文章编号:1671-4598(2023)07-0258-07 DOI:10.16526/j.cnki.11—4762/tp.2023.07.037 中图分类号:TP332 文献标识码:A

基于符号补偿的 RISC-V 处理器乘法器优化

高嘉轩[1,2]，刘鸿瑾[2]，施 博[2]，张绍林[2]，华更新[1]
(1. 北京控制工程研究所，北京 100190；2. 北京轩宇空间科技有限公司，北京 100080)

摘要：针对高性能 RISC-V 处理器乘法运算延迟过长的问题，改进了基本乘法器中的基 4-Booth 编码以及 Wallace 树型结构，提出了基于符号补偿的基 4-Booth 编码以及交替使用 3-2 压缩器和 4-2 压缩器的 Wallace 树型结构；基于符号补偿的基 4-Booth 编码减少了部分积的数量，降低了符号位进位翻转带来的功耗；改进的 Wallace 树型结构减少了部分积累加所花费的时钟周期，缩短了乘法器的关键路径，降低了乘法指令的执行延迟；利用 VCS 仿真验证了改进的乘法器功能正确性，通过板级测试评估了其性能；结果表明，文章的乘法器功能正确，相较于 PicoRV32，执行整型乘法指令所花费的时钟周期缩短了 88.2%。Dhrystone 分数提高了 71.7%，功耗降低了 4.9%。

关键词：RISC-V；处理器；乘法器；符号补偿；Booth 编码；Wallace 树型结构

Optimization of RISC-V Processor Multiplier Based on Sign Compensation

GAO Jiaxuan[1,2]，LIU Hongjin[2]，SHI Bo[2]，ZHANG Shaolin[2]，HUA Gengxin[1]
(1. Beijing Institute of Control Engineering，Beijing 100190，China；
2. Beijing Xuanyu Space Technology Co.，Ltd.，Beijing 100080，China)

Abstract：For the problem of excessive latency in multiplication operations of high-performance RISC-V processors，the basic multiplier by modifying the radix-4 Booth based encoding and Wallace tree architecture are improved. A sign-compensation-based radix-4-Booth encoding method and a Wallace tree structure with alternating use of 3-2 compressors and 4-2 compressors are proposed. The radix-4 Booth based encoding with signed compensation reduces the number of partial products and decreases the power consumption caused by sign bit carry flipping. The improved Wallace tree structure reduces the clock cycles required for partial product accumulation，shortening the critical path of the multiplier and reducing the execution delay of multiplication instructions. VCS simulation is used to verify the functionality of the improved multiplier，and its performance is evaluated through the board-level testing. The results show that the proposed multiplier functions are correct. Compared to the PicoRV32，the required clock cycles for executing the integer multiplication instructions are reduced by 88.2%. The Dhrystone score increased by 71.7%，and the power consumption decreased by 4.9%.

Keywords：RISC-V；processor；multiplier；signed compensation；Booth encoding；Wallace tree structure

0 引言

目前嵌入式领域主要应用的处理器为 ARM 架构处理器，然而该架构经过多年的研究与扩展，其实现逐渐变得复杂。同时，ARM 架构需要高昂的授权费，导致应用开发成本增高，不利于进行自主扩展开发。而 RISC-V 是由加州大学伯克利分校提出的一个开源指令集架构，其特点是开源、简单、可扩展等，具有良好的内部结构[1]。其高度模块化的指令集架构可应用于各种需求的处理器设计[2]。出于自主可控的考虑，开源的 RISC-V 架构将成为处理器开发的新方向。作为一种新兴的开源指令集架构，RISC-V 已经在学术界和工业界都得到了广泛的关注[3]。RISC-V 生态系统的开源社区为其提供了丰富的对应工具链与开源软件，使研究者们能够在研究中快速利用 RISC-V[4]。基于 RISC-V 官方指令集手册，已发布了多款 RISC-V 处理器[5]，比如 Rocket Core[6]、SiFive[7]、BOOM Core[8]和 LowRISCSoC[5]。

乘法器是处理器的运算核心，其运行速度影响了处理器的运行速度[9]。许多数字信号处理和机器学习应用需要进行大量乘法计算，其表现在很大程度上受到乘法器性能的限制。以卷积神经网络为例，超过 90%的 CNN 计算为乘法累加计算[10]。因此面对嵌入式领域乘法算力需求较高的应用场景，研究开源指令集架构的 RISC-V 乘法器算力与功耗优化十分必要。乘法器主要包括三个阶段：操作数相乘产生部分积、部分积累加产生两个结果以及两个结果相加产生最终结果。目前整型乘法器的算法设计主要有串行累加阵列、Booth 编码和 Wallace 树型结构[11]。针对 Booth 编码的研究，通过改进后的 4 位编码加快了对部分积最低位的获取，但编码位数更多，逻辑更为复杂[12]。而缩减基 4-Booth 编码位数，则无法对最低位获取进行优化，浪费了芯片面积[13]。有研究采用了符号扩展，从而减少了

收稿日期:2023-05-18； **修回日期**:2023-05-22。
作者简介:高嘉轩(1998-)，女，北京人，硕士研究生，主要从事乘法器与处理器架构方向的研究。
通讯作者:刘鸿瑾(1980-)，男，湖南湘阴人，博士，研究员，主要从事宇航军用处理器及微系统技术方向的研究。
引用格式:高嘉轩，刘鸿瑾，施 博，等. 基于符号补偿的 RISC-V 处理器乘法器优化[J]. 计算机测量与控制，2023，31(7)：258-264，270.

