TARTU ÜLIKOOL
LOODUS- JA TEHNOLOOGIATEADUSKOND
TEHNOLOOGIAINSTITUUT

Koit Summatavet

FPGA kasutamine analoogheli digitaliseerimiseks ja töötlemiseks

Bakalaureusetöö

Juhendaja: Margus Rosin, MSc

Konsultant: Fred Valk, MSc

Kaitsmisele lubatud .....

Juhendaja .....

allkiri, kuupäev

Tartu 2013

Sisukord

SISSEJUHATUS

Muusika loomine ja kirjutamine on alati seisnenud selles, et autor peab noot noodi haaval heliteose noodipilti kirjutama, mis on väga aeganõudev töö. Enne 15. sajandit kirjutati kõik noodid ükshaaval käsitsi kasutades selleks paberit ja tindiga sulge. Esimene masinaga prinditud muusikapala ilmus 20 aastat pärast trükipressi leiutamist J. Gutenbergi poolt (Reese 1934). Tollal oli trükkimine aeganõudev protsess ning vajas mitut trükkimise protseduuri enne, kui kõik detailid sai noodipildile kanda. Kui trükimasin oli 20. sajandi alguseks saavutanud standardse disaini (Polt 1995), loodi 1936. aastal Robert H. Keatoni poolt muusika trükimasin, mis oli esimene trükimasin loodud spetsiaalselt nootide trükkimiseks noodipildile (Keaton 1936).

Projekti idee kasvas algselt välja nii uurimishuvist kui ka praktilisest elukogemusest. Projektiga tegelema innustas küsimus, kuidas olemasoleva tehnoloogia abil saaks muusikat luues ka ise kergema vaevaga heliteoseid noodijoonestikule digitaliseerida. Ka õpingute käigus tundma õpitud FPGA arendusplaadi võimalused tundusid inspireeriva väljakutsena.

Bakalaureusetöö eesmärgiks on muuta muusika kirjutamine automaatseks, kasutades kaasaegseid tehnoloogilisi võimalusi selleks, et autor saaks keskenduda muusika mängimisele ja kogu töö seoses nootide saamisega noodipilti sooritatakse automaatselt. Töötlemiseks ja tuvastamiseks ei kasutata varem lindistatud valmis lugu, vaid reaalajas võetakse mängitavast loost lugemeid. Seejärel mainitud lugemeid digitaliseeritakse ja töödeldakse signaalitöötluse meetoditega, et leida mängitud noodid, mida kuvatakse digitaalsel noodipildil.

Uurimustöö on jaotatud kaheks osaks, millest mõlemas kaetakse FPGA põhifunktsionaalsus. Käesolev bakalaureusetöö tegeleb analoogheli sisselugemise ja selle töötlemisega. Lõpptulemusena leitakse heli sagedus ning selle esitamise kestvus. Teine osa uurimustööst kannab nime „FPGA kasutamine analoogheli digitaliseerimiseks ja visuaalseks kuvamiseks“, mida teostas kaastudeng Andres Randmaa. Teises osas tegeletakse tuvastatud noodi mällu salvestamisega ning seejärel kõikide tuvastatud ja salvestatud nootide kuvamisega kuvaril läbi VGA liidese.

Käesoleva bakalaureusetöö tulemuse saavutamiseks on kasutatud Digilent Inc. poolt loodud Basys2 arendusplaati, mida tutvustati õppeainetes digitaalne loogika (LOFY.03.009) ja intensiivkursus digitaalses tehnoloogias (LOTI.05.018). Basys2 arendusplaadile lisaks on kasutatud lisatavat PmodMic moodulit, mille ülesandeks on heli (näiteks kitarril mängitud noodi) sisselugemine ja seejärel analooghelisignaali muundamine digitaalsele vormile, kasutades moodulil leiduvat analoog-digitaalmuundurit. Heli võib tulla ka heligeneraatorist, aga tähtis on ühe kindla noodi olemasolu. Teise osana käesolevas bakalaureusetöös tegeletakse digitaalse signaalitöötluse protseduuridega ning tuvastatakse sisse loetud heli kõrgus ja selle mängimise kestvus.

1. TEOREETILINE ÜLEVAADE

1.1. Heli füüsikalised alused

Inimene tunnetab heli kõrva välise pinnaga, tuvastades ka väiksemaid õhurõhu muutusi (Lu 2006). Taolisi intensiivseid muutusi on võimalik kõlaritega reprodutseerida või mikrofoni kasutades lindistada. Muutused loovad mustreid, mis viitavad heli 3 põhiomadusele: helikõrgus, tämber ja dünaamika.

David Lu järgi on taolised mustrid tsükliliste omadustega, võnkudes kindla sagedusega. Helikõrguse saame kindlaks määrata signaali põhisagedusega. Helikõrgus väljendab seda, kuidas noot inimeste jaoks kostub.

Tämbrite erinevus tuleneb pillide omaduste erinevusest, mis tähendab, et lisaks põhisagedusele on lisandunud ka teisi sagedusi. Kõige intensiivsemad teised sagedused esinevad kordsete põhisageduste korral. Taolisi sagedusi nimetatakse harmoonikuteks. Näiteks mängides A nooti, mille sagedus on 440Hz, siis selle esimene harmoonik on kahe kordse sagedusega – 880Hz, teine 1320Hz, mis on põhisagedusest kolm korda suurem (vt Lu 2006: 2).

Dünaamika väljendab pilli mängimise valjust hõlmates lööke ja hajumist. See tähendab, et kuigi noodid on mängimise alguses valjud, siis ajapikku nad hajuvad (*Ibid*.).

1.2. Muusika tuvastamise taust

Muusika automaatseks tõlgendamiseks nimetatakse protsessi, kus programm loeb heli lindistust ning väljastab tuvastatud noodid ja nende kõlamise kestvuse (Lu 2006). Polüfoonilise muusika korral peab tuvastama samaaegselt rohkem kui ühe noodi, mis on kas ühe või mitme pilli peal mängitud.

Muusika tõlgendamine on seniajani olnud keskmisest keerulisem väljakutse. Vaatamata mitmetele katsetele antud probleemi lahendada, pole siiani leitud praktiliselt

rakendatavat ja üldotstarbelist tõlgendamise süsteemi (Klapuri 2004). Suudetud on mitmete lähenemiste abil leida kuni kolm erinevat samaaegset heli (Lu 2006).

Helikõrgusi on suudetud hinnata kahaneva protsessiga, tuvastades noote ükshaaval ja lahutades neid maha sageduse spektrist (de Cheveigné, Kawahara 1999). Antud meetodi probleemiks osutusid heli väikesed hälbed, millest tulenesid ebatäpsed lahutamised. Ebatäpsused mõjutasid oluliselt kogu tõlgendamistulemust.

Samaaegselt 3 heli suudeti tuvastada kasutades tõenäosuse ja statistika Bayes' mudelit, mis kirjeldab igal ajahetkel noodi komponente: põhitoon, hormoonikud ja amplituud (Davy, Godsill 2003).

Meetodiga, millega arvati iga võimalik põhisagedus, oli võimalik leida täpne tõlge ligikaudu 80% juhtudest (Goto 2004).

Mitmed kasutatud meetodid ja lähenemised muusika tõlgendamiseks tuginevad signaalitöötlusele ehk uuritakse heli sageduse spektrit, tehakse matemaatilisi otsuseid ja jätkatakse sealt (Lu 2006; vt Christensten, Strong, Palmer 1974; Nicolae, Rugina, Vasile 2000; Tan, Karnjanadecha 2003; Rabiner, Schafer 2011; Wong 2011).

M. S. Manalo ja A. Ashrafi uurimustöös disainiti MATLAB programmis digitaalne filter ja seejärel selgitati protsessi, kuidas teisendada disain VHDL riistvarakirjelduskeelde implementeerimaks filtrit FPGA riistvaral (Manalo, Ashraf 2012).

Teine võimalus on kasutada geneetilist algoritmi (*genetic algorithm*), mis loob generatsioonidena (*generations*) erinevaid tõlgendamise võimalusi (Lu 2006).

Käesoleva uurimustöö puhul on töö esimeses eksperimentaalses pooles toetutud Dubnowski, Schafer ja Rabineri (1976) uurimustööle. See meetod on mõeldud hääle töötlemiseks ning seda on võimalik rakendada ka käesolevas bakalaureusetöös, mille eesmärk on tuvastada lühikese aja jooksul esinev helikõrgus. Nimetatud uurijad lõid kõrgekvaliteedilise helikõrguse tuvastamise süsteemi. Bakalaureusetöös kasutan nende keskmestamise ja autokorrelatsiooni meetodeid ning samu lugemite võtmise parameetreid. Töö teises eksperimentaalses pooles kasutan Fourier' teisendust helikõrguse tuvastamiseks.

1.2.1. Helikõrguse tuvastamine

Helikõrguse tuvastamine on antud uurimustöös üheks kriitilise osatähtsusega ülesandeks. Põhitoon (edaspidi: F0) on helikõrguse tuvastamiseks peamine, aga sellega seoses on väga raske luua usaldusväärset statistilist mudelit. Probleemiks osutuvad tekkivad vead helikõrguse hinnangus ja seosetus põhitooni piirides (vt Tan, Karnjanadecha 2003). Seetõttu on vaja töökindlat helikõrguse tuvastamise süsteemi.

Helikõrguse tuvastamise algoritmid kasutavad lühiajalisi analüüsi tehnikaid. L. Tan ja M. Karnjanadecha (2003) kasutavad meetodit, mis seisneb iga valimi lugemi kohta tehtud autokorrelatsiooni tulemusest kõige suurema väärtuse leidmisele. Autokorrelatsioon seisneb signaali muutmises helilaine ehituse illustreerimiseks. Kui helikõrguse tuvastamises eeldame, et lugem *x(n)* (helisignaali lugemite kogum, kus *n* kirjeldab, mitmes lugem kogumist) on perioodiline perioodil P, ehk $x(n) = x(n + P)$ kõikide *n* puhul, siis on näha, et $Rx(m) = Rx(m + P)$, kus *Rx* on autokorrelatsiooni tulemuse lugem. Sellest võib järeldada, et autokorrelatsioon on sama intervalliga perioodiline. Autokorrelatsiooni perioodilisus väljendab ka signaali perioodilisust (Tan, Karnjanadecha 2003: 551).

Muusika mängimisel on helikõrgus ajaliselt pidevalt muutuv, seetõttu on otstarbekam kasutada lühiajalist autokorrelatsiooni funktsiooni, mis töötleb signaali lühikesi lugemite kogumeid. Võttes signaalist 300 lugemit sagedusel 10kHz (tsüklit sekundi kohta), saame helisignaalist 30ms pikkuse lõigu. Antud lõiku autokorrelatsiooniga töödeldes on võimalik leida heli sagedus vahemikus 50Hz - 500Hz (sagedusel 10kHz loeme lugemi iga 0,1ms tagant, järelikult autokorrelatsiooni lugemi 20 korral on tuvastatav sagedus 1/(20 x 0,1ms) = 500Hz ja autokorrelatsiooni lugemi 200 korral on tuvastatav sagedus 1/(200 x 0,1ms) = 50Hz) sel juhul, kui arvestame, et esimesed 20 autokorrelatsiooni tulemuse lugemi väärtust ei saa arvestada ja alates lugemist 200 on signaal liiga hajunud. Põhjuseks on lugemi korrutamine iseendaga, mistõttu esimesed 20 väärtust on alati suuremad kui teised ja ei väljenda tegelikku heli sagedust (Tan, Karnjanadecha 2003: 551–552).

