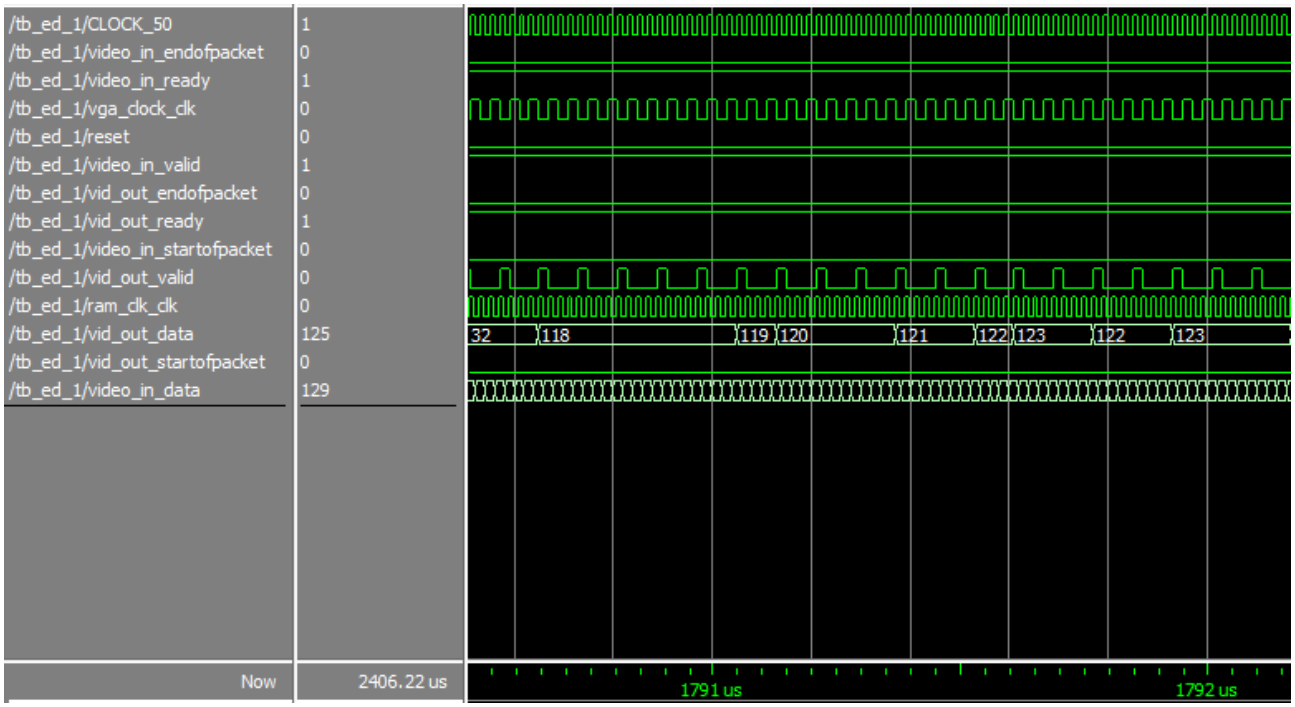
- Gauso glodinimo filtras – skirtas sumažinti triukšmus vaizde;
- sobel operatorius skirtas apskaičiuoti gradientus;
- ne-maksimalus slopinimas skirtas suploninti kraštus, randant gradiento kryptis;
- histerezės modulis apskaičiuoja kurie gradientai yra kraštai, taip užtikrinant, kad kraštai nebūtų nutrukę;
- vaizdo taškų postūmio registras reguliuoja kada bus išsiųstas ir kada stabdomas vaizdo kadro priėmimas pagal signalų „startofpacket“ ir „endofpacket“ reikšmes.



4.4 pav. Kraštų aptikimo modulio išėjimo dalis

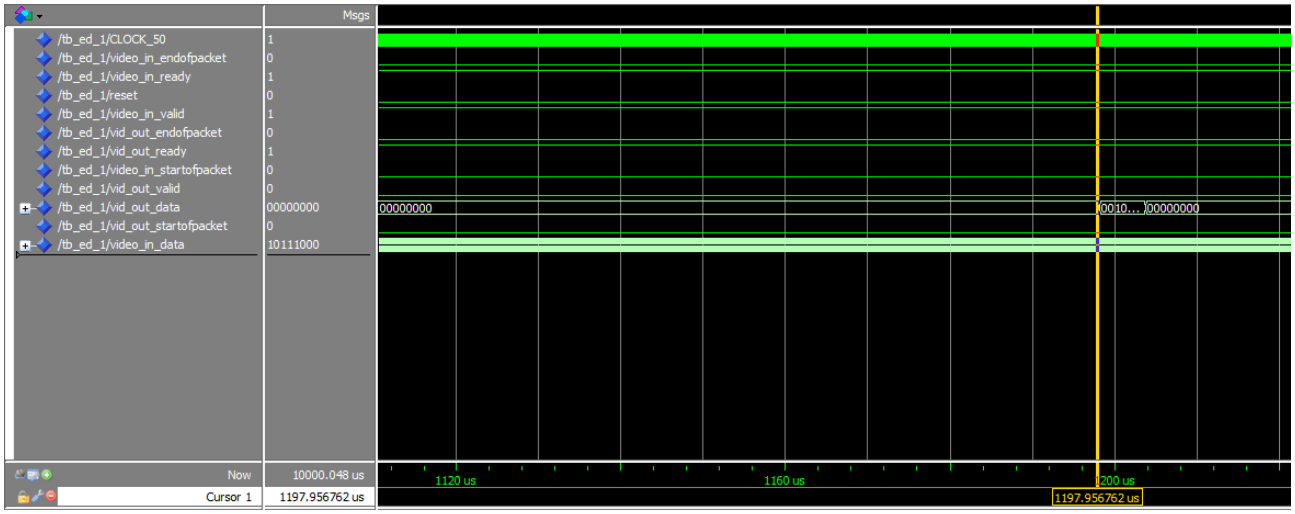
4.4 pav. pavaizduotoje schemoje matyti kas sudaro modulio išėjimus. D trigeriai panaudojami kaip buferiai duomenims. Multiplekseriai panaudoti, kad nesant duomenims būtų perduodamos 0 reikšmės, nes kitaip sistema gali pereiti į neapibrėžtumą.

Buvo atlikta sistemos simuliacija, kuri pavaizduota 4.5 pav.



