基于 ARM920T 内核的 FFT 算法的高效实现

李宏佳,魏权利
LI Hong-jia,WEI Quan-li

青岛科技大学 信息科学技术学院,山东 青岛 266042
College of Information and Scientific Technology,Qingdao University of Science and Technology,Qingdao,Shandong 266042,China
E-mail:Prince_lhj@163.com

LI Hong-jia,WEI Quan-li.High-efficient implementation of FFT algorithm based on ARM920T core.Computer Engineering and Applications,2008,44(3):114-116.

Abstract: With the development of the architecture of ARM,ARM processor can be used in many DSP applications.In order to thoroughly explore the ability of DSP in ARM processor,the high-efficient ARM program of radical 4-FFT is designed according to the architecture characters of ARM core.In the process of code designing,instruction scheduling and register allocation are controlled subtly and the fixed-point number's storage format and computing method of float number are provided.Barrel shifter and 5-level pipeline are fully utilized and pipeline interlock is avoided.The experimental results indicate that the instruction cycles of optimized code are greatly shortened and the result is more precise.These methods of code optimization have practical meaning to optimization of ARM programs.

Key words: code optimization;ARM core;pipeline interlock;FFT;floating number

摘 要:随着 ARM 体系结构的发展,ARM 处理器已经可以胜任许多 DSP 应用。为了充分挖掘 ARM 处理器数字信号处理能力,结合 ARM 内核设计特点设计了基 4-FFT 算法的高效 ARM 程序。代码设计中,对寄存器分配和指令调度作了精细地控制,提出了 ARM 汇编中浮点数的定点格式存储和计算方法,充分利用桶形移位器和 5 级流水线,避免了流水线互锁问题。实验结果表明优化后的程序指令周期总数减少并且运算精度很高。这些优化方法对 ARM 程序优化具有实际指导意义。

关键词:代码优化;ARM 内核;流水线互锁;FFT;浮点数
文章编号:1002-8331(2008)03-0114-03 **文献标识码**:A **中图分类号**:TN313

1 引言

“十五”国防预研项目提出了“可穿戴计算机系统”,这对嵌入式处理器提出了更高的要求,既要有高速数字信号处理能力又要求有丰富的外围控制器,传统需要数字信号处理的嵌入式或便携设备一般包括双处理器:一个微控制器处理用户接口而另一个独立的 DSP 处理器处理数字信号,也就是“双核方案”。随着 ARM 体系结构的加强和主频的不断提高使得 ARM 可以很好地适应许多 DSP 应用,并且 ARM 处理器具有丰富的外围控制器,开发者可以使用单 ARM 处理器完成双核的任务,这种单核设计能够减少费用和降低功耗。快速傅立叶变换(FFT)是数字信号处理中最为重要的工具之一, 所以本文以 FFT 算法在 ARM 处理器中高效的实现体现 ARM 内核在 DSP 处理中的能力。本文采用 ARM 汇编语言实现 FFT 算法,对寄存器分配和指令调度进行了精细的控制,并且提出了 ARM 汇编中浮点数定点格式的存储和计算方法,充分利用桶形移位器和 5 级流水线,避免了流水线互锁(pipeline interlocks)问题。

2 FFT 算法概述

2.1 各种算法的比较

Winograd 快速傅立叶变换算法(WFTA)和素因子(Prime Factor Algorithm,PFA)是将 DFT 转变为卷积,利用计算卷积的方法计算,Cooley-Tukey 算法、Rader-Brenner 算法和分裂基算法等则是递归型算法, 将一维 DFT 化为容易计算的二维或多维 DFT,且这个过程可以重复,相比较而言,PFA 和 WFTA 在运算量上占优,所用乘法数比 Cooley-Tukey 算法少,但控制复杂,控制单元实现起来相对麻烦。分裂基算法也具有一定的优势,综合了基 4 和基 2 的运算特点,但其 L 型蝶式运算结构在控制上要复杂一些[1]。而在基 4 和基 2 算法软件实现的比较中,假设采样数据为 $N=4^M$(M 为自然数),采用基 2 算法需要乘法次数 MN,而采用基 4 算法所需乘法的次数为 $3MN/4$,显然可以节省 25%的乘法,同理采用基 8 算法可以进一步节省乘法的次数,但是由于 ARM920T 只有 14 可以利用的寄存器,使用基 8 算法每次蝶形运算需要大量从内存装载和存储数据的操作,总体来看减少乘法次数获得的效率被存储器访问所抵消,所以采用基 4 算法是 ARM920T 最佳的选择,故本文采用基 4 算法。

