一种基于阵列配置加速比模型的无损压缩算法

徐金甫[①] 刘 露*[①] 李 伟[①②] 王周闯[①] 杨宇航[①]
[①](解放军信息工程大学 郑州 450001)
[②](复旦大学专用集成电路与系统国家重点实验室 上海 201203)

摘 要：针对现有压缩算法通过增加复杂度来降低压缩率，获得信息高效传输的问题。该文提出阵列配置加速比模型，证明低压缩率不一定能提高传输效率，并找到影响信息传输效率的因子，即解压模块吞吐率和数据块压缩率。将影响因子与配置信息特征结合，设计了一种新的无损压缩算法，并硬件实现了解压模块，吞吐率可达到 16.1 Gbps。采用 AES, A5-1 和 SM4 对无损压缩算法进行测试，然后与主流无损压缩算法 LZW, Huffman, LPAQ1 和 Arithmetic 对比。结果表明，整体压缩率相当，但该文压缩算法产生的数据块压缩率经过优化，不仅能满足加速需求，且具有高吞吐率的解压性能；该文无损压缩算法获得的配置加速比，比硬件吞吐率理想情况下的 LPAQl, Arithmetic, Huffman, LZW 算法分别高 8%, 9%, 10%, 22%左右。

关键词：阵列；配置加速比；无损压缩；吞吐率；压缩率
中图分类号：TP338.7 **文献标识码**：A **文章编号**：1009-5896(2018)06-1492-07
DOI: 10.11999/JEIT170900

A New Lossless Compression Algorithm Based on Array Configuration Speedup Model

XU Jinfu[①] LIU Lu[①] LI Wei[①②] WANG Zhouchuang[①] YANG Yuhang[①]
[①](*The PLA Information Engineering University*, *Zhengzhou* 450001, *China*)
[②](*State Key Laboratory of Fudan University Specific Integrated Circuit and System*, *Shanghai* 201203, *China*)

Abstract: In order to obtain efficient information transmission, the existing compression algorithms reduce the compression ratio by increasing complexity. In view of this problem, an array configuration speedup model is proposed in this paper. It is proved that low compression ratio may not improve the transmission efficiency and the factors, decompression module throughput and data block compression rate which affect the efficiency of information transmission, are found. Combined the influencing factors with the configuration information, a new lossless compression method is designed and the decompression hardware circuit is implemented, whose throughput can reach 16.1 Gbps. The lossless compression algorithm is tested using AES, A5-1 and SM4. Compared with the mainstream lossless compression algorithms LZW, Huffman, LPAQ1 and Arithmetic, the results show that the overall compression ratio is equivalent. However, the compression ratio of data block generated by the compression algorithm is optimized, which can not only meet the demand of acceleration, but also possesses high throughput decompression performance. The configuration speedup ratio obtained by lossless compression algorithm is about 8%, 9%, 10% and 22% higher than LPAQl, Arithmetic, Huffman, and LZW with ideal hardware throughput.

Key words: Array; Configuration speedup; Lossless compression; Throughput; Compression ratio

1 引言

如今，在图像处理、密码运算等许多领域，数据流驱动的逻辑阵列结构以高速且不失灵活的独特优势正逐渐被关注[1,2]。国内外学者对其结构[3]、运算单元[4]、互连方式[5]等都进行了深入研究。由于配置信息的传输效率直接影响阵列对数据的加速效果，因此，阵列配置信息的高效传输也是研究热点之一。

为减少配置信息传输时间，国内外学者主要研究配置信息压缩力度，提出了许多改进的无损压缩算法[6,7]。也有学者针对不同配置信息特征，设计出新的无损压缩算法[8–10]，以此获得更低的压缩率。

首先，改进或创新的算法能实现低压缩率，但从时间和资源消耗、算法完备性、硬件适用性等方

收稿日期：2017-09-22；改回日期：2018-04-13；网络出版：2018-04-18
*通信作者：刘露 liulu13213238773@163.com
基金项目：国家自然科学基金(61404175)
Foundation Item: The National Natural Science Foundation of China (61404175)

面分析，存在许多缺陷。文献[6]基于 LZW 算法提出了能释放大量无效内存、利用二分查表减少时间复杂度的方法，但并没有从解压模块硬件实现考虑，字典存储所占面积仍然很大。且该算法解压吞吐率小，导致信息传输缓慢。文献[9]提出了一种位重组标记编码方法，根据数据块中各符号所占比例自适应地选择编码方式。该方法需要统计各个模块中符号的概率，时间消耗大，编码效率低，且对编码方式选择所需的概率阈值没有科学的评判标准。文献[11]针对 PAQ[12]算法计算复杂度高、内存占用大等缺陷，提出了 LPAQl 算法，将预测模块改进并设计了有效的硬件结构。但其缺陷仍然明显，且研究压缩硬件结构意义不大。文献[13]针对电力系统参数的存储与传输问题，充分利用该参数规律，改进 Arithmetic 编码[14]，压缩效果明显，但其应用范围受到极大限制。其次，大多数压缩算法复杂度很高，只能软件实现[15]。设计时没有考虑解压吞吐率，也没有考虑其硬件实现时，与存储器、配置接口等相邻单元吞吐率匹配问题，导致不能实现高配置加速比[16]。

针对以上问题，本文分析影响配置信息传输效率因素，发现低压缩率并不一定能带来高的配置加速比，还与存储器、解压模块和配置接口的吞吐率有关。为获得最高加速比，本文提出阵列配置加速比模型。在该模型下，分析出了得到最高配置加速比的条件，找到了无损压缩算法的设计方向。然后结合阵列配置信息特征，指导压缩算法设计，这是本文压缩算法的设计思路和理论支撑。

