Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2021

Jimi Toiviainen

AUDIOSIGNAALIN KÄSITTELY FPGA-PIIREJÄ JA MATLAB-OHJELMISTOA HYÖDYNTÄEN

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2021 | 39 sivua

Jimi Toiviainen

AUDIOSIGNAALIN KÄSITTELY FPGA-PIIREJÄ JA MATLAB-OHJELMISTOA HYÖDYNTÄEN

Tämän opinnäytetyön tavoitteena oli toteuttaa kitaraefektipedaali hyödyntäen digitaalista signaalinkäsittelyä. Työssä piirrettiin ohjelma kaaviokuvana MATLAB-ympäristössä, josta se vietiin FPGA-piirille käyttäen Simulink- ja Xilinx Vivado -ohjelmistoja.

Opinnäytetyön teoreettisessa osassa tutkittiin digitaalisen äänen historiaa ja työssä vaadittavia ohjelmistoja, vaatimuksia ja arkkitehtuuria.

Toteutusosassa suunniteltiin erilaisia efektejä kaaviokuvaan, asennettiin FPGA-piirille vaadittava ohjelmisto ja siirrettiin suunniteltu kaaviokuva FPGA-piirille.

Testausosiossa testattiin vaatimuksia MATLAB-simulaatiossa ja FPGA-piirillä.

Työn tuloksena MATLAB-simulaatiossa saatiin suurin osa halutuista efekteistä toimimaan. FPGA-piirille saatiin asennettua tarvittavat alustukset ja tässä työssä suunniteltu ohjelma MATLAB-ympäristön kautta. FPGA-piirillä todettiin myös toimivaksi prosessoimattoman signaalin läpikulku. Todettiin myös, että jatkokehityksenä voitaisiin selvittää tarkempia parametreja ja efektityyppejä, jotka toimisivat paremmin FPGA-ympäristössä.

Asiasanat:

DSP, FPGA, MATLAB, Simulink, VHDL.

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2021 | 39 pages

Jimi Toiviainen

AUDIO SIGNAL PROCESSING USING FPGA AND MATLAB

The aim of this thesis was to implement a guitar-effects pedal utilizing digital signal processing. In the thesis, a program was drawn as a block diagram in a MATLAB environment where it was programmed onto an FPGA circuit using Simulink and Xilinx Vivado software.

The theoretical section of the thesis investigated the history of digital sound and the software, requirements and architecture required in the work.

In the implementation section various effects were designed for the schematic, the software required for the FPGA was installed, and the designed schematic was programmed onto the FPGA.

In the test section the requirements were tested in MATLAB simulation and the FPGA circuit.

As a result of the work, most of the desired effects were made to work in the MATLAB simulation. The necessary installations and the programming of the schematic were performed on the FPGA circuit via the MATLAB environment. The FPGA circuit was found to pass through unprocessed signal as intended. However, more precise parameters and better effect types for the FPGA board could be investigated for further development.

Keywords:

DSP, FPGA, MATLAB, Simulink, VHDL.

Sisältö

Kuvat

Taulukot

1 Johdanto

Ääntä on ruvettu tallentamaan jo 1800-luvulla. Thomas Edisonin 1800-luvun lopulla keksimä fonografi oli ensimmäinen ääntä tallentava laite. Fonografi oli tarkoitettu pääasiassa puheen äänittämiseen, mutta jo varhaisessa vaiheessa ruvettiin taltioimaan myös musiikkia. Fonografin pääperiaate on se, että ääni tallennetaan uraksi vahalieriöön, josta se voidaan toistaa uudelleen. 1920-luvulla alettiin käyttää sähköavusteisia menetelmiä äänen tallennuksessa, vaikka ääni tallennettiinkin fyysisiksi uriksi levyille. Ääntä voitiin muuttaa signaaliksi mikrofoneilla ja esimerkiksi vahvistaa vahvistimilla. Äänenlaatu oli tässä vaiheessa parantunut jo niin paljon, että se oli synnyttänyt uuden ammatin: ääniteknikko. Ääniteknikot etsivät parempia äänitystekniikoita, jotta tallenteista saataisiin yhä aidomman kuuloisia. [1]

Toisen maailmansodan aikaan saksalaiset alkoivat tallentaa ääntä magneettinauhoille. Äänenlaatu oli tällä teknologialla jo niin hyvä, etteivät vastapuolen liittoutuneet erottaneet enää ennalta äänitettyjä ohjelmia suorista lähetyksistä. 1950-luvulla magneettinauhoista tuli nopeasti standarditapa äänen nauhoitukseen. Magneettinauhalle voitiin tallentaa entistä pidempiä ottoja ja tallennettua ääntä oli helppo editoida ennennäkemättömillä tavoilla, esimerkiksi leikkaamalla ja liimaamalla. Yksi suurimmista vaikuttavista tekijöistä oli se, että useampia äänilähteitä voitiin toistaa samanaikaisesti, rinnakkain. Tämä mahdollisti sen, että kun aikaisemmin piti kaikki instrumentit äänittää yhtä aikaa, nyt voitiin jokainen instrumentti soittaa erikseen peräkkäin. [1]

Kaikki aikaisemmin mainitut teknologiat ovat analogista ääntä. Analoginen tarkoittaa sitä, että joku fyysinen jatkuva-aikainen signaali muutetaan toiseen, suoraan verrannolliseen muotoon, joka voi saada äärettömän määrän eri arvoja. 1970-luvulla tietokoneiden yleistymisen yhteydessä audiota alettiin tallentamaan digitaaliseen muotoon. Digitaalisessa muodossa on analogiseen erona se, että äänestä otetaan näytteitä, jotka tallennetaan diskreetissä muodossa. Tästä tallennetusta näytejonosta voidaan puolestaan

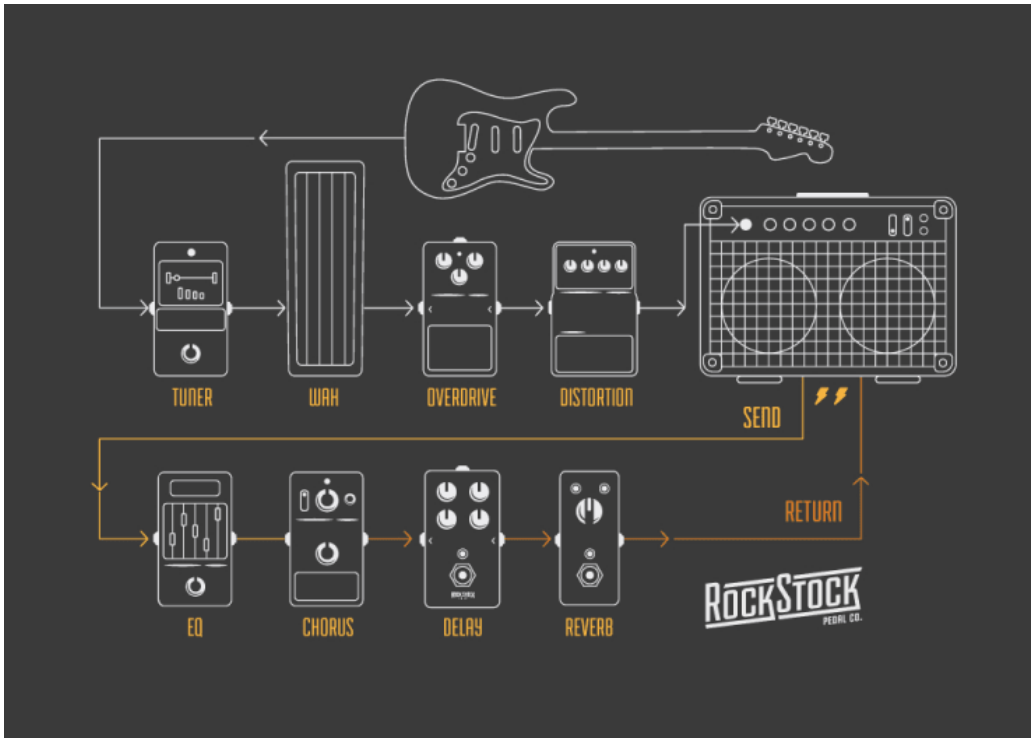
uudelleenrakentaa eli rekonstruoida analogisia signaaleita, joilla voidaan tuottaa tallennettu ääni uudelleen. [2]

Digitaalinen ääni on tuonut paljon lisää hyötyjä. Koska ääni on tallennettu diskreetissä muodossa, sen laatu ei heikkene ajan tai käytön myötä. Diskreetti muoto tarkoittaa sitä, että signaalin aika- ja amplitudikomponentit voivat saada vain tiettyjä arvoja. Signaalin aikakomponentin mahdolliset arvot määriytyvät signaalin pituuden ja näytteenottotaajuuden mukaan, kun taas amplitudikomponentin mahdolliset arvot määriytyvät näytteenottoresoluution mukaan. Myös tallennusmediat ovat lukutapojensa myötä sellaisia, että ne eivät kulu. Analogisia levyjä luetaan käyttäen hyväksi mekaanista kontaktia, joka kuluttaa levyn pintaa. Esimerkiksi digitaalisia CD-levyjä luettaessa käytetään optista teknologiaa, jolloin mekaanista kontaktia ei synny ja levyn pinta ei kulu. [2]

Tässä työssä tavoitteena on tehdä digitaalinen FPGA-piiriä hyödyntävä kitaraefektilaite käyttämällä MATLAB-ohjelmistoa. Kitaraefekti on laite, joka laitetaan käyttötarkoituksesta riippuen joko kitaran ja etuvahvistimen väliin tai etu- ja päätevahvistimen väliin. Kitaraefektin tarkoituksena on muokata ääntä halutulla tavalla. Yleisimpiä efektityyppejä ovat esimerkiksi särö, erityyppiset kaiut, taajuuskorjaimet, häiriönpoisto ja chorus. Esimerkkejä efektipedaaleista voidaan nähdä kuvassa Kuva 1.

Tyypillisesti kitarefektejä ketjutetaan useampia peräkkäin kuvan Kuva 2 mukaisesti.

