DIGITALNI PROCESORI SIGNALA – Implementacija brze Furijeove transormacije na DSP platformi

Vladimir Petrović, Aleksandra Lekić

Univerzitet u Beogradu – Elektrotehnički fakultet

2018/2019

Furijeov red

- Periodični signal $x(t) = x(t + T_0)$ sa periodom T_0 se može razviti u red

$$x(t) = \sum_{k=-\infty}^{+\infty} C_K \, \mathrm{e}^{\mathrm{j}\, k\omega_0 t} \, .$$

- Koeficijenti razvoja u kompleksni red se računaju kao

$$C_k = \frac{1}{T_0} \int_{(T_0)} x(t) \, \mathrm{e}^{-\mathrm{j}\, k\omega_0 t} \, .$$

- Ovakav signal ima u frekvencijskom domenu spektralne komponente na umnošcima osnovne učestanosti $k\, f_0$.

Furijeova transformacija

- Aperiodični signal $x(t)$ se može odrediti kao

$$x(t) = \frac{1}{2\pi} \int_{\omega=-\infty}^{+\infty} X(\omega)\ \mathrm{e}^{\mathrm{j}\,\omega t}\ d\omega.$$

- Furijeova transformacija aperiodičnog signala je

$$X(\omega) = \int_{t=-\infty}^{+\infty} x(t)\ \mathrm{e}^{-\mathrm{j}\,\omega t}\ dt.$$

Discrete-Time Fourier Transform

- Furijeova transformacija signala $x[n] = x(nT)$ je

$$X(\Omega) = \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}\,\Omega n},$$

$$X(F) = \sum_{n=-\infty}^{+\infty} x[n]\, \mathrm{e}^{-\mathrm{j}\,2\pi F}\,.$$

- Kako je $F = \frac{\Omega}{2\pi} = \frac{f}{f_S}$, spektar je periodičan sa periodom 2π.

Discrete Fourier Transform

- Diskretni signal $x[n]$ ima N nenultih odbiraka i važi $x[n] = 0$ za $n \notin [0,\, N-1]$.
- DFT je data sa

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

- Spektralne komponente se nalaze na učestanostima $\Omega_k = \frac{2\pi k}{N}$.

DFT osobine

- Linearnost

$$DFT[a\,x[n]+b\,y[n]] = a\,DFT[x[n]]+b\,DFT[y[n]] = a\,X(k)+b\,Y(k)$$

- Konjugovano-kompleksna spektralna komponenta

$$X(-k) = X^*(k)$$

Za parno N je $M = \frac{N}{2}$ i $X(M+k) = X^*(M-k)$, a za neparno $M = \frac{N-1}{2}$ je $X(M+k) = X^*(M-k+1)$.

DFT osobine

- Z-transformacija

$$X(k) = X(z)\big|_{z=\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}k}}$$

- Cirkularna konvolucija

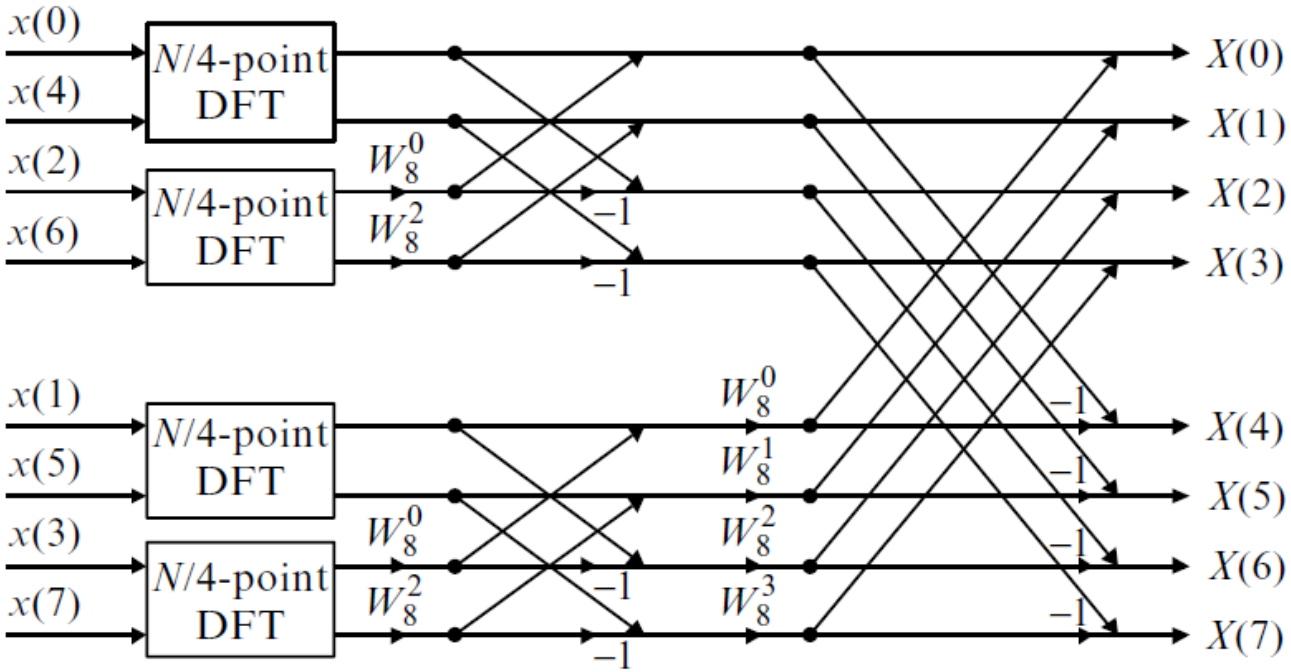
$$y[n] = h[n] \otimes x[n] = \sum_{m=0}^{N-1} h[m]\, x[(n-m) \mod N]$$

