DOI: 10.16452/j.cnki.sdkjzk.2025.05.010 文章编号:1672-3767(2025)05-0101-10

基于改进 Radix-4 Booth 算法的逻辑综合中有符号乘法器设计

王立华,张家胜,徐 丽
(山东科技大学 电子信息工程学院,山东 青岛 266590)

摘 要:逻辑综合是集成电路设计的重要环节,在逻辑综合时乘法器单元需要自行设计。为优化有符号乘法器的电路延时并减小电路面积,提高乘法器的整体性能,本研究基于改进 Radix-4 Booth 算法设计了一种有符号乘法器。采用资源复用 Booth 编码器,将 3 位编码转换为 2 个控制信号,共同控制 Booth 选择器生成部分积,部分积的符号位则使用简单的电路统一扩展;采用进位保留加法器阵列对重组后的部分积进行压缩求和,缩短关键路径,减少电路面积。基于 SMIC 28 nm 工艺库,对采用改进算法设计的 16×16 bit 有符号乘法器进行逻辑等价性检查与逻辑综合,逻辑综合后网表的电路延时、电路面积与资源信息表明,该方法能较好地提升乘法器的电路性能。

关键词:有符号乘法器;Radix-4 Booth 算法;部分积重组;逻辑综合
中图分类号:TN492 **文献标志码**:A

Design of signed multiplier in logic synthesis based on improved Radix-4 Booth algorithm

WANG Lihua, ZHANG Jiasheng, XU Li
(College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China)

Abstract: Logic synthesis is an important part of integrated circuit design. During logic synthesis, the multiplier unit needs to be specially designed. In order to optimize the circuit delay, reduce the circuit area, and improve the overall performance of the multiplier, this study designed a signed multiplier based on the improved Radix-4 Booth algorithm. A resource multiplexing Booth encoder was used to convert the 3-bit code into two control signals, which jointly controlled the Booth selector to generate partial products. The sign bits of the partial products were uniformly expanded using a simple circuit. The reorganized partial products were compressed and summed by using carry-save adder to shorten the critical path and reduce the circuit area. Based on the SMIC 28 nm process library, the logic equivalence checking and logic synthesis were conducted on the 16×16 bit signed multiplier by using the proposed algorithm. The circuit delay, circuit area and resource information of the netlist after logic synthesis show that the proposed method can improve the circuit performance of the multiplier.

Key words: signed multiplier; Radix-4 Booth algorithm; partial product reorganization; logic synthesis

逻辑综合是在标准单元库和特定设计约束的基础上,将数字集成电路设计文件转换成优化门级网表的过程[1-2]。在数字集成电路(integrated circuit,IC)设计中,逻辑综合对电路性能的提升具有重要作用[3-4]。

乘法器是微处理器等数字信号处理电路中不可或缺的算术运算单元,其设计的好坏直接影响逻辑综合工具生成网表文件的面积、时序、功耗等,间接影响电路性能[5-8]。一个设计合理的乘法器能有效减小电路面

收稿日期:2024-06-30
基金项目:国家自然科学基金项目(62201325)
作者简介:王立华(1971—),男,山东临朐人,教授,博士,主要研究领域为嵌入式系统、物联网技术等,本文通信作者.
E-mail:13730917192@163.com

积和乘法周期数,从而提高电路的运算速度。近年来,人工智能(artificial intelligence,AI)领域的各种大模型被相继推出[9]。随着对 AI 大模型处理复杂任务的要求逐步提高,其中人工神经网络所需的乘加单元运算量不断增大,乘法器的运行速度极大地影响处理器的运行速度[10]。第三方标准单元库一般无乘法器硬件单元,逻辑综合过程中无法直接映射到该器件,需要自行设计。由于不同的乘法器设计方法对电路性能的影响不同,因此设计高性能的乘法器尤为重要。

高性能的乘法电路通常使用 Radix-4 Booth 算法,将乘法计算所需要的部分积数量减少为乘数位宽的一半[11-13]。许多研究对 Radix-4 Booth 算法提出优化和改进,使乘法器电路延时更小、面积更小、功耗更低。文献[14]通过基于静态分段方法的 Booth 编码方法生成部分积阵列,并对生成的部分积阵列进行误差补偿优化与近似压缩,以实现硬件性能和精度的折中;文献[15]基于修正 Radix-4 Booth 编码器,采用卡诺图数值替换方法实现新型近似 Booth 编码器,降低乘法器误差率和硬件开销;文献[16]采用逻辑简单的 Booth 电路生成部分积,提高了部分积生成的效率;文献[17]使用优化取反加 1 的方法直接生成被乘数的相反数,并采用单个全加器处理压缩器的中间进位,提高了部分积的压缩效率;文献[18]通过改进对乘数的取补码电路和提前对高位使用纹波进位加法器结构两种方法,加快了乘法器的运行速度。

上述文献分别对 Booth 电路进行了改进,改进后的电路优化了版图面积,提高了电路性能,但是电路延时有待优化、器件之间的复用率也有待提高。本研究以 chipsyn© 逻辑综合工具为例,采用改进的 Radix-4 Booth 算法实现有符号乘法器的设计,同时采用进位保留加法器(carry-save adder,CSA)对重组后的部分积进行压缩求和,进一步优化电路性能。

1 有符号乘法器设计

逻辑综合中存在有符号数相乘,有符号数以二进制补码的形式存在,最高位被用来表示该数字的符号。对于 16 位无符号数,其范围从 0 到 65 535,而有符号数的范围从 −32 768 到 32 767。对于二进制数,通常采用取反加 1 的方法求补码。在读取寄存器传输级(register-transfer level, RTL)设计文件时,需经第三方库将其转换为与内部数据结构匹配的网表文件,只在有符号数相乘时才会转换为有符号乘法器,而有符号乘法器的符号位取决于 RTL 文件中乘法器的连接关系,存在 3 种可能:常量 0、常量 1 和变量。

