

引用格式:程战涛,梁峰,张国和. 面向 RISC-V 指令集架构处理器的代码压缩技术[J]. 微电子学与计算机, 2021, 38(6): 13-19. [CHENG Z T, LIANG F, ZHANG G H. Code compression technology for processors based on RISC-V instruction set architecture[J]. Microelectronics & Computer, 2021, 38(6): 13-19.]

面向 RISC-V 指令集架构处理器的代码压缩技术

程战涛, 梁 峰, 张国和

(西安交通大学 微电子学院, 陕西 西安 710049)

摘 要: 针对嵌入式系统处理器代码量指数式增长带来的设计与验证难题, 提出一种适用于 RISC-V 指令集架构处理器的 Bitmask 代码压缩技术. 结合 RISC-V 指令集特点, 设计了混合编码与分离字典相结合的 Bitmask 代码压缩算法; 在不影响处理器结构和 Cache 工作机制的基础上, 设计精简的硬件解压缩单元, 减小了嵌入式系统处理器所需的程序内存空间. 采用面向 RISC-V 指令集的混合编码压缩指令格式, 减小原始指令码的码字长度, 改善了代码压缩率; 采用分离的两个字典结构, 在不影响代码压缩率的前提下, 减小了硬件解压缩延迟. 结果表明, 在 RISC-V 指令集架构上不增加过多硬件开销的情况下, 代码压缩率平均为 61.1%, 大大减小了处理器所需的程序内存空间.

关键词: 嵌入式系统处理器; RISC-V 指令集; 代码压缩技术; 硬件解压缩单元

中图分类号: TP332

文献标识码: A

文章编号: 1000-7180(2021)06-0013-07

Code compression technology for processors based on RISC-V instruction set architecture

CHENG Zhan-tao, LIANG Feng, ZHANG Guo-he

(School of Microelectronics, Xi'an Jiaotong University, Xi'an 710049, Shaanxi China)

Abstract: Aiming at design and verification problems caused by the exponential growth of the code size in the embedded system processors, the Bitmask-based code compression technology for processors based on RISC-V instruction set architecture is presented. Based on the features of RISC-V instruction set, a Bitmask-based code compression algorithm combined with mixed encoding and separate dictionary is designed. Without affecting the processor structure and Cache working mechanism, a simple and efficient hardware decompression unit is designed to reduce the program memory space required by the embedded system processor. Mixed encoding format for RISC-V instruction set is used to reduce the length of the original instruction, which improves the code compression rate. Besides, two separated dictionaries are used to obtain a small hardware decompression delay without affecting the code compression rate. The experimental results show that the code compression ratio is 61.1% on average without much hardware overhead on the RISC-V instruction set architecture, which greatly reduces the program memory space required by the processor.

Key words: embedded system processor; RISC-V instruction set; code compression technology; decompression hardware unit

收稿日期: 2020-09-08; 修回日期: 2020-10-25

基金项目: 国家自然科学基金资助项目(61474093); 陕西省自然科学基金资助项目(2020JM-006); 航空科学基金项目(20184370012)

1 引言

在当今的嵌入式系统处理器中,程序存储器占据了处理器芯片很大一部分面积,为了缩减代码尺寸,减小程序存储空间, Wolfe 和 Chanin 在 1992 年提出了精简指令集计算机代码压缩 (Compressed Code RISC Processor, CCRP) 的概念^[1],并将代码压缩技术应用在精简指令集计算机 (RISC) 架构处理器上. 在 CCRP 中使用基于字典的代码压缩 (Dictionary-based Code Compression, DCC) 算法^[2],可以达到高效的压缩率,拥有精简的解压缩硬件,并提供更高的解压缩带宽. 基于 DCC 字典编码的代码压缩技术,文献^[3]提出了 BitMask 代码压缩 (BitMask-based Code Compression, BCC) 算法, BCC 是一种非完全的字典压缩,后面的研究多基于此. 文献^[4]针对 FPGA 硬件架构,使用 BCC 编码、游程编码并结合改进的字典构建方法,设计了适用于 FPGA 架构的代码压缩系统. Wang 和 Lin 将代码压缩技术中的各个方法组合成一种新的算法^[5-6],提高了代码压缩性能. 通过文献可以发现,代码压缩技术的研究集中在两个方向:1) 将现有的代码压缩方法应用于各种处理器架构进行优化;2) 结合多种代码压缩方法提高压缩性能.

