第34卷 第15期 计 算 机 工 程 2008年8月
Vol.34 No.15 Computer Engineering August 2008
·软件技术与数据库· 文章编号：1000—3428(2008)15—0047—04 文献标识码：A 中图分类号：TP311

# 一种有效的XML数据清洗方法

韩京宇[1,2]，成 瑜[2]，董逸生[2]
（1. 南京邮电大学计算机学院，南京 210003；2. 东南大学计算机科学与工程系，南京 210096）

**摘 要：**研究XML格式的重复数据元素的特点，提出对于特定应用领域，在具体的上下文环境中主动学习XML重复元素的识别规则。通过结构转换，将结构不尽相同的XML数据映射成结构一致的数据，并通过学习不同层次数据元素间的依赖关系权重来获得匹配规则。根据学习得到的转换和匹配规则，采用哈希过滤的方法来提高检测重复XML元素的效率。该方法能够有效地解决XML重复检测面临的结构多样性的问题，理论分析和实验表明，该方法有较高的精度和效率。

**关键词：**主动学习；匹配规则；哈希

# Efficient Cleaning Approach for XML Data

HAN Jing-yu[1,2], CHENG Yu[2], DONG Yi-sheng[2]
（1. School of Computer, Nanjing University of Posts & Telecomunications, Nanjing 210003;
2. Department of Computer Science and Engineering, Southeast University, Nanjing 210096）

**【Abstract】** By studying characteristics of duplicate XML data, this paper proposes an active machine learning method for a specific application, which is applied to glean transformation rules and matching rules, and accurately identify duplicate XML elements. Transfomation rules are used to eliminate the structural diversities among elements and matching rules are used to identify the relationships between parent and child nodes. In turn, during the detection phase an efficient hash filter algorithm is proposed to reduce computational complexity. Theory and experiment shows that the method can solve this problem efficiently and effectively.

**【Key words】** active learning; matching rules; hash

## 1 概述

XML数据作为具有代表性的半结构化数据，已成为网上数据传输和交换的标准。当多个不同数据源以XML格式来描述现实世界的实体时，格式和内容不同的元素可能代表现实世界中的同一实体。因此，在集成多个XML数据源时，如何有效地识别重复的XML元素成为有意义的课题。XML重复元素识别面临的主要问题是：(1)XML作为一种松散结构的数据，同一类型实体可能用不尽相同的结构来表达，即结构的多样性[1]，给元素相似性判断带来困难。(2)XML作为一种层次结构的数据，嵌套元素间的依赖关系要比传统的平面关系数据库数据复杂。本文提出对于特定应用领域，应该在具体的上下文环境中学习XML重复元素的识别规则，它包含结构转换规则和匹配规则。结构转换规则是将XML数据转换成结构统一的数据的规则。匹配规则是用来判断结构统一的2个XML元素是否重复的规则，它包含2个方面的内容：(1)确定不同类型的子元素节点在父元素节点中的权重；(2)确定不同类型元素重复的相似度阈值。根据学习得到的转换和匹配规则，最后给出了一种高效的重复XML元素识别算法。

## 2 转换规则和匹配规则的学习

本文提出的XML数据清洗主要分成3个步骤：预处理(pre-clean)，检测规则学习(detection rule learn)和清洗(result clean)。

预处理阶段主要是将各个不同的XML文档进行处理，抽取查询需要的根元素并进行如下的处理，以便于后面的重复识别规则的学习和最终重复元素识别。它主要包含以下几项工作：(1)执行拼写检查；(2)统一字符数据的大小写；(3)处理出现多余的符号和空白；(4)建立标签的同义词表，以便于后文的转换规则学习；(5)对元素内容中出现的高频虚词(stop words)，如the, of, as等进行剔除。

检测规则学习主要是对于特定的应用，根据有限的训练样本学习重复元素识别的转换规则和匹配规则，这是识别XML重复元素的关键。一方面，XML作为一种半结构化数据，即便是对同一应用，同一类型的实体也常采用不尽相同结构的元素来表达。在重复元素识别时，为了按照对应的结构进行2个元素相似度计算，必须学习2个元素结构的对应关系——这是转换规则学习的任务。另一方面，XML作为一种嵌套结构的数据，每个元素都是由若干个子元素构成的，准确识别出各个子元素对父元素的决定程度及重复的相似度阈值，对识别相似重复元素至关重要——这是匹配规则学习的任务。清洗阶段的作用是利用学习得到的转换规则和匹配规则对XML数据建立哈希表进行重复元素的识别。

