# VEŽBA 4 – FIR filtri

## Potrebno predznanje

- Poznavanje programskog jezika C
- Urađena Vežba 1 – Uvod u Digitalnu Obradu Signala
- Odslušana predavanja iz predmeta OAiS DSP na temu Diskretni sistemi
- Odslušana predavanja iz predmeta OAiS DSP na temu FIR filtri

## Šta će biti naučeno tokom izrade vežbe

U okviru ove vežbe naučićete:

- Šta su to diskretni filtri
- Koje vrste diskretnih filtara postoje
- Koje su prednosti filtara sa konačnim odzivom
- Kako izgleda prenosna karakteristika, a kako impulsni odziv fitara sa konačnim odzivom
- Koja je osnovna podela filtara na osnovu prenosne karakteristike
- Na realnom primeru videćete kako prenosna karakteristika filtra utiče na filtrirani signal
- Kako red filtra utiče na prenosnu karakteristiku filtra sa konačnim odzivom
- Na koji način je moguće podeliti signal na frekventne podopsege i vršiti različitu obradu nad tim podopsezima

## Motivacija

Posmatrani signal uglavnom sadrži i nepoželjne spektralne komponente. Ako znamo frekvenciju, amplitudu i fazu neželjenih komponenti mogli bi ih ukloniti dodavanjem signalu istih tih komponenti ali fazno pomerenih za π (u kontra fazi). Nedostatak ovog metoda je potreba za tačnim poznavanjem karakteristika neželjenih komponenti u svakom trenutku vremena i za mogućnost generisanja odgovarajućeg signala čijom superpozicijom sa ulaznim signalom bi uklonili neželjene komponente. Potreban nam je sistem kojim možemo ukloniti neželjene komponente iz ulaznog signala bez potrebe za konstantom spektralnom analizom signala. Takav sistem bi konstrukcijom imao definisane spektralne opsege (spektralne komponente) koje bi potiskivao nezavisno od njihovih trenutnih vrednosti. Ostatak spektra (željeni deo signala) bi ostao nepromenjen. Jedan takav sistem je diskretni filtar. Najčešća izvedba diskretnog filtra je potiskivanje određenih spektralnih komponenti (nepoželjan deo signala) i propuštanje preostalih spektralnih komponenti (željeni deo signala).

# 1 TEORIJSKE OSNOVE

Diskretni (digitalni) filtri su jedan od najčešće primenjivanih sistema u digitalnoj obradi signala. Osnovna funkcija diskretnog filtra je frekvencijski-selektivna modifikacija ulaznog signala. Najčešće je reč o propuštanju bez izobličenja spektralnih komponenti signala iz jednog opsega učestanosti (propusni opseg) i anuliranju ostalih spektralnih komponenti (nepropusni opseg).

$$A(f) = \begin{cases} 1 & f \in propusni\ opseg \\ 0 & f \in nepropusni\ opseg \end{cases}$$

Diskretni filtri su linearni, vremenski invarijantni sistemi (LVIS) pa se generalno mogu opisati sa diferencijalnom jednačinom koja povezuje odbirke izlaznog signala y(n) sa odbircima ulaznog signala x(n):

$$y(n) = \sum_{k=0}^{L-1} a(k) \cdot x(n-k) - \sum_{k=1}^{M} b(k) \cdot y(n-k) .$$

Parametri filtra (koeficijenti) *a(k)* opisuju direktan deo koji je zavisan od ulaznih odbiraka. Parametri *b(k)* opisuju rekurzivni deo koji je zavisan od prethodnih izlaznih odbiraka. Postoje dva tipa diskretnih filtara, filtri sa konačnim impulsnim odzivom i filtri sa beskonačnim impulsnim odzivom. Za filtre sa konačnim impulsnim odzivom se koristi skraćenica *FIR (Finite Impulse Response)* i njih karakteriše da ne sadrže rekurzivni deo (*b=0*). Filtri sa beskonačnim impulsnom odzivom*, IIR (Infinite Impulse Response)*, imaju i diretan i rekurzivan deo.

Prenosna karakteristika filtra *H(f)*, preko z-transformacije $H_z(z)$, definiše se kao Furijeova transformacija diskretnog impulsnog odziva filtra *h(n)*:

$$x(n) = \delta(n) \Rightarrow y(n) = h(n) \quad \Leftrightarrow \quad H_z(z) = \sum_n h(n) \cdot e^{-2j\pi fn} = \frac{\sum_{k=0}^{L-1} a(k) \cdot z^{-k}}{1 + \sum_{k=1}^{M} b(k) \cdot z^{-k}} \quad \underset{z=e^{2j\pi f}}{\Leftrightarrow} \quad H(f) = H_z\left(z^{2j\pi f}\right) \quad \begin{matrix} A(f) = |H(f)| \\ \varphi(f) = \arg\{H(f)\} \end{matrix}$$

Amplitudna karakteristika *A(f)* definiše modifikaciju amplituda a fazna karakteristika *φ(f)* definiše modifikaciju faza spektralnih komponenti ulaznog signala.

