第 31 卷　第 11 期　　计　算　机　工　程　　2005 年 6 月
Vol.31　№ 11　　Computer Engineering　　June 2005
·基金项目论文·　文章编号：1000—3428(2005)11—0052—04　文献标识码：A　中图分类号：TP 303

一种高速定点 FFT 处理器的设计与实现

付　博[1,2]，李　栋[1,2]，谢应科[2]
（1.中科院研究生院，北京 100080；2.中科院计算所）

摘　要：提出了一种高速定点 FFT 处理器的设计方法，此方法在 CORDIC 算法的基础上，通过优化操作数地址映射方法和旋转因子生成方法，每周期完成一个基 4 蝶形运算，具有最大的并行性。同时按照本文提出的因子生成方法，每个周期可生成 3 个旋转因子，且硬件实现简单，无须额外的 ROM 资源。整个系统采用 Xilinx 公司的 XCV2P30 仿真，系统频率达到了 130MHz，对于 1k 点 16 位的复数 FFT 需要 9.8μ s，16k 点需要 221μ s，优于目前绝大多数已有的 FFT 处理器。

关键词：快速傅立叶变换；FFT 处理器；CORDIC 算法

Design and Implementation of High Speed and Fixed-point FFT Processor

FU Bo[1,2], LI Dong[1,2], XIE Yingke[2]
(1. Graduate School, CAS, Beijing 100080; 2. Computing Technology Institute, CAS)

【**Abstract**】The paper presents the design method of a high speed and fixed-point FFT processor. By optimizing the memory mapping algorithm and the generation of twiddle factors, the method based on CORDIC algorithm can calculate a radix-4 butterfly in one clock cycle and has the maximal degree of parallelism. This paper also introduces a method of twiddle factor generation, which can simultaneously generate three twiddles of one radix-4 butterfly. The implementation of the twiddle factor generation in hardware is simple and don't need extra ROM resources. The processor has been implemented on a Xilinx chip XCV2P30 and obtains the operating clock frequency at 130MHz. The processor can compute a complex 1024-point FFT with 9.8μs and 16384-point FFT with 221μs. The performance of the FFT processor is better than most available FFT processors.

【**Key words**】FFT ; FFT Processor; CORDIC algorithm

离散傅立叶变换（DFT）是描述离散信号时域和频域关系的重要数学工具，它在无线通信、图像处理和视频系统等领域得到了广泛的应用。常用的 FFT 模块完成 1 024 点，16 位字长的定点、块浮点、浮点运算需要的时间为几十或者上百微秒[1]，不能满足日益增加的实时要求，如何设计出高效的 FFT 处理器成为这些系统的关键。

提高 FFT 处理器性能的常用方法是增加运算的流水级和提高功能部件的并行度[2, 3]，随着运算部件的高度并行，访问数据的速度成为系统的瓶颈，提高访问数据的并行性成为一个关键问题。

1 算法及相关

1.1 FFT 算法

N 个样本点的 DFT 定义为

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

其中 $W_N=e^{-j\cdot\frac{2p}{N}}$，$N=2^n$，$k=0,1,\cdots,N-1$。从式(1)可知，直接计算 N 个样本点的 DFT 需要 $O(N^2)$ 次复数乘法和复数加法运算，运算量很大，快速算法显得非常重要。

常用 FFT 算法分为基 2、基 4 和分裂基 3 种：基 2 算法运算结构简单，蝶形运算数较多，乘法运算量大；基 4 算法的蝶形运算比较复杂，数据访问次数较少，计算量也少；分裂基算法结合了基 2 和基 4 的特点，但其 L 型蝶形运算单元在控制上实现复杂。本文从算法复杂性和硬件实现的难易程度考虑，采用基 4 频域抽取算法。算法概述如下：当 $N=4^r$ 时，对式(1)可按如下方法作频率抽取：

$$X(k)=\sum_{n=0}^{N/4-1}x(n)W_N^{nk}+\sum_{n=N/4}^{N/2-1}x(n)W_N^{nk}+\sum_{n=N/2}^{3N/4-1}x(n)W_N^{nk}+\sum_{n=3N/4}^{N-1}x(n)W_N^{nk}$$

