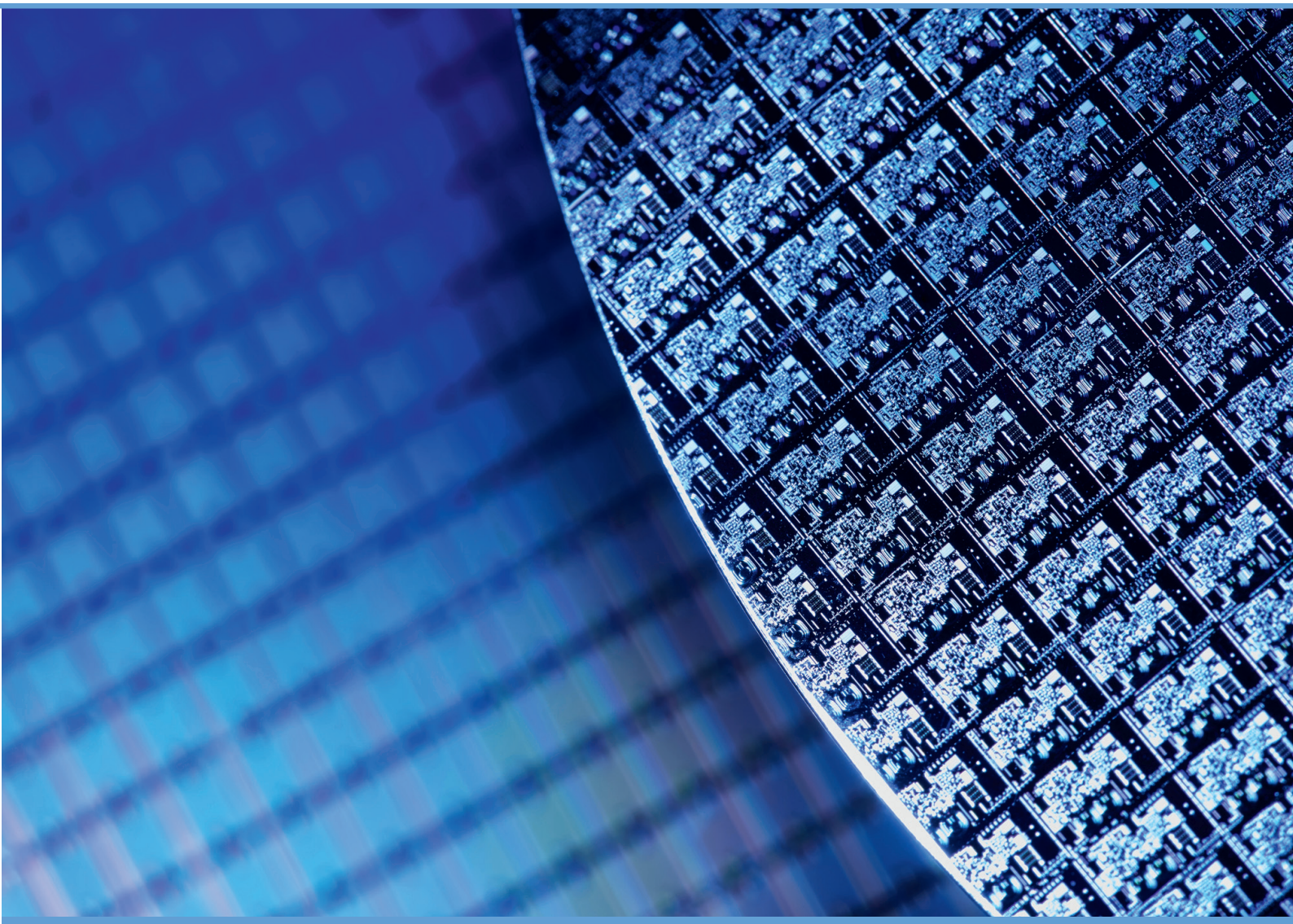
RĪGAS TEHNISKĀ UNIVERSITĀTE

Rihards Novickis

STEREOREDZES ALGORITMU IZPĒTE UN REALIZĀCIJA HETEROGĒNĀ IEGULTĀ SISTĒMĀ

Promocijas darba kopsavilkums



RTU Izdevniecība
Rīga 2022

RĪGAS TEHNISKĀ UNIVERSITĀTE
Elektronikas un telekomunikāciju fakultāte
Radioelektronikas institūts

Rihards Novickis
Doktora studiju programmas ”Elektronika” doktorants

Stereoredzes algoritmu izpēte un realizācija heterogēnā iegultā sistēmā

Promocijas darba kopsavilkums

Zinātniskie vadītāji:
Dr. sc. ing., vadošais pētnieks
Artūrs Āboltiņš

Dr. sc. ing., pētnieks
Rolands Šāvelis

RTU Izdevniecība
Rīga 2022

Novickis R. Stereoredzes algoritmu izpēte un realizācija heterogēnā iegultā sistēmā. Promocijas darba kopsavilkums. – Rīga: RTU Izdevniecība
2022, 46 lpp.

Iespiests saskaņā ar 2021. gada 7. decembra Elektronikas un telekomunikāciju fakultātes promocijas padomes P-08 lēmumu Nr. 6.

https://doi.org/10.7250/9789934227462
ISBN 978-9934-22-746-2 (pdf)

PROMOCIJAS DARBS IZVIRZĪTS INŽENIERZINĀTNES DOKTORA GRĀDA IEGŪŠANAI RĪGAS TEHNISKAJĀ UNIVERSITĀTĒ

Promocijas darbs inženierzinātnes doktora (Ph. D.) grāda iegūšanai tiek publiski aizstāvēts 2022. gada 11. martā plkst. 13 Rīgas Tehniskajā universitātē, Elektronikas un telekomunikāciju fakultātē, Āzenes ielā 12, 201. auditorijā.

OFICIĀLIE RECENZENTI

Dr. sc. ing., asociētais profesors Dmitrijs Pikuļins,
Rīgas Tehniskā universitāte, Latvija

Dr. sc. ing., docents Paolo Meloni,
Kaljāri Universitāte, Itālija

Dr. sc. ing., asociētais profesors *Wassim Hamidouche*,
Rennas Universitāte, Francija

APSTIPRINĀJUMS

Apstiprinu, ka esmu izstrādājis šo promocijas darbu, kas iesniegts izskatīšanai Rīgas Tehniskajā universitātē inženierzinātnes doktora (Ph. D.) grāda iegūšanai. Promocijas darbs zinātniskā grāda iegūšanai nav iesniegts nevienā citā universitātē.

Rihards Novickis . . . . . (Paraksts)

Datums: . . . . . .

Promocijas darbs ir uzrakstīts angļu valodā, tajā ir ievads, četras nodaļas, secinājumi, literatūras saraksts, 10 pielikumu, satura rādītāju, 76 attēli, septiņas tabulas, kopā 122 lappuses. Literatūras sarakstā ir 132 nosaukumi.

SATURS

DARBĀ IZMANTOTO SAĪSINĀJUMU SARAKSTS 6

DARBA VISPĀRĒJAIS RAKSTUROJUMS 7

1. TEHNOLOĢIJU KONTEKSTS 12
   1.1 Apstrādes paradigmas 12
   1.2 Heterogēnas vienčipa sistēmas 14
   1.3 *Linux* operētājsistēma. 15

2. DATORREDZE 17
   2.1 Attēla formēšana un punktu atbilstību meklēšana 17
      2.1.1 Attēlu formēšana un lēcu ienestie kropļojumi 17
      2.1.2 Epipolārā ģeometrija 19
      2.1.3 Stereo attēlu atbilstību meklēšana 20
   2.2 Mākslīgā intelekta algoritmi 21

3. HETEROGĒNAS APRĒĶINU ARHITEKTŪRAS 23
   3.1 Uz atmiņas tiešpiekļuvi balstīta heterogēna apstrādes arhitektūra 23
   3.2 Asinhrona multiapstrādes apakšsistēma 25
   3.3 Pieeja programmatūras komponenšu pārvaldībai 26

4. ATTĒLU APSTRĀDES ALGORITMU PIELĀGOŠANA UN IMPLEMENTĀCIJA . . 28
   4.1 Pieeja neironu tīklu caurlaidespējas-optimizētai implementācijai *FPGA* 28
   4.2 Heterogēna sistēmas arhitektūra stereo attēlu apstrādei 31
   4.3 Stereoattēlu apstrāde 32
      4.3.1 Ieejas attēlu plūsmas sadalīšana 32
      4.3.2 Bajersa mozaīkas interpolācija un RGB intensitātes konvertācija 32
      4.3.3 Pieeja telpiskai attēlu transformācijai 34
      4.3.4 Iezīmju ekstrakcija 37
      4.3.5 Atbilstību meklēšana 38
   4.4 Demonstrācijas sistēma 38

5. SECINĀJUMI 40

IZMANTOTĀS LITERATŪRAS SARAKSTS 42

DARBĀ IZMANTOTO SAĪSINĀJUMU SARAKSTS

ADC – *Analog-to-Digital Converter*, analogais ciparu pārveidotājs.

AI – *Artificial Intelligence*, mākslīgais intelekts.

AMP – *Asynchronous Multi-Processing*, asinhrona multiapstrāde.

ANN – *Artificial Neural Network*, mākslīgie neironu tīkli.

API – *Application Programming Interface*, aplikācijas programmēšanas interfeiss.

ASIC – *Application Specific Integrated Circuit*, lietojumspecifiska integrālshēma.

AXI – *Advanced eXtensible Interface*, uzlabota paplašinātā saskarne.

CFA – *Color Filter Array*, krāsu filtru masīvs.

CLA-FPU – *Control Law Accelerator Floating Point Unit*, kontroles likumu paātrinātāja peldošā punkta bloks.

CMA – *Contiguous Memory Allocator*, nepārtrauktas atmiņas alokators.

CMOS – *Complementary Metal-Oxide Semiconductor*, komplementārā struktūra metāls-oksīds-pusvadītājs.

CNN – *Convolutional Neuron Networks*, konvolūcijas neironu tīkls.

CPU – *Central Processing Unit*, centrālais procesors.

DDR – *Double Data Rate*, dubulta datu pārraide.

DMA – *Direct Memory Access*, atmiņas tiešpiekļuve.

DNN – *Deep Neural Network*, dziļie neironu tīkli.

DSP – *Digital Signal Processing*, digitālo signālu apstrāde.

ELF – *Executable Linked Format*, izpildāms savienots formāts.

FFNN – *Feed Forward Neural Network*, Vienvirziena neironu tīkls.

FIFO – *First-In-First-Out*, pirmais iekšā, pirmais ārā.

FPGA – *Field Programmable Gate Array*, programmējamie loģikas masīvi.

HLS – *High-Level Synthesis*, augsta līmeņa sintēze.

HPC – *High Performance Computing*, superdators.

HSoC – *Heterogeneous System on Chip*, heterogēna vienčipa sistēma.

I/O – *Input/Output*, ieeja/izeja.

IC – *Integrated Circuit*, integrālshēma.

IP – *Intellectual Property*, intelektuālais īpašums.

ISA – *Instruction Set Architecture*, instrukciju komplekta arhitektūra.

MM – *Memory-Mapped*, atmiņas kartēts.

MPU – *Micro Processing Unit*, mikroprocesors.

NN – *Neural Network*, neironu tīkls.

OCRAM – *On-Chip Random Access Memory*, iekščipa operatīvā atmiņa.

OS – *Operating System*, operētājsistēma.

PCIe – *Peripheral Component Interconnect Express*, peripherālo komponenšu kopne.

PWM – *Pulse Width Modulation*, pulsa platuma modulācija.

RT – *Real-Time*, reāllaiks.

RTL – *Register Transfer Level*, starpreģistru pārsūtīšanas līmenis.

SCU – *Snoop Control Unit*, noklausīšanās kontroles bloks.

SIP – *Silicon Intellectual Property*, silīcija intelektuālais īpašums.

SIPO – *Serial-In-Parallel-Out*, serijveida ieeja, paralēla izvade.

SoC – *System on Chip*, vienčipa sistēma.

SRAM – *Static Random Access Memory*, statiskā brīvpiekļuves atmiņa.

ST – *Streaming*, straumēšana.

VHDL – *Very High Speed Integrated Circuit Hardware Description Language*, ļoti ātru integrālshēmu aparatūras projektēšanas valoda.

DARBA VISPĀRĒJS RAKSTUROJUMS

Tēmas aktualitāte

Promocijas darbā aplūkota mijiedarbība starp datorredzi un arvien sarežģītākām heterogēnām vienkristālshēmu (*HSoC*) tehnoloģijām. Sīkāk tiek aplūkots: integrētās shēmas (*IC*) aparatūras (*on-chip hardware*) un programmatūras koparhitektūras, stereoredzes un mākslīgā intelekta (*AI*) algoritmu implementācija un ar to saistītie reāllaika izpildes apsvērumi.

Izstrādātās metodes un algoritmi ir cieši saistīti ar Mūra likumu [1], kas ir kļuvis par ko vairāk nekā vienkāršu prognozi, jo tā dēļ ir izveidojusies mijiedarbība starp inovācijām un mūsdienu pusvadītāju uzņēmumiem. Sabiedrība sagaida uzlabotu veiktspēju un inovatīvus risinājumus, savukārt investori gaida jaunas tehnoloģijas.

Mūsdienu vienkristālshēmu (*SoC*) tehnoloģijām ir potenciāls atrisināt datorredzes izaicinājumus komerciāli izdevīgā veidā, bet vēl joprojām ir nepieciešams risināt vairāku skaitļošanas paradigmu sistēmu izstrādes izaicinājumus, kas iekļauj abstrakcijas no starpreģistru pārsūtīšanas līmeņa (*RTL*) līdz operētājsistēmai (*OS*) vai līdz pat vairāku sistēmu vadībai. Būtiski izaicinājumi ir arī algoritmu sadalīšana starp dažādām skaitļošanas paradigmām, uzticamas *IC* komunikācijas arhitektūras izveide un atbilstība reāllaika kontroles sistēmu veiktspējai.

Darba mērķis

Darba mērķis ir **izstrādāt un uzlabot datorredzes izstrādes paņēmienus un metodes *HSoC* tehnoloģijas lietojumos.** Izstrādātās metodes tiecas izveidot jaunas paaudzes inženierus un pētniekus, kuri izceltos ar izcilību dažādos *HSoC* izstrādes aspektos, kā rezultātā viņi būtu spējīgi risināt problēmas, lietojot dažādas komplementāras tehnoloģijas. Lai sasniegtu darba mērķi, definēti šādi uzdevumi:

- identificēt metodes *RTL* un programmatūrā bāzētu skaitļošanas paradigmu savstarpējā papildināšanā;
- izstrādāt programmatūras arhitektūras un rīkus *HSoC* tehnoloģijas lietošanai;
- izstrādāt heterogēnu pieeju attēlu apstrādes konveijeru (*pipeline*) implementācijai;
- implementēt un veikt izstrādāto rīku un algoritmu eksperimentālos pētījumus;
- veikt secinājumus no iegūtajiem rezultātiem.

Darba metodoloģiskais pamats

Pirmie promocijas darba uzdevumi veikti izmantojot analītisko metodi, ar kuru veikts pieejamās literatūras apskats par atbilstību meklēšanas algoritmiem, kā arī *HSoC* attēlu apstrādes konveijera izstrādē. Tālākie pētījumi un izstrāde balstīta inženiermetodēs un realizēta (*VHDL*), *C* un *Python* programmēšanas valodās. Izstrādātie rīki un *RTL* ir eksperimentāli testēti un validēti. Veiktie testi novērtē izstrādāto rīku un metožu efektivitāti, precizitāti un reāllaika veiktspēju.

Zinātniskā novitāte un galvenie rezultāti

Promocijas darba novitāte ir izstrādātās metodes un rīki *HSoC* bāzētu arhitektūru implementācijā un realizācijā reāllaika kontroles lietojumos un attēlu apstrādes konveijeros. Izstrādātie rīki un sistēmas arhitektūru realizācijas programmatūra ir pieejama tīmeklī.

Compage* un *icom. Iegultām sistēmām piemēroti modulāru programmatūras komponenšu menedžmenta satvari. Šie satvari kopā vienkāršo sistēmu prototipēšanu, kurās ir nepieciešama to izpilde reāllaikā (*RT*). *Compage* ir programmatūras komponenšu menedžmenta satvars, kuram ir mazs virstēriņš (*overhead*) un kas nodrošina iespēju programmatūras komponenšu konfigurācijai un replikācijai, savukārt *icom* nodrošina nemanāmu pārslēgšanos starp bezkopiju un dziļo kopiju komunikāciju pieejām un nodrošina standarta komunikācijas paradigmas – grūst–vilkt (*push–pull*), publicēt–abonēt (*publish-subscribe*) un pieprasīt–atbildēt (*request-reply*).

Asinhronās multiapstrādes (*AMP*) apakšsistēma. Jauna pieeja, kurā vismaz viens no procesora kodoliem tiek atvēlēts *RT* lietotnei, kamēr pārējā sistēmā darbojās augsta līmeņa *OS*. Izstrādātais risinājums sniedz šādas priekšrocības: *RT* apstrādes iespējas, ko sniedz *AMP* kodols, un (*FPGA*) un *Linux* programmatūras nodrošinājuma plašā funkcionalitāte. Saskarne ar *AMP* apakšsistēmu ir realizēta kā *Linux* dzinis (*driver*), kas nodrošina: (1) vadību pār *AMP* kodolu; (2) tiešsaskares (*baremetal*) izpildāmā saistītā formāta (*ELF*) lietotnes ielādi, lai to palaistu uz *AMP* kodola, ieskaitot tā virtuālās atmiņas konfigurāciju; (3) piekļuvi lietotnes izpildes diagnostikai; (4) lietotnes konfigurāciju tās darbības laikā; (5) komunikāciju ar *AMP* kodolu, izmantojot *stdin*, *stdout*, *stderr* straumes.

