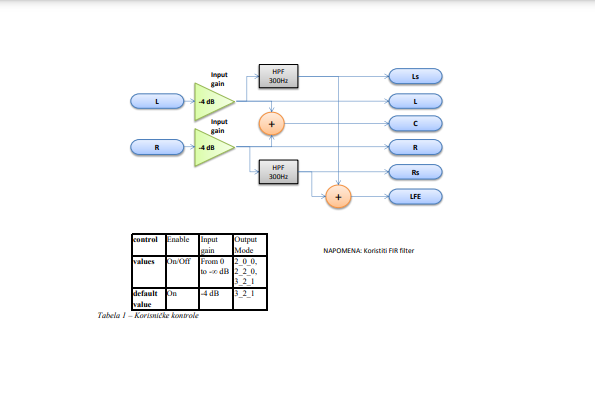
|  |
| --- |
| **Departman za računarstvo i automatiku**  **Smer računarstvo i automatika**  **PROJEKAT**  **Kandidat: Luka Šarenac**  **Broj indeksa: RA36/2020**  **Predmet: AiADSP**  **Tema rada: Obrada signala**  22/11/2023 |

**Uvod**

Zadatak: realizovati DSP semu sa slike

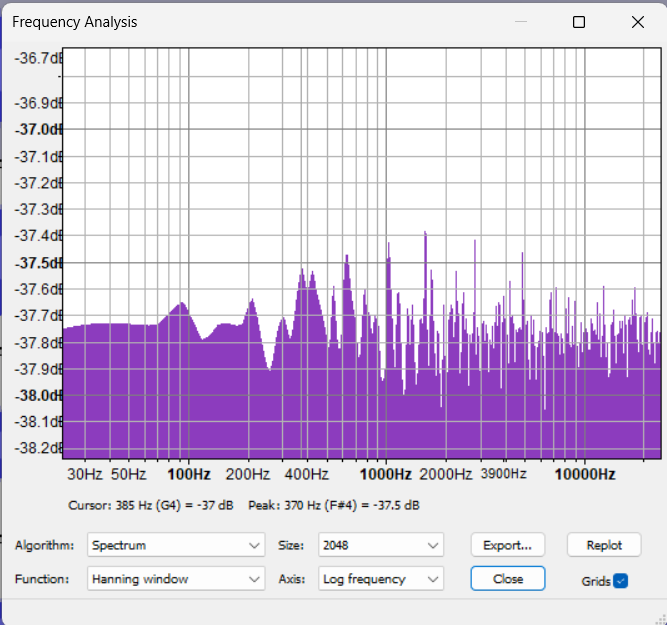
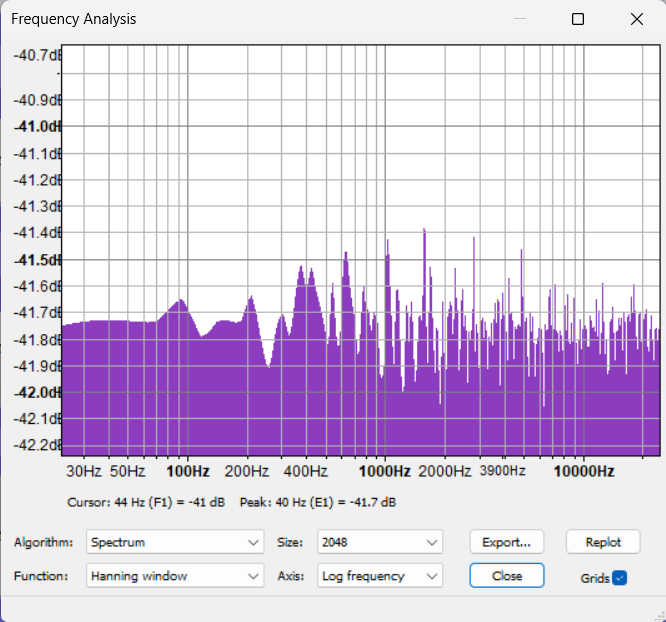


Projektni zadatak realizovan je u 5 modela, modeli 0 do 3 i finalni model, uz to napravljena je i python skripta koja automatizuje testiranje modela 0 do 2, koji su realizovani u Visual Studiju.