资源消耗，但未考虑部分积累加时的压缩[14]。而针对 Wallace 树型结构，有研究表明仅由基本压缩器构成 Wallace 树结构，需要经过 5 级压缩才能将 9 个部分积压缩为 2 个结果，其运算效率十分低下[15]。文献［16］对 Wallace 树型结构进行了改进，缩短了压缩延时，解决了蜂鸟 E203 部分积压缩延时大的问题，但其未考虑有符号部分积扩展带来的损失。对压缩器的研究主要是对其硬件结构进行研究，通过传输门结构设计基本压缩器，节省了压缩器面积，但不能实现双轨输出[17]。而高阶压缩器可以由多个基本压缩器串联形成，但关键路径上门数较多，延时较长影响压缩器性能[18]。

为了在不提升处理器功耗的前提下，提高 RISC-V 乘法器算力，减少进位次数，压缩延时，本文提出了一种基于符号补偿、基 4-Booth 编码以及 Wallace 树型结构的优化乘法器。由于基 4-Booth 编码的系数存在负数，在压缩时，需要进行符号扩展补齐，负数部分积累加时将产生大量进位以及比特翻转。针对该问题，本文提出了基于符号补偿的基 4-Booth 编码，有效压缩了负数符号扩展时连续比特 1，减少了在加法时造成的进位以及比特翻转，降低了乘法器功耗。传统 Wallace 树型结构由于硬件压缩器功能简单，因此其层次过多，关键路径过长，结构不对称。针对该问题，本文提出了交替使用 3-2 压缩器与 4-2 压缩器的 Wallace 树型结构。改进后的 Wallace 树型结构对称，有效减少了树型结构的层次，缩短了关键路径的长度，压缩了累加过程的延迟，降低了乘法指令执行延迟，提高了处理器乘法算力。同时由于压缩器阶数不高，因此对电路复杂度以及延迟未有明显影响。本文从仿真、综合以及板级层面对其进行了功能性验证以及性能评估，测试结果表明，与未改进的 PicoRV32 相比，使用改进乘法器的 PicoRV32 在功耗有所降低的情况下，乘法器性能得到了显著提升。

1　RISC-V 指令集架构

作为一种新兴指令集架构，RISC-V 架构在设计之初总结了传统指令集的优缺点，避免了传统指令集的不合理设计。RISC-V 指令编码简单，并且为扩展指令集预留出了足够的编码空间。RISC-V 指令主要分为 6 个类型[19]，包括：R 型指令：寄存器－寄存器操作；I 型指令：短立即数和访存 load 操作；S 型指令：访存 store 操作；B 型指令：条件跳转操作；U 型指令：长立即数操作；J 型指令：无条件跳转操作。各个类型指令结构如图 1 所示。

31 27	26 25	24 20	19 15	14 12	11 7	6 0	
funct7		rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]			rs1	funct3	rd	opcode	I-type
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]					rd	opcode	U-type
imm[20|10:1|11|19:12]					rd	opcode	J-type

图 1　RISC-V 指令结构图

其中，opcode 为操作码，表示指令操作。funct3 和 funct7 部分作为 opcode 的附加字段，和 opcode 共同决定指令的具体操作。rs1 和 rs2 表示两个源操作数的寄存器编号，rd 表示目的操作数的寄存器编号。该编码结构位置固定，便于译码模块对指令进行解析。

RISC-V 指令集的整数乘法指令共有 4 条，分别为 MUL、MULH、MULHSU、MULHU，均为 R 型指令。指令具体描述及用法如表 1 所示。

表 1　RISC-V 乘法指令表

指令语法		指令操作	指令描述
mul	rd,rs1,rs2	rd←(rs1 * rs2)[31:0]	有符号乘法指令
mulh	rd,rs1,rs2	rd←(rs1 * rs2)[63:32]	有符号乘法取高位指令
mulhsu	rd,rs1,rs2	rd←(rs1 * rs2)[63:32]	有符号无符号乘法取高位指令
mulhu	rd,rs1,rs2	rd←(rs1 * rs2)[63:32]	无符号乘法取高位指令

其中 MUL 指令将源操作数进行符号位扩展，选取计算结果的低 32 位，MULH 指令将源操作数进行符号位扩展，选取计算结果的高 32 位，MULHSU 指令的源操作数 rs1 为有符号数，rs2 为无符号数，指令结果选取计算结果的高 32 位，MULHU 指令将源操作数进行零扩展，选取计算结果的高 32 位。

2　基 4-Booth 编码

1951 年，A. D Booth 在其论文“A Signed binary multiplication technique”中提出一种快速乘法算法——Booth 算法[20]，即为了解决有符号乘法运算中复杂的符号修正问题而提出一种乘法算法，将乘数转变数据表示形式，使其数据出现尽可能多的 0，其编码原理如式（1）所示。

$$y = 2^{n-1}(-y_{n-1}+y_{n-2})+2^{n-2}(-y_{n-2}+y_{n-3})+\cdots 2(-y_1+y_0)+(-y_0+y_{-1}) \tag{1}$$

