Realizacija digitalnih filtara na ADSP-21489 EZ-KIT Lite

16. decembar 2013

1 Upoznavanje sa radnim okruženjem VisualDSP++

Pokrenuti VisualDSP++ **Programs** > **Analog Devices** > **VisualDSP++ 5.0** > **VisualDSP++ Environment**. Otvoriti projekat *21489 AD1939 C Sampled-Based Talkthru 48 or 96 kHz*: **File** > **Open** > **Project** i izabrati projekat koji se nalazi na putanji `<folder_u_kojem_je_instaliran_VisualDSP++>\Analog Devices\VisualDSP 5.0\214xx\Examples\ADSP-21489 EZ-Board\21489 AD1939 C Sampled-Based Talkthru 48 or 96 kHz`. U lijevom dijelu radnog prostora otvoriti listu *Source Files*. U listi *DSP System Init* nalazi se kod kojim se inicijalizuje razvojni sistem, a u listi *DSP Audio Processing Routines* kod kojim je implementirana obrada signala. Za potrebe ovog zadatka dovoljno je uočiti da se u listi *DSP System Init* u fajlu init1939viaSPI.c može izabrati vrijednost frekvencije odmjeravanja koju koristi kodek. Ova vrijednost može biti 48 kHz ili 96 kHz i podešava se pretprocesorskom direktivom:

```
#define USE_48_KHZ_SAMPLE_RATE
```

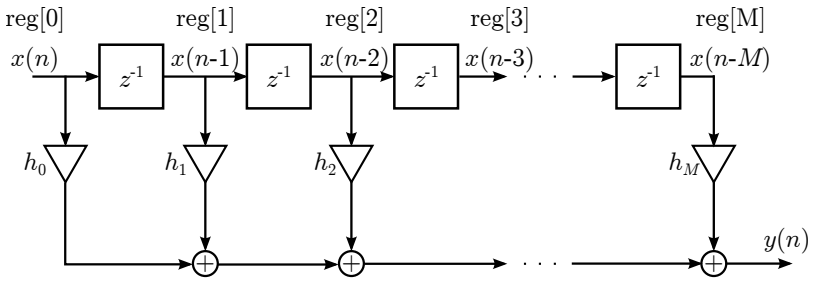
odnosno,

```
#define USE_96_KHZ_SAMPLE_RATE
```

Koristićemo frekvenciju odmjeravanja od 48 kHz pa je potrebno aktivirati odgovarajuću direktivu, a drugu staviti pod komentar.

U ovom primjeru stvarna obrada je data kodom u funkciji `process_AD1939_samples` koja se nalazi u fajlu `process_audio.c` u listi *DSP Audio Processing Routines*. Ovaj kod se izvršava za svaki odmjerak ulaznog signala. Ovo se postiže tako što je u fajlu `Main.c` funkcija `process_AD1939_samples` postavljena kao funkcija za obradu prekida koje A/D konvertor generiše za svaki odmjerak

```
interrupts(SIG_SP1, process_AD1939_samples);
```



Slika 1: Realizacija FIR filtra direktnom formom.

U funkciji `process_AD1939_samples` obrada počinje pozivom funkcije `Receive_ADC_Samples` koja je implementirana u fajlu `SPORT1_ISR_process_samples_.asm`. Ova funkcija prima ulazne podatke sa A/D konvertora i smješta ih u promjenljive koje odgovaraju lijevim i desnim kanalima za ulaze 1 i 2: `Left_Channel_In1`, `Right_Channel_In1`, `Left_Channel_In2`, `Right_Channel_In2`.

Nakon završene obrade, funkcija `Transmit_ADC_Samples` prenosi izlazne podatke smještene u promjenljivim `Left_Channel_OutX` i `Right_Channel_OutX`, gdje je X=1, 2, 3 ili 4, na četiri stereo D/A konvertora.

U datom primjeru odmjerci sa ulaza 1 se prenose na izlaze 3 i 4. Odmjerci sa ulaza 2 se prenose na izlaz 3. Takođe, odmjerci lijevog kanala sa ulaza 2 se prenose na lijevi kanal na izlazu 2, a odmjerci desnog kanala sa ulaza 2 se sabiraju sa zakašnjenim odmjercima desnog kanala sa ulaza 2 i šalju na desni kanal na izlazu 2 čime se na desnom kanalu na izlazu 2 realizuje efekat eha. Obratite pažnju na realizaciju linije za kašnjenje u ovom primjeru.

