

LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

**VIRTUĀLO MŪZIKAS INSTRUMENTU UN SIGNĀLU AP-
STRĀDES EFEKTU ARHITEKTŪRA IEROBEŽOTU RESURSU
APARATŪRĀM.**

MAĢISTRA DARBS

Autors: **Gusts Kaksis**

Stud. apl. Nr.: gk17025

Darba vadītājs: profesors dr.comp.sc. Leo Seļāvo

RĪGA 2019

ANOTĀCIJA

Dators mūsdienu dzīvās mūzikas izpildījumā ir kļuvis par vienu no instrumentiem uz skatuves, diemžēl no lietošanas ērtuma paaugstināta stresa situācijās datora un grafiskās operētājsistēmas vispusība var būt traucējošs faktors. Kā alternatīvas darbā tika pētītas procesoru arhitektūru un operētājsistēmu kombinācijas virtuālo mūzikas instrumentu un virtuālo skaņas efektu lietotņu darbināšanai uz ierobežotu resursu iegultām ierīcēm priekš dzīvā mūzikas izpildījuma.

Tika apskatīti divu veidu iekārtas un programmatūras darbināšana uz tām caur Linux operētājsistēmu un tiešā realizācijā uz procesora. Kā galvenie pētījuma kritēriji tika ņemti vērā izstrādes rīku lietošanas ērtums, procesoru veiktspēja peldošā punkta aprēķiniem un strāvas patēriņš ar mērķi lietot iekārtu ar bateriju.

Tika secināts, ka ar mūsdienās samērā lēti pieejamām iegultajām iekārtām ir iespējams darbināt virtuālo mūzikas instrumentu programmas, šīs iekārtas spēj sāknēties pietiekami ātri un tās spēj darboties pietiekami ilgi no baterijas, lai tās varētu izmantot dzīvajam izpildījumam uz skatuves.

ARHITEKTŪRA, VST, DAW, IEGULTĀS IERĪCES, MŪZIKAS INSTRUMENTI, SKAŅU EFEKTI, SKAŅAS SIGNĀLA APSTRĀDE, SKAŅAS SIGNĀLA SINTĒZE.

ABSTRACT

Virtual music instrument and signal processing effect architecture for limited resource hardware.

In today's live music performance a computer has become one of the music instruments on stage, but unfortunately it might become a cumbersome factor because of its universal nature. As an alternative this thesis researches processor architecture and operating system combinations for running virtual music instrument and virtual sound effects applications on specially developed embedded hardware for live music performances.

The research was done on two different devices with programs running on Linux operating system and directly on processor (*bare metal*). The main criteria for the research were - the ease of use of development tools, processor performance of floating point calculations and power consumption with an aim to use the device with battery power.

Conclusions were drawn that with today's relatively cheap embedded hardware devices it's possible to run virtual music instrument software, these devices boot up fast enough and they can perform long enough on battery power to use them for live performances on stage.

ARCHITECTURE, VST, DAW, EMBEDDED HARDWARE, MUSIC INSTRUMENTS, SOUND EFFECTS, DIGITAL SIGNAL PROCESSING, SOUND SIGNAL SYNTHESIS.

AUTORREFERĀTS

Darbā tika pētīts digitālo mūzikas instrumentu nišas tehnoloģijas un iespēju tās uzlabot ar atvērtā koda programmatūru un populārām iekārtu arhitektūrām. Galvenā darba novitāte ir savienot esošus iekārtu risinājumus ar esošiem programmatūras risinājumiem, lai atvieglotu jaunu mazu resursu iekārtu un platformu izstrādi uz kurām darbināt virtuālos mūzikas instrumentus un skaņu efektus.

Literatūras apskatā tika pētīti gan zinātniski raksti par šādu vai līdzīgu iekārtu dizainu un uzlabojumiem, gan arī esošā tirgus situācija un gatavie produkti - subjektīvi novērtējot to plusus un mīnus.

Problēmu izklāsts darbā ir ieturēts virspusīgā veidā, padziļināti tikai pētot 3 faktorus - veikspēju, enerģijas patēriņu un pastarpināti arī tālāku izstrādes ērtumu.

Darbā ir izmantots iepriekš ieguldīts aptuveni 7 gadu plānošanas un pētniecības darbs, savukārt pašā darba praktiskajā daļā ieguldītais apjoms lielākoties izmantots apgūstot dažādas izstrādes vides un iekārtu arhitektūras īpatnības. Pats darbs izstrādāts uz 3 operētājsistēmām, par pamatu lietojot macOS, bet atsevišķu arhitektūru prasību dēļ izmantotas arī Linux un Windows operētājsistēmas.

SATURA RĀDĪTĀJS

Apzīmējumu saraksts	6
Ievads	8
1. Digitālās mūzikas tehnoloģijas	10
1.1. MIDI	10
1.2. OSC	10
1.3. LPCM	10
1.4. VST	11
1.5. LV2 (LADSPA version 2)	11
1.6. Grafiskās programmēšanas vides Pure Data un Max/MSP	11
2. Procesoru arhitektūras	13
2.1. Intel64	13
2.2. ARMv8	13
2.3. SHARC	14
2.4. FPGA	14
3. Operētājsistēmas	15
3.1. Microsoft Windows	15
3.2. Linux	15
3.3. BareMetal	15
3.4. Wine	16
4. Arhitektūras un problēmu risinājumi citā literatūrā	17
5. Esoši produkti	19
6. Platformas prasības	23
6.1. Iekārtas arhitektūra	24
6.2. Programmatūras arhitektūra	25
7. Procesoru arhitektūru un operētājsistēmu veikspējas pētījums.	27
7.1. Diskrētā Furjē transformācija	28
7.2. Radix-2 ātrā Furjē transformācija	29
7.3. Eksperimentālais uzstādījums	32
7.4. SHARC Audio Module	33
7.5. Raspberry Pi un Xilinx Spartan 3E Starter Kit	35
8. Mērījumu rezultāti	38
Secinājumi	42
Literatūras saraksts	44

APZĪMĒJUMU SARAKSTS

Saīsinājums	Skaidrojums
ADC	angliski - Analog-Digital Converter, jeb analogi-digitālais pārveidotājs
DAC	angliski - Digital-Analog Converter, jeb digitāli-analogais pārveidotājs
DAW	angliski - Digital Audio Workstation, jeb digitālā audio darbstacija
DFT	Diskrētā Furjē Transformācija
DSP	angliski - Digital Signal Processor, jeb digitālā signāla procesors
ELF	angliski - Executable and Linkable Format - Linux izpildāmo programmu failu formāts
FFT	angliski - Fast Fourier Transform, jeb ātrā Furjē transformācija
FPGA	angliski - Field Programmable Gate Array, jeb lauka programmējamo loģikas vārtu masīvs
FPU	angliski - Floating point processing unit, jeb peldošā punkta procesēšanas vienību
GPU	angliski - Graphical processing unit, jeb video apstrādes palīgprocesors
HTTP	angliski - Hyper Text Transport Protocol
JTAG	angliski - Joint Test Action Group
LPCM	angliski - Linear Pulse-code modulation
LV2	angliski - Linux Audio Developer's Simple Plugin API version 2
MCAPI	angliski - Multi-Core API - daudzkodolu saziņas protokols
MIDI	angliski - Musical Instrument Digital Interface, jeb mūzikas instrumentu digitālā saskarne
OSC	angliski - Open Sound Control, jeb atvērtā skaņas kontrole
PCIe	angliski - Peripheral Component Interconnect Express - iekārtu paplašinājumu saskarnes standarts
POSIX	angliski - Portable Operating System Interface - standarts, kas definē vienotu sistēmas izsaukumu standartu
RTOS	angliski - Real time operating system, jeb reālā laika operētājsistēmas
SHARC	angliski - Super Harvard Architecture Single-Chip Computer
SIMD	angliski - Single Instruction Multiple Data, jeb vektoru aritmētikas

Saīsinājums	Skaidrojums
UART	angliski - Universal Asynchronous Receiver-Transmitter, jeb universāls asinhronais uztvērējs un raidītājs
USB	angliski - Universal Serial Bus, jeb universālais seriālais pieslēgums
VST	angliski - Virtual Studio Technology

IEVADS

Šodien personālais dators jau ir ieņēmis stabilu lomu dzīvās mūzikas izpildījumā, kā instruments vai skaņas efektu procesors. Tos dēvē pār DAW, kas parasti ir standarta galda vai portatīvais dators ar labas kvalitātes skaņas ierīci un speciālu skaņas apstrādes programmatūru. Virtuālie instrumenti un virtuālie skaņas efekti ir programmas, kas sintezē vai apstrādā skaņas signālu, tos lieto kā spraudņus, slēdz dažādās kombinācijās. Šie risinājumi ir ļoti populāri salīdzinot ar specializētām iekārtām, kurām ir viens pielietojums, jo tas ļauj daudz ātrāk, lētāk un ērtāk nomainīt skanējumu pārlādējot šos spraudņus.

Sākotnēji DAW pielietojums bija tikai ierakstu studiju vajadzībām, jo šādi datori, kas spēj veikt skaņas apstrādi pietiekami ātri bija izmēros lieli, līdz ar to stacionāri. Šobrīd, pieaugot datoru veiktspējai, DAW pielietojums izmantojot portatīvos datorus ir atradis arī vietu dzīvajā mūzikas izpildījumā.

Datora lietošanai uz skatuves ir divi lieli mīnusi:

1. Tā kā dators ir radīts vispārējai lietošanai, tas pieprasa paaugstinātu vizuālu uzmanību, kas mūziķa gadījumā nozīmē uzmanības novēršanu no publikas vai mūzikas instrumenta (DAW gadījumā tas var būt arī speciāls kontrolieris), jo lietotāju kontrole prasa vizuālu uzmanību, t.i. mūziķim ir jāskatās datora ekrānā, lai koordinētu kursoru vai kādu citu ievades rīku, kas no skatuves estētikas viedokļa bieži izskatās, ka mākslinieks lasa e-pastu vai dara kaut ko citu ar mūziku nesaistītu un nerada saiknes sajūtu starp mākslinieku un viņa radīto skaņu.
2. Papildus tam var minēt lietotāja saskarnes, kurām trūkst fiziskās atdeves mehānismu, līdz ar to tas pieprasa papildus uzmanību no mūziķa, lai koordinētu savu saskarni ar grafiskās vides elementiem. Tāpat arī grafiskā vide ir mainīga un katru reizi pieprasa lietotāja pārorientēšanos, kas prasa laiku. Gluži pretēji mūzikas instrumentiem ar taustiņiem, stīgām vai mūzikas efektiem ar pāris fiziskām pogām mūziķis var brīvi paļauties uz tausti un atmiņu.

Šim nolūkam šajā darbā apskatītas dažādas arhitektūras un tehniskie risinājumi, kuri paredzēti virtuālo instrumentu un virtuālo skaņas efektu darbināšanai dzīvās mūzikas izpildījumā - aparatūras un programmatūras kopne ar minimālu fizisko saskarni, kompakto izmēru, kā arī pieņemamu cenu.

Tirgū pēdējos 10 gados šādi risinājumi ir parādījušies un arī pazuduši, iespējams ekonomisku iemeslu dēļ, bet neviens no tiem nav bijis lēts un kā lielāko problēmu tajā var saskatīt vispusīgu un dārgu tehnoloģiju izmantošanu.

Galvenie šī darba mērķi ir:

1. uzlabot digitālo iekārtu pielietojumu dzīvajā mūzikā - noņemot uzmanību no datortehnikas uz skatuves;
2. saglabāt lietošanas ērtumu, kas dzīvās uzstāšanās laikā ir īpaši svarīga, jo izpildījuma laikā mākslinieks var veltīt maz uzmanības ($t < 1s$) lielākoties paļaujoties uz iepriekšēju un paredzamu pieredzi;
3. saglabāt lietošanas stabilitāti, jo uzstāšanās laikā instrumentam ir jādarbojas paredzami un bez pārtraukumiem, bet kritiskos gadījumos nodrošina ātru pārlādes spēju ($t < 10s$);
4. maksimāli izmantot jau esošas tehnoloģijas, lai mazinātu iespējamās ražošanas izmaksas;
5. nodrošināt pilnu izstrādes rīku komplektu un labu dokumentāciju, lai nodrošinātu vieglu lietoņu izstrādi;
6. izpētīt iespējas darbināt iekārtas izmantojot akumulatoru enerģiju (ar ietilpību $\sim 1800mAh$ un spriegumu $7,4 V$).

Vispirms darbā tiek apskatītas mūzikas un skaņas apstrādes tehnoloģijas, kuras tiek lietotas uz DAW, pēc tam padziļinātāk pētītas procesoru arhitektūras, operētājsistēmas un visu beidzot esošie risinājumi tirgū - to plusi un mīnusi.

1. DIGITĀLĀS MŪZIKAS TEHNOLOĢIJAS

1.1. MIDI

MIDI ir viens no galvenajiem mūzikas instrumentu standartiem, ko 1982. gadā izveidoja kompānija Moog Music, ļauj digitāli pārsūtīt vēlamās nots un citu modulējošu atribūtu informāciju.

Tas ir balstīts uz seriālo UART signāla apmaiņas standartu, izmantojot 31.25 Kbit/s datu pārsūtīšanas ātrumu, datus kodējot ar 1 sākuma bitu, 8 datu bitiem un 1 apturēšanas bitu. No 8 bitiem lielākā daļu ziņojumu izmanto tikai zemākos 7 bitus, līdz ar to modulējošo parametru diskretizācija ir tikai 127 soļi.

