一种 LZ4 无损压缩电路设计

刘　勇，郭建刚，方　震
（中国电子科技集团公司第五十八研究所，江苏 无锡 214035）

摘　要： 为缓解存储系统中软件压缩算法对计算资源的大量占用，使用 LZ4 算法设计了一种无损压缩电路。提出了一种循环队列结构的滑动字典，结合散列表模块构建 LZ4 无损压缩电路，最后使用 Calgary 和 Canterbury 语料库，在 Xilinx KC705 FPGA 平台对电路进行验证并与软件压缩对比。结果表明所设计的 LZ4 无损压缩电路保持了软件压缩相同的压缩率的同时，在压缩效率上远超过软件压缩。

关键词： LZ4；无损压缩；滑动字典
中图分类号： TN46　　**文献标识码：** A　　DOI：10.16157/j.issn.0258-7998.222840

中文引用格式： 刘勇，郭建刚，方震. 一种 LZ4 无损压缩电路设计[J].电子技术应用，2022，48(12)：59-64.
英文引用格式： Liu Yong，Guo Jiangang，Fang zhen. The design of the lossless compression circuit based on LZ4 algorithm[J]. Application of Electronic Technique，2022，48(12)：59-64.

The design of the lossless compression circuit based on LZ4 algorithm

Liu Yong，Guo Jiangang，Fang Zhen
(No.58 Institute，China Electronic and Technology Corporation，Wuxi 214035，China)

Abstract： To decrease the costs of the implementation of compression by software in a storage system，the lossless compression circuit based on LZ4 algorithm is proposed in this paper. The sliding dictionary with circulating queue structure is used，together with the Hash Table module to design the lossless compression circuit based on LZ4. Finally，the design is tested and verified with Calgary corpus and Canterbury corpus on Xilinx KC705 FPGA platform. The results are compared with LZ4 software method and it can be concluded that the compression efficiency of LZ4 circuit method greatly faster than the software method obviously，while the compress ratios keep the same.

Key words： LZ4；lossless compression；sliding dictionary

0 引言

随着计算机和网络技术的飞速发展和用户的激增，互联网产生的数据量也呈现爆发式增长的态势。如何提高存储器利用效率，存储不断产生的海量数据，成为存储系统领域的一大难题。自两位以色列研究者 Ziv 和 Lempel 在 1977 年提出了 LZ77 压缩算法[1]以来，各种基于字典匹配的 LZ 压缩算法的变体相继被提出，其中包括 LZ78、LZW、LZO、LZSS 等。其中，基于 LZ77 的变体被广泛用于文本和位图的无损压缩，其压缩编码的效率可以很大程度上逼近信源的信息熵值[2]。LZ4 正是 LZ77 压缩算法面向处理速度进行优化所得的变体算法，其处理速度可达传统 LZ77 压缩算法的 6 倍以上[3]，目前已被广泛用于高吞吐量的存储系统[4-5]。LZ4 压缩算法的速度优势在于建立字典的过程中，减少了计算散列值和更新散列表单元的次数，并且采用直接编码的方式输出编码，减小了输出延迟[6-7]。但同样由于降低散列表更新次数，导致 LZ4 压缩算法的压缩率会高于其他变体压缩算法。

与此同时，现有的 LZ4 压缩算法基本基于 x86 架构计算机的软件实现，处理效率低。而且在存储系统的访问过程中，后台运行的压缩程序将会占用大量中央处理器(CPU)的运算资源，造成存储系统请求响应延迟上升，严重情况下甚至抵消 LZ4 压缩算法的速度优势。因此需要设计一种专用硬件电路实现 LZ4 无损压缩，以释放存储系统中的运算资源，提高系统实时性。

1 LZ4 无损压缩电路架构

LZ4 压缩算法从执行流程上可以分为扫描、匹配、编码、输出这几个主要操作步骤，所以 LZ4 压缩硬件电路架构也由此设计相应功能的子模块构成。如图 1 所示，该电路包含数据输入模块、字符串拼接模块、绝对地址产生模块、地址转换模块、滑动字典模块、散列表模块以及控制模块。