RISC-V 指令集是加利福尼亚大学伯克利分校设计的基于精简指令集计算 (RISC) 原理的开源指令集架构^[7]. RISC-V 指令集自发布以来,受到了很多知名公司和研究机构的关注,市场上出现了很多受用户欢迎的 RISC-V 指令集处理器,如兆易创新的 GD32VF103 系列微控制单元,阿里巴巴旗下半导体公司平头哥的玄铁 910 等. 尚未见专门面向 RISC-V 指令集架构处理器的代码压缩技术.

本文从兼顾代码压缩率和硬件复杂度方面出发,采用 BCC 代码压缩技术,研究一种面向 RISC-V 指令集架构处理器的代码压缩技术,用以减小处理器所需的程序内存空间.

2 嵌入式系统代码压缩技术分析

2.1 BCC 代码压缩算法分析

基于字典的代码压缩算法是将重复出现的指令码放在字典中,用字典的索引号压缩原始的指令码,只要字典与压缩后指令码的存储空间小于原始指令码的存储空间,就能达到压缩代码的目的. 在利用 DCC 算法压缩代码时,如果压缩后的代码全部由字典索引号所组成,称之为完全的字典压缩,反之,则称之为非完全的字典压缩.

为了实现完全的字典压缩,假设 n 表示指令数目, d 表示字典索引数目, w 表示每条指令的位宽 (bit), 则原始指令码的存储空间为 mw , 字典的存储空间为 dw , 字典索引的存储空间为 $n[\log_2 d]$, 完全的字典压缩需要满足下面的公式:

$$mw > n[\log_2 d] + dw \tag{1}$$

完全的字典压缩要实现很好的代码压缩率,需要构造很大的字典,然而过大的字典会给系统带来额外的软硬件开销. 非完全的字典压缩引入掩码位等标识,不用构造很大的字典就可以得到很好的代码压缩率. 假设这些标识的平均长度为 s , 则压缩后指令码的存储空间为 $n([\log_2 d] + s)$, 非完全的字典压缩需要满足下面的公式:

$$mw > n([\log_2 d] + s) + dw \tag{2}$$

BCC^[3] 代码压缩算法是一种非完全的字典压缩,是在压缩的指令码中增加了掩码位等标记位. 图 1 展示了一个 BCC 代码压缩过程案例. 编码时根据原始指令码出现的频率构建字典查找表,压缩后的指令码中有两个标记位,一个用于标识指令是否被压缩,另一个用于标识指令是仅由字典压缩还是由 BitMask 方法压缩. 如果指令是通过 BitMask 方法压缩的,则压缩的指令码必须包含 BitMask 值及其位置;如果指令码通过字典压缩的,则被压缩成指向字典的索引. 这种基于 BCC 的代码压缩方法有很好的压缩性能,而且能使用精简的硬件在中央处理器 (CPU) 进行解压缩.