Par pamata fizisko savienojumu tiek izmantoti atsevišķi DIN-5 standarta savienojumi - attiecīgi viens datu saņemšanai, otrs nosūtīšanai, bez obligātas prasības izmantot abus. [1]

Tāpat MIDI ir iespējams pārsūtīt caur USB savienojumu, kur to klasificē speciāls iekārtas tips. [2]

1.2. OSC

OSC ir daudz modernāks un pilnībā atvērts standarts, kas radies 1997. gadā, bet atšķirībā no MIDI, šis ir tīkla bāzēts protokols, kura dati tiek pārsūtīti UDP vai TCP paketēs. Līdz ar to šis standarts ir pakļauts tīkla traucējumiem.

Standarts paredz savienojumu starp OSC klientu (OSC Client - datu paketes sūtītājs) un OSC serveris (OSC Server - datu paketes saņēmējs) un bināras datu paketes ar kurām abas puses apmainās. Sīkāk OSC paketes (OSC Packet) dalās OSC ziņojumos (OSC Message) un OSC sainī (OSC Bundle).

Atšķirībā no MIDI, kur vienu vērtību var aprakstīt tikai 7 bitos, OSC paketes ir mainīga garuma un ļauj izmantot pat 64 bitu skaitļus dažādu modulējošu vērtību aprakstīšanai, kas sniedz daudz augstāku diskretizācijas pakāpi un attiecīgi gludākas modulācijas līknes. Tāpat standarts paredz paplašinājumus. [3]

1.3. LPCM

LPCM ir audio signāla kodēšanas standarts, kurš radās līdz ar kompaktdisku un ir iekļauts tā sauktajā "Compact Disc Digital Audio" (pazīstam arī kā "Red Book") standartā, kas izstrādāts 1982. gadā.

Kodēšanas standarts paredz diskrētu audio signāla datu aprakstīšanu, kur signāls tiek mērīts noteiktos laika intervālos (piemēram 44100 reizes sekundē) un aprakstīts izmantojot 4, 8, 16 vai 24 bitu skaitļus.

Šādu diskretizāciju elektronikas līmenī veic ADC savukārt demodulāciju veic DAC. [4, 5]

Šis standarts ir kļuvis par de-facto standartu skaņas signāla apstrādē un parasti tiek izmantots skaņas signāla iegūšanai no audio ieejām un skaņas signālu nodošanai uz audio izejām, bet parasti iekšējā apstrāde tiek veikta izmantojot nevis veselus skaitļus, bet gan peldošā punkta decimāldaļskaitļus.

1.4. VST

VST ir kompānijas Steinberg izstrādāts virtuālo instrumentu un virtuālo skaņas efektu standarts. Tas ir pasaulē populārākais standarts savā jomā un to atbalsta gandrīz visas DAW programmatūras, kas balstītas uz visām trīs populārākajām operētājsistēmām (t.i. Windows, macOS un Linux).

Standarts definē C++ valodas saskarni no spraudņa ("Plug-In") puses un saimnieka vides ("Host Environment") puses. Standarts paredz divus spraudņu tipus:

1. Skaņas apstrādes spraudņi, kuri saņem ienākošo audio signālu kā maza izmēra buferi, veic šajā buferī esošo diskrēto vērtību apstrādi un atgriež izejošo audio signālu kā citu buferi.
2. Skaņas ģenerācijas spraudņi, kuri saņem informāciju par notīm un modulācijas parametriem, veic skaņas sintēzi vai kopēšanu no gatavie skaņas datiem un atgriež izejošo skaņas signālu kā maza izmēra buferi.[6]

1.5. LV2 (LADSPA version 2)

LV2 ir atvērts virtuālo instrumentu un virtuālo skaņas efektu standarts. Būtībā identisks standarts VST, bet nav ierobežojumu komerciālam pielietojumam. [7]

1.6. Grafiskās programmēšanas vides Pure Data un Max/MSP

Lielu popularitāti mūziķu un skaņu inženieru vidū ir ieguvušas grafiskās programmēšanas vides, kas atslogo izstrādātāju no sarežģītu arhitektūras problēmu risināšanas un ļauj koncentrēties uz vizuāli saprotamu signālu ceļu plānošanas. Pamata elementi sastāv no

signāla ievada, izvada, aritmētikas elementiem, kā arī nedaudz augstāka līmeņa elementiem, kā piemēram: filtriem, buferiem un trigonometriskām funkcijām, kuras kombinējot iespējams veidot dažādus skaņu efektus. [8, 9]

2.

PROCESORU ARHITEKTŪRAS

Skaņu signāla apstrādē bieži ir jāveic aritmētiskas darbības ar peldoša punkta decimāldaļskaitļu vektoriem, līdz ar to jāizskata arī dažādu procesoru arhitektūru piedāvātā funkcionalitāte un veiktspēja. Tāpat jāizvērtē procesoru strāvas patēriņa rādītāji.

2.1. Intel64

Populārākā procesoru arhitektūra, kuru izmanto datoros uz kuriem tiek darbinātas DAW. Jaunākās (8.) paaudzes procesori piedāvā AVX-512 instrukciju kopu, kas nodrošina 32 512 bitus platu vektoru reģistrus, kuros iespējams katrā ierakstīt 16 32 bitu peldošā punkta skaitļus. Tas principā nodrošina iespēju ielādēt reģistros veselu 512 punktu buferi un veikt, piemēram, vektoru saskaitīšanas un reizināšanas neiesaistot operatīvo atmiņu tādējādi krietni paātrinot programmas darbību. Tāpat ir arī pieejami vēsturiskie SSE, SSE2, SSE3, SSE4, AVX un AVX2 instrukciju seti, kas tiek emulēti uz jaunā AVX-512 bāzes.

Strāvas patēriņa ziņā zemākais patēriņš 8. paaudzes Intel procesoriem ir 5W (Core i5-8200Y, 2 kodoli, 1,30 GHz takts frekvence, 4MB pagaidu atmiņa), kas jaudas ziņā pilnībā pietiktu iegultas iekārtas vajadzībām.

Uzreiz jāpiezīmē - tādēļ, ka šī ir populārākā galda un portatīvo datoru procesoru arhitektūra tai ir īpaša priekšrocība izstrādes un testēšanas rīku izstrādei, jo nav jādomā par emulācijas risinājumiem un kļūdu gadījumā tos ir vieglāk atklāt darbinot spraudni uz tā paša datora, kur to izstrādā. [10]

2.2. ARMv8

Netiek izmantota datoros, bet gan iegultajās iekārtās, mobilajos tālruņos un planšetdatoros, kuri mūsdienās arī ir iekarojuši savu nišu dzīvās mūzikas izpildījumā. Atšķirībā no Intel64 arhitektūras, šo procesoru specifikācijas ir vairāk informatīva un reāli pieejamie procesori ir atkarīgi no to ražotājiem - kādā konfigurācijā tie būs.

Tāpat kā Intel64 arī ARMv8 arhitektūrai, ir pieejami vektoru aritmētikas instrukcijas, ko piedāvā SVE instrukciju kopas paplašinājums, tiesa šīs instrukciju kopa atbalsta tikai 32 256 bitu reģistrus, līdz ar to vienlaicīgi apstrādāt būtu iespējams tieši divas reizes mazāk vektora elementus. Bez SVE instrukciju kopas paplašinājuma ARMv8 piedāvā arī SIMD instrukciju kopu, pazīstamu kā NEON, ar 32 128 bitu reģistriem. [11, 12]

Interesanti, ka ARM piedāvā ne tikai tā saukto lietotņu (application) procesorus, bet arī reālā laika arhitektūru zem ARMv8-R standarta, kas iespējams nebūtu piemērots pilnvērtīgas operētājsistēmas darbināšanai, toties būtu labs risinājums reālā laika signāla apstrādei. [13]

Salīdzinājumam ar Intel Core i5 procesoru ir pieejami risinājumi no Qualcomm, piemēram, procesors APQ8053 ar 1.8 GHz takts frekvenci un strāvas patēriņu 6,5 W. Papildus tam, šis procesors iekļauj savu DSP palīgprocesoru, kas būtu tieši noderīgs skaņas signālu apstrādei, noņemot slodzi no galvenā procesora. [14]

2.3. SHARC

SHARC ir specializēta DSP procesora arhitektūra, kas balstīta uz Harvardas arhitektūru (pretēji Intel un ARM izmantotajai von Neuman arhitektūrai), kuras galvenais pluss ir nodalītās instrukciju un datu atmiņas piekļuves līnijas. Tas padara šo arhitektūru perfektu straumējošu datu apstrādei, bet tā nav paredzēta vispārēju lietoņu darbināšanai. Šajā darbā tā tiek apskatīta kā palīgprocesora variants, lai atslogotu galvenā procesora darbu. [15]

2.4. FPGA

FPGA, nav gluži CPU arhitektūra, bet gan tāda veida integrētā shēma, kas sastāv no loģisko vārtu masīva, kur savienojumus starp vārtiem iespējams pārveidot pēc vajadzības. Šī arhitektūra dod iespēju ļoti zemā līmenī (digitālā signāla līmenī) izstrādāt pielāgotu procesora arhitektūru, piemēram, ļoti veikspējīgu diskreto Furjē transformāciju procesoru, kas spētu transformēt lielus datu buferus mazā laika vienībā.

Kā galvenais mīnuss šāda tipa integrētajai shēmai ir vienotu standartu trūkums, kas produkta plānošanu un izmaksas piesaistītu konkrēta FPGA ražotāja licenzēšanas nosacījumiem un izcenojumiem, kā arī būt jāizmanto papildus izstrādes rīki, kas ļautu sagatavot binārās datnes no Verilog vai VHDL valodām. No otras puses lietotņu izstrādātājiem var piedāvāt gatavas binārās datnes, kuras pārveido iebūvēto FPGA integrēto shēmu uz konkrētas funkcionalitātes palīgprocesoru. [16]

3.

OPERĒTĀJSISTĒMAS

3.1. Microsoft Windows

Populārākā darbstaciju operētājsistēma pasaulē, kas vairāk orientējas uz grafisko saskarni, uzņēmumu vidi un attiecīgu atpakaļejošu saderību ar agrākām operētājsistēmas versijām, kas svarīgas lielu uzņēmumu infrastruktūrām. Tieši šie abi faktori var spēlēt šīs operētājsistēmas izvēlē, jo grafiskā vide un atpakaļejošas savietojamības slāņi bieži vien uzliek papildus slodzi datora resursiem. Līdz ar to uzreiz jāpiezīmē, ka tā ir pārāk liela priekš šāda tipa arhitektūras, arī Microsoft Windows Embedded versija, kas tieši paredzēta plānajiem klientiem, automašīnu informatīvajiem centriem u.c. iegultajiem risinājumiem. Papildus jāņem vērā, ka šī ir komerciāla operētājsistēma ar dažādiem licenzēšanas modeļiem, kas palielinātu produkta izcenojumu.

3.2. Linux

Populārākā operētājsistēma serveru un iegulto risinājumu radīšanai. Sistēmas kodols ir pielāgots darbam ar gandrīz jebkādu procesora arhitektūru, tai ir ļoti plašs iekārtu draiveru atbalsts un pats galvenais tā ir bezmaksas. Sistēmas kodola licence, gan nosaka, ka jebkādas izmaiņas kodola pirmkodā ir jāpublicē zem GPL licences, kas dažkārt komerciāla risinājumu gadījumā nav pieņemami, bet tā kā kodols ir licenzēts izmantojot GPL 2 versijas licenci, to iespējams apiet, ja izmaiņas nav jāveic tieši kodola pirmkodā, tad visa veida pielāgojumus ir iespējams veikt ar kodola moduļu palīdzību, ielādējot tos dinamiski un nekompilējot vienā binārā failā.