图 1 中的数据输入缓存模块用于外部数据的接收和暂存，从外部文件中读取待压缩数据块，并暂存在先进先出队列(FIFO)中，等待后级模块的读取。

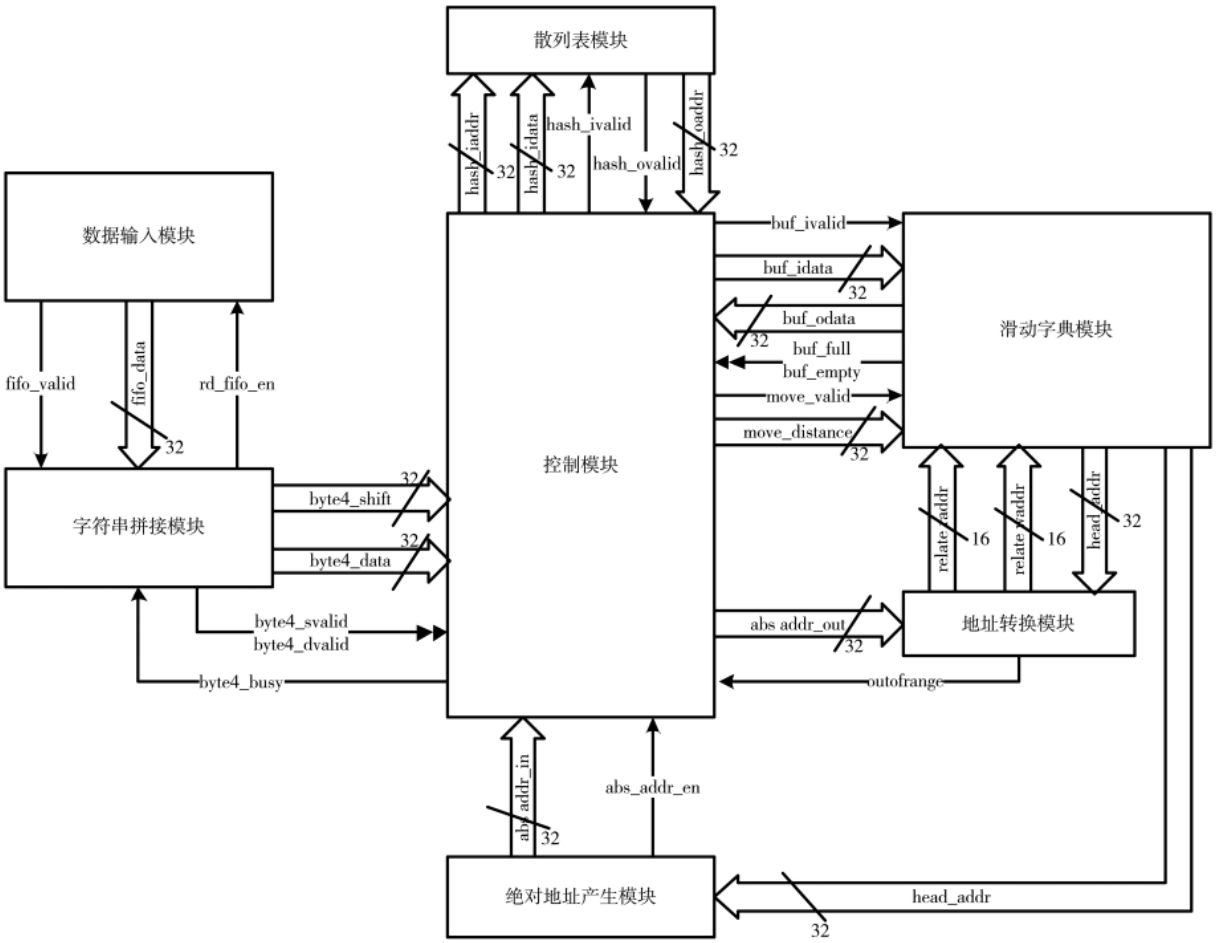

图 1　LZ4 硬件压缩电路框图

字符串拼接模块用于输入数据的切割和整合，从数据输入缓存中接收数据块中连续的双字(32 位)数据，并按照控制模块提供的匹配状态信息实时改变数据读取和拼接方式，将上一个 4 个字节数据中剩余未处理的字节与下一个 4 字节数据中的部分字节拼接成一个完整的 4 字节数据，送给控制模块。其目的是适应来自后级散列表模块或控制模块的不同类型的数据读取请求。

散列表模块用于查找匹配字符串，并将匹配距离、匹配长度等信息传递给控制模块。其内部使用运算速度较快的单级散列算法，目的是将该散列键值计算的延迟控制在 1 个时钟周期以内。

绝对地址产生模块用于生成正在处理的字符串的 32 位绝对首地址，由于每个字符在文件中的位置是独一无二的，为了区分文件中的每个字符，需要对文件中的每个字符产生独一无二的绝对地址。相应的，地址转换模块用于将 32 位绝对首地址转换为滑动字典存储器的 16 位相对地址。

滑动字典模块是本文 LZ4 压缩电路中的核心，它将外部待处理数据依次存入内部环形队列之中，并且通过控制模块提供的匹配状态信息，向控制模块送出匹配命中地址单元所存储的数据及其后续的可能存在匹配的数据，该模块工作过程中实时调整滑动字典的首尾地址，同时记录当前正在从外部存入未处理字符的绝对地址，并接收来自控制模块的读请求以及来自地址转换模块转换而成的相对地址，送出该相对地址存储单元内的数据。

