SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4242

OBRADA AUDIO SIGNALA NA BEAGLBONE BLACK PLATFORMI

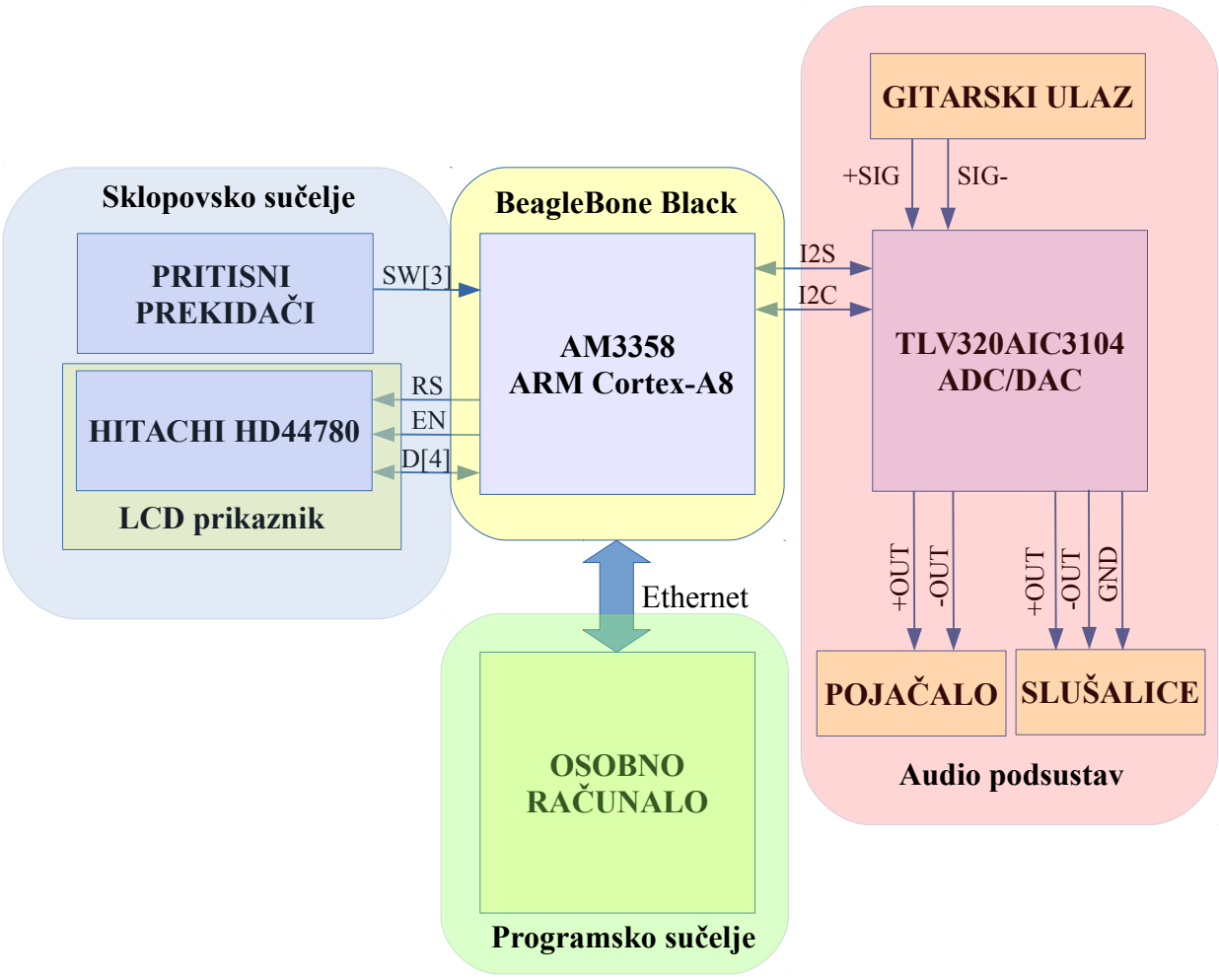
Ivan Pavić

Zagreb, lipanj 2015.

Sadržaj

1. Uvod

Ugradbeni računalni sustavi poput mobitela, dlanovnika i ostalih multimedijskih uređaja imaju sposobnost reprodukcije i snimanja zvuka. Cilj ovog završnog rada je razviti ugradbeni računalni sustav koji bi mogao reproducirati i snimati zvuk te obrađivati zvuk u stvarnom vremenu. Sustav je zasnovan na *BeagleBone Black* platformi. U nastavku je idejna blok shema sustava.



