Accelerating Software Correlation of Spreading Signal Tracking Using SSE

ZHANG Wen, LIANG Hui-min
The Academy Of Equipment Command & Technology, Beijing, China, 101416
hpc.visu@gmail.com

Abstract: The spread spectrum receiver tracks spreading signals by the correlation of pseudo-random codes. In the spread spectrum signal tracking, the local reference signals (such as: local carrier, local PN codes) are replicated firstly, and then are correlated with the received sequence of digital IF, involving a large number of multiply-add operations. In spread spectrum receiver based software-defined radio, the software correlations could spend 80 percent of the overall CPU time, moreover require to complete by sampling rate (MHz order), and so face challenges for a huge amount of computations. In this paper, instruction-level parallelism of software correlation is exploited using Intel processor's Streaming SIMD Extension (SSE), and the local reference signals are generated quickly with look-up tables. Multi-user receiver could give full play to the Cache performance of modern processors. Test results show that, the speed-up ratio with SSE is about 10, tracking performance for multi-user receiver up to 140MCOPS.

Keywords: spread spectrum signal tracking; software correlation; SSE; multi-user receiver

利用 SSE 加速扩频信号跟踪的软件相关

张　文，梁慧敏
装备指挥技术学院，北京，中国，101416
hpc.visu@gmail.com

【摘要】扩频接收机利用伪随机码的相关性跟踪扩频信号。跟踪过程首先复现本地参考信号（载波、PN 码），然后与接收的数字中频序列进行相关计算，涉及大量的乘加运算。基于软件无线电的扩频信号接收，80%以上的 CPU 时间用于相关，而且相关要求以采样速度（MHz 量级）完成，面临庞大计算量的挑战。本文利用 Intel 处理器的流 SIMD 扩展（SSE）实现指令级并行的软件相关，本地参考信号的复现采用查找表方式快速生成，多用户同时接收充分发挥现代 CPU 的 Cache 性能。测试结果表明，SSE 对相关计算的加速比可达 10 左右，多用户接收的跟踪性能可达 140MCOPS。

【关键词】扩频信号跟踪；软件相关；SSE；多用户接收

1 引言

扩频技术可以实现恶劣环境下通信，是一种鲁棒性很强的通信机制，应用在降低能量密度、高精度测距、多址接入等领域[1]。最常用的是直接序列扩频，广泛应用于卫星导航、卫星通信等军事领域[2]，下面针对导航信号软件接收，讨论如何加速扩频信号跟踪的相关计算。

接收机利用 PN 码的相关性与伪随机性跟踪扩频信号，首先复现本地参考信号（载波、PN 码），然后与接收的数字中频序列进行相关计算，涉及大量的乘加运算。硬件接收机中相关一般是在专用 ASIC 芯片上实现，采用多通道并行，执行效率高。对基于软件无线电的导航接收机，软件相关是其实现瓶颈，也是其核心[3]。测试表明，软件接收中 80%以上的 CPU 时间用于相关，而且软件相关要求以采样速度（MHz 量级）完成，面临庞大计算量的挑战。现有的软件接收一般是在低采样率、低量化位情况下，这会损失计算精度[4,5]。

2 扩频信号跟踪

扩频信号跟踪[2]首先用捕获得到的载波频率和码相位初始化载波环和码环，通过反馈环路不断调整本地码和本地载波，进一步减小本地码、本地载波和输入信号的误差，使本地码、本地载波与输入信号的码、载波近似同频同相。复现的本地参考信号与接收的数字中频序列进行相关，产生跟踪环的控制输入信号。跟踪过程的原理如图 1，ID 表示累加与清零模块。

载波环通常采用锁频环（FLL）牵引锁相环（PLL）

978-1-935068-01-3

的结构。FLL 捕捉较大的频率误差，而 PLL 跟踪精度高。捕获得到的载波频偏仍有较大的误差，载波环先工作在 FLL，然后过渡到 PLL。

码环通常采用延迟锁相环（DLL），接收机产生早、即时、迟三路本地码并与输入信号作相关，码环利用早、迟两路相关结果之差得到输入信号和本地码之间的相位误差，从而调整本地码发生器。