如图 1 所示,乘法器的设计分 3 步:部分积生成、部分积压缩和部分积相加[16,19-21]。在得到全部的部分积阵列后进行部分积的符号扩展,一般此步骤包含在部分积生成部分。为了减少加法器的数量,在部分积压缩前,一些乘法器设计会进行部分积重组[22]。

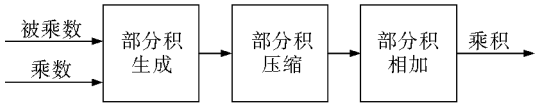


图 1 乘法器设计流程

Fig. 1 Flowchart of multiplier design

1.1 部分积生成

Radix-4 Booth 算法将 3 位乘数作为一组,每次重复前一组编码的高位,部分积结果可通过对被乘数进行简单的移位和求补码获得。使用 Radix-4 Booth 算法计算 $A\times B$:

$$\begin{aligned}A\times B &= -A\times B_{n-1}2^{n-1}+A\times B_{n-2}2^{n-2}+\cdots+A\times B_0 2^0+A\times B_{-1}\\ &=A\times\sum_{i=0}^{\frac{n}{2}-1}\left[(-2B_{2i+1}+B_{2i}+B_{2i-1})2^{2i}\right]。\end{aligned}\tag{1}$$

式中,B_{-1} 为 0。由式(1)可以看出,两个操作数的相乘仅生成 $n/2$ 个部分积,有效减少了计算部分积时所使用的加法器数量,进而减少了乘法器拆分后的电路面积。由式(1)中的基系数$(-2B_{2i+1}+B_{2i}+B_{2i-1})$可以得到对应的编码结果,编码操作如表 1 所示。

表 1 中的编码结果就是不同编码所对应的部分积。编码结果为 $+2A$,表示被乘数 $A\times(+2)$,而在二进制运算中,$A\times 2$ 的结果等同于 A 左移一位;编码结果为 $+A$,表示此次的部分积为 A 本

表 1 编码操作

Table 1 Encoding operation

B_{2i+1}	B_{2i}	B_{2i-1}	编码结果
0	0	0	$+0$
0	0	1	$+A$
0	1	0	$+A$
0	1	1	$+2A$
1	0	0	$-2A$
1	0	1	$-A$
1	1	0	$-A$
1	1	1	-0

身,无需额外操作;编码结果为+0 或−0,表示此次部分积为 0;编码结果为$-A$,表示被乘数 A 需要进行取反+1;编码结果为$-2A$,表示被乘数 A 取反+1 后还需左移一位。

根据 Radix-4 Booth 算法的基本原理,在对乘数被乘数编解码之前,先进行扩位处理:有符号数中,若乘数与被乘数为奇数位宽,则低位补一位 0,高位补最高位;若为偶数位宽,只需低位补一位 0。补位结束后选择乘数 B 作为 Booth 编码器的输入,3 位一编码,生成对应的控制信号。每次编码都需要把前一次编码的高位作为此次编码的低位,再选择 2 个高位数值进行编码,以此循环,编码到乘数 B 的最高位结束。编码规则如图 2 所示。使用 Booth 编码器生成的控制信号依次选择被乘数 A 的每一位作为 Booth 选择器的输入生成部分积。

根据表 1 计算出编码结果$+A$、$+2A$、$-A$ 和$-2A$ 的逻辑表达式,并且默认这些逻辑表达式全为 0 的状态,即为编码结果 0 的状态,对+0、−0 进行了统一处理,无需多余的逻辑生成数值为 0 的部分积,对应的电路结构如图 3(a)所示。从图 3(a)可以看出,Booth 编码器由 1 个异或门、6 个与门和 3 个非门组成,其中异或门可以表示为$(B_{2i}+B_{2i-1})\&((!\ B_{2i})+(!\ B_{2i-1}))$,也可以表示为$!((B_{2i}\&B_{2i-1})+((!\ B_{2i})\&(!\ B_{2i-1})))$,而$(B_{2i}\&B_{2i-1})$及$((!\ B_{2i})\&(!\ B_{2i-1}))$是 Booth 编码器的组成部分。由此可看出,图 3(a)所示的 Booth 编码器还可以进一步优化。

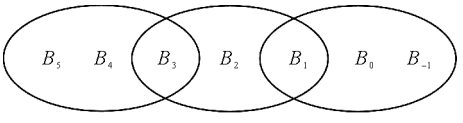
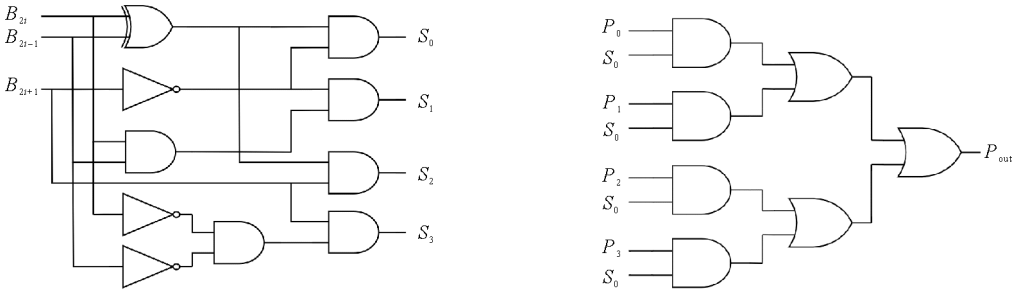