1.2.2. Autokorrelatsioon

Dubnowski, Schafer ja Rabiner kasutasid M. M. Sondhi (1968) poolt pakutud autokorrelatsiooni meetodeid. Siinkohal kasutab autokorrelatsioon keskmestamise (*center clipping*) ja lõputu kasvuga haripunkti (*infinite peak*) meetodit, mis lävestab (*threshold*) signaali ühtedeks ja nullideks ning hindab neid. Antud meetod vähendab suuresti arvutuslikku keerukust, andes samal ajal piisavalt täpseid tulemusi (Dubnowski, Schafer, Rabiner 1976: 2–3).

Esmalt töödeldakse sisenev signaal keskmestamise meetodiga. Selle tulemusena eemaldatakse üleliigne signaali osa. Keskmestamine vähendab signaali müra ning säilitab sellest olulise osa. Lisaks vähendab süsteemis talletamiseks vajalikkude ressursside kasutamist. Seejärel keskmestamisega saadud signaaliga sooritatakse autokorrelatsioon, mis seisneb signaali lugemi korrutamises iseendaga. Autokorrelatsiooni tulemusena on saadud järjend, mis koosnes signaali sageduste spektrist. Signaali sageduse spektri lugemid väljendavad helisignaalis esinenud sagedusi. Kõige suurema väärtusega lugem määrab ära siseneva heli sageduse ehk helikõrguse (vt Dubnowski, Schafer ja Rabineri uurimust, kus nad kasutasid M. M. Sondhi poolt 1968. aastal pakutud ettepanekuid).

1.3. Kasutatud mõisted

FPGA - *Field Progammable Gate Array* – taas programmeeritav integraalskeem, mis koosneb kolmest peamisest komponendist: sisend/väljundviigud, sisemised ühendused ja loogikaplokid. Kui teiste sarnaste tehnoloogiate puhul saab skeemi programmeerimine toimuda ainult tehases, siis FPGAd saab lõpptarbija korduvalt programmeerida (Rosin 2009: 5).

VHDL – *VHSIC Hardware Description Language* – tegemist on riistvarakirjelduskeelega, mis on standardiseeritud IEEE (Institude of Electrical and Electronic Engineers) poolt kui IEEESTD-1076 ja hilisem versioon IEEE-STD-1164. VHDLi saab kasutada nii sünteesimiseks kui ka simuleerimiseks (*Ibid.*).

Autokorrelatsioon – signaali lugemi korrutamine iseendaga. Autokorrelatsiooni tulemusena saame järjendi, mis koosneb signaali sageduste spektrist (Dubnowski, Schafer, Rabiner 1976).

Keskmestamine (*center clipping*) – signaali silumise meetod, millega vähendatakse signaali väärtused lävendi võrra (vt Dubnowski, Schafer, Rabiner 1976).

Fourier' teisendus – algoritm heli- jt. signaalide sageduskomponentide (spektri) väljaarvutamiseks (Vallaste 2013).

Hamming aknafunktsioon – aknafunktsiooni tööks on signaali silumine, millega eemaldatakse järsud üleminekud.

2. UURIMUSTÖÖ ETAPID JA MEETODID

1. Teoreetiline osa: tutvumine teaduskirjandusega (võtmesõnad: *autocorrelation algorythm*, *center clipping*, *infinite peak clipping*, *clipping threshold*, PmodMic, FPGA, VHDL).
2. Eksperimentaalne uurimustöö:
   1) Heli sisselugemine ja analoogsignaali töötlemine digitaalseks kasutades PmodMic moodulit. Selleks on loodud VHDLis kontroller, mis loeb sisse jadamisi analoog-digitaalkonverteri sisendit, andmed paigutatakse 12-bitisesse vektorisse.
   2) Signaali keskmestamine ja lävestamine ning seejärel autokorrelatsiooni kasutamine signaali lugemite töötlemiseks. Autokorrelatsiooni tulemuse lugemite analüüsimine helikõrguse ja kestvuse tuvastamiseks. Saadud tulemus edastatakse signaali kujul.
   3) Signaali lugemite silumine Hamming aknafunktsiooniga, A noodile kuuluva siinuse väärtuste leidmine, teistele nootidele kuuluva siinuse väärtuse leidmine, Fourier'i pööre signaali lugemite ja tuvastavale noodile kuuluva siinuse lugemitega. Suurima väärtuse korral on tegu vastava noodiga. Ühe noodi asemel kontrollitakse paralleelselt väärtust mitme noodi vahel. Noodi helikõrgus edastatakse signaali kujul.

Uurimustöös kasutatakse VHDL riistvarakirjelduskeele Xilinx ISE 14.4 tarkvara ja Digilent Inc. Basys2 FPGA arendusplaati koos lisamooduliga PmodMIC.

3. KASUTATAV RIIST- JA TARKVARA

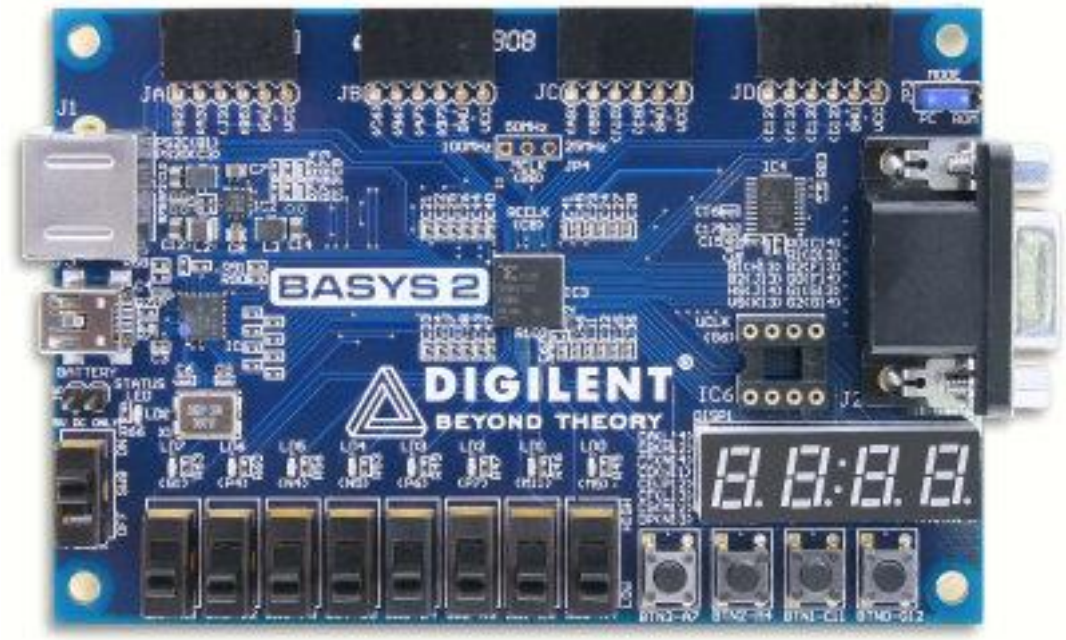
3.1. Riistvara

Käesoleva bakalaureusetöö realiseerimiseks kasutati Digilent Inc. poolt toodetud Basys2 (vt Joonis 1), mis sisaldab Xilinx Spartan 3E xc3s250e FPGAd. Väliseks lisandiks Basys2 vahendile on PmodMIC moodul.

3.1.1. Basys2

(vt Basys2 käsiraamat)

- Xilinx Spartan3E -250 FPGA CP132;
- 18-bitised korrutid, 27 kilobaiti (216 kibibitti) mälu ning tegu on kiire kahepordilise plokkmäluga, lisaks toetab süsteem taktsagedusi üle 500MHz;
- Täiskiirusega USB 2.0 välissiini port FPGA konfigureerimiseks ja andmevahetuseks;
- XCF02 välkmälu ROM, milles jäädavalt salvestatakse FPGA seadistused;
- Kasutaja poolt valitav taktsignaali generaatori sagedus (25, 50 ja 100MHz). Lisavõimalusena saab lisada välise taktsignaali generaatori;
- 3 plaadi peal paiknevat pingeregulaatorit (1.2V, 2.5V ja 3.3V), mis võimaldavad kasutada väliseid toiteallikaid pingevahemikus 3.5V – 5.5V;
- 8 valgusdioodi, 4-kohaline ja 7-segmendilise numbriga ekraan, 4 sisendnuppu, 8 sisendlülitit, PS/2 port ja 8-bitiline VGA väljund;
- 4 x 6 klemmiga pistikud Digilent Inc. Pmod moodulite lisamiseks;
- Tarkvara saab laadida nii üle USB kui ka välkmälust;
- Pingestada saab nii USB pordist, adapterist kui ka akust;



Joonis 1. Illustreeriv joonis Basys2 vahendist

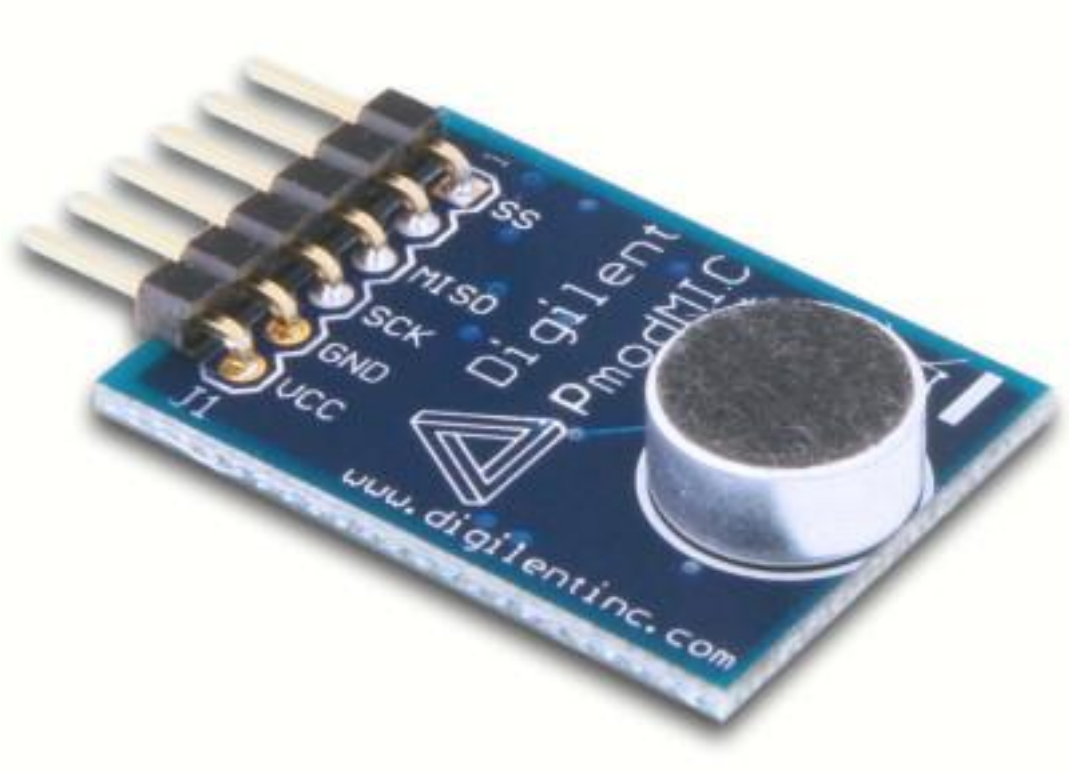
3.1.2. PmodMIC

(vt PmodMic moodul Digilent Inc. kodulehel)

- OnSemi SA575 madala pingeline *compander* (süsteem, mis parandab signaali ja müra suhet), mille kasutamine on signaalitöötluse jaoks suurepärane, kuna signaal on silutud ja esinev müra on minimaalne.
- ADCS7476AIM 12-bitine analoog-digitaalmuundur
- Tööpinge vahemikus 3V-5V
- Suurus tollides 0.80“x1.1“

Antud moodul (vt Joonis 2) loeb esmalt mikrofonist analoogse helisignaali, mis läbib eelvõimendi ning seejärel konverteeritakse 12-bitiseks digitaalseks andmesignaaliks. 12-bitise analoog-digitaalmuunduri resolutsioon on 4096 (2^{12}) . Ühendustega J1 ja J18 saab moodul plaadilt toitepinge väärtuses 3.3V ja maa ühenduse (GND). Analoog-digitaalmuunduri jaoks on vaja genereerida taktsignaal sagedusega 12.5MHz. Selle saamiseks kasutatakse Basys2 plaadi taktsignaali sagedusega 50MHz. Esimese viigu, märgistusega SS, külge on ühendatud mooduli kiibi valiku signaal nCS (not Chip Select, aktiivse signaali korral on kiip mitteaktiveeritud ja mitteaktiivse signaali korral

on kiip aktiveeritud). Teine viik on signaalita. Kolmas viik, märgistusega MISO, vahetab moodul analoog-digitaalmuunduri järjestikandmeid. Neljanda viiguga, märgistusega SCK, jagab Basys2 plaat taktisignaali. Viienda viiguga, märgistusega GND, on ühendatud moodul maaga. Kuuenda viiguga, märgistusega VCC, on ühendatud moodul toitega. Basys2 I/O port JA viigud on järgmised: esimene viik B2, teine viik A3, kolmas viik J3, neljas viik B5, viies viik GND ja kuues viik VCC.