### 2.1 转换规则

同一类型实体在不同文档中常采用不尽相同的XML结构，元素内容的表达也不一定相同。为了发现不同文档中元素间的重复关系，对于特定应用要在其上下文中学习相应的结构转换规则和内容转换规则。如何学习内容转换规则在文

**基金项目：**江苏省“十五”高科技计划基金资助项目(BG2001013)
**作者简介：**韩京宇(1976－)，男，讲师、博士，主研方向：数据质量，数据清洗，移动对象数据库；成 瑜，工程师、硕士；董逸生，教授、博士生导师
**收稿日期：**2008-01-20 **E-mail：**hjysky@gmail.com

献[2]已有论述，本文只讨论结构转换规则。

两个文档中对应同一实体的元素常采用不尽相同的结构来表达，为了有效地计算元素间的相似度，须对其结构进行转换使其结构趋同。结构转换规则力求将 2 个 XML 元素结构在基本保持语义的前提下转换为一种统一的对应结构，以实现元素间结构的匹配。本文主要考虑 3 类结构转换状况：(1)同层次标签的一对多关系；(2)不同层次标签的转换；(3)标签缺补转换。

任何一条结构转换规则只处理 2 个文档($s_1$, $s_2$)中对应的某一元素类型(tag)和其某一子元素类型(subtags)。规则采用如下的模板描述形式：

switch tag,subtags

case Issim($s_1$.tag, $s_2$.tag) and $s_1$.subtags is like <子元素标签模板> and $s_2$.subtags is like <子元素标签模板> and <其他模板>then <转换模板>

case Issim($s_1$.tag, $s_2$.tag) and $s_1$ .hascontent and $s_2$.hasnocontent then <人工处理>

其中，Issim($s_1$.tag, $s_2$.tag)是根据标签的同义词表判断 $s_1$, $s_2$ 是否具有相同语义。

### 2.2 匹配规则

匹配规则用来判定不同文档中的 XML 元素是否代表同一实体。不同 XML 文档间同类型的 XML 元素经过转换操作后，结构和内容格式的多样性基本趋同。匹配规则的作用是对经过结构转换处理的元素寻找合适的相似性度量，以确定元素是否重复。为了准确度量 XML 元素间的相似度，匹配规则学习的任务是：

(1)确定不同类型元素重复的相似度阈值。设 2 个相似度阈值 $\theta_1$ 和 $\theta_2(\theta_1 \le \theta_2)$，任意一对元素 $e_1$ 和 $e_2$ 根据相似度阈值 $\theta_1$ 和 $\theta_2$ 按下式划分到 3 个状态空间：重复，不重复和不确定。

$$IsDup(e_1,e_2)=\begin{cases} true(sim(e_1,e_2)>\theta_2) \\ false(sim(e_1,e_2)<\theta_1) \\ confusing(\theta_1 \le sim(e_1,e_2) \le \theta_2) \end{cases} \quad (1)$$

(2)确定各层的子元素在父元素中的局部权重。XML 是一种嵌套的层次数据，一方面各个不同层次的子树区分父元素的能力，应当在其各自所属的父节点范围内确定；另一方面，嵌套元素的多层依赖关系导致一个叶节点元素的区分能力，是由其各层祖先节点在其各自的父元素内的局部权重的乘积决定的。例如，一个子元素的区分能力在局部范围内很大，但如果其祖先节点在根元素中的权重很小，则其区分根元素的能力很弱，反之亦然。为此，可以自下而上地迭代计算各个子元素相对于父元素的局部权重，最终通过链式相乘可以有效地识别出各个叶节点元素相对根元素的全局权重，后文的清洗阶段即是根据这一原则来计算根元素的相似度。

设 $e(1,1)$表示根节点元素，$e(i,j)$表示第 $i$ 层的第 $j$ 个元素，本文可以将第 $i$ 层的每一个元素看作是由它所有 $m$ 个子元素构成的向量，即 $e(i,j)=<e(i+1,j^1), e(i+1,j^2),\cdots, e(i+1,j^m)>$。设 $w(i+1,j^1), w(i+1,j^2),\cdots, w(i+1,j^m)$分别表示 $e(i,j)$中各子元素 $e(i,j)$的局部权重，且满足 $w(i+1,j^1)+w(i+1,j^2)+\cdots+w(i+1,j^m)=1$。一个元素的路径表示从根到它所经过的节点序列，即 $path(e(i,j))=<e(1,1), e(2,*),\cdots,e(i-1,*), e(i,j)>$。

