第五章 实验及结果分析

第三章和第四章分别给出了 LZ4 压缩电路和半静态 Huffman 编码电路的硬件架构，并将两级压缩电路进行级联，构成了优化的 LZ4 无损压缩电路。本章首先将说明实验平台的搭建和压缩测试源的选择，然后分别对 LZ4 压缩电路和两级压缩电路进行实现和测试。

5.1 实验平台的搭建

5.1.1 硬件平台

Xilinx 公司的 KC705 板卡是一种被广泛用于功能性数字电路和高速接口开发工作的 FPGA 开发板[49]。如图 5-1 所示，板卡内包含一片型号为 Kintex-7 325T 的 FPGA 芯片，其内部包含 50950 个逻辑簇（Slice），每个 Slice 包含 4 个六输入查找表（LUT）和 8 个触发器（FF），足以实现大部分的逻辑功能。并且该 FPGA 内部还包含 890 个 18Kbit 高速片内 SRAM 单元和 840 个专用硬件乘法器，可以在实现逻辑功能的前提下提高数据处理的效率。在接口方面，FPGA 内包含 10 个时钟管理器（MMCM）和锁相环（PLL）、16 条 10.125Gbps 串行解串器（GTX Serdes）和 1 个专用 PCIe 模块，板卡上预留了 PCIe 接口，可以接入通用计算机。在 Gen2 X4 模式下，板卡与上位机之间最高的数据传输速率可达 20Gbps，高于绝大部分测试场景的数据带宽要求。

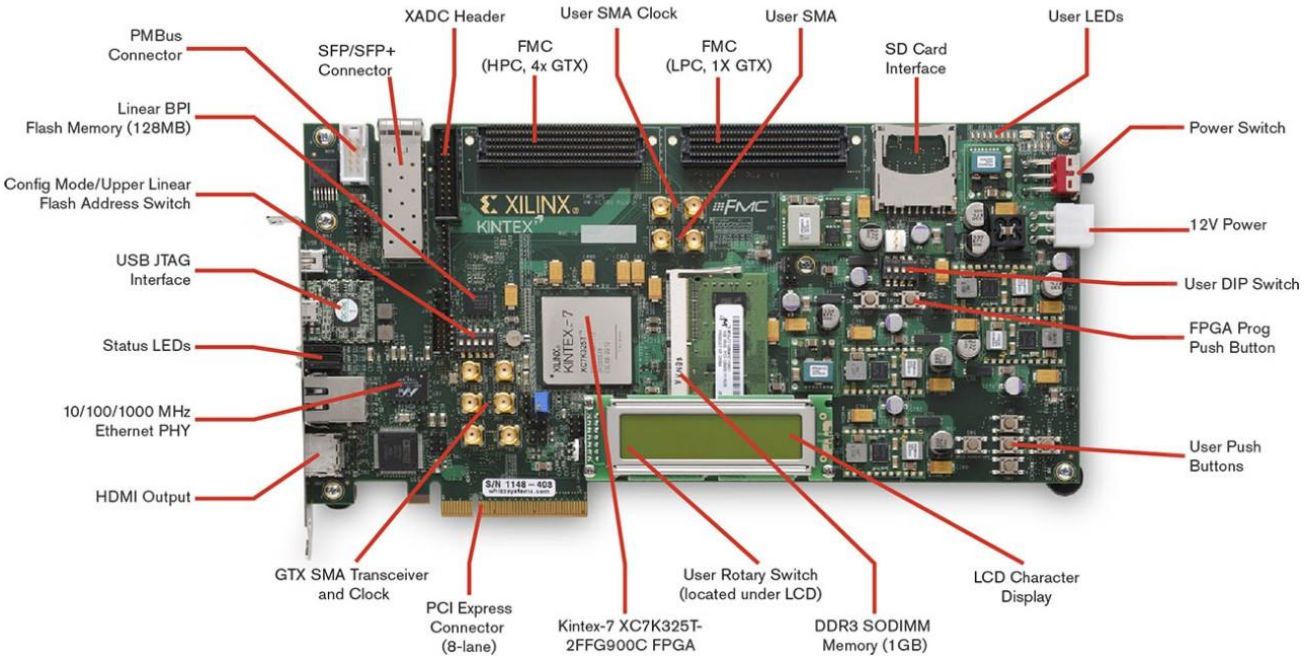


图 5-1 KC705 板卡实物图

硬件平台如图 5-2 所示，由包含压缩电路和测试通路的 KC705 板卡和通用计算机两部分组成，之间通过 PCIe 接口进行连接。数据传输使用 DMA 的方式，即板卡通过 PCIe DMA 通路在通用计算机内存中直

接写入数据块或读取数据块；寄存器访问则使用可编程输入输出模型（Programming Input/Output Model，PIO）的方式实现，即通用计算机 CPU 通过 PCIe 接口收发数据包来读写板卡中的寄存器。为了提高测试平台的监控能力，使用 FPGA 内部空闲的逻辑单元虚拟一个微控制器（Microcontroller Unit，MCU），并将其与上位机串行接口（Universal Asynchronous Receiver/Transmitter，UART）连接，用于读取板卡中的寄存器状态，并实时监控压缩电路工作状态。

通用计算机配置 CPU 为 3.3GHz 主频的 Intel Core i3 3220，内存为 1600MHz 等效频率的 8GB 双通道 DDR3 存储器，主板芯片组型号为 B75，具备 Gen3 X16 模式的 PCIe 接口，并向下兼容 Gen2 X4 模式。操作系统使用基于 32 位 Linux 2.6.32 内核的 Ubuntu10.04 LTS。除了作为实验平台的上位机，通用计算机还将用于测试 Gzip、Snappy、LZ4 等压缩软件的压缩性能，便于对比本文所述压缩电路的压缩性能。

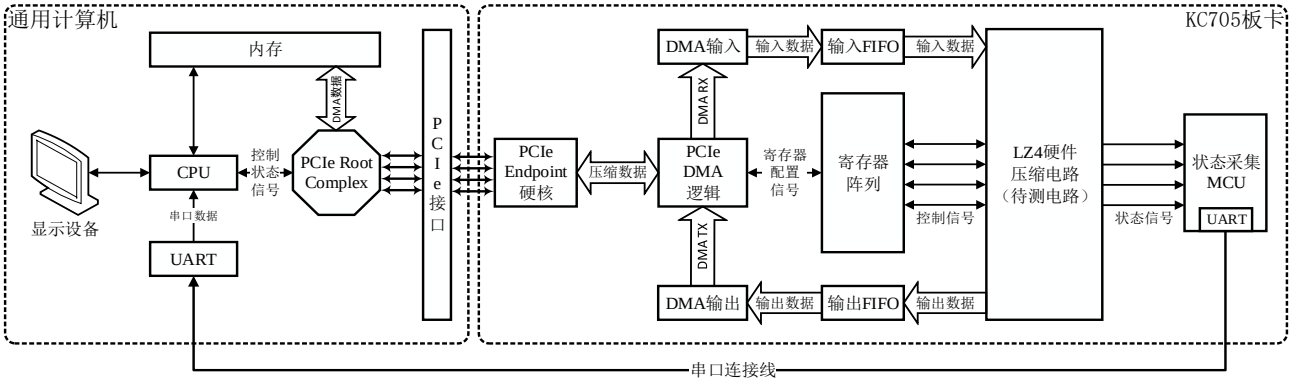


图 5-2 硬件平台

5.1.2 软件平台

软件平台的核心功能是基于 PCIe 接口的 PIO 和 DMA 通路。Xilinx 公司提供了一种总线主机模式的 DMA（Bus Master DMA）参考设计[50]，其中包含可在 Linux 操作系统下运行的 PCIe DMA 驱动程序和图形用户接口程序（GUI）。经过修改，上述参考设计中的驱动程序可以用于 DMA 数据传输和寄存器访问操作，且传输速率满足实验要求。

单次软件控制下的文件压缩测试流程如图 5-3 所示。首先，上位机通过板卡中的 DMA 寄存器监视板卡状态，一旦板卡空闲，就将 DMA 读数据长度、读内存首地址、写数据长度、写内存首地址等信息写入对应的 DMA 寄存器中，完成初始化过程。

然后，从待压缩的输入文件中依次读取固定长度的数据块，放入读内存首地址及后续的内存空间中，并通过写寄存器的方式启动 DMA 读操作，使待压缩数据从内存中被 DMA 读入板卡，完成单次写入待压缩数据块的过程。

在 DMA 读内存的过程中，如果上位机接收到来自板卡的中断，代表板卡需要送出至少一个数据块长度的已压缩数据，此时上位机响应中断并配置启动 DMA 写操作的寄存器，然后板卡将已压缩数据块放入写内存首地址及后续的内存空间中，经过一定的延时后，上位机从对应地址取出已压缩数据，以追加的方

式写入到输出文件中，最后清除中断，完成一次读出已压缩数据块的过程。

完成中断响应并返回断点后，如果输入文件还未发送完，则再次进行文件读取等一系列操作，否则，判断板卡中是否还有未被接收的数据，如果存在未收完数据，则等待下一次来自板卡的中断，直到板卡中的所有已压缩数据都被取出，测试过程结束。

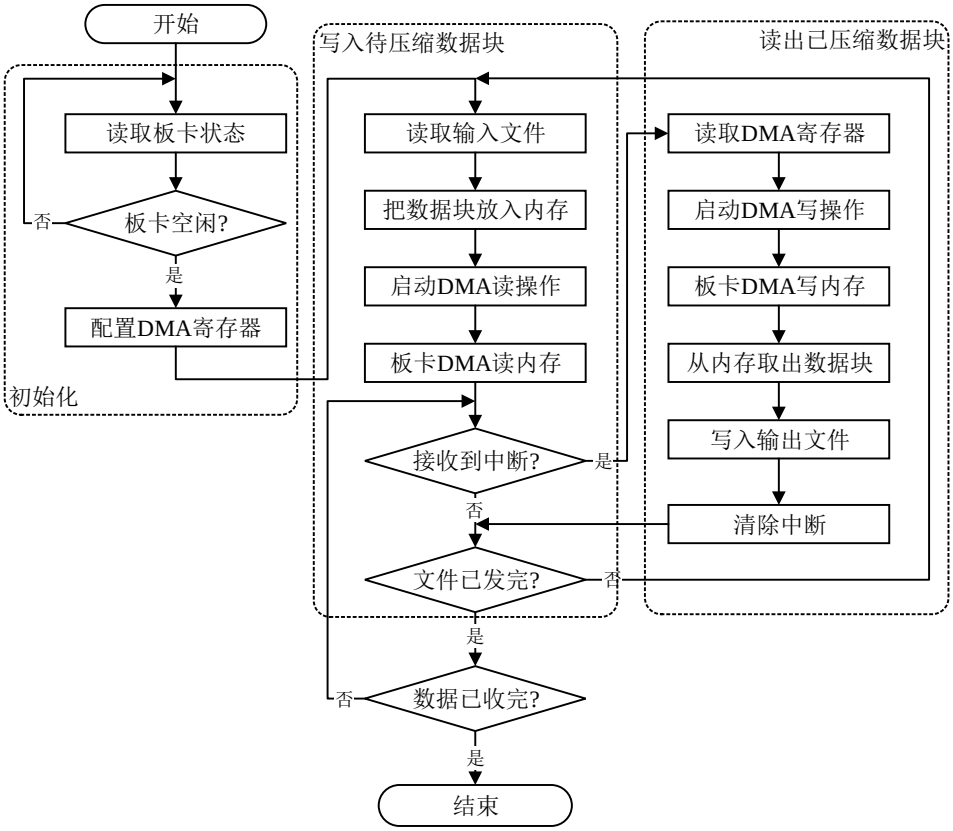


图 5-3 文件压缩测试流程

测试完成后，上位机使用软件对压缩电路输出结果进行解压并与源文件对比，如果解压后的文件与源文件内容完全相同，则认为压缩无误，其性能数据被写入测试日志中。由于本文所述的 LZ4 无损压缩电路存在兼容模式和优化模式两种工作状态，其输出的编码格式完全不同。所以，针对兼容模式的解压缩软件为版本 R131 的 LZ4 源码目标程序，针对优化模式的解压软件为添加了 Huffman 解码功能的 LZ4 源码目标程序。

5.2 标准压缩测试文件

5.2.1 Calgary 语料库

Calgary 语料库[51]是一种自上世纪 90 年代至今，被广泛用于无损压缩性能评估的文件集合，其完整版本包含 18 个类型不同的样本文件，分别代表着具有不同统计特性的信源样本。

Calgary 语料库所包含的各文件的尺寸，文件类型等信息参见表 5-1。

表 5-1 Calgary 语料库文件

文件名	尺寸（Byte）	文件类型	文件名	尺寸（Byte）	文件类型
bib	111261	refer 格式圣经文本	paper3	46526	技术论文
book1	768	小说文本	paper4	13286	技术论文
book2	610856	troff 格式非小说文本	paper5	11954	技术论文
geo	102400	二进制地理数据	paper6	38105	技术论文文本
news	377109	USENET 批处理文件	pic	513216	黑白传真图片
obj1	21504	用于 VAX 的目标代码	progc	39611	C 语言源代码
obj2	246814	用于 MAC 的目标代码	progl	71646	LISP 语言源代码
paper1	53161	技术论文	progp	49379	PASCAL 语言源代码
paper2	82199	技术论文	trans	93695	终端会话副本文件

5.2.2 Canterbury 语料库

现有的许多无损压缩算法，尤其是统计类算法，压缩小尺寸文件的压缩率性能较差，原因在于这些待压缩文件包含的符号样本不够多，难以准确的反映信源特性，不利于冗余信息的发现和消除。于是，R.Arnold 和 T.Bell 于 1997 年提出了包含 11 个测试文件的 Canterbury 语料库[51][52]。在评估无损压缩性能时，通常使用 Canterbury 语料库作为 Calgary 语料库的补充和增强，进一步提高测试的公平性。Canterbury 语料库包含的文件如表 5-2 所示。

表 5-2 Canterbury 语料库文件

文件名	尺寸（Byte）	文件类型	文件名	尺寸（Byte）	文件类型
alice29.txt	152089	英文文本	lcet10.txt	426754	技术写作文本
asyoulik.txt	125179	莎士比亚文本	plrabn12.txt	481861	诗歌文本
cp.html	24603	HTML 语言源代码	ptt5	513216	CCITT 测试集
fields.c	11150	C 语言源代码	sum	38240	SPARC 可执行文件
grammar.lsp	3721	LISP 语言源代码	xargs.1	4227	GNU 用户手册
kennedy.xls	1029744	EXCEL 表格			

5.3 压缩性能指标介绍

无论是对于压缩软件或是压缩电路，其性能指标主要由压缩率和压缩速率组成。压缩率通常有多种表示方法，本文中压缩率的定义如式（5.1）所示，即压缩率（Compression Ratio，CR）是已压缩数据尺寸（*size_dest*）和原始数据尺寸（*size_src*）的百分比，在不发生数据膨胀的情况下，*CR* 的范围为 0 到 100%。

$$CR = \frac{size\_dest}{size\_src} \times 100\% \tag{5.1}$$

压缩速率代表压缩软件或压缩电路在单位时间内处理数据的平均长度。如式（5.2）所示，压缩速率（Compression Speed，CS）的单位为兆字节每秒（Mega Byte Per Second，MBps），相关变量为起始时间（*t_start*）、结束时间（*t_end*）和已被压缩的原始数据尺寸（*size_src*）。

$$CS = \frac{size\_src\ (\text{Bytes})}{t\_end - t\_start\ (\mu s)}\ (\text{MBps}) \quad (5.2)$$

如式（5.3）所示，处理效率（Process Efficency，PE）从另一个侧面表示了压缩电路的压缩速率性能，计算方法是从所有测试结果中选取压缩速率（*CS*）最低的值，除以电路的工作频率（*freqency*），处理效率（*PE*）的单位是字节每周期（Byte Per Cycle，Byte/Cycle），周期即电路工作主时钟一个周期的时间。

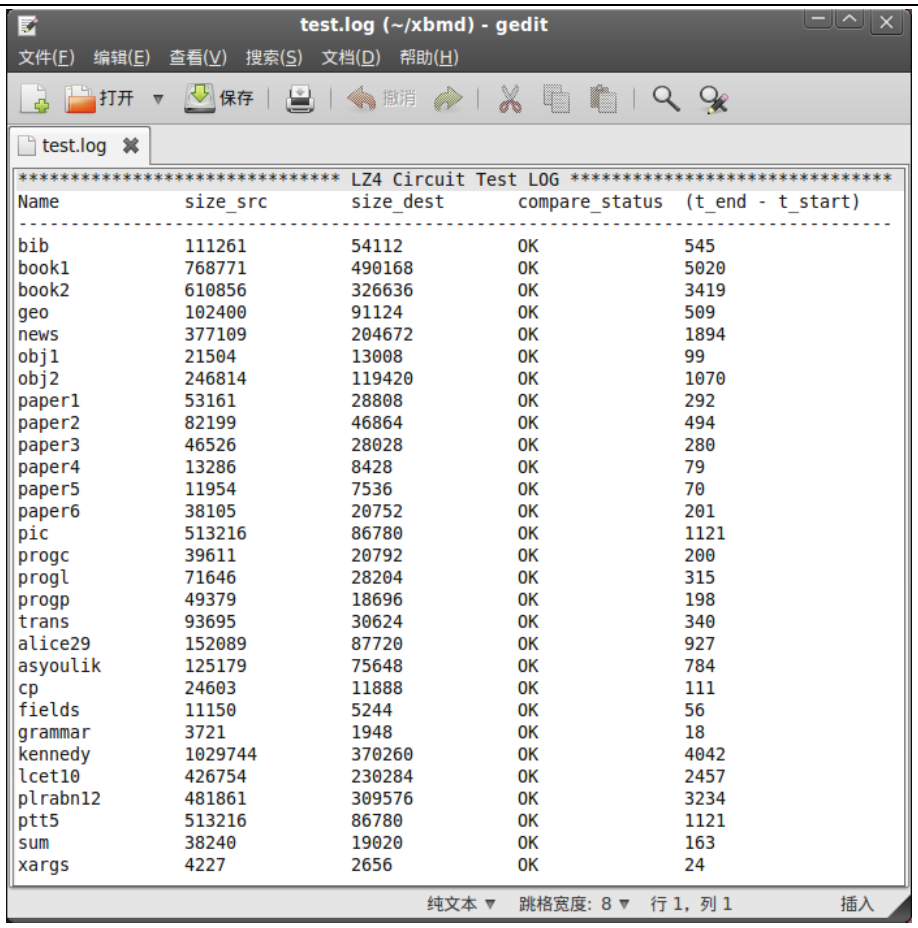
$$PE = \frac{\min(CS_1, CS_2, CS_3, ..., CS_n)\ (\text{MBps})}{freqency\ (\text{MHz})}\ (\text{Byte/Cycle}) \quad (5.3)$$

5.4 无损压缩电路功能

测试 LZ4 无损压缩电路的功能时，一方面将电路的工作状态设置为兼容模式，通过实验平台的处理，依次获得 LZ4 电路针对 Calgary 和 Canterbury 语料库内一共 29 个测试文件的压缩结果，使用原始 LZ4 软件对压缩结果进行解压缩，依次将解压缩后生成的各文件依次与对应的源文件进行二进制对比，并将源文件字节数（size_src）、压缩结果文件字节数（size_dest）、对比结果（compare_status）和压缩消耗的时间（t_end – t_start）信息记录到日志文件中，日志文件内容如图 5-4 所示。解压缩成功且对比一致，“compare_status”状态显示“OK”，解压缩失败或对比不一致则显示“ERROR”。

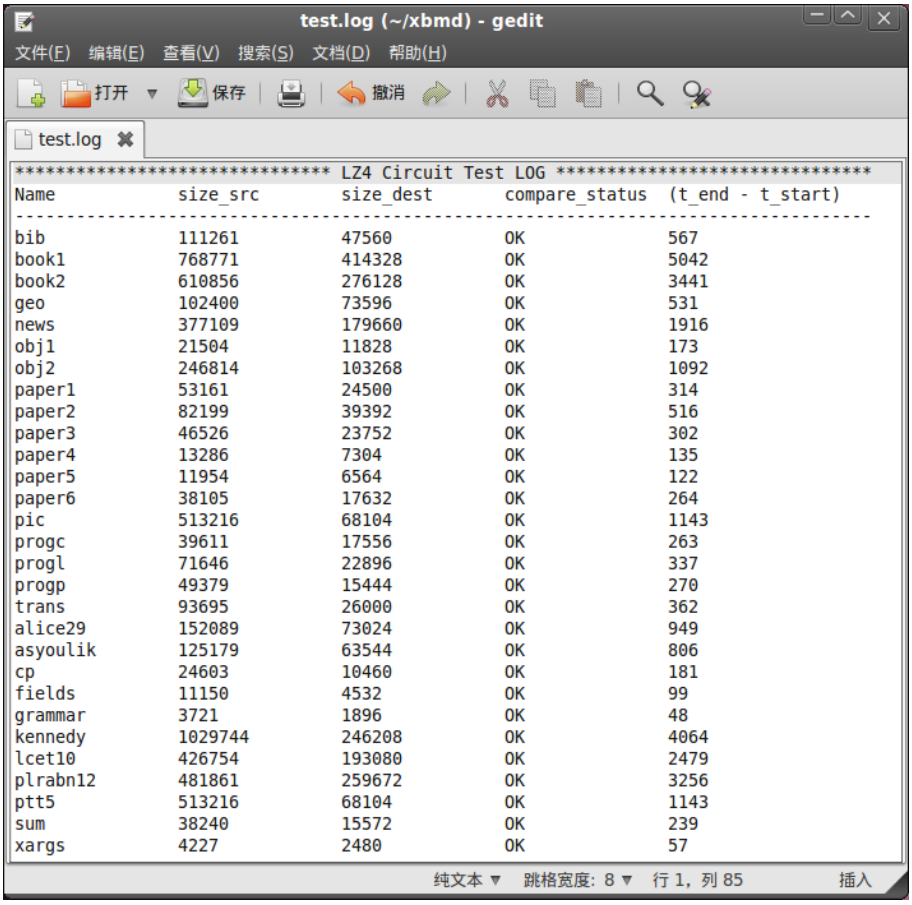
另一方面将电路的工作模式设置为优化模式，通过实验平台的处理，获得 LZ4 电路的 29 个压缩结果文件，再使用带半静态 Huffman 解码功能的 LZ4 软件对压缩结果进行解压缩，依次将解压缩后生成的各文件依次与对应的源文件进行二进制对比，并将源文件字节数、压缩后文件字节数、对比结果和压缩消耗的时间信息记录到日志文件中，日志文件内容如图 5-5 所示。

结果表明，在兼容模式和优化模式下，LZ4 压缩电路对 29 个测试文件的压缩结果字节数都比源文件的字节数少，说明压缩后体积减少。压缩结果经过 LZ4 软件解压缩后，与源文件对比均一致无误，达到了基本压缩功能指标。在优化模式下，由于加入了半静态 Huffman 编码电路，输出格式与原始 LZ4 算法不同。而在兼容模式下，使用原始 LZ4 软件可以正确的对压缩结果进行解压缩，说明兼容模式下的 LZ4 压缩电路输出格式与原始 LZ4 算法一致，达到了兼容性指标。




```
******************************* LZ4 Circuit Test LOG *******************************
Name            size_src        size_dest       compare_status  (t_end - t_start)
-------------------------------------------------------------------------------------
bib             111261          54112           OK              545
book1           768771          490168          OK              5020
book2           610856          326636          OK              3419
geo             102400          91124           OK              509
news            377109          204672          OK              1894
obj1            21504           13008           OK              99
obj2            246814          119420          OK              1070
paper1          53161           28808           OK              292
paper2          82199           46864           OK              494
paper3          46526           28028           OK              280
paper4          13286           8428            OK              79
paper5          11954           7536            OK              70
paper6          38105           20752           OK              201
pic             513216          86780           OK              1121
progc           39611           20792           OK              200
progl           71646           28204           OK              315
progp           49379           18696           OK              198
trans           93695           30624           OK              340
alice29         152089          87720           OK              927
asyoulik        125179          75648           OK              784
cp              24603           11888           OK              111
fields          11150           5244            OK              56
grammar         3721            1948            OK              18
kennedy         1029744         370260          OK              4042
lcet10          426754          230284          OK              2457
plrabn12        481861          309576          OK              3234
ptt5            513216          86780           OK              1121
sum             38240           19020           OK              163
xargs           4227            2656            OK              24
```

图 5-4 兼容模式下 LZ4 压缩电路测试日志文件




```
******************************* LZ4 Circuit Test LOG *******************************
Name            size_src        size_dest       compare_status  (t_end - t_start)
-------------------------------------------------------------------------------------
bib             111261          47560           OK              567
book1           768771          414328          OK              5042
book2           610856          276128          OK              3441
geo             102400          73596           OK              531
news            377109          179660          OK              1916
obj1            21504           11828           OK              173
obj2            246814          103268          OK              1092
paper1          53161           24500           OK              314
paper2          82199           39392           OK              516
paper3          46526           23752           OK              302
paper4          13286           7304            OK              135
paper5          11954           6564            OK              122
paper6          38105           17632           OK              264
pic             513216          68104           OK              1143
progc           39611           17556           OK              263
progl           71646           22896           OK              337
progp           49379           15444           OK              270
trans           93695           26000           OK              362
alice29         152089          73024           OK              949
asyoulik        125179          63544           OK              806
cp              24603           10460           OK              181
fields          11150           4532            OK              99
grammar         3721            1896            OK              48
kennedy         1029744         246208          OK              4064
lcet10          426754          193080          OK              2479
plrabn12        481861          259672          OK              3256
ptt5            513216          68104           OK              1143
sum             38240           15572           OK              239
xargs           4227            2480            OK              57
```

图 5-5 优化模式下 LZ4 压缩电路测试日志文件

5.5 无损压缩电路性能

测试 LZ4 无损压缩电路的性能时，首先将电路工作状态设置为兼容模式，以获得单级 LZ4 压缩电路的压缩性能。然后将电路切换为优化模式，以获得两级压缩电路的性能。并且在测试过程中，使用压缩率较好的 Gzip，压缩速率较高的 Snappy，以及 LZ4 三种压缩软件进行对比测试。根据式（5.1）和图 5-4、图 5-5 中的源文件尺寸（size_src）和压缩结果文件尺寸（size_dest）计算压缩率。根据式（5.2）和图 5-4、图 5-5 中的源文件尺寸（size_src）和压缩消耗的时间（t_end – t_start）计算压缩速率。根据式（5.3）和各测试集中最差情况下的压缩速率计算压缩效率。

5.5.1 电路工作频率和逻辑资源消耗

一般情况下，对 Verilog HDL 语言描述的数字电路进行测试之前，需要确定其最高工作频率和典型工作频率，以保证测试过程中不会因为电路时序不稳定导致错误的测试结果。对本文所述的电路代码进行逻辑综合，获得的综合报告如图 5-6 所示。

```
---------------------------

Selected Device : 7k325tffg900-3


Slice Logic Utilization:
 Number of Slice Registers:           18158  out of  407600     4%
 Number of Slice LUTs:                40786  out of  203800    20%
    Number used as Logic:             40786  out of  203800    20%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:  41819
   Number with an unused Flip Flop:   23661  out of  41819     57%
   Number with an unused LUT:          1033  out of  41819      2%
   Number of fully used LUT-FF pairs: 17125  out of  41819     41%
   Number of unique control sets:       422

IO Utilization:
 Number of IOs:                         80
 Number of bonded IOBs:                 80  out of    500    16%

Specific Feature Utilization:
 Number of Block RAM/FIFO:             122  out of    445    27%
    Number using Block RAM only:       101
 Number of BUFG/BUFGCTRLs:               3  out of     32     9%
 Number of DSP48E1s:                    28  out of    840     3%

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------
```

（a）资源消耗报告

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 4.879ns (Maximum Frequency: 204.968MHz)
   Minimum input arrival time before clock: 1.747ns
   Maximum output required time after clock: 1.124ns
   Maximum combinational path delay: No path found

Timing Details:
---------------
All values displayed in nanoseconds (ns)
```

（b）时序性能报告

图 5-6 优化的 LZ4 压缩电路综合报告

图 5-6（a）表示出了整个电路在 FPGA 上实现所需的逻辑资源，说明 5.1.1 节描述的实验平台资源量完全可以满足本文所述电路的设计实现要求。并且，FPGA 逻辑资源占用率较低，意味着本文所述的一种优化的 LZ4 无损压缩电路在 FPGA 上具有比较理想的实现面积。

图 5-6（b）表明，设计的电路中存在的关键路径所允许的最高工作频率为 204.968MHz，达到且大幅超过 100MHz 的设计指标，电路具有较低的组合逻辑延迟，具有优良的时序性能。

为了方便与参考数据进行对比，将本文所述优化的 LZ4 压缩电路的工作频率设置为 125MHz，与文献[53]中的 Gzip 压缩电路频率相同，比压缩软件运行平台的 CPU 频率 3.3GHz 低 1 个数量级以上。

5.5.2 兼容模式压缩率

使用 Calgary 语料库作为测试集时，单级 LZ4 压缩电路的压缩率性能如表 5-3 所示。以 Calgary 语料库作为测试集时，Gzip 电路的性能数据来自于文献[53]，其中缺少对测试源文件 paper3、paper4、paper5、paper6 和 trans 的测试结果，以 N/A 代替，不参与平均压缩率性能的计算。使用 Canterbury 语料库作为测试集时，其压缩率性能如表 5-4 所示。表 5-3 和表 5-4 均引入了当前主流的 Gzip，LZ4 和 Snappy 压缩软件的性能数据，用于与 LZ4 压缩电路的压缩率性能对比。

由表 5-3 和 5-4 可知，单级 LZ4 无损压缩电路压缩率性能与 Snappy、LZ4 软件基本持平，对部分测试源文件的压缩结果略优于 LZ4 压缩软件。LZ4 压缩电路的压缩率性能略低于 Gzip 电路，相比 Gzip 软件的压缩率性能还有较大的差距。

可以得出的结论是，LZ4 压缩电路在功能指标和压缩率性能指标方面已经达到甚至小幅超越 LZ4 压缩软件的水平，但与当今主流的 Gzip 压缩软件或电路相比还存在比较大的差距。

一方面，单级 LZ4 压缩电路受限于原始 LZ4 算法的缺陷，其压缩率性能是短板，尽管通过提高 Hash 表更新频次的方法优化了压缩率，但其压缩率性能仍旧难以与当今主流的高压缩率性能的 Gzip 等压缩算法比拟。另一方面，单级 LZ4 压缩电路的压缩率性能水平达到了以高压缩速率著称的 Snappy 或 LZ4 压缩

软件的水平，甚至对测试源文件 bib、book1、geo、news、asyoulik、plrabn12 的压缩率性能比 LZ4 软件提升超过 4%，原因在于这些测试源文件中存在大量匹配距离很小的字符串，相比于其它测试源文件，能够更为显著的体现提高 Hash 表更新频次获得优化压缩率的效果。同时，对测试源文件 obj1、paper5、paper6、pic、trans、fields、grammar、ptt5、sum 的压缩率性能比 LZ4 软件降低不超过 2%，原因在于提高 Hash 表更新频次会导致一些距离当前待处理字符串较远的长匹配串被较近的短匹配串取代，损失了部分压缩率。

结合表 5-3 和表 5-4，兼容模式下的 LZ4 无损压缩电路对 Calgary 语料库的平均压缩率为 52.76%，相比 LZ4 软件性能提升 2%，对 Canterbury 语料库的平均压缩率为 49.95%，相比 LZ4 软件性能提升 1%。兼容模式下的输出编码格式与 LZ4 软件完全相同，压缩率性能提升非常有限。

表 5-3 兼容模式下 Calgary 语料库压缩率性能

测试源文件	压缩率 CR（%）				
	Gzip 软件	Snappy 软件	LZ4 软件	GZIP 电路[53]	LZ4 电路
bib	31.66	52.20	52.18	44.10	48.64
book1	40.79	66.15	66.93	48.50	63.76
book2	33.83	54.70	55.37	42.90	53.47
geo	66.83	94.69	94.88	69.10	88.99
news	38.40	57.97	57.69	46.90	54.27
obj1	47.98	61.22	60.11	52.90	60.49
obj2	33.02	49.08	48.72	42.74	48.38
paper1	34.91	52.93	54.43	42.50	54.19
paper2	36.20	57.31	59.10	43.50	57.01
paper3	38.86	59.93	60.81	N/A	60.24
paper4	41.52	61.75	63.76	N/A	63.44
paper5	41.63	60.84	62.37	N/A	63.04
paper6	34.91	52.79	54.08	N/A	54.46
pic	11.00	18.11	16.82	14.60	16.91
progc	33.69	50.97	52.77	40.80	52.49
progl	22.69	36.99	39.45	33.60	39.37
progp	22.74	35.55	37.90	32.40	37.86
trans	20.12	34.42	32.41	N/A	32.68
平均	35.04	53.20	53.88	42.66	52.76

表 5-4 兼容模式下 Canterbury 语料库压缩率性能

测试源文件	压缩率 CR（%）			
	Gzip 软件	Snappy 软件	LZ4 软件	LZ4 硬件
alice29	35.77	57.87	59.66	57.68
asyoulik	39.06	61.93	63.13	60.43
cp	32.36	48.12	48.39	48.32
fields	28.00	42.40	46.77	47.03
grammar	32.84	48.37	51.38	52.35
kennedy	19.81	41.23	36.49	35.96
lcet10	33.95	54.92	55.61	53.96

plrabn12	40.52	66.22	67.11	64.25
ptt5	11.00	18.11	16.82	16.91
sum	33.97	48.96	49.19	49.74
xargs	41.07	59.36	62.88	62.83
平均	31.67	49.77	50.68	49.95

5.5.3 兼容模式压缩速率

表 5-5 和表 5-6 分别为使用 Calgary 和 Canterbury 语料库对兼容编码模式下 LZ4 压缩电路的针对压缩速率性能的测试结果，以及其他压缩软件、压缩电路的对比测试数据。其中，来自文献[53]的 Gzip 电路与本文所述的 LZ4 电路的工作频率均为 125MHz。与之对比的 Gzip、Snappy 和 LZ4 压缩软件运行的计算机的 CPU 主频为 3.3GHz。

由表 5-5 和表 5-6 可以看出，LZ4 压缩电路在兼容模式下的压缩速率性能接近 LZ4 压缩软件，只有测试源文件 pic 下，LZ4 软件的压缩速率性能比 LZ4 压缩电路高 31.13%，原因在于 pic 是位图文件，其中包含大量的重复字符，在压缩过程中非常容易扩展匹配到很长的字符串，而扩展匹配过程本质上是从字典顺序读取字符串的过程，其读取速率由存储器带宽决定。LZ4 软件把字典放置在 CPU 的高速缓冲存储器（Cache）内，带宽高达数十 GBps，远高于 LZ4 压缩电路中 32bit 宽度，125MHz 下仅 500MBps 的带宽。但 LZ4 压缩电路的压缩速率与 LZ4 软件处于同一数量级。

LZ4 压缩电路的压缩速率略高于 Snappy 压缩软件，远高于 Gzip 压缩软件和压缩电路。对 Calgary 和 Canterbury 语料库的平均压缩速率分别高达 213.09MBps 和 217.93MBps，比 Gzip 软件或 Gzip 电路的压缩速率性能高一个数量级。

表 5-5 兼容模式下 Calgary 语料库压缩速率

测试源文件	压缩速率 CS（MBps）				
	Gzip 软件	Snappy 软件	LZ4 软件	Gzip 电路[53]	LZ4 电路
bib	19.50	177.45	237.23	22.41	204.15
book1	12.89	147.67	200.72	19.07	153.14
book2	17.35	154.22	182.89	21.45	178.67
geo	7.28	263.92	256.64	13.35	201.18
news	18.68	149.17	239.59	21.93	199.11
obj1	25.21	147.29	221.69	20.50	217.21
obj2	19.08	201.48	255.77	22.40	230.67
paper1	19.81	173.16	269.85	19.07	182.06
paper2	18.67	146.00	210.23	20.50	166.39
paper3	18.12	151.06	220.50	N/A	166.16
paper4	25.16	142.86	204.40	N/A	168.18
paper5	25.54	145.78	239.08	N/A	170.77
paper6	20.61	107.64	272.18	N/A	189.58
pic	33.13	556.03	664.79	33.37	457.82

progc	26.27	130.30	277.00	21.93	198.06
progl	28.26	233.37	330.17	24.31	227.45
progp	24.79	183.57	254.53	24.79	249.39
trans	34.61	276.39	367.43	N/A	275.57
平均	21.94	193.74	272.48	21.93	213.09

需要注意的是，LZ4 压缩电路的工作频率比通用计算机的 CPU 主频低 26 倍，而平均压缩速率性能仅损失了 22%，单个文件压缩速率损失最大为 31.13%，最小为 2.02%，充分体现出了专用压缩电路相对于计算机软件的效率优势。

表征压缩速率性能的另一种表达方式是处理效率，以兼容模式下 LZ4 电路压缩速率最低的情况（149.00MBps）来计算处理效率，可得处理效率为 1.192Byte/Cycle，达到并高于设计指标 0.02Byte/Cycle。

表 5-6 兼容模式下 Canterbury 语料库压缩速率

测试源文件	压缩速率 CS(MBps)			
	Gzip 软件	Snappy 软件	LZ4 软件	LZ4 电路
alice29	16.85	163.71	218.21	164.07
asyoulik	15.81	133.03	200.29	159.67
cp	31.66	150.94	276.44	221.65
fields	24.08	129.65	227.55	199.11
grammar	23.40	120.03	186.05	206.72
kennedy	25.36	436.15	484.81	254.76
lcet10	17.17	169.62	233.84	173.69
plrabn12	12.21	145.27	203.40	149.00
ptt5	42.50	577.30	677.96	457.82
sum	20.82	100.90	316.03	234.60
xargs	24.02	91.89	156.56	176.13
平均	23.08	201.68	289.19	217.93

5.5.4 优化模式压缩率

优化模式下，LZ4 压缩电路增加了一级半静态 Huffman 编码电路，对兼容模式输出的编码进行二次压缩，以优化压缩率性能。由于半静态 Huffman 编码电路的部分统计长度是可配置的，且部分统计长度不能无限长，所以在测试过程中设置了多个不同的统计长度，以研究其对压缩率性能的影响。

理论上，部分统计的长度越长，统计结果越接近信源概率分布，压缩率越好。如图 5-7 和图 5-8 所示，分别对 Calgary 和 Canterbury 语料库进行测试的结果证实了这一思想，统计长度依次为 1KB、2KB、4KB、8KB 时，压缩率明显减少，性能明显提升。而统计长度大于 16KB 后，其压缩率性能的提升幅度非常小。并且，预设的统计长度是按指数规律增加的，随着统计长度的指数增加，其压缩率性能增量减少，体现出

部分统计长度的增加对压缩率的影响趋于减少的特点。

将部分统计长度设置为 16KB，优化模式下的 LZ4 压缩电路与 Gzip、Snappy、LZ4 压缩软件、Gzip 压缩电路以及兼容模式下的 LZ4 压缩电路的压缩率性能对比参见表 5-7 和表 5-8。

针对 Calgary 语料库的测试结果显示，在优化模式下，设置 16KB 统计长度所获得的压缩率性能超过兼容模式下的 LZ4 压缩电路，平均压缩率性能提升了 14.92%。与原始 LZ4 压缩软件的压缩率对比，其优化模式下的平均压缩率性能提升了 16.68%。

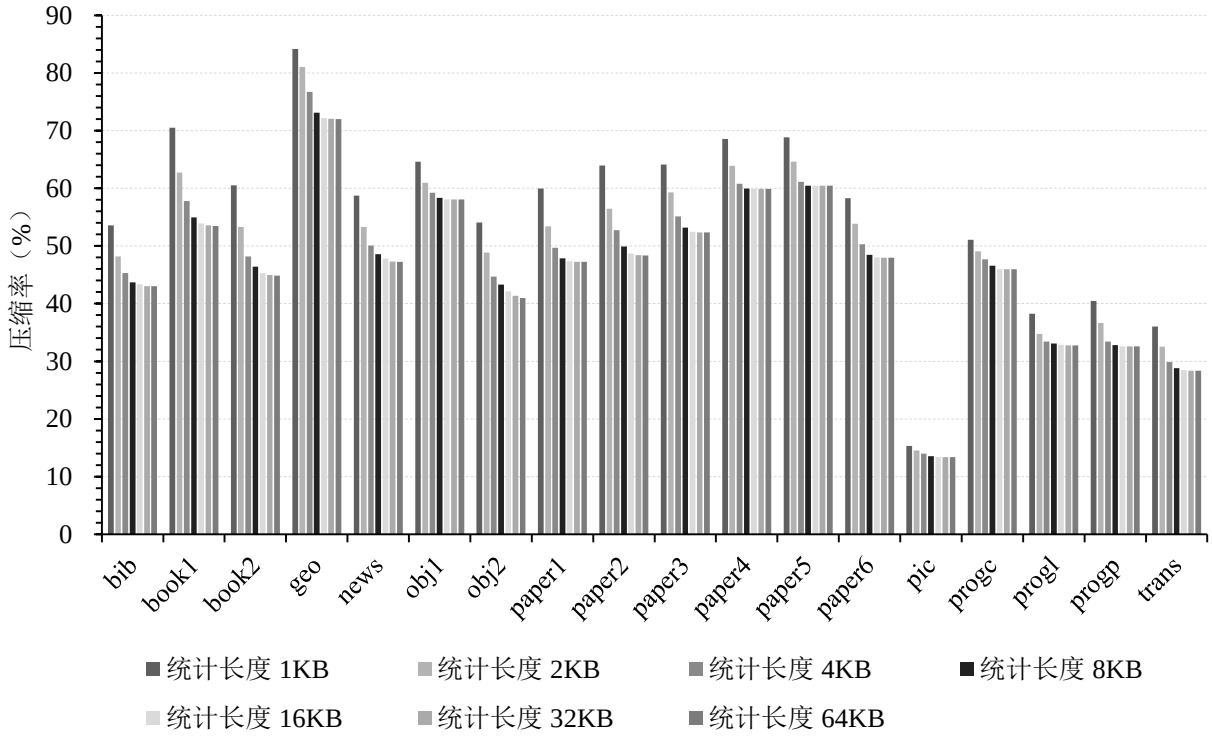


图 5-7 不同的统计长度对 Calgary 语料库压缩率的影响

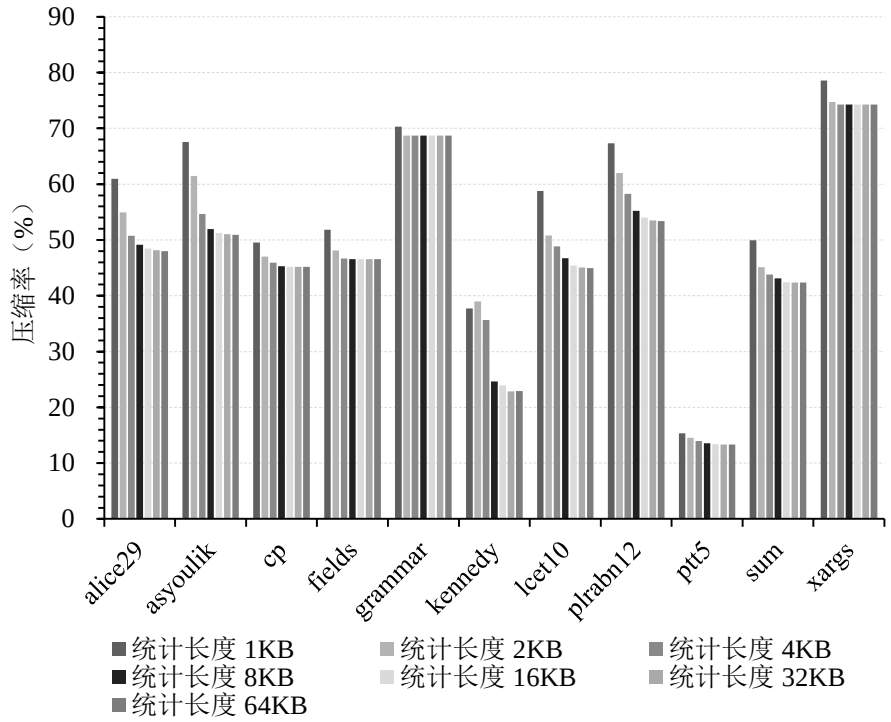


图 5-8 不同的统计长度对 Canterbury 语料库压缩率的影响

针对 Canterbury 语料库的测试结果显示，优化模式下的 LZ4 压缩电路相对于兼容模式，其平均压缩率性能提升了 14.72%，相对于 LZ4 软件，平均压缩率性能提升了 15.94%。注意到测试源文件 grammar 和 xargs 的压缩率比兼容模式下的 LZ4 电路优化效果不明显，即使用半静态 Huffman 编码进行二次压缩并未有效的降低数据体积，原因在于这两个压缩源文件本身包含的数据量非常少，只有 4KB 左右，使用半静态 Huffman 编码后，文件头部用于解码的码表等信息会占据较大比例的空间，抵消了部分二次压缩的效果，使压缩率优化的效果被弱化。

所以，在压缩源文件数据量足够的前提下，相对于 LZ4 压缩软件甚至兼容模式下的 LZ4 压缩电路，优化模式下的 LZ4 压缩电路能提供更好的压缩率性能。其压缩率接近 Gzip 电路的性能，性能提升程度已达到所期望的技术指标。

表 5-7 统计长度 16KB 的优化模式下 Calgary 语料库压缩率

测试源文件	压缩率 CR（%）					
	Gzip 软件	Snappy 软件	LZ4 软件	GZIP 电路	LZ4 电路兼容模式	LZ4 电路优化模式
bib	31.66	52.20	52.18	44.1	48.64	42.75
book1	40.79	66.15	66.93	48.5	63.76	53.89
book2	33.83	54.70	55.37	42.9	53.47	45.20
geo	66.83	94.69	94.88	69.1	88.99	71.87
news	38.40	57.97	57.69	46.9	54.27	47.64
obj1	47.98	61.22	60.11	52.9	60.49	55.00
obj2	33.02	49.08	48.72	42.74	48.38	41.84
paper1	34.91	52.93	54.43	42.5	54.19	46.09

paper2	36.20	57.31	59.10	43.5	57.01	47.92
paper3	38.86	59.93	60.81	N/A	60.24	51.05
paper4	41.52	61.75	63.76	N/A	63.44	54.98
paper5	41.63	60.84	62.37	N/A	63.04	54.91
paper6	34.91	52.79	54.08	N/A	54.46	46.27
pic	11.00	18.11	16.82	14.6	16.91	13.27
progc	33.69	50.97	52.77	40.8	52.49	44.32
progl	22.69	36.99	39.45	33.6	39.37	31.96
progp	22.74	35.55	37.90	32.4	37.86	31.28
trans	20.12	34.42	32.41	N/A	32.68	27.75
平均	35.04	53.20	53.88	42.66	52.76	44.89

表 5-8 统计长度 16KB 的优化模式下 Canterbury 语料库压缩率

测试源文件	压缩率 CR(%)				
	Gzip 软件	Snappy 软件	LZ4 软件	LZ4 电路 兼容模式	LZ4 电路 优化模式
alice29	35.77	57.87	59.66	57.68	48.01
asyoulik	39.06	61.93	63.13	60.43	50.76
cp	32.36	48.12	48.39	48.32	42.52
fields	28.00	42.40	46.77	47.03	40.65
grammar	32.84	48.37	51.38	52.35	50.95
kennedy	19.81	41.23	36.49	35.96	23.91
lcet10	33.95	54.92	55.61	53.96	45.24
plrabn12	40.52	66.22	67.11	64.25	53.89
ptt5	11.00	18.11	16.82	16.91	13.27
sum	33.97	48.96	49.19	49.74	40.72
xargs	41.07	59.36	62.88	62.83	58.67
平均	31.67	49.77	50.68	49.95	42.60

5.5.5 优化模式压缩速率

与 LZ4 编码电路不同，半静态 Huffman 编码电路处理延迟的主要来自统计数据转换成编码表所消耗的时间，在编码表生成后，编码过程等价于查表替换过程。需要注意的是，不同的统计长度会影响优化模式下的 LZ4 压缩电路的压缩速率，如图 5-9 和图 5-10 所示。可以看出，与兼容模式相比，优化模式下的 LZ4 压缩电路在压缩速率上有稍许降低。其中，统计长度为 1KB、2KB、4KB、8KB 时，其压缩速率几乎相同，统计长度在 16KB 以上时，压缩速率性能急剧下降。

原因在于，优化模式下的 LZ4 压缩电路实质上是一个两级流水线，如果第二级半静态 Huffman 编码电路能够在第一级 LZ4 编码电路将全部数据处理完成之间就生成编码表，则整个电路的压缩速率基本取决于 LZ4 编码电路的处理速率。否则，压缩速率将取决于半静态 Huffman 编码电路在等待来自前级的处理结果数据的时间，等待的时间越长，则压缩耗时越长，压缩速率越低。换言之，统计长度越长，半静态 Huffman

电路造成的延迟越大，对压缩速率性能的影响越大。综合 5.5.4 节的测试结果，在设置第二级半静态 Huffman 编码电路的统计长度的过程中，需要对压缩速率和压缩率进行折衷。

本文所述的在优化模式下的 LZ4 无损压缩电路，已将半静态 Huffman 编码电路的统计长度设置为了 16KB，其压缩速率与 Gzip、Snappy、LZ4 软件、Gzip 电路以及兼容模式下的 LZ4 电路的性能对比数据如表 5-9 和表 5-10 所示。在优化模式下，LZ4 压缩电路对体积大于 16KB 的测试源文件 bib、book1、book2、geo、news、obj1、obj2、paper1、paper2、paper3、paper6、pic、progc、progl、progp、trans、alice29、asyoulik、cp、fields、kennedy、lcet10、plrabn12、ptt5、sum 的压缩速率相比于兼容模式性能下降非常少，且尺寸越大的待压缩文件，其压缩速率下降越不明显，而对于体积小于 16KB 的源文件 paper4、paper5、grammar、xargs，压缩速率明显下降。

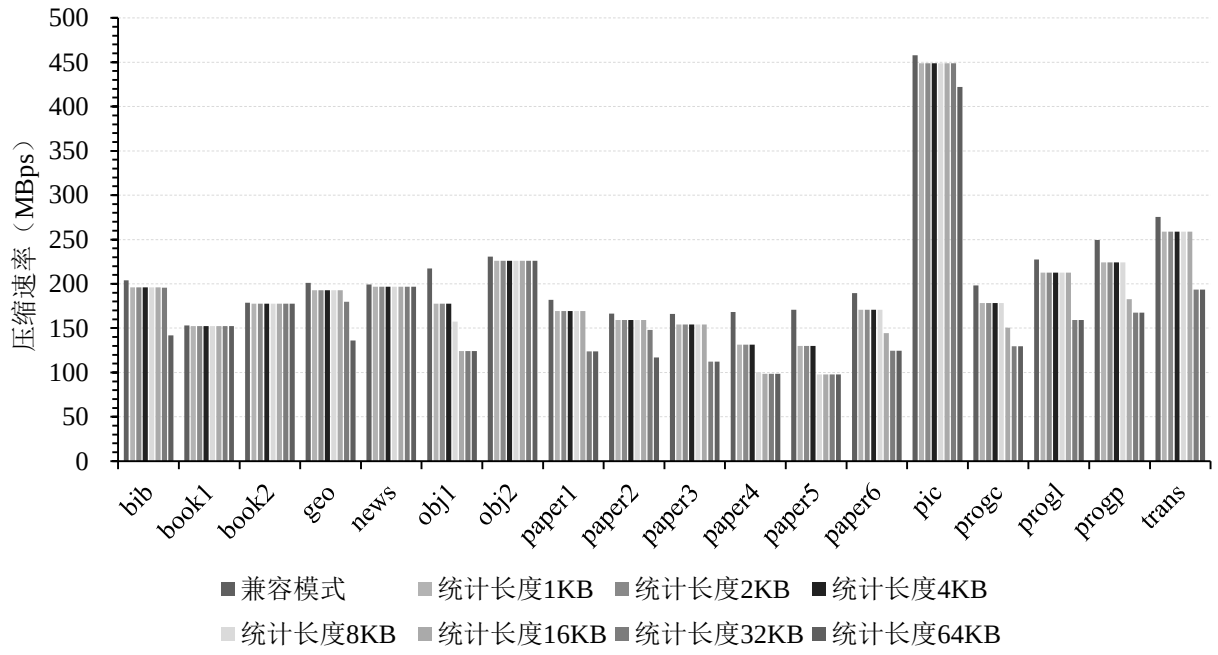


图 5-9 不同的统计长度对 Calgary 语料库压缩速率的影响

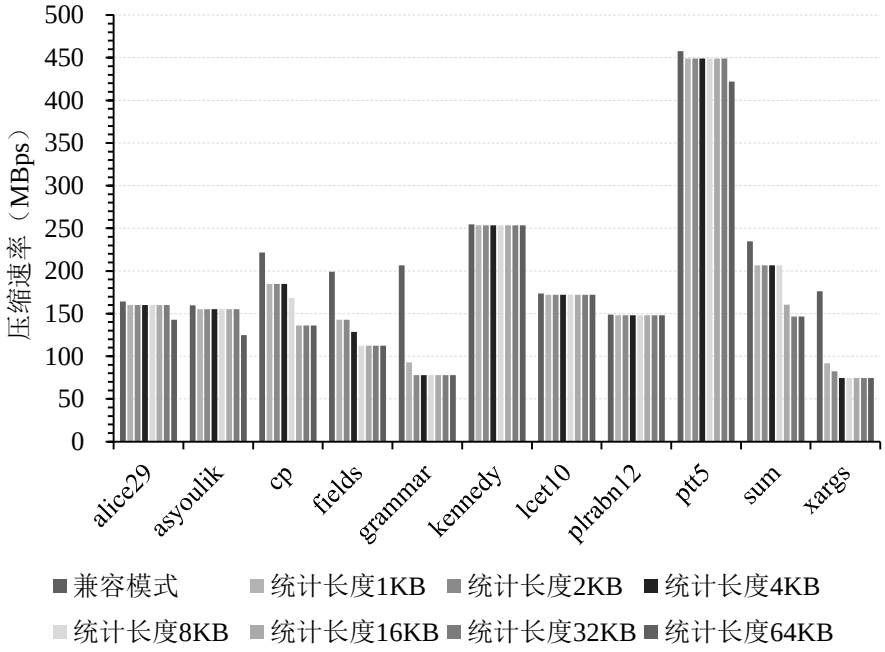


图 5-10 不同的统计长度对 Canterbury 语料库压缩速率的影响

原因在于，半静态 Huffman 编码电路必须要获取到足够的统计字符才可以进行编码，两级压缩电路在处理体积较小的测试源文件时，LZ4 电路和半静态 Huffman 编码电路几乎是串行工作的，其处理延迟是两级电路延迟之和，导致压缩速率降低。

表 5-9 兼容模式下 Calgary 语料库压缩速率

测试源文件	压缩速率 CS(MBps)					
	Gzip 软件	Snappy 软件	LZ4 软件	Gzip 电路	LZ4 电路兼容模式	LZ4 电路优化模式
bib	19.50	177.45	237.23	22.41	204.15	196.19
book1	12.89	147.67	200.72	19.07	153.14	152.47
book2	17.35	154.22	182.89	21.45	178.67	177.52
geo	7.28	263.92	256.64	13.35	201.18	192.81
news	18.68	149.17	239.59	21.93	199.11	196.81
obj1	25.21	147.29	221.69	20.50	217.21	124.21
obj2	19.08	201.48	255.77	22.40	230.67	226.00
paper1	19.81	173.16	269.85	19.07	182.06	169.25
paper2	18.67	146.00	210.23	20.50	166.39	159.27
paper3	18.12	151.06	220.50	N/A	166.16	154.01
paper4	25.16	142.86	204.40	N/A	168.18	98.55
paper5	25.54	145.78	239.08	N/A	170.77	97.79
paper6	20.61	107.64	272.18	N/A	189.58	144.45
pic	33.13	556.03	664.79	33.37	457.82	448.97
progc	26.27	130.30	277.00	21.93	198.06	150.69
progl	28.26	233.37	330.17	24.31	227.45	212.54
progp	24.79	183.57	254.53	24.79	249.39	182.62
trans	34.61	276.39	367.43	N/A	275.57	258.75
平均	21.94	193.74	272.48	21.93	213.09	185.72

优化模式下的 LZ4 压缩电路基本保持了 LZ4 压缩算法本身的速率优势，其压缩速率远高于 Gzip 软件和电路，在源文件数据量足够的前提下，压缩速率接近兼容模式下的 LZ4 电路性能和 LZ4 软件的性能，与 Snappy 软件的性能持平。

针对 Calgary 语料库的平均压缩速率可达 185.72MBps，针对 Canterbury 语料库的平均压缩速率可达 172.66MBps，相比兼容模式下的 LZ4 压缩电路，平均压缩速率性能分别下降了 13%和 21%。

以优化模式下电路压缩速率最低的情况（74.52MBps）来计算处理效率，可得处理效率最差为 0.5962Byte/Cycle，达到并高于设计指标。

表 5-10 兼容模式下 Canterbury 语料库压缩速率

测试源文件	压缩速率 CS(MBps)				
	Gzip 软件	Snappy 软件	LZ4 软件	LZ4 电路 兼容模式	LZ4 电路 优化模式
alice29	16.85	163.71	218.21	164.07	160.25
asyoulik	15.81	133.03	200.29	159.67	155.29
cp	31.66	150.94	276.44	221.65	136.19
fields	24.08	129.65	227.55	199.11	112.54
grammar	23.40	120.03	186.05	206.72	77.70
kennedy	25.36	436.15	484.81	254.76	253.38
lcet10	17.17	169.62	233.84	173.69	172.14
plrabn12	12.21	145.27	203.40	149.00	147.99
ptt5	42.50	577.30	677.96	457.82	448.97
sum	20.82	100.90	316.03	234.60	160.28
xargs	24.02	91.89	156.56	176.13	74.52
平均	23.08	201.68	289.19	217.93	172.66

5.5.6 性能与设计指标对比

如表 5-11 所示，将本章所测得的 LZ4 压缩电路的性能数据与 1.3.2 节提出的设计指标进行对比，得出的结论是，本文提出的一种优化的 LZ4 压缩算法电路均已实现功能性指标，达到性能指标。

表 5-11 性能与设计指标对比

指标名称	指标属性	指标要求	测试结果
基本压缩功能	功能性	体积减少且解压正确	体积减少且解压正确
兼容性	功能性	兼容 LZ4 软件	兼容模式下满足兼容性要求
压缩率性能	性能	提升 10%以上	提升 14%以上
处理效率	性能	≥ 0.2Byte/Cycle	≥ 0.5962Byte/Cycle
工作频率	性能	≥ 100MHz	204MHz

5.6 本章小结

本章对前几章描述的一种优化的 LZ4 无损压缩算法电路的功能和性能在 FPGA 芯片上进行了测试。测

试结果表明，压缩电路的功能和性能均已达到技术指标。并且，针对兼容模式和优化模式下的电路性能分别进行了测试，在兼容模式的压缩速率高于 Snappy 软件、Gzip 软件和电路，与 LZ4 软件基本持平，压缩率与 LZ4 软件持平；在优化模式下，压缩率优于 LZ4 软件，与 Gzip 电路持平，而压缩速率远高于 Gzip 软件和电路，略低于 LZ4 软件，与 Snappy 软件持平。本文所述的压缩电路在实验平台上测试的工作频率为 125MHz，由于设计的最高频率可达 204MHz，所以该电路的性能还有很大的提升空间。

第六章 总结与展望

6.1 总结

本文针对 LZ4 无损压缩算法，提出了一系列在不显著降低压缩速率的前提下，优化 LZ4 算法压缩率性能的方法。并且，为了保证算法的性能，对半静态 Huffman 编码过程进行了诸多优化，使其各个步骤都很容易使用硬件电路实现。为了充分发挥电路的并发优势，针对电路中容易造成速率瓶颈的模块设计了有利于提升处理效率和压缩速率的结构。整体电路提供兼容和优化两种工作模式，分别提供与 LZ4 软件相兼容的输出格式，或压缩率性能更好的压缩结果。本文所述的一种优化的 LZ4 无损压缩电路在 Xilinx KC705 开发平台上使用 Calgary 和 Canterbury 语料库进行验证和测试，测试结果表明该电路在 125MHz 的工作频率下，压缩率达到 Gzip 电路的性能，压缩速率接近或达到 Snappy、LZ4 软件的性能。

本文主要工作的总结：

(1) 在信息存储和传输的应用背景下，介绍了无损压缩尤其是字典压缩的核心思想；针对目前不断增长的压缩速率和压缩率性能需求，引入了 LZ4 无损压缩和 Huffman 编码算法；分析了无损压缩软件的局限性，以及国内外现有压缩电路的性能瓶颈；提出了本课题的研究目的和设计指标。

(2) 描述 LZ4 压缩算法的基本原理和关键的处理步骤，并分析其中的造成性能缺陷的原因；分析 LZ4 算法输出数据流格式和校验字的生成方法；基于变长编码的独特可译性原理，分别描述现有的两种 Huffman 编码方法，并分析动态 Huffman 编码在码长方面的缺陷，提出相应的优化方法。

(3) 分析 LZ4 压缩算法的压缩率性能缺陷，提出一种优化方法；针对硬件电路的并发特点，设计高效的分时访问字典缓冲器，带有四路并行 Hash 计算器的匹配电路，并行编码电路，以及配套的字符串分割电路和校验电路；将所有模块电路组合成流水线，提升电路的处理效率和压缩速率。

(4) 提出了一种半静态 Huffman 编码的实现方法，描述了其中的关键步骤和细节；设计并行统计电路，包含堆排序逻辑和双存储器结构的排序电路，建树电路，码长优化电路，码表生成电路以及编码电路，所有电路模块组合构成流水线；半静态 Huffman 编码电路与 LZ4 编码电路进行级联，实现电路在兼容模式和优化模式之间的转换。

(5) 阐述了实验平台的结构，测试方法和流程；介绍了用于评估无损压缩性能的测试集；对设计的电路进行逻辑综合，分析逻辑资源消耗量和时序性能，选取合适的工作频率来测试压缩电路性能；针对兼容模式和优化模式下的电路压缩性能分别进行测试，并与其他压缩软件和压缩电路的性能数据对比；将理论与实践相结合，着重测试和研究了不同的统计长度参数对优化模式下的电路性能的影响；将验证和测试结果与技术指标进行对比。

6.2 展望

根据本文提出的一种优化的 LZ4 无损压缩算法和电路结构，提出对本文进一步工作的展望如下。

(1) 设计命中率和速率更高的 Hash 表，使每次查表过程能够在多个可能的匹配串之中选择匹配长度最长的那个字符串，从而提高去冗余的效率，并减少伪匹配出现的概率，达到进一步提升压缩率性能的目的。

(2) 研究统计长度可变的自适应半静态 Huffman 编码方法，使统计长度根据待压缩数据量和编码延迟自动调节，提升统计电路在整个压缩过程中的利用率，统计更多的数据，从而获得更接近信源特性的统计结果，使压缩结果更加接近信源熵。

(3) 基于本文所述压缩电路，进行整体复用，实现多核心并行压缩，使压缩速率成倍提升，以满足对压缩速率要求更加苛刻的应用场合。

(4) 设计与本文所述压缩电路对应的解压缩电路，具备兼容模式和优化模式的解压缩功能。

(5) 由于实验室条件有限，现有的 FPGA 实验平台难以更好的展示优化的 LZ4 压缩算法电路的性能，如果使用先进工艺下的 ASIC 实现方式，电路的时钟频率和压缩速率有很大的提升空间。

(6) 将本文所述的 LZ4 压缩电路转化成与工艺相关的知识产权核（Intellecture Property Core，IP Core）形式，集成到各类存储或传输 ASIC 芯片中，使无损压缩的过程对用户透明。

参考文献

[1] Ebner F, Schneider V. Analysis of Web Data Compression and its Impact on Traffic and Energy Consumption[C]. Agent-Oriented Software Engineering International Workshop, Estoril, Portugal, 2014: 272-287.

[2] Kuo H C, Lin Y L. A Hybrid Algorithm for Effective Lossless Compression of Video Display Frames[J]. IEEE Transactions on Multimedia, 2012, 14(3): 500-509.

[3] Pizzolante R, Carpentieri B. Multiband and Lossless Compression of Hyperspectral Images[J]. Algorithms, 2016, 9(1): 16.

[4] Hernández-Cabronero M, Blanes I, Pinho A J, et al. Progressive Lossy-to-Lossless Compression of DNA Microarray Images[J]. IEEE Signal Processing Letters, 2016, 23(5): 698-702.

[5] Patauner C, Marchioro A, Bonacini S, et al. A Lossless Data Compression System for a Real-Time Application in HEP Data Acquisition[J]. IEEE Transactions on Nuclear Science, 2011, 58(4): 1738-1744.

[6] Sim J Y, Kim C S, Lee S U. Lossless compression of 3-D point data in QSplat representation[J]. IEEE Transactions on Multimedia, 2014, 7(6): 1191-1195.

[7] Louie H, Miguel A. Lossless Compression of Wind Plant Data[J]. IEEE Transactions on Sustainable Energy, 2012, 3(3): 598-606.

[8] Ziv J, Lempel A. 1977, A Universal Algorithm for Sequential Data Compression[J]. IEEE Transaction on Information theory, 1977, 23(3): 337-343.

[9] Nicolae B. High Throughput Data-Compression for Cloud Storage[C]. International Conference on Data Management in Grid and P2P Systems, 2010: 1-12.

[10] Xie N, Dong G, Zhang T. Using Lossless Data Compression in Data Storage Systems: Not for Saving Space[J]. IEEE Transactions on Computers, 2010, 60(3): 335-345.

[11] Nicolae B. On the Benefits of Transparent Compression for Cost-Effective Cloud Data Storage[J]. Lecture Notes in Computer Science, 2011, 6790: 167-184.

[12] Balogh G. High speed compression algorithm for columnar data storage[J]. Infocommunications Journal, 2014, 6(2): 49-52.

[13] 林涛, 蔡文婷, 陈先义, 等. 一种高性能低复杂度的基于串匹配的屏幕图像无损压缩算法[J]. 电子与信息学报, 2017, 39(2): 351-359.

[14] Danny H, Ety K, Dmitry S, et al. A Fast Implementation of Deflate[C], 2014 Data Compression Conference, Snowbird, Utah, USA, 2014, 10(1109): 223-232.

[15] Khalid Sayood. Introduction to data compression fourth edition[M]. Elsevier, 2012, 13-40, 217-249.

[16] Roelofs G. Lossless Compression Handbook[M]. Elsevier, 2003, 371–390.

[17] 李冰, 王超凡, 顾巍, 等. Gzip 压缩的硬件加速电路设计[J]. 电子学报, 2017, 45(3): 540-545.

[18] Suzanne Rigler. FPGA-Based Lossless Data Compression Using GNU Zip[D]. Canada, University of Waterloo, 2010.

[19] 李冰, 张林, 刘勇. LZMA 压缩算法 FPGA 硬件实现[J]. 北京航空航天大学学报, 2015, 41(3): 375-382.

[20] Lee J. Hardware Implementation of LZMA Data Compression Algorithm[J]. International Journal of Applied Information Systems, 2013, 5: 52-56.

[21] Li B, Zhang L, Shang Z, et al. Implementation of LZMA compression algorithm on FPGA[J]. Electronics Letters, 2014, 50(21): 1522-1524.

[22] Wei B, Li S, Zhang X. A improved hardware model for adaptive binary arithmetic coding Algorithm[J]. Lecture Notes in Electrical Engineering, 2012, 126: 487-492.

[23] Ozsoy A, Swany M, Chauhan A. Optimizing LZSS compression on GPGPUs[J]. Future Generation Computer Systems, 2013, 30(1): 170–178.

[24] Zuñiga G V, Feregrino U C, Cumplido P R. Parallel Hardware/Software Architecture for the BWT and LZ77 Lossless Data Compression Algorithms[J]. Computación Y Sistemas, 2006, 10(2): 172-188.

[25] Fogg C E. Survey of software and hardware VLC architectures[C]. SPIE - The International Society for Optical Engineering, 1994: 29-37.

[26] Lin C W, Wu J L, Chuang Y J. Two algorithms for constructing efficient huffman-code based reversible variable length codes[J]. IEEE Transactions on Communications, 2007, 56(1): 81-89.

[27] 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: 623-638.

[28] Lin M B, Chang Y Y. A New Architecture of a Two-Stage Lossless Data Compression and Decompression Algorithm[J]. IEEE Transactions on Very Large Scale Integration Systems, 2009, 17(9): 1297- 1303.

[29] Nunez J L, Jones S. Gbit/s lossless data compression hardware[J]. IEEE Transactions on Very Large Scale Integration Systems, 2003, 11(3): 499-510.

[30] Zaretsky D C, Mittal G, Banerjee P. Streaming implementation of the ZLIB decoder algorithm on an FPGA[C]. IEEE International Symposium on Circuits and Systems (ISCAS), China, Taiwan, 2009: 2329-2332.

[31] Hwang S A, Wu C W. Unified VLSI systolic array design for LZ data compression[J]. IEEE Transactions on Very Large Scale Integration Systems, 2001, 9(4): 489-499.

[32] Papadopoulos K, Papaefstathiou I, Titan R. A Multigigabit Reconfigurable Combined Compression/ Decompression Unit[J]. Acm Transactions on Reconfigurable Technology & Systems, 2010, 3(2): 126-129.

[33] Gomes R D, Neto M G D S, Filho G L D S. A solution for transmitting and displaying UHD 3D raw videos using lossless compression[C]. Brazilian Symposium on Multimedia and the Web. ACM, 2013: 173-176.

[34] Bartík M, Ubik S, Kubalik P. LZ4 compression algorithm on FPGA[C]. IEEE International Conference on Electronics, Circuits, and Systems. IEEE, 2016: 179-182.

[35] Ji H J, Lee S M, Gwon O S, et al. An FPGA Based Compression Accelerator for Forex Trading System[C]. International Conference on Information Technology, Singapore, 2016: 711-720.

[36] Lin M B, Lee J F, Jan G E. A Lossless Data Compression and Decompression Algorithm and Its Hardware Architecture[J]. IEEE Transactions on Very Large Scale Integration Systems, 2006, 14(9): 925 - 936.

[37] Kate D M. Hardware Implementation of the Huffman Encoder for Data Compression Using Altera DE2 Board[J]. International Journal of Advances in Engineering Sciences, 2012, 2(2): 11-15.

[38] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms, Third Edition[M]. The MIT Press, 2009, 253-253.

[39] Huffman D A. A method for the construction of minimum-redundancy codes[J]. Resonance, 2006, 40(2): 1098-1101.

[40] Riley J A. The Sardinas/Patterson and Levenshtein theorems[J]. Information and Control, 1967, 10(2): 120-136.

[41] IETF. RFC 3003, The audio/mpeg Media Type[S]. IETF. 2000.

[42] ITU. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Advanced Video Coding for Generic Audiovisual Services[S]. ITU-T Rec. 2003.

[43] L P Deutsch. RFC 1951, DEFLATE Compressed Data Format Specification version 1.3[S]. IETF. 2014.

[44] Jeon B. Application of dynamic Huffman coding to image sequence compression[J]. Proceedings of SPIE - The International Society for Optical Engineering, 1994, 2308: 1636-1647.

[45] 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, China, Beijing, 2010: 265-268.

[46] Akil M, Perroton L, Grandpierre T. FPGA-based architecture for hardware compression/decompression of wide format images[J]. Journal of Real-Time Image Processing, 2006, 1(2): 163-170.

[47] Lee T, Park J. Design and implementation of static Huffman encoding hardware using a parallel shifting algorithm[J]. IEEE Transactions on Nuclear Science, 2004, 51(5): 2073-2080.

[48] Tahghighi M, Mousavi M, Khadivi P. Hardware implementation of a novel adaptive version of Deflate compression algorithm[C], IEEE 18th Iranian Conference on Electrical Engineering (ICEE), Iran, 2010: 566-569.

[49] KC705 Evaluation Board for the Kintex-7 FPGA[EB/OL]. https://china.xilinx.com/support/documentation/boards_and_kits/kc705/ug810_KC705_Eval_Bd.pdf, 2016-07-08.

[50] Bus Master Performance Demonstration Reference Design for the Xilinx Endpoint PCI Express Solutions[EB/OL]. https://china.xilinx.com/support/documentation/application_notes/xapp1052.pdf, 2015-04-03.

[51] The Canterbury corpus[EB/OL]. http://corpus.canterbury.ac.nz/descriptions, 2016-12-14.

[52] Arnold R, Bell T. A Corpus for the Evaluation of Lossless Compression Algorithms[C]. 1997 Data Compression Conference, 1997: 201-210.

[53] 汤晓东. 基于 LZ77 算法的数据无损压缩的硬件设计[D]. 东南大学, 2013.