图 2 Radix-4 Booth 算法编码规则

Fig. 2 Encoding rule of Radix-4 Booth algorithm

Booth 选择器由 4 个与门和 3 个或门组成,如图 3(b)所示。该结构逻辑简单,只需要根据生成的控制信号 S_0、S_1、S_2、S_3,选择部分积可能值 P_0、P_1、P_2、P_3 中的一个,就能得到正确部分积 P_{out}。



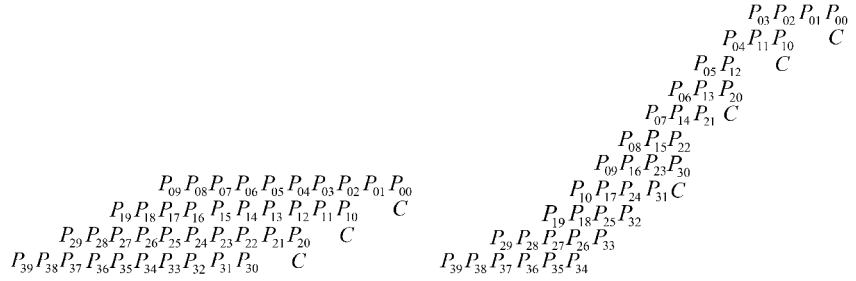
(a) Booth编码器 (b) Booth选择器

图 3 Booth 电路结构

Fig. 3 Structure of Booth circuit

1.2 部分积重组

为了减小部分积压缩过程中使用的加法器数量,本研究对部分积阵列进行重组。以 8×8 乘法器为例,图 4(a)为部分积重组前的排列方式,每一行代表一个部分积行,遵循逐位对齐原则。这种排列方式的缺点是部分积在某些列中分布过密,导致这些列需要使用较多的全加器和半加器进行压缩。图 4(b)为部分积重



(a) 部分积重组前的排列方式 (b) 部分积重组后的排列方式

图 4 部分积排列方式

Fig. 4 Arrangements of partial product

组后的排列方式,可以看出,重组后的部分积项更加均匀地分布在各列中,避免部分列中加数项过多,可以有效减少某些列中压缩所需的加法器数量。

1.3 部分积压缩与求和

采用 4-2 压缩器进行部分积的压缩运算,最终将全部部分积压缩为 2 个部分积。再使用加法器将压缩后的 2 个部分积相加,得到有符号乘法器的乘积结果。4-2 压缩器的电路结构如图 5 所示,其中 X_1、X_2、X_3 和 X_4 为 4 个输入信号位,sum、carry 和 E_i 为压缩器的 3 个输出信号位,且 E_i 为压缩器输出结果的传递位,该传递位会作为下一组 4-2 压缩器的部分积压缩器 E_{i-1} 的输入;XOR 和 MUX 分别是异或门和多路选择器,MAJORITY 器件是由逻辑与器件和逻辑或器件组成的多数门,其逻辑表达式为 $(X_4\&X_3)||(X_3\&X_2)||(X_4\&X_2)$。

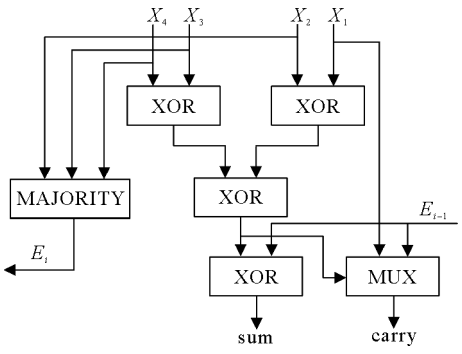


图 5 4-2 压缩器的内部结构

Fig. 5 Internal structure of 4-2 compressor

2 改进的有符号乘法器设计

如图 6 所示,有符号乘法器由数据处理、部分积生成(Booth 编码及解码)、部分积扩展、部分积重组以及部分积压缩求和等 5 个模块组成。对有符号乘法器设计的改进主要体现在数据处理、部分积生成及部分积重组模块。

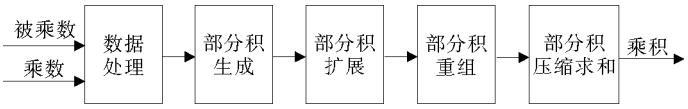


图 6 有符号乘法器组成

Fig. 6 Composition of signed multiplier

2.1 数据处理

网表文件中的乘法器,其输入可能存在无效数据,如无符号数高位为常量 0,有符号数含多余符号位。根据操作数相乘的原理,这两种情况都能被优化,即剔除无效的高位数据,获得进行计算的有效输入,减少拆分过程中产生的多余逻辑单元,最后通过扩展输出结果的高位数据来获得原始数据的结果。乘法器输入数据的预处理流程如图 7 所示,为乘法器的拆分准备有效的输入数据。

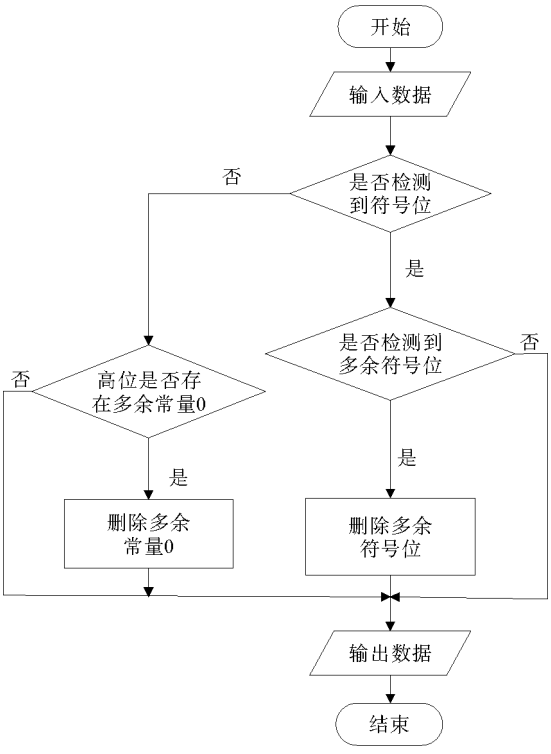


图 7 数据预处理流程图

Fig. 7 Flowchart of data preprocessing

2.2 部分积生成

Radix-4 Booth 算法决定了部分积数量由乘数确定,根据存储的输入信息,即输入位宽及其奇偶性,选择更合理的数据作为乘数进行编码,可以产生最少的部分积。输入作为待检查数据,根据存储的输入信息及数据检查原则进行处理,获得乘数与被乘数对应的数据。数据检查原则是:

1) 获取乘法器两个输入的位宽,小位宽输入暂时作为乘数处理,大位宽输入暂时作为被乘数处理。

2) 获取两个输入位宽对应的奇偶性,若输入位宽皆为奇数或偶数,处理结果不变;若作为乘数的输入位宽为奇数,另一输入位宽为偶数,处理结果不变;若作为乘数的输入位宽为偶数,另一输入位宽为奇数,且位宽差小于 4,则乘数与被乘数数据互换,否则处理结果不变。

为了优化传统算法中 Booth 电路的性能,本研究采用一种资源复用的 Booth 编码器,如图 8(a)所示。由于编码结果$+A$、$+2A$、$-A$ 和$-2A$ 中,$+A$ 与$-A$、$+2A$ 与$-2A$ 只是符号不同,因此可通过表 1 得到编码结果 A 与 $2A$ 的逻辑表达式,A 与 $2A$ 的符号则由 3 位编码信号中的 B_{2i+1} 位确定,此操作可减少一部分重复的逻辑。编码结果 A 对应图 8(a)中的控制信号 1,编码结果 $2A$ 对应控制信号 2。此外,控制信号 1 的产生是基于控制信号 2 的逻辑电路,充分利用了产生控制信号 2 的资源,节省了电路面积。

图 8(b)为本研究所使用的 Booth 选择器,由 1 个异或门、2 个与门和 1 个或门组成。根据 Booth 编码器得到的控制信号 2,从扩位后被乘数的低位开始依次选择 2 位数值,每次重复一位输入到 Booth 选择器即可获得部分积。以第一次得到的部分积作为固定的位置,每一次得到的部分积需要在前一个部分积的位置左移两位,每一行部分积低位补相应数量的常量 0。

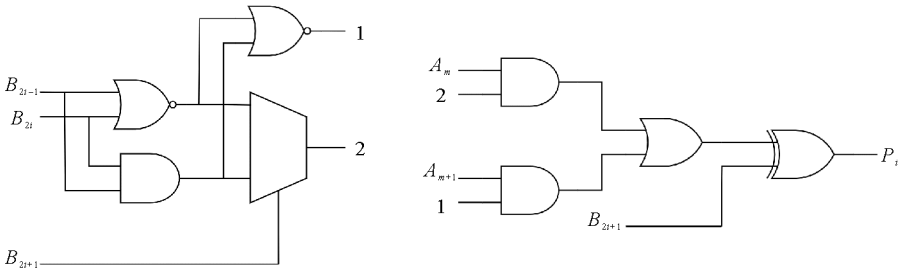


图 8 改进的 Booth 电路

Fig. 8 Improved Booth circuit

2.3 部分积扩展

由 Radix-4 Booth 算法可知,每 3 位一编码获得一行部分积。除第一行部分积外,其余的部分积都需要在上一次部分积的位置左移两位,因此只需将 Booth 编码器的输入高位,即 B_{2i+1} 替代下一轮 3 位编码生成的部分积低位,就能得到部分积$+1$ 的效果。

有符号乘法器补位及扩位操作如图 9 所示。图 9 中,S 表示扩展的符号位,其值为每一行部分积对应的最高位;□表示部分积的每一位数据,由 Booth 选择器生成;B_{2i+1} 为每次编码的高位,低位则依次补常量 0。以 16×16 有符号乘法器为例,需补齐乘法器输出所需要的 32 位数据。

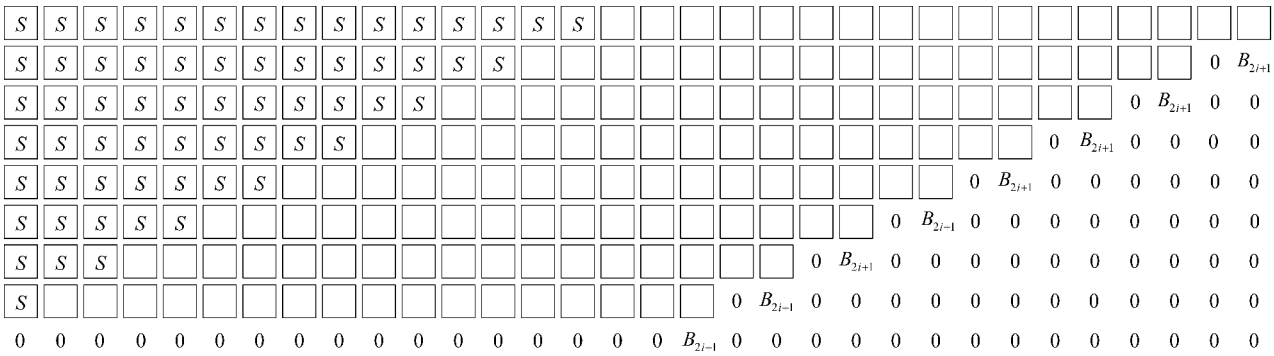


图 9 部分积阵列补位及扩位操作

Fig. 9 Operation of bit padding and bit expansion of partial product array

2.4 改进的部分积重组方式

与传统的部分积重组不同,本研究采用的部分积重组是基于部分积中每个数据之间的关系设计的。因为逻辑表达式一般情况下是可以化简的,比如$(A+(!\ A)\&B)$可以化简为$(A+B)$,不仅节省了一个与门和一个非门,还缩短了关键路径,减少了延时。而部分积阵列中的每一个数据都是一组逻辑表达式的结果,如

