TARTU ÜLIKOOL

LOODUS- JA TEHNOLOOGIATEADUSKOND

TEHNOLOOGIAINSTITUUT

Andres Randmaa

FPGA KASUTAMINE DIGITALISEERITUD ANALOOGHELI VISUAALSEKS KUVAMISEKS

Bakalaureusetöö (12 EAP)

Juhendaja: Margus Rosin, MSc

Kaitsmisele lubatud .....

Juhendaja .....

allkiri, kuupäev

Tartu 2013

Sisukord

1. Sissejuhatus

Muusika transleerimine noodipildiks on muusikute jaoks vaevanõudev töö, mida on tehtud juba aastasadu. Palju lihtsam oleks, kui leiduks võimalus mängida viis sisse, salvestada see arvutis või mõnel muul seadmel ning lasta tarkvaral kuvada seda noodipildis. Kahjuks ei leidu veel tänapäeval selliseid programme. Analoogsete helisignaalide tõlgendamine ning helikõrguste ja rütmide täpne kirjeldamine noodipildiks on keskmisest keerukam ettevõtmine.

Kõige lähedasem viis, kuidas tänapäeval muusikat tarkvara abil transleerida, on näiteks MIDI failide koostamine. Tehnikaid, nagu *vibrato* või venitamine, on raske tuvastada ning need tekitavad palju vigu helikõrguse ja rütmi kindlaks määramisel. Leidub tarkvaraprogramme, mis võimaldavad transleerida lindistatud heliteosed MIDI failideks, kuid need ei suuda teha tööd väga täpselt ning võimaldavad transleerida üldjuhul vaid ühehelilisi vokaale või ainult ühte instrumenti.

Käesoleva bakalaureusetöö eesmärgiks on arendada FPGA riistvaral välja eriline tarkvara, mis võimaldab muundada sissemängitava muusikalise analoogheli digitaalseks ning kuvada seda omakorda visuaalselt üle VGA monitori noodipildis. Tegemist on uurimistööga, mis on jaotatud kaheks osaks. Esimene osa kannab nime „FPGA kasutamine analoogheli digitaliseerimiseks ja töötlemiseks“ ning seda teostas kaastudeng Koit Summatavet. Esimeses osas tegeletakse signaali vastuvõtu, selle töötlemise ja töödeldud andmete edastamisega. Teine osa on käesolev töö, milles tegeletakse töödeldud andmete vastuvõtu ja nende esitamisega üle VGA kuvaril. Eesmärgi saavutamiseks oli vaja lahendada mitu ülesannet:

- õppida kasutama Digilent Inc. (edaspidi Digilent) poolt välja arendatud PMODmic mikrofonimoodulit ja seda kasutades helisignaal FPGAle töötlemiseks edastada;
- arendada välja tarkvara VHDL programmeerimiskeeles, mis töötleks uuritavaid signaale;
- koostada samas keeles vajalik tarkvara, et töödeldud signaale läbi VGA vastavalt kuvada;
- katsetada ja testida loodud süsteemi;
- arendada projektijuhtimise ja meeskonnatöö kogemust.

Uurimistöö VGA kuvamise osa tehti FPGAga, kuna eesmärgiks oli luua terviklik süsteem ühe töövahendi peal, mis hõlmaks nii signaalitöötlust kui ka kuvarit. Lisaks võimaldas see tutvuda

FPGA poolt pakutavate võimalustega pildi ekraanile tekitamiseks. FPGA võime andmeid riistvaras paralleelselt töödelda tagab helitöötluse efektiivse jõudluse. Uurimistöös kaetakse põhiliselt osa FPGA põhifunktsionaalsusest koos signaali visuaalse esitlusega.

2. Valdkonna ülevaade

Muusiku poolt esitatava loo automatiseeritud transleerimine on protsess, milles programm võtab sisendiks lindistuse ning annab väljundiks vastavalt mängitud noodid. Polüfoonilise (mitmehelilise) muusika transleerimine on arvutuslikult omakorda veel keerulisem ülesanne, mida on uuritud juba mõnda aega, kuid senimaani peetud lahendamatuks.

Me tunnetame heli, eristades hetkelisi õhurõhu muutusi väljaspool kõrvu. Need on võnkumised, mida saab reprodutseerida kõlaritega või lindistada mikrofoniga. Võnkumistega kaasnevad kujundatud mustrid viitavad nootide kolmele põhiomadusele: helikõrgus, tämber ja dünaamika.

Mustritel on tsükliline laad, mis tähendab, et need võnguvad teatud määral. Me saame kindlaks määrata signaali põhisageduse, millest tuleneb helikõrguse mõiste. Helikõrgus määrab ära, kui kõrge või madalana noot meie jaoks kõlab. Teiseks on noodi tämber, mis määrab ära, millisena noot meie jaoks kõlab. Tavaliselt pärinevad erinevad tämbrid näiteks muusikainstrumentide omavahelistest erinevustest. See tähendab, et lisaks helikõrgusele, mida teatakse põhivõnkesagedusena, on olemas veel ka teiste helide sagedused. Kõige intensiivsemad teiste helide sagedused esinevad kordsete põhivõnkesageduste korral. Kui me näiteks mängiksime A nooti (ehk la - C-ga algava diatoonilise helirea kuues heli), mis on 440 hertsi (võnget sekundi kohta), siis esimene harmoonik oleks sellest kahekordse sagedusega 880 hertsi ja teine harmoonik kolmekordse sagedusega 1320 hertsi. Harmoonikud ehk osahelid on põhitoonid koos ülemhelidega. Osahelid tekivad harmoonilise võnkumise korral sellest, et keha võngub ühel ajal nii tervikuna kui ka kahendik-, kolmandik-, neljandik- jne osadena. Lõpuks mängib rolli veel dünaamika, mis määrab ära, kui valjusti muusikainstrument mängib. See sisaldab lööke ja hajumist, mis tähendab, et kuigi noodid kõlavad alguses suhteliselt valjult, siis ajapikku nende helitugevus hajub [1].

Varasemad muusika automaatse transleerimise probleemi uurinud tööd kirjeldavad näiteks ära, kuidas „vaatamata arvukatele katsetele lahendada probleemi, ei leidu veel tänapäeval praktiliselt kasutuskõlblikke ja üldotstarbelisi süsteeme“ [2]. Teadlased on kasutanud erinevaid lähenemisi, transleerimaks kuni 3 erinevat samaaegset heli [3].

Alain de Cheveigné ja Hideki Kawahara suutsid hinnata helikõrgusi läbi kahandava protsessi, eemaldades korraga ühe noodi helispektrist. See meetod oli aga piiratud väikestest häiritustest helis, mis viisid edasi suuremate probleemideni üldisel transleerimisel [4].

Masataka Goto lõi meetodi, mis võimaldas transleerida meloodiat ja bassi, sõltumata instrumentide arvust. Goto oli võimeline transleerima õigesti kuni 80% juhtudest, ennustades ette igat võimalikku põhiheli [5].

Leidub veel mitmeid teisi lähenemisi probleemile. Gordana Velikic üritas transleerida muusikat, uurides lisaks sagedustele ka iga üksiku laine faasi [6]. Mitmed lähenemised muusika transleerimise probleemile rajanevad signaalitöötlusel, mis tähendab, et uuritakse helispektreid, tehakse matemaatilisi järeldusi ja kulgetakse sealt edasi. See ei pruugi olla parim meetod. On kasutatud ka geneetilisi algoritme koos muusika genereerimise süsteemiga, mis on toonud juba suhteliselt kõrgekvaliteedilisi tulemusi. On võimalik eristada nii helisid kui ka muusikainstrumente, mis neid produtseerivad [1].

FPGA riistvara peal loodud erinevate rakenduste vahel võiks esile tuua näiteks kitarriplokid heli moonutamiseks ja FPGA abil loodud digitaalsed süntesaatorid. FPGAd VGAga kasutavate süsteemide seas on loodud lahendusi, mis võimaldavad paigutada noote ekraanil paiknevale noodivihikule ja lasta FPGAl komponeeritud muusikat esitada. Kuna FPGA on loomult paralleelse arvutlusega, siis see on tõhus vahend kvaliteetseks ja kiireks helitöötluseks.

2.1 Kasutatud mõisted

FPGA (*Field Programmable Gate Array*) - programmeeritav loogika, mis võimaldab luua suure mittetöötamise riskiga valmisriistvara asemel riistvara, mida saab tarkvaraliselt hiljem muuta. FPGA koosneb kolmest peamisest komponendist: sisend/väljundviigud, sisemised ühendused ja loogikaplokid [7]. Programmeeritava integraalskeemi loomine on mõistlikum, kuna on võimalus hoida kokku ressursse ja viia toode kiiremini turule riistvara kas või sadu kordi uuesti ümber programmeerides [8].

VHDL (*VHSIC Hardware Description Language*) - riistvarakirjelduskeel, mis on standardiseeritud IEEE (*Institute of Electrical and Electronic Engineers*) poolt kui IEEE-STD-1076 ja hilisem versioon IEEE-STD-1164 ning mis sai alguse 1980. aastatel Ameerika

Ühendriikide valitsuse poolt alustatud programmist arendamaks väga kiireid integraalskeeme - VHSIC (*Very High Speed Integrated Circuit*). VHDLi saab kasutada nii sünteesimiseks kui ka simuleerimiseks [9].

VGA (*Video Graphics Array)* - video graafikamassiiv, vahel ka viidatud kui graafikaadapter. VGA viitab eelkõige kuvari riistvarale, mida tutvustati esmakordselt 1987. aastal IBM PS/2 tüüpi arvutitel [10]. Tänu selle laialdasele kasutuselevõtule on hakatud seda lühendit kasutama ka analoogse arvutikuvari 15-kontaktilise D-*subminiature* VGA pistikühenduse, aga ka 640x480 pikslist koosneva lahutusvõime puhul [11].

VGA konnektor – kolmerealine 15 viiguga DE-15 konnektor. Esineb paljudel videokaartidel, arvutikuvaritel ja kõrgekvaliteetsetel televiisorikomplektidel.