Kuva 1: Erilaisia kitarapedaaleja [3]



Kuva 2: Kaaviokuva kitaraefekteistä eri vaiheissa [4]

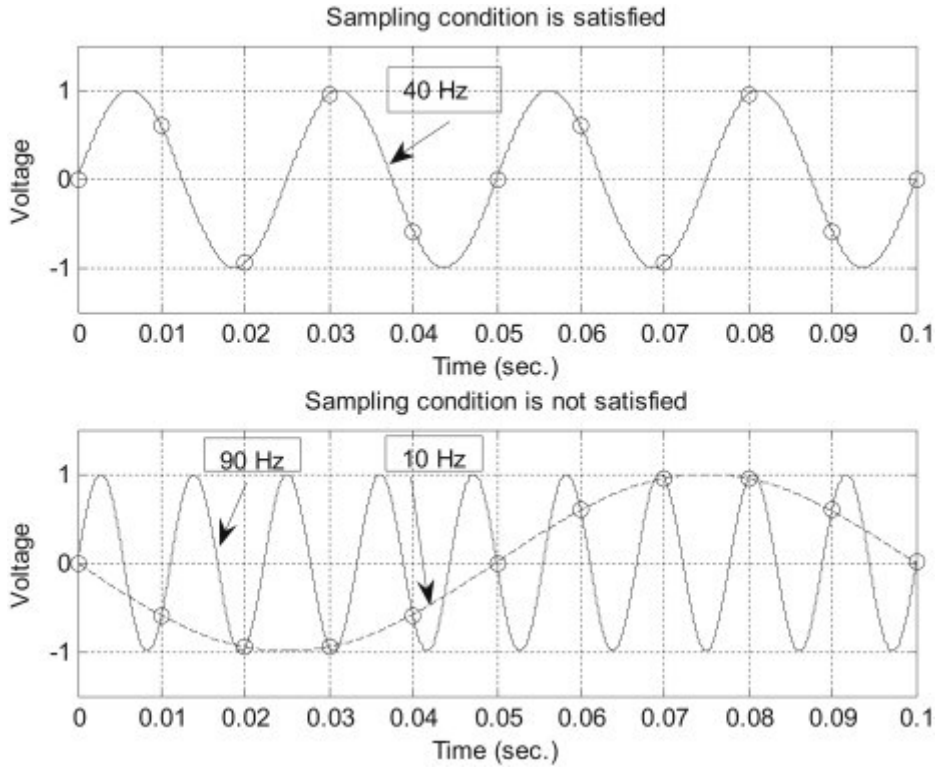
Luvussa 2 esitellään tässä työssä käytetyt menetelmät ja teknologiat. Luvussa 3 kuvataan tässä työssä rakennettavan kitaraefektilaitteen vaatimuksia käyttäen MoSCoW-metodia. Luvussa 4 paneudutaan arkkitehtuurivalintoihin, ja niiden syihin. Luvussa 5 käydään läpi toteutus. Luvussa 6 testataan toteutuksen eri osat. Luvussa 7 keskustellaan tuloksista ja pohditaan mahdollista jatkokehitystä.

2 Menetelmät ja teknologiat

2.1 Digitaalinen ääni

Digitaalinen ääni koostuu aikasarjasta näytteitä, jotka kuvaavat signaalin voimakkuutta, eli amplitudia kyseisellä ajanhetkellä. Näytteiden tarkkuus, eli resoluutio on yleensä 16-, 24- tai 32-bittinen, tarkoittaen, että esimerkiksi 24-bittisessä järjestelmässä signaalin voimakkuudelle on 2^{24} (16777216) eri arvoa, jossa $2^{24/2}$ on nollakohta. Tämän alle menevät arvot kuvaavat negatiivisia arvoja. [5]

Näytteenottotaajuus määrää sen, kuinka tiheästi näytteitä otetaan tai toistetaan. Tyypillisimpiä näytteidenottotaajuuksia ovat esimerkiksi 44,1kHz, 48kHz, 96kHz tai jopa 192kHz sovelluksen laatuvaatimuksista riippuen. Näytteenottotaajuus 44,1kHz ja 24-bittinen resoluutio riittää tuottamaan kaikki taajuudet, joita ihmiskorva kuulee. Äänelle saatetaan tehdä prosessointia, jossa muutokset korkeissa taajuuksissa vaikuttavat alempiin taajuuksiin. Tällöin korkeampia näytteenottotaajuuksia tarvitaan esimerkiksi laskostumisen estämiseksi. Kuvassa Kuva 3 nähdään esimerkki laskostumisesta. [5]



Kuva 3: Esimerkki laskostumisesta [5]

2.2 Signaaliketju

Digitaalisen audion signaaliketju koostuu lukuisista osista, joista seuraavassa lueteltu yleisimmät, tyypillisessä järjestyksessä.

Vahvistin

Signaali vahvistetaan AD-muunnokselle sopivaksi voimakkuudeksi. Tulosignaalin voimakkuuden ollessa liian pieni signaali-kohinasuhde vähenee, mikä aiheuttaa taustakohinaa. Yhdistettäessä tätä esimerkiksi säröefektiin, jossa signaalia vahvistetaan erittäin paljon, tulee kohinasta erittäin huomattavaa. Tulosignaalin ollessa liian voimakas signaali säröytyy. Vaikka säröytyminen saattaa olla haluttu efekti, yleisesti ottaen AD-muunnosta tehtäessä ei haluta tapahtuvan säröytymistä.

Laskostumisen esto (Anti-aliasing)

Signaalista poistetaan laskostumisesta johtuvia häiriöitä, esimerkiksi leikkaamalla kaikki taajuudet, jotka ylittävä niin kutsutun Nyqvistin taajuuden. Nyquist-taajuus on puolet näytteenottotaajudesta. Teorian mukaan signaali tulee näytteistää vähintään $F_{max} <= F_s/2$, jotta laskostumista ei tapahtuisi rekonstruktiovaiheessa. [6]

AD-muunnos (Analog to Digital - muunnos)

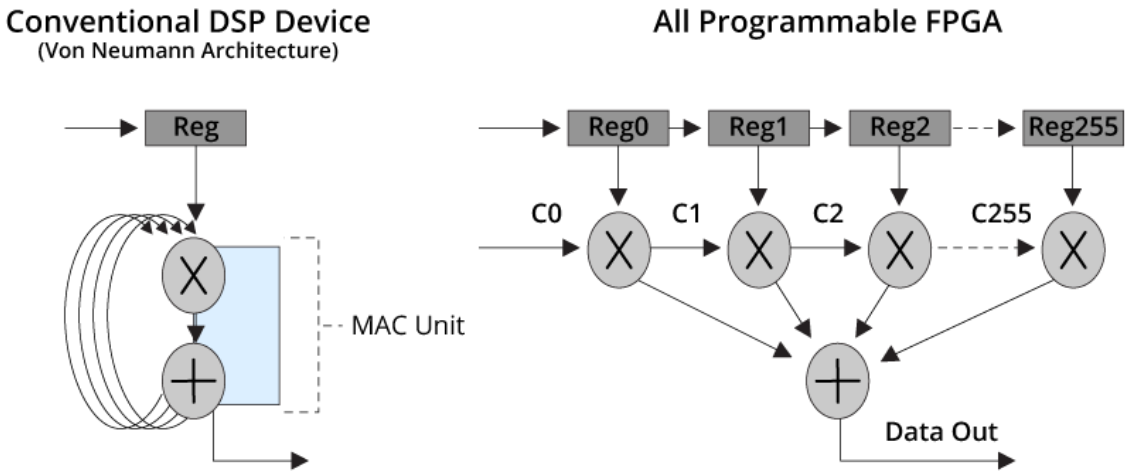
Signaali muunnetaan digitaaliseksi ottamalla näytteitä signaalista haluttua näytteenottotaajuutta vastaavalla näytteenottovälillä. Yleisimpiä AD-muunnintyyppejä ovat Successive Approximation (SAR), Delta-sigma (ΔΣ), Dual Slope, Pipelined ja Flash. SAR ja Delta-sigma ovat käytössä enimmäkseen datankeräyksessä. SAR-tyypin muuntimen hyvänä puolena on hyvä nopeus/resoluutiosuhde. Delta-sigmassa hyvänä puolena on korkea dynaaminen alue ja sisäänrakennettu laskostumisen esto. Näistä syistä audion yhteydessä käytetään yleensä Delta-sigma-muuntimia. Dual Slope-tyyppiset muuntimet ovat tarkkoja ja halpoja tuottaa, mutta niiden näyttenottotaajuus on hidas. Näitä käytetään esimerkiksi jännitemittareissa. Pipelined ja Flash ovat nopeita, mutta epätarkkoja AD-tyyppejä, joita käytetään esimerkiksi oskilloskoopeissa, kun mitataan erittäin nopeita signaaleita.

DSP (Digitaalinen Signaalinkäsittely, Digital Signal Processing)

Digitaalista signaalia voidaan prosessoida käyttäen digitaalisia työkaluja. Erilaisia digitaalisia työkaluja ovat esimerkiksi yleisprosessorit (CPU), digitaalisen signaalin prosessointiin erikoistuneet prosessorit (DSP), tai tiettyä tarkoitusta varten luodut sähköpiirit. CPU- ja DSP-alustat ovat molemmat prosessoreita, mutta DSP:n suorittamat ohjeet ovat tehty käsittelemään nimenomaan digitaalista signaalia, kun taas CPU:n ohjeet ovat tehty yleisesti sopiviksi kaikenlaiseen prosessointiin ja datankäsittelyyn. Tästä johtuen DSP:t

ovat signaalinkäsittelyyn paljon tehokkaampia, ja antavat esimerkiksi musiikin soittamisessa vähemmän viivettä. Sähköpiirinä luodut efektit eivät tuota lähes lainkaan viivettä, mutta ovat monimutkaisia suunnitella ja toteuttaa. FPGA:lla luoduissa sähköpiireissä yhdistyvät nopeus ja uudelleenohjelmoitavuus.

Kuvassa Kuva 4 nähdään FIR-suotimen ero DSP- ja FPGA-laitteella toteutettuna. DSP:llä vaaditaan n-tappisessa FIR-suotimessa n-kappaletta kellojaksoja, kun taas FPGA:lla sama saadaan aikaan parhaimmillaan yhdessä kellojaksossa.



Kuva 4: DSP ja FPGA FIR-suodin

DA-muunnos (Digital to Analog Converter)

Signaali muutetaan takaisin analogiseksi, jolloin se voidaan muuttaa esimerkiksi jännitteestä sähkömagneetin avulla ilmanpaineeksi ja täten ääneksi. Yleisimpiä DA-muunnintyyppejä ovat Delta-Sigma -muunnin, Binary Weighted Resistor, R-2R tikapuuverkko ja pulssinleveysmodulaatio.