Joonis 2. Illustreeriv joonis PmodMIC moodulist

3.2. Tarkvara

3.2.1. Xilinx ISE 14.4

(vt Xilinx ISE WebPACK 14.4.)

FPGA programmeerimiseks kirjutatakse kood Xilinx ISE tarkvara versiooniga 14.4 WebPACK. Xilinx ISE on ideaalne lahendus FPGA programmeerimiseks võimaldades HDL sünteesimist ja simuleerimist, implementeerimist, seadmele paigutamist ja JTAG

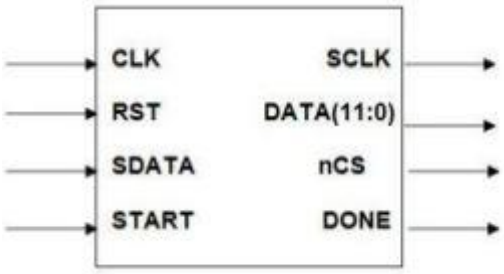
liidese abil programmeerimist. Xilinx ISE WebPACK on tasuta tarkvara, mille saab alla laadida Xilinxi kodulehelt. Toetatud on järgnevad operatsioonisüsteemid: Winsows XP, Windows 7, Windows Server 2008, Red Hat Enterprise Linux 5/6 ja SUSE Linux Entreprise 11. Hetkel esineb probleeme Windows 8 keskkonnas, kus mõned tarkvara osad töötavad 32-bitise ning ülejäänud 64-bitise operatsioonisüsteemiga, kuid mitte ühes versioonis korraga. Käesolev bakalaureusetöö programmeeriti Windows 7 Home Premium 64-bitise operatsioonisüsteemi peal. Programmi koodi on võimalik kirjutada VHDL või Verilog riistvarakirjelduskeeles. Xilinx ISE töökeskkonnas toetatakse ka MATLABis tehtud projektide kaasamist. Xilinx ISE tarkvaraga on võimalik lasta oma kood üle kontrollida võimalike vigade vältimiseks, teostada ühildamine kasutatava riistvaraga, viia läbi FPGA loogikaplokkide valik ning luua nendevahelised ühendused ja seejärel luua FPGA kivile loetav kood .bit formaadis. Eelnevalt mainitu saab kivi peale kirjutada Digilent Inc. poolt spetsiaalse tarkvaraga – Adept või Xilinx ISE-sse sisseehitatud ISE iMPACT tarkvaraga. Adept on Digilent Inc. kodulehelt allalaetav (http://www.digilentinc.com/Products/Detail.cfm?Prod=ADEPT2).

4. PROJEKTI ARENDUS

4.1. PmodMic kasutamine

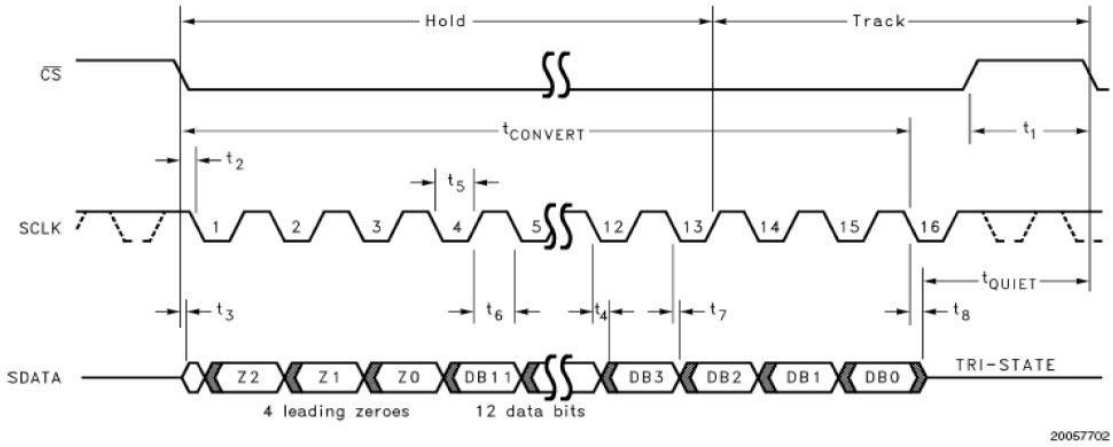
(vt PmodMic)

Plokkdiagrammi mooduli (vt Joonis 3) sisenditeks on 50MHz taktsignaal (CLK), asünkroonne Reset nupp (RST) ja mooduli kiibist ADCS7476 väljuvad andmed. Viimased on igal taktsignaalil jadamisi sisse nihutatud (SDATA). Väljunditeks on SCLK signaal, mis on taktsagedusega 12.5 MHz PmodMic jaoks; kiibi valimise signaal (nCS, inglise keeles *not Chip Select*), mis lülitab sisse PmodMic mooduli ADCS7476 kiibi ja ADCS7476 kiibis genereeritud 12-bitine vektor (DATA), mida saavad kasutada kõik FPGA komponendid. START signaal on komponendile teavitamaks, millal konverteerimist alustada. Kui konverteerimine on lõpetatud, siis aktiveeritakse DONE signaal.



Joonis 3. Plokkdiagramm mooduli sisenditest ja väljunditest

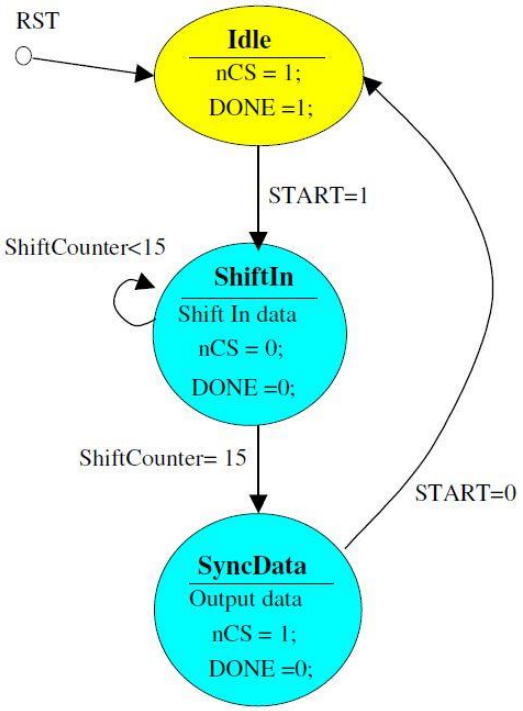
Ajadiagrammi (vt. Joonis 4) kasutatakse, et määrata täpne takteerimise järjekord olekumasinale, mis on PmodMic taktsignaaliks. See on ajaline järjestus, millega genereeritakse 16-bitti andmeid kasutades PmodMic mooduli kiipi ADCS7476. Andmete genereerimisel peab signaal nCS olema madalas või null olekus. Andmeid genereeritakse taktsignaali langeva frondiga. Vahetult pärast andmete ülekannet tuleb nCS signaal kõrgeks määrata, et anda märku uute andmete genereerimiseks.



Joonis 4. PmodMIC mooduli kiibi ADCS7476 signaalide ajadiagramm

PmodMIC mooduli Moore'i olekumasinat (vt Joonis 5) kasutati ajastamise järjekorra loomiseks, et jadamisi vastu võtta analoog-digitaalmuunduri väärtus SDATA ja asetada see 16-bitisesse vektorisse, samuti ka väljundite nCS ja SLCK takteerimiseks.

Kokku on olekumasinal kolm olekut: **Idle** (tühikäigu olek), **ShiftIn** (nihuta sisse) ja **SyncData** (andmete sünkroniseerimine). Tühikäigu (**Idle**) olekus (vt Joonis 5) peab konverteerimise algamiseks olema DONE väljundsignaal kõrges olekus. Väljundsignaal nCS on kõrges olekus, mis määrab ära, et kiip ei ole aktiveeritud olekus. START signaali kõrgeks muutudes läheb olekumasin ShiftIn olekusse.



Joonis 5. PmodMIC mooduli lõplik olekumasin

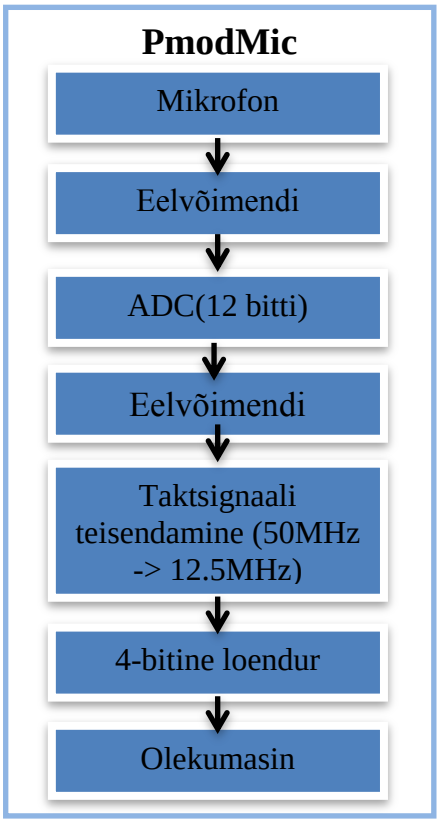
ShiftIn olekus (vt Joonis 5) seatakse DONE signaal madalaks ja PmodMic andmed nihutatakse jadamisi, alates suurima kaaluga bitist kuni väikseima kaaluga bitini ja seda 16 taktsignaali tsükli vältel, et tagada iga kiibi kõigi 16 andmebiti kohale jõudmist. ShiftCouner on 4-bitine (0-15) loendur. Iga nihutamisega suurendatakse ShiftCounteri väärtust, mis omakorda näitab

mitmes nihe on toimunud. Kui selle väärtus on võrdne 15-ga, siis on andmete nihutamine loetud lõppenuks ning olekumasin läheb SyncData olekusse.

SyncData olekus (vt Joonis 5) asetatakse PmodMic moodulist saadud helisignaali lugemid 12-bitisele väljundile DATA. Väljundsignaal seatakse kõrgesse olekusse kirjeldamaks, et kiip ei ole aktiveeritud olekus. Signaal DONE seatakse jäädavalt madalaks, kuna uus konverteerimine ei või veel alata. Kui START sisendsignaal on madal ja helisignaali lugemid on asetatud väljundile DATA, siis läheb masin tagasi tühikäigu olekusse järgmist konverteerimist ootama.