2　逻辑阵列配置加速比模型

配置解压缩系统架构如图 1 所示。SRAM 用于存储压缩后的数据，解压模块用于解压压缩数据，配置接口用于接收解压后的数据，并分发数据至逻辑阵列内部各基本运算单元的配置寄存器。DMA 用于调动压缩数据通过总线传至解压模块。由于本文主要解决配置信息在硬件系统的传输问题，压缩过程一般用软件实现，因此只针对解压缩硬件系统建模讨论分析。

2.1 模型参数定义

设未压缩数据大小为 D bit，将其分为 k 块进行压缩。每块数据大小为 D_i，在 D bit 数据中占比例为 P_i，块压缩率为 σ_i。压缩后的数据大小为

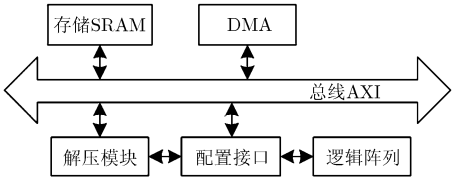

图 1 阵列配置解压缩系统架构

$$\sum_{i=1}^{k} D_i \cdot \sigma_i = D \cdot \sum_{i=1}^{k} P_i \cdot \sigma_i \tag{1}$$

设读存储 SRAM 吞吐率为 $\mathrm{ThrPut}_{\mathrm{SRAM}}$，解压模块吞吐率为 $\mathrm{ThrPut}_{\mathrm{Dec}}$ (输入端处理数据速度)，配置接口吞吐率为 $\mathrm{ThrPut}_{\mathrm{It}}$，配置压缩系统实际吞吐率为 $\mathrm{ThrPut}_{\mathrm{Cfg}}$，单位都为 bit/s。则吞吐率之间关系可表示为

$$\lambda_1 = \frac{\mathrm{ThrPut}_{\mathrm{It}}}{\mathrm{ThrPut}_{\mathrm{SRAM}}},\ \lambda_2 = \frac{\mathrm{ThrPut}_{\mathrm{It}}}{\mathrm{ThrPut}_{\mathrm{Dec}}} \tag{2}$$

设采用配置压缩系统传输大小为 D bit 数据所需时间为 T_{Cfg}，未采用配置压缩系统传输大小为 D bit 数据所需时间为 T_{NoDec}，配置加速比为 η。则 η 可表示为

$$\begin{aligned}\eta &= \frac{T_{\mathrm{NoDec}}}{T_{\mathrm{Cfg}}} \\ &= \begin{cases} \dfrac{D}{\mathrm{ThrPut}_{\mathrm{It}} \cdot T_{\mathrm{Cfg}}}, & \mathrm{ThrPut}_{\mathrm{It}} \le \mathrm{ThrPut}_{\mathrm{SRAM}} \\ \dfrac{D}{\mathrm{ThrPut}_{\mathrm{SRAM}} \cdot T_{\mathrm{Cfg}}}, & \mathrm{ThrPut}_{\mathrm{It}} > \mathrm{ThrPut}_{\mathrm{SRAM}} \end{cases}\end{aligned} \tag{3}$$

2.2 模型建立

文献[16]指出算法低压缩率不一定带来高的配置加速比，与读存储器速度和配置接口处理速度有关。将解压模块速度理想化，不影响传输效率，导致建模不完备，对自适应哈夫曼算法改进方法也不合理。

本文考虑解压模块吞吐率 $\mathrm{ThrPut}_{\mathrm{Dec}}$，对配置加速比 η 完备建模。通过建模分析 σ_i，$\mathrm{ThrPut}_{\mathrm{SRAM}}$，$\mathrm{ThrPut}_{\mathrm{Dec}}$ 和 $\mathrm{ThrPut}_{\mathrm{It}}$ 对 η 的实际影响权重，以在实际应用中通过合理选择 SRAM 或合理设计解压模块和配置接口，实现较高的配置加速比。

首先，结合图 1 对配置压缩系统的实际吞吐率 $\mathrm{ThrPut}_{\mathrm{Cfg}}$ 进行分析可知：

$$\mathrm{ThrPut}_{\mathrm{Cfg}} = \begin{cases} \mathrm{ThrPut}_{\mathrm{It}}, & \mathrm{ThrPut}_{\mathrm{Dec}} \ge \mathrm{ThrPut}_{\mathrm{SRAM}} \text{且} \mathrm{ThrPut}_{\mathrm{SRAM}}/\sigma_i \ge \mathrm{ThrPut}_{\mathrm{It}} \\ & \mathrm{ThrPut}_{\mathrm{Dec}} < \mathrm{ThrPut}_{\mathrm{SRAM}} \text{且} \mathrm{ThrPut}_{\mathrm{Dec}}/\sigma_i \ge \mathrm{ThrPut}_{\mathrm{It}} \quad \text{(a)} \\ \mathrm{ThrPut}_{\mathrm{SRAM}}/\sigma_i, & \mathrm{ThrPut}_{\mathrm{Dec}} \ge \mathrm{ThrPut}_{\mathrm{SRAM}} \text{且} \mathrm{ThrPut}_{\mathrm{SRAM}}/\sigma_i < \mathrm{ThrPut}_{\mathrm{It}} \quad \text{(b)} \\ \mathrm{ThrPut}_{\mathrm{Dec}}/\sigma_i, & \mathrm{ThrPut}_{\mathrm{Dec}} < \mathrm{ThrPut}_{\mathrm{SRAM}} \text{且} \mathrm{ThrPut}_{\mathrm{Dec}}/\sigma_i < \mathrm{ThrPut}_{\mathrm{It}} \quad \text{(c)} \end{cases} \tag{4}$$