## 1.1 Filtar sa konačnim odzivom (FIR)

Kao što je rečeno FIR filtri ne sadrže rekurzivni deo, već samo direktni, tako da se opisuju sa *L* koeficijenata filtra *a(k)*, gde je *k=0,...,L-1*. Parametar *L* predstavlja dužinu filtra, dok red filtra odgovara redu diferencijalne jednačine kojom je filtar opisan. Filtriranje signala predstavlja operaciju konvolucije ulaznog signala i koeficijenata FIR sistema

$$y(n) = \sum_{k=0}^{L-1} a(k) \cdot x(n-k)$$

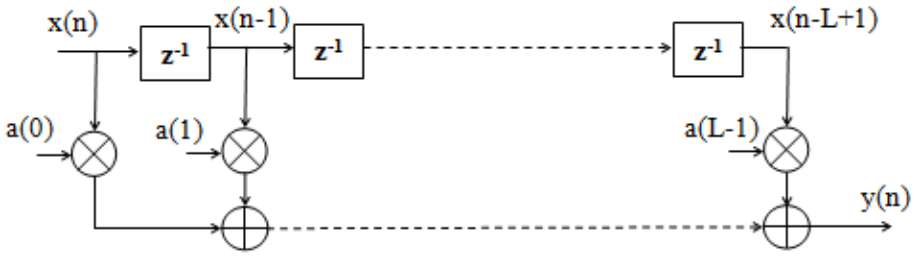
Koeficijenti FIR filtra jednaki su odbircima impulsnog odziva *h(n)*. Prenosna karakteristika *H(f)*, koja definiše i amlitudnu karakteristiku *A(f)* i faznu karakteristiku *φ(f)*, odgovara Furijeovoj transformaciji impulsnog odziva, i data je sa:

$$h(n)=\begin{cases} a(n) & n=0,\ldots,L-1 \\ 0 & drugde \end{cases} \Rightarrow H_z(z)=\sum_{k=0}^{L-1} h(k)\cdot z^{-k} \Rightarrow H(f)=H_z\left(e^{2j\pi f}\right) \Rightarrow A(f)=\left|H(f)\right| \quad \varphi(f)=\arg\{H(f)\}$$

Pošto je impulsni odziv FIR filtra konačan (*L* odbiraka počevši od *n=0*), FIR filtar je kauzalan i uvek stabilan jer nema polova (z-transformacija nema polinom u imeniocu).

Prednosti FIR filtra u odnosu na filtre sa beskonačnim odzivom su:

- garantovana stabilnost (nema polova)
- jednostavno ispunjavanje zahteva linearnosti fazne karakteristike
- poseduju manju osetljivost na kvantizacionu grešku koeficijenata od IIR filtra
- jednostavniji za implementaciju




Slika 1 - Prva kanonična forma FIR filtra

## 1.2 Implementacija FIR filtra

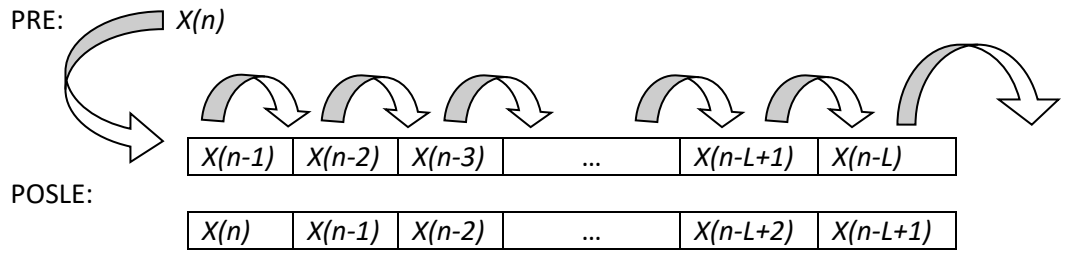
Na slici 1 prikazan je blok dijagram FIR filtra. Kao što je već pomenuto, vrednost izlaznog signala kod FIR filtra jednaka je rezultatu konvolucije ulaznog signala i koeficijenata FIR sistema. Za razliku od funkcije za vršenje konvolucije dva signala iz prethodnog zadatka, funkcija za FIR filtriranje prilagođena je izvršenju u realnom vremenu. To znači da je FIR filtar implementiran tako da za svaki odbirak ulaznog signala da jedan odbirak na izlazu. Iz jednačine filtra može se zaključiti da je za računanje izlaznog odbirka *y(n)* neophodno poznavanje vrednosti poslednjih *L* ulaznih odbiraka *x(n)…x(n-L+1),* gde je *L* dužina filtra. Iz tog razloga potrebno je u okviru funkcije za filtriranje vršiti skladištenje prethodnih *L* vrednosti ulaznog signala.

