VEŽBA 4 – Kvantizacija

Potrebno predznanje

- Poznavanje programskog jezika C
- Urađena Vežba 1 – Uvod u Digitalnu Obradu Signala
- Odslušana predavanja iz predmeta APS na temu Diskretizacija signala
- Odslušana predavanja iz predmeta APS na temu Kvantizacija, AD/DA, formati signala

Šta će biti naučeno tokom izrade vežbe

U okviru ove vežbe naučićete:

- Na koji način se kontinualna vrednost amplitude signala prevodi u diskretnu
- Kako funkcioniše proces kvantizacije
- Koje vrste kvantizacije postoje
- Kakav je uticaj kvantizacije na nivo prisutnosti šuma u signalu
- Načini ograničenja dinamičkog opsega signala, i njihov uticaj na signal

Motivacija

Amplituda realnih signala je uglavnom kontinualna funkcija. Za njeno verno predstavljanje u digitalnom računaru potreban bi bio beskonačan broj bita. Kvantizacija (diskretizacija po amplitudi) omogućuje predstavljanje kontinualnih veličina ograničenim (konačnim) brojem bita.

1 TEORIJSKE OSNOVE

Na prethodnim vežbama obrađivana je tema diskretizacije signala. Diskretizacija (digitalizacija) signala je neophodan korak da bi se omogućila njihova obrada na računarima. Proces diskretizacije se sastoji od dva koraka:

- Diskretizacija po vremenu ili **odabiranje**
- Diskretizacija po amplitudi ili **kvantizacija**

Videli smo da odabiranje predstavlja proces koji nam omogućava da se jedan kontinualni signal $x(t)$ predstavi sekvencom digitalnih vrednosti $\{x_n\}$ koje predstavljaju vrednosti signala u ekvidistantnim momentima vremena $x_n = x(nT_s)$, gde je T_s perioda odabiranja.

Na današnjim vežbama bavićemo se procesom diskretizacije signala po amplitudi.

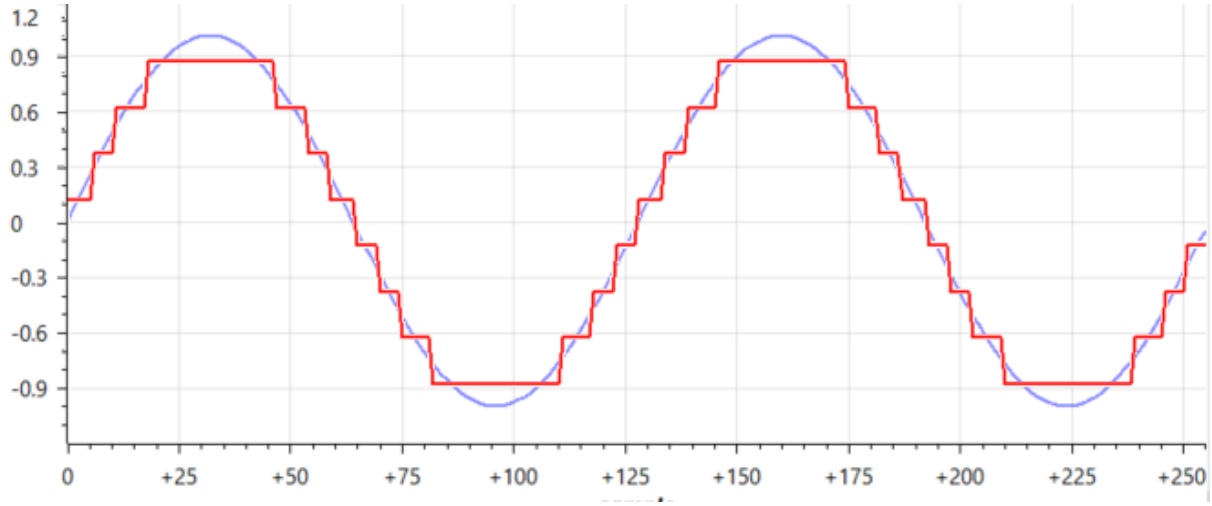
1.1 Diskretizacija po amplitudi - kvantizacija

