UNIVERZITET U BANJOJ LUCI
ELEKTROTEHNIČKI FAKULTET

UNIVERZITET U BANJOJ LUCI
ELEKTROTEHNIČKI FAKULTET

DOKUMENTACIJA ISPITNOG RADA

Naziv zadatka: Realizacija algoritma kombinovanja kanala

Student: Vladan Stojnić
Broj indeksa: 1125/13

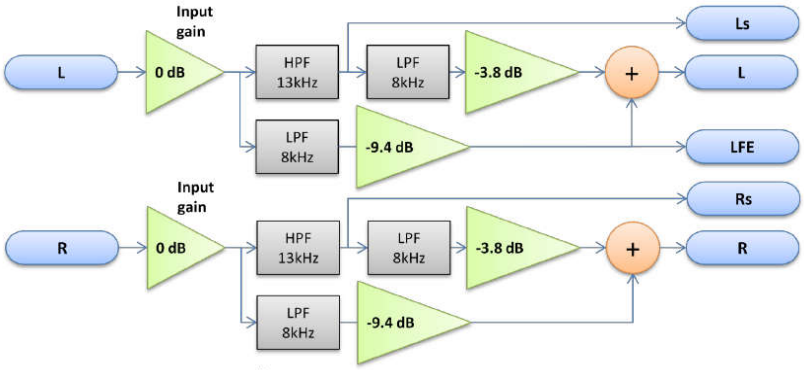
Predmet: Projektovanje digitalnih sistema

Banja Luka, januar 2016.

SADRŽAJ

1. Uvod

Cilj zadatka je da se na osnovu šeme (slika 1.) i tabele kontrola (tabela 1.) realizuje referentni C kod za algoritam kombinovanja kanala.



Slika 1 – Šema algoritma kombinovanje kanala

control	Enable	Input gain	Output Mode
values	On/Off	From 0 to -∞ dB	2_0_0, 2_0_1, 2_2_0, 2_2_1,
default value	On	-0 dB	2_0_1

Tabela 1 – Korisničke kontrole

Obrada se vrši po blokovima, gdje jedan blok predstavlja 16 odabiraka signala. Na ulaz dolaze 2 kanala, a na izlazu se dobija 5 kanala.

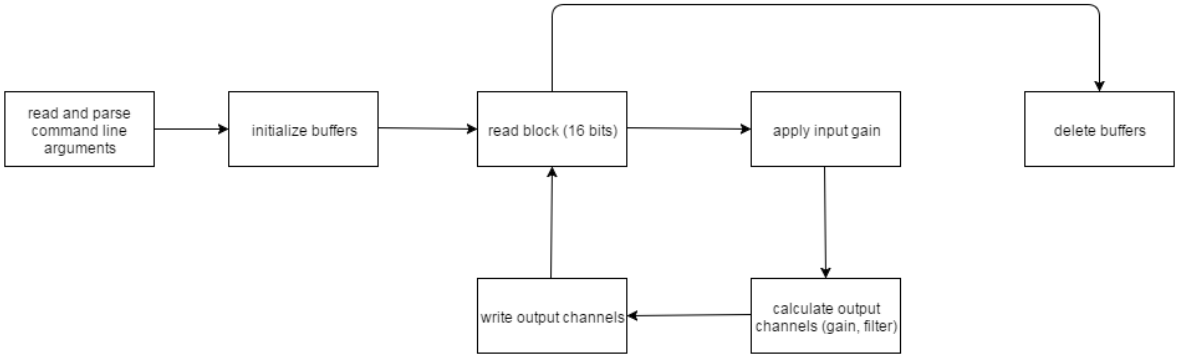
2. Teorijske osnove

Impulsna kodna modulacija (eng. Pulse Code Modulation, PCM) predstavlja postupak digitalizacije signala[2]. Impulsna kodna modulacija je kombinacija tri postupka:

- odabiranja,
- kvantizacije,
- kodovanja.

Kod PCM najčešće velične blokova su blokovi od 8, 16, 20, 24 odabiraka. Također najčešće korištene frekvencije odabiranja su 48kHz (DVD) i 44.1kHz (CD) dok se kod modernije opreme mogu koristiti i frekvencije odabirnja od 96kHz ili 192kHz. Ukoliko je datoteka sadrži više kanala oni su predstavljeni naizmjenično prvo odabirak za prvi kanal, pa za drugi, treći,...[3]

3. Koncept rešenja



Slika 2 – blok dijagaram algoritma

Pri pokretanju programa prvo se vrši parsiranje argumenata komandne linije gdje se provjerava da li su kontrolni ulazi iz dozvoljenog opsega vrijednosti. Nakon toga se vrši kreiranje i inicijalizacija bafera koji se koriste pri realizaciji nisko propusnih i visoko propusnih FIR filtara. Zatim slijedi čitanje odabiraka ulaznog signala i njihovo procesiranje na osnovu šeme date na Slika 1. Nakon završene obrade potrebno je osloboditi memoriju koju su koristili baferi.