果相加的数据之间存在公共因子或者本身相等,使用加法器求和就能简化逻辑关系,达到减少延时和面积的效果。因此,对扩展后的部分积阵列进行重组,将存在公共因子的数据(以下简称同源数据)或者相等的数据尽可能排列在一起。

首先,以列为单位,根据每一列部分积是否含有同源数据或者两个数据本身是否相等,对部分积的行顺序进行排列,并对排列完成后的部分积阵列再进行一次重组。然后,将不符合上述规则的部分积阵列按照行顺序依次两两比较,看两行之间是否出现存在同源数据或者本身相等的情况,并将符合此规则的部分积位宽间隔小的数据优先排列在一起。部分积重组能简化加法器之间的逻辑关系,减少一定的冗余资源。

部分积阵列重组示意图如图 10 所示,4 行部分积中存在($A+B$)与 C、D、A 相加。在检测到该列数据含有公共因子后,将含 A 的第四行数据与含 C 的第 2 行数据互换,确保($A+B$)先与数据 A、D 求和,其结果可以化简为((! A)&B)^D,与原结果($A+B$)^C^D 相比,逻辑关系得到了优化。同时,因为含 A 的第 4 行数据移到第 2 行,与第 1 行数据中的 B 也产生了电路化简的效果。

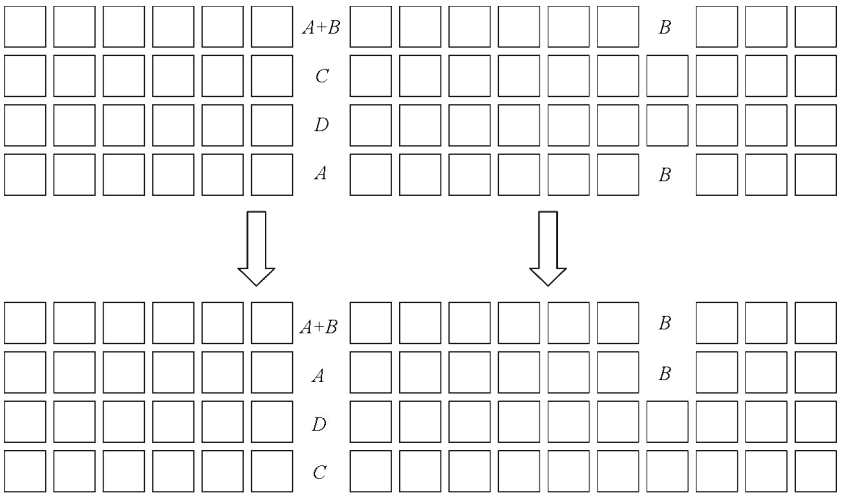


图 10 部分积重组示意图

Fig. 10 Schematic diagram of partial product recombination

2.5 CSA 阵列压缩求和

在乘法器设计中通常使用 Dadda Tree[23]、Wallace Tree[24] 或 CSA 阵列进行部分积的压缩求和。虽然 Dadda Tree、Wallace Tree 提高了乘法器速度,但是本身规则性较差。而 CSA 本身比较规则,形成的阵列版图非常紧凑,能将 3 个操作数压缩为 2 个,并在压缩过程中保留进位,不仅提高了加法器的速度,还能减小加法器的面积。因此,本研究采用 CSA 阵列完成部分积的压缩求和。

CSA 本质上是全加器,而全加器的逻辑表达式不唯一,为电路优化提供了多种可能。结合本研究设计,采用 XOR+MUX 结构组成 CSA,其电路结构如图 11 所示,图中 a、b 为 CSA 的输入信号位,c_{in} 为 CSA 的进位输入。

根据改进的 Radix-4 Booth 算法,有符号 16×16 bit 乘法电路共生成 9 个部分积,在获得 9×32 的部分积重组阵列后,以部分积阵列为基础按列计算数据。第 1 列虽然为 9 个数据,但其有效数据仅为 2 个,本研究采用半加器计算这 2 个数据。第 2 列的有效数据加上来自低位的进位,也仅有 2 个有效数据,采用半加器实现。从第 3 列开始采用 CSA 进行计算,按照图 12 所示的连接关系对部分积进行压缩求和。

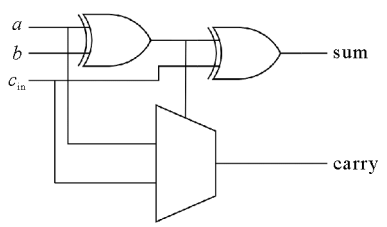


图 11 CSA 的电路结构

Fig. 11 Circuit structure of CSA

图 13 为本研究最终设计的 16×16 bit 有符号乘法器的具体结构图,其中 X 和 Y 是乘法器的两个 16 位输入,分别作乘法器的乘数和被乘数,O 是乘法器的 32 位输出。

