VEŽBA 6 – Diskretni sistemi za obradu signala

Potrebno predznanje

- Poznavanje programskog jezika C
- Urađena Vežba 1 – Uvod u Digitalnu Obradu Signala
- Odslušana predavanja na temu Diskretni sistemi
- Koncept kružnog buffer-a
- Koncept blokovske obrade

Šta će biti naučeno tokom izrade vežbe

U okviru ove vežbe naučićete:

- Šta su to diskretni sistemi za obradu signala
- Osobine diskretnih sistema
- Kako funkcioniše sistem za digitalno kašnjenje
- Šta je to eho efekat, a šta reverberacija.
- Jedan od načina za proizvodnju eho efekta i efekta reverberacije u realnim sistemima
- Šta je diskretna kovolucija, kako se implementira.
- Kako se uz pomoć diskretne konvolucije, zvučni signal može modifikovati tako da zvuči kao da je reprodukovan u nekom akustički specifičnom prostoru (operska sala)
- Šta je to parametrizovana realizacija diskretnih sistema i koje su prednosti ovakvog pristupa.

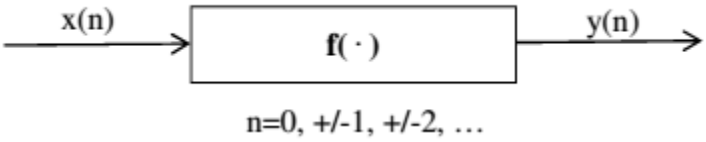
Motivacija

U kontinualnim sistemima za obradu signala (npr. oko) u istom trenutku obavlja se vise različitih operacija nad vrednostima signala. U diskretnim sistemima za obradu signala moguće je vršiti ograničen broj operacija nad ograničenim brojem vrednosti signala. U opštem slučaju vrši se jedna operacija nad jednom vrednošću signala. Zbog toga je potrebno uvesti organizaciju odbiraka signala u vremenu kao i organizaciju sekvencijalnog izvodjenja operacija nad tim odbircima. Digitalno kašnjenje omogućava održanje vremenske zavisnosti izmedju odbiraka signala u vremenu i redosleda operacija nad tim signalima. Iako je odredjen broj odbiraka u istom trenutku smešten u memoriji diskretnog sistema digitalno kašnjenje omogućuje da se operacije nad njima obavljaju prema vremenima njihovog ulaska u diskretan sistem.

1 TEORIJSKE OSNOVE

1.1 Diskretni sistemi

Ova vežba ima za cilj da studenta upozna sa osnovnim načinima softverske realizacije linearnih vremenski nepromenljivih diskretnih sistema. Diskretni sistem je funkcija koja vrši preslikavanje skupa diskretnih signala u samog sebe. Signal koji se preslikava naziva se ulazni signal ili pobuda, a signal u koji se preslikava izlazni signal ili odziv. Osnovni model diskretnog sistema prikazan je na slici 1.



Slika 1 – Osnovni model diskretnog sistema

Ulazni signal je sekvenca odbiraka x(n) kojem odgovara izlazni signal koji je takođe sekvenca odbiraka y(n). Diskretni sistem je opisan funkcionalnom zavisnošću f izlaznih i ulaznih odbiraka:

$$y(n) = f\{x(n)\}$$

Da bi se neki diskretan sistema mogao realizovati, taj sistem mora ispunjavati dva uslova:

- *Uslov stabilnosti* glasi da su izlazni odbirci ograničenih vrednosti ako su i ulazni odbirci ograničeni.
- *Uslov kauzalnosti* utvrđuje da posledica ne može prethoditi uzroku, to jest da se na izlazu ne mogu pojaviti promene pre nego što se jave na ulazu. Opštije rečeno, izlaz sistema zavisi samo od trenutnih i prošlih vrednosti ulaza.

Diskretne sisteme možemo podeliti po osobinama koje poseduju na:

- Linearnost:
  - *Linearni sistem* ispunjava uslov da je izlaz sistema za linearnu kombinaciju na ulazu istovetna linearna kombinacija na izlazu
  - *Nelinearni sistem* ne ispunjava prethodni uslov, to jest funkcionalna zavisnost f sadrži nelinearne zavisnosti izlaznih odbiraka od ulaznih odbiraka
- Vremenska zavisnost
  - *Vremenski invarijantan sistem* ima nepromenjivu funkcionalnu zavisnost f, to jest ona nije zavisna od momenta (n).
  - *Vremenski zavisan sistem* ima vremenski zavisnu funkciju, odnosno odziv sistema na pobudu x(n) zavisi i od momenta n u kom se ta pobuda javila.
- Memorija
  - *Sistem bez memorije* ima zavisnost izlaznog odbirka u jednom momentu y(n) samo od ulaznog odbirka u istom momentu vremena x(n).

