TALLINNA
TEHNIKAÜLIKOOL

INFOTEHNOLOOGIA TEADUSKOND
Automaatikainstituut
Automaatjuhtimise ja süsteemianalüüsi õppetool

Alfred-Arnold Liin

FIR-filtrite rakendamine helisüsteemides

Magistritöö

Juhendaja: Olev Märtens
juhtivteadur

Tallinn
2014

AUTORIDEKLARATSIOON

Olen koostanud magistritöö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Autor: Alfred-Arnold Liin
Kuupäev:
Allkiri:

Eessõna

Magistritöö teema on inspireeritud Olev Märtensi õppeainest “Sensorsignaalitöötlus” (IEM0250), samuti Paul Annuse õppeainest “Andmehõivesüsteemid” (IEM3040). Loengutes käsitletud temaatika põimus isikliku huviga arendada praktilisi helisüsteeme, eriti seoses helisignaali töötlemisega.

Autor soovib tänada Olev Märtensit teoreetilise baasi tugevdamise eest antud valdkonnas ja vajaliku riistvara võimaldamise eest platvormi realiseerimisel. Samuti praktiliste näpunäidete eest DSP programmkoodi kirjutamisel.

Lisaks soovib autor tänada GNU/Linux operatsioonisüsteemi arendajaid, kelle töö tulemusena on projekt dokumenteeritud vabavara abil. Ühtlasi tänab autor GNU “Octave” tarkvara arendajaid, kes võimaldasid projektis käsitletavate signaalitöötlusega seotud arvutuste teostamist ja vastavate tulemuste visualiseerimist vabavaraliselt.

Annotatsioon

Mikroskeemide ja -protsessorite tormiline areng on loonud eeldused digitaalse signaalitöötluse realiseerimiseks helisüsteemides. Käesolev töö keskendub platvormi loomisele digitaalsete FIR-filtrite rakendamiseks helisüsteemides piiratud energiavajaduse ja rahalise ressursi tingimustes. Platvormi riistvara põhineb "TMS320C5505 eZdsp USB Stick" arendusplaadil, mis on toodetud Texas Instruments Inc. poolt. Märkimisväärne osa platvormi tarkvaralisest lahendusest põhineb TI näitekoodi "C5505 eZdsp Audio Filter Demo" raamistikul.

Digitaalse signaalitöötluse ja FIR-filtrite teoreetilisi aluseid on varasemalt dokumenteeritud väga suures mahus. Käesolevas töös on kirjeldatud vaid valikuliselt disainitava süsteemi teoreetilisi aluseid, sest täieliku ülevaate andmine ei mahuks selle magistritöö piiridesse. Autor eeldab, et lugeja on kursis näiteks terminitega ühikimpulss, Nyquisti sagedus ja digitaal-analoog muundamine.

Töö algab ülesandepüstitusega, millele järgneb platvormi kui LTI-süsteemi kirjeldus ning konvolutsiooni mõiste tutvustamine digitaalse signaalitöötluse alustalana. Järgnevalt on formuleeritud matemaatilised alused FIR-filtrite realiseerimiseks ja praktilised näited kõige lihtsamate FIR-filtrite disainimiseks "kaalutud akna" meetodil. Põgusalt on käsitletud ka alternatiivseid algoritme FIR-filtrite disainimiseks.

Töö kolmas peatükk annab ülevaate kasutatavast DSP riistvarast ja selle tööpõhimõtetest. Samuti on kirjeldatud digitaalsete helisigaalide edastamiseks kasutatavate andmesideprotokollide (I2S ja SPDIF) olemust ja transporditavate signaalide ajastamist ning kodeerimist. Vastavalt andmesideprotokollide iseärasustele on toodud ka SPDIF-vastuvõtja ja DAC liidestamise teostus DSP-ga.

Neljas peatükk keskendub platvormi tarkvaralisele lahendusele alustades TI CCS arenduskeskkonna kasutamise põhitõdedest. Platvormi tarkvaralise lahenduse kirjeldus ei sisalda kommenteeritud programmkoodi, vaid tarkvaraliste funktsioonide ning nende vaheliste seoste ja andmevahetusprotsesside kirjeldust üldisemalt. Programmkood on esitatud koos tööga digitaalsel kujul.

Viimaks on toodud lihtsustatud katsetulemused platvormi töövõimelisuse hindamiseks. Põgusalt on käsitletud ka võimalusi projekteeritud platvormi tarkvaraliseks edasiarenduseks ning alternatiivseid võimalusi digitaalsete FIR-filtrite realiseerimiseks helisüsteemides.

Lõputöö on kirjutatud eesti keeles ning sisaldab 71 lehekülge teksti, 7 peatükki, 24 joonist, 1 tabelit.

Abstract

Application of FIR Filters to Audio Systems

Recent developments in the field of microchips and microprocessors have made it possible to integrate digital signal processing tasks to consumer-audio systems. The aim of this thesis is to design a platform to apply digital FIR filtering to audio systems in the presence of limited financial resources and minimal energy requirements. The hardware of the platform is based on the "TMS320C5505 eZdsp USB Stick" development board produced by Texas Instruments. The software developed for the board is mostly based on the framework of the "C5505 eZdsp Audio Filter Demo" source code by TI.

The theoretical basis of digital signal processing and design of FIR filters has been well-documented over the years. The thesis only contains a minimal set of theoretical issues regarding the designed platform, because giving a thorough theoretical overview would be out of scope of the thesis. The author assumes that the reader is familiar with the terms e.g. delta impulse, Nyquist rate, digital-to-analog converison.

The thesis begins by formulating the task and continues with defining the platform as an LTI system. Next, the process of time-domain convolution is introduced as a basis of digital signal processing. The thesis also contains a mathematical basis to design simple FIR filters using the weighted window technique. Also, a quick overview of some alternative FIR filter design methods is given.

The third chapter gives an overview of the DSP hardware used in the project. In addition, the basic properties of data communication protocols (I2S and SPDIF) are described to clarify the timing and coding of digital audio signals. The principles of inter-connecting the SPDIF receiver, DSP and DAC in the project are also described.

The fourth chapter focuses on the software developed for the platform starting from the basics of TI's CCS IDE. The software design for the platform is documented as a description of the relations and data exchange processes between various software functions in general. Source code is attached to the thesis digitally and cannot be found in any of the chapters.

In the fianl chapters, simple experimental mearurements have been performed to verify the functionality of the software designed for the platform. Also, some alternative solutions have been proposed to solve the task of FIR filtering in audio applications.

The thesis is in Estonian and contains 71 pages of text, 7 chapters, 24 figures, 1 tables.

Sisukord

Eessõna....3
Annotatsioon....4
Abstract....5
Kasutatavad mõisted ja lühendid....8
1. Sissejuhatus....9
1.2 Ülesandepüstitus....10
2. Digitaalsete filtrite teoreetilised alused....11
2.1 LTI-süsteemi omadused....11
2.2 Konvolutsioon....12
2.3 FIR-filter ja kaalukoefitsentide leidmine....15
2.3.1 Ideaalne madalpääsfilter....15
2.3.2 Fourier' teisendus....16
2.3.3 Sinc funktsioon ja aknameetod....18
2.3.4 Kõrgpääsfilter ja spektraalne inversioon....24
2.3.5 Alternatiivsed meetodid FIR-filtrikoefitsentide leidmiseks ja rakendamiseks....26
3. Ülevaade kasutatavast riistvarast....30
3.1 TMS320VC5505 eZdsp USB Stick....30
3.2 I2S-andmesideliides....32
3.3 SPDIF-vastuvõtja sidumine C5505 DSP-ga....35
4. Tarkvaraline lahendus....39
4.1 TI Code Composer Studio....39
4.2 “eZdsp Audio Filter Demo” näitekood ja selle modifitseerimisvajadus....41
4.3 Arendusplaadi tarkvaraline initsialiseerimine....43
4.4 Ülevaade tarkvaralisest andmevahetusprotsessist....46
4.5 Võimalikud tarkvaralised edasiarendused....49
5. Hinnang platvormi töövõimelisusele ja madalpääsfiltri spektri-analüüs....51
6. Alternatiivsed lahendused....54
6.1 DSP-l baseeruvad alternatiivsed lahendused....54
6.2 Sardarvutil baseeruvad alternatiivsed lahendused....55
7. Kokkuvõte....57
Kasutatud kirjandus....58
Lisa 1. Joonisel 6 toodud impulsskaja võendite (koefitsentide) väärtused....60
Lisa 2. Aknafunktsioonide ja kaalutud sinc funktsiooni kuvamiseks kasutatud programmkäsud “Ocatve” tarkvara baasil....61
Lisa 3. Faasi- ja sageduskarakteristiku kuvamiseks kasutatud programmkäsud “Octave” tarkvara baasil....62
Lisa 4. FIR-madalpääsfiltri koefitsendid vastavalt joonisele 8....63
Lisa 5. FIR-kõrgpääsfiltri koefitsendid vastavalt joonisele 11....64
Lisa 6. Kõrgpääsfiltri FIR-koefitsentide genereerimine ja kuvamine koos madalpääsfiltriga “Octave” tarkvara baasil....65
Lisa 7. DMA-kontrollerite poolt kasutatavad perifeeriaseadmed ja mälu jaotus [12]....66
Lisa 8. I2S plokkskeem [15]....67
Lisa 9. Tarkvaralise lahenduse graafiline esitus....68
Lisa 9.1 Stereosignaali vasaku kanali filtreerimise illustratsioon....68
Lisa 9.2 I2S-i, DMA ja DSP vahelised protsessid....69

Lisa 10. Platvormi töövõimelisuse hindamiseks koostatud programm-käsud (MatLab) 70

Tabelite loetelu

Tabel1. Konvolutsiooni näide: y[15] arvutamine.....14

Jooniste loetelu

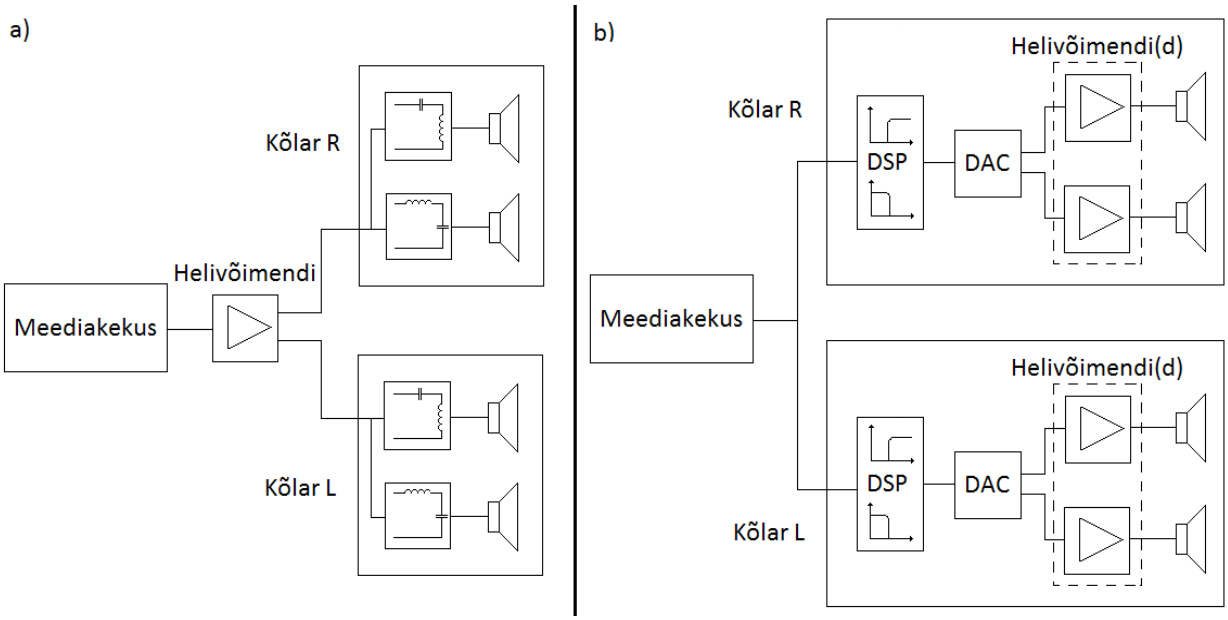
Joonis 1. Passiivne (a) ja aktiivne (b) helisüsteem.....9
Joonis 2. Diskreetse süsteemi mudel.....11
Joonis 3. Konvolutsiooni näide 1 [2].....13
Joonis 4. Konvolutsiooni näide 2 [2].....13
Joonis 5. Sinc funktsioon vahemikus -250⩽n⩽250, kui murdesagedus w_c=0.08(3)π rad.....18
Joonis 6. M/2 võrra hilistatud sinc funktsioon korrutatuna nelinurkse akna-funktsiooniga pikkusega 101 võendit, kui w_c=0.08(3)π rad.....19
Joonis 7. Joonisel 6 kujutatud impulsskajale vastav sagedus- ja faasikarakteristik.....21
Joonis 8. Aknafunktsioonid ja nendele vastavad sinc funktsioonid, kui w_c=1/12 π rad ja M=100.....22
Joonis 9. Faasi- ja sageduskarakteristikud, mis vastavad joonisel 8 kujutatud impulsskajadele.....23
Joonis 10. Spektraalse inversiooni illustratsioon.....24
Joonis 11. Spektraalne inversioon joonisel 8 kujutatud madalpääsfiltrist.....25
Joonis 12. Madal- ja kõrgpääsfiltrite faasi- ja sageduskarakteristikud vastavalt lisadele 4 ja 5.....26
Joonis 13. Ideaalne madalpääsfilter (katkendjoon) lõikesagedusega w_c ja sellele vastava realse madalpääsfiltri amplituudikarakteristik (pidevjoon) [4].....27
Joonis 14. VC5505 püsikoma DSP plokkskeem [9].....31
Joonis 15. TMS320VC5505 eZdsp USB Stick rev B [13].....32
Joonis 16. I2S-andmeside signaalide ajastus [15].....33
Joonis 17. AES3/SPDIF alam-kaadri struktuur [16].....35
Joonis 18. SPDIF ja signaali kodeerimine kasutades BMC meetodit.....36
Joonis 19. SPDIF-vatuvõtja ja DAC sidumine TMS320C5505 USB Stick arendusplaadiga [18].....36
Joonis 20. Programmkoodi genereerimine CCS keskkonnas [22].....40
Joonis 21. Platvormi töövõimelisuse hindamiseks loodud katsesüsteemi tööpõhimõte. 51
Joonis 22. Platvormi töövõimelisuse hindamiseks loodud katsesüsteemi väljundid aja- ja sagedusvallas.....52
Joonis 23. Platvormi töövõimelisuse hindamisel arvutatud ülekandefunktsioon.....53
Joonis 24. Pilootplatvorm.....53

Kasutatavad mõisted ja lühendid

ADC	Analog to Digital Converter, analoog-digitaal muundi
AES3	Audio Engineering Society standard digitaalse heli transportimiseks
BMC	Biphase Mark Code, kahend-faas kodeering
CCS	Code Composer Studio, tarkvara arenduskeskkond
COFF	Common Object File Format, failitüüp
CPU	Central Processing Unit, kesktöötlusseade/protsessor
DAC	Digital to Analog Converter, digitaal-analoog muundi
DARAM	Dual Access Random Access Memory, topelt ligipääsuga suvapöördusmälu
DFT	Discrete Fourier' Trasform, perioodilise diskreetse signaali Fourier' teisendus
DMA	Direct Memory Access, sõltumatu ligipääs mälule
DSP	Digital Signal Processor, digitaalne signaaliprotsessor
DTFT	Discrete Time Fourier Transform, aperioodilise diskreetse signaali Fourier' teisendus
EBSR	External Bus Selection Register, register välise andmesiini määramiseks
EBU	European Broadcasting Union, Euroopa avaliku meedia liit
EEPROM	Electrically Erasable Programmable Memory, elektriliselt kustutatav püsimälu
EMIF	External Memory Interface, välise mälu liides
FFT	Fast Fourier' Transform, kiire Fourier' teisendus
FIR	Finite Impulse Response, lõplik impulsskaja
FS	Fourier' Series, pideva perioodilise signaali Fourier' teisendus ehk Fourier' seeria
FT	Fourier' Transform, pideva aperioodilise signaali Fourier' teisendus
GPL	General Public License, GNU vabavara litsents
HARC	Harvard Architecture, arvutisüsteemi arhitektuuritüüp
I2S	Integrated Interchip Sound, standard digitaalse helisignaali transportimiseks
IDFT	Inverse Discrete Fourier' Transform, DFT pöördteisendus
IDTFT	Inverse Discrete Time Fourier' Transform, DTFT pöördteisendus
IIR	Infinite Impulse Response, lõpmatu impulsskaja
ISR	Interrupt Service Routine, katkestuse teenindusrutiin
jitter	sünkroniseerimissignaali perioodi hälve
LSB	Least Significant Bit, vähima kaaluga bitt (noorim bitt)
LTI	Linear Time Invariant, lineaarne ja aja (nihke) suhtes invariantne
MAC	Multiply-Accumulate, korrutamise ja akumuleerimise tehe
MIPS	Million Instructions Per Second, miljonit instruktsiooni sekundis
MSB	Most Significant Bit, suurima kaaluga bitt (vanim bitt)
PC	Personal Computer, personaalarvuti
PCM	Pulse-Code Modulation, impulsskoodmodulatsioon
PLL	Phase Locked Loop, faasilukk
PWM	Pulse-Width Modulation, impulsslaiusmodulatsioon
RAM	Random Access Memory, suvapöördusmälu
ROM	Read Only Memory, püsimälu
RTC	Real-Time Clock, reaalaja kell
sinc	Sinus Cardinalis, kardinaalsiinus
SPDIF	Sony/Philips Digital Interface Format, standard digitaalse helisignaali transportimiseks
TI	Texas Instruments Inc., mikroskeemide tootja
UDP	User Datagram Protocol, kasutajadatagrammi protokoll
USD	United States Dollar, Ameerika Ühendriikide valuuta

1. Sissejuhatus

Klassikalises stereo helisüsteemis, mida illustreerib joonis 9a, on keskne väline helivõimendi, mis võimendab helisignaali piisavale tasemele, et panna võnkuma valjuhääldi membraan. Teatavasti toodetakse erinevate helisagedusvahemike jaoks spetsiaalsed valjuhääldid. Toodud illustratiivses näites on kujutatud nn "kahe sagedusribalised" kõlarid, mis tähendab, et üles võimendatud helisignaal on mõistlik jagada kaheks sagedusribaks: madalad- ka kõrged helisagedused, mida eraldab murdesagedus. Klassikaliselt toimub see sageduslik jagamine passiivsete filtrite abil – madal- ja kõrgpääsfiltrid, mille realiseerimiseks piisab üldiselt kondensaatoritest, induktiivpoolidest ja takistitest.



Joonis 1. Passiivne (a) ja aktiivne (b) helisüsteem.

Joonisel 9b on kujutatud alternatiivset helisüsteemi, kus helisignaal jõuab kõlariteni digitaalsel kujul. Signaali töötlemiseks asub kõlaris DSP (digitaalne signaaliprotsessor, ingl. k *Digital Signal Processor*), mille eesmärk on eraldada omavahel kõrgemad- ja madalamad helisagedused. Nii DSP sisend kui ka väljund on digitaalsed, mistõttu on vajalik ka DAC (digitaal-analoog muundi, ingl. k *Digital-to-Analog Converter*). Analoogsignaal juhitakse helivõimendini, mille väljundis asub vahetult valjuhääldi. Joonisel 9b kujutatud kõlarit võib nimetada digitaalse filtriga aktiivseks kõlariks, sest helivõimendi asub valjuhäälditega samas kastis ja sagedusribade filtreerimine toimub digitaalselt. Joonisel 9a kujutatud helisüsteemi nimetatakse passiivseks, seda nii analoogfiltrite kui ka välise helivõimendi osas.

Käesolevas töös on propageeritud digitaalse filtriga aktiivset helisüsteemi, passiivset helisüsteemi eraldi rohkem ei käsitleta. Keskseks teemaks on helisignaali digitaalne filtreerimine, sealjuures helivõimendeid ja valjuhääldeid ei käsitleta. Töö eesmärgiks on luua platvorm digitaalsete filtrite katsetamiseks ja arendamiseks, kusjuures eesmärk ei ole leida "õiget" või "parimat" digitaalse filtri algoritmi.

1.2 Ülesandepüstitus

Signaalitöötluse ülesannete lahendamiseks kasutada TI (Texas Instruments) C5000 seeria DSP-l põhinevat arendusplaati "TMS320VC5505 eZdsp USB Stick" (Texas Instruments). Nimetatud arendusplaadi kasuks räägivad suhteliselt madal hind (umbes 50 EUR), ülimadal energiatarve, hea tootja poolne tugi FIR-filtrite realiseerimiseks ja CCS (ingl. k *Code Composer Studio*) tarkvara arenduskeskkond. Lisaks ka riistvaralised omadused: kaks MAC-korrutit (ingl. k *multiplier–accumulator*), 4 I2S (ingl. k *Integrated Interchip Sound*) kanalit ja 4x4 DMA kanalit (ingl. k *Direct Memory Access*).

Helisignaal peab meediakeskusest helisüsteemini jõudma digitaalsel kujul, et vähendada pikkade maanduskaablitega seotud häirete mõju helisignaalile ja vältida liigset digitaal-analoog ja analoog-digitaal muundamist (eelduse kohaselt on helisalvestis algselt digitaalsel kujul). Helisüsteem peab toetama laiatarbelist SPDIF (ingl. k *Sony/Philips Digital Interconnect Format*, samuti tuntud kui *Sony/Philips Digital Interface* ja *IEC60958*) protokolli ja omama nii optilise- (Toslink) kui ka koaksiaal-liidesega meedia sisendit.

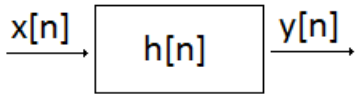
Filtrisüsteem peab olema suuteline vastu võtma digitaalset helisignaali, mis on salvestatud võendamissagedusega 44.1kHz, 48kHz, 88.2kHz, 96kHz ja 192kHz. Sealjuures ei ole sisendsignaali võendamissagedus kasutaja poolt määratud, vaid süsteem peab olema suuteline lülituma erinevate võendamissageduste režiimist automaatselt ümber. Riistvara peab võimaldama eelmainitud võendamissagedustel vähemalt 100-ndat järku FIR (ingl. k *Finite Impulse Response*) filtri realiseerimist vähemalt kahele väljundkanalile.

Süsteemi latentsusele otseselt tähelepanu ei pöörata, st ei ole määratud maksimaalset lubatud hilistumist (see sõltub puhvrite pikkusest ja sisendsignaali võendamissagedusest). Küll aga peab helisalvestise esitamise jooksul olema väljundsignaali hilistumine (filtri rühmahilistus) konstantne. Iga sageduskomponendi faas peab olema konstantse viitega (vähendab faasi-moonutusi). Nendest tingimustest tulenevalt on mõistlik kasutada lineaarse faasiga FIR-filtrit.

2. Digitaalsete filtrite teoreetilised alused

2.1 LTI-süsteemi omadused

Vastavalt ülesandepüstitusele on filtrisüsteem diskreetne, mistõttu võib antud süsteemi kirjeldada joonisel 2 kujutatud mudeliga, kus



Joonis 2. Diskreetse süsteemi mudel.

x[n] – sisendsignaal
y[n] – väljundsignaal
h[n] – süsteemi impulsskaja
n – sümbol tähistamaks asjaolu, et tegemist on muutujaid sisaldava massiiviga

Järgnevalt on defineeritud süsteemi lineaarsusomadused:
1. Aditiivsusomadus, mis on väljendatav kujul:

$$(x_1[n] \rightarrow y_1[n]) \wedge (x_2[n] \rightarrow y_2[n]) \Rightarrow (x_1[n] + x_2[n]) \rightarrow (y_1[n] + y_2[n]) \qquad (1)$$

Kui sisendsuurus $x_1[n]$ tekitab väljundi $y_1[n]$ ning sisendsuurus $x_2[n]$ tekitab väljundi $y_2[n]$, siis sisendite summa $x_1[n] + x_2[n]$ tekitab väljundi $y_1[n] + y_2[n]$. Ehk sisendmuutujate summa sisendis tekitab vastavate väljundmuutujate summa väljundis.
2. Homogeensusomadus, mis on mistahes reaalarvulise konstandi *k* korral väljendatav kujul:

$$x[n] \rightarrow y[n] \Rightarrow k \times x[n] \rightarrow k \times y[n] \qquad (2)$$

Sisendsuuruse korrutamisel reaalarvuga muutub väljundmuutuja sama arvu kordselt.
Kui rahuldatud on nii tingimus (1) kui ka tingimus (2), võib süsteemi nimetada lineaarseks. Nendest kahest tingimusest tuleneb ka signaalitöötluse üks olulisemaid printsiipe: superpositsiooni printsiip, mida võib tõlgendada järgnevalt: iga sisendsignaali mingi komponendi põhjustatud reaktsioon süsteemi väljundis ei sõltu teiste sisendkomponentide põhjustatud väljundreaktsioonidest ning kõik reaktsioonid väljundis liituvad. [1]

See omadus võimaldab dekomposeerida keerulise kujuga sisendsignaalid lihtsamateks sisendsignaali komponentideks, mis põhjustavad lihtsamaid väljundsignaali komponente. Tekkinud väljundsignaali komponendid sünteesitakse (liidetakse) lõplikuks väljundsignaaliks. Sellisel moel sünteesitud väljundsignaal on identne signaaliga, mille põhjustab algne keerulise kujuga sisendsignaal, kui see läbib lineaarse süsteemi. Seega, selle asemel, et mõista, kuidas muudab süsteem keerulisi sisendsignaale, piisab teadmisest, kuidas süsteem muudab lihtsaid sisendsignaali komponente.
3. Aja (diskreetsete süsteemide korral nihke) suhtes invariantsus (ingl k. *time (shift) invariance*) on iga reaalarvulise konstandi *s* korral väljendatav kujul:

$$x[n] \rightarrow y[n] \Rightarrow x[n+s] \rightarrow y[n+s] \qquad (3)$$

Teisisõnu, nihe sisendsignaalis põhjustab samaväärse nihke väljundsignaalis iga sisendi $x[n]$ korral ja iga reaalarvulise konstandi s korral. Nihke suhtes invariantsus on

antud töö kontekstis oluline omadus, sest see tähendab, et süsteemi omadused ajas ei muutu. [2]

Käesolevas töös on tehtud *a priori* eeldus, et disainitaval süsteemil on kõik kolm eelkirjeldatud omadust. Seega süsteem on lineaarne ja nihke suhtes invariantne. Edaspidi on kasutatud sellise süsteemi kirjeldamiseks laialt levinud mõistet LTI (ingl k. *Linear Time Invariant*), kuigi antud töö kontekstis on peetud LTI-süsteemi all silmas lineaarset nihke suhtes invatiantset süsteemi. LTI-süsteemidel on signaalitöötluse kontekstis väga oluline omadus: süsteemi impulsskaja (h[n] joonisel 2) defineerib täielikult süsteemi käitumise.

2.2 Konvolutsioon

Konvolutsioon on signaalitöötluse üks olulisi tehteid, sest see seob omavahel süsteemi sisendi, väljundi ja impulsskaja. Konvolutsioon sisendi poolt vaadelduna on seletatav sisendsignaali dekompositsiooniga, mis tähendab, et vaadeldakse igat võendit eraldi kui impulssi. Sellisel juhul võib võendit (impulssi) signaalis a[n] iseloomustada kujul:

$$a[n]=k\times\delta[n-s] \quad \text{,kus} \qquad (4)$$

k – impulsi (võendi) amplituud
$\delta[\,]$ - ühikimpulss
s - nihe(võendi järjekorra number)

Teisisõnu, sisendsignaal dekomposeeritakse selliselt, et tekib N sisendsignaali komponenti, mis koosnevad skaleeritud ja nihutatud ühikimpulsist. Kasutades LTI-süsteemi omadusi, võib väita, et:

$$\delta[n]\rightarrow h[n]\Rightarrow k\times\delta[n-s]\rightarrow h[k\times\delta[n-s]] \qquad (5)$$

iga selline skaleeritud ja nihutatud ühikimpulss tekitab väljundisse vastava skaleeritud ja nihutatud impulsskaja. Kui sisendsignaali dekompositsiooni käigus tekkis N sisendsignaali komponenti, siis tekib ka N vastavat väljundsignaali komponenti. Lõpliku väljundi saamiseks liidetakse kõik N väljundsignaali komponenti (liidetakse nihutatud ja skaleeritud impulsskajad).[2]

Joonisel 2 kujutatud LTI-süsteemi mudeli kohta võib esitada järgmise võrduse [3]:

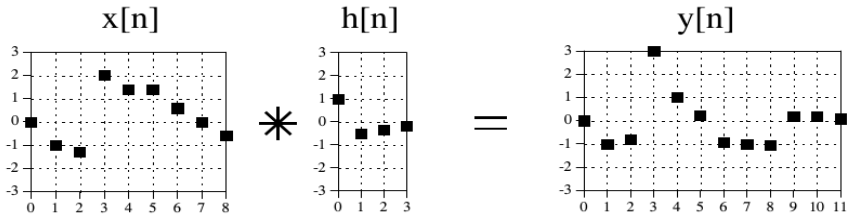
$$x[n]*h[n]=y[n] \quad \text{, kus} \qquad (6)$$

* - tähistab konvolutsiooni tehet

Konvolutsiooni võib seletada kasutades eelkirjeldatud dekompositsiooni põhimõtet: kui sisendsignaal on pikkusega N võendit, siis tekib väljundisse N nihutatud ja skaleeritud impulsskaja komponenti, mis sünteesitakse (liidetakse) üheks väljundsignaaliks. Võttes impulsskaja pikkuseks väärtuse M, tekib seega väljund, mille pikkus L on arvutatav valemist 7:

$$L=N+M-1 \qquad (7)$$

Kirjeldatud konvolutsiooni põhimõtet demonstreerib joonis 3, kus sisendsignaali pikkus N=9 ja impulsskaja pikkus M=4, tekitades väljundi pikkusega L=12:



Joonis 3. Konvolutsiooni näide 1. [2]

Konvolutsiooni on mugavam seletada vaadelduna sisendi poolelt (kuidas iga üksik võend sisendis mõjutab väljundit kui tervikut), kuid praktilisem kasutada vaadelduna väljundi poolelt (kuidas iga üksik väljundväärtus on tekkinud teatava hulga sisendite mõjul). Matemaatiliselt on seega konvolutsiooni summat mõistlik esitada valemiga 8 [4]:

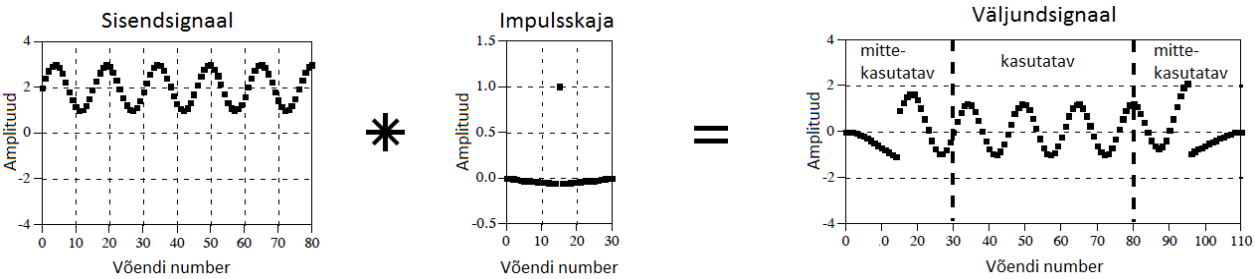
$$y[n]=\sum_{k=0}^{M-1} h[k]x[n-k] \quad \text{,kus} \qquad (8)$$

n - tähistab parasjagu käesolevat ajahetke (diskreeti)

k - tähistab impulsskaja võendi järjekorranumbrit

Lahtiseletatult, väljundi $y[n]$ leidmiseks korrutatakse käesoleva ajahetke n suhtes k diskreeti hilistatud (nihutatud) sisendsignaali väärtus $x[n-k]$ vastava impulsskaja võendi väärtusega $h[k]$. Tekib vahekorrutis iga impulsskaja võendi kohta vahemikus (0...M-1). Vahekorrutiste summa moodustab antud ajahetke väljundväärtuse $y[n]$, kusjuures n järjekorranumbrid jooksevad vahemikus (0...N+M-2), kus N on sisendvõendite koguarv.