CP132 – FPGA töövahendil Basys on 132 kuuliga kiibikorpus [12], mille kuulid paiknevad korpuse all võrdselt 3 rea ja veeru kaupa ruudukujuliselt [13].

XC3S250 – Spartan 3-E perekonda kuuluv seade, millel on 250K loogikaelementi, 38Kb hajusmälu, 216 Kb plokkmälu, 12 korrutit, 4 DCMi (*Digital Clock Manager*) ehk digitaalset taktihaldurit ning maksimaalselt 172 sisend-väljundviiku kasutajale [14].

Pmod – odavad analoogsed ja digitaalsed sisend-väljund moodulid, mis pakuvad võimalusi analoog-digitaal ja vastupidi muundamiseks, mootorite juhtimiseks, andurite sisendina kasutamiseks ja muud [15].

RTL (*Resistor-transistor logic*) – takisti-transistor loogika.

Helivältus ehk noodivältus ehk noodi väärtus (*Note value*) - muusikas heli relatiivne kestus, mida tähistab noodikirjas noot. Heli puudumise relatiivset kestust tähistab muusika noodikirjas paus [16]. Helivältus on üks muusikalisi parameetreid.

Alteratsioonimärk (*Accidental*) - noodikirjas märk, mis muudab sellele järgneva noodi kõrgust poole või terve tooni võrra. Alteratsioonimärgid kirjutatakse alati noodi ette nii, et märgi keskkoht asetseks joonestikul kohakuti noodipeaga [17]. Võtmemärkidena näitavad samad märgid, missugune aste on kõrgendatud või madaldatud, st missuguse helistikuga on tegu.

Abijoon (*Ledger line*) - muusika noodikirja märk, mida kasutatakse nootide märkimiseks väljaspool noodijoonestikku. Ülemisi abijooni loetakse alt ülesse, alumisi ülalt alla [18].

Dünaamika (*Dynamics*) - muusikas kitsamalt õpetus muusika kõlajõu muutustest ning laiemalt muusikas toimuvatest muutumisprotsessidest. Tavaliselt räägitakse muusikas dünaamikast seoses helitugevusega, kuid muusika arenemisel või muutumisel võivad suurt rolli mängida ka muud muusikalised parameetrid, nagu näiteks harmoonia, artikulatsioon, karakter, tämber, tekstuur, vorm, kontuur. Dünaamikat tähistatakse lisaks märkidele ka itaaliakeelsete terminitega [19].

Intervall (*Interval*) - kahe heli kõrguste vaheline suhe, mis väljendub toonides [18].

Akord (*Chord*) - kolme või enama tertsidena ülesehitatud helide kooskõla. Peale tertsilise struktuuri esineb veel kooskõlasid, mis koosnevad kvartidest, sekunditest (niinimetatud kobarkooskõlad ehk klastrid) jne [18].

3. Kasutatav tark- ja riistvara

3.1 Kasutatav FPGA

Töö realiseerimiseks kasutati Digilenti poolt toodetud 250k loogikaväratitega Basys2 töövahendit, mis on varustatud Xilinx Spartan 3-E FPGAga [20]. Nimetatud töövahendi omadused loetletuna on:

- 18-bitised korrutid, 216 Kb kiiret kahepordilist plokkmälu ning kasutatavad taktsagedused üle 500 MHz;
- USB 2 port FPGA konfigureerimiseks ja andmevahetuseks (kasutades Adept 2.0 tarkvara, mida on võimalik Digilenti kodulehelt tasuta alla laadida);
- XCF02 platvormi ROM välkmälu, milles salvestatakse FPGA seadistused;
- Kasutaja poolt valitav taktgeneraatori sagedus (25, 50 ja 100 MHz) ning lisaks sokkel teise taktgeneraatori lisamiseks;
- 3 plaadi peal paiknevat pingeregulaatorit (1.2V, 2.5V ja 3.3V), mis võimaldavad kasutada väliseid vooluallikaid pingevahemikus 3.5V – 5.5V;
- 8 valgusdioodi, 4-kohaline 7-segmendiline numberindikaator, 4 nupplülitit, 8 liuglülitit, PS/2 port ja 8-bitiline VGA port;
- 4 x 6 klemmiga porti Digilenti PMOD moodulite lisamiseks;
- Tarkvara saab laadida nii üle USB kui ka välkmälust.

Joonis 1. Illustreeriv pilt Basys2 töövahendist.

Basys2 töövahend on platvorm elektroonikaskeemide koostamiseks ja rakendamiseks, mida saavad kasutada kõik, kellel on huvi omandada teadmisi digitaalsete elektroonikaskeemide loomise kohta. Basys2 on ehitatud ümber Xilinx Spartan 3-E kohapeal programmeeritavate loogikaväratite ja Atmel AT90USB2 USB kontrolleri. See lahendus võimaldab pakkuda valmisriistvara kõikvõimalike keerukusastmega ahelate ja kontrollerite koostamiseks, mida piiravad ainult FPGA kivi peal olevate loogikaelementide koguarv ning Basys2 peal olevate moodulite hulk. Viimaseid on arendusplaadi peal eri rakenduste loomiseks mitmeid, ilma et tekiks vajadus lisamoodulite järele.

Neli standardset laienduspesa võimaldavad lisada kasutajaloodud trükkplaate või Pmode. Antud projekti teises osas on heli tuvastamiseks kasutusel PmodMIC mikrofon. Kuue viiguga pesade signaalid on kaitstud läbilöökide ja lühiste eest, tagades nii pika eluea ka väiksemate kasutajate eksimuste korral. Basys2 peal töötavad sujuvalt enamik Xilinx ISE tarkvara versioonidest, kaasa arvatud tasuta WebPACK versioonid. Töövahendiga on kaasas USB kaabel, mis varustab seadet toite ja liidesega seadme programmeerimiseks.

Pildi sujuvaks kuvamiseks ekraanil tuli võtta kasutusele kvaliteetsem väline 50MHz taktgeneraator. Sobiva taktgeneraatorina sai kasutusele võetud ACH seeria kristallostsillaator [21].

3.2 Kasutatav tarkvara

FPGA programmeerimiseks vajaliku tarkvarana oli kasutusel Xilinx ISE 14.4 keskkond [22] ja tarkvara FPGAsse laadimisvahend Adept 2.0. Adepti saab tasuta alla laadida Digilenti kodulehelt. Xilinx ISE loob lihtsad võimalused FPGA programmeerimiseks, sisaldades kogu vajalikku tarkvarade paketti. Keskkond on mõeldud kasutamiseks Linuxi, Windows XP ja Windows 7 operatsioonisüsteemide peal, kuid seda saab kasutada osaliselt ka Windows 8 operatsioonisüsteemiga. Programmeerida on võimalik visuaalselt, ühendades omavahel erinevaid loogikaelemente ja realiseerida seeläbi funktsionaalsus, või programmi koodi kirjutades. Lisaks programmi kirjutamisele on võimalik disaini sünteesida, simuleerida ja Adepti kasutades seadmele laadida. FPGA programmeerimiseks on mitmekesine valik erinevaid programmeerimiskeeli, millest enamlevinud on VHDL ja Verilog. Selle projekti realiseerimiseks kasutati VHDL programmeerimiskeelt.

Adept on Digilenti poolt välja arendatud spetsiaalne tarkvara, millega saab FPGA jaoks kirjutatud koodi riistvara peale laadida. Selle erilisuseks on tema lihtsus ja kiirus. Adepti võimalused:

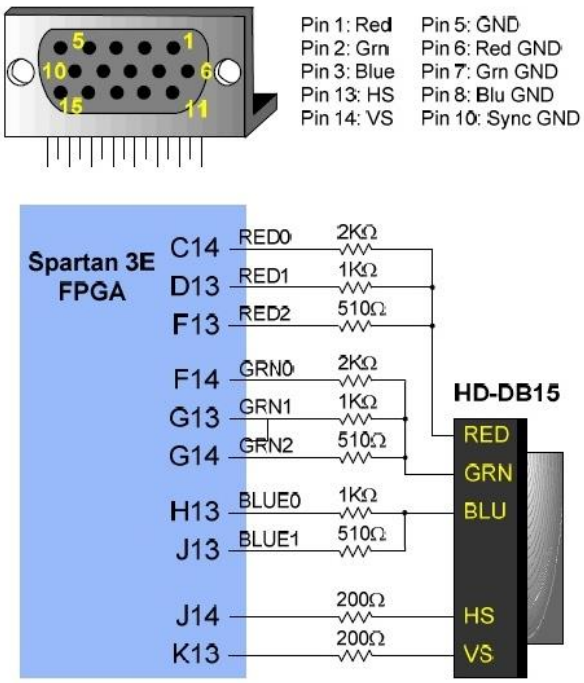
- Xilinx loogikaseadmete seadistamine, seadme otsimine, FPGA, CPLD ja PROM programmeerimine, konfiguratsioonifailide haldamine;
- Andmeedastus FPGA peale ja selle pealt, andmevoo kirjutamine registritesse ja lugemine registritest;
- Sidemoodulite kiire haldamine ja ühendamine;
- Xilinx XCFS välkseadmete programmeerimine kasutades .bit või .mcs failiformaate;
- Xilinx CoolRunner2 CPLD programmeerimine kasutades .jed failiformaate;
- Enamike Spartan ja Virtex seeria FPGAde programmeerimine kasutades .bit failiformaate [23].

