TMS320F2812 DSP 的 FFT 运算和 DCT 实现*

艾 红,邓大伟

(北京信息科技大学 自动化学院,北京 100192)

摘 要:介绍了快速傅里叶变换(FFT)算法的原理,利用DSP实现了FFT算法,利用TMS320F2812 DSP内部的ADC模块与事件管理器的定时器实现信号的实时采集。分析了DSP中数据采集ADC的功能。基于CCS调试软件显示了输入输出信号波形。在CCS环境下,采用C语言编程,实现了FFT算法和离散余弦变换。

关键词:DSP;快速傅里叶变换;A/D转换;离散余弦变换

中图分类号:TP332 **文献标识码**:A **文章编号**:1674-7720(2012)09-0020-04

Realization of FFT algorithm and DCT based on TMS320F2812 DSP

Ai Hong, Deng Dawei

(Automation Institute, Beijing Information Science&Technology University, Beijing 100192, China)

Abstract: The principle of Fast Fourier Transform (FFT) algorithm is described in this paper. FFT algorithm is realized by using of DSP. The system uses the internal ADC module and event-manager's general timer of TMS320F2812 DSP to achieve the real-time signal acquisition. ADC function of data collection is analyzed. The waveforms of input and output signals for FFT operations are displayed based on CCS debugging software. C language programming is used in environment of CCS, and FFT algorithm and DCT are realized.

Key words: DSP; FFT; A/D conversion; DCT

傅里叶变换是一种将信号从时域变换到频域的变换方式,而快速傅里叶变换FFT(Fast Fourier Transform)是数字信号处理技术的基石。FFT和离散余弦变换DCT(Discrete Cosine Transform)都是数字信号处理技术中的基本算法,也是数字信号处理的基本工具。DSP芯片的出现使FFT和DCT的实现更为方便。本文利用TMS320F2812 DSP内部的ADC模块与事件管理器(EVA)构建了数据采集与数据变换并行处理的信号处理系统,充分利用TMS320F2812强大的数据处理能力,实现了FFT运算,提高了运算速度[1-2]。

1 FFT算法的实现

TI公司的TMS320F2812 DSP是目前控制领域性能较高的处理器,它将各种高级数字控制功能集成于一块芯片上,整合了Flash存储器、快速的A/D转换器等外设,强大的数据处理和控制能力大幅度提高了应用效率。

*基金项目:北京市教育委员会科技计划面上项目(KM200910772008)

1.1 数据采集ADC功能

DSP系统的模拟输入电压范围为0~3 V。通过使用事件管理器的定时器1下溢中断启动ADC。系统设计时晶振为30 MHz,经过锁相环倍频后CPU时钟频率SYSCLKOUT是150 MHz,事件管理器采用高速外设时钟HSPCLK,经过程序设计6分频得到高速外设时钟HSPCLK为25 MHz。

```
SysCtrlRegs.HISPCP.all=0x3;  //HSPCLK= SYSCLKOUT/6
```

将事件管理器中通用定时器1的周期寄存器值设置为0x07FF,每经过2 048(0x07FF+1)个通用定时器的时钟周期启动一次ADC。事件管理器中的通用定时器1由于没有对高速外设时钟分频,因此通用定时器1的时钟频率为25 MHz。

```
EvaRegs.T1PR=0x07FF;//设置通用定时器1周期寄存器
EvaRegs.GPTCONA.bit.T1TOADC=1;
                              //通用定时器1启动ADC
EvaRegs.T1CON.all=0x1042;
```

//通用定时器1连续增计数模式,不分频,采用HSPCLK

为了实现数据采集,设置ADC工作在级联排序器模式,最大转换通道数为1,并且采集数据来自通道ADCINA4,使能事件管理器EVA的触发信号启动ADC排序器SEQ1,允许ADC产生中断。相关程序设计如下。

```
AdcRegs.ADCTRL1.bit.SEQ_CASC=1;  //级联排序器模式
AdcRegs.ADCMAXCONV.all=0x0000;
//设置1个转换通道 AdcRegs.ADCCHSELSEQ1.bit.CONV00=0x4;
                                  //设置转换通道 ADCINA4
AdcRegs.ADCTRL2.bit.EVA_SOC_SEQ1=1;
                          //使能 EVA 的触发信号启动排序器 SEQ1
AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1=1;
                                  //使能 SEQ1 产生的中断请求
```