4.5 pav. Kraštų aptikimo Canny algoritmu sistemos simuliacija

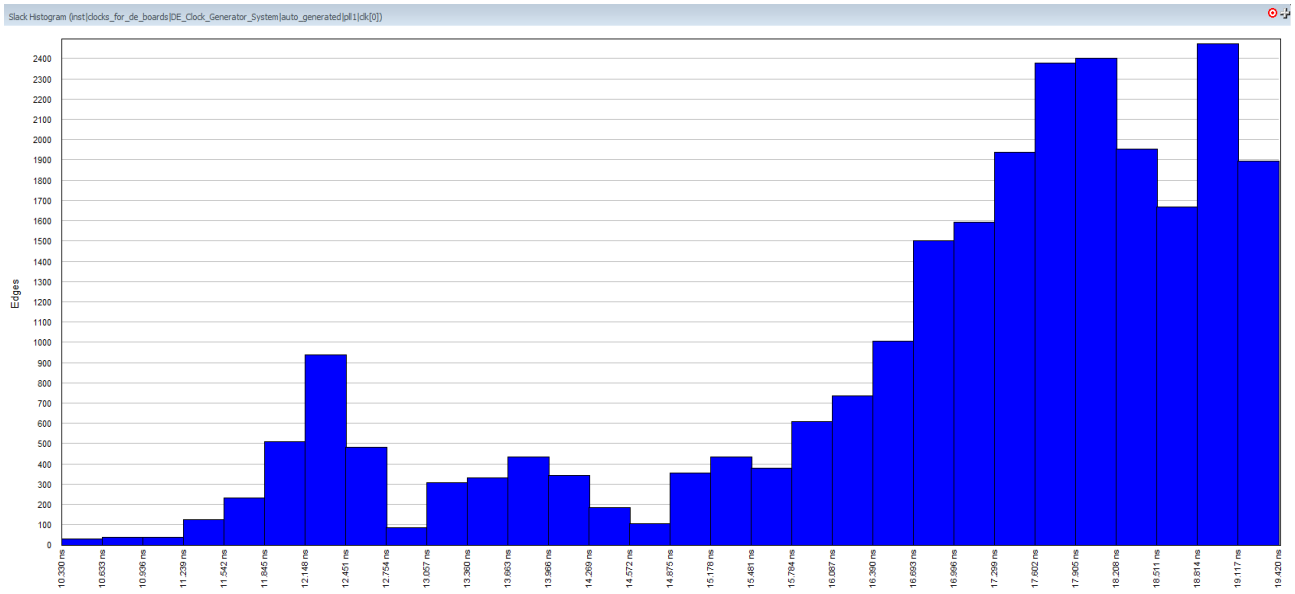
Vykdant simuliaciją duomenys įėjime buvo paduodami kas 1 taktinio dažnio periodą – 20 ns. Iš simuliacijos matyti, kad išėjime gaunamos reikšmės vaizduoja kraštus. Iš reikšmių galima spręsti, kad aptinkami kraštai yra pilki. Kadangi pagrindinis simuliacijos tikslas buvo išsiaiškinti sistemos vėlinimą, todėl yra pateikiamas 4.6 pav. pavaizduotas grafikas.



4.6 pav. Kraštų aptikimo Canny algoritmu sistemos vėlinimas

Kaip matyti iš 4.6 pav. pavaizduoto grafiko, yra parodomas momentas nuo duomenų padavimo iki apdorotų duomenų gavimo. Šios sistemos vėlinimas yra 1,197 ms.

Pasinaudojant „TimeQuest“ laiko analizatoriumi buvo nustatytas sistemos stabilumas kiekviename jos mazge. Sistemos stabilumo histograma pavaizduota 4.7 pav.



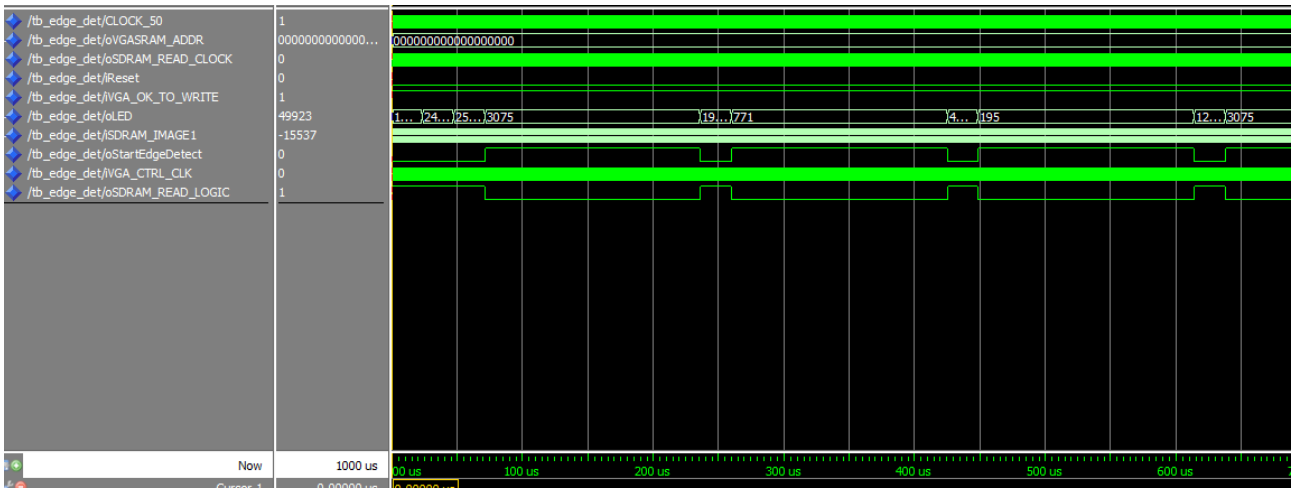
4.7 pav. Kraštų aptikimo Canny algoritmu sistemos stabilumo įvertinimo histograma

X pav. pavaizduotoje histogramoje ordinačių ašyje atidėti visi sistemos mazgai, abscisių ašyje – laiko skirtumas tarp tikėtino signalo atkeliavimo laiko tarpo ir laiko tarpo per kurį iš tikrųjų atkeliavo signalas. Šis skirtumas turi būti teigiamas ir geriau kuo didesnis. Kaip matyti, visi sistemos mazgai yra stabilūs ir iš to seka, kad pati sistema yra stabili.

4.2 Sobel algoritmo tyrimas

Šio algoritmo įgyvendinimo schema yra tokia pati, kokia pavaizduota 4.1 pav. Tik pats algoritmas yra sudarytas vien iš sobel operacijos bloko. Šis modulis yra aprašytas Verilog kalba, kurį galima rasti darbo prieduose.

Buvo atlikta sistemos simuliacija, kuri pateikta 4.8 pav.