- *Sistem sa memorijom* ima zavisnost izlaznog odbirka u jednom momentu vremena od ulaznih odbiraka u trenutnom i svim prethodnim momentima. U ovu grupu spadaju i sistemi sa povratnom spregom u kojima je izlazni odbirak u jednom momentu zavisan i od izlaznih odbiraka u prethodnim momentima.

- Kauzalnost
  - *Deterministički sistem* ima eksplicitno definisanu funkcionalnu zavisnost *f*.
  - *Stohastički sistem* ima definisane verovatnoće vrednosti izlaznih odbiraka y_n za zadate vrednosti ulaznih odbiraka x_n

1.2 Diskretna konvolucija

Kod *linearnih vremenski invarijantnih sistema* (LVIS) prenosna karakteristika sistema opisuje se impulsnim odzivom datog sistema. Impulsni odziv diskretnog LVIS *h(n)* definisan je kao odziv na δ-impulsnu pobudu. Diskretan δ impuls ili Dirakov impuls definisan je sa:

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

Vrednost izlaznog signala *y(n)* kod LVIS sistema računa se operacijom **diskretne konvolucije** ulaznog signala *x(n)* i impulsnog odziva sistema *h(n)*.

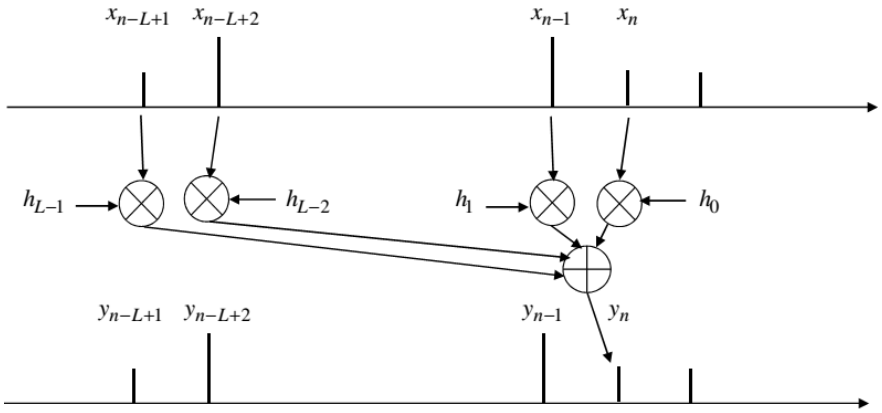
Operacija diskretne konvolucije predstavljena je sledećom jednačinom:

$$y(n) = x(n) * h(n) = \sum_k x(k) \cdot h(n-k)$$

Za operaciju diskretne konvolucije važi komutativnost:

$$y(n) = x(n) * h(n) = h(n) * x(n) \;\leftrightarrow\; y(n) = \sum_k x(k) \cdot h(n-k) = \sum_k h(k) \cdot x(n-k)$$

Ako se pretpostavi da je impulsni odziv dužine L, diskretna konvolucija može se vizuelno prikazati kao:



Slika 2 - Diskretna kovolucija za impulsni odziv dužine L

Vrednost izlaznog odbirka u momentu *n* linearno zavisi od L ulaznih odbiraka u momentima *(n), (n-1), ..., (n-L+1)*, pri čemu su težinski faktori vrednosti impulsnog odziva u momentima *(0), (1), ..., (L-1)*.

