Zastosowania Procesorów Sygnałowych

Implementacja algorytmu Radix-2 FFT w procesorze ADSP-21161N

Piotr Matuszczak

1. Założenia projektu

Głównym założeniem była implementacja algorytmu Radix-2 FFT w procesorze sygnałowym ADSP-21161N i zbadanie wpływu skalowania liczb na dokładność obliczeń. W tym celu powstały dwa programy, które realizują ten algorytm. Programy te znajdują się w osobnych folderach. Oba programy zostały w całości napisane w asemblerze. Ich zasadniczą częścią jest (dla obu ta sama nazwa) plik *FFT.asm* . Zawiera on procedury konieczne do realizacji algorytmu FFT. W folderze *FFT_asm* znajduje się projekt z programem operującym na liczbach stałoprzecinkowych w formacie fractional 1.31 . Natomiast w folderze *FFT_asm_float* znajduje się program operujący na liczbach zmiennoprzecinkowych. W niniejszym opisie zostanie zawarte porównanie wyżej wymienionych rozwiązań.

2. Algorytm Radix-2 FFT

W tym punkcie zostanie pokrótce opisany algorytm FFT DIT (Decmation In Time), który został zastosowany w projekcie. Opis ograniczy się do minimum teorii niezbędnej z punktu widzenia analizy programów.

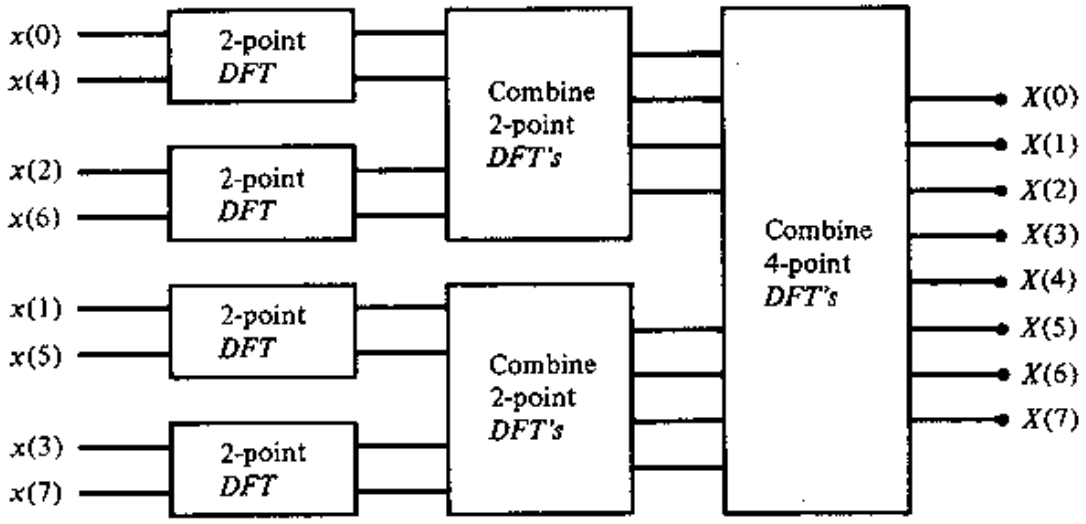
Dyskretna transformata Fouriera definiowana jest zależnością:

$$x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X(k) W_N^{-nk}, \qquad 0 \le n \le N-1$$