4. Projekti arendus

4.1 VGA kasutamine

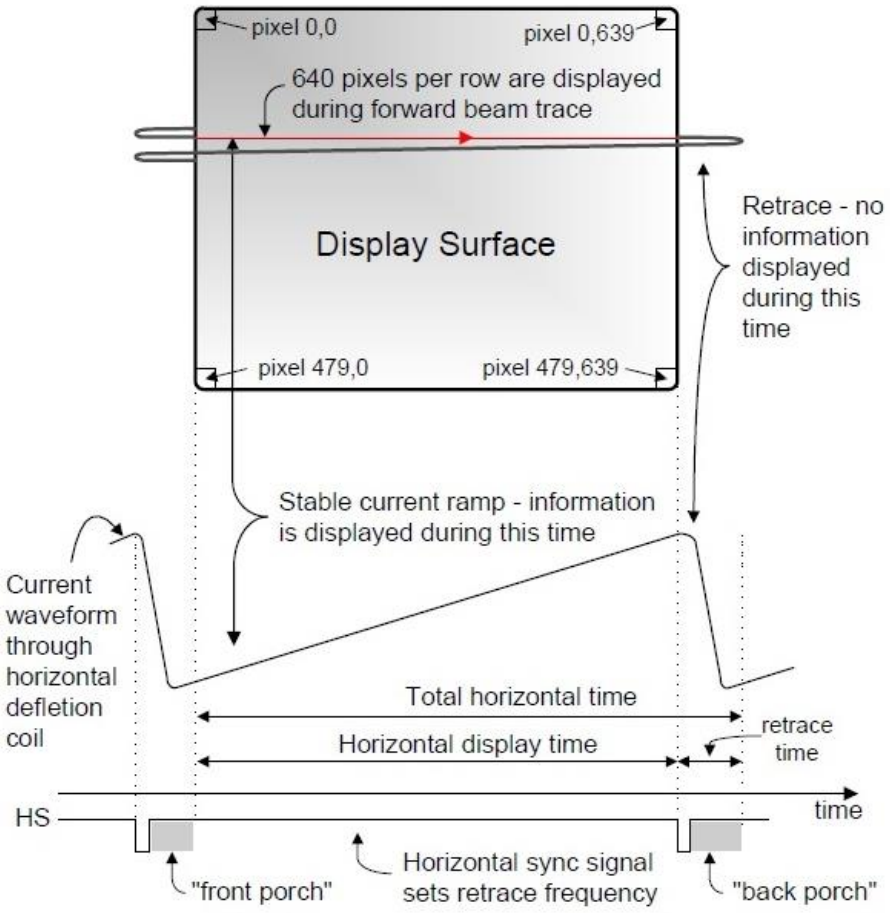
Selleks, et kuvada ekraanil pilti, oli vaja kõigepealt luua VGA kontroller. VGA kontrolleri loomiseks oli vaja teada, kuidas töötab VGA konnektor ning kuidas läbi selle andmeid ekraanil kuvatakse. VGA kontroller loodi „Intensiivkursus digitaalses loogikas“ käigus, mille juhendajaks oli prof. Mircea Dabacan, kes on õppejõud Rumeenia Cluj-Napoca nimelises tehnikaülikoolis. Nimetatud kontroller jäi bakalaureusetöö raames suures osas muutmata. Kõik järgnevad seletused ja illustratiivsed pildid on saadaval ka Basys2 käsiraamatus Digilenti kodulehel [24].

Basys2 kasutab 10 FPGA signaali, et luua 8-bitilise värvilisusega VGA port ja 2 standardset sünkroonset signaali (HS – horisontaalne sünkronisatsioon ja VS – vertikaalne sünkronisatsioon). Värvisignaalid kasutavad takisti-jagajate ahelaid, mis töötavad koos 75-ohmise VGA kuvari terminaatortakistiga, et luua kolme siiniga 2 astmes 3 eritasemelist signaali punasel ja rohelisel ning 2 astmes 2 sinisel VGA signaalil (inimsilm ei erista sinist värvi nii hästi võrreldes punase ja rohelise värviga). Joonisel 2 on toodud video värvisignaale loov järjestikune kasvavate võrdsete sammudega ahel vahemikus 0V (väljas) ja 0,7V (sees).



Joonis 2. VGA viikude selgitus ja Basys2 skeem [24].

VGA kasutamisel on vaja teada, kuidas töötab CRT ekraan (sama tehnikat kasutatakse ka uuematel LCD ekraanidel). Järgnevalt on toodud illustratiivne pilt VGA tööks vajalikest signaalidest 640x480 resolutsioonil:



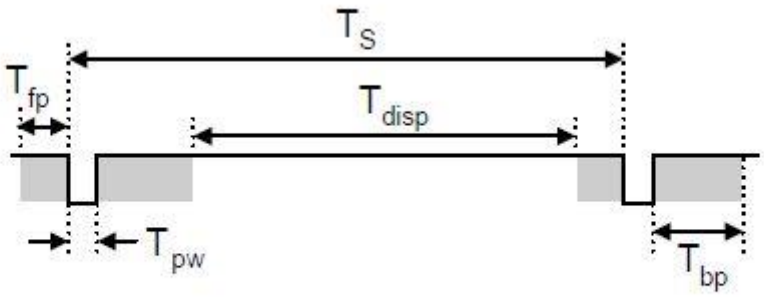
Joonis 3. VGA süsteemi signaalid [24].

Värvilised CRT ekraanid kasutavad kolme elektronkiirt pildi kuvamiseks (üks punase, üks sinise ja üks rohelise värvi jaoks). Optiline kujutis saadakse peene elektronkiire põrkumisel vastu ekraani, mille luminofooriga kaetud kiht jätab elektronkiire liikumise teest nähtava jälje. Elektronkahuris moodustunud peen suunatud elektronkiir liigub ekraanil vastavalt hälvitussüsteemi toimele. [25] Lainekujud juhitakse läbi mähiste, et tekitada magnetvälju, mis töötavad koos katoodkiirtega ning põhjustavad nende liikumist kuvaril rasterkujul horisontaalselt vasakult paremale ja vertikaalselt ülevalt alla.

Informatsiooni kuvatakse ainult siis, kui kiir liigub edasi suunas (vasakult paremale ja ülevalt alla) ning mitte ajal, kui kiir liigub tagasi kuvari vasakusse või ülemisse äärde (vt. joonis 3). Kuna elektronkiire taastamine ja stabiliseerimine on aeganõudev, siis läheb kaduma palju võimalikku kuvatavat pilti. Olenevalt kiirte suurusest, selle jälgimise sagedusest kuvaril ning algseisundisse kohandamise kiirusest on võimalik määrata kuvari resolutsioon.

VGA kontroller peab genereerima sünkroniseerivaid 3.3V (või 5V) impulsse, et seada paika sagedus voolu liikumisel läbi mähiste. Lisaks peab kontroller tegema kindlaks, et elektronkahur saaks õigel ajahetkel õige informatsiooni videopildi tekitamiseks. Rastervideokuva määrab ära ridade ja veergude arvu, mis vastavad katoodkiirte horisontaalsele ja vertikaalsele liikumisele üle ekraani. Kiire asukoht horisontaalsel ja vertikaalsel teljel määrab omakorda ära ühe pildielemendi ehk piksli. Andmed video kuvamiseks salvestatakse ühe või rohkema baidiga piksli kohta videomälus (Basys2 kasutab 3 bitti piksli kohta). Seega peab VGA kontroller videomälu indekseerima, et kuvada elektronkiire liikumisele vastavalt kindlaid andmeid kindlatel pikslitel.

VGA kontroller peab veel genereerima HS ja VS ajasignaale ning kooskõlastama videoandmeid pikslitega. Selleks kasutatakse pikslite jaoks loodud loendurit, mis määrab ära aja, kui kaua ühte pikslit kuvatakse. VS signaal määrab ära video värskendussageduse kuvaril või aja, mil ekraanilolevat pilti uuesti joonistatakse. Käesoleva projekti jaoks kasutasin 60Hz värskendussagedust ning 25MHz taktgeneraatorit pikslite kuvamiseks.



Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

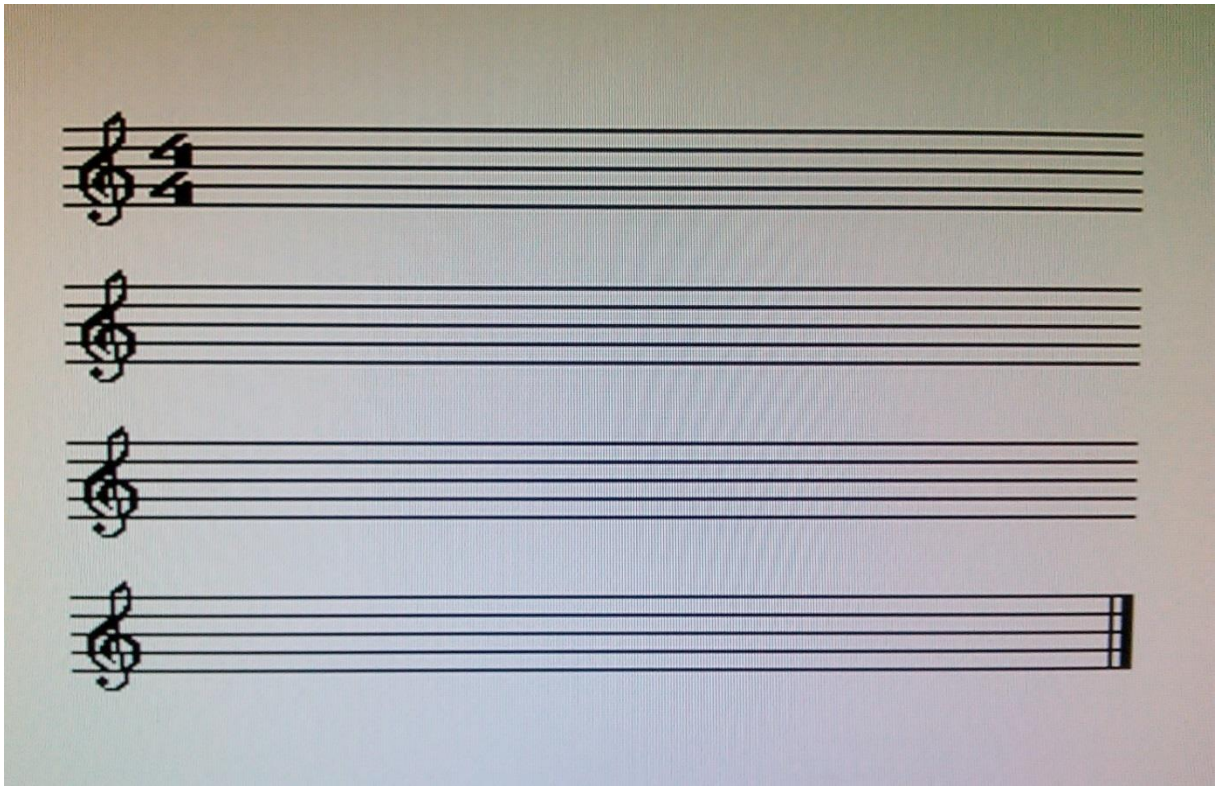
Joonis 4. VGA süsteemisignaalide ajastused 640x480 resolutsioonil [24].