Slika 1: Idejna blok shema sustava

Središnji dio sustava je *BeagleBone Black* koji prikuplja podatke sa audio kodeka, komunicira sa sklopovljem i pokreće *web*-poslužitelj kojem se može pristupiti preko *ethernet* sučelja. Obrada pročitanih uzoraka u stvarnom vremenu i komunikacije sa sklopovljem

mogu se ostvariti kroz operacijski sustav *Linux* bez korištenja *real time* jezgre operacijskog sustava ili programabilnih jedinica za rad u stvarnom vremenu. Programska i hardverska podrška za *BeagleBone Black* razvijena sklopu ovog završnog rada namijenjena je i testirana na *Debian Linux* distribuciji[1]. Dodatno sklopovlje koje nije dio *BeagleBone Black* platforme je realizirano kao modularno proširenje koje se može priključiti na bilo koju *BeagleBone Black* pločicu.

Namjena ovakvog sustava je obrada signala sa zvučnih osjetila malih signalnih razina(dinamički mikrofona sa zavojnicom ili električne gitare). Sustav bi omogućavao prikupljanje, obradu i reprodukciju u stvarnom vremenu. Primjena mu nije ograničena i može služiti za reprodukciju glazbe ili snimanje zvuka. Preko *ethernet* sučelja kranjem korisniku su dostupni interaktivni alati za obradu zvuka. Pomicanjem klizača ili uključivanjem ili isključivanjem pojedinih filtara oblikuje se signal u frekvencijskoj ili vremenskoj domeni . Korisnik na taj način može mijenjati funkcije sklopki i osigurati da se uključenjem ili isključenjem određene sklopke mijenja algoritam obrade zvuka, a samim time i zvuk. Drugim riječima interaktivno *web* sučelje korisniku omogućuje da odredi koji će algoritam biti pokrenut i što će se ispisati na LCD-u kad se pritisne određeni prekidač.

Osim asimetričnog niskoomskog izlaza za slušalice ili linijske zvučnike ima i diferencijalne visokoomske izlaze čiji je izlazni otopor veći i namijenjen je za gitarska ili mikrofonska pojačala ulaznog otpora nekoliko desetaka kΩ.

Obrada zvuka vrši se pomoću FIR i IIR filtara te bikvadratnih filtara koji su implementirani u vremenskoj domeni kao diferencijske jednadžbe. Zvuk se i analizira u frekvencijskoj domeni i pritom se određuje dominantna frekvencija u spektru zvuka što je korisno prilikom ugađanja glazbenih instrumenata.

1 Točnije, namijenjen je inačicama jezgre operacijskog sustava(engl. *kernel*) 3.8 ili novijim.

2. Zvukovne mogućnosti i posebnosti BeagleBone Black platforme

BeagleBone Black platforma zasnovana je na aplikativnom procesoru Sitara AM3358 ARM Cortex-A8. AM3358 za potrebe reprodukcije i snimanja i zvuka ima 2 višekanalna serijska zvučna priključka[2]. McASP moduli omogućuju komunikaciju *Inter-Integrated Sound*(I2S), *Time-Divison Multiplex*(TDM), *Digital Audio Interface*(DIT) protokolima. Nadalje, McASP moduli sastavljeni su od prijamne i odašiljačke strane. Prijamna i odašiljačka strana modula mogu raditi potpuno odvojeno, na različitim frekvencijama s različitim veličinama riječi, ali mogu biti i sinkronizirani. Kod nekih audio kodeka uzorci dolaze do kodeka u I2S *left justified* formatu, a moraju se slati u I2S *right justified* formatu. U tom slučaju nezavisnost signala takta između prijamne i odašiljačke strane iznimno je korisna. Za potrebe sustava koristi se I2S sabirnica.

2.1 Konfiguracija McASP modula putem device tree overlay-a

Da bi komunikacija između audio kodeka i procesora bila moguća potrebno je McASP modul ispravno konfigurirati i inicijalizirati. Odnedavno se na *BeagleBone Black* platformi i još nekim *embedded Linux* platformama baziranim na ARM aplikativnim procesorima za mijenjanje funkcije određenog dijela sklopovlja, odnosno multipleksiranje linija procesora koriste *device tree overlay-i*. *Device Tree* je podatkovna struktura namijenjena opisu sklopovlja. Ugrađivanje u kod detalja o sklopovlju u operacijski sustav je nepregledno, a mala izmjena sklopovlja ili koda upravljačkog programa može stvoriti puno problema u radu operacijskog sustava. Stoga se detalji o sklopovlju zapisuju u posebnu podatkovnu strukturu koju se prevodi u binarnu datoteku koju *bootloader* predaje jezgri operacijskog sustava prilikom podizanja sustava(engl. *boot time*). Podatkovna struktura *Device Tree* je sastavljena od čvorova i svojstava. Čvorovi mogu sadržavati podčvor i svojstva, a svojstva su parovi vrijednosti ime, ključ. Za prijevod *Device Tree* izvornog koda u binarnu datoteku koristi se *Device Tree Compiler*. Rezultat prevođenja je *Device Tree Overlay* datoteka koju operacijski sustav može učitati prilikom podizanja, ali pritom treba voditi računa da se datoteka nalazi na mjestu u memoriji na kojoj ju *bootloader* može pronaći.