字典查找表:		
	序号	内容
	0	00000000
	1	01011101
指令码压缩结果:		
指令类型	待压缩指令码	压缩指令码
A	00000000	0 0 0
A	00000000	0 0 0
A	00000000	0 0 0
B	01010101	1 01010101
C	01011101	0 0 1
C	01011101	0 0 1
C	01011101	0 0 1
D	01010111	1 01010111
E	11000011	1 11000011
F	00001100	1 00001100
F	00001100	1 00001100
G	11000000	0 1 00 11 0
G	11000000	0 1 00 11 0

图 1 BitMask 压缩算法示例

2.2 代码压缩与解压缩的过程

代码压缩本质上是无损数据压缩,压缩的对象是二进制指令码。代码压缩通过宿主机离线实现,解压缩通过在嵌入式系统处理器中增加额外的硬件实时解压缩实现。因此,将 CCRP 应用于嵌入式系统处理器时,采用合适的代码压缩算法来减小代码尺寸并使用精简的解压缩硬件至关重要。

主流的嵌入式软件开发方式是利用高级语言实现算法,再利用编译器将高级语言汇编得到机器码。机器码分为代码段和数据段。图 2 展示了宿主机的代码压缩过程。代码压缩过程中利用代码压缩单元只压缩代码段,不对数据段进行压缩,因为数据段在程序执行过程中需要动态的改写,这对系统的性能和功耗开销影响都很大。

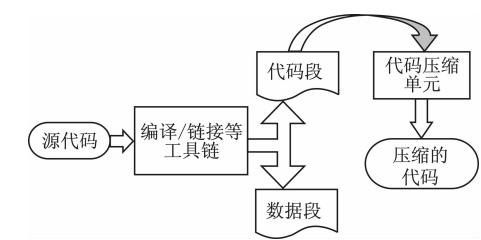


图 2 代码压缩过程

图 3 展示了中央处理器(CPU)的解压缩过程。CPU 的程序存储器存放的是压缩的指令,CPU 的指令缓存存放的是解压后的指令,CPU 的数据存储器存放的是原始的数据段数据。代码解压缩单元是程序存储器和指令缓存器之间的桥梁,其解压缩速度能够满足 CPU 指令访存所需的带宽要求。CPU 执行按照程序计数器(PC)的指示进行寻址,当程序计数器产生的目标地址不在指令缓存中,目标指令所在的基本块要从指令存储器中解压到指令缓存,基本块的大小是缓存行大小。

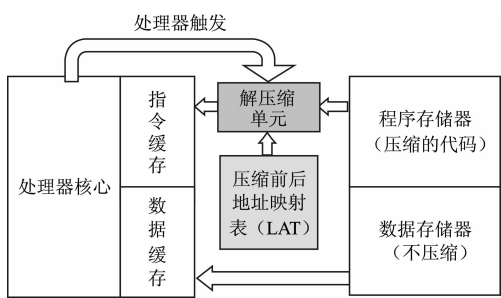


图 3 CPU 的解压缩过程

由于压缩后的指令码是不等长的,这样就可能导致压缩后的基本块起始地址不是字节对齐的。字节不对齐的基本块可以在宿主主机上进行比特填充,

确保基本块可以按字节寻址^[8]。另外,由于压缩后指令的地址也发生了变化,指令所在的压缩程序地址与指令所在的原始程序地址可以利用高效的行地址表(LAT)进行映射^[1]。这样在不改变原来 CPU 系统结构和 Cache 工作机制的基础上^[9],增加少量的硬件资源,就可以达到代码压缩的目的。

3 面向 RISC-V 指令集的代码压缩技术

3.1 代码压缩算法

3.1.1 MESDBCC-RV 算法

代码压缩率 CR 是衡量代码压缩算法性能的关键指标之一,假设 S_1 为字典查找表的空间大小, S_2 为压缩指令码的空间大小, S_3 为原始指令码的空间大小,则代码压缩率 CR 可以表示为:

$$CR = \frac{S_1 + S_2}{S_3}$$
 (3)