Na sledećem primeru data je realizacija funkcije za filtriranje primenom filtra sa konačnim impulsnim odzivom. Kod date implementacije kao parametri funkcije prosleđuju se:

- *input* – odbiraka ulaznog signala
- *coeffs* - niz sa koeficijentima kojima je definisan sistem
- *history* - niz koji predstavlja memoriju u kojoj će se čuvati *L* poslednjih ulaznih odbiraka (potrebni za računanje vrednosti izlaznih odbiraka u narednom bloku, n je jednako dužini filtra)
- *n_coeff* dužina filtra (broj koeficijenata)

```
Int16 fir_basic(Int16 input, Int16* coeffs, Int16 *history, Uint16 n_coeff)
{
    Int16 i;
    Int32 ret_val = 0;

    for (i = n_coeff - 2; i >= 0; i--)
    {
        history[i + 1] = history[i];
    }

    history[0] = input;

    for (i = 0; i < n_coeff; i++)
    {
        ret_val = _smac(ret_val, coeffs[i], history[i]);
    }

    return (Int16)(ret_val >> 16);
}
```

Prvi korak u okviru date funkcije jeste pomeranje vrednosti unutar memorije koja služi za skladištenje prethodnih odbiraka (*history*) za jedno mesto u desno. Time se ujedno odbacuje i najstariji odbirak (*x(n-L)*). Potom se vrednost ulaznog odbirka upisuje na prvo mesto unutar ovog niza.




Slika 2 – Pomeranje elemenata unutar *history* niza

Nakon izvršenog pomeranja pristupa se računanju same operacije konvolucije. Unutar *history* niza, nakon pomeranja, nalaze se odbirci poređani suprotno od redosleda kojim su primani (najnoviji odbirak se nalazi na prvom mestu, a najstariji na poslednjem). Iz tog razloga indeksiranje niza sa koeficijentima i indeksiranje niza sa odbircima vrši se direktno koristeći parametar *i.*

Za računanje kanvolucije potrebno je izvršiti operaciju „pomnoži i akumuliraj“ ili MAC (eng. Multiply and accumulate). Jedna od karakteristika DSP procesora je postojanje instrukcije koja ovu operaciju radi u jednom taktu (zbog njenog čestog korišćenja). Za izvršenje MAC instrukcije u C domenu kod TMS320C5535 arhitekture koristi se ugrađena 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.

Na kraju funkcije, kako bi se dobio 16-bitni rezultat neophodno je izračunatu vrednost podeliti sa $2^{16}$ (logički pomeraj u desno za 16 mesta).

Prikazana implementacija nakon svakog primljenog odbirka vrši pomeranje vrednosti koje zahteva L-1 čitanja i pisanja u memoriju. Ovaj problem jednostavno se rešava korišćenjem kružnog bafera. Na sledećem primeru data je implementacija funkcije FIR filtra korišćenjem kružnog bafera. Lista parametara koja se prosleđuje funkciji za filtriranje koja koristi kružni bafer jednaka je listi parametara prethodne funkcije uz dodatno polje *p_state* koje označava trenutni indeks pisanja u kružni bafer. Polje *p_state* se prosleđuje funkciji po adresi, kako bi se njegova vrednost mogla menjati unutar funkcije. Dužina niza *history* jednaka je broju koeficijenata (*n_coeff*).

```
Int16 fir_circular(Int16 input, Int16 *coeffs, Int16 *history, Uint16 n_coeff, Uint16
*p_state)
{
    Int16 i;
    Uint16 state;
    Int32 ret_val;

    state = *p_state;

    history[state] = input;
    if (++state >= n_coeff)
    {
        state = 0;
    }
    ret_val = 0;
    for (i = n_coeff - 1; i >= 0; i--)
    {
        ret_val = _smac(ret_val, coeffs[i], history[state]);
        if (++state >= n_coeff) /* incr state and check for wrap */
        {
            state = 0;
        }
    }

    *p_state = state;

    return (Int16)(ret_val >> 16);
}
```

U samoj implementaciji funkcije prva razlika jeste izbacivanje pomeranja vrednosti unutar *history* memorijskog niza. Umesto toga vrši se upis vrednosti ulaza na mesto *state* unutar *history* memorijskog niza, i povećanje promenljive *state* po modulu, tako da state predstavlja indeks poslednjeg upisanog elementa. Promenljiva *state* predstavlja lokalnu kopiju vrednosti promenljive *p_ state.*

Računanje MAC operacije vrši se takođe korišćenjem funkcije *_smac.* Prilikom računanja MAC operacije niz sa koeficijentima indeksira se koristeći parametar *i* čija vrednost ide od poslednjeg koeficijenta do prvog (i = *n-k, k=1, 2 … n*). Indeksiranje *history* niza se vrši korišćenjem promenljive *state* koja se u svakoj iteraciji povećava po modulu.