Joonisel 4 on toodud vajalikud ajastused VGA süsteemisignaalide kasutamiseks 640x480 resolutsioonil. Viimases kahes veerus on toodud ajastused, mil andmeid ekraanil ei kuvata ning elektronkiir tuuakse tagasi ekraani vasakusse äärde (*front porch*, *back porch*). Aeg, mil informatsiooni ei kuvata koos impulsside kestvusega on toodud töötavate VGA kuvarite põhjal.

HS signaali ajastuseks dekodeerib VGA kontroller väljundid, mis saadakse horisontaalse sünkroniseerimise loendurist. Mainitud loendurit juhitakse 25MHz taktsagedusega (taktgeneraator pikslite kuvamiseks). Seda loendurit saab kasutada piksli asukoha leidmiseks antud real. Sarnaselt juhitakse ka VS signaale, millega saab leida mistahes rea, milles mingit konkreetset pikslit kuvatakse. Neid kahte, jätkuvalt jooksvaid loendureid, saab kasutada videomälu adresseerimiseks ekraanil piltide kuvamisel.

4.2 Pildi loomine

Kui VGA kontroller oli töökorras, sai hakata joonistama ekraanile pilti. Idee piltide joonistamiseks tuli ainest „Intensiivkursus digitaalses loogikas“ ning sama võte tuli kasutusse hiljem ka nootide joonistamisel. Esimese sammuna tuli luua ekraanile noodivihik, kuhu saab hiljem noote lisada. Noodivihikuks sai 320x480 suurune ala ekraani keskel. Seejärel tuli kuvada lehel noodiread. Viimase noodirea lõppu tähistab lõpujoon, mis koosneb ühest peenemast vertikaalsest joonest ja sellele vahetult järgnevast paksemast joonest. Noodiridade arv peab olema kooskõlas Basys2 peal olemasoleva BRAM mäluga, mis oleks suuteline mahutama piisavalt suure ala, kuhu hiljem paigutatud noodid salvestada. Viimasena tuli luua ridadele noodivõtmed, selgitamaks kasutatavat helistikku nootide kuvamisel. Noodivõtme loomiseks tuli kasutusse Microsoft Excel, milles sai joonistatud välja kõik noodivõtmele kuuluvad pikslid 16x36 suuruses alas. Kuna Excelis on kõik tühjad ruudud väärtusega 0, siis tuli kasutada noodivõtmele kuuluvate pikslite iseloomustamiseks väärtust 1. Seejärel tuli koostada Excelis algoritm, mis luges kokku kõik nullid ja ühed antud reas, moodustades ühe binaarse arvu ning konverteeris selle kümnendarvuks. Saadud arvude kasutamiseks tuli luua projektis pildi joonistamise all üks kahemõõtmeline järjend ning lisada need sinna vektoriteks konverteerituna. Edasi tuli määrata ära ala, milles piksleid joonistatakse (16x36) ning asukoht, kus seda kuvatakse ning lasta FPGAl käia järjend bitt haaval läbi, kuvades kõiki noodivõtmele kuuluvaid ühtesid mustana. Esimesele noodireale sai joonistatud samamoodi veel taktimõõt, milleks käesoleva projekti raames sai määratud vaikimisi 4/4. Loodud noodivihik on toodud joonisel 5.



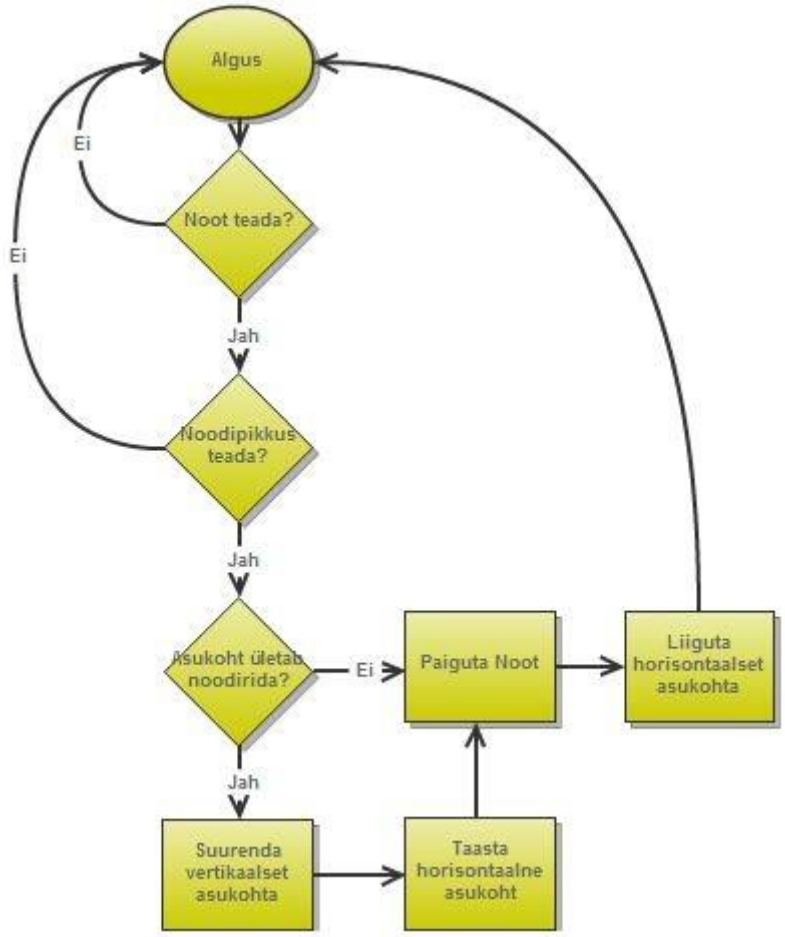
Joonis 5. Läbi VGA loodud noodivihik kuvaril.

4.3 Nootide kuvamine vastavalt signaalidele

Kui noodivihik oli loodud, sai hakata planeerima nootide noodivihikul kuvamisele. Kirjeldamaks ära kõikvõimalikud nootide ja pauside variatsioonid, tuli kasutada MS Excelit, et joonistada need välja üksteise alla (12x21). Kui erinevad variandid olid olemas, tuli lasta programmil neid järjendist otsida. Selleks, et nooti paigutada, oli vaja teada selle vertikaalset paiknemist noodiridade suhtes. Tuli luua konstandid, mis määrasid ära vertikaalse asukoha, kuhu järjendis paiknevat nooti joonistatakse. Lisaks tuli ära määrata ka horisontaalne konstant esimese noodi paiknemiseks noodireal. Kui noot on juba paigutatud, siis konstandist lähtuvalt liigutatakse uue noodi asukohta.

Töödeldud helid loetakse sisse üle kahe signaali, millest üks määrab ära, mis noodiga on tegemist, ning teine määrab ära noodivältuse. Noodi signaali järgi vaadatakse selle vertikaalne asukoht ning vältuse järgi valitakse õige kestvusega noodikuju. Järjendis paiknevad ka pauside märgistused. Juhul, kui sissetuleva signaalina antakse märku, et heli pole, siis on tegemist pausiga ning vastavalt selle kestvusele joonistatakse vastav pausimärk või mitu. Kui signaalid on vastuvõetud ning noot paigutatud, siis nihutatakse horisontaalset muutujat ühe

koha võrra edasi järgmise noodi jaoks. Selleks sai kasutusele võetud *Moore'i* olekumasin. Kui noot, mida tuleb kuvada, on teada, oodatakse selle noodivältust. Senikaua noodivihikule joonistamist ei toimu. Kui ka vältus on teada, edastatakse signaaliga informatsioon, et enne uut signaali vastu ei võeta, kuni noot on paigutatud ning horisontaalset muutujat suurendatud ühe koha võrra. Juhul, kui nimetatud muutuja ületab noodijoonestikku, siis taastatakse muutuja algseisundisse ehk noodi alguspositsiooni ning suurendatakse kõikide nootide vertikaalset paiknemist. See tagab nootide järgnevuse järgmisel noodireal.



Joonis 6. *Moore'i* olekumasin nootide paigutamiseks ekraanil.