根据 BCC 代码压缩原理和非完全的字典压缩公式,本文提出面向 RISC-V 指令集的混合编码与分离字典相结合的代码压缩技术(Mixed Encoding and Separated Dictionary BitMask Code Compression for RISC-V, MESDBCC-RV)。表 1 展示了 MESDBCC-RV 算法。结合 RISC-V 指令集特征,算法使用位置编码和字典索引组成的压缩指令码,编码与字典中仅有少量几位相异的原始指令码,以减小压缩指令码的码字长度;同时,算法使用固定掩码或滑动掩码与 BitMask 编码相结合的方法,以提高原始指令对字典指令的命中率;算法还使用完全分离的两个大小字典,在保证代码压缩率的前提下,减小硬件解压缩延迟;另外,算法使用基于频率的字典选择(FDS)算法,不用构建过大的字典就可以使更多指令使用字典中的高频指令进行压缩,提高了指令对字典的匹配度。

表 1 MESDBCC-RV 算法

算法: MESDBCC-RV	
输入:	32 位的二进制指令码
输出:	压缩的二进制指令码 字典查找表
算法过程:	
开始:	
步骤 1: 计算所有输入的二进制指令码的频率分布;	
步骤 2: 基于步骤 1 计算二进制指令码的频率分布直方图,将高频的指令放在小字典中,将相对低频的指令放在大字典中;	
步骤 3: 读取一条二进制指令码,基于字典匹配原则优先选择码字长度短的编码格式压缩原始的二进制指令码;	
步骤 4: 重复步骤 3,直到完成对所有的二进制指令码压缩;	
步骤 5: 计算代码压缩率 CR。	
结束	

3.1.2 MESDBCC-RV 算法的字典构建

字典的构建方法在很大程度上影响了代码压缩率. 本文基于 RISC-V 指令集使用基准测试程序进行了指令频率分布的统计实验, 实验发现部分 RISC-V 指令始终具有极高的执行频率. 另外, RISC-V 指令集指令规模很小, 因此, 针对小规模 RISC-V 指令集, 不用字典更新, 使用高频指令构建字典, 就能使更多的指令匹配到字典中的指令, 从而使代码压缩效率更高、硬件复杂度更小.

RISC-V 的非同寻常之处在于它是模块化的. RV32I 指令模块是 RISC-V 指令集的基础模块, RV32IMFD 指令集模块是把乘法指令模块、单精度浮点指令模块、双精度浮点指令模块添加到 RV32I 指令模块中形成的. 因此, 结合 RISC-V 指令集模块化的特点, 可以形成不同模块的若干字典查找表. 此外, 较大的程序通常需要较大的字典查找表, 这会造成字典索引码长度过长. 为了克服这个问题, 可以把字典查找表分成若干不同大小的字典查找表, 不同频率的指令使用不同的字典索引码长度进行压缩, 以缩短压缩指令码的长度.

使用较大的字典进行代码压缩可以达到更好的压缩效率, 但不可避免地增加解压缩硬件的字典访问延迟. 本文基于 RV32I 指令, 从兼顾代码压缩效率和解压缩硬件复杂度的角度出发, 构建两个分离的字典, 小字典存储 8 条高频指令, 大字典存储 16 条次高频指令, 通过重复实验发现, 这样的设计可以使代码压缩效率更高、字典访问延迟更小.

3.1.3 MESDBCC-RV 算法的编码格式

编码格式是指经过宿主机压缩后的机器指令码格式. 字典匹配原则是原始指令码只能使用一种压缩指令码格式、并且与字典中唯一的指令码相对应、保证压缩后的指令码码字长度最短. 压缩过程是在字典中进行遍历, 匹配原始指令码. 遍历时如果该指令码与字典中的某条指令码完全匹配, 则将字典的索引进行编码; 如果有若干位不匹配, 则使用混合编码格式中的一种进行压缩, 选择的编码格式使压缩后的指令码字长度尽可能短; 如果按照匹配原则, 发现该指令码与字典中的所有指令码都不匹配, 则该条指令码不进行压缩.