式中，y 为乘数，n 为乘数位数，y_{n-1} 为 y 的 $n-1$ 位。

从式（1）可以看出，基础 Booth 编码并不能在硬件乘法器电路中起到真正的优化作用，实际部分积个数并未减少。因此在基 2-Booth 编码的基础上提出了改进的 Booth 编码，即基 4-Booth 编码，进一步减少部分积个数，从而达到减少硬件加法器的目的，其编码原理如式（2）所示。

$$y = 2^{n-2}(-2y_{n-1}+y_{n-2}+y_{n-3})+2^{n-4}(-2y_{n-3}+y_{n-4}+y_{n-5})+\cdots 2^2(-2y_3+y_2+y_1)+(-2y_1+y_0+y_{-1}) \tag{2}$$

式中，y 为乘数，n 为乘数位数，y_{n-1} 为 y 的 $n-1$ 位。

从式（2）可以看出，多项式的项数相较于式 1 减少了一半，因此通过基 4-Booth 编码可以有效减少部分积的个数。

然而基 4-Booth 编码的系数存在负数的情况，以 2^{n-2} 项系数为例，当 $y_{n-2}=0$，$y_{n-3}=0$ 且 $y_{n-1}=1$ 时，该项系数为 -2，因此需要考虑负数部分积对压缩的影响。由于部分积位数不同，在压缩时，需要进行符号扩展进行补齐，当部

分积为负数时，符号位补齐为 1，累加时将产生大量进位以及比特翻转，增加乘法器功耗。

3 Wallace 树型结构

基 4-Booth 编码优化了整型乘法中部分积的数量，从而减少累加次数。而为了进一步减少累加操作产生的延迟，1964 年，C. S. Wallace 提出的一种高效快速的加法树结构，被后人称为 Wallace 树[21]，其核心思想为将每 3 个加数分为一组，压缩至 2 个加数，即计算结果与进位，循环往复，最终得到累加结果，其结构如图 2 所示。

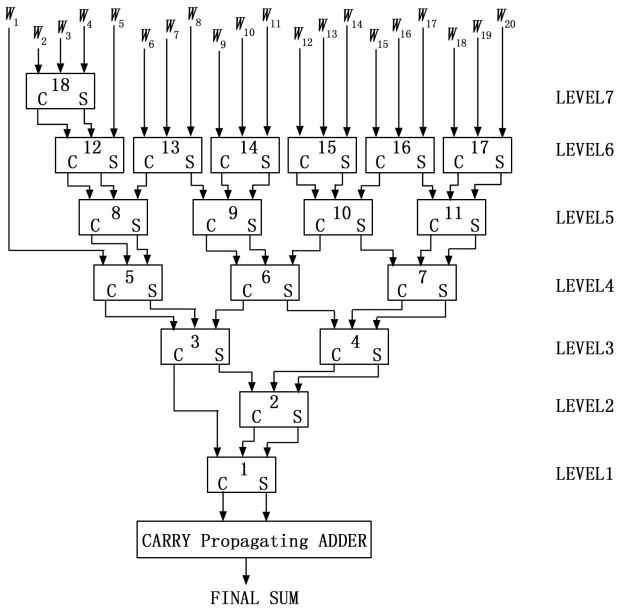

图 2 Wallace 树型结构图

Wallace 树型结构提高了累加操作的执行并发性，通过树型结构提高了硬件压缩器的复用性，缩短了关键路径的长度，有效地缩短了累加操作的执行时间。

然而传统 Wallace 树型结构由于硬件压缩器功能简单，其层次依旧过多，并且对于 32 位数据计算，其结构不对称，增加了电路的复杂度，消耗了更多布线资源。不规则的布局布线结构，增大了关键路径的时延。随着硬件压缩器的不断发展，Wallace 树型结构可通过使用高阶压缩器实现进一步优化。而阶数过高的压缩器，例如 7-3 压缩器、6-3 压缩器存在电路设计复杂、硬件占用资源多、功耗高以及面积大等问题，因此面对嵌入式领域对低功耗的需求，需要在控制乘法器功耗以及面积的情况下，优化 Wallace 树型结构，从而提高乘法器算力。

4 乘法器优化设计

本文设计的乘法器用于单发射顺序执行五级流水线 RISC-V 处理器 32×32 位有符号整数乘法运算，主要模块包括：通过改进的基 4-Booth 编码产生部分积，通过改进的 Wallace 树型结构对部分积进行压缩。RISC-V 处理器结构图以及乘法器的结构图如图 3 所示。

在处理器五级流水线结构中，乘法器位于执行阶段，译码模块从指令中解析出被乘数与乘数，通过改进的基 4-Booth 编码产生部分积，部分积阵列通过 Wallace 树型结构计算出累加结果和进位两个数据，最后通过加法器计算出最终结果，通过多选器根据指令功能选取对应的结果位。

本文对基 4-Booth 编码进行改进，将补码计算以及符号位扩展与基 4-Booth 编码相结合，减少符号位扩展对压缩效率的影响。对 Wallace 树型结构进行改进，加入 4-2 压缩