Pomoću **Project > Build Project** čitav projekat se kompajlira, linkuje i učitava u memoriju DSP-a. Izvršavanje programa se započinje pomoću **Debug > Run**. Na ulaz 1 ili 2 dovedite audio signal i poslušajte izlazne signale na izlazima 1-4. Izvršavanje se prekida pomoću **Debug > Halt**. Ranije kompajlirani projekat se može učitati u memoriju DSP-a pomoću **File > Load Program**.

Aktivni projekat se zatvara pomoću **File > Close > Project <Ime projekta>**.

2 Realizacija FIR filtra

Na Slici 1 prikazana je blok-šema realizacije FIR filtra M-tog reda direktnom formom. Odziv ovog filtra dat je sa

$$y(n) = \sum_{k=0}^{M} h_k x(n-k). \tag{1}$$

U realizaciji FIR filtra direktnom formom mogu se uočiti dvije osnovne operacije: pomjeranje odmjeraka u pomjeračkom registru i izračunavanje

sume proizvoda koeficijenata filtra i odmjeraka u pomjeračkom registru.

Neka sadržaj niza `reg` odgovara sadržaju pomjeračkog registra. Kada od A/D konvertora stigne jedan odmjerak signala potrebno je odmjerke u pomjeračkog registru pomjeriti za jedno mjesto udesno, kao što je dato sljedećim kodom:

```
for (i = NB; i > 0; i--) {
    reg[i] = reg[i-1];
}
```

nakon čega se u `reg[0]` smješta novi odmjerak:

```
reg[0] = Left_Channel_In1;
```

Slijedi izračunavanje sume proizvoda koeficijenata filtra i odmjeraka u pomjeračkom registru, što odgovara jednačini 1.

```
sum = 0.0;
for (i = 0; i < NB; i++) {
    sum += B[i] * reg[i];
}
```

Konačno, odziv filtra treba pripremiti za prosljeđivanje na D/A konvertor

```
Right_Channel_Out1 = sum;
```

Dakle, fajl `process_audio.c` treba da ima sljedeći sadržaj:

```
#include "ADDS_21489_EzKit.h"
#include "fircoefs.h"              /* koeficijenti filtra */

float reg[NB];                      /* pomjeracki registar */

void  process_AD1939_samples(int sig_int)
{
    int i;
    float sum;

    Receive_ADC_Samples();         /* odmjerci sa A/D konvertora */
                                   /* se upisuju u promjenljive */

    /* pomjeracki registar */
    for (i = NB; i > 0; i--) {
        reg[i] = reg[i-1];
    }
```

```
    reg[0] = Left_Channel_In1;     /* novi odmjerak se upisuje u registar */

    /* konvoluciona suma */
    sum = 0.0;
    for (i = 0; i < NB; i++) {
        sum += B[i] * reg[i];
    }

    /* izlaz iz filtra se prosljedjuje na D/A konvertor */
    Left_Channel_Out1 = sum;

    /* odmjerak sa desnog kanala proslijediti na izlaz */
    Right_Channel_Out1 = Right_Channel_In1;

    Transmit_DAC_Samples();
}
```

U ovom primjeru, koeficijenti FIR filtra se nalaze u zaglavlju `fircoefs.h` koje se generiše MATLAB funkcijom `exportCoefs.m`. Funkcija se poziva na sljedeći način:

```
exportCoefs(b, a, ime_fajla)
```

gdje se `b` i `a` vektori koeficijenata filtra, a `ime_fajla` je string koji sadrži ime fajla u koji će koeficijenti biti upisani. Funkcija se može iskoristiti kako za IIR tako i za FIR filtre. U slučaju da se radi o FIR filtru, drugi argument treba da bude prazan vektor `[]`.

Kreirati novi projekat **File** > **New** > **Project**. Sada treba postaviti opcije projekta u nekoliko narednih okvira za dijalog:

- U polju **Project types** izabrati *Standard application* i u polju **Name** unijeti ime projekta, npr *FIR filtar*. Kliknuti na **Next** >.
- U polju **Processor family** izabrati *SHARC*, a u polju **Processor types** izabrati *ADSP-21489*. Kliknuti na **Next** >.
- Isključiti kreiranje inicijalnog koda **Add template source code to the application**. Kliknuti na **Next** >.
- U poslednjem okviru za dijalog samo kliknite na **Finish**.

Sada je kreiran novi projekat i potrebno je dodati kod kojim se inicijalizuje razvojni sistem. Ovaj kod se nalazi u arhivi `kostur.zip` koja se može preuzeti sa web stranice predmeta. Raspakovati sadržaj arhive `kostur.zip` u folder u kojem je smješten novi projekat – obično je to folder `Visual DSP Projects` u `My Documents` folderu korisnika.