控制模块负责从散列表模块、滑动字典模块、绝对地址产生模块以及字符串拼接模块中读取状态信息，据此产生控制信号；负责待匹配数据与字典数据之间的比对、相应的压缩编码的产生。

2 LZ4 主要模块

2.1 滑动字典模块

滑动字典是实现所有 LZ77 及其变体压缩算法的关键模块，其处理速度直接决定了算法的执行效率和速度。核心是字典更新与查找。一种方案是使用片外动态随机访问存储器(DDR SDRAM)实现滑动字典，是最接近软件方法的实现方案[8]，但由于 SDRAM 的访问延迟，并不适配 LZ4 的处理速度，不适用于 LZ4 压缩电路中的滑动字典的实现。另一种实现方案是使用片内分布式存储器[9]，这种电路将会限制字典的容量，且分布式存储器单元在级联时，极易造成时钟网络偏移(skew)，降低电路的时钟频率。本文采用的是在片内块内存(Block RAM)实现滑

动字典的方案，将容量为 128 KB 的片内块存储器封装成一个如图 2 所示的循环队列，以较小的资源实现数据平滑接入。该循环队列仅占用 2 倍于滑动字典长度(64 KB)的片内 RAM 容量。其基本原理是，假定队列头顺时针移动，那么在队列头移动方向前方是还未处理的待匹配数据，后方是滑动字典；队列尾移动方向前方是已经处理完成的历史数据，后方则是未定义区或更新的还未被处理的数据。