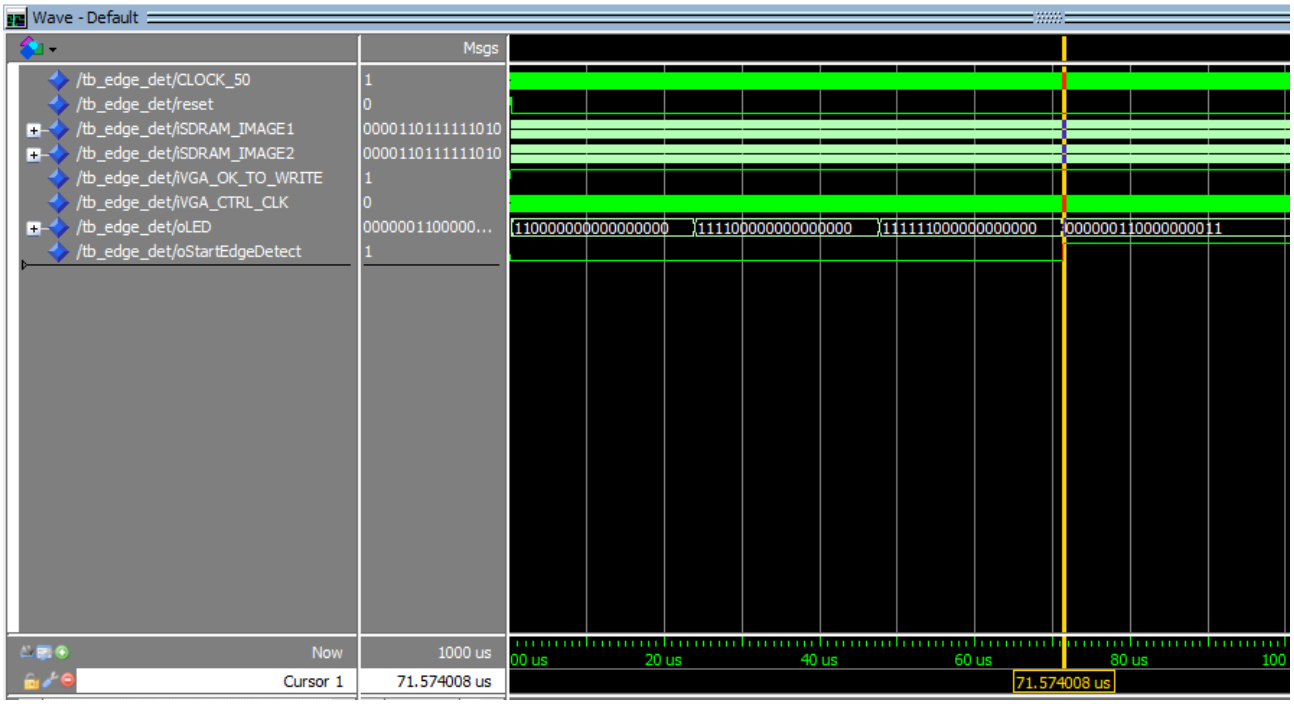


4.8 pav. Sistemos su sobel kraštų aptikimo algoritmu simuliacijos grafikas

4.8 pav. matyti, kad įėjime duomenys yra 16 bitų (signalas „isdram_IMAGE1“). Signalas „oStartEdgeDetect“ sindikuoja kada yra pradedama ir užbaigiama kraštų aptikimo operacija.

Išėjimo duomenis rodo signalas „oLED“. Iš duomenų matyt, kad kraštai aptinkami ne ties maksimaliomis reikšmėmis, taip yra dėl to, kad algoritmas nevykdo nei ne-maksimalaus suspaudimo, nei histerezės radimo operacijų.

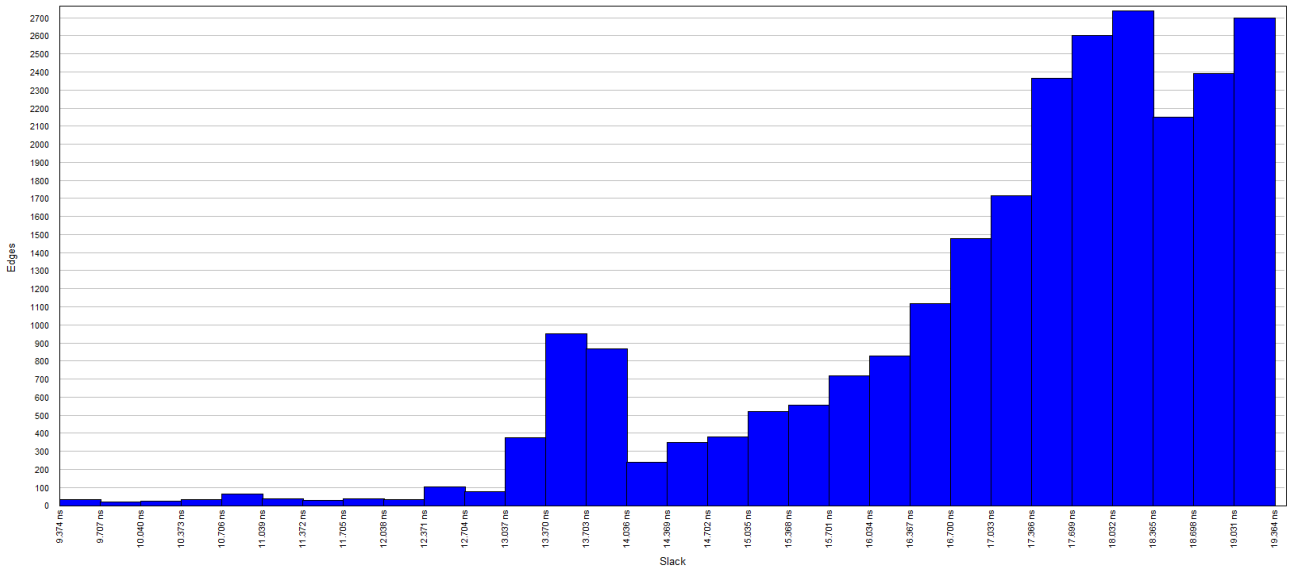
Kitame grafike (4.9 pav.) matyti sistemos vėlinimo laikas.



4.9 pav. Sistemos su sobel kraštų aptikimo algoritmu vėlinimo laikas

Grafike matyti, kad nuo duomenų pateikimo iki kraštų aptikimo operacijos pradžios yra 71,57 µs vėlinimas.

Sistemos stabilumo įvertinimo histograma pavaizduota 4.10 pav.



4.10 pav. Sistemos su sobel algoritmu stabilumo įvertinimo histograma

Iš 4.10 pav. pavaizduotos histogramos matyti, kad sistema yra stabili kiekviename mazge, iš to seka, kad ir pati sistema yra stabili.

4.3 Sistemos su RIR filtravimu tyrimas

Sistemai sudaryti buvo panaudota tapati struktūrinė schema kuri yra pavaizduota 4.1 pav. Sistema buvo sudaryta pasinaudojus VHDL kalba. Toliau pateikiamas programos kodas.

