第 34 卷 第 11 期 计 算 机 工 程 2008 年 6 月
Vol.34 No.11 Computer Engineering June 2008
·开发研究与设计技术· 文章编号：1000—3428(2008)11—0260—03 文献标识码：A 中图分类号：TP301

一种 FPGA 配置文件压缩算法

邢 虹，童家榕，王伶俐
(复旦大学专用集成电路国家重点实验室，上海 201203)

摘 要：基于现场可编程门阵列(FPGA)的可重构系统具有高性能和高灵活性，但随着 FPGA 规模的不断扩大，配置文件规模相应增加，导致可重构计算时间过长。该文提出一种 FPGA 配置文件压缩算法 VLZW，降低了对片外存储器的容量要求，通过减少每次重构传送的配置数据缩短了系统重构时间。

关键词：现场可编程门阵列；配置文件压缩；可重构

FPGA Configuration Compression Algorithm

XING Hong, TONG Jia-rong, WANG Ling-li
(State Key Lab of Application Specific Integrated Circuit, Fudan University, Shanghai 201203)

【Abstract】The reconfigurable system based on Field Programmable Gate Array(FPGA) has high performance and flexibility. With the enlarging of FPGA size, the dramatic increase in configuration data size has resulted in a corresponding increase in the time required for reconfiguration. This paper proposes a FPGA configuration compression algorithm——VLZW which can reduce the off-chip memory required for storing FPGA configurations. This algorithm cuts down the reconfiguration time of the system by reducing the transmittal configureation data.

【Key words】 Field Programmable Gate Array(FPGA); configuration compression; reconfigurable

1 概述

随着大规模高性能可编程器件的出现及软硬件设计方法的改进，可重构计算技术逐渐成为计算系统研究的新热点。可重构计算通过在时域上重用硬件资源来完成对计算任务的并行操作[1]。在可重构系统中，硬件信息，即现场可编程门阵列(Field Programmable Gate Array, FPGA)的配置信息，可以像软件程序一样被动态调用或修改，该系统保留了硬件计算性能且兼具软件的灵活性。

采用可重构系统实现一个计算逻辑的运行开销包括 2 个部分：(1)配置时间，配置内部可编程器件的延时；(2)计算时间，用于实际计算的时间[1]。随着FGPA规模的不断增大，配置时间越来越长，已成为制约可重构计算应用的一个重要因素。配置文件从片外存储器传送到FPGA耗费的时间是FPGA配置时间的主要部分。因此，降低配置文件的规模能显著减小FPGA配置时间。本文提出一种基于LZW(Lempel Ziv Welch)的改进压缩算法VLZW能有效减小FPGA配置文件的大小。

2 FPGA 配置文件压缩

通用的 FPGA 结构如图 1 所示。

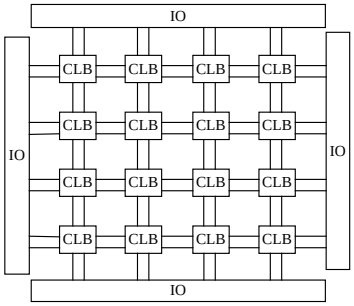

图 1 通用的 FPGA 结构示意图

FPGA 通常包括大量可编程逻辑单元(Configurable Logic Block, CLB)和可编程布线资源。由图 1 可以看出 FPGA 可编程资源重复度很高，因此，利用配置数据间的相似度可以压缩 FPGA 配置文件。

2.1 配置文件压缩简介

图 2 和图 3 分别表示压缩过程和解压缩过程。

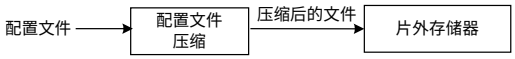

图 2 压缩过程

图 3 解压缩过程

先压缩 FPGA 配置文件，将压缩后的配置文件存储在片外存储器中，当可重构系统需要这个配置文件时，就从片外存储器中读取数据，经片内解压缩后送入片内存储单元，继续可重构计算过程。

由图 2 和图 3 可知，FPGA 配置文件压缩必须满足以下 2 个条件：

(1)解压缩电路输出的数据必须与原始配置数据完全一致。配置数据的丢失会导致芯片功能出错，甚至严重损害芯片，因此，须选择无损压缩算法。

(2)为了减小系统开销，配置文件压缩算法对应的硬件解

基金项目：国家自然科学基金资助项目(60676020)
作者简介：邢 虹(1984－)，女，硕士研究生，主研方向：FPGA 软件系统开发；童家榕，教授、博士生导师；王伶俐，副教授
收稿日期：2008-03-12 **E-mail：**052052004@fudan.edu.cn