从式(4)分析，当出现情况(a)时，接口吞吐率会限制解压后配置数据的传输速度；当出现情况(b)时，$\mathrm{ThrPut}_{\mathrm{It}}$，$\mathrm{ThrPut}_{\mathrm{Dec}}$ 大于 $\mathrm{ThrPut}_{\mathrm{SRAM}}$，因此 $\mathrm{ThrPut}_{\mathrm{SRAM}}$ 和 σ_i 会限制解压后配置数据的传输速度；当出现情况(c)时，$\mathrm{ThrPut}_{\mathrm{It}}$，$\mathrm{ThrPut}_{\mathrm{SRAM}}$ 大于 $\mathrm{ThrPut}_{\mathrm{Dec}}$，因此 $\mathrm{ThrPut}_{\mathrm{Dec}}$ 和 σ_i 会限制传输速度。

其次，对配置压缩系统的配置时间 T_{Cfg} 进行分析。在实际应用中，压缩算法对不同数据块产生的压缩率 σ_i 不会一直稳定大，$\mathrm{ThrPut}_{\mathrm{Cfg}}$ 保持在情况(b)或情况(c)。也不可能一直稳定小，$\mathrm{ThrPut}_{\mathrm{Cfg}}$ 保持在情况(a)。基于此，再结合式(1)，式(2)，对 T_{Cfg} 分情况讨论。

情况 1：当 $\mathrm{ThrPut}_{\mathrm{Dec}} \ge \mathrm{ThrPut}_{\mathrm{SRAM}}$ 时，T_{Cfg} 计算如式(5)所示。当 $\mathrm{ThrPut}_{\mathrm{SRAM}}/\sigma_i \ge \mathrm{ThrPut}_{\mathrm{It}}$ 时，$f(i)=1$；当 $\mathrm{ThrPut}_{\mathrm{SRAM}}/\sigma_i < \mathrm{ThrPut}_{\mathrm{It}}$ 时，$f(i)=0$。

$$\begin{aligned} T_{\mathrm{Cfg}}^{'} &= \sum_{i=1}^{k}\frac{D_i}{\mathrm{ThrPut}_{\mathrm{It}}}\cdot f(i) \\ &\quad + \sum_{i=1}^{k}\frac{D_i}{\mathrm{ThrPut}_{\mathrm{SRAM}}/\sigma_i}\cdot(1-f(i)) \\ &= \frac{D}{\mathrm{ThrPut}_{\mathrm{It}}}\cdot\left(\sum_{i=1}^{k}P_i\cdot f(i)+\lambda_1 \right. \\ &\quad \left. \cdot\sum_{i=1}^{k}P_i\cdot\sigma_i\cdot(1-f(i))\right) \end{aligned} \tag{5}$$

情况 2：当 $\mathrm{ThrPut}_{\mathrm{Dec}} < \mathrm{ThrPut}_{\mathrm{SRAM}}$ 时，T_{Cfg} 计算如式(6)所示。当 $\mathrm{ThrPut}_{\mathrm{Dec}}/\sigma_i < \mathrm{ThrPut}_{\mathrm{It}}$ 时，$g(i)=1$；当 $\mathrm{ThrPut}_{\mathrm{Dec}}/\sigma_i < \mathrm{ThrPut}_{\mathrm{It}}$ 时，$g(i)=1$。

$$\begin{aligned} T_{\mathrm{Cfg}}^{''} &= \sum_{i=1}^{k}\frac{D_i}{\mathrm{ThrPut}_{\mathrm{It}}}\cdot g(i) \\ &\quad + \sum_{i=1}^{k}\frac{D_i}{\mathrm{ThrPut}_{\mathrm{Dec}}/\sigma_i}\cdot(1-g(i)) \\ &= \frac{D}{\mathrm{ThrPut}_{\mathrm{It}}}\cdot\left(\sum_{i=1}^{k}P_i\cdot g(i)+\lambda_2 \right. \\ &\quad \left. \cdot\sum_{i=1}^{k}P_i\cdot\sigma_i\cdot(1-g(i))\right) \end{aligned} \tag{6}$$

将式(5)和式(6)整合可得

$$\begin{aligned} T_{\mathrm{Cfg}} &= T_{\mathrm{Cfg}}^{'}\cdot H + T_{\mathrm{Cfg}}^{''}\cdot(1-H), \\ H &= \begin{cases} 1, & \mathrm{ThrPut}_{\mathrm{Dec}} \ge \mathrm{ThrPut}_{\mathrm{SRAM}} \\ 0, & \mathrm{ThrPut}_{\mathrm{Dec}} < \mathrm{ThrPut}_{\mathrm{SRAM}} \end{cases} \end{aligned} \tag{7}$$

最后，对配置加速比 η 进行分析。结合式(3)、式(4)、式(5)、式(6)和式(7)可知，变量 T_{NoDec} 和 T_{Cfg} 都与 $\mathrm{ThrPut}_{\mathrm{SRAM}}$，$\mathrm{ThrPut}_{\mathrm{Dec}}$，$\mathrm{ThrPut}_{\mathrm{It}}$ 和 σ_i 有关。结合式(3)，η 可根据吞吐率间大小关系，分以下 5 种情况进行讨论。

情况 1：当 $\mathrm{ThrPut}_{\mathrm{Dec}} \ge \mathrm{ThrPut}_{\mathrm{SRAM}} \ge \mathrm{ThrPut}_{\mathrm{It}}$ 时，$H=t, f(i)=1$ 则

$$\eta = \frac{T_{\mathrm{NoDec}}}{T_{\mathrm{Cfg}}} = \frac{D}{\mathrm{ThrPut}_{\mathrm{It}}\cdot T_{\mathrm{Cfg}}^{'}} = 1 \tag{8}$$