2.2 基 4-DIT 的 FFT 算法与程序实现

离散傅立叶变换将时域信息转换为频域信息,而反变换则将频域信息转换为时域信息。长度为 $N=4^M$ 的系列 $x(n)$ 的离散傅立叶变换可写为:

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk},\quad k,n=0,1,2,\cdots,N-1 \tag{1}$$

作者简介:李宏佳(1981-),男,硕士研究生,主要研究领域为嵌入式系统、无线数据通信;魏权利(1960-),男,教授,主要研究领域为嵌入式系统、计算机网络。

基 4 按时域抽取(DIT)是在时域将 $x(n)$分解为:

$$x_m(r)=x(4r+m)\quad 0\leqslant m\leqslant 3,0\leqslant r\leqslant \frac{N}{4}-1 \tag{2}$$

其中,子序列 $x_m(r)$均为$\frac{N}{4}$点序列,又设 $x_m(r)$的 DFT 为 $X_m(r)$,则:

$$\begin{cases} X(r)=X_0(r)+W_N^{r}X_1(r)+W_N^{2r}X_2(r)+W_N^{3r}X_3(r) \\ X(r+\frac{N}{4})=X_0(r)+W_N^{r+\frac{N}{4}}X_1(r)+W_N^{2(r+\frac{N}{4})}X_2(r)+W_N^{3(r+\frac{N}{4})}X_3(r) \\ X(r+\frac{N}{2})=X_0(r)+W_N^{r+\frac{N}{2}}X_1(r)+W_N^{2(r+\frac{N}{2})}X_2(r)+W_N^{3(r+\frac{N}{2})}X_3(r) \\ X(r+\frac{3N}{4})=X_0(r)+W_N^{r+\frac{3N}{4}}X_1(r)+W_N^{2(r+\frac{3N}{4})}X_2(r)+W_N^{3(r+\frac{3N}{4})}X_3(r) \end{cases}$$

$0\leqslant r\leqslant \frac{N}{4}-1$

又因为 $W_N^{\frac{N}{4}}=-j,W_N^{\frac{N}{2}}=-1,W_N^{\frac{3N}{4}}=j,W_N^{\frac{6N}{4}}=-1,W_N^{\frac{9N}{4}}=-j,W_N^{N}=1$,所以上式可化为:

$$\begin{pmatrix} X(r) \\ X(r+\frac{N}{4}) \\ X(r+\frac{N}{2}) \\ X(r+\frac{3N}{4}) \end{pmatrix}=\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & j \end{pmatrix}\begin{pmatrix} X_0(r) \\ W_N^{r}X_1(r) \\ W_N^{2r}X_2(r) \\ W_N^{3r}X_3(r) \end{pmatrix} \tag{3}$$

由上式可以得到基 4 蝶形单元,因为 $X_0(r),X_1(r),X_2(r),X_3(r)$中 $r=4^{M-1}$,所以将它们继续分解 $M-1$ 次,也就是说整个基 4-FFT 运算共有 M 级,每一级 $N/4$ 个蝶形单元。根据文献[2,6]的推导,知道要使用同址运算,对输入数据必须二进制倒序排列。

根据上述思想将 FFT 实现程序分为了两个步,第一步完成倒序并在其中完成第一级蝶形运算,第二步进行剩余的 $M-1$ 级蝶形运算。考虑代码精简问题,将两步中重复使用的运算以宏的形式实现,如表 1 所示共包括 4 个宏,分别是蝶形单元运算宏 FFT4、复数装载宏 CXLDR 和 CXSTR 存储宏以及复数共轭乘法宏 CXMUL。