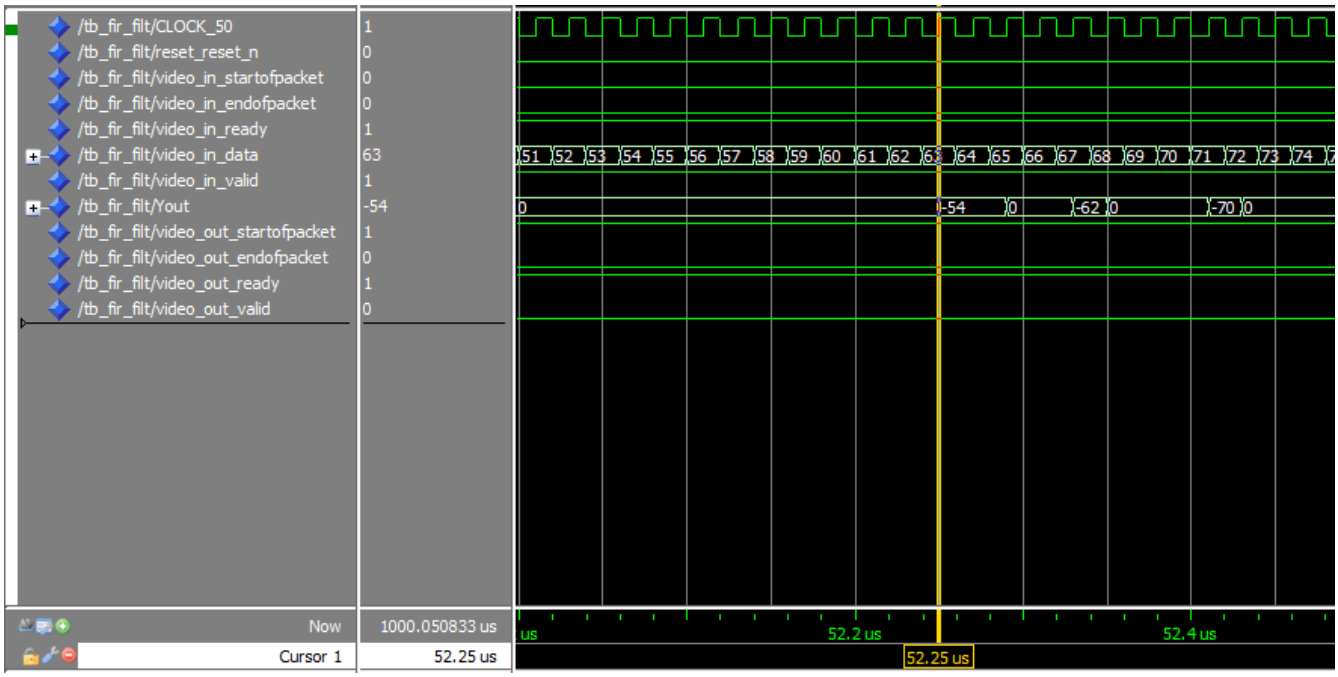
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity fir is
port(   CLOCK_50 : in std_logic; --taktinis daznis
        Xin : in signed(7 downto 0); --iejimas
        Yout : out signed(15 downto 0);  --isejimas
        Q : out signed(15 downto 0);     --isejimas prijungtas prie sudeties elemento
        D :in  signed(15 downto 0)     -- isejimas is MCM
        );
end fir;
architecture Behavioral of fir is
signal H0,H1,H2,H3 : signed(7 downto 0) := (others => '0');
signal MCM0,MCM1,MCM2,MCM3,add_out1,add_out2,add_out3 : signed(15 downto 0) := (others => '0');
signal Q1,Q2,Q3 : signed(15 downto 0) := (others => '0');
signal qt : signed(15 downto 0) := (others => '0');
begin
--filtro koef
--H = [-2 -1 3 4].
 H0 <= to_signed(-2,8);
 H1 <= to_signed(-1,8);
 H2 <= to_signed(3,8);
 H3 <= to_signed(4,8);