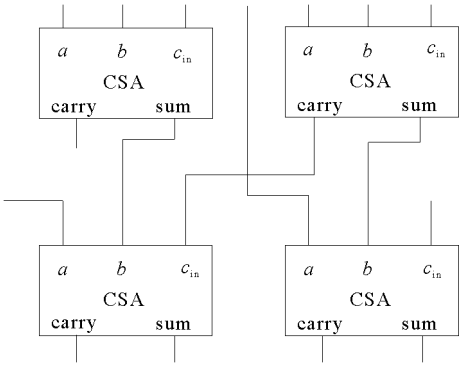


图 12 CSA 阵列连接方式

Fig. 12 Connection method of CSA

3 实验结果与分析

3.1 逻辑等价性检查

本研究采用 Cadence 公司开发的电子设计自动化(electronic design automation,EDA)工具 Conformal 进行逻辑等价性检查(logical equivalence checking,LEC),验证所设计的有符号 16×16 bit 乘法器网表文件与原始 RTL 文件是否等价。逻辑等价性检查是一种形式验证方法,用于比较两个电路逻辑功能的一致性,通过采用匹配点并比较这些点之间的逻辑来完成等价性检查,其具体过程主要分为设置、映射和比较 3 个步骤,如图 14 所示。

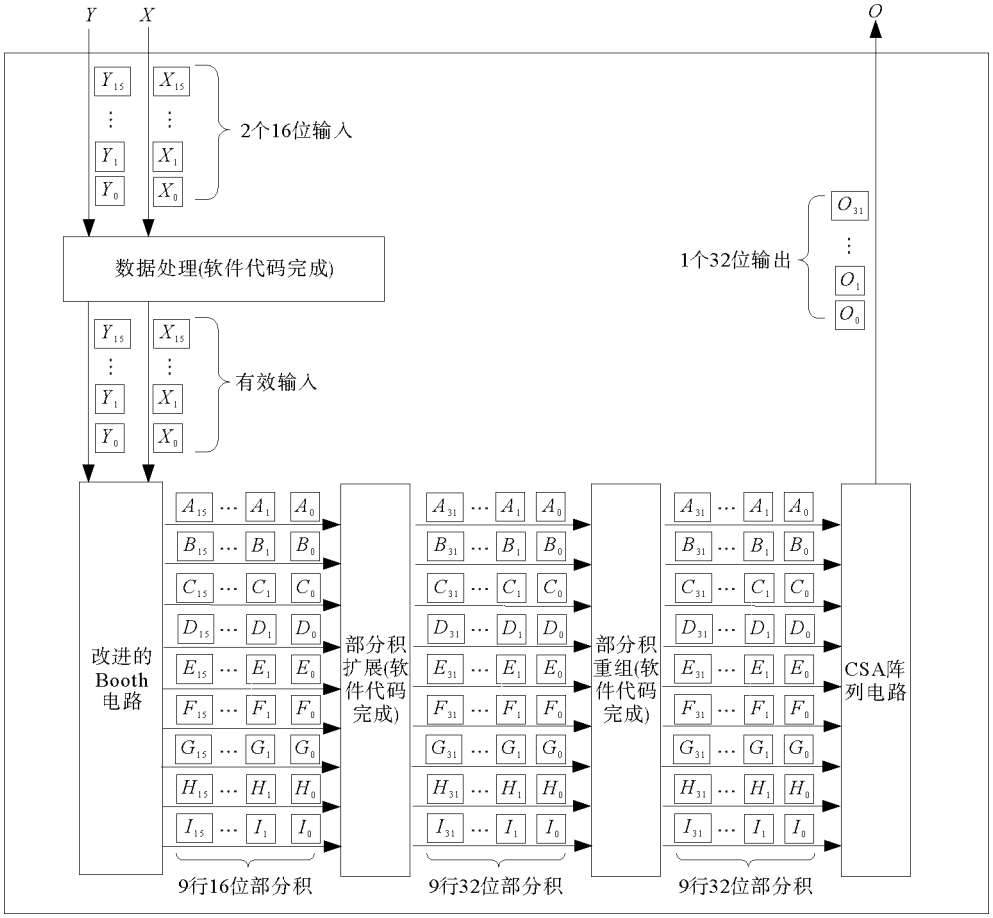


图 13 有符号乘法器具体结构图

Fig. 13 Specific structure diagram of signed multiplier

采用 Verilog HDL 语言编写有符号 16×16 bit 乘法器的 RTL 设计文件,将其作为形式验证的 Golden 文件(综合网表),优化修改后的网表文件作为 Revised 文件。设置 Conformal 工具的标准单元库为 SMIC 28 nm 工艺库,读入 Golden 与 Revised 文件,输入相关命令,Conformal 工具自动映射关键点并进行比较。比较检查关键点以确定其是否等价,若不等价则输出 Non-equivalent。逻辑等价性检查结果如图 15 所示,

