SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2111

BRZA FOURIEROVA TRANSFORMACIJA NA PROCESORU TMS320VC5505

Tomislav Horvat

Zagreb, lipanj 2011.

Sadržaj

1.Uvod

U okviru ovog rada biti će prikazan postupak brze Fourierove transformacije I njezine izvedbe na procesoru za digitalnu obradu signala TMS320VC5505. Signal će se čitati sa AD pretvornika audio kodeka. Izračunati koeficijenti transformacije ,u logaritamskom mjerilu, biti će prikazani pomoću DA pretvornika na osciloskopu.

Brza Fourierova transformacija je *divide-and-conquer* algoritam za brzo i efikasno izvođenje diskretne Fourierove transformacije, kao i njezinog inverza, u logaritamskom vremenu. Brzu Fourierovu transformaciju označavat ćemo nadalje sa FFT (kratica za ”fast Fourier transform”). Ove transformacije, kao i ovakva vrsta analize, nose naziv po istaknutom francuskom matematičaru i fizičaru Josephu Fourieru. Algoritam se prvi put spominje u Gaussovom tekstu “Nachlass: Theoria interpolationis methodo nova tractata” 1805. godine. Implementaciju algoritma na računalima u raznim radovima opisali su Cooley i Tukey 1965. godine.

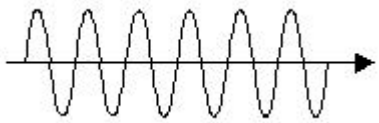
2.Fourierova analiza

Za početak ćemo reći nešto o Fourierovoj analizi. Bilo koji vremensko promjenjivi signal može se konstruirati zbrajanjem sinusnih signala različite frekvencije, amplitude i faze. Fourierova analiza omogu**ć**uje prikaz nekog signala kao zbroj sinusnih signala odnosno kao zbroj više različitih frekvencijskih komponenti. Ukoliko je pojedina frekvencijska komponenta male amplitude ili je jednaka nuli tada ona ne pridonosi amplitudi ukupnog signala. Prema tome ako želimo napraviti Fourierovu analizu nad sinusnim signalom dobiti ćemo šiljak na frekvencijskoj komponenti koja odgovara frekvenciji tog sinusa dok će doprinosi ostalih frekvencijskih komponenti biti jednaki nuli.

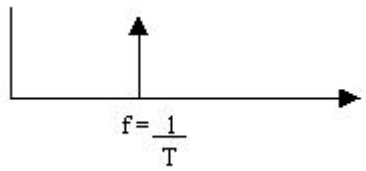
Fourierova analiza nam omogu**ć**uje da signal promatramo u vremenskoj ili frekvencijskoj domeni. Frekvencijska domena se naziva spektrom signala.

Izgled spektra nekih signala:

Sinus: amplituda



vrijeme

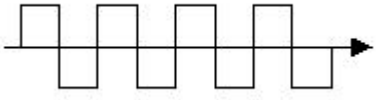


frekvencija

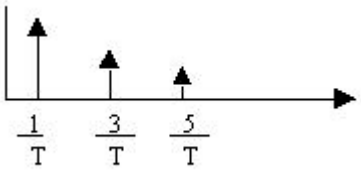
Slika 1.a *Slika 1.*

Pravokutni signal:

amplituda



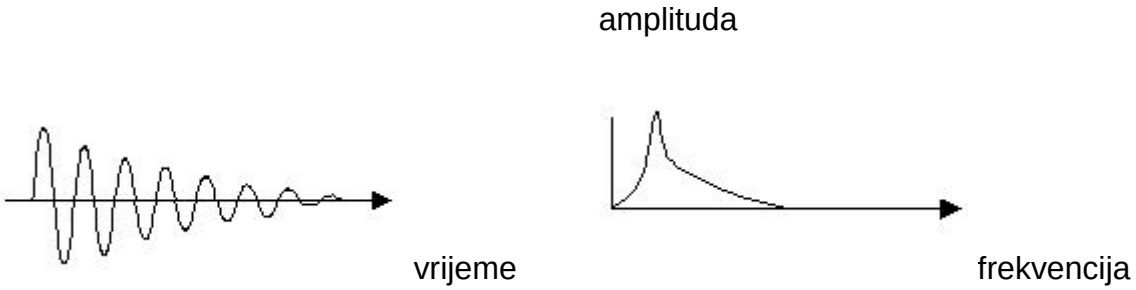
vrijeme



frekvencija

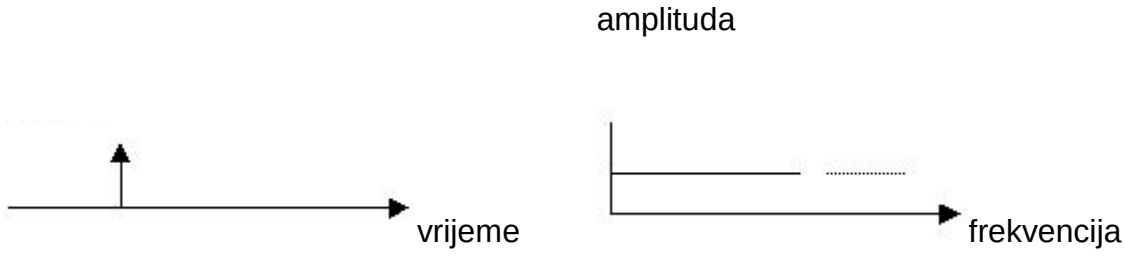
Slika 2.a *Slika 2.b*

Prigušeni sinus:



Slika 3.a *Slika 3.b*

Impuls:



Slika 4.a *Slika 4.b*

U Fourierovoj analizi najčešće se koristi vremenski kontinuirana fourierova transformacija(CTFT), razvoj u Fourerov red, vremenski diskretna transformacija(DTFT) te diskretna transformacija (DFT).

2.1. Vremenski kontinuirana transformacija (CTFT)

Vremenski kontinuirana transformacija (CTFT) definirana je izrazom 1. pomoću nje računamo spektar ne periodičnih signala čija funkcija ima za argument kontinuiranu varijablu.

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt \qquad (1.)$$

Inverzna transformacija računa se prema formuli 2.

$$f(t) = \int_{-\infty}^{\infty} F(s)e^{j\omega t}\,ds \qquad (2.)$$

2.2. Fourierov red

Razvojem u Fourierov red dobivamo frekvencijske komponente periodičnih kontinuiranih signala.(3.)

$$Xk = \frac{1}{T0}\sum_{-\infty}^{\infty} X(t)e^{-j\omega kt} \qquad (3.)$$

Inverz je dan sljedećim izrazom:

$$Xk = \frac{1}{T0}\sum_{-\infty}^{\infty} X(t)e^{-j\omega kt} \qquad (4.)$$

2.3. Vremenski diskretna transformacija (DTFT)

Vremenski diskretna transformacija računa spektar vremenski diskretnih signala Računa se prema izrazu 5.

$$X\left(e^{j\omega}\right) = \sum_{-\infty}^{-\infty} x(n)e^{-j\omega n} \qquad (5.)$$

Izraz za vraćanje iz frekvencijske domene u vremensku :

$$X(n\ \ ) = \frac{1}{2\pi}\int_{-\infty}^{-\infty} X\left(e^{j\omega}\right)\ \ e^{j\omega n} \qquad (6.)$$

2.3. Diskretna Fourierova transformacija (DFT)

Velik broj signala nije moguće matematički definirati. Zbog toga se uvodi diskretna Fourierova transformacija koja numerički računa spektar signala koji se sastoji od konačnog broja diskretnih uzoraka. Ti uzorci se dobivaju utipkavanjem nekog kontinuiranog signala (npr. govora). Spektar diskretnih, aperiodičnih signala je kontinuiran i periodičan. Prema tome općenito možemo zapisati:

$$\mathrm{X}\left(\mathrm{e}^{\mathrm{j}\Omega}\right) = \sum_{-\infty}^{\infty} \mathrm{x(n)}\, e^{-j\Omega n} \qquad (7.)$$

Pošto je spektar periodičan dovoljno očitati N uzoraka osnovnog perioda sa razmakom 2π/N između susjednih uzoraka. Dobivamo sljedeći izraz

$$\mathrm{X}\left(\mathrm{e}^{\frac{\mathrm{j2\pi k}}{\mathrm{N}}}\right) = \sum_{-\infty}^{\infty} \mathrm{x(n)} \qquad (8.)$$

$$\mathrm{X}\left(\mathrm{e}^{\frac{\mathrm{j2\pi k}}{\mathrm{N}}}\right) = \cdots + \sum_{-N}^{-1} \mathrm{x(n)}\, e^{-\frac{j2\pi kn}{N}} + \sum_{0}^{N-1} \mathrm{x(n)} e^{-j2\pi n/N} + \cdots$$

$$= \sum_{-\infty}^{\infty} \sum_{n=mN}^{mn+N-1} \mathrm{x(n)}\, e^{-\frac{j2\pi kn}{N}} \qquad (9.)$$

U unutarnjoj sumi zamijenimo indekse n-mN sa n, te zamijenimo redoslijed zbrajanja :

$$\mathrm{X}\left(\mathrm{e}^{\frac{\mathrm{j}2\pi\mathrm{k}}{\mathrm{N}}}\right) = \sum_{\mathrm{n}=0}^{\mathrm{N}-1}\sum_{-\infty}^{\infty}\mathbf{x}(\mathbf{n})\,\mathrm{e}^{-\frac{\mathrm{j}2\pi\mathrm{kn}}{\mathrm{N}}} \qquad (10.)$$

za k=0,1,2...N-1

Gdje je

$$\mathbf{x}(\mathbf{n}) = \sum_{m=-\infty}^{\infty}\mathrm{x}(\mathrm{n}-\mathrm{m}) \qquad (11.)$$

Za aperiodične signale duljine L vrijedi pri čemu je L<=N vrijedi:

$$\mathbf{x}(\mathbf{n}) = \mathrm{x}(\mathrm{n}) \qquad \text{za } 0<n<N-1 \qquad (12.)$$

Ako uvedemo oznaku za koeficijente $X(k) = X(e^{j2\pi k/N})$ dobivamo jednadžbu za izračunt DFT-a (13.) i inverz IDFT (14.)

$$\mathrm{X}(\mathrm{k}) = \sum_{0}^{N-1}\mathrm{x}(\mathrm{n})\,e^{-\frac{j2\pi kn}{N}} \qquad (13.)$$

Za k=0,1,2...N-1

$$\mathrm{x}(\mathrm{n}) = \frac{1}{\mathrm{N}}\sum_{\mathrm{k}=0}^{\mathrm{N}-1}\mathrm{X}(\mathrm{k})\,\mathrm{e}^{\frac{\mathrm{j}2\pi\mathrm{kn}}{\mathrm{N}}} \qquad (14.)$$

k=0,1,2...N-1

Najveća frekvencijska komponenta spektra izračunatog DFT-om naziva se Nyquistova frekvencija i iznosi :

fmax=S/2

Gdje je S frekvencija kojom je otipkan signal. Ako signal sadrži komponente veće od Nyquistove frekvencije doći ce do aliasinga. Kako bi se to izbjeglo signali se filtriraju

prije utipkavanja kako bi se iz njega maknule visoke frekvencijske komponente. Aliasing je moguće izbjeći i povećanjem frekvencije utipkavanja, međutim to za posljedicu ima smanjenje rezolucije spektra, odnosno povećavanje razmaka između susjednih frekvencijskih komponenti.