Nakon izvršenog procesa odabiranja, dobijamo signal predstavljen sa N odbiraka, gde svaki odbirak ima vrednost iz kontinualnog intervala (A_{min}, A_{max}), gde A predstavlja amplitudu signala. Međutim teorijski gledano ukoliko bi želeli da predstavimo dati signal u digitalnom obliku, svaki odbirak bi bio predstavljen sa reči beskonačne dužine, kako bi mogli predstaviti sve vrednosti iz opsega (A_{min}, A_{max}).

Diskretizacija signala po amplitudi ili **kvantizacija**, je proces u kojem vrednost iz kontinualnog intervala (A_{min}, A_{max}) biva predstavljen kodnim rečima konačne i jednake dužine od B bita.

Proces kvantizacije sastoji se iz sledećih koraka:

1. Opseg (A_{min}, A_{max}) se izdeli na Q podopsega, gde je $Q=2^B$ (kvantova).
2. Formiramo konačan skup od Q vrednosti, gde svaka vrednost odgovara jednom podopsegu. Sve vrednosti iz skupa mogu se predstaviti u digitalnom obliku sa B bita.
3. Vrednost svakog odbirka predstavljamo sa vrednošću koja odgovara opsegu u kom se odbirak nalazi



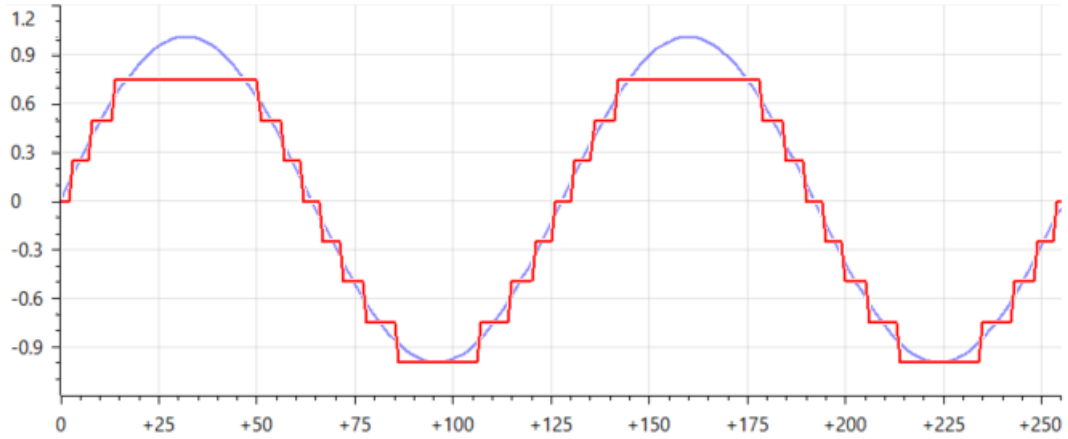
Slika 1 - Analogni signal (plavo) i Kvantizovan signal za B=3 (crveno)

Opšte pravilo kvantizacije dato je sledećom jednačinom:

$$q(s) = q_k \quad za \quad S_{k-1} \le s < S_k \qquad q_k = \frac{S_k + S_{k-1}}{2} \qquad k = 1,2,\ldots,Q$$

U zavisnosti od toga na koji način vršimo prvi korak kvantizacije, odnosno podelu na opsege, kvantizacija može biti linearna i nelinearna. Linearna kvantizacija podrazumeva da su svi podopsezi jednaki ($S_k - S_{k-1} = \Delta$, za bilo koje k), dok kod nelinearne kvantizacije opsezi imaju različite veličine. Primer upotrebe nelinearne kvantizacije jeste kod namenskih sistema kod kojih postoje očekivane vrednosti signala. Za opsege sa većom verovatnoćom vrednosti signala koristi se manja veličina opsega, kako bismo imali veću preciznost, dok se za opsege sa manjom verovatnoćom koriste veći opsezi.