3.3. BareMetal

Minimālistisks sistēmas kodola risinājums, kas rakstīts Intel64 arhitektūrai izmantojot Intel64 asamblejas valodu, un piedāvā ielādēt vienu programmu un darbināt to uz visiem pieejamajiem kodoliem nodrošinot gan virtuālās atmiņas aizsardzību, gan pavedienu laika plānošanu. Licenzēts ar MIT licenci, kas nozīmē, ka sistēmas kodu drīkst modificēt un izmaiņas nav jāpublicē. Izstrādātāji plāno izstrādāt arī ARMv8 versiju, kas pavērtu plašākas pielietojuma iespējas tieši šīs arhitektūras vajadzībām. Ka lielāko mīnusu jāmin draiveru trūkums, kas šobrīd aprobežojas ar virtuālo mašīnu tīkla un datu glabāšanas iekārtu

draiveriem, bet sistēmas uzbūve ļauj samērā viegli ieviest savus draiverus atbilstošās arhitektūras un konfigurācijas vajadzībām. [17]

3.4. Wine

Microsoft Windows operētājsistēmas emulācijas slānis, kas darbojas tādās operētājsistēmās kā Linux, BSD un macOS. Pēdējās versijas nodrošina gandrīz perfektu bināru saderību ar Microsoft Windows lietotnēm, t.i. spēj darbināt Microsoft Windows lietotni citas operētājsistēmas vidē. Protams, risinājums nav pilnībā stabils un pie tā turpinās darbs, jo programmatūra tiek izstrādāta izmantojot apgriezto inženieriju. [18]

4. ARHITEKTŪRAS UN PROBLĒMU RISINĀJUMI CITĀ LITERATŪRĀ

Deviņdesmito gadu literatūrā par audio signālu apstrādi bieži tiek pieminētas RTOS, kā risinājums nepārtrauktai digitālā skaņas signāla plūsmai, tas droši vien ir skaidrojams ar tā laika DAW veikspējas ierobežojumiem, piemēram, 1992. gadā Intel x86 arhitektūras procesori darbojās ar vienu kodolu, nebija aprīkoti ar tik sarežģītu instrukciju konveijeru un darbojās ar no 40 līdz pat 140 reizes zemāku takts frekvenci (33MHz vs. 1.3-4.6GHz). Līdz ar to saprotams, ka tajos laikos nodrošināt paredzamu funkciju izpildes laiku bija ļoti svarīgi, turpretī mūsdienās šādam nolūkam noder parastas pārtraucošas operētājsistēmas.[19]

Savukārt Francijas akustiskas un mūzikas izpētes institūts 1991. gadā pētnieciskos nolūkos izstrādājuši savu RTOS operētājsistēmu, kas darbināta uz vairākiem standarta Intel i860 arhitektūras procesoriem paralēli. Atšķirībā no iepriekšminētā x86, Intel i860 ir RISC arhitektūras procesors ar iebūvētu FPU. Tā kā šis risinājums darbojās ar vairākiem paralēliem procesoriem un tam ir iebūvēts FPU tā veikspēja ir krietni lielāka par iepriekš minēto i386 procesoru, kuram FPU ir pieejams tikai kā ārējs palīgprocesors. Šī sistēma ir pirmā, kurā skaņas signālu apstrādei tika nolemts lietot peldošā punkta vektoru aprēķinus, aizstājot tajā laikā populāro veselo skaitļu vektoru aprēķinus, kuriem pastāv piesātināšanas un pārplūšanas problēmas. [20]

Salīdzinājumam ar iepriekš minētajām RTOS, citā darbā tiek apskatīts, kā risināt tieši pārtraucošas operētājsistēmas neparedzamo audio signāla nodošanas laiku uz skaņas izvadu. Darbā tiek apskatīti bufermehānismi un pētīts optimālākais bufera izmērs, lai samazinātu skaņas aizturi, kas dzīvā atskaņojuma laikā ir būtisks faktors. Buferēšana nodrošina iespēju sagatavot nelielu skaņas fragmentu un nodot to skaņas izvadei, kamēr operētājsistēma nodod kontroli citiem procesiem sistēmā. Ja buferis ir pārāk mazs, var rasties skaņas pārrāvumi, ja tas ir pārāk liels, rodas jūtama aizture. [21]

Šie faktori ir tik pat būtiski arī dzīvās mūzikas atskaņošanai no DAW.

2012. gadā Starptautiskās digitālo audio efektu konferences dokumentā tiek apskatīts FPGA pielietojums dzīvo mūzikas instrumentu modelēšanai. Darbā FPGA tiek salīdzināts ar GPU, kas bieži tiek izmantots kā populārs lielu apjomu peldošā punkta vektoru aritmētikas procesors, bet tā kā tas ir standartizēts noteiktu darbību veikšanai, tā veikspēja nav tik plaša,

kā FPGA, kura paralelizāciju un iebūvētos algoritmus iespējams pārkonfigurēt atbilstoši specifiskajai situācijai. [22] Savukārt 2007. gada IEEE rakstā apskatītas detalizētas skaņas sintēzes ieviešanu uz FPGA, kas norāda uz plašu pielietojumu tieši šādā projektā. [23]

5.

ESOŠI PRODUKTI

Veicot tirgus izpēti uz 2018. gada decembri ir atrodami sekojoši gatavi risinājumi vai uzņēmumi, kas veido daļējus šajā darbā apskatītus risinājumus:

1. SM Pro Audio V-Machine [24]

5.1. Tabula

Procesora arhitektūra	Intel64
Operētājsistēma	pielāgots Linux
Emulācijas slāņi	Wine
Lietotņu arhitektūra	VST
Grafiskā vide	Grafisko vidi pārsūta caur tīklu uz saimniekdatoru, kur to attēlo caur speciālu kontroles programmu
Vairāku lietotņu atbalsts	Ir, iespēja slēgt ķēdē
Izmērs	18 cm x 11,6 cm x 3,6 cm
Cena	~175 EUR
Stāvoklis	Pārtraukta ražošana

Šī iekārta ir paredzēta tieši dzīvā izpildījuma vajadzībām - tā ir kompakta un neparedz pārāk lielu uzmanības piesaisti. Kā galveno plusu uzskatāma iespēja darbināt jau gatavus VST standarta spraudņus, bet kā lielākais mīnuss uzskatāms Wine emulācijas slānis, kas nenodrošina pietiekamu stabilitāti dzīvās uzstāšanās laikā.

2. Seelake AudioStation X64 [25]

5.2. Tabula

Procesora arhitektūra	Intel64
Operētājsistēma	Microsoft Windows 7
Emulācijas slāņi	Nav
Lietotņu arhitektūra	VST
Grafiskā vide	Grafisko vidi attēlo caur pievienotu monitoru
Vairāku lietotņu atbalsts	Ir, sarežģīta signālu maršrutēšana, līdz 76 lietotnēm
Izmērs	8,9 cm x 44,6 cm x 42,5 cm (U2 statīva formāts)
Cena	1300 EUR
Stāvoklis	Ir nopērkams

Šī iekārta ir vairāk paredzēta ierakstu studijas vajadzībām - par ko liecina tās formāts - U2 formāta korpuss. Kā arī nepieciešamība pēc ārēja monitora - līdz ar to tas nav praktisks risinājums dzīvajam izpildījumam. Tāpat arī cena, kuru iespējams palielina Microsoft Windows licenzēšanas izmaksas. Galvenais pluss ir VST spraudņu darbināšana īstā Microsoft Windows vidē

3. Muse Research Receptor [26]

5.3. Tabula

Procesora arhitektūra	Intel64
Operētājsistēma	pielāgots Linux
Emulācijas slāņi	Wine
Lietotņu arhitektūra	VST, bez grafiskās vides
Grafiskā vide	Grafisko vidi attēlo caur pievienotu monitoru
Vairāku lietotņu atbalsts	Ir
Izmērs	8,9 cm x 44,6 cm x 42,5 cm (U2 statīva formāts)
Cena	1600 EUR
Stāvoklis	Pārtraukta ražošana

Gluži, kā Seelake AudioStation X64 spēj šī iekārta ir vairāk paredzēta ierakstu studijas vajadzībām. Tā spēj darbināt VST spraudņus, bet tāpat kā iepriekšminētais, ir ļoti dārgs un papildus tas izmanto Wine emulācijas slāni, kas rada riskus tā stabilitātei dzīvā izpildījuma laikā.

4. MOD Duo [27]

5.4. Tabula

Procesora arhitektūra	ARMv7
Operētājsistēma	pielāgots Linux
Emulācijas slāņi	Nav
Lietotņu arhitektūra	LV2, Pure Data, Max/MSP
Grafiskā vide	Grafisko vidi veido izmantojot WWW tehnoloģijas (HTML/CSS) un attēlo interneta pārlūkā, kā arī miniatūri melnbalti LCD ekrāni iebūvēti korpusā.
Vairāku lietotņu atbalsts	Nav
Izmērs	18,0 cm x 7,47 cm x 16,0 cm
Cena	700 EUR

Stāvoklis	Ir nopērkams
------------------	---------------------

Iekārta dzīvā izpildījuma vajadzībām - šī brīža labākais risinājums, jo izmanto zemas enerģijas patēriņa procesoru, minimālu Linux kodolu un atbalsta abus populārākos grafiskās programmēšanas projektu formātus, kas paplašina iespēju nodrošināt iekārtas lietotājus ar gataviem spraudņiem un vieglāku iespēju pašiem lietotājiem programmēt savus. Kā galvenie mīnusi būtu minami WWW tehnoloģiju izmantošana grafiskajai videi, kas liecina par potenciālu signālu aizturi starp grafisko vidi un pašu spraudni (mūziķi parasti jūt aizturi $t > 5\text{ms}$), un ARMv7 procesora izvēle:

1. Tas ir 32-bit procesors.
2. Izstrādes testēšana ir iespējama tikai uz pašas iekārtas.

5. UAD - Uzņēmuma Universal Audio risinājumus iekārtām, kas darbina signāla apstrādes lietotnes un darbojas ar iebūvētiem signāla apstrādes paātrinātājiem (DSP). Pašas lietotnes darbojas daļēji uz saimniekdatora un daļēji uz pašas iekārtas. UAD darbojas tikai ar savu formāta programmatūru un izmanto SHARC arhitektūras procesorus. [28]

6. Elk - uzņēmums, kas izstrādā speciāli pielāgotu Linux operētājsistēmu, kas spēj darbināt VST spraudņus izmantojot Wine emulācijas slāni. Interessants ir fakts, ka izstrādātāji apgalvo, ka viņi ir pārveidojuši Linux kodolu, lai tas darbotos kā RTOS, kā arī tas ka tā darbojas uz Intel un ARM arhitektūrām. [29]

Amerikas patentos ir atrodami vairāki risinājumi, kas ir tuvu vēlamajam rezultātam, bet nav pilnībā piemēram:

- Planšetes risinājums ar virtuālajiem instrumentiem, kas ļauj izvēlēties instrumentu ko spēlēt un izmantojot planšetes ievadi - skārienjūtīgo ekrānu, akselerometrus, u.c. ievades veidus. Diemžēl šis risinājums nenodrošina skaņas ievadi un ārēju kontrolieru izmantošanu, bet patentā aprakstītajai skārienjūtīgajai ievadei trūkst fiziskas atbildes reakcijas, ko sniedz īstas ģitāras stīgas vai klavieru taustiņi. Tāpat lielā skārienjūtīgā ekrāna saskarne līdzīgi kā jebkurš cits dators novērš mūziķa uzmanību, jo tam ir jākoncentrējas koordinēt savas darbības, lai precīzi trāpītu grafiskās saskarnes elementiem. [30]
- Ģitāras procesora risinājums, kas izmanto Intel x86 arhitektūras procesoru un ļauj lietotājam augšupielādēt savas efektu lietotnes no datora izmantojot USB saskarni. Šis risinājums paredz dažus ar kāju aktivizējamus slēdžus, kas ir standarta risinājums ģitāristu skaņas efektiem - līdz ar to praktiski tiek izmantotas jau lietotājam zināma saskarne. [31]

- Mūzikas instruments ar iebūvētu DAW funkcionalitāti, bāzēts uz Intel x86 standarta procesoru arhitektūras (patentā specificēts kā AMD 586-133). Šim risinājumam atšķirībā no iepriekš minētajiem ir paredzēts fizisks ievads, bet pietrūkst risinājuma ienākošas skaņas apstrādei (izņemot mikrofona ieraksta iespēju). Tāpat šis risinājums ir paredzēts kā lielformāta klavieres, kas lielā mērā ir mūzikas instruments, tomēr nav pietiekami kompakts risinājums. [32]

6.

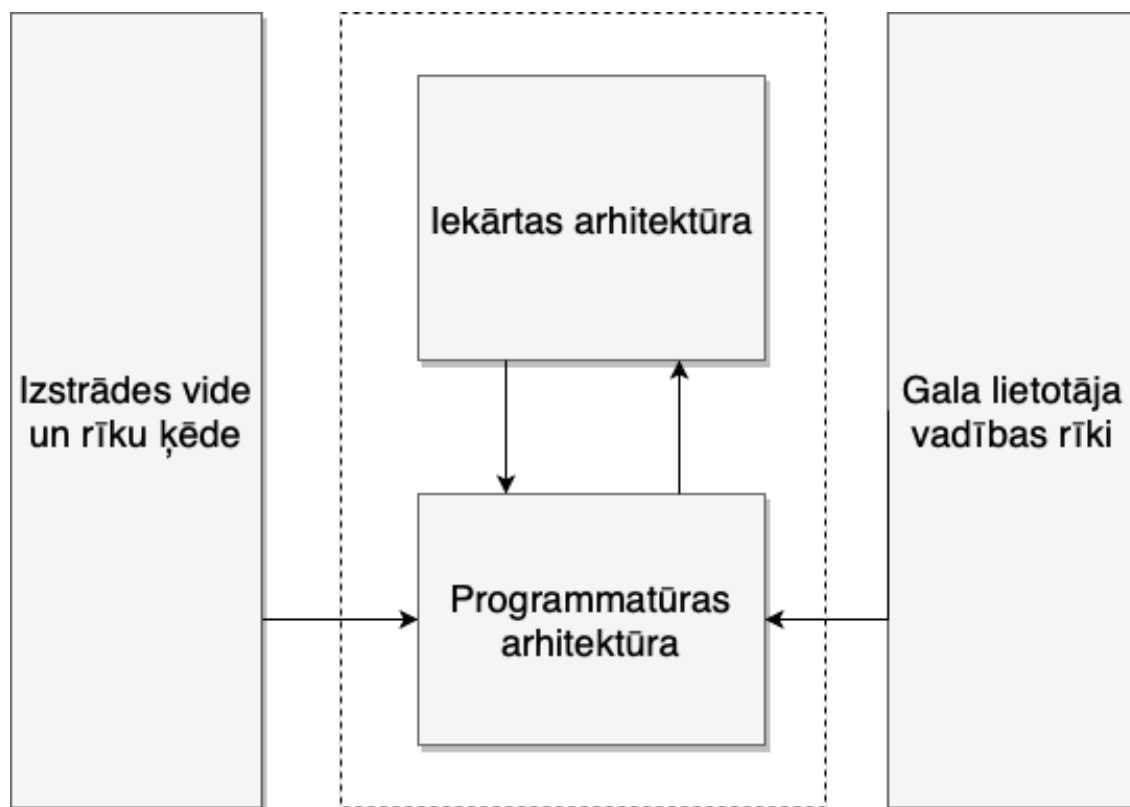
PLATFORMAS PRASĪBAS

Novērtējot visus esošos risinājumus ir skaidrs, ka šādu sistēmu izstrādei ir jāveido vesela ekosistēma, kas sastāvētu no 3 apgabaliem:

1. Šajā darbā apskatītās sfēras - iekārtas aparatūras un programmatūras arhitektūras - šajā darbā apskatītās aparatūras un operētājsistēmu kombinācijas, kas nodrošinātu visefektīvāko resursu izmantošanu attiecībā pret strāvas patēriņu un izmaksām;
2. Izstrādes vides un programmnodrošinājuma - kompilatora, linkera un pats galvenais atklūdošanas rīkiem, kurus būtu ērti izmantot instrumentu un efektu izstrādei;
3. Gala lietotāja rīkiem - bieži vien maz novērtēts faktors - gala lietotāja saskarne un instrumentu un efektu pārvaldības programmatūra.