Rekonstruktiosuodin

Näytteiden välinen analoginen signaali luodaan rekonstruktiosuotimella. Jos alkuperäisessä analogisessa signaalissa ei olisi tehty laskostumisen estämistä, saattaisi olla otettu näytteitä signaalista, jossa taajuus nousisi yli Nyqvistin taajuuden, jolloin rekonstruktiosuotimella tuotettu tulos olisi väärä.

2.3 MATLAB-ohjelmisto

MATLAB on MathWorksin tekemä tieteellinen laskenta- ja simulaatioympäristö, joka on laajalti käytössä digitaalisessa signaalinkäsittelyssä. MATLABiin on saatavilla runsaasti lisäpaketteja, jotka laajentavat sen käyttötarkoitusta. [7]

2.4 Simulink-ohjelmisto

Simulink on MATLABin laajennus, joka mahdollistaa graafisen ohjelmointialustan erilaisien järjestelmien mallinnukseen, simulointiin ja analysointiin. Simulinkin pääkäyttöliittymänä toimii graafinen lohkokaaviotyökalu ja erilaisia muokattavia lohkokaaviokirjastoja. Simulink mahdollistaa saumattoman integraation MATLAB-ympäristössä. Sitä voidaan käyttää joko omasta käyttöliittymästä tai funktiokutsuina MATLABin skriptien sisältä. Simulink on laajasti käytetty automatisoinnin ja digitaalisen signaalien prosessoinnin simuloinnissa ja suunnittelussa. [8]

HDL Coder on Simulinkin laajennusosa, jonka avulla Simulinkissä luotavista malleista voidaan luoda yhdessä Xilinx Vivadon kanssa syntesoitavaa VHDL- ja Verilog-koodia. Vastaavasti MATLAB Coder ja Simulink Coder pystyvät tuottamaan malleista C-kieltä, joka on suunnattu esimerkiksi ZYNQ-piirissä olevalle ARM-mikrokontrollerille. [8]

2.5 Xilinx Vivado kehitysympäristö

Vivado Design Suite on Xilinxin kehittämä ohjelmisto, joka on tarkoitettu HDL-mallien syntesointiin ja analysointiin. Jotta MATLAB ja Xilinx toimisivat yhdessä, tarvitsee näistä olla yhteensopivat versiot. Versionumerot määräytyvät sen mukaan, kumpaa työkalua käytetään pääasiallisena työkaluna. Tässä työssä käytetään MATLAB versiota R2020b pääasiallisena työkaluna jolloin HDL Coder vaatii version 2019.2 Xilinx Vivadosta. [9]

2.6 VHDL-kuvauskieli

VHDL (VHSIC-HDL, Very High Speed Integrated Circuit Hardware Description Language) on laitteistokuvauskieli, jota käytetään kuvaamaan digitaalisia- ja monimuotosignaalijärjestelmiä, joita ovat esimerkiksi FPGA ja sulautetut järjestelmät. VHDL:ää voidaan myös käyttää rinnakkaislaskentakielenä. [10]

2.7 FPGA-piirit

FPGA (field-programmable gate array) on järjestelmäpiiri, jota voidaan helposti ohjelmoida uudelleen. FPGA konfigurointi tehdään yleensä käyttäen jotain laitteistokuvauskieltä. FPGA:t sisältävät kokoelmia ohjelmoitavia logiikkalohkoja, joita voidaan yhdistellä toisiinsa eri konfiguraatioissa saaden aikaan erilaisia toiminnallisuuksia. Suurimmassa osassa FPGA:sta on myös muistielementtejä. [11]

FPGA:t ovat suuri hyöty sulautettujen järjestelmien kehityksessä, koska ne mahdollistavat ohjelmiston ja laitteiston yhtäaikaisen suunnittelun ja kehityksen ja järjestelmien testaamisen kehityksen varhaisessa vaiheessa. Monimutkaisimmat FPGA-piirit sisältävät ohjelmoitavan logiikan lisäksi myös prosessoriytimiä. Esimerkki prosessoriytimiä sisältävästä FPGA-piiristä on Xilinxin Zynq-7000 perhe. [11]

3 Vaatimukset

Vaatimukset on kuvattu taulukossa Taulukko 1 käyttäen MoSCoW-metodia [12] siten, että vaatimukset on priorisoitu neljään eri kategoriaan:

- Must have -vaatimukset ovat pakollisia vaatimuksia – tässä opinnäytetyössä tulee minimissään toteuttaa tämän kategorian vaatimukset
- Should have -vaatimukset pitäisi toteuttaa työssä, mutta eivät ole pakollisia.
- Could have -vaatimukset voisivat kuulua työn sisältöön aikataulun salliessa.
- Won’t have -vaatimukset on rajattu tämän työn ulkopuolelle.

Taulukko 1. Tämän opinnäytetyön vaatimukset.

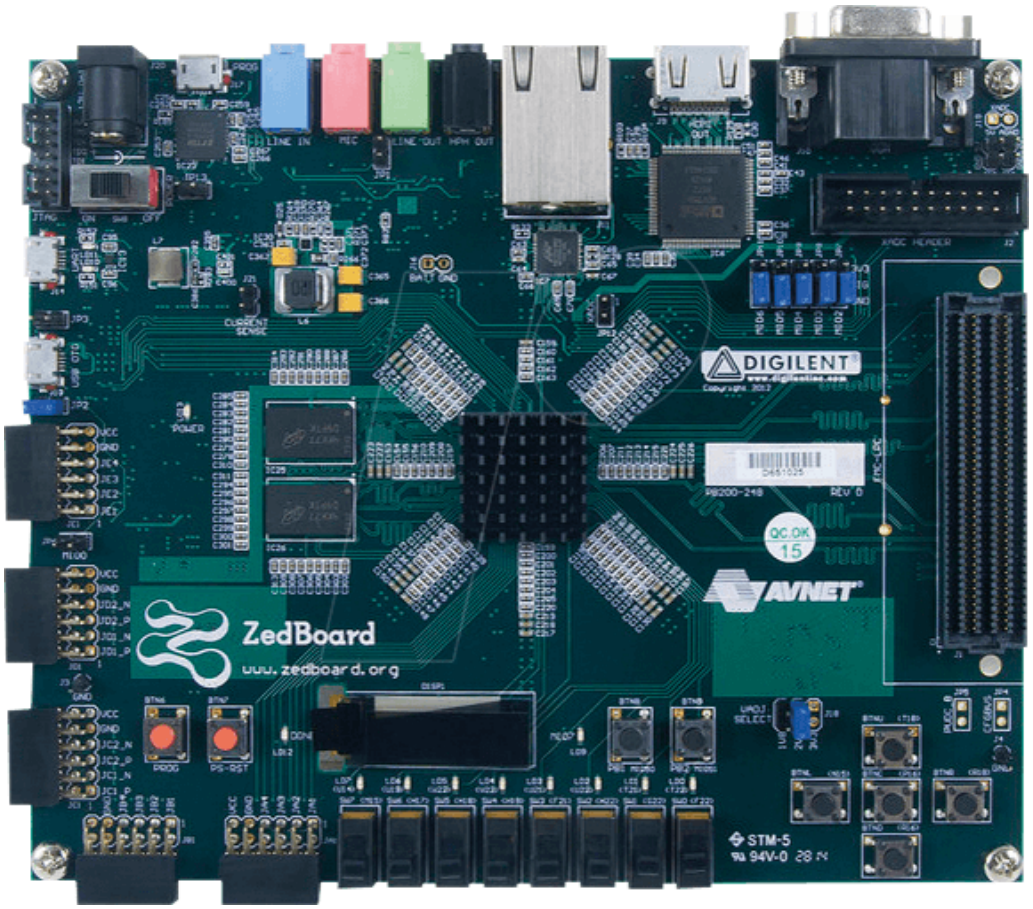
Must have	- Audion sisäänmeno ja saman audion ulostulo samana tai muunneltuna - 3-4 eri ekvalisaattoria - Master volume - Jonkin tyyppinen särö
Should have	- Eri särötyyppejä - Säröefektin parametrien muuttaminen, esim vahvistuksen kertoimen muuttaminen
Could have	- Käyttäjärajapinta efektien vaihtamiseen/aktivoimiseen/deaktivoimiseen - Efektien ketjuttaminen - Eri tyyppisiä kaikuja - Phaser, chrous ja flanger tyyppisiä viivelinjaan perustuvia efektejä
Won't have	- Kitaralle tarkoitettu etuvahvistustus - Päätevahvistus

4 Arkkitehtuuri

4.1 Zedboard-kehityskortti

Laitteistoksi haluttiin valita jokin Zynq-7000 -perheen järjestelmäpiiri. Zynq-7000 koostuu Artix-7 tai Kintex-7 FPGA-piiristä ja kaksiytimisestä ARM Cortex-A9 prosessorista. [13]

Tässä työssä laitteistoksi valittiin kuvassa Kuva 5 nähtävä Zedboard. [14] Vaihtoehtoinen PYNQ [15] antaisi enemmän korkean tason mahdollisuuksia, mutta se on vasta varhaisessa vaiheessa kehitystä, jonka takia oppimisresursseja on sille vielä liian vähän. ZedBoardille löytyy paljon esimerkkejä ja dokumentaatiota.



Kuva 5: ZedBoard

4.2 Signaalipolku

Signaalipolussa kitara kytketään ensimmäisenä kitarasignaalille tarkoitettuun DI-boksiin [16], jonka tehtävänä on muuttaa korkeaimpedanssinen signaali matalaimpedanssiseksi, ja täten säilyttää korkeat taajuudet äänessä. DI-boksista tuleva matalaimpedanssinen signaali syötetään ZedBoardista valittuun sisäänmenoon. ZedBoardissa oleva ohjelma muuttaa signaalin analogisesta digitaaliseksi, jonka jälkeen signaali voidaan viedä erilaisten digitaalisten prosessoreiden läpi. Tämän jälkeen signaali muunnetaan uudelleen analogiseksi ja syötetään ulos ZedBoardista valittuun ulosmenoon. Tähän ulosmenoon voidaan kytkeä esimerkiksi kuulokkeet tai kaiuttimet, tai signaalia voidaan ajaa päätevahvistimelle, jossa se vahvistetaan soitettavaksi kitarakaiuttimen läpi.

