PROJEKTNI ZADATAK 2

Predmet: Osnove Algoritama i Struktura

DSP 1

Student: Milan Kapetanović RA 184/2018

CILJ PROJEKTA

Cilj ovog projekta je bio da koristeći naš Encoder i Decoder uz Huffmanov Encoder i Decoder, izvršimo kompresiju zadatog audio signala.

Za ovaj projekat zadati audio signal bio je isečak iz pesme “Somebody” od pevača Bryan Adamsa.

Projekat se sastoji iz 2 kontrolne tačke:

- Kontrolna tačka 1
  U ovoj kontrolnoj tački za cilj smo imali da uradimo analizu i kvantizaciju u Encoderu do u Decoderu radimo vraćanje u opseg rekonstrukcijom i sintezu.

- Kontrolna tačka 2
  Ova kontrolna tačka se sastoji iz 3 zadatka u kojima se radi: Nelinearna obrada, Združeno kodovanje i poseban vid kvantizacije kada je parametar B jednak nuli.

Nakon što su kontrolne tačke ispunjene dobijamo kompresovane verzije audio signala od kojih su neke lossless a druge lossy kompresije.

Zadatak 1

- U ovom zadatku je bilo potrebno proširiti encoder.c i decoder.c tako da u enkoderu primenimo brzu Furijeovu transformaciju za prelazak u spektralni domen i odradimo kvantizaciju dobijenog spektra koristeći broj bita koji smo odredili u zaglavlju.

Za dekoderski deo je bilo potrebno da izvršimo vraćanje u opseg pozivajući funkciju reconstructB() i da izvršimo sintezu.

Enkoderski deo smo uradili na ovaj način:

```
//Analiza za levi kanal
for(i = 0; i < N; i++)
{
    fft_bufferL[i] = in_delayL[i];
    fft_bufferL[N+i] = inL[i];
    in_delayL[i] = inL[i];
}

for(i = 0; i < 2*N; i++)
{
    fft_bufferL[i] = _smpy(fft_bufferL[i], window[i]);
}

rfft(fft_bufferL, FFT_SIZE, SCALE);

//Kvantizacija za levi kanal
for(i =0; i < 2*N; i++)
{
    out[2*i] = quantB(fft_bufferL[i], BL);
}
```

Dok smo dekoderski deo uradili na ovaj način:

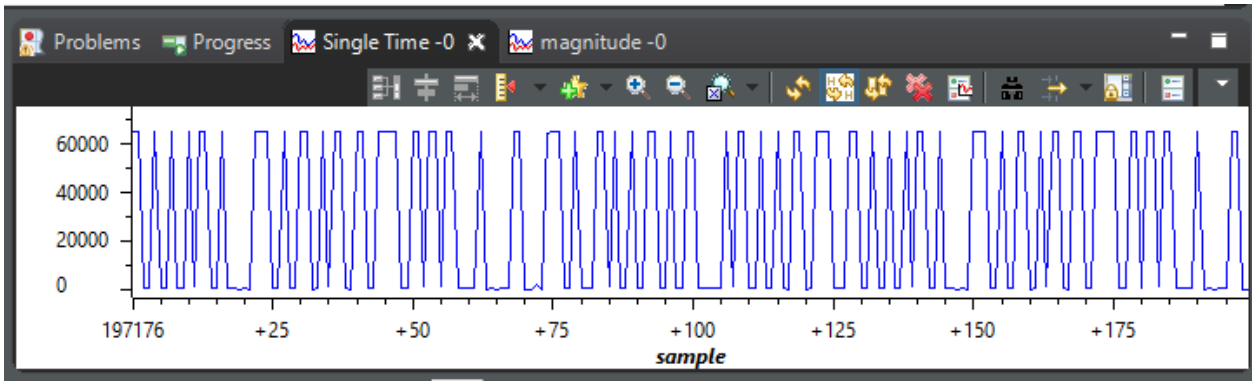
```
//Vracanje u opseg
for(i = 0; i < 2*N; i++)
{
    fft_bufferL[i] = reconstructB(in[2*i], BL);
}
//Sinteza
rifft(fft_bufferL, 2*N, NOSCALE);

for(i = 0; i < N; i++)
{
    outL[i] = _smpy(fft_bufferL[i], window[i]) + _smpy(out_delayL[i], window[i + N]);
    outL[i] *= 4;
    out_delayL[i] = fft_bufferL[N+i];
}
```

Prvi zadatak je rađen za vrednosti BL i BR: 16, 10 i 8.

Nakon kompresije za vrednosti 16 kvalitet zvuka je nepromenjen dok za vrednost 10 možemo da primetimo krčenje i šum a za vrednost 8 zvuk se čuje “kao iz bunara” i čuje se pucanje zvuka.