Projekti arendamise käigus sai proovitud erinevaid meetodeid nootide paigutuse katsetamiseks. Esimene samm oli paigutada noot alguspositsioonile ning lüliti abil vahetada kahte nooti. Kui lüliti oli üleval, siis kuvati neljanda oktaavi A nooti ning kui lüliti oli all, siis neljanda oktaavi E nooti. Järgmine samm oli kasutada nuppu nootide paigutamiseks ning ühe koha võrra liigutamiseks. Kui noodi asukoht ületas noodirida, siis tuli see taastada algseisundisse. Lahenduse teostamiseks sai kasutatud olekumasinat. Kõigepealt oodati nupuvajutust ning kui nuppu vajutati, siis oodati nupu vabastamist. Kui nupp oli vajutatud ja

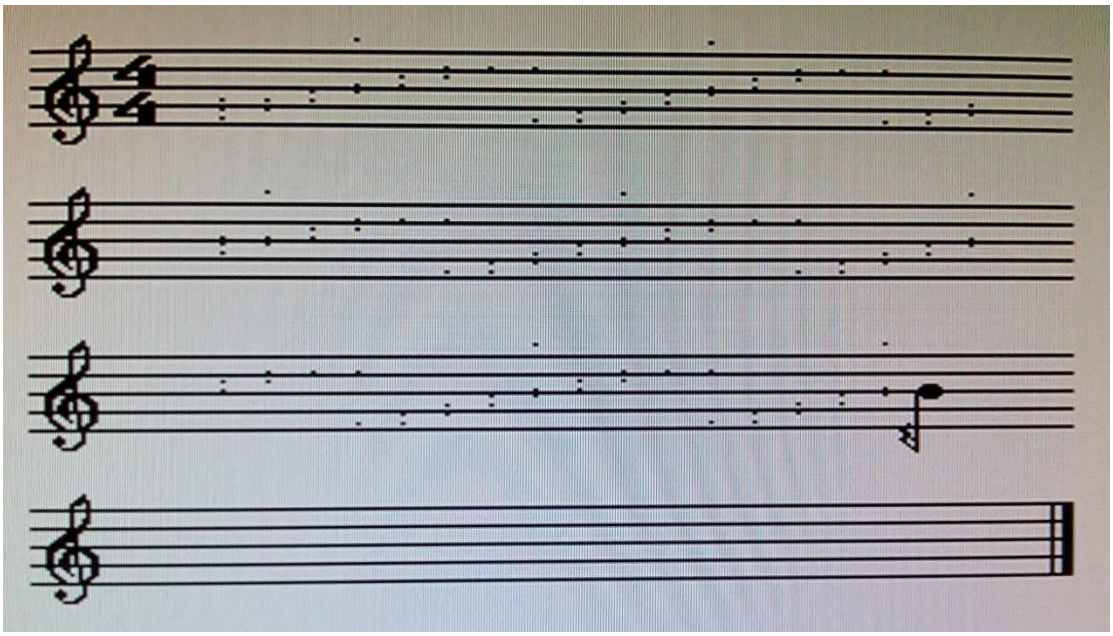
ühtlasi vabastatud, siis paigutati noot noodireale, suurendati horisontaalset asukohta ning oodati uuesti, kuni vajutatakse nuppu. Probleemiks sai asjaolu, et noodi liigutamisel üle noodirealõpu ei taastatud noodi algset positsiooni. See sai esialgu lahendatud analüüsimise tulemusel, uurides välja, mis alani noot võib liikuda, et ta algseisundisse taastataks.

Järgmine eesmärk oli saavutada noodirea vahetus noodirea lõppedes. Kui noodid olid paigutatud, siis noodi horisontaalse asukoha taastamisel suurendati noodi vertikaalset asukohta. Üheks takistuseks noodirea vahetusel sai nootide ebatäpne paigutamine teisel ja kolmandal noodireal. Probleemi lahendamiseks viidi läbi mitmeid teste ja analüüsi tulemusel ilmnes veana vigaselt defineeritud signaali ulatus. Tuli välja, et noodi vertikaalse asukoha suurendamisel ületas noot talle määratud ala, ületanud osa liideti signaali algväärtusele ja vastavalt sellele paigutati noot valele kohale. Probleem sai lahendatud korrektse signaali ulatuse määratlusega.

Kui noote oli võimalik paigutada, nootide välimust vastavalt kestvusele kuvada ning rea lõppedes uuelt realt alustada, tuli mõelda nende mälus salvestamise peale. Digitaalse loogika intensiivkursuses käsitleti plokkmälu (BRAM) ühe piksli salvestamiseks ekraanil määratud asukohas. Selleks sai kasutusele võetud Xilinx ISE poolt pakutavad mallid mälu lähtestamiseks. Antud projekti raames oli kasutusel 16k x 1 RAMB16_S1_S1 mälu. Kuna intensiivkursuse raames kasutati mälu jaoks vaid väikest ala pikslite salvestamiseks (128x128), siis nimetatud mälu uurimistöö jaoks ei sobinud ning tekkis vajadus suurema mälu järele. Selleks tuli luua mälu programmiliselt suurusega 256x256 pikslit, mis oli piisav nelja noodirea jaoks. Nootide puhul on tegemist kahemõõtmeliste järjenditega, mis tähendab, et tuli luua viis, kuidas kirjutada kuvatavad pikslid mälus õigetele mäluaadressidele. Ülesanne osutus oodatumast keerulisemaks. Raskusi tekkis järjendi osade korrektsel kirjutamisel õigetele mäluaadressidele, nagu neid ekraanil kuvati. Tulemuseks kuvati üle terve ekraani vigaseid noote. Lihtsama variandina sai võetud kasutusele üks piksel, et jätta meelde noodi varasem asukoht ja kuvada seda vastavalt noodi esinemisele noodivihikul. Noodipildi korrektne mällu kirjutamine kuulub hetkeseisuga projekti edasiarendusfaasi.

Visuaalse kuvari testimiseks nootide paigutusel, tuli luua veel lisamoodul, mis matkiks sisendina saadavaid helisignaale ja nende vältusi. Lisapõhjus mooduli loomiseks oli antud uurimistöö raames töötava signaalitöötluse ploki puudumine. Kõigepealt tuli kasutada ära FPGA süsteemi taktgeneraatorit, mis töötab sagedusel 50MHz. Kuna 50MHz nootide paigutamiseks on inimsilmale liiga kiire, siis tuli tekitada loendur, mis registreeriks takte

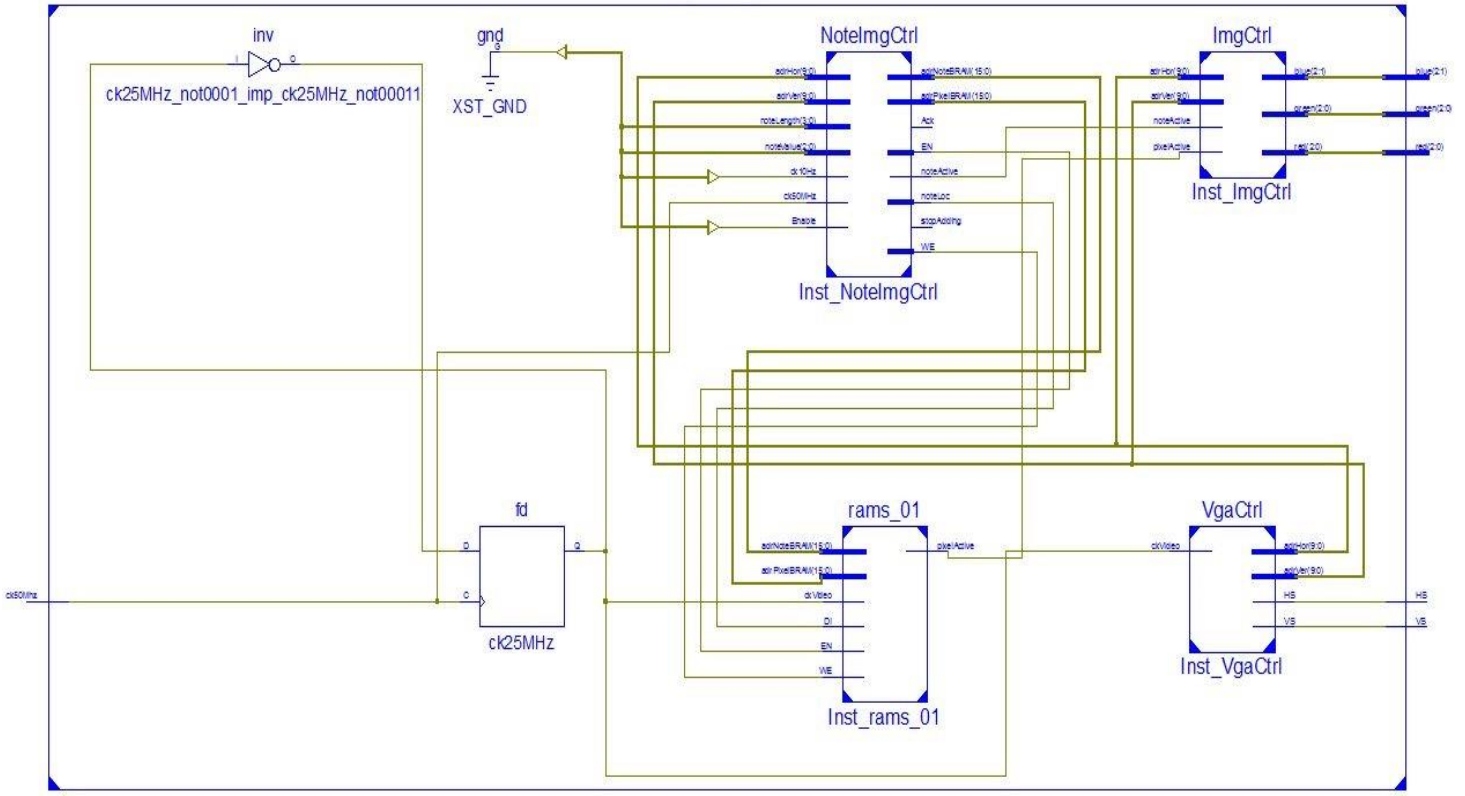
kindla aja tagant, et luua aeglasem 10Hz taktgeneraator. Aeglasemat taktgeneraatorit kasutati nootide paigutamise vaadeldavaks kuvamiseks. Kui taktgeneraator oli loodud, sai nootide paigutuseks tekitatud 2 astmes 3 vektoriga loendur, mida suurendati iga takti tagant. Vastavalt muutuja seisundile suurendatakse noodiväärtust ühe võrra neljanda oktaavi E noodist viienda oktaavi E noodini ning alustatakse algusest kuni noodivihiku lõpuni. Lisaks sellele pöörab lipuga noot alates neljanda oktaavi B-st muusikateooria kohaselt lipu tagurpidi.



Joonis 7. Nootide paigutamine VGA kuvaril. Noodile eelnevad pikslid tähistavad noodi varasemaid asukohti.

