SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3780

ODREĐIVANJE OSNOVNOG TEMPA MUZIKE POMOĆU PROCESORA ZA DIGITALNU OBRADU SIGNALA

Emanuel Zajc

Zagreb, srpanj 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 13. ožujka 2014.

ZAVRŠNI ZADATAK br. 3780

Pristupnik: **Emanuel Zajc (0036400628)**
Studij: Računarstvo
Modul: Obradba informacija

Zadatak: **Određivanje osnovnog tempa muzike pomoću procesora za digitalnu obradbu signala**

Opis zadatka:

U okviru završnog rada potrebno je istražiti postupke za određivanja osnovnog tempa muzičkog zapisa primjenom postupaka digitalne obradbe signala. Muzički tempo odrediti u jedinici muzičkih udara u minuti (engl. Beats Per Minute, BPM). Postupak određivanja tempa implementirati u programskoj okolini Matlab i testirati na unaprijed snimljenim muzičkim zapisima. Algoritam realizirati i u programskom jeziku C u frakcionalnoj aritmetici i usporediti s referentnim modelom u Matlabu. U konačnici, algoritam prilagoditi za rad u stvarnom vremenu i demonstrirati rad na razvojnom sustavu za digitalnu obradbu signala TMS320VC5505 eZdsp firme Texas Instruments. Testirati rad sustava i njegovu točnost i robusnost na većem broju glazbenih primjera. Za dodatne informacije obratiti se mentoru.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 13. lipnja 2014.

Mentor:
Prof.dr.sc. Davor Petrinović

Djelovođa:
Doc.dr.sc. Marko Subašić

Predsjednik odbora za završni rad modula:
Prof.dr.sc. Sven Lončarić

Sažetak

U radu je predstavljen mogući algoritam za određivanje tempa glazbenog zapisa dekompozicijom na skup frekvencijskih područja, pronalaženjem ovojnice signala pojedinih područja te pronalaženjem vrhova koji nose informaciju o tempu autokorelacijskom funkcijom. Algoritam je modeliran u programskom okruženju Matlab i implementiran na sustavu za digitalnu obradu signala TMS320VC5505 uz modifikacije za rad u stvarnom vremenu. Algoritam je testiran na većem broju glazbenih zapisa.

Sadržaj

Popis oznaka i kratica

BPM (*engl. beats per minute*) - udaraca u minuti
MIDI (*engl. Musical Instrument Digital Interface*) - digitalno sučelje za glazbene instrumente
CD (*engl. compact disk*) - kompaktni disk
ERB (*engl. equivalent rectangular bandwidth*) - ekvivalentna pravokutna propusnost
USB (*engl. universal serial bus*) - univerzalna serijska sabirnica
DMA (*engl. direct memory access*) - memorija sa direktnim pristupom
MAC (*engl. multiplier-accumulator*) - množitelj-zbrajatelj
SAR ADC (*engl. successive aproximation analogue digital converter*) - analogno digitalni pretvornik uz sljedeću aproksimaciju
I/O (*engl. in/out, input/output*) - ulaz/izlaz
LCD (*engl. liquid crystal display*) - zaslon sa tekućim kristalima
IDE (*engl. integrated development enviroment*) - okolina za integrirani razvoj
MMC/SD (*engl. multimedia card/secure digital*) - memorijska kartica
PLL (*engl. phase-locked loop*) - fazno zaključana petlja
LDO (*engl. low drouput*) - nizak pad (napona)
DSP (*engl. digital signal processor /processing*) - digitalni procesor signala/ digitalna obradba signala
CPU (*engl. central processing unit*) - središnja jedinica za obradu, procesor
I2C (*engl. inter integrated circuit*) - inter-integrirani krug
FFT (*engl. fast Fourier transform*) - brza Fourierova transformacija
IFFT (*engl. inverse fast Fourier transform*) - inverzna brza Fourierova transformacija
HWAFFT (*engl. hardware accelerated fast Fourier transform*) - hardverski ubrzana brza Fourierova transformacija
ISA (*engl. instruction set architecture*) - arhitektura skupa instrukcija

Uvod

Mjera i dinamika su osnovne mjerljive značajke glazbenog zapisa. Dinamika označava glasnost izvođenja, a mjera propisuje koliko se nota nalazi u jednom taktu i koja nota traje jednu dobu. Tempo označava brzinu izmjene doba te je mjerljiva veličina koja daje direktnu informaciju o opaženoj brzini glazbenog zapisa.

Prve oznake tempa na notnim zapisima bile su u vidu pridjeva, najčešće na talijanskom jeziku, koji je označavao brzinu kojom se izvodi skladba poput *lento* (sporo), *moderato* (srednje brzo) ili *allegro* (brzo). Izumom metronoma počinje se tempo označavati brojem udaraca u minuti, skraćeno **bpm** i u notnom zapisu označava koliko se doba odsvira u jednoj minuti.

Određivanje tempa nekog glazbenog zapisa većini ljudi dolazi prirodno, "od uha". Većina će slušatelja bez većih problema pratiti glazbu uz tapkanje ili pljeskanje te time procijeniti tempo. Ukoliko se tapkanje upari sa aplikacijom koja broji razmak između pojedinih udara možemo lako dobiti procijenjenu vrijednost udaraca u minuti za glazbeni zapis (**bpm**).

Računalno određivanje tempa glazbenog zapisa korisno je u mnogo primjena poput automatske transkripcije u notni zapis, pretraživanje veće baze glazbenih zapisa prema tempu, automatsko generiranje lista za reprodukciju, automatiziranje sekvenciranje glazbenih zapisa, definiranje granica operacija prilikom editiranja (npr. rezanje, kopiranje, lijepljenje), usklađivanje vizualnih animacija sa glazbenim zapisom i druge.

Prvi algoritmi oslanjali su se na simbolički zapis MIDI formata gdje je nastup nota lako dostupan algoritmu. Današnji pristupi rade sa CD i sličnim digitalnim audio zapisima i koriste razne metode digitalne obrade signala kako bi odredile tempo glazbenog zapisa.

Klasični pristup problemu sastoji se od određivanja trenutka nastupa pojedinih nota koje karakteriziraju dobe ritma poput skupa filtara na frekvencijskim podpodručjima ili upotrebom Fourierove transformacije. Drugi dio algoritma je određivanje periodičnosti

nastupa doba bilo korištenjem češlja filtara nastojeći procijeniti najizgledniju vrijednost, analiziranje histograma intervala između nastupa nota ili vjerojatnosnih modela za izražavanje vjerojatnosti položaja nastupa doba.