Sljedeći problem koji se javlja je to što je ulazni signal potrebno ograničiti na nekom konačnom intervalu. To se postiže njegovim množenjem sa funkcijom koja u željenom intervalu ima „prozor“, dok je ostalo vrijeme jednak nuli. Posljedica toga je izobličenje spektra. Spektar se produljuje te može doći do aliasinga. Rješenje za ovaj problem je odabrati takvu prozorsku funkciju koja će minimalno proširiti spektar.

Svojstvo DFT-a da računa spektar signala koji se sastoje od konačnog broja uzoraka,čini ju idealnom za obradu informacija pohranjenih u računalima. Osim analize spektra nalazi ona nalazi svoju primjenu u računanju diferencijalnih jednadžbi, množenju velikih cijelih brojeva, u operacijama poput konvolucije i slično. Kako bi te sve primjene bile moguće , potrebni su efikasni algoritmi. U praksi se najčešće koriste algoritmi brze Fourierove transformacije (FFT).

3. Brza Fourierova transformacija

FFT algoritmi se zasnivaju na podijeli osnovnog DFT-a na više kraćih DFT-a. Duljina DFT-ova je najčešće potencija broja 2 što rezultira vrlo brzom izvođenju. Takvi algoritmi se efikasni zbog toga što više puta iskorištavaju pojedine međurezultate i time smanjuju ukupno potreban broj množenja. Sumu za izračun DFT-a možemo raspisati na sljedeći način:

$$\mathrm{X(k)} = \sum_{0}^{N-1} \mathrm{x(n)}\, e^{-\frac{j2\pi kn}{N}} =$$

$$\sum_{0}^{\frac{N}{2}-1} \mathrm{x(2n)}\, e^{-\frac{j2\pi k2n}{N}} + \sum_{0}^{\frac{N}{2}-1} \mathrm{x(2n+1)}\, e^{-\frac{j2\pi k(2n+1)}{N}} =$$

$$\sum_{0}^{\frac{N}{2}-1} \mathrm{x}(2\mathrm{n})\, e^{-\frac{j2\pi nk}{\frac{N}{2}}} + e^{-\frac{j2\pi k}{N}} \sum_{0}^{\frac{N}{2}-1} \mathrm{x}(2\mathrm{n}+1)\, e^{-\frac{j2\pi kn}{\frac{N}{2}}} =$$

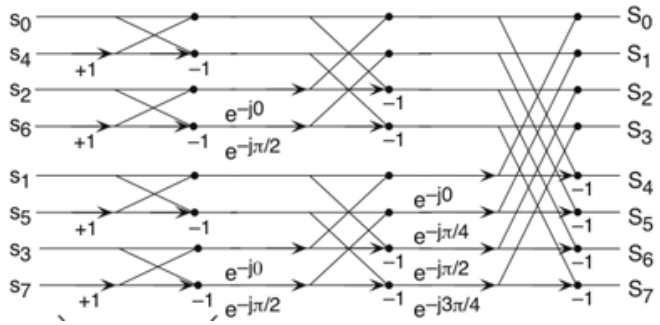
$$DFT\frac{n}{2}[\mathrm{x}(0) + \mathrm{x}(2) + \mathrm{x}(4) + \cdots + \mathrm{x}(\mathrm{N}-2)] + \mathrm{W}(\mathrm{kn}) DFT\frac{n}{2}[\mathrm{x}(1) + \mathrm{x}(3) \ \ldots + \mathrm{x}(\mathrm{N}-1)] \qquad (15.)$$

$\mathrm{W}(\mathrm{kn})$ se naziva *twiddle* faktor i iznosi $\mathrm{W}(\mathrm{kn}) = \mathrm{e}^{-\frac{\mathrm{j}2\pi\mathrm{k}}{\mathrm{N}}}$

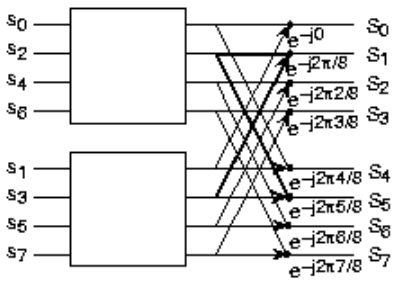
Ovakav postupak rastavljanja originalnog DFT-a (veličine N) na dva manja DFT-a (veličine N/2) od kojih jedan sadrži parne, a drugi neparne koeficijente naziva se decimacija u vremenu.

U ovom slučaju FFT iskorištava rezultate dobivene izračunom upola kraćih DFT-ova, kombinira ih, zbraja te množi sa twiddle faktorom. Dakle računamo transformacije duljine N/2 i složenosti $2O(\frac{N^2}{4})$, množimo jednu od njih sa $\mathrm{e}^{-\frac{\mathrm{j}2\pi\mathrm{k}}{\mathrm{N}}}$ (složenost O(n)) i međusobno zbrojimo (složenost O(n)). Ukupnu složenost dominantno određuje izračun dvaju DFT-ova. Međutim ako je veličina početnog DFT-a potencija broja 2, ($N = 2^l$) moguće je svaki od $DFT\frac{N}{2}$ dalje podijeliti, pa ćemo tako dobiti četiri $DFT\frac{N}{4}$. Tako možemo raspolavljati DFT-ove sve dok ne dobijemo DFT-ove veličine 2.

Izračun DFT-a možemo podijeliti na korake i pri tome broj koraka biti jednak $\log_2$N. Ukupna složenost takvog algoritma je O (N $\log_2$(n)). Ovaj postupak je prikazan na slikama 4. i 5.

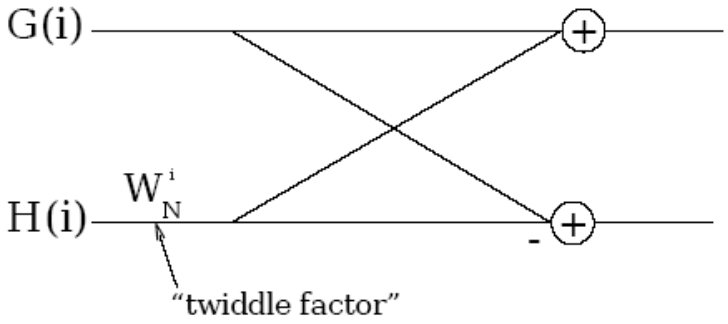


Slika 4. Koraci FFT-a



Slika 5. Izračun FFT-a

Još jedna od prednosti ovakve implementacije FFT-a je to što se iskorištava činjenica da za izračun DFT-a za N=2 nisu potrebna dodatna množenja osim množenja jednog ulaza s twiddle faktorom. Izračun se vrši pomoću leptir operacija:



Slika 6. Izračun DFT-a za N=2

3.1. Permutacija članova ulaznog niza.

Pri opisanom postupku implementacije FFT-a dolazi do permutacije početnog niza. Ukoliko raspišemo indekse permutiranih elemenata za *n* = 8 u binarnom obliku i usporedimo ih sa indeksima ne permutiranih elemenata možemo uočiti pravilnost:

Tablica 1.

Permutirani indeksi	Permutirani indeksi (binarno)	početni indeksi	početni indeksi (binarno)
0	000	0	000
4	100	1	001

2	010	2	010
6	110	3	011
1	001	4	100
5	101	5	101
3	011	6	011
7	111	7	111

Binarna reprezentacija indeksa permutiranih elemenata je u stvari binaran broj kojemu su znamenke obratnog redoslijeda od binarne reprezentacije indeksa elemenata početnog niza. U kodu ce dakle biti potrebno implementirati funkciju koja računa permutirane indekse te na osnovu toga permutira članove niza. Takve funkcije imaju se zovu „bit-reversal“.

Uzmimo za primjer indeks 6 kome je potrebno znamenke posložiti obrnutim redoslijedom:

n=6

0	0	0	1	1	0

Prvo deklariramo novu varijablu „n_kopija „ na koju spremimo vrijednost n:

n_kopija=n

0	0	0	1	1	0

Posmaknemo n za jedan bit u desno, a n_kopija za jedan bit u lijevo:

n

0	0	0	0	1	1

n_kopija

0	0	1	1	0	0

Sada izdvojimo najmanje značajan bit od posmaknutog n, što je drugi najmanje značajan bit početnog n te ga spremimo na varijablu „maska“. U ovom slučaju maska

ce biti jednaka „1“. Tu jedinicu je potrebno dodati se na mjesto najmanje značajnog bita varijable“ n_kopija“. Pošto ce posmaknuta varijabla “ n_kopija“ na mjestu najmanje značajnog bita uvijek imati nulu, to se jednostavno može implementirati sa „ILI“ operatorom. Varijabla “ n_kopija“ ce nakon te operacije izgledati ovako:

0	0	1	1	0	1

Sada se cijeli niz operacija ponavlja. Permutacija ce biti gotova nakon N-1 ponavljanja, gdje je N broj bitova koji se permutiraju. U našem slučaju N=3, pa ce biti potrebna dva ponavljanja za dobivanje obrnutog poredaka bitova.

