LZW算法的优化及其在FPGA上的实现

王孔华， 李若仲， 丁 浩， 陈 灿
(空军工程大学防空反导学院,西安,710051)

摘要 针对目前在高速数据采集系统的大容量存储和无损传输过程中,存在硬件难以有效进行数据处理而软件实时性不足的缺点,采用并优化了LZW数据压缩算法,并将该算法在FPGA上进行了实现。在分析LZW基本算法的基础上,给出了3个方面的优化计算方法,并进行了仿真验证。实验结果表明:优化后的LZW算法有效的提高了数据压缩的执行速度和压缩效果,验证了设计的正确性,提高了存储和传输效率。

关键词 数据处理;LZW;无损压缩;算法优化;FPGA
DOI 10.3969/j.issn.1009-3516.2015.03.009
中图分类号 TP391 **文献标志码** A **文章编号** 1009-3516(2015)03-0041-04

Implementation and Optimization of LZW Algorithm on FPGA

WANG Konghua,LI Ruozhong,DING Hao,CHEN Can
(Air and Missile Defense College, Air Force Engineering University,Xi′an 710051, China)

Abstract:Against the shortcomings of hardware in data processing and of software in real-time performance, LZW algorithm is optimized and implemented in FPGA in the bulk-storage memory and lossless transmission process. Based on analyzing LZW algorithm, the calculation method is optimized in three aspects. The optimization is inspected and verified by simulation. The results show that the use of optimized LZW algorithm can deliver data compression execution speed and efficiency.

Key words:data processing;LZW;lossless compression;algorithm optimization;FPGA

随着信息技术的发展,数据压缩技术已成为高速数据采集领域和通信领域的一项关键技术,在实现数据快速、高效存储与传输方面得到广泛应用[1]。当前,数据压缩技术一般分为有损压缩和无损压缩2类[2-3],语音、视频数据压缩一般采用有损压缩,而无损压缩技术则主要应用于文本数据、电子档案等不允许有信息遗失的文件中。常用的无损压缩有算数编码算法、Huffman编码算法、LZ编码算法等,在LZ系列数据压缩算法中,LZW(Lempel Ziv Welch,LZW)算法则是其中的杰出代表,在很多领域都得到广泛应用。目前LZW算法的压缩、解压方案大多是基于软件实现的,其缺点是速度慢,过多的占用CPU资源,所以一般应用在对实时性要求不高的场合,而对数据的实时压缩则需要用硬件完成[4]。然而,硬件压缩、解压在通用数据的无损压缩领域解决方案还不多见。随着大规模集成电路的高速发展,FPGA由于其开发周期短、并行运算能力强、可移植性好而得到广泛应用。但是,目前LZW

收稿日期:2015-01-30
基金项目:陕西省自然科学基金资助项目(2014JM8344)
作者简介:王孔华(1989—),男,山东定陶人,硕士生,主要从事集成电路与武器系统运用研究.E-mail:wkonghua@163.com

引用格式:*王孔华,李若仲,丁浩,等.LZW算法的优化及其在FPGA上的实现[J].空军工程大学学报:自然科学版,2015,16(3):41-44.*
WANG Konghua,LI Ruozhong,DING Hao,et al.Implementation and Optimization of LZW Algorithm on FPGA[J].Journal of Air Force Engineering University:Natural Science Edition,2015,16(3):41-44.

算法的 FPGA 实现还存在算法优化不合理、硬件资源分配科学、自适应性差、数据匹配及压缩性能不高等问题。因此,研究基于 FPGA 硬件实现的优化无损压缩技术具有很高的应用价值[5]。

针对上述问题,本文在多种无损压缩算法中,选取了自适应能力强、编码效率高的 LZW 算法做为基础算法进行研究,同时对 LZW 算法在编码过程、字典结构和更新方式上结合 FPGA 的实现特点对其不足作进一步的优化和改进,并在此算法的基础上设计了一种基于 FPGA 的数据实时无损压缩系统,进行了分析与验证。