U ovom radu pristupit ćemo problemu promatranjem ovojnice audio signala dobivene ispravljanjem i izglađivanjem sinusoida glazbenog zapisa. Prije toga glazbeni zapis provući ćemo kroz skup filtara na različitim frekvencijskim područjima. Za detekciju periodičnosti u signalu koristit ćemo autokorelacijsku funkciju i pretpostaviti da vrhovi sa najvećom amplitudom odgovaraju tempu promatranog glazbenog zapisa. Opisani algoritam implementirat ćemo na razvojnom sustavu za digitalnu obradu signala TMS320VC5505 eZdsp firme Texas Instruments. Uz manje modifikacije algoritam će biti moguće koristiti za dobivanje informacije o tempu glazbenog zapisa u realnom vremenu. Konačno će sustav biti testiran na većem broju glazbenih zapisa raznog žanra kako bi dobili ideju o robusnosti implementiranog rješenja. Zadnji dio rada opisuje kvalitetu i robusnost implementacije i daje ideje o mogućoj praktičnoj primjeni i mogućnosti poboljšanja algoritma.

Upotrijebljeni algoritam

Modelirat ćemo algoritam za rad sa testnim primjercima za koje pretpostavljamo da su konstantnog ritma čije vrijednosti su između 60 do 220 bpm, raspon u koji ulazi većina glazbenih sadržaja.

Prvi dio posla odnosi se na izdvajanje djela informacija iz glazbenog zapisa koji sadrži najviše informacija o ritmu, odnosno tempu. Iz iskustva možemo reći da se ritam nalazi u području nižih frekvencija kojim dominiraju udaraljke i bas sekcija izvođača. Primjenom prikladnog filtra na glazbenom zapisu možemo izdojiti frekvencijske dijelove koji odgovaraju navedenim značajkama i provoditi daljnju analizu na njima smanjujući utjecaj frekvencijskih područja koji ne pridonose informaciji o tempu glazbenog zapisa ili čak mogu značajno utjecati na dobivanje krivog rezultata.

Ograničavanje na jedan nisko pojasni filtar kojim ćemo sačuvati samo niske frekvencije će u velikom broju slučaja dati dobre rezultate no problem nastaje pri obradi glazbenih zapisa kod kojih izostaje pratnja udaraljka ili bass linija poput solo izvedbe glazbenog instrumenta.

Rješenje koje se nameće je upotreba niza filtara kojima ćemo podijeliti frekvencijski spektar na područja (kanale) nad kojima ćemo provoditi daljnju analizu te u konačnici promatranjem svih kanala moći zaključiti o najtočnijoj vrijednosti tempa glazbenog zapisa.

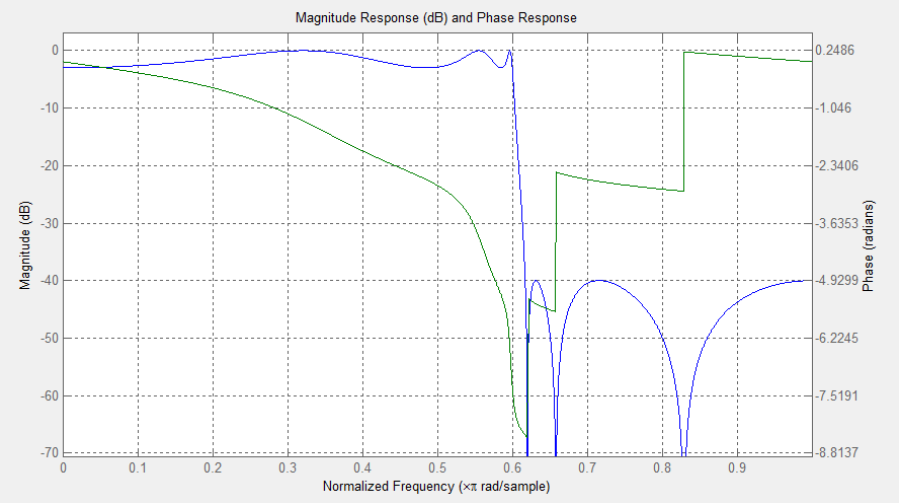
Razni autori predlažu različita rješenja pri odabiru načina podjele frekvencijskog spektra. Tolonen i Karjalainen [8] predlažu jednostavno i računski efikasno rješenje upotrebom dva kanala. Prvi koji sadrži frekvencije od 70 do 1000 Hz i drugi za frekvencije od 1000 do 10000 Hz. Dodatno za drugi kanal se vrši izdvajanje ovojnice poluvalnim ispravljanjem i istim nisko pojasnim filtrom koji je upotrijebljen za filtriranje prvog kanala.

Možda najprikladnije rješenje bilo bi upotreba skupa *gammatone* filtara za dekompoziciju [9]. Radi se o ERB (Equivalent Rectangular Bandwidth) skupu filtara

gdje je širina svakog pojasa određena bazirajući se na psihoakustičke značajke slušnog modela te sukladno tome dobro simulira odziv bazilarne membrane.

Scheirerovo [2] rješenje čini skup filtara od šest područja oštrih prijelaza kod kojeg svako područje obuhvaća otprilike tonski razmak od jedne oktave. Najniži je niskopojasni filtar na 200 Hz. Slijede ga pojasnopropusni filtri na 200 i 400Hz, 400 i 800 Hz, 800 i 1600 Hz i 1600 i 3200 Hz. Zadnji je visokopojasni filtar na 3200 Hz. Filtri su modelirani kao eliptični filtri šestog reda, sa 3 dB mreškanjem u propusnom području i 40dB u nepropusnom. Pojačanje eliptičnog filtra dano je sa:

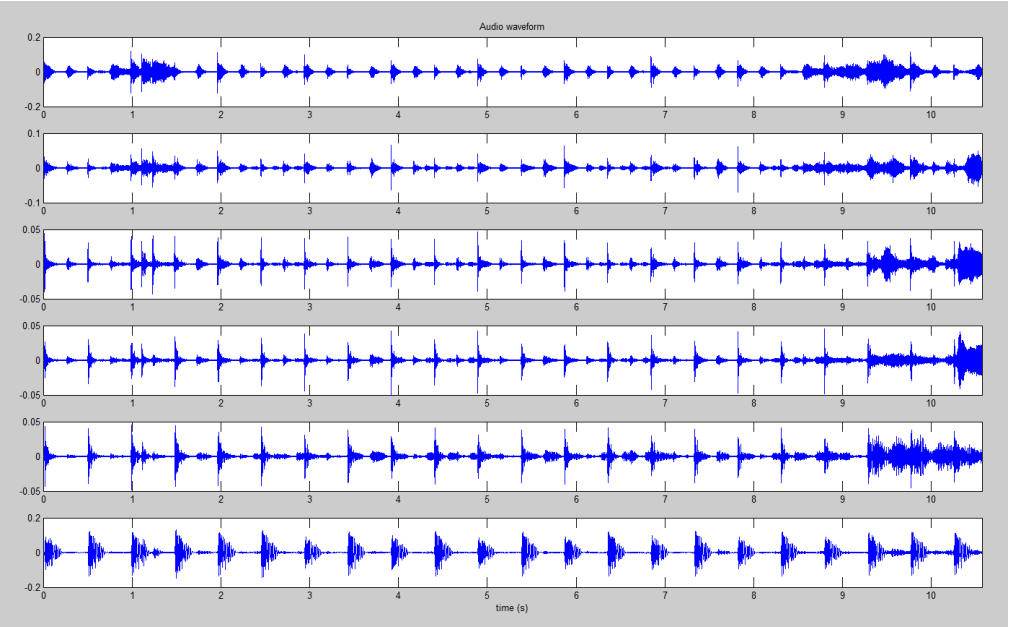
$$A_n(\omega) = \frac{1}{\sqrt{1+\varepsilon^2 R_n^2(\xi, \omega/\omega_0)}} \qquad (1)$$



Slika 1 Fazna i frekvencijska karakteristika eliptičnog filtra

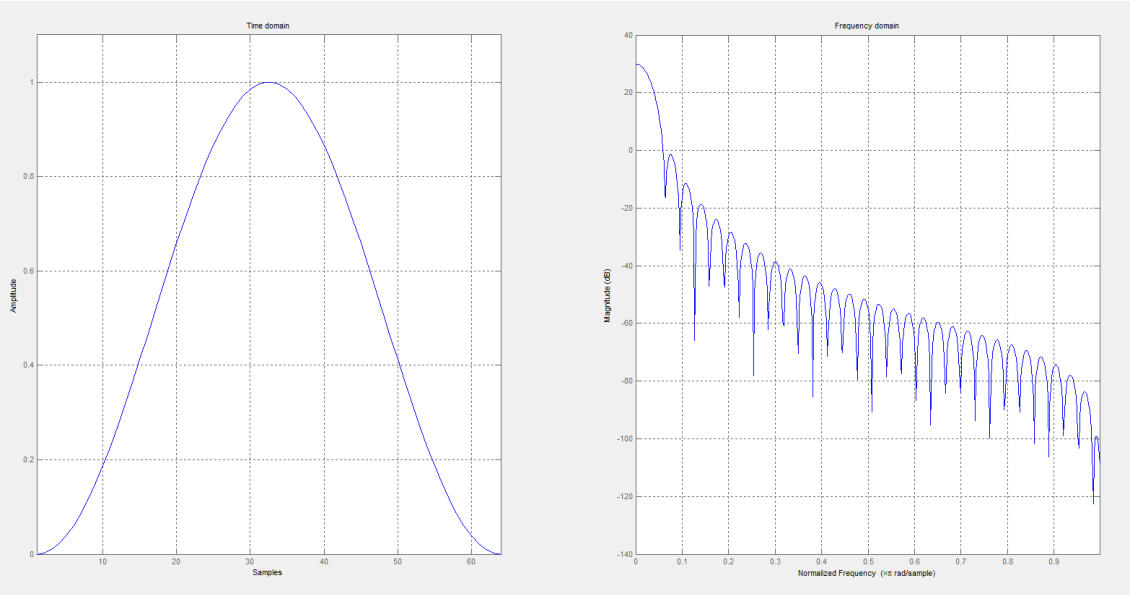
Mreškanje u propusnom području određuje ε, a u području gušenja kombinacija ε i ξ.

Primjenom skupa filtara na ulazni glazbeni zapis dobivamo šest zasebnih zapisa nad kojima nastavljamo obradu.



Slika 2 Glazbeni zapis nakon filtriranja na šest frekvencijskih područja

Za svako područje izvlačimo ovojnicu zvučnog signala. Ovojnicu ćemo dobiti tako da najprije signal ispravimo. Ispravljeni signal konvoluiramo sa pola Hannovog prozora od 200 ms. Prozor ima niskopropusnu karakteristiku sa graničnom frekvencijom od 10 Hz. Prozor ima diskontinuitet u t=0 i pada od 0 do 200 ms. Integriranje energije u prozoru provodi se na način koji je blizak psihoakustičnom modelu gdje naglašava nedavne ulaze, a maskira brze modulacije.



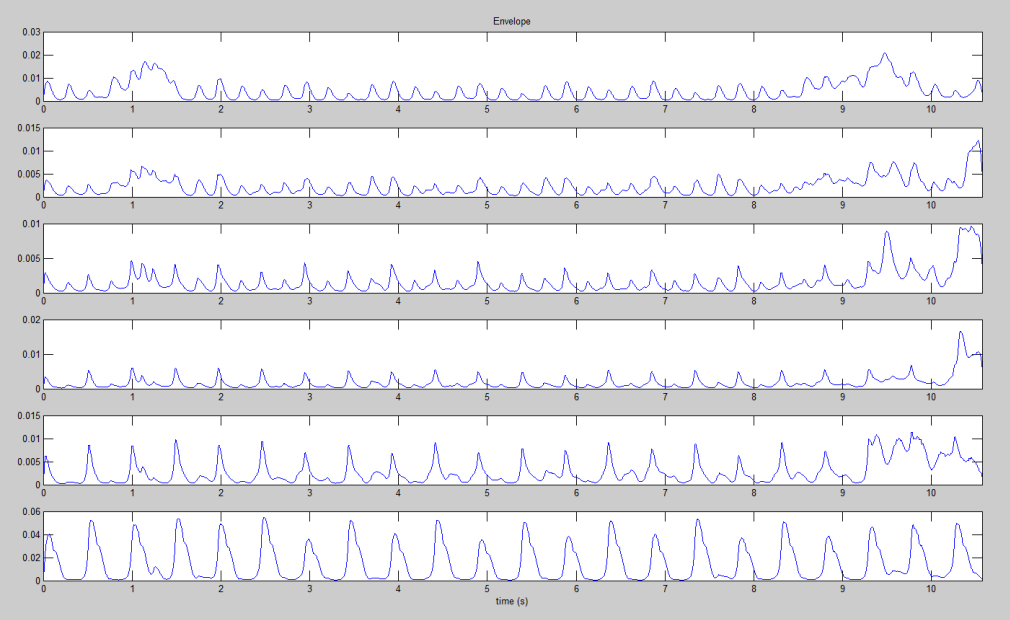
Slika 3 Hannov prozor za 64 uzorka

Jednadžba koja opisuje diskretni Hannov prozor je :

$$w(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right) \qquad (2)$$

N označuje broj uzoraka u prozoru.