Nakon kvantizacije u enkoderu za vrednosti BL i BR 10 bita dobijamo ovakav izgled FFT buffera:



Zadatak 2

- U ovom zadatku smo za cilj imali da u enkoderu uradimo nelinearnu obradu po formuli:

$X = sign(X) * \sqrt{X}$

Sign predstavlja funkciju koja vraća vrednost 1 ukoliko je X pozitivan ili -1 ukoliko je X negativan broj.

Takođe smo morali i da modifikujemo dekoder tako da odradi inverznu operaciju nelinearne obrade odmah nakon vraćanja u opseg.

U enkoderu smo to uradili na ovaj način:

```
//Zadatak 2
for(i = 0; i < 2*N; i++)
{
    if (fft_bufferL[i] > 0)
    {
        sign1[i] = 1;
    }
    else if(fft_bufferL[i] < 0)
    {
        sign1[i] = -1;
    }
    else sign1[i] = 0;
}

sqrt_16(x, r, nx);

for(i = 0; i < 2*N; i++)
{
    fft_bufferL[i] = sign1[i] * fft_bufferL[i];
}
```

Dok smo u dekoderu to ovako uradili:

```
//Zadatak2
//Za desni kanal
for (i = 0; i < 2*N; i++)
{
    fft_bufferR[i] = sign(fft_bufferR[i]) * _smpy(fft_bufferR[i], fft_bufferR[i]);
}
```

Funkcija sign izgleda ovako:

```
int sign(Int16 x)
{
    if (x > 0)
    {
        return 1;
    }
    else if (x < 0)
    {
        return -1;
    }
    else return 0;
}
```

Nakon implementacije smo radili kompresiju za vrednosti B 10 i 8 bita i rezultati su:

Datoteka	quantBL	quantBR	MS	SK*	Komentar*
stream1.wav	10	10	0	2.06	Cuje se dosta bolje nego u prvom zadatku.
stream1.wav	8	8	0	2.99	Cuje se isto kao i za 10 bita.

Takođe treba napomenuti da SK predstavlja Stepen Kompresije koji se dobija tako što nakon izlaza iz HuffmanEnc dobijamo .dsp1 i .dict fajlove čiju veličinu sabiramo i delimo je od originalne veličine ulaznog fajla.

Zadatak 3

Za ovaj zadatak smo trebali da proširimo koder tako da omogući združeno kodovanje. Združeno kodovanje se omogućuje ubacivanjem dva nova kanala M i S i formulom:

$$M(i) = \frac{InputBufferL(i) + inputBufferR(i)}{2}$$

$$S(i) = \frac{InputBufferL(i) - inputBufferR(i)}{2}$$

To smo radili implementacijom ovog koda u main enkodera:

```
for(j = 0; j < AUDIO_IO_SIZE; j++)
{
    Int16 M = (InputBufferL[j] + InputBufferR[j])>>2;
    Int16 S = (InputBufferL[j] - InputBufferR[j])>>2;
    InputBufferL[j] = M;
    InputBufferR[j] = S;
}
```

Nakon toga je bilo potrebno proširiti i dekoder koji bi trebao da rekonstruiše MS u LR na osnovu formule:

$$InputBufferL(i) = M(i) + S(i)$$

$$InputBufferR(i) = M(i) - S(i)$$

U dekoderu se ta formula obradila na sledeći način:

```
//Zadatak 3
for(i = 0; i < 2*N; i++)
{
    M[i] = fft_bufferL[i];
    S[i] = fft_bufferR[i];
    fft_bufferL[i] = M[i] + S[i];
    fft_bufferR[i] = M[i] - S[i];
}
```

Nakon implementacije smo trebali da popunimo tabelu koja izgleda ovako:

Datoteka	quantBL	quantBR	MS	SK*
stream1.wav	16	16	1	1.83
stream1.wav	12	8	1	3.95
stream1.wav	11	6	1	8.04

I potrebno je bilo komentarisati svaku vrednost za B:

- BL = 16 BR = 16: "Potpun" zvuk ali jako tih, čuje se kao iz daljine pogotovo glas i čuje se blago "pucanje" zvuka.
- BL = 12 BR = 8: Šušti, tiho je, dešavaju se "pucanja" zvuka.
- BL = 11 BR = 6: JAKO šušti i "puca" zvuk, osećaj je kao da su zvučnici pri kraju svog života.

Zadatak 4

- Zadatak 4 je imao za cilj da se omogući poseban vid kvantizacije koji se aktivira kada se funkciji encode prosledi parametar B = 0.