Attiecībā pret izstrādes rīkiem izvēle var krist uz diviem atvērtā pirmkoda risinājumiem:

1. GCC - viena no populārākajām kompilatoru pakotnēm, kas arī ir ļoti populārs iegulto iekārtu izstrādes sfērā. [33]
2. LLVM - modernu kompilatoru infrastruktūra, kas balstīta uz starpniek-valodu un attiecīgi kompilatora priekšplānu, kas pārvērš kompilējamo valodu starpniek-valodā, un asamblera aizmuguri, kas starpniek-valodu pārvērš vēlamajā procesora arhitektūras mašīnkodā. [34]



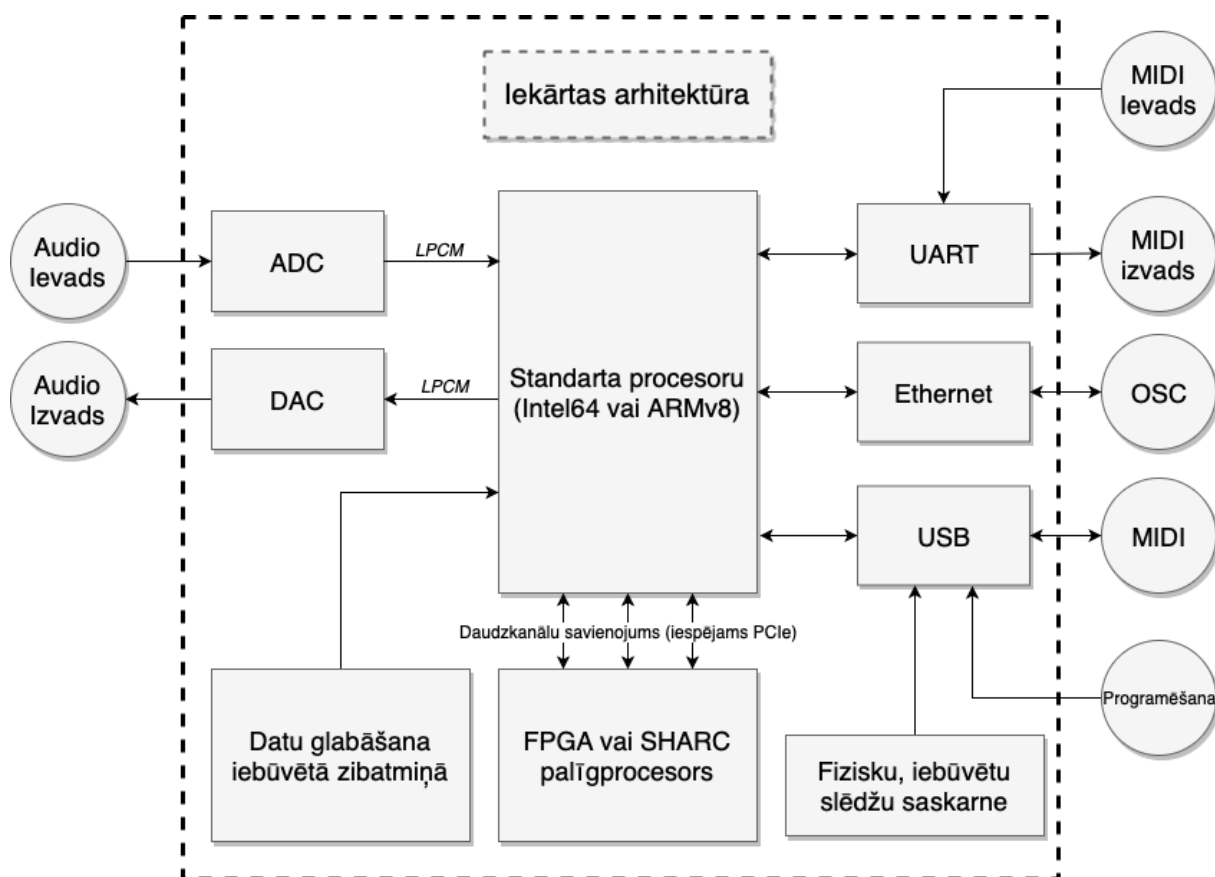
6.1. att. Ekosistēmas komponentu shēma.

6.1. Iekārtas arhitektūra

Izvēloties iekārtas arhitektūras un operētājsistēmas komplektāciju šajā darbā tika ņemti vērā sekojoši faktori:

- Brīvi pieejama aparatūras dokumentācija;
- Operētājsistēmas atbalsts iekārtas procesora arhitektūrai un iekārtu draiveri. Obligāta prasība - skaņas saskarnes draiveri;
- Kompilatora atbalsts izvēlētajai operētājsistēmai vai tīri iekārtas procesora arhitektūrai un vēlams kroskompilatora iespēja, lai programmatūras izstrādi varētu veikt no cita datora un vides;
- Procesora veiktspēju, kas būtu pietiekama digitālā skaņas signāla apstrādei reālajā laikā;
- Pietiekams daudzums operatīvās atmiņas, lai nodrošinātu gan operētājsistēmas, gan virtuālo instrumentu un skaņas efektu programmatūras, gan skaņas failu uzglabāšanu;
- Iekārtai jānodrošina sekojoša minimālā perifērija:
 - Skaņas saskarne ar labas kvalitātes ADC un DAC (min. 16-bit amplitūdas izšķirtspēju un 44.1kHz diskretizācijas frekvenci);
 - Atmiņas karšu lasīšanas iekārta - programmatūras un operētājsistēmas ielādei;
 - USB saskarne, mūzikas kontroles instrumentu, kā arī ārējo datu nesēju pievienošanai;
 - Tīkla saskarne, lai nodrošinātu skaņas un kontroles signālu pārsūtīšanu starp iekārtām;
 - Seriālā datu saskarne, lai nodrošinātu MIDI standarta savienojumu.

Papildus šiem faktoriem liela nozīme ir arī papildus palīgprocesoru pieejamība. Arhitektūras komponentu diagrammā (attēls Nr. 6.2.) redzams plānotās ievades un izvades saskarnes, iekšējās komponentes un to savstarpējie savienojumi.



6.2. att. *Provīzorisks iekārtas arhitektūra.*

6.2. Programmatūras arhitektūra

Galvenās iekārtas programmatūras arhitektūras prasības būtu:

1. Nodrošināt izpildīt vienu programmu vienlaicīgi - t.i. nav obligāta prasība nodrošināt vairāku procesu atbalstu;
2. Nodrošināt minimālu un standartizētu ielādes mehānismu (bootloader), vēlams ar iespēju ielādēt programmu no atmiņas kartes, kas formatēta kādā no populārākajiem failu sistēmas formātiem, piemēram, FAT32 vai exFAT;
3. Nodrošināt dinamiskās atmiņas un laika plānošanas funkcionalitāti, kas atbilstu populāriem standartiem, piemēram, POSIX (attiecīgi, malloc un pthreads);
4. Nodrošināt dinamisko bibliotēku ielādes mehānismu;
5. Nodrošināt perifērijas iekārtu draiverus, kā dinamiskās bibliotēkas - obligāta prasība ir nodrošināt draiveris tieši:
 - a. skaņas saskarnei - ar atzvanīšanas funkciju;
 - b. tīkla saskarnei - transporta līmeni, bet iespējams arī augstākus - HTTP, OSC, u.c.;

- c. MIDI kontroles saskarnei, kuru nodrošinātu caur UART (vēsturiskajam pieslēgumam) vai USB;
 - d. failu sistēmai - failu ielādei un rakstīšanai no atmiņas kartes
6. Citas standarta bibliotēkas, lai nodrošinātu pirmkoda portēšanas iespējas no citām platformām.

Ņemot vērā šos kritērijus, šajā darbā tika veikts salīdzinājums programmas izstrādes ērtumam un veiktspējai starp Linux operētājsistēmu un programmatūru, kas darbināma tieši uz iekārtas, lai salīdzinātu vai operētājsistēmas sniegtās abstrakcijas atsver veiktspējas kritumu un citus radītos sarežģījumus

Linux gadījumā tika izstrādāta standarta komandrindas programma, kas iekļauta ELF formāta izpildāmajā failā un darbojas lietotāja atmiņā. Operētājsistēma šajā gadījumā jau nodrošina atmiņas pārvaldi, pavedienu un procesu pārvaldi un iekārtu draiverus.

Savukārt bez Linux programmatūras izpilde ir atkarīga no iekārtas arhitektūras - vai nu izpildāms mašīnkods ko ielasa no datu nesēja, vai tas tiek ierakstīts iekārtas zibatmiņā izmantojot programmēšanas iekārtas. Tāpat svarīgi nodrošināt programmēšanas bibliotēku kopu, kas pildītu minimālas operētājsistēmas abstrakcijas lomu, nodrošinot dinamiskās atmiņas pārvaldību, kooperatīvo vairākpavedienošanu un iekārtu draiverus.

7. PROCESORU ARHITEKTŪRU UN OPERĒTĀJSISTĒMU VEIKTSPĒJAS PĒTĪJUMS.

Darbā tika apskatīti daži no populārākajiem procesoriem, dažādi palīgprocesoru veidi un programmatūras veiktspēja operētājsistēmā Linux salīdzinājumā ar programmatūras darbību patstāvīgajā režīmā.

Ņemot vērā iepriekšējā nodaļā minētos faktorus šī darba pētnieciskiem nolūkiem tika izvēlēti sekojoši iekārtu, operētājsistēmu un kompilatoru komplekti (skat. tabulu 7.1.).

7.1. tabula

Pētījumā izmantotās procesoru, operētājsistēmu un kompilatoru kombinācijas

Iekārta	Operētājsistēma	Kompilators
SHARC Audio Module. Procesori: 1 ARMv7 un 2 SHARC Operatīvā atmiņa: 2 Gb	Bez	GCC un CrossCore C/C++ Compiler
SHARC Audio Module. Procesori: 1 ARMv7 un 2 SHARC Operatīvā atmiņa: 2 Gb	Buildroot Linux	GCC un CrossCore C/C++ Compiler
Raspberry Pi: Procesors: ARMv6 Operatīvā atmiņa: 512Mb	Bez	LLVM/Clang
Raspberry Pi: Procesors: ARMv6 Operatīvā atmiņa: 512Mb	Raspbian Linux	LLVM/Clang
Raspberry Pi un Xilinx Spartan 3E: Procesori: ARMv6 un FPGA Operatīvā atmiņa: 512Mb	Raspbian Linux	LLVM/Clang, Xilinx ISE priekš FPGA programmēšanas

Procesora jaudas pētījuma galvenais faktors ir sistēmas spēja veikt liela daudzuma peldošā punkta skaitļu aprēķinus noteiktā laika intervālā.

7.1. Diskrētā Furjē transformācija

Viens no svarīgākajiem algoritmiem signālu apstrādes sfērā ir diskrētā Furjē transformācija - to izmanto diskrēta signāla pārveidošanai no laika apgabala uz frekvenču apgabalu un otrādi. Šis algoritms balstās uz peldošā punkta aprēķiniem, kas ļoti noder dažādu procesoru veikspējas testēšanai.

Diskrētā Furjē transformācija tiek veikta pēc formulas:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot [\cos(2\pi kn/N) - i \cdot \sin(2\pi kn/N)],$$

kur:

N - signāla diskrēto punktu skaits,

n, k - signāla punkta un transformācijas punkta kārtas skaitļi,

x_n - signāla amplitūda punktā n,

X_k - transformētā amplitūda punktā k,

i - punkta imaginārā daļa.