Korišćenjem MATLAB-a, pomoću metode prozorskih funkcija (`help fir1`) projektovati niskopropusni FIR filtar reda $N = 200$ sa graničnom frekvencijom 1 kHz. U MATLAB-u izračunati i nacrtati njegovu frekvencijsku karakteristiku. Smatrati da je frekvencija odmjeravanja 48 kHz. Generisati zaglavlje sa koeficijentima filtra pomoću funkcije `exportCoefs`. Kopirati zaglavlje u folder u kojem je smješten projekat.

Dodati u projekat **Project** > **Add to Project** > **File(s)** sve fajlove iz foldera u kojem je smješten projekat.

Otvoriti fajl `process_audio.c` i unijeti kod kojim se realizuje FIR filtar. Kompajlirati i linkovati projekat, te učitati izvršni kod u memoriju DSP-a. Pokrenuti program i testirati ga dovođenjem zvučnog signala na ulaz i slušanjem signala na izlazu.

Projektovati visokopropusni FIR filtar reda $N = 200$ sa graničnom frekvencijom 2 kHz. Implementirati ovaj filtar i testirati ga dovođenjem zvučnog signala na ulaz i slušanjem signala na izlazu.

Sadržaj određenog dijela memorije je moguće grafički prikazati pomoću **View** > **Debug Windows** > **Plot** > **New...**. U polje *Address* se unese ime promjenljive, a u polje *Count* broj elemenata vektora koje želimo prikazati. U polju *Data* biramo tip podataka. Kada se ovi podaci postave kliknuti na **Add**, a zatim **OK**.

Postavke grafika je moguće izmijeniti tako što se desnim tasterom miša klikne na grafik i izabere **Configure**. Ukoliko se desnim tasterom miša klikne na grafik i izabere **Modify Settings** moguće je nacrtati i amplitudni spektar podataka tako što se izabere *FFT Magnitude* i u polje *Sample rate (Hz)* unese vrijednost frekvencije odmjeravanja.

2.1 Statističko profilisanje i optimizacija

Statističko profilisanje omogućava analizu koda kako bi se otkrile funkcije i dijelovi koda u kojima se troši najveći procenat vremena pri izvršavanju programa. Takvi dijelovi koda su onda pogodni za optimizaciju. Statističko profilisanje se pokreće pomoću **Tools** > **Statistical Profiling** > **New**. Potom se pokrene izvršavanje programa i, nakon što se statističke vrijednosti ustale, procenti vremena izvršavanja provedenog u pojedinim funkcijama će biti prikazani u vidu histograma.

Cilj optimizacije je generisanje koda koji se izvršava brzo i zauzima malo memorije. Pošto nisu svi načini optimizacije uvijek prikladni, odnosno upotrebljivi, postoje različiti nivoi optimizacije koji se kontrolišu opcijama kompajlera ili pragma direktivama preprocesora. Podrazumijevanim opcijama projekta optimizacija je isključena, a uključena je mogućnost debagovanja programa. Uključivanjem optimizacija mogućnost debagovanja programa se može smanjiti zato što kompajler, ukoliko su istovremeno uključene opcije debagovanja i optimizacije, prednost daje optimizaciji.

Optimizacije se mogu uključiti tako što se izabere meni **Project > Project Options...**, a zatim se u stablu na lijevoj strani prozora izabere **Compile > General** i označi polje **Enable Optimization**. Klizačem je moguće podešavati kompromis između optimizacije brzine izvršavanja i zauzeća memorije.

Statistički profilisati izvršavanje programa kojim je realizovan FIR filtar reda $N = 200$. Uočiti procenat vremena izvršavanja koje se troši u funkciji `process_AD1939_samples`. Uključiti optimizaciju brzine izvršavanja i uočiti smanjenje vremena provedenog u funkciji `process_AD1939_samples`.

Još jedna mogućnost za optimizaciju je korišćenje Harvard arhitekture procesora. Kod ove arhitekture memorija za podatke je odvojena od programske memorije. Oba tipa memorije su odvojenim magistralama povezane sa procesorom, pa je moguće u istom ciklusu pristupiti i jednoj i drugoj memoriji. Ovo se može iskoristiti tako što se, na primjer, koeficijenti filtra smjeste u programsku memoriju, a sadržaj pomjeračkog registra u memoriju za podatke. Tada se u istom ciklusu mogu pročitati i koeficijent filtra i odmjerak iz pomjeračkog registra. Korišćenje memorije za podatke se specificira pomoću ključne riječi `dm`, a programske memorije pomoću ključne riječi `pm`. Podrazumijevano je korišćenje memorije za podatke. Na primjer

```
float dm reg[N];
float pm B[N];
```

Smjestiti koeficijente filtra u programsku memoriju i provjeriti procenat vremena provedenog u funkciji `process_AD1939_samples`.