5. Funktsionaalsus ja skeem

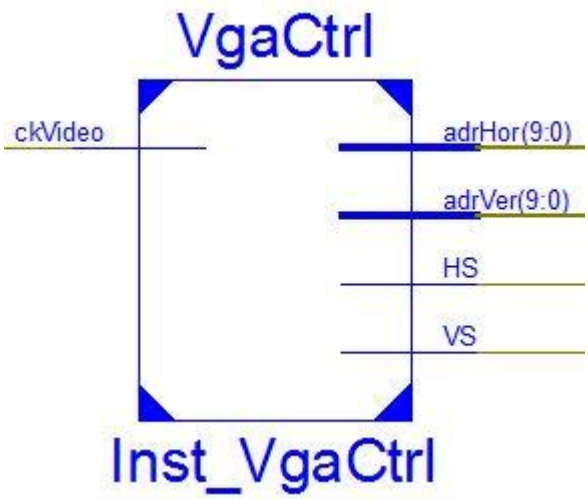
Analoogheli digitaalne kuvar koosneb neljast moodulist: VGA kontroller *VgaCtrl*, pildi kuvamise moodul *ImgCtrl*, nootide paigutuse moodul *NoteImgCtrl* ja mälumoodul *rams_01*. Joonisel 7 on toodud Xilinx ISE töövahendiga süsteemi üldskeem RTL tasandil. Kuna käesoleva uurimistöö raames pole jõutud nii kaugele, et signaalitöötlusest edastatakse visuaalse kuvari jaoks täpne info tuvastatud nootide ja vältuste kohta, siis on lisatud juurde ka täiendav moodul *SimulateSignals*, mis simuleerib nootide paigutust noodilehel. Kõigi moodulite tööd juhib Basys2 standardne 50MHz taktgeneraator välja arvatud juhul, kui pole öeldud teisiti. Süsteemi skeem, koos lisamooduliga on toodud lisas 1. Kõik nimetatud moodulid on detailsemalt lahti seletatud järgnevates alapeatükkides.



Joonis 8. Analoogheli digitaalse kuvamise süsteemi üldine skeem RTL tasandil.

5.1 VGA kontroller

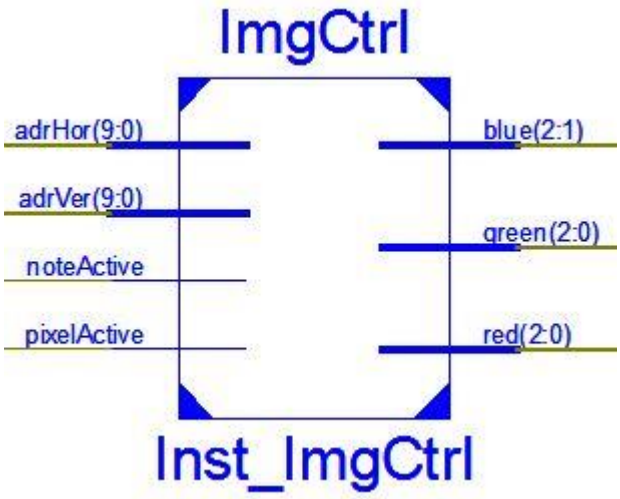
VGA kontrolleri eesmärgiks on tagada andmete õigeaegne esitamine kuvaril. Selleks on FPGA peal kaks signaali – HS ja VS (vastavalt horisontaalne ja vertikaalne sünkronisatsioon). Signaalide ajastus on toodud varasemalt joonisel 4. Mooduli väljundite hulka kuuluvad kaks vektorit – horisontaalne ja vertikaalne asukoht ekraanil. Neid kahte signaali kasutavad *ImgCtrl* ja *NoteImgCtrl* moodulid vastavalt piksli asukohale pildi joonistamiseks ekraanile. VGA kontroller töötab 25MHz taktgeneraatoriga (*ckVideo*).



Joonis 9. VGA kontrolleri moodul.

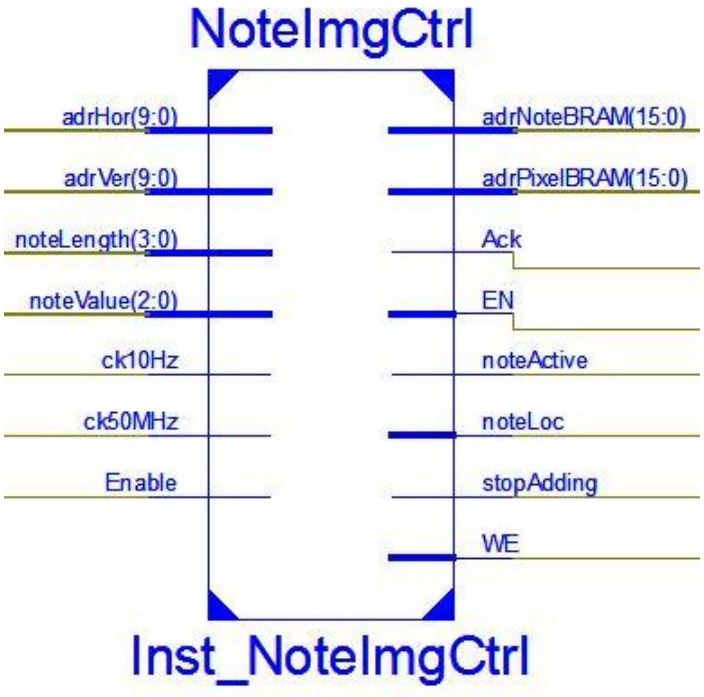
5.2 Pildi kuvamise moodul

Pildi kuvamise mooduliga laetakse kuvarile noodivihik koos noodijoonte, noodivõtmete, taktimõõdu ja lõpujoonega. Jooned ja pildid joonistatakse kasutades horisontaalse ja vertikaalse asukoha vektoreid. Nootide joonistamiseks on kaks sisendsignaali: *noteActive* ja *pixelActive*. Viimane signaal on ühenduses mäluga, kust loetakse infot, kas hetkel kuvatav piksel on aktiivne ehk kõrge seisundiga või vastupidi. Mooduli väljundsignaalideks on FPGA värvid vastavalt vektoritega 2 astmes 3 sinise ja rohelise värvi jaoks ning 2 astmes 2 sinise värvi jaoks. Käesolev projekt kasutab ainult must-valgeid värve.



Joonis 10. Pildi kuvamise moodul.

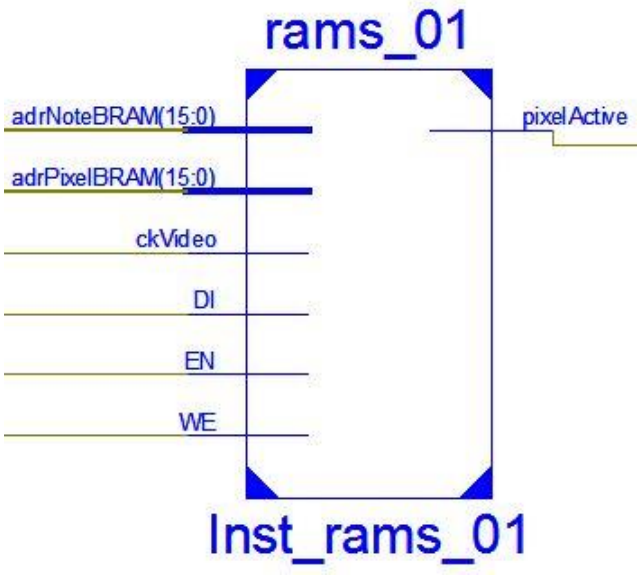
5.3 Nootide paigutuse moodul



Nootide paigutuse moodul on loodud nootide paigutamiseks noodivihikule. Kahedimensioonilises järjendis on toodud kõik vältuste variatsioonid ning signaal *noteActive* määrab ära kuvatava noodi. Noodi paigutuse ja vältuse määravad ära sisendsignaalid *noteLength* ja *noteValue*. Paigutatud noodi mällu kirjutamiseks on kaks mäluaadressi: *adrNoteBRAM* ja *adrPixelBRAM.* Selleks, et mällu kirjutamine oleks võimalik, on olemas kaks signaali – EN ja WE vastavalt mälu kasutamiseks ja kirjutamise lubamiseks. Signaal *noteLoc* on selleks, et kirjutada noodi asukohal paiknev piksel mällu. Signaalid *Enable* ja *Ack* on võetud kasutusele olekumasina kontrolliks. Nende abil tehakse kindlaks, et järgmise noodi uut asukohta ei muudeta enne, kui noodi väärtus kui ka selle vältus on teada. Vältimaks viimase noodirea ületamist on ka signaal *stopAdding*, mis peatab olekumasina töö. Aeglasem 10Hz taktgeneraator saadakse sisendina nootide paigutamist matkivast lisamoodulist.

5.4 Mälumoodul

Mälumoodulis on loodud plokkmälu suurusega 16k x 4 bitti, mis saab sisenditeks kaks mäluaadressi: *adrNoteBRAM* kirjutamiseks ning *adrPixelBRAM* lugemiseks. Lisaks saab moodul sisendiks veel hetkel joonistatava piksli väärtuse (DI) ja mälu kasutamiseks ja kirjutamiseks lubavad signaalid EN ja WE. Mooduli väljundiks on mällu kirjutatud pikslid, mis edastatakse *ImgCtrl*

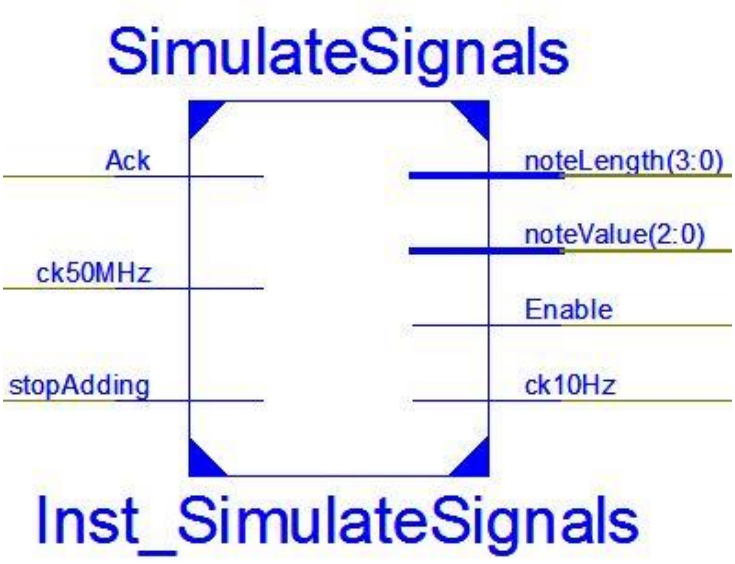


Joonis 12. Mälumoodul.

moodulile üle *pixelActive* signaali. Mälumoodul töötab 25MHz taktgeneraatoriga.