Koristeći programski jezik C, operacija konvolucije se može implementirati na različite načine. Jedan od prostijih načina implementacije dat je sa:

```
for ( n = 0; n < xLength; n++ )
{
    y[n] = 0;
    for ( k = 0; k < hLength; k++ )
    {
        y[n] += x[n - k] * h[k];
    }
}
```

Gde su ulazni signali sa *x* i *h*, a njihove dužine sa *xLength* i *hLength.* Izlazni niz definisan je sa *y.* Obratite pažnju da za računanje vrrednosti y(n), potrebno je da vrednosti [x(n-hLength), x(n)] budu poznate. Problem se javlja kod prvih hLength članova ulaznog niza x. Jedno od rešenja jeste da se podrazumeva da su vrednosti signala x sa negativnim indeksom (-1, -2, -3…) jednake 0. U tom slučaju izmenjeni kod konvolucije izgledao bi:

```
for ( n = 0; n < xLength; n++ )
{
    y[n] = 0;
    for ( k = 0; k < hLength; k++ )
    {
        if ( n < k)
              break;
        y[n] += x[n - k] * h[k];
    }
}
```

U slučaju pojave negativnog indeksa (n-k), u svakoj sledećoj iteraciji indeks će i dalje biti negativan (zbog uvećanja k). S obzirom da je podrazumevana vrednost odbiraka signala x za negativan indeks 0, a rezultat množenja bilo koje vrednosti nulom takođe jednak 0, *for* petlja se prekida. Ključna reč *break* označava prekid izvršenja petlje (*for* ili *while*). U slučaju ugnježdene petlje odnosi se samo na trenutnu petlju, ne i na spoljašnje.

1.2.1 MAC instrukcija

S obzirom da se u procesu digitalne obrade signala veoma često susreće sračunavanje konvolucije, arhitektura DSP procesora prilagođena je njenom sračunavanju. Ono što razlikuje DSP procesore u odnosu na procesore opšte namene, ili druge namenske procesore, a omogućava brzu realizaciju matematičke operacije konvolucije, jeste mogućnost izračunavanja sledeće operacije u jednom taktu:

$$Y = Y + X \cdot A$$

Ova operacija naziva se *MAC* instrukcija (eng. Multiply and accumulate). U okviru jedne instrukcije vrši se množenje dva broja i dodavanje rezultata na prethodnu vrednost akomulatora.

Za izvršenje MAC instrukcije u C domenu kod TMS320C5535 arhitekture koristi se unutrašnja kompajlerska funkcija:

```
Int32 _smac(Int32 src, Int16 op1, Int16 op2)
```

Ova funkcija množi dva 16-obitna operanda *op1* i *op2* (poput _smpy funkcije u prethodnoj vežbi), rezultat množenja dodaje na vrednost parametra *src* i vraća dobijenu vrednost.

1.3 Sistemi sa konačnim i beskonačnim impulsnim odzivom (FIR i IIR)

Videli smo da je za linearni vremenski invarijantni sistem veza između odziva i pobude u vremenskom domenu predstavljena konvolucionom sumom. Funkcija prenosa ovakvog sistema se definiše kao količnik z transformacije signala odziva i signala pobude:

$$H(z) = \frac{Y(z)}{X(z)}$$

Funkcija prenosa sistema čije su ulazno-izlazne relacije predstavljene diferencijalnom jednačinom sa konstantnim koeficijentima je količnik dva polinoma po z^{-1} :

$$H(z) = \frac{Q(z^{-1})}{P(z^{-1})}$$

Ovim polinomima se mogu odrediti koreni, koji predstavljaju polove (polinom P) i nule (polinom Q).

Na osnovu funkcije prenosa mogu se razlikovati dva tipa sistema: sistemi sa konačnim impulsnim odzivom (FIR) i sistemi sa beskonačnim impulsnim odzivom (IIR). Kod FIR sistema, polinom P u jednačini prenosa ne postoji, dakle odziv sistema zavisi samo od ulaza sistema u posmatranom trenutku i prethodnom vremenu. Kod IIR sistema, odziv zavisi kako od ulaza, tako i od vrednosti izlaza u prethodnim vremenskim trenucima.

Uslov stabilnosti sistema je da svi polovi funkcije prenosa leže unutar jediničnog kruga. S obzirom da sistemi sa konačnim impulsnim odzivom nemaju polove, sledi da su uvek stabilni. Frekvencijski odziv sistema dobija se kada se u funkciji prenosa argument z zameni sa $e^{j\omega}$.