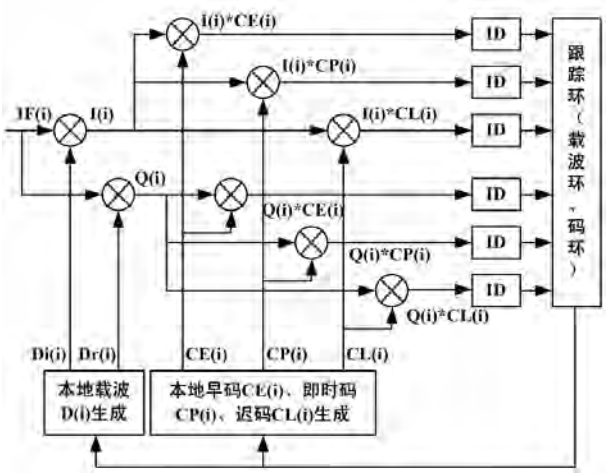

图 1 跟踪过程的原理图

3 流 SIMD 扩展

流 SIMD 扩展 (Streaming SIMD Extensions，SSE) 是 Intel 在 AMD 的 3D Now!发布一年之后，在其计算机芯片 Pentium III 中引入的指令集，是 MMX 的超集[6]。随后逐步推出了 SSE2、SSE3、SSE4。

SSE4[7]是 Intel 自从 SSE2 之后对 ISA 扩展指令集最大的一次的升级扩展，SSE4 增加的 47 条指令改进了整数和浮点操作，支持 DWORD 和 QWORD 操作，新的单精度 FP 操作、快速寄存器操作、面向性能优化的内存操作等。SSE 系列指令（下面统称 SSE）可应用在图形/图像处理、视频处理、2D/3D 创作、多媒体、游戏、内存敏感负载、高性能计算等应用中。

SSE 支持一种紧缩数据类型（Packed Data Type，记为 *PDT* ），*PDT* 数据是将八个单字整数或 4 个单精度浮点打包而成的 128bits 的数据。

SSE 支持一条指令对 *PDT* 中的数执行相同的算术、比较、逻辑等运算，实现了数据的指令级 SIMD 并行，如图 2，$A_{0\sim7}$、$B_{0\sim7}$分别为单字整数，运算符 *op* 可为+、-、*、/、逻辑、比较等。SSE 还提供访问 *PDT* 数据、打包 *PDT* 数据等的指令。

利用 SSE 开发实际应用时，主要考虑：SIMD 并行化、缓冲控制和特殊 SSE 指令[6~8]。SIMD 并行化为了开发实际应用中的 SIMD 并行性，并行处理输入数据流。缓冲控制指令可以控制数据 I/O 到指定层次的 Cache，提高 Cache 效率。缓冲控制指令还提供数据预取指令，隐藏 I/O 时延。

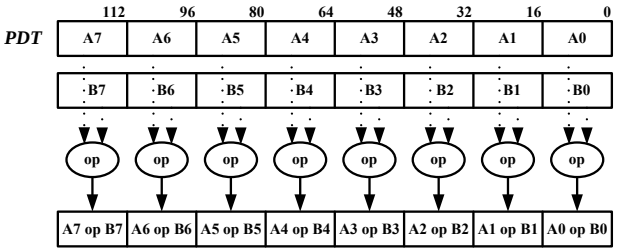

图 2 SSE 的 SIMD 运算示意

4 基于 SSE 的算法设计

4.1 软件相关的 SIMD 并行化

图 1 中，跟踪过程以采样时钟 *fs* 工作，假定在第 *i* 个时钟，输入中频信号 *IF[i]*，对应的本地载波 *D(i)*（实部 *Dr[i]*、虚部 *Di[i]*）、即时码 *CP[i]*、早码 *CE[i]*、迟码 *CL[i]*等参考信号。相关计算完成输入中频信号与本地参考信号的相关运算，即 *IF[i]* 先与 *Dr[i]*、*Di[i]* 相乘，得到两路基带信号 *I(i)*、*Q(i)*，然后分别与 *CP[i]*、*CE[i]*、*CL[i]*相乘，最后经过 ID 模块完成累加与清零。累加周期 $T>(N/fs)$，N 为累加的总的采样点。