1 LZW 压缩算法原理分析

LZW 算法属于字典编码,其实现数据压缩的核心思想是用简单的编码来代替复杂的字符串,在压缩过程中自适应的建立一个字典(编码表),将输入字符串映射成定长的码字输出,即字典中字符串形成的词条对应相应的编码(地址),通过查询字典确定字符串压缩代码的输出[6]。其编码流程见图 1。

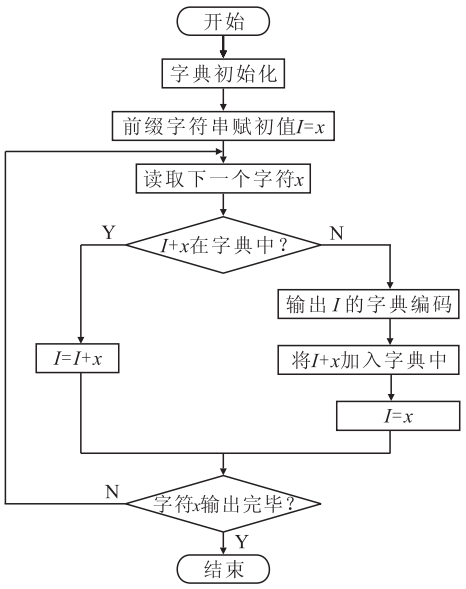


图 1 LZW 算法压缩流程

Fig.1 LZW compression algorithm flow chart

在字典大小为 1 K、字符数据为 8 bit 位时,LZW 数据压缩过程如下:首先将 0～255 初始化到字典的前 256 个词条中,这样下一个输入字符一定能在字典中查到。然后编码器逐个输入字符并累积成字符串 I,每个输入字符 x 先接在 I 后面,然后在字典中搜索 Ix,找到 Ix 则继续执行该过程;找不到 Ix 则将其存入字典,同时输出字符串 I 的指针(编码),将 I 预置为 x[7]。例如,当输入数据流为 a,b,b,a,b,a,b,a,c 时,LZW 压缩过程见表 1。

表 1 LZW 压缩编码过程

Tab.1 LZW compression coding process

Step	x	I	字典编码	更新字典	LZW 输出
			1	a	
			2	b	
			3	c	
1	b	a	4	ab	1
2	b	b	5	bb	2
3	a	b	6	ba	2
4	a	ab	7	aba	4
5	c	aba	8	$abac$	7
结束		c			3

经表 1 所示的 LZW 压缩编码后,编码输出为 1,2,2,4,7,2,3。通过以上对基本的 LZW 算法分析可以看出 LZW 算法根据输入数据的不同即时建立字典,最大限度地保证了当前字典中的内容可以最有效的压缩当前数据。

2 LZW 字典压缩算法的优化策略

由于传统的 LZW 数据压缩主要依赖编码时产生的字典(串表),而在数据压缩初期,字典中只有初始化表项,输入数据流和字典词条匹配度很低,所以压缩效果不好;当文件数据量比较大时,由于编码数据长度固定,压缩效果也会降低;同时考虑到随着压缩的进行,字典中的词条数据量越来越多,当有新数据输入时,和字典中词条比对时间大大增加,影响了字典压缩的速度[8-9]。

针对传统的 LZW 压缩算法采用 8 位数据输入、固定长编码输出时的不足,同时结合使用 FPGA 硬件实现时资源有限的实际情况,对 LZW 算法在字典大小、字典更新方式,字典结构这 3 个方面进行优化。

2.1 字典大小的优化策略

LZW 算法软件实现时由于实际情况未知,总是预分配较大的固定内存空间,这种分配方式在硬件实现是不现实的,会造成很大的资源浪费。需要结合考虑 FPGA 的内部资源来选择合适的字典大小。