Un tās inversā forma ir:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot [\cos(2\pi kn/N) + i \cdot \sin(2\pi kn/N)]$$

Tās realizācija C valodā attiecīgi:

```
typedef struct
{
    float re;
    float im;
} complex_float_t;

void dft(complex_float_t* in, complex_float_t* out, int num_samples, bool
inverse)
{
    int i, j;
    float a, f;
    complex_float_t s, w;
    f = inv ? 1.0 / num_samples : 1.0;
    for (i = 0; i < num_samples; i++)
    {
        s.re = 0;
        s.im = 0;
        for (j = 0; j < num_samples; j++)
        {
            a = -2 * M_PI * i * j / num_samples;
            if (inv)
            {
                a = -a;
            }
            w.re = cos(a);
            w.im = sin(a);
            s.re += in[j].re * w.re - in[j].im * w.im;
```

```

        s.im += in[j].im * w.re + in[j].re * w.im;
    }
    out[i].re = s.re * f;
    out[i].im = s.im * f;
}
}[35]

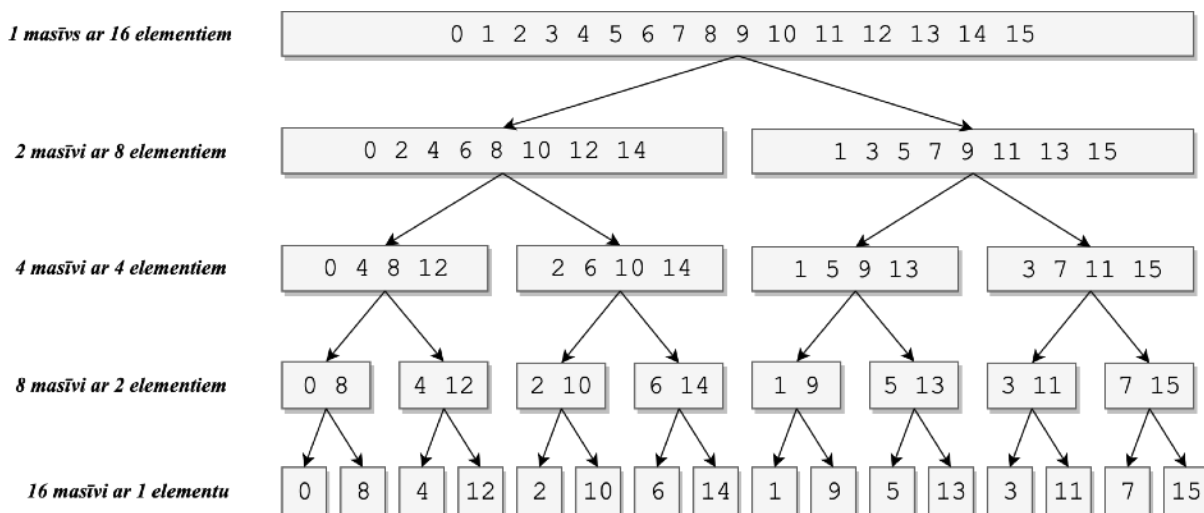
```

Šis konkrētais algoritms ir lēns ($O(n^2)$), kas labāk atspoguļotu procesora noslodzi peldošā punkta aprēķinu veikšanai jau pie maziem datu apjomiem.

7.2. Radix-2 ātrā Furjē transformācija

Kā otrs algoritms tika izvēlēts Radix-2 ātrā Furjē transformācija (pazīstams arī kā Cooley-Tukey), kuru ir viegli realizēt uz FPGA procesora un tā ātrums ir $O(n \log n)$.

Algoritms veic N elementu datu masīva decimāciju laikā vairākās iterācijās, sadalot to N atsevišķos 1 elementa masīvos. Decimācija tiek veikta masīva elementu indeksu binārās vērtības apgriežot spoguļskatā. Šāds risinājums ļauj veikt transformācijas tikai uz masīviem, kuru elementu skaits ir skaitļa 2 pakāpē.

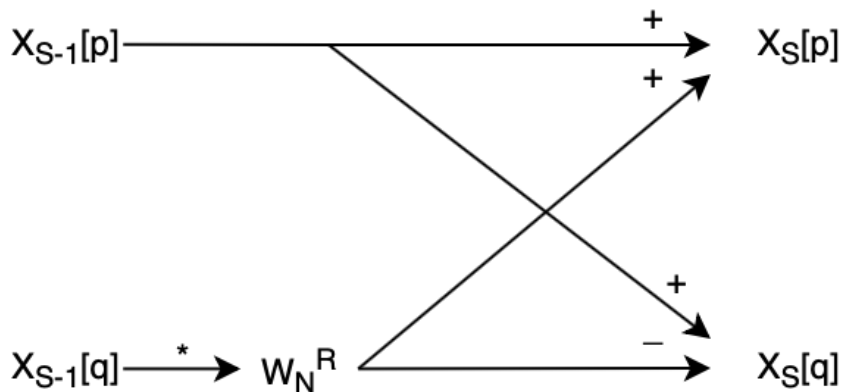


7.1. att. masīva elementu decimācija laikā

Tālāk šos masīvus izmanto, lai veiktu diskrēto Furjē transformāciju taruiņa aprēķina formā un izveidotu atpakaļ N elementu frekvenču apgabala masīvā.

**Iepriekšējā posma
vērtības**

Rezultāti



7.2. att tauriņa formas aprēķins

7.2. attēlā redzamais tauriņa aprēķins:

$$X_s[p] = X_{s-1}[p] + X_{s-1}[q] \cdot W_N^R$$

$$X_s[q] = X_{s-1}[p] - X_{s-1}[q] \cdot W_N^R, \text{ kur}$$

p - pāra elementa indekss

q - nepāra elementa indekss

$$W_N^R = \cos(2\pi R/N) - j \cdot \sin(2\pi R/N), \text{ kur :}$$

N - datu masīva garums,

R - atkarīgs no N, transformācijas iterācijas un tauriņa aprēķinu attāluma.

Algoritma inversā forma iegūstama invertējot elementu imaginārās daļas, veicot Furjē transformāciju un rezultātu elementiem invertējot atpakaļ imaginārās daļas un izdalot reālo daļu ar visu elementu skaitu masīvā.

Algoritma realizācija C valodā attiecīgi:

```
#define M_2PI 6.283185307179586476925286766559 // 2 * PI
#define M_LOG10_2_INV 3.3219280948873623478703194294948 // 1 / Log10(2)

void CMult(complex_float_t* dest, complex_float_t* c1, complex_float_t* c2)
{
    dest->re = (c1->re * c2->re) - (c1->im * c2->im);
    dest->im = (c1->re * c2->im) + (c1->im * c2->re);
}

void CAdd(complex_float_t* dest, complex_float_t* c1, complex_float_t* c2)
{
    dest->re = c1->re + c2->re;
    dest->im = c1->im + c2->im;
}

void CSub(complex_float_t* dest, complex_float_t* c1, complex_float_t* c2)
{
    dest->re = c1->re - c2->re;
    dest->im = c1->im - c2->im;
}
```

```

void Radix2iFFT(complex_float_t *in, complex_float_t *out, int N)
{
    double iN = 1 / (double)N;
    complex_float_t* pIn = in;
    for (int i = 0; i < N; i++, pIn++)
    {
        pIn->im *= -1;
    }

    Radix2FFT(in, out, N);

    complex_float_t* pOut = out;
    for (int i = 0; i < N; i++, pOut++){
        pOut->re *= iN;
        pOut->im *= -1;
    }
}

void Radix2FFT(complex_float_t *in, complex_float_t *out, int N)
{
    int stages = (int)ceil(log10((double)N) * M_LOG10_2_INV);

    // Decimation - sort in samples to out samples
    complex_float_t* out_start = out;
    for (int i = 0; i < N; i++, out++)
    {
        complex_float_t* x = in + i;
        unsigned int i_out = 0;
        unsigned int i_in = i;
        for (int j = 0; j < stages; j++)
        {
            if (i_in & 0x01)
            {
                i_out += (1 << (stages - 1 - j));
            }
            i_in >>= 1;
            if (!i_in)
            {
                break;
            }
        }
        out = out_start + i_out;
        out->re = x->re;
        out->im = x->im;
    }

    // FFT Computation by butterfly calculation
    complex_float_t WN;
    complex_float_t tmp;
    for (int stage = 1; stage <= stages; stage++)
    {
        int butterfly_separation = (int)(pow(2, stage));
        int butterfly_width = butterfly_separation / 2;
        double P = M_2PI / butterfly_separation;
        for (int j = 0; j < butterfly_width; j++)
        {
            if (j != 0)
            {
                WN.re = cos(P * j);
                WN.im = -sin(P * j);
            }
            for (int k = j; k < N; k += butterfly_separation)
            {
                complex_float_t* pHi = out_start + k;
                complex_float_t* pLo = pHi + butterfly_width;

```

```

        if (j != 0)
        {
            CMult(&tmp, pLo, &WN);
        }
        else
        {
            tmp.re = pLo->re;
            tmp.im = pLo->im;
        }
        CSub(pLo, pHi, &tmp);
        CAdd(pHi, pHi, &tmp);
    }
}
}

```

[36]

7.3. Eksperimentālais uzstādījums

Eksperimentālais uzstādījums sastāv no:

1. 45056 punktu skaņas līknes punktu masīva, kas imitē aptuveni 1 sekundi garu skaņas paraugu pie diskretizācijas frekvences 44,1kHz;
2. Dažāda garuma fragmentēšanas masīviem (no 16 līdz 1024 punktiem), kuri iet cauri datu masīvam un veic Furjē un inversās Furjē transformācijas;
3. Vēlams, ka šīs darbības iekļautos 1 sekundes intervālā, tādējādi radot iespaidu, ka sistēma šo aprēķinu varētu veikt reālā laikā.

Pati eksperimenta programma ir veidota tā, lai tā varētu izpildīt šīs darbības gan lokāli, gan attālināti - uz citiem procesora kodoliem, kā arī paredz iespēju veikt šīs darbības paralēli uz vairākiem kodoliem vienlaicīgi.

Programma mēra reālo laiku un procesora ciklu skaitu un izrēķina vidējo laika patēriņu, kas nepieciešams veicot transformācijas visam datu masīvam.

Paralēli procesora veikspējas pētījumam tika veikts arī strāvas patēriņa pētījums, kas tika veikts mērot vidējo strāvas stiprumu, ievietojot ampērmetru ķēdē starp iekārtu un strāvas avotu. Iegūto rezultātu ievietojot Vata likuma formulā:

$$P = I \cdot V,$$

kur:

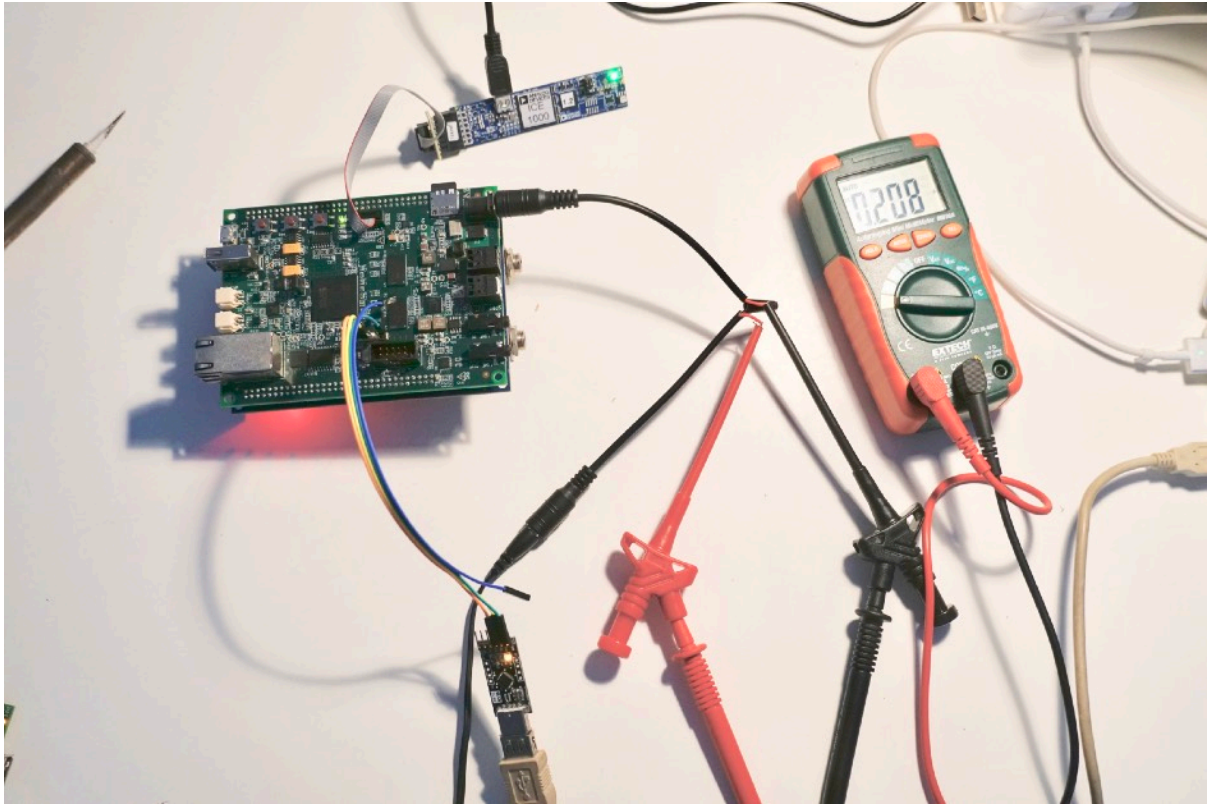
P - jauda (W),

I - strāvas stiprums (A),

V - strāvas spriegums (V).