--konstantu sandaugos
 MCM3 <= H3*Xin;
 MCM2 <= H2*Xin;
 MCM1 <= H1*Xin;
 MCM0 <= H0*Xin;
--sudejimas
 add_out1 <= Q1 + MCM2;
 add_out2 <= Q2 + MCM1;
 add_out3 <= Q3 + MCM0;
 Q <= qt;
--velinimas
dff1 : DFF port map(Q1,Clk,MCM3);
 dff2 : DFF port map(Q2,Clk,add_out1);
 dff3 : DFF port map(Q3,Clk,add_out2);
--isejimas ties kiekvieno taktikio danio ciklu
process(CLOCK_50)
begin
    if(rising_edge(CLOCK_50)) then
        qt <= D;
        Yout <= add_out3;
    end if;
end process;
end Behavioral;
```

Aukščiau esančiame VHDL kalba parašytame kode matyti, kad į įėjimą yra paduodami 8 bitų duomenys. Toliau yra sudaromi filtro koeficientai. Sasukos operacija atliekama filtro koeficientus dauginant iš įėjimo, toliau atliekama sudėtis ir vėlinimo operacija. Išėjimas formuojamas su kiekvieno taktinio dažnio kylančiu frontu.

Tokios sistemos simuliacija pateikta 4.11 pav.

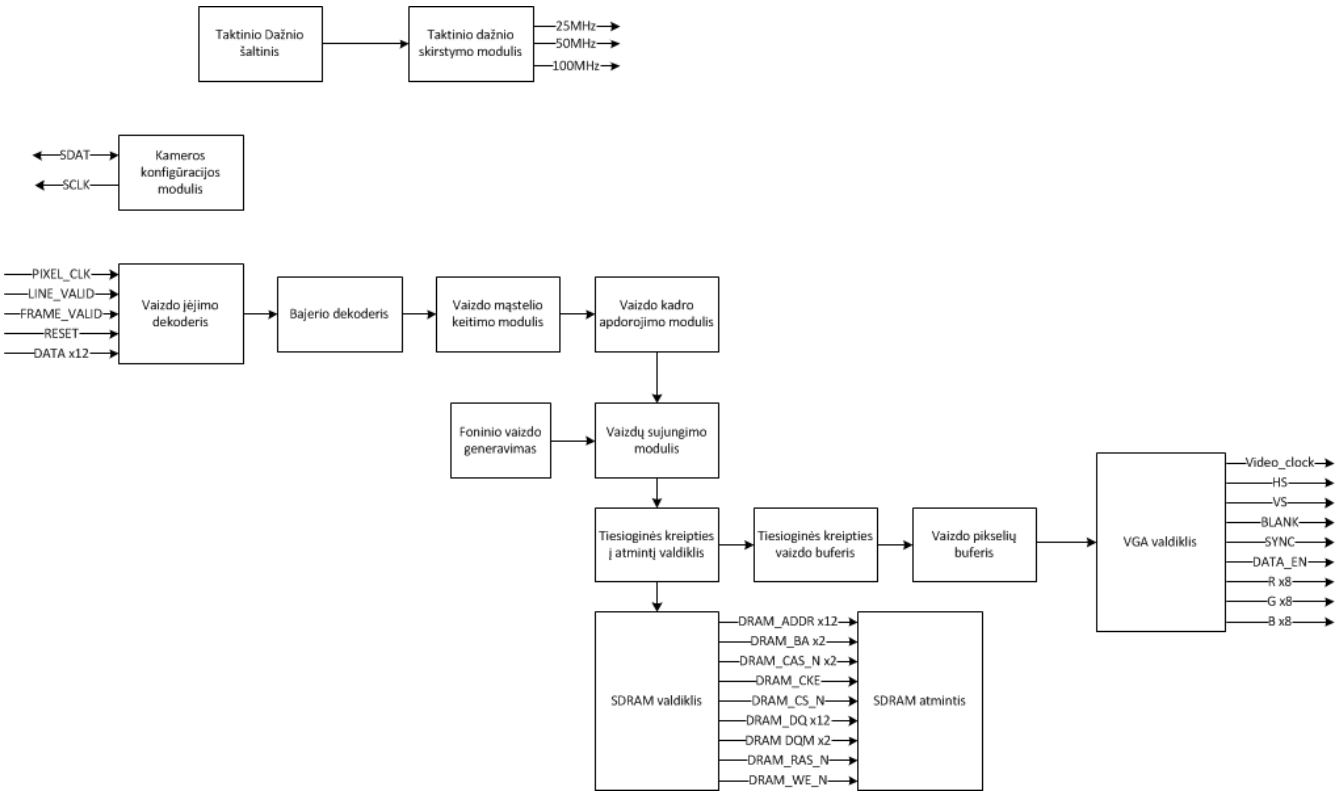


4.11 pav. Sistemos su RIR filtravimo algoritmu simuliacija

Iš simuliacijos matyti išėjimo koeficientai. Taip pat yra parodomas sistemos vėlinimas, kuris yra 52,25 µs.

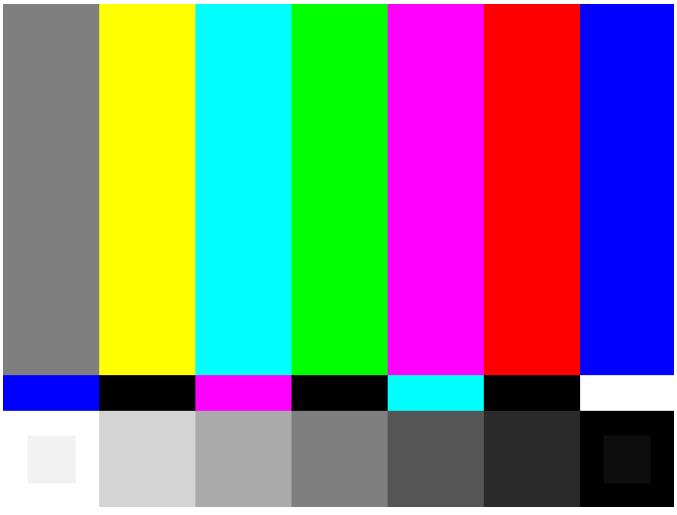
4.4 Dviejų vaizdų sudėjimo sistemos tyrimas

Tiriant šią sistemą, reikėjo šiek tiek modifikuoti struktūrinę schemą, kuri pavaizduota 4.12 pav.



4.12 pav. Vaizdų sudėjimo sistemos struktūrinė schema

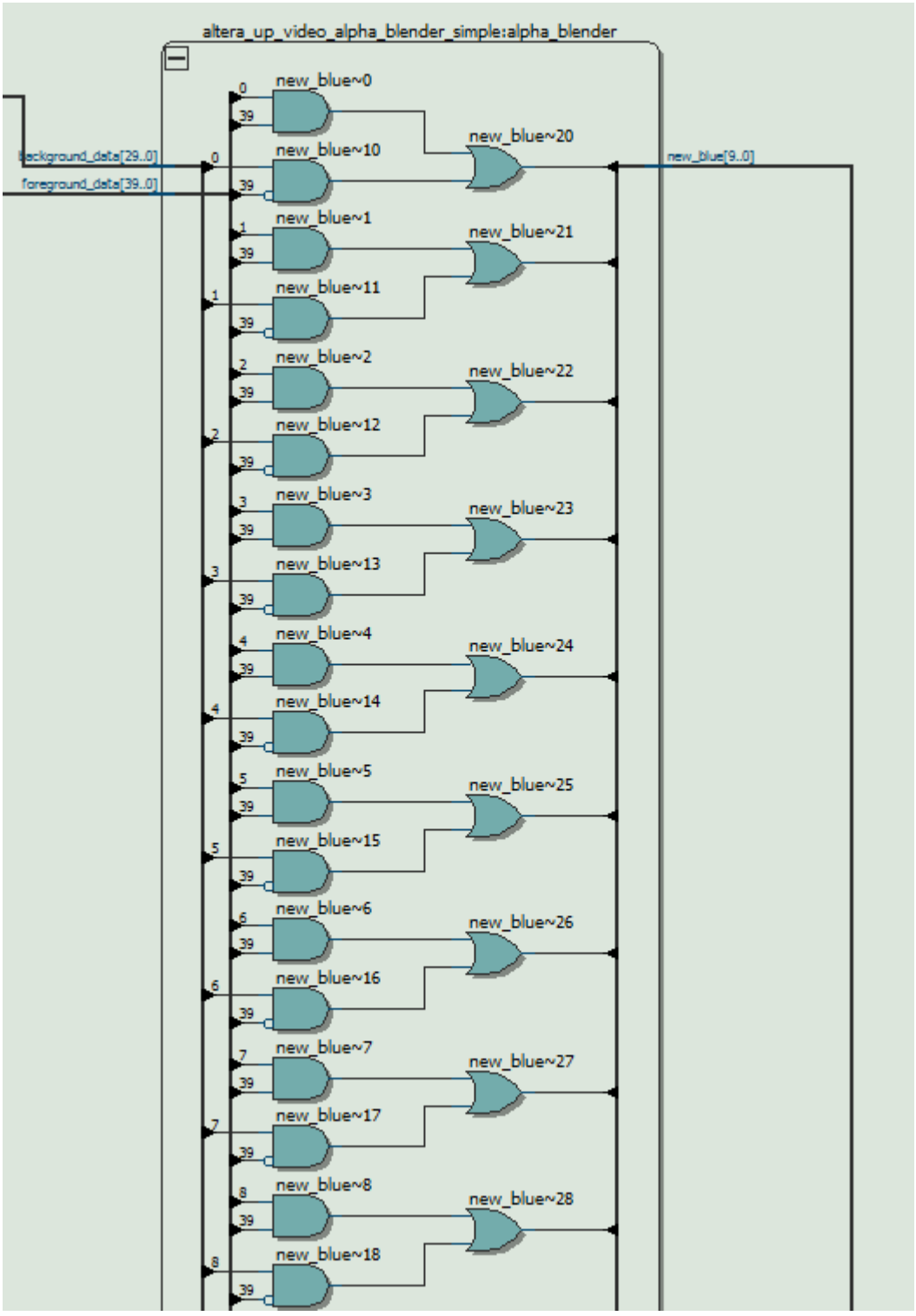
4.12 pav. matyti, kad nebeliko spalvinės gamos keitimo modulio lyginant su 4.1 pav. pavaizduota schema. Toks sprendimas buvo padarytas todėl, kad vaizdų sujungimo modulis priima ne spalvotą, o RGB vaizdą. Tiriamoje sistemoje yra sujungiami du vaizdai: foninis ir aktyvusis vaizdas. Foninį vaizdą generuoja specialus foninio vaizdo generatorius. Generuojamas vaizdas pavaizduotas 4.13 pav.



4.13 pav. Generuojamo fono vaizdas

Foninis vaizdas ir aktyvusis vaizdas, kuris gaunamas iš vaizdo kameros yra sudedami alfa smulkinimo maišytuve (angl. Alpha Blending mixer)

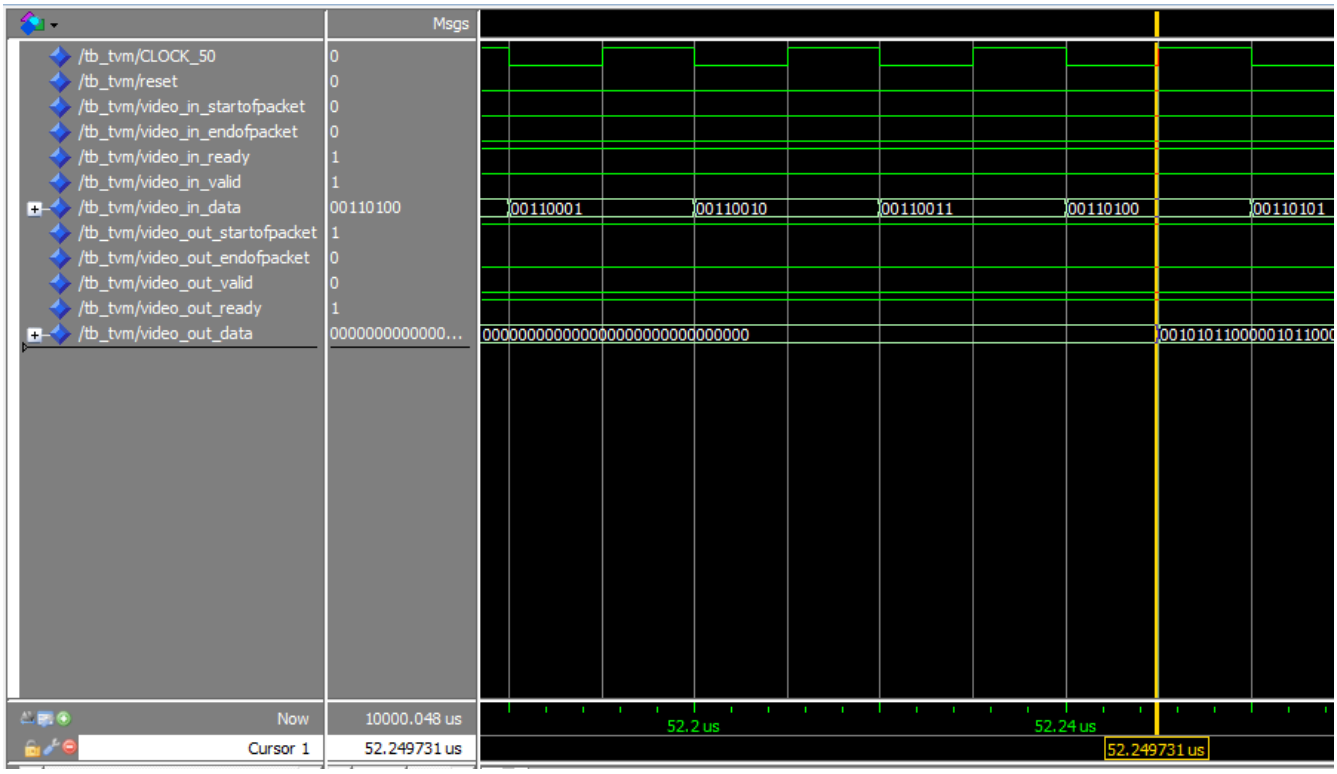
Dalinė tokio modulio funkcinė schema pavaizduota 4.14 pav.



4.14 pav. Vaizdų sujungimo modulio funkcinės schemos dalis

Iš 4.14 pav. matosi, kad yra panaudojami IR ir ARBA elementai, t. y. atliekama loginė daugyba ir loginė sudėtis su dviem vaizdais. Sistema sudaryta taip, kad nesant kažkokios spalvos, pavyzdžiui mėlynos, įėjimas aktyviam vaizde yra invertuojamas. Kadangi spalvos nėra, yra paduodami loginiai nuliai, kurie yra invertuojami ir atliekama loginė daugyba su fonu. Iš to seka, kad išėjime gaunamas vien tik fonas. Balto aktyvaus vaizdo atveju – fonas yra užgožiamas.

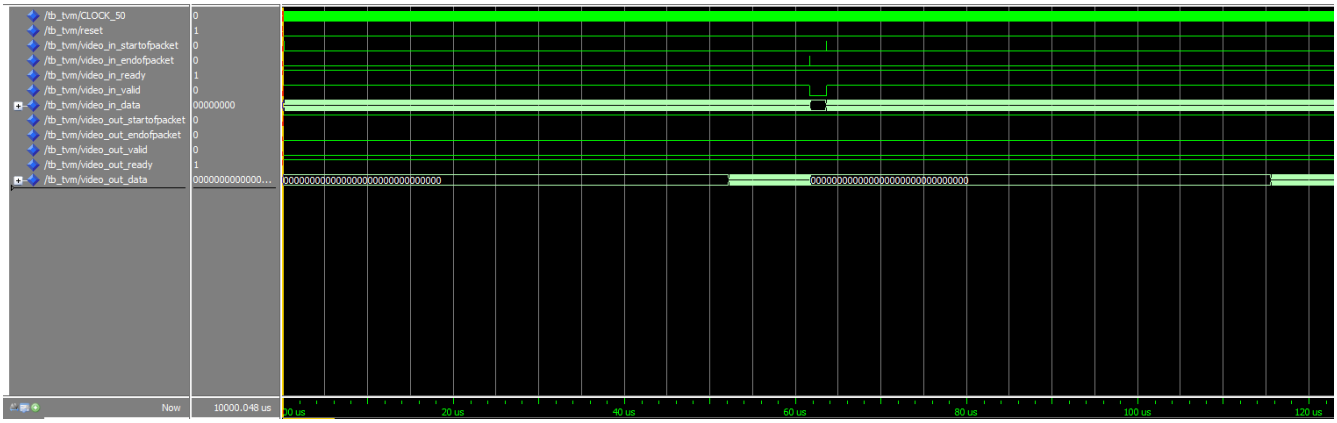
Buvo atlikta sistemos simuliacija, kuri pavaizduota 4.15 pav.



4.15 pav. Dviejų vaizdų sujungimo sistemos simuliacija

Šioje simuliacijoje matyti sistemos vėlinimo laikas, kuris yra 52,24 µs.

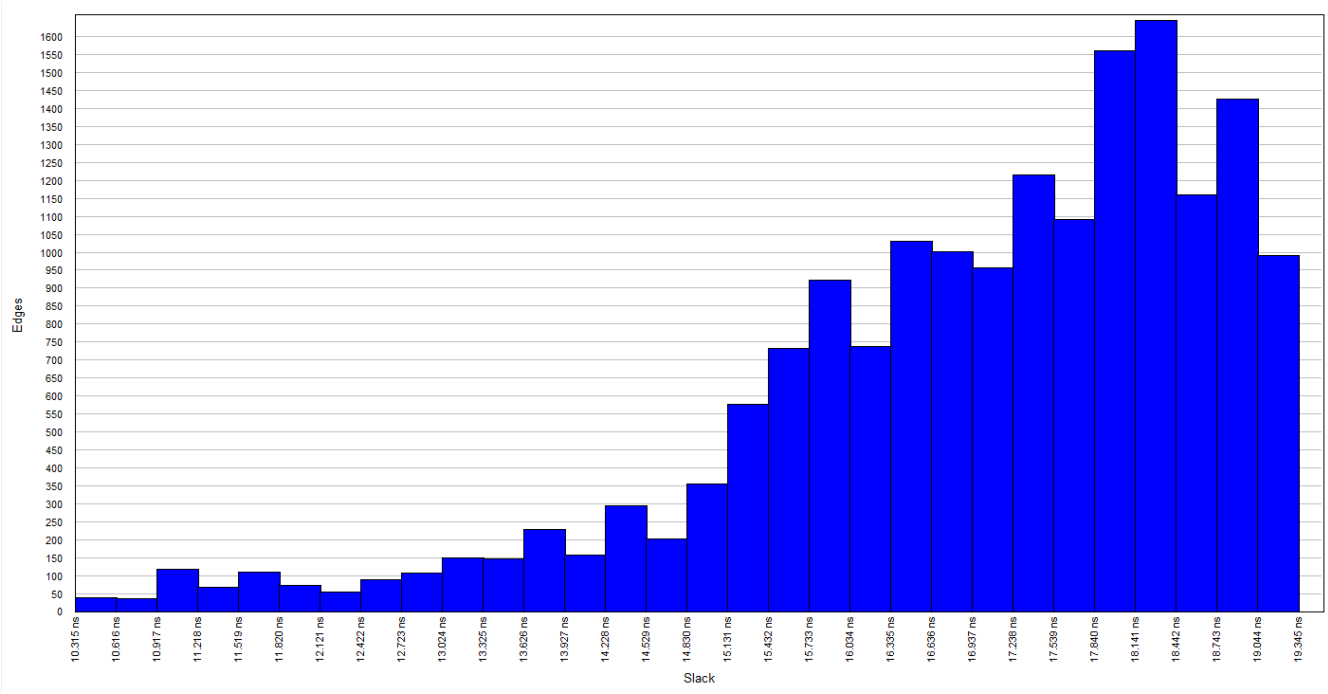
Bendras simuliacijos vaizdas pavaizduotas 4.16 pav.