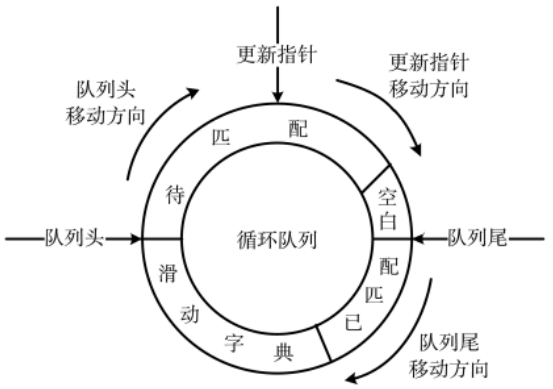

图 2　循环队列结构的滑动字典示意图

LZ4 压缩算法处理的最小单位是 1 字节，自字符串拼接模块来的 32 位数据以 4 字符形式进入本模块。初始状态下，滑动字典和已匹配区长度为 0，最初的输入字符不存在匹配，将其放入字典并更新字典长度为 4 字节。然后继续获取后续字符，经过字符拼接后搜索，进行匹配或更新字典。这个过程一直持续到字典长度达到64 KB。之后，字典前边界将跟随后边界一起移动，保持字典中数据始终是 64 KB。此时已匹配区长度开始更新。

在循环队列处理输入数据的过程中，队列头和队列尾之间始终保持了不超过一个字典长度(64 KB)的历史数据用于后续的匹配查找工作。而更新指针与队列头之间是长度不定的待压缩数据。

极端情况下，滑动字典更新时，如果队列头追赶上了更新指针，则循环队列需要等待新的数据进入待匹配区，此时循环队列显示“写满”状态，要等待更新指针往后移动才能继续更新字典，也就是队列头不能越过更新指针。同样，更新指针也不能越过队列尾，在追上队列尾时，循环队列将显示“读空”状态，需要暂停外部待压缩数据的输入。这样的规则保证了滑动字典与待压缩数据区之间的相对隔离，有利于滑动字典同时响应写入更新数据和读取已找到的字典匹配数据的请求，提高处理的并行度。

2.2 散列表模块

由于滑动字典中的数据量较大，查字典的过程中，字符串与字典内容的依次比对将耗费大量时间，为此在散列表模块中引入了基于 Hash 值的快速查找方法[10-11]。

散列表模块的作用是通过建立地址与数据之间随机的映射实现两者间的耦合。本文中使用的散列表深度为 32 KB。该表中每条存储数据是字符串在字典中的地址，访问地址则是字符串的 Hash 值，以此表征字符串。

为了将 32 位的字符串值对应到散列表的 15 位地址空间上，LZ4 算法中采用了黄金分割 Hash 函数，如式(1)所示：

$$\mathrm{Hash}_{val}=\frac{\mathrm{Strings}\times 2\ 654\ 435\ 761}{2^{17}}\bmod(2^{32}) \tag{1}$$

散列表与滑动字典之间的关系如图 3 所示，散列表的地址是由图 3 中处理指针指向的待匹配字符串计算而来的，对应地址的散列表中存储的数据是该匹配字符串的绝对首地址，即字典指针。字典指针只有在处于前边界和后边界之间时才有效，否则将被舍弃。

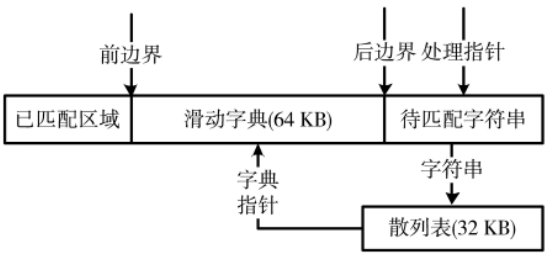

图 3　散列表与滑动字典关系图

计算 Hash 值的电路包含 1 个硬件乘法器和 1 个移位寄存器。散列表电路框图如图 4 所示。

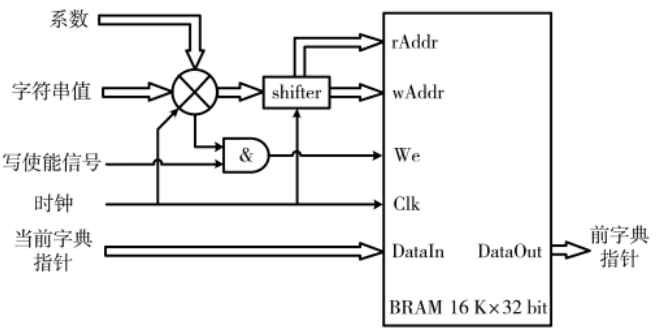

图 4　散列表电路框图

散列表所采用的片内 RAM 工作在“读优先”的模式下，先将散列表对应单元内存储的匹配串绝对地址读出，送往滑动字典模块进行判别，并用当前字符串的地址替换该散列表单元内的值，完成一次散列表单元更新过程。散列表与滑动字典模块协同进行一次更新操作需要消耗至少 4 个时钟周期才能保证流水线电路的时序稳定性，这也是整个电路中耗时最多的部分。