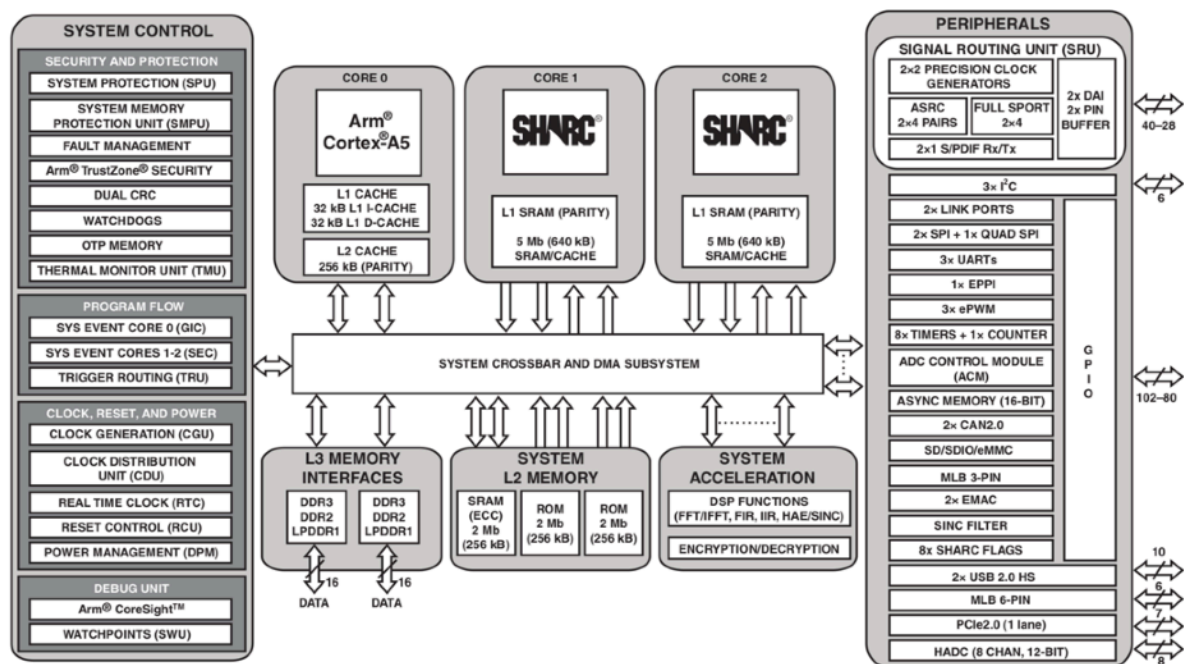
No tā tālāk iespējams aprēķināt, vai būtu iespējams šo iekārtu apgādāt ar strāvu, izmantojot standarta litija polimēra akumulatoru ar ietilpību 1800 mAh, kas dod spriegumu 7,4V.

7.4. SHARC Audio Module



7.1. att. *Analog Devices SHARC Audio Module ar ADZS-AUDIOPROJECT paplašinājumu, ICE-1000 JTAG programmētāju un CP2102 seriālo datu saskarni.*

Analog Devices SHARC Audio Module ir miniatūra izmēra dators, kas paredzēts SHARC procesoru testēšanai un izstrādei. Tas sastāv no *ADSP-SC589* procesora ar vienu ARM arhitektūras kodolu un diviem SHARC arhitektūras kodoliem, kas visi darbojas ar 450MHz takts frekvenci. Pamata sistēmas ielāde notiek ar ARM kodola palīdzību, kurš tālāk iedarbina pārējos divus kodolus.



7.2. att. **ADSP-SC589 procesora komponentu shēma** [37]

Darba ietvaros tika izmantota *Analog Devices* piedāvātā *CrossCore Embedded Studio* izstrādes vide gan Linux, gan Microsoft Windows operētājsistēmās, jo veikt izstrādi *Analog Devices* izmantotajam *Buildroot Linux* ir nepieciešams izmantot izstrādes rīku ķēdi, kas darbojas tikai Linux vidē uz x86 arhitektūras procesora, savukārt, lai izstrādātu programmas, kas darbosies uz SHARC kodoliem, un tās ielādētu SPI zibatmiņā, jāizmanto šī pati izstrādes vide, bet Microsoft Windows operētājsistēmā, jo SHARC izstrādes rīku ķēde un JTAG programmētāja rīki ir pieejami tikai šajā operētājsistēmā.

Bez operētājsistēmas izstrādei *Analog Devices* piedāvā *SHARC Bare Metal Framework*, kas nodrošina visas pamata funkcijas darbam ar *SHARC Audio Module* un visiem tā kodoliem.

Saziņu starp kodoliem nodrošina MCAPI standarta bibliotēka, kura piedāvā trīs veidu saziņu - bez-savienojuma ziņu apmaiņu, pakešu apmaiņu un skalāro lielumu apmaiņu. Diemžēl oficiāli atbalstīta pagaidām tiek tikai ziņu apmaiņas specifikācija. Vēl lielāku apgrūtinājumu uzliek ziņu izmēra ierobežojums, kas *Buildroot Linux* sistēmā ir 1024 baiti, bet SHARC programmēšanas bibliotēkās ir vēl zemāks - 256 baiti, jeb 29 kompleksie peldošā punkta skaitļi un ziņas galvene, kas liek skaņas buferus sūtīt vairākās ziņās un iespējami uzliek lielāku slodzi datu apmaiņai.

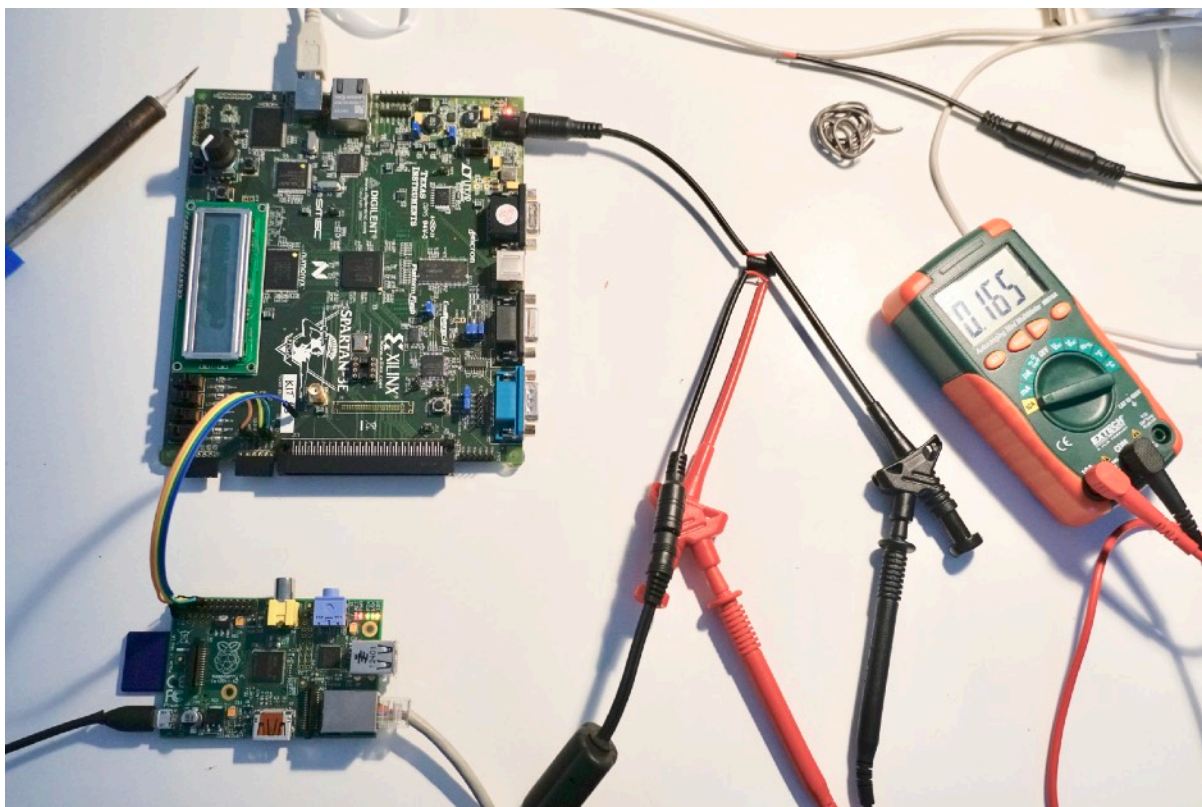
Tāpat pētījuma laikā tika secināts, ka MCAPI ziņu apmaiņas protokols uz *Buildroot Linux* nav stabils un lielu datu apjoma gadījumā, tas pārstāj strādāt, iespējamās maskētas

klūdas (*Heisenbug*) dēļ. Kā arī visi mēģinājumi nodibināt savienojumu ar otru SHARC kodolu no *Buildroot Linux* beidzās neveiksmīgi.

Visu kombināciju, izņemot paralēlo SHARC izpildi no Linux vides, kuru nezināmu tehnisku iemeslu dēļ neizdevās realizēt, jo neizdevās nodibināt savienojumu ar otro SHARC kodolu, vidējos izpildes laika intervālus var redzēt tabulās 7.6.1. un 7.6.2. - kolonnas attēlo izpildes laiku sekundēs atkarībā no fragmentācijas masīvu izmērus. Tāpat SHARC izpilde no Linux vides izpildes laiki tika ekstrapolēti par pamatu ņemot līdzīgo kombināciju SHARC izpildei no ARMv7 kodola bez operētājsistēmas, jo pie lielākiem fragmentācijas buferu izmēriem un palielinot datu apjomu, kas tika pārsūtīts caur MCAPI, tas pārstāja strādāt, iespējamās maskētas klūdas (*Heisenbug*) dēļ.

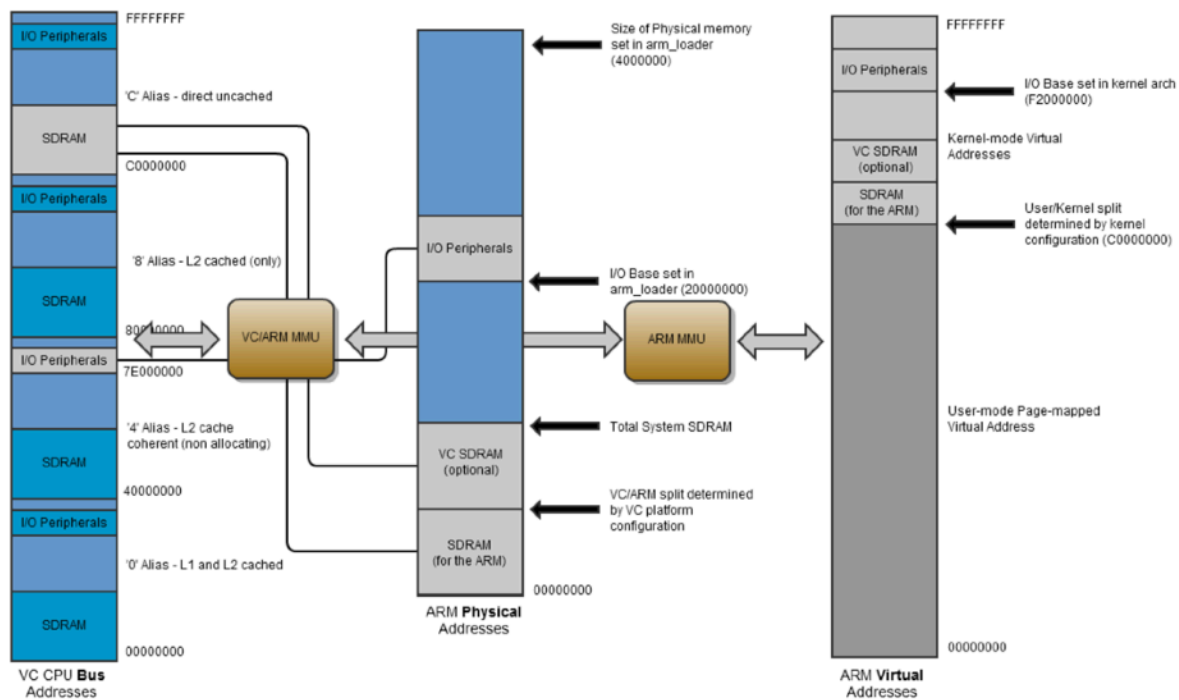
Strāvas patēriņa mērījums tika veikts izmantojot speciālu pāreju, kas ļauj ampērmetru ieslēgt strāvas barošanas ķēdē. Strāvas patēriņš tika mērīts tukšgaitas un slodzes (izpildot testa algoritmus) režīmos.

7.5. Raspberry Pi un Xilinx Spartan 3E Starter Kit



7.4. att. *Raspberry Pi* (pirmās paaudzes) un *Xilinx Spartan 3E Starter Kit* ar *FX2-BB* paplašinājumu.

Raspberry Pi ir miniatūra izmēra dators, kas darbojas ar *Broadcom BCM2835* mikrokontroliera, kas savukārt sastāv no *VideoCore IV* grafiskā kodola un ARMv6 arhitektūras palīgprocesora kodola, kas darbojas ar 700MHz takts frekvenci.



Salīdzinājumā ar *SHARC Audio Module*, *Raspberry Pi* nav nepieciešama specifiska izstrādes vide, jo Linux operētājsistēma *Raspbian* nodrošina pilnu izstrādes rīku ķēdi uz pašas iekārtas, tāpat arī ir iespēja darbināt izpildāmo kodu pa taisno uz ARM procesora ielādējot izpildāmo kodu caur SD atmiņas karti un norādot grafiskā procesora sistēmas ielādes programmai ceļu uz testa programmu.

Šī darba nolūkos tika izmantota gatava ātrā diskrētās Furjē transformācijas risinājums Verilog valodā no Xilinx piedāvātā IP Core kataloga - tika izmantota Radix-2 FFT risinājuma 7.1. versija. FPGA atšķirībā no parasta instrukciju izpildes procesora darbojas ar tīriem digitāliem signāliem un to dara ātri un paralēli, līdz ar to tas dod iespēju ieprogrammēt signāla

apstrādes loģiku, tad caur seriālo datu saskarni padoto tīru apstrādājamo datu straumi un uzreiz saņemot atpakaļ apstrādāto rezultātu.