Metode caurlaidspējas maksimizēšanai tiešās izplatības neironu tīklu (*FFNN*) apstrādes konveijerā. Izstrādātā jaunā pieeja sniedz iespēju izstrādāt caurplūsmas optimizētu *FFNN*

apstrādes konveijeru. Tā pārskata neironu tīklu (*NN*) implementācijas uzdevumu un pielāgo *FFNN* aprakstu konveijerizācijai. *NN* tiek sadalīts elementārslāņos, kur katrs no slāņiem tiek saistīts ar abstraktu resursu. Šos resursus var piešķirt katram no slāņiem, nosakot arī visa konveijera aizturi (*delay*). Pieejas realizācijai tiek izstrādāts rīks, kas pārvērš *NN* topoloģiju par abstrahētu konveijera augsta līmeņa sintēzes (*HLS*) kodu. Rīkam ir nepieciešama tīkla topoloģija un mērķa latentums vienā konveijera posmā (elementārslānī). Izstrādātā metode ir piemērota virtuāla sensora implementācijai, it īpaši, ja nepieciešams liels nolašu ātrums.

***HSoC* bāzēta arhitektūra datorredzes lietojumiem**. *HSoC* bāzēts attēlu apstrādes konveijers, kas apvieno *Linux* bāzētu mikroprocesoru (*MPU*) sistēmas kontrolei un *FPGA* apstrādei. Izstrādātais atbilstību meklēšanas konveijers ir pilnīgi realizēts programmējamā loģikā. Tas sastāv no plūsmu sadalīšanas (*deinterleaving*), Bajersa mozaīkas interpolācijas (*Bayers pattern interpolation*), lēcas kropļojumu korekcijas, rektifikācijas, iezīmju ekstrakcijas (*feature extraction*) un punktu (pikseļu) meklēšanas (*correspondence matching*). Sistēmas kontrolieri galvenokārt izmanto paātrinātāja kontrolei, attēlu iegūšanai, izmantojot augstas veikstpējas (*PCIe*) kopni, un iegūto attēlu pārsūtīšanai uz demonstrācijas sistēmu. Viens no iespaidīgākajiem sasniegumiem ir pilnībā konveijerizēta attēla transformācijas shēma, kas labo attēlu lēcas kropļojumus un veic perspektīvas transformāciju. Izstrādātajā sistēmā tiek izmantota jauna pieeja N dimensionālu datu paralēlai piekļuvei digitālā loģikā, kurā arī tiek taupīti atmiņas resursi.

Aizstāvāmās tēzes

1. Izstrādātais asinhronās multiapstrādes apakšsistēmas risinājums ļauj vienlaikus izmantot modernu operētājsistēmu un atbalstīt reāllaika apstrādi, kurā kontroles cilpas latentuma trīces (*jitter*) standartnovirze nepārsniedz 0,1 ms.

2. Izstrādātā *FFNN* caurplūsmas optimizācijas izstrādes metode sniedz labāku veiktspēju, salīdzinot ar ap neironiem centrētām metodēm (2 – 30 reizes) un ar *RTL* izstrādātām metodēm (> 1000 reižu).

3. *HSoC* tehnoloģija ir piemērota, lai implementētu attēlu apstrādes konveijerus pilnīgi konveijerizētā veidā, nodrošinot mūsdienu reāllaika sistēmām atbilstošu veiktspēju, kur kontroles cilpas garums ir mazāks par 50 ms 1,5 MP lieliem attēliem.

Pētījuma galveno rezultātu aprobācija un praktiskais lietojums

Izstrādātie *Linux* dziņi un bibliotēkas, kas ir pieejami tiešsaistē saskaņā ar *MIT* licenci, nodrošina integrētās shēmas komunikāciju starp *FPGA* un programmatūru *HSoC* ierīcēs. Izstrādātie satvari sistēmas arhitektūras implementācijai – *compage* un *icom* – tiek lietoti *AI* uztveres konveijerā autonomā transportlīdzeklī, kā arī programmatūrā kontroles arhitektūras izveidē autonomos dronos. Turklāt iegūtās attēlu apstrādes zināšanas un attīstītie paātrinātāji patlaban tiek komercializēti Eiropas Reģionālā attīstības fonda (ERAF) projektā "Silīcija intelektuālā īpašuma izstrādes nams" (*SilHouse*) Nr. KC-PI-2020/12.

Promocijas darbs izstrādāts saistībā ar vairākiem projektiem, kas īstenoti Elektronikas un datorzinātņu institūtā (EDI), Latvijā.

- Valsts pētījumu programma "Kiberfizikālās sistēmas, ontoloģijas un biofotonika drošai&viedai pilsētai un sabiedrībai" (*SOPHIS*). Projekts Nr. 1 "Kiberfizikālo sistēmu tehnoloģiju attīstība un to pielietojumi medicīnā un viedā transporta jomā".
- Apvārsnis 2020 *ECSEL* projekts "*Integrated Components for Complexity Control in Affordable Electrified Cars*" (*3Ccar*), *G. A.* 662192.
- Apvārsnis 2020 *ECSEL* projekts "*Intelligent Motion Control Platform for Smart Mechatronic Systems*" (*I-MECH*), *G. A.* 737453.
- Apvārsnis 2020 *ECSEL* projekts "Programmējamas sistēmas inteliģencei automobiļos" (*PRYSTINE*), *G. A.* 783190.
- Apvārsnis 2020 *ECSEL* projekts "Uzlabota fotonikas, optisko un elektroniko komponenšu iepakošana/montāža zemu izmaksu ražošanai Eiropā" (*APPLAUSE*), *G. A.* 826588.
- Apvārsnis 2020 *ECSEL* projekts "Pamattehnoloģiju ietvars drošu un autonomu dronu lietojumiem" (*COMP4DRONES*), *G. A.* 826610.

Promocijas darba rezultāti ir atspoguļoti vairākās publikācijās [2]—[8]. Par rezultātiem ir ziņots vairākās starptautiskās konferencēs.

- 2018 5th International Conference on Control, Decision and Information Technologies (CoDIT), Thessaloniki, Grieķija.
- 2019 24th IEEE International Conference on Emerging Technologies and Factory, Zaragoza, Spānija.
- 2020 17th Biennial Baltic Electronics Conference (BEC), Tallina, Igaunija.

- 2020 23rd Euromicro Conference on Digital System Design (DSD), Kranj, Slovēnija.
- 24th Euromicro Conference on Digital System Design (DSD), Palermo, Itālija.

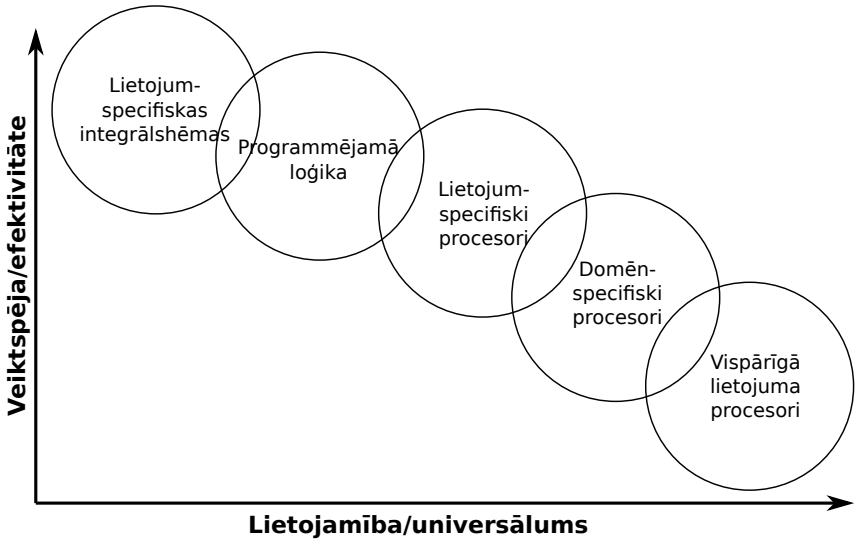
Darba struktūra

Promocijas darbā ir 122 lappuses, 76 attēli, septiņas tabulas, 132 literatūras avoti un 10 pielikumu. Promocijas darbs ir iedalīts piecās galvenajās sadaļās. 1. sadaļā sniegts kopsavilkums par skaitļošanas paradigmām un principiem, 2. sadaļā aprakstīti attēlu apstrādes principi un dažādas stereoredzes metodes. 3. sadaļa apraksta *HSoC* sistēmu arhitektūru izstrādi un to reāllaika īpašības. 4. sadaļā aplūkoti datorredzes algoritmu implementācijas un adaptācijas izaicinājumi *HSoC* sistēmās. 5. sadaļā tiek sniegti promocijas darba gaitā iegūtie secinājumi.

1. TEHNOLOĢIJU KONTEKSTS

1.1. Apstrādes paradigmas

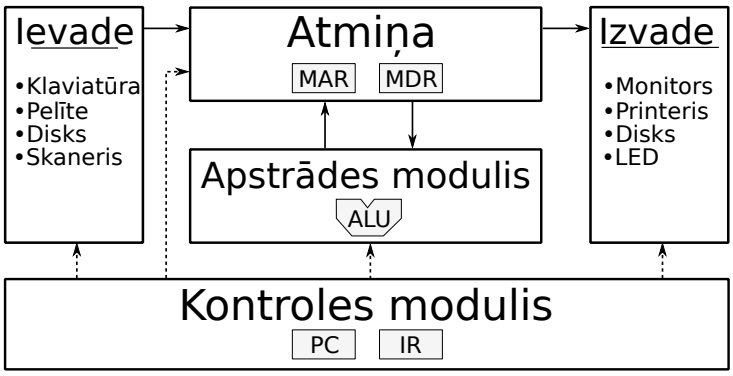
Protams, dažāda pielietojuma skaitļošanas īpašības un prasības ir veicinājušas aparatūras specializāciju un ietver kompromisu starp skaitļošanas platformas universālumu un veiktspēju, tādējādi radot virkni tehnoloģiju: vispārēja lietojuma procesori, domēnspecifiski procesori, lietojumspecifiski procesori, programmējamā loģika un specializētas integrētās shēmas.



1.1. att. Salīdzinājums starp dažādām skaitļošanas platformām.

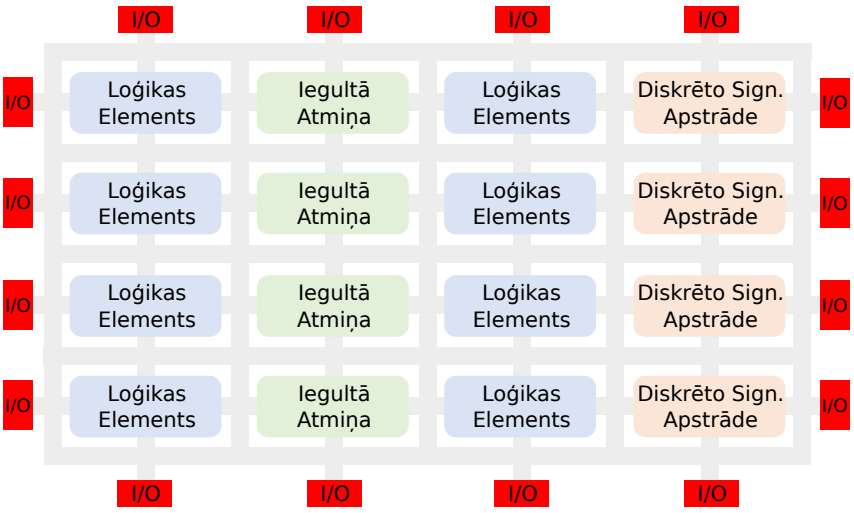
Mikroprocesora funkcionalitāti pilnībā raksturo instrukciju kopa, ko tas spēj izpildīt, ko mēdz saukt arī par instrukciju kopas arhitektūru (*ISA*). *ISA* ir definēta kā vienošanās starp programmatūru un aparatūru, kas ļauj neatkarīgi izstrādāt katru no tiem. Visbiežāk tā definē instrukciju kopumu, ko sauc par asamblera instrukcijām (*assembly instructions*), ko pēc tam nodrošina mikroarhitektūra. *ISA* pārsvarā attīstās ļoti lēni, jo veidojas inerce programmatūras pārkompilēšanas un atkārtotas izstrādes dēļ. Līdz ar to inovācija bieži ir mikroarhitektūras attīstības rezultāts [9].

Procesora instrukciju cikls ietver secīgās apstrādes paradigmas raksturīgo priekšrocību. Jāuzsver, ka šī sistēma ir universāla un to var izmantot, lai realizētu gandrīz jebkāda veida algoritmu. Šī īpašība ir radījusi plašu procesoru lietojumu, sākot no vienkāršiem tastatūras kontrolieriem līdz masīviem augstas veiktspējas skaitļošanas (*HPC*) serveriem. Tomēr šīs apstrādes paradigmas būtiskie trūkumi ir neregulārā atmiņas piekļuve un atkarība starp instrukcijām, kas var izraisīt konveijera apstāšanos (*pipeline stall*).



1.2. att. Neimaņa modelis [10] (*MAR* - Atmiņas Adreses Reģistrs, *MDR* – atmiņas datu reģistrs, *PC* – programmas skaitītājs, *IR* – instrukcijas reģistrs, *ALU* – aritmētikas un loģikas bloks).

Vēl viena būtiska tehnoloģija ir *FPGA*, kas ir kļuvusi par vienu no galvenajām vidēm ciparu shēmu realizācijai. *FPGA* ir iepriekš radīta silīcija ierīce, ko var elektroniski ieprogrammēt, lai tā kļūtu par gandrīz jebkāda veida digitālo shēmu vai sistēmu [11]. Vienkāršota *FPGA* struktūra redzama 1.3. attēlā. Tā sastāv no vispārīgas loģikas, atmiņas, digitālās signālu apstrādes (*DSP*) blokiem, starpsavienojumu loģikas (*routing fabric*) un programmējamiem ievadizvades (*I/O*) blokiem.



1.3. att. Vienkāršota *FPGA* struktūra.

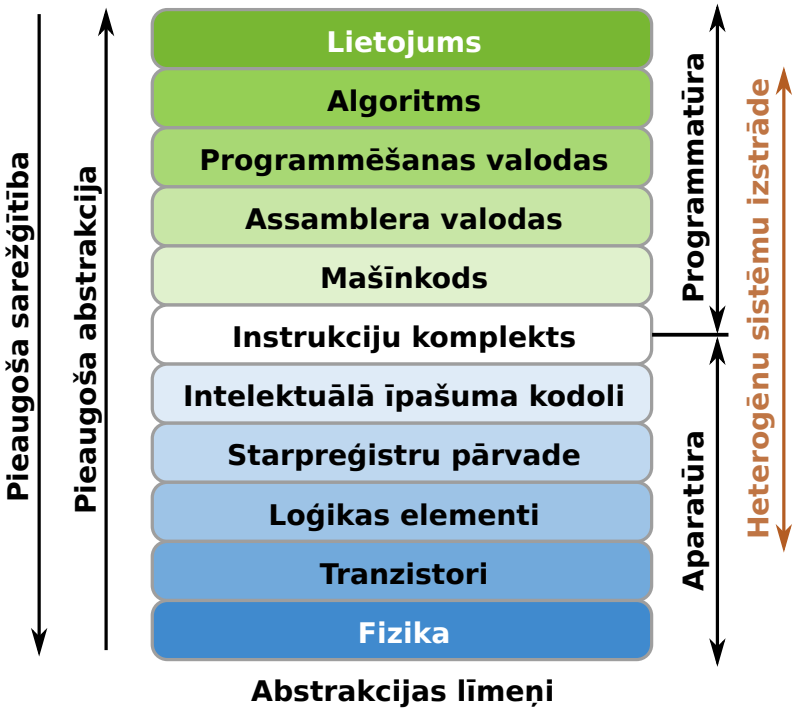
FPGA izstrāde bieži vien ietver kompromisa atrašanu starp veiktspēju un resursu izmantošanu. Atšķirībā no procesoru sistēmām, kurās programmatūra virza algoritma izpildi, *FPGA* dažādas daļas var būt paredzētas konkrētiem uzdevumiem. Viena no galvenajām veiktspējas palielināšanas metodēm jebkurai ciparu skaitļošanas sistēmai ir konveijerizācija (*pipelining*), kas ir īpašs paralēlisms, kas palielina sistēmas veiktspēju, pārklājot dažādu uzdevumu apstrādi [12].

Pilnīgi konveijerizēts risinājums var pieņemt jaunus ievades datus katrā taktī, un to raksturo fiksēts latentums.

Programmējamība un finansiālie apsvērumi veicina *FPGA* tehnoloģijas daudzos pielietojumus: trūkstošu vai neeksistējošu *IC* aizvietošana, pielietojumspecifisku *IC* (*ASIC*) prototipēšana/emulācija, zema latentuma pielāgoti sakari un datu maršrutēšana, datu priekšapstrāde un analīze, *HPC*, dziļās apmācības neironu tīkli (*DNN*) un daudzi citi.

1.2. Heterogēnas vienčipa sistēmas

Galvenie virzītāji enerģijas patēriņa samazināšanā un personalizētās skaitļošanas iespējās ir *SoC* un to paplašinājums - *HSoC*. *SoC* integrē virkni silīcija intelektuālā īpašuma (*SIP*) kodolus, ko ir izstrādājušas dažādas komandas visā pasaulē. Šie *SIP* kodoli integrē procesorus, kešatmiņas hierarhijas, starpsavienojumus (*interconnects*), līdzprocesorus, paātrinātājus, saskarnes kontrolierus (*interfacing controllers*), atmiņas, kriptēšanas blokus (*encryption engines*), perifērijas kontrolierus, analogas shēmas un daudz citas vienā mikroshēmā.



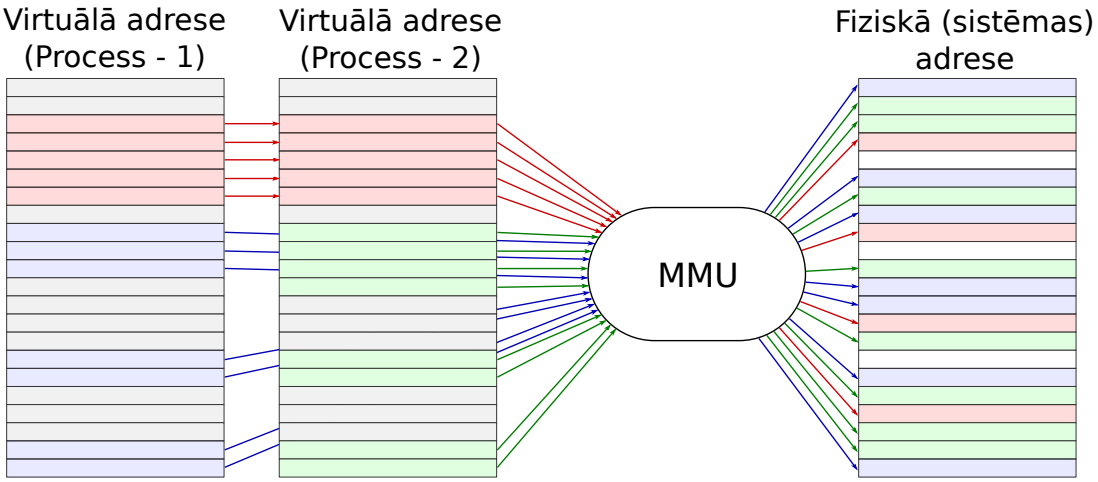
1.4. att. Abstrakcijas līmeņi skaitļošanas ierīcē.