3 Realizacija IIR filtara

Na Slici 2 prikazana je blok-šema realizacije IIR filtra prvom direktnom formom. Odziv filtra dat je jednačinom

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{l=1}^{N} a_l y(n-l). \tag{2}$$

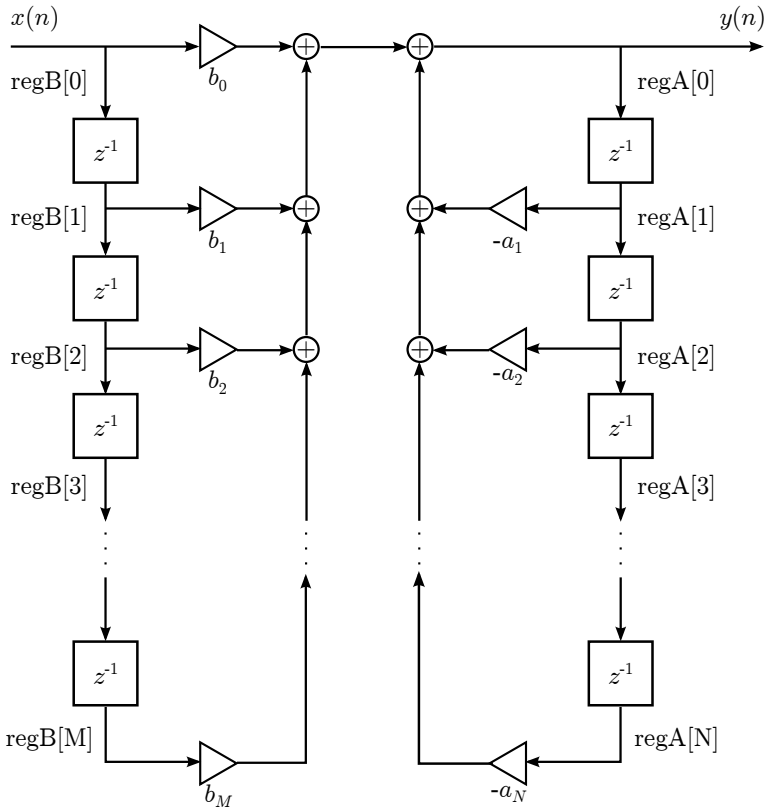
Dakle, kod IIR filtra realizovanog prvom direktnom formom postoje dva pomjeračka registra i dvije sume oblika (1).

Blok šema IIR filtra realizovanog drugom direktnom formom prikazana je na Slici 3.

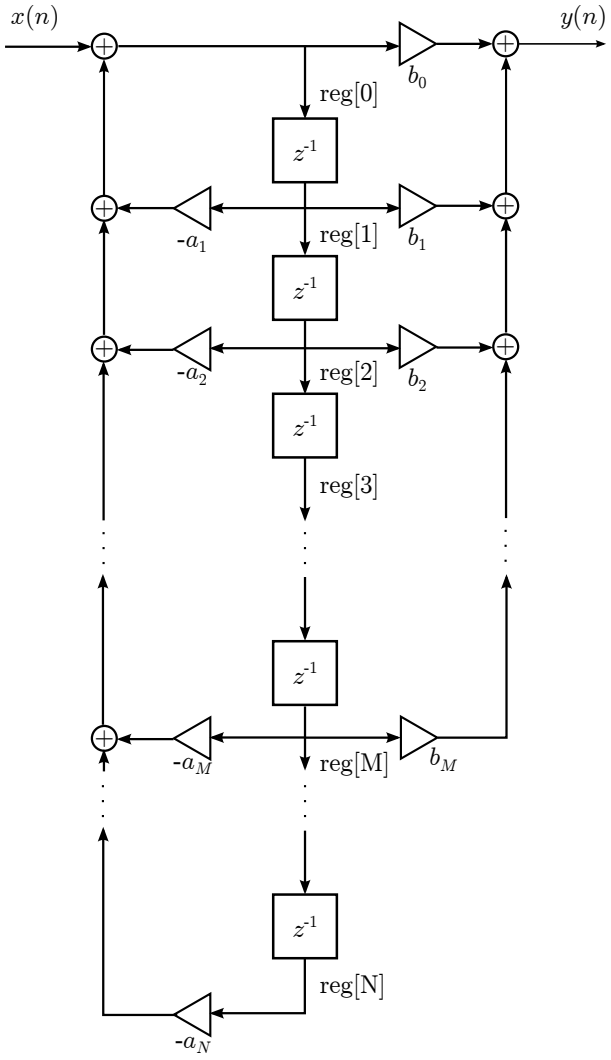
U ovom slučaju postoji samo jedan pomjerački registar, ali se i dalje računaju dvije sume. Smatraćemo da je $M \leq N$.