Međutim potreban je još jedan korak da bi dobili konačno rješenje. Ako pogledamo varijablu “ n_kopija“ nakon permutacije, ona će izgledati ovako:

0	1	1	0	1	1

Drugim riječima imati ce dekadsku vrijednost „27“. Prema tome potrebno je još maskirati sve bitove osim 3 najmanje značajna. U ovom slučaju vrijednost maska će imati vrijednost 7, dok će općenito imati vrijednost 2^N-1. Konačno dobivamo vrijednost permutiranog indeksa:

n_permutirano=3

0	0	0	0	1	1

Funkcija koja za ulazne parametre ima indeks člana ulaznog niza i broj bitova čije je znamenke potrebno obrnuti, u C jeziku izgleda ovako:

```
unsigned int permutacija( short n,  short br_bitova)
{
   unsigned int i, nrev,maska
   maska= 1<<br_bitova;         /
   nrev = n;
```

```
for(i=1; i<br_bitova; i++)
   {
      n >>= 1;
      nrev <<= 1;
      nrev |= n & 1;
   }
   nrev &= maska-1;
  return nrev;
}
```

Funkcija na prethodno opisani način okreće bitove indeksa i vraća vrijednost novog permutiranog indeksa. Nakon što smo to izračunali, potrebno je međusobno zamijeniti članove niza koji se nalaze na mjestu početnog i permutiranog indeksa. Sljedeći programski odsječak to izvršava:

```
for(z=1;z<n-1;z++){
  j=permutacija(z,korak);
   if(j>z){
          k++;
          pom=ul_real[z];
          ul_real[z]=ul_real[j];
          ul_real[j]=pom;
          pom=ul_imag[z];
          ul_imag[z]=ul_imag[j];
          ul_imag[j]=pom;

        }}
```

ul_real[z] i ul_imag[j] su nizovi koji sadrže realni i imaginarni dio signala čiji ćemo spektar računati i čije članove permutiramo.

3.2. Implementacija FFT-a na razvojnoj pločici TMDX320VC5505

Nakon što smo razmjestili članove niza na željene indekse možemo pristupiti izračunu FFT-a.

3.2.1. Frakcionalna aritmetika

Općenito u obradi signala najčešće se koriste procesori sa cjelobrojnom aritmetikom jer su jeftiniji, brži i jednostavniji. Pošto se kao rezultati proračuna spektra pojavljuju koeficijenti množenja koji nisu cijeli brojevi već decimalni, postavlja se pitanje kako realizirati potrebne matematičke operacije na cjelobrojnim procesorima. Ukoliko koeficijente skaliramo tako da im se vrijednosti nalaze između -1 i 1,možemo koristiti frakcijonalnu aritmetiku. Vezu između cjelobrojne i frakcijonalne aritmetike uspostavimo tako da cjelobrojnu B-bitnu vrijednost podijelimo sa brojem 2^B-1. U tablici 2. je prikazan primjer za slučaj kada je B=3. Zapis cjelobrojnih vrijednosti je u obliku dvojnog komplementa.

Tablica 2.

Cjelobrojna vrijednost	-4	-3	-2	-1	0	1	2	3
Frakcijonalna vrijednost	-1	-0.75	-0.5	-0.25	0	0.25	0.5	0.75

TMDX320VC5505 se zasniva na 16 bitnoj arhitekturi. Pa je tako najveći pozitivni broj koji je moguće zapisati 2^{16-1}-1=32767 i to će nam predstavljati frakcijonalnu vrijednost 1, dok je najveći negativni 2^{16-1}=32768 i to će predstavljati frakcijonalnu vrijednost -1.

Pogledajmo kako će izgledati množenje u frakcijonalnoj aritmetici. Pomnožimo brojeve 0.5 i 0.25. Za B=16 oni imaju cjelobrojne reprezentante 16384 i 8192.. Kao produkt dobiva se vrijednost 134 217 728 u registru širine 2*B*-1. Frakcijonalni pandan

ovog broja dobivamo tako da broj podijelimo s 2^{2B-2} što upravo daje vrijednost 0.5*0.25=0.125.

Nakon operacije množenja produkt širine 2*B*-1 moramo skratiti na istu širinu koju imaju i faktori (tj. *B* bita), te je potrebno od produkta uzeti najznačajnijih *B* bita. Tih *B* bita predstavljaju novu *B*-bitnu frakciju dobivenu kvantizacijom dugačke frakcije. Vrijednost u registru širine 2*B*-1 je 134 217 728 . Uzmimo najviših 16 bita 1000000000000000b što odgovara cijelom broju -4096, odnosno frakciji .0.125 Ukoliko izračunati broj ne postoji u 16-bitnim frakcijama uzimamo prvu manju vrijednost koja postoji.

3.2.2. Skaliranje

Na slici 4. Vidimo da se u svakom koraku zbrajaju se dva uzorka. Prema tome vrijednosti koeficijenata spektra proporcionalno rastu s brojem koraka. Postoji realna mogućnost da dođe do preljeva, tj. da im ukupna vrijednost prijeđe maksimalnu 16 bitnu vrijednost. Iz tog razloga potrebno ih je skalirati.

Skaliranje se često izvodi na način da, nakon što se izračunaju koeficijenti, sve podijeli sa ukupnim brojem uzoraka N. Međutim to nije uvijek najbolje rješenje jer se na taj način gubi na preciznosti te se smanjuje odnos signala i šuma (SNR -Signal to noise ratio). Bolji izbor je dinamičko skaliranje. Ono se izvodi na način da se u svakom koraku izračuna FFT-a provjerava da li je upravo izračunati koeficijent prešao najveću dozvoljenu vrijednost.

Postavimo najveću dozvoljenu vrijednost na 16 384 odnosno frakcionalno 0.5. Ukoliko koeficijent prijeđe tu vrijednost posmaknemo sve koeficijente za jedan bit u desno što odgovara dijeljenju sa 2. Time smo osigurali da niti jedan koeficijent nema frakcionalnu vrijednost veću od 0.5 i u sljedećem koraku neće doći do preljeva.

Ukoliko želimo koeficijente prikazati u logaritamskom mjerilu potrebno je nad njima napraviti jednostavnu operaciju logaritmiranja:

$$X(k)\_dB=20\log(X(k)) \qquad (16.)$$

Kako bi dobili što bolji SNR možemo deklarirati novu varijablu „e“ u kojoj ćemo pamtiti koliko smo puta koeficijente podijelili sa 2. Prilikom prebacivanja u logaritamsko mjerilo koeficijente pomnožimo sa 2^e:

$$X(k)\_dB=20\log(X(k)* 2^e) \quad (17.)$$

To se može raspisati :

$$X(k)\_dB=20\log(X(k)) + 20e\log(2)= 20\log(X(k)) + 6.02e \quad (18)$$

C kod koji implementira FFT algoritam opisan u poglavlju X uz frakcionalnu aritmetiku i dinamičko skaliranje :

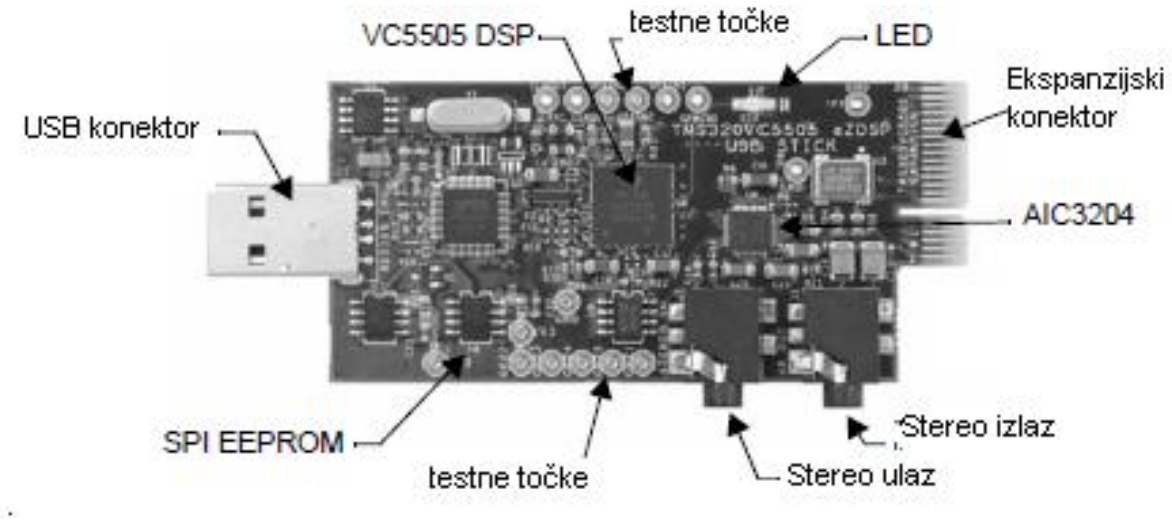
```
korak=round(log(N)/log(2));
n1 = 0;
n2 = 1;
for (i=0; i < korak; i++) {       // FFT
  n1 = n2;
  n2 *=2;
  a = 0;
f=0;
 for (j=0; j < n1; j++)  {
    c =round (cos(-2*pi*a/n2)*32767);
    s =round (sin(-2*pi*a/n2)*32767);
    a ++;

    for (k=j; k < n; k=k+n2) {
      pom1 =(long)c*ul_real[k+n1] - (long)s*ul_imag[k+n1];
      pom2 =(long)s*ul_real[k+n1] + (long)c*ul_imag[k+n1];
      pom1_16bit=pom1>>15;
      pom2_16bit=pom2>>15;
      ul_real[k+n1] = ul_real[k] - pom1_16bit;
      ul_imag[k+n1] = ul_imag[k] -pom2_16bit;
```

```
    ul_real[k] = ul_real[k] + pom1_16bit;
    ul_imag[k] = ul_imag[k] + pom2_16bit;
    max_doz=16382;
if ((ul_real[k+n1]>max_doz) || (ul_real[k]>max_doz)  || (ul_imag[k]>max_doz) ||
(ul_imag[k+n1]>max_doz) ){
                    f=1;
                    skaliranje++;

                    }  } }
   if(f){                          //skaliranje

  for(z=0;z<(N-1);z++){
                ul_real[z] = ul_real[z] >> 1;
                ul_imag[z] = ul_imag[z] >> 1;
                }   } }
j=0;
k=2;
while (j<N){
   izlaz[j]=(long)ul_real[j]*ul_real[j]+(long)ul_imag[j]*ul_imag[j];
   pom=sqrt(izlaz[j]);
   izlaz_dB[k]=round (20*( log(pom) / log(10)) );
   izlaz_dB[k+1]=-1*izlaz_dB[k];
    k+=2;
    j++;
return;
}
```

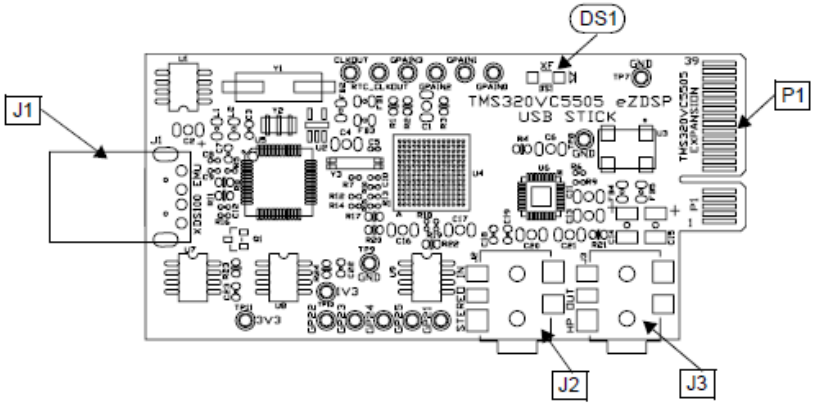
Spektar ćemo prikazivati na osciloskopu, pa nakon svakog pozitivnog koeficijenta ubacimo jedan negativni iste vrijednosti kako bi slika na osciloskopu bila stabilna i kako ne bi plesala gore dolje.