Uticaj polova i nula funkcije prenosa na karakteristike frekventnog odziva sistema predstavljen je sledećim pravilima:

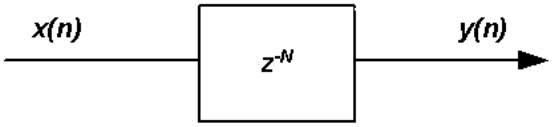
- Nula ili pol imaju najveći uticaj na deo frekvencijske karakteristike koji odgovara delu jediničnog kruga koji je najbliži posmatranoj nuli ili polu
- Približavanje nule i pola jediničnom krugu povećava njihov uticaj na frekvencijsku karakteristiku
- Za pol koji se nalazi blizu jediničnog kruga, amplitudska karakteristika ima lokalni maksimum na odgovarajućoj frekvenciji
- Za nulu koja se nalazi blizu jediničnog kruga, amplitudska karakteristika ima lokalni minimum na odgovarajućoj frekvenciji

1.4 Sistemi zasnovani na kašnjenju

1.4.1 Kašnjenje

Modul za kašnjenje signala (eng. *Delay*) predstavlja linearan vremenski invarijantan sistem sa memorijom. Vrednost izlaznog signala u momentu n jednak je vrednosti ulaznog signala u momentu *(n-N),* gde N predstavlja vrednost kašnjenja predstavljen brojem diskretnih odbiraka. Funkcija prenosa sistema definisana je sa:

$$H(z) = z^{-N}$$



Slika 3 – Modul za kašnjenje signala

Realizacija ovog modula podrazumeva pamćenje prethodnih N ulaznih odbiraka u memoriji sistema. Efikasna implementacija podrazumeva korišćenje kružnog bafera kod koga je pokazivač čitanja postavljen na memorijsku lokaciju koja se nalazi N mesta pre lokacije na koju pokazuje pokazivač pisanja.

Primer implementacije modula za kašnjenje signala sadrži sledeće promenljive koje služe za čuvanje informacija o stanju modula:

- *current_delay* – trenutna vrednost kašnjenja
- *delay_buffer* – memorijski prostor korišten za pamćenje odbiraka, statički zauzet, veličine maksimalne podržane vrednosti kašnjenja
- *dbuff_read_index* – pokazivač čitanja iz kružnog bafera
- *dbuff_write_index* – pokazivač pisanja u kružni bafer

Veličina memorijskog niza korišćenog za pamćenje odbiraka data je sa konstantom BUFFER_SIZE. Maksimalna vrednost kašnjenja jednaka je MAX_DELAY = BUFFER_SIZE-1.

U okviru modula definisane su tri funkcije. Prva u nizu jeste funkcija koja služi za inicijalizaciju modula za kašnjenje, *delay_init.* U okviru ove funkcije je realizovana inicijalizacija kružnog bafera (popunjavanje memorije nulama), kao i pokazivača za čitanje i pisanje u buffer. Parametar funkcije *N* predstavlja kašnjenje.

```
void delay_init(int N)
{
    clear_delay_buffer();
    current_delay = N;
    dbuff_read_index = 0;
    dbuff_write_index = current_delay;
}
```

Druga po redu funkcija jeste *set_current_delay*, koja služi za promenu vrednosti kašnjenja u toku izvršenja programa. U okviru funkcije računa se nova vrednost pokazivača čitanja, tako da kasni za pokazivačem pitanja N lokacija.

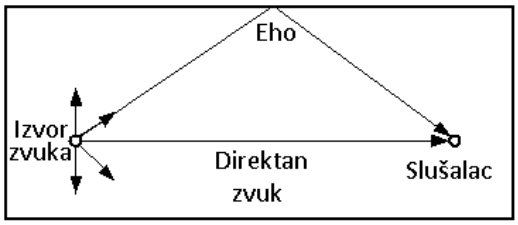
```
void set_current_delay(Int16 N)
{
    dbuff_read_index = (dbuff_write_index + BUFFER_SIZE - N) & MAX_DELAY;
    current_delay = N;
}
```

Poslednja funkcija jeste ujedno i glavna funkcija modula *delay*. Ova funkcija smešta prosleđeni odbirak u memoriju,a kao povratnu vredost vraća zakasneli odbirak. Svaki put kada se pozove funkcija, u bafer se upisuje vrednost ulaznog odbirka, i izvrši se uvećanje pokazivača pisanja. Sa pozicije na koju pokazuje pokazivač čitanja vrši se čitanje zakasnele vrednosti, koja ujedno predstavlja i povratnu vrednost funkcije, i nakon toga se povećava pokazivač čitanja.

```
Int16 delay(Int16 input)
{
    Int16 return_value;
    delay_buffer[dbuff_write_index] = input;
    dbuff_write_index++; dbuff_write_index = dbuff_write_index & MAX_DELAY;
    return_value = delay_buffer[dbuff_read_index];
    dbuff_read_index++; dbuff_read_index = dbuff_read_index & MAX_DELAY;
    return return_value;
}
```