3.1 Realizacija IIR filtra prvom direktnom formom

Koristeći MATLAB projektovati niskopropusni Batervortov IIR filtar četvrtog reda sa graničnom frekvencijom 1 kHz. Frekvencija odmjeravanja je 48 kHz. Generisati C zaglavlje sa koeficijentima filtra.



Slika 2: Realizacija IIR filtra prvom direktnom formom.



Slika 3: Realizacija IIR filtra drugom direktnom formom.

Kreirati novi projekat u VisualDSP++ okruženju. Kopirati kod za inicijalizaciju iz arhive `kostur.zip` i C zaglavlje koje ste generisali korišćenjem MATLAB-a u folder u kojem je smješten novi projekat. U fajl `process_audio.c` dodati kod kojim se implementira IIR filtar realizovan prvom direktnom formom. Pogledajte u C zaglavlju imena promjenljivih u kojima se nalaze koeficijenti filtra, kao i redovi polinoma u brojiocu i imeniocu funkcije prenosa. Kompajlirati i linkovati projekat, te učitati izvršni program u memoriju DSP-a. Testirati program.

Koristeći MATLAB projektovati niskopropusni Batervortov IIR filtar osmog reda sa graničnom frekvencijom 1 kHz. Generisati C zaglavlje sa koeficijentima filtra. Zamijeniti zaglavlje u prethodnom projektu i testirati novu implementaciju.

Problem koji se javlja posljedica je kvantovanja koeficijenata filtra. Da bi se bolje razumio ovaj problem korisno je prikazati mapu nula i polova projektovanog filtra pomoću naredbe `zplane`. Nakon toga sačuvajte vrijednosti koeficijenata u tekstualnom fajlu, obrišite radni prostor MATLAB-a, a zatim ponovo učitajte vrijednosti koeficijenata

```
save koeficijenti.dat -ascii b a
clear
load koeficijenti.dat
b = koeficijenti(1, :);
a = koeficijenti(2, :);
```

Na ovaj način su koeficijenti sačuvani sa 8 značajnih cifara što približno odgovara tačnosti koju, prema IEEE standardu, obezbjeđuje 32-bitni float tip. Nacrtajte mapu nula i polova koja odgovara ovim koeficijentima i komentarišite razlike u odnosu na prethodni slučaj.

Realizacija IIR filtra drugom direktnom formom je implementaciono slična realizaciji filtra prvom direktnom formom i ima slične nedostatke koji se, prije svega, odnose na moguću pojavu nestabilnosti filtra prouzrokovane kvantovanjem njegovih koeficijenata. Da bi se ovo izbjeglo, IIR filtri se najčešće realizuju pomoću kaskadne veze sekcija drugog reda.

3.2 Realizacija IIR filtra kaskadnom vezom sekcija drugog reda

Polazeći od niskopropusnog Batervortovog filtra osmog reda iz prethodnog odjeljka, pomoću MATLAB funkcije `exportBiquadCoefs` generisati C zaglavlje u kojem se nalaze koeficijenti sekcija drugog reda. Proučiti funkciju `exportBiquadCoefs` i generisano zaglavlje kako biste se upoznali sa načinom čuvanja koeficijenata pojedinih sekcija drugog reda u dvodimenzionalnim nizovima.

Kreirati novi projekat u VisualDSP++ okruženju, kopirati kostur programa i zaglavlje sa koeficijentima filtra. U fajlu `process_audio.c` imple-

mentirati realizaciju IIR filtra kaskadnom vezom sekcija drugog reda. Sekcije drugog reda implementirati drugom direktnom formom. Vodite računa o tome da svaka sekcija drugog reda sada mora imati sopstveni pomjerački registar. Moguća implementacija je da se svi pomjerački registri čuvaju u jednom dvodimenzionalnom nizu, slično kao koeficijenti filtara.

Kod kaskadne veze filtara izlaz iz jednog filtra je ulaz u drugi, sa izuzetkom prvog i poslednjeg filtra u nizu. Prijedlog za implementaciju bi bio da se napiše funkcija koja kao ulazne argumente prima tekući odmjerak i indeks filtra u kaskadi i izračunava odziv tog filtra. Sada je moguće odzive svih filtara izračunati u petlji.

Realizovati niskopropusni Batervortov filtar osmog reda pomoću kaskadne veze sekcija drugog reda. Testirati dobijeni filtar.