4. Razvojni sustav TMDX5505EZDSP

Signal čiji ćemo spektar računati čitati ćemo sa AD pretvornika audio kodeka, dok ćemo izračunati spektar slati na DA pretvornik, te ćemo ga prikazati na osciloskopu. Kako bi to uspješno mogli implementirati potrebno je razumjeti arhitekturu same razvojne pločice. Sustav se sastoji od procesora TMS320VC5505, audio kodeka TLP320AIC3204, upravljive LED diode, SPI EEPROM memorije od 512K-bit, USB XDS100 JTAG emulatora, ekspanzijskog konektora i testnih točaka. AD i DA konvertori su dio audio kodeka. Slika 7. pokazuje fizički izgled i razmještaj komponenti na pločici.



Slika 7. Razvojni sustav TMDX320VC5505



Slika 8. Raspored konektora

Sustav ima četiri konektora čije su funkcije prikazane u tablici 3.

Tablica 3.

Konektor	Funkcija	Strana pločice	Br. pinova
J1	USB	gornja	2
J2	Stereo ulaz	gornja	2
J3	Stereo izlaz	gornja	2
P1	Ekspanzija	gornja/donja	20x2

Sustav se napaja preko USB konektora i za rad nije potrebno eksterno napajanje.

J1, XDS100 USB konektor

Preko ovog konektora sustav se spaja na osobno računalo ili lapotop. Tablica 4. Prikazuje koji se signali nalaze na pojedinom pinu.

Tablica 4.

Pin	Signal
1	Napajanje, 5V_USB
2	D+
Pin	Signal
3	D-

4	masa
5	masa
6	masa

J2 konektor, stereo ulaz

Konektor služi za dovođenje signala na TLP320AIC3204 kodek.

Tablica 5.

Pin	Signal	Pin na kodeku
1	Masa	
2	AIC_LINE2L	15
3	AIC_LINE2R	16
4	nije spojeno	
5	nije spojeno	

J3 konektor, stereo izlaz

Konektor odvodi signal sa TLP320AIC3204 kodeka.

Tablica 6.

Pin	Signal	Pin na kodeku
1	Masa	
2	HEADPHONE_LOUT	25
3	HEADPHONE_ROUT	27
4	nije spojeno	
5	nije spojeno	

Na Shemi 1. u privitku su detaljno prikazani svi ulazni i izlazni signali iz kodeka. Sheme 2. I 3. prikazuju ulazne i izlazne signale iz procesora. Možemo uočiti kako su kodek i procesor povezani preko signala I2S0_CLK, I2S0_FS, I2S0_RX, I2S0_DX, GPIO_CODEC_RESET, I2C_SCL,I2C_SDA. U nastavku rada biti će pojašnjeno čemu služi pojedini od tih signala.

4.1 Procesor TMS320VC5505

Procesor TMS320VC5505 pripada porodici TMS320C5000 procesora za digitalnu obradu signala (DSP) tvrtke Texas Instruments. Namijenjen je aplikacijama sa malom potrošnjom energije.

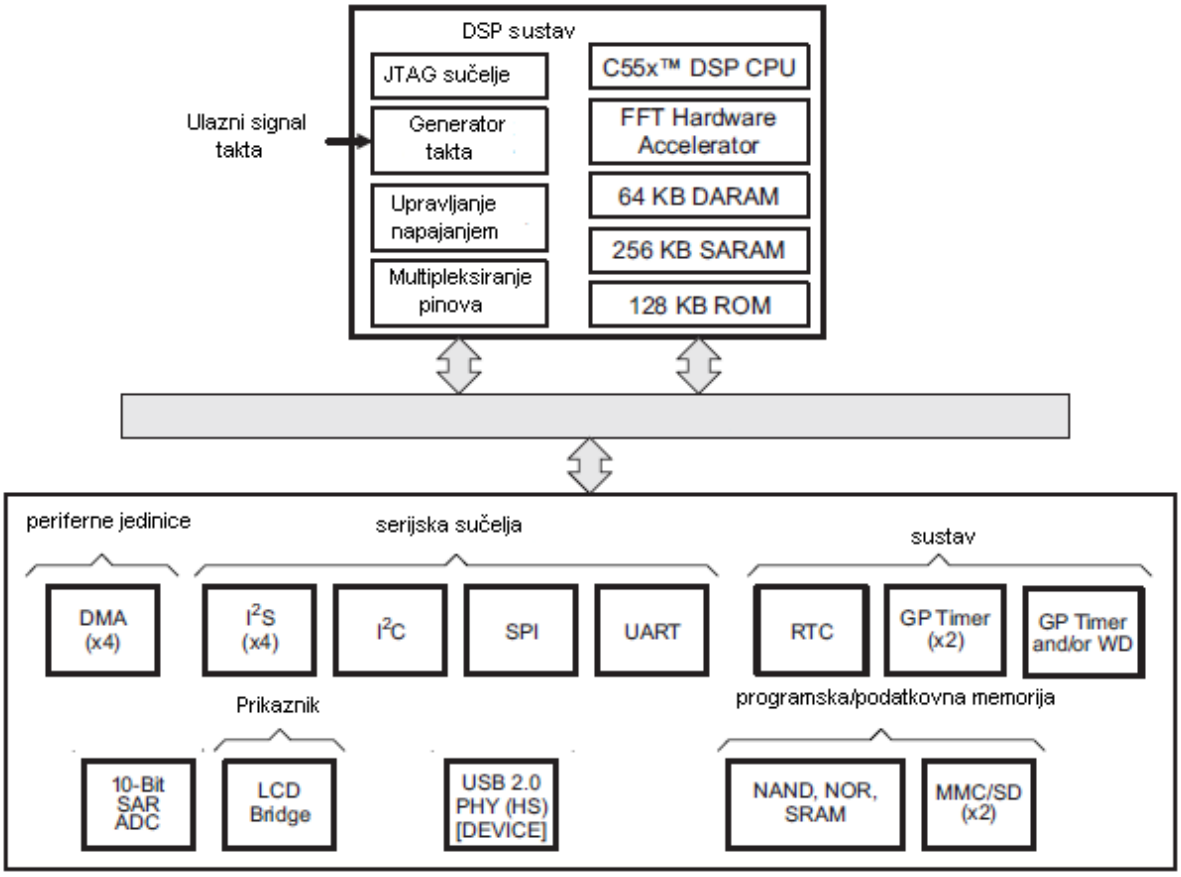
Temelji se na C55X arhitekturi, koja postiže visoke performanse uz vrlo malu potrošnju energije. Procesor ima jednu programsku sabirnicu, jednu 32 bitnu podatkovnu sabirnicu, četiri podatkovne 16 bitne sabirnice od čega su dvije za čitanje,a dvije za pisanje te dodatne sabirnice za periferne jedinice. Te sabirnice omogućuju procesoru čitanje četiriju i pisanje dvaju 16 bitnih podataka po ciklusu.

CPU je u mogućnosti izvršiti množenje 17 bita x 17 bita i jedno 32 bitno zbrajanje po ciklusu. Postoje dvije aritmetičko logičke jedinice, jedna 40 bitna i jedna pomoćna 16 bitna.

Serijski prijenos podataka omogućen sa dvije periferne jedinice za čitanje „Secure digital“ kartica(MMC/SD), četiri Inter-IC modula za prijenos zvuka (I2S sabirnica), serijskog priključka(SPI), I2C modula za prijenos podataka te univerzalnog asinkronog prijemnika/odašiljača(UART). Od perifernih jedinica snalazimo još AD pretvornik, GPIO jedinica za konfiguriranje pinova opće namjene, univerzalnu serijsku sabirnicu (USB), generator takta te memorijsko sučelje EMIF koje omoguće procesoru pristup asinkronim memorijama poput EPROM,NAND,NOR i SRAM. Na shemi 2. je prikazan raspored ovih perifernih jedinica po pinovima.

Procesor sadrži i FFT akcelerator koji omogućava izračun FFT-a veličine do 1024 uzorka.

Funkcijski blok dijagram prikazan je na slici 9.



Slika 9. Funkcijski blok dijagram procesora TMS320VC5505

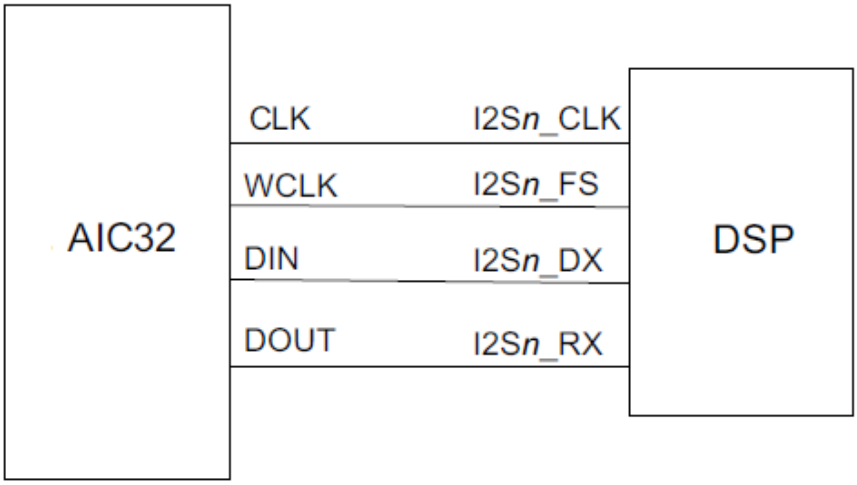
4.1.1 Modul za prijenos zvuka (I2S sabirnica)

Na shemama 1 i 2 u privitku vidimo da su procesor i kodek povezani preko I2C modula za prijenos podataka, I2S0 modula te preko GPIO [26] . Prvo pogledajmo koja je funkcija modula I2S0.