当事件管理器的通用定时器1产生下溢中断时,启动ADC。在ADC转换完成中断服务程序中读取12 bit A/D转换结果。程序设计如下:

```
interrupt  void  adc_isr(void)
{
 px[ConversionCount]=AdcRegs.ADCRESULT0>>4;
 if(ConversionCount==128)
   {ConversionCount=0;}
 else    ConversionCount++;
  AdcRegs.ADCTRL2.bit.RST_SEQ1=1;
                                        //复位排序器 SEQ1
  AdcRegs.ADCST.bit.INT_SEQ1_CLR=1;
                                //清除排序器 SEQ1 中断标志位
  PieCtrlRegs.PIEACK.all=PIEACK_GROUP1;
          //写1清零中断应答寄存器 PIEACK 相应位,
                                //以便能够响应该组随后的中断
  return;
}
```

1.2 FFT算法原理与程序流程图

FFT是DFT的快速运算。由于有些信号在时域很难看出特性,使用FFT将其变换到频域,就会很容易看出其特性。DFT算法的基本公式为:

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn},k=0,1,\cdots,N-1 \quad (1)$$

其中,$x(n)$表示时域信号,$X(k)$表示频域信号,W_N^{kn}为运算蝶式权。

FFT算法程序的基本流程如图1所示。

首先需要对时域序列进行比特排序,即接收处理单元把放在数据空间中的每个采样点按地址读出,按比特逆序再放入数据空间,准备进行运算[3]。

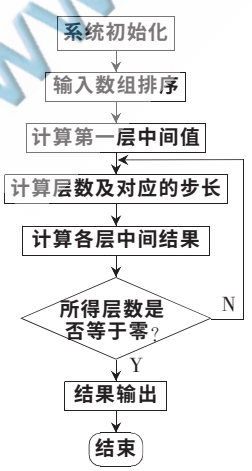

图1 FFT算法基本流程图

然后计算蝶式运算的重要元素:运算蝶式权W_N^{kn}。此处可充分利用W_N^{kn}的周期性,减少运算量,节省DSP的存储空间。以8点FFT为例,因为有$W_2^0=W_4^0=W_8^0$,$W_4^1=W_8^2$,则计算8点FFT所有的运算蝶式权,只需将第三级即最后一级的运算蝶式权W_8^0、W_8^1、W_8^2算出即可。所以对于N($N=2^M$)点FFT来说,只需计算出第M级的M个运算蝶式权即可得到各级的运算蝶式权。所有的FFT运算,第一级权值为1,所以第一级权值不用计算,仅此一项就可大大减少运算量。W_N^{kn}可以分为实部和虚部两部分进行计算,即:$W_N^{kn}=\cos(2\pi K/N)-j\sin(2\pi K/N)$。在FFT中,每级的蝶式运算都具有不同数量的蝶群和不同的翅间距(第M级的翅间距为$2M-1$),这些都需要定义相应的变量来控制。

1.3 FFT算法程序运行结果

调整模拟信号的频率和幅值,通过A/D采集可以看到输入信号波形及其FFT算法程序执行结果,如图2和图3所示。

图2 时域正弦波的FFT运行结果1

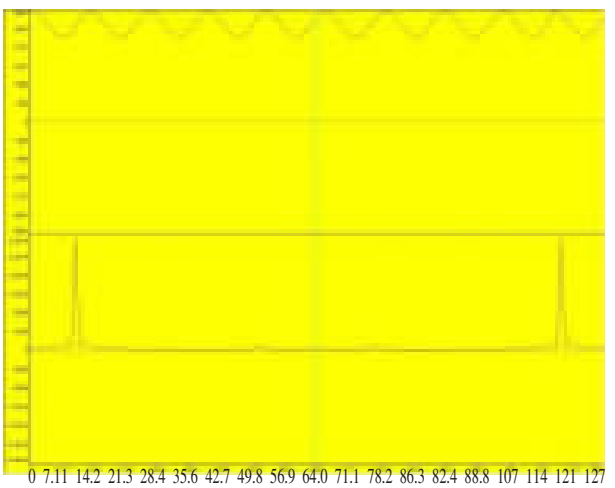

图3 时域正弦波的FFT运行结果2

2 DCT 的实现

2.1 DCT 基本原理

DCT 是一种与傅里叶变换紧密相关的数学运算。在傅里叶级数展开式中，如果被展开的函数是实偶函数，则其傅里叶级数中只包含余弦项，再将其离散化可导出余弦变换，因此称之为离散余弦变换。DCT 被认为是性能接近 K-L 变换的准最佳变换，是对语音和图像信号进行变换的最佳方法。DCT 变换的快速算法有以下两种方式：