Savienojums starp *Raspberry Pi* un *Xilinx Spartan 3E Starter Kit* nodrošina UART seriālā saskarne, kas vienlaicīgi ir arī visas sistēmas vājākais punkts, jo datu caurlaides ātrums ir ļoti lēns (pieejamais maksimums bija 4Mbit/s, jeb 500KB/s). Reāla produkta gadījumā vislabākais risinājums būtu izmantot FPGA uz PCIe kartes (ātrums līdz pat 1000MB/s), kuru diemžēl nav iespējams izmantot ar *Raspberry Pi*, jo tam nav attiecīgās pieslēgvietas.

Pētījuma ietvaros programmatūra tika darbināta sekojošās kombinācijās:

1. Kā Linux aplikācija uz ARMv6 kodola;
2. Tīrā ARMv6 mašīnkodā;
3. Kā Linux aplikācija, kas nosūta datus apstrādei uz FPGA kodolu caur UART pieslēgvietu.

Papildus tika veikti tukšgaitas testi, kur UART dati tika nosūtīti uz FPGA un uzreiz atgriezti, tādējādi novērtējot visu ķēdes aizturi un dodot iespēju izdalīt paša FPGA izpildes laiku.

Strāvas patēriņa mērījums tika veikts atsevišķi *Raspberry Pi* un *Xilinx Spartan 3E Starter Kit* platēm, jo tās katra tika barotas ar atsevišķiem barošanas blokiem. Mērījums tika veikts izmantojot speciālas pārejas, kas ļauj ampērmētru ieslēgt strāvas barošanas ķēdē un tas tika atkārtots tukšgaitā un pie slodzēm izpildot testa algoritmus.

8.

MĒRĪJUMU REZULTĀTI

Pētījuma ietvaros programmatūra tika darbināta sekojošās kombinācijās:

1. Kā Linux aplikācija uz ARMv7 kodola uz SHARC Audio Module;
2. Kā Linux aplikācija, kas aprēķinus deleģē SHARC kodolam nosūtot datus un komandas izmantojot MCAPI ziņu pārsūtīšanas protokolu;
3. Kā Linux aplikācija, kas aprēķinus deleģē diviem paralēliem SHARC kodoliem, datus un komandas nosūtot izmantojot MCAPI ziņu pārsūtīšanas protokolu;
4. Tīrā ARMv7 mašīnkodā uz SHARC Audio Module;
5. Tīrā SHARC mašīnkodā;
6. Tīrā ARMv7 mašīnkodā, kas aprēķinus deleģē SHARC kodolam nosūtot datus un komandas izmantojot MCAPI ziņu pārsūtīšanas protokolu;
7. Tīrā ARMv7 mašīnkodā, kas aprēķinus deleģē diviem paralēliem SHARC kodoliem datus un komandas nosūtot izmantojot MCAPI ziņu pārsūtīšanas protokolu;
8. Kā Linux aplikācija uz ARMv6 kodola uz Raspberry Pi;
9. Kā Linux aplikācija, kas aprēķinus deleģē Xilinx Spartan 3E FPGA procesoram caur UART ar ātrumu 4MBit/s - tikai Radix-2 FFT algoritms;

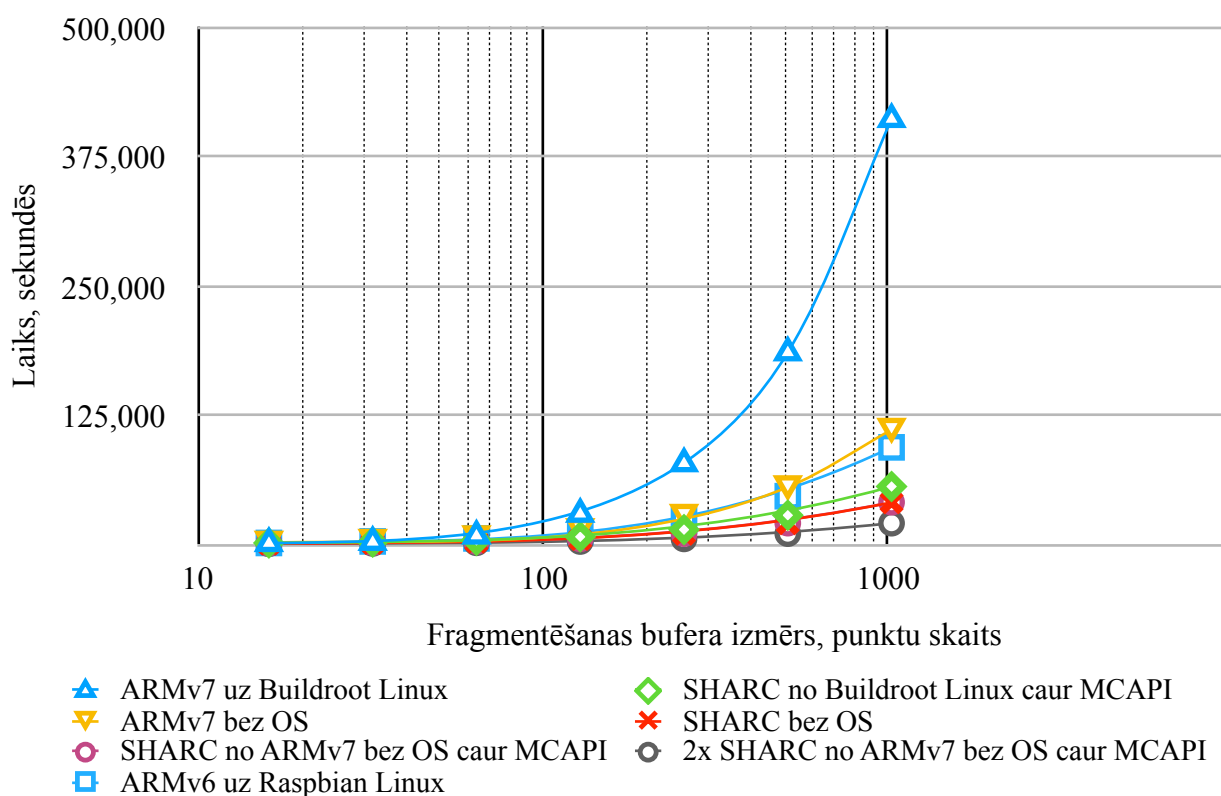
Papildus uz Raspberry Pi tika veikts UART apkārtceļa veiktspējas tests, lai noteiktu UART savienojuma ietekmi uz datu apmaiņu. Mērījumu vidējie rādītāji bija:

- 7,42 sekundes pie UART ātruma 1MBit/s;
- 1,86 sekundes pie UART ātruma 4MBit/s.

Ņemot vērā šos rādījumus tabulā 7.6.2 uzrādītie rezultāti ar FPGA procesoru ir uzrādīti ar un bez šīs apkārtceļa laika.

Diskrētās Furjē transformācijas izpildes laiki dažādās procesoru un operētājsistemu kombinācijās (rindas) pie dažādu masīvu lieluma (kolonnas)

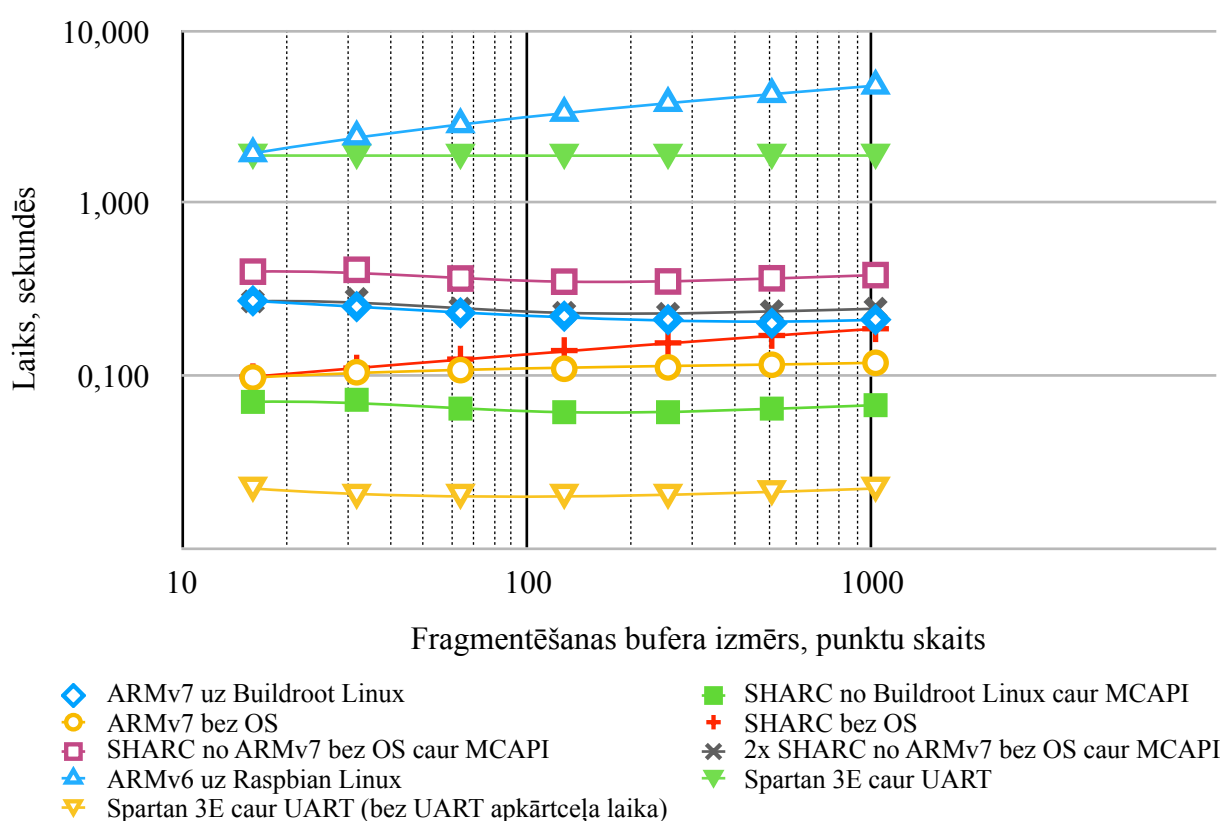
	16	32	64	128	256	512	1024
1.	1,150	2,310	8,390	29,490	78,340	185,610	411,310
2.	1,280	2,145	3,802	7,238	14,171	28,076	55,887
4.	1,691	3,445	6,918	13,833	27,647	55,298	110,838
5.	0,630	1,262	2,528	5,060	10,123	20,248	40,498
6.	0,932	1,562	2,768	5,270	10,318	20,443	40,693
7.	0,534	0,853	1,445	2,690	5,211	10,274	20,399
8.	1,290	2,770	5,700	11,520	23,120	46,510	93,750



7.6.1. att. *Diskrētās Furjē transformācijas algoritma izpildes salīdzinājums dažādās operētājsistemu un procesoru kodolu konfigurācijās.*

Radix-2 Furjē transformācijas izpildes laiki dažādās procesoru un operētājsistēmu kombinācijās (rindas) pie dažādu masīvu lieluma (kolonnas)

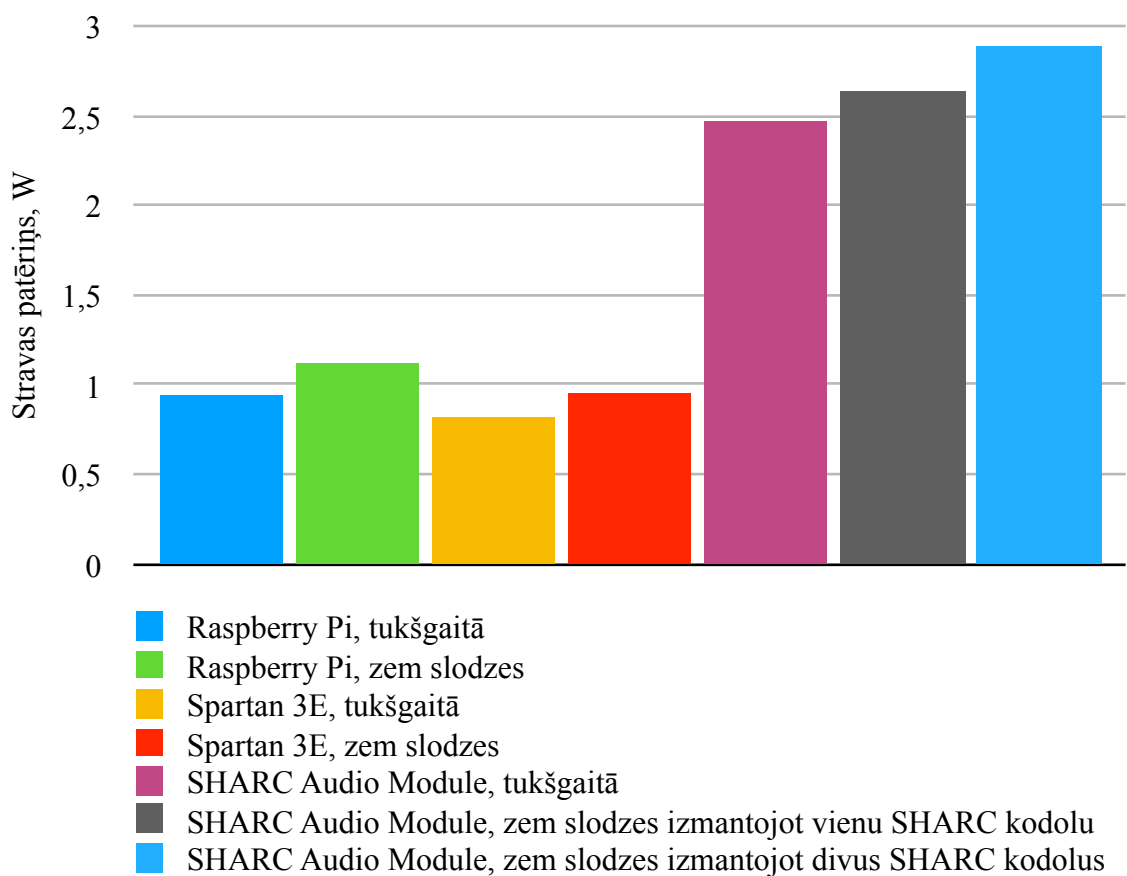
	16	32	64	128	256	512	1024
1.	0,270	0,250	0,230	0,220	0,210	0,200	0,210
2.	0,070	0,072	0,064	0,061	0,061	0,064	0,067
4.	0,097	0,104	0,107	0,109	0,111	0,115	0,118
5.	0,098	0,110	0,124	0,139	0,154	0,170	0,186
6.	0,399	0,409	0,364	0,349	0,350	0,365	0,381
7.	0,268	0,277	0,243	0,230	0,227	0,235	0,243
8.	1,940	2,430	2,860	3,330	3,800	4,280	4,790
9.	1,882	1,880	1,880	1,880	1,880	1,881	1,882
9.*	0,022	0,020	0,020	0,020	0,020	0,021	0,022



7.6.2. att. Ātrās Furjē transformācijas algoritma izpildes salīdzinājums dažādās operētājsistēmu un procesoru kodolu konfigurācijās.