Nakon što smo propustili pojasno filtrirani signal kroz Hannov prozor i ispravili ga, ovojnice signala izgledaju ovako:



Slika 4 Ovojnice pojedinih frekvencijskih komponenti glazbenog zapisa

Iz slika vidljivo je da ovojnica sadrži informaciju o nastupu doba u glazbenom zapisu. No prije nogo što krenemo iščitavati tu informaciju možemo ovojnice poduzorkovati kako bi uštedjeli na performansama pri izvođenju algoritma.

Iz zapisa glazbenog zapisa izbacit ćemo sve članove osim svakog N-tog čime vršimo poduzorkovanje faktora N. Ukoliko je ulazni glazbeni zapis uzorkovan tipičnom frekvencijom od 44100 Hz možemo za N izabrati N=1000 čime za rezultat dobivamo glazbeni zapis uzorkovan sa 441 Hz.

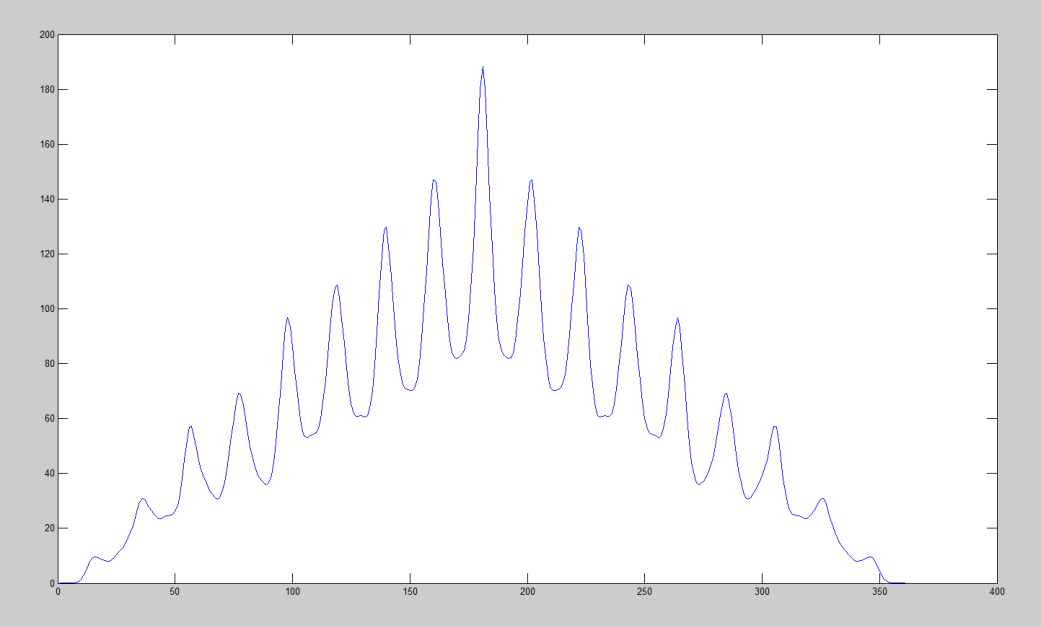
Iako na ovojnicama uočavamo vrhove kako bi dobili informaciju o tempu glazbenog zapisa potrebno je shvatiti koji od tih vrhova predstavljaju ritam i koji se od njih ponavljaju u redovitom razmaku čineći tempo glazbenog zapisa.

Za utvrđivanje vrhova koji nose informaciju o tempu glazbenog zapisa koristi ćemo autokorelacijsku funkciju. Autokorelacija se definira kao povezanost vrijednosti procesa u različitim trenutcima vremena, bilo kao funkcija dva vremena ili vremenskog pomaka. Diskretna autokorelacija definira se kao:

$$R_{xx}(j) = \sum_n x_n \overline{x_{n-j}} \qquad (3)$$

Rezultat autokorelacijske funkcije će za periodični signal, a to upravo jest naš ispitivani signal ukoliko sadrži dovoljno jasnu informaciju o ritmu, dati niz vrhova od kojih je najviši za pomak od t=0. Ukoliko se radi o glazbenom zapisu sa jednostavnim, dobro definiranim ritmom oko središnjeg vrha propagiraju se simetrično sa obje strane vrhovi nižih amplituda. Prvi susjedni vrh središnjem označava događaje koji se ponavljaju u najbližem vremenskom razmaku i vremenska udaljenost od središnjeg vrha do prvog susjednog vrha predstavlja informaciju o tempu. Ustvari vrijednost koju direktno očitavamo je vremenski razmak *D* između dvije doba ritma glazbenog zapisa no iz njega se vrlo lako dolazi do tempa glazbenog zapisa:

$$tempo = \frac{1}{D} \cdot 60 \qquad (4)$$



Slika 5 Autokorelacijska funkcija za jedno frekvencijsko područje glazbenog signala

Zadnji problem je odabrati pravu vrijednost tempa. Budući da naš algoritam dijeli glazbeni zapisa na šest frekvencijskih područja prvo pitanje koje se nameće je iz kojeg od njih izabrati vrijednost za rezultantni tempo.

Uzevši u obzir da tempo većine glazbenih zapisa ulazi u raspon od 40 do 220 bpma možemo izbaciti rezultate koji izlaze iz tog raspona.

Mogli bismo reći da se u većini slučaja informacija o tempu nalazi u najnižim frekvencijama te pridijeliti težinske faktore veće značajnosti nižim frekvencijskim područjima. Ukoliko se rezultat podudara u dva ili više frekvencijska područja možemo sa sigurnošću zaključiti da se radi o najboljem rezultatu.

Za glazbene zapise čiji ritam čine karakteristike koje se ponavljaju dvostruko ili četverostruko većom ili manjom periodičnošću korisno je u obzir uzeti i dvostruko ili čak četverostruko manje ili veće vrijednosti ukoliko one ulaze u zadani raspon od 40 do 220 bpma.