处理器的指令集架构是混合编码格式制定的基础, 结合 RISC-V 指令集的特点, 制定以下压缩指令的编码格式:

① RISC-V 指令集非常简洁. 鉴于成本对设计复杂度的要求, 简单的高频指令通常相比复杂的低

频指令容易被调用. 因此, 在 MESDBCC-RV 编码格式中, 将这些容易被重复调用的高频指令存储在字典中, 使用字典的索引直接编码, 编码格式如图 5 的 C-type 类型. 另外, 针对有一位或者连续两位与字典中的指令码相异的指令, 使用不同的类型码进行标识, 并使用 5 位的位置编码(5L)标记相异位的起始位, 编码格式如图 5 的 D-type 类型.

② RISC-V 指令集具有很整齐的编码格式^[7]. RISC-V 指令格式如图 4, 指令中的源寄存器、目的寄存器、相同的立即数类型都被固定在了相同的位置, 指令编码变化很有规律. 同时, RISC-V 指令集有 32 个寄存器, 提供了 3 个寄存器操作数(rs1, rs2, rd), 位宽为 5 位. 因此, 在 MESDBCC-RV 编码格式中, 针对有 5 位与字典中的指令码相异的指令, 设计了 5 位位置编码(5L)和 5 位滑动掩码(5S)的编码格式. 使用滑动掩码时, 32 位的指令需要 5 位位置编码对指令的每一位进行标识, 编码格式如图 5 的 E-type 类型.

R-type					
funct7	rs2	rs1	funct3	rd	opcode
I-type					
Imm[11:0]		rs1	funct3	rd	opcode
S-type					
Imm[11:5]	rs2	rs1	funct3	Imm[4:0]	opcode
U-type					
Imm[31:12]				rd	opcode

图 4 RISC-V 指令格式

C-type						
0/1	类型码	字典索引				
D-type						
0/1	类型码	5L	字典索引			
E-type						
0/1	类型码	5L	5S	字典索引		
F-type						
0/1	类型码	3L	4F	4L	2F	字典索引
G-type						
0/1	原始指令码					

图 5 面向 RISC-V 指令集的压缩指令格式

③文献[3]研究了掩码位宽和掩码类型的组合方式, 他们得出的结论是 4 位固定掩码(4F)和 1 位滑动掩码(1S)可实现最佳的代码压缩率. 然而, 文

献[5]发现除了使用 4 位固定掩码外,还可以使用 4 位固定掩码(4F)与 2 位固定掩码(2F)相结合的方式获得更好的代码压缩率。因此,在 MESDBCC-RV 编码格式中,针对有多位与字典中的指令码相异的指令,设计了 4 位固定掩码(4F)与 2 位固定掩码(2F)相结合的编码格式。使用固定掩码时,4 位固定掩码将 32 位的指令码分为 8 部分,因此需要 3 位位置编码;2 位的固定掩码将 32 位指令码分为 16 部分,因此需要 4 位位置编码。编码格式如图 5 中的 F-type 类型。

根据上述分析,制定面向 RISC-V 指令集架构的 MESDBCC-RV 压缩指令格式,如图 5 所示。格式的首位为 0,对应使用小字典编码,格式的首位为 1,对应使用大字典编码,格式的 2-4 位对应编码类型(C/D/E/F/G-type),紧接着是位置编码(L),滑动掩码(S)位或者固定掩码(F)位,最后是字典的索引位。

3.2 压缩代码的硬件解压缩器

在不影响处理器结构和 Cache 工作机制的基础上,本文设计了精简的压缩代码的硬件解压缩单元。压缩代码的硬件解压缩单元主要完成的工作是将存储单元的压缩指令码在处理器触发的条件下将一个基本块进行解压缩,而后将其存放在指令缓存行中。压缩代码的硬件解压缩单元结构如图 6 所示,其由核心控制器,解码器,选择器,输入输出缓冲器,字典查找表(LUT)和 BitMask 运算单元等组成,字典查找表采用分离的两个字典结构,系统精简,执行高效。