Olekust sõltumatult lähtestab RST sisendsignaal olekumasina ja seab selle tühikäigu olekusse.

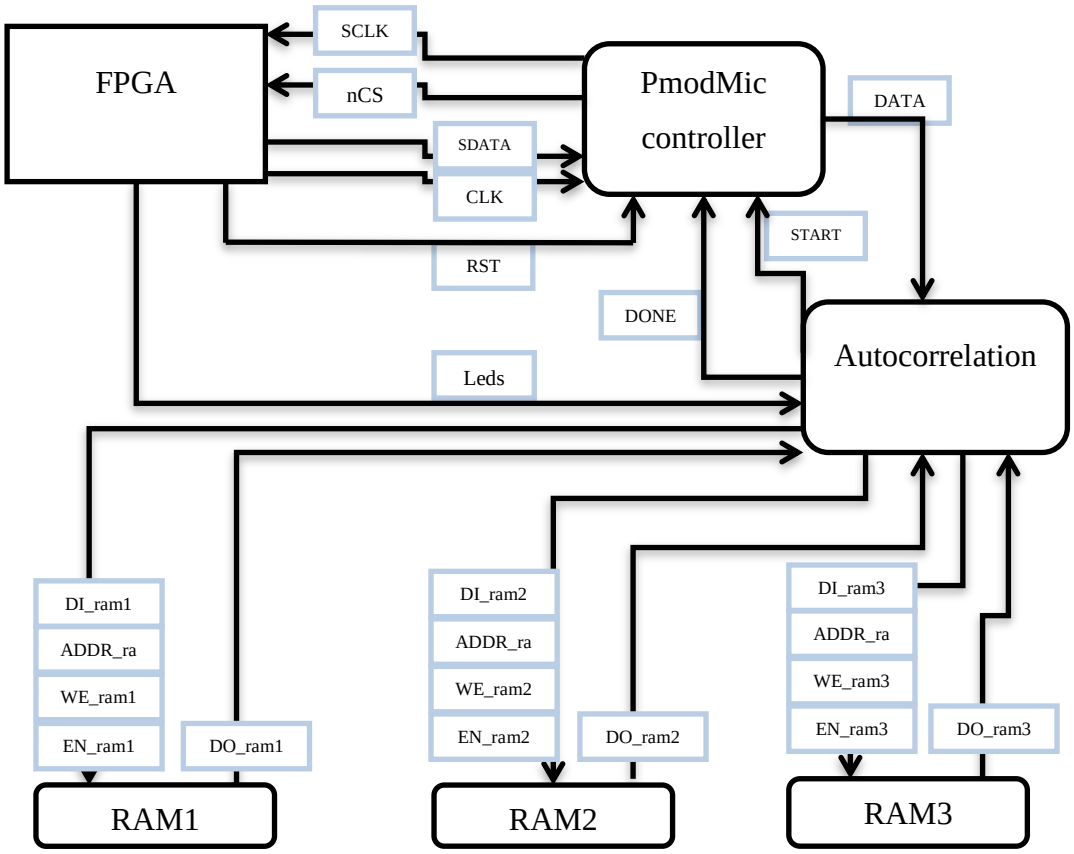
PmodMic mooduli tööpõhimõte seisneb esmalt mikrofoniga heli vastuvõtmises, seejärel analoogsignaal läbib eelvõimendi ning teisendatakse digitaalsele kujule 12-bitise analoog-digitaalmuudnuri abil. Digitaliseeritud signaal läbib samuti eelvõimedit. Analoog-digitaalmuundur töötab sagedusega 12.5MHz, mille saame arendusplaadi kella taktsignaali teisendamisel. Muunduri tulemused võetakse vastu ja tulemuste arvu kontrollimiseks kasutame 4-bitist loendurit. Kogu protsessi juhib olekumasin, mis vastavalt olekule kas võtab vastu muunduri tulemusi või saadab 12-bitise lugemi järgmistele protsessidele (vt Joonis 6).



Joonis 6. PmodMIC mooduli ehitus

4.2. *Signaalitöötlus kasutades autokorrelatsiooni*

Signaalitöötlus, kasutades autokorrelatsiooni koosneb kolmest erinevast moodulist: PmodMic kontroller, heli töötluse ja autokorrelatsiooni moodul ja kolm mälumoodulit (vt Joonis 7).

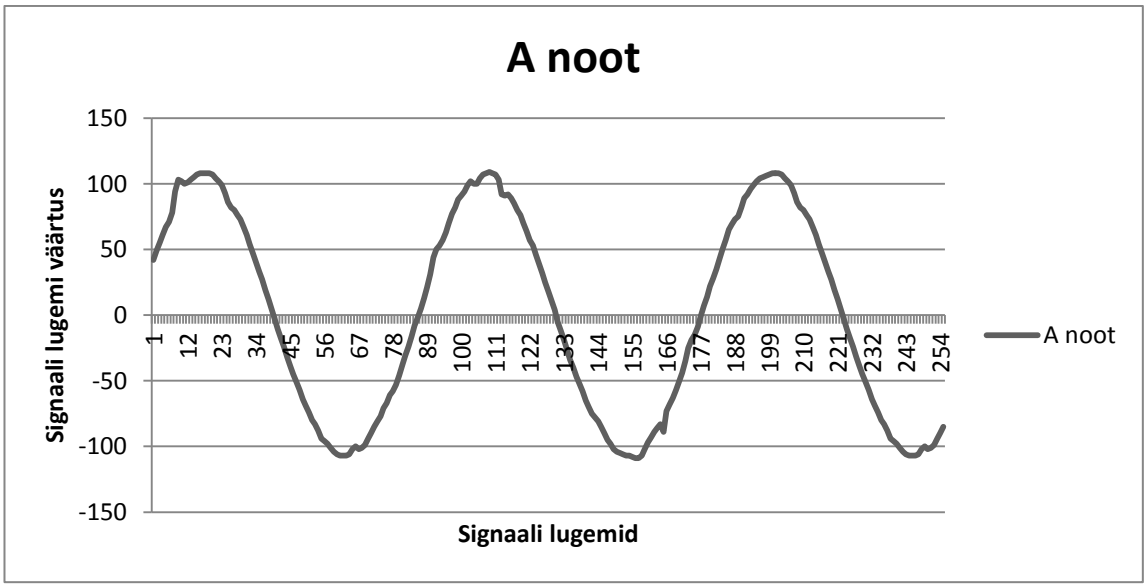


Joonis 7. Signaalitöötlus kasutades autokorrelatsiooni üldine skeem

4.2.1. PmodMic väljundsignaali DATA asetamine järjendisse

Käesoleva uurimustöö eksperimentaalse osa kõige esimese katsena uuriti, kuidas PmodMic moodul töötab ja milline on väljuv signaal. Selleks, et PmodMic signaali loeks ja töötleks analoog-digitaalkonverteriga, peab ära määrama START signaali. START signaal on indikaatoriks, kas uue signaali lugemisega võib alustada. Mainitud signaali võib seada aktiivsesse olekusse alles siis, kui analoog-digitaalkonverter on andnud signaaliga DONE märku, et eelmine signaal on sisse loetud. Analoog-digitaalmuunduri DONE signaali mitteaktiivne olek tähendab, et teisendus pole

lõppenud ja aktiivne olek tähendab, et eelmine teisendus on lõpetatud ja võib alustada uuega. Vastasel juhul on START signaal mitteaktiivses olekus. Signaal START ei pea olema sünkroonne kindla taktsignaaliga, seetõttu muudame seda taktsignaaliga sagedusel 50MHz. Kui START signaal on õigesti seatud, siis saame sisse lugeda 12-bitise analoog-digitaalkonverteri signaali – DATA. Signaal sõltub heli võngetest ning üleliigsest mürast. Vaikses ruumis on sisend 800-900 ühiku vahel. Minimaalne ühik on 0 ja maksimaalne ühik on 4095 (2^{12}). Signaali DATA väärtus on lugem helisignaalist mõõtmise ajahetkel. Et heli analüüsida peame võtma mitu lugemit ning seejärel asetame need kas järjendisse või talletame mälus (BRAM ehk plokk RAM). Nii on võimalik helist võtta lugemeid kasutajale vajalikul kiirusel, näiteks prooviti sagedust 40kHz, et analüüsida A noodi (440Hz) siinust (vt Joonis 8).



Joonis 8. PmodMic-ga A noodi (440Hz) signaali 256 lugemi väärtused sagedusega 40kHz

4.2.2. Keskmestamine

Keskmestamise meetodit kasutatakse uuritava signaali lugemite silumiseks, kaotades kõrge sagedusega signaali osad ja müra. Seega on keskmestatud spektriliselt silutud signaali palju lihtsam mõõta võrreldes silumata signaaliga.

Esimese etapina peab leidma sisse loetud signaali lugemite väärtuste põhjal väljalõike lävendi. Lävendi peab valima nii, et ei kaoks meile olulist osa signaalist. Lävendi arvutamiseks valitakse esimesed ja viimased 1/3 signaali lugemite väärtustest. Esmalt leitakse esimese 1/3 signaali lugemitest maksimaalse signaali väärtus, mis võetakse selle lugemite osa kõige suurema signaali lugemi väärtuse teguriks *max1*. Seejärel leitakse viimasest 1/3 signaali lugemitest maksimaalse signaali väärtus ning selle väärtus määratakse tegurile *max2*. Kui soovitakse leida helikõrguse sagedus vahemikus 50-500Hz lugemite võtmise sagedusega 10kHz, siis selleks võetakse 300 lugemit. Kui 300 lugemit on analüüsitud (seda kontrollitakse loenduri abil), siis leitakse, kumb teguritest on väiksem. Väiksem tegur seatakse väljalõike lävendiks. Kuigi lävendi väärtus on tavaliselt 60-80%, siis antud eksperimendi puhul kasutati selle väärtuse saamiseks teist meetodit. Väljalõige saadi lävendist lahutatud väärtusest, mis saadi suurema ja väiksema teguri väärtuse vahest saadud tulemusest. Probleeme võib tekkida, kui tegurid on võrdsed ning ära kaotatakse vajalik signaali osa. Sel juhul ei ole signaalist võimalik otsitavat helikõrgust tuvastada, kuna seda kirjeldava sageduse osa kaob.

Keskmestamise meetod:

$$y(n) = clc[x(n)] = \begin{cases} (x(n) - C_L), x(n) \geq C_L \\ 0 \qquad\qquad\quad , x(n) < C_L \end{cases} \tag{1}$$

Antud meetodis kasutati eelnevalt arvutatud lävendit – C_L, millega võrreldi signaali väärtusi – *x(n)*, kus *n* on täisarvuline väärtus 0-st 299-ni. Kui signaali lugem on suurem-võrdne lävendist C_L, siis vähendati signaali lugemi väärtust C_L võrra. Kui signaali lugem on väiksem lävendist C_L, siis seatakse selle väärtus 0-ks

4.2.3. Signum funktsioon

Signum funktsiooniga signaali töötlemise meetodi puhul leitakse lõike lävend sarnaselt keskmestamise meetodile. Erinevus on lugemi lävendist suurema väärtuse seadmine. Kui lugemi väärtus on suurem kui lävend, siis lugemi väärtuseks on 1. Kui lugemi väärtus on väiksem kui lävend C_L, siis on lugemi väärtuseks 0.