### 2.3 转换规则和匹配规则的学习过程

为了便于下文说明，引入如下符号：

$e^r$ 代表一个类型为 $r$ 的复杂元素；

$e^t$ 代表一个类型为 $t$ 的 $e^r$ 的子元素；

$T$ 代表一个子元素集合；

$w_t^r$ 代表子元素 $e^t$ 相对于父元素 $e^r$ 的局部权重；

则 2 个类型为 $e^r$ 的元素间的相似性定义为式(2)：

$$sim(e^r)=\sum_{t\in T} w_t^r \times sim(e^t) \quad (2)$$

同时，设 2 个元素的叶节点分别为 $a_1, a_2,\cdots,a_n$ 和 $b_1, b_2,\cdots,b_n$，其全局权重分别记为 $w_1^g,w_2^g,\cdots,w_n^g$，则可知：

$$sim(e^r)=\sum_{i=1}^{n} w_i^g \times sim(a_i,b_i)$$

在学习的过程中，为了尽可能减少手工标注样本的工作量，每次应选择最有价值的样本进行标注，本文采用多个学习器投票的主动学习方式[3]：设有 $n$ 个学习器，每个学习器最初的输入是互不相同的训练样本集，各个学习器学习获得最初的参数。在随后各轮学习过程中，各个学习器学习相同的输入样本，标注样本的选取根据 $n$ 个学习器的不一致判断数来决定，对某一个候选样本元素如果所有学习器的投票都为 true 或 false，则视为一致，否则，视为不一致。选取不一致判断最多的样本进行手工标注。

整个学习过程按照自下而上的顺序，每次学习一个局部子树的匹配和转换规则后，进入父元素类型的学习，直至根节点元素。在每一个局部子树的学习过程中，由于结构和内容会同时影响相似度的计算，因此匹配规则学习和结构转换规则学习是一个交替的过程。在学习过程中，如果 $n$ 个学习器连续若干次具有一致的判断结果，则表示学习过程已收敛，停止学习。

#### 2.3.1 匹配规则中元素权重和相似度阈值的确定

匹配规则学习 2 个方面内容：权重的学习和相似度阈值的学习。

子元素权重的确定根据该子元素的信息增益来确定：如果某类型的子元素实例具有更强的区分能力，相应地该类型子元素具有更大的权重。一个子元素的信息增益是由于使用这个子元素分割样例而导致熵降低的程度。一个子元素类型 $E$ 相对于样例集合 $S$ 的信息增益定义成下式：

$$Gain(E)=Entropy(S)-\sum_{v\in values(E)} \frac{|S_v|}{|S|} Entropy(S_v) \quad (3)$$

其中，$values(E)$是样本中 $E$ 元素类型所具有的不同的实例值；$Entropy(S)$是集合 $S$ 的熵，它刻画集合 $S$ 的纯度(purity)，样例集 $S$ 相对于 $c$ 个状态分类的熵定义为下式：

$$Entropy(S)=\sum_{i=1}^{c} -p_i \mathrm{lb} p_i \quad (4)$$

其中，$Entropy(S_v)$是集合 $S_v$ 的熵，它刻画集合 $S_v$ 的纯度。

设某个父元素类型 $e^r$ 有 $e^t_1, e^t_2,\cdots,e^t_n$ 等 $n$ 个子元素类型，各个不同类型子元素相对于父元素类型 $e^r$ 的信息增益分别是 $Gain(e^t_1), Gain(e^t_2),\cdots,Gain(e^t_n)$。则各个子元素 $e^t_i$ $(1\le i\le n)$ 相对于父元素 $e^r$ 的局部权重为下式：

$$w(e^t_i)=\frac{Gain(e^t_i)}{\sum_{j=1}^{n} Gain(e^t_j)} \quad (5)$$

其中，$Gain(e^t_i)(1\le i\le n)$是 $e^t_i$ 在当前的训练集中的信息增益。

在学习过程中，要学习 2 个相似度阈值 $\theta_1$ 和 $\theta_2(\theta_1\le\theta_2$，初始时 $\theta_1=0$, $\theta_2=1)$。对训练样本中的正例(重复)和反例(不重复)采取不同策略：如果是正例则调整上界 $\theta_2$, $\theta_2=\theta_2-\triangle\theta_2$；如果是反例则调整下界 $\theta_1$, $\theta_1=\theta_1+\triangle\theta_1$。