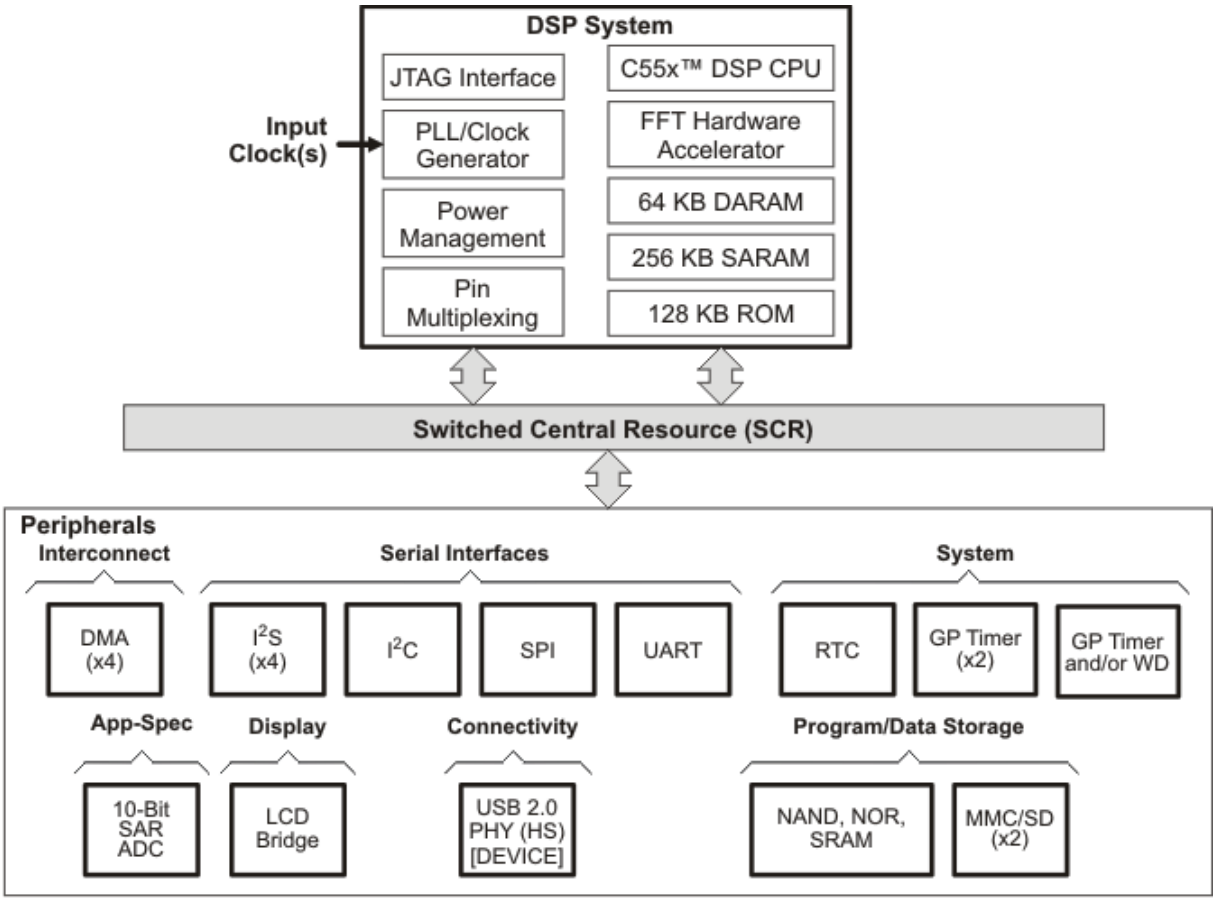
Implementacija na TMS320VC5505

Sustav za digitalnu obradu signala TMS320VC5505 je USB platforma građena oko TMS320C550x procesora firme Texas instruments. Procesor podržava unutarnju sabirničku strukturi koja se sastoji od programske sabirnice, 32 bitne sabirnice za čitanje i dvije 16 bitne sabirnice za čitanje, dvije 16 bitne sabirnice za pisanje i dodatne sabirnice za periferiju i DMA periferiju. Ove sabirnice omogućuju do četiri 16 bitna čitanja i dva 16 bitna pisanja u jednom ciklusu. TMS320VC5505 također sadrži četiri DMA kontrolera, svaki sa četiri kanala koji pružaju protok podataka za 16 nezavisnih kanala bez intervencije procesora. Svaki DMA kontroler može izvesti jedan 32 bitni prijenos podataka po ciklusu, paralelno i nezavisno o zauzetosti procesora.

C55x ima dvije jedinice za množenje-zbrajanje (MAC) svaka sposobna za 17 bitno x 17 bitno množenje i 32 bitno zbrajanje u jednom ciklusu. Središnja 40 bitna aritmetičko-logička jedinica je podržana sa dodatnom 16 bitnom jedinicom.

Standardne ulazno izlazne funkcije zajedno sa 10 bitnim SAR ADC pružaju dovoljno pinova za statuse, prekide i bitne I/O za LCD displeje, tipkovnice i medijske periferije. Dodatna proširenja omogućuju dvije MMC/SD periferije.

VC5505 podržava Code Composer Studio IDE. Code composer Studio IDE sadrži alate za generiranje kod od kojih C compiler i linker te više emulatora uređaja i module za procjene.



Slika 6 Funkcionalni blok dijagram VC5505 uređaja

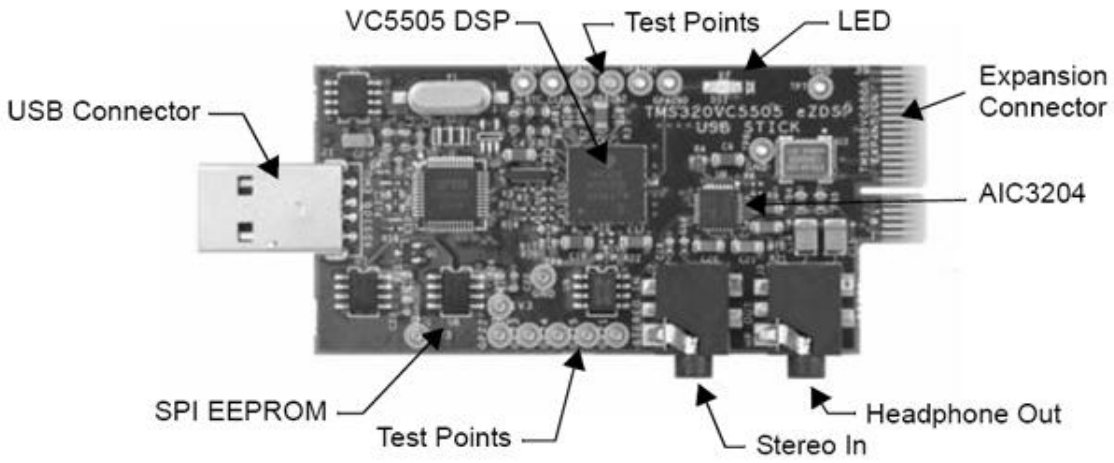
Sustav za digitalnu obradu signala TMS320VC5505 sadrži stereo audio kodek niske snage, niske voltaže TLV320AIC3204 (AIC3204). Karakteriziraju ga programabilni ulazi i izlazi, PowerTune mogućnosti, predefinirani blokovi za obradu signala sa promjenjivim parametrima, integrirani PLL, integrirani LDO regulatori i fleksibilno digitalno sučelje. Putem registara omogućuje kontrolu snage, ulazno/izlaznih operacija kanala, pojačanja, efekata i takta omogućujući precizno oblikovanje uređaja njegovoj namjeni. Obuhvaća reproduciranje zvuka od 8 kHz mono do 192 kHz stereo čineći ga prikladnim za audio i telefonijske aplikacije pogonjene baterijom.