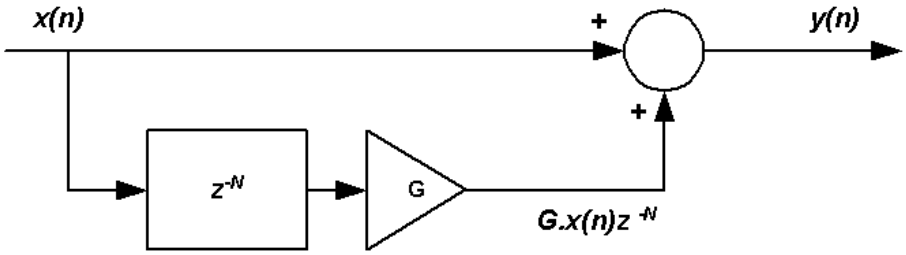
1.4.2 Eho

Eho efekat u prirodi nastaje kada do slušaoca, pored zvuka koj dopire direktno od izvora zvuka, dopire i zvučni signal odbijen od određenu prepreku. Signal odbijen od prepreku prelazi duži put do slušaoca, i deo energije signala biva apsorbovan od strane prepreke. Iz tog razloga komponenta odbijenog signala pomerena je u vremenu (kasni određeno vreme) i ima slabiji intenzitet u odnosu na signal koji dopire do slušaoca direktno od izvora zvuka.



Slika 4 – Eho efekat

Modul za proizvodnju Eho efekta funkcioniše tako što zakasneli odbirak signala pomnožen određenom vrednošću pojačanja dodajemo na trenutni ulazni odbirak. Vrednost pojačanja nalazi se u opsegu [0, 1) i simulira slabljenje signala. Zbir originalnog ulaznog i zakašnjenog oslabljenog odbirka predstavlja vrednost izlaznog odbirka.



Slika 5 – Sistem za dodavanje Eho efekta u signal

Funkcija prenosa modula za proizvodnju eho efekta data je sa:

$$H(z) = z + G \cdot z^{-N}$$

gde N predstavlja vrednost kašnjenja, a G vrednost pojačanja. Modul za dodavanje eho efekta se može implementirati koristeći realizovani modul za kašnjenje iz prethodnog zadatka. Implementacija ovog modula sastoji se iz dve funkcije. Prva funkcija je *echo_init* koja služi za inicijalizaciju sistema. Kao parametar prima kašnjenje „odbijene" komponente od prepreku i faktor pojačanja.

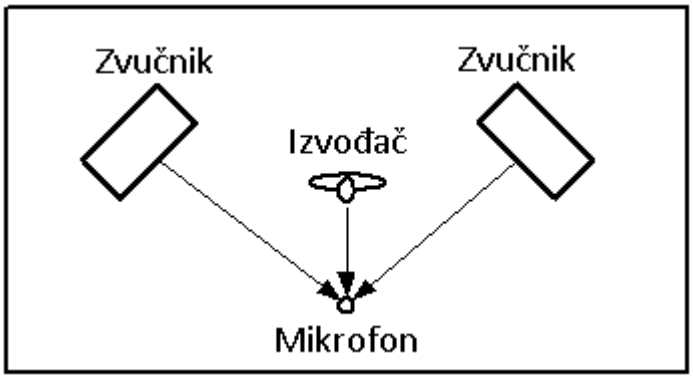
```
void echo_init(Int16 N, Int16 gain)
{
    delay_init(N);
    current_gain = gain;
}
```

Druga funkcija, *echo* pretstavlja funkciju obrade. Kao parametar prima ulazni udbirak u sistem, a kao povratnu vrednost daje vrednost odbirka na koji je dodat eho efekat. Za računanje zakasnelog odbirka koristi se *delay* modul realizovan u prethodnom zadatku. Primetite da je pojačanje predstavljeno kao označena celobrojna vrednost u opsegu [-32768, 32768). Ova vrednost predstavlja opseg pojačanja [-1.0, 1.0) preslikan na opseg označenog celobrojnog tipa. Na primer, vrednost pojačanja 16384 odgovara pojačanju 0.5, vrednost 8192 pojačanju 0.25 itd. Zbog toga, prilikom množenja vrednosti odbirka sa faktorom pojačanja koristiti se unutrašnja kompajlerska funkcija *_smpy*.

```
Int16 echo(Int16 input)
{
    Int16 delayed_input = delay(input);
    Int16 return_value = _smpy(delayed_input, current_gain);
    return_value += input;
    return  return_value;
}
```