4.16 pav. Dviejų vaizdų sistemos simuliacija. Bendras vaizdas

Iš 4.16 pav. pavaizduoto grafiko matyti, kada prasideda kiekvienas kadras ir koks matosi vėlinimas.

Sistemos stabilumo įvertinimo histograma pavaizduota 4.17 pav.



4.17 pav. Dviejų vaizdų sujungimo sistemos stabilumo histograma

Iš 4.17 pav. pavaizduotos histogramos matyti, kad sistema yra stabili visuose mazguose. Iš to seka, kad ir pati sistema yra stabili.

5 Bandymų rezultatai ir jų aptarimas

Atliekant tyrimus buvo atsižvelgiama į įvairius aspektus, tokius kaip: loginių elementų naudojimas, atminties naudojimas, naudojama galia, sistemų efektyvumas, stabilumas, taip pat vėlinimo laikas.

5.1 lentelėje pavaizduoti visų tirtų sistemų rezultatai.

5 lentelė

Resursų išnaudojimas įvairiose vaizdų apdorojimo sistemose

Resursas	Panaudota				Iš viso
	Kraštų aptikimas Canny algoritmu	Kraštų aptikimas sobel algoritmu	RIR filtravimas	Dviejų vaizdų sudėjimas	
Loginiai elementai	2888 (13%)	1191 (5%)	426 (2%)	2082 (9%)	22320
Kombinacinės funkcijos	2684 (12%)	1108 (5%)	387 (2%)	1818 (8%)	22320
Dedikuoti loginiai registrai	2203 (10%)	511 (2 %)	277 (1%)	1448 (6%)	22320
Visi registrai	2271	511	277	1514	—
Išvadai	90 (58%)	109 (71%)	67 (44%)	90 (58%)	154
Atminties bitai	82568 (14%)	19200 (3%)	20720 (3%)	28154 (5%)	608256
9 bitų dauginimo elementai	6 (5%)	0	0	0	132
PLL	1 (25%)	1 (25%)	1 (25%)	1 (25%)	4
Galia	192,71 mW	133,4 mW	148,75 mW	167,49 mW	—
Efektyvumas (perdavimai per sekundę)	5,768 MT/s	4,6 MT/s	6,001 MT/s	5,279 MT/s	—
Vykdymo laikas	1,19ms	71,57 µs	52,25 µs	52,24 µs	—

5 lentelėje matyti kiek kokia sistema naudoja kokį resursą ir kiek dar laisvos vietos lieka Cyclone IV LPLM luste.

Apibendrinant rezultatus galima teigti, kad sistemas galima tobulinti, nes yra nemažai laisvos vietos luste. Taip pat vykdymo laikai yra labai geri realaus laiko sistemoms, nes žmogaus akis realų vaizdą suprantą esant jau 24 kadrams per sekundę. Galios naudoja taip pat nedaug, pati galingiausia tirta sistema vieną kilovatvalandę energijos sunaudotų per 5189,1 valandų arba apytiksliai per 7 mėnesius.

Išvados

1. Tirtose sistemose buvo atsižvelgiama į įvairius resursus. Buvo tiriamas vaizdas, gautas iš skaitmeninės 5MP kameros ir paverstas į 640x480 formatą.
2. Kraštų aptikimas buvo vykdomas dviem būdais: universitetinės programos moduliu, kuris aprašytas Canny algoritmu ir sobel algoritmu. Sistema su sobel algoritmu naudoja 41% mažiau loginių elementų, galios naudoja 59,31 mW mažiau, vykdymo laikas yra ženkliai greitesnis, net 16,62 karto, bet sistemos efektyvumas yra 1,25 karto mažesnis.
3. Efektyviausia iš tirtų sistemų yra sistema su RIR filtravimu, kurios efektyvumas 6,001 MT/s
4. Visos tirtos sistemos yra stabilios visuose mazguose.
5. Mažiausias vykdymo laikas užfiksuotas sistemoje su dviem sudedamais vaizdais.
