

## 高性能 SIMD 乘法阵列体系结构

吴虎成, 刘洋徐瑞, 刘建平

(国防科技大学 计算机学院, 湖南 长沙 410072)

**摘 要:** 描述了一种新型的高性能高效 SIMD 乘法阵列的结构. 该乘法阵列支持同时执行 1 个 64 位乘法, 4 个 32 位乘法或 16 个 16 位有符号/无符号乘法. 通过修改乘法算法实现结构, 提高了乘加单元的面积复用度, 在较小的面积和性能开销下实现了上述功能. 并引入了“溢出补偿技术”解决了复数矩阵乘法运算的判溢出问题. 通过牺牲非关键路径上短位宽乘法性能, 提高关键路径上高位宽乘法性能. 所述结构与文献[1]中乘法簇结构相比, 64 位乘法延时减少 3.65%, 面积降低 3.92%, 功耗提高 5.71%.

**关键词:** SIMD; 乘法; 定点; 乘加; Booth 编码; 点积乘法; 复数乘法

**中图分类号:** TP331.22

**文献标识码:** A

**文章编号:** 1000-7180(2014)03-0009-05

## Architecture of a High Performance SIMD Multiplication Array

WU Hu-cheng, LIU Yang-xu-rui, LIU Jian-ping

(School of Computer Science, National University of Defense Technology, Changsha 410072, China)

**Abstract:** This paper presents a 64-bit fixed-point SIMD multiplication array. This array has capable of supporting one  $64 \times 64$ , four  $32 \times 32$  or sixteen  $16 \times 16$  bit signed/unsigned. The component is used more efficient with multiplication cell array. Implement all above function with a small increase in delay and area. An algorithm of “overflow compensate” is introduced, which solve the overflow judgment in fixed-point complex multiplication. The synthesize result shows that the multiplication reduces the 64 bit multiplication's critical path by 3.65%, reduces the area by 3.92% and increases the power consumption by 5.71%.

**Key words:** SIMD; multiply; fix-point; MAC; Booth code; dot multiply; complex number multiply

### 1 引言

对于现代微处理器, 乘加操作是一种基本的操作. 微处理器性能的一项重要指标就是峰值乘加次数/秒(GMAC). 由于峰值乘加次数/秒 = 运算单元数量/SIMD 宽度/主频, 所以现代处理器通过不断提高核内的运算单元的数量来提高峰值运算性能<sup>[1]</sup>. 然而由于处理器内部的面积有限, 所以高性能低面积的乘加单元对于提高处理器的性能有着重要的意义. 随着 SIMD 宽度的不断提升, 乘加器需要的功能种类也不断提高. 同时, 乘加单元的设计受到面积、功耗、主频、功能的约束越来越严格<sup>[2]</sup>. 本文提出了一种高性能低面积高宽度 SIMD 乘加部件的体系

结构在较低的面积开销和性能开销下, 极大增强了乘加单元的功能, 使其能够同时执行 1 个 64 位乘法, 4 个 32 位乘法或 16 个 16 位有符号/无符号乘法. 该结构使用了乘法单元阵列结构, 该结构的关键在于对于所有的运算宽度, 都使用一个运算阵列, 从而提高了硬件复用度, 节省了硬件面积和功耗.

区分量的下标字母的正斜体: 量符号和代表变动性数字及坐标轴的字母作下标, 采用斜体字母.

### 2 SIMD 乘法阵列体系结构

#### 2.1 总览

SIMD 乘法阵列体系结构如图 1 所示. SIMD 乘加部件由操作数分配, 乘法单元阵列, 部分及压缩阵

收稿日期: 2013-05-12; 修回日期: 2013-07-03

基金项目: 国家自然科学基金项目“多核多线程 DSP 适应性存储结构研究”(61070036)

列和加法器组成. 经过操作数分配后, 乘法运算阵列中的每一个运算模块进行单独的 Booth 编码和 ES 编码, 并将 Booth 编码的部分积进行压缩.