字典的大小影响到算法的执行速度和压缩比,同时为了精简硬件系统和提高存取速度,一般使用 FPGA 的内部 RAM 资源,因此硬件字典空间不能设置太大。LZW 算法字典中词条由母节点、目录和字符构成,其在 FPGA 中采用结构见图 2。

图 2　字典数据结构

Fig.2　Dictionary data structure

在这种结构下 1 K 大小字典占用空间为 1 K * 8 b+2 * 1 K * 10 b=28 Kbit；4 K 大小的字典占用空间为 4 K * 8 b+4 K * 12 b * 2=128 Kbit；8 K 的字典需要 8 K * 8 b+2 * 8 K * 13 b=272 Kbit。因此在综合考虑压缩速度和效率以及 FPGA 内部 RAM 资源的情况下，在 FPGA 上的硬件实现选择字典大小为 4 K，和软件实现方式相比大大节约了存储空间，更利于硬件实现。

2.2　字典的更新方式

LZW 算法在软件实现时字典空间相对于压缩数据量小得多，在数据量较大时，几 K 的字典很快就会被填满。字典填满后需要重新选择字典更新方式，在 FPGA 硬件实现时考虑 3 种方式：停止更新字典；清空字典，重新更新；采用 FIFO 方式更新字典。第 1 种停止更新字典会导致后面数据失去适应性，压缩效果变差；第 2 种清空字典以后再重新填写字典会延长压缩数据的适应时间，影响压缩效率[10]。为了有效比较 3 种更新方式的压缩效果，选定一组数据分别进行仿真，结果见表 2。

表 2　LZW 算法不同更新方式比较

Tab.2　Compare of various LZW algorithm update method

更新方式	压缩时间/s	压缩比
停止更新	1.00	1.68∶1
清空	3.20	1.71∶1
FIFO	1.26	1.74∶1

比较表中的 3 种更新方式可以看出采用 FIFO 更新能获得较高的压缩比和相对较短的压缩时间，同时硬件也容易实现，因此采用 FIFO 更新策略。

2.3　字典结构的优化

传统的 LZW 算法给字典不同的码字分配固定长度编码，在文件数据较大时，在一定程度上使得压缩效率变低，因此改进 LZW 算法使用变长长度的编码。优化后变长长度编码的 LZW 算法的实现以 4 K 大小字典为例，标准编码长 12 bit，字典可以容纳 4 096 个编码，此时，可以把这个字典分为 5 部分，输出编码长度分别为 8 bit，9 bit，10 bit，11 bit，12 bit，同时在算法中设置变长标志，即编码长度变化时输出一个变长标志，作为解码算法的提示。字典编码见表 3，使用变长编码优化策略可以提高一定的压缩效率，节约 FPGA 的 RAM 资源。

表 3　LZW 算法字典的变长编码

Tab.3　LZW algorithm dictionary variable-length encoding

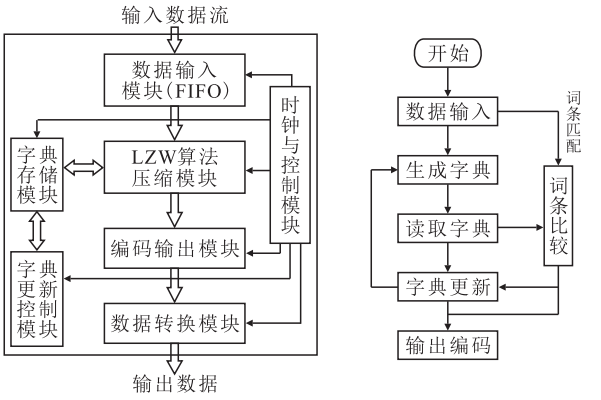
0～255	码长 8 bit 编码
256～511	码长 9 bit 编码
512～1 023	码长 10 bit 编码
1 024～2 047	码长 11 bit 编码
2 048～4 095	码长 12 bit 编码