Razvojna okolina za TMS320VC5505 dolazi sa gotovim kodom za pokretanje kodeka AIC3204 sa osnovnim uputama za čitanje i pisanje sa ulaza/izlaza kodeka. Nakon konfiguracije kodeka, pokretanja takta i PLLa, pokretanja i spajanja DACa prijenos se obavlja putem I2C sučelja. I2C pruža sučelje između DSP CPUa i ostalih uređaja kompatibilnih sa I2C sabirničkom specifikacijom. Vanjske komponente koje su

spojene na ovu dvožičnu sabirnicu mogu slati i primati podatke široke do osam bita prema i od DSPa.

Kako DSP CPU podržava hardverski ubrzanu FFT sa razvojnim sustavom dolazi softverska implementacija iste.

Sam procesor sastoji se od dvostrukog MAC radnog takta do 100 Mhz, priručne memorije i to 320 KB RAMa i 128 KB ROMa, četiri DMA kontrolera i vanjskog memorijskog sučelja, modula za upravljanje snagom i spomenutim I2C i ostalom periferijom te jedinicom za hardverski ubrzanu FFT (HWAFFT) koja podržava realni i kompleksni izračun FFTa od 8 do 1024 točke.



Slika 7 Osnovne komponente VC5505 eZDSP USB sustava

Sustav za digitalnu obradu signala TMS320VC5505 na računala se spaja putem USB sučelja koje osim za komunikaciju sa računalom koristi i za napajanje sustava.

Audio ulaz u sustav čini 3.5 mm priključak na sustavu od kojeg signal ide preko ADCa u AIC3204 koder.

Početak implementacije počinje sa čitanjem glazbenom zapisa u ulaza u TMS320VC5505. Signal sa 3.5 mm ulaza preko AD pretvornika dolazi u AIC3204 kodek. Kako bi mogli čitati sa kodeka potrebno ga je inicijalizirati. U datoteci aic3204.h nalaze se dvije funkcije:

```
int16 startup_aic3204();
```

Funkcija inicijalizira AIC3204. Odabire stranice, komunicira sa I2S, postavlja DA i AD pretvornik, postavlja PLL, konfigurira takt.

Funkcija

```
int16 shutdown_aic3204();
```

poziva se pri kraju izvođenja i ona isključuje I2S i resetira kodek AIC3204.

Čitanje podataka sa ulaza odvija se u jednoj *for* petlji. Podaci sa lijevog i desnog kanala se usrednjuju u mono signal i zapisuju u privremenu varijablu:

```
for ( i = 0 ; i < 4 ; i++ ){

      for ( j = 0 ; j < 1000 ; j++ ){

            for ( sample = 0 ; sample < 48 ; sample++ ){

                  /* čitanje sa audio ulaza */

                  while((RcvR & I2S0_IR) == 0);

                        // Čekanje na prekid za primanje

                  lch = I2S0_W0_MSW_R;

                        // 16 bitni audio podatak sa lijevog kanala
                  rch  = I2S0_W1_MSW_R;

                        // 16 bitni audio podatak sa desnog kanala

                  buff[(i+1)*(j+1)+(sample+1)]=(lch + rch)/2;

}}}
```

Kako je frekvencija uzorkovanja 48000 Hz jednim prolazom se učita 4*1000*48 puta što je 4 sekunde audio signala u privremenoj varijabli. Na privremenoj varijabli provodimo FFT.

Razvojna okolina dolazi sa implementacijom FFT algoritma uz upotrebu hardverskog ubrzanja u vidu softverski kontroliranog koprocesora dizajniranog za izračun FFT i inverzne FFT (IFFT) nad kompleksnim vektorima podataka u dužini od 8 do 1024 točke. Implemenitran je kao hardverska "leptir", korijen-2 struktura koja izračunava FFT ili IFFT u obrnutom redoslijedu bitova. Implementacija podržava "double-stage" mod u kojem se rezultat iz prvog stadija vraća u hardverski leptir kako bi se izračunao drugi stadij u jednom prolazu. Kako bi se maksimizirala propusnost kompleksna množenja sa težinskim faktorima izvode se u prvom dijelu cjevovoda, a kompleksno zbrajanje i oduzimanje u drugom dijelu.

No softver je nužan za komuniciranje između CPU i HWAFFT. Arhitektura seta instrukcija (ISA) za CPU uključuje klasu koprocesorskih instrukcija koje omogućuju da CPU započne, dohvati podatke i izvrši leptir proračune na HWAFFT. Na raspolaganju su i funkcije za pozivanje u C-u sa optimiziranim sekvencama koprocesorskih instrukcija za svaku raspoloživu dužinu FFTa. Kako bi se sačuvala programska memorija ove su funkcije pohranjene u DPSovom ROMu. Također radi čuvanja propusnosti memorijske sabirnice težinski faktori pohranjeni su u tablici unutar HWAFFT koprocesora.

Kako su proračuni sa nepomičnim zarezom osjetljivi na preljev, podljev i zasićenja, ovisno o dinamičkom rasponu ulaznih podataka moguće je da će bit potrebno skaliranje da bi se izbjegle ove nepoželjne posljedice. Skaliranje se može izvesti prije izračuna FFT izračunom dinamičkog raspona ulaznih točaka i smanjenjem istog. Ukoliko je amplituda svakog kompleksnog ulaznog elementa manja od 1/N gdje je N dužina FFTa tada FFT u N točaka neće imati preljev. Uniformno dijeljenje ulaznih elemenata vektora sa N jednako je posmaku u desno za log2(N) bitova što unosi značajnu pogrešku u izračun. Alternativno dijeljenje sa dva zaokruživanje nakon svakog leptira pruža dovoljno dobar kompromis između preciznosti i zaštite od preljeva u isto vrijeme smanjujući računske cikluse.