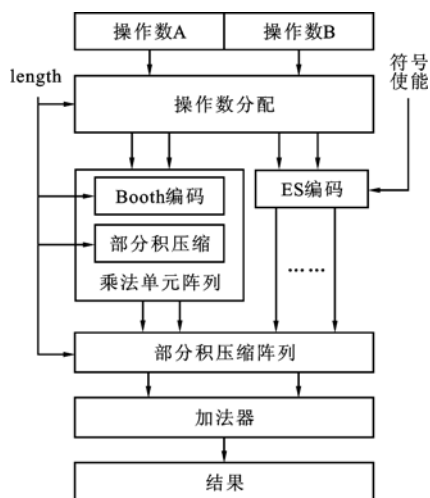


图1 乘加部件体系结构

经过压缩后, 每个运算模块产生 2 个 31 位的部分积, 称为 CS, 共生成 16 组 CS. ES 编码器产生乘法个数相同的 ES 对. 每个生成的 ES 与经过组合的 CS 与再次压缩后产生两部分积. 经过加法器后便得到最终结果. 总的运算部件被运算长度 (length) 和两个操作数的符号位使能所控制 (Sign1, Sign2). Booth 编码与部分积压缩单元组成了乘法运算阵列. 该阵列为所有乘法模式所共用.

## 2.2 乘法运算阵列

乘法运算阵列由 16 个乘法单元组成, 如图 2 所

示, 该阵列以  $4 \times 4$  的方式排列. 每个乘法单元与  $16 \times 16$  乘法器类似. 输入为两个经过操作数分配的 16 位数据. 每个乘法单元内包含 Booth 编码和部分积压缩单元.

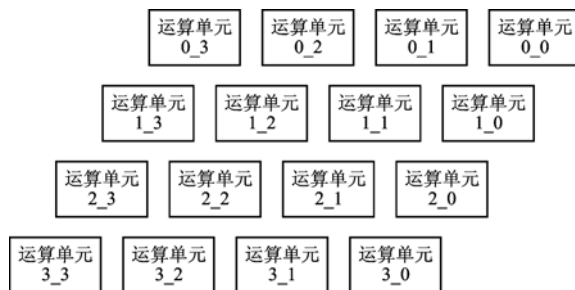


图2 运算单元排列结构

乘法阵列通过运算长度 length 控制组成 16/32/64 位乘法的乘法阵列. 如图 3 所示, 执行 64 位乘法时, 0\_0 号~3\_3 号乘法单元组成 64 位乘法. 执行 32 位乘法时, 0\_0, 0\_1, 1\_0 和 1\_1 号乘法单元, 即图 2 中右上角的 4 个乘法单元组成乘法通路 1, 其他通路以此类推. 执行 16 位乘法时每个乘法单元为 1 个运算通路.

如图 3 所示, 不同位上所需压缩的 CS 对数量不同. 其中数量最多的位于 64 位乘法的 48~64 位和 64~80 位两个区间内, 其 CS 对分别为 7 对, 加上 ES 对一共 8 对. 这 8 对部分积经过 3 次 4-2 压缩后形成 2 对最终部分积 result, 该路径即为整个乘法阵列的关键路径.