Pre samog završetka funkcije potrebno je vrednost lokalne promenljive *state* dodeliti prosledjenoj promenljivoj *p_state.*

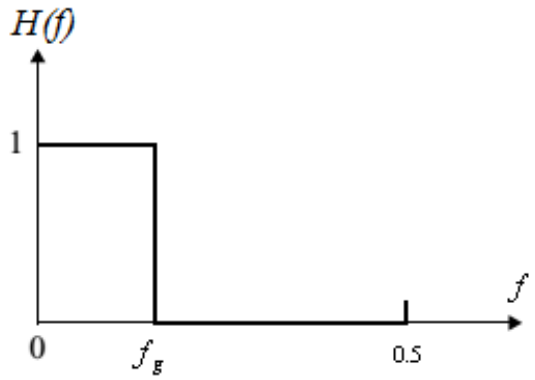
## 1.3 Podela filtara u zavisnosti od amplitudne karakteristike

U zavisnosti od amplitudne prenosne karakteristike filtra definišu se tri osnovna tipa filtara. Svi ostali filtri mogu dobiti njihovom kombinacijom.

***Idealni nisko-propusni filtar (NF)*** propušta samo komponente čije su učestanosti manje od zadate granične učestanosti *fg* a sve komponente čije učestanosti su iznad *fg* nulira:

$$A_{NF}(f)=\begin{cases}1 & |f|\le f_g\\ 0 & |f|>f_g\end{cases}\quad \varphi_{NF}(f)=-2\pi f\tau \quad\Rightarrow\quad H_{NF}(f)=\begin{cases}e^{-2j\pi f\tau} & |f|\le f_g\\ 0 & |f|>f_g\end{cases}.$$

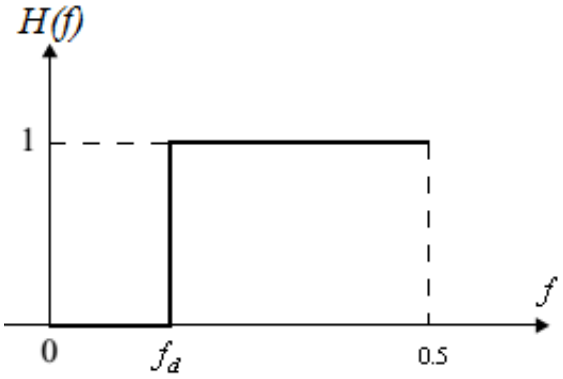
Uz pretpostavku da je učestanost *f* normalizovana na učestanost odabiranja ($f_s=1$ *i* $T_s=1/f_s=1$), granična učestanost $f_g$ može biti u opsegu od 0 do 0.5.




Slika 3 - Idealan nisko-frekventni filtar

***Idealni visoko-propusni filtar (VF)*** ne propušta komponente do zadate granične učestanosti koja je nazvana donja granična učestanost $f_d$:

$$A_{VF}(f)=\begin{cases}0 & f\le f_d\\ 1 & f_d<f\le 0.5\end{cases}\quad \varphi_{VF}(f)=-2\pi f\tau \quad\Rightarrow\quad H_{VF}(f)=A(f)\cdot e^{j\varphi(f)}.$$

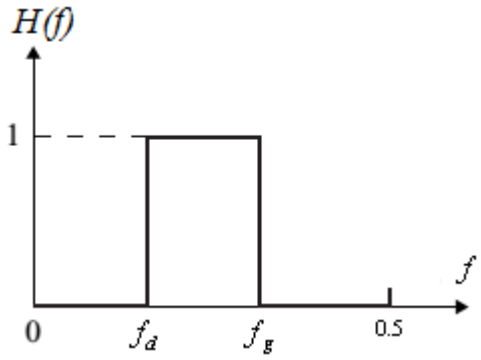



Slika 4 - Idealan visoko-frekventni filtar

***Idealni pojasni filtar*** (PF) propušta samo komponente iz zadatog opsega učestanosti od $f_d$ do $f_g$:

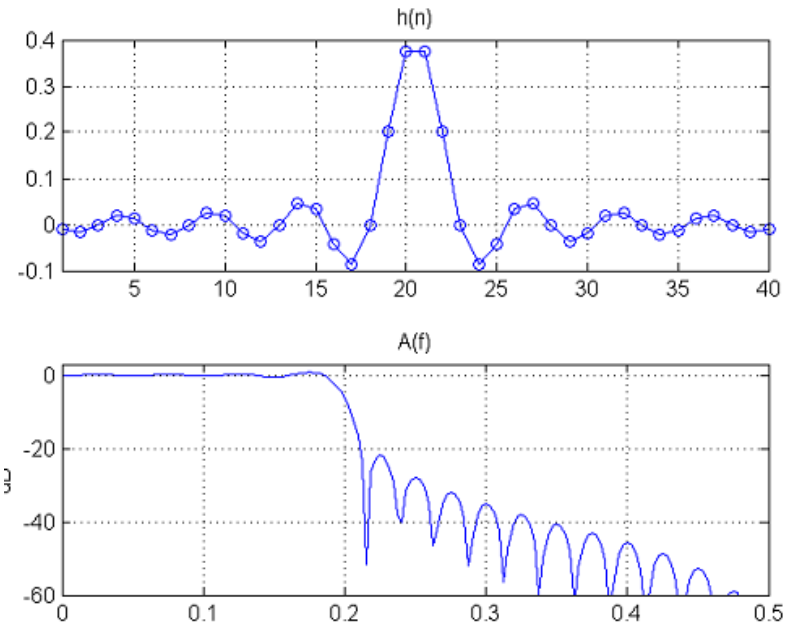
$$A_{PF}(f)=\begin{cases}1 & f_d \le f \le f_g \\ 0 & drugde\end{cases} \quad \varphi_{PF}(f)=-2\pi f\tau \quad \Rightarrow \quad H_{PF}(f)=A(f)\cdot e^{j\varphi(f)}.$$

PF filtar se češće definiše preko centralne učestanosti $f_C=(f_d+f_g)/2$ i širine propusnog opsega $B=f_g-f_d$. U tom slučaju propusni opseg je definisan sa donjom učestanošću $f_C-B/2$ i gornjom učestanošću $f_C+B/2$.




