**Arhitekture i algoritmi DSP II**

Projektni zadatak:

Realizovanje algoritma i kombinovanje kanala na Cirrus Logic DSP platformi

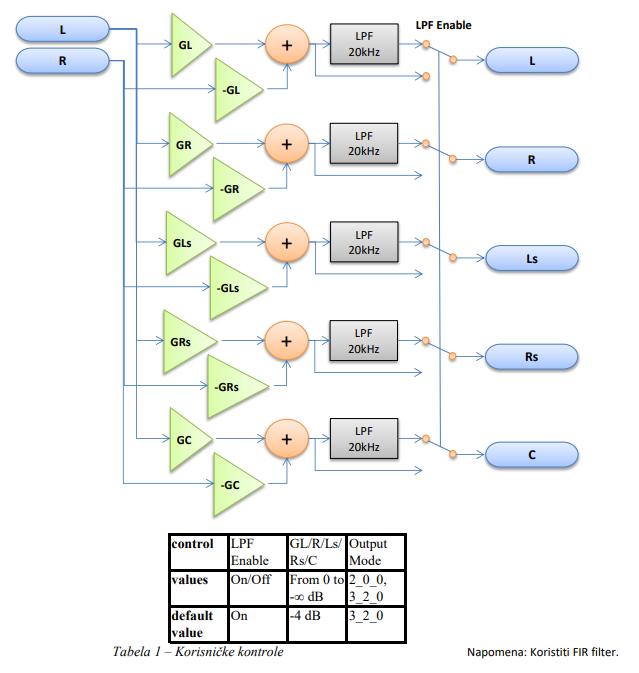
Student: Ognjen Stojisavljevic RA155-2019

Mentor: Nenad Pekez

Novi Sad, 2022

# Opis zadatka

Moj zadatak je bio da se implementira algoritam sa date slike:



Bitni parametri ove audio obrade su:

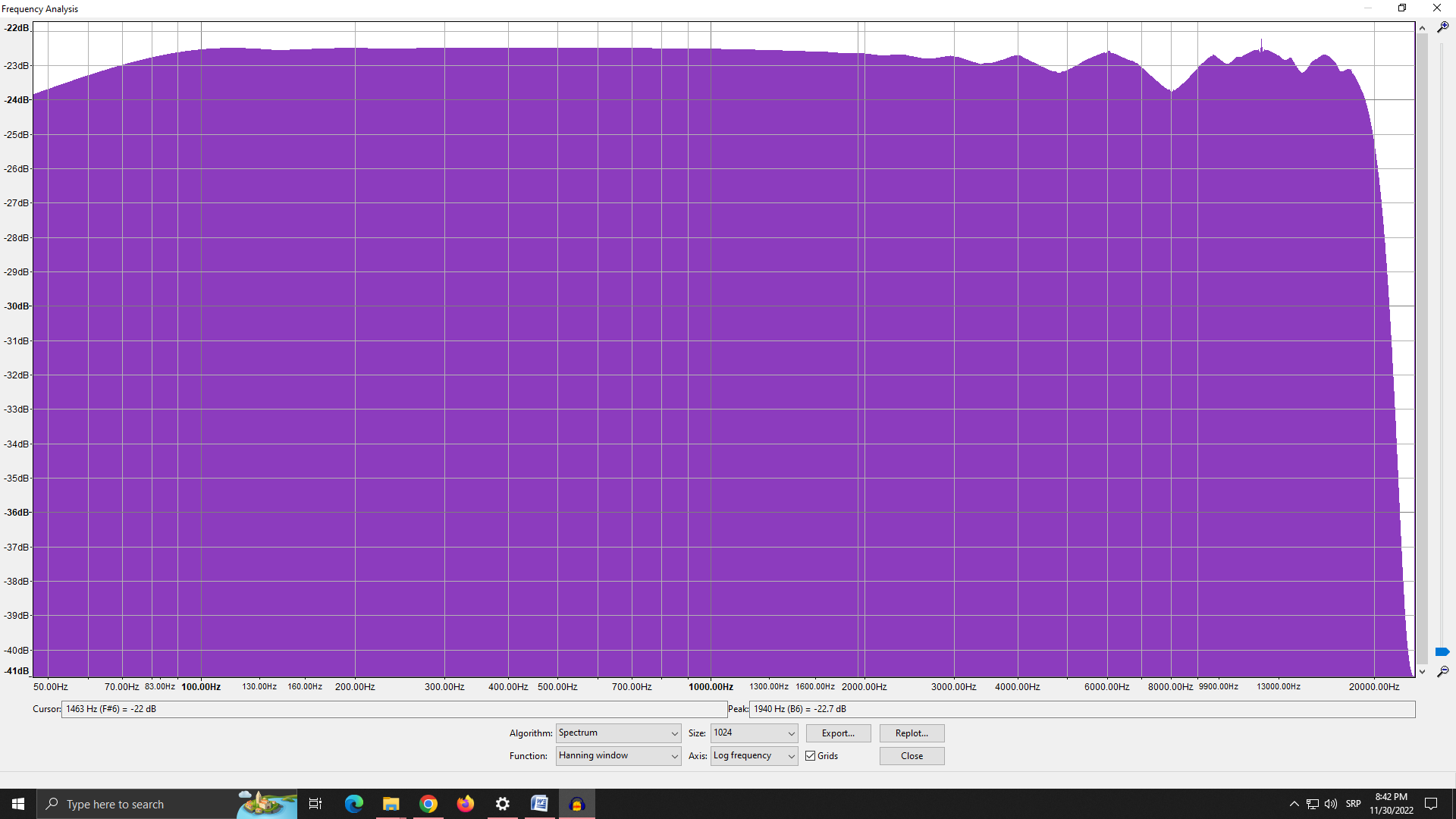
* Gain (-inf, +inf) ograničen na 24dB
* Low Pas Filter(LPF) filter granične frekvencije 20kHz
* Kanalni mod koji je ili 2\_0\_0 ili 3\_2\_0(default vrednost)

# Opis realizacije zadatka

Zadatak je urađen postupno ravojem različitih modela dok nismo došli do modela koji se pokreće na ciljanoj platformi, u našem slučaju Cirrus Logic DSP.

**Model 0:** zadužen za referentni izlaz pomoću kojeg verifikujemo ostale modele.

Pišemo ga u Visual Studio, c++ pogramskom jeziku, nezavisno od platforme na kojoj želimo da izvršavamo naš finalni model.



Analiziramo izlaz iz referentnog modela, na osnovu slušanja, spektralnih analiza kanala i očekivanih rezultata, proveravamo referentni izlaz.

Na osnovu ove slike vidimo da nas filter filtrira signal na 20kHz.

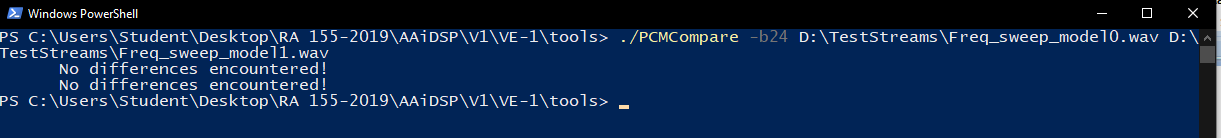
Sve bitne argumente u VS unosili smo preko argumenata main funkcije(LPFenable, gains, mode).

**Model 1:** nastavljamo da razvijamo naš model, a suština je u funkcionalnoj optimizaciji koda.

Umesto direktnog indeksiranja, sve radimo preko pokazivača, izbacujemo dodatne argumente funckije procesiranja da bi dalje rasteretili stek u asembleru.

Izlaz iz modela 1 mora biti u potpunosti isti kao i model 0, što i jeste testiranjem pomocu PCMCompare.