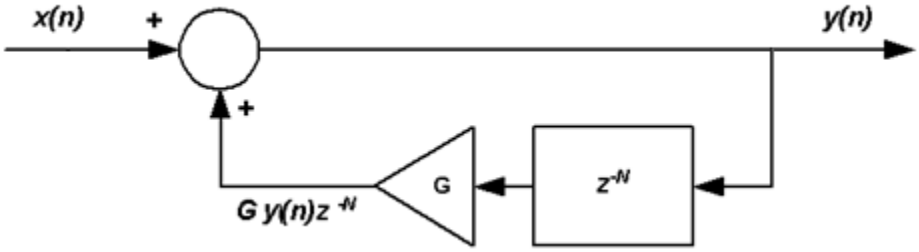
1.4.3 Reverberacija i konvoluciona reverberacija

Reverberacija je veoma slična Eho efektu. Razlika je u tome što do slušaoca ne dopire ponovljen ulazni signal sa određenim kašnjenjem, već zakasnela vrednost izlaznog signala. Primer nastanka reverberacije jeste na muzičkim koncertima, kada se iza izvođača nalaze zvučnici. U tom slučaju do mikrofona istovremeno dopire zvuk proizveden od strane izvođača i zvuk sa zvučnika.



Slika 6 – Efekar reverberacije

Pojednostavljeni modul za proizvodnju efekta reverberacije funkcioniše tako što se u memoriji pamti prethodnih N izlaznih odbiraka signala. Zakasneli izlazni odbirak signala pomnožen određenom vrednošću pojačanja (obično manje od 1) dodajemo na trenutni ulazni odbirak. Zbir ova dva odbirka predstavlja vrednost izlaznog odbirka.



Slika 7 - Sistem za dodavanje efekta reverberacije u signal

Funkcija prenosa modula za dodavanje efekta reverberacije definisana je sa:

$$H(z) = \frac{z}{1 - G \cdot z^{-N}}$$

U realnim uslovima reverberacija se veoma retko javlja kao ponovljen tačno jedan zakasneli izlazni odbirak, već to obično bude skup oslabljenih odbiraka sa različitim vrednostima kašnjenja. U sistemima za obradu zvuka efekat reverberacije najčešće se realizuje upotrebom tehnike koja se naziva **konvoluciona reverberacija.**

Konvoluciona reverberacija omogućava da se dodavanjem reverberacionog efekta modifikuje signal tako da on zvuči kao da je reprodukovan u nekoj karakterističnoj prostoriji ili okruženju. Ovaj efekat postiže se tako što se prvobitno izračuna odziv željenog okruženja ili prostorije na impulsnu pobudu. Impulsni odziv prostorije snimljen je u digitalnom formatu i smešten u memoriju diskretnog sistema. Vrednost odbiraka izlaznog signala računa se kao konvolucija ulaznog signala i impulsnog odziva okruženja, otuda i naziv konvoluciona reverberacija.

2 ZADACI

2.1 Zadatak 1

Cilj prvog zadatka jeste upoznavanje sa realizacijom modula za digitalno kašnjenje signala.

U okviru datoteke *delay.c* implementiran je modul za digitalno kašnjenje upotrebom kružnog bafera. Koristeći realizovani modul audio signal na levom kanalu zakašnjen je za 0.5s (8000 odbiraka pri frekvenciji odabiranja 16kHz).

1. Uvući projektni zadazak 1 u radni prostor
2. Izvršiti analizu izvornog koda u okviru *delay.c* datoteke
3. Pokrenuti program, i uporediti zvuk na jednom i drugom kanalu.

2.2 Zadatak 2

U okviru istog projektnog zadatka implementiran je modul za proizvodnju eho efekta koja koristi realizovani *delay* modul. Data su funkcije:

- `void echo_init(Int16 N, Int16 gain)` – u okviru koje je realizozovana inicijalizacija *delay* modula i postavljanje vrednosti pojačanja
- `Int16 echo(Int16 input)` - U okviru ove funkcije poziva se modul za kašnjenje koji vraća vrednost zakasnelog odbirka, zatim se taj zakasneli odbirak množi sa zadatim pojačanjem i sabira sa vrednošću ulaznog odbirka.