情况 2：当 $\mathrm{ThrPut}_{\mathrm{SRAM}} \ge \mathrm{ThrPut}_{\mathrm{It}} > \mathrm{ThrPut}_{\mathrm{Dec}}$ 时，$H=0$，$g(i)$ 无法确定，则

$$\begin{aligned} \eta &= \frac{D}{\mathrm{ThrPut}_{\mathrm{It}}\cdot T_{\mathrm{Cfg}}^{''}} \\ &= \frac{1}{\sum_{i=1}^{k}P_i\cdot g(i)+\lambda_2\cdot\sum_{i=1}^{k}P_i\cdot\sigma_i\cdot(1-g(i))} \end{aligned} \tag{9}$$

情况 3：当 $\mathrm{ThrPut}_{\mathrm{SRAM}} > \mathrm{ThrPut}_{\mathrm{Dec}} \ge \mathrm{ThrPut}_{\mathrm{It}}$ 时，$H=1$，$g(i)=1$ 则

$$\eta = \frac{T_{\mathrm{NoDec}}}{T_{\mathrm{Cfg}}} = \frac{D}{\mathrm{ThrPut}_{\mathrm{It}}\cdot T_{\mathrm{Cfg}}^{''}} = 1 \tag{10}$$

情况 4：当 $\mathrm{ThrPut}_{\mathrm{SRAM}} \le \mathrm{ThrPut}_{\mathrm{Dec}} \le \mathrm{ThrPut}_{\mathrm{It}}$ 或 $\mathrm{ThrPut}_{\mathrm{SRAM}} < \mathrm{ThrPut}_{\mathrm{It}} \le \mathrm{ThrPut}_{\mathrm{Dec}}$ 时，$H=1$，$f(i)$ 无法确定，则

$$\begin{aligned} \eta &= \frac{D}{\mathrm{ThrPut}_{\mathrm{SRAM}}\cdot T_{\mathrm{Cfg}}^{'}} \\ &= \frac{1}{\frac{1}{\lambda_1}\cdot\sum_{i=1}^{k}P_i\cdot f(i)+\sum_{i=1}^{k}P_i\cdot\sigma_i\cdot(1-f(i))} \end{aligned} \tag{11}$$

情况 5：当 $\mathrm{ThrPut}_{\mathrm{Dec}} < \mathrm{ThrPut}_{\mathrm{SRAM}} < \mathrm{ThrPut}_{\mathrm{It}}$ 时，$H=0$，$g(i)$ 无法确定，则

$$\begin{aligned} \eta &= \frac{D}{\mathrm{ThrPut}_{\mathrm{SRAM}}\cdot T_{\mathrm{Cfg}}^{''}} \\ &= \frac{1}{\frac{1}{\lambda_1}\cdot\left(\sum_{i=1}^{k}P_i\cdot g(i)+\lambda_2\cdot\sum_{i=1}^{k}P_i\cdot\sigma_i\cdot(1-g(i))\right)} \end{aligned} \tag{12}$$

2.3 模型分析

从模型可知，压缩算法与 $\mathrm{ThrPut}_{\mathrm{It}}$，$\mathrm{ThrPut}_{\mathrm{SRAM}}$ 无关，可将其看作可配常量。与压缩算法有关且影响 η 的因素只有 $\mathrm{ThrPut}_{\mathrm{Dec}}$ 和 σ_i。首先，讨论 $\mathrm{ThrPut}_{\mathrm{Dec}}$ 对 η 的影响。从式(8)、式(10)和式(11)可知，满足表达式条件的 $\mathrm{ThrPut}_{\mathrm{Dec}}$ 对 η 无影响，从式(9)和式(12)可知，λ_1 增大，λ_2 减小，则 η 增大。因此在其他变量不变时，$\mathrm{ThrPut}_{\mathrm{Dec}}$ 增大会使 η 增大。且从 2.2 节 5 种情况的条件可知，$\mathrm{ThrPut}_{\mathrm{Dec}}$ 增大会使解压缩系统适应更多不同吞吐率的存储和配置接口。

其次，讨论 σ_i 对 η 的影响。从式(9)可知，当 σ_i

减小，但仍满足 $\mathrm{ThrPut}_{\mathrm{Dec}}/\sigma_i < \mathrm{ThrPut}_{\mathrm{It}}$，即 $\sigma_i > 1/\lambda_2$ 时，$\sum P_i \cdot g(i)$ 保持不变，$\lambda_2 \cdot \sum P_i \cdot \sigma_i \cdot (1-g(i))$ 减小，所以 η 增大。当 σ_i 减小，但满足 $\mathrm{ThrPut}_{\mathrm{Dec}}/\sigma_i \geq \mathrm{ThrPut}_{\mathrm{It}}$，即 $\sigma_i \leq 1/\lambda_2$ 时，$\sum P_i \cdot g(i)$ 会增大，设其增量 $\Delta P = \sum P_i$，$\lambda_2 \cdot \sum P_i \cdot \sigma_i \cdot (1-g(i))$ 会减小，设其减小量为 $\lambda_2 \cdot \sum P_i \cdot \sigma_i$，由于减少的 σ_i 都满足 $\sigma_i > 1/\lambda_2$，所以 $\lambda_2 \cdot \sum P_i \cdot \sigma_i > \sum P_i = \Delta P$，即减少值比增加值大，因此 η 增大。但当 σ_i 满足 $\sigma_i \leq 1/\lambda_2$ 后继续减小，$\lambda_2 \cdot P_i \cdot \sigma_i \cdot (1-g(i)) = 0$ 不变，因此 σ_i 继续减小对增大 η 无任何意义。可以用同样的方法证明，当 σ_i 增大时，η 减小。对式(11)、式(12)的分析方法与式(9)相似。