5 Toteutus

Toteutusvaihe alkoi tarvittavien ohjelmistojen asennuksella. MATLAB- ja Simulink-ohjelmistojen lisäksi oli tarve asentaa lisäosia, jotta Xilinx-kehitysympäristö ja Zedboard ovat tuettuna. Tämä asennusprosessi on kuvattu seuraavassa alakappaleessa.

5.1 Ohjelmistojen ja ZedBoardin käyttöjärjestelmän asennus

Asennetaan MATLAB R2020b, Simulink ja Simulinkin lisäpaketit HDL Coder, DSP System Toolbox, Embbedded Coder ja Simulink Coder sekä Vivado 2019.2.

Lisätään Vivado MATLABin konsolista polkuun.

```
hdlsetuptoolpath('ToolName', 'Xilinx Vivado', 'ToolPath',
                    'W:\Muut\Xilinx\Vivado\2019.2\bin\viva
                    do.bat');
```

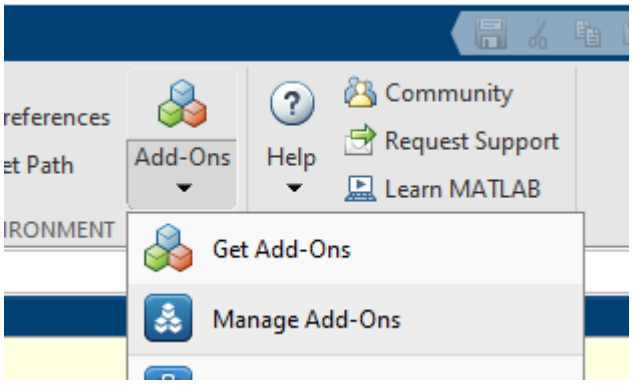
Lisätään myös ipcore ja ZedBoardin konfiguraatioita polkuun.

```
addpath(fullfile(matlabroot,'toolbox','hdlcoder','hdlcoderd
                    emos','customboards','ipcore'));

addpath(fullfile(matlabroot,'toolbox','hdlcoder','hdlcoderd
                    emos','customboards','ZedBoard'));
```

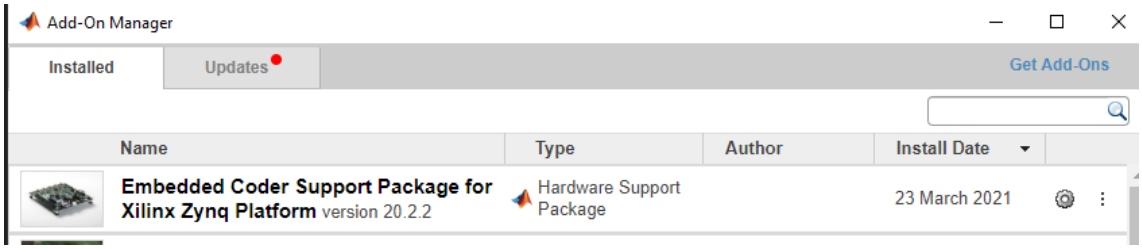
Kirjoitetaan ZedBoardin tarvitsema käyttöjärjestelmä SD-kortille.

Mennään MATLABissa valikkoon Add-Ons -> Manage Add-Ons kuvan Kuva 6 mukaan.



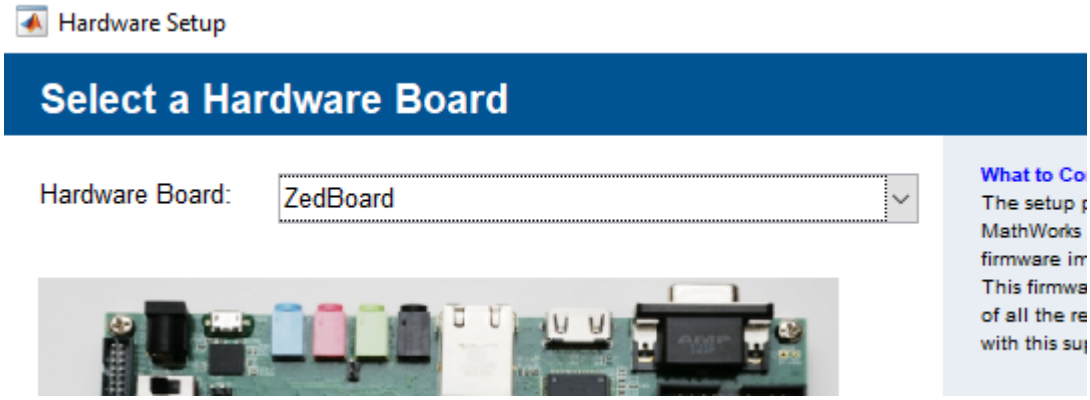
Kuva 6. Simulinkin lisäosavalikko

Painetaan hammasrattaan kuvaa Add-on Managerista Embedded Coder Support Package for Xilinx Zynq Platform vierestä. Kuvassa Kuva 7 nähdään Add-On Managerin näkymä.



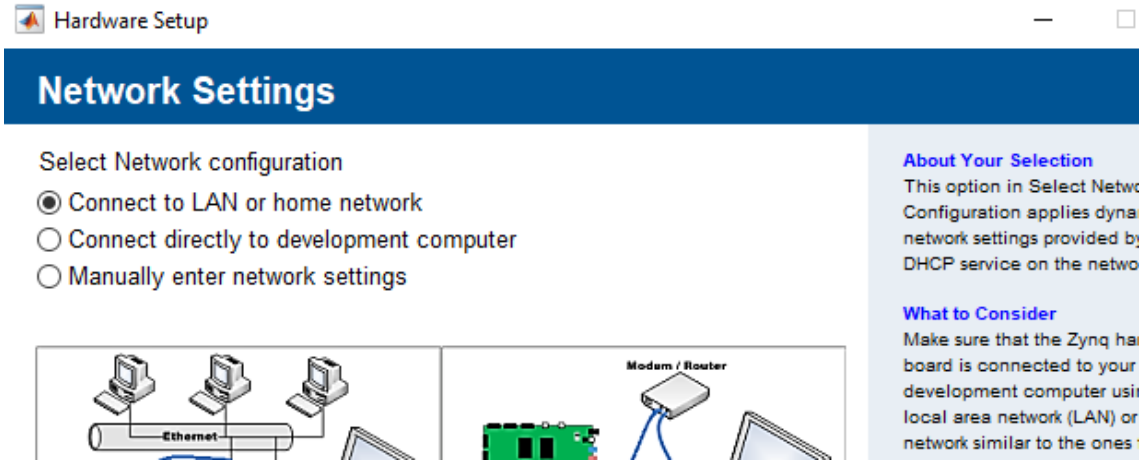
Kuva 7. Lisäosavalikkonäkymä

Valitaan hardware boardiksi ZedBoard kuvan Kuva 8 osoittamalla tavalla.



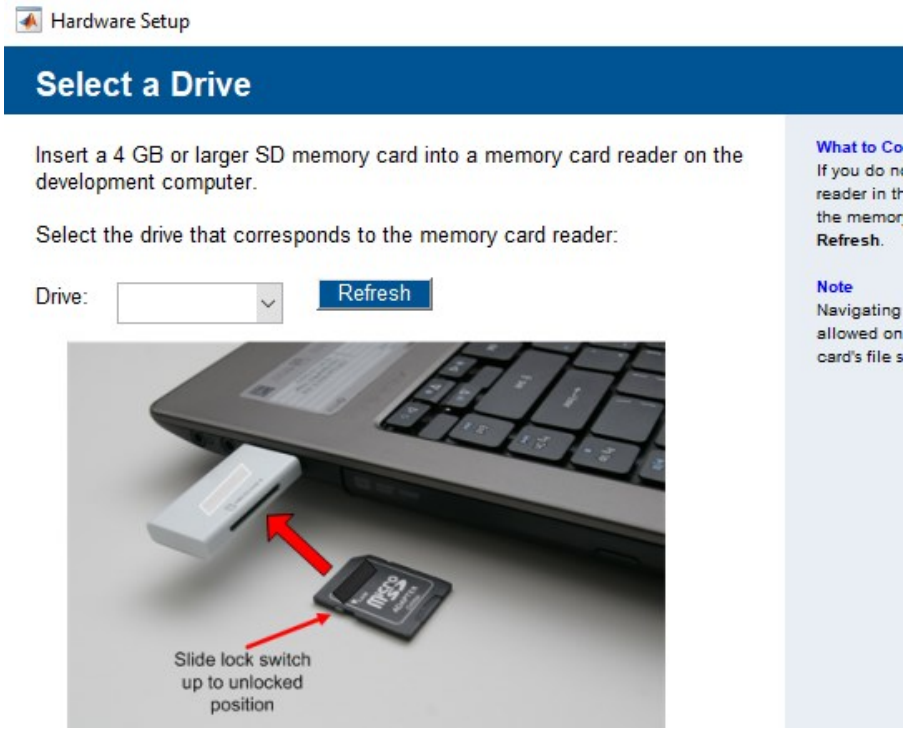
Kuva 8. Laitteen valintanäkymä

Valitaan kuvan Kuva 9 osoittamasta valikosta “Connect to LAN or home network”, jotta ZedBoard löydetään automaattisesti, kun kummatkin laitteet kytketään samaan verkkoon. Tämä helpottaa isäntäkoneen konfiguroimista.



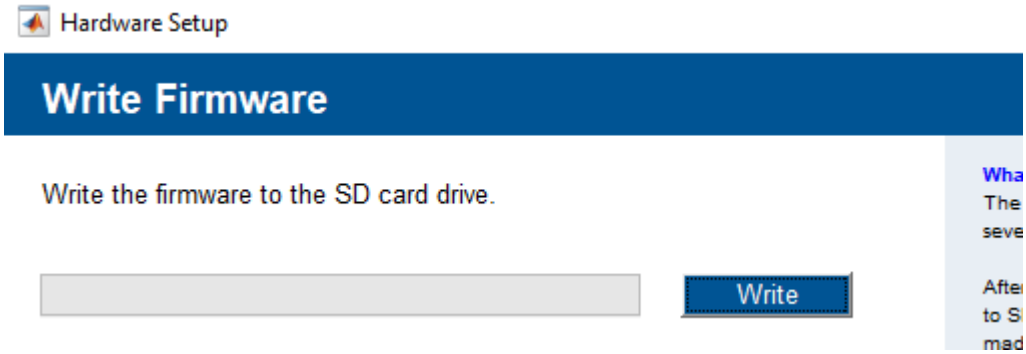
Kuva 9. Verkkoasetukset

Laitetaan SD-kortti tietokoneeseen kiinni ja valitaan sille määrätty kirjain kuvassa Kuva 10 näkyvästä valikosta.