2 Višekanalni serijski zvučni priključak(engl. *Multichannel Audio Serial Port*) je razvila tvrtka *Texas Instruments* za potrebe digitalne obrade signala. Dalje u tekstu koristi se kratica McASP.

Da bi operacijski sustav prihvatio *Device Tree Overlay, Device Tree Source* mora biti napisan u obliku koji je predviđen za određeni uređaj. Predlošci(engl. *Bindings)* za *Device Tree Source* definiraju oblik *Device Tree* izvornog koda za veliki broj uređaja i većina ih je dobro dokumentirana. U konkretnoj *Debian* distribuciji predlošci za Device Tree izvorni kod za sklopovlje kompatibilno s *BeagleBone Black* platformom mogu se naći u kazalu /opt/source/dtb-3.14-ti/Bindings/. Na *BeagleBone Black* platformi osim prilikom podizanja operacijskog sustava *Device Tree Overlay* može se učitati i u *runtime-u*. To omogućava poseban upravljački program *Cape Manager*. Nažalost, isti overlay ne može se ukloniti na isti način već je potrebno uređaj resetirati. Ako se ipak *overlay* pokuša ukloniti to rezultira panikom u jezgri operacijskog sustava i uređaj u većini slučajeva ne reagira.

Na temelju predloška može se napisati *Device Tree* fragment za McASP modul. McASP modul potrebno je konfigurirati za rad u I2S modu. Osim McASP modula potrebno je konfigurirati i upravljački program za kodek. To obuhvaća podešavanje parametara I2C i I2S sabirnice. U nastavku je prikazan fragment za konfiguraciju McASP modula.

Placholder: Device Tree Fragment

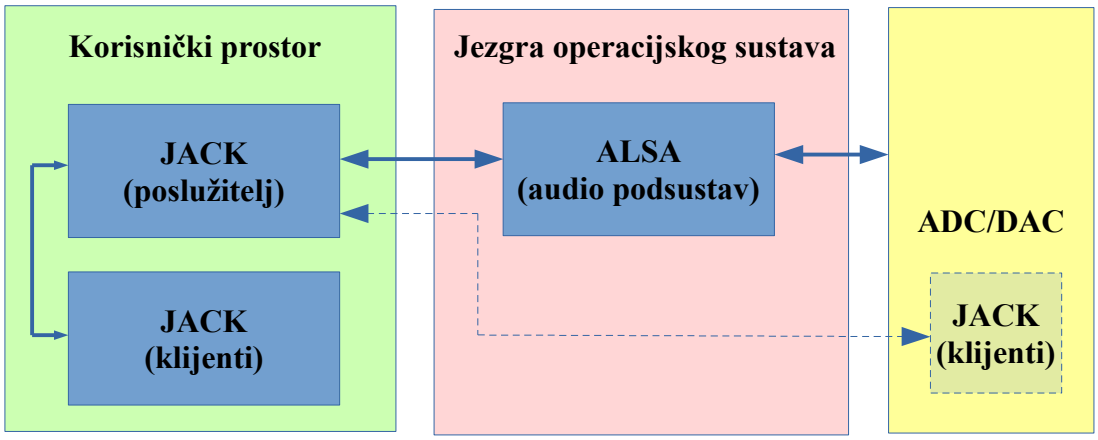
Ključno svojstvo u fragmentu je *"compatible"* jer će konkretne promjene biti moguće samo na sklopovlju koje je tu navedno i čiji upravljački programi podržavaju određeni *DTS* fragment odnosno način opisa sklopa.

3. Zvukovne mogućnosti Debian distribucije

Obrada zvučnih signala u *Linux* operacijskim sustavima može se raditi pomoću biblioteka otvorenog koda baziranim na ALSA(engl. *Advanced Linux Sound Architecture)* upravljačkim programima. ALSA sadrži biblioteku koja omogućuje interakciju aplikacija u korisničkom prostoru sa jezgrom operacijskog sustava. Jedan od API-a temeljen na ALSA upravljačkim programima je JACK(engl. *Jack Audio Connection Kit)*. JACK omogućuje apstrakciju visoke razine u razvoju aplikacija za obradu zvučnih signala. To znači da se prilikom razvoja aplikacija komunikacija između audio kodeka i procesora vrši neovisno o aplikaciji. Aplikacija obrađuje uzorke koje joj u određenom trenutku isporuči jezgra operacijskog sustava. Uzorci se pritom nalaze na dijeljenim memorijskim lokacijama.