Dažādu skaitļošanas paradigmu izmantošana vienlaikus rada izaicinājumus dažādu izstrādes plūsmu saskaņošanā. 1.4. attēlā redzami dažādi abstrakcijas līmeņi, kas ir daļa no jebkura pusvadītājos bāzēta lietojuma. Parasti *ISA* pārvalda saskarni starp aparatūru un programmatū-

ru. Tādēļ klasiskā programmatūras izstrādē viss ir fiksēts zemāk par *ISA* abstrakcijas līmeni (ieskaitot). Turpretim programmatūra nav tik būtiska klasiskā *FPGA* izstrādes plūsmā. *HSoC* integrācija ļauj izstrādāt daudz pielāgotākus aparatūras un programmatūras risinājumus. Šiem risinājumiem ir nepieciešamas daudznozaru zināšanas ne tikai par implementāciju, bet arī par algoritmu, kas ietekmē visus izstrādes posmus, kā arī prasa apsvērt tādus faktorus kā paātrinātāja aritmētiskā precizitāte, algoritma darbību lokālā daba un konveijerizācijas iespējas, datu plūsma mikroshēmā, kodola (*kernel*) dziņa implementācijas robustums, kodola/lietotāja aplikācijas programmēšanas saskarnes (*API*) efektivitāte, sistēmas reāllaika īpašības utt.

1.3. *Linux* operētājsistēma

Linux ir ļoti ietekmīga *OS*, un to izmanto dažādas sistēmas, sākot ar liela mēroga serveriem un beidzot ar modernām iegultām sistēmām [13]. *Linux* ir salīdzinoši mazas prasības, tā atbalsta plašas pielāgošanas iespējas un nodrošina vairākpavedienu darbību (*multi-threading*), kā arī tam ir pieejams plašs atvērtā koda programmatūras klāsts, kas var paātrināt izstrādi. *OS* ir atbildīga par sistēmas pieejamo resursu vienkāršu lietošanu un pārvaldību, t. i., procesora izpildes laika, atmiņas, glabātuves elementu, pievienoto ierīču, tīkla saskarni u.c.



1.5. att. Vienkāršots lapu virtuālo-fizisko adrešu kartējuma piemērs diviem procesiem. Lapas, kas iezīmētas ar sarkanu krāsu, apzīmē kodola apgabalu (*kernelspace*), savukārt ar zilu un zaļu krāsu iezīmētas lapas diviem atsevišķiem procesiem.

Viens no viskritiskākajiem apsvērumiem *HSoC* izstrādē ir virtuālās atmiņas koncepts, kas izolē lietojumprogrammas/procesus, nodrošina efektīvu starpprocesu komunikāciju (*inter-process communication*), sniedz iespēju pieprasīt vairāk dinamiskās atmiņas nekā ir pieejams (izmantojot lapumaiņu jeb *page-swapping*) un kopumā uzlabo sistēmas drošību [14]. Uz lapām balstīta

atmiņas virtualizācija nodrošina, ka fiziski nesecīga (*non-contiguous*) atmiņa programmatūrai izskatās kā secīga (*contiguous*), kā tas redzams 1.5. attēlā. Tomēr citi kopnes vedēji jeb *bus masters* (izņemot (*CPU*)) darbojas fiziskajā adrešu apgabalā. Lai gan daži vedēji (piemēram, *PCIe*) var rezervēt atmiņu startēšanās laikā, tas tāpat ir izaicinājums *HSoC* sistēmu izstrādē, jo gan centrālais procesors *CPU*, gan *FPGA* koplieto to pašu sistēmas atmiņu.

Pilnīga *HSoC* sistēmas izstrāde nozīmē arī kodola līmeņa izstrādi, jo kodola apgabals ir atbildīgs par komunikāciju starp programmatūru un izstrādāto aparatūru (paātrinātajiem), kā arī realizē saskarni ar lietotāja apgabalu (*userspace*). Lai nodrošinātu pareizu saskarni starp izveidoto aparatūras paātrinātāju un lietojumprogrammu *HSoC* tehnoloģijas kontekstā, ir jāņem vērā visi programmatūras abstrakcijas līmeņi no ierīces dziņa izstrādes līdz pat lietojumprogrammai.

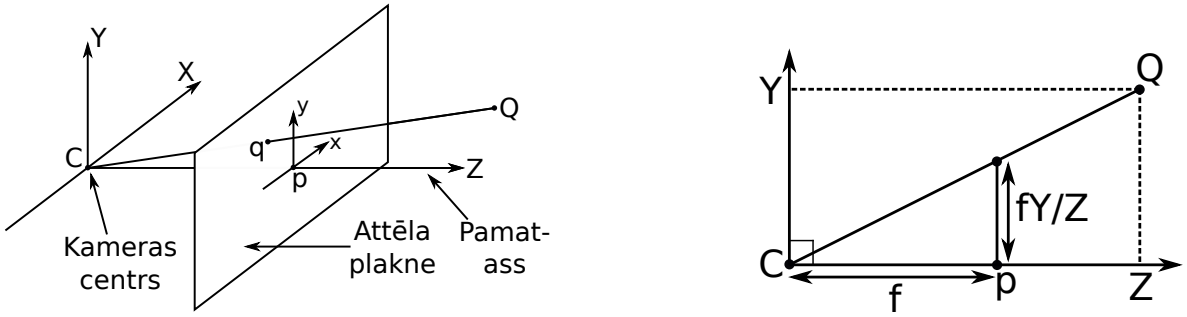
2. DATORREDZE

2.1. Attēla formēšana un punktu atbilstību meklēšana

2.1.1. Attēlu formēšana un lēcu ienestie kropļojumi

Pēc būtības [15] kamera ir translācija starp 3D pasauli un 2D attēlu. Kameru modelēšana ir balstīta uz vispārējo projekcijas kameras principu, kas var tikt izteikts kā matrica *P*, kas nodrošina kartē pasaules punktus Q kā punktus attēlu plaknē q.

Pieņemsim, ka plakne $Z = \mathrm{f}$, kuru sauc par attēla plakni vai fokālo plakni. Projekcijas kameras modelis nosaka, ka telpas punkts $\mathrm{Q} = (X, Y, Z)^T$ atbilst punktam attēlu plaknē, kur līnija no Q līdz projekcijas centram šķērso attēlu plakni, kā parādīts 2.1. attēlā.



2.1. att. Vispārējā projekcijas kameras ģeometrija (attēlu plakne ir ilustrēta spoguļattēlā uz pasaules pusi).

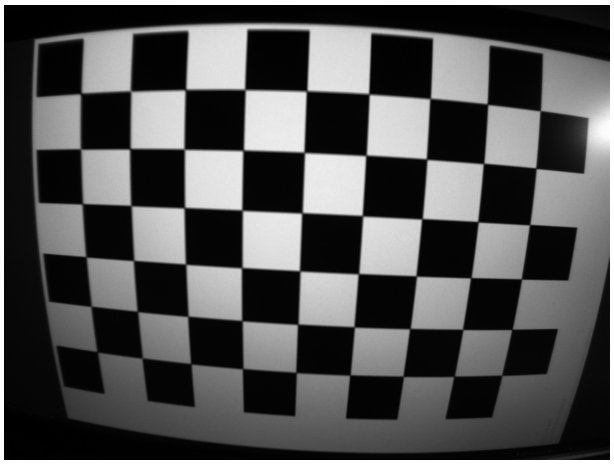
Pasaules un attēlu punkti tiek reprezentēti, izmantojot homogēnās koordinātes un ņemot vērā transformācijas sākumpunktu, projekcija var tikt izteikta kā matricu reizināšana:

$$\begin{pmatrix} x \\ y \\ \omega \end{pmatrix} = \begin{pmatrix} \mathrm{f} & 0 & p_x & 0 \\ 0 & \mathrm{f} & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.1}$$

Bieži vien matrica tiek pārrakstīta, apzīmējot matricu *K*, ko dēvē par *kameras kalibrācijas matricu*, līdz ar to projekcijas modeļa apraksts ieņem kodolīgu pierakstu:

$$q = K[\mathrm{I}|\mathrm{O}]\mathrm{Q}. \tag{2.2}$$

Attēlu apstrādes modelis pieņem, ka kameras atbilst *lineāram* projekcijas modelim, kur taisna līnija pasaules koordinātu sistēmā atbilst taisnai līnijai attēla koordinātu sistēmā [16]. Diemžēl lēcas, kas fokusē gaismu uz kameras pikseļu masīvu, ievieš redzamus lēcu kropļojumus, kas izpaužas taisnu līniju projekcijas izliekumā, kā tas ilustrēts 2.2. attēlā.



2.2. att. Piemērs ar *Bumblebee* stereo kameras lēcu ieviestiem kropļojumiem, attēlojot kalibrācijas attēlu.

Ja vien kropļojumi nav izlaboti, nav iespējams izveidot precīzas fotoreālistiskas rekonstrukcijas [16]. Radiālie (lēcu) kropļojumi var tikt modelēti kā:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(\widetilde{r}) \begin{pmatrix} \widetilde{x} \\ \widetilde{y} \end{pmatrix}, \tag{2.3}$$

kur:

- $(\widetilde{x}, \widetilde{y})$ – koordinātes ideālā attēlā (kurš atbilst lineārai projekcijai);
- (x_d, y_d) – koordinātes patiesajā attēlā pēc radiālajiem kropļojumiem;
- $\widetilde{r}$ – radiālā distance $\sqrt{\widetilde{x}^2 + \widetilde{y}^2}$ no radiālo kropļojumu centra;
- $L(\widetilde{r})$ – kropļojumu faktors, kurš ir funkcija no rādiusa $\widetilde{r}$.

Pikseļu koordinātēs korekcija var tikt pārrakstīta kā:

$$\begin{aligned} \widetilde{x} &= x_c + (x - x_c)L(\widetilde{r}) \\ \widetilde{y} &= y_c + (y - y_c)L(\widetilde{r}) \end{aligned}, \tag{2.4}$$

kur (x, y) ir "nepieciešamās" koordinātes izlabotajā attēlā, savukārt $(\widetilde{x}, \widetilde{y})$ ir koordinātes izkropļotā (ieejas) attēlā.

Kropļojumu faktors ir patvaļīga funkcija, kas ir definēta pozitīvām vērtībām, un kropļojumu centram $L(0)$ ir 1. Kropļojuma faktors var tikt izteikts Teilora rindas veidā:

$$L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + ..., \tag{2.5}$$

kur radiālās korekcijas koeficienti $k_1, k_2, k_3, ...$ tiek uzskatīti par daļu no kameras iekšējās kalibrācijas [15].

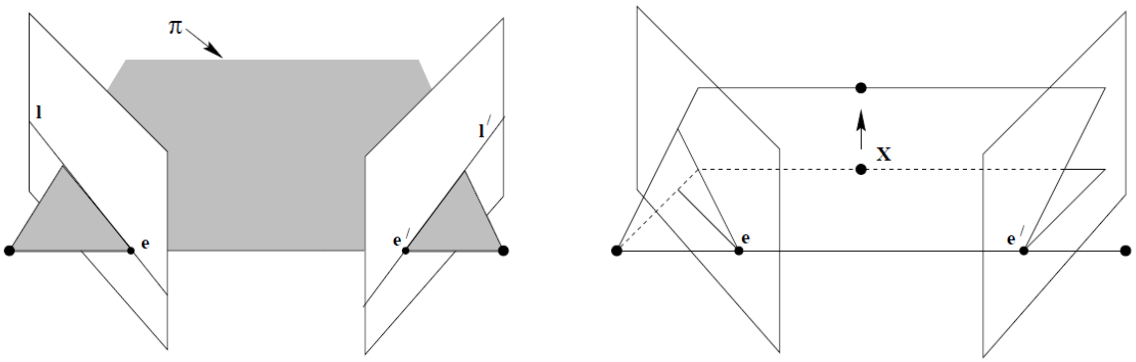
2.1.2. Epipolārā ģeometrija

Svarīga jebkura lietojuma, kas ietver 3D rekonstruēšanu no attēliem, daļa ir epipolārā ģeometrija, kas sasaista divus skatus, kas balstīti tikai uz kameras iekšējiem parametriem un relatīvo novietojumu. Šo sakarību apraksta *fundamentālā matrica* F. F ir 3×3 matrica, un tai piemīt tāda īpašība, ka, ja punkts 3D telpā X projicējas kā x pirmajā skatā un kā x' otrajā skatā, tad tiek apmierināta šāda sakarība:

$$x'Fx = 0. \tag{2.6}$$

Epipolārā ģeometrija starp diviem skatiem pēc būtības raksturo attēla plakņu krustošanās ģeometriju ar to plakņu telpas punktu un asi, ko veido kameru centri [15]. Epipolāro ģeometriju aktualizē atbilstošo punktu meklēšana dažādos skatos, jo atbilstošie punkti vienmēr atradīsies uz epipolārās plaknes, un tie atradīsies uz līnijām, kuras veido šīs plaknes šķērsgriezums ar attēlu plaknēm, šīs līnijas arī sauc par *epipolārajām līnijām*.

2.3.a attēla punkti x, x', telpas punkts X un kameru centri ir koplanāri (atrodas vienā plaknē). Apzīmējot šo plakni ar π, atpakaļ projicētie stari no x un x' veido krustpunktus ar X, kas ir koplanāri un atrodas uz π.



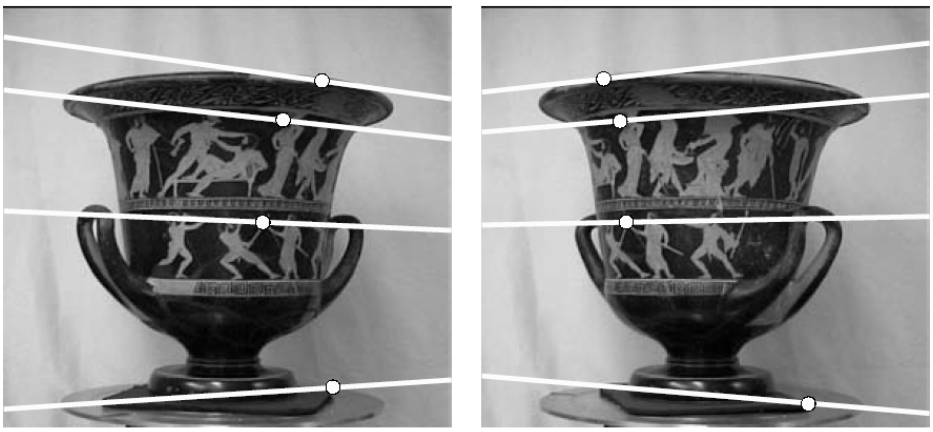
(a) Kameru pamatlīniju šķērsgriezums ar attēlu plaknēm.

(b) Vairākas plaknes, kas iegūtas, rotējot epipolāro plakni ap kameru pamatlīniju.

2.3. att. Epipolārā ģeometrija [15].

Ja x ir zināms, epipolārā ģeometrija ierobežo x' atrašanās vietu plaknē π, tādējādi iespējamā atbilstība (korespondence) jāmeklē uz epipolārās līnijas. 2.3.b attēlā redzama π plaknes rotācija, tādējādi radot epipolārās plaknes un attēlu plakņu šķērsgriezumus (epipolārās līnijas).

Šīs epipolārās līnijas var tikt "iztaisnotas", pielietojot attēlu rektifikāciju. Rektifikācija ir transformācijas process, kura rezultātā stereo attēlu epipolārās līnijas tiek pārveidotas tā, lai tās



2.4. att. Piemērs stereo attēlu atbilstību meklēšanai uz epipolārajām līnijām [15].

abos attēlos pārklātos, kas nozīmē, ka epipolārās līnijas kļūst paralēlas x asij. Tādējādi punktu atbilstības meklēšanas process var tikt vienkāršots, meklējot atbilstības vien x virzienā, pēc būtības izslēdzot atbilstību meklēšanu pa y asi.

2.1.3. Stereo attēlu atbilstību meklēšana

Stereo attēlu atbilstību meklēšana ir bijusi un turpina būt viena no vispētītākajiem tematiem datorredzes zinātnē. Termins "*disparity*" (atšķirība) pirmo reizi tikai ieviests cilvēka redzes kontekstā [17], un tas tika izmantots, lai aprakstītu atbilstošās iezīmes, kuras redz cilvēka labā un kreisā acis.

Pastāv divas plašas stereo attēlu apstrādes algoritmu klases – lokālie un globālie algoritmi. Lokālās (uz logu balstītās) metodes diferenču aprēķins ir atkarīgs tikai no konkrētā punkta vērtības kādā galīgā loga, savukārt globālās metodes izvirza svarīgus nepārtrauktības pieņēmumus un risina optimizācijas problēmu. Abas šo algoritmu klases apvieno iteratīvi algoritmi, kuriem ir samērā lokāla algoritma atbilstības funkcijas minimizācija.

Szeliski [15] norāda kopu ar šādiem stereo attēlu apstrādes algoritmiskiem "blokiem", no kuriem var tikt izveidots plašs algoritmu klāsts [18]: (1) atbilstības funkcijas aprēķins; (2) atbilstības metriku apkopošana; (3) diferences attēla (atbilstību) aprēķins; (4) diferences attēla uzlabošana.