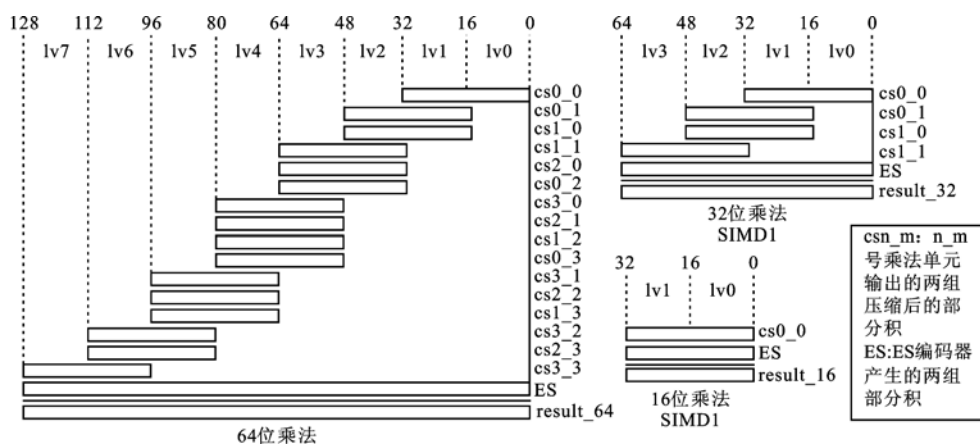


图3 不同长度乘法的组成结构

每个乘法单元通过 length 生成的 msb 和 lsb 控制. 每个乘法单元有一对单独的 msb 和 lsb 作为输入. 当该乘法单元位于所执行乘法行的最高位时, msb 为 1, 当该乘法单元位于所执行乘法行的最低位时, lsb 为 1.

乘法单元内的 Booth 编码器采用修改后的基 4Booth2 编码, 使用有限符号扩展技术<sup>[3]</sup>, 设文献中提到的补在 msb 上的为 E, 补在 lsb 上的为 S. 产生 8 行部分积. 经过部分积阵列 1 即 2 级 4-2 压缩后, 降低为 2 行 31 位部分积.

对于 16 位乘法,每个 1 乘法单元可执行 1 个乘法,即可同时执行 16 个 16 位乘法.对于 32 位乘法,4 个乘法单元联合完成 1 个乘法,即可同时执行 4 个 32 位乘法.对于 64 位乘法,16 个乘法单元联合执行 1 个乘法,即单位时间内只可执行 1 个 64 位乘法.

### 2.3 部分积压缩阵列

乘加器通过乘法运算单元的拼接组合成不同长度的乘法.如图 4 所示,运算单元的上下拼接较为简单,深色部分为部分积.

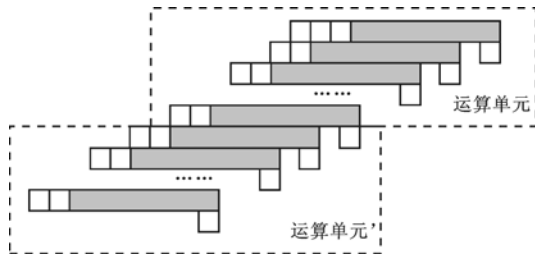


图 4 上下相邻乘法单元的组合

由于纵向的运算阵列拼接不存在重叠的部分,所以运算单元的纵向拼接不需要对 booth 编码和运算阵列进行修改.但横向拼接在左侧的运算阵列的最低位与右侧的运算单元的最高位有一位存在重叠,如图 5 所示.所以需要对 booth 编码进行修改,以此完成拼接.

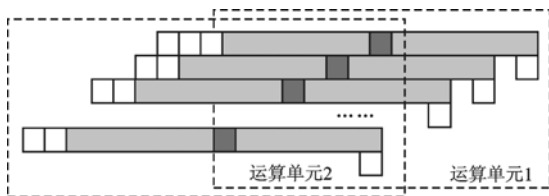


图 5 左右相邻乘法单元的组合

### 2.4 修改后的 Booth 编码

由上节提到的部分积重叠问题,所以需要对该 booth 编码进行修改.由于同一位上只需要 1 个数,所以将该数向前提 1 位.该 booth 编码与长度 length 相关.修改后的最终 booth 编码如表 1 所示.