Prilikom čitanja odabiraka ulaznog signala potrebno je izvršiti skaliranje ulaznih vrijednosti na opseg [-1, 1].

4. Programsko rešenje

Za realizaciju delay buffera korišteni su kružni baferi. Na ovaj način je povećana brzina rada algoritma jer nije potrebno vršiti pomjeranje u desno svih do sada učitanih odabiraka signala.

int initBuffer(**Buffer*** buffer, **int** n);

Ova funkcija omogućava zauzimanje potrebnog memorijskog prostora za kreiranje kružnog bafera veličine n. Funkcija vrća vrijednost 1 ukoliko je bafer uspješno kreiran, a vrijednost 0 ukoliko bafer nije kreiran.

int insert(**Buffer*** buffer, **float** x);

Funkcija koja vrši dodavanje vrijednosti x na kraj kružnog bafera. Ukoliko je bafer pun prvo se vrši pomjeranje front indeksa (indeks elementa koji je prvi ušao u kružni bafer), pa se tek onda dodaje nova vrijednost.

int isEmpty(**Buffer*** buffer);

int isFull(**Buffer*** buffer);

Ove dvije funkcije vrše provjeru da li je bafer prazan odnosno puni.

void deleteBuffer(**Buffer*** buffer);

Funkcija koja vrši oslobađanje memorijskog prostora kojeg su zauzimali baferi. Ovu funkciju je potrebno pozvati nakon izvršavanja algoritma da bi se izbjeglo curenje memorije (memory leak).

void ReadChannels(**FILE*** input);

Ova funkcija vrši čitanje odabiraka signala iz ulazne datoteke. Pri jednom pozivu funkcije vrši se čitanje onoliko odabiraka signala koliko je određeno BLOCK_SIZE konstantom, za ovaj zadatak veličina BLOCK_SIZE je 16. U ovoj funkciji se vrši i skaliranje ulaznog signala.

void ProcessChannels(**int** outputMode, **float** gain);

Ovdje se vrši obrada ulaznog signala na osnovu šeme kombinovanja kanala (Slika 1.). Prvo se na sve ulazne kanale primjenje definisani input gain, a onda se vrši filtriranje i pojačanje određenih kanala onako kako je specifikovano na šemi.

void LPFilter(**float*** sample, **Buffer*** delay);

Ova funkcija implementira niskopropusni FIR filtar. FIR filtar se implementira na nacin da se ulazni odabirak (sample) i odabirci iz delay buffera množe sa koeficijentima filtra (LPfilterCoefs) i zatim zajedno akumuliraju čime se dobija izlaz iz filtra, a ulazni odabirak se dodaju u delay buffer.

void HPFilter(**float*** sample, **Buffer*** delay);

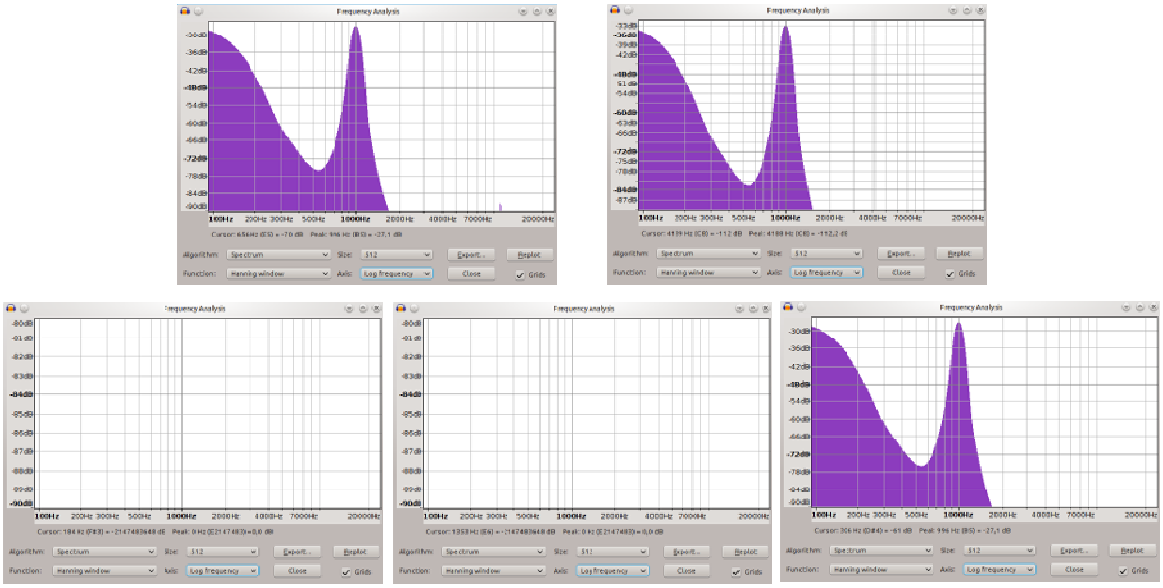
Ova funkcija realizuje visokopropusni FIR filtar. Jedina razlika u odnosu na niskopropusni filtar je ta da se ovdje koriste koeficijeni visokopropusnog FIR filtra (HPfilterCoefs).