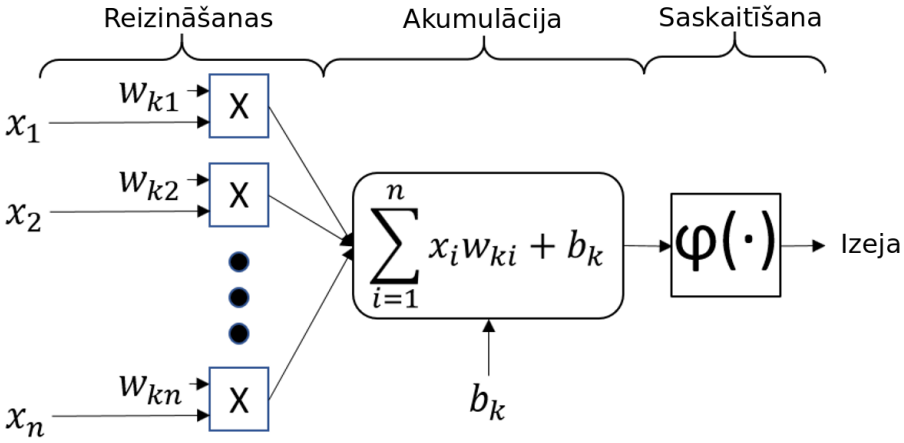
Korespondences problēmas risināšanai nepieciešama metrika, lai salīdzinātu atsevišķus stereo attēlu pikseļu pārus. Metrikas pamatā ir vai nu pikseļu pelēktoņu vērtības, vai pēcapstrādātas iezīmes, kas atšķiras pēc to spējas atklāt pikseļu unikalitāti, to uzņēmības pret

kameras pastiprinājumu un to skaitļošanas sarežģītības. Tādas metrikas kā *census* transformācija, ir ne tikai nejutīgas pret kameras pastiprinājuma maiņu, bet arī ērti ieviešamas digitālajā loģikā. Viens no šādu algoritmu galvenajiem trūkumiem ir to paaugstinātā jutība pret troksni.

Atbilstību metriku apkopošana tiek veikta, vienkārši saskaitot dažādas salīdzināšanas metrikas kopā kādā galīgā logā ar noteiktu diferences vērtību. Lokālajās metodēs tiek likts uzsvars uz atbilstības metriku aprēķina un apkopošanas soļiem. Gala diferences aprēķins ir triviāls: tiek izvēlēta diference ar vismazāko apkopoto atbilstības metrikas vērtību, pēc būtības lietojot lokālu "uzvarētājs iegūst visu" pieeju. Šādu metožu trūkums ir tas, ka to atbilstību unikalitāti nosaka tikai viens attēls, t. i. references attēls, savukārt punkti citā attēlā var atbilst vairākiem punktiem [16].

2.2. Mākslīgā intelekta algoritmi

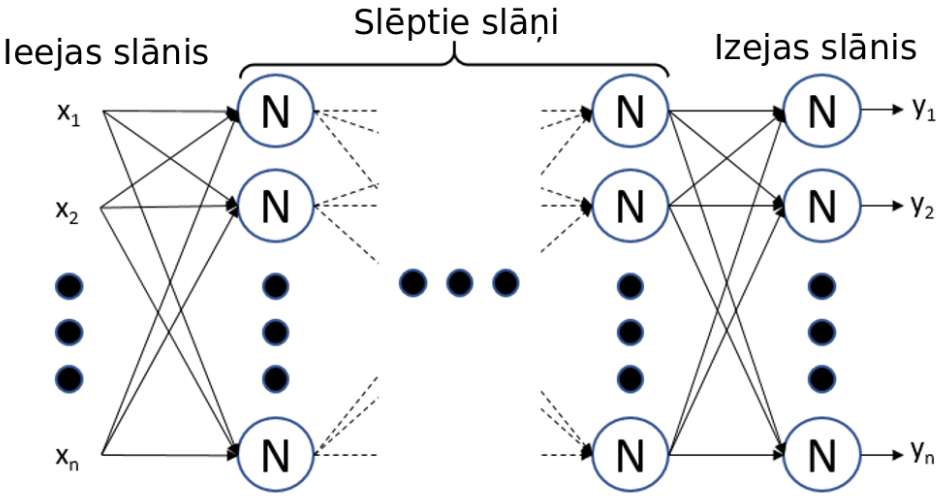
Neironu tīkli *NN* ir sistēma, kas ir izstrādāta, modelējot smadzenes konkrēta uzdevuma vai funkcijas izpildi [19]. Tīkls var tikt sadalīts fundamentālos apstrādes elementos – neironos, kas veido mākslīgā neironu tīkla izstrādes bāzi. Bloku diagramma, kas redzama 2.5. attēlā, ilustrē neirona matemātisko modeli. Neirona ieejas x_i ir sareizinātas ar koeficientiem w_{ki}, ko devē par svariem, un tālāk tos sasummē kopā ar nobīdes koeficientu b_k. Šo summu tālāk nodod aktivācijas funkcijai φ, kas ir izmantota, lai normalizētu neirona izeju, k un i apzīmē konkrēto neironu un tā ieeju.



2.5. att. Viena neirona struktūra.

Samērā bieži neironu tīkli tiek būvēti organizējot neironu slāņos, kur katra neirona izeja iepriekšējā slānī ir savienota ar visiem neironiem nākamajā slāni, t. i. to dēvē par pilnībā savienotu

neironu tīklu. Šāda veida tīkla struktūra redzama 2.6. attēlā.



2.6. att. Vispārīgā pilnībā savienota neironu tīkla struktūra.

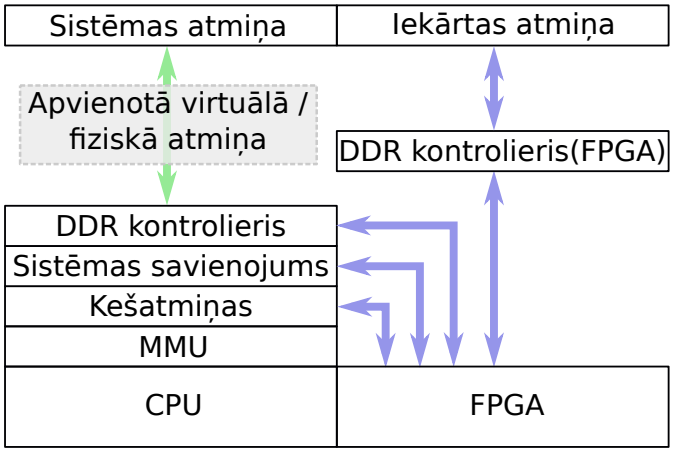
Kopš *NN* apraksta formalizēšanas [20], ir veltītas pūles un lietotas visdažādākās paradigmas neironu struktūru pielāgošanai digitālām shēmām. Piemēram, konvolūcijas neironu tīkli (*CNN*) ir plaši izmantoti attēlu atpazīšanas un klasificēšanas uzdevumos, lai gan pašu klasifikāciju nodrošina pilnībā savienoti neironu tīkli *FFNN*. Dažāda veida paradigmas, sākot no kopapstrādes sistēmām [21] līdz *OpenCL* bāzētiem risinājumiem [22]—[26], ir izmantotas, lai atrastu optimālo kompromisu starp resursu izmantošanu, latentumu un caurlaides spēju.

Programmējamās loģikas raksturīgi paralēlā daba liek domāt, ka šī tehnoloģija ir labi piemērota *FFNN* implementācijai. Lai gan ir lietotas dažādas arhitektūras un izstrādes paradigmas, *FFNN* implementācijai jāsastopas ar loģikas resursu trūkuma izaicinājumu. Turklāt neironu tīklu topoloģiju ekspansija [27] un čipu ražošanas tehnoloģiju izaugsmes piesātinājums [28] liek manīt resursu nepietiekamības izaicinājumu.

3. HETEROGĒNAS APRĒĶINU ARHITEKTŪRAS

3.1. Uz atmiņas tiešpiekļuvi balstīta heterogēna apstrādes arhitektūra

HSoC tehnoloģija paredz tādu algoritmu realizāciju, kur katru apakšuzdevumu izpildīta vispiemērotākā aprēķinu tehnoloģija, t. i. procesors ir labāk piemērots sekvenciālai apstrādei, piemēram, lēmumu pieņemšanai un kontrolei, savukārt *FPGA* tehnoloģija piemērota augstas caurlaides aprēķiniem un samērā lokāliem algoritmiem, piemēram, pikseļu operācijām, konvolūcijai un iezīmju ekstrakcijai.



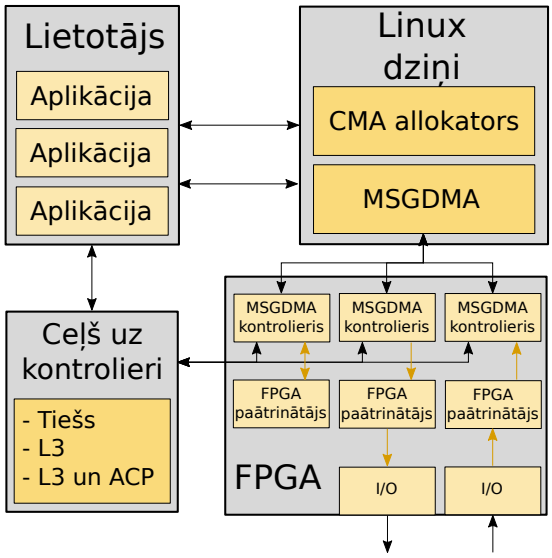
3.1. att. Modernu *HSoC* iekārtu atmiņas modelis un datu komunikācijas iespējas, kur *FPGA* ir pieeja atsevišķai atmiņai, kopīgai atmiņai, iekšējai sistēmas savienojumloģikai un procesora kešatmiņai.

Daudzas mūsdienu *HSoC* iekārtas dod unikālu iespēju datu koordinēšanai, kā tas redzams 3.1. attēlā. *FPGA* var: (1) implementēt savu dubultas datu pārraides (*DDR*) atmiņas kontrolieri, tādējādi nodrošinot lielu apjomu ar sev privātu atmiņu; (2) tieši piekļūt procesora sistēmas *DDR* kontrolierim, iespējojot veiktspējīgu, bet nekoherentu datu pārvadi; (3) piekļūt sistēmas starpsavienojuma loģikai, piekļūstot arī procesora iekšējās operatīvās atmiņas (*OCRAM*) apgabalam; (4) piekļūt koherentām kešatmiņas saskarnēm, kas nodrošina datu koherenci digitālas implementācijas līmenī. Iepriekšminētie aspekti nodrošina unikālas iespējas sistēmas izstrādātājam, kamēr vienlaikus radot izaicinājumus paātrinātāju abstrakcijai un lietotāja programmatūras kontrolei.

Adresējot fundamentālos *HSoC* tehnoloģijas iekščipa komunikācijas izaicinājumus, tika izstrādāta atmiņas tiešpiekļuvei (*DMA*) centriska augstas veiktspējas komunikācijas arhitektūra, kas nodrošina komunikāciju starp *FPGA* un programmatūru iegultās *Linux* sistēmas. Izstrā-

dātais risinājums ņem vērā *Linux* atmiņas fragmentācijas problēmu un izmanto *Linux* kodola nepārtrauktas atmiņas alokatora (*CMA*) funkciju. Turklāt komunikācijas ceļu veiktspēja ir nomērīta, izmantojot *Cyclone V SoC* iekārtu, jo šādi dati zinātniskajā literatūrā vēl nebija pieejami [29]—[31]. Izstrādātie *Linux* moduļi un bibliotēkas ir publicētas tiešsaistē ar *MIT* licenci[1].

Svarīgs konkrētās sistēmas struktūras elements ir trešā līmeņa (L3) savienojumloģika, kas ir centrālais slēdzis un nodrošina datu pārvadi starp atmiņu, *FPGA* loģiku, procesoru un perifēriju [32]. *DMA* kontroles, atmiņas virtualizācijas un atmiņas fragmentācijas apsvērumi ir veicinājuši modulāras *FPGA* loģikas pārvaldībā balstītas arhitektūras izstrādi, kas redzama 3.2. attēlā.



3.2. att. Konceptuāla *FPGA* pārvaldībā balstīta arhitektūra.

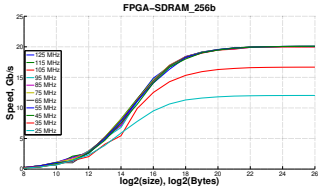
Veiktspējas mērījumu procedūrā lietots *VEEK-MT Cyclone V SoC* izstrādes rīks, mainot komunikācijas saskarņu platumus, transakciju izmērus un *FPGA* loģikas taktēšanas frekvenci. 3.1 tabulā apkopotas datu pārvades maksimālās (piesātinājuma) caurlaides spējas atkarībā no sistēmas konfigurācijas, savukārt 3.3. attēlā redzami datu caurlaides spējas mērījumi visplatākajai kopnes konfigurācijai ar dažādām *FPGA* loģikas takts frekvencēm. Nomērītais jaudas patēriņš visās konfigurācijās tika mērīts 300 sekundes ar izstrādes plates jaudas monitora mikroshēmu, kas nodrošina $\pm 1,0\%$ kļūdu, tomēr dažādu saskarsņu platumu mērījumus nav iespējams izšķirt. Jaudas patēriņš dīkstāvē ir $8,11$ W, savukārt, realizējot datu pārvades scenārijus, $8,18$ W.

Apskatītā tehnoloģija (*Intel Cyclone V SoC*) var nodrošināt 20,08 *Gbps* iekščipa datu pārvadi, kas ir pietiekami reāllaika attēlu kopapstrādei. 3.3.c attēlā redzami interesanti rezultāti, kur komunikācijas caurlaidspēja sasniedz maksimumu un tad samazinās līdz noteiktai piesāti-

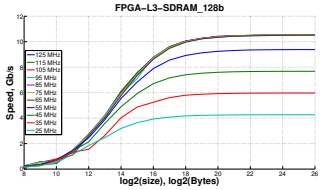
[1] http://git.edi.lv/rihards.novickis/FPSoC_Linux_drivers

3.1 tabula. Visu komunikācijas saskarņu konfigurāciju maksimālās noteiktās caurlaidības vērtības veicot rakstīšanas un lasīšanas operācijas

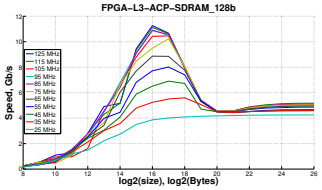
Datu ceļš	Saskarnes platums	Maksimālā caurlaidība	Piesātinājuma frekvence
FPGA-L3-SDRAM	32 biti	5,05 Gbps	120 MHz
	64 biti	10,10 Gbps	120 MHz
	128 biti	10,52 Gbps	65 MHz
FPGA-L3-ACP-SDRAM	32 biti	6,90 Gbps	–
	64 biti	8,64 Gbps	120 MHz
	128 biti	11,26 Gbps	90 MHz
FPGA-SDRAM	32 biti	7,52 Gbps	–
	64 biti	14,64 Gbps	–
	128 biti	17,68 Gbps	80 MHz
	256 biti	20,08 Gbps	45 MHz



(a) *FPGA-SDRAM* saskarne ar 256 bitu kopnes konfigurāciju.



(b) *FPGA-L3-SDRAM* saskarne ar 128 bitu kopnes konfigurāciju.



(c) *FPGA-L3-ACP-SDRAM* saskarne ar 128 bitu kopnes konfigurāciju.

3.3. att. Caurlaides spējas mērījumi, izmantojot dažādus *FPGA* pārvaldītus komunikācijas ceļus.

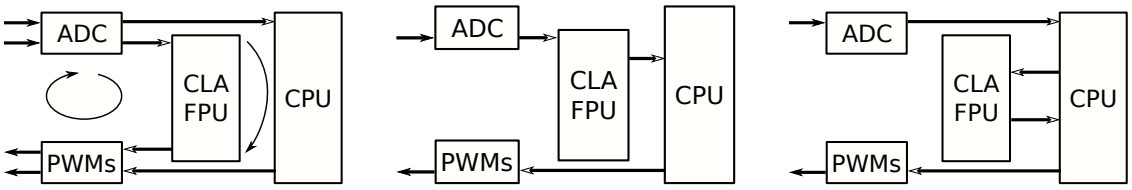
nājuma vērtībai. Šo īpašību var izskaidrot ar datu pieprasījumu ”nokļūšanu” kešatmiņā, kuras izmērs *Intel Cyclone V* tehnoloģijā ir 512 KB.

3.2. Asinhrona multiapstrādes apakšsistēma

Ievērojams šī promocijas darba pienesums ir saistīts ar jaunu heterogēno arhitektūru izmantošanu reāllaika kontroles cilpas sistēmās, izmantojot pieeju, kas balstīta uz asinhronas multiapstrādes *AMP* apakšsistēmu. Kamēr *FPGA* tehnoloģija ir labi pielāgota reāllaika kontrolei tās noteiktības īpašību dēļ, dažu apakšuzdevumu paātrināšana, kur nepieciešama globāla piekļuve atmiņai, var būt problemātiska vai pat neiespējama.

Vienkāršots piemērs ar dažādām iegulto sistēmu kontroles cilpu konfigurācijām ilustrēts 3.4. attēlā. Ņemot vērā latentuma, uzticamības, nepieciešamības pēc lokālām atgriezeniskām cilpām un lietojuma līmeņa konfigurācijas prasības, tika radīta koncepcija modulārai signālu apstrādes apakšistēmai kustības kontroles aplikācijām, kas balstīta *HSoC* tehnoloģijā [33]. Izstrādātā *AMP*

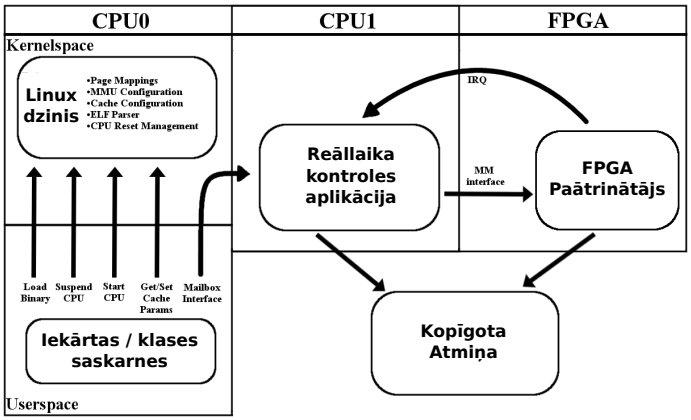
apakšistēma ir daļa no modulāras apstrādes sistēmas [33].