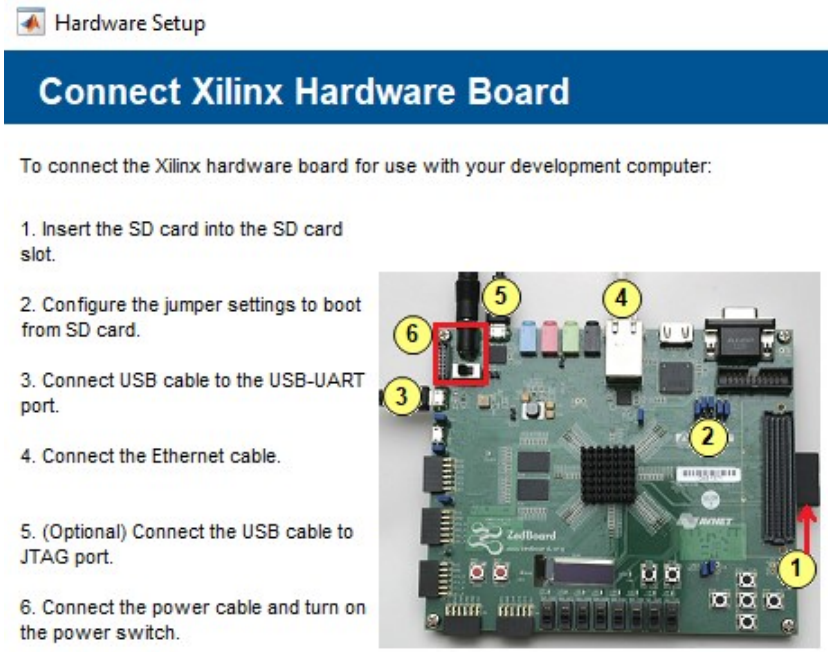
Kuva 10. SD-kortin valinta

Painetaan ‘Write’ kuvassa Kuva 11 näkyvästä valikosta.



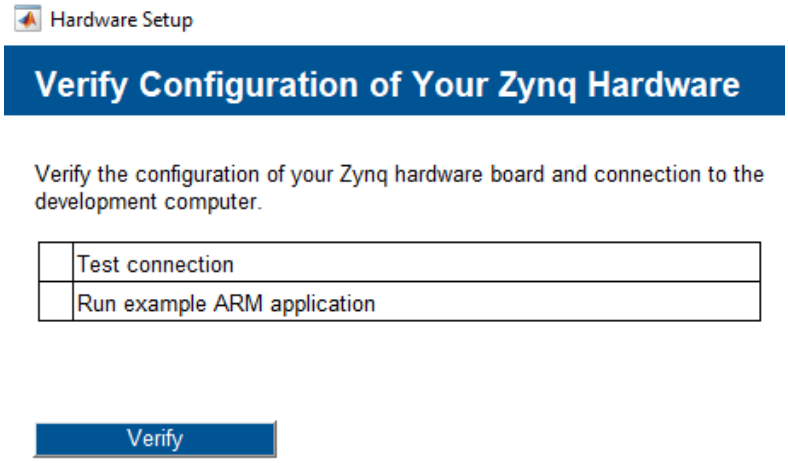
Kuva 11. SD-kortin kirjoitusikkuna

Kytketään johdot ja konfiguroidaan siltausasetukset kuvassa Kuva 12 nähtävällä tavalla.



Kuva 12. Zedboardin kytkennät

Varmistetaan, että asetukset ovat tehty oikein painamalla “Verify” kuvassa Kuva 13 näkyvässä ikkunassa.



Kuva 13. Varmistusikkuna

5.2 FPGA-suunnittelu

FPGA-ohjelma rakennetaan Simulinkissa käyttäen pääosin HDL Coderin kirjastosta löytyviä lohkoja. Ohjelmaan rakennetaan seuraavat tulot ja lähdöt kuvan Kuva 14 osoittamalla tavalla:

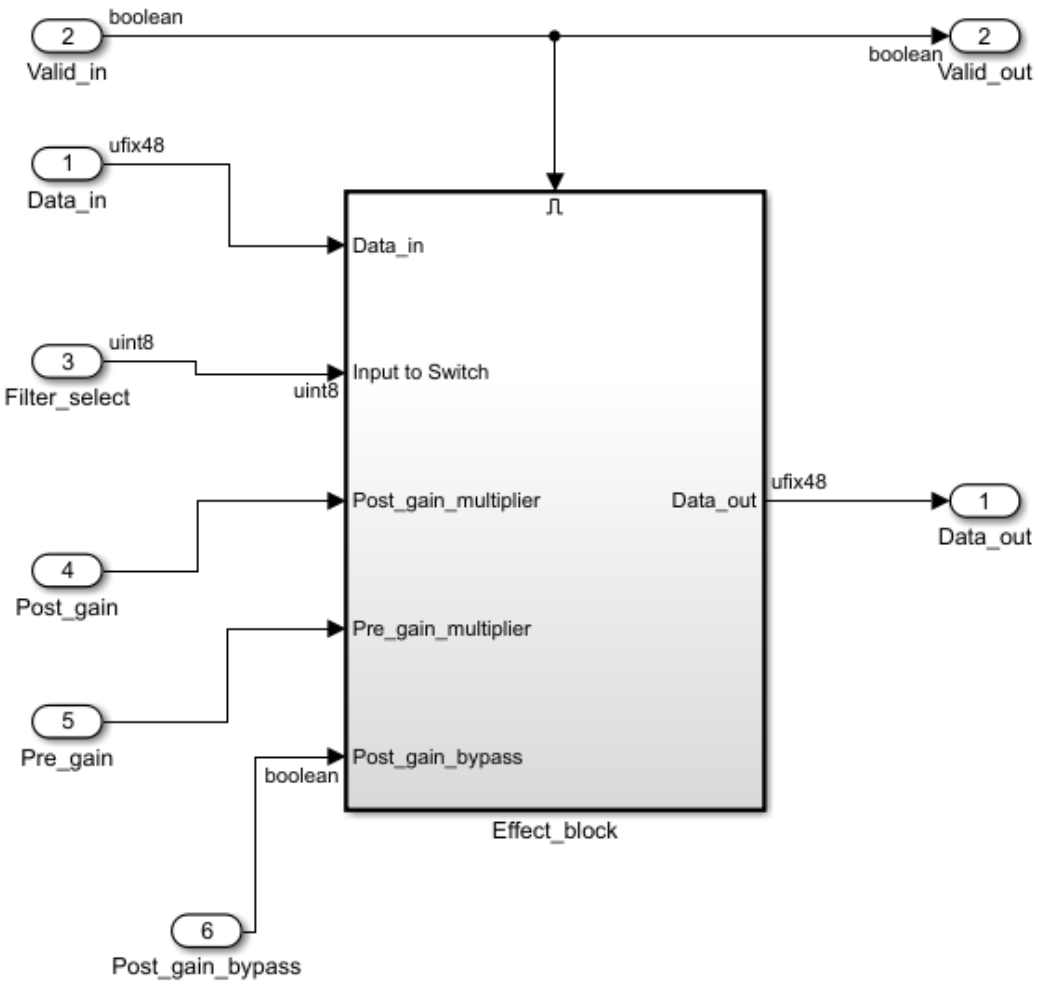
Tulot

- Valid_in
  - Kertoo onko sisään tuleva äänidata kelpaavaa äänidataa. Efektin tuottava signaalinkäsittelylohko laitetaan tämän perusteella päälle tai pois.
- Data_in
  - Sisään tuleva AXI4 audio datajono.
- Filter_select
  - Kokonaisluku, jolla valitaan päällä oleva efekti.
- Pre_gain
  - Pre_gain efektilohkon vaatima vahvistuskerroin. Käytetään säröttämiseen.

- Post_gain
    - Efektien jälkeen oleva vahvistuskerroin. Toimii äänenvoimakkuuden säätimenä.
- Post_gain_bypass
    - Kytkin, jolla voidaan ohittaa Post_gain lohko

Lähdöt

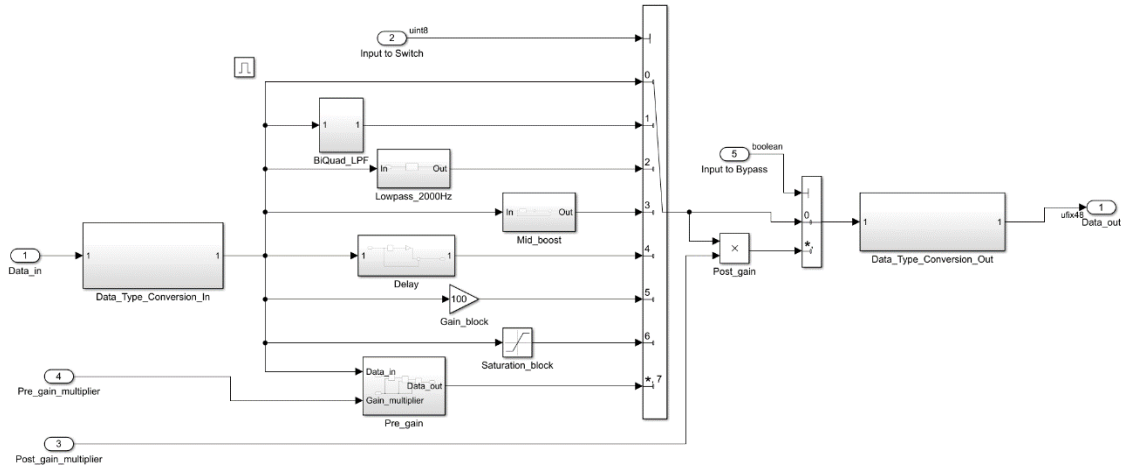
- Valid_out
    -
- Data_out
    - Ulos menevä efektien muuttama AXI4 audio datajono.