3.1. Primjer JACK programa za obradu zvuka

JACK program sastoji se od dva ili više odvojenih procesa. Jedan od procesa je uvijek JACK audio server, a ostali procesi su klijenti. Svaki klijentski proces ima registriran ulazni i izlazni priključak odnosno memorijske lokacije na kojima se nalaze uzorci zvuka. Komunikacija između poslužitelja i klijenata je *event-driven*, a podaci su klijentima dostupni u kratkom vremenu koje određuje server ovisno brzini uzoraka odnosno frekvenciji uzorkovanja. S klijentske strane ne postoje ni upravljački programi ni stvarno sklopovlje koje je izvor podataka već postoji samo server koji omogućuje komunikaciju s drugim klijentima. Gledano sa strane poslužitelja osim klijenata stvorenih pokretanjem aplikacije, klijenti su i svi hardverski ulazi i izlazi, a vidljivi su zahvaljujući ALSA upravljačkim programima.



Slika 2: Arhitektura sustava koji koristi JACK

Ako klijentski proces ne uspije na vrijeme obraditi podatke u jednom ciklusu, poslužitelj će javiti pogrešku. Stoga je bitno da se podaci na klijentskoj strani obrađuju što brže kako ne bi došlo do gubitaka uzoraka.

Najjednostavniji klijentski program bi podatke kopirao sa svog ulaznog priključka na izlaz. Kad server proziva pojedinog klijenta ustvari poziva funkciju čija je adresa predana kao argument prilikom incijalizacije klijenta i registriranja priključaka. Primjer takve funkcije dan je u nastavku.

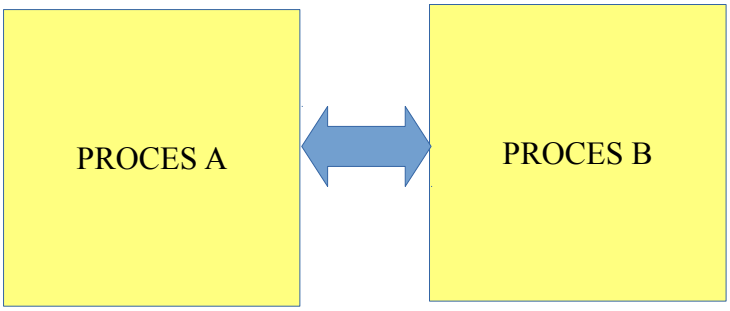
Placeholder: pseudokod ili snippet funkcije

Prije kopiranja podataka na izlaz mogu se nad ulaznim podacima izvršiti različiti algoritmi za obradu podataka ili se podaci mogu jednostavno kopirati na drugu memorijsku lokaciju na kojoj će ih neki drugi proces ili dretva analizirati.

4. Analiza programske podrške

4.1. Osnovna struktura programske podrške

Programska podrška za sustav digitalne obrade zvuka sastavljena je od dva programa odnosno procesa. Prvi proces(dalje u tekstu proces A) komunicira sa sklopovskim sučeljem, prikuplja uzorke i obrađuje ih. Drugi proces(dalje u tekstu proces B) je poslužitelj koji krajnjem korisniku omogućuje pristup kontroli parametara algoritama obrade zvuka preko ethernet sučelja.



Treba dovršiti sliku!

Zadaće procesa A su slijedeće:

- upravljanje LCD prikaznikom;
- akvizicija uzoraka signala i njihova obrada;
- provjeravanje stanje sklopki;
- isporuka podataka procesu B ukoliko proces B zatraži podatke;
- i mijenjanje parametara obrade signala na zahtjev procesa B.

Zadaće procesa B su slijedeće:

- omogućava prikaz interaktivnog sučelja koje korisniku omogućava upravljanje procesom A;
- upravlja promjenom razine kontrasta i pozadinskog osvijetljenja na zahtjev korisnika;
- drži stalno otvorenu vezu prema korisniku;
- i šalje odgovarajuće poruke procesu B.

4.2. Struktura procesa za obradu signala

Princip rada osnovnog procesa za obradu signala opisan je u poglavlju x.x.x. U konkretnoj implementaciji između primanja i slanja uzoraka s ulaza na izlaz uzorci se obrađuju. Algoritmi za obradu signala opisani su u poglavlju 5.

Za upravljanje LCD prikaznikom koristi se biblioteka razvijena za upravljanje GPIO izlazima *BeagleBone Black* platforme. Upravljanje GPIO izlazima vrši se iz korisničkog prostora po protokolu koji je predviđen za komunikaciju s LCD prikaznikom. Provjeravanja stanja sklopke vrši jedna dretva koja čita vrijednost na GPIO ulazu u određenom kratkom vremenskom intervalu. Promjena vrijednosti drugim dretvama vidljiva je kroz dijeljenu memoriju. Ovakvo praćenje stanja GPIO ulaza nije efikasno, a za optimizaciju procesa potrebno je napisati upravljački program koji bi proces obavijestio o promjeni na GPIO ulazu. Ipak, u prvoj iteraciji sustav će raditi ispravno.