Jedan primera testiranja za model 0 i 1.



Bitska razlika mora biti 0.

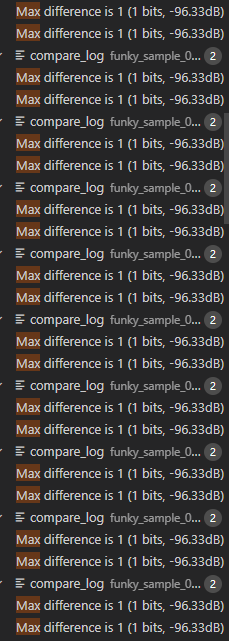
**Model 2:** problem u modelu 2 je prelaženje sa aritmetike pokretnog zareza u aritmetiku nepotrebnog koja priliči našoj ciljanoj platformi.

Pomoću emulacionih biblioteka simuliramo takve tipove, i prilagodjavamo naš kod.

Bitan problem je da moramo da pazimo da naš processing ne napravi izlazak iz opsega koji je definisan u aritmetici nepokretnog zareza.

Naš cilj je da imamo najviše 1 bit razlike u odnosu na model 0 odnosno na model 1.

Za testiranje izlaza koristimo python skriptu, za sve granične i default gainove, kao i za ostale modove.



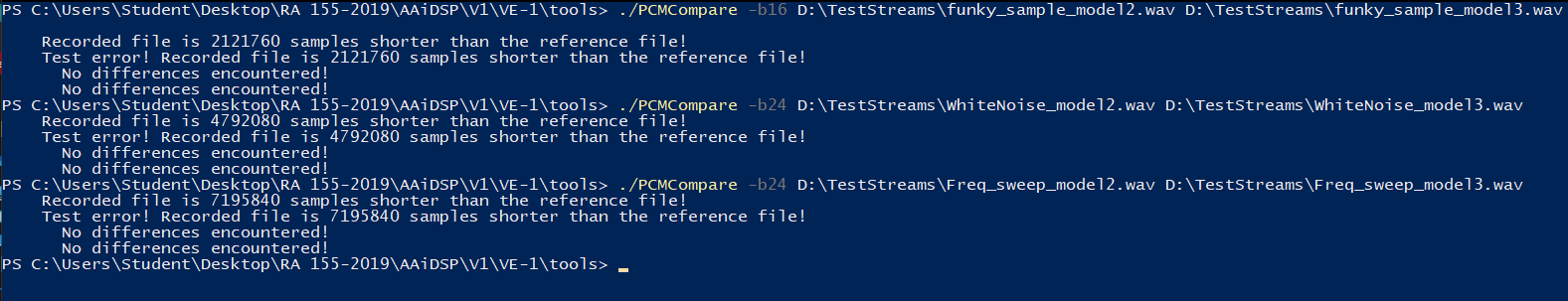
Maksimalna razlika izmedju modela 2 i ostalih modela je maksimalno 1 bit. Tako verifikujemo modele 0,1 i 2.

**Model 3:** Prebacujemo model 2 u naše emulaciono okruženje Clide.

Prva stvar koju radimo je da vršimo sitne izmene, izmedju ostalog promene iz c++ u c, da bi kod proradio u ovom okruženju.

Ne unosimo više vrednosti preko agumenata (osim ulaznog i izlaznog fajla) već hardkodujemo gejnove i modove.

Nakon toga proveravamo izlaze 2 i 3 modela gde nam mora biti razlika 0 bita.



Proveravali smo za razne test strimove, osnovne zadate po zadatku (FreqSweep, WhiteNoise) i neke dodatne.

Naredni korak je razvijanja naše funkcije procesuiranja zvuka u asembleru, da bi u mainu mogli da pozivamo asemblersku funkciju umesto c-ovske.  
  
Algoritam je implementiran u asembleru medjutim zbog, grešaka neću navoditi dalje korake u dokumentaciji.