2.3 控制模块

控制模块负责电路中各个模块的协调和调度。因为在 LZ4 压缩电路中每个模块都有其各自的处理延迟，为了最大程度上利用硬件电路的并发性，设计的控制模

块采用流水线结构，以缩减处理一个待压缩文件的总体延迟。控制模块完整地处理一个待压缩文件的流程如图5所示，其中每个步骤消耗1个时钟周期，在流水线实现的条件下，每次从找到匹配到编码完成所消耗的时钟周期数4倍于已处理字节数。

图5所示的流程图与控制模块中的电路状态转移过程一致，每次匹配过程会生成一个编码帧，存于输出缓冲器内，最终将被整合成为一个完整的输出文件，对应的单个压缩编码帧格式[12]如图6所示。

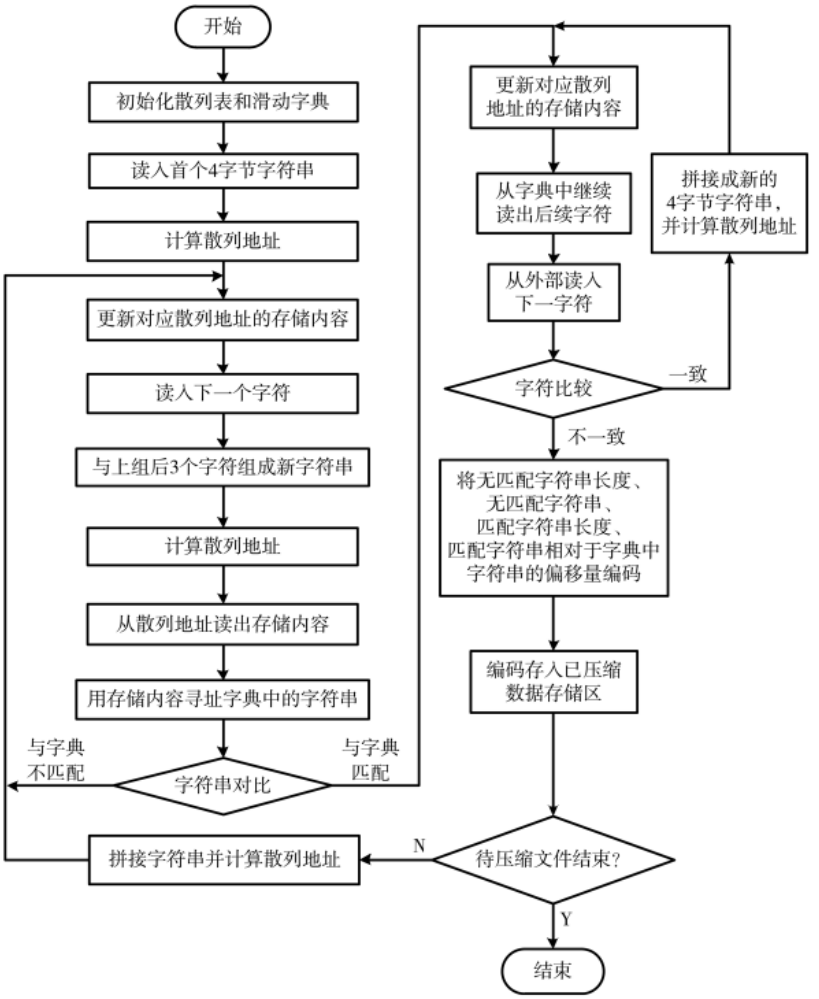

图5 控制模块工作流程图

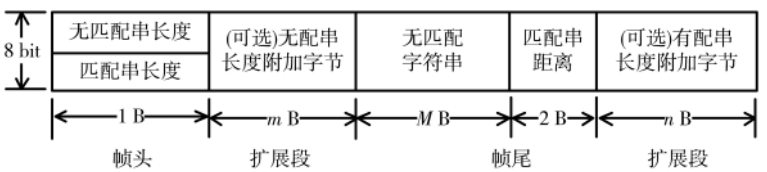

图6 单个压缩编码帧格式

压缩过程完成后，输出的文件还需要加上文件头、控制信息、校验头、文件长度、停止字以及文件校验字，LZ4压缩电路在处理结束后，输出的文件格式如图7所示。

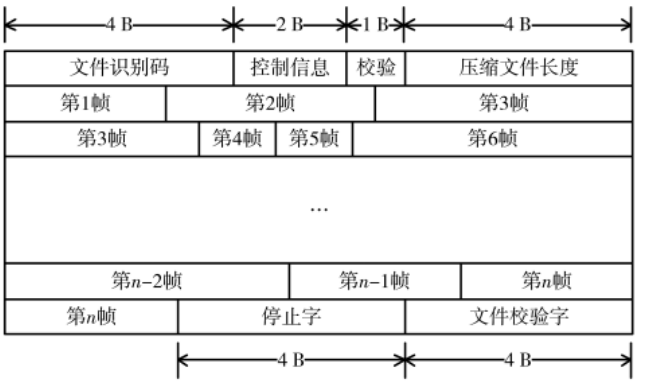

图7 单个压缩编码帧格式

3 实验验证