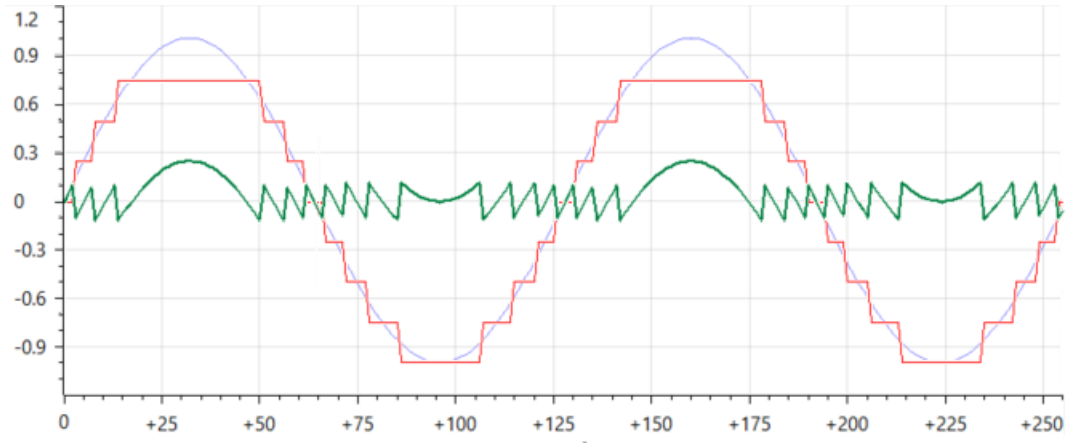
Takođe kvantizacija može biti parno-simetrična ili asimetrična. Parno-simetrična kvantizacija podrazumeva takvu podelu amplitudnog intervala na opsege da imamo jednak broj pozitivnih i negativnih opsega. Ovakav pristup podele ne podrazumeva nultu kvantnu vrednost, već su sve vrednosti signala približno jednake nuli kvantovane na vrednosti -1 ili 1. Asimetrična kvantizacija podrazumeva nultu kvantnu vrednost, ali je asimetrična za maksimalne negativne i pozitivne kvantne vrednosti. Početni interval u tom slučaju je upola manji a krajnji je upola veći od svih ostalih intervala. Prikaz asimetrične linearne kvantizacije dat je na slici 2.



Slika 2 - Analogni signal (plavo) i Kvantizovan signal koristeći asimetričnu linearnu kvantizaciju za B=3 (crveno)

S obzirom da proces kvantizacije predstavlja preslikavanje opsega kontinualnih vrednosti odbiraka (beskonačno mnogo mogućih vrednosti) opseg konačan opseg od Q kvantovanih vrednosti, možemo zaključiti da je kvantizacija ireverzibilni proces. To jest iz kvantovanih vrednosti odbiraka se vise ne mogu egzaktno rekonstruisati kontinualne vrednosti. Zato kažemo da kvantizacija unosi nepopravljivu grešku u signal (razlika između kvantovane i kontinualne vrednosti jednog odbirka) koja se naziva šum kvantizacije.

Na slici 3 prikazan je realan signal plavom bojom. Crvenom bojom je predstavljen je kvantizovan signal. Zelenom bojom predstavljen je šum kvantizacije koji predstavlja razliku prethodna dva signala.



Slika 3 - Šum kvantizacije

Mera za kvalitet kvantizacije je *odnos signal-šum kvantizacije* SNR_q (eng. *Signal to Noise Ratio*).

$$SNR_q = 10 * \log_{10}\left(\frac{P_s}{P_e}\right) dB$$

Gde je P_s snaga (srednja kvadratna vrednost) signala a P_e snaga šuma. Kod linearne kvantizacije, za signale unifomne raspodele, odnos signal-šum direktno je proporcionalan broju bita *B* koji predstavlja dužinu reči kojom se koduje signal i iznosi približno SNR_q=6*B* dB.