Kuva 14: Efektilohkon tulot ja lähdöt

5.2.1 Efektilohkon sisältö

Efektilohkossa nähdään kaikki toteutetut efektit. Data_in portista saadaan efektilohkoon tuleva äänisignaali. Tämä muutetaan efekteille sopivaan muotoon. Kaikkien efektien ulostulo menee kytkimeen, jolla voidaan valita efekteistä yksi vuorollaan päällle. Tämän kytkimen ulostulo menee päävoimakkuudensäätölohkoon, josta se viedään datatyyppimuunnoksen jälkeen ulos efektilohkosta. Efektilohkon sisältö on nähtävissä kuvassa Kuva 15. Efektilohkon sisältämistä efekteistä kerrotaan enemmän seuraavissa alakappaleissa.



Kuva 15: Yleiskuva efektilohkon sisällöstä

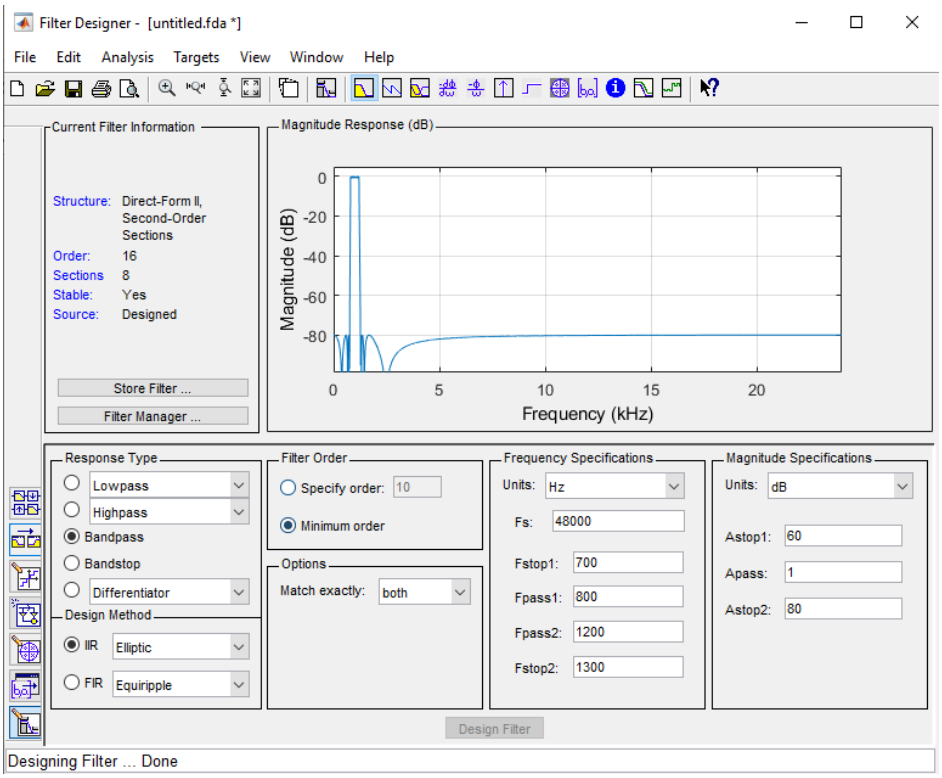
5.2.2 Taajuuskorjaussuodin

Taajuuskorjaus, eli ekvalisaatio on prosessi, jossa eri taajuusalueiden voimakkuuksia voidaan nostaa tai laskea. Taajuuskorjausta käytetään musiikkituotannossa efektinä, mutta myös ei-haluttujen taajuuksien voimakkuuden pienentämiseen tai jopa kokonaan poistamiseen. [17]

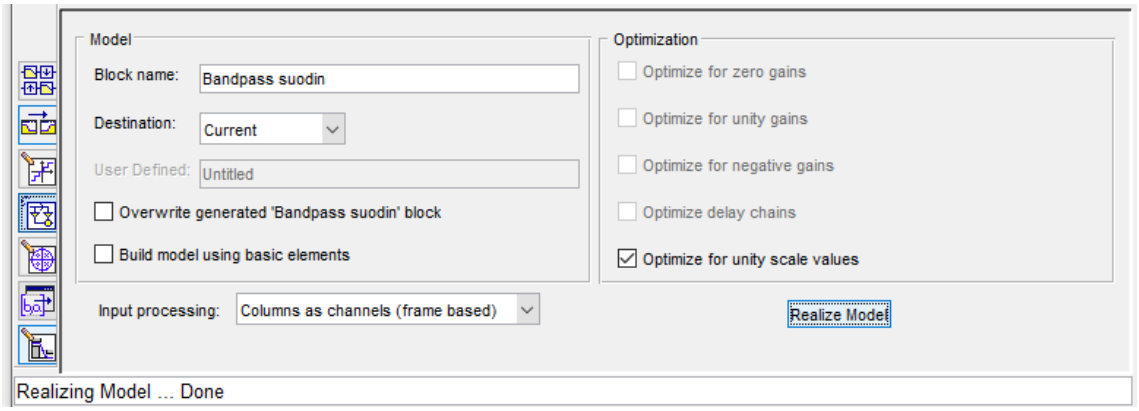
Signaalin käsittelyssä dynaamisen järjestelmän impulssivaste tarkoittaa järjestelmän vastetta kun sille syötetään nopea signaali, jota kutsutaan impulssiksi.

Impulssivasteita käytetään hyväksi vastekorjaussuotimissa, jotka ovat hyvin suuressa osassa musiikin prosessoinnissa ja tuottamisessa.

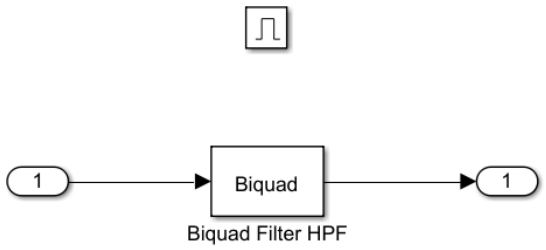
Luodaan esimerkkiprojektin suotimien tilalle omia suotimia käyttäen MATLABin Filter Designeria. Esimerkki Filter Designeristä kuvassa Kuva 16. Tuodaan luodut suotimet projektin kaaviokuvaan käyttämällä kuvassa Kuva 17 näkyvää Filter Designerin Realize Model -näkymää. Laitetaan luodut suotimet projektissa kuvassa Kuva 18 näkyvän esimerkkisuotimien tilalle. Vaihdetaan suotimen arvot sopivaksi. Esimerkki suotimen datatyyppiarvoista kuvassa Kuva 19.



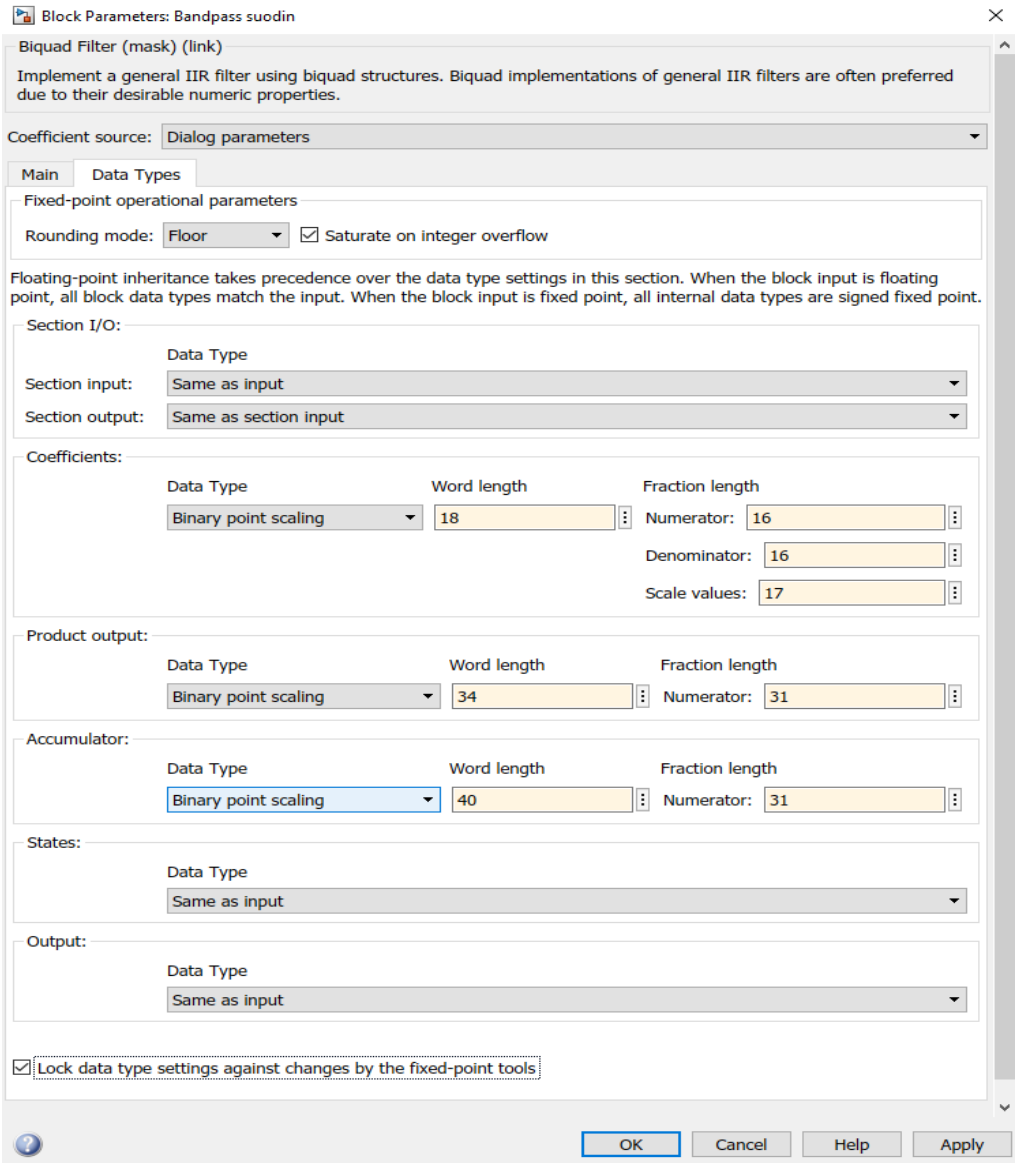
Kuva 16: Esimerkki Filter Designerin näkymästä



Kuva 17: Realize Model -näkymä



Kuva 18: Esimerkkiprojektin ylipäästösuodin



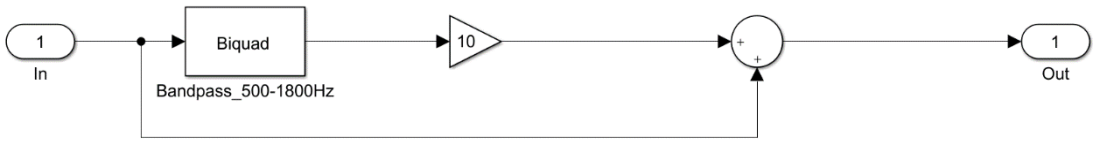
Kuva 19: Esimerkki suotimen datatyyppiarvoista

Kuvasta Kuva 15 nähtiin tässä työssä luotuja suotimia. Efektivalitsimen portteihin 1, 2 ja 3 menevät efektit ovat taajuuskorjaimia.