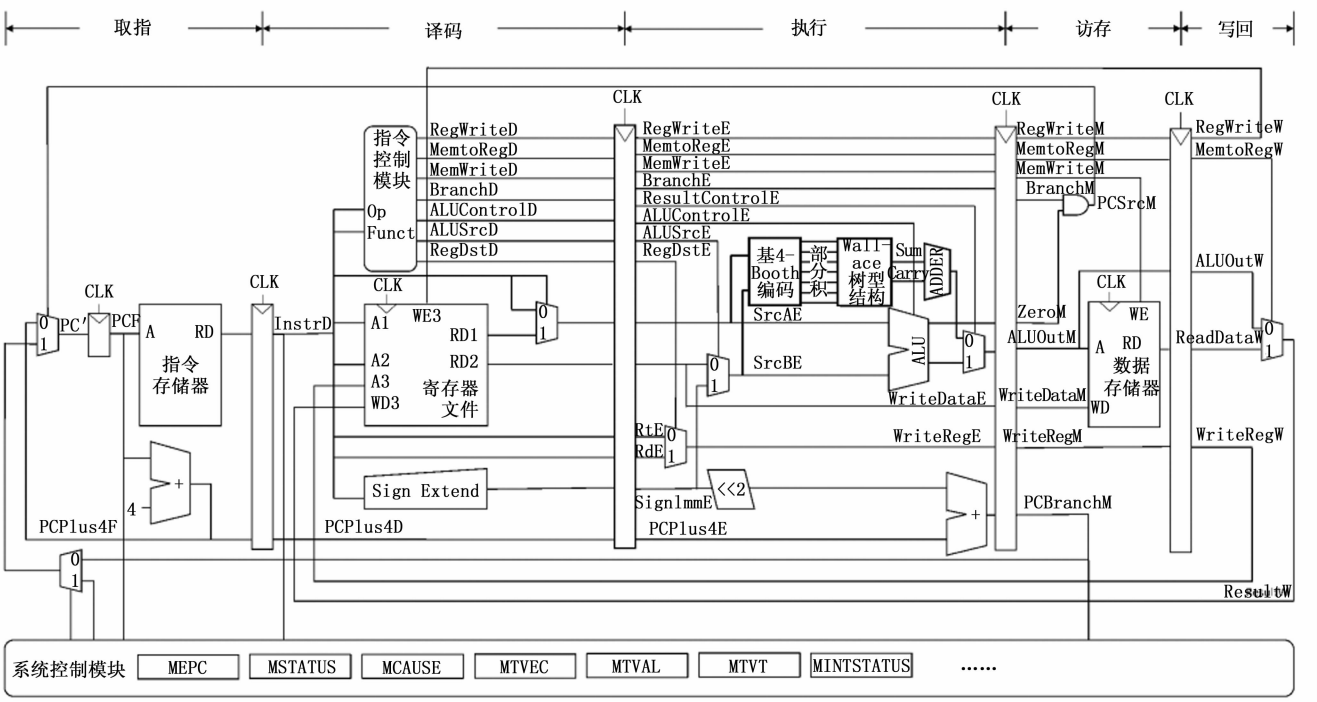

图 3 RISC-V 处理器结构图

器，使改进的 Wallace 树型结构对称，减少其层次数量，缩短关键路径长度。部分积阵列经过改进后的 Wallace 树型结构，产生两个结果，通过全加器得到乘法计算结果。

4.1　改进的基 4-Booth 编码

基 4-Booth 编码减少了部分积的数量，其中存在负数部分积，而当部分积为负数时，符号位补齐为 1，累加时将产生大量进位以及比特翻转，增加乘法器功耗。针对该符号位问题，有研究提出了一种有效的符号补偿方式[22]，有效减少了符号位扩展以及补码引起的功率消耗。采用符号位直接扩展时，以 8 位数据 snXXXXXXX 为例，其中 sn 表示符号位，将其扩展为 16 位数据，其结果为 sn sn sn sn sn sn sn sn snXXXXXXX。而使用该符号补偿方式，可对其进行等价逻辑变换，如式（3）所示：

$$\begin{aligned} &sn\ sn\ sn\ sn\ sn\ sn\ sn\ sn\ snXXXXXXX = \\ &11111111\ \overline{sn}XXXXXXX + 0000000010000000 = \\ &00000000\ \overline{sn}XXXXXXX + 1111111110000000 \end{aligned} \tag{3}$$

从式（3）可以得出，由于符号位扩展部分为连续相同的数据，因此根据二进制计算特点，可将符号扩展变换为只与符号位及固定值有关的运算，而固定值可提前根据部分积数量以及位数进行计算，其计算结果可在部分积累加前通过编码实现。因此本文通过将该符号补偿方式、补码计算与基 4-Booth 编码相结合，形成改进后的基 4-Booth 编码，扩展方式如图 4 所示。

其中：E 代表乘上系数后的符号位，S 代表补码操作造成的加 1 或加 2。在有效位最低的部分积前扩展 $E\overline{E}\overline{E}$，在有效位最高的部分积前扩展 E，在其余部分积前扩展 1E。低有效位部分积的 S 拼接在高有效位的部分积后面，从而在累加过程中完成补码的加法操作。

本文在基 4-Booth 编码的基础上，加入对符号补偿以及补码的编码，从而将上述符号补偿方式与基 4-Booth 编码相结合，减少部分积压缩的功率消耗。本文采用如表 2 所示的编码。

表 2　改进的基 4-Booth 编码表

$y_{2i+1}\ y_{2i}\ y_{2i-1}$	$Coef_i$	E	S	部分积
000	0	sn	2'b00	0
001	1	-sn	2'b00	X
010	1	-sn	2'b00	X
011	2	-sn	2'b00	2X
100	−2	sn	2'b10	−2X
101	−1	sn	2'b01	−X
110	−1	sn	2'b01	−X
111	0	sn	2'b00	0