FFT u 1024 točke poziva se sa:

```
out_sel = hwafft_1024pts(data, scratch, fft_flag,scale_flag);
```

`Int32 *data` - ulazni vektor podataka. Sadrži kompleksne elemente podataka, realni dio kojih je u najviših 16 bita, kompleksni u nižih 16 bita. Nakon izračuna FFTa rezultat je pohranjen ili u data vektoru ako je povratna vrijednost 0 ili u scratch vektoru ako je povratna vrijednost jednaka 1. Podaci u data i scratch vektorima nalaze se u odvojenim blokovima RAM memorije kako bi se maksimizirala memorijska propusnost

`Int32 *scratch` - vektor podataka u kojem se pohranjuju međurezultati između FFT stadija

Četiri zastavice određuju da li će algoritam računati FFT ili IFFT, da li će skalirati međurezultat za dva nakon svakog prolaza kroz leptir:

- fft_flag=FFT_FLAG računa se FFT
- fft_flag=IFFT_FLAG računa se IFFT
- scale_flag=SCALE_FLAG dijeli se međurezultat sa dva nakon svakog prolaza kroz leptir
- scale_flag=NOSCALE_FLAG ne provodi se skaliranje, mogući preljev

Prilikom opisa algoritma i modeliranja algoritma u programskom okruženju Matlab način primjene tehnika obrade signala, pogotovo primjena filtara, nije nam bio od značaja zbog transparentnosti kojom se ti zadaci izvršavaju u programskom okruženju Matlab. Prilikom implementacije na sustavu za digitalnu obradu signala TMS320VC5505 morat ćemo detaljnije promotriti način na koji ćemo provesti pojedine operacije nad ulaznim signalom.

Ukoliko imamo ulazni signal y(t), primjenu filtra f(f) predstavljamo kao konvoluciju ta dva signala:

$$x(t) = \int_{-\infty}^{\infty} f(\tau) y(t-\tau) d\tau \qquad (5)$$

Kako kodek obavlja očitavanje ulaznog, vremenski kontinuiranog signala frekvencijom 48 kHz dobivamo digitalni signal za koji je konvolucija sa filtrom dana sa:

$$x[n] = \sum_{k=0}^{P+Q-2} f[n]y[n-k] \qquad (6)$$

gdje P predstavlja broj uzoraka signala f[n], a Q broj uzoraka signala y[n].

Iako bi konvoluciju mogli računati direktno prema formuli (6), budući nam je dostupna implementacija FFTa koristit ćemo teorem o konvoluciji koji kaže da za dva vremenska signala f[n] i y[n] čije su diskretne Fourierove transformacije F[m] i Y[m] diskretna Fourierova transformacija konvolucije signala f[n]*y[n] jednaka umnošku F[m]·Y[m]:

$$f[n] * y[n] \xrightarrow{DFT} \xleftarrow{IDFT} F[m] \cdot Y[m] \qquad (7)$$

Upravo ćemo to svojstvo koristiti u implementaciji algoritma. Korištenjem HWAFFTa računamo Fourierovu transformaciju ulaznog signala. Niskopropusni filtar kojim ćemo srezati visoke frekvencije primijenit ćemo tako da na već prije izračunate koeficijente izabranog filtra spremljene u priručnoj datoteci primijenimo HWAFFT. Dobivene transformacije ulaznog signala i odabranog filtra kompleksno množimo u frekvencijskoj domeni. Rezultat prosljeđujemo HWAFFTu uz postavljenu zastavicu za inverznu Fourierovu transformaciju te dobivamo naš ulazni signal niskopojasno filtriran sa prethodno izračunatim filtrom. Prototip funkcije koja primjenjuje filtriranje:

```
Uint16 FFT_Filter(Int32 *In, Int32 *Out);
```

Int32 *In sadržava signal sa ulaza koji nakon filtriranja i kompleksnog množenja u frekvencijskog domeni vraća se u varijabli Int32 *Out.

Prototip funkcije kompleksnog množenja:

```
Int32 CPLX_Mul(Int32 op1, Int32 op2);
```

Ulazni argumenti su u 32 bitnoj preciznosti gdje je u viših 16 bita pohranjen realni dio, a u nižih kompleksni dio te se pomakom prebacuju realni i kompleksni dio u pomoćne varijable.

```
p1_r = op1 >> 16;

op1_i = op1 & 0x0000FFFF;
```

```
op2_r = op2 >> 16;

op2_i = op2 & 0x0000FFFF;
```

Prema algoritmu nastavljamo sa ispravljanjem ulaznog signala. Koristit ćemo jednostavno prolaz kroz sve elemente prozora za obradu signala i izračunati apsolutnu vrijednost elementa.

Zaglađivanje ulaznog signala vršimo ponovo prethodno modeliranim filtrom. Ulazni signal i izračunate koeficijente niskopropusnog filtra odvojeno dovodimo na HWAFFT. Nakon kompleksnog množenja dvaju signala i još jedne IFFT od našeg ulaznog signala imamo sada ovojnicu nad kojom vršimo autokorelaciju.

Prije računanja autokorelacije provodimo poduzorkovanje prozora ulaznog signala kako bi uštedjeli na performansama prije računanja autokorelacije. Koristit ćemo visokopropusni filtar već opisanom metodom te nakon toga izvršiti decimaciju faktorom 1000.

Autokorelaciju nakon poduzorkovanja možemo implementirati prema defiiniciji (3). Pretragom po polju elemenata prozora ulaznog signala pronalazimo najviši vrh koji odgovara pomaku u t=0. Od upravo određenog t=0 nastavljamo pretragu u desno i sljedeći maksimum predstavlja vrh čija udaljenost od t=0 predstavlja tempo.

Prototip autokorelacijske funkcije:

```
Int16 autocorr(Int32 *In);
```

Autokorelacijsku funkciju predstavlja polje `corrp` dužine N sa umnošcima vlastitih pomaka.

```
for (i=0; i<N; i++) {

     if((i-j)>=0)

          corrp(j)=In(i)In(i-j);

}
```

Množimo uzorke ulaznog signala sa pomaknutim uzorcima istog i dobivamo novo polje sa autokorelacijom iz kojeg izvlačimo prvi maksimum nakon indeksa 0:

```
for (i=0; i<N; i++) {
      if(corrp(i+1)<corrp(i)) {
            if (max!= 0)
                  break;
      }
      else if(corrp(i+1)>corrp(i)) {
            max=i+1;
            l=0;
      }
      else {
            if(max != 0) {
                  avg += max;
            }
      l++;
      }
if (l)
      max=avg/l;
else
      max=max;
```

Kvaliteta algoritma

Za testiranje robusnosti algoritma upotrijebljen je skup od 200 glazbenih zapisa različitih žanrova, sukladno tome očekivano različitog tempa. Veliki dio zapisa čini presjek glazbe drugog djela 20. stoljeća žanrova poput rocka, popa, reggae, rapa, techno. Radi kompletnosti dodani su deseci glazbenih zapisa klasične glazbe, jazza, latino i plesne glazbe. Glazbeni zapisi koji ne spadaju striktno u jednu od navedenih kategorija svrstani su pod ostalo. Ideja je da su svi žanrovi više manje jednako zastupljeni.