表 1 修改后的 booth 编码

| 被乘数 | 部分积                         |
|-----|-----------------------------|
| 000 | 0                           |
| 001 | {msb&Sign1&A[15],A}         |
| 010 | {msb&Sign1&A[15],A}         |
| 011 | {A,0}                       |
| 100 | {~A,lsb}                    |
| 110 | {A[15]? msb&! Sign1:msb,~A} |
| 101 | {A[15]? msb&! Sign1:msb,~A} |
| 111 | {msb,16'hFFFF}              |

### 2.5 ES 编码

本文采用 ES 编码与部分积分开编码和压缩的方式隐藏了 Booth 编码的部分延时开销.2.3 节中提到的部分积压缩阵列排除了有限符号扩充技术中的 ES.该 ES 在部分积压缩阵列中进入压缩阵列,如图 6 所示.ES 编码器中将图 4 中白色部分编为两行 32 位的数据,称为 E 行与 S 行.编码方式如图 6 所示.灰色部分为参与编码的部分,其中 partial 为无符号乘法时的最后一行部分积,也参与 ES 编码.

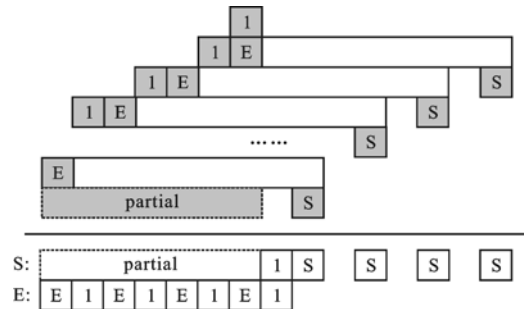


图 6 ES 编码

### 2.6 乘加运算溢出补偿技术

执行点积乘法或复数乘法操作  $A * B + C * D$  时.由于加法器延时和面积开销大于 4-2 压缩器,所以将  $A * B$  与  $C * D$  的 2 组共 4 个部分积经过 4-2 压缩后产生 2 个部分积再进行相加.但由于判溢出需要提前扩充加数与被加数的符号位,而上述乘加操作的过程中没有加数与被加数,只有 2 个部分积.但部分积无符号位(最高位位无意义),所以乘加操作无法通过提前扩充加数与被加数符号位判溢出.所以我们引入了乘加运算溢出补偿技术.即提前判断运算结果,补偿至与判溢出等同,该补偿位设为  $n_1$ .如图 7 所示.

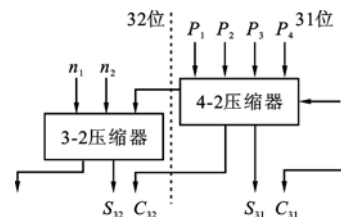


图 7 预判溢出硬件逻辑

由于  $A * B$  的结果与  $C * D$  的结果均为有符号数,所以可以用其符号预判  $A * B$  的结果的符号.设  $S_{A*B}$  为  $A * B$  结果的符号位,  $S_A, S_B$  为  $A, B$  的符号位,则:

$$S_{A*B} = S_A \wedge S_B \quad (1)$$

设  $A * B$  部分积压缩最后一步产生的 2 个部分积为  $p_1, p_2, C * D$  部分积压缩最后一步产生的 2 个

部分积为  $p_3, p_4$ . 在执行 16 位乘法时:

$$(A * B + C * D)[31:0] = (p_1 + p_2 + p_3 + p_4)[31:0] \quad (2)$$

当单独执行乘法  $A * B$  时,  $p_1$  加  $p_2$  时会产生第 32 位的进位, 设为  $m$ , 则:

$$A * B = p_1 + p_2 - m_{A*B} * 2^{32} \quad (3)$$

为了判溢出, 需要将  $A * B$  和  $C * D$  的符号位补到第 32 位, 则最终结果带入式(3)可得

$$\{S_{A*B}, A * B\} = (S_{A*B} - m_{A*B}) * 2^{32} + p_1 + p_2 \quad (4)$$

如图 8 所示, 由于  $P_1, P_2$  第 31 位与  $S_{A*B}$  都已知, 根据真值表得出  $m_{A*B} = P_1 \cdot !P_2 + !P_1 \cdot S_{A*B} + P_2 \cdot !S_{A*B}$ . 进而可得出  $S_{A*B} - m_{A*B}$ . 将  $S_{A*B} - m_{A*B}$  补偿到  $p_1 + p_2$  的第 32 位即等价于将  $A * B$  的符号位补到第 32 位.