1.1.1 Implementacija kvantizacije upotrebom programskog jezika C

Prikazana je funkcija *quantB* koja vrši linearnu asimetričnu kvantizaciju realnog broja iz opsega [-1, 1) u 2^B nivoa (kvantova). Ovako kvantizovane vrednosti mogu se predstaviti sa *B* bita. Funkcija kao parametar prima realan broj u opsegu [-1, 1) i broj bita *B.* Povratna vrednost funkcije jeste celobrojna vrednost u opsegu [-2^{B-1}, 2^{B-1}).

Prvi korak koji je potrebno izvršiti jeste množenje vrednosti ulaznog odbirka sa 2^{B-1}. U okviru programskog jezika C ne postoji operator za računanje stepena broja. Za računanje pozitivnog stepena broja 2 može se iskoristiti operator za aritmetički pomeraj na nivou bita u levo (*shift,* <<). S obzirom da kod binarne predstave brojeva aritmetički pomeraj u levo za jedno mesto odgovara množenju broja sa 2, za računanje N-tog stepena broja 2 potrebno je broj 1 pomeriti u levo za N.

$$2^N = 1 \ll N$$
$$2^5 = 1 \ll 5 = 32$$

Broj 1 (predstavljen sa 8 bita)

0	0	0	0	0	0	0	1

Broj 32 (predstavljen sa 8 bita)

0	0	1	0	0	0	0	0

Nakon množenja vrednosti sa 2^{B-1} potrebno je izvršiti aritmetičko zaokruživanje na najbliži ceo broj. Jedan od načina da se to uradi jeste da se broju doda vrednost 0.5, a zatim pozove funkcija *floor* iz standardne matematičke biblioteke (*math.h*). S obzirom da se vrši asimetrična kvantizacija neophodno je izvršiti dodatnu proveru za poslednji opseg. Nakon ovakvog zaokruživanja broj koji se nalazi u intervalu [$1-2^{-B}$, 1) biće zaokružen na vrednost koja je za 1 veća od maksimalnog broja koji se može predstaviti sa B bita. Iz tog razloga potrebno je izvršiti proveru da li je dobijeni broj jednak broju 2^B, i ukoliko jeste, umanjiti ga za 1. Sledi konverzija broja u celobrojni tip, i dobija se celobrojna vrednost predstavljena sa *B* bita.

```
Int16 quantB(float input, Uint16 B)
{
    Int16 Q = (1L << (B - 1));
    float output_float = floor(input * Q + 0.5);

    if(output_float == Q)
        output_float = Q-1;

    Int16 output_int = output_float;
    return output_int;
}
```