Konvolutsiooni võib nimetada matemaatiliseks meetodiks kahe signaali "korrutamisel". Antud töö kontekstis, kus $x[n]$ on sisendvõendite jada ja $y[n]$ on filtreeritud väljundvõendite jada, tuleb leida vaid sobilik impulsskaja ja helifilter ongi konvolutsiooni abil realiseeritav. Siiski, sellisel moel filtrit realiseerides tuleb tähelepanu pöörata paarile nüansile, mis on kirjeldatud joonise 4 baasil.



Joonis 4. Konvolutsiooni näide 2 .[2]

Olgu sisendvõendite koguarv N=81 ja impulsskaja pikkus M=31 võendit. Vastavalt valemile 7 on väljundignaali võendite koguarv L=111. Kui rakendada konvolutsiooni summa valemit 8 väljundi $y[15]$ arvutamiseks, siis tuleb korrutada ja summeerida komponendid tabelis 1:

Tabel 1. Konvolutsiooni näide: y[15] arvutamine.

Impulsskaja võend		Sisendvõend
h[0]	x	x[15]
h[1]	x	x[14]
h[2]	x	x[13]
…	x	…
…	x	…
…	x	…
h[13]	x	x[2]
h[14]	x	x[1]
h[15]	x	x[0]
h[16]	x	x[-1]
h[17]	x	x[-2]
h[18]	x	x[-3]
…	x	…
…	x	…
…	x	…
h[28]	x	x[-13]
h[29]	x	x[-14]
h[30]	x	x[-15]

$$\sum_{k=0}^{30} h[k]\times x[15-k] = \quad ???$$

Tabeli 1 põhjal ilmneb, et esimese 30 väljundvõendi arvutamiseks on vajalik teada ka sisendsignaali väärtusi, mille järjekorranumber on negatiivne. Mõistagi ei ole esimesele võendile eelnenud võendite kohta võimalik infot leida, mistõttu loetakse kõik negatiivse järjekorranumbriga sisendvõendid nullisteks:

$$x[n-k]<0 \Rightarrow x[n-k]=0 \qquad (9)$$

Sellest tulenevalt ei ole esimesed M-1 väljundvõendit korrektsed ja võivad osutuda mittekasutatavateks. Analoogne juhtum ilmneb ka viimase 30 väljundvõendiga, kus valemi 8 kohaselt on väljundi arvutsamiseks tarvis "tuleviku" sisendvõendeid. Antud näite puhul loetakse jällegi puuduvad võendid nullisteks,

$$x[n-k]>80 \Rightarrow x[n-k]=0 \qquad (10)$$

mistõttu võivad ka viimased M-1 võendit olla kasutuskõlbmatud.

Eelkirjeldatud süsteemi, kus negatiivsete järjekorranumbritega võendid loetakse nullisteks, nimetatakse kausaalseteks süsteemideks. Kausaalse süsteemi puhul on määratud, et sisendvõend järjekorranumbriga N ei saa mõjutada väljundvõendit, mille järjekorranumber on väiksem kui N.

Kausaalsest süsteemist tulenenud väljundsignaali algusosa ja lõpuosa mittekasutatavus ei pruugi olla süsteemis kuigi määrav, sest helisüsteemi korral on sisendsignaal väga pikk ja helisalvestise alguses ja lõpus ei sisaldu tihti palju infot (signaal on nulline ehk on vaikus). Pealegi, kui helisignaal on salvestatud võendamissagedusega 48kHz ja impulsskaja pikkus on 31 võendit, siis kestab mittekasutatav signaal 0,625 ms (valemist 11), mis ei pruugi olla helisüsteemis määrava pikkusega aeg. Lisaks muule on võimalik tarkvaraliselt kõrvaldada mittesobilik alguse- ja lõpuosa.

$$t=\frac{M-1}{fs}=\frac{30}{48000\text{Hz}}=0{,}625\,ms \quad \text{, kus} \qquad (11)$$

fs - helisignaali võendamissagedus

Küll aga on oluline mõista kausaalse süsteemi omapärasid programmkoodi kirjutamisel, Kui rakendada konvolutsiooni summa valemit (8) ilma tingimust 9 ja 10 täitmata, siis

programmkood ilmselt hangub, kui pöördutakse olematu massiivi muutuja poole, mis on väljaspool massiivi piire.

Tähelepanuväärne asjaolu sellise filtri realiseerimisel on filtri arvutuslik kulu. Kui sisendsignaali pikkus on oluliselt suurem impulsskaja pikkusest (st $N \gg M$), võib valemist 8 välja lugeda, et vastava konvolutsiooni teostamiseks tuleb sooritada $N \times M$ korrutamis- ja liitmistehet ehk MAC-i. Seega, näiteks 1 sekundi pikkuse sisendi korral, mille võendamissagedus on $f_s = 48\text{kHz}$ ja süsteemi impulsskaja pikkus on 50 võendit, tuleb teostada 24 miljonit MAC-tehet vastavalt valemile 12

$$NR_{MAC} = t \times f_s \times M = 1 \times 48000 \times 50 = 2400000\, MAC = 2{,}4 \times 10^6\, MAC \quad (12)$$

Kõrgemat järku filtrite korral võib arvutuslik kulu muutuda problemaatiliseks, kui ei kasutata vastava ülesande jaoks loodud spetsiaalset riistvara ja/või efektiivsemat algoritmi.

2.3 FIR-filter ja kaalukoefitsentide leidmine

2.3.1 Ideaalne madalpääsfilter

Eelmises peatükis kirjeldatud konvolutsiooni meetod on üks lihtsamalt mõistetavaid ja otsesemaid võimalusi FIR-filtrite realiseerimiseks. Nagu eelpool kirjeldatud, tuleb leida vaid impulsskaja, mis vastab soovitud süsteemi omadustele. Helisüsteemi ideaalse madalpääsfiltri sageduskarakteristik on matemaatiliselt kirjeldatav seosega 13 [4]:

$$H_d(\omega) = \begin{cases} 1, & kui \quad \omega \in [0,\ \omega_c) \\ 0, & kui \quad \omega \in (\omega_c, \pi] \end{cases}, \text{ kus} \qquad (13)$$

$H_d(\omega)$ - soovitud filtri sageduskarakteristik
ω - sagedus diskreedi kohta
ω_c - lõikesagedus

Seosest on näha, et ideaalne madalpääsfilter laseb muutumata kujul läbi kõik sagedused, mis on väiksemad lõikesagedusest ω_c , ning sellest kõrgemad sagedused eemaldatakse täielikult. Teiste sõnadega, ideaalse madalpääsfiltri läbilaskeala (ingl. k *pass-band*) on vahemikus $\omega \in [0, \omega_c)$ ja tõkkeala (ingl. k *stop-band*) vahemikus $\omega \in (\omega_c, \pi]$.[4]

Kuna töös on käsitletud vaid diskreetseid süsteeme, tuleb märkida, et ω ei tähista antud kontekstis klassikalist ringsagedust, mille ühikuks on $[rad/s]$. Diskreetsete funktsioonide puhul on sageduse ühikuks radiaani diskreedi kohta ehk lihtsalt $[rad]$. See tuleneb tõsiasjast, et diskreetsete funktsioonide ja signaalide puhul ei ole aeg määratletud, vaid horisontaalteljel on diskreedi järjekorra number. Ajavahemik kahe diskreedi vahel sõltub diskreetimissagedusest $[f_s]$. Lisaks on oluline märkida, et diskreetsete signaalide sagedusspekter on perioodiline perioodiga 2π ja reaalsete väärtustega funktsioonide korral on sagedusspekter ka sümmeetriline π suhtes .[3] Seega piisab, kui amplituudikarakteristik esitada sagedusvahemikus $\omega \in [0, \pi]$. Ringsagedus ω ja sagedus f (esitatuna hertsides (Hz)) on seotav valemiga 14 [4]:

$$\omega = \frac{2\pi f}{f_s} \rightarrow f = \frac{\omega f_s}{2\pi} \qquad (14)$$

Seos 13 (ideaalne madalpääsfilter) kujutab endast ristkülikulist signaali sagedusvallas. Helisignaali sagedusfiltri realiseerimine konvolutsiooni meetodil eeldab, et impulsskaja on ajavalla signaal (sest sisendsignaal koosneb ajavallas esitatud võenditest). Järelikult tuleb leida selline impulsskaja (ajavallas), mille sageduskarakteristik (sagedusvallas) on ristküliku-kujuline. Selleks, et leida seosed aja- ja sagedusvalla vahel, tuleb uurida Fourier´ teisenduse olemust, eriti aga diskreetse Fourier' teisenduse olemust arvestades käesolevas töös käsitletavat diskreetset signaalitöötlust.

2.3.2 Fourier' teisendus

Fourier' teisenduse abil on mis tahes signaali võimalik üheselt esitada sinusoidide summana. Signaali spekter (signaali Fourier' teisendus) näitab, kui suure amplituudiga ja millises faasis erineva sagedusega sinusoide antud signaali esitus sisaldab. Seega on signaali spekter sageduse funktsioon.[3]

Nii nagu signaalitüübidki, saab ka Fourier' teisenduse jagada nelja kategooriasse. Signaal võib olla pidev või diskreetne ning mõlemad võivad olla perioodilised või aperioodilised signaalid.

1. Aperioodiline pidev signaal on määratud vahemikus $(-\infty, +\infty)$, kusjuures signaal ei korda ennast perioodiliselt. Sellist tüüpi signaali teisendust nimetatakse Fourier' teisenduseks ehk FT-ks (ingl. k *Fourier' Transform*).
2. Perioodiline pidev signaal on määratud vahemikus $(-\infty, +\infty)$, kusjuures signaal kordab ennast perioodiliselt (näiteks sinusoidid). Sellist tüüpi signaali teisendust nimetatakse Fourier' seeriaks ehk FS-ks (ingl. k *Fourier' Series*).
3. Aperioodiline diskreetne signaal on määratud diskreetsetel ajahetkedel vahemikus $(-\infty, +\infty)$, kusjuures signaal ei korda ennast perioodiliselt. Sellist tüüpi signaali teisendust nimetatakse DTFT-ks (ingl. k *Discrete Time Fourier' Transform*).
4. Perioodiline diskreetne signaal on määratud diskreetsetel ajahetkedel vahemikus $(-\infty, +\infty)$, kusjuures signaal kordab ennast perioodiliselt. Sellist tüüpi signaali teisendust nimetatakse DFT-ks (ingl. k *Discrete Fourier' Transform*). [2]

Digitaalse helisignaali puhul on ilmselt tegemist lõpliku signaaliga. Samas, Fourier' teisendus on otseselt rakendatav vaid lõpmatute signaalide korral. See on seletatav asjaoluga, et ka teisenduse tulemusena leitud sinusoidid on lõpmatud signaalid, millest ei saa sünteesida midagi lõplikku. Fourier' teisenduse metoodika kasutamiseks lõpliku signaali korral on kaks võimalust [2]:

1. Vaadelda lõplikku signaali selliselt, et enne esimest ja peale viimast lõpliku signaali võendit on signaalis lõpmatu arv nulliseid võendeid. Selliselt vaadeldud signaal paistab olevat lõpmatu aperioodiline diskreetne signaal, mistõttu on sel puhul rakendatav DTFT.
2. Vaadelda lõplikku signaali selliselt, et sellele eelnes (ja järgneb) lõpmatu arv identseid lõplikke signaale. Piltlikult öeldes, panna signaali algus ja lõpp kokku. Tulemuseks on lõpmatu perioodiline diskreetne signaal, mille periood on võrdne algse lõpliku signaali pikkusega. Sellise signaali puhul on rakendatav DFT.

Arvutisüsteemides on kasutatav vaid DFT tüüpi teisendus, mis tuleneb näiteks asjaolust,

et arvuti suudab salvestada vaid lõplikke digitaalseid arve.

Ideaalse filtri sageduskarakteristiku matemaatilise kirjelduse saamiseks on otstarbekas vaadelda sisendsignaali kompleksse eksponentfunktsioonina: [5]

$$x[n]=\mathrm{e}^{j\omega n} \quad \text{, kus} \quad -\infty<n<+\infty$$

Konvolutsiooni summa (valem 8) saab sel juhul kuju:

$$y[n]=\sum_{k=-\infty}^{+\infty} h[k]\mathrm{e}^{j\omega(n-k)}=\mathrm{e}^{j\omega n}\left(\sum_{k=-\infty}^{+\infty} h[k]\mathrm{e}^{-j\omega k}\right) \qquad (15)$$

Valemist 15 saab defineerida süsteemi kompleksarvulise sageduskarakteristiku $H(\mathrm{e}^{j\omega})$ järgmiselt:

$$H_d(\mathrm{e}^{j\omega})=\sum_{k=-\infty}^{+\infty} h[k]\mathrm{e}^{-j\omega k} \qquad (16)$$

Nagu ideaalse madalpääsfiltri sageduskarakteristiku defineerimisel mainitud, on sageduskarakteristik perioodiline perioodiga 2π, mistõttu piisab kui sageduskarakteristikut kujutada vahemikus $-\pi<\omega<\pi$. Arvestades valemit 16, võib valemi 15 esitada järgmiselt:

$$y[n]=H_d(\mathrm{e}^{j\omega})\mathrm{e}^{j\omega n} \qquad (17)$$

Nagu eelpool mainitud, põhineb seos ajavalla impulsskaja ja selle sagedusvalla esituse vahel "tagurpidi" Fourier' teisendusel ehk IDTFT-l (ingl. k. *Inverse Discrete Time Fourier' Transform*), mis on defineeritud järgmiselt [5]:

$$h[n]=\frac{1}{2\pi}\int_{-\pi}^{\pi} H(\mathrm{e}^{j\omega})\mathrm{e}^{j\omega n}d\omega \qquad (18)$$

Arvestades sageduskarakteristiku perioodilisust võib ideaalse madalpääsfiltri ajavalla impulsskaja $h_d[n]$ leida selle sageduskarakteristikust (seos 13) kasutades valemit 18 järgmiselt [5]:

$$h_d[n]=\frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} H_d(\mathrm{e}^{j\omega})\mathrm{e}^{j\omega n}d\omega=\frac{1}{2\pi jn}\left(\mathrm{e}^{j\omega n}\right)\Big|_{-\omega_c}^{\omega_c}=\frac{1}{(2\pi jn)}\left(\mathrm{e}^{(j\omega_c n)}-\mathrm{e}^{(-j\omega_c n)}\right)=\frac{\sin\omega_c n}{\pi n} \quad ,(19)$$

kus $n\neq 0$

Valemist 19 ilmneb, et ideaalse madalpääsfiltri impulsskaja $h_d[n]$ (ajavallas) on lõpmatu pikkusega, mis on seotud asjaoluga, et ideaalse madalpääsfiltri sageduskarakteristikus on katkevuspunkt punktis ω_c . Lisaks ilmneb ka tõsiasi, et ideaalne madalpääsfilter ei ole kausaalne, kuna väärtustel $n<0$ ei ole impulsskaja väärtus alati nulline. Sisuliselt tõestavad need asjaolud järjekordselt, et ideaalse madalpääsfiltri realiseerimine ei ole arvutisüsteemides võimalik.

Valemi 16 eeskujul saab tuletada ka seose sageduskarakteristiku magnituudi ja faasi leidmiseks [6]:

$$H(\omega)=\sum_{k=-\infty}^{+\infty} h[k]\mathrm{e}^{-j\omega k}=\sum_{k=-\infty}^{+\infty} h[k]\cos\omega k-j\sum_{k=-\infty}^{+\infty} h[k]\sin\omega k=H_R(\omega)+jH_I(\omega) \qquad (20)$$

Sageduskarakteristiku reaalne osa tähisega $H_R(\omega)$ ja imaginaarne osa tähisega $H_I(\omega)$ on seega defineeritud järgmiselt [6]:

$$H_R(\omega)=\sum_{k=-\infty}^{+\infty} h[k]\cos\omega k \qquad (21)$$

$$H_I(\omega)=-\sum_{k=-\infty}^{+\infty} h[k]\sin\omega k \qquad (22)$$

Sageduskarakteristiku $H_d(\omega)$ magnituud $|H(\omega)|$ ja faas $\theta(\omega)$ on esitatavad järgmiselt [6]:

$$|H(\omega)|=\sqrt{H_R^2(\omega)+H_I^2(\omega)} \qquad (23)$$

$$\theta(\omega)=\arctan\frac{H_I(\omega)}{H_R(\omega)} \qquad (24)$$

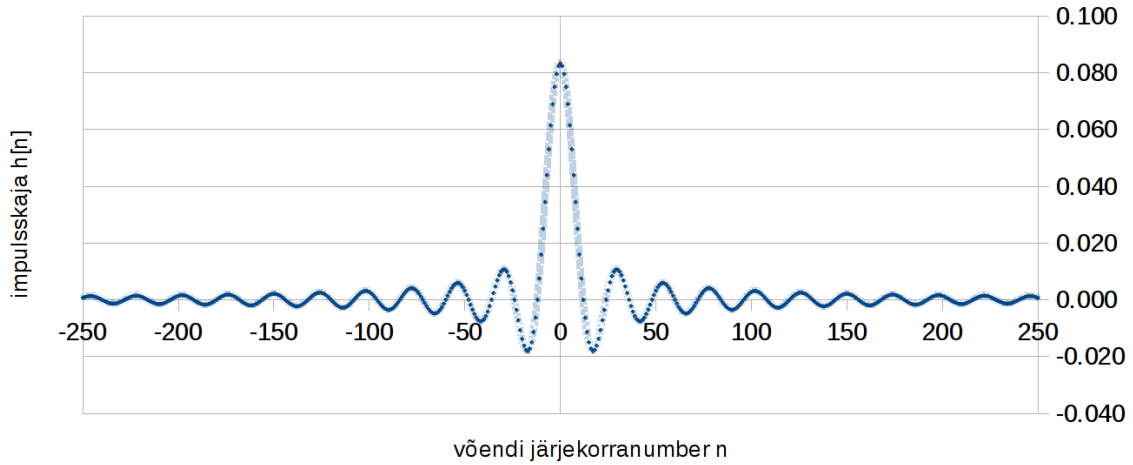
Arvutisüsteemide korral kirjeldatakse üldiselt sageduskarakteristik N diskreediga (kuigi sageduskarakteristik võib olla pidev), mis on jaotatud võrdsete vahedega vahemikus $0<\omega<2\pi$, st võendite vaheline kaugus on $\frac{2\pi}{N}$. Sel juhul võib teatud eeldustel kasutada järgmisi DTFT ja DFT vahelisi seoseid [6]:

$$H(\omega)=\sum_{n=-\infty}^{+\infty}h[n]e^{-j\omega n} \quad \Leftrightarrow \quad H[k]=\sum_{n=0}^{N-1}h[n]e^{-j\frac{2\pi kn}{N}} \text{ , kus } k=0,1,\dots,N-1 \qquad (25)$$

$$h[n]=\frac{1}{2\pi}\int_{-\pi}^{\pi}H(\omega)e^{j\omega n}d\omega \Leftrightarrow \quad h[n]=\frac{1}{N}\sum_{k=0}^{N-1}H[k]e^{j\frac{2\pi kn}{N}} \text{ ,kus } n=0,1,\dots,N-1 \qquad (26)$$

2.3.3 Sinc funktsioon ja aknameetod

Valemi 19 tulemuseks on nn "sinc" funktsioon, mis on antud töö kontektis olulise tähtsusega signaal. Joonisel 5 on illustreeritud valemi 19 abil koostatud sinc funktsiooni, kus murdesagedus $\omega_c=0.08(3)\pi\,rad$ (vastab murdesagedusele $f_c=2\text{kHz}$, kui võendamissagedus $f_s=48\text{kHz}$). Impulsskaja väärtus $h_d[n]$ on joonisel 5 välja arvutatud vahemikus $-250\le n\le 250$, kuigi sinc funktsioon on tegelikult lõpmatu. Tegemist on siinusfunktsiooniga, mille amplituud hääbub pöördvõrdeliselt $|n|$ kasvamisega. Punktis $n=0$ on impulsskaja väärtus $h[0]=\frac{\omega_c}{\pi}$.



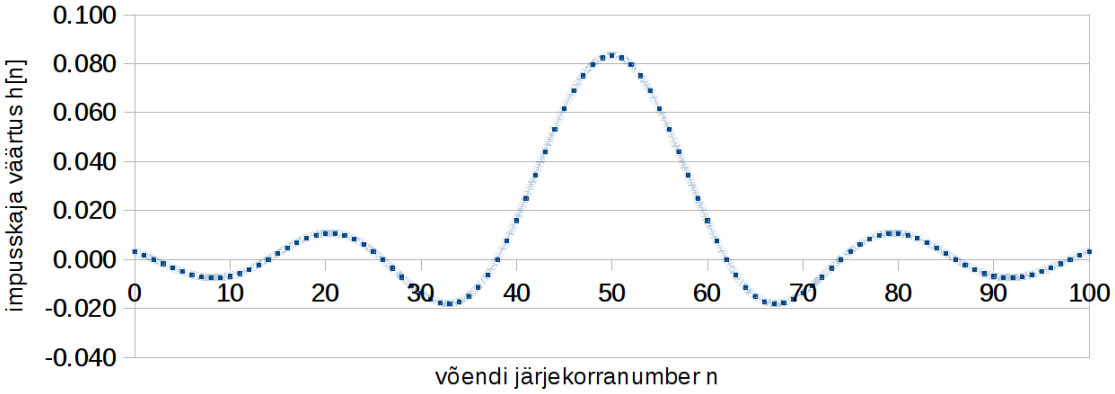
Joonis 5. Sinc funktsioon vahemikus $-250\le n\le 250$ *, kui murdesagedus* $\omega_c=0.08(3)\pi\,rad$.

Selleks, et lõpmatu pikkusega sinc funktsioon (impulsskaja) oleks kasutatav arvutiriistvaral, tuleb signaalist valida teatud lõplik arv võendeid. Mida suurem arv võendeid valida, seda lähedasem on filtri sageduskarakteristik ideaalsele madalpääsfiltrile (seda selektiivsem on filter). Võimaliku valitavate võendite arvu määrab tihti protsessori arvutuslik võimsus ja nõuded süsteemi lubatud hilistumisele, sest impulsskaja pikkus on tihedalt seotud konvolutsiooni realiseerimiseks vajalike MAC-tehete arvuga, mille protsessor peab olema võimeline teatud aja jooksul

sooritama.

Kuna joonisel 5 kujutatud sinc funktsioon on perioodiline võendi $n=0$ suhtes, on kasulik valida impulsskaja pikkuseks M+1 võendit, kusjuures M on valitud paarisarvude seast. Ülejäänud võendid võib lugeda nullisteks.

Lisaks tuleb kogu impulsskaja signaali (pikkusega M+1) nihutada M/2 diskreedi võrra edasi, nii et impulsskaja võendite järjekorranumbrid hakkavad olema vahemikus 0 kuni M. Selline signaal on kujutatud joonisel 6, kus impulsskaja pikkuseks on valitud 101 võendit (st. valitud on M=100).



Joonis 6. $\frac{M}{2}$ *võrra hilistatud sinc funktsioon korrutatuna nelinurkse akna-funktsiooniga pikkusega 101 võendit, kui* $\omega_c=0.08(3)\pi\, rad$.

Joonisel 6 kujutatud signaali võib nimetada valemi 19 abil arvutatud sinc funktsiooniks, mis on nihutatud $\frac{M}{2}$ diskreedi võrra ja korrutatud "nelinurkse aknafunktsiooniga" (ingl. k *rectangular window*) pikkusega M+1. Vastav ajavalla aknafunktsioon $\omega[n]$ on matemaatiliselt esitatav kujul [6]:lk624

$$\omega_{nelinurk}(n)=\begin{cases} 1, & kui \quad n=0,1,\ldots,M \\ 0, & kui \quad n\neq 0,1,\ldots,M \end{cases} \qquad (27)$$

Joonisel 6 kujutatud signaali saab seega esitada järgmiselt:

$$h(n)=h_d(n-\frac{M}{2})\omega(n)=\begin{cases} h_d[n-\frac{M}{2}], & kui \quad n=0,1,\ldots,M \\ 0, & kui \quad n\neq 0,1,\ldots,M \end{cases} \qquad (28)$$

Selliselt loodud lõpliku impulsskaja sageduskarakteristik $H(\omega)$ ei ole enam võrdne ideaalse madalpääsfiltri sageduskarakteristikuga $H_d(\omega)$. Vastavad impulsskaja võendite väärtused (koefitsendid) on toodud lisas 1. Pidades silmas, et korrutamine ajavallas vastab konvolutsioonile sagedusvallas, saab antud olukorras huvi pakkuva sageduskarakteristiku leida valemist 29 [6]:

$$H(\omega)=\frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\upsilon)W(\omega-\upsilon)d\upsilon \quad \text{, kus} \qquad (29)$$

$W(\omega)$ - nelinurkse aknafunktsiooni sageduskarakteristik (Fourier' teisendus signaalist $\omega[n]$).

Praktikas on otstarbekas rakendada Fourier' teisendust tarkvara abil arvutiriistvaral. FIR-filtrite koefitsentide leidmiseks on palju veebirakendusi ja tarkvarapakette, näiteks "FDATool" (ingl. k *Filter Design and Analysis Tool*) "MatLab" keskkonnas, või vabavaralise GNU/Linux "Octave" tarkvara "*signal package*", mida

võib kasutada GPL (ingl. k *General Public License*) litsentsiga.
Antud töös on propageeritud GNU/Linux operatsioonisüsteemil põhinevat vabavaralist tarkvara, mistõttu on järgnevad filtrid arvutatud "Octave" tarkvara abil. Näiteks joonisel 6 kujutatud sinc funktsiooni saab genereerida järgmise käsuga:

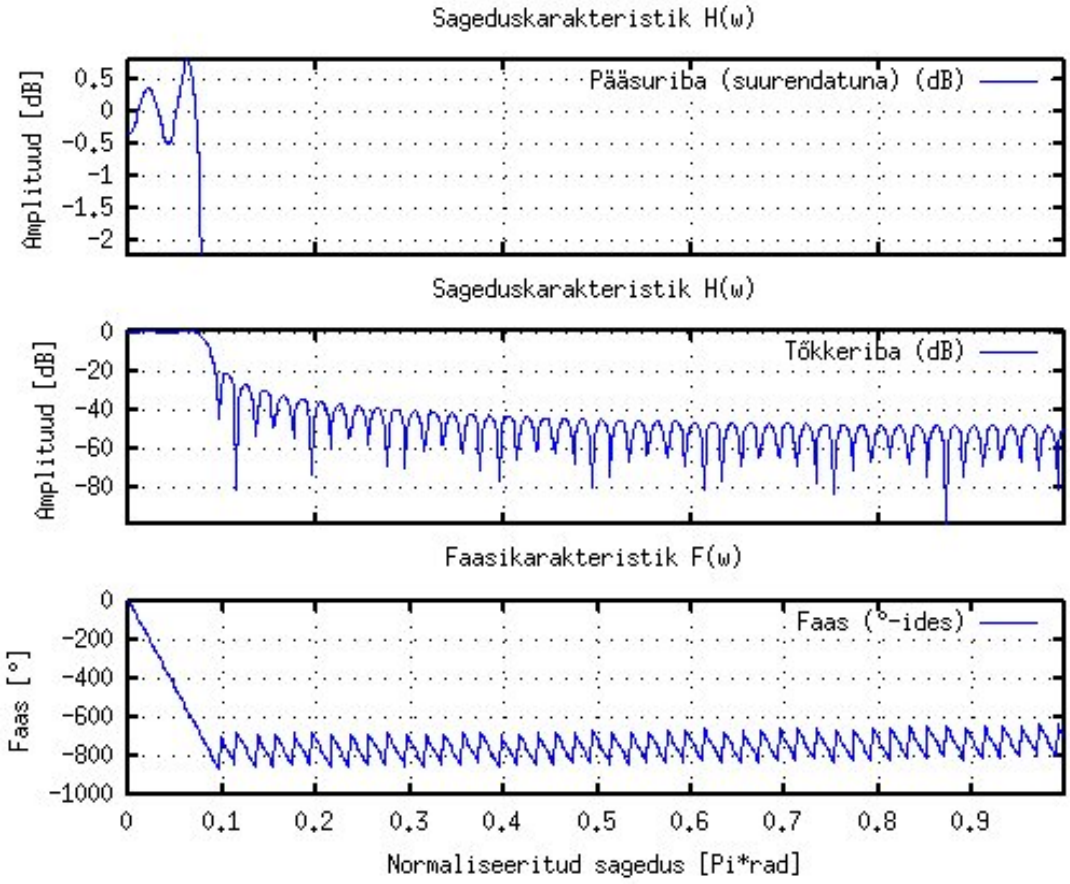
b = fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, rectwin(101));

Tulemuseks on massiiv b, mis sisaldab nelinurk-aknaga realiseeritud 101-punktilise impulsskaja $h[n]$ väärtusi, kui lõikesagedus $\omega_c = \frac{1}{12} = 0.08(3)\pi\, rad$. Tasub mainida, et arvutatud koefitsendid massivis b on ligilähedased, st valemist 19 arvutatud $h[n]$ väärtused on praktikas võimalik arvutada täpsemalt. See on seotud asjaoluga, et *fir2* funktsioon kasutab ideaalse sageduskarakteristiku esitamiseks (sagedusvallas) teatud arvu punkte (toodud näites 2^{24} ehk umbes 16.78 miljonit punkti). Täpsuse kasvatamieks tuleb suurendada punktide arvu arvestades ka asjaoluga, et seoses arvutusmahu kasvuga kulub väljundi arvutamisele märkimisväärselt rohkem aega (olenevalt riistvara arvutusvõimsusest). Antud näites genereeritud ajavalla impulsskaja maksimaalväärtus on max(b) = 0.0833333035310109, kuid valemi 19 abil arvutatud täpne sinc funktsiooni maksimaalväärtus on 0.083(3). Käesolevas töös on "Octave" tarkvara kasutamisel saadud tulemus loetud piisavalt täpseks ja parameeter 2^{24} on kasutatud ka edaspidi filtrikoefitsentide genereerimisel.

Seoses nelinurk-akna rakendamisega ideaalsele impulsskajale ei ole sageduskarakteristik enam ideaalne. Nagu eelnevalt mainitud, saab genereeritud lõplikule impulsskajale vastava reaalse sageduskarakteristiku $H(\omega)$ arvutada näiteks valemi 29 abil arvestades seoseid DTFT ja DFT vahel (seosed 25 ja 26). Süsteemi faasikarakteristiku saab leida valemist 24. Oluliselt mugavam on kasutada sageduskarakteristiku ja vastava faasikarakteristiku leidmiseks "Octave" tarkvara "freqz" funktsiooni näiteks järgmise käsuga:

freqz(b)

Tulemuseks kuvatakse joonis 7, mis kirjeldab massiivile *b* vastavat sagedus- ja faasikarakteristikut. Sageduskarakteristiku ordinaatteljel on kujutatud signaali magnituud logaritmilisel skaalal (detsibellides) ja koordinaatteljel normaliseeritud sagedus $\omega\ [\pi\, rad]$.
Faasikarakteristiku ordinaatteljestikul on kujutatud signaali faas (kraadides) ja koordinaatteljestikul jällegi normaliseeritud sagedus.