压缩电路的实现不能太复杂。

2.2 相关研究成果

目前主流的无损压缩算法有算术编码、霍夫曼编码和字典式的LZ系列编码[2]等。基于统计模型算术编码的压缩比最高，但其硬件解压缩实现很复杂。如果采用常规霍夫曼编码，解压缩时每个输出字符串都需要查找霍夫曼树，此过程很难用流水线方式实现，解压缩效率很低。与以上 2 种压缩方法不同，基于字典式的LZ系列算法的压缩和解压缩过程并不对称，解压缩过程比压缩过程简单很多，易于用硬件实现，且该算法的实现不依赖特定FPGA结构。

基于字典式的LZ系列算法是动态建立字典的过程，使输入的字符组合能用其在字典中对应的地址(码字)来表示，从而使输入字符流转变为字典中对应的码字流输出。根据建立字典和映射码字方式的不同，可以细分LZ系列算法，主要包括LZ77, LZ78, LZSS, LZW等。LZ77 中的字典偏向于采用正文中最新的内容，它会丢弃以前有价值的字典词条；在LZ78中，字典是一个保留先前所有短语的无限序列[2]；LZSS是LZ77 的一种变体；LZW则是LZ78 的一种有效变体[2]。

已有的研究工作几乎都集中在基于字典式的 LZ 系列算法。文献[3]根据 Xilinx 公司 Virtex 系列器件的特定结构，采用改进的 LZSS 算法，取得的压缩比为 4。文献[4]改进了经典 LZW 算法，取得了较好效果，但没有仔细考虑这种改进给硬件解压缩带来的难度。文献[5]用到的压缩算法利用了 Xilinx 公司 XC6200 器件的特殊结构，取得的压缩比为 7，在几种算法中最高，但只适用于 XC6200 一种器件。另外，在主流 FPGA 厂商 Xilinx 公司的商业软件里也实现了一种基于 LZ77 的算法，用来压缩其配置文件。

2.3 VLZW 算法实现

经典LZW算法编码前先将字典初始化，使其包含所有可能出现的单个字符，剩余的项将随着输入动态添加到字典中。编码时采用一种贪婪分析算法，每次分析都要串行地检查来自字符流的字符串，从中分解出能识别的最长字符串，即字典中已出现的最长前缀(*prefix*)[2]。具体过程如下：先用已知前缀加上当前输入字符 C 形成新的扩展字符串 *string=prefix+C*；然后在字典中查找存在与该字符串相同的前缀，如果有，就以当前扩展字符串作为前缀，继续输入新的字符，否则，将该扩展字符串加入字典中，并赋与相应的固定长度码字(code)。如果码字长度为 N，则字典最多能存放 2^N项。

经典 LZW 算法为有效长度不同的码字分配固定长度的码字，浪费了一些高位，在字典建立前期和中期尤其明显。

针对上述不足，考虑到字典编码是按位逐渐增大的，改进的 VLZW 算法将 LZW 算法的定长码字输出改为变长码字输出。因为 FPGA 配置文件是二进制文件，所以初始项只有 0 和 1 两项，用 1 位表示即可。当字典的条目 i 逐渐增多时，输出的码字长度 L 也相应改变，具体表达式如下：

$$L_i=\begin{cases}1 & i=0,1\\ \lceil \mathrm{lb}i \rceil & 1<i<D_{\max}\end{cases}$$

其中，$D_{\max}$是字典的最大容量。当字典的条目增加到 3 项时，输出 2 位码字；当字典条目增加到 4 项时，输出 3 位的码字；……；当字典中的条目为$N(N>2)$位时，已输出的所有码字长度和为 $\sum_{k=3}^{N}\lceil \mathrm{lb}k \rceil+2$。而如果采用的是经典的LZW算法，输出的码字长度为 $N\lceil \mathrm{lb}D_{\max} \rceil$，采用这种变长码字输出，能在不增加算法复杂度的情况下尽可能降低输出数据的冗余度，减小输出码字的长度。