令

$k=4m, k=4m+2, k=4m+1$ 及 $k=4m+3$，　$m=0,1,\cdots,N/4-1$

有：

$$X(4m)=\sum_{n=0}^{N/4-1}\begin{bmatrix}(x(n)+x(n+N/2))+(x(n+N/4)+\\ x(n+3N/4))\end{bmatrix}W_{N/4}^{nm}$$

$$X(4m+2)=\sum_{n=0}^{N/4-1}\begin{bmatrix}(x(n)+x(n+N/2))-(x(n+N/4)+\\ x(n+3N/4))\end{bmatrix}W_N^{2n}W_{N/4}^{nm}$$

基金项目：国家自然科学基金资助项目（60303017）；计算所青年基金资助项目（20026180-13）；科技部国际合作重点项目(2001AA111090)

作者简介：付　博(1978—)，男，硕士生，研究方向：计算机体系结构,VLSI 设计等；李　栋，硕士生；谢应科，博士、副研究员

定稿日期：2004-04-31　　**E-mail：**ykxie@ict.ac.cn

$$X(4m+1)=\sum_{n=0}^{N/4-1}\left[\begin{array}{l}(x(n)-x(n+N/2))-j(x(n+N/4)-\\ x(n+3N/4))\end{array}\right]W_N^{n}W_{N/4}^{nm}$$

$$X(4m+3)=\sum_{n=0}^{N/4-1}\left[\begin{array}{l}(x(n)+x(n+N/2))+(x(n+N/4)+\\ x(n+3N/4))\end{array}\right]W_N^{3n}W_{N/4}^{nm} \tag{2}$$

式(2)是 FFT 基 4 频域抽取算法的基本运算单元，一般称为蝶形运算。下一步再将 $X(4m+i)$，$i=0,1,2,3$ 分解成 4 个 $N/4^2$ 序列，迭代 r 次后完成计算，整个算法的复杂度减少为 $O(N\log_4 N)$。

1.2 CORDIC 算法

从式(2)知，每个周期完成一个基 4 蝶形运算，需要 9 个乘法部件，硬件开销和设计复杂度比较大。本文采用 CORDIC 算法来解决这个问题。 CORDIC 算法[4]可用来快速求解三角、双曲、对数等一系列函数。其基本思想是把矢量的旋转分解成为一系列小角度的旋转，当这些小角度的 tan 值为 2^{-i} 时(i=0，1，2，…)，整个旋转过程就可用移位和加法完成。FFT 算法中的复数乘法运算用 CORDIC 算法表示如下：

$$\begin{aligned}&X_{i+1}=X_i-Y_i*d_i*2^{-i}\\&Y_{i+1}=Y_i+X_i*d_i*2^{-i}\\&Z_{i+1}=Z_i-d_i*\tan^{-1}(2^{-i})\\&d_i=-1,\text{如果}Z_i<0,\text{否则}d_i=+1\end{aligned} \tag{3}$$

X_i，Y_i 为第 i 次旋转的输入值，根据角度 Z_i 的符号分别加上或减去 Y_i，X_i 右移 i 位的值，就可得到第 i 级的输出结果 X_{i+1}，Y_{i+1}。

2 FFT 处理器设计和实现

2.1 总体结构

提高并行度是处理器设计的关键问题，FFT 处理器的并行结构可分为：(1)运算部件并行：如在基 4 算法中采用 3 次复数乘法器和八次复数加法器；(2) 蝶形单元并行：采用多个蝶形运算单元，如双基 2 结构；(3) 数据存储的并行：采用多个存储体存放数据，以提高数据的并行度。

本文研究的目标就是提高芯片数据的并行度，图 1 是系统的整体结构框图。操作数据按照一定规则存储在多个双端口存储器中；地址生成器每个周期完成 4 个输入、4 个输出数据的并行读写；蝶形运算部件每个周期完成一个基 4 运算（包括加法和相应 CORDIC 运算）；旋转因子生成器提供 CORDIC 所需要的 3 个因子；输入输出转换完成外部数据接口和内部存储之间的数据转换。各部分的设计将在以后章节详细介绍。

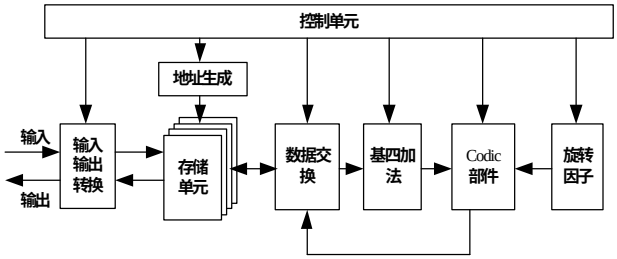

图 1 FFT 处理器结构框图

2.2 地址映射

数据存取并行性是 FFT 芯片设计的重要问题。本文在文献[2，5]的基础上将讨论一种同址、并行的地址映射方法，并给出其基 4 频率抽取算法的实现。对于基 4 运算的任何一个操作数地址 A，记