On služi kao sučelje preko kojeg se dovode signali (zvuk) sa AD pretvornika, te odvode sa procesora na DA pretvornik. TMS320VC5505 ima četiri takva modula I2S0,I2S1,I2S2 i I2S3. U našem slučaju AD i DA pretvorbu radi audio kodek TLP320AIC3204.

Postoje četiri signala preko kojih procesor komunicira s kodekom. To su: signal takta (I2S*n*_CLK), signal za sinkronizaciju blokova podataka koji se prenose

(I2S*n*_FS), signal kojim se odašilju podatci sa procesora (I2S*n*_DX) i signal kojim se primaju digitalni signali zvuka sa kodeka (I2S*n*_RX). (slika 10.)



Slika 10. Signali I2S

Ako je rad modula omogućen, prijenos podataka će započeti kada se pojavi odgovarajuća naponska razina na signalu za sinkronizaciju I2S*n*_FS. Podatci se iz registra za odašiljanje postavljaju na I2S*n*_DX počevši sa najznačajnijim bitom. Podatci sa I2S*n*_RX se posmiču u registar za prijem podataka na rastući ili padajući brid signala takta. Ako se prenosi dvokanalni stereo zvuk, najprije se odašilju i primaju podatci lijevog kanala, a zatim desnog. Kada se registar za prijam napuni odnosno kada se registar za odašiljanje isprazni, I2S postavlja zahtjev za prekid u registar I2SINTFL. Prekide je potrebno omogućiti upisom heksadekadske vrijednosti 3f u registar za omogućavanje prekida I2SINTMASK.

U tablici 8. Možemo vidjeti na kojim se adresama nalaze spomenuti registri modula I2S0.

Tablica. 8

Registar	Adresa	Opis
I2SSCTRL	2800h	Registar za odabir načina rada modula I2S0.

Registar	Adresa	Opis
I2SSRATE	2804h	Registar za upravljanje signalom takta
I2STXLT0	2808h	Slanje podataka lijevog kanala, registar 0
I2STXLT1	2809h	Slanje podataka lijevog kanala, registar 1
I2STXRT0	280Ch	Slanje podataka desnog kanala, registar 0
I2STXRT1	280Dh	Slanje podataka desnog kanala, registar 1
I2SINTFL	2810h	Registar u koji se postavlja zahtjev za prekid
I2SINTMASK	2814h	Registar za omogućavanje/onemogućavanje prekida
I2SRXLT0	2828h	Primanje podataka lijevog kanala, registar 0
I2SRXLT1	2829h	Primanje podataka lijevog kanala, registar 1
I2SRXRT0	282Ch	Primanje podataka desnog kanala, registar 0
I2SRXRT1	282Dh	Primanje podataka desnog kanala, registar 1

Pripadni C kod koji inicijalizira sve potrebne registre, I2S0 modul te razmjenjuje digitalne signale zvuka između procesora i kodeka:

```
#define I2SSCTRL      *(volatile ioport Uint16*)(0x2800)
#define I2SSRATE      *(volatile ioport Uint16*)(0x2804)
```

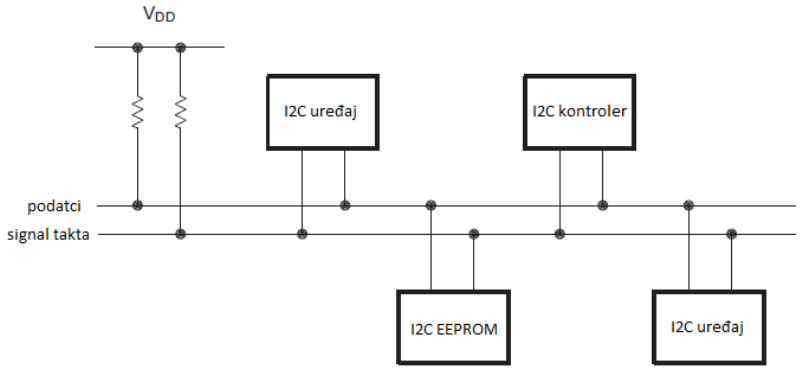
```
#define I2SINTFL       *(volatile ioport Uint16*)(0x2810)
#define I2SINTMASK  *(volatile ioport Uint16*)(0x2814)
#define I2STXLT1       *(volatile ioport Uint16*)(0x2809)
#define I2STXRT1       *(volatile ioport Uint16*)(0x280D)
#define I2SRXLT1       *(volatile ioport Uint16*)(0x2829)
#define I2SRXRT1       *(volatile ioport Uint16*)(0x282D)
 /* konfiguriranje I2S0 */
   I2SSRATE = 0x0;
   I2SSCTRL = 0x8010;    // 16-bit riječi, slave, omogući I2S
   I2SINTMASK = 0x3f;    // Omogući prekide

  while((0x08 & I2SINTFL) == 0);  // čekanje da se postavi prekid
    data1= I2SRXLT1;  // prijem 16 bitnih podataka (lijevi kanal)
    data2 = I2SRXRT1;  // prijem 16 bitnih podataka (desni kanal)

   while((0x20 & I2SINTFL) == 0); // // čekanje da se postavi prekid
     I2STXLT1= data3;  // odašiljanje 16 bitnih podataka (lijevi kanal)
     I2STXRT1= data4;  // odašiljanje 16 bitnih podataka (desni kanal)
```

4.1.2 I2C periferna jedinica

I2C jedinica služi kao sučelje preko kojeg procesor može razmjenjivati podatke sa nekom vanjskom jedinicom. Podatci koji se prenose su 8 bitni. Na razvojnoj pločici TMDX320VC5505 preko ovog modula su povezani procesor i kodek. Pošto je kodek potrebno konfigurirati, preko ovog modula ćemo prenositi podatke potrebne za njegovu konfiguraciju.

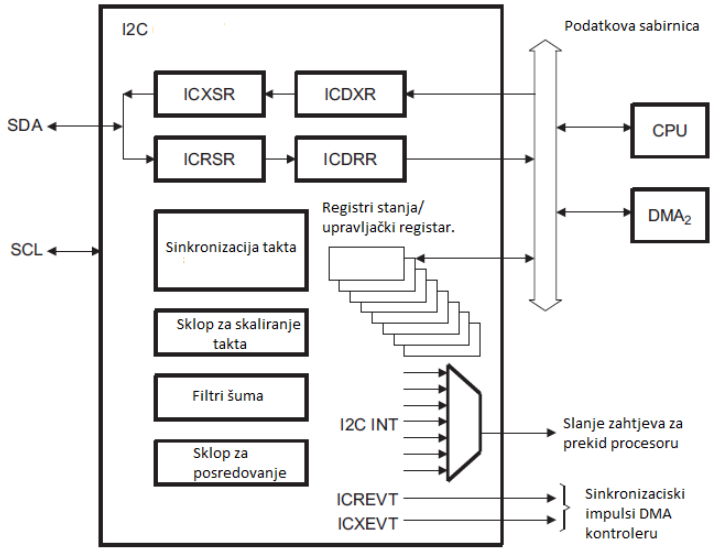
Sučelje se sastoji od jednog pina za prijem i odašiljanje podataka (SDA) i jednog pina preko kojeg se prenosi signal takta (SCL). Postoje registri u koje se privremeno spremaju vrijednosti prenošenih podataka, registar stanja i upravljački registar. S CPU-om ih povezuje zasebna sabirnica. Jedinica još ima sklopovlje za prilagodbu signala takta , filtar na svakom od pinovima koji smanjuje utjecaj šuma te

sklop koji posreduje pri razmijeni podataka ukoliko imamo više uređaja spojenih na I2C .



Slika 11. Spajanje više uređaja na I2C

Na slici 12. možemo vidjeti kako je I2C spojena na sabirnicu. Ako želimo odaslati neki podatak potrebno ga je upisati u registar ICDXR, a ako primamo podatke čitamo ih iz registra ICDRR. Kada I2C radi kao odašiljač, podatak upisan u ICDXR se kopira u registar ICXR nakon čega se posmiče na SDA pin. Ukoliko pak konfiguriramo jedinicu da radi kao prijemnik, podatci se sa SDA pina prvo upisuju u registar ICSR pa nakon toga kopiraju u ICDRR.



Slika 12. Funkcijski blok dijagram I2C periferne jedinice

Za razvojni sustav TMDX320VC5505 postoji posebna biblioteka sa funkcijama C jezika koje omogućuju jednostavnu primjenu I2C jedinice. Prototipi korištenih funkcija u sklopu ovog rada izgledaju ovako:

Int16 USBSTK5505_I2C_init( ) -funkcija inicijalizira I2C

Int16 USBSTK5505_I2C_write(Uint16 i2c_addr, Uint8 data, Uint16 len)*-funkcija za parametre prima 8 bitni podatak, adresu na koju se on odašilje te broj bajtova koje je potrebno poslati. Kao vrijednost vraća 0 ukoliko je podatak uspješno odaslan, a -1 ukoliko je došlo do pogreške.

Int16 USBSTK5505_I2C_read(Uint16 i2c_addr, Uint8 data, Uint16 len)* -funkcija za parametre prima 8 bitni podatak, adresu sa koje se on čita te broj bajtova koje je potrebno pročitati. Kao vrijednost vraća 0 ukoliko je podatak uspješno pročitan, a -1 ukoliko je došlo do pogreške.

4.1.3 Ulazi i izlazi opće namjene (GPIO)

GPIO (General purpose input/output) periferna je jedinica koja nam omogućuje da konfiguriramo pojedine pinove za opću namjenu (njih ukupno 32). Pinove možemo konfigurirati kao ulazne te čitati bitove sa nekog vanjskog registra ili uređaja, a možemo i konfigurirati kao izlazne pa slati bitove na vanjske uređaje. Na shemama 1 i 3. u privitku vidimo kako je na pin za resetiranje kodeka spojen GPIO[26] procesora. Na taj će pin biti potrebno poslati bit „1“ kako bi se kodek inicijalizirao.

O tome hoće li se neki pin koristiti kao ulaz ili kao izlaz određuje vrijednost upisana u odgovarajuće registre. 16 bitni registar IODR1 kontrolira pinove GPIO[0] do GPIO[15], a registar IODR2 pinove GPIO[16]-GPIO[31]. Kada su pinovi konfigurirani kao ulazni, primljene podatke sa pinova GPIO[0]-GPIO[15] čitamo iz registra IOINDATA1, a podatke s pinova GPIO[15]-GPIO[31] iz registra IOINDATA2.Na

jednak način šaljemo podatke u registre IOOUTDATA1 i IOOUTDATA2 u slučaju kada su pinovi konfigurirani kao izlazni. Tablica 9. Pokazuje adrese tih registara.