5.5 Lisamoodul nootide genereerimiseks

Käesoleva uurimistöö arendusfaas pole veel jõudnud selleni, et signaalitöötlusest edastatakse adekvaatsed andmed noodi ja selle vältuse kohta. Seetõttu on juurde lisatud moodul *SimulateSignals* nootide paigutamise simuleerimiseks. Moodulisse on loodud loendur, mis edastab vektorile *noteValue* noodi, mida kuvada. Signaal *noteLength* edastab *NoteImgCtrl* moodulile noodi vältuse. Signaaliga *noteValue* määratakse ära noot. Suhtluseks *NoteImgCtrl* mooduliga nootide paigutamisel on signaalid *Ack* ja *Enable*, mis kontrollivad olekumasinat. Enne uue noodi asukohta ei seata ning noodile vältust ei omistata, kuni noot on paigutatud. Selleks, et järgmine noot ei ületaks viimast noodirida, on olemas signaal *stopAdding*. Mooduli väljundite hulka kuulub ka aeglasem 10Hz sagedusega taktgeneraator nootide vaadeldavaks paigutuseks noodilehel.



Joonis 13. Lisamoodul nootide genereerimiseks.

6. Arendusplaane edaspidiseks

Projekti edasiarenduseks on plaanis teostada veel mitmeid tegevusi. Kõigepealt oleks vaja luua korralik töötav liides andmevahetuseks signaalitöötluse ja visuaalse kuvamise vahel. See tähendab, et kui heli mängitakse sisse, siis FPGA suudab tuvastada täpselt noodi ja selle mängukestvuse, et edastada adekvaatne informatsioon selle paigutuseks noodilehel õige noodi ja tema vältusega. Sellega saaks lugeda analoogheli visuaalse kuvamise esimese etapi edukalt lõppenuks.

Järgmine ettevõtmine oleks täiustada projekti nii, et see rakendaks rohkem muusikateooriat helide kuvamisel. Kindlasti tuleks kasutada ära taktimõõte ning vastavalt nendele takte. See tähendab, et näiteks 4/4 taktimõõdul mängides arvutatakse kokku kõikide nootide kestvus ning jälgitakse, mitu takti nendest tuleb ja kui 4 takti on täpselt täis, siis paigutatakse kõik vastavad noodid ühe takti sisse. Ühtlasi tähendab see ka seda, et tuleb ära määratleda kõikvõimalikud taktimõõdud ning nende eripärad ehk mitu lööki on taktis ning milline vältus vastab ühele löögile.

Määramaks ära kõikvõimalikud helikõrgused, tuleks veel kasutusele võtta alteratsioonimärgid. Muusikateoorias on nendest levinumalt kasutusel diees, bemoll, bekaar, kuid ka dubldiees ja dublbemoll. Hetkel on süsteemis kasutusel ainult täistoonid. Esimene samm oleks võtta alteratsioonimärgid kasutusele nõnda, et oleks võimalik määratleda kõik noodid ilma takte ja taktijooni kasutamata. Peale selle tuleks jälgida igat nooti, mis alteratsioonimärgiga noodile järgneb. Kui eelmine noot oli näiteks pool tooni kõrgendatud või madaldatud (diees või bemoll) ning sellele järgnev noot vastavalt pool tooni madaldatud või kõrgendatud, siis kasutatakse omakorda dieesi või bemolli asemel bekaari ehk tühistatakse eelnev märk. Ilma taktijoonteta tuleks kuvada vajadusel alteratsioonimärk iga kõrgendatud või madaldatud noodi puhul. Juhul, kui on juba korra tühistatud eelmine alteratsioonimärk bekaariga, siis pole vaja seda rohkem kui korra noodi ees kuvada. Taktide puhul kehtib märk muusikateooria kohaselt takti lõpuni.

Nootide paiknemise suhtes tuleks võtta kasutusele abijooned kõikide nootidele puhul, mis ei paigutu täpselt noodireale. See kehtib viiulivõtmes nootide puhul alates neljanda oktaavi C noodist allapoole ja viienda oktaavi G noodist ülespoole ning bassivõtmes teise oktaavi E noodist allapoole ja neljanda oktaavi C noodist ülespoole.

Edasi võiks arendada veel kasutajaliidest ning teha süsteem kasutajasõbralikuks. Ideaalis töötaks kuvar nii, et mistahes heli sissemängimisel see tuvastatakse ning otsitakse välja kõige optimaalsem viis heli kuvamiseks. See tähendab seda, et süsteem üritab ise mõista, mis tempoga kasutaja mängib ning vastavalt sellele pakkuda kõige loogilisem transkriptsioon. Esialgu oleks mõistlikum panna süsteem tööle järgnevalt, et kasutaja ise määrab ära tempo ning taktimõõdu, millega muusikat sisse mängida. Selleks saab kasutada näiteks FPGA peal olevaid nuppe või liuglüliteid. Ühega neist saab seada tempot ning teisega vahetada taktimõõtu. Tempo ning taktimõõt kuvatakse siis vastavalt valitule kuvaril määratud alal. Lisaks tempole ja taktimõõdule võiks kasutajal olla võimalus valida ka helistikku ning noodivõtit, milles transleeritud heli kuvatakse.

Sissemängitava muusika täiustamiseks võiks kasutajal olla võimalus ka transleeritud heli redigeerida. Lisaks FPGA nuppudele, millega saab määrata tempot, vahetada taktimõõtu, valida helistikku või noodivõtit, võiksid kasutusel olla ka teised lülitid noodilehel muudatuste tegemiseks. Kui süsteem ei suuda tuvastada mõnda heli päris korrektselt ning ekraanile tekkiv kujutis ei iseloomusta seda, mida kasutaja mängimisel mõtles, võib ta nuppudega nootide vahel navigeerida ning vahetada vajadusel nende helikõrgust ja vältust. Seda ei pea kasutama ainult juhul, kui heli loetakse vigaselt sisse. Muudatusi võib läbi viia vastavalt kasutaja enda soovile ka olukorras, kus tahetakse noodipilti enda nägemise järgi muuta või parandada. Näiteks vahetada taktijoon kordusmärgi vastu.

Juhul kui signaalitöötlus on tasemel, kus on võimalik tuvastada lisaks helikõrgusele ja noodi kestvusele ka dünaamikat, siis võiks olla võimalus selle kujutamiseks visuaalselt. Kasutusele tuleks võtta helidünaamikat tähistavad (astmete) märgid ja heliatakid. Kuna nende tuvastamine on juba kõrgem signaalitöötluse tase, siis esialgu on võimalik kasutajal ise neid märke kuvaril vastavate kohtade peale paigutada.

Ideaalis suudaks süsteem tuvastada ka intervalle ning hiljem ka akorde. See eeldab jällegi signaalitöötluse paremat tundmist.

FPGA KASUTAMINE DIGITALISEERITUD ANALOOGHELI VISUAALSEKS KUVAMISEKS

Andres Randmaa

Kokkuvõte

Käesoleva bakalaureuse töö eesmärgiks oli luua FPGA peal töötav süsteem, mis kuvaks läbi VGA analoogheli digitaalsel kujul noodilehel. Bakalaureuse töö oli osa suuremast projektist, milles mängitakse heli FPGAga ühenduses oleva mikrofoniga sisse, töödeldakse ja seejärel kuvatakse digitaalselt reaalajas. Esmalt tuli tutvuda FPGA poolt pakutavate võimalustega ning õppida kasutama VGA kontrollerit pildi loomiseks ekraanile. Selleks pidi teadma, kuidas töötab VGA peal horisontaalne ja vertikaalne sünkroniseerimine ning sellele vastavalt saatma õigetel ajahetkedel andmeid pildi kuvamiseks.

Kui VGA kontrolleri tööpõhimõte oli selge, tuli seda kasutades luua ekraanil kuvatav noodileht. Selleks tuli ära määrata ekraanil ala, kus noodileht paikneb ning selle peale joonistada noodijoonestik, noodivõtmed ja taktimõõt.

Disaini järgmises osas tuli hakata paigutama noote noodijoonestikule. Selleks oli määratud omakorda kahemõõtmelised vektorid eri vältustega nootide hoidmiseks ja joonistamiseks vastavalt tuvastatud andmetele. Signaalitöötluse tulemusena oodatakse kahte signaali – noodi väärtust ja selle vältust. Kui viimane on teada, paigutatakse olekumasinat kasutades noot noodijoonestikule. Kui heli parasjagu sisse ei tule, on tegemist pausiga ning seda kuvatakse vastavalt pausi pikkusele kuni järgmise helini või noodijoonestiku lõpuni.

Viimase osana kasutatakse mälu nootide salvestamiseks noodilehel. Selleks tuli määrata ära kuvatava ala suurus, kus noote paigutatakse ning kirjutatakse hetkel kuvatav noot vastava koha peale mällu. Tulemuseks on ühehäälne sissemängitav muusika noodilehel transleeritult.

Edasiseks arenduseks tuleks ühendada signaalitöötluse osa käesolevaga, et testida töödeldud heli kuvamist ekraanil. Kui on olemas töötav lahendus, mis suudab korralikult heli sisse lugeda ja seda ekraanil kuvada, tuleks arendada edasi süsteemi nõnda, et see rakendaks rohkem teadmisi muusikateooriast. Kasutusele tuleks võtta taktid, taktimõõdud, erinevad noodivõtmed, helistikud ja palju muud. Lisaks sellele tuleks katsetada kõikide moodulite tööd ning vigade ilmumisel nendega tegeleda.

DISPLAYING ANALOG SOUND DIGITALLY AS SHEET MUSIC USING AN FPGA

Andres Randmaa

Summary

The aim of the present bachelor thesis was to create a working system using an FPGA that would display analog sound digitally through VGA on a music sheet. The bachelor thesis was part of a bigger project in which music produced by an instrument is recorded with a microphone, processed in an FPGA and displayed thereafter digitally in real-time. At first thorough research was made with design possibilities offered by an FPGA and also a VGA controller for producing an image on screen was learned. To do this, it was important to know how the horizontal and vertical synchronization works on the VGA and produce correct timing signals in order to draw an image.