#### 2.3.2 主动学习中转换操作的调整

转换规则学习的目的是对不同类型的元素学习适合它的

转换操作，使结构趋同。对于每一类型元素，转换规则学习目的是逐步确定最适合于该类型元素的转换规则。主动学习的每轮迭代中，每次对处于投票不一致和 *confusing* 区域的元素的转换规则进行重新调整。

2.3.3 主动学习算法

在学习的过程中，从最底层的元素类型开始，每次将一个底层元素类型的匹配规则和转换规则学习完成后，再转入其父节点的元素类型的学习。算法 1 是匹配和转换规则的主动学习算法。

**算法 1** 匹配和转换规则的主动学习

输入：元素类型 $e^r$ 的训练样本集 $S$；主动学习的学习器数目 $n$；主动学习停止条件 $C_{stop}$；学习过程中每轮迭代输入的样本数 $num_G$；各个子元素类型 $e^t_j(1\le j\le m)$及相关参数。

输出：$e^r$ 的各个子元素类型 $e^t_j(1\le j\le m)$的权重 $w(e^t_j)$；$e^r$ 的相似度阈值 $\theta_1$ 和 $\theta_2(\theta_1=0, \theta_2=1)$；$e^r$ 的转换规则集合 $T^r$。

从 $S$ 中选择 $n$ 组初始化学习器的输入样本 $S^l_1, S^l_2,\cdots,S^l_n$ 并手工标注，每一组包含一半重复元素对(不妨记为 $S^l_i+$)和一半不重复元素对(不妨记为 $S^l_i-$)；

```
for i=1 to n do
{
  根据训练样本 S^l_i 确定 e^r 的转换规则集合 T^r_i；
  totalgain=0;
  for each sub_element type e^t_j in S^l_i
  {
     根据式(3)计算 Gain(e^t_j)；
     totalgain=totalgain+ Gain(e^t_j)；     }
for  each  j 计算  w_i(e^t_j)；
θ_1^i= max(sim(e_1,e_2))((e_1,e_2)∈S^l_i−)
θ_1^i= min(sim(e_1,e_2))((e_1,e_2)∈S^l_i+)
}
S_rem=S−S^l_1−S^l_2,⋯,−S^l_n；
S_G=ϕ；
从 S_rem 中选择 num_G 个训练样本 S_G 作为 n 个学习器的输入；
S_rem=S_rem−S_G；
while(not C_stop)
{
   T1=n 个学习器根据各自的参数输出不一致的样本集合；
   T2= T1∪confusing 区域的元素；
   对 T1, T2 集合中的元素手工标注是否重复；
   for each i
{
根据 T1−调整 θ_1^i；
根据 T1+调整 θ_2^i；
根据 S^l_i+T1 调整子元素类型权重 w_i(e^t_j)；
根据 T2 调整 T^r_i；
}
从 S_rem 中选择 num_G 个训练样本 S_G 作为 n 个学习器的输入；
S_rem=S_rem−S_G；
}
θ_1=max{θ_1^i|1≤i≤n}；
θ_2=min{θ_2^i|1≤i≤n}；
for each j, w(e^t_j)= 1/n Σ_{i=1}^{n} w_i(e^t_j) ；
T^r= ∪_i^n T^r_i ；
```

## 3 重复 XML 元素的识别清洗算法

不失一般性，设各个叶节点元素内容是字符数据。由于传统的“排序&合并”方法中滑动窗口内元素比较的操作是一个非常耗时的原子操作[4]。在对字符串进行相似性计算时，如果采用编辑距离作为相似性度量标准，对于两个长度分别为 $m$ 和 $n$ 的字符串，由于编辑距离计算的复杂度为 $O(mn)$，计算量大。为此，本文针对 XML 设计了一种先过滤、后比较的重复识别方法。过滤的目的在于用高效的算法识别出潜在的重复元素，然后对潜在重复元素采用更精确、计算复杂度高的编辑距离进一步计算。这样可以大大加快识别的速度。由于 q-gram[5]相似性计算复杂度低(复杂度为 $O(m+n)$)，便于索引，为此过滤时采用字符串的 q-gram 余弦相似度进行度量。设 2 个叶节点元素 $l_1, l_2$ 对应的 q-gram 集合分别为 $G_{l1}, G_{l2}$，则 2 个叶节点元素的相似度 $sim(l_1, l_2)=\dfrac{|G_{l1}\cap G_{l2}|}{\max(|G_{l1}|,|G_{l2}|)}$。