Tablica 9.

adresa	registar
1C06H	IODR1
1C07h	IODR2
1C08h	IOINDATA1
1C0Ah	IOINDATA2
1C0Bh	IOOUTDATA1
1C0Ch	IOOUTDATA2

Kao i kod I2C periferne jedinice postoji biblioteka sa funkcijama koje olakšavaju implementaciju u C jeziku:

Int16 USBSTK5505_GPIO_init ( )- inicijalizacija GPIO-a.

Int16 USBSTK5505_GPIO_setDirection (Uint16 pin, Uint16 smjer)- funkcija za parametre prima broj pina koji konfiguriramo te broj koji označava smjer (za „0“ pin se konfigurira kao izlazni, a za „1“ kao ulazni) .

Int16 USBSTK5505_GPIO_setOutput (Uint16 pin, Uint16 vrijednost_izlaza)-funkcija šalje zadanu vrijednost na željeni pin.

Int16 USBSTK5505_GPIO_getInput (Uint16 number)-funkcija prima broj koji označava pin s kojeg čitamo, a vraća vrijednost signala na tom pinu.

4.1.4 Konfiguriranje vanjske sabirnice

Na shemi 2.vidimo kako je na pojedine pinove procesora spojeno više perifernih jedinica Tako su na primjer na pin L10 spojeni signali GPIO[0], I2S0_CLK, MMC0_CLK. Kako bi odabrali koji će se od tih signala propustiti na vanjsku sabirnicu potrebno je izvršiti multipleksiranje. To se radi na način da se upiše odgovarajuća vrijednost u registar EBSR (External Bus Selection Register) koji se nalazi na adresi 0x1c00h. Nakon što se vrijednost upiše u registar, multipleksiranje pinova će se izvršiti u sljedećem ciklusu procesora. Registar je 16 bitni, a s kojim bitom se koji pin podešava pokazuje tablica 7. *Napomena: Oznake S0,S1 na shemi 2. predstavljaju serijske priključke, a oznake PD paralelne.*

Tablica 7.

Bit	Naziv	Opis
15	rezervirano	Moguće čitanje, pisanje nema učinka
14:12	PPMODE	Bitovi za odabir načina rada paralelnih priključaka 000=način 0, kroz 21 paralelni pin propuštaju se signali sa LCD modula. 001=način 1, 7 signala SPI modula, 6 GPIO signala, 4 signala UART modula,4 signala I2S2 modula se propuštaju kroz 21 paralelni pin. 010=način 2, 8 signala sa LCD modula i 8 GPIO signala se propuštaju na izlaz 011=način 3, propušta se 8 signala sa LCD modula,4 sa SPI modula, 4 sa I2S2 modula i 4 signala UART modula 100=način 4, propušta se 8 signala sa LCD modula 4 signala UART modula,4 signala I2S2 modula 101=način 5, propušta se 8 signala sa LCD modula, 4 sa SPI modula, 4 signala UART modula. 110=način 6, propušta se 7 signala sa SPI modula, 4 sa I2S2 modula, 4 sa I2S3 modula i 4 sa UART modula 111= rezervirano.

Bit	Naziv	Opis
11:10	SP1MODE	Bitovi za odabir načina rada serijskih priključaka S0 00=način 0,svih 6 signala sa MMC1 modula se propušta na 6 izlaznih signala 01=način 1,propuštaju se 4 signala sa I2S1 modula i 2 signala sa GPIO modula GPIO[11:10]. 10=način 2, propušta se 6 GPIO signala na izlaz. GPIO[11:6].. 11=rezervirano
9:8	SP0MODE	Bitovi za odabir načina rada serijskih priključaka S1 00=način 0,svih 6 signala sa MMC0 modula se propušta na 6 izlaznih signala 01=način 1,propuštaju se 4 signala sa I2S0 modula i 2 signala sa GPIO modula GPIO[5:4]. 10=način 2, propušta se 6 GPIO signala na izlaz GPIO[5:0]. 11=rezervirano
7	rezervirano	Moguće čitanje, pisanje nema učinka
6	rezervirano	Moguće čitanje, pisanje nema učinka
5	A20_MODE	Odabir načina rada pina A20 0=pin služi za adresiranje EMIF EM_A[20] 1=na pinu je signal sa GPIO modula GPIO[26]
4	A19_MODE	Odabir načina rada pina A19 0=pin služi za adresiranje EMIF EM_A[19] 1=na pinu je signal sa GPIO modula GPIO[25]
3	A18_MODE	Odabir načina rada pina A18 0=pin služi za adresiranje EMIF EM_A[18] 1=na pinu je signal sa GPIO modula GPIO[24]
2	A17_MODE	Odabir načina rada pina A17 0=pin služi za adresiranje EMIF EM_A[17] 1=na pinu je signal sa GPIO modula GPIO[23]
1	A16_MODE	Odabir načina rada pina A16 0=pin služi za adresiranje EMIF EM_A[16]

		1=na pinu je signal sa GPIO modula GPIO[22]
0	A15_MODE	Odabir načina rada pina A15 0=pin služi za adresiranje EMIF EM_A[15] 1=na pinu je signal sa GPIO modula GPIO[21]

Kako komunicira s procesorom preko modula I2S0 biti će potrebno multipleksirati pinove procesora na način da se signali I2S0 propuštaju na vanjsku sabirnicu. Prema tablici 7. možemo utvrditi kako će biti potrebno upisati heksadekadski broj 0x100h u registar EBSR.

U C jeziku najprije definiramo na kojoj je adresi registar EBSR naredbom:

```
#define EBSR       *(volatile ioport Uint16*)(0x1c00)
```

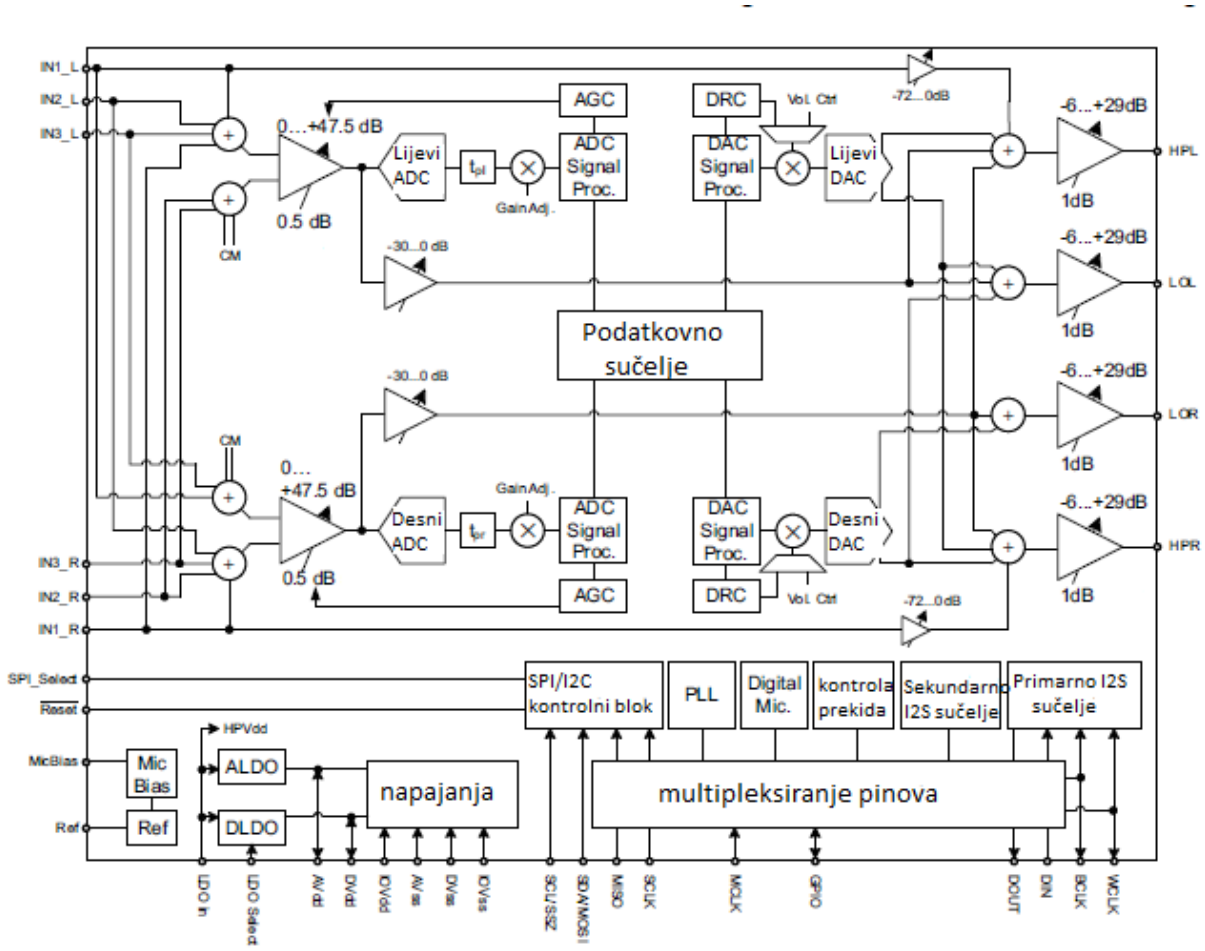
Zatim u njega upisujemo željenu vrijednost:

```
EBSR |= 0x010
```

4.2. Kodek TLP320AIC3204

TLP320AIC3204 je fleksibilni stereo kodek s niskom potrošnjom energije. Omogućuje konfiguraciju ulaznih i izlaznih kanala, kontrolu pojačanja, konfiguraciju signala takta. Iz tog se razloga kodek može prilagoditi aplikaciji za koju se koristi. Idealan je za baterijski pogonjene prijenosne audio (npr. mp3 player) i telefonske aplikacije. Područje rada mu je od 8kHz mono do 192kHz stereo signala.