表中 $y_{2i+1}\ y_{2i}\ y_{2i-1}$ 代表乘数的第 $2i+1$、$2i$ 和 $2i-1$ 位，对于 32 位处理器，i 的范围为 0 到 16；$Coef_i$ 代表第 i 个部分积系数；E 代表乘上系数后的符号位，sn 代表被乘数的符号位；S 代表补码加 1 操作产生的中间值，本级的中间值拼接至下一级部分积，在压缩过程中实现补码计算。改进的基 4-Booth 编码电路图如图 5 所示。

其中：Y_2i+1、Y_2i 和 Y_2i-1 代表 $y_{2i+1}\ y_{2i}\ y_{2i-1}$，即乘数的第 $2i+1$、$2i$ 和 $2i-1$ 位，sign 代表 $Coef_i$ 的符号，coef1 代表 $Coef_i$ [1]，coef0 代表 $Coef_i$ [0]，E 代表被乘数符号位的系数，S1 代表中间值 S [1]，S0 代表中间值 S [0]。

从图 5 可以看出，改进的 Booth 编码通过与门、或门、非门以及二选一多选器实现，经过 4 个门的时间延迟即可得出编码结果。

根据 $Coef_i$ 的编码结果，分别对被乘数进行相应操作。对于系数为负数的部分积，对被乘数进行按位取反，对于系数绝对值为 2 的部分积，对被乘数进行左移 1 位操作。将乘以系数的部分积与编码后的符号位以及中间值进行拼接，形成改进后的部分积，该部分积阵列输入 Wallace 树型结构进行累加计算。

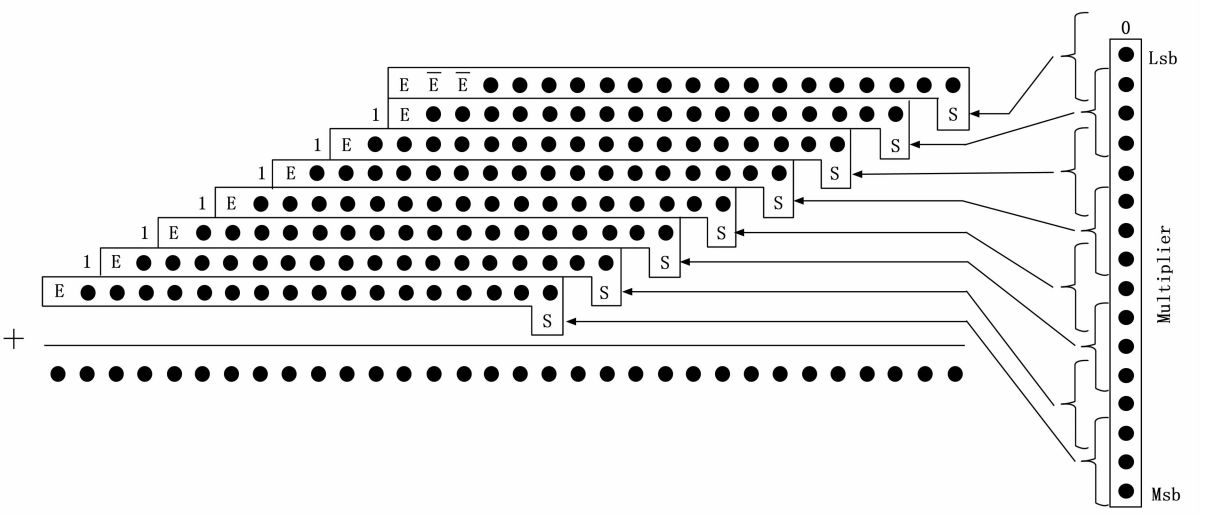

图 4　改进符号扩展图

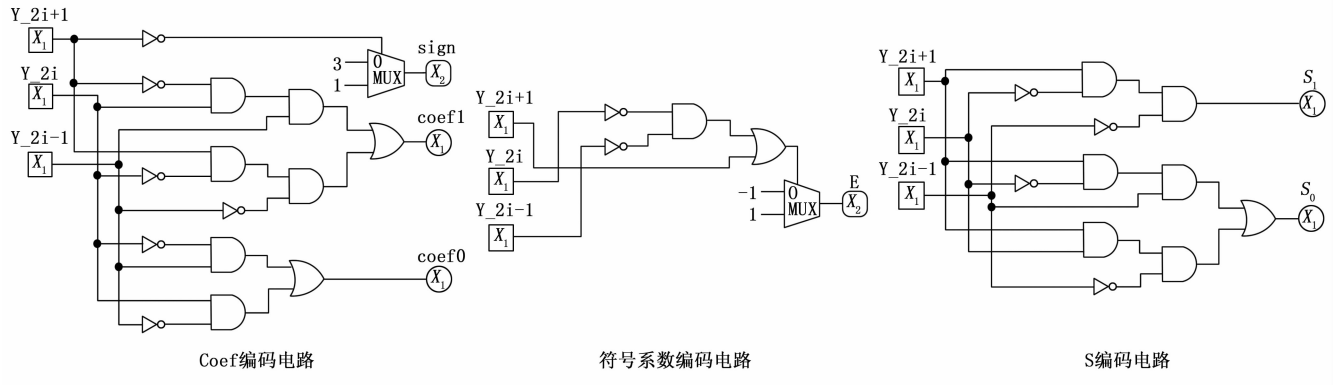

图 5 改进基 4-Booth 编码电路图

采用基于符号补偿的基 4-Booth 编码，不需要扩展全部符号位，减少了由于扩展大量连续的 1，在加法时造成的进位以及翻转，降低了乘法器功耗。

4.2 改进的 Wallace 树型结构

基础的 Wallace 树型结构采用 3-2 压缩器，通过树型结构有效提高累加操作的并行性，同时从硬件设计角度，增加了加法器的复用性，减少了硬件加法器的数量。为了更好地配合 32 位处理器的运算需求，在不提高处理器功耗的情况下，尽可能减少 Wallace 树型结构的层次，提出了一种交替使用 3-2 压缩器与 4-2 压缩器的 Wallace 树型结构，改进后的 Wallace 树型结构示意图如图 6 所示。