3　LZW 优化算法的实现与验证

LZW 算法的关键在于匹配新字符串是否在已建立的字典中，软件实现采用顺序执行的方式，速度慢，过多占用 CPU 资源。而 FPGA 硬件实现可以并行执行，逻辑控制能力强，数据压缩速度快，同时 LZW 算法易于 FPGA 硬件实现。优化后的 LZW 算法 FPGA 上的硬件实现与验证，采用 Verilog 语言在 Cyclone IV 系列的 EP4CE6E22C8N 芯片上进行了测试仿真。

3.1　LZW 优化算法的 FPGA 实现

LZW 优化算法在 FPGA 上实现时，其内部功能模块主要分为数据输入缓存模块、数据处理模块、时钟与控制模块、数据输出模块，FPGA 系统内部结构组成见图 3(a)，图 3 中(b)是优化后的 LZW 算法在 FPGA 上的实现流程，与图 3(a)中各模块功能实现相对应。



(a) FPGA内部模块组成　　(b) LZW优化算法压缩流程

图 3　FPGA 内部模块框图及优化算法实现流程

Fig.3　Diagram of FPGA internal block and implementation flow of improved LZW algorithm

数据输入输出模块：主要完成 FPGA 数据传输，为保证数据与异步时钟源同步，对输入数据进行缓存的 FIFO 由 FPGA 的片内 Block RAM 构成。

时钟与控制模块：主要完成对时钟的匹配与控制，对各个功能模块分配时钟，并初始化各使能端信号，根据指令产生各个模块的控制信号，协调系统各个模块正确工作。

LZW 算法数据处理部分：主要包括 LZW 算法压缩模块、字典存储模块、字典更新控制模块，这是

LZW 算法实现的最核心部分。其中字典存储模块用来存储初始化和数据输入压缩过程中的所有字符串;字典更新模块主要完成在字典满以后对字典内容和编码写入地址的更新,根据 2.2 节分析,在此采用的是 FIFO 更新策略;LZW 算法压缩模块主要完成字符串查找,比较字典相应内容是否匹配,同时将匹配到的字符串进行编码输出。在查找过程中采用并行查找方式,通过组合逻辑来实现,所有比较在一个时钟周期内完成。

3.2 时序仿真及结果分析

根据本文分析的 LZW 优化策略,在 FPGA 上进行各个模块的硬件实现,使用 4 K 容量的字典,FIFO 更新方式,用 Verilog 语言描述整个压缩的过程。确定硬件设计参数后,输入一个测试文本文件,用一组循环数据 1,2,3,4,5,1,2,3,4,5……进行仿真,其时序仿真结果见图 4,综合后的 FPGA 资源占用情况见图 5。

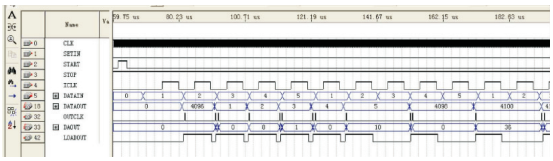


图 4 LZW 算法压缩模块时序仿真

Fig.4 LZW algorithm compression module time sequence simulation

Device	EP4CE6E22C8
Timing Models	Final
Total logic elements	1,099 / 6,272 (18 %)
Total combinational functions	983 / 6,272 (16 %)
Dedicated logic registers	475 / 6,272 (8 %)
Total registers	475
Total pins	76 / 92 (83 %)
Total virtual pins	0
Total memory bits	16,384 / 276,480 (6 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	1 / 2 (50 %)

图 5 FPGA 资源占用情况

Fig.5 FPGA resource occupancy

仿真结果表明,在测试最高时钟频率为 50 MHz 时,输入数据按照 LZW 算法进行数据压缩,输出数据 4 096,1,2,3,4……,输入输出数据与理论计算值一致,模块设计正确有效。同时 LZW 优化算法的 FPGA 硬件实现资源占用较少,节约了 FPGA 的内部资源,大大提高了数据压缩效率。