Strāvas patēriņa mērījumu rezultāti

	Stiprums, mA	Spriegums, V	Patēriņš, W
Raspberry Pi, tukšgaitā	189,000	5,0	0,945
Raspberry Pi, zem slodzes	225,000	5,0	1,125
Spartan 3E, tukšgaitā	165,000	5,0	0,825
Spartan 3E, zem slodzes	190,000	5,0	0,950
SHARC Audio Module, tukšgaitā	206,000	12,0	2,472
SHARC Audio Module, zem slodzes izmantojot vienu SHARC kodolu	220,000	12,0	2,640
SHARC Audio Module, zem slodzes izmantojot divus SHARC kodolus	241,000	12,0	2,892

7.6.3. att. *Dažādu iekārtu vidējais strāvas patēriņš tukšgaitā un zem slodzes.*

SECINĀJUMI

Darbā izmantotās iekārtas viennozīmīgi pierāda, ka to izmēri (izņemot varbūt *Xilinx Spartan 3E Starter Kit*), funkcionalitāte un strāvas patēriņš varētu būt risinājums digitālo mūzikas instrumentu un skaņas apstrādes efektu miniaturizēšanai. Lietošanas ērtuma jautājums, gan paliek neizskatīts, jo tas ir atkarīgs gan no arhitektūras vadlīnijām, gan lietotāja programmatūras, gan izstrādātāju veikuma.

Lai arī sāknēšanās laiks *Raspbian Linux* (~50 sekundes) un *Buildroot Linux* (~9 sekundes + 5 sekundes u-boot gaidīšanas laiks) ir relatīvi mazs, tas nespēj konkurēt ar programmatūras sāknēšanu pa tiešo uz procesora bez operētājsistēmas, kas *Raspberry Pi* gadījumā ir ~2 sekundes, bet SHARC Audio Module gadījumā zem 1 sekundes. Šie rādītāji ir perfekti radīti dzīvā izpildījuma situācijām, kad iekārta zaudē strāvas avotu un ir nepieciešams to palaist no jauna pēc iespējas ātrāk.

No šiem risinājumiem visgatavākais risinājums šāda tipa produktiem ir *SHARC Audio Module*, pirmkārt jau tādēļ, ka tas ir izstrādāts ar līdzīgu mērķi. Tam jau ir pieejami speciāli paplašinājumi ar MIDI un profesionāla līmeņa skaņas ieejām un izejām. Tāpat Analog Devices tam nodrošina zema līmeņa bibliotēkas darbam pa taisno ar procesoru un perifērijām, kā arī jau gatavu skaņu efektu būvēšanas bloku funkcijas. Vienīgā vieta uzlabojumiem būtu paplašināt šo bibliotēku funkcionalitāti un izstrādāt ērtu ielādes programmu, kas spētu ielādēt virtuālo mūzikas instrumentu un skaņu efektu programmas caur USB vai no SD kartes.

No veikspējas testiem var secināt, ka FPGA ir viennozīmīgi galvas tiesu pārrāks par jebkuru risinājumu, gan izpildes laika, gan strāvas patēriņa ziņā.

Interesanti, ka veikspējas testos programma, kas darbojas pa tiešo uz ARMv7 kodola izpildījās lēnāk nekā darbojoties no Linux operētājsistēmas, tam skaidrojumi varētu būt vairāki:

1. Atšķirīgas kompilatoru optimizācijas;
2. Tīrā mašīnkoda programma neveica nekādu procesora optimizāciju (kešatmiņu, u.c.) konfigurāciju;
3. Iespējama kļūda laika mērīšanā - tīrajā mašīnkodā pēc katra testa cikla tika nolasīts procesora instrukciju skaitītājs, savukārt Linux operētājsistēmu gadījumā tika izmantots sistēmas nodrošinātā POSIX times() funkcionalitāte, kas dažādu optimizāciju dēļ (piemēram pārāk dārgo pārslēgšanos starp kodola un lietotāja atmiņas apgabaliem) nenodrošina augstas precizitātes pulksteņa granularitāti.

Tāpat interesanti, ka uz *Raspberry Pi* ARMv6 procesora DFT izpildījās ātrāk par *SHARC Audio Module* ARMv7 procesoru, bet turpretī Radix-2 FFT algoritms aptuveni 10 reizes lēnāk. Iespējamais iemesls varētu būt ARMv7 uzlabotais peldošā punkta vektoru palīgprocesors un iespējamās kompilatora optimizācijas, kas to izmanto.

Cenu ziņā gan *SHARC Audio Module* ir samērā dārga - pamata plate maksā 195 USD, bet paplašinājums mūzikas projektiem maksā 95 USD. Salīdzinājumam jaunākie *Raspberry Pi* (trešās paaudzes) maksā sākot no 35 EUR. Tiesa gan jebkurš Xilinx FPGA risinājums ir pieejams tikai sākot no 250 USD (par Arty A7-100).

No pētījuma rezultātiem, varam secināt, ka praktiskākā pieeja būtu izmantot standarta procesoru arhitektūru, bez operētājsistēmas un papildus FPGA vai SHARC procesoru kā aprēķinu paātrinātājus vai gatavas signāla apstrādes un ģenerēšanas līnijas. Vislabākais kandidāts viennozīmīgi ir *SHARC Audio Module*, jo tas piedāvā standarta ARM procesoru ar iebūvētiem SHARC palīgprocesoriem.

SHARC procesora arhitektūras galvenās priekšrocības ir instrukciju paralelizācija un ciklu instrukcijas, kas ļoti uzlabo aprēķinu ātrdarbību tāda tipa algoritmiem kā diskrētā Furjē transformācija.

FPGA procesoru arhitektūras galvenā priekšrocība ir to spēja tapt pārkonfigurētiem par jebkāda cita veida digitālā signāla apstrādes procesoriem.

Strāvas patēriņu ziņā abi risinājumi bija teju līdzvērtīgi - jāņem vērā, ka FPGA tika mērīts atsevišķi no *Raspberry Pi*, līdz ar to summārais patēriņš bija tuvs *SHARC Audio Module* patēriņam - t.i. vidēji 2W *Raspberry Pi* un vidēji 2,4W *SHARC Audio Module* gadījumā. Ar šādu patēriņu 1800mAh baterija, kas dod spriegumu 7,4V, varētu barot šīs iekārtas no 5h 30min līdz 6h 40min, kas ir pietiekams laiks, lai nospēlētu dzīvo koncertu (~2h).

Pārsteidzošs secinājums bija, ka UART spēj stabili darboties ar ātrumu 4MBit/s (pat ar samērā resniem un 10cm gariem vadiem), kas ir pats minimums, lai nodrošinātu 1 kanāla 16-bitu 44,1kHz audio straumes pārsūtīšanu turp un atpakaļ (ieskaitot start un stop bitus, nepieciešami 3,5MHz). Darbā gan tika lietoti kompleksie skaitļi, kas ir 4x lielāki par 16-bitu veselajiem skaitļiem, līdz ar to caurlaidība nebija pietiekama, lai nodrošinātu reālā laika datu apstrādi. Lai nodrošinātu labāku datu caurlaidību, būtu jāizmanto kāds cits pieslēguma standarts, piemēram PCIe, kas spēj attīstīt līdz pat 1GBit/s datu pārraides ātrumu.

LITERATŪRAS SARAKSTS

1. The Complete MIDI 1.0 Detailed Specification (<https://www.midi.org/specifications-old/item/the-midi-1-0-specification>, 1996. gads)
2. Universal Serial Bus Device Class Definition for MIDI Devices (<https://www.usb.org/sites/default/files/midi10.pdf>, 1999. gads)
3. The Open Sound Control 1.0 Specification (http://opensoundcontrol.org/spec-1_0, 2002. gads)
4. Compact Disc Red Book, Phillips & Sony, 1982. gads
5. Wikipedia: Pulse-code modulation (https://en.wikipedia.org/wiki/Pulse-code_modulation, 2019. gads)
6. VST3 Standard (<https://www.steinberg.net/en/company/technologies/vst3.html>, 2008. gads)
7. LV2 Standard (<http://lv2plug.in/ns/>, 2018. gads)
8. Pure Data grafiskā programmēšanas valoda (<https://puredata.info>)
9. Max/MSP grafiskā programmēšanas valoda (<https://cycling74.com/products/max/>)
10. Intel® 64 and IA-32 Architectures Software Developer's Manual (<https://software.intel.com/en-us/download/intel-64-and-ia-32-architectures-sdm-combined-volumes-1-2a-2b-2c-2d-3a-3b-3c-3d-and-4>, 2018. gads)
11. ARM® Architecture Reference Manual ARMv8, for ARMv8-A architecture profile (https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf, 2018. gads)
12. ARM Architecture Reference Manual Supplement - The Scalable Vector Extension (SVE), for ARMv8-A (<https://developer.arm.com/products/architecture/cpu-architecture/a-profile/docs/arm-architecture-reference-manual-supplement-armv8-a>, 2018. gads)
13. ARM Architecture Reference Manual Supplement - ARMv8, for the ARMv8-R AArch32 architecture profile (https://static.docs.arm.com/ddi0568/a/DDI0568A_b_armv8_r_supplement.pdf, 2018. gads)
14. Qualcomm APQ8053 processors (<https://www.qualcomm.com/products/apq8053>)
15. SHARC+ Core Programming Reference (<https://www.analog.com/media/en/dsp-documentation/processor-manuals/SC58x-2158x-prm.pdf>, 2017. gads)
16. Realtime finite difference physical models of musical instruments on a field programmable gate array (FPGA), Florian Pfeifle, Rolf Bader, 2012, 15th Int. Conference on Digital Audio Effects.
17. Return Infinity BareMetal exo-kernelis (<https://gitlab.com/ReturnInfinity/BareMetal>)
18. Wine (<https://www.winehq.org/about>)
19. Kernel support for live digital audio and video, Kevin leffay, Donald L Stone, F Donelson Smith, 1992, Butterworth-Heinemann Ltd.

20. The Architecture of IRCAM Musical Workstation, Eric Lindemann, François Dechelle, Bennett Smith, Michael Stariker, 1991, MIT Press
21. Overcoming Workstation Scheduling Problems in a Real-Time Audio Tool, Isidor Kouvelas, Vicky Hardman, 1997, USENIX Annual Technical Conference
22. Realtime finite difference physical models of musical instruments on a field programmable gate array (FPGA), Florian Pfeifle, Rolf Bader, 2012, 15th Int. Conference on Digital Audio Effects.
- 23 Design Methodology for Real-Time FPGA-Based Sound Synthesis, Erdem Motuk, Roger Woods, Stefan Bilbao, John McAllister, 2007, IEEE
24. SM Pro Audio V-Machine (<http://www.smproaudio.com/index.php/en/products/v-machines/v-machine>)
25. Seelake AudioStation X64 (<http://www.seelake.com/en/as-overview.html>)
26. Muse Research Receptor (<https://museresearch.com/products/index.php>)
27. MOD Duo (<https://www.moddevices.com/products/mod-duo>)
28. UAD Apollo (<https://www.uaudio.com/audio-interfaces/apollo-x6.html>)
29. Elk (<https://www.mindmusiclabs.com>)
30. ASV patents Nr. US20110316793A1 (<https://patents.google.com/patent/US20110316793A1/en>)
31. ASV patents Nr. US20100269670A1 (<https://patents.google.com/patent/US20100269670A1/en>)
32. ASV patents Nr. US20020108484A1 (<https://patents.google.com/patent/US20020108484A1/en>)
33. GNU Compiler Collection (<https://gcc.gnu.org>)
34. LLVM (<https://llvm.org>)
35. Gusta Kakša maģistra darba GIT pirmkoda glabātuve (<https://github.com/gusc/Masters-Thesis/blob/master/Source/DFT.c>)
36. Gusta Kakša maģistra darba GIT pirmkoda glabātuve (<https://github.com/gusc/Masters-Thesis/blob/master/Source/Radix2FFT.c>)
37. Analog Devices ADSP-SC589 specifikācija (<https://www.analog.com/en/products/ADSP-SC589.html>)
38. Broadcom BCM2835 procesoru atmiņas apgabalu abstrakciju shēma (<https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf>)