下面是具体方法：扫描 XML 文档，对 XML 元素按上文获得的规则进行转换后，在内存中建立 3 张哈希矩阵 ***M***, ***L*** 和 ***W***。四维矩阵 ***M*** 的每一项 ***M***[*i*][*j*][*m*][*n*]对应第 *i* 层第 *j* 个叶子元素含有第 *m* 个字母和第 *n* 个字母组成的 2-gram(本算法以 2-gram 为例讨论)的所有元素 *id* 列表；三维矩阵 ***L***[*i*][*j*][*m*]记录了第 *i* 层第 *j* 个叶元素序号为 *m* 的元素的字符内容长度；二维矩阵 ***W*** 的每一项 ***W***[*i*][*j*]对应第 *i* 层第 *j* 个叶元素相对于根节点元素的全局权重。同时存有一张哈希表 ***SimVal***，按所有元素 *id* 哈希以记录各个元素与被处理元素的相似度值。具体的过滤算法如 hash_filter 算法所示。

**算法 2** hash_filter

输入：XML 文档 file，哈希矩阵 ***M***, ***L***, ***W***, ***SimVal***，相似度阈值 $\theta$

输出：潜在重复元素对

根据连乘得到的各个类型的叶节点元素的权重，存入哈希表 W，并按权重由大到小排列；

```
扫描 file，对每一个序号为 E_i 的 XML 元素
{
创建一个列表 list；
   按权重自大到小解析出 E_i 的各个叶节点元素内容 l_j(不妨设为第 m 层第 n 个元素)
 {
w=W[m][n];
对 l_j 中的各个 2-gram 子字符串 p_1p_2
{
在 M[m][n][p_1][p_2]中哈希得到含有 p_1p_2 的各个元素序号列表 id_1,id_2,⋯,id_z;
for each id_k(1≤k≤z)
  {
    将 id_k 插入列表 list;
    len=max(L[m][n][id_k]-1,L[m][n][E_i]-1);
  SimVal[id_k]=SimVal[id_k]+ w/len ;
If (SimVal[id_k]>θ)
{
判断为重复并输出；
}  }
将 E_i 插入到 M[m][n][p_1][p_2]中；
}
  }
  将 SimVal 中对应 list 的各元素的相似度值置零；
}
```

采用这种哈希过滤的方法，对每一个元素可以迅速找到与其共同的 2-gram 的元素。按权重排序各个叶子节点来计算相似度可以及早识别出潜在重复元素，避免冗余的计算。后续的进一步清洗阶段可以采用诸如编辑距离等的度量，来进一步处理。实验结果证实了这种方法的有效性。

## 4 实验分析

实验时 XML 解析采用 SAX 编程接口，算法通过 Java

实现。实验主要考虑 2 个方面的问题：主动学习方法的效果和哈希过滤的效果。试验数据采用 IWIZ 系统的 XML 数据。首先取出一部分异构的 XML 数据，有干净的 247 本书的信息，它作为基础数据。在这个基础数据集上，按照一定的比例引入相似重复(简称重复)元素，形成实验数据。实验数据的重复元素合成从结构和内容 2 个角度随机引入。其中，前者是指分别按同层次标签的一对多关系、单层标签对多层标签和标签缺失等 3 种情况引入。而后者是指按字符插入(insertion)、字符删除(deletion)、字符替换(substitution)、字符交换(transpostion)和单词交换(word switching)来引入相似重复的元素内容。

### 4.1 规则学习效果比较

为了衡量方法的精度，采用常用的查全率(recall)作为标准，即正确识别出的重复记录数与实际的重复记录数比率。实验数据产生产生如下：首先为原始数据集的每一本书分配一个唯一的标志(为了统计精度)，从结构和内容角度按 100%, 200%, 300%, 400%的比例引入不一致的重复元素形成 4 个实验数据集，不妨分别记为 *set*1, *set*2, *set*3, *set*4。本文在表 1、表 2 分别给出了 *set*1 和 *set*4 上主动学习和被动学习(随机选择下一个训练样本)的训练样例数目和在测试样例上精度的关系(取 5 次平均值)。其中对于每一个数据集，训练数据集包含在测试数据集中。在主动学习过程中，采用 4 个学习器同时学习，且 $num_G=2$。可以看出主动学习的效果很好。

**表 1 测试集 1 的学习效果**

| 训练集样本数 | 主动学习查全率/(%) | 被动学习查全率/(%) |
|---|---|---|
| 16 | 85 | 98 |
| 32 | 92 | 100 |
| 48 | 90 | 100 |
| 64 | 94 | 100 |
| 80 | 93 | 100 |
| 96 | 95 | 100 |
| 112 | 96 | 100 |