该算法的主要流程如图 4 所示。

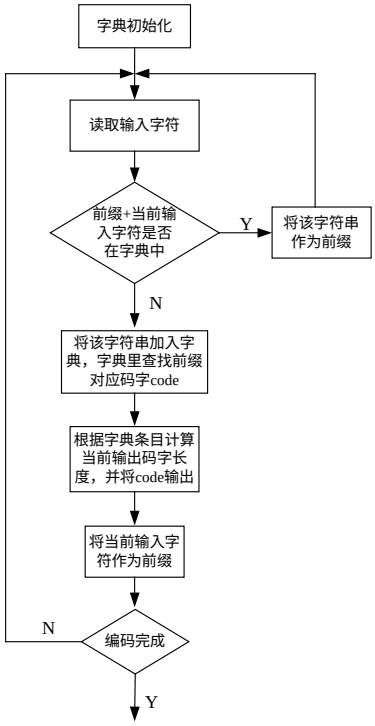

图 4 VLZW 压缩流程

下面举一个简单的例子来说明 VLZW 算法的全过程。输入字符串为“11011110010…11011110010”。首先，字典初始化前两项“0”和“1”；然后开始读取输入字符；最终输出：“0101000010010000001000100…11000”。编码过程见表 1。

表 1 编码过程

条目	输入字符串	输出码字	字典映射码字-字符串
3	11	01	10-11
4	0	01	11-10
5	1	000	100-01
6	11	010	101-111
7	10	010	110-110
8	0	000	111-00
9	10	0100	1000-010
…	…	…	…
31	1011110010	11000	11110-11011110010

在表 1 中，字典初始化时*prefix*为空，字典中有“0”和“1”两项。读入第 1 个输入字符：C=1，当前的*string=prefix+C*=1，发现字典里有此项，此时*prefix=string*=1。返回继续读下一个字符：C=1，这时组成新的字符串*string=prefix+C*=11，在字典中查找不到该项，将“11”加入字典。在字典中查找到前缀“1”对应的码字是 1，计算当前输出码字长度$L_3=\lceil \mathrm{lb}3 \rceil$=2，将码字高位补 0，输出两位码字 01。然后输入字符C=0，输出码字 01，…，每输出一次码字，字典中就新增加一项。当总输入为“11011110010”时，字典有 10 项条目，一共输出 20 位码字：01010000010010000001000。当字典条目达 31 项时，输入字符串为“101111001011”，与最初的 11 位输入一样，但输出码字长度$L_{31}=\lceil \mathrm{lb}31 \rceil$=5，即仅用 5 位的输出码字就能表示

13 位的输入字符。由此过程可知，在字典建立初期，压缩效果很差，但当字典达到一定规模时，输出码字就能替换比它长很多的源字符串。如果输入的原始数据有大量子串多次重复出现，VLZW算法能达到很好的压缩效果。

2.4 实验结果及分析

本文采用的配置文件压缩算法在 Microsoft Visual studio 6.0 环境下实现，使用的测试网表来源于 opencores，采用 Xilinx 公司的 ISE 7.1 软件生成相应配置文件，这些测试网表都采用自动工艺映射方式生成对应的配置文件，芯片资源覆盖率在 4%～80%之间，选用的器件规模较大，具体如表 2 所示。图 5 为实验的部分结果，其中，横坐标为测试用的网表例子；纵坐标为压缩比，即源配置文件与压缩后的配置文件规模之比。

表 2　部分测试网表信息

测试网表	选用器件	配置文件规模/B	VLZW		LZW	
			压缩结果/B	压缩比	压缩结果/B	压缩比
fpu100	xc4vlx40	1 532 539	50 167	30.55	59 631	25.70
tv80	xc4vlx40	1 532 534	119 430	12.83	159 752	9.59
oc_aes_core	xc4vlx40	1 532 542	128 908	11.90	202 502	7.57
Sha512	xc4vlx40	1 532 537	144 728	10.59	195 782	7.83
top_sha_core	xc4vlx40	1 532 534	222 258	6.90	295 730	5.18
multi90	xc4vlx40	1 532 539	302 276	5.07	385 060	3.98
me_jpeg	xc2v3000	1 448 812	112 868	12.84	133 012	10.90

