VEŽBA 2 – Odabiranje i kvantizacija

Potrebno predznanje

- Poznavanje programskog jezika C
- Urađena vežba – Uvod u Digitalnu Obradu Signala
- Urađena Vežba 1 – Generisanje signala
- Odslušana predavanja iz predmeta OAiS DSP na temu Diskretizacija signala

Šta će biti naučeno tokom izrade vežbe

U okviru ove vežbe naučićete:

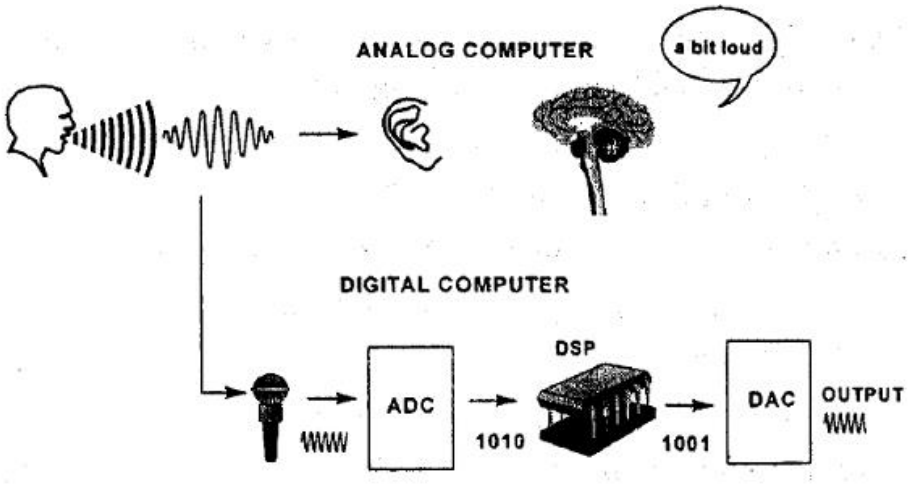
- Osnovne koncepte prevođenja signala iz analognog u diskretni domen i obrnuto.
- Šta je to odabiranje signala i kako se vrši
- Koja su to ograničenja prilikom odabiranja i šta je teorema o odabiranju
- Šta se dešava sa signalom ukoliko se ta ograničenja ne ispoštuju
- Kako frekvencija odabiranja utiče na kvalitet zvuka
- Šta se dešava kada zvuk odabiramo koristeći jednu frekvenciju, a reprodukujemo koristeći drugu
- 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

Signali u prirodi su uglavnom kontinualne funkcije i predstavljeni su beskonačnim brojem vrednosti. Sa druge strane digitalni računari imaju memoriju ograničenog (konačnog) kapaciteta. Odabiranje (diskretizacija po vremenu) omogućava da se jedan kontinualan signal predstavi sekvencom diskretnih vrednosti. Prilikom odabiranja potrebno je očuvati bitne karakteristike signala u cilju što vernije predstave kontinualnog signala koji posmatramo. 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

Digitalna obrada signala može se vršti na dva načina, a to su obrada signala u realnom vremenu, i obrada signala koja nije u realnom vremenu. Obrada signala koja ne podrazumeva realno vreme vrši se nad signalima koji su ranije prikupljeni i sačuvani u digitalnom obliku. Ovakvi signali obično ne predstavljaju trenutno stanje, i zahtevi za rezultatima obrade ne sadrže vremenska ograničenja. Obrada signala u realnom vremenu podrazumeva čvrste zahteve, i prema hardveru i prema softveru, da svoje zadatke izvrše u određenom vremenskom intervalu. Najčešće obrada signala u realnom vremenu podrazumeva učitavanje signala iz prirode, obradu nad učitanim signalom i njegovu reprodukciju.