$$A=\sum_{i=0}^{r-1}4^i\cdot A(i),\quad A(i)=0,1,2,3;\ r=\log_4 N \tag{4}$$

把 A 写作

$$A=4\cdot\sum_{i=1}^{r-1}4^{i-1}\cdot A(i)+A(0) \tag{5}$$

构造

$$B=4\cdot\sum_{i=1}^{r-1}4^{i-1}\cdot A(i)+\left(\sum_{i=0}^{r-1}A(i)\right)\bmod 4 \tag{6}$$

显然，对于任何一个地址 $A(0\le A<N)$，A 和 B 是一一对应的。令

$$m=\sum_{i=1}^{r-1}4^{i-1}\cdot A(i)，\quad b=\left(\sum_{i=0}^{r-1}A(i)\right)\bmod 4$$

则 $B=4m+b$ (7)

可以把地址 A 映射到 4 个存储体上，由 b 对应体号，m 对应体内地址，对于基 4 运算所需要的 4 个操作数，可以很容易地证明它们都映射在不同的存储体上[5]。表 1 是 32 个地址的映射关系，其中，灰色部分每一行代表一个存储体。

表 1 32 点地址的映射关系

	0	1	2	3	4	5	6	7
0	0	7	10	13	19	22	25	28
1	1	4	11	14	16	23	26	29
2	2	5	8	15	17	20	27	30
3	3	6	9	12	18	21	24	31

2.3 地址生成器

结合 2.2 节的地址映射和基 4 频域抽取算法，本文提出一种简单、易于扩展的地址生成器实现方法。考虑第 p（$p=0,1,\cdots,r-2$）级的情况，运算的 4 个数仅第 $r-p-1$ 位不同，分别为 0,1,2,3。地址映射的 m 有如下的关系：

$$\begin{aligned}&m_0=\sum_{i=r-p}^{r-1}4^{i-1}\cdot A(i)+4^{r-p-2}\cdot 0+\sum_{i=1}^{r-p-2}4^{i-1}\cdot A(i)\\&m_1=m_0+4^{r-p-2}\\&m_2=m_0+2*4^{r-p-2}\\&m_3=m_0+3*4^{r-p-2}\end{aligned} \tag{8}$$

为了得到上式的 4 个值，使用一个计数器表示 $\sum_{i=r-p-1}^{r-2}4^{i-1}\cdot A(i+1)+\sum_{i=1}^{r-p-2}4^{i-1}\cdot A(i)$

然后在第 $r-p-1$ 位插入 $A(r-p-1)$ 得到 m_0。实现时使用 3 个寄存器（图 2），分别是计数器 C，移位控制寄存器 M 和结果寄存器 R。C 的初始值为 0，每进行 1 次蝶形运算，C 加 1；M 初始值为 0，每完成一级运算，就右移 1 格(2 位)填入 11。结果寄存器 R 的值就是 m_0。

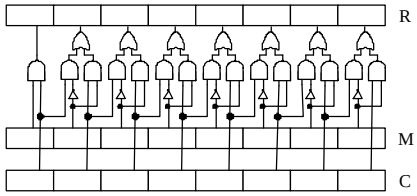

图 2 操作数地址产生方法

要得到蝶形运算的其它 3 个操作数地址，只需对第 P 位进行相应的赋值就可以了。当 p=r-1 时，4 个操作数的地址相同，且都等于计数器 C 的值。

b 的取模运算可以由半加器完成，对于基 4 FFT，$A(i)$ 是

2 位，用两位半加器就可以得到 b 值。

2.4 因子生成器

因子生成常用的方法是将角度或角度的正弦、余弦值存放在只读存储器（ROM）中，然后设计一个存取电路选择不同的因子送往蝶形运算单元。这种方法随着并行度的增加，需要的 ROM 资源随之增加，存取电路也变得复杂。根据 CORDIC 算法的特点，本文提出了一种新的方法，每周期可生成基 4 运算所需的 3 个角度值，且硬件实现简单。具体方法如下：一个基 4 蝶形运算需要 3 个旋转因子$W_N^n, W_N^{2n}, W_N^{3n}$，其中$W_N^n = e^{-j\cdot\frac{2pn}{N}}$，$N=4^r$。考虑第 p（$p=0,1,\cdots,r-1$）级的情况,令$w = 4^p\cdot\frac{2p}{N}\cdot n,\quad n=0,1,\cdots,4^{r-1-p}-1$。用 2 的幂次位表示$2p$，对于 CODIC 算法，只要设计电路生成$4^p$和 n，就可以计算出$w$，从而得到频域抽取基 4 运算所需要的 3 个因子$W_N^n, W_N^{2n}, W_N^{3n}$。图 3 是生成 n 的结构原理图。