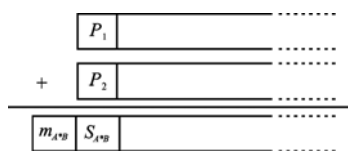


图 8  $m$  与  $P, S$  关系

即

$$\begin{aligned} n &= S_{A*B} - m_{A*B} = S_{A*B} + !m_{A*B} \\ &= !(S_{A*B} \cdot P_1 \cdot P_2 + !S_{A*B} \cdot !P_1 \cdot !P_2) \end{aligned}$$

其中生成  $n$  的硬件映射, 如图 9 所示.

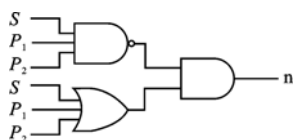


图 9  $n$  的生成逻辑

### 3 流水线实现

现代 DSP 的流水线, 一般如图 10 所示. 分为取指, 译码/取数, 执行和写回段.

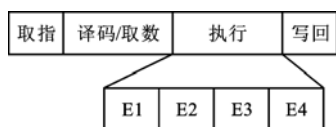


图 10 流水线

本文中所示乘法单元使用 TSMC 45 nm 工艺库, 执行段分为 4 栈的情况下, 最高可达到 1 GHz. 其中 E1 栈包含取数, E4 栈写回.

E1: 从寄存器取操作数  $A, B$ , 并将操作数根据指令种类, 操作长度分发至 16 个操作单元和 ES 编码器. 操作单元首先按修改过的 Booth 编码产生 8 个部分积, 经过第 1 级 4-2 压缩后减为 4 个部分积.

ES 编码后生成 2 组部分积. 共 6 组部分积传递至 E2 栈.

E2 栈: 将操作单元产生的 4 个部分积 4-2 压缩至 2 个, 与 ES 带入部分积压缩阵列 2. 压缩至 2 个部分积.

E3 栈: 将两个部分积发入 CLA 加法器, 得到最终结果. 并进行操作数例外处理, 一处判断和结果调整压缩.

E4 栈: 结果处理和写回.

### 4 综合结果比较

综上所述, 我们的乘法单元实现了 1 个支持多功能的乘加器, 该乘法阵列能够同时执行 1 个  $64 \times 64$ , 4 个  $32 \times 32$  或 16 个  $16 \times 16$  有符号/无符号乘法运算. 我们将其与单个的 64/32/16 位有/无符号乘法生成压缩后的部分积的延时与面积作比较, 与文献[4]比较结果如图 11 所示. 综合环境为 TM-SC45 nm 工艺, 综合工具为 Design Compiler.

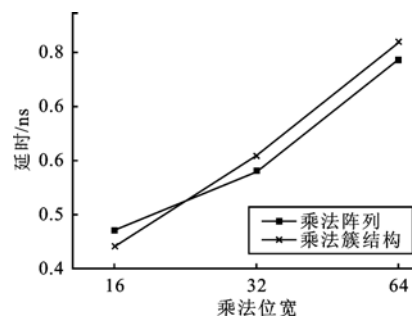


图 11 延时比较

我们将本文所述结构在不同位宽下工作的延时与乘法簇的结构相比. 高位宽乘法降低延时较大. 这是因为乘法阵列结构 64 位乘法的关键路径为 5 个 4-2 压缩器, 而乘法簇结构 64 位乘法的关键路径为 1 个 5-2 压缩器, 3 个 4-2 压缩器, 2 个 3-2 压缩器. 简单分析如下, 单个 5-2 压缩器比 4-2 压缩器延时长, 而 2 个 3-2 压缩器比 1 个 4-2 压缩器延时长, 因此对与 64 位乘法, 该算法比乘法簇结构延时获得较大提升. 在低位宽乘法运算下 (如 16 bit), 延时开销较大. 对于乘法阵列整体来说, 关键路径位于高位宽乘法上. 所以算法对于低位宽乘法性能上的牺牲是十分有效.