4.3. Struktura procesa programskog sučelja

Sučelje za upravljanje procesom čini HTTP poslužitelj ostvaren na NodeJs platformi. Jednostavno mu se pristupa preko *ethernet* sučelja. Kako bi sve promjene između klijentske i poslužiteljske strane bile trenutne, koristi se *socket.io* modul koji omogućava održavanje stalne veze i nakon što je HTTP zahtjev završen. Klijentsku stranu čine HTML kod i JavaScript uz korištenje jQuery API-a. HTTP modul sastavni je dio NodeJs-a. Ipak, socket.io potrebno je dodatno instalirati.

Sučelje se sastoji od klizača koji mijenjaju parametre za obradu signala. Pomak klizača uzrokuje poziv funkcije na poslužiteljskoj strani koja šalje poruku procesu za obradu audio signala. Sučelje omogućava korsniku da promijeni razinu pozadinskog osvijetljenja i kontrast simbola LCD prikaznika pomoću klizača.

4.4. Međuprocesna komunikacija između procesa

Međuprocesna komunikacija ostvarena je korištenjem UNIX *domain socket* protokola. Korištenje UNIX IPC priključnih točki slično je korištenju klasičnih TCP ili UDP priključnih točaka u mrežnoj komunikaciji. Ipak, komunikacija IPC priključcima puno je brža jer se odvija u sloju jezgre operacijskog sustava. Na UNIX platformama ugrađena je biblioteka koja omogućava rad s IPC priključcima. Oba procesa su u ravnopravnom položaju u odnosu na protok informacija odnosno nijedan proces nije isključivo poslužitelj.

Format poruka u međuprocesnoj komunikaciji je JSON. U poruci se tipično prenose dva para ključ vrijednost. Prva vrijednost prenosi informaciju o strukturi podataka koju treba promijeniti, a druga novu vrijednost nekog elementa strukture podataka. Struktura podataka u procesu A uglavnom su filtri.

5. Obrada zvuka

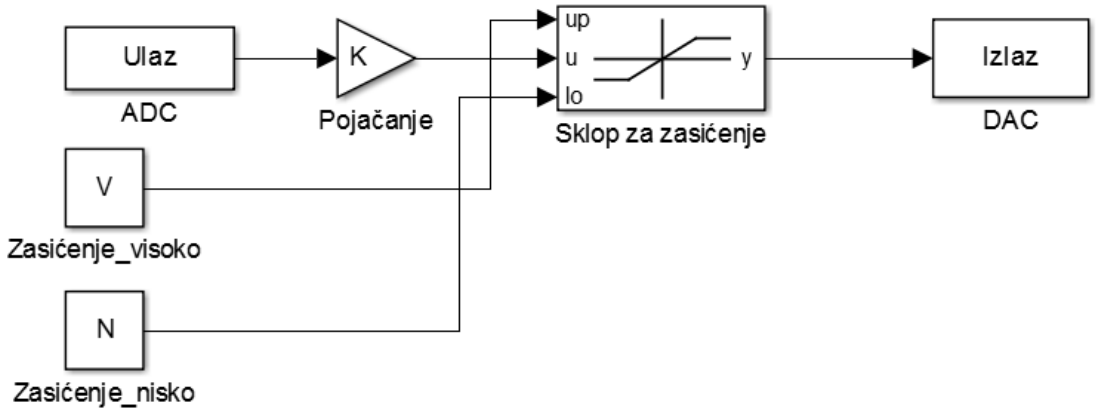
Obrada zvuka vrši se FIR i IIR filtrima. Dvije su skupine implementiranih filtara:

- filtri za oblikovanje zvučnog signala u vremenskoj domeni;
- filtri za oblikovanje frekvencijske karkateristike zvučnog signala.

Iako su promjene signala uvijek vidljive u frekvencijskoj i vremenskoj domeni, kod prve skupine filtara cilj je oblikovati signal u vremenskoj domeni. Primjerice, ograničiti amplitudu signala. Druga skupina filtara je očito frekvencijski selektivna i prigušit će ili pojačati samo pojedine kompnente u spektru signala. To uzrokuje pormjenu u vremenskoj domeni koja nije intuitivna ako je na ulazu kompozitni zvučni signal. Ipak, prva i druga skupina filtara implementirane su vremenskoj domeni.

5.1. Obrada zvuka u vremenskoj domeni

Sustav ima implementirane filtre za distorziju, reverbraciju i *chorus*. Filtar za distorziju zvučnog signala najjednostavniji je. Prijenosna karakteristika takvog sustava jednaka je karakteristici realnog operacijskog pojačala s bipolarnim napajanjem. Drugim riječima, sustav uzorke ulaznog zvučnog signala množi konstantom, a zatim ih ako njihova vrijednosti prelazi prag zasićenja ograničava na nekoj konstantnoj vrijednosti.