由于LZ4无损压缩电路处理待压缩文件的时间与文件中字符串出现的概率分布以及字符串的连续程度有关，基于实验结果普适性考虑，使用了在无损压缩性能评估中广泛应用的Calgary语料库[13]和Canterbury语料库[14]的标准压缩测试集，在Xilinx KC705平台上对LZ4无损压缩电路进行测试验证，工作频率为125 MHz。为进行参照比对，在运行环境为Core i3 3220，工作频率3.3 GHz，Windows 7 64位操作系统的通用计算机上，对LZ4无损压缩软件进行同样测试集的对比测试。

验证平台如图8所示，由KC 705板卡和通用计算机构成，两者间通过PCIe连接，采用DMA方式传输数据。寄存器阵列可以由通用计算机通过PCIe接口访问，电路工作状态实时向通用计算机上报。

无损压缩性能评估中，压缩率和压缩速度是最重要的两项指标。压缩率用来表征压缩前后文件大小变化，本文压缩率(Compression Ratio，CR)的计算方法如式(2)所示。

$$CR=\frac{\text{压缩后文件大小}}{\text{源文件大小}}\times 100\% \tag{2}$$

Calgary语料库压缩率测试结果见表1，Canterbury语料库压缩率测试结果见表2。

从表1、表2的数据可以看到，LZ4硬件压缩电路的压缩率与LZ4软件实现基本一致。其中在部分压缩测试源上，LZ4电路压缩的结果比软件压缩的结果略差，其原因是软件实现的LZ4算法中的慵懒匹配模式(Lazy Match)不利于硬件电路流水线实现，且不能带来显著的压缩率提升，为了保证硬件压缩的处理速度，并没有在硬件电路之中使用该模式。