Slika 1 – Sistem za digitalnu obradu signala u realnom vremenu

Kao što znamo, svi signali u prirodi su analogni. Dakle ukoliko želimo da vršimo digitalnu obradu signala, pre toga taj signal moramo nekako predstaviti u diskretnom obliku. Proces diskretizacije signala sastoji se iz dva koraka:

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

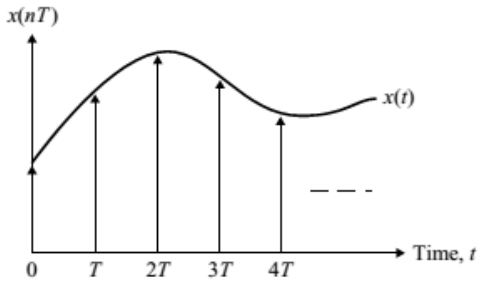
U okviru sistema za digitalnu obradu signala, komponenta koja vrši diskretizaciju signala naziva se analogno-digitalni konvertor (*ADC*). Nasuprot procesa diskretizacije signala postoji i proces za pretvaranje digitalnog signala u analogni, za koji je zadužen digitalno-analogni konvertor (*DAC*).

1.1 Diskretizacija po vremenu - odabiranje

Diskretizacija po vremenu je 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 period odabiranja.

Na slici 1 je prikazan sistem za obradu audio signala. Kao ulazni senzor imamo mikrofon koji predstavlja analognu komponentu koja pretvara zvučni talas u električni signal. Postoje različite vrste mikrofona, međutim, ono što je zajedničko za sve jeste da u svakom trenutku vrednost jačine električnog signala

oslikava trenutnu jačinu zvučnog talasa koji dopire do mikrofona. U datom sistemu odabiranje signala po vremenu se svodi na čitanje vrednosti jačine električnog signala na izlazu iz mikrofona na svakih T_s.



Slika 2 – Primer analognog signala *x(t)* i njegove diskretne prezentacije *x(nT)*

Period odabiranja možemo izraziti i kao: $T_S = \frac{1}{f_S}$ gde f_s predstavlja frekvenciju odabiranja. Frekvencija odabiranja nam govori sa koliko odbiraka će biti predstavljen jedan sekund signala u diskretnom domenu.

1.2 Teorema o odabiranju

Prilikom izbora frekvencije odabiranja postoji kriterijum koji se mora ispoštovati. Ovaj kriterijum opisan je teoremom o odabiranju koja glasi:

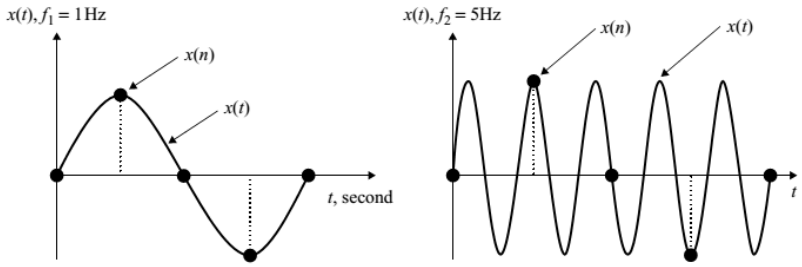
Frekvencija f_s odabiranja signala x(t) treba da je dva puta veća od najveće frekvencije prisutne u signalu da bi se taj signal mogao rekonstruisati bez izobličenja iz signala $x_s(t)$ dobijenog odabiranjem signala x(t).

$$f_S \geq 2f_M$$

Granična frekvencija određena frekvencijom odabiranja $f_N = \frac{f_S}{2}$ naziva se Nikvistova frekvencija.

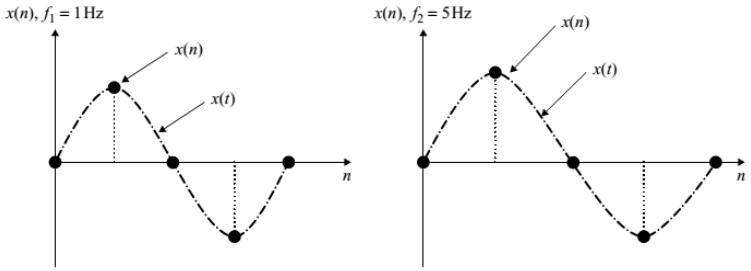
Šta se dešava ukoliko signal koji se odabira sadrži frekvencije više od Nikvistove?

Uzmimo za primer dva sinusna signala frekvencija f_1=1Hz i f_1=5Hz. Po teoremi odabiranja minimalna frekvencija odabiranja za ova dva signala bila bi 10Hz. Umesto toga izvršićemo odabiranje frekvencijom 4Hz.