Signum funktsiooniga signaali töötlemise meetod:

$$y(n) = sgn[x(n)] = \begin{cases} 1 & , x(n) \geq C_L \\ 0 & , x(n) < C_L \end{cases} \quad (2)$$

4.2.4. Autokorrelatsioon

Kui signaali lugemite väärtused on keskmestatud või kasutatud signum funktsiooniga signaali töötlemise meetodit, siis järgmine etapp on signaal lugemite töötlemine autokorrelatsiooni funktsiooniga. Autokorrelatsiooni valem:

$$R(m) = \sum_{n=0}^{N-1-m} x(n) \times x(n+m), \qquad 0 \leq m \leq M_0 \quad (3)$$

Autokorrelatsioon arvutatakse *m* - esialgse viivise ja M_0 - lõpliku viivise põhjal. Need väärtused võib seada vastavalt kasutajale. Käesolevas uurimustöös loetakse 300 lugemit (N), M_0 väärtuseks valime 200 ning *m* väärtuseks 25, mis vastab sagedusvahemikule 50Hz – 500Hz.

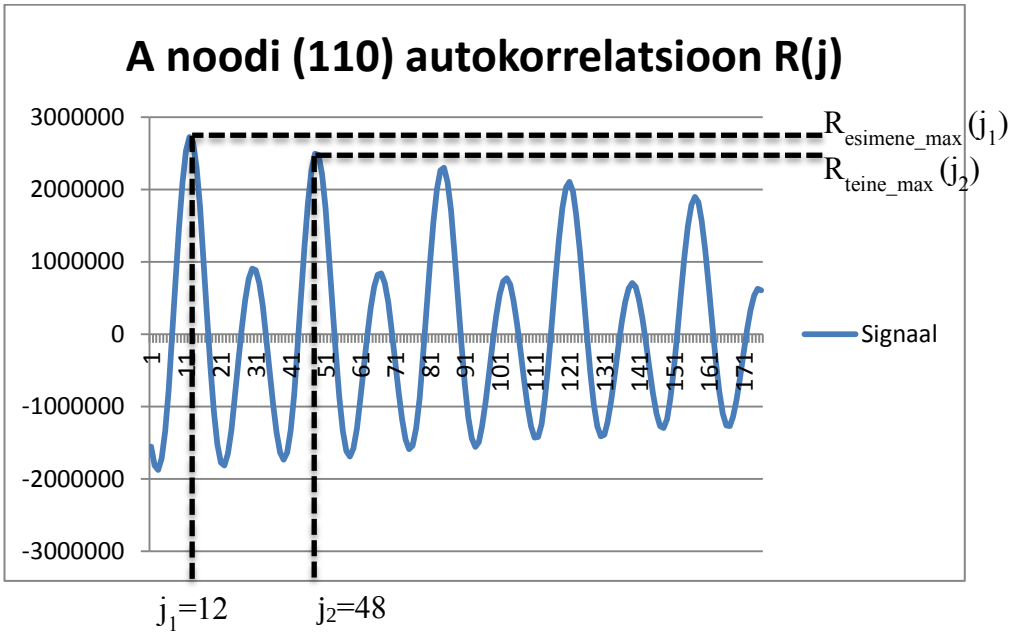
Funktsioonina kirjutatult VHDL-is on autokorrelatsioon järgmine:

```
for m in 24 to N-1-m loop
        for n in 0 to N-1 loop
                if (n + m) < N then
                        R(m) := R(m) + a(n) * a(n + m);
                end if;
                exit when (n + m) = N;
        end loop;
end loop;
```

Antud funktsioonis on näitena kasutatud *for* tsüklit, mis on reaalses programmis asendatud loenduritega, kuna *for* tsükkel nõuab palju rohkem FPGA ressursse. VHDL keeles summa funktsiooni puudumise tõttu kasutati *for* tsüklit. Antud näites kasutatakse muutujat *m*, mis vastab autokorrelatsioonis viivise väärtustele ja muutujat *n*, mis vastab signaali lugemi väärtuse arvule.

4.2.5. Sageduse leidmine autokorrelatsiooni andmetest

Sageduse leidmiseks autokorrelatsiooni andmetest leitakse kaugus esimese maksimaalse ($R_{esimene\_max}$ (j_1)) ja teise maksimaalse (R_{teine_max} (j_2)) autokorrelatsiooni lugemi vahel. Järgnevalt kasutame kitarril mängitud A noodi (vt Joonis 15) signaali ja leiame autokorrelatsiooni kasutades valemit (3) (vt Joonis 9).



Joonis 9. Autokorrelatsiooni andmed ja viivise j_1 ja j_2 leidmine.

Sagedus arvutatakse välja järgnevalt:

$$f = \frac{f_L}{(j_2 - j_1)} \tag{4}$$

Valemis on lugemise sageduse tähis f_L, esimese viivise tähis j_1 ja teise viivise tähis j_2. Antud näites on lugemise sageduseks 4kHz, esimeseks viiviseks 12 ja teiseks viiviseks 48. Kasutades sageduse arvutamise valemit leiame, et sagedus on 111Hz, mis eksib küll ühe hertsiga, aga tulemus on õigele sagedusele (110Hz) väga lähedal.

4.3. Signaalitöötlus kasutades Fourier'i teisendust

4.3.1. Hamming aknafunktsioon

Hamming aknafunktsiooni kasutati signaali lugemi silumiseks. Selleks määrati ära akna väärtused ja hoiti neid järjendis. Olenevalt signaali lugemite kogusest võeti ka akna väärtusi sama arv. Silumiseks korrutati signaali lugemi väärtused akna väärtustega ning tulemust hoiti mälus.

4.3.2. Fourier'i pööre

Fourier'i pööret kasutati helikõrguse tuvastamiseks. Esmalt määrati ära otsitavate nootide siinuselised väärtused. Need väärtused on muutumatud ja hoiti järjendis.

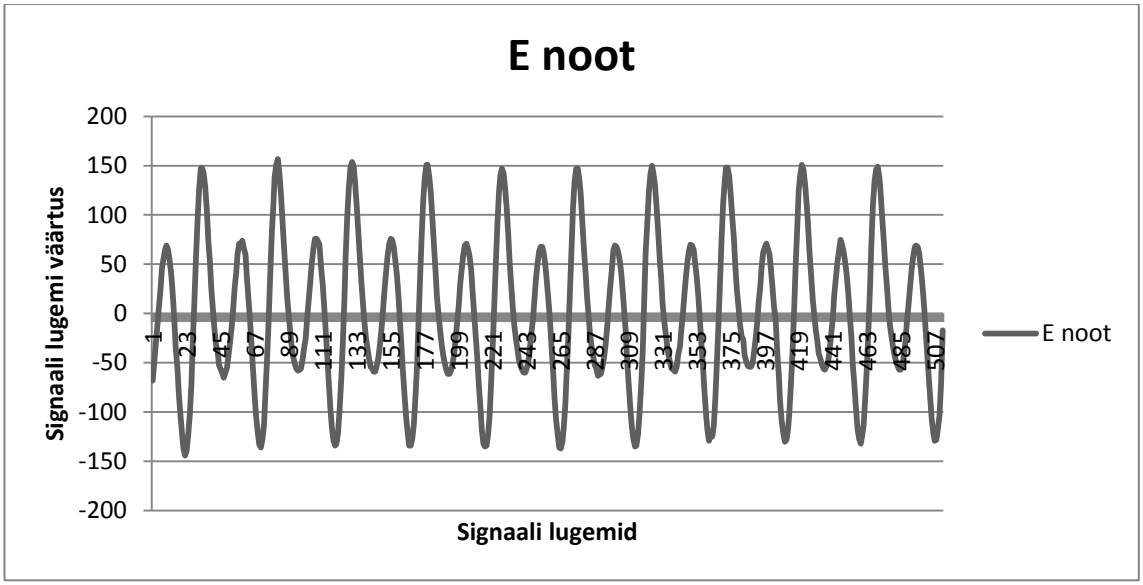
Kui signaali lugemi väärtused olid Hamming aknafunktsiooniga silutud, siis korrutati silutud väärtused nootide siinuseliste väärtustega. Fourier'i pöörde väärtus saadi eelnevalt saadud väärtuste summast. Suurim korrutiste summa väärtus määras ära noodi.

4.3.3. Signaali uurimine

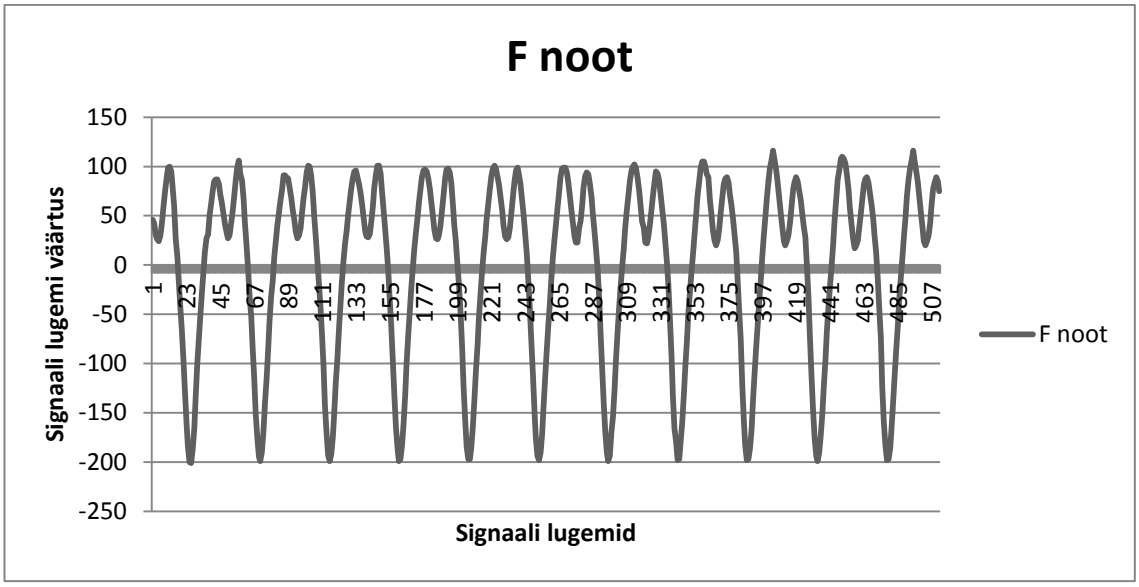
Signaali uurimiseks valiti esmalt lugemise sagedus. Lugemise sagedus peab olema 2 korda suurem maksimaalse tuvastatava signaali helikõrgusest. Kõige kõrgemaks noodiks võeti 6. oktavi B noot, mille sagedus on 1975.53Hz. Järelikult selleks peab olema lugemise sagedus 4kHz. Järgmisena valiti resolutsioon, millega noote eristatakse. Kui on vaja leida teise oktavi E nooti, mille sageduseks on 82.4Hz, ja teise oktavi F nooti, mille sageduseks on 87.3Hz, siis järelikult peame mõõtma resolutsiooniga 4.9Hz või väiksemaga. Et antud resolutsiooni saavutada lugemise sagedusel 4kHz, peab võtma helist vähemalt 816 lugemit (saame lugemise kiiruse jagamisel resolutsiooniga). Kui soovitakse kasutada vähem lugemeid, siis sel juhul peab lugemise sagedust vähendama. Selleks valiti 2kHz lugemise kiiruseks ning 512 lugemit, mis annab resolutsiooniks 3.906Hz. Helikõrguse leidmise täpse tulemuse katsetamiseks mängiti kitarril kuus 2. oktavi nooti: E noot, F noot, F# noot, G noot, G# noot ja A noot.