(1)由于 FFT 算法的普遍采用，直接利用 FFT 实现 DCT 变换的快速算法相对容易。但是这种方法也有不足之处，即计算过程会涉及复数的运算。由于 DCT 变换前后的数据都是实数，计算过程中引入了复数，而一对复数的加法相当于两对实数的加法，一对复数的乘法相当于 4 对实数的乘法和两对实数的加法，显然是增加了运算量，也给硬件存储提出了更高的要求。

(2)直接在实数域进行 DCT 快速变换。显然，这种方法的计算量和硬件要求都要优于前者。鉴于此，本文采用第二种方法实现 DCT 变换的快速算法。

给定序列 $x(n)$，$n=0,1,\cdots,N-1$，其离散余弦变换定义为：

$$X_c(0)=\frac{1}{\sqrt{N}}\sum_{n=1}^{N-1}x(n)$$

$$X_c(k)=\sqrt{\frac{2}{N}}\sum_{n=1}^{N-1}x(n)\cos\frac{(2n+1)k\pi}{2N},k=1,2,\cdots,N-1$$

显然，其变换的核函数

$$C_{k,n}=\sqrt{\frac{2}{N}}\sum_{n=1}^{N-1}g_k\cos\frac{(2n+1)k\pi}{2N},k,n=0,1,\cdots,N-1$$

为实数，其中系数 $g_k=\begin{cases}1/\sqrt{2}, & k=0\\ 1, & k\neq 0\end{cases}$

逆变换为：

$$x(n)=\frac{1}{\sqrt{N}}X(0)+\sqrt{\frac{2}{N}}\sum_{K=1}^{N-1}X(K)\cos\left[\frac{\pi}{2N}(2x+1)k\right]$$

其中 $k,n=0,1,\cdots,N-1$。

2.2 DCT 程序设计

DCT 程序设计流程图如图 4 所示。

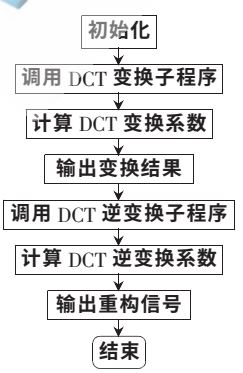

图 4 DCT 流程设计流程图

程序先将一序列 $x(n)$ 作为输入信号进行离散余弦的正变换，再将得到的结果进行离散余弦逆变换，从而还原出输入序列 $x(n)$。程序实现的是 128 点的DCT 变换。程序设计中，px 是正变换的输入序列，pz 是逆变换的输出序列。x、y 和 z 是 3 个中间变量，在正变换的子程序中，x 是输入序列，y 是输出序列；在逆变换的子程序中，y 是输入序列，z 是输出序列。程序设计如下：

```
#define pi 3.1415927
int px[128];                                          //输入序列
int pz[128];                                          //输出序列
double x[128],y[128],z[128];
int n=128;
void dct1c2 (double x[],double y[],int n)
  {   double s,t;
      int i,j;
      s=0.0;
    for (i=0; i<=n-1; i++)
      { s=s+x[i];}
    for (i=1; i<=n-1; i++)
    { t=0.0;
      for (j=0; j<=n-1; j++)
        {t=t+x[j]*cos((2.0*j+1)*i*pi/(2.0*n));}
      y[i]=sqrt(2.0/n)*t;}
    y[0]=s/(sqrt(n/1.0));}                            //DCT 正变换子程序
void idct1c2(double y[], double z[], int n)
{   double t;
    int i,j;
    for (i=0; i<=n-1; i++)
      {t=0.0;
    for (j=0; j<=n-1; j++)
      {t=t+y[j]*cos((2.0*i+1)*j*pi/(2.0*n));}
      z[i]=y[0]/(sqrt(n/1.0))+sqrt(2.0/n)*t;}}
                                                      //DCT 逆变换子程序
void main(void)
{   int k=0;
    int i;
    int xm,zm;
    double xmid=0;
for(;;)
  {  xmid=0;
     for (i=0; i<=127; i++)
     {   xm = px[i];
         xmid=xmid+xm;
         x[i] = xm;}
     xmid=xmid/128.0;
     for(i=0;i<128;i++)
     { x[i]=x[i]-xmid;}                               //压缩数据，防止溢出
     dct1c2 (x,y,n);                                  //进行 DCT 正变换
     idct1c2 (y,z,n);                                 //进行 DCT 逆变换
     for (i=0; i<=127; i++)
     {   zm=(int)z[i];                                //输出变换结果
         pz[i] = zm;}                                 //输出重构信号
    k++; }}
```