1. Otvoriti datoteku *echo.c* i izvršiti analizu izvornog koda.
2. Umesto digitalnog kašnjenja na levom kanalu, primeniti eho efekat za vrednosti kašnjenja odbirka 0.25s i pojačanja zakasnelog odbirka 0.5.
3. Pokrenuti program, i uporediti zvuk na jednom i drugom kanalu.

2.3 Zadatak 3

Cilj ovog zadatka je upoznavanje sa efektom reverberacije i realizacijom modula za dodavanje ovog efekta. U okviru projektnog zadatka realizovan je deo pojednostavljenog modula za proizvodnju efekta reverberacije u okviru datoteke *reverb.c.* R

Realizovana je funkcija za inicijalizaciju modula za reverberaciju slično kao i kod eho efekta. Razlika je što ovaj modul ne koristi postojeći modul za kašnjenje.

- `void simple_reverb_init(Int16 N, Int16 gain)` – inicijalizacija modula za dodavanje efekta reverberacije

Potrebno je implementirati funkciju za dodavanje efekta reverberacije:

- `Int16 simple_reverb(Int16 input)`

1. Implementirati funkciju *simple_reverb* prateći sledeće korake:
    a. Pročitati zakasneli odbirak iz memorije koristeći indeks čitanja *rbuff_read_index* i uvećati indeks po modulu

b. Pomnožiti vrednost pročitanog odbirka faktorom pojačanja *reverb_gain* koristeći funkciju *_smpy*
c. Sabrati rezultat množenja zakasnelog odbirka i faktora pojačanja sa ulaznim odbirkom
d. Upisati vrednost izlaznog odbirka u memoriju koristeći indeks pisanja *rbuf_write_index* i uvećati ineks po modulu
e. Vratiti vrednost izlaznog odbirka

2. Koristeći realizovanu funkciju primeniti efekat reverberacije na levi ulazni kanal za vredosti kašnjenja 4000 odbiraka i pojačanja 0.25.

2.4 Zadatak 4

U prethodno realizovanom zadatku memorija za smeštanje zakasnelih odbiraka kao i pokazivači za čitanje i pisanje implementirani su kao statičke promenljive vezane za dati modul. Mana ovakvog pristupa implementaciji diskretnih sistema jeste to što je nemoguće koristiti isti kod za instanciranje dva sistema sa različitim parametrima. Na primer ukoliko bi želeli da na jednom kanalu dodamo efekat reverberacije sa kašnjenjem od 0.2 sekunde a na drugom sa kašnjenjem 0.4 sekunde.

Kako bi se ovaj poroblem izbegao, i kako bi korisnici bili u mogućnosti da koriste implementirani digitalni sistem za različite parametre, praksa je da se koristi parametrizovana realizacija. Ovaj pristup podrazumeva da se svi potrebni elementi sistema (ulazni odbirci, memorijski buffer-i, vrednosti dužine buffer-a, pokazivači, koeficijenti, red sistema...) prosleđuju kao parametri funkcije.

U okviru datoteke *param_reverb.c* prikazana je parametrizovana realizacija funkcija za inicijalizaciju modula za dodavanje efekta reverberacije.

```
static void clear_reverb_buffer(Int16* buffer, Int16 buf_size)
{
	Int16 i;
	for (i = 0; i < buf_size; i++)
	{
		buffer[i] = 0;
	}
}

void param_reverb_init(Int16* buffer, Int16 buf_size, Int16* read_index, Int16*
write_index, Int16 N)
{
	clear_reverb_buffer(buffer, buf_size);
	*read_index = 0;
	*write_index = N;
}
```

Potrebno je realizovati parametrizovanu funkciju koja vrši istu obradu identičnu *simple_reverb* funkciji iz prethodnog zadatka.

- `Int16 param_reverb(Int16 input, Int16* buffer, Int16 buf_size, Int16* read_index, Int16* write_index, Int16 gain)`

gde parametri funkcije označavaju ulazni odbirak, početnu adresu kružnog buffer-a (potrebno zbog vraćanja pokazivača čitanja i pisanja na početak), veličina kružnog buffer-a, trenutni indeks čitanja, trenutni indeks pisanja i pojačanje. Indeksi čitanja i pisanja prosleđuju se po adresi, jer će im u okviru funkcije biti promenjena vrednost.