软件接收可以对接收的数字中频进行缓冲，形成一段采样序列（一帧），然后与对应的本地参考信号序列进行乘加运算。序列乘加运算就是矢量点积，可利用 SSE 指令加速。假设我们缓冲的一帧（N 个采样点）输入序列为 $\overrightarrow{IF}=(IF[0],IF[1],...,IF[N-1])^T$，相应地产生一帧本地参考信号序列：

$$\begin{aligned}
&\overrightarrow{Dr}=(Dr[0],Dr[1],...,Dr[N-1])^T\\
&\overrightarrow{Di}=(Di[0],Di[1],...,Di[N-1])^T\\
&\overrightarrow{CP}=(CP[0],CP[1],...,CP[N-1])^T\\
&\overrightarrow{CE}=(CE[0],CE[1],...,CE[N-1])^T\\
&\overrightarrow{CL}=(CL[0],CL[1],...,CL[N-1])^T
\end{aligned}\qquad(1)$$

于是有：

$$\begin{aligned}
&\vec{I}=(I[0],I[1],...,I[N-1])^T=\overrightarrow{IF}\times\overrightarrow{Di}\\
&\vec{Q}=(Q[0],Q[1],...,Q[N-1])^T=\overrightarrow{IF}\times\overrightarrow{Dr}
\end{aligned}\qquad(2)$$

因此有：

$$\begin{aligned}
&\sum_{i=0}^{N-1} IF[i]*Di[i]*CP[i] = \vec{I}\bullet\overrightarrow{CP}\\
&\sum_{i=0}^{N-1} IF[i]*Dr[i]*CP[i] = \vec{Q}\bullet\overrightarrow{CP}\\
&\sum_{i=0}^{N-1} IF[i]*Di[i]*CE[i] = \vec{I}\bullet\overrightarrow{CE}\\
&\sum_{i=0}^{N-1} IF[i]*Dr[i]*CE[i] = \vec{Q}\bullet\overrightarrow{CE}\\
&\sum_{i=0}^{N-1} IF[i]*Di[i]*CL[i] = \vec{I}\bullet\overrightarrow{CL}\\
&\sum_{i=0}^{N-1} IF[i]*Dr[i]*CL[i] == \vec{Q}\bullet\overrightarrow{CL}
\end{aligned} \qquad (3)$$

式(3)中$\bullet$表示矢量的点积。式(2)、(3)可利用 SSE 指令实现，假定所有的序列已按 *PDT* 数据类型组织，16-字节的边界对齐。指令 *movapd* 可以实现快速访问 16-字节边界对齐的 *PDT* 。对式(2)，指令 *pmullw xmm1, xmm2* 完成 8 个单字整型构成的紧缩字类型的乘法，结果存在 *xmm1* 中。

对式(3)，指令 *pmaddwd xmm1, xmm2* 完成 8 个单字整型构成的紧缩字类型的乘法，然后将相邻的双字整型结果相加，存放在 *xmm1* 中， 形成紧缩双字类型（4 个 32bit 整型形成）。多次循环完成计算，得到 4 部分结果（每部分为 32bit 整型），构成紧缩双字类型，累加起来就得到式(3)中点积。

4.2 数字中频数据的组织

SSE 能够快速处理 *PDT* 数据类型，必须将输入信号、本地参考信号以 *PDT* 类型组织。中频数据采样一般低于 8bit 量化，而本地载波低于 8bit，本地码参考信号一般只有+1、-1 两个值（为了克服干扰，可以多 bit 量化），因此所有数据可以用 16bit 的单字整数表示。对于一般应用，利用 SSE 同时处理八个输入数据，可以达到很高效率，但 SIMD 并行性要求对每个输入数据做同样操作。在运算过程中，中间变量都采用 *PDT* 类型。SSE 对内存中 16-字节边界对齐的 *PDT* 数据访问速度快，计算中的 *PDT* 数据采用 16-字节的边界对齐。

输入采样序列以数据包从 AD 采样板或数据文件中读入，充分利用 DMA 特性，包大小 100KB 左右。数据包分成若干数据帧，帧是相关计算处理的最小单位，其时间周期不大于累加周期 *T*。为了利用 Cache 特性，帧可以分为若干数据块。