Slika 5 - Idealan pojasni filtar

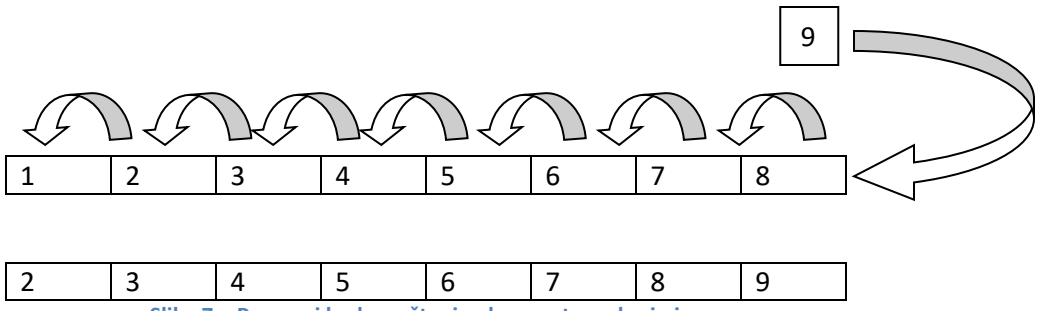
Na prethodnim slikama prikazana je prenosna karakteristika za idealan filtar, međutim, impulsni odziv takvog filtra je beskonačan i ne ispunjava uslov kauzalnosti. Kako bi se omogućila implementacija filtara u realnim sistemima, vrši se aproksimacija idealne prenosne karakteristike. Na slici 5 (gornji prikaz) prikazan je impulsni odziv idealnog NF filtra (linija) i njegova aproksimacija u konačnom broju tačaka (kružići). Ispod toga prikazana je prenosna karakteristika filtra čiji su koeficijenti tačke aproksimacije impulsnog odziva.




Slika 6 - Impulsni odziv i prenosna karakteristika realnog NF filtra za $f_g$=0.2

## 1.4 Dodatak 1: Kružni bafer

U oblasti digitalne obrade signala česta je potreba za čuvanjem prethodnih N vrednosti datog signala (jedan od najčešćih primera jeste filtar sa konačnim odzivom). Pomenute vrednosti možemo čuvati upotrebom prostog linearnog bafera tako što ćemo nakon dobijanja svakog odbirka trenutne vrednosti bafera pomeriti za jedno mesto u levo, i novopridošli odbirak smestiti na kraj.

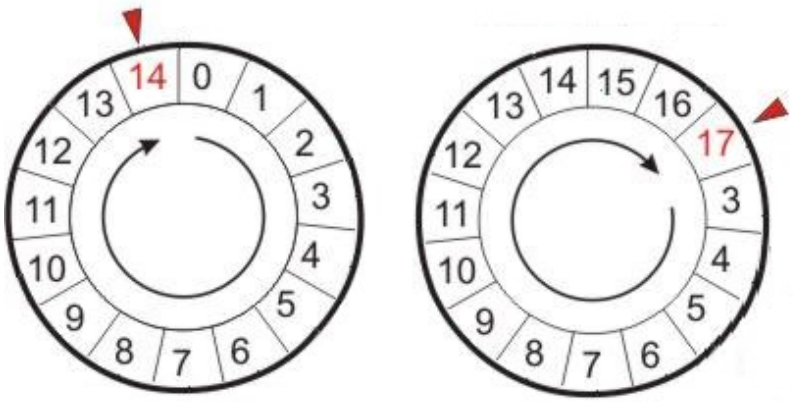



Slika 7 – Pomeraj kod smeštanja elemenata na kraj niza

Ovakav pristup može biti veoma neefikasan u slučaju velikih bafera, jer za svaki primljeni odbirak, vrši se N-1 pomeranja (gde je N veličina bafera).