表明逻辑综合前后文件逻辑等价,乘法器设计正确无误。

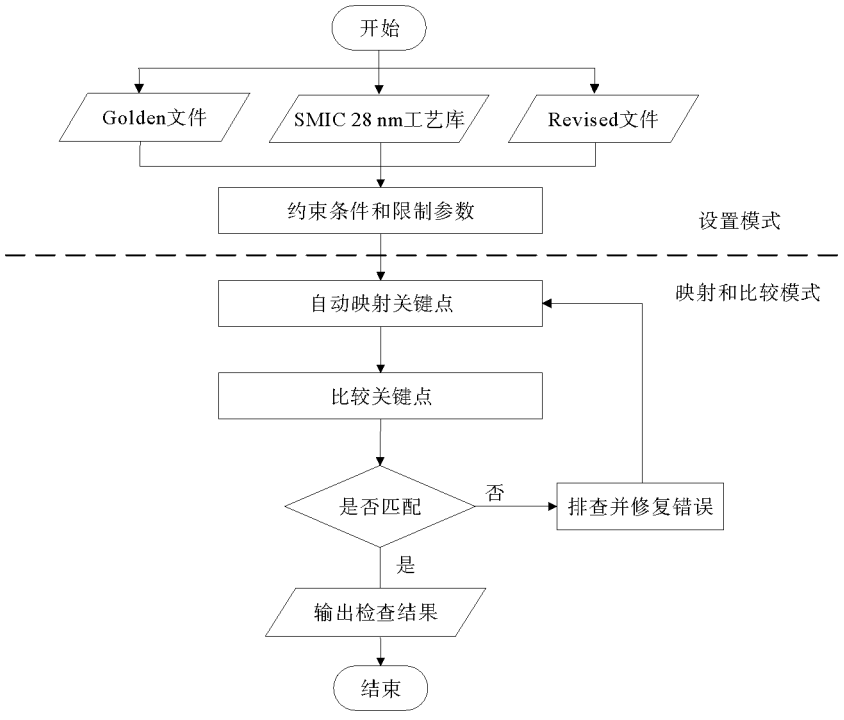


图 14 逻辑等价性检查流程图

Fig. 14 Flowchart of logical equivalence checking

```
================================================================
Mapped points: SYSTEM class
----------------------------------------------------------------
Mapped points     PO        Total
----------------------------------------------------------------
Golden            32        32
----------------------------------------------------------------
Revised           32        32
================================================================
// Running automatic setup...
// Automatic setup finished.
// Command: add compared points -all
// 32 compared points added to compare list
// Command: compare
================================================================
Compared points      PO        Total
----------------------------------------------------------------
Equivalent           32        32
================================================================
```

图 15 逻辑等价性检查结果图

Fig. 15 Result of logical equivalence checking

3.2 结果分析

本研究在逻辑综合工具 chipsyn© 中进行乘法器设计,并输出其逻辑综合后的面积、延时及所使用的器件资源。表 2 列出了采用本研究的资源复用 Booth 编码器、文献[16]的新型 Booth 选择器和文献[18]的改进 Booth 编码器的乘法器面积、延时及所使用的资源数量。

表 2 乘法器性能比较

Table 2 Performance comparison of multipliers

乘法器	电路延时/ns	电路面积/μm^2	电路资源/个
本研究乘法器	0.676 9	448.546	1 061
新型 Booth 选择器的乘法器[16]	0.707 1	455.112	1 123
改进 Booth 编码器的乘法器[18]	0.702 8	453.624	1 107

由表 2 可知,在电路延时方面,本研究采用资源复用 Booth 编码器的乘法器延时为 0.676 9 ns,比采用新型 Booth 选择器的乘法器延时减少 0.030 2 ns,比采用部分积生成器的乘法器延时减少 0.025 9 ns;在电路面积方面,本乘法器面积比采用新型 Booth 选择器的乘法器面积减小 6.566 μm^2,比采用改进 Booth 编码器的乘法器面积减小 5.078 μm^2;在电路资源方面,该乘法器所用器件数量为 1 061 个,比采用新型 Booth 选择器的乘法器减少 62 个,比采用改进 Booth 编码器的乘法器减少 46 个。

4 结论

为提升逻辑综合工具中有符号乘法器的电路性能,本研究采用一种资源复用的 Booth 编码器,同时采用 CSA 阵列对重组后的部分积进行压缩求和,以获得更小的电路延时和面积。与采用新型 Booth 选择器的乘法器相比,本研究乘法器的电路延时减小 4.27%,面积减小 1.44%;与采用改进 Booth 编码器的乘法器相比,本研究乘法器的电路延时减小 3.69%,面积减小 1.12%。逻辑综合实验结果表明,本研究设计的有符号乘法器能较好地提升乘法器电路的性能,在对乘法器有高速需求的集成电路设计中有很好的应用前景。

参考文献:

[1] NETO W L,LI Y,GAILLARDON P E,et al. FlowTune:End-to-end automatic logic optimization exploration via domain-specific multi-armed bandit[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2023,42(6):1912-1925.

[2] 韩进,程勇,齐现英. VHDL 在数字集成电路设计中的应用[J]. 山东科技大学学报(自然科学版),2003,22(4):74-77.
HAN Jin,CHENG Yong,QI Xianying. Application of VHDL in the design of digital integrated circuit[J]. Journal of Shandong University of Science and Technology (Natural Science),2003,22(4):74-77.

[3] SAITO R,AYALA C L,CHEN O,et al. Logic synthesis of sequential logic circuits for adiabatic quantum-flux-parametron logic[J]. IEEE Transactions on Applied Superconductivity,2021,31(5):1-5.

[4] KIM S,LEE S Y,PARK S,et al. A logic synthesis methodology for low-power ternary logic circuits[J]. IEEE Transactions on Circuits and SystemsI:Regular Papers,2020,67(9):3138-3151.

[5] 唐俊龙,汤孟媛,吴圳羲,等. 32 位 RISC-V 处理器中乘法器的优化设计[J]. 电子设计工程,2022,30(6):61-65.
TANG Junlong,TANG Mengyuan,WU Zhenxi,et al. Optimization design of multiplier in 32-bit RISC-V processor[J]. Electronic Design Engineering,2022,30(6):61-65.

[6] TSOUMANIS K, AXELOS N, MOSCHOPOULOS N, et al. Pre-encoded multipliers based on non-redundant Radix-4 signed-digit encoding[J]. IEEE Transactions on Computers,2016,65(2):670-676.

[7] 惠亚娟,李青朕,王雷敏,等. 基于电压调控自旋轨道矩器件多数决定逻辑门的存内华莱士树乘法器设计[J]. 电子与信息学报,2024,46(6):2673-2680.
HUI Yajuan,LI Qingzhen,WANG Leimin,et al. In-memory Wallace tree multipliers based on majority gates with voltage gated spin-orbit torque magnetoresistive random access memory devices[J]. Journal of Electronics and Information Technology,2024,46(6):2673-2680.