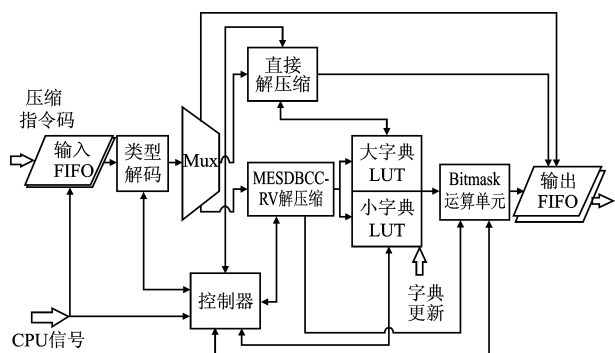


图 6 压缩代码的硬件解压缩结构

在解压缩时,对于压缩的指令码,通过字典索引还原出其在字典中对应的指令码,而后计算出原始指令码;对于未压缩的指令码,直接将其存入指令缓存单元。当处理器向解压缩器发出启动信号,解压缩器进入准备状态。在准备状态,解压缩器从外部存储

空间相应地址单元中分别收集字典指令码和压缩指令码对字典查找表和输入先进先出存储器(FIFO)进行初始化。字典查找表和输入 FIFO 初始化完成后,解压缩器进入等待状态。在等待状态,核心控制器接收到处理器的触发信号后,控制输入 FIFO 在一条压缩的指令码解压缩完成后移位出 FIFO 中未压缩的指令码并收集一条新的压缩指令码;控制输出 FIFO 存储解压缩后的指令,并将其传递给处理器或缓存。在解压缩过程中,指令类型解码器还原出待解压缩指令码的编码格式,并由多路选择器 MUX 将任务分配给直接解压缩器或者 MESDBCC-RV 解压缩器。直接解压缩器解压 C-type 类型压缩指令码, MESDBCC-RV 解压缩器解压 D/E/F-type 类型压缩指令码。G-type 类型指令未进行压缩,将其直接传递给输出 FIFO。MESDBCC-RV 解压缩器的计算由 BitMask 运算单元完成。BitMask 单元根据来自字典 LUT 的指令和由固定(滑动)掩码(nF/S)与位置编码(nL)形成的 32 位 Bitmask 执行异或(XOR)运算,以获取原始指令。

字典查找表 LUT 是影响解压缩器性能的关键模块。较大的 LUT 会产生较长的 LUT 访问延迟,从而限制解压缩硬件的性能。但是,限制 LUT 的尺寸又会降低原始指令码对字典指令码的命中率,从而导致代码压缩效率不高。本文使用完全分离的字典结构来优化字典查找表 LUT,减少解压缩延迟。如图 7 所示,完全分离的字典结构由 LUT,选择器 Sector 和路由器 Router 组成。n 入口的 LUT 被分成 i 个 LUT,在 MESDBCC-RV 代码解压缩器中,i 取 2。选择器 Sector 用于确定访问哪个 LUT,路由器 Router 用于将确定的 LUT 指令路由到输出缓冲区。这样完全分离的字典结构不会影响代码压缩率,并且减小了 LUT 访问延迟。

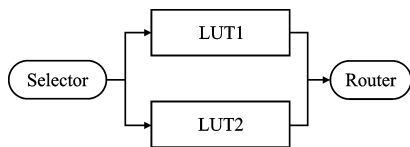


图 7 分离的字典结构

4 实验结果

本文基于 RISC-V 指令集,使用官方开源的工具链 riscv32-unknown-elf-gcc(C 运行库为 newlib),对本文提出的面向 RISC-V 指令集架构处理器的代码压缩算法 MESDBCC-RV 进行了测试

实验.

表 2 展示了三种应用场景下基于 RISC-V 指令集 MESDBCC-RV 算法的代码压缩率. 结果表明, MESDBCC-RV 算法使用的编码格式和字典构建方法, 可以达到 61.1% 的代码压缩率, 大大减小了 RISC-V 指令集处理器所需的程序存储器空间.