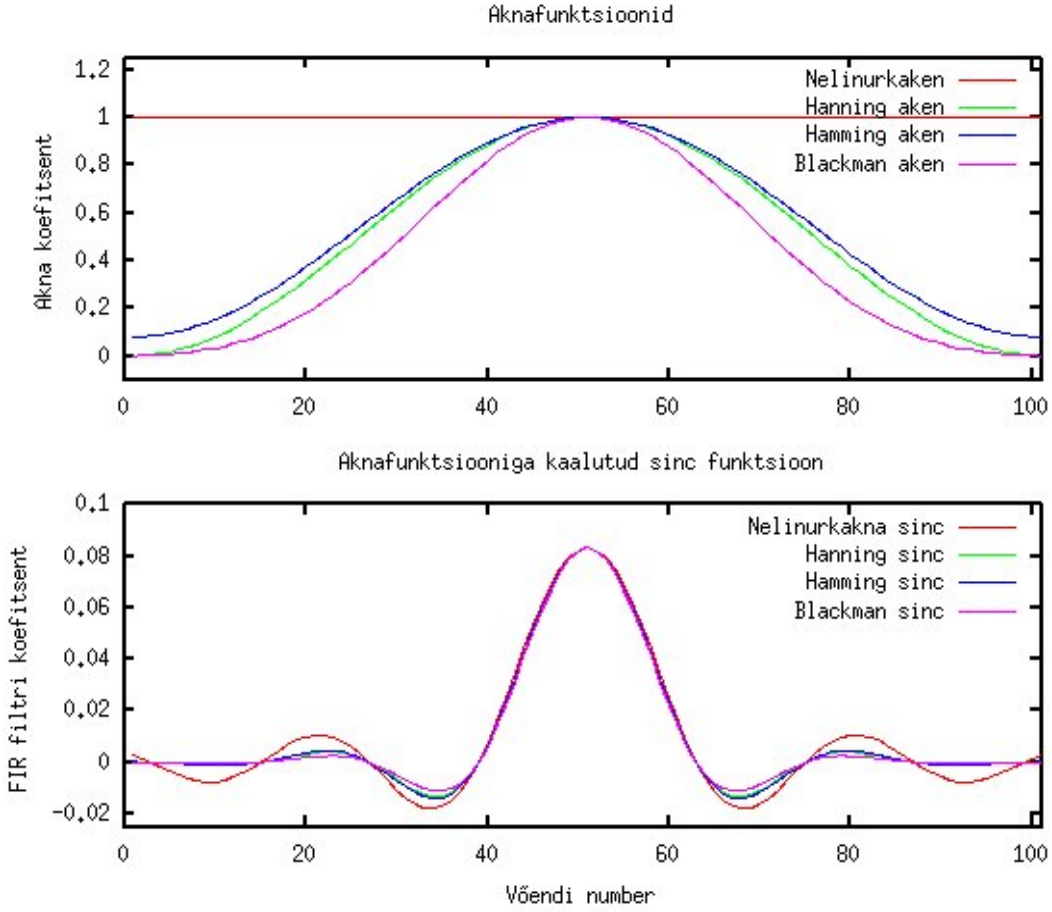
Joonis 7. Joonisel 6 kujutatud impulsskajale vastav sagedus- ja faasikarakteristik.

Jooniselt 7 ilmneb, et nelinurk-akna meetodil genereeritud impulsskajale vastav sageduskarakteristik ei ole kaugeltki ideaalne. Faasikarakteristik on ootuspäraselt lineaarne (tulenevalt antud töös käsitletavast lineaarse faasiga FIR-meetodist), kuid tõkkeriba amplituud on vähenenud vaid 20dB ja pääsuribas esineb küllaltki suuri võnkumisi (ingl. k *ringing effect* ja *ripple*). Võnkumised pääsuriba murdesageduse lähistel tulenevad nelinurk-akna olemusest. Sellist nähtust nimetatakse ka Gibbs'i efektiks (ingl. k *Gibbs phenomenon*). Ilmneb, et filtri järgu M suurendamine toob endaga kaasa võnkumiste tihenemise pääsuribas ja koondumise murdesageduse lähistele, kusjuures võnkumiste amplituud oluliselt ei vähene [5]. Seega ei paranda nelinurk-akna meetodil realiseeritud filtri järgu suurendamine võnukmisi pääsuribas.

Ilmneb, et on olemas oluliselt paremate omadustega aknafunktsioone. Järgnevalt on toodud mõnede populaarsemate aknafunktsioonide matemaatilised esitused [7]:

$$\begin{aligned}
&\omega_{hanning}(n)=0.5-0.5\cos\left(\frac{2\pi n}{M}\right), && kui \quad 0\leqslant n\leqslant M\\
&\omega_{hamming}(n)=0.54-0.46\cos\left(\frac{2\pi n}{M}\right), && kui \quad 0\leqslant n\leqslant M\\
&\omega_{blackman}(n)=0.42-0.5\cos\left(\frac{2\pi n}{M}\right)+0.08\cos\left(\frac{4\pi n}{M}\right), && kui \quad 0\leqslant n\leqslant M
\end{aligned} \tag{30}$$

Joonisel 8 on kujutatud kõnealused neli aknafunktsiooni ja ideaalse madalpääsfiltri impulsskaja $h[n]$ (joonis 6) kaalutuna vastava aknafunktsiooniga. Erinevalt nelinurkaknast on valemis 30 kirjeldatud aknafuntsioonid sujuvad ja meenutavad Gaussi kõverat. Sellest tingituna väheneb oluliselt ka Gibbsi efekt. Punase joonega kujutatud sinc vastab joonisel 6 kujutatud sinc funktsioonile. Ülejäänud funktsioonid on seoses kaalumisega originaalist erinevad.

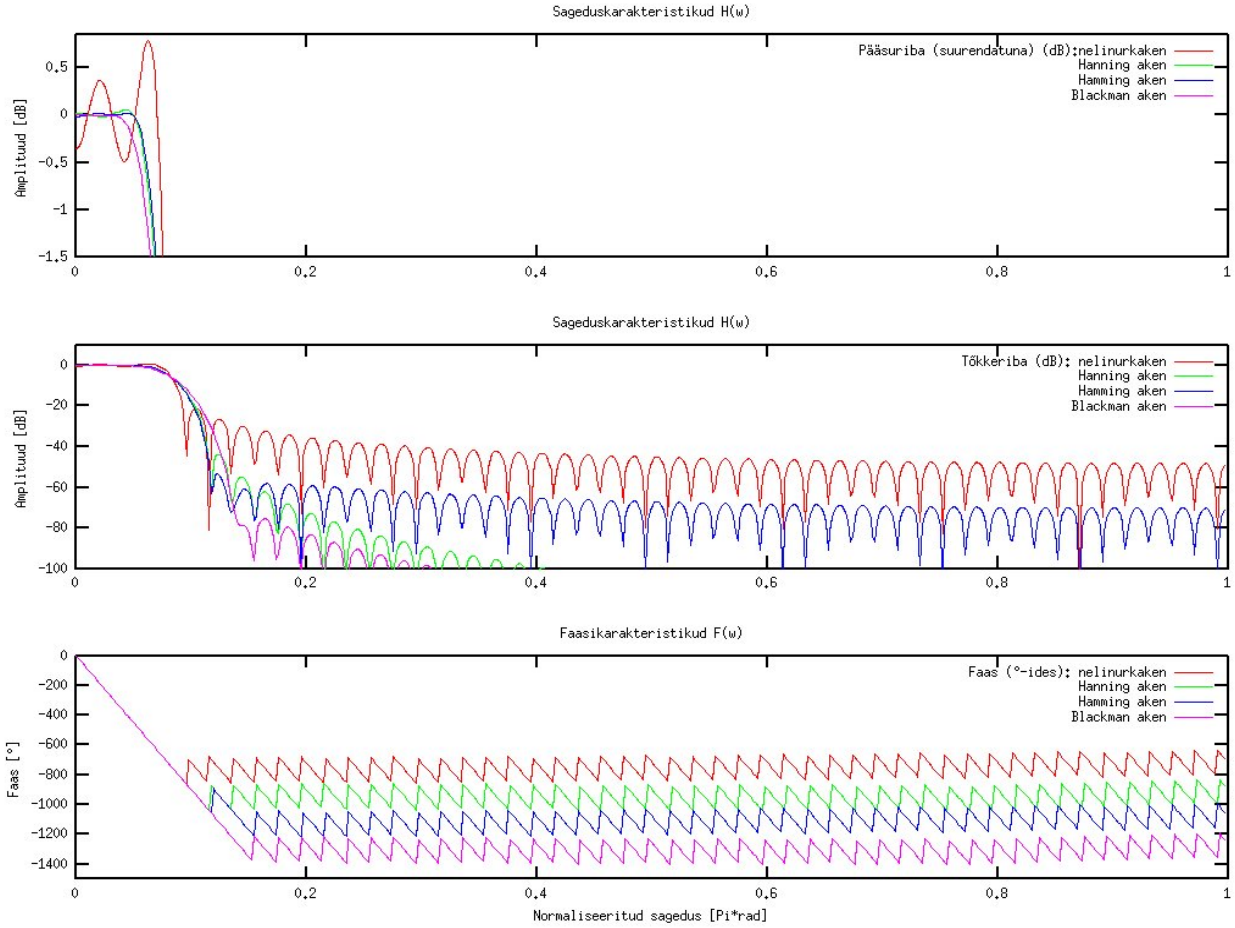


Joonis 8. Aknafunktsioonid ja nendele vastavad sinc funktsioonid, kui $\omega_c = \frac{1}{12}\pi\, rad$ *ja M=100.*

Selleks, et kujutada mitut graafikut ühel joonisel tuleb "Octave" tarkvara kasutamisel paraku veidi vaeva näha. Antud joonise genereerimiseks kasutatud programmkäsud on leitavad lisast 2. Pealtnäha ei ole sinc funktsioonid joonisel 8 üksteisest palju erinevad. Küll aga on selliste sinc funktsioonide (teisisõnu impulsskajade) sageduskarakteristikud märkimisväärselt erinevate omadustega. Joonisel 8 kujutatud sinc funktsioonide faasi- ja sageduskarakteristik on kujutatud joonisel 9. Joonise genereerimiseks kasutatud programmkäsud on toodud lisas 3.

Jooniselt 9 on näha, et tõkkeriba amplituud väheneb oluliselt kasutades nelinurk-akna asemel valemis 30 defineeritud aknaid. Selles osas on parim tulemus kasutades Blackman-akent (umbes -75dB). Samas võib öelda, et tõkkeriba summutamisega kaasneb Blackman-akent kasutades üleminekuriba laienemine. Näiteks Hamming-akna korral on üleminekuriba kitsam, kuid tõkkeriba amplituud suurem (umbes -53dB). Hanning-akna kahjuks räägib vähene tõkkeriba summutamine (umbes -44dB) ja kerge

võnkumine pääsuribas. Ilmneb, et nelinurkaken on kõige kitsama üleminekuribaga, kuid äärmiselt suure võnkumisega pääsuribas (mida ei paranda oluliselt ka filtri järgu M suurendamine) ja liiga vähene tõkkeriba summutamine (umbes -21dB). Antud võrdluses võib tunnistada siiski Blackman-akna paremust võrreldes nelinurk-, Hanning-, ja Hamming-aknafunktsiooniga. Kui üleminekuriba laiust on tarvis vähendada, siis piisab filtri järgu M suurendamisest.



Joonis 9. Faasi- ja sageduskarakteristikud, mis vastavad joonisel 8 kujutatud impulsskajadele.

Blackman-aknafunktsiooniga realiseeritud filtri umbkaudse üleminekuriba laiuse $\Delta\omega_{trans}$ võib leida seosest 31, kasutades konstanti $c_{blackman}=5.5$ [7]:

$$\Delta\omega_{trans}\simeq\frac{c_{blackman}}{(M+1)}=\frac{5.5}{101}=0.0545\pi\,rad \quad (31)$$

Tulemuseks on normaliseeritud sagedus, mis iseloomustab üleminekuriba laiust. Selleks, et esitada antud tulemus hertsides, tuleb teada võendamissagedust. Näiteks, kui võendamissagedus $f_s=48\text{kHz}$, saab vastavalt valemile 14 leida üleminekuriba laiuse Δf_{trans} järgmiselt:

$$\Delta f_{trans}=\frac{\Delta\omega_{trans}f_s}{2\pi}=\frac{0.0545\pi\,48\text{kHz}}{2\pi}\simeq 1.3\text{kHz} \quad (32)$$

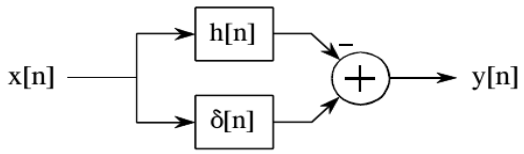
Joonisel 9 kujutatud Blackman-akna meetodil realiseeritud madalpääsfiltri impulsskaja (sinc funktsiooni) matemaatiline esitus on tuletatav sidudes omavahel valemid 19 ja 30 järgmiselt [2]:

$$h[n]=\frac{\sin(\omega_c(n-\frac{M}{2}))}{\pi(n-\frac{M}{2})}[0.42-0.5\cos(\frac{2\pi n}{M})+0.08\cos(\frac{4\pi n}{M})], \quad kui \quad 0\leqslant n\leqslant M \quad (33)$$

Valemi 33 kasutamise eelduseks on paarisarvuline parameeter M (filtri järk) ja nulliga jagamise vältimiseks lugeda funktsiooni väärtuseks $\frac{\omega_c}{\pi}$ punktis $h[\frac{M}{2}]$. Valem seob endas algse sinc funktsiooni, impulsskaja hilistamise ja Blackman-akna. Valemile 33 vastavad filtrikoefitsendid on toodud lisas 4, kus filtri järk M=100 ja murdesagedus $\omega_c=\frac{1}{12}\pi rad$.

2.3.4 Kõrgpääsfilter ja spektraalne inversioon

Kui sobiv madalpääsfilter on leitud ja vastavad koefitsendid genereeritud, saab disainitud madalpääsfiltri abil leida lihtsa meetodiga komplementaarse kõrgpääsfiltri. Selleks võib kasutada kahte meetodit: spektraalne inversioon (ingl. k *spectral inversion*) ja spektraalne reversseerimine (ingl. k *spectral reversal*). Esimene neist põhineb madalpääsfilti läbinud signaali lahutamisel originaalsignaalist (st järgi jääb kõrgpääsfilter). Teine meetod põhineb asjaolul, et reaalsete väärtustega diskreetsete signaalide sagedusspekter on perioodiline perioodiga 2π ja sümmeetriline π suhtes (pt 2.3.1). Seega piisab sageduskarakteristiku "nihutamisest" kuni madalpääsfiltri "peegelpilt" (kõrgpääsfiltri sageduskäik) jõuab soovitud murdesageduseni ω_c .
Antud töös on tutvustatud esimest ehk spektraalse inversiooni meetodit, mida illustreerib joonis 10 [2]:



Joonis 10. Spektraalse inversiooni illustratsioon.

Joonisel 10 on kujutatud sisendsignaali $x[n]$ läbimas kahte paralleelset alamsüsteemi. Esimene neist (ülemine) on eelmises alampeatükis disainitud madalpääsfilter impulsskajaga $h[n]$. Teise (alumise) alamsüsteemi impulsskajaks on ühikimpulss $\delta[n]$ ehk teisisõnu see alamsüsteem ei mõjuta sisendsignaali, vaid laseb selle muutumatul kujul läbi. Madalpääsfiltri süsteemi läbinud signaal lahutatakse teist alamsüsteemi läbinud signaalist (muutumata jäänud sisendsignaalist). Piltlikult öeldes lahutatakse sisendsignaalist madala sagedusega komponendid, mille tulemusena jäävad järele vaid kõrgema sagedusega komponendid. Selleks, et selline meetod oleks korrektne, peavad mõlemat süsteemi läbinud signaalid olema sama faasiga. Järelikult peab madalpääsfilter olema disainitud kui lineaarse- või nullise faasiga filter ja parameeter M peab olema paarisarvuline (tekib M+1 koefitsenti). Teisisõnu peab disainitud madalpääsfiltri impulsskaja olema sümmeetriline punkti $h[\frac{M}{2}]$ suhtes. Eelmises alampeatükis genereeritud filtrid vastavad nendele eeldustele. Lisaks peab

antud meetodi rakendamiseks olema punktis $h[\frac{M}{2}]$ lisatud impulsskajale ühikimpulss (ehk tegemist on sisendsignaali hilistamisega). Seega, matemaatiliselt (ja tarkvaraliselt) on tegemist ülilihtsa teisendusega: madalpääsfiltri impulsskaja inverteeritakse ja lisatakse väärtus 1 impulsskaja keskpunktile $h[\frac{M}{2}]$.[2]

Kui "Octave" keskkonnas on madalpääsfilter genereeritud käsuga

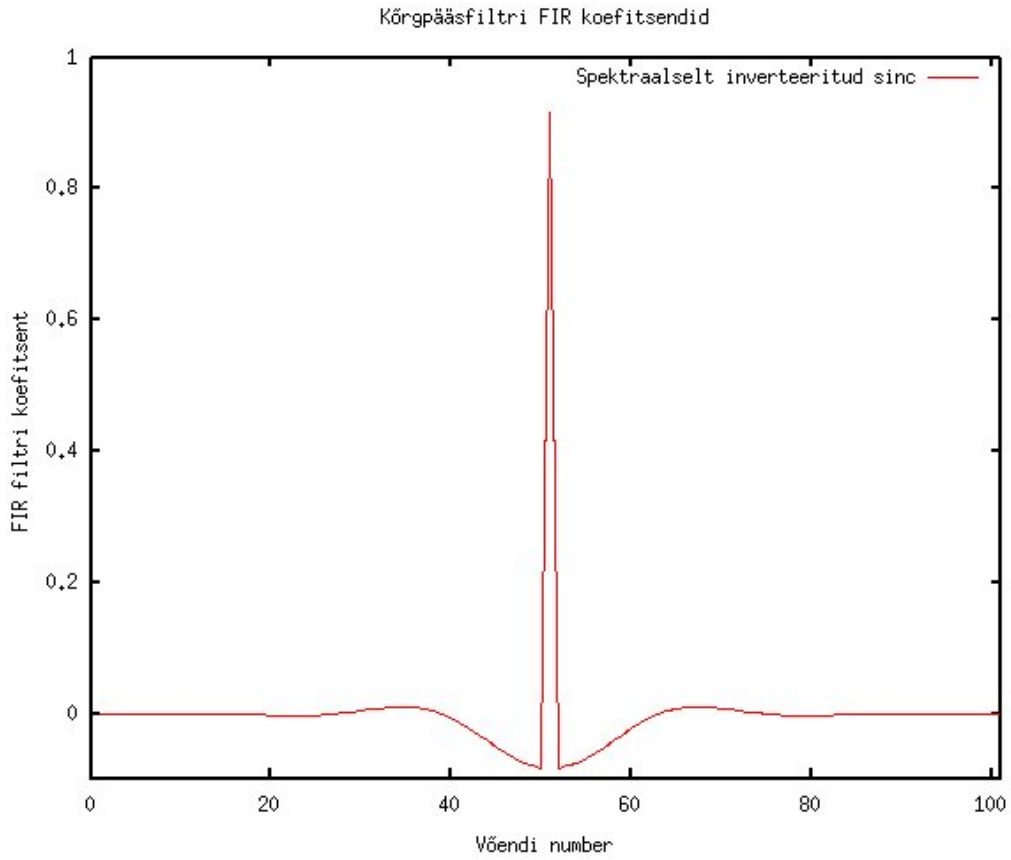
b4=fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, blackman(101));

siis massiivile b4 vastava sageduskäigu spektraalseks inverteerimiseks piisab järgmistest käskudest:

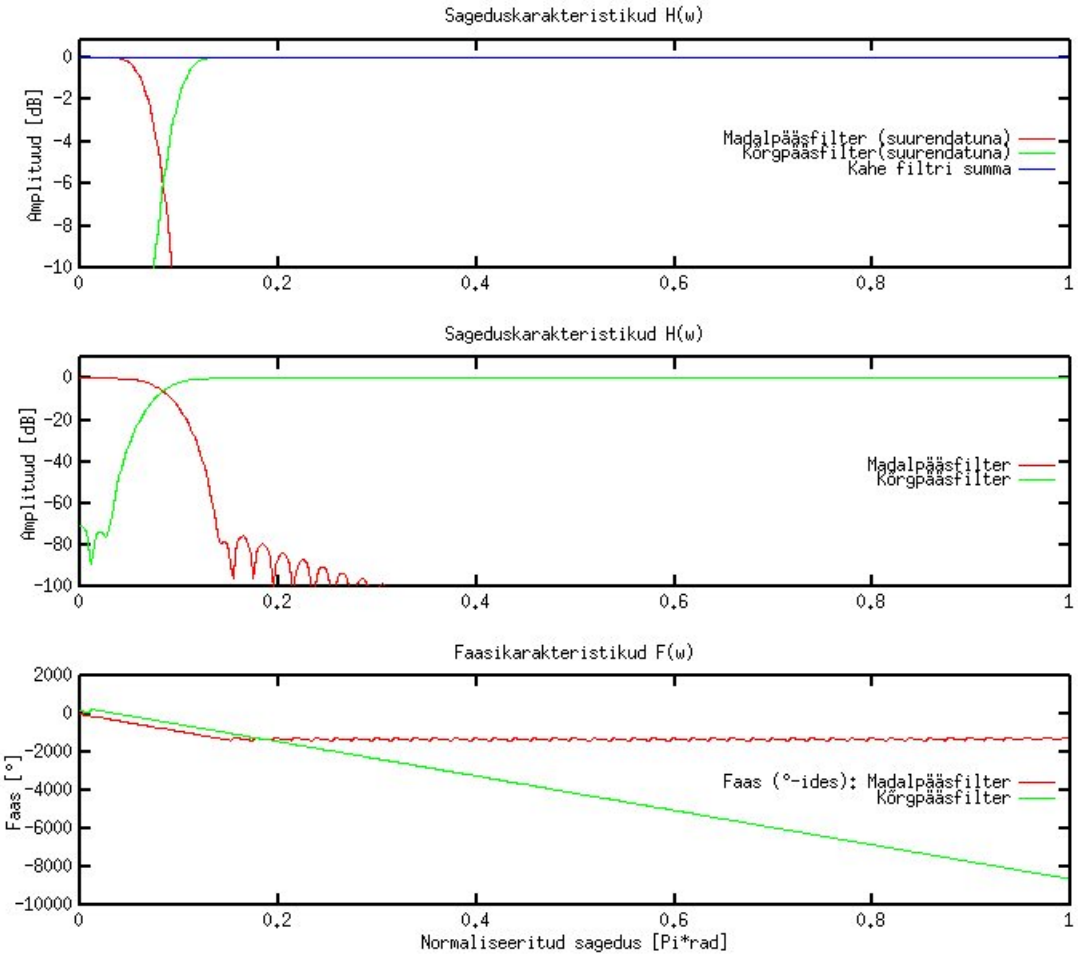
b4_high=-b4;

b4_high(51)=b4_high(51)+1;

Joonisel 11 on kuvatud spektraalse inversiooni meetodil genereeritud FIR-kõrgpääsfiltri impulsskaja. Spektraalne inversioon on rakendatud eelmises alampeatükis genereeritud Blackman-aknafunktsiooniga genereeritud FIR-madalpääsfiltrile, kus filtri järk M=100 ja murdesagedus $\omega_c = \frac{1}{12}\pi rad$. Leitud kõrgpääsfiltri impulsskaja (FIR-filtri koefitsendid) on toodud lisas 5. Järgnevalt tekib võimalus vaadelda nii madalpääs- kui ka kõrgpääsfiltri faasi- ja sageduskarakteristikut samal joonisel (joonis 12). Joonise 11 ja 12 genereerimiseks kasutatud programmkäsud "Octave" keskkonnas on toodud lisas 6 .



Joonis 11. Spektraalne inversioon joonisel 8 kujutatud madalpääsfiltrist.



Joonis 12. Madal- ja kõrgpääsfiltrite faasi- ja sageduskarakteristikud vastavalt lisadele 4 ja 5.

Tulenevalt spektraalse inversiooni olemusest on joonisel 12 kujutatud madal- ja kõrgpääsfiltri ülekannete summa (kujutatud sinist värvi joonena) väga lähedane väärtusele 1 (võib erineda pisut tänu ujukoma väärtuse ümardamisveale). Madal- ja kõrgpääsfiltri karakteristiku jooned ristuvad murdesagedusel väärtusel umbes -6dB ehk teisisõnu punktis, kus väljundsignaali amplituud on vähenenud kaks korda (valem 34) [2].

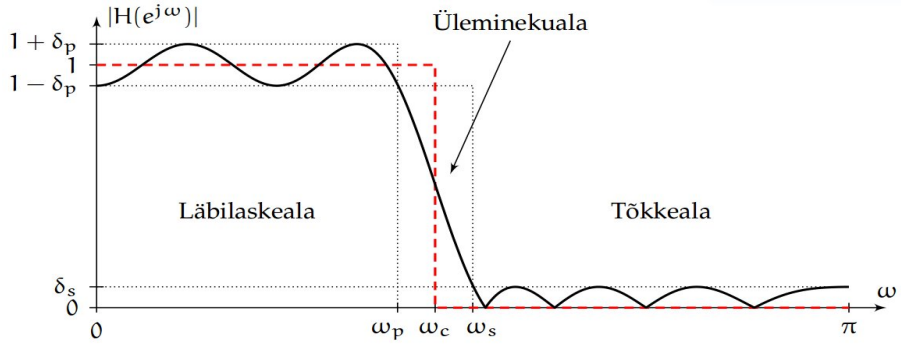
$$A_{dB} = 20\log\frac{A_{väljund}}{A_{sisend}} = 20\log 0.5 \simeq -6.02\text{dB} \qquad (34)$$

Analoogfiltrite korral kasutatav -3dB punkt ei oma antud FIR-filtrite kontektis erilist sisu. -6dB kasutamine on otstarbekam, sest aknameetodil genereeritud FIR-filtrid on 50% amplituudpunkti suhtes sümmeetrilise pääsu- ja tõkkeribaga. Ehk võnkumine pääsuribas on sümmeetriline võnkumisega tõkkeribas.[2]

2.3.5 Alternatiivsed meetodid FIR-filtrikoefitsentide leidmiseks ja rakendamiseks

Aknameetod ei ole kaugeltki ainus võimalus FIR-filtrite genereerimiseks. Selliselt leitud filtreid on lihtne mõista ja analüütiselt ning matemaatiliselt arvutada,

kuid nad ei ole “optimaalsed” klassikalise filtridisaini mõttes. Lisaks on tegemist “katse-eksituse” meetodiga, ehk sageduskarakteristiku täpsed omadused selguvad alles siis kui filter on lõpuni disainitud. Samas, teatud olukordades on otstarbekam alustada filtri disaini justnimelt soovitud sageduskarakteristiku omaduste määramisega.



Joonis 13. Ideaalne madalpääsfilter (katkendjoon) lõikesagedusega ω_c *ja sellele vastava realse madalpääsfiltri amplituudikarakteristik (pidevjoon). [4]*

Joonisel 13 on tähistatud läbilaskeala vahemikus $\omega\in[0,\omega_p]$, tõkkeala vahemikus $\omega\in[\omega_s,\pi]$ ja nende vahele jääv üleminiku- ehk siirdeala vahemikus $\omega\in[\omega_p,\omega_s]$. Vastavates vahemikes määratakse filtri lubatud sageduskarakteristiku amplituudi väärtused, milleks on läbilaskeala korral $1-\delta_p\leqslant|H(e^{j\omega})|\leqslant 1+\delta_p$ ja tõkkeala korral $0\leqslant|H(e^{j\omega})|\leqslant\delta_s$. Teisisõnu määratakse parameetritega δ_p ja δ_s sageduskarakteristiku amplituudi väärtuse maksimaalne lubatud viga võrreldes ideaalse amplituudkarakteristikuga. Neid vigu esitatakse tihti logaritmiliselt (dB) [4]:

$$A_p=20\log\frac{1+\delta_p}{1-\delta_p}[dB] \qquad (35)$$
$$A_s=-20\log(\delta_s)\ [dB]$$

Soovitud üleminekuala laius $\omega\in[\omega_p,\omega_s]$ määratakse samuti filtri disainiülesande alguses, erinevalt aknameetodist.

Ideaalse filtri aproksimatsiooni kvaliteedi hindamiseks on kasulik filtrit hinnata teatud kriteeriumi järgi. Järgnevalt on toodud kolm enamlevinud kriteeriumit filtri hindamisks [5]:

1. Minmaks-veaga disain. [4]

Minmaks-veaga disaini käigus minimiseeritakse maksimaalse vea amplituudi ehk erinevust disainitava ja ideaalse filtri sageduskarakteristiku amplituudide vahel. Kaalutud veafunktsioon $E(\omega)$ on defineeritud järgmiselt:

$$E(\omega)=W(\omega)(|H(e^{j\omega})-H_d(\omega)|),\quad kus$$

$$W(\omega)=\begin{cases}1, & kui\quad \omega\in[0,\ \omega_p]\\ \dfrac{\delta_p}{\delta_s}, & kui\quad \omega\in[\omega_s,\pi]\end{cases} \qquad (36)$$

$$H_d(\omega)=\begin{cases}1, & kui\quad \omega\in[0,\ \omega_p]\\ 0, & kui\quad \omega\in[\omega_s,\pi]\end{cases}$$

$H(e^{j\omega})$ - reaalne sageduskarakteristik (aproksimatsioon funktsioonist $H_d(\omega)$)

$H_d(\omega)$ - soovitud sageduskarakteristik

$W(\omega)$ - kaalufunktsioon

Minmaks-veaga disaini puhul minimeeritakse veafunktsiooni $E(\omega)$ absoluutväärtuse suurimat väärtust läbilaske- ja tõkkeala hõlmavas sagedusvahemikus, kusjuures ϵ on etteantud suurus (lubatud hälve) :

$$\epsilon = max|E(\omega)|, \quad kus \quad \omega \in [0;\omega_p] \cup [\omega_s;\pi] \qquad (37)$$

Minmaks-disaini tulemusena saadakse tüüpiliselt filter, mille amplituudikarakteristikus esinevad ühesuguse amplituudiga lainetused kogu läbilaskeala või tõkkeala ulatuses (ingl. k *equiripple linear phase FIR filters*) [4]. Üheks levinumaks minmaks-veaga disaini veafunktsiooni minimeerimise algoritmiks on Parks-McClellani meetod.