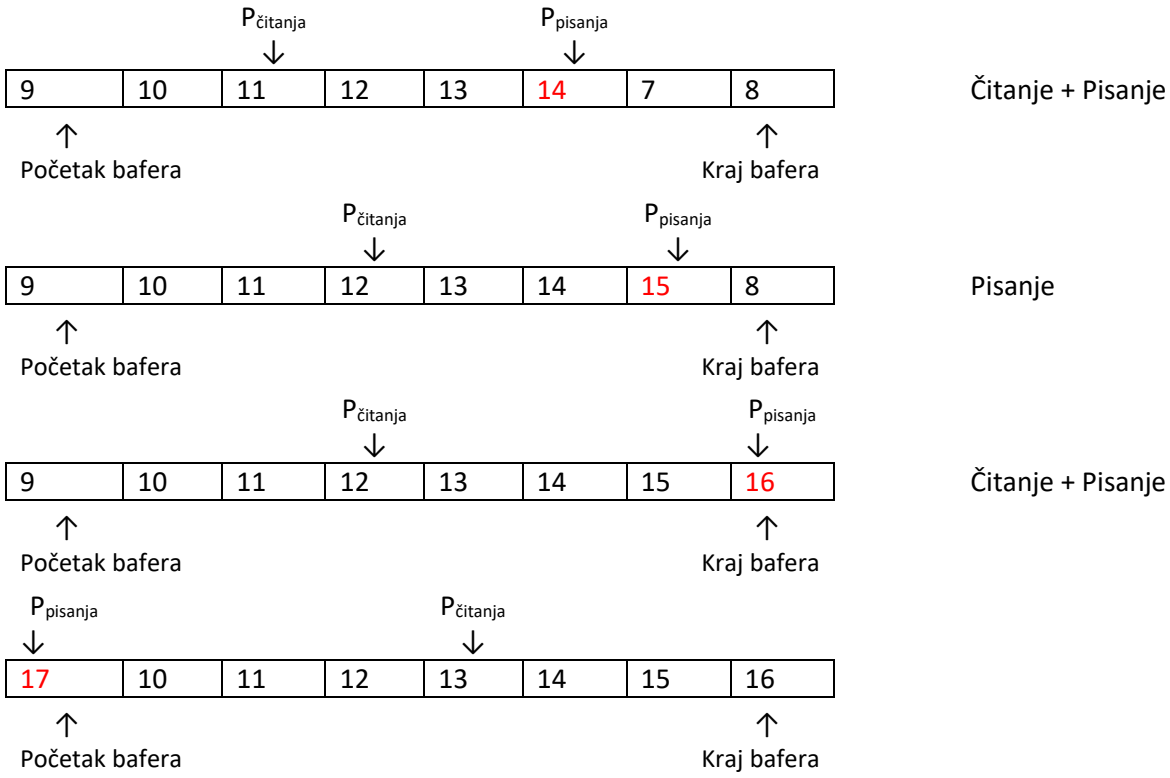
Drugi pristup, znatno efikasniji jeste korišćenje tzv. kružnog ili cirkularnog bafer.

Koncept kružnog bafera zasniva se na posmatranju alociranog memorijskog prostora kao krug.




Slika 8 – Prikaz kružnog bafera

Kružni bafer je realizovan tako što uz alocirani memorijski prostor pamtimo i dva pokazivača, pokazivač upisa i pokazivač čitanja. Pokazivač upisa pokazuje na lokaciju koja će biti korišćena za smeštanje naredne vrednosti, a pokazivač čitanja najstariju nepročitanu vrednost u baferu. Pokazivač pisanja uvećava se nakon svake operacije pisanja, a pokazivač čitanja nakon svake operacije čitanja. Pokazivači se uvećavaju po modulu veličine bafer-a. Dakle pokazivač se uvećava dok ne pokazuje na poslednju alociranu lokaciju u memoriji (pokazivač == kraj bafera). U narednom pokušaju povećanja pokazivača, umesto uvećanja za jednu lokaciju, pokazivaču će biti dodeljena vrednost adrese prve lokacije u alociranom baferu (pokazivač = početna adresa bafera). Na narednoj slici prikazan je primer čitanja i pisanja vrednosti u i iz kružnog bafera.




**Slika 9 – Prikaz rada kružnog bafera**

# 2 ZADACI

Napomena: U toku izrade, za svaki signal za koji je rečeno da je potrebno iscrtati, potrebno je sačuvati prikaz kao slikovnu datoteku. Za čuvanje koristiti *PrtScn* i neki od prostih alata za obradu slike (dovoljan je i *MS Paint*). Nazivi slikovnih datoteka treba da budu u formatu *Zadatak_X_Slika_Y.*

## 2.1 Zadatak 1

1. Uvući projektni zadatak 1 u radni prostor
2. Povezati *line out* izlaz vašeg računara na *line in* ulaz na razvojnoj ploči.
3. Na računaru reprodukovati datoteku *signal1.wav*
4. Pokrenuti program, zaustaviti izvršenje i iscrtati vrednost primljenog signala na jednom kanalu u vremenskom i frekventnom domenu.

**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 bloka sa vrednostima signala ili broju koeficijenata za iscrtavanje odziva filtra. Polje *Dsp Data Type* postaviti na *16 bit signed integer*. Polje *Starting Address* treba da sadrži adresu početka niza sa odbircima ili koeficijentima.

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 *Data Plot Style* postaviti na *Bar. FFT Order* postaviti na 10.

Ova podešavanja koristiti prilikom iscrtavanja svih signala.