(a) Paralēlu kontroles cilpu izpilde. **(b)** Ieejas datu priekšapstrāde. **(c)** Kopapstrāde ar galveno procesoru.

3.4. att. Uzņēmuma "Teksasas instrumenti" piemēri kontroles cilpas konfigurācijai, izmantojot viņu kontroles likumu paātrinātāja peldošā punkta bloku (*CLA-FPU*) [34]. Analogais ciparu pārveidotājs (*ADC*) mēra atgriezeniskās saites signālus un sāk kontroles cilpas izpildi, savukārt pulsa platuma modulators (*PWM*) reprezentē sistēmas izeju – aktuatoru kontroli.

Izstrādātās *AMP* apakšsistēmas risinājums nodrošina labāko no divām pasaulēm: reāllaika apstrādes īpašības, ko nodrošina *AMP* kodols, un *FPGA* un *Linux* operētājsistēmas programmatūras funkcionalitāti. 3.5. attēlā redzama izstrādātās *AMP* apakšsistēmas vispārējā koncepcija, kur divkodolu procesora sistēmas nultais kodols izpilda *Linux* operētājsistēmas kodu, savukārt cits kodols (*AMP* apakšsistēma) sadarbojas ar *FPGA* reāllaika uzdevumu izpildē.

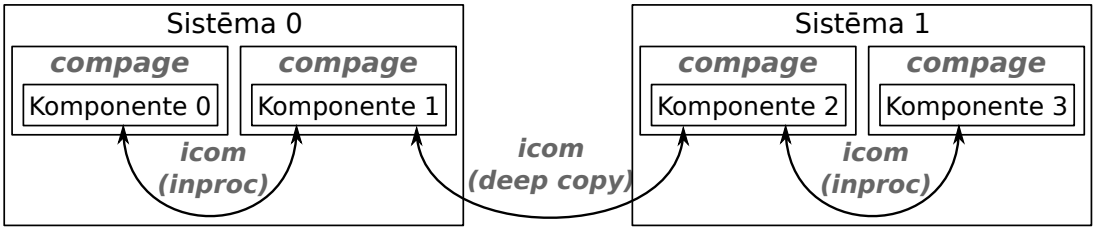


3.5. att. Reāllaika lietojumu apakšsistēma.

3.3. Pieeja programmatūras komponenšu pārvaldībai

Ir izstrādāta jauna programmatūras komponentēs balstīta arhitektūras koncepcija, kas jau ir veiksmīgi izmantota autonomu transportlīdzekļos un dronu sistēmās. Izstrādājot komplicētas un laikā mainīgas sistēmas, var rasties nepieciešamība pēc arhitektūras, kas atvieglo programmatūras

komponentu pārvaldību un nodrošina ērtus līdzekļus to savstarpējai saziņai. Šī pieeja atgādina *"blackboard"* paradigmu, kas izmantota programmatūras izstrādes teorijā [35].



3.6. att. Programmatūras pārvaldības pieeja balstīta uz programmatūras komponentēm.

3.6. attēlā redzama izstrādātās komponenšu pārvaldības un to starp-komunikācijas programmatūras risinājuma principiālā struktūra. Izstrādātais risinājums balstās divos modulāros rīkos:

- ***compage*** (*component management framework*) – nodrošina komponenšu pārvaldību vienā sistēmā, ieskaitot komponentu inicializāciju un izpildi atsevišķos pavedienos vai procesos. Rīks ietver arī iespēju ģenerēt un izmantot konfigurāciju, lietojot *.ini* failus. Izstrādātais rīks ir pieejams tiešsaistē[2];
- ***icom*** (*component communication framework*) – nodrošina koherentu saziņu starp dažādām programmatūras komponentēm, izmantojot dažādus saziņas pamata mehānismus, tādējādi nodrošinot nulles datu kopēšanas īpašību, ja vien to pieļauj programmatūras komponenšu uzstādījums. Būtībā, ja programmatūras komponentes tiek izpildītas vienā sistēmā, tās var koplietot atsauces uz datiem un izvairīties no datu kopēšanas, piemēram, izmantojot divu buferu komunikācijas metodi [36]. Izstrādātais rīks ir pieejams tiešsaistē[3].

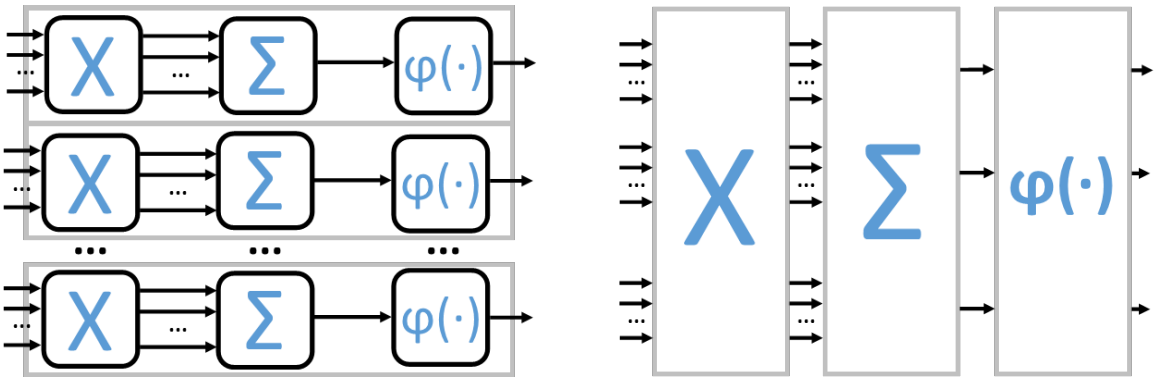
[2] https://gitlab.com/rihards.novickis/compage.
[3] https://gitlab.com/rihards.novickis/icom.

4. ATTĒLU APSTRĀDES ALGORITMU PIELĀGOŠANA UN IMPLEMENTĀCIJA

4.1. Pieeja neironu tīklu caurlaidespējas-optimizētai implementācijai *FPGA*

Tika izveidota jauna pieeja *FFNN* tīklu realizācijai, kas apskata implementācijas uzdevumu, ņemot vērā nepieciešamību sasniegt maksimālo datu caurlaidi, apskatot tīklu kā elementāru struktūru un izmantojot konveijerizācijas apstrādes paradigmu. *FFNN* ir sadalīts vienkāršos slāņos, kur katram slānim piesaistīts konkrēta resursa tips, piemēram, saskaitītājs, reizinātājs, aktivizācijas funkcija. Šiem slāņiem tiek rezervēts dažāds daudzums resursu, lai panāktu vienmērīgu aizkavju sadali un optimālu konveijera principa ieviešanu, kur katra slāņa latentums ir mazāks vai vienāds ar laiku, kas nepieciešams tīkla jaunu ieeju pieņemšanai. Risinājuma ietvaros tika izstrādāts rīks, kas konvertē konkrētā tīkla topoloģiju *C* kodā, kas tālāk tiek nodots *HLS* rīkam. Uzģenerētajā kodā jau ir ietvertas nepieciešamās direktīvas, lai nodrošinātu risinājumu ar uzstādīto konveijerizācijas iterācijas intervālu.

Lai gan neirons ir intuitīva abstrakcija, tas sadala arhitektūru kā redzams 4.1.a attēlā. Šāda pieeja nepieļauj resursu atkārtotu izmantošanu starp paralēli izvietotiem neironiem. Līdz ar to tiek piedāvāta cita pieeja, kura redzama 4.1.b attēlā, šeit neironu tīkla struktūra ir sadalīta elementāros slāņos, kur katrs slānis tiek asociēts ar konkrētu resursu – saskaitītājs, reizinātājs vai aktivācijas funkcija.



(a) Neoptimizēts aiztures modelis vienam slānim. **(b)** Optimizēts aiztures modelis vienam slānim.

4.1. att. Dažādi aiztures modeļi. a) Aiztures tiek analizētas, izmantojot neironu abstrakciju. b) Aiztures tiek analizētas, izmantojot vienkāršotu reizināšanas, saskaitīšanas un aktivācijas slānu pieeju.

Ņemot vērā galvenos ierobežojumus, t. i. datu komunikācijas ierobežojumus vai resursu pieejamību, ir iespējams optimizēt tīklu, piešķirot atbilstošu resursu apjomu katram vienkāršo-

tajam slānim. Vēl viens svarīgs mākslīgo neironu tīklu (*ANN*) ieviešanas apsvērums ir datu tipa izvēle, jo peldošā komata datu tipi nodrošina augstu precizitāti un diapazonu, taču to ieviešana un konveijerizācija ir dārga. Turklāt literatūra [37], [38] ierosina, ka fiksētā punkta datu tipus var izmantot ar salīdzinoši nelielu precizitātes zudumu, it īpaši, ja apmācības procedūra *ANN* apzinās fiksētā punkta datu tipa izmantošanu.

Ir izstrādāts rīks, lai automatizētu *FFNN* implementāciju, kas par pamatu izmanto *FFNN* topoloģiju un ģenerē *C* kodu Xilinx *HLS* programmatūrai. Viens no rīka galvenajiem parametriem, kas kontrolē koda sintēzes procesu, ir konveijera posmiem pieņemamā maksimālā aizkave, kas tiek uzrādīta takts ciklos. Parametrs nodrošina minimālu resursu piešķiršanu, lai tomēr nodrošinātu norādīto slāņa aizkavi.

Izstrādātā rīka atvērtais kods ir pieejams tiešsaistē[4]. Uzģenerētie *FFNN* intelektuālā īpašuma (*IP*) kodoli ir testēti, lietojot divas komunikācijas saskarnes: atmiņas kartēta (*MM*) saskarne latentuma mērījumiem un straumēšanas (*ST*) saskarne latentuma un caurlaides spējas noteikšanai. *MM* saskarne paredz aktīvu procesora kontroli, savukārt *ST* saskarne izmanto *LogiCORE DMA IP* kodolus [39] un uzlabotās paplašinātās saskarnes (*AXI*) augstas veiktspējas saskarni [40].

4.1 tabula. Nelielu tīkla topoloģiju izmantoto resursu un veiktspējas salīdzinājums ar sekojošiem darbiem: [41], [42], [43], [44] (*LUT–Look-up-Table*, *FF–Flip Flop*, *DSP–Digital Signal Processor core*, *BRAM–Block Random Access Memory*)

Topoloģija	Saskarne	LUTs	FFs	DSPs	BRAM	Vēlamais inic. intervāls	Teorēt. inic. intervāls	Teorētiskā aizture	Aizture		Caurlaides spēja (Paķetes/s)	
									Literatūrā	Sasniegtais	Literatūrā	Sasniegtais
[41] 2-2-1	Memory-mapped	246 (0,46%)	118 (0,11%)	6 (2,73%)	3 (2,14%)	1	2	6	34 ns	555 ns $\sigma = 3,4$ ns	29 412 000	1 803 200
	Streaming	205 (0,39%)	135 (0,13%)	6 (2,73%)	3 (2,14%)	1	2	9		2,68 µs $\sigma = 37,5$ ns		49 272 000
[42] 2-4-1	Memory-mapped	980 (1,84%)	800 (0,75%)	48 (21,82%)	9 (6,43%)	1	2	10	44 ns	586 ns $\sigma = 3,4$ ns	2 272 700	1 705 600
	Streaming	983 (1,85%)	946 (2,92%)	48 (21,82%)	9 (6,43%)	1	2	12		2,69 µs $\sigma = 44$ ns		49 231 000
[43] 4-8-3-3	Memory-mapped	1304 (2,45%)	912 (0,86%)	0 (0,0%)	3,5 (2,5%)	1	4	12	1,16 µs	620 ns $\sigma = 8,7$ ns	862 070	1 612 400
	Streaming	1356 (2,55%)	1028 (0,97%)	0 (0,0%)	3.5 (2,5%)	1	4	19		2,78 µs $\sigma = 91$ ns		24 796 000
[44] 1-5-1	Memory-mapped	257 (0,5%)	78 (0,1%)	0 (0,0%)	1,5 (1,1%)	1	3	5	683 ns	578 ns $\sigma = 9,5$ ns	1 463 100	1 730 100
	Streaming	257 (0,5%)	81 (0,1%)	0 (0,0%)	1,5 (1,1%)	1	3	7		2,68 µs $\sigma = 35$ ns		33 005 000

[4] http://git.edi.lv/rihards.novickis/generation_tool_hls_c_fully_connected_feed_forward_neural_network.

4.2 tabula. Implementācijas izmantoto resursu un veiktspējas salīdzinājums ar [45] (Teorētiskais II = 100, Teorētiskā aizture = 987). (*LUT–Look-up-Table*, *FF–Flip Flop*, *DSP–Digital Signal Processor core*, *BRAM–Block Random Access Memory*)

Implementācija	LUTs	FFs	DSPs	BRAM	Aizture	Caurlaide (Paķetes/s)
[45]A	2232 (4,2%)	1210 (1,1%)	2 (0,9%)	-	33,1 ms	30,2
[45]B	3306 (6,2%)	1326 (1,3%)	4 (1,8%)	-	24,7 ms	40,5
[45]C	41 297 (77,6%)	33,395 (31,4%)	33 (15,0%)	-	5,7 ms	175,4
[45]D	51 028 (95,9%)	35 655 (33,5%)	65 (29,5%)	-	3,5 ms	285,7
Kartēta implementācija	30 197 (56,8%)	55 231 (51,9%)	122 (55,5%)	6 (4,3%)	10,4 µs σ = 0,011	96 435
Straumēta implementācija	31 246 (58,7%)	56 067 (52,7%)	122 (55,5%)	6 (4,3%)	12,5 µs σ = 0,027	997 852

Visi testi tika veikti, lietojot *Xilinx Zynq ZC702 SoC* izstrādes rīku, izmantojot bez-operētājsistēmas programmatūru ar *FPGA* loģikas takts signāla frekvenci 100 MHz. Laika mērījumi veikti, lietojot procesora *Cortex-A9* kodola noklausīšanās kontroles bloka (*SCU*) taimeri. Iegūtie testu rezultāti un detalizēts salīdzinājums apkopoti 4.1 un 4.2 tabulā.

4.1 tabulā apkopotie rezultāti liecina, ka risinājumiem, kas balstīti *ST* saskarnē, caurlaides spēja tuvojas teorētiski noteiktajiem un to aizture ir aptuveni 2,7 μs.

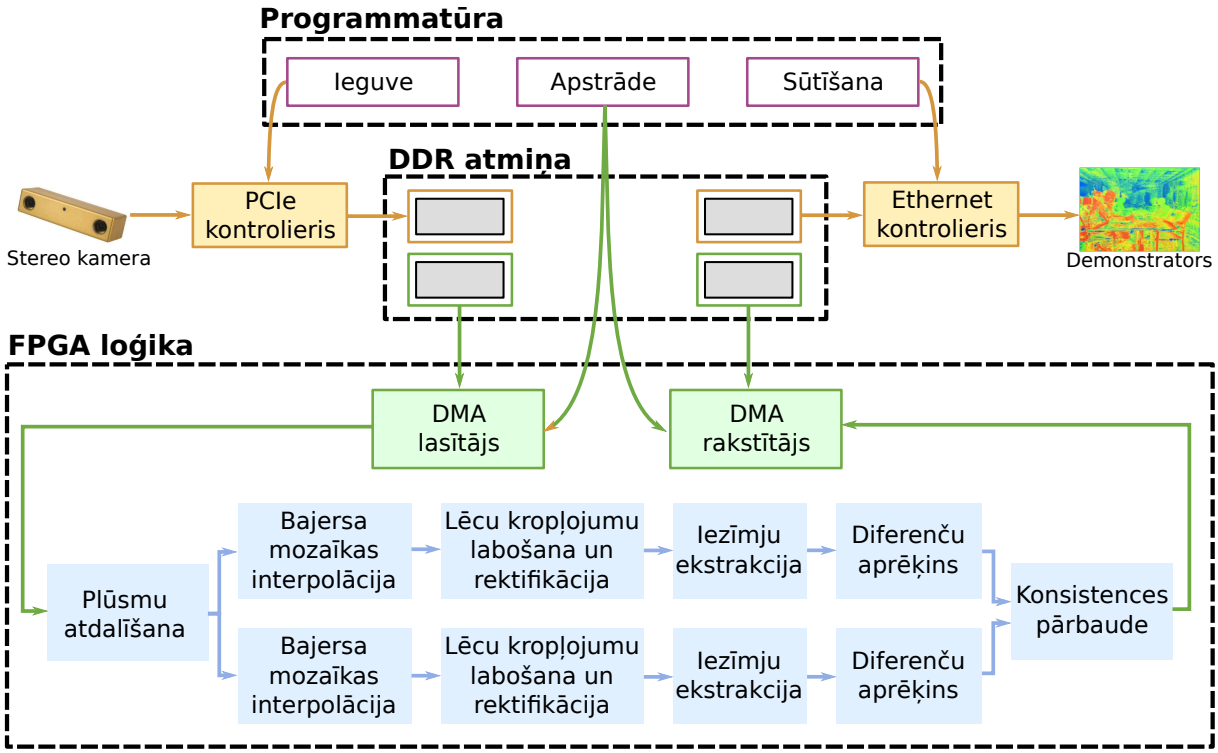
Kamēr topoloģija [45] realizēta, izmantojot hiperboliskā tangensa funkciju, balstās uz *Xilinx LOGICore IP* kodoliem, izstrādātais risinājums pārspēj jebkuru no publicētajām versijām, tādējādi liecinot par risinājuma derīgumu peldošā punkta implementācijām. Iemesls šādai iespaidīgai veikstpējas atšķirībai ir pieeju dažādie mērķi. Publikācijas [45] autori par prioritāti nosaka tīkla reāllaika rekonfigurāciju, savukārt piedāvātais risinājums cenšas sasniegt maksimāli iespējamo tīkla caurlaides spēju.

Izstrādātais risinājums tiek izmantots virtuālo sensoru lietojumos, un dažas no konfigurācijām ir sasniegušas iespaidīgus 2 miljonus novērtējumu sekundē, kas pārspēj veiktspēju, norādītu oriģinālajā rakstā [3].