以上对 FFT 算法和程序结构进行了分析,但是要在 ARM 处理器中高效地实现仍有许多问题需要解决。

3　高效实现方法

3.1　信号的表示

由于 ARM920T 内核不支持浮点,ARM 汇编程序中无法直接对浮点数直接计算,因此程序中选择了定点表示浮点数的“移位化整算法”,这种算法的思想是根据计算中所需精度的要求将浮点数左移,然后根据舍入算法保留整数部分,这种算法的数学描述为:设 x 为浮点形式的信号值,X 为 x 的整型表示,k 为达到误差要求需要的移位次数,那么:$X=\begin{cases}\lfloor|2^k x|+\alpha\rfloor, & x>0\\ -\lfloor|2^k x|+\alpha\rfloor, & x<0\end{cases}$,其中 α 为精度调整因子,通过实验本文取 α=0.5;“$\lfloor x\rfloor$”表示取小于等于 x 的最大整数;k 次移位可以利用 ARM 核中的移位器完成,根据 ARM 指令的特点移位操作可以作为四则运算指令的一部分,而不占用额外的指令周期,这极大地提高了“移位化整算法”的效率。

在整个 FFT 运算中用到加、减和乘运算,使用移位化整算法后的算术运算表达为:

加减:$Y=A\pm B=2^k\times(a\pm b)=(a\pm b)<<k$

乘法:$Y=A\times B=2^k a\times 2^k b=(a\times b)<<2k$

可以看出乘法运算的结果 Y 超出了 k 次移位所表示的精度,为了保持精度范围,在乘法运算后对结果要右移 k 位。

移位化整算法实现的 c 语言的程序如下所示:

```
int CxToInt (double x,int shifter){//其中 x 为输入浮点数,shifter 为移位数
    int scale=1<<shifter;
    x=x*(double)(shifter);
    x+=(x>=0)? 0.5:-0.5;
    if(x>=0x7FFF)x=0x7FFF;
    return(int)x;
}
```

3.2　旋转因子表

为了提高 FFT 的运算速度,预先计算出旋转因子表的形式存储。由式(3)得旋转因子 $W_N^k=\exp(-j2\pi/N)=\cos(2k\pi)-j\sin(2k\pi/N)$,其中 N 的取值为当前计算点数,第二级为 N=16,其旋转因子为 $W_{16}^0,W_{16}^1,W_{16}^2,W_{16}^3$;第三级为 N=64,其旋转因子为 $W_{64}^0,W_{64}^1,W_{64}^2,\cdots,W_{64}^{15}$;第四级为 N=256,其旋转因子为 $W_{256}^0,W_{256}^1,W_{256}^2,\cdots,W_{256}^{63}$,以此类推。

令与旋转因子相乘的复数 $a+jb$,设 $c=\cos(\frac{2k\pi}{N})$,$s=\sin(\frac{2k\pi}{N})$,则得复数 $a+jb$ 与旋转因子相乘结果为:

$(a+jb)(c-js)=[(b-a)s+a(c+s)]+j[(b-a)s+b(c-s)]$

$(a+jb)(c+js)=[(a-b)s+a(c-s)]+j[(a-b)s+b(c+s)]$

根据上式,给定 $c-s$、s、a 和 b 四个值,可以通过三次实乘

表 1　宏定义表

宏名称及调用形式	实现算法描述(“<<”表示左移,“>>”表示右移)	说明
FFT4:蝶形单元运算宏 FFT4　$s	$x_0=(x_0+(x_2>>s)+(x_1>>s)+(x_3>>s))$ $x_1=(x_0-j*(x_2>>s)-(x_1>>s)+j*(x_3>>s))$ $x_2=(x_0-(x_2>>s)+(x_1>>s)-(x_3>>s))$ $z_3=(x_0+j*(x_2>>s)-(x_1>>s)-j*(x_3>>s))$	s 为移位因子;x_0,x_1,x_2,x_3 是将进行 FFT4 运算的输入数据;输出结果存储在 x_0,x_1,x_2,z_3 中。因为 x_1,x_2,x_3 在输入前与旋转因子相乘,根据“移位化整算法”乘法的运算规则乘法运算结果需要在下一步运算前右移 s 位,“移位化整算法”见 3.1 节
CXLDR:复数装载宏 CXLDR　$x,$a,$*offset*	$x=[a],a+=offset$	x 是输出数据寄存器;a 是数据存储首地址;*offset* 是偏移地址
CXSTR:复数存储宏 CXSTR　$x,$a,$*offset*	$[a]=x,a+=offset$	x 是输入数据寄存器;a 是数据存储首地址;*offset* 是偏移地址
CXMUL:复数共轭乘法宏 CXMUL　$a,$x,$w	$a=(x_r+j*x_i)*(c_r-j*c_i)$ $x=x_r+j*x_i$ $w=(c_r-c_i)+j*c_i$	a 是输出数据寄存器;x 是输入数据;w 是旋转因子表项。调用中 a 与 w 使用同一寄存器以节省 ARM 寄存器的使用。旋转因子表见 3.2 节