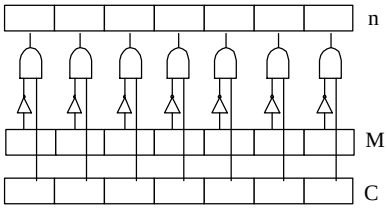

图 3 旋转因子中 n 的产生方法

其中 C 为计数器从 0 到 N/4-1，M 为移位控制寄存器。计数器 C 和移位控制寄存器 M 初值为 0，C 每周期加 1，n 等于 M 的值取反和 C 的值相与。当计数器 C 的值等于 N/4-1，移位控制寄存器 M 的值右移 1 格(2 位)，填入 11。4^p的生成，只需设计一个计数器 p，初始值为 1，当 C 值为 N/4-1 时，计数器 p 左移 2 位即可。有了4^p和 n 值以后，根据点数和表示$2p$的位数，通过简单的移位和加法运算就可以得到$W_N^n, W_N^{2n}, W_N^{3n}$。

2.5 CORDIC 运算单元

CORDIC 运算单元由数据预旋转、角度查找表和移位计算 3 部分组成。数据预旋转通过将输入数据进行简单的取反或交换，完成旋转因子从$[0,2p)$到$[0,\frac{p}{2})$的映射，映射关系如表 2：q 为旋转因子，I、Q 为映射前数据的实部和虚部，I'、Q'为映射后数据的实部和虚部。经预旋转的数据和归一化的角度被送往移位计算部件。这里用 20 位数据宽度表示$2p$，$tg^{-1}2^0, tg^{-1}2^{-1},\cdots tg^{-1}2^{-18}$的值分别为 0x20000，0x12E40,…，0x00001，以后都为 0。

整个移位计算过程采用 20 级流水，可保证角度的截断误差小于$\frac{2p}{2^{20}}$。考虑第 i 级情况(图 4)，X_i，Y_i，Z_i为第 i 级的输入。

表 2 旋转因子映射关系

	$0\le\theta<\frac{\pi}{2}$	$\frac{\pi}{2}\le\theta<\pi$	$\pi\le\theta<\frac{3\pi}{2}$	$\frac{3\pi}{2}\le\theta<2\pi$
I'	I	$-Q$	$-I$	Q
Q'	Q	I	$-Q$	$-I$

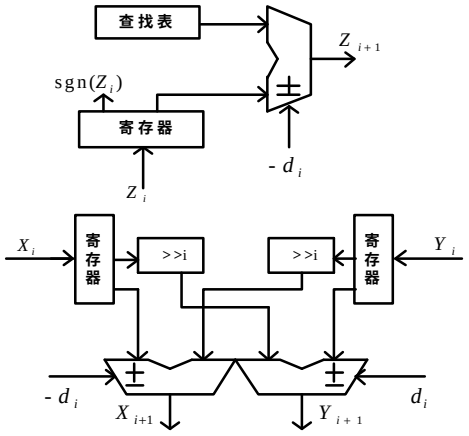

图 4 CORDIC 实现结构

2.6 输入输出转换单元

实际应用中，芯片和外部数据交换的带宽有限。为了解决这个问题，本文设计了一个转换电路，可在芯片内部和外围电路之间进行多路数据转换。实现方法如下，记录输入、输出数据的个数 C，根据式(7)得到 C 的 m 和 b 的值，m 是片内存储器的体内地址，b 值和数据宽度结合可得到不同的映射方式。图 5 是数据宽度为 2 路和 4 路时，输入、输出到片内存储器的转换单元。