2. Vähimruutude veaga disain.[4]

Vähimruutude veaga (L_2 veaga) disaini korral minimeeritakse järgmist suurust:

$$E_2 = \int_{\Omega} [W(\omega)(|H(e^{j\omega})| - H_d(\omega))]^2 d\omega, \quad kus \quad \Omega \in [0;\omega_p] \cup [\omega_s;\pi] \quad (38)$$

Minimeeritakse disainitava ja soovitud filtri sageduskarakteristiku amplituudidevahelise vea energiat. Vähimruutude veaga disainile on iseloomulik, et amplituudikarakteristikus esinevad lainetuse amplituudid suurenevad, kui sagedusega liikuda filtri lõikesagedusele lähemale. [4]

3. Maksimaalselt lame disain.[4]

Maksimaalselt lameda disaini korral kasutatakse sageduskarakteristiku amplituudi Taylori-ritta arendust teatud sageduspunktiks, milleks on sageli $\omega = 0$. Selliselt disainitud filtri amplituudkarakteristik on monotoonselt kahanev, kui tegemist on madalpääsfiltriga. Sel meetodil disainitud filter on tunduvalt kõrgema järguga kui samadele tingimustele vastav minmaks- või vähimruutude veaga disainitud filter.

Mis iganes meetodil FIR-koefitsendid ka ei leita, on vajalik nende rakendamine praktikasse, et katsetada filtrite koosmõju teiste süsteemi osadega: helivõimendite ja valjuhäälditega, samuti ruumiomadustega jne. Igal helisüsteemil on oma eripärad, mistõttu üks filtrikomplekt, mis osutub teatud süsteemis kõige sobivamaks, ei pruugi seda olla teises helisüsteemis ja/või ruumis. Järelikult on vajalik korduv katsetamine ehk süsteemi väljunud-helisignaali kuulamine (või mõõtmine – kuidas keegi soovib ja suudab).

Nagu varem kirjeldatud, muutub sisendsignaali konvoleerimine filtrisüsteemi impulsskajaga väga arvutusnõudlikuks protsessiks, kui filtri järk on suur. Eriti kõrget järku filtrisüsteemide realiseerimiseks võib loobuda signaali töötlemisest ajavallas (konvolutsioon) ja rakendada vastavaid meetodeid sagedusvallas. Sel juhul toimitakse järgnevalt [2]:

- Teisendatakse digitaalne sisendsignaal DFT-d rakendades sagedusvalda ($x[n] \Rightarrow X[n]$)
- Korrutatakse sagedusvalla sisendsignaal soovitud sageduskarakteristikuga
  $$Y[n] = X[n] \times H[n] \qquad (39)$$
- Teisendatakse sagedusvalla väljundsignaal IDFT-d rakendades ajavalda ($Y[n] \Rightarrow y[n]$)

Sagedusvalla signaalitöötluse puhul oluline mõista, et korrutamine sagedusvallas on ekvivalentne konvolutsiooniga ajavallas. See ilmneb ka võrreldes sagedusvalla valemit 39 ajavalla valemiga 6. Konverteerimine ajavallast sagedusvalda ja tagasi paistab olevat veel mahukam kui ajavallas konvolutsiooni arvutamine, kuid teatud filtrijärgust alates muutub sagedusvalla signaalitöötlus otstarbekamaks, kasutades nutikalt ära FFT-d (ingl. k F*ast Fourier' Transform*) ja süsteemi iseloomulikke omadusi

(näiteks impulsskaja sümmeetrilisus).

Siiani on antud töös toodud ülevaade FIR-filtrite ja signaalitöötluse alustaladest – konvolutsioon, sinc funktsioon ja filtrikoefitsentide leidmine. Käesolevaks hetkeks on loetud FIR-filtrite teoreetiliste aluste kirjeldus piisavaks, et alustada nende realiseerimisega TI C5000 seeria DSP-l vastavalt ülesandepüstitusele.

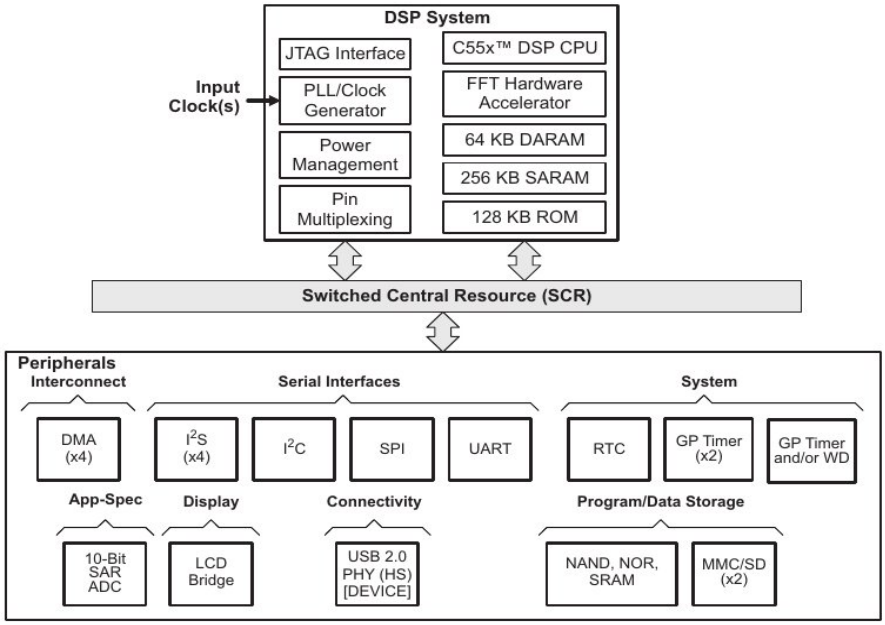
3. Ülevaade kasutatavast riistvarast

3.1 TMS320VC5505 eZdsp USB Stick

Selles peatükis on toodud ülevaade DSP-dest üldisemalt, TI VC5505 püsikoma DSP spetsiifilisematest omadustest ja "TMS320VC5505 eZdsp USB Stick" arendusplaadist.

Nagu juba mainitud, on DSP-d ideaalsed signaalitöötluse ülesannete lahendamiseks. Võrreldes "von Neumann'i" arhitektuuriga mikroprotsessoritest ja laiatarbe-arvutitest on DSP-l oluline eelis: protsesori ja mälu vahelisi andmesiine on mitu. Näiteks olgu üks neist seotud mäluga "A", kus hoitakse tüüpiliselt programmi instruktsioone (mis kirjeldab teatud matemaatilist tehet) ja teine mäluga "B", kus hoitakse andmeid. Lisaks on DSP-l eraldi siinid andmete adresseerimiseks ja andmete transpordiks. Selline arhitektuur annab paremad eeldused teostada kiiresti matemaatilisi tehteid nagu näiteks korrutamine ja liitmine (MAC), kuna instuktsioone saab laadida CPU-sse (ingl. k *Central Processing Unit*) paralleelselt andmetega. Moodsamatel DSP-del on CPU-ssse integreeritud vahemälu "C" tihedalt kasutatavate instruktsioonide tarbeks (ingl. k *Instruction Cache* või *I-Cache*). Vahemälu "C" olemasolu võimaldab paigutada osa signaali töötlemiseks vajalikest andmetest mällu "A" (nn sekundaarsed andmed). FIR-filtreerimise ülesande puhul võiks näiteks mälus "A" hoida filtrikoefitsente ja mälus "B" võiks olla sisendvõendite puhver. Kui korrutamise ja liitmise instruktsioonid on varasemalt laetud mällu "C", saab ideaalis ühe takti jooksul transportida CPU-sse nii sisendvõendi, filtrikoefitsendi kui ka instruktsiooni. Selline informatsiooni transportimine on märksa efektiivsem kui klassikalise "von Neumann" arhitektuuriga arvuti puhul. Antud töös kasutatava VC5505 DSP erinevad andmesiinid ja CPU arhitektuur on täpsemalt kirjeldatud TI dokumendis SWPU073E [8].

Vastavalt joonisele 14 (TI dokument SPRUFP0C [9]) on VC5505 püsikoma-protsessoril 128KB ROM (ingl. k *Read Only Memory*) mälu ja 320KB RAM (ingl. k *Random Access Memory*) mälu. Viimane jaguneb 64KB DARAM (ingl. k *Dual-Access RAM*) ja 256KB SARAM (ingl. k *Single-Access RAM*) tüüpi mäluks. Lisaks on protsessoril ka EMIF-liides (ingl. k *External Memory Interface*), mille detailsem kirjeldus on toodud TI dokumendis SPRUFO8 [10]. EMIF hõlpsustab lisada CPU-le välist asünkroonset tüüpi mälu (näiteks EEPROM, NOR, NAND ja SRAM) [11]. Antud töös kasutatav "TMS320VC5505 eZdsp USB Stick" arendusplaat sisaldab välist 64KB EEPROM mälu (ingl. k *Electrically Erasable Programmable ROM*), kuhu võib EMIF vahendusel salvestada programmkoodi, mida seade kasutab programmi üles-laadimiseks (ingl. k *bootloader*). Tulenevalt EEPROM olemusest püsib selles mälus programmkood ka peale toitekatkestust, mistõttu saab arendusplaati kasutada eraldiseisva süsteemina (ingl. k s*tandalone device*).



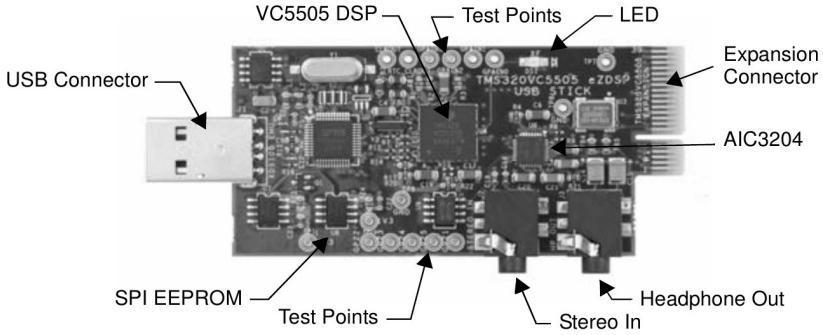
Joonis 14. VC5505 püsikoma DSP plokkskeem. [9]

DSP-l on signaalitöötluse kiirendamiseks riistvaraline lahendus otse mällu kirjutamiseks – DMA (ingl k. *Direct Memory Access*) kontroller. See tähendab, et andmeid (antud töö kontekstis helisignaali sisend- ja väljundvõendite jada) saab transportida mällu (ja perifeerseadmetesse) ilma CPU registreid läbimata. Teisisõnu, CPU saab tegeleda väljundvõendite välja arvutamisega samal ajal, kui uued sisendvõendid ja töödeldud väljundvõendid DMA abil mällu (puhvritesse) ja/või perifeerseadmesse kirjutatakse. VC5505 DMA kirjeldus on toodud TI dokumendis SPRUFO9 [12], mille kohaselt on antud DSP-l neli DMA-kontrollerit, millest igaühel on neli kanalit, seega kokku saab kasutada kuni 16 DMA kanalit. Iga kanal on võimeline saatma CPU-le katkestuse signaali (ingl. k *interrupt*), mida võib kasutada näiteks CPU teavitamiseks teatud hulga sisendvõendite kohalolekust mälus. Peale selle on võimalus iga DMA kanal sünkroniseerida teatud perifeerseadme sündmusega, ehk teavitada DMA-d andmete valmisolekust perifeerseadme puhvris. Antud töös tuleb arvestada ka teatavate kitsendustega VC5505 DSP DMA osas – nimelt on erinevate DMA-kontrolleritega võimalik kasutada ainult teatud perifeerseadmeid ja mälu. Näiteks välise mäluga saab siduda vaid kontrolleri DMA3. Täpsem kirjeldus on toodud lisas 7. Lisaks tuleb arvestada adresseerimise erinevustega. Näiteks DMA kasutab adresseerimiseks baidi aadresse, aga CPU kasutab "sõna" aadresse (ingl. k *word address*) ehk 16-bitise andmeühiku aadresse. Seega on vajalik tarkvaraline aadresside konverteerimine.[12]

DSP-del on tüüpiliselt riistvaraline lahendus korrutamise ja liitmise teostamiseks – MAC-üksus, mida on VC5505 protsessoril kaks tükki. VC5505 MAC-üksused võimaldavad kahepeale kokku sooritada ühe takti jooksul kaks 17 x 17 bitiste andmete korrutamist ja kaks 40-bitist liitmise (või lahutamise) tehet. Taktsageduse 100MHz juures on VC5505 DSP teoreetiliselt võimeline 200-ks MIPS-iks (ingl. k *Million Instructions Per Second*). Sellise tulmuse võimaldab ilmselt asjaolu, et MAC-üksuseid on kaks. Reaalselt võib 200 MIPS-i osutuda kättesaamatuks olenevalt sellest, milliseid instruktsioone kasutatakse ja kui efektiivselt on DSP programmeeritud.

Joonis 15 kujutab TI poolt disainitud "TMS320VC5505 eZdsp USB Stick"

arendusplaati. Programmeerimiseks on kasutusel USB-liides, mille kaudu saab arendusplaat ka toite (VC5505 poolt tarbitav elektriline võimsus on alla 50mW taktsagedusel 100MHz [11]). Programmeerimiseks ja seadme töö jälgimiseks on arendusplaadile integreeritud "XDS100" emulaator, mis teeb arendusplaadi sidumise PC-ga (ingl. k *Personal Computer*) hõlpsaks. Ametlikult toetab "TMS320VC5505 eZdsp USB Stick" arendustarkvarana CCSv4 (Code Composer Studio version 4), kuid praktikas sobib kasutamiseks ka uuem CCSv5. Viimane on muuhulgas kasutatav ka vabavaralisel GNU Linux operaratsioonisüsteemil.



Joonis 15. TMS320VC5505 eZdsp USB Stick rev B. [13]

Joonisel 15 on kujutatud ka stereo analoog-helisignaali sisend- ja väljundliides. Vastavad DAC ja ADC (ingl. k *Analog-to-Digital Converter*) muundid asuvad TLV320AIC3204 mikrokiibis, nn "audio koodek"-is. Nii DAC kui ka ADC tulevad toime kuni 192kHz võendamissagedusega, sealjuures 32-bitiste võenditega, ja põhinevad delta-sigma modulaatoril. TLV320AIC3204 audio koodekil on palju programmeeritavaid funktsioone, mis on täpsemalt kirjeldatud TI dokumendis SLAA557 [14]. Antud töö ülesandepüstituse kohaselt peab süsteemi sisendiks olema digitaalne helisignaal, mistõttu pakub esialgu suuremat huvi DAC osa koodekist. Samas jätab ADC võimaluse seadistada süsteemi sisendiks ka analoog-helisignaali.

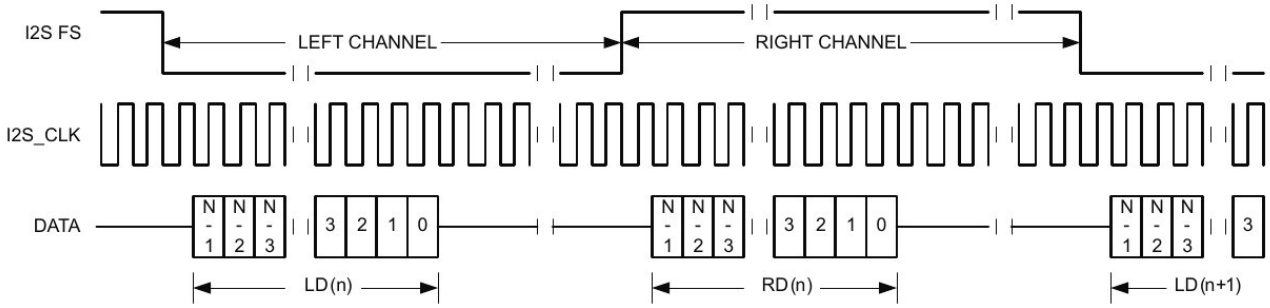
Digitaalse helisignaali sisend- ja väljundliidesena on kasutatav I2S-liides, mille kontaktid on toodud plaadi serval asuvale laienduspistikule (ingl. k *expansion connector*). Vastavalt joonisele 14 on I2S-kanaleid 4, mis on võimalik laienduspistikule juhtida teatud registri - EBSR (ingl. k *External Bus Selection Register*) abil. Nimetatud register koosneb omakorda SP0MODE, SP1MODE ja PPMODE bittidest, mis määravad vastavalt jadaliidese 0, jadaliidese 1 ja paralleel-liidese seadistuse [9]. Iga I2S-kanal on ligipääsetav teatud DMA-kontrolleri poolt. Tasub märkida, et I2S-kanalile vastava DMA-kontrolleri nimetus ei ole üksüheses seoses, st I2S0 saab siduda DMA0-ga, aga näiteks I2S1 on võimalik siduda hoopis DMA3-ga [15].

3.2 I2S-andmesideliides

Arendusplaadi TMS320VC5505 eZdsp USB Stick ainsaks võimaluseks vastu võtta digitaalset helisignaali on kasutades I2S-liidest. Seetõttu on I2S-liidese standardil oluline koht antud töös. Järgnevalt on kirjeldatud I2S standardi olemust ja seadistusvõimalusi.

I2S (kasutatakse ka lühendeid I²S ja IIS) on loodud mikroskeemide vaheliseks elektrilise helisignaali edastamiseks ja vastuvõtmiseks (ingl. k *Integrated-Interchip Sound*), st tegemist on liidesega, mis võimaldab vahetada PCM (ingl. k *Pulse Coded*

Modulation) meetodil esitatud helisignaali andmeid. I2S on sünkroonne andmeedastusliides, st andmed ja sünkroniseerimis-signaalid edastatakse üksteisest füüsiliselt erineval liinil. I2S sünkroonse andmeedastuse olemust kirjeldab joonis 16.



Joonis 16. I2S-andmeside signaalide ajastus. [15]

Arendusplaadil "TMS320VC5505 eZdsp USB Stick" on kaks andmeedastuskanalit (DATA joonisel 16), et võimaldada samaaegne kahesuunaline andmeedastus seadmete vahel, st antud arendusplaadil on neli täisdupleks I2S liidest. Andmeid vastuvõtvat kanalit nimetatakse edaspidi RX ja andmeid väljastavat kanalit DX. Lisaks RX- ja DX-kanalitele on I2S siinis biti-sünkroniseerimise kanal (I2S_CLK joonisel 16), mis on edaspidi nimetatud B_CLK-ks. B_CLK määrab ajastuse, millal saata/vastu võtta vastavalt DX- ja RX-kanalites bitte. Vajadusel on B_CLK signaal inverteeritav parameetriga CLKPOL=1 (bitt 9 registris I2SCTRL). Kuna stereo helisignaali korral edastatakse kaks helisignaali võendit (nt vasak ja parem kanal), on kasutusel ka "kaadri" (ingl. k *frame*) sünkroniseerimise signaal (I2S_FS joonisel 16), mis määrab kaadri alguse ning ühtlasi ka vasaku ja parema helisignaalikanali võendi alguse. Kaader sisaldab endas bitijada, mis moodustab "sõnad" (võendid) mida seadmete vahel edastatakse. Tüüpilises stereosignaali kaadris on kaks 16 bitist sõna (võendit). Üldiselt tähistab FS_CLK signaali loogiline "0" vasakut kanalit ja loogiline "1" paremat kanalit. Vajadusel saab seda signaali ka inverteerida väärtustades parameeter FSPOL=1 (bitt 10 registris I2SSCTRL). Kaadri sünkroniseerimise signaali sagedus ühtib helisignaali võendamissagedusega, mistõttu on seda edaspidi nimetatud FS_CLK-ks. Kirjanduses võib kohata ka tähistusi WCLK, WDCLK ja WS vihjates mõistele "sõna" (ingl. k *word clock* ja *word select*).

Vastavalt joonisele 16 edastatakse helisignaali võendid vasak-järjestatult ja vanim bitt (ingl. k MSB – *Most Significant Bit*) esimesena. MSB esimesena saatmine lisab süsteemile painduvust – saatja ei pea teadma, millist võendi sügavust (resolutsiooni) vastuvõtja on suuteline vastu võtma. Sarnaselt ei pea vastuvõtja signaali vastuvõtmiseks teadma, mis sügavusega võendeid saatja edastab. Näiteks kui saatja saadab 16-bitiseid võendeid, aga vastuvõtja ootab 24-bitiseid võendeid, lisab vastuvõtja saabunud 16-bitisele võendile vastava arvu nulle. Antud näites viimased 8 LSB-d (ingl. k *Least Significant Bit*) väärtustatakse 0-ga. Vastupidises näites, kui saatja edastab 24-bitiseid võendeid, aga vastuvõtja suudab opereerida vaid 16-bitiste võenditega, ignoreerib vastuvõtja viimast kaheksat bitti saadetud võendist. FS_CLK ja DATA (RX/DX) signaalide vahel on tüüpiliselt 1-bitine viide, mis on vajadusel seadistatav ka kahe-bitiseks viiteks parameetirga DATADLY=1 (bitt 8 registris I2SSCTRL). Lisaks RX, TX, B_CLK ja FS_CLK signaalidele vajavad paljud seadmed (millel pole sisemist taktsageduse generaatorit) välist taktsageduse signaali, mis peab I2S kasutamisel olema B_CLK kordne. Sellist signaali nimetatakse M_CLK-ks (ingl. k *master-clock*).

Tüüpiliselt vajavad M_CLK signaali DAC-d, näiteks delta-sigma modulaatori töö jaoks.

FS_CLK, B_CLK ja M_CLK signaalid genereerib siinil olev I2S *master*-seade. Sellest tulenevalt peab I2S *master*-seadme taktsagedus olema B_CLK (ja M_CLK) kordne. Näiteks rakendustes, kus DSP valitakse I2S *master*-seadmeks, peab DSP sisemine taktsagedus (edaspidi SYS_CLK) olema valitud selliselt, et selle jagamine täisarvu kordselt võimaldaks tekitada sobiva B_CLK signaali. SYS_CLK võib olla genereeritud sisemise RTC (ingl. k *Real-Time Clock*) poolt või välise taktsageduse generaatori abil. Kui SYS_CLK on tekitatud sisemise RTC poolt, võib tekkida probleeme DSP *master*-seadmena kasutamisel. Ülesandepüstituse kohaselt peab arendatav süsteem olema võimeline toime tulema näiteks võendamissagedustega 44.1kHz ja 48kHz. Ühine nimetaja mõlema sageduse jaoks on 7.056MHz (160 x 44.1kHz ja 147 x 48kHz). Sisemise RTC kasutamisel peab sellise ülesande teostamiseks olema RTC sagedus konfigureeritav 7.056MHz kordseks. Lisaks peab olema võimalik vastav sagedus jagada nii 44.1kHz kui ka 48kHz sagedusega võendatud helisignaali B_CLK (ja M_CLK) genereerimiseks. Paraku on RTC ja selle jagajate seadistamisvõimalused tihtipeale piiratud, mistõttu ei pruugi selline ülesanne olla lahendatav, kui SYS_CLK signaali genereerib RTC ja DSP peab olema I2S *master*-seade. Üheks alternatiivseks võimaluseks on välise generaatori kasutamine SYS_CLK jaoks, teine võimalus on kasutada DSP-d I2S *slave*-seadmena. Viimasel juhul pärinevad B_CLK, FS_CLK ja M_CLK välisest (I2S *master*) seadmest.

I2S standardi peamiseks heaks omaduseks on asjaolu, et tegemist on sünkroonse andmesideprotokolliga, st B_CLK, M_CLK ning FS_CLK on füüsiliselt eraldi andmekanalitest RX ja DX. Paljude I2S protokolli alternatiivide (näiteks SPDIF) saatjad kodeerivad eelkirjeldatud CLK signaalid kokku andmekanaliga. Hiljem dekodeeritakse vastuvõtjaga kätte saadud signaal, mille käigus eraldatakse andmed CLK-signaalidest. Selle protsessi käigus on oht kaotada näiteks B_CLK-signaali perioodi täpsuses, st B_CLK-perioodi hälve (ingl. k *jitter*) ilmselt suureneb. I2S korral on vähem ohtu, et näiteks B_CLK on mõjutatud andmekanalite poolt ja vastupidi. I2S-sünkrosignaalid saab genereerida suhteliselt täpselt eeldusel, et need pärinevad kvaliteetsest generaatorist (nt ostsillaatorist).

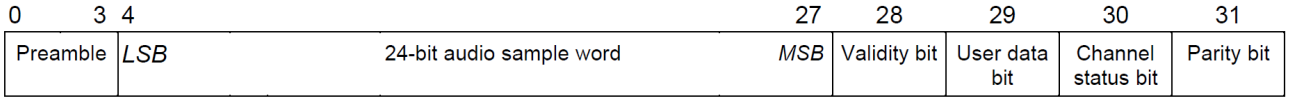
Peamiseks puuduseks I2S protokolli kasutamisel on vajadus võrdlemisi mitme kanali jaoks (vähemalt kolm: B_CLK, FS_CLK ja DATA). Peale selle, nagu I2S nimetusestki ilmneb, on see standard loodud mikroskeemide vaheliseks andmesideprotokolliks (ingl. k *Integrated Inter-chip Sound bus*). Oma olemuselt ei ole see loodud andmete transportimiseks üle pikkade vahemaade, vaid sobib andmeside edastuseks peamiselt vahemaadel, mille suurusjärk jääb sentimeetritesse. Sealjuures tuleb tähelepanu pöörata andmesiini skeemitehnilisele lahendusele, et vältida näiteks raadiosageduslikku interferentsi. Tüüpilise I2S B_CLK-signaali sagedus on megahertsi suurusjärgus. Kui näiteks kaadripikkuseks valida 64 bitti (st maksimaalselt kaks 32-bitist sõna), peab B_CLK olema vähemalt 64 korda kiirem FS_CLK-st ehk võendamissagedusest. Kui võendamissagedus on antud näites 96kHz, peab eelkirjeldatud olukorras olema BCLK vähemalt 6.144MHz. M_CLK signaal seevastu peab olema B_CLK-kordne. Näiteks kahekordse M_CLK kasutamisel eelmises näites kujuneb M_CLK-signaali sageduseks 12.288MHz. Arvestades, et sünkrosignaalid on oma kujult “kandilised” ja võrdlemisi kõrge sagedusega, on signaali spekter võrdlemisi lai (spektris on palju harmoonilisi sageduskomponente) ja sisaldab teatavat alaliskomponenti. Sellise signaali edastamine pikemate vahemaade taha (nt meetrite kaugusele) nõuab suurt pingutust. Üldiselt realiseeritakse sellised ülesanded signaali

kavala kodeerimisega.

3.3 SPDIF-vastuvõtja sidumine C5505 DSP-ga

SPDIF (ingl. k *Sony/Philips Digital Interconnect Format*, samuti tuntud kui *Sony/Philips Digital Interface* ja *IEC60958*) on liides digitaalse helisignaali edasikandmiseks. SPDIF-protokoll põhineb AES3 standardil (ingl. k *Audio Engineering Society*). Erinevus laiatarbelisele- (SPDIF) ja professionaalsele sihtgrupile suunitletud (AES3) standardites seisneb peamiselt kasutatud pingenivoode ja pistikühenduste poolest. SPDIF-protokoll võimaldab tüüpiliselt transportida kaks helisignaali kanalit võendamissagedusega kuni 192kHz ja resolutsiooniga kuni 24 bitti. Igale kuni 24-bitisele helivõendile lisandub 8 bitti vastavalt joonisele 17. Järgnevalt on toodud nende bittide lühikirjeldus ja/või eesmärk:

- Kõlblikkuse bitt (ingl. k *validity bit*) määrab võendi sobilikkuse (kas sobib muundamiseks DAC-s)
- Kanali staatus (ingl. k *channel status*) kannab endas lisainformatsiooni iga helikanali kohta. Kanali staatuse defineerimise osas erinevad SPDIF ja AES3/EBU standardid üksteisest.
- Kasutaja defineeritud bitt (ingl. k *user data*) on vaba bitt, mille funktsiooni võib määrata kasutaja.
- Paarsusbitti (ingl. k *parity bit*) kasutatakse paaritu arvu vigade tuvastamiseks andmeedastusel. Teisisõnu on bitt määratud selliselt, et bittides 4 kuni 31 (kokku 28 bitti) sisaldub paarisarv väärtusi "1" ja paarisarv väärtusi "0".
- Algbitid (ingl. k *preambles*) on spetsiifilised bitid mida rakendatakse sünkroniseerimise eesmärgil. Algbitid ei ole kahendfaas-kodeeritud.

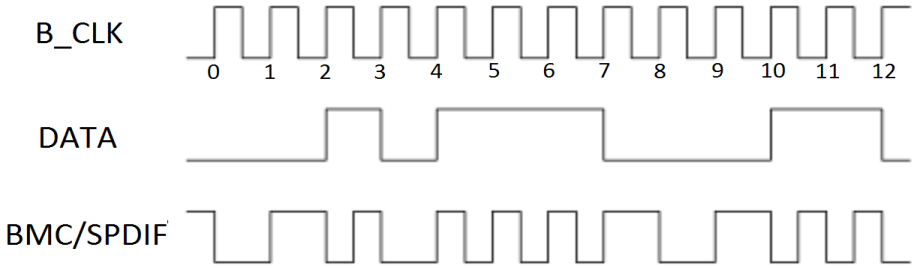


Joonis 17. AES3/SPDIF alam-kaadri struktuur. [16]

Transpordiks kasutatav meedia võib olla elektriline koaksiaalkaabel (näiteks varjestatud 75Ω takistusega koaksiaalkaabel signaali transportimiseks kuni 20 meetri kaugusele) või optiline (näiteks nn Toslink plastikfiiber helisignaali transportimiseks kuni 10 meetri kaugusele). SPDIF- (ka AES3) protokollis edastatakse kõik andmed ühel füüsilisel kanalil. Saatja kodeerib sünkrosignaali ja helisignaali võendid kokku kasutades kahendfaas-kodeerimise meetodit (ingl. k BMC - *Biphase Mark Code*). Teisisõnu on SPDIF puhul tegemist asünkroonse andmeedastus-protokolliga erinevalt I2S-st, mis on oma olemuselt sünkroonne.

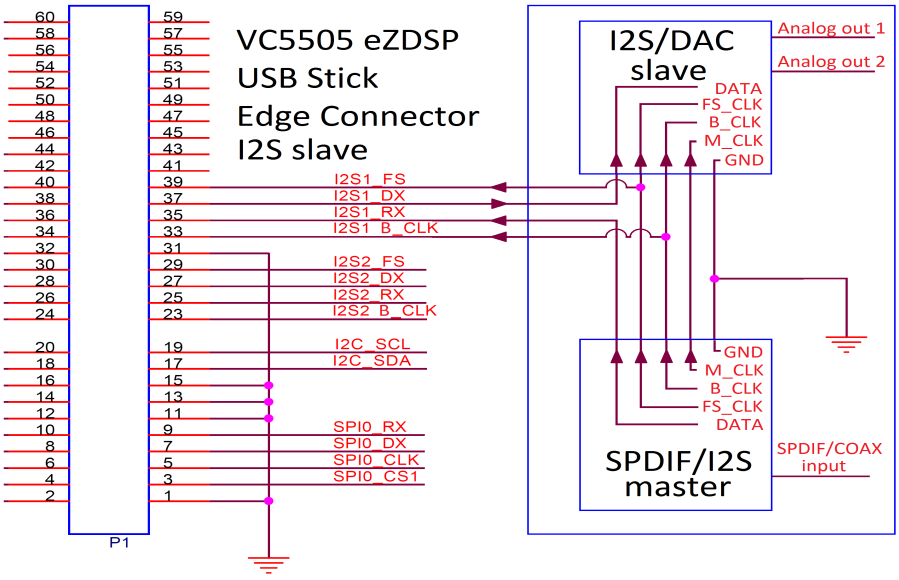
Kahendfaas-kodeerimisel toimub alati kodeeritud andmekanalis siire (tõusev või langev front) vastavalt sünkrosignaali ajastusele (vastavalt B_CLK perioodi algusele) olenemata sellest, kas edastatav DATA-bitt sisaldab väärtust "1" või "0". Kui DATA-bitt sisaldab väärtust "1", edastatakse see info lisasiirdega (tõusev või langev front) B_CLK perioodi keskel (B_CLK poolperioodi siirde ajal). Kui DATA bitt on väärtusega "0", siis B_CLK perioodi keskel lisasiiret ei tehta. Teisisõnu, kui DATA-bitt on väärtusega "0", tehakse BMC signaalis üks siire B_CLK perioodi jooksul ja kui DATA-bitt on väärtusega "1", tehakse BMC-signaalis kaks siiret B_CLK perioodi jooksul. Näide BMC kodeerimisest on toodud joonisel 18, kus B_CLK perioodi algused on tähistatud täisarvudega 0 kuni 12. SPDIF-vastuvõtja dekodeerib BMC-signaali tagasi kaheks

erinevaks signaaliks: B_CLK-ks ja DATA-ks. BMC meetod suurendab küll edastatava signaali spektri laiust, aga vähendab signaali alaliskomponenti, mis annab kokkuvõttes paremad eeldused signaali transportimiseks võrreldes I2S-ga.