计算一次复乘,其中 $c+s$ 可以通过$(c-s)+s\times2$ 得到,所以旋转因子设计为以$(c-s,s)$为单位。考虑到计算基 4 单位蝶形运算时采用了展开为基 2 运算的方法,为了保证基 2 同址运算,所以必须将旋转因子倒序,按 $E(3t)$、$E(t)$、$E(2t)$为一组存储。例如 $N=16$,$E(t)=(\cos t-\sin t)+j\sin t$,其中 $t=2k\pi/N$,$k=0,1,2,\cdots,N/4-1$,实部与虚部各以一个字存储,采用 *shifter*=14 的整型为:

```
DCW 0x4000,0x0000,0x4000,0x0000,0x4000,0x0000
DCW 0x22a3,0x187e,0x0000,0x2d41,0xdd5d,0x3b21
DCW 0xdd5d,0x3b21,0xa57e,0x2d41,0xdd5d,0xe782
```

3.3　流水线互锁

指令的执行时间依赖于流水线,ARM920T 采用 5 级流水线,它可以并行处理取指、译码、执行、存储转载($L1$)和字节、半字数据调整($L2$)。若指令时序安排的不合理,当前指令需要前一指令的结果,而前一条指令未执行完,那么处理器就要等待,这就称为流水线互锁。例如,装载指令 LDR 后,异或指令 EOR 计算使用 LDR 装载的参数,就会出现如下图 1 所示的情况,在 ALU 阶段的 EOR 指令等待 LDR 指令在 $L1$ 阶段完成数据装载,从而 EOR 指令在 ALU 阶段停留一个周期。

取址	译码	执行	$L1$	$L2$
…	EOR	LDR	…	…
…	…	EOR	LDR	…
…	…	EOR	空	LDR

图 1　ARM920T **流水线执行过程**

由于 FFT 运算中需要大量数据读取和存储操作，所以克服数据转载引起的流水线互锁可以节省大量的程序执行时间。在程序设计中,使用了连续装载和存储的技术,即多条数据存取指令连续执行,以充分利用 ARM 的流水线。

3.4　倒序算法的实现

倒序是实现同址运算的条件,本文采用倒序二进制加法的方法实现输入数据的倒序重排,基本思想是将 LSB 看作 MSB,然后进行二进制加法，例如,$1000B+1000B=0100B$,$0100B+1100B=1010B$。ARM 汇编程序实现如下:

```
MOV  counter,#0;counter 是位反转计数
first_step
     ADD  addr,x,counter,LSL#2;x 是输入数据存储首地址
     ……;以 x0,x2,x1,x3 的顺序装载数据,完成第一级的 FFT 运算并存储结果
     EOR  counter,counter,N,LSR#3;N 为采用点数
     TST  counter,N,LSR#3
     BNE  first_step
     EOR  counter,counter,N,LSR#4
     TST  counter,N,LSR#4
     BNE  first_step
     MOV  addr,N,LSR#5
bit_reversed
     EOR  counter,counter,addr
     TST  counter,addr
     BNE  first_step
     MOVS addr,addr,LSR#1
     BNE  bit_reversed
```