void WriteChannels(**FILE*** output);

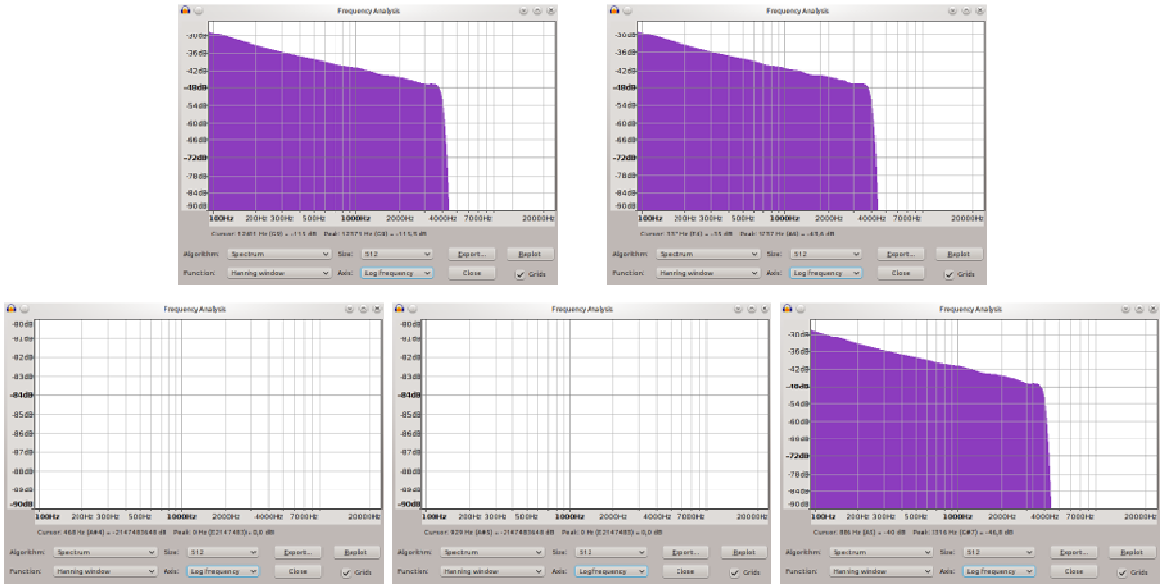
Funkcija koja vrši upisivanje izlaznih kanala u izlaznu datoteku(output).

5. Rezultati

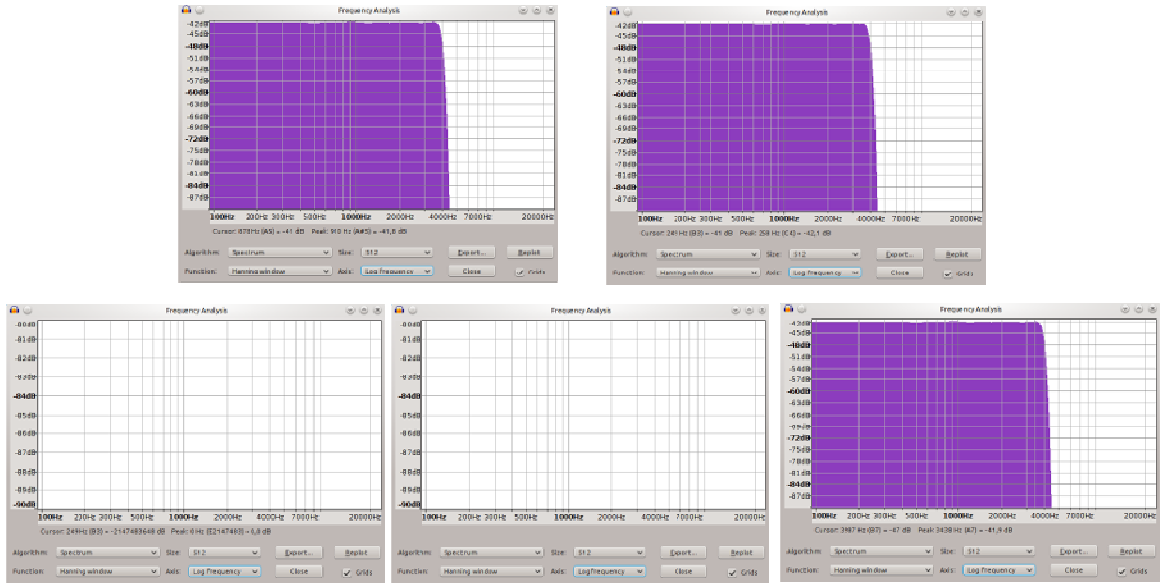
Rad algoritma za datu šemu je testiran nad ulazima 4_tones_48kHz_16bit_2ch.pcm, pink_noise_48kHz_16bit_2ch.pcm i white_noise_48kHz_16bit_2ch.pcm datim na vježbama. Na narednim slikama dati su rezultati tih izvršavanja.