Porttiin 1 menevä suodin on esimerkkiprojektin alipäästösuodin, joka leikkaa yli 3kHz:n taajuudet pois.

Porttiin 2 menevä suodin on alipäästösuodin joka leikkaa yli 2kHz:n taajuudet pois.

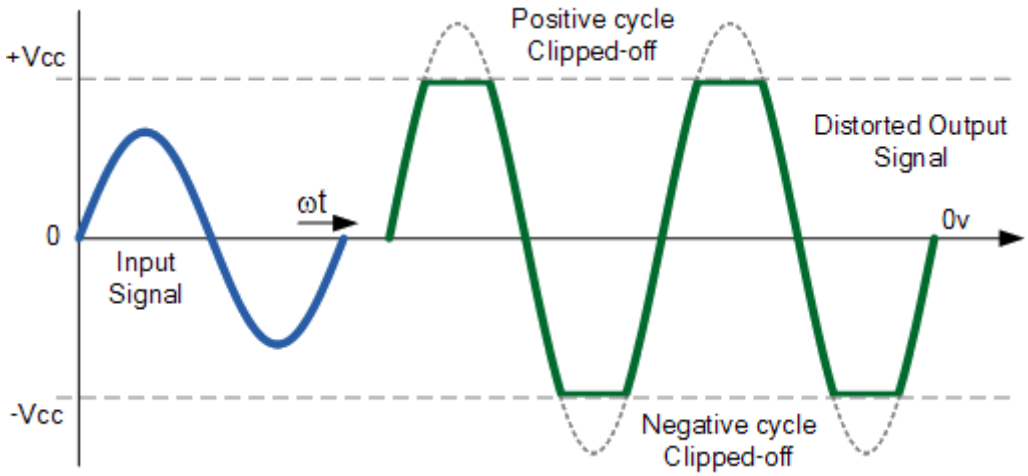
Porttiin 3 menevä suodin on keskitaajuusaluevoimistin, joka voimistaa 500–1 800Hz taajuuksia. Kuvassa Kuva 20 voidaan nähdä keskitaajuusaluevoimistinsuotimen (mid-boost) lohkokaavio. Lohkokaaviossa oleva suodinlohko suodattaa kaikki paitsi keskitaajuudet. Tämän signaali vahvistetaan sopivan vahvuiseksi, ja lisätään alkuperäiseen signaaliin, jolloin saadaan lopputuloksena signaali, jossa keskitaajuudet ovat hieman voimakkaammat kuin alkuperäisessä signaalissa. Tätä voidaan käyttää kitaraefektinä esimerkiksi soolojen aikaan, jotta saadaan soolo kuulumaan paremmin muiden instrumenttien seasta.



Kuva 20: Lohkokaavio keskitaajuusaluevoimistimesta

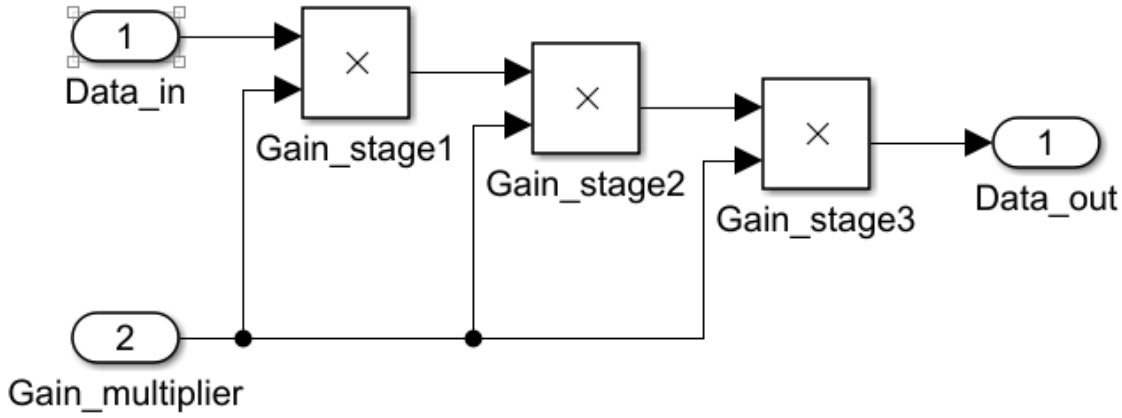
5.2.3 Säröefekti

Kitaraefekteissä säröytyminen tarkoittaa yleensä jonkinlaista signaalin saturaatiota. [18] Esimerkki signaalin saturaatiosta voidaan nähdä kuvassa Kuva 21, jossa siniaallon kärjet leikkautuvat kanttiaaltomaisiksi.



Kuva 21: Esimerkki signaalin saturaatiosta

Kuvassa Kuva 22 nähtävissä oleva särölohko on toteutettu kertolaskulohkoilla, jossa signaali kerrotaan liukuluvulla. Kun kerrottu arvo menee yli resoluution maksimiarvosta, signaali saturoituu maksimiarvoon, jolloin syntyy säröä. Kertolaskulohkoja on kolme peräkkäin, jotta saataisiin enemmän saturaatiota aikaiseksi.



Kuva 22: Särölohko

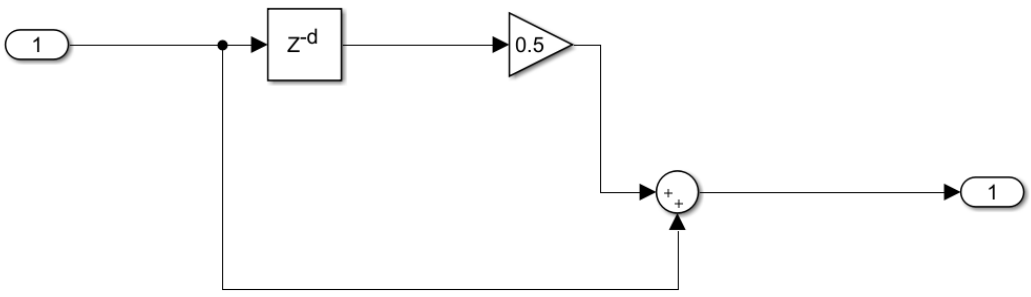
5.2.4 Voimakkuussäätö

Voimakkuussäätö luotu yksinkertaisella kertolaskulohkolla efektien jälkeen, ennen datatyyppikonversiota. Kertolaskulohkossa kerrotaan signaali Post_gain-

liukulukuarvolla. Nämä liukulukuarvot ovat yleensä jotain nollan ja yhden väliltä, mutta suurempia lukuja voidaan myös käyttää, jos halutaan säröttää signaalia myös efektien jälkeen. Post_gain-sisääntuloarvo määritellään laitteeseen ohjelmoitavan ohjelman yhdeksi sisääntuloksi, jotta sitä voidaan muokata, joko fyysisillä nappuloilla tai verkon kautta ZedBoardin prosessorin läpi suoraan MATLABista.

5.2.5 Kaikuefekti

Kaiku syntyy, kun jo toistettu ääni toistetaan uudelleen myöhemmin. Musiikin tuotannossa on yleensä haluttua, että kaikuefekti on myös hiljaisempi kuin alkuperäinen ääni. Esimerkki kaikuefektistä MATLABissa luotuna on kuvan Kuva 23 mukaan viivelohko, johon annetaan viiveen määrä näytteissä. Annetaan viiveen määräksi 44 100 näytettä, joka vastaa 44,1kHz:n näytteenottotaajuudella yhtä sekuntia. Laitetaan viivelohkon jälkeen vahvistinlohko, jolla pienennetään signaalin voimakkuus puoleen. Seuraavaksi tulevassa summalohkossa summataan alkuperäinen signaali ja viivästetty signaali toisiinsa ja viedään ne ulos lohkosta.



Kuva 23: Kaikuefektilohko

5.3 MATLABista ZedBoardiin

MATLABissa luodusta kaaviokuvasta generoidaan HDL-koodia, joka siirretään ZedBoardiin.