Lugemite esitamiseks kasutati 7-segmendilist LED kuvarit, millega saab kuvada numbreid 0000-st kuni 9999-ni. Järelikult sobib kuvar ideaalselt 12-bitise signaali kuvamiseks, kuna signaali digitaalne võimalik väärtus on vahemikus 0st kuni 4095ni.

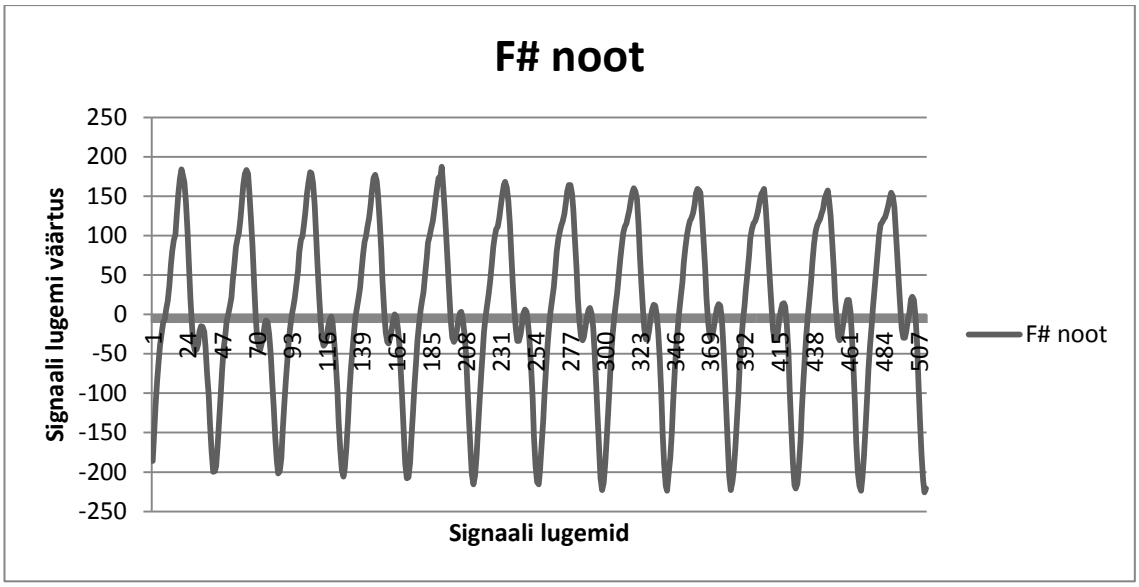
Signaali lugemite väärtuste saamine oli aeganõudev töö. Esmalt luges FPGA 512 signaali lugemit, aga eksimuse tõttu oli lugemise taktsagedus 4kHz, seejärel lugemid talletati järjendisse. Järgmisena loeti ükshaaval lugemite väärtusi ning muudeti need BCD signaaliks, mille abil tõlgendati binaararvuline 12-bitine signaal kuvamiseks vajalikuks 16 bitiseks signaaliks. Ühe arvu kuvamiseks 0-st 9-ni kulub 4 bitti. Vaja on 16 bitti, et kuvada lugemit 4-l 7-segmendilisel kuvaril. Lugemeid kuvati iga sekundi tagant ja seejärel asetati lugemite väärtused käsitsi tabelisse. Signaali lugemid on võetud kitarri peal mängitud nootidest (vt. Joonis 10, Joonis 11, Joonis 12, Joonis 13, Joonis 14, Joonis 15). Seejärel sai lugemeid analüüsida, et leida signaali sagedusi ja tuvastada mängitud helikõrgusi.



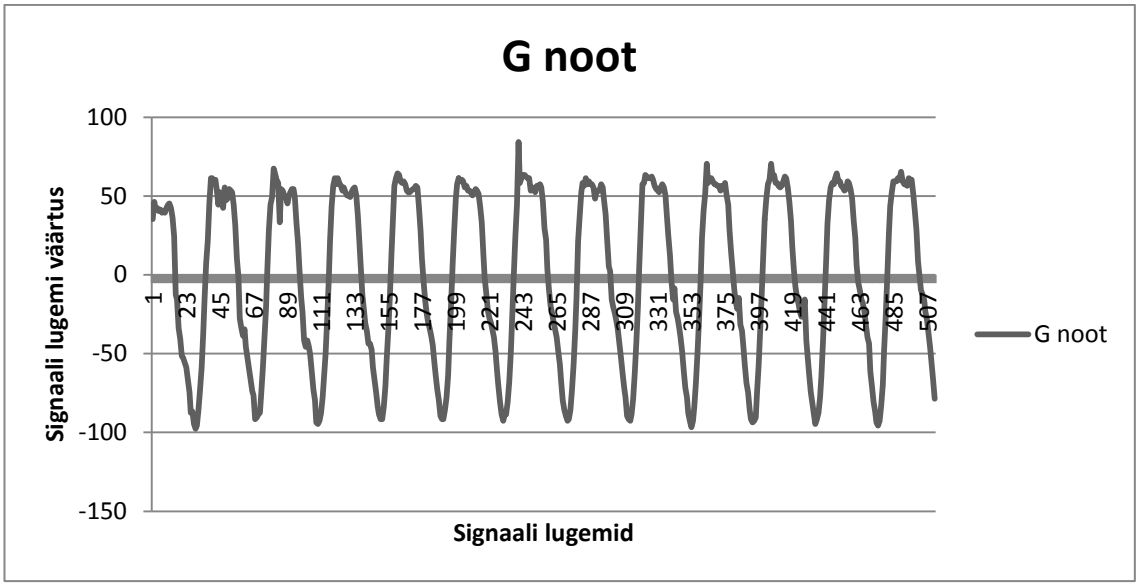
Joonis 10. Kitarri peal mängitud E noodi (82.407Hz) signaali lugemid



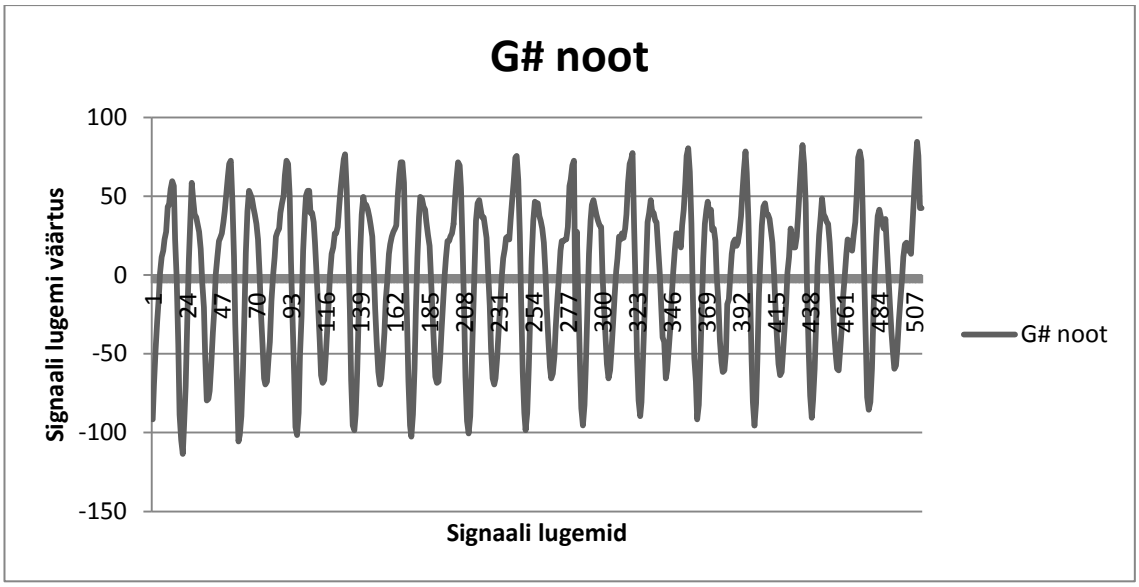
Joonis 11. Kitarri peal mängitud F noodi (87.31Hz) signaali lugemid



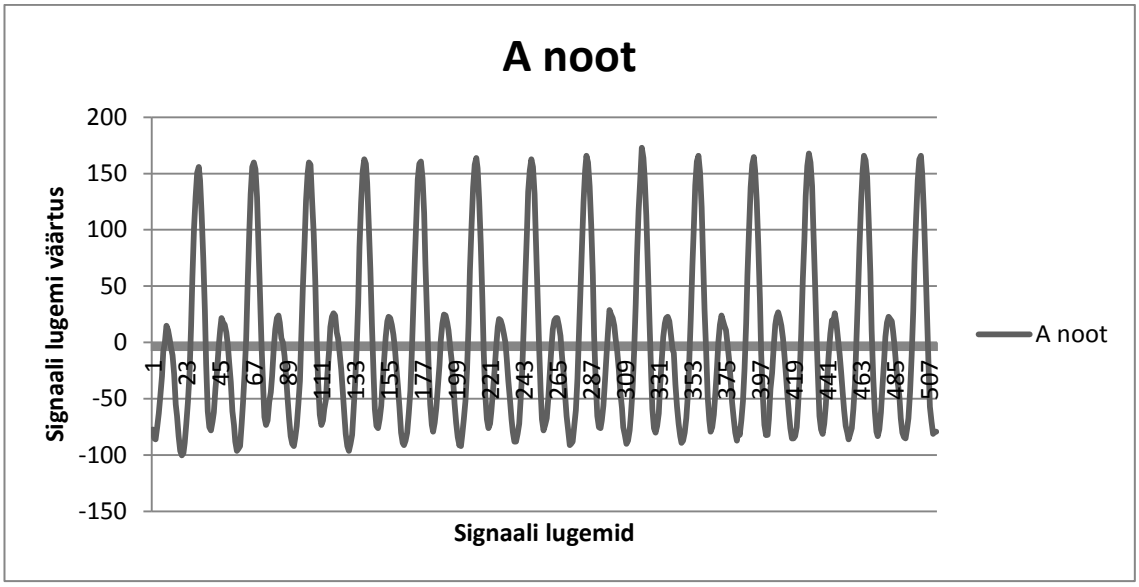
Joonis 12. Kitarri peal mängitud F# noodi (92.50Hz) signaali lugemid



Joonis 13. Kitarri peal mängitud G noodi (98Hz) signaali lugemid



Joonis 14. Kitarri peal mängitud G# noodi (103.83Hz) signaali lugemid



Joonis 15. Kitarri peal mängitud A noodi (110Hz) signaal lugemid

Kui lugemid signaalist on võetud, siis leiame korrelatsiooni. Selleks kasutame Li Tan õpiku valemit:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}} = \sum_{n=0}^{N-1} x(k) W_N^{kn}, kus\ k = 0,1, \dots, N-1 \quad (6)$$