从以上分析可知，为获得更高的配置加速比 η，可增加无损压缩算法的解压模块吞吐率 $\mathrm{ThrPut}_{\mathrm{Dec}}$。块压缩率 σ_i 小于 $1/\lambda_1$ 或 $1/\lambda_2$ 后，继续减小 σ_i 对增大 η 没有任何帮助。此结论将指导无损压缩算法设计。

3　无损压缩算法设计

3.1 阵列配置信息特征

本文在一款粗粒度逻辑阵列上映射了 AES, SM4, A5-1 等算法，得到 3 种不同的配置信息。为不破坏粗粒度阵列单元结构对应的配置信息规律[17]，将其以 32 bit 为单位划分为块。根据块信息特征不同，将块信息分为 18 类，如表 1 所示。为验证分类合理性，对 3 种算法配置信息分别统计各类块信息所占比例，结果如表 2 所示。从统计结果可知，前 17 类特征占配置信息 95%左右，说明特征分类覆盖率高，分类方法合理。

3.2 无损压缩算法设计

基于加速比模型结论和阵列配置信息特征，本文无损压缩算法主要设计思想是分类压缩、优化压缩率 σ_i 和提高解压吞吐率 $\mathrm{ThrPut}_{\mathrm{Dec}}$，三者紧密相关。无损压缩算法伪代码描述见表 3。

在 ACA(D)算法中，可分 3 步描述。

第 1 步(1~8 行)：压缩率阈值配置。由式(2)计算 λ_1 和 λ_2。然后根据吞吐率 $\mathrm{ThrPut}_{\mathrm{It}}$，$\mathrm{ThrPut}_{\mathrm{SRAM}}$ 和 $\mathrm{ThrPut}_{\mathrm{Dec}}$ 大小关系判断归属情况。若属于情况 2 和情况 5，则 $S=1$，$1/\lambda_2$ 作为压缩率 σ_i 的阈值；若属于情况 4，则 $S=1$，$1/\lambda_1$ 作为压缩率 σ_i 的阈值。

第 2 步(9~18 行)：分类压缩。输入数据块 D_i，数据类型检测函数 Data_Type_Detection(·) 对其类型检测，子压缩函数 Subcompression(·) 进行 18 种不同方式的编码，检测结果 DTD 选择最合适压缩结果，并暂存到 Reg_i。

第 3 步(19~37)：包平均压缩率 τ_j 优化。组包时数据位宽 $\mathrm{lengh(pack}_j\mathrm{)}$ 小于 64 且 τ_j 小于压缩率阈值 σ，则将 Reg_i 数据读入包内；否则用函数 Optimization(·) 优化。具体优化过程：若 $\mathrm{lengh(pack}_j\mathrm{)}$ 大于 64 位或 τ_j 大于 σ 时，则不读入包内，继续扫描下一寄存器；当位宽总和小于 64 且 τ_j 大于 σ 时，则

表 1 数据块分类及编码方式

类型	包头	编码方式	编码长度	压缩率
全 0	0000/0001	包头	4	0.13
全 1	0010	包头	4	0.13
1 bit 为 1	0011/0100	包头+位置	9	0.28
1 bit 为 0	0101	包头+位置	9	0.28
2 bit 为 1	0110	包头+0+位置 1+位置 2	15	0.47
2 bit 为 0	0110	包头+1+位置 1+位置 2	15	0.47
1 个不是 0	0111/1000	包头+位置+数值	11	0.34
2 个不是 0	1001	包头+位置 1+数值 1+位置 2+数值 2	18	0.56
1 个不是 F	1010	包头+0+位置+数值	12	0.38
2 个不是 F	1010	包头+1+位置 1+数值 1+位置 2+数值 2	19	0.59
3 个不是 0	1011	包头+0+位置标志+数值 1+数值 2+数值 3	25	0.78
3 个不是 F	1011	包头+1+位置标志+数值 1+数值 2+数值 3	25	0.78
4 个不是 0	1100	包头+0+位置标志+数值 1+数值 2+数值 3+数值 4	29	0.91
4 个不是 F	1100	包头+1+位置标志+数值 1+数值 2+数值 3+数值 4	29	0.91
5 个不是 0	1101	包头+0+位置标志+数值 1+数值 2+数值 3+数值 4+数值 5	33	1.03
5 个不是 F	1101	包头+1+位置标志+数值 1+数值 2+数值 3+数值 4+数值 5	33	1.03
XYXYXYXY	1110	包头+XY	12	0.38
其他	1111	包头+原数据	36	1.12

表 2 各类数据块比例(%)

块数据类型	AES	SM4	A5-1
全 0	30.1	37.7	71.4
全 1	1.3	2.7	3.7
1 bit 为 1	8.2	5.8	4.6
1 bit 为 0	3.4	0.7	0.4
2 bit 为 1	0.8	0.4	0.7
2 bit 为 0	0.6	0.7	0.5
1 个不是 0	8.9	3.5	2.7
2 个不是 0	7.8	1.7	3.5
1 个不是 F	0.5	0.2	0.9
2 个不是 F	1.2	0.4	0.6
3 个不是 0	1.6	0.8	0.7
3 个不是 F	1.3	0.6	0.4
4 个不是 0	2.3	0.7	0.9
4 个不是 F	0.4	0.4	0.3
5 个不是 0	0.6	0.3	0.5
5 个不是 F	0.3	0.6	1.7
XYXYXYXY	27.9	41.3	2.3
其他	2.8	1.5	4.2