1. Na osnovu modula simple_reverb realizovati funkciju param_reverb. Umesto statičkih promenljivih koristiti parametre funkcije.
    a. Voditi računa da su indeksi čitanja i pisanja prosleđeni po adresi.
2. U okviru *main* datoteke inicijalizovati dva parametrizovana modula za dodavanje efekta reverberacije. Na levom kanalu dodati efekat reverberacije sa kašnjenjem 0.1 sekunde i pojačanja 0.3, a na desnom kanalu efekat reverberacije sa kašnjenjem 0.3 sekunde i pojačanja 0.5. Za realizaciju oba efekta koristiti parametrizovani modul za reverberaciju.
    a. Zauzeti dva memorijska niza veličine 8196 memorijskih lokacija *reverb_buffer_L* i *reverb_buffer_R*
    b. Definisati po dva indeksa za čitanje i pisanje *rb_write_L, rb_read_L* i *rb_write_R, rb_read_R*
    c. Na početku *main* funkcije pozvati funkciju za inicijalizaciju modula za dodavanje efekta reverberacije i za levi i za desni kanal sa odgovarajućim parametrima
    d. Dodati poziv funkcija za dodavanje efekta reverberacije nad odbircima primljenim sa levog i desnog kanala.
3. Prevesti program i pokrenuti.

2.5 Zadatak 5.

U okviru ovog zadatka data je nekompletna implementacija konvolucione reverberacije. U okviru niza *signal* nalazi se unapred snimljeni zvučni signal na koji je potrebno primeniti konvolucionu reverberaciju. Niz *impulse_response* sadrži impulsni odziv prećine. U okviru *main* funkcije poziva se operacija diskretne konvolucije između pomenuta dva signala, i rezultat se smešta u niz *output_signal*. Nakon toga vrši se reprodukcija ulaznog i izlaznog signala naizmenično. U okviru funkcije convoluton potrebno je implementirati operaciju diskretne konvolucije. Deklaracija funkcije data je sa:

- `void convolution(Int16* x, Uint16 xLength, Int16* h, Uint16 hLength, Int16* output )`

gde je *x* ulazni signal, *xLength* njegova dužina, *h* impulsni odziv, *hLength* njegova dužina, i *output* izlazni niz jednake dužine kao i ulazni.

1. Uvući pojektni zadatak 3 u radni prostor.
2. Implementirati funkciju convolution
3. Za računanje MAC instrukcije koristiti *_smac* funkciju.
4. Voditi računa da promenljiva Y u kojoj se akumuliraju vrednosti bude veličine Int32, kako ne bi došlo do prekoračenja opsega.
5. Pre smeštanja u izlazni niz, izvršiti skaliranje promenljive Y koristeći logički pomeraj u desno za 16 mesta.

6. Iscrtati vrednost ulaznog signala, izlaznog signala i vrednosti impulsnog odziva prostorije u vremenskom i frekventnom domenu korišćenjem alata *Graph -> Single Time* i *Graph -> Magnitude.*

3 Zaključak

U okviru ove vežbe upoznali ste se diskretnim sistemima za digitalnu obradu signala. Kroz praktične primere uvideli ste kako funkcioniše sistem za kašnjenje signala i na koje načine se može iskoristiti za proizvodnju željenih efekata. Naučili ste zašto i na koji način se u zvučnim signalima javlja eho efekat ili efekat reverberacije, kao i način da se ova dva efekta veštački izazovu upotrebom odgovarajućeg diskretnog sistema.

Operacija diskretne konvolucije predstavlja osnov većine diskretnih sistema za obradu signala. Iz tog razloga arhitektura DSP procesora prilagođena je izvršavanju ove operacije. Na primerima je pokazan jedan od načina za implementaciju ove operacije.

Naučili ste kako se upotrebom konvolucije na osnovu snimljene karakteristike određenog akustički specifičnog prostora može modifikovati određeni zvučni signal tako da zvuči kao da je reprodukovan u tom prostoru. Osnov za dalji rad i razumevanje narednih vežbi predstavlja razumevanje parametrizovane realizacije diskretnih sistema. Uvideli ste kako parametrizovana realizacija omogućava ponovnu upotrebu jednom implementiranog sistema.