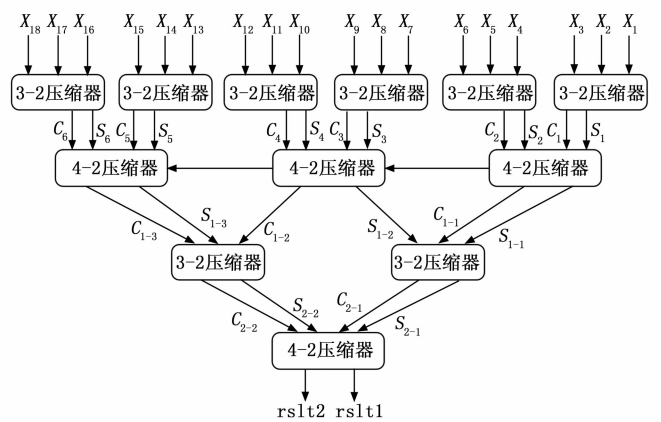

图 6 改进的 Wallace 树型结构图

由于使用符号补偿方式，32 位操作数将产生 17 个部分积，因此为了使 Wallace 树型结构对称，将 32 位乘法运算经过符号扩展，产生 18 个部分积，18 个部分积经过 3-2 压缩器产生 12 个操作数，再经过 4-2 压缩器产生 6 个操作数，6 个操作数经过 3-2 压缩器产生 4 个操作数，在经过 4-2 压缩器产生最终的 2 个结果。

其中 3-2 压缩器的逻辑表达式如式（4）～（5）所示：

$$S_0 = X_0 \oplus X_1 \oplus X_2 \tag{4}$$

$$C_1 = X_0X_1 + X_0X_2 + X_1X_2 \tag{5}$$

式中，S_0 为第 0 个 3-2 压缩器的压缩结果，C_1 为第 0 个 3-2 压缩器的进位结果，X_0、X_1 和 X_2 为 3-2 压缩器的 3 个输入操作数。

其 1 位电路示意图如图 7 所示。

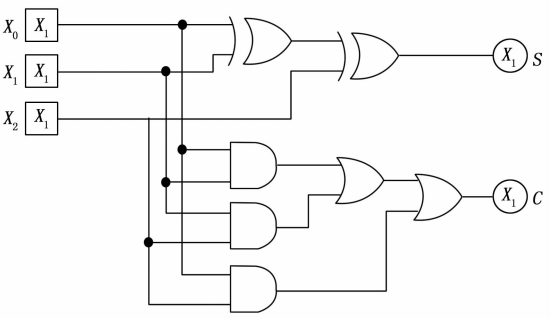

图 7 3-2 压缩器 1 位电路图

从图 7 可以看出，3-2 压缩器仅使用与门、或门和或非门构成，经过 3 个门的时间延迟即可得出结果。

4-2 压缩器，又称 5-3 计数器，包括 5 个输入和 3 个输出，输入和输出分别包括一个进位。传统的 4-2 压缩器是由两个串行连接的 3-2 压缩器组成，而改进后的 4-2 压缩器可以通过异或门和 2-1 的多选器构成，改进后的 4-2 压缩器的逻辑表达式如下所示：

$$C_{out} = X_0X_1 + X_0X_2 + X_1X_2$$

$$Xor = X_0 \oplus X_1 \oplus X_2 \oplus X_3$$

$$S_0 = Xor \oplus C_{in}$$

$$C_1 = XorC_{in} + \overline{Xor}X_3$$

式中，S_0 为第 0 个 4-2 压缩器的压缩结果，C_1 为第 0 个 4-2 压缩器的进位结果，C_{out} 为同级 4-2 压缩器的进位结果，X_0、X_1、X_2 和 X_3 为 4-2 压缩器的 4 个输入操作数。

由于 C_{out} 的逻辑表达式与 C_{in} 无关，所以 4-2 压缩器之间虽有信号连接，但不会形成行波进位链，4-2 压缩器之间依旧是并行的，其结构示意图如图 8 所示。