将包内中等大小压缩率的块数据“抛弃”，将未读入包内且包头为 0000, 0011, 0111 的数据分别进行压缩率和位宽判断，在使包内数据同时满足位宽和压缩率要求的数据中，选择最接近 σ 的数据，将其读入包内且变换包头，并在包头后添加 3 bit 位置标记位，以标记此数据在 8 个数据块中的位置。若没有使包内数据同时满足位宽和压缩率要求的数据，则将包内具有更大压缩率的块数据“抛弃”，重新压缩率和位宽判断。优化完成后，将 pack_j 输出，并用函数 Fill(·) 填充 8 个寄存器。

如表 1 所示。数据块特征不同对应的编码方式也不相同，每类数据块都分配 4 bit 包头用于解压模块识别编码方式。编码思想是只传递非 0 值信息，“位置”指非 0 值在 32 位数据块内的位置。全 0 和全 1，直接用包头编码。此种编码方式不仅复杂度小、压缩力度强，还包括以下优点：(1)可以提高吞吐率 $\text{ThrPut}_{\text{Dec}}$。解压模块解压非常容易硬件实现，只需要简单查表和数据拼接，1 个时钟完成解压且硬件关键路径很小。(2)编码长度固定，压缩率能预知，有利于不同程度地优化 τ_j。由于全 0、只有 1 个 1 和 1 个不是 0 这 3 类数据所占比例较大，且编码长度小，非常适合用于优化包平均压缩率。

3.3 解压算法设计与硬件实现

由模型结论可知，解压模块必须实现高吞吐率 $\text{ThrPut}_{\text{Dec}}$，因此，要求处理一次数据位宽要大，所耗时钟周期要少且关键路径要短。基于此，解压模块硬件设计如图 2 所示。解压模块包括拆包单元、16 个子解压单元、16 个寄存器和数据块重排单元。解压是压缩的逆过程，将 64 位压缩包拆包，提取包头及相应的数据，分别输入到子解压单元，并行解压。同时拆包单元将数据块位置标记传到数据块重排单元，指导数据重排。然后将前 4 个寄存器作为解压模块 1 个周期的解压结果输出。

表 3 无损压缩算法伪代码

算法 阵列压缩 ACA(D) //对 D bit 数据压缩

(1)if $(\text{ThrPut}_{\text{SRAM}} \geq \text{ThrPut}_{\text{It}} > \text{ThrPut}_{\text{Dec}}$
or $\text{ThrPut}_{\text{Dec}} < \text{ThrPut}_{\text{SRAM}} < \text{ThrPut}_{\text{It}})$ then
(2) $S = 1$;
(3) $\sigma = 1/\lambda_2$;
(4)end if
(5)if ($\text{ThrPut}_{\text{SRAM}} < \text{ThrPut}_{\text{It}} \leq \text{ThrPut}_{\text{Dec}}$
or $\text{ThrPut}_{\text{SRAM}} \leq \text{ThrPut}_{\text{Dec}} \leq \text{ThrPut}_{\text{It}})$ then
(6) $S = 0$;
(7) $\sigma = 1/\lambda_1$;
(8)end if
(9)for all $i \in \{1,2,\cdots,k\}$ do
(10) $\text{DTD} \leftarrow \text{Data\_Type\_Detection}(D_i)$;
(11) switch(DTD){
(12) case1: $\text{D\_Cpr}_i \leftarrow \text{Subcompression1}(D_i)$;
(13) case2: $\text{D\_Cpr}_i \leftarrow \text{Subcompression2}(D_i)$;
(14) ……
(15) case18: $\text{D\_Cpr}_i \leftarrow \text{Subcompression18}(D_i)$;
(16) default: break; }
(17) $\text{Reg}_i \leftarrow \text{D\_Cpr}_i$;
(18) $\sigma_i \leftarrow \text{lengh}(\text{D\_Cpr}_i)/32$;
(19) if ($\text{Reg}_8 \neq \text{null}$) then
(20) $i \leftarrow 1$;
(21) for all $i \leq 8$ do
(22) if $\left(\tau_j \leq \sigma \ \&\ \&\text{lengh}\left(\text{pack}_j\right) \leq 64\right)$ then
(23) $\tau_j \leftarrow \tau_j + \sigma_i$;
(24) $\text{temp}_j \leftarrow \text{pack}_j$;
(25) $\text{pack}_j \leftarrow \text{pack}_j + \text{Reg}_i$;
(26) $\text{Reg}_i \leftarrow \text{null}$;
(27) if $(i == 8)$ then
(28) Fill(·);
(29) end if
(30) else
(31) ptimization(pack_j, Reg_i, τ_j, σ, temp_j);
(32) Print pack_j;
(33) Fill(·);
(34) end if
(35) end for
(36) end if
(37) return(pack);

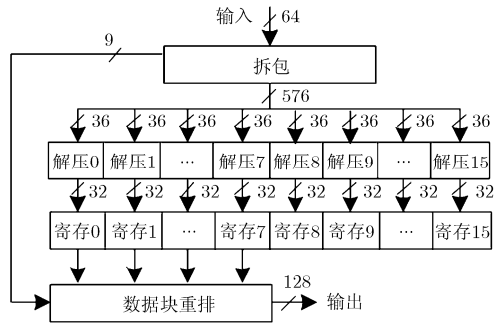

图 2 解压模块硬件结构

4 功能验证与性能分析

用 Java 语言实现本文无损压缩算法，LZW, Huffman, LPAQ1 和 Arithmetic 压缩程序，分别对粗粒度密码逻辑阵列 3 种算法(AES, A5-1, SM4)的配置数据进行压缩率测试，测试结果如表 4 所示。可以看出，本文设计的无损压缩算法压缩 3 个配置信息文件得到的整体压缩性能比 LAW, LPAQl 略低，比 Huffman, Arithmetic 高。