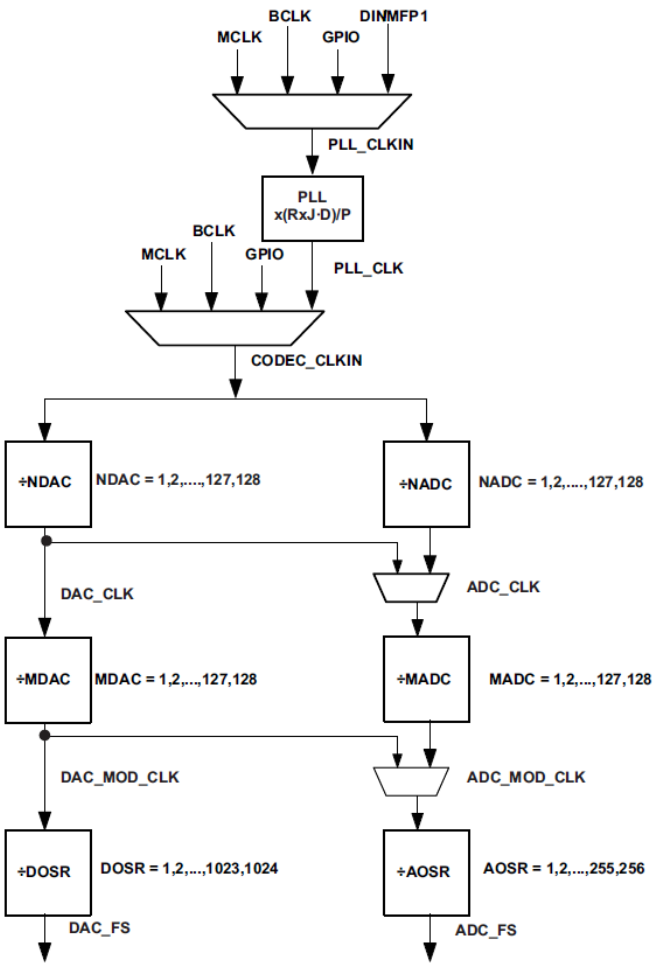
Pojednostavljeni funkcijski blok dijagram prikazan je na slici 13.



Slika 13. Funkcijski blok dijagram kodeka.

Na shemi 1. u privitku vidimo da ulazni signal primamo sa pinova IN2_L i IN2_R, pa ćemo njih propuštati na AD pretvornik. Izlazni signal treba ići iz DA pretvornika na izlaze HPR i HPL. Pojačala na izlaznim pinovima omogućuju pojačanje izlaznog signala za do 29dB.

Za ispravan rad kodeka potrebno je prilagoditi signal takta aplikaciji za koju ga koristimo. Slika 14. prikazuje funkcijski blok dijagram sklopovlja za prilagodbu signala takta koji se dovodi na AD i DA pretvornike.



Slika 14. Sklopovlje za prilagodbu signala takta

Multipleksiranjem se prvo odabire jedan od signala MCLK, BCLK, GPIO, DINMFP1. Za potrebe programa koji je u okviru ovog rada odabrati ćemo MCLK (glavni signal takta) te propustiti na PLL. PLL je sklop koji na svom izlazu generira signal koji je istoj fazi kao i signal kojeg prima na ulazu. Tada ponovo radimo multipleksiranje i propuštamo izlaz iz PLL-a. Sklopovi NDAC,NADC,MDAC,MADC, DOSR i AOSR nam služe kako bi podijelili signal takta željeni broj puta i tako mu namjestili frekvenciju.

Konfiguracija kodeka vrši se upisom vrijednosti u odgovarajuće registre. Kodek sadrži 38 stranica,a na svakoj do 128 registara. Pisanje podataka u registre raditi ćemo pomoću sljedeće funkcije:

```
Int16 AIC3204_rset( Uint16 broj_registra, Uint16 vrijednost ){
   Uint8 naredba[2];
   naredba[0] = broj_registra & 0x007F;       // 7-bitna adresa registra
   naredba[1] = vrijednost;                   // 8-bitni podatak

   return USBSTK5505_I2C_write(0x18, naredba, 2 );}
```

Moramo konfigurirati AD pretvornik s kojeg čitamo signal i DA pretvornik na koji šaljemo spektar. C kod za konfiguraciju kodeka:

```
   AIC3204_rset( 0, 0 );      // odaberi stranicu 0
   AIC3204_rset( 1, 1 );      // Reset kodeka
   AIC3204_rset( 0, 1 );      // odabir stranice 1
   AIC3204_rset( 1, 8 );      // Disable crude AVDD generation from DVDD
   AIC3204_rset( 2, 1 );      // Enable Analog Blocks, use LDO power
   AIC3204_rset( 0, 0 );
```

Konfiguracija takta :

```
   AIC3204_rset( 27, 0x0d );  // signal sa I2S0, 16 bitne riječi
   AIC3204_rset( 28, 0x00 );  //posmak podataka=0
   AIC3204_rset( 4, 3 );      //takt kodeka sa PLL, PLL dobiva takt sa MCLK
   AIC3204_rset( 6, 7 );      //PLL J dijeli ulazni takt sa 7
   AIC3204_rset( 7, 0x06 );   //postavka višeg bajta PLL D djelitelja =1680
   AIC3204_rset( 8, 0x90 );   //postavke nižeg bajta PLL D djelitelja =1680
   AIC3204_rset( 30, 0x88 );
   AIC3204_rset( 5, 0x91 );  //PLL R djelitelj=1, P djelitelj=1
   AIC3204_rset( 13, 0 );     //DOSR=128
   AIC3204_rset( 14, 0x80 );  //DOSR=128
   AIC3204_rset( 20, 0x80 );  //AOSR =128
   AIC3204_rset( 11, 0x82 );  //uključi NDAC djelitelj i postavi ga na 2
   AIC3204_rset( 12, 0x87 );  //uključi MDAC djelitelj i postavi ga na 7
   AIC3204_rset( 18, 0x87 );  //uključi NADC djelitelj i postavi ga na 7
   AIC3204_rset( 19, 0x82 );  //uključi MADC djelitelj i postavi ga na 2
```

Postavke DA pretvornika:

```
AIC3204_rset( 0, 1 );      // stranica 1
AIC3204_rset( 0x0c, 8 );   // signal s lijevog DAC na HPL
AIC3204_rset( 0x0d, 8 );   // signal desnog DAC na HPR
AIC3204_rset( 0, 0 );      // Stranica 0
AIC3204_rset( 64, 2 );     // glasnoca lijevog kanala=glasnoca desnog kanala
AIC3204_rset( 65, 0);      // lijevi DAC pojacanje 0dB
AIC3204_rset( 63, 0xd4 );  // lijevi DAC prima podatke s lijevog kanala audio
sučelja, desni sa desnog
AIC3204_rset( 0, 1 );      // stranica 1
AIC3204_rset( 0x10, 0x00 );// Unmute lijevi kanal , 0dB pojačanje
AIC3204_rset( 0x11, 0x00 );// Unmute desni kanal , 0dB pojačanje
AIC3204_rset( 9, 0x30 );   // uključi HPL i HPR
AIC3204_rset( 0, 0 );      // stranica  0
USBSTK5505_wait( 100 );
```

Postavke AD pretvornika:

```
AIC3204_rset( 0, 1 );      // stranica 1
AIC3204_rset( 0x34, 0x30 );// STEREO 1 Jack
                           // ulaz IN2_L ide na lijevi ADC
AIC3204_rset( 0x37, 0x30 );// ulaz IN2_R ide na desni ADC
AIC3204_rset( 0x36, 3 );
AIC3204_rset( 0x39, 0xc0 );
AIC3204_rset( 0x3b, 0 );   //pojačanje signala prije ulaska u lijevi ADC=0dB
AIC3204_rset( 0x3c, 0 );  //pojačanje signala prije ulaska u desni ADC=0dB
AIC3204_rset( 0, 0 );      // stranica 0
AIC3204_rset( 0x51, 0xc0 );// uključi lijevi i desni ADC
AIC3204_rset( 0x52, 0 );   // Unmute lijevog i desnog  ADC
```

Kada je i kodek konfiguriran aplikacija će moći uspješno izračunavati spektar signala i prikazivati ga na osciloskopu.

5. Zaključak

U ovom radu smo proučili brzu Fourierovu transformaciju i njena svojstva te je implementirali na konkretnom procesoru. Aplikacija je čitala signal sa AD pretvornika i preko DA pretvornika slala izračunati spektar na osciloskop. Opisali smo razvojni sustav TMDX5505EZDSP te njegove dijelove koje smo koristili.

Vidjeli smo da se FFT može računati sa različitim brojem ulaznih uzoraka. Što je veći broj uzoraka to se povećava preciznost izračuna spektra jer ulazni uzorci bolje reprezentiraju signal. Problem je u tome što povećanjem broja uzoraka raste vrijeme izvođenja programa. Ako trebamo računati spektar u realnom vremenu, veliko kašnjenje može stvarati probleme. U ovisnosti o tome treba li nam za neku primjenu preciznost ili brzina prilagoditi ćemo broj uzoraka s kojim računamo FFT.

Primjena FFT je vrlo široka. Područja primjene su primjerice kod analize glasova, analize vibracija mehaničkih konstrukcija, analize seizmograma, analize kardiograma i moždanih valova, određivanje prijenosne funkcije sustava, računanje korelacije i konvolucije, digitalno filtriranje, kodiranje audio i video signala. U moderno doba nalazi primjena je sve više (uglavnom računalnih). Isto tako, koliko FFT ima primjena toliko ima i raznih modifikacija i optimizacija, koje je, zbog svojstava diskretne Fourierove transformacijom, moguće napraviti na mnogo različitih načina.

6.Literatura

[1.]Profesor Branko Jeren, dipl. ing., Diskretna Fourierova transformacija DFT, http://www.fer.hr/_download/repository/sis11_Pred06.pdf, 24.52011.

[2.]Poljak S. Brza Fourierova transformacija i primjene. Diplomski rad. Sveučilište J. J. Strossmayera u Osijeku ,odjel za matematiku Sveučilišni nastavnički studij matematike i informatike

[3.]Fast Fourier transform, http://en.wikipedia.org/wiki/Fast_Fourier_transform, 26.5.2011

[4.]Fast Fourier Transform (FFT) FAQ, http://www.dspguru.com/dsp/faqs/fft, 24.5.2011

[5.]Fast Fourier Transform (FFT), http://cnx.org/content/m10250/latest/, 26.5.2011

[6.]Tomislav Petković, DOS- Brza Fourierova transformacija, http://dos.zesoi.fer.hr/vjezbe/pdf/dos_2005_avpr07.pdf, 27.5.2011

[7.]Prikaz brojeva u obliku frakcija, http://www.fer.hr/_download/repository/DOS0304_vj3.pdf, 27.5.2011

[8.]TMS320VC5505 eZdsp USB Stick, *Technical Reference,* http://support.spectrumdigital.com/boards/usbstk5505/revb/files/usbstk5505_TechRef_revb.pdf, 28.5.2011

[9.]TMS320VC5505 Fixed-Point Digital Signal Processor http://focus.ti.com/lit/ds/symlink/tms320vc5505.pdf, 30.5.2011

[10.]TMS320C5515/14/05/04/VC05/VC04 DSP Inter-Integrated Circuit (I2C) Peripheral, http://focus.ti.com/lit/ug/sprufo1a/sprufo1a.pdf, 30.5.2011.