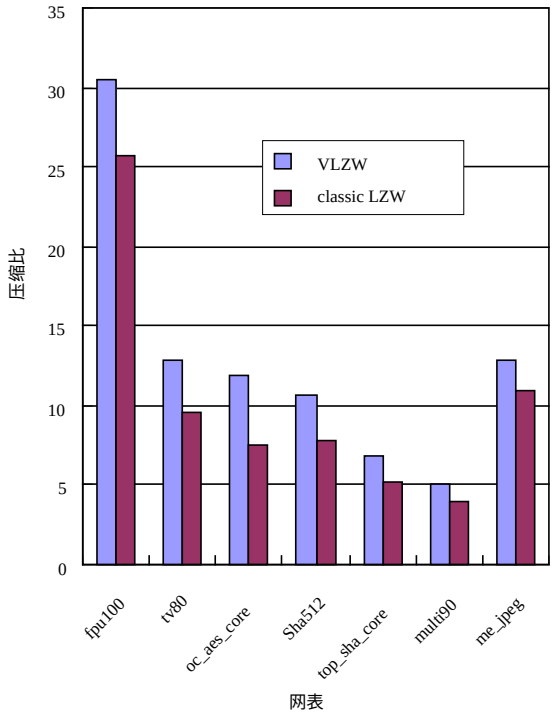

图 5　压缩结果

实验结果表明，VLZW 算法压缩比达 5.0 以上，对小规模的网表压缩效果尤其明显，比经典 LZW 算法取得的压缩比平均提高了 12%，且 VLZW 算法的通用性很强。

2.5 硬件解压缩方案

解压缩由硬件实时实现，由于编码时字典不传递给硬件，因此硬件解压缩模块在解压缩过程中需要根据码字流动态建立一个与编码过程对应的字典。字典内的数据可以保存在片内存储器块(如 RAM)中。简化的硬件解压缩设计如图 6 所示。

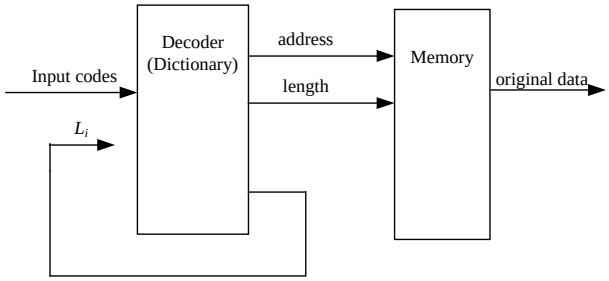

图 6　硬件解压缩结构

解压时，需要根据当前字典中的条目i计算当前码字长度L_i，并从码字流中读取L_i位长的码字，在字典(译码电路)中查询该码字，获得对应的源字符串在存储器中的地址及该字符串的长度。读出该地址指定长度的源字符串，并将其输出。与压缩过程不同的是解压缩建立字典时除了存放该字符串在存储器中的地址外还须保留该字符串本身的长度，以便最终正确读取源字符串。

此过程比压缩编码简单很多，无须使用复杂的树状结构来存储字典。在此过程中，存储器读取及处理均能用有效的流水线方式实现，因此，解压缩时间基本取决于片内存储器的读取时间。

3　结束语

本文提出了一种基于 LZW 改进的 FPGA 配置文件压缩算法 VLZW，采用变长码字输出，能有效降低输出码字的冗余度、减小输出码字流的规模并加快可重构系统的计算速度，取的了良好效果。实验结果表明，使用本算法后 FPGA 配置文件大小可以压缩为原文件的 20%以下，有效减小了芯片配置时间、降低了系统开销。该算法所对应的硬件解压缩电路较简单，有很高实用价值。

参考文献

[1] 周　博. 可重构计算的操作系统支持研究[D]. 上海: 复旦大学计算机与信息技术系, 2006.

[2] Nelson M, Gailly J. The Data Compression Book[M]. New York, USA: M & T Books, 1996.

[3] Li Zhiyuan. Scott Hauck Configuration Compression for Virtex FPGAs[C]//Proc. of IEEE Symposium on Field-programmable Custom Computing Machines. [S. l.]: IEEE Press, 2001: 147-159.

[4] Dandalis A, Prasanna V K. Configuration Compression for FPGA-based embedded systems[C]//Proc. of ACM International Symposium on Filed Programmable Gate Arrays. Monterey, California, USA: [s. n.], 2001: 173-182.

[5] Li Zhiyuan, Scott Hauck. Don't Care Discovery for FPGA Configuration Compression[C]//Proc. of ACM International Symposium on Field Programmable Gate Arrays. Monterey, California, USA: [s. n.], 1999: 91-98.