[8] 邓建,徐洁. 一种自动生成 Wallace 树形乘法器 Verilog 源代码方法[J]. 实验室研究与探索,2018,37(7):122-125.
DENG Jian,XU Jie. An efficient method of generating Verilog source code for Wallace tree multiplier[J]. Laboratory Research and Exploration in Laboratory,2018,37(7):122-125.

[9] 张旭,王旭强,田雨婷,等. 基于主题感知的跨模态序列到序列生成模型[J]. 山东科技大学学报(自然科学版),2021,40

(3):71-79.
ZHANG Xu, WANG Xuqiang, TIAN Yuting, et al. Topic-aware based cross-modal sequence-to-sequence generation model based on topic[J]. Journal of Shandong University of Science and Technology(Natural Science), 2021, 40(3): 71-79.

[10] 朱建芹,韩进. 基于 FPGA 的 IDCT 变换的设计与实现[J]. 山东科技大学学报(自然科学版),2011,30(6):91-96.
ZHU Jianqin, HAN Jin. Design and implementation of IDCT transform based on FPGA[J]. Journal of Shandong University of Science and Technology(Natural Science), 2011, 30(6): 91-96.

[11] 卢文娟,江明嘉,方华,等. 改进型定制乘法器的优化设计[J]. 中国集成电路,2023,32(4):46-54.
LU Wenjuan, JIANG Mingjia, FANG Hua, et al. Optimized design of improved custom multipliers[J]. China Integrated Circuit, 2023, 32(4): 46-54.

[12] LIU W L, QIAN L Y, WANG C H, et al. Design of approximate Radix-4 Booth multipliers for error-tolerant computing [J]. IEEE Transactions on Computers, 2017, 66(8): 1435-1441.

[13] 黄焘,闰闰,胡毅,等. 一种高能效基 4-Booth 编码并行乘法器设计[J]. 电子技术应用,2023,49(4):117-122.
HUANG Tao, RUN Run, HU Yi, et al. An energy efficient radix-4 Booth encoding parallel multiplier design[J]. Application of Electronic Technique, 2023, 49(4): 117-122.

[14] 侯博文,彭泽阳,贺雅娟. 一种基于静态分段补偿的近似乘法器设计[J]. 微电子学,2023,53(5):814-819.
HOU Bowen, PENG Zeyang, HE Yajuan. Design of an approximate multiplier based on static segment method[J]. Microelectronics, 2023, 53(5): 814-819.

[15] 盛勇侠,梁华国,肖远,等. 基于新型压缩树的近似 Booth 乘法器[J]. 微电子学,2022,52(3):425-430.
SHENG Yongxia, LIANG Huaguo, XIAO Yuan, et al. An approximate Booth multiplier based on novel Wallace tree[J]. Microelectronics, 2022, 52(3): 425-430.

[16] 王佳乐,胡越黎. 基于新型 Booth 选择器和压缩器的乘法器设计[J]. 微电子学与计算机,2020,37(3):5-8.
WANG Jiale, HU Yueli. Design and implementation multiplier based on new Booth selector and compressor[J]. Microelectronics and Computer, 2020, 37(3): 5-8.

[17] 李娅妮,郎世坤,王雅,等. 一种高效 16-bit 有符号数乘法器设计[J]. 集成电路与嵌入式系统,2024,24(6):41-45.
LI Yani, LANG Shikun, WANG Ya, et al. Design and implementation of an efficient 16-bit signed number multiplier[J]. Integrated Circuits and Embedded Systems, 2024, 24(6): 41-45.

[18] 范文兵,周健章. 基于 Radix-4 Booth 编码的并行乘法器设计[J]. 郑州大学学报(工学版),2025,46(1):26-33.
FAN Wenbing, ZHOU Jianzhang. Design of parallel multiplier based on Radix-4 Booth coding[J]. Journal of Zhengzhou University (Engineering Science), 2025, 46(1): 26-33.

[19] 彭泽阳,侯博文,贺雅娟. 基于新型 4-1 压缩器的低功耗近似乘法器[J]. 微电子学,2023,53(5):820-826.
PENG Zeyang, HOU Bowen, HE Yajuan. A low-power approximate multiplier based on novel 4-1 compressor[J]. Microelectronics, 2023, 53(5): 820-826.

[20] 仲亚,叶瑶瑶. 基于新型压缩器的乘法器设计[J]. 微电子学与计算机,2019,36(3):28-31.
ZHONG Ya, YE Yaoyao. Design of multiplier based on new compressor[J]. Microelectronics and Computer, 2019, 36(3): 28-31.

[21] CUI X P, LIU W Q, CHEN X, et al. A modified partial product generator for redundant binary multipliers[J]. IEEE Transactions on Computers, 2016, 65(4): 1165-1171.

[22] 徐东明,卢斌. 一种改进的 CSA 低功耗阵列乘法器的实现[J]. 微电子学与计算机,2016,33(9):19-23.
XU Dongming, LU Bin. A low power multiplier with modified carry save array[J]. Microelectronics and Computer, 2016, 33(9): 19-23.

[23] POORNIMA H S, NAGARAJU C, YADAV S T S. Synthesis and simulation of a low-power, high-efficiency and effective dadda multiplier[C]//2022 Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology. Mandya: IEEE Press, 2022: 1-6.

[24] LAKSHMI V, REUBEN J, PUDI V. A novel in-memory Wallace tree multiplier architecture using majority logic[J]. IEEE Transactions on Circuits and SystemsI: Regular Papers, 2022, 69(3): 1148-1158.

(责任编辑:齐敏华)