Slika 3: Blokovski model distorzije

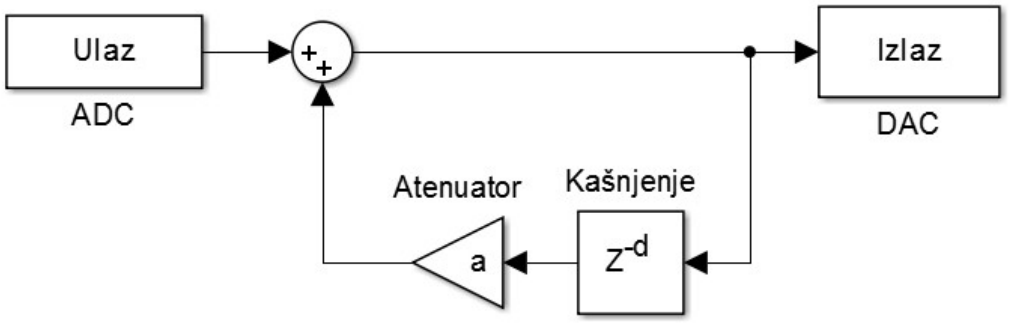
Filtar za reverbraciju jednostavan je IIR filtar. Diferencijska jednadžba filtra je:

$$y[n]=u[n]+ay[n-D]$$

Prijenosna funkcija takvog filtra je:

$$H(Z)=\frac{1}{1-az^{-D}}$$

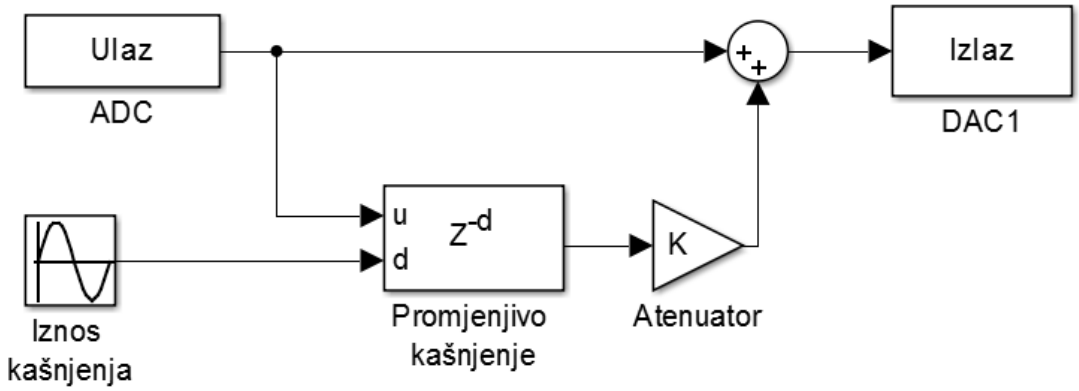
Simulink model filtra je:



Slika 4: Blokovski model filtra za reverbraciju

Za određene vrijednosti konstanti a i D sustav za reverbraciju može biti nestabilan što rezultira nekontroliranim porastom izlaza pa se vrijednosti a i D moraju postaviti u one granice u kojima korisnik neće moći sustav dovesti u nestabilno stanje. Dovoljno je postaviti uvjete na polove na prijenosne funkcije.

Chorus filtar stvara specifični zvučni efekt. To je FIR filtar koji na izlaz superponira uzorke ulaznog signala čije kašnjenje varijabilno, ali se mijenja unutar ograničenih vrijednosti. Superponirani uzorci su atenuirani.



Slika 7: Blokovski dijagram *chorus* filtra

5.2. Obrada frekvencijskog spektra zvuka

Diferencijska jednadžba za filtar koji bi atenuirao zvučni signal na određenoj frekvenciji ne može se napisati jednostavno kao jednadžba diferencija sustava za dodavanje odjeka. Koeficijenti takvog filtra nemaju izraženo fizikalno značenje kao kod sustava za reverbraciju. Diferencijska jednadžba takvih filtara može se dobiti iz kontinuirane prijenosne funkcije njihovog analognog prototipa. Iz kontinuirane prijenosne funkcije u s-domeni može se dobiti prijenosna funkcija u z-domeni. Pritom treba voditi računa o odabiru postupka diskretizacije jer svaki postupak diskretizacije zadržava različita svojstva početnog analognog sustava. Pošto se želi očuvati frekvencijska karakteristika analognog sustava za diskretizaciju koristi se bilinearna transformacija. Preciznije, koristi se modifcirana bilinerana transformacija ili Tustinov postupak zbog pojave promjene frekvencijskog mjerila kod bilinearne transformacije. Analogni prototipovi filtara su drugog reda zbog stabilnosti, a drugim redom može se izvesti većina prijenosnih funkcija od interesa za obradu zvuka. Prijenosne funkcije digitalnih filtara koji se dobiju u najopćenitijem slučaju u brojniku i nazivniku imaju polinom drugog reda. Takvi filtri se nazivaju bikvadratni filtri. U nastavku je razrađen izvod prijenosne funkcije bikvadratnih filtara pomoću bilinearne transformacije.