表 3 展示了 WANG 和 LIN 基于 ARM 指令集获得的代码压缩率^[6]: 1) DCC^[2]; 2) 4 位固定掩码 (4F) 和 1 位滑动掩码 (1S) 的 BCC^[3]; 3) 4 位固定掩码 (4F) 与 2 位固定掩码 (2F) 的 CLCBCC^[5]; 4) ARM Thumb-2^[10]. 面向 ARM 指令集的最优秀算法 CLCBCC (Codeword Length Constrained BitMask Code Compression, CLCBCC) 达到了 61% 的代码压缩率.

表 2 MESDBCC-RV 的代码压缩率(%)

算法类型	应用代码			
	fft	crc	mpeg2dec	average
MESDBCC-RV	60.4	61.0	61.9	61.1

表 3 基于 ARM 指令集的代码压缩率(%)

算法类型	应用代码			
	fft	crc	mpeg2dec	average
DCC ^[2]	72	73	69	72
BCC(1s+4f) ^[3]	65	67	66	66
CLCBCC ^[5]	61	62	60	61
Thumb-2 ^[10]	71	68	72	70

表 2 和表 3 展示的基于 RISC-V 指令集架构的 MESDBCC-RV 算法和基于 ARM 指令集架构的 CLCBCC 算法使用的指令集不同. 由于目前还没有出现面向 RISC-V 指令集的代码压缩技术, 本文间接借助基于 ARM 指令集架构的代码压缩算法做说明. 本文设计的基于 RISC-V 指令集 MESDBCC-RV 算法达到了目前 ARM 指令集架构优秀算法 CLCBCC 旗鼓相当的代码压缩率.

表 4 展示了在 RISC-V 指令集架构上, 图 5 设计的压缩指令格式的概率分布. 通过表 4 实验数据可以看出, C-type 类型的编码格式占比接近一半, 使用固定掩码或滑动掩码和位置编码相结合的 D-type 和 E-type 类型的编码格式占了很大一部分比例, G-type 类型的编码格式占比不足 10%. 本文提出的面向 RISC-V 指令集的编码格式大大提高了原始指令码对字典查找表的命中率, 达到了压缩后的指令码码字长度尽可能短的目的, 从而提高了代码压缩效率.

表 4 编码格式类型的概率分布(%)

算法类型	应用代码			
	fft	crc	mpeg2enc	average
C-type	48.1	52.3	46.7	49.0
D-type	20.2	19.1	20.1	19.8
E-type	13.4	10.1	10.8	11.4
F-type	11.1	12.4	13.3	12.3
G-type	7.2	6.1	9.1	7.5

Genesys2 是 DIGILENT 公司的一套高性能评估套件, 其搭载的 FPGA 型号为 Xilinx Kintex7-325T. 将 MESDBCC-RV 解压缩器在 Vivado 2018.3 上进行综合, 其 FPGA 资源消耗情况如表 5, 其中字典查找表由 FPGA 的 LUT 资源搭建. 同时, 表 5 借助嵌入式应用的最具代表性处理器 ARM Cortex-M3^[11], 对比了 MESDBCC-RV 解压缩器的 FPGA 资源消耗情况, 即使用 FPGA 的 LUT 资源搭建字典查找表, LUT 的资源消耗为 ARM Cortex-M3 处理器的 8.9%, FF 的资源消耗仅为 ARM Cortex-M3 处理器的 3.4%.

MESDBCC-RV 解压缩器在 Xilinx Kintex7-325T 上以 200 MHz 的时钟频率仿真运行, 1 秒平均可以解压缩 2.028×10^7 条 32 位的二进制指令, 带宽为 618.8 Mbps, 可以满足特定处理器的指令访存带宽需求. MESDBCC-RV 解压缩单元包含分离的两个大小字典查找表, 通过将字典查找表 LUT 分开, 将寄生电容分成较小的部分, 减少了指令访问延迟.