6. Pagal užimamą vietą Cyclone IV LPLM įrenginyje galima teigti, kad lustas turi plačias galimybes sudėtingesnei sistemai, tik reikia atsižvelgti į išvadų kiekį ir vykdymo laiką. Taip pat iš rezultatų matyti, kad sudėtingesnei sistemai mažėja ir jos efektyvumas.

Literatūra

1. Athanas P.M., Abbott A. L. Real-Time Image Processing on a Custom Computing Platform [interaktyvus]. USA, Virginia. Virginia Polytechnic Institute and State University, 1995 [žiūrėta 2014-05-30]. Prieiga per internetą: < https://www.doc.ic.ac.uk/~wl/teachlocal/cuscomp/papers/Athanas95a.pdf >
2. Sorel Y. Real-Time Embeded Image Processing Applications Using The A3 Methodology. France, INRIA, Domaine de Voluceau, Rosquencourt.
3. Altera [interaktyvus], [žiūrėta 2014-05-30]. Prieiga per internetą: http://www.altera.com/
4. Cyclone IV FPGA Device Family Overview [interaktyvus], Altera Corporation, 2014, [žiūrėta 2014-05-30]. Prieiga per internetą: http://www.altera.com/literature/hb/cyclone-iv/cyiv-51001.pdf?GSA_pos=1&WT.oss_r=1&WT.oss=cyclone%20IV
5. Maini R., Dr. Aggarwal H., Study and Comparison of Various Image Edge Detection Techniques [interaktyvus]. Patiala, India, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.math.tau.ac.il/~turkel/notes/Maini.pdf >
6. Gao W., Yang L., Zhang X., Liu H. An improved Sobel Edge Detection. China, Beijing, 2010.
7. Lavry D. Understanding FIR Filters – An Intuitive approach [interaktyvus]. Lavry Engineering, 1997, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://lavryengineering.com/pdfs/lavry-understanding-fir-filters.pdf >
8. Bailey D. G. Design for embedded image processing on FPGAs. New Zealand: Massey University, John Wiley & Sons. 2011, p.482. ISBN: 978-0-470-82849-6.
9. Shoup R. G. Parameterized convolution filtering in a field programmable gate array [interaktyvus]. Oxford, United Kingdom: Abingdon EE&CS Books, 1994, p.274-280 [žiūrėta 2012-06-05]. Prieiga per internetą: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.1074&rep=rep1&type=pdf>.
10. Joy K. I. Alpha Blending for Virtualization Applications. [interaktyvus] USA, California, University of California, Davis, Departament of Computer Science, Vizualization and Graphics Research Laboratory, 1999, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://graphics.cs.ucdavis.edu/~joy/ecs277/other-notes/Alpha-Blending.pdf>
11. Saluja S., Agrawal K., Sivasakaran K. K., Bandalmudi L., Ursu C. Real Time Vehicular Camera Vision Acquisition System Using FPGA. [interaktyvus] 2013, [žiūrėta 2013-05-30]. Prieiga per internetą: < http://delphi.com/pdf/techpapers/2013-01-1340.pdf>
12. Madhavan R. Quick Reference for Verilog HDL. [interaktyvus] USA, California, San Jose, ABMIT Design Systems, Inc. 1995, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.stanford.edu/class/ee183/handouts_win2003/VerilogQuickRef.pdf>
13. Chiuchisan I., Cerlinca M., Potorac A., Graur A. Image Enchancement Methods Approach usig Verilog Hardware Description Language. [Interaktyvus] Romania, Suceava, 2012, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.dasconference.ro/cd2012/data/papers/C60.pdf>