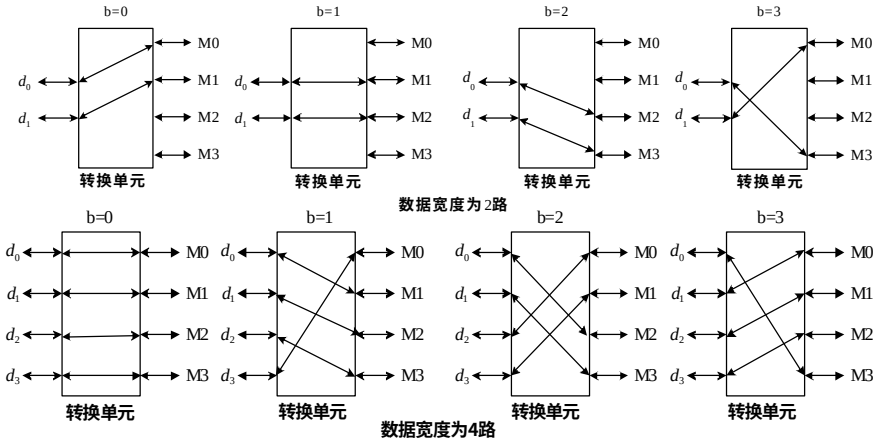

图 5 输入输出转换单元

3 系统仿真和比较

表 3 是部分仿真结果和比较。

表 3 处理器性能比较[1,6,7]

处理器	点数	数据宽度(位)	执行时间(ì s)	时钟频率(MHz)	处理器数
本文设计	1 024	16	9.8	130	1
本文设计	16 384	16	221	130	1
XilinxFFT/IFFTcore	1 024	16	41.5	100	1
Altera FFT IP	4 096	16	262	94	1
Plessey,16510	1 024	16	98	40	1
Dassault Electronique	1 024	12	6.4	40	6
TexMemSys,TM-66	1 024	32	65	50	2+
DoubleBW	1 024	24	10	128	1

4 结论

本文在 CORDIC 算法的基础上，针对基 4 频域抽取算法，提出了一种并行的地址和因子生成方法，提高了芯片的数据吞吐能力。处理器的核心运算部件每周期能完成一个基 4 蝶形运算，既可以是一个专用的 FFT 处理器，也可以用作通用处理器的协处理器。同时本文设计的输入、输出转换单元根据需要可配置成不同的数据宽度，增加了芯片的灵活性。

参考文献

1 http://www-star.stanford.edu/~bbaas/fftinfo.html

2 Johnson L G. Conflict Free Memory Addressing for Dedicated FFT Hardware. IEEE Trans. Circuits and Sys. II, 1992,39: 312-316

3 Cheng-han Sung, Lee Kunbin, Jen Cheinwei. Design and Implementation of A Scalable Fast Fourier Transform Core. ASIA-PACIFIC CONFERENCE ON ASICs, 2002: 295-298

4 Andrake R J. A Survey of CORDIC Algorithms for FPGA Based Computers. FPGA '98. Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays, Monterey, CA, 1998-02-22:191-200

5 谢应科，付 博.数据全并行 FFT 处理器的设计.计算机研究与发展. 录用

6 Xilinx. http://www.xilinx.com/ipcenter/catalog/logicore/docs/vfft1024v2.pdf

7 Altera.FFT MegaCore Function User Guide (Version 1.02). 2001-03

（上接第 29 页）

(1) 尽量先用 ⊓ 和 ⊔ 规则，并及时检查 $A(x_0)$，$\neg A(x_0)$中是否存在冲突；

(2) 用 ∃ 和 ≥ 规则产生所有 x_0 的后继；

(3) 用 ≤ 规则产生后继所有必要的标识，并检查冲突；

(4) 依次重复处理其它后继。

当 Inclusion()判定结果成立后,将 M 中相应的位置置 1，表示该行对应的概念包含于该列对应的概念。

表 1 规则

规则种类	条 件	动 作
⊓	Ã 包含$(C_1 \sqcap C_2)(x)$但不同时包含 $C_1(x)$和 $C_2(x)$	在 Ã 中将$(C_1 \sqcap C_2)(x)$替换为 $C_1(x)$和 $C_2(x)$
⊔	Ã 包含$(C_1 \sqcup C_2)(x)$但不包含 $C_1(x)$和 $C_2(x)$	在 Ã 中将$(C_1 \cup C_2)(x)$去除，Ã 分解为 $\tilde{A}'=\tilde{A}\cup\{C_1(x)\}$，$\tilde{A}''=\tilde{A}\cup\{C_2(x)\}$
∃	Ã 包含$(\exists R.C)(x)$但不包含个体 y 使 C(y)和 R(x,y)在 Ã 中存在	在 Ã 中去$(\exists R.C)(x)$，$\tilde{A}=\{\tilde{A}\cup C(y)\cup R(x,y)\}$
∀	Ã 包含$(\forall R.C)(x)$和 R(x,y)但不包含 C(y)	$\tilde{A}=\tilde{A}\cup\{C(y)\}$
≥	Ã 包含$(\geq nR)(x)$但不包含 $R(x,y_k)(1\leq k\leq n)$与 $y_k \neq y_j$ $(1\leq k<j\leq n)$	$\tilde{A}=\tilde{A}\cup\{R(x,y_k)\mid 1\leq k\leq n\}\cup\{y_k\neq y_j \mid 1\leq k<j\leq n\}$
≤	Ã 包含$(\leq nR)(x)$和 $R(x,y_1),\cdots,R(x,y_{n+1})$，但对某些 k,j 不包含 $y_k \neq y_j$	对每对 y_k 和 y_j 只要 k>j 与 $y_k \neq y_j$ 不在 A 中，通过将 A 中的 y_k 替换成 y_j，生成 $A_{k,j}=[y_k/y_j]A$