Joonis 18. SPDIF ja signaali kodeerimine kasutades BMC meetodit.

SPDIF-väljundliides on laialt levinud laiatarbelistes heliseadmetes ja meediakeskustes. Nagu eelnevalt mainitud, on tüüpiliselt füüsiliseks meediaks vask (RCA või BNC tüüpi pistikühendusega koaksiaalkaabel) või optika (1mm plastoptika Toslink tüüpi pistikühendusega). Et SPDIF-liidesega edastatud digitaalne helisignaal oleks vastuvõetav C5505 DSP-l, tuleb antud signaal konverteerida tagasi I2S kujule, sest C5505 DSP-l puudub SPDIF-sisend. Joonisel 19 on illustreeritud antud töös kasutatavat lahendust.



Joonis 19. SPDIF-vatuvõtja ja DAC sidumine TMS320C5505 USB Stick arendusplaadiga. [18]

Joonisel 19 on kujutatud "TMS320C5505 eZdsp USB Stick" arendusplaadi äärel asuvat laienduspistikut (P1) ja sellel asuvat täisdupleks I2S1 porti [18]. Helisignaal jõuab SPDIF-liidese kaudu SPDIF/I2S-konverterisse (joonisel 19 kujutatud all paremas nurgas). Konverter dekodeerib SPDIF-signaali ja genereerib vajalikud I2S DATA ja sünkrosignaalid (M_CLK, B_CLK ja FS_CLK). Antud süsteemis on SPDIF/I2S-konverter I2S *master*-seade ja järelikult peab DSP olema I2S *slave*-seade. Antud pilootprojektis on ka DAC I2S *slave*-seade, mis võimaldab läbi saada vaid ühe DSP I2S pordiga (tänu eraldi DX ja RX kanalitele DSP arendusplaadil), kuigi see ei pea

tingimata nii olema. Sisuliselt on võimalik ka stsenaarium, kus DSP on DAC suhtes *master*, aga SPDIF-konverteri suhtes *slave*. Sellisel juhul tuleb muidugi kasutusele võtta kaks eraldi I2S-liidest (nt I2S1 ja I2S2).

Asjaolu, et SPDIF-vastuvõtja on I2S *master*-seade, on mugav eelkõige sellepärast, et DSP ei pea otseselt teadma, mis võendamissagedusega ja resolutsiooniga sisendsignaali edastatakse. Näiteks kui sisendsignaali võendamissagedus on 44.1kHz, genereerib SPDIF-vastuvõtja B_CLK-signaali sagedusega 44.1kHz*64=2.8224MHz. Kui aga sisendsignaali võendamissagedus on näiteks 192kHz, genereerib SPDIF-vastuvõtja B_CLK-signaali sagedusega 192kHz*64=12.288MHz. DSP sisemine taktsagedus (SYS_CLK=100MHz) on märgatavalt suurem maksimaalsest võimalikust B_CLK-st (12.288MHz), mida antud töös kasutatav vastuvõtja on suuteline edastama.

Suurim puudus antud lahendusel on asjaolu, et SPDIF-vastuvõtja poolt genereeritud sünkrosignaale (eriti B_CLK ja M_CLK) ei ole lihtne kvaliteetselt genereerida. Nagu eelpool mainitud, on SPDIF-vastuvõtja poolt genereeritud sünkrosignaalide perioodis pahatihti suur hälve (*jitter*). Joonisel 19 on DAC I2S *slave*-seade, mis tähendab, et analoog-helisignaal genereeritakse SPDIF-vastuvõtja poolt genereeritud sünkrosignaalide järgi. Kui kasutatavad sünkrosignaalid on suure *jitter'*iga, siis tähendab see moonutust DAC poolt väljastatud analoog-helisignaalis. Teisisõnu, analoog-helisignaali kvaliteet sõtlub peamiselt SPDIF-vastuvõtja kvaliteedist. Näiteks kõrgekvaliteetse DAC kasutamisel sellises süsteemis ei ole ilmselt suuremat sisu, kui SPDIF-vastuvõtja on madalakvaliteediline.

Antud pilootprojektis on joonisel 19 kujutatud SPDIF-vastuvõtjana kasutatud Cirrus Logic Inc. poolt toodetavat mikroskeemi CS8416. CS8146 mikroskeem on disainitud SPDIF digitaalse helisignaali vastuvõtmiseks ja dekodeerimiseks I2S-protokollile ning tuleb toime helisinglaaliga, mille võendite resolutsioon on kuni 24 bitti ja võendamissagedus kuni 192kHz. Genereeritud I2S M_CLK-sünkrosignaali perioodi tüüpiline hälve (*jitter*) on 200ps. Hälve võib olla ka suurem sõltudes peamiselt CS8416 toitepinge (nominaalne on +3.3V alalispinge) kvaliteedist – elektriline müra toiteliinil suurendab hälvet (*jitter'*it) sünkrosignaalide perioodis [19]. Olenevalt rakendusest võib perioodi hälve 200ps olla asjaarmastajatele liiga suur. Spetsiaalse sünkrosignaali-generaatori kasutamine digitaalsetes helisüsteemides võimaldab saavutada oluliselt paremaid tulemusi *jitter'*i osas (kümneid kuni sadu kordi väiksem *jitter*). Käesolevas pilootprojektis ei ole *jitter'*i minimiseerimise peale täiendavat ressurssi kulutatud.

Joonisel 19 kujutatud DAC-na on kasutatud Cirrus Logic Inc. mikroskeemi C4344 [20]. C4344 mikroskeem on disainitud I2S-formaadis esitatud digitaalse helisignaali muundamiseks analoog-helisignaaliks. Helisignaali sisendvõendite maksimaalseks resolutsiooniks on 24 bitti ja maksimalseks võendamissageduseks on 192kHz. C4344 DAC põhineb delta-sigma modulaatoril (neljandat järku *multi-bit delta-sigma* modulaator [20]) ja sisaldab endas ka integreeritud madalpääsfiltrit *alias'*te vähendamiseks analoogsignaalis (DAC väljundis). *Alias* tuleneb asjaolust, et digitaalsete signaalide sagedusspekter on perioodiline (perioodiga 2π [rad] normaliseerituna võendamissagedusele).

Cirrus Logic mikroskeemide kasutamine antud pilootprojektis ei ole seotud sellega, et nad oleksid teistest märgatavalt paremad. Põhjus seisneb hoopis selles, et sellise komplekti (CS8146 + C4344) rakendamine on küllaltki hästi dokumenteeritud Cirrus Logic poolt (“C4344 Evaluation Board” [21]). Ilmselt on tänu sellele toodetud ka palju vastavaid valmislahendusi. Näiteks Muse Audio toodab DAC-d, mis põhineb samal komplektil – CS8146 ja C4344. Skeemitehniline lahendus on peaaegu sama, mis

Cirrus Logic “C4344 Evaluation Board” [21] dokumentatsioonis. Sellise valmislahenduse hind koos korpusega on umbes 20USD (ehk ligi 15 EUR). Piiratud ressursside olukorras on antud pilootprojektis kasutatud just kirjeldatud Muse Audio SPDIF-sisendiga DAC-d. Joonisel 19 on CS8146 ja C4344 ümbritsetud sinise raamiga, mis tähistab asjaolu, et tegemist on samal trükkplaadil olevate mikroskeemidega.

Erinevalt Muse Audio originaalsest kontseptsioonist ei ole antud projektis CS4816 mikroskeemi I2S-väljundi DATA-liin ühendatud otse C4344 (DAC) mikroskeemi külge. DATA, mis tuleb CS8416 mikroskeemist (SPDIF/I2S-konverter) saadetakse hoopis DSP I2S-sisendisse RX vastavalt joonisele 19. DSP töötleb vastuvõetud sisendsignaali (FIR-filtreerimine vastavalt valitud filtrikoefitsentidele) ja saadab juba töödeldud signaali C4344 mikroskeemi sisendisse. Seega tuleb Muse Audio DAC plaadile teha vastav muutus skeemis. Lihtsaim meetod on eemaldada DATA-liinis olev takisti ja joota sisend/väljund vastava raja tekkinud otstele. Lisaks sellele on antud projektis rakendatud veel üks põhimõtteline erinevus originaalkontseptsioonist – kui originaalis on Muse Audio DAC kaks väljundit planeeritud stereo helisignaali esitamiseks, siis käesolevas projektis on need väljundid kasutatud sama kanali (parem või vasak vastavalt programmkoodile) madal- ja kõrgpääsfiltri väljundi esitamiseks. Seda olukorda illustreerib joonis 1.b. Stereo lahenduse saavutamiseks on vaja rakendada kaks DSP-d ja kaks DAC-d. Ühe komplekti DSP kasutab tarkvaraliselt vasakut kanalit, teise komplekti DSP vastavalt paremat kanalit stereo signaalist. Vasaku ja parema kõlari sisendsignaal (SPDIF koaksiaal- või optiline Toslink-liides) on identne. Antud projektis on planeeritud kasutada optilist meediat (Toslink), mille paljundamiseks on müügil vastavad jagajad (ingl. k *splitter*). Jagaja hind jääb paari euro suurusjärku.

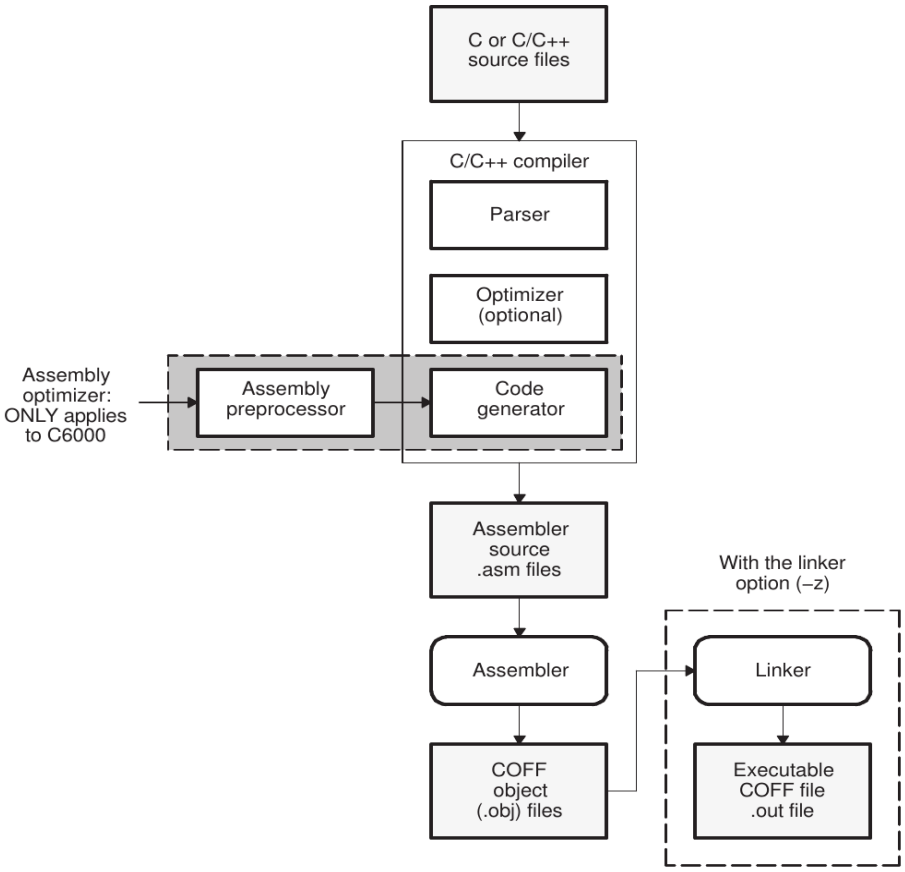
4. Tarkvaraline lahendus

Peatükk kirjeldab projekti realiseerimiseks kasutatud "TMS320C5505 eZdsp USB Stick" arendusplaadi tööks vajalikku tarkvaralist lahendust. Tehakse lühike ülevaade magistritöös kasutatavat TI programmeerimis- ja analüüsikeskkonnast Code Composer Studio (CCS), samuti antud arendusplaadile suunitletud näitekoodist FIR-filtrite realiseerimiseks ("C5505 EZDSP Audio Filter Demo"). Järgnevalt on kirjeldatud reaalset tarkvaralist lahendust detailsemalt (modifitseeritud versioon eelmainitud näitekoodist). Eesmärgiks ei ole kirjeldada igat programmkoodi rida. Pigem on vaadeldud tarkvaralist lahendust üldisemalt, tuues välja erinevate funktsioonide ja katkestus-rutiinide (ingl. k ISR - *Interrupt Service Routine*) vahelisi olulisi seoseid. Detailne kommenteeritud programmkood (projekt) on esitatud koos käesoleva tööga digitaalsel kujul.

4.1 TI Code Composer Studio

Texas Instruments Inc. poolt toodetud riistvara tarkvaralise lahenduse arenduskeskkonnaks on Code Composer Studio ehk CCS või CCStudio. CCS sisaldab tarkvara lähtekoodi kirjutamiseks, selle kompileerimiseks ja programmkoodi analüüsimiseks ning silumiseks (ingl. k *debug*) vajalikke tööriistu. Erineva seeria protsessorite jaoks sisaldab CCS ka vastavaid kompilaatoreid. Viimased CCS versioonid (v5 ja v6) baseeruvad vabavaralisel "Eclipse" raamistikul ja toetavad muuseas ka GNU Linux operatsioonisüsteeme. Uuemate CCS versioonide kasutamiseks on tarvis litsentsi. Arendusplaatide, nagu käesolev "eZdsp", kasutamiseks piisab TI kodulehel registreerumisest, peale mida on võimalik alla laadida tasuta litsents. Teatud CCS funktsionaalsus on tasuta litsentsiga piiratud, kuid antud projekti tarkvaralise lahenduse väljatöötamiseks on võimaldatud funktsionaalsus igati piisav.

Joonis 20 kirjeldab CCS-keskkonnas arendatava programm-lähtekoodi teekonda masinkoodini – COFF (ingl. k *Common Object File Format*) failini. Kompilaatori eesmärk on tõlkida standardses C/C++ programmeerimiskeeles kirjutatud programmkood assemblerkeelde (madaltaseme programmeerimiskeel). Oluline aspekt sellise tõlkimise juures on optimeerimine protsessori efektiivsuse tõstmiseks. Optimeerimise efektiivsus sõltub muuhulgas ka C/C++ programmkoodist endast ehk kokkuvõttes programmeerija pädevusest. Assembleri eesmärgiks on tõlkida assembler-lähtekoodid masinkoodi, mis antud DSP kontekstis on COFF-formaadis. Genereeritud masinkoodi COFF-objektifailid kombineeritakse üheks käivitatavaks objektifailide mooduliks (".out" laiendiga fail) nn "Linker"-i poolt. Linker määrab muuhulgas ka selle, kuidas paigutatakse programmkood protsessori mällu. Lõpuks salvestatakse genereeritud COFF (".out") fail DSP mällu ja kood käivitatakse. Täpsem info CCS-i ja kirjeldatud protsessi kohta on saadaval TI poolt avaldatud kirjanduses, näiteks dokumendis SPRU509H. [22]



Joonis 20. Programmkoodi genereerimine CCS keskkonnas. [22]

Kui programmkoodi arendustegevus on lõpetatud, saab lõplikust COFF-failist genereerida käivitatava meediafaili (ingl. k *boot image*), mille salvestamisel arendusplaadi teatud mälupiirkonda on võimalik DSP-l käivituda iseseisvalt (ilma CCS-ta). Antud riistvara korral tuleb "*boot image*" genereerimiseks konverteerida COFF-fail binaarkujule kasutades niinimetatud "hex55" utiliiti. Vastav utiliit on leitav operatsioonisüsteemile installeeritud CCS vastavast kompilaatori kataloogist, näiteks GNU Linux vaikimisi installeerimise korral järgmisest asukohast:

/opt/ti/ccsv5/tools/compiler/c5500_4.4.1/bin/hex55

Klassikaline süntaks "hex55" utiliidi kasutamisel C5505 protsessorile sobiva "*boot image*" loomiseks on järgmine:

hex55 -i filename.out -o boot_image_filename.bin -boot -v5505 -b -serial8

Utiliidi väljundiks on binaarfail, mis on võimalik laadida DSP-le sobivasse mälupiirkonda CCS vahendusel. Kõige mugavam lahendus selle realiseerimiseks on C5505 arendusplaadile laadida spetsiaalne COFF-fail: "programmer_USBKey.out", mis on alla laetav "TMS320C5505 eZdsp USB Stick" arendusplaadi näitekoodidele suunitletud veebilehelt [23]. Pärast mainitud COFF-faili käivitamist DSP-l oodatakse kasutajalt "hex55" utiliidi poolt genereeritud binaarfaili asukoha (ingl. k *path*) sisestamist konsooliaknasse, peale mida alustatakse binaarfaili paigutamist DSP mällu. Asukoha määramisel ei ole lubatud kasutada tühikuid. Järgneval DSP pingestamisel otsitakse käivitatavat meediat ("*boot image*") järgmistest asukohtadest (toodud järjekorras):

- NOR Flash
- NAND Flash
- 16-bit SPI EEPROM
- I2C EEPROM
- MMC/SD.

Kui esimene käivitatav meedia on DSP poolt leitud, laetakse see RAM-mällu ja alustatakse programmkoodi täitmist. CCS ja "Programmer_USBKey.out" rakenduse vahendusel laetakse binaarfail SPI EEPROM mällu, mis on C5505 suhtes väline mälu suurusega 64KB ja on kujutatud ka joonisel 15. Täpsem informatsioon seoses C5505 "*bootloader"*-iga on toodud TI dokumendis SPRAB92 [24].

4.2 "eZdsp Audio Filter Demo" näitekood ja selle modifitseerimisvajadus

TI on avaldanud hea näite FIR-filtrite realiseerimiseks C5000 seeria protsessoritel. Näitekood on alla laetav veebist [23]. Antud näitekoodis rakendatakse arendusplaadil asuvat audiokoodekit TLV320AIC3204 (kujutatud joonisel 15), mis sisaldab nii analoog-digitaal muundit (ADC) kui ka digitaal-analoog muundit (DAC). Sisendsignaaliks on näitekoodis analoog-helisignaal, mis võendatakse TLV320AIC3204 koodeki ADC abil võendamissagedusega 48kHz. Väljundsignaaliks on samuti analoog-helisignaal, mis saadakse peale helisignaali digitaalset töötlemist (FIR-filtreerimine), kusjuures digitaal-analoog muundamine realiseeritakse TLV320AIC3204 koodeki DAC abil. Näitekoodi sisu ja tööpõhimõtte võib kokku võtta järgmiste etappidena:

- ADC muundab siseneva analoog-helisignaali digitaalseks ja esitab selle I2S protokollis
- I2S0 kontroller kogub võendeid ADC-lt ja salvestab need puhvritesse perifeerseadme mälus
- I2S0 kontroller edastab puhvri täitumisel DMA0-kontrollerile vastava sündmuse ("*event*")
- DMA0-kontroller transpordib sisendvõendite puhvri CPU-le sobilikku mälupiirkonda
- CPU teostab sisendvõenditega FIR-filtreerimise algoritmi ja väljastab uue (väljund-) puhvri. Kasutatakse 50 punktist kausaalset Hanning-aknameetodiga genereeritud impulsskaja, mis on oma olemuselt madalpääsfilter lõikesagedusega 2kHz. Signaalitöötluse algoritm seisneb konvolutsiooni rakendamises.
- Võendite väljastamisel teavitab I2S0-kontroller DMA0-kontrollerit puhvri tühjenemisest I2S0-perifeeria mälus genereerides vastava sündmuse ("*event*")
- Vastava sündmuse ilmnemisel I2S0-kontrolleri poolt transpordib DMA0-kontroller filtreerimise tulemusena saadud järgmise väljundpuhvri I2S0 mällu.
- I2S0-kontroller väljastab CPU poolt töödeldud signaali võendid DAC-le kasutades I2S protokolli.
- DAC muundab vastuvõetud digitaalse signaali analoog-helisignaaliks.

Tegemit on kasuliku näitekoodiga, mis sisaldab käesoleva projekti realiseerimiseks võtmelahendusi DMA ja signalitöötluse osas FIR-filtrite rakendamisel.

Samas tuleb näitekoodi nii mõneski osas muuta, et seda kohandada käesoleva projekti jaoks sobivaks. Järgnevalt on toodud olulisemad muudatused, mis vajavad tähelepanu:

- Ülesandepüstituse kohaselt peab helisignaal jõudma süsteemini digitalsel kujul, et vältida liigseid analoog-digitaal ja digitaal-analoog muundamisi ning vähendada elektromagnetiliste häirete mõju helisignaalile. Seetõttu ei ole peamiseks eesmärgiks TLV320AIC3204 koodeki ADC kasutamine. I2S0 asemel on käesolevas projektis kasutusel I2S1-kontroller, mille sisendsignaal pärineb SPDIF-formaadis helisignaali muundurist (joonis 19). I2S1-kontrolleri kanalid tuleb juhtida laienduspistikule seadistades vastav DSP register – EBSR.
- Vastavalt joonisele 19 on DSP antud lahenduses I2S *slave*-seade erinevalt näitekoodist, kus DSP on TLV320AIC3204 koodeki suhtes I2S *master*-seade. Lisaks I2S *master/slave* seadistuse erinevusele tuleb I2S0-kontrolleri asemel kasutusele võtta I2S1-kontroller. See omakorda tingib DMA0 asemel DMA3-kontrolleri kasutamise, sest erinevate I2S-kontrollerite mälule on ligipääs erinevatel DMA-kontrolleritel. Samuti tuleb ümber seadistada DMA3 käivitumaks I2S1-kontrolleri sündmuste peale (mitte I2S0 sündmuste peale).
- Vastavalt joonisele 1a on planeeritavas stereo-helisüsteemis kaks DSP-d – üks kummaski kõlarikastis (vasak ja parem kõlarikast). Mõlema DSP FIR-filtrite impulsskajad on valitud selliselt, et eraldatud sagedusribad on komplementaarsed. Teisisõnu on ühe DSP väljundiks stereosignaali ühe kanali (vasak või parem) madal- ja kõrgpääsfiltri sagedusribad. See erineb "eZdsp Audio Filter Demo" näitekoodi teostusest, kus ühe DSP väljundiks on stereosignaal, mille mõlemad kanalid (vasak ja parem) on läbinud madalpääsfiltri. Seega tuleb ümber korraldada näitekoodi FIR-algoritmi rakendamise ülesehitus ja lisada programmkoodi ka kõrgpääsfiltri koefitsentide massiivid. Ilmselt tuleb vasaku- ja parema kõlarikasti DSP-le kompileerida erinev tarkvara – üks filtreerib stereosignaali paremat kanalit, teine vasakut kanalit.
- Vastavalt ülesandepüstitusele peab arendatav platvorm toetama digitaalset helisignaali võendamissagedustega 44.1kHz, 48kHz, 88.2kHz, 96kHz ja 192kHz. Sõltumata digitaalse helisignaali võendamissagedusest peab murdesagedus analoog-domeenis olema ilmselt fikseeritud (näiteks sagedusele 2kHz). Samas, digitaalses domeenis sõltub helisignaali murdesagedus (filtri impulsskajas) võendamissagedusest. Seega tuleb iga võendamissageduse jaoks tekitada eraldi impulsskajade massiivid, et saavutada iga võendamissageduse korral üks ja seesama analoog-domeeni murdesagedus. DSP peab olema võimeline tuvastama sisendsignaali võendamissageduse ja valima vastavalt sellele sobivad impulsskajade massiivid (kõrg- ja madalpääsfiltri impulsskajad).
- Näitekoodis kasutatakse madalpääsfiltri impulsskaja, mis koosneb 48-st filtrikoefitsendist ja on leitud Hanning-aknameetodil. Ideaalilähedasema sageduskarakteristiku saavutamiseks on käesolevas töös suurendatud filtrite sügavust 101-le koefitsendile. See tähendab, et näitekoodi analoogia põhjal on mõistlik vastavalt muuta ka sisend- ja väljundpuhvrite suurusi.

Nimetatud muudatuste sisseviimiseks tuleb muuta palju (praktiliselt kõiki) olulisi näitekoodi funktsioone. Peamisteks eranditeks on InitSystem() ja fir() funktsioonid, mis on käesoleva töö kirjutamise ajal originaalsest näitekoodist jäänud muutumatuks.

4.3 Arendusplaadi tarkvaraline initsialiseerimine

Peatükis on kirjeldatud "TMS320C5505 eZdsp USB Stick" arendusplaadi tarkvaralises initsialiseerimises kasutatud funktsioone, mille eesmärk on muuseas seadistada platvormi sisemine kell, laienduspistiku väljundkanalid, I2S- ja DMA-kontrollerid ning ette valmistada katkestusrutiini (ISR) kasutamine. Järgnevalt on toodud loetelu vajalikest sammudest DSP algseadistamisel, mis ühtib teatavas osas TI vastavas dokumentatsioonis toodud loeteluga (SPRUFP4 - *Steps for I2S Configuration and I2S Interrupt Service Routine*). Initsialiseerimiseks käivitatakse main() rutiinis järgmised funktsioonid:

1. InitSystem()

Funktsiooni peamine eesmärk on seadistada DSP sisemine kell taktsagedusele 100MHz rakendades sisemist PLL-i (ingl. k *Phase-Locked Loop*). DSP sisemise kella seadistamiseks väärtustatakse neli registrit - PLL_CNTL1 kuni PLL_CNTL4 (*PLL Control Register Bit Fields*). TI dokumentatsioonis on vastavate registrite nimetuseks CGCR1...CGCR4 (*Clock Generator Control Register 1...4*). Vastavate registrite seadistamiseks võib kasutada TI veebilehelt allalaetavat "C5505 PLL kalkulaatorit" [25], detailsem info on toodud TI dokumentatsioonis [9]. Nimetatud funktsioonis seadistatakse DSP sisemine kell parameetri PLL_100M abil, mis on väärtustatud konstandiga 1 päisefailis nimega control.h. Lisaks taaskäivitatakse InitSystem() funktsioonis perifeerseadmete poolt kasutatav kell kasutades PCGCR registrit (*Peripheral Clock Gating Configuration Register*). Viimaks taaskäivitatakse I2S- ja DMA-kontrollerid kasutades PER_RSTCOUNT ja PER_RESET registreid, mis TI dokumentatsioonis kannavad vastavalt PSRCR (*Peripheral Software Reset Counter Register*) ja PRCR (*Peripheral Reset Control Register*) nimesid.[9] InitSystem() funktsioonis kasutatavad registrid on defineeritud päisefailis register_system.h.

2. ConfigPort()

Funktsioon määrab arendusplaadi laienduspistikule multipleksitavad perifeerseadmete väljaviigud seadistades PERIPHSEL0 registrit, mis on defineeritud päisefailis lpva200.inc. TI dokumentatsioonis on vastava registri nimetus EBSR (*External Bus Selection Register*). Antud töös on laienduspistikule juhitud I2S0-, I2S1- ja I2S2-perifeerseadmete väljaviigud. Sellise valikuga kaasneb ka mitme sisend-väljundviigu (*GPIO*) ja mõne teise perifeerseadme väljaviikude multipleksimine laienduspistikule, mida käesoleval hetkel tarkvaras ei rakendata. Täpsem informatsioon EBSR-seadistuse kohta on leitav TI dokumentatsioonist [9].

3. SYS_GlobalIntEnable()

Nimetatud funktsioon määrab DSP initsialiseerimisel kasutatavate instruktsioonide ja katkestusrutiinide printsiibid. Programmkoodis määratakse globaalse katkestuse lubamiseks ST1-registri INTM-bitt väärtusele "0", vastav register on defineeritud päisefailis lpva200.inc. TI dokumentatsioonis nimetatakse vastavat registrit ST1_55 (*Status Register 1*), täpsem info on leitav TI dokumendist SPRU317K [26].

4. reset_RTC(); enable_rtc_second_int()

Esimeses funktsioonis võetakse RTC katkestusrutiinid tööst välja seadistades registris RTC_CTR noorima biti väärtuseks "0". Vastav register kannab TI dokumentatsioonis nime RTCINTEN (*RTC Interrupt Enable Register*). Järgnevalt algväärtustatakse RTC kella-aeg (millisekunditest kuni aastani). Seejärel taastatakse üldine katkestusrutiinide

režiim RTC_CTR registri vahendusel algsele väärtusele.
Teises funktsioonis võimaldatakse üldine RTC-katkestuste genereerimine RTC_CTR registri vahendusel ja samuti võimaldatakse konkreetsemalt sekundi-katkestusrutiini töö registri RTC_INT vahendusel. TI dokumentatsioonis kannab vastav register nime RTCINTREG (*RTC Interrupt Register*).[27]

5. setDMA_address()

DMA lähte- ja sihtpunktide adresseerimisel tuleb arvestada asjaoluga, et DMA kasutab adresseerimiseks baidi aadresse, aga CPU kasutab "sõna" aadresse (ingl. k *word address*) ehk 16-bitise andmeühiku aadresse. Seega on vajalik tarkvaraline aadresside konverteerimine [12]. Programmkoodis tehakse see operatsioon setDMA_address() funtsioonis. Registri aadresside konversioon DMA-le sobiva baidi aadressi kujule koosneb kahest sammust:

- Sobiva registri "sõna" aadress nihutatakse 1 biti võrra vasakule, moodustades 32-bitine baidi aadress.
- Eelmises punktis tekkinud aadressile lisatakse sobiv nihe. Antud näites tuleb lisada baidi-aadressile väärtus 01 0000h (seoses DARAM mälu kasutamisega).

Programmkoodis on teostatud näiteks RcvL1 puhvri aadressi konversioon DMA-le sobivale kujule järgmiselt:

```
RxL1_DMA_address = (Uint32)RcvL1;
RxL1_DMA_address = (RxL1_DMA_address<<1) + 0x10000;
```

Seega, kui antud peatükis ja lisas 9 toodud joonistel on DMA-ga seotud puhvreid nimetatud RcvL1, RcvL2, RcvR1, RcvR2, FilterOutL1, FilterOutL2, FilterOutR1 ja FilterOutR2 puhvriteks, siis reaalselt on DMA seadistamisel kasutatud samade puhvrite baidi-kujule teisendatud algusaadresse. Näiteks RcvL1 jaoks on kasutatud aadressi nimega RxL1_DMA_address.