4.3 本地参考信号的复现

如果每次根据本地载波的当前频率与初相位调用 cos()与 sin()函数计算本地载波序列，由于 cos()和 sin()函数运算时间较长，这样本地载波的产生也会成为跟踪的瓶颈之一。

假设本地载波频偏的范围为 $-5KHz \sim 5KHz$，以 *200Hz* 为频率间隔将频偏分为 51 个离散频点：*-5000Hz*、*-4800Hz*、……、*4800Hz*、*5000Hz*，加上标准中频频率得到 51 带频偏的频点。首先在这些离散的频点上以采样速率产生初相位为 0 的载波序列，包括实部、虚部，共 51 组。序列长度必须大于跟踪过程中可能出现的最大相关运算长度。

假设跟踪过程中某次相关运算对应的本地载波频率为 *fc*、初始相位为 φ，首先从 51 个频点中找到与 *fc* 最接近的那个频点，该频点对应的那组载波序列就为本地载波多普勒序列，此时初相位为 0。然后与输入信号作相关，对相关结果用 φ 作修正。用这种方法，本地载波的产生只需要做一次查表操作，节省了产生本地载波的时间，可以提高跟踪速度。本地早 PN 码、即时 PN 码、迟 PN 码等也可以用类似方法产生。

5 算法实现

Intel® SSE C/C++开发工具支持嵌入汇编语言、内在指令函数、向量类库、自动向量化等四种方式来开发 SSE 实际应用[8]。在 C/C++中嵌入 SSE 汇编指令，编译器能够识别新的 SSE 汇编指令和寄存器，产生相应代码。内在指令函数与相应的 SSE 指令一一对应，以 C/C++函数形式来使用 SSE 指令。向量类库支持八个单字整型构成的 *PDT* 数据类型，用 SSE 方式处理 *PDT* 类型。自动向量化是在编译时，编译器自动将程序循环中的操作向量化为 SSE 指令。

嵌入汇编语言和内在指令函数两种方式都可采用人为优化的寄存器分配、指令调度等，二者性能差不多，在四种方式中最优，但编码较复杂。向量类库方式是编译器自动实现寄存器分配、指令调度等，性能比前两种差一些，其编码较容易。自动向量化完全由编译器自动实现 SIMD 并行化，性能最差。采用嵌入汇编语言、自动向量化两种方式来分析相关计算，实现过程中，依据 Intel®处理器的体系结构，采用寄存器分配、指令调度、循环展开等优化技术。另外，为了进行比较，还利用 C++实现了传统的非 SSE 算法。

6 结果分析

下面利用 Intel 公司的 T7500 2.2GHz 双核处理器、一级 Cache 为 32I+32D、二级 Cache 为 4MB、1G 内存、Windows XP 平台上，对软件相关的非 SSE、自动向量化、SSE 嵌入汇编等三种实现进行分析。测试结果如图 3，经过分析，100 个采样点序列的相关计算，

978-1-935068-01-3 © 2009 SciRes.

非 SSE、自动向量化、SSE 嵌入汇编三种实现分别需要大约 3350 个时钟、1600 个时钟、330 个时钟，SSE 嵌入汇编的并行加速比为 10.0 左右。SSE 嵌入汇编相关计算的理想性能为 733MCOPS，MCOPS 表示每秒 10^6 个采样点的相关计算。

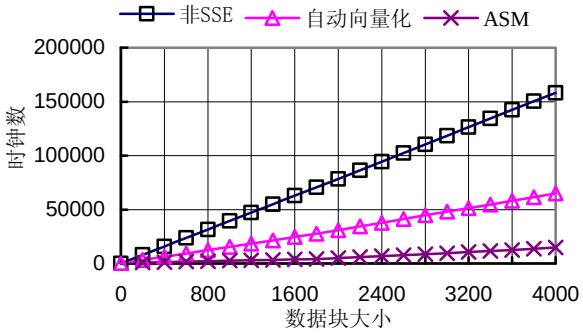

图 3　相关计算的性能分析

软件相关中还包括本地参考信号的生成。本地信号生成虽然可以查表产生，但不能利用 SSE 加速，因此，相关计算受限于参考信号的生成。加上跟踪模块的其它开销，最后跟踪效果会大大下降。