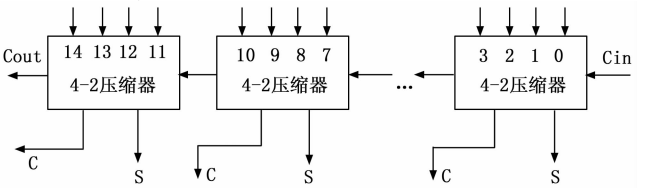

图 8 4-2 压缩器结构图

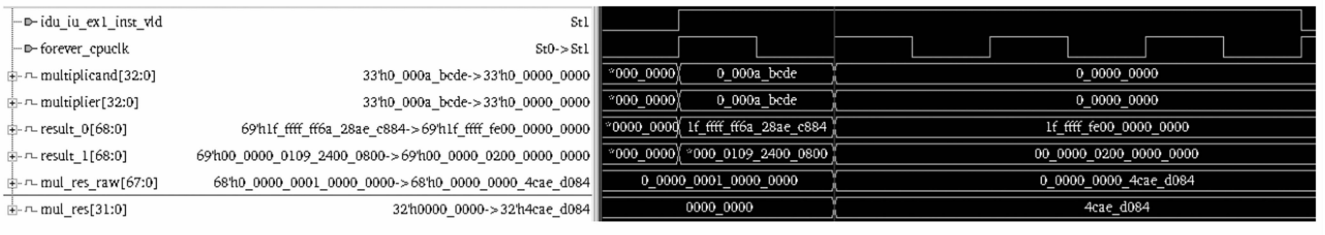

图 9　乘法指令波形图

采用 3-2 压缩器和 4-2 压缩器交替的 Wallace 树型结构对称，减少了布局布线资源，有效减少了树型结构的层次，缩短了关键路径的长度，压缩了累加过程的延迟。同时，使用的 3-2 压缩器和 4-2 压缩器电路简单，电路最大延时短，未造成处理器功耗以及面积大幅提高。

5　实验结果与分析

本文的实验环境如表 3 所示。

表 3　实验环境表

实验系统环境	实现语言	测试语言	仿真软件	处理器
Ubuntu 20.04	Verilog	RISC-V 汇编	VCS 2018	PicoRV32

采用 Verilog 语言对设计的 32 位乘法器进行实现，并嵌入到 PicoRV32 中，通过汇编语言对乘法器进行功能仿真，将结果与 Spike 仿真结果进行对比验证乘法器功能的正确性。

图 9 为乘法器执行乘法指令的仿真波形，其结果与 Spike 仿真的结果一致。

从图 9 可以看出，通过 MUL 指令计算 32’habcde 与 32’habcde 相乘，其计算结果为 32’h4caed084，与预期结果相符。通过如下所示代码，对其余指令以及边界值进行验证。

```
.global MUL
MUL:
li      x3, 0x80000000
li      x4, 0x7fffffff
li      x10, 0xabcde
mul     x5, x0, x3
bne     x5, x0, __fail
mul     x5, x0, x0
bne     x5, x0, __fail
mul     x5, x0, x4
bne     x5, x0, __fail
mul     x5, x4, x4
li      x6, 0x1
bne     x5, x6, __fail
mul     x5, x4, x0
bne     x5, x0, __fail
mul     x5, x4, x3
li      x6, 0x80000000
bne     x5, x6, __fail
mul     x5, x4, x10
li      x6, 0xfff54322
bne     x5, x6, __fail
mul     x5, x3, x3
li      x6, 0x0
bne     x5, x6, __fail
mul     x5, x3, x0
bne     x5, x0, __fail
mul     x5, x3, x4
li      x6, 0x80000000
bne     x6, x5, __fail
mul     x5, x3, x10
li      x6, 0x0
bne     x6, x5, __fail
mul     x5, x10, x10
li      x6, 0x4caed084
bne     x6, x5, __fail
```

上述代码对 32 位操作数的最大正值 0x7fffffff、最小负值 0x80000000、中间值 0xabcde 以及 0 进行乘法计算，通过对乘法器计算结果的判断，进行程序跳转，从而测试指令执行结果是否正确。当指令结果正确，即符合预期时，程序顺序执行，仿真成功结束。当指令结果不正确时，程序跳转至 _ _fail，表示程序执行错误，仿真错误结束。其余 3 条乘法指令与 MUL 指令测试程序相同。如图 10 所示，测试结果为仿真成功结束，表明指令执行结果正确，即设计的乘法器功能正确。

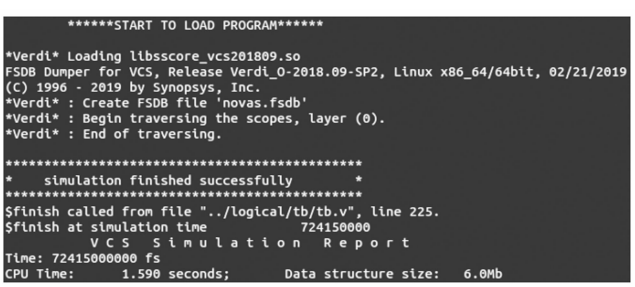

图 10　乘法指令测试结果图

通过图 9 波形图，本文的乘法器接收到乘法指令，第一个时钟周期计算累加结果以及进位，第二个时钟周期两者相加得出最终结果，因此乘法所需时钟周期为 2 个时钟周期。为了更加直观地体现本文乘法器的性能，将优化后的乘法器与 PicoRV32 乘法器以及文献［16］进行了性能对

比，结果如表 4 所示。

表 4 乘法器时钟周期数对比表

处理器	PicoRV32 乘法器	文献[16]	本文乘法器
时钟周期数	17	2	2

从表 4 可以看出，与文献 [16] 相比，本文的乘法器执行乘法运算花费的时钟周期数相同。而相较于 PicoRV32 乘法器，本文改进后的乘法器大幅减少了乘法计算所花费的时钟周期数，乘法指令执行时间缩短了 88.2%，减少了乘法指令的执行延迟，提高了处理器的算力。

本文使用 Synopsys 公司的 Design Compiler 软件，基于 smic40nm 工艺库，对乘法器进行了综合，对其面积进行了评估，结果如表 5 所示。