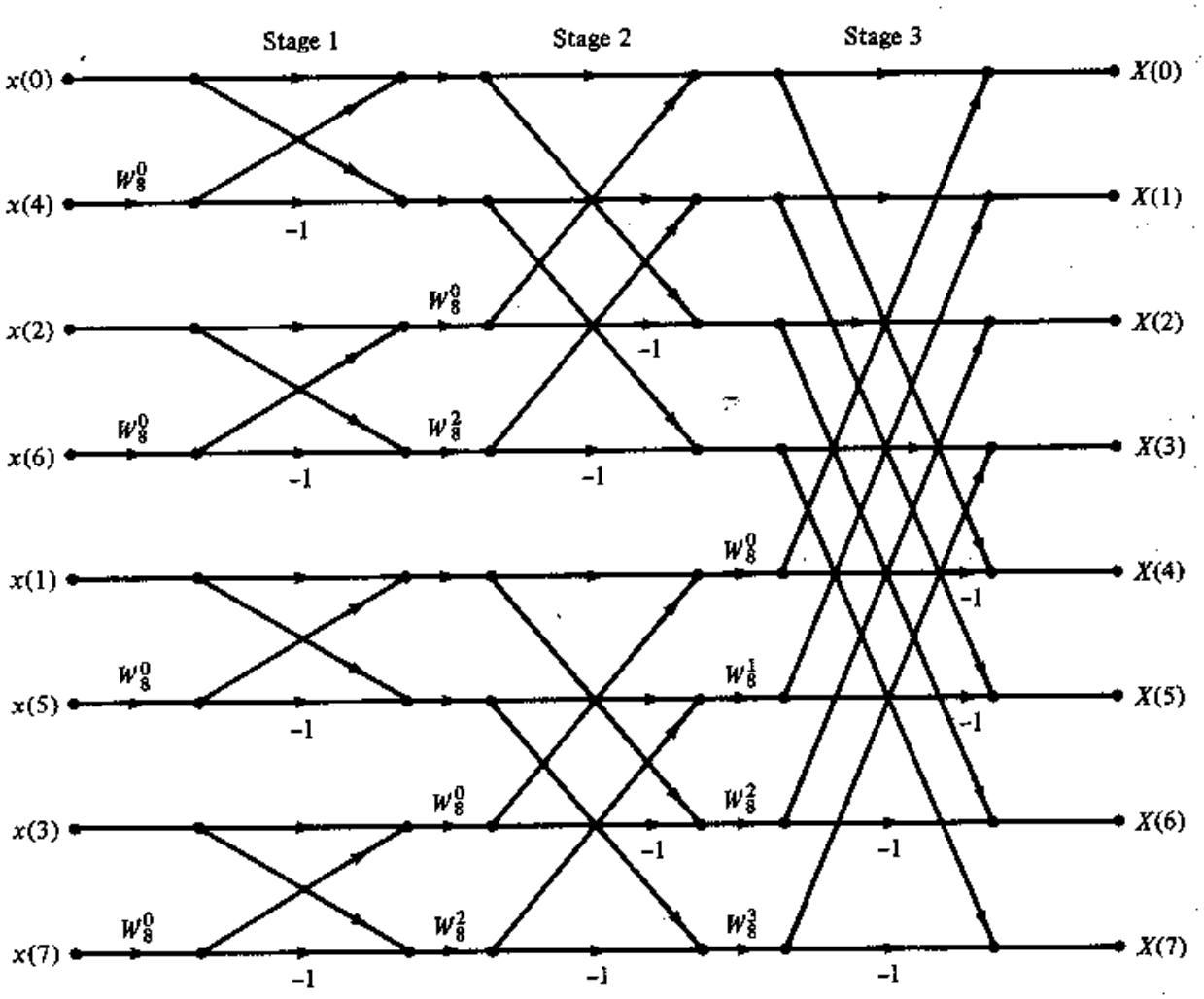
Wykorzystując własności funkcji sinus i cosinus możemy napisać:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad k = 0,1,\ldots,N-1 \\ &= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn} \\ &= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)} \end{aligned}$$

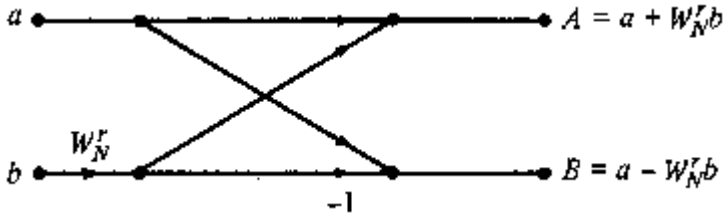
Rozdzieliwszy próbki na parzyste i nieparzyste możemy kontynuować proces rozdzielania aż dostaniemy zbiory po 2 próbki. Poniżej pokazana jest graficznie ośmiopunktowa transformata FFT DIT.



Dalej można to rozrysować za pomocą motylków:



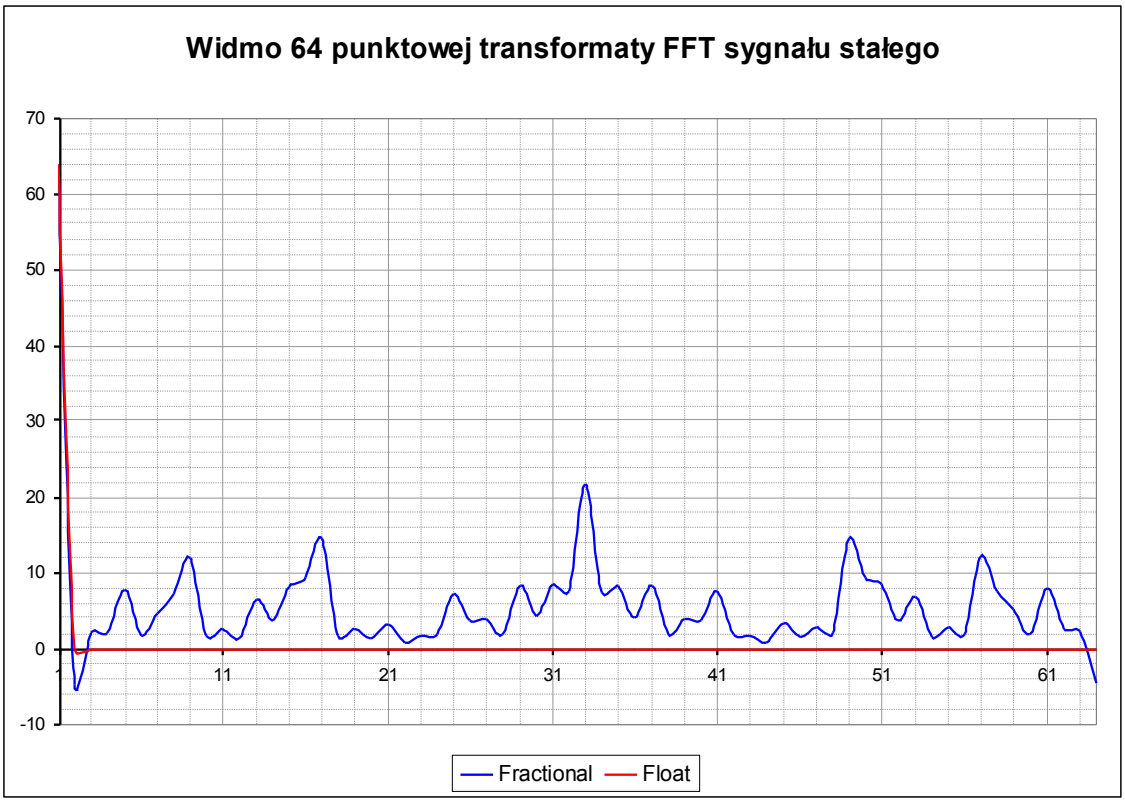
Dobrze widać iteracyjny charakter algorytmu. Poniżej znajduje się narysowany pojedynczy motylek dla FFT DIT.