[11.]TMS320C5515/14/05/04/VC05/VC04 DSP General-Purpose Input/Output (GPIO), http://focus.ti.com/lit/ug/sprufo4/sprufo4.pdf 30.5.2011

[12.]TMS320VC5505/5504 DSP Inter-IC Sound (I2S) Bus, http://focus.ti.com/lit/ug/sprufp4a/sprufp4a.pdf, 30.5.2011

[13.]TLV320AIC3204 Ultra Low Power Stereo Audio Codec, http://focus.ti.com/lit/ds/symlink/tlv320aic3204.pdf, 1.6.2011

7.Sažetak

Bilo koji vremensko promjenjivi signal može se konstruirati zbrajanjem sinusnih signala različite frekvencije, amplitude i faze. Fourierova analiza omogućuje prikaz nekog signala kao zbroj sinusnih signala odnosno kao zbroj više različitih frekvencijskih komponenti. Brza Fourierova transformacija je *divide-and-conquer* algoritam za brzo i efikasno izvođenje diskretne Fourierove transformacije. Na procesorima za digitalnu obradu signala izvodi se pomoću frakcionalne aritmetike.

Izračunati koeficijenti mogu poprimiti velike vrijednosti te je potrebno skaliranjem osigurati da ne prijeđu maksimalnu vrijednost koju procesor može zaprimiti. Najmanju pogrešku unosi dinamičko skaliranje. Također je potrebno obratiti pažnju na to da FFT algoritmi permutiraju koeficijente.

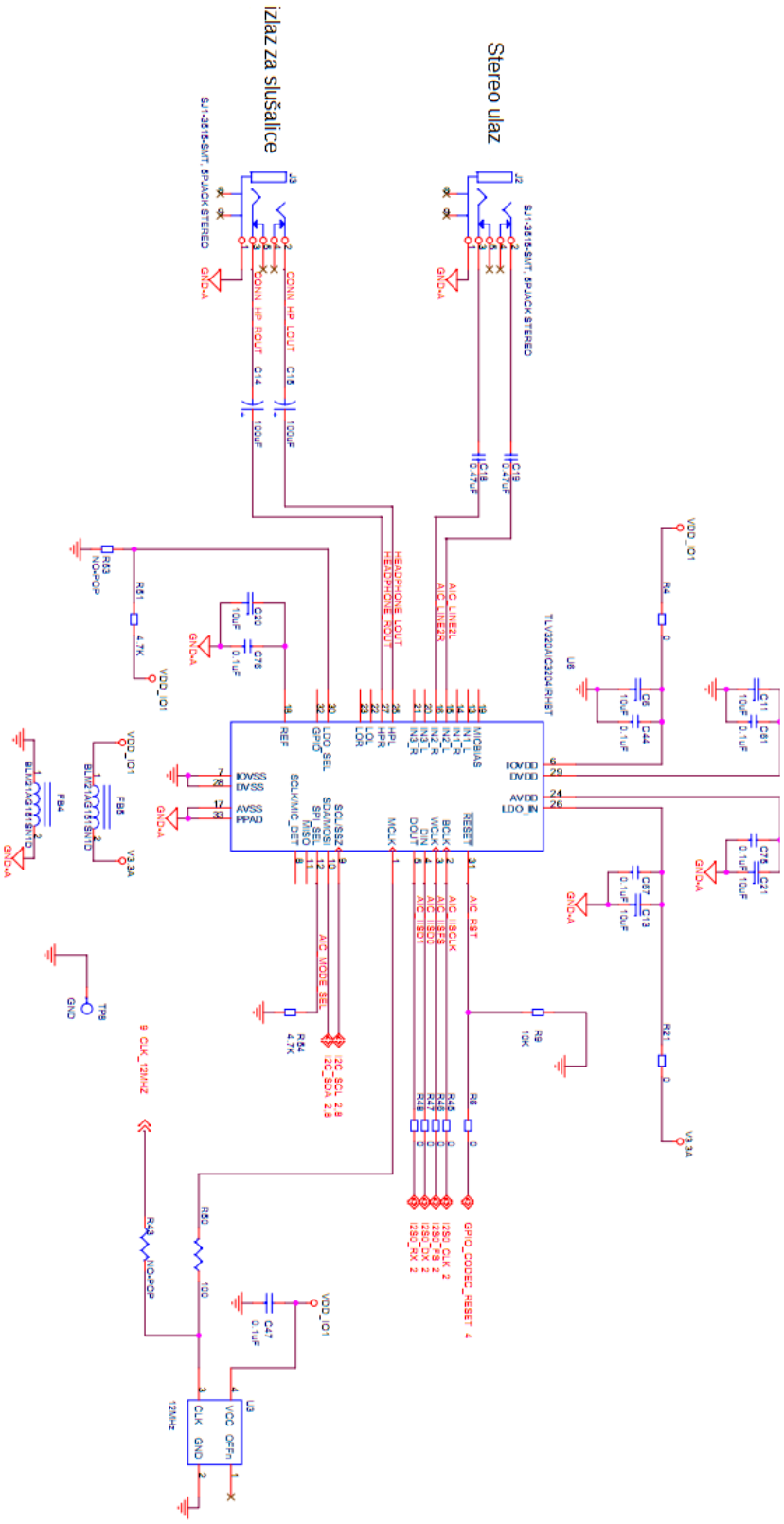
Razvojni sustav TMDX5505EZDSP se sastoji od procesora TMS320VC5505, kodeka TLP320AIC3204, upravljive LED diode, SPI EEPROM memorije od 512K-bit, USB XDS100 JTAG emulatora, ekspanzijskog konektora i testnih točaka.

Signal se čita sa AD pretvornika audio kodeka, a izračunati spektar se šalje na DA pretvornik preko kojeg je spojen osciloskop. Procesor dobiva signal sa kodeka preko I2S sabirnice i istom sabirnicom šalje natrag koeficijente FFT-a. S kodekom je još povezan preko I2C i GPIO modula preko kojih se postavljaju parametri kodeka.

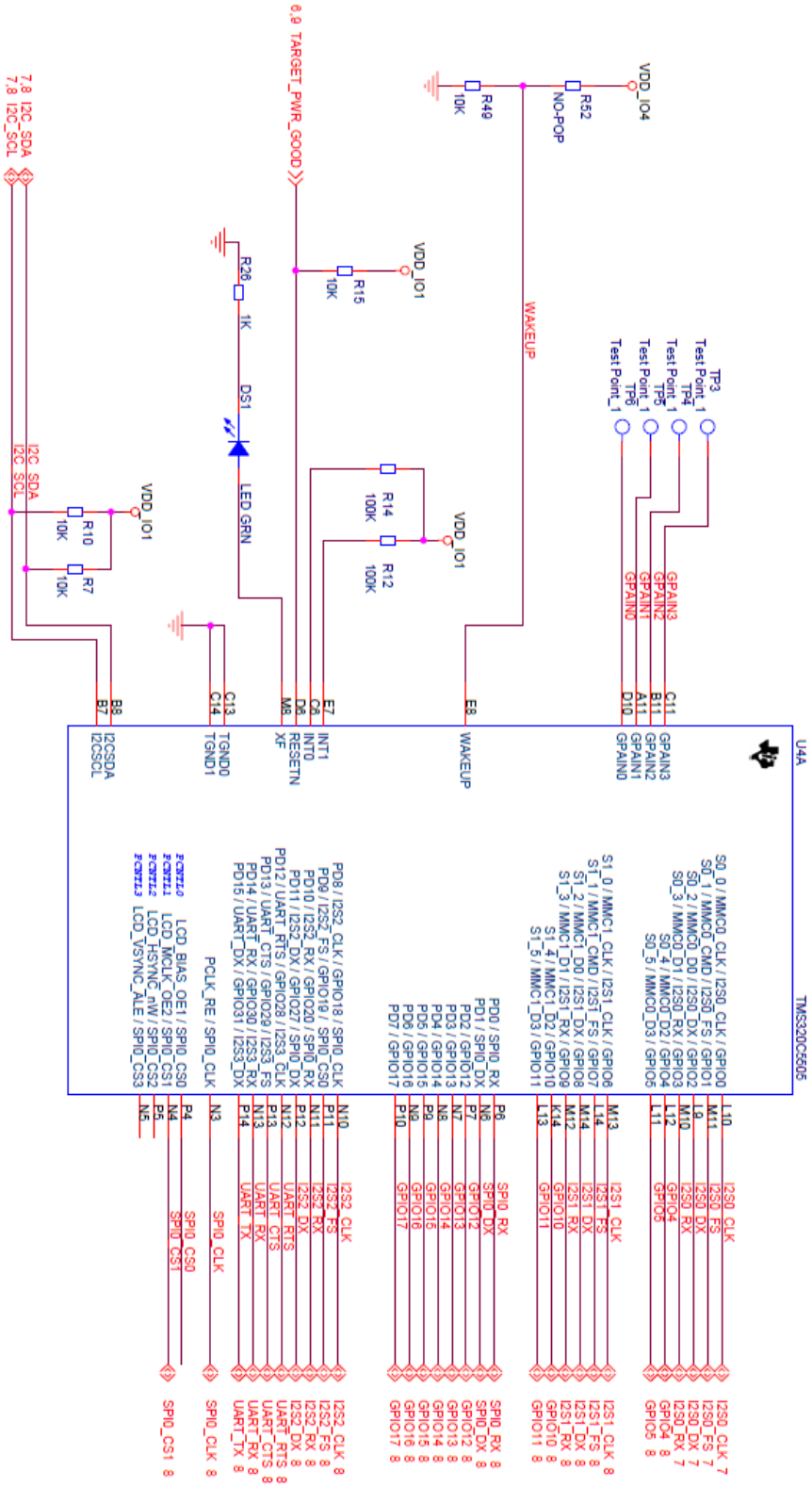
Kod FFT je bitno znati da veći broj uzoraka ulaznog signala povećava vrijeme izračuna koeficijenata, ali povećava preciznost. U ovisnosti tome treba li nam preciznost ili brzina prilagoditi ćemo broj uzoraka za izračun FFT-a.

8. PRIVITAK

Shema 1. Audio kodek TLP320AIC3204



Shema 2. procesor TMS320VC5505



Shema 3. procesor TMS320VC5505