Fast Fourier Transform

- DFT zahteva N^2 kompleksnih množenja i $N^2 - N$ kompleksnih sabiranja.
- Kako je $W_N^{kn} = W_N^{(kn) \mod N}$ za $W_N^{kn} = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}k}$, broj operacija se može smanjiti na $N \log_2 N$.

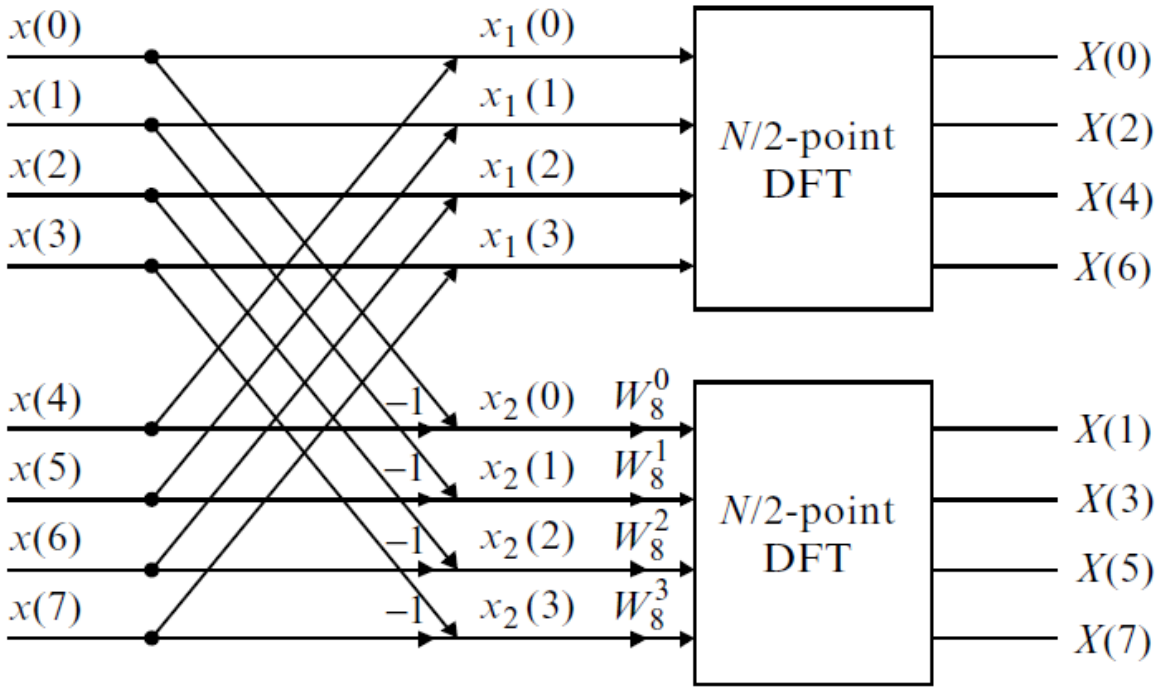
Preuređivanje u vremenu

Dele se odbirci na pola i računa se $N/2$ DFT, zatim $N/4$ DFT, ... , dok se ne dobije kombinacija dva elementa.