压缩速度(Compression Speed，CS)是指在单位时间内处理数据的能力，通常采用式(3)计算，单位是MB。

$$CS=\frac{\text{源文件大小}}{\text{压缩时间}} \tag{3}$$

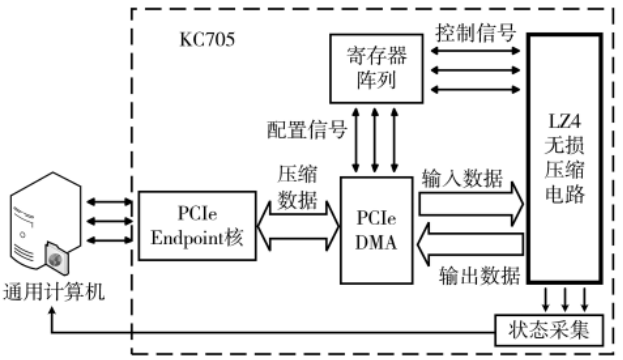

图 8 LZ4 无损压缩电路测试平台

表 1 Calgary 语料库压缩率比较

(%)

文件名	压缩率	
	LZ4 软件	LZ4 电路
bib	52.18	48.64
book1	66.93	63.76
book2	55.37	53.47
geo	94.88	88.99
news	57.69	54.27
obj1	60.11	60.49
obj2	48.72	48.38
paper1	54.43	54.19
paper2	59.10	57.01
paper3	60.81	60.24
paper4	63.76	63.44
paper5	62.37	63.04
paper6	54.08	54.46
pic	16.82	16.91
progc	52.77	52.49
progl	39.45	39.37
progp	37.90	37.86
trans	32.41	32.68

表 2 Canterbury 语料库压缩率比较

(%)

文件名	压缩率	
	LZ4 软件	LZ4 电路
alice29.txt	59.66	57.68
asyoulike.txt	63.13	60.43
cp.html	48.39	48.32
fields.c	46.77	47.03
grammar.lsp	51.38	52.35
kennedy.xls	36.49	35.96
lcet10.txt	55.61	53.96
plrabn12.txt	67.11	64.25
ptt5	16.82	16.91
sum	49.19	49.74
xargs.1	62.88	62.83

Calgary 语料库压缩速度测试结果见表 3，Canterbury 语料库压缩速度测试结果见表 4。

表 3 Calgary 语料库压缩速度比较

(MB/s)

文件名	压缩速度	
	LZ4 软件	LZ4 电路
bib	237.23	204.15
book1	200.72	153.14
book2	182.89	178.67
geo	256.64	201.18
news	239.59	199.11
obj1	221.69	217.21
obj2	255.77	230.67
paper1	269.85	182.06
paper2	210.23	166.39
paper3	220.5	166.16
paper4	204.4	168.18
paper5	239.08	170.77
paper6	272.18	189.58
pic	664.79	457.82
progc	277	198.06
progl	330.17	227.45
progp	254.53	249.39
trans	367.43	275.57
平均速度	272.48	213.09

表 4 Canterbury 语料库压缩速度比较

(MB/s)

文件名	压缩速度	
	LZ4 软件	LZ4 电路
alice29.txt	218.21	164.07
asyoulike.txt	200.29	159.67
cp.html	276.44	221.65
fields.c	227.55	199.11
grammar.lsp	186.05	206.72
kennedy.xls	484.81	254.76
lcet10.txt	233.84	173.69
plrabn12.txt	203.4	149
ptt5	677.96	457.82
sum	316.03	234.6
xargs.1	156.56	176.13
平均速度	289.19	217.93

从表 3、表 4 中数据来看，LZ4 软件比 LZ4 电路略快。但考虑到表中 LZ4 软件压缩速度是在 3.3 GHz 工作频率下测试得到的，而 LZ4 电路压缩速度仅是在125 MHz 工作频率测试得到的，两者的工作频率相差近 26 倍。简单用压缩速度对比并不能直观反映出方案优劣，因此，本文引入压缩效率(Compression Efficiency，CE)的概念进行衡量。CE 表示每周期能处理的数据的能力，在跨越软件、硬件平台时衡量处理数据的能力更直观更适合，压

缩效率计算公式如式(4)所示，单位是 B/c。

$$CE=\frac{CS}{Frequence} \tag{4}$$

为了方便讨论，仅根据表 3、表 4 中的平均压缩速度计算，得到表 5 所示压缩效率对比结果。

表 5 平均压缩效率比较

(B/c)

语料库	压缩效率	
	LZ4 软件	LZ4 电路
Calgary	0.083	1.7
Canterbury	0.088	1.74

明显可见，LZ4 电路的压缩效率远远超过 LZ4 软件的压缩效率。

4 结论

本文设计一种 LZ4 无损压缩电路，使用了循环队列结构的滑动字典以节省资源消耗，使用散列表模块以提高比对匹配速度。电路方案在 Xilinx KC705 平台上实现，并使用 Calgary 与 Canterbury 语料库进行验证，结果表明电路形式比软件形式有更高的压缩效率的同时，保持了相同的压缩率，达到释放 CPU 运算资源的目的。此外，由于 LZ4 电路的可复用性，实现多个压缩电路并行工作，成倍地缩短压缩时间也是可行的，这将进一步提升 LZ4 无损压缩硬件电路的吞吐率，为更高性能的存储系统提供高效的服务。

参考文献

[1] ZIV J，LEMPEL A.A universal algorithm for sequential data compression[J].IEEE Transaction on Information Theory，1977，23(3)：337-343.

[2] SAYOOD K.Introduction to data compression fourth edition[M].4 ed.Elsevier，2012.

[3] HARNIK D，KHAITZIN E，SOTNIKOV D，et al.A fast implementation of Deflate[C]//2014 Data Compression Conference (DCC)，2014：223-232.

[4] ALMEIDA S，OLIVEIRA V，PINA A，et al.Two high-performance alternatives to ZLIB scientific-data compression[C]//Computational Science and Its Applications-ICCSA，2014.

[5] NICOLAE B.High throughput data-compression for Cloud storage[M].Berlin Heidelberg Springer，2010.

[6] JIANG H，LIN S J.A rolling hash algorithm and the implementation to LZ4 data compression[J].IEEE Access，2020，PP(99)：1-1.

[7] LIU W，MEI F，WANG C，et al.Data compression device based on modified LZ4 algorithm[J].IEEE Trans. Consum. Electron.，2018，64(1)：110-117.

[8] RIGLER S.FPGA-based lossless data compression using GNU zip[D].Canada，University of Waterloo，2010.

[9] Ouyang Jian，Luo Hong，Wang Zilong，et al.FPGA implementation of GZIP compression and decompression for IDC services[C]//2010 International Conference on Field-Programmable Technology，2010：265-268.

[10] SADRI M J，ASAAR M R.An efficient hash-based authentication protocol for wireless sensor networks in Internet of Things applications with forward secrecy[J].International Journal of Communication Systems，2021，34(10).

[11] HAQ I U，WANG J，ZHU Y，et al.An efficient hash-based authenticated key agreement scheme for multi-server architecture resilient to key compromise impersonation[J].数字通信与网络(英文版)，2021，7(1)：11.

[12] LIU W，MEI F，WANG C，et al.Data compression device based on Modified LZ4 algorithm[J].IEEE Transactions on Consumer Electronics，2018：110-117.

[13] Calgary Corpus.Accessed：dec. 13，2019[DB/OL].[2022-04-11].http://www.data-compression.info/Corpora/Calgary-Corpus/.

[14] Canterbury Corpus.Accessed:：dec. 13，2019[DB/OL].[2022-04-11].http://www.data-compression.info/Corpora/CanterburyCorpus/.

(收稿日期：2022-04-11)

作者简介：

刘勇(1979-)，通信作者，男，博士，工程师，主要研究方向：数字通信、信号处理，E-mail：wxseugs@163.com。

郭建刚(1984-)，男，本科，工程师，主要研究方向：数字通信、信号处理。

方震(1989-)，男，本科，工程师，主要研究方向：数字通信、信号处理。

扫码下载电子文档

版权声明