2.3 运行结果

设置观察窗口，可以看到 DCT 变换的输入输出信号，如图 5 和图 6 所示。

图 5 为正变换结果，其中上方为输入信号，下方为

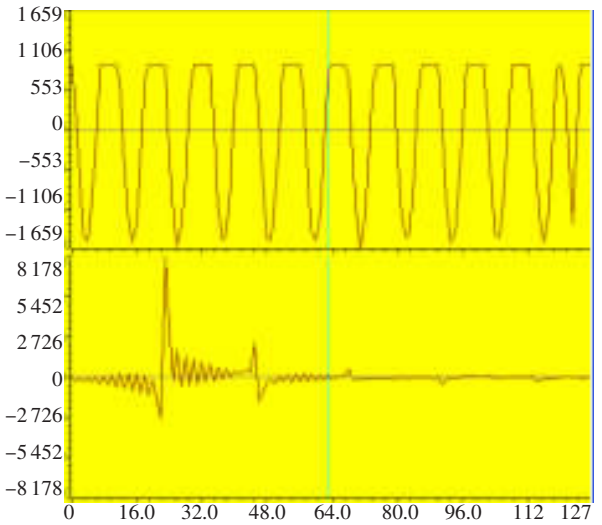

图 5 DCT 正变换结果

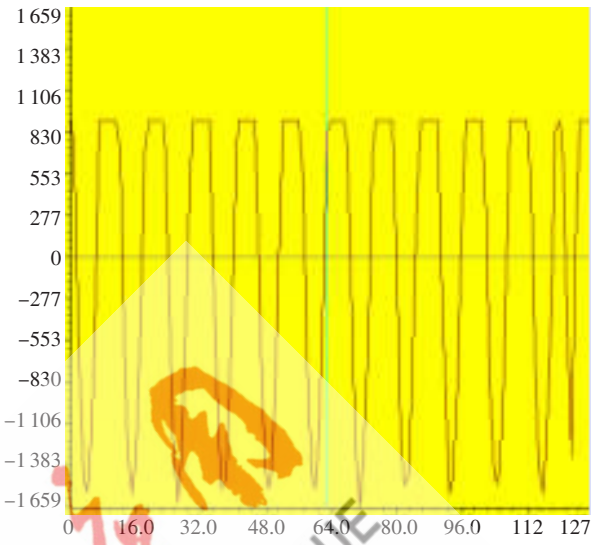

图 6 DCT 逆变换结果

输出信号。图 6 为逆变换输出结果，此输出波形与图 5 的输入波形一致，由此可以验证程序的正确性。

本文说明了数据采集 ADC 的功能和 FFT 算法的原理以及程序设计流程图，在 CCS 调试平台下，采用 C 语言编程实现了 FFT 算法，并且实时性好。阐述了离散余弦变换 DCT 基本原理，基于 TMS320F2812 DSP 实现了离散余弦变换。程序运行结果表明，DSP 能够快速高效地完成一系列数字信号处理算法[4]。

参考文献

[1] 贾玮，杨录，张艳花.基于 TMS320VC5416 的 FFT 算法的实现[J].山西电子技术，2009(2)：11-13.

[2] 万浩平，马进，王锋.基于 TMS320F2812 的高精度数据采集及 FFT 实现[J].工业控制计算机，2009，22(4)，54-55.

[3] 胡广书. 数字信号处理 [M]. 北京：清华大学出版社，2003.

[4] 伍小芹，吴秋丽.FIR 数字滤波器在 DSP 上的实现[J].现代电子技术，2007(1)：85-87.

（收稿日期：2011-11-12）

作者简介：

艾红，女，1962 年生，硕士，副教授，硕士生导师，主要研究方向：检测技术与自动化装置。