**表 2 测试集 4 的学习效果**

| 训练集样本数 | 主动学习查全率/(%) | 被动学习查全率/(%) |
|---|---|---|
| 20 | 90 | 80 |
| 60 | 95 | 80 |
| 100 | 100 | 92 |
| 140 | 100 | 90 |
| 160 | 100 | 89 |
| 180 | 100 | 93 |
| 200 | 100 | 94 |

与被动学习方法相比，它可以通过少量的样例学习即可得到精确的规则。这主要是因为主动学习的方法每次选择信息量大、最有利于提高分类器性能的样本来学习，从而可以及早获得好的分类性能，同时可以有效地降低增量式学习中的振荡性。

### 4.2 哈希过滤的效率

哈希过滤的方式能够高效地识别出潜在的重复元素，它虽然对每个叶子元素增加了 $O(m+n)$的过滤计算量，但有效地避免了大部分元素间无效的比较计算量(这些元素不存在重复可能性)。采用的数据集是在基础数据上，分别按 100%, 200%, 300%, 400%, 600%, 800%的比例引入重复元素形成 6 个测试数据集。表 3 是采用哈希过滤方法(记为 hash_filter+comparison)与传统的排序聚类方法(记为 sort+merge)在 6 个测试集上的运行时间比较。可见随着数据量增大，采用哈希过滤所带来的计算量减少的效果会越来越明显。

**表 3 哈希过滤时间效果比较**

| 重复记录比率/(%) | 排序聚类方法/ms | 哈希过滤方法/ms |
|---|---|---|
| 100 | 260 | 300 |
| 200 | 450 | 360 |
| 300 | 1 000 | 440 |
| 400 | 2 500 | 550 |
| 600 | 4 100 | 980 |
| 800 | 6 000 | 1 200 |

## 5 结束语

针对 XML 相似重复元素识别具有元素结构多样和元素间依赖复杂的特点，提出对于特定的应用领域可以采用主动学习的方法，得到识别异构的重复 XML 元素的结构转换规则和匹配规则，并在这个框架下给出了一种先哈希过滤再细化比较的重复识别算法。这种方法能够有效地解决 XML 重复识别时面临的结构多样性的问题，理论分析和实验表明，这是 XML 相似重复元素识别的一种有效方法。

### 参考文献


[1] Weis M, Naumann F. Detecting Duplicate Objects in XML Documents[C]//Proceedings of the 2004 International Workshop on Information Quality in Information Systems. Paris, France: [s. n.], 2004: 10-19.

[2] Tejada S, Knoblock C A, Minton S. Learning Object Identification Rules for Information Integration[D]. CaliFornia, USA: University of Southern California, 2002.

[3] Breiman l. Bagging Predictors Machine Learning[J]. 1996, 24(2): 123-140.

[4] Sung S Y, Li Zhao, Sun Peng. A Fast Filtering Scheme for Large Database Cleansing[C]//Proceedings of the 11th International Conference on Information and Knowledge Management. Virginia, USA: [s. n.], 2002: 76-83.

[5] Ukkonen E. Approximate String-matching with Q-grams and Maximal Matches[J]. Theore-tical Computer Science, 1992, 92(1): 191-212.


（上接第 46 页）

### 参考文献


[1] Park N, Parker A. Sehwa: A Software Package for Synthesis of Pipelines from Behavioral Specifications[J]. IEEE Trans. on Computer-aided Design, 1988, 7(3): 356-370.

[2] Hwang C T, Lee J H. A Formal Approach to the Scheduling Problem in High Level Synthesis[J]. IEEE Trans. on Computer-aided Design, 1991, 10(4): 464-475.

[3] Snider G. Performance-constrained Pipelining of Software Loops onto Reconfigurable Hardware[C]//Proc. of the 10th International Symposium on Field Programmable Gate Arrays. Monterey, USA: ACM Press, 2002: 177-186.

[4] Sun W, Wirthlin M J. FPGA pipeline Synthesis Design Exploration Using Module Selection and Resource Sharing[J]. IEEE Trans. on Computer-aided Design, 2007, 26(2): 254-265.

[5] Rau B R. Iterative Modulo Scheduling: An Algorithm for Software Pipelining Loops[C]//Proc. of the 27 th International Symposium on Microarchitecture. San Jose, USA: [s. n.], 1994.