4.2. Heterogēna sistēmas arhitektūra stereo attēlu apstrādei

Darba gaitā izstrādāta atbilstošo punktu meklēšanas sistēma, balstīta *HSoC* tehnoloģijā – izaicinājums, kas ietver diapazonu dažādu izstrādes abstrakciju, t. i. digitālo shēmu projektēšana, iekščipa komunikācija, *Linux* dziņu un programmatūras izstrāde, sistēmas arhitektūra. Izstrādātā risinājuma funkcionālā arhitektūra parādīta redzama 4.2. attēlā.

Procesora (programmatūras) daļa nodrošina vispārējo sistēmas vadību un sakarus ar bezsaistes aparatūru (stereo kameru lietojot *PCIe* un demonstrēšanas sistēmu lietojot *Ethernet*), izmantojot *Linux* operētājsistēmai pieejamo programmatūru.



4.2. att. Izstrādātā stereoredzes risinājuma funkcionālā arhitektūra un sadalījums starp heterogēnās vienčipa sistēmas apstrādes paradigmām un komponentēm.

Izstrādātais savstarpējās saziņas mehānisms izmanto kopīgu, koherentu atmiņu, jo tas dod iespēju citiem komunikācijas kopnes pārvaldītājiem (ne procesoram) veikt datu pārsūtīšanu; tomēr vēl aizvien pastāv programmatūras komponenšu sinhronizācijas problēma. Šis izaicinājums ir atrisināts, izmantojot divu buferu metodi [36].

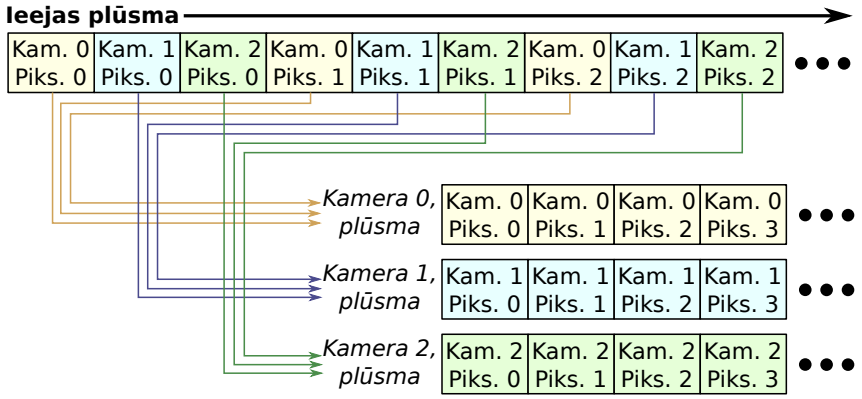
Divu buferu metode izstrādātajā sistēmā izmantota divas reizes – starp iegūšanas–apstrādes un apstrādes–pārsūtīšanas komponentēm. Kopīgotie atmiņas apgabali nodrošina programmatūras komponenšu vienlaikus izpildi, piemēram, kamēr iegūšanas pavediens ieraksta iegūto attē-

lu vienā no atmiņas apgabaliem, apstrādes pavediens izmanto otru apgabalu, lai nokonfigurētu *DMA* transakcijas un pārsūtītu datus uz/no paātrinātājiem. Tāds pats mehānisms nodrošina saziņu starp apstrādes un pārsūtīšanas komponentēm; tādējādi *SoC* vienlaikus veic attēlu iegūšanu, attēlu apstrādi un rezultējošā attēla pārsūtīšanu, kas izpaužas kā augsta līmeņa konveijerizācija.

4.3. Stereoattēlu apstrāde

4.3.1. Ieejas attēlu plūsmas sadalīšana

Pirmā procedūra, kas tiek īstenota stereoredzes aloritmu virknē, ir ieejas plūsmas sadalīšana, kas sadala ievades attēlu plūsmu no *Bumblebee* kameras vairākās nodalītās izejas plūsmās. 4.3. attēlā redzama šāda ieejas plūsma, kur izmantotā kameras sistēma iekļauj trīs kameras.



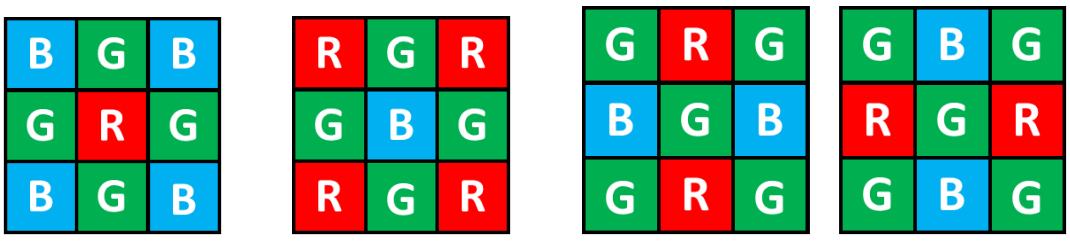
4.3. att. *Bumblebee* kameras savītā ieejas plūsma.

Izaicinājums ir atrisināts digitālajā loģikā, izmantojot plūsmas platuma adapteri, kas konvertē ievades plūsmas platumu uz 24 bitiem, t. i. iegūstot trīs vienlaikus pieejamus pikseļus, kur katrs pikselis aprakstīts, izmantojot vienu baitu. Jo īpaši gadījumos, kad kamera pievienota tieši *FPGA* loģikai, nepieciešami papildu pirmais iekšā, pirmais ārā (*FIFO*) buferi, jo attēla straumēšanas komponente kamerā pieņem, ka saņēmēja komponente vienmēr ir gatava pieņemt kārtējo datu paku.

4.3.2. Bajersa mozaīkas interpolācija un RGB intensitātes konvertācija

Mūsdienu kameras izmanto krāsu filtru masīvu (*CFA*), kur dažādus sensoru pikseļus pārklāj krāsu filtri. Visbiežāk mūsdienu kamerās filtri tiek izvietoti Bajersa mozaīkas veidā [46]. Iztrūkstošās pikseļu *RGB* krāsu vērtības iegūst, izmantojot interpolāciju no reģionā pieejamajām vērtībām.

Lai gan ir izstrādāta virkne dažādu rekonstrukcijas metožu [47], izstrādātā sistēma izmanto vienkāršus lineāras un bilineāras interpolācijas algoritmus. 4.4. attēlā redzamas dažādas mozaīkas, ko rekonstrukcijas algoritmam jāspēj interpolēt.



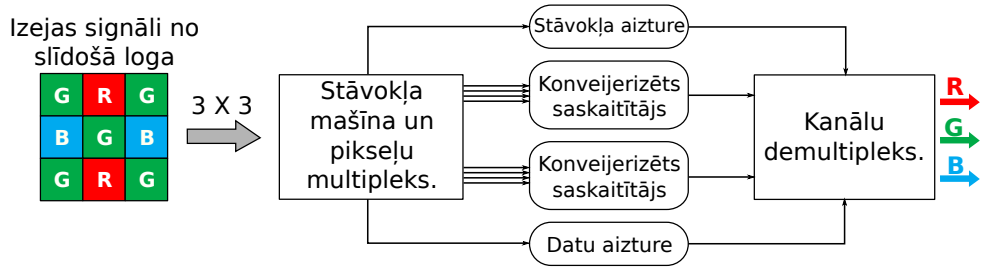
(a) Sarkanā vērtība ir zināma, zaļā un zilā pikseļu vērtības tiek interpolētas no blakus pikseļiem.

(b) Zilā vērtība ir zināma, sarkanā un zaļā pikseļu vērtības tiek interpolētas no blakus pikseļiem.

(c) Zaļā vērtība ir zināma, sarkanā un zilā pikseļu vērtības tiek interpolētas no blakus pikseļiem.

4.4. att. Bajersa mozaīkas reģioni, ko izmanto interpolācijas algoritms.

Veicot izpēti, kļūst redzams, ka jebkurā noteiktā sistēmas konfigurācijā nepieciešams rekonstruēt divas krāsu vērtības. Turklāt lietotais interpolācijas algoritms pārbauda 3×3 lielu apgabalu, izmantojot slīdošā loga pieeju, turklāt jebkurā rekonstrukcijas procesa taktī tiek izmantotas četras (divi un divi) vai astoņas (četri un četri) vērtības. 4.5. attēlā redzama izstrādātās digitālās shēmas augsta līmeņa struktūra.



4.5. att. Bajersa mozaīkas interpolācijas shēmas augsta līmeņa struktūra.

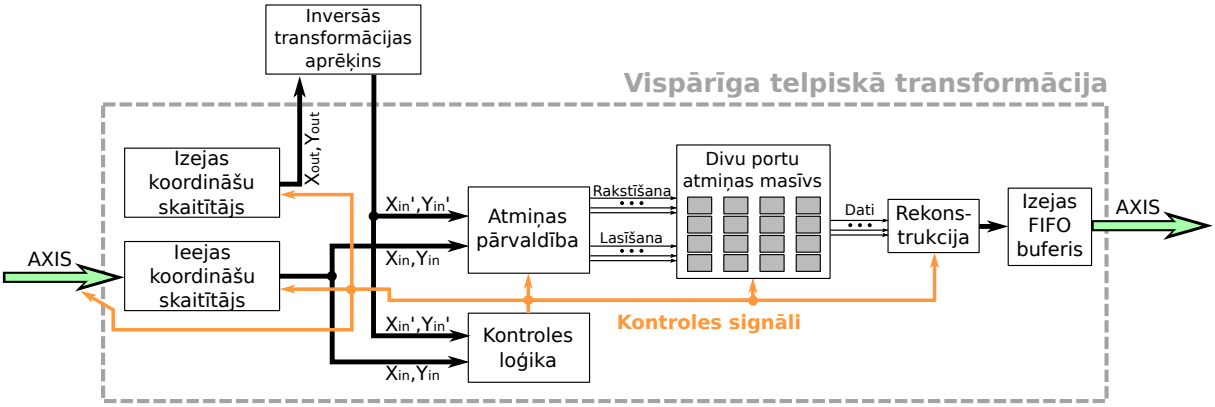
Signāla rekonstrukcijas loģikas ieejas dati tiek padoti no slīdošā loga bloka, ko ir ērti implementēt digitālajā loģikā [48]. Pilnībā konveijerizētais signāla rekonstrukcijas bloks saņem datus kā redzams 4.4. attēlā. Interpolācijas bloka struktūra ietver stāvokļu ģeneratoru, kur kats stāvoklis atbilst paraugiem, kuri ilustrēti Att. 4.4.. Bloks sakārto šos ieejas datus konveijerizētajiem saskaitītājiem, savukārt centra pikselis tiek vienkārši aizturēts, jo tā vērtību nav nepieciešams rekonstruēt. Uzģenerētais stāvoklis arī tiek aizturēts, jo tas ir nepieciešams demultipleksēšanas blokam izejas vērtību sakārtošanai (demultipleksēšanai).

Signāla rekonstrukcijas bloks ietver arī konfigurējamu *RGB* intensitātes attēlu konvertācijas funkcionalitāti, jo pelēktoņu attēlu izmantošana trīskārši samazina nepieciešamo aprēķinu darbību skaitu, savukārt, neieviešot būtiskus precizitātes zudumus, piemēram, attēla robežu noteikšanas precizitāte var samazināties mazāk kā par 10 % [49], [50]. Tāpēc tiek izmantota aparatūrai draudzīga spilgtuma (*lightness*) krāsu telpas transformācijas metode [51].

4.3.3. Pieeja telpiskai attēlu transformācijai

Būtiska jebkura attēlu priekšapstrādes daļa ir algoritmi pikseļu telpiskai transformēšanai vai kartēšanai, t. i. lēcu kropļojuma korekcija, attēlu rektifikācija, digitālā tālummaiņa. Šādu uzdevumu izpilde digitālajā loģikā ir saistīta ar lielu sarežģītību, jo atmiņas piekļuves modeļi ir samērā globāli. Konkrētāk, visa attēla saglabāšana *OCRAM* programmējamo loģikas masīvu atmiņā ir vai nu neiespējama (*FPGA* mikroshēmām bieži vien ir mikroshēmas atmiņa, kas ir mazāka par 1 MB), vai dārga (viena divu portu statiskā brīvpiekļuves atmiņas (*SRAM*) šūna mikroshēmas atmiņā izmanto astoņus komplementārās struktūras metāls–oksīds–pusvadītājs (*CMOS*) tranzistorus).

4.6. attēlā redzama telpiskās attēlu transformācijas paātrinātāja bloka funkcionālā arhitektūra, kas ietver ieejas un izejas koordinātu skaitītājus, divu portu atmiņu masīvu, datu ierakstīšanas un nolasīšanas komponentes, izejas datu salāgošanas loģiku, kontroles signālu nodrošināšanas komponenti un ārēju inversās transformācijas aprēķinu loģiku.



4.6. att. Pieeja tepliskās transformācijas pilnībā konveijerizētas digitālās shēmas implementācijai.

Jebkuru telpisku attēlu transformāciju var aprakstīt kā konkrētu ieejas punktu atbilstību izejas punktiem:

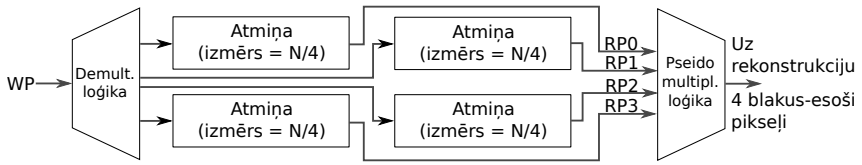
$$x_{out}, y_{out} = f(x_{in}, y_{in}), \tag{4.1}$$

kur x_{in}, y_{in} un x_{out}, y_{out} ir ieejas/izejas attēlu punktu koordinātes un f ir kāda patvaļīga funkcija, ko bieži vien lieneāru transformāciju gadījumā apraksta kā matricu. Tādos gadījumos ieejas koordināšu secīga palielināšana izraisa "lēkāšanu" izejas koordinātēs. Piedāvātajam risinājumam nepieciešams pretējais: inversā transformācija, pēc būtības meklējot ieejas attēla koordinātes secīgi pieaugošām izejas koordinātēm, t. i.:

$$x_{in}, y_{in} = f^{-1}(x_{out}, y_{out}). \tag{4.2}$$

Risinājums izmanto unikālu metodi signālu rekonstrukcijai, kas iespējo signāla rekonstrukciju no N-punktiem, vienlaikus izmantojot nelielus atmiņas resursus. Šī metode ir vispārināta jebkuram dimensiju skaitam. Nogurdinošais uzdevums ģenerēt adreses atsevišķām atmiņu masīva atmiņām ir uzticēts atmiņas ierakstīšanas un nolasīšanas blokiem.

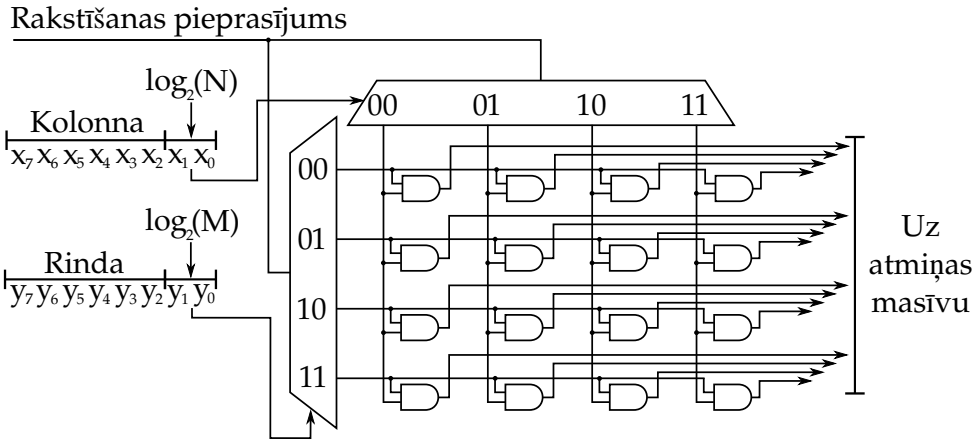
Viena no galvenajām prasībām pilnībā konveijerizētai telpiskajai attēlu transformācijai ar interpolāciju ir atmiņas buferēšana, kas iespējotu interpolāciju, vienlaikus dodot piekļuvi attēla blakus pikseļiem. Vispārpieņemtās pieejas var tikt optimizētas, pieņemot, ka interpolācijai nepieciešamie pikseļi vienmēr atrodas blakus. 4.7. attēlā redzams daudz efektīvāks risinājums, kur atmiņas izmērs ir samazināts un katrā atmiņā tiek saglabātas tikai nepieciešamās datu (pāra/nepāra) kolonnas un rindas.



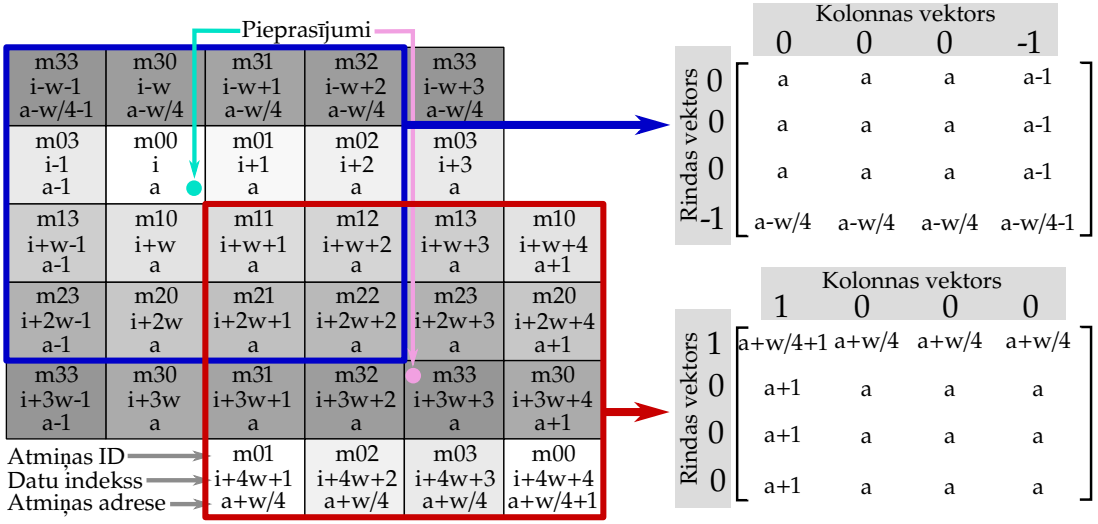
4.7. att. Optimizēts mehānisms datu piekļuvei četru punktu rekonstrukcijai ar četras reizes mazāku atmiņas izmantošanu.