Preuređivanje u frekvencijskom domenu

Dele se odbirci na pola i računa se $N/2$ DFT. Deljenje se nastavlja dok se u poslednjem stepenu ne dobije DFT sa dva elementa.



Primeri

1. *DFT Using Floating-Point C*
2. *DFT Using the C55xx Assembly Program*
3. *FFT Using Floating-Point C*
4. *FFT Using the C55xx Hardware Accelerator*

DFT Using Floating-Point C

Fajl	Opis
`float_dft128Test.c`	program za testiranje *floating-point* DFT-a
`float_dft128.c`	C funkcija za *floating-point* DFT sa 128 tačaka
`float_mag128.c`	C funkcija za računanje amplitudske karakteristike
`float_dft128.h`	C header fajl
`tistdtypes.h`	header fajl standardnih tipova podataka
`c5505.cmd`	linker fajl
`input.dat`	ulazni fajl

- Množenje kompleksnog signala $x[n] = x_r[n] + \mathrm{j}\, x_i[n]$ sa *twiddle* faktorom $W_N^{kn} = \cos\left(\frac{2\pi kn}{N}\right) - \mathrm{j}\, \sin\left(\frac{2\pi kn}{N}\right) = W_r - \mathrm{j}\, W_i$ je dato sa

$$x[n]\, W_N^{kn} = x_r[n]\, W_r + x_i[n]\, W_i + \mathrm{j}\, (x_i[n]\, W_r - x_r[n]\, W_i).$$

- Prethodna jednačina se može zapisati u formatu $X = X_r + \mathrm{j}\, X_i$.

$$X_r = x_r[n]\, W_r + x_i[n]\, W_i \qquad X_i = x_i[n]\, W_r - x_r[n]\, W_i$$

U programu se

- koriste nizovi `Xin[2*N]` i `Xout[2*N]` za smeštanje kompleksnog ulaznog i izlaznog niza odbiraka.
- računaju se *twiddle* faktori u toku izvršavanja.
- za realni ulazni signal se konstruiše kompleksni signal tako što se imaginarni deo izjednači sa nulom.
- računa se amplitudska karakteristika u 128 tačaka signala datog u datoteci `input.dat`.

```c
#include <math.h>
#include "tistdtypes.h"
#include "float_dft128.h"

#define PI 3.1415926536

void float_dft_128(Int16 Xin[], Int16 Xout[])
{
  Int16 i,n,k,j;
  float angle;
  float Xr[N],Xi[N];
  float W[2];

  for (i=0, k=0;k<N;k++)
  {
      Xr[k]=0;
      Xi[k]=0;
      for(j=0,n=0;n<N;n++)
      {
        angle =(2.0*PI*k*n)/N; // Calculate twiddle factor angle
        W[0]=cos(angle); // Compute twiddle complex factor
        W[1]=sin(angle);
        // Multiple complex data by complex twiddle factor
        // The input integer data is convered to floating-point by /32676.0
        Xr[k]=Xr[k]+(float)Xin[j]*W[0]/32767.0+(float)Xin[j+1]/32767.0*W[1];
        Xi[k]=Xi[k]+(float)Xin[j+1]*W[0]/32767.0-(float)Xin[j]/32767.0*W[1];
        j+=2; // Advance data pointer
      }
      // Save the real and the imaginary part of the DFT result with rounding
      // Floating-point data is converted to integer by *32767.0
      Xout[i++] = (Int16)(Xr[k]*32767.0 + 0.5);
      Xout[i++] = (Int16)(Xi[k]*32767.0 + 0.5);
   }
}
```

Zadatak

- Pokrenuti program. Izmeriti vreme izvršavanja funkcija `float_dft_128` i `float_mag_128`.
- Prikazati amplitudsku karakteristiku izlaznog signala.