Slika 3 – Originalni analogni signali

Na slici 3 prikazani su dati signali u vremenskom domenu, sa obeleženim vrednostima odbiraka. Slika 4 sadrži prikaz vrednosti odbiraka diskretnog signala i analogni signal rekonstruisan iz datih odbiraka.



Slika 4 – Odbirci signala i rekonstruisan analogni signal

Kao što možemo zaključiti sa datih slika, originalni signal se može rekonstruisati za sinus frekvencije f_1 = 1Hz. Međutim za signal frekvencije f_2 = 5Hz, dobijeni rekonstruisani signal je takođe sinusni signal frekvencije 1Hz. Za ovakvu pojavu kažemo da je došlo do preklapanja signala frekvencije f_2 u signal frekvencije f_1.

Preklapanje (eng. *Aliasing*) predstavlja izobličenje signala usled neispunjenosti uslova teoreme odabiranja, koje se manifestuje preklapanjem spektralne komponente signala koja je van opsega $f \leq \frac{f_s}{2}$ u neku od spektralnih komponenti u datom opsegu.

U većini realnih sistema za digitalnu obradu signala prilikom A/D konverzije, pre koraka odabiranja signal se propušta kroz pojasno propusni filtar, koji odseca sve spektralne komponente signala van opsega definisanog teorijom odabiranja. Ovakav filtar naziva se „anti-aliasing“ filtar. Upotreba „anti-aliasing“ filtra rešava problem preklapanja neželjenih frekvencija u opseg koji je od značaja, međutim za posledicu ima pojavu da sve spektralne komponente signala više od Nikvistovog kriterijuma budu eliminisane iz signala.

1.3 Podešavanje frekvencije odabiranja kod TMS320C5535

Kao što je prikazano u okviru vežbe 1, TMS320C5535 eZdsp razvojna ploča poseduje TLV320AIC3204 audio kodek koji sadrži *ADC* i *DAC*. Podešavanje željene frekvencije odabiranja vrši se konfiguracijom ove komponente. Konfiguracija kodeka vrši se upisom odgovarajućih vrednosti na predefinisane lokacije u njegovoj memoriji. Memorijska mapa koja sadrži opis značenja i mogućih vrednosti za svaku memorijsku lokaciju data je u pratećem dokumentu koji sadrži tehnički opis TLV320AIC3204 kodeka. U okviru biblioteke za rad sa razvojnom pločom realizovane su funkcije za inicijalizaciju kodeka, inicijalizaciju komunikacije sa kodekom i promenu frekvencije odabiranja i pojačanja ulaznog signala. Funkcije za inicijalizaciju nemaju parametre i date su sa:

- `void aic3204_hardware_init();`
- `void aic3204_init();`

Funkcija za postavljanje, odnosno promenu frekvencije odabiranja i pojačanja ulaznog signala data je sa:

- `unsigned long set_sampling_frequency_and_gain(unsigned long, unsigned int);`

Prvi parameter funkcije predstavlja željenu frekvenciju odabiranja u Hz. Podržane su sledeće vrednosti: 48000, 24000, 16000, 12000, 9600, 8000, 6857.

Pre poziva ove funkcije neophodno je izvršiti inicijalizaciju kodeka. Dat je primer sekvence poziva koji rezultuje uspešno inicijalizovanim i konfigurisanim TLV320AIC3204.

```
aic3204_hardware_init();

aic3204_init();

set_sampling_frequency_and_gain(FS_24kHz, GAIN_IN_dB);
```