**Napomena:** Za potrebe drugog zadatka, pretpostavćemo da nam prva komponenta signala (komponenta najniže frekvencije) u frekventnom domenu predstavlja signal od važnosti, a da su ostale komponente smetnje u signalu i da ih je potrebno ukloniti. Za potrebe trećeg zadatka pretpostavićemo da komponenta najviše frekvencije predstavlja signal od važnosti, dok ćemo u četvrtom zadatku pretpostaviti da je signal od važnosti središnja komponenta.

## 2.2 Zadatak 2

U okviru projektnog zadatka dat je primer parametrizovane realizacije diskretnog sistema sa konačnim odzivom (FIR). S obzirom na činjenicu da kod diskretnih sistema sa konačnim odzivom polinom P ne postoji, funkcija prenosa predstavljena je polinomom $Q(z)$. Prenosna karakteristika ovakvog sistema određena je redom datog polinoma i koeficijentima.

Vrednost izlaznog signala jednaka je rezultatu konvolucije ulaznog signala i koeficijenata FIR sistema.

Kod parametrizovane implementacije sistema sa konačnim odzivom, kao parametre funkcije potrebno je proslediti:

- niz ulaznih odbiraka signala

- niz sa koeficijentima kojima je definisan sistem
- niz koji predstavlja memoriju u kojoj će se čuvati *n* poslednjih ulaznih odbiraka (potrebni za računanje vrednosti izlaznih odbiraka u narednom bloku, n je jednako redu filtra)
- red sistema (broj koeficijenata)
- veličinu bloka obrade (broj ulaznih/izlaznih odbiraka)
- niz u koji će biti smeštene izračunate vrednosti izlaznog signala

```
Int16 fir_basic(Int16* input, Int16* coeffs, Int16 *z, unsigned short order,
                unsigned short nx, Int16* output);
```

1. Uvući projektni zadatak 4 u radni direktorijum.
2. Analizirati datu funkciju *fir_basic*
3. Implementirati funkciju:

```
Int16 fir_circular(Int16* input, Int16 *coeffs, Int16 *z, unsigned short order,
                   unsigned short *p_state, unsigned short nx, Int16 *output);
```

koja za smeštanje odbiraka u memoriju koristi kružni buffer. Funkciji koja sadrži realizaciju diskretnog sistema sa upotrebom kružnog buffer-a potrebno je proslediti i pokazivač pisanja u kružni buffer. Pokazivač je potrebno proslediti po referenci, pošto se vrednost pokazivača menja unutar funkcije.

4. Filtrirati signal na jednom kanalu koristeći funkciju *fir_circular*, a na drugom kanalu koristeći *fir_basic*.
5. Prevesti i pokrenuti program. Izlazni signali na levom i desnom izlaznom kanalu moraju biti identični.

## 2.3 Zadatak 3

1. U okviru *main* funkcije uključiti zaglavlje sa koeficijentima **NF** (*lowpass*) filtra 8-og reda.

   Dati koeficijenti predstavljaju nisko frekventni filtar sa graničnom frekvencijom 5 kHz.

2. Filtrirati ulazni signal na oba kanala upotrebom funkcije *fir_circular*.
3. Pokrenuti program, zaustaviti izvršenje upotrebom tačke prekida.
4. Iscrtati impulsni odziv filtra (koeficijenti filtra u vremenskom domenu) i prenosnu karakteristiku filtra. Za iscrtavanje prenosne karakteristike filtara koristiti alat *FFT Magnitude Phase* koji prikazuje faznu i amplitudnu karakteristiku u različitim prozorima.

   **Napomena:** Za prikaz fazne karakteristike potrebno je podesiti opseg koji će se prikazati. Pritisnuti desni taster miša na grafik i isključiti opciju *AutoScale*. Zatim otvoriti podešavanja grafika i za Min Phase i Max Phase postaviti vrednosti (-5, 5). Ovaj opseg je dovoljan pošto je fazna karakteristika prikazana u opsegu $-\pi, \pi$.

5. Iscrtati vrednosti izlaznog signala u vremenskom i frekventnom domenu, nakon filtriranja. Uporediti sa prikazom ulaznog signala prikazanim u prethodnom zadatku.
6. Šta se dešava sa frekvencijama višim od granične, a šta sa nižim?
7. Ponoviti korake 1-5 za nisko frekventne filtre 32-og i 120-og reda sa istom graničnom frekvencijom.

8. Komentarisati uticaj reda filtra na prenosnu karakteristiku. Čime je ograničeno povećanje reda filtra? Kako utiče povećanje reda filtra na zahteve u pogledu memorije i brzine izvršavanja?

## 2.4 Zadatak 4

1. U okviru *main* funkcije uključiti zaglavlje sa koeficijentima **VF** (*highpass*) filtra 32-og reda.

Dati koeficijenti predstavljaju visoko frekventni filtar sa graničnom frekvencijom 12 kHz.