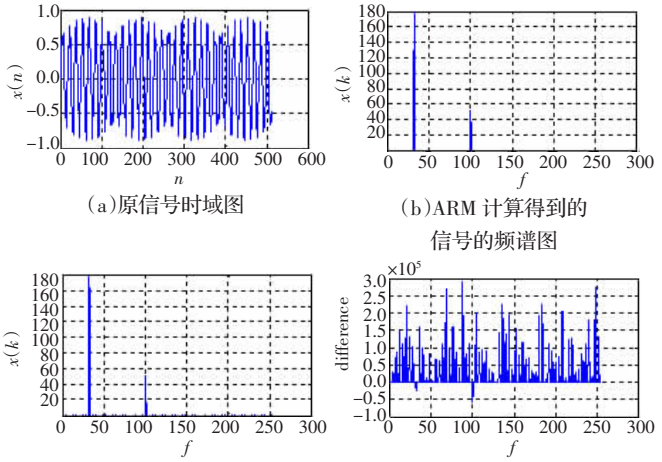
4　实验与测试结果

在 ADS1.2 编译器下采用 ARM920T 内核，采用优化方法后实现的基 4 的 FFT 程序与文献[5]中的 C 语言实现的基 4 的 FFT 程序比较结果如表 2 所示。从表中可以看出,在同样采样点数的条件下,优化的 ARM 汇编程序上的指令周期明显少于未优化的 C 语言程序,并且所需指令周期总数至少减少了20%。

表 2　**指令周期比较表**

采样点数	ARM 汇编使用指令周期	C 语言使用指令周期
64	3 524	5 664
256	19 514	28 074
1 024	99 946	134 186
4 096	487 632	624 592

对信号 $0.7\times\sin(2\times\pi\times100t)+0.2\times\sin(2\times\pi\times32t)$以采样频率 fs=512 Hz 采样,取移位因子 *shifter*=14,将设计的 FFT 程序在基于 ARM920T 内核的处理器 S3C2410 中运行,通过串口获得的运算结果与 MATLAB 仿真得到结果比较如图 2 所示，其中图 2(a)为原信号的时域图,图 2(b)与图 2(c)分别为 ARM 和 MATLAB 对原信号 FFT 变换的频谱图,图 2(d)是图 2(b)与图 2(c)的各点幅值差,显然采用“移位化整算法”的计算结果的绝对误差小于 3×10^{-5},而相对误差小于 1.67×10^{-7},这表明“移位化整算法”对信号的表示满足了高精度和高效率的要求。



(a)原信号时域图　(b)ARM 计算得到的信号的频谱图

(c)MATLAB 计算得到的信号的频谱图　(d)误差计算

图 2　FFT **变换精度比较结果**

5　结论

本文对 ARM 嵌入式系统特别是基于 ARM920T 内核的 S3C2410 处理器的代码优化作了深入地探索,提出了数字信号处理中最为重要的工具之一的 FFT 算法在其上高效的实现方法,这些方法对于其它 ARM 程序的设计也是有益的指导。通过实验测试可以看出 ARM920T 单核可以较好的胜任以往双核的工作,并且随着 ARM9E、ARM10E 等内核的出现,ARM 核在逐渐加强对 DSP 处理的支持,这对降低嵌入式设备的功耗、体积和价格是十分有利的。(收稿日期:2007 年 8 月)

参考文献:

[1] 韩泽耀,韩雁,郑为民.一种高速实时定点 FFT 处理器的设计[J].电路与系统学报,2002,7(1):18.
[2] 胡广书.数字信号处理——理论、算法与实现[M].北京:清华大学出版社,2002.
[3] 伍仲祥,孙名松.浅析嵌入式系统编程中的代码优化[J].计算机应用,2005,24(12):50.
[4] Sloss AN,Dominic Symes D,Wright C.ARM system developer's guide-designing and optimizing system software[M].[S.l.]:Elsevier,2004.
[5] 孙洪,余翔宇.数字信号处理——基于计算机的方法[M].2 版.北京:电子工业出版社,2005.
[6] 张晓林.数字信号处理:原理、算法与应用[M].3 版.北京:电子工业出版社,2004.
[7] 杜春雷.ARM 体系结构与编程[M].北京:清华大学出版社,2004.