表 5 FPGA 资源消耗对比

FPGA 资源	MESDBCC-RV	ARM Cortex-M3 ^[11]	占比
FF	923	27 254	3.4%
LUT	2 025	22 712	8.9%

5 结束语

本文提出了首个面向 RISC-V 指令集架构处理器的代码压缩技术 MESDBCC-RV. 根据 RISC-V 指令集架构的特点, 对代码压缩算法和硬件解压缩器进行了设计, 通过使用高效的算法和精简的硬件系统, 有效减小了 RISC-V 指令集架构处理器所需的内存空间. 在以后的工作中, 希望集中于计算机体系结构的研究, 探索 Cache 与高性能解压缩器之间更加高效的耦合工作机制, 将 MESDBCC-RV 代码压缩技术应用于更高带宽要求的处理器.

参考文献:

- [1] WOLFE A, CHANIN A. Executing compressed programs on an embedded RISC architecture[C]//Proceedings of the 25th Annual International Symposium on Microarchitecture. Portland, USA: IEEE, 1992: 81-91. DOI:10.1109/MICRO.1992.697002.
- [2] LEFURGY C, BIRD P, CHEN I C, et al. Improving code density using compression techniques[C]//Proceedings of the 30th Annual International Symposium on Microarchitecture. Research Triangle Park, USA: IEEE, 1997: 194-203. DOI: 10.1109/MICRO.1997.645810.
- [3] SEONG S W, MISHRA P. A bitmask-based code compression technique for embedded systems[C]//Proceedings of 2006 IEEE/ACM International Conference on Computer Aided Design. San Jose: IEEE, 2006: 251-254. DOI: 10.1109/ICCAD.2006.320144.
- [4] QIN X K, MUTHRY C, MISHRA P. Decoding-aware compression of FPGA bitstreams[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2011, 19(3): 411-419. DOI: 10.1109/TVLSI.2009.2035704.
- [5] WANG W J, LIN C H. An improved BitMask based code compression algorithm for embedded systems[C]//Proceedings of 2011 International Symposium on Electronic System Design. Kochi, India: IEEE, 2011: 152-157. DOI: 10.1109/ISED.2011.15.
- [6] WANG W J, LIN C H. Code compression for embedded systems using separated dictionaries[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2016, 24(1): 266-275.
- [7] WATERMAN A, LEE Y, PATTERSON D A, et al. The RISC-V instruction set manual, volume I: User-level ISA, version 2.1[R]. Berkeley, CA: University of California, 2016.
- [8] 阳晔. 面向嵌入式处理器的代码压缩研究[D]. 杭州: 浙江大学, 2007: 17-18.
- YANG Y. Research on code compression for embedded processors[D]. Hangzhou: Zhejiang University, 2007: 17-18.
- [9] 洪兴勇. 高性能 BWDSP 处理器指令 Cache 研究与设计[D]. 合肥: 合肥工业大学, 2013: 55-56.
- HONG X Y. The research and design of high performance BWDSP processor instruction Cache[D]. Hefei: Hefei University of Technology, 2013: 55-56.
- [10] ARM. ARM architecture reference manual ARMv7-A and ARMv7-R edition[EB/OL]. [2020-08-14]. <https://developer.arm.com/documentation/ddi0406/c/>.
- [11] ARM. ARM cortex-M3 DesignStart eval FPGA user guide[EB/OL]. [2020-08-14]. <https://developer.arm.com/documentation/100896/0000/>

作者简介:

程战涛 男,(1994-),硕士研究生. 研究方向为超大规模集成电路设计与计算机体系结构.

梁 峰(通讯作者) 男,(1976-),博士,副教授. 研究方向为超大规模集成电路设计、未来存算一体化集成电路架构研究、人工智能嵌入式系统设计与实现.

E-mail: fengliang@xjtu.edu.cn.

张国和 男,(1981-),博士,教授. 研究方向为数模混合集成电路设计、视觉信息与图像处理 VLSI 设计、VLSI 可测性设计、类脑计算.