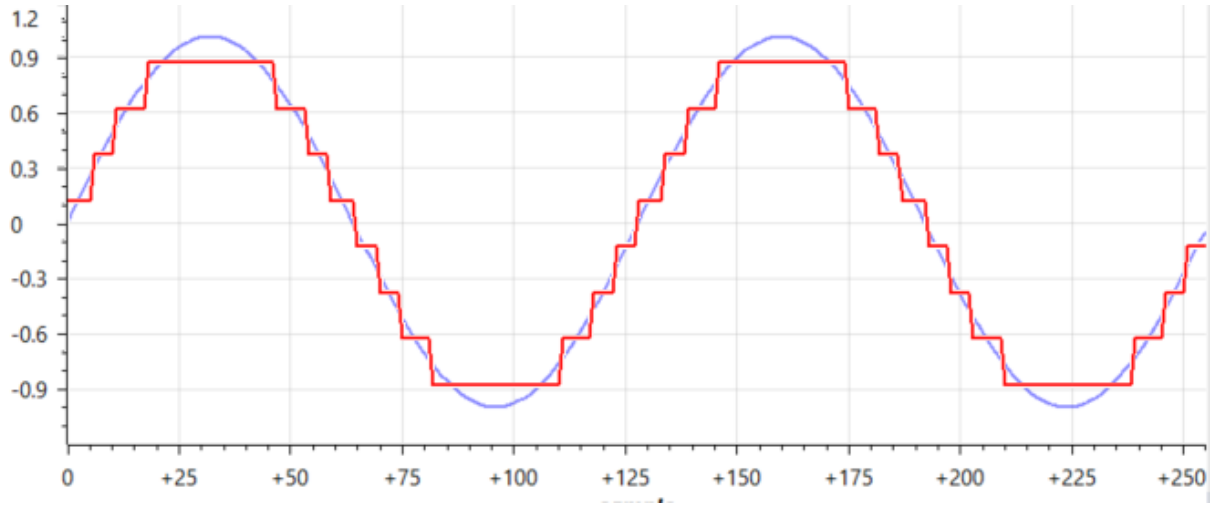
1.4 Diskretizacija po amplitudi - kvantizacija

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



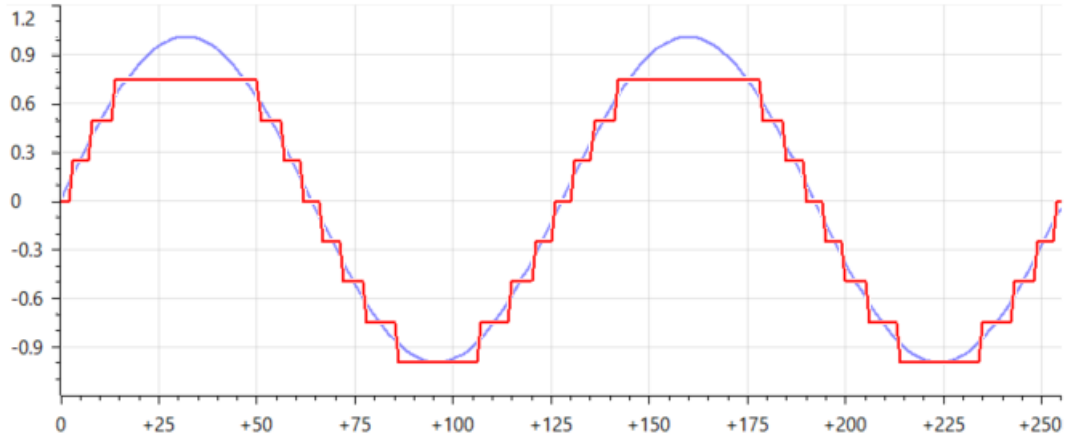
Slika 5 - Analogni signal (plavo) i Kvantizovan signal za B=3 (crveno)

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.

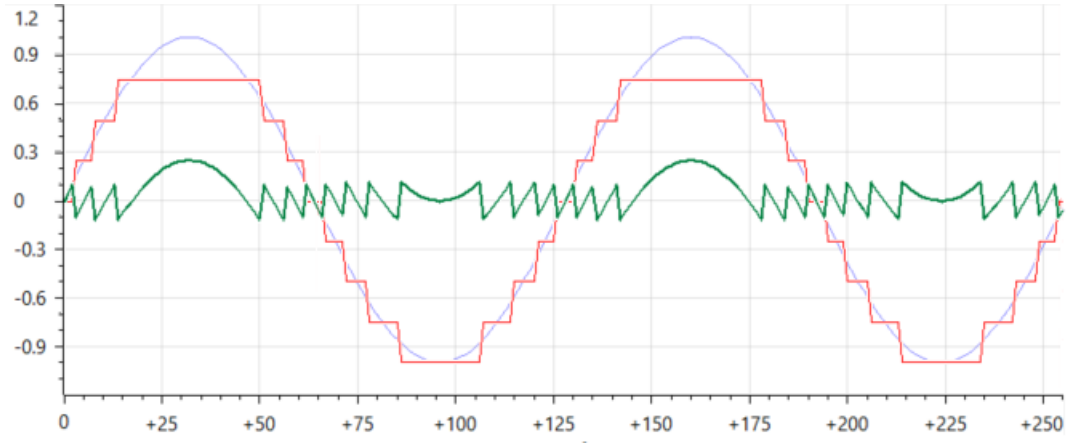
Takođe kvantizacija može biti parno-simetrična ili asimetrična. Parno-simetrična kvantizacija podrazumeva takvu podelu amplitudnog intervala na opsege da imamo jednak broj pozitivnih i negativnih opsega. Ovakav pristup podele ne podrazumeva nultu kvantnu vrednost, već su sve vrednosti signala približno 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 6 - 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 7 - 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=6B dB.