用硬件描述语言 Verilog 实现了解压模块，并在 Xilinx ISE 上进行 FPGA 逻辑综合。最高频率达到 257.6 MHz，比 LZW[18]解压频率更高。解压模块吞吐率达到 16.1 Gbps，远超现有存储器吞吐率和配置接口吞吐率，适用性强。将 A5-1 测试文件在不同压缩率阈值条件下用本文算法、LZW 算法和 Huffman 算法进行压缩，分别统计数据块压缩率和满足阈值的压缩率比例，结合式(9)、式(11)和式(12)，计算相应的配置加速比 η。如表 5 所示，可以看出本文压缩算法配置加速比 η 比 LPAQl, Arithmetic, Huffman, LZW 算法分别高 8%, 9%, 10%, 22%左右，即配置信息传输效率高。且表中其它 4 种算法解压吞吐率是理想化的情况，在实际硬件设计与实现中，受自身局限性，其吞吐率很难满足表中压缩率阈值设定的参数。

5 结束语

首先，本文把现有压缩算法一味追求低压缩率的现象和现有阵列配置信息传输效率不高的问题相结合，通过对配置加速比建模说明了低压缩率不一定能提高加速比，并进一步说明了获得高加速比的方法，即解压模块吞吐率要够大，更多数据块的压缩率接近并小于阈值。

其次，基于配置加速比模型结论，针对粗粒度密码逻辑阵列配置信息特征，设计了无损压缩算法，并用 3 种算法的配置数据进行压缩验证，与算法 LAW, Huffman, LPAQl 和 Arithmetic 进行压缩率对比。对比结果表明，本文无损压缩算法整体压缩率比 LAW, LPAQl 略低，比 Huffman, Arithmetic 高，但由于更多数据块压缩率接近并小于阈值，从而获得了更高的配置加速比。另外，可对本文设计的压缩算法进一步研究，以满足更多特征的配置信息压缩与传输。

表 4 算法压缩率测试结果(%)

配置文件	大小(kb)	LZW[18]	Huffman[19]	LPAQl[11]	Arithmetic[13]	本文
A5-1.input	22.1	33.7	30.8	28.3	29.4	27.1
AES.input	28.6	28.8	47.6	30.1	38.3	36.2
SM4.input	19.3	25.1	41.8	28.8	32.5	31.4

表 5 配置加速比测试结果

压缩率阈值参数		压缩率阈值	满足阈值压缩率比例(%)					配置加速比 η				
λ_1	λ_2		LZW	Huffman	LPAQl	Arithmetic	本文	LZW	Huffman	LPAQl	Arithmetic	本文
0.50	2.17	0.46	72.87	74.86	78.7	77.6	92.47	0.96	0.96	0.98	0.97	1.00
2.50	0.80	0.40	62.36	68.32	75.3	73.8	83.66	2.28	2.48	2.51	2.49	2.70
1.50	2.80	0.36	54.97	63.35	72.9	71.4	76.28	1.29	1.43	1.46	1.45	1.58

参 考 文 献

[1] 张海斌, 朱苏磊, 徐明亮. 基于可编程逻辑阵列的索贝尔边缘检测算法的两种实现方案[J]. 上海师范大学学报(自然科学版), 2017, 46(2): 247–253. doi: 10.3969/J.ISSN.1000-5137.2017.02.013.
ZHANG Haibin, ZHU Sulei, and XU Mingliang. Two kinds of implementations of sobel edge detection algorithm based on field programmable gate array[J]. *Journal of Shanghai Normal University* (*Natural Science Edition*), 2017, 46(2): 247–253. doi: 10.3969/J.ISSN.1000-5137.2017.02.013.

[2] 冯晓, 李伟, 戴紫彬, 等. 面向分组密码的可重构异构多核并行处理架构[J]. 电子学报, 2017, 45(6): 1311–1320. doi: 10.3969/j.issn.0372-2112.2017.06.005.

FENG Xiao, LI Wei, DAI Zibin, *et al*. Reconfigurable asymmetrical multi-core architecture for block cipher[J]. *Acta Electronica Sinica*, 2017, 45(6): 1311–1320. doi: 10.3969/j.issn.0372-2112.2017.06.005.

[3] 肖艺, 鲁华祥, 陈刚, 等. 基于仿生学的多层自适应容错重构阵列研究[J]. 仪器仪表学报, 2016, 37(2): 437–445. doi: 10.3969/j.issn.0254-3087.2016.02.026.
XIAO Yi, LU Huaxiang, CHEN Gang, *et al*. Bio-inspired method to develop the multi-layer and self-adaptive reconfigurable array[J]. *Journal of Instrumentation*, 2016, 37(2): 437–445. doi: 10.3969/j.issn.0254-3087.2016.02.026.

[4] DHAOUI F, MCCOLLUM J, HAWLEY F, *et al*. Non-volatile programmable memory cell and array for programmable logic array[P]. US, US7838944, 2010.

[5] 陈锐, 杨海钢, 王飞, 等. 基于自路由互连网络的粗粒度可重构阵列结构[J]. 电子与信息学报, 2014, 36(9): 2251–2257. doi: 10.3724/SP.J.1146.2013.01646.
CHEN Rui, YANG Haigang, WANG Fei, *et al*. Coarse-grained reconfigurable array based on self-routing interconnection network[J]. *Journal of Electronics & Information Technology*, 2014, 36(9): 2251–2257. doi: 10.3724/SP.J.1146.2013.01646.