5.2.1. Izvod prijenosne funkcije bikvadratnih filtara

Osnovna bilinearna transformacija komplesnoj pseudofrekvenciji pridružuje:

$$w = \frac{1 - z^{-1}}{1 + z^{-1}}$$

Ako se kompleksna varijabla z supstituira komplesnom eksponencijalom, uz sređivanje korištenjem Eulerove relacije, za komplesnu pseudofrekvenciju dobiva se:

$$w = j \tan\left(\frac{\omega T_s}{2}\right)$$

Radi promjene frekvencijskog mjerila uvodi se nova kompleskna frekvencija:

$$\Omega = \frac{2}{T} w$$

$$\Omega = j\omega'$$

Ova supstitucija se uvodi jer će za dovoljni mali period uzorkovanja vrijediti da je pseudofrekvencija jednaka realnoj frekvenciji. Za kompleksnu frekvenciju u s domeni može se napisati:

$$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$$

Normalizacijom pseudofrekvencije i korištenjem prethodnih relacija proizlazi:

$$s = \frac{1}{\tan\left(\frac{\omega T_s}{2}\right)}\frac{1-z^{-1}}{1+z^{-1}}$$

Na ovaj način ostvarena je veza između kontinurane i diskretne domene. Potrebno je izračunati preostale potencije kompleksne frekvencije:

$$s^0 = \left(1+2z^{-1}+z^{-2}\right)\left(1-\cos(\xi)\right)$$

$$s^1 = \left(1-z^{-2}\right)\sin(\xi)$$

$$s^2 = 2\left(1-2\cos(\xi)z^{-1}+z^{-2}\right) \quad ,$$

gdje je:

$$\xi = \frac{2\pi f_0}{f_s} \quad ,$$

pri čemu je f_0 frekvencija na kojoj filtar dominantno djeluje, a f_s frekvencija uzimanja uzoraka. Pomoću ovih izraza može se opisati bilo koji filtar drugog reda i prebaciti u z domenu. Promotri li se oblik supstitucije lako se da zaključiti da će opći oblik prijenosne funkcije u z domeni odnosno prijenosne funkcije bikvadratnog filtra biti[3]:

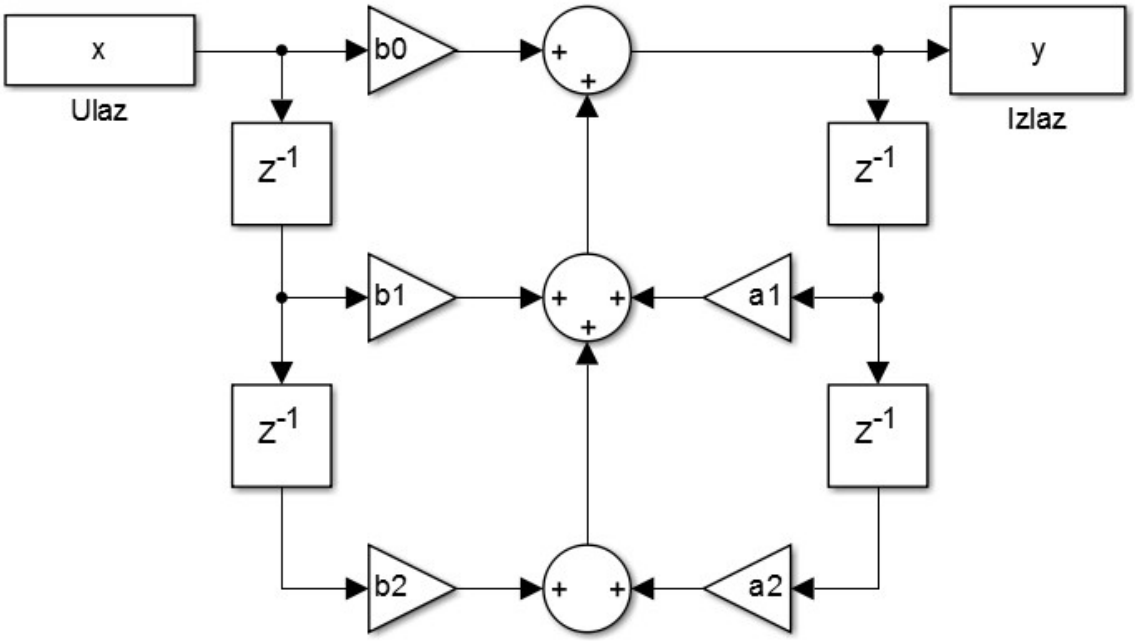
$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

3 U nekim varijantama prijenosna funkcija sadrži i šesti koeficijent a_0, ali to u konačnici nema nikakvog utjecaja.

Iz prijenosne funkcije u z domeni lako se dobije konačni izraz za diferencijsku jednadžbu:

$$y[n]=b_0 x[n]-b_1 x[n-1]-b_2 x[n-2] \\ -a_1 y[n-1]-a_2 y[n-2]$$

Najlakši način za implementaciju diferencijske jednadžbe je direktna I realizacija.