6. set_i2s1_slave(); enable_i2s1()

I2S1-kontrolleri seadistamiseks on programmkoodis kasutusel funktsioon set_i2s1_slave(). Funktsiooni peamiseks eesmärgiks on registri I2SSCTRL (*I2S1 Serializer Control Register*) väärtustamine, mis teostatakse läbi eeldefineeritud funktsiooni i2s1_write_CR(). I2SSCTRL abil määratakse põhilised I2S-sätted, näiteks *master/slave*- ja mono/stereo režiimid, samuti kaadri-sünkrosignaali FS ja biti-sünkrosignaali B_CLK polaarsus (vastavalt parameetrid FSPOL ja CLKPOL), andmete hilistamise viis DATA-liinil (ühe- või kahe bitine nihe), vastuvõetava võendi pikkus jmt. Antud töös on I2S1-kontrollerit kasutatud I2S *slave*-seadmena, mis töötab koostöös SPDIF-konverteriga, mis omakorda kasutab üldlevinud I2S-seadistust. Seetõttu piisab I2SSCTRL registris sisendvõendi pikkuse määramisest vastaval väljal, ülejäänud parameetrid jäävad antud juhul väärtusele 0 (välja arvatud vanim bitt, mis lubab kontrolleri töö). Täpne kirjeldus I2SSCTRL registrist on toodud TI dokumendis SPRUFP4 tabelis 17[15]. I2S1-kontrolleri töö lubamine toimub läbi funktsiooni enable_i2s1(), mis on defineeritud assemblerkoodis nimega i2s_register.asm. Funktsiooni sisuks on määrata aktiivseks vanim bitt registris I2SSCTRL.

7. enable_dma_int(void)

DMA katkestuste genereerimise luba määratakse enable_dma_int(void) funktsioonis. Registri DMA_MSK sisu määrab, millised DMA-kontrollerid ja kanalid tekitavad katkestuse DSP-le. Vastava registri nimetus TI dokumentatsioonides on DMAIER (*DMA Interrupt Enable Register*). Täpsem info registri struktuuri kohta on leitav TI dokumendist SPRUFP0C [9]. Samas funktsioonis algväärtustatakse DMA

katkestusrutiini väljakutsumist indikeeriv register, mis on defineeritud lähtekoodis nimega dma_bypass1.c. Algväärtustamine toimub DMA_IFR registris seades kõik bitid väärtusele "1". DMA_IFR registri nimetus TI vastavas dokumentatsioonis on DMAIFR [9].

8.

- set_dma3_ch0_i2s1_Lout(void)
- set_dma3_ch1_i2s1_Rout(void)
- set_dma3_ch2_i2s1_Lin(void)
- set_dma3_ch3_i2s1_Rin(void)

Antud töös on kasutusel DMA3-kontrolleri kõik kanalid, mille algseadistamine toimub eelnevalt toodud neljas funktsioonis, mis omakorda on defineeritud lähtekoodis nimega dma_bypass1.c. Esimesed kaks kanalit on seotud võendite väljastamisega I2S1-kontrollerile, järmised kaks võendite vastuvõtmisega I2S1-kontrollerilt. Seetõttu on nende funktsioonide sisu mõnevõrra erinev. Seadistatavad registrid on siiski samad, mistõttu on järgnevalt kirjeldatud kõiki nelja funktsiooni ühiselt, kusjuures *m* tähistab DMA kanali numbrit.

- DMA3_Ch*m*_TC_LSW registris määratakse DMA poolt transporditava puhvri suurus. TI dokumentatsioonis on vastva registri nimeks DMACH*m*TCR1 (*Transfer Control Register 1*) [12]. Puhvri suurus on seotud konstantse parameeteriga XMIT_BUFF_SIZE, mis on defineeritud lähtekoodi päisefailis nimega ref_data_bypass.h.
- DMA3_Ch*m*_TC_MSW registris määratakse DMA töörežiim, muuhulgas ka käivitumise-, lähte- ja sihtregistri adresseerimise- ning transporditavate üksuste suuruste printsiibid. TI dokumentatsioonis on vastva registri nimeks DMACH*m*TCR2 (*Transfer Control Register 2*) [12].
- DMA3_CH10_EVENT_SRC ja DMA3_CH32_EVENT_SRC registrites määratakse DMA vastava kanali käivitumist põhjustav sündmus, mis antud töös pärineb I2S1-kontrollerist. TI dokumentatsioonis on vastvate registrite nimedeks DMA3CESR1 ja DMAnCESR2 (*DMA3 Channel Event Source Register*). Need registrid sisaldavad CH*m*EVT väljasid, mis on täpsemalt kirjeldatud TI dokumendis SPRUFP0C [9].
- DMA3_Ch*m*_SRC_LSW ja DMA3_Ch*m*_SRC_MSW registrites määratakse DMA vastava kanali lähtepuhvri esimese registri aadress. TI dokumentatsioonis on vastvate registrite nimedeks DMACH*m*SSAL ja DMACH*m*SSAU (vastavalt *Source Start Address Lower Part* ja *Source Start Address Upper Part Register*) [12]. Lähtepuhvriks võib olla I2S1-perifeerseadme mälus olev sisendvõendite puhver või fir() funktsiooni väljundpuhver olenevalt DMA3 kanalist ja selle ülesandest. DMA lähteregistrite adresseerimine on detailsemalt kirjeldatud järgmises peatükis (pt 4.4).
- DMA3_Ch*m*_DST_LSW ja DMA3_Ch*m*_DST_MSW registrites määratakse DMA vastava kanali sihtpuhvri esimese registri aadress. TI dokumentatsioonis on vastvate registrite nimedeks DMACH*m*DSAL ja DMACH*m*DSAU (vastavalt *Destination Start Address Lower Part* ja *Destination Start Address Upper Part Register*) [12]. Sihtpuhvriks võib olla fir() funktsiooni sisendpuhver või I2S1-perifeerseadme mälus asuv

väljundvõendite puhver olenevalt DMA3 kanalist ja selle ülesandest. DMA sihtregistrite adresseermine on detailsemalt kirjeldatud järgmises peatükis.

4.4 Ülevaade tarkvaralisest andmevahetusprotsessist

Selles peatükis on detailsemalt iseloomustatud tarkvaralise lahenduse andmevahetusprotsesside olemust. Eesmärgiks on kirjeldada I2S-formaadis vastuvõetud sisendvõendi teekonda DSP poolt teostatava filtreerimisalgoritmini (fir() funktsioon) ja tulemuseks olevate võendite teekonda väljundvõenditeni I2S-formaadis. Protsesside graafiline esitus on toodud lisas 9, mis sisaldab kahte joonist: esimene kujutab main() funktsiooni tööd (fir() funktsiooni rakendamine) ja teine I2S-i, DMA ja CPU vaheliste protsesside seoseid (I2S sündmused, andmete puhverdamine ja DMA katkestusrutiin). Illustratsioonid meenutavad UML (ingl. k *Unified Modeling Language*) keelt, kuid ei ole mõeldud vastama kõikidele UML-spetsifikatsioonidele. Jooniste eesmärk on programmkoodis defineeritud protsesside vaheliste seoste kirjeldamine ja nende visualiseerimine näidates ära ka muutujate ja puhvrite jaoks kasutatud nimetused. Joonised lisas 9 hõlbustavad antud peatüki ja programmkoodi lugemist ning mõistmist.

Programmkoodis kasutatakse I2S1-kontrollerist pärinevate võendite vastuvõtmiseks DMA-d, mille kanalite sätted on kirjeldatud set_dma3_ch2_i2s1_Lin() ja set_dma3_ch3_i2s1_Rin() funktsioonides. DMA ülesandeks on transportida sisendvõendid I2S1-kontrolleri puhvritest vastavalt RcvL ja RcvR puhvritesse. Vastavate DMA kanalite lähteregistriks on I2S1-kontrolleri "*Receive Left Data*" puhvri esimene register (DMA3_CH2_SRC_LSW=0x2928) vasaku stereosignaali kanali jaoks ja "*Receive Right Data*" puhvri esimene register (DMA3_CH3_SRC_LSW=0x292C) parema stereosignaali kanali jaoks. Vastavad registrite aadressid on toodud TI dokumentatsioonis [15] (SPRUFP4 *I2S1 Register Mapping Summary*). Nagu varem mainitud, alustab DMA andmete transporti teatud sündmuse ("*event*") ilmnemisel.

Järgnevalt on kirjeldatud I2S-porti siseneva võendi teekonda RcvL ja/või RcvR puhvrisse, mida kasutab fir() funktsioon. Andmed I2S1_RX liinil nihutatakse "*Recieve Shift Register*"-isse sagedusega, mille määrab I2S *master*-seadme poolt genereeritud B_CLK-signaal. I2S-kontrolleri ülesehitus on toodud lisas 8 [15]. Protsess algab pärast õiget fronti I2S_FS-liinil - tõusev või langev front vastavalt I2S_FS polaarsuse seadistusele DSP-s (CLKPOL-bitt I2SSCTRL registris), arvestades ka DATA-liinil olevate võendite 1-bitist nihet FS frondi suhtes. Kui vajalik kogus võendeid (stereo puhul kaks võendit) on kohal, kopeeritakse vastuvõetud võendid "*Receive Buffer Register*"-i vastavalt I2S funktsionaalplokile. Järgnevalt lahutatakse vasaku ja parema kanali andmed ja paigutatakse need I2S1 "*Receive Left/Right Data"* registritesse. Nii parema kui ka vasaku kanali puhul on DMA kontekstis tegemist 32-bitise registriga. Kui "*Receive Left Data*" või "*Receive Right Data*" puhver on täis, siis genereeritakse vastav "*event*" DMA-le, mille peale DMA alustab andmete transportimist I2S1 "*Receive Left/Right Data Register*" registrist enda FIFO puhvrisse ja seejärel RcvR ja RcvL registritesse.

Sarnaselt võendite vastuvõtmisele I2S-kontrollerist (I2S1_RX-liinilt) toimub ka andmete väljastamine I2S1-kontrollerist (I2S1_DX-liinile). Selguse huvides on ka see protsess järgnevalt kirjeldatud. DMA alustab I2S1-perifeerseadme registritesse väljundvõendite kirjutamist pärast seda, kui I2S-kontroller saadab DMA-le vastava sündmuse (I2S1 "T*ransmit Left/Right Data*" registrid vajavad täitmist). Sel puhul transpordib DMA andmed FilterOutL ja FilterOutR registritest I2S1 "*Transmit*

Left/Right Data" registrisse (vasak ja parem kanal eraldi registritesse). Järgnevalt transpordib I2S1-kontroller need võendid oma "*Transmit Shift*" registrisse läbi "*Transmit Buffer*" registri. I2S1-kontroller nihutab seejärel andmed I2S1_DX-liinile vastavalt I2S *master*-seadme poolt genereeritud sünkrosignaalidele ja I2S_FS polaarsuse seadistusele DSP-s (CLKPOL-bitt I2SSCTRL registris). Kui "*Transmit Left/Right Data*" puhver on kopeeritud "*Transmit Buffer*" registrisse, genereeritakse järgmine sündmus DMA-le ja tsükkel kordub. Muidugi peab DMA jõudma "*Transmit Left/Right Data*" registrid täita enne kui I2S *master* saadab uue I2S_FS frondi,
millega algab uue võendi nihutamine I2S1_DX liinile). Kirjeldatud arhitektuur lubab nihutada võendeid I2S1_DX-liinile samal ajal kui DMA täidab I2S1 "*Transmit Left/Right*" registrit uute võenditega.

Kirjeldatud väljundvõendite transportimiseks I2S1_DX-liinile tuleb seadistada täiendavad DMA kanalid. Programmkoodis teostatakse see set_dma3_ch0_i2s1_Lout() ja set_dma3_ch1_i2s1_Rout() funktsioonides, mille sisu on sarnane sisendvõendite transportimiseks kasutatud DMA kanali seadistamiseks kasutatavatele funktsioonidele. Jällegi määratakse vastavate DMA kanalite lähte- ja lõppregistrid. Lähtepunktiks on sedapuhku FilterOutL ja FilterOutR puhvrite esimese registri aadress, sihtpunktiks aga I2S1 kontrolleri "*Transmit Left Data*" puhvri esimese registri aadress (*DMA3_CH0_DST_LSW = 0x2908*) vasaku helisignaali kanali jaoks ja "*Transmit Right Data*" puhvri esimese registri aadress (*DMA3_CH1_DST_LSW = 0x290C*) parema helisignaali kanali jaoks. Vastavad registrite aadressid on toodud TI dokumentatsioonis [15] (SPRUFP4 *I2S1 Register Mapping Summary*).

I2S1-kontrolleri "*Transmit Left/Right Data*" ja "*Receive Left/Right Data*" registrite täitmine võenditega organiseeritakse vastavalt I2S1-kontrolleri parameetri "PACK" seadistusele. "PACK" parameeter (I2SSCTRL registris bitt 7) on mõistlik määrata aktiivseks, kui edastatavate helivõendite resolutsioon on kuni 16 bitti. Kuna DMA transpordib 32-bitiseid sõnu, oleks ebapraktiline transportida ühe tsükli jooksul vaid üks 16-bitine võend (ülejäänud 16 bitti oleksid ilmselt nullid vastavalt I2S spetsifikatsioonile). 16-bitise võendi resolutsiooni korral on DMA efektiivsus kaks korda suurem, kui ühe tsükli jooksul transporditakse I2S-kontrollerisse kaks 16-bitist võendit (vasak ja parem kanal korraga). Samuti oleks antud näite korral kaks korda efektiivsem vastava mälu kasutus (pole mõtet hoida mälus bitte, mis ei sisalda kasulikku infot). Seega "PACK" parameeter mõjutab seda, kuidas organiseeritakse võendid I2S- ja DMA registrite mälus ja kui tihti saadetakse DMA-le I2S puhvrite täitmise/tühjendamise vajaduse kohta signaale ("*event*"-e). Antud töös ei ole "PACK" funktsiooni kasutusele võetud, kuna programmkoodi edasisel arendamisel on sihiks ka 24- ja 32-bitiste võendite signaalitöötlus. Töö dokumenteerimise hetkel on programmkood vastava resolutsiooniga võendite töötlemiseks võimeline, kuid nagu varem mainitud, ei oma see hetkel erilist sisu, kuna fir() funktsioon eeldab kõigest 16-bitise resolutsiooniga filtrikoefitsente.

Nii "eZdsp Audio Filter Demo" näitekoodis kui ka antud projekti realiserimiseks loodud programmkoodis on kasutusel topelt puhverdamine – nn. "ping-pong" puhverdamine. Kuna nii DMA kui ka fir() funktsioon kasutavad RcvL, RcvR, FilterOutL ja FilterOutR puhvreid, võib tekkida olukord, kus fir() funktsioon kasutab vastavat mälupiirkonda samal ajal kui DMA. Probleemide vältimiseks on kõik mainitud puhvrid duubeldatud. Seega kasutavad fir() funktsioon ja DMA tegelikkuses kordamööda kokku kaheksat puhvrit: RcvL1, RcvL2, RcvR1, RcvR2, FilterOutL1, FilterOutL2, FilterOutR1 ja FilterOutR2. Näiteks DMA, saades sündmuse I2S vasaku

sisendpuhvri täitumisest, toimetab vastava puhvri perifeerseadme mälust RcvL1 puhvrisse. Järgmise samalaadse sündmuse korral toimetab DMA vastava sisendpuhvri aga RcvL2 puhvrisse. Fir() funktsioon käitub sarnaselt, aga vastasfaasis – esimisel korral kasutatakse RcvL2 puhvrit, teisel korral RcvL1 puhvrit sisendina. Selline arhitektuur hoiab ära konflikti, kus näiteks DMA täidab puhvreid samal ajal kui mõni DSP funktsioon sama puhvrit loeb. Realiseeritud on selline lähenemine kasutades programmkoodis lisamuutujaid Current_RxL_Channel, Current_RxR_Channel, Current_TxL_Channel ja Current_TxR_Channel, mille algväärtus on 1. Näiteks eelkirjeldatud olukorras, kus I2S-kontroller saadab esimese sündmuse DMA-le vasaku sisendpuhvri täitumisest, transpordib DMA andmed RcvL1 puhvrisse (sest Current_RxL_Channel=1 ja sihtpunktiks on seadistatud RcvL1 puhver).

Kui DMA kanal on oma töö lõpetanud, genereerib ta DSP-le katkestuse (“*interrupt”*). DSP seiskab ajutiselt oma tavapärase töö (main() funktsiooni täitmine) ja alustab DMA katkestusrutiini täitmist. Rutiin on kirjeldatud spetsiaalses katkestus-funktsioonis nimega DMA_Isr(void), mis asub lähtekoodi failis dma_bypass1.c. Katkestusrutiini käigus seatakse Current_RxL_Channel muutuja väärtusele 2 ja konfigureeritakse ringi DMA sihtpunkt (RcvL1 asemel RcvL2 puhver). Samuti algväärtustatakse katkestust indikeeriv bitt DMAIFR registris (*DMA Interrupt Flag Register* [9]). Kuna DMA sihtpunkti aadress on muudetud, täidab DMA järgneva I2S-sündmuse korral RcvL2 puhvi. Pärast transportimise lõppu genereeritakse uus katkestus DSP-le. Katkestusrutiini käigus seatakse parameeter Current_RxL_Channel tagasi väärtusele 1 ja DMA sihtpunkt muudetakse samuti tagasi RcvL1 puhvrile. Sarnane protsess toimub ka RcvR, FilterOutL ja FilterOutR puhvritega, seda nii DMA kontrolleri kui ka main() funtsiooni töös. Protsessi illustreerivad lisas 9 toodud joonised.

Lisaks DMA_Isr(void) katkestusrutiinile on kasutusel ka teine katkestusrutiin – RTC_Isr(void). Viimane käivitatakse igal täissekundil. Selle katkestusrutiini eesmärk on tuvastada sisendsignaali võendamissagedus, mis on realiseeritud I2S1-sisendkanali poolt DMA-le genereeritud sündmuste loendamise teel. Selle tarbeks on kasutusele võetud muutuja nimega eventCounter, mida suurendatakse I2S1 sisendsignaali kanali sündmuse tekkimisel. Kuna iga südmus toob endaga kaasa DMA käivitumise, mis omakorda toob kaasa DMA katkestuse DSP jaoks, toimub eventCounter muutuja inkrementeerimine DMA_Isr(void) katkestusrutiinis. Teisisõnu võib öelda, et eventCounter loendab I2S-sisendkanali võendite transportimisega tegeleva DMA kanali katkestusi. Ühe sekundi jooksul loendatud sündmuste hulga abil on lihtne arvutada sisendsignaali võendamissagedust. Lisaks eventCounter väärtusele tuleb arvestada ka DMA poolt transporditava puhvri suuruse- ja “PACK” režiimi seadistusega. Vastavalt tuvastatud võendamissagedusele võetakse main() funktsioonis kasutusele sobiv madal- ja kõrgpääsfiltri koefitsentide komplekt (et analoog-domeenis oleks murdesagedus fikseeritud).

Filtrikoefitsentide massiivid asuvad lähtekoodi failis ref_data_bypass1.c. Massiivide suurus on määratud konstantse parameetriga COEF_48, mis on defineeritud päisefailis ref_data_bypass1.h. Parameetri nimetus viitab näitekoodis kasutatud konstandile 48, kuid antud töös on suurendatud filtri järku 100-ni, mistõttu on COEF_48 parameeteri väärtus antud projektis 101. Koefitsendid on arvutatud vastavalt käesoleva töö teoreetilises osas kirjeldatud Blackman-akna meetodile, kusjuures madal- ja kõrgpääsfiltri paarid on komplementaarsed. Defineeritud on 5 komplekti filtrikoefitsente (seega 10 massiivi), mis kannavad järgmisi nimetusi:

- *coeff_fir_44_1_lowpass_2kHz*
- *coeff_fir_44_1_highpass_2kHz*
- *coeff_fir_48_lowpass_2kHz*
- *coeff_fir_48_highpass_2kHz*
- *coeff_fir_88_2_lowpass_2kHz*
- *coeff_fir_88_2_highpass_2kHz*
- *coeff_fir_96_lowpass_2kHz*
- *coeff_fir_96_highpass_2kHz*
- *coeff_fir_192_lowpass_2kHz*
- *coeff_fir_192_highpass_2kHz*

Tulles tagasi ülesandepüstituse juurde, on antud platvormi üheks peamiseks eesmärgiks erinevate FIR-filtrite katsetamiseks vajaliku keskkonna realiseerimine. Erinevate filtrikoefitsendi-massiivide loomine ja kasutamine lähtekoodis on seega vägagi ootuspärane. Näiteks Blackman-akna asemel tasub katsetada ka optimaalseid FIR-filtreid (Parks-McClellan meetodid), Kaiser-aknaid, miks mitte ka enda disainitud aknaid või muul meetodil leitud filtrikoefitsente. Samuti tasub katsetada erinevat järku filtreid muutes COEF_48 parameetrit ja sellega seoses muidugi ka filtrikoefitsentide massiive.

4.5 Võimalikud tarkvaralised edasiarendused

"TMS320C5505 eZdsp USB Stick" arendusplaadi riistvara võimaldab eelnevalt kirjeldatud tarkvaralise lahenduse edasiarendusi, mis on käesolevaks hetkeks realiseerimata. Järgnevalt on toodud mõned näited, milliste meetoditega on võimalik lisada platvormile funktsionaalsust ja/või efektiivsust signaalitöötlemise ülesannete lahendamisel.

1. Fir() funktsiooni modifitseerimine 32-bitiste filtrikoefitsentide rakendamiseks.

Fir() funktsioon rakendab sisendvõenditele konvolutsiooni valitud filtrikoefitsentidega (impulsskajaga), andes tulemuseks väljundvõendid, mis moodustavad filtreeritud väljundsignaali. Fir() funktsioon eeldab 16-bitise resolutsiooniga filtrikoefitsente, mistõttu läheb suurema resolutsiooniga võendite töötlemisel kaduma informatsiooni. Näiteks 24-bitise võendi korrutamisel 16-bitise koefitsendiga on viimased kaheksa bitti sisuliselt kasutud. Programmkoodi edasisel arendamisel tuleb ilmselt muuta fir() funktsiooni, et effektiivselt filtreerida ka kuni 32-bitiseid võendeid. Töö dokumenteerimise hetkel seda funktsiooni siiski muudetud ei ole ja filtrikoefitsendi sügavus on endiselt 16 bitti.

2.FFT algoritmi kasutamine signaalitöötluseks

Eelmises punktis toodud konvolutsioonimeetodit rakendav fir() funktsiooni võib ka välja vahetada mõne FFT (ingl. k *Fast Fourier' Transform*) algoritmi vastu. Võib öelda, et FFT on digitaalse signaalitöötluse jaoks optimeeritud versioon DFT-st. Helisignaali töötlemise kontekstis tuleb protsessoril teisendada ajavallas esitatud digitaalne signaal sagedusvalda, teostada vajalikud tehted vastavalt algoritmile ja lõpuks teisendada sagedusvallas saadud tulemus tagasi ajavalda. Kuigi signaali edasi-tagasi konverteerimine paistab olevat ressursimahukas töö, on ta oluliselt efektiivsem

konvolutsioonist, kui filtri järk on suur. Mida sügavam on filtri järk, seda suurem mõte on FFT kasutamisel võrreldes konvolutsiooniga.

FFT kasutamist antud platvormil soodustab ka asjaolu, et C5505 DSP sisaldab riistvaralist üksust FFT-arvutuste kiirendamiseks, mis võimaldab kuni 1024-ndat järku filtri realiseerimist. Lisaks on TI kollektiivi poolt kirjutatud ka näiteprojekt FFT rakendamiseks antud töös kasutatava riistvara jaoks.[23]

3. FIR-koefitsentide genereerimine programmkoodis

Hetkel on salvestatud filtrikoefitsente sisaldavad massiivid DSP mällu, mille ressurss on piiratud (64kB EEPROM). Kui filtrikoefitsendi massive on palju ja massiivi pikkus on suur, ei pruugi nad mällu ära mahtuda. Näiteks 1024-ndat järku filtri realiseerimisel (võimalik antud riistvaraga ilmselt vaid FFT algoritmi rakendades) kulub 10 filtrikoefitsentide massiivi jaoks (16-bitine koefitsent) 20kB mälu. 32-bitiste filtrikoefitsentide korral aga juba 40kB. Mälupuuduse probleemi vältimiseks võib filtrikoefitsente genereerida jooksvalt programmkoodis.

4. TMS320C5505 arendusplaadil oleva audiokoodeki rakendamine

Arendusplaat sisaldab endas audiokoodekit TLV320AIC3204, mis omakorda sisaldab küllaltki kõrgekvaliteedilist DAC-muundurit. Antud näitekoodis on see jäänud kasutamata, kuna projektis kasutatud SPDIF-vastuvõtja (protokollikonverter) sisaldab trükkplaadil ka DAC-d, mille rakendamine on lihtsam. Lisanduva DAC kasutamine võimaldab näiteks heli filtreerimist kuni neljaribalise kõlari jaoks.

Samuti võib kasutusele võtta audiokoodekis asuva ADC näiteks selleks, et lisada platvormile analoogdomeeni stereo helisignaali sisend. Ülesandepüstituse kohaselt ei olnud see eesmärk, mistõttu ei ole seda sisendit rakendatud. Platvormi universaalsuse suurendamise kaalutlustel võib vastava tarkvaralise edasiarenduse sellegipoolest teha.

5. Väljundsignaali ülesvõendamine (ingl. k *upsampling* või *oversampling*)

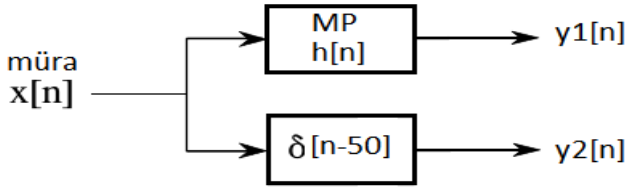
Eelmises punktis kirjeldatud DAC kasutamine võib osutada kasulikuks ka tänu sellele, et ta asub erinevas I2S-pordis kui SPDIF-vastuvõtja (vastupidiselt antud projekti lahendusest, kus sisend ja väljund on mõlemad I2S1-pordis). Kui DAC asub füüsiliselt eraldi pordis, võib DSP olla SPDIF-konverteri suhtes I2S *slave*-seade, DAC suhtes aga I2S *master*-seade. See omakorda võimaldab helisignaali tarkvaralist ülesvõendamist. Näiteks võib teha tarkvaralise lahenduse, kus olenemata sisendsignaali võendamissagedusest on väljundsignaal DAC-le alati 192kHz võendamissagedusega. See lihtsustab DAC väljundi analoogfiltrite disaini ja/või parandab nende filtrite efektiivsust. Meenutagem, et diskreetsete signaalide sagedusspekter on perioodiline perioodiga 2π [rad].

Tasub ka arvestada asjaoluga, et kui DSP on I2S *master*-seade, peab ta sisemine taktsagedus olema jaguv I2S kaadrisünkroniseerimis-sagedusega (antud näites 192kHz-ga). Teisisõnu, kui DSP on DAC suhtes I2S *master*-seade, aga väljund-võendamissagedus ühtib sisend-võendamissagedusega (ei teostata ülesvõendamist), peab DSP sisemine taktsagedus jaguma kõikide toetatud helisignaali võendamissagedustega (kui just DSP sisemist kella käigu pealt ümber ei programmeerita või ei kasutata selleks välist sünkrosignaali). Sobiva sisemise taktsageduse vabalt valimine DSP jaoks ei pruugi olla alati võimalik – selles olukorras on ilmselt mõistlik fikseerida väljund-võendamissagedus ülesvõendamise näol ja valida sellele vastavalt DSP sisemine taktsagedus.

5. Hinnang platvormi töövõimelisusele ja madalpääsfiltri spektri-analüüs

Helisüsteemide analüüsimisel on tüüpiliselt kaks lähenemist – helisignaali kuulamine ja helisignaali mõõtmine. Helisignaali hindamine kuulamise meetodil on vägagi mõistuspärane tegevus, aga sel meetodil saab anda vaid subjektiivseid hinnanguid. Selles peatükis on mõõdetud platvormi madalpääsfiltri väljundsignaali ADC vahendusel. Samuti on toodud katsetulemuse joonised sagedusvalla-esituses ja ülekandefunktsioonina.

Joonisel 21 on kirjeldatud katsesüsteemi, kus sisendsignaaliks on "valge müra" (digitaalne sisend). Platvormisiseselt rakendatakse väljundi y1[n] tekitamiseks magistritöös disainitud madalpääsfiltrit (Blackman-aken) ja kasutatakse vastavaid filtrikoefitsente. Väljundile y2[n] vastavad filtrikoefitsendid on nullised, välja arvatud keskpunktis h[50], mis on väärtustatud täisskaalaga (väärtus 32767). Teisisõnu on impulsskajaks valitud nihutatud ja skaleeritud ühikimpulss, mis vaid hilistab sisendsignaali. Selline ülesehitus võimaldab platvormi analoogväljunditest mõõta kahte huvipakkuvat signaali samaaegselt: originaalne sisendsignaal ja sama signaal läbinuna platvormi madalpääsfiltrit.



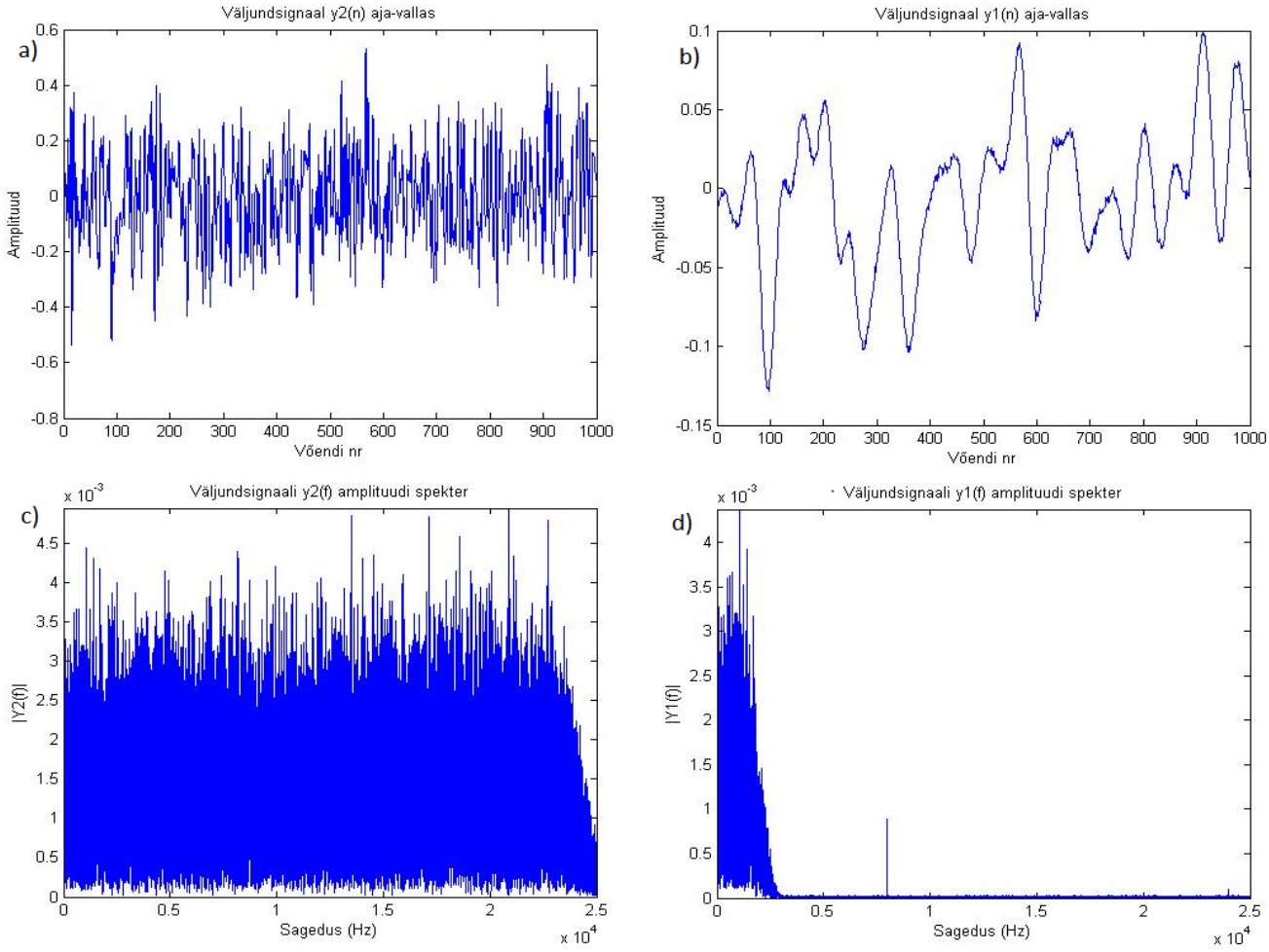
Joonis 21. Platvormi töövõimelisuse hindamiseks loodud katsesüsteemi tööpõhimõte.