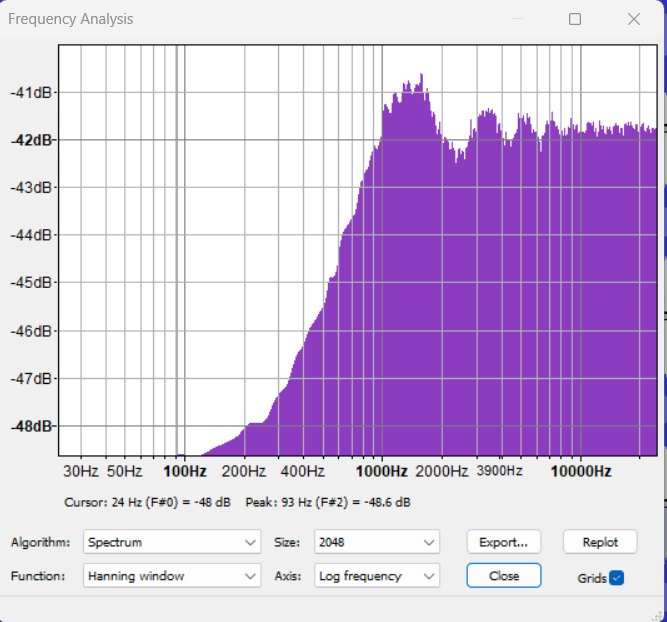
Model 3 kao i finalni realizovani su u CLIDE okruzenju. Za ulazne strimove korisceni su: ''2ch\_lvl\_amt\_48k.wav'', ''Freq\_sweep.wav'', ''funky\_sample.wav'', ''Tone\_L1k\_R3k.wav'' i ''WhiteNose.wav''.

**Model 0**

Cilj modela 0 jeste da ulazne signale provuce kroz potrebnu obradu i da se dobije ocekivani izlal. Za obradu ulaznig signala koristi se funkcija processing u kojoj se u dve odvojene for petlje obavlja obrada nad ulaznim kanalima (levi i desni) i upisuje se u izlazne kanale. Ispravnost ovog modela otvrdjena je pomocu audacity alata, posmatrajuci izlazne kanale.

Ne levi i desni ulazni kanal primenjen je gain od -4db, i kao takvi prosledjeni su na izlazni levi i desni kanal. Na datim slikama mozemo videti kako se ulazni signal sa levog kanala (slika 1.) menja kada ga provucemo kroz gain (slika 2.) i kroz high pass filter 300Hz (slika 3.) 

slika 1. slika 2.



slika 3.

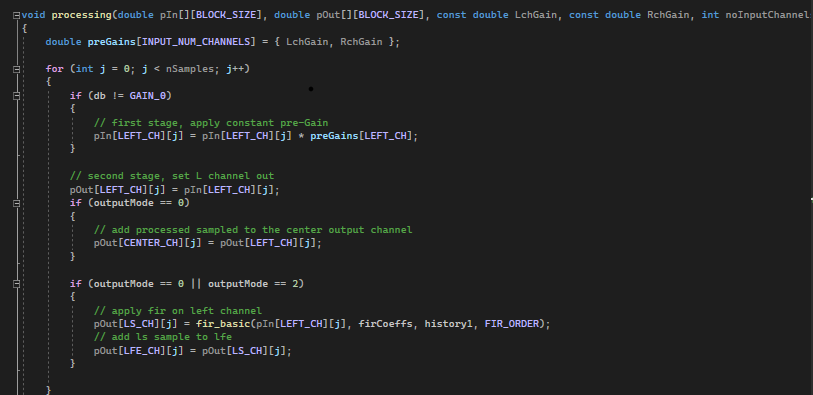
Kao takve, signale prosledjujemo na levi i ls kanal. Ista obrada desava se i sa desnim kanalom i signali se prosledjuju na desni i rs kanal. Zbir levog i desnog ide na centralni, a zbir ls i rs na lfe kanal.

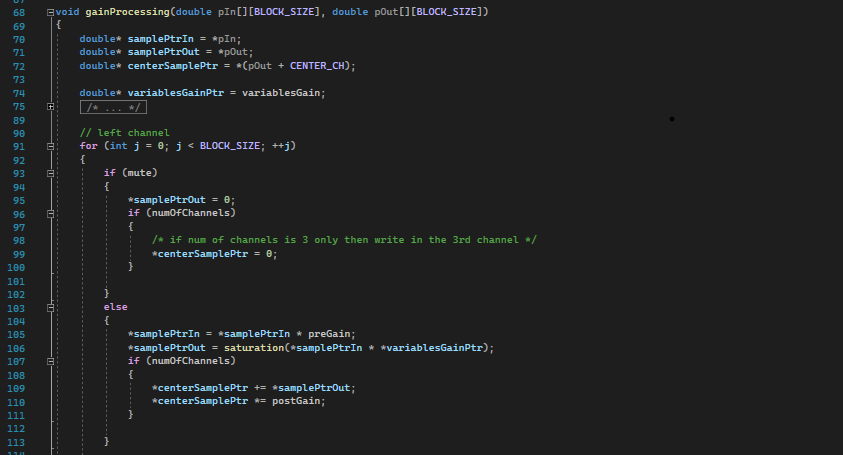
Koeficijenti za fir filter generisani su pomocu WinFilter alata, generisano je 50 koeficijenata.