4 结语

本文在分析了基本的 LZW 算法的基础上结合 FPGA 的特点提出了 LZW 数据压缩算法在字典大小、字典更新方式、字典结构优化方面的改进方法。同时,对优化后的 LZW 算法在 FPGA 上进行了设计实现与验证,与其他采用传统 LZW 算法及其在 FPGA 上的硬件实现相比,优化后的 LZW 算法节约了 FPGA 内部资源,提高了通用数据压缩速度,优化了内部模块结构,适应性及压缩性能得到提高。经过仿真验证了 LZW 算法 FPGA 实现的正确性,更适合工程的实际应用,同时该算法还可进一步改进以提高 FPGA 硬件实现的通用性和移植性。

参考文献(References):

[1] 李定雷. 基于 FPGA 的数据实时无损压缩系统设计[D]. 太原:中北大学,2009.
LI Dinglei. Design of Real-time Lossless Compression System Based on FPGA [D]. Taiyuan: North University of China, 2009.(in Chinese)

[2] 闫阳,张正炳. 浅谈数据压缩技术[J].长江大学学报:自然科学版,2004,1(4):120-121.
YAN Yang, ZHANG Zhengbing. Techniche for Compression[J]. Journal of Yangtze University: Natural Science Edition, 2004,1(4): 120-121.(in Chinese)

[3] Ozsoy A. Swany LZSS Lossless Data Compression on CUDA[C]// IEEE International Conference on Cluster Computing (CLUSTER). [S.l.]:IEEE Press,2011:403-411.

[4] 李锦明,张文栋,毛海央,等.实时无损数据压缩算法硬件实现的研究[J].哈尔滨工业大学学报,2006,38(2):315-317.
LI Jinming, ZHANG Wendong, MAO Haiyang, et al. Research of Hardwired Real-time and No-wear Data Compression Algorithm[J]. Journal of Harbin Institute of Technology, 2006, 38(2):315-317.(in Chinese)

[5] 刘鑫,任勇峰,刘小华,等.基于遥测数据的压缩算法设计与实现[J].电子技术应用,2008,34(11):79-81.
LIU Xin, REN Yongfeng, LIU Xiaohua, et al. The Design and Realization of Compression Algorithm Based on Telemetered Data[J]. Application of Electronic Technique, 2008,34(11):79-81.(in Chinese)

[6] 王怀光,张培林,吴定海,等.分布式监测系统缓变信号无损压缩方法研究[J]. 计算机测量与控制,2014,22(3):926-929.
WANG Huaiguang, ZHANG Peiling, WU Dinghai, et al. Lossless Compression Method Research of Slowly Varying Signal for Distributed Monitoring System[J]. Computer Measurement & Control, 2014,22(3):926-929.(in Chinese)

[7] 陈庆辉,陈小松,韩德良. 中文文本压缩的 LZW 算法[J]. 计算机工程与应用,2014,50(3):112-116.
CHEN Qinghui, CHEN Xiaosong, HAN Deliang. Compression Algorithm LZW on Chinese Text [J]. Computer Engineering and Applications, 2014,50(3):112-116.(in Chinese)

[8] CUI Wei, WU Siliang. An Improved LZW Data Compression Algorithm and Its VLSI Implementation [J]. Journal of Electronic, 2008, 17(2):320-324.

[9] 许霞,马光思,鱼涛. LZW 无损压缩算法的研究与改进[J].计算机技术与发展,2009,19(4):125-127.
XU Xia, MA Guangsi, YU Tao. Research and Improvement on LZW Lossless Compression Algorithm[J]. Computer Technology and Development,2009,19(4):125-127.(in Chinese)

[10] 刘志杨,郭继昌,关欣,等.利用 FPGA 实现同步 FIFO 设置方法[J].电子测量技术,2006,29(1):65-66.
LIU Zhiyang, GUO Jichang, GUAN Xin, et al. Method of Using FPGA to Realize Synchronous FIFO Configuration [J]. Electronic Measurement Technology, 2006,29(1):65-66.(in Chinese)

(编辑:徐楠楠)