МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»

Інженерно-технічний факультет

Кафедра комп’ютерних систем та мереж

**Розрахунково-графічна робота**

з дисципліни

«Цифрова обробка сигналів»

на тему:

«Розрахунок параметрів виконання алгоритму ШПФ»

Виконав:

студент IV-го курсу

КСМ, ІТФ

Бородін Д.В.

Перевірив:

Ваврук Є.Я.

**Ужгород – 2012**

**Постановка завдання**

*Варіант № 5*

Розрахувати параметри виконання алгоритму ШПФ з такими вхідними даними:

|  |  |
| --- | --- |
| Кількість точок | *16384* |
| Основа ШПФ | *2* |
| Прорідження | *часове* |
| Частота роботи процесора | *1 МГц* |
| Розрядність вхідних даних | *20(10+10)* |
| Тип вхідного інтерфейсу | *PCI* |
| Тип вихідного інтерфейсу | *Зовнішня пам'ять* |

**Анотація**

Дана розрахунково-графічна робота представляє один із методів швидких перетворень, а саме алгоритм швидкого перетворення Фур'є (ШПФ). Для прикладу було взято 16384-точкове ШПФ за основою 2 для 20-розрядних вхідних даних з часовим прорідженням. В роботі описано механізми обчислення швидкого перетворення Фур’є за заданною основою, обчислено часові ресурси для виконання обчислення, створена функціональна схема системи та написана програма, що реалізує даний алгоритм ШПФ.

**Зміст**

ВСТУП.......................................................................................................................5

[1 Теоретичні відомості 6](#_Toc338621080)

[1.1 Основні визначення ШПФ 6](#_Toc338621081)

[1.2 Повертаючі множники 6](#_Toc338621082)

[1.3 Основні формули для обчислення ШПФ 7](#_Toc338621083)

[2 Розробка графа алгоритму ШПФ, побудова блок-схеми алгоритму та біт-інверсного прорідження 9](#_Toc338621084)

[2.1 Побудова графа алгоритму ШПФ з основою 2 9](#_Toc338621085)

[2.2 Біт інверсний порядок видачі даних 10](#_Toc338621086)

[2.3 Блок-схема перетворення 12](#_Toc338621087)

[3 Розрахунки алгоритму ШПФ 13](#_Toc338621088)

[4 Розробка функціональної схеми 15](#_Toc338621089)

[5 Розробка програми виконання алгоритму ШПФ 17](#_Toc338621090)

ВИСНОВКИ............................................................................................................22

Список використаної літератури...........................................................................23

**ВСТУП**

Аналіз Фур'є закладає основи багатьох методів, що застосовуються в області цифрової обробки сигналів (ЦОС). По суті справи, перетворення Фур'є (фактично існує кілька варіантів таких перетворень) дозволяє співставити сигналу, заданому в часовій області, його еквівалентне представлення в частотній області, і навпаки, якщо відома частотна характеристика сигналу, то зворотне перетворення Фур'є дозволяє визначити відповідний сигнал у часовій області.

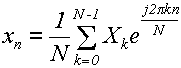
В даній роботі представлено алгоритм 16384-точкового ШПФ за основою 2 для 20-розрядних вхідних даних з прорідженням по часу, розроблена функціональна схема алгоритму з використанням вхідного інтерфейсу PCI та вихідного інтерфейсу типу зовнішньої пам’яті, а саме USB-контролера, та створена програма на мові C, шо описує даний алгоритм ШПФ.

# 1 Теоретичні відомості

## 1.1 Основні визначення ШПФ

1. Дано кінцеву послідовність x0, x1, x2,..., xN-1 (у загальному випадку комплексних чисел). ДПФ полягає в пошуку послідовності X0, X1, X2,..., XN-1, елементи якої обчислюються за формулою:

Описание: image006 (1)  
2.Зворотне ДПФ полягає в пошуку послідовності x0, x1, x2,..., xN-1, елементи якої обчислюються за формулою:

(2)  
Основною властивістю перетворень (1) і (2) є те, що з послідовності {x} отримується (при прямому перетворенні) послідовність {X}, а якщо застосувати до {X} зворотне перетворення, то знову отримується вихідна послідовність {x}.

3. Величина Описание: image177 називаєтьсяповертаючим множником.

## 1.2 Повертаючі множники

При *k* = 1  Описание: image010

Пряме перетворення Фур’є можна виразити через повертаючі множники. Тоді формула (1) матиме вигляд:

Описание: image012 (3)

З формули (3) можна визначити геометричний зміст перетворення Фур’є таким чином: представити *N* комплексних чисел-векторів з набору *{x},* кожне у вигляді суми векторів з набору *{X},* повернених на кути, кратні *2π/N*.

## 1.3 Основні формули для обчислення ШПФ

*Теорема 1*. Якщо комплексне число представлене у вигляді *e j2πN*, де N - ціле, то *e j2πN = 1*.

*Теорема 2.* Величина Описание: image014періодична по *k* і по *n* з періодом *N*. Тобто, для будь-яких цілих *l* і *m* виконується рівність:

Описание: image016 (4)

*Теорема 3*.

Для величини Описание: image014справедлива формула: Описание: image025 (5)

З наведених теорем визначається основна ідея алгоритму ШПФ:

1. Необхідно розділити суму (1) з *N* доданків на дві суми по *N/2* доданків, і обчислити їх окремо. Для обчислення кожної з підсум, треба їх теж розділити на дві і т.д.
2. Необхідно повторно використовувати вже обчислені доданки.

При обчисленні алгоритму ШПФ застосовують або "проріджування за часом" (в першу суму попадають доданки з парними номерами, а в другу - з непарними), або "проріджування за частотою" (в першу суму попадають перші *N/2* доданків, а в другу – інші). Обидва варіанти рівноцінні. В силу специфіки алгоритму доводиться застосовувати тільки *N*, що є ступенями 2.

*Теорема 4.* Визначимо ще дві послідовності: *{x[even]}* і *{x[odd]}* через послідовність *{x}* таким чином:

*x[even]n = x2n, x[odd]n = x2n+1,* (6)

*n = 0, 1,..., N/ 2-1*

Нехай до цих послідовностей застосовані ДПФ і отримані результати у вигляді двох нових послідовностей *{X[even]}* і *{X[odd]}* по *N/2* елементів у кожній.

Елементи послідовності *{X}* можна виразити через елементи послідовностей *{X[even]}* і *{X[odd]}* за формулою:

 (7)



Формула (7) дозволяє скоротити число множень удвічі (не враховуючи множень при обчисленні *X[even]*k і *X [odd]k*), якщо обчислювати *Xk* не послідовно від *0* до *N - 1*, а попарно: *X0* і *XN/2, X1* і *XN/2+1,..., XN/ 2-1* і *XN*. Пари утворяться за принципом: *Xk* і *XN/2+k*.

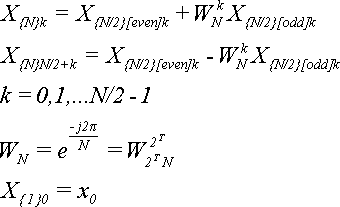
*Теорема 5.* ДПФ можна обчислити і за формулою:

Описание: image087 (8)

З теореми випливає, що можна не зберігати обчислені значення *X[even]k* і *X[odd]k* після обчислення чергової пари, і одне обчислення Описание: image059можна використовувати для обчислення двох елементів послідовності *{X}.*

На цьому кроці буде виконане *N/2* множень комплексних чисел. Якщо застосувати подібні схеми для обчислення послідовностей *{X[even]}* і *{X[odd]},* то кожна з них вимагатиме *N/4* множень, разом ще *N/2*. Продовжуючи аналогічно далі *log2N* разів, можна дійти до сум, що містять лише один доданок, так що загальна кількість множень рівна (N/2)log2N.

Отже, в основі алгоритму ШПФ лежать такі формули:

 (9)

Відомі два різновиди алгоритмів ШПФ - проріджування за часом (decimation in time - DIT) і проріджування по частоті (decimation in frequency - D1F), що мають однакову обчислювальну складність.

# 2 Розробка графа заданого алгоритму ШПФ, побудова блок-схеми алгоритму та біт-інверсного прорідження

## 2.1 Побудова графа алгоритму ШПФ з основою 2

Якщо число *N* є степенем 2, то його можна записати як *(N/2) \* 2*; аналогічно *N/2 = (M/4) \* 2* і т.д. В результаті елементи початкового одновимірного масиву можна розподілити таким чином, щоб елементарними операціями були двохточкові ШПФ. Для побудови графу *N*-точкового ШПФ з основою 2 потрібно визначити кількість ярусів (*Nяр*) графу та кількості метеликів (*Nм*) на кожному ярусі таким чином:

*Nяр = Log2N*, для *N=16384*, *Log216384 = 14*;

*Nм = N/2 = 16384/2= 8192.*

Метелик ШПФ показаний на рис. 3, де

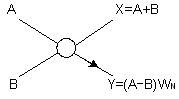


Рис. 1. Двохточкове ШПФ (Метелик)

*X, Y* – результати базової операції; *А, B* – вхідні відліки; *WN* – повертаючі множники.

Граф-алгоритм для 16-точкового ШПФ за основою 2 з часовим прорідженням показаний на рис. 2.

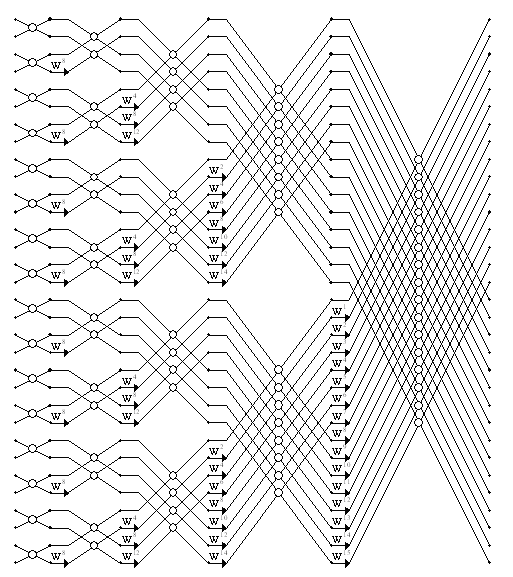


Рис.2. Граф 16-точкового ШПФ за основою 2 з прорідженням по часу

На рис.2 кільцям відповідають двохточкові ШПФ, стрілками зображено процедуру множення на повертаючі множники. Елементи пам’яті зображено точками.

## 2.2 Біт інверсний порядок видачі даних

Для реалізації ШПФ з прорідженням по часу для 16384-точкового ШПФ виконують нарощування ДПФ починаючи з двохточкових, які в свою чергу об’єднуються в групу двохточкових ДПФ, і так далі до об’єднання 16384-точкових ДПФ.

Результат виконання ШПФ отримується у біт-інверсному порядку. Тому для адресів вихідних відліків необхідно застосувати біт-інверсний алгоритм. Суть даного алгоритму полягає у наступному: десятковий індекс n перетворюють в його двійковий еквівалент. Потім двійкові розряди розташовуються у зворотньому порядку і перетворюються назад у десяткове число. Приклад наведений у табл. 1.

*Таблиця 1. Біт-інверсне прорідження*

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Двійкове представлення | Двійкова інверсія | Двійково-інверсний номер |
| 0 | 00000000000000 | 00000000000000 | 0 |
| 1 | 00000000000001 | 10000000000000 | 8192 |
| 2 | 00000000000010 | 01000000000000 | 4096 |
| 3 | 00000000000011 | 11000000000000 | 12228 |
| 4 | 00000000000100 | 00100000000000 | 2048 |
| 5 | 00000000000101 | 10100000000000 | 10240 |
| 6 | 00000000000110 | 01100000000000 | 6144 |
| 7 | 00000000000111 | 11100000000000 | 14336 |
| 8 | 00000000001000 | 00010000000000 | 1024 |
| 9 | 00000000001001 | 10010000000000 | 9216 |
| 10 | 00000000001010 | 01010000000000 | 5120 |
| 11 | 00000000001011 | 11010000000000 | 13312 |
| … | … | … | … |
| 16381 | 11111111111101 | 10111111111111 | 12287 |
| 16382 | 11111111111110 | 01111111111111 | 8191 |
| 16383 | 11111111111111 | 11111111111111 | 16383 |

## 2.3 Блок-схема перетворення

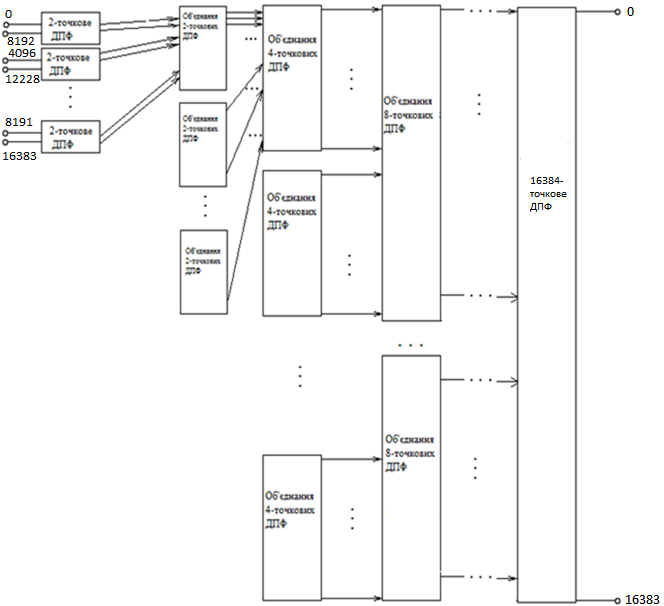


Рис. 3. Блок-схема алгоритму 16384-точкового перетворення за основою 2

# 3 Розрахунки алгоритму ШПФ

Частота роботи процесора: , звідси цикл виконання команди: .

base – основа базової операції «метелик»;

N – к-ть точок вх. перетворення;

base=2;

N=16384;

 – кількість етапів перетворення;

 – кількість базових операцій «метелик» на одному етапі;

 – кількість базових операцій у всьому перетворенні;

;

;



Для виконання базової операції «метелик» необхідно:

* 4 операцій множення;
* 6 операцій додавання;
* 6 операцій читання з пам`яті:
* 2\*2=4 (для читання дійсної та уявної частини вхідних відліків);
* 1\*2=2 (для читання дійсної та уявної частини комплексних коефіцієнтів);
* 4 операцій запису:

- 2\*2=4 (для запису дійсної та уявної частини вхідних відліків);

В результаті на одну базову операцію припадає 20 операцій: Nна 1 мет=20 (оп).

Тривалість виконання обчислення ШПФ:



Тривалість надходження даних у процесор для обробки (PCI):

Тнадх=75*нс* – такт надходження даних в PCI;



Тривалість виходу даних із процесору (зовнішня пам'ять):

Частота роботи USB-контролера . Але так як частота процесора =1МГц, то тривалість виходу даних із процесора буде обчислюватись при частоті роботи USB-контролера рівній 1МГц:

Твих=1000*нс* – такт надходження даних до USB;



Тривалість надходження даних у процессор, виходу із нього та тривалість обчислення ШПФ:



# 4 Розробка функціональної схеми

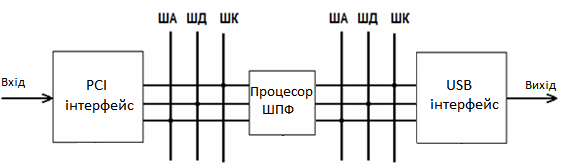


Рис.4. Фунціональна схема

**PCI** (Peripheral Component Interconnect – з'єднання зовнішніх компонентів) – [шина вводу/виводу](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D1%88%D0%B8%D0%BD%D0%B0) для підключення периферійних пристроїв до [материнської плати](http://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%BD%D1%81%D1%8C%D0%BA%D0%B0_%D0%BF%D0%BB%D0%B0%D1%82%D0%B0) [комп'ютера](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80). Цей інтерфейс підтримує тактову частоту до 33 МГц (варіант PCI 2.1 – до 66 МГц), має максимальну пропускну здатність до 132 Мбайт / с на частоті 33 МГц для 32-х розрядної шини (264 Мбайт / с для 32-x розрядних і 528 Мбайт / с для 64-х розрядних даних на частоті 66 МГц). Конструктивно роз'єм складається з двох розьашованих одна за одною секцій по 64 контакти. Всередині другій секції знаходиться пластмасова поперечна перегородка (ключ) для запобігання неправильної установки карт. Роз'єми PCI і карти до них підтримують рівні сигналів або 5 В., або 3,3 В., або обидва рівня (універсальні). У перших двох випадках карти повинні відповідати рівню сигналу роз'єму, універсальні карти ставляться в будь-який роз'єм. Інтерфейс PCI забезпечує підтримку режимів Bus Mastering і автоматичної конфігурації компонентів при установці (Plug-and-Play). Всі слоти PCI на материнській платі згруповані в сегменти, число роз'ємів в сегменті обмежено чотирма. Якщо сегментів кілька, вони з'єднуються за допомогою так званих мостів (bridge).

**USB** (Universal Serial Bus - «Універсальна послідовна шина») – послідовний інтерфейс передачі даних для середньошвидкісних і низькошвидкісних периферійних пристроїв в обчислювальній техніці. Для підключення периферійних пристроїв до шини USB використовується чьотирьохпровідний кабель, при цьому два дроти (вита пара) в диференціальному включенні використовуються для прийому і передачі даних, а два дроти – для живлення периферійного пристрою. Завдяки вбудованим лініям живлення USB дозволяє підключати периферійні пристрої без власного джерела живлення (максимальна сила струму, споживаного пристроєм по лініях живлення шини USB не повинна перевищувати 500 мА, у USB 3.0 – 900 мА). До одного USB каналу можна ланцюжком підключити до 127 зовнішніх пристроїв. На сучасних материнських платах зазвичай є по два каналу USB на контролер. Обмін даними по шині USB проходить в пакетному режимі при максимальній пропускній здатності до 12 Мбіт / с, у версії 2.0 швидкість збільшена до 300Мбіт / с.

# 5 Розробка програми виконання алгоритму ШПФ

Структуру програми, що виконує обчислення за алгоритмом ШПФ можна уявити наступним чином:

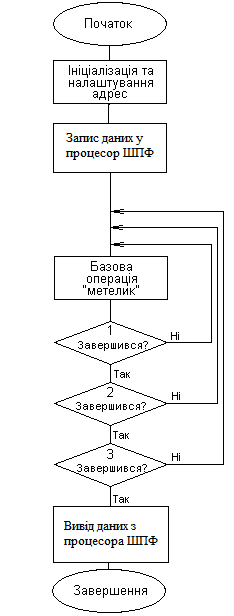


Рис.5. Узагальнена блок-схема алгоритму

Кожен з трьох циклів призначений для правильного визначення номеру відліку в конкретний момент обчислення. Перший цикл визначає номер ярусу, другий – номер базової операції у ярусі, третій – номер відліку у базовій операції.

Вводиться масив, що зберігає відліки, в програмі названий matrix, його номер відповідно – N (кількість точок перетворення). Кожен елемент масиву – комплексне число. Інший массив W призначений для зберігання повертаючи множників. На кожну базову операцію припадає 1 повертаючий множник, тому його розмір:

14\*16384\*1=229376 (14 – кількість ярусів, 16384 – кількість базових операцій в ярусі). Елемент цього масиву є так само комплексним числом.

Текст програми, написаної на мові С:

#define NUMBER\_IS\_2\_POW\_K(x) ((!((x)&((x)-1)))&&((x)>1)) // x is pow(2, k), k=1,2, ...

#define FT\_DIRECT -1 // Direct transform.

#define FT\_INVERSE 1 // Inverse transform.

bool FFT(float \*Rdat, float \*Idat, int N, int LogN, int Ft\_Flag)

{

// parameters error check:

if((Rdat == NULL) || (Idat == NULL)) return false;

if((N > 16384) || (N < 1)) return false;

if(!NUMBER\_IS\_2\_POW\_K(N)) return false;

if((LogN < 2) || (LogN > 14)) return false;

if((Ft\_Flag != FT\_DIRECT) && (Ft\_Flag != FT\_INVERSE)) return false;

register int i, j, n, k, io, ie, in, nn;

float ru, iu, rtp, itp, rtq, itq, rw, iw, sr;

static const float Rcoef[14] =

{ -1.0000000000000000F, 0.0000000000000000F, 0.7071067811865475F,

0.9238795325112867F, 0.9807852804032304F, 0.9951847266721969F,

0.9987954562051724F, 0.9996988186962042F, 0.9999247018391445F,

0.9999811752826011F, 0.9999952938095761F, 0.9999988234517018F,

0.9999997058628822F, 0.9999999264657178F

};

static const float Icoef[14] =

{ 0.0000000000000000F, -1.0000000000000000F, -0.7071067811865474F,

-0.3826834323650897F, -0.1950903220161282F, -0.0980171403295606F,

-0.0490676743274180F, -0.0245412285229122F, -0.0122715382857199F,

-0.0061358846491544F, -0.0030679567629659F, -0.0015339801862847F,

-0.0007669903187427F, -0.0003834951875714F

};

nn = N >> 1;

ie = N;

for(n=1; n<=LogN; n++)

{

rw = Rcoef[LogN - n];

iw = Icoef[LogN - n];

if(Ft\_Flag == FT\_INVERSE) iw = -iw;

in = ie >> 1;

ru = 1.0F;

iu = 0.0F;

for(j=0; j<in; j++)

{

for(i=j; i<N; i+=ie)

{

io = i + in;

rtp = Rdat[i] + Rdat[io];

itp = Idat[i] + Idat[io];

rtq = Rdat[i] - Rdat[io];

itq = Idat[i] - Idat[io];

Rdat[io] = rtq \* ru - itq \* iu;

Idat[io] = itq \* ru + rtq \* iu;

Rdat[i] = rtp;

Idat[i] = itp;

}

sr = ru;

ru = ru \* rw - iu \* iw;

iu = iu \* rw + sr \* iw;

}

ie >>= 1;

}

for(j=i=1; i<N; i++)

{

if(i < j)

{

io = i - 1;

in = j - 1;

rtp = Rdat[in];

itp = Idat[in];

Rdat[in] = Rdat[io];

Idat[in] = Idat[io];

Rdat[io] = rtp;

Idat[io] = itp;

}

k = nn;

while(k < j)

{

j = j - k;

k >>= 1;

}

j = j + k;

}

if(Ft\_Flag == FT\_DIRECT) return true;

rw = 1.0F / N;

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

{

Rdat[i] \*= rw;

Idat[i] \*= rw;

}

return true;

}

// Пример вычисления БПФ от одного периода косинусного

// действительного сигнала

void Test\_FFT()

{

static float Re[8];

static float Im[8];

float p = 2 \* 3.141592653589 / 8; // будет 8 отсчетов на период

int i;

// формируем сигнал

for(i=0; i<8; i++)

{

Re[i] = cos(p \* i); // заполняем действительную часть сигнала

Im[i] = 0.0; // заполняем мнимую часть сигнала

}

FFT(Re, Im, 8, 3, -1); // вычисляем прямое БПФ

// выводим действительную и мнимую части спектра и спектр мощности

FILE \*f = fopen("spectrum.txt", "w");

for(i=0; i<8; i++)

{

fprintf(f, "%10.6f %10.6f %10.6f\n", Re[i], Im[i], Re[i]\*Re[i]+Im[i]\*Im[i]);

}

fclose(f);

}

const double TwoPi = 6.283185307179586;

void FFTAnalysis(double \*AVal, double \*FTvl, int Nvl, int Nft) {

int i, j, n, m, Mmax, Istp;

double Tmpr, Tmpi, Wtmp, Theta;

double Wpr, Wpi, Wr, Wi;

double \*Tmvl;

n = Nvl \* 2; Tmvl = new double[n];

for (i = 0; i < Nvl; i++) {

j = i \* 2; Tmvl[j] = 0; Tmvl[j+1] = AVal[i];

}

i = 1; j = 1;

while (i < n) {

if (j > i) {

Tmpr = Tmvl[i]; Tmvl[i] = Tmvl[j]; Tmvl[j] = Tmpr;

Tmpr = Tmvl[i+1]; Tmvl[i+1] = Tmvl[j+1]; Tmvl[j+1] = Tmpr;

}

i = i + 2; m = Nvl;

while ((m >= 2) && (j > m)) {

j = j - m; m = m >> 1;

}

j = j + m;

}

Mmax = 2;

while (n > Mmax) {

Theta = -TwoPi / Mmax; Wpi = Sin(Theta);

Wtmp = Sin(Theta / 2); Wpr = Wtmp \* Wtmp \* 2;

Istp = Mmax \* 2; Wr = 1; Wi = 0; m = 1;

while (m < Mmax) {

i = m; m = m + 2; Tmpr = Wr; Tmpi = Wi;

Wr = Wr - Tmpr \* Wpr - Tmpi \* Wpi;

Wi = Wi + Tmpr \* Wpi - Tmpi \* Wpr;

while (i < n) {

j = i + Mmax;

Tmpr = Wr \* Tmvl[j] - Wi \* Tmvl[j-1];

Tmpi = Wi \* Tmvl[j] + Wr \* Tmvl[j-1];

Tmvl[j] = Tmvl[i] - Tmpr; Tmvl[j-1] = Tmvl[i-1] - Tmpi;

Tmvl[i] = Tmvl[i] + Tmpr; Tmvl[i-1] = Tmvl[i-1] + Tmpi;

i = i + Istp;

}

}

Mmax = Istp;

}

for (i = 0; i < Nft; i++) {

j = i \* 2; FTvl[Nft - i - 1] = Sqrt(Sqr(Tmvl[j]) + Sqr(Tmvl[j+1]));

}

delete []Tmvl;

}

**ВИСНОВКИ**

В даній розрахунково-графічній роботі було реалізовано алгоритм швидкого перетворення Фур'є (ШПФ) за основою 2 для 20-розрядних вхідних даних з часовим прорідженням. Виконуючи реалізацію даного алгоритму було обчислено тривалість виконання обчислення ШПФ, тривалість надходження даних у процесор та тривалість обчислення ШПФ з використанням входу PCI та виходу USB (зовнішня пам'ять). Також була побудована функціональна схема для реалізації алгоритму з цими інтерфейсами та написана програма на мові С. Вхідні дані представляються 16384-ма вхідними відліками, що містять дійсну та уявну частину.

**Список використаної літератури**

1. Куприянов М. С., Матюшкин Б. Д. Цифровая обработка сигналов: процессоры, алгоритмы, средства проектирования. – Спб. : Политехника, 1998.
2. Бабак В.П., Хандецький А.І., Шрюфер Е. Обробка сигналів: підручник для вузів., К., Либідь, 1996.- 390с.
3. Применение цифровой обработки сигналов/ Под ред. Э.Оппенгейма.- М. Мир, 1980.- 552с.
4. Справочник по устройствам цифровой обработки информации/ Н.А.Виноградов, В.Н.Яковлев, В.В.Воскресенский и др.- К:Тэхника, 1988.- 455с.
5. Сверхбольшие интегральные схемы и современная обработка сигналов: Пер. с англ.- М.: Радио и связь, 1989.- 472с.
6. Яцимірський М. М. Швидкі алгоритми ортогональних тригонометричних перетворень. - Львів: Академічний Експрес, 1997. - 219 с.
7. www.analog.com
8. www.dsplib.ru