1.2 Ograničavanje signala

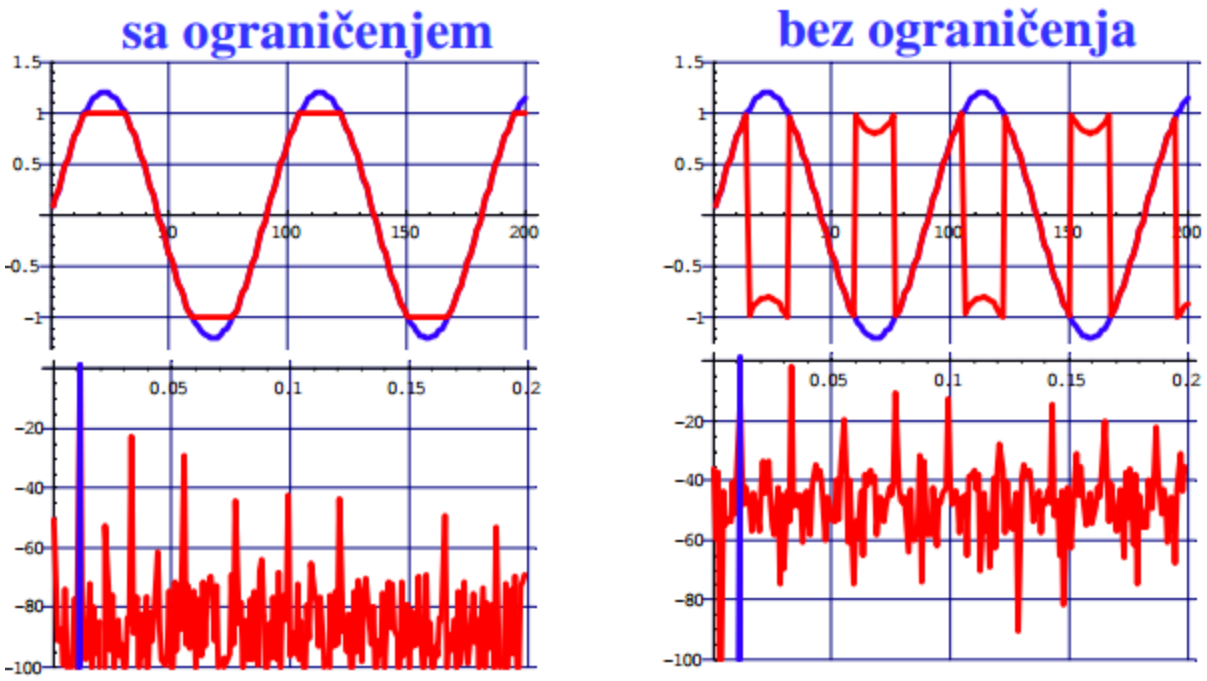
Kao što je navedeno, u procesu kvantizacije, očekivanu vrednost amplitude (A_{min}, A_{max}) delimo u Q podopsega. Međutim u realnim sistemima moguće je da na ulazu u blok za kvantizaciju dobijemo vrednost signala koja nije u očekivanom opsegu ($A > A_{max}$ ili $A < A_{min}$). U slučaju da je u okviru digitalnog sistema dozvoljeno prekoračenje opsega dolazi do preslikavanja vrednosti u opseg tako što se uzima donjih B bita vrednosti i rezultat tretira kao broj u drugom komplementu sa B bita. Ova pojava unosi velika izobličenja u signal i najčešće nije poželjna u realnim sistemima.

Kako bi se pomenuta pojava izbegla najčešće se vrši tzv. ograničavanje signala na zadati opseg. Postoje različiti algoritmi za ograničenje opsega signala, a jedan od najprostijih jeste odsecanje signala ili klipovanje (eng. *clipping*). Klipovanje podrazumeva preslikavanje vrednosti u opseg (A_{min}, A_{max}) ograničavanjem, odnosno ukoliko je vrednost signala veća od maksimalne, postavlja se na maksimalnu vrednost, a ako je manja od minimalne, postavlja se na minimalnu vrednost.

Data je funkcija za klipovanje signala realizovana upotrebom programskog jezika C. Na ulazu je data 16-bitna vrednost koja predstavlja ulazni odbirak, i broj bita na koliko je potrebno klipovati ulaznu vrednost.

```
Int16 clipB(Int16 input, Uint16 B)
{
    Int16 max = (1L << (B-1)) - 1;
    Int16 min = - max-1;
    if(output > max)
        return max;
    else if (output < min)
        return min;
    return output;
}
```

Klipovanje izaziva pojavu harmonika unutar signala. Harmonici su spektralne komponente koje se javljaju na frekvencijama koje predstavljaju celobrojni umnožak frekvencije ulaznog tona. Zbog ove osobine mnogi muzičari namerno izazivaju ovaj efekat (najčešće na električnoj gitari) kako bi postigli reprodukciju “zaprljanih” tonova.