Katsesüsteemi väljundite mõõtmiseks on rakendatud National Instruments tootevalikus olevat andmehõiveplaati PCI-6221. Kasutatud on vastava andmehõiveplaadi analoog-digitaal muundit 16-bitise resolutsiooni ja 100kHz võendamissageduse juures. Mõõtmine teostati ühe sekundi jooksul, mistõttu tekkis analüüsimiseks 100000 võendit mõlema väljundi kohta. Sisendsignaal ("valge müra") on genereeritud tarkvaraliselt ja edastatakse platvormini digitaalsel kujul resolutsiooniga 16 bitti ja võendamissagedusega 48kHz.

"Valge müra" kasutamine sisendsignaalina on tingitud asjaolust, et see sisaldab teoreetiliselt kõiki sageduskomponente, mistõttu on ideaalse "valge müra" sageduskarakteristik lame ja pidev kogu sagedustelje ulatuses. Praktikas on "valge müra" tekitamine keeruline ja selle kvaliteet alati vaieldav. Antud katse ja töö eesmärgiks ei ole "valge müra" kvaliteedi- ja juhuslikkuseprobleemide hindamine ega lahendamine. Ka madalakvaliteediline "valge müra" sisendis annab võimaluse platvormi töövõimelisuse hindamiseks.

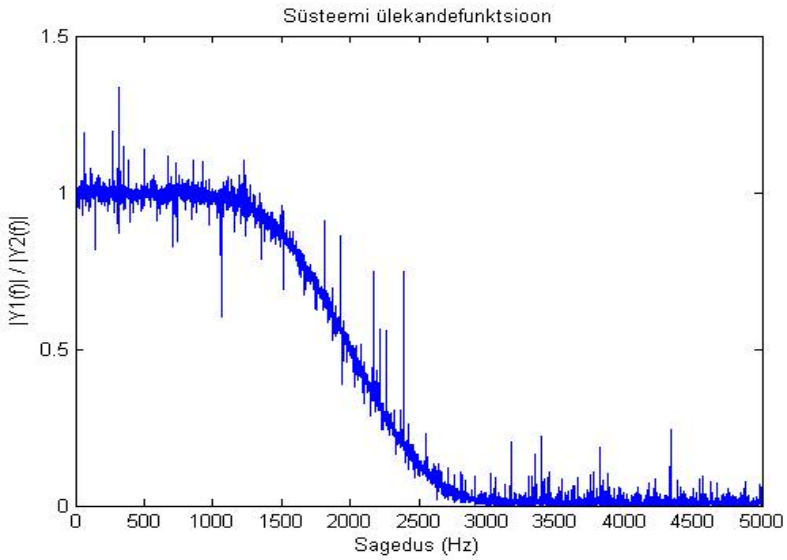
Joonisel 22 on toodud ADC abil mõõdetud platvormi väljundsignaalid y1[n] (joonis 22b) ja y2[n] (joonis 22a) ajavallas. Kujutatud on vaid esimese 1000 võendi väärtused, et signaalid oleksid visuaalselt hinnatavad. Samuti on samal joonisel kujutatud ka aja-valla signaalide Fourier' teisendust (FFT). Jooniselt 22c ilmneb, et "valge müra" sisendsignaalina katab tõepoolest suure osa sagedusribast, kuid sisaldab

endas teatavaid hüppeid signaali amplituudides teatud sagedustel. Sarnased hüpped ilmnevad samas signaalis ka madalpääsfiltrit läbinuna (joonis 22d). Märkimisväärne asjaolu joonise 22 juures on see, et madalpääsfilter on oluliselt vähendanud signaalide amplituudi, mille sagedus on üle 3kHz. Töö teoreetilises osas disainitud madalpääsfiltri lõikesagedus 48kHz võendamissageduse juures on 2kHz ja vastavalt valemile 32 on madalpääsfiltri siirdeala laius ligi 1.3kHz. Neid asjaolusid arvestades on mõõtmiste tulemus ootuspärane. Sellest omakorda võib järeldada, et platvormi tarkvaraline- ja riistvaraline lahendus on töövõimeline.



Joonis 22. Platvormi töövõimelisuse hindamiseks loodud katsesüsteemi väljundid aja- ja sagedusvallas.

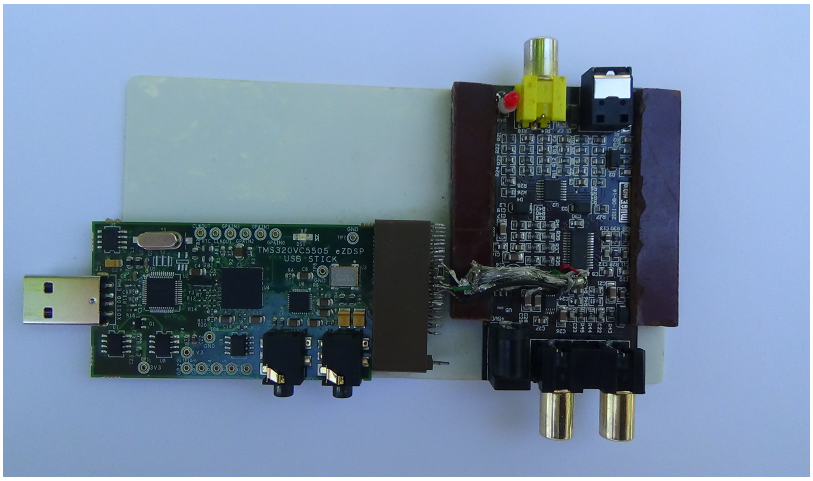
Joonisel 22c ja d kujutatud signaalide lihtsustatud sagedusvallaesitus võimaldab arvutada süsteemi madalpääsfiltri ülekandefunktsiooni jagades väljundsignaali (madalpääsfiltri läbinud signaali) FFT-amplituudid Y1(f) sisendsignaali FFT vastavate amplituudidega Y2(f). Vastava kompleksarvulise jagamise tulemus on visualiseeritud joonisel 23 suhtarvudena.



Joonis 23. Platvormi töövõimelisuse hindamisel arvutatud ülekandefunktsioon.

Joonis 23 visualiseerib sarnaselt joonisele 22 süsteemi töövõimelisust. Ülekanne väärtusega 1 tähistab signaali sageduskomponendi läbipääsemist filtrist ilma ampiltuudi vähenemiseta, ülekande nullilähedane suhtarv viitab sageduskomponendi amplituudi olulist vähenemist. Lõikesagedusel 2000Hz on signaali ampiltuud vähenenud kaks korda, mis vastab teoreetilisetele alustele (-6dB punkt joonisel 12). Nii joonisel 22 kui ka 23 on FFT amplituudides hüpped, mis on ilmselt põhjustatud häiretest. Häireallikaks võib olla katsesüsteemis kasutatud toiteplokid, juhtmestuse kehva kvaliteet, kõrvalekalded DAC ning ADC töös, FFT arvutamisel, "valge müra" generaatori töös ja teataval määral ka digitaalsete signaalide kvantimisel tekkinud vead.

Platvormi väljundsignaali andmehõive- ja katsetulemuseks olevate jooniste genereerimisega seotud programmkäsud on toodud lisas 10. Katse on realiseeritud MatLab keskkonnas. Joonisel 24 on kujutatud pilt lahenduse töövõimelisuse hindamisel kasutatud pilootplatvormist. Kahe plaadi vahelised ühendused on kirjeldatud joonisel 19.



Joonis 24. Pilootplatvorm.

6. Alternatiivsed lahendused

Seoses asjaoluga, et helisüsteeme on arendatud pikka aega, on joonisel 1b kujutatud kontseptsiooni juba varasemalt realiseeritud, peamiselt professionaalsetes helisüsteemides ja helientusiastide seas. Lisaks on viimastel aastatel tohutu kiirusega arenenud DSP-d ja sardarvutid (ingl. k *embedded PC*), mis loovad eeldused kirjeldatud kontseptsiooni realiseerimiseks madala rahalise ressursi tingimuses. Järgnevalt on toodud mõned näited alternatiivsetest lahendustest FIR-filtrite realiseerimiseks helisüsteemides, tuues sealjuures ära ka alternatiivsete lahenduste peamised eelised ja puudused võrreldes antud projektis kasutatud lahendusega. Laias laastus võib jagada alternatiivsed lahendused kahte leeri: DSP-l ja sardarvutitel baseeruvad lahendused.

6.1 DSP-l baseeruvad alternatiivsed lahendused

Joonisel 1b kujutatud kontseptsioonis on kasutatud analoog-helivõimendeid, mis toovad endaga kaasa vajaduse lisada süsteemi ka DACd. Sellise lähenemise kasuks räägib asjaolu, et analoogvõimendeid on arendatud pikka aega, mistõttu on nende korrektsel disainimisel võimalik saavutada minimaalseid moonutusi väljund-helisignaalis. Samas, tehniliselt on joonise 1b kontseptsioonile ka omapärane alternatiiv, kus analoog-helivõimendi asemel kasutatakse digitaalset (D-klass) helivõimendit. Sellise alternatiivse lahenduse teeb atraktiivseks tõsiasi, et näiteks TI tootevalikus on ka D-klassi helivõimendid, mis võtavad vastu I2S-formaadis esitatud digitaalset helisignaali. Üheks selliseks näiteks on TI mikroskeem TAS5731B [28]. Mikroskeemi TAS5731B rakendamisel kaob platvormis vajadus DAC järele, kuna helivõimendi sisendliideseks on I2S ja DSP võib seega ühenduda otse helivõimendi külge. Helivõimendi väljundsignaal on D-klassile omaselt impulsilaiusmodulatsioon (PWM), mis moduleeritakse selliselt, et valjuhääldi esitaks soovitud helisagedused (antud juhul vastavalt I2S-sisendile). Kirjeldatud alternatiivset platvormi võib nimetada tõeliselt diskreetseks helisüsteemiks. D-klassi helivõimendi tarbib kvaliteetse analoogvõimendiga võrreldes ka vähem võimsust, mis teeb ilmselt lihtsamaks toiteplokkide disaini. Kahjuks ei suudeta käesoleval hetkel digitaalsete võimendite väljundsignaali moonutusi kuigi madalal hoida, kuid pikemas perspektiivis on tegemist väljakutsuva alternatiivse lahendusega.

Käesolevale projektile sarnaneb suures osas WAF-Audio (Wroclaw Audio Force) poolt välja arendatud lahendus nimega “Nadja” [29]. Võrreldes antud töös kirjeldatud projektiga on “Nadja” lahendus loodud paiknema väljaspool kõlarit, jagades stereoheli nii vasaku kui ka parema kõlari valjuhäälditele samalt plaadilt. Seoses sellega sisaldab “Nadja” lahendus ka rohkem analoogväljundeid (8 väljundit) kui käesolevas projektis kirjeldatud lahendus. Samas tuleb arvestada, et mitme signaali väljastamiseks kulub DSP-l ka rohkem arvutuslikku ressurssi. Sisendsignaali osas on mõlema projekti riistvara võrdlemisi sarnane – võimalik on kasutada analoog- ja digitaalsisendit (nii I2S kui ka SPDIF). “Nadja” projekti peamine eelis käesoleva projektiga võrreldes on paindlikkus läbi tarkvaralise lahenduse. Nadja lahendust saab rakendada nii IIR (ingl. k *Infinite Impulse Response*) režiimis (rakendades kuni 8. järku analoogfiltrite impulsskaja) kui ka FIR-režiimis. “Nadja” lahenduse riistvaraline võimekus FIR-filtreerimisel ületab antud projektis kasutatud riistvara võimekust (arvatavasti ka C5505 FFT-kiirendit realiseerides) rakendades Freescale Semiconductor Inc. poolt toodetud protsessorit DSP56725. Lisaks on “Nadja” lahendus hallatav läbi

konfiguratsioonitarkvara, mis ühendub platvormiga USB-liidese vahendusel. FIR-filtrikoefitsendid tuleb lahendusele siiski sisestada mälukandjal (SD-kaart).

WAF-Audio "Nadja" lahendus FIR-filtrite realiseerimiseks paistab olema põhjalikult läbimõeldud ja vastavalt realiseeritud. Antud ülesandepüstitusele vastava lahenduse realiseerimiseks, kus DSP asub kõlari kasti sees, peab stereoheli filtreerimiseks soetama kaks "Nadja" platvormi. Paraku on "Nadja" riistvara kallim magistritöös kasutatavast riistvarast, hind algab alates 245EUR-ist. Lisaks on "Nadja" platvormi gabariidid oluliselt suuremad antud töös kirjeldatud lahendusest, mis võib osutuda probleemiks väikese kõlarikasti korral – lisaks DSP-le peavad kõlarikasti mahtuma ka võimendid ja toiteplokid, mistõttu on eelistatud väikesed gabariidid.

Teiseks sarnaseks platvormiks on MiniDSP Ltd. poolt toodetav arendusplaat nimega "miniSHARC"[30]. Nagu nimest võib aimata, on projektis rakendatud Analog Devices "SHARC" arhitektuuriga ujukoma-protsessorit ADSP21369. "MiniSHARC" sisendid ja väljundid põhinevad I2S- ja SPDIF-liidestel, mistõttu tuleb püstitatud ülesande realiseerimiseks siduda väline DAC plaadiga ise. Nagu ka "Nadja" platvormi puhul, on miniDSP arendusplaatide realiseerimiseks arendatud mugav kasutajaliides, mis peab sidet USB vahendusel. FIR-filtrite osas saab rakendada nii IIR-režiimis (rakendades kuni 8. järku analoogfiltrite impulsskaja) kui ka FIR-režiimis, kus filtri sügavus on kuni 1024 koefitsenti. Võrreldes käesoleva projekti lahendusega puudub "miniSHARC" arendusplaadil optiline SPDIF-liides ja audio koodek, mis on siiski võrdlemisi lihtsasti süsteemi lisatavad. DSP signaalitöötlemise ressursi kontekstis võib pidada "miniSHARC" arendusplaati paremaks käesoleva projekti DSP-st (VC5505) seoses kiirema sisemise kellaga (400MHz), kuigi FFT-kiirendid võimaldavad mõlemal juhul kuni 1024 filtrikoefitsendi kasutamist. "MiniSHARC" arendusplaadi hind Ameerika Ühendriikides on 185 USD (ligi 130 EUR), millele tuleb püstitatud ülesande realiseerimiseks teha lisakulutusi optilise SPDIF-sisendi ja DAC osas.

"Nadja" ja "miniSHARC" lahendusi on audiohuviliste seas korduvalt rakendatud käesoleva töö ülesandepüstitusele sarnaneva helisüsteemi realiseerimiseks, näiteks "DIYaudio" kommuuni siseselt. Üldistavalt võib öelda, et nimetatud platvorme tasub eelistada antud töö lahendusele, kui rahaline ressurss ei ole kuigi piiratud ja vajatakse graafilist konfiguratsioonitarkvara platvormi mugavaks seadistamiseks ja haldamiseks.

Laiatarbe-helitehnikana toodetakse ka aktiivkõlareid, mis sisaldavad operatsioonivõimenditel baseeruvaid aktiivfiltreid (analoogfiltrid). Heaks näiteks on Behringer Truth seeriast leitav B2030A stuudio-monitor, mis sisaldab neljandat järku Linkwitz-Riley analoogfiltreid. Toodud näide ei kvalifitseeru siiski antud ülesandepüstitusele vastavaks, kuna heli edastatamine kõlarini ja ka filtreerimine toimub analoog-domeenis.

6.2 Sardarvutil baseeruvad alternatiivsed lahendused

FIR-filtrite realiseerimine on võimalik ka klassikalisel "von Neumann"-i arhitektuuriga arvutil (PC), sealhulgas ka sardarvutitel. Võrreldes DSP-lahendustega on tüüpiliselt PC-del teatavad eelised (universaalsemad tarkvaralised ja riistvaralised lahendused), kuid omavad ka suuri puudusi:

- aeglane andmeedastus protsessori ja mälu vahel (signaalitöötluse kontekstis) võrreldes DSP-ga (nn "von Neumann-i arhitektuuri pudelikael")

- võrdlemisi suur energiatarve ja seega vajadus riistvara jahutamiseks, mis ei sobi kuigi hästi kokku aktiivse helisüsteemi kontseptsiooniga (joonis 1b), kus kogu riistvara asub kõlari sees

Antud töös on ilma pikema aruteluta eelistatud DSP (HARC) arhitektuuri arvestades asjaolu, et DSP on spetsiaalselt loodud lahendama signaalitöötluse ülesandeid, eriti konvoultsiooni (MAC-tehted realiseeritakse riistvaras). Siiski tasub mainida, et digitaalseid filtreid realiseeritakse edukalt ka PC-del, näiteks "Foobar" tarkvara abil koos vastava tarkvaraliidesega [31] Windows operatsioonisüsteemidel. Samuti realiseeritakse FIR-filtreid näiteks "BruteFIR" tarkvara abil [32] Linux operatsioonisüsteemidel. Mõlemal juhul rakendatakse tarkvaras sagedusvalla-signaalitöötlust kasutades FFT-d.

Sardarvuti kasutamisel on DSP-ga võrreldes lihtsam rakendada *ethernet*-liidest näiteks kaughalduseks, samuti ka helisignaali edastamiseks. Sisuliselt on võimalik joonisel 1b kujutatud helisüsteem, kus SPDIF-liidese asemel on sisendiks ethernet liidesest üle kantav helivõendite jada kasutades kasutaja-datagrammi protokolli (ingl. k *UDP - User Datagram Protocol*). Keeruliseks teeb selle lahenduse puhul vasaku ja parema stereosignaali kanalite omavaheline sünkroniseerimine sardarvutite vahel. Helisignaali transportimiseks erineva riistvara vahel on loodud vabavaralisi "JACK" utiliite, näiteks *ethernet*-liidese kasutamiseks on loodud "NetJack" utiliit [33].

Kui antud projekti kontektis asendada DSP sardarvutiga ja digitaalse helisignaali edastamiseks sardarvutini kasutada näiteks SPDIF-liidest, peab ilmselgelt olema sardarvutil vastav sisend. Samuti peab sardarvutil olema võimalus helikaardi (või DAC) ühendamiseks. Kuigi paljud sardarvutid on võrdlemisi odavad, on vajalike sisend- ja väljundliideste lisamine tihtipeale keeruline või kulukas. Erandiks võib lugeda näiteks populaarse sardarvuti "Raspberry Pi", millele on võimalik osta "Hifiberry" nimeline DAC-lisaplaat (valikus digitaalsed- või analoogväljundid)[34], komplekti hind on alla 70 EUR. Paraku on "Raspberry Pi"-l suure tõenäolisusega liiga vähe ressurssi antud töö kontekstis vajaliku signaalitöötluse ülesannete lahendamiseks lisaks operatsioonisüsteemi käigushoidmisele. Oluliselt võimekamad on näiteks "Cubieboard" seeria sardarvutid, mis kasutavad "Allwinner A20" protsessorit. Eelistada võiks "Cubietruck" mudelit, millel on plaadi servale välja toodud ka I2S- ja SPDIF-väljundid, mida saab kasutada sobiva DAC lisamiseks sardarvutile [35]. "Cubietruck" sardarvuti hind (ilma DAC-ta) on ligi 100 EUR.

Sardarvutid arenevad väga kiiresti ja erinevaid mudeleid toodetakse aina juurde, mistõttu ei ole neid rohkem konkreetselt nimetatud. Sardarvuti kasutamisel FIR-filtrite rakendamiseks tuleb ilmselt tundma õppida Linux operatsioonisüsteemi ja selle arendamist, mis on mahukas töö. Kui sisend- ja väljundliideste lahendus on valitud, on sardarvuti kasutamisel ilmselt kõige lihtsam signaalitöötluse ülesannete realiseerimiseks pruukida "BruteFIR" tarkvara Linux operatsioonisüsteemil.

7. Kokkuvõte

Töö annab ülevaate DSP-l baseeruvast platvormist FIR-filtrite realiseerimiseks helisüsteemides. Välja on pakutud helisüsteemi kontseptsioon, mis ei ole küll esmakordne maailmas, kuid on olnud kättesaamatu laiatarbe-helitehnikas. Disainitud platvorm on loodud võrdlemisi väikese rahalise ressursi tingimustes. Töö hõlmab endas teoreetilisi aluseid, mille tundmine on otseselt vajalik lihtsamate FIR-filtrite disainimiseks ja digitaalse signaalitöötluse ülesannete lahendamiseks üldisemalt. Kirjeldatud on ka kasutatud DSP-plaadi "TMS320VC5505 eZdsp USB Stick" riistvaralist ülesehitust ja funktsionaalsust ning digitaalsete helisignaalide transportimiseks kasutatavate andmesideprotokollide (I2S ja SPDIF) olemust. Töö teoreetiline osa on realiseeritud ka praktikas läbi DSP tarkvaralise lahenduse, mis on antud tööle lisatud digitaalsel kujul. Tarkvaraline lahendus on ka graafiliselt illustreeritud, mis hõlpsustab kirjeldatud protsesside mõistmist.

Töö käigus valminud realisatsiooni näol on tegemist pilootplatvormiga, mille tarkvaraline lahendus on töövõimeline rakendades sajandat järku FIR-filtreid konvolutsioonimeetodil, kuid tarkvara kuulub edasiarendamisele. Olulisemateks planeeritavateks muudatusteks on 32-bitise resolutsiooniga FIR-filtrikoefitsentide kasutuselevõtmine 16-bitiste asemel ja sellega seoses assemblerkeeles kirjutatud fir() funktsiooni modifitseerimine. Lisaks on plaanis realiseerida filtreerimist sagedusvallas kasutades DSP sisemist FFT riistvaralist kiirendit. Sel meetodil on võimalik suurendada rakendatud FIR-filtrite järk (antud töös 100) kuni 1024ni, mille vajadus ilmneb peamiselt kõrge võendamissagedusega (nt 192kHz) esitatud helisignaalide töötlemisel.

Pilootplatvormil puudub graafiline kasutajaliides platvormi seadistamiseks, mistõttu muudatused DSP töös tuleb teostada tarkvaraliselt programmkoodis. Graafilise kasutajaliidese vajaduse ilmnemisel tasub kaaluda alternatiivseid lahendusi näiteks MiniDSP ja WAF-Audio vastava funktsionaalsusega toodete näol. Alternatiivsete lahenduste kasuks otsustamisel tuleb arvestada märkimisväärselt kõrgema hinnaga (arvestamata kulutusi pilootplatvormi tarkvaralisele arendusele).

Hoolimata disainitud platvormi teatavatest puudujääkidest on tegemist helisüsteemi kontseptsiooniga, mida võib liigitada professionaalseks helitehnikaks. Platvormi funktsionaalsus vastab töös esitatud ülesandepüstitusele, mistõttu võib disainitud lahenduse lugeda õnnestunud projektiks.

Kasutatud kirjandus

[1] Hando Sillamaa (2003). Süsteemiteooria; kolmas, täiendatud väljaanne.
[2] Steven W. Smith (1999). The Scientist and Engineer's Guide to Digital Signal Processing. Second Edition. California Technical Publishing, San Diego, California.
[3] Tarmo Lipping (2007). Digitaalne signaali- ja pilditöötlus – loengukonspekt. TTÜ kirjastus.
[4] Rain Ferents (2011). Digitaalne signaalitöötlus – valitud FIR-filtrite disainimine. TTÜ kirjastus
[5] Alan V. Oppenheim, Roland W. Schafer, John R. Buck (1999). Discrete-Time Signal Processing. Second edition. Prentice Hall Inc.
[6] John G. Proakis, Dimitris G. Manolakis (1996). Digital Signal Processing – principles, algorithms, and applications. Third edition. Prentice Hall International Inc.
[7] Emmanuel C. Ifeachor, Barrie W. Jervis (2001). Digital signal processing – a practical approach. Second edition. Pearson UK
[8] [WWW] Texas Instruments, (2009). Literature number SWPU073E - C55x v3.x CPU Reference Guide. http://www.ti.com/lit/ug/swpu073e/swpu073e.pdf (13.05.2014)
[9] [WWW] Texas Instruments, (2009, parandatud 2012). Literature number SPRUFP0C – TMS320VC5505 DSP System User's Guide. http://www.ti.com/lit/ug/sprufp0c/sprufp0c.pdf (13.05.2014)
[10] [WWW] Texas Instruments, (2009, parandatud 2010). Literature number SPRUFO8 – TMS320VC5505/5504 DSP External Memory Interface (EMIF) User's Guide. http://www.ti.com/lit/ug/sprufo8a/sprufo8a.pdf (13.05.2014)
[11] [WWW] Texas Instruments, (2009, parandatud 2010). Literature number SPRS503B – TMS320VC5505 Fixed-Point Digital Signal Processor. http://www.ti.com/lit/ds/sprs503b/sprs503b.pdf (13.05.2014)
[12] [WWW] Texas Instruments, (2009). Literature number SPRUFO9 – TMS320VC5505/5504 DSP Direct Memory Access (DMA) Controller User's Guide. http://www.ti.com/lit/ug/sprufo9/sprufo9.pdf (13.05.2014)
[13] [WWW] Spectrum Digital Inc. (2009). TMS320VC5505 eZdsp TM USB Stick Technical Reference. http://support.spectrumdigital.com/boards/usbstk5505/revb/files/usbstk5505_TechRef_revb.pdf (13.05.2014)
[14] [WWW] Texas Instruments, (2012). Literature number SLAA557 – TLV320AIC3204 Application Reference Guide. http://www.ti.com/lit/ml/slaa557/slaa557.pdf (13.05.2014)
[15] [WWW] Texas Instruments, (2009, parandatud 2010). Literature number SPRUFP4 - TMS320VC5505/5504 DSP Inter-IC Sound (I2S) Bus User's Guide http://www.ti.com/lit/ug/sprufp4a/sprufp4a.pdf (13.05.2014)
[16] [WWW] European Broadcasting Union, (2004). Specification of the digital audio interface (The AES/EBU interface). Third Edition. https://tech.ebu.ch/docs/tech/tech3250.pdf (13.05.2014)
[17] Steve Kilts, (2007). Advanced FPGA Design – Architecture, Implementation, and Optimization; John Wiley & Sons Inc.
[18] [WWW] Spectrum Digital Inc. (2009). TMS320C5505 eZdsp USB Stick schematics (rev. c). http://support.spectrumdigital.com/boards/usbstk5505/revc/files/usbstk5505_Schematics_revc.pdf (13.05.2014)

[19] [WWW] Cirrus Logic Inc. (2007). CS8416 - 192 kHz Digital Audio Interface Receiver http://www.cirrus.com/en/pubs/proDatasheet/CS8416_F3.pdf (13.05.2014)
[20] [WWW] Cirrus Logic Inc. (2013). C4344/5/8 - 10-Pin, 24-Bit, 192 kHz Stereo D/A Converter. http://www.cirrus.com/cn/pubs/proDatasheet/CS4344-45-48_F2.pdf (13.05.2014)
[21][WWW] Cirrus Logic Inc. (2004). Evaluation Board for CS4344 http://www.cirrus.com/jp/pubs/rdDatasheet/CDB4344-EB.pdf (13.05.2014)
[22] [WWW] Texas Instruments, (2006). Literature number SPRU509H – Code Composer Studio Development Tools v3.3 Getting Started Guide. http://www.ti.com/lit/ug/spru509h/spru509h.pdf (13.05.2014)
[23] [WWW] Reference Starting Point for C5000-based eZdsp USB Stick Development Tool. http://code.google.com/p/c5505-ezdsp/ (13.05.2014)
[24] [WWW] Texas Instruments, (2010). Literature number SPRAB92 – Using the TMS320VC5505/04 Bootloader. http://www.ti.com/lit/an/sprab92a/sprab92a.pdf (13.05.2014)
[25] [WWW] Texas Instruments Wiki (2010), C5505 PLL Calculator 060210 http://processors.wiki.ti.com/images/f/f0/C5505_PLL_Calculator_060210.zip
[26] [WWW] Texas Instruments, (2006, parandatud 2011). Literature number SPRU317K – TMS320C55x DSP Peripherals Overview User's Guide http://www.ti.com/lit/ug/spru317k/spru317k.pdf (13.05.2014)
[27] [WWW] Texas Instruments, (2009). Literature number SPRUFO7 – TMS320VC5505/5504 DSP Real-Time Clock (RTC) User's Guide. http://www.ti.com/lit/ug/sprufo7/sprufo7.pdf (13.05.2014)
[28] [WWW] Texas Instruments, (2013). Literature number SLOS838B - TAS5731B - 2×30-W Digital Audio Power Amplifier With DSP and 2.1 Mode http://www.ti.com/lit/ds/symlink/tas5731m.pdf (13.05.2014)
[29] [WWW] Wroclaw Audio Force. “Nadja” audio board. http://www.waf-audio.com/products.php?lang=en (13.05.2014)
[30] [WWW] MiniDSP Ltd. miniSHARC Kit, http://www.minidsp.com/products/opendrc-series/minisharc-kit (13.05.2014)
[31] [WWW] AEDIO Y.I. FIR active crossover plugin for Foobar http://www.aedio.co.jp/download/index_e.html (13.05.2014)
[32] [WWW] Anders Torger. BruteFIR. http://www.ludd.luth.se/~torger/brutefir.html (13.05.2014)
[33] [WWW] Paul Davis. Jack Audio Connection Kit. http://jackaudio.org/ (13.05.2014)
[34] [WWW] Modul 9 GmbH, Hifiberry. http://www.hifiberry.com/ (13.05.2014)
[35] [WWW] Cubieboard. A Series of Open ARM mini PC-s. http://cubieboard.org/ (13.05.2014)

Lisa 1. Joonisel 6 toodud impulsskaja võendite (koefitsentide) väärtused.
Filtri tüüp: Madalpääsfilter
Aknafunktsiooni tüüp: Nelinurk-aken
Murdesagedus : $\omega_c = \frac{1}{12}\pi\, rad$
Filtri järk: $M = 100$