1.4.1 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;
}
```

1.5 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.

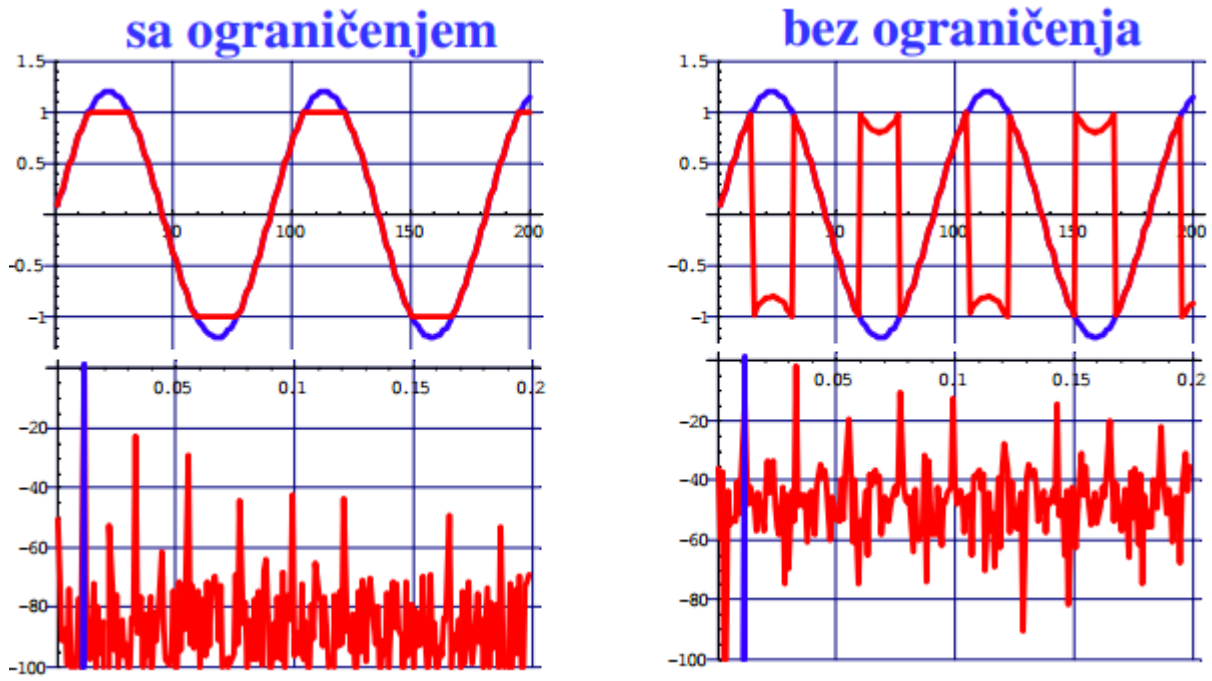
Data je funkcija za klipovanje signala realizovana upotrebom programskog jezika C. Na ulazu je data 16-bitna vrednost koja predstavlja ulazni odbirak, i broj bita na koliko je potrebno klipovati ulaznu vrednost.