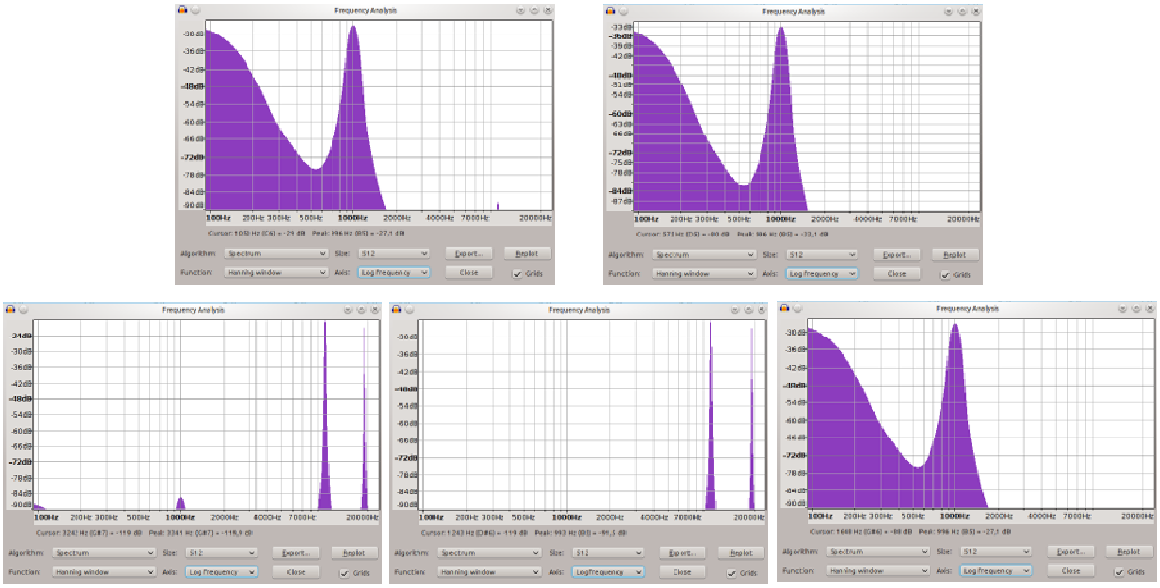
Slika 3 - Izvršavanje algoritma za default kontrole nad ulazom 4_tones



Slika 4 - Izvršavanje algoritma za default kontrole nad ulazom pink_noise



Slika 5 - Izvršavanje algoritma za default kontrole nad ulazom white_noise



Slika 6 - Izvršavanje algoritma sa uključenim svim izlazima nad ulazom 4_tones

Program bi se mogao proširiti na način da se omogući izvršavanje algoritma i nad drugim formatima ulaznih datoteka. Takođe moguća je i optimizacija algoritma za određene DSP platforme.

Prosječno vrijeme izvršavanja algoritma na datim ulazima je 3.18s.

6. Literatura

[1] Vladimir Kovačević, Miroslav Popović, Miodrag Temerinac, Nikola Teslić: *Arhitektura i algoritmi digital signal procesora 1,* FTN Izdavaštvo, Novi Sad, 2005.

[2] Željen Trpovski: *Osnovi telekomunikacija,* Novi Sad, 2004.

[3] *Pulse-Code modluation,* Wikipedia, posjećen 08.01.2016. https://en.wikipedia.org/wiki/Pulse-code_modulation