Slika 4 - Primer izgleda klipovanog signala i signala bez ograničenja (iznad - vrednosti signala, ispod - SNRq)

2 ZADACI

2.1 Zadatak 1

1. Uvući projektni zadatak ***Vežba4a*** u radni prostor

Implementirana je funkcija *quantB* za asimetričnu kvantizaciju realnog broja iz opsega [-1, 1) u 2^B nivoa (kvantova).

- `Int16 quantB(float input, Uint16 B);`

Funkcija kao parametar prima realan broj u opsegu [-1, 1) i broj bita *B.* Povratna vrednost funkcije jeste celobrojna vrednost u opsegu [-2^{B-1}, 2^{B-1}).

U okviru zadatka data je funkcija *reconstructB* koja na osnovu kvantizovane vrednosti izračunava realan broj u opsegu [-1, 1) koji ta kvantizovana vrednost predstavlja.

U okviru date *main* funkcije izvršen je poziv kvantizacije nad signalom *p_signal* za vrednost *B* = 4.

Nakon toga izvršena je rekonstrukcija signala na osnovu kvantizovanih vrednosti.

2. Pokrenuti program, i prikazati originalni i rekonstruisan signal u vremenskom i frekventnom domenu.

Napomena: Za prikaz signala koristiti alate *Single Time* i *FFT Magnitude.* Polje *Acquisition Buffer Size* treba da odgovara veličini signala. Polje *Dsp Data Type* postaviti na *32 bit floating point* za signale čiji su odbirci tipa float i *16 bit signed int* za signale čiji su odbirci tipa Int16. Polje *Starting Address* treba da sadrži adresu početka niza. Za *FFT Magnitude* polje *Data Plot Style* postaviti na Linear a *Magnitude*

Display Style na *Logarithmic*. Korišćenje logaritamske skale daje bolju i precizniju predstavu nivoa šuma u signalu. Polje *FFT Order* postaviti na 10.

Ova podešavanja koristiti prilikom iscrtavanja svih signala.

2.2 Zadatak 2

U okviru istog projektnog zadatka:

1. Izračunati vrednost kvantizacionog šuma na sledeći način:
   šum = rekonstruisan _signal – originalni_signal
2. Pokrenuti program i prikazati nivo šuma u vremenskom i frekventnom domenu.
3. Implementirati funkciju za računanje odnosa signal-šum (*SNR*):
   - `float snr(float* signal, float* noise, Uint16 n);`

gde su parametri vrednost originalnog signala, vrednost šuma i dužina signala predstavljena sa brojem odbiraka.

Za računanje koristiti formulu:

$$SNR_q = 10 * \log_{10}\left(\frac{P_s}{P_e}\right) dB$$

Za računanje logaritma koristiti funkciju *log10* iz standardnog *math.h* zaglavlja. Snagu signala računati kao srednju kvadratnu vrednost signala:

$$P = \frac{1}{N}\sum_{n=0}^{N} x_n^2$$

4. Izračunati vrednost odnosa signal-šum za kvantizovani signal i prikazati u konzoli (upotrebom funkcije *printf*).
5. Ponoviti kvantizaciju i računanje vrednosti šuma i *SNR* za vrednosti *B* = 8 i *B* = 16. Prikazati rekonstruisan signal i šum kvantizacije u vremenskom i frekventnom domenu i izračunati *SNR* za oba signala.
6. Komentarisati uticaj broja bita na izgled signala u vremenskom i frekventnom domenu i nivo šuma.

2.3 Zadatak 3

Implementirati funkcije:

- `Int16 clipB(Int16 input, Uint16 B)`
- `Int16 wrapAroundB(Int16 input, Uint16 B)`

koje ograničavaju opseg vrednosti signala na B bita (celobrojne označene vrednosti) i to tako da funkcija *clipB* vrši klipovanje signala na opseg [-2^{B-1}, 2^{B-1}), a funkcija *wrapAroundB* samo uzima donjih *B* bita vrednosti, bez ograničenja i rezultat tretira kao broj u drugom komplementu sa *B* bita.