14. Bittibssi T. M., Salama G. I., Mehaseb Y. Z., Henawy A. E. Image Enhancement Algorithms using FPGA. [interaktyvus] Egypt, Cairo, Ain Shams University, Departament

of ECE, p. 536-542, ISSN:2249-5789, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.ijcscn.com/Documents/Volumes/vol2issue4/ijcscn2012020417.pdf>

15. Chaithra N. M., Reddy K. V. R. Implementation of Canny Edge Detection Algorithm on FPGA and displaying Image through VGA Interface. [interaktyvus] International Journal of Engineering and Advanced technology, vol 2, p. 243-247, 2013, ISSN: 2249 – 8958, [žiūrėta 2013-05-30]. Prieiga per internetą: < http://www.ijeat.org/attachments/File/v2i6/F2053082613.pdf>
16. Quartus II Design Software. [interaktyvus] Altera Corporation, USA, California, San Jose, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.altera.com/literature/br/br-quartus2-software.pdf>
17. VHDL handbook. [interaktyvus] AB Electronics, Sweden, Lund, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.csee.umbc.edu/portal/help/VHDL/VHDL-Handbook.pdf>
18. Qsys Interconect. [interaktyvus] Altera Corporation, USA, California, San Jose, 2013, [žiūrėta 2014-05-30], Prieiga per internetą: < http://www.altera.com/literature/hb/qts/qsys_interconnect.pdf>
19. The Quartus II TimeQuest Timing Analyzer. [interaktyvus] Altera Corporation, USA, California, San Jose, 2013, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.altera.com/literature/hb/qts/qts_qii53018.pdf>
20. PowerPlay Power Analysis. [Interaktyvus] Altera Corporation, USA, California, San Jose, 2013, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.altera.com/literature/hb/qts/qts_qii53013.pdf>
21. Using ModelSim to Simulate Logic Circuits for Altera FPGA Devices. [Interaktyvus] Altera Corporation, USA, California, San Jose, 2011, [žiūrėta 2014-05-30]. Prieiga per internetą: < ftp://ftp.altera.com/up/pub/Altera_Material/10.1/Tutorials/Using_ModelSim.pdf>
22. Simulating Altera Designs. [Interaktyvus] Altera Corporation, USA, California, San Jose, 2013, [žiūrėta 2013-05-30]. Prieiga per internetą: < http://www.altera.com/literature/hb/qts/qts_qii53025.pdf>
23. TRDB_D5M. [Interaktyvus] Headquarter Hsinchu, Taiwan, Hsinchu city, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=281&FID=09c534c7a8354c5fa0b5fd1e72f3ed96>
24. Jean. R Demosaicing with The Bayer Pattern. [Interaktyvus] USA, University of North Carolina, Departament of Computer Science, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.unc.edu/~rjean/demosaicing/demosaicing.pdf>
25. Avalon Interface. [Interaktyvus] Altera Corporation, USA, California, San Jose, 2014, [žiūrėta 2014-05-30]. Prieiga per internetą: < http://www.altera.com/literature/manual/mnl_avalon_spec.pdf>

Priedai

1 Priedas. Kompaktinis diskas.

Kompaktinio disko turinys

1. Baigiamasis magistro darbas;
2. Tirtų sistemų projektai;