Prikaz dijagrama

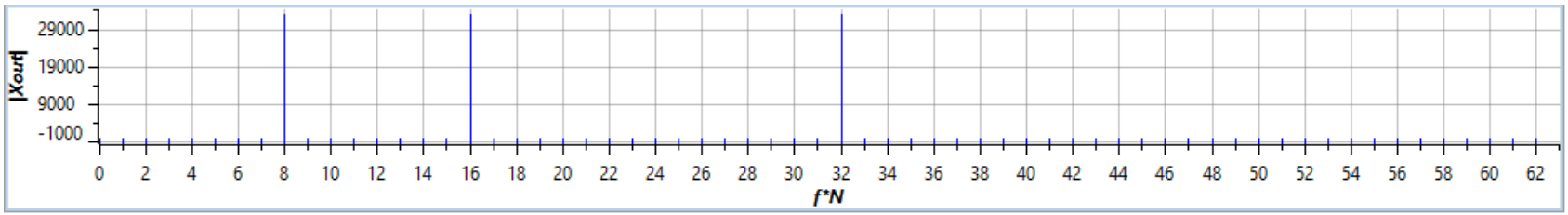
- `Tools` → `Graph` → `Single Time`
- Podesiti:

  `Acquisition Buffer Size` 64

  `Dsp Data Type` 16-bit signed integer

  `Starting Address` spectrum

  `Display Data Size` 64



Rezultati

- Vide se tri spektralne komponente na normalizovanim učestanostima 0.125, 0.25 i 0.5.
- Funkcija `float_dft_128` se izvršava 326 082 276 taktnih intervala.
- Funkcija `float_mag_128` se izvršava 121 531 taktnih intervala.

DFT Using the C55xx Assembly Program

Fajl	Opis
`asm_dft128Test.c`	program za testiranje DFT-a
`dft_128.asm`	asemblerska funkcija DFT sa 128 tačaka
`mag_128.asm`	asemblerska funkcija za računanje amplitudske karakteristike
`sine_cos.asm`	asemblerska funkcija za računanje *twiddle* faktora
`asm_dft128.h`	C header fajl
`tistdtypes.h`	header fajl standardnih tipova podataka
`c5505.cmd`	linker fajl
`input.dat`	ulazni fajl

```
_dft_128:
    mov XAR1, dbl(*(#Xout)) ; Save output array address
    mov XAR0, dbl(*(#Xin))  ; Save input array address
    mov #0, *(#k)      ; k = 0;
    mov #N-1,BRC0   ; Repeat counter for outter loop N times
    mov #N-1,BRC1   ; Repeat counter for inner loop N times

    rptb outer_loop-1   ; for(i=0, k=0;k<N;k++) {
    mov #0, AC0
    mov AC0, dbl(*(#Xr))    ; Xr = 0;
    mov AC0, dbl(*(#Xi))    ; Xi = 0;
    mov AC0, *(#n)     ; n = 0;
    mov dbl(*(#Xin)), XAR3  ; AR3 pinter to input array

    rptb inner_loop-1   ; for(j=0,n=0;n<N;n++) {
    bclr ST1_FRCT
    mov *(#n), T1
    mpym *(#k), T1, AC0   ; k*n
    and #(N-1), AC0   ; m = (k*n)&0x127;
    sftl AC0, #9    ; 2*pi/N*m = 0x7fff/128*m=512*m
    amov #W, XAR0    ; Pointer to W[]

    mov AC0, T0
|| call _sine_cos    ; Compute W[]

    bset ST1_FRCT
    amov #W, XAR0    ; Pointer to W[]

    bset ST1_SATD
|| mpym *AR3+, *AR0+, AC0
    macm *AR3-, *AR0, AC0
    mpym *AR3+, *AR0-, AC1
|| sfts AC0, #-16

    add dbl(*(#Xr)), AC0
    masm *AR3+, *AR0, AC1
    sfts AC1, #-16
    mov AC0, dbl(*(#Xr))  ; Xr += Xin[j]*W[0]+Xin[j+1]*W[1];

    add dbl(*(#Xi)), AC1
    mov AC1, dbl(*(#Xi))  ; Xi += Xin[j+1]*W[0]-Xin[j]*W[1];
    bclr ST1_SATD
    add #1, *(#n)
inner_loop

    mov dbl(*(#Xout)), XAR2
    mov AC0, *AR2+   ; Xout[j++] = Xr;
    mov AC1, *AR2+   ; Xout[j++] = Xi;
    mov XAR2, dbl(*(#Xout))
    add #1, *(#k)
outer_loop
```

U programu se

- računa *twiddle* faktor za ugao

$$angle = \frac{2\pi}{N}\, kn,$$

gde je 16-bitna fiksna predstava π vrednost `0x7FFF`.