Pieņemot, ka atmiņu matrica sastāv no M rindām un N kolonnām, pēdējie $log_2(M)$ biti no rindas signāla un $log_2(N)$ biti no kolonnas signāla kontrolē demultipleksēšanas loģiku rakstīšanas pieprasījuma signālam, kā redzams 4.8. attēlā. Visiem atmiņu portiem ir kopīgotas datu un adreses kopnes.

Datu nolasīšana ir daudz lielāks izaicinājums, jo ir nepieciešamība pēc dažādām adresēm katram atmiņu lasīšanas portam. Darbā formalizēts šo adrešu ģenerēšanas process, un tas tika attiecināts uz jebkuru dimensiju skaitu. 4.9. attēlā redzams, kā kombinācija no vertikālā un horizontālā mehānisma adrešu vektoriem tiek ģenerētas nepieciešamās nolases adreses visām atmiņām.



4.8. att. Piemērs rakstīšanas signāla demultipleksēšanai, pielietojot 4×4 atmiņu matricu.



4.9. att. Adrešu ģenerēšanas process 4×4 atmiņas masīvam.

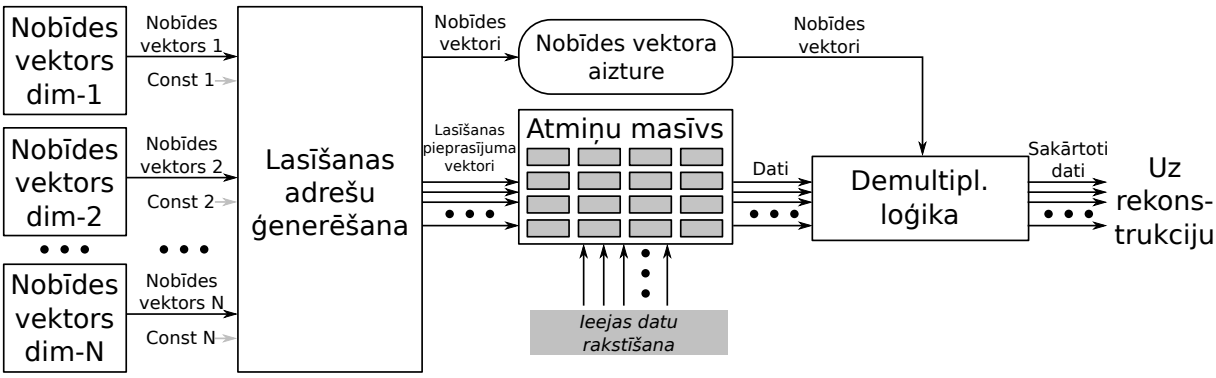
Visbeidzot, pieeja attiecināma uz jebkuru dimensiju skaitu. N dimensiju atmiņas masīvam adreses veidotu N dimensiju matricu $A \in E^N$, kur katru elementu var aprēķināt ar šādu izteiksmi:

$$A_{i_1 i_2 \cdots i_N} = \sum_{n=1}^{N} C_n S_n v_{o_{i_n}}, \tag{4.3}$$

kur S_n apzīmē nobīdes matricu, v_o – nobīdes vektoru, C_n – nobīdes konstanti, ko nosaka apstrādājamo datu konkrētās dimensijas izšķirtspēja.

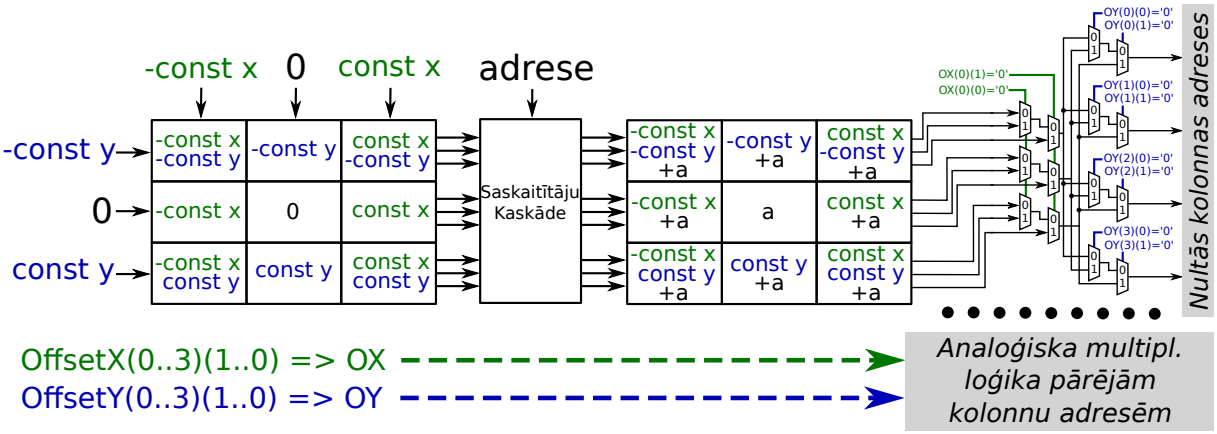
Izstrādāto matemātisko modeli var izmantot tālākai digitālo shēmu ģenerēšanai un izstrādei. 4.10. attēlā redzama vispārējo datu ieguves koncepcija un organizatoriskā struktūra visām nepieciešamajām komponentēm vispārīgam N dimensionālam gadījumam.

Viena no izstrādātā modeļa sarežģītākajām komponentēm ir nolasīšanas adrešu ģenerēšana, kas tiek realizēta atsevišķi atkarībā no dimensiju skaita, tomēr izstrādes procesā novērojamas



4.10. att. Koncepts vispārīgai adrešu vektoru ģenerēšanai N dimensionālam gadījumam.

sakarības. Piemēram, divu dimensiju nolasīšanas adrešu ģenerēšanas loģika ilustrēta 4.11. attēlā. Konceptuālā pieeja iekļauj trīs daļas: visu iespējamo nobīdes konstanšu aprēķins; visu iespējamo atmiņu adrešu aprēķins; adrešu multipleksēšana visām atmiņām.



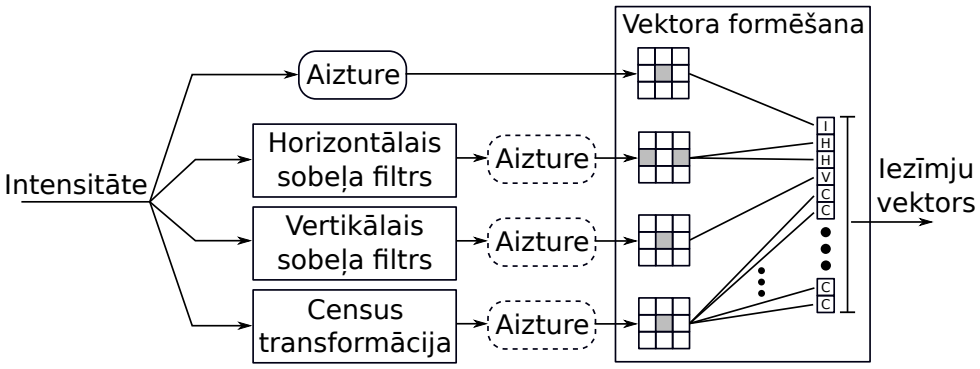
4.11. att. Shēmas koncepcija nolasīšanas adrešu aprēķinam.

Visbeidzot, visas iespējamās adreses ir pieejamas, un tās tiek novadītas uz atmiņu matricu.

4.3.4. Iezīmju ekstrakcija

Tālāk ir nepieciešams izšķirt iezīmes korespondences punktu meklēšanai. 4.12. attēlā redzams kā iezīmju ekstrakcijas process ir implementēts demonstrācijas sistēmā. Iezīmju ekstrakcijas bloks nodrošina četru iezīmju veidus: (1) pikseļu intensitāte; (2) horizontālā Sobeļa filtra vērtības blakus (pa labi un pa kreisi) pikseļos; (3) vertikālā Sobeļa filtra vērtība; (4) census transformācija 5×5 reģionam. Eksperimentēšanas nolūkā, *RTL* apraksts ir izstrādāts vispārīgs, dodot iespēju izmēģināt dažādas iezīmju konfigurācijas. Katras iezīmes ģenerācija iekļauj arī aiztures blokus, kuru ģenerācija atkarīga no attiecīgo iezīmju ekstrakcijas bloku aizturēm. Iezīmju ekstrakcija realizēta pilnīgi konveijerizētā veidā, izmantojot slīdošā loga pieeju. Visbeidzot,

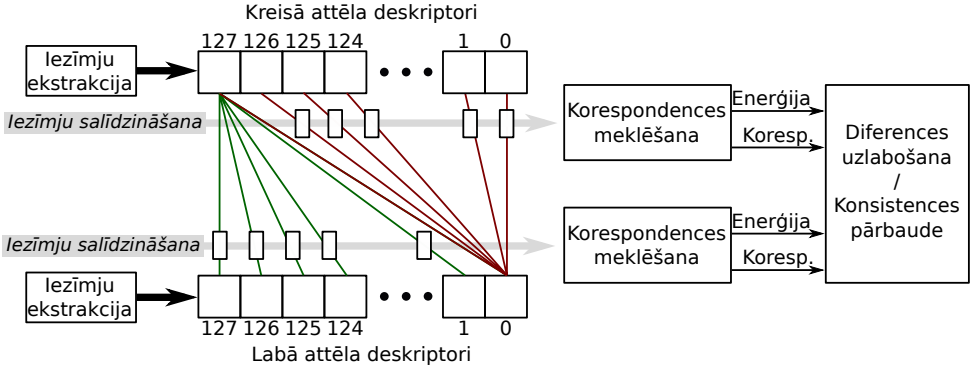
iezīmes tiek nokomplektētas vektorā, kas tālāk tiek padots punktu atbilstību meklēšanas komponentēm.



4.12. att. Izstrādātā iezīmju ekstraktora augsta līmeņa kompozīcija.

4.3.5. Atbilstību meklēšana

Atbilstību meklēšanas daļa patērē visvairāk digitālās loģikas resursu no visas stereo attēlu apstrādes sistēmas, jo pilnībā konveijerizētai implementācijai nepieciešams salīdzināt visus iezīmju vektorus vienlaikus. 4.13. attēlā redzama vispārējā koncepcija labā-kreisā un kreisā-labā attēlu punktu atbilstību meklēšanai, izmantojot 128 punktus. Izceltie iezīmju deskriptori tiek aizturēti, izmantojot sērijveida ieeja, paralēla izeja (*SIPO*) buferus, un tos saista iezīmju salīdzināšanas bloki. Salīdzināšanas rezultāti tālāk tiek novirzīti uz atbilstību meklēšanas komponentēm, kas identificē atbilstības ar vismazāko "enerģiju" (pretēji pārliecībai).

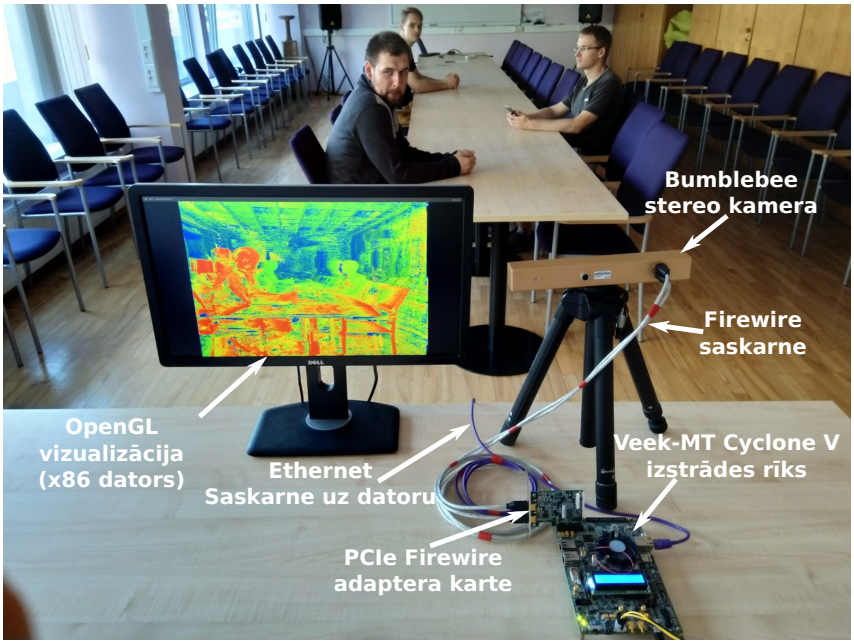


4.13. att. Digitālā loģika atbilstību meklēšanai un aprēķiniem.

4.4. Demonstrācijas sistēma

Demonstrācijas sistēma ir izstrādāta pēc iepriekš noteiktās sistēmas arhitektūras. 4.14. attēlā redzama demonstrācijas sistēma darbībā, kas sastāv no *Bumblebee BBX3* stereo kameras, *Te-*

rasic VEEK-MT Cyclone V heterogēna *SoC* izstrādes rīka un *OpenGL* bāzēta demonstrācijas programmatūras datora. Izstrādātā tehnoloģija domāta augstas veiktspējas aprēķinu veikšanai ar zemu enerģijas patēriņu (< 10 W) un izmaksām. *SoC* sistēmas programmatūra nodrošina attēlu iegūšanu, izmantojot *PCIe* kopni, *FPGA* paātrinātāju kontroli un apstrādāto attēlu nosūtīšanu demonstrācijas datoram, izmantojot *Ethernet*. Visi ar diferences kartes izveidošanu saistītie aprēķini ir veikti *FPGA* loģikā (shēmas aprakstītas, izmantojot *VHDL*), ieskaitot Bajersa mozaīkas interpolāciju, lēcu kropļojumu labošanu, rektifikāciju, iezīmju ekstrakciju un diferences kartes aprēķināšanu.



4.14. att. Stereo redzes demonstrācijas sistēmas darbība.

Šī promocijas darba izstrādes laikā *Intel* izstrādāja *ASIC* bāzētu stereo attēlu apstrādes sistēmu, kas strauji pārņēmusi robotikas lietojumus. Neskatoties uz to, balstoties iegūto punktu skaitā, izstrādāto tehnoloģiju varētu izmantot dziļuma noteikšanas aplikācijām sistēmās ar zemu enerģijas patēriņu, piemēram, autonomos dronos (lai izvairītos no šķēršļiem) un dažos lietu interneta lietojumos.

5. SECINĀJUMI

Promocijas darbs aplūko sakarības starp datorizētu uztveri un heterogēnām vienčipa tehnoloģijām, kuru sarežģītība vēl aizvien pieaug. Īpaša uzmanība pievērsta aparatūras un programmatūras koparhitektūrai, stereo redzes algoritmiem un mākslīgo neironu tīklu algoritmu realizācijai, kā arī reāllaika apsvērumiem. Šī darba galvenais mērķis ir izstrādāt un uzlabot metodes datorredzes apstrādes realizācijai un *HSoC* tehnoloģiju lietošanai. Lai sasniegtu noteikto mērķi, tika definēti pieci uzdevumi.

1. Identificēt metodes *RTL* un programmatūrā bāzētu skaitļošanas paradigmu savstarpējā papildināšanā. Šis uzdevums paveikts 1. nodaļā un 3.1. apakšnodaļā. Literatūrā identificētas dažādu skaitļošanas paradigmu galvenās iezīmes, kā arī analizētas to priekšrocības. Literatūras apskats veicināja arhitektūras izstrādi, kuras pamatā ir kontroles uzdevumu uzticēšana procesoram, savukārt skaitļošanas uzdevumus veic *FPGA*. Abas skaitļošanas ierīces kā galveno datu apmaiņas mehānismu lieto atmiņu, kas dod iespēju panākt maksimālo sistēmas veiktspēju.

2. Izstrādāt programmatūras arhitektūras un rīkus *HSoC* tehnoloģijas lietošanai. Šis uzdevums paveikts 3. nodaļā, izstrādājot arhitektūru, kurā *FPGA* veic datu pārraidi, savukārt *SoC* izpilda *Linux* operētājsistēmas kodu, ieskaitot programmatūras dziņus un bibliotēkas koherentas un nepārtrauktas atmiņas pārvaldībai, kā arī atmiņas tiešpiekļuves bloku kontrolei. Reāllaika apstrādei izveidota *AMP* apakšsistēma, kas ļauj izmantot *Linux* operētājsistēmas priekšrocības, kritiskos uzdevumus uzticot nošķirtam procesora kodolam. Izstrādātais risinājums apvieno atvērtā koda programmatūras plašo klāstu ar reāllaika vadības iespējām, izmantojot *Linux* dziņa saskarni. Dzinis nodrošina reāllaika programmas un tās konfigurācijas iestatīšanu, *AMP* kodola kontroli, kā arī dod iespējas reāllaika veikstpējas raksturojuma apkopošanai izmantojot *sysfs* saskarni. Visbeidzot, ir izstrādāti rīki – *compage* un *icom* – sistēmas arhitektūru ieviešanai, kas balstās programmatūras komponentēs. Rīki ļauj lietot ”tāfeles” (*blackboard*) programmēšanas modeli sistēmās ar ierobežotiem resursiem, kā arī dod iespējas lietotājam dažādu programmatūras komponenšu konfigurēšanai, replicēšanai un savstarpējas komunikācijas realizācijai.

3. Izstrādāt heterogēnu pieeju attēlu apstrādes konveijeru (*pipeline*) implementācijai. Šis uzdevums paveikts 3.1. un 4.2. apakšnodaļās, realizējot heterogēnu arhitektūru attēlu apstrādes algoritmiem, par pamatu izmantojot stereo redzes algoritmu virkni. Izstrādātajā arhitektūrā programmatūra iegūst attēlus no *Bumblebee* kameras (izmantojot PCIe saskarni) un ieraksta tos koherentajā atmiņā, pārrauga *FPGA* implementēto algoritmu izpildi un nodod iegūtos rezultātus demonstratora sistēmai, izmantojot *Ethernet* saskarni. Pēc būtības izstrādātajā sistēmā visas komponentes – programmatūra un aparatūra – veic apstrādi vienlaikus, tādējādi panākot paralēlu izpildi visā heterogēnajā aparatūrā.