Primer:

- *wrapAroundB* (1011b,3) = 011b (decimalno 3)
- *wrapAroundB* (0100b,3) = 100b (decimalno -4)
- *wrapAroundB* (0101b,3) = 101b (decimalno -3)
- *clipB* (1011b,3)=100b (decimalno -4)

Gde *b* označava binarni broj. (1011b = decimalno -5 za B=4)

1. U okviru *main* funkcije ograničiti kvantizovan signal predstavljen sa 16 bita na 15 bita upotrebom funkcije *wrapAroundB*.
   - `sin16_clip[i] = reconstructB(wrapAroundB(quant_sine[i], 15), 16);`
2. Prikazati vrednosti rekonstruisanog signala u vremenskom i frekventnom domenu.
3. Ponoviti korake 1 i 2, pritom ograničiti vrednosti signala upotrebom funkcije *clipB* umesto *wrapAroundB.*
4. Komentarisati uticaj ograničenja na izgled signala u vremenskom i frekventnom domenu.

2.4 Zadatak 4

1. Uvucite projektni zadatak ***Vežba4b*** u radni prostor

Dat je primer audio reprodukcije ulaznog audio signala bez ikakve obrade. Čitanje i pisanje signala na kodek vršeno je blokovski, upotrebom *DMA* kontrolera. *AIC3204* kodek je podešen tako da se na ulazu u Analogno-Digitalni konvertor vrši kvantizacija analognog signala na 16 bita. Odbirci su predstavljeni kao označene celobrojne vrednosti veličine 16 bita (*Int16*).

1. Ograničiti ulazne odbirke koristeći funkciju *wrapAroundB* na jednom i funkciju *clipB* na drugom kanalu na 15 bita. Koristiti funkcije *wrapAroundB* i *clipB* iz prethodnog zadatka.
2. Povezati mikrofon kao izvor zvuka na razvojnu ploču. Pokrenuti program, komentarisati subjektivni osećaj prilikom slušanja zvuka na jednom i na drugom kanalu.
3. Ponoviti korake 1 i 2 za vrednosti ograničenja 13 bita.

2.5 Zadatak 5

1. Prilikom inicijalizacije AIC3204 kodeka postaviti vrednost pojačanja (*gain*) na 0 dB.
2. Izmeniti kod tako da na jednom kanalu propušta neizmenjen zvuk, a na drugom zvuk ograničen na 10 bita upotrebom funkcije *clipB*, pomnožen sa 2^6.

```
SampleOut = clipB(SampleIn, 10) << 6
```

Pomeraj za 6 mesta (16 - 10 = 6) se vrši kako bismo maksimalnu amplitudu izlaznog signala izjednačili sa maksimalnom amplitudom neograničenog signala.

3. Povezati *Line out* izlaz vašeg računara na *stereo in* ulaz na razvojnoj ploči.
4. Pokrenuti program. Na računaru reprodukovati datoteku acoustic_guitar.mp3
5. Uporediti zvuk na jednom i drugom kanalu.

3 Zaključak

Nakon uspešno savladane vežbe zaokruženo je znanje o procesu prevođenja signala iz analognog u diskretni domen. Naučili ste kako se vrednost amplitude signala iz kontinualnog opsega preslikava u

diskretnu vrednost. Na primerima je pokazano kako veličina odbirka utiče na količinu šuma nastalog prilikom prevođenja vrednosti iz kontinualnog u opseg diskretnih vrednosti. Upoznali ste se sa merom kvaliteta zvuka, odnosom signala naspram prisutnog šuma (*SNR*), šta ta vrednost predstavlja i kako se izračunava u realnim sistemima.

Uočili ste da ograničavanje opsega signala rezultuje dodavanjem harmonika. Na realnom primeru je prikazano kako ovaj efekat utiče na karakteristike zvuka i kako ga je moguće primeniti u audio tehnici za postizanje željenog zvuka.