- `angle = (2 * 32767 / N) * k * n`

Rezultati

- Funkcija `dft_128` se izvršava 1 132 822 taktnih intervala.
- Funkcija `mag_128` se izvršava 217 taktnih intervala.

FFT Using Floating-Point C

Fajl	Opis
`float_fftTest.c`	program za testiranje *floating-point* FFT-a
`fft_float.c`	C funkcija za *floating-point* FFT
`fbit_rev.c`	C funkcija za *bit reversal*
`float_fft.h`	C header fajl
`fcomplex.h`	C header za definiciju *floating-point* kompleksni tip podataka
`tistdtypes.h`	header fajl standardnih tipova podataka
`c5505.cmd`	linker fajl
`input_f.dat`	ulazni fajl

- Računa se FFT funkcijom `fft` čiji je prvi argument kompleksni ulazni niz, a drugi argument $\log_2 N$ (za N broj tačaka računanja FFT-a), a treći niz *twiddle* faktora.
- Funkcijom `bit_rev` se menja redosled odbiraka signala prema sledećoj tabeli.

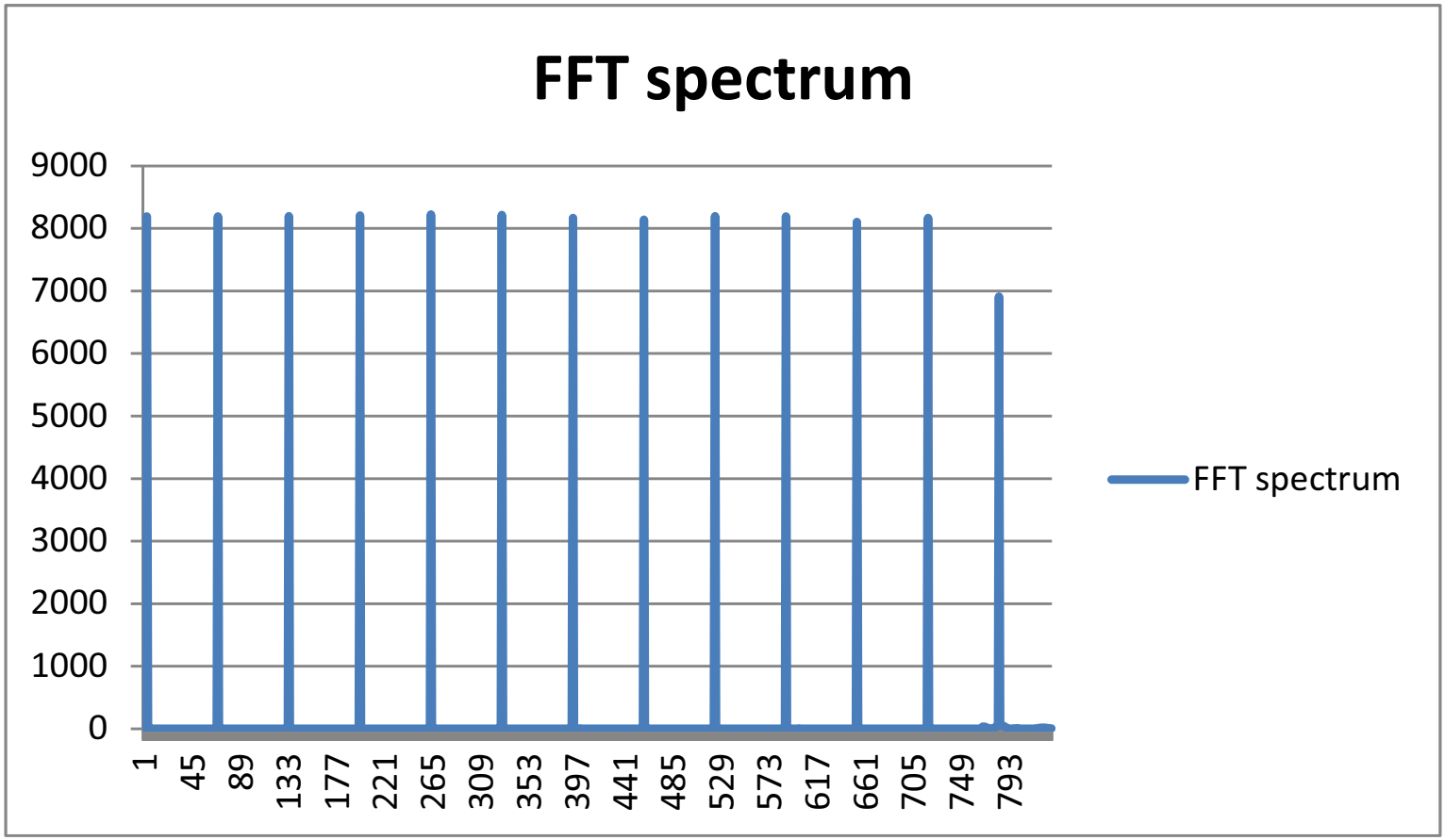
Input sample index		Bit-reversed sample index	
Decimal	Binary	Binary	Decimal
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

```c
#include "tistdtypes.h"
#include "fcomplex.h" /* Floating-point complex.h header file */

void fft(complex *X, Uint16 EXP, complex *W, Uint16 iFlag, Uint16 sFlag)
{
    complex temp; /* Temporary storage of complex variable */
    complex U; /* Twiddle factor W^k */
    Uint16 i,j;
    Uint16 id; /* Index for lower point in butterfly */
    Uint16 L; /* FFT stage */
    Uint16 LE; /* Number of points in sub DFT at stage L and offset to next DFT in stage */
    Uint16 LE1; /* Number of butterflies in one DFT at stage L. Also is offset to lower point in butterfly at stage L */
    float scale;
    Uint16 N=1<<EXP;/* Number of points for FFT */

    if (sFlag == 1) scale = 1.0; /* NOSCALE_FLAG=1 */
    else scale = 0.5;/* SCALE_FLAG = 0, Scaling of 0.5 at each stage */

    if (iFlag == 1) scale = 1.0; /* FFT_FLAG=0, IFFT_FLAG=1, Without scaling for IFFT */

    for (L=1; L<=EXP; L++)  /* FFT of length 2^EXP */
    {
        LE=1<<L;  /* LE=2^L=points of sub DFT */
        LE1=LE>>1;  /* Number of butterflies in sub-DFT */
        U.re = 1.0;
        U.im = 0.0;

        for (j=0; j<LE1;j++)
        {
            for(i=j; i<N; i+=LE) /* Butterfly computations */
            {
                id=i+LE1;
                temp.re = (X[id].re*U.re - X[id].im*U.im)*scale;
                temp.im = (X[id].im*U.re + X[id].re*U.im)*scale;

                X[id].re = X[i].re*scale - temp.re;
                X[id].im = X[i].im*scale - temp.im;

                X[i].re = X[i].re*scale + temp.re;
                X[i].im = X[i].im*scale + temp.im;
            }

            /* Recursive compute W^k as U*W^(k-1) */
            temp.re = U.re*W[L-1].re - U.im*W[L-1].im;
            U.im = U.re*W[L-1].im + U.im*W[L-1].re;
            U.re = temp.re;
        }
    }
}
```