```
Int16 clipB(Int16 input, Uint16 B)
{
    Int16 max = (1L << (B-1)) - 1;
    Int16 min =  - (1L << (B-1));
    Int16 output;

    if(input > max)
        output = max;
    else if (output < min)
        output = min;
    else
        output = input;

    return output;
}
```

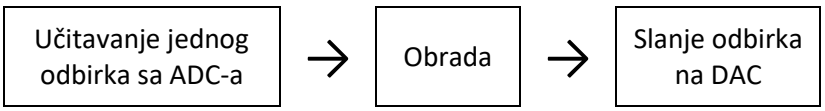
Klipovanje izaziva pojavu harmonika unutar signala. Harmonici su spektralne komponente koje se javljaju na frekvencijama koje predstavljaju celobrojni umnožak frekvencije ulaznog tona. Zbog ove osobine mnogi muzičari namerno izazivaju ovaj efekat (najčešće na električnoj gitari) kako bi postigli reprodukciju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



Slika 8 - Primer izgleda klipovanog signala i signala bez ograničenja (iznad - vrednosti signala, ispod - SNRq)

1.6 Dodatak 1: Blokovska obrada

Tipična obrada signala u realnom vremenu podrazumeva učitavanje vrednosti odbirka na ulazu u sistem, vršenje obrade i slanje vrednosti obrađenog odbirka na izlaz iz sistema. Međutim u ovom slučaju vreme trajanja obrade ograničeno je periodom odabiranja T_s (=1/f_s). Ukoliko bi vreme trajanja obrade bilo veće od perioda odabiranja, dolazi do preskakanja odbiraka i izobličenja signala.



Slika 9 - Tipična obrada u okviru DSP sistemima

To je jedan od razloga zbog čega se u oblasti digitalne obrade signala često koristi pristup blokovske obrade signala. Blokovska obrada podrazumeva učitavanje bloka od N odbiraka, vršenje obrade nad blokom i prosleđivanje obrađenih podataka na izlaz. U tom slučaju dobija se period od NT_s da se obradi N odbiraka.

Većina DSP arhitektura sadrži podršku za direktan pristup memoriji bez posredstva glavnog procesora (eng. *Direct Memory Access* ili *DMA*). DMA funkcioniše tako što DMA kontroleru prosledimo početnu adresu podataka koje želimo da prepišemo, početnu adresu odredišta i količinu podataka. Adresa može biti adresa u memoriji ili adresa perifernog uređaja (u našem slučaju ADC i DAC). Zahvaljujući ovom proširenju moguće je vršiti učitavanje podataka i obradu u paraleli.

Jedna od tehnika koja se može koristiti kako bi se omogućilo vršenje učitavanja odbiraka, obrade podataka i prosleđivanja izračunatih izlaznih odbiraka na DAC u paraleli, jeste tehnika dvostrukog skladištenja podataka ili *Ping Pong* tehnika. Ukoliko se vrši obrada nad blokovima od N odbiraka,

potrebno je za svaki transfer zauzeti memoriju za 2*N odbiraka. Zauzeta memorija sastoji se iz dva dela po N odbiraka, *Ping* i *Pong*. Dvostruko skladištenje podataka funkcioniše tako što DMA kontroler čita vrednosti sa ADC konvertora i piše u Ping bafer. U isto vreme vrši se obrada nad odbircima koji se nalaze Pong baferu. Kada se obrada i DMA transfer završe, dolazi do zamene, DMA kontroler piše u Pong bafer, dok se obrada vrši nad odbircima koji se nalaze u Ping baferu. Isto važi i za izlazne odbirke, dok se odbirci izračunati u toku obrade zapisuju u Ping bafer, DMA kontroler vrednosti smeštene u Pong bafer prosleđuje na DAC, i obrnuto.

Prva iteracija:

Druga iteracija:

Slika 10 - Prikaz blokovske obrae sa dvostrukim skladištenjem podataka (ping-pong)

Konfiguracija DMA kontrolera, inicijalizacija prenosa i čitanje i pisanje vrednosti realizovano je u okviru *ezdsp5535_aic3204_dma.c* datoteke. Podrazumevani broj kanala na ulazu i izlazu iz sistema je 2 (levi i desni), iz tog razloga funkcije za čitanje i pisanje kao parametar primaju po dva memorijska niza. Dat je primer inicijalizacije DMA kontrolera, čitanja i pisanja odbiraka.