4. Implementēt un veikt izstrādāto rīku un algoritmu eksperimentālos pētījumus. Šis uz-

devums paveikts 4. nodaļā. Pirmkārt, *FPGA* aparatūrā tika realizēti mākslīgie neironu tīkli, lai pārliecinātos par iespējamību algoritmus pilnībā konveijerizēt. Izstrādāto pieeju komplementē programmatūras rīks *FFNN* topoloģiju transformēšanai silīcija *IP* kodolos, turklāt kodoli var izmantot gan straumēšanas gan arī atmiņas kartētas saskarnes. Sasniegtie rezultāti nevien pārspēj citus literatūrā aprakstītos risinājumus, bet arī uzrāda izstrādātās metodes piemērojamību virtuālo sensoru realizēšanai, piemēram, izmantot elektrisko transportlīdzekļu griezes momenta paredzēšanai.

Turklāt ir izstrādāta virkne pilnībā konvejerizētu paātrinātāju: kombinēto datu plūsmu izdalīšana; Bajersa mozaīkas interpolācija, attēlu telpiskā transformācija (ieskaitot lēcu kropļojumu korekciju un attēlu rektifikāciju); attēlu iezīmju izgūšana un saistīto punktu (korespondenču) meklēšana. Viens no galvenajiem ieguldījumiem ir attēlu transformācijas matemātiskā modeļa vispārināšana, kas dod iespēju pilnībā konveijerizētā veidā piekļūt blakus esošiem pikseļiem, vienlaikus saglabājot minimālu nepieciešamību pēc atmiņas. Lai gan izstrādātais risinājums tiek izmantots attēlu apstrādei, vispārinātais modelis nodrošina iespēju sintezēt shēmas jebkuram datu dimensiju skaitam, līdz ar to paveras iespēja rekonstruēt (interpolēt) cita veida datus, piemēram, volumetriskos. Visbeidzot, izstrādātie paātrinātāji kalpo par pamatu stereoredzes demonstratora izstrādei.

5. veikt secinājumus, ņemot vērā iegūtos rezultātus. Noslēdzot darbu, tiek secināts, ka stereoredzes algoritma realizācija, izmantojot heterogēnu iegultu sistēmu ir iespējama, turklāt *SoC* tehnoloģijas nodrošina unikālu programmatūras un aparatūras kombināciju, kas paver iespējas panākt nebijušu energoefektivitāti, vienlaicīgi nodrošinot iespēju paplašināt risinājuma funkcionalitāti ar programmatūru. Izstrādātā pieeja attēlu apstrādei heterogēnā *SoC* tehnoloģijā ir realizējama arī citiem lietojumiem, piemēram, lielu (> 50 MP) attēlu apstrādei.

Par sasniegto rezultātu būtiskumu liecina arī vairāki iesāktie un pabeigtie starptautiskie pētniecības projekti. Piemēram, sistēmas arhitektūru ieviešanas ietvari tiek izmantoti, lai izstrādātu mākslīgajā intelektā balstītu uztveres sistēmu transportlīdzekļiem (*PRYSTINE*, Nr. 783190, AI4CSM, Nr. 101007326) un vadības programmatūru autonomiem droniem (*COMP4DRONES*, Nr. 826610), savukārt izstrādātā attēlu apstrādes plūsma tiek lietota infrasarkano attēlu priekšapstrādes algoritmu realizācijā (*APPLAUSE*, Nr. 826588).

Turklāt šī promocijas darba rezultāti kalpo par pamatu komercializācijas aktivitātēm (*SilHouse*, Nr. KC-PI-2020/12), kurā tiek izstrādāts ietvars, kas apvieno vairākus paātrinātājus un nodrošina ērtu to lietojamību industrijā bez iepriekšējām zināšanām par digitālo shēmu projektēšanu.

IZMANTOTĀS LITERATŪRAS SARAKSTS

[1] Moore, G. E. Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter* Vol.11. Nr.3. 33—35 p.

[2] Novickis, R. un Greitāns, M. FPGA Master based on chip communications architecture for Cyclone V SoC running Linux// *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*. ISSN: 2576-3555. Apr. 403—408 p. DOI: 10.1109/CoDIT.2018.8394842.

[3] Dendaluce Jahnke, M., Cosco, F., Novickis, R., Pérez Rastelli, J. un Gomez-Garay, V. Efficient Neural Network Implementations on Parallel Embedded Platforms Applied to Real-Time Torque-Vectoring Optimization Using Predictions for Multi-Motor Electric Vehicles en *Electronics* febr. Vol.8. Nr.2. 250 p. Febr. ISSN: 2079-9292. DOI: 10.3390/electronics8020250. / URL - http://www.mdpi.com/2079-9292/8/2/250 (aplūkots 25.07.2020.).

[4] Setka, V., Jezek, O. un Novickis, R. Modular Signal Processing Unit for Motion Control Applications Based on System-on-Chip with FPGA en// *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*.. Zaragoza, Spain: IEEE, sept. 857—863 p. ISBN: 978-1-72810-303-7. DOI: 10.1109/ETFA.2019.8869121. / URL - https://ieeexplore.ieee.org/document/8869121/ (aplūkots 25.07.2020.).

[5] Novickis, R., Justs, D. J., Ozols, K. un Greitāns, M. An Approach of Feed-Forward Neural Network Throughput-Optimized Implementation in FPGA *Electronics* dec. Vol.9. Nr.12. 2193 p. Dec. DOI: 10.3390/electronics9122193. / URL - https://www.mdpi.com/2079-9292/9/12/2193.

[6] Novickis, R., Levinskis, A., Kadikis, R., Fescenko, V. un Ozols, K. Functional Architecture for Autonomous Driving and its Implementation *2020 17th Biennial Baltic Electronics Conference (BEC)* okt. okt. DOI: 10.1109/bec49624.2020.9276943. / URL - https://ieeexplore.ieee.org/abstract/document/9276943.

[7] Druml, N., Debaillie, B., Anghel, A. u. c. Programmable Systems for Intelligence in Automobiles (PRYSTINE) Technical Progress after Year 2 *2020 23rd Euromicro Conference on Digital System Design (DSD)* aug. aug. DOI: 10.1109/dsd51259.2020.00065. / URL - https://ieeexplore.ieee.org/document/9217654.

[8] Druml, N., Ryabokon, A., Schorn, R. u. c. Programmable Systems for Intelligence in Automobiles (PRYSTINE): Final results after Year 3// *2021 24th Euromicro Conference on Digital System Design (DSD)*.. 268—277 p. DOI: 10.1109/DSD53832.2021.00049.

[9] Shen, J. P. un Lipasti, M. H. L. Modern processor design: Fundamentals of superscalar processors. en. Waveland Press, Inc., ISBN: 1-3786-0783-1 978-1-4786-0783-0.

[10] Patt, Y. N. un Patel, S. J. Introduction to computing systems: from bits and Gates to C and beyond. en. Boston: McGraw-Hill Higher Education, OCLC: 145555083 ISBN: 978-0-07-124501-2.

[11] Kuon, I., Tessier, R. un Rose, J. FPGA Architecture: Survey and Challenges en *Foundations and Trends® in Electronic Design Automation* Vol.2. Nr.2. 135—253 p. ISSN: 1551-3939, 1551-3947. DOI: 10.1561/1000000005. / URL - http://www.nowpublishers.com/article/Details/EDA-005 (aplūkots 26.08.2017.).

[12] Chu, P. P. RTL hardware design using VHDL.. John Wiley & Sons, ISBN: 9780471720928. DOI: 10.1002/0471786411.

[13] Insight Technologies „State of Linux in the public cloud for enterprises" angļu Red Hat Solution overview. / URL - https://www.redhat.com/cms/managed-files/cl-state-of-linux-in-public-cloud-for-enterprises-f11154kc-201802-en_0.pdf (aplūkots 10.08.2020.).

[14] Corbet, J., Rubini, A. un Kroah-Hartman, G. Linux Device Drivers. Third. O'Reilly, dec.

[15] Szeliski, R. Multiple view geometry in computer vision. en. OCLC: 171123855 ISBN: 978-0-511-18711-7 978-0-511-18618-9 978-0-511-18895-4 978-0-511-18535-9 978-0-511-18451-2 978-0-511-81168-5 978-1-280-45812-5. / URL - http://dx.doi.org/10.1017/CBO9780511811685 (aplūkots 12.03.2019.).

[16] ——, Computer Vision: Algorithms and Applications en 979 p. ISSN: 1868-0941.

[17] Marr, D. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. en Nr.2.. W. H. Freeman un Company, ISBN: 0-7167-1567.

[18] Scharstein, D., Szeliski, R. un Zabih, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms// *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*.. Dec. 131—140 p. DOI: 10.1109/SMBV.2001.988771.

[19] Haykin, S. Neural Networks: A Comprehensive Foundation. 2nd. Upper Saddle River, NJ, USA: Prentice Hall PTR, ISBN: 978-0-13-273350-2.

[20] Fine, T. L. Feedforward Neural Network Methodology. 1st. Cornell University, Ithaca, NY, USA: Springer-Verlag New York, ISBN: 978-0-387-98745-3.

[21] Chakradhar, S., Sankaradas, M., Jakkula, V. un Cadambi, S. A dynamically configurable coprocessor for convolutional neural networks// *ACM SIGARCH Computer Architecture News*. Vol.38.. ACM, 247—257 p. / URL - http://dl.acm.org/citation.cfm?id=1815993 (aplūkots 06.09.2017.).

[22] Suda, N., Chandra, V., Dasika, G. u. c. Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks en// *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate.*. ACM Press, 16—25 p. ISBN: 978-1-4503-3856-1. DOI: 10.1145/2847263.2847276. / URL - http://dl.acm.org/citation.cfm?doid=2847263.2847276 (aplūkots 06.09.2017.).

[23] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B. un Cong, J. Optimizing fpga-based accelerator design for deep convolutional neural networks// *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*.. ACM, 161—170 p. / URL - http://dl.acm.org/citation.cfm?id=2689060 (aplūkots 06.09.2017.).

[24] Foumani, S. N. A. An FPGA Accelerated Method for Training Feed-forward Neural Networks Using Alternating Direction Method of Multipliers and LSMR. maģ. darbs Imperial College London, Department of Computing sept.

[25] Blott, M., Preußer, T. B., Fraser, N. J. u. c. FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks *ACM Transactions on Reconfigurable Technology and Systems* dec. dec. DOI: 10.1145/3242897.

[26] Guan, Y., Liang, H., Xu, N. u. c. FP-DNN: An Automated Framework for Mapping Deep Neural Networks onto FPGAs with RTL-HLS Hybrid Templates// *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. ISSN: 978-1-5386-4038-8. Jūl. DOI: 10.1109/FCCM.2017.25.

[27] Khan, A., Sohail, A., Zahoora, U. un Qureshi, A. S. A survey of the recent architectures of deep convolutional neural networks en *Artif Intell Rev* dec. Vol.53. Nr.8. 5455—5516 p. Dec. ISSN: 1573-7462. DOI: 10.1007/s10462-020-09825-6. / URL - https://doi.org/10.1007/s10462-020-09825-6 (aplūkots 31.10.2020.).

[28] Shalf, J. un Leland, R. Computing beyond Moore's Law *Computer* dec. Vol.48. 14—23 p. Dec. DOI: 10.1109/MC.2015.374.

[29] Sadri, M., Weis, C., Wehn, N. un Benini, L. Energy and Performance Exploration of Accelerator Coherency Port Using Xilinx ZYNQ// *FPGAworld '13 Proceedings of the 10th FPGAworld Conference.*. Sept.

[30] Molanes, R. F., Salgado, F., Fariña, J. un Rodríguez-Andina, J. J. Characterization of FPGA-master ARM communication delays in Cyclone V devices// *41st Annual Conference of the IEEE Industrial Electronics Society.*. Nov. 4229—4234 p.

[31] Vogel, P., Marongiu, A. un Benini, L. An Evaluation of Memory Sharing Performance for Heterogeneous Embedded SoCs with Many-Core Accelerators// *COSMIC '15 Proceedings of the 2015 International Workshop on Code Optimisation for Multi and Many Cores.*. Febr.

[32] Altera corp. *Cyclone V Hard Processor System Technical Reference Manual* - okt.

[33] Šetka, V., Ježek, O. un Novickis, R. Modular Signal Processing Unit for Motion Control Applications Based on System-on-Chip with FPGA// *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).*. 857—863 p. DOI: 10.1109/ETFA.2019.8869121.

[34] Sangmin Chon „What it Takes to do Efficient and Cost-Effective Real-Time Control with a Single Microcontroller The C2000™Advantage" angļu Texas Instruments White paper. / URL - https://www.ti.com/lit/wp/spry157/spry157.pdf?ts=1637148859226\&ref_url=https\%253A\%252F\%252Fwww.google.com\%252F (aplūkots 17.11.2021.).

[35] F. Buschmann, K. Henney un D. C. Schmidt Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing.. Chichester: Wiley, Vol.Volume 4.

[36] NVIDIA Corporation NVIDIA GPU Programming Guide en 80 p.

[37] Fu, Y., Wu, E., Sirasao, A., Attia, S., Khan, K. un Wittig, R. Deep Learning with INT8 Optimization on Xilinx Devices en 11 p.

[38] Dettmers, T. 8-Bit Approximations for Parallelism in Deep Learning en *arXiv:1511.04561 [cs]* nov. nov. arXiv: 1511.04561. / URL - http://arxiv.org/abs/1511.04561 (aplūkots 23.10.2018.).

[39] AXI DMA v7.1, LogiCORE IP Product Guide apr. 95 p. Apr. / URL - https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf.

[40] Zynq-7000 All Programmable SoC Technical Reference Manual (UG585) en 1863 p.

[41] Hariprasath, S. un Prabakar, T. N. FPGA implementation of multilayer feed forward neural network architecture using VHDL// *Computing, Communication and Applications (ICCCA), 2012 International Conference on.*. IEEE, 1—6 p. / URL - http://ieeexplore.ieee.org/abstract/document/6179225/ (aplūkots 06.09.2017.).

[42] Youssef, A., Mohammed, K. un Nasar, A. A Reconfigurable, Generic and Programmable Feed Forward Neural Network Implementation in FPGA// *2012 UKSim 14th International Conference on Computer Modelling and Simulation.*. IEEE, marts 9—13 p. ISBN: 978-1-4673-1366-7 978-0-7695-4682-7. DOI: 10.1109/UKSim.2012.12. / URL - http://ieeexplore.ieee.org/document/6205543/ (aplūkots 06.09.2017.).

[43] Yuan Jing, Youssefi, B., Mirhassani, M. un Muscedere, R. An efficient FPGA implementation of Optical Character Recognition for License Plate Recognition en// *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE).*. IEEE, apr. 1—4 p. ISBN: 978-1-5090-5538-8. DOI: 10.1109/CCECE.2017.7946734. / URL - http://ieeexplore.ieee.org/document/7946734/ (aplūkots 17.04.2018.).

[44] Oliveira, J. G. M., Moreno, R. L., Oliveira Dutra, O. de un Pimenta, T. C. Implementation of a reconfigurable neural network in FPGA en// *2017 International Caribbean Conference on Devices, Circuits and Systems (ICCDCS).*. Cozumel, Mexico: IEEE, jūn. 41—44 p. ISBN: 978-1-5386-1962-9. DOI: 10.1109/ICCDCS.2017.7959699. / URL - http://ieeexplore.ieee.org/document/7959699/ (aplūkots 08.09.2018.).

[45] Hajduk, Z. Reconfigurable FPGA implementation of neural networks en *Neurocomputing* sept. Vol.308. 227—234 p. Sept. ISSN: 09252312. DOI: 10.1016/j.neucom.2018.04.077. / URL - https://linkinghub.elsevier.com/retrieve/pii/S0925231218305393 (aplūkots 08.09.2018.).

[46] Bayer, B. E. Color imaging array. ASV pat. Nr.US3971065A.

[47] Li, X., Gunturk, B. un Zhang, L. Image demosaicing: a systematic survey en Pearlman, W. A., Woods, J. W. un Lu, L., izdev.. San Jose, CA, janv. 68221J p. DOI: 10.1117/12.766768. / URL - http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.766768 (aplūkots 09.08.2020.).

[48] Yu, H. un Leeser, M. Automatic Sliding Window Operation Optimization for FPGA-Based// *2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines.*. 76—88 p. DOI: 10.1109/FCCM.2006.29.

[49] Hagara, M., Stojanović, R., Bagala, T., Kubinec, P. un Ondráček, O. Grayscale image formats for edge detection and for its FPGA implementation *Microprocessors and Microsystems* Vol.75. 103056 p. ISSN: 0141-9331. DOI: https://doi.org/10.1016/j.micpro.2020.103056. / URL - https://www.sciencedirect.com/science/article/pii/S0141933119305034.

[50] Huntsberger, T. un Descalzi, M. Color edge detection *Pattern Recognition Letters* Vol.3. Nr.3. 205—209 p. ISSN: 0167-8655. DOI: https://doi.org/10.1016/0167-8655(85)90054-6. / URL - https://www.sciencedirect.com/science/article/pii/0167865585900546.

[51] Cook, J. D. Three algorithms for converting to grayscale. (aug.) / URL - https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/.

Rihards Novickis dzimis 1993. gadā Daugavpilī. Rīgas Tehniskajā universitātē (RTU) ieguvis bakalaura grādu (2014) un maģistra grādu (2016) elektronikā. Strādājis SIA "Belam-Riga" un AS "SAF Tehnika". Kopš 2016. gada strādā Elektronikas un datorzinātņu institūtā (EDI), ieņemot elektronikas inženiera amatu, patlaban – pētnieka amatu. Kopš 2019. gada strādā arī RTU, ieņemot asistenta (izglītības jomā) amatu. Saņēmis Latvijas Zinātņu akadēmijas Atzinības rakstu par pētījumu "Oriģināla pieeja mākslīgo neironu tīklu arhitektūras transformēšanai programmējamo loģisko masīvu struktūrās". Bijis nacionālais vienkristāla sistēmu eksperts, pārstāvot Latviju. Zinātniskās intereses saistītas ar heterogēnām vienkristāla sistēmām un to izmantošanu algoritmu veiktspējas un efektivitātes uzlabošanai.