Funkcija za primenu fir filtra preuzetaje iz datih materijala, a jos se koristi I funkcija initGainProcessing koja inicijalizuje gain vrednost.

Model 1

U ovom modelu uvodimo optimizacije C koda iz modela 0, gledamo da ne definisemo lokalne promenjive unutar nase processing funkcije, da izbacimo sto vise parametara u processing funkciji jer prilikom poziva funkcije u asembleru moramo za te iste parametre odvajati regisre a ovako smanjujemo broj istih i da umesto C-ovskog indeksiranja elementima pristupamo preko pokazivaca da bi se smanjio broj instrukcija.





Model 1 mora generisati signal na izlazu takav da razlika izmedju njega i signala izgenerisanog iz modela 0 ne postoji. Za ovu proveru koriscen je alat PCMCompare koji je dat na vezbama.

**Model 2**

U modelu 2 prilagodjavamo kod prema aritmetici DSP-a. Vodimo racuna da u kodu ne dolazi do neceg sto nasa aritmetika nece moci da obradi.

Uvodimo i fixed point emulacione klase za rad sa emulacionim tipovima a to su fract i accum.

Sve definove iz main.c prebacujemo u common.h, gde, pored toga, definisemo i osnovne tipove. Sve promenjive tipa double treba promeniti da budu fract. Kod istih tih promenjivih sada vrednosti treba dodeljivati sa FRACT\_NUM(x).

Treba obratiti paznju i na vrednosti koje izlaze iz opsega fract tipova (1, -1], u mom slucaju nije bilo takvih situacija, ali ukoliko se to desi potrebno je vrsiti skaliranje i reskaliranje nad njima.

Razlika izmedju modela 2 i modela 1 treba da bude makimalno 2 bita.

**Automatizacija testiranja**

Za automatizaciju testiranja ova tri modela napravljena je python skripta run\_test.py. Ona sadrzi testiranje za 5 .wav strimova koji su gore navedeni.

Pored toga, u svakom modelu imamo vise testnih slucajeva u zavisnosti od argumenata koje prosledjujemo preko komande linije. Programi primaju 5 argumenata:

1. i 2. parametar su ulazni i izlazni strim.

3. parametar predstavlja enable, kada je enable na 1 desice se izvrsavanje gainProcessing funkcije kada je na 0 nece.

4. parametar je output\_mode, u zavisnosti od tog parametra menjaju se izlazni kanali, 0 za 3\_2\_1, 1 za 2\_0\_0 i 2 za 2\_2\_0.

5. parametar odredjuje koliki ce biti gain, 1 za -4db, 0 za 0db i 2 za -16db

Ova skripta prolazi kroz sve moguce slucajeve i generise direktorijum „test\_outputs“ u kom se nalaze svi generisani signali kao i njihova poredjenja sa ostalim modelima pomocu PCMCompare alata.

**Model 3**

Ovaj model se implementuje u CLIDE okruzenju i satoji se od dva dela, realizacije u C-u i u asembleru. Promenjena je C++ logika koja nije podrzana u C jeziku. Uzet je common.h koji je dat na vezbama da bi se mogle izbaciti „stdfix\_emu.c“ i „fixed\_point\_math.c“ koje su koriscene u modelu 2. Ove promene su bile dovoljne da model 3 proradi i da ne postoji razlika u odnosu na model 2.

Nakon ovoga je odradjena priprema za prelazak na asembler, IO baferi su prebaceni u y\_memoriju i implementovana je logika tako da se umesto gainProcessing funkcije iz main.c poziva \_gainProcessing funkcija realizovana u asembleru.



Da bi se pozvala funkcija koja je u asm, potrebno je definisati GAINPROCESSING\_ASM u common.h i gainProc.a.

Pored \_gainProcessing funkcije, u asembleru su implementovane i \_initGainProcessing i \_fir\_basic funkcija.

Finalni model

Ovaj model implementovan je pomocu datih projekata, example\_module, example\_ovly i example\_app. U example\_module nalazi se .a fajl i sada je omoguceno da korisnik enable dugmeta i knobova sa menjanje gejna utice na izvrsavanje programa. Obrada tih dogadjaja odvija se u examplpe\_module\_main.a