表 5 乘法器面积对比表

处理器	PicoRV32 原乘法器	文献[16]乘法器	本文乘法器
面积/μm^2	9 472.598 613	12 000	10 730.059 276

从表 5 可以看出，由于扩展了基 4-Booth 编码以及使用了 4-2 压缩器，本文乘法器相较于 PicoRV32 原乘法器在面积上增加了 13.3%。而与文献 [16] 乘法器相比，本文乘法器在面积上优于文献 [16]，减少了 10.6%。从表 4 可以得出，本文乘法器与文献 [16] 乘法器执行乘法指令时钟周期数相同，因此本文设计优化的乘法器优于文献 [16]。

Dhrystone 可对处理器的整型运算性能进行测量。因此本文通过运行测量处理器运算能力的基准程序之一的 Dhrystone，在板级对带有本文乘法器的 RISC-V 处理器进行了进一步性能与功耗测试，其性能与 PicoRV32 对比如表 6 所示。

表 6 处理器性能对比表

处理器	PicoRV32	文献[16]	PicoRV32＋本文乘法器
主频 (MHz)	100	100	100
Dhrystone (DMIPS/MHz)	1.231 930	2.001 643	2.115 728
功耗(W)	0.037 26	0.038 64	0.035 42

从表 6 可以看出，与 PicoRV32 相比，本文设计的乘法器在算力方面提高了 71.7%，而处理器算力提高的同时，处理器功耗降低了 4.9%。文献 [16] 提升了处理器算力，但处理器功耗也有所提高，而本文的乘法器与其相比，算力有所提高，功耗有所降低，提高了处理器的能耗比。

6 结束语

本文通过对 RISC-V 架构中整数乘法指令的研究以及乘法器的研究，提出了基于符号补偿的基 4-Booth 编码以及使用 3-2 压缩器和 4-2 压缩的改进 Wallace 树型结构，设计实现了改进后的 RISC-V 处理器乘法器。通过 RISC-V 汇编语言对改进后的乘法器进行功能仿真，验证了其功能正确性。使用 Design Compiler 软件对乘法器面积进行了综合，与处理器原乘法器以及已有工作进行了对比分析。通过板级测试，对处理器算力以及功耗进行了评估，并与原处理器以及已有工作进行了对比分析。结果表明，本文改进的乘法器功能正确，执行整型乘法指令所花费的时钟周期为 2，相较于 PicoRV32 乘法器，缩短了 88.2%。Dhrystone 分数为 2.115 728 DMIPS/MHz，功耗为 0.035 42 W，相较于未改进的 PicoRV32，性能提高了 71.7%，功耗降低了 4.9%，在功耗有少许降低的情况下，大幅提高了处理器性能，提高了处理器计算速度。与已有研究工作相比，本文改进的乘法器在执行乘法指令时钟周期数相同的情况下，面积与功耗均优于已有工作，适用于嵌入式领域对处理器面积、功耗以及算力有较高需求的应用场景。本文的乘法器确实在缩短整型乘法执行时间方面有一定效果，但在功耗方面优化并不明显，同时由于增加了符号扩展编码，乘法器面积有所增加，未来的改进方向为：

1) 优化编码设计，尽可能降低编码逻辑复杂度；

2) 考虑乘法器的低功耗优化。

参考文献：

[1] PATTERSON D A, WATERMAN A. The RISC-V Reader: An Open Architecture Atlas [M]. Berkeley, CA, USA, 2017.

[2] DU Z, FASTHUBER R, CHEN T, et al. ShiDianNao: Shifting vision processing closer to the sensor [J]. Acm Sigarch Computer Architecture News, 2015, 43 (3): 92 - 104.

[3] 刘 畅，武延军，吴敬征，等. RISC-V 指令集架构研究综述 [J]. 软件学报，2021，32 (12)：3992 - 4024.

[4] ASANOVIC K, PATTERSON D A. Instruction sets should be free: the case for RISC-V [R]. Berkeley, CA: University of California, 2014.

[5] 刘先强. 基于 RISC-V 的五级流水线处理器的设计与研究 [D]. 济南：山东大学，2021.

[6] 陈炳欣. RISC-V 借力 IoT “登堂入室” 中国 CPU 架构又迎发展机遇？[N]. 中国电子报，2018 - 05 - 04 (8).

[7] 何小庆. RISC-V 处理器嵌入式开发概述 [J]. 单片机与嵌入式系统应用，2020，20 (11)：1 - 6.

[8] 郭俊. 基于 RISC-V 处理器的分支预测研究与设计 [D]. 无锡：江南大学，2022.

[9] 李飞雄，蒋林. 一种结构新颖的流水线 Booth 乘法器设计 [J]. 电子科技，2013，26 (8)：46 - 48，67.

[10] CHANG Y J, CHENG Y C, LIAO S C, et al. A Low Power Radix-4 Booth Multiplier with Pre-Encoded Mechanism [J]. IEEE Access, 2020 (99): 1.

[11] 朱鑫标，施隆照. 一种高压缩 Wallace 树的快速乘法器设计 [J]. 微电子学与计算机，2013，30 (2)：46 - 49.

[12] 石 敏，王 耿，易清明. 基于改进的 Booth 编码和 Wallace 树的乘法器优化设计 [J]. 计算机应用与软件，2016，33 (5)：13 - 16.

[13] 曾宪恺，郑丹丹，严晓浪，等. 基于标准单元库扩展的快速乘法器设计 [J]. 计算机应用研究，2012，29 (5)：1778 - 1780，1814.

(下转第 270 页)