When the VGA controller was working, a sheet had to be produced on the screen using it. An area was defined for the sheet to be drawn and staffs, clefs and a time signature were added.

The next step in the design was to start placing notes on the sheet. In order to do that, another 2-dimensional array was created to store note values according to the data received from the signal processing unit. Two signals are required to place a note – the note to be displayed and the note value. When the latter is received, the note is placed immediately on the sheet using a state machine. When there is no incoming music, a pause is drawn instead until the next note or the end of the sheet.

The last step is to use memory to store the notes on the sheet. To do this, the area where the notes are displayed was defined and the current displaying note on the sheet would be written in the corresponding memory address. The result would be the transcription of the music.

For further improvement the two sides of the projects should be connected in order to test the signal processing and correct display of the notation. When there is a working solution, that is able to process incoming music and display it on the screen, adjustments should be made to incorporate elements from music theory including and not limited to bars, time signatures, clefs and keys. In addition thorough testing should be conducted on all the modules to ensure proper functioning.

Kasutatud kirjanduse loetelu

[1] D. Lu, „Automatic Music Transcription Using Genetic Algorithms and Electronic Synthesis,“ (2006) http://www.csug.rochester.edu/ug.research/dlupaper.pdf

[2] A. P. Klapuri, „Automatic Music Transcription as We Know it Today,“ *Journal of New Music Research,* kd. 33, nr 3, pp. 269-282, (2004).

[3] M. Davy ja S. J. Godsill, „Bayesian Harmonic Models for Musical Signal Analysis,“ in *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, (Oxford University Press, 2003) pp. 105-124.

[4] A. d. Cheveigné ja H. Kawahara, „Multiple period estimation and pitch perception model,“ *Speech Communication,* kd. 27, nr 3-4, pp. 175-185, April 1999.

[5] M. Goto, „A Real-time Music-scene-description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines in Real-world Audio Signals,“ *Speech Communication,* kd. 43, nr 4, pp. 311-329, September 2004.

[6] G. Velikic, „The Use of Phase in Automatic Music Transcription,“ (2006).

[7] B. Zeidman, *Designing with FPGAs and CPLDs*, (CMP Books, 2002).

[8] M. Rosin, „Kursuse „mikroprotsessorid“ FPGA ja VHDL õppemoodul,“ (Tartu, 2009).

[9] V. A. Pedroni, *Circuit Design with VHDL*, (MIT Press, 2002).

[10] H. Vallaste, „Entsüklopeedia "e-teatmik" vastus otsingule "VGA",“ http://www.vallaste.ee/index.htm?Type=UserId&otsing=482 (17.05.2013).

[11] „D-sub konnektorid,“ http://www.pcmag.com/encyclopedia/term/55439/d-sub-connectors (17.05.2013).

[12] H. Vallaste, „Entsüklopeedia "e-teatmik" vastus otsingule „kiibikorpus“.,“ http://www.vallaste.ee/index.htm?Type=UserId&otsing=3701 (17.05.2013).

[13] „Chip Scale BGA (CP132/CPG132) Package,“ Xilinx, 25 June 2008.

http://www.xilinx.com/support/documentation/package_specs/cp132.pdf (17.05.2013).

[14] „Spartan-3E FPGA Family Data Sheet," Xilinx, 29 October 2012. http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf. (17.05.2013).

[15] „Pmod välismoodulid," http://www.digilentinc.com/Products/Catalog.cfm?NavPath=2,401&Cat=9 (20.05.2013).

[16] „Noodivältused," http://www.viisikannel.ee/et/note-duration (17.05.2013).

[17] „Alteratsioonimärgid," http://www.muusikakool.net/pdf/Alteratsioonimargid.pdf (17.05.2013).

[18] „Muusikateooria algteadmised," http://www.hot.ee/lehola/algteadmised.htm (20.05.2013).

[19] J. Kiho, „Muusika tempo ja dünaamika termineid," (Tartu, 2005).

[20] „Basys 2 töövahend," Digilent, http://www.digilentinc.com/Products/Detail.cfm?Prod=BASYS2 (17.05.2013).

[21] „Half-Size Dip Low Voltage 5.0V Crystal Clock Oscillator," Abracon Corporation,. http://www.mouser.com/ds/2/3/ach-46940.pdf (20.05.2013).

[22] „ISE WebPACK disainitarkvara," Xilinx, http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.htm (17.05.2013).

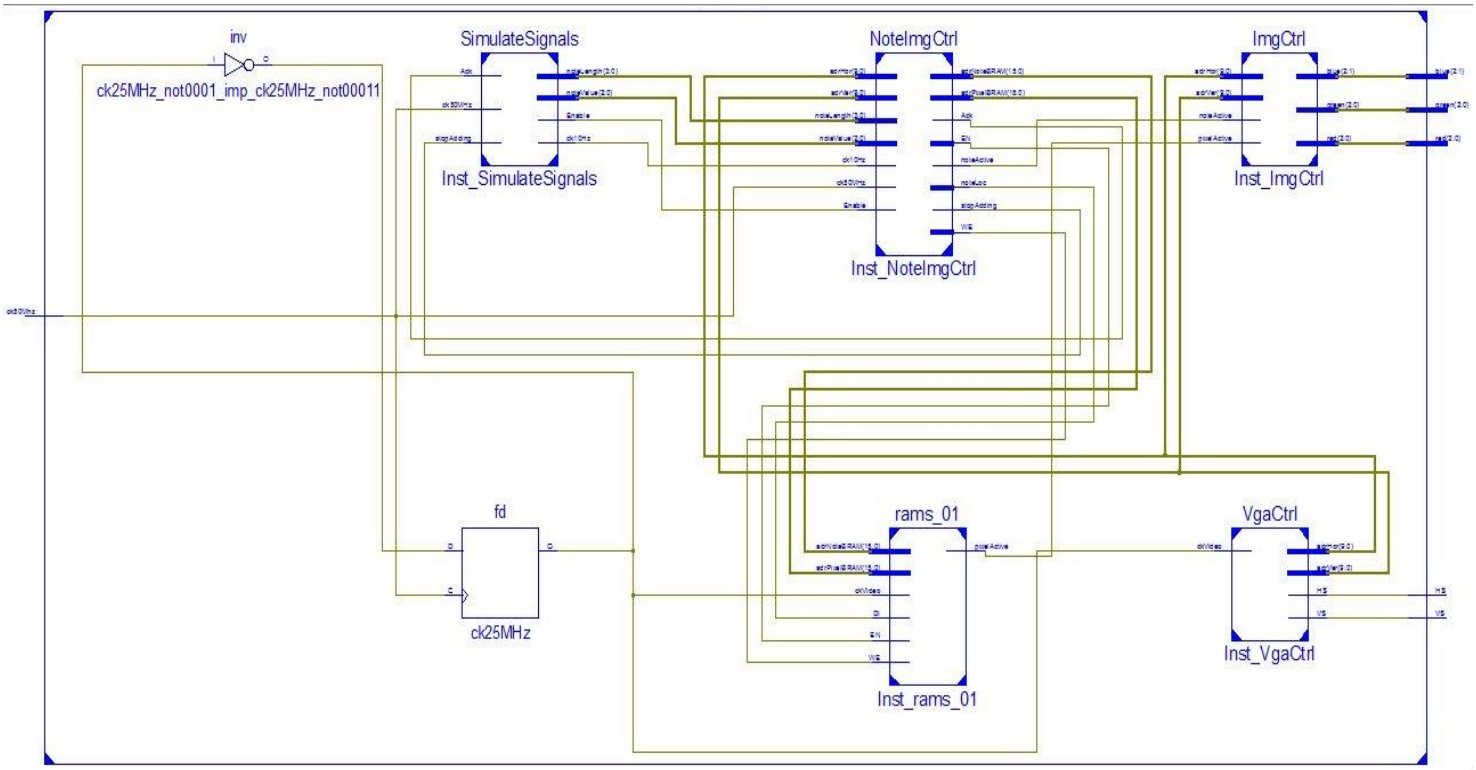
[23] „Digilent Adept," Digilent, http://www.digilentinc.com/Products/Detail.cfm?Prod=ADEPT2 (17.05.2013).

[24] „Digilent Basys2 Board Reference Manual," 11 November 2010. www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf (17.05.2013).

[25] „Elektronkiiretoru," http://opiobjektid.tptlive.ee/Ekraanid/elektronkiiretoru.html (18.05.2013).

Lisad

Lisa 1 - Digitaalse kuvari üldskeem koos lisamooduliga



Lisa 1. Analoogheli digitaalse kuvamise süsteemi üldine skeem koos lisamooduliga RTL tasandil.

Lisa 2 - CD

CD sisu:

1.1 Programmikoodi failid:

- VgaCtrl.vhd (VGA kontroller)
- ImgCtrl.vhd (Pildi kuvamise moodul)
- NoteImgCtrl.vhd (Nootide paigutuse moodul)
- rams_01.vhd (Mälumoodul)
- SimulateSignals.vhd (Lisamoodul nootide genereerimiseks)
- TopLevelVga.vhd (top level fail)
- DisplayDefinition640.vhd (konstantide määramise fail)
- Basys2_100_250General.ucf (FPGA viikude ühendamise fail)
- toplevelvga.bit (käivitatav programmifail FPGA peale laadimiseks)

1.2 Käesolev bakalaureusetöö - *FPGA kasutamine digitaliseeritud analoogheli visuaalseks kuvamiseks.pdf*

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Andres Randmaa
(*autori nimi*)
(sünnikuupäev: 16.10.1990)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

„FPGA kasutamine digitaliseeritud analoogheli visuaalseks kuvamiseks“,
(*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 alates **21.05.2013** kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

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

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

Tartus/Tallinnas/Narvas/Pärnus/Viljandis, 21.05.2013 (*kuupäev*)