[6] 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. doi: 10.3969/j.issn.1673-629X.2009.04.036.
XU Xia, MA Guangsi, and YU Tao. Research and improvement on LZW lossless compression algorithm[J]. *Computer Technology and Development*, 2009, 19(4): 125–127. doi: 10.3969/j.issn.1673-629X.2009.04.036.

[7] 鄢海舟, 胥布工, 石东江, 等. 无损压缩算法 LZW 前缀编码优化及应用[J]. 计算机工程, 2017, 43(3): 299–303. doi: 10.3969/j.issn.1000-3428.2017.03.050.
YAN Haizhou, XU Bugong, SHI Dongjiang, *et al*. Prefix encoding optimization and application of lossless compression algorithm LZW[J]. *Computer Engineering*, 2017, 43(3): 299–303. doi: 10.3969/j.issn.1000-3428.2017.03.050.

[8] 徐勇, 李珂, 冯国平, 等. 一种 FPGA 在轨重构配置数据压缩算法[J]. 航天器工程, 2015, 24(6): 75–78. doi: 10.3969/j.issn.1673-8748.2015.06.013.
XU Yong, LI Ke, FENG Guoping, *et al*. Configuration data compression algorithm for FPGA on-orbit reconfiguration[J]. *Spacecraft Engineering*, 2015, 24(6): 75–78. doi: 10.3969/j.issn.1673-8748.2015.06.013.

[9] 蔡明, 乔文孝, 鞠晓东, 等. 一种新的数据无损压缩编码方法[J]. 电子与信息学报, 2014, 36(4): 1008–1012. doi: 10.3724/SP.J.1146.2013.00863.
CAI Ming, QIAO Wenxiao, JU Xiaodong, *et al*. A new coding method for lossless data compression[J]. *Journal of Electronics & Information Technology*, 2014, 36(4): 1008–1012. doi: 10.3724/SP.J.1146.2013.00863.

[10] 杨国为, 涂序彦, 庞杰. 基于虚拟信源的无损数据压缩方法研究[J]. 电子学报, 2003, 31(5): 728–731. doi: 10.3321/j.issn:0372-2112.2003.05.023.
YANG Guowei, TU Xuyan, and PANG Jie. The research of lossless data compression based on a virtual information source[J]. *Acta Electronica Sinica*, 2003, 31(5): 728–731. doi: 10.3321/j.issn:0372-2112.2003.05.023.

[11] WANG Zhisheng, LIN Jun, and WANG Zhongfeng. An efficient hardware architecture for lossless data compression in data center[C]. IEEE International Workshop on Signal Processing Systems. Orlando, USA, 2016: 159–164.

[12] MAHONEY M. Adaptive weighing of context models for lossless data compression[J]. *Florida Institute of Technology*, 2005, 16: 1–6.

[13] SARKAR S J, SARKAR N K, DUTTA T, *et al*. Arithmatic coding based approach for power system parameter data compression[J]. *Indonesian Journal of Electrical Engineering and Computer Science*, 2016, 2(2): 268–274.

[14] HASHEMPOUR H and LOMBARDI F. Application of arithmetic coding to compression of VLSI test data[J]. *IEEE Transactions on Computers*, 2005, 54(9): 1166–1177.

[15] 高放, 孙长建, 邵庆龙, 等. 基于 K-均值聚类和传统递归最小二乘法的高光谱图像无损压缩[J]. 电子与信息学报, 2016, 38(11): 2709–2714. doi: 10.11999/JEIT151439.
GAO Fang, SUN Changjian, SHAO Qinglong, *et al*. Lossless compression of hyperspectral images based on *k*-mean clustering and conventional recursive least-squares[J]. *Journal of Electronics & Information Technology*, 2016, 38(11): 2709–2714. doi: 10.11999/JEIT151439.

[16] 杨鹏. 可重构片上系统配置数据压缩算法研究[D]. [硕士论文], 湖南大学, 2010: 20–29.
YANG Peng. Research on configuration data compression algorithm for reconfigurable system-on-chip[D]. [Master dissertation], Hunan University, 2010: 20–29.

[17] 古海云, 李丽, 许居衍, 等. 一种 Virtex 系列 FPGA 配置数据无损压缩算法[J]. 计算机研究与发展, 2006, 43(5): 940–945.
GU Haiyun, LI Li, XU Juyan, *et al*. Lossless configuration bitstream compression for Virtex FPGA[J]. *Computer Research and Development*, 2006, 43(5): 940–945.

[18] 刘仕东. 一种通用 FPGA 配置数据流压缩与解压缩系统的研究[D]. [硕士论文], 哈尔滨工业大学, 2011: 49–50.
LIU Shidong. Research for compression and decompression system of a generic FPGA configuration data stream[D]. [Master dissertation], Harbin Institute of Technology, 2011: 49–50.

[19] 王防修, 刘春红. 一种哈夫曼编码的改进算法[J]. 武汉轻工大学学报, 2016, 35(1): 88–91.
WANG Fangxiu and LIU Chunhong. An improved algorithm of huffman encoding[J]. *Journal of Wuhan Polytechnic University*, 2016, 35(1): 88–91.

徐金甫： 男，1965 年生，教授，硕士生导师，研究方向为专用集成电路设计.

刘 露： 男，1992 年生，硕士生，研究方向为专用集成电路设计.

李 伟： 男，1983 年生，副教授，硕士生导师，研究方向为安全专用芯片设计.

王周闯： 男，1992 年生，博士，研究方向为网络安全处理器设计.

杨宇航： 男，1991 年生，硕士生，研究方向为专用集成电路设计.