Testiranje je vršeno usporedbom rezultata dobivenog primjenom opisanog algoritma sa rezultatom slušatelja tapkanjem uz aplikaciju za detektiranje tempa[1].

U Tablici 1 prikazani su rezultati testiranja algoritma. Po žanru zabilježeno je podudaranje rezultata dobivenog algoritmom sa rezultatom slušača u postotcima. Za svaki glazbeni zapis zabilježeno je podudaranje R po formuli:

$$R = \frac{R_s}{R_a} \cdot 100 \qquad (8)$$

R_s je rezultat slušača, R_a je rezultat algoritma. U slučaju $R_s > R_a$ brojnik i nazivnik mijenjaju mjesta. Rezultat po žanrovima te ukupan rezultat dobiveni su kao aritmetička sredina pojedinih rezultata.

Dodatno u Tablici 1 zabilježen je ukupan broj pogođenih rezultata. Naime algoritam je za ritmove kojih se slične ritamske karakteristike ponavljaju sa dvostruko ili četverostruko većom ili manjom periodičnošću u teoriji dao upotrebljiv rezultat koji se može dodatno interpretirati no kao gotov rezultat predstavlja pogrešan rezultat. Stoga je u stupcu *Ukupno pogođeno* dodatno zabilježen ukupan postotak rezultata koji nisu višekratnici baznog tempa već predstavljaju upravo traženi tempo.

[1] http://www.tempotap.com/

Žanr glazbe	Podudaranje rezultata [%]	Ukupno pogođeno [%]
rock	88,58	76
jazz	85,86	60
latino	82,39	52
pop	81,77	76
reggae	96,57	64
rap	87,78	76
techno	90,15	100
klasična	89,05	64
ostalo	86,02	60
ukupno	**87,57**	**69,77**

Tablica 1 Rezultati testiranja algoritma

Vidimo da žanrovi u kojoj je tempo dobro izražen u nižim frekvencijskim linijama prolaze dobro na algoritmu. Klasična glazba i glazba sa kompleksnijim sinkopiranim ritmovima postižu najlošije rezultate.

Iako sama detekcija tempa postiže čak vrlo dobre rezultate ovisno o žanru broj rezultata koji su dvostruko/četverostruko manji/veći od pravog tempa je značajan.

Zaključak

U radu je predstavljen jedan mogući algoritam za određivanje tempa glazbenog zapisa i uspješno realiziran u programskoj okolini Matlab. Modifikacija algoritma u stvarnom vremenu implementirana je na sustavu za digitalnu obradu signala TMS320VC5505.

Algoritam se pokazao solidnim za glazbene zapise koje imaju izraženi ritam no potreban su poboljšanja za određene žanrove glazbe kao što je klasična glazbe, pogotovo solo izvedbe bez pratnje orkestra.

Moguće poboljšanje algoritma je u segmentaciji frekvencijskog spektra, koristeći područja koja bolje opisuju slušni model poput gammatone skupa filtra. Današnje metode prepoznavanje nastupa doba autokorelacijom kombiniraju sa spektralnom analizom te je ukupni rezultat kombinacija obje metode.

Implementacija na sustavu za digitalnu obradu signala može se unaprijediti korištenjem većeg paralelizma u algoritmu i stvaranju dodatnih dretvi već prema mogućnosti sustava za digitalnu obradu signala TMS320VC5505.

Iako je sam algoritam više eksperimentalne vrijednosti njegova implementacija na gotovo prijenosnoj platformi i univerzalnom kodeku AIC3204 omogućuje korištenje rješenja za sortiranje većeg broja glazbenih zapisa prema tempu ili pak korištenje gotovog rješenja za sinkronizaciju glazbenog zapisa sa vizualnim animacijama.

Bibliografija

1. M. Alonso, B. David, G. Richard, "Tempo and beat estimation of musical signals", *Proc. of ISMIR 2004*, pp. 158-163, 2004.
2. E. Scheirer, "Tempo and beat analysis of acoustic musical signals", *Journal of Acoustic Society of America, 1998,* vol. 103, pp. 588-601, 1998.
3. F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, P. Cano, "An experimental conparison of audio tempo induction algorithms*", IEEE Trans. Audio, Speech and Language Processing.* IEEE, pp. 1832-1844, 2006.
4. J. C. Brown, "Determination of meter of musical scores by autocorelation", *Journal of the Acoustical Society of America*, 94, pp. 1953-1957, 1993.
5. H. Ishizaki, K. Hoashi, Y. Takishima, "Autocorelation-based beat estimation adaptive to drastic tempo change in a song", *IEEE International Conference on Multimedia and Expo*, pp. 478-481, 2009.
6. A. Klapuri, "Sound Onset Detection by Applying Psychoacoustic Knowledge", *Proceedings IEEE Int. Conf. Acoustics Speech and Sig. Proc (ICASSP)*, pp. 3089-3092, Phoenix AR, USA March 1999.
7. J. Laroche, "Efficient Tempo and Beat Tracking in Audio Recordings*", J. Audio. Eng. Soc.*, vol. 51, No. 4, pp. 226-233, April 2003.
8. T. Tolonen, M. Karjalainen," A computationally efficient multipitch analysis model", *Speech and Audio Processing, IEEE Transactions on, Vol. 8, No. 6,* pp. 708-716, 2000.
9. R. D. Patterson, K. Robinson, J. Holdsworth, D. McKeown, C.Zhang, M. H. Allerhand, “Complex sounds and auditory images,” *Auditory Physiology and Perception,* Oxford, pp. 429-446, 1992.
10. M. McKeown. (lipanj 2014.). FFT Implementation on the TMS320VC5505, TMS320C5505, and TMS320C5515 DSPs, [Online] Dostupno: http://www.ti.com/lit/an/sprabb6b/sprabb6b.pdf
11. T. Horvat. (lipanj 2014.). Brza Fourierova transformacija na procesoru TMS320VC5505, [Online] Dostupno: http://bib.irb.hr/datoteka/658798.zavrsni_rad.pdf