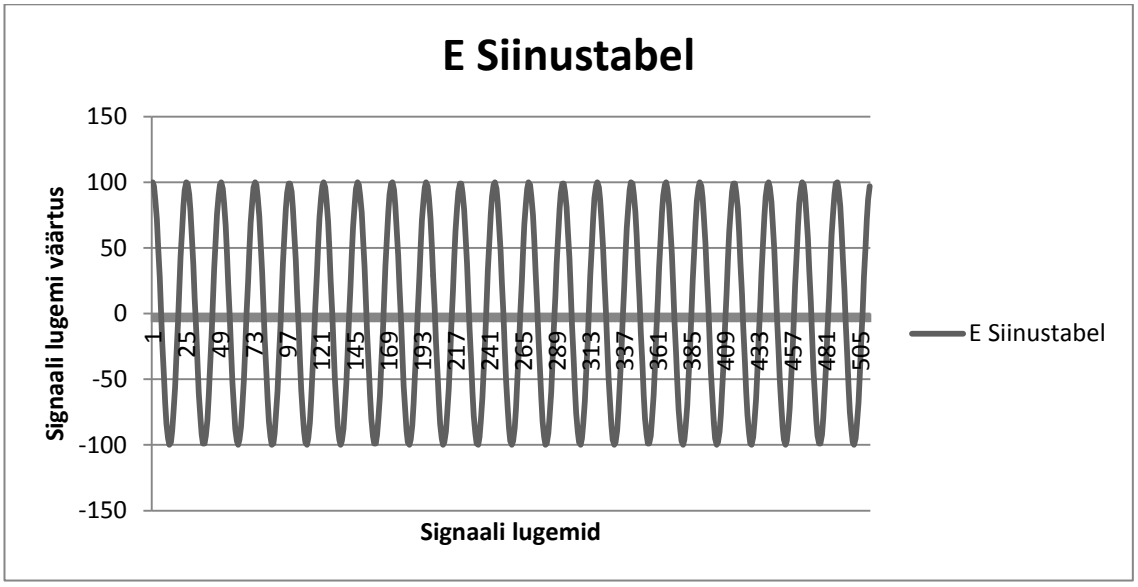
VHDL-is implemeteeritavaks tehteks oleks korruta ja liida tehe ehk MAC tehe:

$$MAC(V1, V2) = \sum_{n=0}^{length(V1)-1} V1(n) \times V2(n) \quad (7)$$

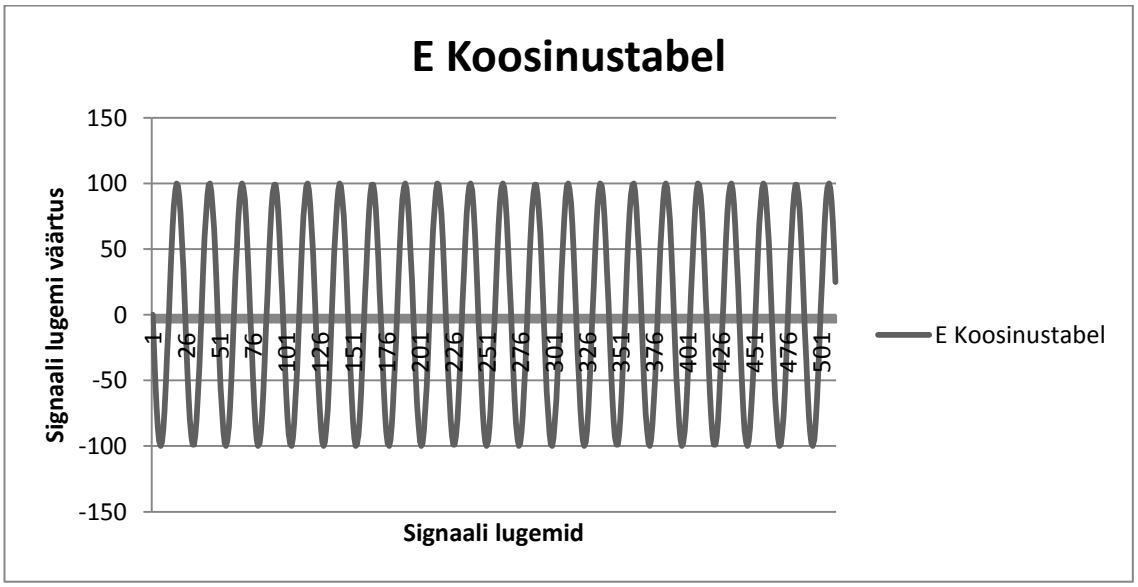
Vastavalt valemile 6 tuleb korrutada sisendsignaali *x* ja kordajat W_N. Selle peame enne välja arvutama, täisarvuks teisendama ja siis FPGA sisse tõstma. W_N kordaja arvutamine on mahukas, seega hoiame nii palju arvutusaega kokku. Idee on selles, et W_N iseenesest ei muutu, ainult sisendsignaal *x* muutub. Seega *V1* on sisendsignaal *x* ja *V2* on sinussoid W_N. MAC näitab kas sisendsignaalis sellist sinussoidi esineb või ei. Summa arvutamiseks kasutan loendurit ja protsessi vajalike arvutustega, kus loenduri väärtus määrab ära lugemi punkti asukoha *n*.

Esmalt defineerime, mitme punktiga korrelatsiooni leiame. Antud juhul on punktide arvuks 512. Seejärel defineerime E ja A noodile vastava koosinusfunktsiooni ja siinusfunktsiooni tabeli (vt Joonis 16, Joonis 17, Joonis 18, Joonis 19). Vektori

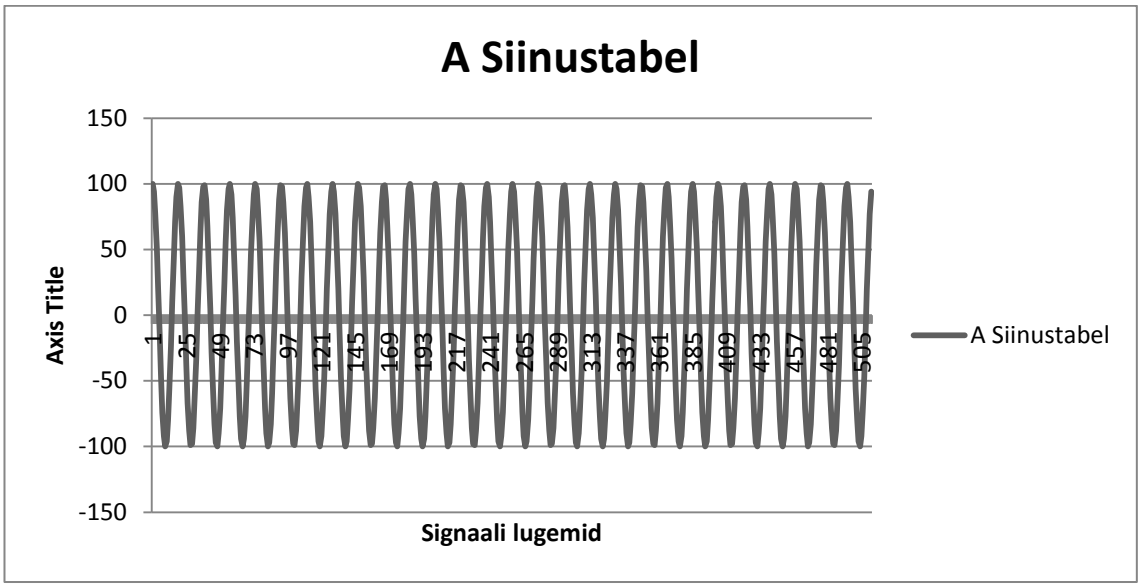
väärtusteks valime täisarvud. Vektorite pikkus peab olema sama pikk kui signaali lugemite vektor. Vektorid tõstame FPGA sisse ja valime tüübiks int. Kui signaalist on lugemid võetud, siis teeme MAC tehte nii siinustabeliga kui ka koosinustabeliga. MAC tehte tüüp peab olema `signed long int`. Tulemuse väärtustega leiame moodi, mis näitab, kas sellist sagedust on sisendsignaalis või mitte. E noodile vastab sagedus 82,407Hz ja A noodile vastab sagedus 110Hz. Kui tuvastatud on E noot, siis lülitame esimese LEDi tööle. Kui tuvastatud on A noot, siis lülitame teise LEDi tööle. Kui esine ühtegi kindlat nooti, siis seda loeme selle müraks ja vilgutame kolmandat LEDi.



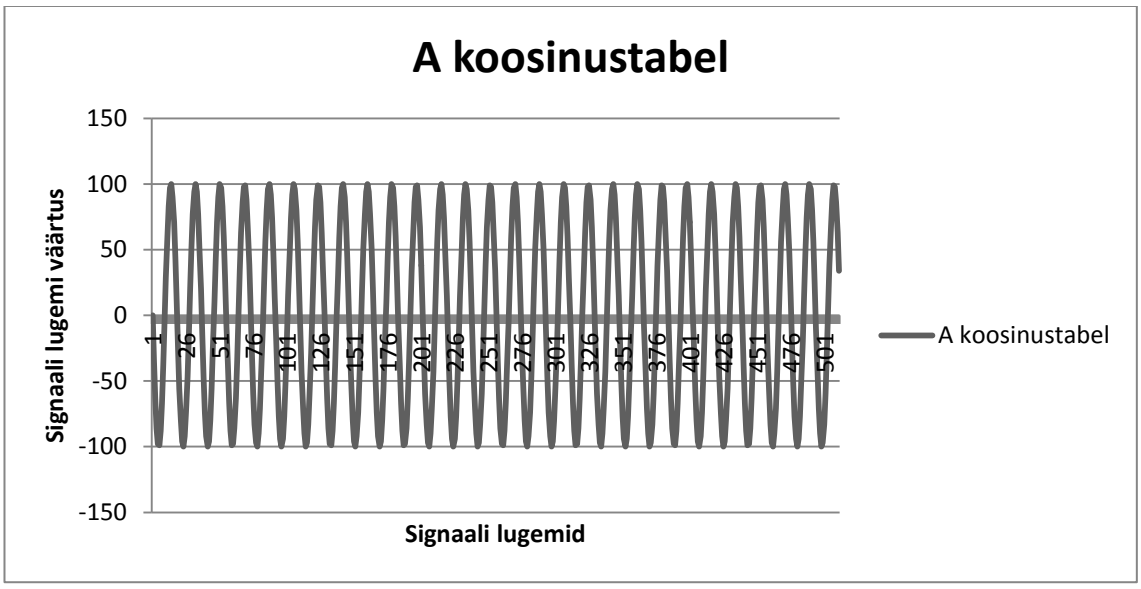
Joonis 16. E noodi siinustabel



Joonis 17. E noodi koosinustabel



Joonis 18. A noodi siinustabel



Joonis 19. A noodi koosinustabel

5. TÖÖPROTSESSI HETKESEIS

PmodMic moodul ja helisignaali lugemine töötab laitmatult.

Lugemite talletamiseks ja nende seejärel lugemiseks on implementeeritud mäluelemendid (BRAM, ehk plokk RAM).

Digitaalse signaalitöötluse osas töötab keskmestamine, aga probleeme on autokorrelatsiooniga, millega töödeldud lugemeid korreleeritakse. Sageduse tuvastamise puhul on tulemus liiga varieeruv. Tulemuse väärtus on vahemikus 500Hz-200Hz helisignaali puhul, mille sageduseks on 440Hz. Seetõttu ei ole tulemus hetkel adekvaatne.

Fourier' teisendus on järgus, kus otsitavate helikõrguste (E noot sagedusega 82.407Hz ja A noot sagedusega 110Hz) siinuste nii reaalsed kui ka imaginaarsed lugemi väärtused on mällu paigutatud. Signaalist võetakse 256 lugemit taktsagedusega 2kHz ja sooritatakse nihe keskmise väärtuse võrra. Seejärel töödeldakse signaali lugemid korrelatsiooni meetodiga, kus paralleelselt leitakse nelja korrelatsioon tulemused ja seejärel leitakse reaalse ja imaginaarse tulemuse mood. Moodi võrdlemisel leiame, kumba sagedust leidub helisignaalis.

Süsteemi programm ja FPGA kivi peale kirjutamiseks .bit fail leiduvad lisana CD-plaadi pealt.

6. ARENDUSPLAANE EDASPIDISEKS

Plaanis on jätkata projektiga ka pärast bakalaureusetöö esitamist, kuna projektil on potentsiaali ja arendamisvõimalusi mitmes erinevas suunas. Esiteks on vaja eksperimentaalne uurimustöö ja teaduskatsetused lõpule viia. Plaanis on uurimustöö arendada edasi magistritööks. Teiseks on sellel projektil ka kommertsialiseerimisväljund. Edaspidi on otstarbekas jätkata uurimustööd rahvusvahelisel tasandil.