Rezultati

- Ulazni signal je *sweeping tone* (sinusoida promenljive učestanosti sa vremenom).
- Svaki frejm sadrži 64 odbirka. Zato se prikazom `spectrum` signala može prikazivati frejm po frejm.
- Funkcija `fft` se izvršava 1 220 523 taktnih intervala, a `bit_rev` 8498 taktnih intervala po jednom frejmu.

Rezultati


FFT spectrum
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1
45
89
133
177
221
265
309
353
397
441
485
529
573
617
661
705
749
793
FFT spectrum

FFT Using the C55xx Hardware Accelerator

Fajl	Opis
`hwfftTest_.c`	program za testiranje hardverskog FFT-a
`tistdtypes.h`	header fajl standardnih tipova podataka
`c5505.cmd`	linker fajl
`input.dat`	ulazni fajl

- C55xx procesor ima *built-in* hardverski modul za efikasno računanje FFT-a.
- Podržan je 8-, 16-, 32-, 64-, 128-, 256-, 512-, 1024-bitni FFT osnove-2 (*radix-2*).
- Za računanje kompleksnog FFT-a se koriste paralelni podaci dobijeni preko sve tri magistrale podataka: B, C i D.
- Hardverski FFT modul sadrži kompleksni *twiddle* faktor u 1024 tačke u vidu *lookup* tabele.