Slika 5: Direktna I realizacija bikvadratnog filtra

Na ovaj način pomoću jednog algoritma mijenjajući koeficijente može implementirati bilo koju vrstu filtra. Vrste filtara koje su potrebne za realizaciju frekvencijskoj ujednačivača obrađen se u nastavku.

5.2.2. Filtri za realizaciju frekvencijskog ujednačivača

Pošto sustav za realizaciju filtara nije ograničen na specifične oblike kontinuiranih prijenosnih funkcija mogu se ostvariti različita filtriranja. Najčešći filtri su niskopropusni, visokopropusni i pojasno-propusni. Za realizaciju frekvencijskog ujednačivača sustav koristi kaskadu pojasno propusnih *peakingEQ* filtara. Kaskada je ostvarena uzastopnim pozivanjem algoritma nad istim uzorcima. Analogna prijenosna funkcijia *peakingEQ* filtra je:

$$H(s)=\frac{s^2+\frac{A}{Q}s+1}{s^2+\frac{1}{AQ}s+1} \quad ,$$

Uvrštavanjem supstitucije iz prethodnog poglavlja mogu se izračunati koeficijenti odgovarajućeg bikvadratnog filtra.

Tipična frekvencijska karakteristika *peakingEQ* filtra izdiže amplitudnu karakteristiku oko centralne frekvencije za određeno pojačanje.

Placeholder: amplitudno frekvencijska karakteristika peakingEQ filtra

Kaskadom više filtara ostvaruje je se mogućnost oblikovanja cijelog spektra oko središnjih frekvencija bikvadratnih filtara.

Placeholder: amplitudno frekvencijska karakteristika kaskade

5.3. Analiza frekvencijskog spektra zvuka

Jedan od početnih zahtjeva sustava je i mogućnost ugađanja glazbenih instrumenata. To se može ostvariti analizom frekvencijskog spektra. Koeficijenti frekvencijskog spektra mogu se izračunati iz uzoraka ulaznog zvučnog signala diskretnom Fourirerovom transformacijom. Diskretna Fourireova transformacija uzoraka računa se algortimom brze Fourierove transformacije. Frekvencijska razlučivost je pritom proporcionalna broju uzoraka jer je frekvencijska udaljenost između koeficijenata u brzoj Fourierovoj transformaciji:

$$\Delta f=\frac{fs}{N} \quad ,$$

gdje je *fs* frekvencija uzorkovanja, a N početni broj uzoraka.

Nakon izračuna koeficijenata lako se odredi spektar snage iz kojeg se može izraču-

nati dominatna frekvencija u spektru. Podaci o dominantnoj frekvenciji mogu se prikazati na odgovarajućem sučelju. Za optimizaciju algoritma koristi se činjenica da su uzorci zvuka realni niz. Nadalje, potrebno je uzeti u obzir da će spektar sadržavati koeficijente koji nemaju fizikalno značenje. To su koeficijenti čija je frekvencija veća od Nyquistove frekvencije odnosno u obzir se uzimaju prvih N/2 koeficijenata.

6. Audio podsustav

6.1. Posebnosti audio kodeka TLV320AIC3104

TL320AIC3104 je *low-power* audio kodek. Uglavnom se primjenjuje u ugradbenim računalnim sustavima. Ima ugrađeno stereo pojačalo za slušalice i više ulaza i izlaza koji su programabilni i mogu biti asimetrični ili diferencijalni. Kodek s mikroprocesorom komunicira preko više sučelja. Kodek je kontroliran preko I2C sučelja. U registrima koji su vidljivi putem I2C sučelja postavlja se brzina uzorkovanja ADP i DAP-a, širina riječi i audio protokol. Drugo sučelje je sabirnica za prijenos audio podataka koje se može konfigurirati za rad u I2S, I2S Left-Justified, I2S Right-Justified modu. U konkretnom sustavu koristi se I2S sabirnica, a na taj način je konfiguriran McASP modul na procesorskoj strani. Iako ima ugrađeno mikrofonsko predpojačalo ono se koristi za kondezatorske mikrofone kojima je potreban napon napajanja(bias voltage). Zahtjevi sustava su na induktivna osjetila(dinamički mikrofon, električna gitara) koji ne trebaju napajanje, ali su naponske razine signala puno niže od kapacitivnih osjetila. Stoga se koristi jedan diferencijalni ulaz. Na taj način nije potrebno dodatno sklopovlje koje bi prilagodilo razinu signala za asimetrične ulaze kodeka.

6.2. Karakteristike ulaznog signala

Karaktersitike ulaznog signala prilagođene su ulazu kodeka. Ulazna impendancija diferencijalnog ulaza kodeka može se podešavati u određenom rasponu vrijednosti.

Placholder: nastavak