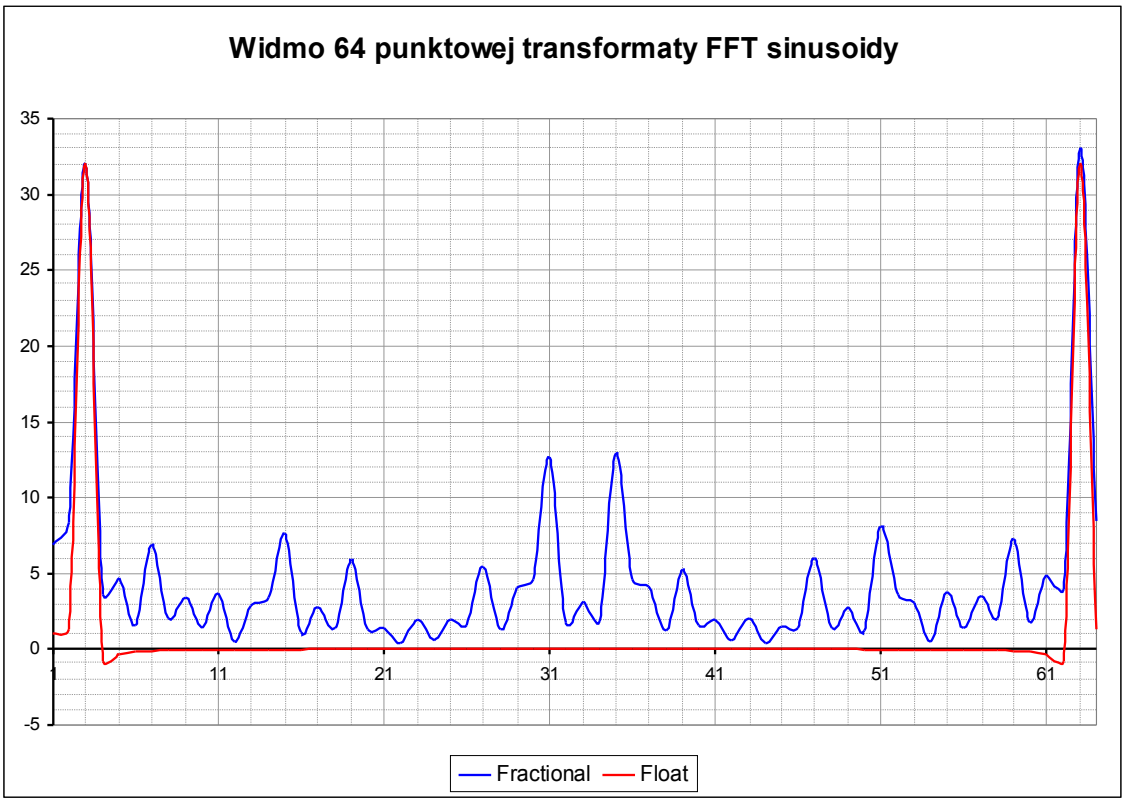


Przy obliczeniach pojawia się problem ograniczonej długości słowa binarnego i szybkiego przepełniania się rejestrów procesora. Rozwiązać go można na trzy sposoby. Pierwszy to zastosowanie liczb typu float. Mają one duży zakres zmienności i nie powodują przepełnień. Drugim sposobem jest zastosowanie skalowania zmiennych całkowitych. Przeprowadza się je po każdej operacji mogącej wygenerować przepełnienie lub tylko po jego wykryciu. Trzecim sposobem jest zastosowanie liczb stałoprzecinkowych typu fractional. Nie powodują one przepełnienia przy mnożeniu, jednak ciągle przepełnienie jest możliwe przy dodawaniu, więc ciągle trzeba stosować skalowanie. W programie zostało zastosowane najprostsze skalowanie, po każdej operacji.

3. Rezultaty

Poniżej znajdują się wyniki obliczeń wykonanych za pomocą obu programów. Obliczana była 64-punktowa FFT sygnału stałego oraz sinusoidy.





Oczywiście, wynik typu fractional został przeskalowany tak, aby możliwe było porównanie wyników obliczeń.

Jak widać, wyniki dla liczb typu fractional są obarczone błędami. Wynikają one z zaokrągleń przy mnożeniu a także skalowania. Aby optymalnie wykorzystać zakres dynamiki liczb typu fractional należałoby zrobić bardziej optymalne skalowanie.

Jakkolwiek wyniki dla liczb typu float cechują się znacznie większym stosunkiem sygnału do szumu, to jednak obliczenia na nich trwają nieco dłużej. Musimy zatem wybrać odpowiedni kompromis pomiędzy dokładnością a szybkością obliczeń.

4. Bibliografia

1. Dokumentacja rodziny procesorów ADSP-2100
2. Dokumentacja procesora ADSP-21161N
3. www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html – opis algorytmu Radix-2 FFT

5. Kody źródłowe

Poniżej znajduje się kod źródłowy FFT DIT dla liczb typu fractional.

```
/*
        Algorytm radix-2 FFT przystosowany dla roznej dlugosci transformaty.

        Obliczenia staloprzecinkowe (fractional).

        Do rewersji binarnej (mod_value):
        0x00400000 <-> 0x00000200 - 1024 punkty
        0x10000000 <-> 0x00000008 - 16 punktow
        0x04000000 <-> 0x00000020 - 64 punkty

*/


#include "def21161.h"
#include "asm_sprt.h"


#define N 64                                    //dlugosc FFT
#define N_div_2 32                              //(dlugosc FFT)/2
#define log2N 6         //log2(dlugosc FFT)
#define nover2 32                               //(dlugosc FFT)/2
#define nover4 16                               //(dlugosc FFT)/4
#define Ntimes2 128                             //(dlugosc FFT)*2
#define mod_value H#04000000//(dlugosc FFT)/2 przedstawiona w rewersji binarnej

// Deklaracje zmiennych globalnych
.GLOBAL inplacereal;
.GLOBAL inplaceimag;
.GLOBAL inputreal;
.GLOBAL inputimag;
.GLOBAL groups;
.GLOBAL bflys_per_group;
.GLOBAL node_space;
.GLOBAL blk_exponent;

.GLOBAL twid_real;
.GLOBAL twid_imag;

//Deklaracje procedur wystepujacych w programie
```

```
.GLOBAL _main;
.GLOBAL _scramble;
.GLOBAL _fft_strt;

/*
Niezbedne do uruchomienia programu na plytce testowej

.EXTERN                 RX_left_flag;

.EXTERN                 Left_Channel_In0;
.EXTERN                 Right_Channel_In0;
.EXTERN                 Left_Channel_In1;
.EXTERN                 Right_Channel_In1;

.EXTERN                 Left_Channel_Out0;
.EXTERN                 Right_Channel_Out0;

.EXTERN                 Left_Channel_Out1;
.EXTERN                 Right_Channel_Out1;
.EXTERN                 Left_Channel_Out2;
.EXTERN                 Right_Channel_Out2;

.EXTERN         Left_Channel_AD1852;
.EXTERN         Right_Channel_AD1852;
*/

.SECTION/DM seg_dmda;                           //Segment pamieci danych

.VAR inplacereal [N];
.VAR inplaceimag [N];
.VAR padding[4] = 0,0,0,0;
.VAR inputreal [N];
.VAR inputimag [N];
.VAR groups = N_div_2;
.VAR bflys_per_group = 2;
.VAR node_space = 2;
.VAR blk_exponent = 0;

.SECTION/PM seg_pmda;                           //Segment pamieci programu (dane)

.VAR twid_real [N_div_2] = "twid_real.dat";             //deklaracja tablic wspolczynnikow zespolonych
.VAR twid_imag [N_div_2] = "twid_imag.dat";             //inicjalizowanych z plikow .dat


.SECTION/PM seg_pmco;                           //Segment pamieci programu (kod)

_main:                                                  //Glowna procedura programu

        CALL _scramble;
        CALL _fft_strt;
        RTS;                                            //Powrot z procedury main

._main.END:

_scramble:                                              //Procedura zapisujaca probki wejsciowe w odwroconym
porzadku (rewersja binarna)
        I1=inputreal;                           //I4 <- adres bufora z probkami wejsciowymi w normalnej
kolejnosci
        I0=inplacereal;                 //I0 <- adres bufora z probkami zapisanymi w odwroconej kolejnosci
        BITREV(I0,0);                           //Zapisanie w I0 adresu w rewersji binarnej
        M1=1;                                           //M1 <- modyfikator adresu
        M0=mod_value;                           //M0 <- modyfkator zapisany w rewersji binarnej
        L1=0;                                           //L1 i L0 ustawione na 0 -> bufory kolowe wylaczone
```

```
        L0=0;
        LCNTR = N;                              //tyle iteracji petli, ile mamy probek
        BIT SET MODE1 BR0;                      //Wlaczenie dla I0 w DAG1 trybu adresowania z rewersja binarna
        DO brev UNTIL LCE;
        R1=DM(I1,M1);                           //Odczyt probek wejsciowych po kolei
brev:
        DM(I0,M0)=R1;                           //Zapis danych w odwroconej kolejnosci
        BIT CLR MODE1 BR0;                      //Wylaczenie dla I0 w DAG1 trybu adresowania z rewersja binarna
        RTS;                                    //Powrot z procedury

._scramble.END:



_fft_strt:
        M0=0;
        M1=1;
        L1=0;
        L2=0;
        L3=0;
        B8=twid_real;                           //Wpisanie do rejesrow bazowych B8 i B9 adresow tablic
        B9=twid_imag;                           //ze wspolczynnikami zespolonymi
        L8=N_div_2;                             //Inicjalizacja buforow kolowych o dlugosci N/2
        L9=N_div_2;
        L10=0;
        BIT SET MODE1 CBUFEN;                   //Umozliwienie korzystania z buforow kolowych
        R3=-2;
        //{————————— ETAP 1 —————————}
        I0=inplacereal;                         //Umieszczenie w rejestrach indeksowych I0-I3 adresow
        I1=inplacereal + 1;                     //bazowych buforow dla czesci rzeczywistej i urojonej
        I2=inplaceimag;                         //transformaty FFT.
        I3=inplaceimag + 1;
        I8=twid_real;
        M8=0;
        M2=2;
        LCNTR=nover2;                           //inicjalizacja N/2 obiegow petli
        R0=DM(I0,M0);
        R8=DM(I1,M0);
        R9=DM(I3,M0), R2=PM(I8,M8);
        DO group_lp UNTIL LCE;
        MRF=R8*R2(SSFR);                        //mnozenie dwoch liczb typu fractional z zaokragleniem w MAC
        R8=MR1F;
        R7=R0+R8, R1=DM(I2,M0);
        R7=ASHIFT R7 BY -1;                     //skalowanie (dzielenie przez 2) celem unikniecia przepelnienia
        DM(I0,M2)=R7;
        R7=R0-R8;
        R7=ASHIFT R7 BY -1;
        DM(I1,M2)=R7;
        MRF=R1*R2(SSFR);
        R1=MR1F;
        R7=R1+R9;
        R7=ASHIFT R7 BY -1;
        DM(I2,M2)=R7;
        R7=R1-R9, R0=DM(I0,M0);
        R7=ASHIFT R7 BY -1;
        DM(I3,M2)=R7;
        R8=DM(I1,M0);
group_lp:
        R9=DM(I3,M0);
        LCNTR=log2N - 2;                                //inicjalizacja log2N-2 obiegow petli
        //{————————————————ETAPY OD 2 DO N-1————————————————————————————}
        DO stage_loop UNTIL LCE;
        I0=inplacereal;                         //Umieszczenie w rejestrach I0 i I2 adresow
```

```
        I2=inplaceimag;                         //bazowych buforow czesci rzeczywistej i urojonej FFT
        R2=DM(groups);                                  //R2 <- liczba grup (liczba grup = N / (ETAP*2) )
        R2=ASHIFT R2 BY -1;                     //liczba grup = liczba grup /2
        DM(groups)=R2;
        LCNTR=R2;                                               //inicjalizacja groups obiegow petli
        M8=R2;                                                          //M8 <- modyfikator adresow dla
wspolczynnikow zespolonych
        M2=DM(node_space);                      //M2 <- modyfikator "szerokosci skrzydelek" motylka
        I1=I0;
        MODIFY(I1,M2);                                  //I1 <- I0 + szerokosc skrzydelek motylka
        I3=I2;
        MODIFY(I3,M2);                                  //I3 <- I2 + szerokosc skrzydelek motylka
        DO group_loop UNTIL LCE;
        I8=twid_real;                                   //I8 <- cos(2kpi/N)
        I9=twid_imag;                                   //I9 <- -sin(2kpi/N)
        LCNTR=DM(bflys_per_group);              //inicjalizacja liczba_motylkow obiegow petli
        R12=PM(I8,M8),R4=DM(I1,M0); //R12=cos(2kpi/N),R4=x1
        R13=PM(I9,M8),R5=DM(I3,M0); //R13=-sin(2kpi/N),R5=y1
        DO bfly_loop UNTIL LCE;
        MRF=R4*R13(SSFR),R0=DM(I0,M0);          //MRF=x1*(-sin(2kpi/N)),R0=x0
        MRF=MRF+R5*R12(SSFR),R1=DM(I2,M0);      //MRF=(y1*(cos(2kpi/N))+x1*(-sin(2kpi/N))),R1=y0
        R9=MR1F;                                                //R9=y1*(cos(2kpi/N))+x1*(-
sin(2kpi/N))
        MRF=R4*R12(SSFR);                                       //MRF=x1*(cos(2kpi/N))
        MRF=MRF-R5*R13(SSFR);                                   //MRF=x1*(cos(2kpi/N))-y1*(-sin(2kpi/N))
        R8=MR1F;                                                //R8=x1*(cos(2kpi/N))-y1*(-
sin(2kpi/N))
        R7=R1-R9;                                               //R7=y0-[y1*(cos(2kpi/N))+x1*(-
sin(2kpi/N))]
        R7=ASHIFT R7 BY -1;                                     //skalowanie (dzielenie przez 2)
        DM(I3,M1)=R7;                                           //zapis wyniku do bufora
        R7=R0-R8,R5=DM(I3,M0),R13=PM(I9,M8);//R7=x0-[x1*(cos(2kpi/N))-y1*(-sin(2kpi/N))],
R5=nastepny(y1),R13=nastepny(-sin(2kpi/N))
        R7=ASHIFT R7 BY -1;
        DM(I1,M1)=R7;
        R7=R0+R8,R4=DM(I1,M0),R12=PM(I8,M8);//R7=x0+[x1*(cos(2kpi/N))-y1*(-sin(2kpi/N))],
R4=nastepny(x1),R12=nastepny(cos(2kpi/N))
        R7=ASHIFT R7 BY -1;
        DM(I0,M1)=R7;
        R7=R1+R9;                                               //{R7=y0+[y1*(cos(2kpi/N))+x1*(-
sin(2kpi/N))]}
        R7=ASHIFT R7 BY -1;
bfly_loop:
        DM(I2,M1)=R7;
        MODIFY(I0,M2);                                          //modyfikacja I0,I1,I2,I3 o szerokosc
motylka
        MODIFY(I1,M2);
        MODIFY(I2,M2);
group_loop:
        MODIFY(I3,M2);
        R2=DM(node_space);
        R2=ASHIFT R2 BY 1;
        DM(node_space)=R2;                                      //node_space=node_space*2 (zwiekszenie
szerokosci motylka)
        R2=DM(bflys_per_group);
        R2=ASHIFT R2 BY 1;
stage_loop:
        DM(bflys_per_group)=R2;                 //bflys_per_group=bflys_per_group*2 (zwiekszenie ilosci
motylkow w grupie)
        //{————— OSTATNI ETAP ————————}
        I0=inplacereal;
        I1=inplacereal+nover2;
        I2=inplaceimag;
```

```
	I3=inplaceimag+nover2;
	LCNTR=N_div_2;
	M2=DM(node_space);
	M8=1;
	I8=twid_real;
	I9=twid_imag;
	R12=PM(I8,M8),R4=DM(I1,M0);
	R13=PM(I9,M8),R5=DM(I3,M0);
	DO bfly_lp UNTIL LCE;
	MRF=R4*R13(SSFR),R0=DM(I0,M0);
	MRF=MRF+R5*R12(SSFR),R1=DM(I2,M0);
	R9=MR1F;
	MRF=R4*R12(SSFR);
	MRF=MRF-R5*R13(SSFR);
	R8=MR1F;
	R7=R1-R9;
	R7=ASHIFT R7 BY -1;
	DM(I3,M1)=R7;
	R7=R0-R8,R5=DM(I3,M0),R13=PM(I9,M8);
	R7=ASHIFT R7 BY -1;
	DM(I1,M1)=R7;
	R7=R0+R8,R4=DM(I1,M0),R12=PM(I8,M8);
	R7=ASHIFT R7 BY -1;
	DM(I0,M1)=R7;
	R7=R1+R9;
	R7=ASHIFT R7 BY -1;
bfly_lp:
	DM(I2,M1)=R7;
	RTS;

._fft_strt.END:
```

Poniżej znajduje się kod źródłowy FFT DIT dla liczb typu float.

```
/*
	Algorytm radix-2 FFT przystosowany dla roznej dlugosci transformaty.

	Obliczenia zmiennoprzecinkowe (32bit float).

	Do rewersji binarnej (mod_value):
	0x00400000 <-> 0x00000200 - 1024 punkty
	0x10000000 <-> 0x00000008 - 16 punktow
	0x04000000 <-> 0x00000020 - 64 punkty

*/


#include "def21161.h"
#include "asm_sprt.h"


#define N 64                                    //dlugosc FFT
#define N_div_2 32                              //(dlugosc FFT)/2
#define log2N 6                                 //log2(dlugosc FFT)
#define nover2 32                               //(dlugosc FFT)/2
#define nover4 16                               //(dlugosc FFT)/4
#define Ntimes2 128                             //(dlugosc FFT)*2
#define mod_value H#04000000                    //(dlugosc FFT)/2 przedstawiona w rewersji binarnej

// Deklaracje zmiennych globalnych
.GLOBAL inplacereal;
.GLOBAL inplaceimag;
```

```
.GLOBAL inputreal;
.GLOBAL inputimag;
.GLOBAL groups;
.GLOBAL bflys_per_group;
.GLOBAL node_space;
.GLOBAL blk_exponent;

.GLOBAL twid_real;
.GLOBAL twid_imag;

//Deklaracje procedur wystepujacych w programie
.GLOBAL _main;
.GLOBAL _scramble;
.GLOBAL _fft_strt;

/*
Niezbedne do uruchomienia programu na plytce testowej

.EXTERN		RX_left_flag;

.EXTERN		Left_Channel_In0;
.EXTERN		Right_Channel_In0;
.EXTERN		Left_Channel_In1;
.EXTERN		Right_Channel_In1;

.EXTERN		Left_Channel_Out0;
.EXTERN		Right_Channel_Out0;

.EXTERN		Left_Channel_Out1;
.EXTERN		Right_Channel_Out1;
.EXTERN		Left_Channel_Out2;
.EXTERN		Right_Channel_Out2;

.EXTERN	Left_Channel_AD1852;
.EXTERN	Right_Channel_AD1852;
*/

.SECTION/DM seg_dmda;                                   //Segment pamieci
danych

.VAR inplacereal [N];
.VAR inplaceimag [N];
.VAR padding[4] = 0,0,0,0;
.VAR inputreal [N];
.VAR inputimag [N];
.VAR groups = N_div_2;
.VAR bflys_per_group = 2;
.VAR node_space = 2;
.VAR blk_exponent = 0;

.SECTION/PM seg_pmda;                                   //Segment pamieci
programu (dane)

.VAR twid_real [N_div_2] = "twid_real.dat";             //deklaracja tablic wspolczynnikow zespolonych
.VAR twid_imag [N_div_2] = "twid_imag.dat";             //inicjalizowanych z plikow .dat


.SECTION/PM seg_pmco;                                   //Segment pamieci
programu (kod)

_main:                                                  //Glowna
procedura programu
```

```
        CALL _scramble;
        CALL _fft_strt;
        RTS;                                                    //Powrot z
procedury main

._main.END:



_scramble:                                      //Procedura zapisujaca probki wejsciowe w odwroconym
porzadku (rewersja binarna)
        I1=inputreal;                   //I4 <- adres bufora z probkami wejsciowymi w normalnej
kolejnosci
        I0=inplacereal;         //I0 <- adres bufora z probkami zapisanymi w odwroconej kolejnosci
        BITREV(I0,0);                   //Zapisanie w I0 adresu w rewersji binarnej
        M1=1;                           //M1 <- modyfikator adresu
        M0=mod_value;                   //M0 <- modyfkator zapisany w rewersji binarnej
        L1=0;                           //L1 i L0 ustawione na 0 -> bufory kolowe wylaczone
        L0=0;
        LCNTR = N;                      //tyle iteracji petli, ile mamy probek
        BIT SET MODE1 BR0;              //Wlaczenie dla I0 w DAG1 trybu adresowania z rewersja binarna
        DO brev UNTIL LCE;
        F1=DM(I1,M1);                   //Odczyt probek wejsciowych po kolei
brev:
        DM(I0,M0)=F1;                   //Zapis danych w odwroconej kolejnosci
        BIT CLR MODE1 BR0;              //Wylaczenie dla I0 w DAG1 trybu adresowania z rewersja binarna
        RTS;                            //Powrot z procedury

._scramble.END:



_fft_strt:
        M0=0;
        M1=1;
        L1=0;
        L2=0;
        L3=0;
        B8=twid_real;                   //Wpisanie do rejesrow bazowych B8 i B9 adresow tablic
        B9=twid_imag;                   //ze wspolczynnikami zespolonymi
        L8=N_div_2;                     //Inicjalizacja buforow kolowych o dlugosci N/2
        L9=N_div_2;
        L10=0;
        BIT SET MODE1 CBUFEN;           //Umozliwienie korzystania z buforow kolowych
        F3=-2.0;
        //{————————— ETAP 1 —————————}
        I0=inplacereal;                 //Umieszczenie w rejestrach indeksowych I0-I3 adresow
        I1=inplacereal + 1;             //bazowych buforow dla czesci rzeczywistej i urojonej
        I2=inplaceimag;                 //transformaty FFT.
        I3=inplaceimag + 1;
        M8=0;
        M2=2;
        LCNTR=nover2;                   //inicjalizacja N/2 obiegow petli
        F0=DM(I0,M0);
        F8=DM(I1,M0);
        F9=DM(I3,M0);
        DO group_lp UNTIL LCE;
        F7=F0+F8, F1=DM(I2,M0);
        DM(I0,M2)=F7;
        F7=F0-F8;;
        DM(I1,M2)=F7;
        F7=F1+F9;
        DM(I2,M2)=F7;
```

```
        F7=F1-F9, F0=DM(I0,M0);
        DM(I3,M2)=F7;
        F8=DM(I1,M0);
group_lp:
        F9=DM(I3,M0);
        LCNTR=log2N - 2;                                    //inicjalizacja log2N-2 obiegow petli
        //{————————————————ETAPY OD 2 DO N-1————————————————————————————————————}
        DO stage_loop UNTIL LCE;
        I0=inplacereal;                             //Umieszczenie w rejestrach I0 i I2 adresow
        I2=inplaceimag;                             //bazowych buforow czesci rzeczywistej i urojonej FFT
                R2=DM(groups);                              //R2 <- liczba grup (liczba grup = N / (ETAP*2) )
                R2=ASHIFT R2 BY -1;                         //liczba grup = liczba grup /2
                DM(groups)=R2;
                LCNTR=R2;                                           //inicjalizacja groups obiegow petli
                M8=R2;                                                      //M8 <- modyfikator adresow dla
wspolczynnikow zespolonych
                M2=DM(node_space);                          //M2 <- modyfikator "szerokosci skrzydelek" motylka
                I1=I0;
                MODIFY(I1,M2);                                      //I1 <- I0 + szerokosc skrzydelek motylka
                I3=I2;
                MODIFY(I3,M2);                                      //I3 <- I2 + szerokosc skrzydelek motylka
                DO group_loop UNTIL LCE;
                I8=twid_real;                               //I8 <- cos(2kpi/N)
                I9=twid_imag;                               //I9 <- -sin(2kpi/N)
                LCNTR=DM(bflys_per_group);          //inicjalizacja liczba_motylkow obiegow petli
                F12=PM(I8,M8),F4=DM(I1,M0);  //F12=cos(2kpi/N),F4=x1
                F13=PM(I9,M8),F5=DM(I3,M0);  //F13=-sin(2kpi/N),F5=y1
                DO bfly_loop UNTIL LCE;
                F10=F4*F13,F0=DM(I0,M0);
                F6=F5*F12,F1=DM(I2,M0);
                F9=F10+F6;
                F10=F4*F12;
                F6=F5*F13;
                F8=F10-F6;
                F7=F1-F9;
                DM(I3,M1)=F7;
                F7=F0-F8,F5=DM(I3,M0),F13=PM(I9,M8);
                DM(I1,M1)=F7;
                F7=F0+F8,F4=DM(I1,M0),F12=PM(I8,M8);
                DM(I0,M1)=F7;
                F7=F1+F9;
bfly_loop:
                DM(I2,M1)=F7;
                MODIFY(I0,M2);                                              //modyfikacja I0,I1,I2,I3 o szerokosc
motylka
                MODIFY(I1,M2);
                MODIFY(I2,M2);
group_loop:
                MODIFY(I3,M2);
                R2=DM(node_space);
                R2=ASHIFT R2 BY 1;
                DM(node_space)=R2;                                  //node_space=node_space*2 (zwiekszenie
szerokosci motylka)
                R2=DM(bflys_per_group);
                R2=ASHIFT R2 BY 1;
stage_loop:
                DM(bflys_per_group)=R2;                     //bflys_per_group=bflys_per_group*2 (zwiekszenie ilosci
motylkow w grupie)
                //{{————— OSTATNI ETAP ——————}
                I0=inplacereal;
                I1=inplacereal+nover2;
                I2=inplaceimag;
                I3=inplaceimag+nover2;
```

```
        LCNTR=nover2;
        M2=DM(node_space);
        M8=1;
        I8=twid_real;
        I9=twid_imag;
        F12=PM(I8,M8),F4=DM(I1,M0);
        F13=PM(I9,M8),F5=DM(I3,M0);
        DO bfly_lp UNTIL LCE;
        F10=F4*F13,F0=DM(I0,M0);
        F6=F5*F12,F1=DM(I2,M0);
        F9=F10+F6;
        F10=F4*F12;
        F6=F5*F13;
        F8=F10-F6;
        F7=F1-F9;
        DM(I3,M1)=F7;
        F7=F0-F8,F5=DM(I3,M0),F13=PM(I9,M8);
        DM(I1,M1)=F7;
        F7=F0+F8,F4=DM(I1,M0),F12=PM(I8,M8);
        DM(I0,M1)=F7;
        F7=F1+F9;
bfly_lp:
        DM(I2,M1)=F7;
        RTS;

._fft_strt.END:
```