图 12 中将本文所述乘法阵列中的各种压缩器的数量与文献[4]中乘法簇结构比较. 表 2 中列出了综合的面积和功耗结果.

本文所述结构与由上表可见中乘法簇结构相比面积降低 3.92%, 功耗提高 5.71%.

表2 综合的面积和功耗结果

|       | 面积/ $\mu\text{m}^2$ | 功耗/mW    |
|-------|---------------------|----------|
| 乘法阵列  | 65 091              | 28.730 1 |
| 乘法簇结构 | 68 198              | 27.177 3 |

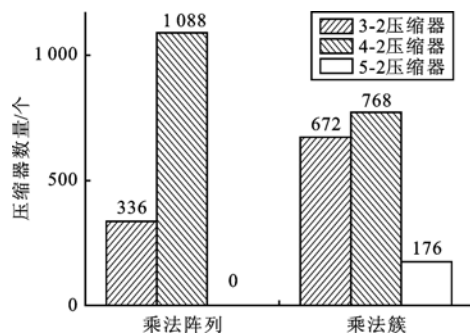


图12 压缩器数量

## 5 结束语

本文展示了一个高性能低面积高宽度 SIMD 乘法阵列体系结构,该阵列能够同时执行 1 个  $64 \times 64$ , 4 个  $32 \times 32$  或 16 个  $16 \times 16$  有符号/无符号乘法运算. 该运算阵列在较少的延时消耗和面积开销下,极大的扩充了乘法器的功能. 与乘法簇结构相比具有延时低,面积低的特点.

## 参考文献:

- [1] Albert Danysh, Dimitri Tan. Architecture and imple-

mentation of a vector/SIMD multiply-accumulate unit [J]. IEEE Transactions on Computers, 2005, 54(3): 284-293.

- [2] Abdelgawad A, Magdy Bayoumi. High speed and area-efficient multiply accumulate (MAC) unit for digital signal processing applications [C]//Circuits and Systems, ISCAS 2007, New Orleans, 2007: 3199-3202.
- [3] Yeh Wenchang, Jen Cheinwei. High-speed booth encoded parallel multiplier design [J]. Transactions on Computers, 2000, 49(7): 692-701.
- [4] Timothy Anderson. A 1.5 Ghz VLIW DSP CPU with integrated floating point and fixed point instructions in 40 nm CMOS [C]// IEEE 20th Symposium on Computer Arithmetic, Germany, Tubingen, 2011.

## 作者简介:

吴虎成 男, (1970—), 硕士研究生, 副研员. 研究方向为 VLSI 设计.



刘洋徐瑞 男, (1991—), 博士研究生. 研究方向为高性能微处理器设计.

刘建平 男, (1986—), 硕士研究生. 研究方向为高性能微处理器设计.

(上接第8页)

- [10] Chen S B, Donoho D L, Saunders M A. Atomic decomposition by basis pursuit [J]. Society for Industrial and Applied Mathematics, 2001, 43(1): 129-159.
- [11] 李民. 基于稀疏表示的超分辨率重建和图像修复研究 [D]. 成都: 电子科技大学, 2011.
- [12] 赵佳佳, 唐峥远, 杨杰, 等. 基于图像稀疏表示的红外小目标检测算法 [J]. 红外与毫米波学报, 2011, 30(3): 156-166.

- [13] Engan K, Aase S O, Husey J H. Method of optimal-directions or frame design [C]//IEEE International Conference on Acoustics, Speech and Signal Processing. USA: IEEE Computer Society, 1999: 2443-2446.

## 作者简介:

黄晓生 男, (1972—), 博士, 副教授. 研究方向为图形图像处理、多智能体理论与技术.