2. Filtrirati ulazni signal na oba kanala upotrebom funkcije *fir_circular*.
3. Pokrenuti program, zaustaviti izvršenje upotrebom tačke prekida.
4. Iscrtati impulsni odziv filtra (koeficijenti u vremenskom domenu) i prenosnu karakteristiku (FFT nad koeficijentima) filtra.
5. Iscrtati vrednosti izlaznog signala u vremenskom i frekventnom domenu, nakon filtriranja. Uporediti sa prikazom ulaznog signala prikazanim u prvom zadatku.
6. Šta se dešava sa frekvencijama višim od granične, a šta sa nižim?

## 2.5 Zadatak 4

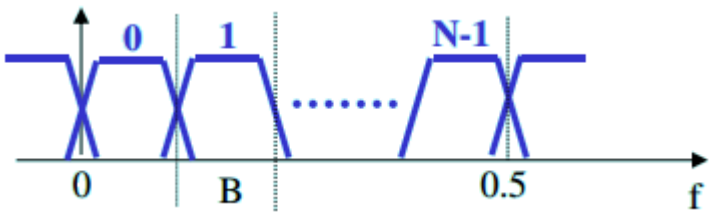
1. U okviru *main* funkcije uključiti zaglavlje sa koeficijentima **PF** (*highpass*) filtra 32-og reda.

Dati koeficijenti predstavljaju pojasni filtar sa graničnim frekvencijama 5 kHZ i 12 kHz.

2. Filtrirati ulazni signal na oba kanala upotrebom funkcije *fir_circular.*
3. Pokrenuti program, zaustaviti izvršenje upotrebom tačke prekida.
4. Iscrtati impulsni odziv filtra (koeficijenti u vremenskom domenu) i prenosnu karakteristiku (FFT nad koeficijentima) filtra.
5. Iscrtati vrednosti izlaznog signala u vremenskom i frekventnom domenu, nakon filtriranja. Uporediti sa prikazom ulaznog signala prikazanim u prvom zadatku.
6. Šta se dešava sa frekvencijama koje upadaju u opseg između graničnih frekvencija? A šta sa onima van zadatog opsega?

## 2.6 Zadatak 5

Ponekad je prilikom digitalne obrade signala potrebno primeniti različitu obradu nad spektralnim komponentama signala koje pripadaju različitim frekventnim podopsezima. Za postizanje ovakvog efekta moguće je signal podeliti filtriranjem na više signala, gde svaki signal sadrži spektralne komponente iz jednog podopsega. Prilikom podele, raspodela podopsega može biti uniformna i neuniformna.




Slika 10 - Podela spektra na podopsege upotrebom pojasno propusnih filtara

Kada imamo na raspolaganju komponente signala, moguće je nad svakom komponentom vršiti željenu obradu.

1. Upotrebom kernela za filtre 32-gog reda korišćene u prethodnim zadacima podeliti ulazni signal na 3 podsignala. Granične vrednosti opsega su:
    a. Opseg 1 = 0 – 5kHz
    b. Opseg 2 = 5kHz – 12kHz
    c. Opseg 3 = 12kHz – 24kHz
2. Izračunati vrednost izlaznog signala sabiranjem odbiraka različitih komponenti, pri tom vrednosti signala iz Opsega 1 oslabiti za 25%, a vrednosti signala iz opsega 3 za 50%.

$$y[n] = 0.75 \cdot x_1 + x_2 + 0.5\, x_3$$

3. Iscrtati vrednost izlaznog signala u vremenskom i frekventnom domenu.

# 3 Zaključak

U okviru ove vežbe na primeru su pokazani rezultati filtriranja određenog signala. Naučili ste kako funkcionišu tri osnovna tipa filtra u zavisnosti od prenosne karakteristike (niskofrekventni, visokofrekventni i pojasni filtar). Videli ste kako prenosna karakteristika filtra utiče na to koje će spektralne komponente signala biti oslabljene.

U relanim sistemima je nemoguće implementirati idealan filtar, iz tog razloga koristi se aproksimacija prenosne karakteristike. Na prikazanim primerima ste uvideli kako red filtra sa konačnim odzivom utiče na prenosnu karakteristiku. Prilikom projektovanja realnih sistema potrebno je naći optimum između zahteva za preciznošću filtra i računarskih zahteva. Uvideli ste da su za svako povećanje reda filtra za jedan potrebne dve dodatne memorijske lokacije, jedna za koeficijent i jedna za smeštanje elemenata prethodnog bloka (blok za kašnjenje). U pogledu zahteva dodatnih instrukcija, za svako povećanje reda filtra za 1, imamo jednu dodatnu MAC instrukciju prilikom računanja svakog izlaznog odbirka.

U toku ove vežbe upoznali ste se sa načinima na koje je moguće istisnuti neželjene spektralne komponente iz signala. Pored toga upoznali ste se sa metodom deljenja signala na podopsege kako bi vršili različitu obradu nad različitim podopsezima signala.

Tema naredne vežbe su filtri sa beskonačnim odzivom (IIR). U okviru te vežbe uporedićemo performanse IIR filtara u odnosu na FIR, i izdvojiti mane i prednosti jednog i drugog pristupa.