算法的第 3 步用来求取 LCS,即每个术语的直接上级术语，类似父子关系。在 TBOX 中，每个术语可能有许多上级术语，包括直接和间接的。首先对矩阵中的概念逐个分析，对每行对应的概念,通过分析行中的 1 来判断包含关系,并将对应的包含者取出放入字符串数组 C 中，它由 GetcolConceptName 完成。然后对 C 中每个概念察看对应的矩阵中的行，如果此概念有包含者在 C 中出现，则由 RemoveFromeStringArray(C,conceptname)从 C 中移去。现在,C 中剩下了 C_i 的包含者，通过 Output(Conceptname1 ⊐ Conceptname2)逐个输出 C_i 的包含关系。

第 4 步的 XMLGEN()是根据输出的概念之间的包含关系，产生用 XML 表示的、适合 CSCL 中知识导航模块的概念本体的框架表示，其 DTD 包含术语名和上下级术语名以及注释。上下级术语名用来提供导航功能。

2.3 算法复杂性

算法的第 1 步用到函数 clash(),它在规则应用顺序正确安排的前提下是 NPSPACE 完全的。算法第 2 步 inclusion 为 PSPACE。算法的后两步都是 PTIME 的。由于 NPSPACE 和 PSPACE 是等同的，因此分类算法的复杂性为 PSPACE 完全的。由于术语定义式中包含了其它术语，因此术语所对应的 ABOX 中指派的个体数量为 $2^{n+1}-1$。然而，由于对规则的引用顺序作了优先选择，因此算法只需存储一条可能的路径以及在该路径上的个体的直接后继，其复杂性为 PSPACE。如果算法需要保存每一条可能的路径，并对各种可能运算，则其复杂性为 EXPTIME。因此，规则引用的优先顺序非常重要。

3 结论

许多计算机支持的协同学习平台包括知识导航模块，但由于其本体库采用框架表示法使语义不清，不利于学生全面理解概念。为解决此问题，本文提出了用描述逻辑语言来建立本体库。通过对术语之间的包含关系进行分析，得出 ABOX 的一致性和 TBOX 的可满足性的对应，从而得出包含关系成立的充要条件是 ABOX 中相应的式子不满足一致性，为概念包含的判断函数 inclusion()的设计奠定了基础。为对 Tbox 中的概念分类，以生成适合知识导航模块使用的框架表示的本体库，本文进一步讨论了概念的分类算法。由于 inclusion 仅判断两概念之间的包含，为对所有概念分类，算法首先生成包含矩阵，引入最小概念包含 LCS 并通过对概念包含矩阵分析生成概念的 LCS；再由框架生成函数对其产生适合知识导航使用的 XML 文档。虽然算法的复杂度为 PSPACE 的，但在实际数据测试(TBOX 包含 200 个概念定义式) 中分类算法的运行时间为 200ms(450MHz Pentium Ⅲ)还是令人满意的。目前该系统正在进一步测试，如何提高大容量 TBOX 时分类算法的计算速度有待进一步研究。

参考文献

1 Diego C, Maurizio L. Description Logics for Conceptual Data Modeling. In Logics for Databases and Information Systems, Kluwer Academic Publisher, 1998:229-263

2 Bechhofer S Horrocks I, Goble C, et al. OilEd: A Reasonable Ontology Editor for the Semantic Web. To Appear in 24th German / 9th Austiran Conference on Artificial Intelligence (KI 2001). Formal, 2001

3 Donini F M, Massacci F. EXPTIME Tableaux for ALC. Artificial Intelligence, 2000, 124(1): 87-138