FFT funkcija

`hwafft_NUMpts(long* in, long* out, short fftFlag, short scaleFlag)`

Za 1024-FFT je ime funkcije `hwafft_1024pts`.

`in` - kompleksni ulazni podatak, predstavljen kao niz koji sadrži realni i imaginarni deo date jedan za drugim
`out` - kompleksni izlazni niz istog formata kao `in`
`fftFlag` - ima vrednost 0 za FFT i 1 za IFFT
`scaleFlag` - ako je nula, skalira se svaki ulazni odbirak sa 0.5

Bit-reversal funkcija

```
hwafft_br(long* in, long* out, short len)
```

`in` - kompleksni ulazni podatak, predstavljen kao niz koji sadrži realni i imaginarni deo date jedan za drugim
`out` - kompleksni izlazni niz istog formata kao `in`
`len` - veličina niza

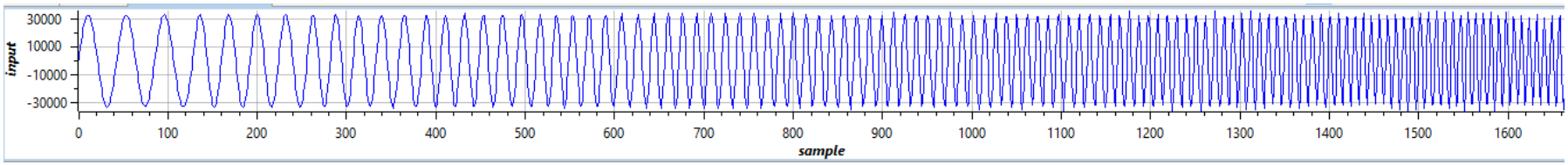
- Kompleksni podaci moraju da budu dati u vidu dve 16-bitne reči.
- Za ispravno korišćenje `hwafft_br` funkcije, izlazni niz treba da bude na memorijskoj lokaciji koja ima bar $\log_2(4\,N)$ binarnih nula. (za 1024-FFT mora biti bar 12 nula - lokacije tipa `0x1000`).
- Poravnanje memorije se postiže pragmom `DATA_ALIGN`.
- Da bi se koristile ove funkcije, moraju se dodati na kraj linkerskog fajla.

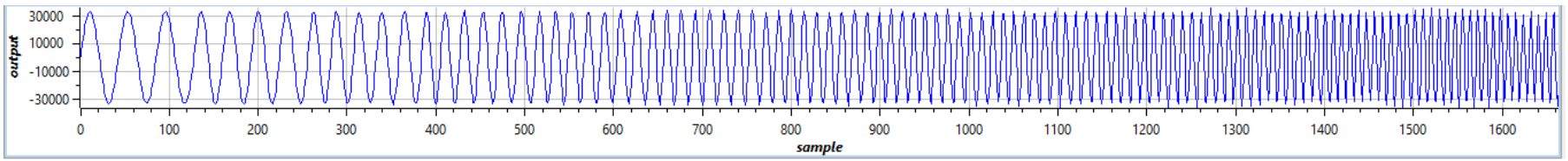
```
_hwafft_br = 0x00ff6cd6;
_hwafft_1024pts = 0x00ff7a56;
```

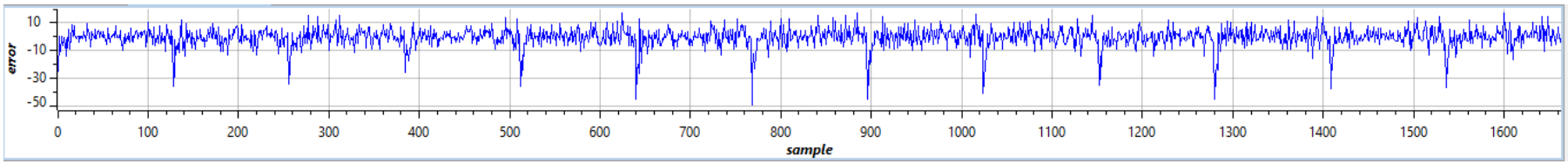
Program

računa FFT ulaznog signala, a zatim na to primenjuje IFFT i dobija se izlazni signal. Nakon određivanja izlaznog signala računa se greška oduzimanjem.

Rezultati


input
30000
10000
-10000
-30000
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
sample


output
30000
10000
-10000
-30000
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
sample


error
10
-10
-30
-50
0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
sample

KRAJ!