在进行 Cache 优化时，在计算式(2)、(3)中，为了使数据能够在一级 Cache 中充分使用，需要将一帧分成若干小数据块，这些数据块就可以在一级 Cache 多次使用，增加命中率，如图 4。

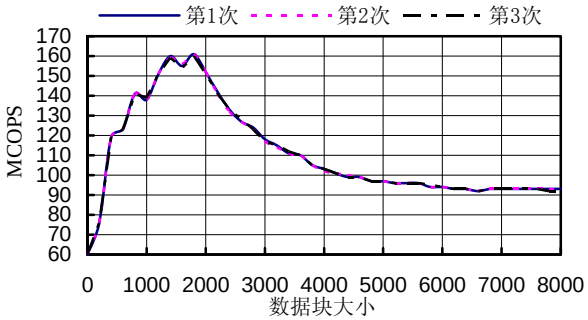

图 4 Cache 对跟踪性能的影响

图 4 显示了数据分块大小带来的跟踪性能的影响。数据分块在 1000~2000 个采样点时，跟踪效果最好，达到 150MCOPS。而数据分块大于 5000 个采样点时，一级 Cache 不会影响性能，跟踪性能大概为 90MCOPS 左右。

一般来讲，CDMA 依据不同的码区分不同用户，可以多用户接收，形成多通道同时跟踪多路信号。输入同一帧采样信号可以被多通道的相关计算同时使用，应该考虑二级 Cache 大小。输入帧不应太大，太大使得输入帧不能呆在二级 Cache 中，缺失率增加，降低 Cache 性能，如图 5。

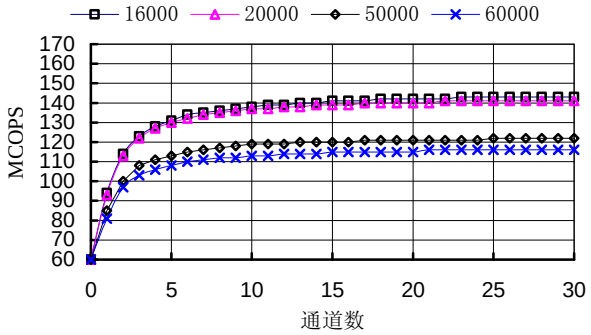

图 5 多用户接收的性能分析

当输入帧大小为 16000、20000 个采样点时，10 个以上的通道同时跟踪，性能约为 140MCOPS 之间。而当帧大小增加到 50000 以上时，帧数据太大，会发生二级 Cache 缺失，降低性能，跟踪性能约为 110MCOPS。图 5 可以看出，多通道跟踪同时使用同一个输入帧序列，增加了 Cache 的时间、空间局部性，所以性能有所提高。

7 小结

利用 SSE 加速软件相关，本地参考信号采用查表方式快速生成，多用户接收进一步改进 Cache 性能。测试表明，SSE 对相关的加速比可达 10 左右，多用户接收的跟踪性能可达 140MCOPS。

References (参考文献)

[1] Bernard Sklar. Digital Communications: Fundamentals and Applications (2nd Edition) [M]. Prentice Hall PTR, 2001.
[2] Elliott Kaplan, Christopher Hegarty. Understanding GPS: Principles and Applications, Second Edition [M]. Artech House Publishers, 2005.
[3] Mitola, J. The software Radio Architecture [J]. IEEE Communications Magazine, IEEE, 1995(5): 26–38.
[4] B. M. Ledvina, etc. A 12-Channel Real-time GPS L1 Software Receiver[C]. Proc. of the Institute of Navigation National Technical Meeting, Anaheim, CA, January 22–24, 2003:767–782.
[5] B.M. Ledvina, etc. Bit-Wise Parallel Algorithms for Efficient Software Correlation Applied to a GPS Software Receiver [J]. IEEE Transactions on Wireless Communications, 2004(5): 1469–1473.
[6] Intel® Architecture Software Developer's Manual, Volume 1: Basic Architecture. Intel Corporation. available at http://developer.intel.com/design/PentiumIII/manuals/.
[7] Introducing the 45nm Next-Generation Intel Core Microarchitecture. Intel Corporation. available at http://www.intel.com/technology/architecture-silicon/intel64/.
[8] Intel® SSE4 Programming Reference. Intel Corporation. available at http://software.intel.com/file/18187/.