- Kada se to desi potrebno je izvršiti kvantizaciju na sledeći način:
    - Opseg 0-100Hz – anulirati
    - Opseg 100Hz - 512Hz – kvantizovati sa 8 bita
    - Opseg 512Hz - 4096Hz – kvantizovati sa 12 bita
    - Opseg 4096Hz - 14336Hz – kvantizovati sa 6 bita
    - Sve iznad 14336Hz anulirati

- U enkoderu se to ovako implementira:

```
if(BL == 0)
{
    //Zadatak 4
    for(i = 0; i < 2*N; i++)
    {
        if(i <= ogranicenje1)
        {
            out[2*i] = 0;
        }else if(i <= ogranicenje2)
        {
            out[2*i] = quantB(fft_bufferL[i], 8);
        }else if(i <= ogranicenje3)
        {
            out[2*i] = quantB(fft_bufferL[i], 12);
        }else if(i <= ogranicenje4)
        {
            out[2*i] = quantB(fft_bufferL[i], 6);
        }else
        {
            out[2*i] = 0;
        }
    }
}else
{
    //Kvantizacija za levi kanal
    for(i =0; i < 2*N; i++)
    {
        out[2*i] = quantB(fft_bufferL[i], BL);
    }

}
```

- Gde smo ograničenja računali na ovaj način:

```
int ogranicenje1 = (256/48000) * 100;
int ogranicenje2 = (256/48000) * 512;
int ogranicenje3 = (256/48000) * 4096;
int ogranicenje4 = (256/48000) * 14336;
```

- 256 predstavlja broj elemenata u nizu, 48000 predstavlja sampling rate i množimo sa frekvencijom koja nam je zadata i tako dolazimo do pozicije komponente te frekvencije u nizu.

- Nakon toga je bilo potrebno i proširiti dekoder tako da na isti način vrši vraćanje vrednosti u opseg ako je B = 0.

- Takva funkcionalnost u dekoderu izgleda ovako:

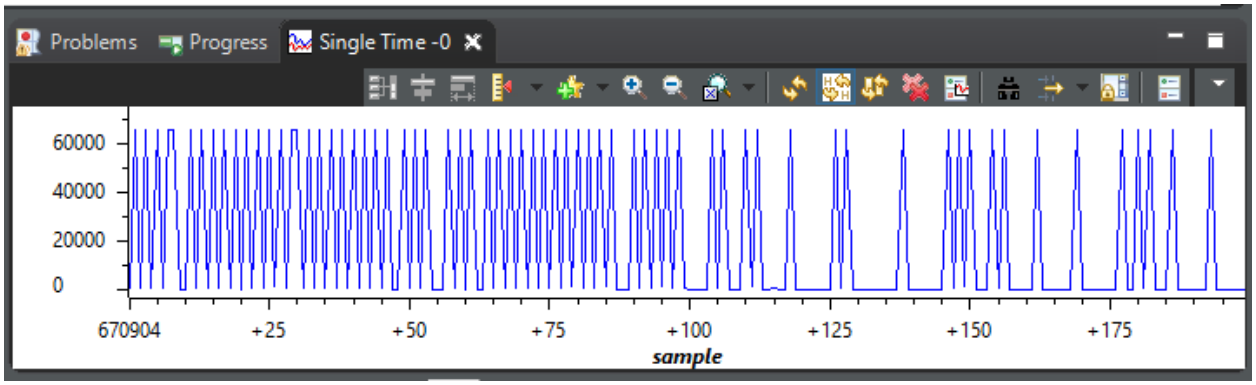
```
int ogranicenje1 = (256/48000) * 100;
int ogranicenje2 = (256/48000) * 512;
int ogranicenje3 = (256/48000) * 4096;
int ogranicenje4 = (256/48000) * 14336;

if(BL == 0)
{
    //Zadatak 4
    for(i = 0; i < 2*N; i++)
    {
        if(i <= ogranicenje1)
        {
            fft_bufferL[i] = 0;
        }else if(i <= ogranicenje2)
        {
            fft_bufferL[i] = reconstructB(fft_bufferL[i], 8);
        }else if(i <= ogranicenje3)
        {
            fft_bufferL[i] = reconstructB(fft_bufferL[i], 12);
        }else if(i <= ogranicenje4)
        {
            fft_bufferL[i] = reconstructB(fft_bufferL[i], 6);
        }else
        {
            fft_bufferL[i] = 0;
        }
    }
}else
{
    //Vracanje u opseg
    for(i = 0; i < 2*N; i++)
    {
        fft_bufferL[i] = reconstructB(in[2*i], BL);
    }
}
```

Nakon toga je bilo potrebno popuniti tabelu:

Datoteka	quantBL	quantBR	MS	SK*	Komentar*
stream1.wav	0	0	0	948.5	Nema zvuka uopste jer je sve podeseno na 0.
stream1.wav	0	8	1	5.69	Zvuk jako susti i glas se jako lose cuje.

Takođe je bilo potrebno prikazati izgled fft buffera nakon poziva rfft funkcije:



I nakon kvantizacije za B = 0: