Быстрое преобразование Фурье для обработки сигналов в устройствах автоматизации

Сергей Лазарев, Евгений Рогожкин, Феодосий Захарук

В статье приведена программа быстрого преобразования Фурье для цифровой обработки сигналов на базе современных персональных компьютеров и встраиваемых контроллеров. Отмечается, что при оптимальном программировании алгоритмов сигнальной обработки можно во многих случаях обойтись без применения дополнительных специализированных сигнальных процессоров.

Цифровая обработка сигналов (ЦОС) является разновидностью обработки данных с помощью компьютера. В современных системах обработки реального времени для ускорения алгоритмов ЦОС зачастую применяются дополнительные модули сигнальных процессоров (DSP) типа TMS320, M56000, M96002 и другие. Они эффективно выполняют однообразные операции обработки массивов данных. Однако введение в состав систем модулей DSP приводит к дополнительным накладным расходам, связанным

- с подбором и приобретением подходящего сигнального процессора;
- с разработкой или покупкой дополнительного программного обеспечения и библиотек программ для выбранного DSP;
- с сопряжением уникального аппаратного и программного обеспечения с проектируемой системой.

Поддержка программ сигнальной обработки требует содержать инженера-аналитика — специалиста по ЦОС, инженера-специалиста по сигнальному процессору, а также программиста, разбирающегося в ЦОС.

Цифровая обработка сигналов была бы проще для программистов, если бы под рукой были готовые листинги эффективных программ для типовых операций ЦОС на языке программирования высокого уровня, например С, которые можно гибко адаптировать к возможностям проектируемых систем для решения конкретной задачи. Фирмы-разработчики программного обеспечения для сигнальных процессоров предлагают на рынок довольно громоздкие пакеты объектных библиотек программ ЦОС для DSP. Существуют проблемы с надежностью и эффективностью таких пакетов при их приспособлении для конкретных систем, особенно для систем реального времени, в которых набор необходимых программ

```
/*Листинг  программы БПФ:/*

#include <stdio.h>
#include <math.h>
#include <time.h>

BPF(x,y,N,I)                                        /*Процедура БПФ/*
register float *x,*y;                    /*x,y-входные массивы данных/*
register int N,I;              /*размерностью I=1 -БПФ I=-1 -ОБПФ/*
{
        register  float  c,s,t1,t2,t3,t4,u1,u2,u3;
        register  int i,j,p,l,L,M,M1,K;
        L=N;
        M=N/2;
        M1=N-1;
        while(L>=2){
                  l=L/2; u1=1.; u2=0.; t1=PI/(float)l;
                  c=cos(t1); s=(-1)*I*sin(t1);
                  for(j=0; j<l;j++)
                  {
                        for(i=j;i<N;i+=L)
                        {
                                 p=i+l;
                                 t1=*(x+i)+*(x+p);
                                 t2=*(y+i)+*(y+p);
```

```
                                t3=*(x+i)-*(x+p);
                                t4=*(y+i)-*(y+p);
                                *(x+p)=t3*u1-t4*u2;
                                *(y+p)=t4*u1+t3*u2;
                                *(x+i)=t1;     *(y+i)=t2;
                            }
                            u3=u1*c-u2*s;
                        u2=u2*c+u1*s;  u1=u3;
                        }
                        L/=2;
        }
        j=0;
        for(i=0;i<M1;i++)
        {
                if(i>j)
                {
                        t1=*(x+j); t2=*(y+j);
                        *(x+j)=*(x+i); *(y+j)=*(y+i);
                        *(x+i)=t1; *(y+i)=t2;
                }
                K=M;
                while(j >=K)
                {
                        j-=K;K/=2;
                }
                j+=K;
      }
}

sinsignal(P,F,A,N)              */моделирование входного сигнала/*
                                  */в форме синусоиды/*
float *P,F,A;                  */P-массив сигнала размерности  N/*
int  N;                          */F-частота сигнала,/*
                                  */A-амплитуда сигнала/*
{
         register int i;
         register float r,re,re1,im,im1;

         re=cos(2.*PI*F/(float)N);
         im=sin(2.*PI*F/(float)N);
         re1=A;im1=0.;
         for(i=0;i< N;i++)


         {
                 *(P+i)=re1;r=re1;
                 re1=r*re-im1*im;
                 im1=im1*re+r*im;
         }
}

main()
{
         int j,N;
         float *x,*y,F,A,Re,Im;

         printf("\t\t N     :");  scanf("%d",&N);
         printf("\t\t  F(gc):");  scanf("%f",&F);
         printf("\t\t A     :");  scanf("%f",&A);

         x=(float*)calloc(N,sizeof(float));
         y=(float*)calloc(N,sizeof(float));

         sinsignal(x,F,A,N);

         for(j=0;j < N;j++) printf(" X[%d] -  %.1f  \n",j,*(x+j));
         BPF(x,y,N,1);

         for(j=0;j  <    N/2;j++)
         {
            Re=*(x+j);
            Im=*(y+j);
            A=2.*sqrt(Re*Re+Im*Im)/(float)N;
            printf("  X[%d] - %d \n",j,(int)A);

         free(x); free(y);
}
```

```
Тестовый пример
N = 1024
F = 100
A = 100

Входной массив :
X[0]=   100
X[1]=   81.8
X[2]=   33.7
X[3]= - 26.7
X[4]= - 77.7
X[5]= - 99.7

.....................................
X[1018]= - 85.8
X[1019]= - 99.7
X[1020]= - 77.3
X[1021]= - 26.7
X[1022]=   33.7
X[1023]=   81.8

Выходной массив:
X[0]= 0
X[1]= 0
X[2]= 0
X[3]= 0
X[4]= 0

.....................................
X[100]= 100

.....................................
X[508]= 0
X[507]= 0
X[508]= 0
X[509]= 0
X[510]= 0
X[511]= 0
```

Таблица 1. Время выполнения типовой операции ЦОС — БПФ (мс) в зависимости от вычислительного модуля, среды исполнения и объема входных данных (N) для приведенного листинга программы

Тип процессора с сопроцессором	М 68030 33 МГц	М 68040 25 МГц	М 68060 60 МГц	Pentium MMX 166 МГц	P-2 300 МГц
Носитель	Беста-88	MVME-162	MVME-172		
Язык программирования	ANSI C	Ultra C	Ultra C	Visual C++ debug	Visual C++ debug
Операционная система	OS-9 V 2.1	OS-9 V 3.0	OS-9 V 3.0	Window 95	WindowsNT
Тип данных	Float	Float	Float	Float	Float
Объем памяти хранения данных (байт)	8xN	8xN	8xN	8xN	8xN
Объем программного кода (байт)	22672	22672	22672	126464	126464
N = 128	24,3 мс	3,2	0,8	7,7	1,3
256	53,8	6,3	1,4	9,9	3,1
512	117,6	13,4	2,9	18,0	7,2
1024	260,4	30,7	6,6	33,5	16,3
2048	568,0	66,7	27,1	72,0	37,0
4096	1230,4	142,3	64,0	149,0	83,2
8192	2652,0	305,6	141,7	334,0	186,7
16384	5695,0	650,0	307,4	739,0	412,8

Таблица 2. Время выполнения БПФ (мс) с алгоритмом одновременного вычисления сразу двух массивов входных данных размерностью N в пересчете на один массив

Тип процессора с сопроцессором		M 68030 33 МГц	M 68040 25 МГц	M 68060 60 МГц	Pentium MMX 166 Мгц	P-2 300 МГц
Объем памяти данных (байт)		16 x N	16 x N	16 x N	16 x N	16 x N
N =	128	5	1,2	0,3	3,4	0,6
	256	10	3,2	0,7	8,2	1,6
	512	40	6,0	1,3	9,0	3,3
	1024	80	14,1	3,1	17,0	8,2
	2048	200	24,8	10,0	34,7	19,0
	4096	450	49,7	22,3	72,3	42,0
	8192	950	108,0	49,3	170,1	87,0
	16348	2100	228,0	107,8	360,0	205,4

ЦОС может составить всего 5% от предлагаемого. Хотя платить придется за весь пакет.

С появлением мощных сопроцессоров с плавающей запятой, стандартно входящих в состав процессоров обработки данных, а также с решением проблем по объему оперативной памяти ситуация может кардинально измениться для многих приложений не в пользу модулей сигнальных процессоров.

Типовые операции цифровой обработки сигналов могут быть эффективно выполнены с помощью сопроцессоров без использования дорогостоящих модулей DSP. Достаточно написать С-код программы ЦОС и обычным способом внедрить его в код приложения, предварительно убедившись, что быстродействие алгоритмов ЦОС достаточно для правильной работы проекта в целом. От инженера-разработчика требуется лишь правильно скомпоновать последовательность С-функций и корректно внедрить их в проект.

Эффективность алгоритмов ЦОС при подготовке программ на языке программирования высокого уровня можно повысить двумя путями. Первый — это написание эффективного кода, связанного с особенностями языка программирования, компилятора и возможностями процессора. Второй путь — это постоянно отслеживать достижения математики в области абстрактной алгебры, теории групп, индексной арифметики, что позволяет только программными методами повысить эффективность ЦОС во много раз. При недостатке специальных сборников программ ЦОС для большинства программистов этот раздел обработки данных остается вещью в себе.

Во врезке статьи приводится листинг типовой процедуры ЦОС на языке С, реализующей алгоритм быстрого преобразования Фурье (БПФ), используемый в системах спектральной обработки и фильтрации данных. Программа выполняет вычисление дискретного комплексного спектра сигналов, представленных массивом оцифрованных отсчетов определенной размерности.

Для большинства приложений важна скорость выполнения БПФ. В таблице 1 приводятся результаты измерения времени выполнения этой программы для современных контроллеров и ПЭВМ.

В табл. 2 и табл. 3 показана возможность повышения быстродействия типовой операции БПФ для тех же самых процессоров, когда ускорение вычислений достигается только за счет программно-вычислительных трюков и знания особенностей этих процессоров и компилятора.

Анализ представленных таблиц наглядно убеждает в наличии зависимости времени исполнения как от алгоритма выполнения БПФ, так и от искушенности программиста. Чисто программными методами можно добиться того, что выполнение операций ЦОС на микропроцессорах общего назначения будет происходить со скоростью, соизмеримой с возможностями доступного сигнального процессора. ●

Листинг приведенной в статье программы можно загрузить с Web-сайта www.cta.ru

Таблица 3. Время исполнения процедуры БПФ для процессора P-2/300 для различных типов данных, параметров компилятора, операционных систем и объема входных данных N

Операционная система		Window NT	Window NT	Window NT	Window 95	Window NT
Среда		Visual C ++ Debug	Visual C ++ Debug optim	Visual C ++ Speed optim	Visual C ++ Speed optim	Visual C ++ Speed
Тип данных		Float	double	double	double	double
Объем памяти данных (байт)		8 x N	16 x N	16 x N	32 x N	32 x N
N =	128	1,3 мс	0,2	0,15	0,082	0,026
	256	3,1	0,4	0,2	0,18	0,05
	512	7,2	0,9	0,35	0,34	0,12
	1024	16,3	1,9	0,8	0,81	0,32
	2048	37,0	5,3	3,1	2,57	1,33
	4096	83,2	14,9	10,0	5,47	3,92
	8192	186,3	32,0	22,3	12,0	8,6
	16384	412,8	70,0	48,7	27,7	23,8