```
aic3204_hardware_init();

aic3204_init();

aic3204_dma_init();

set_sampling_frequency_and_gain(FS_24kHz, GAIN_IN_dB);

aic3204_read_block(InputBufferL, InputBufferR);

... Obrada ...

aic3204_write_block(OutputBuffer, OutputBuffer);
```

Za čitanje odbiraka iz ping-pong bafera koristi se funkcija:

- `void aic3204_read_block(Int16* buffer_left, Int16* buffer_right);`

koja dobavlja N odbiraka predstavljenih celobrojno sa 16-bita sa levog i desnog kanala.

Za prosleđivanje odbiraka na DAC konvertor koristi se:

- `void aic3204_write_block(Int16* buffer_left, Int16* buffer_right);`

Veličina bloka (*N*) definisana je sa AUDIO_IO_SIZE. Ova vrednost korišćena je prilikom inicijalizacije DMA kontrolera za čitanje i pisanje.

U okviru funkcija za čitanje i pisanje odbiraka je implementirana opisana Ping Pong tehnika.

2 ZADACI

2.1 Zadatak 1

1. Uvući projektni zadatak *Vezba2a* u radni prostor.
2. Uz pomoć funkcija realizovanih u prethodnoj vežbi generisani su signali:
   - Multiton signal u 2000 odbiraka, amplitude 10, početne frekvencije 1 kHz, sa korakom između harmonika *df*=5 kHz, sa frekvencijom odabiranja 48 kHz
   - Logaritamski „sweep“ signal u 2000 odbiraka, amplitude 10, početne frekvencije f_1=1 kHz, krajnje frekvencije f_2=10 kHz, frekvencije odabiranja 48 kHz

Napomena: U prethodnoj vežbi funkcije za generisanje signala realizovane su tako da kao parametar primaju normalizovanu frekvenciju izračunatu sa $\frac{f}{f_s}$, gde je *f* željena frekvencija, a f_s zadata frekvencija odabiranja.

Generisani Multiton signal predstavlja kako bi izgledao u diskretnom obliku analogni multiton signal zadate frekvencije da je odabran sa frekvencijom odabiranja 48 kHz.

Napravljen je novi signal uzimanjem svakog drugog odbirka generisanog multiton signala.

Novonastali signal predstavlja kako bi izgledao u diskretnom obliku analogni multiton signal zadate frekvencije da je odabran sa frekvencijom odabiranja 24 kHz (odbacivanje svakog drugog odbirka rezultuje duplo većim periodom odabiranja, odnosno duplo nižom frekvencijom).

3. Prikazati generisani *multiton_fs48kHz* i novonastali *multiton_fs24kHz* signal u vremenskom i frekventnom domenu. Komentarisati rezultate.

Napomena: Za prikaz signala u vremenskom domenu koristiti alat *Single Time,* a za prikaz u frekventnom *FFT Magnitude.* I kod jednog i kod drugog količina podataka (*Acquisition Buffer Size*) treba da odgovara veličini vašeg signala (broju odbiraka). Polje *Dsp Data Type* postavite na *32 bit floating point* ukoliko su odbirci vašeg signala tipa *float* ili *16 bit signed integer* ukoliko su tipa *Int16*. Polje *Starting Address* treba da sadrži adresu početka niza (vrednost ili naziv promenljive). Polje *Sampling Frequency* postaviti na vrednost frekvencije odabiranja koju ste koristili za odabiranje vašeg signala.
Za *Single Time* alat polje koje određuje koliko od pročitanih odbiraka želimo da prikažemo (*Display Data Size*) takođe treba da odgovara veličini signala, jer želimo da prikažemo čitav signal.
Za *FFT Magnitude,* polje *Magnitude Data Plot Style* postaviti na *Bar. FFT Order* postaviti na 10. Ova podešavanja koristiti prilikom iscrtavanja svih signala.

4. Na osnovu generisanog logaritamskog „sweep“ signala napraviti nove signale:
    - Uzimanjem svakog drugog odbirka
    - Uzimanjem svakog trećeg odbirka

Novonastali signali predstavljaju kako bi izgledao „sweep“ signal zadate frekvencije odabran frekvencijama odabiranja 24 kHz i 16 kHz redom.

5. Prikazati sva tri „sweep“ signala u vremenskom i frekventnom domenu. Komentarisati rezultate.

Očekivani izlaz iz zadatka:

- „*screenshot*“ grafika svakog signala za koji je rečeno da ga je potrebno prikazati.
- Tekstualna datoteka koja sadrži komentare.

2.2 Zadatak 2

1. Uvući projektni *Vezba2b* u radno okruženje.

Cilj ovog zadatka jeste demonstriranje subjektivnog osećaja prilikom slušanja zvuka koji je odabran jednom frekvencijom odabiranja, a zatim reprodukovan koristeći drugu. Ovaj efekat se koristi prilikom davanja glasova likovima u crtanim filmovima, te nosi naziv „Miki Maus efekat“.

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*). Ulazni signal je odabran frekvencijom odabiranja 16kHz.

Početni kod ne radi nikakvu obradu, samo prepisuje ulazne odbirke u izlazni bafer. Petlja obrade data je sa:

```
for(i = 0; i < number_of_blocks; i++)
{
  aic3204_read_block(inputBufferL, inputBufferR);

  for(j = 0; j < AUDIO_IO_SIZE; j+=2)
  {
        outputBufferL[j] = inputBufferL[j];
        outputBufferR[j] = inputBufferR[j];
  }

  aic3204_write_block(outputBufferL, outputBufferR);
}
```

Da bi se postigao željeni efekat, potrebno je:

1. U funkciji *main*, neposredno pre zapisa zaglavlja izlazne datoteke (*aic3204_write_wav_header*), postaviti veličinu izlazne datoteke na duplo manju:

```
outputWAVhdr.data.SubChunk2Size /= 2;
```

2. U okviru petlje obrade, u izlazni bafer pisati svaki drugi odbirak iz ulaznog bafera. Ovo ponoviti i za levi i za desni kanal. Voditi računa da funkcija za zapis uvek zapisuje čitav bafer, te je neophodno pročitati dva ulazna bloka podataka, pre zapisa jednog izlaznog.
3. Pokrenuti rešenje, otvoriti dobijenu datoteku i poslušati rezultat obrade.
4. Ponoviti korake 1-3, pri tom povećati frekvenciju 3 puta umesto 2.
5. Ponoviti korake 1-3 pri tom smanjiti frekvenciju 2 puta (zapisivati svaki ulazni odbirak dva puta).

Šta se dešava sa zvukom kada ga odabiramo jednom frekvencijom, a reprodukujemo drugom? Šta se dešava kada je frekvencija kojom reprodukujemo signal manja, a šta kada je veća od frekvencije odabiranja na ulazu?

Očekivani izlaz iz zadatka:

- Tekstualna datoteka koja sadrži komentare i odgovore na pitanja.

2.3 Zadatak 3

1. Uvući projektni zadatak *Vezba2c* 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.

2.4 Zadatak 4

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.

2.5 Zadatak 5

Implementirane su 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*.
   - `sin_wrap[i] = reconstructB(wrapAroundB(quant_sine[i], 15), 15);`
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.

2.6 Zadatak 6

1. Uvucite projektni zadatak *Vezba2d* u radni prostor
2. 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. Nakon ograničenja, pomnožiti obirak sa 2, kako bi pojačanje ostalo nepromenjeno u odnosu na ulaznu datoteku.
3. Pokrenuti program, poslušati izlaznu datoteku i komentarisati subjektivni osećaj prilikom slušanja zvuka na jednom i na drugom kanalu.
4. Ponoviti korake 1 i 2 za vrednosti ograničenja 13 bita (potrebno pomnožiti odbirke sa 8 nakon ograničenja).

2.7 Zadatak 7

1. 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
```

2. Odabrati kao ulaznu datoteku acustic_guitar.wav.
3. Pokrenuti program
4. Uporediti zvuk u izlaznom fajlu, na jednom i drugom kanalu.

3 Zaključak

Proces prevođenja signala iz kontinualnog u diskretan domen sastoji iz dva koraka: diskretizacija u vremenu (odabiranje) i diskretizacija po amplitudi (kvantovanje). U okviru ove vežbe obrađen je prvi korak odnosno operacija odabiranja signala. Upoznali ste se sa pojmom frekvencije odabiranja i teoremom koja definiše ograničenja prilikom njenog odabira. Videli ste na primeru kako frekvencija odabiranja utiče na signal.

Nakon uspešno savladanog drugog dela 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.