Edasise uurimustöö käigus tuleb leida vastused kolmele põhiküsimusele: 1) kas FPGA baasil loodud helituvastussüsteemi loomine õnnestub; 2) kas uurimustöö kaks erinevat osa – helituvastus ja selle visuaalne kuvamine – hakkavad omavahel tööle; 3) ja juhul, kui eksperimentaalne katsetustöö õnnestub, siis kas selle süsteemi baasil on võimalik luua ka toode.

Eeldatavasti on peale uurimustöös osalejate ka teistel muusikutel antud toote vastu huvi. Käesoleva projekti idee avalik esmaesitlus toimus Tartus rahvusvahelise konkursi *Digilent Design Contest 2013* Eesti presentatsioonil (03.05.2013, vt http://ddc2013.eu/). Projekt pälvis teise koha FPGA tootjafirma Digilent Inc. tooteid kasutava õppeaine Programmeeritava loogika projekt (LOTI.05.017) raames (rahvusvahelise konkursi koduleht).

KOKKUVÕTE

Käesoleva bakalaureusetöö eesmärgiks oli luua FPGA peal töötav süsteem, mis loeb sisse analoogheli ning digitaliseerib selle ja töötleb signaalitöötluse meetoditega. Esmalt loeb süsteem analoogheli FPGA-ga ühenduses oleva mikrofoniga sisse, muundab analoogheli analoog-digitaalmuunduriga digitaalseks signaaliks, seejärel võtab signaalist lugemeid ja lugemite töötlemisel tuvastab heli sagedused, mille põhjal süsteem leiab mängitud noodi helikõrguse. Bakalaureusetöö on osa suuremast projektist, milles lisaks antud töö käigus loodud helide tuvastamise süsteemile kuvatakse tuvastatud noodid ka digitaalsel noodilehel.

Kõigepealt on analüüsitud erinevaid teaduspublikatsioone signaalitöötluse meetodite kohta, millega on võimalik helisignaalist sagedusi tuvastada. Meetod pidi olema küllaltki lihtsa ehitusega, et seda oleks võimalik VHDL riistvarakirjelduskeeles rakendada, aga samas olema suuteline leidma helisignaalis esinevaid sagedusi. Meetoditeks valiti autokorrelatsioon ja Fourier'i teisendus, mida katsetati eraldi.

Analoogsignaali digitaliseerimiseks tutvuti FPGA-le lisatava mooduliga PmodMic, mis koosnes mikrofonist ja analoog-digitaalmuundurist. Selleks pidi teadma, kuidas ja mis sagedusel muundur töötab ja kuidas kasutada Moore'i olekumasinat, et tulemuseks saada digitaliseeritud signaal.

Autokorrelatsiooni meetodi rakendamise puhul tuli esmalt võtta helisignaalist lugemid ja seejärel talletada need mällu. Signaali silumiseks töödeldi lugemid keskmestamise meetodiga. Keskmestamiseks analüüsiti lugemeid ja nende põhjal määrati ära

keskmestamise lävend. Autokorrelatsiooni meetod seisneb lugemite iseendaga korrutamise väärtuste summeerimises. Sageduse leidmiseks tuli leida autokorrelatsiooni tulemuse lugemite põhjal viivis, mis vastab helisignaali sagedusele.

Fourier'i teisenduse meetodi rakendamise puhul tuli esmalt leida tuvastatava helisageduse siinuseline väärtus ning talletada selle lugemid mällu. Seejärel võeti helisignaalist lugemid ja leiti summa tuvastatava helisageduse ja helisignaali lugemite korrutiste vahel. Summa väärtuse järgi on võimalik leida, kas helisignaalis esines tuvastatav sagedus. Süsteemi edasiarenduseks tuleks ühendada käesoleva osaga noodi kuvamise osa.

Töö autor tänab oma juhendajat Margus Rosinat, kes andis probleemiseade ja jagas oma teadmisi ja kogemusi. Samuti tänab Fred Valku, kes andis lahendusideid helikõrguse tuvastamise osas.

Processing and digitizing analog sound using an FPGA

Koit Summatavet

SUMMARY

The aim of this bachelor thesis was to create a working system on an FPGA that would capture analog audio signal and digitize it and process it using digital signal processing methods. First the system captures analog audio signal with a microphone connected to an FPGA, afterwards converts the signal using analog to digital converter (ADC). Thereafter the signal is sampled and by further processing these samples finds the frequency of the signal. With the frequency the system can describe the pitch of the note played. The Bachelor thesis was a part of a larger project in which in addition to the system identifying played notes, displayed the notes digitally through VGA on a music sheet.

Firstly, research publications on digital signal processing methods were thoroughly analyzed to find possibilities on how to identify pitch from audio signal. The method had to be with a simple architecture so that it would be possible to describe with the VHDL hardware description language, but at the same time capable of finding frequencies in the signal. The different methods used in this bachelor thesis were autocorrelation and Fourier transform.

For the analog signal conversion to digital form, the PmodMic module for the FPGA had to be familiarized with. In order to know how and with what clock frequency the ADC needed and how to use the Moore's state machine, to generate the digitized signal.

Signal samples had to be taken and stored in the memory for the aurocorrelation method. Center clipping method was used to spectrally flatten the signal simples. Center clipping threshold was found by measuring the signal samples. Autocorrelation is cross-correlation of a signal with itself, meaning sum of multiplication with its values. To find the frequency the lag in the autocorrelated samples had to be found, that represented the pitch.

To implement the Fourier transform firstly the sinusoidal value of the sound pitch that was to be identified had to be found and samples of the sinusoid needed to be stored in the memory. Secondly, samples were taken of the sound signal and the sum of multiplication between signal samples and sound pitch sinusoid samples. From value of the sum the frequency in the sound signal could be identified. For further improvement this project should be connected with the note displaying project.

The author of this thesis would like to thank his thesis supervisor who described the subject and shared his knowledge and experiences concerning the matter. Fred Valk is also thanked for his solutions concerning pitch detection.

KASUTATUD KIRJANDUS

Christensen, R. L., Strong, W. J., Palmer, E. P. (1974), "A Comparison of Three Methods of Extracting Resonance Information from Predictor-Coefficent Coded Speech", Journal of the Acoustical Society of America, Vol. 56, Issue **S1**, S15–S16.

de Cheveigné, A., Kawahara, H. (1999), "Multiple period estimation and pitch perception model", Speech Communication, Vol. **27(3)**, 175–185.

Davy, M., Godsill, S. J. (2003), "Bayesian harmonic models for musical signal analysis", Bayesian Statistics, **7**, 105–124.

Dubnowski, J. J., Schafer, R. W., Rabiner, L. R. (1976), "Real-Time Digital Hardware Pitch Detector", IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. **ASSP-24**, No. **1**, February, 2–8.

Goto, M. (2004), "A Real-time Music-scene-description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines", Real-world Audio Signals, Speech Communication (ISCA Journal), Vol. **43**, No. **4**, September, 311–329.

Haskell, R. E., Hanna, D. M. (2010), "Digital design: using digilent FPGA boards: VHDL/Active-HDL edition", Rochester: LBE Books. Available http://tartu.ester.ee/record=b2579899~S1*est (07.04.2013)

Klapuri, A. P. (2004), "Automatic music transcription as we know it today", Journal of New Music Research, **33(3)**, 269–282.

Keaton, R. H. (1936), "U.S. Patent No. 2,047,690", Washington, DC: U.S. Patent and Trademark Office.

Lu, D. (2006), "Automatic music transcription using genetic algorithms and electronic synthesis", Computer Science Undergraduate Research, University of Rochester, USA: New York.

Manalo, M. S., Ashraf, A. (2012), "Implementing Filters on FPGAs. Department of Electrical and Computer Engineering", USA: San Diego State University, Real-Time

DSP and FPGA Development Lab.

Nicolae, M., Rugina, I., Vasile, A. (2000), "Optimizing Implementation of Autocorrelation Function", The Annual Symposium of the Institute of Solid Mechanics SISOM'2000, Bucharest, 118–122. Available http://www.imsar.ro/SISOM_Papers_2000/D_13.pdf (07.04.2013)

Pedroni, V. A. (2004), "Circuit Design with VHDL", London: MIT Press. Available http://tartu.ester.ee/record=b1793044~S1*est (07.04.2013)

Polt, R. F. H. (1995), "A brief history of typewriters", The classic typewriter page. Available http://site.xavier.edu/polt/typewriters/ (07.04.2013)

Rabiner, L. R., Schafer, R. W. (2011), "Digital Speech Processing", The Froehlich/Kent Encyclopedia of Telecommunications, Vol. **6**, 237–258.

Reese, G. (1934), "The First Printed Collection of Part-Music: (The Odhecaton)", The Musical Quarterly, **20(1)**, 39–76.

Rosin, M. (2009), "Kursuse „mikroprotsessorid" fpga ja vhdl õppemoodul", magistritöö, Tartu: Tartu Ülikool Loodus- ja Tehnoloogiateaduskond, Füüsika Instituut. Kättesaadav http://www.fi.tartu.ee/loputood2009/infotehnoloogia/rosin_margus.pdf (07.04.2013)

Tan, L., Karnjanadecha, M. (2003), "Pitch detection algorithm: autocorrelation method and AMDF", Proceedings of the 3rd International Symposium on Communications and Information Technology, Vol. **2**, September, 551–556. Available http://fivedots.coe.psu.ac.th/~montri/Research/Publications/iscit2003_pda.pdf (07.04.2013)

Vallaste, H. (2013), "E-teatmik". Kättesaadav http:/www.vallaste.ee/ (10.04.2013)

Wong, K. H. (2011), "Ch3: Pitch estimation for music signal processing", MSc Programme in Computer Science: CMSC5707 Advanced Topics in Artificial Intelligence, Hong Kong: Department of Computer Science and Engineering, the Chinese University of Hong Kong. Available http://appsrv.cse.cuhk.edu.hk/~khwong/www2/cmsc5707/cmsc5707.html (10.04.2013)

DIGITAALSED ANDMEBAASID

Digilent Inc. moodul Basys2 käsiraamat.
http://www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf

Digilent Inc. lisamooduli PmodMic käsiraamat ja kontroller.
http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,517&Prod=PMOD-MIC

Xilinx ISE WebPACK 14.4. http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.htm

JOONISED

Joonis 1. http://www.digilentinc.com/Data/Products/BASYS2/BASYS2-top-400.jpg (17.04.2013)

Joonis 2. http://www.digilentinc.com/Data/Products/PMOD-MIC/PmodMIC-obl-400.jpg (17.04.2013)

Joonis 3. http://www.slideshare.net/lkerin/eee467-add-finalprojectahmetilkerin070203012 (17.04.2013)

Joonis 4. http://www.slideshare.net/lkerin/eee467-add-finalprojectahmetilkerin070203012 (17.04.2013)

Joonis 5. http://www.slideshare.net/lkerin/eee467-add-finalprojectahmetilkerin070203012 (17.04.2013)

Joonis 6. http://www.slideshare.net/lkerin/eee467-add-finalprojectahmetilkerin070203012 (17.04.2013)

Joonis 9. www.cse.cuhk.edu.hk/~khwong/www2/cmsc5707/5707_3_pitch.ppt

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Koit Summatavet

(*autori nimi*)

(sünnikuupäev: 28.12.1990)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
FPGA kasutamine analoogheli digitaliseerimiseks ja töötlemiseks,

(*lõputöö pealkiri*)

mille juhendaja on Margus Rosin,

(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **11.05.2013**