Võendi nr n	h[n], skaleeritud	h[n], 16 bit integer
0	3.18310E-03	104
1	1.68132E-03	55
2	-3.24848E-18	0
3	-1.75287E-03	-57
4	-3.45989E-03	-113
5	-5.00176E-03	-164
6	-6.26510E-03	-205
7	-7.15032E-03	-234
8	-7.57881E-03	-248
9	-7.49912E-03	-246
10	-6.89161E-03	-226
11	-5.77126E-03	-189
12	-4.18829E-03	-137
13	-2.22661E-03	-73
14	3.24848E-18	0
15	2.35385E-03	77
16	4.68103E-03	153
17	6.82058E-03	223
18	8.61451E-03	282
19	9.91819E-03	325
20	1.06103E-02	348
21	1.06022E-02	347
22	9.84516E-03	323
23	8.33626E-03	273
24	6.12134E-03	201
25	3.29539E-03	108
26	-3.24848E-18	0
27	-3.58194E-03	-117
28	-7.23432E-03	-237
29	-1.07181E-02	-351
30	-1.37832E-02	-452
31	-1.61823E-02	-530
32	-1.76839E-02	-579
33	-1.80861E-02	-593
34	-1.72290E-02	-565
35	-1.50053E-02	-492
36	-1.13682E-02	-373
37	-6.33728E-03	-208
38	3.24848E-18	0
39	7.48951E-03	245
40	1.59155E-02	522
41	2.50088E-02	819
42	3.44581E-02	1129
43	4.39234E-02	1439
44	5.30516E-02	1738
45	6.14927E-02	2015
46	6.89161E-02	2258
47	7.50264E-02	2458
48	7.95775E-02	2608
49	8.23847E-02	2699
50	8.33333E-02	2731

Võendi nr n	h[n], skaleeritud	h[n], 16 bit integer
100	3.18310E-03	104
99	1.68132E-03	55
98	-3.24848E-18	0
97	-1.75287E-03	-57
96	-3.45989E-03	-113
95	-5.00176E-03	-164
94	-6.26510E-03	-205
93	-7.15032E-03	-234
92	-7.57881E-03	-248
91	-7.49912E-03	-246
90	-6.89161E-03	-226
89	-5.77126E-03	-189
88	-4.18829E-03	-137
87	-2.22661E-03	-73
86	3.24848E-18	0
85	2.35385E-03	77
84	4.68103E-03	153
83	6.82058E-03	223
82	8.61451E-03	282
81	9.91819E-03	325
80	1.06103E-02	348
79	1.06022E-02	347
78	9.84516E-03	323
77	8.33626E-03	273
76	6.12134E-03	201
75	3.29539E-03	108
74	-3.24848E-18	0
73	-3.58194E-03	-117
72	-7.23432E-03	-237
71	-1.07181E-02	-351
70	-1.37832E-02	-452
69	-1.61823E-02	-530
68	-1.76839E-02	-579
67	-1.80861E-02	-593
66	-1.72290E-02	-565
65	-1.50053E-02	-492
64	-1.13682E-02	-373
63	-6.33728E-03	-208
62	3.24848E-18	0
61	7.48951E-03	245
60	1.59155E-02	522
59	2.50088E-02	819
58	3.44581E-02	1129
57	4.39234E-02	1439
56	5.30516E-02	1738
55	6.14927E-02	2015
54	6.89161E-02	2258
53	7.50264E-02	2458
52	7.95775E-02	2608
51	8.23847E-02	2699

Lisa 2. Aknafunktsioonide ja kaalutud sinc funktsiooni kuvamiseks kasutatud programmkäsud “Ocatve” tarkvara baasil.

```
r=rectwin(101); save r_rectwin_100_coeff.mat r;
han=hanning(101); save han_hanning_100_coeff.mat han;
ham=hamming(101); save ham_hamming_100_coeff.mat ham;
bl=blackman(101); save bl_blackman_100_coeff.mat bl;
clear all;
load  r_rectwin_100_coeff.mat;
load  han_hanning_100_coeff.mat
load  ham_hamming_100_coeff.mat
load  bl_blackman_100_coeff.mat
figure(11);
subplot(2,1,1);
plot (r,'r'); hold on;
plot(han, 'g');
plot (ham, 'b');
plot (bl, 'm');
title('Aknafunktsioonid'); ylabel('Akna koefitsent');
legend({"Nelinurkaken","Hanning aken", "Hamming aken", "Blackman aken"}); legend("location",
"northeast");
axis([0, 101, -0.1,1.25]);
b = fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, rectwin(101)); save b_rectangular_coefficents.mat b;
b2 = fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, hanning(101)); save b2_hanning_coefficents.mat b2;
b3=fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, hamming(101)); save b3_hamming_coefficents.mat b3;
b4=fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, blackman(101)); save b4_blackman_coefficents.mat b4;
load  b_rectangular_coefficents.mat;
load b2_hanning_coefficents.mat;
load b3_hamming_coefficents.mat;
load b4_blackman_coefficents.mat;
subplot(2,1,2);
plot(b, 'r'); hold on;
plot(b2, 'g');
plot(b3, 'b');
plot(b4, 'm');
title('Aknafunktsiooniga kaalutud sinc funktsioon');
xlabel(strcat('V',char(245),'endi number')); ylabel('FIR filtri koefitsent');
legend({"Nelinurkakna sinc","Hanning sinc", "Hamming sinc", "Blackman sinc"}); legend("location",
"northeast");
axis([0, 101, -0.025,0.1]);
hold off;
```

Lisa 3. Faasi- ja sageduskarakteristiku kuvamiseks kasutatud programmkäsud "Octave" tarkvara baasil.

```
clear all
b = fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, rectwin(101)); save b_rectangular_coefficents.mat b;
b2 = fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, hanning(101)); save b2_hanning_coefficents.mat b2;
b3=fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, hamming(101)); save b3_hamming_coefficents.mat b3;
b4=fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, blackman(101)); save b4_blackman_coefficents.mat b4;
clear
load  b_rectangular_coefficents.mat;
load b2_hanning_coefficents.mat;
load b3_hamming_coefficents.mat;
load b4_blackman_coefficents.mat;

[H,w]=freqz(b);
[H2,w2]=freqz(b2);
[H3,w3]=freqz(b3);
[H4,w4]=freqz(b4);

subplot(3,1,1);
plot(w/pi,20*log10(abs(H)),'r');
hold on;
plot(w2/pi,20*log10(abs(H2)), 'g');
plot(w3/pi,20*log10(abs(H3)),'b');
plot(w4/pi,20*log10(abs(H4)),'m');
title('Sageduskarakteristikud H\(\omega\)'),
ylabel('Amplituud [dB]'),
legend({strcat('P',char(228), char(228), 'suriba (suurendatuna) (dB):nelinurkaken'),"Hanning aken",
"Hamming aken", "Blackman aken"});
legend("location", "northeast");
axis([0, 1, -1.5,0.85]);

subplot(3,1,2);
plot(w/pi,20*log10(abs(H)),'r');
hold on;
plot(w2/pi,20*log10(abs(H2)), 'g');
plot(w3/pi,20*log10(abs(H3)),'b');
plot(w4/pi,20*log10(abs(H4)),'m');
title('Sageduskarakteristikud H\(\omega\)');
ylabel('Amplituud [dB]');
legend ({strcat('T',char(245), 'kkeriba (dB): nelinurkaken'),"Hanning aken", "Hamming aken", "Blackman
aken"}, "location", "northeast");
axis([0, 1, -100,10]);

subplot(3,1,3);
plot(w/pi,(unwrap (arg (H))*360/(2*pi)),'r');
hold on;
plot(w2/pi,(unwrap (arg (H2))*360/(2*pi)),'g');
plot(w3/pi,(unwrap (arg (H3))*360/(2*pi)),'b');
plot(w4/pi,(unwrap (arg (H4))*360/(2*pi)),'m');
title('Faasikarakteristikud \Phi(\omega)');
ylabel(strcat('Faas [',char(176),']'));
xlabel ("Normaliseeritud sagedus [\Pi*rad]");
legend ({strcat('Faas (',char(176), '-ides): nelinurkaken'), "Hanning aken", "Hamming aken", "Blackman
aken"}, "location", "northeast");
axis([0, 1, -1500,0]);
hold off;
```

Lisa 4. FIR-madalpääsfiltri koefitsendid vastavalt joonisele 8.

Filtri tüüp: Madalpääsfilter

Aknafunktsiooni tüüp: Blackman-aken

Murdesagedus : $\omega_c = \frac{1}{12}\pi\, rad$

Filtri järk: $M = 100$

Võendi nr n	blackman h_b[n]	blackman h_b[n]
		16bit integer
0	0.00E+00	0
1	5.98E-07	0
2	-4.64E-21	0
3	-5.68E-06	0
4	-2.01E-05	-1
5	-4.60E-05	-2
6	-8.41E-05	-3
7	-1.33E-04	-4
8	-1.87E-04	-6
9	-2.39E-04	-8
10	-2.77E-04	-9
11	-2.87E-04	-9
12	-2.54E-04	-8
13	-1.62E-04	-5
14	2.80E-19	0
15	2.39E-04	8
16	5.52E-04	18
17	9.29E-04	30
18	1.34E-03	44
19	1.76E-03	58
20	2.13E-03	70
21	2.39E-03	78
22	2.48E-03	81
23	2.33E-03	76
24	1.89E-03	62
25	1.12E-03	37
26	-1.21E-18	0
27	-1.45E-03	-48
28	-3.18E-03	-104
29	-5.08E-03	-167
30	-7.03E-03	-230
31	-8.83E-03	-289
32	-1.03E-02	-337
33	-1.12E-02	-366
34	-1.13E-02	-369
35	-1.03E-02	-339
36	-8.23E-03	-270
37	-4.80E-03	-157
38	2.56E-18	0
39	6.14E-03	201
40	1.35E-02	443
41	2.19E-02	718
42	3.10E-02	1017
43	4.06E-02	1329
44	5.00E-02	1640
45	5.90E-02	1935
46	6.72E-02	2200
47	7.39E-02	2423
48	7.91E-02	2591
49	8.23E-02	2695
50	8.33E-02	2731

Võendi nr n	blackman h_b[n]	blackman h_b[n]
		16bit integer
100	0.00E+00	0
99	5.98E-07	0
98	-4.64E-21	0
97	-5.68E-06	0
96	-2.01E-05	-1
95	-4.60E-05	-2
94	-8.41E-05	-3
93	-1.33E-04	-4
92	-1.87E-04	-6
91	-2.39E-04	-8
90	-2.77E-04	-9
89	-2.87E-04	-9
88	-2.54E-04	-8
87	-1.62E-04	-5
86	2.80E-19	0
85	2.39E-04	8
84	5.52E-04	18
83	9.29E-04	30
82	1.34E-03	44
81	1.76E-03	58
80	2.13E-03	70
79	2.39E-03	78
78	2.48E-03	81
77	2.33E-03	76
76	1.89E-03	62
75	1.12E-03	37
74	-1.21E-18	0
73	-1.45E-03	-48
72	-3.18E-03	-104
71	-5.08E-03	-167
70	-7.03E-03	-230
69	-8.83E-03	-289
68	-1.03E-02	-337
67	-1.12E-02	-366
66	-1.13E-02	-369
65	-1.03E-02	-339
64	-8.23E-03	-270
63	-4.80E-03	-157
62	2.56E-18	0
61	6.14E-03	201
60	1.35E-02	443
59	2.19E-02	718
58	3.10E-02	1017
57	4.06E-02	1329
56	5.00E-02	1640
55	5.90E-02	1935
54	6.72E-02	2200
53	7.39E-02	2423
52	7.91E-02	2591
51	8.23E-02	2695

Lisa 5. FIR-kõrgpääsfiltri koefitsendid vastavalt joonisele 11.

Filtri tüüp: Kõrgpääsfilter
Aknafunktsiooni tüüp: Blackman-aken
Murdesagedus : $\omega_c = \frac{1}{12}\pi\, rad$
Filtri järk: $M = 100$

Võendi nr n	blackman h_b[n]	blackman h_b[n]
		16bit integer
0	-0.00E+00	0
1	-5.98E-07	0
2	4.64E-21	0
3	5.68E-06	0
4	2.01E-05	1
5	4.60E-05	2
6	8.41E-05	3
7	1.33E-04	4
8	1.87E-04	6
9	2.39E-04	8
10	2.77E-04	9
11	2.87E-04	9
12	2.54E-04	8
13	1.62E-04	5
14	-2.80E-19	0
15	-2.39E-04	-8
16	-5.52E-04	-18
17	-9.29E-04	-30
18	-1.34E-03	-44
19	-1.76E-03	-58
20	-2.13E-03	-70
21	-2.39E-03	-78
22	-2.48E-03	-81
23	-2.33E-03	-76
24	-1.89E-03	-62
25	-1.12E-03	-37
26	1.21E-18	0
27	1.45E-03	48
28	3.18E-03	104
29	5.08E-03	167
30	7.03E-03	230
31	8.83E-03	289
32	1.03E-02	337
33	1.12E-02	366
34	1.13E-02	369
35	1.03E-02	339
36	8.23E-03	270
37	4.80E-03	157
38	-2.56E-18	0
39	-6.14E-03	-201
40	-1.35E-02	-443
41	-2.19E-02	-718
42	-3.10E-02	-1017
43	-4.06E-02	-1329
44	-5.00E-02	-1640
45	-5.90E-02	-1935
46	-6.72E-02	-2200
47	-7.39E-02	-2423
48	-7.91E-02	-2591
49	-8.23E-02	-2695
50	9.17E-01	30036

Võendi nr n	blackman h_b[n]	blackman h_b[n]
		16bit integer
100	-0.00E+00	0
99	-5.98E-07	0
98	4.64E-21	0
97	5.68E-06	0
96	2.01E-05	1
95	4.60E-05	2
94	8.41E-05	3
93	1.33E-04	4
92	1.87E-04	6
91	2.39E-04	8
90	2.77E-04	9
89	2.87E-04	9
88	2.54E-04	8
87	1.62E-04	5
86	-2.80E-19	0
85	-2.39E-04	-8
84	-5.52E-04	-18
83	-9.29E-04	-30
82	-1.34E-03	-44
81	-1.76E-03	-58
80	-2.13E-03	-70
79	-2.39E-03	-78
78	-2.48E-03	-81
77	-2.33E-03	-76
76	-1.89E-03	-62
75	-1.12E-03	-37
74	1.21E-18	0
73	1.45E-03	48
72	3.18E-03	104
71	5.08E-03	167
70	7.03E-03	230
69	8.83E-03	289
68	1.03E-02	337
67	1.12E-02	366
66	1.13E-02	369
65	1.03E-02	339
64	8.23E-03	270
63	4.80E-03	157
62	-2.56E-18	0
61	-6.14E-03	-201
60	-1.35E-02	-443
59	-2.19E-02	-718
58	-3.10E-02	-1017
57	-4.06E-02	-1329
56	-5.00E-02	-1640
55	-5.90E-02	-1935
54	-6.72E-02	-2200
53	-7.39E-02	-2423
52	-7.91E-02	-2591
51	-8.23E-02	-2695

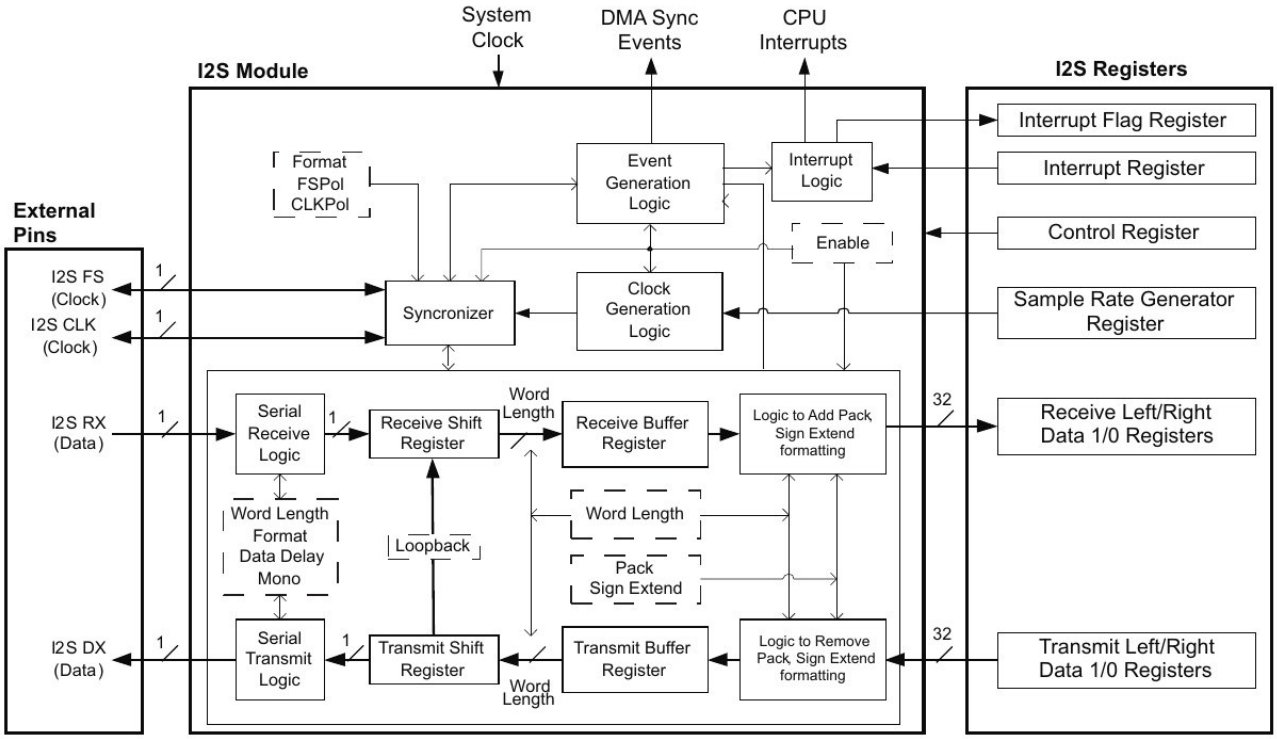
Lisa 6. Kõrgpääsfiltri FIR-koefitsentide genereerimine ja kuvamine koos madalpääsfiltriga “Octave” tarkvara baasil.

```
b4=fir2 (100, [0,1/12,1/12,1], [1,1,0,0], 2^24, 2, blackman(101)); save b4_blackman_coefficents.mat b4;
b4_high=-b4;
b4_high(51)=b4_high(51)+1;
figure(1);
plot(b4_high, 'r'); hold on;

title(strcat('K',char(245),'rgp', char(228), char(228), 'sfiltri FIR koefitsendid'));
xlabel(strcat('V',char(245),'endi number')); ylabel('FIR filtri koefitsent');
legend({"Spektraalselt inverteeritud sinc"}); legend("location", "northeast");
axis([0, 101, -0.1,1]);
hold off;
[H4,w4]=freqz(b4);
[H4_high,w4_high]=freqz(b4_high);
figure(2);
subplot(3,1,1);

plot(w4/pi,20*log10(abs(H4)),'r');
hold on;
plot(w4_high/pi,20*log10(abs(H4_high)),'g');
plot(w4/pi,20*log10((abs(H4)+abs(H4_high))/1),'b');
title('Sageduskarakteristikud H\(\omega\)'),
ylabel('Amplituud [dB]'),
legend({strcat('Madalp',char(228), char(228), 'sfilter (suurendatuna)'), strcat('K',char(245), 'rgp',
char(228), char(228), 'sfilter(suurendatuna)'),"Kahe filtri summa" });
legend("location", "east");
axis([0, 1, -10,0.85]);

subplot(3,1,2);
plot(w4/pi,20*log10(abs(H4)),'r');
hold on;
plot(w4_high/pi,20*log10(abs(H4_high)), 'g');
%plot(w4/pi,20*log10((abs(H4)+abs(H4_high))/1),'b');
title('Sageduskarakteristikud H\(\omega\)');
ylabel('Amplituud [dB]');
legend({strcat('Madalp',char(228), char(228), 'sfilter'), strcat('K',char(245), 'rgp', char(228), char(228),
'sfilter'),"Kahe filtri summa"});
legend("location", "east");
axis([0, 1, -100,10]);


subplot(3,1,3);
plot(w4/pi,(unwrap (arg (H4))*360/(2*pi)),'r');
hold on;
plot(w4_high/pi,(unwrap (arg (H4_high))*360/(2*pi)),'g');
title('Faasikarakteristikud \Phi(\omega)');
ylabel(strcat('Faas [',char(176),']'));
xlabel ("Normaliseeritud sagedus [\Pi*rad]");
legend ({strcat('Faas (',char(176), '-ides): Madalp', char(228), char(228), 'sfilter'), strcat('K',char(245),
'rgp', char(228), char(228), 'sfilter')}, "location", "east");
hold off;
```

Lisa 7. DMA-kontrollerite poolt kasutatavad perifeeriaseadmed ja mälu jaotus [12].

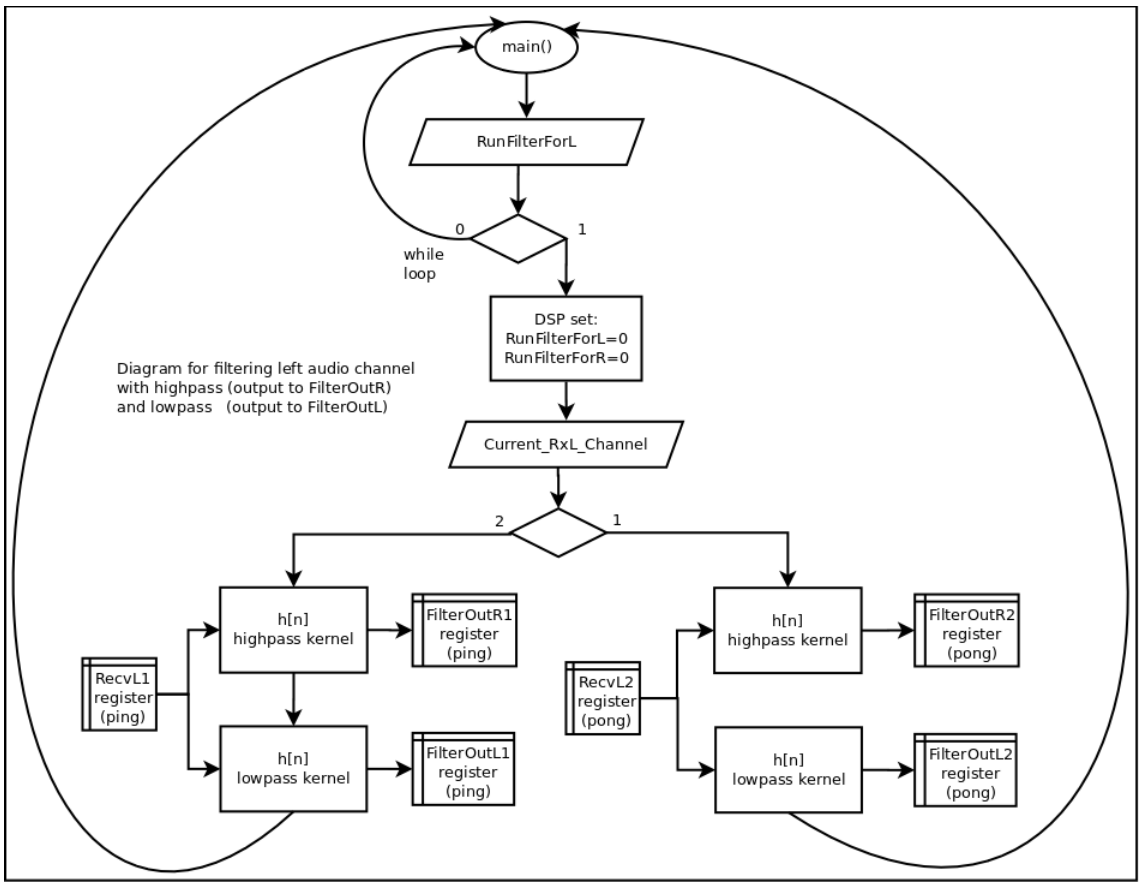
DMA Start *Byte* Address	CPU Start *Word* Address (I/O Space)	CPU Start *Word* Address (Data Space)	DSP Memory Map	DMA Controller 0 Memory Map	DMA Controller 1 Memory Map	DMA Controller 2 Memory Map	DMA Controller 3 Memory Map
0000 2800h	00 2800h	-	I2S0	I2S0	Reserved	Reserved	Reserved
0000 3A00h	00 3A00h	-	MMC/SD0	MMC/SD0			
0000 3B00h	00 3B00h	-	MMC/SD1	MMCSD1			
0001 0000h*	-	00 0000h*	DARAM	DARAM	DARAM	DARAM	DARAM
0009 0000h	-	00 8000h	SARAM	SARAM	SARAM	SARAM	SARAM
0000 1B00h	00 1B00h	-	UART	Reserved	UART	Reserved	Reserved
0000 2A00h	00 2A00h	-	I2S2		I2S2		
0000 1A00h	00 1A00h	-	I2C		Reserved	I2C	
0000 2B00h	00 2B00h	-	I2S3			I2S3	
0000 2900h	00 2900h	-	I2S1			Reserved	I2S1
0000 7000h	00 7000h	-	10-bit SAR			10-bit SAR	Reserved
0200 0000h	-	40 0000h	EMIF CS2			Reserved	EMIF CS2
0300 0000h	-	60 0000h	EMIF CS3				EMIF CS3
0400 0000h	-	70 0000h	EMIF CS4				EMIF CS4
0500 0000h	-	78 0000h	EMIF CS5				EMIF CS5

Lisa 8. I2S plokkskeem [15].

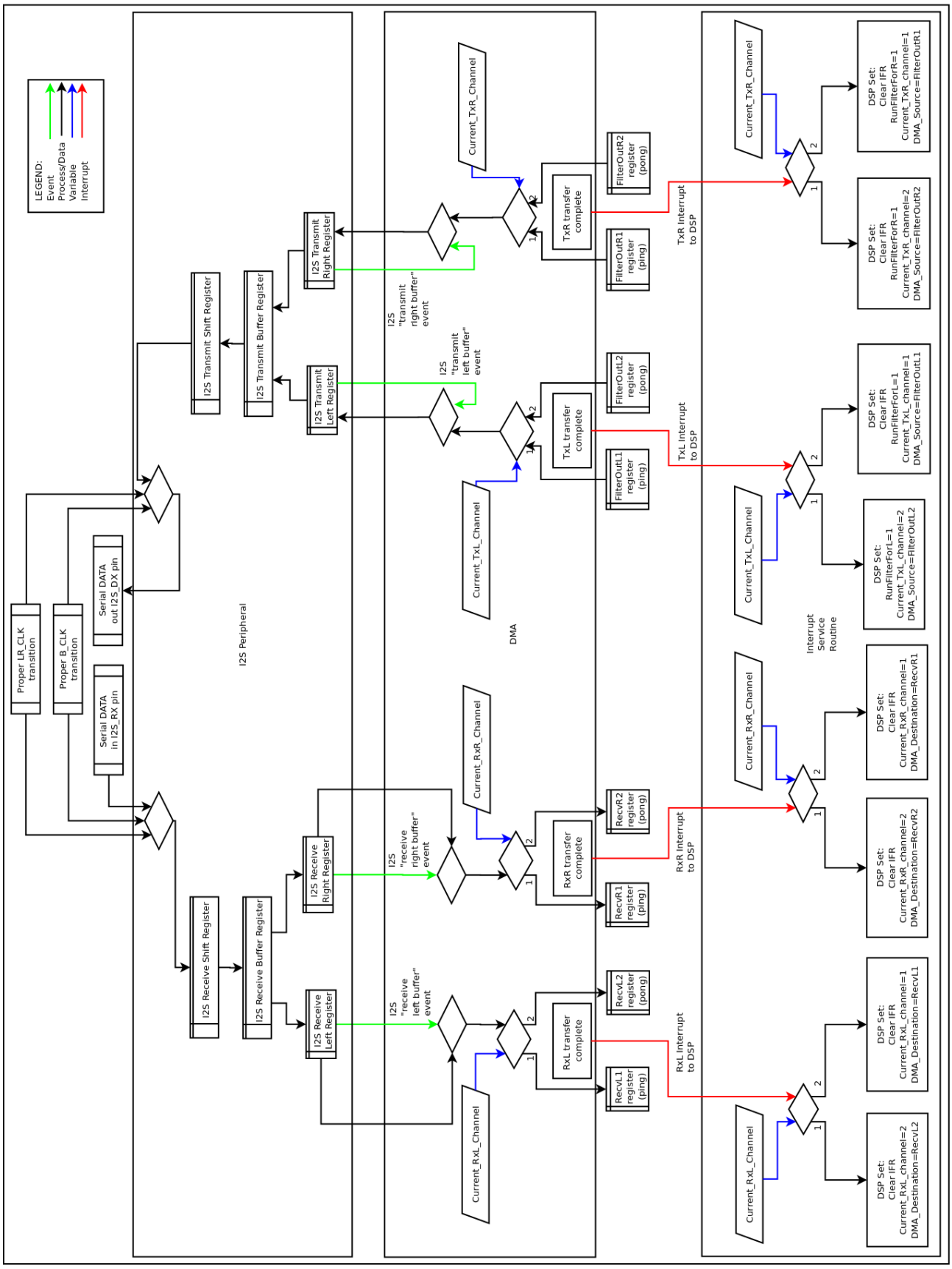


Lisa 9. Tarkvaralise lahenduse graafiline esitus.

Lisa 9.1 Stereosignaali vasaku kanali filtreerimise illustratsioon



Lisa 9.2 I2S-i, DMA ja DSP vahelised protsessid



Lisa 10. Platvormi töövõimelisuse hindamiseks koostatud programm-käsud (MatLab)

```
FsamplingADC=100.0E3
Tmoote=1.0          % võendamine 1s jooksul
xsampleperTrigger= Tmoote*FsamplingADC;
x=daqhwinfo('nidaq'); %   InstalledBoardIds:
ai = analoginput('nidaq','Dev1');
addchannel(ai,0:1); %AI2
set(ai,'SampleRate', FsamplingADC);
set(ai,'SamplesPerTrigger',xsampleperTrigger);
set(ai,'InputType','SingleEnded')%NonReferenced Differential
%set(ai,'InputOverRangeFcn','-1.2 1.2')
set(ai, 'TriggerType', 'Immediate'); %'software' ,'Immediate'
start(ai) %käivita analoog sisend
data = getdata(ai);
out1=(data (1:100000,1))  %bypass
out2=(data (1:100000,2))  %highpass
%---------------
Fs = 100000;                % Sampling frequency
T = 1/Fs;                   % Sample time
L = 100000;                 % Length of signal
t = (0:L-1)*T;              % Time vector

figure(1)
plot(Fs*t(1:1000),out1(1:1000))
title('Väljundsignaal y1(n) aja-vallas')
xlabel('Võendi nr')
ylabel('Amplituud')

figure(2)
plot(Fs*t(1:1000),out2(1:1000))
title('Väljundsignaal y2(n) aja-vallas')
xlabel('Võendi nr')
ylabel('Amplituud')

NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y1 = fft(out1,NFFT)/L;
Y2 = fft(out2,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);
figure(3)
% Plot single-sided amplitude spectrum.
plot(f,2*abs(Y1(1:NFFT/2+1)))
title('Väljundsignaali y1(f) amplituudi spekter')
xlabel('Sagedus (Hz)')
ylabel('|Y1(f)|')
axis([0 24000 0 inf])
```

```
figure(4)
% Plot single-sided amplitude spectrum.
plot(f,2*abs(Y2(1:NFFT/2+1)))
title('Väljundsignaali y2(f) amplituudi spekter')
xlabel('Sagedus (Hz)')
ylabel('|Y2(f)|')
axis([0 24000 0 inf])

figure(5)
H=(2*abs(Y1(1:NFFT/2+1)))./(2*abs(Y2(1:NFFT/2+1)));
plot(f,H)
title('Süsteemi ülekandefunktsioon')
xlabel('Sagedus (Hz)')
ylabel('|Y1(f)| / |Y2(f)|')
axis([0 5000 0 1.5])
```