Klikkaa hiiren oikealla Audio_Filter blockia ja valitse HDL Code -> HDL Workflow Advisor

HDL Workflow Advisorissa valitaan seuraavasti:

1.1 Set Target Device and Synthesis tool

Target workflow: IP Core Generation

Target platform: ZedBoard

Synthesis tool: Xilinx Vivado

Ajetaan osio painamalla ‘Run This Task’

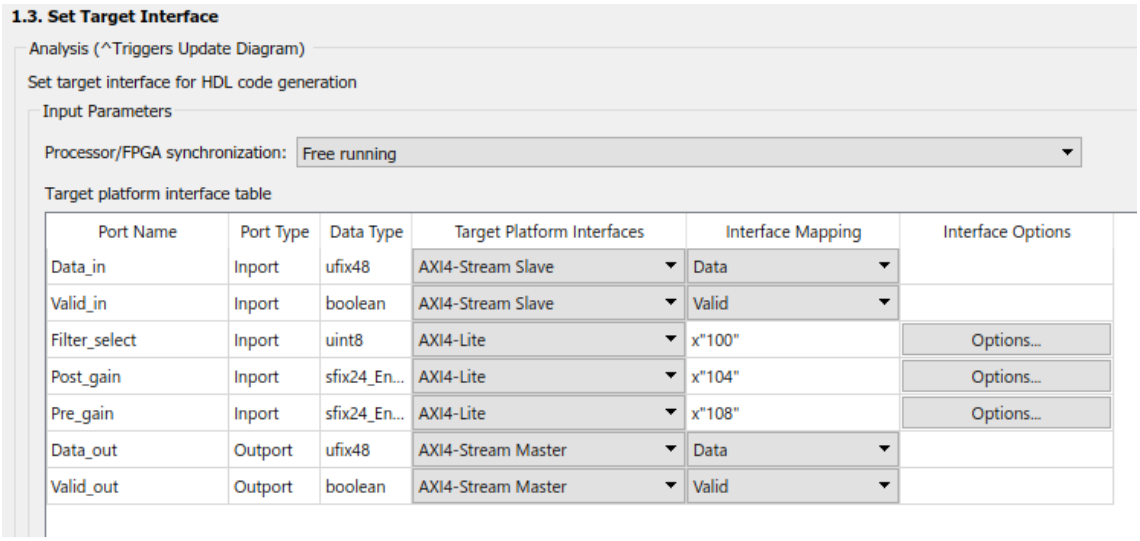
1.2 Set Target Reference Design

Reference design: Audio System with AXI4 Stream Interface

Ajetaan osio painamalla ‘Run This Task’

1.3 Set Target Interface

Valitaan kuvan Kuva 24 mukaan sisäänmenoiksi AXI4-Stream Slave ja ulostuloiksi AXI4-Stream Master, jotka vastaat sisäänmenoja ja ulostuloja AXI4-arkkitehtuurissa. Säädettäville parametreille valitaan AXI4-Lite, jonka kautta ARM-prosessori osaa muuttaa FPGA-ohjelman parametreja.



Kuva 24: AXI4-sisäänmenot ja -ulostulot

Ajetaan osio painamalla ‘Run This Task’

3.2 Generate RTL Code and IP Core

Ajetaan osio, ja kaikki edeltävät joita ei olla ajettu klikkaamalla osiota hiiren oikealla ja valitsemalla ‘Run to Selected Task’

4.1 Create Project

Ajetaan osio painamalla ‘Run This Task’

4.2 Generate Software Interface

Ajetaan osio painamalla ‘Run This Task’

4.3 Build FPGA Bitstream

Ajetaan osio painamalla ‘Run This Task’

4.4 Program Target Device

Programming method: Download

IP Address: ZedBoardin IP-osoite. Voidaan selvittää esimerkiksi MATLAB komentorivillä komennolla `h = zynq` kuvan Kuva 25 mukaan.

```
>> h = zynq

h =

  LinuxShell with properties:

    IPAddress: '192.168.1.50'
     Username: 'root'
         Port: 22
```

Kuva 25: Esimerkkitulostus MATLABin zynq komennosta

SSH Username: root

SSH Password: root

Ajetaan osio painamalla ‘Run This Task’

Tämän jälkeen aukeaa toinen Simulink-instanssi, josta voidaan painaa “Monitor & Tune”, jolloin tähän instanssiin tehdyt muutokset tapahtuvat reaaliajassa ZedBoardilla olevassa logiikassa.

5.4 Mallin muuttaminen

Tehdään muutoksia kaaviokuvaan. Kuvaan lisättäviä HDL-lohkoja voidaan löytää Simulinkistä ‘Library Browser’-valikon alta. Vain HDL yhteensopivat lohkot toimivat. Näitä ovat esimerkiksi HDL Coderin ja DSP System Toolbox HDL Support lohkot.

Kaaviokuvan muuttamisen jälkeen ajetaan HDL Workflow Advisorista uudelleen kohdat 3.2 ja kaikki kohdat osasta 4.

6 Testaus

Testausta tehdään ZedBoardilla ajon lisäksi myös MATLABin simulaatiossa. Soitettaessa ääntä simulaation läpi esimerkkiprojektin suotimet toimivat mallikkaasti, mutta Filter Designerillä tehdyt suotimet tuottavat säröytynyttä ääntä. Katsottaessa spektrogrammilla voidaan kuitenkin nähdä, että myös Filter Designerillä tehdyt suotimet laskevat äänentasoa niille määrätyillä alueilla, vaikka ääni onkin muilta alueilta säröytynyttä. Säröefektit toimivat oikein ja vastaavat kertoimen arvoihin oletetusti. Myös päävoimakkuudensäätö toimii oikein, ja toimii myös särönä efektien jälkeen tarvittaessa. Kaikuefektiä kokeiltaessa simulaatioon syntyy katkoja prosessointiviiveen vuoksi, mutta haluttu efekti on kuultavissa. Ottaen huomioon katkojen tuoman latenssin myös kaikuefektille annettu viive näyttää vastaavan yhtä sekuntia.

Vietäessä ohjelmaa ZedBoardille huomataan, että se ei onnistu. Tämä johtuu siitä, että kaikuefektin vaatima muistimäärä on niin suuri, että ZedBoardissa ei riitä muistilohkot. Kun kommentoidaan kaikuefekti pois lohkokaaviosta, saadaan ohjelma vietyä ZedBoardille. Soitettaessa ääntä ilman kaikkia efektejä ja voimakkuudensäätöjä, voidaan huomata, että ääni kuuluu ulostulosta samanlaisena kuin sisään mentäessä. Voidaan siis todeta, että ZedBoardille asetetut portit ovat oikein, ja ohjelman tyyppimuunnokset toimivat oikein. Käytettäessä etuastesäröä, tai päävoimakkuuden säätöä voidaan huomata, että ääni katoaa lähes kokonaan niin, että vain välillä kuuluu pieniä säröytyneitä osia alkuperäisestä äänestä.

7 Tulokset

Työn tavoitteena oli luoda digitaalinen kitaraefektipedaali käyttäen hyödyksi FPGA-piirejä ja MATLAB-ohjelmistoa digitaaliseen signaalinkäsittelyyn.

Työn tuloksena FPGA-piirille saatiin asennettua tarvittavat alustukset ja suunniteltu ohjelma MATLAB-ympäristön kautta. MATLAB-simulaatiossa saatiin toimimaan erityyppiset säröefektit, äänenvoimakkuuden säätö, kaikuefekti ja esimerkkiprojektin suotimet. Filter Designerillä luodut suotimet tuottivat aluksi päänvaivaa, mutta kokeilemalla erilaisia asetuksia päästiin lähemmäksi oikeaa toiminnallisuutta. Havaittiin silti, että näissä itse tehdyissä suotimissa esiintyy vielä pientä säröytymistä ja muita häiriööääniä.

FPGA-piirillä todettiin toimivaksi prosessoimattoman signaalin läpikulku. Säröefektit päästivät läpi vain pienen määrän ääntä, ja ääni oli rikkoutunutta. Kaikuefektiä ei saatu lisättyä FPGA-piirille asennettavaan ohjelmaan, koska valitulla FPGA-piirillä ei ollut tarpeeksi muistia käytettävissä tähän tarkoitukseen.

Jatkokehityksenä voitaisiin selvittää tarkemmat parametrit suotimien luontiin, jotta voitaisiin luoda toimivia suotimia, joko Filter Designerilla tai käsin. Voitaisiin myös tarkastella HDL Coderin luomaa VHDL-koodia ja siitä syntesoitavaa mallia, jotta saataisiin selville tarkka kohta, jossa signaali rikkoutuu. Tämän jälkeen voitaisiin miettiä korjaustoimenpiteitä edellä mainittuihin signaalin rikkoutumiskohtiin.

Lähteet

[1] D. L.-W. Roger Beardsley, ”CHARM,” [Online]. Available: https://charm.rhul.ac.uk/history/p20_4_1.html. [Haettu 30 11 2021].

[2] Los Senderos Studio, LLC, ”Los Senderos Studio,” [Online]. Available: https://www.lossenderosstudio.com/article.php?subject=10. [Haettu 30 11 2021].

[3] zZounds.com, ”Gator GPB-BAK-1 Aluminum Guitar Pedalboard,” [Online]. Available: https://www.zzounds.com/item--GATGPBBAK1. [Haettu 16 11 2021].

[4] ”The Ultimate Guide To Guitar Pedal Signal Chain,” [Online]. Available: https://www.rockstockpedals.com/the-ultimate-guide-to-guitar-pedal-signal-chain/. [Haettu 26 10 2021].

[5] ScienceDirect, ”Reconstruction Filter,” ScienceDirect, [Online]. Available: https://www.sciencedirect.com/topics/engineering/reconstruction-filter. [Haettu 26 10 2021].

[6] Y.-C. Wang ja V. K. Garg, The Electrical Engineering Handbook, Elsevier Inc, 2005.

[7] MathWorks, ”MATLAB,” MathWorks, [Online]. Available: https://se.mathworks.com/products/matlab.html. [Haettu 23 10 2021].

[8] MathWorks, ”Simulink,” MathWorks, [Online]. Available: https://se.mathworks.com/products/simulink.html. [Haettu 23 10 2021].

[9] MathWorks, ”Which versions of Xilinx Vivado are supported with which release of HDL Coder?,” MathWorks, [Online]. Available: https://se.mathworks.com/matlabcentral/answers/518421-which-versions-

of-xilinx-vivado-are-supported-with-which-release-of-hdl-coder. [Haettu 23 10 2021].

[10] Penn Engineering, ”VHDL Tutorial,” Penn Engineering, [Online]. Available: https://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html. [Haettu 23 10 2021].

[11] Xilinx, ”What is an FPGA?,” Xilinx, [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html. [Haettu 23 10 2021].

[12] G. Simues, ”A Guide to the Business Analysis Body of Knowledge 2.0,” p. 102, 2009.

[13] Xilinx, ”Zynq-7000 SoC,” [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. [Haettu 2021 11 24].

[14] Xilinx, ”ZedBoard,” [Online]. Available: https://www.xilinx.com/products/boards-and-kits/1-8dyf-11.html. [Haettu 2021 11 24].

[15] PYNQ, ”PYNQ,” [Online]. Available: http://www.pynq.io/. [Haettu 2921 11 24].

[16] Pulu Studio, ”DI-boksi,” [Online]. Available: https://www.pulustudio.com/kirjasto/musiikkiteknologia/di-boksi/. [Haettu 2021 11 24].

[17] J. D. R. Vesa Välimäki, ”All About Audio Equalization: Solutions and Frontiers,” 2016.

[18] S. Temme, ”Audio Distortion Measurements,” p. 2, 1992.